



On Programming Models for Service-Level High Availability



Christian Engelmann^{1,2}, Stephen L. Scott¹,
Chokchai (Box) Leangsuksun³, Xubin (Ben) He⁴

¹ Oak Ridge National Laboratory, Oak Ridge, USA

² The University of Reading, Reading, UK

³ Louisiana Tech University, Ruston, USA

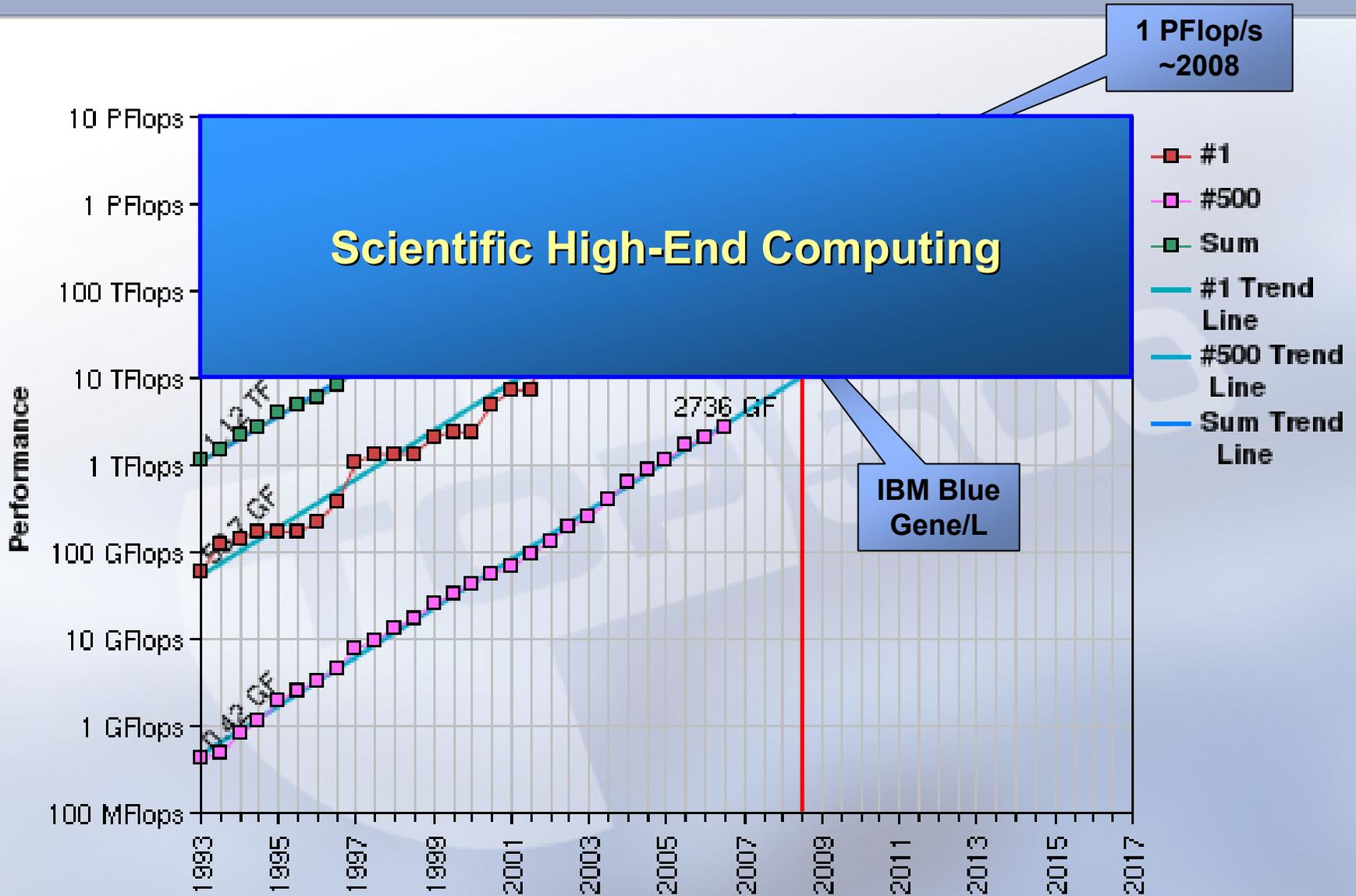
⁴ Tennessee Tech University, Cookeville, USA

Talk Outline

- Scientific high-end computing (HEC)
- Availability deficiencies of today's HEC systems
- Projects and accomplishments overviews
- High availability (HA) models for services
- Developed prototypes overview
- Existing limitations and most pressing issues
- Generalization of HA programming models
- Generic HA framework infrastructure

Scientific High-End Computing (HEC)

- Large-scale HPC systems.
 - Tens-to-hundreds of thousands of processors.
 - Current systems: IBM Blue Gene/L and Cray XT4
 - Next-generation: petascale IBM Blue Gene and Cray XT
- Computationally and data intensive applications.
 - 10 TFLOP – 1PFLOP with 10 TB – 1 PB of data.
 - Climate change, nuclear astrophysics, fusion energy, materials sciences, biology, nanotechnology, ...
- Capability vs. capacity computing
 - Single jobs occupy large-scale high-performance computing systems for weeks and months at a time.



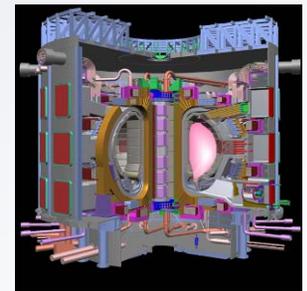
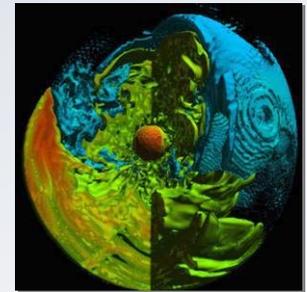
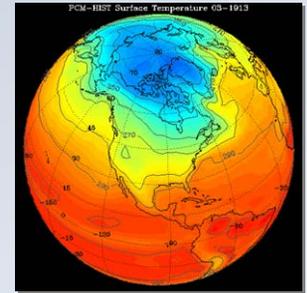
National Center for Computational Sciences

- **40,000 ft² (3700 m²) computer center:**
 - 36-in (~1m) raised floor, 18 ft (5.5 m) deck-to-deck
 - 12 MW of power with 4,800 t of redundant cooling
 - High-ceiling area for visualization lab:
 - 35 MPixel PowerWall, Access Grid, etc.
- **2 systems in the Top 500 List of Supercomputer Sites:**
 - Jaguar: 10? Cray XT3, MPP with 12500 dual-core Processors ⇒ 119 TFlop.
 - Phoenix: 32? Cray X1E, Vector with 1014 Processors ⇒ 18 TFlop.



At Forefront in Scientific Computing and Simulation

- Leading partnership in developing the National Leadership Computing Facility
 - Leadership-class scientific computing capability
 - 100 TFlop/s in 2007 (recently installed)
 - 250 TFlop/s in 2007/8 (commitment made)
 - 1 PFlop/s in 2008/9 (proposed)
- Attacking key computational challenges
 - Climate change
 - Nuclear astrophysics
 - Fusion energy
 - Materials sciences
 - Biology
- Providing access to computational resources through high-speed networking (10Gbps)



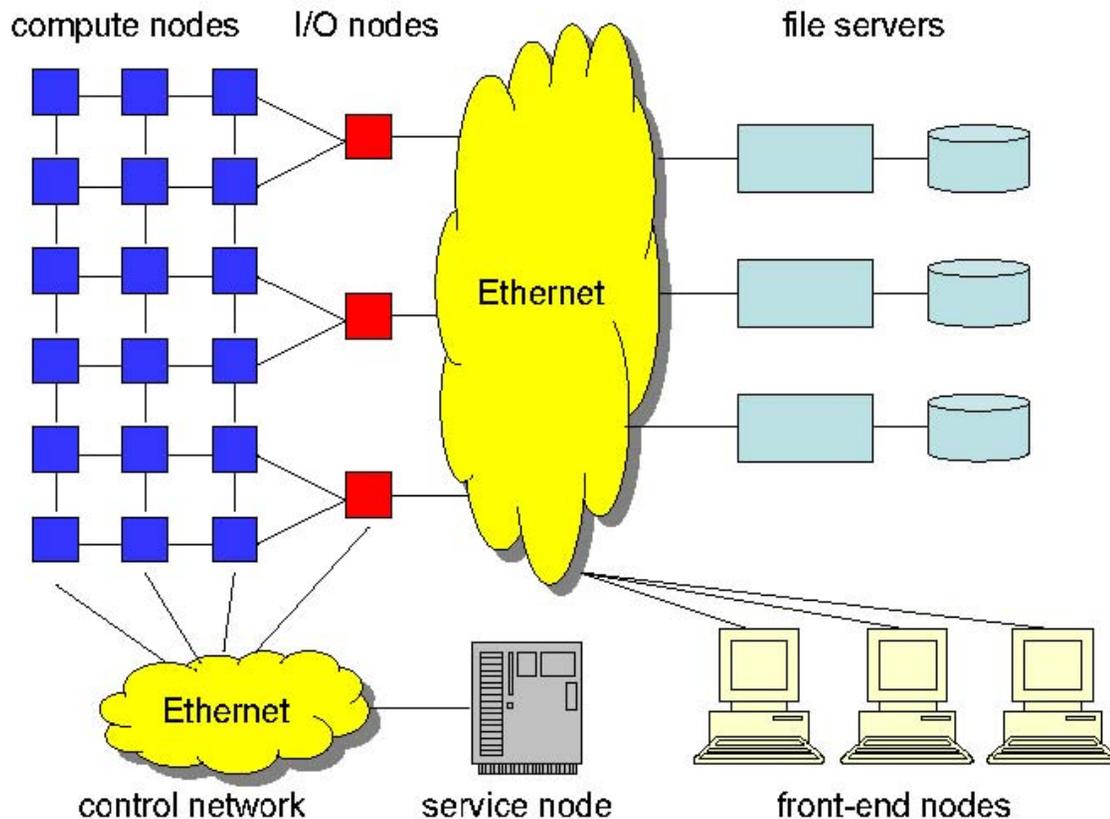
Availability Measured by the Nines

see <<http://info.nccs.gov/resources>> for current status of HPC systems at Oak Ridge National Laboratory

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

Typical HEC System Architecture



Typical failure causes:

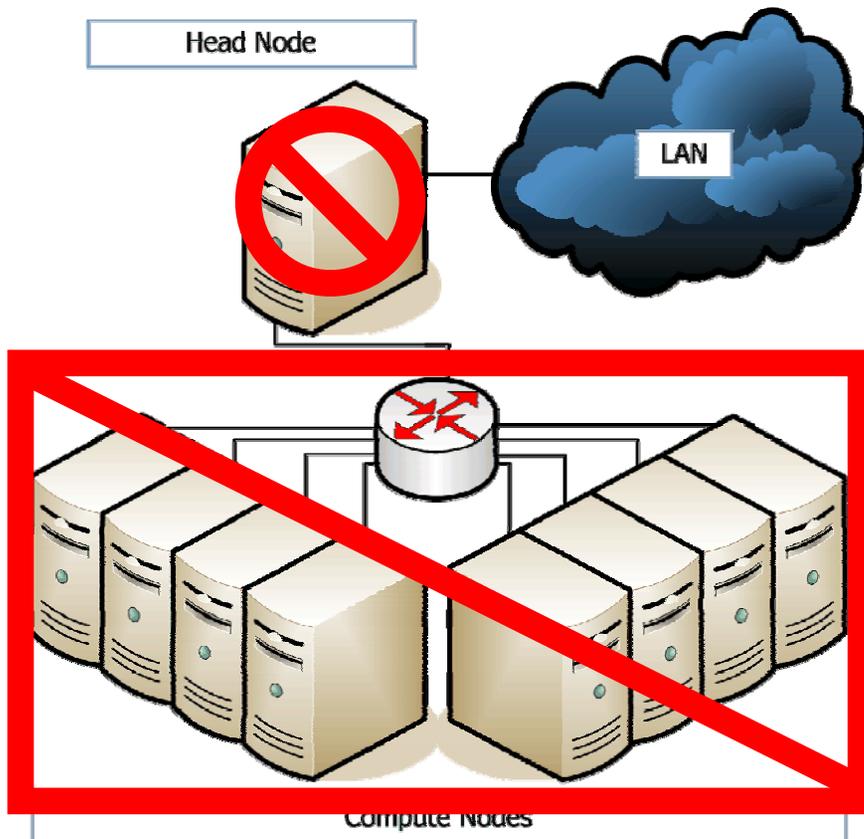
- Overheating !!!
- Memory errors
- Network errors
- Other hardware issues
- Software bugs

Different scale requires different solutions:

- Compute nodes (10,000+)
- Front-end, service, and I/O nodes (50+)

Image source: Moreira et al., "Designing a Highly-Scalable Operating System: The Blue Gene/L Story"
Proceedings of the 2006 ACM/IEEE Conference on Supercomputing, Nov. 11-17, Tampa, FL, USA.

Single Head/Service Node Problem



- Single point of failure.
- Compute nodes sit idle while head node is down.
- $A = \text{MTTF} / (\text{MTTF} + \text{MTTR})$
- MTTF depends on head node hardware/software quality.
- MTTR depends on the time it takes to repair/replace node.
- $\text{MTTR} = 0 \rightarrow A = 1.00$ (100%) **continuous availability.**

Projects Overview

- Initial **HA-OSCAR** research in active/standby technology for the batch job management system
- Ongoing **MOLAR** research in active/standby, asymmetric and symmetric active/active technology
- Recent **RAS LDRD** research in symmetric active/active technology
- ➔ 3-4 years of research and development in high availability for high-performance computing system services

Accomplishments Overview

- Investigated the overall background of HA technologies in the context of HPC
 - Detailed problem description
 - Conceptual models
 - Review of existing solutions
- Developed different replication strategies for providing high availability for HPC system services
 - Active/standby
 - Asymmetric active/active
 - Symmetric active/active
- Implemented several proof-of-concept prototypes

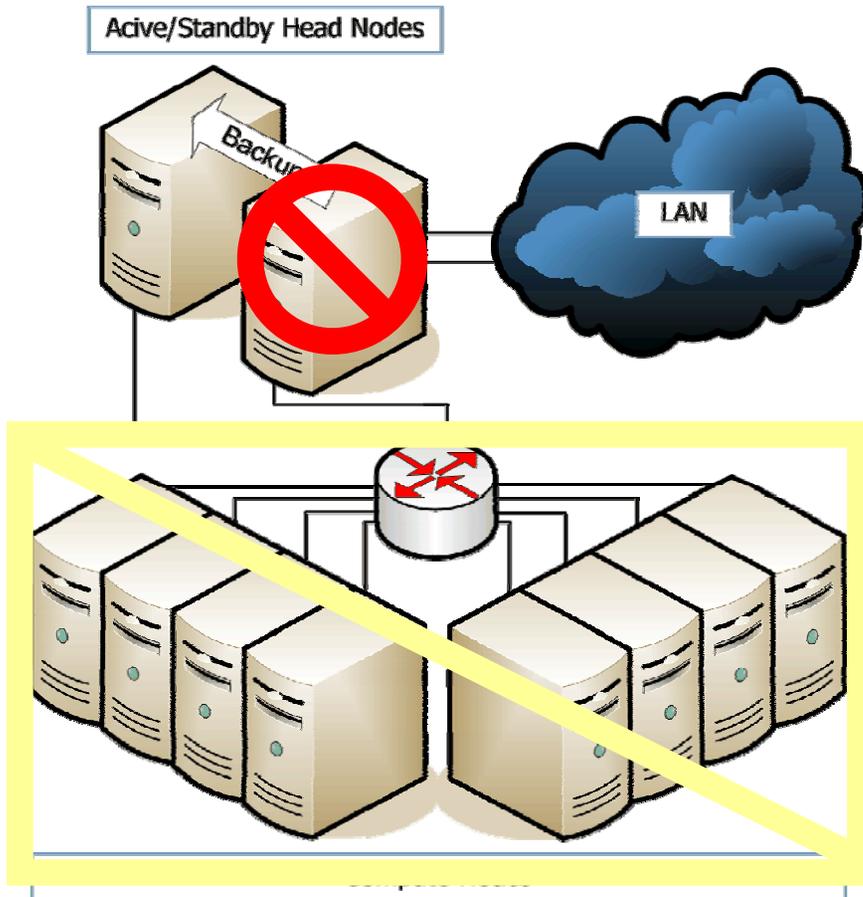
Active/Standby with Shared Storage

Active/Standby Head Nodes with Shared Storage



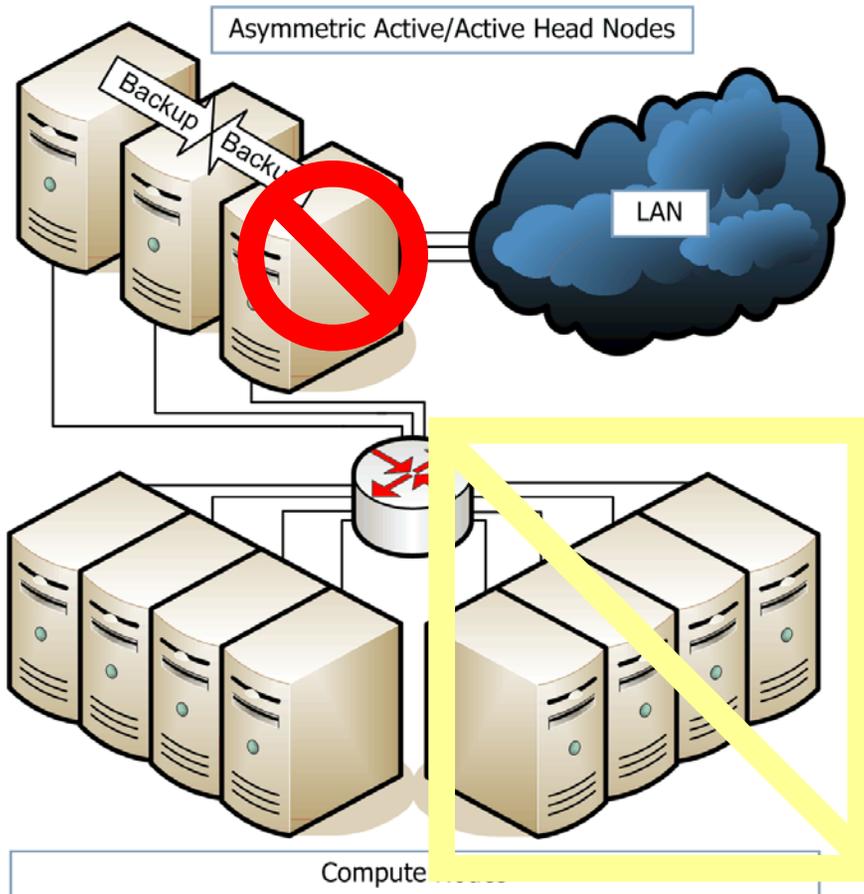
- Single active head node
- Backup to shared storage
- Simple checkpoint/restart
- Fail-over to standby node
- Possible corruption of backup state when failing during backup
- Introduction of a new single point of failure
- Correctness and availability are NOT ALWAYS guaranteed
- ➔ SLURM, meta data servers of PVFS and Lustre

Active/Standby Redundancy



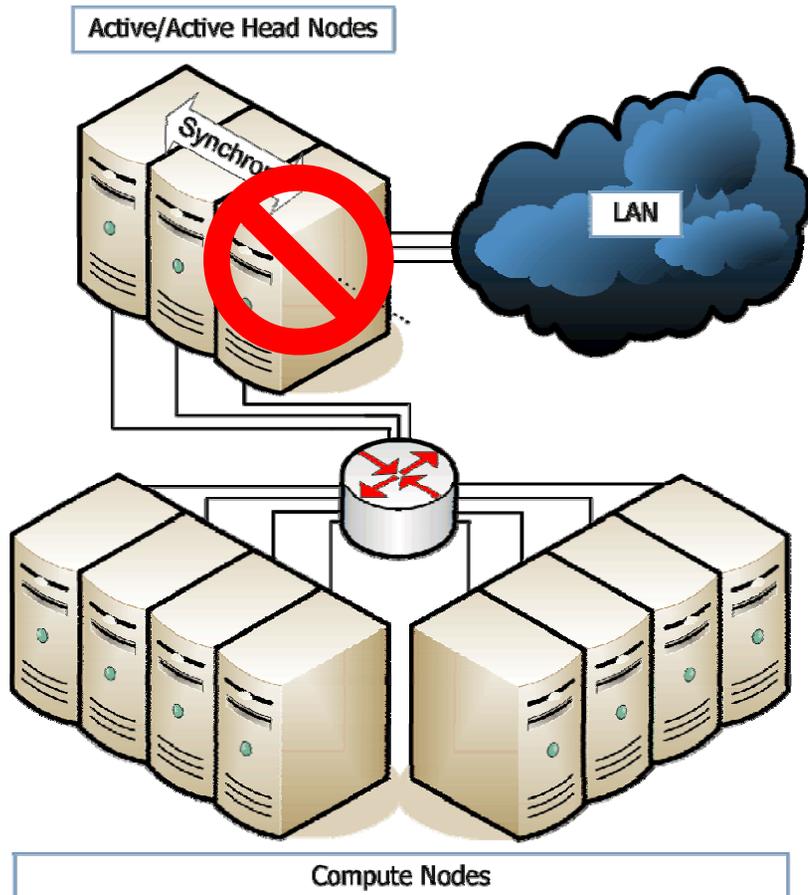
- Single active head node
- Backup to standby node
- Simple checkpoint/restart
- Fail-over to standby node
- Idle standby head node
- Rollback to backup
- Service interruption for fail-over and restore-over
- ➔ Torque on Cray XT
- ➔ HA-OSCAR prototype

Asymmetric Active/Active Redundancy



- Many active head nodes
 - Work load distribution
 - Optional fail-over to standby head node(s) ($n+1$ or $n+m$)
 - No coordination between active head nodes
 - Service interruption for fail-over and restore-over
 - Loss of state w/o standby
 - Limited use cases, such as high-throughput computing
- ➔ Prototype based on HA-OSCAR

Symmetric Active/Active Redundancy



- Many active head nodes
- Work load distribution
- Symmetric replication between head nodes
- Continuous service
- Always up-to-date
- No fail-over necessary
- No restore-over necessary
- Virtual synchrony model
- **Complex algorithms**
- JOSHUA prototype for Torque

Developed Prototypes Overview (1/2)

- **Active/Standby HA-OSCAR**
 - High availability for Open PBS/TORQUE
 - Integration with compute node checkpoint/restart
- **Asymmetric active/active HA-OSCAR**
 - High availability for Open PBS & SGE
 - High throughput computing solution
- **Symmetric active/active JOSHUA**
 - High availability for PBS TORQUE
 - Fully transparent replication

Existing Limitations

- The active/standby and asymmetric active/active technology interrupts the service during fail-over
- Generic $n+1$ or $n+m$ asymmetric active/active configurations have not been developed yet
- The $2+1$ asymmetric active/active configuration uses two different service implementations
- The developed symmetric active/active technology has certain stability and performance issues
- All developed prototypes use a customized high availability environment
- Missing interaction with compute node fault tolerance mechanisms (except for HA-OSCAR for head node fail-over)

Most Pressing Issues

- For production-type deployment
 - Stability – guaranteed quality of service
 - Performance – low replication overhead
 - Interaction with compute node fault tolerance mechanisms – e.g. procedure for failing PBS mom
 - ➔ Testing, enhancements, and staged deployment
- For extending the developed technologies
 - Portability – ability to apply technology to different services
 - Ease-of-use – simplified service HA management (RAS)
 - ➔ Generic HA framework needed

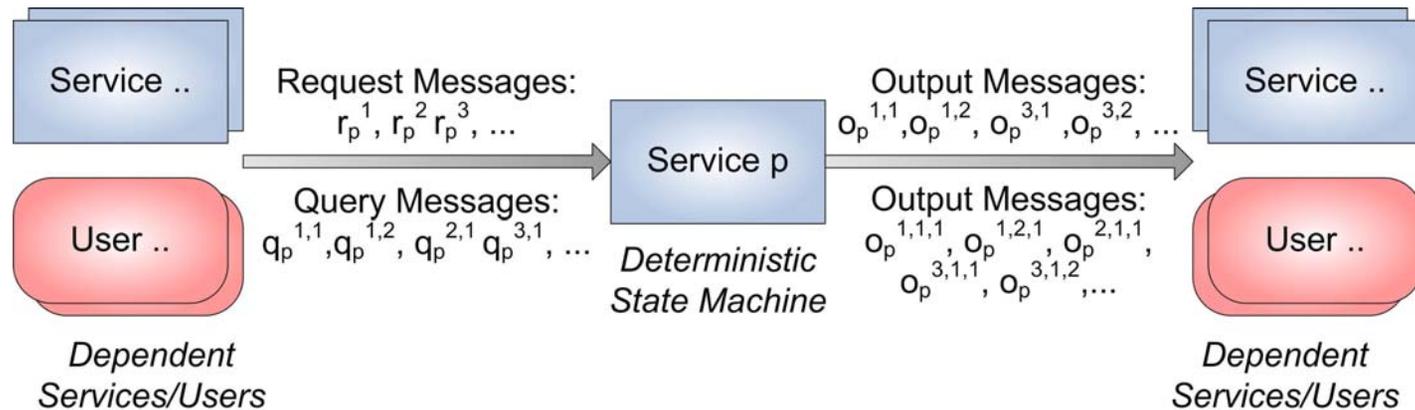
Next Step: Generic HA Framework

- Generalization of HA programming models
 - Active/Standby
 - Asymmetric active/active
 - Symmetric active/active
- Enhancing the transparency of the HA infrastructure
 - Minimum adaptation to the actual service protocol
 - Virtualized communication layer for abstraction
- ➔ Portability
- ➔ Ease-of-use

Failure Model

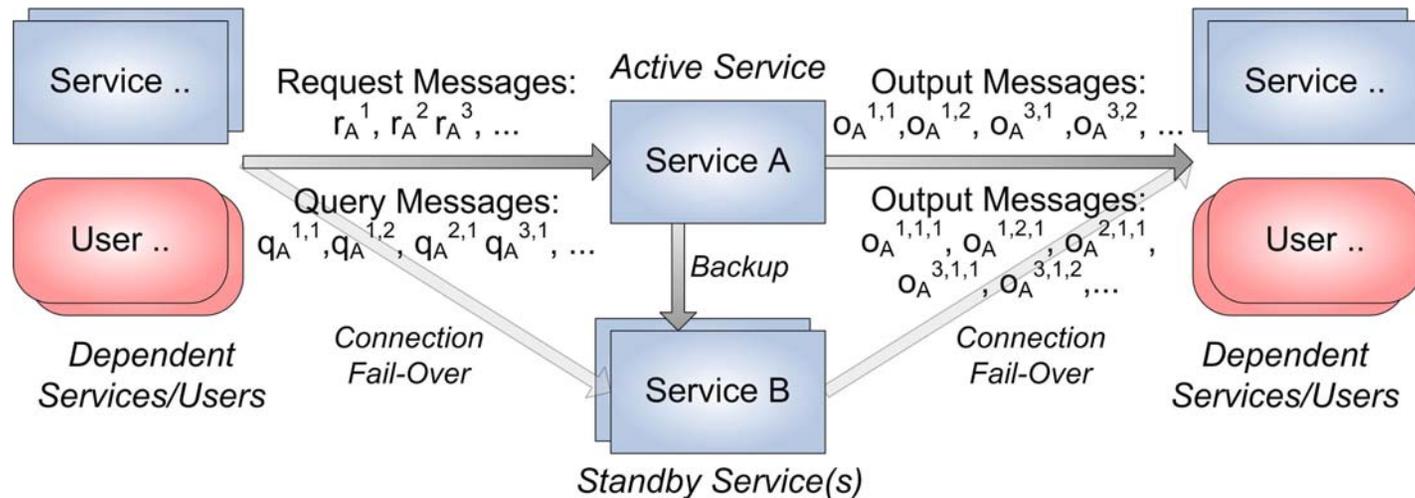
- Fail-stop
 - The service, its node, or its communication links, fail by simply stopping.
 - Failure detection mechanisms may be deployed to assure fail-stop behavior in certain cases, such as for incomplete or garbled messages.
- Permanent failures
 - Non-transient behavior assured by detection mechanisms via node fencing.
 - Recovery requires external intervention, such as repair or replacement of the failed component.
- Both assumptions match real-world properties.

Communicating Process Generalization



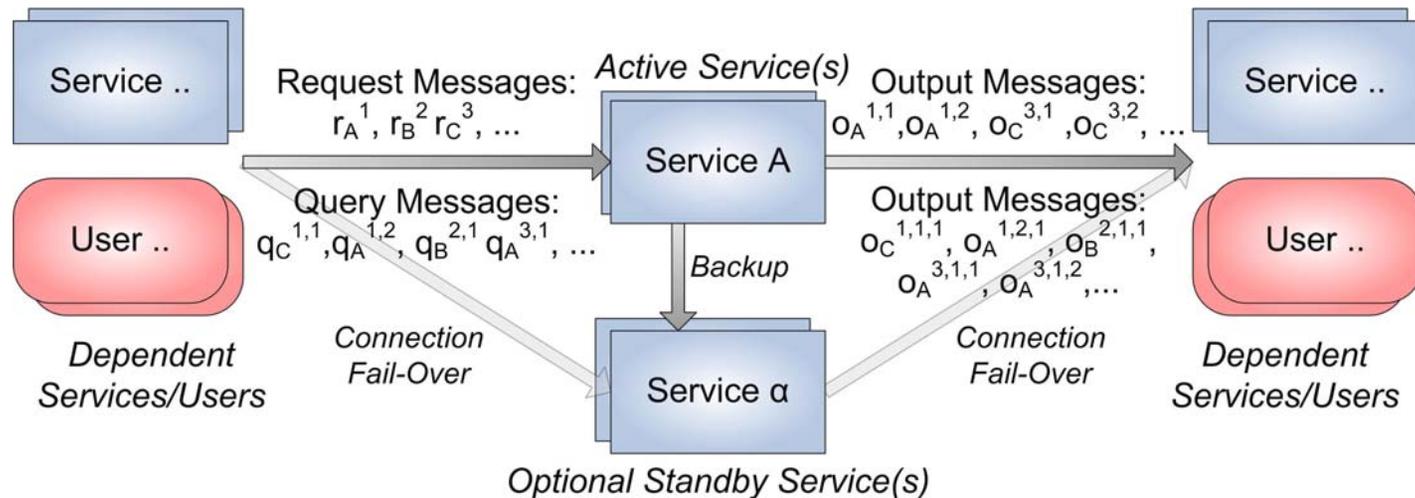
- Most, if not all, HPC system services are deterministic
- Non-determinism introduced by random number generators or unsynchronized timers:
 - Removal of the use of random number generators in HPC system services
 - Synchronization of timers (clocks) between replicas is trivial:
 - Closely coupled local area networks with low and constant latency
 - Clock skew tolerable within certain boundaries (not real-time, not fully synchronous)

Active/Standby Generalization



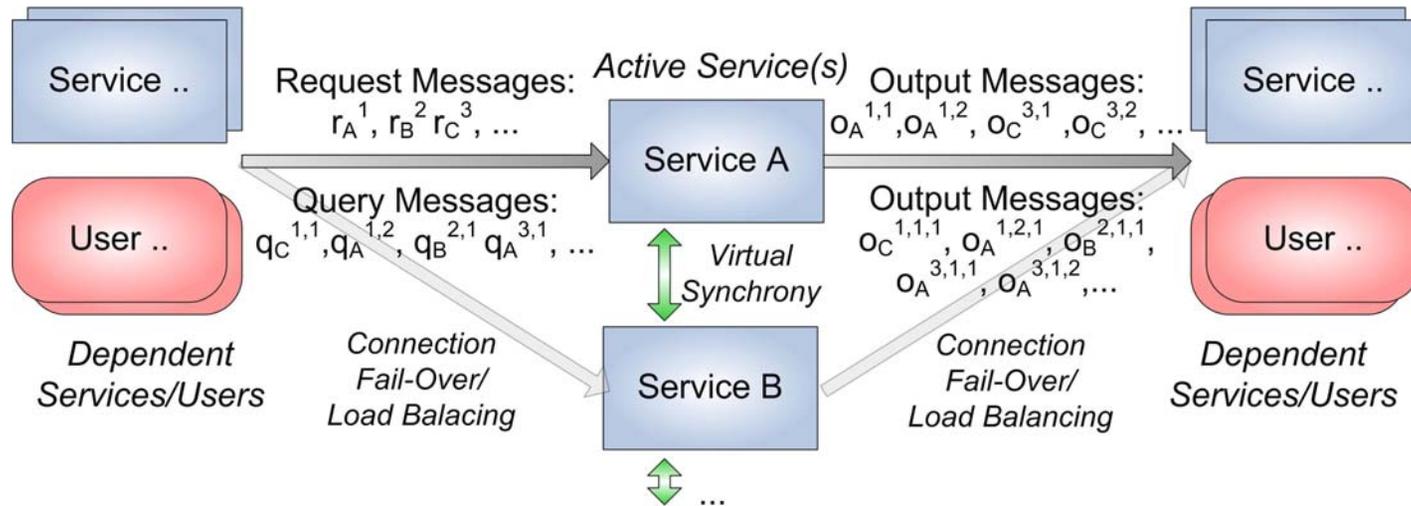
- Warm-Standby:
 - Regular state updates from Active Service to Standby Service (push or pull)
- Hot-Standby
 - On-change state updates from Active Service to Standby Service (push)
- Group communication style consistency required for state updates to multiple Standby Services
- Note: ARES Paper on extended Hot/Passive Replication semantics

Asymmetric Active/Active Generalization



- Replication of service capability via multiple Active Services
- No replication of state among Active Services
- Mechanisms and semantics for optional Standby Services are the same as for Active/Standby

Symmetric Active/Active Generalization



- Replication of service capability via multiple Active Services
- Replication of state among Active Services
- Virtual synchrony (active replication) model

Comparison of Replication Methods

Model	MTTR	Latency Overhead
Warm-Standby	$T_d + T_f + T_c + T_r$	0
Hot-Standby	$T_d + T_f + T_r$	$2l_{A,B}$
Asymmetric	$T_d + T_f + T_c + T_r$ or $T_d + T_f + T_r$	0 or $2l_{A,\alpha}$
Symmetric	T_r	$O(n)$

T_d = time between failure occurrence and detection

T_f = time between failure detection and fail-over

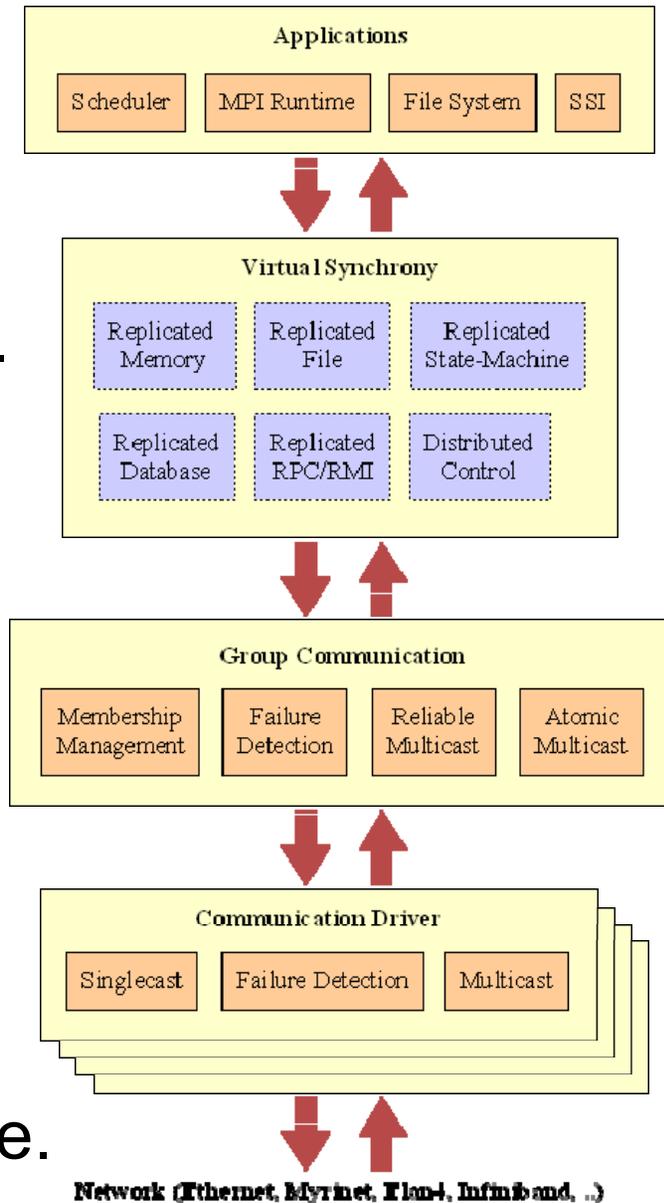
T_c = time to recover from checkpoint to previous state

T_r = time to reconfigure client/user connection

$l_{A,B}$ = communication latency between A and B

Modular HA Framework

- Pluggable component framework.
 - Communication drivers.
 - Group communication.
 - Virtual synchrony.
 - *Applications.*
- Interchangeable components.
- Adaptation to application needs, such as level of consistency.
- Adaptation to system properties, such as network and system scale.



Current Prototype



- Unique, flexible, dynamic, C-based component framework: Adaptive Runtime Environment (ARTE)
- Dynamic component loading/unloading on demand
- *XML as interface description language (IDL)*
- “Everything” is a component:
 - Communication driver modules
 - Group communication layer modules
 - Virtual synchrony layer modules

Future Work

- Continued implementation of framework components
 - Implementation of HA programming model components
- Integration with existing prototypes
 - For example, replacing Transis with the framework
- Availability and reliability modeling
- Testing and benchmarking
- What about communication security/integrity?
 - For client-server connections across administrative domains
 - For distributed computing scenarios

MOLAR: Adaptive Runtime Support for High-end Computing Operating and Runtime Systems

- Addresses the challenges for operating and runtime systems to run large applications efficiently on future ultra-scale high-end computers.
- Part of the Forum to Address Scalable Technology for Runtime and Operating Systems (FAST-OS).
- MOLAR is a collaborative research effort (www.fastos.org/molar):



OAK RIDGE NATIONAL LABORATORY

MANAGED BY UT-BATTELLE FOR THE DEPARTMENT OF ENERGY



NC STATE UNIVERSITY



LOUISIANA TECH
UNIVERSITY



The University of Reading

CRAY THE SUPERCOMPUTER COMPANY

On Programming Models for Service-Level High Availability



Christian Engelmann^{1,2}, Stephen L. Scott¹,
Chokchai (Box) Leangsuksun³, Xubin (Ben) He⁴

¹ Oak Ridge National Laboratory, Oak Ridge, USA

² The University of Reading, Reading, UK

³ Louisiana Tech University, Ruston, USA

⁴ Tennessee Tech University, Cookeville, USA