SSD-Assisted Designs for MPI Fault-Tolerance and Accelerating Web 2.0

Presentation at OFA Developer Workshop (2013)

by

Dhabaleswar K. (DK) Panda

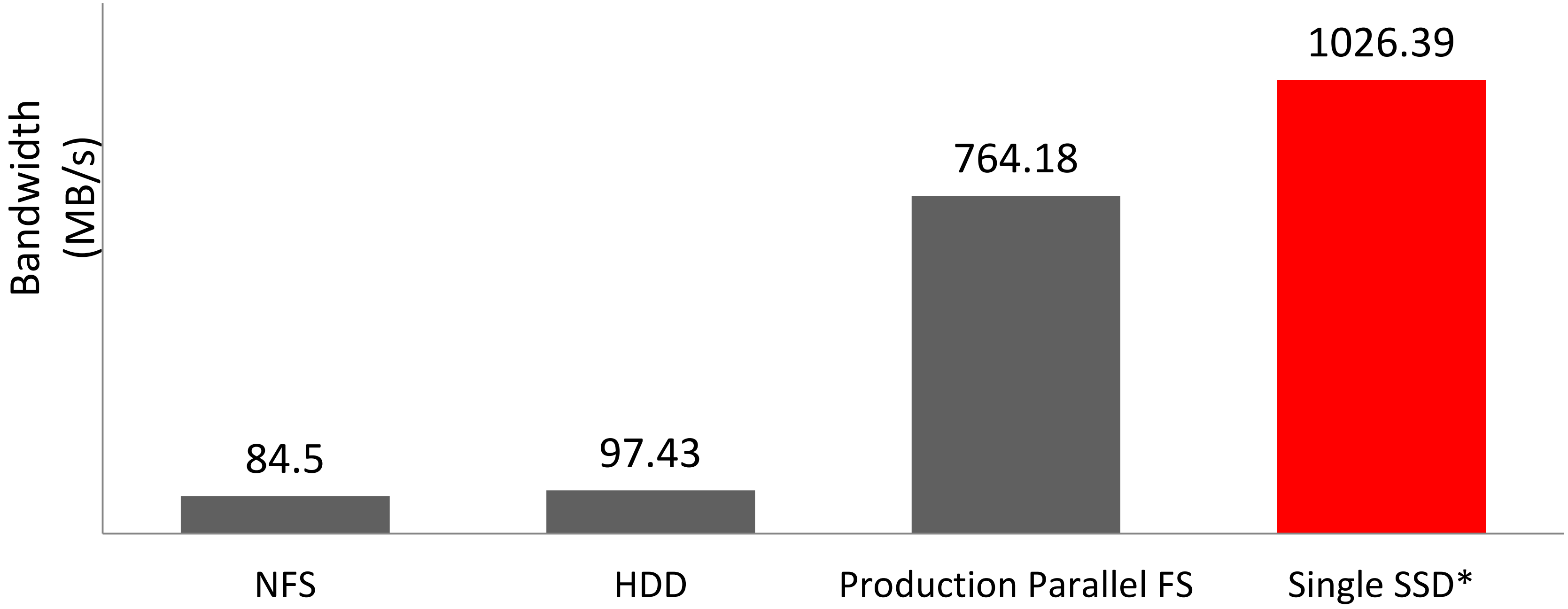
The Ohio State University

E-mail: panda@cse.ohio-state.edu

http://www.cse.ohio-state.edu/~panda

Introduction

- SSD technology is improving steadily
- Significant performance benefits with PCIe-based SSDs



* Fusion-iO ioDrive PCI-e SSD

Open Challenges

- How do SSDs change the landscape of designing Next generation systems (scientific and enterprise)
- Can RDMA be utilized together with SSDs: Challenges and Benefits?

Experience in Using SSDs

- Use of SSDs to accelerate MPI Fault-Tolerance
  - Accelerating checkpoint-restart and migration with hierarchical data-staging and high-throughput SSDs
  - Multi-Level checkpointing using SSDs with Scalable Checkpoint/Restart (SCR)
- Use of SSDs to accelerate Web 2.0
  - Using SSDs as a Virtual Memory Swap (existing naïve solution)
  - Accelerating Memcached with a SSD-based Hybrid-Memory architecture

Process-level Fault-Tolerance

- High probability of component failures in large-scale systems
- Long-running applications should continue to execute
- Broad approaches for process-level fault -tolerance
  - Transparent checkpoint-restart
    - Periodically store checkpoint (memory footprint of all processes)
    - In case of failures, go back to the last checkpoint and restart
  - Proactive migration
    - Monitor nodes for failure symptoms
    - With high-probability of impending failures , migrate the process to a spare node and continue execution
  - Applications-level fault-tolerance
    - Applications periodically store the main results (at the end of an iteration)
    - Restart application from results from a previous iteration in case of failure

Recap: MVAPICH2/MVAPICH2-X Software

- High Performance open-source MPI Library for InfiniBand, 10Gig/iWARP and RDMA over Converged Enhanced Ethernet (RoCE)
  - MVAPICH (MPI-1) ,MVAPICH2 (MPI-3.0), Available since 2002
  - MVAPICH2-X (MPI + PGAS), Available since 2012
  - Used by more than 2,000 organizations (HPC Centers, Industry and Universities) in 70 countries
  - More than 165,000 downloads from OSU site directly
  - Empowering many TOP500 clusters
    - 7th ranked 204,900-core cluster (Stampede) at TACC
    - 14th ranked 125,980-core cluster (Pleiades) at NASA
    - 17th ranked 73,278-core cluster (Tsubame 2.0) at Tokyo Institute of Technology
    - and many others
  - Available with software stacks of many IB, HSE and server vendors including Linux Distros (RedHat and SuSE)
  - http://mvapich.cse.ohio-state.edu
- Partner in the U.S. NSF-TACC Stampede (9 PFlop) System

Process-Level and Applications-Level Fault-Tolerance in MVAPICH2

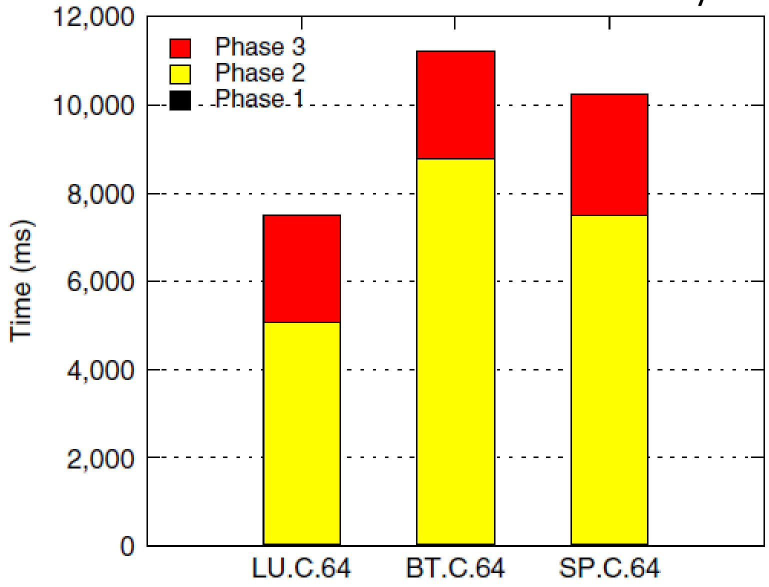
- Transparent Checkpoint-Restart
  - Basic Checkpoint-Restart scheme
  - Node-level Checkpoint write-aggregation scheme
- Proactive Process-Migration
  - File-copy bases process snapshot migration
  - RDMA-based pipelined process migration
- Applications-level Checkpointing with SCR

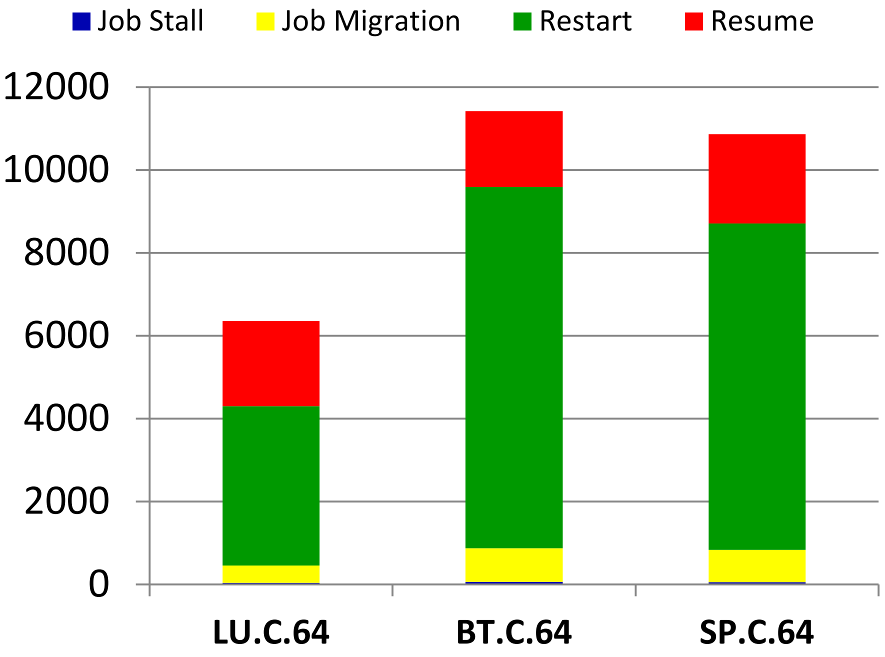
I/O Overheads with Checkpoint-Restart and Process-Migration

Phase 1: Suspend network activity

Phase 2: Write checkpoints

Phase 3: Resume network activity

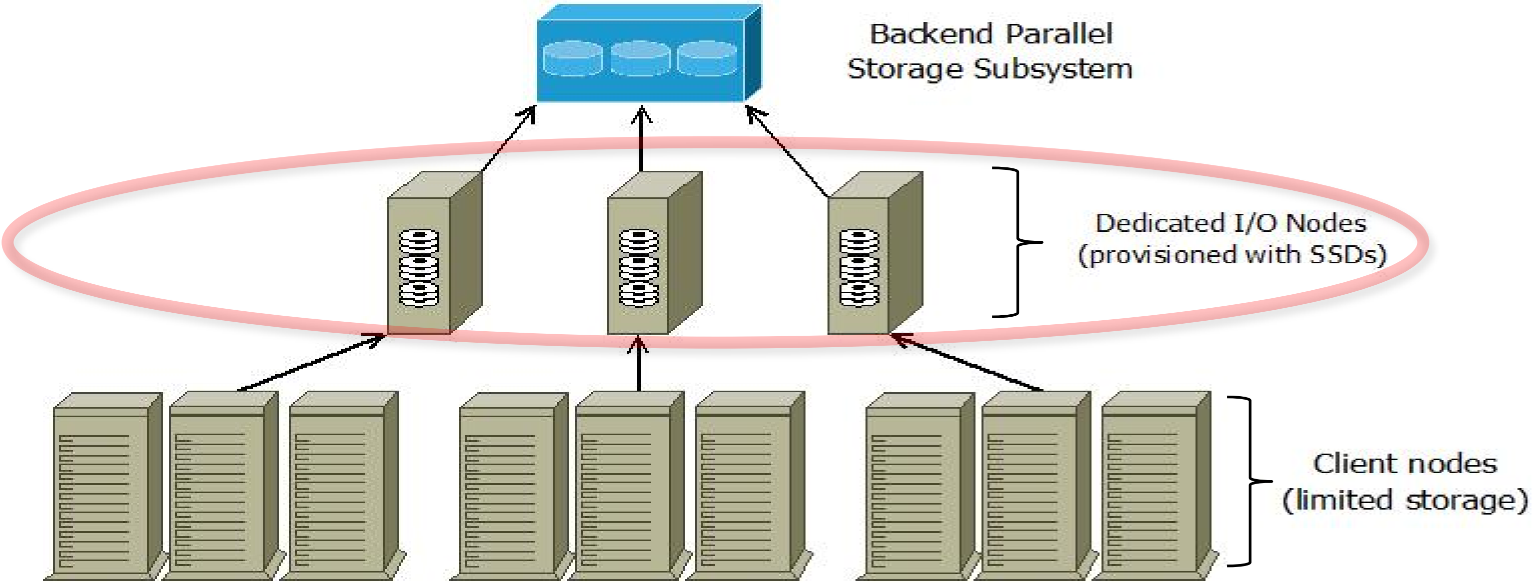




- Checkpoint writing phase is the most time-consuming
- Restarting a job after file-copy based process migration is the most time-consuming
- Both solutions can benefit from high-throughput write and read operations of SSDs

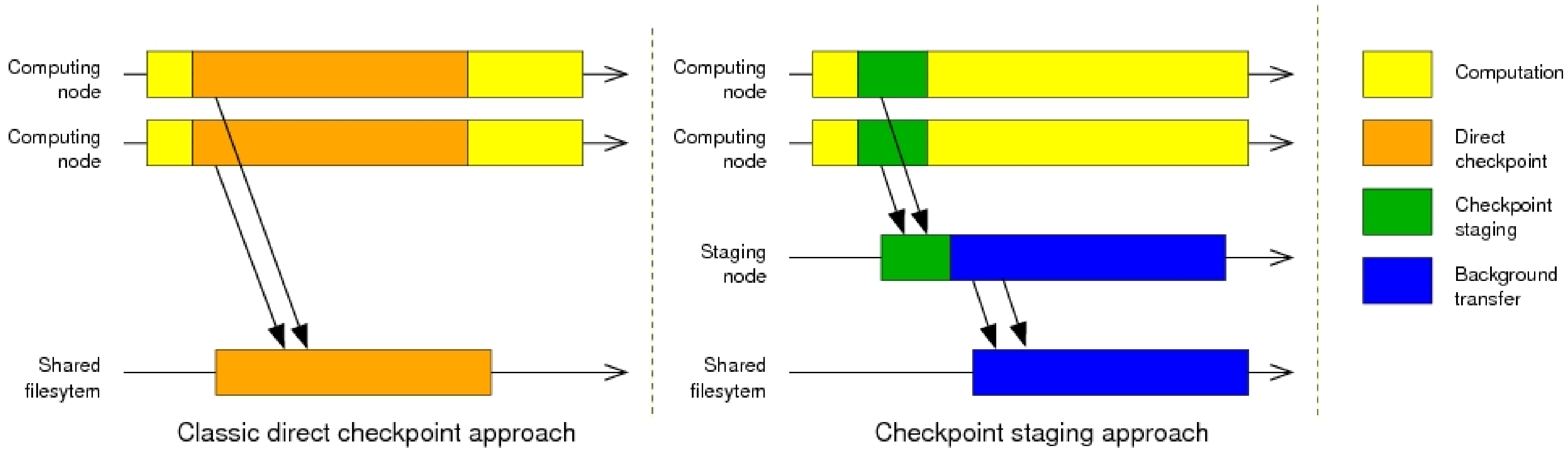
But SMARTER solutions are needed to leverage the inherent benefits of SSDs!

Hierarchical Data Staging Servers



- Compute nodes that are diskless/ with limited storage in terms of space
- Dedicated I/O nodes with SSDs can be placed in-between that orchestrate data transfer between compute nodes and parallel file system
- A few such nodes per rack

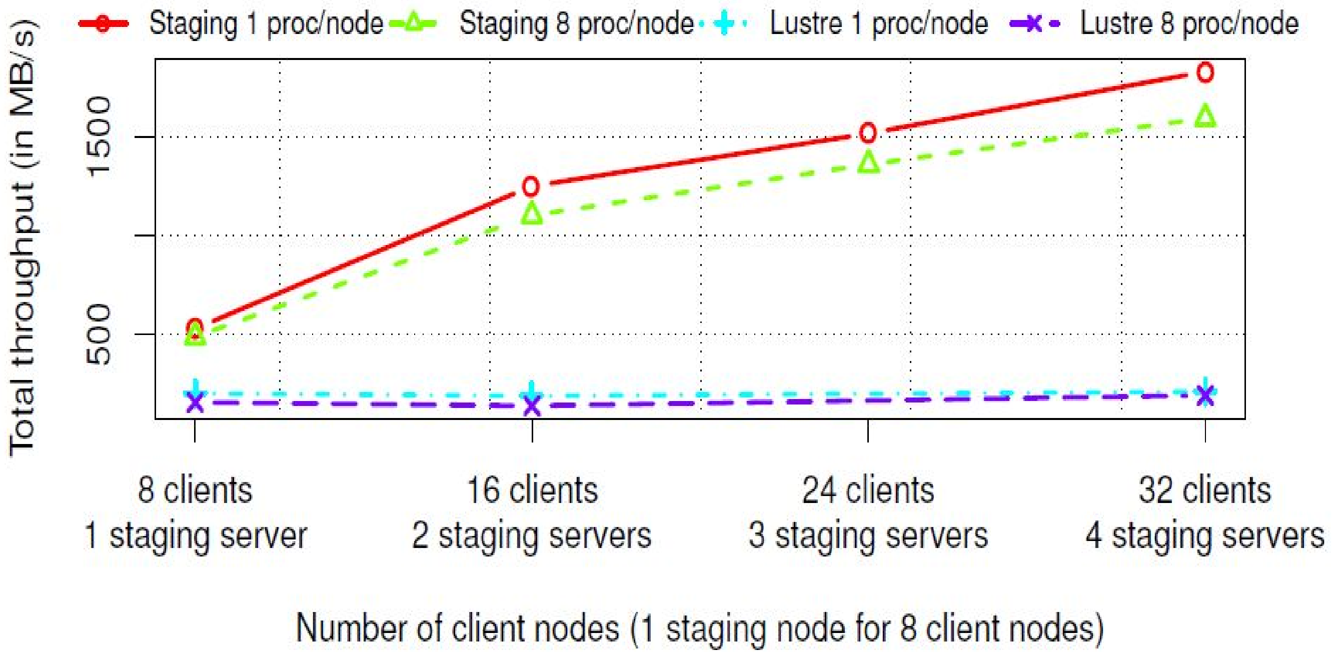
Using Dedicated Staging Servers



- Checkpoint files are written to staging servers
- Application resumes as soon as the data is written to the staging server
- Checkpoint files are transferred in background to the back-end filesystem
- Computation and data transfer are overlapped
- Checkpoint files eventually reach the backend file systems

Scalability of Hierarchical Data Staging

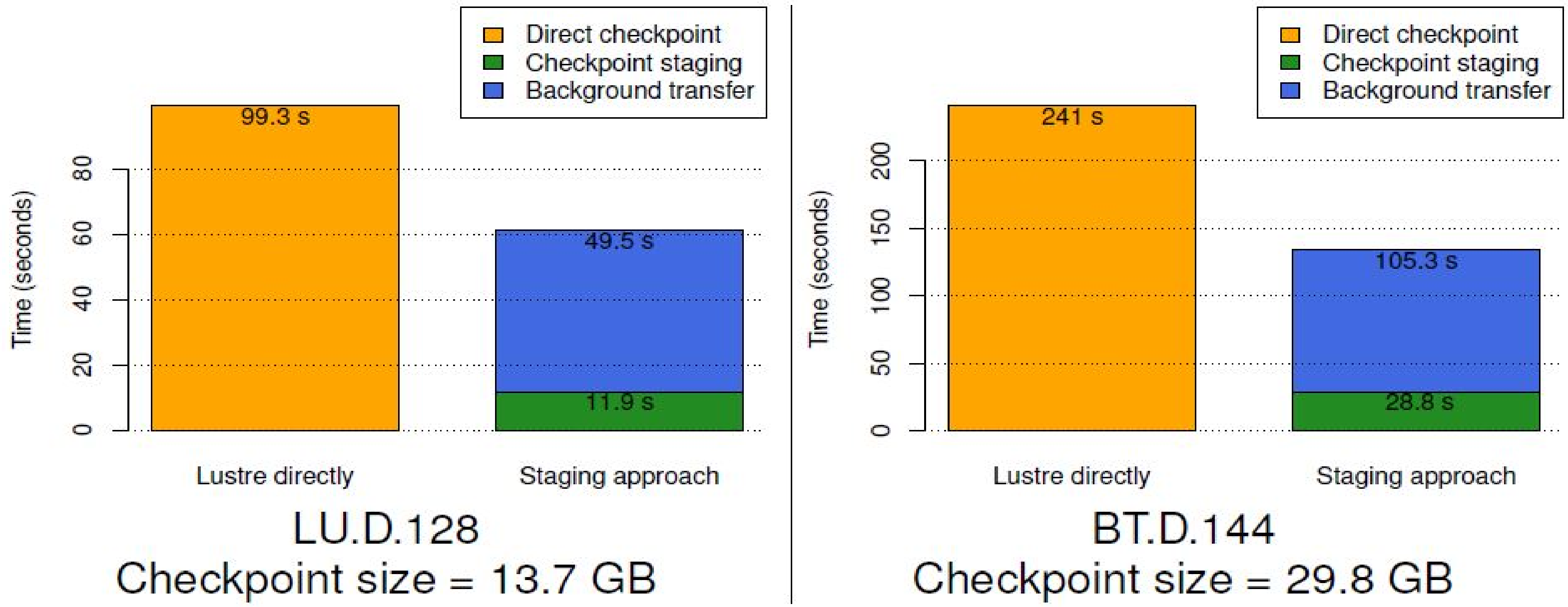
IO throughput with increasing number of staging servers (IOZone benchmark)



- Each process writes 1 GB with a 1 MB record size
- Staging architecture scales as the staging groups are increased
- Achieved aggregated throughput: **1,834 MB/s**
- Theoretical aggregated write throughput of all SSDs: **1,900 MB/s***

*OCZ RevoDrive PCI-e SSD

Evaluation with Applications (NAS Benchmarks)

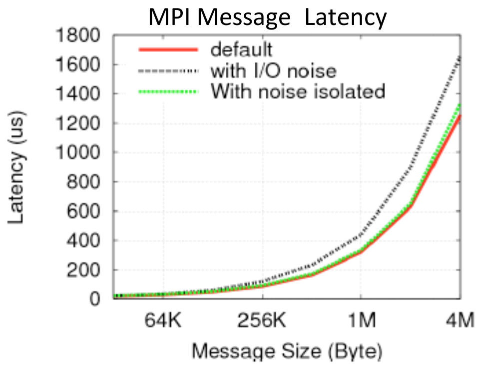


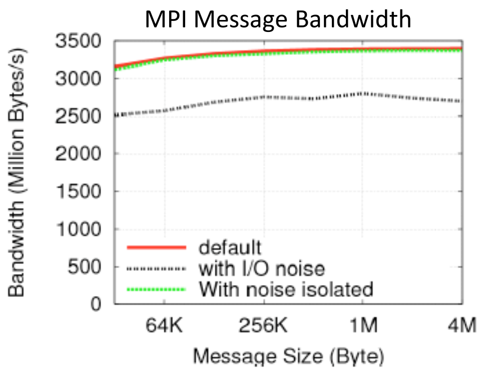
- Background transfer time is lesser than direct checkpointing time due to reduced contention on the Parallel filesystem
- Checkpointing time, as seen by the application, is **8.3** times lesser with the staging approach

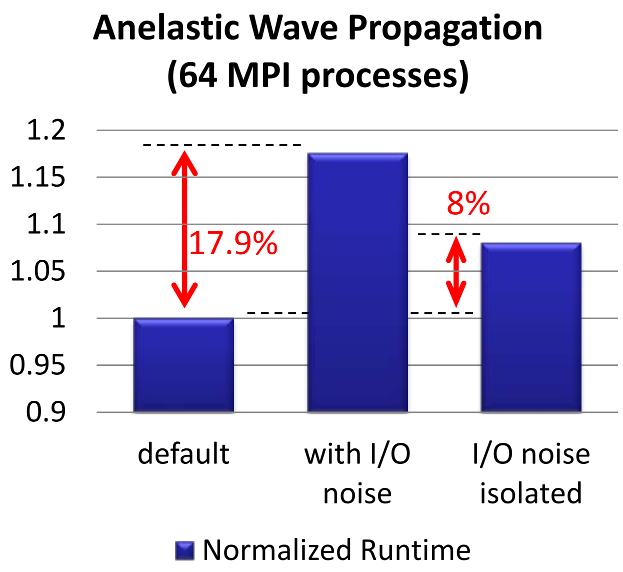
R. Rajachandrasekar, X. Ouyang, X. Besseron, V. Meshram and D. K. Panda, Can Checkpoint/Restart Mechanisms Benefit from Hierarchical Data Staging? Workshop on Resiliency in High Performance Computing in Clusters, Clouds, and Grids (Resilience '11)

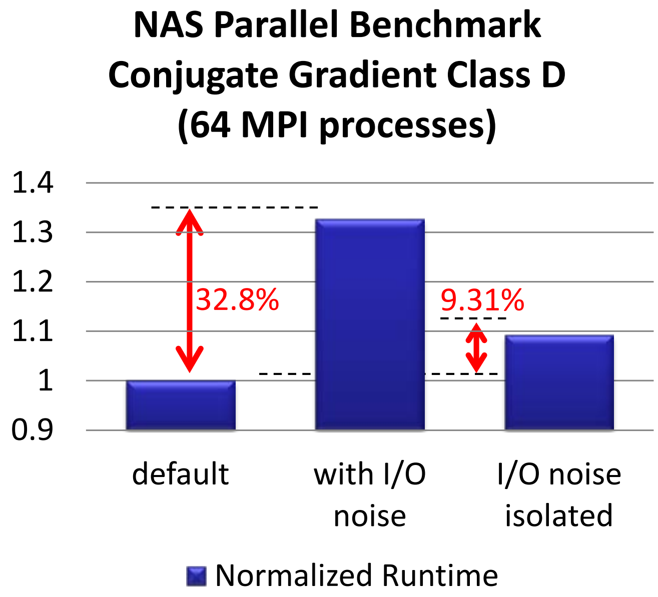
QoS-Aware Data Staging

- Asynchronous I/O introduces contention for network-resources
- How should data be orchestrated in a data-staging architecture to eliminate such contention?
- Can the QoS capabilities provided by cutting-edge interconnect technologies be leveraged by parallel filesystems to minimize network contention?









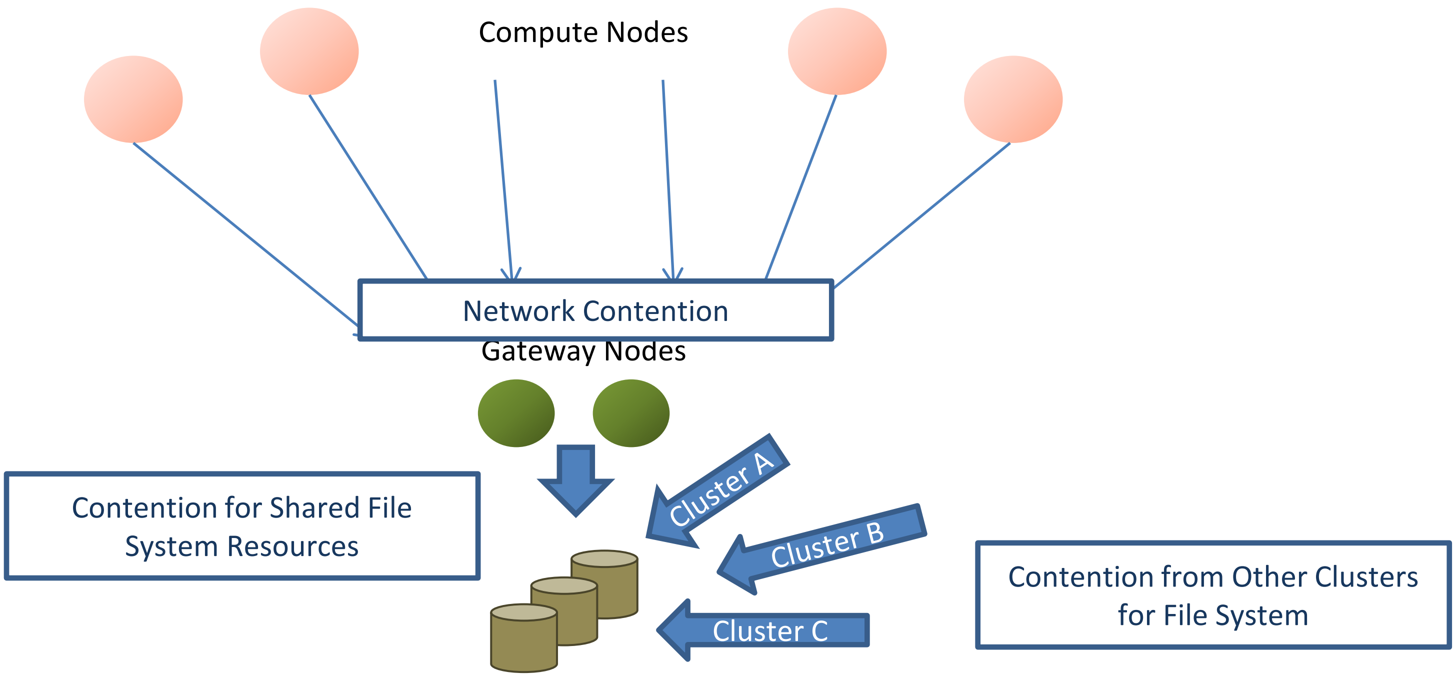
- **Reduces runtime overhead from 17.9% to 8% and from 32.8% to 9.31%, in case of AWP and NAS-CG applications respectively**

R. Rajachandrasekar, J. Jaswani, H. Subramoni and D. K. Panda, Minimizing Network Contention in InfiniBand Clusters with a QoS-Aware Data-Staging Framework, IEEE Cluster, Sept. 2012

Experience in Using SSDs

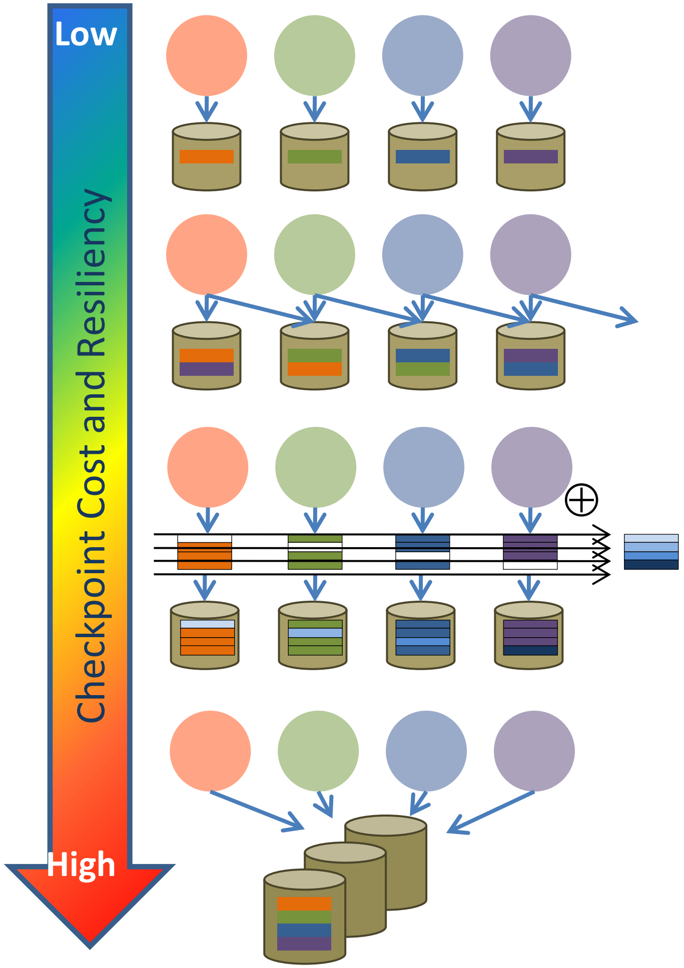
- Use of SSDs to accelerate MPI Fault-Tolerance
  - Accelerating checkpoint-restart and migration with hierarchical data-staging and high-throughput SSDs
  - Multi-Level checkpointing using SSDs with Scalable Checkpoint/Restart (SCR)
- Use of SSDs to accelerate Web 2.0
  - Using SSDs as a Virtual Memory Swap (existing naïve solution)
  - Accelerating Memcached with a SSD-based Hybrid-Memory architecture

Multi-Level Checkpointing with ScalableCR (SCR)



- Periodically saving application data to persistent storage
- Application- / System-level checkpointing mechanisms
- I/O intensive operation – bottleneck in the application
- Effective utilization of storage hierarchy is indispensable!
- LLNL's Scalable Checkpoint/Restart library – novel solution!

Multi-Level Checkpointing with ScalableCR (SCR)



Local: Store checkpoint data on node's local storage, e.g. SSDs, ramdisk

Partner: Write to local SSD and on a partner node

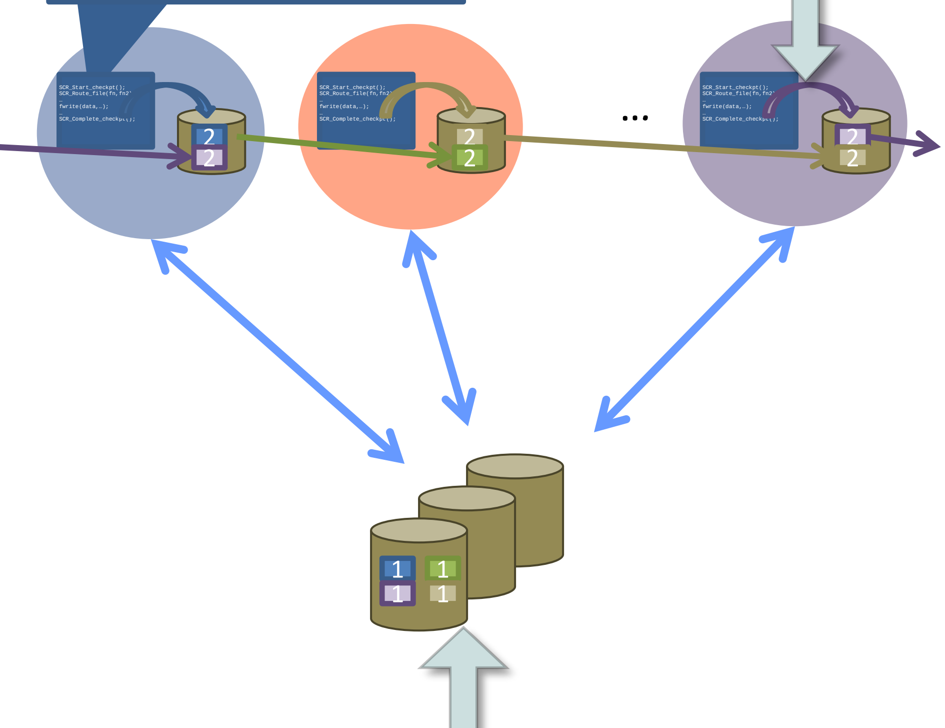
XOR: Write file to local SSD and small sets of nodes collectively compute and store parity redundancy data (RAID-5)

Stable Storage: Write to parallel file system

Application-guided Multi-Level Checkpointing

```
SCR_Start_checkpt();
SCR_Route_file(fn,fn2);
…
fwrite(data,…);
…
SCR_Complete_checkpt();
```

- First write checkpoints to node-local storage
- When checkpoint is complete, apply redundancy schemes



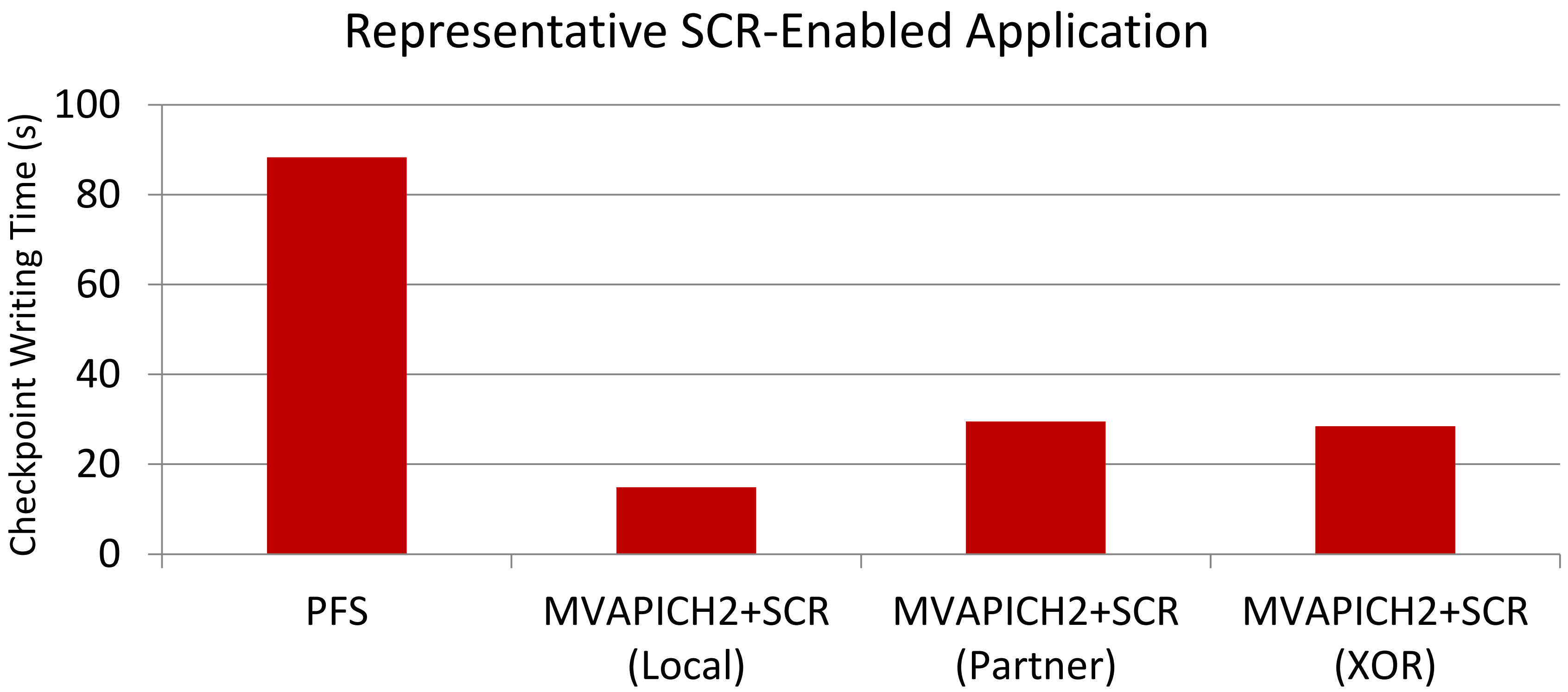
- Users select which checkpoints are transferred to global storage
- Automatically drain last checkpoint of the job

```
void checkpoint() {
  SCR_Start_checkpoint();

  int rank;
  MPI_Comm_rank(MPI_COMM_WORLD, &rank);

  char file[256];
  sprintf(file, “rank_%d.ckpt”, rank);

  char scr_file[SCR_MAX_FILENAME];
  SCR_Route_file(file, scr_file);
  FILE* fs = fopen(scr_file, “w”);
  if (fs != NULL) {
    fwrite(state, ..., fs);
    fclose(fs);
  }

  SCR_Complete_checkpoint(1);
  return;
}
```

SCR Support in MVAPICH2

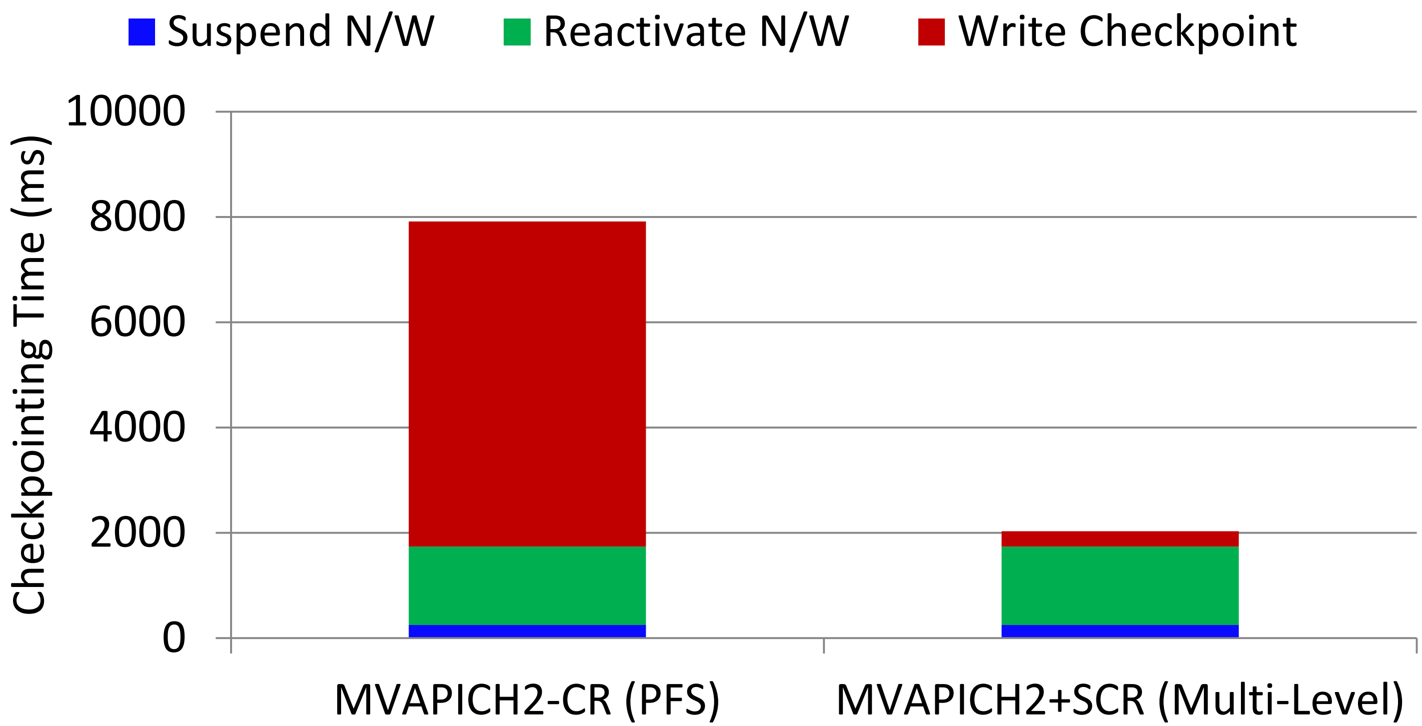
- Introduced in 1.9 (since 1.9b)
- Supports both
  - Systems-level transparent checkpointing
  - Applications-level checkpointing

Application-guided Multi-Level Checkpointing



- Checkpoint writing phase times of representative SCR-enabled MPI application
- **512** MPI processes (8 procs/node)
- Approx. **51 GB** aggregate checkpoints

Transparent Multi-Level Checkpointing



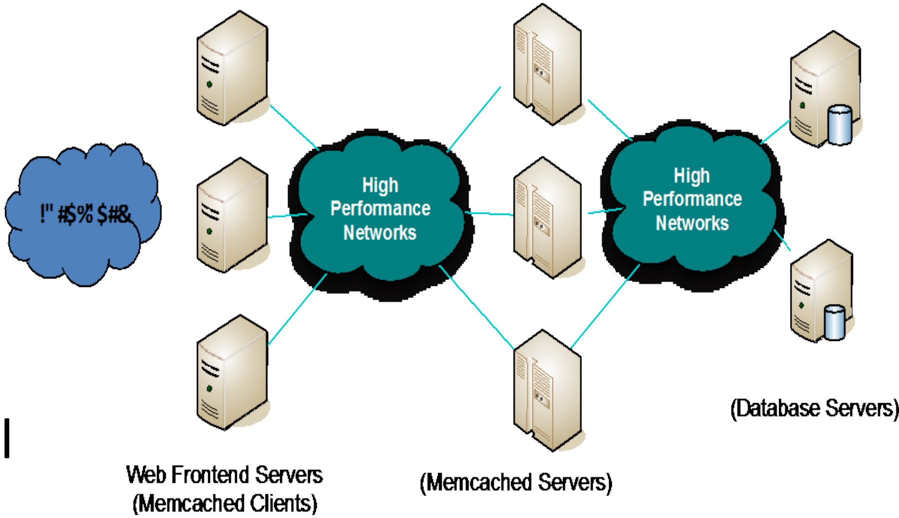
- **ENZO Cosmology application** – Radiation Transport workload
- Using MVAPICH2's CR protocol instead of the application's in-built CR mechanism
- **512** MPI processes (8 procs/node)
- Approx. **12.8 GB** aggregate checkpoints

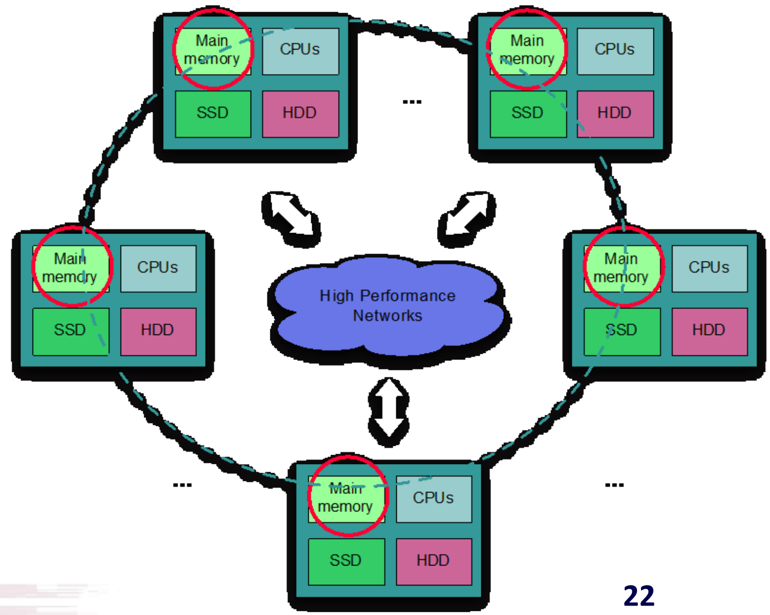
Experience in Using SSDs

- Use of SSDs to accelerate MPI Fault-Tolerance
  - Accelerating checkpoint-restart and migration with hierarchical data-staging and high-throughput SSDs
  - Multi-Level checkpointing using SSDs with Scalable Checkpoint/Restart (SCR)
- Use of SSDs to accelerate Web 2.0
  - Using SSDs as a Virtual Memory Swap (existing naïve solution)
  - Accelerating Memcached with a SSD-based Hybrid-Memory architecture

Enhancing Memcached Server with Hybrid Memory

- Many applications cache large amount of data in RAM for high performance
- Memcached is a distributed-memory object-caching system
- Memcached performance directly depends on aggregated memory pool size
- Difficult to scale memory pool size
  - Hardware cost
  - Power/thermal concern
  - Floor plan limits
- Existing solution: mmap() an SSD into virtual memory system
  - Significant overhead

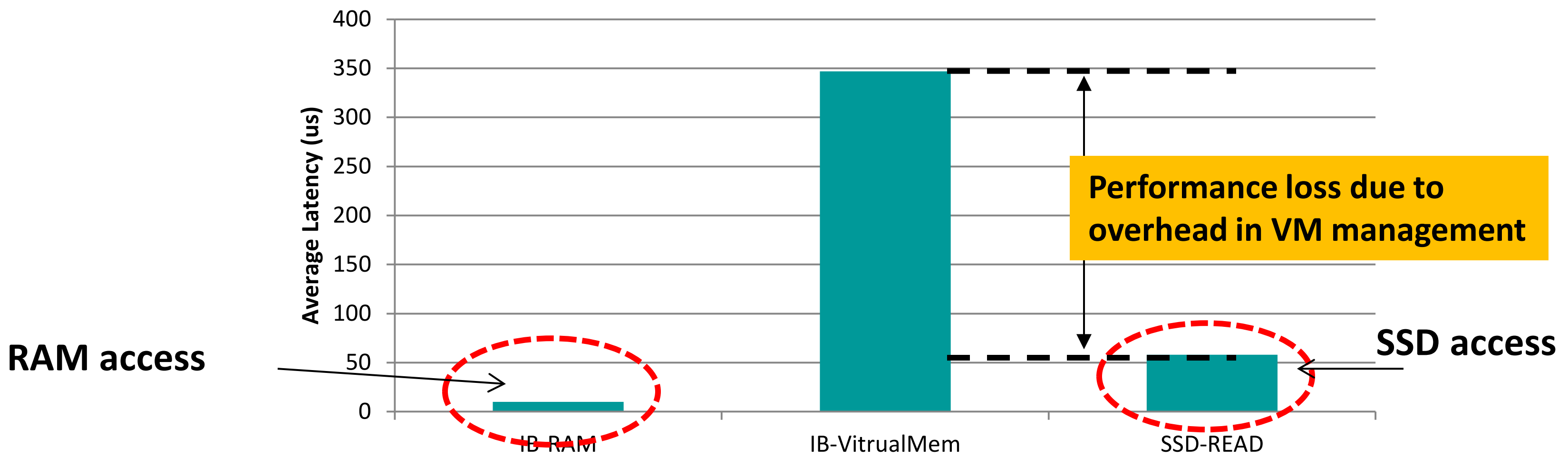




Drawback of Existing Virtual Memory Subsystem

- In-kernel VM Management System manipulates SSD at page granularity
- Entire flash page has to be loaded/overwritten even for a single byte read/update
- Excessive read/write traffic undermines SSD lifespan
- Heavy software stack overhead inside the kernel

SSD Used as Virtual Memory Swap Device



- Memcached Get Latency at 1KB Object Size:
  - 10 us from IB-RAM
  - 347 us from IB-VirtualMem (SSD-Mapped VM)
  - 68 us from SSD random read

Performance loss due to overhead in VM Management

SSD Used as Virtual Memory Swap Device

Get Latency (us)

	IB Verbs	IPoIB	10GigE	1GigE
MySQL	N/A	10763	10724	11220
Memcached (In RAM)	10	60	40	150
Memcached (Naïve mmap from SSD)	347	387	362	455

SSD Basic Performance (us) (PCI-e SSD)

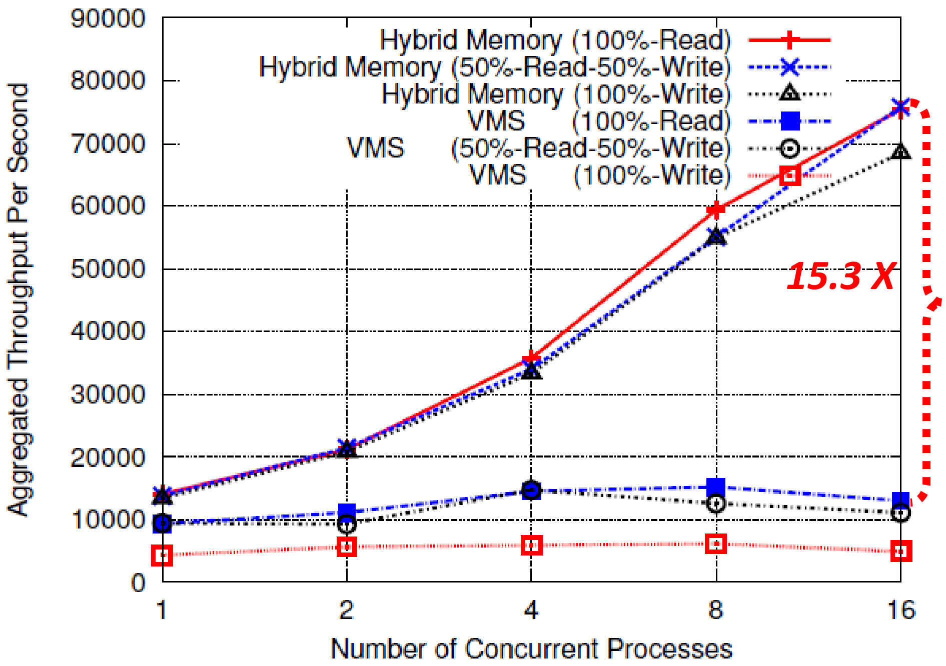
	Random Read	Random Write
Latency	68	70

SSD-Assisted Hybrid Memory

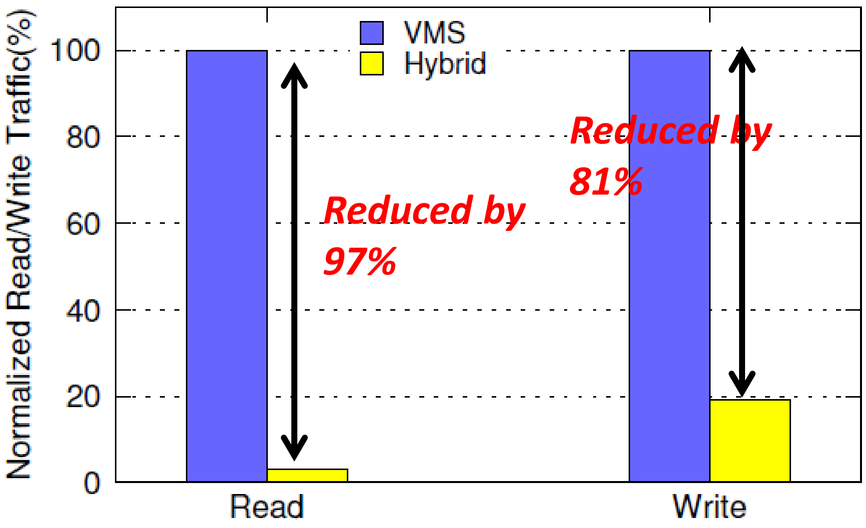
- Hybrid memory works as an object cache
- Manages resource allocation at object granularity
  - More efficient than allocation at page granularity
- Low latency object access due to SSD fast random read property
- Batched write to amortize writing cost
- Append-only write model to avoid in-place update to SSD
  - SSD is treated as a log-structured sequences of blocks

Microbenchmark: Raw Performance

Operation Throughput



Read/Write Volume to SSD

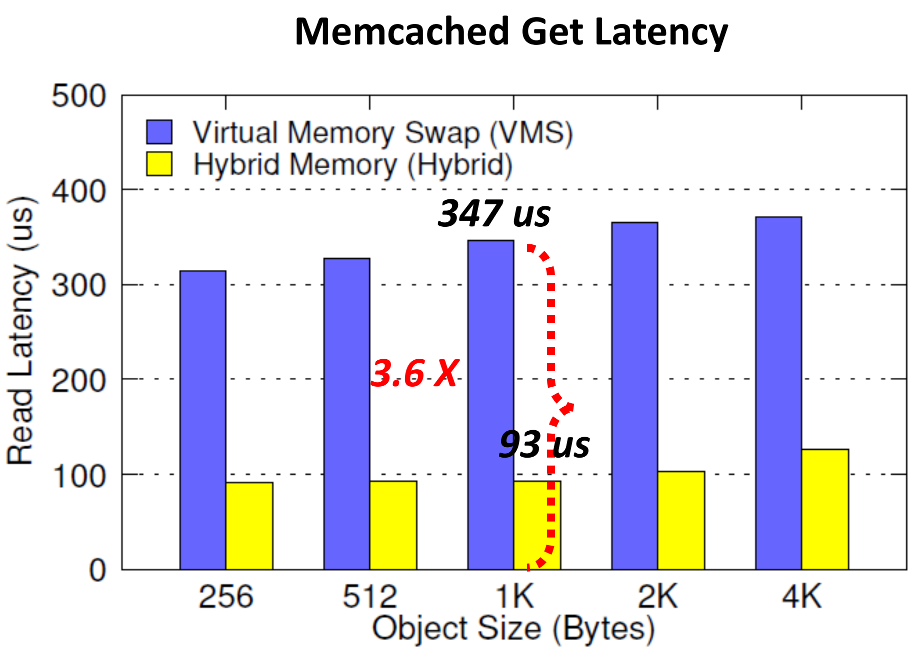


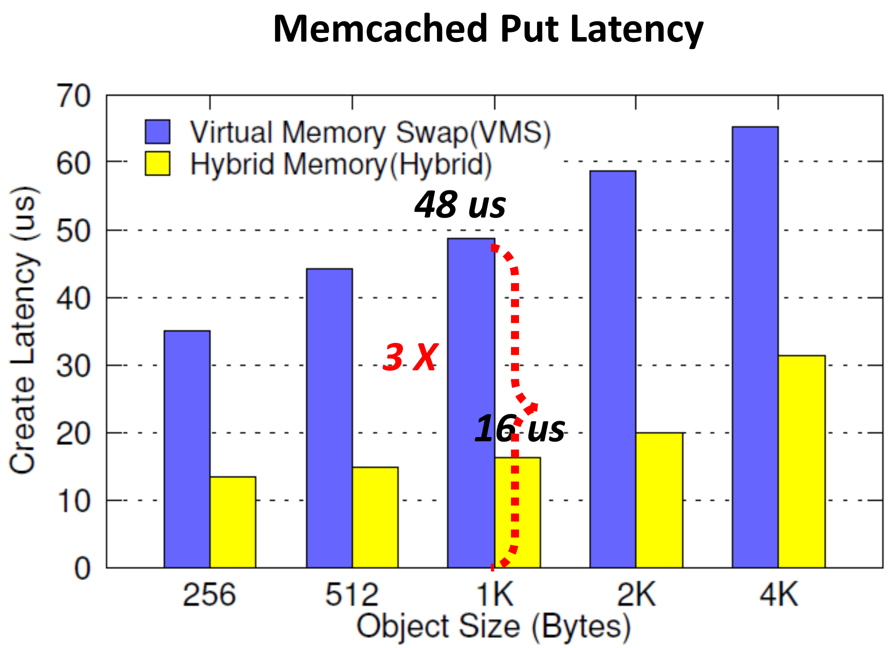
- 30GB data in SSD, 256 MB read/write buffer
- 1KB object size

- Read is reduced by 97%
- Write is reduced by 81% => 5.3X longer lifespan

X. Ouyang, N. Islam, R. Rajachandrasekar, J. Jose, M. Luo, H. Wang and D. K. Panda, SSD-Assisted Hybrid Memory to Accelerate Memcached over High Performance Networks, Int'l Conference on Parallel Processing (ICPP '12), September 2012.

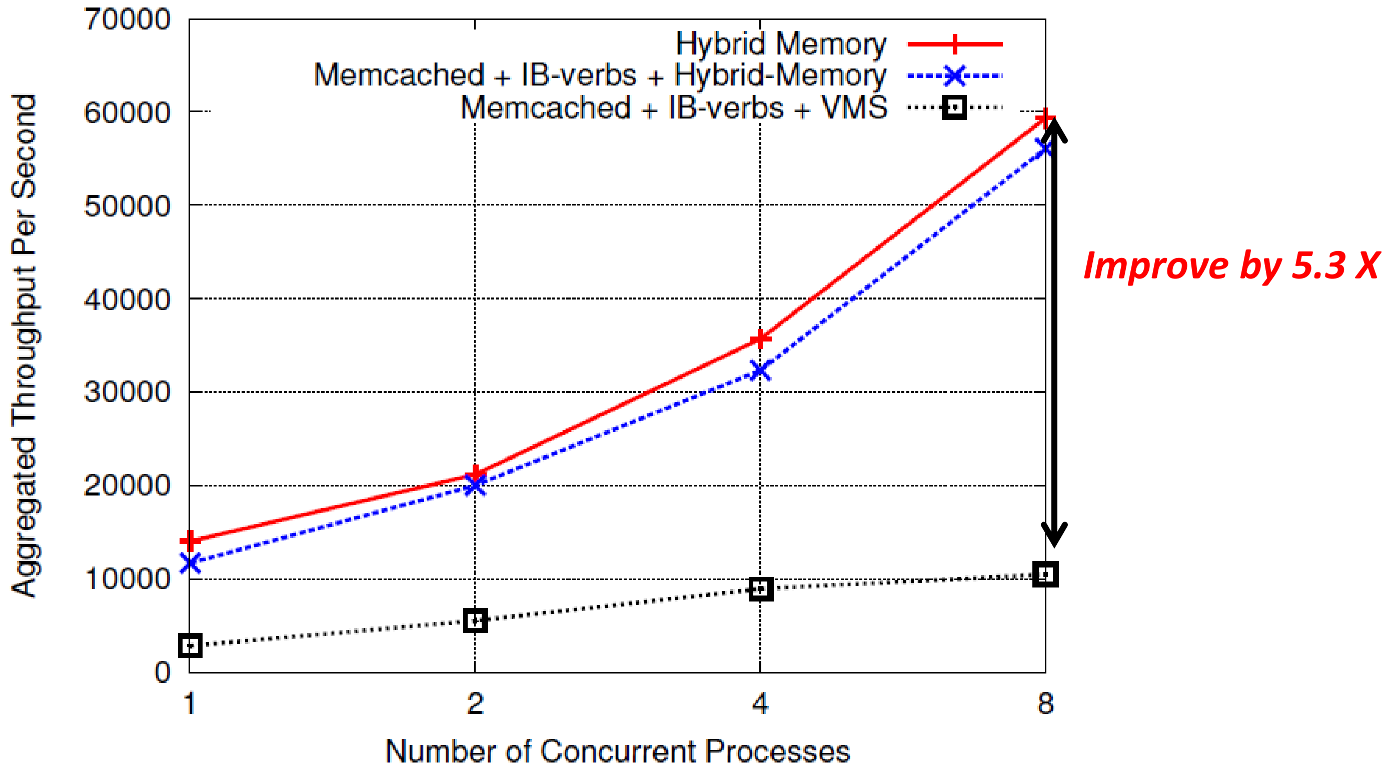
Memcached: Operation Latency





- Memcached-1.4.5 with InfiniBand DDR
- 30GB data in SSD, 256 MB read/write buffer
- Get / Put a random object

Memcached: Get Throughput



- Memcached-1.4.5 with InfiniBand DDR
- 30GB data in SSD, object size = 1KB, 256 MB read/write buffer
- 1,2,4,8 client process to perform random get()

Concluding Remarks

- SSD technology is emerging
- Special performance benefits with PCIe-based SSDs
- Presented some case studies to take benefits of SSDs for scientific and enterprise environments
- Provides new opportunities to be used in designing next generation HPC systems