Distributed Application Reliability on Unstable, Dynamic, P2P-based platforms

Wallace Wadge

Department of Computer Science and AI,
University of Malta

Abstract. Application developers are used to a homogeneous, reliable and easily manageable platform on which to deploy their applications. Increasingly however, the need for a highly scalable distributed environment is becoming prevalent. Just as TCP/IP offers reliability over unreliable links, we aim at providing a simple API to hide an underlying unstable, dynamic, P2P platform and present a consistent, reliable view to the applications requesting it. The API should provide robustness and features found in present Grid-systems, which have similar objectives but numerous inherent and logistical differences. Applications would be able to run in a distributed manner over such a platform.

1 Introduction

It is often desirable to abstract away the software components from the hardware requirements and allow the latter to be dispersed across a cluster of networks. For example, enterprises might wish to outsource a number of services to third-parties while retaining full control. To this end, significant progress has been made via the use and construction of grids [1], a collection of interconnected systems each providing a number of generic services [2].

The grid environment is designed for dedicated and fully co-operative systems integrated to each other via a consistent interface, relatively stable connections and dedicated links. By contrast, a peer-to-peer (P2P) environment is barely cooperative, node failures are common and the resources are very unbalanced and inconsistent. This is unfortunate since a P2P system implies that the users of the service are the ones supporting it's existence.

While we normally locate resources via a globally unique identifier (such as www.cs.um.edu.mt), in P2P systems we are more interested in locating a resource by a physical property or service; for example, find a Windows 2000 machine with 20MB of free RAM and 1.3 GB of disk space or a machine running the apache web server with an ability to execute cgi scripts. The problem is compounded by the fact that, unlike grids, the underlying platform is highly volatile; by the time the attribute search result is returned to the originator of the request, the target machine in question might have been replaced or removed.

Connectivity shifts might not be a problem for typical P2P applications in use today, namely, those dedicated to simple file sharing or task farming. For an application which requires consistency however, the maintenance required to handle all the routine housekeeping might very well result in overheads too large to perform any useful work. For example the overhead traffic grows exponentially in Gnutella [3], a cluster-based P2P application. It is not unusual for 63% of traffic devoted to platform overheads such as resource discovery and keep-alives [4].

Grid technologies rely on the fact that the deployment of grid systems are motivated by professional communities devoted to providing a degree of trust, accountability and sanctions for problem nodes.

In P2P systems by contrast, individuals have very little incentive for honesty, anonymity is sought and resource sharing is a hit-or-miss affair. Indeed, it is common to have a large percentage of users who use up valuable resources and offer nothing in return. Wilcox [5] showed that the typical P2P node remained connected for just 28% of the time and over 70% contributed nothing in return. It has also been shown that people also deliberately misreport their resources to minimise their chance of contributing to the grid. The end result is that a P2P platform is a hostile environment to work with and presents numerous challenges at a number of stages.

2 Proposal

We propose to create a middleware platform whereby applications can seamlessly offload their parallel-based computations across the network in a smooth manner.

Under this proposal, each P2P node advertises its services to its peers. Thereafter, a node can, via a defined protocol (possibly abstracted by the provided library), perform the following tasks:

- Accept a new task
- Accept a task and delegate it to subnodes staying completely out of the loop
- Accept a task but divide the workload
- Add new functionality to the P2P module
- Perform operations in a sandboxed environment.

For example, a P2P node client wishing to perform a complex, time-consuming task can opt to distribute the workload across a cluster of similar P2P nodes. Rather than utilising the P2P platform as a simple file-sharing application, it sends along a program which can be executed by one or more clients — ideally in a sandboxed environment. For instance, suppose we want to employ the quicksort algorithm on a huge set of data possibly scattered around the nodes itself via another module. Therefore from the point of view of the requester we would have:

1. A file pointer or equivalent. This file pointer would just be a way for the requester to indicate the data it is to operate on. This data may also not be physically present on the same machine, the underlying platform would handle this transparently.

2. Now the host would issue the recursive quicksort. Therefore, instead of starting to sort itself and just fetching in the data, it will distribute a sorting code fragment to its immediate peers to recursively sort their parts and instruct each node to return the result already pre-sorted, leaving the original requester to simply merge the parts.

3. Each one of its peers can opt to delegate the given task again.

4. Each peer can then either send the results back to the requester or else to its parent, who will in turn, forward the results. Again the original requester does not see the virtual cluster beneath it, it just sees a way to delegate tasks.

5. At each step the underlying nodes might suddenly fail or new nodes join in; it is up to the platform to hide these problems, for example, by issuing the request again to a new node.

3 Research Venues

Significant effort is currently being undertaken to make grids as flexible as P2P systems and, at the same time, considerable work is being undertaken to make P2P technologies as robust as Grids. Several issues still need to be addressed however; these include:

Robustness P2P systems are significantly unstable. The bottom layer must be sufficiently robust to try every possible avenue to make a connected network stay connected.

Trust and anonymity This is often perceived by users as an important attribute in any P2P-based platform. This is the main focus of Freenet [6].

Standardization At the moment there are few real P2P or grid standard protocols with the end result that developers tend to re-invent the wheel each time.

Extendibility The platform must not rely on too many assumptions and offer abstractions at each logical layer, for example, by letting the applications themselves define what a service or resource is.

4 Conclusion and future work

The project is still in its infancy and therefore many of the aforementioned features are not yet available. To date, the basic framework is in place, a pluggable interface has been implemented and simple tests are possible. Future papers will present further refinements to this model, highlight any problems encountered during development and present detailed performance results.

References

1. Foster, I. The Grid: A New Infrastructure for 21st Century Science.*Physics Today, 55 (2). 42-47. 2002.*
2. The Physiology of the Grid An Open Grid Services Architecture for Distributed Systems Integration. *Ian Foster, Carl Kesselman, Jeffrey M. Nick, Steven Tuecke.*
3. Adar, E. and Huberman, B.A. Free Riding on Gnutella. *First Monday, 5 (10). 2000.*
4. Early Measurements of a Cluster-based Architecture for P2P Systems. *Balachander Krishnamurthy, Jia Wang, Yinglian Xie*
5. B. Wilcox-O'Hearn. Experiences deploying a large-scale emergent network. *In 1st International Workshop on Peer-to-Peer Systems (IPTPS'02), 2002.*
6. Clarke, I., Sandberg, O., Wiley, B. and Hong, T.W., Freenet: A Distributed Anonymous Information Storage and Retrieval System. *International Workshop on Designing Privacy Enhancing Technologies, Berkeley, CA, USA, 2000, Springer-Verlag.*