Office of Science
U.S. DEPARTMENT OF ENERGY

OAK RIDGE NATIONAL LABORATORY
MANAGED BY UT-BATTELLE FOR THE DEPARTMENT OF ENERGY

Concepts for High Availability in Scientific High-End Computing

Christian Engelmann [1,2] and Stephen L. Scott [1]

[1] Computer Science and Mathematics Division
Oak Ridge National Laboratory, Oak Ridge, USA

[2] Department of Computer Science
The University of Reading, Reading, UK

High Availability and Performance Computing Workshop (HAPCW) 2005
Santa Fe, NM, USA

Research Motivation

- Today's supercomputers typically need to reboot to recover from a single failure.
- Entire systems go down (regularly and unscheduled) for any maintenance or repair (MTBI=40-50h).
- Compute nodes sit idle while their head node or one of their service nodes is down.
- Availability will get worse in the future as the MTBI decreases with growing system size.

➢ *Why do we accept such significant system outages due to failures, maintenance or repair?*

Availability Measured by the Nines

9's	Availability	Downtime/Year	Examples
1	90.0%	36 days, 12 hours	Personal Computers
2	99.0%	87 hours, 36 min	Entry Level Business
3	99.9%	8 hours, 45.6 min	ISPs, Mainstream Business
4	99.99%	52 min, 33.6 sec	Data Centers
5	99.999%	5 min, 15.4 sec	Banking, Medical
6	99.9999%	31.5 seconds	Military Defense

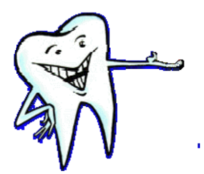
- Enterprise-class hardware + Stable Linux kernel = 5+
- Substandard hardware + Good high availability package = 2-3
- Today's supercomputers = 1-2
- My desktop = 1-2

Research Goals

- Provide high-level RAS capabilities similar to the IT/telecommunication industry (3-4 nines).
- Eliminate many of the numerous single-points of failure and control in today's HEC systems.
- Improve scalability and access to systems and data.
- *Development of techniques to enable HEC systems to run computational jobs 24x7.*
- *Development of proof-of-concept implementations as blueprint for production-type RAS solutions.*

RAS Research for HEC Systems

- We need to analyze current HEC systems and identify their high availability deficiencies.
- We need to show how high availability concepts can be applied to HEC systems.
- Let us take a look at current HEC systems and their high availability deficiencies.



Single Points of Failure and Control

- Single point of failure (SPoF):
  - A failure at a SPoF interrupts an entire system.
  - However, the system is able to continue to run after reconfiguration into a degraded operating mode.
  - Reconfiguration may involve a full or partial restart.
- Single point of control (SPoC):
  - A failure at a SPoC additionally renders an entire system useless until the failure has been repaired.

Single Points of Failure and Control

- A system may have multiple SPoFs and SPoCs.
- A system may consist of multiple subsystems.
- System services and nodes can be single points of failure and single points of control.
- There are two system service classes and several different node types.

Critical System Service

- System cannot operate without it.
- Single point of failure and control.
- Critical system services in HEC systems:
  - User login.
  - Network file system (I/O).
  - Job and resource management.
  - Communication services (MPI).
  - In some cases the OS itself (e.g. for SSI systems).

Non-Critical System Service

- System can operate without it in a degraded mode.
- Single point of failure.
- Non-critical system services in HEC systems:
  - User management.
  - Software management.
  - Programming environment.

Node Types

- Critical and non-critical system services may run on the following node types:
  - Head node.
  - Service nodes and partition service nodes.
  - Compute nodes and partition compute nodes.
- Nodes with critical system services are SPoF and SPoC for the entire system or system partition.
- Nodes with non-critical system services are SPoF for the entire system or system partition.

Loss of State

- Loss of state may occur in case of any failure.
- HEC system state consists of:
  - System state, i.e. system services and OS.
  - Application state, i.e. process states of parallel application including dependent system service state (e.g. MPI buffer).

Additional HEC System Deficiencies

- System scale:
  - MTTF shrinks and MTTR grows with increasing system size (number of dependent, non-redundant components).
  - If the MTTF of a system gets shorter than its MTTR, the system becomes permanently inoperable.
  - Scalable recovery mechanisms are essential for large-scale systems.
- System downtime:
  - MTTR is dominated by fault tolerance mechanisms, such as checkpoint/restart and message logging.
  - Overhead during normal system operation is downtime.

High Availability Concepts

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

➢ MTTR of a system can be significantly decreased.

➢ Loss of state can be considerably reduced.

➢ SPoF and SPoC can be completely eliminated.

High Availability Models

- Active/Standby:
  - For one active component at least one redundant inactive (standby) component.
  - Fail-over model with idle standby component(s).
  - Level of high-availability depends on replication strategy.
- Active/Active:
  - Multiple redundant active components.
  - No wasted system resources.
  - State change requests can be accepted and may be executed by every member of the component group.

Active/Cold-Standby

- Hardware, but not software redundancy.
- Standby component is automatically initialized and replaces the failed component.
- Any component state is lost.
- Can be used for any type of node.
- ➢ Having a spare node without any replication is not a very efficient way of using expensive equipment.

Active/Warm-Standby

- Hardware and software redundancy.
- State is regularly replicated to the standby.
- Standby component automatically 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 intervals or after a state change took place.

Active/Warm-Standby

- Stateless components:
  - Do not maintain internal state, but still react to external events, like for example a simple Web server.
  - Warm-standby solutions do not replicate any state.
  - Seamless failover without the need of standby initialization.
- Warm-standby may be used for any kind of node:
  - To eliminate the single points of control.
  - To avoid the degraded operating mode.
  - To improve the MTTR of a system.
  - However, single points of failure remain.

Active/Warm-Standby

- Warm-standby for compute nodes involves replication to backup storage.
- Examples:
  - Checkpoint/restart mechanisms (e.g. BLCR).
  - Diskless checkpointing.
  - HA-OSCAR.
  - SLURM.

Active/Hot-Standby

- Hardware and software redundancy.
- State is replicated to the standby during change.
- Standby component automatically replaces the failed component and continues to operate based on the current state.
- Component state is copied using *active replication*, i.e. by commit protocols that ensure consistency.
- ➢ Continuous availability without any interruption.

Active/Hot-Standby

- Hot-standby may be used for any kind of node:
  - To eliminate the single points of failure and control.
  - To avoid the degraded operating mode.
- Hot-standby for compute nodes may involve a significant replication overhead.
- Examples:
  - PBSPro for the Cray XT3.
  - MPICH-V message logging facility.

Asymmetric Active/Active

- Hardware and software redundancy.
- However, <u>no component state replication</u>.
- Multiple uncoordinated redundant active system components that do not share state.
- In case of a failure, all other active system components continue to operate.
- Stateful components loose all of their state.
- Additional hot-standby components may offer continuous availability.

Asymmetric Active/Active

- Very useful for stateless components.
- System is able to seamlessly downgrade into a degraded operating mode.
- Typically used in the telecommunication industry.
- HEC system services are typically stateful.
- Previous research showed that AAA is possible.
- ➢ Not recommended due to uncoordinated behavior.

Symmetric Active/Active

- Hardware and software redundancy.
- Component state is *actively replicated* within an active component group using advanced commit protocols (*distributed control*, *virtual synchrony)*.
- All other active system components continue to operate using the current state.
- Component state is shared in form of *global state.*
- Continuous availability without any interruption and without wasting resources.

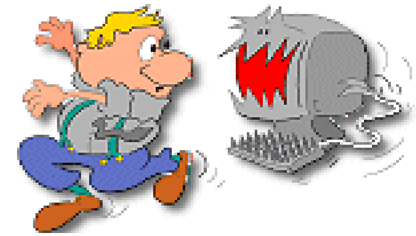
Symmetric Active/Active

- SAA may be used for any kind of node:
  - To eliminate the single points of failure and control.
  - To avoid the degraded operating mode.
- SAA for compute nodes may involve a significant replication overhead.
- Examples:
  - Group communication systems, e.g. Transis.
  - Distributed virtual machines (DVMs), e.g. Harness.
  - Stock market exchange systems.
  - Military: AEGIS battle radar system.

What Next

- We have analyzed current HEC systems and identified their high availability deficiencies.
- We have presented of several HA concepts, explained how they can be applied to HEC systems.
- Main focus of future efforts needs to be on active/hot-standby and active/active solutions that include all system services as well as applications.

Problems Ahead



- 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.
- The transition to active/active involves complex commit protocols that are difficult to understand.
- Substantial modification of existing code may be involved as well.

Office of Science
U.S. DEPARTMENT OF ENERGY

OAK RIDGE NATIONAL LABORATORY

MANAGED BY UT-BATTELLE FOR THE DEPARTMENT OF ENERGY

Concepts for High Availability in Scientific High-End Computing

Christian Engelmann [1,2] and Stephen L. Scott [1]

[1] Computer Science and Mathematics Division
Oak Ridge National Laboratory, Oak Ridge, USA

[2] Department of Computer Science
The University of Reading, Reading, UK

High Availability and Performance Computing Workshop (HAPCW) 2005
Santa Fe, NM, USA