Big Data, easy as Pi

Phillipe Loher

This education session generously supported by Amazon via the AWS in Education Grant award

Outline

* Background
* Hadoop
* Amazon Web Services
* Word Count example ‘hands-on deployment’
* Pi ‘Battle the Greeks’ via Monte Carlo simulation

‘Big Data’ buzz

* Extremely large datasets that can be analyzed to reveal patterns, trends, and associations
* Affecting every industry:
  * Cancer/genetics research, climate and weather forecasting, physics simulations, supply chain optimization, food recipe optimization, …
* Changes the way experiments are done
* New algorithms are being formed
* Will add fuel to artificial intelligence and the semantic web

Traditional SAN based storage example

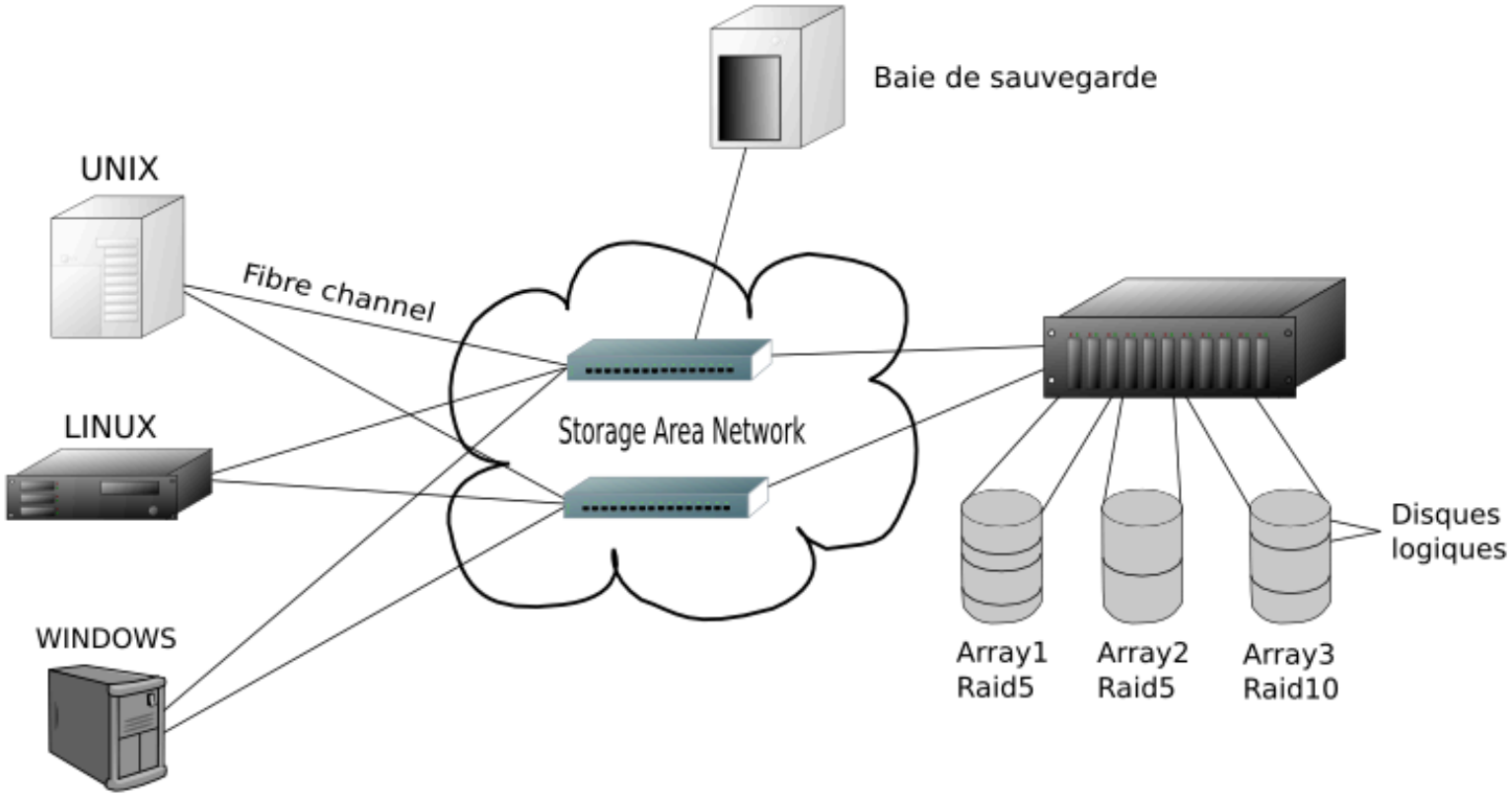


Image from: http://commons.wikimedia.org/wiki/File:Sch%C3%A9ma_SAN.png

A little background/history

* Multi-threading (e.g. POSIX threads, OpenMP)
* Job Scheduling Systems (e.g. PBS)
* MPI (Message Passing Interface)

What is Hadoop

* Software framework for distributed storage and distributed processing of Big Data
* Open source and Java based – inspired by Google's MapReduce & Google File System (proprietary & C++)
* Created in 2005, Yahoo is one of many prominent contributors
* 'Hadoop' often refers to the entire Hadoop Ecosystem which includes packages built on top of Hadoop (such as Apache Hive and Pig)

Hadoop disadvantages - not for everyone

* Not all algorithms can be easily tailored to fit within the framework
* Jobs don't communicate with one another
* Finding 'best' MapReduce settings across the nodes for a particular workload is not always obvious
* Not real time (work in progress) – batch processing
  * see Google Caffeine - http://www.theregister.co.uk/2010/09/09/google_caffeine_explained/

Hadoop advantages

* Well-defined framework that makes it easy to start utilizing distributed computing
* Reliable and fault tolerant
* Once your algorithm is written, can easily scale up by adding more compute power
* Bring the computing to the data (not the other way around)
* Free and Open Source

Hadoop Common Components

* **Hadoop Common** – common libraries/utilities
* **Hadoop YARN** – "Yet Another Resource Negotiator"
* **Hadoop Distributed File System (HDFS)** – a file system designed to run in a reliable and redundant manner across many low commodity machines
* **Hadoop MapReduce** – software framework that facilitates processing huge amounts of data on a large number of compute nodes, while reliably handling hardware and infrastructure failures
* The map reduce framework takes advantage of the distributed nature of HDFS (move workload to data)

Hadoop Distributed File System (HDFS)

* Namenode/Master (meta-data)
* Datanode (distributed)
* Large block sizes (64Mb / 128 Mb typical)
* Each data block replicated multiple times (for reliability and performance)
* Better for larger files

Word Count

* **Problem:** mine extremely large text files and count the # of occurrences of each word
* **Mapper:** write our own leveraging Hadoop Streaming
* **Reducer:** can write your own or use the built-in aggregator

http://hadoop.apache.org/docs/current/api/org/apache/hadoop/mapred/lib/aggregate/package-summary.html

MapReduce Overview

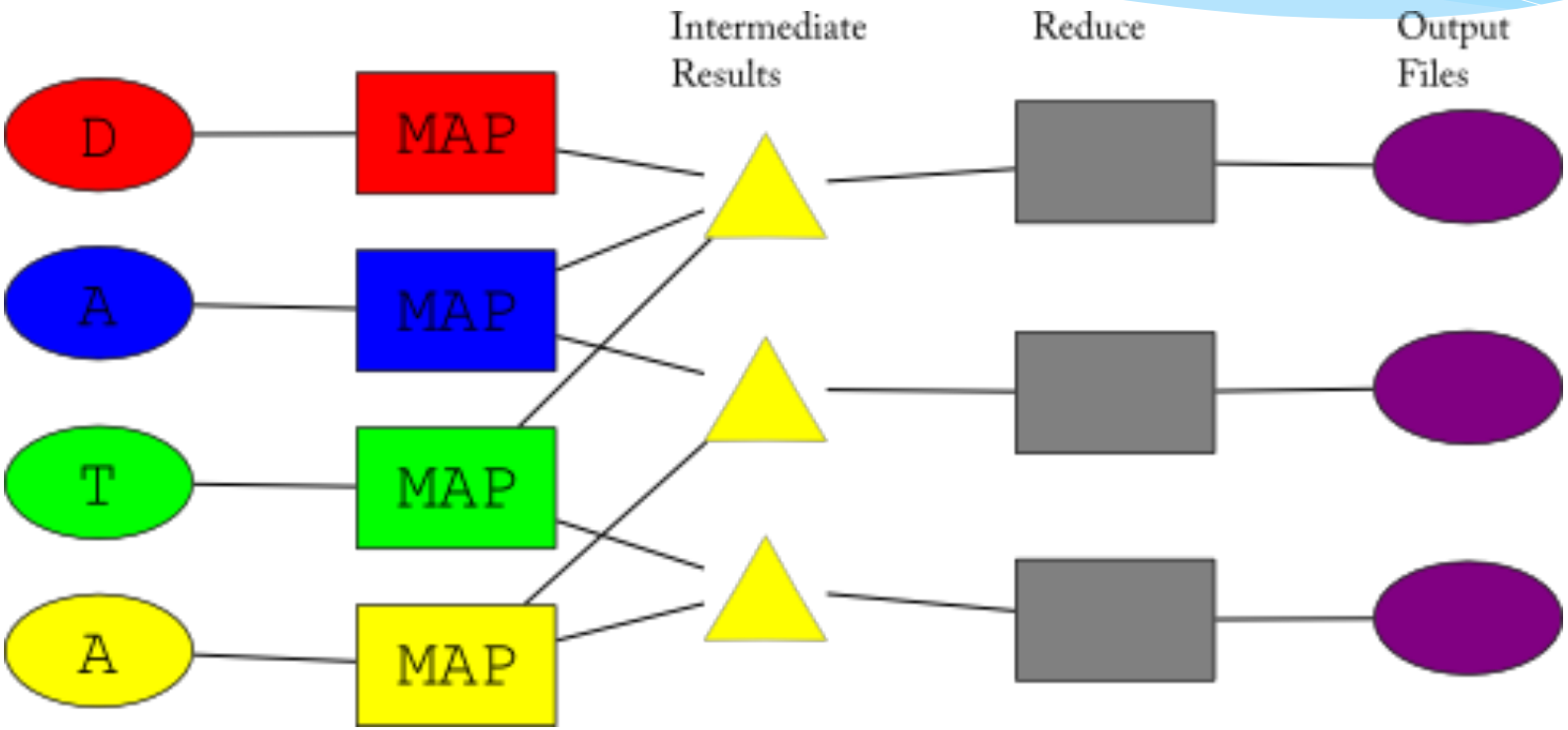


Image from: http://commons.wikimedia.org/wiki/File:Mapreduce_(Ville_Tuulos).png

Mapper Task

* Parallel Workhorse
* Transform input records to intermediate records
* Records are stored in key/value pairs
* **Other:**
  * Specifying a mapper is required
  * User can 'recommend' how many Mappers will be invoked. Ultimately determined by number of data blocks composing input files
  * Input files are optional (Pi example at end)

Reducer Task

* Takes intermediate records and reduces them
* Again, in key/value pairs
* **Important:** All matching keys from the Mapper's go to the same Reducer
* Can't start until all mappers have completed (though transferring/copying can begin)
* **Other:**
  * Can set number of reducers when running job
  * Reducers are optional (mapper-only workloads)

Hadoop Streaming

* Part of Hadoop that allows writing the Mapper and Reducer in any language you want
* Uses STDIN/STDOUT
* Disadvantages: Inter-process overhead and can't override/extend many of the MapReduce constructs

Mapper_Wordcount.pl

```
# read one line at a time
while (my $inputline = <STDIN>)
{
    # split the line into individual words
    my @matches = ($inputline =~ m/[a-zA-Z][a-zA-Z0-9]*/g);
    foreach my $match (@matches)
    {
        # print each word (in lower case)
        printf ("LongValueSum:%s\t1\n", lc ($match));
    }
}
~
```

Reducer_Wordcount.pl

```
# read in <key, value> pair from intermediate results generated by the MAPPER
foreach my $input (<STDIN>)
{
   chomp $input;

   # only get the key, throw away the value since we know it's always 1
   my $key = $input;
   $key =~ s/^LongValueSum://;
   $key =~ s/\t.*//;

   # aggregate the counts for that word (key)
   if (!defined ($aggregate{$key}))
   {
      $aggregate{$key} = 1;
   }
   else
   {
      $aggregate{$key} = $aggregate{$key} + 1;
   }
}

# print the count for each word (key)
foreach my $key (keys %aggregate)
{
   printf ("%s\t%ld\n", $key, $aggregate{$key});
}
~
~
```

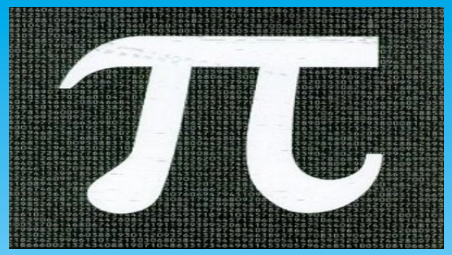
Resource Management

* **Jobtracker**
  * Receives the job submission and dispatches the work to all the tasks
  * Scheduling
  * Speculative execution (optional for stragglers)
  * Fault tolerance handling
  * Shuffle & Sort, synchronization
* **Tasktracker**
  * Sends heart beat messages to Jobtracker
  * Responsible for running the user's code
  * Sends machine availability info (such as free slot availability)
  * If using HDFS, runs alongside datanode daemon
* **YARN**
  * New in Hadoop 2
  * ResourceManager & NodeManager

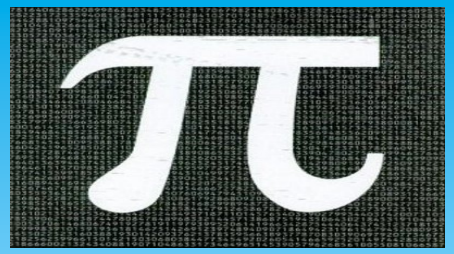
Amazon AWS

* **Amazon Web Services** – collection of remote computing services that form a cloud computing environment.
* **Amazon Simple Storage Service (S3)** – secure, durable, and scalable cloud-based storage
* **Amazon Elastic Compute Cloud (EC2)** – virtual computing environments
  * Different size (memory, cpu, storage, network capabilities) instances
  * Can pre-configure using Amazon Machine Images (AMIs)
* **Amazon Elastic Map Reduce (EMR)** – allows for quick deployment of Hadoop using the cloud

Hands-on AWS / Hadoop demo



Pi



* Using the Pythagorean Theorem and a 96-sided polygon, Archimedes of Syracuse (born 287 BC), approximated Pi to be between ~3.140845 and ~3.142857
* Pi is the 16th letter in the Greek alphabet
* Using distributed processing using MapReduce, can you do better than ancient math wizard Archimedes?

Enter Monte Carlo

* Monte Carlo allows for solutions to complicated problems to be approximated by the use of random numbers
* Modern version was invented during the development of the atom bomb. ENIAC was coded to carry out Monte Carlo simulations by John von Neumann.
* Useful when closed-form expression or deterministic algorithm is not possible (fluids, computational biology, quantum physics)

Approach and equations

* Randomly throw darts (in silico!) on dart board millions of times
* Use 1x1 unit dart board (quarter of a circle)
* Euclid: the area of a circle is proportional to r^2 and some constant.
* Area of circle: Pi*r^2
* Probability: (favorable outcomes / total outcomes)
* Probability of hitting in quarter of the unit circle: Pi / 4
* Estimate probability: (random darts landed in circle / total random darts thrown)
* Random darts: randomly choose x & y coordinate within unit circle. A^2 + B^2 = C^2 (Pythagorean theorem)