Apache Spark
Introduction to Data Science
DATA11001

Nitinder Mohan
Collaborative Networking (CoNe)
nitinder.mohan@helsinki.fi
What is Apache Spark?

Scalable, efficient analysis of Big Data
What is Apache Spark?

Scalable, efficient analysis of Big Data
What is this Big Data?

Anything happening online, can be recorded!

- Click
- Ad impression
- Billing event
- Transaction
- Network message
- Fault
- .....
Where is this **Big Data** coming from?

User Generated Content (Web & Mobile)

Facebook
Instagram
Yelp
YouTube
Amazon
Twitter
and many more…..
Where is this **Big Data** coming from?

Health and Scientific Computing
Where is this **Big Data** coming from?

**Graph Data**
- Social Networks
- Telecommunication Networks
- Computer Networks
- Railway Networks
- LinkedIn Relationships
- Collaborations
- .......
What to do with **Big Data**?

Ummm… Remember what we said about Spark?
What is Apache Spark?

Scalable, efficient analysis of Big Data
What to do with Big Data?

Crowdsourcing + Physical modelling + Sensing + Data Assimilation

Basis of most scientific research
Traditional Analysis Tools

Unix shell commands (grep, awk, sed), pandas, numpy, R ..... 

Run on a Single machine node!

CPU
Memory
Disk
The Big Data Problem

1. Data is growing faster than compute speeds

Facebook’s daily logs: 60TB
1,000 genomes project: 200TB
Google web index: 10+PB
The Big Data Problem

1. Data is growing faster than compute speeds

2. Storage is getting cheaper
   - Size doubles after every 18 months

<table>
<thead>
<tr>
<th>Cost of 1TB disk:</th>
<th>$40</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time to read 1TB from disk:</td>
<td>3 hours (100MB/s)</td>
</tr>
</tbody>
</table>
The Big Data Problem

1. Data is growing faster than compute speeds

2. Storage is getting cheaper
   - Size doubles after every 18 months

Stalling CPU speeds and storage → Bottleneck!
The Big Data Problem

One machine cannot process or store all data for analysis!

Solution → Distribute!

Lots of hard drives
.. and CPUs
.. and memory!
Era of Cloud Computing
Datacenter Architecture

courtesy: ResearchGate
Breaking down analysis, “Parallely”

Q. Count the fans of two teams in a stadium!
Breaking down analysis, “Parallely”

Q. Count the fans of two teams in a stadium!
Breaking down analysis, “Parallely”

Solution 1

“I am stadium owner! Come to me and state your loyalty!”
OR...
Breaking down analysis, “Parallely”

Solution 2

“One person counts fans in each section and reports it to me!”
In Spark Cluster

Master/Driver node

Worker/Slave nodes
Principle of breaking computation in parts

Spark is based on MapReduce
MapReduce

Programming framework allowing distributing computation

- Two distinct tasks: Map and Reduce
- Divides dataset into key/value pairs
MapReduce in Action: Word Count

Map(1-2)  
Map(3-4)  
Map(5-6)  
Map(7-8)  

Reduce(A-H)  
Reduce(H-N)  
Reduce(O-Z)

"a banana is a banana not an orange because the orange unlike the banana is orange not yellow"
MapReduce in Action: Word Count

1. Each mapper receives the input in (key, value)

(Key, Value)
(1, a banana)
(2, is a banana)
(3, not an orange)
(4, because the)
(5, orange)
(6, unlike the banana)
(7, is orange)
(8, not yellow)
Sidenote: Meanwhile in Spark

Spark computation is based on Resilient Distributed Datasets (RDDs)

(1, a banana)
(2, is a banana)
(3, not an orange)
(4, because the)
(5, orange)
(6, unlike the banana)
(7, is orange)
(8, not yellow)

MapReduce \approx \text{Spark (key, value)}
RDD

Want to know more? Join Distributed Data Infrastructes (DDI) course (DATA11003) Instructed by Prof. Dr. Jussi Kangasharju. Registration opens 8\textsuperscript{th} Oct!

Moving on ...
MapReduce in Action: Word Count

1. Each mapper receives the input in (key, value)
2. Each mapper processes the (key, value)

(1, a banana) (2, is a banana) (3, not an orange) (4, because the) (5, orange) (6, unlike the banana) (7, is orange) (8, not yellow)

Map(1-2)

Map(3-4)

Map(5-6)

Map(7-8)

(1, a) (1, banana) (1, is) (1, a) (1, banana)

(1, not) (1, an) (1, orange) (1, because) (1, the)

(1, orange) (1, unlike) (1, the) (1, banana)

(1, is) (1, orange) (1, not) (1, yellow)

Reduce(A-H)

Reduce(H-N)

Reduce(O-Z)
MapReduce in Action: Word Count

(1, a banana)  
(2, is a banana)  
(3, not an orange)  
(4, because the)  
(5, orange)  
(6, unlike the banana)  
(7, is orange)  
(8, not yellow)  

1. Each mapper receives the input in (key, value)  
2. Each mapper processes the (key, value)  
3. Each (k,v) pair is sent to the reducer  

MapReduce:

- **Map(1-2)**: (1, a) (1, banana) (1, a)
- **Map(3-4)**: (1, is) (1, not)
- **Map(5-6)**: (1, because) (1, banana)
- **Map(7-8)**: (1, orange) (1, unlike)

- **Reduce(A-H)**: (1, an) (1, banana)
- **Reduce(H-N)**: (1, not)
- **Reduce(O-Z)**: (1, orange) (1, yellow)
MapReduce in Action: Word Count

1. Each mapper receives the input in (key, value)
2. Each mapper processes the (key, value) pair
3. Each (k,v) pair is sent to the reducer
4. The reducers sort their input by key

(1, a banana)
(2, is a banana)
(3, not an orange)
(4, because the)
(5, orange)
(6, unlike the banana)
(7, is orange)
(8, not yellow)

Each mapper receives the input in (key, value)

Map(1-2)
(1, a) (1, a)
(1, an)

Map(3-4)
(1, banana) (1, banana)
(1, banana)
(1, because)

Map(5-6)
(1, is) (1, is)
(1, not) (1, not)

Map(7-8)
(1, orange) (1, orange)
(1, orange)
(1, the) (1, the)
(1, unlike)
(1, yellow)

Reduce(A-H)

Reduce(H-N)

Reduce(O-Z)
**MapReduce in Action: Word Count**

1. Each mapper receives the input in (key, value)
2. Each mapper processes the (key, value) pair
3. Each (k,v) pair is sent to the reducer
4. The reducers sort their input by key
5. Reducers process their input one group at a time

- Map(1-2): (1, a banana), (2, is a banana)
- Map(3-4): (3, not an orange), (4, because the)
- Map(5-6): (5, orange), (6, unlike the banana)
- Map(7-8): (7, is orange), (8, not yellow)

- Reduce(A-H): (a, 2), (an, 1)
- Reduce(H-N): (is, 2), (not, 2)
- Reduce(O-Z): (orange, 3), (the, 2), (unlike, 1), (yellow, 1)
Spark use in industry

... and research of course!
Lets start a Spark cluster on a *real* datacenter eh?

Datacenter in question: **Ukko2**

<table>
<thead>
<tr>
<th>Component</th>
<th>Details</th>
</tr>
</thead>
</table>
| **31 compute nodes** | 28 cores  
              | 256GB RAM each! |
| **2 memory nodes**   | 96 cores  
                        | 3TB RAM each!   |
| **2 GPU nodes**      | 28 cores  
                        | 512GB RAM        |
|                        | 4 Tesla GPUs each!               |

Read about Ukko2 and how to use it:
https://wiki.helsinki.fi/display/it4sci/Ukko2+User+Guide
Spark Programming

Spark provides easy programming API for:

1. Scala
2. Python ✔️ PySpark
Anatomy of PySpark Application

application.py

Spark Context

Rest of your Python code
Spark Context

Tells the application on how to access the cluster

```python
conf = (SparkConf()
    .setAppName("First app")
    .setMaster("spark://ukko2-01:7077")
    .set("spark.cores.max", "10"))
sc = SparkContext(conf=conf)
```

read more on: https://spark.apache.org/docs/latest/configuration.html
Hands-on

Let’s make a PySpark application