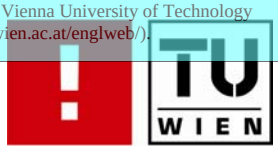
Die approbierte Originalversion dieser Diplom-/Masterarbeit ist an der Hauptbibliothek der Technischen Universität Wien aufgestellt (http://www.ub.tuwien.ac.at).

The approved original version of this diploma or master thesis is available at the main library of the Vienna University of Technology (http://www.ub.tuwien.ac.at/englweb/).



FAKULTÄT FÜR !NFORMATIK

Benchmarking of Middleware Systems

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Informatik

eingereicht von

Bernhard Löwenstein
Matrikelnummer 9726426

an der
Fakultät für Informatik der Technischen Universität Wien

Betreuung:
Betreuerin: Ao.Univ.Prof. Dipl.-Ing. Dr. eva Kühn
Mitwirkung: Univ.Ass. Dipl.-Ing. Mag. Richard Mordinyi

Wien, 28.10.2008 ....
(Unterschrift Verfasser) (Unterschrift Betreuerin)

Abstrakt

In der heutigen IT-Welt sind viele unterschiedliche Middlewaresysteme verfügbar. Oftmals verkünden die Anbieter solcher Anwendungen, dass ihr Produkt die beste Performance und Skalierbarkeit aufweist, und überhaupt das Beste von allen sei.

Diese Arbeit veranschaulicht, wie man die Performance und Skalierbarkeit unterschiedlicher Middlewaresysteme auf realistische Art und Weise messen und vergleichen kann. Die Benchmarks dazu werden anhand einiger maßgeschneiderter Testfälle durchgeführt.

Der erste Teil der Arbeit gibt eine ausführliche Einführung in die Begriffe Benchmark, Performance, Skalierbarkeit und Middleware und präsentiert drei verschiedene Middlewarekonzepte: eXtensible Virtual Shared Memory (XVSM), JavaSpaces und Java 2 Plattform, Enterprise Edition (J2EE).

Im zweiten Teil werden einige Benchmarkszenarien für die zugehörigen Systeme MozartSpaces, GigaSpaces eXtreme Application Platform (XAP) und JBoss Application Server (AS) definiert und durchgeführt. Die Ergebnisse ihrer Ausführung werden anschließend visualisiert und interpretiert.

Abstract

In today's IT world many different middleware systems are available. Often, the vendors of such applications proclaim that their product offers the best performance and scalability and actually is the best of all.

This work demonstrates how to evaluate and compare the performance and scalability of different middleware systems in a realistic manner. The benchmarks for this purpose are implemented via several customized test cases.

The first part of the work gives a detailed introduction into the terms benchmark, performance, scalability and middleware and presents three different middleware concepts: eXtensible Virtual Shared Memory (XVSM), JavaSpaces and Java 2 Plattform, Enterprise Edition (J2EE).

In the second part, several benchmark scenarios are defined and implemented for the associated systems MozartSpaces, GigaSpaces eXtreme Application Platform (XAP) and JBoss Application Server (AS). The results of their execution are visualized and interpreted thereafter.

Table of Contents

Table of Abbreviations

API	Application Programming Interface
AS	Application Server
CMP	Container Managed Persistence
CPU	Central Processing Unit
EJB	Enterprise JavaBeans
FIFO	First In, First Out
GB	Gigabyte
GHz	Gigahertz
IT	Information Technology
J2EE	Java 2 Plattform, Enterprise Edition
JMS	Java Message Service
JMX	Java Management Extensions
JVM	Java Virtual Machine
LIFO	Last In, First Out
MB	Megabyte
MBit/s	Megabit per Second
OS	Operating System
RAM	Random Access Memory
RMI	Remote Method Invocation
RPC	Remote Procedure Call
SBC	Space Based Computing
TPM	Transaction Processing Monitor
TSC	Triple Space Communication
VSM	Virtual Shared Memory
XAP	eXtreme Application Plattform
XVSM	eXtensible Virtual Shared Memory

Table of Figures

1 Introduction

In the context of this thesis the performance and the scalability of different middleware systems, namely MozartSpaces, GigaSpaces XAP and JBoss AS, will be measured and compared among each other. For doing so, a new approach will be considered and evolved, because so far there is no method available for comparing these systems or only parts of them. The aim of this work is to get a feeling of the strengths and weaknesses of these technologies and to enable the identification of the systems' room for improvement.

In chapter 3, definitions for benchmark, performance, scalability and middleware will be given and topics associated with these terms will be discussed. Since the concepts of performance and scalability are often not clearly distinguished in the literature, this work will offer a precise differentiation between these terms. The diverse middleware concepts of XVSM, JavaSpaces and J2EE will also be introduced as well as their associated systems.

Another important task is finding metrics as simple as possible, which allow the measuring of the performance and the scalability of a middleware system. Such a metric can be seen as a function, to which the test case and the benchmarked middleware system are handed over and that returns an objective indicator after the execution of the test. Comparing different products among each other is always only possible on the basis of such a measured value.

Furthermore, the common denominator of all benchmarked systems must be found, on the basis of which the benchmarks can be executed. Due to the differences between the investigated middleware concepts, only properties can be benchmarked which are common in all technologies. Indeed, the selection of such a high-level approach is suboptimal, but there is no alternative that will make such different concepts comparable.

Tools, which support the implementation of the benchmarks, will also be introduced and analysed regarding their usability. They may simplify the development and also the execution of the individual scenarios.

In chapter 4, the implementation of the benchmarks will be prepared, the designed scenarios will be executed and the results will be visualized and interpreted. Fulfilling these points will be the lion's share of this work.

It is of particular importance for this work that the selected test cases will be significant and will cover the established use cases. To make the benchmarks traceable, a precise definition of the scenarios will be required. And not only the test cases, but also the used middleware configuration and used test environment must be described accurately.

In addition to the selection of adequate test cases, it will also be relevant that the ideal environment is located, in which the benchmarks are executed. In the majority of cases middleware is applied in a distributed environment, and that must be taken into account in the design of the benchmarks. So, an adequate test environment, namely concerning hardware and software, must be set up.

For the real testing of performance and scalability, a software application will be designed and implemented, which encapsulates the test cases and enables the execution of the devised scenarios. This benchmark tool will only use the public and standardized interfaces of the different middleware technologies, namely XVSM (MozartSpaces), JavaSpaces (GigaSpaces XAP) and J2EE (JBoss AS). The advantage of this procedure is that it also allows the testing of other middleware systems that are based on these concepts.

Another important thing is to make the execution of the benchmarks fast and simply repeatable, so that they can be performed after each source modification by the developers. Testing the performance and scalability of a system should not be a unique thing, but go with the whole process. So it is desirable that a way for the automatic execution of the benchmarks can be found.

Of particular significance will also be that the test results are clearly represented and that the relevant information can be obvious at a glance. Finding a tool, which visualizes the measured data and generates the charts automatically, would be ideal, because this saves work and time and simplifies handling.

In chapter 5, the achieved knowledge will be transferred to the development of TripCom. This is a middleware system, which is currently in development and follows a new approach, namely the Triple Space Communication (TSC). Some lectures learned by the implementation of the benchmarks may be interesting for the implementation of this system, too.

2 Related Work

This chapter gives a survey of papers that deal with issues similar to the topics discussed in this work.

A procedure for measuring TripCom's performance is presented in [1]. It uses a performance model for evaluating the system's performance and informs also about the problems using this approach. The approach is presented in chapter 3.2.4 of this work.

[2], [3], [4], [5], [6] and [7] give definitions for the term scalability and discuss related topics. They allow a clear differentiation between performance and scalability and can be looked up in chapter 3.3.1.

Different classifications are introduced in [2], [5], [6] and [8]. Chapter 3.3.2 gives an overview of these diverse categorizations.

Miscellaneous metrics for measuring the scalability of a system are presented in [7], [9] and [10]. Of particular importance is the metric in [9] which is based on the calculation of the efficiency value and is presented in chapter 3.3.3.1. It provides the basis for the advanced metric used later in this work. The metric in [10] rates the productivity of a system and evaluates the system's scalability on basis of this value. Chapter 3.3.3.2 introduces this metric to the reader. [7] even goes a step further and defines a scalability framework which can be used for benchmarking a system's scalability. This metric is given in chapter 3.3.3.3 of this work.

[6], [11] and [12] provide recommended procedures for measuring a system's scalability. The former work proposes seven steps – starting with the identification of the critical use cases and ending with the presentation of the results – for such measurements. The entire approach is presented to the reader in chapter 3.3.6.

An insight into middleware and definitions for this term are provided in [13], [14] and [15]. These papers introduce the reader to the concept of middleware which is defined as the software layer between the operating system (OS) and the applications. Chapter 3.4.1 of this work summarizes the core information.

[16] and [17] present different classifications for middleware which can be looked up in chapter 3.4.3.

The concepts of XVSM, JavaSpaces and J2EE and their belonging systems MozartSpaces, GigaSpaces XAP and JBoss AS are described on different websites and papers with various focuses. So, finding an adequate source for its needs should easily be possible for the interested reader. Corresponding resources are listed below at the references, if necessary.

[18] discusses approaches, results and experiences in connection with middleware benchmarking and is a key work in this field. It identifies the incomparableness of the benchmark results as one of the largest problems in this field of study. That difficulty results from the fact that most developers use their own test suite for evaluating their system's performance and scalability. All attempts of standardization failed because of the diversity of available middleware systems. Furthermore, the paper distinguishes

between benchmarks that give support in the design phase and benchmarks that evaluate the performance and scalability of a middleware system. As future prospect the establishment of a common knowledge base is formulated.

[19] deals with performance tests concerning the Tuple Space technology. The paper presents a framework called Space bEnchmarking and TesTing moduLEs (SETTLE) which supports the execution of Tuple Space benchmarks. Moreover, it describes and demonstrates a method for measuring the throughput and response time of a local JavaSpace which is concurrently accessed. Also, an approach for identifying the worst-case performance of a JavaSpace is shown.

The focus of [20] is on the performance and scalability of Enterprise JavaBeans (EJB). It compares five diverse EJB implementations – from stateless session beans to the session façade pattern using remote/local interfaces – of the same project and measures the throughput with two different EJB containers, namely JBoss and JOnAS, thereof. The work shows that the performance and scalability of a J2EE application depends both on the concrete implementation as well as on the EJB container.

The activities presented at [21] and [22] should also not be forgotten. The first work, SPECjAppServer2004, allows that an application server based on the J2EE platform can be subjected to a standardized benchmark. The relevant components of such an application server are tested thoroughly and their performance is evaluated. The second work, SPECjbb2005, rates the entire server – from hardware over operating system to Java environment – with regard to performance, partly scalability. However, both are inappropriate as tools for the implementation and execution of individual test cases.

Finally, the ACM Workshop on Software and Performance (WOSP) and the Computer Measurement Group (CMG), which are active in this scope, too, should also be mentioned with its journals and proceedings published at [23] and [24].

3 Terms and Definitions

The following chapter provides different definitions of benchmark, performance, scalability and middleware. It also explains topics associated with these terms.

3.1 Benchmark

Basically, a benchmark enables the comparison of two or more systems, perhaps only parts of them, on basis of objective indicators. The term itself comes from the economy.

3.1.1 Definitions

A general and an IT-specific definition should be sufficient to gain an understanding of the term benchmark.

At [25] the following general definition of benchmark can be found:

- *A standard by which something is evaluated or measured.*
- *A surveyor's mark made on some stationary object and shown on a map; used as a reference point.*

In connection with computer science benchmark is described at [26] as follows:

> *In computing, a benchmark is the act of running a computer program, a set of programs, or other operations, in order to assess the relative performance of an object, normally by running a number of standard tests and trials against it. The term, benchmark, is also mostly utilized for the purposes of elaboratly-designed benchmarking programs themselves. Benchmarking is usually associated with assessing performance characteristics of computer hardware, for example, the floating point operation performance of a CPU, but there are circumstances when the technique is also applicable to software. Software benchmarks are, for example, run against compilers or database management systems.*
>
> *Benchmarks provide a method of comparing the performance of various subsystems across different chip/system architectures.*

3.1.2 Tools

Below, there is a summary of free available tools which assist in the implementation and the execution of benchmarks. Most of them are written in Java. All tools have been tested with regard to functionality and application for this work.

3.1.2.1 Apache JMeter

Apache JMeter released at [27] enables the organization and execution of distributed load tests on a server, network or object. The target system, that is to be examined, is requested by several slaves that are distributed over different computers. The slaves can be controlled by the testing person via a central master, which has a graphical user interface available. For the communication between the master and its slaves Remote Method Invocation (RMI) is used.

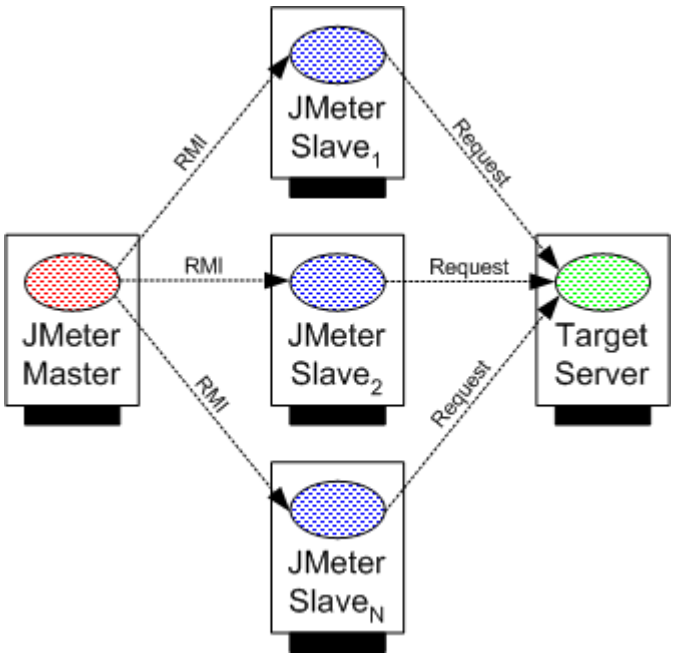


Figure 1: Functionality of Apache JMeter

Even if Apache JMeter provides an interesting and sophisticated environment for the execution of distributed load tests, it will not be used. The reason is that running the benchmarks without the use of any tool will be simpler and above all sufficient in this work.

3.1.2.2 Faban

Faban released at [28] supports in the development and running of benchmarks. It consists of a harness and the driver framework. The former allows the automatic execution of test cases which were generated using the associated framework. Via this driver framework the lifecycle of an implemented benchmark can be controlled. The test results are given by throughput and response time and can be visualized over the whole test period in form of charts.

Benchmarks which are supposed to run with Faban must be implemented on the basis of the driver framework. This facilitates the implementation after a certain induction phase, but also commits fully to this system. Since the effort for the familiarization with this tool is higher than the benefits of its use, it will not be used for the implementation of the benchmarks.

3.1.2.3 Gnuplot

Gnuplot released at [29] allows the graphical representation of functions and data via script and command line. It generates two- or three-dimensional plots that can be saved in various formats. The tool was originally written for Unix, but meanwhile it is available for many other platforms.

Below is an example of a Gnuplot script which uses data points from several files for the chart generation:

```
#!/gnuplot

set terminal gif
set output "chart.gif"
```

```
set size 2/3., 2/3.
set grid
set key left top box lw 2
set xlabel "Write Operations"
set ylabel "Execution Time [ms]"

plot 'fifo.dat' t " FIFO" with linespoints,\
     'key.dat' t " KEY" with linespoints,\
     'lifo.dat' t " LIFO" with linespoints,\
     'linda.dat' t " LINDA" with linespoints,\
     'random.dat' t " RANDOM" with linespoints,\
     'vector.dat' t " VECTOR" with linespoints
```

The generated diagram for this Gnuplot script and the transferred data is displayed in the following:

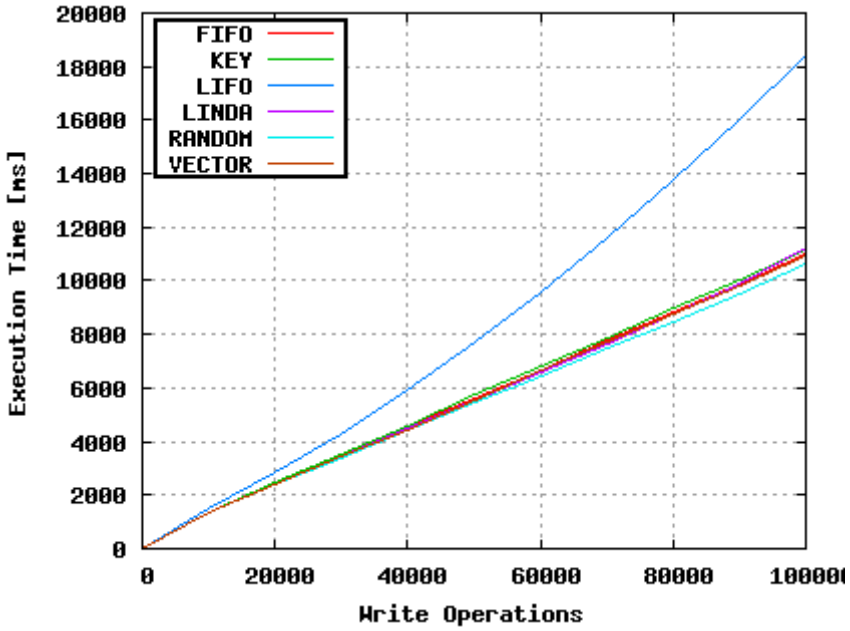


Figure 2: Example of Gnuplot Diagram

It should be obvious that Gnuplot is not really a benchmark tool, but it is useful for the automated generation of charts. In this work it will be used for exactly that purpose.

3.1.2.4 JAMon

JAMon released at [30] is an application programming interface (API) which provides methods for monitoring the runtime behaviour of Java applications. By adding a few additional lines of code, several statistical data can be requested and displayed.

A handicap of this tool is that for the export of statistical data no methods are finished, yet. But the main disadvantage of JAMon is that milliseconds instead of nanoseconds are used for time resolution. Since tests have been demonstrated that this is not sufficient, this tool cannot be used in this work.

3.1.2.5 JBento

JBento released at [31] is a toolkit that supports benchmarking a system's performance and provides methods for measuring, analysing results and converting results into charts.

The description of JBento sounds promising. However, since the entire documentation is available currently only in Japanese language, using this tool is unthinkable at present.

3.1.2.6 jMonit

jMonit released at [32] represents a monitoring toolkit that works similar to JAMon. The architecture of jMonit is made up of three layers, namely the instrumentation, computation and formatting layer. Each layer has its own function.

In contrast to JAMon, useful methods for the export of the collected data are available and nanoseconds are used for time resolution. Nevertheless, the benefits resulting from the usage of this toolkit are ultimately too low.

3.1.2.7 JPerf

JPerf released at [33] is based on the JUnit framework and allows the evaluation of performance and scalability of a system. It uses threads for simulating concurrent access and returns the throughput.

The use of a narrow interface makes JPerf easy to integrate and universally applicable. However, the JUnit structure does not fit for the purpose of this work, which is why it will not be used for the implementation of the benchmarks.

3.1.2.8 JUnitPerf

JUnitPerf released at [34] allows the user to define limits for the execution time of existing JUnit test cases. The compliance with these bounds can be tested in an automated way. Even the behaviour under load – a specified number of competing users execute the same test case simultaneously – can be tested using JUnitPerf.

This tool is well-suited to ensure a priori defined performance and scalability requirements. However, for exclusive measurement of these values as demanded in context with this work, it is inappropriate.

3.1.2.9 Loadunit

Loadunit released at [35] also works on basis of JUnit test cases. It is generally more powerful than JUnitPerf. For instance, it provides such features like the generation of realistic test data and graphical representation of throughput versus users and performance versus users.

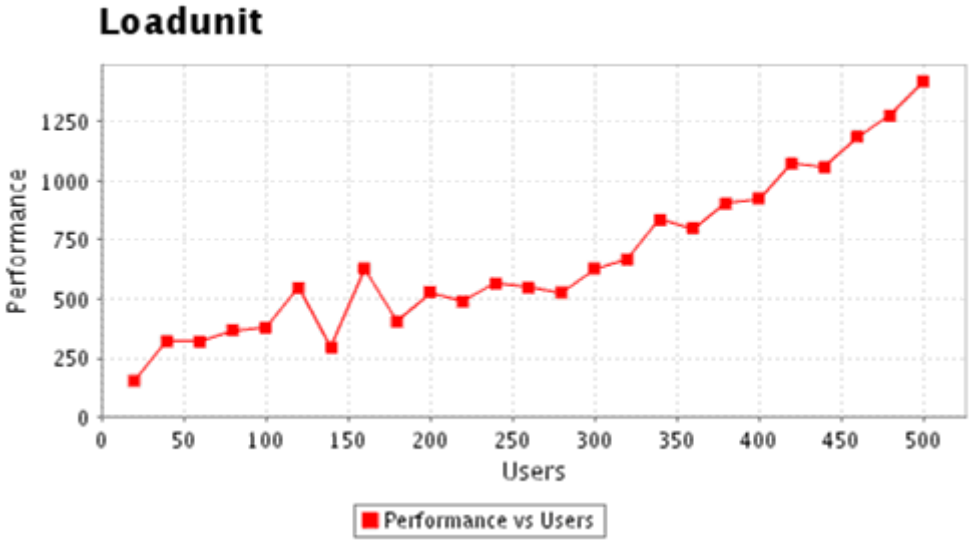


Figure 3: Example of Loadunit Diagram

As for the two benchmark tools before, which were also based on JUnit, the structure of Loadunit prevents its use in this work.

3.1.2.10 p-unit

p-unit released at [36] corresponds approximately to the structure and scope of services with Loadunit.

Therefore, it should be clear why this tool is also not applied for the implementation of the benchmarks.

3.2 Performance

Performance itself is always associated with the execution of a particular activity in a certain period of time. The technical term performance comes from physics, where it is called power.

3.2.1 Definitions

The physical definition at [37] already provides a good approximation for the term in computer science:

> *In physics, power is the rate at which work is performed or energy is transmitted, or the amount of energy required or expended for a given unit of time. As a rate of change of work done or the energy of a subsystem, power is:*
>
> *P = W / t*
>
> *where P is power, W is work and t is time.*

A definition of performance with regard to computer science can be found at [38]:

> *Computer performance is characterized by the amount of useful work accomplished by a computer system compared to the time and resources used.*
>
> *Depending on the context, good computer performance may involve one or more of the following:*

- *Short response time for a given piece of work*
- *High throughput (rate of processing work)*
- *Low utilization of computing resource(s)*
- *High availability of the computing system or application*

For the purpose of limitation, in this work the term performance is always used in connection with speed. That means it is only relevant how quickly a task is finished.

3.2.2 Metrics

For evaluating the performance of a system, or only a part of it, two quite simple metrics are sufficient:

- Measure the time, how long the execution of a specified number of actions will take. As an objective indicator for all compared systems a duration, which is called execution time, is delivered. Under certain conditions it is referred to as response time.
- Measure the number of actions that can be performed within a specified time. As an objective indicator for all compared systems a number is delivered, which is usually referred to as throughput.

3.2.3 Criteria

The criteria regarding performance are always dependent on the specific investigation and cannot be determined a priori.

In connection with middleware systems simple operations such as reading and writing of a large number of records for different system configurations (with or without caching, with or without replication, with or without transactions, with or without distribution) are essential.

[2] even goes a step further and limits the evaluation to the number of transactions per second or the number of transferred bytes per second.

3.2.4 Procedure

For the evaluation of TripCom's performance [1] proposes the following seven steps, which may be also suitable for the assessment of other middleware systems:

1. Define component performance models
2. Review of component performance models
3. Define system performance model
4. Review of system performance model
5. Analysis of interdependences between component's and system performance
6. Build performance functions
7. Analyse results with respect to performance definition used, i.e. determine performance

A problem of this approach is that for realistic results fine-granular and complex models are necessary, whose evaluation can be often very time-consuming. For this reason the paper recommends the implementation of real experiments – the execution of the components in a representative operating environment with simultaneous time measurement – instead of elaborate simulation modelling.

3.3 Scalability

The behavior of a system, possibly even a part of it, in connection with an increasing problem size is basically for scalability. The term is an IT-specific one.

3.3.1 Definitions

There is no precise definition for the term scalability. Therefore, several explanations are quoted to enable an orientation.

At [39] the following definition of scalability can be found:

> *In telecommunications and software engineering, scalability is a desirable property of a system, a network, or a process, which indicates its ability to either handle growing amounts of work in a graceful manner, or to be readily enlarged. For example, it can refer to the capability of a system to increase total throughput under an increased load when resources (typically hardware) are added.*
>
> *Scalability, as a property of systems, is generally difficult to define and in any particular case it is necessary to define the specific requirements for scalability on those dimensions which are deemed important. It is a highly significant issue in electronics systems, database, routers, and networking. A system whose performance improves after adding hardware, proportionally to the capacity added, is said to be a scalable system. An algorithm, design, networking protocol, program, or other system is said to scale if it is suitably efficient and practical when applied to large situations (e.g. a large input data set or large number of participating nodes in the case of a distributed system). If the design fails when the quantity increases then it does not scale.*

This means that scalability deals with the behaviour of a system while increasing the computational requirements and the available resources at the same time. A system, which offers optimal scalability, provides the same performance for the n-fold problem size and the n-fold resources as the reference system.

An explanation of the term in connection with performance provides [3]:

> *We define performance as:*
>
> - *The response time seen by a user under normal working conditions.*
> - *The cost of the system per user; a representation of the hardware requirements of the system.*
>
> *We consider scalability to be a measure of how these change as the system size is increased – for example, by adding more users.*

No further definition, but a clear differentiation to the term performance can be found in [4]:

Scalability is frequently associated with performance and, at times, the terms are erroneously used interchangeably. Performance, in our opinion, can be an indicator of scalability only when the stakeholder's interests are performance indexes, such as throughput and execution time. Otherwise, performance is simply another requirement to be met by the system, like message reliability, memory usage and other metrics associated with software qualities. For this reason, we believe that the scalability of a system should be seen in the context of its requirements.

Scalability in relation to distributed systems is explained in [5]. Therefore, the ability of good scaling must be taken into account in the design phase already:

A system is said to be scalable if it can handle the addition of users and resources without suffering a noticeable loss of performance or increase in administrative complexity. Scale has three components: the number of users and objects that are part of the system, the distance between the farthest nodes in the system, and the number of organizations that exert administrative control over pieces of the system.

If a system is expected to grow, its ability to scale must be considered when the system is designed. Naming, authentication, authorization, accounting, communication, and the use of remote resources are all affected by scale. Scale also affects the user's ability to easily interact with the system.

Additional understanding can be achieved referring to the explanation of the term in [2], which allows the adjustment of a system for rising problem size only to a limited extent:

Scalability is the capacity to address additional users or transactions by adding resources without fundamentally altering the implementation architecture or implementation design.

That scalability is a quality factor of today's systems is shown to the reader in [6]:

Scalability is one of the most important qualities of today's software applications. As businesses grow, the systems that support their functions also need to grow to support more users, process more data, or both. As they grow, it is important to maintain their performance (responsiveness or throughput). Poor performance in these applications often translates into substantial costs.

[7] deals with the relevance of scalability in the development of distributed systems. The importance of scalability is justified in consequence of the frequent change of the environment in which such a system runs:

A major problem in the development of these distributed information systems, is that we cannot assume that the environment in which the system is to operate will remain the same over time. This means that developers must take into account that the system should be easy to adapt to meet requirements that are unknown during the development process. At best, such unknown requirements are formulated imprecisely. One particularly fuzzy requirement is that the information system should be scalable.

Based on these explanations, the reader should be able to evolve an idea for the term scalability. It is important to understand that there is indeed a significant connection between performance and scalability, but each of these terms stands for something different.

3.3.2 Classifications

There are different approaches to classify scalability. Some of them are listed below.

3.3.2.1 Numerical / Geographical / Administrative

A classification by numerical, geographic and administrative aspects is given in [5]:

- Numerical dimension: This category deals with the number of users, objects and services which participate in the system.
- Geographical dimension: This category gives attention to the spatial distribution of the system.
- Administrative dimension: This category concerns with the number of organizations that control the system or parts of it.

Each of these categories brings along its own problems and has different effects on the individual components of a distributed system. To get a grip on the difficulties the techniques of replication, distribution and caching are introduced. In addition to these three concepts, which make the construction of scalable distributed systems possible in the first place, best practices are shown.

3.3.2.2 Linear / Sub-linear / Super-linear

In [6] a classification is done according to how the behaviour of the capacity changes by adding additional processors compared to only one. Depending on that, the terms of linear, sub-linear and super-linear scalability are distinguished:

- Linear scalability: Increasing the number of processors to n results in a throughput that is n times the capacity with one processor. From this it follows that doubling the number of processors will double the system's throughput.
- Sub-linear scalability: Increasing the number of processors to n results in a throughput that is less than n times the capacity with one processor. From this it follows that doubling the number of processors will not double the system's throughput.
- Super-linear scalability: Increasing the number of processors to n results in a throughput that is more than n times the capacity with one processor. From this it follows that doubling the number of processors will more than double the system's throughput.

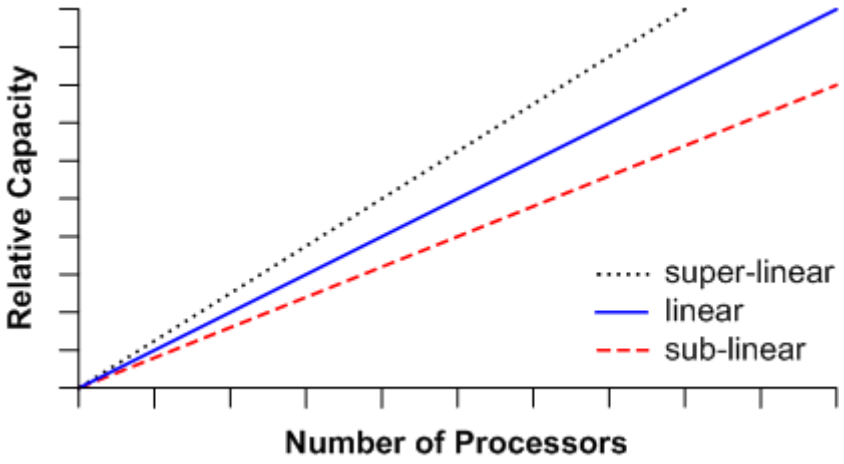


Figure 4: Linear, Sub-linear and Super-linear Scalability

It should still be noted that in the context of distributed systems the multiplication of system resources should not only be restricted to processors. Choosing a more general approach is useful.

3.3.2.3 Vertical / Horizontal

[2] distinguishes between two different scaling methods for the addition of resources. One possibility is providing more resources in one server machine, the other is making more server machines available. Depending on that, the used scaling method is called vertical or horizontal:

- Vertical scaling: The system's performance is improved by adding more hardware resources on a single server. This method is commonly referred to as scaling up.
- Horizontal scaling: The system's performance is improved by making additional servers available. This method is commonly referred to as scaling out.

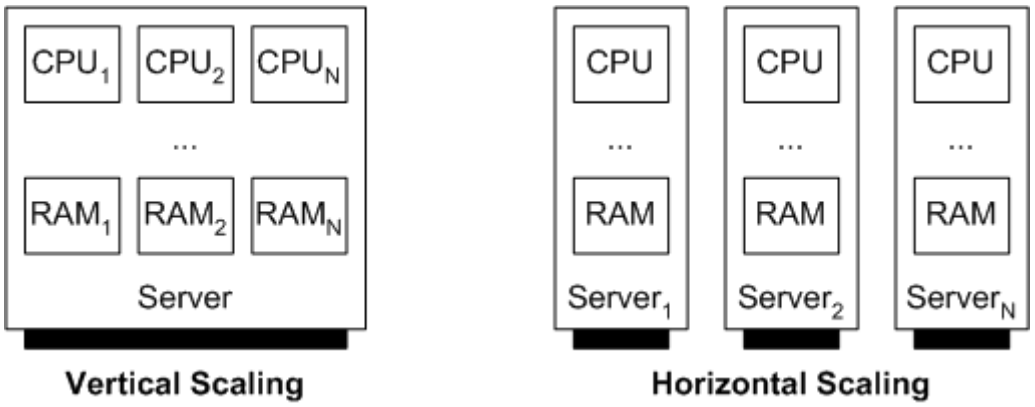


Figure 5: Vertical and Horizontal Scaling

It is important to put on record that the use of these two procedures with the same application leads to different results. Furthermore, the elimination of problematic positions can raise new bottlenecks. Both statements, each with an example, are demonstrated in [40].

3.3.2.4 Load / Space / Space-time / Structural / Distance / Speed/Distance

Another approach for the classification of scalability can be found in [8]. Thereby, each system and each system component features several attributes at the same time:

- Load scalability: A system is referred to as being load scalable if it functions well under light, moderate and heavy load through efficient utilization of the available resources.
- Space scalability: A system or application is referred to as being space scalable if increasing the number of objects does not result in oversized growth of its memory requirements.
- Space-time scalability: A system is referred to as being space-time scalable if the algorithms and data structures which were used for the implementation function well, independent of the problem size.
- Structural scalability: A system is referred to as being structurally scalable if its implementations and standards do not limit the systems functionality in case of increasing objects.

For systems, whose nodes are distributed over long distances, two further scaling methods are distinguished:

- Distance scalability: An algorithm or protocol is referred to as being distance scalable if it functions well for both short and long distances.
- Speed/Distance scalability: An algorithm or protocol is referred to as being speed/distance scalable if it functions well for both short and long distances, with low and high speeds.

According to this, the algorithms and protocols are characterized by their changing of behaviour as a result of different distances and speeds.

3.3.3 Metrics

For making the scalability of a system measurable and comparable, it requires a metric. Some of these methods are introduced below.

3.3.3.1 Speedup → Efficiency

A simple metric to rate the scalability of a system is presented in [9]:

Let time(n, x) be the time that a machine with n processors needs to execute a program with problem size x.

If such a program is executed on a machine with n processors, then the speedup in comparison to a machine with only one processor can be formally expressed as follows:

$$speedup(n, x) = time(1, x) / time(n, x)$$

The ratio between the speedup and the number of processors returns a value for the efficiency. In formulas that relation can be expressed as follows:

$$efficiency(n, x) = speedup(n, x) / n = time(1, x) / (time(n, x) \cdot n)$$

According to this metric, a system is called scalable, if and only if for all algorithms, any number of processors n and any problem size x efficiency(n, x) = 1 holds.

If this strict restriction, that hardly permits scalable systems, is toned down, then this method provides a simple way to estimate the scalability of a system and to make different systems comparable among each other.

3.3.3.2 Throughput / Response Time / Costs → Productivity

[10] describes why the metric above based on efficiency is not sufficient for distributed systems without adaption. Among other things it is argued that the size of a distributed system cannot only be reduced to the number of processors – it is more complex. From this it follows that size in a distributed system becomes a multidimensional concept. The greater variety of communication mechanisms also brings up new problems, which must not be ignored, either.

Now, the new approach includes throughput and response time as productivity factors. The proposed metric is based on the productivity and is formally defined for a scaling factor k as follows:

Let $\lambda(k)$ be the throughput of responses per second, let f(k) be the response time per response on average, which was calculated via quality of service, and let C(k) be the ongoing costs to ensure the required throughput. Then, productivity can be expressed in formulas as follows:

$$F(k) = \lambda(k) \cdot f(k) / C(k)$$

If the calculated productivity values for two different scaling factors k_1 and k_2 are set in relation, then the desired scale value is returned, which informs about the scalability of the system:

$$\psi(k_1, k_2) = F(k_1) / F(k_2)$$

3.3.3.3 Attributes / Resources / Costs → Conditions

[7] provides a formal framework for evaluating the scalability of a distributed application:

According to the underlying model, such an application is made up of several cooperating programs that run on different machines and use a common database. The environment of a distributed application is divided into the following four areas: client, development, administration and execution.

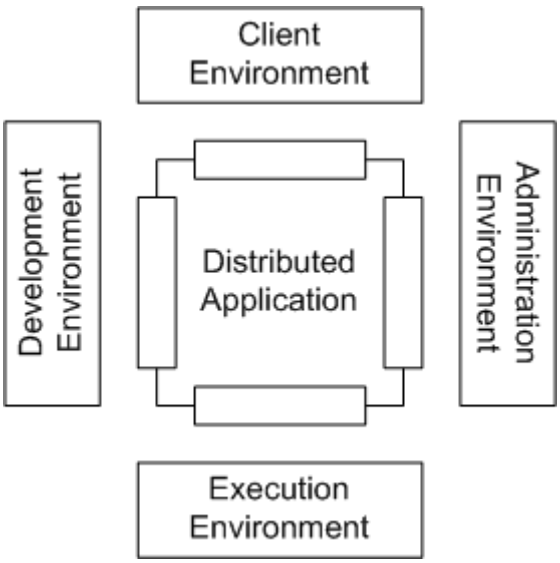


Figure 6: Model of Scalability Framework

Now, different attributes $Attr_1$, ..., $Attr_N$ are associated with the client environment. Then, an instance of the client environment is formally defined by the vector $a = \langle a_1, ..., a_N \rangle$. For example, such attributes can be the number of clients, the number of client requests or the transferred data between the applications.

In analogy to the client environment different resources Res_1, ..., Res_N are associated with the execution environment. Formally, an instance of the execution environment is defined by the vector $r = \langle r_1, ..., r_N \rangle$. For example, the number of a certain resource can be an entry of this vector.

In the next step, the performance function $Perf_A(a, r)$ is defined for the distributed application, which returns a numerical value. A low value represents poor performance and a high value indicates good performance. Also, it is expected that increasing an attribute results in performance degradation and that increasing a resource leads to better performance. Formally, this can be expressed as follows:

$$\Delta Perf_A / \Delta a \leq 0 \text{ and } \Delta Perf_A / \Delta r \geq 0$$

The cost function Cost(r) describes the costs of the resource capacities given by r, and $Cost_A(a, r)$ returns the costs for the realisation of the performance $Perf_A(a, r)$. Furthermore, $r_{min}(a)$ expresses the resources needed to implement $Perf_A(a, r)$ with minimal costs.

Now, let A be a distributed application, whose execution in the reference environment with the attributes a_{ref} and resources $r_{ref} = r_{min}(a_{ref})$ causes minimal costs, and let $a_{ref}\langle i : a \rangle$ be the vector with the i^{th} component. Moreover, let $\delta(a)$ be a function which expresses an accepted performance degradation in relation to the problem size, and let $\gamma(a)$ be a function which limits the maximum costs.

Then, the application A is called scalable, if and only if the following three conditions hold:

1. A can accommodate values $a_{ref}[i] < a \leq a_{max}$

2. $\forall a_{ref}[i] < a \leq a_{max}\ \exists r::\ Perf_A(a_{ref}, r_{ref}) - Perf_A(a_{ref}\langle i : a\rangle, r) \leq \delta(a_{ref}\langle i : a\rangle)$
3. $\forall a_{ref}[i] < a \leq a_{max}::\ Cost_A(a_{ref}\langle i : a\rangle, r_{min}(a_{ref}\langle i : a\rangle)) \leq \gamma(a_{ref}\langle i : a\rangle)$

Based on this formal framework particular metrics can be defined. Depending on the problem size, such a metric tolerates a predetermined decrease of performance and also considers the accumulated costs.

3.3.4 Criteria

[3] analyses a distributed system in regard to performance and scalability. Thereby, the following four important criteria are identified which influence the scalability of a distributed system.

3.3.4.1 System Design

Poor scalability results often from crude system design. Therefore, it is reasonable to always scrutinize and analyse the system design during scalability investigations. Furthermore, it is important that considerations regarding to scalability must already be done in the design phase, because the later a change in design must take place, the larger the effort to do so.

3.3.4.2 Communications

The communication over a network is often the main reason for poor performance of a distributed system. It should be noted that the delivery of large messages takes longer than of short ones and that doubling the number of messages also doubles the demands on the network. Thus, it is necessary to examine whether a network can meet the communication demands if the number of users reaches the maximum.

3.3.4.3 Data Input/Output

The required time for the input/output operations is also not negligible. Therefore, it is recommended to represent and analyse all of these operations. If a problem is diagnosed in the test phase, then eventually the deficit can be addressed faster.

3.3.4.4 Computational Requirements

It is relatively easy to detect problems related to communication and data input/output, but it is difficult to estimate the computational and memory requirements for each specific function. That results from the fact that the development of a function can happen in various ways at a higher level. Above all, it is quite difficult to assess the seriousness of such scaling problems correctly.

3.3.5 Challenges

The building of scalable distributed systems poses several challenges to their designers. [41] returns the following issues for this purpose:

- Controlling the cost of physical resources
- Controlling the performance
- Preventing software resources running out
- Avoiding performance bottlenecks

These aspects are explained in more detail below.

3.3.5.1 Controlling the Cost of Physical Resources

The expansion of a system for achieving the demands should always be feasible with reasonable costs. In general, for a scalable system with n users the amount of physical resources should not exceed $O(n)$.

3.3.5.2 Controlling the Performance

Algorithms with hierarchical structures scale better than those with linear. However, even such algorithms cause a loss of performance if problem size increases. This loss should be at most $O(\log n)$ – equivalent to the access time on hierarchical structures – for a set of data with size n.

3.3.5.3 Preventing Software Resources Running Out

In most cases, estimating the demands for years in advance is very difficult to accomplish. An over-compensation for future growth may be even worse than the adaptation of the system on occasional demands.

3.3.5.4 Avoiding Performance Bottlenecks

Algorithms should be decentralized to avoid performance bottlenecks. For shared resources, which are accessed frequently, the concepts of caching and replication should be used.

3.3.6 Procedure

For measuring the scalability of a system [6] defines seven steps, which are described in more detail below:

1. Identify critical use cases
2. Select representative scalability scenarios
3. Determine scalability requirements
4. Plan measurement studies
5. Perform measurements
6. Evaluate data
7. Present results

In [11] the assessment of scalability is introduced in a similar way for an e-business application. In particular, the process character is highlighted. Therefore, it is not sufficient to evaluate the scalability of an application only once in the test phase. Considerations about scalability should be fully integrated into the entire product life cycle – from planning to maintenance.

3.3.6.1 Identify Critical Use Cases

In the first step, the typical use cases, which are most commonly performed, must be identified, because they play a decisive role. Use cases that are not often called, but require many resources, are also important. Overall, the critical use cases represent only a small subset of all possible.

3.3.6.2 Select Representative Scalability Scenarios

In the second step, the main scenarios for scalability in relation to the identified critical use cases must be selected. It is important to note that not all possible scenarios are significant for the scalability tests. Again, it is useful to select the scenarios, which often take place or rather demand many resources during execution.

For the purpose of documentation the use of Unified Modeling Language (UML) is recommended. Typically, sequence diagrams are used for illustrating the links and flows:

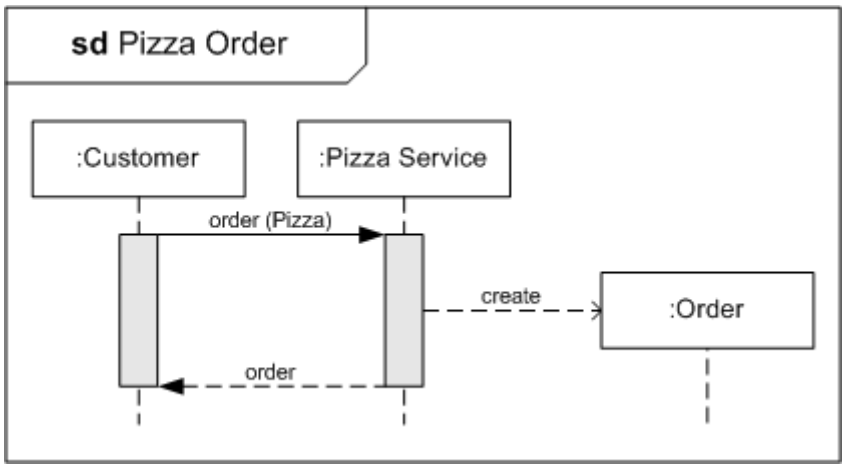


Figure 7: Example of UML Sequence Diagram

3.3.6.3 Determine Scalability Requirements

In the third step, accurate, quantitative and measurable requirements regarding scalability must be defined. Only those allow the selection of a meaningful scale strategy. There are different ways to express these requirements, but one must make sure that they are quantifiable and measurable. Vague descriptions do not help. The conditions, under which the required performance can be achieved, must also be specified.

3.3.6.4 Plan Measurement Studies

In the fourth step, the bottlenecks related to resources must be identified, because they push the measurements. Afterwards, four different configurations for each scaling strategy have to be considered. Next, a generator, which brings the system up into a representative system state, must be developed. Also, the system parameters have to be determined in detail. It makes sense to use adequate tools for the implementation of the measurements. At this step, the selection of such tools can be performed, because all system parameters are known. The documentation of the test plan, in order to make the measurements retraceable and repeatable, is the final task.

3.3.6.5 Perform Measurements

In the fifth step, the measurements are performed and the results are collected and documented. In order to be able to clearly assign the results to the development stages the date and time must always be saved to the benchmarks. Furthermore, the configuration data as well as the version numbers of the software products have to be recorded.

While executing the experiments the following points should also be taken into account:

- The number of competing users in the system should be increased steadily.
- The maximal throughput can be determined by increasing the number of competing users while measuring the throughput at every point. If the total throughput no longer rises or even falls, the maximum throughput has been reached.
- The individual experiments should always take long enough so that a stable state is reached. That is the only chance to obtain a representative value for a measurement.
- For demonstrating the reproducibility of the test results, the individual experiments should always be repeated several times.

3.3.6.6 Evaluate Data

In the sixth step, the measured data must be evaluated. Thereby, it must be checked whether the requirements defined in the third step hold. In addition, the best scaling strategy is selected.

3.3.6.7 Present Results

In the seventh and last step, the results must be presented to the stackholders. Recommendations in terms of scalability should be distributed. In any case, the identified, specified and measured data from the steps above have to be expressed in written form.

3.4 Middleware

Today, the realization of large and distributed software projects is unthinkable without the use of middleware. Even if middleware is a commonly used term, an introduction to this IT-specific concept will be explained in this chapter.

3.4.1 Definitions

In the literature the first definition of middleware can be found in [13] written in 1970. Therein, the term middleware is defined as follows:

> *Computer manufacturer's software which has been tailored to the particular needs of an installation.*

In 1972, a still common definition of middleware was published in [14]. This article gives the following definition:

> *A comparatively new term middleware was introduced because, as some systems had become uniquely complex, standard operating systems required enhancement or modification; the programs that effected this were called middleware because they came between the operating system and the application programs.*

Finally, the concept of middleware becomes popular in computer science due to [15]. The term middleware services is described as quoted below:

> *To help solve customers' heterogeneity and distribution problems, and thereby enable the implementation of an information utility, vendors are offering distributed*

system services that have standard programming interfaces and protocols. These services are called middleware services, because they sit "in the middle", in a layer above the OS and networking software and below industry-specific applications.

In connection with a distributed system [42] provides a brief, but meaningful definition of the searched term:

In a distributed computing system, middleware is defined as the software layer that lies between the operating system and the applications on each site of the system.

Now, the reader should have a sufficient idea of the concept of middleware and be able to understand the following remarks.

3.4.2 Characteristics

[15] lists fundamental properties and requirements which are characteristic for middleware services. These characteristics can be summarized as follows and are discussed in more detail below:

- Middleware services are multi-purposed and distributed services which are always positioned between the platforms and the applications.
- Middleware services are defined by their APIs and protocols and support at least a standard API and a standard protocol or at least a published one.
- Middleware services are available in different implementations which allow the execution on multiple platforms.

The previous definitions show the reader already quite plainly, where middleware in a computer system is located, namely between applications and platforms. This context can be illustrated as follows:

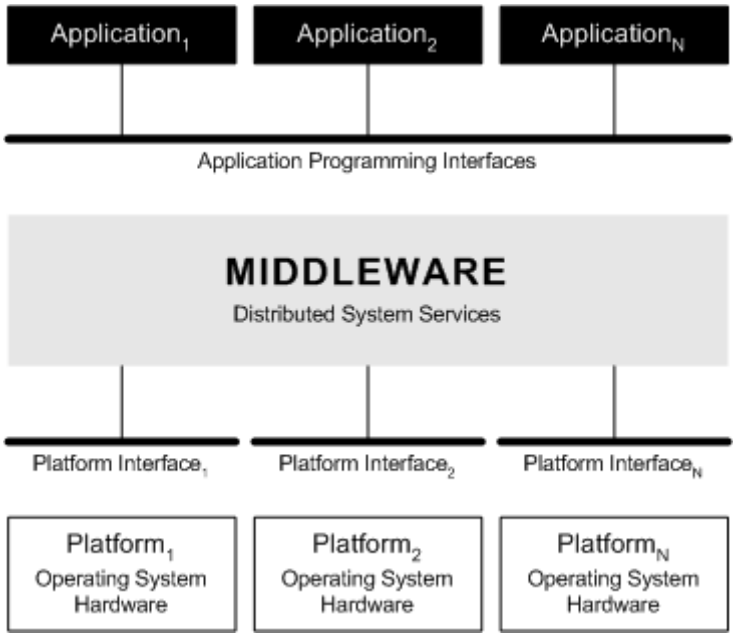


Figure 8: Positioning of Middleware

The applications access the middleware and their provided services via interfaces. Thus, each middleware is defined by the APIs and the protocols it supports. To minimize the

dependencies on the different platforms, middleware is not directly set up on the platforms, but uses interfaces itself. That ensures the portability of middleware as well as their services.

Therefore, middleware is a system that acts as a broker between applications, whereby the complexity of these applications and their infrastructure is hidden. It can also be seen as a protocol at a higher layer, which enables decoupled software components the exchange of data. On this account, middleware is a key technology for the implementation of distributed systems.

Today, a common model for the realization of large information systems is the 3-tier architecture. Since each tier can be divided in further layers, the term n-tier or multi-tier architecture has been established for this model. Each layer can be understood as a separate sub-system that implements one of the aspects of a system, namely data presentation, data processing and data storage. While system components can communicate within a layer freely, the communication between components of different layers is subjected to strict guidelines.

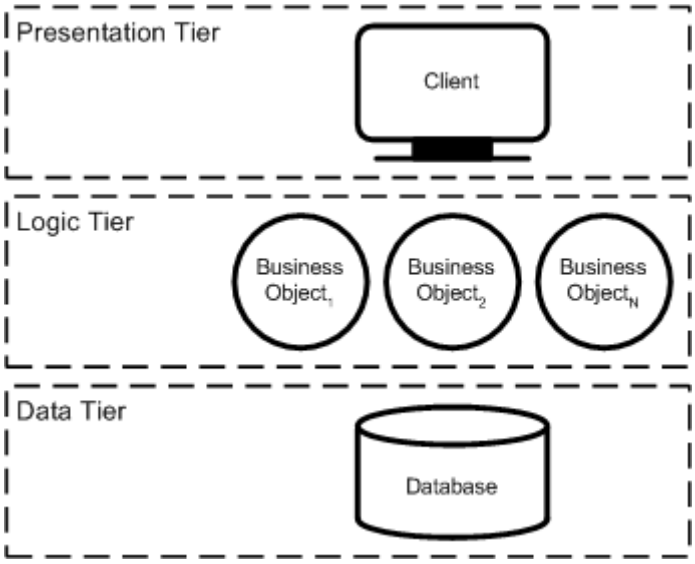


Figure 9: 3-tier Architecture

In relation to this architecture model, middleware is mainly located in the logic tier. There it provides the most commonly required services for the implementation of distributed systems as well as a runtime environment for the business objects. In this context, such a system is often referred to as application server. The integration of the user-defined business objects into an application server – this procedure is called deployment – is performed using well-defined interfaces.

However, from the point of view of a developer the use of middleware systems makes sense only if efficient services can be used for their implementation. An overview of issues, for which a middleware system should provide corresponding services, gives [43]:

- Communication facilities
- Naming

- Persistence
- Distributed transactions
- Security

3.4.3 Classifications

There are several approaches for the classification of middleware systems. Some of these are presented in the following.

3.4.3.1 Transactional / Procedural / Message-oriented / Object-oriented

[16] distinguishes between transactional, procedural, message-oriented and object-oriented middleware and describes them as follows:

- Transactional middleware: This middleware, also referred to as Transaction Processing Monitor (TPM), supports the use of distributed synchronous transactions. One of the main tasks of such a monitor is the coordination of the transactional requests between the clients and the processing servers.
- Procedural middleware: This middleware, also referred to as Remote Procedure Call (RPC), enables the invocation of functions in different address spaces. Normally, the called functions are executed on a computer that is different from the one that hosts the calling program. Many operating systems support this type of middleware, but the implementations are often incompatible.
- Message-oriented middleware: This middleware uses messages for communication. A differentiation can be made between message queuing and message passing. Message queuing is an indirect communication model, because all communication partners use queues for sending and receiving messages. Message passing is a direct communication model. Here, the information is sent directly to the interested participants. The best-known variant of this middleware type is the publish-subscribe model.
- Object-oriented middleware: This middleware extends the concept of RPC by adding object-oriented aspects such as inheritance, object references and exceptions.

For all of these categories the referenced article lists typical products and deals with the issues network communication, coordination, reliability, scalability and heterogeneity. Furthermore, it explores the advantages and disadvantages for each middleware class and gives advices for the use.

3.4.3.2 Distributed Tuples / Remote Procedure Calls / Message-oriented / Distributed Object

Another classification is presented in [17] which differentiates four categories of middleware:

- Distributed tuples: This middleware is currently the most popular one. Such tuples represent the abstraction of data which are managed by distributed relational databases. Using the Structured Query Language (SQL), which is based on the set theory and predicate calculus, allows the manipulation of these tuples. In addition,

a distributed relational database provides the abstraction of a transaction. For end-to-end resource management of client queries, server-side process management and managing multi-database transactions TPMs are used.

- Remote procedure calls: as procedural middleware above
- Message-oriented middleware: as message-oriented middleware above
- Distributed object middleware: as object-oriented middleware above

In comparison to the previous classification only the first category, the distributed tuples, is newly introduced. The remaining three categories correspond to the last three in chapter 3.4.3.1 and differ from them only marginally.

3.4.3.3 Communication-oriented / Application-oriented / Message-oriented

Finally, the categorization of middleware systems published at [44] is given. This alternative distinguishes between the following three types:

- Communication-oriented middleware: The main focus is on the abstraction of network programming. RPC, Java RMI and Web Services are listed as examples.
- Application-oriented middleware: In addition to the communication mainly the support of distributed applications takes centre stage. As examples Common Object Request Broker Architecture (CORBA), J2EE and .NET are mentioned.
- Message-oriented middleware: This category does not use method and function calls, but is based on the exchange of messages. The message format is specified by the middleware system and the message exchange can happen in synchronous and asynchronous way. The asynchronous variant uses queues, in which the producers put messages and from which the consumers take messages. As a result of this procedure the system components become completely decoupled, so that the error rate of the system decreases. An example for that category is Java Message Service (JMS).

3.4.4 Concepts

Next, the middleware concepts are described in detail, which are relevant in this work. Please note that these are only a few of many existing concepts.

3.4.4.1 eXtensible Virtual Shared Memory

The concept of eXtensible Virtual Shared Memory was developed by the Space Based Computing Group of the Institute of Computer Languages at the Vienna University of Technology.

First, the concept of Virtual Shared Memory (VSM) is presented, because XVSM is based on it. VSM is described at [45] as a shared data space for the communication and collaboration of several autonomous software components called peers. The concept of VSM is well-known from parallel computing.

In contrast to conventional middleware concepts this approach pursues a new goal. Since the peers use a shared virtual data space to interact among each other, they are only loosely coupled and the way of thinking is changed from client-server to peer-to-

peer. Due to these changes the building of more stable and scalable systems is made possible. The paradigm of VSM is also known as Space Based Computing (SBC).

XVSM expands the concept of VSM, so that the functionality of the basic system can be extended, if required, through the usage of aspects. Hence, XVSM is also called programmable VSM. The expandability offers immense possibilities for someone who uses such a system.

Now, XVSM can be adapted by everyone in such a manner that it satisfies the user's demands. If necessary, new pluggable components can be added to the system, existing components can be replaced by others and components that are no longer needed can be removed. Time is over, when a user had to handle a lot of functionality, and therefore complexity, which is not needed. Another advantage is that the dependency on individual vendors decreases, because components from different suppliers can be used in this middleware.

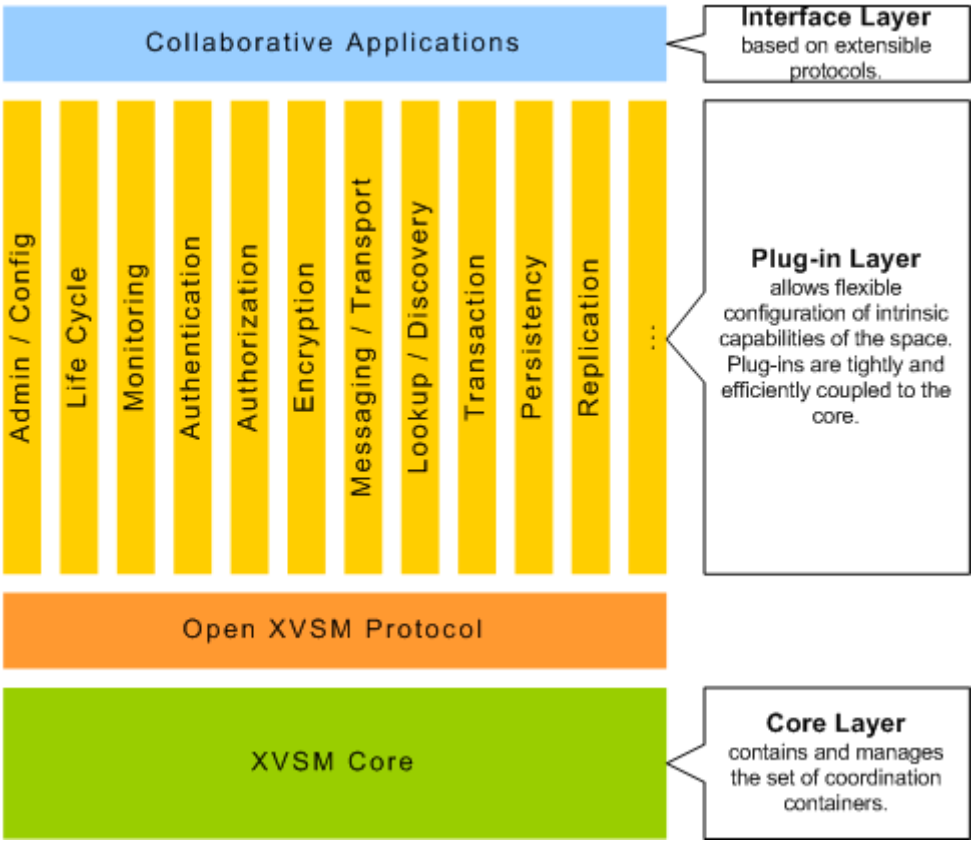


Figure 10: Architecture of XVSM

From a technical point of view, the container is the basic part of XVSM. A container – there can also be several instances of it – represents the shared virtual data space and is responsible for managing the entries which can be structured Linda tuples or serializable objects. Each container is accessible via a Uniform Resource Locator (URL) which enables the access via the Internet. In addition, for each container the maximum number of entries can be limited.

For managing the entries in a container or rather for retrieving entries from a container the following coordinators and selectors are available, which are described in detail in [46]:

- Random
- First In, First Out (FIFO)
- Last In, First Out (LIFO)
- Key
- Vector
- Linda

A peer can execute five operations in connection with a container:

- write: This operation inserts an entry into the container. If the container reaches the maximum number of entries, then this operation blocks.
- read: This operation reads an entry from the container using a selector. If no relevant entry can be found, then this operation blocks.
- take: This operation reads and at the same time removes an entry from the container using a selector. If no relevant entry can be found, then this operation blocks.
- shift: This operation inserts an entry into the container. If the container reaches the maximum number of entries, then an existing entry is replaced.
- destroy: This operation removes an entry from the container using a selector. If no relevant entry can be found, then this operation blocks.

These actions can also be performed in form of bulk operations. This means that through one invocation more than one entry is manipulated or retrieved. Furthermore, the container supports implicit and explicit transaction handling.

In addition, aspects can be defined on each container, so that on the occurrence of a specific event user-defined actions can be executed.

With regard to the classifications above the basic system of XVSM belongs to the category of distributed tuples. In view of the expandability of XVSM this concept holds the potential to become an application-oriented middleware.

3.4.4.2 JavaSpaces

JavaSpaces was specified by Sun Microsystems based on the concepts of Linda and defined within the context of Jini. Therefore, short descriptions of these two technologies will be given at the beginning.

[47] characterizes Linda as coordination and communication model for parallel processes that operate on objects which are stored in and retrieved from a VSM. Such objects are called tuples and the associated VSM is referred to as Tuple Space. The model is implemented as coordination language and provides primitive operations for the transfer of tuples.

Jini which stands for "Jini is not initial" is explained at [48] as service-oriented architecture that defines a programming model on the basis of Java technology. It allows the construction of secure and distributed systems that are scalable, evolvable and flexible.

However, the main focus here is on JavaSpaces, and not on Linda or Jini. Therefore, the following introduction based on [49] informs about the essentials of this middleware concept:

JavaSpaces defines a model which enables the exchange of objects in a distributed Java application. A JavaSpace can be considered as a form of VSM, which allows a client to write objects into the space and to read or take objects from the space. Furthermore, a client can register an event listener with the space to be notified about the occurrence of certain events.

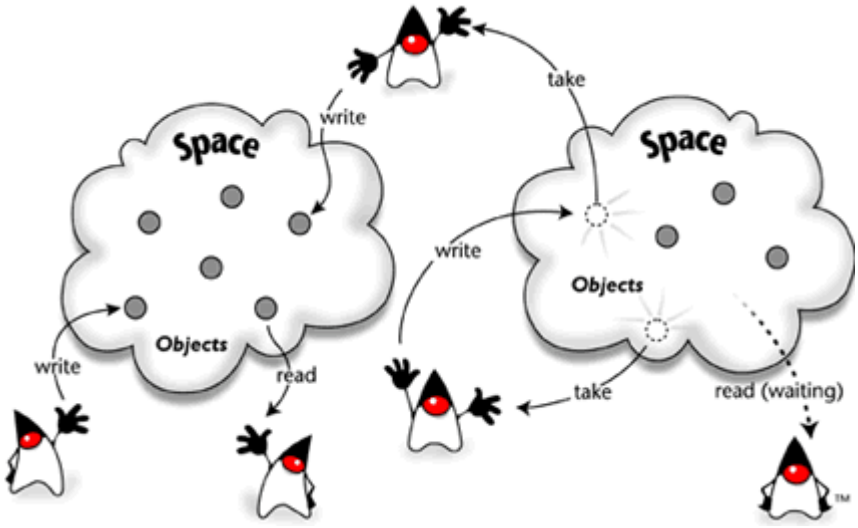


Figure 11: Operations in JavaSpaces

From a technical point of view, the JavaSpace is the central component. In such a space, all serializable objects can be stored, which implement a specific interface and offer public fields and a public constructor. Due to these restrictions such entries are suitable as communication objects only. The concept, that each entry has a unique identifier, is not supported innately.

The clients use the lookup service, which provides a central registry of services available, to get a reference on a JavaSpace. Also, each client can register its JavaSpace with this service so that other clients can access it. However, it should be noted that this is only one possibility to share a space between several clients.

In connection with a JavaSpace five operations can be executed by a client:

- write: This operation inserts an entry into the space.
- read: This operation reads an entry from the space using a template for selection. If no relevant entry can be found, then this operation blocks until the passed timeout is reached.
- readIfExists: as read operation, but without blocking

- take: This operation reads and at the same time removes an entry from the space using a template for selection. If no relevant entry can be found, then this operation blocks until the passed timeout is reached.
- takeIfExists: as take operation, but without blocking
- notify: This operation registers an event listener with the space.

Depending on the JavaSpaces implementation, there are different possibilities to keep the entries:

- Hold the entries in the memory of a Java Virtual Machine (JVM).
- Hold the entries in the memory of a cluster of several JVMs.
- Store the entries in synchronous or asynchronous way in a persistent memory such as a file or a database.

Of course, the concept of JavaSpaces supports transactions. It uses the transaction mechanism of Jini, which deals with distributed transactions including two-phase commit. In order to participate in a transaction the client has to pass a reference of the transaction object to the method. The transaction manager is responsible for controlling the transactions. A reference on it usually is available via the lookup service.

An unusual feature of JavaSpaces is the usage of leases, as is customary in Jini. If required, the life time of an entry as well as the validity of a transaction can be defined by such a lease. After the expiration of the deadline the entry is removed from the space, or rather the transaction is rolled back.

Referring to the categorizations above JavaSpaces belongs also to the distributed tuples.

3.4.4.3 Java 2 Platform, Enterprise Edition

The concept of Java 2 Platform, Enterprise Edition – recently, the name Java Platform, Enterprise Edition (Java EE) has been established – was developed and published by Sun Microsystems. Following, a brief introduction to J2EE inspired by [50] is presented to give the reader an overview about this concept.

J2EE is a specification that defines a standard for enterprise solutions through a component-based development model on the basis of a powerful set of APIs. The figure below gives an architectural overview of J2EE.

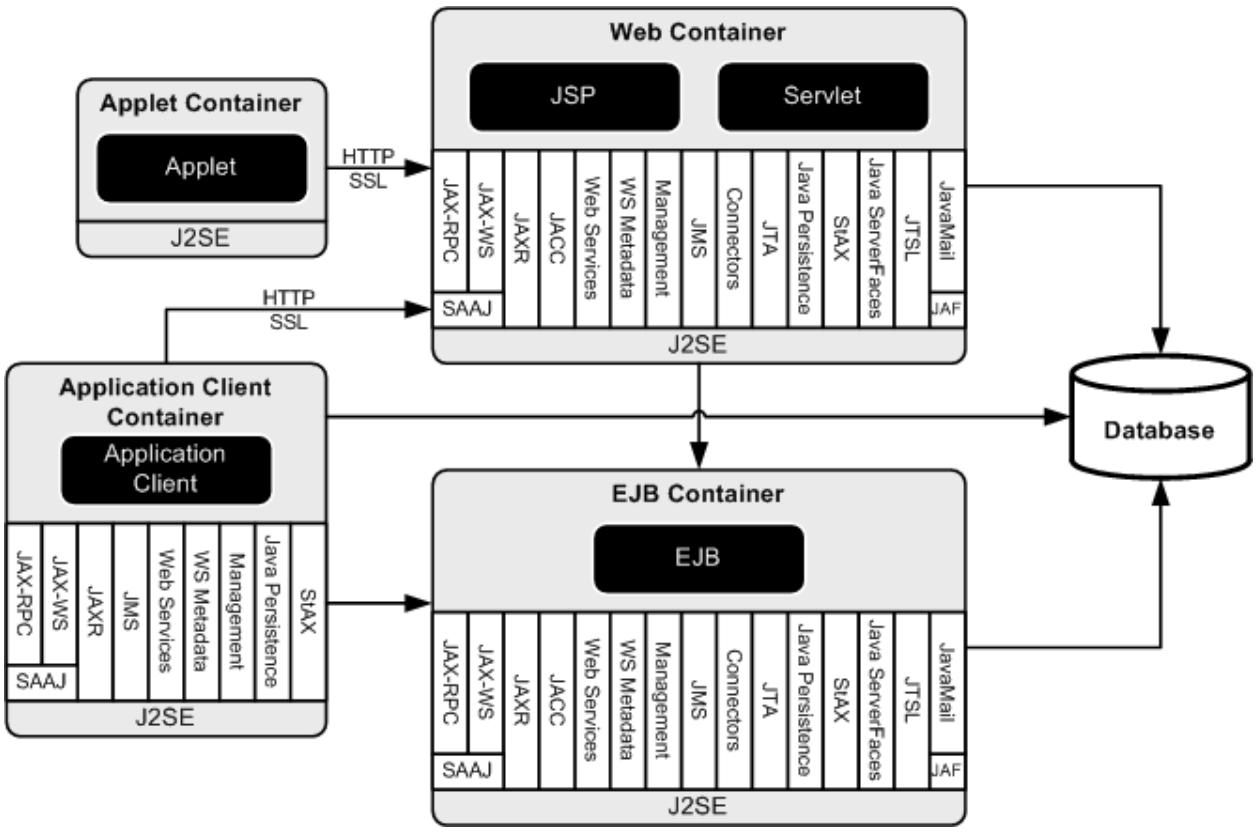


Figure 12: Architecture of J2EE

Each J2EE application is composed of components. These are self-contained functional software units which are written in Java and must be compiled as usual. The components can communicate among each other and have to comply with the J2EE specification which provides the following types of components:

- Applets, Application clients: These are client components that run on the client.
- JavaServer Pages (JSP), Servlets: These are web components that run on the server.
- Enterprise JavaBeans: These are business components that run on the server.

In order to simplify the development of such components, the J2EE specification defines various containers which provide underlying services such as security, transaction management, Java Naming and Directory Interface (JNDI) lookups and remote connectivity. To take advantage of the containers' services, the components must be assembled and deployed into such a container. Only then the components get access to the provided services via the corresponding APIs. Therefore, a container acts as an interface between the component and the platform specific functionalities.

Well, this should be sufficient information to get an overview of J2EE. Following, only a few words about EJB, because this concept will be of importance below.

The EJB components, or enterprise beans, symbolize and encapsulate the business logic in a J2EE application. The structure and body of such a component is given by the specification, so the developer can concentrate on implementing the business functionalities. Of course, EJB components can invoke one another.

There are three kinds of enterprise beans:

- Entity beans: These components represent and model the non-transient data of the system. The persistence of the data can be managed either by the developer, referred to as Bean Managed Persistence (BMP), or by the EJB container, referred to as Container Managed Persistence (CMP).
- Session beans: These components represent the business logic and model the processes of the system. J2EE differentiates between stateless and stateful session beans.
- Message-driven beans: These components establish the asynchronous communication to the EJB technology. JMS provides the basis for that.

For the design of a J2EE application based on EJB components, [51] advises the following approach, which is known under the name of session façade:

To minimize the dependencies of a client on the underlying business logic as well as to avoid unnecessary network traffic, the client should never access an entity bean directly. All the client actions should rather be encapsulated by one or more session beans. This session façade hides the complexity of the business logic from the client and provides a service interface for it. So the client depends only on this service interface. That has not only the advantages listed above, but also simplifies the development and the maintenance of such an application.

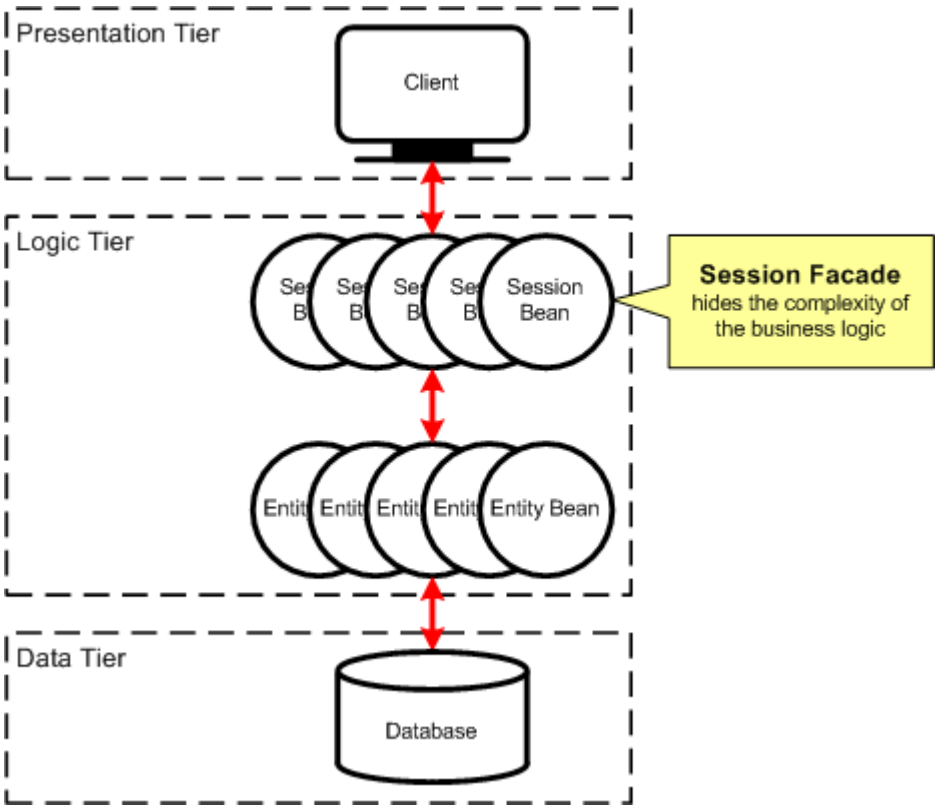


Figure 13: Session Façade Pattern

According to the foregoing classifications, the J2EE specification defines an application-oriented middleware concept.

3.4.5 Systems

After the description of the concepts, in this chapter the systems that implement these paradigms will be presented.

3.4.5.1 MozartSpaces

MozartSpaces released at [52] is the Java based reference implementation of the XVSM concept which was introduced in chapter 3.4.4.1. Its intuitive API enables easy use of this middleware system as the following code example demonstrates:

```
ICapi capi = new Capi();

URI site = new URI("tcpjava://mssrv01.complang.tuwien.ac.at:4321");
ContainerRef cref = capi.lookupContainer(null, site, null);

AtomicEntry entry = new AtomicEntry<String>("Hello Space!");
capi.write(cref, 0, null, entry);

capi.shutdown(null, true);
```

Several XVSM extensions such as replication, improved transaction support, distributed sessions as well as deployment and visualization tools for the configuration are currently in development for this middleware system. Also, new APIs for JavaSpaces, JavaScript, JMS, Python and Scheme are in integration for expanding the use of MozartSpaces.

3.4.5.2 GigaSpaces eXtreme Application Platform

GigaSpaces XAP released at [53] introduces the virtualization on the level of middleware.

In contrast to traditional middleware systems, where data, messaging and service implementations are centralized, GigaSpaces XAP visualizes these tiers and replaces them by a virtual one. It is designed to operate in a "cloud" which distributes the underlying implementations on a cluster of several physical servers.

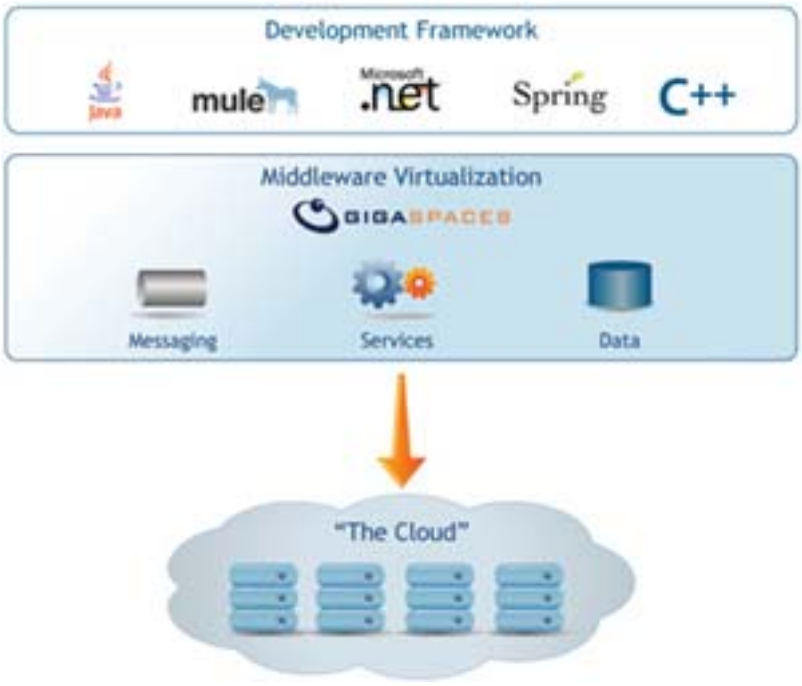


Figure 14: Architecture of GigaSpaces XAP

For applications using GigaSpaces XAP as a platform the middleware system appears as a centralized server. On demand, multiple applications can share the same containers and environment. This enables the building of loosely coupled service architectures. All supported APIs, languages and logical tiers are provided with a shared cluster for transaction semantics, distributed state management, reliability and scalability.

GigaSpaces XAP supports, implements and even extends the concept of JavaSpaces which was introduced in chapter 3.4.4.2. Applications can use the corresponding API for getting access to the platform. The behaviour of the middleware can be configured and does not need to be implemented, which is advantageous.

3.4.5.3 JBoss Application Server

JBoss AS released at [54] is an application server that fully complies with the J2EE concept which was introduced in chapter 3.4.4.3 and provides a highly flexible service-oriented architecture on which developers can build their own products.

The application server is made up of several modules. This principle allows the extension and adaption of this middleware system according to the users' requirements. The modularization is achieved by the use of Java Management Extensions (JMX), an excellent tool for the integration of software. JMX provides a framework that allows the user to integrate modules, containers and plug-ins. JBoss AS uses JMX as an integration bus, into which the different JBoss modules plug in. These are declared as MBean services and can be administered using JMX. The figure above gives a survey of the JBoss modules.

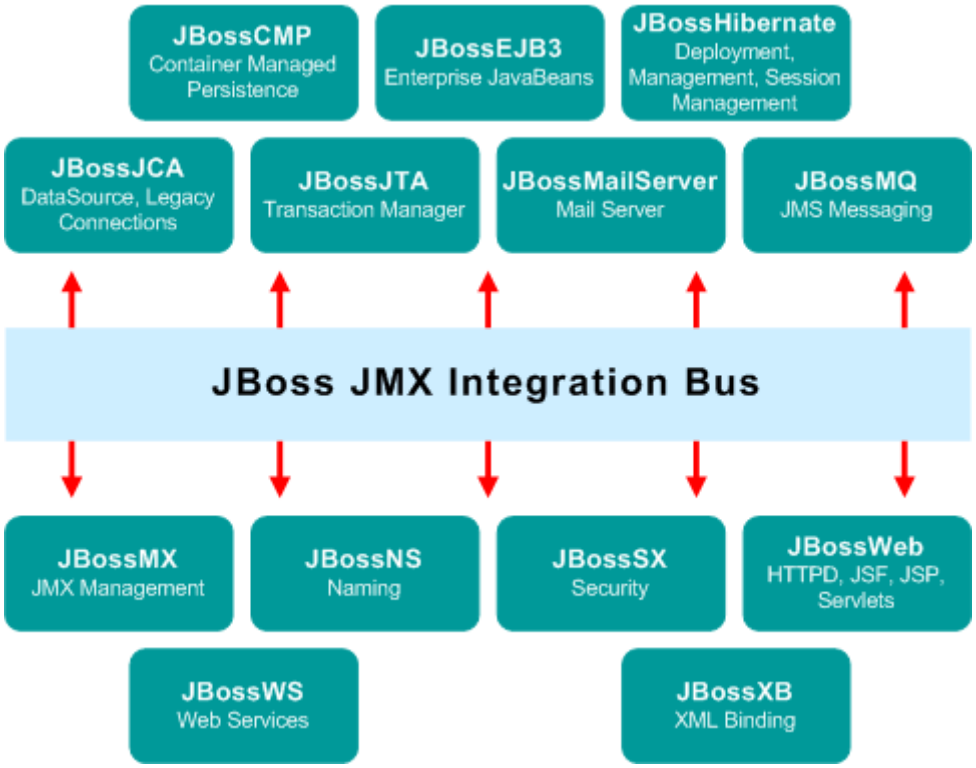


Figure 15: Architecture of JBoss AS

All in all, JBoss AS provides plenty of functionalities required by enterprise solutions and enables the implementation and execution of powerful J2EE applications on its basis.

4 Benchmarks

In the following chapter the environment and the different benchmark scenarios which describe the particular test cases and allow an estimation of the system's performance and scalability will be specified. Furthermore, the results of their executions will be presented as well as their interpretations.

4.1 Preface

Issues associated with the implementation of benchmarks will be discussed below.

4.1.1 Benchmark Suite

The execution of the benchmark scenarios is carried out using a custom-built software tool written in Java, the benchmark suite. It can be used for measuring the performance of certain test cases with different middleware systems and on the basis of these measurements appropriate scalability studies can be performed. The tutorial and the source code of this software tool including the scenarios below are available at [52].

The benchmark suite is made up of different applications which encapsulate the various test cases. Depending on which middleware technology will be benchmarked and what kind of benchmark will be executed, the user must select one of the following applications for benchmark implementation:

- Serial benchmark for XVSM (MozartSpaces)
- Concurrent benchmark for XVSM (MozartSpaces)
- Block benchmark for XVSM (MozartSpaces)
- Serial benchmark for JavaSpaces (GigaSpaces XAP)
- Concurrent benchmark for JavaSpaces (GigaSpaces XAP)
- Block benchmark for JavaSpaces (GigaSpaces XAP)
- Serial benchmark for J2EE (JBoss AS)
- Concurrent benchmark for J2EE (JBoss AS)
- Block benchmark for J2EE (JBoss AS)

The different types of benchmarks can be described as follows:

- Serial benchmarks: A defined number of a particular operation is executed in series. This corresponds to the scheme that one client accesses a server.
- Concurrent benchmarks: A defined number of a particular operation is executed concurrently by a defined number of threads. This corresponds to the scheme that several clients access a shared server.
- Block benchmarks: A defined number of a particular operation block is executed concurrently by a defined number of threads. This also corresponds to the scheme that several clients access a shared server.

The benchmark suite supports the following concepts and systems:

- XVSM (MozartSpaces): The container can be addressed in both embedded and remote mode. In addition, implicit or explicit transaction mechanism can be selected. Except for aspect-oriented methods, all operations are supported. All coordinators listed in chapter 3.4.4.1 are available.
- JavaSpaces (GigaSpaces XAP): The access to the space can also be realized in embedded or remote mode. Furthermore, either a transient or a persistent scheme can be used. For storing the entries permanently, the database engine H2 released at [55] is used. In connection with transactions, the testing person has the choice between none and explicit transactions. Except for notification, all operations can be benchmarked.
- J2EE (JBoss AS): The application server can be accessed in remote mode only. For emulating the established operations of XVSM and JavaSpaces, a session façade pattern is used. The operations create, find and remove can be executed via a stateless session bean, which operates on entity beans using CMP. All data are made persistent in the default database, namely Hypersonic SQL, which is provided by the application server and released at [56]. This persistence mode has been selected, because J2EE applications typically use databases for holding the entity beans' state. Again, one can select between none and explicit transactions.

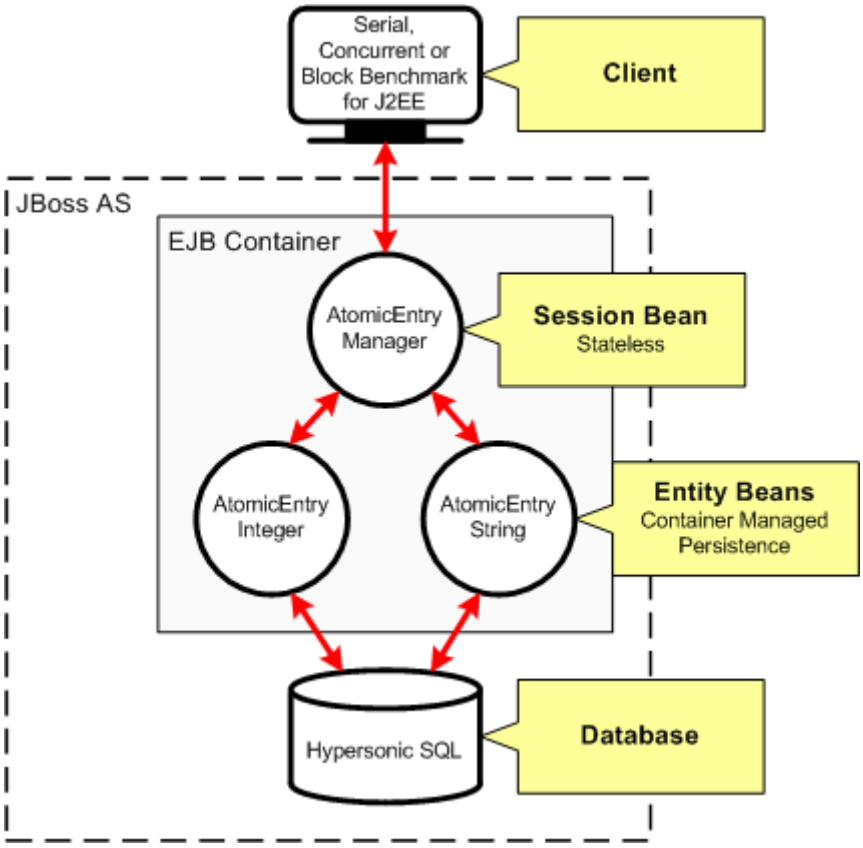


Figure 16: Design of Benchmark Suite for J2EE

Regarding the implementation, only the standard APIs of the concepts are used in the benchmark suite, which ensures that other middleware systems, based on these concepts, can easily be integrated. From this it follows that native system calls are not applied.

Each of these applications is adaptable to individual needs of the user via command line arguments or rather configuration file. This enables the definition and specification of individual and useful benchmark scenarios. As a representative for all applications the usage of the serial benchmark for XVSM (MozartSpaces) is given below:

```
USAGE:
  org.xvsm.benchmarks.BenchmarkSerial {arguments}
  org.xvsm.benchmarks.BenchmarkSerial {property-file}

ARGUMENTS:
  -datafile {string}
  -containertype EMBEDDED|REMOTE
  -containeruri {string}
  -containersize {integer}
  -coordinatortype FIFO|KEY|LIFO|RANDOM|VECTOR
  -transactiontype EXPLICIT|IMPLICIT
  -atomicentrytype INTEGER|STRING
  -atomicentrysize {integer}
  -iterations {integer}
  -operationnumber {integer}
  -operationtype DESTROY|READ|SHIFT|TAKE|WRITE
  -monitorreset {boolean}
```

The following pseudo code demonstrates the connection between the arguments transactiontype, iterations, operationnumber, operationtype and monitorreset:

```
FOR i = 1 TO iterations DO

    IF monitorreset = true THEN
        resetMonitor()
    END IF

    startMonitor()

    IF transactiontype = explicit THEN
        beginTransaction()
    END IF

    FOR o = 1 TO operationnumber DO
        executeOperation(operationtype)
    END FOR

    IF transactiontype = explicit THEN
        commitTransaction()
    END IF

    stopMonitor()

    logMonitor()

END FOR
```

This means that a particular operation is repeated as often as defined. Such an operation block can be embedded in a transaction. The needed time is measured by a monitor whose counter is resettable, if required, and then logged into the data file. This procedure represents a single iteration. The total number of iterations can be set by the user, too.

With some test cases there are specific things to consider:

- XVSM (MozartSpaces) with Key or Linda coordinator, JavaSpaces (GigaSpaces XAP), J2EE (JBoss AS): New entries are always inserted with a unique key. The natural numbers, starting at 0 and increased by 1, are used as identifiers.
- XVSM (MozartSpaces) with Vector coordinator: New entries are always appended after the last entry of the vector.
- XVSM (MozartSpaces) with Key, Vector or Linda selector, JavaSpaces (GigaSpaces XAP), J2EE (JBoss AS): The keys or rather indices used for reading, taking or destroying entries are calculated in such a way that the accesses are evenly distributed over all data.

The execution of such a benchmark application returns a commented data file with the number of operations (x-dimension) and the associated execution times (y-dimension). Here is an example of such a data file:

```
# ================================================
# BENCHMARKSERIAL - XVSM [MozartSpaces]
# ================================================
#
# ------------------------------------------------
# Configuration
# ------------------------------------------------
# datafile=random.dat
# containertype=EMBEDDED
# containeruri=tcpjava://localhost:4321
# containersize=-1
# coordinatortype=RANDOM
# transactiontype=IMPLICIT
# atomicentrytype=INTEGER
# atomicentrysize=1
# iterations=10
# operationnumber=10000
# operationtype=WRITE
# monitorreset=false
# spacespropertyfile=null
#
# ------------------------------------------------
# Data
# ------------------------------------------------
0      0
10000  1316
20000  2370
30000  3389
40000  4410
50000  5446
60000  6455
70000  7480
80000  8497
90000  9540
100000 10627
```

To make sure that each individual benchmark scenario will return realistic data, the test cases are repeated exactly three times, which results in three data files. These files are processed by another application of the benchmark suite, the data processor. It reads all data from these three files and writes the minimum execution time for each number of operations into a new data file. This procedure ensures that each benchmark scenario delivers representative values. In case of scalability investigations, the data

processor also provides suitable methods for generating data files in an adequate format.

```
USAGE:
  org.xvsm.tools.DataProcessor {mode} {infile1} {infile2} ... {infileN} {outfile}

MODE:
  -d   generate data file
  -n   generate n-fold file
  -p   generate performance histogram file
  -s   generate scalability histogram file
```

Finally, the chart is generated on basis of the processed data. The visualization of the data is done by using Gnuplot which was introduced in chapter 3.1.2.3.

In summary, the workflow for the implementation of a benchmark scenario, based on the benchmark suite, can be illustrated as follows:

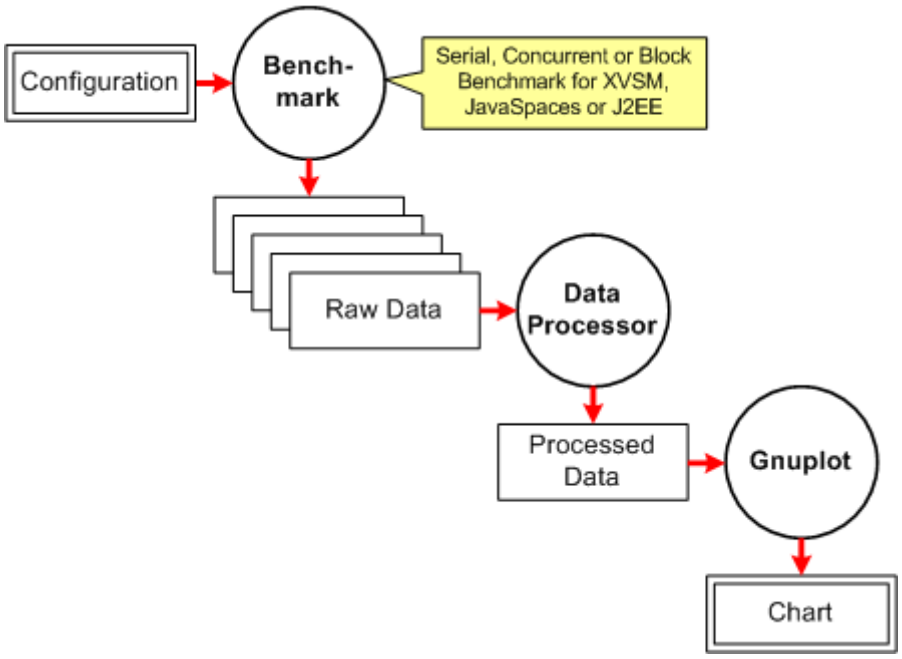


Figure 17: Workflow in Benchmark Suite

4.1.2 Middleware Configurations

The number of possible configurations for each middleware system is enormous. Thus, a reasonable and representative subset of these configurations must be selected.

Depending on the benchmark scenario, the criteria listed below are used in variation:

- XVSM (MozartSpaces):
  - Container types: embedded, remote
  - Persistence types: transient
  - Transaction types: implicit, explicit
  - Coordinator/Selector types: Random, FIFO, LIFO, Key, Vector, Linda
- JavaSpaces (GigaSpaces XAP):
  - Space types: embedded, remote

- Persistence types: transient, persistent
- Transaction types: none, explicit

- J2EE (JBoss AS):
  - Container types: remote
  - Persistence types: persistent
  - Transaction types: none, explicit

Principally, the execution time of the operations depends on the mode used for accessing the container or space. Also, the persistence mode plays a decisive role. An embedded container or space with transient data management results in fast execution times. The converse configuration, namely remote container or space with persistent data management, leads to worse performance. The whole relation is given by the following figure:

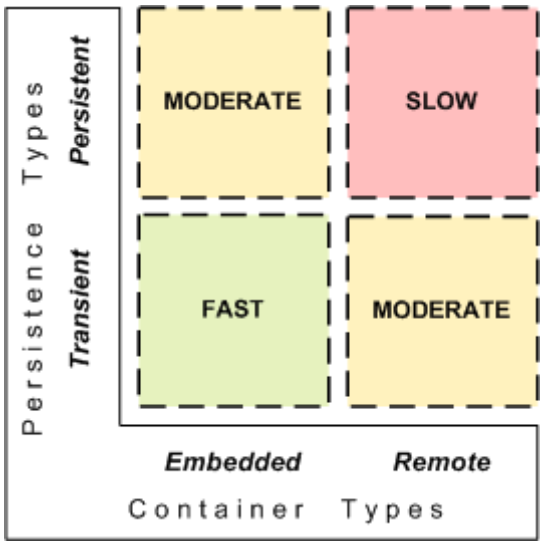


Figure 18: Middleware Configuration and Execution Time

Furthermore, the default configurations of the middleware systems are modified only slightly. This means that each system is nearly executed as delivered by its vendor.

If the systems are accessed remotely, the server instance of MozartSpaces is started by the Java class org.xvsm.server.Server, GigaSpaces XAP is launched with embedded Mahalo via the script gsInstance and JBoss AS is called via the script run. All systems are run with the JVM options –server, -Xms256 and -Xmx1024.

4.1.3 Machine Configurations

The number of potential machine configurations and network structures is almost endless. In this work three arrangements are distinguished. Depending on the corresponding benchmark scenario, different layouts for the central processing unit (CPU), different sizes of random access memory (RAM) and also different operating systems are employed.

It must be noted that the processing power of a machine powered by a CPU with 2 cores is not the same as of a machine powered by 2 separate CPUs. The general rule is

that such a machine has at most the 1.5-fold computing power. The following table gives an overview about the relations:

Processors	Cores per Processor	Processing Power
1	1	1-fold
	2	1.5-fold
2	1	2-fold
	2	3-fold

4.1.3.1 Configuration (1)

In the event that a serial benchmark accesses the middleware in embedded mode, both applications run within the same JVM and, of course, on the same machine.

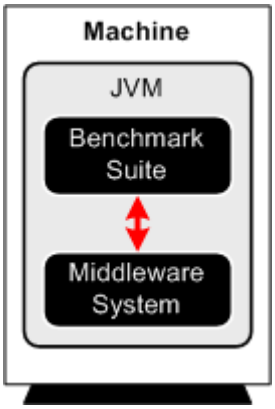


Figure 19: Serial Benchmark with Embedded Middleware System

The machine is then composed in such a way:

- Machine:
  - CPU: 2 processors, each with 2 cores activated (2x Dual-Core AMD Opteron 2210 with 1.8 GHz, 1 MB Cache)
  - RAM: 3 GB
  - OS: Windows Vista
  - Architecture: x86

4.1.3.2 Configuration (2)

It is also imaginable that a serial benchmark calls the middleware remotely. In this case the two applications run within different JVMs and on different machines.

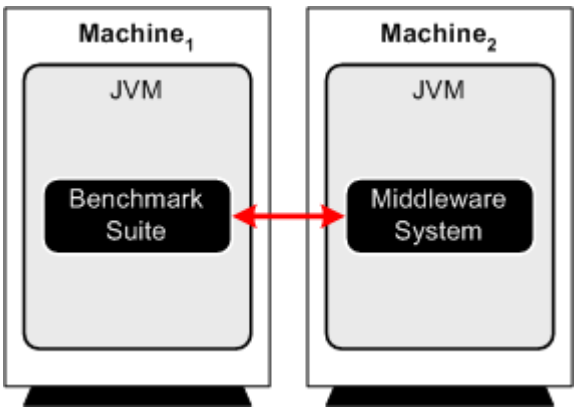


Figure 20: Serial Benchmark with Remote Middleware System

The machines and the intermediate network are then realized as follows:

- $Machine_1$:
  - CPU: 1 processor with 2 cores activated (Dual-Core AMD Opteron 2210 with 1.8 GHz, 1 MB Cache)
  - RAM: 4 GB
  - OS: Linux
  - Architecture: x86
- $Machine_2$:
  - CPU: 2 processors, each with 2 cores activated (2x Dual-Core AMD Opteron 2210 with 1.8 GHz, 1 MB Cache)
  - RAM: 3 GB
  - OS: Windows Vista
  - Architecture: x86
- Network:
  - Protocol: Ethernet
  - Bitrate: 100 MBit/s

4.1.3.3 Configuration (3)

The third and last case is that the execution of a benchmark is based on threads which access the middleware system remotely. Each thread represents a client. This approach takes place during concurrent and block benchmarks. Here, of course, the two applications run again within different JVMs and on different machines.

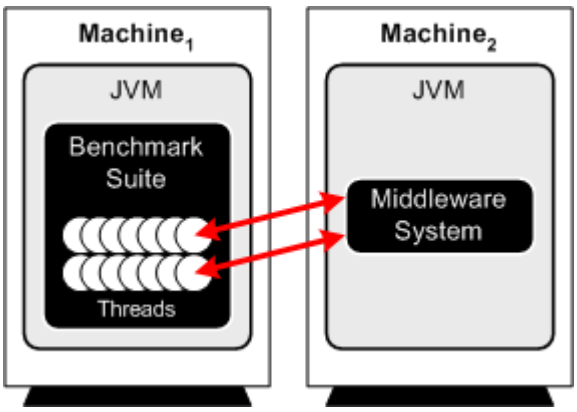


Figure 21: Concurrent or Block Benchmark with Remote Middleware System

Since such benchmarks target on the measurement of scalability, there are different versions for the machines' realizations:

- $Machine_1$:
  - CPU: 1 processor with 2 cores activated (Dual-Core AMD Opteron 2210 with 1.8 GHz, 1 MB Cache)
  - RAM: 4 GB
  - OS: Linux
  - Architecture: x86
- $Machine_2$ (Variant A → 1-fold Processing Power):
  - CPU: 1 processor with only 1 core activated (Dual-Core AMD Opteron 2210 with 1.8 GHz, 1 MB Cache)
  - RAM: 1 GB
  - OS: Windows Vista
  - Architecture: x86
- $Machine_2$ (Variant B → 2-fold Processing Power):
  - CPU: 2 processors, each with only 1 core activated (2x Dual-Core AMD Opteron 2210 with 1.8 GHz, 1 MB Cache)
  - RAM: 2 GB
  - OS: Windows Vista
  - Architecture: x86
- $Machine_2$ (Variant C → 3-fold Processing Power):
  - CPU: 2 processors, each with 2 cores activated (2x Dual-Core AMD Opteron 2210 with 1.8 GHz, 1 MB Cache)
  - RAM: 3 GB
  - OS: Windows Vista
  - Architecture: x86
- Network:
  - Protocol: Ethernet
  - Bitrate: 100 MBit/s

4.1.4 Software Versions

In the following a survey of the version numbers of the software products used in connection with the benchmark implementation is given:

- Operation Systems:
  - SUSE Linux 2.6.18.2-34-bigsmp
  - Microsoft Windows Vista Business 6.0 (Build 6001: Service Pack 1)
- Java Environment:
  - Java SE Runtime Environment 1.6.0_07-b06
  - Java Hotspot Client/Server VM 10.0-b23
- Middleware Systems:
  - MozartSpaces 1.0-alpha (Build 3221)
  - GigaSpaces XAP Community 6.5.0-GA (Build 2352)
  - JBoss AS 4.2.3.GA
- Benchmark Tools:
  - BenchmarkSuite 1.0-alpha (Build 3221)
  - Gnuplot 4.2.3

4.1.5 Metrics and Diagrams

The metrics and diagrams which are used for rating and illustrating the performance and scalability of the middleware systems by the benchmark scenarios are described subsequently.

4.1.5.1 Performance

For testing the middleware systems' performance the first metric, described at chapter 3.2.2, is used. This metric returns the total execution time for each middleware configuration per benchmark scenario. These values can be presented in a bar diagram:

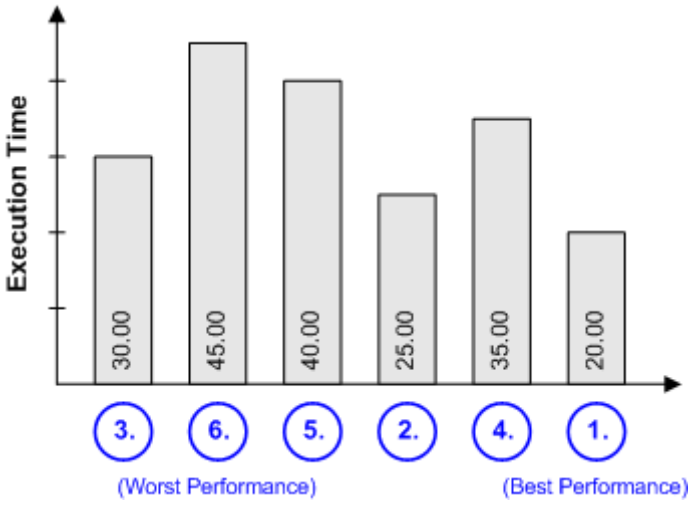


Figure 22: Performance Diagram in Summary

For the reason that not only the total execution time is determined, but also intermediate measurement points are logged, execution time in relation to the number of operations can be illustrated in a diagram, too:

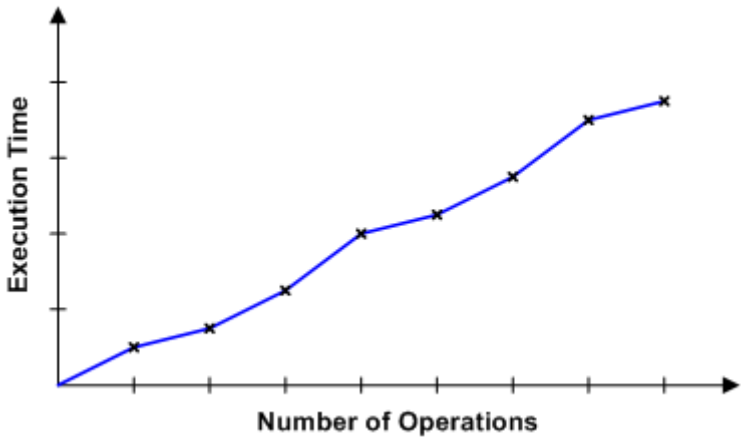


Figure 23: Performance Diagram in Detail

4.1.5.2 Scalability

For testing the middleware systems' scalability the metric, described in chapter 3.3.3.1, is used with the following adaptions:

- The computing power of a system is not only defined by the number of processors, but instead a general approach is introduced: Now, the n-fold computing power is realized by n-fold resources, and not only by n-fold processors.
- Furthermore, the ratio between the time required for executing a problem on a machine and the time required for executing the quasi-same problem, but with n-fold problem size, on the quasi-same machine, but with n-fold resources, returns the efficiency of a system. This context can be expressed in formulas as follows:

  $$\text{efficiency}(n, x) = \text{time}(1, x) / \text{time}(n, n \cdot x)$$

A benchmark scenario, which targets on scalability investigations, executes each test case three times, namely with 1-fold, 2-fold and 3-fold problem size using the corresponding machine configurations described above. This results in three values for the execution time, which are ideally equal and can be displayed side by side in form of bars. Such a diagram enables the user to estimate the scalability of the systems at a glance.

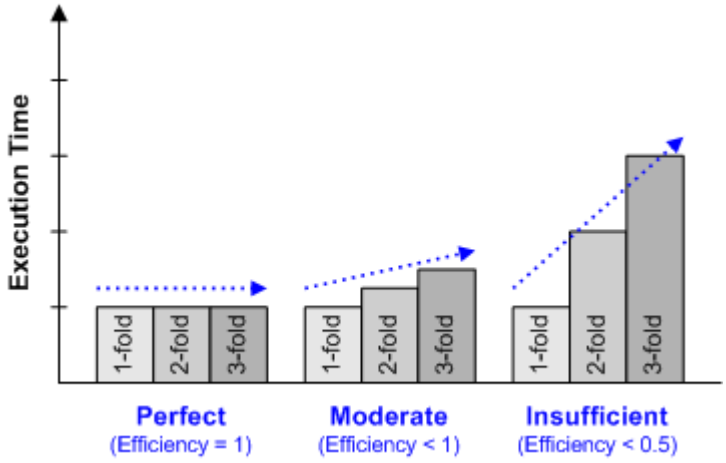


Figure 24: Scalability Diagram in Detail

The average value of the efficiencies gives also information about a system's scalability and enables the comparison between the middleware systems on the basis of absolute measures. It is calculated for the procedure described above in the following way:

$$efficiency(x) = (time(1, x) / time(2, 2 \cdot x) + time(1, x) / time(3, 3 \cdot x)) / 2$$

However, this average value should be taken with a pinch of salt. Especially, in case that the efficiency(3, x) is much smaller than the efficiency(2, x), there is a risk of exponential growth. So, the value shows a trend of the system's scalability, but that is it.

All calculated average values can be presented together in the form of a bar diagram. Such a chart allows to compare the systems' scalability at a glance:

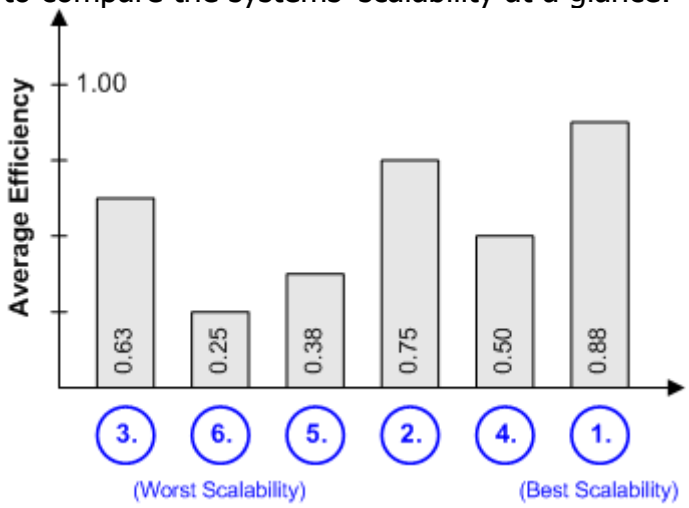


Figure 25: Scalability Diagram in Summary

4.1.6 Operations

The scenarios' descriptions use synonyms for the operations to simplify matters. The following table gives a survey of the relations between the operations:

Synonym	**XVSM** (MozartSpaces)	**JavaSpaces** (GigaSpaces XAP)	**J2EE** (JBoss AS)
write	write	write	create
shift	shift	–	–

read	read	read readIfExists	find
take	take	take takeIfExists	–
destroy	destroy	–	remove

All operations are executed using entries as parameters, which are composed of a string with 10 characters. In Java, this fact corresponds with String(10).

4.2 Serial Benchmarks

First the operations will be executed in series and the execution time will be measured. The serial benchmarks rate the performance only. All scenarios will be carried out without explicit transactions and use implicit transactions or none, if supported, instead. The containers or rather spaces will be accessed both embedded, if supported, and remotely.

The following table gives a survey of the middleware configurations which are tested by the serial benchmarks:

Middleware	Container / Space	Persistence	Transaction	Coordinator / Selector
XVSM (MozartSpaces)	embedded	transient	implicit	Random
				FIFO
				LIFO
				Key
				Vector
				Linda
	remote			Random
				FIFO
				LIFO
				Key
				Vector
				Linda
JavaSpaces (GigaSpaces XAP)	embedded	transient	none	–
		persistent		
	remote	transient		
		persistent		
J2EE (JBoss AS)	remote	persistent	none	–

Note that the parameters used for executing the serial benchmark scenarios via the benchmark suite are mentioned in detail in chapter 8.1.1 of the appendix.

4.2.1 Write

4.2.1.1 Scenario

60 iterations, each with 1K write operations, are performed. Altogether, 60K write operations are executed on the container or rather space. Since no records are deleted, the container or rather space is filled with more and more entries. The monitor is not reset between the passes. This means that the durations are summed up and that the last value represents the execution time for 60K write operations in series. The maximum number of entries which can be written into the container is not limited.

Iteration	Entries	Operations	Entries

	(before)	(measured)	(after)
1	0	1K write	1K
2	1K	1K write	2K
...	...	...	...
n	1K · (n – 1)	1K write	1K · n
...	...	...	...
59	58K	1K write	59K
60	59K	1K write	60K
TOTAL		**60K write**	

4.2.1.2 Expectance

Writing new entries into the container or rather space will be done with relatively constant execution time for each pass. Therefore, the visualization of the measured data will show a straight line:

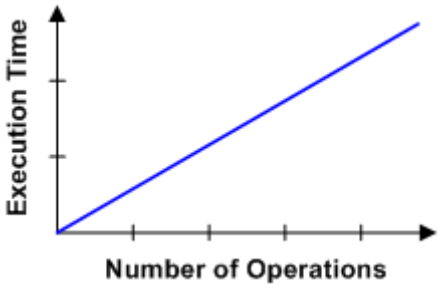


Figure 26: Expected Performance Diagram of Serial/Write Scenario

4.2.1.3 Results

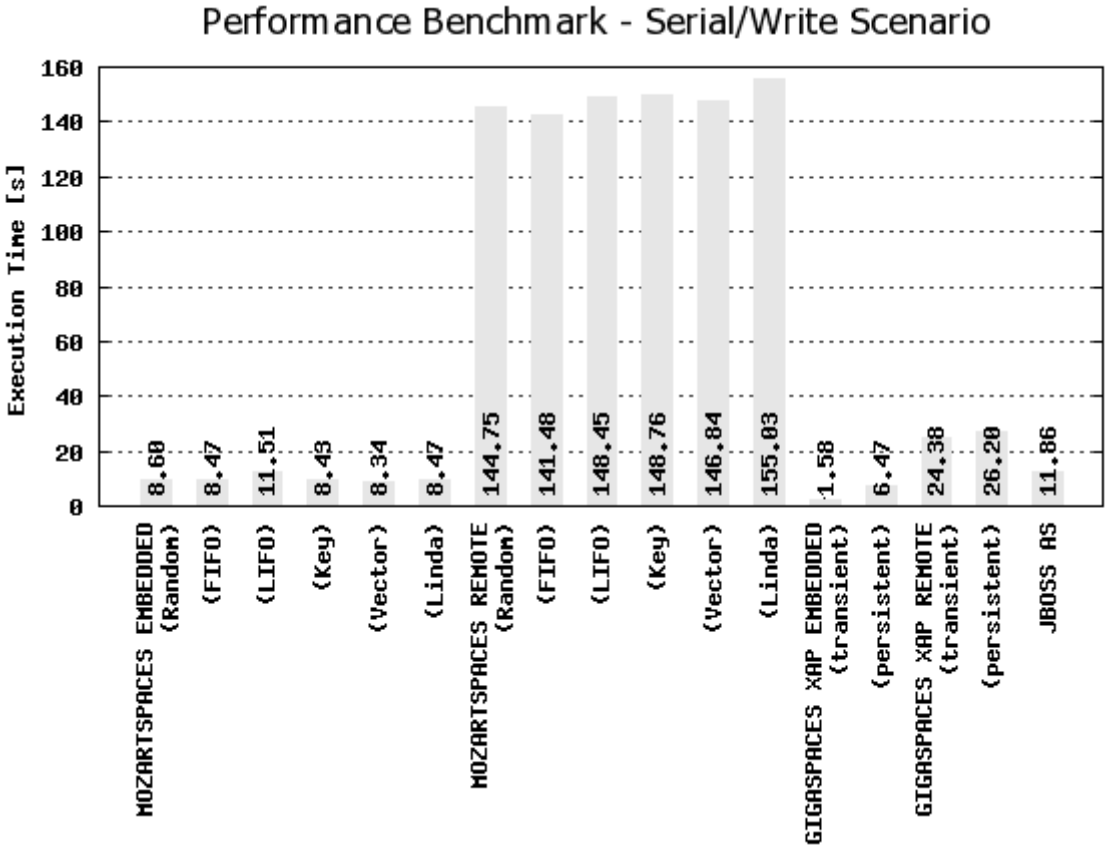


Figure 27: Performance Diagram in Summary of Serial/Write Scenario

MozartSpaces shows mean performance for writing entries into an embedded container, but poor performance in connection with a remote container.

By contrast, GigaSpaces XAP impresses with excellent write performance. Writing entries into an embedded space with transient data management is executed very fast. Interestingly, the used type of persistence looses ground if a remote space is accessed.

JBoss AS also shows an excellent performance, given that this middleware system is accessed remotely and the entries are made durable persistent in a database.

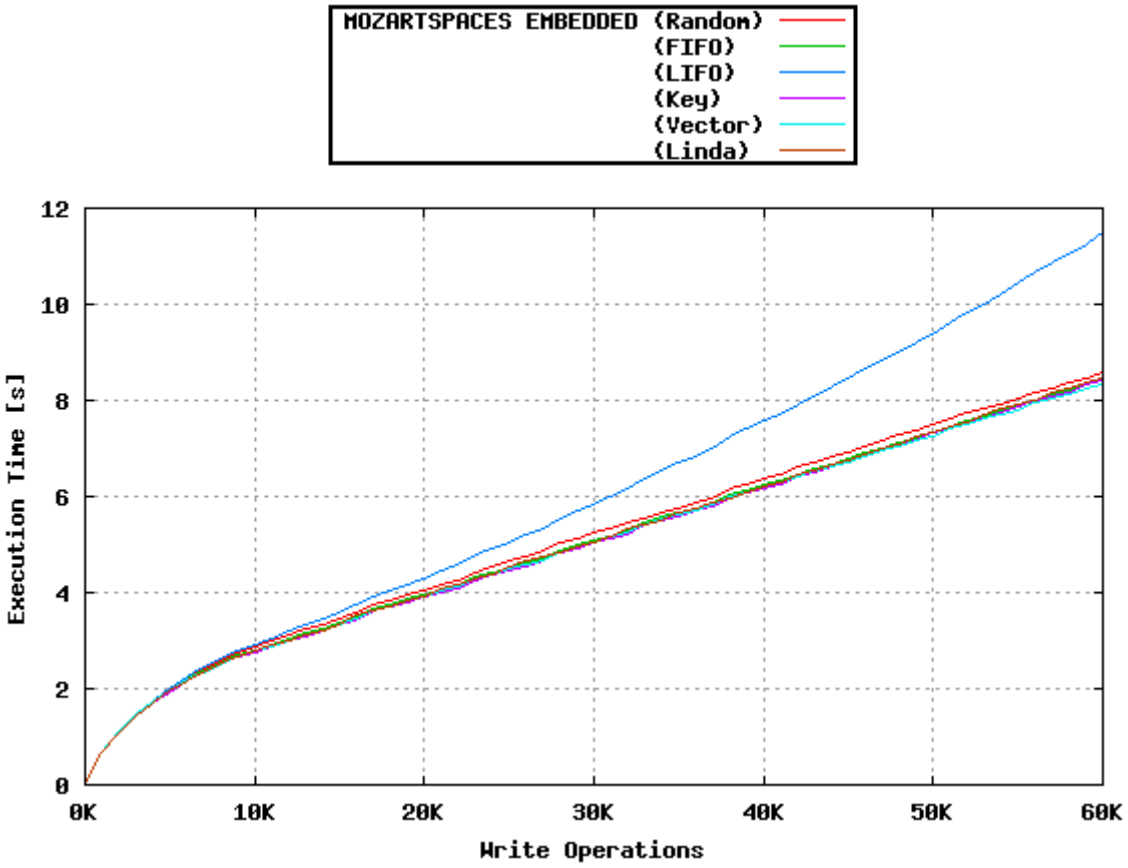


Figure 28: Performance Diagram in Detail of Serial/Write Scenario (1)

The progressions of the performance curves are nearly the same, except for the LIFO coordinator. MozartSpaces needs approximately 7.5K write operations, until it reaches its performance optimum. After that, the write performance is nearly linear.

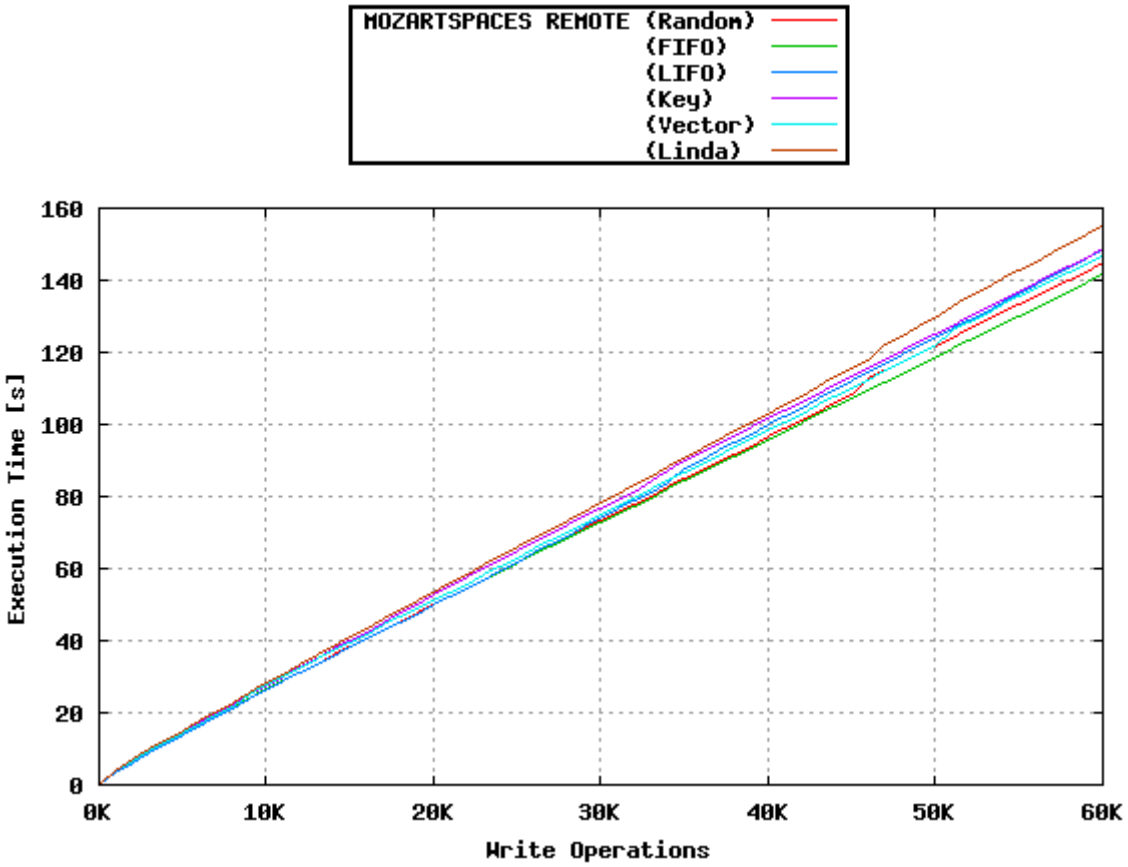


Figure 29: Performance Diagram in Detail of Serial/Write Scenario (2)

The use of a remote container in connection with MozartSpaces results in curve linearities. The best performance is shown by the FIFO coordinator, the worst by the Linda coordinator.

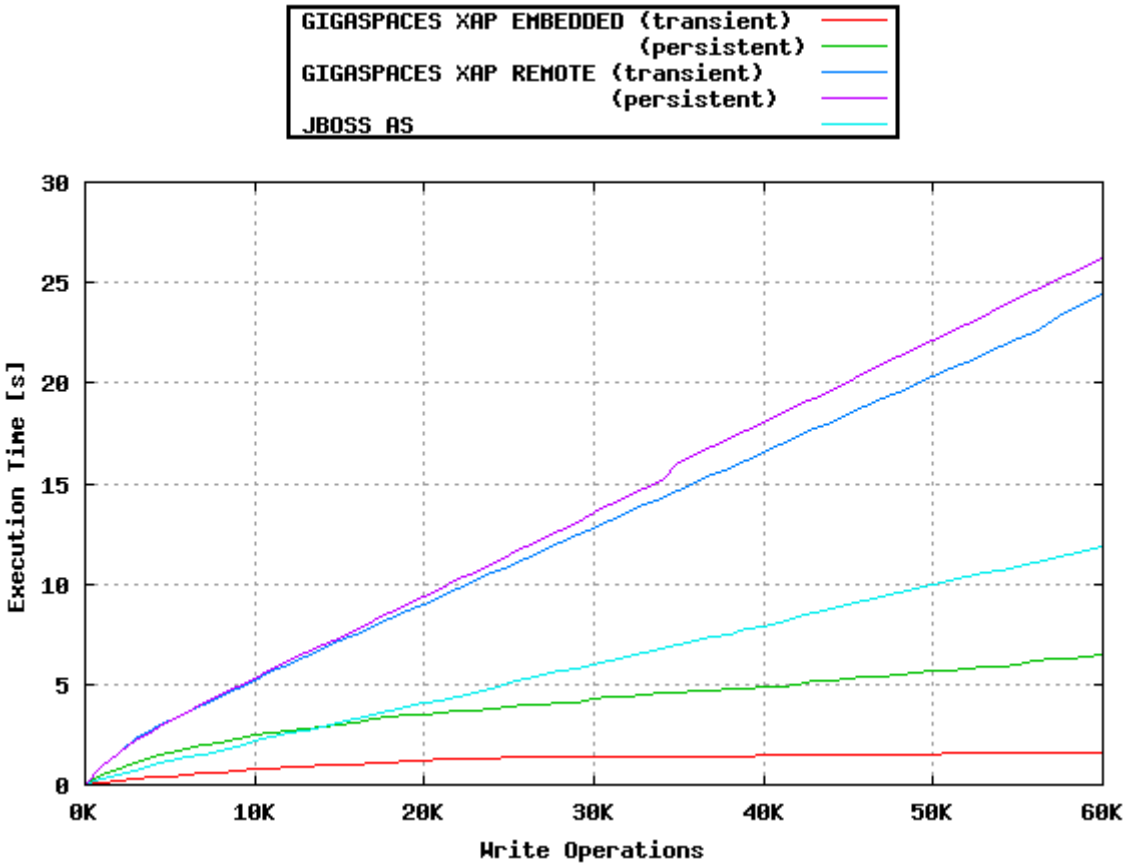


Figure 30: Performance Diagram in Detail of Serial/Write Scenario (3)

For these benchmarked systems entries are written into a space or rather container with linear performance as well. In particular, JBoss AS shows an absolute linear curve progression. This diagram demonstrates also, that both systems reach their operating maximum faster than MozartSpaces.

4.2.2 Shift

4.2.2.1 Scenario

This scenario corresponds with the previous one, but now shift operations instead of write operations are performed. In addition, the container is limited to 1K entries. This implicates that the number of entries in the container remains constant.

Iteration	**Entries** (before)	**Operations** (measured)	**Entries** (after)
1	0	1K shift	1K
2	1K	1K shift	1K
...	...	...	...
n	1K	1K shift	1K
...	...	...	...
59	1K	1K shift	1K
60	1K	1K shift	1K
TOTAL		**60K shift**	

JavaSpaces (GigaSpaces XAP) and J2EE (JBoss AS) do not support shift operations, which is why this benchmark scenario cannot be executed with these technologies.

4.2.2.2 Expectance

The execution time for shifting new entries will also be constant, which results in a straight line as graph:

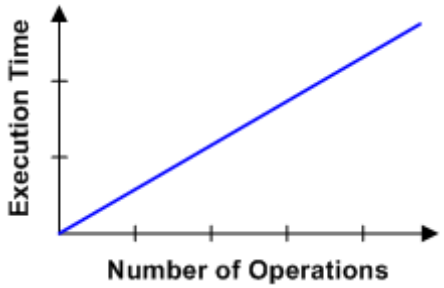


Figure 31: Expected Performance Diagram of Serial/Shift Scenario

4.2.2.3 Results

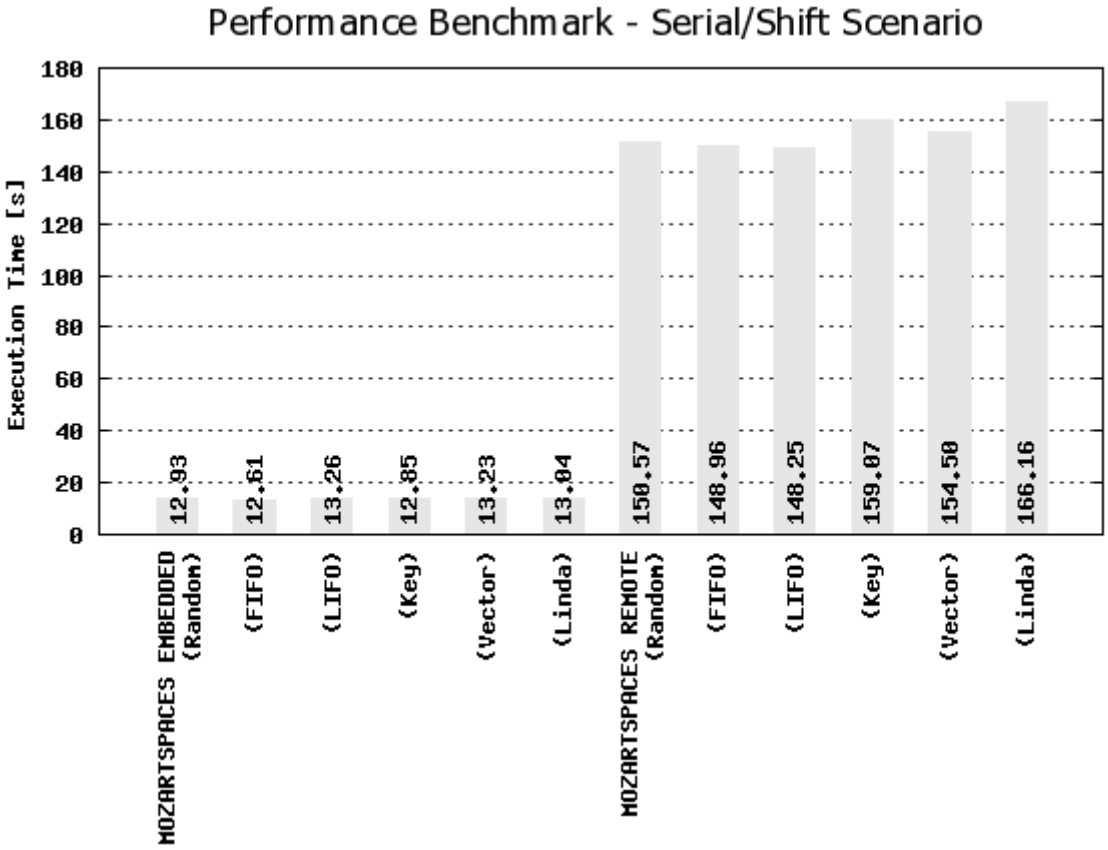


Figure 32: Performance Diagram in Summary of Serial/Shift Scenario

It is no surprise that shifting entries with the same coordinator consumes more time than writing entries into the container. The reason for this is that in addition to the data insertion, often existing data must be accessed and destroyed.

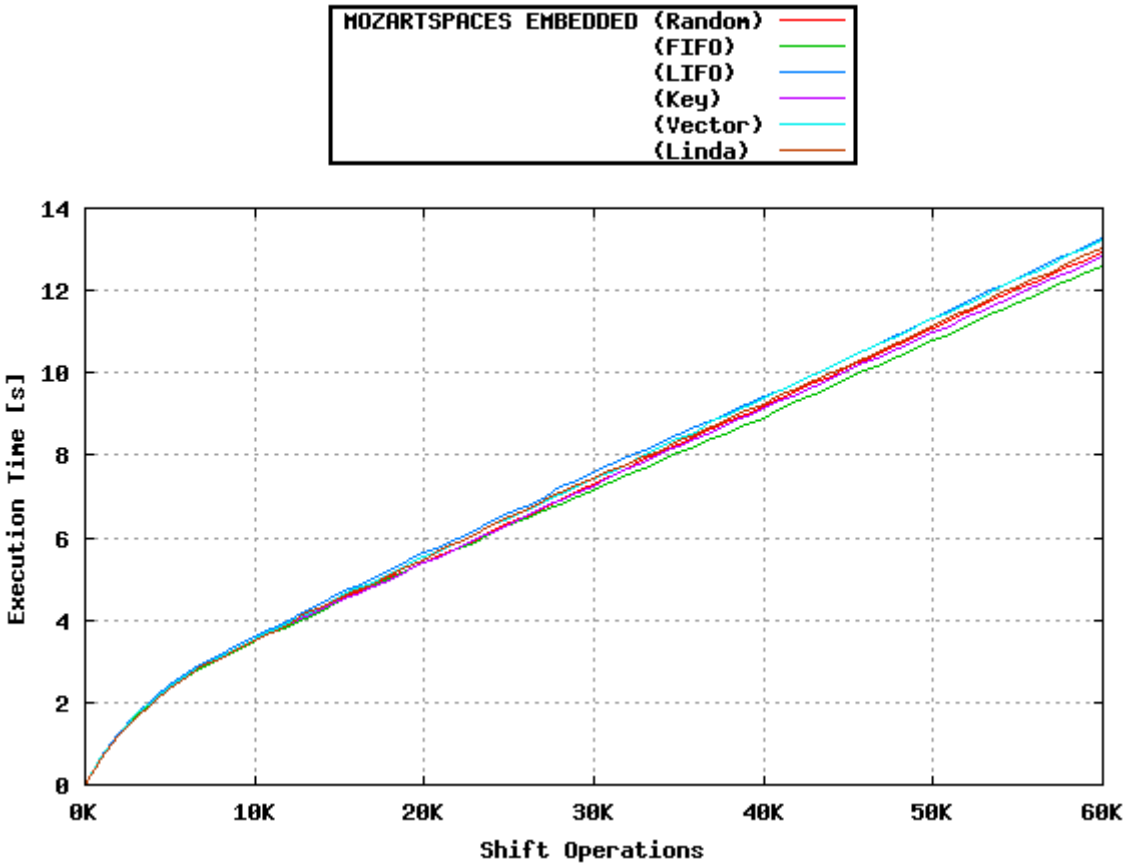


Figure 33: Performance Diagram in Detail of Serial/Shift Scenario (1)

The curve progressions for shifting entries are nearly the same as for writing entries.

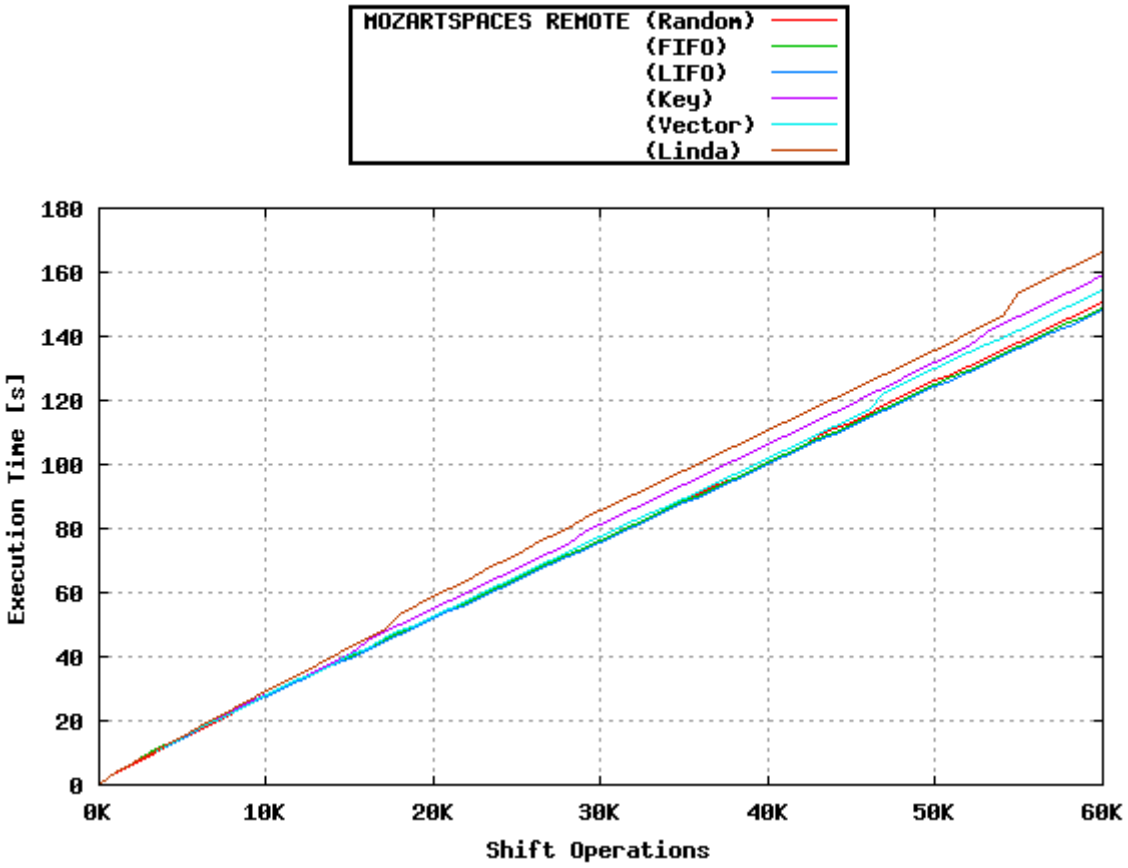


Figure 34: Performance Diagram in Detail of Serial/Shift Scenario (2)

The performance diagram for the shift operations on a remote container also corresponds with the related write diagram above. Again, the FIFO coordinator shows the best and the Linda coordinator shows the worst performance.

4.2.3 Read

4.2.3.1 Scenario

There are 60 iterations, each with 1K write and 1K read operations. The monitor measures the time for the read operations only and is not reset. So, writing entries to the container or rather space has the purpose of test preparation only. It increases the number of entries in the container or rather space for each pass. The last value provides the duration needed for the execution of 60K read operations in series. A container limit is not set.

Iteration	**Entries** (before)	**Operations** (preparatory)	**Entries** (after/before)	**Operations** (measured)	**Entries** (after)
1	0	1K write	1K	1K read	1K
2	1K	1K write	2K	1K read	2K
…	…	…	…	…	…
n	1K · (n – 1)	1K write	1K · n	1K read	1K · n
…	…	…	…	…	…
59	58K	1K write	59K	1K read	59K
60	59K	1K write	60K	1K read	60K
TOTAL				**60K read**	

4.2.3.2 Expectance

The execution time will increase with each pass in consequence of filling the container or rather space with more and more entries. The visualized data will show a curve that slightly grows upwards:

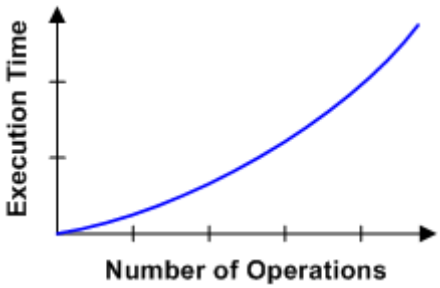


Figure 35: Expected Performance Diagram of Serial/Read Scenario

4.2.3.3 Results

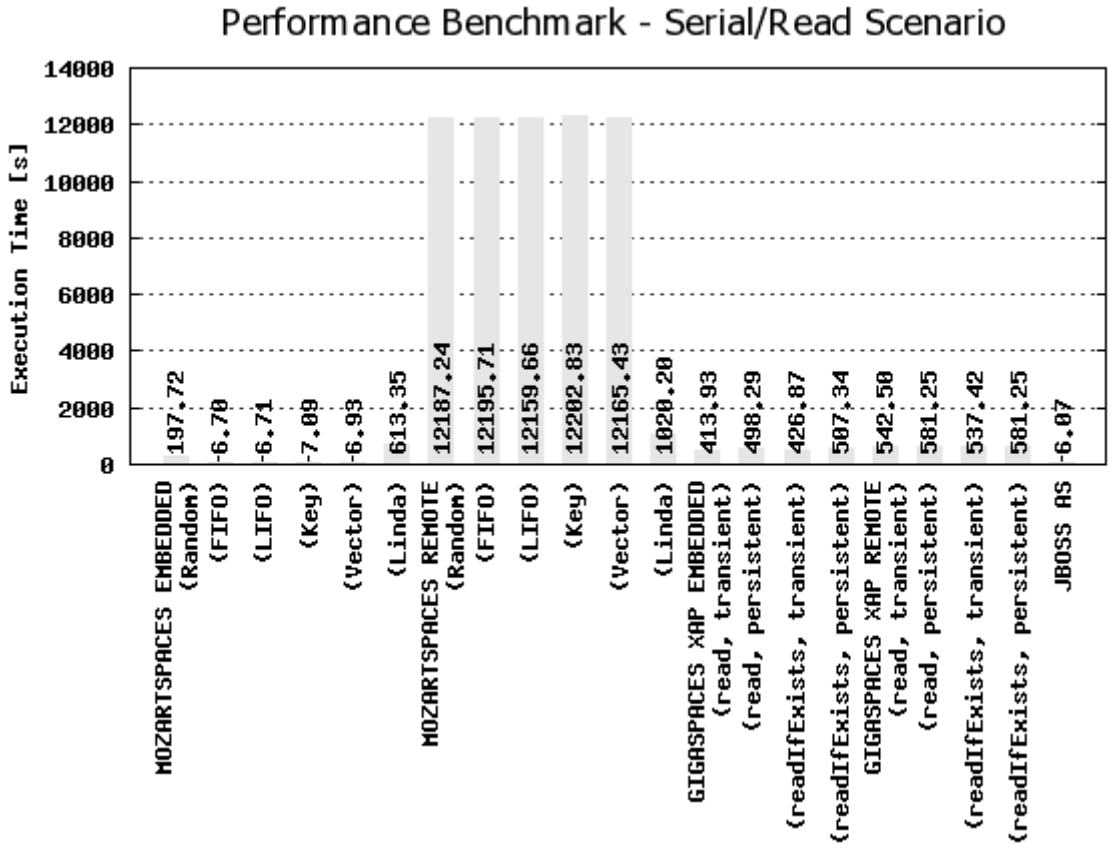


Figure 36: Performance Diagram in Summary of Serial/Read Scenario

In MozartSpaces, the performance of the read operations depends highly on the used selector. Interestingly, in comparison with the other selectors the Linda selector has worst performance with an embedded container, but best performance with a remote one.

The benchmarks show also, that the execution times for read and readIfExists are very similar in GigaSpaces XAP.

JBoss AS impresses with superb performance compared to its competitors.

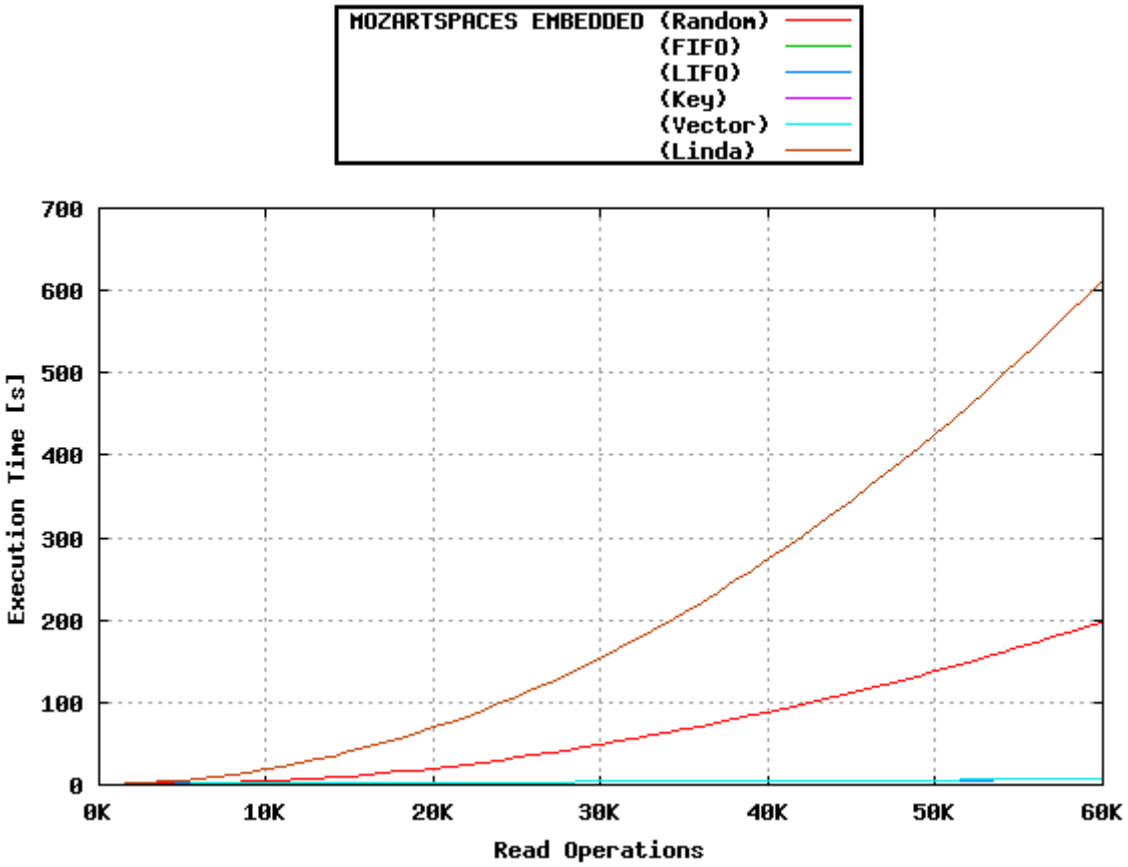


Figure 37: Performance Diagram in Detail of Serial/Read Scenario (1)

As it was expected before the implementation of the benchmarks, reading entries from an embedded container takes place, only with the Linda and the Random selector. The other selectors offer linear progression with respect to their performance. Due to the fact that the read operations with Linda selector are based on pattern matching, the execution time is fundamentally greater than with the other selectors.

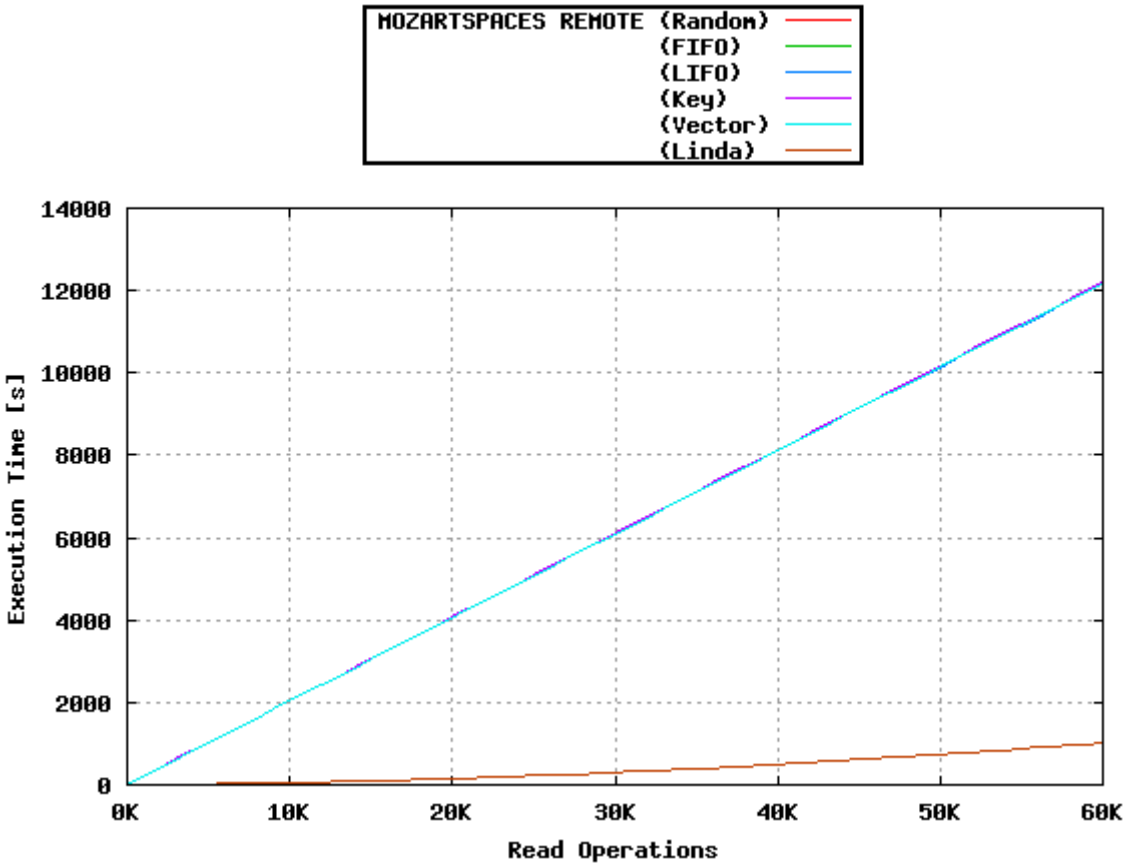


Figure 38: Performance Diagram in Detail of Serial/Read Scenario (2)

On the other hand, if the entries are read from a remote container, the Linda selector offers the best performance by far. This results from more efficient implementation of this selector. Reading entries from a remote container with another selector than Linda leads to extremely poor performance. Due to the strict curve linearity, it can be assumed, that the cause of this insufficiency is not the real read access, but most probably it is the transaction and communication overhead.

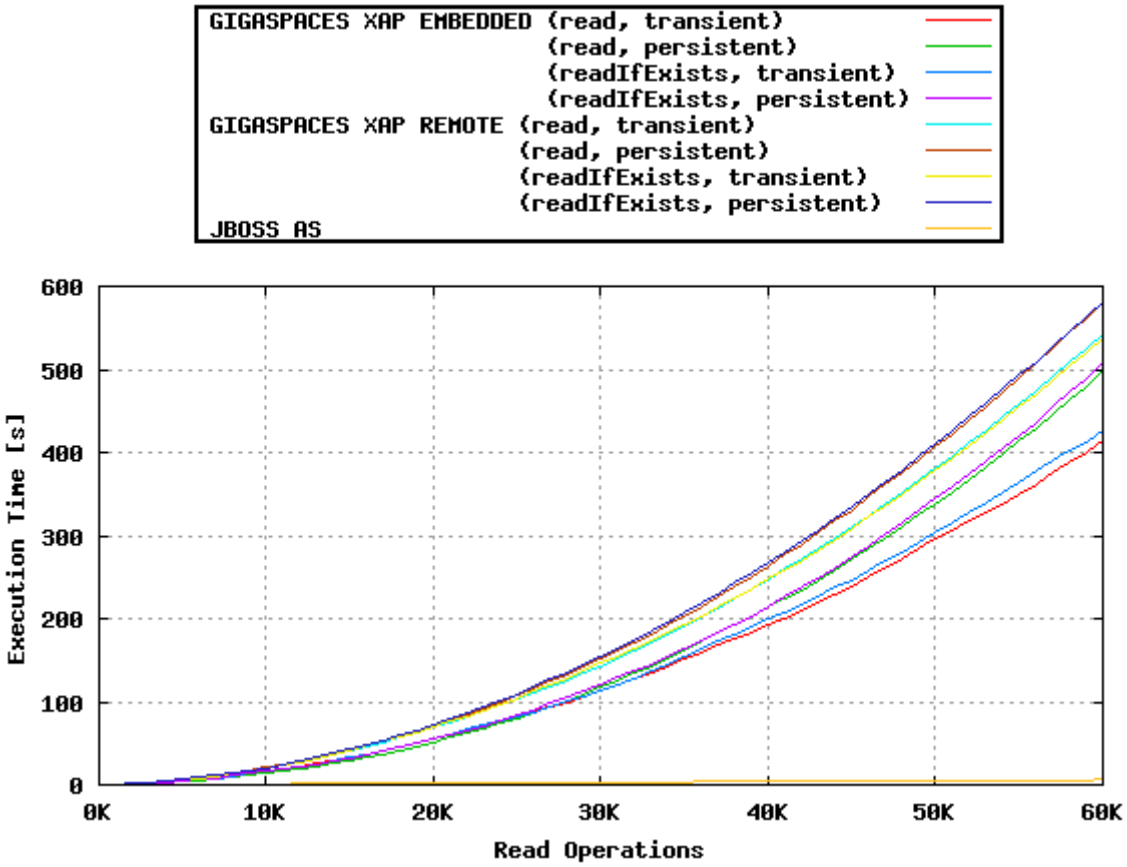


Figure 39: Performance Diagram in Detail of Serial/Read Scenario (3)

For reading entries from a space of GigaSpaces XAP the benchmark results present the curve progressions as expected. The pairwise arrangement of the graphs for the read and the corresponding readIfExists operation can be clearly recognized.

JBoss AS plays out its strength, namely that the business logic is located and executed in the middleware system itself. Thereby, the communication overhead is reduced to a minimum, which results in excellent performance.

4.2.4 Take

4.2.4.1 Scenario

The container or rather space is filled with 60K entries initially. Afterwards, 60 iterations are performed, each with 1K take operations. In the wake of that the container or rather space becomes emptier and emptier as time goes by. The monitor is not reset between the passes. Therefore, the last value represents the duration for executing 60K take operations in series. A limit for the maximum number of entries in the container is not set.

Iteration	**Entries** (before)	**Operations** (measured)	**Entries** (after)
1	60K	1K take	59K
2	59K	1K take	58K
...	...	...	...
n	1K · (61 – n)	1K take	1K · (60 – n)
...	...	...	...
59	2K	1K take	1K

60	1K	1K take	0
TOTAL		**60K take**	

J2EE (JBoss AS) does not support take operations, which is why this benchmark scenario cannot be executed with this technology.

4.2.4.2 Expectance

Since the number of entries in the container or rather space decreases with each pass, the execution time for the take operations will become shorter and shorter. This context, illustrated in a diagram, will evolve to a curve that slightly falls downwards in comparison with linear growth:

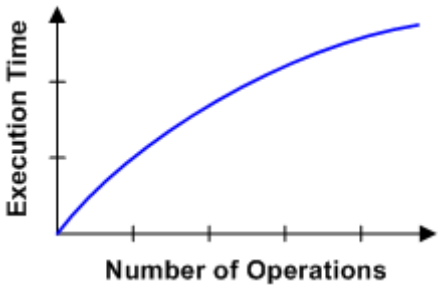


Figure 40: Expected Performance Diagram of Serial/Take Scenario

4.2.4.3 Results

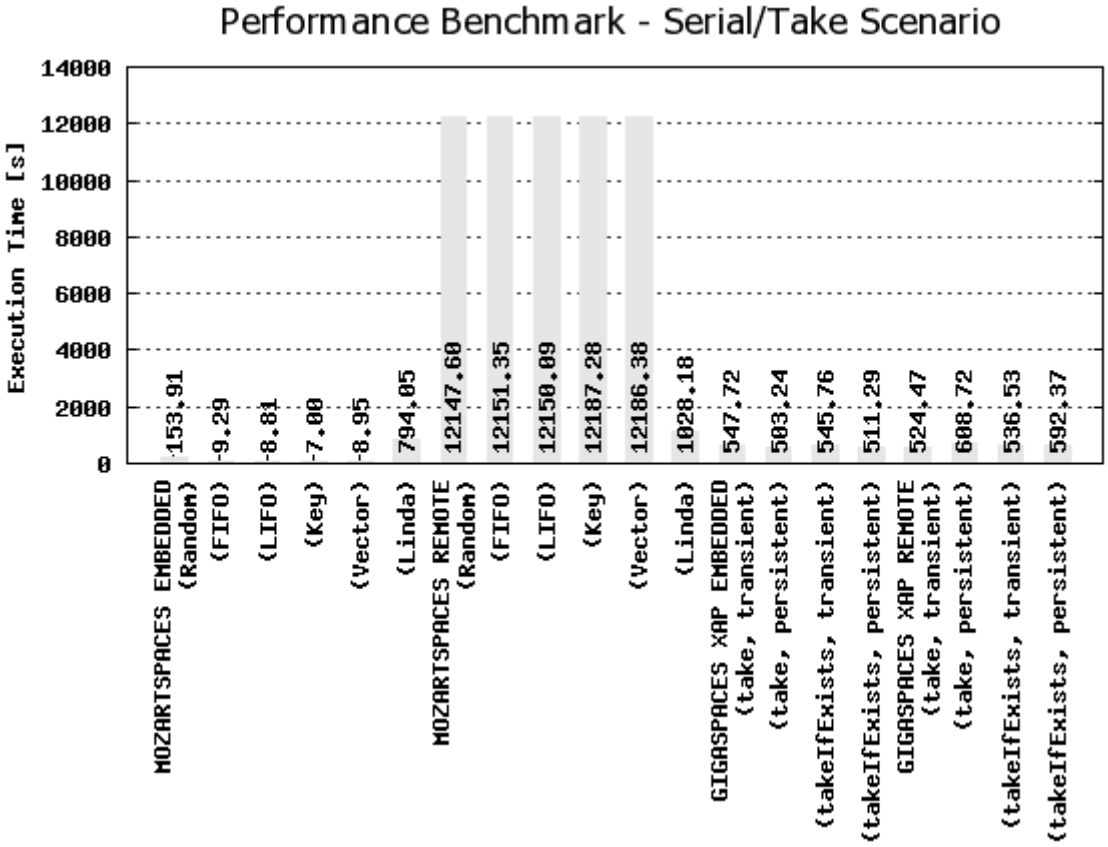


Figure 41: Performance Diagram in Summary of Serial/Take Scenario

The benchmark results for taking entries from the container or rather space are nearly the same as for reading entries. So, the interpretations above are also valid for this benchmark scenario and further comments can be omitted.

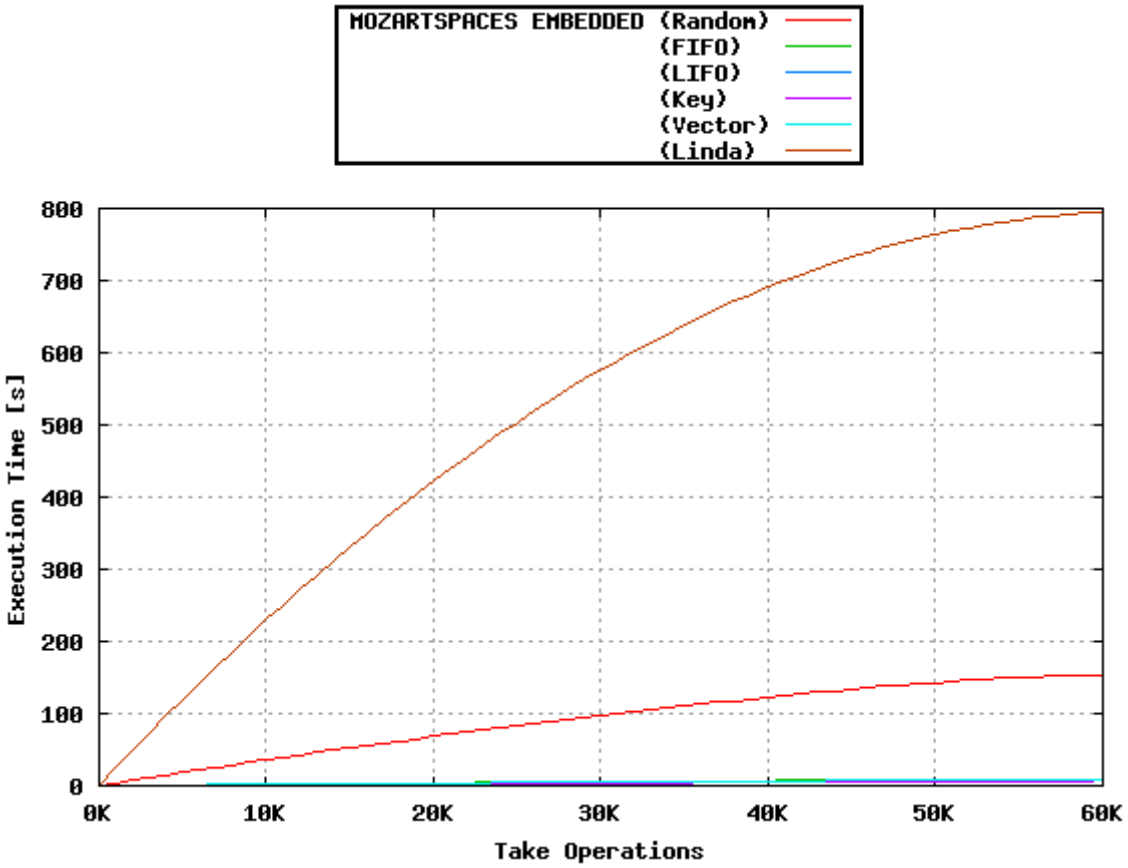


Figure 42: Performance Diagram in Detail of Serial/Take Scenario (1)

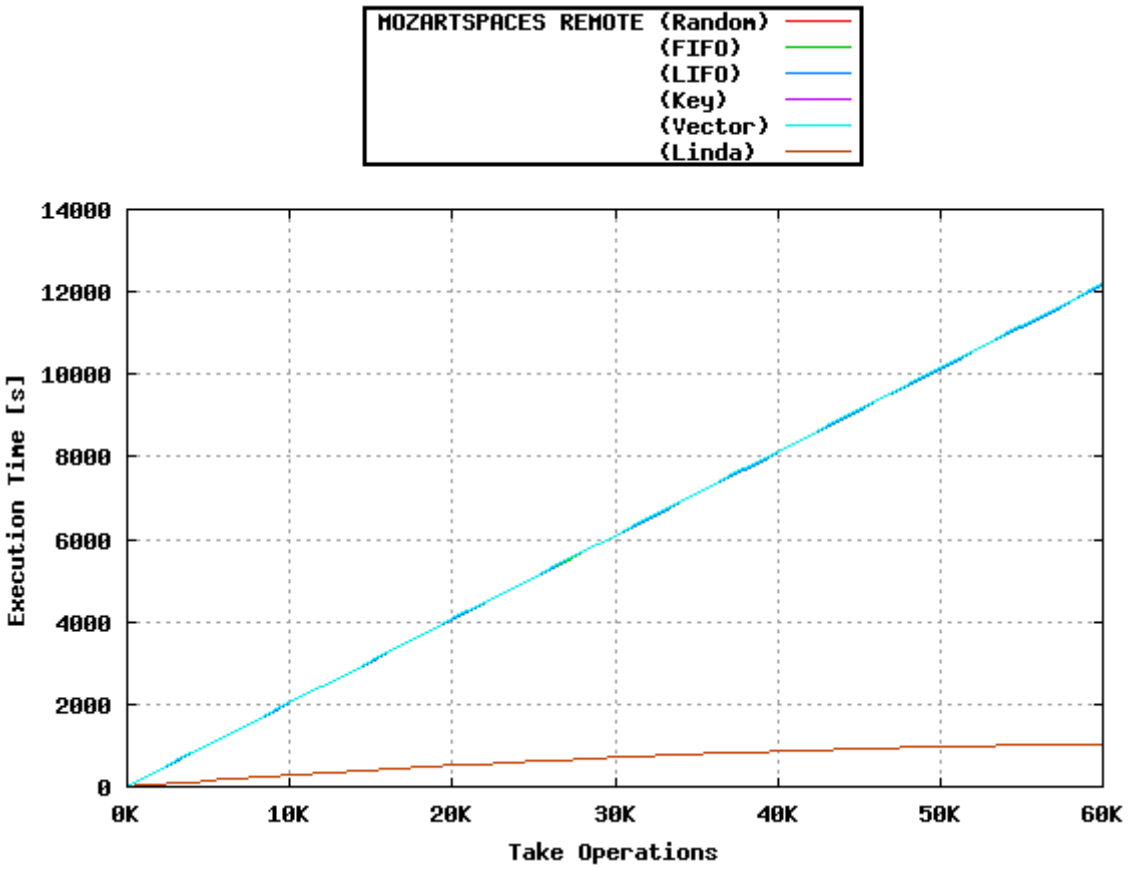


Figure 43: Performance Diagram in Detail of Serial/Take Scenario (2)

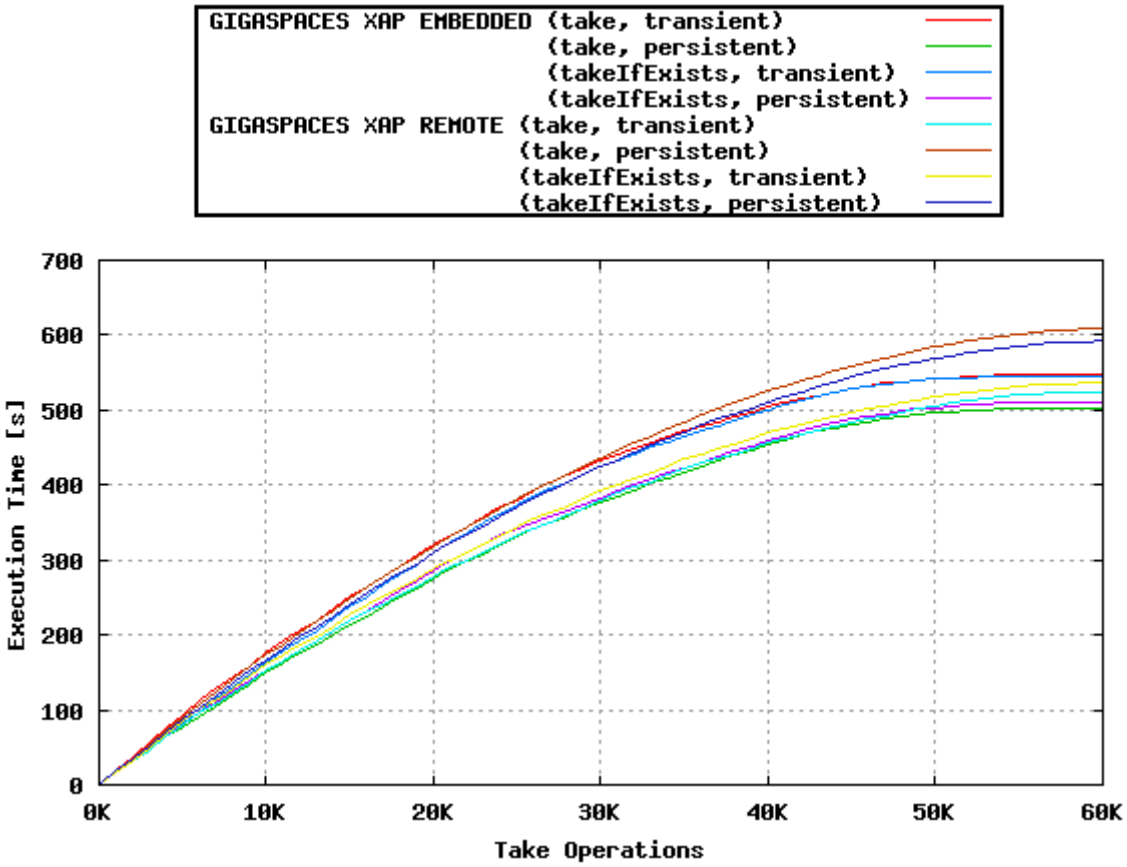


Figure 44: Performance Diagram in Detail of Serial/Take Scenario (3)

4.2.5 Destroy

4.2.5.1 Scenario

This scenario complies with the previous one, but it uses destroy operations instead of take operations.

Iteration	**Entries** (before)	**Operations** (measured)	**Entries** (after)
1	60K	1K destroy	59K
2	59K	1K destroy	58K
...	...	...	...
n	1K · (61 – n)	1K destroy	1K · (60 – n)
...	...	...	...
59	2K	1K destroy	1K
60	1K	1K destroy	0
TOTAL		**60K destroy**	

JavaSpaces (GigaSpaces XAP) does not support destroy operations, which is why this benchmark scenario cannot be executed with this technology.

4.2.5.2 Expectance

Here, with each iteration the execution time for the destroy operations will become shorter and shorter. The curve progression will be similar to the previous one.

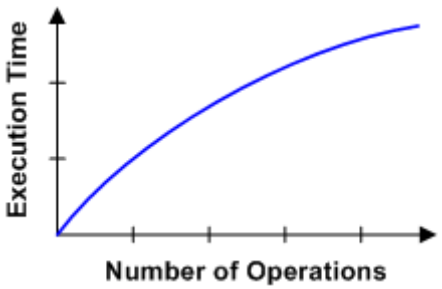


Figure 45: Expected Performance Diagram of Serial/Destroy Scenario

4.2.5.3 Results

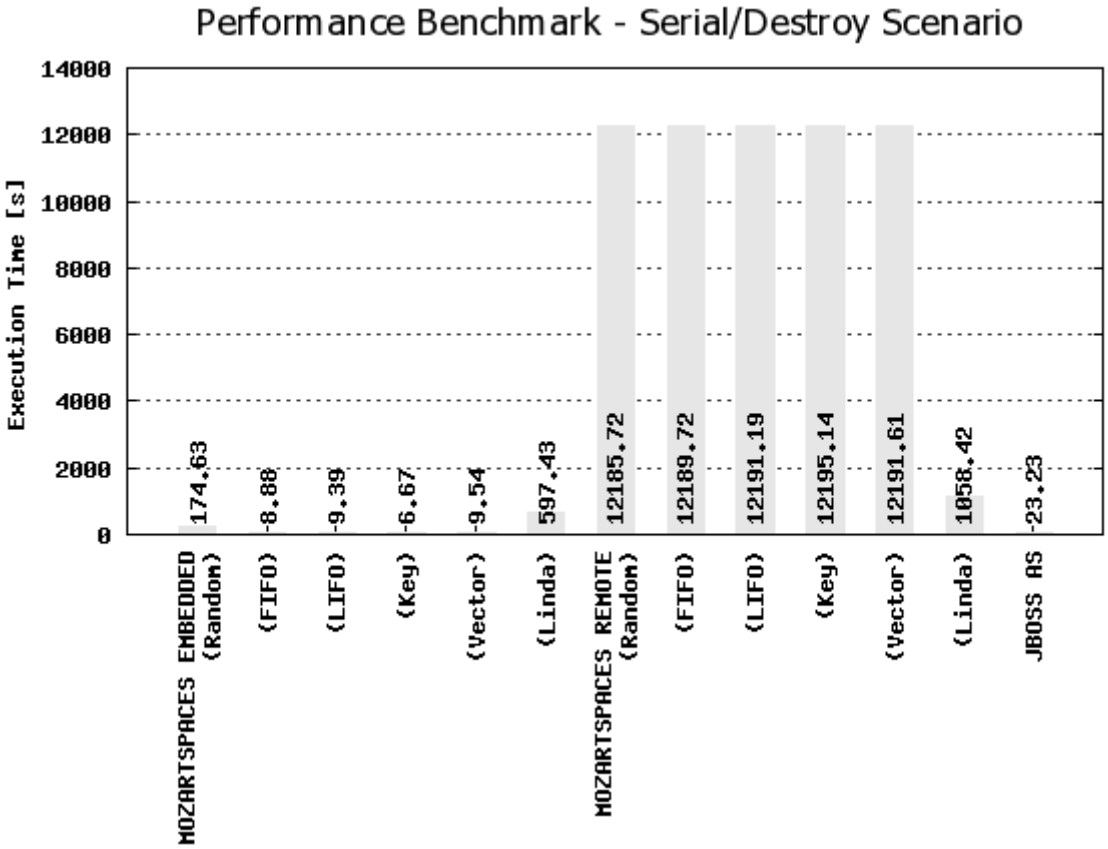


Figure 46: Performance Diagram in Summary of Serial/Destroy Scenario

The benchmark results for destroying entries are very similar to the measurements for reading and taking entries. This means that the interpretations above are valid for this benchmark scenario, too.

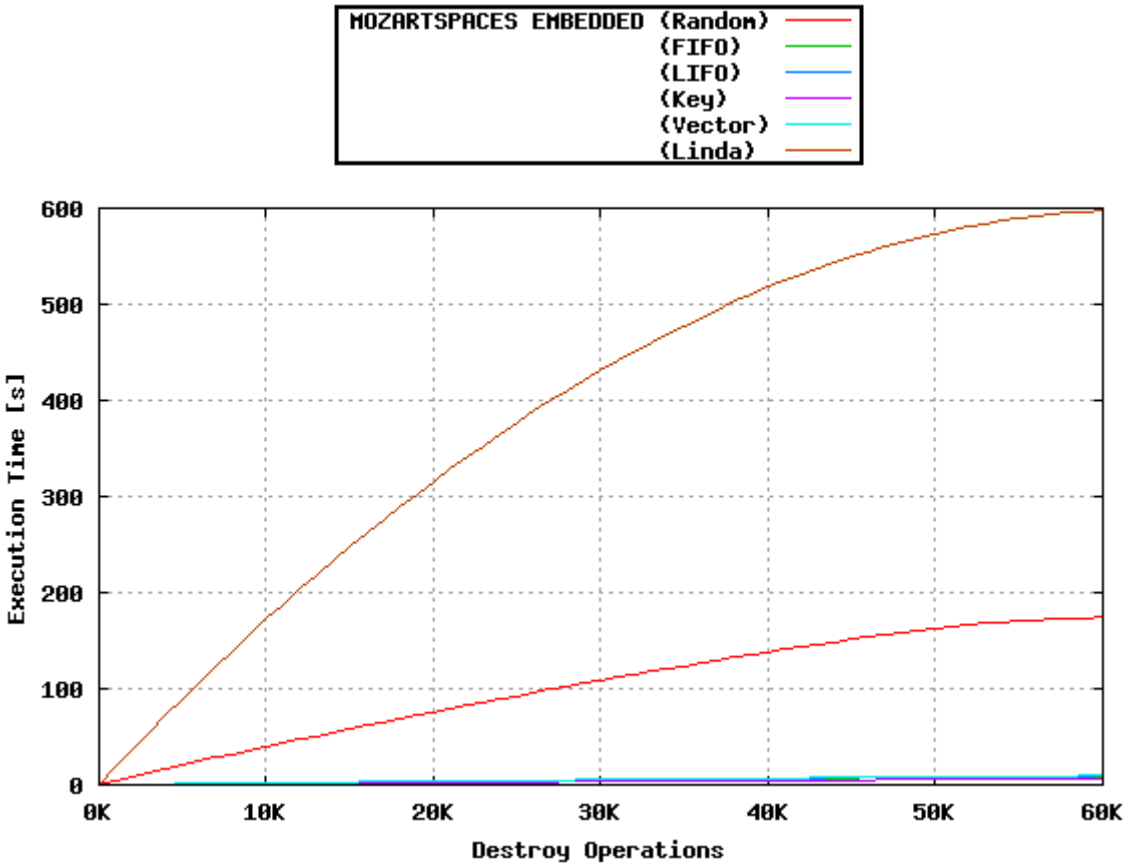


Figure 47: Performance Diagram in Detail of Serial/Destroy Scenario (1)

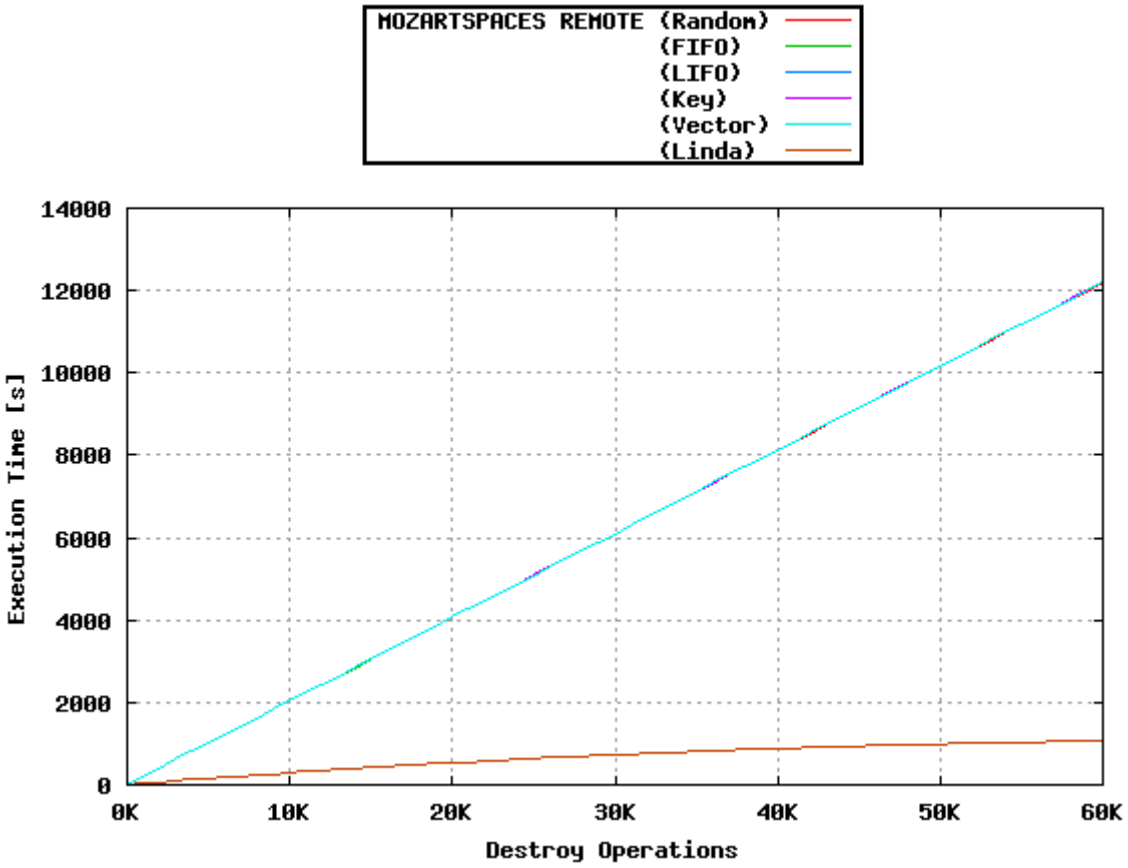


Figure 48: Performance Diagram in Detail of Serial/Destroy Scenario (2)

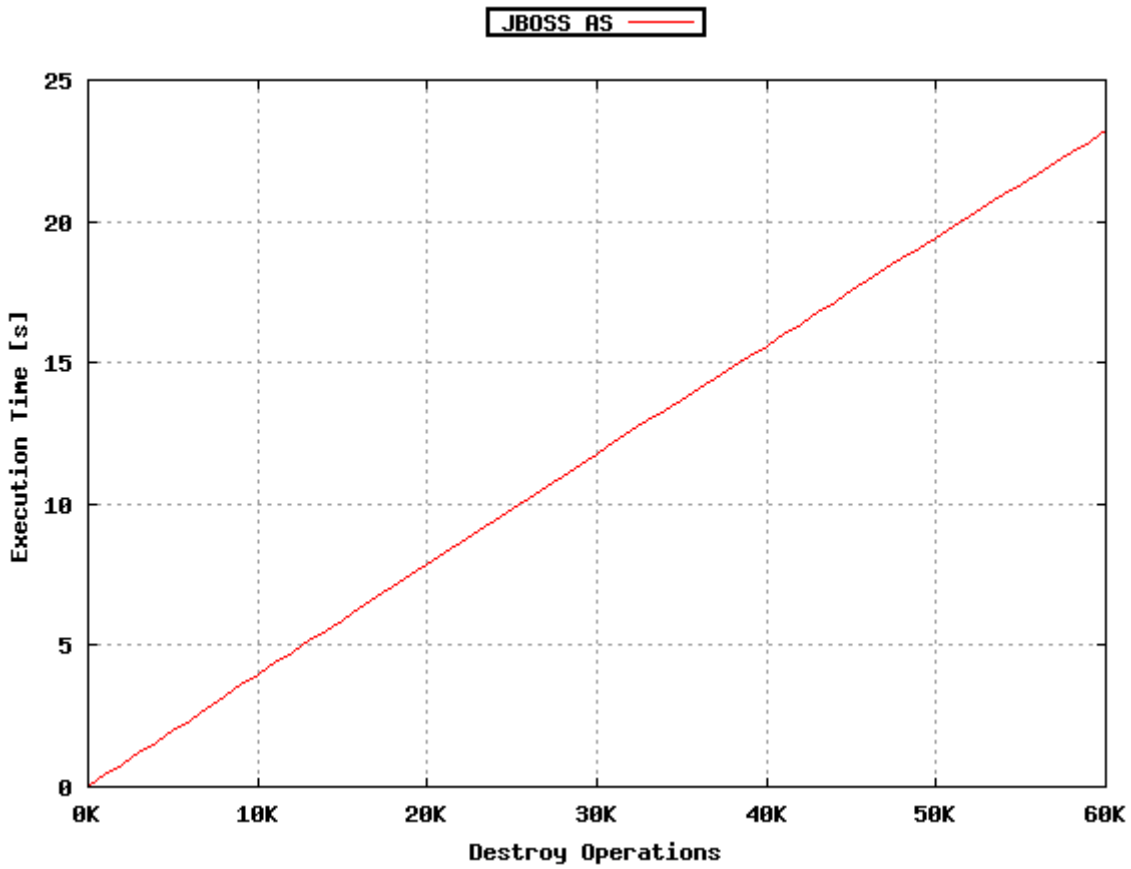


Figure 49: Performance Diagram in Detail of Serial/Destroy Scenario (3)

4.3 Concurrent Benchmarks

Next, the operations will be executed in parallel by several clients, simulated by threads, and the total time for all clients will be measured. The concurrent benchmarks investigate the scalability of the systems. Because of that each test case is executed in three different variants. Again, all scenarios forgo explicit transactions and, this time, all middleware systems are accessed in remote mode only.

The table below sums up the middleware configurations, which are tested by the concurrent benchmarks:

Middleware	Container / Space	Persistence	Transaction	Coordinator / Selector
XVSM (MozartSpaces)	remote	transient	implicit	Random
				FIFO
				LIFO
				Key
				Vector
				Linda
JavaSpaces (GigaSpaces XAP)	remote	transient	none	–
		persistent		
J2EE (JBoss AS)	remote	persistent	none	–

Note that the parameters used for executing the concurrent benchmark scenarios via the benchmark suite are mentioned in detail in chapter 8.1.1 of the appendix.

4.3.1 Write

4.3.1.1 Scenario

A defined number of threads is executed concurrently, each thread performs 1K write operations on the container or rather space. The hardware configuration, used for the implementation, depends on the variant of the test case. The monitor measures the total execution time, beginning with the first and ending with the last write operation. The maximum number of entries which can be written into the container is not limited.

Variant A → 1-fold Problem

20 threads perform a total of 20K write operations and the hardware configuration, referred to as Variant A → 1-fold Computing Power above, is used.

Thread	Entries (before)	Operations (measured)	Entries (after)
1	0	1K write	20K
2		1K write	
...		...	
n		1K write	
...		...	
19		1K write	
20		1K write	
TOTAL		**20K write**	

Variant B → 2-fold Problem

40 threads perform a total of 40K write operations and the hardware configuration, referred to as Variant B → 2-fold Computing Power above, is used.

Thread	Entries (before)	Operations (measured)	Entries (after)
1	0	1K write	40K
2		1K write	
...		...	
n		1K write	
...		...	
39		1K write	
40		1K write	
TOTAL		**40K write**	

Variant C → 3-fold Problem

60 threads perform a total of 60K write operations and the hardware configuration, referred to as Variant C → 3-fold Computing Power above, is used.

Thread	Entries (before)	Operations (measured)	Entries (after)
1	0	1K write	60K
2		1K write	
...		...	
n		1K write	
...		...	
59		1K write	
60		1K write	
TOTAL		**60K write**	

4.3.1.2 Expectance

The scalability of the systems will be moderate. Therefore, the visualization of the three measured execution times will result in the following chart:

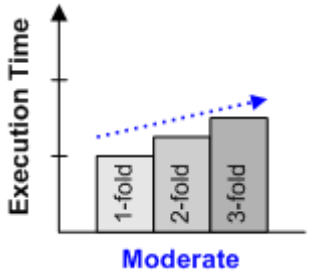


Figure 50: Expected Scalability Diagram of Concurrent/Write Scenario

4.3.1.3 Results

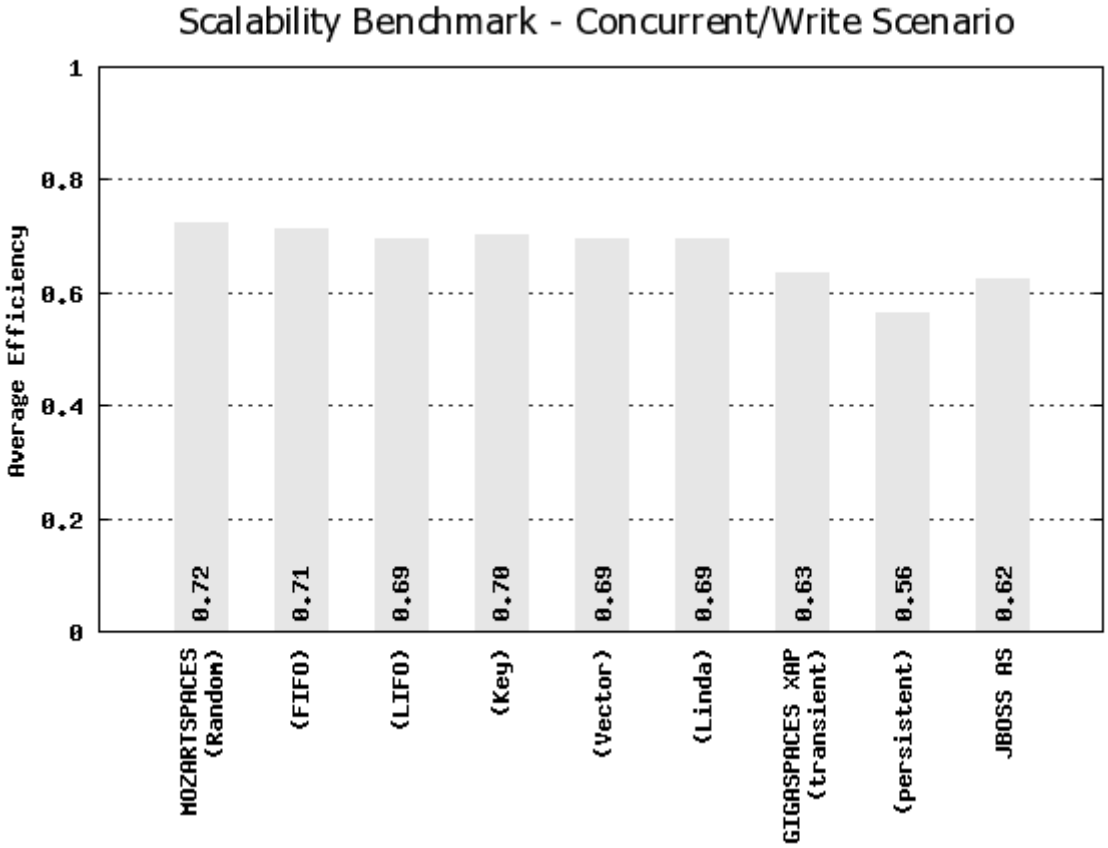


Figure 51: Scalability Diagram in Summary of Concurrent/Write Scenario

All middleware systems show moderate scalability regarding concurrent write operations.

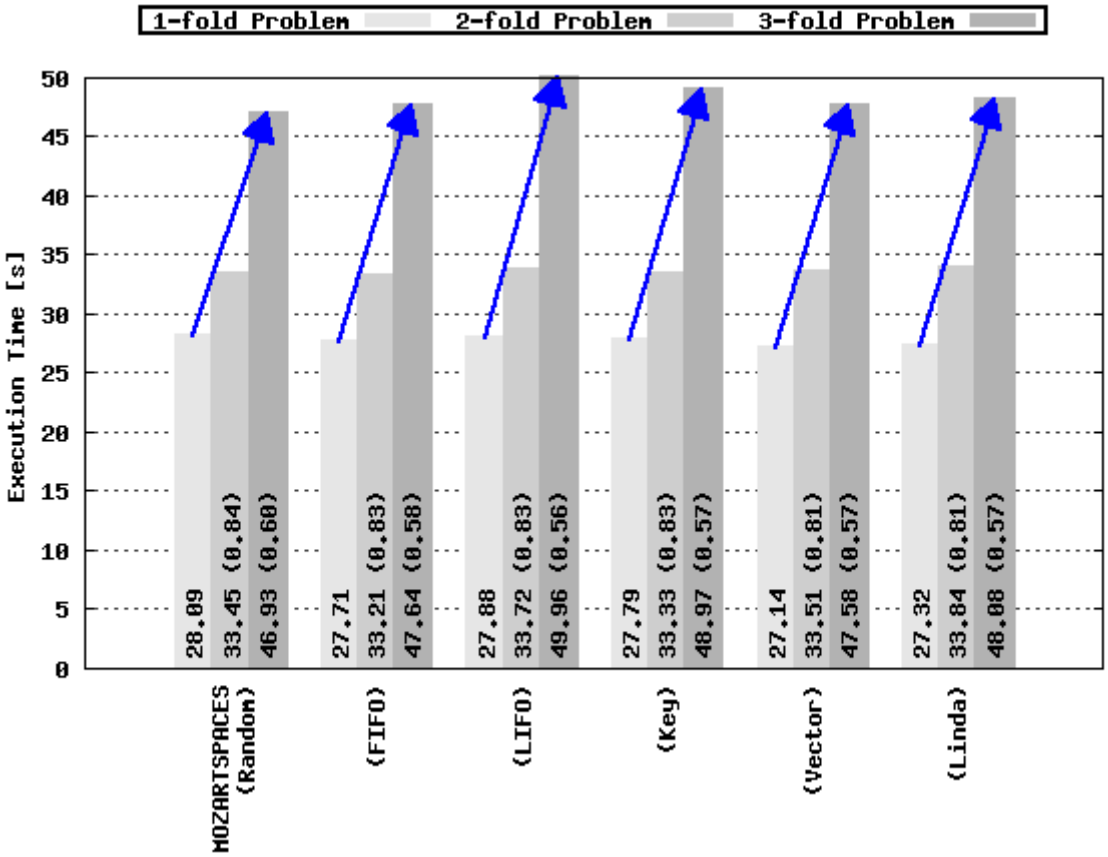


Figure 52: Scalability Diagram in Detail of Concurrent/Write Scenario (1)

The differences between the measured execution times and the calculated efficiencies are insignificant for all coordinators. Due to the fact that the efficiency values of the 3-fold problem are worse than of the 2-fold problem, non-linear scalability must be expected.

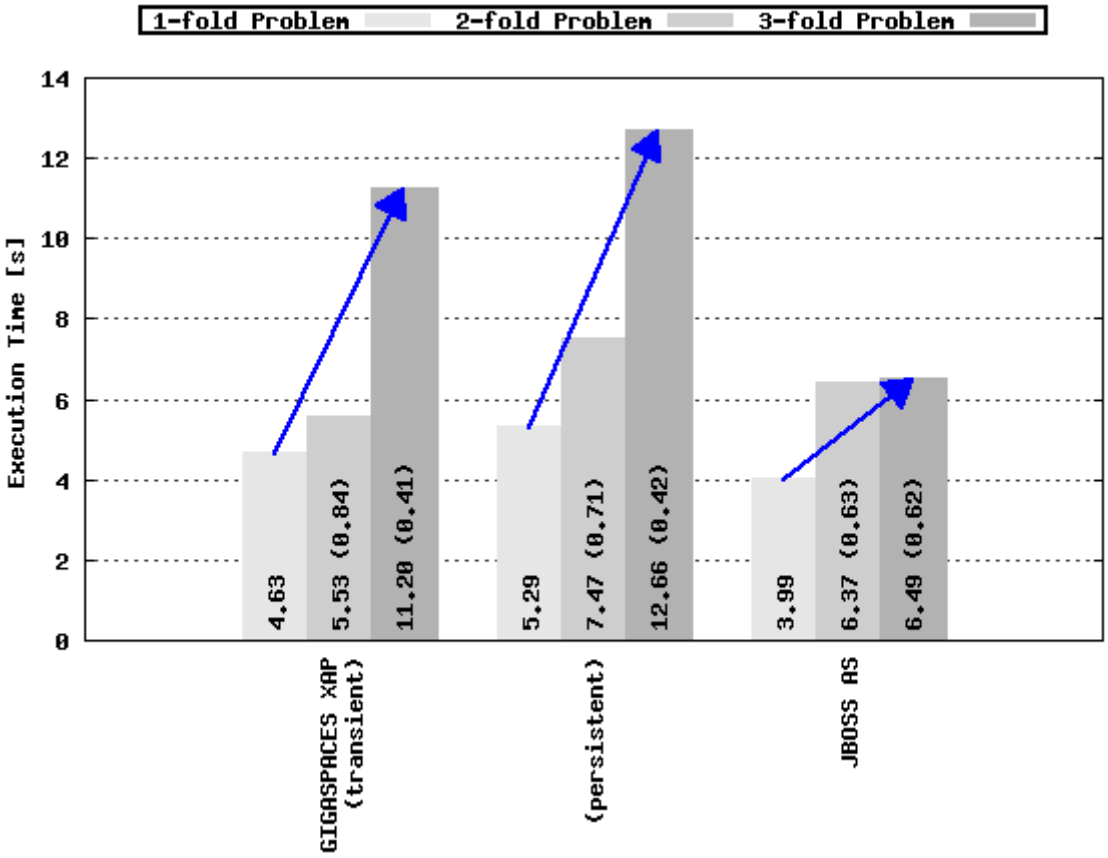


Figure 53: Scalability Diagram in Detail of Concurrent/Write Scenario (2)

Writing entries into a space of GigaSpaces XAP with 3-fold problem size results in efficiency values that stand for poor scalability. The analysis of this problem presents that the test design is inadequate for this test case, because the machine running the benchmark suite does not put enough pressure onto the middleware system. So, more than one test client machine would be necessary. However, this is the only test case with such a characteristic. For this reason the test design is not changed belated.

The calculated efficiencies for JBoss AS are akin, which signifies a constant, linear behaviour in matters of scalability.

4.3.2 Shift

4.3.2.1 Scenario

This scenario corresponds with the previous one, but now shift operations are performed. In addition, the container size is limited to 1K entries, which implicates that it must be shifted.

Variant A → 1-fold Problem

20 threads perform a total of 20K shift operations and the hardware configuration, referred to as Variant A → 1-fold Computing Power above, is used.

Thread	**Entries** (before)	**Operations** (measured)	**Entries** (after)
1	0	1K shift	1K
2		1K shift	
...		...	

n		1K shift	
...		...	
19		1K shift	
20		1K shift	
Total		**20K shift**	

Variant B → 2-fold Problem

40 threads perform a total of 40K shift operations and the hardware configuration, referred to as Variant B → 2-fold Computing Power above, is used.

Thread	**Entries** (before)	**Operations** (measured)	**Entries** (after)
1	0	1K shift	1K
2		1K shift	
...		...	
n		1K shift	
...		...	
39		1K shift	
40		1K shift	
TOTAL		**40K shift**	

Variant C → 3-fold Problem

60 threads perform a total of 60K shift operations and the hardware configuration, referred to as Variant C → 3-fold Computing Power above, is used.

Thread	**Entries** (before)	**Operations** (measured)	**Entries** (after)
1	0	1K shift	1K
2		1K shift	
...		...	
n		1K shift	
...		...	
59		1K shift	
60		1K shift	
TOTAL		**60K shift**	

JavaSpaces (GigaSpaces XAP) and J2EE (JBoss AS) do not support shift operations, which is why this benchmark scenario cannot be executed with these technologies.

4.3.2.2 Expectance

The scalability of the shift operations will be moderate, but better as for the previous scenario.

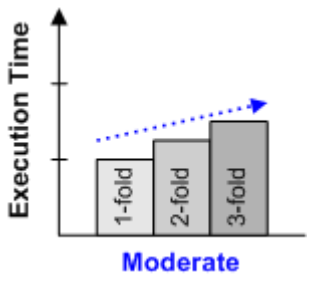


Figure 54: Expected Scalability Diagram of Concurrent/Shift Scenario

4.3.2.3 Results

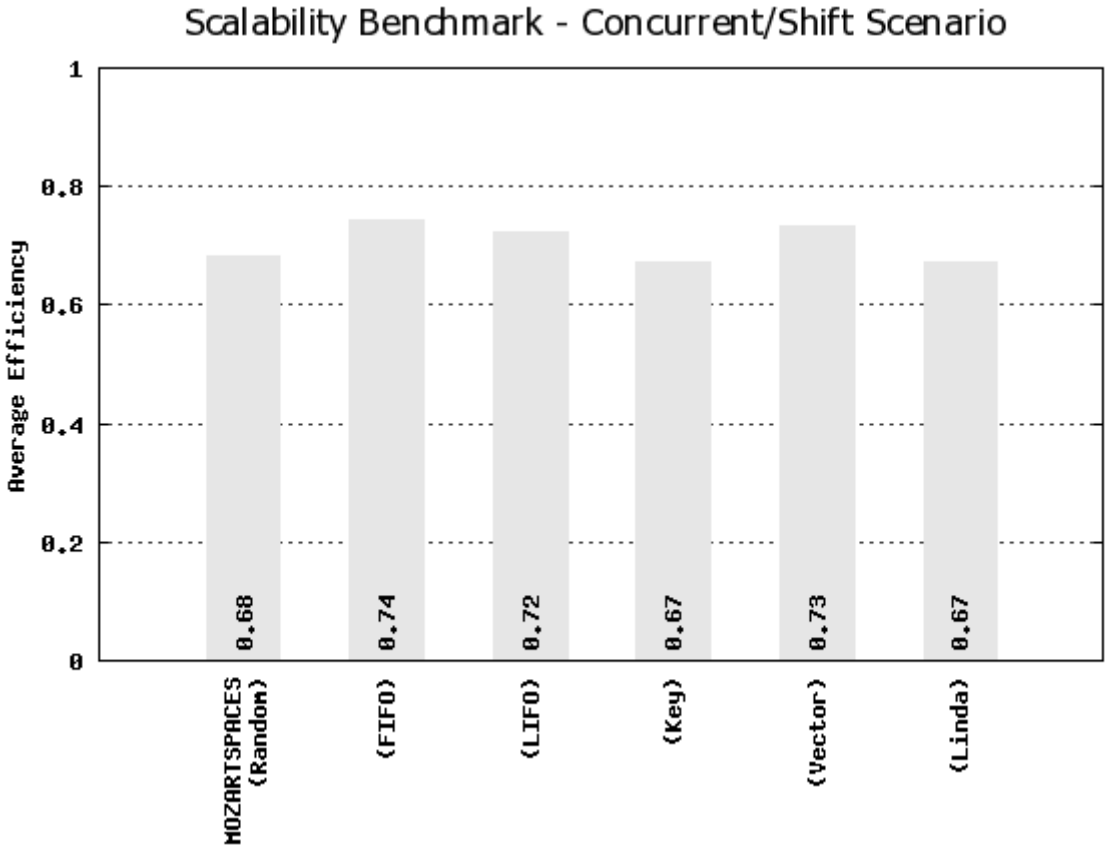


Figure 55: Scalability Diagram in Summary of Concurrent/Shift Scenario

It is no surprise that the values in the scalability diagram of concurrent shift operations are very similar to the measured and calculated values before.

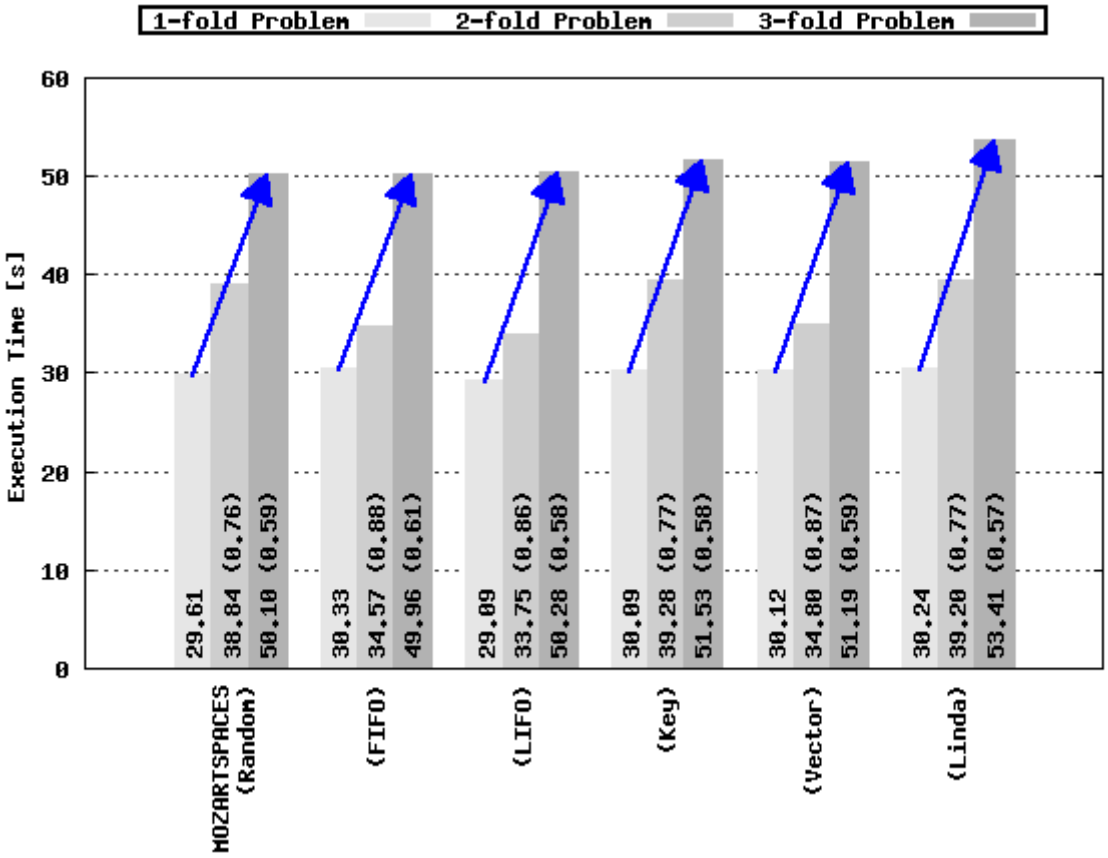


Figure 56: Scalability Diagram in Detail of Concurrent/Shift Scenario

From the diagram above it follows that non-linear scalability must be expected for this scenario, too.

4.3.3 Read

4.3.3.1 Scenario

In preparation for this scenario the container or rather space is filled with a defined number of entries. After that, a defined number of threads performs 1K read operations on the container or rather space each. Again, the used hardware configuration depends on the variant of the test case. The monitor measures the total execution time for all read operations. The maximum number of entries which can be added to the container is not limited.

Variant A → 1-fold Problem

20 threads perform a total of 20K read operations and the hardware configuration, referred to as Variant A → 1-fold Computing Power above, is used.

Thread	**Entries** (before)	**Operations** (measured)	**Entries** (after)
1	20K	1K read	20K
2		1K read	
...		...	
n		1K read	
...		...	
19		1K read	
20		1K read	

TOTAL		20K read	

Variant B → 2-fold Problem

40 threads perform a total of 40K read operations and the hardware configuration, referred to as Variant B → 2-fold Computing Power above, is used.

Thread	Entries (before)	Operations (measured)	Entries (after)
1	40K	1K read	40K
2		1K read	
...		...	
n		1K read	
...		...	
39		1K read	
40		1K read	
TOTAL		**40K read**	

Variant C → 3-fold Problem

60 threads perform a total of 60K read operations and the hardware configuration, referred to as Variant C → 3-fold Computing Power above, is used.

Thread	Entries (before)	Operations (measured)	Entries (after)
1	60K	1K read	60K
2		1K read	
...		...	
n		1K read	
...		...	
59		1K read	
60		1K read	
TOTAL		**60K read**	

4.3.3.2 Expectance

The execution time will increase extremely with growth of the problem size. The differences between the bars in the diagram will be essential larger than before:

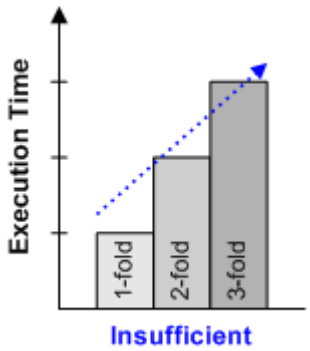


Figure 57: Expected Scalability Diagram of Concurrent/Read Scenario

4.3.3.3 Results

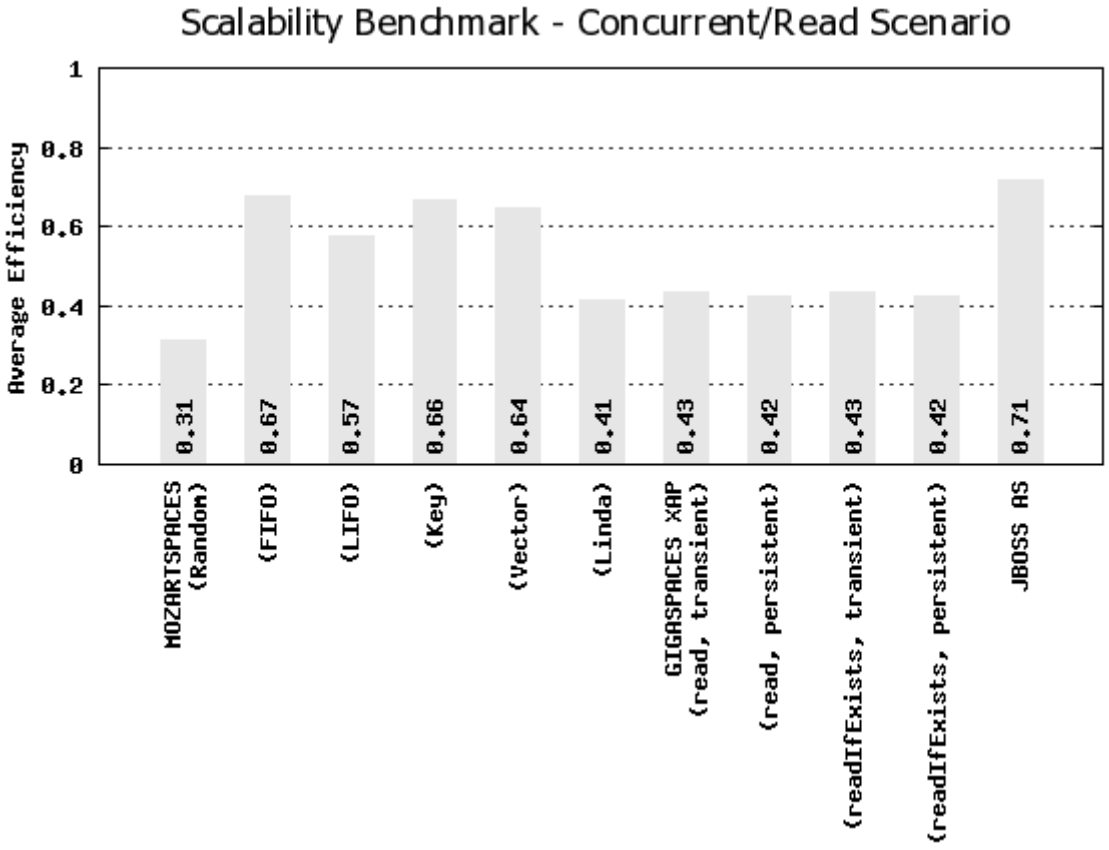


Figure 58: Scalability Diagram in Summary of Concurrent/Read Scenario

The assumption above comes true only for some of the middleware configurations. The scalability regarding concurrent read operations in MozartSpaces depends on the used selector. Using the Random or the Linda selector results in poor scalability, but the remaining selectors meet the ordinary requirements.

By contrast, the calculated efficiencies for the benchmarked GigaSpaces XAP configurations are nearly indistinguishable. The values indicate also that this middleware system does not scale well in relation to concurrent read operations.

As before, the benchmark results offer good scalability of JBoss AS for this scenario.

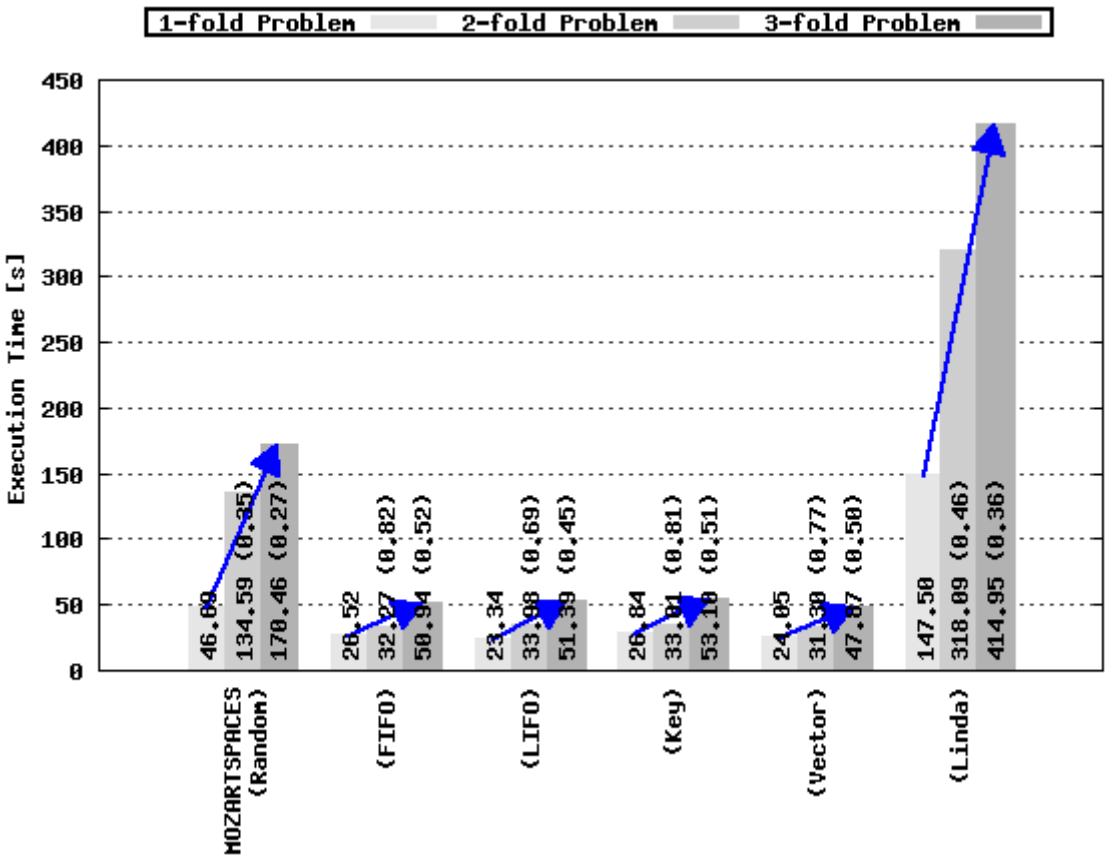


Figure 59: Scalability Diagram in Detail of Concurrent/Read Scenario (1)

The serial benchmarks have already shown performance deficits of the Random and Linda selector in comparison with the other selectors. These drawbacks also seem to have an adverse effect on these selectors' scalability behaviour. The remaining selectors lead to a scalability that is much better.

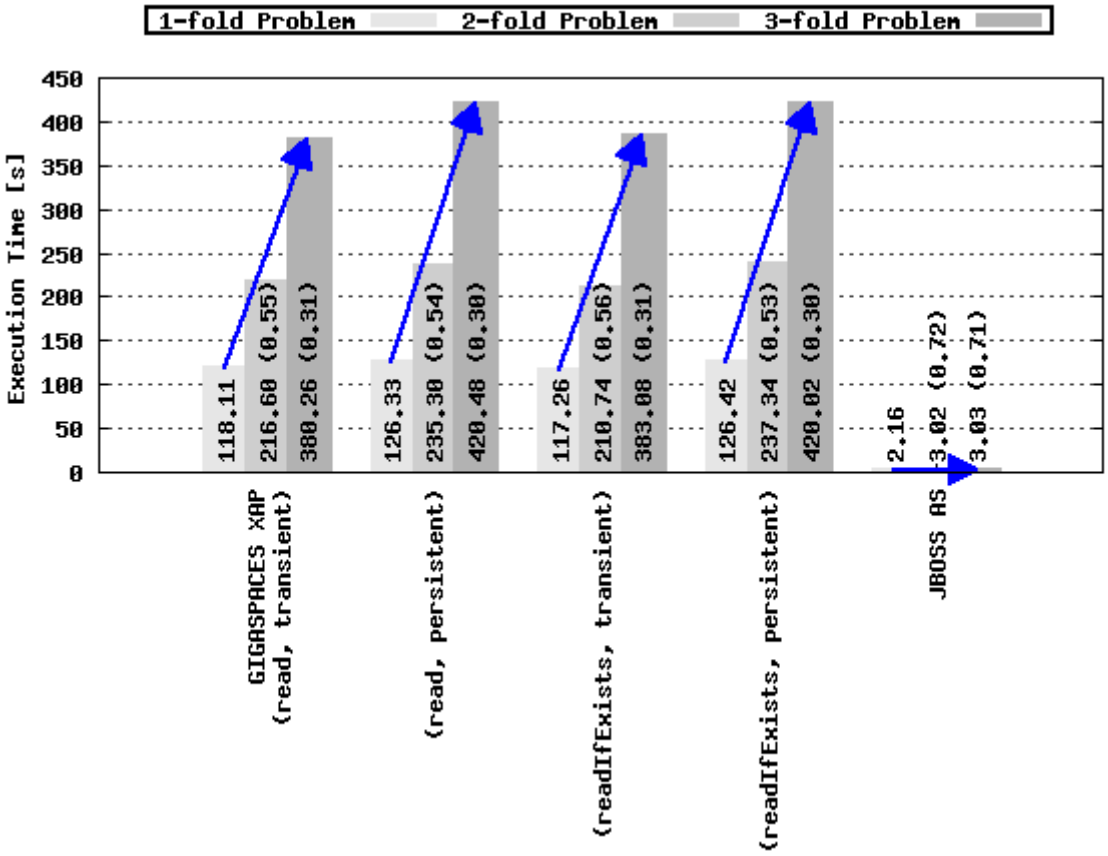


Figure 60: Scalability Diagram in Detail of Concurrent/Read Scenario (2)

The diagram shows that the efficiency of GigaSpaces XAP in connection with this scenario decreases nearly linear with increasing problem size. Accordingly, the 2-fold problem has an efficiency of circa $^1/_2$ and the 3-fold problem has an efficiency of circa $^1/_3$. This behaviour characterizes poor scalability.

By contrast, JBoss AS impresses with pretty constant and satisfying efficiency values, which means that concurrent reading of entries scales well.

4.3.4 Take

4.3.4.1 Scenario

Again, the container or rather space is filled initially with a defined number of entries. After that a defined number of threads is executed. Thereby, each thread performs 1K take operations on the container or rather space. The hardware configuration depends on the test case's variant. The monitor times the duration for all take operations. A container limit is not set.

Variant A → 1-fold Problem

20 threads perform a total of 20K take operations and the hardware configuration, referred to as Variant A → 1-fold Computing Power above, is used.

Thread	**Entries** (before)	**Operations** (measured)	**Entries** (after)
1	20K	1K take	0
2		1K take	
...		...	

n		1K take	
...		...	
19		1K take	
20		1K take	
TOTAL		**20K take**	

Variant B → 2-fold Problem

40 threads perform a total of 40K take operations and the hardware configuration, referred to as Variant B → 2-fold Computing Power above, is used.

Thread	**Entries** (before)	**Operations** (measured)	**Entries** (after)
1	40K	1K take	0
2		1K take	
...		...	
n		1K take	
...		...	
39		1K take	
40		1K take	
TOTAL		**40K take**	

Variant C → 3-fold Problem

60 threads perform a total of 60K take operations and the hardware configuration, referred to as Variant C → 3-fold Computing Power above, is used.

Thread	**Entries** (before)	**Operations** (measured)	**Entries** (after)
1	60K	1K take	0
2		1K take	
...		...	
n		1K take	
...		...	
59		1K take	
60		1K take	
TOTAL		**60K take**	

J2EE (JBoss AS) does not support take operations, which is why this benchmark scenario cannot be executed with this technology.

4.3.4.2 Expectance

The scalability will be insufficient, because the costs for taking an entry increase extremely with the number of entries in the container or rather space.

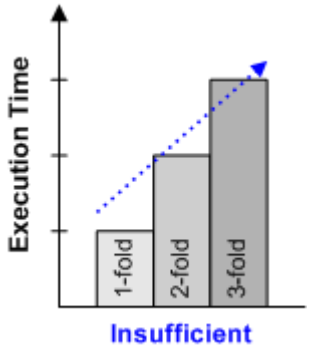


Figure 61: Expected Scalability Diagram of Concurrent/Take Scenario

4.3.4.3 Results

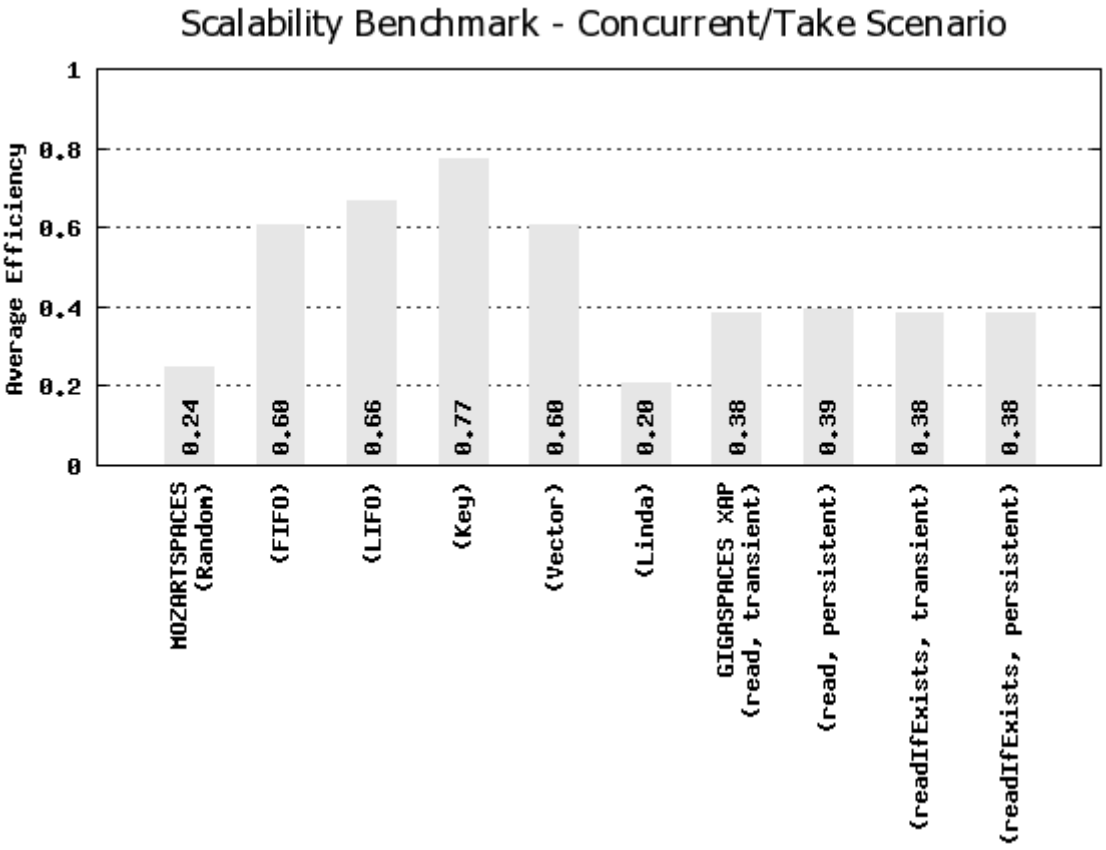


Figure 62: Scalability Diagram in Summary of Concurrent/Take Scenario

The scalability diagrams for taking entries concurrently from the container or rather space does not differ from that for reading entries. Consequently, the previous interpretations apply also to this scenario, so no additional comments are necessary here.

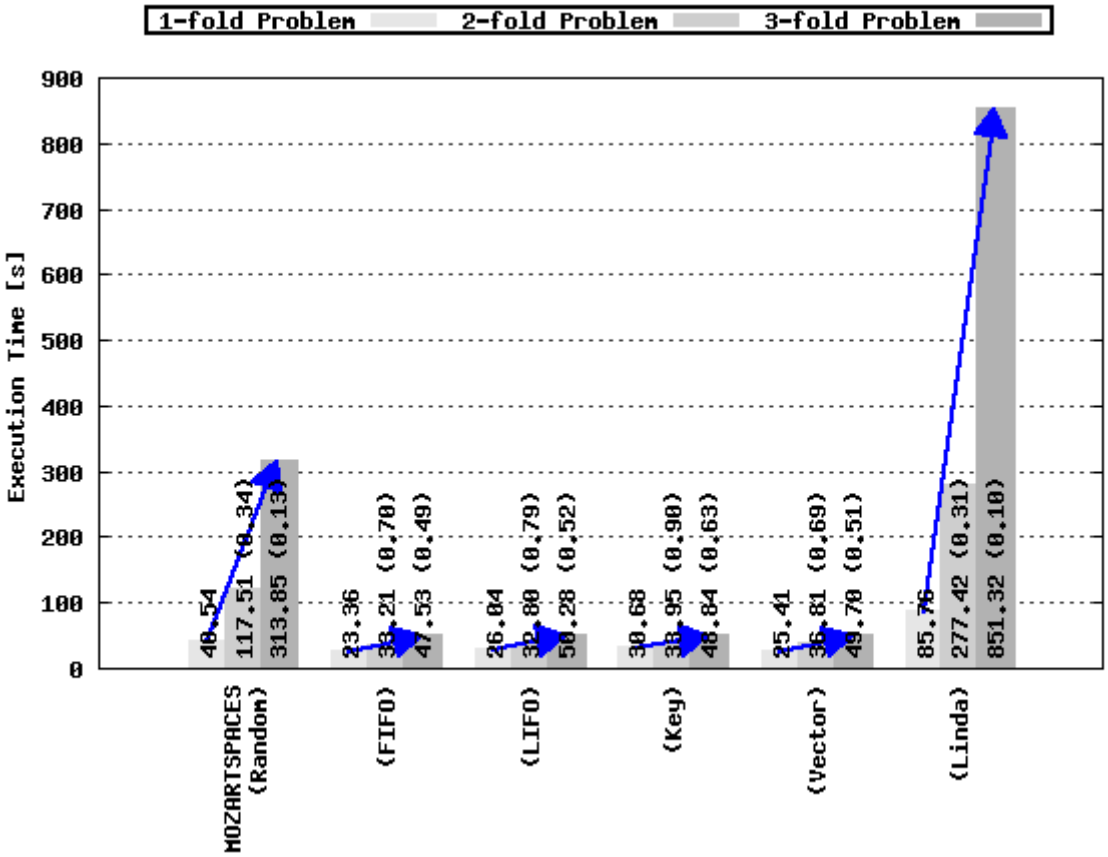


Figure 63: Scalability Diagram in Detail of Concurrent/Take Scenario (1)

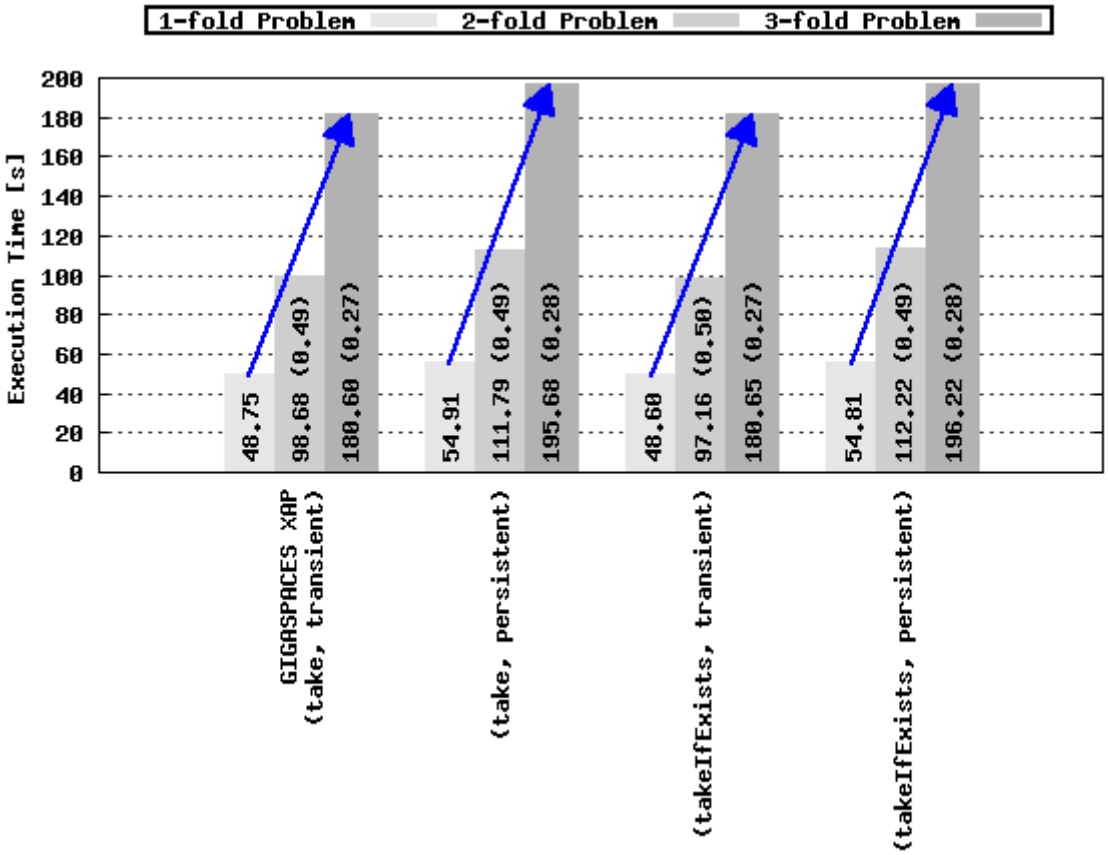


Figure 64: Scalability Diagram in Detail of Concurrent/Take Scenario (2)

4.3.5 Destroy

4.3.5.1 Scenario

This scenario complies with the previous one, but now destroy operations are employed.

Variant A → 1-fold Problem

20 threads perform a total of 20K destroy operations and the hardware configuration, referred to as Variant A → 1-fold Computing Power above, is used.

Thread	Entries (before)	Operations (measured)	Entries (after)
1	20K	1K destroy	0
2		1K destroy	
...		...	
n		1K destroy	
...		...	
19		1K destroy	
20		1K destroy	
TOTAL		**20K destroy**	

Variant B → 2-fold Problem

40 threads perform a total of 40K destroy operations and the hardware configuration, referred to as Variant B → 2-fold Computing Power above, is used.

Thread	Entries (before)	Operations (measured)	Entries (after)
1	40K	1K destroy	0
2		1K destroy	
...		...	
n		1K destroy	
...		...	
39		1K destroy	
40		1K destroy	
TOTAL		**40K destroy**	

Variant C → 3-fold Problem

60 threads perform a total of 60K destroy operations and the hardware configuration, referred to as Variant C → 3-fold Computing Power above, is used.

Thread	Entries (before)	Operations (measured)	Entries (after)
1	60K	1K destroy	0
2		1K destroy	
...		...	
n		1K destroy	
...		...	
59		1K destroy	
60		1K destroy	
TOTAL		**60K destroy**	

JavaSpaces (GigaSpaces XAP) does not support destroy operations, which is why this benchmark scenario cannot be executed with this technology.

4.3.5.2 Expectance

The scalability will be insufficient, too, because the costs for destroying an entry increase extremely with the number of entries in the container or rather space also in this case.

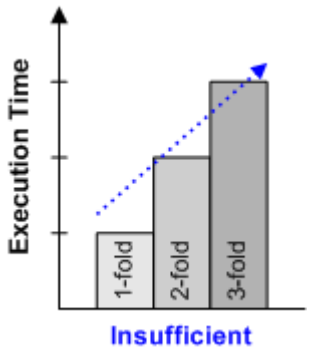


Figure 65: Expected Scalability Diagram of Concurrent/Destroy Scenario

4.3.5.3 Results

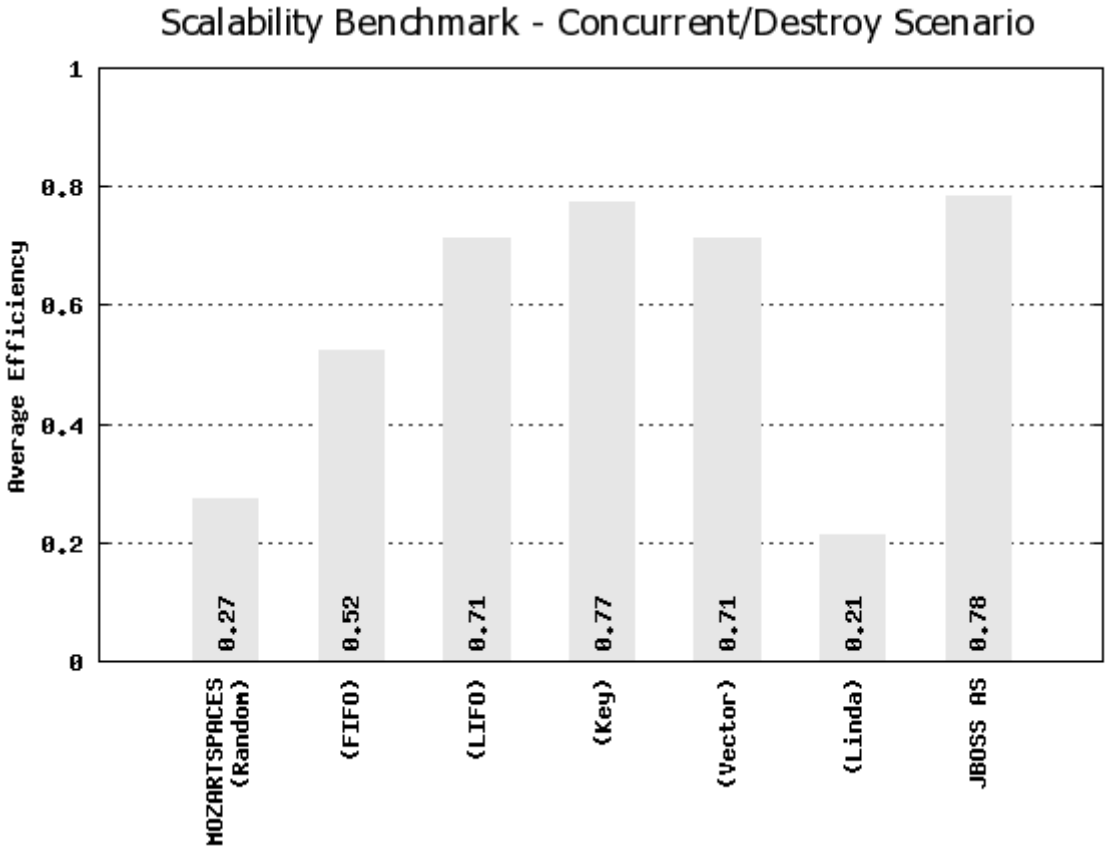


Figure 66: Scalability Diagram in Summary of Concurrent/Destroy Scenario

For the concurrent destroying of entries the scalability diagrams are very similar as for reading and taking entries. Therefore, the interpretations above hold true for this benchmark scenario, too.

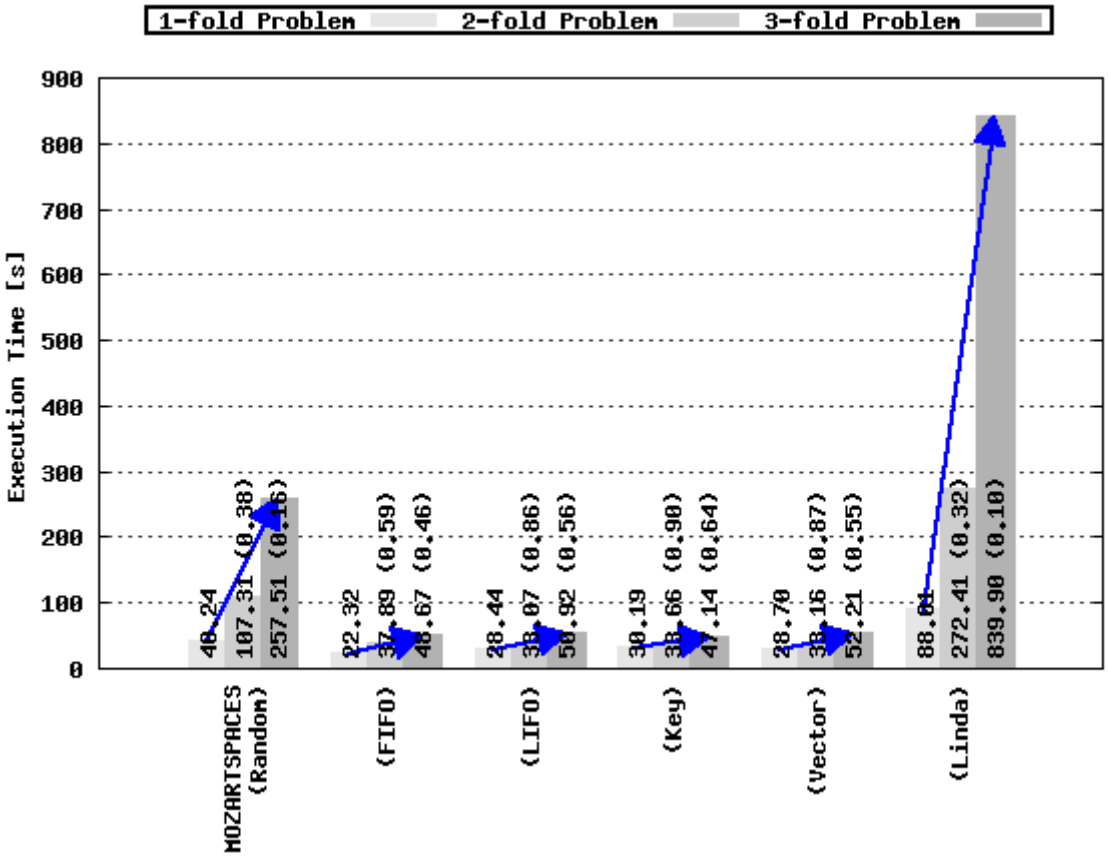


Figure 67: Scalability Diagram in Detail of Concurrent/Destroy Scenario (1)

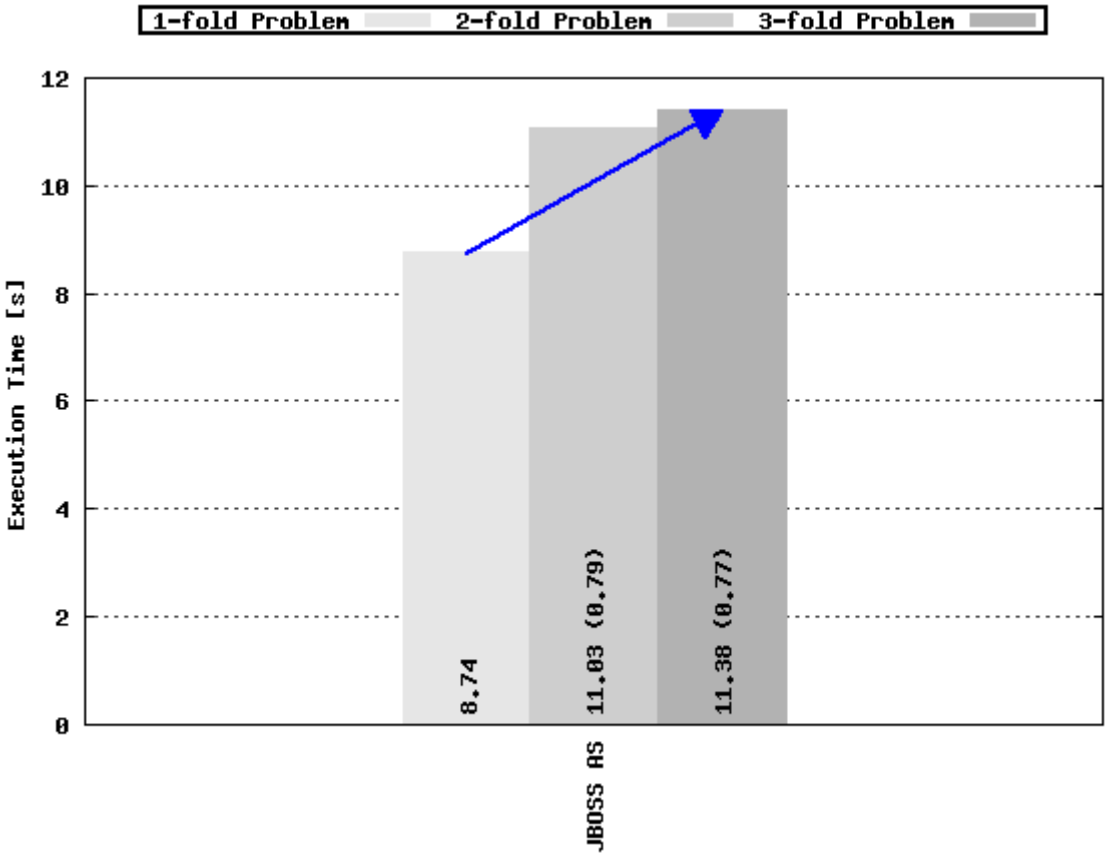


Figure 68: Scalability Diagram in Detail of Concurrent/Destroy Scenario (2)

4.4 Block Benchmarks

Finally, selected operation sequences will be executed by a defined number of threads, which simulate the clients and will run concurrently. Again, the total time for all clients will be measured. This type of benchmark evaluates the systems' scalability too. Each test case will be executed under three different circumstances. Now, all operation blocks will be embedded in explicit transactions. All middleware systems will be accessed remotely.

The following table gives a survey of the middleware configurations which are tested by the block benchmarks:

Middleware	Container / Space	Persistence	Transaction	Coordinator / Selector
XVSM (MozartSpaces)	remote	transient	explicit	Random
				FIFO
				LIFO
				Key
				Vector
				Linda
JavaSpaces (GigaSpaces XAP)	remote	transient	explicit	–
		persistent		
J2EE (JBoss AS)	remote	persistent	explicit	–

Note that the parameters used for executing the block benchmark scenario via the benchmark suite are mentioned in detail in chapter 8.1.2.2 of the appendix.

4.4.1 Write + Read + Take / Destroy

4.4.1.1 Scenario

10 iterations are performed, which are synchronized and run in series accordingly. At the beginning of each pass a defined number of threads is started. Each thread creates a new explicit transaction, executes 100 write, 100 read and 50 take operations on the container or rather space and commits its transaction. Since J2EE (JBoss AS) does not support the taking of entries, destroy operations are used instead. Again, the hardware configuration depends on the test case. The monitor is not reset and measures the time for the execution of all operations. The container size is not limited.

Variant A → 1-fold Problem

20 threads perform a total of 20K write, 20K read and 10K take or rather destroy operations during 10 iterations. The costs for the transaction handling must be added. The hardware configuration, referred to as Variant A → 1-fold Computing Power above, is used.

Iteration	Threads	Entries (before)	Operations per Thread (measured)	Entries (after)
1	20	0	beginTransaction 100 write + 100 read + 50 take/destroy commitTransaction	1K
2	20	1K	beginTransaction 100 write + 100 read + 50 take/destroy commitTransaction	2K
…	…	…	…	…

n	20	1K · (n – 1)	beginTransaction 100 write + 100 read + 50 take/destroy commitTransaction	1K · n
…	…	…	…	…
9	20	8K	beginTransaction 100 write + 100 read + 50 take/destroy commitTransaction	9K
10	20	9K	beginTransaction 100 write + 100 read + 50 take/destroy commitTransaction	10K
TOTAL			**20K write** **20K read** **10K take/destroy**	
			200 beginTransaction **200 commitTransaction**	

Variant B → 2-fold Problem

40 threads perform a total of 40K write, 40K read and 20K take or rather destroy operations during 10 iterations. The costs for the transaction handling must be added. The hardware configuration, referred to as Variant B → 2-fold Computing Power above, is used.

Iteration	**Threads**	**Entries** (before)	**Operations per Thread** (measured)	**Entries** (after)
1	40	0	beginTransaction 100 write + 100 read + 50 take/destroy commitTransaction	2K
2	40	2K	beginTransaction 100 write + 100 read + 50 take/destroy commitTransaction	4K
…	…	…	…	…
n	40	2K · (n – 1)	beginTransaction 100 write + 100 read + 50 take/destroy commitTransaction	2K · n
…	…	…	…	…
9	40	16K	beginTransaction 100 write + 100 read + 50 take/destroy commitTransaction	18K
10	40	18K	beginTransaction 100 write + 100 read + 50 take/destroy commitTransaction	20K
TOTAL			**40K write** **40K read** **20K take/destroy**	
			400 beginTransaction **400 commitTransaction**	

Variant C → 3-fold Problem

60 threads perform a total of 60K write, 60K read and 30K take or rather destroy operations during 10 iterations. The costs for the transaction handling must be added. The hardware configuration, referred to as Variant C → 3-fold Computing Power above, is used.

Iteration	**Threads**	**Entries** (before)	**Operations per Thread** (measured)	**Entries** (after)
1	60	0	beginTransaction	3K

			100 write + 100 read + 50 take/destroy commitTransaction	
2	60	3K	beginTransaction 100 write + 100 read + 50 take/destroy commitTransaction	6K
…	…	…	…	…
n	60	3K · (n – 1)	beginTransaction 100 write + 100 read + 50 take/destroy commitTransaction	3K · n
…	…	…	…	…
9	60	24K	beginTransaction 100 write + 100 read + 50 take/destroy commitTransaction	27K
10	60	27K	beginTransaction 100 write + 100 read + 50 take/destroy commitTransaction	30K
TOTAL			**60K write** **60K read** **30K take/destroy**	
			600 beginTransaction **600 commitTransaction**	

4.4.1.2 Expectance

The scalability for this scenario will be moderate, maybe insufficient. The uncertainty results from the questionable behaviour of the read and take or rather destroy operations.

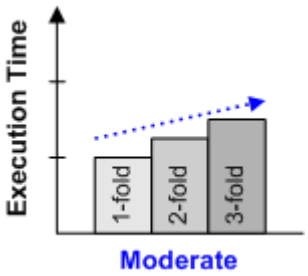


Figure 69: Expected Scalability Diagram of Block Scenario

4.4.1.3 Results

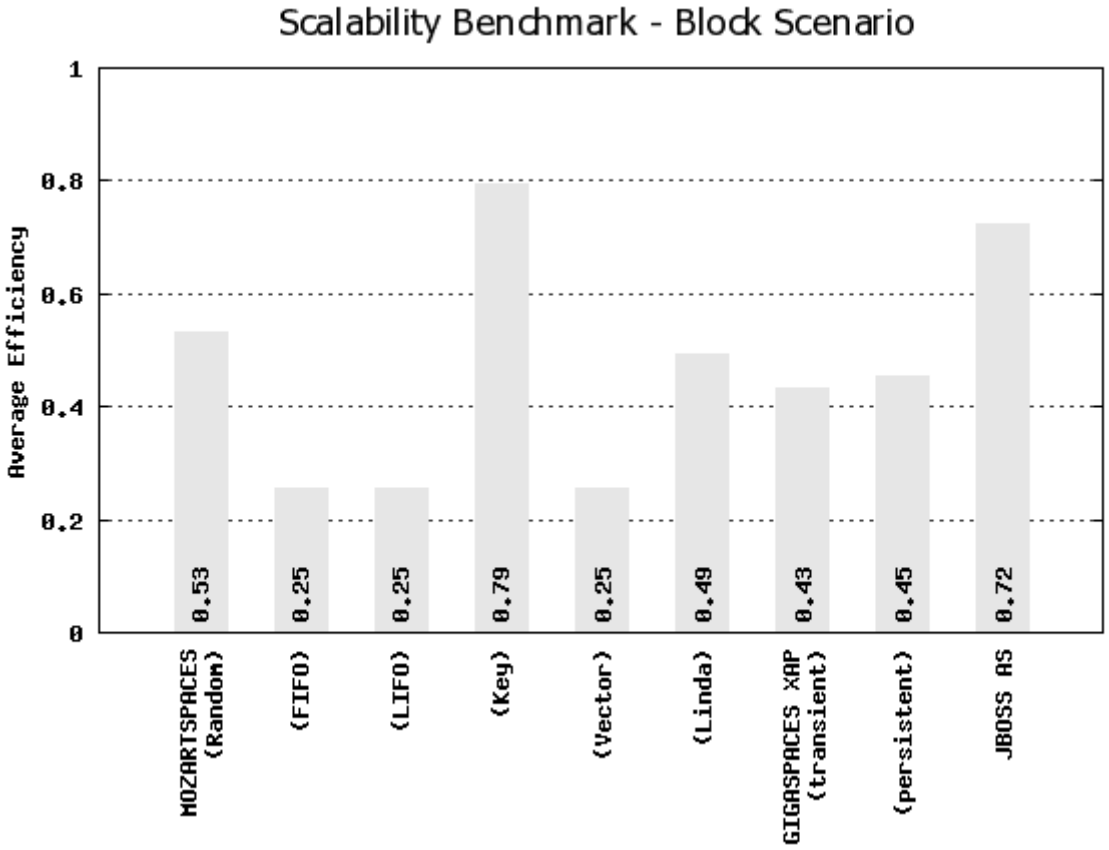


Figure 70: Scalability Diagram in Summary of Block Scenario

The concurrent execution of the operation sequences returns diverse average efficiencies. While the scalability of MozartSpaces for this scenario depends highly on the used coordinator and selector, GigaSpaces XAP shows nearly the same scalability behaviour for both transient and persistent data management.

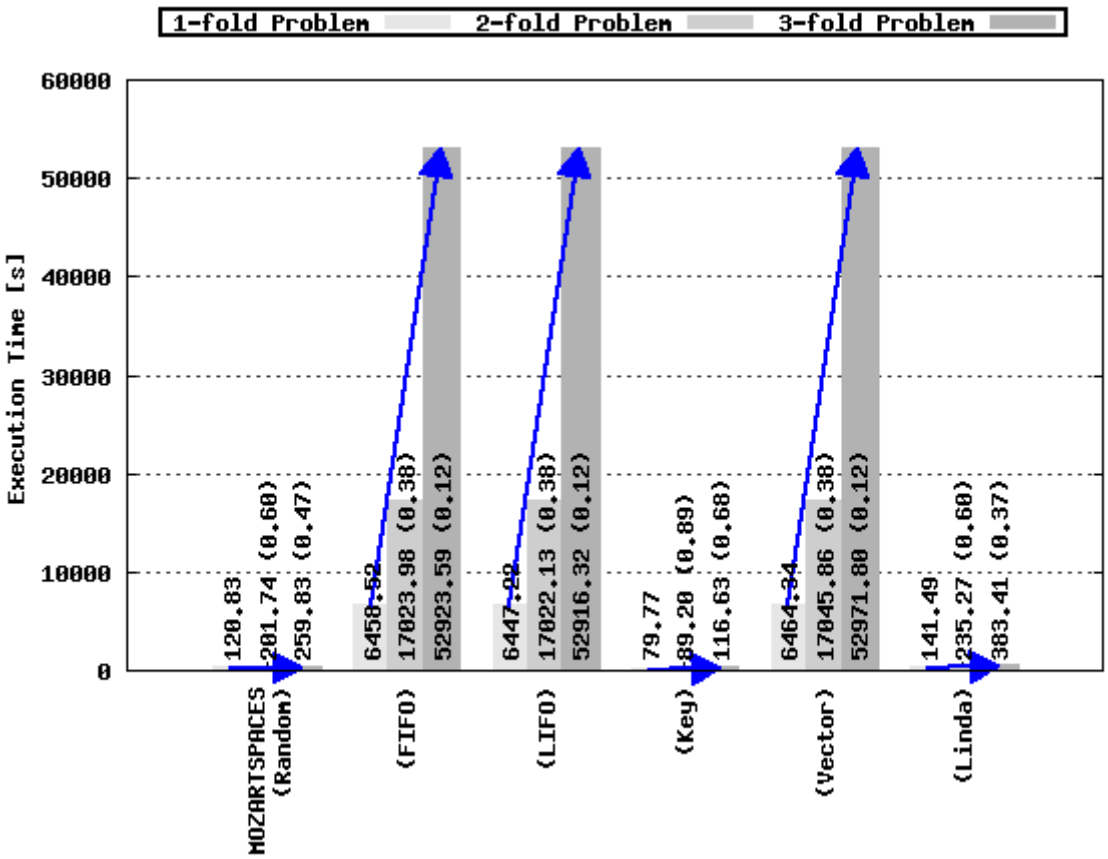


Figure 71: Scalability Diagram in Detail of Block Scenario (1)

MozartSpaces scales well with the Key coordinator and selector. If the Random or Linda coordinator and selector are used, the execution of the operation sequences can even be considered halfway satisfactory, and the calculated efficiencies are reasonably okay. But using FIFO, LIFO or Vector as coordinator and selector results in disastrous performance and scalability measurements. The analysis of this problem gives a reasonable explanation for this mismatch. The use of explicit transactions in connection with these coordinators and selectors has the effect that the operation sequences are executed strictly in series. This means that each thread locks the container in exclusive mode for the execution of its operation sequence, and that is very expensive. In other words, any parallelism is eliminated and as a result of this, the systems' performance and scalability collapse.

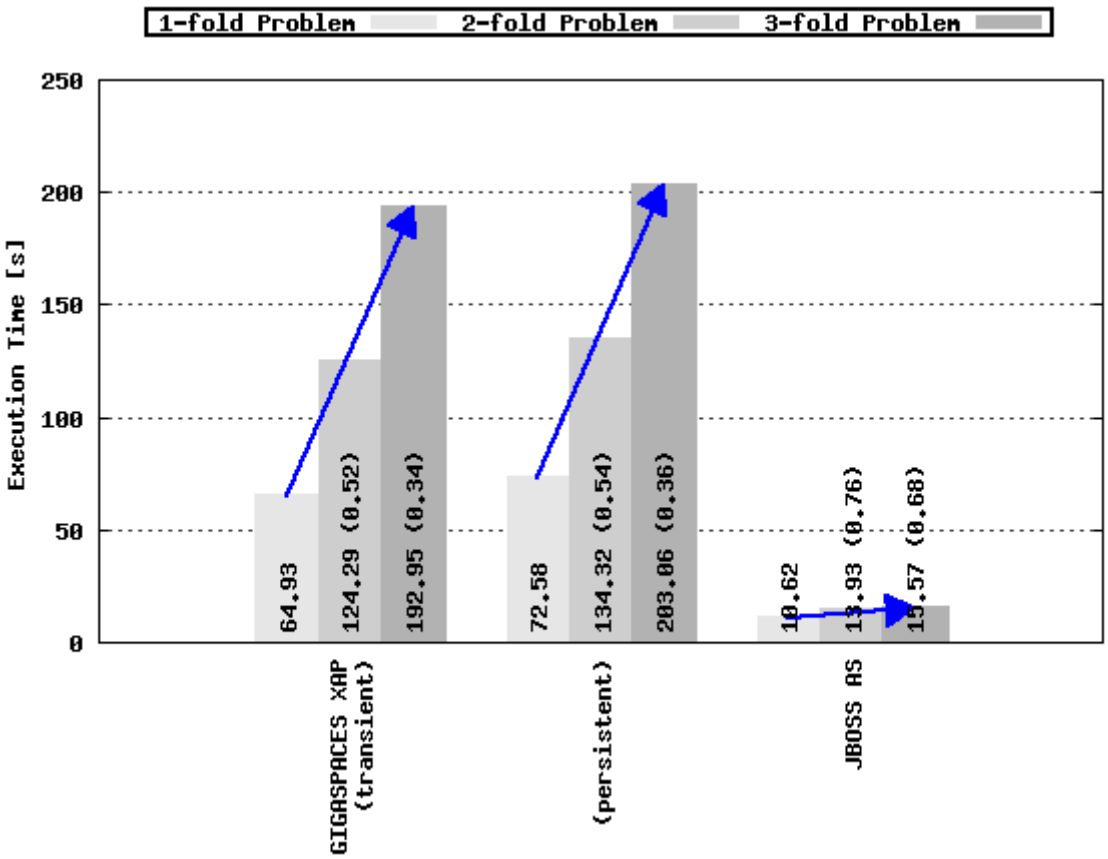


Figure 72: Scalability Diagram in Detail of Block Scenario (2)

The benchmark results for using GigaSpaces XAP or rather JBoss AS are not really surprising, because they are just the continuation of the measured and calculated values above.

5 Knowledge Transfer

In this chapter the body of acquired knowledge will be transferred for supporting the development of TripCom.

5.1 TripCom – Triple Space Communication

TripCom released at [57] is a middleware system, which is currently in development and pursues a new approach, the Triple Space Communication. The project is funded by the European Commission and implemented by nine partners, among others the Vienna University of Technology, from seven European countries.

TSC is a communication paradigm for anonymous and asynchronous information exchange between machines. It joins the concepts of Web Services, Semantic Web and Tuple Space with the aim that this technology will become the web for machines, like HyperText Markup Language (HTML) became the web for humans. A short overview is given by the following figure:

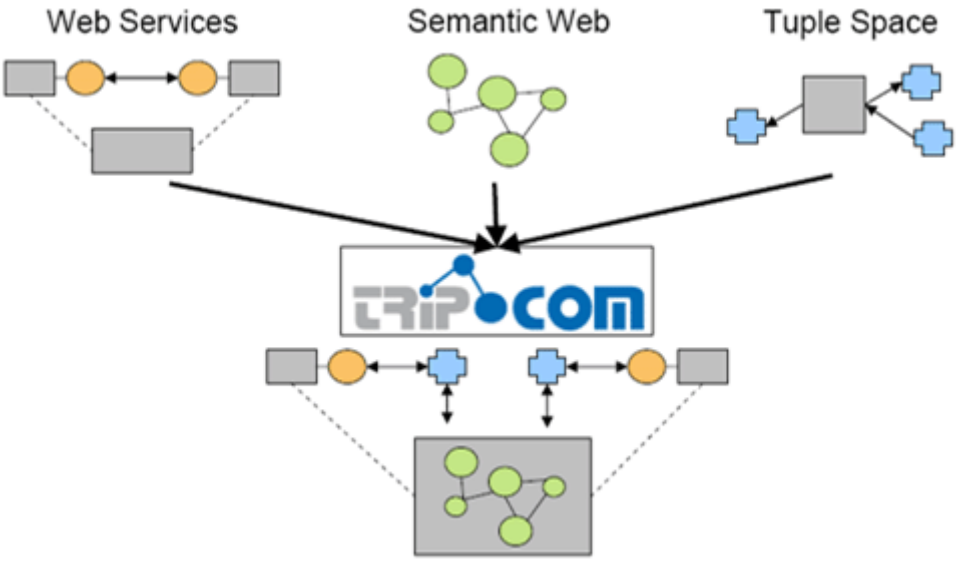


Figure 73: Architecture of TripCom

The concept of TSC is strongly influenced by the Semantic Web technology and extends the tuples so that Resource Description Framework (RDF) triples can be processed. Furthermore, this technology enhances the primitive operations to support the special features of semantic tuples and, in addition, it uses inference for matching.

One of the benefits of TSC is its high degree of autonomy. In [58], the following four types are distinguished:

- Time autonomy: There are no time dependencies between the data provider and the reader. All participants can access the triple space for writing and reading data whenever they want.
- Location autonomy: Storing data in the triple space is performed in such a manner that the triple space's repository and the storage of the data provider or reader remain independent. This is accomplished by passing the triples to and from the triple space always by value and in a specified format.

- Reference autonomy: The communication between the data provider and the reader can be carried out through the triple space without knowing each other. Writing and reading data can be carried out in anonymous way.
- Data schema autonomy: A particular data format based on RDF triples is used for the data exchange. All data written to and read from the triple space offer this data format. Through this approach the data provider and the reader become independent in regard to the format of their data.

5.2 Lectures Learned

Now, the experiences and the knowledge gained during the implementation of the performance and scalability benchmarks will be summarized with regard to the development of TripCom.

5.2.1 Communication

The benchmarks with remote container or rather space demonstrate that the performance of a middleware system depends highly on the implementation of the communication layer. Dividing the execution of a remote operation into its individual parts and measuring their particular durations shows that communication issues, and not the execution of the real operation on the callee, require a good deal of the time.

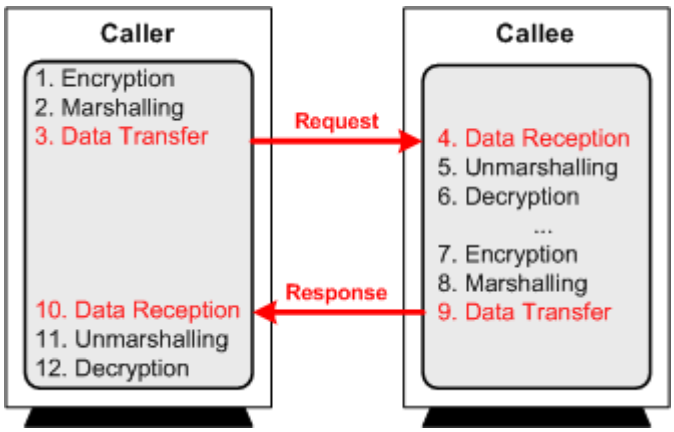


Figure 74: Communication Issues of Remote Operation Execution

Since the domain of TripCom in the majority of cases will be a distributed environment, communication plays a fundamental role for the development of TripCom. The high degree of autonomy, as described above, effects an additional increasing of the communication demands, because the distributed spaces must communicate with each other.

So, it is very important that TripCom will provide efficient communication protocols. Furthermore, the transferred data volume will have to be reduced to a minimum and the transmission of needless data will have to be avoided. The use of caching and replication will also make a contribution to reach the required performance level. However, using replication raises the transmission of data again.

Another conclusion of the benchmarks' implementation is that it may be advantageous to move the execution of the business logic onto the machine, where the data are located, because doing so reduces the communication demands enormously. JBoss AS

pursues this policy and scales well. But maybe this approach conflicts with the TripCom's concept.

5.2.2 Concurrency

A middleware system is almost always a multi-user system, in which concurrent access and shared resources are a matter of course. These issues must be considered during design and development of TripCom right from the beginning.

The results of the block benchmarks with the FIFO, LIFO and Vector coordinators and selectors in MozartSpaces show that accessing the container in exclusive mode implicates disastrous performance and scalability. So, whenever it is possible, the locking of resources in exclusive mode should be prevented in TripCom. If this is not possible, it should be reflected, if this feature is really essential.

Furthermore, the call of remote procedures in asynchronous way, if procurable, also speeds up the systems' performance, because the operations need not be executed instantly and the system can handle such calls more flexible. Sometimes, only a part of an operation can be executed asynchronously, but even so, the performance of the middleware system will increase. For this reason, it is important that TripCom will pursue a multi-threaded design from the beginning.

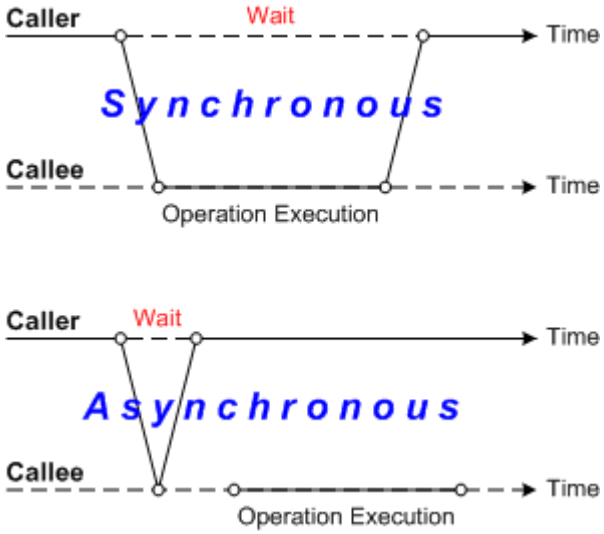


Figure 75: Synchronous and Asynchronous Remote Operation Execution

The support of transactions is another important issue in a middleware system. But sometimes the execution of operations in a non-transactional environment is sufficient for fulfilling the requirements. The performance of MozartSpaces is adversely affected by its transaction management, because each operation which is called without explicit transaction is executed within an implicit transactional context. So, it will be a desirable feature of TripCom to support the execution of operations in a non-transactional environment, because this speeds up system's performance.

5.2.3 Interfaces

All benchmarked middleware systems impressed with support and efficient implementations of established APIs. This allows the solid and simple use of these

middleware systems. Therefore, the support of popular APIs will be important for the acceptance and pervasion of TripCom.

5.2.4 Documentation

Software products without adequate documentation are designated to fail concerning their market penetration. GigaSpaces XAP and JBoss AS provide an excellent documentation, which simplifies their application enormously.

By contrast, the tutorials and papers currently available for MozartSpaces are rather insufficient, which sometimes makes it difficult to use this middleware system. But it could be observed that new documentation for MozartSpaces was published continuously.

TripCom's documentation should be sufficient, complete and, of course, faultless.

6 Evaluation and Conclusion

Many definitions and related topics for benchmark, performance, scalability and middleware have been discussed in the first part of this work. The versatile definitions of performance and scalability should allow a clear differentiation of these terms. Once again, it should be noted that performance measurements are indeed a precondition for benchmarking the scalability of a system, but the concepts per se are standing for something different. Showing this was of particular importance, because it is a precondition for the understanding of the further work.

For measuring and comparing the performance and scalability rather simple metrics were used, which was not disadvantageous belatedly. It was useful to give up on rating the whole system and to benchmark only the performance and scalability of individual test cases. The performance was measured by timing the execution time. The evaluation of the scalability was performed by calculating the efficiency, which is the ratio between the time required for executing a problem on a machine and the time required for executing the quasi-same problem, but with n-fold problem size, on the quasi-same machine, but with n-fold resources. Each benchmark scenario, which targeted on scalability, was executed with 1-fold, 2-fold and 3-fold problem size and the average efficiency value was calculated thereof. But rating the scalability of a test case by its average efficiency presents a non-satisfying approach, because this method sweeps important information under the carpet. Especially, in case that the efficiency for the 3-fold problem size is much smaller than for the 2-fold problem size, there is a risk of exponential growth. However, this does not emerge from the calculated average value. On the other hand, the diagrams allow the comparison of the separate efficiency values, even if this method is more difficult and unpractical. So, an ideal solution could not be found.

Furthermore, the middleware concepts of XVSM, JavaSpaces and J2EE were introduced as well as the belonging systems, namely MozartSpaces, GigaSpaces XAP and JBoss AS. The choice of J2EE (JBoss AS) seemed unpassable in the beginning. But finally, the selection of the benchmarked concepts and systems turned out to be suitable. Adding the diverse concept of J2EE to the two similar SBC concepts enabled the winning of interesting and valuable insights. In addition, the comparison of such unequal middleware technologies was one of the innovations of this work.

Due to the differences between the middleware concepts, the functionalities had to be reduced to a common denominator. This was accomplished by selecting the overlapping operations such as write, shift, read, take and destroy as benchmark criteria. This was a good approach, even if the high-level approach allowed the comparison of only such functionalities that all concepts provide. The implemented scenarios – the operations were executed in series, in parallel and blockwise in operation sequences – might appear to be simple, but they covered all established use cases and showed the systems' strengths and weaknesses. For more detailed benchmarks, it would have been necessary to come to an agreement about a fixed middleware concept. Indeed, this would allow low-level tests, but then only systems based on this concept could be compared.

The description of the test cases and visualization of the test results has succeeded. So, the multiple adaptions for the improvement of these issues paid off. The performance

and scalability of a middleware configuration with regard to a test case is obvious at a glance.

The benchmarks showed that the performance and scalability of MozartSpaces depends highly on the used coordinator and selector. Both can be rated in summary as poor compared with the other two systems. Fortunately, as a result of the benchmarks the performance of this middleware in connection with some test cases could be improved considerably. Some of the deficits, which were identified by the benchmarks, could be fixed by the developers and the correction effected a significant improvement of performance. Among others, the execution of MozartSpaces' operations on a remote container could speed up by factor 2.5. But also the stability of this middleware system could be enhanced in consequence of some error corrections after the concurrent benchmarks. Looking at the results reveals, thanks to the clear representation, further room for improvement. For example, the performance of some remote operations in MozartSpaces with certain coordinators and selectors is still insufficient. This problem should be fixed rapidly. Moreover, locking entries in connection with an explicit transaction should be analysed, too, because the block benchmarks identified certain imperfections. In support of MozartSpaces must be noted that this system is still not as long as the other two systems in development.

GigaSpaces XAP's performance and scalability was dependent on the executed operations and can be classified in summary as mean. Writing entries into a space performed and scaled well, however the read and take operations took place in a non-satisfactory manner. The execution of concurrent write operations in connection with GigaSpaces XAP highlighted the limits of the selected test design. It demonstrated that using only one test client machine in connection with remote tests might not always be enough.

JBoss AS impressed with excellent performance and scalability. This application server played out its strength, namely that the business logic is located and executed in the middleware system itself, so that the communication overhead is reduced to a minimum. Also, it benefited from additional computing resources most of all benchmarked technologies, which resulted in constant efficiency values for all problem sizes.

At the end of this work, the experiences and the knowledge gained during the implementation of the performance and scalability benchmarks were transferred to the development of TripCom, a middleware system following the TSC approach. Thereby, the importance of communication, concurrency, interfaces and documentation for middleware was highlighted once again. Since there was no stable prototype of TripCom available, this middleware system could not be benchmarked.

As a result of this work a powerful benchmark suite for XVSM (MozartSpaces), JavaSpaces (GigaSpaces XAP) and J2EE (JBoss AS) as well as a working test environment that allows the automated execution of the benchmark scenarios, remains in addition to the actual benchmark results. Therefore, the repetition of the benchmarks will be possible at the push of a button and will generate updated diagrams. This allows a fast and simple performance and scalability inspection of an advanced product version.

Furthermore, the developed benchmark suite enables the creation of own scenarios. The loose coupling of the components, which was worth one's weight in Gold, offers many possibilities. And based on the present work self-acting should be simple, because mostly it will be sufficient to copy an adequate scenario and adapt it.

More time and effort could have been invested in finding the ideal configuration of each middleware system. Perhaps, some scalability problems of GigaSpaces XAP would have been got under control. However, getting a feeling for the strengths and weaknesses of the benchmarked systems was more important than getting a concrete value for each test case. And this aim seems to have been accomplished successfully.

Finally, some issues should be reflected, which can be benchmarked in further works. Consider the fact that the following catchwords are by far not complete and should only be a motivation for the reader:

- Aspects
- Notifications
- Transactions (including rollbackTransaction)
- Bulk operations
- Entries with different data types
- Several containers or rather spaces concurrently
- Replicated or clustered containers or rather spaces

7 References

[1] R. Krummenacher, E. Simperl, D. Foxvog, V. Momtchev, D. Cerizza, L. Nixon, D. Cerri, B. Sapkota, K. Teymourian, P. Obermeier, D. Martin, H. Moritsch, O. Shafiq, D. De Francisco. Towards a Scalable Triple Space. Technical Report, EU FP6-02732 TripCom, March 2008.

[2] N. Stoodley. Business Intelligence System Scalability – A Primer. Technical Paper, Crystal Decisions Incorporated, 2002.

[3] A.-L. Burness, R. Titmuss, C. Lebre, K. Brown, A. Brookland. Scalability Evaluation of a Distributed Agent System. Distributed Systems Engineering 6, 1999.

[4] L. Duboc, D. Rosenblum, T. Wicks. A Framework for Modelling and Analysis of Software Systems Scalability. Proceedings of the 28th International Conference on Software Engineering, May 2006.

[5] B. Neuman. Scale in Distributed Systems. Readings in Distributed Computing Systems. IEEE Computer Society Press, Los Alamitos (USA), 1994.

[6] L. Williams, C. Smith. QSEM: Quantitative Scalability Evaluation Method. PerfX and Performance Engineering Services, 2005.

[7] M. Van Steen, S. Van der Zijden, H. Sips. Software Engineering for Scalable Distributed Applications. Proceedings of the 22nd International Computer Software and Applications Conference, 1998.

[8] A. Bondi. Characteristics of Scalability and Their Impact on Performance. Proceedings of the 2nd International Workshop on Software and Performance, Ottawa (CAN), 2000.

[9] M. Hill. What is Scalability? ACM SIGARCH Computer Architecture News, Volume 18, Issue 4, December 1990.

[10] P. Jogalekar, M. Woodside. Evaluating the Scalability of Distributed Systems. IEEE Transactions on Parallel and Distributed Systems, Volume 11, Issue 6, Pages 589-603, June 2000.

[11] B. Shea. Avoiding Scalability Shock: Five Steps to Managing Performance of E-Business Applications. Software Testing and Quality Magazine, May/June 2000.

[12] C. Smith, L. Williams. Best Practices for Software Performance Engineering. Proceedings of the 29th International CMG Conference, Dallas (USA), 2003.

[13] A. Chandor. Dictionary of Computers. Penguin Books, 1970.

[14] B. Jenkins. Developments in Computer Auditing. Accountant 537, 1972.

[15] P. Bernstein. Middleware: A Model for Distributed System Services. Communications of the ACM, Volume 39, Issue 2, Pages 86-98, 1996.

[16] H. Pinus. Middleware: Past and Present a Comparison. Seminar Paper, Darmstadt University of Technology (GER), Software Technology Group, 2004.

[17] D. Bakken. Middleware. Encyclopedia of Distributed Computing, Kluwer Academic Press, 2003.

[18] P. Brebner, E. Cecchet, J. Marguerite, P. Tuma, O. Ciuhandu, B. Dufour, L. Eeckhout, S. Frenot, A. Krishna, J. Murphy, C. Verbrugge. Middleware Benchmarking: Approaches, Results, Experiences. Concurrency and Computation, Volume 17, Issue 15, Pages 1799-1806, December 2005.

[19] D. Fiedler, K. Walcott, T. Richardson, G. Kapfhammer, A. Amer, P. Chrysanthis. Towards the Measurement of Tuple Space Performance. ACM SIGMETRICS Performance Evaluation Review, Volume 33, Issue 3, Pages 51-62, December 2005.

[20] E. Cecchet, J. Marguerite, W. Zwaenepoel. Performance and Scalability of EJB Applications. ACM SIGPLAN Notices, Volume 37, Issue 11, Pages 246-261, November 2002.

[21] http://www.spec.org/jAppServer2004/ (last visited: 2008-10-10)

[22] http://www.spec.org/jbb2005/ (last visited: 2008-10-10)

[23] http://portal.acm.org/browse_dl.cfm?linked=1&part=series&idx=SERIES850&coll=ACM&dl=ACM&CFID=6015312&CFTOKEN=39760032 (last visited: 2008-10-10)

[24] http://www.cmg.org (last visited: 2008-10-10)

[25] http://en.wiktionary.org/wiki/benchmark (last visited: 2008-10-10)

[26] http://en.wikipedia.org/wiki/Benchmark_(computing) (last visited: 2008-10-10)

[27] http://jakarta.apache.org/jmeter/ (last visited: 2008-10-10)

[28] http://faban.sunsource.net (last visited: 2008-10-10)

[29] http://www.gnuplot.info (last visited: 2008-10-10)

[30] http://jamonapi.sourceforge.net (last visited: 2008-10-10)

[31] http://jbento.sourceforge.net (last visited: 2008-10-10)

[32] http://jmonit.sourceforge.net (last visited: 2008-10-10)

[33] http://sourceforge.net/projects/jperf/ (last visited: 2008-10-10)

[34] http://www.clarkware.com/software/JUnitPerf.html (last visited: 2008-10-10)

[35] http://loadunit.sourceforge.net (last visited: 2008-10-10)

[36] http://p-unit.sourceforge.net (last visited: 2008-10-10)

[37] http://en.wikipedia.org/wiki/Power_(physics) (last visited: 2008-10-10)

[38] http://en.wikipedia.org/wiki/Computer_performance (last visited: 2008-10-10)

[39] http://en.wikipedia.org/wiki/Scalability (last visited: 2008-10-10)

[40] L. Williams, C. Smith. Web Application Scalability: A Model-Based Approach. Proceedings of the Computer Measurement Group, Las Vegas (USA), December 2004.

[41] G. Coulouris, J. Dollimore, T. Kindberg. Distributed Systems – Concepts and Design. Addison-Wesley, 2001.

[42] http://middleware.objectweb.org (last visited: 2008-10-10)

[43] A. Tanenbaum, M. Van Steen. Distributed Systems – Principles and Paradigms. Prentice Hall, 2002.

[44] http://de.wikipedia.org/wiki/Middleware (last visited: 2008-10-10)

[45] E. Kühn. Virtual Shared Memory for Distributed Architectures. Nova Science Publishers, 2001.

[46] M. Wittmann. XVSM Tutorial. Diploma Thesis, Vienna University of Technology (AUT), E185/1, Space Based Computing Group, 2008.

[47] http://en.wikipedia.org/wiki/Linda_(coordination_language) (last visited: 2008-10-10)

[48] http://www.jini.org (last visited: 2008-10-10)

[49] G. Löffler. Java in Space. Java Magazin – Internet & Enterprise Technology, Issue 2, February 2004.

[50] E. Jendrock, J. Ball, D. Carson, I. Evans, S. Fordin, K. Haase. The Java EE 5 Tutorial – Third Edition. Addison-Wesley, 2006.

[51] D. Alur, J. Crupi, D. Malks. Core J2EE Patterns – Best Practices and Design Strategies. Pearson Education, 2001.

[52] http://www.mozartspaces.org (last visited: 2008-10-10)

[53] http://www.gigaspaces.com (last visited: 2008-10-10)

[54] http://www.jboss.org (last visited: 2008-10-10)

[55] http://www.h2database.com (last visited: 2008-10-10)

[56] http://www.hsqldb.org (last visited: 2008-10-10)

[57] http://www.tripcom.org (last visited: 2008-10-10)

[58] C. Bussler. A Minimal Triple Space Computing Architecture. Proceedings of the 2nd WSMO Implementation Workshop, Innsbruck (AUT), June 2005.

8 Appendix

This chapter provides additional information for the reader. Due to the high degree of detail, it has been outsourced to the appendix.

8.1 Benchmarks

In the following the used parameters for executing the different scenarios via the benchmark suite are mentioned.

8.1.1 Serial Benchmarks

8.1.1.1 Write

MOZARTSPACES EMBEDDED (Random)

```
org.xvsm.benchmarks.BenchmarkSerial
        -datafile mozartspaces-embedded-random.dat
        -containertype EMBEDDED
        -coordinatortype RANDOM
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -iterations 60
        -operationnumber 1000
        -operationtype WRITE
        -monitorreset false
        -spacespropertyfile spaces-local.prop
```

MOZARTSPACES EMBEDDED (FIFO)

```
org.xvsm.benchmarks.BenchmarkSerial
        -datafile mozartspaces-embedded-fifo.dat
        -containertype EMBEDDED
        -coordinatortype FIFO
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -iterations 60
        -operationnumber 1000
        -operationtype WRITE
        -monitorreset false
        -spacespropertyfile spaces-local.prop
```

MOZARTSPACES EMBEDDED (LIFO)

```
org.xvsm.benchmarks.BenchmarkSerial
        -datafile mozartspaces-embedded-lifo.dat
        -containertype EMBEDDED
        -coordinatortype LIFO
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -iterations 60
        -operationnumber 1000
        -operationtype WRITE
        -monitorreset false
        -spacespropertyfile spaces-local.prop
```

MOZARTSPACES EMBEDDED (Key)

```
org.xvsm.benchmarks.BenchmarkSerial
        -datafile mozartspaces-embedded-key.dat
        -containertype EMBEDDED
```

```
        -coordinatortype KEY
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -iterations 60
        -operationnumber 1000
        -operationtype WRITE
        -monitorreset false
        -spacespropertyfile spaces-local.prop
```

MOZARTSPACES EMBEDDED (Vector)

```
org.xvsm.benchmarks.BenchmarkSerial
        -datafile mozartspaces-embedded-vector.dat
        -containertype EMBEDDED
        -coordinatortype VECTOR
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -iterations 60
        -operationnumber 1000
        -operationtype WRITE
        -monitorreset false
        -spacespropertyfile spaces-local.prop
```

MOZARTSPACES EMBEDDED (Linda)

```
org.xvsm.benchmarks.BenchmarkSerial
        -datafile mozartspaces-embedded-linda.dat
        -containertype EMBEDDED
        -coordinatortype LINDA
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -iterations 60
        -operationnumber 1000
        -operationtype WRITE
        -monitorreset false
        -spacespropertyfile spaces-local.prop
```

MOZARTSPACES REMOTE (Random)

```
org.xvsm.benchmarks.BenchmarkSerial
        -datafile mozartspaces-remote-random.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -coordinatortype RANDOM
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -iterations 60
        -operationnumber 1000
        -operationtype WRITE
        -monitorreset false
        -spacespropertyfile spaces-remote.prop
```

MOZARTSPACES REMOTE (FIFO)

```
org.xvsm.benchmarks.BenchmarkSerial
        -datafile mozartspaces-remote-fifo.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -coordinatortype FIFO
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
```

```
        -iterations 60
        -operationnumber 1000
        -operationtype WRITE
        -monitorreset false
        -spacespropertyfile spaces-remote.prop
```

MOZARTSPACES REMOTE (LIFO)

```
org.xvsm.benchmarks.BenchmarkSerial
        -datafile mozartspaces-remote-lifo.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -coordinatortype LIFO
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -iterations 60
        -operationnumber 1000
        -operationtype WRITE
        -monitorreset false
        -spacespropertyfile spaces-remote.prop
```

MOZARTSPACES REMOTE (Key)

```
org.xvsm.benchmarks.BenchmarkSerial
        -datafile mozartspaces-remote-key.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -coordinatortype KEY
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -iterations 60
        -operationnumber 1000
        -operationtype WRITE
        -monitorreset false
        -spacespropertyfile spaces-remote.prop
```

MOZARTSPACES REMOTE (Vector)

```
org.xvsm.benchmarks.BenchmarkSerial
        -datafile mozartspaces-remote-vector.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -coordinatortype VECTOR
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -iterations 60
        -operationnumber 1000
        -operationtype WRITE
        -monitorreset false
        -spacespropertyfile spaces-remote.prop
```

MOZARTSPACES REMOTE (Linda)

```
org.xvsm.benchmarks.BenchmarkSerial
        -datafile mozartspaces-remote-linda.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -coordinatortype LINDA
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -iterations 60
        -operationnumber 1000
```

```
        -operationtype WRITE
        -monitorreset false
        -spacespropertyfile spaces-remote.prop
```

GIGASPACES XAP EMBEDDED (transient)

```
net.jini.space.benchmarks.BenchmarkSerial
        -datafile gigaspaces-embedded-transient.dat
        -spacetype EMBEDDED-TRANSIENT
        -transactiontype NONE
        -atomicentrytype STRING
        -atomicentrysize 10
        -iterations 60
        -operationnumber 1000
        -operationtype WRITE
        -monitorreset false
```

GIGASPACES XAP EMBEDDED (persistent)

```
net.jini.space.benchmarks.BenchmarkSerial
        -datafile gigaspaces-embedded-persistent.dat
        -spacetype EMBEDDED-PERSISTENT
        -transactiontype NONE
        -atomicentrytype STRING
        -atomicentrysize 10
        -iterations 60
        -operationnumber 1000
        -operationtype WRITE
        -monitorreset false
```

GIGASPACES XAP REMOTE (transient)

```
net.jini.space.benchmarks.BenchmarkSerial
        -datafile gigaspaces-remote-transient.dat
        -lookupserviceurl jini://192.168.123.70:4162
        -spacetype REMOTE
        -spacename mySpaceTransient
        -transactiontype NONE
        -atomicentrytype STRING
        -atomicentrysize 10
        -iterations 60
        -operationnumber 1000
        -operationtype WRITE
        -monitorreset false
```

GIGASPACES XAP REMOTE (persistent)

```
net.jini.space.benchmarks.BenchmarkSerial
        -datafile gigaspaces-remote-persistent.dat
        -lookupserviceurl jini://192.168.123.70:4162
        -spacetype REMOTE
        -spacename mySpacePersistent
        -transactiontype NONE
        -atomicentrytype STRING
        -atomicentrysize 10
        -iterations 60
        -operationnumber 1000
        -operationtype WRITE
        -monitorreset false
```

JBOSS AS

```
javax.enterprise.benchmarks.BenchmarkSerial
        -datafile jboss.dat
        -namingproviderurl jnp://192.168.123.70:1099
        -transactiontype NONE
```

```
        -atomicentrytype STRING
        -atomicentrysize 10
        -iterations 60
        -operationnumber 1000
        -operationtype CREATE
        -monitorreset false
```

8.1.1.2 Shift

MOZARTSPACES EMBEDDED (Random)

```
org.xvsm.benchmarks.BenchmarkSerial
        -datafile mozartspaces-embedded-random.dat
        -containertype EMBEDDED
        -containersize 1000
        -coordinatortype RANDOM
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -iterations 60
        -operationnumber 1000
        -operationtype SHIFT
        -monitorreset false
        -spacespropertyfile spaces-local.prop
```

MOZARTSPACES EMBEDDED (FIFO)

```
org.xvsm.benchmarks.BenchmarkSerial
        -datafile mozartspaces-embedded-fifo.dat
        -containertype EMBEDDED
        -containersize 1000
        -coordinatortype FIFO
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -iterations 60
        -operationnumber 1000
        -operationtype SHIFT
        -monitorreset false
        -spacespropertyfile spaces-local.prop
```

MOZARTSPACES EMBEDDED (LIFO)

```
org.xvsm.benchmarks.BenchmarkSerial
        -datafile mozartspaces-embedded-lifo.dat
        -containertype EMBEDDED
        -containersize 1000
        -coordinatortype LIFO
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -iterations 60
        -operationnumber 1000
        -operationtype SHIFT
        -monitorreset false
        -spacespropertyfile spaces-local.prop
```

MOZARTSPACES EMBEDDED (Key)

```
org.xvsm.benchmarks.BenchmarkSerial
        -datafile mozartspaces-embedded-key.dat
        -containertype EMBEDDED
        -containersize 1000
        -coordinatortype KEY
        -transactiontype IMPLICIT
        -atomicentrytype STRING
```

```
        -atomicentrysize 10
        -iterations 60
        -operationnumber 1000
        -operationtype SHIFT
        -monitorreset false
        -spacespropertyfile spaces-local.prop
```

MOZARTSPACES EMBEDDED (Vector)

```
org.xvsm.benchmarks.BenchmarkSerial
        -datafile mozartspaces-embedded-vector.dat
        -containertype EMBEDDED
        -containersize 1000
        -coordinatortype VECTOR
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -iterations 60
        -operationnumber 1000
        -operationtype SHIFT
        -monitorreset false
        -spacespropertyfile spaces-local.prop
```

MOZARTSPACES EMBEDDED (Linda)

```
org.xvsm.benchmarks.BenchmarkSerial
        -datafile mozartspaces-embedded-linda.dat
        -containertype EMBEDDED
        -containersize 1000
        -coordinatortype LINDA
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -iterations 60
        -operationnumber 1000
        -operationtype SHIFT
        -monitorreset false
        -spacespropertyfile spaces-local.prop
```

MOZARTSPACES REMOTE (Random)

```
org.xvsm.benchmarks.BenchmarkSerial
        -datafile mozartspaces-remote-random.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -containersize 1000
        -coordinatortype RANDOM
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -iterations 60
        -operationnumber 1000
        -operationtype SHIFT
        -monitorreset false
        -spacespropertyfile spaces-remote.prop
```

MOZARTSPACES REMOTE (FIFO)

```
org.xvsm.benchmarks.BenchmarkSerial
        -datafile mozartspaces-remote-fifo.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -containersize 1000
        -coordinatortype FIFO
        -transactiontype IMPLICIT
        -atomicentrytype STRING
```

```
        -atomicentrysize 10
        -iterations 60
        -operationnumber 1000
        -operationtype SHIFT
        -monitorreset false
        -spacespropertyfile spaces-remote.prop
```

MOZARTSPACES REMOTE (LIFO)

```
org.xvsm.benchmarks.BenchmarkSerial
        -datafile mozartspaces-remote-lifo.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -containersize 1000
        -coordinatortype LIFO
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -iterations 60
        -operationnumber 1000
        -operationtype SHIFT
        -monitorreset false
        -spacespropertyfile spaces-remote.prop
```

MOZARTSPACES REMOTE (Key)

```
org.xvsm.benchmarks.BenchmarkSerial
        -datafile mozartspaces-remote-key.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -containersize 1000
        -coordinatortype KEY
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -iterations 60
        -operationnumber 1000
        -operationtype SHIFT
        -monitorreset false
        -spacespropertyfile spaces-remote.prop
```

MOZARTSPACES REMOTE (Vector)

```
org.xvsm.benchmarks.BenchmarkSerial
        -datafile mozartspaces-remote-vector.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -containersize 1000
        -coordinatortype VECTOR
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -iterations 60
        -operationnumber 1000
        -operationtype SHIFT
        -monitorreset false
        -spacespropertyfile spaces-remote.prop
```

MOZARTSPACES REMOTE (Linda)

```
org.xvsm.benchmarks.BenchmarkSerial
        -datafile mozartspaces-remote-linda.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -containersize 1000
        -coordinatortype LINDA
```

```
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -iterations 60
        -operationnumber 1000
        -operationtype SHIFT
        -monitorreset false
        -spacespropertyfile spaces-remote.prop
```

8.1.1.3 Read

MOZARTSPACES EMBEDDED (Random)

```
org.xvsm.benchmarks.BenchmarkSerial
        -datafile mozartspaces-embedded-random.dat
        -containertype EMBEDDED
        -coordinatortype RANDOM
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -iterations 60
        -operationnumber 1000
        -operationtype READ
        -monitorreset false
        -spacespropertyfile spaces-local.prop
```

MOZARTSPACES EMBEDDED (FIFO)

```
org.xvsm.benchmarks.BenchmarkSerial
        -datafile mozartspaces-embedded-fifo.dat
        -containertype EMBEDDED
        -coordinatortype FIFO
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -iterations 60
        -operationnumber 1000
        -operationtype READ
        -monitorreset false
        -spacespropertyfile spaces-local.prop
```

MOZARTSPACES EMBEDDED (LIFO)

```
org.xvsm.benchmarks.BenchmarkSerial
        -datafile mozartspaces-embedded-lifo.dat
        -containertype EMBEDDED
        -coordinatortype LIFO
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -iterations 60
        -operationnumber 1000
        -operationtype READ
        -monitorreset false
        -spacespropertyfile spaces-local.prop
```

MOZARTSPACES EMBEDDED (Key)

```
org.xvsm.benchmarks.BenchmarkSerial
        -datafile mozartspaces-embedded-key.dat
        -containertype EMBEDDED
        -coordinatortype KEY
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -iterations 60
```

```
        -operationnumber 1000
        -operationtype READ
        -monitorreset false
        -spacespropertyfile spaces-local.prop
```

MOZARTSPACES EMBEDDED (Vector)

```
org.xvsm.benchmarks.BenchmarkSerial
        -datafile mozartspaces-embedded-vector.dat
        -containertype EMBEDDED
        -coordinatortype VECTOR
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -iterations 60
        -operationnumber 1000
        -operationtype READ
        -monitorreset false
        -spacespropertyfile spaces-local.prop
```

MOZARTSPACES EMBEDDED (Linda)

```
org.xvsm.benchmarks.BenchmarkSerial
        -datafile mozartspaces-embedded-linda.dat
        -containertype EMBEDDED
        -coordinatortype LINDA
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -iterations 60
        -operationnumber 1000
        -operationtype READ
        -monitorreset false
        -spacespropertyfile spaces-local.prop
```

MOZARTSPACES REMOTE (Random)

```
org.xvsm.benchmarks.BenchmarkSerial
        -datafile mozartspaces-remote-random.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -coordinatortype RANDOM
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -iterations 60
        -operationnumber 1000
        -operationtype READ
        -monitorreset false
        -spacespropertyfile spaces-remote.prop
```

MOZARTSPACES REMOTE (FIFO)

```
org.xvsm.benchmarks.BenchmarkSerial
        -datafile mozartspaces-remote-fifo.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -coordinatortype FIFO
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -iterations 60
        -operationnumber 1000
        -operationtype READ
        -monitorreset false
        -spacespropertyfile spaces-remote.prop
```

MOZARTSPACES REMOTE (LIFO)

```
org.xvsm.benchmarks.BenchmarkSerial
        -datafile mozartspaces-remote-lifo.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -coordinatortype LIFO
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -iterations 60
        -operationnumber 1000
        -operationtype READ
        -monitorreset false
        -spacespropertyfile spaces-remote.prop
```

MOZARTSPACES REMOTE (Key)

```
org.xvsm.benchmarks.BenchmarkSerial
        -datafile mozartspaces-remote-key.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -coordinatortype KEY
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -iterations 60
        -operationnumber 1000
        -operationtype READ
        -monitorreset false
        -spacespropertyfile spaces-remote.prop
```

MOZARTSPACES REMOTE (Vector)

```
org.xvsm.benchmarks.BenchmarkSerial
        -datafile mozartspaces-remote-vector.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -coordinatortype VECTOR
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -iterations 60
        -operationnumber 1000
        -operationtype READ
        -monitorreset false
        -spacespropertyfile spaces-remote.prop
```

MOZARTSPACES REMOTE (Linda)

```
org.xvsm.benchmarks.BenchmarkSerial
        -datafile mozartspaces-remote-linda.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -coordinatortype LINDA
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -iterations 60
        -operationnumber 1000
        -operationtype READ
        -monitorreset false
        -spacespropertyfile spaces-remote.prop
```

GIGASPACES XAP EMBEDDED (read, transient)

```
net.jini.space.benchmarks.BenchmarkSerial
        –datafile gigaspaces-embedded-read-transient.dat
        -spacetype EMBEDDED-TRANSIENT
        -transactiontype NONE
        -atomicentrytype STRING
        -atomicentrysize 10
        -iterations 60
        -operationnumber 1000
        -operationtype READ
        -monitorreset false
```

GIGASPACES XAP EMBEDDED (read, persistent)

```
net.jini.space.benchmarks.BenchmarkSerial
        –datafile gigaspaces-embedded-read-persistent.dat
        -spacetype EMBEDDED-PERSISTENT
        -transactiontype NONE
        -atomicentrytype STRING
        -atomicentrysize 10
        -iterations 60
        -operationnumber 1000
        -operationtype READ
        -monitorreset false
```

GIGASPACES XAP EMBEDDED (readIfExists, transient)

```
net.jini.space.benchmarks.BenchmarkSerial
        –datafile gigaspaces-embedded-readifexists-transient.dat
        -spacetype EMBEDDED-TRANSIENT
        -transactiontype NONE
        -atomicentrytype STRING
        -atomicentrysize 10
        -iterations 60
        -operationnumber 1000
        -operationtype READ-IF-EXISTS
        -monitorreset false
```

GIGASPACES XAP EMBEDDED (readIfExists, persistent)

```
net.jini.space.benchmarks.BenchmarkSerial
        –datafile gigaspaces-embedded-readifexists-persistent.dat
        -spacetype EMBEDDED-PERSISTENT
        -transactiontype NONE
        -atomicentrytype STRING
        -atomicentrysize 10
        -iterations 60
        -operationnumber 1000
        -operationtype READ-IF-EXISTS
        -monitorreset false
```

GIGASPACES XAP REMOTE (read, transient)

```
net.jini.space.benchmarks.BenchmarkSerial
        -datafile gigaspaces-remote-read-transient.dat
        -lookupserviceurl jini://192.168.123.70:4162
        -spacetype REMOTE
        -spacename mySpaceTransient
        -transactiontype NONE
        -atomicentrytype STRING
        -atomicentrysize 10
        -iterations 60
        -operationnumber 1000
        -operationtype READ
        -monitorreset false
```

GIGASPACES XAP REMOTE (read, persistent)

```
net.jini.space.benchmarks.BenchmarkSerial
        -datafile gigaspaces-remote-read-persistent.dat
        -lookupserviceurl jini://192.168.123.70:4162
        -spacetype REMOTE
        -spacename mySpacePersistent
        -transactiontype NONE
        -atomicentrytype STRING
        -atomicentrysize 10
        -iterations 60
        -operationnumber 1000
        -operationtype READ
        -monitorreset false
```

GIGASPACES XAP REMOTE (readIfExists, transient)

```
net.jini.space.benchmarks.BenchmarkSerial
        -datafile gigaspaces-remote-readifexists-transient.dat
        -lookupserviceurl jini://192.168.123.70:4162
        -spacetype REMOTE
        -spacename mySpaceTransient
        -transactiontype NONE
        -atomicentrytype STRING
        -atomicentrysize 10
        -iterations 60
        -operationnumber 1000
        -operationtype READ-IF-EXISTS
        -monitorreset false
```

GIGASPACES XAP REMOTE (readIfExists, persistent)

```
net.jini.space.benchmarks.BenchmarkSerial
        -datafile gigaspaces-remote-readifexists-persistent.dat
        -lookupserviceurl jini://192.168.123.70:4162
        -spacetype REMOTE
        -spacename mySpacePersistent
        -transactiontype NONE
        -atomicentrytype STRING
        -atomicentrysize 10
        -iterations 60
        -operationnumber 1000
        -operationtype READ-IF-EXISTS
        -monitorreset false
```

JBOSS AS

```
javax.enterprise.benchmarks.BenchmarkSerial
        -datafile jboss.dat
        -namingproviderurl jnp://192.168.123.70:1099
        -transactiontype NONE
        -atomicentrytype STRING
        -atomicentrysize 10
        -iterations 60
        -operationnumber 1000
        -operationtype FIND
        -monitorreset false
```

8.1.1.4 Take

MOZARTSPACES EMBEDDED (Random)

```
org.xvsm.benchmarks.BenchmarkSerial
        -datafile mozartspaces-embedded-random.dat
        -containertype EMBEDDED
        -coordinatortype RANDOM
```

```
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -iterations 60
        -operationnumber 1000
        -operationtype TAKE
        -monitorreset false
        -spacespropertyfile spaces-local.prop
```

MOZARTSPACES EMBEDDED (FIFO)

```
org.xvsm.benchmarks.BenchmarkSerial
        -datafile mozartspaces-embedded-fifo.dat
        -containertype EMBEDDED
        -coordinatortype FIFO
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -iterations 60
        -operationnumber 1000
        -operationtype TAKE
        -monitorreset false
        -spacespropertyfile spaces-local.prop
```

MOZARTSPACES EMBEDDED (LIFO)

```
org.xvsm.benchmarks.BenchmarkSerial
        -datafile mozartspaces-embedded-lifo.dat
        -containertype EMBEDDED
        -coordinatortype LIFO
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -iterations 60
        -operationnumber 1000
        -operationtype TAKE
        -monitorreset false
        -spacespropertyfile spaces-local.prop
```

MOZARTSPACES EMBEDDED (Key)

```
org.xvsm.benchmarks.BenchmarkSerial
        -datafile mozartspaces-embedded-key.dat
        -containertype EMBEDDED
        -coordinatortype KEY
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -iterations 60
        -operationnumber 1000
        -operationtype TAKE
        -monitorreset false
        -spacespropertyfile spaces-local.prop
```

MOZARTSPACES EMBEDDED (Vector)

```
org.xvsm.benchmarks.BenchmarkSerial
        -datafile mozartspaces-embedded-vector.dat
        -containertype EMBEDDED
        -coordinatortype VECTOR
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -iterations 60
        -operationnumber 1000
        -operationtype TAKE
```

```
        -monitorreset false
        -spacespropertyfile spaces-local.prop
```

MOZARTSPACES EMBEDDED (Linda)

```
org.xvsm.benchmarks.BenchmarkSerial
        -datafile mozartspaces-embedded-linda.dat
        -containertype EMBEDDED
        -coordinatortype LINDA
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -iterations 60
        -operationnumber 1000
        -operationtype TAKE
        -monitorreset false
        -spacespropertyfile spaces-local.prop
```

MOZARTSPACES REMOTE (Random)

```
org.xvsm.benchmarks.BenchmarkSerial
        -datafile mozartspaces-remote-random.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -coordinatortype RANDOM
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -iterations 60
        -operationnumber 1000
        -operationtype TAKE
        -monitorreset false
        -spacespropertyfile spaces-remote.prop
```

MOZARTSPACES REMOTE (FIFO)

```
org.xvsm.benchmarks.BenchmarkSerial
        -datafile mozartspaces-remote-fifo.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -coordinatortype FIFO
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -iterations 60
        -operationnumber 1000
        -operationtype TAKE
        -monitorreset false
        -spacespropertyfile spaces-remote.prop
```

MOZARTSPACES REMOTE (LIFO)

```
org.xvsm.benchmarks.BenchmarkSerial
        -datafile mozartspaces-remote-lifo.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -coordinatortype LIFO
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -iterations 60
        -operationnumber 1000
        -operationtype TAKE
        -monitorreset false
        -spacespropertyfile spaces-remote.prop
```

MOZARTSPACES REMOTE (Key)

```
org.xvsm.benchmarks.BenchmarkSerial
        -datafile mozartspaces-remote-key.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -coordinatortype KEY
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -iterations 60
        -operationnumber 1000
        -operationtype TAKE
        -monitorreset false
        -spacespropertyfile spaces-remote.prop
```

MOZARTSPACES REMOTE (Vector)

```
org.xvsm.benchmarks.BenchmarkSerial
        -datafile mozartspaces-remote-vector.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -coordinatortype VECTOR
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -iterations 60
        -operationnumber 1000
        -operationtype TAKE
        -monitorreset false
        -spacespropertyfile spaces-remote.prop
```

MOZARTSPACES REMOTE (Linda)

```
org.xvsm.benchmarks.BenchmarkSerial
        -datafile mozartspaces-remote-linda.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -coordinatortype LINDA
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -iterations 60
        -operationnumber 1000
        -operationtype TAKE
        -monitorreset false
        -spacespropertyfile spaces-remote.prop
```

GIGASPACES XAP EMBEDDED (take, transient)

```
net.jini.space.benchmarks.BenchmarkSerial
        –datafile gigaspaces-embedded-take-transient.dat
        -spacetype EMBEDDED-TRANSIENT
        -transactiontype NONE
        -atomicentrytype STRING
        -atomicentrysize 10
        -iterations 60
        -operationnumber 1000
        -operationtype TAKE
        -monitorreset false
```

GIGASPACES XAP EMBEDDED (take, persistent)

```
net.jini.space.benchmarks.BenchmarkSerial
        –datafile gigaspaces-embedded-take-persistent.dat
        -spacetype EMBEDDED-PERSISTENT
```

```
        -transactiontype NONE
        -atomicentrytype STRING
        -atomicentrysize 10
        -iterations 60
        -operationnumber 1000
        -operationtype TAKE
        -monitorreset false
```

GIGASPACES XAP EMBEDDED (takeIfExists, transient)

```
net.jini.space.benchmarks.BenchmarkSerial
        –datafile gigaspaces-embedded-takeifexists-transient.dat
        -spacetype EMBEDDED-TRANSIENT
        -transactiontype NONE
        -atomicentrytype STRING
        -atomicentrysize 10
        -iterations 60
        -operationnumber 1000
        -operationtype TAKE-IF-EXISTS
        -monitorreset false
```

GIGASPACES XAP EMBEDDED (takeIfExists, persistent)

```
net.jini.space.benchmarks.BenchmarkSerial
        –datafile gigaspaces-embedded-takeifexists-persistent.dat
        -spacetype EMBEDDED-PERSISTENT
        -transactiontype NONE
        -atomicentrytype STRING
        -atomicentrysize 10
        -iterations 60
        -operationnumber 1000
        -operationtype TAKE-IF-EXISTS
        -monitorreset false
```

GIGASPACES XAP REMOTE (take, transient)

```
net.jini.space.benchmarks.BenchmarkSerial
        -datafile gigaspaces-remote-take-transient.dat
        -lookupserviceurl jini://192.168.123.70:4162
        -spacetype REMOTE
        -spacename mySpaceTransient
        -transactiontype NONE
        -atomicentrytype STRING
        -atomicentrysize 10
        -iterations 60
        -operationnumber 1000
        -operationtype TAKE
        -monitorreset false
```

GIGASPACES XAP REMOTE (take, persistent)

```
net.jini.space.benchmarks.BenchmarkSerial
        -datafile gigaspaces-remote-take-persistent.dat
        -lookupserviceurl jini://192.168.123.70:4162
        -spacetype REMOTE
        -spacename mySpacePersistent
        -transactiontype NONE
        -atomicentrytype STRING
        -atomicentrysize 10
        -iterations 60
        -operationnumber 1000
        -operationtype TAKE
        -monitorreset false
```

GIGASPACES XAP REMOTE (takeIfExists, transient)

```
net.jini.space.benchmarks.BenchmarkSerial
        -datafile gigaspaces-remote-takeifexists-transient.dat
        -lookupserviceurl jini://192.168.123.70:4162
        -spacetype REMOTE
        -spacename mySpaceTransient
        -transactiontype NONE
        -atomicentrytype STRING
        -atomicentrysize 10
        -iterations 60
        -operationnumber 1000
        -operationtype TAKE-IF-EXISTS
        -monitorreset false
```

GIGASPACES XAP REMOTE (takeIfExists, persistent)

```
net.jini.space.benchmarks.BenchmarkSerial
        -datafile gigaspaces-remote-takeifexists-persistent.dat
        -lookupserviceurl jini://192.168.123.70:4162
        -spacetype REMOTE
        -spacename mySpacePersistent
        -transactiontype NONE
        -atomicentrytype STRING
        -atomicentrysize 10
        -iterations 60
        -operationnumber 1000
        -operationtype TAKE-IF-EXISTS
        -monitorreset false
```

8.1.1.5 Destroy

MOZARTSPACES EMBEDDED (Random)

```
org.xvsm.benchmarks.BenchmarkSerial
        -datafile mozartspaces-embedded-random.dat
        -containertype EMBEDDED
        -coordinatortype RANDOM
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -iterations 60
        -operationnumber 1000
        -operationtype DESTROY
        -monitorreset false
        -spacespropertyfile spaces-local.prop
```

MOZARTSPACES EMBEDDED (FIFO)

```
org.xvsm.benchmarks.BenchmarkSerial
        -datafile mozartspaces-embedded-fifo.dat
        -containertype EMBEDDED
        -coordinatortype FIFO
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -iterations 60
        -operationnumber 1000
        -operationtype DESTROY
        -monitorreset false
        -spacespropertyfile spaces-local.prop
```

MOZARTSPACES EMBEDDED (LIFO)

```
org.xvsm.benchmarks.BenchmarkSerial
        -datafile mozartspaces-embedded-lifo.dat
```

```
        -containertype EMBEDDED
        -coordinatortype LIFO
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -iterations 60
        -operationnumber 1000
        -operationtype DESTROY
        -monitorreset false
        -spacespropertyfile spaces-local.prop
```

MOZARTSPACES EMBEDDED (Key)

```
org.xvsm.benchmarks.BenchmarkSerial
        -datafile mozartspaces-embedded-key.dat
        -containertype EMBEDDED
        -coordinatortype KEY
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -iterations 60
        -operationnumber 1000
        -operationtype DESTROY
        -monitorreset false
        -spacespropertyfile spaces-local.prop
```

MOZARTSPACES EMBEDDED (Vector)

```
org.xvsm.benchmarks.BenchmarkSerial
        -datafile mozartspaces-embedded-vector.dat
        -containertype EMBEDDED
        -coordinatortype VECTOR
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -iterations 60
        -operationnumber 1000
        -operationtype DESTROY
        -monitorreset false
        -spacespropertyfile spaces-local.prop
```

MOZARTSPACES EMBEDDED (Linda)

```
org.xvsm.benchmarks.BenchmarkSerial
        -datafile mozartspaces-embedded-linda.dat
        -containertype EMBEDDED
        -coordinatortype LINDA
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -iterations 60
        -operationnumber 1000
        -operationtype DESTROY
        -monitorreset false
        -spacespropertyfile spaces-local.prop
```

MOZARTSPACES REMOTE (Random)

```
org.xvsm.benchmarks.BenchmarkSerial
        -datafile mozartspaces-remote-random.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -coordinatortype RANDOM
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
```

```
        -iterations 60
        -operationnumber 1000
        -operationtype DESTROY
        -monitorreset false
        -spacespropertyfile spaces-remote.prop
```

MOZARTSPACES REMOTE (FIFO)

```
org.xvsm.benchmarks.BenchmarkSerial
        -datafile mozartspaces-remote-fifo.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -coordinatortype FIFO
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -iterations 60
        -operationnumber 1000
        -operationtype DESTROY
        -monitorreset false
        -spacespropertyfile spaces-remote.prop
```

MOZARTSPACES REMOTE (LIFO)

```
org.xvsm.benchmarks.BenchmarkSerial
        -datafile mozartspaces-remote-lifo.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -coordinatortype LIFO
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -iterations 60
        -operationnumber 1000
        -operationtype DESTROY
        -monitorreset false
        -spacespropertyfile spaces-remote.prop
```

MOZARTSPACES REMOTE (Key)

```
org.xvsm.benchmarks.BenchmarkSerial
        -datafile mozartspaces-remote-key.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -coordinatortype KEY
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -iterations 60
        -operationnumber 1000
        -operationtype DESTROY
        -monitorreset false
        -spacespropertyfile spaces-remote.prop
```

MOZARTSPACES REMOTE (Vector)

```
org.xvsm.benchmarks.BenchmarkSerial
        -datafile mozartspaces-remote-vector.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -coordinatortype VECTOR
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -iterations 60
        -operationnumber 1000
```

```
        -operationtype DESTROY
        -monitorreset false
        -spacespropertyfile spaces-remote.prop
```

MOZARTSPACES REMOTE (Linda)

```
org.xvsm.benchmarks.BenchmarkSerial
        -datafile mozartspaces-remote-linda.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -coordinatortype LINDA
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -iterations 60
        -operationnumber 1000
        -operationtype DESTROY
        -monitorreset false
        -spacespropertyfile spaces-remote.prop
```

JBOSS AS

```
javax.enterprise.benchmarks.BenchmarkSerial
        -datafile jboss.dat
        -namingproviderurl jnp://192.168.123.70:1099
        -transactiontype NONE
        -atomicentrytype STRING
        -atomicentrysize 10
        -iterations 60
        -operationnumber 1000
        -operationtype REMOVE
        -monitorreset false
```

8.1.2 Concurrent Benchmarks

8.1.2.1 Write

MOZARTSPACES (Random)

```
org.xvsm.benchmarks.BenchmarkConcurrent
        -datafile mozartspaces-random-1fold.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -coordinatortype RANDOM
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 20
        -iterations 1
        -operationnumber 1000
        -operationtype WRITE
        -monitorreset false
        -spacespropertyfile spaces-remote.prop
```

```
org.xvsm.benchmarks.BenchmarkConcurrent
        -datafile mozartspaces-random-2fold.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -coordinatortype RANDOM
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 40
        -iterations 1
```

```
        -operationnumber 1000
        -operationtype WRITE
        -monitorreset false
        -spacespropertyfile spaces-remote.prop
```

```
org.xvsm.benchmarks.BenchmarkConcurrent
        -datafile mozartspaces-random-3fold.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -coordinatortype RANDOM
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 60
        -iterations 1
        -operationnumber 1000
        -operationtype WRITE
        -monitorreset false
        -spacespropertyfile spaces-remote.prop
```

MOZARTSPACES (FIFO)

```
org.xvsm.benchmarks.BenchmarkConcurrent
        -datafile mozartspaces-fifo-1fold.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -coordinatortype FIFO
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 20
        -iterations 1
        -operationnumber 1000
        -operationtype WRITE
        -monitorreset false
        -spacespropertyfile spaces-remote.prop
```

```
org.xvsm.benchmarks.BenchmarkConcurrent
        -datafile mozartspaces-fifo-2fold.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -coordinatortype FIFO
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 40
        -iterations 1
        -operationnumber 1000
        -operationtype WRITE
        -monitorreset false
        -spacespropertyfile spaces-remote.prop
```

```
org.xvsm.benchmarks.BenchmarkConcurrent
        -datafile mozartspaces-fifo-3fold.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -coordinatortype FIFO
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 60
        -iterations 1
        -operationnumber 1000
        -operationtype WRITE
```

```
        -monitorreset false
        -spacespropertyfile spaces-remote.prop
```

MOZARTSPACES (LIFO)

```
org.xvsm.benchmarks.BenchmarkConcurrent
        -datafile mozartspaces-lifo-1fold.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -coordinatortype LIFO
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 20
        -iterations 1
        -operationnumber 1000
        -operationtype WRITE
        -monitorreset false
        -spacespropertyfile spaces-remote.prop
```

```
org.xvsm.benchmarks.BenchmarkConcurrent
        -datafile mozartspaces-lifo-2fold.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -coordinatortype LIFO
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 40
        -iterations 1
        -operationnumber 1000
        -operationtype WRITE
        -monitorreset false
        -spacespropertyfile spaces-remote.prop
```

```
org.xvsm.benchmarks.BenchmarkConcurrent
        -datafile mozartspaces-lifo-3fold.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -coordinatortype LIFO
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 60
        -iterations 1
        -operationnumber 1000
        -operationtype WRITE
        -monitorreset false
        -spacespropertyfile spaces-remote.prop
```

MOZARTSPACES (Key)

```
org.xvsm.benchmarks.BenchmarkConcurrent
        -datafile mozartspaces-key-1fold.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -coordinatortype KEY
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 20
        -iterations 1
        -operationnumber 1000
        -operationtype WRITE
        -monitorreset false
```

```
        -spacespropertyfile spaces-remote.prop
```

```
org.xvsm.benchmarks.BenchmarkConcurrent
        -datafile mozartspaces-key-2fold.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -coordinatortype KEY
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 40
        -iterations 1
        -operationnumber 1000
        -operationtype WRITE
        -monitorreset false
        -spacespropertyfile spaces-remote.prop
```

```
org.xvsm.benchmarks.BenchmarkConcurrent
        -datafile mozartspaces-key-3fold.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -coordinatortype KEY
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 60
        -iterations 1
        -operationnumber 1000
        -operationtype WRITE
        -monitorreset false
        -spacespropertyfile spaces-remote.prop
```

MOZARTSPACES (Vector)

```
org.xvsm.benchmarks.BenchmarkConcurrent
        -datafile mozartspaces-vector-1fold.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -coordinatortype VECTOR
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 20
        -iterations 1
        -operationnumber 1000
        -operationtype WRITE
        -monitorreset false
        -spacespropertyfile spaces-remote.prop
```

```
org.xvsm.benchmarks.BenchmarkConcurrent
        -datafile mozartspaces-vector-2fold.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -coordinatortype VECTOR
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 40
        -iterations 1
        -operationnumber 1000
        -operationtype WRITE
        -monitorreset false
        -spacespropertyfile spaces-remote.prop
```

```
org.xvsm.benchmarks.BenchmarkConcurrent
        -datafile mozartspaces-vector-3fold.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -coordinatortype VECTOR
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 60
        -iterations 1
        -operationnumber 1000
        -operationtype WRITE
        -monitorreset false
        -spacespropertyfile spaces-remote.prop
```

MOZARTSPACES (Linda)

```
org.xvsm.benchmarks.BenchmarkConcurrent
        -datafile mozartspaces-linda-1fold.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -coordinatortype LINDA
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 20
        -iterations 1
        -operationnumber 1000
        -operationtype WRITE
        -monitorreset false
        -spacespropertyfile spaces-remote.prop
```

```
org.xvsm.benchmarks.BenchmarkConcurrent
        -datafile mozartspaces-linda-2fold.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -coordinatortype LINDA
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 40
        -iterations 1
        -operationnumber 1000
        -operationtype WRITE
        -monitorreset false
        -spacespropertyfile spaces-remote.prop
```

```
org.xvsm.benchmarks.BenchmarkConcurrent
        -datafile mozartspaces-linda-3fold.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -coordinatortype LINDA
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 60
        -iterations 1
        -operationnumber 1000
        -operationtype WRITE
        -monitorreset false
        -spacespropertyfile spaces-remote.prop
```

GIGASPACES XAP (transient)

```
net.jini.space.benchmarks.BenchmarkConcurrent
```

```
        -datafile gigaspaces-transient-1fold.dat
        -lookupserviceurl jini://192.168.123.70:4162
        -spacetype REMOTE
        -spacename mySpaceTransient
        -transactiontype NONE
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 20
        -iterations 1
        -operationnumber 1000
        -operationtype WRITE
        -monitorreset false
```

```
net.jini.space.benchmarks.BenchmarkConcurrent
        -datafile gigaspaces-transient-2fold.dat
        -lookupserviceurl jini://192.168.123.70:4162
        -spacetype REMOTE
        -spacename mySpaceTransient
        -transactiontype NONE
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 40
        -iterations 1
        -operationnumber 1000
        -operationtype WRITE
        -monitorreset false
```

```
net.jini.space.benchmarks.BenchmarkConcurrent
        -datafile gigaspaces-transient-3fold.dat
        -lookupserviceurl jini://192.168.123.70:4162
        -spacetype REMOTE
        -spacename mySpaceTransient
        -transactiontype NONE
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 60
        -iterations 1
        -operationnumber 1000
        -operationtype WRITE
        -monitorreset false
```

GIGASPACES XAP (persistent)

```
net.jini.space.benchmarks.BenchmarkConcurrent
        -datafile gigaspaces-persistent-1fold.dat
        -lookupserviceurl jini://192.168.123.70:4162
        -spacetype REMOTE
        -spacename mySpacePersistent
        -transactiontype NONE
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 20
        -iterations 1
        -operationnumber 1000
        -operationtype WRITE
        -monitorreset false
```

```
net.jini.space.benchmarks.BenchmarkConcurrent
        -datafile gigaspaces-persistent-2fold.dat
        -lookupserviceurl jini://192.168.123.70:4162
        -spacetype REMOTE
        -spacename mySpacePersistent
        -transactiontype NONE
        -atomicentrytype STRING
```

```
        -atomicentrysize 10
        -threads 40
        -iterations 1
        -operationnumber 1000
        -operationtype WRITE
        -monitorreset false
```

```
net.jini.space.benchmarks.BenchmarkConcurrent
        -datafile gigaspaces-persistent-3fold.dat
        -lookupserviceurl jini://192.168.123.70:4162
        -spacetype REMOTE
        -spacename mySpacePersistent
        -transactiontype NONE
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 60
        -iterations 1
        -operationnumber 1000
        -operationtype WRITE
        -monitorreset false
```

JBOSS AS

```
javax.enterprise.benchmarks.BenchmarkConcurrent
        -datafile jboss-1fold.dat
        -namingproviderurl jnp://192.168.123.70:1099
        -transactiontype NONE
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 20
        -iterations 1
        -operationnumber 1000
        -operationtype CREATE
        -monitorreset false
```

```
javax.enterprise.benchmarks.BenchmarkConcurrent
        -datafile jboss-2fold.dat
        -namingproviderurl jnp://192.168.123.70:1099
        -transactiontype NONE
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 40
        -iterations 1
        -operationnumber 1000
        -operationtype CREATE
        -monitorreset false
```

```
javax.enterprise.benchmarks.BenchmarkConcurrent
        -datafile jboss-3fold.dat
        -namingproviderurl jnp://192.168.123.70:1099
        -transactiontype NONE
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 60
        -iterations 1
        -operationnumber 1000
        -operationtype CREATE
        -monitorreset false
```

8.1.2.2 Shift

MOZARTSPACES (Random)

```
org.xvsm.benchmarks.BenchmarkConcurrent
```

```
        -datafile mozartspaces-random-1fold.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -containersize 1000
        -coordinatortype RANDOM
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 20
        -iterations 1
        -operationnumber 1000
        -operationtype SHIFT
        -monitorreset false
        -spacespropertyfile spaces-remote.prop
```

```
org.xvsm.benchmarks.BenchmarkConcurrent
        -datafile mozartspaces-random-2fold.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -containersize 1000
        -coordinatortype RANDOM
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 40
        -iterations 1
        -operationnumber 1000
        -operationtype SHIFT
        -monitorreset false
        -spacespropertyfile spaces-remote.prop
```

```
org.xvsm.benchmarks.BenchmarkConcurrent
        -datafile mozartspaces-random-3fold.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -containersize 1000
        -coordinatortype RANDOM
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 60
        -iterations 1
        -operationnumber 1000
        -operationtype SHIFT
        -monitorreset false
        -spacespropertyfile spaces-remote.prop
```

MOZARTSPACES (FIFO)

```
org.xvsm.benchmarks.BenchmarkConcurrent
        -datafile mozartspaces-fifo-1fold.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -containersize 1000
        -coordinatortype FIFO
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 20
        -iterations 1
        -operationnumber 1000
        -operationtype SHIFT
        -monitorreset false
        -spacespropertyfile spaces-remote.prop
```

```
org.xvsm.benchmarks.BenchmarkConcurrent
        -datafile mozartspaces-fifo-2fold.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -containersize 1000
        -coordinatortype FIFO
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 40
        -iterations 1
        -operationnumber 1000
        -operationtype SHIFT
        -monitorreset false
        -spacespropertyfile spaces-remote.prop
```

```
org.xvsm.benchmarks.BenchmarkConcurrent
        -datafile mozartspaces-fifo-3fold.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -containersize 1000
        -coordinatortype FIFO
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 60
        -iterations 1
        -operationnumber 1000
        -operationtype SHIFT
        -monitorreset false
        -spacespropertyfile spaces-remote.prop
```

MOZARTSPACES (LIFO)

```
org.xvsm.benchmarks.BenchmarkConcurrent
        -datafile mozartspaces-lifo-1fold.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -containersize 1000
        -coordinatortype LIFO
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 20
        -iterations 1
        -operationnumber 1000
        -operationtype SHIFT
        -monitorreset false
        -spacespropertyfile spaces-remote.prop
```

```
org.xvsm.benchmarks.BenchmarkConcurrent
        -datafile mozartspaces-lifo-2fold.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -containersize 1000
        -coordinatortype LIFO
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 40
        -iterations 1
        -operationnumber 1000
        -operationtype SHIFT
```

```
        -monitorreset false
        -spacespropertyfile spaces-remote.prop
```

```
org.xvsm.benchmarks.BenchmarkConcurrent
        -datafile mozartspaces-lifo-3fold.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -containersize 1000
        -coordinatortype LIFO
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 60
        -iterations 1
        -operationnumber 1000
        -operationtype SHIFT
        -monitorreset false
        -spacespropertyfile spaces-remote.prop
```

MOZARTSPACES (Key)

```
org.xvsm.benchmarks.BenchmarkConcurrent
        -datafile mozartspaces-key-1fold.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -containersize 1000
        -coordinatortype KEY
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 20
        -iterations 1
        -operationnumber 1000
        -operationtype SHIFT
        -monitorreset false
        -spacespropertyfile spaces-remote.prop
```

```
org.xvsm.benchmarks.BenchmarkConcurrent
        -datafile mozartspaces-key-2fold.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -containersize 1000
        -coordinatortype KEY
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 40
        -iterations 1
        -operationnumber 1000
        -operationtype SHIFT
        -monitorreset false
        -spacespropertyfile spaces-remote.prop
```

```
org.xvsm.benchmarks.BenchmarkConcurrent
        -datafile mozartspaces-key-3fold.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -containersize 1000
        -coordinatortype KEY
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 60
        -iterations 1
```

```
        -operationnumber 1000
        -operationtype SHIFT
        -monitorreset false
        -spacespropertyfile spaces-remote.prop
```

MOZARTSPACES (Vector)

```
org.xvsm.benchmarks.BenchmarkConcurrent
        -datafile mozartspaces-vector-1fold.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -containersize 1000
        -coordinatortype VECTOR
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 20
        -iterations 1
        -operationnumber 1000
        -operationtype SHIFT
        -monitorreset false
        -spacespropertyfile spaces-remote.prop
```

```
org.xvsm.benchmarks.BenchmarkConcurrent
        -datafile mozartspaces-vector-2fold.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -containersize 1000
        -coordinatortype VECTOR
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 40
        -iterations 1
        -operationnumber 1000
        -operationtype SHIFT
        -monitorreset false
        -spacespropertyfile spaces-remote.prop
```

```
org.xvsm.benchmarks.BenchmarkConcurrent
        -datafile mozartspaces-vector-3fold.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -containersize 1000
        -coordinatortype VECTOR
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 60
        -iterations 1
        -operationnumber 1000
        -operationtype SHIFT
        -monitorreset false
        -spacespropertyfile spaces-remote.prop
```

MOZARTSPACES (Linda)

```
org.xvsm.benchmarks.BenchmarkConcurrent
        -datafile mozartspaces-linda-1fold.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -containersize 1000
        -coordinatortype LINDA
        -transactiontype IMPLICIT
        -atomicentrytype STRING
```

```
        -atomicentrysize 10
        -threads 20
        -iterations 1
        -operationnumber 1000
        -operationtype SHIFT
        -monitorreset false
        -spacespropertyfile spaces-remote.prop
```

```
org.xvsm.benchmarks.BenchmarkConcurrent
        -datafile mozartspaces-linda-2fold.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -containersize 1000
        -coordinatortype LINDA
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 40
        -iterations 1
        -operationnumber 1000
        -operationtype SHIFT
        -monitorreset false
        -spacespropertyfile spaces-remote.prop
```

```
org.xvsm.benchmarks.BenchmarkConcurrent
        -datafile mozartspaces-linda-3fold.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -containersize 1000
        -coordinatortype LINDA
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 60
        -iterations 1
        -operationnumber 1000
        -operationtype SHIFT
        -monitorreset false
        -spacespropertyfile spaces-remote.prop
```

8.1.2.3 Read

MOZARTSPACES (Random)

```
org.xvsm.benchmarks.BenchmarkConcurrent
        -datafile mozartspaces-random-1fold.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -coordinatortype RANDOM
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 20
        -iterations 1
        -operationnumber 1000
        -operationtype READ
        -monitorreset false
        -spacespropertyfile spaces-remote.prop
```

```
org.xvsm.benchmarks.BenchmarkConcurrent
        -datafile mozartspaces-random-2fold.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -coordinatortype RANDOM
```

```
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 40
        -iterations 1
        -operationnumber 1000
        -operationtype READ
        -monitorreset false
        -spacespropertyfile spaces-remote.prop
```

```
org.xvsm.benchmarks.BenchmarkConcurrent
        -datafile mozartspaces-random-3fold.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -coordinatortype RANDOM
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 60
        -iterations 1
        -operationnumber 1000
        -operationtype READ
        -monitorreset false
        -spacespropertyfile spaces-remote.prop
```

MOZARTSPACES (FIFO)

```
org.xvsm.benchmarks.BenchmarkConcurrent
        -datafile mozartspaces-fifo-1fold.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -coordinatortype FIFO
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 20
        -iterations 1
        -operationnumber 1000
        -operationtype READ
        -monitorreset false
        -spacespropertyfile spaces-remote.prop
```

```
org.xvsm.benchmarks.BenchmarkConcurrent
        -datafile mozartspaces-fifo-2fold.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -coordinatortype FIFO
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 40
        -iterations 1
        -operationnumber 1000
        -operationtype READ
        -monitorreset false
        -spacespropertyfile spaces-remote.prop
```

```
org.xvsm.benchmarks.BenchmarkConcurrent
        -datafile mozartspaces-fifo-3fold.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -coordinatortype FIFO
        -transactiontype IMPLICIT
        -atomicentrytype STRING
```

```
        -atomicentrysize 10
        -threads 60
        -iterations 1
        -operationnumber 1000
        -operationtype READ
        -monitorreset false
        -spacespropertyfile spaces-remote.prop
```

MOZARTSPACES (LIFO)

```
org.xvsm.benchmarks.BenchmarkConcurrent
        -datafile mozartspaces-lifo-1fold.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -coordinatortype LIFO
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 20
        -iterations 1
        -operationnumber 1000
        -operationtype READ
        -monitorreset false
        -spacespropertyfile spaces-remote.prop
```

```
org.xvsm.benchmarks.BenchmarkConcurrent
        -datafile mozartspaces-lifo-2fold.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -coordinatortype LIFO
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 40
        -iterations 1
        -operationnumber 1000
        -operationtype READ
        -monitorreset false
        -spacespropertyfile spaces-remote.prop
```

```
org.xvsm.benchmarks.BenchmarkConcurrent
        -datafile mozartspaces-lifo-3fold.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -coordinatortype LIFO
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 60
        -iterations 1
        -operationnumber 1000
        -operationtype READ
        -monitorreset false
        -spacespropertyfile spaces-remote.prop
```

MOZARTSPACES (Key)

```
org.xvsm.benchmarks.BenchmarkConcurrent
        -datafile mozartspaces-key-1fold.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -coordinatortype KEY
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
```

```
        -threads 20
        -iterations 1
        -operationnumber 1000
        -operationtype READ
        -monitorreset false
        -spacespropertyfile spaces-remote.prop
```

```
org.xvsm.benchmarks.BenchmarkConcurrent
        -datafile mozartspaces-key-2fold.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -coordinatortype KEY
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 40
        -iterations 1
        -operationnumber 1000
        -operationtype READ
        -monitorreset false
        -spacespropertyfile spaces-remote.prop
```

```
org.xvsm.benchmarks.BenchmarkConcurrent
        -datafile mozartspaces-key-3fold.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -coordinatortype KEY
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 60
        -iterations 1
        -operationnumber 1000
        -operationtype READ
        -monitorreset false
        -spacespropertyfile spaces-remote.prop
```

MOZARTSPACES (Vector)

```
org.xvsm.benchmarks.BenchmarkConcurrent
        -datafile mozartspaces-vector-1fold.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -coordinatortype VECTOR
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 20
        -iterations 1
        -operationnumber 1000
        -operationtype READ
        -monitorreset false
        -spacespropertyfile spaces-remote.prop
```

```
org.xvsm.benchmarks.BenchmarkConcurrent
        -datafile mozartspaces-vector-2fold.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -coordinatortype VECTOR
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 40
        -iterations 1
```

```
        -operationnumber 1000
        -operationtype READ
        -monitorreset false
        -spacespropertyfile spaces-remote.prop
```

```
org.xvsm.benchmarks.BenchmarkConcurrent
        -datafile mozartspaces-vector-3fold.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -coordinatortype VECTOR
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 60
        -iterations 1
        -operationnumber 1000
        -operationtype READ
        -monitorreset false
        -spacespropertyfile spaces-remote.prop
```

MOZARTSPACES (Linda)

```
org.xvsm.benchmarks.BenchmarkConcurrent
        -datafile mozartspaces-linda-1fold.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -coordinatortype LINDA
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 20
        -iterations 1
        -operationnumber 1000
        -operationtype READ
        -monitorreset false
        -spacespropertyfile spaces-remote.prop
```

```
org.xvsm.benchmarks.BenchmarkConcurrent
        -datafile mozartspaces-linda-2fold.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -coordinatortype LINDA
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 40
        -iterations 1
        -operationnumber 1000
        -operationtype READ
        -monitorreset false
        -spacespropertyfile spaces-remote.prop
```

```
org.xvsm.benchmarks.BenchmarkConcurrent
        -datafile mozartspaces-linda-3fold.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -coordinatortype LINDA
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 60
        -iterations 1
        -operationnumber 1000
        -operationtype READ
```

```
        -monitorreset false
        -spacespropertyfile spaces-remote.prop
```

GIGASPACES XAP (read, transient)

```
net.jini.space.benchmarks.BenchmarkConcurrent
        -datafile gigaspaces-read-transient-1fold.dat
        -lookupserviceurl jini://192.168.123.70:4162
        -spacetype REMOTE
        -spacename mySpaceTransient
        -transactiontype NONE
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 20
        -iterations 1
        -operationnumber 1000
        -operationtype READ
        -monitorreset false
```

```
net.jini.space.benchmarks.BenchmarkConcurrent
        -datafile gigaspaces-read-transient-2fold.dat
        -lookupserviceurl jini://192.168.123.70:4162
        -spacetype REMOTE
        -spacename mySpaceTransient
        -transactiontype NONE
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 40
        -iterations 1
        -operationnumber 1000
        -operationtype READ
        -monitorreset false
```

```
net.jini.space.benchmarks.BenchmarkConcurrent
        -datafile gigaspaces-read-transient-3fold.dat
        -lookupserviceurl jini://192.168.123.70:4162
        -spacetype REMOTE
        -spacename mySpaceTransient
        -transactiontype NONE
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 60
        -iterations 1
        -operationnumber 1000
        -operationtype READ
        -monitorreset false
```

GIGASPACES XAP (read, persistent)

```
net.jini.space.benchmarks.BenchmarkConcurrent
        -datafile gigaspaces-read-persistent-1fold.dat
        -lookupserviceurl jini://192.168.123.70:4162
        -spacetype REMOTE
        -spacename mySpacePersistent
        -transactiontype NONE
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 20
        -iterations 1
        -operationnumber 1000
        -operationtype READ
        -monitorreset false
```

```
net.jini.space.benchmarks.BenchmarkConcurrent
```

```
        -datafile gigaspaces-read-persistent-2fold.dat
        -lookupserviceurl jini://192.168.123.70:4162
        -spacetype REMOTE
        -spacename mySpacePersistent
        -transactiontype NONE
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 40
        -iterations 1
        -operationnumber 1000
        -operationtype READ
        -monitorreset false
```

```
net.jini.space.benchmarks.BenchmarkConcurrent
        -datafile gigaspaces-read-persistent-3fold.dat
        -lookupserviceurl jini://192.168.123.70:4162
        -spacetype REMOTE
        -spacename mySpacePersistent
        -transactiontype NONE
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 60
        -iterations 1
        -operationnumber 1000
        -operationtype READ
        -monitorreset false
```

GIGASPACES XAP (readIfExists, transient)

```
net.jini.space.benchmarks.BenchmarkConcurrent
        -datafile gigaspaces-readifexists-transient-1fold.dat
        -lookupserviceurl jini://192.168.123.70:4162
        -spacetype REMOTE
        -spacename mySpaceTransient
        -transactiontype NONE
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 20
        -iterations 1
        -operationnumber 1000
        -operationtype READ-IF-EXISTS
        -monitorreset false
```

```
net.jini.space.benchmarks.BenchmarkConcurrent
        -datafile gigaspaces-readifexists-transient-2fold.dat
        -lookupserviceurl jini://192.168.123.70:4162
        -spacetype REMOTE
        -spacename mySpaceTransient
        -transactiontype NONE
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 40
        -iterations 1
        -operationnumber 1000
        -operationtype READ-IF-EXISTS
        -monitorreset false
```

```
net.jini.space.benchmarks.BenchmarkConcurrent
        -datafile gigaspaces-readifexists-transient-3fold.dat
        -lookupserviceurl jini://192.168.123.70:4162
        -spacetype REMOTE
        -spacename mySpaceTransient
        -transactiontype NONE
        -atomicentrytype STRING
```

```
        -atomicentrysize 10
        -threads 60
        -iterations 1
        -operationnumber 1000
        -operationtype READ-IF-EXISTS
        -monitorreset false
```

GIGASPACES XAP (readIfExists, persistent)

```
net.jini.space.benchmarks.BenchmarkConcurrent
        -datafile gigaspaces-readifexists-transient-1fold.dat
        -lookupserviceurl jini://192.168.123.70:4162
        -spacetype REMOTE
        -spacename mySpacePersistent
        -transactiontype NONE
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 20
        -iterations 1
        -operationnumber 1000
        -operationtype READ-IF-EXISTS
        -monitorreset false
```

```
net.jini.space.benchmarks.BenchmarkConcurrent
        -datafile gigaspaces-readifexists-transient-2fold.dat
        -lookupserviceurl jini://192.168.123.70:4162
        -spacetype REMOTE
        -spacename mySpacePersistent
        -transactiontype NONE
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 40
        -iterations 1
        -operationnumber 1000
        -operationtype READ-IF-EXISTS
        -monitorreset false
```

```
net.jini.space.benchmarks.BenchmarkConcurrent
        -datafile gigaspaces-readifexists-transient-3fold.dat
        -lookupserviceurl jini://192.168.123.70:4162
        -spacetype REMOTE
        -spacename mySpacePersistent
        -transactiontype NONE
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 60
        -iterations 1
        -operationnumber 1000
        -operationtype READ-IF-EXISTS
        -monitorreset false
```

JBOSS AS

```
javax.enterprise.benchmarks.BenchmarkConcurrent
        -datafile jboss-1fold.dat
        -namingproviderurl jnp://192.168.123.70:1099
        -transactiontype NONE
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 20
        -iterations 1
        -operationnumber 1000
        -operationtype FIND
        -monitorreset false
```

```
javax.enterprise.benchmarks.BenchmarkConcurrent
        -datafile jboss-2fold.dat
        -namingproviderurl jnp://192.168.123.70:1099
        -transactiontype NONE
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 40
        -iterations 1
        -operationnumber 1000
        -operationtype FIND
        -monitorreset false
```

```
javax.enterprise.benchmarks.BenchmarkConcurrent
        -datafile jboss-3fold.dat
        -namingproviderurl jnp://192.168.123.70:1099
        -transactiontype NONE
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 60
        -iterations 1
        -operationnumber 1000
        -operationtype FIND
        -monitorreset false
```

8.1.2.4 Take

MOZARTSPACES (Random)

```
org.xvsm.benchmarks.BenchmarkConcurrent
        -datafile mozartspaces-random-1fold.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -coordinatortype RANDOM
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 20
        -iterations 1
        -operationnumber 1000
        -operationtype TAKE
        -monitorreset false
        -spacespropertyfile spaces-remote.prop
```

```
org.xvsm.benchmarks.BenchmarkConcurrent
        -datafile mozartspaces-random-2fold.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -coordinatortype RANDOM
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 40
        -iterations 1
        -operationnumber 1000
        -operationtype TAKE
        -monitorreset false
        -spacespropertyfile spaces-remote.prop
```

```
org.xvsm.benchmarks.BenchmarkConcurrent
        -datafile mozartspaces-random-3fold.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
```

```
        -coordinatortype RANDOM
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 60
        -iterations 1
        -operationnumber 1000
        -operationtype TAKE
        -monitorreset false
        -spacespropertyfile spaces-remote.prop
```

MOZARTSPACES (FIFO)

```
org.xvsm.benchmarks.BenchmarkConcurrent
        -datafile mozartspaces-fifo-1fold.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -coordinatortype FIFO
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 20
        -iterations 1
        -operationnumber 1000
        -operationtype TAKE
        -monitorreset false
        -spacespropertyfile spaces-remote.prop
```

```
org.xvsm.benchmarks.BenchmarkConcurrent
        -datafile mozartspaces-fifo-2fold.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -coordinatortype FIFO
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 40
        -iterations 1
        -operationnumber 1000
        -operationtype TAKE
        -monitorreset false
        -spacespropertyfile spaces-remote.prop
```

```
org.xvsm.benchmarks.BenchmarkConcurrent
        -datafile mozartspaces-fifo-3fold.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -coordinatortype FIFO
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 60
        -iterations 1
        -operationnumber 1000
        -operationtype TAKE
        -monitorreset false
        -spacespropertyfile spaces-remote.prop
```

MOZARTSPACES (LIFO)

```
org.xvsm.benchmarks.BenchmarkConcurrent
        -datafile mozartspaces-lifo-1fold.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -coordinatortype LIFO
```

```
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 20
        -iterations 1
        -operationnumber 1000
        -operationtype TAKE
        -monitorreset false
        -spacespropertyfile spaces-remote.prop
```

```
org.xvsm.benchmarks.BenchmarkConcurrent
        -datafile mozartspaces-lifo-2fold.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -coordinatortype LIFO
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 40
        -iterations 1
        -operationnumber 1000
        -operationtype TAKE
        -monitorreset false
        -spacespropertyfile spaces-remote.prop
```

```
org.xvsm.benchmarks.BenchmarkConcurrent
        -datafile mozartspaces-lifo-3fold.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -coordinatortype LIFO
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 60
        -iterations 1
        -operationnumber 1000
        -operationtype TAKE
        -monitorreset false
        -spacespropertyfile spaces-remote.prop
```

MOZARTSPACES (Key)

```
org.xvsm.benchmarks.BenchmarkConcurrent
        -datafile mozartspaces-key-1fold.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -coordinatortype KEY
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 20
        -iterations 1
        -operationnumber 1000
        -operationtype TAKE
        -monitorreset false
        -spacespropertyfile spaces-remote.prop
```

```
org.xvsm.benchmarks.BenchmarkConcurrent
        -datafile mozartspaces-key-2fold.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -coordinatortype KEY
        -transactiontype IMPLICIT
        -atomicentrytype STRING
```

```
        -atomicentrysize 10
        -threads 40
        -iterations 1
        -operationnumber 1000
        -operationtype TAKE
        -monitorreset false
        -spacespropertyfile spaces-remote.prop
```

```
org.xvsm.benchmarks.BenchmarkConcurrent
        -datafile mozartspaces-key-3fold.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -coordinatortype KEY
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 60
        -iterations 1
        -operationnumber 1000
        -operationtype TAKE
        -monitorreset false
        -spacespropertyfile spaces-remote.prop
```

MOZARTSPACES (Vector)

```
org.xvsm.benchmarks.BenchmarkConcurrent
        -datafile mozartspaces-vector-1fold.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -coordinatortype VECTOR
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 20
        -iterations 1
        -operationnumber 1000
        -operationtype TAKE
        -monitorreset false
        -spacespropertyfile spaces-remote.prop
```

```
org.xvsm.benchmarks.BenchmarkConcurrent
        -datafile mozartspaces-vector-2fold.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -coordinatortype VECTOR
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 40
        -iterations 1
        -operationnumber 1000
        -operationtype TAKE
        -monitorreset false
        -spacespropertyfile spaces-remote.prop
```

```
org.xvsm.benchmarks.BenchmarkConcurrent
        -datafile mozartspaces-vector-3fold.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -coordinatortype VECTOR
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 60
```

```
        -iterations 1
        -operationnumber 1000
        -operationtype TAKE
        -monitorreset false
        -spacespropertyfile spaces-remote.prop
```

MOZARTSPACES (Linda)

```
org.xvsm.benchmarks.BenchmarkConcurrent
        -datafile mozartspaces-linda-1fold.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -coordinatortype LINDA
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 20
        -iterations 1
        -operationnumber 1000
        -operationtype TAKE
        -monitorreset false
        -spacespropertyfile spaces-remote.prop
```

```
org.xvsm.benchmarks.BenchmarkConcurrent
        -datafile mozartspaces-linda-2fold.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -coordinatortype LINDA
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 40
        -iterations 1
        -operationnumber 1000
        -operationtype TAKE
        -monitorreset false
        -spacespropertyfile spaces-remote.prop
```

```
org.xvsm.benchmarks.BenchmarkConcurrent
        -datafile mozartspaces-linda-3fold.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -coordinatortype LINDA
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 60
        -iterations 1
        -operationnumber 1000
        -operationtype TAKE
        -monitorreset false
        -spacespropertyfile spaces-remote.prop
```

GIGASPACES XAP (take, transient)

```
net.jini.space.benchmarks.BenchmarkConcurrent
        -datafile gigaspaces-take-transient-1fold.dat
        -lookupserviceurl jini://192.168.123.70:4162
        -spacetype REMOTE
        -spacename mySpaceTransient
        -transactiontype NONE
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 20
        -iterations 1
```

```
        -operationnumber 1000
        -operationtype TAKE
        -monitorreset false
```

```
net.jini.space.benchmarks.BenchmarkConcurrent
        -datafile gigaspaces-take-transient-2fold.dat
        -lookupserviceurl jini://192.168.123.70:4162
        -spacetype REMOTE
        -spacename mySpaceTransient
        -transactiontype NONE
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 40
        -iterations 1
        -operationnumber 1000
        -operationtype TAKE
        -monitorreset false
```

```
net.jini.space.benchmarks.BenchmarkConcurrent
        -datafile gigaspaces-take-transient-3fold.dat
        -lookupserviceurl jini://192.168.123.70:4162
        -spacetype REMOTE
        -spacename mySpaceTransient
        -transactiontype NONE
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 60
        -iterations 1
        -operationnumber 1000
        -operationtype TAKE
        -monitorreset false
```

GIGASPACES XAP (take, persistent)

```
net.jini.space.benchmarks.BenchmarkConcurrent
        -datafile gigaspaces-take-persistent-1fold.dat
        -lookupserviceurl jini://192.168.123.70:4162
        -spacetype REMOTE
        -spacename mySpacePersistent
        -transactiontype NONE
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 20
        -iterations 1
        -operationnumber 1000
        -operationtype TAKE
        -monitorreset false
```

```
net.jini.space.benchmarks.BenchmarkConcurrent
        -datafile gigaspaces-take-persistent-2fold.dat
        -lookupserviceurl jini://192.168.123.70:4162
        -spacetype REMOTE
        -spacename mySpacePersistent
        -transactiontype NONE
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 40
        -iterations 1
        -operationnumber 1000
        -operationtype TAKE
        -monitorreset false
```

```
net.jini.space.benchmarks.BenchmarkConcurrent
        -datafile gigaspaces-take-persistent-3fold.dat
```

```
        -lookupserviceurl jini://192.168.123.70:4162
        -spacetype REMOTE
        -spacename mySpacePersistent
        -transactiontype NONE
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 60
        -iterations 1
        -operationnumber 1000
        -operationtype TAKE
        -monitorreset false
```

GIGASPACES XAP (takeIfExists, transient)

```
net.jini.space.benchmarks.BenchmarkConcurrent
        -datafile gigaspaces-takeifexists-transient-1fold.dat
        -lookupserviceurl jini://192.168.123.70:4162
        -spacetype REMOTE
        -spacename mySpaceTransient
        -transactiontype NONE
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 20
        -iterations 1
        -operationnumber 1000
        -operationtype TAKE-IF-EXISTS
        -monitorreset false
```

```
net.jini.space.benchmarks.BenchmarkConcurrent
        -datafile gigaspaces-takeifexists-transient-2fold.dat
        -lookupserviceurl jini://192.168.123.70:4162
        -spacetype REMOTE
        -spacename mySpaceTransient
        -transactiontype NONE
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 40
        -iterations 1
        -operationnumber 1000
        -operationtype TAKE-IF-EXISTS
        -monitorreset false
```

```
net.jini.space.benchmarks.BenchmarkConcurrent
        -datafile gigaspaces-takeifexists-transient-3fold.dat
        -lookupserviceurl jini://192.168.123.70:4162
        -spacetype REMOTE
        -spacename mySpaceTransient
        -transactiontype NONE
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 60
        -iterations 1
        -operationnumber 1000
        -operationtype TAKE-IF-EXISTS
        -monitorreset false
```

GIGASPACES XAP (takeIfExists, persistent)

```
net.jini.space.benchmarks.BenchmarkConcurrent
        -datafile gigaspaces-takeifexists-transient-1fold.dat
        -lookupserviceurl jini://192.168.123.70:4162
        -spacetype REMOTE
        -spacename mySpacePersistent
        -transactiontype NONE
        -atomicentrytype STRING
```

```
        -atomicentrysize 10
        -threads 20
        -iterations 1
        -operationnumber 1000
        -operationtype TAKE-IF-EXISTS
        -monitorreset false
```

```
net.jini.space.benchmarks.BenchmarkConcurrent
        -datafile gigaspaces-takeifexists-transient-2fold.dat
        -lookupserviceurl jini://192.168.123.70:4162
        -spacetype REMOTE
        -spacename mySpacePersistent
        -transactiontype NONE
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 40
        -iterations 1
        -operationnumber 1000
        -operationtype TAKE-IF-EXISTS
        -monitorreset false
```

```
net.jini.space.benchmarks.BenchmarkConcurrent
        -datafile gigaspaces-takeifexists-transient-3fold.dat
        -lookupserviceurl jini://192.168.123.70:4162
        -spacetype REMOTE
        -spacename mySpacePersistent
        -transactiontype NONE
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 60
        -iterations 1
        -operationnumber 1000
        -operationtype TAKE-IF-EXISTS
        -monitorreset false
```

8.1.2.5 Destroy

MOZARTSPACES (Random)

```
org.xvsm.benchmarks.BenchmarkConcurrent
        -datafile mozartspaces-random-1fold.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -coordinatortype RANDOM
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 20
        -iterations 1
        -operationnumber 1000
        -operationtype DESTROY
        -monitorreset false
        -spacespropertyfile spaces-remote.prop
```

```
org.xvsm.benchmarks.BenchmarkConcurrent
        -datafile mozartspaces-random-2fold.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -coordinatortype RANDOM
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 40
        -iterations 1
```

```
        -operationnumber 1000
        -operationtype DESTROY
        -monitorreset false
        -spacespropertyfile spaces-remote.prop
```

```
org.xvsm.benchmarks.BenchmarkConcurrent
        -datafile mozartspaces-random-3fold.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -coordinatortype RANDOM
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 60
        -iterations 1
        -operationnumber 1000
        -operationtype DESTROY
        -monitorreset false
        -spacespropertyfile spaces-remote.prop
```

MOZARTSPACES (FIFO)

```
org.xvsm.benchmarks.BenchmarkConcurrent
        -datafile mozartspaces-fifo-1fold.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -coordinatortype FIFO
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 20
        -iterations 1
        -operationnumber 1000
        -operationtype DESTROY
        -monitorreset false
        -spacespropertyfile spaces-remote.prop
```

```
org.xvsm.benchmarks.BenchmarkConcurrent
        -datafile mozartspaces-fifo-2fold.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -coordinatortype FIFO
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 40
        -iterations 1
        -operationnumber 1000
        -operationtype DESTROY
        -monitorreset false
        -spacespropertyfile spaces-remote.prop
```

```
org.xvsm.benchmarks.BenchmarkConcurrent
        -datafile mozartspaces-fifo-3fold.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -coordinatortype FIFO
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 60
        -iterations 1
        -operationnumber 1000
        -operationtype DESTROY
```

```
        -monitorreset false
        -spacespropertyfile spaces-remote.prop
```

MOZARTSPACES (LIFO)

```
org.xvsm.benchmarks.BenchmarkConcurrent
        -datafile mozartspaces-lifo-1fold.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -coordinatortype LIFO
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 20
        -iterations 1
        -operationnumber 1000
        -operationtype DESTROY
        -monitorreset false
        -spacespropertyfile spaces-remote.prop
```

```
org.xvsm.benchmarks.BenchmarkConcurrent
        -datafile mozartspaces-lifo-2fold.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -coordinatortype LIFO
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 40
        -iterations 1
        -operationnumber 1000
        -operationtype DESTROY
        -monitorreset false
        -spacespropertyfile spaces-remote.prop
```

```
org.xvsm.benchmarks.BenchmarkConcurrent
        -datafile mozartspaces-lifo-3fold.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -coordinatortype LIFO
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 60
        -iterations 1
        -operationnumber 1000
        -operationtype DESTROY
        -monitorreset false
        -spacespropertyfile spaces-remote.prop
```

MOZARTSPACES (Key)

```
org.xvsm.benchmarks.BenchmarkConcurrent
        -datafile mozartspaces-key-1fold.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -coordinatortype KEY
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 20
        -iterations 1
        -operationnumber 1000
        -operationtype DESTROY
        -monitorreset false
```

```
        -spacespropertyfile spaces-remote.prop
```

```
org.xvsm.benchmarks.BenchmarkConcurrent
        -datafile mozartspaces-key-2fold.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -coordinatortype KEY
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 40
        -iterations 1
        -operationnumber 1000
        -operationtype DESTROY
        -monitorreset false
        -spacespropertyfile spaces-remote.prop
```

```
org.xvsm.benchmarks.BenchmarkConcurrent
        -datafile mozartspaces-key-3fold.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -coordinatortype KEY
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 60
        -iterations 1
        -operationnumber 1000
        -operationtype DESTROY
        -monitorreset false
        -spacespropertyfile spaces-remote.prop
```

MOZARTSPACES (Vector)

```
org.xvsm.benchmarks.BenchmarkConcurrent
        -datafile mozartspaces-vector-1fold.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -coordinatortype VECTOR
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 20
        -iterations 1
        -operationnumber 1000
        -operationtype DESTROY
        -monitorreset false
        -spacespropertyfile spaces-remote.prop
```

```
org.xvsm.benchmarks.BenchmarkConcurrent
        -datafile mozartspaces-vector-2fold.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -coordinatortype VECTOR
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 40
        -iterations 1
        -operationnumber 1000
        -operationtype DESTROY
        -monitorreset false
        -spacespropertyfile spaces-remote.prop
```

```
org.xvsm.benchmarks.BenchmarkConcurrent
        -datafile mozartspaces-vector-3fold.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -coordinatortype VECTOR
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 60
        -iterations 1
        -operationnumber 1000
        -operationtype DESTROY
        -monitorreset false
        -spacespropertyfile spaces-remote.prop
```

MOZARTSPACES (Linda)

```
org.xvsm.benchmarks.BenchmarkConcurrent
        -datafile mozartspaces-linda-1fold.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -coordinatortype LINDA
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 20
        -iterations 1
        -operationnumber 1000
        -operationtype DESTROY
        -monitorreset false
        -spacespropertyfile spaces-remote.prop
```

```
org.xvsm.benchmarks.BenchmarkConcurrent
        -datafile mozartspaces-linda-2fold.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -coordinatortype LINDA
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 40
        -iterations 1
        -operationnumber 1000
        -operationtype DESTROY
        -monitorreset false
        -spacespropertyfile spaces-remote.prop
```

```
org.xvsm.benchmarks.BenchmarkConcurrent
        -datafile mozartspaces-linda-3fold.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -coordinatortype LINDA
        -transactiontype IMPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 60
        -iterations 1
        -operationnumber 1000
        -operationtype DESTROY
        -monitorreset false
        -spacespropertyfile spaces-remote.prop
```

JBOSS AS

```
javax.enterprise.benchmarks.BenchmarkConcurrent
```

```
        -datafile jboss-1fold.dat
        -namingproviderurl jnp://192.168.123.70:1099
        -transactiontype NONE
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 20
        -iterations 1
        -operationnumber 1000
        -operationtype REMOVE
        -monitorreset false
```

```
javax.enterprise.benchmarks.BenchmarkConcurrent
        -datafile jboss-2fold.dat
        -namingproviderurl jnp://192.168.123.70:1099
        -transactiontype NONE
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 40
        -iterations 1
        -operationnumber 1000
        -operationtype REMOVE
        -monitorreset false
```

```
javax.enterprise.benchmarks.BenchmarkConcurrent
        -datafile jboss-3fold.dat
        -namingproviderurl jnp://192.168.123.70:1099
        -transactiontype NONE
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 60
        -iterations 1
        -operationnumber 1000
        -operationtype REMOVE
        -monitorreset false
```

8.1.3 Block Benchmarks

8.1.3.1 Write + Read + Take/Destroy

MOZARTSPACES (Random)

```
org.xvsm.benchmarks.BenchmarkBlock
        -datafile mozartspaces-random-1fold.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -coordinatortype RANDOM
        -transactiontype EXPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 20
        -iterations 10
        -operationnumbers 100,100,50
        -operationtypes WRITE,READ,TAKE
        -monitorreset false
        -spacespropertyfile spaces-remote.prop
```

```
org.xvsm.benchmarks.BenchmarkBlock
        -datafile mozartspaces-random-2fold.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -coordinatortype RANDOM
        -transactiontype EXPLICIT
        -atomicentrytype STRING
```

```
        -atomicentrysize 10
        -threads 40
        -iterations 10
        -operationnumbers 100,100,50
        -operationtypes WRITE,READ,TAKE
        -monitorreset false
        -spacespropertyfile spaces-remote.prop
```

```
org.xvsm.benchmarks.BenchmarkBlock
        -datafile mozartspaces-random-3fold.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -coordinatortype RANDOM
        -transactiontype EXPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 60
        -iterations 10
        -operationnumbers 100,100,50
        -operationtypes WRITE,READ,TAKE
        -monitorreset false
        -spacespropertyfile spaces-remote.prop
```

MOZARTSPACES (FIFO)

```
org.xvsm.benchmarks.BenchmarkBlock
        -datafile mozartspaces-fifo-1fold.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -coordinatortype FIFO
        -transactiontype EXPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 20
        -iterations 10
        -operationnumbers 100,100,50
        -operationtypes WRITE,READ,TAKE
        -monitorreset false
        -spacespropertyfile spaces-remote.prop
```

```
org.xvsm.benchmarks.BenchmarkBlock
        -datafile mozartspaces-fifo-2fold.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -coordinatortype FIFO
        -transactiontype EXPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 40
        -iterations 10
        -operationnumbers 100,100,50
        -operationtypes WRITE,READ,TAKE
        -monitorreset false
        -spacespropertyfile spaces-remote.prop
```

```
org.xvsm.benchmarks.BenchmarkBlock
        -datafile mozartspaces-fifo-3fold.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -coordinatortype FIFO
        -transactiontype EXPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 60
```

```
        -iterations 10
        -operationnumbers 100,100,50
        -operationtypes WRITE,READ,TAKE
        -monitorreset false
        -spacespropertyfile spaces-remote.prop
```

MOZARTSPACES (LIFO)

```
org.xvsm.benchmarks.BenchmarkBlock
        -datafile mozartspaces-lifo-1fold.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -coordinatortype LIFO
        -transactiontype EXPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 20
        -iterations 10
        -operationnumbers 100,100,50
        -operationtypes WRITE,READ,TAKE
        -monitorreset false
        -spacespropertyfile spaces-remote.prop
```

```
org.xvsm.benchmarks.BenchmarkBlock
        -datafile mozartspaces-lifo-2fold.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -coordinatortype LIFO
        -transactiontype EXPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 40
        -iterations 10
        -operationnumbers 100,100,50
        -operationtypes WRITE,READ,TAKE
        -monitorreset false
        -spacespropertyfile spaces-remote.prop
```

```
org.xvsm.benchmarks.BenchmarkBlock
        -datafile mozartspaces-lifo-3fold.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -coordinatortype LIFO
        -transactiontype EXPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 60
        -iterations 10
        -operationnumbers 100,100,50
        -operationtypes WRITE,READ,TAKE
        -monitorreset false
        -spacespropertyfile spaces-remote.prop
```

MOZARTSPACES (Key)

```
org.xvsm.benchmarks.BenchmarkBlock
        -datafile mozartspaces-key-1fold.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -coordinatortype KEY
        -transactiontype EXPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 20
        -iterations 10
```

```
        -operationnumbers 100,100,50
        -operationtypes WRITE,READ,TAKE
        -monitorreset false
        -spacespropertyfile spaces-remote.prop
```

```
org.xvsm.benchmarks.BenchmarkBlock
        -datafile mozartspaces-key-2fold.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -coordinatortype KEY
        -transactiontype EXPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 40
        -iterations 10
        -operationnumbers 100,100,50
        -operationtypes WRITE,READ,TAKE
        -monitorreset false
        -spacespropertyfile spaces-remote.prop
```

```
org.xvsm.benchmarks.BenchmarkBlock
        -datafile mozartspaces-key-3fold.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -coordinatortype KEY
        -transactiontype EXPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 60
        -iterations 10
        -operationnumbers 100,100,50
        -operationtypes WRITE,READ,TAKE
        -monitorreset false
        -spacespropertyfile spaces-remote.prop
```

MOZARTSPACES (Vector)

```
org.xvsm.benchmarks.BenchmarkBlock
        -datafile mozartspaces-vector-1fold.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -coordinatortype VECTOR
        -transactiontype EXPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 20
        -iterations 10
        -operationnumbers 100,100,50
        -operationtypes WRITE,READ,TAKE
        -monitorreset false
        -spacespropertyfile spaces-remote.prop
```

```
org.xvsm.benchmarks.BenchmarkBlock
        -datafile mozartspaces-vector-2fold.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -coordinatortype VECTOR
        -transactiontype EXPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 40
        -iterations 10
        -operationnumbers 100,100,50
        -operationtypes WRITE,READ,TAKE
```

```
        -monitorreset false
        -spacespropertyfile spaces-remote.prop
```

```
org.xvsm.benchmarks.BenchmarkBlock
        -datafile mozartspaces-vector-3fold.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -coordinatortype VECTOR
        -transactiontype EXPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 60
        -iterations 10
        -operationnumbers 100,100,50
        -operationtypes WRITE,READ,TAKE
        -monitorreset false
        -spacespropertyfile spaces-remote.prop
```

MOZARTSPACES (Linda)

```
org.xvsm.benchmarks.BenchmarkBlock
        -datafile mozartspaces-linda-1fold.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -coordinatortype LINDA
        -transactiontype EXPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 20
        -iterations 10
        -operationnumbers 100,100,50
        -operationtypes WRITE,READ,TAKE
        -monitorreset false
        -spacespropertyfile spaces-remote.prop
```

```
org.xvsm.benchmarks.BenchmarkBlock
        -datafile mozartspaces-linda-2fold.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -coordinatortype LINDA
        -transactiontype EXPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 40
        -iterations 10
        -operationnumbers 100,100,50
        -operationtypes WRITE,READ,TAKE
        -monitorreset false
        -spacespropertyfile spaces-remote.prop
```

```
org.xvsm.benchmarks.BenchmarkBlock
        -datafile mozartspaces-linda-3fold.dat
        -containertype REMOTE
        -containeruri tcpjava://192.168.123.70:4321
        -coordinatortype LINDA
        -transactiontype EXPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 60
        -iterations 10
        -operationnumbers 100,100,50
        -operationtypes WRITE,READ,TAKE
        -monitorreset false
        -spacespropertyfile spaces-remote.prop
```

GIGASPACES XAP (transient)

```
net.jini.space.benchmarks.BenchmarkBlock
        -datafile gigaspaces-transient-1fold.dat
        -lookupserviceurl jini://192.168.123.70:4162
        -spacetype REMOTE
        -spacename mySpaceTransient
        -transactiontype EXPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 20
        -iterations 10
        -operationnumbers 100,100,50
        -operationtypes WRITE,READ,TAKE
        -monitorreset false
```

```
net.jini.space.benchmarks.BenchmarkBlock
        -datafile gigaspaces-transient-2fold.dat
        -lookupserviceurl jini://192.168.123.70:4162
        -spacetype REMOTE
        -spacename mySpaceTransient
        -transactiontype EXPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 40
        -iterations 10
        -operationnumbers 100,100,50
        -operationtypes WRITE,READ,TAKE
        -monitorreset false
```

```
net.jini.space.benchmarks.BenchmarkBlock
        -datafile gigaspaces-transient-3fold.dat
        -lookupserviceurl jini://192.168.123.70:4162
        -spacetype REMOTE
        -spacename mySpaceTransient
        -transactiontype EXPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 60
        -iterations 10
        -operationnumbers 100,100,50
        -operationtypes WRITE,READ,TAKE
        -monitorreset false
```

GIGASPACES XAP (persistent)

```
net.jini.space.benchmarks.BenchmarkBlock
        -datafile gigaspaces-persistent-1fold.dat
        -lookupserviceurl jini://192.168.123.70:4162
        -spacetype REMOTE
        -spacename mySpacePersistent
        -transactiontype EXPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 20
        -iterations 10
        -operationnumbers 100,100,50
        -operationtypes WRITE,READ,TAKE
        -monitorreset false
```

```
net.jini.space.benchmarks.BenchmarkBlock
        -datafile gigaspaces-persistent-2fold.dat
        -lookupserviceurl jini://192.168.123.70:4162
        -spacetype REMOTE
```

```
        -spacename mySpacePersistent
        -transactiontype EXPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 40
        -iterations 10
        -operationnumbers 100,100,50
        -operationtypes WRITE,READ,TAKE
        -monitorreset false
```

```
net.jini.space.benchmarks.BenchmarkBlock
        -datafile gigaspaces-persistent-3fold.dat
        -lookupserviceurl jini://192.168.123.70:4162
        -spacetype REMOTE
        -spacename mySpacePersistent
        -transactiontype EXPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 60
        -iterations 10
        -operationnumbers 100,100,50
        -operationtypes WRITE,READ,TAKE
        -monitorreset false
```

JBOSS AS

```
javax.enterprise.benchmarks.BenchmarkBlock
        -datafile jboss-1fold.dat
        -namingproviderurl jnp://192.168.123.70:1099
        -transactiontype EXPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 20
        -iterations 10
        -operationnumbers 100,100,50
        -operationtypes CREATE,FIND,REMOVE
        -monitorreset false
```

```
javax.enterprise.benchmarks.BenchmarkBlock
        -datafile jboss-2fold.dat
        -namingproviderurl jnp://192.168.123.70:1099
        -transactiontype EXPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 40
        -iterations 10
        -operationnumbers 100,100,50
        -operationtypes CREATE,FIND,REMOVE
        -monitorreset false
```

```
javax.enterprise.benchmarks.BenchmarkBlock
        -datafile jboss-3fold.dat
        -namingproviderurl jnp://192.168.123.70:1099
        -transactiontype EXPLICIT
        -atomicentrytype STRING
        -atomicentrysize 10
        -threads 60
        -iterations 10
        -operationnumbers 100,100,50
        -operationtypes CREATE,FIND,REMOVE
        -monitorreset false
```