Understanding Hadoop Performance on Lustre

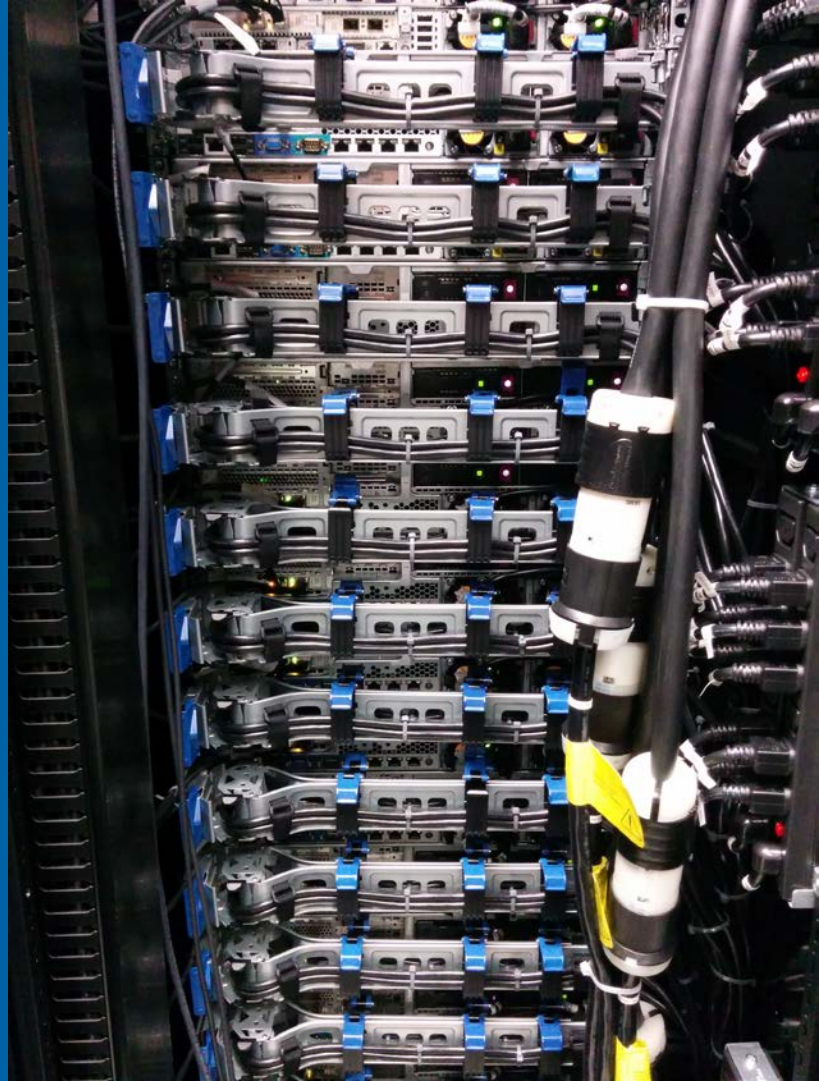
Stephen Skory, PhD | Seagate Technology

Collaborators: Elsie Betsch, Daniel Kaslovsky, Daniel Lingenfelter, Dimitar Vlassarev, and Zhenzhen Yan

LUG Conference | 15 April 2015

Our live demo of Hadoop on Lustre at Supercomputing November 17-20, 2014

- Why Hadoop on Lustre?
- LustreFS Hadoop Plugin
- Our Test System
- HDFS vs Lustre Benchmarks
- Performance experiments on Lustre and Hadoop Configuration
- Diskless Node Optimization
- Final Thoughts



Why Hadoop on Lustre?

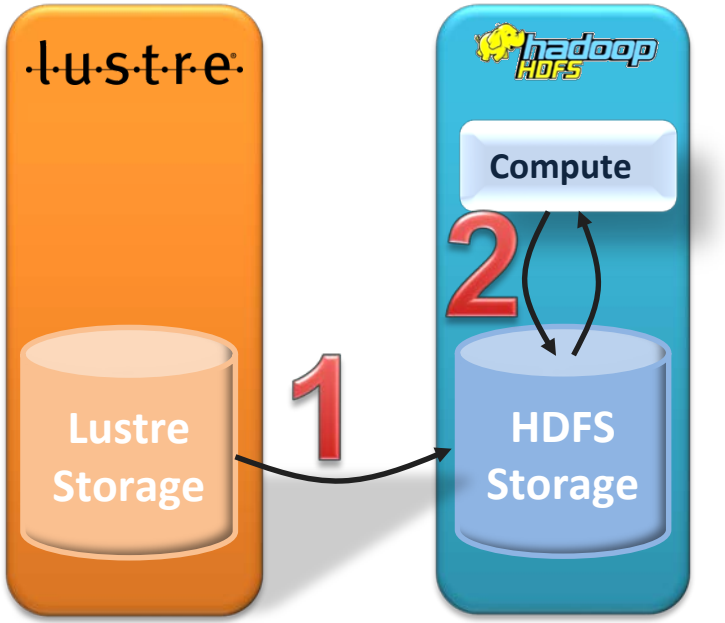
- **Lustre is POSIX compliant and convenient for a wide range of workflows and applications, while HDFS is a distributed object store designed narrowly for the write once, read many Hadoop paradigm**
- **Disaggregate computation and storage – HDFS requires storage to accompany compute, with Lustre each part can be scaled or resized according to needs**
- **Lower Storage Overhead – Lustre is typically backed by RAID with ~40% overhead, while HDFS has a default 200% overhead**
- **Speed – In our testing we see nearly 30% TeraSort v2 Performance boost over HDFS. When transfer time is considered the improvement is over 50%**

Hadoop on HDFS and on Lustre Data Flow

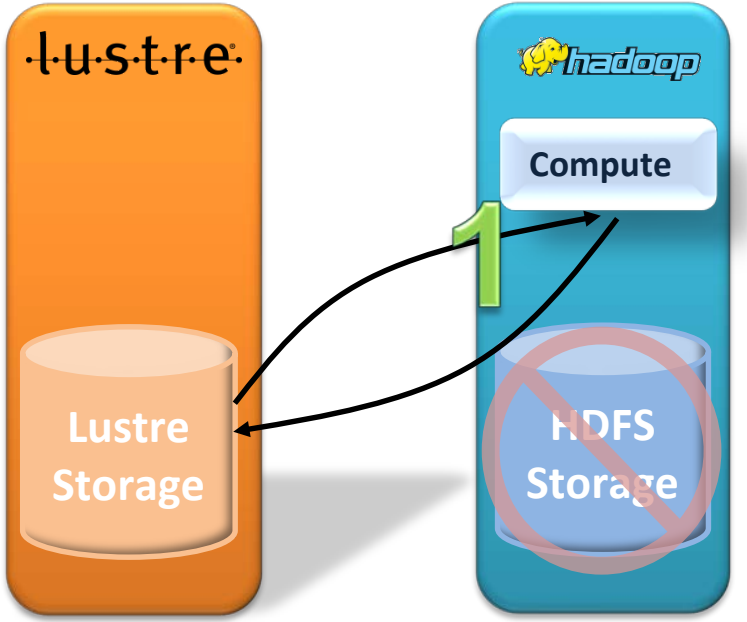
Typical Workflow:

Ingestion of Data before Analysis

1) Transfer (and usually duplicate) data on HDFS
2) Analyze data



1) Analyze data on Lustre



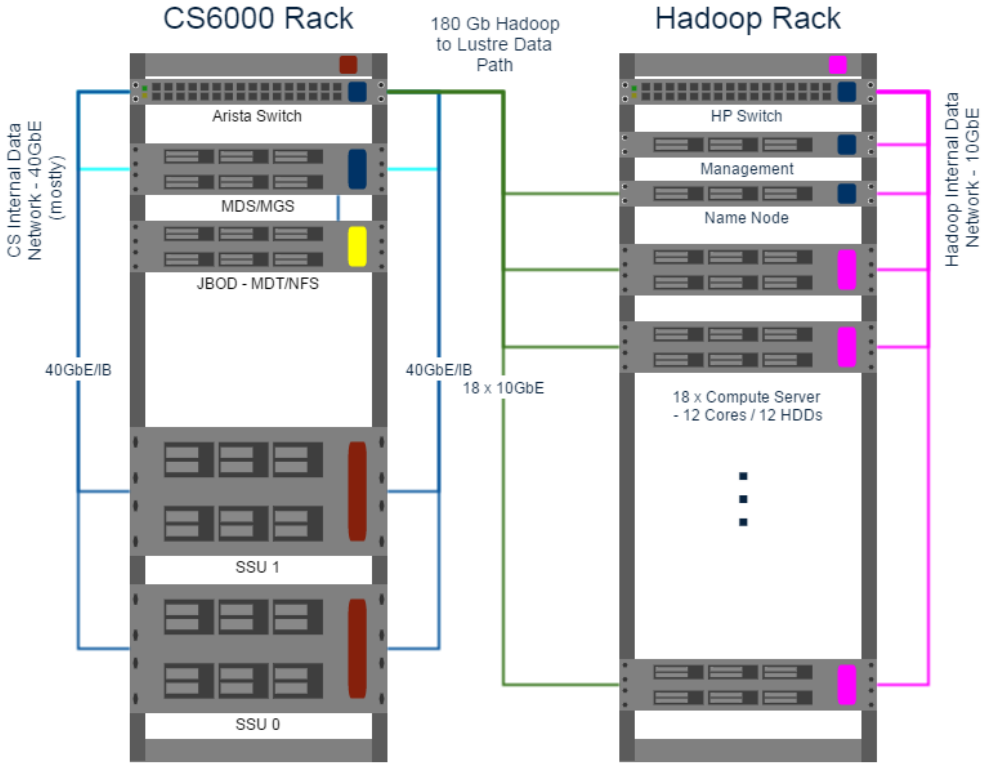
LustreFS Plugin

- **It is a single Java jar file that needs to be placed in the CLASSPATH of whatever service that needs to access Lustre file**
- **Requires minimal changes to Hadoop configuration XML files**
- **Instead of file:/// or hdfs://hostname:8020/, files are accessed using lustrefs:///**
- **HDFS can co-exist with this plugin, and jobs can run between the two file systems seamlessly**
- **Hadoop service accounts (e.g. “yarn” or “mapred”) need to have permissions to a directory hierarchy on Lustre for temporary and other accounting data**
- ***Freely available* at http://github.com/seagate/lustrefs today, with documentation**
- **A PR has been issued to include the plugin with Apache Hadoop**

Our Hadoop on Lustre Development Cluster

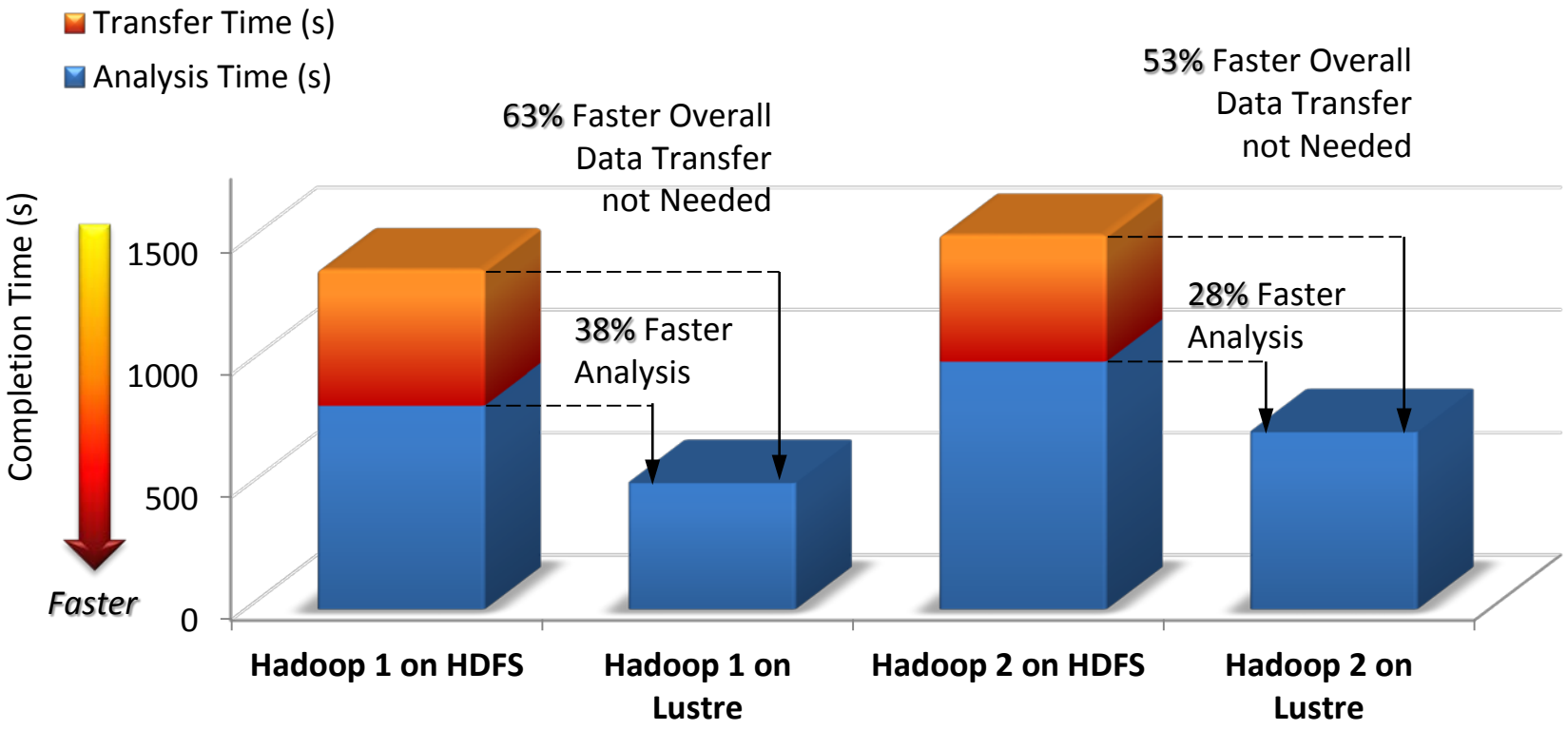
All benchmarks shown were performed on this system

- **Hadoop Rack:**
  - 1 Namenode, 18 Datanodes
  - 12 cores & 12 HDDs per Datanode, total of 216 for both
  - 19x10 GbE connection within rack for Hadoop traffic
  - Each Hadoop node has a second 10GbE connection to the ClusterStor network for a total of 190Gb full duplex bandwidth
- **ClusterStor Rack:**
  - 2 ClusterStor 6000 SSUs
  - 160 data HDDs



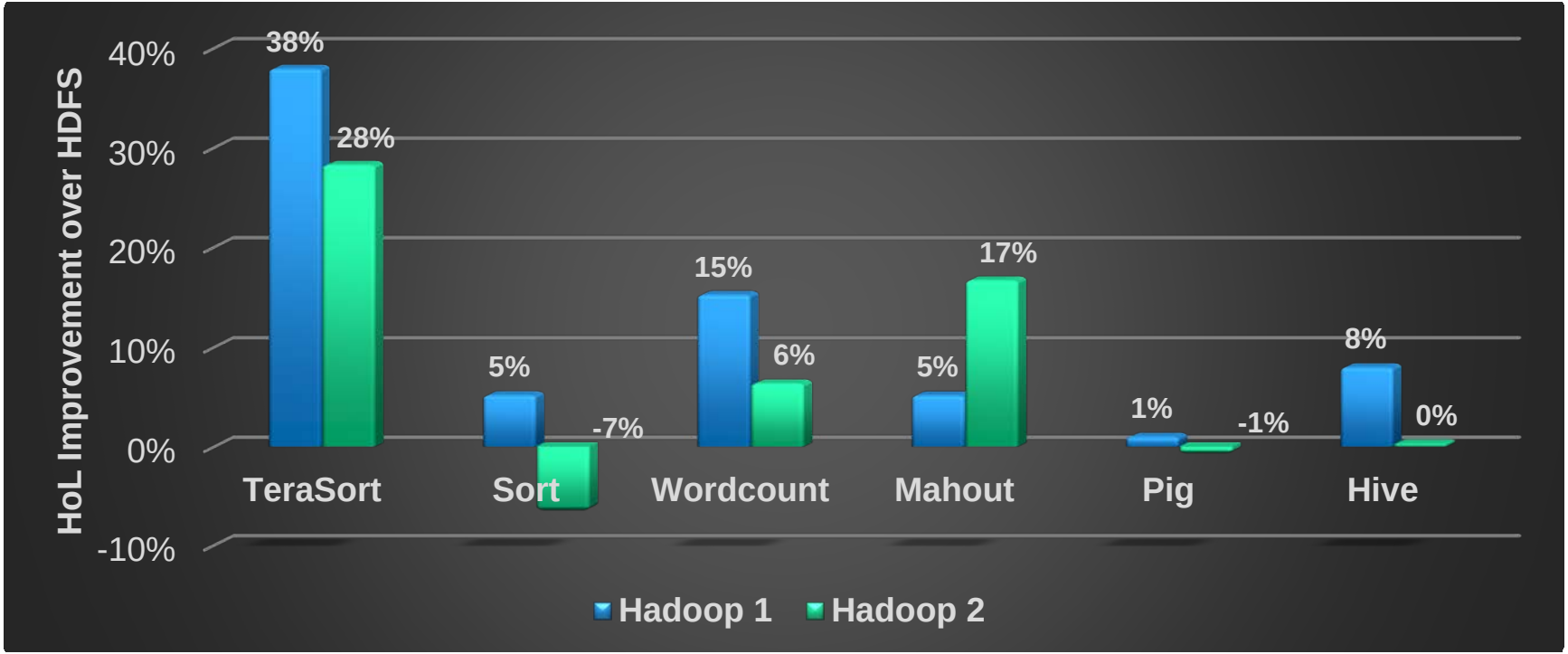
How Much Faster is Hadoop on Lustre?

Terasort timings



How Much Faster is Hadoop on Lustre?

Ecosystem Timings Compared to HDFS (transfer times not included)



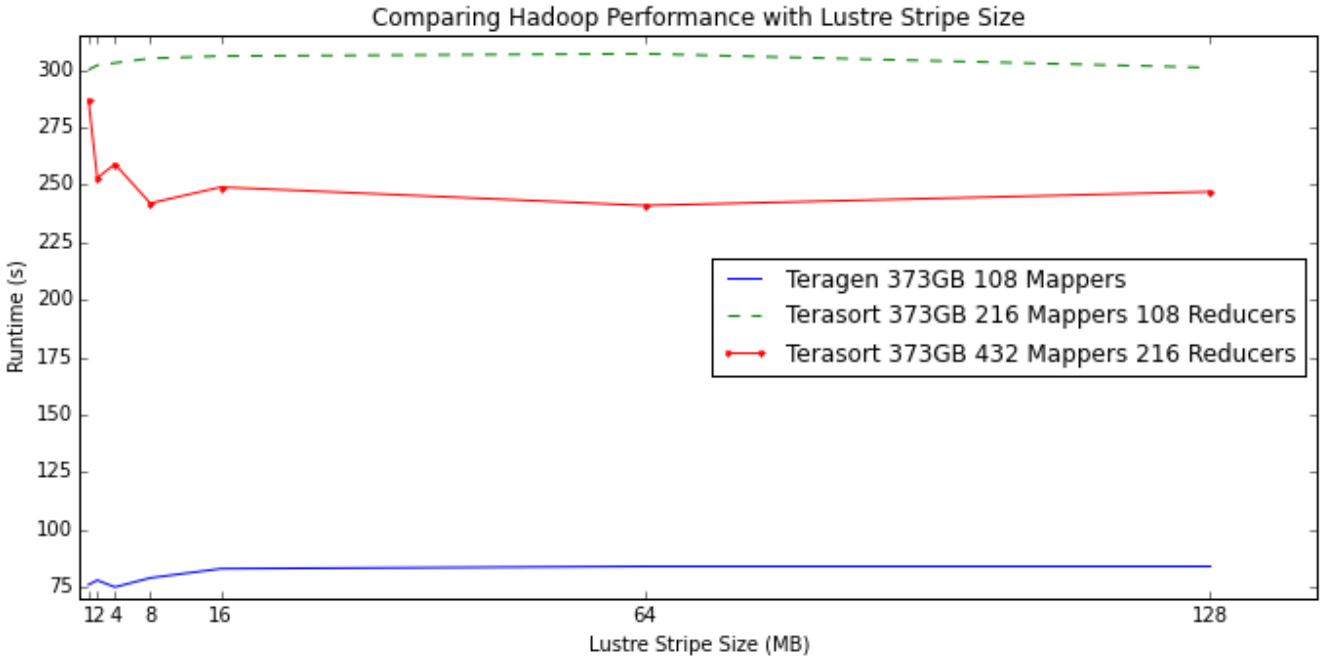
All remaining benchmarks presented were performed on Hadoop v2

Hadoop on Lustre Performance

Stripe Size Performance

Stripe size is not a strong factor in Hadoop performance, except perhaps for 1MB stripes in some cases

Recommend 4MB or greater



Hadoop on Lustre Performance

max_rpcs_in_flight and max_dirty_mb

Terasort performance on 373GB with 4MB stripe size, 216 Mappers and 108 Reducers

RPCs	8	16	128
Dirty MB	32	128	512
Time	5m04s	5m03s	5m07s

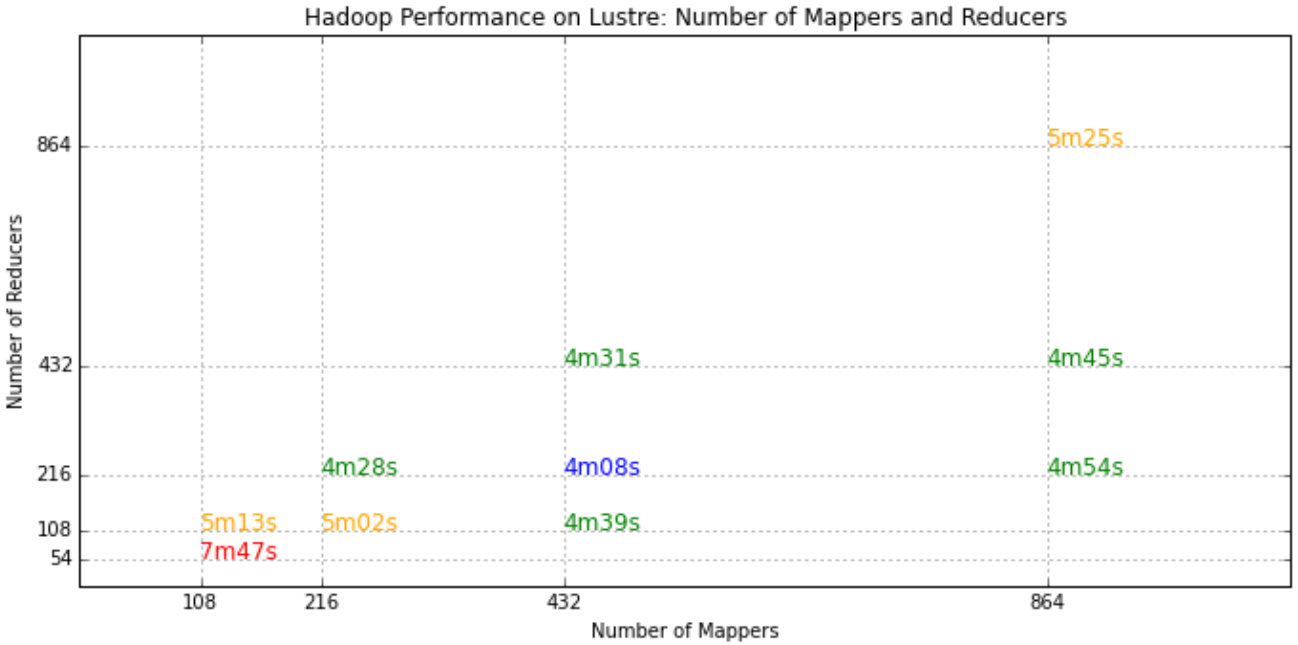
Changing these values does not strongly affect performance.

Hadoop on Lustre Performance

Number of Mappers and Reducers

Terasort 373GB

The number of mappers and reducers has a much stronger effect on performance than any Lustre client configuration

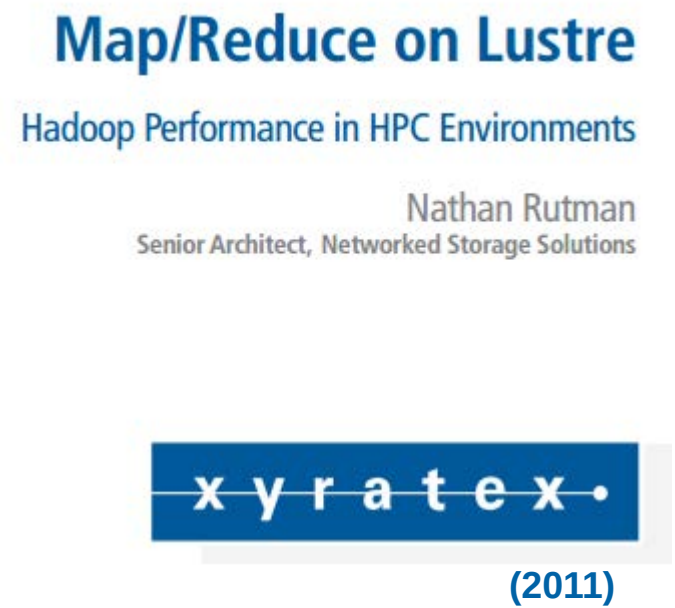


Hadoop on Lustre Performance

Some additional observations

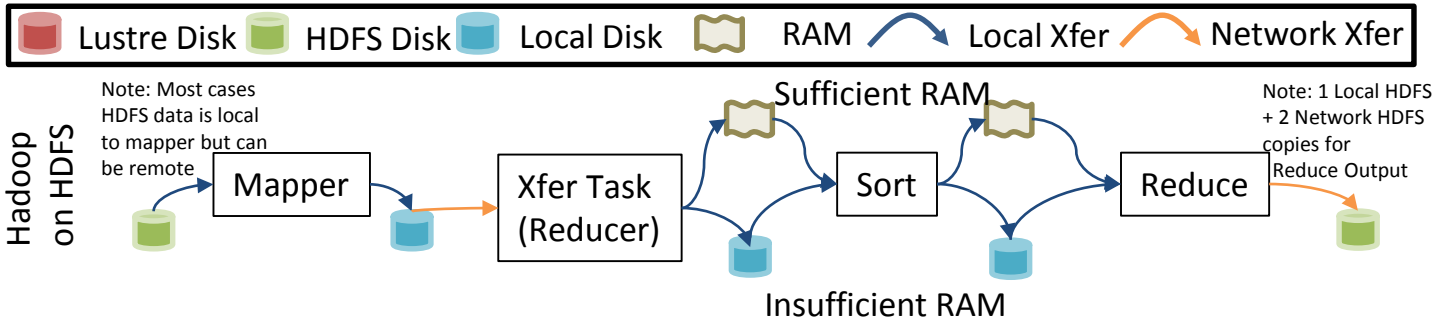
- **When it comes to Terasort, and likely other I/O intensive jobs, most performance gains come from tuning Hadoop rather than Lustre**
- **In contrast to HDFS, this means that Hadoop performance is less tied to how (or where, in the case of HDFS) the data is stored on disk, which gives users more options in how to tune their applications**
- **It's best to focus on tuning Hadoop parameters for best performance:**
  - We have seen that Lustre does better with fewer writers than the same job going to HDFS (e.g. number of Mappers for Teragen, or Reducers in Terasort)
  - A typical Hadoop rule of thumb is to build clusters, and run jobs, following the 1:1 core:HDD rule. With Hadoop on Lustre, this isn't necessarily true
  - Particularly for CPU-intensive jobs, performance tuning for Lustre is very similar to typical Hadoop on HDFS

Hadoop on Lustre Diskless Modification



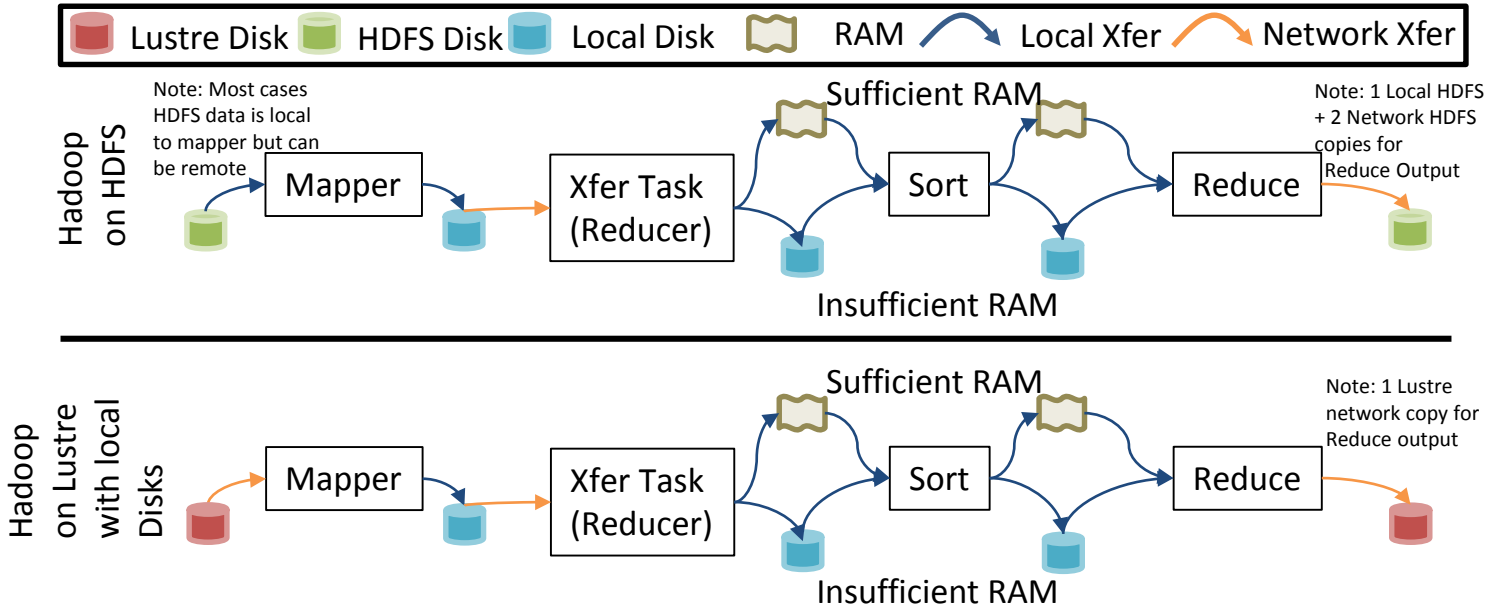
- **Standard Hadoop has local disks used for both HDFS and temporary data**
- **However, when using Hadoop on Lustre on diskless compute nodes, all data needs to go to Lustre, including temporary data.**
- **Beginning in 2011, we investigated an idea of how to optimize Map/Reduce performance when using Lustre as the temporary data store point**
- **This required a modification to core Hadoop functionality, and we released these changes at SC14:**
  - https://github.com/seagate/hadoop-on-lustre
  - https://github.com/seagate/hadoop-on-lustre2

Map/Reduce Data Flow Basics



- **HDFS data is split into blocks which is analogous to Lustre's stripe size. Default size is 128 MB for Hadoop 2**
- **The ResourceManager does best effort to assign computation to nodes that already have the data (Mapper)**
- **When done, the Mapper writes intermediate data to local scratch disk (no replication)**
- **Reducers collect data from Mappers (reading from local disk) over HTTP (called the "shuffle" phase) and sort incoming data either in RAM or on disk (again, local scratch disk)**
- **When data is sorted, the Reducer applies the reduction algorithm and writes output to HDFS that is 3x replicated**

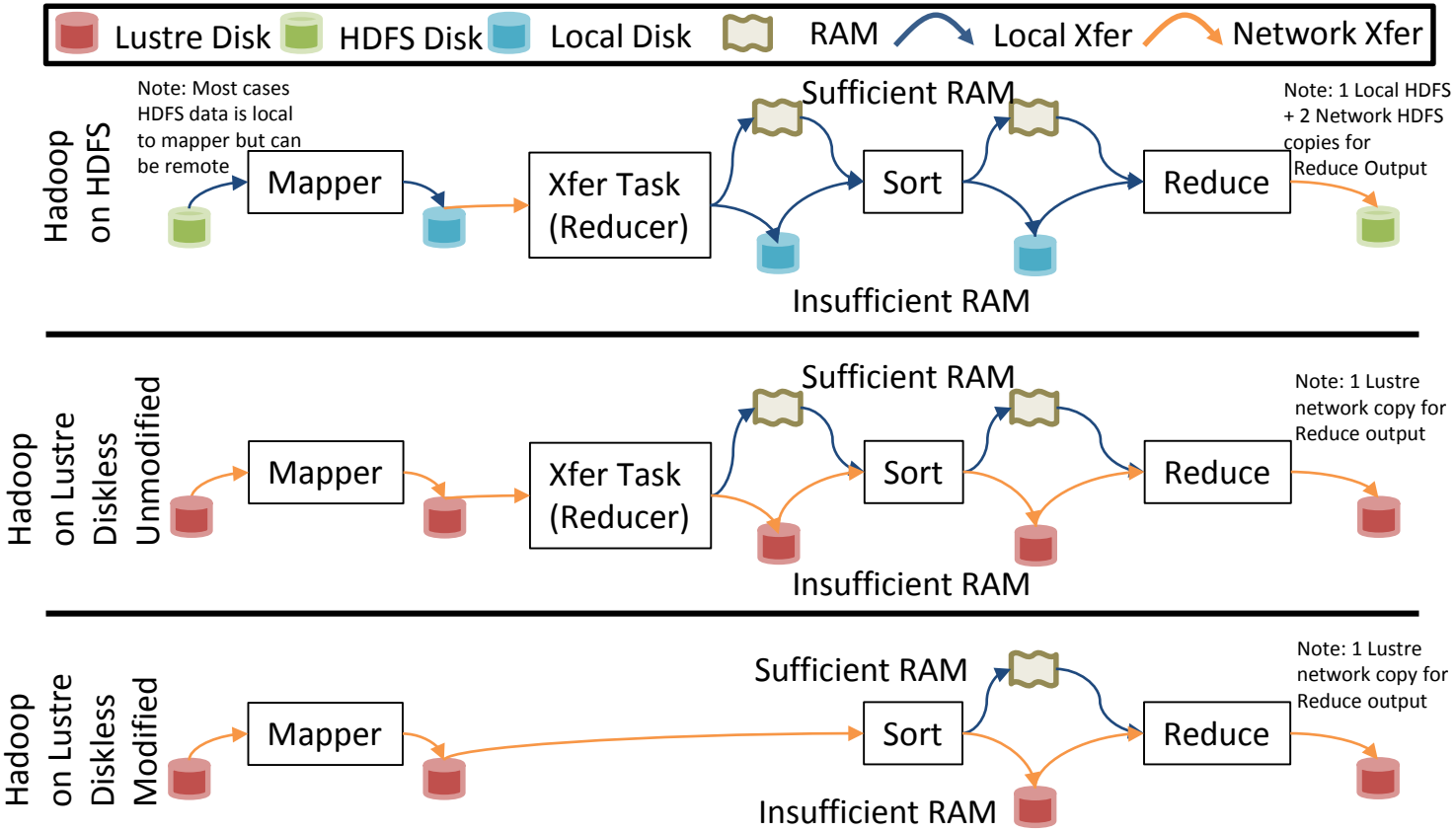
Map/Reduce Data Flow Basics



Using Lustre instead of HDFS is very similar, except the Mapper reads data from Lustre over the network, and the Reducer output is not triple replicated.

The lower diagram describes the setup for how we performed all the benchmarks we've presented so far, by using both Lustre and local disks to each Hadoop compute node.

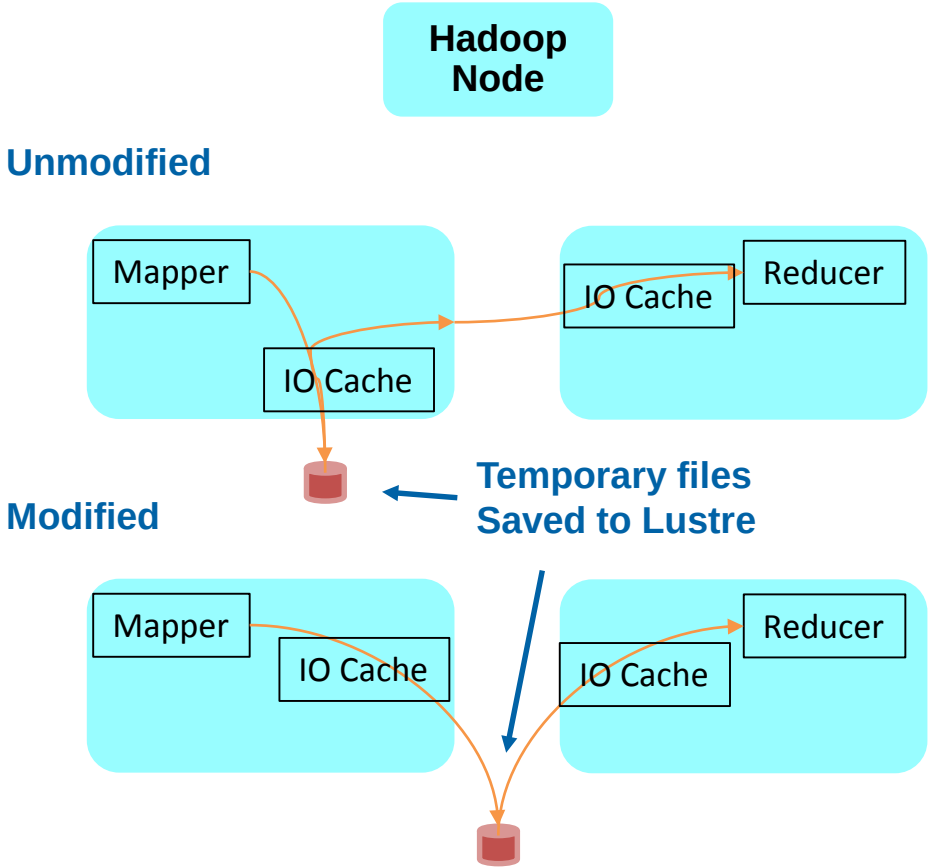
Hadoop on Lustre Map/Reduce Diskless Modification



Hadoop on Lustre Diskless Modification

Terasort Timings:

Size	Unmodified	Modified
100 GB	131s	155s
500 GB	538s	558s



The modified method is slower!

Takeaway: IO Caching speeds up the unmodified method.

Final Thoughts

- **We have explored three methods of Hadoop on Lustre:**
  - Using local disk(s) for temporary storage
  - Using Lustre for temporary storage with no modifications to Map/Reduce
  - Using Lustre for temporary storage, but with modifications to Map/Reduce
- **We find that each method above is slower than the ones above it**
- **The local disks do not need to be close to “big” (where “big” is the largest dataset analyzed), but it is most optimal to have some kind of local disk for temporary data**
- **Thank you to my collaborators Kelsie Betsch, Daniel Kaslovsky, Daniel Lingenfelter, Dimitar Vlassarev, and Zhenzhen Yan**