Hadoop Parallel Data Processing

Kai Shen

MapReduce and Implementation

- A programming interface (two-stage Map and Reduce) and system support such that:
  - the interface is easy to program, and suitable for many big data processing applications;
  - the underlying system can automatically support parallelism, data movement, load balancing, and fault-tolerance behind this interface.
- Original MapReduce implementation
  - Implemented in C/C++, but support many interfaces including Java
  - Used at Google, but not shared with the public

Hadoop

- Hadoop: Java-based implementation of MapReduce
- Originated at Yahoo but made open-source
  - Used by Facebook (more than 100 PetaBytes of data), Amazon (EC2 used by NYTimes), and many other places …
- Hadoop implements MapReduce
  - Multiple TaskTrackers run map()/reduce() tasks
  - Single JobTracker assigns tasks to TaskTrackers and manage the entire workflow
  - Each TaskTracker sends periodic heartbeats to the JobTracker for fault detection and management

Hadoop Distributed File System

- Big data applications typically run on a distributed file system
  - Too much data that must be spread over multiple nodes
    ⇨ a distributed file system tracks where the data is
  - A single file can be chopped up to blocks for distribution (allowing parallel accesses)
  - Replication for reliability
- Google's MapReduce works on Google File System (GFS)
- Hadoop works with Hadoop Distributed File System (HDFS)
  - Inferior to GFS in many ways (poor support for mutable data replication, incompatible with POSIX standard); but open-source

Hadoop System Structure

http://upload.wikimedia.org/wikipedia/en/2/2b/Hadoop_1.png

- JobTracker and NameNode work together to support intelligent task placement

Hadoop Programming

- Java-based, implement several interfaces
- Mapper()
  - Maps input key/value pairs to a set of intermediate key/value pairs
- Reducer()
  - Reduces a set of intermediate values which share a key to a smaller set of values.
- Tool()
  - Handles your custom command inputs or any other custom preparation work

Hadoop Task Mapping

- Hadoop tries to assign map tasks to where the data is
- Surprisingly where the data is also affects how many map tasks are launched
  - The real number is typically no smaller than the number you specify, but could be larger if there are more distributed splits of your data
  - Always check the completion status of your run to be sure about the number of map tasks

Hadoop Performance

- What affects Hadoop performance?
- Parallelism: number of map tasks; load at reduce
- Data locality
- Load balancing: uniform/heterogeneous nodes
- Speed at each map/reduce task

Assignment #2

- You will program three applications on the Hadoop parallel data processing platform.
- Goal: gain practical experience on Hadoop programming and learn the performance implication of parallel data processing.
- Work on your own.

Word Counting

- Where is it used in practice?
- The input is one or more text files.
- You should count the number of files each word appears in. Your word identification should be case-insensitive and ignore anything that isn't a letter.
- We provide two datasets: one for correctness testing and another for performance tests.

Matrix-Vector Multiplication

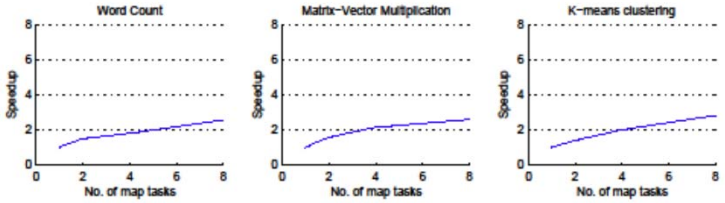
- Where is it used in practice?
- Matrix-vector multiplication takes a file describing a matrix as an input and has to also load a file with a vector. The matrix will be split across map tasks, but every task will need to load the entire vector.

K-means

- Widely used for data clustering in practice.
- Iterative. MapReduce for each iteration.

TA's Solution

- The TA has implemented his solutions for the three applications and tested on some sample datasets



- Speedup isn't always the right performance metric!

Our Hadoop Cluster

- One master node and 9 slave nodes
  - Small scale
- Status/usage check
  - Hadoop and HDFS
- Courtesy of using the shared Hadoop cluster
  - Particular for performance testing

Grading

- Grade on correctness and performance
- On performance, we care about the raw speed at 8 maps, not so much on the speedup over your own sequential solution, which can be poor

Miscellaneous Things

- Will post the assignment tomorrow
- E-turnin
- Start working early
- Strongly encourage early turn-ins (bonuses). Note that the last turn-in counts.
- Possibly run into issues, bear with us, potential award credit if discover and help solve problems with us

Data-Parallel Computing

- For some big data applications, data can fit into one machine and the main challenge is parallel data processing on multiprocessor
- MapReduce on multicore machines
  - Rationale: MapReduce is easy to program; big data application has already been written on MapReduce
  - "Phoenix Rebirth: Scalable MapReduce on a Large-Scale Shared-Memory", developed at Stanford

Data-Parallel Computing

- Data-parallel computing on multicore
  - Phoenix
  - Threads
  - Very different challenges and techniques for high performance and scalability (shared cache, synchronization, operating system scalability, …)
- GPUs