Operating Systems and The Cloud

David E. Culler

CS162 – Operating Systems and Systems Programming

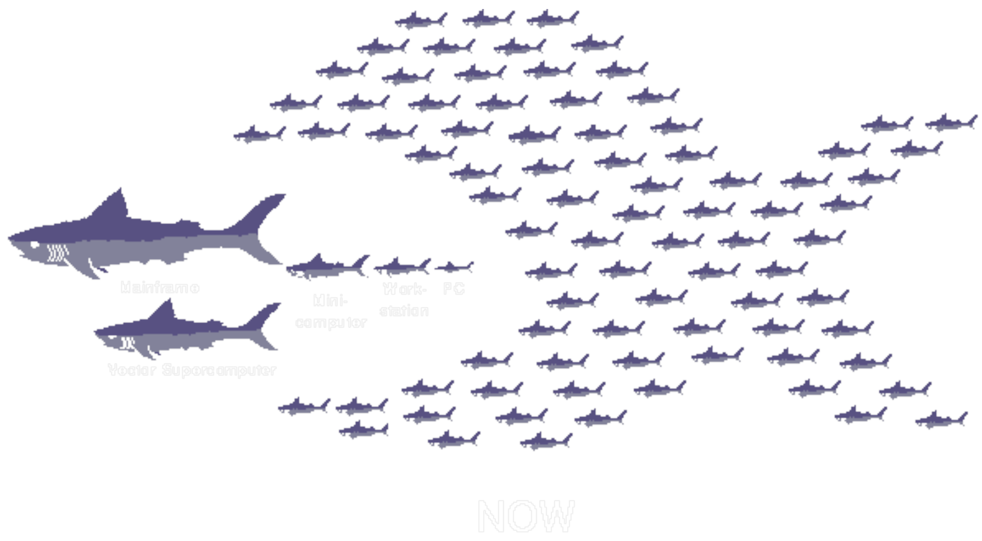
Lecture 39

December 1, 2014

Proj: CP 2 12/3

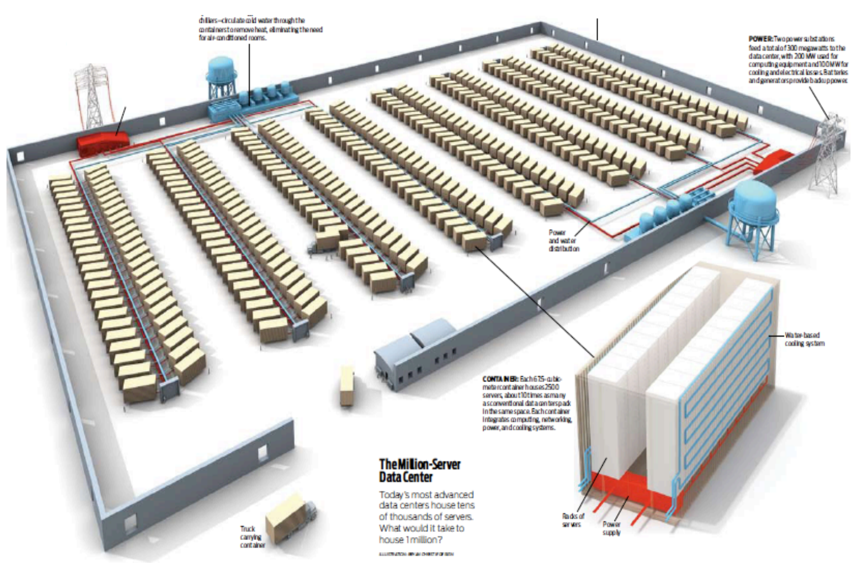
Goals Today

- Give you a sense of kind of operating systems issues that arise in The Cloud
- Encourage you to think about graduate studies and creating what is out beyond what you see around you …



The Datacenter is the new Computer ??

- “The datacenter as a computer” is still young
  - Complete systems as building blocks (PC+Unix+HTTP+SQL+ …)
  - Higher Level Systems formed as Clusters, e.g., Hadoop cluster
  - Scale => More reliable than its components
  - Innovation => Rapid (ease of) development, Predictable Behavior despite variations in demand, etc.



=?

Datacenter/Cloud Computing OS ???

- **If the datacenter/cloud is the new computer,**
- **what is its Operating System?**
  - **Not the host OS for the individual nodes, but for the millions of nodes that form the ensemble of quasi-distributed resources !**

- **Will it be as much of an enabler as the LAMP stack was to the .com boom ?**
- **Open source stack for every Web 2.0 company:**
  - **Linux OS**
  - **Apache web server**
  - **MySQL, MariaDB or MongoDB DBMS**
  - **PHP, Perl, or Python languages for dynamic web pages**

Classical Operating Systems

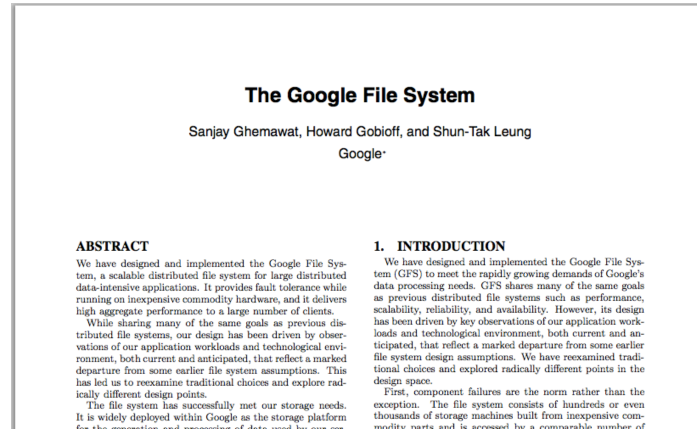
- **Data sharing**
  - Inter-Process Communication, RPC, files, pipes, …
- **Programming Abstractions**
  - Storage & I/O Resources, Libraries (libc), system calls, …
- **Multiplexing of resources**
  - Scheduling, virtual memory, file allocation/protection, …

Datacenter/Cloud Operating System

- Data sharing
  - Google File System, key/value stores
  - Apache project: Hadoop Distributed File System
- Programming Abstractions
  - Google MapReduce
  - Apache projects: Hadoop, Pig, Hive, Spark, …
  - Nyad, Driad, …
- Multiplexing of resources
  - Apache projects: Mesos, YARN (MapReduce v2), ZooKeeper, BookKeeper, …

Google Cloud Infrastructure

- **Google File System (GFS), 2003**
  - Distributed File System for entire cluster
  - Single namespace


The Google File System

Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung
Google*

ABSTRACT

We have designed and implemented the Google File System, a scalable distributed file system for large distributed data-intensive applications. It provides fault tolerance while running on inexpensive commodity hardware, and it delivers high aggregate performance to a large number of clients.

While sharing many of the same goals as previous distributed file systems, our design has been driven by observations of our application workloads and technological environment, both current and anticipated, that reflect a marked departure from some earlier file system assumptions. This has led us to reexamine traditional choices and explore radically different design points.

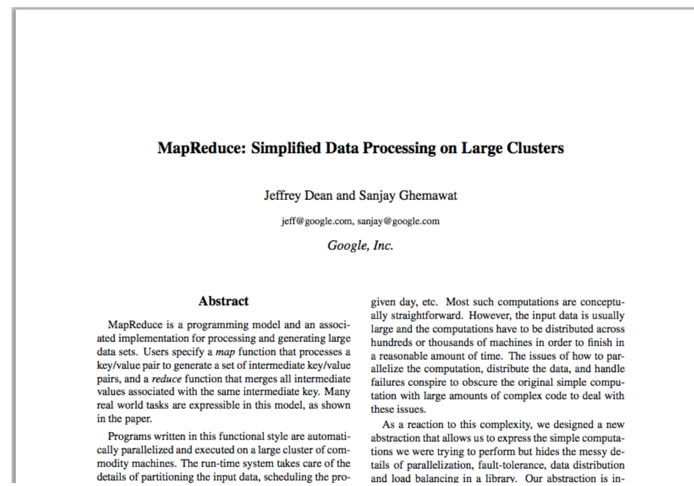
The file system has successfully met our storage needs. It is widely deployed within Google as the storage platform

1. INTRODUCTION

We have designed and implemented the Google File System (GFS) to meet the rapidly growing demands of Google's data processing needs. GFS shares many of the same goals as previous distributed file systems such as performance, scalability, reliability, and availability. However, its design has been driven by key observations of our application workloads and technological environment, both current and anticipated, that reflect a marked departure from some earlier file system design assumptions. We have reexamined traditional choices and explored radically different points in the design space.

First, component failures are the norm rather than the exception. The file system consists of hundreds or even thousands of storage machines built from inexpensive com-

- **Google MapReduce (MR), 2004**
  - Runs queries/jobs on data
  - Manages work distribution & fault-tolerance
  - Colocated with file system


MapReduce: Simplified Data Processing on Large Clusters

Jeffrey Dean and Sanjay Ghemawat

jeff@google.com, sanjay@google.com

Google, Inc.

Abstract

MapReduce is a programming model and an associated implementation for processing and generating large data sets. Users specify a *map* function that processes a key/value pair to generate a set of intermediate key/value pairs, and a *reduce* function that merges all intermediate values associated with the same intermediate key. Many real world tasks are expressible in this model, as shown in the paper.

Programs written in this functional style are automatically parallelized and executed on a large cluster of commodity machines. The run-time system takes care of the details of partitioning the input data, scheduling the pro-

given day, etc. Most such computations are conceptually straightforward. However, the input data is usually large and the computations have to be distributed across hundreds or thousands of machines in order to finish in a reasonable amount of time. The issues of how to parallelize the computation, distribute the data, and handle failures conspire to obscure the original simple computation with large amounts of complex code to deal with these issues.

As a reaction to this complexity, we designed a new abstraction that allows us to express the simple computations we were trying to perform but hides the messy details of parallelization, fault-tolerance, data distribution and load balancing in a library. Our abstraction is in-

- **Apache open source versions: Hadoop DFS and Hadoop MR**

GFS/HDFS Insights

- *Petabyte* storage
  - Files split into large blocks (128 MB) and replicated across many nodes
  - Big blocks allow high throughput sequential reads/writes
- Data *striped* on hundreds/thousands of servers
  - Scan 100 TB on 1 node @ 50 MB/s = 24 days
  - Scan on 1000-node cluster = 35 minutes
- *Failures* will be the norm
  - Mean time between failures for 1 node = 3 years
  - Mean time between failures for 1000 nodes = 1 day
- Use *commodity* hardware
  - Failures are the norm anyway, buy cheaper hardware
- No complicated consistency models
  - Single writer, append-only data

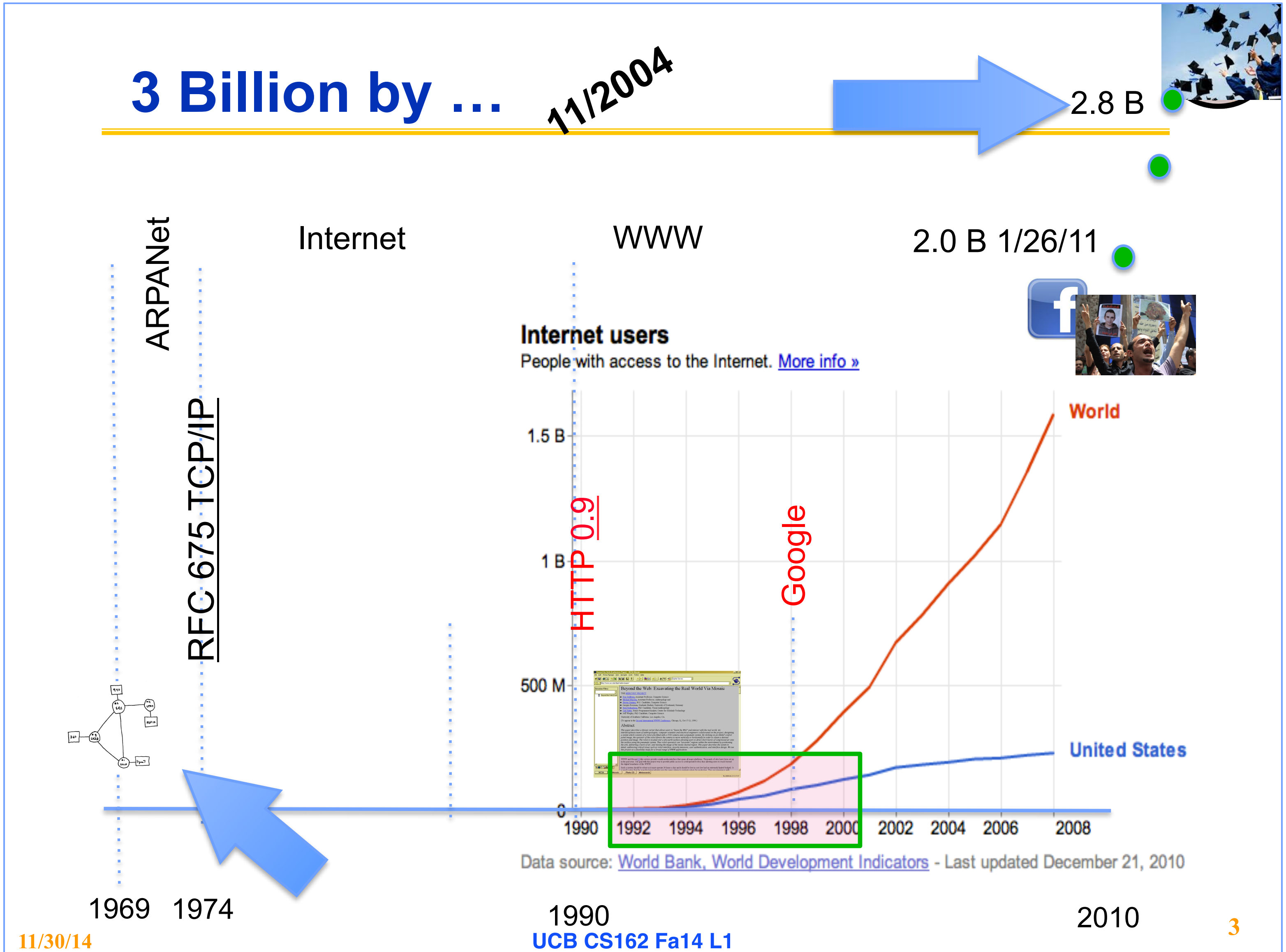
MapReduce Insights

- **Restricted key-value model**
  - Same fine-grained operation (Map & Reduce) repeated on huge, distributed (within DC) data
  - Operations must be deterministic
  - Operations must be idempotent/no side effects
  - Only communication is through the shuffle
  - Operation (Map & Reduce) output saved (on disk)

What is (was) MapReduce Used For?

- At Google:
  - Index building for Google Search
  - Article clustering for Google News
  - Statistical machine translation
  - …
- At Yahoo!:
  - Index building for Yahoo! Search
  - Spam detection for Yahoo! Mail
  - …
- At Facebook:
  - Data mining
  - Ad optimization
  - Spam detection
  - …

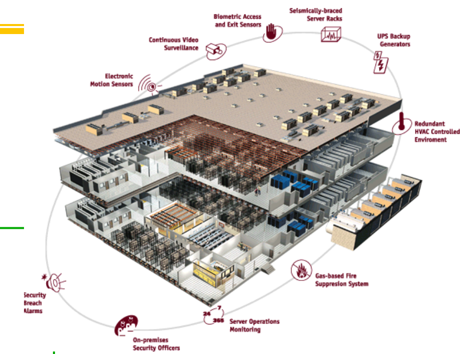
A Time-Travel Perspective



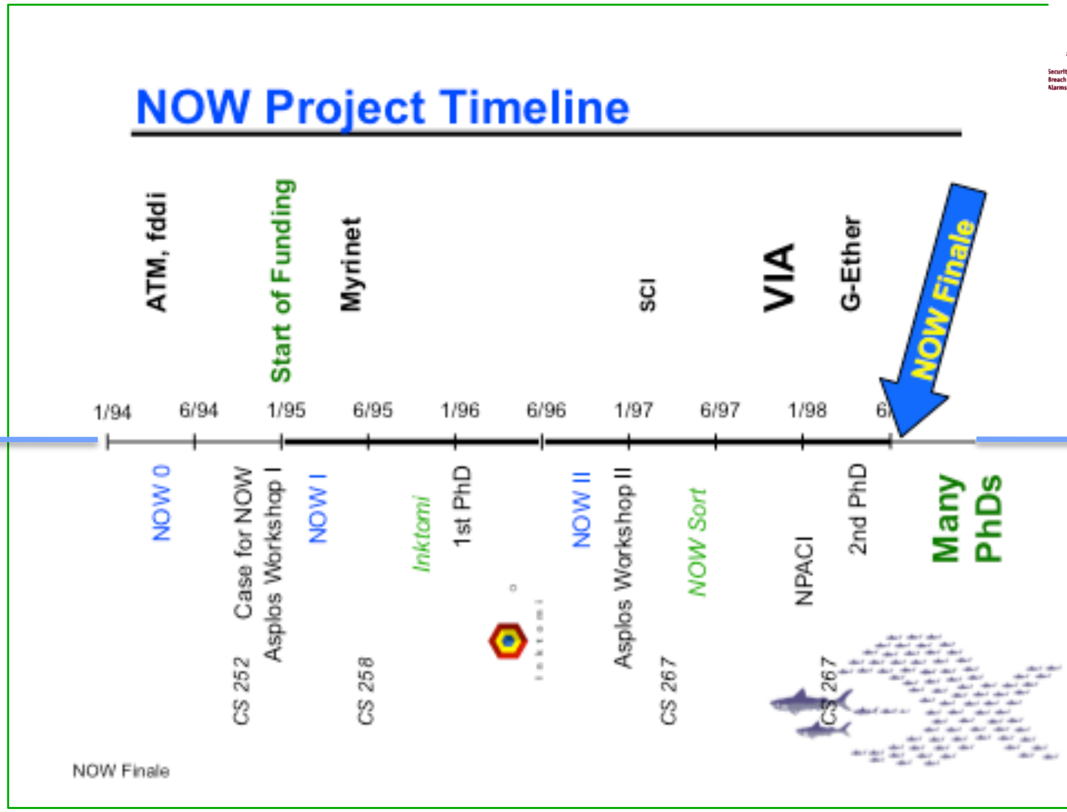
Research as “Time Travel”

- Imagine a technologically plausible future
- Create an approximation of that vision using technology that exists.
- Discover what is True in that world
  - Empirical experience
    - » Bashing your head, stubbing your toe, reaching epiphany
  - Quantitative measurement and analysis
  - Analytics and Foundations
- Courage to ‘break trail’ and discipline to do the hard science

NOW – Scalable Internet Service Cluster Design

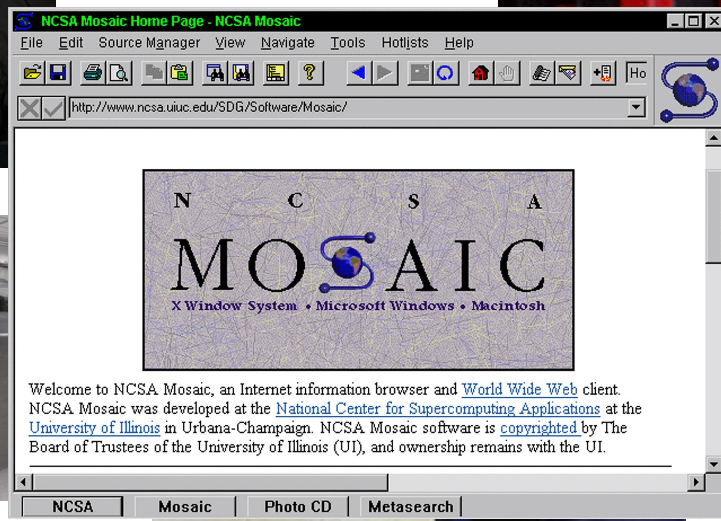


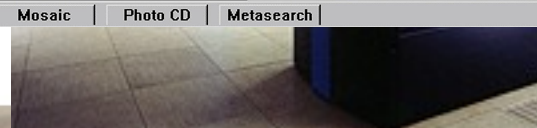
VAX clusters



Google

1993 Massively Parallel Processor is King



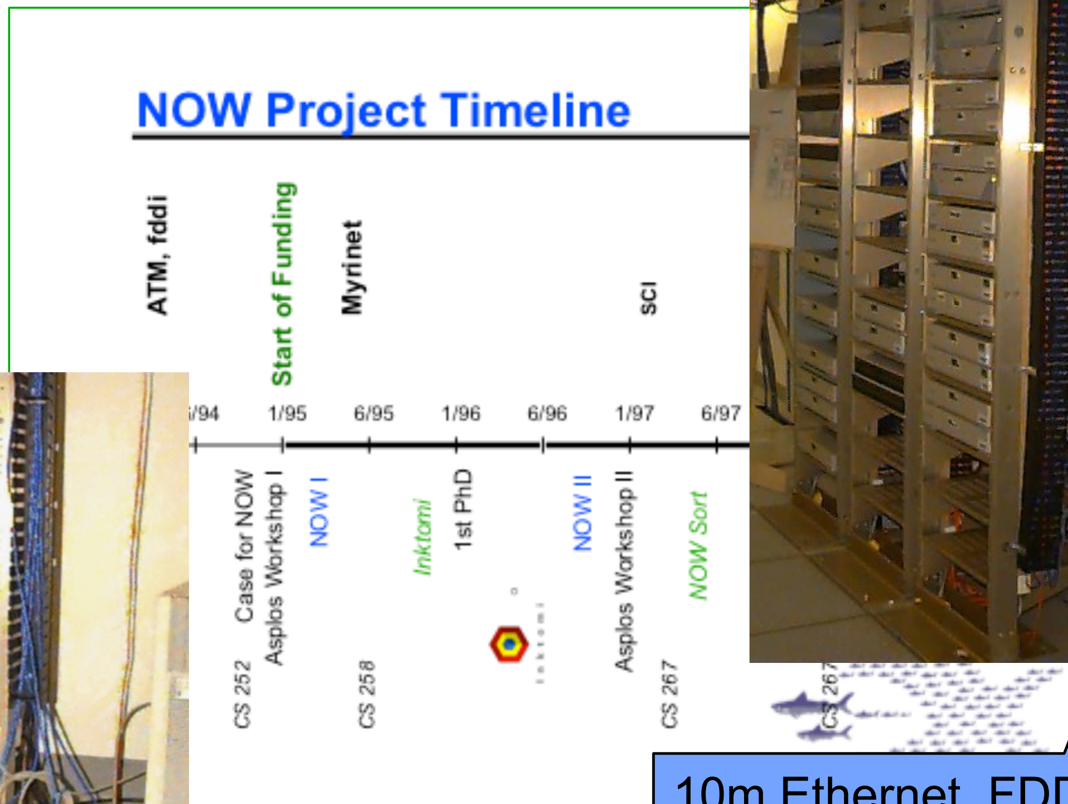


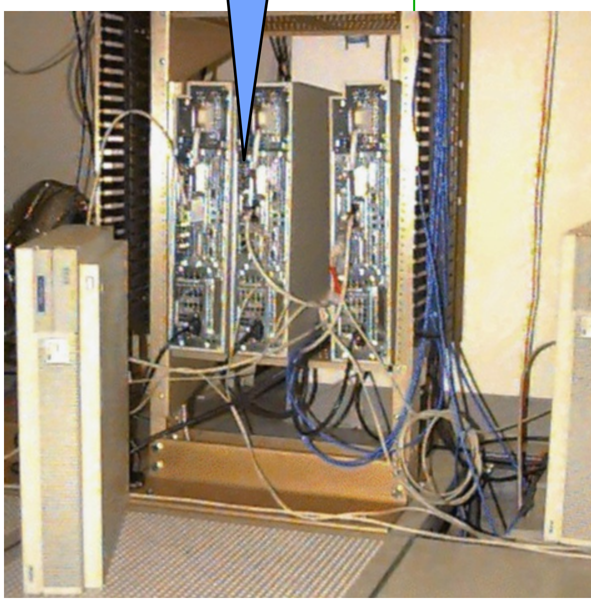
The Accelerated Strategic Computing Initiative Program includes participants from Los Alamos National Laboratory, Lawrence Livermore National Laboratory, and Sandia National Laboratories.

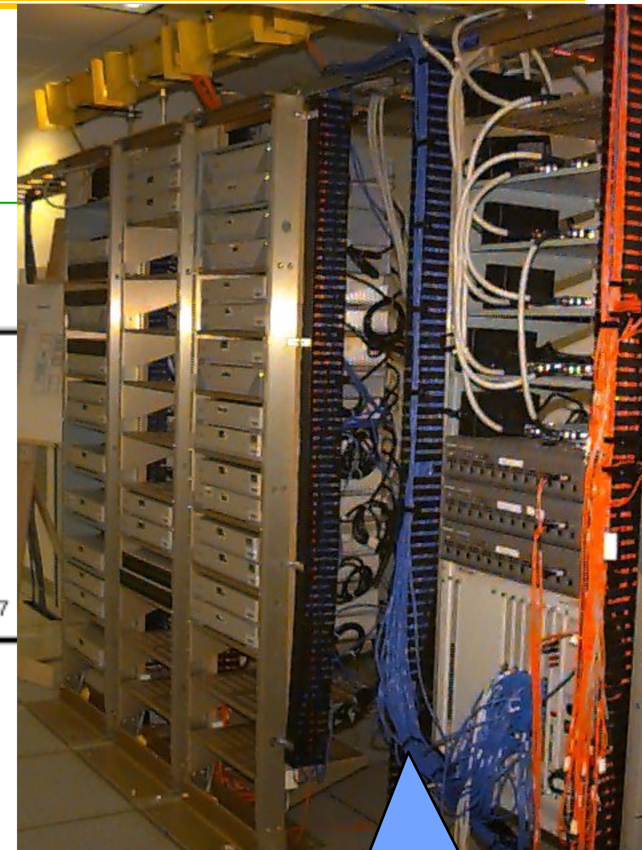
LALP-98-127

NOW – Scalable High Performance Clusters

GSC+ => PCI
=> ePCI …

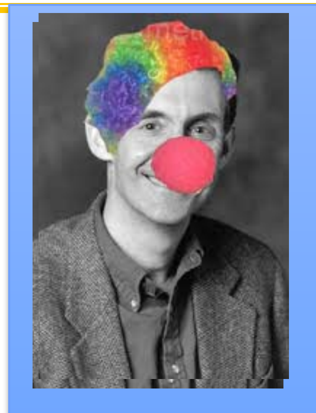


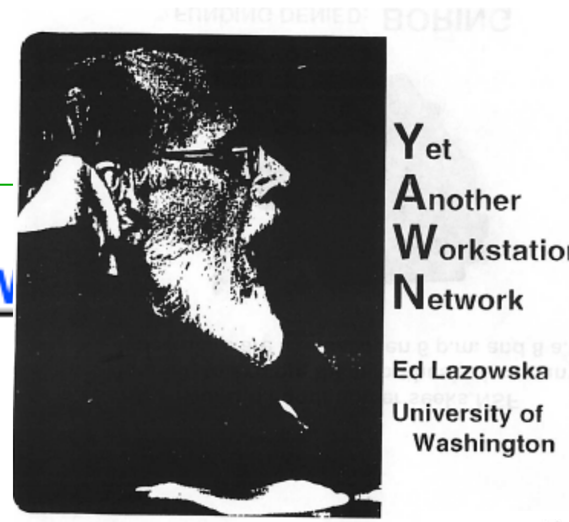


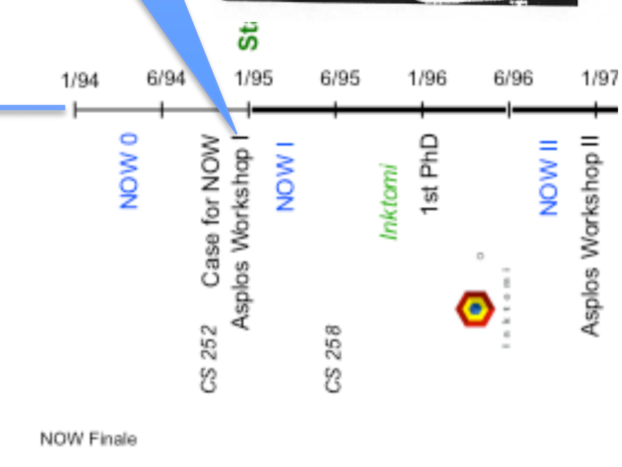


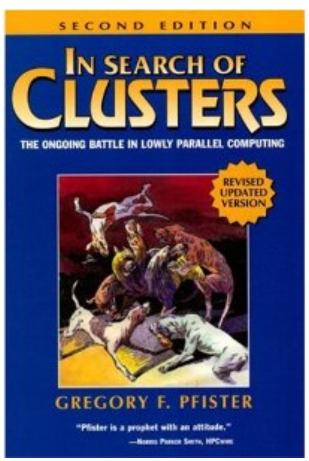
10m Ethernet, FDDI, ATM,
Myrinet, … VIA, Fast Ethernet,
=> infiniband, gigEtherNet

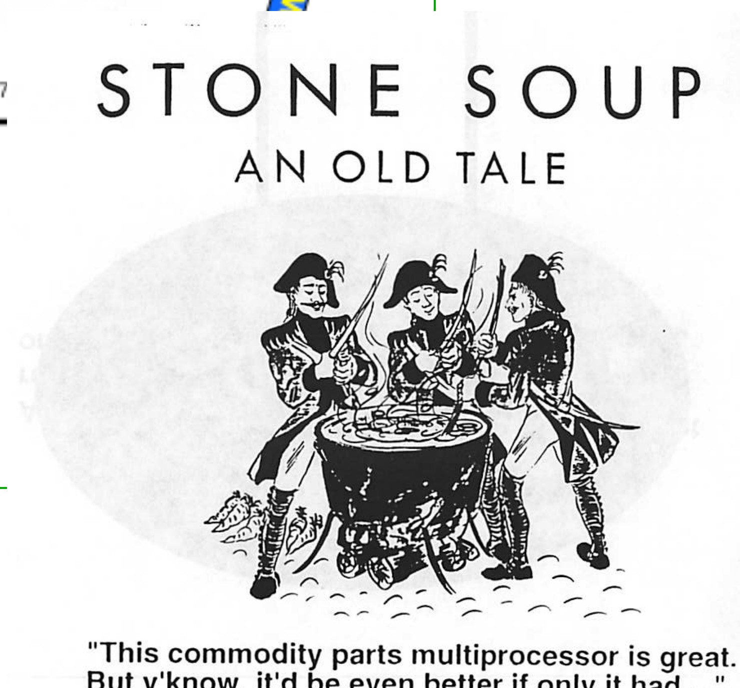
NOW – Scalable High Performance Clusters



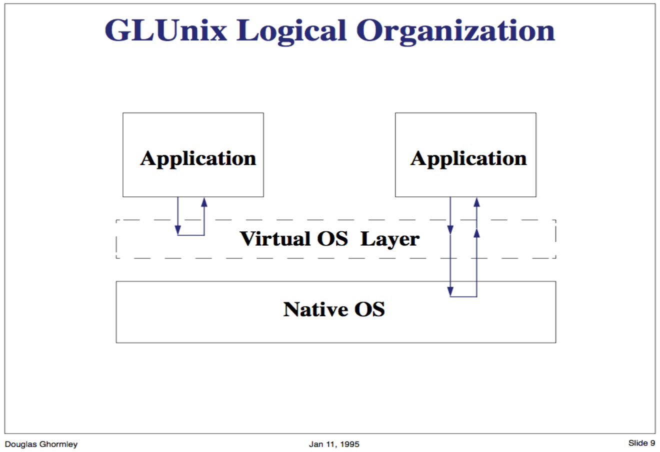


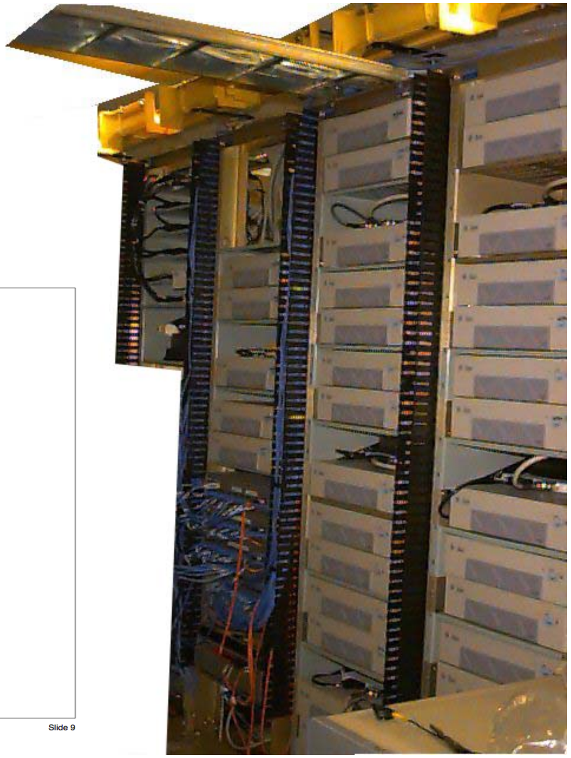






UltraSparc/Myrinet NOW

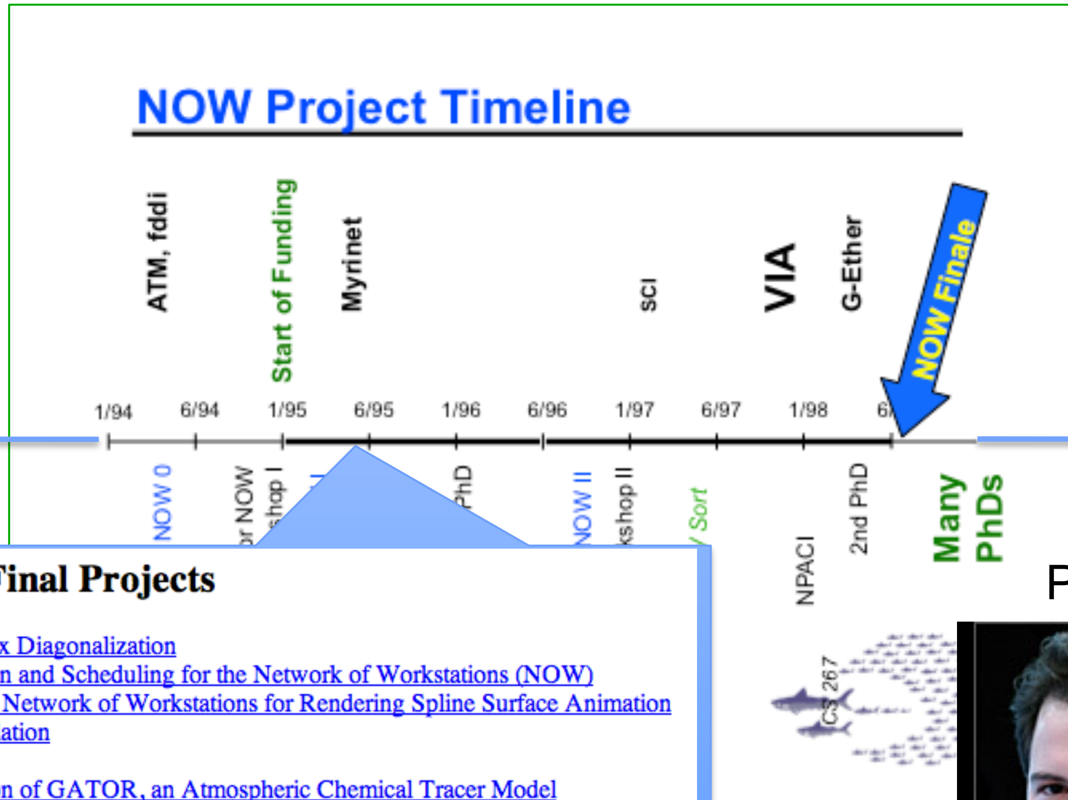




- Active Message: Ultra-fast user-level RPC
- When remote memory is closer than local disk …
- Global Layer system built over local systems
  - Remote (parallel) execution, Scheduling, Uniform Naming
  - xFS – cluster-wide p2p file system
  - Network Virtual Memory

Inktomi – Fast Massive Web Search
Fiat Lux - High Dynamic Range Imaging

Paul Gauthier



Lycos
infoseek

CS267, Spring 1995: Final Projects

- Fast Parallel Iterative Matrix Diagonalization
- Ptolemy C Code Generation and Scheduling for the Network of Workstations (NOW)
- Parallel Raytracing using a Network of Workstations for Rendering Spline Surface Animation
- Parallel Monte Carlo Simulation
- Berkeley Search Engine
- Porting and Characterization of GATOR, an Atmospheric Chemical Tracer Model
- A Distributed Memory Concurrent B-tree Implementation
- Design, Implementation, and Performance Evaluation of a Portable Distributed Task Queue
- Porting The BLACS From MPL To GAM On The SP-1
- Implementation of a Parallel Preconditioned Conjugate Gradient (PCG) Solver in Finite Element
- Parallelizing Impulse, a dynamic simulation system.
- Model of LPARX multigrid performance on the CM5

Paul Debevec

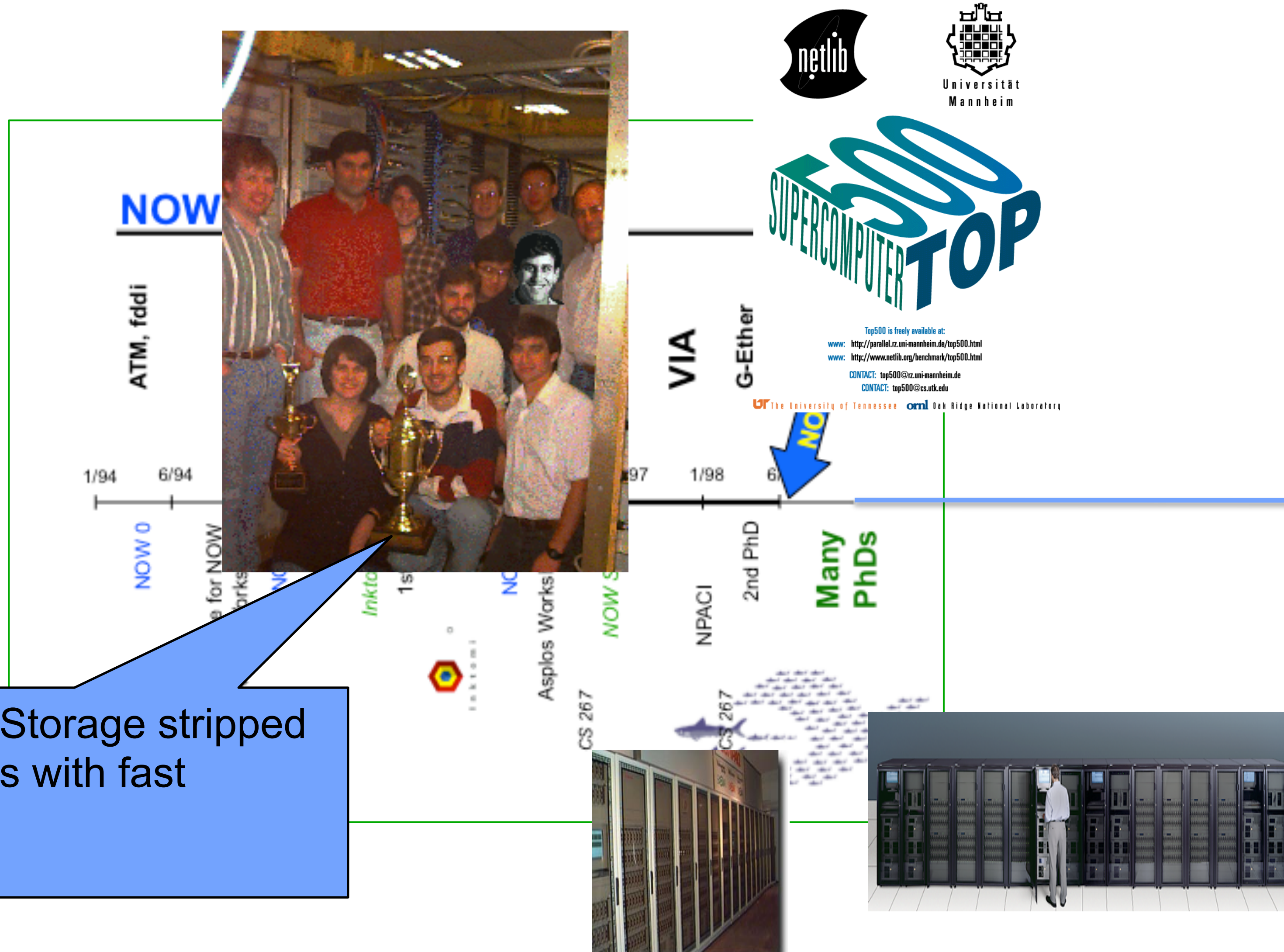
http://www.pauldebevec.com/FiatLux/movie/

inktomi.berkeley.edu

- **World's 1st Massive AND Fast search engine**

1996 inktomi.com

World Record Sort, 1st Cluster on Top 500

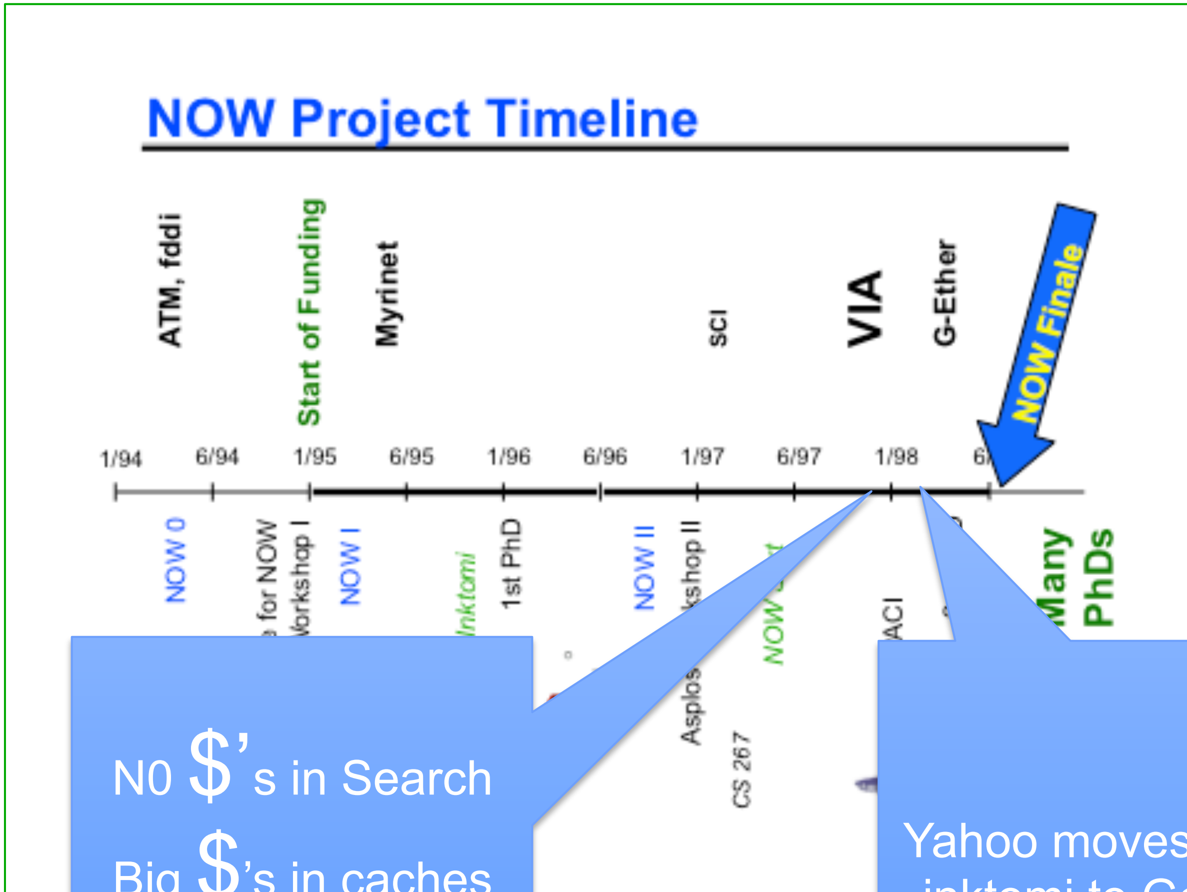


Distributed File Storage stripped over all the disks with fast communication.

Massive Cheap Storage

Serving Fine Art at http://www.thinker.org/imagebase/

… google.com



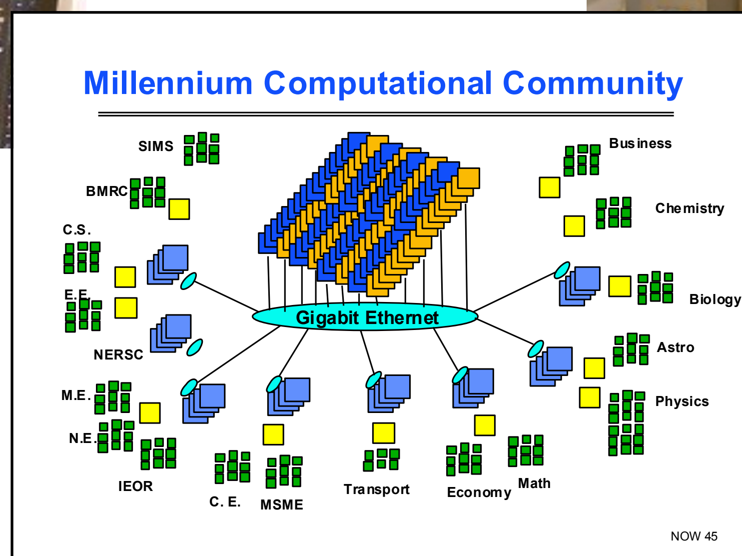
N0 $’s in Search

Big $’s in caches

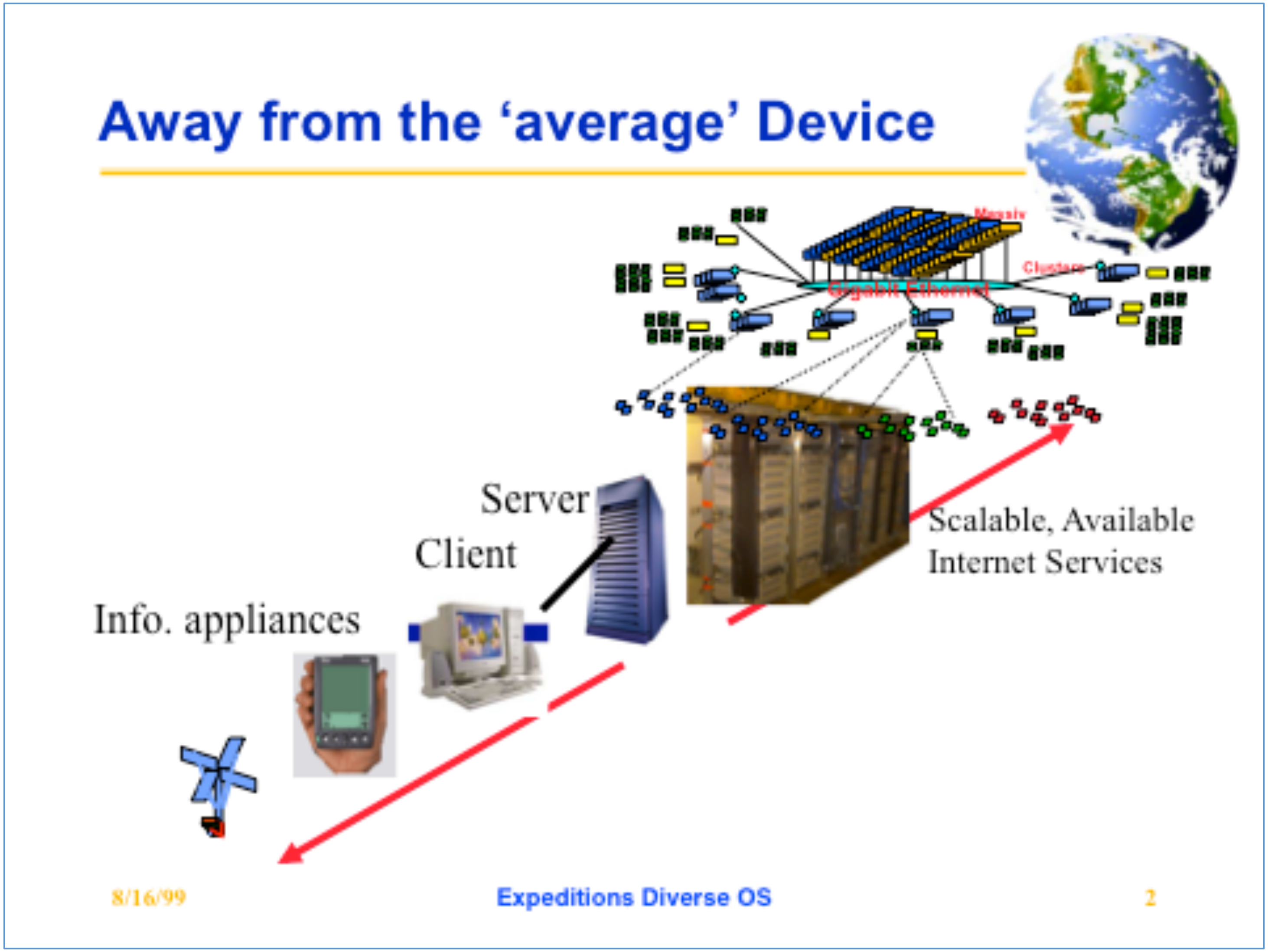
??? $’s in mobile

Yahoo moves from inktomi to Google

meanwhile Clusters of SMPs

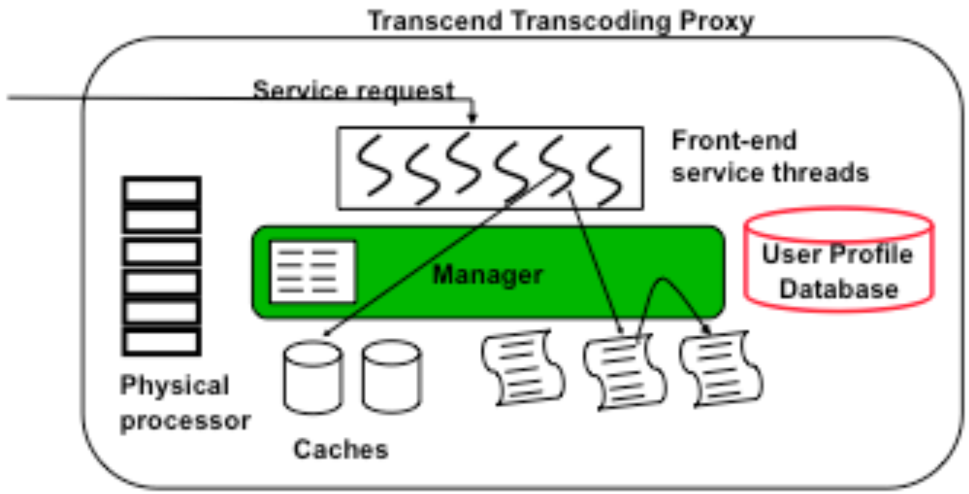


Expeditions to the 21st Century



Internet Services to support small mobile devices

Service Based Applications

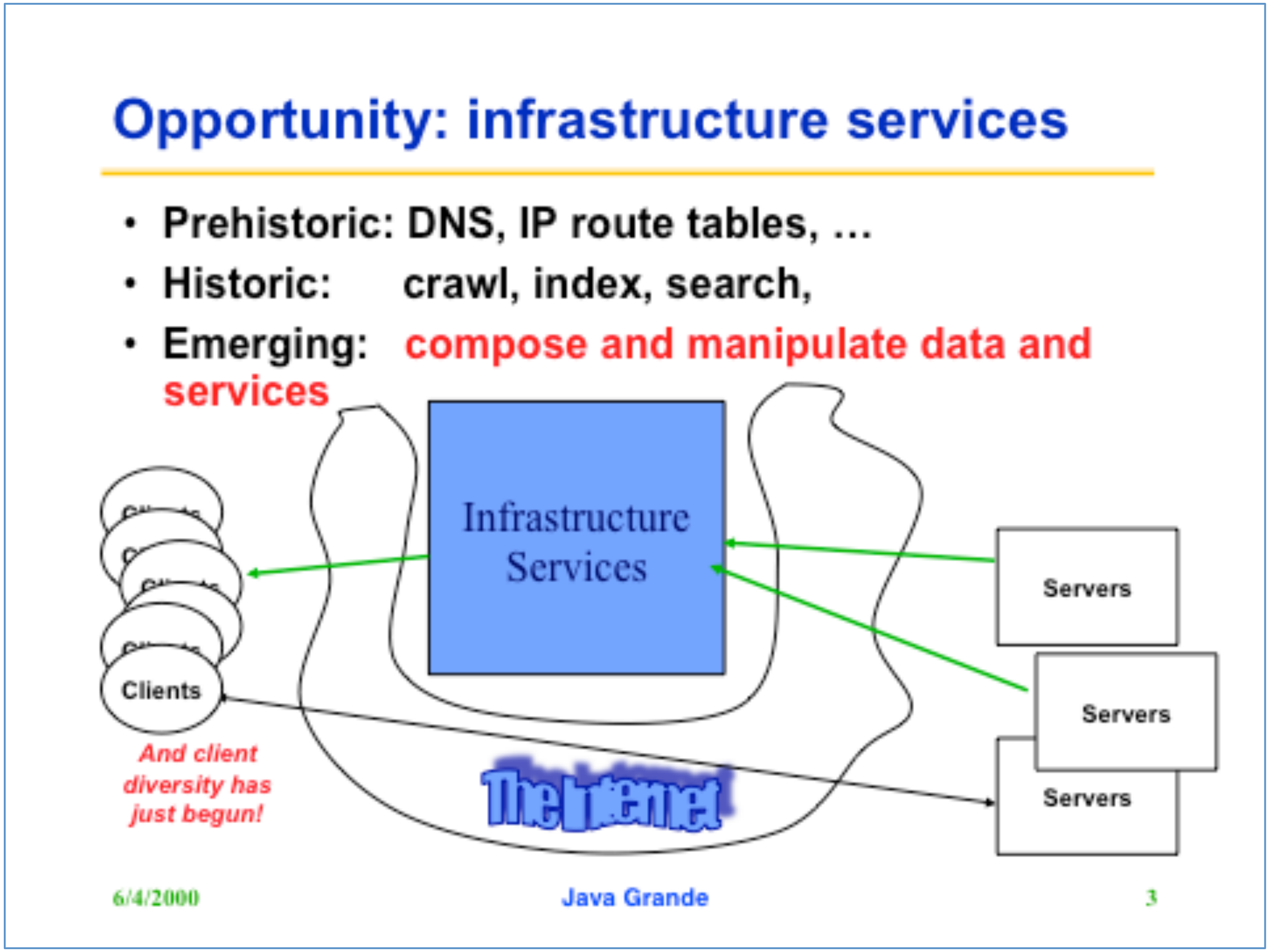


- Application provides services to clients
- Grows/Shrinks according to demand, availability, and faults

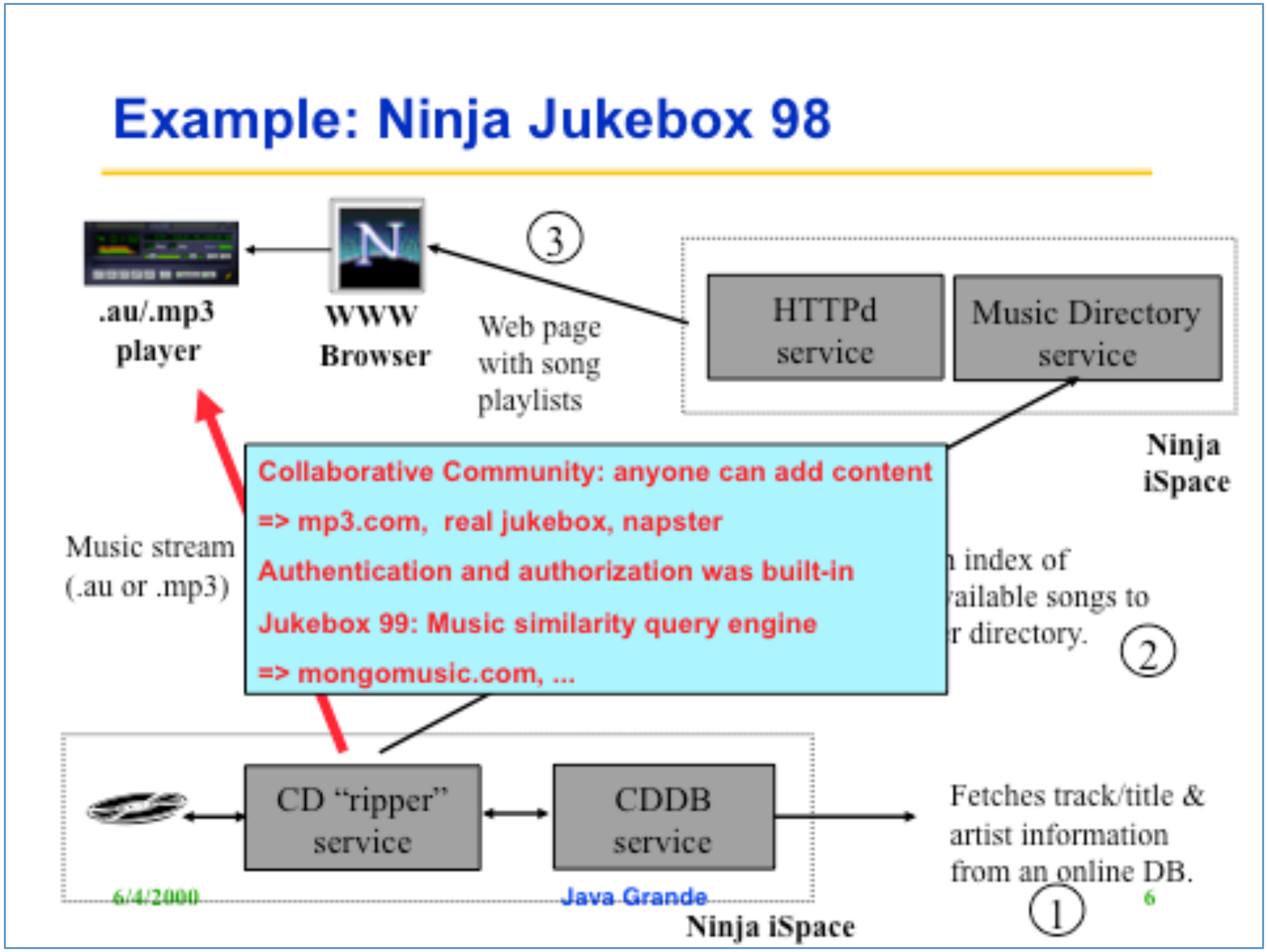
7/30/98 HPDC Panel 17

UNTANGLING THE WEB: UC Berkeley graduate students Steve Gribble, Armando Fox, and Yatin Chawathe (left to right) have created a system called Transend that can speed up modem access to the World Wide Web by distilling image files.

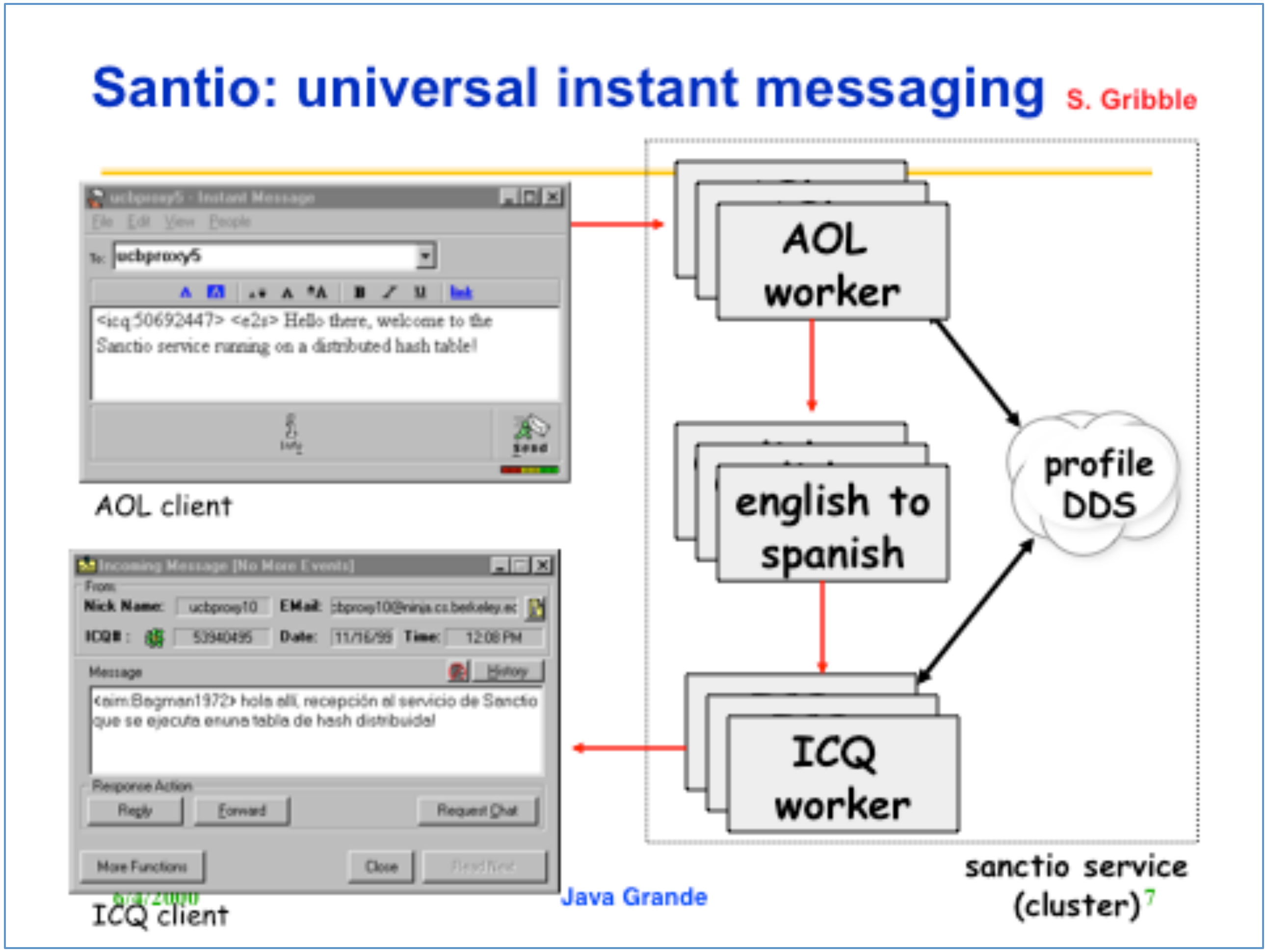
Ninja Internet Service Architecture



Startup of the Week …



… and …



Santio: universal instant messaging
S. Gribble
ucbproxy5
<icq:50692447> <e2s> Hello there, welcome to the Sanctio service running on a distributed hash table!
AOL client
<aim:Bagman1972> hola alli, recepción al servicio de Sanctio que se ejecuta enuna tabla de hash distribuida!
ICQ client
AOL worker
english to spanish
ICQ worker
profile DDS
sanctio service (cluster)
Java Grande

Existing Applications

- Ninja "NOW Jukebox"
  - Harnesses Berkeley Network of Workstations
  - Plays real-time MPEG-3 audio served from 1
- Voice-enabled room control
  - Speech-to-text Operators control room service
  - Eventual integration with GSM cell phones an
- Stock Trading Service
  - Accesses real-time stock data from Internet
  - Programmatic interface to buy/sell/trade stock
- NinjaFAX
  - Programmable remotely-accessed FAX machi
  - Send/receive FAXes; authentication used for a
- Keiretsu: The Ninja Pager Service
  - Provides instant messaging service via Web, l

Scalable, Distributed Data Structures for Internet Service Construction

Steven D. Gribble, Eric A. Brewer, Joseph M. Hellerstein, and David Culler
The University of California at Berkeley

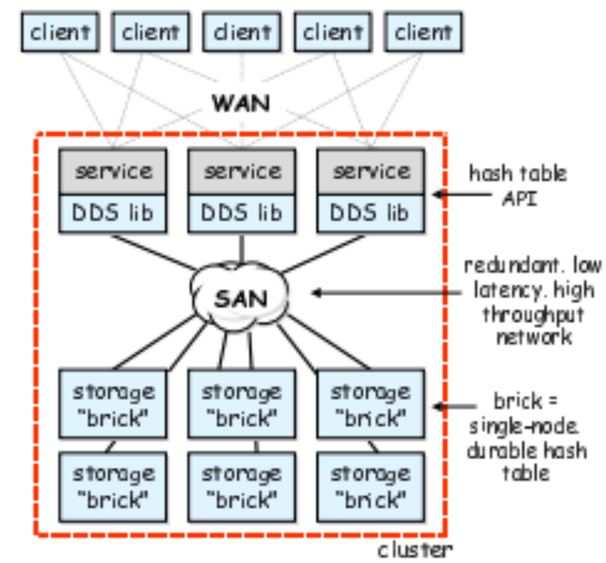
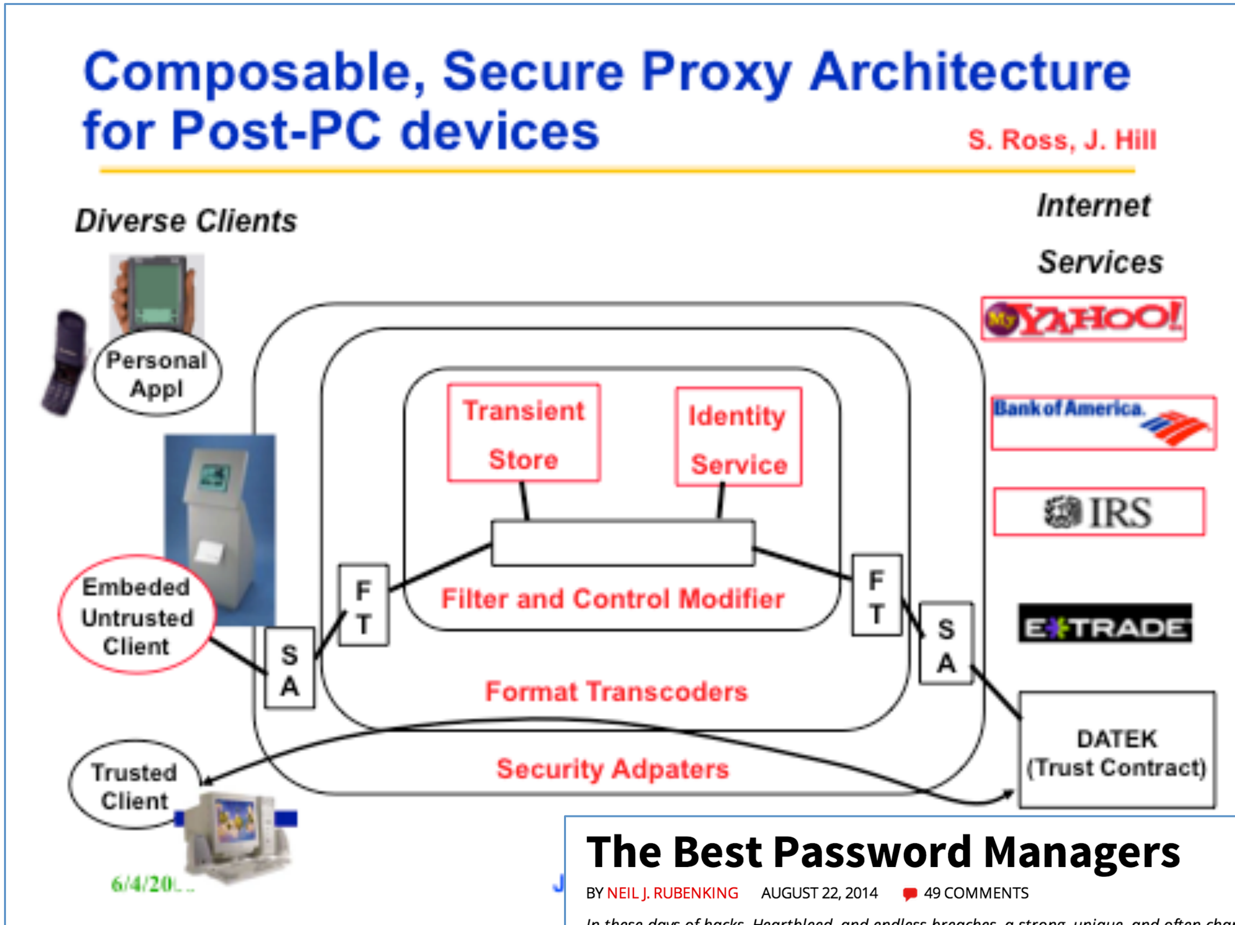


Figure 2: **Distributed hash table architecture:** each box in the diagram represents a software process. In the simplest case, each process runs on its own physical machine, however there is nothing preventing processes from sharing machines.

Gribble, 99

Security & Privacy in a Pervasive Web



The Best Password Managers

BY NEIL J. RUBENKING AUGUST 22, 2014 49 COMMENTS

In these days of hacks, Heartbleed, and endless breaches, a strong, unique, and often-changed password for every site is even more imperative. A password manager can help you attain that goal.

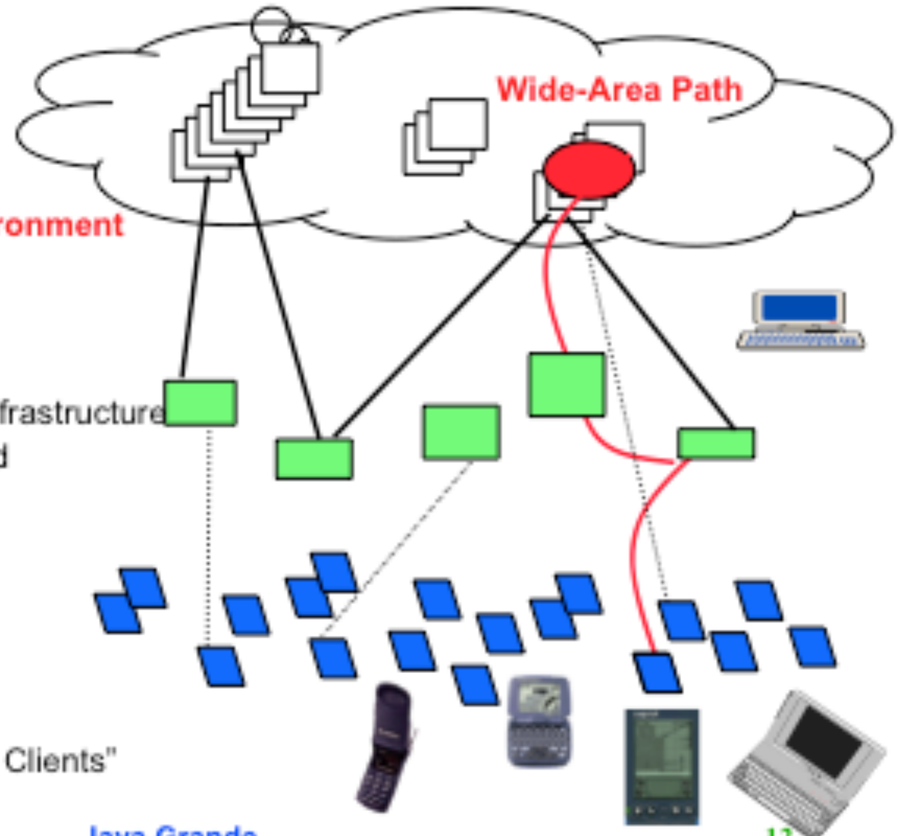
3.1K SHARES

Name	LastPass 3.0	LastPass 3.0 Premium	Dashlane 3	RoboForm Everywhere 7	Intuitive Password 2.9	Keeper Password Manager & Digital Vault 8	Norton Identity Safe	PasswordBox	RoboForm Desktop 7	Sticky Password 7
Editor Rating	EC EDITORS' CHOICE	EC EDITORS' CHOICE	EC EDITORS' CHOICE							

A decade before the cloud

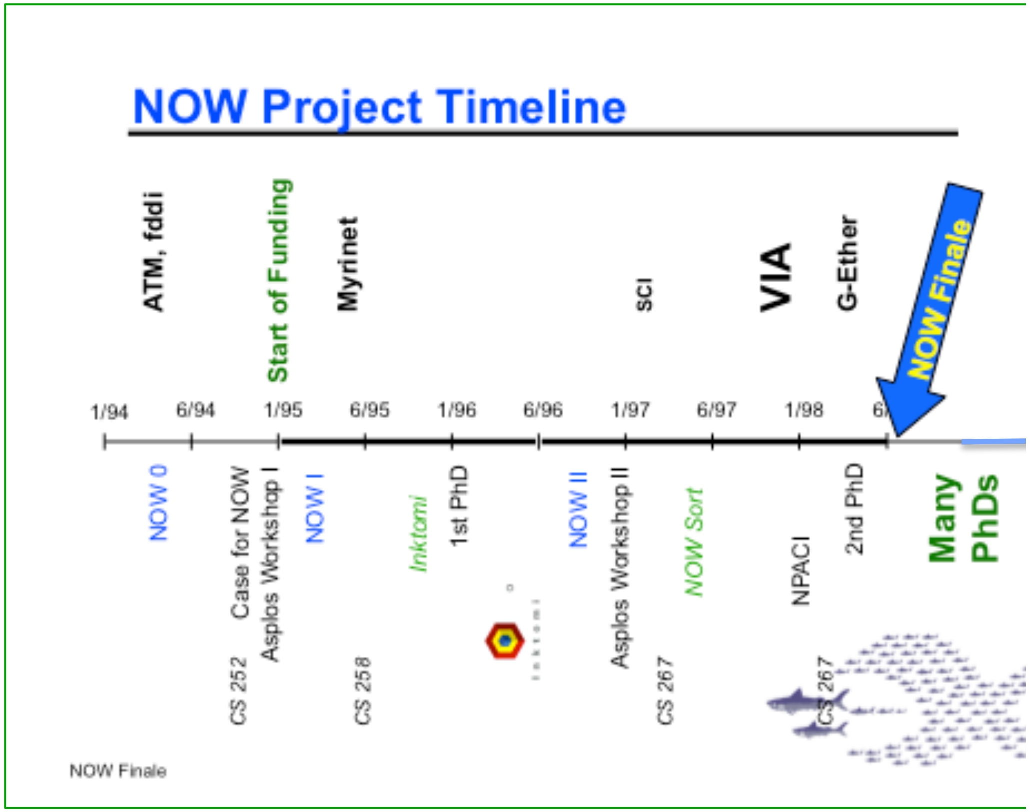
A 'Structured Architecture' Approach

- Bases (1M's)
  - scalable, highly available
  - persistent state
  - databases, agents
  - "home" base per user
  - service programming environment
- Active Proxies (100M's)
  - not packet routers
  - bootstrap thin devices into infrastructure
  - soft-state and well-connected
- Units (1B's)
  - sensors / actuators
  - PDAs / smartphones / PCs
  - heterogeneous
  - Minimal functionality: "Smart Clients"



6/4/2000 Java Grande 12

99.9 Club


NOW Project Timeline
ATM, fddi
Start of Funding
Myrinet
SCI
VIA
G-Ether
NOW Finale
1/94
6/94
1/95
6/95
1/96
6/96
1/97
6/97
1/98
NOW 0
Case for NOW
Asplos Workshop I
NOW I
Inktomi
1st PhD
NOW II
Asplos Workshop II
NOW Sort
NPACI
2nd PhD
Many PhDs
CS 252
CS 258
CS 267
CS 267
NOW Finale

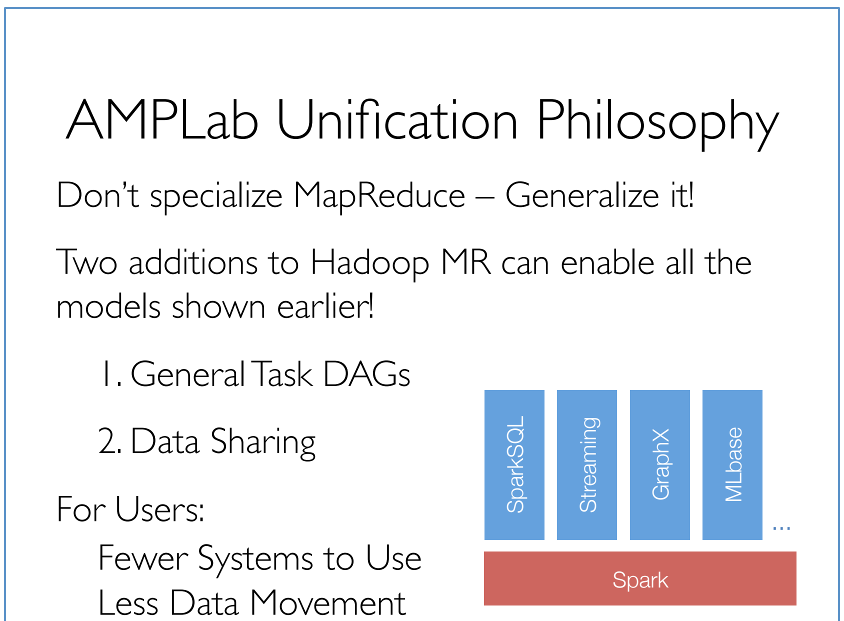
10th ANNIVERSARY REUNION 2008
Network of Workstations (NOW): 1993-98

NOW Team 2008: L-R, front row: Prof. Tom Anderson[†‡] (Washington), Prof. Rich Martin[‡] (Rutgers), Prof. David Culler[*†‡] (Berkeley), Prof. David Patterson[*†] (Berkeley).
Middle row: Eric Anderson (HP Labs), Prof. Mike Dahlin[†‡] (Texas), Prof. Armando Fox[‡] (Berkeley), Drew Roselli (Microsoft), Prof. Andrea Arpaci-Dusseau[‡] (Wisconsin), Lok Liu, Joe Hsu.
Last row: Prof. Matt Welsh[‡] (Harvard/Google), Eric Fraser, Chad Yoshikawa, Prof. Eric Brewer[*†‡] (Berkeley), Prof. Jeanna Neefe Matthews (Clarkson), Prof. Amin Vahdat[‡] (UCSD), Prof. Remzi Arpaci-Dusseau (Wisconsin), Prof. Steve Lumetta (Illinois).

*3 NAE members †4 ACM fellows ‡9 NSF CAREER Awards

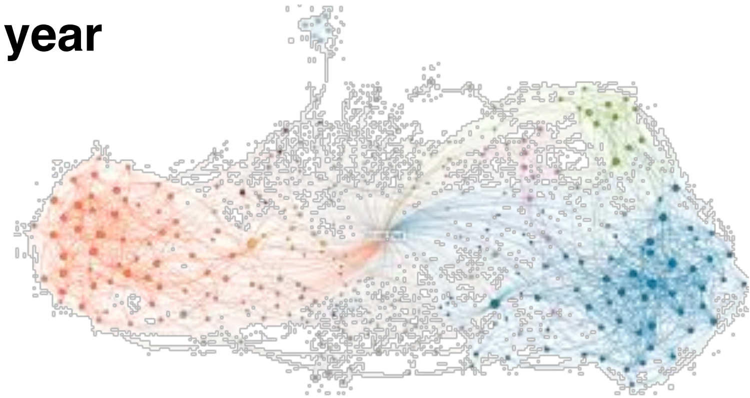
Time Travel

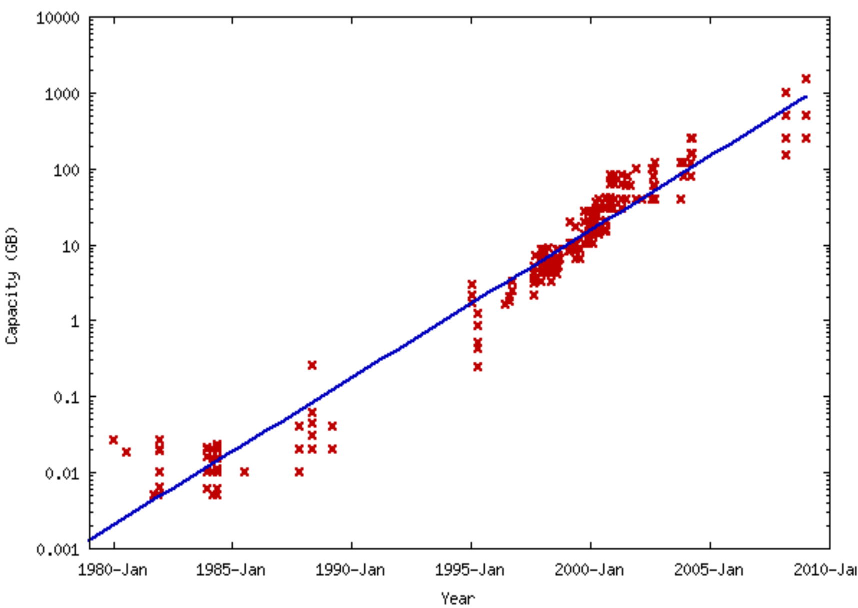
- **It's not just storing it, it's what you do with the data**

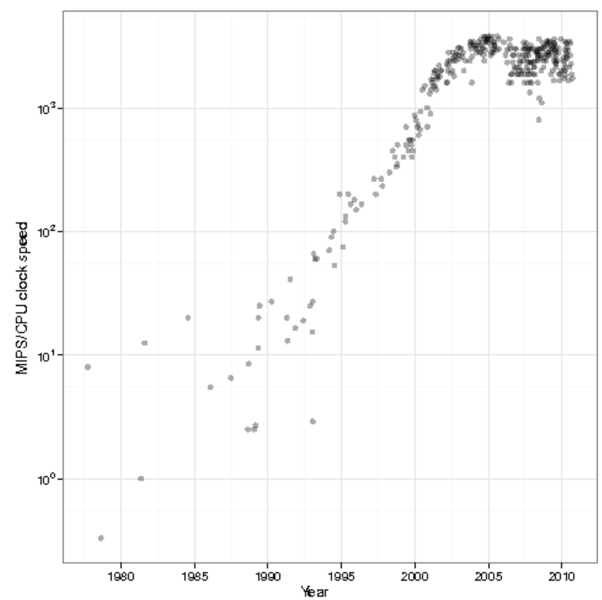


The Data Deluge

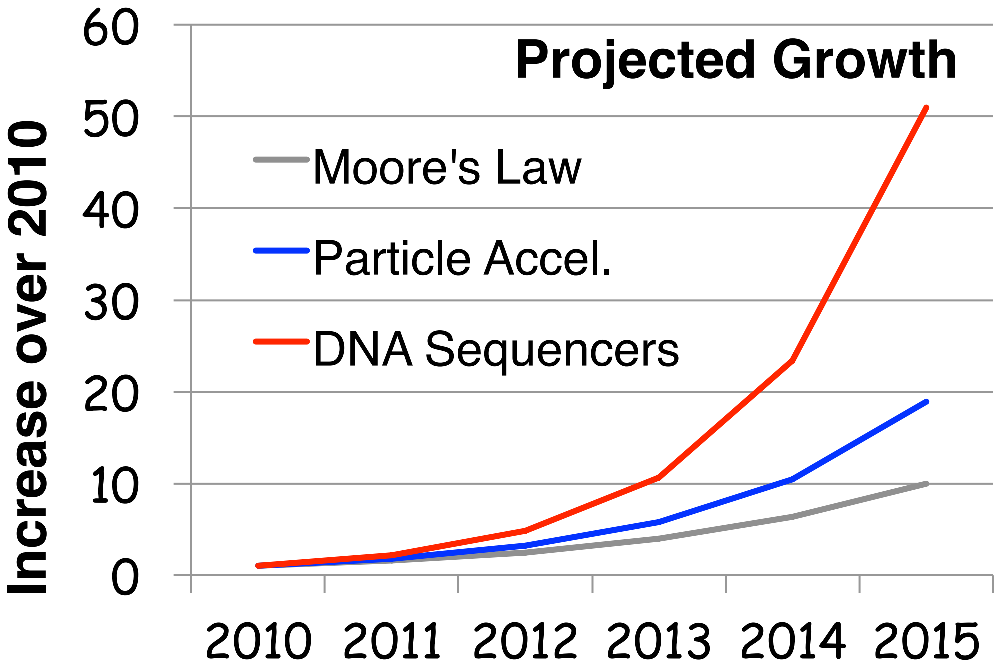
- Billions of users connected through the net
  - WWW, Facebook, twitter, cell phones, …
  - 80% of the data on FB was produced last year
- Clock Rates stalled
- Storage getting cheaper
  - Store more data!





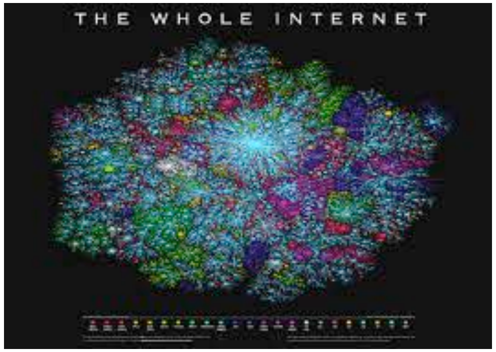


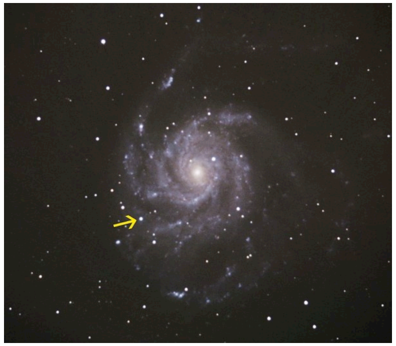
Data Grows Faster than Moore's Law



Complex Questions

- **Hard questions**
  - What is the impact on traffic and home prices of building a new ramp?
- **Detect real-time events**
  - Is there a cyber attack going on?
- **Open-ended questions**
  - How many supernovae happened last year?





MapReduce Pros

- **Distribution is completely transparent**
  - Not a single line of distributed programming (ease, correctness)
- **Automatic fault-tolerance**
  - Determinism enables running failed tasks somewhere else again
  - Saved intermediate data enables just re-running failed reducers
- **Automatic scaling**
  - As operations as side-effect free, they can be distributed to any number of machines dynamically
- **Automatic load-balancing**
  - Move tasks and speculatively execute duplicate copies of slow tasks (*stragglers*)

MapReduce Cons

- **Restricted programming model**
  - Not always natural to express problems in this model
  - Low-level coding necessary
  - Little support for iterative jobs (lots of disk access)
  - High-latency (batch processing)

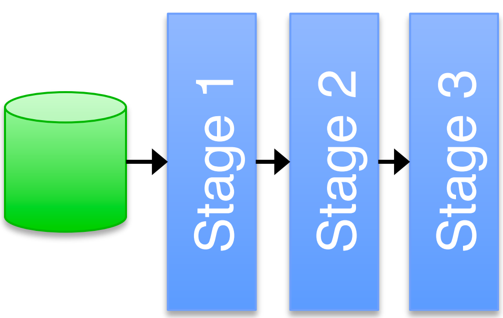
- **Addressed by follow-up research and Apache projects**
  - Pig and Hive for high-level coding
  - Spark for iterative and low-latency jobs

UCB / Apache Spark Motivation

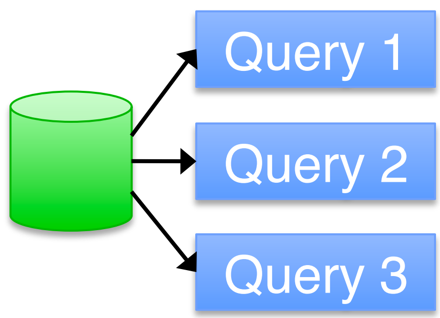
Spark
Lightning-Fast Cluster Computing

Complex jobs, interactive queries and online processing all need one thing that MR lacks:

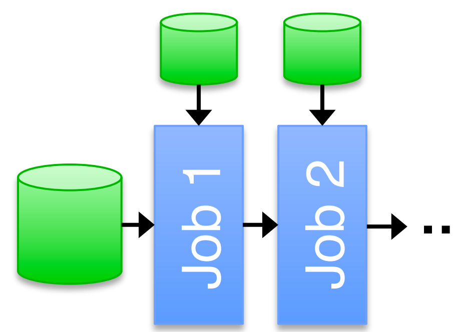
Efficient primitives for data sharing



Iterative job



Interactive mining



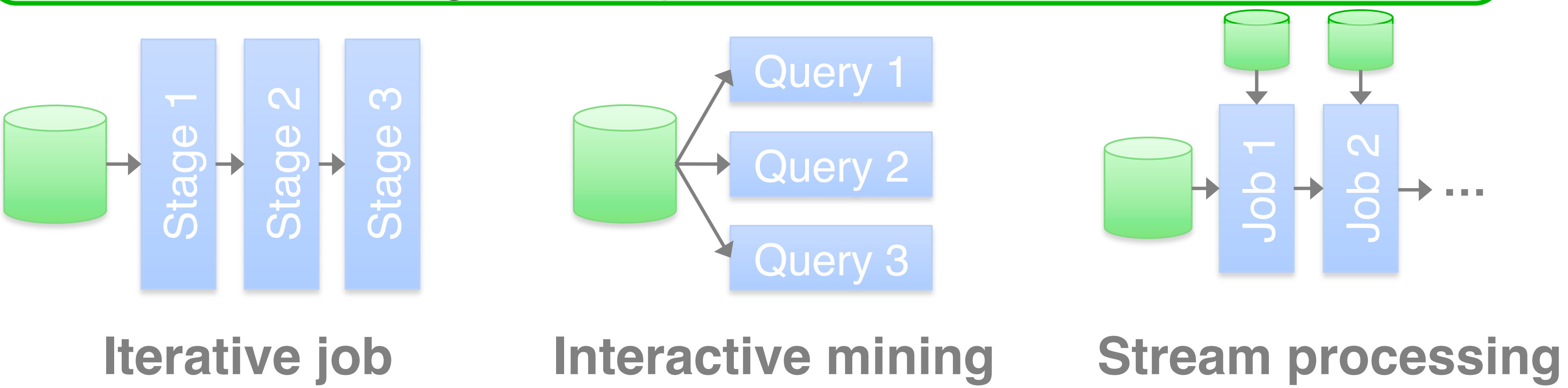
Stream processing

Spark Motivation

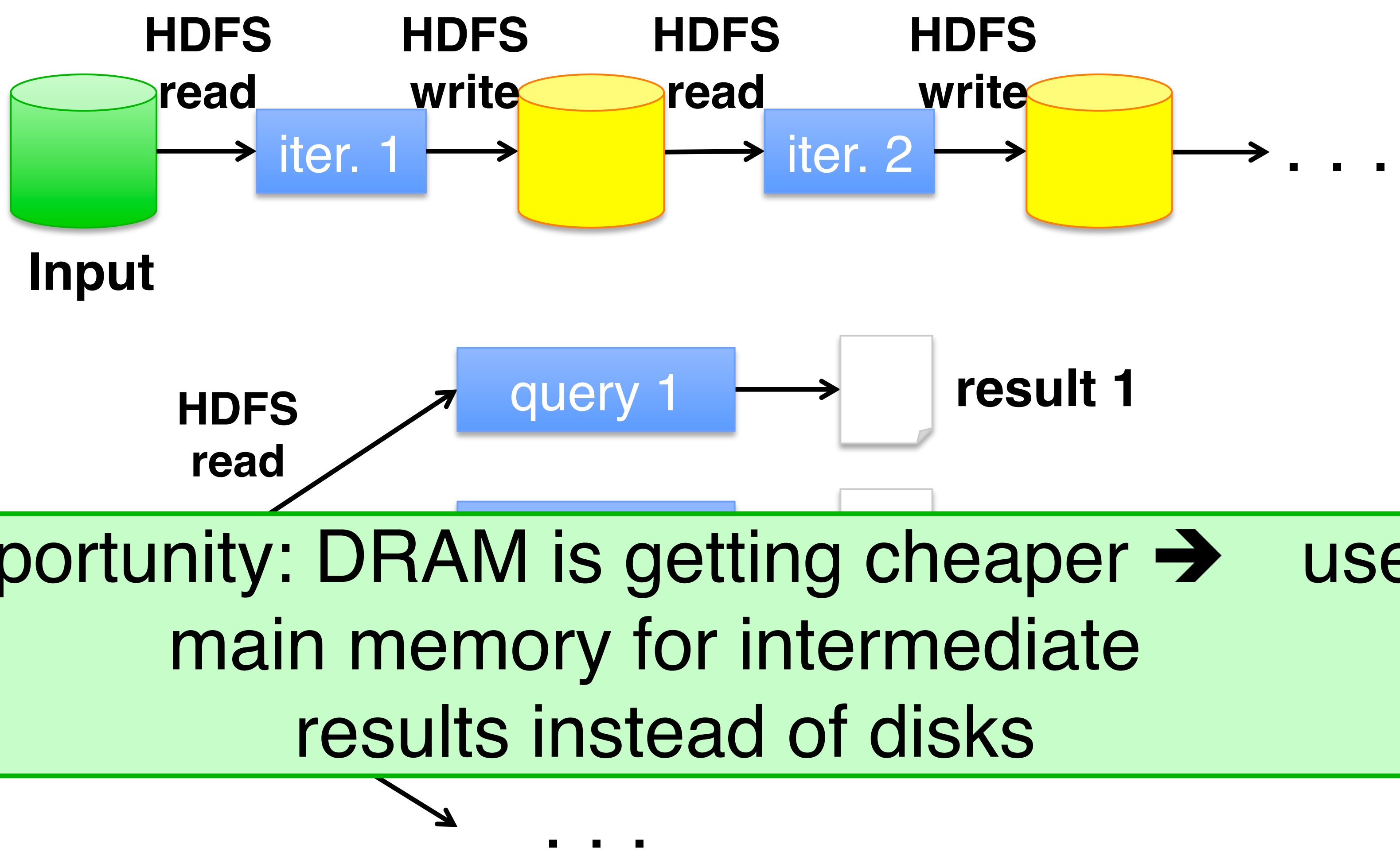
Complex jobs, interactive queries and online processing all need one thing that MR lacks:

Efficient primitives for data sharing

Problem: in MR, the only way to share data across jobs is using stable storage (e.g. file system) ➔ slow!

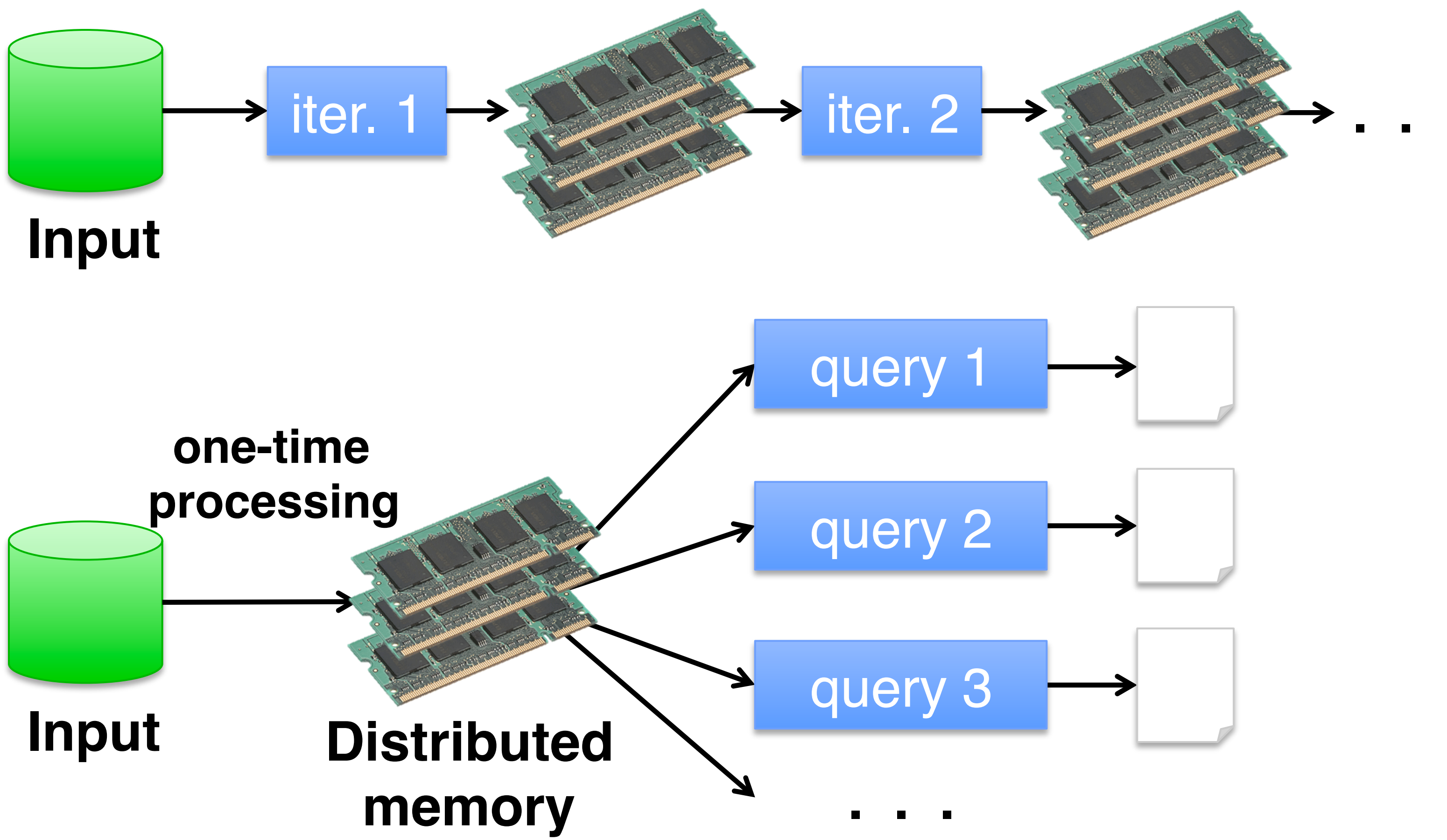


Examples



Opportunity: DRAM is getting cheaper → use main memory for intermediate results instead of disks

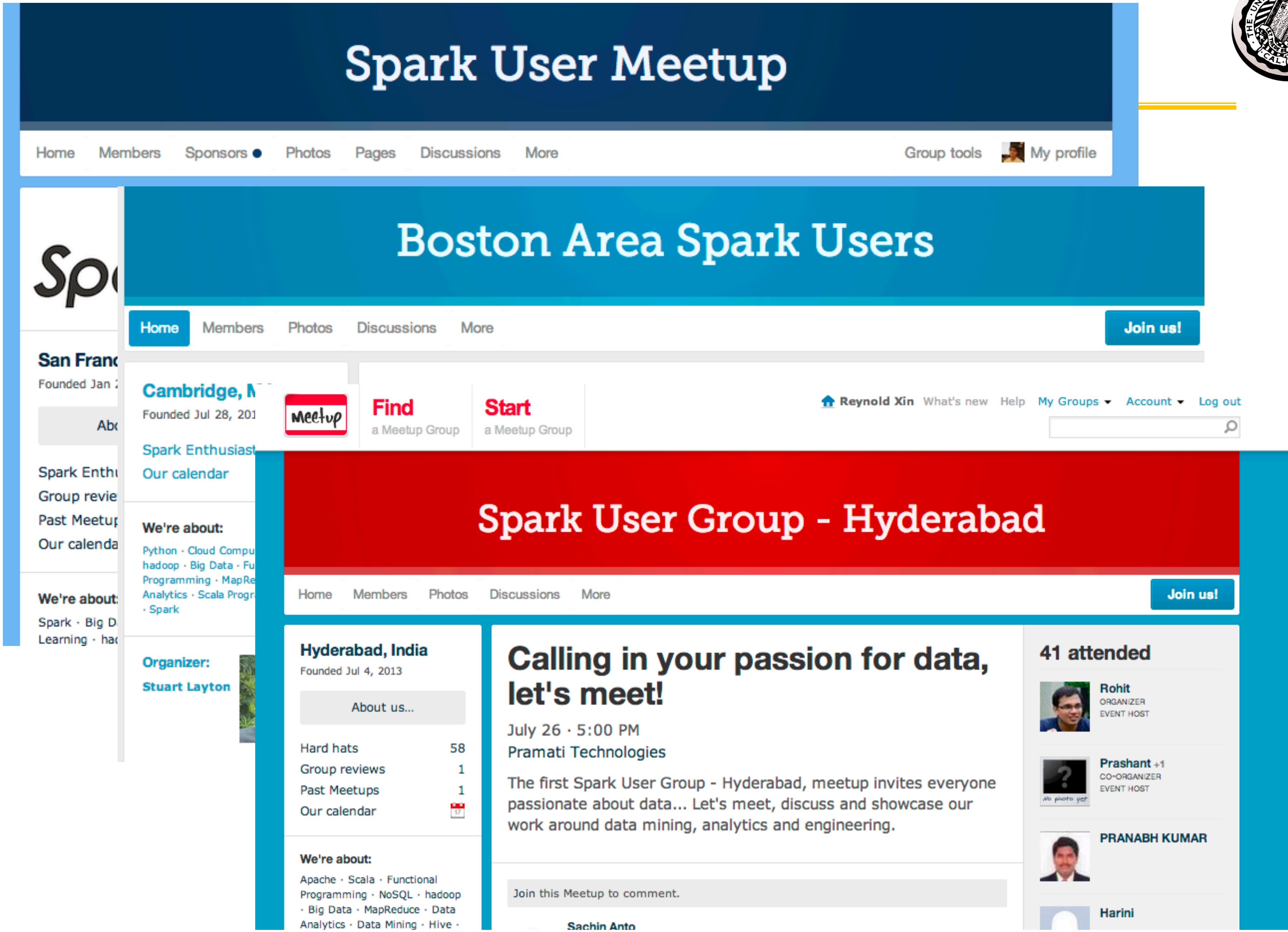
Goal: In-Memory Data Sharing



10-100× faster than network and disk

Solution: Resilient Distributed Datasets (RDDs)

- Partitioned collections of records that can be stored in memory across the cluster
- Manipulated through a diverse set of transformations (map, filter, join, etc)
- Fault recovery without costly replication
  - Remember the series of transformations that built an RDD (its lineage) to recompute lost data
- http://spark.apache.org/


Spark User Meetup
Home Members Sponsors Photos Pages Discussions More
Group tools My profile
San Franc
Founded Jan
Spark Enthu
Group revie
Past Meetu
Our calenda
We're about:
Spark · Big D
Learning · ha
Boston Area Spark Users
Home Members Photos Discussions More
Join us!
Cambridge, M
Founded Jul 28, 201
Spark Enthusiast
Our calendar
We're about:
Python · Cloud Compu
hadoop · Big Data · Fu
Programming · MapRe
Analytics · Scala Progr
· Spark
Organizer:
Stuart Layton
Meetup
Find
a Meetup Group
Start
a Meetup Group
Reynold Xin What's new Help My Groups Account Log out
Spark User Group - Hyderabad
Home Members Photos Discussions More
Join us!
Hyderabad, India
Founded Jul 4, 2013
About us...
Hard hats 58
Group reviews 1
Past Meetups 1
Our calendar
We're about:
Apache · Scala · Functional Programming · NoSQL · hadoop · Big Data · MapReduce · Data Analytics · Data Mining · Hive ·
Calling in your passion for data, let's meet!
July 26 · 5:00 PM
Pramati Technologies
The first Spark User Group - Hyderabad, meetup invites everyone passionate about data... Let's meet, discuss and showcase our work around data mining, analytics and engineering.
Join this Meetup to comment.
Sachin Anto
41 attended
Rohit
ORGANIZER
EVENT HOST
Prashant +1
CO-ORGANIZER
EVENT HOST
PRANABH KUMAR
Harini

Berkeley Data Analytics Stack (open source software)