Scalable, Fault-tolerant Management in a Service Oriented Architecture

Harshawardhan Gadgil Geoffrey Fox Shrideep Pallickara
Community Grids Lab, Indiana University
Suite 224, 501 N. Morton St. Bloomington IN 47404
+1 (812) 856-0756 +1 (812) 856-7977 +1 (812) 856-1311
hgadgil@indiana.edu gcf@indiana.edu spallick@indiana.edu

ABSTRACT
The service-oriented architecture has come a long way in solving the problem of reusability of existing software resources. Grid applications today are composed of a large number of loosely coupled services. While this has opened up new avenues for building large, complex applications, it has made the management of the application components a non-trivial task. Management is further complicated when services exist on different platforms, are written in different languages, present in varying administrative domains restricted by firewalls and are susceptible to failure. This paper investigates problems that emerge when there is a need to uniformly manage a set of distributed services. We present a scalable, fault-tolerant management framework. Our empirical evaluation shows that the architecture adds an acceptable number of additional resources making the approach feasible.

Categories and Subject Descriptors
C.4 [**Computer Systems Organization**]: Performance of Systems – Design Studies, Performance Attributes

General Terms
Design, Performance

Keywords
Scalable, Fault-tolerance, Service Oriented Architecture, Web Services

1. INTRODUCTION

This Service Oriented Architecture (SOA) [3] delivers unprecedented flexibility and cost savings by promoting reuse of software components. This has opened new avenues for building large complex distributed applications by loosely coupling interacting software services.

A distributed application benefits from properly managed (configured, deployed and monitored) services. However the various technologies used to deploy, configure, secure, monitor and control distributed services have evolved independently. This complicates application implementation by requiring the use of different proprietary technologies for managing different types of resources. Management is further complicated due to presence of firewalls and network address translation devices that restrict access to resources. As size of application increases in terms of hardware and software components and geographical scale, it is certain that some parts of application will fail. Using a central management system to manage large number of widely distributed resources poses problems related to scalability and vulnerability to a single point of failure of the management framework itself.

These factors motivate the need for a distributed management infrastructure. We envisage a generic management framework that is capable of managing any type of resource. By implementing interoperable management protocols we can effectively integrate existing management systems. Finally the management framework must be scale to manage a large number of distributed resources and also be tolerant to failures within the management framework itself.

Consider a digital entity that can be controlled by zero or modest state that can be exchanged using very few messages. Digital entities can bootstrap and control components such as services with much higher state. We consider the combination of the digital entity and the component as a manageable resource. An example of such a manageable resource is a broker in messaging system [4]. We assume that this digital entity (implicitly or explicitly) has appropriate Web Service interfaces for management purposes.

We limit the scope of management to appropriately configuring and deploying resources / services while maintaining a valid run-time configuration according to some user-defined criteria.

2. ARCHITECTURE

Our approach uses intrinsically robust and scalable management services and relies only on the existence of a reliable, scalable database to store system state comprising of user defined configuration and policies. Resources are wrapped with a Service Adapter to provide the necessary Web Service interface for management purposes.

A hierarchical system comprised of statically configured bootstrap services is used to scale the framework to wide area deployments. These services periodically spawn a health check routine that ensures that the management framework components (Ref. Figure 1) are always up and running. To provide transport independent communication we leverage a publish/subscribe based distributed messaging system such as NaradaBrokering [9]. Further, NaradaBrokering can also be used to enable secure delivery of messages [10].

Finally, management of resources is done by active stateless agents called managers. A resource specific management component is employed to deal with resource specific management operations. We have implemented manager –

HPDC'07, June 25–29, 2007, Monterey, California, USA.
ACM 978-1-59593-673-8/07/0006.

resource interactions using WS-Management. To maintain consistency, the registry is used to generate a monotonically increasing unique instance id for every new instance of registered service. Every message exchange is tagged using a message id that comprises of the sender's instance id and a sequence number. This allows the recipient to detect and ignore duplicate, invalid or obsolete messages.

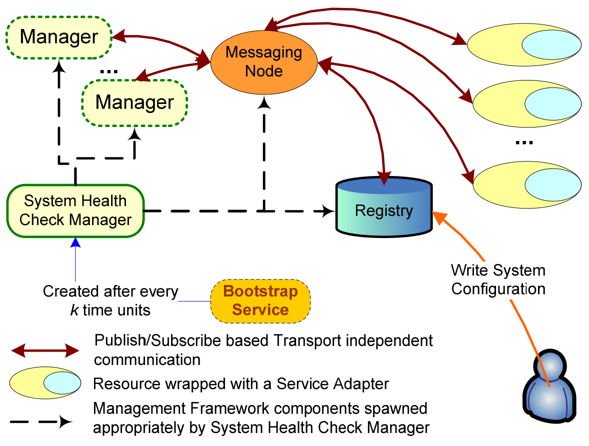
Manager
Manager
...
Messaging Node
...
System Health Check Manager
Registry
Write System Configuration
Created after every k time units
Bootstrap Service
Publish/Subscribe based Transport independent communication
Resource wrapped with a Service Adapter
Management Framework components spawned appropriately by System Health Check Manager

Figure 1 Framework Components

Finally, we compute the percentage of additional infrastructure required to manage N resources. Empirical analysis [5] shows that as the number of manageable resources increases, fault-tolerant management of resources can be achieved by adding about 1% additional resources which correspond to management framework components. This makes the proposed framework feasible.

3. RELATED WORK

The Web Services Resource Framework (WSRF) is a suite of specifications that align the OSGI conceptual model to be in agreement with existing Web standards. The WSRF adopted Web Services Distributed Management (WSDM) for managing state in distributed system. By contrast, we define any service that needs configuration, lifecycle and runtime management as a resource and wrap it with a service interface to expose management capabilities. Management is provided by a complementary specification, WS-Management [1] which is a SOAP-based protocol for managing systems (including Web Services). SNMP (Simple Network Management Protocol) [2] is an application layer protocol that facilitates exchange of management information between network devices. Lack of security features however reduces SNMP to a monitoring facility only. There are a variety of distributed monitoring frameworks such as Ganglia [7], Network Weather Service [11] and MonALISA [8] whose primary purpose is to provide monitoring of global Grid systems and aggregation of metrics. Some systems such as MonALISA also provide the capability of configuring and managing services via RMI. In the Java community, the JMX [6] technology provides tools for building distributed, Web-based management system for managing and monitoring Java applications, devices and service driven networks.

4. CONCLUSION AND FUTURE WORK

We have presented the need and our approach to uniformly manage a set of distributed services. Use of WS-Management helps us to make the management interactions interoperable. The system is tolerant to faults within the management framework while resource failure is handled by implementing user-defined policies. When applied to resources with modest external state, the approach is feasible since it adds about 1% additional resources to provide fault-tolerant management to a large set of distributed resources.

In the future we would like to apply the framework to broader areas that would help carry out more detailed performance benchmarks tests. We believe that application of management framework to such systems can bring up many interesting research issues, specifically challenging scalability of the system.

5. REFERENCES

[1] Arora, A., Cohen, J., Davis, J., Dutch, M. and et.al. Web Services for Management, June 2005.

[2] Case, J., Fedor, M., Schoffstall, M. and Davin, J. A Simple Network Management Protocol (SNMP), 1990.

[3] Channabasavaiah, K., Holley, K. and Edward Tuggle, J. Migrating to a Service Oriented Architecture, Dec 2003.

[4] Gadgil, H., Fox, G., Pallickara, S. and Pierce, M., Managing Grid Messaging Middleware. in *Challenges of Large Applications in Distributed Environments (CLADE)*, (Paris, France, 2006), 83 - 91.

[5] Gadgil, H., Fox, G., Pallickara, S. and Pierce, M. Scalable, Fault-tolerant Management in a Service Oriented Architecture, 2007, CGL Technical Report, http://grids.ucs.indiana.edu/ptliupages/publications.

[6] Kreger, H. Java Management Extensions for application management. *IBM Systems Journal*, *40* (1).

[7] Massie, M., Chun, B. and Culler, D. The Ganglia Distributed Monitoring System: Design, Implementation and Experience. *Parallel Computing*, *30* (7).

[8] Newman, H.B., Legrand, I.C., Glavez, P., Voicu, P. and Cirstoiu, C., MonALISA: A Distributed Monitoring Services Architecture. in *CHEP 2003*, (La Jola, CA, March 2003).

[9] Pallickara, S. and Fox, G., NaradaBrokering: A Middleware Framework and Architecture for Enabling Durable Peer-to-Peer Grids. in *ACM/IFIP/USENIX International Middleware Conference*, (2003).

[10] Pallickara, S., Pierce, M., Gadgil, H., Fox, G., Yan, Y. and Huang, Y., A Framwork for Secure End-to-End Delivery of Messages in Publish / Subscribe Systems. in *7th IEEE/ACM International Conference on Grid Computing (Grid 2006)*, (Barcelona, Spain, 2006).

[11] Wolski, R., Forecasting Network Performance to Support Dynamic Scheduling using the Network Weather Service. in *High Performance Distributed Computing (HPDC)*, (1997), 316 - 325.

Author: Marlon Pierce
Community Grids Lab, Indiana University
+1(812) 856-1212
mpierce@cs.indiana.edu