Map Reduce Discussion Notes
kuang chen (kuangc@eecs.berkeley.edu)

Problem
want to process lots of data - TBs
want to parallelize across datacenter
want to make it easy for lots of programmers

Existing work
use of a restricted programming model - "parallel prefix computation"
higher level abstractions to simplify parallel programming -
  Bulk Synchronous Programming and some MPI primitives
locality optimizations - "active disks"
running backup tasks - "eager scheduling"

Google's Approach
programming model borrows from functional programming
map (in_key, in_value) -> (out_key, intermediate_value) list
records from data source fed in as (key,value) pairs
map produces one or more intermediate values with an output key
reduce (out_key, intermediate value list) -> out_value list
after the map phase is finished, all intermediate values for a output key
  are combined into a list
reduce combines intermediate values into one or more final values
  per output key

MapReduce implementation
  takes care of the dirty work
  designed for google datacenters
key features - (see pg 3 figure 1)
  automatic parallelization, distribution
  1 master : n workers
  master divides work into M pieces (~64MB),
  assigns to workers user-defined map task; map results fall into R pieces
  workers reports map tasks are done; intermediate data is sorted
  master assigns R reduce tasks
  "reduce cannot start until map finishes"
  fault-tolerance
  master pings workers periodically
re-execute completed and in-progress map tasks
re-execute in-progress, but straggling reduce tasks
skips bad records
status and monitoring
simple, clean abstraction for programmers
locality
master assigns work according to location of data - same machine, rack...

Performance
sort experiment - beat the TeraSort benchmark - so is fast
high utilization
data input rate is quite high due to locality optimization
running eager backup tasks shortened runtime by 44%
10%+ simultaneous failure of computing nodes resulted in only 5% increase in execution time

Conclusion
MapReduce is a success at google - it’s used a lot (see pg 10, table 1)
Good a simplifying large-scale computation for programmers
Functional programming paradigm can be used for large-scale apps
A large variety of problems can expressed as MapReduce computations
Implementation scales to datacenter
But... specifically for Google problems, layered on Google infrastructure
It is a restricted programming model - what type of problems cannot run on it?