Modular Redundancy in HPC Systems: Why, Where, When and How?

Christian Engelmann

Computer Science and Mathematics Division
Oak Ridge National Laboratory

HPC Resiliency Summit: Workshop on Resiliency for Petascale HPC, Los Alamos Computer Science Symposium, Santa Fe, NM, 10/15/2008

Outline

- **Background and motivation**
  - **Trends in HPC system reliability and resilience**
  - **Resulting motivation for modular redundancy in HPC**
- **Technical approach**
  - **Traditional modular redundancy**
  - **Hurdles for modular redundancy in HPC**
  - **Extending symmetric active/active replication**
- **Conclusions**

Trends in HPC System Reliability

- **HPC systems continue to increase in size**
  - **Error rate increases due to higher component count**
- **HPC systems may increasingly contain accelerators**
  - **Soft error rate increases due to higher vulnerability**
- **Nanometer technology continues to decrease**
  - **Soft error rate increases further due to higher vulnerability**
- **HPC vendors continue to use mass-market components**
  - **Mass-market demands define HPC system reliability**

➲ ***Future HPC systems won't be as reliable as today's***

➲ ***Soft errors are a major concern for HPC resilience***

Trends in HPC System Resilience

- **Checkpoint/restart has limits**
  - **Efficiency decreases with higher error rate**
  - **Efficiency decreases further with larger aggregated memory**
  - **Incremental/compression approaches help in the short term**
  - **Preemptive migration helps further in the long term**
- **Preemptive migration has also limits**
  - **Error rate increases with lower prediction accuracy**
  - **Errors without precursor or pattern can't be predicted**
    - **Can anyone predict a non-recoverable ECC memory error?**

➲ ***Future HPC systems won't be as resilient as today's***

➲ ***Resiliency strategy for high soft error rates is missing***

Motivation for Modular Redundancy in HPC

- **Redundancy on compute nodes is not entirely new**
  - **Diskless checkpointing (Plank et al.)**
  - **Algorithmic redundancy approaches (Dongarra et al.)**
- **Until now, the HPC community (researchers and vendors) stayed away from modular redundancy**
  - **"Big hammer" approach with fully redundant compute nodes**
- ➲ ***With increasing hard and (especially) soft error rates, compute-node redundancy needs to be considered as an alternative to checkpointing and preemptive migration***
- ➲ ***Respective research and development in modular redundancy for HPC environments is needed***

Outline

- Background and motivation
  - Trends in HPC system reliability and resilience
  - Resulting motivation for modular redundancy in HPC
- **Technical approach**
  - **Traditional modular redundancy**
  - **Hurdles for modular redundancy in HPC**
  - **Extending symmetric active/active replication**
- Conclusions

Traditional Modular Redundancy

- **Dual-modular redundancy (DMR) offers protection against hard errors and some soft errors**
- **Triple-modular redundancy (TMR) offers protection against hard and soft errors**
- **Both have been used in many mission critical systems**
- **2x or 3x requirement in hardware investment**
- **Additional replication hard- and/or software needed**

➲ ***Traditional DMR and TMR offer resiliency at 2x or 3x costs***

➲ ***They provide a resiliency strategy for high soft error rates***

Hurdles for Modular Redundancy in HPC

- **The primary issue for DMR and TMR in HPC is costs**
  - **2x or 3x the number of computational resources required**
- **Another problem is the appearance of wasted resources**
  - **Using only 50% or 33% of provided computational capability**
  - **While this may be really only the inverse of the prior point, resiliency requirements are use-case dependent**
    - **Not every job runs longer than its MTTF**
- **Increased power consumption is an issue as well**

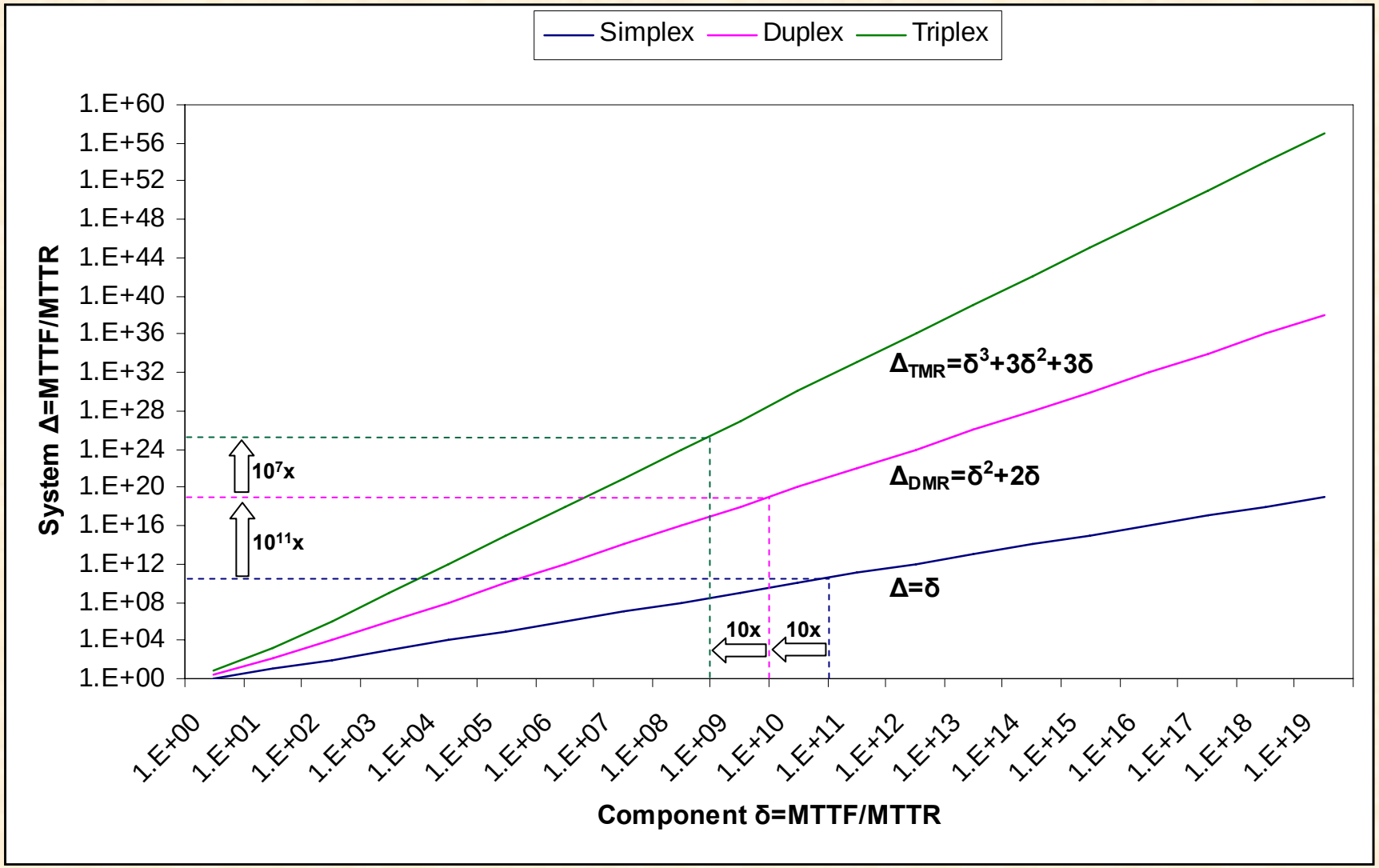
➲ ***DMR and TMR in HPC make only sense at lower costs***

➲ ***Flexible solution needed for individual job requirements***

Reducing Modular Redundancy Costs

- **Modular redundancy can be done at less than 2x/3x costs**
- **DMR and TMR drastically increase reliability by x^2 and x^3**
  - **Next slide will explain this in more detail**
- **Less reliable components (processor and memory) can offset costs and even improve performance**
  - **Server/embedded vs. desktop/gaming/mobile processors**
  - **ECC vs. non-ECC memory (4GB 800MHz DDR2: $300 vs. $50)**

System MTTF Impact of Varying Component MTTF/MTTR in Modular Redundancy Systems



Providing Flexible Modular Redundancy

- **On-demand capability**
  - **No redundancy, DMR or TMR based on job requirements**
  - **No modular redundancy for short running jobs (< AMTTF)**
  - **Application-supported DMR for long running jobs**
  - **Fully-transparent TMR for long running jobs**
- **Different modular redundancy granularity across**
  - **Cores within a processor**
  - **Processors or cores within a node**
  - **Processors, cores, or nodes within the system**
- **Fast recovery through replica cloning**

➲ ***System software solution is needed to provide flexibility***

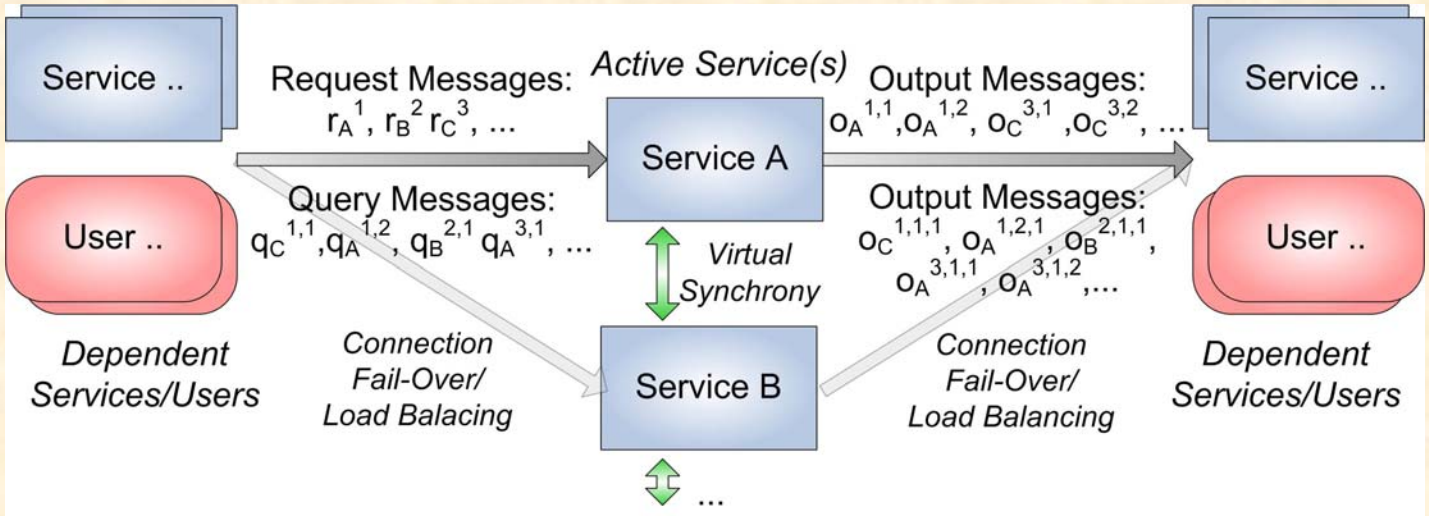
Proposed Technical Approach

- **From a distributed computing perspective**
  - **Each (MPI) process is a networked service**
  - **All (MPI) processes (services) depend on each other**
  - ➲ ***Modular redundancy at the (MPI) process level is needed***
- **From a parallel computing perspective**
  - **Communication and computation performance is crucial**
  - ➲ ***High-performance (MPI) process-level replication is needed***

➲ ***Our recently developed symmetric active/active service replication offers high-performance process replication***

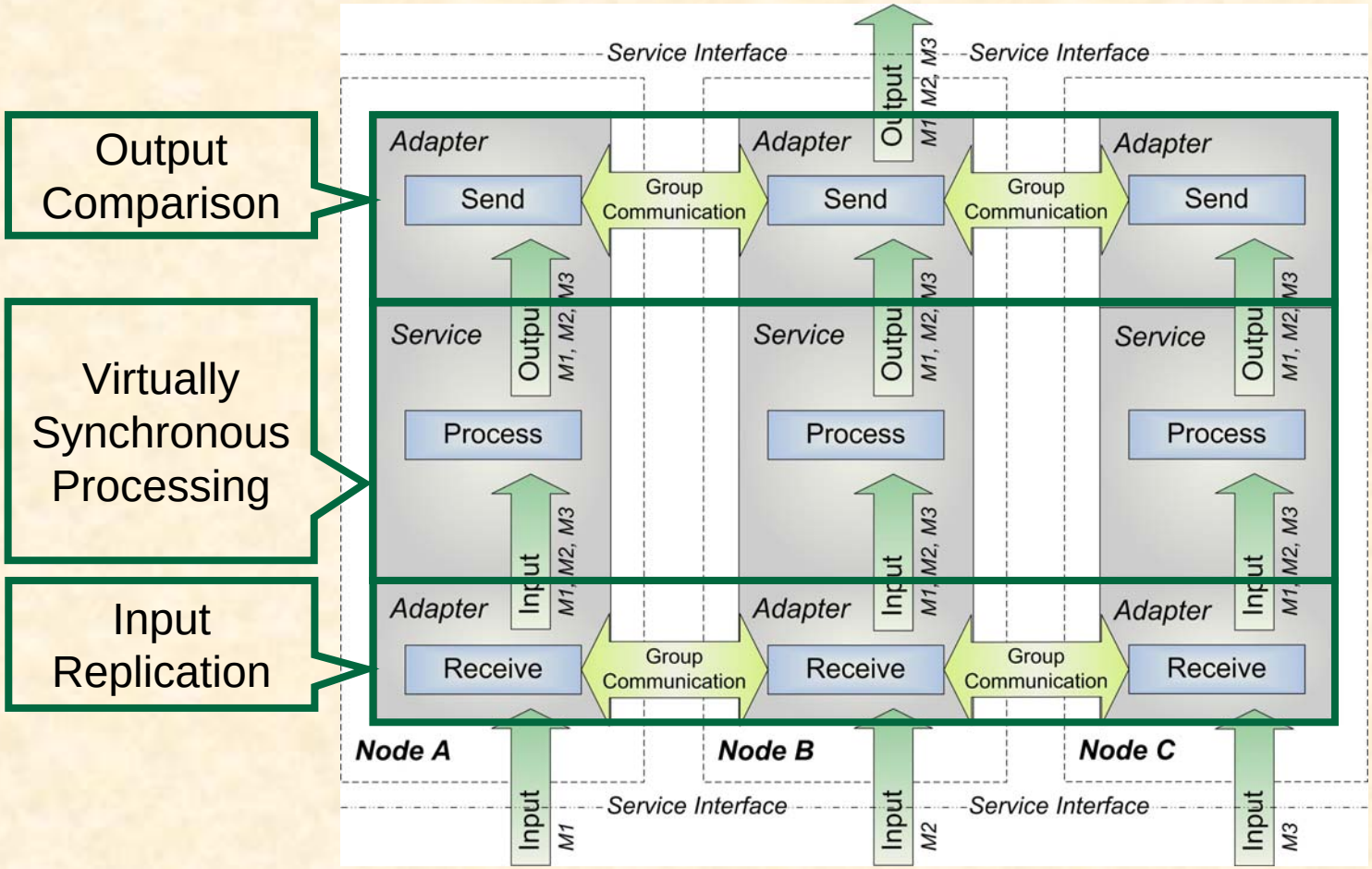
➲ ***Symmetric active/active replication with DMR and TMR features can provide flexible modular redundancy for HPC***

From Symmetric Active/Active Service Replication To Modular Redundancy in HPC



- **Symmetric active/active replication for service processes**
  - **Virtual synchrony (state-machine replication) model**
  - **Replication of input messages & unification of output messages**
- **Extending symmetric active/active replication to modular redundancy**
  - **Comparison of output messages to detect/correct soft errors**
  - **Fast error recovery through replica cloning**

Modular Redundancy using Symmetric Active/Active Replication

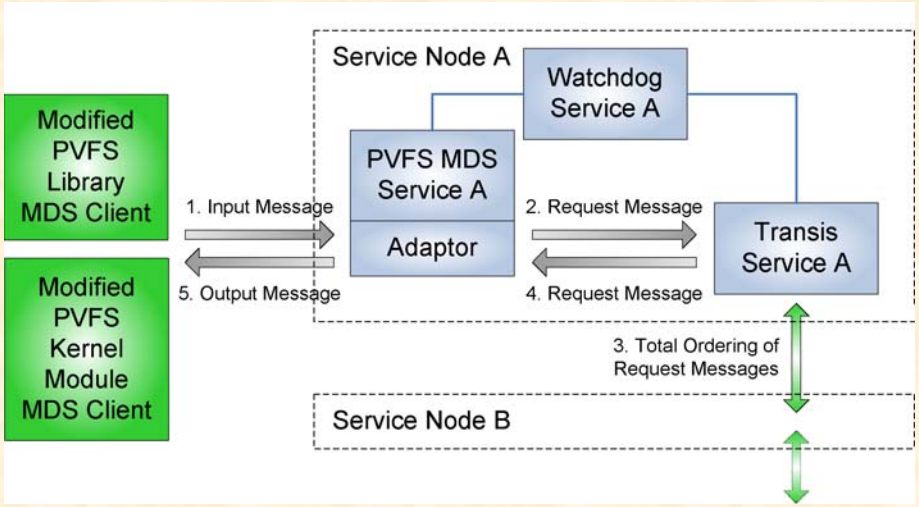


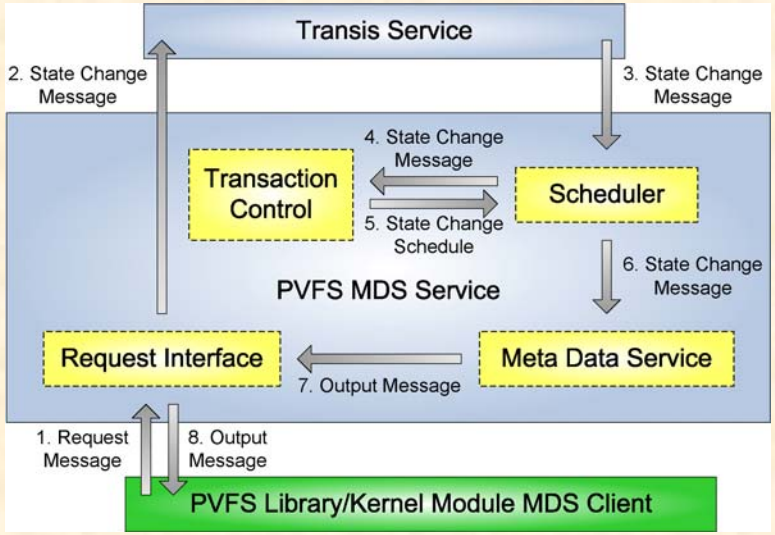
Comparison of Replication Methods

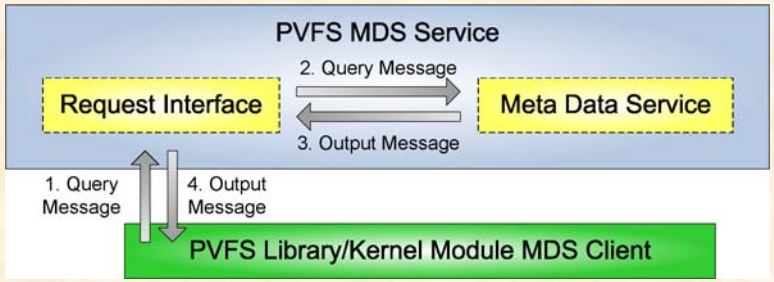
Method	$MTTR_{recovery}$	Latency Overhead
Warm-Standby	$T_d + T_f + T_r + T_c$	0
Hot-Standby	$T_d + T_f + T_r$	$2l_{A,B}$, $O(log_2(n))$, or worse
Asymmetric with Warm-Standby	$T_d + T_f + T_r + T_c$	0
Asymmetric with Hot-Standby	$T_d + T_f + T_r$	$2l_{A,\alpha}$, $O(log_2(n))$, or worse
Symmetric	$T_d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

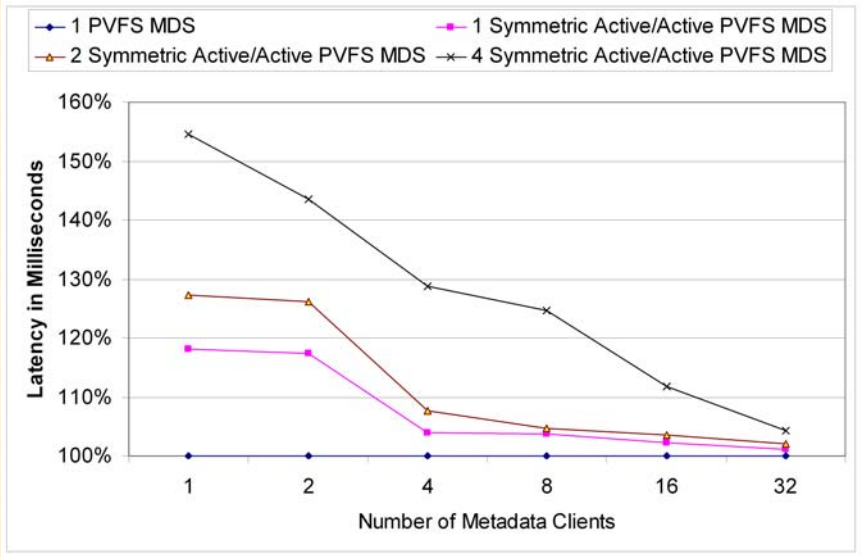
Symmetric Active/Active Replication Example: PVFS Metadata Service (Design)

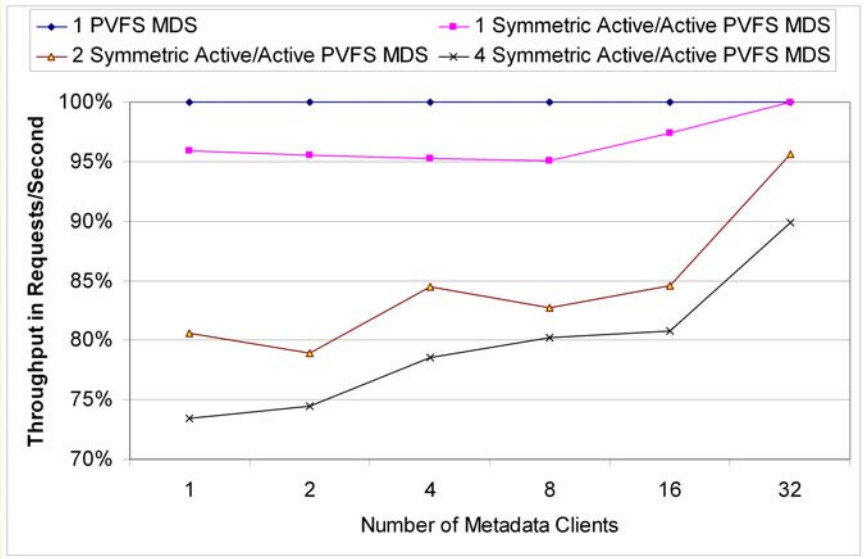


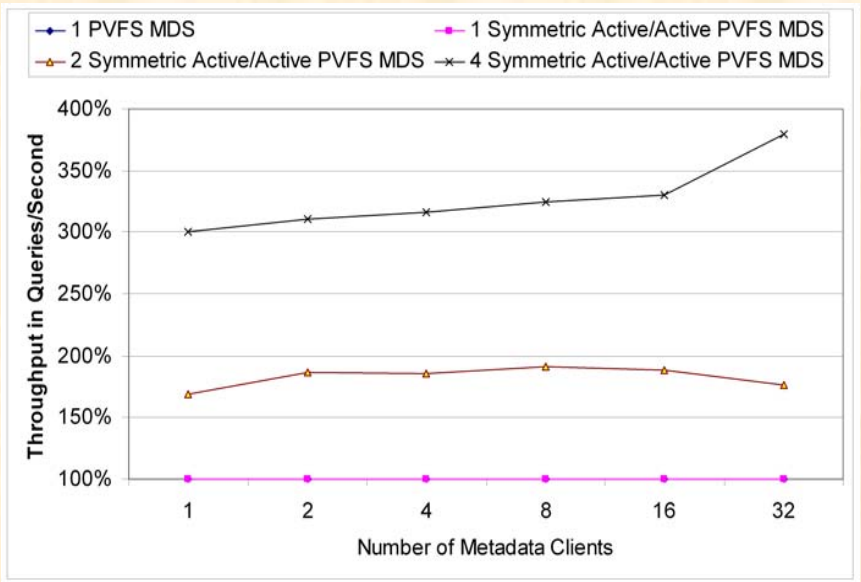


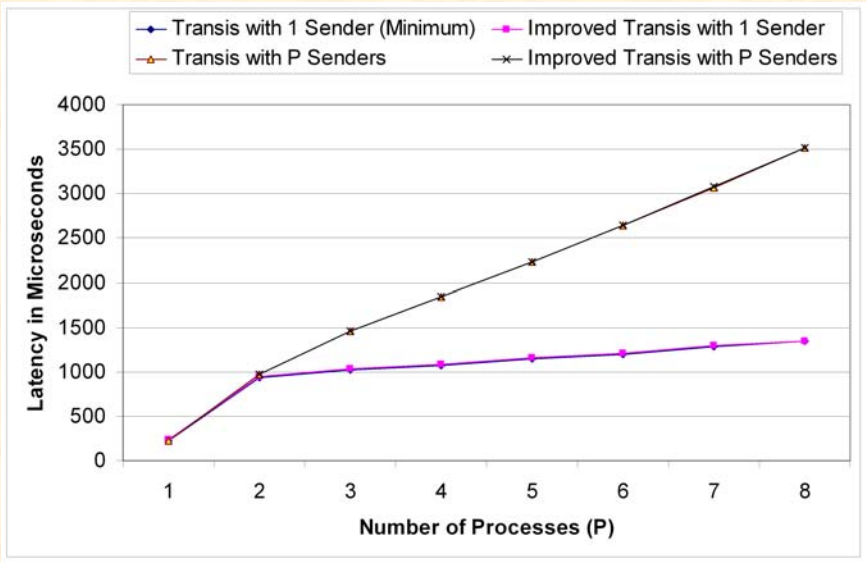


Symmetric Active/Active Replication Example: PVFS Metadata Service (Results)

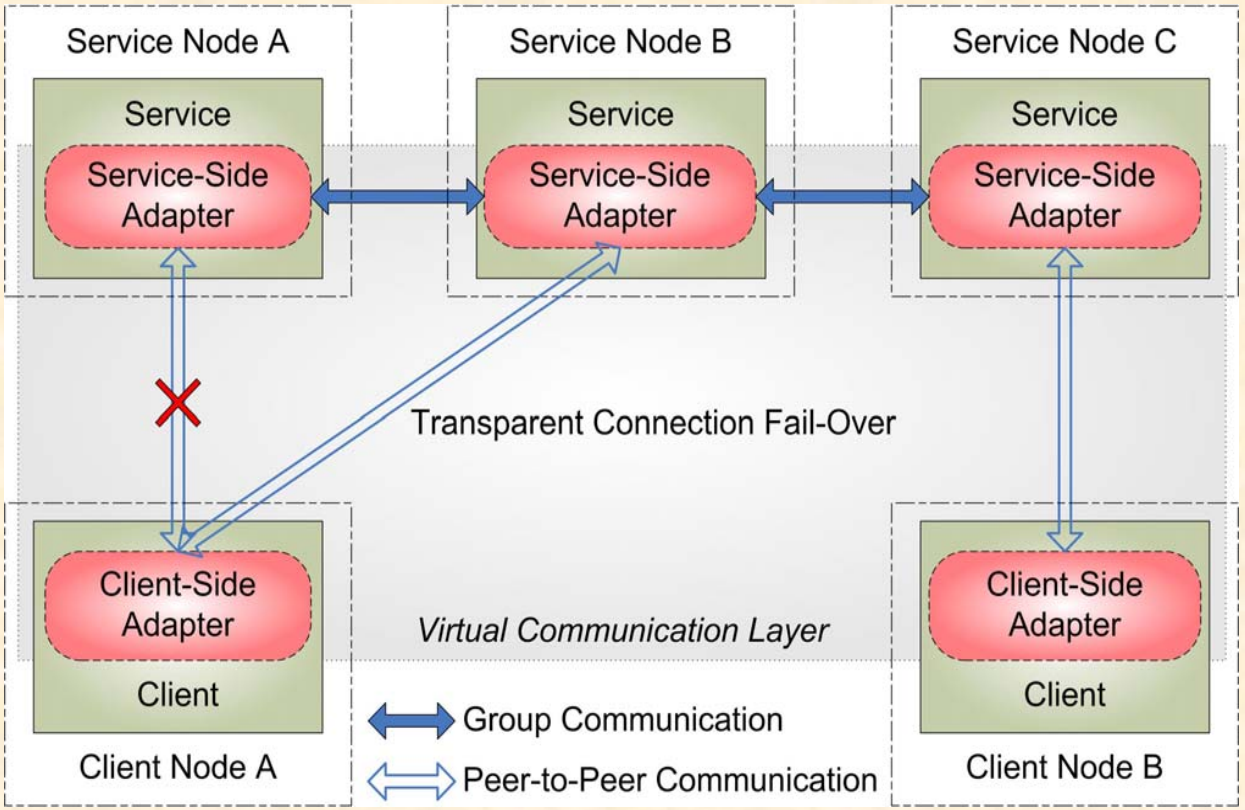




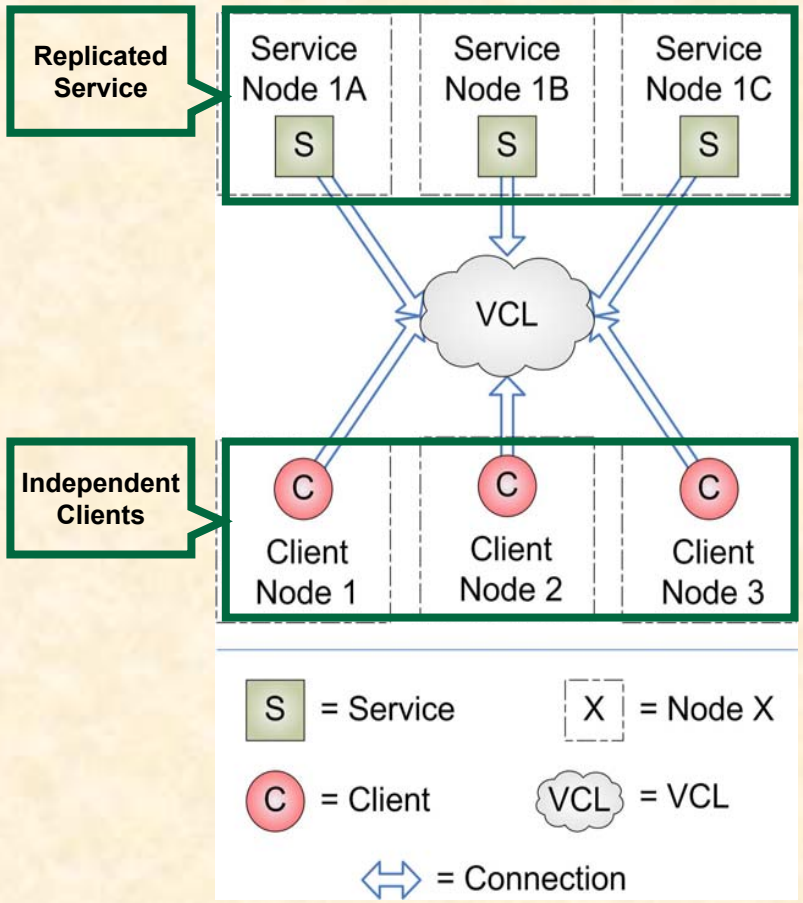




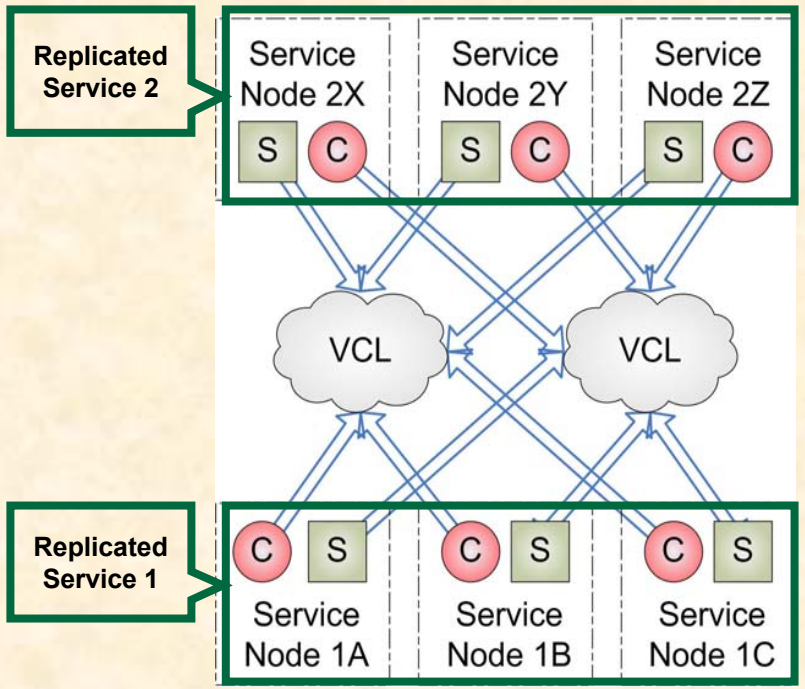
Transparent Symmetric Active/Active Replication for Client/Service Scenarios



Symmetric Active/Active Replication Abstraction for Client/Service Scenarios



Symmetric Active/Active Replication Abstraction for Service/Service Scenarios



Two Replicated Interdependent Services = Two Replicated Compute Nodes

Future Work (Pending Funding)

- Development of high-performance DMR/TMR algorithms
- Implementation of DMR/TMR proof-of-concept prototypes
- Implementation of an analysis and testing framework
- Failure injection analysis and testing of prototypes
- Model the reliability of DMR/TMR prototypes
- Investigate asymmetric replica oversubscription
  - DMR: 66% compute nodes with one replica, 33% with two
  - TMR: 50% compute nodes with one replica, 50% with two
  - Regular replica updates to avoid falling behind too much

Outline

- Background and motivation
  - Trends in HPC system reliability and resilience
  - Resulting motivation for modular redundancy in HPC
- Technical approach
  - Traditional modular redundancy
  - Hurdles for modular redundancy in HPC
  - Extending symmetric active/active replication
- **Conclusions**

Conclusions

- **HPC resiliency is an ongoing research effort**
- **Checkpoint/restart has room for improvement, but also a final limitation on the error rate it can handle**
- **Preemptive migration can complement checkpoint/restart to improve efficiency, but not all errors are predictable**
- **With the rising rate of unpredictable soft errors, modular redundancy concepts may offer an alternative**
- **Research and development in flexible high-performance modular redundancy for HPC environments is needed**

UT-BATTELLE

Questions?