

MDHIM: A Parallel Key/Value Store Framework for HPC

Hugh Greenberg

7/6/2015

LA-UR-15-25039

UNCLASSIFIED

HPC Clusters

- Managed by a job scheduler (e.g., Slurm, Moab)
  - Designed for running user jobs
  - Difficult to run system services

- Parallel file systems
  - High performing with N-N workloads
    - N application processes accessing N files simultaneously

HPC Clusters

- High speed interconnects
  - Infiniband
  - Cray Gemini

- Composed of high-end compute nodes
  - Server class hardware or better

HPC Clusters

- High speed interconnects
  - Infiniband
  - Cray Gemini

- Composed of high-end compute nodes
  - Server class hardware or better

Motivation

- Exascale
  - More processes performing file operations simultaneously
  - Less memory per CPU core

- Existing solutions
  - For cloud storage or web services
  - Do not efficiently utilize HPC environments and programming models

Motivation

- Existing solutions
  - Require long running daemons
  - TCP/IP only
  - Cannot be easily added to HPC applications
    - Lack C/C++ APIs
    - Require additional languages
  - E.g., Cassandra, Dynamo, HBase, Riak

Motivation

- Parallel Log Structured File System
  - Developed at LANL
  - Turns N-1 workloads into N-N
  - Requires each process to read a potentially large index into memory
  - Needed a scalable index

Solution

- MDHIM – Multi-Dimensional Hashing Indexing Middleware
- Distributed key/value store framework designed for HPC
- Written in HPC friendly programming model
  - MPI
  - Easily added to an MPI application

MDHIM - Features

- Doesn't require long running daemons
  - Servers (range servers) are spawned as separate threads
  - Starts with the application and dies with it
- Pluggable data stores
  - LevelDB as default
  - MySQL support
  - Not difficult to additional data stores

MDHIM - Features

- Bulk operations
  - Transfer large packets with many key/value pairs over a high-speed interconnect
- Multiple dimensions
  - The primary index
    - Key/value pairs with arbitrary values
    - Globally ordered
  - Secondary indexes
    - Keys with values that point to the keys of the primary index
    - Globally ordered or local to the range server

MDHIM – Global Indexes

- Supports global ordering
  - Keys can be retrieved in order
    - Order depends on key type
  - Each key maps to a single range server
  - Clients use the paritioner algorithm for the key location
    - Avoids querying range servers
    - Requires a single large data transfer of statistics data (mdhimStatFlush)

MDHIM – Global Indexes

- Cursor operations
  - Get next or previous key
  - Traverses range servers
  - Requires a single large data transfer of statistics data (mdhimStatFlush)

MDHIM- Local Indexes

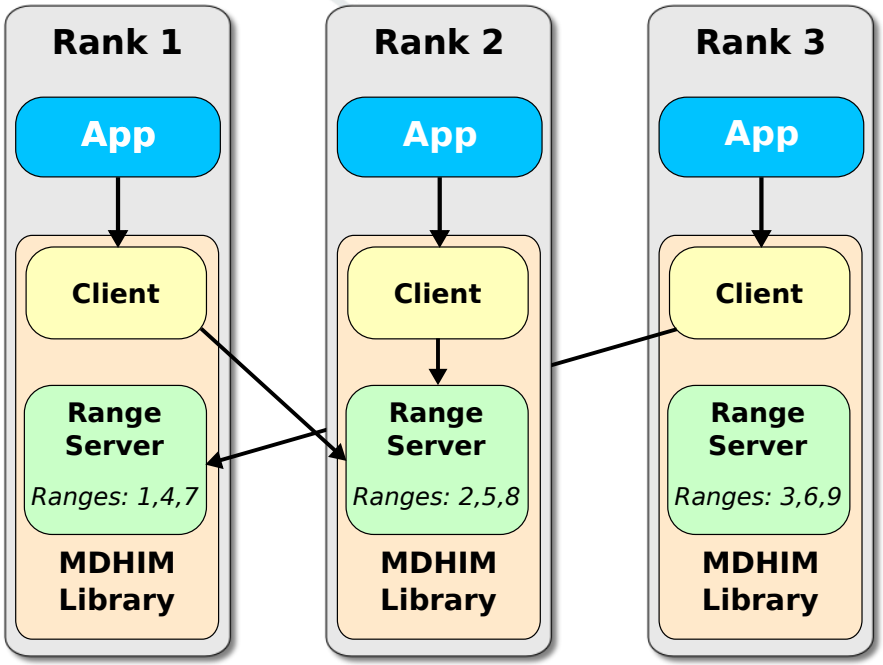
- Supports local indexes
  - Each rank can store key/value pairs to itself
  - Lookups require querying multiple servers
    - mdhimStatFlush can help to reduce the number of queries

MDHIM - Partitioning

- Built-in partitioning algorithm with reasonable defaults
- Pluggable partitioning planned
  - User defined functions for mapping of keys to range servers

MDHIM - Design

- MDHIM contains a client and range server
- Each rank in this image is running a range server.
- Clients use the partitioner to determine which range server to contact



MDHIM software design

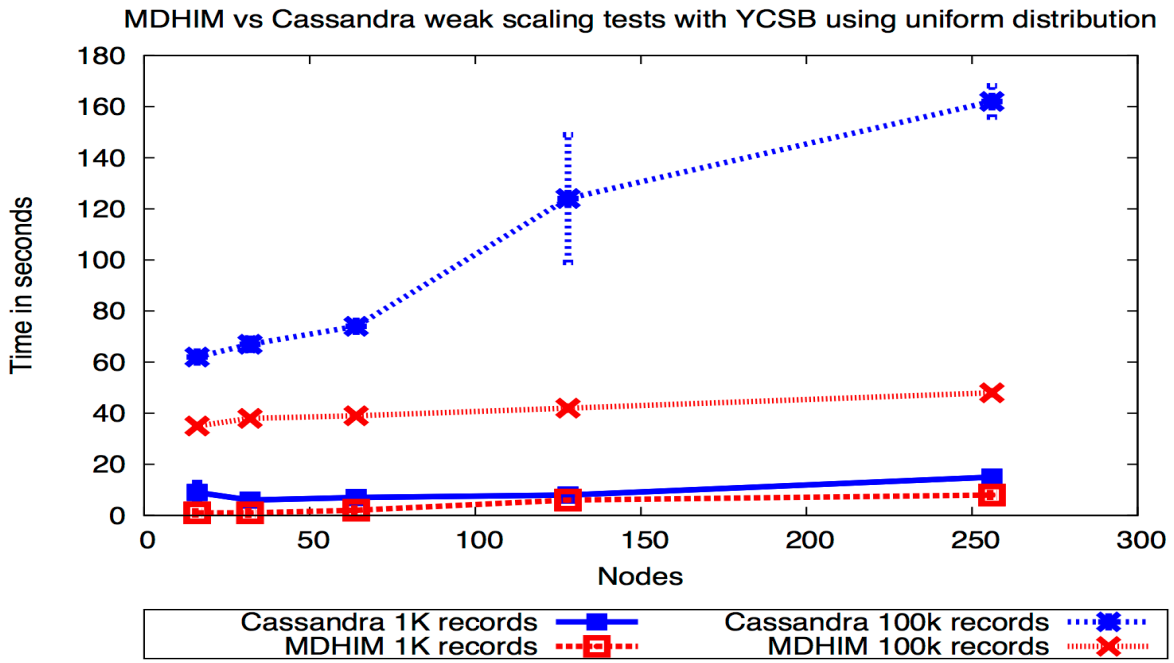
MDHIM - Evaluation

- Compared against Cassandra
- Used the Yahoo Cloud Serving Benchmark
  - Created an MDHIM plugin
  - Used built-in Cassandra plugin
  - Random integers as keys
- Tests performed on LANL Mustang Cluster
  - 2 AMD 12-core MagnyCours
  - 64GB of memory per node
  - 1600 nodes

MDHIM - Evaluation

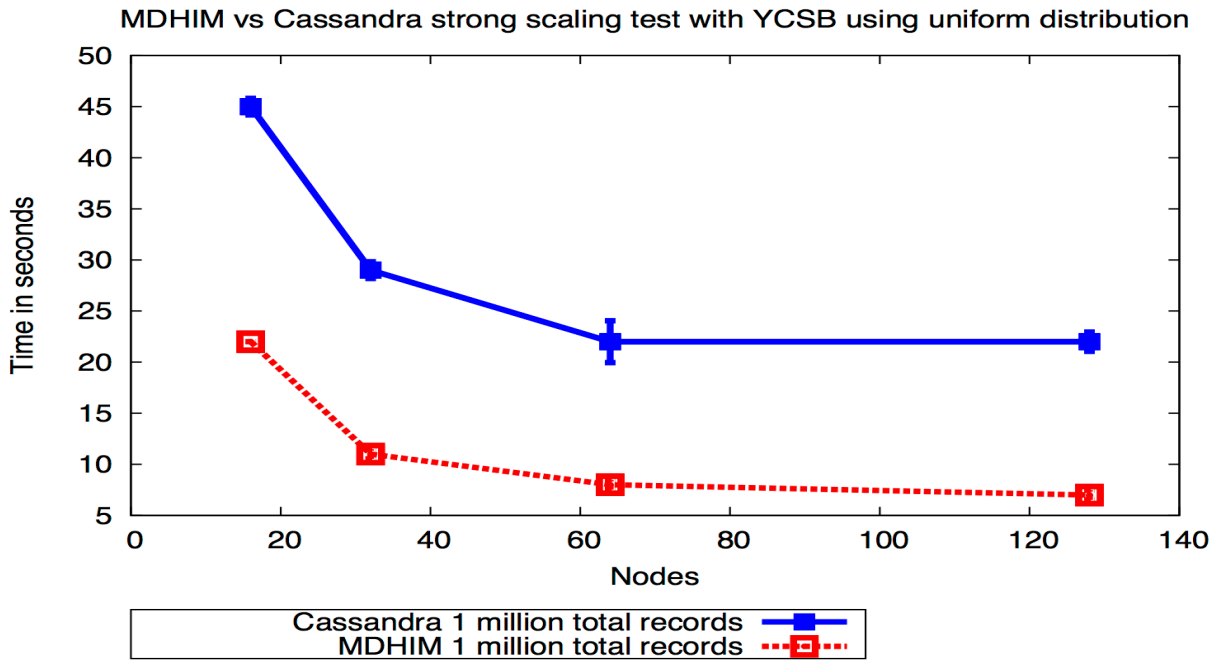
- Cassandra used IP over Infiniband
- MDHIM used native Infinband
- Tuned Cassandra and LevelDB to use 50MB of memory
- Cassandra configured to use batch mode
  - Default periodic
  - Forces Cassandra to wait until data is synced to disk before returning
  - Matches MDHIM/LevelDB

MDHIM - Evaluation



Two types of tests are represented: 1K records per node and 100k records per process. Three runs were performed at each point.

MDHIM - Evaluation



1 million records inserted/retrieved in total for each run. Three runs were performed for each data point.

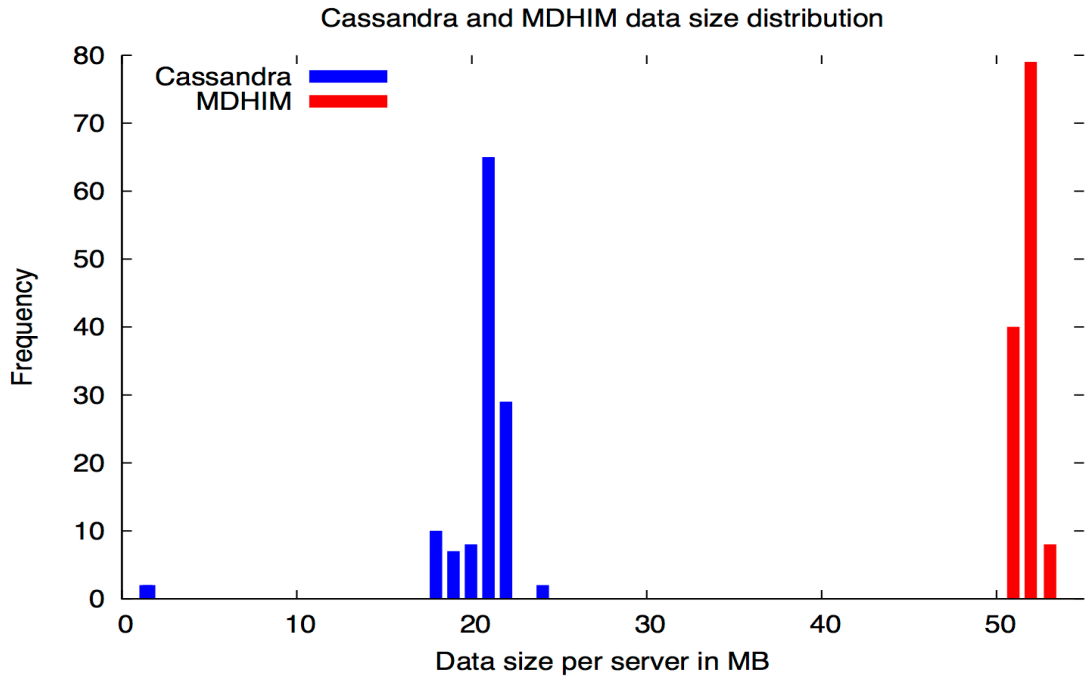
MDHIM - Evaluation

- MDHIM performs slightly better than Cassandra for the 1K records per nodes weak scaling test
- MDHIM out performs Cassandra for the 100K records per node test and the strong scaling test
  - 3.375 times faster with 256 processes

MDHIM - Evaluation

- Reasons for MDHIM's performance
  - Native Infiniband support
  - Better key distribution
  - C vs Java

MDHIM - Evaluation



The frequency of database sizes for Cassandra and MDHIM after a run with 128 nodes and 100K records inserted per node.

Conclusion

- MDHIM is a parallel key/value store framework for HPC
- Designed for HPC systems and job schedulers
  - Utilizes high speed interconnects and MPI
  - Easily added to a scientific application
- Outperformed Cassandra in all tests with the Yahoo Cloud Serving Benchmark (YCSB)

Thank you

- Code
  - http://github.com/mdhim/mdhim-tng

- Contact
  - Hugh Greenberg
  - hng@lanl.gov