Parallel Data Processing with MapReduce

Mikael Högqvist

Zuse Institute Berlin
`hoegqvist@zib.de`

Introduction

- Data-set sizes increases towards infinity
  - Google unique URLs: 1B (2000/06) - 1T (2008/07), x1000 in 8 years!
  - LOFAR, LHC, Pan-STARRS
  - Location: people, vehicles, "things"
- How to process massive data-sets?

Processing System Issues

- Large data-sets are stored and processed on large distributed/parallel systems
- How to deal with
  - failures?
  - data consistency, placement, etc.?
  - how to schedule processing jobs?
- General goal: maximize parallel I/O available in the system

The MapReduce Framework

- Provide a user-friendly programming framework that simplifies parallel data processing
- Data modification, aggregation, filtering, generation
- Implemented as a library
    - Handle failures in software
    - Takes care of load-balancing, data movement and batch scheduling
    - Let the user deal with data formats

Programming with MapReduce

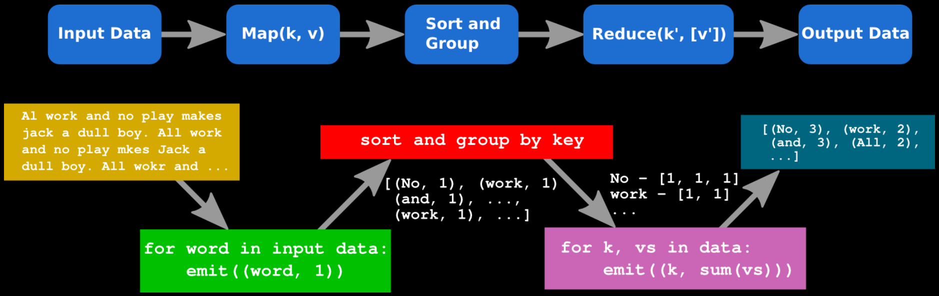
- Input: list of key, value-pairs
- $map(k, v) - (k', v')$
  - execute a function for each (key, value)-pair in the input and output a new (key, value)-pair
- $reduce(k', list(v')) - result$
  - aggregate, filter, transform values in list(v') for each key
- Output: list of results

Word Counting

```
Input: set of documents
Output: list of (word, occurrences)-pairs
def map(docid, content):
   for word in content:
       emit(word, 1)

def reduce(word, occurrence list):
   emit(word, sum(occurrence list))
```

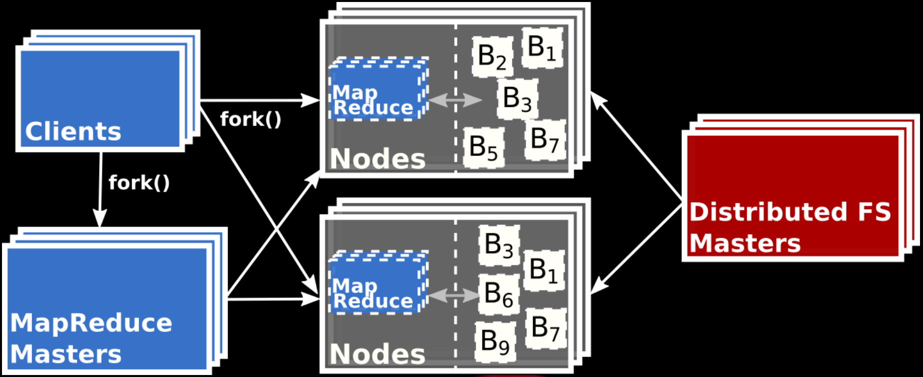
Execution Workflow


Input Data
Map(k, v)
Sort and Group
Reduce(k', [v'])
Output Data
Al work and no play makes jack a dull boy. All work and no play mkes Jack a dull boy. All wokr and ...
for word in input data:
    emit((word, 1))
sort and group by key
[(No, 1), (work, 1) (and, 1), ..., (work, 1), ...]
No - [1, 1, 1]
work - [1, 1]
...
for k, vs in data:
    emit((k, sum(vs)))
[(No, 3), (work, 2), (and, 3), (All, 2), ...]

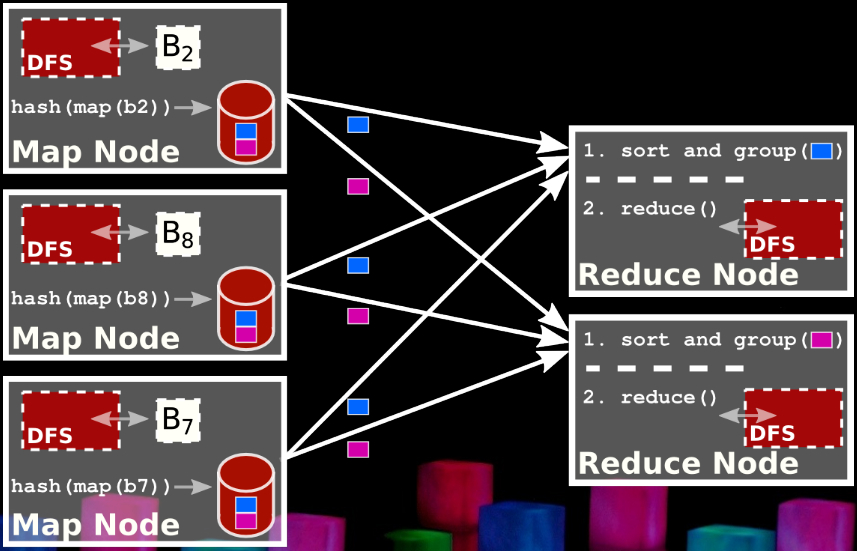
Parallel Execution

- Goal: Maximize available I/O!
- Partition data into equal sized blocks
- Map is independent, reading input data
- Wait for map phase, do sort and group by key on a partitioned set of keys
- Reduce is independent, writing out result data
- Move jobs to data, not data to jobs

System Architecture



Execution of a MapReduce Job



hash(k): 🟦 or 🟪

Real-world Usage Examples

- Search engine problems (e.g. Google, Yahoo)
  - Web access logs, inverted index creation
- Sorting 1PB in 6 hours and 2 minutes over 4000 machines
- NYT 11M old articles into PDF using Hadoop, Amazon EC2 and S3, cost?

Real-world Usage Examples

- Search engine problems (e.g. Google, Yahoo)
  - Web access logs, inverted index creation
- Sorting 1PB in 6 hours and 2 minutes over 4000 machines
- NYT 11M old articles into PDF using Hadoop, Amazon EC2 and S3, cost? $\leq$ **1000$**

Summary for ...

- ... Developers
  - “Automatic” parallel job
  - Simple transition from local to cluster/batch-system execution
  - Don’t worry about failure, load-balancing, scheduling
- ... System designers
  - Shared-nothing system with commodity hardware for nodes
  - Use a distributed/parallel file-system
  - Handle failures in software

Projects @ ZIB

- Data Management
  - XtreemFS - Distributed File System
  - Scalaris - Scalable key/value-store
  - Stellaris - Grid Metadata System (AstroGrid-D)
- Data-intensive processing
- **We are looking for large scale data intensive use cases!**

Links

- Hadoop, http://hadoop.apache.org/
- Cascading, http://cascading.org/
- MapReduce paper, http://labs.google.com/papers/mapreduce.html
- XtreemFS, http://xtreemfs.org/
- Scalaris, http://scalaris.googlecode.com/
- Stellaris, http://stellaris.zib.de/