

GridWeaver: automatic, adaptive, large-scale fabric configuration for Grid Computing

G. Beckett* K. Kavoussanakis* G. Mecheneau† P. Toft†
P. Goldsack† P. Anderson‡ J. Paterson‡ C. Edwards‡

Abstract

The Gridweaver project reviewed the management tools and techniques currently available for Large Scale Configuration. We substantiated that these will not be able to sustain the demands of the Grid services currently in production, unless there is a paradigm shift towards high-level, declarative descriptions of fabric configurations. Our prototype configuration architecture offered this power and allowed our research Grid service, GPrint, to demonstrate ease of maintenance, adaptability to the demands of the fabric and fault tolerance.

This paper introduces system configuration as a major issue to the e-Science community and presents the findings of our research. Interviews of representative large-scale installation managers and our projected case studies for future use of the Grid are distilled to form the requirements of Grid fabrics. We present our prototype architecture, combining the consortium's background approaches to system configuration. We then discuss our GPrint Grid service, which demonstrates the successful investigation of some of the key configuration deficiencies identified by the consortium.

1 Motivation

The emphasis in Grid computing research has predominantly been focused on services. Researchers, hardware manufacturers, and end-users alike dream of the new scientific and commercial world where resources will be readily available and service provision will be regulated and guaranteed. It is not surprising that most of the high-profile, Grid-related work focuses on services and middleware.

In the GridWeaver project, we question the ability of the current infrastructure to respond to the demands of the Grid. We neither doubt the vision, nor underestimate the availability of computing power. What interests us is the *the fabric*, not only the resources and the middleware, but also the glue that keeps these together.

Large installations are not unusual in the modern computing world. The methods of managing them vary widely across organisations, and our research has shown that most depend on a locally generated process, accompanied by heroics from staff (see Section 3). What the Grid adds to the equation is the need for interoperability. Because Grid middleware crosses organisational boundaries, coordination of fabric management will be impractical with the current generation of tools and techniques.

*EPCC, The University of Edinburgh

†HP Labs, Bristol

‡School of Informatics, The University of Edinburgh

2 Existing technologies

The GridWeaver project team conducted a survey [9] to assess and categorise the system configuration technologies that are available.

2.1 System configuration

The term *configuration* is used widely with different meanings in different contexts. We consider the configuration task to encompass everything that is necessary to take a collection of networked computers (with no software) and convert this *bare metal* hardware into a single functioning system, according to a given specification. We also cover the ability to maintain the correspondence between the specification and the actual system, in the event of external changes to the specification, or to the system.

This involves the stages listed below:

Pre-install tasks: Low-level operations that are necessary before software can be loaded from the network (e.g. BIOS configuration).

Software installation: Loading of all necessary software, including the operating system.

Initial configuration: Setting the configuration parameters that differentiate between individual machines running the same software (e.g. the network address, or the services that the particular machine is intended to provide).

Configuration change: If the specification is changed, it is necessary to change the installed software, or the configuration, to reflect the changes in the specification.

Feedback and fault recovery: To maintain the correspondence between the system specification and the actual state, it is necessary to monitor the system for changes due to external events (for example, hardware failure) and to modify the system to correct any discrepancy.

2.2 Automated Configuration

The need for automated configuration tools has been obvious ever since the appearance of networked clusters of workstations. This was initially motivated by the inefficiency of manual configuration and maintenance of large numbers of machines. In a typical cluster intended as a modern Grid resource, the sheer scale makes manual configuration impossible. However, manual configuration also has a number of other serious drawbacks; in particular, it is impossible to guarantee the consistency of individual node configurations, or their conformance to any designated specification.

The following requirements are important considerations for any automatic configuration tool:

- handling large scale installations;
- coping with diversity;
- managing change;
- expressing devolved management;
- balancing explicit specifications versus high-level representation;
- validating consistency;
- fault monitoring and recovery;
- taking control of security;
- representing disconnected operation;
- achieving modularity.

2.3 LCFG and SmartFrog

The School of Informatics at the University of Edinburgh has a strong track record in the development, real-world deployment, and management of large-scale workstation networks. The practical results of this are implemented in the toolkit known as Local Configuration System

(LCFG) [7]. As well as forming the configuration infrastructure for the School, LCFG is currently being used on the testbeds for the European DataGRID [1].

HP Labs' Smart Framework for Object Groups (*SmartFrog*) [17] arose from research into Computing Utilities – a model for computation based on the notion that computing power can be provided on-demand just like conventional utilities (e.g. electricity). The framework defines systems and sub-systems as collections of software components with certain properties. It provides mechanisms for describing these component collections, deploying and instantiating them and then managing them during their entire lifecycle.

While LCFG provides technology for configuration of specific resources, *SmartFrog* offers a high-level language for expressing complex configurations and relationships, and a deployment infrastructure for setting up applications that run across multiple nodes. *GridWeaver* brought the technologies together, as a testbed for exploring large-scale configuration (Section 4).

2.4 Classification of the Technologies

We consider different technologies in terms of an evolution from those that assist with low-level configuration of single nodes, through those that provide more extensive support for collection of nodes, towards technologies that better support diversity and ongoing management of configuration change.

Based on this model, we have categorised the tools into a number of categories.

2.4.1 Low-level installation and configuration

There are several technologies that automate and simplify the process of installing an operating system plus selected application packages onto a blank node, usually accompanied by the ability to do at least some node-specific customisation (setting hostnames, etc.). This customisation includes configuration that is specific to a node's identity, as well as the choice of which application packages to install. Some technologies, (e.g. Solaris JumpStart [13] and Red Hat KickStart [5]) are specific to a particular operating system. Others, such as Rembo [12], support multiple operating systems.

The technologies focus on easily configuring single, discrete nodes where there is a great deal of similarity amongst the nodes (or where there is a fairly small number of different node types),

and therefore where there can be lots of reuse of the configurations.

This class of technology provides no real abstraction mechanism that allows multiple nodes to be thought of as a single system.

2.4.2 Consolidation of multiple nodes

For a collection of similar nodes, we want to have to make configuration changes only once and then have them propagate to all nodes for which the change is appropriate. This has the advantage of reducing the workload and complexity, plus also minimising the opportunities for making configuration errors.

NCAPI Rocks [15] and Rembo [12] provide a central configuration database and image repository that allows a set of machines to be managed centrally, assuming that there is limited diversity across the managed nodes.

In general, these technologies fail to support ongoing management of configuration changes over time.

2.4.3 Managing changing configurations

System configurations change constantly. Application packages appear, bug fixes are released, users are added, nodes are re-purposed, and so on. For many installation contexts, we want a configuration system that is designed with support for such constant changes at its heart.

An approach adopted in many large-scale installations is to complement the methods discussed above with a technology such as Cfengine [4] or SUE [14], to manage ongoing changes to configuration parameters. A more integrated approach is offered by a technology such as LCFG.

However, for these technologies the whole process remains more or less static, in the sense that a centrally held configuration description is enforced onto managed nodes¹.

2.4.4 Doing it all: integrated packages

Technologies such as Tivoli Configuration and Operations Manager [3] and Novell ZENworks [6] are representative of comprehensive commercial offerings. These technologies are capable of configuring a node from its bare-metal state, including application packages, and extend to managing a wide-spectrum of configuration changes over the life-cycle of the node.

¹The latest version of LCFG is experimenting with new facilities, such as fabric-wide queries and handling of dynamic parameters.

These technologies differ in philosophy from LCFG by taking a less abstract and declarative view of the nodes being configured. While supporting broad functionality, they fail to abstract the definition of the fabric, viewing and manipulating configuration in a node-centric manner. As with low-level installation tools, these technologies are best suited to environments in which diversity of node configuration is limited.

2.4.5 Virtualisation: flexible infrastructure provisioning

The HP Utility Data Center (UDC) represents an unusual perspective on automated system configuration, focusing on *virtualisation*. Assemblies of servers, network components and storage components that are physically networked together are composed into virtual server farms under the control of management software. A server farm can be configured easily by connecting the required server nodes to a VLAN, and connecting each server to a selected disk image (and hence operating system and software configuration) in a storage array.

The UDC approach allows virtual farms to be created, modified and removed quickly under software control. Its focus on security is also of note. In its present form, however, only specific hardware configurations are supported.

2.4.6 Automated distributed applications

One missing capability in the technologies discussed so far is the ability to configure systems where services run across a set of nodes, and where there are complex dependencies across the service components running on each node. For example, a web-service may consist of a database tier, an application logic tier and a web-server tier, each of which may be spread over multiple nodes. To realise the service correctly, the service components must be started in the right sequence, and then bound together appropriately. To ensure continuous operation, each service component must be monitored, and appropriate reconfiguration action taken if a component fails.

SmartFrog focuses on this higher level of the configuration task. It is not used for operating system installation and node ignition, but for application/service configuration and ignition. SmartFrog is able to describe applications as configurations of software components that run across a set of distributed nodes. It realises these descriptions by bringing up the software compo-

nents in the correct sequence, managing them as collections with pre-defined semantics through their lifecycles. Because most interesting configuration properties are consolidated into SmartFrog descriptions, it is also very easy to reconfigure applications to run differently – for example, to increase the number of web-servers, or to migrate a service onto a more powerful node.

SmartFrog is distinct from the other technologies considered here, in that it can regard a software system as a single entity, even though it runs over multiple, distributed nodes.

3 Experiences and challenges

In [16], we identify the fabric requirements of the emerging Grid world.

3.1 Real case studies

We present a selection of case studies that illustrate how different organisations manage large-scale system fabrics. The studies are taken from a carefully selected set of institutions, chosen in light of their fundamental reliance on effective configuration management. The examples cover computational clusters, research and development centres, and academic departments. They provide a representative sample of the experiences from the more general community.

It is evident that the configuration strategy of each institution has evolved in an organic manner to tackle problems as they arise. It includes third-party software and home-grown solutions, as well as the technical expertise of administrative staff.

The evidence presented in the various case studies demonstrates the following deficiencies in current configuration technologies:

- a lack of an abstract representation of pan-nodal relationships and dependencies;
- an absence of an automatic sequencing mechanism, that can resolve complex dependencies during the validation of a description and during its deployment;
- an absence of a sufficiently mature, fine-grain representation environment allowing devolution of authority and responsibility between multiple operational staff;
- a lack of support for mobile users – this includes users who regularly work at different terminals, and users who connect to the fabric from laptop computers;

- a critical dependence on the availability of existing, knowledgeable administrative staff.

We infer that these deficiencies lead to undesirable and debilitating restrictions on a user's freedom, such as: (i) inability to guarantee the correctness of a configuration, leading to an increased rate of configuration rot; (ii) limits on the availability/support of hardware, operating environment, and software products; (iii) constraints on the diversity and dynamism of the configuration of localities within the fabric; and (iv) lack of support for transient connections.

3.2 Projected use cases

The report [16] also showcases some projected examples that aim to predict the way in which large-scale system configuration will need to evolve in order to meet the challenges of tomorrow's fabrics. We consider four realistic examples: a large academic department; a web service provider; a large-scale rendering facility; an OGSA-compliant Grid computing application.

We focused on the following influences:

- Increased expectations of users and administrators with regard to dynamism and responsiveness of configuration;
- Expansion of hardware/software diversity;
- An increase in scale;
- The emergence of complex, multi-site installations with overlapping trust domains, and consequent issues of security and reliability;
- The physical (and conceptual) gap introduced between the user and the fabric in the case of utility computing;
- Changes to the scope of user control over scheduling and availability as introduced by utility computing.

The projected use cases highlight a number of fundamental issues relating to automated configuration:

The anticipated expansion in diversity and dynamism makes it impractical for administrative staff to explicitly configure and manage the whole fabric.

Within a multi-level model, conflicts are difficult to identify and an invalid configuration may not be apparent until late on in the compilation

process. Furthermore, it quickly becomes apparent that some conflicts are inevitable. The deployment engine must establish some automatic resolution for these. To facilitate problem rectification and debugging, each individual contribution to the overall description must be authenticated to its author and the origin of attribute values should be traceable back to the source.

The security and stability of the fabric must not be compromised when overlapping configuration descriptions are combined.

We also need to be able to seamlessly change the configuration of one aspect of the fabric in a non-disruptive manner. To do this, one needs a configuration technology that offers versioning, without violating the predicates that are built into the governing model or creating an invalid configuration at any point during the migration.

For large installations, failures become an everyday occurrence; this should be considered during the modelling process. It is important that the user be able to express concepts such as probability of failure and mean response times, and also reason within the description based on these concepts.

3.3 A paradigm shift

The drive towards Grid-based computing is one leading indicator that computing infrastructure is in the initial phases of its next paradigm shift, where a new, over-arching computing model replaces the existing. The studies above show three major influences: hardware proliferation; complexity of demands from the infrastructure; and linear scaling of the labour-to-fabric size ratio [16].

The paradigm shift is one of raising the abstraction level at which we manage our computing resources. Under the new paradigm, we no longer think of individual nodes within computing fabrics; we think at least at the level of collections of nodes (and other infrastructure components), which are configured and managed as single entities. The ultimate goal of our research is to provide fabric configuration and management solutions that are capable of handling the full complexity and diversity found in emerging computing fabrics, and which are able to automate many node-centric activities that would today need to be performed by people.

4 Technological approaches

We turn our attention to how configurations are deployed, i.e. how we move from the descrip-

tion of a desired configuration into its realisation on the fabric. SmartFrog and LCFG represent complementary solutions, covering different aspects of fabric configuration. We thus developed a testbed [8] environment that combines the LCFG and SmartFrog technologies. Specifically, we exploit the following key benefits of the two systems:

- SmartFrog encourages pan-nodal, peer-to-peer interoperability;
- SmartFrog has a mature, versatile, declarative language;
- SmartFrog supports a flexible set of component life-cycles;
- LCFG has a comprehensive resource of components which have been used to manage the configuration of a number of real fabrics;
- LCFG has spawned a wealth of information regarding practical aspects of the implementation of a configuration management infrastructure.

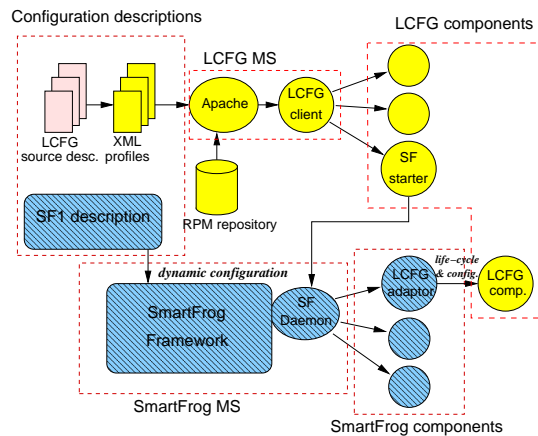


Figure 1: Diagrammatic representation of the architecture for the integration of LCFG and SmartFrog. Elements that have a hashed background relate to SmartFrog; elements that have a uniform shade background relate to LCFG.

Based on the intent to reuse and exploit these features of the two packages, we have designed an integrated configuration environment, presented in Figure 1. Three categories of elements exist within Figure 1: configuration descriptions, management systems (MS), and components.

The configuration descriptions express the intended target configuration of the fabric, either in part or as a whole.

The two management systems co-exist. They each take the configuration description, evaluate it, and generate appropriate instructions, which are then passed to the configuration components.

The configuration components make necessary changes to system files, registries, etc., in order to effect the configuration expressed in the descriptions.

We discern two phases; *static configuration* and *dynamic configuration*.

Static configuration is implemented during the initialisation of the fabric; once deployed, it does not alter. Within our architecture, static configuration is principally handled by the LCFG system.

Dynamic configuration refers to aspects of configuration that may change during the normal operation of the infrastructure. This includes both anticipated changes, and changes in response to a designated combination of events or circumstance. The SmartFrog framework manages all instances of dynamic configuration within the testbed. The dynamic configuration of the installation is implemented in a case-by-case manner. An example of dynamic configuration is provided by the GPrint application (see Section 5).

5 A research prototype

We developed the GPrint prototype Grid service to showcase some of the key concepts of large-scale automatic system configuration [2]. This demonstrator illustrates that we can model, configure, and manage a complex, adaptive system, using our prototype LCFG/SmartFrog architecture (Section 4) as the deployment environment.

The demonstrator has the following key features:

- a declarative configuration, derived from a succinct, modular, multi-level model expressed in the SmartFrog and LCFG languages;
- it allows bare metal OS installation, configuration, and ignition;
- it automatically configures and instantiates an OGSA-enabled environment (Globus Toolkit 3);
- it installs and deploys a demonstration, OGSA-compliant service (GPrint – an adaptive, pan-nodal printing system);

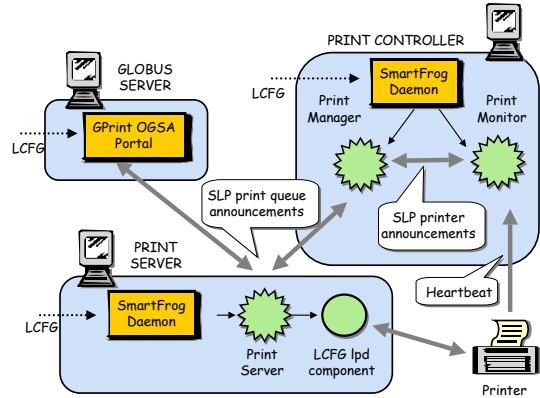


Figure 2: Diagrammatic representation of the architecture of GPrint.

- it automatically responds to events such as the appearance of new resources or the failure of existing elements, maintaining a prescribed level of functionality.

The demonstrator illustrates the following achievements of the project:

Representing multi-level system configuration: We have described the configuration of a printing system, including (at a high level) abstract concepts such as the server-to-print queue-to-printer relationship and (at a low level) real entities, such as configuration of a node. The model presents a view of the system that contains only the elements of configuration which are of interest to the printing administrator, hiding the complexity attributed to other aspects of the fabric.

As an example of this, consider the complexity normally associated with the definition of a user (e.g., disk quota, choice of shell, e-mail aliases, etc.). The majority of these elements are of little consequence to the printing administrator and thus are not present in the extract of the user description which is presented to them. They see only the attributes of a user (e.g. name/group and printing permissions) that have a bearing on the printing model. In a complete configuration solution, these attributes would be augmented to a richer description of a user during the deployment phase.

Deploying system configuration: The complementary nature of SmartFrog and LCFG led naturally to use LCFG to perform low-level configuration actions (especially the basic configuration of nodes) and SmartFrog to configure pan-nodal contributions to the infrastructure. How-

ever, the demonstrator allows to control SmartFrog elements from within LCFG and vice-versa using adaptor technologies, thus offering full flexibility in the decision as to how configuration actions are applied.

From an engineering perspective, a combined system based on LCFG and SmartFrog may not be an ideal solution. However, the separation of responsibility for different stages of the configuration process between the two systems – that is, LCFG deploying a static description for a basic cluster configuration, on top of which SmartFrog establishes and maintains specific pan-nodal services – is a viable proposition.

Weaving a description: A separation exists between the definition of the component elements and the description of the fabric to which they contribute. As part of the demonstrator, we have created prototype descriptions for printers, queues, servers, and so on. These descriptions have been motivated by the requirements and functionality of the LPRng software. Based on these components, we have used the SmartFrog language to weave a reactive, event-driven fabric, which responds to demands for load balancing and reliability in an automatic manner.

In another context, given different demonstrator requirements, it would be equally easy to develop a completely different model but still use the same basic component elements.

Configuring external agents: The demonstrator illustrates how an external agent, such as a printer or a third-party software application, may be assimilated into the configuration environment, using a proxy component. This component encapsulates product-specific configuration attributes and caveats, presenting a simple and consistent interface to the administrator. The use of a proxy allows new technologies to be quickly and easily included in the fabric. For example, printer hardware encapsulates a diverse family of technologies. To manage the configuration of a printer as part of the demonstrator, only a small subset of the configuration parameters needs to be represented. Once a prototype description of a printer has been determined, it is a simple matter to instantiate descriptions of specific printers at runtime, based on core, static data enriched with dynamically discovered information.

Collating information: The demonstrator illustrates the dynamic collation of information from multiple descriptions. Such a construction is similar to the *spanning map* mechanism of

LCFG. Unlike LCFG, contributors to the spanning map do not need to explicitly publish their information.

6 Conclusions

The following overall architecture has been adopted by several configuration tools, including LCFG and others which are particularly focused on the management of entire fabrics.

- A declarative description is maintained for the configuration of the whole fabric. This describes the desired state of the entire fabric, including all configurable devices, and the relationships between them. It does not describe procedures for implementing these configurations.
- Some tool deploys this description by modifying the actual configuration of the nodes to match the specification.
- The same tool, or a different one, is responsible for responding to changes in the fabric or the specification and modifying the node configurations to maintain the correspondence between the actual and desired configuration.

The above general model has many desirable properties, as discussed previously. The GridWeaver project has confirmed that this is an appropriate model for large-scale fabric configuration, but it has also shown that current instances of the model have fundamental problems in both the specification and the deployment areas. A few of the most important of these problems are summarised below (see also [11]):

Specification Descriptions: Current descriptions are too low-level and explicit, allowing no room for autonomic fault recovery, or other dynamic changes. They do not represent the temporal information required to sequence changes reliably and automatically.

Specification Use: Current descriptions do not adequately support the devolved management that is essential for large fabrics, either in terms of authorisation, or in terms of aspect composition. Their associated languages are difficult to understand and use, making them error prone and unsuitable for unskilled staff.

Deployment: Centralised control over all deployment does not scale. Current systems have no uniform support for work-flows which

would allow a smooth change between configuration states. They do not support inter-node transactions on configuration changes, which would allow rollback on failure. Furthermore, they do not provide the ability for localised autonomic reconfiguration under a central policy control. This is necessary both for scaling, and for autonomic fault recovery.

The GridWeaver project investigated some of the above issues in more detail, and provided prototypes for possible solutions. In particular:

- The use of the SmartFrog framework to provide a peer-to-peer layer above LCFG components has demonstrated a system which removes much of the central control and provides a scalable system with autonomic fault-tolerance, and central control over the policy [10].
- The use of the SmartFrog language provides a way to describe LCFG configurations in a clear and high-level way.

7 Recommendations

We believe that it would be possible to implement a new production fabric management system, based on the best of the currently available technology, and augmented by some of the techniques developed in this project. This should provide a significant improvement on existing tools. However, we believe that some of the currently unsolved problems would still render such a system unable to fully meet the requirements of the next generation of Grid fabrics.

We would like to see further fundamental research into a number of areas before considering implementation of a new configuration tool. These areas include:

- Languages for configuration specification, including constraint specification, levels of representation, and support for devolved management of aspects.
- Integration of peer-to-peer technology into the language and deployment architecture.
- Specification of temporal dependencies, and planning and deployment of the workflows necessary to effect smooth changes between configuration states.

References

- [1] The European DataGrid Project. Web page. <http://www.datagrid.cnr.it/>.

- [2] G. Beckett, K. Kavoussanakis, G. Mecheneau, and J. Paterson. design of prototype models for some problems facing large-scale configuration management. Technical report, The GridWeaver Project, July 2003. <http://www.gridweaver.org/WP4/report4.1.pdf>.
- [3] IBM Corporation. Tivoli software from IBM. Web page. <http://www.tivoli.com/>.
- [4] Mark Burgess. Cfengine: a site configuration engine. *USENIX Computing systems*, 8(3), 1995. <http://www.iu.hio.no/%7Emark/papers/paper1.pdf>.
- [5] Martin Hamilton. Red Hat Linux KickStart HOWTO. Web page, January 1999. <http://www.europe.redhat.com/documentation/HOWTO/KickStart-HOWTO.php3>.
- [6] Novell Incorporated. *Novell ZENworks for Desktops 3.2 Administration*, June 2002. Ref. 103-000122-001.
- [7] P. Anderson. *The Complete Guide to LCFG*. University of Edinburgh, 2001. <http://www.lcfg.org/doc/lcfg-guide.pdf>.
- [8] P. Anderson, G. Beckett, K. Kavoussanakis, G. Mecheneau, J. Paterson, and P. Toft. large-scale system configuration with LCFG and SmartFrog. Technical report, The GridWeaver Project, July 2003. <http://www.gridweaver.org/WP3/report3.1.pdf>.
- [9] P. Anderson, G. Beckett, K. Kavoussanakis, G. Mecheneau, and P. Toft. technologies for large-scale configuration management. Technical report, The GridWeaver Project, December 2002. <http://www.gridweaver.org/WP1/report1.pdf>.
- [10] P. Anderson, P. Goldsack, and J. Paterson. SmartFrog meets LCFG - autonomous reconfiguration with central policy control. In *Proceedings of the 2002 Large Installations Systems Administration (LISA) Conference*, Berkeley, CA, 2003. Usenix. To be published.
- [11] P. Anderson and P. Toft. The Gridweaver Project: Summary and Conclusions. Technical report, The GridWeaver Project, July 2003. <http://www.gridweaver.org/WP4/report4.2.pdf>.
- [12] Rembo Technology SaRL. *REMBO: A Complete Pre-OS Remote Management Solution REMBO Toolkit 2.0 Manual*, 2001. <http://www.rembo.com/rembo/docs2/rembo/book.pdf>.
- [13] Sun Microsystems. *Automatic Solaris Installation: A Custom JumpStart Guide*, January 2003.
- [14] CERN. *SUE - Unix Workstation Support*. <http://www.pdp.web.cern.ch/www.pdp/ose/sue/doc/sue.html>.
- [15] Philip M. Papadopoulos, Mason J. Katz, Greg Bruno. NPACI Rocks: Tools and Techniques for Easily Deploying Manageable Linux Clusters. Cluster 2001, October 2001. <http://www.rocksclusters.org/rocks-documentation/papers/clusters2001-rocks.pdf>.
- [16] P. Anderson and G. Beckett and K. Kavoussanakis and G. Mecheneau and J. Paterson and P. Toft. Experiences and Challenges of Large-Scale System Configuration. Technical report, The GridWeaver Project, March 2003. <http://www.gridweaver.org/WP2/report2.pdf>.
- [17] Serano Project, HP Labs, Bristol (UK). SmartFrog - Smart Framework for Object Groups. Web page. <http://www.smartfrog.org/>.