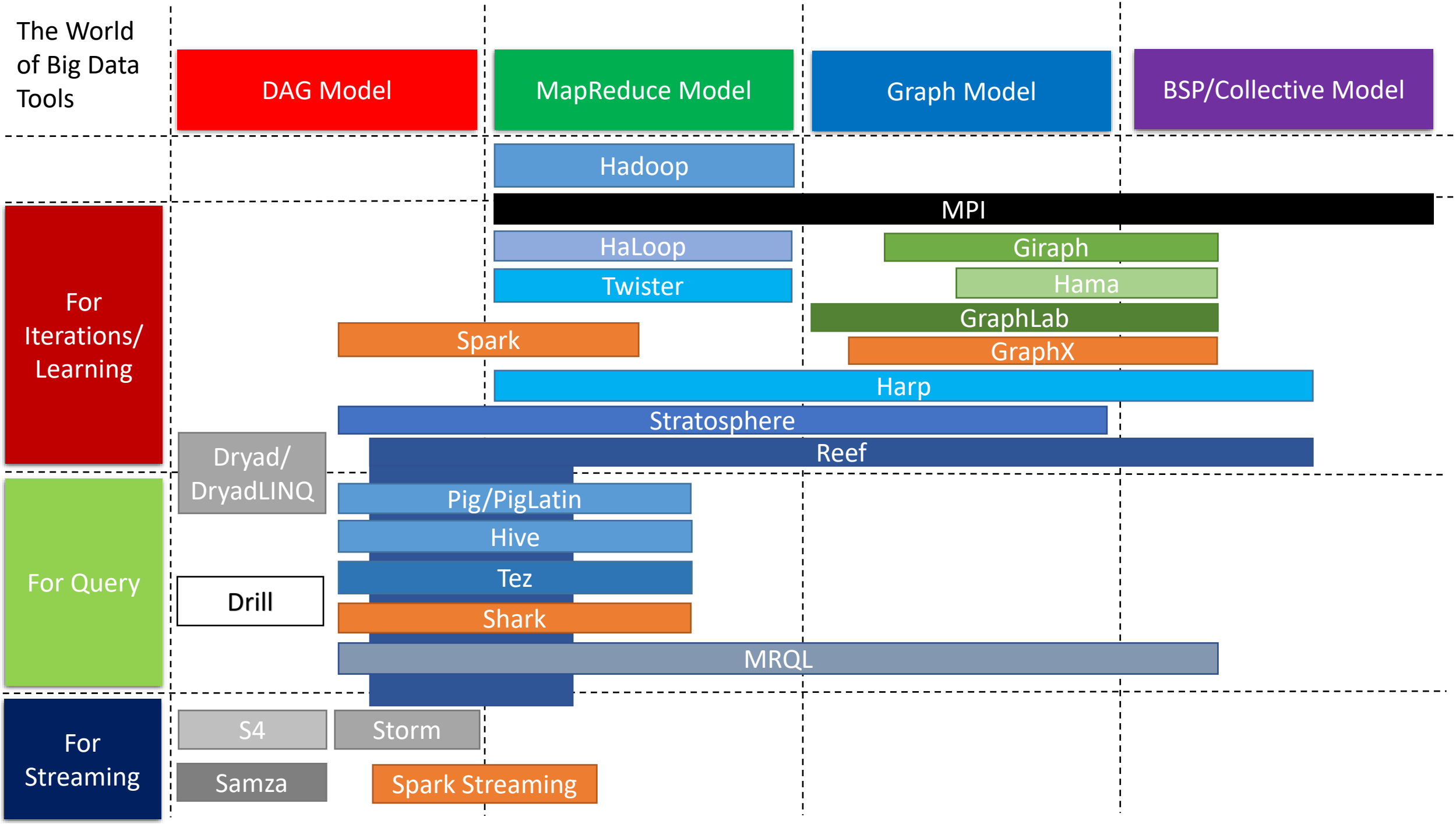

A Brief Introduction of
Existing Big Data Tools

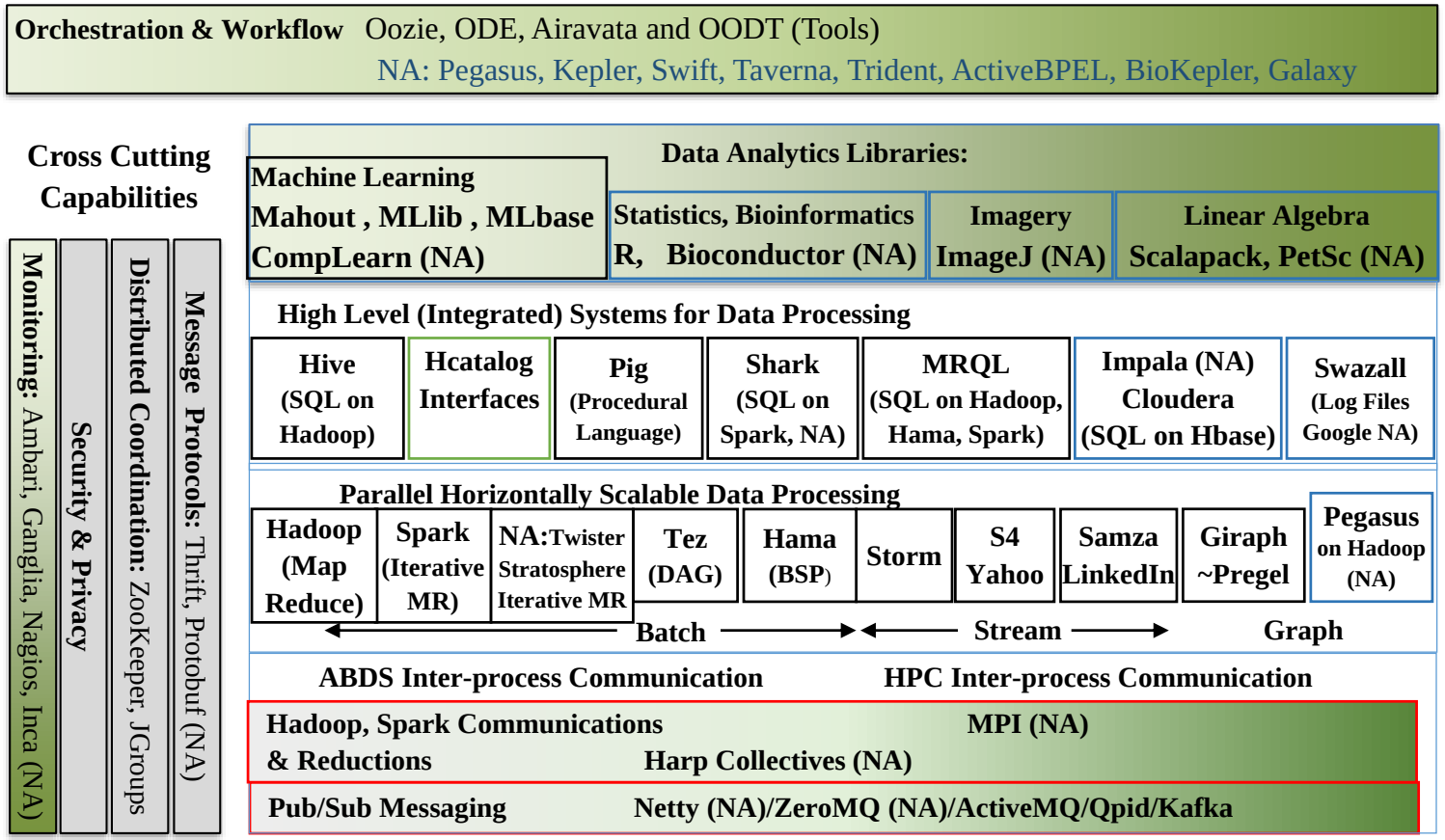
Outline

- The world map of big data tools
- Layered architecture
- Big data tools for HPC and supercomputing
  - MPI
- Big data tools on clouds
  - MapReduce model
  - Iterative MapReduce model
  - DAG model
  - Graph model
  - Collective model
- Machine learning on big data
- Query on big data
- Stream data processing


The World of Big Data Tools
DAG Model
MapReduce Model
Graph Model
BSP/Collective Model
Hadoop
MPI
For Iterations/ Learning
HaLoop
Giraph
Twister
Hama
GraphLab
Spark
GraphX
Harp
Stratosphere
Dryad/ DryadLINQ
Reef
Pig/PigLatin
For Query
Hive
Tez
Drill
Shark
MRQL
For Streaming
S4
Storm
Samza
Spark Streaming

Layered Architecture (Upper)

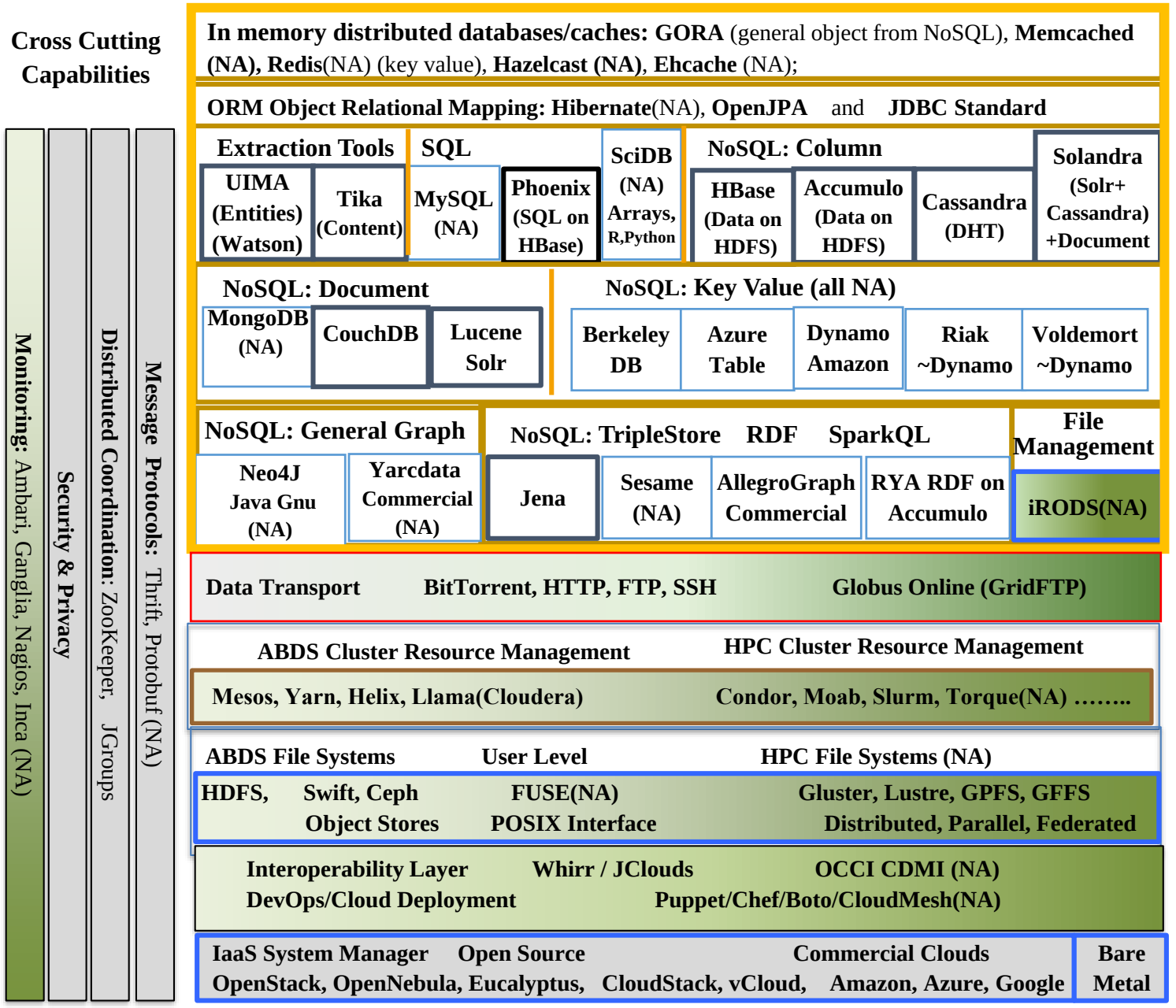
- NA – Non Apache projects
- Green layers are Apache/Commercial Cloud (light) to HPC (darker) integration layers



The figure of layered architecture is from Prof. Geoffrey Fox

Layered Architecture (Lower)

- NA – Non Apache projects
- Green layers are Apache/Commercial Cloud (light) to HPC (darker) integration layers



The figure of layered architecture is from Prof. Geoffrey Fox

Big Data Tools for HPC and Supercomputing

- MPI(Message Passing Interface, 1992)
  - Provide standardized function interfaces for communication between parallel processes.
- Collective communication operations
  - Broadcast, Scatter, Gather, Reduce, Allgather, Allreduce, Reduce-scatter.
- Popular implementations
  - MPICH (2001)
  - OpenMPI (2004)
    - http://www.open-mpi.org/

MapReduce Model

- Google MapReduce (2004)
  - Jeffrey Dean et al. MapReduce: Simplified Data Processing on Large Clusters. OSDI 2004.

- Apache Hadoop (2005)

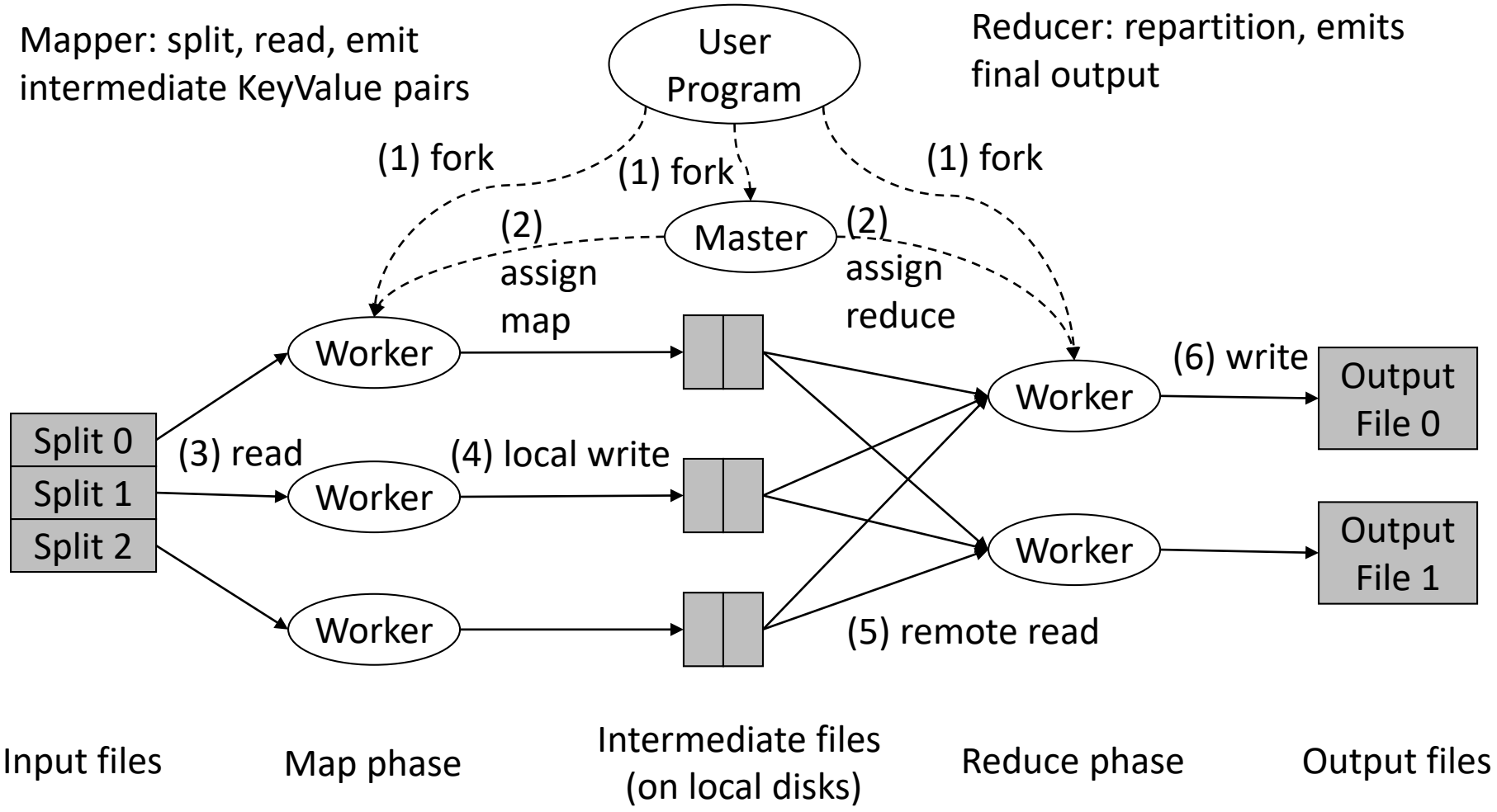
  - http://hadoop.apache.org/
  - http://developer.yahoo.com/hadoop/tutorial/

- Apache Hadoop 2.0 (2012)
  - Vinod Kumar Vavilapalli et al. Apache Hadoop YARN: Yet Another Resource Negotiator, SOCC 2013.
  - Separation between resource management and computation model.

Key Features of MapReduce Model

- Designed for clouds
  - Large clusters of commodity machines
- Designed for big data
  - Support from local disks based distributed file system (GFS / HDFS)
  - Disk based intermediate data transfer in Shuffling
- MapReduce programming model
  - Computation pattern: Map tasks and Reduce tasks
  - Data abstraction: KeyValue pairs

Google MapReduce



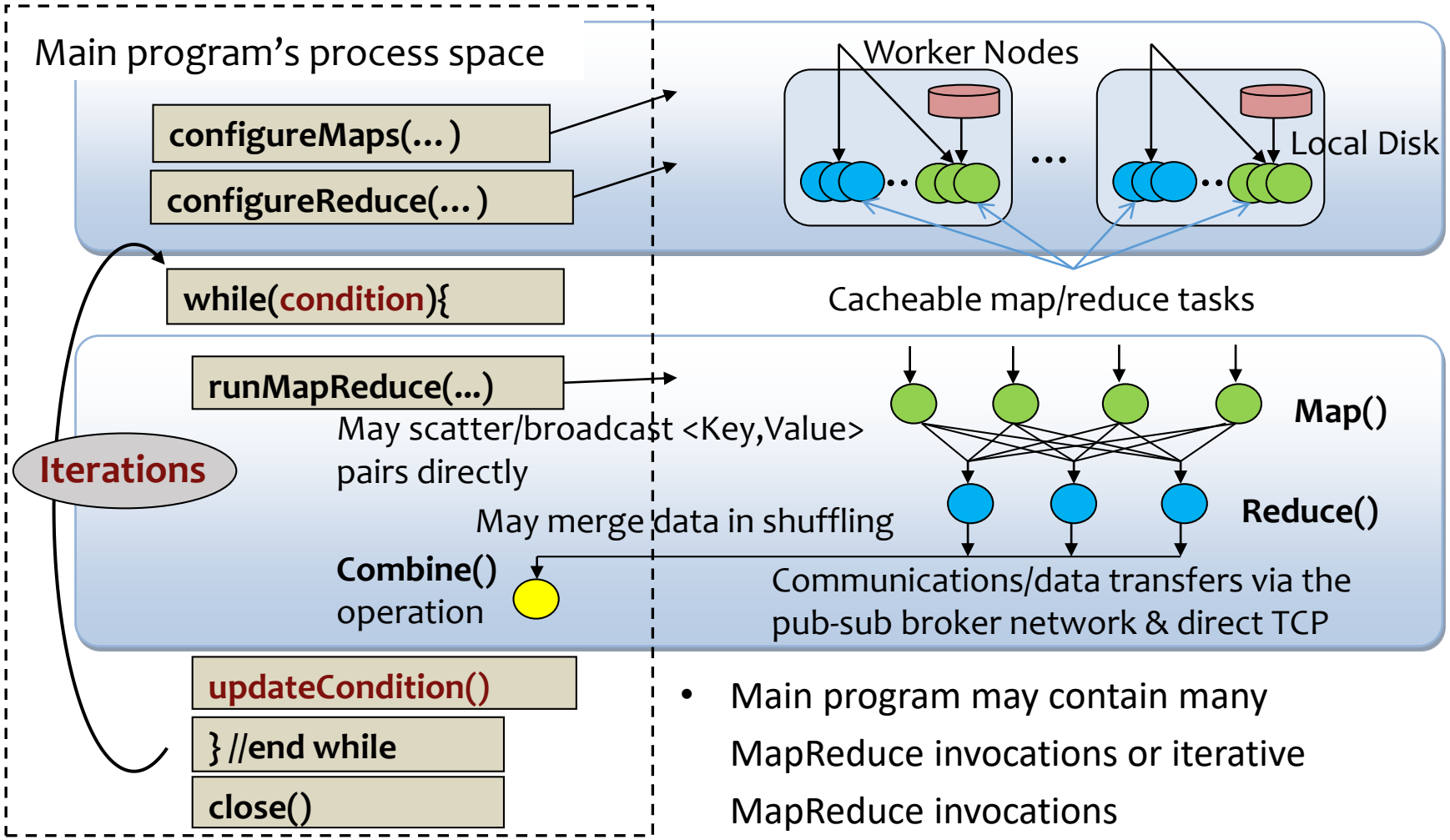
Iterative MapReduce Model

- Twister (2010)

  - Jaliya Ekanayake et al. Twister: A Runtime for Iterative MapReduce. HPDC workshop 2010.
  - http://www.iterativemapreduce.org/
  - Simple collectives: broadcasting and aggregation.

- HaLoop (2010)
  - Yingyi Bu et al. HaLoop: Efficient Iterative Data Processing on Large clusters. VLDB 2010.
  - http://code.google.com/p/haloop/
  - Programming model $R_{i+1} = R_0 \cup (R_i \bowtie L)$
  - Loop-Aware Task Scheduling
  - Caching and indexing for Loop-Invariant Data on local disk

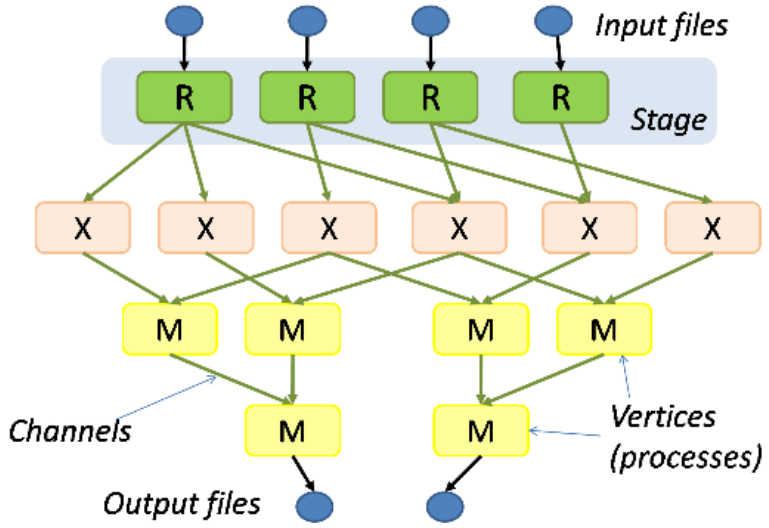
Twister Programming Model



- Main program may contain many MapReduce invocations or iterative MapReduce invocations

DAG (Directed Acyclic Graph) Model

- Dryad and DryadLINQ (2007)
  - Michael Isard et al. Dryad: Distributed Data-Parallel Programs from Sequential Building Blocks, EuroSys, 2007.
  - http://research.microsoft.com/en-us/collaboration/tools/dryad.aspx



Model Composition

- Apache Spark (2010) 
  - Matei Zaharia et al. Spark: Cluster Computing with Working Sets,. HotCloud 2010.
  - Matei Zaharia et al. Resilient Distributed Datasets: A Fault-Tolerant Abstraction for In-Memory Cluster Computing. NSDI 2012.
  - http://spark.apache.org/
  - Resilient Distributed Dataset (RDD)
  - RDD operations
    - MapReduce-like parallel operations
  - DAG of execution stages and pipelined transformations
  - Simple collectives: broadcasting and aggregation

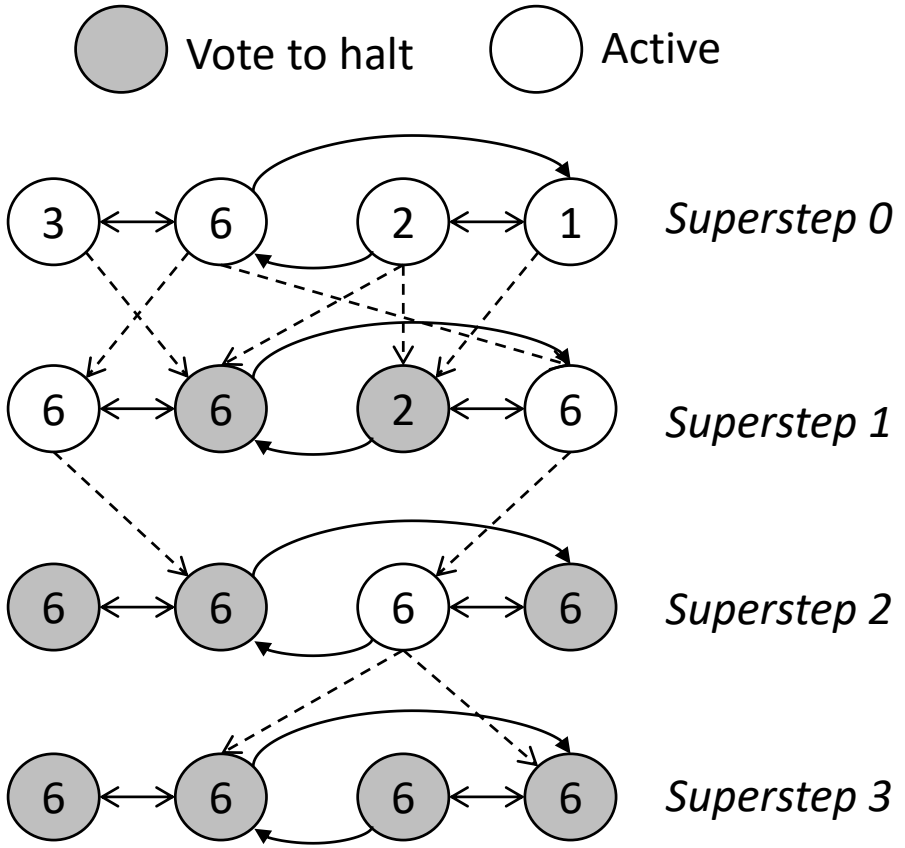
Graph Processing with BSP model

- Pregel (2010)
  - Grzegorz Malewicz et al. Pregel: A System for Large-Scale Graph Processing. SIGMOD 2010.
- Apache Hama (2010)
  - https://hama.apache.org/

- Apache Giraph (2012)
  - https://giraph.apache.org/
  - Scaling Apache Giraph to a trillion edges
    - https://www.facebook.com/notes/facebook-engineering/scaling-apache-giraph-to-a-trillion-edges/10151617006153920

Pregel & Apache Giraph

- Computation Model
  - Superstep as iteration
  - Vertex state machine:
    Active and Inactive, vote to halt
  - Message passing between vertices
  - Combiners
  - Aggregators
  - Topology mutation
- Master/worker model
- Graph partition: hashing
- Fault tolerance: checkpointing and confined recovery



Maximum Value Example

Giraph Page Rank Code Example

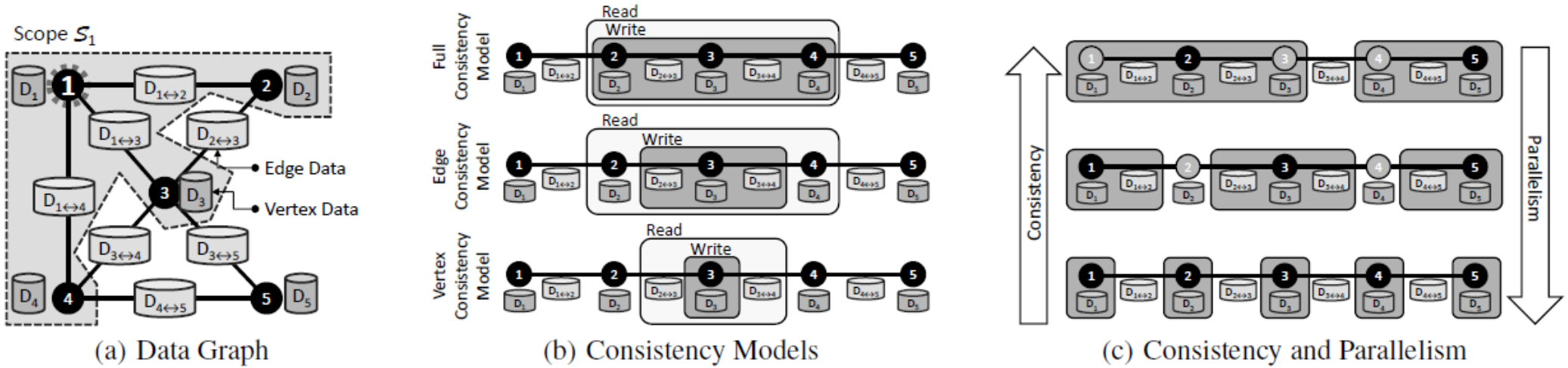
```
public class PageRankComputation
    extends BasicComputation<IntWritable, FloatWritable, NullWritable, FloatWritable> {
  /** Number of supersteps */
  public static final String SUPERSTEP_COUNT = "giraph.pageRank.superstepCount";

  @Override
  public void compute(Vertex<IntWritable, FloatWritable, NullWritable> vertex, Iterable<FloatWritable> messages)
      throws IOException {
    if (getSuperstep() >= 1) {
      float sum = 0;
      for (FloatWritable message : messages) {
        sum += message.get();
      }
      vertex.getValue().set((0.15f / getTotalNumVertices()) + 0.85f * sum);
    }
    if (getSuperstep() < getConf().getInt(SUPERSTEP_COUNT, 0)) {
      sendMessageToAllEdges(vertex,
          new FloatWritable(vertex.getValue().get() / vertex.getNumEdges()));
    } else {
      vertex.voteToHalt();
    }
  }
}
```

GraphLab (2010)

- Yucheng Low et al. GraphLab: A New Parallel Framework for Machine Learning. UAI 2010.
- Yucheng Low, et al. Distributed GraphLab: A Framework for Machine Learning and Data Mining in the Cloud. PVLDB 2012.
- http://graphlab.org/projects/index.html
- http://graphlab.org/resources/publications.html

- Data graph
- Update functions and the scope
- Sync operation (similar to aggregation in Pregel)

Data Graph



(a) Data Graph (b) Consistency Models (c) Consistency and Parallelism

Figure 2: (a) The data graph and scope $\mathcal{S}_1$. Gray cylinders represent the user defined vertex and edge data while the irregular region containing the vertices $\{1, 2, 3, 4\}$ is the scope, $\mathcal{S}_1$ of vertex 1. An update function applied to vertex 1 is able to read and modify all the data in $\mathcal{S}_1$ (vertex data D_1, D_2, D_3, and D_4 and edge data $D_{1\leftrightarrow 2}$, $D_{1\leftrightarrow 3}$, and $D_{1\leftrightarrow 4}$). (b) The read and write permissions for an update function executed on vertex 3 under each of the consistency models. Under the full consistency model the update function has complete read-write access to its entire scope. Under the edge consistency model, the update function has only read access to adjacent vertices. Finally, vertex consistency model only provides write access to the central vertex data. (c) The trade-off between consistency and parallelism. The dark rectangles denote the write-locked regions that cannot overlap. Update functions are executed on the dark vertices in parallel. Under the stronger consistency models fewer functions can run simultaneously.

Vertex-cut v.s. Edge-cut

- PowerGraph (2012)
  - Joseph E. Gonzalez et al. PowerGraph: Distributed Graph-Parallel Computation on Natural Graphs. OSDI 2012.
  - Gather, apply, Scatter (GAS) model
- GraphX (2013)
  - Reynold Xin et al. GraphX: A Resilient Distributed Graph System on Spark. GRADES (SIGMOD workshop) 2013.
  - https://amplab.cs.berkeley.edu/publication/graphx-grades/

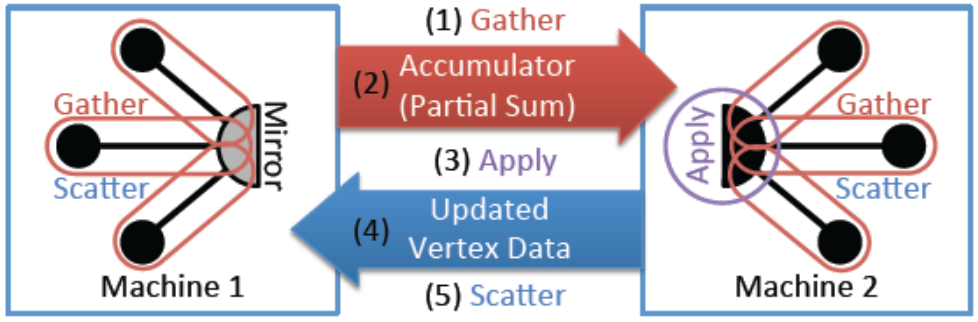
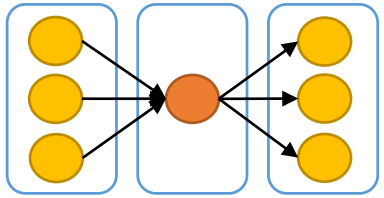
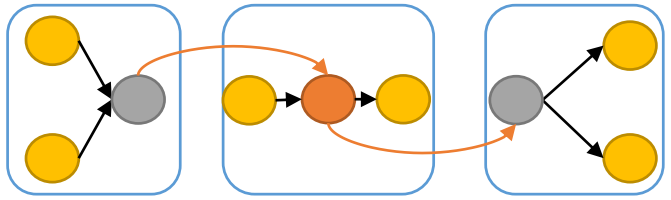


Figure 5: The communication pattern of the PowerGraph abstraction when using a vertex-cut. Gather function runs locally on each machine and then one accumulators is sent from each mirror to the master. The master runs the apply function and then sends the updated vertex data to all mirrors. Finally the scatter phase is run in parallel on mirrors.



Edge-cut (Giraph model)



Vertex-cut (GAS model)

To reduce communication overhead….

- Option 1
  - Algorithmic message reduction
  - Fixed point-to-point communication pattern

- Option 2
  - Collective communication optimization
  - Not considered by previous BSP model but well developed in MPI
  - Initial attempts in Twister and Spark on clouds
    - Mosharaf Chowdhury et al. Managing Data Transfers in Computer Clusters with Orchestra. SIGCOMM 2011.
    - Bingjing Zhang, Judy Qiu. High Performance Clustering of Social Images in a Map-Collective Programming Model. SOCC Poster 2013.

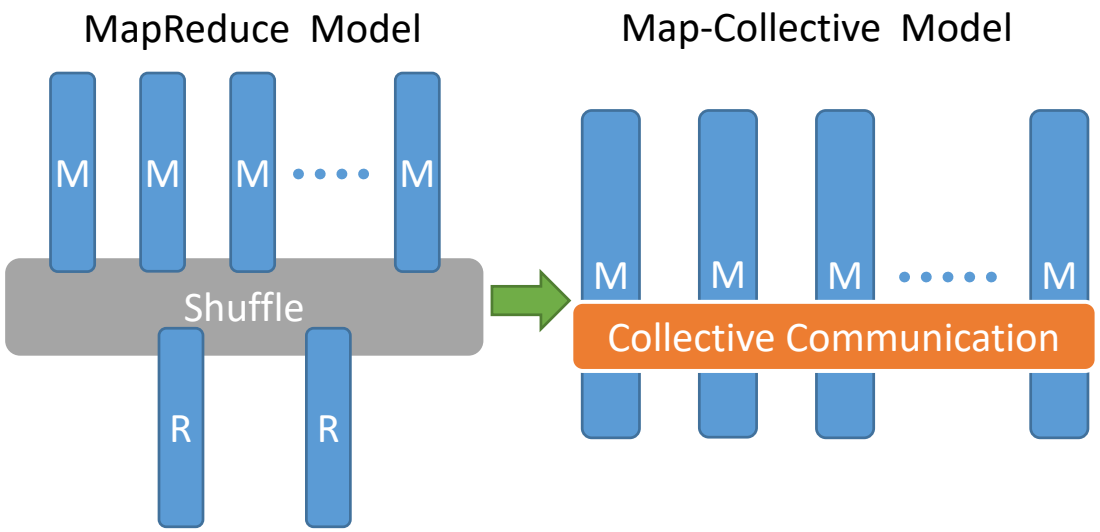
Collective Model

- Harp (2013) 

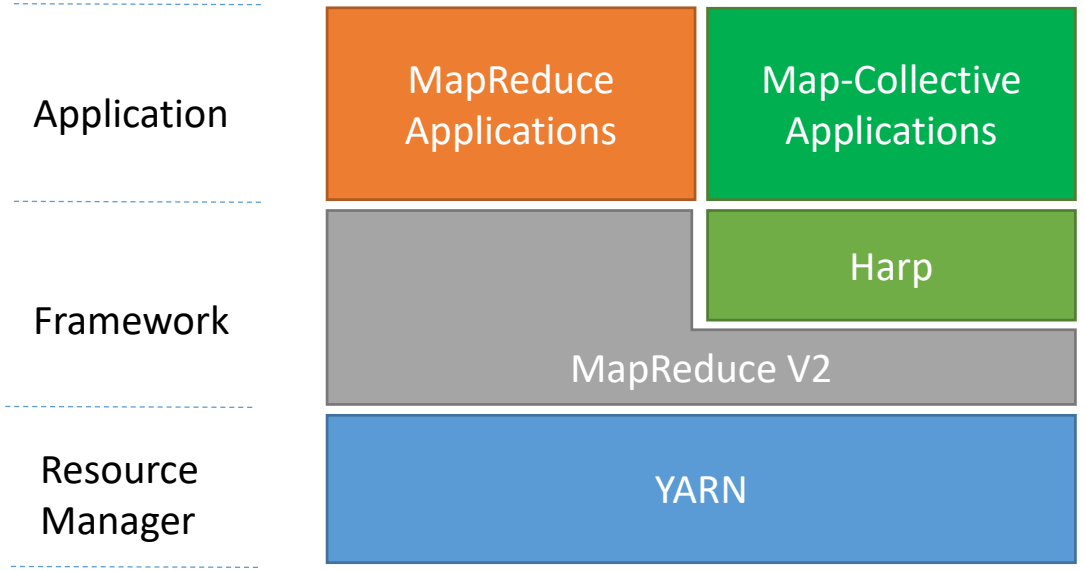
  - https://github.com/jessezbj/harp-project
  - Hadoop Plugin (on Hadoop 1.2.1 and Hadoop 2.2.0)
  - Hierarchical data abstraction on arrays, key-values and graphs for easy programming expressiveness.
  - Collective communication model to support various communication operations on the data abstractions.
  - Caching with buffer management for memory allocation required from computation and communication
  - BSP style parallelism
  - Fault tolerance with check-pointing

Harp Design

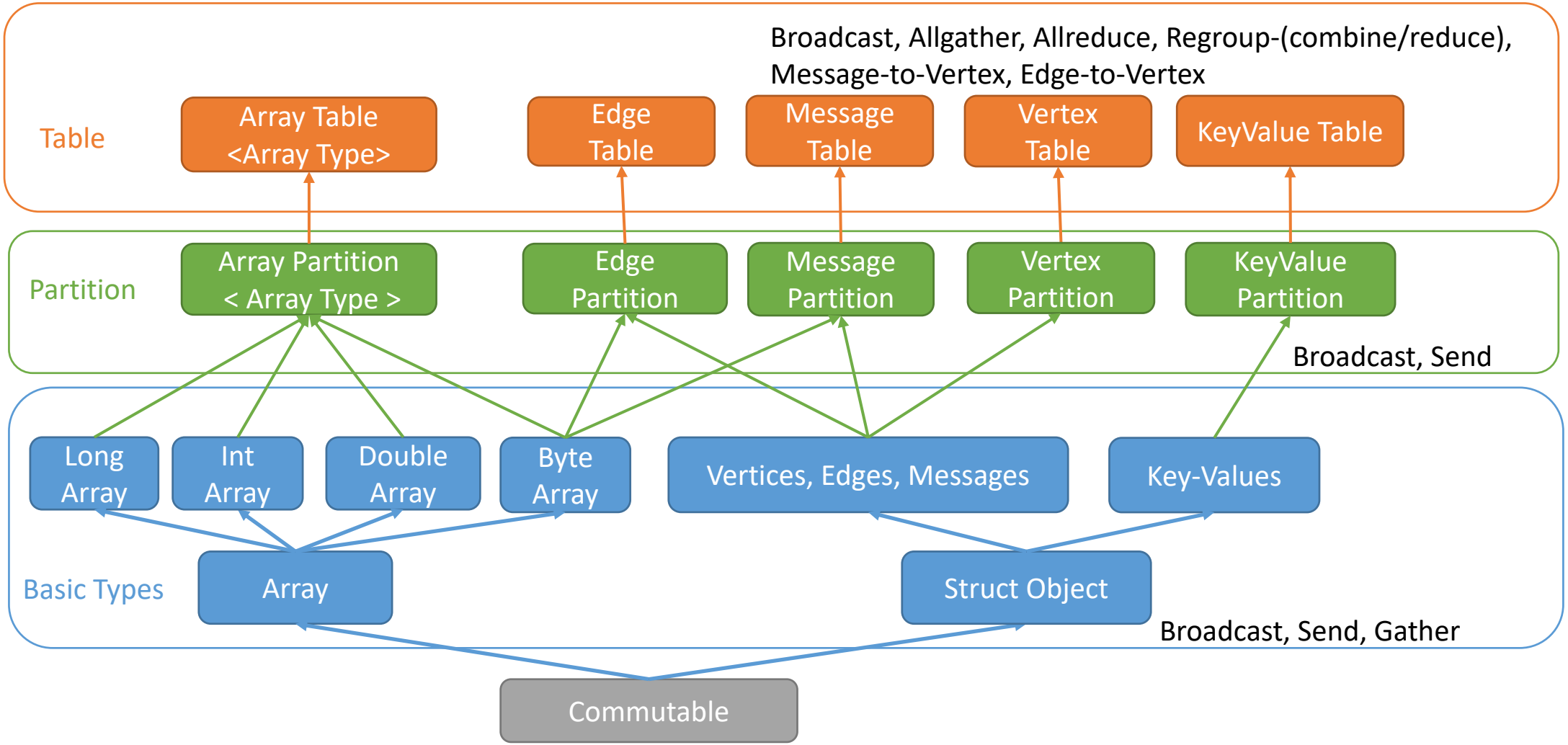
Parallelism Model



Architecture



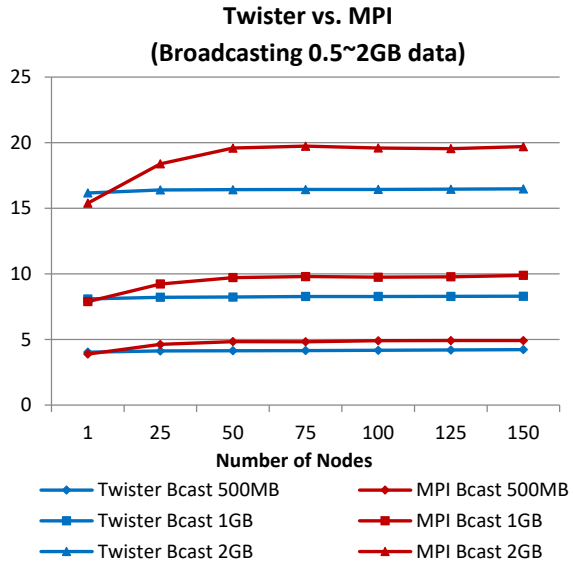
Hierarchical Data Abstraction and Collective Communication

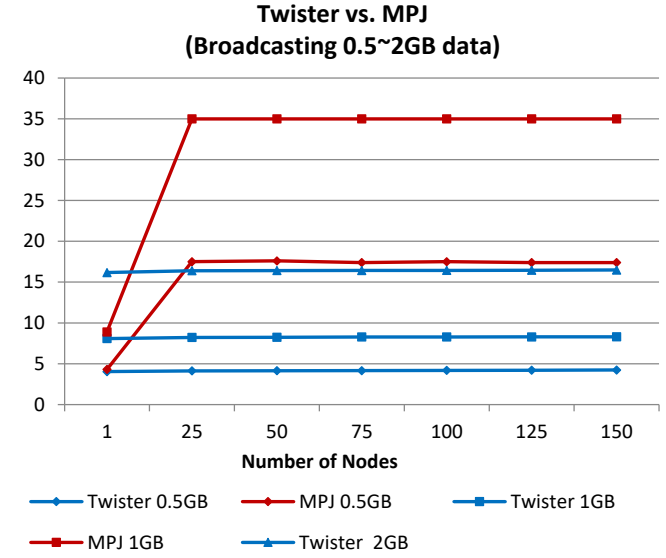


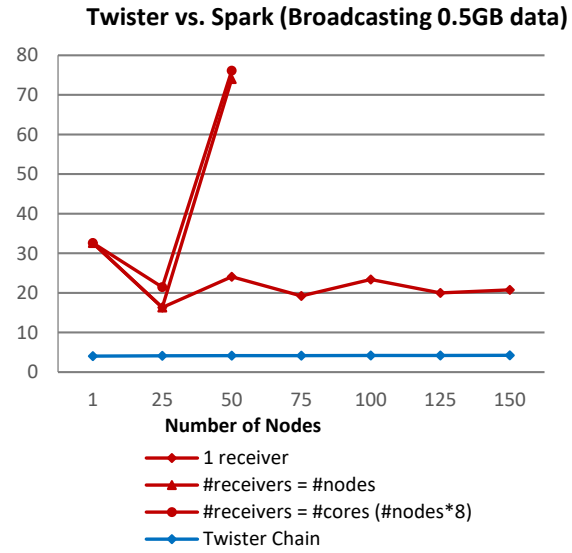
Harp Bcast Code Example

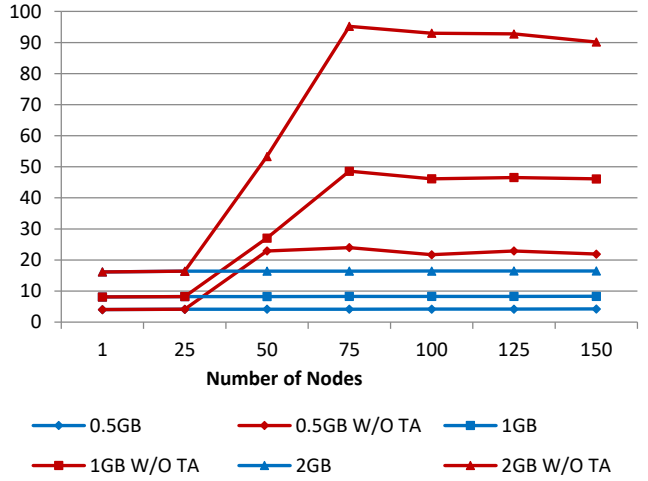
```
protected void mapCollective(KeyValReader reader, Context context)
  throws IOException, InterruptedException {

  ArrTable<DoubleArray, DoubleArrPlus> table =
    new ArrTable<DoubleArray, DoubleArrPlus>(0, DoubleArray.class, DoubleArrPlus.class);

  if (this.isMaster()) {
    String cFile = conf.get(KMeansConstants.CFILE);
    Map<Integer, DoubleArray> cenDataMap = createCenDataMap(cParSize, rest, numCenPartitions,
      vectorSize, this.getResourcePool());
    loadCentroids(cenDataMap, vectorSize, cFile, conf);
    addPartitionMapToTable(cenDataMap, table);
  }

  arrTableBcast(table);
}
```

Pipelined Broadcasting with Topology-Awareness



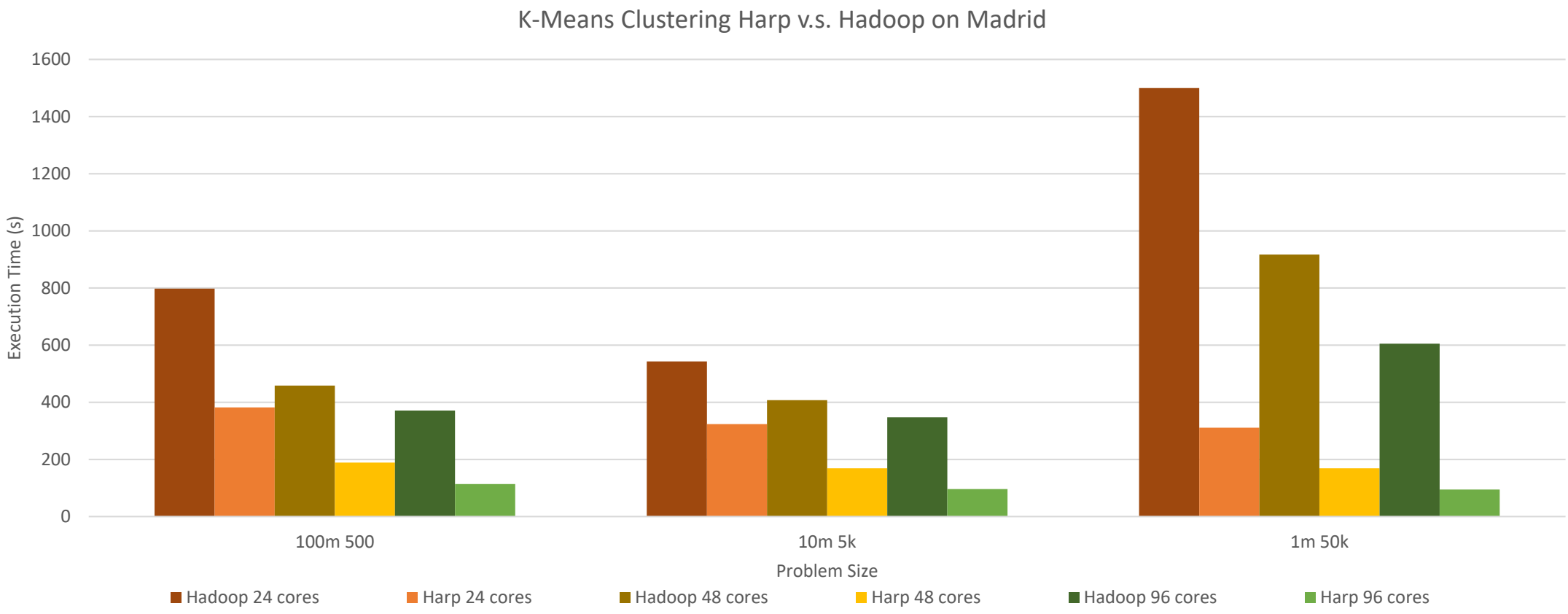






Tested on IU Polar Grid with 1 Gbps Ethernet connection

K-Means Clustering Performance on Madrid Cluster (8 nodes)



K-means Clustering Parallel Efficiency

- Shantenu Jha et al. A Tale of Two Data-Intensive Paradigms: Applications, Abstractions, and Architectures. 2014.

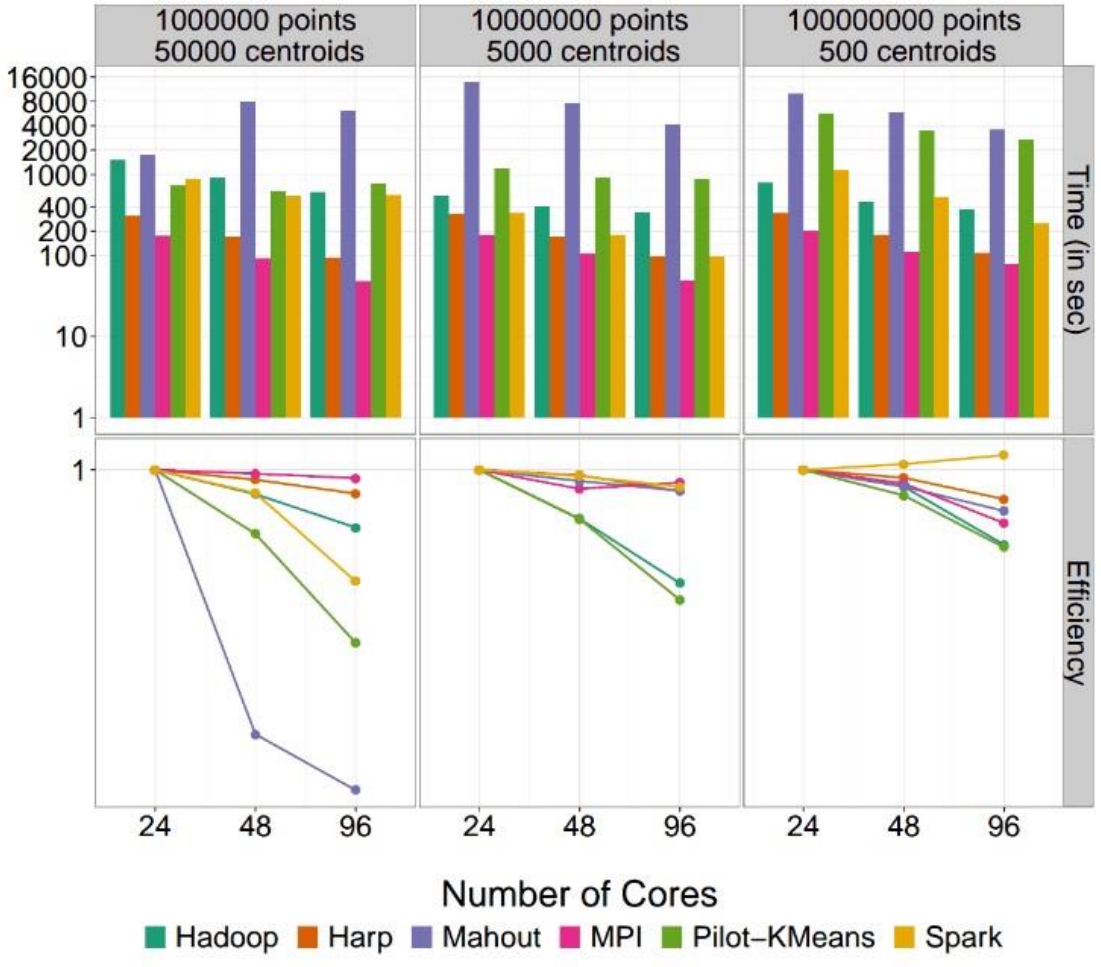
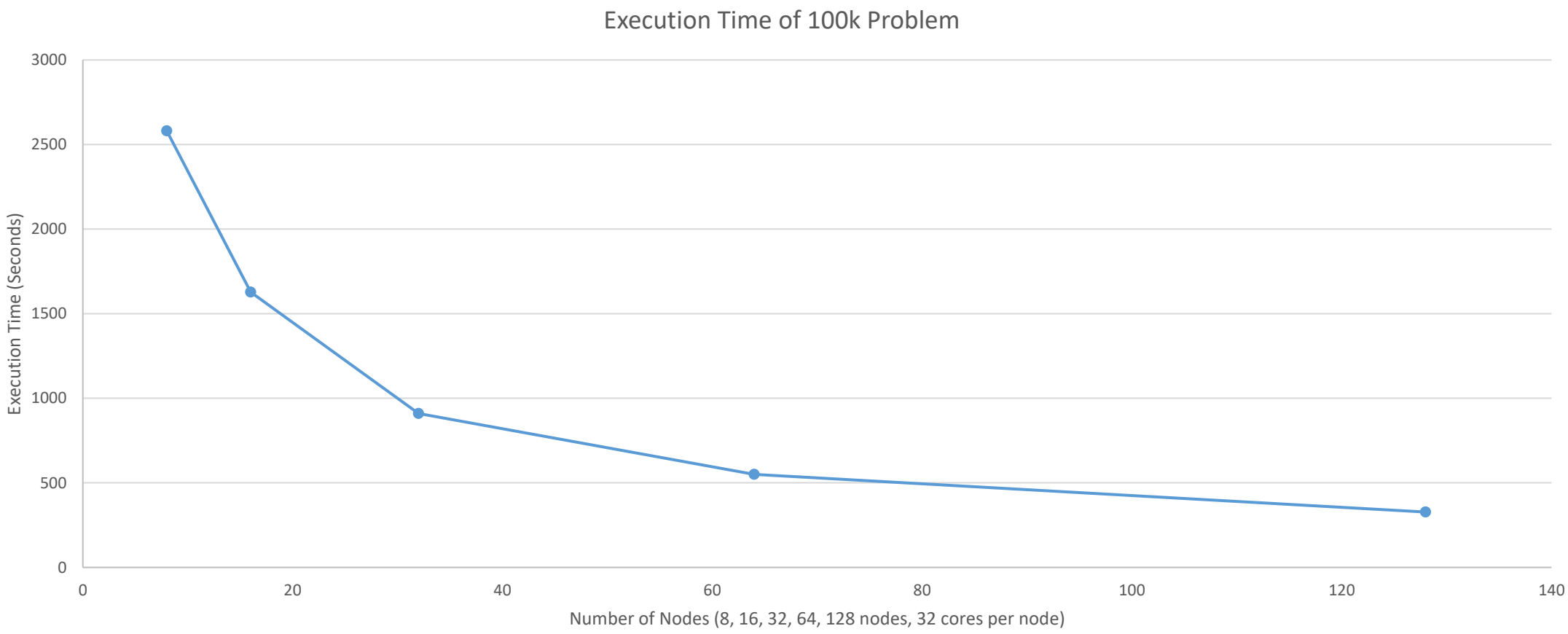


Fig. 5. Time-To-Completion for KMeans on Different Backends

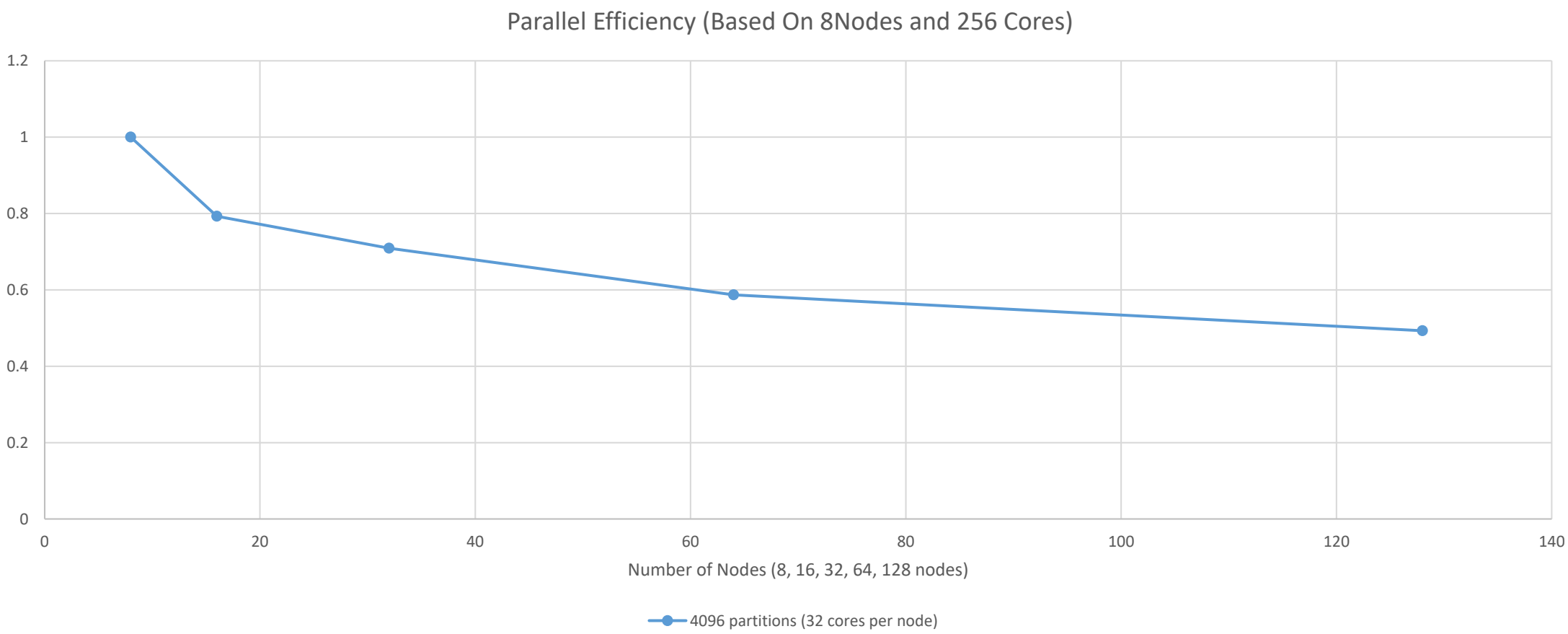
WDA-MDS Performance on Big Red II

- WDA-MDS
  - Yang Ruan, Geoffrey Fox. A Robust and Scalable Solution for Interpolative Multidimensional Scaling with Weighting. IEEE e-Dcience 2013.
- Big Red II
  - http://kb.iu.edu/data/bcqt.html
- Allgather
  - Bucket algorithm
- Allreduce
  - Bidirectional exchange algorithm

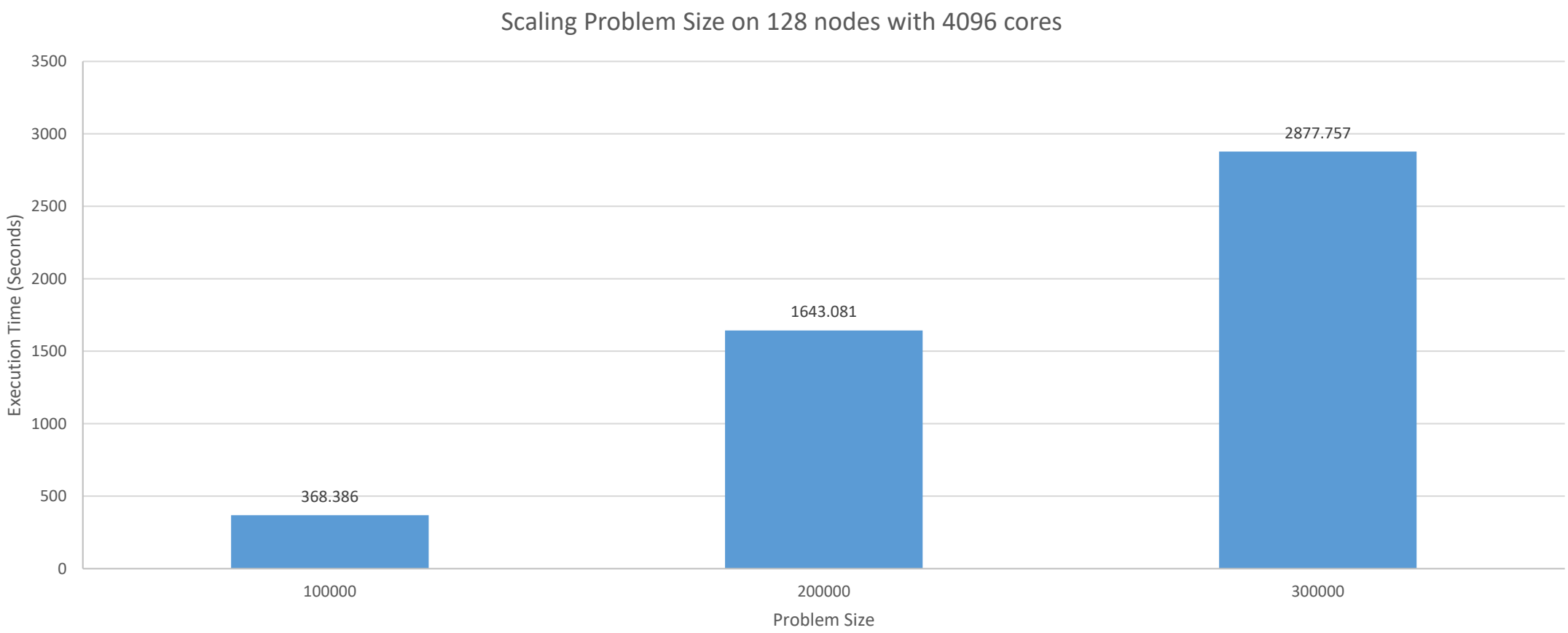
Execution Time of 100k Problem



Parallel Efficiency
Based On 8 Nodes and 256 Cores



Scale Problem Size (100k, 200k, 300k)



Machine Learning on Big Data

- Mahout on Hadoop 

  - https://mahout.apache.org/

- MLlib on Spark
  - http://spark.apache.org/mllib/

- GraphLab Toolkits
  - http://graphlab.org/projects/toolkits.html
  - GraphLab Computer Vision Toolkit

Query on Big Data

- Query with procedural language

- Google Sawzall (2003)
  - Rob Pike et al. Interpreting the Data: Parallel Analysis with Sawzall. Special Issue on Grids and Worldwide Computing Programming Models and Infrastructure 2003.

- Apache Pig (2006) 
  - Christopher Olston et al. Pig Latin: A Not-So-Foreign Language for Data Processing. SIGMOD 2008.
  - https://pig.apache.org/

SQL-like Query

- Apache Hive (2007)

  - Facebook Data Infrastructure Team. Hive - A Warehousing Solution Over a Map-Reduce Framework. VLDB 2009.
  - https://hive.apache.org/
  - On top of Apache Hadoop
- Shark (2012)
  - Reynold Xin et al. Shark: SQL and Rich Analytics at Scale. Technical Report. UCB/EECS 2012.
  - http://shark.cs.berkeley.edu/
  - On top of Apache Spark
- Apache MRQL (2013)
  - http://mrql.incubator.apache.org/
  - On top of Apache Hadoop, Apache Hama, and Apache Spark

Other Tools for Query

- Apache Tez (2013)
  - http://tez.incubator.apache.org/
  - To build complex DAG of tasks for Apache Pig and Apache Hive
  - On top of YARN

- Dremel (2010) Apache Drill (2012)
  - Sergey Melnik et al. Dremel: Interactive Analysis of Web-Scale Datasets. VLDB 2010.
  - http://incubator.apache.org/drill/index.html
  - System for interactive query

Stream Data Processing

- Apache S4 (2011)
  - http://incubator.apache.org/s4/
- Apache Storm (2011)
  - http://storm.incubator.apache.org/
- Spark Streaming (2012)
  - https://spark.incubator.apache.org/streaming/
- Apache Samza (2013)
  - http://samza.incubator.apache.org/

REEF

- Retainable Evaluator Execution Framework
- http://www.reef-project.org/
- Provides system authors with a centralized (pluggable) control flow
  - Embeds a user-defined system controller called the Job Driver
  - Event driven control
- Package a variety of data-processing libraries (e.g., high-bandwidth shuffle, relational operators, low-latency group communication, etc.) in a reusable form.
- To cover different models such as MapReduce, query, graph processing and stream data processing

Thank You!

Questions?