CSinParallel: Using Map-Reduce to Teach Parallel Programming Concepts Across the CS Curriculum

Part 1

SC13

Dick Brown, St. Olaf College
Libby Shoop, Macalester College
Joel Adams, Calvin College
Workshop site

CSinParallel.org:
Workshops tab on left
SC13 Map-Reduce Workshop

See also workshop handout
Goals

– Introduce map-reduce computing, using the WebMapReduce (WMR) simplified interface to Hadoop
  • Why use map-reduce in the curriculum?
– Hands-on exercises with WMR for foundation courses
– Use of WMR for intermediate and advanced courses
  • What’s under the hood with WMR
  • A peek at Hadoop...
– Hands-on exercises for more advanced use
Introduction to Map-Reduce Computing
History

– The computational model of using map and reduce operations was developed decades ago, for LISP
– Google developed MapReduce system for search engine, published (Dean and Ghemawat, 2004)
– Yahoo! created Hadoop, an open-source implementation (under Apache); Java mappers and reducers
Map-Reduce Concept

Key, value pairs

map → K1, v1
map → Ki, v1
... → K2, v4
map → K1, v2
map → K1, v2
... → Kn, vn
map → K3, v5
... → K1, v3
map → reduce
... → reduce
... → reduce
... → reduce
... → reduce
The map-reduce computational model

- Map-reduce is a two-stage process with a "shuffle twist" between the stages.

- Stages are controlled by functions: mapper(), reducer()
The map-reduce computational model

• mapper() function:
  – Argument is one line of input from a file
  – Produces (key, value) pairs

• Example: word-count mapper()
  "the cat in the hat"  -->  [mapper for this line]
  ("the", "1"), ("cat", "1"), ("in", "1"),
  ("the", "1"), ("hat", "1")
The map-reduce computational model

• Shuffle stage:
  – group all mappers’ (key, value) pairs together that have the same key, and feed each group to its own call of reduce()
  – Input: all (key, value) pairs from all mappers
  – Output: Those pairs rearranged, sent to calls of reduce() according to key

• Note: Shuffle also sorts (optimization)
The map-reduce computational model

- reducer() function:
  - Receives all key-value pairs for one key
  - Produces an aggregate result

- Example: word-count reducer()
  
  \[
  \text{("the", "1"), ("the", "1")} \\
  \text{--> [reducer for "the"]} \\
  \text{("the", "2")}
  \]
The map-reduce computational model

– In map-reduce, a programmer codes only two functions (plus config information)
  • A model for future parallel-programming frameworks
– Underlying map-reduce system reuses code for
  • Partitioning the data into chunks and lines,
  • Moving data between mappers and reducers
  • Auto-recovering from any crashes that may occur
  • ...
– Optimized, Distributed, Fault-tolerant, Scalable
The map-reduce computational model

- Optimized, Distributed, Fault-tolerant, Scalable
Demo of WMR
Why teach map-reduce?
Why map-reduce for teaching notions of parallelism/concurrency?

– Concepts:
  • data parallelism;
  • task parallelism;
  • locality;
  • effects of scale;
  • example effective parallel programming model;
  • distributed data with redundancy for fault tolerance; ...
Why map-reduce for teaching notions of parallelism/concurrency?

– Real-World: Hadoop widely used
– Exciting: The appeal of Google, etc.
– Useful: for appropriate applications
– Powerful: scalability to large clusters, large data
Why WMR?

– Introduce concepts of parallelism
– Low bar for entry, feasible for CS1 (and beyond)
– Capture the imaginations of students

• Supports rapid introduction of concepts of parallelism for every CS student
WebMapReduce (WMR)

– Simplified web interface to Hadoop computations

– Goals:
  • Strategically Simple
    suitable for CS1, but not a toy
  • Configurable
    write mappers/reducers in any language
  • Accessible web application
  • Multi-platform, front-end and back-end
WMR Features (Briefly)

– Testing interface
  • Error feedback
  • Bypasses Hadoop -- small data only!

– Students enter the following information:
  • choice of language
  • data to process
  • definition of mapper in that language
  • definition of reducer in that language
WMR system information

- Languages currently supported:
  Java, C++, Python, Scheme, C, C#

- Back ends to date:
  local cluster, Amazon EC2 cloud images

More details about the system in Part 2 of the workshop
Teaching map-reduce with WMR in the introductory sequence
Teaching materials for WMR at an introductory level

• CSinParallel module: *Map-reduce Computing for Introductory Students using WebMapReduce*
  – See [csinparallel.org](http://csinparallel.org)
Experience

- Opportunity to informally introduce a host of PDC concepts
- Don’t expect speedup from our “tiny” problems
  - Hadoop is designed for terascale to petascale, and the overhead of I/O operations is dominant for mere megascale or less
Teaching with frameworks

- The syntax of Python mapper/reducer is familiar to any CS1 student with Python.

**But, programming with a framework:**
- They are used to writing entire programs.
- Challenge to understand how values come/go when it’s not standard input/output.

*Make a toy framework? Little success...*
Off-line exercise for students

• Manual map-reduce for students in-class
  – Makes the process concrete, which informs better understanding of the framework

• Let’s demonstrate by counting occurrences from a paper book
  • program, programmer, programming
  • design
  • code
  • technical
  • people
  • system, systems
Overview of suggested exercises

Available on the csinparallel.org site

– Run word count (provided), with small and large data
– Modify, run variations on word count: strip punctuation; case insensitive; etc.

– Alternative exercises
Quick questions/comments so far?
Hands-on

Module exercises

Data sets available

/shared/MovieLens2/movieRatings
/shared/gutenberg/WarAndPeace.txt
/shared/gutenberg/CompleteShakespeare.txt
/shared/gutenberg/all/group8
/shared/gutenberg/all_nonl/group8/