Performance Analysis of Publish/Subscribe Systems

Heithem Abbes[1], Christophe Cérin[2], Jean-Christophe Dubacq[2], and Mohamed Jemni[1]

[1] École Supérieure des Sciences et Techniques de Tunis, Unité de recherche UTIC
5, Av. Taha Hussein, B.P. 56, Bab Mnara, Tunis, Tunisia
Tel: (+216) 71 496 066 Fax: (+216) 71 391 166
heithem.abbes@esstt.rnu.tn, mohamed.jemni@fst.rnu.tn

[2] LIPN — UMR CNRS 7030 — Institut Galilée — Université Paris-Nord
99, avenue Jean-Baptiste Clément, 93430 Villetaneuse, France
Tel: +33(0)1 49 40 35 78 Fax: +33(0)1 48 26 07 12
{christophe.cerin,jean-christophe.dubacq}@lipn.univ-paris13.fr

Abstract. The Desktop Grid offers solutions to overcome several challenges and to answer increasingly needs of scientific computing. Its technology consists mainly in exploiting resources, geographically dispersed, to treat complex applications needing big power of calculation and/or important storage capacity. However, as resources number increases, the need for scalability, self-organisation, dynamic reconfigurations, decentralisation and performance becomes more and more essential. Since such properties are exhibited by P2P systems, the convergence of grid computing and P2P computing seems natural. In this context, this paper evaluates the scalability and performance of P2P tools for discovering and registering services. Three protocols are used for this purpose: Bonjour, Avahi and Free-Pastry. We have studied the behaviour of theses protocols related to two criteria: the elapsed time for registrations services and the needed time to discover new services. Our aim is to analyse these results in order to choose the best protocol we can use in order to create a decentralised middleware for desktop grid.

Key words: Peer-to-Peer systems, Desktop grid, Performance evaluation, Zero-Configuration, Bonjour, Avahi, mDNSResponder, Free-Pastry

1 Introduction

The exploitation of new instruments, such as, very high energy accelerators, telescopes and satellites in astrophysics, big data bases of imagery in biology and medicine, numerous sensors in geology, generated an important expansion of the needs in scientific computation. Thus, it becomes necessary to establish new computing infrastructures. On the other hand, the computer networks equipments knew these last years, an important development of transmission speed performances and devices became equipped by powerful processors and important storage capacities.

These factors advantaged the emergence of new infrastructures, such as grid computing, to respond to computation needs with an economic cost. One variant of grid computing is the Desktop Grid where nodes are merely desktop PCs. This category constitutes the setting of our work.

Research in grid has developed some specific software (middleware) for the management of data and resources. The most Desktop Grid middlewares are centralised. In this setting, our work consists in conceiving a decentralised grid computing middleware based on peer-to-peer systems. To realise this, we would profit from existing decentralised peer-to-peer systems.

The service discovery in the Grid is among the principle challenges. For instance, Globus middleware implemented the service publish/discovery mechanism based on Monitoring and

Discovery of Services (MDS-2) [7, 6] which uses centralised register server. Although MDS-2 solved the scalability problem using hierarchical architecture, it is still vulnerable to single point of failure. Moreover, adaptation to the dynamic feature of servers is another challenge for MDS-2. Furthermore, another alternative consists of using decentralised approach for service discovery [3, 15]. Recently, P2P communities have developed a number of fully decentralised protocols, such as Bonjour [12, 17, 27], Avahi [21] and Pastry [11, 22] for registering, routing and discovering in P2P networks. The core idea behind these protocols is to build self-organised overlay networks when nodes join the grid. On the other hand, it is important to know the performance and the limits of such systems. In this context, several experiments have been done in this work to analyse the performance of Bonjour, Avahi and Free-Pastry.

We choose Bonjour and Avahi (two popular middlewares running on a local area network) because our working context is the connectivity issues that we are faced to when we try to share resources belonging to different institutions. In this paper, we assume that we have a high level middleware able to virtualise the network (we have no more problems with firewalls and NAT) and we are able to run Bonjour and Avahi on top of such middleware. Instant Grid / Private Virtual Cluster (see [10, 24]) is one of the candidate for network virtualisation. Its main requirements are: 1) simple network configuration 2) no degradation of resource security 3) no need to re-implement existing distributed applications. Under these assumptions, it is reasonable to check if Bonjour and Avahi can scale up.

This paper is organised as follows. In section 2, we remind the notion of large scale distributed systems by focusing on grid computing and peer-to-peer systems. Then, in the section 3, we illustrate the notion of desktop grid. In section 4, we highlight the advantage of peer to peer systems in building a new decentralised middleware for grid computing. In section 5, we describe the experimental setup used to evaluate the performance of Bonjour, Avahi and Free-Pastry. In section 6 and section 7, we provide numerical results obtained from several experiments done on Grid'5000 (we have used up to 308 machines). We finish this paper with some prospective and a conclusion, respectively in section 8 and 9.

2 Distributed large scale systems

2.1 Grid Computing

In [4], Foster and Kesselman define grid computing as follows: "A computational grid is a hardware and software infrastructure that provides dependable, consistent, pervasive, and inexpensive access to high-end computational capabilities".

Thus, a Grid Computing or a Computational grid is a hardware and software infrastructure allowing the sharing of a big number of heterogeneous resources thanks to connection between several sites. The resources sharing objective is to resolve problems confronted in organisations which are often multi-sites and require an important volume of data and an important computation power. These organisations are called virtual organisations (VO). A computational grid is analogous to the electric network which permits to any subscriber, at any time, to accede instantaneously to the electric resource whatever is its origin or location, via standardised interface [8, 7]. But the grid computing offers more services than the electric grid and should guarantee criteria of reliability, security and access transparency while taking account of constraints of the high throughput and the choice of the quality of service (QoS).

2.2 Peer-to-peer System

One definition of a peer-to-peer system is: “peer-to-peer refers to a class of systems and applications that employ distributed resources to perform a critical function in a decentralised manner” [1].

Exchanges between systems can carry on the information, the processors cycles, the memory or the files storage on disk. Contrary to the client/server model, each node is a network entity which has the roles of the server and the client at the same time. With peer-to-peer, the personal computer can be part of the network. The peer-to-peer concerns a class of applications which require hardware or human resources available on the Internet. We distinguish two types of peer-to-peer systems: 1) files sharing systems such as Gnutella, Napster and Kazaa, which knew a great success on Internet and 2) intensive computation oriented systems, equivalent to computational grids, such as SETI@Home [20], XtremWeb [26] and XtremWeb-CH [25].

3 Desktop Grid

Grids aim at providing a powerful infrastructure with quality-of-service (QoS) guarantees to average size, homogeneous resources and certified communities. In contrast, Peer-to-Peer systems focus on constructing a very large infrastructure from larger communities of entrusted, anonymous individuals and volatility resources. However, the convergence of the two systems seems natural [14, 5]. In fact, P2P research focuses more and more on providing infrastructure and diversifying the set of applications; Grid research is starting to pay attention to increasing scalability. Desktop Grid combines the two concepts.

In this context, we aim at developing a new decentralised desktop grid middleware using the features offered by several peer-to-peer tools as Bonjour, Avahi and Free-Pastry. We will not build a new middleware from scratch, but we would choose the most adequate protocol from these three ones to build a decentralised middleware. Remark that others protocols exist such as CAN [9] and CHORD [13], but we choose these three protocols because Bonjour and Avahi are two implementations of Zero-configuration which already proved good success in local area networks and for small organisations, whereas Free-Pastry, which is very similar to CAN and CHORD, is based on DHT (Distributed Hash Table).

In next section, we expose how we can build a Desktop Grid by using Peer-to-Peer technology.

4 Using Peer-to-Peer techniques to build Desktop Grid

To provide a powerful Desktop Grid, it is important to have an important number of resources. Therefore, it is necessary to integrate resources made available by several institutions. The bottleneck, that limits the scalability of such systems, is the centralisation character of existing tools (see [26, 25] for the XtremWeb platforms or [16] for the Boinc platform). Thus, it is primordial that grids need more flexible distributed mechanisms allowing them to be efficiently managed. Such characteristics are presented by Peer-to-Peer systems, which proved their performance and ability to manage very big number of interconnected peers in a decentralised manner. In addition, theses systems support high volatility of resources.

Below, we describe three Peer-to-Peer systems, Bonjour, Avahi and Free-Pastry, which are the candidates of our experiments tests.

4.1 Bonjour

Bonjour, also known as zero-configuration networking, enables automatic discovery of computers, devices, and services on IP networks. Bonjour uses industry standard IP protocols to allow devices to automatically discover each other without the need to enter IP addresses or configure DNS servers. Furthermore, Bonjour can allocate IP addresses without a DHCP server, can translate between names and addresses without a DNS server and can locate or advertise services without using a directory server.

As a technical level, zero-configuration is a combination of three technologies: link-local addressing, Multicast DNS, and DNS Service Discovery. Link local addressing is viewed a safety net. When DHCP fails or is not available, link-local addressing lets a computer make up an address for itself, so that it can, at least, communicate on the local link, even if wider communication is not possible. Like link-local addressing, Multicast DNS is a safety net, so that when conventional DNS servers are unavailable, unreachable, badly configured or otherwise broken, computers and devices can still refer to each other by name in a way that is not dependent on the correct operation of outside infrastructure. DNS Service Discovery is built on top of DNS. It works not only for with Multicast DNS (for discovering local services) but also with good old-fashioned, wide-area Unicast DNS (for discovering remote services).

4.2 Avahi

Avahi is a system which facilitates service discovery on a local network. This means that you can plug your laptop or computer into a network and instantly be able to view other people you can chat with, find printers to print or find files being shared. Avahi is mainly based on mDNS implementation for Linux. It allows programs to publish and discover services and hosts running on a local network with no specific configuration.

Avahi is an Implementation of DNS Service Discovery and Multicast DNS specifications for Zero-configuration Networking. It uses D-Bus for communication between user applications and a system daemon. The daemon is used to coordinate application efforts in caching replies, necessary to minimise the traffic imposed on networks.

4.3 Free-Pastry

Free-Pastry is a generic, scalable and efficient substrate for peer-to-peer applications. Free-Pastry nodes form a decentralised, self-organising and fault-tolerant overlay network within the Internet. Free-Pastry provides efficient request routing, deterministic object location and load balancing in an application-independent manner. Furthermore, Free-Pastry provides mechanisms that support and facilitate application-specific object replication, caching, and fault recovery.

Free-Pastry performs application-level routing and object location in a potentially very large overlay network of nodes connected via the Internet. It can be used to support a variety of peer-to-peer applications, including global data storage, data sharing, group communication and naming.

Each node in the Free-Pastry network has a unique identifier (nodeId). When presented with a message and a key, a Free-Pastry node efficiently routes the message to the node with a nodeId that is numerically closest to the key, among all currently live Free-Pastry nodes. Each Free-Pastry node keeps track of its immediate neighbours in the nodeId space, and notifies applications of new node arrivals, node failures and recoveries. Free-Pastry takes into account network locality; it seeks to minimise the distance messages travel, according to a scalar proximity metric like the number of IP routing hops. Free-Pastry is completely decentralised, scalable, and self-organising; it automatically adapts to the arrival, departure and failure of nodes.

5 Description of the experimental setup

Our goal is to study the scalability and the time response of the tools described in the previous section. In fact, we focus on searching the maximum number of supported registration nodes and the response time to discover a given service. Note that the same benchmarks are applied for the three Peer-to-Peer systems (Bonjour, Avahi and Free-Pastry). The experimental platform is Grid'5000, highly reconfigurable and controllable experimental grid platform gathering 9 sites geographically distributed in France. Every site hosts a cluster from 256 CPU to 1K CPU. All sites are connected by RENATER (10 Gb/s).

5.1 Specific kernel on Grid'5000

Grid'5000 [23] offers an infrastructure with standard kernels. To run our experimental test, it is necessary to customise one kernel to support Avahi, Bonjour and Free-Pastry. Thus, we create a specific kernel containing the entire needed package to run our codes (registration and discovering codes for each system). After that, using the two tools OAR and Kadeploy (see [19, 18]), we reserve and we deploy this specific kernel in all the reserved machines. We use only one site and all machines are made with AMD Opteron processors with a 1 Gb/s network card.

5.2 Sequential registrations

In this test, the first step is to reserve N nodes on Grid'5000 (N will vary from 100 nodes until a value for witch we observe a saturation of the registration service). The number N represents the maximum nodes that can be used for the experiment. Each node requests a registration for a given service at given time. Initially, all nodes have the needed codes to request a service but are inactive. Let δ be the activation time. We activate sequentially all the requests (and we receive back an acknowledgement). Indeed, the k^{th} request will be activated at time $k \times \delta$. We increase δ to analyse the behaviour of the system when the delay between events becomes larger.

Obviously, at the beginning the number of registration is small, thus the time of registration will be fast. We increase N until the saturation value (i.e. the registration service no longer responds for a new registration). We aim at analysing the scalability of the system without overloading the network: in this test, only one multicast appears at a given time.

5.3 Simultaneous registrations

In the first test, the registrations are done sequentially. This leads to a limited number of communications to exchange information. In this experiment, we stress the scalability of the system and its capacity to manage the communications between the registered nodes. Therefore, we request N (the number of reserved nodes) simultaneous registrations and we compute the time to complete the registration step. If we obtain a "reasonable" response time, we increase N until the saturation value. In others words, we are looking for the maximum registered nodes that the system handle when the network is overloaded by several multicast packet headers at the same time.

5.4 Periodic registrations

It's also important to study the efficiency of the system when there are some nodes with the high volatility property. In such case, the system needs to be updated by sending the global state to each node.

To simulate such behaviour, we register N nodes then we cancelled ψ services $(N > \psi)$ and we register them again randomly. It is clear that the value of ψ influences the efficiency of the system. Therefore, we modify this value to obtain the maximum value for the volatility of nodes.

5.5 Real registrations

In the periodic registration experiment, we simulate only one disconnection/registration and this does not correspond to the real behaviour of the operational grid systems since disconnections are more frequent. In this test the same set of nodes is connected/disconnected for several times. In this context, we are approaching the behaviour of P2P systems and we measure the efficiency of such systems if they interact as real grid system.

5.6 Browsing services

The other important metric is the time needed to browse a given service. Indeed, in all the previous tests, we compute the registration time. We need also to compute the discovering time which is the elapsed time between the end of the registration of a unique service and the date at which a browser node has discovered the service.

Note that the response time depends on the replicas number of a given services and the registered nodes. The browsing program listen any new event, i.e. a new registration or deleting services. With the four setup mentioned before, we can analyse the performance of the discovery service of the system. We have also the possibility to increase the number of browsers. We draw the chart where point (i, j, k) is the response time i for browser j when we use a total of k browsers.

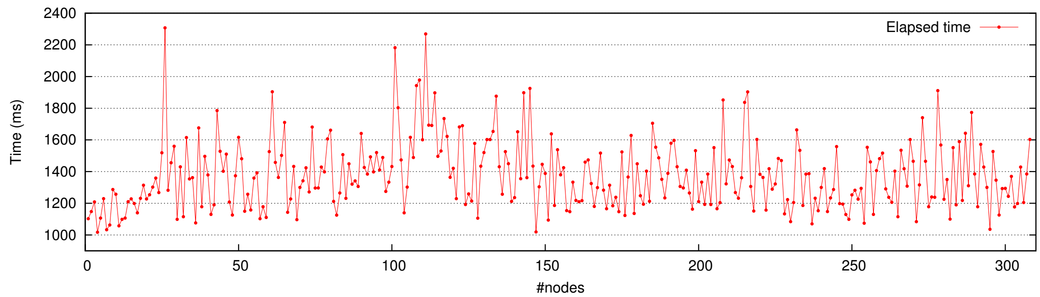

Fig. 1. Elapsed time for simultaneous registrations of Bonjour services

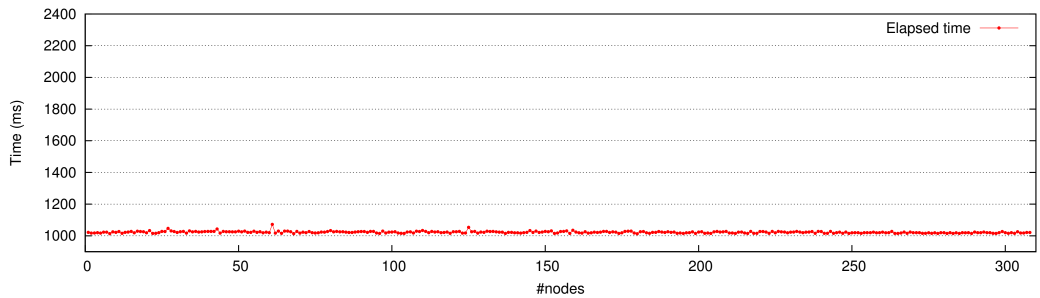

Fig. 2. Elapsed time for sequential registrations of Bonjour services

6 Performance of registration services analysis

6.1 Registration of Bonjour services

Figure 1 corresponds to a simultaneous registration. At the instant t we launch a registration of one service on each machine (the activation time is the same for all registrations). With up to 308 machines, the elapsed time of registration varies between 1017 and 2307 ms. Nevertheless, better registration times are given by the sequential test. In this test, every minute, we activate a registration request of a service. Figure 2 shows that the elapsed time for the majority of services is between 1015 and 1030 ms. We mention that Bonjour does not show saturation until 308 machines in both simultaneous and sequential registration.

6.2 Registration of Avahi services

Figure 3 and 4 point out that Avahi has almost the same registration time in both simultaneous and sequential tests. The elapsed time varies between 760 and 1110 ms. Comparing to Bonjour, Avahi has better registration times. Note that with up to 308 machines Avahi has not been saturated.

6.3 Registration of Free-Pastry services

Contrarily to Avahi and Bonjour, Free-Pastry shows a big difference between sequential and simultaneous registration tests. Indeed, figure 5 shows that in the simultaneous registration, until the 160^{th} service, the elapsed time varies between 600 and 1000 ms. Beyond, the registration time increases from one registration to another to reach 320000 ms.

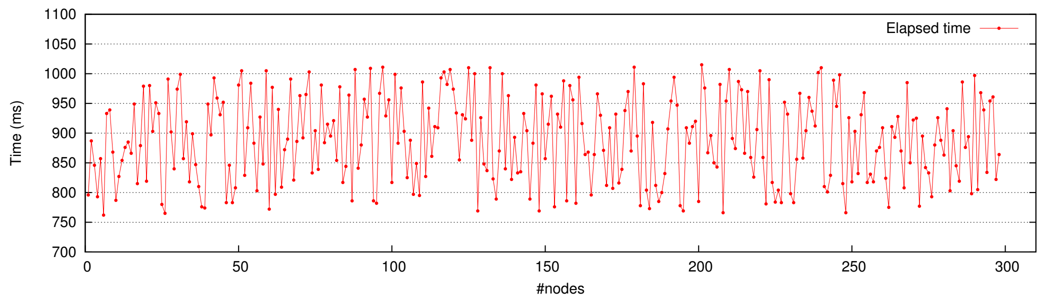

Fig. 3. Elapsed time for simultaneous registrations of Avahi services

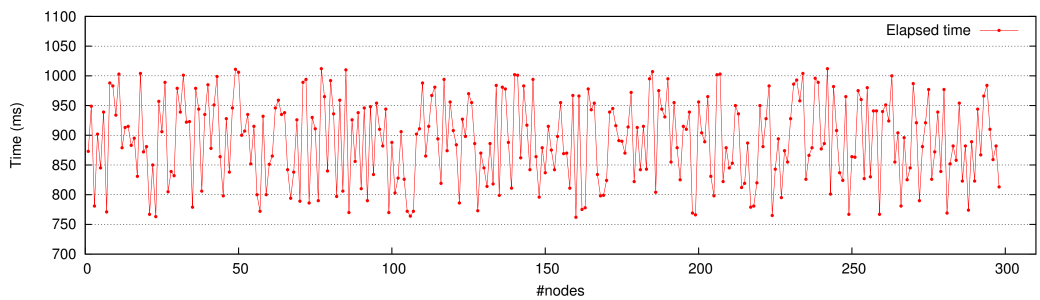

Fig. 4. Elapsed time for sequential registrations of Avahi services

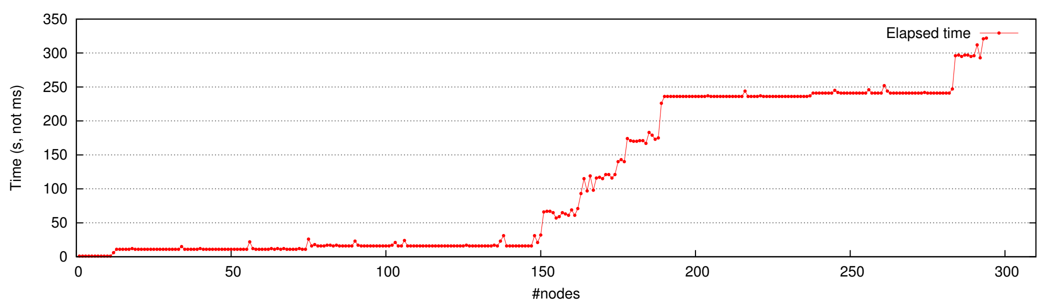

Fig. 5. Elapsed time for simultaneous registrations of Free-Pastry services

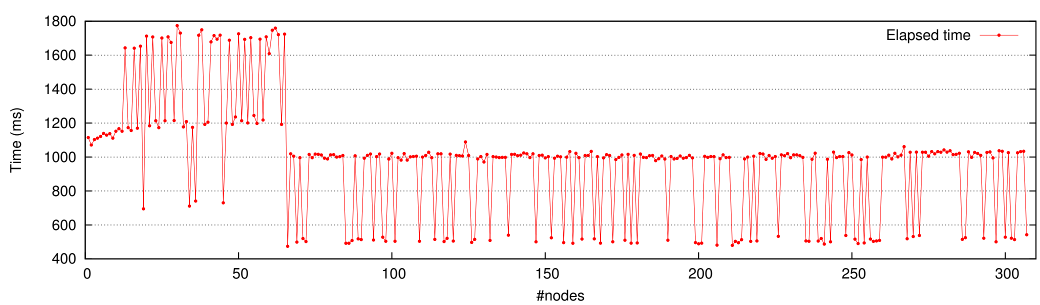

Fig. 6. Elapsed time for sequential registrations of Free-Pastry services

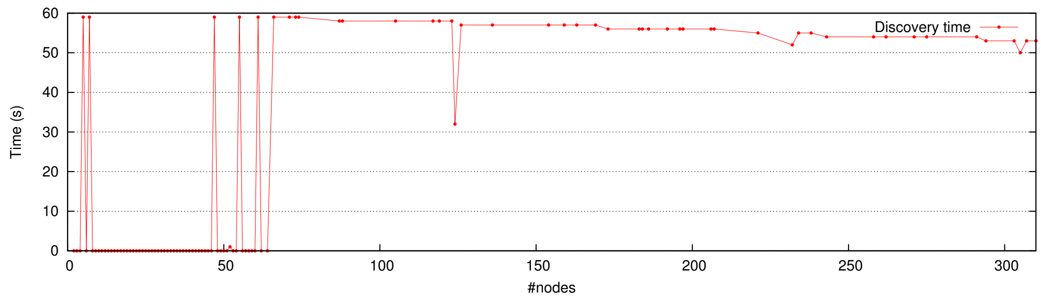

Fig. 7. Browse services after a simultaneous registration

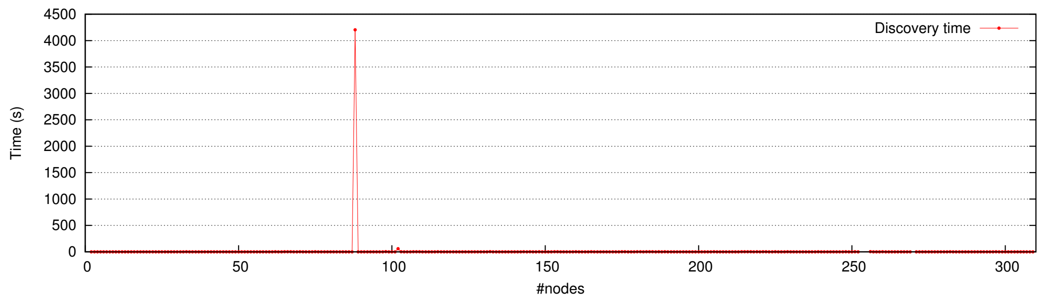

Fig. 8. Browse services after a sequential registrations

On the other hand, as shown in figure 6, the sequential test demonstrates when activating just one request at each minute, we obtain better time registration. Besides, the elapsed time for the registration is almost stable and varies since the 60^{th} service from 500 to 1000 ms. Like Bonjour and Avahi, the 295 registration (one registration on one machine) doses not saturate Free-Pastry.

7 Performance of discovery of services analysis

The second metric is to measure the necessary time to browse a registered service. Then for each system (Bonjour, Avahi and Free-Pastry) we measure the elapsed time between the termination of registration and the discovery time. We repeat the same benchmarks for both simultaneous and sequential registration. For that, we dedicate one machine which runs the browser to discover services.

7.1 Discovery behaviour of Bonjour

Bonjour proves a good performance in discovering services. In fact, it is able to discover 307 services registered on 307 machines (one service on one machine). Furthermore, the discovery time doses not exceed 1 second. We mention that for the sack of simplicity and clearness we did not put the numeric results in this paper for these two tests.

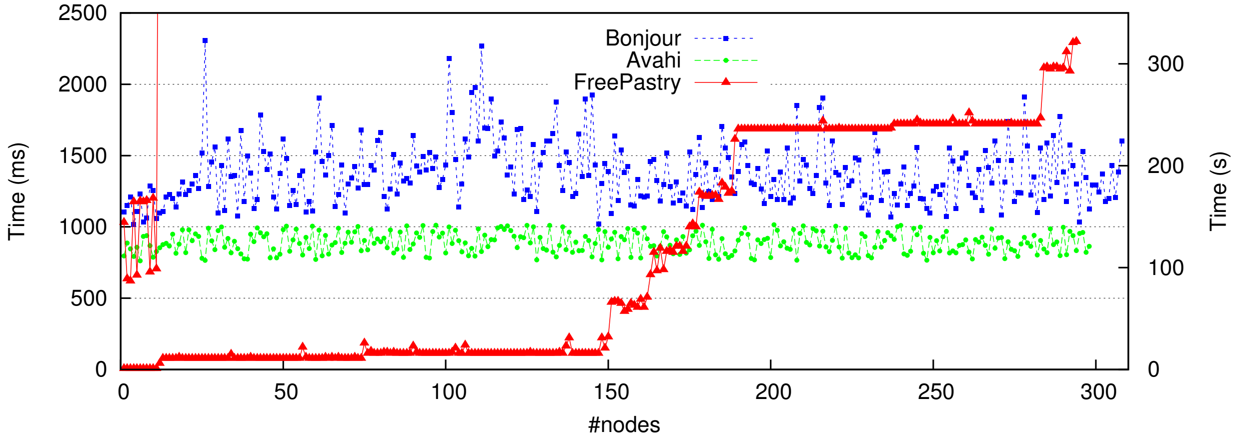

Fig. 9. Elapsed times for simultaneous registrations of services for Bonjour, Avahi and Free-Pastry (note that Free-Pastry is represented twice: once with the left-hand time axis, and a second time with the right-hand time axis, 140 times larger)

7.2 Discovery behaviour of Avahi

Avahi loses almost 60% of registered services. Besides, the discovery time increases beyond 49 registrations to reach 900 s on 73rd. After that, the browse program of avahi spends about 220 seconds to find a registered service (see figure 7).

Conversely, when we browse services after a sequential registration (each minute, we request a registration), Avahi is able to discover more services (303 from 307 registered services). Furthermore, the discovery time is much better (less than 2 seconds for all services). Only for one service did the discovery program spend 4203 seconds to find a service (see figure 8).

7.3 Discovery behaviour for Free-Pastry

Free-Pastry has a good response time of services discovery but it loses several services. In fact, we registered 306 services (one service on one machine) and we launched the browsing program on another machine. The browser can discover the service in at most one second. The experimental tests show that, in sequential registration, the browser finds just 270 from 293 services. Whereas, in simultaneous registrations, the browser finds more services (275 from 292 services).

8 Synthesis and future work

Figure 9 illustrates the difference between the three protocols: Bonjour, Avahi and Free-Pastry in the time elapsed to register simultaneously about 300 services in a Peer-to-Peer Network. Avahi is the best one because it spends the minimum times (900–1000 ms). Bonjour needs more time to register services. However, the difference is not important. Whereas Free-Pastry - which has times near to Avahi up to 150 services – needs distinctly higher times than Avahi and Bonjour as shown in figure 9 (time corresponding to the Free-Pastry curve are given on the secondary axe).

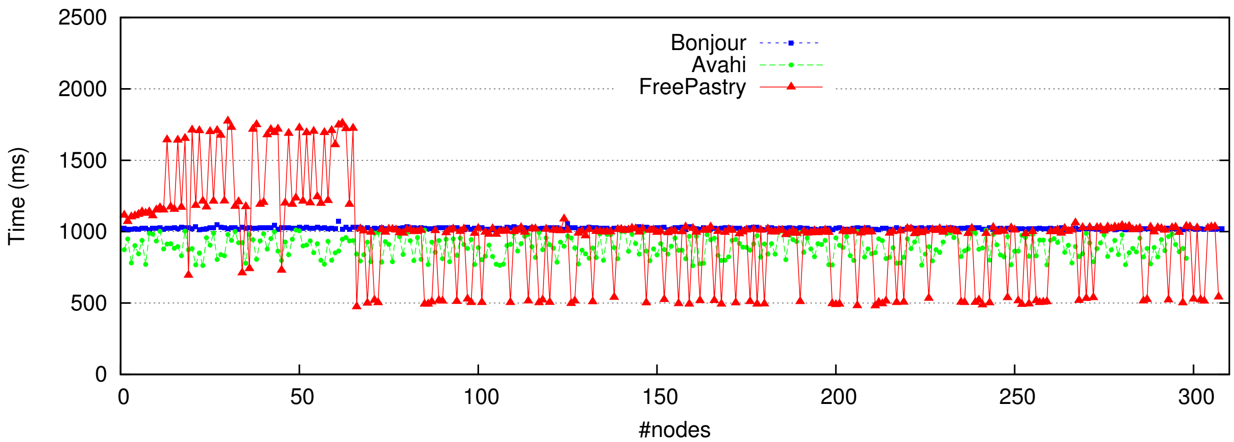

Fig. 10. Elapsed times for sequential registrations of services for Bonjour, Avahi and Free-Pastry

Figure 10 shows the performance of these tools (Bonjour, Avahi and Free-Pastry) when we register sequentially one service on each machine (we attend about 300 machines). We can mention that there is not a big difference between the three protocols. In fact, Bonjour and Avahi give very similar results. Whereas Free-Pastry, need less time to register some services (30% of the total machines), need the same time for others machines (60%) and more time than Avahi and Bonjour for the rest (10%).

The lessons learnt from experiments are the following for our future work that aims to replace the centralised registration service of nodes for a desktop Grid platform such as XtremWeb by a decentralised service:

1. None of the three tools are superior for all the criteria ; it remains to understand (for instance) why Free-Pastry performance decreases drastically for simultaneous registration after 160 registrations and why Avahi loses almost 60% of registered services as soon as we browse more than 75 services which is not a lot. Clearly, Avahi does not scale at present time.
2. A lot of "clock-interval" artifacts can be seen in Free-Pastry and Avahi, especially in figures 6 and 7. These artifacts are due to the event model used in these protocols, where a "watch loop" does periodic polling of resources to be aware of state changes (and thence create notifications).
3. The Bonjour implementation that we have used seems to offer the best compromise. However the Bonjour API is not as rich as Free-Pastry's API. In Free-Pastry we have the PAST module that implements archival storage that could be used, in our context, for storing the properties (CPU Mhz, RAM available...) or data produce on nodes asking for participating to the desktop grid.

Experiments and analyses of P2P networks have been conducted over the Grid'5000 platform for the generic JXTA P2P framework [2]. In this article, the goal of the performed benchmarks is similar to our goal. It concerns to answer common and unanswered questions: how many rendezvous peers are supported by JXTA in a given group and what is the expected time to discover resources in such groups?

Two main protocols of JXTA have been evaluated in [2]: 1) the peerview protocol used to organize super peers, known as rendezvous peers, in a JXTA overlay and 2) the discovery protocol, that relies on the peerview protocol, used to find resources inside a JXTA network. All sites of Grid'5000 were used and a mix of hundreds of rendezvous peers and normal peers, called edge peers, have been deployed on at most 580 nodes. Results show that with default values for parameters of the peerview protocol, the goal of the algorithm is not achieved, even with as few as 45 rendezvous peers. However, parameter tuning makes it possible to reach larger configurations in terms of number of rendezvous peers. For the discovery protocol, authors show that discovery time is rather smaller, provided that all rendezvous peers satisfy a given property. These results give developers a better view of the scalability of JXTA protocols.

Our results, augmented with those of [2] clearly demonstrate that for open source projects as well as for industrial software with production quality, there is a strong need to test and evaluate the properties of the distributed system in real scale platform such as Grid'5000.

9 Conclusion

To launch computing work on the nodes of the grid, it is essential to browse the nodes and then execute works. Furthermore, in the mean time, it is possible that new nodes comes and want to register to the grid. Then it is important to register this new machine in the minimum time possible to be added to the grid.

In this paper, we have illustrated the performance of three P2P systems which are Bonjour, Avahi and Free-Pastry. Several experimental tests were executed on Grid'5000 where the used machines number could go until 308. Bonjour proves a high performance in registration services and is able to browse all registered services (307 services) and, especially, can discover them in at most 1 second each one. Avahi is powerful in registration of new services but it is not able to browse all the services (it loses 60% services in simultaneous registration) and need a long time (4200 seconds) to discover services which have been registered in a sequential manner. On the other hand, Free-Pastry shows good results in the sequential registration of 295 services (at most 1 second needed to register one service on each machine), but it spends more time (300 seconds) to find a services when we activate 295 registration requests of new services at the same time.

Finally, we did not include the results for periodic and real registrations (presented in the above sections 5.4 and 5.5) because there is no important difference in discovering services or registrations to be mentioned. In fact, as illustrated before, the behaviour of discovering or registration depends only of the registrations mode (sequential or simultaneous).

Acknowledgement

Experiments presented in this paper were carried out using the Grid'5000 experimental testbed, an initiative from the French Ministry of Research through the ACI GRID incentive action, INRIA, CNRS and RENATER and other contributing partners (see [23]). We would like to thank Mathieu Jan for his fruitful comment on an early version of this paper.

References

1. Karl Aberer and Manfred Hauswirth. Peer-to-peer information systems: concepts and models, state-of-the-art, and future systems. In *ESEC/FSE-9: Proceedings of the 8th European software engineering conference held jointly with 9th ACM SIGSOFT international symposium on Foundations of software engineering*, pages 326–327, New York, NY, USA, 2001. ACM Press.
2. Gabriel Antoniu, Loïc Cudennec, Mike Duigou, and Mathieu Jan. Performance scalability of the JXTA P2P framework. In *Proc. 21st IEEE International Parallel and Distributed Processing Symposium (IPDPS 2007)*, pages 108–131, Long Beach, CA, USA, March 2007.
3. Miguel Castro, Peter Druschel, Anne-Marie Kermarrec, and Antony Rowstron. One ring to rule them all: service discovery and binding in structured peer-to-peer overlay networks. In *EW10: Proceedings of the 10th workshop on ACM SIGOPS European workshop: beyond the PC*, pages 140–145, New York, NY, USA, 2002. ACM Press.
4. Ian Foster. What is the grid? A three point checklist. *GRIDToday*, pages 1–4, July 2002.
5. Ian Foster and Adriana Iamnitchi. On death, taxes, and the convergence of peer-to-peer and grid computing. In M. Frans Kaashoek and Ion Stoica, editors, *Peer-to-Peer Systems II, Second International Workshop, IPTPS 2003*, volume 2735 of *Lecture Notes in Computer Science*. Springer Verlag, 2003.
6. Ian Foster and Carl Kesselman. The Globus project: a status report. *Future Generation Computer Systems*, 15(5–6):607–621, 1999.
7. Ian Foster, Carl Kesselman, Jeffrey M. Nick, and Steven Tuecke. Grid services for distributed system integration. *Computer*, 35(6):37–46, 2002.
8. Ian Foster, Carl Kesselman, and Steven Tuecke. The anatomy of the grid: Enabling scalable virtual organizations. *Int. J. High Perform. Comput. Appl.*, 15(3):200–222, 2001.
9. Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Schenker. A scalable content-addressable network. In *SIGCOMM '01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 161–172, New York, NY, USA, 2001. ACM Press.
10. Ala Rezmerita, Tangui Morlier, Vincent Néri, and Franck Cappello. Private virtual cluster: Infrastructure and protocol for instant grids. In Wolfgang E. Nagel, Wolfgang V. Walter, and Wolfgang Lehner, editors, *Euro-Par 2006, Parallel Processing, 12th International Euro-Par Conference*, volume 4128 of *Lecture Notes in Computer Science*, pages 393–404. Springer, August 2006.
11. Antony I. T. Rowstron and Peter Druschel. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In *Middleware '01: Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms*, pages 329–350, London, UK, 2001. Springer-Verlag.
12. Daniel Steinberg and Stuart Cheshire. *Zero Configuration Networking: The Definitive Guide*. O'Reilly Media, Inc., first edition, December 2005.
13. Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *SIGCOMM '01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 149–160, New York, NY, USA, 2001. ACM Press.
14. Domenico Talia and Paolo Trunfio. Toward a synergy between P2P and grids. *IEEE Internet Computing*, 7(4):96–95, 2003.
15. Qi Xia, Weinong Wang, and Ruijun Yang. A fully decentralized approach to grid service discovery using self-organized overlay networks. In Peter M. A. Sloot, Alfons G. Hoekstra, Thierry Priol, Alexander Reinefeld, and Marian Bubak, editors, *Advances in Grid Computing - EGC 2005, European Grid Conference*, volume 3470 of *Lecture Notes in Computer Science*, pages 164–172. Springer-Verlag, February 2005.
16. Berkeley Open Infrastructure for Network Computing. URL: `http://boinc.berkeley.edu`.
17. Bonjour. URL: `http://developer.apple.com/networking/bonjour/` .
18. KADeploy. URL: `http://gforge.inria.fr/projects/kadeploy/`.
19. OAR. URL: `http://gforge.inria.fr/projects/oar/`.
20. Seti@home. URL: `http://http://setiathome.berkeley.edu/`.
21. Avahi. URL: `http://www.avahi.org`.
22. FreePastry. URL: `http://www.freepastry.org`.
23. Grid'5000. URL: `http://www.grid5000.fr`.
24. Instant grid. URL: `http://www.lri.fr/~rezmerit/Instant%20Grid.html`.
25. XtremWeb-CH. URL: `http://www.xtremwebch.net`.
26. XtremWeb. URL: `http://www.xtremweb.org`.
27. ZeroConfguration. URL: `http://www.zeroconf.org`.