CSE 4/587
Data Intensive Computing

Dr. Eric Mikida
epmikida@buffalo.edu
208 Capen Hall

Day 10
MapReduce Review and Case Study

Announcements and Feedback

- If you have not registered your team yet, please do so ASAP
  - For those that have registered, you should have gotten an introduction email from your assigned TA
- Example problem statements will be posted to Piazza
- Many questions on data set
- Phase 2 and 3 are coming this week!

Do I Need More Data?

- 2000 lines is just a minimum
- Part of the Phase 1 process is where you have to determine if your data is going to work towards your goals
    - Think back to the demo in class with Real Estate data
    - If my goal was to look specifically at single family homes across different neighborhoods, I would probably need more data
- Don't be afraid to get data from multiple sources!

What to Turn In and How

- One submission **per team** (will fix in the handout)
- Your submission will consist of your code as well as a write up
  - The write up will describe **why** you did the operations you did and explain **what you learned**
- Submission will probably be through UBLearns, but that is still TBD

What to Expect from Phase 2 and 3

- Phase 2 will focus on applying modeling and algorithms to your data to address your problem statement, and visualizing the results
  - You will be required to try a certain number of statistical models/ML algorithms. Some from class, some from external sources.
  - You must explain the motivation for using a particular technique and what you learned from applying it.
- Phase 3 will focus on building a data product based on your findings
  - You will also record a short demo of your product

Recap from Last Class

- Live demo of MapReduce execution on local instance of Hadoop
  - Viewed/manipulated our local Hadoop instance via browser and command-line
  - Defined a Mapper, Reducer, and main for a simple word count problem
  - Executed our job and updated our code based on observations
- Very simple code — scales up without change
- The execution/parallelization details are handled by Hadoop

Where is MR Actually Used?

- Googe uses it (we think) for wordcount, adwords, pagerank, indexing
- Simple algorithms such as grep, text-indexing, reverse indexing
- Bayesian classification: data mining
- Facebook uses it for various things, ie demographic information
- Financial services use it for analytics
- Astronomy: Gaussian analysis for location extra-terrestrial objects
- Expected to play a critical role in semantic web and web3.0

Where is MR Actually Used?

- Googe uses it (we think) for wordcount, adwords, pagerank, indexing
- Simple algorithms such as grep, text-indexing, reverse indexing
- Bayesian classification: data mining
- Facebook uses it for various things, ie demographic information
- Financial services use it for analytics
- Astronomy: Gaussian analysis for location extra-terrestrial objects
- Expected to play a critical role in semantic web and web3.0
- **Bioinformatics and Next-Generation Sequencing (NGS)**

Case Study: Optimizing MapReduce

- Many applications in BioInformatics revolve around next-generation sequencing
  - "NGS has become the leading application area in the domain of bioinformatics" [1]
- Process of analyzing sequences such as DNA to understand different characteristics of an organism
- Very large DNA strings (sequences of A,C,G,T bases)
  - Hundreds of GB of sequence data can be generated from single experiments
- A number of different problems arise: k-mer counting, sequence quality assessment, read alignment, fast similarity search, etc
- Two sources of knowledge required: domain specific, and big data

[1] Lizhen Shi, Zhong Wang, Weikuan Yu, Xiandong Meng, **A case study of tuning MapReduce for efficient Bioinformatics in the cloud**, Parallel Computing, Volume 61, 2017, Pages 83-95, ISSN 0167-8191, https://doi.org/10.1016/j.parco.2016.10.002.

Case Study: Optimizing MapReduce

- Many applications in BioInformatics revolve around next-generation sequencing
  - "NGS has become the leading application area in the domain of bioinformatics" [1]
- Process of analyzing sequences such as DNA to understand different characteristics of an organism
- Very large DNA strings (sequences of A,C,G,T bases)
  - Hundreds of GB of sequence data can be generated from single experiments
- A number of different problems arise: **k-mer counting**, sequence quality assessment, read alignment, fast similarity search, etc
- Two sources of knowledge required: domain specific, and big data

[1] Lizhen Shi, Zhong Wang, Weikuan Yu, Xiandong Meng, **A case study of tuning MapReduce for efficient Bioinformatics in the cloud**, Parallel Computing, Volume 61, 2017, Pages 83-95, ISSN 0167-8191, https://doi.org/10.1016/j.parco.2016.10.002.

MapReduce in Bioinformatics

Table 1
Available MapReduce-based Bioinformatics tools.

Name	Description
CloudBLAST [9]	Combining MapReduce and virtualization on distributed resources for Bioinformatics applications
CloudBurst [10]	Highly sensitive read mapping with MapReduce
Biodoop [11]	Bioinformatics on Hadoop
Crossbow [12]	Searching for SNPs with cloud computing
GATK [13]	A MapReduce framework for analyzing next-generation DNA sequencing data
Myrna [14]	Cloud-scale RNA-sequencing differential expression analysis
Galaxy [15]	A comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences
SEAL [16]	A distributed short read mapping and duplicate removal tool
CloudAligner [17]	A fast and full-featured MapReduce based tool for sequence mapping
Contrail [18]	A de Bruijn Genome assembler that uses Hadoop
FX [19]	an RNA-Seq analysis tool on the cloud
BioPig [20]	A Hadoop-based analytic toolkit for large-scale sequence data
SeqPig [21]	Simple and scalable scripting for large sequencing data sets in Hadoop
Halvade [22]	Scalable sequence analysis with MapReduce

Many established MapReduce libraries exist for Bioinformatics and NGS

k-mer Counting

- k-mer counting is a critical first step for many NGS applications
- A k-mer refers to all the possible subsequences of length k in a DNA/RNA sequence
- k-mer counting returns a count of evey k-mer present in a sequence

"When the k-mer size is large and billions of reads need to be processed, k-mer counting becomes the most difficult problem in Bioinformatics" [1]

k-mer Counting Mapper

k = 3

ACGGCTAGAACGCTAACCGATCAGTCAGCTAGAC…

<"ACG",1>

k-mer Counting Mapper

k = 3

ACGGCTAGAACGCTAACCGATCAGTCAGCTAGAC…

<"ACG",1>

<"CGC",1>

k-mer Counting Mapper

k = 3

ACGGCTAGAACGCTAACCGATCAGTCAGCTAGAC…

<"ACG",1>

<"GGC",1>

<"CGC",1>

k-mer Counting Mapper

k = 3

ACGGCTAGAACGCTAACCGATCAGTCAGCTAGAC…

<"GCT",1>

<"ACG",1> <"GGC",1>

<"CGC",1>

k-mer Counting Mapper

k = 3

<"CTA",1>

<"GCT",1> <"TAA",1>

ACGGCTAGAACGCTAACCGATCAGTCAGCTAGAC…

<"AAC",1>

<"AAC",1>

<"GCT",1>

<"GAA",1> <"ACG",1>

<"ACG",1> <"GGC",1> etc…

<"CTA",1> <"AGA",1>

<"CGC",1> <"CGC",1>

<"TAG",1>

k-mer Counting Mapper

```
map(input):
  for i in [0,input.length - k):
    emit(input[i:i+k], 1)
```

Optimizing k-mer Counting

- Primary goal of the paper is to study optimization techniques for k-mer counting (in BPig library)
- Studied many different configuration parameters, segmented into four groups:
  - CPU, Memory, I/O, and Network

CPU/Memory Relevant Parameters

- CPU and Memory are obviously important, however **many data-intensive MapReduce applications are I/O and network bound**
- Primary parameters in CPU/Memory relate to the number of available cores, and the memory allocated to each container
    - These are heavily dependent on the particular cluster being run used
- The one parameter they did look at in a bit more detail is control over Java garbage collection (GC)

Disk Relevant Parameters

- Intermediate data (largely from mappers) **is also stored in HDFS**
- Hadoop applications are often I/O bound; managing intermediate data size and I/O costs can provide large benefits
- Relevant parameters are block size and and data compression

Network Parameters

- Mapping phase does not require network usage (computation is performed local to the data!)
- Reduce phase requires significant network usage
  - Shuffle – can run during map phase, but finishes after map completes
  - Sort – runs after shuffle completes
  - Reduce – runs after sort completes
- MapReduce allows overlap of shuffle and map phases

Network Parameters

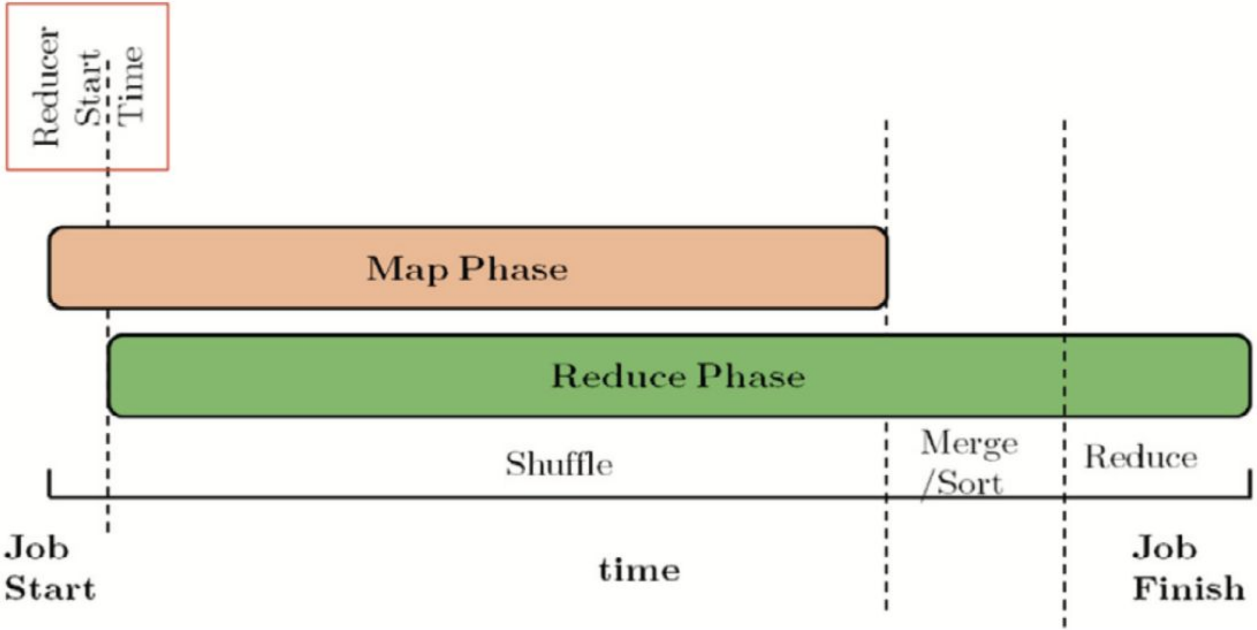
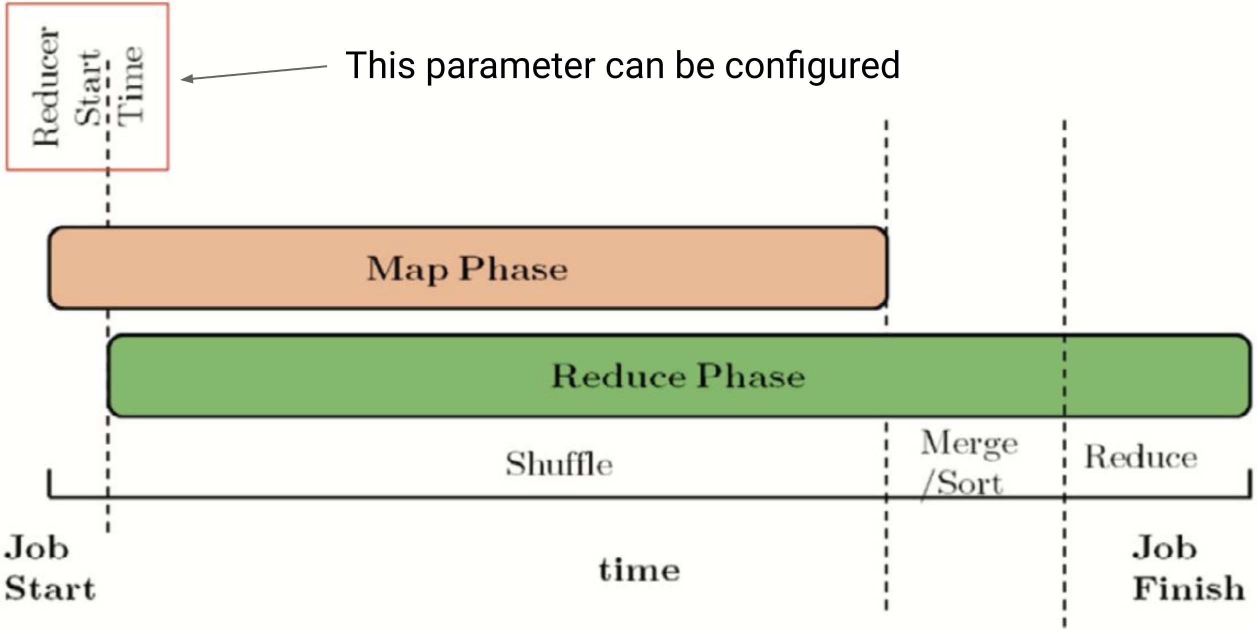


Fig. 2. Decomposition of reducer phases. [1]

Network Parameters



This parameter can be configured

Fig. 2. Decomposition of reducer phases. [1]

Intermediate Data

- Intermediate data is the data that is produced during MapReduce execution (primarily from mappers)
- This data is also stored in HDFS
- Intermediate data is HUGE in k-mer counting
  - Think about how many key-value pairs are produced for a single character in the input string…

Intermediate Data

Table 5
Characteristics of intermediate data for common Hadoop applications.

Job Name	Input data Size (TB)	Int. data Size (TB)	Int./Input
LogProc	1.10	1.10	100%
NdayModel	3.54	3.54	100%
BehaviorModel	3.60	9.47	263%
ClickAttribution	6.80	8.20	121%
SegmentExploder	14.10	25.20	179%
LogRead	1.10	1.10	100%
LogCount	1.10	0.04	4%

Table 6
Characteristics of intermediate data for k-mer (k=20) counting.

Input size (GB)	Int. data size (GB)	Int./Input
1	13.5	1350%
5	67.7	1354%
10	135.4	1354%
20	270.9	1355%
40	541.5	1354%
60	830.0	1383%
100	1381.0	1381%

Intermediate Data

Table 5
Characteristics of intermediate data for common Hadoop applications.

Job Name	Input data Size (TB)	Int. data Size (TB)	Int./Input
LogProc	1.10	1.10	100%
NdayModel	3.54	3.54	100%
BehaviorModel	3.60	9.47	263%
ClickAttribution	6.80	8.20	121%
SegmentExploder	14.10	25.20	179%
LogRead	1.10	1.10	100%
LogCount	1.10	0.04	4%

Table 6
Characteristics of intermediate data for k-mer (k=20) counting.

Input size (GB)	Int. data size (GB)	Int./Input
1	13.5	1350%
5	67.7	1354%
10	135.4	1354%
20	270.9	1355%
40	541.5	1354%
60	830.0	1383%
100	1381.0	1381%

Intermediate Data

Table 5
Characteristics of intermediate data for common Hadoop applications.

Job Name	Input data Size (TB)	Int. data Size (TB)	Int./Input
LogProc	1.10	1.10	100%
NdayModel	3.54	3.54	100%
BehaviorModel	3.60	9.47	263%
ClickAttribution	6.80	8.20	121%
SegmentExploder	14.10	25.20	179%
LogRead	1.10	1.10	100%
LogCount	1.10	0.04	4%

Table 6
Characteristics of intermediate data for k-mer (k=20) counting.

Input size (GB)	Int. data size (GB)	Int./Input
1	13.5	1350%
5	67.7	1354%
10	135.4	1354%
20	270.9	1355%
40	541.5	1354%
60	830.0	1383%
100	1381.0	1381%

More than 10 fold increase to data size

Data Compression

- Compressing data can help balance work between I/O and CPU
- I/O bound applications often find it worth it to spend extra CPU cycles to compress data so that I/O burden is lesser
  - Also helps with communication burden on the network
- For BPigs k-mer counting, enabling compression resulted in more than 50% drop in disk I/O and ~10% decrease in runtime

Data Compression Results

Table 7
IO Improvement from data compression.

Data size	Counter group(GB)	Uncompressed			Compressed			Difference		
		Map	Reduce	Total	Map	Reduce	Total	Map	Reduce	Total
40GB	Number of bytes read	604	598	1202	283	131	414	321	467	788
	Number of bytes written	1200	598	1798	550	131	681	649	467	1117
60GB	Number of bytes read	929	917	1846	442	167	609	487	751	1237
	Number of bytes written	1839	917	2756	853	167	1020	986	751	1736

Data Compression Results

Table 7
IO Improvement from data compression.

Data size	Counter group(GB)	Uncompressed			Compressed			Difference		
		Map	Reduce	Total	Map	Reduce	Total	Map	Reduce	Total
40GB	Number of bytes read	604	598	1202	283	131	414	321	467	788
	Number of bytes written	1200	598	1798	550	131	681	649	467	1117
60GB	Number of bytes read	929	917	1846	442	167	609	487	751	1237
	Number of bytes written	1839	917	2756	853	167	1020	986	751	1736

Spilling

- I/O is a big bottleneck, we want to reduce this cost as much as we can
- What happens when we run out of memory while working with intermediated data
  - The data must be written to disk. This is called a spill.
  - Spills during mapping are particularly problematic
- For k-mer counting, the large intermediate data can cause significant spilling overheads
  - Spill behavior can be controlled by block size and memory per container

Spilling

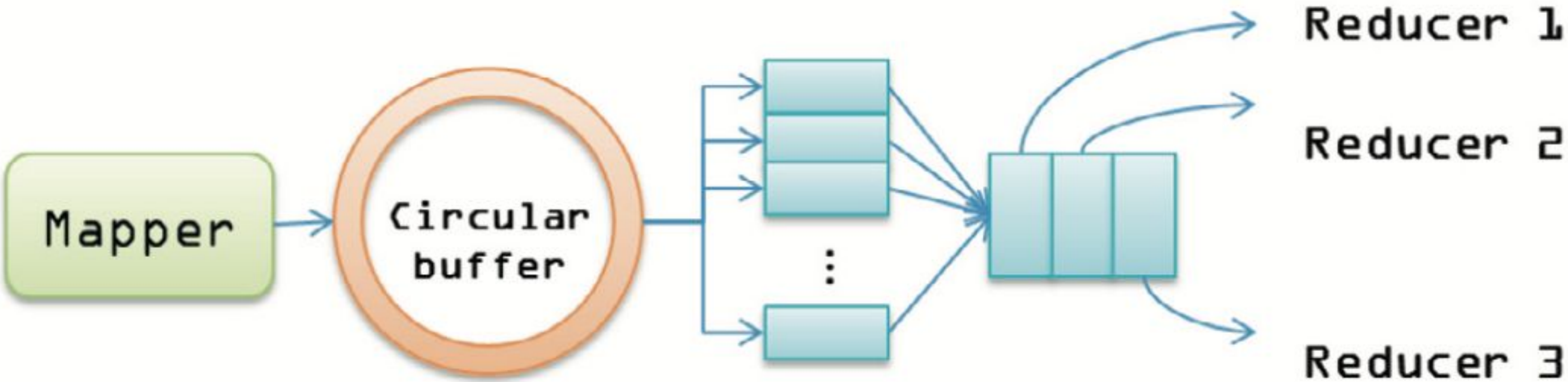
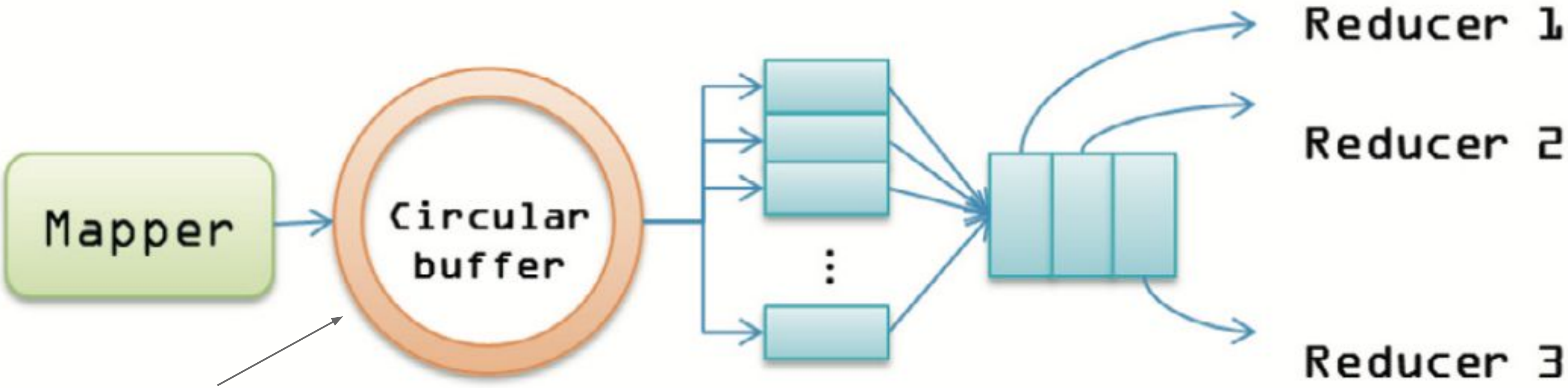


Fig. 1. The flow of data processing across MapReduce tasks.

Spilling



When this runs out of space, a spill occurs!

Fig. 1. The flow of data processing across MapReduce tasks.

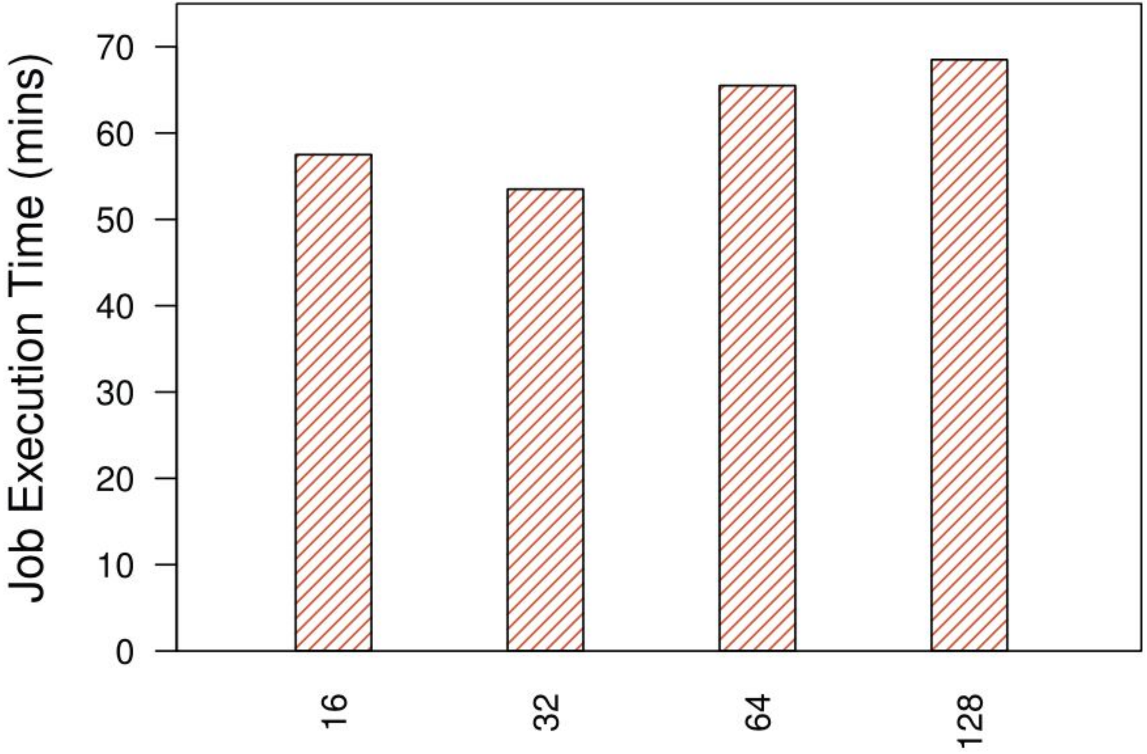
The Cost of Spilling

- One "spill" per map task is the ideal case (intermediate data just written out to disk once when task completes)
- If there is more than one spill for a map task, it results in 3x the I/O cost because the previous spill data must be read in, and sorted and merged with the current overflow data
- Keeping the number of spills to 1 per map task can be one of the most effective ways to improve MapReduce performance

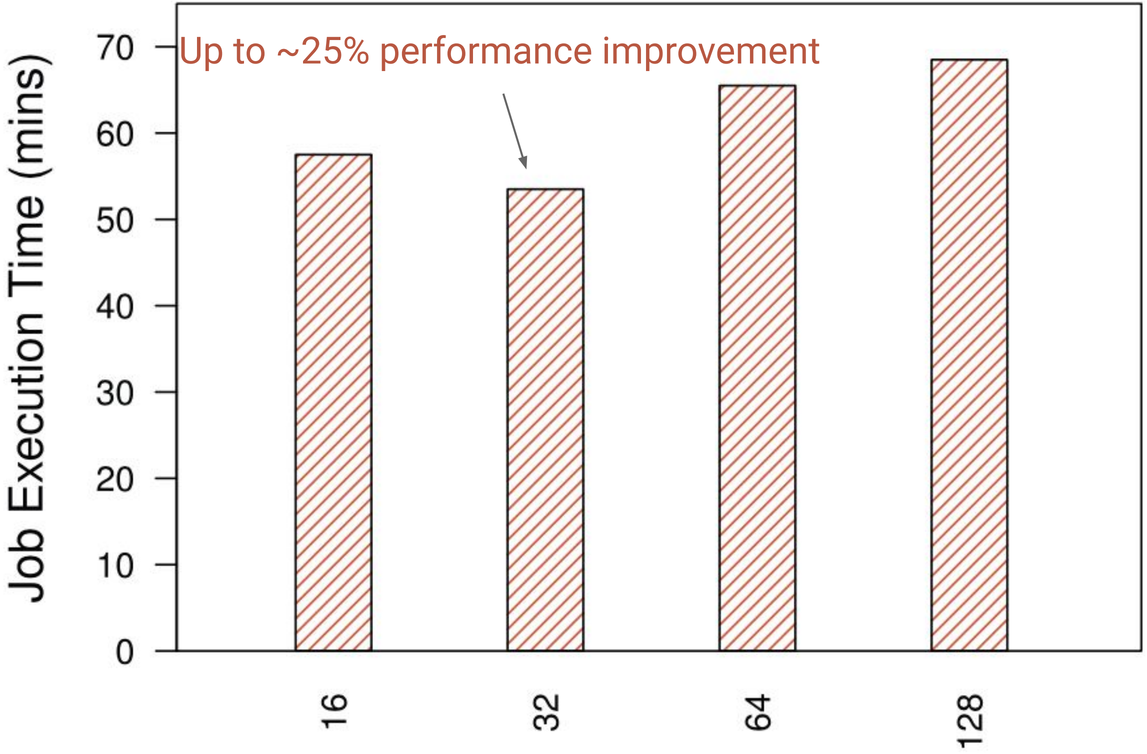
Spilling and Block Size

- Big block size is good to lower startup overhead for mappers
    - But it can lead to more spilling — especially with the large intermediate data size of k-mer
- Experimentation with BPig actually found lower block size to be better overall due to decrease in spill count
- Lower block size still left some mappers with multiple spills, so more memory was allocated to the maps ring buffer

Spilling and Block Size



Spilling and Block Size



Networking

- Overlapping map and shuffle can lead to benefits but can also interfere with mapping, and leave long running reduce tasks dangling
- Experimental results from this work showed best performance with no overlap whatsoever

Networking

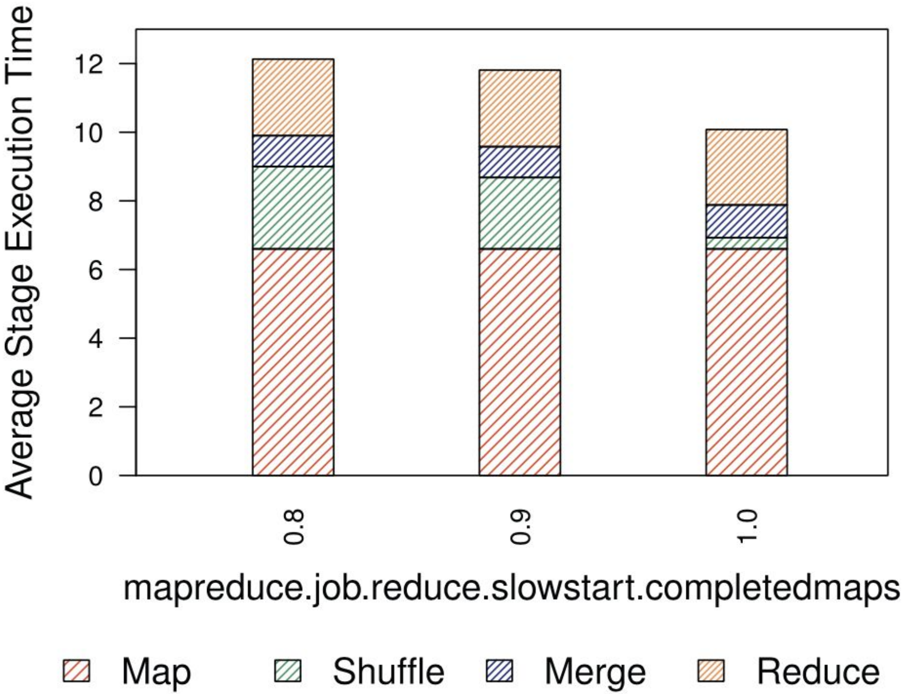
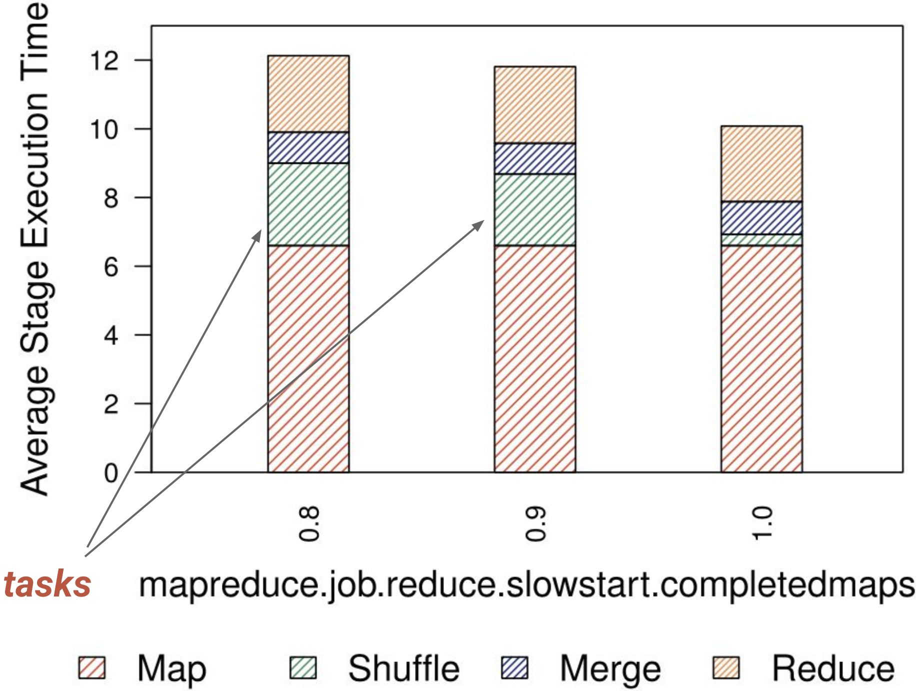


Fig. 5. Impact of reducer start time.

Networking



Long running shuffle tasks

Fig. 5. Impact of reducer start time.

Overall Results

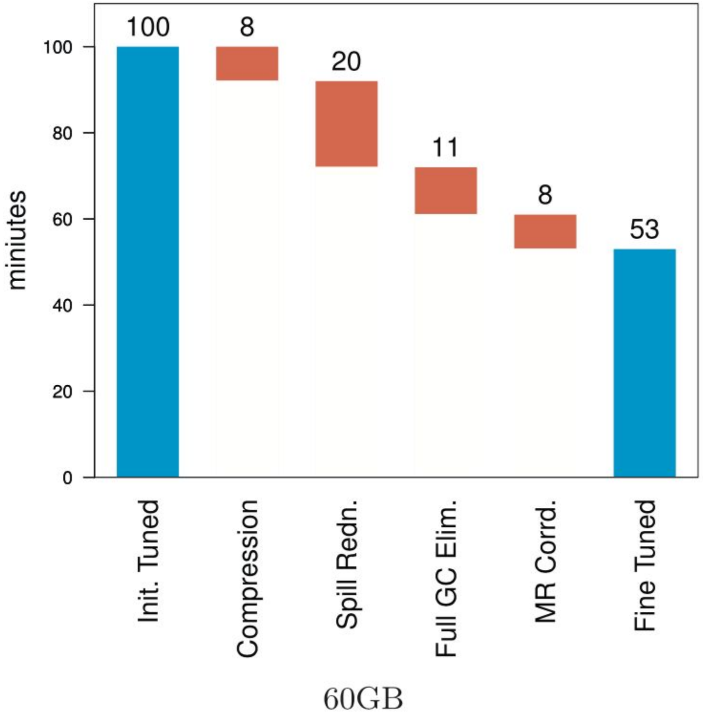


Fig. 6. Impact factors.

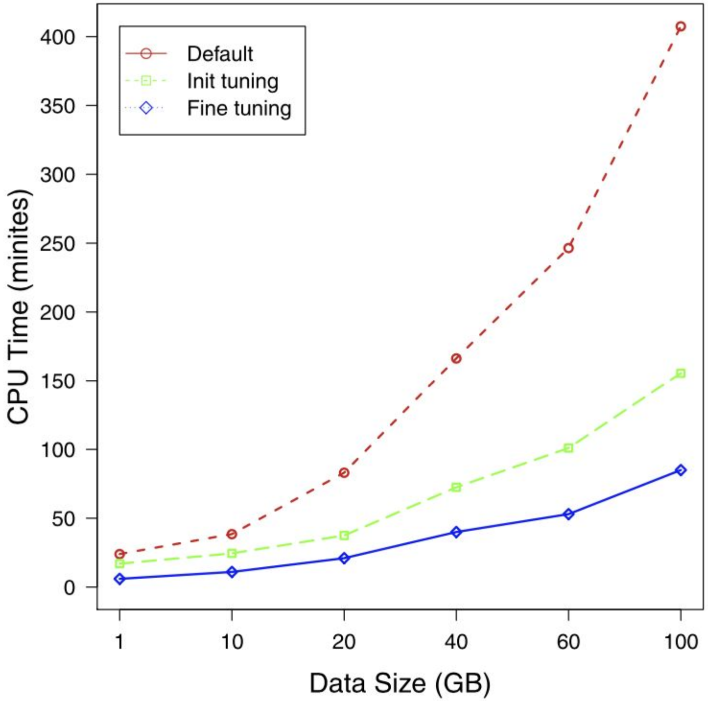


Fig. 7. Performance comparison.

Overall Results

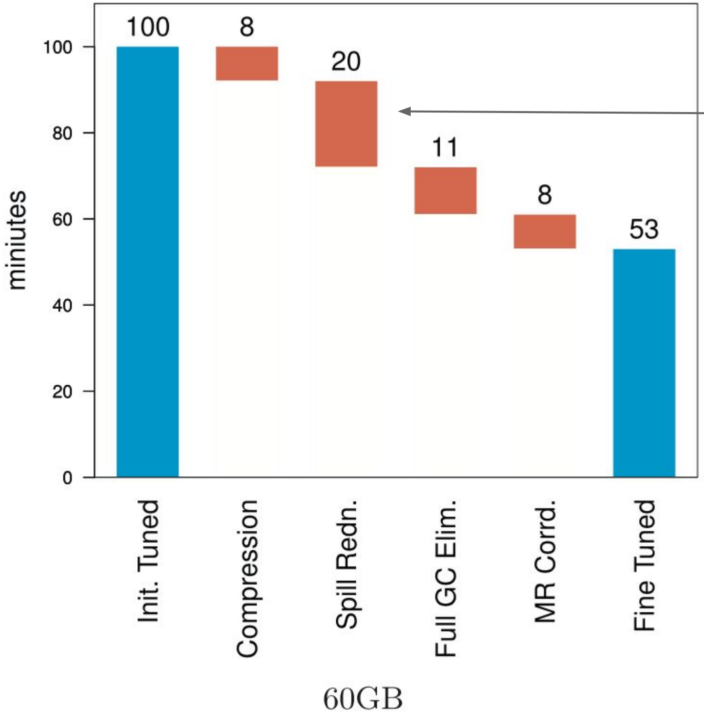


Fig. 6. Impact factors.

Spill reduction had a huge impact!

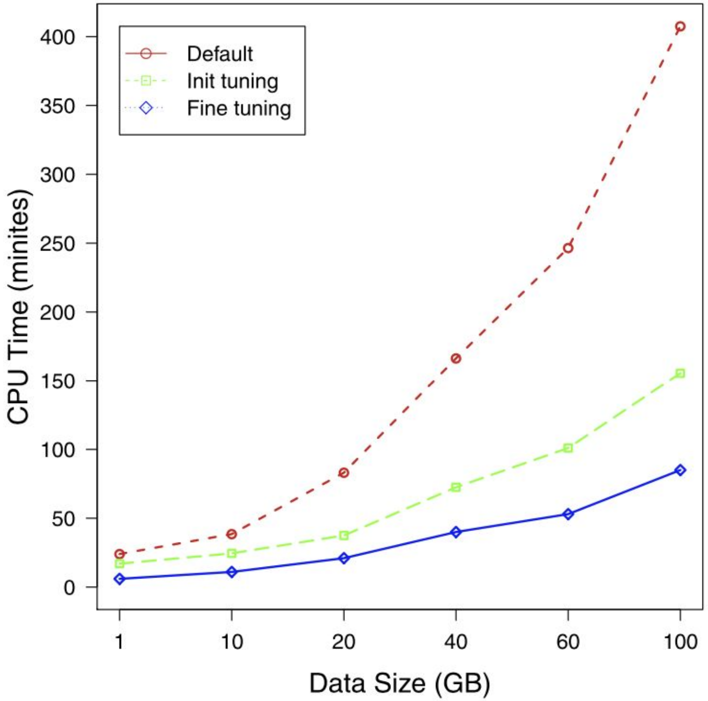


Fig. 7. Performance comparison.

Summary

- Bioinformatics has a number of relevant use cases for MapReduce
  - k-mer counting is a critical component for many other types of analysis
- I/O bottlenecks can cause significant issues for data-intensive MapReduce applications
- Tuning to eliminate extra spills and to keep I/O costs low can yield significant improvement
- Application stayed simple (and the same). Flexibility and simplicity of MapReduce is still a win!