GridWeaver[a]: Exploring Automated Configuration and Management for Grid Computing Fabrics

Peter Toft[1], Paul Anderson[2], George Beckett[3], Kostas Kavoussanakis[3], Guillaume Mecheneau[1], Mark Parsons[3,4]
[1]HP Labs, Bristol, UK; [2]Division of Informatics, The University of Edinburgh; [3]EPCC, The University of Edinburgh; [4]National e-Science Centre (NeSC).

Key words to describe the work: large-scale system configuration, service configuration and management, fabric management, grid resource management, configuration validation.

Key Objectives: The GridWeaver project seeks to develop technologies and approaches for describing and automatically deploying system/service configurations across diverse, large-scale, complex sets of resources.

Motivation for the work (problems addressed): This work is motivated by the difficulties of correctly and efficiently configuring resources to participate in Grids. This is a growing concern as we deploy more, and larger, sets of computational resources to meet the needs of e-Science (and, increasingly, commerce).

Introduction

GRID computing binds together computational resources from multiple organisations to facilitate resource sharing and collaborative research. Grid standards and protocols focus on describing the resources available and their current state, and providing mechanisms for accessing the resources in an organised and secure fashion.

As we deploy larger and more diverse computational resources as part of Grids, there is an emerging need to be able to configure these resources:

- correctly – to avoid errors in resource configuration;
- flexibly – so that resources can be rapidly reconfigured for different purposes;
- automatically – to reduce the labour-intensity of resource configuration, and to speed it up.

Meeting this need has not been the focus of Grid research to date, however we believe that solving this problem will become increasingly important as Grid technologies become more widely deployed. This is the focus of the GridWeaver project.

GridWeaver aims to assess the current state of the art for 'fabric management', understand the requirements of current and next-generation fabrics, and develop approaches and technologies that satisfy the requirements. In addition, we are drawing on experience from two bodies of work conducted by researchers on the GridWeaver team:

1. LCFG (Large-Scale Configuration System), developed at the University of Edinburgh, is a system for the automatic configuration of Linux systems.
2. SmartFrog (Smart Framework for Object Groups) is a configuration system developed at HP for the automatic configuration of complex, distributed systems.

The Goal

THE end goal is to develop technologies that are capable of:

- fully describing the 'state' we wish a set of resources to adopt;
- automatically configuring the resources to be in that state;
- managing reconfiguration (including dealing with failure) during the full lifecycle of the set of configured resources.

There are very many challenges to be overcome, so let's offer a sample. First consider the sheer diversity of resources that we must encompass. We must be able to describe and realize the configuration of the full range of low-level resources (compute nodes, storage elements, network elements, scientific instruments). We must accommodate the configuration of compute nodes that exist in a dedicated, tightly-coupled cluster as well as personal machines that are loosely coupled and used for purposes other than participation in Grids. We want to encompass the configuration of higher-level resources such as software elements (e.g., a database), and so on. Furthermore, the

[a] GridWeaver is also known as the 'HPFabMan' project.

description system should be capable of capturing validation rules that specify unambiguously the range of acceptable configuration combinations for a resource to allow static checking.

Secondly, the resource assemblies that participate in Grid computing are of ever-increasing scale. Any solution for fabric management must scale to (at least) tens of thousands of elements.

Thirdly, the collective configuration of resources can be highly complex, including complex linkages and dependencies between resources, and requirements on the correct sequencing of resource lifecycles. The configuration description must be able to capture these dependencies, and the runtime system must be able to realize the correctly configured system in accordance with them.

To bring these ideas to life, consider a set of computational resources that we wish to set up as 'farm' for rendering frames of computer graphics. We first describe the resources that will participate in the farm (which nodes, connected to which storage and networking resources), which operating systems will be installed on each node, and what software will run on each node. We will need to describe relationships and dependencies (e.g., which node is the master scheduler for jobs, and which nodes are the workers), and to check that the complete configuration of the system is correct according to the validation rules.

Our description is then handed to the runtime configuration system, which prepares the low-level resources, handles OS-load and ignition, loads and starts the appropriate software and also manages the configured systems through their lifecycles (including handling failures and reconfiguring appropriately).

We are therefore aiming to cover the spectrum of configuration activities from low-level resource preparation right up to service configuration and activation. In addition, it should be easy to completely reconfigure the set of resources, simply by applying the description of a new configuration.

Steps Towards the Goal

The poster will discuss our current ideas and insights towards the goal described above. In particular we will present:

- a description of the LCFG and SmartFrog systems, and how they can be applied to this goal;
- requirements and a suggested approach for describing the configuration of resources and services;
- initial findings regarding which technologies are currently in use for Grid system/service configuration, and their strengths and weaknesses;
- a discussion of how the configuration system can present an OGSA interface which accepts configuration descriptions that are subsequently realized;
- a discussion of how services that are realized in this manner can present themselves as OGSA services;
- a discussion of an example architecture for automatically deploying a 'render farm' using an initial service description notation and runtime service deployment environment (currently based on SmartFrog).