Symmetric Active/Active High Availability for High-Performance Computing System Services

Christian Engelmann

The University of Reading, UK
Oak Ridge National Laboratory, USA

Outline

- Background and motivation
- Objectives and methodology
- Previous work
- Taxonomy, architecture and methods
- Developed prototypes
- Summary and future work
- Publications and acknowledgements

Background

- High-performance computing (HPC)
  - Rooted in parallel and distributed computing
  - Today's HPC systems are mostly parallel architectures with some distributed features
- Scientific HPC and computational science
  - Combining domain-specific science, computational methods, parallel algorithms, and collaboration tools to solve problems in
    - Climate dynamics
    - Nuclear astrophysics
    - Fusion energy
    - ...

Motivation

Installed	System	Processors	MTBF	Measured	Source
2000	ASCI White	8,192	40.0h	2002	[15]
2001	PSC Lemieux	3,016	9.7h	2004	[16]
2002	Seaborg	6,656	351.0h	2007	[17]
2002	ASCI Q	8,192	6.5h	2002	[18]
2003	Google	15,000	1.2h	2004	[16]
2006	Blue Gene/L	131,072	147.8h	2006	[19]

- HPC system reliability and availability is deceasing rapidly
  - More frequent failures and less efficient recovery due to higher component count (e.g., processors, memory modules, …)
  - More frequent failures due to higher soft error vulnerability (e.g., double-bit errors in ECC memory)
- ***High availability as well as high performance is needed for next-generation HPC systems***

Objectives

- Provide high availability solutions for HPC head and service nodes
  - They are the command and control backbone (and “Achilles heel”) of a HPC system
- Combine and extend prior high availability research efforts for:
  - HPC head and service nodes
  - Distributed systems
- Focus on state-machine replication solutions based on virtual synchrony

Methodology

- Review related previous work
- Define a high availability taxonomy
- Examine HPC system architectures and their availability deficiencies
- Investigate high availability methods for HPC head and service nodes
- Develop prototype solutions for HPC head and service node high availability

Previous Work

- HPC head/service node high availability
  - Basic mechanisms only (shared 1+1 and 2+1)
  - No symmetric active/active replication
- HPC compute node high availability
  - Relies on head/service node high availability (Checkpoint/restart, message logging, ABFT,...)
- Distributed systems high availability
  - Basic or high-overhead advanced mechanisms (state-machine replication, Byzantine)
- IT and telecommunication industry
  - Basic or high-overhead advanced mechanisms (shared 1+1, N+1, and N+m; 1+1, DMR)

Modern Service-Level High Availability Taxonomy

- No redundancy → Manual masking
- Hardware redundancy → Active/cold standby
- Hardware and software redundancy:
  - Active/warm standby → Replication in intervals, 1+m service nodes
  - Active/hot standby → Replication on change, 1+m service nodes
  - Asymmetric active/active → High availability clustering, n+m service nodes
  - Symmetric active/active → State-machine replication, n service nodes
- *Resolving the ambiguity of active/active*
- *Omitting active and passive replication terms*

Availability Deficiencies in Modern HPC System Architectures

- Single point of failure
  - Interrupts the entire system in case of a failure
  - Degraded system after reconfiguration
  - *Some (partition) service nodes*
  - *Most (partition) compute nodes*
- Single point of failure and control
  - Inoperable system until repair
  - *Head node and most (partition) service nodes*
  - *Some (partition) compute nodes*
- Non-critical system service
  - Single point of failure
  - *User/software management, development environment*
- Critical system service
  - Single point of failure and control
  - *Job & resource management, communication, file system, …*

Unified Definition of Service-Level High Availability Methods

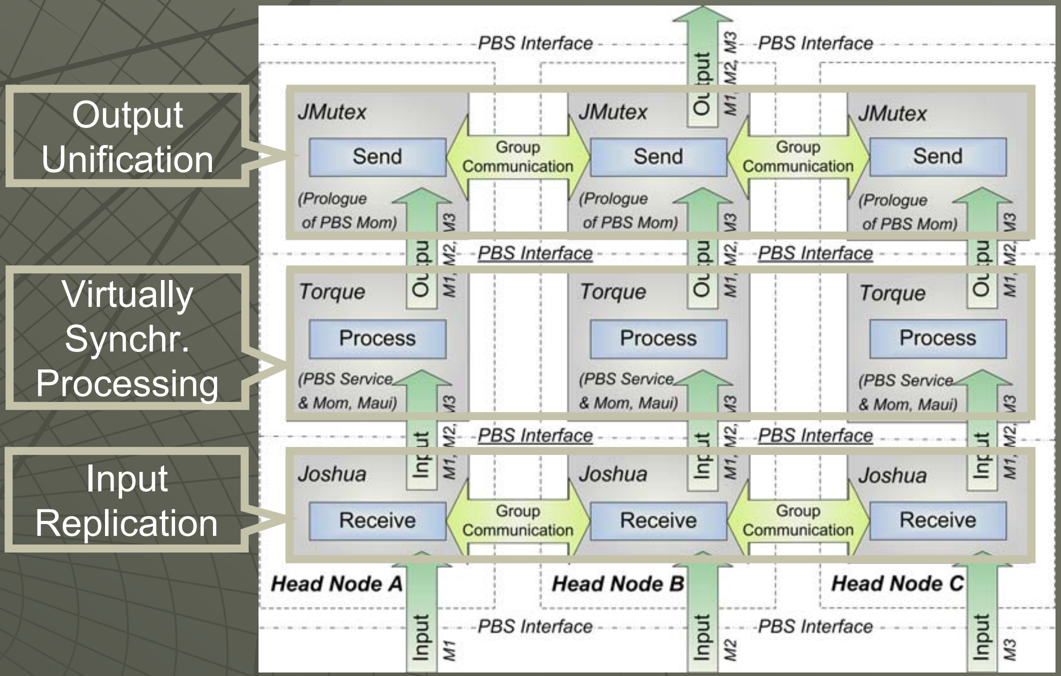
Method	$MTTR_{recovery}$	Latency Overhead
Warm-Standby	$T_d + T_f + T_r + T_c$	0
Hot-Standby	$T_d + T_f + T_r$	$2l_{A,B}$, $O(log_2(n))$, or worse
Asymmetric with Warm-Standby	$T_d + T_f + T_r + T_c$	0
Asymmetric with Hot-Standby	$T_d + T_f + T_r$	$2l_{A,\alpha}$, $O(log_2(n))$, or worse
Symmetric	$T_d + T_f + T_r$	$2l_{A,B}$, $O(log_2(n))$, or worse

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 connections
$l_{A,B}$ and $l_{A,\alpha}$, communication latency between A and B, and A and α

- Communicating process model for high availability methods
- Comparison/ranking of mean-time to recovery ($MTTR_{recovery}$)
  1. Hot/standby, asym. active/active with hot/standby, sym. active/active
  2. Warm/standby, asym. active/active with warm/standby
- Comparison/ranking of failure-free message latency overhead
  1. Warm/standby, asym. active/active with warm/standby
  2. Hot/standby, asym. active/active with hot/standby, sym. active/active
- Query load balancing in sym. active/active improves performance

External Symmetric Active/Active Replication Prototype

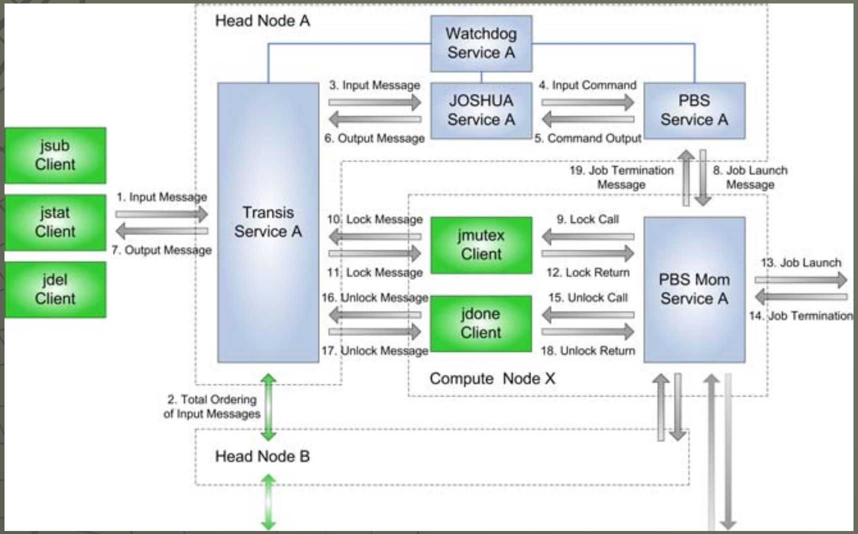
- Solution for the HPC job and resource manager
- Interceptors offer single virtual service to clients



Implemented by Kai Uhlemann, MSc student, The University of Reading

External Symmetric Active/Active Replication Prototype

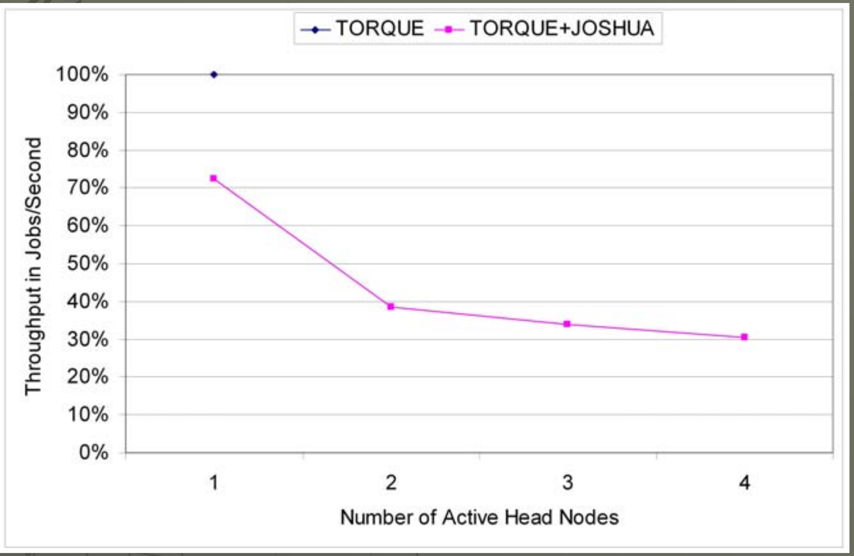
- Solution for the HPC job and resource manager
- Interceptors offer single virtual service to clients
- Implementation based on Transis and TORQUE



Implemented by Kai Uhlemann, MSc student, The University of Reading

External Symmetric Active/Active Replication Prototype

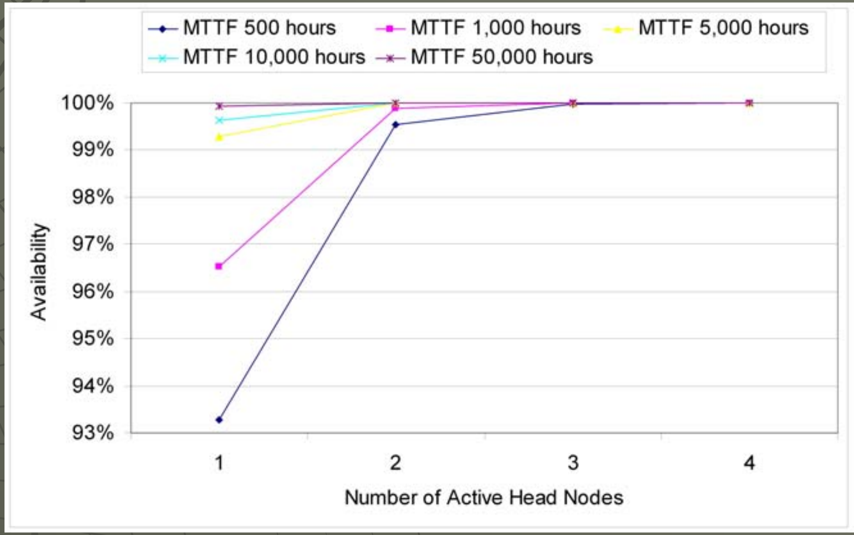
- Solution for the HPC job and resource manager
- Interceptors offer single virtual service to clients
- Implementation based on Transis and TORQUE
- Decent performance



Implemented by Kai Uhlemann, MSc student, The University of Reading

External Symmetric Active/Active Replication Prototype

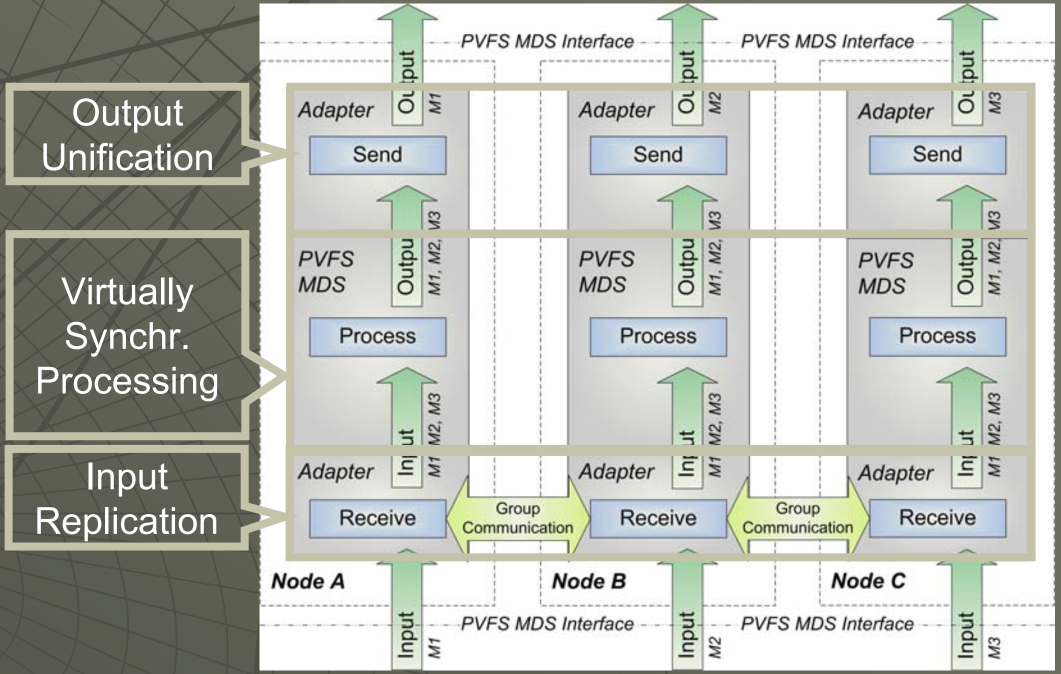
- Solution for the HPC job and resource manager
- Interceptors offer single virtual service to clients
- Implementation based on Transis and TORQUE
- Decent performance
- Significantly improves service availability
  - 1 node: <u>99</u>.3%
  - 2 nodes: <u>99.995</u>%
  - 3 nodes: <u>99.9999</u>6%



Implemented by Kai Uhlemann, MSc student, The University of Reading

Internal Symmetric Active/Active Replication Prototype

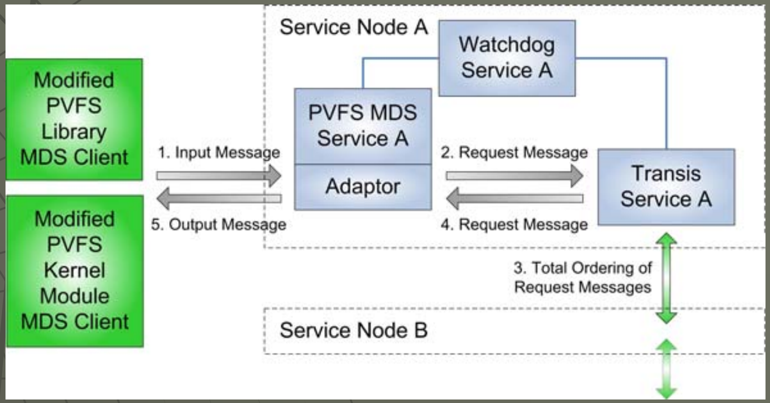
- Solution for the PVFS metadata service
- Adaptors offer single virtual service to clients



Implemented by Li Ou, PhD student, Tennessee Tech University

Internal Symmetric Active/Active Replication Prototype

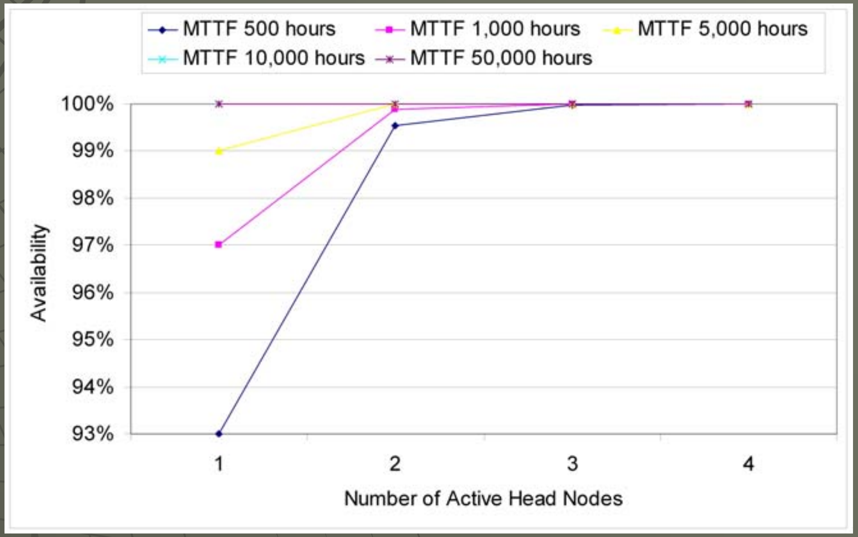
- Solution for the PVFS metadata service
- Adaptors offer single virtual service to clients
- Implementation based on improved Transis and PVFS



Implemented by Li Ou, PhD student, Tennessee Tech University

Internal Symmetric Active/Active Replication Prototype

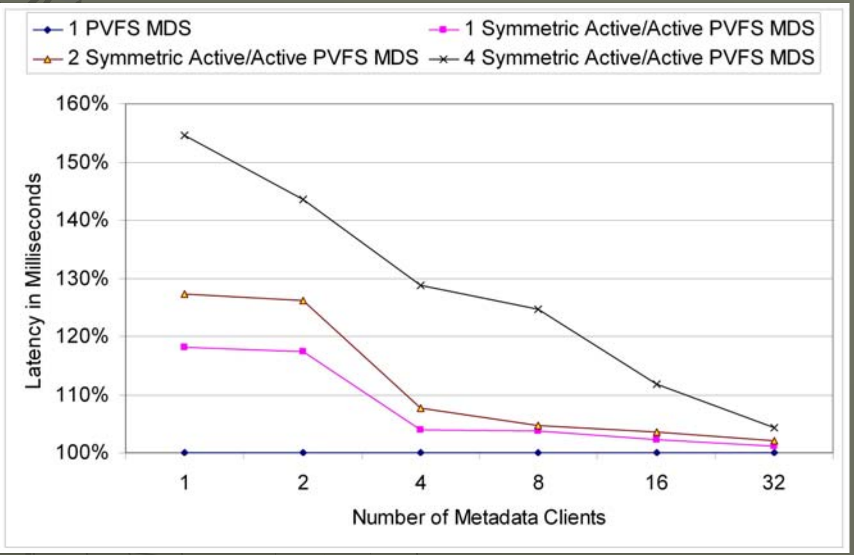
- Solution for the PVFS metadata service
- Adaptors offer single virtual service to clients
- Implementation based on improved Transis and PVFS
- Same high availability as external prototype



Implemented by Li Ou, PhD student, Tennessee Tech University

Internal Symmetric Active/Active Replication Prototype

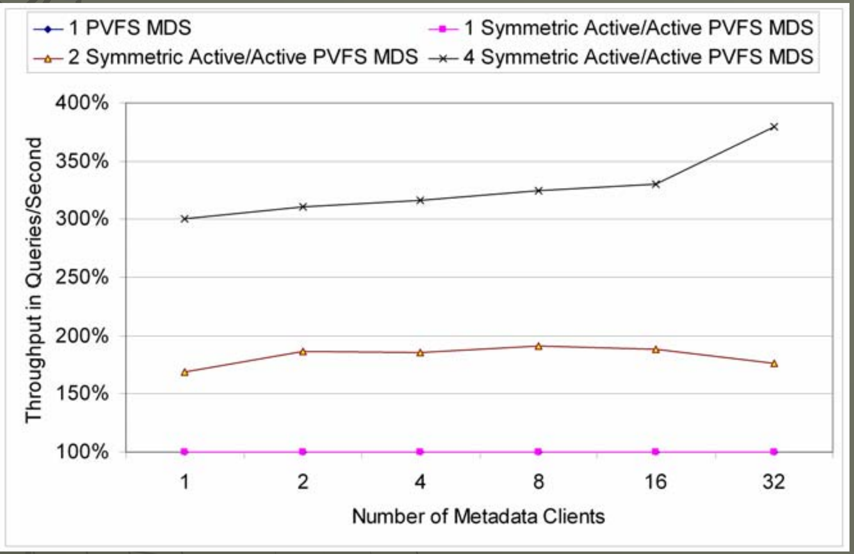
- Solution for the PVFS metadata service
- Adaptors offer single virtual service to clients
- Implementation based on improved Transis and PVFS
- Same high availability as external prototype
- Improved performance
  - 2-26ms latency overhead



Implemented by Li Ou, PhD student, Tennessee Tech University

Internal Symmetric Active/Active Replication Prototype

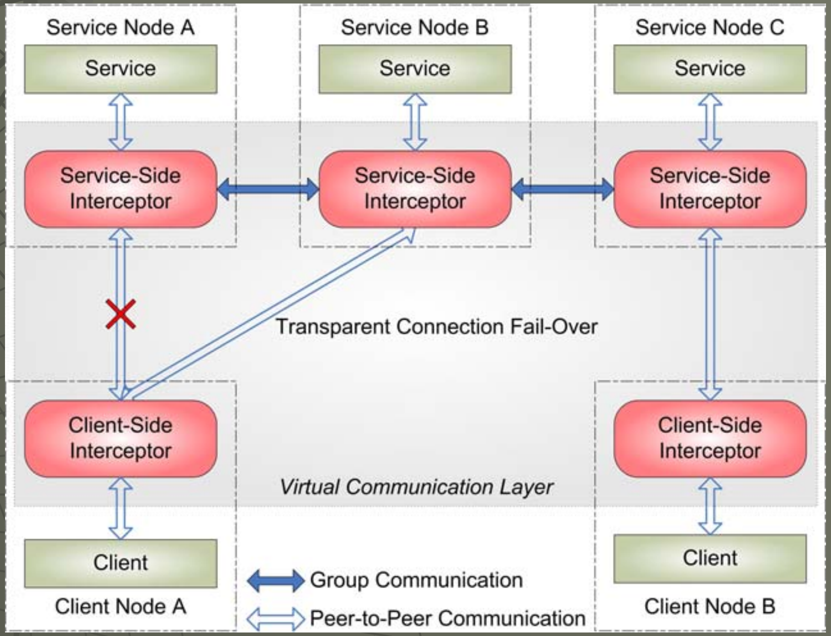
- Solution for the PVFS metadata service
- Adaptors offer single virtual service to clients
- Implementation based on improved Transis and PVFS
- Same high availability as external prototype
- Improved performance
  - 2-26ms latency overhead
  - Up to 380% read (query) throughput improvement



Implemented by Li Ou, PhD student, Tennessee Tech University

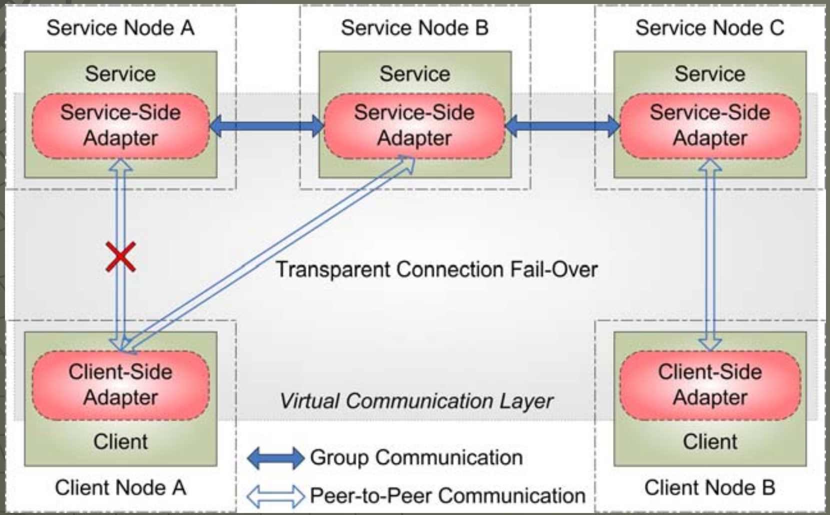
Transparent Symmetric Active/ Active Replication Framework

- Transparent replication framework that improves
  - Reuse of code
  - Ease of use
- Additional client-side
  - Interceptors



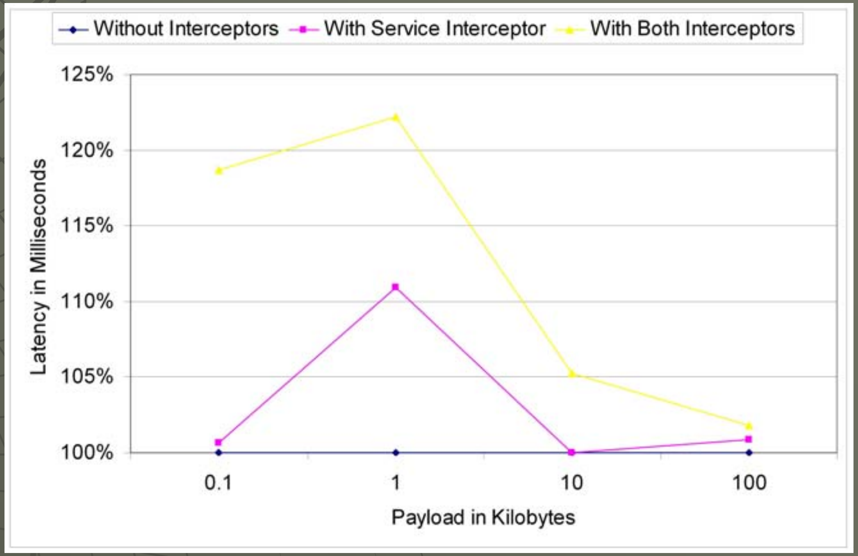
Transparent Symmetric Active/ Active Replication Framework

- Transparent replication framework that improves
  - Reuse of code
  - Ease of use
- Additional client-side
  - Interceptors
  - Adaptors



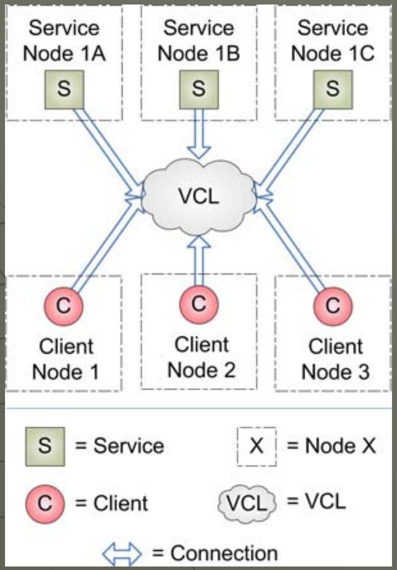
Transparent Symmetric Active/ Active Replication Framework

- Transparent replication framework that improves
  - Reuse of code
  - Ease of use
- Additional client-side
  - Interceptors
  - Adaptors
- Performance hit for client-side interceptors



Transparent Symmetric Active/Active Replication Framework

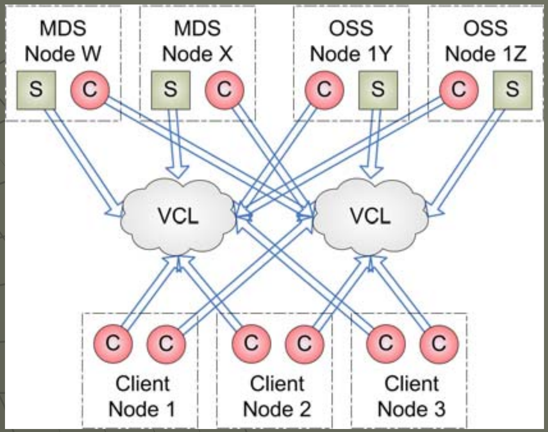
- Transparent replication framework that improves
  - Reuse of code
  - Ease of use
- Additional client-side
  - Interceptors
  - Adaptors
- Performance hit for client-side interceptors
- High-level abstraction for
  - Client/service scenarios



Result of failed attempt to provide symmetric active/active high availability for the Lustre metadata service by Matthias Weber, MSc student, University of Reading

Transparent Symmetric Active/ Active Replication Framework

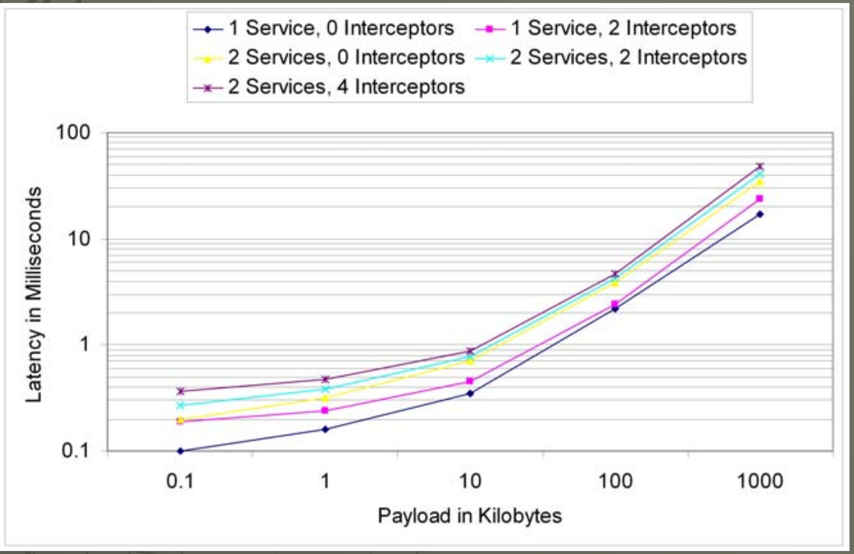
- Transparent replication framework that improves
  - Reuse of code
  - Ease of use
- Additional client-side
  - Interceptors
  - Adaptors
- Performance hit for client-side interceptors/adaptors
- High-level abstraction for
  - Client/service scenarios
  - Dependent services



Result of failed attempt to provide symmetric active/active high availability for the Lustre metadata service by Matthias Weber, MSc student, University of Reading

Transparent Symmetric Active/ Active Replication Framework

- Transparent replication framework that improves
  - Reuse of code
  - Ease of use
- Additional client-side
  - Interceptors
  - Adaptors
- Performance hit for client-side interceptors/adaptors
- High-level abstraction for
  - Client/service scenarios
  - Dependent services
  - Serial VCL performance hit



Result of failed attempt to provide symmetric active/active high availability for the Lustre metadata service by Matthias Weber, MSc student, University of Reading

Contribution Summary

1. Modern service-level high availability taxonomy that removes ambiguities and includes state-machine replication
2. Identification of availability deficiencies in modern HPC systems that clarifies node and service failure impact
3. Unified definition of service-level high availability methods that allows for availability and performance comparison
4. External symmetric active/active replication prototype for a HPC job and resource manager with 99.99% availability
5. Internal symmetric active/active replication prototype for a HPC file system metadata service with high performance
6. Symmetric active/active replication framework prototypes with completely transparent client/service interfaces

Future Work

- Development of a production-type symmetric active/active replication framework
- Development of production-type high availability support for HPC system services
- Extending the framework to support active/standby and asymmetric active/active
- Extending the concepts and prototypes to other service-oriented or -dependent architectures
- Extending the concepts and prototypes for modular redundancy for HPC compute nodes

Publications

- 2 journal papers
  - JCP, OSR
- 1 journal paper still under review
  - JPDC
- 6 conference papers
  - 2 ARES, PDCS, ICCCN, Cluster, ICCSIS
- 7 workshop papers
  - 2 CCGrid, ICCS, 2 LACSI, ARES, ICS
- 3 co-advised/-supervised theses
  - 2 Reading MSc theses, TTU PhD thesis

Acknowledgements

- Students
  - Kai Uhlemann (Reading MSc)
  - Li Ou (TTU PhD)
  - Matthias Weber (Reading MSc)
- Funding
  - U.S. Department of Energy
    - Office of Science, FAST-OS Program
  - Oak Ridge National Laboratory
    - Laboratory Directed R&D Program

Symmetric Active/Active High Availability for High-Performance Computing System Services

Christian Engelmann

The University of Reading, UK
Oak Ridge National Laboratory, USA