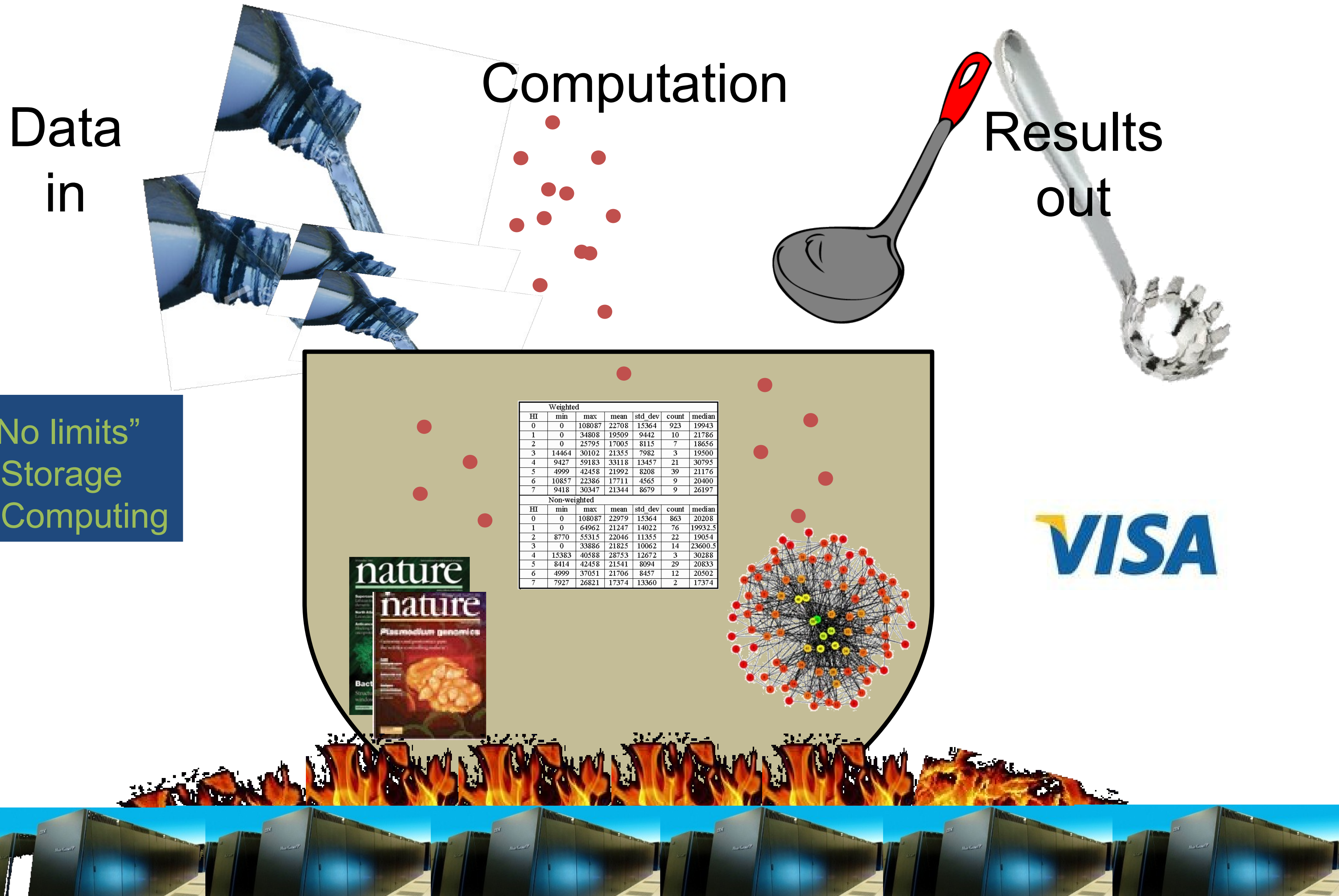
Jon Weissman
Distributed Computing Systems Group

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

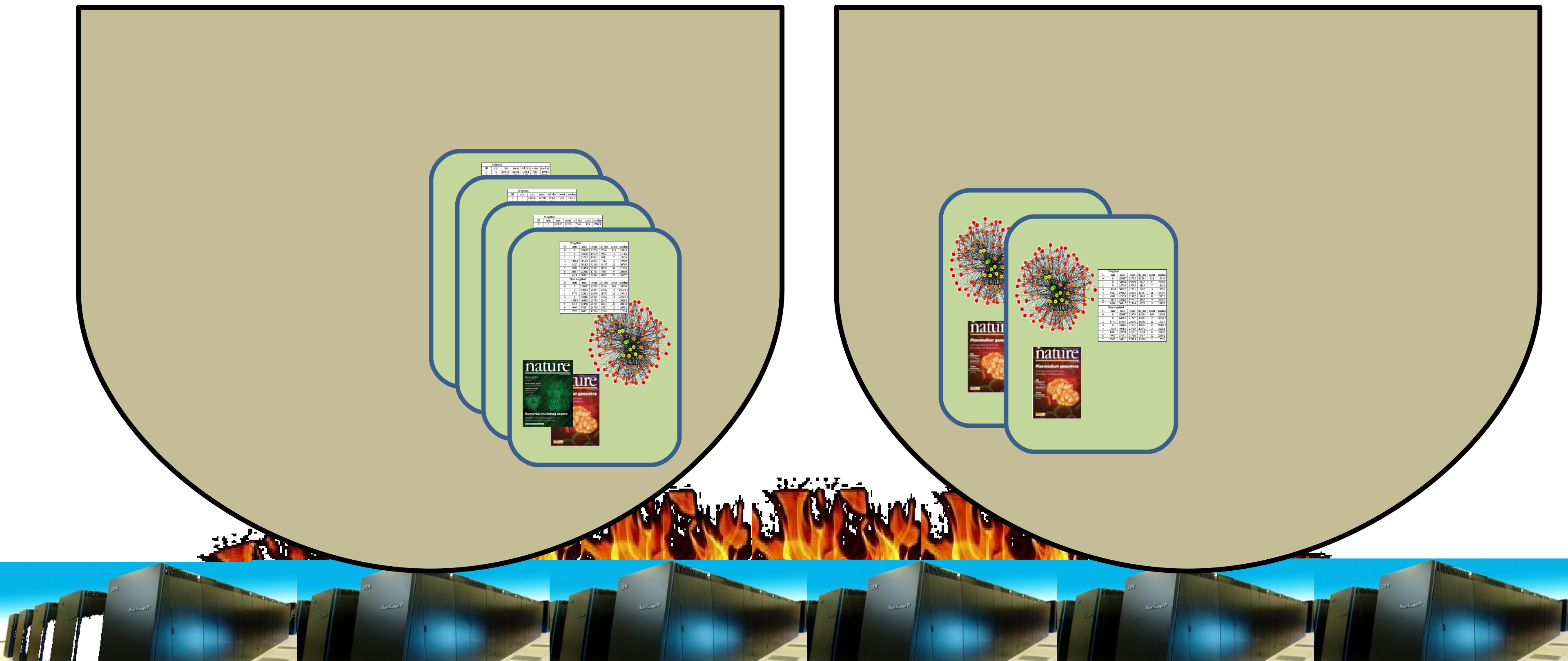


Weighted						
HI	min	max	mean	std_dev	count	median
0	0	108087	22708	15364	923	19943
1	0	34808	19509	9442	10	21786
2	0	25795	17005	8115	7	18656
3	14464	30102	21355	7982	3	19500
4	9427	59183	33118	13457	21	30795
5	4999	42458	21992	8208	39	21176
6	10857	22386	17711	4565	9	20400
7	9418	30347	21344	8679	9	26197
Non-weighted						
HI	min	max	mean	std_dev	count	median
0	0	108087	22979	15364	863	20208
1	0	64962	21247	14022	76	19932.5
2	8770	55315	22046	11355	22	19054
3	0	33886	21825	10062	14	23600.5
4	15383	40588	28753	12672	3	30288
5	8414	42458	21541	8094	29	20833
6	4999	37051	21706	8457	12	20502
7	7927	26821	17374	13360	2	17374

“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

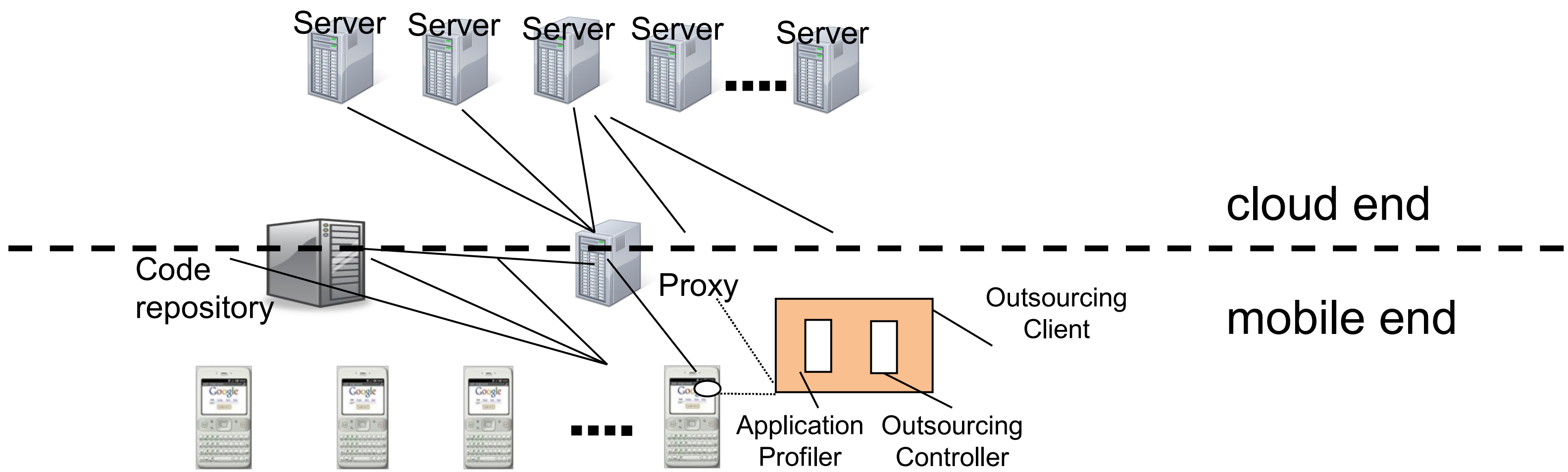
=> **user-centric cloud processing**

Cloud Mobile Opportunity

- **Dynamic outsourcing**
  - move computation, data to the cloud dynamically
- User context
  - exploit user behavior to pre-fetch, pre-compute, cache

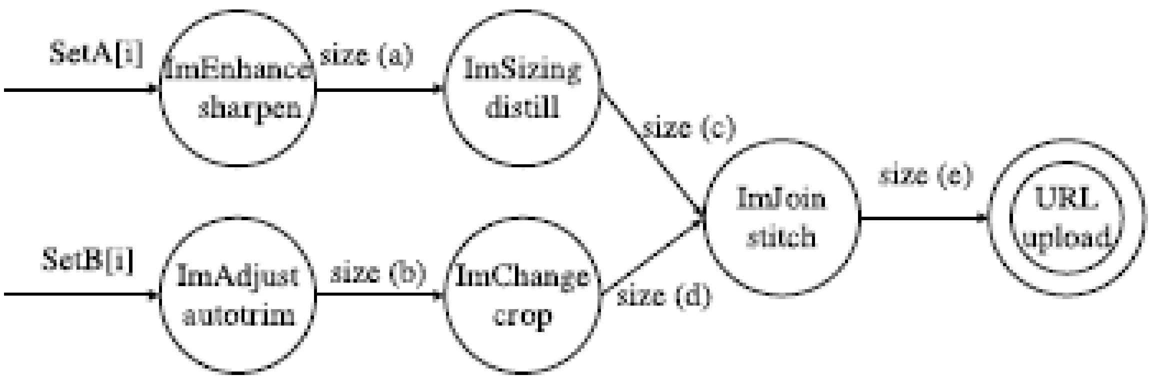
Application Partitioning

- 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);
```

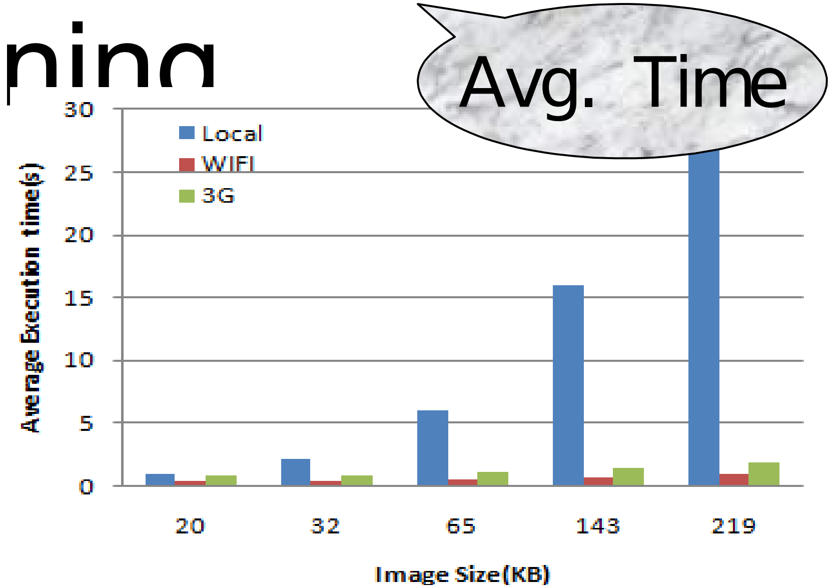


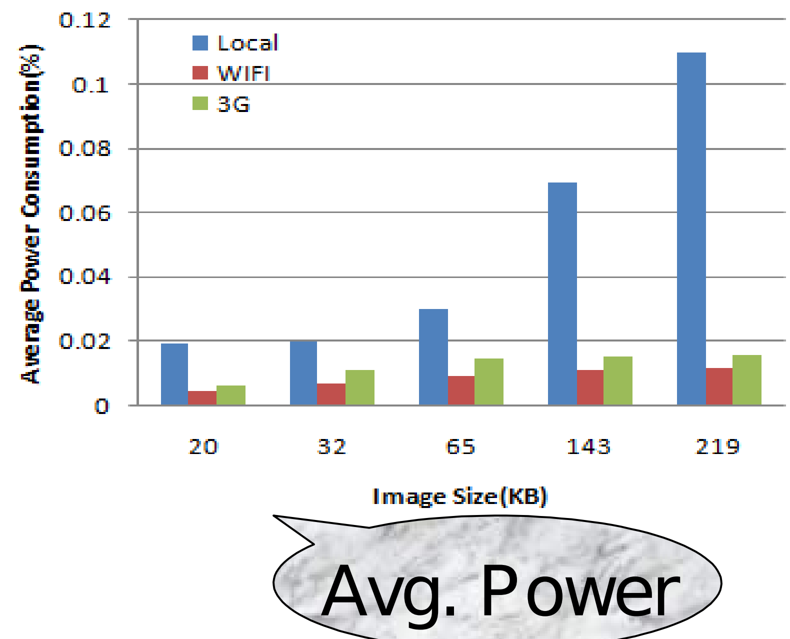
Scheduling Setup

- Components *i, j, …*
- A_{ij} - amt of data flow between components *i* and *j*
- Platforms α, β, γ, ... (mobile, cloud, server, …)
- $D_{\alpha,i}$.*type*
  - execute time, power consumed for *i* running on α
- $Link_{\alpha\beta,k}$.*type*
  - transmit time, power consumed for kth link between αβ
- All assumed to be w/r Input *I*
- On-line runtime measurement based on prior

Experimental Results -Image Sharpening

- Response time
  - both WIFI & 3G
  - up to 27× speedup
  - 219K, WIFI
- Power consumption
  - save up to 9× times
  - 219K, WIFI

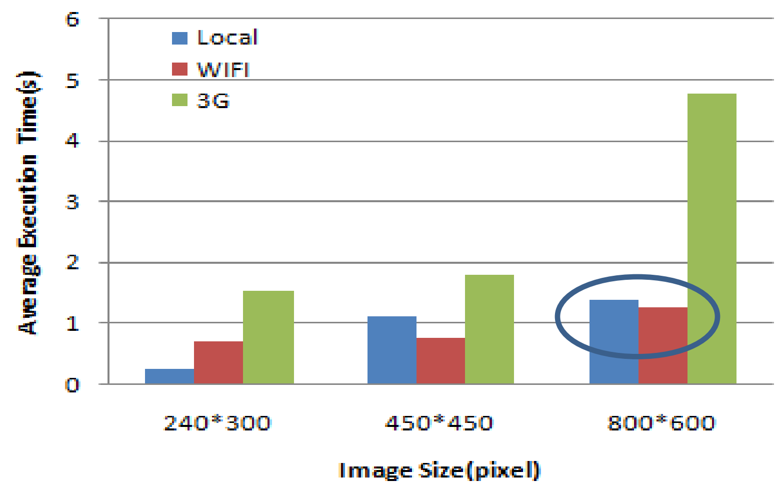




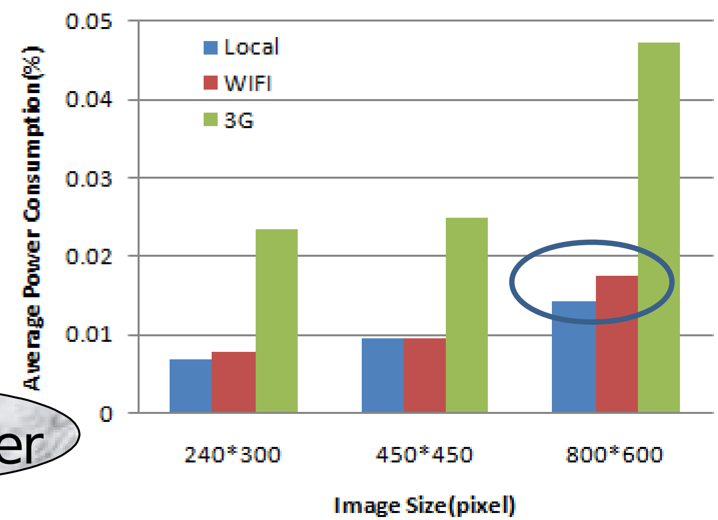
Experimental Results-Face Detection

- Face Detection
  - identify faces in an image
- Tradeoffs
  - power, response
- User specifies tradeoffs

Avg. Time



Avg. Power



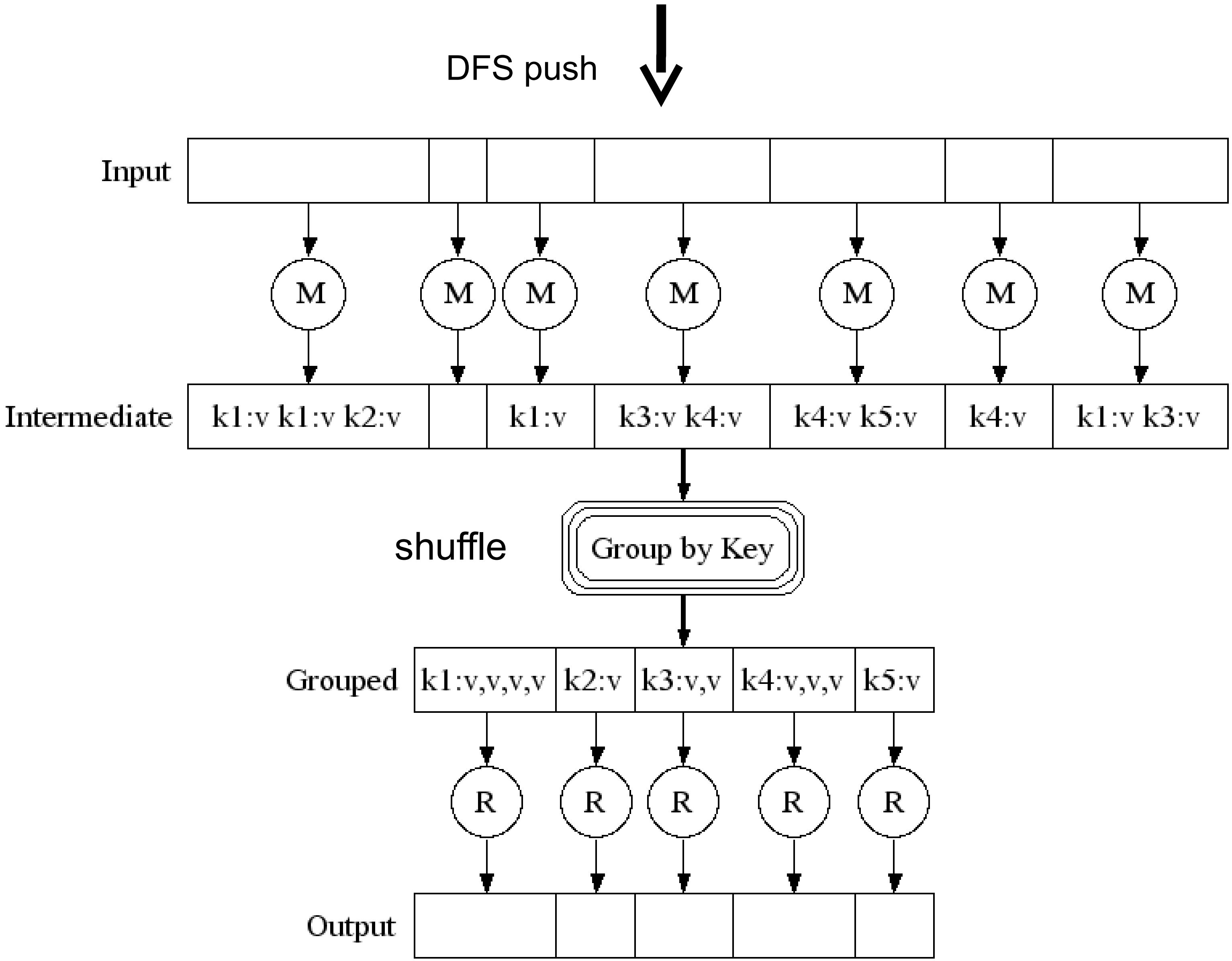
Big Data Trend: MapReduce

- 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

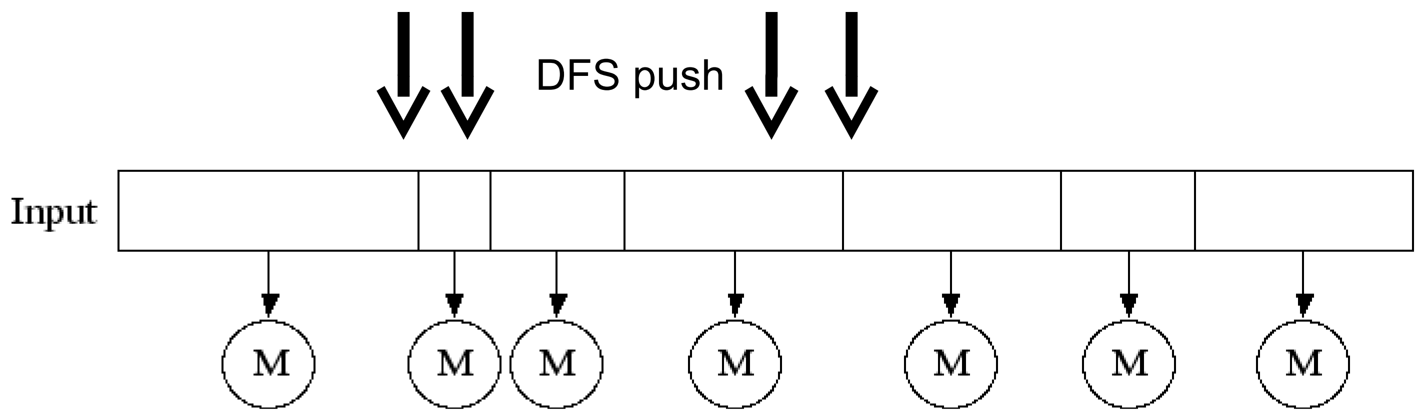
- MapReduce cluster
  - set of nodes N that run MapReduce job
  - specify number of mappers, reducers, <= N
  - master-worker paradigm
- Data set is first injected into DFS
- Data set is chunked (64 MB), replicated three times to the local disks of machines
- Master scheduler tries to run map jobs and reduce jobs on workers near the data

MapReduce Workflow

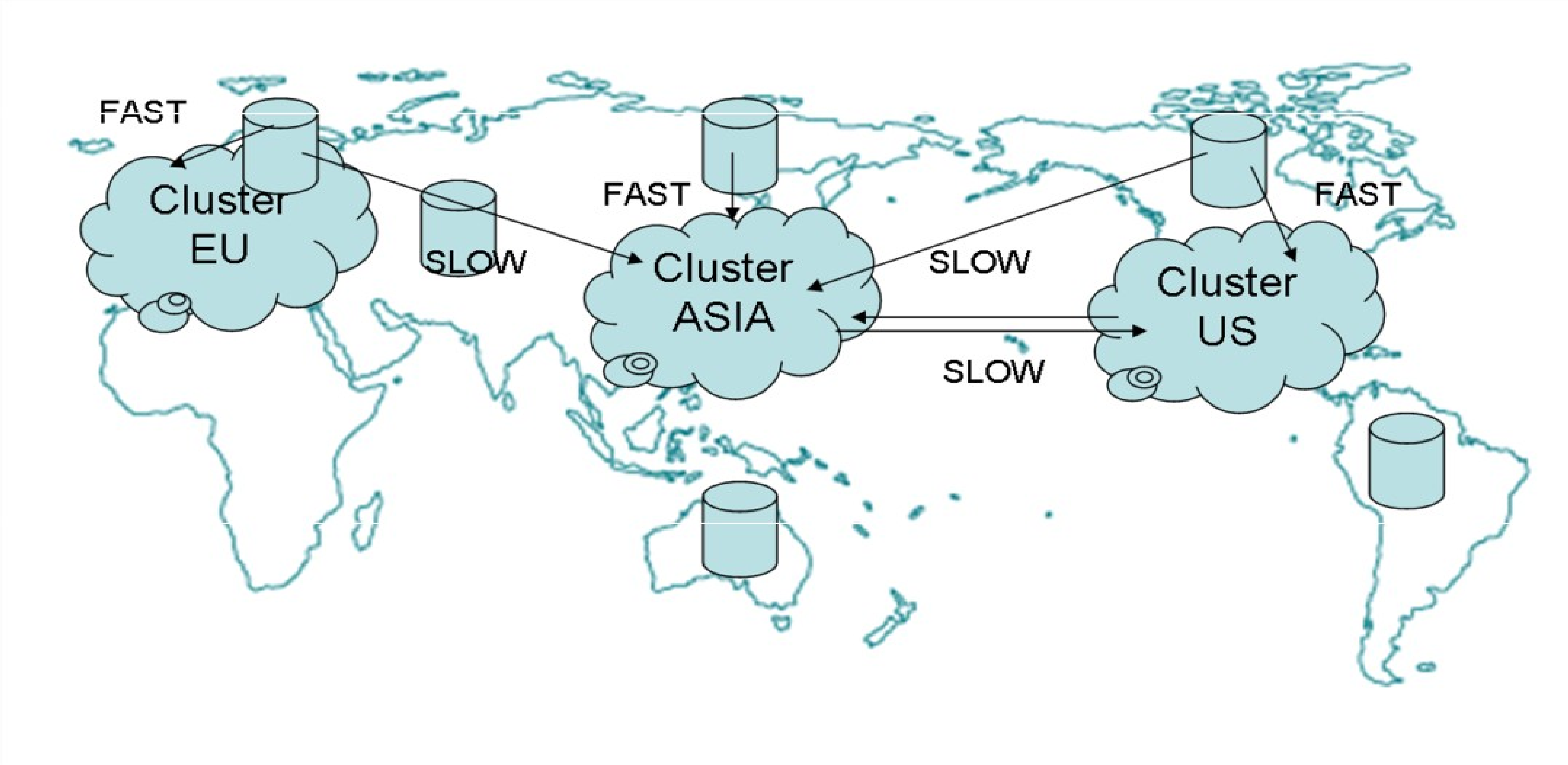


Big Data Trend: Distribution

- 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

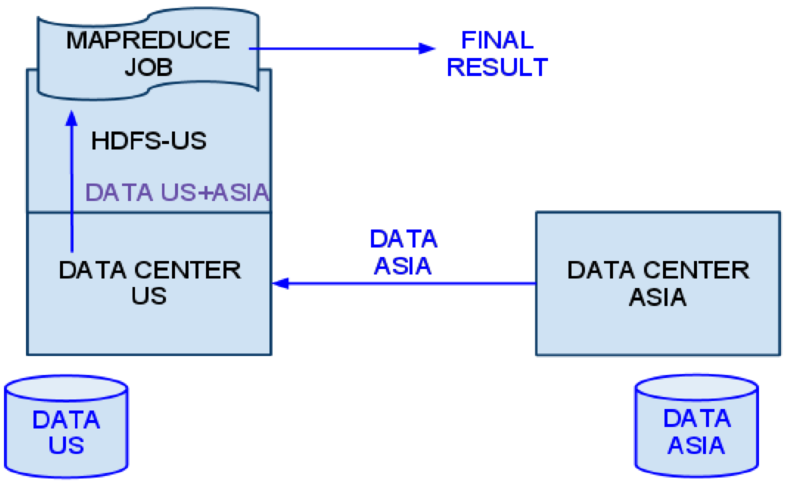


Context: Widely distributed data

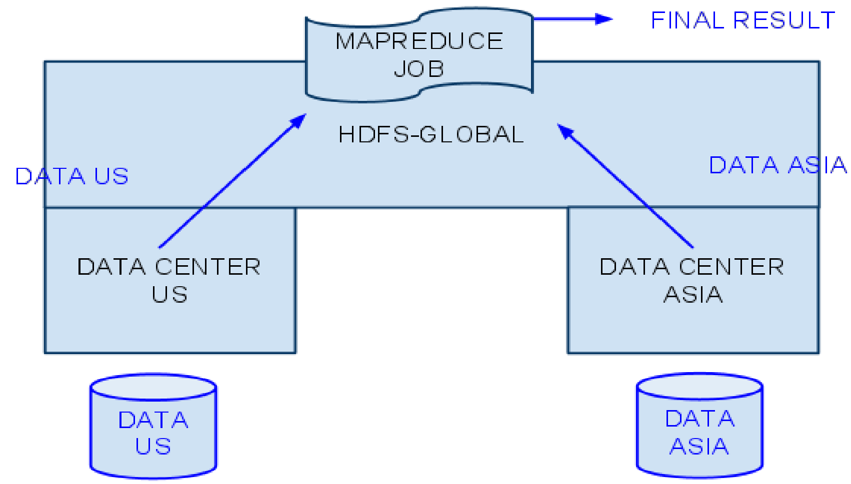


Data in different data-centers
Run MapReduce across them
Data-flow spanning wide-area networks

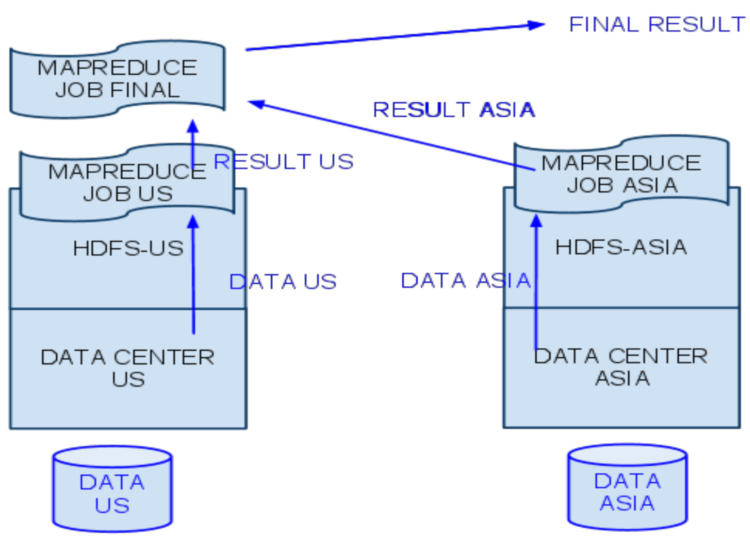
Data Scheduling: Wide-Area MapReduce



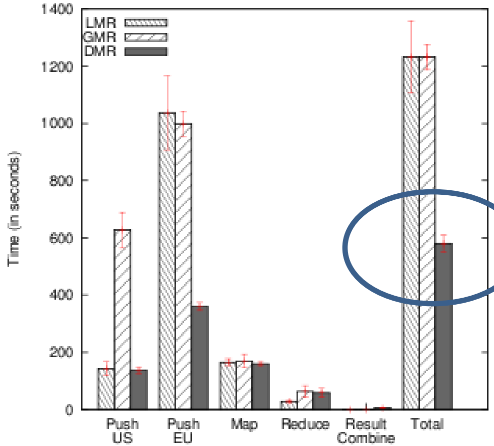
Local MapReduce (**LMR**)



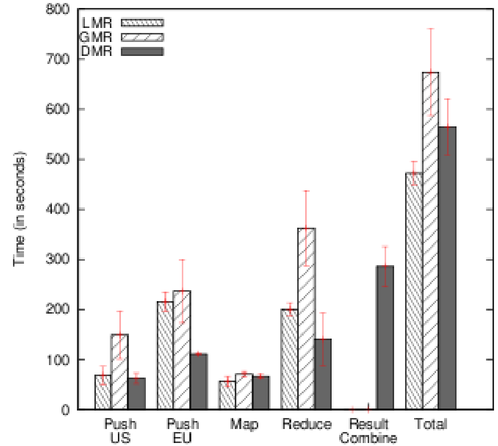
Global MapReduce (**GMR**)



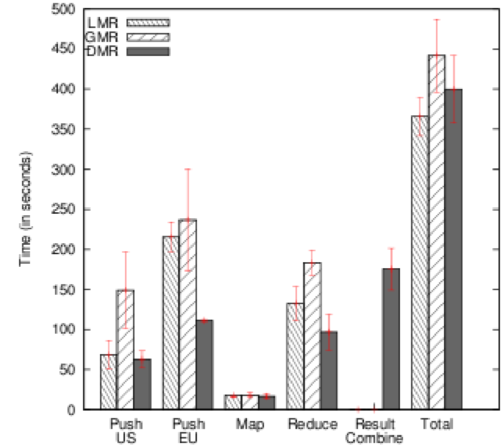
Distributed MapReduce (**DMR**)



(a) Wordcount on 800MB plain-text data



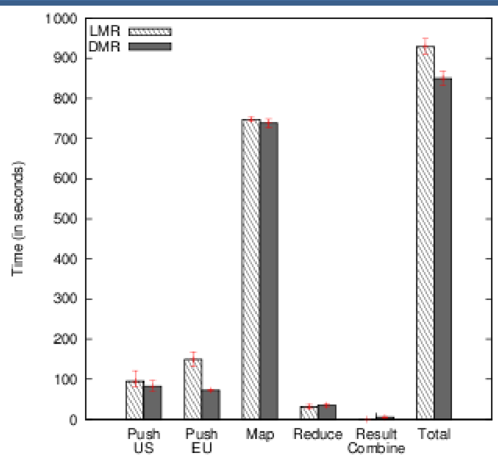
(b) Wordcount on 250MB random data



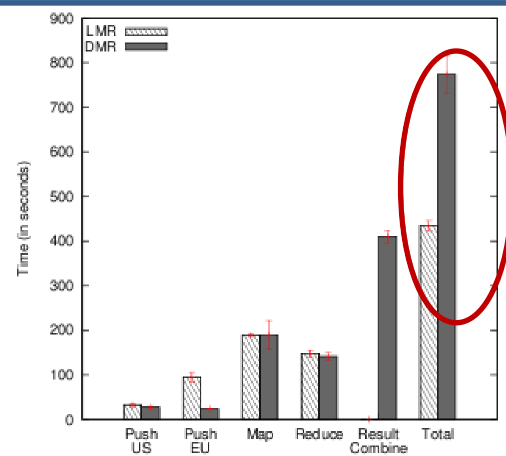
(c) Sort on 250MB random data

PlanetLab

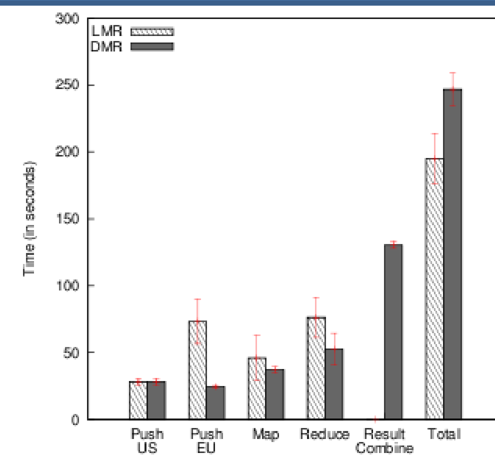
Amazon EC-2



(a) Wordcount on 3.2GB of plain-text data



(b) Wordcount on 1GB of random data



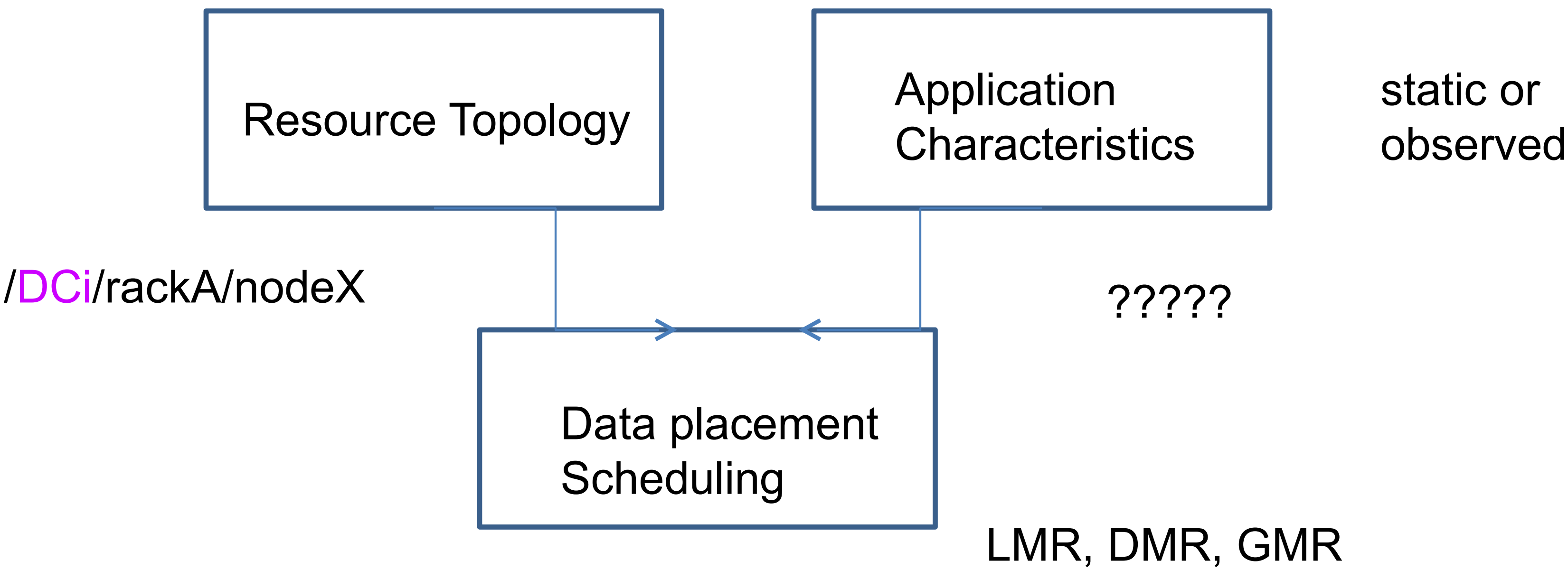
(c) Sort on 1GB of random data

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



Problem: Data Scheduling

- Data movement is dominant
- Data sets located in domains, size: Di, … Dm
- Platform domains: Pj, … Pk
- Inter-platform bandwidth: BDiPj
- Data expansion factors
  - input->intermediate, α
  - Intermediate->output, β

=> 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)