Enabling JXTA for High Performance Grid Computing

Gabriel Antoniu, Mathieu Jan, David Noblet

HAL Id: inria-00070519
https://hal.inria.fr/inria-00070519

Submitted on 19 May 2006

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Enabling JXTA for High Performance Grid Computing

Gabriel Antoniu, Mathieu Jan, David A. Noblet

N˙5488

February 2005

Systèmes communicants

Rapport de recherche

Enabling JXTA for High Performance Grid Computing

Gabriel Antoniu, Mathieu Jan, David A. Noblet

Systèmes communicants
Projet Paris

Abstract: Grid computing has recently emerged as a response to the growing demand for resources (processing power, storage, etc.) exhibited by scientific applications. However, as grid sizes increase, the need for self-organization and dynamic reconfigurations is becoming more and more important. Since such properties are exhibited by P2P systems, the convergence of grid computing and P2P computing seems natural. However, using P2P systems (usually running on the Internet) on a grid infrastructure (generally available as a federation of SAN-based clusters interconnected by high-bandwidth WANs) may raise the issue of the adequacy of the P2P communication mechanisms. This paper evaluates the communication performance of the JXTA P2P library over SANs and WANs, for both J2SE and C bindings. We analyze these results and we evaluate solutions able to improve the performance of JXTA on such grid infrastructures.

Key-words: Peer-to-peer, JXTA, Communication layers, Performance

(Résumé : tsvp)

This paper has been submitted to Euro-Par 2005: Parallel Processing.

JXTA haute performance pour le calcul sur grille

Résumé : Le calcul sur grille est récemment apparu comme une réponse à la demande croissante de resources (capacité de calcul, de stockage, etc.) des applications scientifiques. Toutefois, du fait de la taille croissante des grilles, le besoin d'auto-organisation et de reconfiguration dynamiques devient de plus en plus important. Puisque ces propriétés sont exposées par les systèmes P2P, la convergence entre le calcul sur grille et le P2P semble naturelle. Toutefois, utiliser des systèmes P2P (généralement déployés sur Internet) sur une infrastructure de type grille (généralement disponible en tant que fédération de grappes interconnectées par des liens longue distance à haut débit) soulève le problème de l'adéquation des mécanismes de communication des systèmes P2P. Ce papier évalue les performances des communications de la librairie P2P JXTA sur des réseaux locaux haute performance ainsi que sur des réseaux à longue distance, pour les deux implémentations de JXTA (J2SE et C). Nous analysons ces résultats et nous évaluons les solutions qui permettent d'améliorer les performances de JXTA sur des infrastructure de type grille.

Mots-clé : Pair-à-pair, JXTA, couches de communications, Performance.

1   Using P2P techniques to build grids

Nowadays, scientific applications require more and more resources, such as processors, storage devices, network links, etc. Grid computing provides an answer to this growing demand by aggregating resources made available by various institutions. As their sizes are growing, grids express an increasing need for flexible distributed mechanisms allowing them to be efficiently managed. Such properties are exhibited by Peer-to-Peer (P2P) systems, which have proven their ability to efficiently handle millions of interconnected resources in a decentralized way. Moreover, these systems support a high degree of resource volatility. The idea of using P2P strategies for grid resource management has therefore emerged quite naturally [12, 20].

The convergence of P2P and grid computing can be approached in several ways. For instance, P2P services can be implemented on top of building blocks based on current grid technology (e.g. by using grid services as a communication layer [21]). Alternatively, P2P libraries can be used on physical grid infrastructures, for example, as an underlying layer for higher-level grid services [6]. This provides a way to leverage scalable P2P mechanisms for resource discovery, resource replication and fault tolerance. In this paper, we focus on this second approach.

Using P2P software on physical grid infrastructures is a challenging problem. Grid applications often have important performance constraints that are generally not a usual requirement for P2P systems. One crucial issue in this context is the efficiency of data transfers. Using P2P libraries as building blocks for grid services, for instance, requires the efficient use of the capacities of the networks available on grid infrastructures: System-Area Networks (SANs) and Wide-Area Networks (WANs). Often, a grid is built as a cluster federation. SANs, such as Giga Ethernet or Myrinet (which typically provide Gb/s bandwidth and a few microseconds latency), are used for connecting nodes inside a given high-performance cluster; whereas WANs with a typical bandwidth of 1 Gb/s but higher latency (typically of the order of 10-20 ms) are used between clusters. This is clearly an unusual deployment scenario for P2P systems, which generally target the edges of the Internet (those with low-bandwidth and high-latency links, such as Digital Subscriber Line, or DSL, connections). Therefore, it is important to ask: are P2P communication mechanisms adequate for a usage in such a context? Is it possible to adapt the behavior of P2P communication systems in order to benefit from the new potentials offered by these high-performance networks?

In this paper, we discuss the appropriateness of using the JXTA [4] P2P library on grid infrastructures by evaluating to what extent its communication layers are able to leverage high-performance networks. JXTA is an open-source project on which, to the best of our knowledge, most of the few attempts for realizing P2P-grid convergence have been based. In

its 2.0 version, JXTA consists in a specification of six language- and platform-independent, XML-based protocols that provide basic services common to most P2P applications, such as peer group organization, resource discovery, and inter-peer communication. A detailed description of the communications layers of JXTA can be found in [8]. This paper focuses on the evaluation of JXTA-J2SE and JXTA-C[1] over both SANs and WANs. It also discusses ways to integrate some existing solutions for improving the raw performance.

The remainder of the paper is organized as follows. Section 2 introduces the related work: we discuss some JXTA-based attempts for using P2P mechanisms to build grid services and we mention some performance evaluations of JXTA. Section 3 describes in detail the experimental setup used for both SAN and WAN benchmarks. Sections 4 and 5 present the benchmark results of JXTA over these two types of networks. Finally, Section 6 concludes the paper and discusses some possible future directions.

2   Related Work

Several projects have focused on the use of JXTA as a substrate for grid services. The Cog Kit JXTA Project [1] and the JXTA-Grid [3] project are two examples. However, none of these projects are being actively developed and not one has released any prototypes. The Service-oriented Peer-to-Peer Architecture [6] (SP2A) project aims at using P2P routing algorithms for publishing and discovering grid services. SP2A is based on two specifications: the Open Grid Service Infrastructure (OGSI) and JXTA. None of the above projects has published performance evaluations so far. Finally, JUXMEM [7] proposes to use JXTA in order to build a grid data-sharing service. All of these projects mentioned above share the idea of using JXTA as a low-level interaction substrate over a grid infrastructure. Such an approach brings forth the importance of JXTA's communications performance.

JXTA's communication layers have so far only been evaluated at a cluster level, or over the Internet via DSL connections, but not over grid architectures with high-performance clusters interconnected with high-bandwidth WANs. The performance of JXTA-J2SE communication layers has been the subject of many papers [13, 14, 17, 18, 15, 19] and has served as reference for comparisons with other P2P systems [9, 16, 22]. The most recent evaluation of JXTA-J2SE is [8], which also provides an evaluation of JXTA-C. Moreover, it gives hints on how to use JXTA in order to get good performance on Fast Ethernet LAN networks.

---

[1]the only two bindings compliant to JXTA's specifications version 2.0

3   Description of the Experimental Setup

For all reported measurements we use a *bidirectional bandwidth* benchmark (between two peers), based on five subsequent time measurements of an exchange of 100 consecutive message-acknowledgment pairs sampled at the application level. We chosed this performance metric as it is requently used for benchmarking many other networking protocols, and because of its ability to yield information about important performance characteristics such as bandwidth and latency. Benchmarks were executed using versions 2.2.1 and 2.3.2 of the J2SE binding of JXTA. For the C binding, the CVS head of JXTA-C from the 18th of January 2005 was used. Both bindings were configured to use TCP as the underlying transport protocol. When benchmarks are performed using JXTA-J2SE, the Sun Microsystems Java Virtual Machine (JVM) 1.4.2 is used and executed with `-server -Xms256M -Xmx256M` options. The use of other JVMs is explicitely noted and executed with equivalent options. Finally, the JXTA-C benchmarks are compiled using `gcc` 3.3.3 with the `O2` level of optimization. Tests were performed on the following two types of networks, typically used for building grid infrastructures.

SAN benchmarks.   The networks used for the SAN benchmarks are Giga Ethernet and Myrinet (GM driver, version 2.0.11). When the network layer is Myrinet, nodes consist of machines using 2.4 GHz Intel Pentium IV processors, outfitted with 1 GB of RAM each, and running a 2.4 version Linux kernel. For Giga Ethernet, nodes consist of machines using dual 2.2 GHz AMD Opteron processors, also outfitted with 1 GB of RAM each, and running a 2.6 version Linux kernel. Since direct communication amongst nodes of a SAN-based cluster is available, direct communications between peers has also been configured.

WAN benchmarks.   The platform used for the WAN benchmarks is the Grid5000 French national grid platform [2]. Tests were performed between two of the Grid'5000 clusters located in Rennes and Toulouse. On each side, nodes consist of machines using dual 2.2 GHz AMD Opteron processors, outfitted with 1 GB of RAM each, and running a 2.6 version Linux kernel. The two sites are interconnected through a 1 Gb/s link, with an average measured latency of 11.2 ms. Note that as direct communication between nodes is possible within the Grid5000 testbed, we configured JXTA peers to enable direct exchanges. This is clearly an unusual deployment scenario for P2P systems, where direct communication between peers is the exception rather than the rule (because of firewalls, etc.). However, let us stress that on some grids direct communication is only available between cluster frontends. In that case, additional evaluations would be necessary.

Communication protocols.   JXTA communication layers provide three basic mechanisms for inter-peer communication, with different levels of abstraction. The *endpoint*
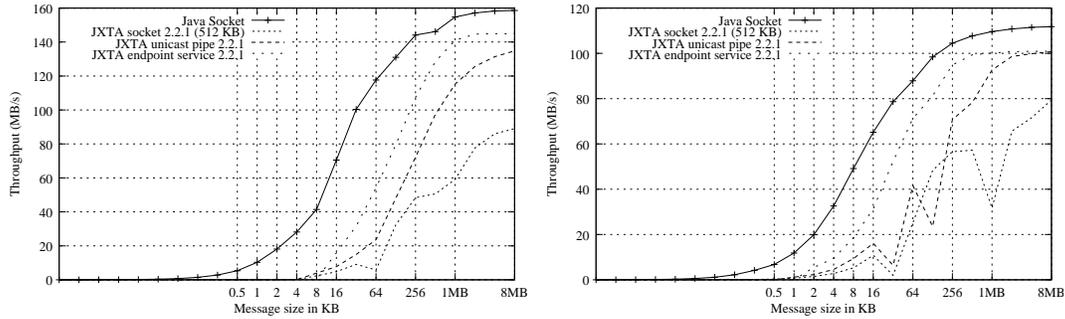
Figure 1: Bandwidth of each layer of JXTA 2.2.1 as compared to Java sockets over a Myrinet network (left) and a Giga Ethernet network (right).

service is JXTA's lowest, point-to-point communication layer which provides an abstraction for available underlying transport protocols. Messages sent by this layer are comprised of a series of named and typed *message elements* [5]. These elements may be required by higher communication layers or added by the application (e.g. the message payload). The *pipe service*, built on top of the endpoint layer, provides virtual communication channels (or *pipes*), which are dynamically bound to peer endpoints at runtime, thus allowing developers to abstract themselves from dynamic, runtime changes of physical network addresses. In this paper, we focus on point-to-point pipes, called *unicast pipes*. Finally, on top of pipes, the *JXTA sockets* add a data-stream interface, and implement reliability guarantees. JXTA sockets extend the BSD socket API, while still preserving the main feature of pipes: independence from the physical network. However, it should be noted that this layer is not part of the core specifications of JXTA. It is currently only available in JXTA-J2SE.

4   Performance Evaluation of JXTA over System-Area Networks

This section analyzes the performance of JXTA's communications layers on SANs. Note that for Myrinet, the *Ethernet emulation* mode of GM 2.0.11 is used and configured with jumbo frames. This mode allows Myrinet to carry any packet traffic and protocols that can be transported by Ethernet, including TCP/IP. Although this capability is bought at the cost of losing the main advantage of a Myrinet network (e.g. OS-bypass mode), it allows the same socket-based benchmarks to be run unmodified. On this configuration, the bandwidth and latency of plain sockets is around 155 MB/s and 60 μs respectively, whereas on Giga Ethernet it is around 115 MB/s for the bandwidth and 45 μs for the latency (average values between C and Java sockets). These values are used as a reference performance bound.
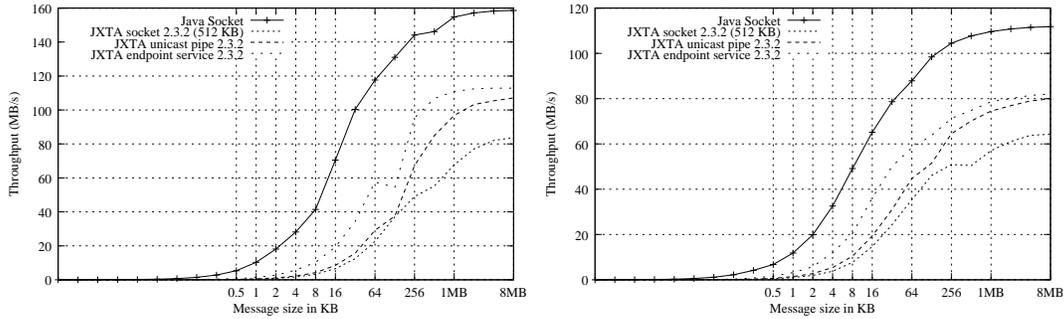
Figure 2: Bandwidth of each layer of JXTA 2.3.2 as compared to Java sockets over a Myrinet network (left) and a Giga Ethernet network (right).

4.1   Analysis of JXTA-J2SE's Performance

JXTA-J2SE endpoint service.   Figure 1 shows that the endpoint service of JXTA 2.2.1 nearly reaches the bandwidth of plain sockets over SAN networks: 145 MB/s over Myrinet and 101 MB/s over Giga Ethernet. However, Figure 2 also shows that the bandwidth of JXTA 2.3.2 endpoint layer has decreased: drops of 32 MB/s over Myrinet and 30 MB/s over Giga Ethernet are observed. These lower bandwidths affect all versions of JXTA above its release 2.2.1 and are explained by a new implementation of the endpoint layer that shipped with JXTA 2.3. The profiling of JXTA has pointed out that this drop of performance is due to the mechanism used for limiting the size of messages sent by the endpoint layer. Moreover, since JXTA 2.3, the limit has been lowered to 128 KB of application-level payload (larger messages are dropped). This limitation was introduced into JXTA to in order to promote some fairness in resource sharing among peers on the network, for instance when messages must be stored on relay peers (the type of peer required to exchange messages through firewalls). However, as no relay peers are needed when using SANs, we removed this limit. Table 1 shows that latency results of JXTA-J2SE have improved since version 2.2.1. The latency of the JXTA 2.3.2 endpoint service over Giga Ethernet reaches a value under 300 μs. Moreover, it goes down even further to 268 μs and 229 μs when using the SUN 1.5 and IBM

Version of JXTA	JXTA-J2SE 2.2.1		JXTA-J2SE 2.3.2	
Network	Myrinet	Giga Ethernet	Myrinet	Giga Ethernet
Endpoint service	890 μs	357 μs	624 μs	294 μs
Unicast pipe	1.9 ms	834 μs	1.7 ms	711 μs
JXTA socket	3.3 ms	1.3 ms	2.4 ms	977 μs

Table 1: Latency results for JXTA-J2SE.

1.4.1 JVMs, respectively. The difference between Myrinet and Giga Ethernet results is due to the hardware employed, as the Ethernet emulation mode is used for Myrinet.

JXTA-J2SE unicast pipe.   In addition, Figures 1 and 2 demonstrate a bandwidth degradation for JXTA-J2SE. For example, while JXTA 2.2.1 unicast pipe attains a good peak bandwidth of 136.8 MB/s over Myrinet, its 2.3.2 counterpart reaches a bandwidth of only 106.5 MB/s. A similar performance degradation can be observed on Giga Ethernet. However, the shape of the curve of unicast pipes 2.2.1 on Giga Ethernet has not been explained so far. We suspect the first drop is due to a scheduling problem. On the other hand, the reason of the drop at 128 KB of application payload is still unknown. At the same payload size, a smaller drop for JXTA unicast pipes 2.3.2 over Giga Ethernet can be observed, but no link have been established with the previously mentioned drop, as this drop also occurs at the endpoint level. Overall, the small performance degradation as compared to the endpoint layer is explained by the composition of a pipe message: the presence of an XML message element requiring a costly parsing prevents this layer from reaching the performance of the endpoint layer. Moreover, as shown on table 1, this extra parsing required for each pipe message also affects latency results: compared to the endpoint layer, latencies increase by more than 400 μs. However, unicast pipes are still able to achieve latencies in the sub-millisecond range, at least on Giga Ethernet.

JXTA-J2SE sockets.   As opposed to previous layers, JXTA sockets are far from reaching the performance of plain Java sockets. Indeed, in their default configuration (e.g. with an output buffer size of 16 KB), JXTA sockets 2.2.1, for instance, attain a peak bandwidth of 12 MB/s over a Myrinet network. As for unicast pipes, a similar low bandwidth result is reported on Giga Ethernet. We were able to significantly improve the bandwidth and achieve 92 MB/s by increasing the size of the output buffer to 512 KB, as shown on Figures 1 and 2. As for the unicast pipes, the irregular shape of JXTA sockets 2.2.1 curves has not been explained so far. Again, we suspect the first drop is due to a scheduling problem. The next drop may be due to some message losses when the message size is around the size of the output buffer, since many reliability issues have been fixed up to JXTA 2.3.2. Table 1 highlights the progress being made by JXTA on latency, as only JXTA Sockets 2.3.2 on Giga Ethernet is able to reach a latency under one millisecond.

Discussion.   In conclusion, JXTA-J2SE 2.2.1 communication layers are able to nearly saturate SANs, but only at the endpoint and pipe levels. The measurements revealed that the bandwidth of JXTA 2.2.1 is higher than JXTA 2.3.x. Latency results have largely improved since JXTA 2.2.1, but without reaching reasonably good performance for SANs. Finally, this evaluation has also highlighted that, in their default configuration, JXTA sockets achieve a very poor bandwidth. However, this result can significantly be improved by increasing the
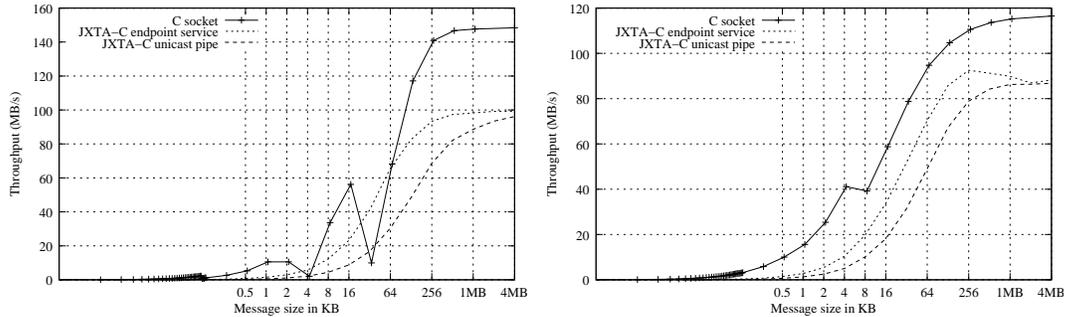
Figure 3: Bandwidth of each layer of JXTA-C as compared to C sockets over a Myrinet network (left) and a Giga Ethernet network (right).

output buffer size. This requires the JXTA socket programmer to explicitly set this parameter in the user code. Based on these results, we can conclude that JXTA-J2SE can be adapted in order to benefit from the potential offered by SANs, at least on the bandwidth side.

4.2   Analysis of JXTA-C's Performance

Figure 3 shows the bandwidth measurements of all the communications layers of JXTA-C over SANs. Note that, as in the previous section, C sockets are used as an upper reference bound. The peak bandwidths of the endpoint service over Myrinet and Giga Ethernet are 109.6 MB/s and 88.2 MB/s, respectively. The upper layer (unicast pipe) reaches bandwidths of 107.7 MB/s and 87.7 MB/s over Myrinet and Giga Ethernet, respectively. These unsatisfactory results are due to memory copies being used in the implementation of the endpoint layer of JXTA-C. Table 2 highlights reduced latencies, especially on Giga Ethernet, as compared to results published in [8]: 820 μs for the endpoint layer and 1.99 ms for the pipe layer. To achieve this improvement, we modified the implementation of the endpoint layer of JXTA-C by disabling TCP packet aggregation mechanism, as it adds significantly to the latency. Consequently, the buffering mechanism is now performed within the endpoint layer allowing one TCP packet to be sent for a single JXTA message with a minimal latency. These modifications have been committed into the CVS of JXTA-C and are publicly available.

Network	Myrinet	Giga Ethernet
Endpoint service	635 μs	322 μs
Unicast pipe	1.7 ms	727 μs

Table 2: Latency results for JXTA-C.

Based on this evaluation, we can conclude that, in their current implementation, the communication layers of JXTA-C are not able to saturate SANs. The non-zero copy implementation of the endpoint layer prevents JXTA-C from approaching Gb/s bandwidths available over SANs. Note, however, that JXTA-C is in the process of being revived; so we believe that the performance of JXTA-C will increase in the near future such that it will be able to efficiently use SANs.

4.3   Fully exploiting SAN capacities

In all previously reported evaluations based on Myrinet, the Ethernet emulation mode of GM is used. However, this removes the ability to by-pass the IP stack of the OS and introduces unneeded overhead. Consequently, communication layers are unable to fully exploit the capacities offered by Myrinet: full-duplex bandwidths of nearly 2 Gb/s and latencies of less than 7 μs thanks to zero-copy communication protocols. PadicoTM [11] is a high-performance framework for networking and multi-threading which allows middleware systems to transparently take advantage of such features. In this section, we focus on the *virtual sockets* feature offered by PadicoTM, which provides a way to directly access GM network interfaces. This is achieved by dynamically mapping, at runtime, standard sockets functions on top of GM API functions, without going through the TCP/IP stack. Zero-copy is therefore possible and allows, for example, plain sockets to transparently reach a bandwidth of more than 230 MB/s and latency of 15 μs on Myrinet, compared to 160 MB/s and 51 μs without PadicoTM.

We have successfully ported JXTA-C to PadicoTM, without changing one line of code of JXTA-C. We only performed some minor modifications inside the OS-independent layer used by JXTA-C: the Apache Portable Runtime (APR). We changed from the default Posix thread library on which APR is based, to the Marcel [10] thread library used by PadicoTM. However, these modifications could be automatically achieved by a single `sed` command. An improvement of 32 MB/s for the bandwidth of JXTA-C's endpoint layer has been measured resulting in a peak bandwidth of 140 MB/s, thus reaching over 1 Gb/s. On the latency side, no significant improvements have been observed, as the non-zero copy communication layers prevents JXTA-C from fully benefiting from the OS-bypass feature of Myrinet. Note that we did not use PadicoTM with JXTA-J2SE, since PadicoTM currently supports only the open-source Kaffe JVM.

In conclusion, our experiments with PadicoTM show that JXTA could fully benefit from the potential performance of SAN networks if: 1) the implementation of all JXTA communication layers should respect a zero-copy policy, 2) PadicoTM adds support for JXTA-compatible JVMs (e.g. compliant to version 1.4 of Java's specifications). However, we believe that these issues will be solved in the near future.
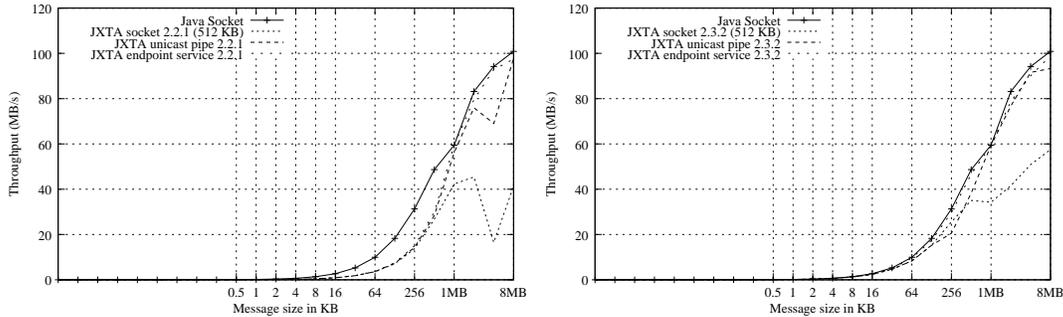
Figure 4: Bandwidth of each layer for JXTA-J2SE 2.2.1 (left) and 2.3.2 (right) compared to Java sockets over a high-bandwidth WAN.

5   Performance Evaluation of JXTA over Wide-Area Networks

We performed the same type of measurements on WAN networks. Note that we had to tune network settings of nodes used for this benchmark. Our default maximum TCP buffer size initially set to 131072 bytes was limiting the bandwidth to only 7 MB/s. Based on the $bandwidth * delay$ law, we computed a theoretical maximum size of 1507328 bytes and increased this value by an arbitrary factor of 1.2. Therefore, we set the maximum TCP buffer sizes on each node to 1959526 bytes; `ttcp` configured with this value measured a raw TCP bandwidth of 107 MB/s, a reasonable level of performance.

JXTA-J2SE's performances.   As for SAN Giga Ethernet benchmarks, Figure 4 shows that the endpoint layer and unicast pipes of JXTA-J2SE are able to perform similarly to plain sockets over a high-bandwidth WAN of 1 Gb/s. This level of performance was reached by modifying JXTA-J2SE's code in order to properly set TCP buffer sizes to 1959526 bytes before binding sockets on both sides. In the default setting, a bandwidth of only 6 MB/s was reached for JXTA 2.2.1 and less than 1 MB/s for JXTA 2.3.2. As opposed to SAN benchmarks, both versions of JXTA-J2SE achieve the same performance. This can be explained by the the fact that the higher latency of WANs hides the cost of the mechanism implemented for limiting the size of JXTA messages. Figure 4 also points out the same performance degradation for JXTA sockets as for SAN benchmarks. However, JXTA socket 2.3.2 achieves a higher bandwidth compared to its 2.2.1 counterpart. Performance drops of unicast pipes and JXTA sockets for JXTA-J2SE 2.2.1 for message size of 4 MB have not been explained so far.

JXTA-C's performances.   Figure 5 shows similar results for the communication layers of JXTA-C over WANs compared to the SAN benchmarks: both layers reach a bandwidth
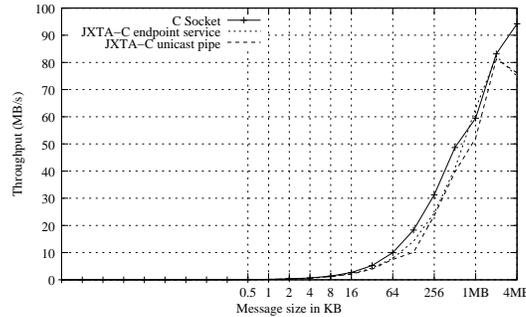
Figure 5: Bandwidth of each layer for JXTA-C compared to C sockets over a high-bandwidth WAN.

slightly above 80 MB/s, for a message size of 2 MB. The observed performance degradation after this message size has not been explained.

Discussion.   Based on this evaluation, we can conclude that JXTA's communication layers, when used on high-bandwidth WANs, are able to reach the same bandwidths as for SAN benchmarks. Both versions of JXTA-J2SE (and not only JXTA-J2SE 2.2.1) are able to efficiently use the bandwidth available on links used for interconnecting sites of a grid, whereas the non-zero copy communication layers prevents JXTA-C from saturating this type of links.

6   Conclusion

In the context of the current efforts for building grid services on top of P2P libraries, the question of the adequacy of the P2P communication mechanisms for performance-constrained usages is important. In this paper, we evaluate the basic performance of the communication layers of the JXTA generic P2P framework on the Grid5000 testbed. Giga Ethernet and Myrinet links are used as SANs within each site, while high-bandwidth WANs interconnect the various sites. We provide an analysis of the performance of the J2SE and C bindings of JXTA over these two types of networks. We show that the JXTA-J2SE 2.2.1 communication layers are able to nearly saturate SANs, at the endpoint and pipe levels, whereas the bandwidth is poor at the JXTA socket level. We also show how to improve this bandwidth by tuning the TCP output buffer size. Performance can further be improved on SANs by using the PadicoTM environment, which provides a direct access to the API of the Myrinet driver by by-passing the OS.

However, these evaluations have revealed some weaknesses of JXTA in both the SAN and WAN areas. JXTA-J2SE bandwidths have degraded since JXTA 2.3, preventing JXTA

from saturating SAN links. Moreover, the communication layers of JXTA-C do not follow a zero-copy policy, therefore limiting the bandwidth and latency results. Therefore, in spite of our initial efforts, JXTA-C still needs some improvements in order to be able to fully benefit of the available bandwidth provided by SANs. When these issues are solved a bandwidth of over 200 MB/s should be reached through the use of PadicoTM. On the WAN side, we plan to use parallel streams for both bindings of JXTA, in order to allow an efficient use of high-bandwidth WANs. Again thanks to PadicoTM, this functionality will be transparently available to JXTA-C, whereas for JXTA-J2SE this would require to implement this functionality. Finally, it would also be interesting to see the effect of on-the-fly compression techniques available in PadicoTM for WAN transfers.

References

[1] Cog Kit JXTA project. `http://www-unix.globus.org/cog/projects/jxta/`.

[2] Grid'5000 project. `http://www.grid5000.org/`.

[3] JXTA-Grid project. `http://jxta-grid.jxta.org/`.

[4] The JXTA project. `http://www.jxta.org/`.

[5] JXTA specification project. `http://spec.jxta.org/`.

[6] Michele Amoretti, Gianni Conte, Monica Reggiani, and Francesco Zanichelli. Service Discovery in a Grid-based Peer-to-Peer Architecture. In *International Workshop on e-Business and Model Based IT Systems Design*, Saint Petersburg, Russia, April 2004.

[7] Gabriel Antoniu, Luc Bougé, and Mathieu Jan. JuxMem: Weaving together the P2P and DSM paradigms to enable a Grid Data-sharing Service. *Kluwer Journal of Supercomputing*, 2005. To appear.

[8] Gabriel Antoniu, Phil Hatcher, Mathieu Jan, and David A. Noblet. Performance Evaluation of JXTA Communication Layers (extended version). In *5th International Workshop on Global and Peer-to-Peer Computing (GP2PC '05)*, Cardiff, UK, May 2005. To appear, preliminary version available as an INRIA Research Report RR-5469.

[9] Sébastien Baehni, Patrick Th. Eugster, and Rachid Guerraoui. OS Support for P2P Programming: a Case for TPS. In *22nd International Conference on Distributed Computing Systems (ICDCS '02)*, pages 355–362, Vienna, Austria, July 2002. IEEE Computer Society.

[10] V. Danjean, R. Namyst, and R. Russell. Integrating Kernel Activations in a Multithreaded Runtime System on Linux. In *Parallel and Distributed Processing. Proc. 4th Workshop on Runtime Systems for Parallel Programming (RTSPP '00)*, volume 1800 of *Lect. Notes in Comp. Science*, pages 1160–1167, Cancun, Mexico, May 2000. In conjunction with IPDPS 2000. IEEE TCPP and ACM, Springer-Verlag.

[11] Alexandre Denis, Christian Pérez, and Thierry Priol. PadicoTM: An Open Integration Framework for Communication Middleware and Runtimes. *Future Generation Computer Systems*, 19(4):575–585, May 2003.

[12] Ian Foster and Adriana Iamnitchi. On Death, Taxes, and the Convergence on Peer-to-Peer and Grid Computing. In *2nd International Workshop on Peer-to-Peer Systems (IPTPS '03)*, number 2735 in Lect. Notes in Comp. Science, Berkeley, CA, February 2003. Springer-Verlag.

[13] Emir Halepovic and Ralph Deters. The Cost of Using JXTA. In *3rd International Conference on Peer-to-Peer Computing (P2P '03)*, pages 160–167, Linköping, Sweden, September 2003. IEEE Computer Society.

[14] Emir Halepovic and Ralph Deters. JXTA Performance Study. In *IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM '03)*, pages 149–154, Victoria, B.C., Canada, August 2003. IEEE Computer Society.

[15] Emir Halepovic and Ralph Deters. JXTA Messaging: Analysis of Feature-Performance Trade-offs. Submitted for publication, 2005.

[16] Markus Junginger and Yugyung Lee. The Multi-Ring Topology - High-Performance Group Communication in Peer-to-Peer Networks. In *2nd International Conference on Peer-to-Peer Computing (P2P '02)*, pages 49–56, Linköping, Sweden, September 2002. IEEE Computer Society.

[17] Jean-Marc Seigneur. Jxta Pipes Performance. `http://bench.jxta.org/papers/jmjxtapipesperformance.pdf`, 2002.

[18] Jean-Marc Seigneur, Gregory Biegel, and Christian Damsgaard Jensen. P2P with JXTA-Java pipes. In *2nd international Conference on Principles and Practice of Programming in Java (PPPJ '03)*, pages 207–212, Kilkenny City, Ireland, 2003. Computer Science Press, Inc.

[19] Kazuyuki Shudo, Yoshio Tanaka, and Satoshi Sekiguchi. P3: Personal Power Plant. GGF10: Open Grid Service Architecture - Peer-to-Peer Research Group (OGSA-P2P RG), March 2004.

[20] Domenico Talia and Paolo Trunfio. Toward a Synergy Between P2P and Grids. *IEEE Internet Computing*, 7(4):94–96, 2003.

[21] Domenico Talia and Paolo Trunfio. A P2P Grid Services-Based Protocol: Design and Evaluation. In *Euro-Par 2004: Parallel Processing*, number 3149 in Lect. Notes in Comp. Science, pages 1022–1031, Pisa, Italy, August 2004. Springer-Verlag.

[22] Phong Tran, Jeffrey Gosper, and Albert Yu. JXTA and TIBCO Rendezvous - An Architectural and Performance Comparison. `http://www.smartspaces.csiro.au/docs/PhongGosperYu2003.pdf`, 2003.