Project no. IST-033576

XtreemOS

Integrated Project
BUILDING AND PROMOTING A LINUX-BASED OPERATING SYSTEM TO SUPPORT VIRTUAL ORGANIZATIONS FOR NEXT GENERATION GRIDS

On the feasibility of cloud computing functionality for Grid applications D3.2.15

Due date of deliverable: December 1st, 2009
Actual submission date: December 1st, 2009

Start date of project: June 1st 2006

Type: Deliverable
WP number: WP3.2
Task number: T3.2.x

Responsible institution: VUA
Editor & and editor's address: Guillaume Pierre
Vrije Universiteit Amsterdam
Dept. of Computer Science
De Boelelaan 1081a
1081HV Amsterdam
The Netherlands

Version 1.1 / Last edited by Guillaume Pierre / December 2nd, 2009

Project co-funded by the European Commission within the Sixth Framework Programme		
Dissemination Level		
PU	Public	√
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

Revision history:

Version	Date	Authors	Institution	Section affected, comments
0.0	02/11/09	Guillaume Pierre	VUA	First draft
1.0	06/11/09	Guillaume Pierre	VUA	Ready for internal review
1.1	2/12/09	Guillaume Pierre	VUA	Updated version according to internal reviews

Reviewers:
Björn Kolbek (ZIB) and Michael Schöttner (UDUS)

Tasks related to this deliverable:

Task No.	Task description	Partners involved°
T3.2.5	Cloud computing services	VUA*

°This task list may not be equivalent to the list of partners contributing as authors to the deliverable
*Task leader

Executive summary

Cloud computing emerged as a new paradigm for utility computing after the start of the XtreemOS project. Although such developments were impossible to predict at the start of the project, XtreemOS and Clouds are very relevant to each other. Cloud computing can be categorized into Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS) and Software-as-a-Service (SaaS). This deliverable claims that:

- XtreemOS is directly comparable to a full IaaS platform;
- XtreemOS is a good platform to further develop PaaS functionality, as demonstrated by our port of the HBase database.
- SaaS is irrelevant to the work on XtreemOS, and we have no plans for development there.

1 Introduction

Cloud computing is an emerging paradigm for utility computing. Although this new field is notoriously difficult to define [4] and most announcements in the domain are dictated by hype rather than actual content [11], it nevertheless represents a fundamental and interesting paradigm shift from more classical forms of utility computing such as Grid computing.

Both paradigms rely on the concept of statistical multiplexing, which exploits the fact that the resource usage of any single user varies over time and that peaks of activities are usually not correlated from one user to the next [15]. Rather than provisioning resources for each user individually based on the expected peak resource usage, statistical multiplexing proposes to share resources between users, which allows to provision resources based on the average utilization. Although statistical multiplexing was originally invented for sharing network bandwidth resources, Grids and Clouds rely on the same principles regarding the sharing of computing and storage resources.

The two paradigms also have a number of important differences. First, Grid computing relies in principle on resource *sharing* that does not necessarily imply a monetary accounting of resources used by each user. Conversely, in Cloud computing resources are usually under the control of a single operator, and users get charged under a pay-as-you-go pricing scheme.

Second, the types of services offered by Grid and Cloud computing are somewhat different. Grid computing originates from the field of high-performance computing, and hence tends to focus on parallel programming (e.g., MPI) and absolute performance figures. Cloud computing originates from enterprise hosting platforms, and focuses on cost efficiency and scalability.

Cloud computing functionality is often categorized as follows:

Infrastructure-as-a-Service (IaaS) is the delivery of computer infrastructure (typically a platform virtualization environment) as a service [12]. Typical IaaS services are Amazon.com's EC2 (which offers generic virtual machines for hosting applications in the Cloud) and S3 (which offers scalable storage of unstructured files). Other Cloud vendors have similar (mostly API-*in*compatible) offers as well.

Platform-as-a-Service (PaaS) is the delivery of a computing platform and solution stack as a service [13]. Typical PaaS services are Google's Map/Reduce (which automates the parallelization of massive data processing applications), BigTable (which provides scalable structured data storage) and AppEngine (which automates the scalable hosting of Web applications). Here as well, many competing offers exist, but there is no consensus on standard APIs that could facilitate the migration of applications from one Cloud platform to another.

Software-as-a-Service (SaaS) is a model of software deployment whereby a provider licenses an application to customers for use as a service on demand [14]. Typical SaaS services include Google's Gmail or Google maps. SaaS is irrelevant to the work on an operating system for the Grid, as SaaS focuses mostly on providing Web interfaces to end-user office applications such as email, spreadsheets and text editors. We therefore do not plan any development in this area.

This deliverable argues that: (i) XtreemOS can be seen as a relatively mature platform for IaaS, where the Application Execution Manager (AEM) offers similar functionality to EC2 and XtreemFS offers similar functionality to S3; (ii) XtreemOS can be extended to become a good platform for PaaS, as illustrated by our port of the HBase open-source scalable database onto XtreemOS.

This deliverable is structured as follows. Section 2 discusses in which sense XtreemOS can be seen as an IaaS platform already. Then, Section 3 shows how we ported HBase to XtreemOS to give it a first PaaS service. Finally, Section 4 concludes.

2 XtreemOS as an IaaS platform

Infrastructure as a Service (IaaS) refers to systems that provide their users with computing resources delivered as a service, including servers, network equipment, memory, CPU and disk space [12]. Although the term was coined after the start of the XtreemOS project, this is precisely the goal that XtreemOS aims to achieve: provide users with computing resources that can be assigned to them dynamically when they need them. In particular, AEM and XtreemFS can legitimately be classified as IaaS components.

AEM allows users to reserve machines through an XtreemOS Virtual Organization (VO) when they need to execute a job. In this sense, it is directly comparable to Amazon's EC2 and other similar services. The two types of services differ in four main aspects:

1. Different computation-as-a-service offers rely on different APIs. No clear consensus emerges at the moment regarding a standard access API for such services. On the other hand, AEM is meant to be invoked via the XOSAGA API, which relies on the standard SAGA API for Grid Applications. One could relatively easily build an EC2-compatible API to the AEM but this is not in our immediate priorities at the moment.

2. IaaS services typically rely on virtualization techiques where the resources are offered to users in the form of a full virtualized operating system instance. On the other hand, the AEM executes jobs directly, with no virtualization. This difference arises from different requirements in different services. Cloud platforms must be usable by a very large range of users who may each want to use a different operating system and work in total isolation from each other. On the other hand, XtreemOS intends to be the standard operating system to develop Grid applications, so there is no need to virtualize XtreemOS on top of XtreemOS. XtreemOS also provides strong isolation between multiple jobs running on the same hardware through the use of Linux containers [6].

3. IaaS platforms use a pay-as-you-go pricing model, while XtreemOS relies on the trust relationships between system administrators of a VO to implement a shared resource available to all users of the VO.

4. The security of IaaS platforms relies on a one-to-one trust relationship between the cloud provider and the cloud customer. On the other hand, the VO support in XtreemOS allows one to support several potentially mutually distrusful Cloud providers, allowing Cloud customers to select (and possibly revoke) the provider of their choice.

XtreemFS allows Grid users to store data efficiently and share them across the whole system. In this sense, it is directly comparable to Amazon's S3 and other similar services. The two types of services differ in two aspects:

1. Again, no standard API for Cloud storage seems to emerge yet. Most Cloud storage services provide simplistic functionality, where a user can write blocks of files that can be later read but remain immutable. Conversely, XtreemFS implements the full Posix API where files can be updated and overwritten. One could relatively easily build an S3-compatible API to XtreemFS but this is not in our immediate priorities at the moment.

2. Performance-wise, it is hard to compare XtreemFS with S3 as the two run on different hardware. As a first indication, we report preliminary performance comparisons between XtreemFS and HDFS (an open-source clone of the Google file system) in Section 3.4.

In conclusion, XtreemOS provides the two main elements necessary for an IaaS platform. Although a number of differences exist with their pure Cloud-computing counterparts, both the AEM and XtreemFS provide similar functionality and could potentially develop API-compatible interfaces to facilitate migration of Cloud users onto XtreemOS.

3 XtreemOS as a PaaS platform

Although XtreemOS was not originally designed for Cloud computing applications, it does provide a good base platform for developing advanced Cloud computing functionality. We selected the specific topic of scalable database support to demonstrate how one can deploy Cloud functionality on XtreemOS.

Relational databases (RDBMS) such as MySQL and Oracle have been popular for decades thanks to their conceptual simplicity, the great expressive power of the SQL query language, and the performance improvements that have been brought by decades of development. However, their great expressive power also makes it very difficult to *scale* them up by using large numbers of computers instead of a single powerful database server. Because users are assumed to be likely to query any set of data items from the database through a single query, distributed RDBMSs often rely on *full replication* to distribute their computation across multiple machines. Read queries can thus be addressed to any replica, and scale very well. On the other hand, update queries must in one way or another be propagated to all replicas. This means that, when using N database replicas, each replica must process $\frac{1}{N} \times ReadQueries + WriteQueries$. When the number of write queries alone grows beyond the capacity of a single replica server, no additional improvement can be brought by adding extra replicas. The only solution to scale an application further is to use data partitioning [9]. Partitioning data manually is a difficult process, so developers prefer to rely on automatic data partitioning.

A new family of scalable database systems is being developed for Cloud computing environments, exemplified by Amazon.com's SimpleDB [1], Google's Bigtable [2], Yahoo's PNUTS [3] and Facebook's Cassandra [7]. Although all these systems are slightly different from each other, they all rely on the same underlying principles. These systems scale nearly linearly with the number of servers they are using, thanks to the systematic use of automatic data partitioning. On the other hand, they do not support the SQL language and rather provide a simpler query language. Data are organized in tables, which can be queried by primary key only. Similarly, these systems do not support join operations. As restrictive as such limitations may look, they do allow to build useful applications. Scalable database systems typically provide weak consistency guarantees such as eventual consistency [8] or single-record transactions, but one can apply stronger consistency as an added layer on top [10].

To demonstrate how XtreemOS can be a great platform for PaaS Cloud computing, we ported the HBase system [5] (an open-source clone of Bigtable) to XtreemOS. This provides XtreemOS with a scalable database service that can be used by Grid applications to store and query their structured data.

3.1 Design

3.1.1 HBase requirements

HBase is a Java program designed to run on many cheap Linux servers. In a default HBase installation, each server requires the following:

Java 1.6.x preferably the Sun JVM although HBase should in theory work with other compatible JVM's.

Hadoop 0.19.x required in a default installation for providing the *Hadoop Distributed FileSystem* to HBase servers.

OpenSSH HBase needs the ssh command-line client to remotely start and stop the HBase master and region servers.

NTP the system time on each node in an HBase cluster is assumed to be in basic alignment. A *Network Time Protocol* client may be used to synchronize time on the nodes.

Configuration at minimum each HBase region server needs to know the location of the HBase master server, and the path to the shared filesystem between all HBase servers. This can be configured using the *hbase.master* and *hbase.root* configuration keys respectively. Additionally, in a default HBase setup the location of all region and master servers are stored in a textfile to automate SSH commands for starting and stopping HBase.

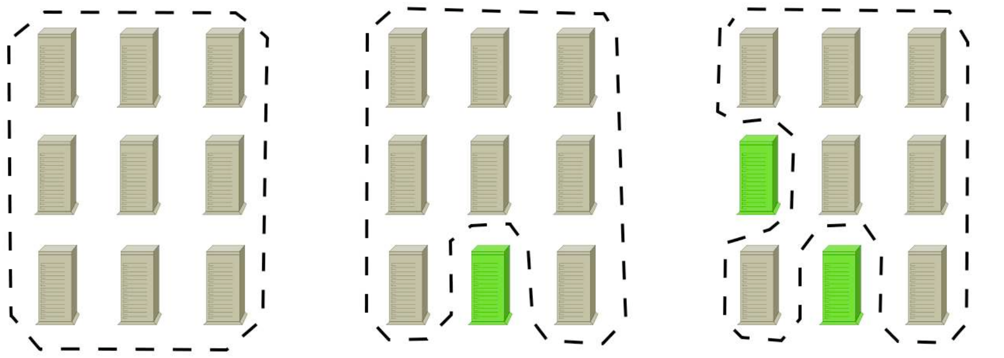
3.1.2 Shared filesystem

In order to successfully run a HBase installation, each HBase "region server" needs to have access to a filesystem which is shared among all HBase region servers. HBase uses various built-in filesystem modules provided by Hadoop which can be configured as the shared filesystem using the *hbase.root* configuration key, including the default HDFS, FTP, Amazon's S3 and the local filesystem. Each supported filesystem can be configured using the corresponding URL in the *hbase.root* configuration key, for example to use HDFS one would configure HBase as illustrated in figure 1.

To port HBase to XtreemOS, we should replace the default HDFS with XtreemFS to provide a shared storage among HBase region nodes, as XtreemFS is already available as shared storage among all XtreemOS resource nodes. The XtreemFS client is built on top of the FUSE kernel module. This allows applications, including HBase running in the OpenJDK Java Virtual Machine, to access XtreemFS with regular POSIX functions on the local filesystem. Fortunately, HBase supports accessing the local filesystem using the *file://* URL in the *hbase.rootdir* configuration key, which we have used to point to the user's XtreemFS home direc-

```
<property>
  <name>hbase.rootdir</name>
  <value>hdfs://my.host.com:9000/hbase</value>
  <description>The directory shared by region servers.
    Should be fully-qualified to include the filesystem to use.
    E.g: hdfs://NAMENODE_SERVER:PORT/HBASE_ROOTDIR
  </description>
</property>
```

Figure 1: Example HBase configuration when using HDFS as shared filesystem


(a) Initially reserve all nodes.

(b) AEM schedules the program on one node from the reservation. Now we reserve all remaining nodes.

(c) Keep reserving and submitting until enough programs running.

Figure 2: Scheduling jobs in AEM on different resource nodes using reservations

tory. This way we where able to run HBase region servers using XtreemFS as shared storage. Therefore, to replace the Hadoop Distributed FileSystem (HDFS) with XtreemFS to provide a shared storage among all HBase region nodes, we did not need to modify the HBase source code.

3.1.3 Remote execution

In a regular HBase installation, the *bin/start-hbase.sh* and *bin/stop-hbase.sh* scripts are used to start and stop HBase using remote SSH commands, respectively. The location of each HBase region is written in the configuration file *conf/regionserver*, which is read by these scripts in order to connect to the right SSH daemon.

For running the HBase region servers on XtreemOS it is important that the AEM schedules them on different resource nodes, as otherwise there would be no performance gains if multiple HBase region servers are launched multiple times on the same machine(s), or worse it could overload them. At the time we ported HBase, the AEM supported multiple-process jobs but unfortunately could not guarantee that different physical nodes would be used for each process. However, it supports reservation of resource nodes. By reserving XtreemOS resource nodes, an application can supply AEM with a number of machines of interest, and the time at which the application wants to use them. The application is then able to run a program on any of the reserved machines. It is then possible to only reserve resource nodes, which the application knows it has not yet started an instance of it's program, as illustrated in Figure 2. Our current HBase implementation uses this method to run HBase on different XtreemOS resource nodes.

With current improved support for multiple-node jobs in the AEM we should now be able to greatly simplify the deployment process.

3.1.4 Time synchronization

HBase region servers are expected to have their system time synchronized to the system time of the HBase master server. HBase developers suggest to use NTP to synchronize clocks across the system. In a grid environment such as XtreemOS it may not be guaranteed that each node has it's system time synchronized, and that all nodes reside in the same timezone. Our current implementation of HBase on XtreemOS does not deal with this problem and assumes that system times are correctly synchronized.

3.1.5 Configuration

HBase is configured using two XML files: `conf/hbase-default.xml` containing the default configuration and `conf/hbase-site` for user configuration.

Configuration for HBase region servers is expected to be in a known location, and should be available for each HBase server.

In a grid environment such as XtreemOS, no assumptions can be made on the contents of the local filesystem in each resource node, such as the location and access permissions of configuration files, as the resource nodes where HBase is scheduled on may change each time it is restarted with a new job. Therefore we use XtreemFS to store the HBase configuration for both the HBase master and region servers. An advantage is that the user can configure all HBase servers using one configuration file, instead of configuring each region server individually as in the default HBase distribution. Administrators of XtreemOS resource nodes can override the HBase user configuration when appropriate. The `conf/hbase-default.xml` file on the XtreemFS is a symbolic link to the `conf/hbase-default.xml` configuration file in the default `/usr/share/hbase` HBase installation directory to allow configuration overrides per XtreemOS resource node. The `conf/hbase-env.sh` file on the user's XtreemFS home directory, containing HBase environment variables, also reads the `conf/hbase-env.sh` if available to also allow overrides of environment variables. For example, HBase can be configured to consume a maximum amount of system memory with the `HBASE_HEAPSIZE` environment variable in the `conf/hbase-env.sh` file. Administrators of XtreemOS resource nodes can set this value to an acceptable level based on the amount of system memory available on the resource node.

3.1.6 Automating administration

To automate the process of submitting jobs to start and stop HBase on XtreemOS, we have written four python scripts which automatically edit the HBase XML configuration files, especially the `hbase.master` configuration key. This configuration key contains the location of the currently active HBase master and needs to be edited each time the HBase master is started on the XtreemOS grid, as the resource node chosen by AEM to run the HBase master may change. Figure 3 illustrates the interaction between the HBase python scripts, AEM and XtreemFS.

3.2 User Programs

3.2.1 hbase-xos

Users can start and stop HBase on an XtreemOS grid using the `hbase-xos` program. It supports the actions *start*, *stop*, *restart*, *status* and *setup*. *setup* must be used by the user to initialize HBase configuration in the user's XtreemFS home directory before starting HBase on XtreemOS. *start*, *stop* and *restart* are used to start, stop and restart HBase on XtreemOS respectively. *status* may be used to output the current status of HBase, whether it is running and optionally on which resource nodes. `hbase-xos` optionally accepts command-line arguments to print out help information, enable verbose debugging output, override the location of the HBase configuration and data, the number of HBase regions to be started or stopped and the location of the XtreemOS user certificate to authenticate to AEM.

3.2.2 hbase-xos-master

When submitting a job to AEM to start the HBase master, `hbase-xos` constructs a JSDL to start the `hbase-xos-master` script. Once started on a XtreemOS resource node, it is responsible for overwriting the `hbase.master` configuration variable with it's own IP address. After updating the HBase configuration, it invokes an HBase master server. Although possible, this script should not be called directly by users.

3.2.3 hbase-xos-region

After `hbase-xos` has submitted a job to run the HBase master, it starts submitting JSDL jobs to run the `hbase-xos-region` program. It is used to start an HBase region server and like `hbase-xos-master` it should not be directly invoked by users.

3.2.4 hbase-xos-setup

To initialize the user's HBase configuration, `hbase-xos` submits a job to run `hbase-xos-setup` on any of the XtreemOS resource nodes. It creates initial XML files and points symbolic links to the appropriate files. Per default `hbase-xos-setup` initializes HBase configuration in `.hbase` in the users XtreemFS home directory.

3.3 Deployment

Mandriva does not currently have a standard HBase RPM package. For easy installation of HBase we have created a RPM package "hbase" for the default HBase installation and an XtreemOS-specific "hbase-xos" package containing the python scripts for managing HBase on XtreemOS.

HBase-xos is documented through man pages ("man hbase-xos") and the `-help` command-line argument for a brief description of its syntax.

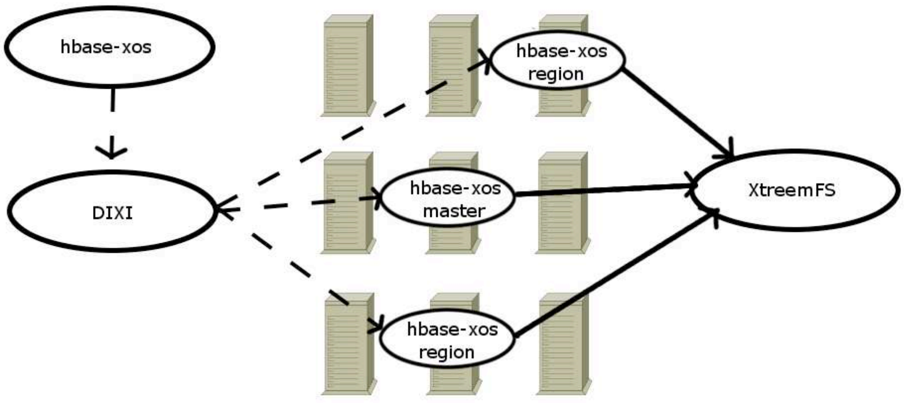


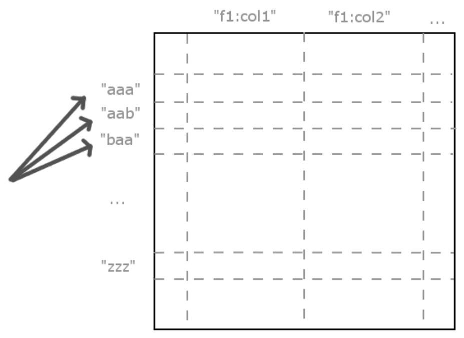
Figure 3: hbase-xos interacts with AEM to submit HBase jobs to XtreemOS.

3.4 Performance Evaluation

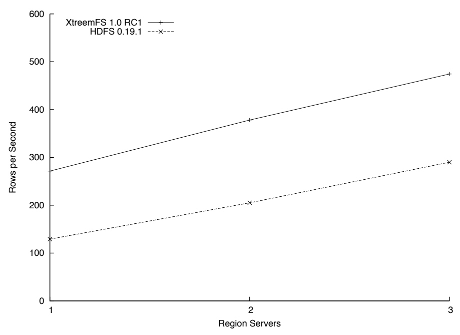
In this section we present the performance of HBase on XtreemOS we measured. For the performance tests we used three nodes from the VU testbed: *node004, node005* and *node007*. Each node had the following hardware and software installed:

- **Processor:** dual core 996.928 Mhz pentium III Coppermine
- **Memory:** 1 GB
- **Disk:** 20 gigabyte Seagate ATA disk
- **Network:** Intel Pro 100Mbit ethernet card
- **XtreemOS:** version 1.0 RC1
- **Kernel:** 2.6.20-0.5mdvsmp
- **Java:** OpenJDK 1.7.0
- **HBase:** version 0.19.2
- **XtreemFS:** version 1.0 RC1
- **Hadoop:** version 0.19.1

We benchmark our HBase installation using the `PerformanceEvaluation` program, which is included in the official HBase distribution. This program supports 5 different tests: random reads and writes, sequential reads and writes and scanning. It starts a given number of HBase clients as a local process or a MapReduce job. A similar number of HBase region servers should be started for each test, to spread the load generated by the clients among the available HBase region servers. For our tests we configured PerformanceEvaluation to read and write



(a) The Sequential Read test performs a read on each row key in order.



(b) Performance results.

Figure 4: Sequential Read test

512 MB of data, and used 64 MB regions in HBase. For comparison with the default Hadoop Distributed FileSystem (HDFS) included in HBase, we used version HDFS 0.19.1.

Each node ran an XtreemFS MRC and OSD service, or a HDFS Datanode respectively. *Node004* ran the HBase master, and *node007* the PerformanceEvaluation test, and all nodes ran an HBase region server as required by the tests. The XtreemFS and HDFS services where restarted on each test, to prevent caching to pollute the performance measurements. For the same reason all HBase master and region servers where restarted before each test was ran.

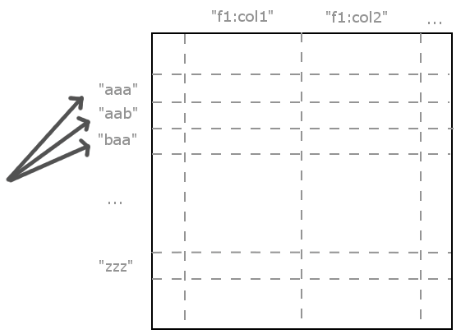
3.4.1 Sequential Read

The *Sequential Read* test performs a single read operation on each row in HBase in lexicographical order, as illustrated in Figure 4(a). Figure 4(b) displays the number of 1000-byte rows read per second. XtreemFS seems to be almost twice as fast as HDFS on Sequential Reads. Both filesystems perform very well.

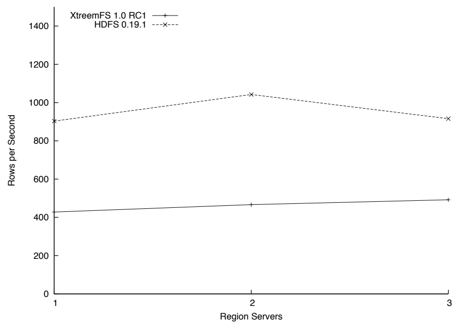
3.4.2 Sequential Write

Sequential Write performs the same test as Sequential Read, except that it writes a new value to each row instead of reading, as Figure 5(a) illustrates. Prformance results are presented in Figure 5(b). For Sequential Writes HDFS appears to be twice as fast as XtreemFS. The throughput barely increases when adding region servers with XtreemFS, while HDFS shows an increasing line. However with three region servers, throughput seems to decrease with HDFS.

We believe that the relatively poor write performance of XtreemFS is due to the fact that the tested version updates file metadata synchronously upon each

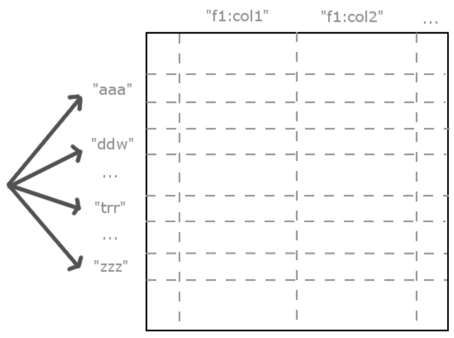


(a) The Sequential Write test performs a write on each row key in order.

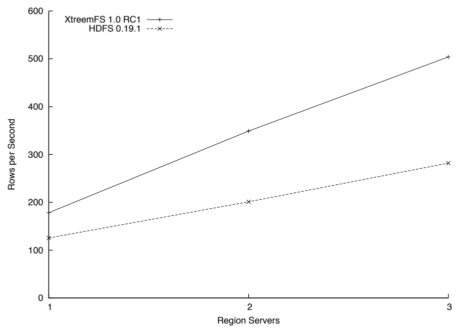


(b) Performance results.

Figure 5: Sequential Write test



(a) The Random Read test requests a read on each row key in random order.



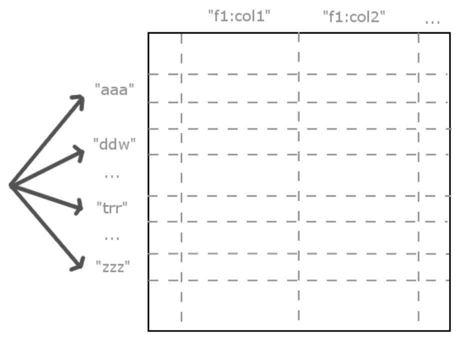
(b) Performance results.

Figure 6: Random Read test

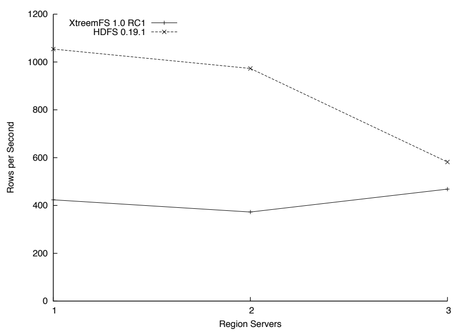
block write. The current XtreemFS release now supports asynchronous metadata updates, which should significantly improve these measurements.

3.4.3 Random Read

In practice, HBase clients may not read or write row keys in order. The *Random Read* test measures the performance of single read operations on all row keys in random order, as illustrated in Figure 6(a). Figure 6(b) displays the performance results. As in the Sequential Read test, these results show that XtreemFS is faster on Random Reads than HDFS. Both scale very well when adding region servers.



(a) The Random Write test requests a write on each row key in random order.



(b) Performance results.

Figure 7: Random Write test

3.4.4 Random Write

Like reads, HBase clients may not write to row keys in the order as they appear in the map, as illustrated in Figure 7(a). *Random Write* measures HBase performance when writing to row keys in a random order. Figure 7(b) presents the results of the Random Write test. Like in Sequential Write HDFS is much faster than XtreemFS. XtreemFS shows a slight performance increase with three region servers, but a performance drop with two region servers. The decreasing line of HDFS shows that it does not scale very well with Random Writes.

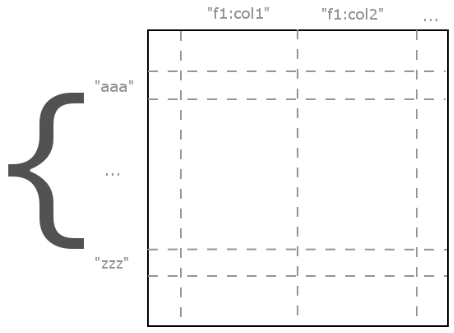
3.4.5 Scans

HBase supports scanning of row ranges. With a scan, HBase clients can read a range of rows a once, as illustrated in Figure 8(a). Scanning reduces the communication overhead between the HBase client and region servers. Performance results are shown in Figure 8(b). For a single node setup, XtreemFS is faster than HDFS. However HDFS scans twice the rate of XtreemFS with more than one region server. HDFS shows better performance increases when adding region servers.

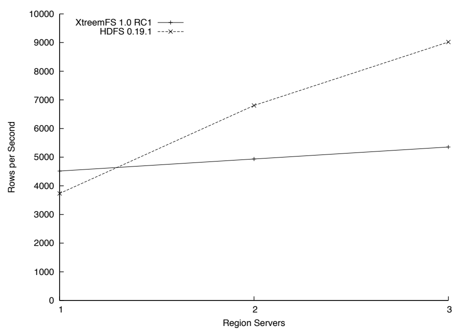
3.4.6 Discussion

Our performance evaluations show that HBase performs well on XtreemOS and allows Grid developers to write scalable data-intensive applications easily.

The evaluations also confirm the potential of XtreemFS to significantly outperform well-known state-of-the-art distributed file systems such as HDFS. Performance for read-intensive workloads is indeed much better than using HDFS.



(a) The Scan test requests a read from a range of row keys.



(b) Performance results.

Figure 8: Scan test

On the other hand, these tests allowed us to attract the attention of XtreemFS developers onto the relatively poor write performance of XtreemFS. Such effects are likely to be caused by the synchronous metadata update strategy implemented by XtreemFS at the time of our benchmarks. The current XtreemFS supports asynchronous metadata updates, which should allow significant write performance improvements. Confirming this is in our immediate agenda.

4 Conclusion

Cloud computing emerged as a new paradigm for utility computing after the start of the XtreemOS project. Although such developments were impossible to predict at the first stages of the projects, the XtreemOS technology is XtreemLY relevant to Cloud computing:

- XtreemOS is directly comparable to a full IaaS platform;
- XtreemOS is a good platform to further develop PaaS functionality, as demonstrated by our port of the HBase database.

We hope that these results contribute to making XtreemOS a major platform for utility computing in the coming years.

References

[1] Amazon.com. Amazon SimpleDB. `http://aws.amazon.com/simpledb`.

[2] Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C. Hsieh, Deborah A. Wallach, Mike Burrows, Tushar Chandra, Andrew Fikes, and Robert E. Gruber. Bigtable : a distributed storage system for structured data. In *Proceedings of The 7th USENIX Symposium on Operating Systems Design and Implementation*, pages 205–218, 2006.

[3] Brian F. Cooper, Raghu Ramakrishnan, Utkarsh Srivastava, Adam Silberstein, Philip Bohannon, Hans-Arno Jacobsen, Nick Puz, Daniel Weaver, and Ramana Yerneni. PNUTS: Yahoo!'s hosted data serving platform. In *Proceedings of the 34th International Conference on Very Large Data Bases*, pages 1277–1288, 2008.

[4] Jeremy Geelan. Twenty-one experts define cloud computing. *Cloud Computing Journal*, January 2009. `http://cloudcomputing.sys-con.com/node/612375`.

[5] HBase. An open-source, distributed, column-oriented store modeled after the Google Bigtable paper. `http://hadoop.apache.org/hbase/`.

[6] Yvon Jégou and Jérôme Gallard. Evaluation of Linux native isolation mechanisms for XtreemOS flavours. XtreemOS deliverable D2.1.6, January 2009.

[7] Avinash Lakshman, Prashant Malik, and Karthik Ranganathan. Cassandra: A structured storage system on a P2P network. In *Keynote talk at the ACM SIGMOD international conference on Management of Data*, 2008.

[8] Werner Vogels. Eventually consistent. *Communications of the ACM*, 52(1):40–44, 2009.

[9] Zhou Wei, Jiang Dejun, Guillaume Pierre, Chi-Hung Chi, and Maarten van Steen. Service-oriented data denormalization for scalable web applications. In *Proceedings of the 17th International World Wide Web Conference*, Beijing, China, April 2008.

[10] Zhou Wei, Guillaume Pierre, and Chi-Hung Chi. Scalable transactions for web applications in the cloud. In *Proceedings of the Euro-Par Conference*, Delft, The Netherlands, August 2009.

[11] Wikipedia.org. Hype cycle. `http://en.wikipedia.org/wiki/Hype_cycle`.

[12] Wikipedia.org. Infrastructure as a service. `http://en.wikipedia.org/wiki/Infrastructure_as_a_service`.

[13] Wikipedia.org. Platform as a service. `http://en.wikipedia.org/wiki/Platform_as_a_service`.

[14] Wikipedia.org. Software as a service. `http://en.wikipedia.org/wiki/Software_as_a_Service`.

[15] Wikipedia.org. Statistical multiplexing. `http://en.wikipedia.org/wiki/Statistical_multiplexing`.