RAD Lab
UC Berkeley

The Potential of Cloud Computing: Challenges, Opportunities, Impact

Tim Kraska, UC Berkeley
Reliable Adaptive Distributed Systems Laboratory

Image: John Curley http://www.flickr.com/photos/jay_que/1834540/


[Anology from IM 2/09 / Daniel Abadi]

Do you want milk?

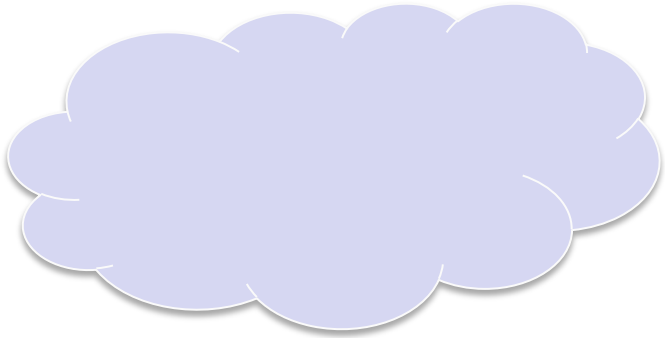
Buy a cow

- High upfront investment
- High maintenance cost
- Produces a fixed amount of milk
- Stepwise scaling

Buy bottled milk

- Pay-per-use
- Lower maintenance cost
- Linear scaling
- Fault-tolerant

Traditional DB vs. Cloud DB



Traditional DB

- High upfront investment
- Fixed amount of TRX
- Stepwise scaling
- High maintenance cost

Cloud

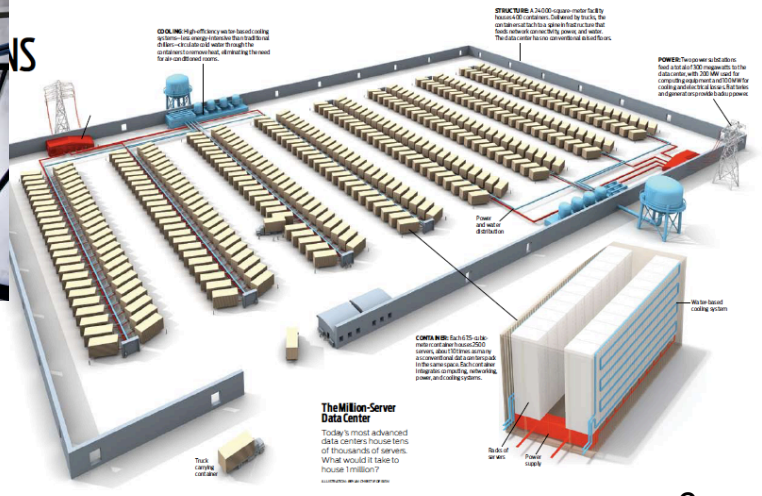
- Pay-per-use
- Linear scaling
- Fault-tolerant
- Lower maintenance cost

Get Your Own Supercomputer

- 41.88 Teraflops on LINPACK benchmark
  - Faster than #145 entry on current Top500 list
  - 880 EC cluster nodes
- No batch queues—start your job in minutes
- No grantwriting—just a credit card ($1400/hour, 1 hour minimum purchase)
- No need to predict usage in advance—pay only for what you use, get more on demand
- Lease several simultaneously
- MPI and Matlab available, among other packages

Warehouse Scale Computers

- Built to support consumer demand of Web services (email, social networking, etc.)
  - "Private clouds" of Google, Microsoft, Amazon, ...
- "Warehouse scale" buying power = 5-7x cheaper hardware, networking, administration cost



photos: Cnet News, Sun Microsystems, datacenterknowledge.com

2007: Public Cloud Computing Arrives

Type & Price/Hour	1GHz core eqv.	RAM	Disk and/or I/O
Small - $0.085	1	1.7 GB	160 GB
Large - $0.34	4	7.5 GB	850 GB, 2 spindles
XLarge - $0.68	8	15 GB	1690 GB, 3 spindles
Cluster ("HPC") - $1.60	32*	23 GB	1690 GB + 10Gig Ethernet

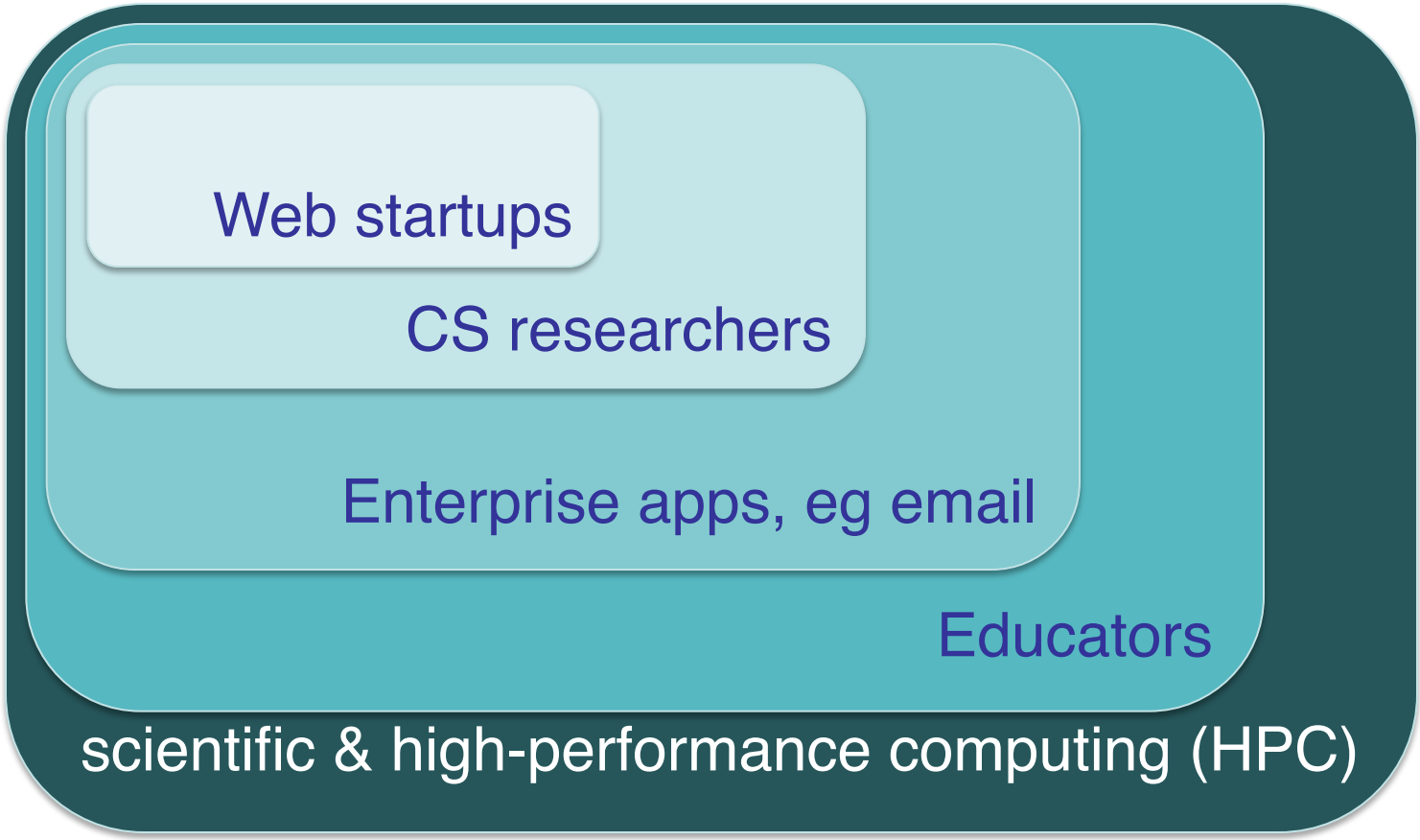
- Unlike Grid, computing on-demand *for the public*
- Virtual machines from $0.085 to $1.60/hr
  - Pay as you go with credit card, 1 hr. minimum
  - Cheaper if willing to share or risk getting kicked off
  - Machines provisioned & booted in a few minutes

1,000 machines for 1 hr = 1 machine × 1,000 hrs

* 2x quad-core Intel Xeon (Nehalem)

What's In It For You?

Cloud Computing progress 2008-2010



What's in it for you?

- **What:** Low cost + on-demand + cost-associativity = Democratization of Supercomputing
- **How:** Sophisticated software & operational expertise mask unreliability of commodity components
- Example: **MapReduce**
  - LISP language construct (~1960) for elegantly expressing *data parallel* operations w/sequential code
  - MapReduce (Google) and Hadoop (open source) provide failure masking & resource management for performing this on commodity clusters

"Warehouse scale" software engineering issues ***hidden*** *from application programmer*

Map/Reduce in a Nutshell

- Given:
  - a (very large) dataset
  - a well-defined computation task to be performed on elements of this dataset (preferably, in a parallel fashion on a large cluster)

- MapReduce programming model/abstraction/framework:
  - Just express what you want to compute (map() & reduce())
  - Don't worry about parallelization, fault tolerance, data distribution, load balancing (MapReduce takes care of these)
  - What changes from one application to another is the actual computation; the programming structure stays similar

[OSDI 04]

Map/Reduce in a Nutshell

- Here is the framework in simple terms:
  - Read lots of data
  - **Map**: extract something that you care about from each record (similar to map in functional programming)
  - Shuffle and sort
  - **Reduce**: aggregate, summarize, filter, or transform (similar to fold in functional programming)
  - Write the results
- One can use as many Maps and Reduces as required to model a given problem

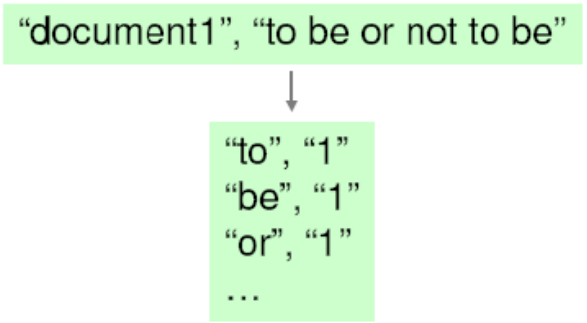
[OSDI 04]

Map/Reduce Example:
Count Word Occurrences in a Document

map in FP

map(k1, v1) → list(k2, v2)

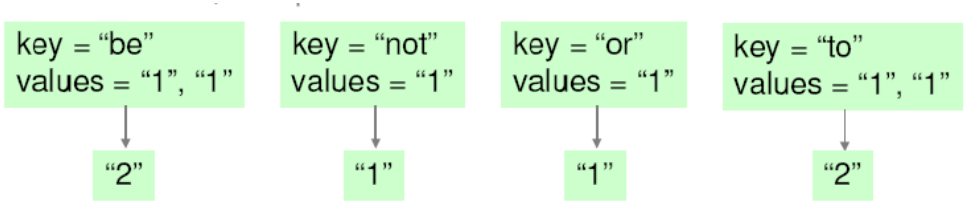
map (String key, String value):
    // key: document name
    // value: document contents
    for each word w in value:
        EmitIntermediate(w, "1");



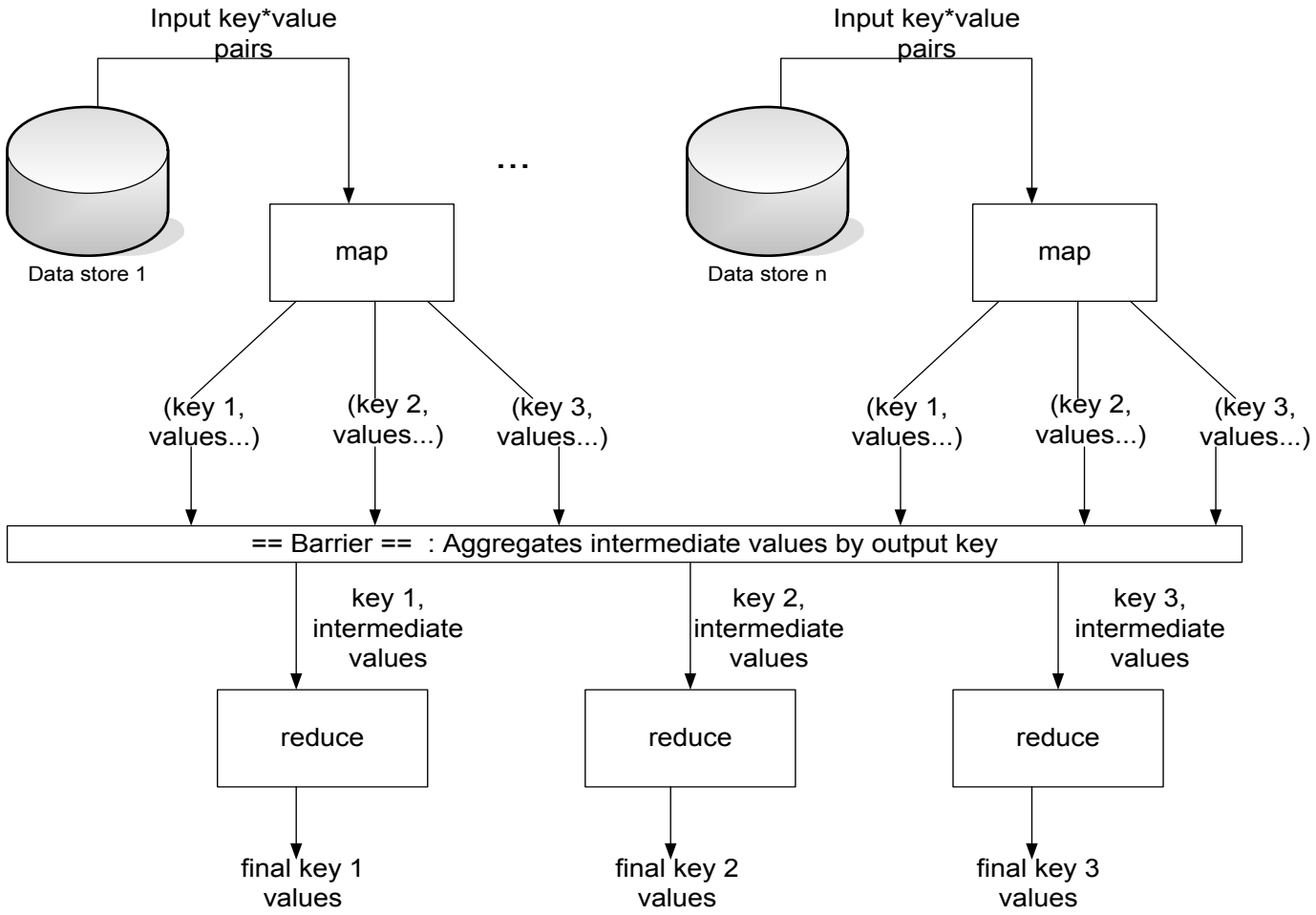
fold in FP

reduce(k2, list(v2)) → list(v2)

reduce (String key, Iterator values):
    // key: a word
    // values: a list of counts
    int result = 0;
    for each v in values:
        result += ParseInt(v);
    Emit(AsString(result));

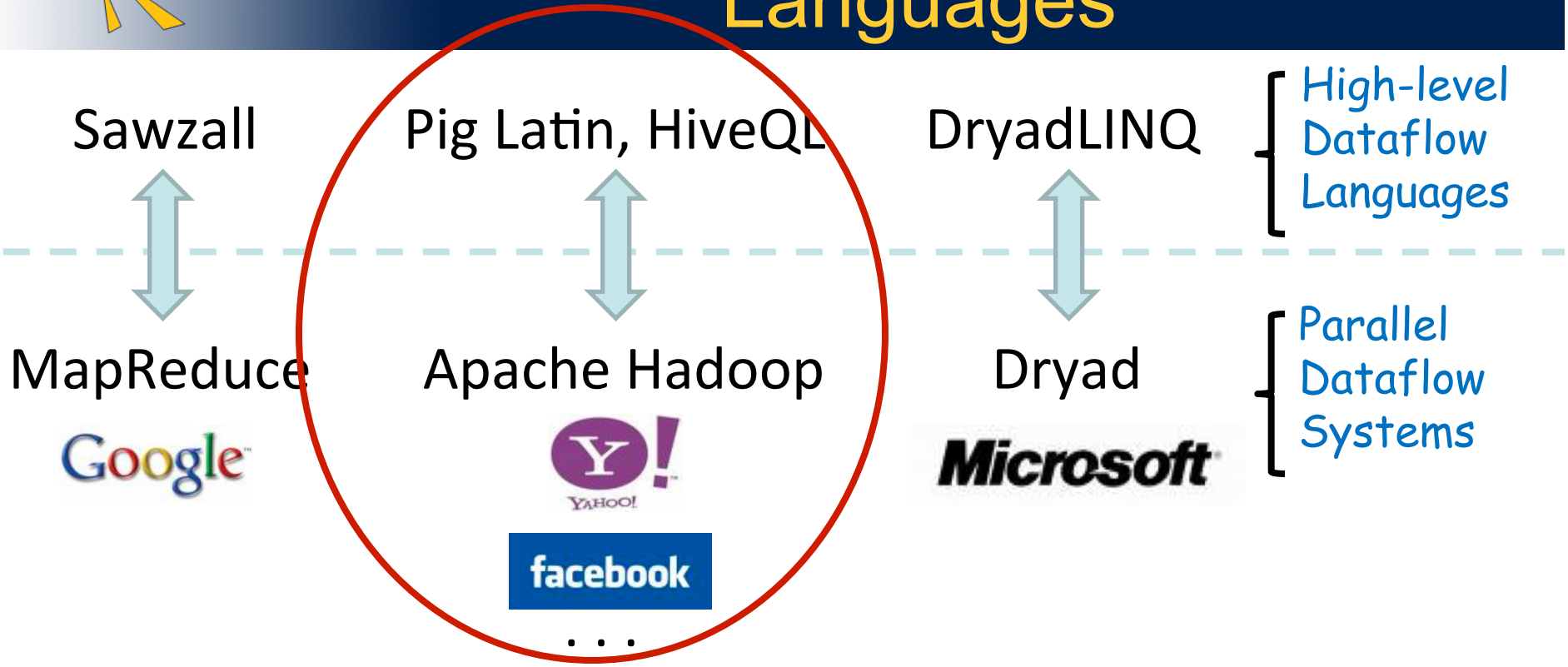


Google's MapReduce Model



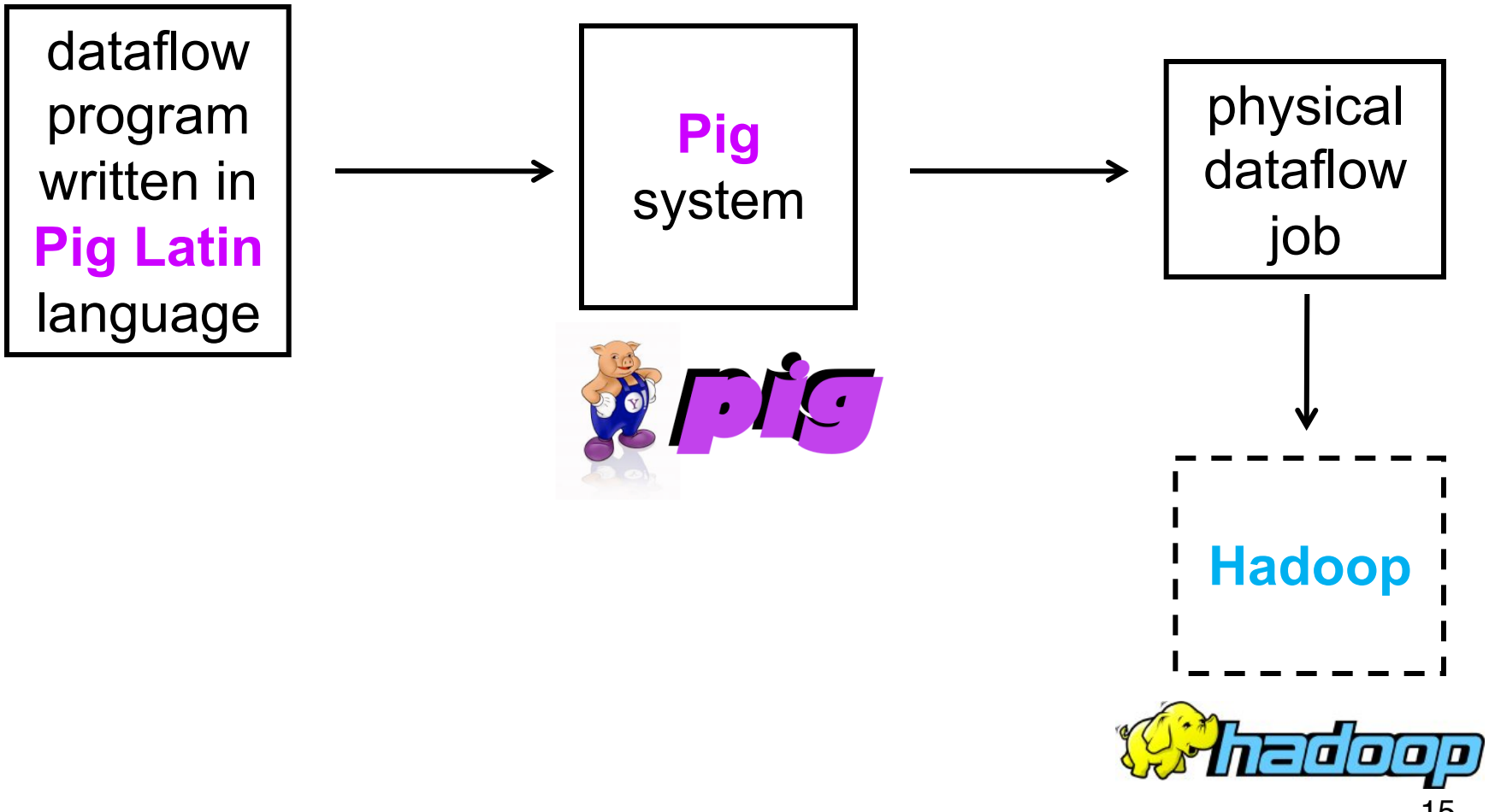
[OSDI 04]

Parallel Dataflow Systems and Languages



- A high-level language provides:
  - more transparent program structure
  - easier program development and maintenance
  - automatic optimization opportunities

Yahoo! Pig & Pig Latin



Example: Web Data Analysis

Find the top 10 most visited pages in each category

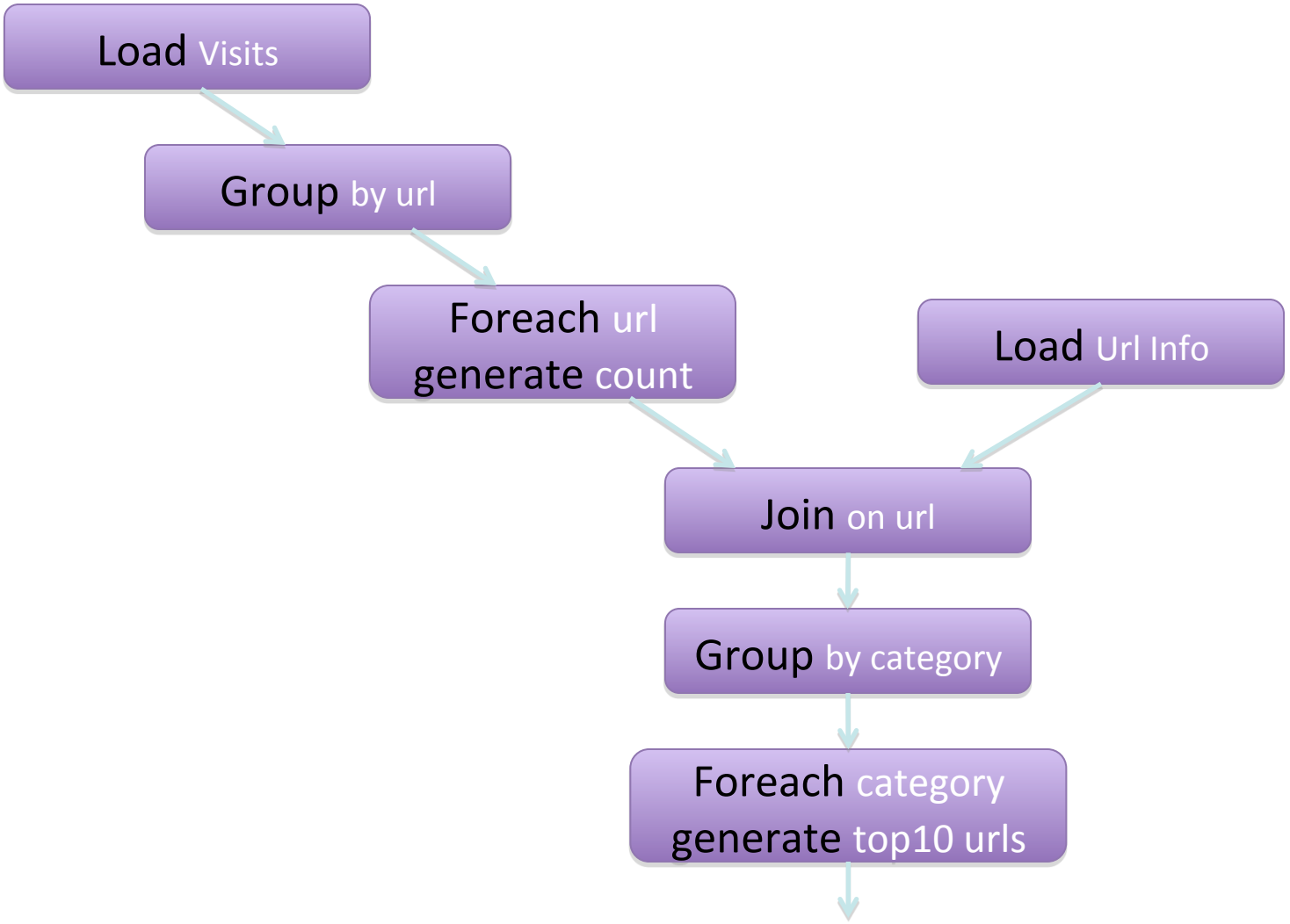
Visits

User	Url	Time
Amy	cnn.com	8:00
Amy	bbc.com	10:00
Amy	flickr.com	10:05
Fred	cnn.com	12:00

Url Info

Url	Category	PageRank
cnn.com	News	0.9
bbc.com	News	0.8
flickr.com	Photos	0.7
espn.com	Sports	0.9

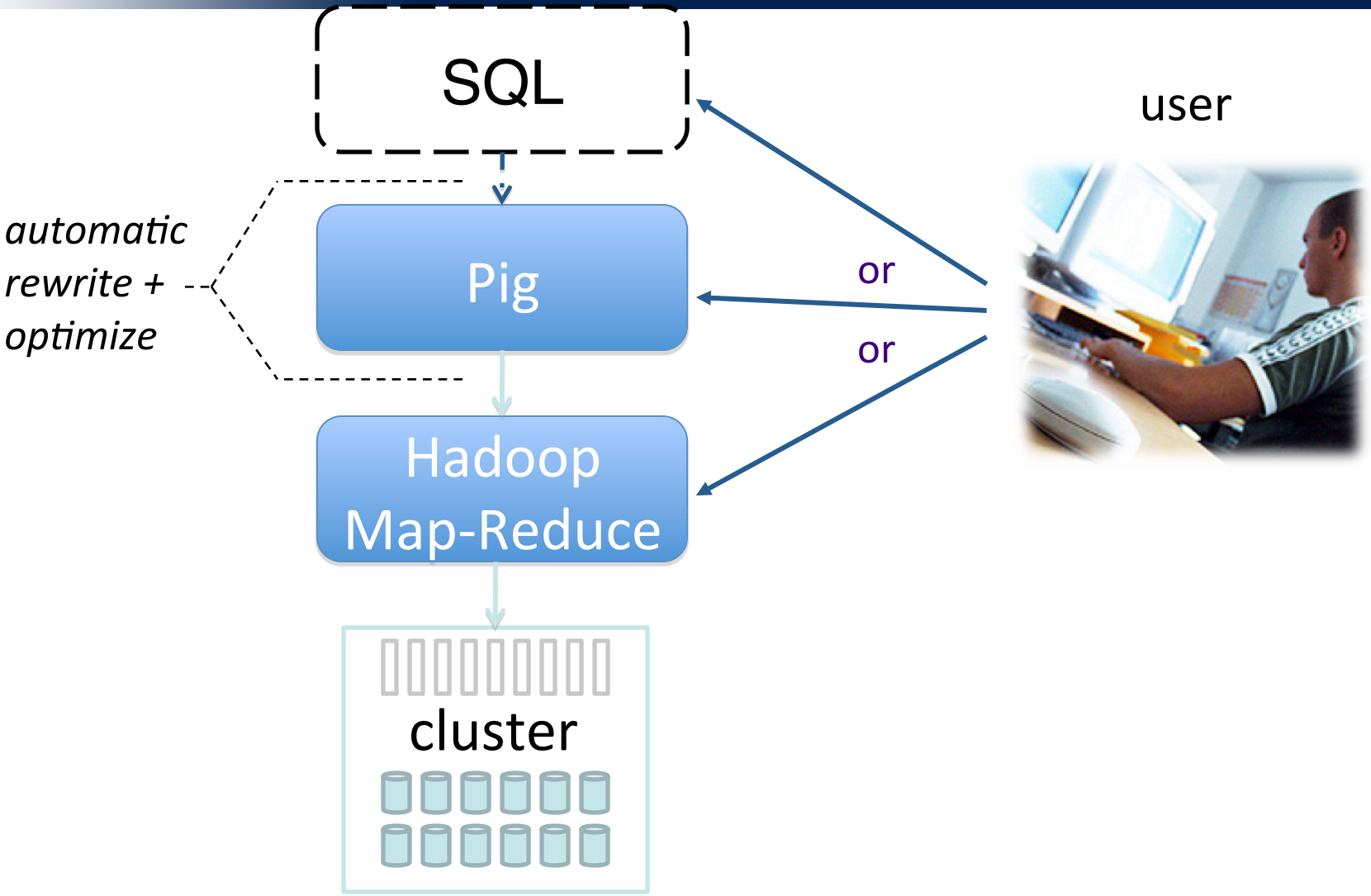
Example: Data Flow Diagram



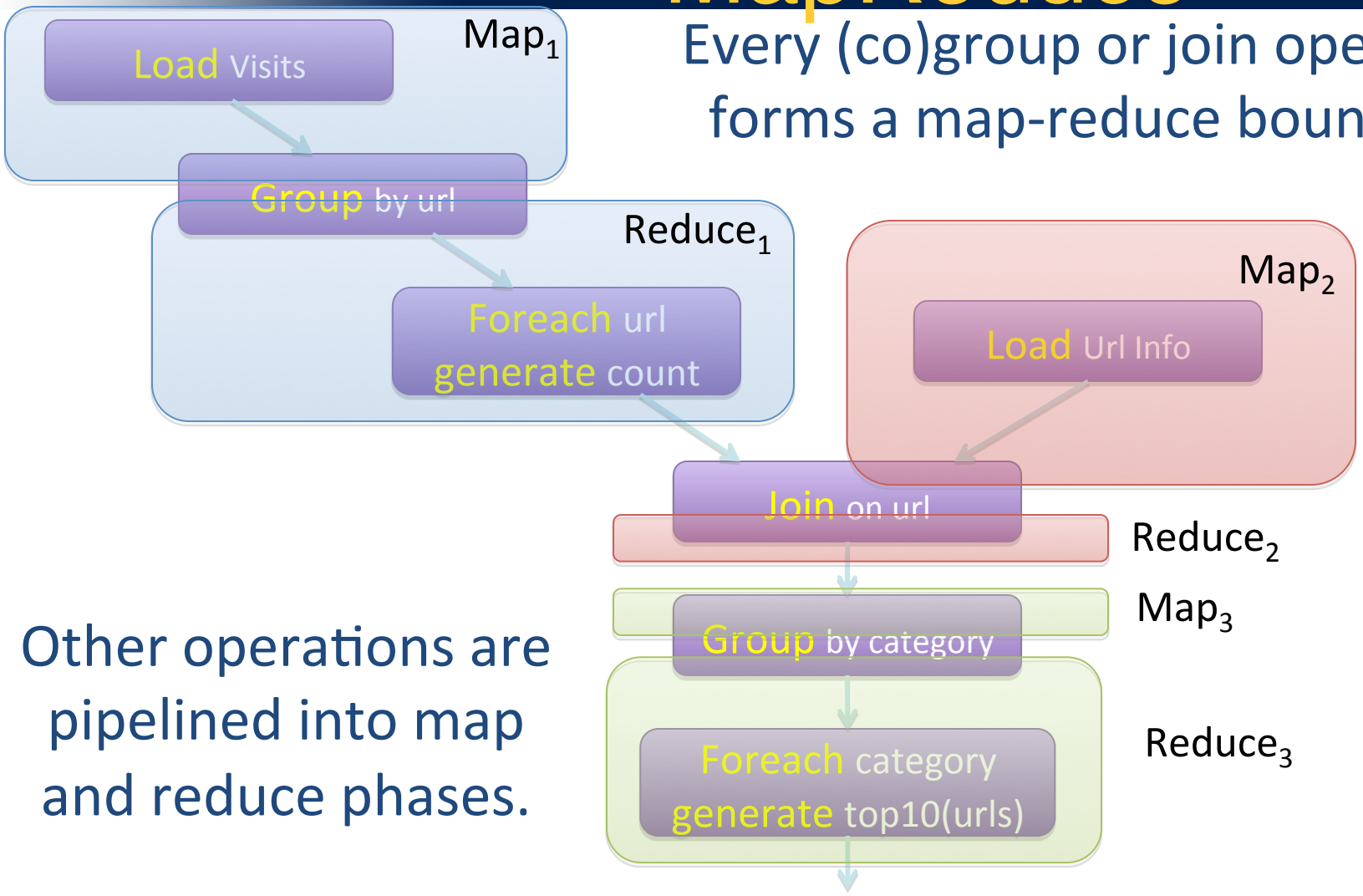
Example in Pig Latin

```
visits       = load '/data/visits' as (user, url, time);
gVisits      = group visits by url;
visitCounts  = foreach gVisits generate url, count(visits);

urlInfo      = load '/data/urlInfo' as (url, category, pRank);
visitCounts  = join visitCounts by url, urlInfo by url;

gCategories = group visitCounts by category;
topUrls      = foreach gCategories generate top
   (visitCounts,10);

store topUrls into '/data/topUrls';
```

Pig System Overview



Compilation into MapReduce



Every (co)group or join operation forms a map-reduce boundary.

Other operations are pipelined into map and reduce phases.

MapReduce in Practice

- Example: spam classification
  - training: 10^7 URLs x 64KB data each = 640GB data
  - One heavy-duty server: ~270 hours
  - 100 servers in cloud: ~3 hours (= ~$255)
- Rapid uptake in other scientific research
  - Large-population genetic risk analysis & simulation (Harvard Medical School)
  - Genome sequencing (UNC Chapel Hill Cancer Ctr)
  - Information retrieval research (U. Glasgow – Terrier)
  - Compact Muon Solenoid Expt. (U. Nebraska Lincoln)

Challenge: Cloud Programming

- Challenge: exposing parallelism
  - MapReduce relies on "embarrassing parallelism"
- Programmers must (re)write problems to expose this parallelism, if it's there to be found
- Tools still primitive, though progressing rapidly
  - MapReduce—early success story
  - Pig (Yahoo! Research), Hive (Apache Foundation)
  - Pregel → New framework for graph computatoins
  - Mesos—share cluster among Hadoop, MPI, interactive jobs (UC Berkeley)
- Challenge for tool authors: parallel software hard to debug and operate reliably (YY Zhou)

Challenge: Big Data

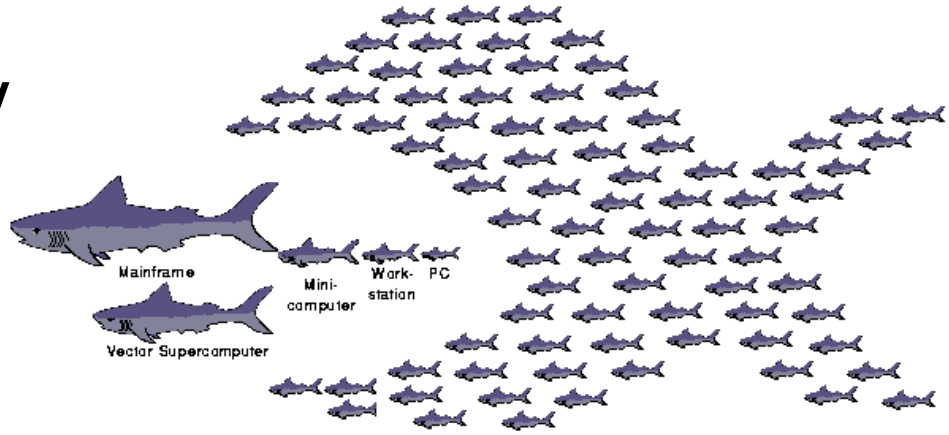
Application	Data generated per day
DNA Sequencing (Illumina HiSeq machine)	1 TB
Large Synoptic Survey Telescope	30 TB; 400 Mbps sustained data rate between Chile and NCSA
Large Hadron Collider	60 TB

- Challenge: Long-haul networking is most expensive cloud resource, and improving most slowly
- Copy 8 TB to Amazon over ~20 Mbps network
  => ~35 days, ~$800 in transfer fees
- How about shipping 8TB drive to Amazon instead?
  => 1 day, ~$150 (shipping + transfer fees)

Source: Ed Lazowska, eScience 2010, Microsoft Cloud Futures Workshop, lazowska.cs.washington.edu/cloud2010.pdf

An Analogy: Clusters of Commodity PC's, c.1995

- Clusters of Commodity PC's vs. symmetric multiprocessors
- *Potential* advantages: incremental scaling, absolute capacity, economy of using commodity HW & SW
- 1995: SaaS on SMPs; software architecture SMP-centric
- 2010: Berkeley undergrads prototype SaaS in 6 8 weeks and deploy on cloud computing



Conclusion

- Democratization of supercomputing capability
  - Time to answer may be faster even if hardware isn't
  - Writing a grant proposal around a supercomputer?
- Software & hardware rapidly improving as vendors listen to scientific/HPC customers
- HPC opportunity to influence design of *commodity* equipment

Impact: Potential democratizing effect comparable to microprocessor

Acknowledgments

- Colleagues at RAD Lab, UC Berkeley and Systems Group, ETH Zurich
- Colleagues at our industrial collaborators

Thank you!