Replication-based Fault Tolerance in a Grid Environment

Paul Townend and Jie Xu

School of Computing,
University of Leeds, Leeds, LS2 9JT
pt@comp.leeds.ac.uk jxu@comp.leeds.ac.uk

Abstract

The potential size and complexity of service-based applications suggests that many such applications will be very prone to errors and failures. The e-Demand project at the University of Leeds proposes a replication-based fault tolerance scheme that allows voting to be performed on results of functionally-equivalent services, with the aim of detecting any incorrect results whilst bringing about potential improvements in performance. We present our initial implementation, constructed using Java and Apache Axis, consisting of a "co-ordination service" that locates, receives, and votes upon jobs submitted by a client program. We also detail DSS-Net, the range of computing nodes and web hosting environments that we are hoping to migrate our services to shortly. In the next few months, we plan to increase the functionality in our service to allow for more complex voting schemes to be specified and performed, and allowing more sophisticated and stochastic services to be used with the scheme.

1 INTRODUCTION

The potential size and complexity of service-based applications – together with empirical evidence suggesting that complex asynchronous and interacting systems are very prone to errors and failures – means that we cannot expect such applications to be fault-free, no matter how much effort is invested in traditional methods such as fault avoidance and fault removal [LYU95].

The likelihood of errors occurring may also be exacerbated by the fact that many service-based applications may perform long tasks that will require several or more days of computation. The cost and difficulty of containing and recovering from faults in service-based applications may be higher than that for normal applications, whilst the heterogeneous nature of nodes means that many service-based applications will be functioning in environments where interaction faults are more likely to occur. The dynamic nature of the Grid means that a service-based application must be able to tolerate (and indeed, expect) resource availability to be fluid.

It is therefore necessary to investigate the application of *fault-tolerant* techniques for Grid computing. Fault tolerance is the survival attribute of computer systems; [AVI85] describes the function of fault tolerance

"…to preserve the delivery of expected services despite the presence of fault-caused errors within the system itself. Errors are detected and corrected, and permanent faults are located and removed while the system continues to deliver acceptable service."

In addition to the obvious problems of unreliability (both in the time and value domain), a further cause for concern when considering service-based applications is that in many cases an organisation may send out jobs/use services on machines upon which it cannot place trust; for example, the machines may be outside of its organisational boundaries. A fault tolerance approach may therefore be useful in order to prevent a potentially malicious node from affecting the overall performance of the application.

The e-Demand project at the University of Leeds is developing an architecture for service-based applications that will thus provide protection against both malicious and erroneous services.

2 A FAULT TOLERANT SOLUTION

One of the most attractive features of the Grid is the ability for a client application to either stage (potentially computationally expensive) tasks or invoke services on Grid resources dynamically at run-time. These resources may be autonomous and extremely heterogeneous, but it is fair to say that at least some of the following properties will hold:

- We may not be able to guarantee that a resource is non-malicious (i.e. it will not alter results)
- We may not be able to guarantee reliability

(either of the resource's software or hardware)

- We may not be able to guarantee performance

In order to minimise the problems caused by the above properties, we propose a replication-based fault tolerance scheme. Such a scheme will allow us to perform voting on the results of functionally-equivalent services in order to detect any anomalous results (be they faults or maliciously generated results), whilst allowing us to potentially improve performance time by only requiring the user to wait for a sub-set of services to finish, rather than a single service which could be affected by unforeseen performance problems. Our scheme is shown in Figure 1 and is detailed below.

Firstly, a client application instantiates an "FT-Grid co-ordination" service; although conceptually a single service, in reality one may use any of a number of different fault-tolerant schemes in order to guard against a single point of failure within the system; for example, in Figure 1, we have used the distributed recovery block approach.

The co-ordination service can be located either on the client side or on a remote machine. The client informs the co-ordination service of the resource it wishes to invoke, and passes the service the relevant job to be processed by that service. The co-ordination service then contacts one or more UDDI registries in order to determine the location and number of compatible services available (potentially in negotiation with appropriate metering services).

The co-ordination service then determines which services use, and proceeds to send the replicas of the client's job to these services. The services process the job until they complete (or until they fail and/or leave the system), and return results to the co-ordination service. The co-ordination service waits until a given number of nodes have completed, and then votes on their results.

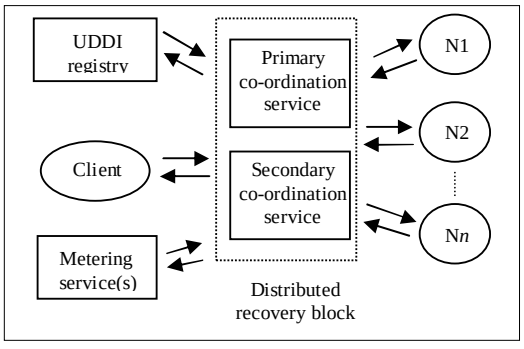

Figure 1: Replication-based fault tolerance

A "correct" result is assumed to be the one returned by the majority (i.e. $n/2 + 1$) of the resources voted upon. Should no majority be reached, then the co-ordinator can either wait for more results to be received, send out more replicas (preferably to services belonging to organisations different from the services whose results did not reach a majority opinion), or return an error message to the client application. When a majority is reached, the result is passed back to the client.

An obvious drawback to this approach is that of increased overhead, as voting cannot commence until a suitable number of results have been generated. This can however be lessened by increasing the number of replicas used; for example, should 3 results be needed in order to perform a satisfactory vote, then 5 replicated jobs can be sent out and voting will begin once the first 3 results have been received. Although traditional replication schemes have assumed that the cost of additional replicas is high, this may not be the case with service-oriented architectures, although cost considerations must be considered (should service use be charged for).

3 IMPLEMENTATION

Our initial FT-Grid implementation – a screenshot of which can be seen in Figure 2 - consists of a client-side Java application that is executed manually by the user. The application allows a user to manually search through any

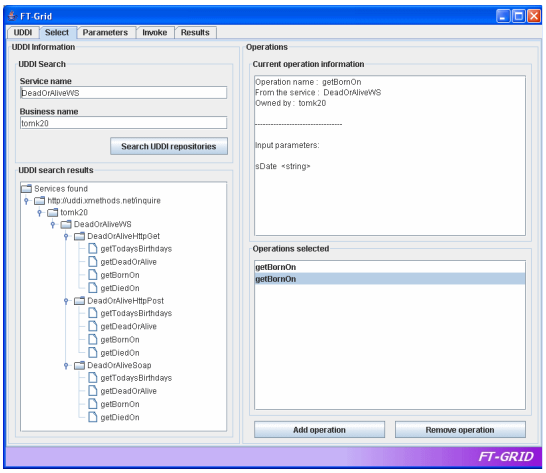

Figure 2: The prototype service

number of public or private UDDI repositories (should more than one UDDI server be specified, then the client will collate and return matching services from all UDDI servers specified), select a number of functionally-equivalent services, choose the parameters to supply to each service, and invoke those services. The application can then

perform voting on the results returned by the services, with the aim of filtering out any anomalous results.

4 DSS-NET

Each node in DSS-Net (Distributed Systems and Services Group Network) is a dual-processor Athlon system, with 1 gigabyte of RAM and a SCSI hard drive. The network is contained within an intranet at the University of Leeds, and is accessible to the outside world through a gateway node. Three nodes in the network run Apache Tomcat servers, with Apache Axis also installed. One of these nodes also runs a jUDDI server, in order to allow us to publish services to a directory. The remaining two nodes run IBM WebSphere, and have IBM UDDI servers running on them.

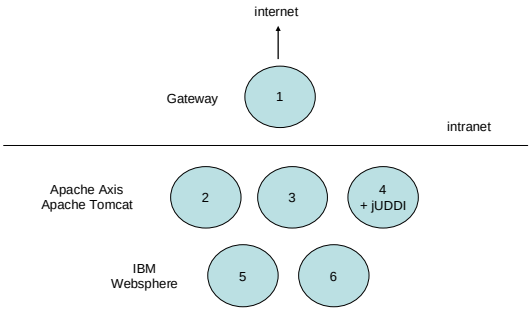

Figure 3: DSS-Net

It is hoped that within the next few months, we will have a range of services deployed across DSS-Net – in addition to the trivial services we have currently deployed to test the network - utilising both web service hosting environments (Tomcat and Websphere) and listed on our UDDI servers. This will then form the basis for more detailed testing of both FT-Grid and the future web services that we develop.

5 FUTURE WORK

The next stage in our work will be to implement an FT-Grid web service, which will provide all the functionality that the Java application currently provides, but which will be able to be utilised programmatically rather than manually. In addition, we will be implementing much more complex voting schemes to allow users greater flexibility when performing voting on the results of services.

6 CONCLUSIONS

Our approach uses an “FT-Grid co-ordination service” to locate, receive, and vote upon jobs submitted by a client program. This is in order to not only reduce the likelihood of faulty results being received by the client, but also to protect against malicious services deliberately altering the results they produce, as well as potentially aiding performance. In addition to this, we show the current progress of our prototype implementation.

In the next few months, we plan to increase the functionality in our FT-Grid service to allow for more complex voting schemes to be specified and performed, allowing more sophisticated and stochastic services to be used with the scheme. We will do this using DSS-Net, a 6 node network of machines located at the University of Leeds, featuring two different web service hosting environments and a UDDI server. FT-Grid will be re-implemented as a web service in order to be able to be invoked dynamically at runtime.

7 REFERENCES

[AVI85] A. Avizienis, “The N-version Approach to Fault-Tolerant Software” - IEEE Transactions on Software Engineering - vol. 11 1985

[LYU95] M.R. Lyu, “Software Fault Tolerance” - John Wiley & Sons - Chichester - UK - 1995