Scalable system software: a component-based approach

2005 J. Phys.: Conf. Ser. 16 546

(http://iopscience.iop.org/1742-6596/16/1/075)

Institute of Physics Publishing Journal of Physics: Conference Series **16** (2005) 546–550
doi:10.1088/1742-6596/16/1/075 SciDAC 2005

Scalable system software: a component-based approach

B Bode[1], R Bradshaw[2], E DeBenedictus[7], N Desai[2], J Duell[3], G A Geist[5], P Hargrove[3], D Jackson[1], S Jackson[6], J Laros[7], C Lowe[4], E Lusk[2], W McLendon[7], J Mugler[5], T Naughton[5], J P Navarro[2], R Oldfield[7], N Pundit[7], S L Scott[5], M Showerman[4], C Steffen[4] and K Walker[6]

[1] Ames National Laboratory, [2] Argonne National Laboratory, [3] Lawrence Berkeley National Laboratory, [4] National Center for Supercomputing Applications, [5] Oak Ridge National Laboratory, [6] Pacific Northwest National Laboratory, [7] Sandia National Laboratory

Email: Stephen L. Scott; scottsl@ornl.gov

Abstract. The growth in computing resources at scientific computing centers has created new challenges for system software. These multi-teraflop systems often exceed the capabilities of the system software and require new approaches to accommodate these large processor counts. The costs associated with development and maintenance of this software are also significant impediments, which are compounded by a lack of interoperability because of site-specific enhancements. The Scalable System Software project seeks to address these issues through a component based approach to system software development. An overview of this design and the benefits of such an approach will be discussed in this paper.

1. Introduction

The nation's premiere scientific computing centers are facing a crisis in which they must rewrite all their home-grown systems software to scale to the multi-teraflops systems that are being installed in their centers. Over the past few years the DOE and others have continued to expand the size of high-end systems. Today there are many systems with more than 1,000 processors and several systems with more than 4,000 processors. This trend is expected to continue in the future with increasing node and processor counts. The first full scale BlueGene/L machine offers a single system with 64 thousand nodes and 128 thousand processors! While this growth has been largely successful on the hardware side, the software needed to manage the growing complexity that comes with larger node counts has continued to lag far behind. In large part the systems management software continues to be based on software developed a decade ago to manage a single system or at most a handful of systems. While many people have worked at patching the software to get it to work on larger systems, there have not been many efforts to design a systems software management infrastructure that is designed for scalability from day one.

The goal of the SciDAC Scalable Systems Software project is to fundamentally change the way future high-end systems software is developed to make it more cost effective and robust [1]. The research involves three efforts. First, collectively getting the DOE centers, NSF centers, and vendors (IBM, Cray, SGI, Intel) to define standardized and flexible interfaces between system components. Second, designing a modular software architecture that is portable across all major supercomputer platforms. Third, producing an open source reference implementation of a fully integrated suite of systems software that can be used across all the terascale computer centers for the cost effective

management and utilization of their computational resources. The scope of our suite encompasses the aspects of system management illustrated in Figure 1.

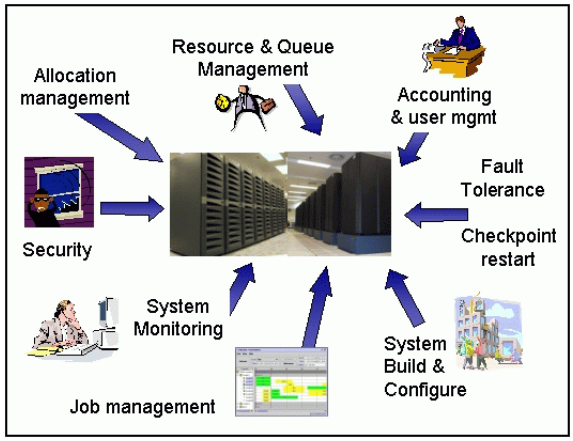

Figure 1. The mission of the Scalable Systems Software center is the development of an integrated suite of systems software and tools for the effective management and utilization of terascale computational resources particularly those at the DOE facilities.

2. Background

The systems software problems for tera-op class computers with thousands of processors are significantly more difficult than for small-scale systems with respect to fault-tolerance, reliability, manageability, and ease of use for systems administrators and users. Layered on top of these are issues of security, heterogeneity and scalability found in today's large computer centers. The computer industry is not motivated to solve these problems because market forces push them towards smaller systems aimed at business uses such as web services, database farms, and departmental sized systems. This matter is underscored as centers prepare for next generation peta-op class computers that will face even greater challenges and require innovative approaches to system software.

The Scalable System Software Center was started in 2001 to address this lack of software for the effective management and utilization of terascale computational resources. This virtual center is comprised of experts from around the country working as a single team to develop an integrated suite of machine independent, scalable systems software components needed for the Scientific Discovery through Advanced Computing (SciDAC) initiative. The goal being to provide open source solutions that work for small as well as large-scale systems.

The SSS project was organized into four different working groups, which worked in a distributed fashion using teleconferences and quarterly face-to-face meetings. These working groups roughly outline the avenues of investigation:

- Node build, configuration, and information services
- Resource management, scheduling, and allocation
- Process management, system monitoring, and checkpointing
- Validation and integration

These working groups were comprised of individuals from DOE labs, NSF Supercomputer Centers and Vendors, i.e., ORNL, ANL, LBNL, PNNL, SNL, LANL, Ames, NCSA, PSC, NCSA, PSC, IBM, Cray, SGI, and Intel.

The first stages of the project focused on the identification of the key components for the system as well as the methods and standards that would be used during the development phase. This initial specification and requirements phase led to the agreement upon the use of XML for interface definitions. Additionally, the use of TCP/IP sockets was to be used for inter-component

communication that would be encapsulated in a library. The focus of the project was to be on the creation of a modular system architecture to facilitate portability and customization across the diverse computing environments that were to be the consumers of the software. The interface definitions would be identified much in the manner that was used during the MPI forums. Additionally, an Open Source reference implementation would also be produced as part of this effort.

3. Project Overview

A critical component of our design is its modularity. The ability to plug and play components is important because of the diversity in both high-end systems and in the individual site management policies. For example, an individual site may have some very specialized requirements for job scheduling that may require them to write their own scheduling component. Figure 2 shows the systems software architecture and how all the components interface with each other. The Event Manager, Service Directory, and Communication library shown in the middle connect to all the other components and allow individual components to announce their participation in the system, find one another, and communicate both synchronously (send/receive) and asynchronously (register/notify). They play a key role in the portability of the architecture to different terascale computing facilities and hardware.

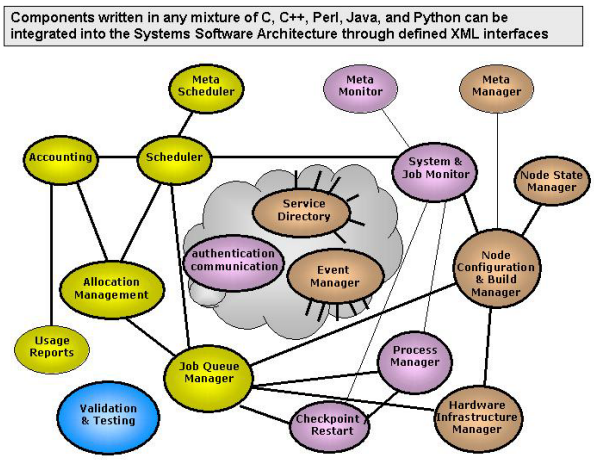

Figure 2. System Software Architecture and component interfaces.

The interfaces between all the components have been fully documented and made publicly available allowing others to write replacement components (or wrap their existing components) as needed. Components communicate through XML messages and can use one or more of the extensible set of wire protocols supplied by the Communication library. The API imposes no restrictions on the language a particular component is written in. Our reference implementation has a mixture of C, C++, Perl, Java, and Python components. This allows a great deal of flexibility to the component author and allows the same interface to work on a wide range of hardware architectures and facility polices.

The reference implementation suite is available in both source form and as a precompiled, integrated package in the form of an OSCAR (Open Source Cluster Application Resources) distribution. The OSCAR based offering, SSS-OSCAR version 1.0, was released at SC2004 [2]. This is being followed by quarterly updates over the next year. OSCAR distributions, of which there are several, have been adopted by many cluster vendors and have been downloaded by thousands all around the world. By leveraging the popularity of OSCAR, we raise our software suites profile and availability.

3.1. SSS-OSCAR

The OSCAR toolkit assists in the configuration, build and installation of a cluster [3]. A modular packaging system is provided, which allows for software to be bundled with a basic configuration and

installed by an end-user without having to have expert knowledge of the tools [4]. This facility was used to deploy a full suite of SSS components using a default configuration [2]. The following sections include a brief description of the components included in the release with information about the actual back-end tools used to implement these components.

3.1.1. Communication Infrastructure: SSSlib
The SSSlib is a communication library that is used by the SSS components for inter-component communication. The library provides a range of default protocols for communication, e.g., HTTP(S), SSL, and has an extensible design. Library bindings exist for several languages including Perl, Python, C and C++. The infrastructure includes a Service Directory and Event Manager for inter-component lookups and communication, respectively [5].

3.1.2. Scheduler: Maui
The widely used Maui scheduling system is configured for the system. It interacts with the system monitor (Warehouse) and queue manager (Bamboo) to schedule jobs across the cluster [6]. If selected for installation, authorization for jobs is acquired from the allocation manager (Gold). This SSS-enabled version of Maui replaces the one typically provided in the standard OSCAR releases.

3.1.3. Queue Manager: Bamboo
The queue manager (Bamboo) receives user submissions and manages the jobs as they are scheduled (Maui) and then run via the process manager (MPD). Bamboo was developed specifically for the SSS project and supports a "PBS like" job submission syntax [6]. It is the gateway between the components responsible for resource and process management.

3.1.4. System Monitor: Warehouse
The monitoring of nodes and their availability for use by the scheduler is controlled through the system monitor. The Warehouse system has been designed for scalable management of the large amounts of data, which are often associated with large computing resources [6]. A central repository houses information acquired from collectors on the compute nodes.

3.1.5. Process Manager: MPD
The startup and management of processes for the system is controlled by the process manager (MPD). The Multi-Purpose Daemon (MPD) provides the backend for the SSS implementation of this component, which is taken from the MPICH distribution [7]. This portion of the system is responsible for the efficient startup and execution of applications on the computing resources as well as the control of I/O, i.e., standard input/output, UNIX signals.

3.1.6. Checkpoint Manager: BLCR
The SSS architecture includes a checkpoint component that can be used to stop running processes and/or restart processes at previously stored instances. A system-level facilities is provided via the Berkeley Labs Checkpoint/Restart (BLCR) system, which enables Linux processes to be checkpointed and later restarted with little to no application knowledge. The SSS-OSCAR release also includes a BLCR-enabled version of LAM/MPI to allow for the checkpointing of MPI jobs across a cluster [8,9].

3.1.7. Allocation Manager: Gold
The accounting and allocation manager (Gold) is responsible for the tracking and granting of resources for users [6]. The Gold implementation of this component interacts with the scheduler (Maui) to enforce these restrictions.

4. Conclusion
The success of this effort can be measured in part by the adoption of the software into the research community and/or industry. The systems software suite has several key features that will make it attractive for others to adopt. First, the suite is modular, which allows others to easily replace a

component that doesn't meet their needs or use only the parts of the suite they need. Second, standard interfaces allow components to be shared across facilities, reducing development and support costs. Finally, the reference systems software is all provided under an open source license to allow others, including commercial interests, to use, modify, and ship it as part of their own offering.

Full system software suites are now in production use for over a year on clusters at Ames laboratory and the Chiba City 200-node cluster at Argonne National Laboratory. Pacific Northwest National Laboratory and the National Center for Supercomputer Applications have adopted one or more components from the suite to use on their production systems. The information gathered from the production use is currently being used as feedback into further developments.

The suite's scheduler component is the widely used Maui Scheduler. The public Maui release (as well as the commercial Moab scheduler) has been updated to use the public XML interfaces and has added new capabilities for fairness, higher system utilization, and improved response time. All new Maui and Moab installations worldwide (more than 3000/month) now use the system software interfaces developed in this project.

Acknowledgements
Research supported by the Mathematics, Information and Computational Sciences Office, Office of Advanced Scientific Computing Research, Office of Science, U. S. Department of Energy, under contract No. DE-AC05-00OR22725 with UT-Battelle, LLC.

References

[1] Al Geist et al. Scalable Systems Software Enabling Technology Center, March 7, 2001. http://www.scidac.org/ScalableSystems/.

[2] John Mugler, Thomas Naughton, and Stephen L. Scott. The Integration of Scalable Systems Software with the OSCAR Clustering Toolkit. In Proceeding of 2nd Annual OSCAR Symposium (OSCAR 2004), Winnipeg, Manitoba Canada, May 16-19 2004.

[3] John Mugler, Thomas Naughton, Stephen L. Scott, Brian Barrett, Andrew Lumsdaine, Jeffrey M. Squyres,Benoit des Ligneris, Francis Giraldeau, and Chokchai Leangsuksun. OSCAR Clusters. In Proceedings of the 5th Annual Ottawa Linux Symposium (OLS'03), Ottawa, Canada, July 23-26, 2003.

Open Cluster Group: OSCAR Working Group. OSCAR: A packaged cluster software for High Performance Computing. http://www.OpenClusterGroup.org/OSCAR.

[4] John Mugler, Thomas Naughton, and Stephen L. Scott. OSCAR Meta-Package System. In Proceeding of 3rd Annual OSCAR Symposium (OSCAR 2005), Guelph, Ontario, Canada, May 15-18 2005.

[5] Narayan Desai, Rick Bradshaw, Ewing Lusk, and Ralph Butler. Component-Based Cluster Systems Software Architecture: A Case Study. Technical Report MCS-P1149-0404, Argonne National Laboratory, April 2005.

[6] Resource Management and Accounting. http://sss.scl.ameslab.gov/home.shtml.

[7] W. Gropp, E. Lusk, N. Doss, and A. Skjellum. A high-performance, portable implementation of the MPI message passing interface standard. Parallel Computing, 22(6):789{828, September 1996.

[8] Berkeley Lab Checkpoint/Restart. http://ftg.lbl.gov/checkpoint.

[9] Sriram Sankaran, Jeffrey M. Squyres, Brian Barrett, Andrew Lumsdaine, Jason Duell, Paul Hargrove, and Eric Roman. The LAM/MPI checkpoint/restart framework: System-initiated checkpointing. In Proceedings LACSI Symposium, Sante Fe, New Mexico, USA, October 2003.