HJ-Hadoop
An Optimized MapReduce
Runtime for Multi-core Systems

Yunming Zhang
Advised by: Prof. Alan Cox and Vivek Sarkar
Rice University

ACM Student Research Competition
SPLASH 13
Hadoop MapReduce Runtime

Figure 1. Map Reduce Programming Model
Hadoop Map Reduce

- Open source implementation of Map Reduce Runtime system
  - Scalable
  - Reliable
  - Available
- Popular platform for big data analytics
Kmeans is an application that takes as input a large number of documents and try to classify them into different topics.
K-means using Hadoop

To be classified documents

Map task in a JVM

Machine 1

Computation

Memory

Topics

Machines …
Kmeans using Hadoop

To be classified documents

Map task in a JVM

Machine 1

Computation

Memory

Topics

Topics

Machines ...
Kmeans using Hadoop

To be classified documents

Map task in a JVM

Machine 1

Machines ...

slice1
slice2
slice3
slice4
slice5
slice6
slice7
slice8

....

Computation

Memory

Topics

Topics

Topics
Kmeans using Hadoop

To be classified documents

Map task in a JVM

Machine 1

Computation

Memory

slice1
slice2
slice3
slice4
slice5
slice6
slice7
slice8
....

Topics
Topics
Topics
Topics

Machines ...
Kmeans using Hadoop

To be classified documents

- slice1
- slice2
- slice3
- slice4
- slice5
- slice6
- slice7
- slice8
- ...

Map task in a JVM

Machine 1

- Computation
- Memory

- Topics 1x
- Topics 1x
- Topics 1x
- Topics 1x

Duplicated In-memory Cluster Centroids

Machines ...

We used 8 mappers from 30 - 80 MB, 4 mappers for 100 – 150 MB, 2 mappers for 180 – 380 for sequential Hadoop.
Memory Wall

• Hadoop’s approach to the problem
  – Increase the memory available to each Map Task JVM by reducing the number of map tasks assigned to each machine.
Kmeans using Hadoop

To be classified documents

Map task in a JVM

Machine 1

Computation

Memory

Topics 2x

Machines ...

slice1
slice2
slice3
slice4
slice5
slice6
slice7
slice8
....
Memory Wall

KMeans Throughput Benchmark

<table>
<thead>
<tr>
<th>Topics data size (MB) with 4KB/topic</th>
<th>Number of topics/Time in min</th>
</tr>
</thead>
<tbody>
<tr>
<td>30 - 80 MB</td>
<td>180</td>
</tr>
<tr>
<td>100 - 150 MB</td>
<td>160</td>
</tr>
<tr>
<td>180 - 380 MB</td>
<td>140</td>
</tr>
</tbody>
</table>

We used 8 mappers from 30 - 80 MB, 4 mappers for 100 - 150 MB, 2 mappers for 180 - 380 for sequential Hadoop.
HJ-Hadoop

To be classified documents

Computation

Map task in a JVM

Dynamic chunking

Memory

Topics 4x

Machines ...

slice1
slice2
slice3
slice4
slice5
slice6
slice7
slice8
....
HJ-Hadoop

To be classified documents

Map task in a JVM

Computation

Dynamic chunking

Machine1

Memory

Topics 4x

No Duplicated In-memory Cluster Centroids

Machines ...

slice1
slice2
slice3
slice4
slice5
slice6
slice7
slice8
....
Habanero Java (HJ)

• Programming Language and Runtime Developed at Rice University
• Optimized for multi-core systems
  – Lightweight async task
  – Work sharing runtime
  – Dynamic task parallelism
  – http://habanero.rice.edu
Results

KMeans Throughput Benchmark

We used 2 mappers for HJ-Hadoop
Results

KMeans Throughput Benchmark

We used 2 mappers for HJ-Hadoop
Results

KMeans Throughput Benchmark

We used 2 mappers for HJ-Hadoop
K Nearest Neighbor Join

![Graph showing comparison between Hadoop and HJ-Hadoop in processing documents per minute across different input document sizes.](graph.png)

- *Hadoop*
- *HJ-Hadoop*
Conclusions

• Our goal is to tackle the memory inefficiency in the execution of MapReduce applications on multi-core systems by integrating a shared memory parallel model into Hadoop MapReduce runtime
  – HJ-Hadoop can be used to solve larger problems efficiently than Hadoop. HJ-Hadoop can process 5x more data at full throughput of the system
  – The HJ-Hadoop can deliver a 4x throughput relative to Hadoop mapper processing large in-memory data sets