

## CSE 451: Operating Systems Spring 2013

### Module 24 Distributed Systems

Ed Lazowska  
lazowska@cs.washington.edu  
Allen Center 570

© 2013 Gribble, Lazowska, Levy, Zahorjan

### What is a “distributed system”?

- Nearly all systems today are distributed in some way
  - they use email
  - they access files over a network
  - they access printers over a network
  - they’re backed up over a network
  - they share other physical or logical resources
  - they cooperate with other people on other machines
  - they access the web
  - they receive video, audio, etc.

© 2013 Gribble, Lazowska, Levy, Zahorjan

2

### Loosely-coupled systems

- Earliest systems used simple explicit network programs
  - FTP (rcp): file transfer program
  - telnet (rlogin/rsh): remote login program
  - mail (SMTP)
- Each system was a completely autonomous independent system, connected to others on the network

© 2013 Gribble, Lazowska, Levy, Zahorjan

3

- Even today, most distributed systems are loosely-coupled (although not that loosely!):
  - each CPU runs an independent autonomous OS
  - computers don’t really trust each other
  - some resources are shared, but most are not
  - the system may look differently from different hosts

© 2013 Gribble, Lazowska, Levy, Zahorjan

4

### Closely-coupled systems

- A distributed system becomes more “closely-coupled” as it
  - appears more uniform in nature
  - runs a “single” operating system
  - has a single security domain
  - shares all logical resources (e.g., files)
  - shares all physical resources (CPUs, memory, disks, printers, etc.)
- In the limit, a distributed system looks to the user as if it were a centralized timesharing system, except that it’s constructed out of a distributed collection of hardware and software components

© 2013 Gribble, Lazowska, Levy, Zahorjan

5

### Tightly-coupled systems

- A “tightly-coupled” system usually refers to a multiprocessor
  - runs a single copy of the OS with a single workload queue
  - has a single address space
  - usually has a single bus or backplane to which all processors and memories are connected
  - has very low communication latency
  - processors communicate through shared memory

© 2013 Gribble, Lazowska, Levy, Zahorjan

6

## Some issues in distributed systems

- Availability (can you get it now?)
- Reliability (may you lose stuff permanently?)
- Performance (on a LAN, on a WAN, global)
- Scalability (can it grow modularly as you add users, without relying on ever-bigger/faster computers?)
- Transparency (how visible is the distribution to users?)
- Programming models (how visible is the distribution to programmers?)
- Communication models (messages, RPC, etc.)
- Security

© 2013 Gribble, Lazowska, Levy, Zahorjan

7

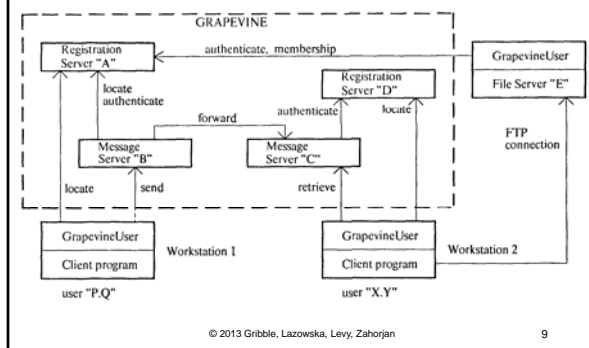
## Example: Grapevine distributed mail service

- Xerox PARC, 1980
  - cf. Microsoft Outlook/Exchange today!!!!
- Goals
  - cannot rely on integrity of client
  - once the system accepts mail, it will be delivered
  - no single Grapevine computer failure will make the system unavailable to any client either for sending or for receiving mail
- Components
  - GrapevineUser package on each client workstation
  - Registration Servers
  - Message Servers
- Implementation: Remote Procedure Call

© 2013 Gribble, Lazowska, Levy, Zahorjan

8

## Grapevine: Functional diagram



© 2013 Gribble, Lazowska, Levy, Zahorjan

9

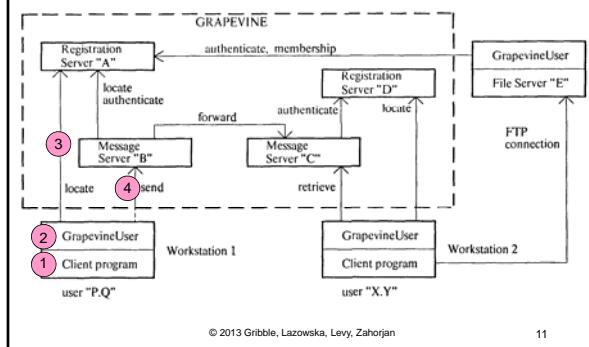
## Grapevine: Sending a message

- User prepares message using mail client
- Mail client contacts GrapevineUser package on same workstation to actually send message
- GrapevineUser package
  - Contacts any Registration Server to get a list of Message Servers
  - Contacts any Message Server to transmit message
    - presents source and destination userids, and source password, for authentication
    - Message Server uses any Registration Server to authenticate
  - sends message body to Message Server
    - Message Server places it in stable storage and acknowledges receipt

© 2013 Gribble, Lazowska, Levy, Zahorjan

10

## Grapevine: Functional diagram



© 2013 Gribble, Lazowska, Levy, Zahorjan

11

## Registries

- Actually, I lied: There's an additional step.
  - For scalability, users are partitioned into registries – "user 'P.Q'" is user P in registry Q.
  - Registries are replicated.
  - There is one registry that is replicated on every registration server: the registry of registries.
  - So, when I said:
    - Message Server uses any Registration Server to authenticate
 what actually happens is the Message Server contacts any Registration Server to obtain a list of those Registration Servers holding the registry of the user, then contacts one of those registration servers to authenticate the user

© 2013 Gribble, Lazowska, Levy, Zahorjan

12

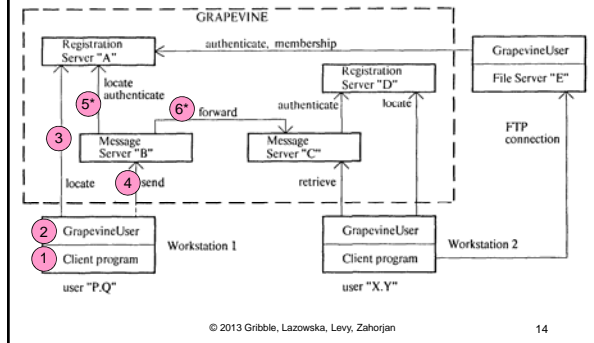
## Grapevine: Transport and buffering

- For each recipient of the message, Message Server contacts any Registration Server to obtain list of Message Servers holding mail for that recipient
  - Same lie as before
- Sends a copy of the message to one of those Message Servers for that recipient

© 2013 Gribble, Lazowska, Levy, Zahorjan

13

## Grapevine: Functional diagram



© 2013 Gribble, Lazowska, Levy, Zahorjan

14

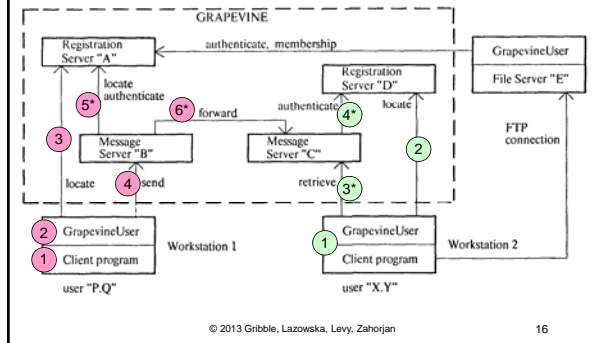
## Grapevine: Retrieving mail

- User uses mail client to contact GrapevineUser package on same workstation to retrieve mail
- GrapevineUser package
  - Contacts any Registration Server to get a list of each Message Server holding mail for the user ("inbox site")
    - Same lie as before
  - Contacts each of these Message Servers to retrieve mail
    - presents user credentials
      - Message Server uses any Registration Server to authenticate
    - acknowledges receipt of messages so that the server can delete them from its storage

© 2013 Gribble, Lazowska, Levy, Zahorjan

15

## Grapevine: Functional diagram



© 2013 Gribble, Lazowska, Levy, Zahorjan

16

## Grapevine: Scalability

- Can add more Registration Servers
- Can add more Message Servers
- Only thing that didn't scale was handling of distribution lists
  - the accepting Message Server was responsible for expanding the list (recursively if necessary) and delivering to an appropriate Message Server for each recipient
  - some distribution lists contained essentially the entire user community
- Jeff Dean (Google) told us they don't even think about more than two decimal orders of magnitude
  - fundamental design decisions will need to change
  - advances in technology will make it possible

© 2013 Gribble, Lazowska, Levy, Zahorjan

17

## Example: Google search infrastructure

- It's likely that Google has several million machines
  - But let's be conservative – 1,000,000 machines
  - A rack holds 176 CPUs (88 1U dual-processor boards), so that's about 6,000 racks
  - A rack requires about 50 square feet (given datacenter cooling capabilities), so that's about 300,000 square feet of machine room space (more than 6 football fields of real estate – although of course Google divides its machines among dozens of datacenters all over the world)
  - A rack requires about 10kw to power, and about the same to cool, so that's about 120,000 kw of power, or nearly 100,000,000 kwh per month (\$10 million at \$0.10/kwh)
    - Equivalent to about 20% of Seattle City Light's generating capacity

© 2013 Gribble, Lazowska, Levy, Zahorjan

18

- There are multiple clusters (of thousands of computers each) all over the world
- *Many hundreds of machines are involved in a single Google search request* (remember, the web is 400+TB)

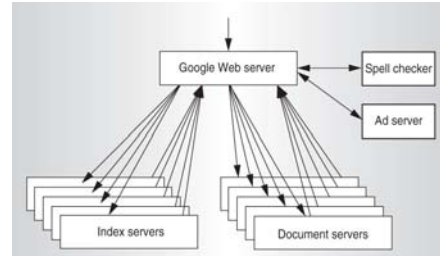
1. DNS routes your search request to a nearby cluster



© 2013 Gribble, Lazowska, Levy, Zahorjan

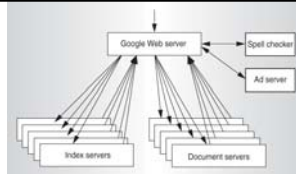
19

- A cluster consists of Google Web Servers, Index Servers, Doc Servers, and various other servers (ads, spell checking, etc.)
  - These are cheap standalone computers, rack-mounted, connected by commodity networking gear



© 2013 Gribble, Lazowska, Levy, Zahorjan

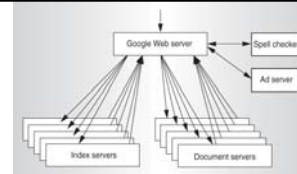
20



2. Within the cluster, load-balancing routes your search to a lightly-loaded Google Web Server (GWS), which will coordinate the search and response

© 2013 Gribble, Lazowska, Levy, Zahorjan

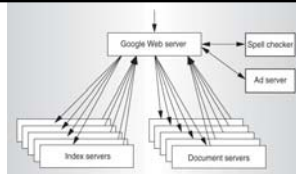
21



- The index is partitioned into “shards.” Each shard indexes a subset of the docs (web pages). Each shard is replicated, and can be searched by multiple computers – “index servers”
- 3. The GWS routes your search to one index server associated with each shard, through another load-balancer
- 4. When the dust has settled, the result is an ID for every doc satisfying your search, rank-ordered by relevance

© 2013 Gribble, Lazowska, Levy, Zahorjan

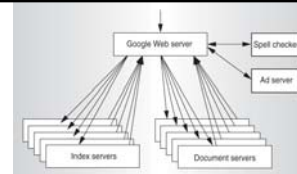
22



- The docs, too, are partitioned into “shards” – the partitioning is a hash on the doc ID. Each shard contains the full text of a subset of the docs. Each shard can be searched by multiple computers – “doc servers”
- 5. The GWS sends appropriate doc IDs to one doc server associated with each relevant shard
- 6. When the dust has settled, the result is a URL, a title, and a summary for every relevant doc

© 2013 Gribble, Lazowska, Levy, Zahorjan

23



- 7. Meanwhile, the ad server has done its thing, the spell checker has done its thing, etc.
- 8. The GWS builds an HTTP response to your search and ships it off
- Many hundreds of computers have enabled you to search 400+TB of web in ~100 ms.

© 2013 Gribble, Lazowska, Levy, Zahorjan

24

## Google: The Big Picture

- Enormous volumes of data
- Extreme parallelism
- The cheapest imaginable components
  - Failures occur all the time
  - You couldn't afford to prevent this in hardware
- Software makes it
  - Fault-Tolerant
  - Highly Available
  - Recoverable
  - Consistent
  - Scalable
  - Predictable
  - Secure

© 2013 Gribble, Lazowska, Levy, Zahorjan

25

## How on earth would you enable mere mortals write hairy applications such as this?

- Recognize that many Google applications have the same structure
  - Apply a "map" operation to each logical record in order to compute a set of intermediate key/value pairs
  - Apply a "reduce" operation to all the values that share the same key in order to combine the derived data appropriately
- Build a runtime library that handles all the details, accepting a couple of customization functions from the user – a Map function and a Reduce function
- That's what MapReduce is
  - Supported by the Google File System and the Chubby lock manager
  - Augmented by the BigTable not-quite-a-database system

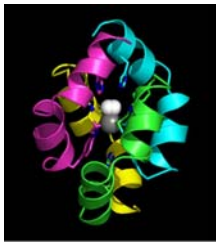
© 2013 Gribble, Lazowska, Levy, Zahorjan

26

## An extremely loosely coupled system: BOINC



David Baker



© 2013 Gribble, Lazowska, Levy, Zahorjan

27



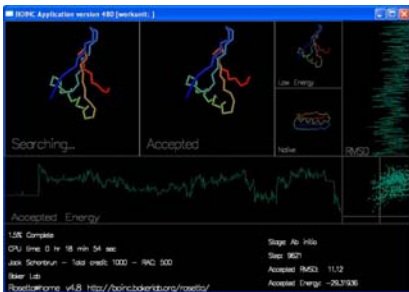
© 2013 Gribble, Lazowska, Levy, Zahorjan

28

## Rosetta@home



Protein Folding, Design, and Docking



© 2013 Gribble, Lazowska, Levy, Zahorjan

29



Berkeley Open Infrastructure for Network Computing



© 2013 Gribble, Lazowska, Levy, Zahorjan

30

## Totally off the subject of OS: Human Computation

David Baker and Zoran Popovic

© 2013 Gribble, Lazowska, Levy, Zahorjan

31

© 2013 Gribble, Lazowska, Levy, Zahorjan

32



Luis von Ahn

- Humans and computers have different computational strengths
- Can we exploit these differences?
  - To differentiate computers from humans?
    - E.g., to make it harder for spambots to acquire new email accounts from which to send spam
  - To create human/machine computational systems that combine the best of each?

© 2013 Gribble, Lazowska, Levy, Zahorjan

33

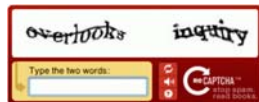
Hours per year, world-wide, spent playing computer solitaire: 9 billion

Hours spent building the Panama Canal: 20 million (less than a day of solitaire)

## Where do the words come from?

### The New York Times

Entire photo archive (years 1851-1980)  
was completed in 2009

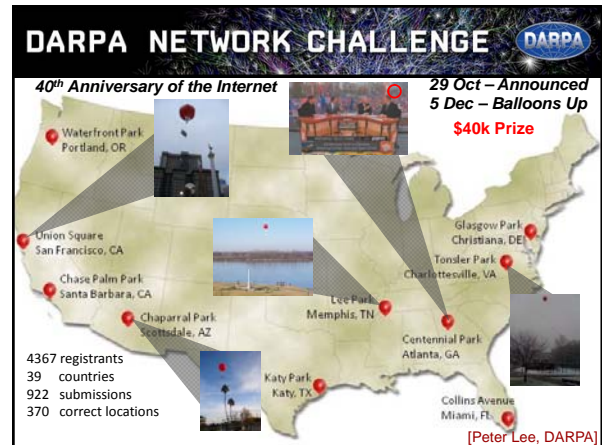
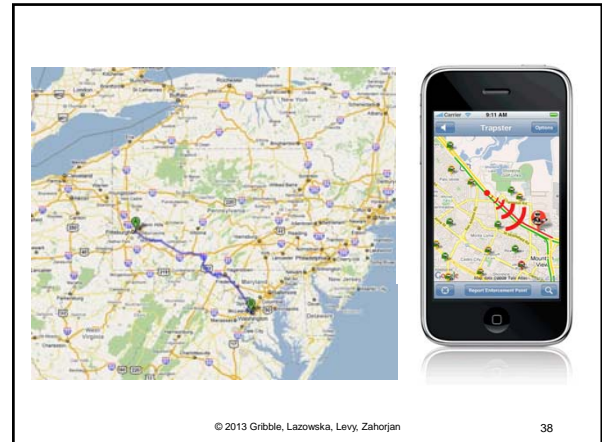


© 2013 Gribble, Lazowska, Levy, Zahorjan

35

© 2013 Gribble, Lazowska, Levy, Zahorjan

36



### Some issues in distributed systems

- Availability (can you get it now?)
- Reliability (may you lose stuff permanently?)
- Performance (on a LAN, on a WAN, global)
- Scalability (can it grow modularly as you add users, without relying on ever-bigger/faster computers?)
- Transparency (how visible is the distribution to users?)
- Programming models (how visible is the distribution to programmers?)
- Communication models (messages, RPC, etc.)
- Security

© 2013 Gribble, Lazowska, Levy, Zahorjan 41