Jon Weissman
Distributed Computing Systems Group

Department of CS&E
University of Minnesota
Introduction

• “Cloud” Context
  - fertile platform for scheduling research
  - re-think old problems in new context

• Two scheduling problems
  - mobile applications across the cloud
  - multi-domain MapReduce
The “Standard” Cloud

Data in

Computation

Results out

“No limits”
• Storage
• Computing
Multiple Data Centers

Virtual Containers
Cloud Evolution => Scheduling

- **Client technology**
  - devices: smart phones, ipods, tablets, sensors

- **Big data**
  - 4th paradigm for scientific inquiry

- **Multiple DCs/clouds**
  - global services

- **Science clouds**
  - explicit support for scientific applications

- **Economics**
  - power and cooling “green clouds”
Our Focus

- Power at the edge
  - local clouds, ad-hoc clouds

- Cloud-2-Cloud
  - multiple clouds

- Big data
  - locality, in-situ

- Mobile user
  - user-centric cloud

Nebula
Proxy
DMapReduce
Mobile cloud
Mobility Trend: Mobile Cloud

- Mobile users/applications: phones, tablets
  - resource limited: power, CPU, memory
  - applications are becoming sophisticated

- Improve mobile user experience
  - performance, reliability, fidelity
  - tap into the cloud based on current resource state, preferences, interests

=> user-centric cloud processing
Cloud Mobile Opportunity

- **Dynamic outsourcing**
  - move computation, data to the cloud dynamically

- User context
  - exploit user behavior to pre-fetch, pre-compute, cache
Application Partitioning

- Outsourcing model
  - local data capture + cloud processing
  - images/video, speech, digital design, aug. reality
Application Model: Coarse-Grain Dataflow

for i=0 to NumImagePairs
    a = ImEnhance.sharpen (setA[i], ...);
    b = ImAdjust.autotrim (setB[i], ...);
    c = ImSizing.distill (a, resolution);
    d = ImChange.crop (b, dimensions);
    e = ImJoin.stitch (c, d, ...);
    URL.upload (www.flickr.com, ...., e);
end-for
Scheduling Setup

- Components $i, j, \ldots$
- $A_{ij}$ - amount of data flow between components $i$ and $j$
- Platforms $\alpha, \beta, \gamma, \ldots$ (mobile, cloud, server, \ldots)
- $D_{\alpha,i}.type$ - execute time, power consumed for $i$ running on $\alpha$
- $Link_{\alpha\beta,k}.type$ - transmit time, power consumed for $k$th link between $\alpha\beta$
- All assumed to be with respect to input $I$
- On-line runtime measurement based on prior inputs from an input set
Experimental Results - Image Sharpening

- **Response time**
  - both WIFI & 3G
  - up to 27× speedup
  - 219K, WIFI

- **Power consumption**
  - save up to 9× times
  - 219K, WIFI
Experimental Results - Face Detection

- **Face Detection**
  - identify faces in an image

- **Tradeoffs**
  - power, response

- **User specifies tradeoffs**
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
MapReduce Workflow

DFS push

Input

Intermediate

k1:v k1:v k2:v
k1:v k3:v k4:v
k4:v k5:v
k4:v k1:v k3:v

shuffle

Group by Key

Grouped

k1:v,v,v,v
k2:v
k3:v,v
k4:v,v,v
k5:v

Output
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
Context: Widely distributed data

Data in different data-centers
Run MapReduce across them
Data-flow spanning wide-area networks
Data Scheduling: Wide-Area MapReduce

Local MapReduce (LMR)

Global MapReduce (GMR)

Distributed MapReduce (DMR)
DMR is a great idea if output $<<$ input
LMR and GMR are better in other settings
Intelligent Data Placement

- **HDFS**
  - local cluster, nearby rack, random rack

```
/DCi/rackA/nodeX
```

```
??????
```

```
LMR, DMR, GMR
```

```
??????
```

```
static or observed
```

```
Resource Topology
```

```
Data placement Scheduling
```

```
Application Characteristics
```

```
``
Problem: Data Scheduling

- Data movement is dominant
- Data sets located in domains, size: \( D_i \), \( \ldots \), \( D_m \)
- Platform domains: \( P_j \), \( \ldots \), \( P_k \)
- Inter-platform bandwidth: \( BD_i P_j \)
- Data expansion factors
  - input->intermediate, \( \alpha \)
  - Intermediate->output, \( \beta \)

\( \Rightarrow \) select LMR, DMR, GMR
Summary

• Cloud Evolution
  - mobile users, big data, multiple clouds/data centers
  - many scheduling challenges

• Cloud Opportunities
  - new context for old problems
  - application partitioning (mobile/cloud)
  - data scheduling (wide-area MapReduce)