Robust Storage Management in the Machine Room and Beyond

Presented by

Sudharshan Vazhkudai

Computer Science Research Group
Computer Science and Mathematics Division

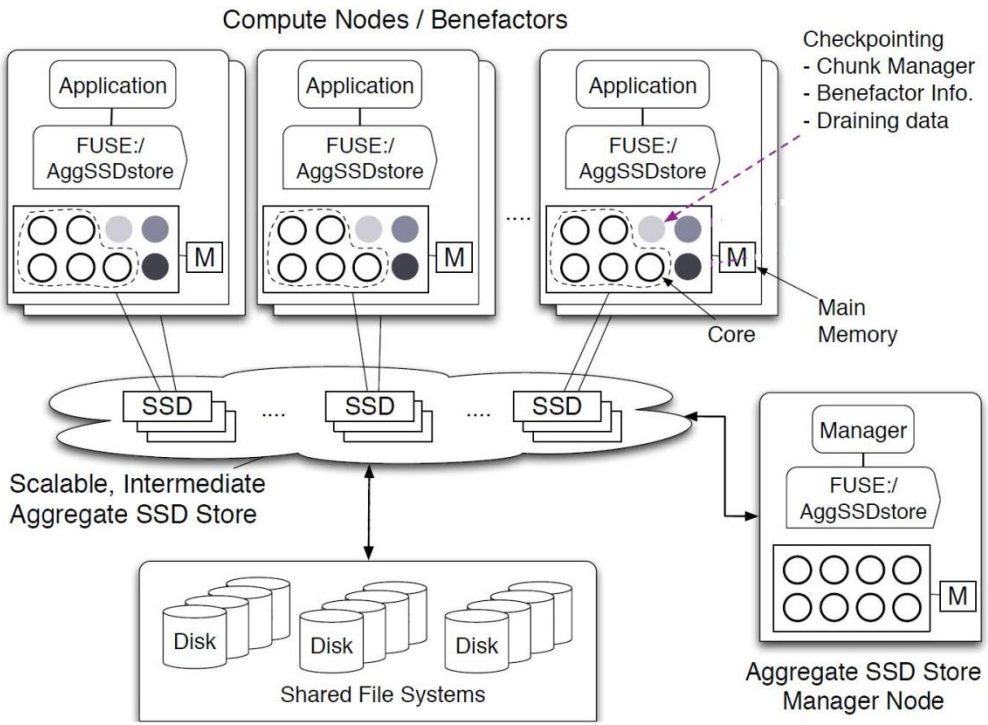
In collaboration with
Virginia Tech: Ali Butt, Henry Monti, Min Li
North Carolina State University: Xiaosong Ma, Fei Meng
ORNL: Chao Wang, Youngjae Kim, Christian Engelmann

Problem space: HPC storage crisis

- Data checkpointing, staging, and offloading are all affected by data unavailability and I/O bandwidth bottleneck issues
  - Checkpointing terabytes of data to a traditional file system results in an I/O bottleneck
  - Compute time wasted on staging at the beginning of the job
  - Early staging and late offloading waste scratch space
  - Delayed offloading renders result data vulnerable to purging
  - Upshot
    - Increased turnaround time, checkpoint bottleneck
    - Increased job wait times due to staging/offloading and storage delays/errors
    - Poor end-user data delivery options

Stdchk: An aggregate SSD/memory-based checkpoint storage system

- Aggregates storage space from compute node-local SSD/memory to present a collective, intermediate checkpoint storage or a staging ground
  - Job's own allocated nodes can contribute storage space
- Transparent FS interface to the storage using FUSE (e.g., /AggregateSSDstore)
- Benefactor process contributes SSD space or memory buffers to a manager
- Manager maintains metadata on benefactor status, contributions and chunk to benefactor mapping



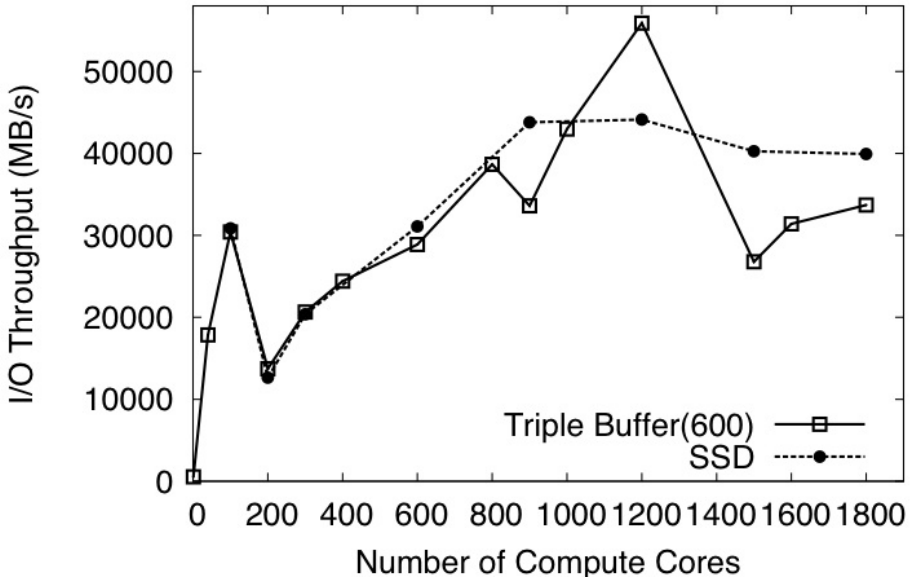
- Application writes to the mount point translated into striping of chunks across a stripe width of benefactors
  - Parallel I/O across distributed SSD or memory

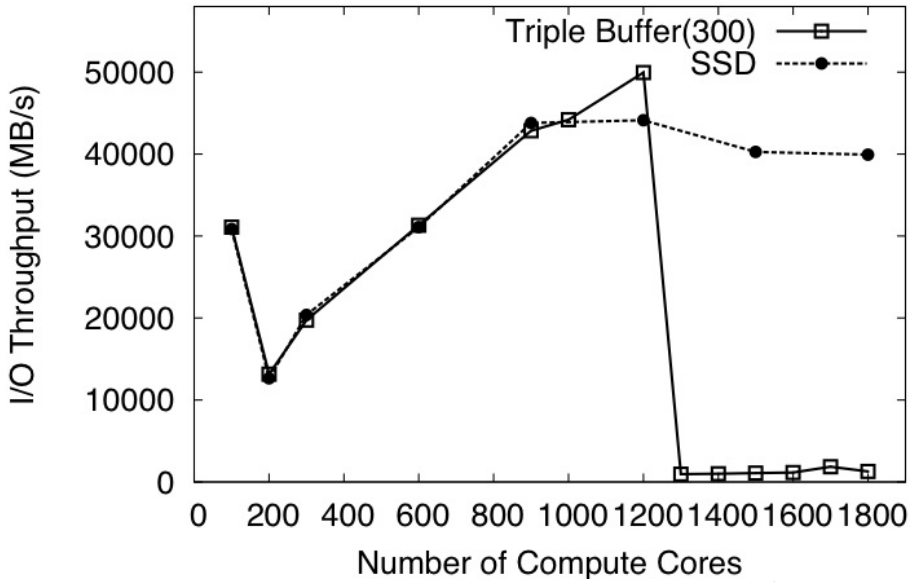
Stdchk (cont'd)

- Features
  - Draining of checkpoint images to a parallel file system
  - Striping policies factor in SSD locality (i.e., preference to node-local SSD)
  - Incremental checkpointing and pruning of checkpoint files
    - Compare chunk hashes from two successive intervals
    - Initial experiments suggest a 10–25% reduction in size for BLCR checkpoints
    - Purge images from previous interval once the current image is safely stored
    - File system is unable to perform such optimizations
  - A multitiered storage of aggregate memory and aggregate SSD layers
  - Applications can also mmap() into the aggregate SSD storage to perform out-of-core computations

Checkpoint throughput

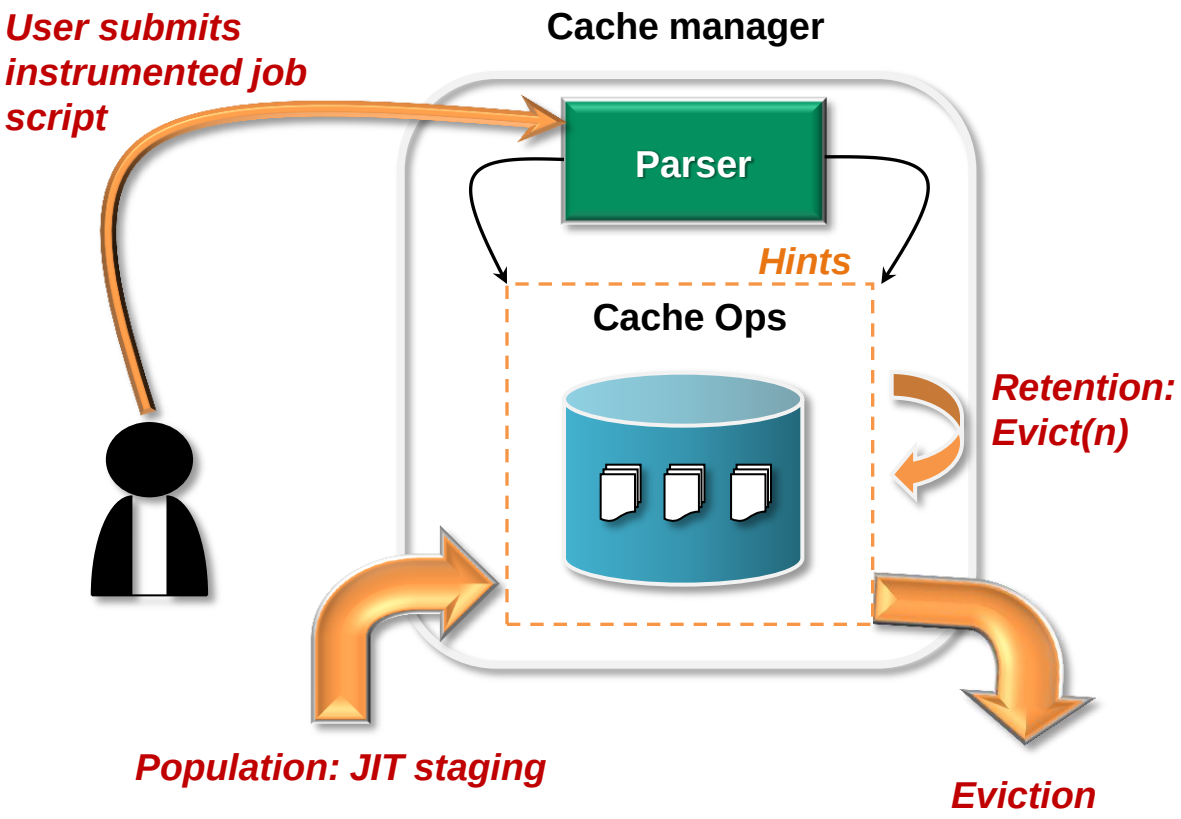
- Results
  - Up to 1800 cores checkpointing 0.25 GB each ~ 0.5 TB overall
  - Aggregate SSD Store
    - 32 GB each
    - Ramdisk SSD emulator ~ 175 MB/s
    - Peak aggregate SSD throughput of 45 GB/s
  - Aggregate Memory Store
    - 600 benefactors with 1 GB each
    - 300 benefactors with 1 GB shows the effect of draining to PFS
    - Peak aggregate memory throughput of 56 GB/s





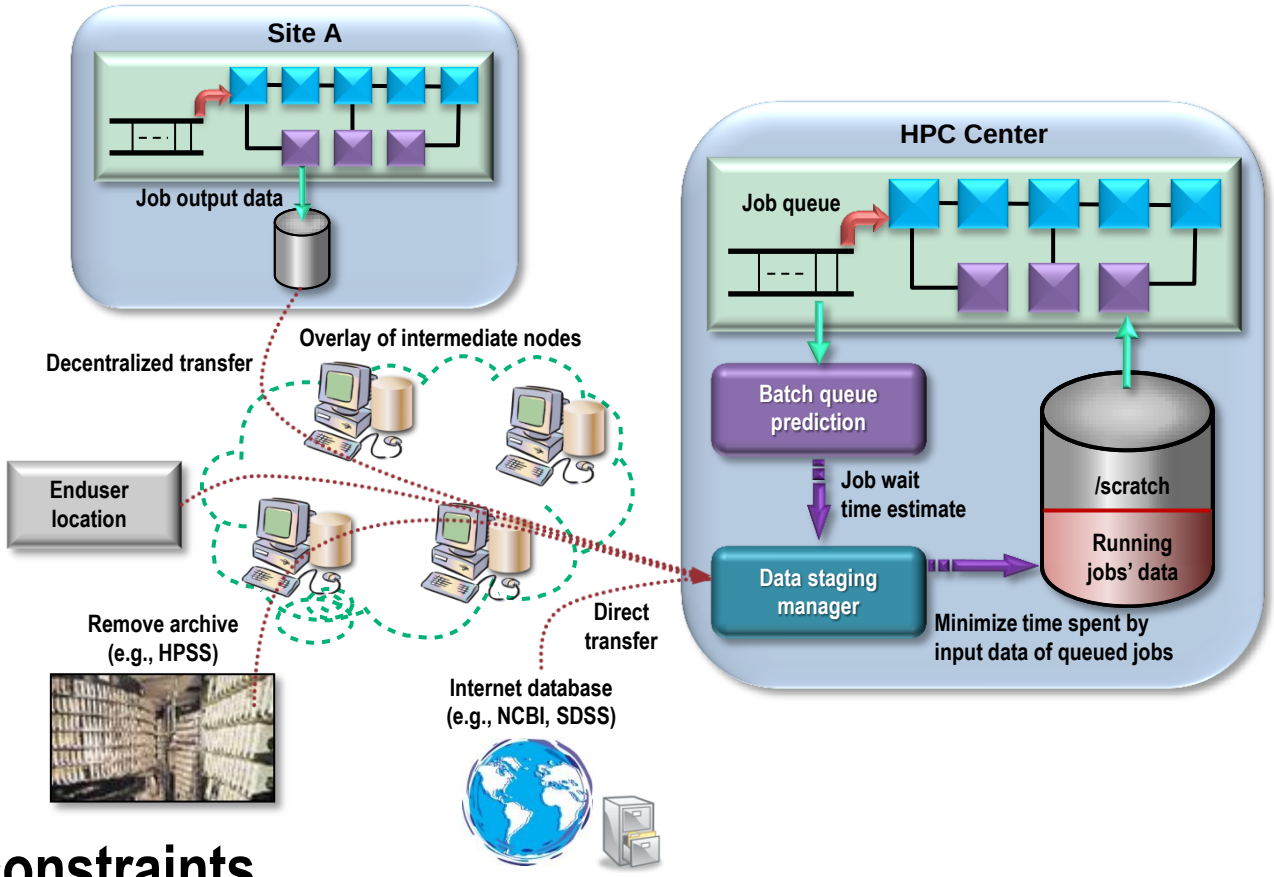
Scratch as cache

- Globally manage the scratch cache
- Data movement is performed using cache population and eviction tools
- Users cannot arbitrarily move data
- Input and output data are not retained beyond the lifetime of the application run



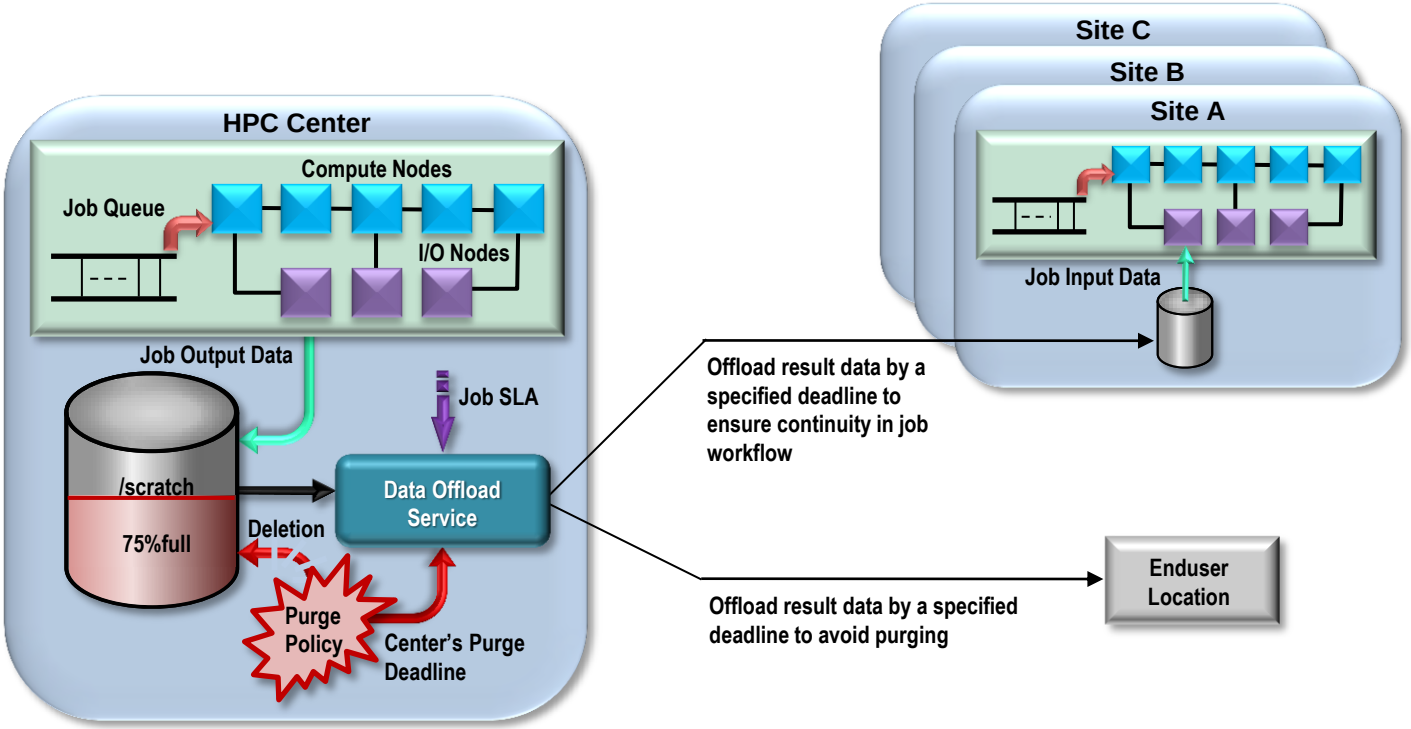
Addresses many of the problems of disjoint management!

Just In Time (JIT) staging



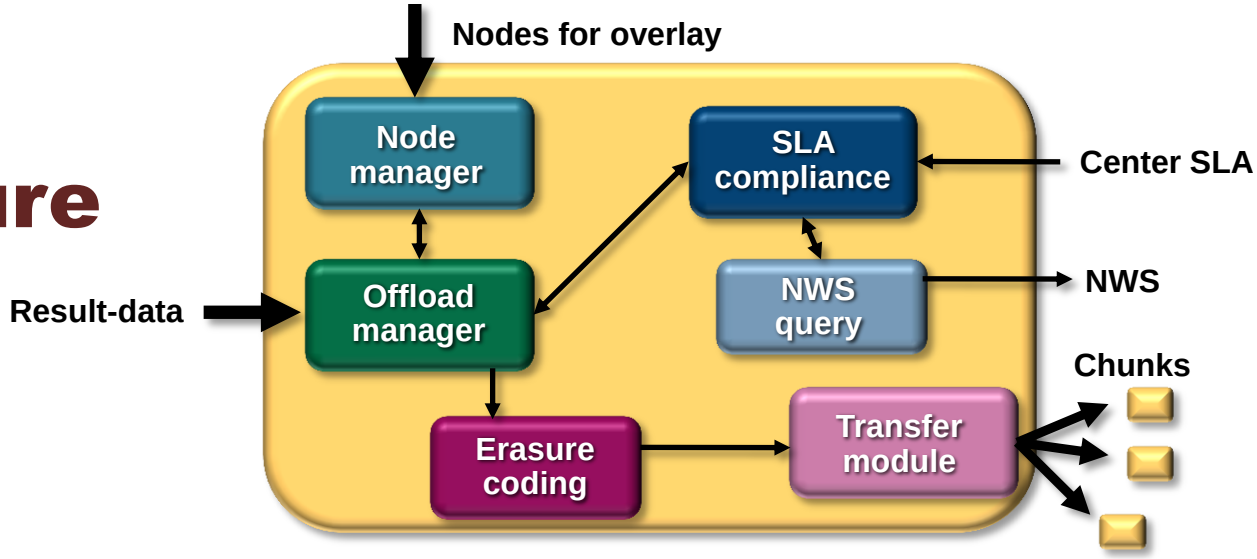
- **Staging constraints**
  - $Max(T_j) \leq T_{JobStartup}$
  - Exposure window of each input dataset, $E_{wj} = T_{JobStartup} - Max(T_j)$; $E_w = Sum(E_{wj})$
  - The closer E_w is to 0, the better

Eager offloading of result data

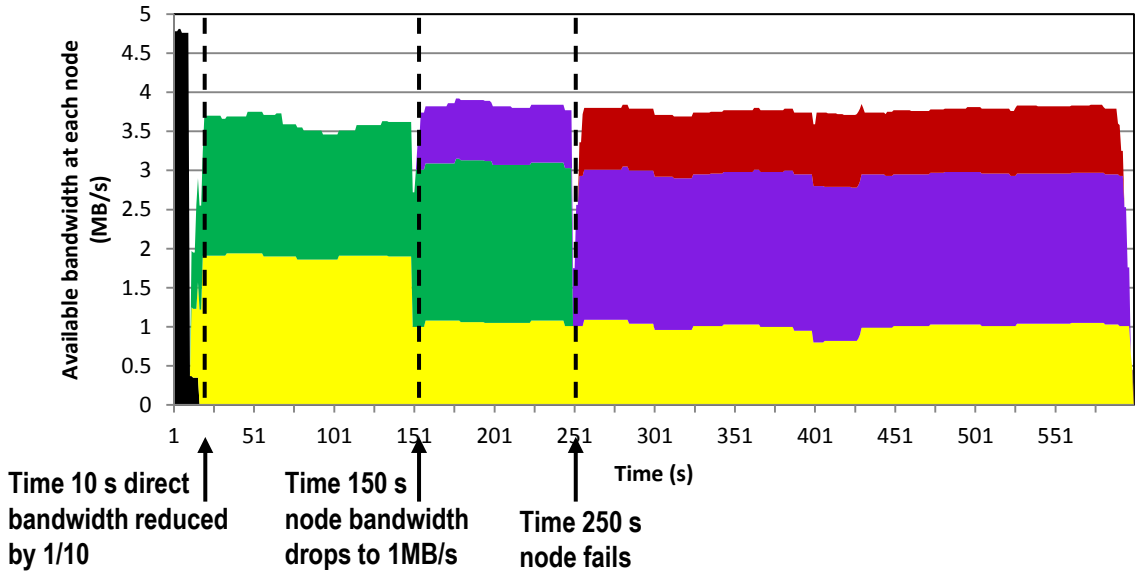


- **Eager offloading features:**
  - Reconcile offload constraint: before center purge and by the user-specified deadline: $T_{offload} < Min(D_{purge}, J_{SLA})$
  - Use replication and erasure coding of chunks for redundancy
  - Integration with PBS, NWS, and Bittorrent

Eager offloading architecture



Results: Adapting to dynamic network behavior



A staged offload is capable of adapting to bandwidth changes or failures

SLA is 600 seconds

Transferring 2.1 GB file

Contact

Sudharshan Vazhkudai

Computer Science Research Group
Computer Science and Mathematics Division
(865) 576-5547
vazhkudaiss@ornl.gov

http://www.csm.ornl.gov/~vazhkuda/Storage.html