

GridWeaver: exploring automated configuration and management for Grid computing fabrics

Guillaume Mecheneau (HP Labs), George Beckett (EPCC), Alexander Holt (Sol, The University of Edinburgh), Peter Toft (HP Labs), Kostas Kavoussanakis (EPCC).

INTRODUCTION

As the world deploys larger and more diverse computational resources as components of computing Grids, there is an emerging need to be able to configure these resources:

- *correctly* to avoid errors in complex resource configurations
- *flexibly* so that resources can be rapidly reconfigured for different purposes
- *automatically* to reduce the labour-intensiveness of resource configuration and management, and to speed it up.

Meeting this need has not been a focus of Grid research to date, but we believe it is a vital issue to address as Grid technologies become more widely deployed. This is the focus of the GridWeaver project. The project started in August 2002, and involves collaborators from HP Laboratories in Bristol; the School of Informatics of The University of Edinburgh; and EPCC. The project brings together researchers with a long-standing interest in the problems of correctly and automatically configuring and managing large, complex assemblies of resources, and the applications and services that run on them.

GridWeaver: Approach

Towards automatic configuration of next-generation Grid computing fabrics:

- **Step 1:** Survey the current state-of-the-art in fabric configuration and management
- **Step 2:** Identify the key requirements of next-generation fabrics
- **Step 3:** Suggest approaches for meeting these requirements
- **Step 4:** Refine suggested approaches via proof-of-concept implementations and/or system architecture development

Leverage our expertise and experience from the *LCFG* and *SmartFrog* technologies

NEXT-GENERATION COMPUTING FABRICS

In addition to developing configuration solutions for current Grid infrastructures, GridWeaver aims to anticipate the needs of 'next-generation' computing fabrics. We believe that next-generation fabrics will be characterised by increasing scale, diversity, complexity and dynamism. Furthermore, as Grid computing becomes used for commercial as well as scientific applications, security becomes a higher priority.

Imagine a next generation Grid fabric that includes elements from specialized supercomputers to commodity clusters, with a wide variety of heterogeneous hardware and software, including dedicated resources as well as spare cycles on non-dedicated resources, with permanently connected systems, and intermittently connected systems, and so on. This will give you an idea of the diversity and complexity that must be configured and managed.

'Next Generation' Fabric Management: Some Challenges

- **Scale:** > tens of thousands of systems participating in each fabric
- **Diversity:** highly heterogeneous hardware and software environments; volatile and non-volatile network connections; dedicated and non-dedicated hardware, ...
- **Complexity:** highly complex configuration of individual fabric elements, and of distributed software services executing across multiple fabric elements
- **Dynamism:** the fabric will change continually due to changes in hardware, software and due to failures
- **Validation:** validation is needed to ensure correctly configured fabrics, at all levels from individual hardware elements up to distributed services
- **Security:** Security moves up the priority list, especially for commercial workloads

THE FABRIC CONFIGURATION AND MANAGEMENT SYSTEM MUST DEAL AUTOMATICALLY WITH THESE CHALLENGES

FABRIC CONFIGURATION COMPONENTS

Our basic philosophy is that a successful fabric configuration and management system can be thought of as consisting of at least two parts.

First, a method of describing the desired configuration that we wish the fabric to adopt. This method must be flexible enough to accommodate the description of all elements of interest from low-level hardware resources, through operating systems, up to the distributed services that must be run on the fabric. We are interested in declarative languages that can be used to model the full breadth and complexity of the fabrics we want to configure.

We are also interested in language features that have properties such as:

- validation predicates that allow static checking to determine whether a configuration is permissible or not
- extension of configuration descriptions to allow desired configurations to be easily and cleanly specialized/parameterised for different purposes
- composition of configurations to allow larger system configurations to be composed of several smaller ones—including the ability to combine validation predicates to allow checking of the composite system

Secondly, a method of realising the desired configuration. This is embodied by runtime system components that convert the declarative model of the desired configuration into a correctly configured fabric.

Next-generation fabrics pose several difficult problems:

- they may be so large that it is impossible to have any consistent notion of the 'state' of the fabric. The best we may be able to expect is that the fabric asymptotically converges on the desired configuration—and may never reach it if the desired configuration changes
- the system must accommodate volatile connections (eg laptops), and this is a specialized case of dealing gracefully with fabric and configuration failures.

In addition to realising the desired configuration, we would like the fabric management system to monitor the configuration, notice and accommodate failures, manage reconfiguration (eg updating the version of a running system), and so on.

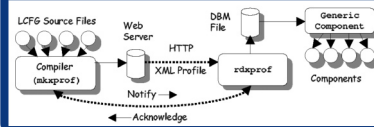
Another desire is to support the runtime checking of validation predicates eg a service level agreement that requires a system to have a response time that falls within a certain range—and to take reconfiguration action if this condition is not met.

Finally, we would like the runtime system to encompass all stages of configuration from hardware configuration, through OS install and ignition, up to deployment and management of complex, distributed services.

In our research, we are drawing up two mature technologies developed by the project collaborators: LCFG and SmartFrog.

LCFG: Local Configuration System

- Developed at Edinburgh University School of Informatics
- Used as a production system to configure and manage hundreds of diverse Linux servers, desktops and laptops
- Based on a declarative specification of the configuration of each system, exported into a configuration database, and runtime components to keep each system synchronized with its specification



ABOUT LCFG

LCFG supports automatic configuration of Unix systems. It represents an entire site's configuration information in a collection of declarative source files. Typically, there is one source file for each main kind of hardware, for each sub-network, for each major pattern of software installation, and so on. Changes to these files are automatically propagated to the machines that they affect.

LCFG is used as the basis for fabric management in the EU DataGrid Testbed 1. The testbed software is large and fairly complicated: getting it up and running on a cluster at each testbed site is not a trivial task. One reason for this is the need to appropriately configure each machine in a cluster—not only so that it works properly as an ordinary computer in that cluster, but also so that the DataGrid software is set up in the right way. Doing this 'by hand' on each computer is barely feasible in a small testbed setting, and is out of the question for clusters of hundreds (or thousands) of machines. What's needed is a mechanism for automating the configuration process. It's important though, that such a mechanism should be flexible, and should not require every machine to be identical, since in practice, both software and hardware varies considerably from machine to machine—and across time.

HP Labs SmartFrog

Smart Framework for Object Groups

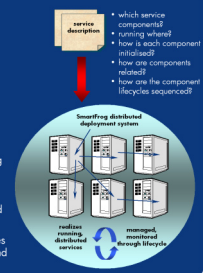
A framework for:

- Describing the configuration of a service
- Deploying and managing the service through its lifecycle

Consisting of:

- A language system for describing service configurations
- A runtime deployment system
- A component model for deployed components

The framework is 'pluggable': provides many useful tools and components, and is easily modified and extended



ABOUT SMARTFROG

SmartFrog is a framework for building 'configuration-driven' systems. It consists of a declarative configuration description language and a runtime deployment system that permit the automatic deployment of distributed applications and services. SmartFrog allows us to express flexibly, as a system configuration description, the system's configuration-specific data, component relationships and sequencing information. The system configuration description is then automatically deployed by the runtime system. This allows systems to be very rapidly reconfigured for different deployment contexts.

SmartFrog technology has been used within products at HP.

CONTACT DETAILS AND ACKNOWLEDGEMENTS

GridWeaver team: Paul Anderson, Carwyn Edwards, Lex Holt (School of Informatics, The University of Edinburgh), George Beckett, Kostas Kavoussanakis (EPCC, The University of Edinburgh), Guillaume Mecheneau, Patrick Goldsack, Peter Toft (HP Labs Bristol).

You can email the team at gridweaver@inf.ed.ac.uk

You can find more about LCFG at <http://www.lcfg.org/>.

GridWeaver is supported by the UK e-Science Core programme, under the project ID 'HPFabMan'.

