Towards Decoupling Storage and Computation in Hadoop with SuperDataNodes

George Porter* <gmporter@cs.ucsd.edu>
Center for Networked Systems
U. C. San Diego

LADIS 2009 / Big Sky, MT

* Work performed at Sun Labs, Burlington, MA.

Data-intensive computing and 

- Hadoop is growing, gaining adoption, and used in production:

  - Facebook imports 25 TB/day to 1K Hadoop nodes
- A key to that growth and efficiency relies on coupling compute and storage
  - Benefits of moving computation to data
  - Scheduling, locality, reduce network traffic, map parallelism
    - 'Grep' type workloads especially

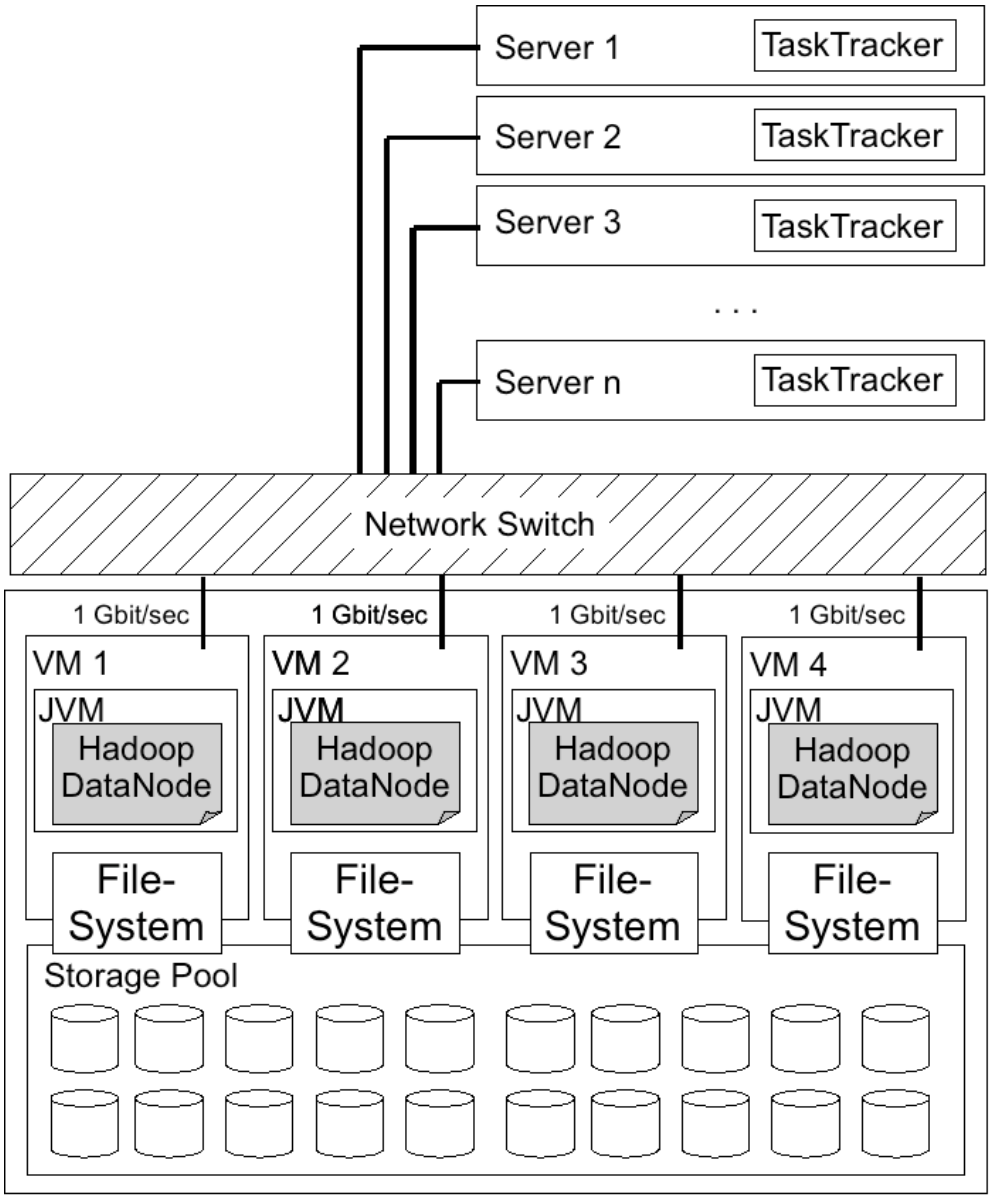
When to couple storage with computation?

- Critical (yet complicated) design decision
- Emerging best practices with dedicated clusters
- Your datacenter design may not be based on needs of Hadoop
  - Adding Map/Reduce functionality to existing cluster
  - Small workgroups who like the programming model
    - Pig, Hive, Mahout…
- Mixture may change over time
  - Non-uniform data access patterns
  - Desire to power down some compute functionality during periods of low utilization
    - Without affecting storage functionality

Goal

- Support late binding between storage and computation
  - Explore alternative balances between the two
    - Specifically the extreme point of separating all storage from the workers and consolidating it into a *SuperDataNode (SDN)*
  - Facebook observations:
    - Small and medium sized jobs exhibit large rack-local workers, but not node-local workers
- Non-goal: Replacing traditional Hadoop deployments

SuperDataNode Approach



- Key Features
  - "Stateless" worker tier
  - Storage node with shared pool of disks under single O/S
    - O/S as central broker of disk requests
  - High bisection bandwidth to worker tier
    - 4x1GigE; 10GigE
- Artifacts of my experiments
  - Per net-interface VM exporting virtual storage nodes

Advantages

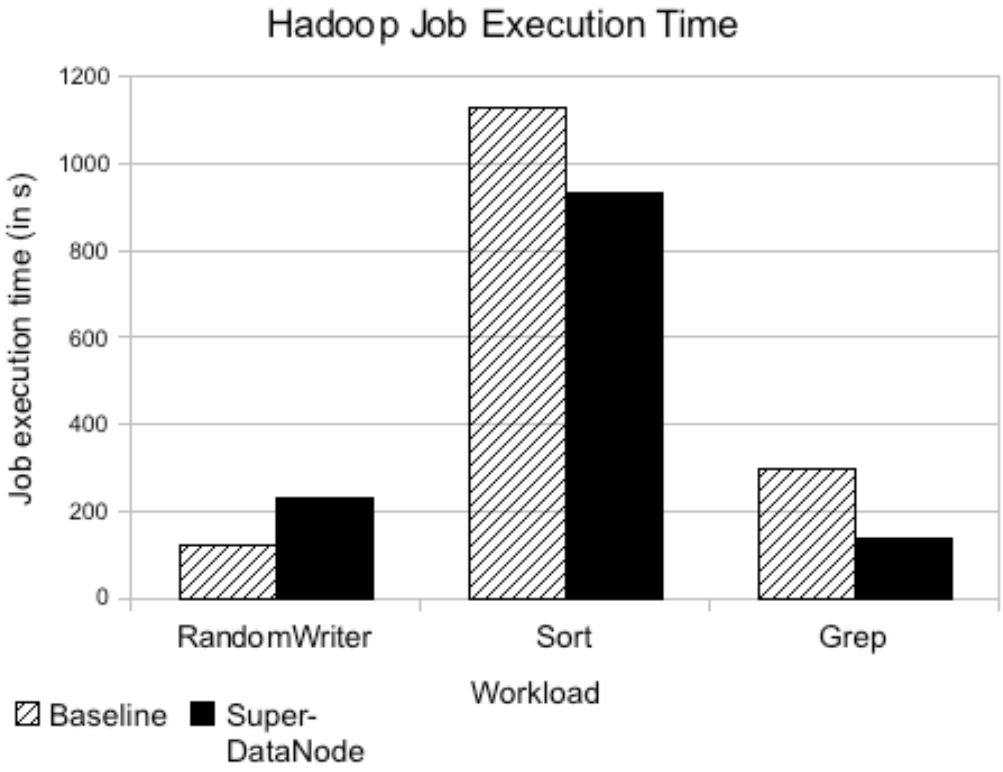
- Decouple amount of storage from number of worker nodes
- More intra-rack bandwidth than inter-rack bandwidth
- Support for “archival” data
  - Subset of data with low probability of access
- Increased uniformity for job scheduling and block placement
- Ease of management
  - Workers become stateless; SDN management similar to that of a regular storage node
- Replication only for node failures

Limitations

- Scarce storage bandwidth between workers and SDN
  - Effective throughput with N disks in SDN (@ 100MB/sec each)
    - 1:N ratio of bandwidth between local and remote disks
    - 4 Gbit/sec: min(100N, 400/N) MB/sec
    - 10 Gbit/sec: min(100N, 1000/N) MB/sec
- Effect on fault-tolerance
  - Disk vs Node vs Link failure model
  - Replication
- Cost
- Performance depending on the workload

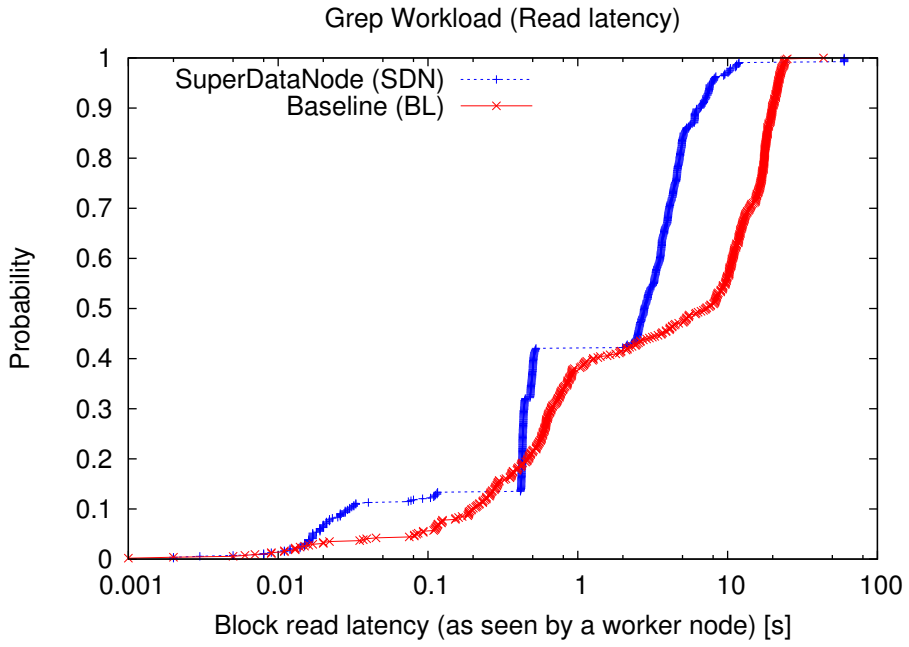
Evaluation

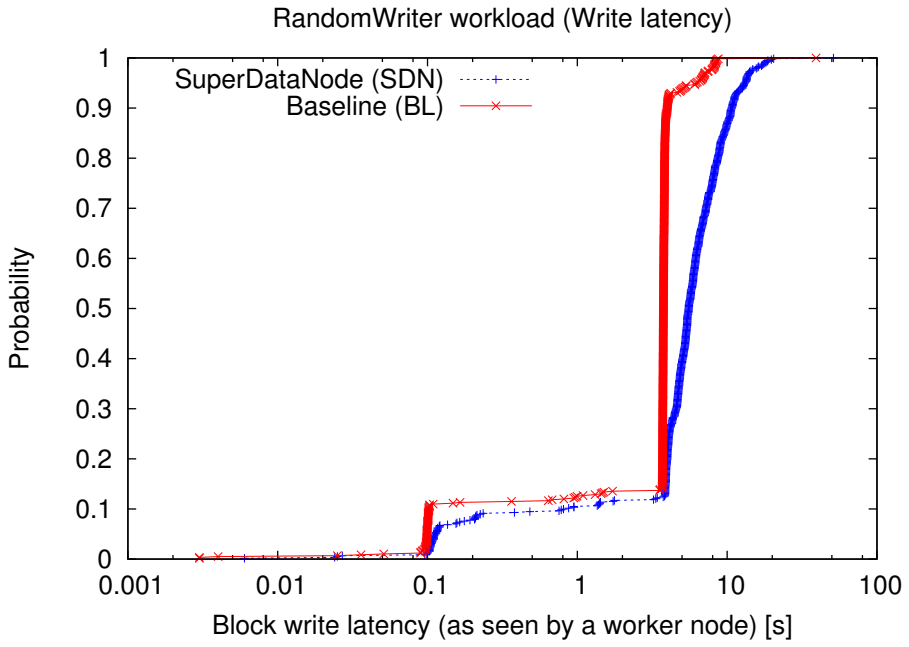
- Baseline
  - 10 1u servers, 2 disks each for data w/ ZFS; worker and storage node co-located
- Experimental setup
  - 10 1u servers with no data; 20 disks in SDN w/ ZFS (Thumper successor); 4 virtual datanodes in SDN
- Observations
  - RandonWriter exhibits perfect parallelism and 100% local-only write behavior (worst case against SDN)



Impact of O/S on the disk pool

- Isolate block read/write latency within the host only
  - Ignoring network, JVM, Map function, …
- Instrumented HDFS data path with X-Trace events
- Management of a central pool of disks based on visibility of every node using the SDN





Related Work

- Advantages of moving computation towards the data [Jim Gray, *Queue*, 2008]
- FAWN: A Fast Array of Wimpy Nodes [SOSP09]
- Archival workloads [SAM/QFS]
- Deployed Hadoop installations
  - 3800 node Terasort [Yahoo]
  - Counterpoint to the storage-to-I/O balance [Joseph M. Hellerstein]
  - As a service on EC2 [http://aws.amazon.com]

Conclusions

- Choosing the balance of storage to computation critical
  - Performance, efficiency, power, job scheduling
- Desire mechanism to delay this binding until runtime and decouple the two
  - Can support changing storage/CPU ratios, new datasets and workloads, conserve power during periods of low demand, greater management flexibility
- Comparable performance for a variety of workloads

Discussion

- Thank you

- Thanks to Hsianglung Wu, Matei Zaharia, Steve Heller