STATS 769
High Performance Computing

Paul Murrell

The University of Auckland

October 14, 2019

Overview

- This section of the course (two lectures) explores some more complex and larger-scale computational problems and solutions

Large Data and Parallel Computing Problems

- Data larger than mass storage (not just RAM).
- Jobs that take days (or weeks or months) to run.

- Job scheduling and load balancing (at scale).
- Fault tolerance (redundancy).
- Checkpointing.

Large Data and Parallel Computing Solutions

- Get an even bigger/faster computer
  - Supercomputers (NeSI)
  - GPUs
- Combine lots of smaller computers
  - Hadoop MapReduce
  - Apache Spark

Apache Hadoop

- HPC on standard hardware.
- Fault tolerance (data replication, node monitoring)
- Distributed file system (HDFS)
- Resource management and job scheduling (YARN)
- MapReduce engine

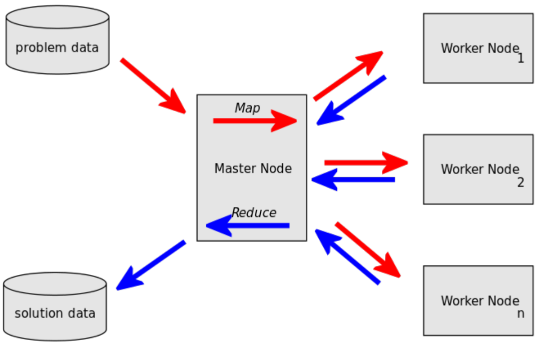
- We are only going to look at an R interface to Hadoop.

MapReduce

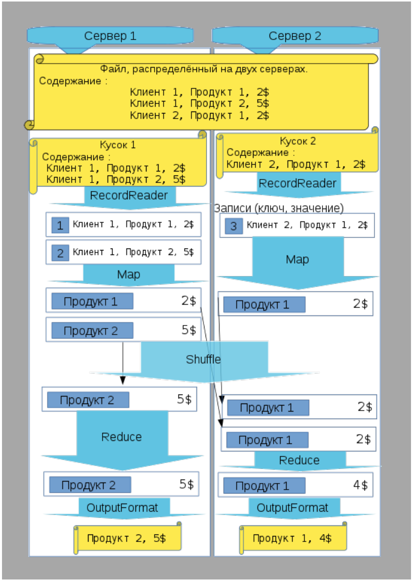
- Split step.
  Break input into chunks.
- **Map step.**
  For each chunk, generate one or more key-value pairs.
- Combine step.
  Collect all key-value pairs with same key.
- Shuffle/sort step.
  Transfer key-value pairs with same key to reducers.
- **Reduce step.**
  For each key, generate one or more key-value pairs.

- Fast and easy for certain types of jobs; slow and difficult for others (Spark the new alternative)
- Significant set up/configuration costs

MapReduce


problem data
Map
Master Node
Reduce
solution data
Worker Node 1
Worker Node 2
Worker Node n

MapReduce



Сервер 1
Сервер 2
Файл, распределённый на двух серверах.
Содержание :
Клиент 1, Продукт 1, 2$
Клиент 1, Продукт 2, 5$
Клиент 2, Продукт 1, 2$
Кусок 1
Содержание :
Клиент 1, Продукт 1, 2$
Клиент 1, Продукт 2, 5$
Кусок 2
Содержание :
Клиент 2, Продукт 1, 2$
RecordReader
RecordReader
Записи (ключ, значение)
1 Клиент 1, Продукт 1, 2$
2 Клиент 1, Продукт 2, 5$
3 Клиент 2, Продукт 1, 2$
Map
Map
Продукт 1 2$
Продукт 2 5$
Продукт 1 2$
Shuffle
Продукт 2 5$
Продукт 1 2$
Продукт 1 2$
Reduce
Reduce
Продукт 2 5$
Продукт 1 4$
OutputFormat
OutputFormat
Продукт 2, 5$
Продукт 1, 4$

rmr2

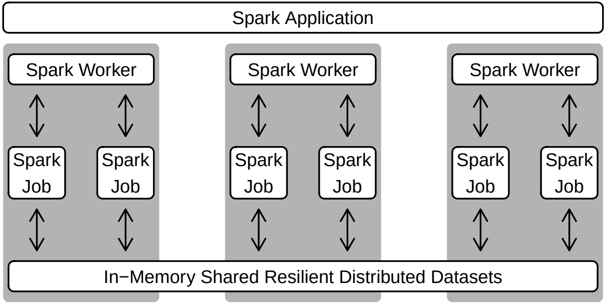
- The **rmr2** package provides an R interface to MapReduce (and Hadoop).
- We can define mappers and reducers in R code.
- Mapper and reducer work done in R.
- Input and output in Hadoop (disk based).
- Useful for Tidying and Transforming; harder to perform Modelling.
- `from.dfs()` to get result back to R.

Apache Spark

- Alternative to MapReduce.
- Reuse HDFS and YARN (or alternatives).
- In-memory computation (rather than disk-based).
- Better for iterative algorithms (as used in Modelling).
- Machine Learning library provided (for Modelling).

- We are only going to look at an R interface to Spark.

Spark


Spark Application
Spark Worker
Spark Worker
Spark Worker
Spark Job
Spark Job
Spark Job
Spark Job
Spark Job
Spark Job
In−Memory Shared Resilient Distributed Datasets

sparklyr

- The **sparklyr** package provides an R interface to Spark.
- R is just used to direct traffic; R code is converted to Spark (SQL) code.
- Work done in Spark (RAM based).
- R code creates references to Spark DataFrames (memory is in Spark not R).
- Useful for Tidying and Transformation and Modelling.
- Use **dplyr** functions for Tidying and Transforming; `dplyr` code converted to Spark code.
- Use `ml_*()` functions for Modelling.
- `spark::collect()` to get result back to R.

NeSI

- Mahuika:
  - Cray CS400 Cluster High Performance Computer
    8,424 x 2.1 GHz Intel Broadwell cores
    30 Terabytes of memory (in total)
  - Cray CS400 Virtual Labs Cluster
    640 x 2.1GHz Intel Broadwell Cores
    12.3 Terabytes of memory (in total)
    8 Nvidia Tesla P100 GPGPUs
- Maui:
  - Cray XC50-LC Supercomputer
    18,650 x 2.4GHz Intel Skylake cores
    66.8 Terabytes of memory (in total)
  - Cray CS500 Virtual Labs Cluster
    1,120 x 2.4GHz Intel Skylake cores
    21.5 Terabytes of memory (in total)
    8 Nvidia Tesla P100 GPGPUs

NeSI

- Designed for BATCH computing.
- `ssh` (and `scp`) to access.
- Write R script.
- Define jobs with a SLURM script
  (Simple Linux Utility for Resource Management)
- Use `sbatch` to submit SLURM script.
- Use `squeue` to view job status

GPUs

- GPUs have (many) more cores than CPUs
  (my desktop GPU has 1792 cores)
- GPUs can perform certain operations much faster than CPUs
- GPUs tend to have smaller RAM
  (my desktop GPU has 8 GB RAM)
- Code has to be written in a special language
  (e.g., CUDA for NVIDIA graphics cards, OpenCL for AMD and NVIDIA)
- The **gpuR** package provides an R interface for matrix operations via OpenCL.
- Some packages, like **keras**, automatically make use of GPUs.

Resources

- NeSI High performance computing and analytics
  `https://www.nesi.org.nz/services/high-performance-computing`
- **sparklyr**
  `https://spark.rstudio.com/`
- **RHadoop** (including **rmr2**)
  `https://://github.com/RevolutionAnalytics/RHadoop/wiki`