HaCeph: Scalable Metadata Management for Hadoop using Ceph

Esteban Molina-Estolano, Amandeep Khurana, Alex Nelson, Carlos Maltzahn, Ben Reed†, Scott Brandt

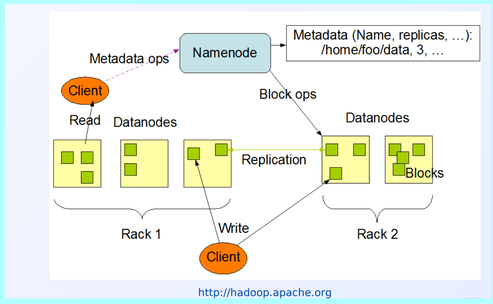
UC Santa Cruz, Yahoo! Inc.†

Motivation

- Hadoop is popular for large-scale data analysis
- Hadoop filesystems need to manage huge datasets with large numbers of files
- Scalability problem Hadoop's HDFS filesystem uses only one *namenode* for metadata
- Ceph filesystem: uses multiple metadata servers with dynamic subtree partitioning
- Goal: make Ceph available for use by Hadoop, with similar or better performance

HDFS scalability limitations

- Single NameNode must keep namespace and location of all data blocks in memory (about 150 bytes per file, directory, and data block)
- Adding nodes requires explicit rebalancing, with NameNode's cooperation
- NameNode must receive heartbeats from each DataNode
- NameNode must run re-replication on DataNode failure
- Slow create calls
- Current workaround: Hadoop Archive, a packed file format to reduce number of files in HDFS



Running Hadoop on Ceph

We have two ways of running Hadoop on Ceph.

Method 1: Mount Ceph with its kernel client, and run Hadoop on the "local" filesystem

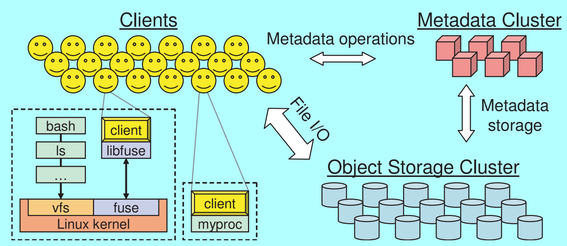
- Can be run without special support in Hadoop
- Accesses must go through the kernel
- Hadoop cannot exploit locality this way (yet)

Method 2: Use a JNI interface to add Ceph support (in development)

- Ceph patch exists for Hadoop trunk (submitted by Gregory Farnum)
- Must use JNI code, but runs entirely in userspace

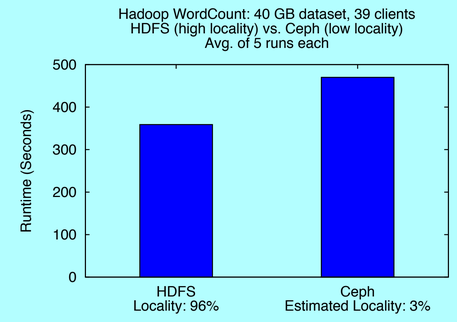
Ceph scalability advantages

- Ceph's metadata server is dynamically distributed over many nodes
- Ceph's *reliable object store* handles replication, rebalancing, and failure recovery in a peer-to-peer fashion
- Metadata server does not store block locations
- Data placement is computed, not stored; cluster state is compact
- Each file can specify its own striping strategy



Preliminary results

WordCount comparison: MapReduce WordCount job on 40 GB input file, across 39 storage/client nodes.



Ceph runs using kernel client, with no locality information. Any locality was coincidental.

- HDFS: only 4% of input data read over network
- Ceph: estimated 97% of input data read over network
- Task takes 31% longer on Ceph
- Prediction: adding locality information will close performance gap

Conclusions and Future Work

- Preliminary results show somewhat slower Ceph performance; expected, as we are not yet exploiting locality
- Allow Hadoop to exploit locality in Ceph
- Compare kernel client and JNI code performance
- Run benchmarks with huge numbers of files, to compare HDFS and Ceph namespace scalability

Acknowledgements

This work was supported in part by the Yahoo! Faculty Research and Engagement Award and the LANL/UCSC Institute for Scalable Scientific Data Management. We thank the Ceph development team for debugging assistance, and for work on the Hadoop/Ceph shim layer. We also thank the members of the UCSC Systems Research Lab whose advice helped guide this research.