LOUISIANA TECH UNIVERSITY

OAK RIDGE NATIONAL LABORATORY

MANAGED BY UT-BATTELLE FOR THE DEPARTMENT OF ENERGY

CRAY THE SUPERCOMPUTER COMPANY

NORTH CAROLINA STATE UNIVERSITY

# High Availability for Ultra-Scale High-End Scientific Computing

Christian Engelmann

Oak Ridge National Laboratory

Office of Science
U.S. DEPARTMENT OF ENERGY

# Overview

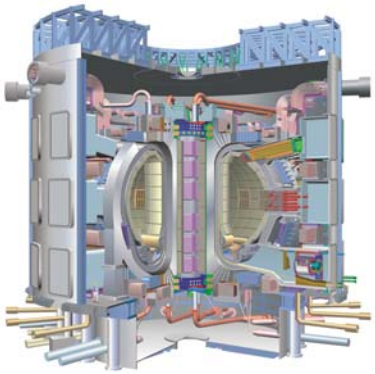
- Background.
  - Computer science at Oak Ridge National Laboratory.
  - Ultra-scale high-end scientific computing.
  - High availability in ultra-scale high-end scientific computing.
- A modular pluggable high availability framework.
  - High availability framework use cases.
  - Framework architecture and design.
  - Implementation issues.
- Related research.
- Future work.

# 1. Background

- Computer science at ORNL.
- Ultra-scale high-end scientific computing.
- High availability in ultra-scale HEC.

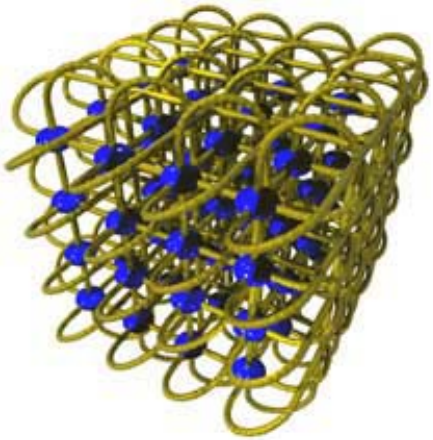
# Oak Ridge National Laboratory

- Multiprogram science and technology laboratory.
- Privately managed for the U.S. Department of Energy.
- Basic and applied research and development.
- In biological, chemical, computational, engineering, environmental, physical, and social sciences.
- Staff: 3800 total, 1500 scientists and engineers
- Budget: $1.06 billion, 75% from DOE.
- Total land area: 58 square miles
- ~3000 guest researchers each year
- ~30,000 visitors each year



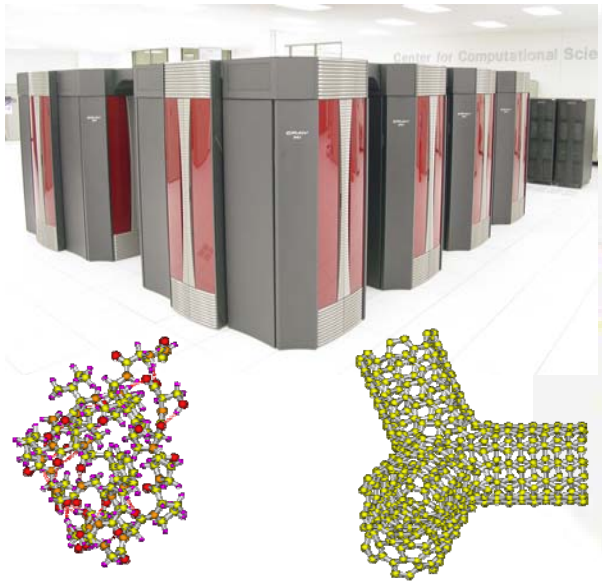
# Computer Science at ORNL

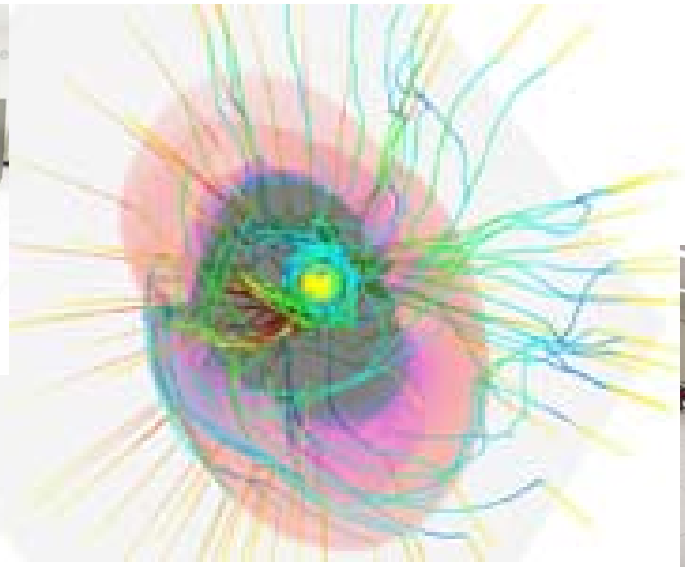
- Computer Science and Mathematics Division.
  - Applied research focused on computational sciences, intelligent systems, and information technologies.
- CSM Research Groups:
  - Advanced Computing Research Testbed
  - Climate Dynamics
  - Computational Chemical Sciences
  - Computational Materials Science
  - Computational Mathematics
  - *Network Cluster Computing (PVM, HARNESS, OSCAR, CCA, …)*

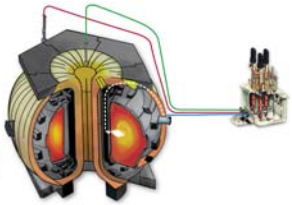


# Computer Science at ORNL

- Center for Computational Sciences.
  - State-of-the-art resources for high performance computational science and computing science research.
  - 3 Top 500 Supercomputers: Cray X1, SGI Altix, IBM Power 4





# High-End Scientific Computing

- A.k.a. high performance or supercomputing.
- Computer systems with many processors.
  - Up-to 130,000 processors in IBM BlueGene\L.
  - Cluster, parallel, distributed  and vector systems.
- Computationally and data intensive applications.
  - Climate, supernovae (stellar explosions), nuclear fusion, material science and nanotechnology simulations.
  - Many research areas: physics, chemistry, biology…
- Ultra-scale = upper end of processor count (+5,000).
  - 10+ TeraFLOPS (10,000,000,000,000 FLOPS and more).

# Availability of HEC Systems

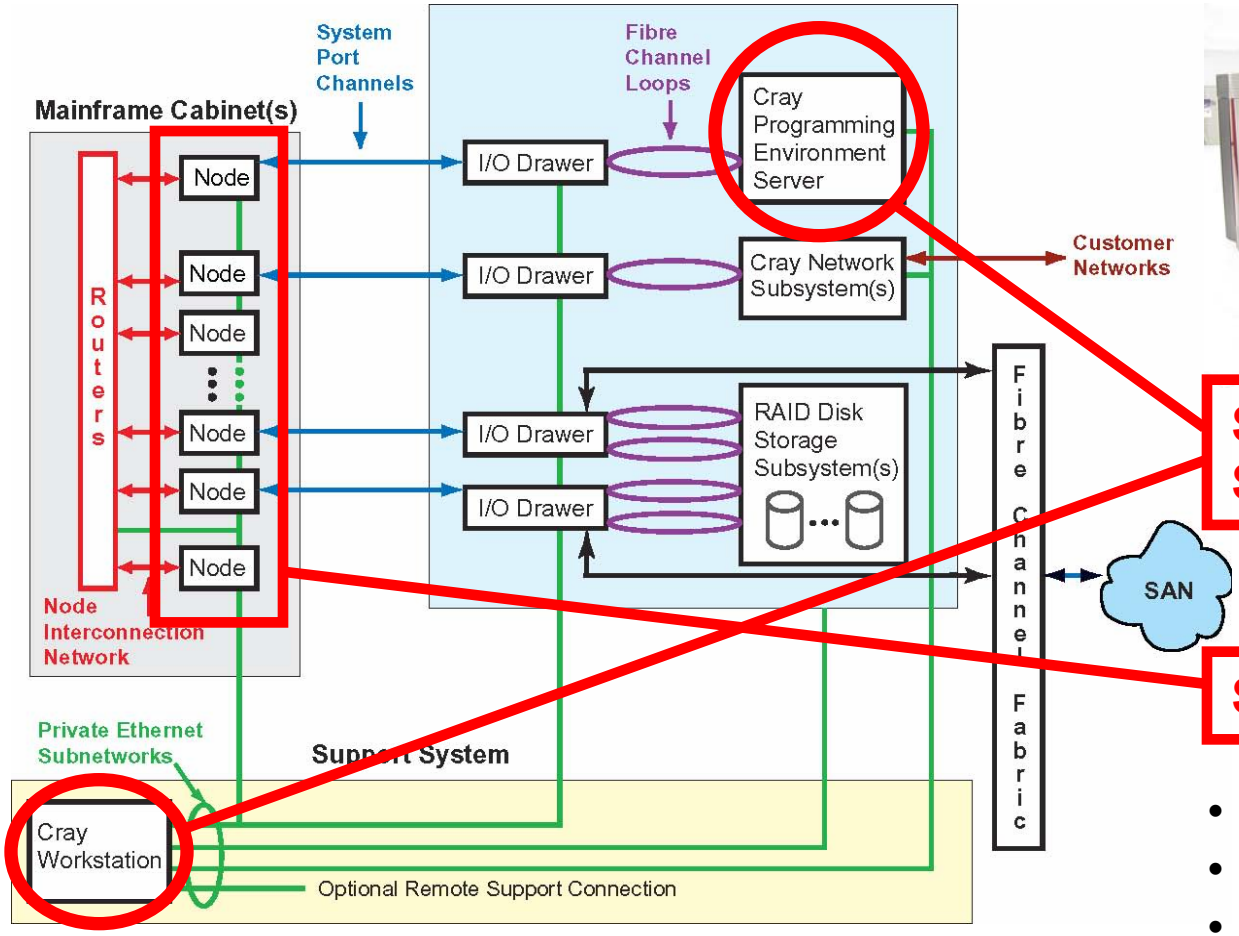
- Today's supercomputers typically need to reboot to recover from a single failure.
- Entire systems go down (regularly and unscheduled) for any maintenance or repair.
- Compute nodes sit idle while their head node or one of their service nodes is down.
- Availability will get worse in the future as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

# Vector Machines: Cray X1 (Phoenix)



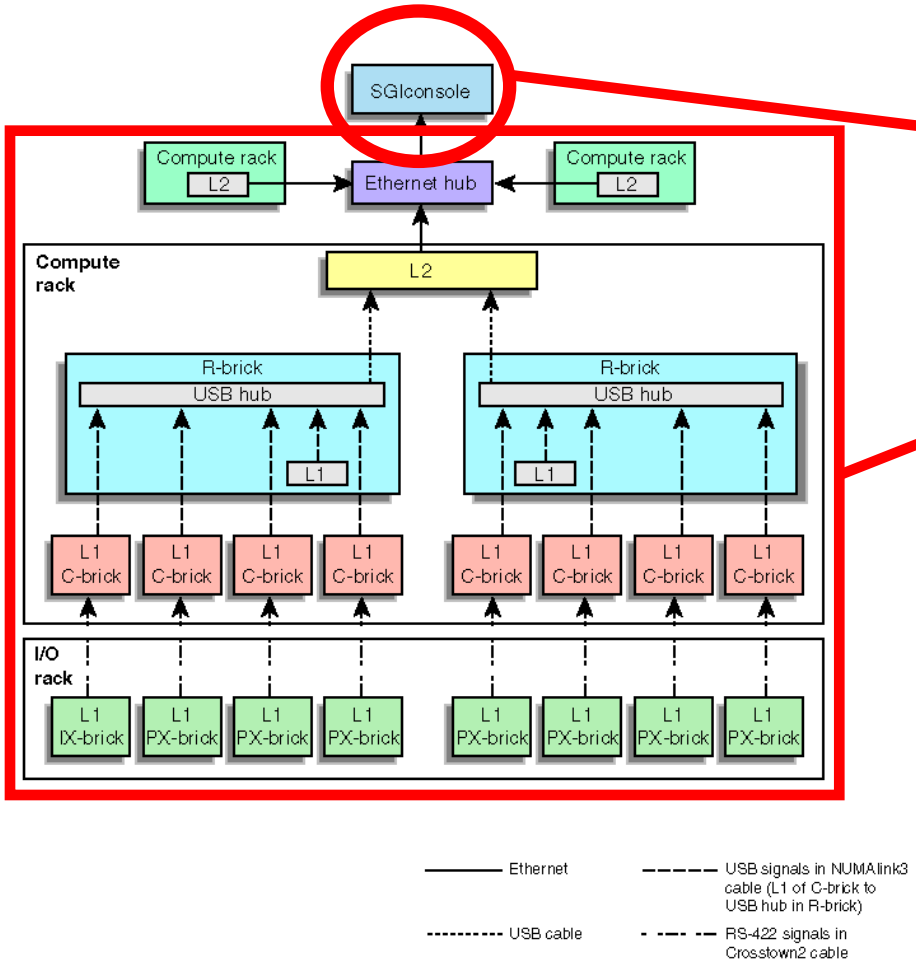

**Single Points of Failure, Single Points of Control**

**Single Points of Failure**

- 512 Vector Processors
- 2 Tera Byte RAM
- 6,4 TFLOPS (29/Top500)

# SSI Clusters: SGI Altix (Ram)

- 256 Itanium 2 Processors
- 2 Tera Byte RAM
- 1,5 TFLOPS (245/Top500)

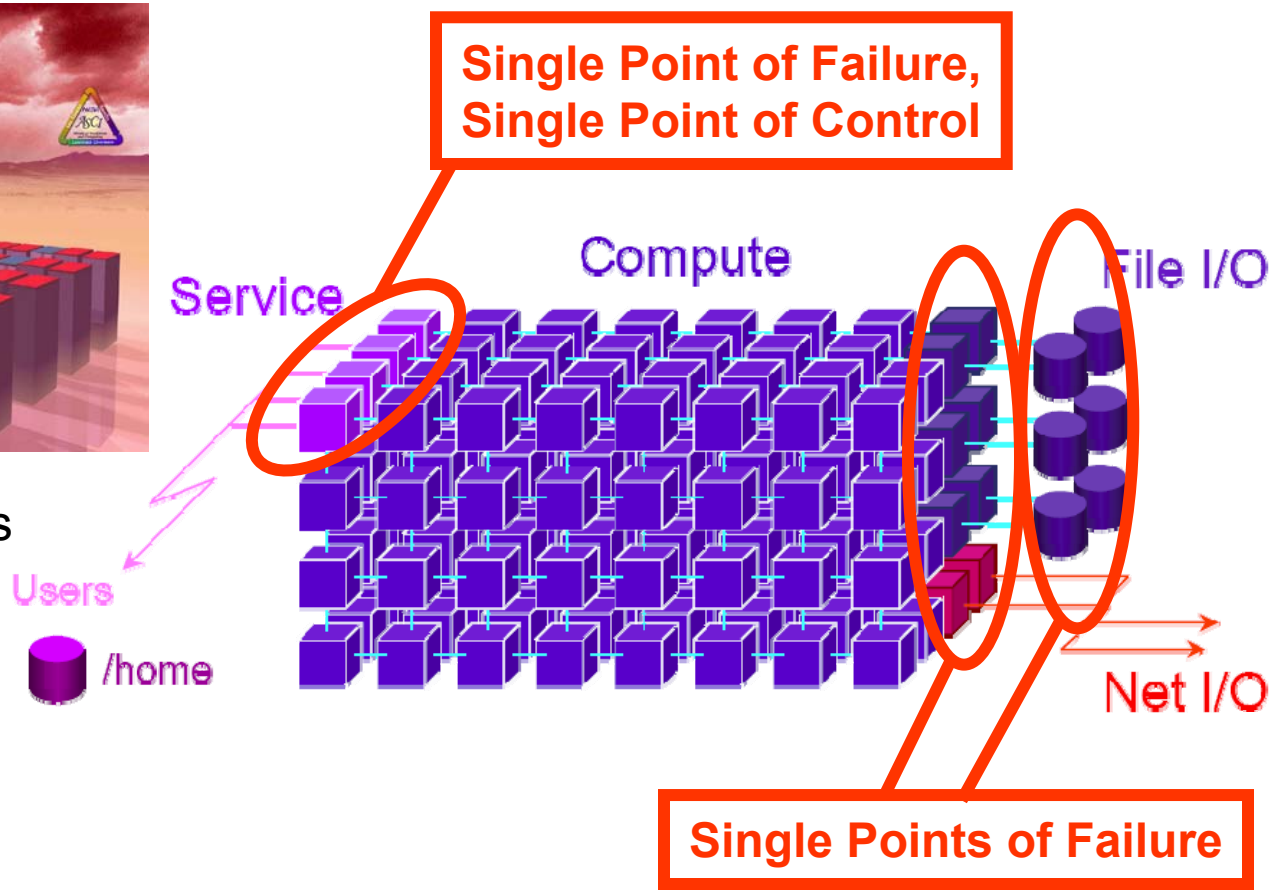



**Single Point of Failure, Single Point of Control**

**Single Points of Failure**

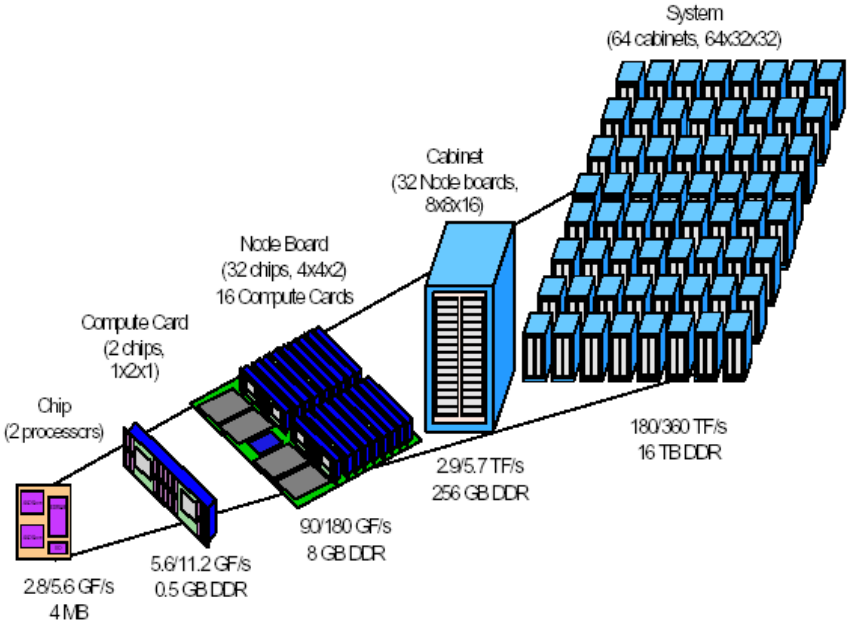
# Clusters/MPPs: Cray XT3 (Jaguar)


- AMD Opteron Processors
- Installation in progress
- 10 TFLOPS in Summer
- 20 TFLOPS in Fall
- 100 TFLOPS in 2006
- 250 TFLOPS in 2007



# IBM BlueGene\L

- 64K diskless nodes with 2 processors per node.
- 512MB RAM per node.
- Additional service nodes.
- 360 Tera FLOPS.
- Over 150k processors.
- Various networks.
- Operational in 2005.
- Partition (512 nodes) outages on single failure.
- MTBF = hours, minutes?




# High Availability Methods

Active/Hot-Standby:

- Single active head node.
- Backup to shared storage.
- Simple checkpoint/restart.
- Rollback to backup.
- Idle standby head node(s).
- Service interruption for the time of the fail-over.
- Service interruption for the time of restore-over.

Active/Active:

- Many active head nodes.
- Work load distribution.
- Symmetric replication between participating nodes.
- Continuous service.
- Always up-to-date.
- No restore-over necessary.
- Virtual synchrony model.
- Complex algorithms.

# High Availability Technology

Active/Hot-Standby:

- HA-OSCAR with active/hot-standby head node.
- Similar projects: HA Linux …
- Cluster system software.
- No support for multiple active/active head nodes.
- No middleware support.
- No support for compute nodes.

Active/Active:

- HARNESS with symmetric distributed virtual machine.
- Similar projects: Cactus …
- Heterogeneous adaptable distributed middleware.
- No system level support.
- Solutions not flexible enough.

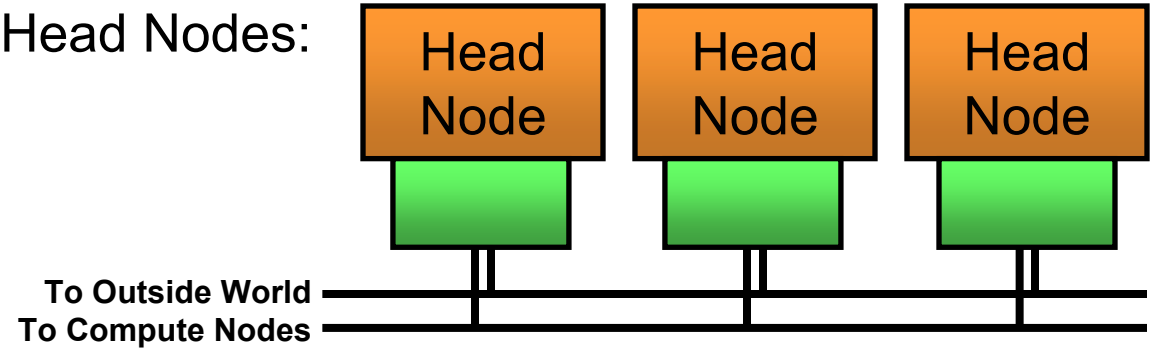
- *System-level data replication and distributed control service needed for active/active head node solution.*
- *Reconfigurable framework similar to HARNESS needed to adapt to system properties and application needs.*

# 2. A Modular Pluggable High Availability Framework

- High availability framework use cases.
- Framework architecture and design.
- Implementation issues.

# Modular HA Framework on Active/ Active Head Nodes

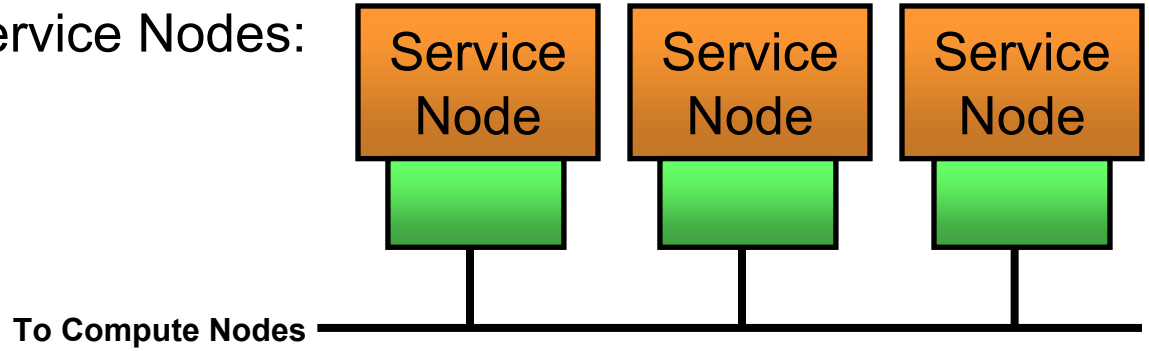
Highly Available Head Nodes:




| | | | |
|---|---|---|---|
| Reliable Services: | Scheduler | MPI Runtime | ... |
| Virtual Synchrony: | Distributed Control Service | | |
| Symmetric Replication: | Data Replication Service | | |
| Reliable Server Groups: | Group Communication Service | | |
| Communication Methods: | TCP/IP | Shared Memory | Etc. |

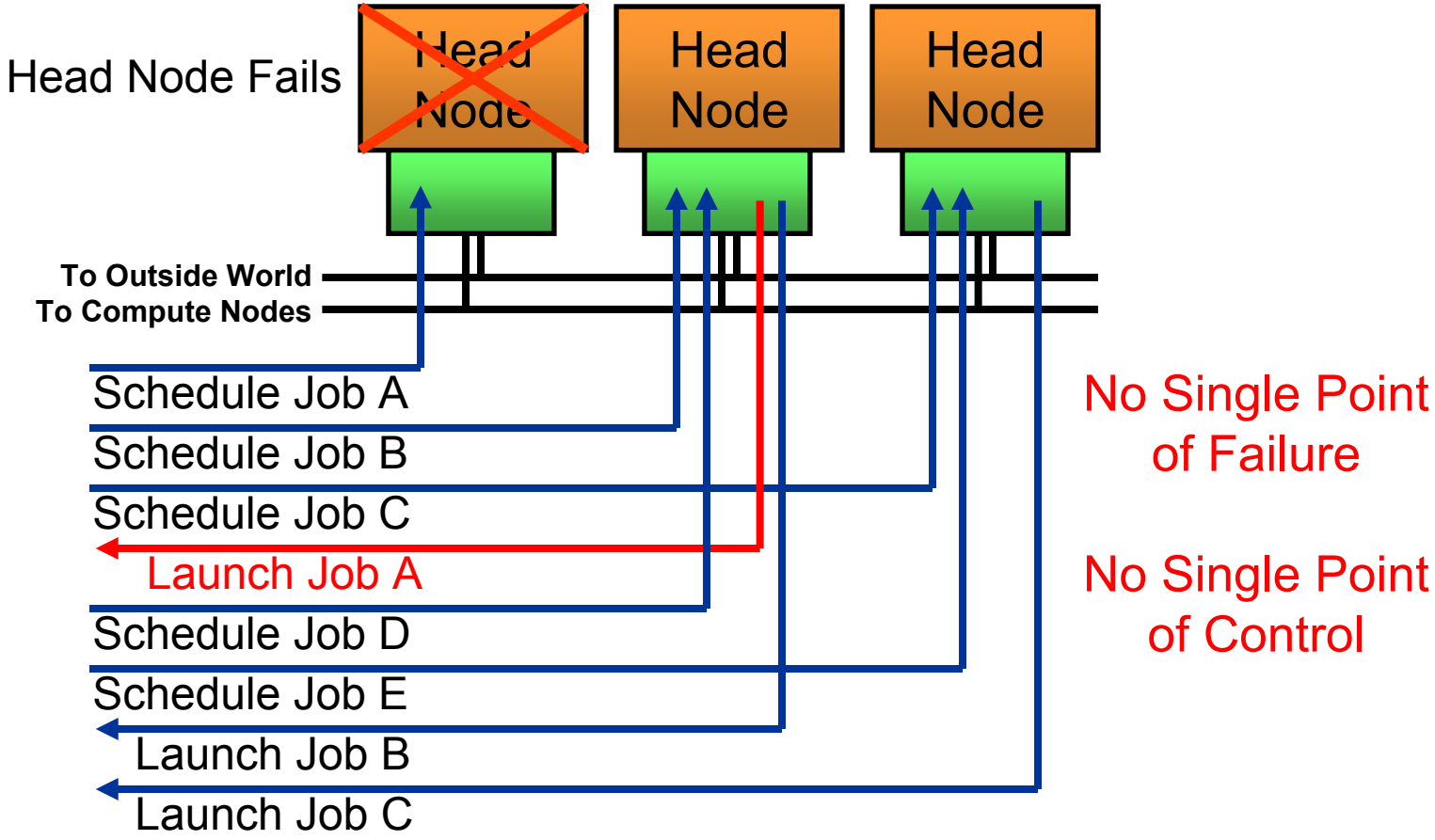
# Modular HA Framework on Active/ Active Service Nodes

Highly Available Service Nodes:




| | | | |
|---|---|---|---|
| Reliable Services: | File System | MPI Runtime | ... |
| Virtual Synchrony: | Distributed Control Service | | |
| Symmetric Replication: | Data Replication Service | | |
| Reliable Server Groups: | Group Communication Service | | |
| Communication Methods: | TCP/IP | Shared Memory | Etc. |

# Modular HA Framework on Active/ Active Head Nodes: Scheduler Example



# Many HA Framework Use Cases

- Active/Active and Active/Hot-standby process state replication for multiple head or service nodes.
  - Reliable system services, such as scheduler, MPI-runtime and system configuration/management/monitoring.
- Memory page replication for SSI and DSM.
- Meta data replication for parallel/distributed FS.
- Super-scalable peer-to-peer diskless checkpointing.
- Super-scalable localized FT-MPI recovery.
- Replicated coherent caching: checkpointing, FS.
- !!! No protection from Byzantine failures !!!

# HA Framework Architecture

- Research in process groups has been around for a while (e.g. Lamport, Amoeba, Birman).
- Research in unified models for process group communication and behavior is about 10 years old.
- Recent (last 4 years) research defined a unified model for process group communication.
- Clear mathematical definitions of process group communication service properties exist today.
- The HA framework architecture is based on the unified model for process group communication.

# Preliminary HA Framework Architecture

**Highly Available Services/Applications**
- Job Scheduler Example
- HA OSCAR Interpartition fSM Management
- OpenSSI, Kerrighed?
- FT-MPI, Open MPI?

**Virtual Synchrony Abstraction**
- Passive Replication Models (Deterministic):
  - Memory, File, Database, State-machine
- Active Replication Models (Non-deterministic):
  - Distributed Control

**Configurable Group Communication**
- Reliable/Atomic Multicast
- External Failure Detection
- Group Membership

**Communication**
- Singlecast, Multicast
- Failure Detection
- TCP/IP, Myrinet, Infiniband, MPI, …

- Interface Specifications in UML.
- C/C++ headers with interfaces.
- Core library with base classes.
- Service libraries and headers.
- Static and dynamic libraries.

*Open MPI Communication Component?*

# Why Pluggable Component Framework?

- A pluggable component framework allows different implementations for the same problem.
- Adaptation to different system properties such as:
  - Network technology and topology (IP, Elan4, Myrinet).
  - Average MTBF of involved components (strong vs. weak).
- Adaptation to different application needs such as:
  - Programming model (active vs. passive replication).
  - Programming interface (memory vs. state machine).
  - Scalability for large number of compute nodes (SSI).

# Why Pluggable Component Framework?

- A pluggable component framework also enables competition for efficient implementations.
- Furthers collaboration by allowing other researchers to contribute their partial solutions.
- It avoids “reinventing the wheel”, which seems to be a common problem in this research area:
  - Harness, FT-MPI, Horus, Coyote, Transis, …, all implement group communication at different levels.
  - These implementations have their advantages and disadvantages in different areas.
  - They are not interchangeable.

# Which Pluggable Framework Properties?

- System properties, such as failure rate and network load, may change at runtime.
- Application needs may change at runtime for collaborative software environments.
- Runtime plug-n-play for adaptation to runtime changes of system properties or application needs.
- On demand runtime loading of components (HARNESS plug-in technology).
- Runtime exchange (transition) of active components is a hard problem to be solved another day.

# Which Programming Language?

The Open MPI Way ↓ My Way ↓

| | C | OOC | C++ | C/C++ |
|---|---|---|---|---|
| Methods, Con/Destructors | No | Yes | Yes | Yes |
| Runtime Type Information | No | Poss | Yes | Yes |
| Virtual Functions | No | Poss | Yes | Yes |
| Inheritance | No | Poss | Yes | Yes |
| Multiple Inheritance | No | Poss | Yes | Yes |
| Virtual Base Classes | No | Poss | Yes | Yes |
| Performance | ++ | 0 | -- | 0 |

# Which OS/Hardware?

- Linux and Unix-type systems.
- Initial prototype on Linux for IA32 on TCP/UDP.
- Target platforms:
  - Cray XT3 and XD1: AMD 64
  - SGI Altix: IA64
  - Cray X1: …, Shem, TCP
  - IBM BlueGene: IBM PowerPC 400
  - IBM SP: IBM Power 5
  - Typical Linux Clusters: IA32 and AMD 64
  - UDP/IP, TCP/IP, SHMEM, MPI, Elan4, Myrinet, Infiniband

# 3. Related Research

- What is the relation to …?

# **MOLAR**: Modular Linux and Adaptive Runtime Support for High-end Computing Operating and Runtime Systems

- The HA Framework is part of the MOLAR project.
- MOLAR addresses the challenges for operating and runtime systems to run large applications efficiently on future ultra-scale high-end computers.

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

LOUISIANA TECH UNIVERSITY

UNM

THE OHIO STATE UNIVERSITY

NORTH CAROLINA STATE UNIVERSITY

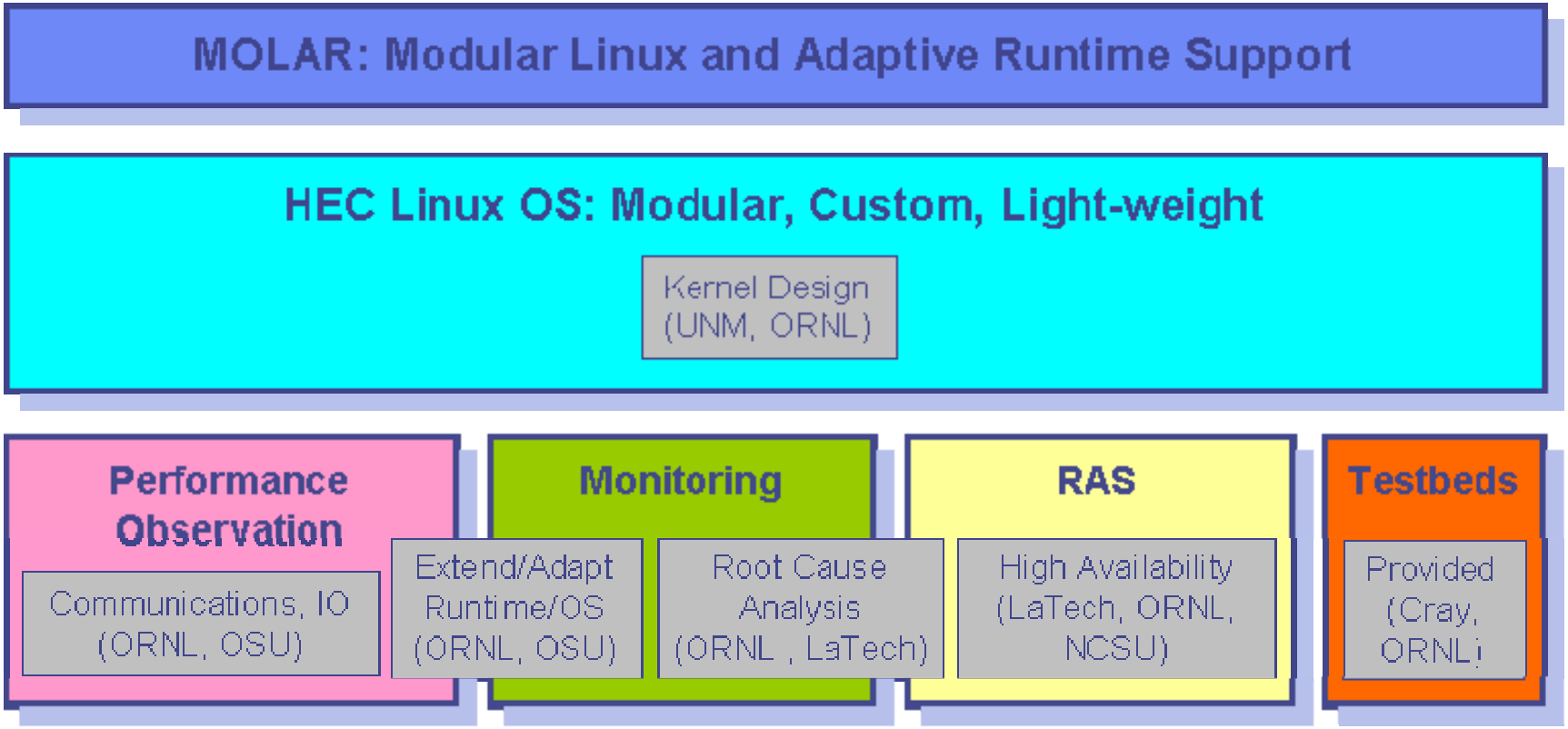
CRAY THE SUPERCOMPUTER COMPANY

Office of Science
U.S. DEPARTMENT OF ENERGY

# **MOLAR**: Research Goals

- Development of a modular and configurable Linux framework.
- Runtime systems to provide a seamless coordination between system levels.
- Monitoring and adaptation of the operating system, runtime, and applications.
- Reliability, availability, and serviceability (RAS)
- Efficient system management tools.

# **MOLAR:** HEC OS/R Research Map



# What is the relation to Horus?

- *Horus* is a group communication framework in C.
- It is based on stackable micro-protocol layers.
- It uses a fixed unified interface for all protocols.
- It is only free for research purpose.
- NDA for code and binary access, which can result in licensing issues later (e.g. reuse of code).
- The micro-protocol layer programming model will be supported in the HA framework among others.
- *Ensemble* is a free follow-on developed in OCaml.

# What is the relation to Coyote?

- *Coyote* is a group communication framework in C.
- It is based on a reconfigurable event-driven micro-protocol state machine.
- It allows protocol adaptation (transition) at runtime.
- The micro-protocol state machine programming model will also be supported in the HA framework.
- *Cactus* is an ongoing follow-on project targeting real-time properties and configurable QoS.

# What is the relation to …?

- Other group communication solutions exist.
- *Transis, Totem*, … (still counting).
- Most implement one specific algorithm.
- They were developed in the late 90s, while group communication models were still a research issue.
- None of these solutions provide full configurability.
- Their protocols and algorithms can be easily moved into the HA framework.
- The HA framework is based on a generalization of the developed solutions.

# 4. Future Work

- Plan for next months.
- Further research opportunities.

# Plan For The Next Months

- Framework specification - publication.
  - Clearly define individual services and their interfaces.
  - Solve implementation issues, like use cases, multi-threading, mutual exclusions and daemons.
- Framework implementation.
  - Implement basic services that others depend on first.
  - Implement one protocol to allow application testing.
  - Implement multiple protocols with different features.
- Further exploration of applications and use cases.
- !!! Come up with a nice project name !!!
- !!! Summer students !!!

# Further Research Opportunities

- Scalable group communication algorithms.
  - 100,000 processors and beyond.
  - Peer-to-peer diskless checkpointing.
  - SSI: Open SSI, Kerrighed.
- Runtime adaptation to system changes.
  - Weak vs. strong protocols.
  - Mobile wireless devices.
- Automatic configuration on startup.
  - Simplify ease of use.
- Highly available applications.
- Carrier Grade Linux.

# High Availability for Ultra-Scale High-End Scientific Computing

Christian Engelmann
Oak Ridge National Laboratory