Jon Weissman
Distributed Computing Systems Group
Department of CS&E
University of Minnesota
Outline

• Motivation: Wide-area Clouds

• Multiple data centers: DMapReduce

• Multiple clouds: Proxy Cloud
Motivation

• Conventional cloud is an attempt to “tame” a distributed world
  – data, computation, people

• Wide-area bottlenecks
The Need

• Make the cloud more “wide-area aware”
  – consider data locality
  – consider end-user locality
  – consider cloud-cloud locality

• How?
  – global multi-data-center services
  – exploit the rich collection of edge computers
Big Data Trend: MapReduce

• Large-Scale Data Processing
  – Want to use 1000s of CPUs on TBs of data

• MapReduce provides
  – Automatic parallelization & distribution
  – Fault tolerance

• User supplies two functions:
  – map
  – reduce
Inside MapReduce

• MapReduce cluster
  – set of nodes N that run MapReduce job
  – specify number of mappers, reducers, \( \leq N \)
  – master-worker paradigm

• Data set is first injected into DFS

• Data set is chunked (64 MB), replicated three times to the local disks of machines

• Master scheduler tries to run map jobs and reduce jobs on workers near the data
Traditional MapReduce Clusters

• Distributed data
  – Replicated chunks
• Distributed computation
  – Map/reduce tasks
• Assumptions:
  – Data source close to compute nodes
  – Good connectivity
MapReduce Workflow

Lots of Dataflow
Big Data Trend: Distribution

• Big data is distributed
  – earth science: weather data, seismic data
  – life science: GenBank, NCI BLAST, PubMed
  – health science: GoogleEarth + CDC pandemic data
  – web 2.0: user multimedia blogs

Objective: How to run MapReduce across distributed data?
Wide-Area MapReduce

- Data in different data-centers
- Run MapReduce across them
- Data-flow spanning wide-area networks

DFS push
Option 1: Local MapReduce

Data Source (US) → MapReduce Job → Final Result

Data Push (Fast)

Data Center (US) → Data Source (EU)

Data Push (Slow)
Option 2: Global MapReduce

MapReduce Job

Final Result

Data Push (Fast)
Data Source (US)
Data Center (US)
Data Push (Slow)

Data Push (Fast)
Data Source (EU)
Data Center (EU)
Data Push (Slow)
Option 3: Distributed MapReduce

Data Source (US) → MapReduce Job (US) → Combine Results → Final Result → MapReduce Job (EU) → Data Source (EU)

Data Push (Fast)
Experimental Setup

• PlanetLab: 4 US, 4 EU nodes
• Amazon EC2: 3 US, 3 EU small instances
• Benchmarks:
  – Word count with text data: High compression
    • 800 MB PLab, 3.2 GB EC2
  – Word count with random data: Ballooning
    • 250 MB PLab, 1 GB EC2
  – Sort with random data: No aggregation
    • 250 MB PLab, 1 GB EC2
• Distributed MR works best in presence of data compression
PlanetLab: WordCount (random data)

- Local MR works best in presence of data ballooning
PlanetLab: Sort

- Similar performance in presence of “equal” data
EC2 Results

WordCount (Text)

WordCount (Random)

Sort
Observations

• Amount of data compression is a critical factor
  – High compression => avoid initial data push cost
  – High ballooning => avoid result combine cost

• Make MapReduce topology-aware
  – Data push should consider locality
  – Task scheduling should consider co-placed execution
Global MR

- Skew in data sizes at each site
- Uneven compute resources in different DCs
  - Can be useful in presence of heterogeneity, failures
- Is a generalization of Local MR and Distributed MR
Intelligent Data Placement

• HDFS push
  — local node, same rack, random rack

  Resource Topology
  Application Characteristics

Data placement Scheduling

/DC/rack\textsubscript{A}/node\textsubscript{X}

Data expansion factors
input->intermediate, \( \alpha \)
Intermediate->output, \( \beta \)

select LMR, DMR, GMR

Generalize beyond HDFS/MapReduce!
Proxy Cloud
Cloud landscape: Diversity and Specialization

• Data clouds
  – S3, SkySurvey, GoogleHealth

• Compute clouds
  – EC2, IronScale

• Service clouds
  – Gmail, Gmaps, Google-earth

• Specialization
  – Non-functional: security, reliability, SLAs, cost
Confluence

• Cloud diversity and specialization =>
  – (1) No single cloud model will rule
  – (2) New distributed models are attractive
  – (3) Emerging applications will utilize multiple clouds “multi-cloud” applications
Multi-Cloud Applications

• Distributed data mining
  – Ex: weather data + commodity prices

• Scientific workflows
  – Ex: life science: GenBank<->BLAST<->PubMed, ...

• Mashups
  – Ex: GoogleEarth + CDC pandemic data

• Multi-cloud/DC parallel frameworks
  – Ex: MapReduce
The Problem

- Cloud-cloud bottlenecks
- User-cloud bottlenecks
Multiple Clouds: Proxy Cloud

• Using edge nodes with the commercial cloud(s)

• Goal to provide
  – cloud <-> cloud locality
  – cloud <-> end-user locality
  – cloud- <-> data locality
Solution: Proxy Architecture: 50K ft
Proxy Roles

• Cloud service interaction
  – Proxy as a client

• Routing
  – Proxy routes data to other proxies

• Computing => Grids
  – Proxy computes data operators: compress, filter, merge, mine, ...

• Caching => P2P
  – Proxy caches data (from cloud, computations, ...)
Networking Benefits

(a) TCP bandwidth (KBps) (Higher is better)

(b) UDP bandwidth (KBps) (Higher is better)

(c) UDP delay (ms) (Lower is better)

(d) UDP jitter (ms) (Lower is better)
Many better data paths ...
Some acceleration is large ...

How do we get this information?
Network dashboard: netstat.cs.umn.edu
Workflow Acceleration
Example: Montage
A-E communication role
F-G computation role
From data server to compute node

Green proxy accelerates communication 20-75%
Inside Montage ...
From Montage to the end-user
Desktop User
Mobile user
Summary

• Conventional clouds attempt to tame a “distributed world”
  – data, computation, people

• Wide-area “awareness” can be key to performance
  – Cloud-cloud, Cloud-user interactions
  – Wide-area MapReduce
Thank you! Questions?