Distributed Computation Framework

Present on 10/04/2017 for COMP 5311
By YEUNG Chun Kit, LI Qingjie
Outline

- Background and Motivation
- MapReduce
  - Design and Abstraction
  - Example and Result
- Graphlab
  - Background and Motivation
  - How it works
- Summary
  - Pros and cons
We have been fancied by the term of **Big Data** and **Artificial Intelligence**.

2,500,000 TB/day

- >4.51 billions of index web
- >1.81 billions of active user
- >6 billions of flickr photo

- Playing Chess
- Image Recognition
- Speech Recognition
- Natural Language Processing
- ……
However, our computation power haven’t grown as the data does. But Grace Hopper said:

“We shouldn’t be trying for bigger computer, but for more systems of computers.”
Today we are going to talk about two distributed computation framework:

Map-Reduce

Graphlab/Turi

Higher level of programming model to:

a. Hide lots of messy details, e.g. synchronization, race condition, communication among nodes...

b. Applied to large number of problems
Map-Reduce
Typical big data problems

- **Log Analysis**
  - How many GET request received in a day?
  - e.g. 127.0.0.1 user-identifier frank [10/04/2017:12:05:36 -0700] "GET /apache_pb.gif HTTP/1.0" 200 2326

- **Web Mining**
  - Which facebook pages about John Tsang have been viewed the most times during the 2017 HK Chief Executive Election?
  - How many “like” I received from each of my friends on facebook in the last year?
These kinds of problem requires computer to...

1. Reads each document in enormous of records.
2. Extract the interested information from each document.
3. Shuffle and sort the intermediate results in step 2.
4. Aggregate the intermediate results.
5. Generate the final output.

Map

Reduce

How to implement?

Functional Programming
Functional Programming

Map

Fold

Dataset

Intermediate Result

Aggregation Result

Final Result
Functional Programming - Example (Square Sum)

$$f(x) = x^2$$

$$g(x,y) = x+y$$

Dataset

Intermediate Result

Aggregation Result

Final Result
MapReduce Model

- **Map**
  - Maps \{ k1: v1 \} -> \[ \{ k2: v2 \} \]
  - Takes an input pair and produces a set of intermediate key/value pairs.
  - Groups all intermediate values associated with the same intermediate key I
  - Passes them to Reduce Function

- **Reduce**
  - Reduces \{ k2: [v2] \} -> \[ \{ k3, v3 \} \]
  - Accepts an intermediate key I and a list of values for that key.
  - Merges together these values to form a possibly smaller set of values.
  - Outputs the result
MapReduce Example (Word Count) - Pseudo Code

void map(String key, String value):
    // key: webpage url
    // value: webpage contents
    for each word w in value:
        EmitIntermediate(w, "1");

void reduce(String key, Iterator partialCounts):
    // key: a word
    // partialCounts: a list of aggregated partial counts
    int result = 0;
    for each pc in partialCounts:
        result += ParseInt(pc);
    Emit(AsString(result));
MapReduce Example (Word Count) - 2

Dataset split map sort shuffle Reduce Result

UST, DATA, 5311 → UST, DATA, 5311 → “UST”: 1 “DATA”: 1 “5311”: 1 → “UST”: [1] → “UST”: 1
COMP, COMP, 5311 → COMP, COMP, 5311 → “COMP”: 1 “COMP”: 1 “5311”: 1 → “DATA”: [1, 1] → “DATA”: 2
DATA, COMP, 5311 → DATA, COMP, 5311 → “DATA”: 1 “COMP”: 1 “5311”: 1 → “5311”: [1,1,1] → “5311”: 3

Result: “UST”: 1 “DATA”: 2 “5311”: 3 “COMP”: 3

Dataset: UST, DATA, 5311 COMP, COMP, 5311 DATA, COMP, 5311
More than Map Reduce

- **Combine**
  - Combine \{ k2: [v2] \} -> \{ k2: v2\_combined \}
  - Combine the data before it is sent over the network
  - Reduce the network traffic load.

- **Partition**
  - Partition the intermediate result
  - Allocate them into different reducer
MapReduce Example (Word Count) - Refined

Dataset
- UST, DATA, 5311
- COMP, COMP, 5311
- DATA, COMP, 5311

split
- UST, DATA, 5311
- COMP, COMP, 5311
- DATA, COMP, 5311

map combine
- "UST": 1
  "DATA": 1
  "5311": 1
- "COMP": 2
  "5311": 1
- "DATA": 1
  COMP": 1
  "5311": 1

sort shuffle
- "UST": [1]
- "DATA": [1, 1]
- "5311": [1, 1, 1]

Reduce
- Reduce worker 1
  "UST": 1
  "DATA": 2
  "5311": 3
- Reduce worker 2
  "COMP": 3

Result
- "UST": 1
- "DATA": 2
- "5311": 3
- "COMP": 3
Execution Overview

Adopt from MapReduce: simplified data processing on large clusters. Communications of the ACM
MapReduce in Machine Learning

- **Example: Linear Regression**
  - For a data $x$, predict the label: $y = \theta^T x$
  - For all $x_i$ in dataset $X$, find $\theta^*$ such that $y = X\theta^*$ produces minimum square error.
  - $\theta^*$ can be solved via normal equation $X^Ty = X^TX\theta^* \rightarrow \theta^* = (X^TX)^{-1}X^Ty$.
  - Denote $\theta^* = A^{-1}b$, where $A = X^TX$ and $b = X^Ty$.
  - $A = \sum_{i=1}^n(x_i x_i^T)$ and $b = \sum_{i=1}^n(x_i y_i)$

- **Other Algorithms**
  - When it does sum over the data, we can distribute the calculations over multiple cores.
  - Far more others algorithm...
  - However, not so good for iterative algorithm, e.g. Stochastic Gradient Descent
How it started?

Founded by Prof. Carlos Guestrin in Carnegie Mellon University in 2008

Two of his students worked on large scale distributed machine learning algorithms

They ran their model and found it took too long time and thus decide to build a system – Graphlab.
What is Graphlab?

- A new parallel framework for machine learning written in C++

- Open sourced and designed considering the scale, variety and complexity of real world data

- Incorporates various high level algorithms such as Stochastic Gradient Descent (SGD), Gradient Descent & Locking to deliver high performance experience
Benefits of using Graphlab

- Handles Large Data
  - Data structure of GraphLab can handle large data sets which result into scalable machine learning
  - SFrame:
    - Efficient disk-based tabular data
    - Helps to scale analysis and data processing to handle large data set
  - SGraph:
    - Helps understand networks by analyzing relationships between pair of items which is represented by edges
    - Scalable graph data structure which store vertices and edges in SFrames to perform a graph-oriented data analysis
Benefits of using Graphlab (1)

- Integration with various data sources
  - Supports various data sources like S3, JSON, CSV and more

- Data exploration and visualization
  - Browser-based interactive GUI which allows you to explore tabular data, summary statistics and bi-variate plots

- Feature Engineering
  - Has an inbuilt option to create new useful features to enhance model performance
Benefits of using Graphlab (2)

- **Modeling**
  - Has various toolkits to deliver easy and fast solution for ML problems

- **Production automation**
  - Data pipelines allow you to assemble reusable code task into jobs

- **GraphLab Create SDK**
  - Advance users can extend the capabilities of GraphLab Create using GraphLab Creat SDK
How Graphlab works

GraphLab

Model

Data Graph

Update Functions

Shared Data Table

Scheduling
Data Graph

A Graph with data associated with every vertex and edge.
Update Functions

Update Functions are operations which are applied on a vertex and transform the data in the scope of the vertex.

Allow asynchronous computation on the sparse dependencies.
Scheduler

Scheduler determines the order of Update Function Evaluations

Update function schedule:
Scheduler

Scheduler determines the order of Update Function Evaluations

Update function schedule:
Dynamic Schedule

- Update Functions can insert new tasks into the schedule
- Allow us to “focus” computation on the difficult part of the problem
Shared Data Table

Store global constant parameters

Sync Operation (Perform global computation)

- **Sync** is a fold/reduce operation over the graph
- **Accumulate** performs an aggregation over vertices
- **Apply** makes a final modification to accumulated data
## Summary

<table>
<thead>
<tr>
<th></th>
<th>MapReduce</th>
<th>Graphlab</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Pros</strong></td>
<td>(1) Simplicity of the model</td>
<td>(1) Allow dynamic asynchronous scheduling</td>
</tr>
<tr>
<td></td>
<td>(2) Code is generic and portable</td>
<td>(2) More expressive consistency model</td>
</tr>
<tr>
<td></td>
<td>(3) Easy to scale on large number of clusters</td>
<td>(3) Faster and more efficient runtime performance</td>
</tr>
<tr>
<td></td>
<td>(4) Applicable to many different systems and a wide variety of problems</td>
<td>(4) Able to model dependent and iterative tasks</td>
</tr>
<tr>
<td><strong>Cons</strong></td>
<td>(1) Restricted programming Constructs (only map and reduce)</td>
<td>(1) Non-deterministic execution</td>
</tr>
<tr>
<td></td>
<td>(2) Does not scale well for dependent tasks</td>
<td>(2) Substantially more complicated to implement</td>
</tr>
<tr>
<td></td>
<td>(3) Does not scale well for iterative algorithm</td>
<td></td>
</tr>
</tbody>
</table>
Reference:


Thank you! Any Questions?

YEUNG Chun Kit
MPhil Student, Computer Science and Engineering Department, HKUST
e-mail: ckyeungac@connect.ust.hk

Li Qingjie
MPhil Student, Computer Science and Engineering Department, HKUST
e-mail: qjli@connect.ust.hk