Concepts for High Availability in Scientific High-End Computing [*†]

C. Engelmann[1,2] and S. L. Scott[1]
[1]*Computer Science and Mathematics Division*
Oak Ridge National Laboratory, Oak Ridge, TN 37831, USA
[2]*Department of Computer Science*
The University of Reading, Reading, RG6 6AH, UK
{engelmannc, scottsl}@ornl.gov

Abstract

Scientific high-end computing (HEC) has become an important tool for scientists world-wide to understand problems, such as in nuclear fusion, human genomics and nanotechnology. Every year, new HEC systems emerge on the market with better performance and higher scale. With only very few exceptions, the overall availability of recently installed systems has been lower in comparison to the same deployment phase of their predecessors. In contrast to the experienced loss of availability, the demand for continuous availability has risen dramatically due to the recent trend towards capability computing. In this paper, we analyze the existing deficiencies of current HEC systems and present several high availability concepts to counter the experienced loss of availability and to alleviate the expected impact on next-generation systems. We explain the application of these concepts to current and future HEC systems and list past and ongoing related research. This paper closes with a short summary of the presented work and a brief discussion of future efforts.

1. Introduction

During the last decade, scientific high-end computing (HEC) has become an important tool for scientists world-wide to understand problems, such as in nuclear fusion, human genomics and nanotechnology. Computer simulations of real-world and theoretical experiments using mathematical models have provided us with the advantage to gain scientific knowledge without the need or the capability of performing physical experiments. Furthermore, high-end computing has played a significant role in engineering, where computer simulations have aided the design and testing of machinery, cars, air planes and buildings.

Every year, new HEC systems emerge on the market with better performance and higher scale. For example, the fastest machine (#1 in Top 500 List [22] - IBM ASCI White) in June 2001 had 8192 processors and a maximum performance of 7.2 TFLOPS, while in June 2004 the processor count of the fastest system (IBM Blue Gene/L [4]) reached 65,536 with a performance of 136.8 TFLOPS.

Within three years, the number of processors increased by almost a magnitude. This significant growth of system scale poses a substantial challenge for system software and applications as the reliability of a system decreases with an increase of the number of its components. With only very few exceptions, the availability of recently installed systems has been lower in comparison to the same deployment phase of their predecessors. In some cases the mean time between failure (MTBF) is as low as 40-50 hours.

In contrast to the experienced loss of availability, the demand for continuous availability has risen dramatically. The notion of capability computing is driven by the race for scientific discovery through advanced computing by running applications on the fastest machines available for a significant amount of time (weeks and months) without interruption. The U.S. Department of Energy recently established the National Leadership Computing Facility [17] at Oak Ridge National Laboratory as a national center for capability computing.

[*]Research sponsored in part by the Laboratory Directed Research and Development Program of Oak Ridge National Laboratory, managed by UT-Battelle, LLC for the U. S. Department of Energy under Contract No. De-AC05-00OR22725.

[†]This research is also sponsored in part by the Mathematical, Information, and Computational Sciences Division; Office of Advanced Scientific Computing Research; U.S. Department of Energy. The work was performed at the Oak Ridge National Laboratory, which is managed by UT-Battelle, LLC under Contract No. De-AC05-00OR22725.

In the following sections, we analyze the existing deficiencies of current HEC systems by identifying their individual single points of failure and single points of control. We continue with a presentation of several high availability concepts. We explain their application to current and future HEC systems and list past and ongoing related research. This paper closes with a short summary of the presented work and a brief discussion of future efforts.

2. Existing Deficiencies

High availability deficiencies of any type of system can be categorized into *single points of failure* and *single points of control*.

A failure at a *single point of failure* interrupts the entire system. However, the system is able to continue to run after reconfiguration into a degraded operating mode. Such reconfiguration may involve a full or partial restart of the system.

A failure at a *single point of control* interrupts the entire system and additionally renders the system useless until the failure has been repaired.

Loss of state may occur in case of any failure. For HEC systems we distinguish between *system state* and *application state*. While system state consists of the states of all system services including the OS state itself, application state comprises of the process states of a parallel application including dependent system state, such as communication buffers.

In the following, we describe the individual single points of failure and single points of control of systems for scientific high-end computing. We discuss critical system services and their impact on system availability, and illustrate the role of individual system components (nodes) in more detail.

2.1. Critical System Services

HEC systems run *critical* and *non-critical system services* on head, service and compute nodes, such as job and resource management and communication services (MPI), to allow applications to perform scientific computation in parallel on compute nodes.

A service is *critical* to its system if it cannot operate without it. Any such critical system service is a single point of failure and control for the entire system. As long as one of them is down, the entire system is not available. Critical system services may cause a loss of system and application state in case of a failure.

If a critical system service depends on another service, this other service is an additional point of failure and control for the critical system service and therefore also a critical system service by itself.

Interdependent critical system services do not necessarily reside at the same physical location, i.e. on the same node. Any node and any network connection a critical system service depends on is an additional point of failure and control for the critical system service and therefore also for the entire system.

A service is *non-critical* to its system if it can operate without it in a degraded mode. Any such non-critical system service is still a single point of failure for the entire system. Non-critical system services may also cause a loss of system and application state in case of a failure.

A *system partition service* is critical to its system partition if it cannot operate without it. Any such service is a single point of failure and control for the respective partition it belongs to.

If the system is not capable of operating in a degraded mode, any such critical system partition service is also a critical system service. However, if the system is capable of operating in a degraded mode, any such critical system partition service is also a non-critical system service.

Typical critical system services on systems for scientific high-end computing are: user login, network file system (I/O), job and resource management and communication services (MPI), and in some cases the OS or parts of the OS itself (e.g. for SSI systems).

User management, software management and programming environment are usually non-critical system services, while network file system (I/O) and communication services (MPI) are typical critical system partition services.

2.2. Head Node

If a system has a *head node running critical system services*, this head node is a single point of failure and control for the entire system. As long as it is down, the entire system is not available. A head node failure may cause a loss of system and application state.

A typical head node on systems for scientific high-end computing may run the following critical system services: user login, network file system (I/O) and job and resource management. It may also run the following non-critical services: user management, software management and programming environment.

Head nodes can be found in almost all systems currently in use for scientific high-end computing, including clusters (e.g. IBM MareNostrum [14]), vector machines (e.g. Cray X1 [23]), massively parallel processing (MPP) systems (e.g. Cray XT3 [24]), and single system image (SSI) solutions (e.g. SGI Altix [2]). However, some SSI solutions do not have a head node (e.g. Kerrighed [12]).

2.3. Service Nodes

If a system has *service nodes running critical system services*, any such service node is a single point of failure and control for the entire system. As long as one of them is down, the entire system is not available. Similar to a head node failure, a service node failure may cause a loss of system and application state.

If a system has *service nodes running non-critical system services*, any such service node is a single point of failure for the entire system. A failure of a service node running non-critical system services may still cause a loss of system and application state.

Service nodes typically offload head node system services, i.e. they may run the same critical and non-critical system services.

Service nodes can be found in almost all advanced HEC systems currently in use (e.g. Cray X1, XT3, IBM Blue Gene/L, and MareNostrum) with the exception of some advanced SSI systems.

2.4. Partition Service Nodes

If a system has *partition service nodes running critical system partition services*, any such partition service node is a single point of failure and control for the respective partition it belongs to. As long as any such partition service node is down, the respective partition of the system is not available. Similar to a service node failure, a partition service node failure may cause a loss of system and application state.

If the system is not capable of operating in a degraded mode, any such partition service node is a single point of failure and control for the entire system. As long as any one of them is down, the entire system is not available.

Partition service nodes typically offload critical head/service node system services, but not non-critical system services.

Partition service nodes can be found in more advanced large-scale cluster and MPP systems (e.g. Cray XT3 and IBM Blue Gene/L). Furthermore, federated cluster solutions use head nodes of individual clusters as partition service nodes.

2.5. Compute Nodes

Each *compute node* that is *running critical system services* is a single point of failure and control for the entire system. As long as any such compute node is down, the entire system is not available. A failure of a compute node running critical system services may cause a loss of system and application state.

Each *compute node* that is *not running critical system services* is still a single point of failure for the entire system. If the system is not capable of operating in a degraded mode, any such compute node is a single point of failure and control for the entire system. As long as one of them is down, the entire system is not available. A failure of a compute node not running critical system services may cause a loss of application state, but not necessarily a loss of system state.

Communication services and in some cases the OS or parts of the OS itself are typical critical system services that may run on compute nodes.

Compute nodes that do not run critical system services can be found in almost all systems, with the exception of SSI systems where the OS on the compute nodes itself is a critical service (e.g. SGI Altix and Kerrighed).

2.6. Partition Compute Nodes

Each *partition compute node* that is *running critical system partition services* is a single point of failure and control for the respective partition it belongs to. As long as any such partition compute node is down, the respective partition of the system is not available. Similar to a failure of a compute node, a failure of a partition compute node running critical system services may cause a loss of system and application state.

If the system is not capable of operating in a degraded mode, any partition compute node is a single point of failure and control for the entire system. As long as any one of them is down, the entire system is not available.

Each *partition compute node* that is *not running critical system services* is still a single point of failure for the respective partition it belongs to. If the system is not capable of operating in a degraded mode, any such compute node is a single point of failure and control for the entire system. As long as one of them is down, the entire system is not available. A failure of a partition compute node not running critical system services may cause a loss of application state, but not necessarily a loss of system state.

Partition compute nodes may run the same critical system services that run on normal compute nodes.

Partition compute nodes that do not run critical system services can be found in more advanced large-scale cluster and MPP systems (e.g. Cray XT3 and IBM Blue Gene/L). Partition compute nodes that do run critical system services can be found in federated SSI solutions (e.g. NASAs SGI Altix system Columbia [5]), where each partition is a SSI system.

2.7. System Scale

The mean time to failure of a system shrinks with the number of its dependent (non-redundant) components. Furthermore, mean time to recover of a system grows with the number of its components if the recovery involves the entire system. The more nodes a HEC system consists of, the more frequent is the occurrence of node failures and the more time is needed for a system-wide recovery. If the mean time to failure of a system is shorter than its mean time to recover, it becomes permanently inoperable.

Some of today‘s largest systems (e.g. IBM Blue Gene/L) have over 32,000 nodes and will have over 64,000 nodes in the near future. Research projects, such as the IBM Blue Gene/C [1] development, target the deployment of systems with 1,000,000 nodes within the next decade. Scalable recovery mechanisms are essential for such large-scale systems.

2.8. System Downtime

The availability of a system can be increased by reducing its failure frequency, i.e. by increasing its mean time to failure, but also by reducing its downtime, i.e. by decreasing its mean time to recover.

The mean time to recover of HEC fault-tolerance mechanisms, such as checkpoint/restart and message logging, depends on the system size and the application type. Furthermore, they introduce additional overhead during normal system operation, which needs to be counted as scheduled system downtime.

3. High Availability Concepts

High availability solutions are based on system component redundancy [20]. If a component fails, the system is able to continue to operate using a redundant component. The level of high-availability depends on high availability model and replication strategy.

As a result, the mean time to recover of a system can be significantly decreased, the loss of system and application state can be considerably reduced and single points of failure and control can be eliminated.

There are two distinct high availability models: *active/standby* and *active/active*. While the active/standby model uses for one active component at least one redundant standby component, the active/active model is based on multiple redundant active components.

Component redundancy can be provided in different ways. Systems for scientific high-end computing need to provide at least hardware redundancy (additional nodes). Since the function of a HEC system is to run parallel applications, software redundancy is needed to achieve a higher level of availability.

In the following, we present several high availability concepts and show how they can be applied to HEC systems in order to eliminate individual single points of failure and single points of control.

3.1. Active/Standby

Active/standby high availability follows the failover model. In case of a failure, an idle standby component takes over for the failed component. The level of high-availability depends on the replication strategy for component state.

3.1.1. Active/Cold-Standby. A cold-standby solution provides hardware redundancy, but not software redundancy. In case of a failure, the standby component is automatically initialized and replaces the failed component. However, any component state is lost.

In scientific high-end computing, the active/cold-standby model can be used for all node types in order to reduce the mean time to recover. However, having a spare node without any replication is not a very efficient way of using expensive equipment.

3.1.2. Active/Warm-Standby. A warm-standby solution provides hardware redundancy as well as some software redundancy. State is regularly replicated to the standby. In case of a failure, the standby component replaces the failed component and continues to operate based on the previously replicated state. Only those component state changes are lost that occurred between the last replication and the failure.

Component state is copied using *passive replication*, i.e. in regular intervals or after a state change took place. The standby component needs to ensure that an old replica can only be replaced by a new replica if it has been fully received.

Stateless components are components that do not maintain internal state, but still react to external events, like for example a simple Web server. Warm standby solutions for stateless components do not replicate any state, but provide a seamless failover without the need of standby initialization.

HEC systems may use active/warm-standby solutions for all types of nodes that do run critical system services in order to eliminate the single points of control they represent and to improve the mean time to recover. They still remain single points of failure as some component state is lost in case of a failure.

Service and partition service nodes that do not run critical system services may use warm-standby high availability to avoid the degraded operating mode.

Compute nodes that do not run critical system services may use warm-standby high availability to avoid the degraded operating mode and to improve the mean time to recover. In this case, component state is not replicated to a standby component, but rather to a backup storage to allow any compute node to be replaced by a standby node.

Existing active/warm-standby solutions for scientific high-end computing include: checkpoint/restart mechanisms (e.g. BLCR [3] and diskless checkpointing [7, 19]) and critical system service packages (e.g. HA-OSCAR [11] and SLURM [21])

3.1.3. Active/Hot-Standby.

A hot-standby solution also provides hardware redundancy as well as software redundancy. However, component state is replicated to the standby on any change, i.e. the standby component is always up-to-date. In case of a failure, the standby component replaces the failed component and continues to operate based on the current state.

Component state is copied using *active replication*. A commit protocol is used to announce state changes to the standby component before they are executed at the active component. Once executed, the standby component receives a second message to commit the state change. Any uncommitted state changes are executed by the standby component upon failover.

The active/hot-standby model offers continuous availability without any interruption of service.

Systems for scientific high-end computing may use active/hot-standby solutions for all types of nodes that run critical system services to eliminate the single points of failure and control they represent.

Service and partition service nodes that do not run critical system services may use hot-standby high availability to eliminate the single point of failure they represent and to avoid the degraded operating mode.

Compute nodes that do not run critical system services may use hot-standby high availability to eliminate the single point of failure they represent and to avoid the degraded operating mode. However the operational overhead may be too high at a larger scale.

Existing active/hot-standby solutions for scientific high-end computing include: critical system services (e.g. PBSPro for the Cray XT3 [18]) and message logging facilities (e.g. MPICH-V [16]).

Future work should target complete solutions that include all system services as well as applications.

3.2. Active/Active

Active/active high availability allows more than one redundant system component to be active, while it does not waste system resources by relying on idle standby components. State change requests can be accepted and may be executed by every member of a replicated component group.

3.2.1. Asymmetric Active/Active.

An asymmetric active/active solution provides hardware and software redundancy. However, its software redundancy is not supported by component state replication, but rather by multiple uncoordinated redundant active system components that do not share state. In case of a failure, all other active system components continue to operate.

Since component state is not replicated, components that maintain internal state loose all of their state in case of a failure. While additional hot-standby components may offer continuous availability, state is still not shared between active components and the active component group does not act in a coordinated way.

Asymmetric active/active high availability is very useful for stateless services as it allows a system to seamlessly downgrade into a degraded operating mode. The asymmetric active/active high availability model is typically used in the telecommunication industry.

Critical system services within a HEC system are typically stateful. Previous research showed that asymmetric active/active high availability for HEC systems is possible [13]. However, it is not a recommended model due to its uncoordinated behavior. Further research is needed to determine its application to non-critical HEC system services that may be stateless.

3.2.2. Symmetric Active/Active.

A symmetric active/active solution also provides hardware and software redundancy. Though, component state is replicated within an active system component group using advanced commit protocols, such as *distributed control* [9] and *virtual synchrony* [15]. In case of a failure, all other active system components continue to operate using the current state.

Furthermore, component state is shared in form of *global state*. All components within an active system component group have the same state and are able to change it using mutual exclusive access.

Similar to active/hot-standby, HEC systems may use symmetric active/active solutions for all types of nodes that run critical system services to eliminate the single points of failure and control they represent.

Service and partition service nodes that do not run critical system services may use symmetric active/active high availability to eliminate the single point of failure they represent and to avoid the degraded operating mode.

Compute nodes that do not run critical system services may also use symmetric active/active high availability to eliminate the single point of failure they represent and to avoid the degraded operating mode. However the operational overhead may be too high at a larger scale (probably higher than for active/hot-standby high availability).

Existing solutions for HEC systems include: group communication systems (e.g. Transis [6]) and distributed metacomputing systems (e.g. Harness [10]).

Future research needs to target the transition of existing active/hot-standby solutions to active/active, and the development of complete active/active solutions that include all system services as well as applications.

4. What Next

In this paper, we have analyzed current HEC systems and identified their high availability deficiencies regarding individual single points of failure and single points of control. We have presented of several high availability concepts, explained how they can be applied to current and future HEC systems, and listed related past and ongoing research.

As already mentioned before, the main focus of future research efforts in high availability for scientific high-end computing needs to be on active/hot-standby and active/active solutions that include all system services as well as applications.

There are two major challenges on the way towards active/active high availability for HEC systems.

First, individual critical and non-critical system services need to be identified. This can only be performed on a case by case basis for every single HEC system as the implementation of system services depends on system architecture, vendor and model.

Second, the transition to active/active high availability involves the use of complex commit protocols that are difficult to understand and may involve substantial modification of existing code.

We have began to work on a flexible, pluggable and component-based high availability framework [8] that allows adaptation to system properties and application needs, while providing simple abstraction models for complex commit protocols.

Other ongoing work concentrates on proof-of-concept implementations for active/active HEC system services, such as job management and file system metadata server, in order to provide blueprint solutions for other researchers and for vendors.

References

[1] G. Almasi, C. Cascaval, J. Castanos, M. Denneau, D. Lieber, J. Moreira, and H.S. Warren Jr. Dissecting cyclops: a detailed analysis of a multithreaded architecture. *SIGARCH Comput. Archit. News*, 31(1):26–38, 2003.

[2] Altix Computing Platform at SGI, Mountain View, CA, USA. http://www.sgi.com/products/servers/altix.

[3] Lawrence Berkeley National Laboratory, Berkeley, CA, USA. BLCR Project at http://ftg.lbl.gov/checkpoint.

[4] Blue Gene/L Computing Platform at IBM Research. http://www.research.ibm.com/bluegene.

[5] Columbia SGI Altix Supercluster Computing Platform at NASA. http://www.nas.nasa.gov/About/Projects/Co lumbia/columbia.html.

[6] D. Dolev and D. Malki. The Transis approach to high availability cluster communication. *Communications of the ACM*, 39(4):64–70, 1996.

[7] C. Engelmann and G. A. Geist. A diskless checkpointing algorithm for super-scale architectures applied to the fast fourier transform. *Proceedings of CLADE*, pages 47–52, 2003.

[8] C. Engelmann and S. Scott. High availability for ultra-scale high-end scientific computing. *Proceedings of COSET-2*, 2005.

[9] C. Engelmann, S. L. Scott, and G. A. Geist. Distributed peer-to-peer control in Harness. *Lecture Notes in Computer Science: Proceedings of ICCS 2002*, 2330:720–728, 2002.

[10] G. Geist, J. Kohl, S. Scott, and P. Papadopoulos. HARNESS: Adaptable virtual machine environment for heterogeneous clusters. *Parallel Processing Letters*, 9(2):253–273, 1999.

[11] HA–OSCAR at Louisiana Tech University, Ruston, LA, USA. http://xcr.cenit.latech.edu/ha-oscar.

[12] Kerrighed SSI-Cluster OS at IRISA/INRIA. Rennes, France. http://www.kerrighed.org.

[13] C. Leangsuksun, V. Munganuru, T. Liu, S. Scott, and C. Engelmann. Asymmetric active-active high availability for high-end computing. *Proceedings of COSET-2*, 2005.

[14] MareNostrum eServer Computing Platform at IBM. http://www.ibm.com/servers/eserver/linux/power/mare nostrum.

[15] L. Moser, Y. Amir, P. Melliar-Smith, and D. Agarwal. Extended virtual synchrony. *Proceedings of DCS*, pages 56–65, 1994.

[16] MPICH-V Project at University of Paris South, France. http://www.lri.fr/~gk/MPICH-V.

[17] National Leadership Computing Facility at Oak Ridge National Laboratory. Oak Ridge, TN, USA. http://www.nlcf.gov.

[18] PBSPro Job Management System for the Cray XT3 at Altair Engineering, Inc., Troy, MI, USA. http://www.altair.com/pdf/PBSPro_Cray.pdf.

[19] J. S. Plank, K. Li, and M. A. Puening. Diskless checkpointing. *IEEE Transactions on Parallel and Distributed*

Systems, 9(10):972–986, 1998.

[20] R. I. Resnick. A modern taxonomy of high availability. 1996.

[21] SLURM at Lawrence Livermore National Laboratory, Livermore, CA, USA. http://www.llnl.gov/linux/slurm.

[22] Top 500 List of Supercomputers. http://top500.org.

[23] X1 Computing Platform at Cray, Seattle, WA, USA. http://www.cray.com/products/x1.

[24] XT3 Computing Platform at Cray, Seattle, WA, USA. http://www.cray.com/products/xt3.