Journal of Physics: Conference Series **331** (2011) 052002 doi:10.1088/1742-6596/331/5/052002

Cluman: Advanced cluster management for the large-scale infrastructures

Journal of Physics: Conference Series

Marian Babik, Ivan Fedorko, David Rodrigues
European Organization for Nuclear Research (CERN), 1211 Geneva 23, Switzerland
E-mail: Marian.Babik@cern.ch, Ivan.Fedorko@cern.ch, David.Rodriguez@cern.ch

Abstract. The recent uptake of multi-core computing has produced a rapid growth of virtualisation and cloud computing services. With the increased use of the many-core processors this trend will likely accelerate and computing centres will be faced with the management of the tens of thousands of the virtual machines. Furthermore, these machines will likely be geographically distributed and need to be allocated on demand. In order to cope with such complexity we have designed and developed an advanced cluster management system that can execute administrative tasks targeting thousands of machines as well as provide an interactive high-density visualisation of the fabrics. The job management subsystem can perform complex tasks while following their progress and output and report aggregated information back to the system administrators. The visualisation subsystem can display tree maps of the infrastructure elements with data and monitoring information, thus providing a very detailed overview of the large clusters at a glance. The initial experience with development and testing of the system will be presented as well as an evaluation of its performance.

1. Introduction

An essential part of automation of data centres is the ability to monitor the facility and take corresponding actions to mitigate any undesirable states. Computing centres, such as the one at CERN, face a challenging task to keep up with the day-to-day operations involving hundreds of configuration changes on thousands of nodes. With the recent uptake of virtualisation and many-core computing this task becomes even more difficult as the number of nodes can triple in matter of months. Creating a decision-support system that can monitor, display and run thousands of administrative jobs is crucial. While there are many existing cluster management systems in place, additional functionality is needed to keep up with the growing infrastructure.

In this article we describe Cluman, an advanced cluster management system, which was designed and developed to improve existing monitoring and cluster management solutions at CERN. The following motivating factors were considered during the course of the project:

- Be able to deal with the rapid increase in the number of nodes and daily reconfigurations of the services.
- Provide possibility to get an overall view of the state of the clusters with improved visualisation methods.
- Enhance the existing administrative job management and provide support for complex reconfiguration with a chain of administrative actions.

- Improve the existing security measures and restrict access to nodes using a role-based authorisation schema.

In particular, Cluman is focused on creating an advanced interactive visualisation tool for fabrics and a job management system supporting workflows for administrative actions. In addition, the system should be scalable to support thousands of nodes and provide appropriate security measures to support the daily work of the service managers. The following features are supported by the system:

- Execute reconfigurations spanning multiple operations on clusters with hundreds of nodes.
- Web-based user interface with role-based authorisation system.
- Interactive visualisation helping service managers get an overview of various monitoring metrics.
- Batch system to run and follow up on the state of reconfigurations (job management system).
- Support for various monitoring and cluster management systems.

The structure of the paper is organised as follows: in Sec. 2 we provide an overview of the two basic and most challenging aspects of the Cluman architecture, i.e. visualisation and job management. A brief description of the actual implementation is given in Sec. 3. In Sec. 4 we present the performance evaluation in a real-life application scenario and we conclude the paper with a brief overview of future and related work.

2. Approach

The architectural approach is based on two distinct components, i.e. visualisation and job management. In the visualisation section we describe an application of the squarified treemap algorithm for cluster visualisation. The job management section presents a Web-based architecture for a stateless distributed middleware that is contacted by a large number of clients concurrently.

2.1. Visualisation

While there are many existing visualisation tools for cluster monitoring data, creating an interactive map with a hierarchical structure of clusters containing hundreds of nodes is not an easy task. Treemaps have been often used for the purpose of tree visualisation since they offer good performance and can use the maximum display space provided [2]. Squarified treemaps tackle the problem of approaching a square-like aspect for the rectangles of the treemap [1]. Although the original problem in general is NP-hard[1], the squarified treemap develops an approximation that can be completed in a short time, thus making the visualisation possible on request.

Since the structure of clusters is often a tree with a large number of leaf nodes, the treemap algorithm can be used to layout the tree structure and then degrade to a simple grid for the leaf nodes. This creates a browsable set of clusters in which each cluster is represented by a rectangle and each node of a cluster by a square. Making the rectangles and squares selectable creates the notion of interactivity, which is needed to select a range of nodes, subclusters or entire clusters.

In addition, the treemap visualisation can be coloured according to the information provided by the monitoring system. This can be accomplished simply by creating either continuos or discrete mappings of the monitoring data to specific colours. This in turn produces a high density image of a cluster as shown in Fig. 1.

[1] http://en.wikipedia.org/wiki/NP-hard

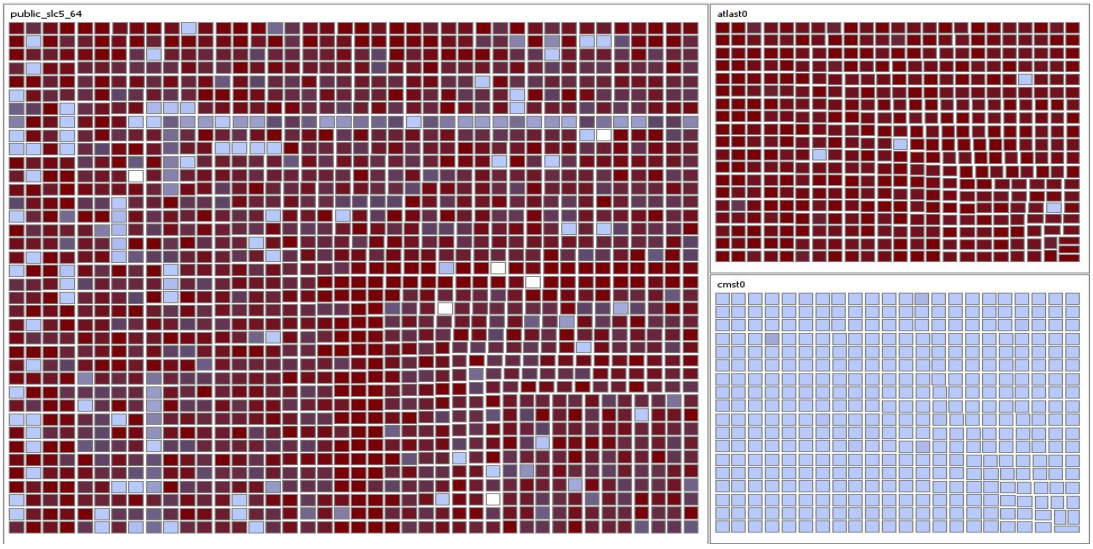

Figure 1. Squarified treemap of a sample cluster with coloured visualisation of the load average.

However, cluster visualisation with treemaps also has certain limitations. One issue is that the actual resolution of the browser window affects how many nodes can be displayed. A possible solution would be to introduce a hybrid grid and treemap approach that can span multiple browser windows. Another issue is the necessity to tune the implementation to a particular Web technology since drawing as many as 10000 squares[2] in a window requires many advanced optimisations.

2.2. Job Management

The aim of the job management is to support the decision-making process of the service managers and enable them to perform administrative actions using existing cluster management systems. This means the ability to support a large number of administrative jobs with different backends together with the possibility to follow their life-cycle and report on any problems that arise during the course of their execution. Since some of the administrative actions can be very complex (e.g. reconfiguration of Castor queues) a set of jobs need to be composed into a workflow.

Conceptually, we have followed the representational state transfer paradigm (REST) to design a network-based stateless event system with cache [3]. With this a large number of clients can be supported as the state of the system is distributed and the middleware only needs to register the state transitions. Administrative job management is proposed as a set of states with well known transitions (scheduled, running, finished, etc.) that are exchanged via a Web-based protocol using a centralised middleware.

This has significant advantages as relying on e.g. the hypertext transfer protocol (HTTP) allows to scale the system with already existing components and infrastructure (HTTP servers, proxy, etc.). In addition the current interface relies on a well established protocol (HTTP), which can be easily extended and implemented with various different technologies that will stay interoperable. Finally, intermediate components can be used to reduce latency or enforce security (proxy), while independent deployment of components is already well established practice in the domain of Web technologies.

Generally, the basic functionality follows the REST paradigm moving through various states

[2] This is also the maximum number achieved with the current implementation at the resolution 1600x1200.

of the system by exchanging different representations of the resources. This implies that there is no state in the system requiring concurrent modifications of the same resource from multiple clients or any state that would need an ordered-access to the resource from multiple clients. Although possible, supporting such requirements would mean implementing various experimental methods in the representation state transfer.

3. Implementation

Cluman's implementation is based on a standard three-tier web architecture. The central component is the middleware with a relational database backend and a set of front-ends. Fig. 2 shows an overview of the core components.

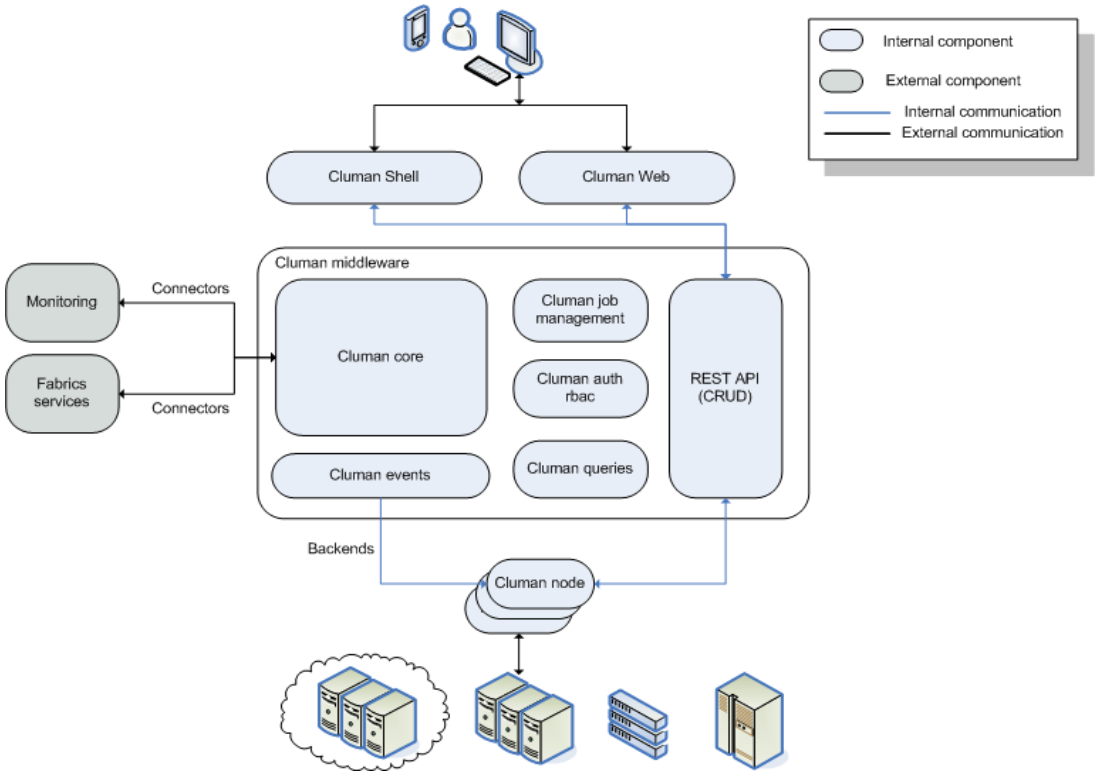

Figure 2. The Cluman network-based three-tier architecture.

The central component is a Cluman middleware providing basic functionality including authentication, authorisation, REST API, job management, visualisation, events and a query system. Front-ends to the system are Cluman shell, Cluman Web and Cluman node, which are essentially just different HTTP clients interfacing with the REST API. Each Cluman client needs to get authenticated in order to access the middleware and be authorised to access a given REST API method. The communication is based on the HTTP protocol with extensive use of all the possible HTTP methods, status codes and resource formats (HTTP, JSON, XML and plain text). The actual implementation is based on the Django Web framework[3] with an Oracle database backend[4]. The command line client and node-client are implemented in Python and the Web based interface is a pure Google Web Toolkit (GWT)[5] application.

[3] www.djangoproject.com
[4] www.oracle.com
[5] code.google.com/webtoolkit/

4. Evaluation

The performance evaluation of Cluman is focused on the scalability of the job submission process, which is determined by the throughput (requests per seconds) of the middleware. Since the middleware is lightweight, the overall throughput is determined by the performance of the database. We have therefore implemented a benchmark and studied what throughput we can achieve with given middleware and database backend. We have also studied different approaches to the database connection pooling as the current implementation of Django does not support a client-based connection pool. However, this is likely to change in the future.

The implemented benchmark simulates a typical workload of the Cluman middleware and tests the database load and various approaches to the database connection pooling. Each benchmark cycle tries to run n jobs concurrently and measures the overall response time of the server. For each job a typical lifecycle is simulated (from scheduled to running to finished with random timeouts set to a maximum of 5 seconds between the running and finished states). Basic runs were performed with 100, 200, 300, 400 and 500 concurrent threads for each possible configuration (for more than 100 threads several machines were used).

The following configurations were tested with the Oracle Instant Client 11.2g and 10.2.0.1:

- Oracle connection manager 11.1g (CMAN)
- Oracle shared server 11.2g (SHS)
- Oracle server side connection pool 11.2g (POOLS)

Each benchmark result reports the number of requests sent to the middleware, the overall response time, the number of requests per second achieved and timing statistics of the HTTP client. In Fig.3 the overall results are presented and it is shown that with a simple deployment, Cluman was able to achieve a typical load of 10 000 administrative jobs running at the same time. This should be sufficient for the current daily operations at CERN's computing centre. Whilst the overall database load remained almost idle, the number of connections that had to be made to the middleware imposed the biggest bottleneck . In terms of connection pool backends, the shared server achieved the highest throughput, but imposed a high load on the middleware, while the server side pool showed the best potential in terms of handling multiple middleware servers.

Benchmark				Response/client			
	Reqs*	Time	Req/s	Avg	Median	Min	Max
CMAN 100	6,951.00	126.05	56.13	0.14	0.08	0.04	2.14
CMAN 200	8,317.00	131.04	63.46	0.79	0.32	0.04	6.61
SHS 100	10,781.00	144.37	75.67	0.07	0.06	0.03	1.05
SHS 200	21,562.00	148.59	145.11	0.13	0.11	0.03	0.74
SHS 300	21,562.00	132.42	162.85	0.23	0.20	0.03	2.16
SHS 400	32,343.00	151.61	213.34	0.40	0.34	0.04	8.81
SHS 500	32,343.00	150.17	215.47	0.66	0.51	0.04	16.74
POOLS 100	10,781.00	145.20	75.24	0.08	0.07	0.03	1.02
POOLS 200	21,562.00	176.65	122.51	0.45	0.34	0.03	1.27
POOLS 300	21,562.00	165.11	130.59	0.78	0.83	0.04	1.82
POOLS 400	32,343.00	210.15	154.01	1.32	1.28	0.24	4.60
POOLS 500	32,343.00	210.40	153.98	1.74	1.78	0.81	10.74
*succesful for all clients							

Figure 3. Evaluation of the REST API with various different Oracle connectors.

5. Related work

There has been an extensive work done on management systems for infrastructures. Commercially available products include RedHat Cluster suites [5], IBM CMS [6], CMAv3 [7], HP Cluster Management Utility [8] and many others. Unlike Cluman they are mostly vendor specific and support only certain hardware configurations and operating systems.

Recently numerous management consoles have been released as part of the virtualisation products and allow a different ways of image-based reconfigurations, where images can be cloned, updated or migrated as a way of changing the machine's state. Again there are many different solution both commercial and open-source, e.g. RedHat Virtualisation, Windows Virtual Control Centre, OpenNebula, VMware and OracleVM are some of the most widely used. While these solutions offer similar functionality to that of Cluman, they are mainly focused on the deployment of virtualisation solutions and many are coupled with only the vendor specific hypervisors.

Finally, a set of open-source projects exist for large-scale clustering that are based on configuration templates, which enable the use of inheritance and composition to build up various complex configurations. Furthermore push or pull-based solutions exist to evolve a machine's state to a particular template. Examples of such systems include Quattor [9], Puppet[6], Chef[7] and CfEngine [10]. Such solutions can complement the functionality of Cluman, as demonstrated by the production instance at CERN, which delegates many management tasks to the Quattor system.

6. Conclusion

We have provided a description of the design and implementation of the large-scale cluster management solution with focus on the transparent extension and integration of the existing solutions. Although many different systems currently exist, there is a limited support for the management across heterogeneous solutions via a web-based interactive console such that provided by Cluman. In the future we would like to focus on integrating virtualisation solutions and extending the performance of the existing system.

References

[1] Bruls, M. and Huizing, K. and van Wijk, J., Squarified Treemaps, Proc. of Joint Eurographics and IEEE TCVG Symp. on Visualization (TCVG 2000), pp. 33-42
[2] B. Johnson and B. Shneiderman. Treemaps: a space-lling approach to the visualization of hierarchical information structures. In Proc. of the 2nd International IEEE Visualization Conference, pages 284291, October 1991.
[3] R. T. Fielding, REST: Architectural Styles and the Design of Network-based Software Architectures, Doctoral dissertation thesis, University of California, Irvine, 2000
[4] R. Fielding and J. Gettys and J. Mogul and H. Frystyk and L. Masinter and P. Leach and T. Berners-Lee, RFC 2616, Hypertext Transfer Protocol – HTTP/1.1, 1999
[5] RedHat Cluster Suite, http://www.redhat.com/cluster_suite
[6] IBM CMS, http://www-03.ibm.com/systems/clusters/software/csm/index.html
[7] CMA3, http://www.streamline-computing.com/index.php?wcId=118&xwcId=
[8] HP Cluster Management Utility, http://h20311.www2.hp.com/HPC/cache/412128-0-0-0-121.html
[9] Poznanski P. and Meli, G. C. and Leiva, R. G. and Cons, L., Quattor - a framework for managing grid-enabled large scale computing fabrics, In Memorias de la XXX Cconferencia Latinoamericana de Inform'atica, pp 777–789, 2004
[10] Burges, M., Recent developments in CFEngine, In Proceedings of the 2nd Unix.nl conference, 2001

[6] http://www.puppetlabs.com/
[7] http://wiki.opscode.com/display/chef/Home