

# *Distributed Computing at Multi-dimensional Scale*



---

Dr. Alfred Z. Spector  
VP, Research and Special Initiatives  
Google, Inc.  
Middleware 2008, Dec. 3, 2008

# Outline

---

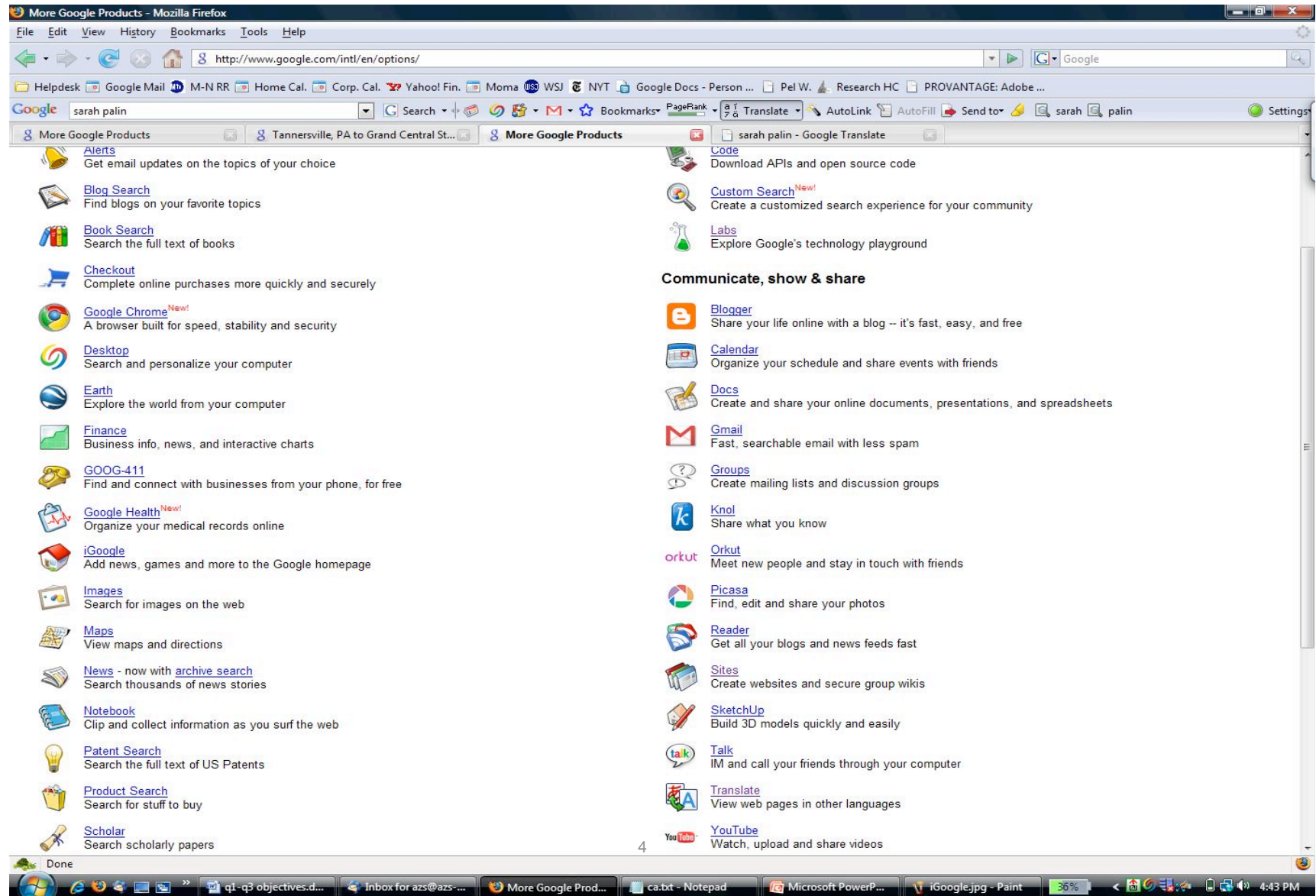
1. Google's Mission, Approach, Innovation @ Google
2. Distributed System Design Challenges
3. "Orthodox View of Distributed Computing" of 70's/80's
4. Technologies Now Employed to Meet Challenges
5. Challenges Unmet
6. The Broader Challenge
7. Summary

# Google's Mission

**Organizing the world's information and  
Making it universally accessible and useful.**



# Search and more...



# How (1): A Focus on Services

---

- **Google primarily delivers services to achieve it's mission**
- **Implications**
  - Lower cost of Development
  - Economies of scale
    - Lower cost of Installation
    - Lower cost of Operation
  - Resilience
  - Location transparency
  - Service Integration
  - Aggregated user feedback
  - ...

## How (2): The Google Common Distributed System

---

- **Vast:**

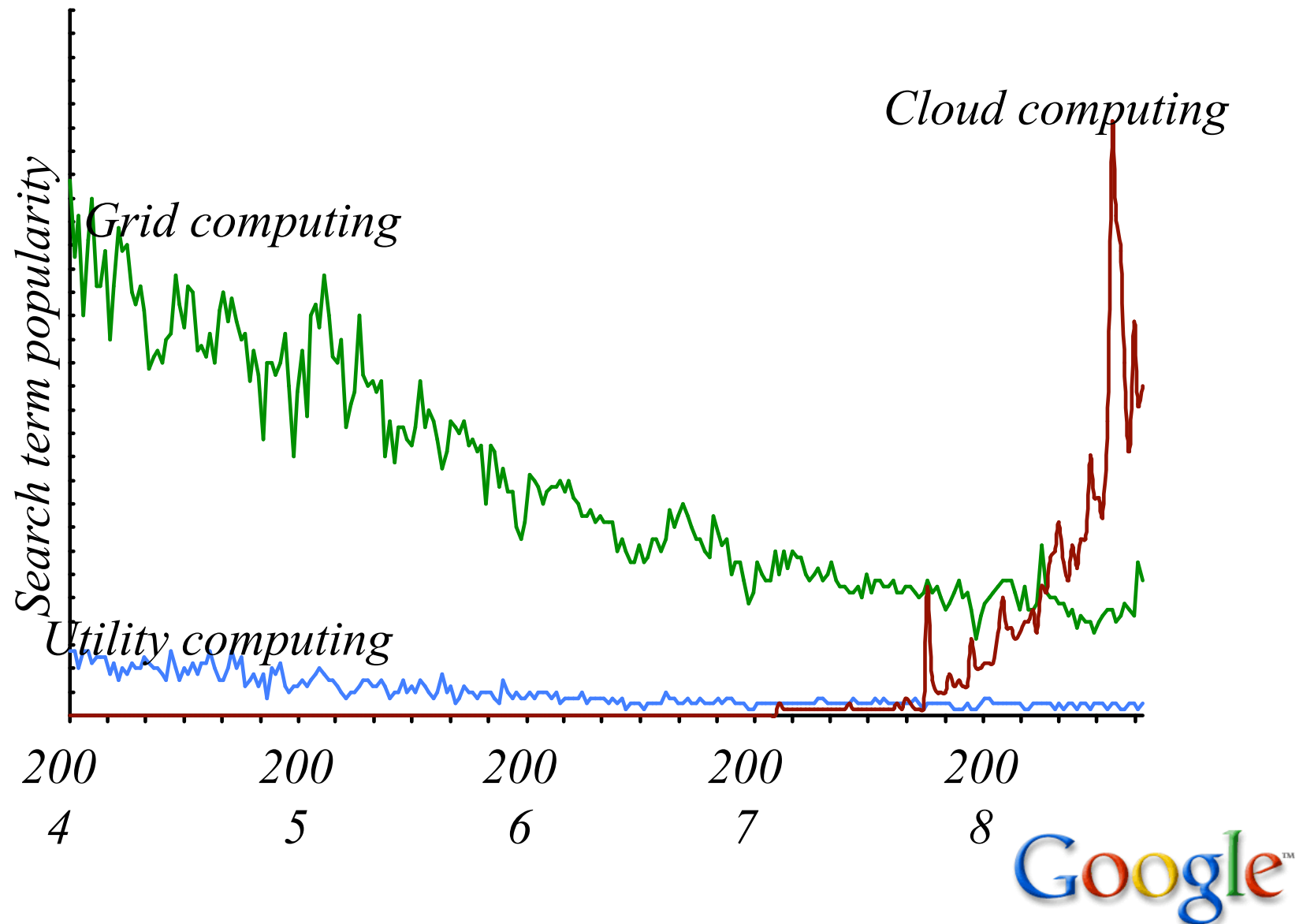
- Data in the cloud
- Processing in the cloud
- Global Usage

*(Feasible due to large clusters using decades of distributed computing research)*

- **Implications:**

- Economies of scale from *shared* infrastructure
- Low barriers to product launch
- Decentralized development more feasible

# Terminological Evolution



Credit to Luiz André Barroso presentation on: A Case for Energy Proportional Computing

## *An elucidating moment: Cloud or Utility Computing*

---



Credit to Luiz André Barroso presentation on: A Case for Energy Proportional Computing

## How (3): Empiricism - Let Measurement & Feedback Rule

---

*108 milliseconds*      *-0.54*      *2.7M*      *0.55060*

*\$4.78 RPM*

*-0.0000339*      *2,800,000,000 views*

*425,440.01*

*56.76%*

*17.35*

*108 seconds/search*

*9995.55*

*1.3 searches per user*

*480,000,000 total pageviews*

*1607.44*

*10,400*

*6.55*

*\$0.303 CPA*

*108*

*\$7,660,400*



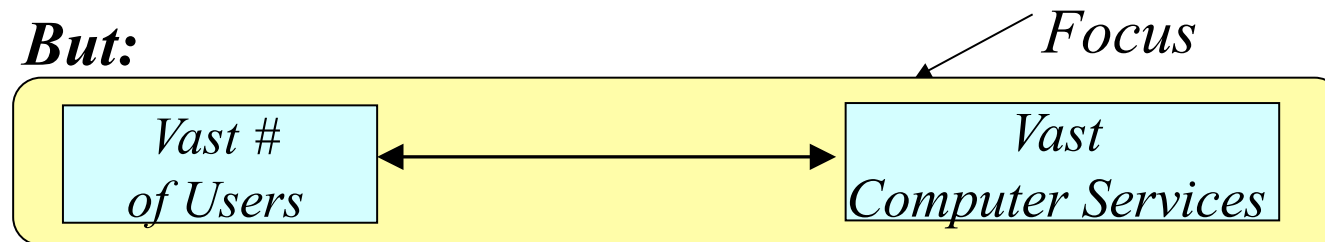
# Key Change: Holistic Approach To Design

---

***Not:***



***But:***



- **Implications**
  - Users and computers doing more than either could individually.
  - Virtuous circle from: *Data and Processing, Reach, Feedback in a virtuous circle.*

# Google's Research Mission

---

To innovate, and to catalyze innovation, and to learn in ways that collectively help Google achieve its mission

- Most areas of CS and related fields are relevant to Google
- We recognize much of our work can contributed to the field
- We pursue a diversified portfolio (w/various points on risk/reward curve)
- Strong relationship to academic community
- Increasing emphasis on publication



# Back to the Core Mission: Our Design Challenges

---

- Size of user community
- Storage Scale (requiring various characteristics)
  - E.g., security, privacy, availability,
- Processing Scale
  - High performance batch processing
  - High throughput
  - Low latency
- Rapid dynamics
- Communication Scale
  - Bandwidth
  - Endpoints
- Efficiency
  - Equipment
  - Communication
  - Power
  - Management
- Extensibility
- Compliance
- *And more to come, no doubt*



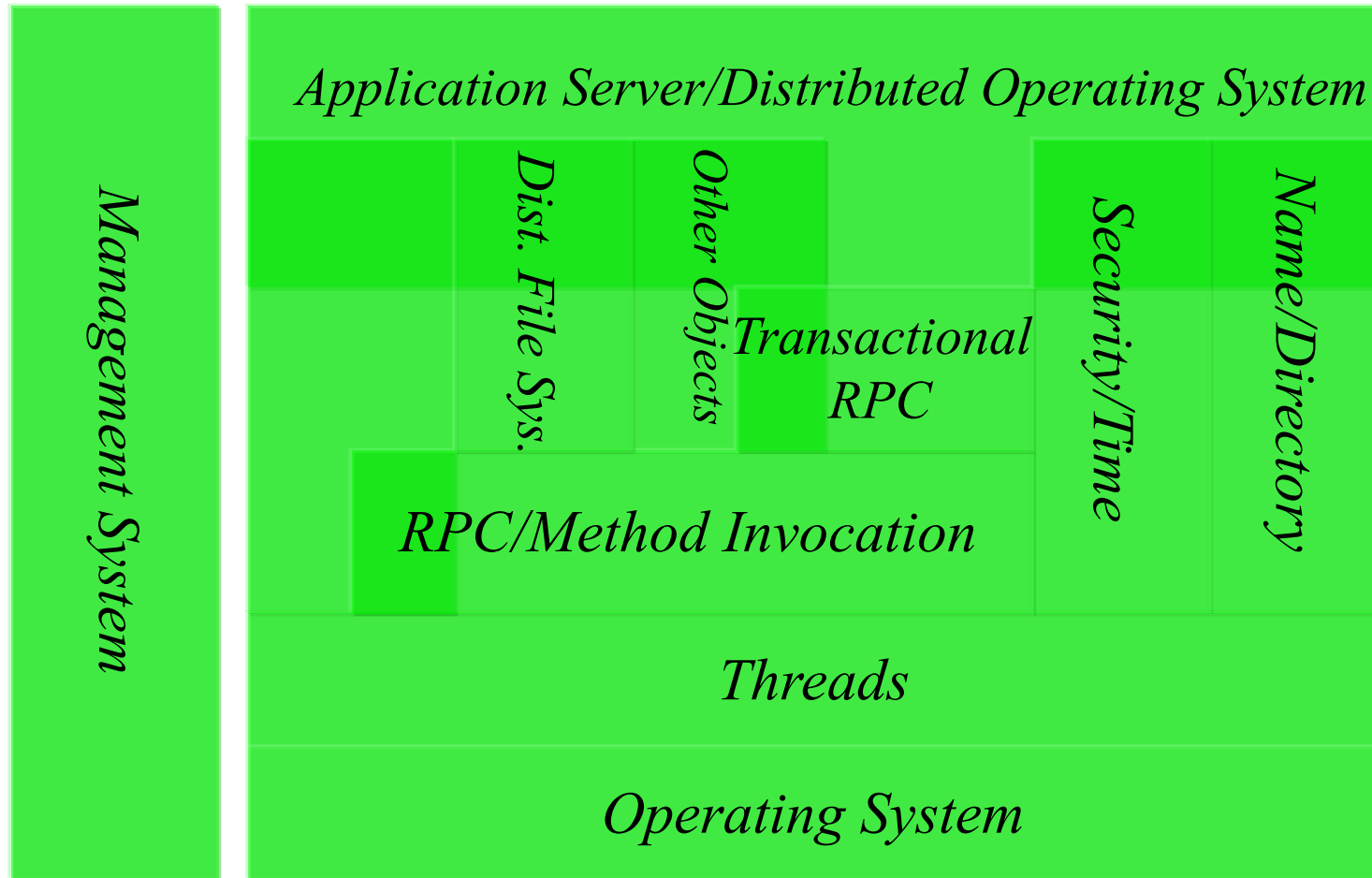
## Much Known About Distributed Computing in 70's

---

- Conceptually, the multicomputer understood
  - E.g., my thesis: Multiprocessing Architectures for Local Area Networks (1981) 😊
- Primitives: distributed synchronization, RPC, threading, ACL, authentication, replication, ...
- Most of the “ities:” consistency, availability, reliability, etc.
- Economics of technology evolution; e.g., limits to scaling individual CPUs
- Various memory architectures, from CM\* to flat networks

# Orthodox Architecture of 70's and 80's

---



My involvement: TABS, Camelot, DCE, Replication, Load balancing, RPC, (T)RPC, this architecture picture, etc.

*From AZS Pres. To CSTB Study on Dependability, May 18, 2004*

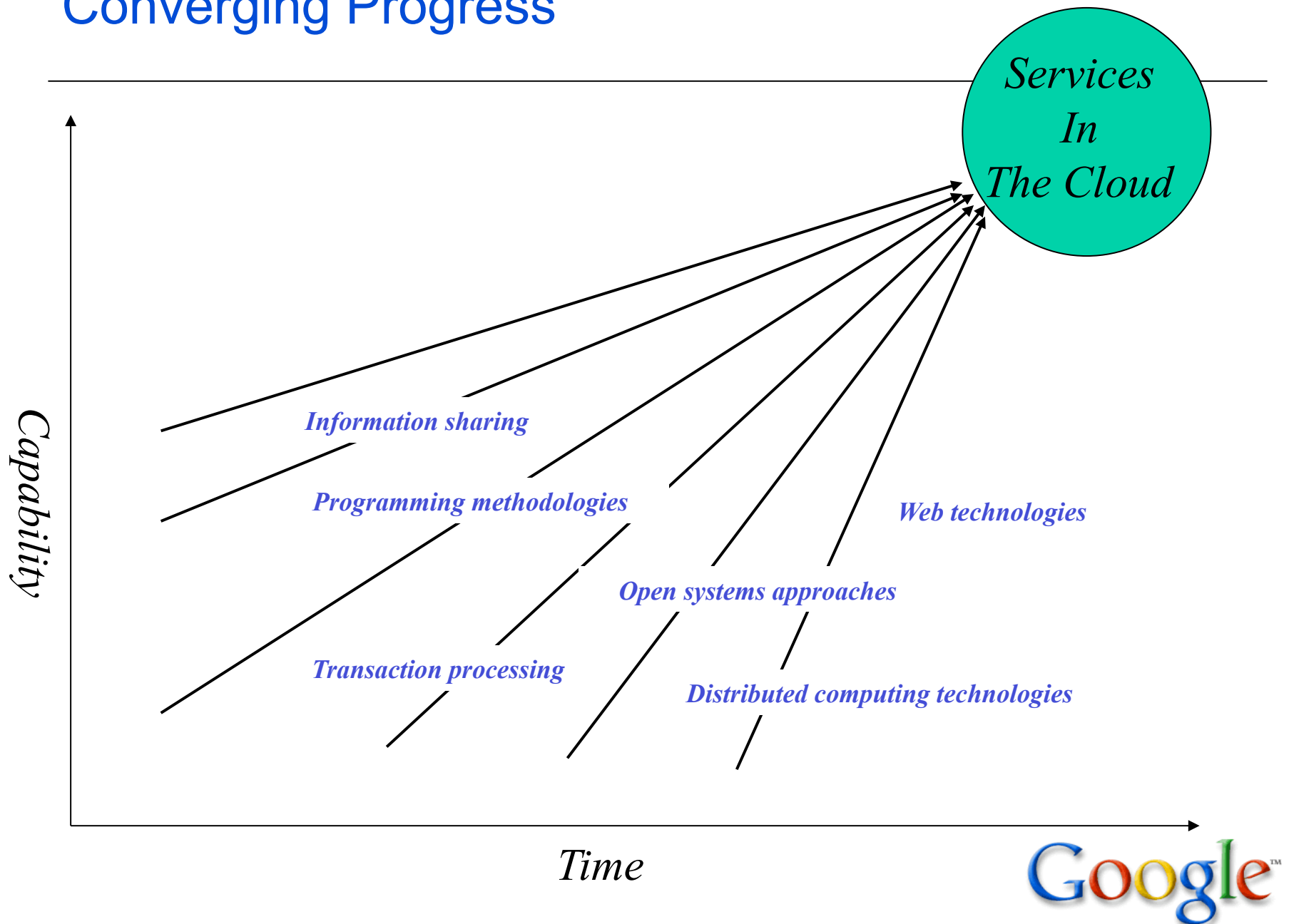


# What Wasn't Internalized Very Well

---

- The application mix
- The true nature of global, open systems:
  - Implications on systems, applications, mix and match.
- The implications of *operations* at true scale
  - E.g., work on programming & runtimes predominated system mgmt.
- The complexity of the architecture that would result
  - We tend to assume, *if we can conceive it, it's okay*.
- The collection of further abstractions that would build on fundamentals then known
- In summary, overall, limitation of understanding of (truly) large-scale, open integrated distributed systems

# Converging Progress



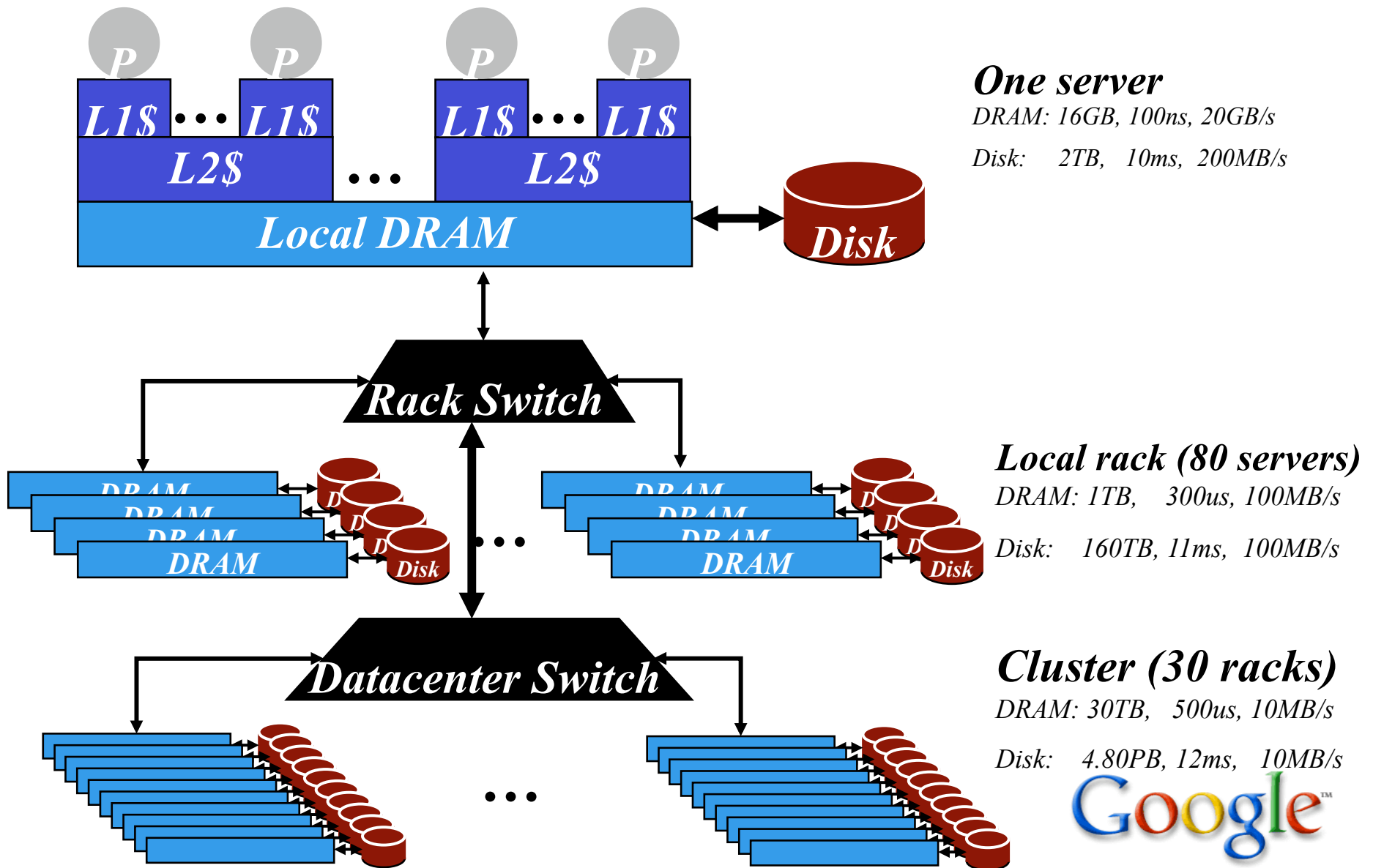
# Representative Technologies Google Employs Today

---

- Reasonably conventional single image OS's (Linux)
- Cluster architecture
- Network Communication:
  - TCP-IP, etc.
  - RPC
  - Web protocol stack
- Management
  - Cluster and network monitoring system
  - Cluster management system
- Security
  - Authentication, authorization, audit, statistical analysis,
- Storage
  - Distributed file system
  - Distributed record storage
  - Storage hierarchy management
- Distributed Operation
  - Load Balancing
  - Replication algorithms
  - Directory Structures
  - Map Reduce
  - Time Synchronization
  - Consensus algorithm (e.g., Paxos)
- *And probably more I've left out*



# Architectural view of the storage hierarchy



Credit to Luiz André Barroso presentation on: A Case for Energy Proportional Computing

# Reliability & Availability

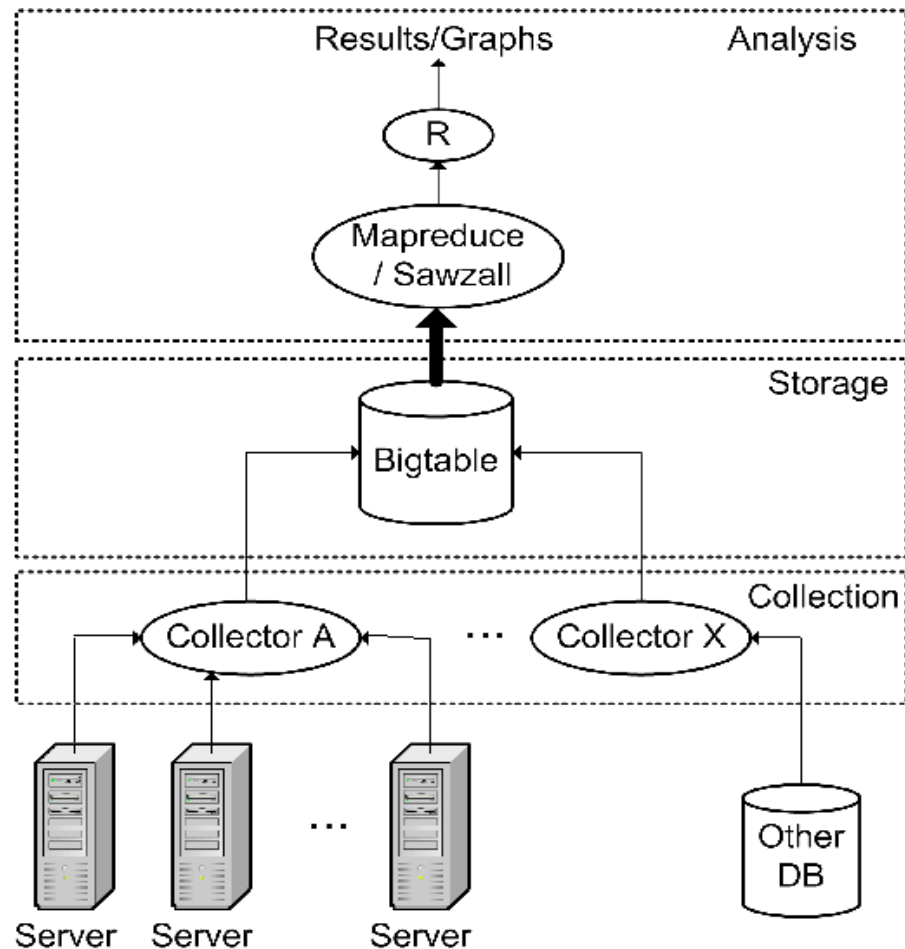
---

- Things will crash.
  - Take super reliable servers (MTBF of 30 years)
  - Build a machine with 10 thousand of those
  - Watch one fail per day
- Fault-tolerant software is inevitable
- Typical yearly flakiness metrics
  - 1-5% of your disk drives will die
  - Servers will crash at least twice (2-4% failure rate)
- Internet is not a “five nines” fabric
  - ~99% availability, varying significantly with geography
  - Wild dogs, sharks, dead horses, thieves, blasphemy, drunken hunters
  - Very strange bugs are common when you have lots of gear



# System Health Infrastructure

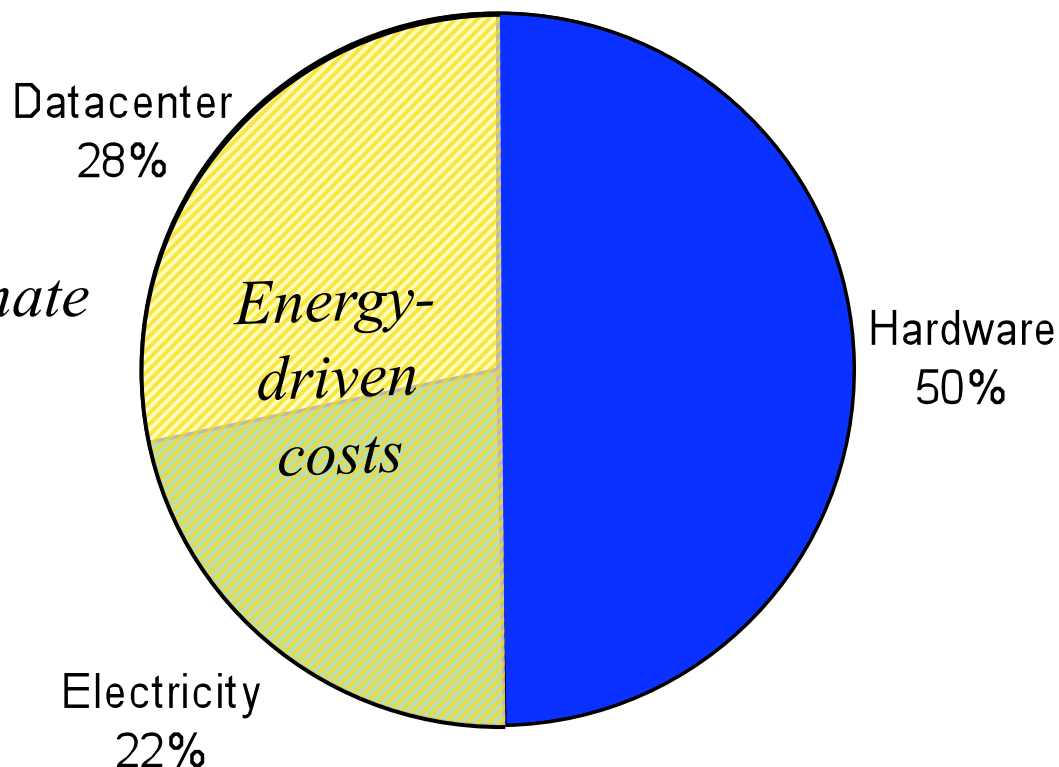
- Data collected on every machine, periodically
- Stored indefinitely
- Analysis done offline



## Rough cost breakdown: data center, electricity, hardware

---

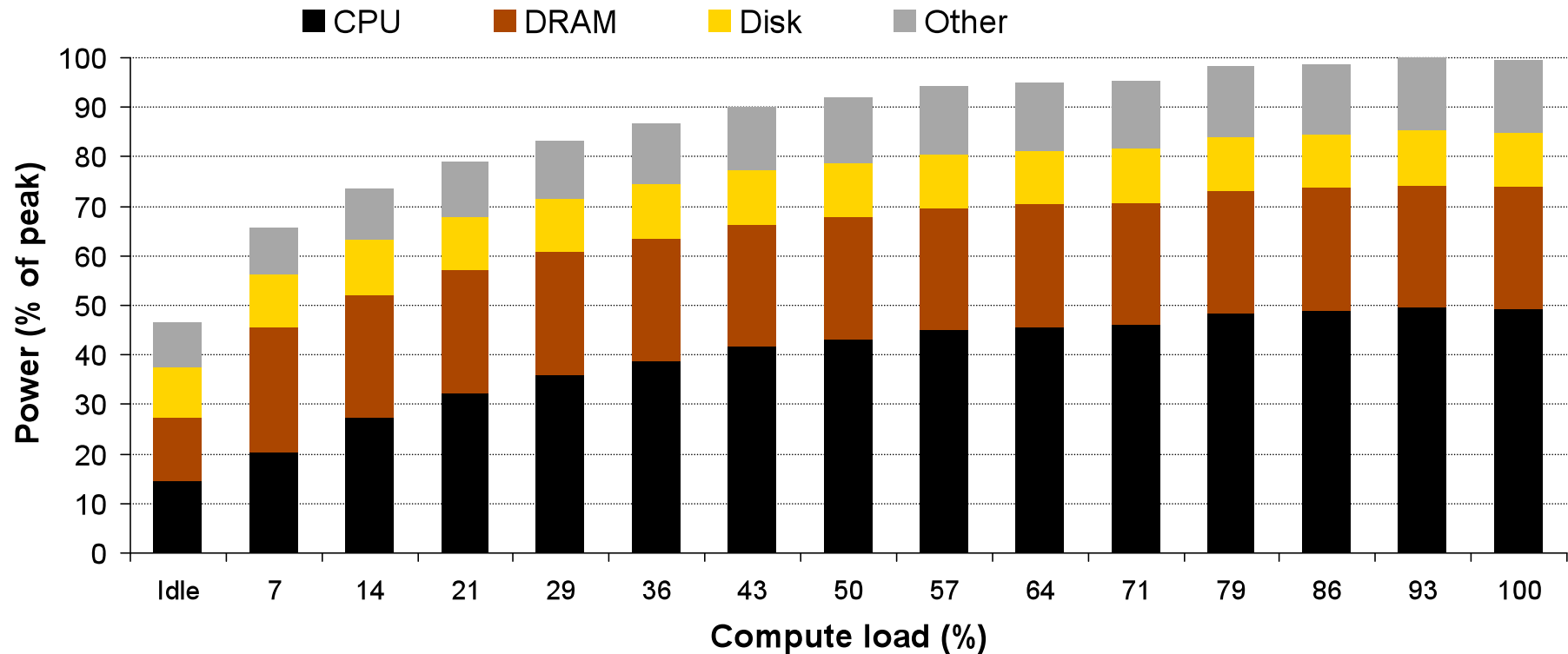
*Electricity bill doesn't dominate  
...but  
total energy-driven costs  
are already substantial!*



Hardware: Dell PowerEdge 2950, 4-year depreciation  
Energy: \$0.09/Kw-h, Average is 70% of peak  
Datacenter: \$17/watt of critical power, 12-year depreciation



# Power by component at different activity levels



- *CPU no longer dominates system power*
- *CPUs are more energy-proportional than the rest of the system*



# Open Challenges

---

- Problems span a wide range of areas:

*Focus of this talk on areas in bold*

|                                                                |                                |                                            |
|----------------------------------------------------------------|--------------------------------|--------------------------------------------|
| <i>Product design</i>                                          | <i>Algorithms &amp; Theory</i> | <i>...and<br/>much,<br/>much<br/>more!</i> |
| <i>User interfaces</i>                                         |                                |                                            |
| <i>Machine learning, Statistics, Information retrieval, AI</i> |                                |                                            |
| <b><i>Compilers, Programming languages</i></b>                 |                                |                                            |
| <b><i>Networking, Distributed systems, Fault tolerance</i></b> |                                |                                            |
| <b><i>Hardware, Mechanical engineering</i></b>                 |                                |                                            |

*Credit to Jeff Dean et al, Google Faculty Summit, Mountain View, 2008 for this and following systems challenge slides: various mods made for Middleware 2008*



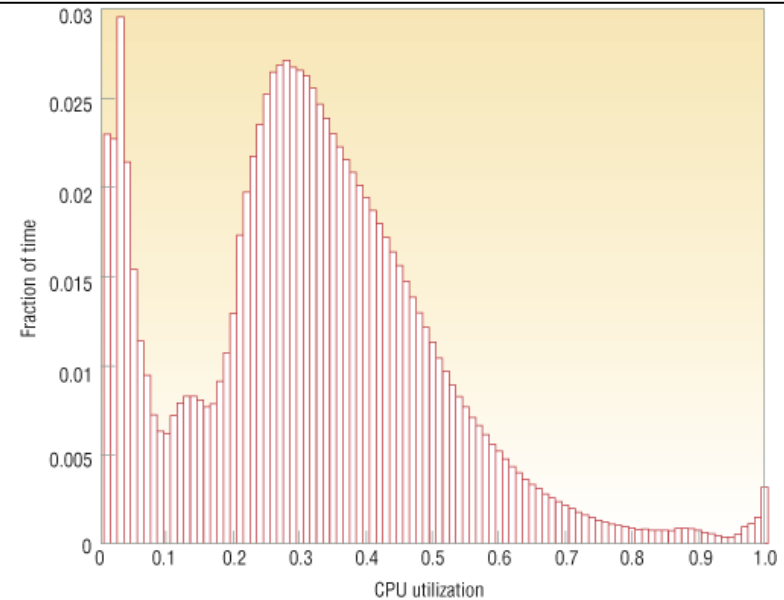
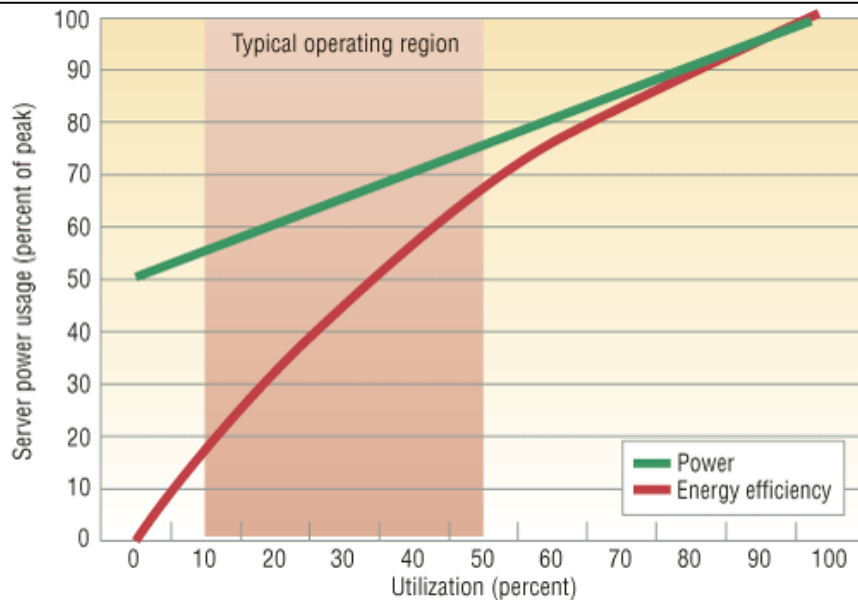
# Hardware & Energy Efficiency

---

- Moore's law is now scaling # cores instead of MHz
  - Fine with us: we love multi-core machines.
- Still want more computing capabilities, though...
  - Easy to get more computation by using more energy
  - Proportion of costs for energy will continue to grow, since Moore's law keeps computing cost roughly fixed
- Challenge: for every increase in HW performance, we need a corresponding increase in energy efficiency



# Energy Efficiency at Lower Utilization



- In a datacenter, machine utilization is usually 0.2 to 0.5
- In this range, energy efficiency is less than half the efficiency of a machine at 100% utilization
- Great for laptops, not so good for servers
- Challenge: Are there alternative designs that would give better energy efficiency at lower utilization?

# Operating System Design

---

- Our production machines all run Linux
  - Design largely inspired by UNIX design from the 70s
  - Still works reasonably well for us, but:
    - We don't use many aspects of the system (e.g. paging to disk)
    - Clusters of tens of thousands of machines are pretty removed from original design point



# Operating System Design

---

- Challenge: Server O.S. design aimed at 1000s of highly-connected machines in one building
  - remote paging to other machine's memory?
  - redo networking stack (given RTTs of 0.1 ms, not 100s of ms)?
  - different security model?
  - top-to-bottom performance isolation across apps/machines?



# Distributed Systems Abstractions

---

- High-level tools/languages/abstractions for building distributed systems
  - e.g. For batch processing, MapReduce handles parallelization, load balancing, fault tolerance, I/O scheduling automatically within a simple programming model
- Challenge: Are there unifying abstractions for other kinds of distributed systems problems?
  - e.g. systems for handling interactive requests & dealing with intra-operation parallelism
    - load balancing, fault-tolerance, service location & request distribution, ...
  - e.g. client-side AJAX apps with rich server-side APIs
    - better ways of constructing client-side applications?



# Building Applications w/Right Consistency Models

---

- Many applications need replicated state, across cluster or wide area, for reliability and availability
  - Consistent operations (e.g. use Paxos)
  - Inconsistent replication (many variants)
- Many apps need to use a mix of both of these:
  - e.g. Gmail:
    - Marking a message as read is asynchronous
    - Sending a message is requires greater consistency
- Challenge: General model of consistency choices, explained and codified
- Challenge: Easy-to-use abstractions for resolving conflicting updates to multiple versions of a piece of state



# The Joys of Real Hardware

---

- Typical first year for a new cluster:
  - ~0.5 **overheating** (power down most machines in <5 mins, ~1-2 days to recover)
  - ~1 **PDU failure** (~500-1000 machines suddenly disappear, ~6 hours to come back)
  - ~1 **rack-move** (plenty of warning, ~500-1000 machines powered down, ~6 hours)
  - ~1 **network rewiring** (rolling ~5% of machines down over 2-day span)
  - ~20 **rack failures** (40-80 machines instantly disappear, 1-6 hours to get back)
  - ~5 **racks go wonky** (40-80 machines see 50% packetloss)
  - ~8 **network maintenances** (4 might cause ~30-minute random connectivity losses)
  - ~12 **router reloads** (takes out DNS and external vips for a couple minutes)
  - ~3 **router failures** (have to immediately pull traffic for an hour)
  - ~dozens of minor **30-second blips for dns**
  - ~1000 **individual machine failures**
  - ~thousands of **hard drive failures**
- **slow disks, bad memory, misconfigured machines, flaky machines, etc.**



# Automated Systems Management via Machine Learning

---

- Challenge: machine learning techniques applied to monitoring/controlling such systems
  - automatic monitoring
    - learn to spot potential problems before they happen?
    - learn to spot unexpected failure modes?
    - automatically identify root causes of problems based on observed symptoms?
  - automatically figure out right strategies to adapt?



# Adaptivity in World-Wide Systems

---

- Challenge: automatic, dynamic world-wide placement of data & computation to minimize latency and/or cost, given constraints on:
  - bandwidth
  - packet loss
  - power
  - resource usage
  - failure modes
  - ...
- Users specify high-level desires:
  - “99%ile latency for accessing this data should be <50ms”
  - “Store this data on at least 2 disks in EU, 2 in U.S. & 1 in Asia”



# Privacy vs. Sharing

---

- There are undoubtedly people in the world that share some of your interests that you don't know
  - e.g. others with interests in “kite-based power generation”
- Data driven services are growing and increasingly important
  - ... but policy and technical reasons can limit what can be done
- Challenge: How can we build useful services that match people together based on their interests/behavior (for example) in ways that preserve their privacy?



# ACLs in Information Retrieval Systems

---

- Retrieval systems with mix of private, semi-private, widely shared and public documents
  - e.g. e-mail vs. shared doc among 10 people vs. messages in group with 100,000 members vs. public web pages
- Challenge: building retrieval systems that efficiently deal with ACLs that vary widely in size
  - best solution for doc shared with 10 people is different than for doc shared with the world
  - sharing patterns of a document might change over time
- Provide useful U.I. metaphors for information control to users
  - Users cannot manager ACLs as commonly construed



# Other Security Topics

---

- We know how to make systems rather secure, but it is difficult
  - Challenge: Reducing the cost, particularly over the lifecycle
- SPAM
  - The world can be considered one big open system
  - At least two challenges:
    - There are many services that are nearly free and, hence, subject to abuse
    - We desire to feedback to train systems (as per chart my Holistic Approach to Design chart)
  - Challenge: Novel approaches to reducing spam

# Architecture and Complexity

---

*Software systems score high on many metrics:*

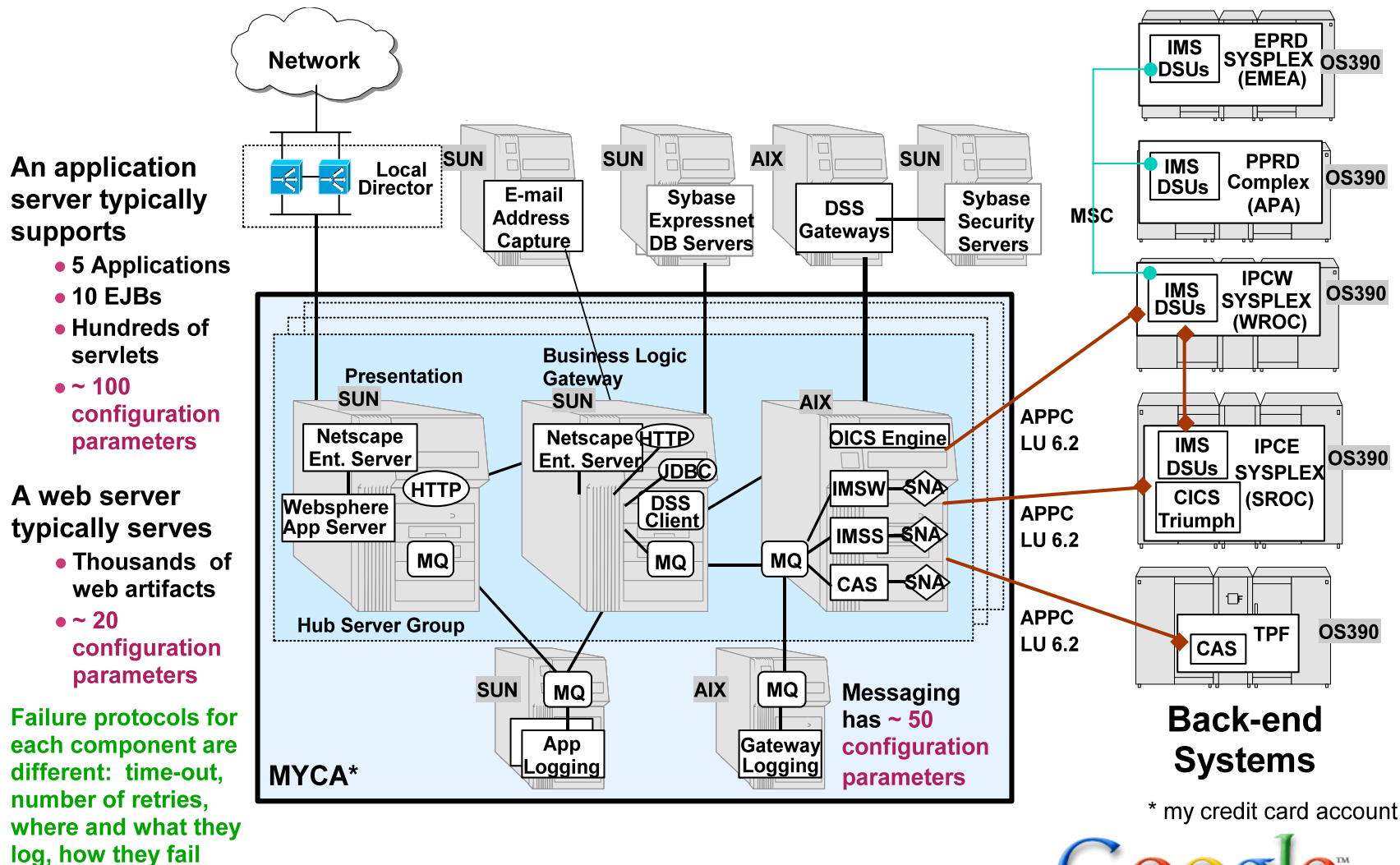
- Amount of code
- # of dependencies
- # of programmatic interfaces
- # of layers
- Administrative interface & configuration difficulty
- Non-uniformity
- Non-orthogonality
- Defects
- Documentation
- # of programmers involved

*Cloud computing helps, but doesn't solve all problems*



# Example: Credit Card Processing

(A commonly used IBM Slide)



\* my credit card account



# RosettaNet Purchase Orders

- There are **551** XML fields in the PurchaseOrderRequest
- There are **700** XML fields in the PurchaseOrderConfirmation

*Excerpted First lines of purchase order confirmation:*

|                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <u>fromRole.PartnerRoleDescription</u><br> -- <u>ContactInformation</u><br>     -- <u>contactName.FreeFormText</u><br>     -- <u>EmailAddress</u><br> -- <u>facsimileNumber.CommunicationsNumber</u><br>     -- <u>telephoneNumber.CommunicationsNumber</u><br> -- <u>GlobalPartnerRoleClassificationCode</u><br> -- <u>PartnerDescription</u><br>     -- <u>BusinessDescription</u><br>       -- <u>GlobalBusinessIdentifier</u><br>       -- <u>GlobalSupplyChainCode</u><br>     -- <u>GlobalPartnerClassificationCode</u><br><u>GlobalDocumentFunctionCode</u><br><u>PurchaseOrder</u><br> -- <u>AccountDescription</u><br>     -- <u>accountName.FreeFormText</u><br>     -- <u>AccountNumber</u><br>     -- <u>billTo.PartnerDescription</u> | -- <u>BusinessDescription</u><br>         -- <u>businessName.FreeFormText</u><br>         -- <u>GlobalBusinessIdentifier</u><br>         -- <u>PartnerBusinessIdentification</u><br>           -- <u>ProprietaryBusinessIdentifier</u><br>           -- <u>ProprietaryDomainIdentifier</u><br>           -- <u>ProprietaryIdentifierAuthority</u><br>       -- <u>ContactInformation</u><br>         -- <u>contactName.FreeFormText</u><br>         -- <u>EmailAddress</u><br>         -- <u>facsimileNumber.CommunicationsNumber</u><br>         -- <u>PhysicalLocation</u><br>           -- <u>GlobalLocationIdentifier</u><br>           -- <u>PartnerLocationIdentification</u><br>           -- <u>ProprietaryDomainIdentifier</u><br>           -- <u>ProprietaryIdentifierAuthority</u><br>      ... |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

Note: RosettaNet is a consortium of major companies working to create and implement industry-wide, open e-business process standards, that will form a common e-business language, globally aligning processes between supply chain partners. (From RosettaNet Home Page.)



# 3 Categories of Complexity

---

## • Classic Complexity • Usage Complexity

- Time
- Space
- Implementation Complexity
- Logical
- Structural
- Comprehensibility

| <i>Task</i> | Pre-Use | Novice | Middle | Expert | Exception |
|-------------|---------|--------|--------|--------|-----------|
| Install     |         |        |        |        |           |
| Configure   |         |        |        |        |           |
| Administer  |         |        |        |        |           |
| Use         |         |        |        |        |           |



# Example Cause: Design Methodology

---

## 2 related tenets of software methodology

- Generalization, Encapsulation, Re-use
  - Complex things get more complex if they support re-use
  - E.g., a packaged software product is ~10x cost of a one-of solution
- Integration of Components
  - (Don't build new, assemble and connect)
  - This exposes many more interfaces
  - Boundaries are imperfect and “don't connect”
  - Internal interfaces rarely recede into the woodwork
  - Component success may be based on context of use
  - E.g., “Business Process Integration” as a market category
- These may contribute to complexity, but it's not clear what choice there is



# The Conundrum of Systems

---

- Complexity is too great, despite all we have done in computer science, from Simon's *Sciences of the Artificial* to modern programming languages, to better software engineering, and to cloud based computing
- There is valuable, rewarding, and concrete work for Computer Science in combating complexity and improving dependability:
  - Meaning
  - Measuring
  - Methodology
  - System Architecture
- Also:
  - Acknowledgment
  - Science and Technology
- Reducing complexity could prove as valuable as direct functional innovation.
- Probably no silver bullet in the open systems world for dependable systems.
  - We need progress on multiple fronts
  - Our systems will mirror our world, and at best, approach perfection.
  - But some things in the world are pretty nice



<sup>1</sup>From: *The Conundrum of Systems*, Alfred Z. Spector, Dertouzos Lecture, MIT (then) LCS, 2/06/03

# Design

---

- Is there a science of design for software systems?
- Or are there at least re-usable patterns?
- Should there be more specialization?
- There are, after all, structural engineers that specialize in bridges
- How does one balance reuse vs bloat?
- How do you balance fault-tolerance vs. provably correct?
- Top Down vs. Bottom Up?



# Summary

---

- Fundamentals of Distributed Systems known for ~30 years
  - A fine basis and broad applicability
- To degree then understood, things have progressed as expected
- But future not fully foreseen, inducing
  - Vastly many more abstractions (and still growing)
  - Dynamic assembly over decades or centuries (?)
  - Management issues well more complex
  - General concern over complexity
- Thus, there remain great challenges in engineering global scale distributed systems
- (Google is glad you all are here working in these problems!)

## Google Relationship with Academe (cont.)

---

- Check out growing sites:
  - [//research.google.com](http://research.google.com)
  - [//research.google.com/university](http://research.google.com/university)
  - [//code.google.com](http://code.google.com)
  - [//code.google.com/edu](http://code.google.com/edu)
- Technology Round Table Videos (new!)
  - <http://research.google.com/roundtable/>

# A Few Other References:

---

## Google Cluster Infrastructure

Luiz Barroso, Jeffry Dean, Urs Hoelzle: Web Search for a Planet: The Google Cluster Infrastructure, IEEE Micro, Volume 23, Issue 2, March 2003.

## GFS (Google File System)

Sanjay Ghemawat, Howard Gobioff, Shun-Tak Leung: The Google File System, 19th ACM Symposium on Operating System Principles, Lake George, NY, October 2003.

## MapReduce Programming Model for generating & processing large data sets

Jeffrey Dean, Sanjay Ghemawat: MapReduce: Simplified Data Processing on Large Clusters, OSDI'04: Sixth Symposium on Operating System Design and Implementation, San Francisco, CA, December, 2004.

## BigTable: A Distributed Storage System for Structured Data

Chang, et al. OSDI 06.

## The Case for Energy-Proportional Computing

Luiz André Barroso, Urs Hölzle, IEEE Computer, Dec 2007.

## Type: “Google Faculty Summit 2008” to YouTube

The Conundrum of Systems Presentation: <http://www.research.ibm.com/people/a/aspector/presentations/AZSDertouzos.pdf>





Thank you very much!