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

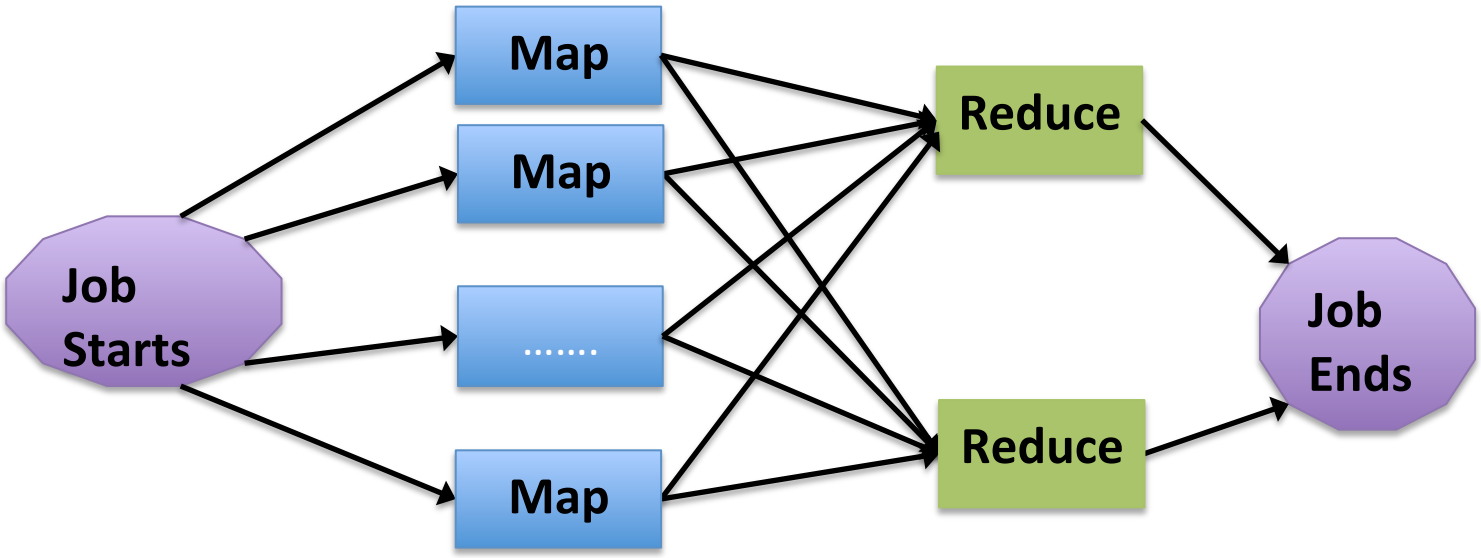
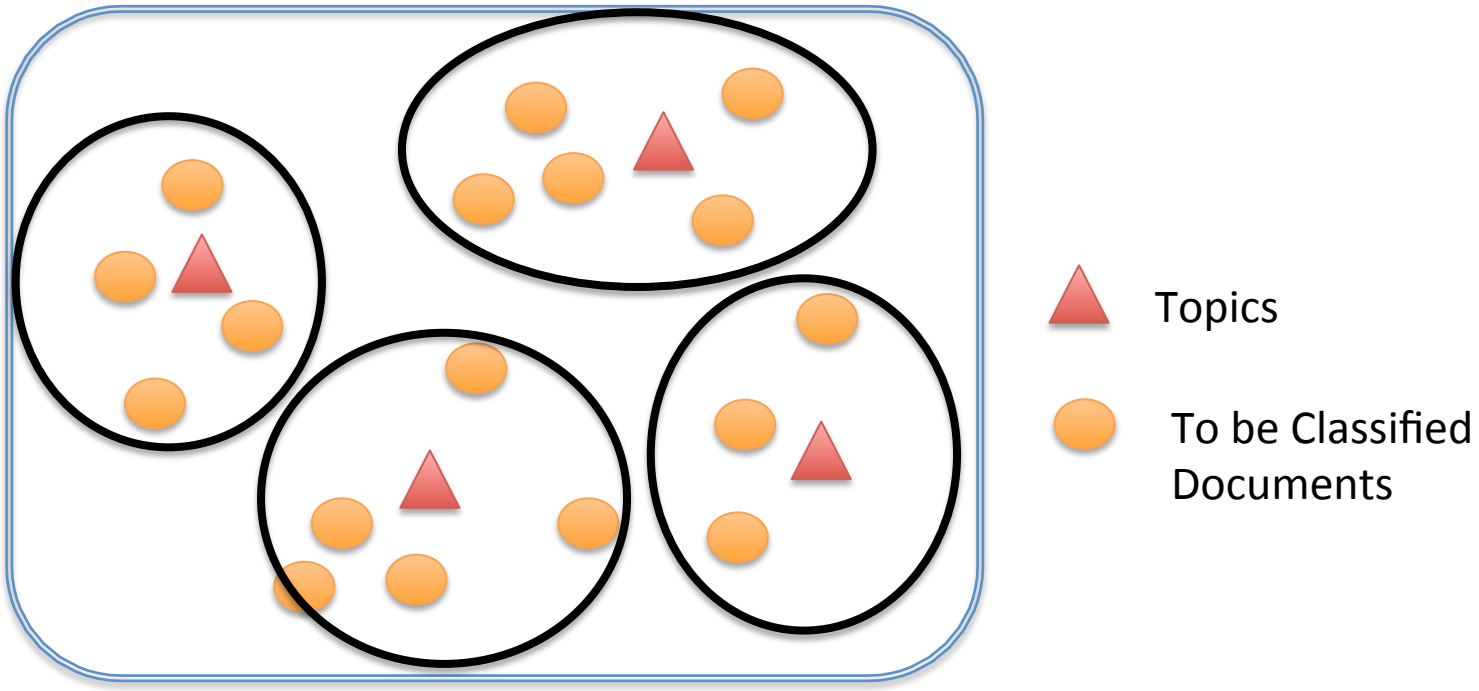


Figure 1. Map Reduce Programming Model

Hadoop Map Reduce

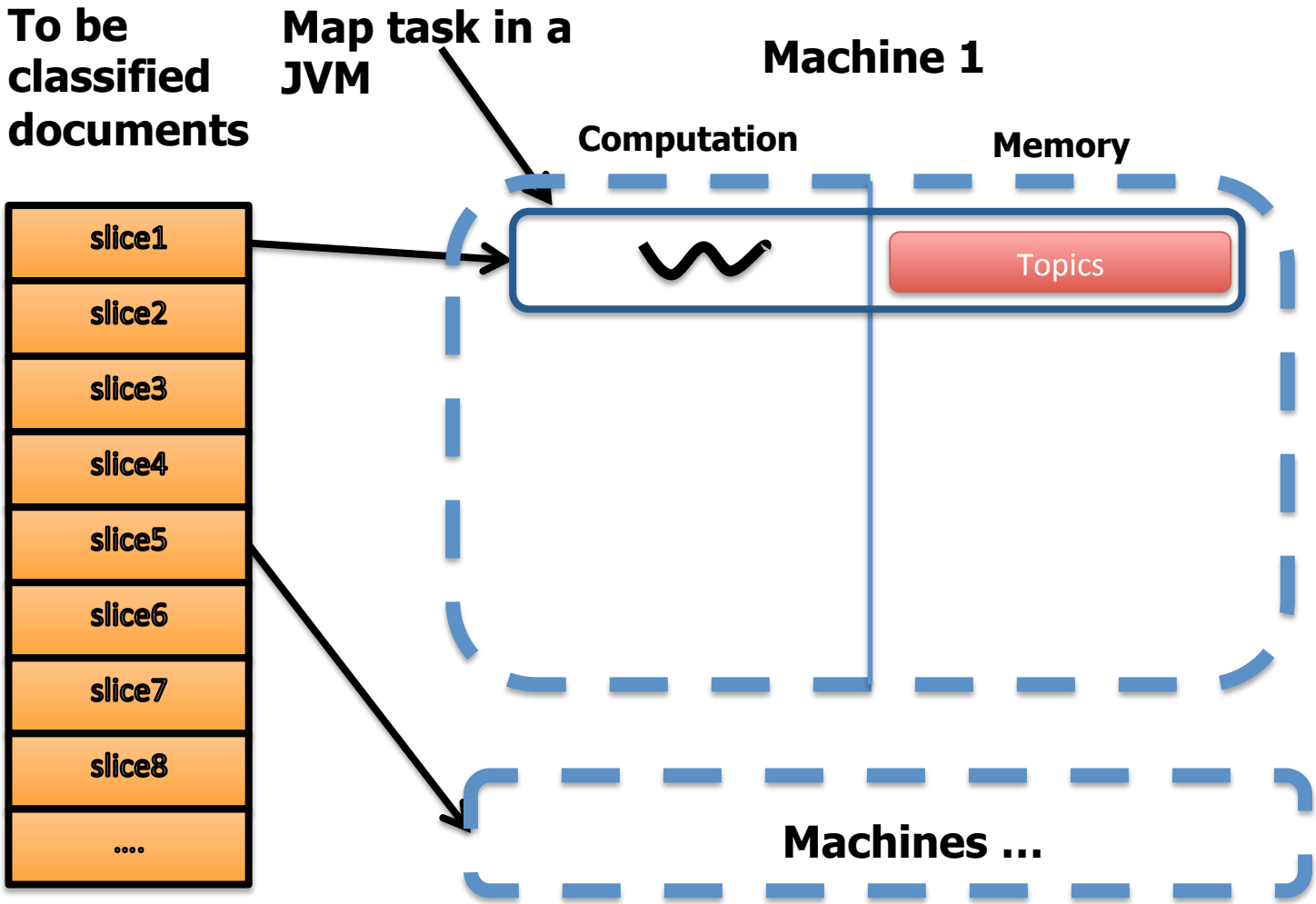
- Open source implementation of Map Reduce Runtime system
  - Scalable
  - Reliable
  - Available
- Popular platform for big data analytics

Kmeans

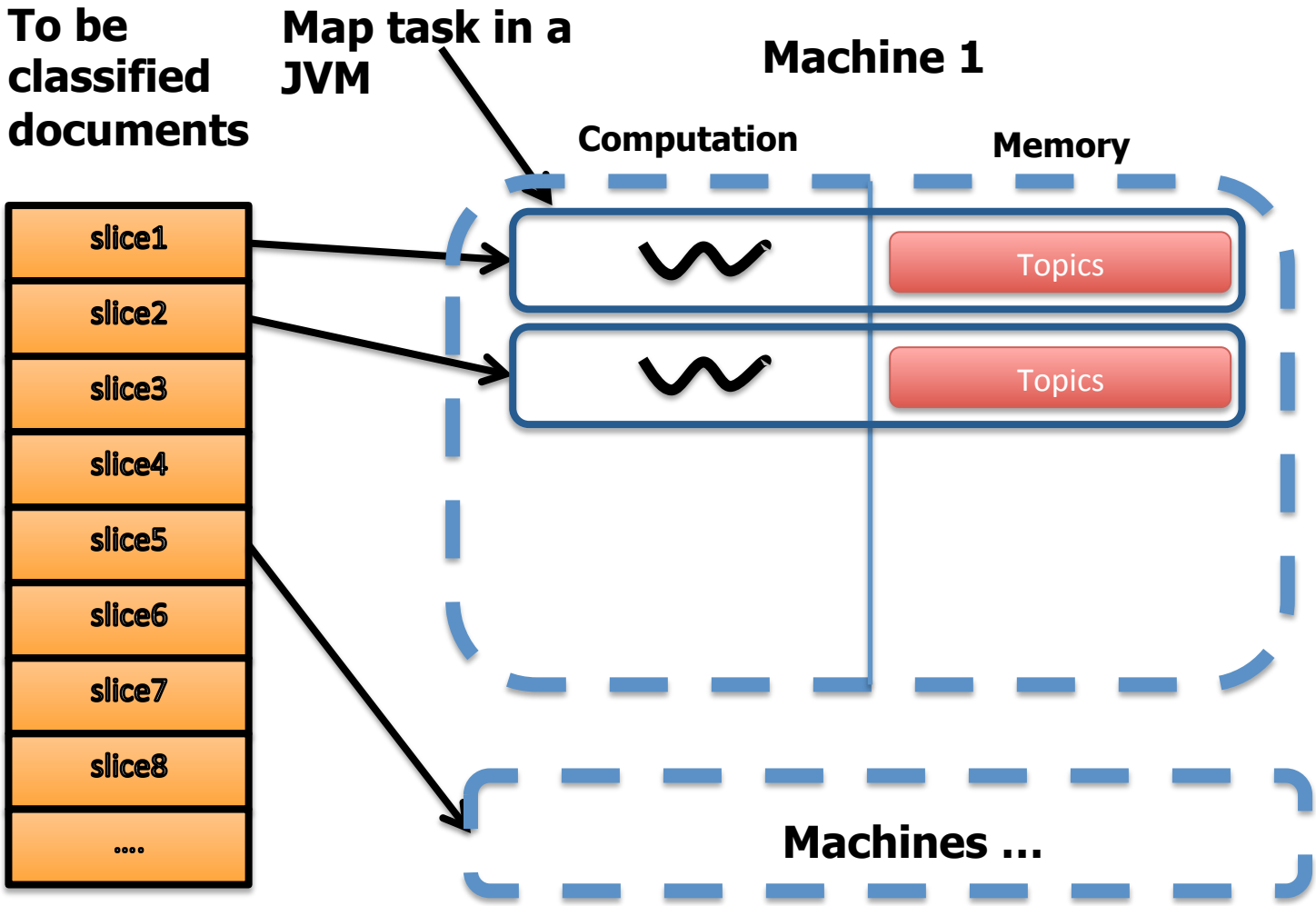


Kmeans is an application that takes as input a large number of documents and try to classify them into different topics

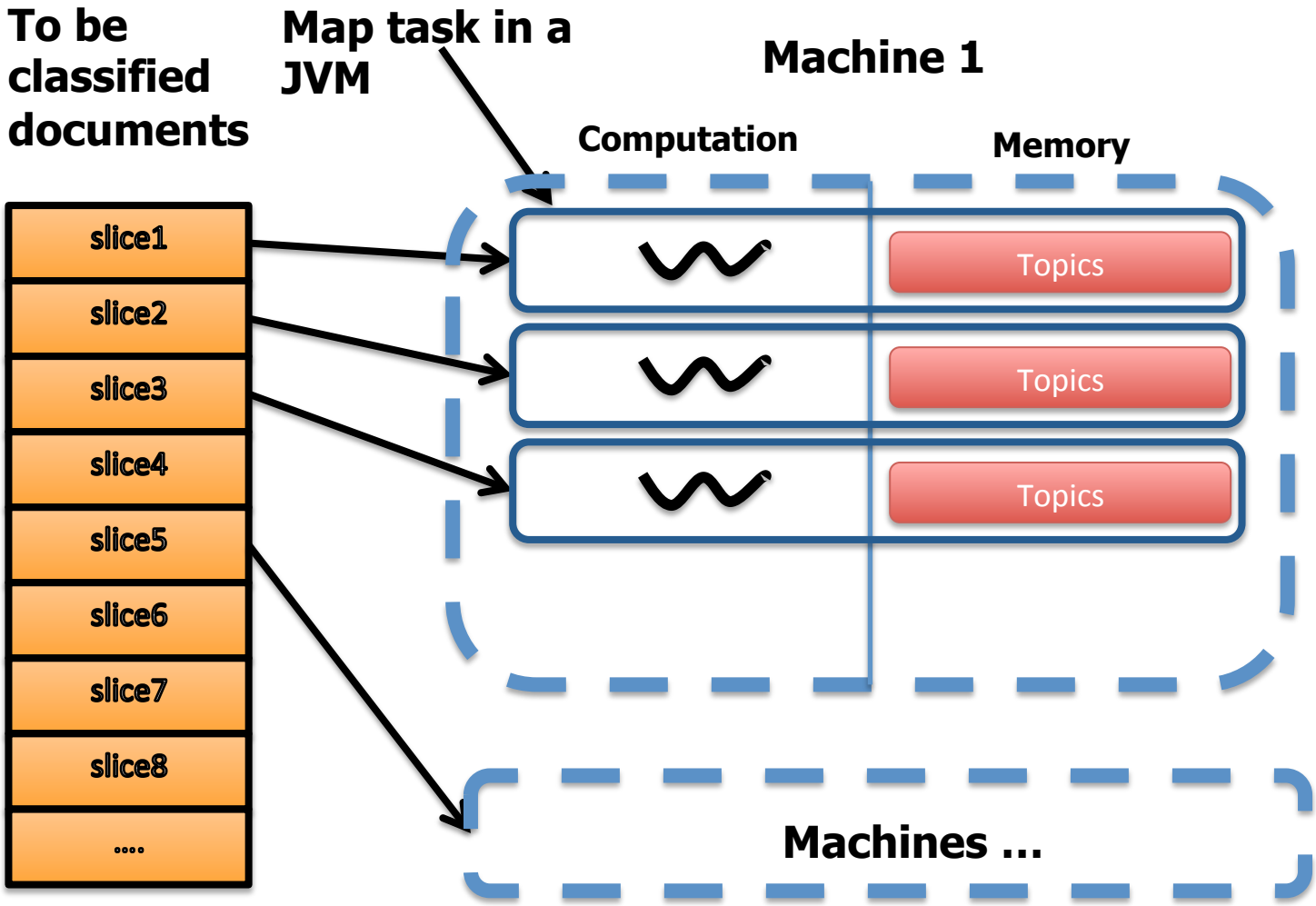
Kmeans using Hadoop



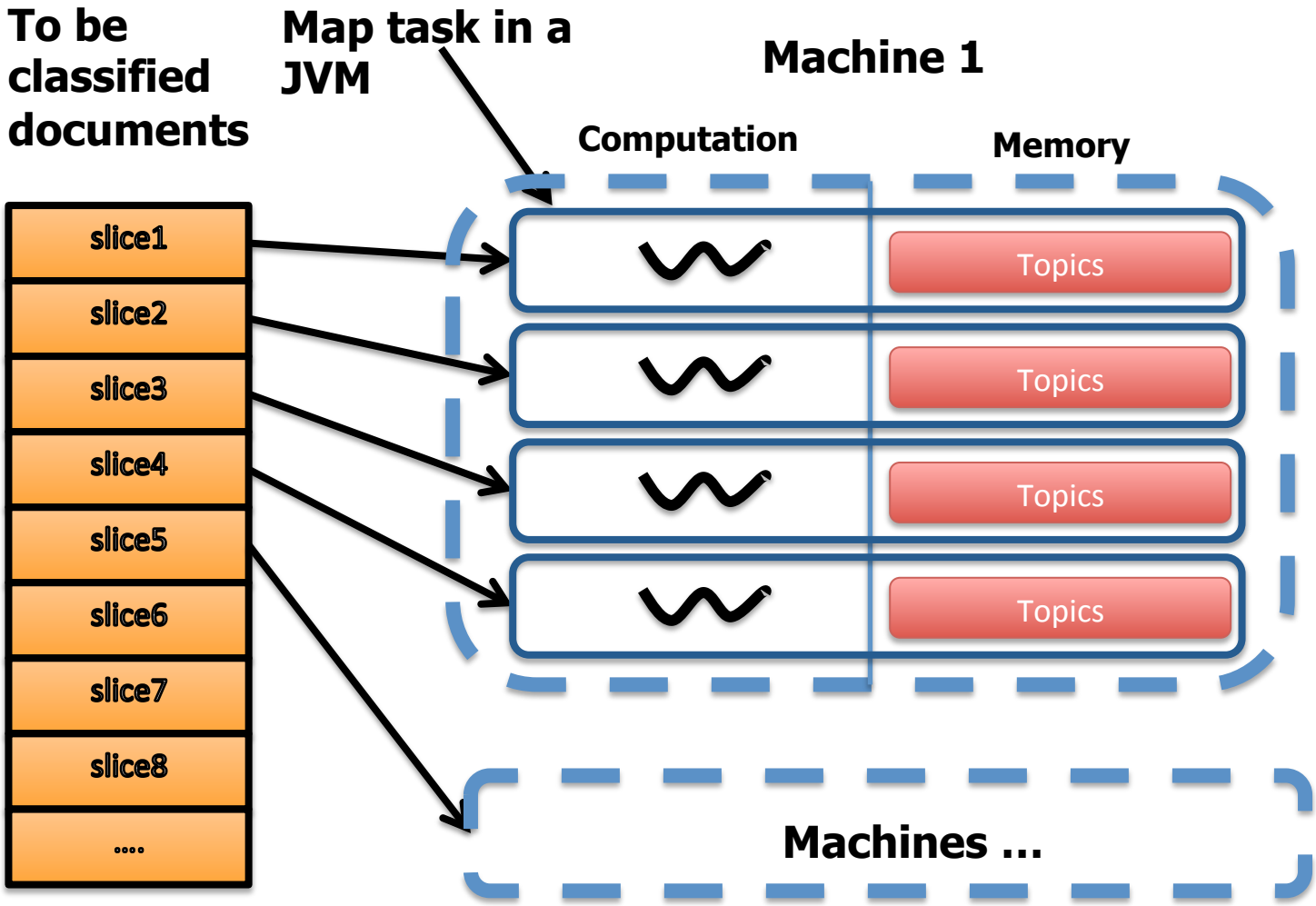
Kmeans using Hadoop



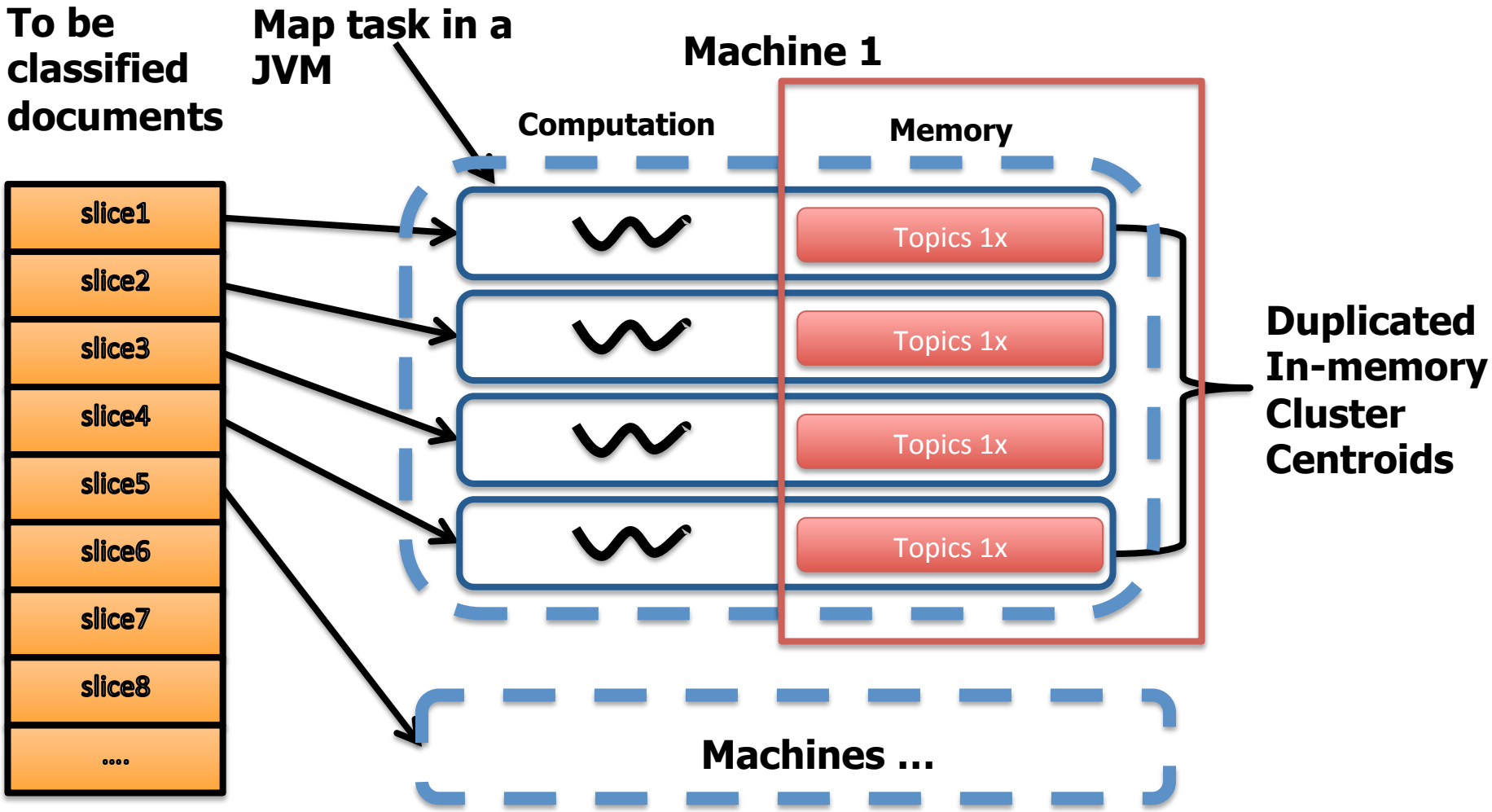
Kmeans using Hadoop



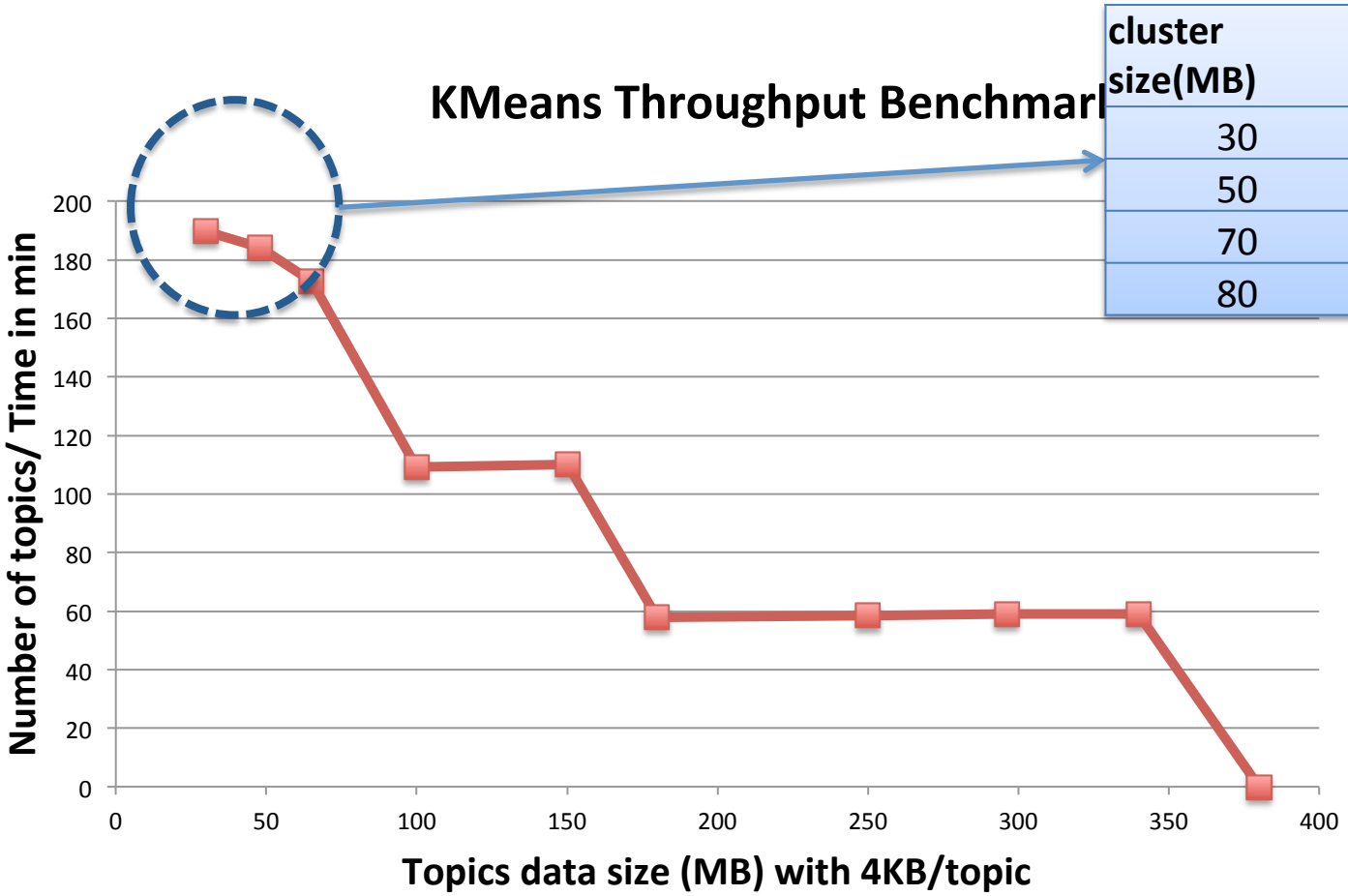
Kmeans using Hadoop



Kmeans using Hadoop



Memory Wall



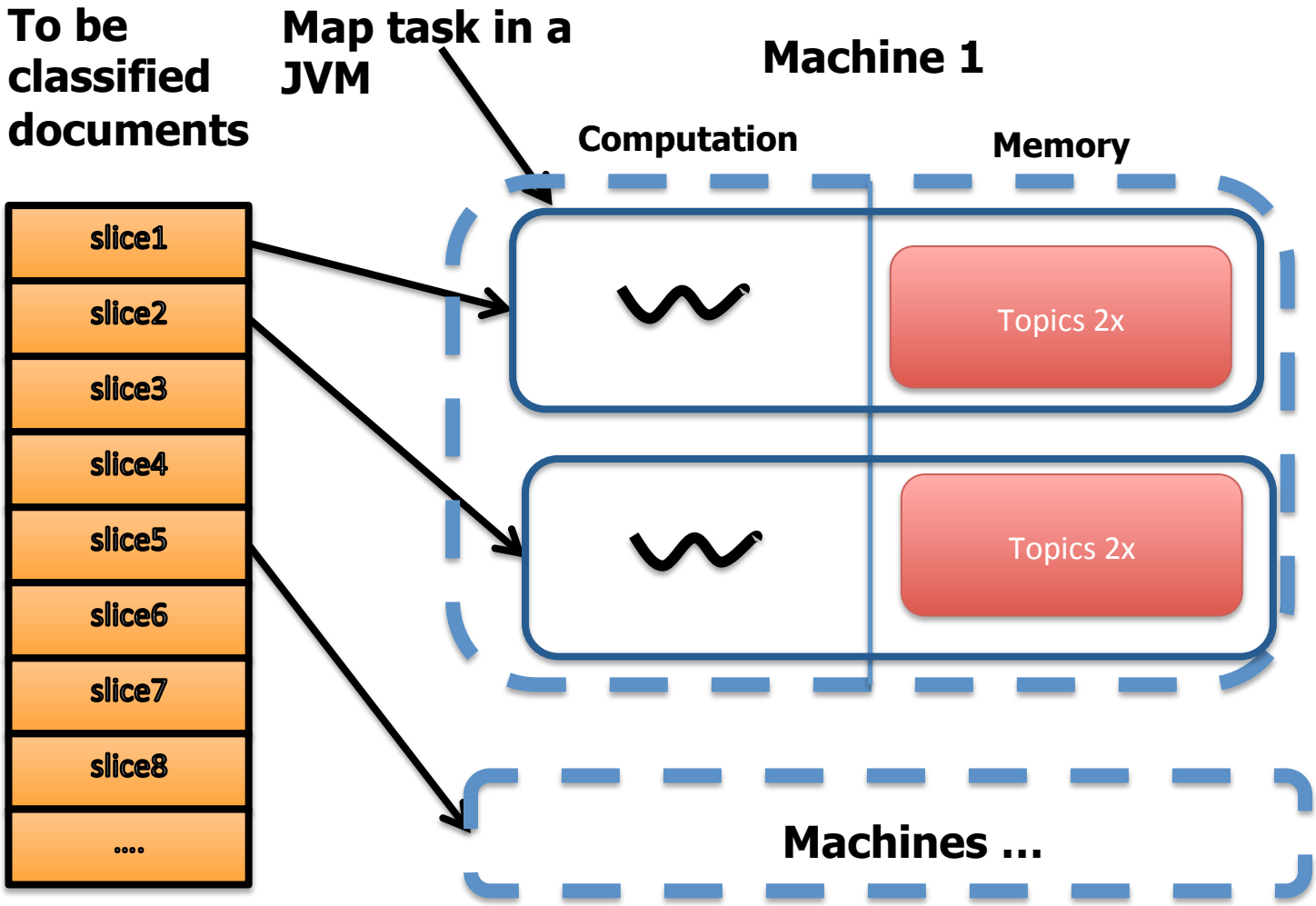
cluster size(MB)	Hadoop Full GC calls
30	3,542
50	4,390
70	5,186
80	1,108,888

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

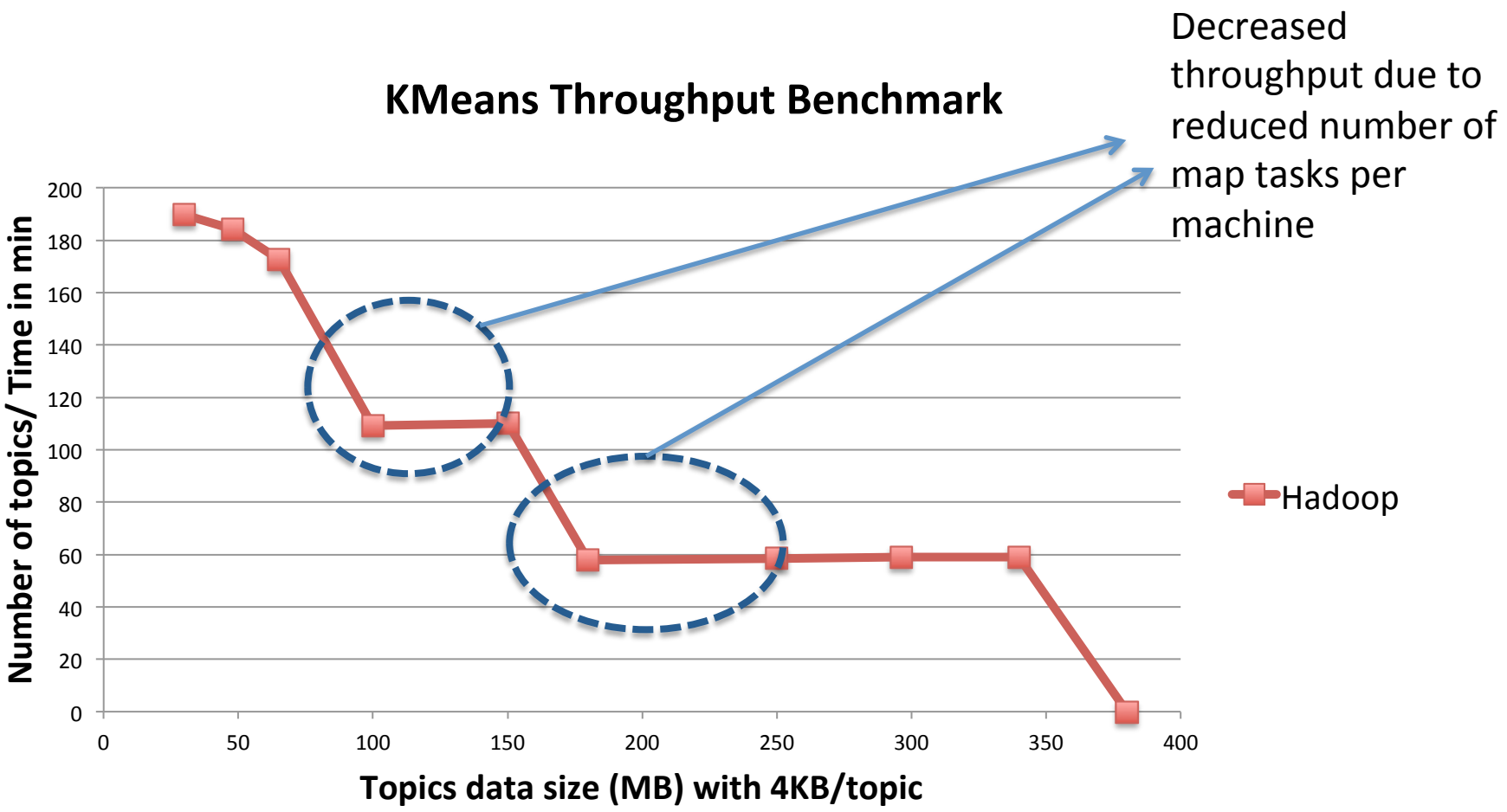
Memory Wall

- 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

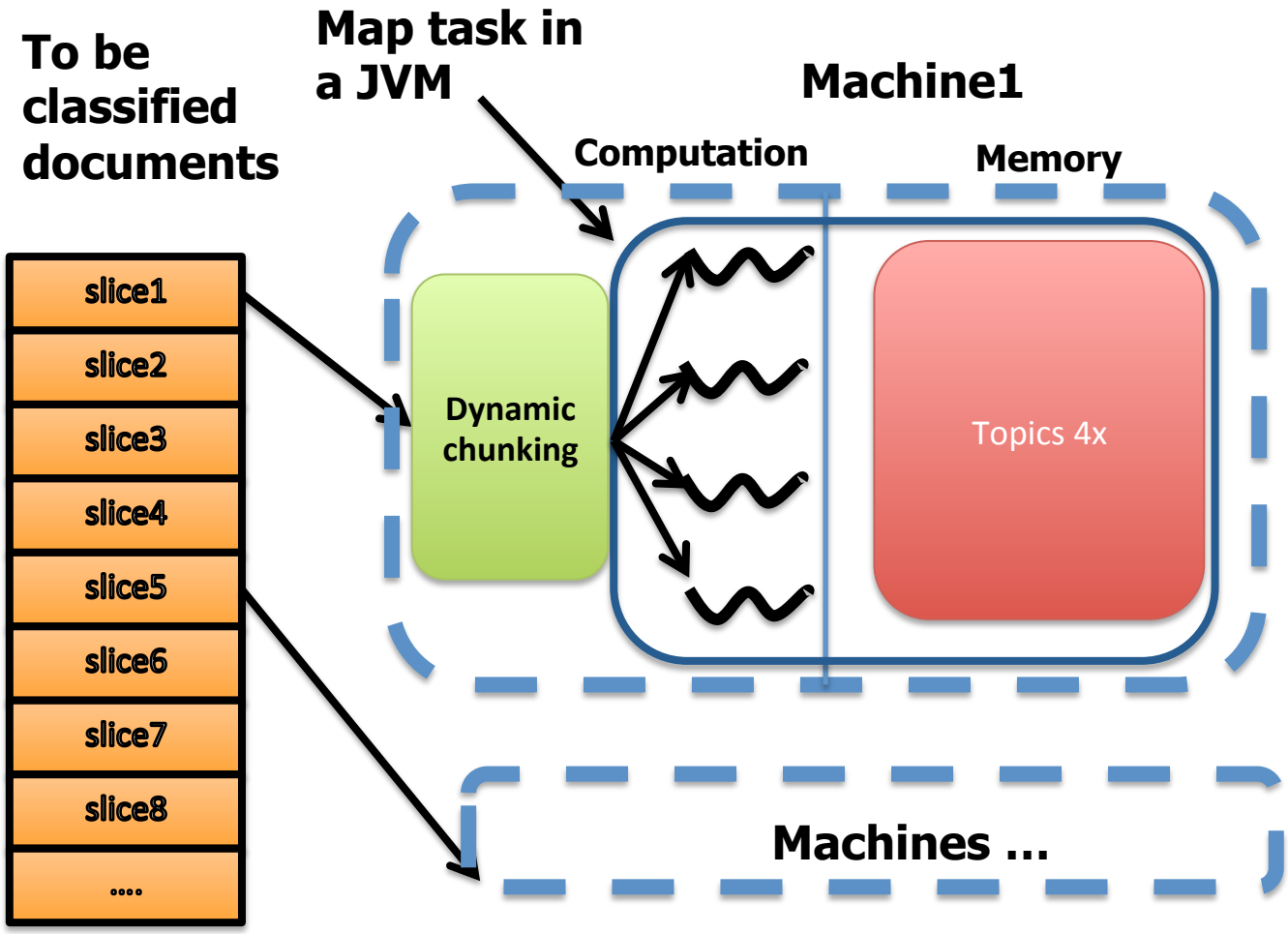


Memory Wall

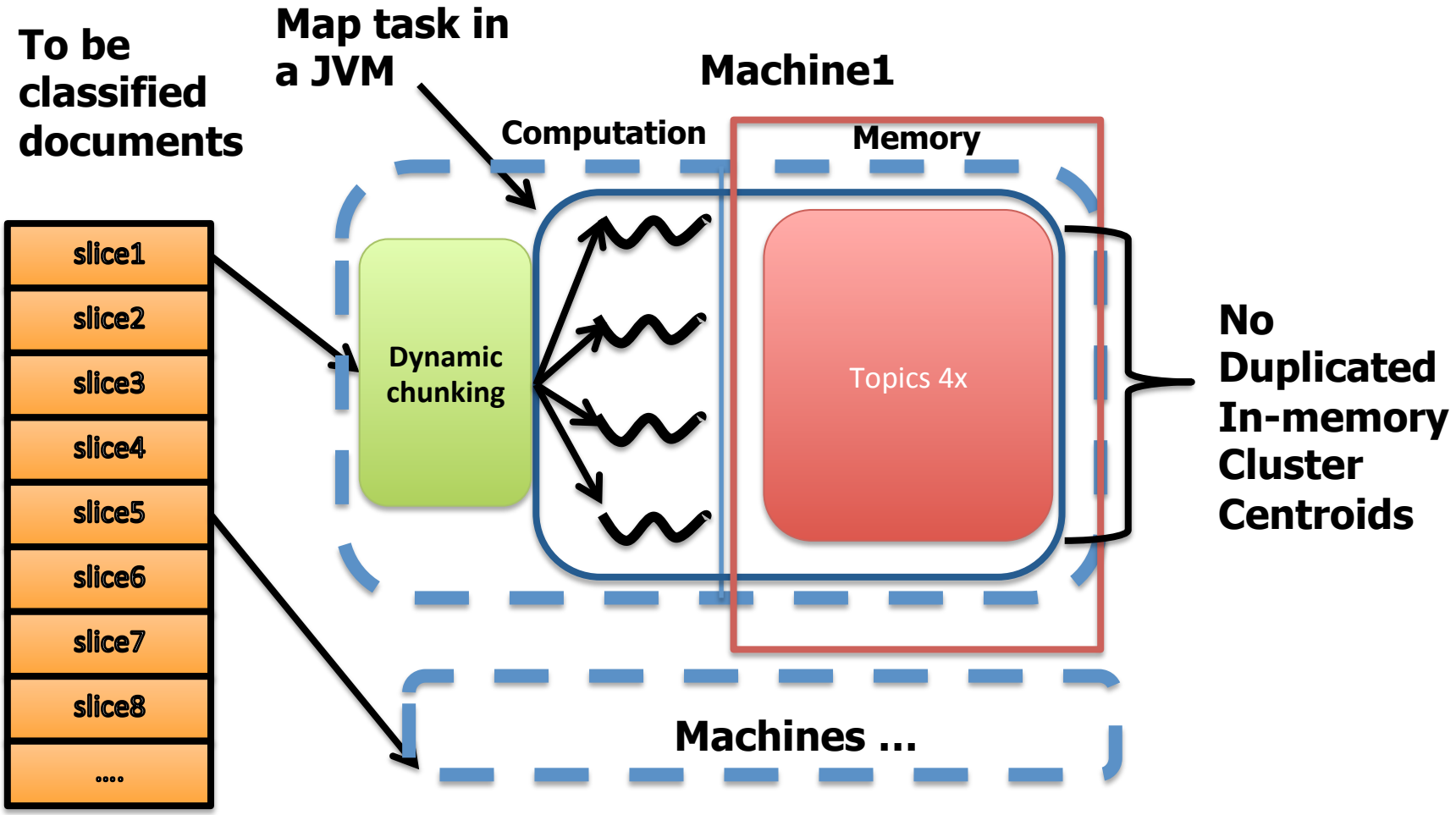


We used 8 mappers from 30 -80 MB, 4 mappers for 100 – 150 MB, 2 mappers for 180 – 380 for sequential Hadoop.

HJ-Hadoop


To be classified documents
Map task in a JVM
Machine1
Computation
Memory
slice1
slice2
slice3
slice4
slice5
slice6
slice7
slice8
....
Dynamic chunking
Topics 4x
Machines ...

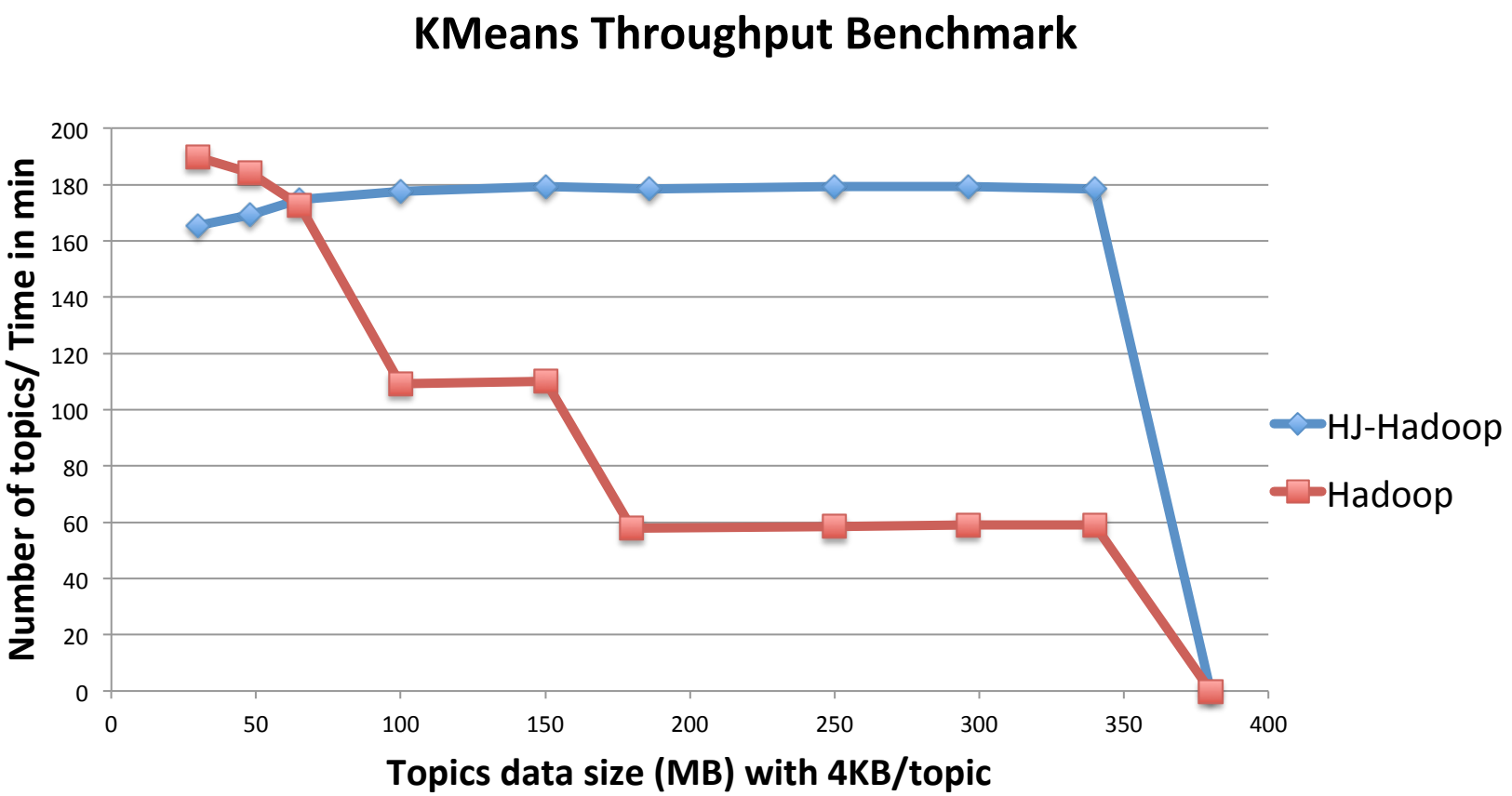
HJ-Hadoop


To be classified documents
Map task in a JVM
Machine1
Computation
Memory
slice1
slice2
slice3
slice4
slice5
slice6
slice7
slice8
....
Dynamic chunking
Topics 4x
No Duplicated In-memory Cluster Centroids
Machines ...

Habanero Java(HJ)

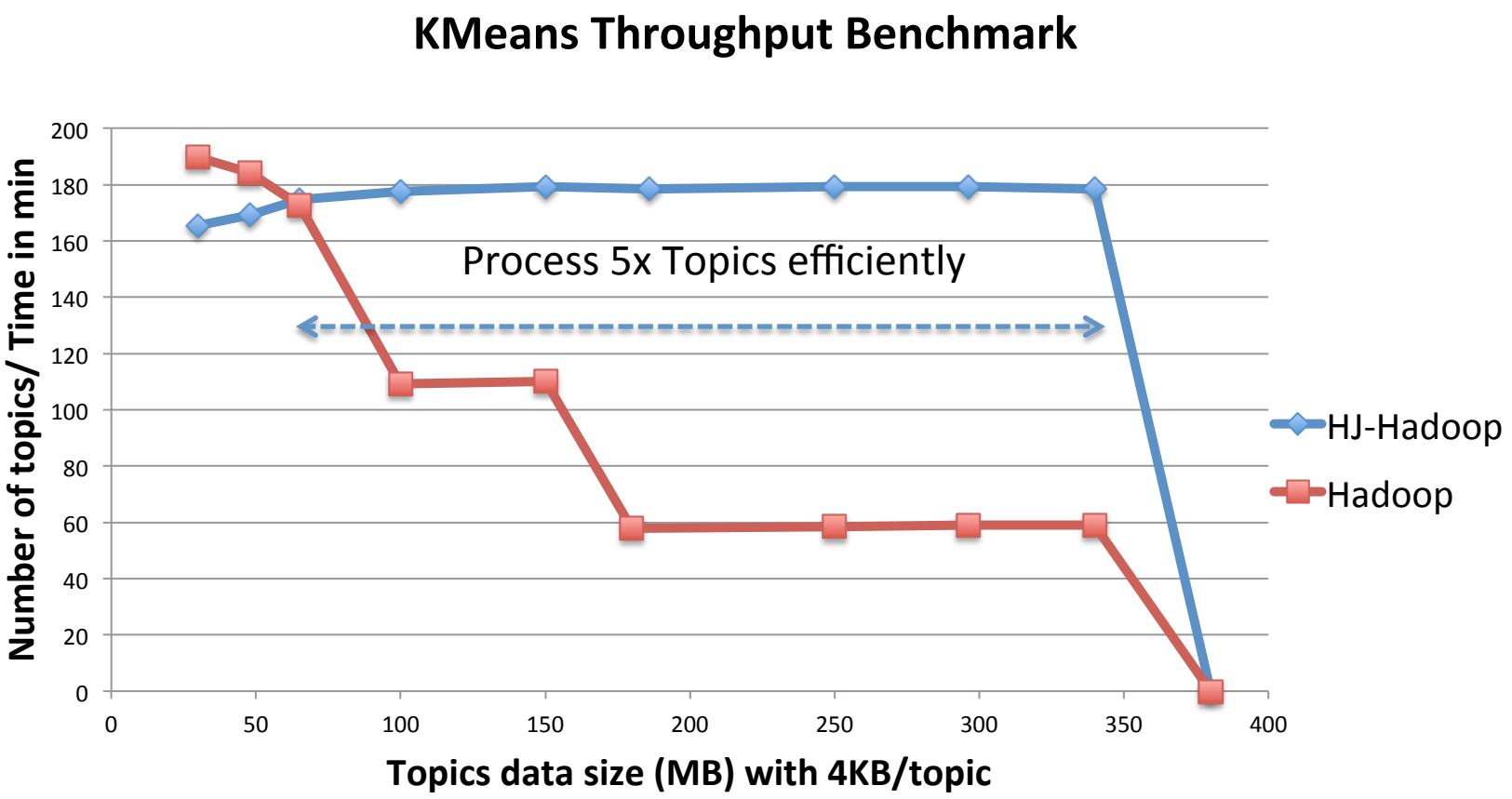
- Programming Language and Runtime Developed at Rice University
- Optimized for multi-core systems
  - Lightweight async task
  - Work sharing runtime
  - Dynamic task parallelism
  - http://habanero.rice.edu

Results



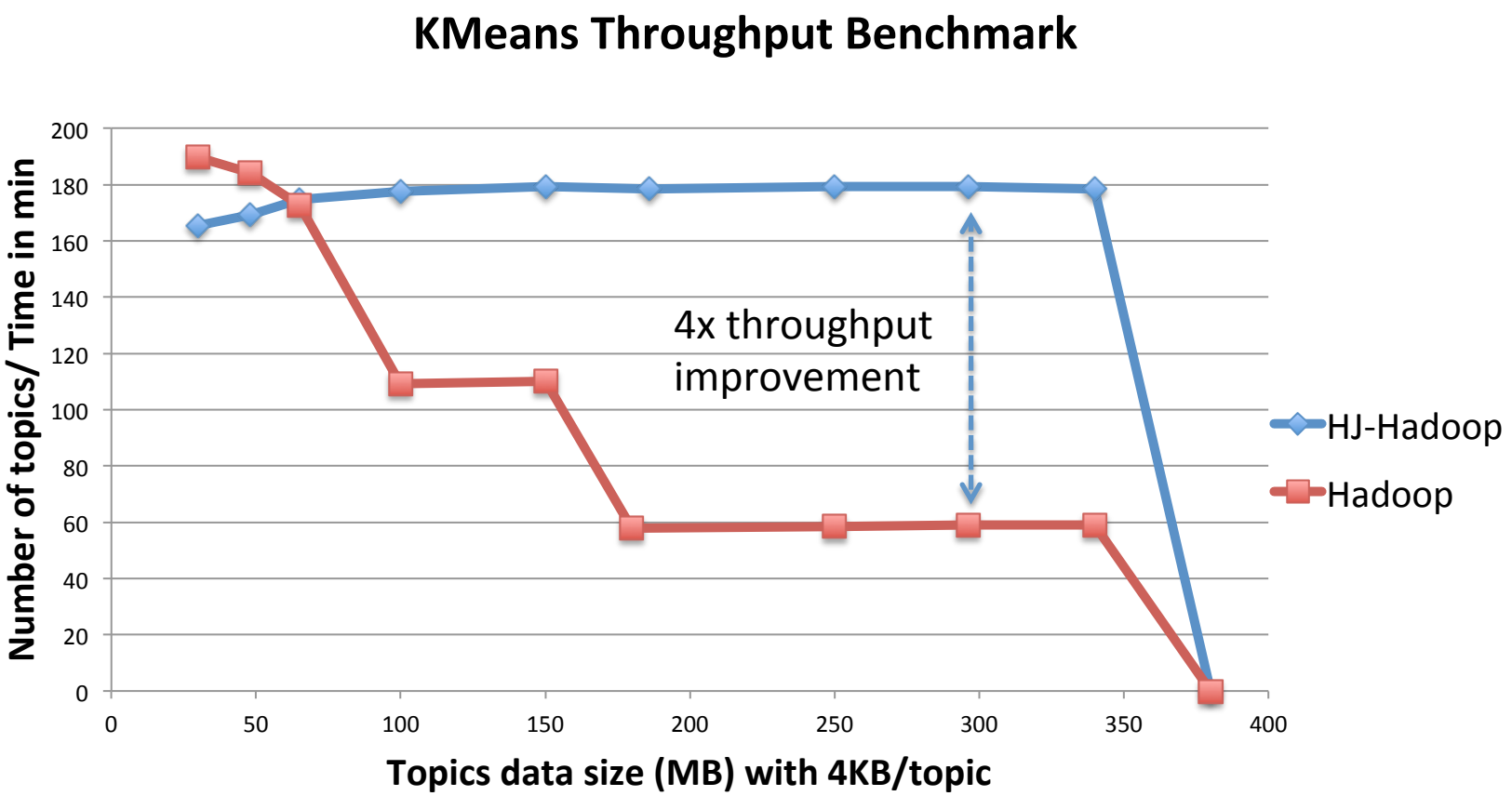
We used 2 mappers for HJ-Hadoop

Results



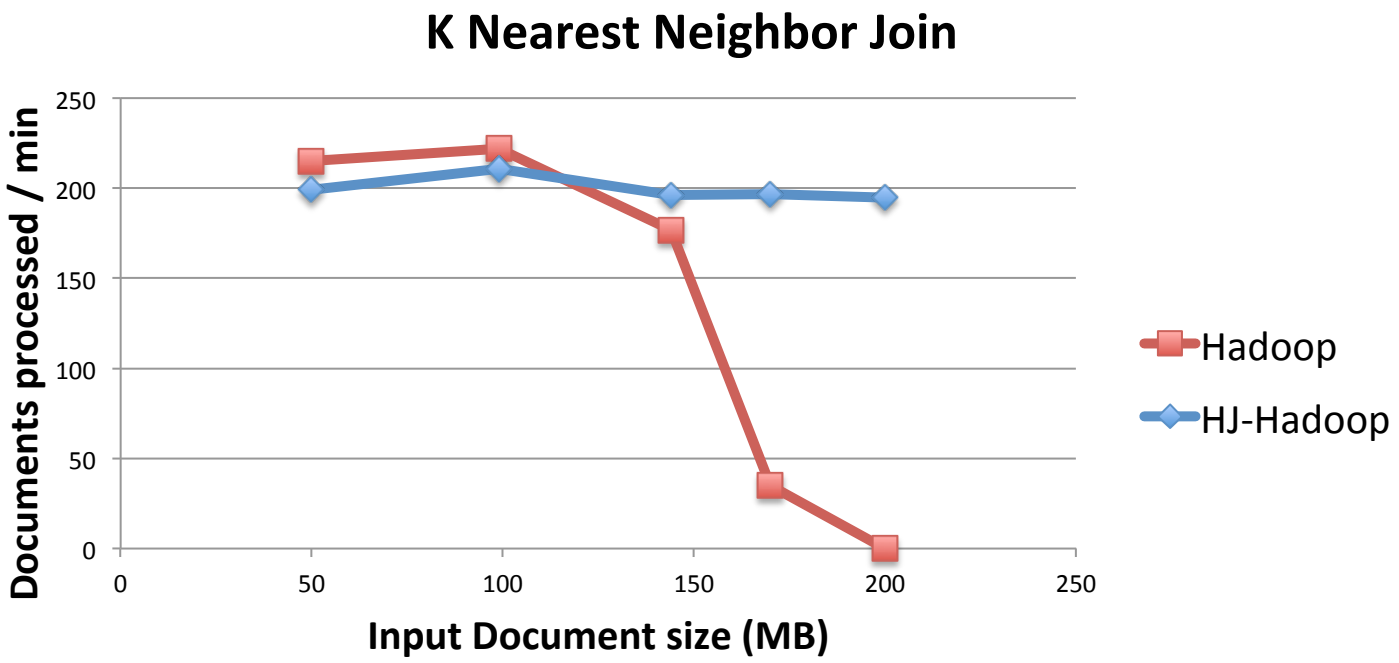
We used 2 mappers for HJ-Hadoop

Results



We used 2 mappers for HJ-Hadoop

K Nearest Neighbor Join



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