Table of Contents

- Introduction and Motivation
- Theoretical Foundations
- Distributed Programming Languages
- Distributed Operating Systems
- Distributed Communication
- Distributed Data Management
- Reliability
- Applications
- Conclusions
- Appendix

Distributed Operating Systems

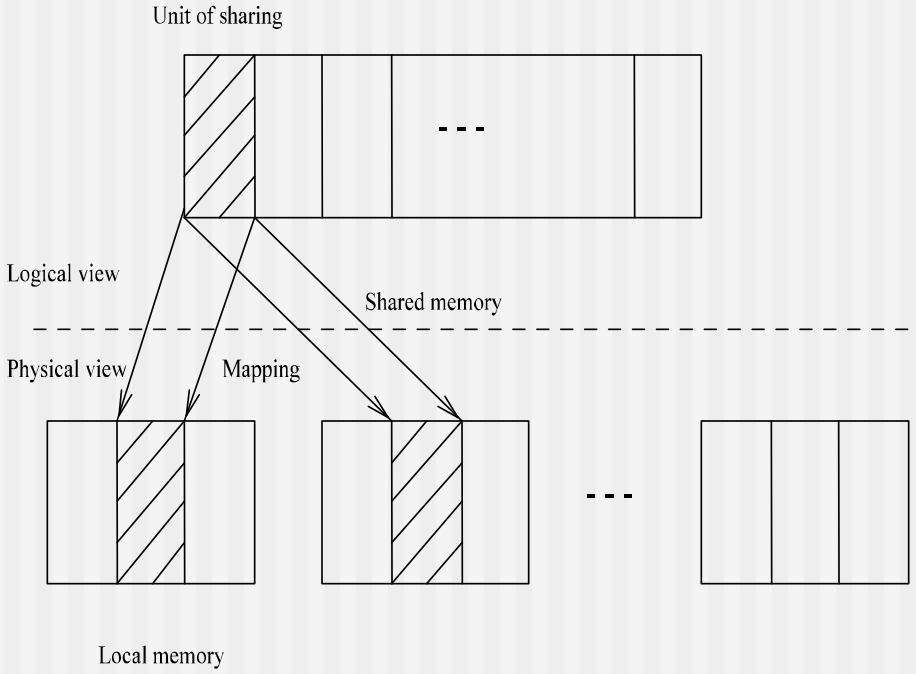
- Key issues
  - Communication primitives
  - Naming and protection
  - Resource management
  - Fault tolerance
  - Services: file service, print service, process service, terminal service, file service, mail service, boot service, gateway service
- **Distributed operating systems** vs. **network operating systems**
- Commercial and research prototypes
  - Commercial: cloud virtualization (VM and container)
    cloud orchestration (Kubernetes)
  - Research: Wiselow, Galaxy, Amoeba, and Mach

Distributed File Systems

- A **file system** is a subsystem of an operating system whose purpose is to provide long-term storage.
- **Main issues**:
  - Merge of file systems
  - Protection
  - Naming and name service
  - Caching
  - Writing policy
- **Research prototypes**:
  - UNIX United, Coda, Andrew (AFS), Frangipani, Sprite, Plan 9, DCE/DFS, and XFS
- **Commercial:**
  - Amazon S3, Google Cloud Storage, Microsoft Azure, SWIFT (OpenStack)

Distributed Shared Memory

A **distributed shared memory** is a shared memory abstraction what is implemented on a loosely coupled system.



Distributed shared memory.

Focus 24: Stumm and Zhou's Classification

- **Central-server algorithm** (nonmigrating and nonreplicated): central server
  - (Client) Sends a data request to the central server.
  - (Central server) Receives the request, performs data access and sends a response.
  - (Client) Receives the response.

Focus 24 (Cont'd)

- **Migration algorithm** (migrating and non-replicated): single-read/single-write
  - (Client) If the needed data object is not local, determines the location and then sends a request.
  - (Remote host) Receives the request and then sends the object.
  - (Client) Receives the response and then accesses the data object (read and /or write).

Focus 24 (Cont'd)

- **Read-replication algorithm** (migrating and replicated): multiple-read/single-write
  - (Client) If the needed data object is not local, determines the location and sends a request.
  - (Remote host) Receives the request and then sends the object.

Focus 24 (Cont'd)

- (Client) Receives the object and accesses the data object
- (Client for write operation only) then multicasts by sending either invalidate or update messages to all sites that have a copy of the data object.
- (Remote host for write operation only) Receives an invalidation signal and then invalidates its local copy, or receives an update signal and then updates the local copy.

Focus 24 (Cont'd)

- **Full-replication algorithm** (non-migrating and replicated): multiple-read/multiple-write using a global sequencer
  - (Client) If it is a write, sends a signal to the sequencer.
  - (Sequencer) Receives the signal and adds a sequence number. Sends the client a signal with the sequence number.

Focus 24 (Cont'd)

- (Client) Receives the sequence number, multicast the data object together with the sequence number and updates the local memory based on the sequence number of each data object.
  (Other sites) Receive the data object and update local memory based on the sequence number of each data object.

Another option is for the sequencer to multicast directly data object with a sequence number to clients upon receiving a request and data.

DSM Implementation (Li and Hudak)

Read-replication with coherence enforced by either invalidation or write-through (expensive)

Centralized manager:

- The *owner* field points to the most recent processor to have write access
- The *copy* set field lists all processors that have copies
- The *lock* field is used for synchronizing requests

Distributed manager:

- Using a hash function to map request to different managers

DSM Implementation (cont'd)

Dynamic distributed manager:

- Keep track of the ownership of all pages in the processor's local pagetable.
- In pagetable, *owner* field is replaced by *probowner*.
- The probowner field forms a chain from node to node that eventually leads to the real owner.
- The average length of a probable owner chain is 2 or 3.
- The chain can be made fault tolerant by maintaining two link-disjoint chain*

* L. Brown and J. Wu, "Snooping fault-tolerant distributed shared memories," Proc. of IEEE ICDCS, 1994.

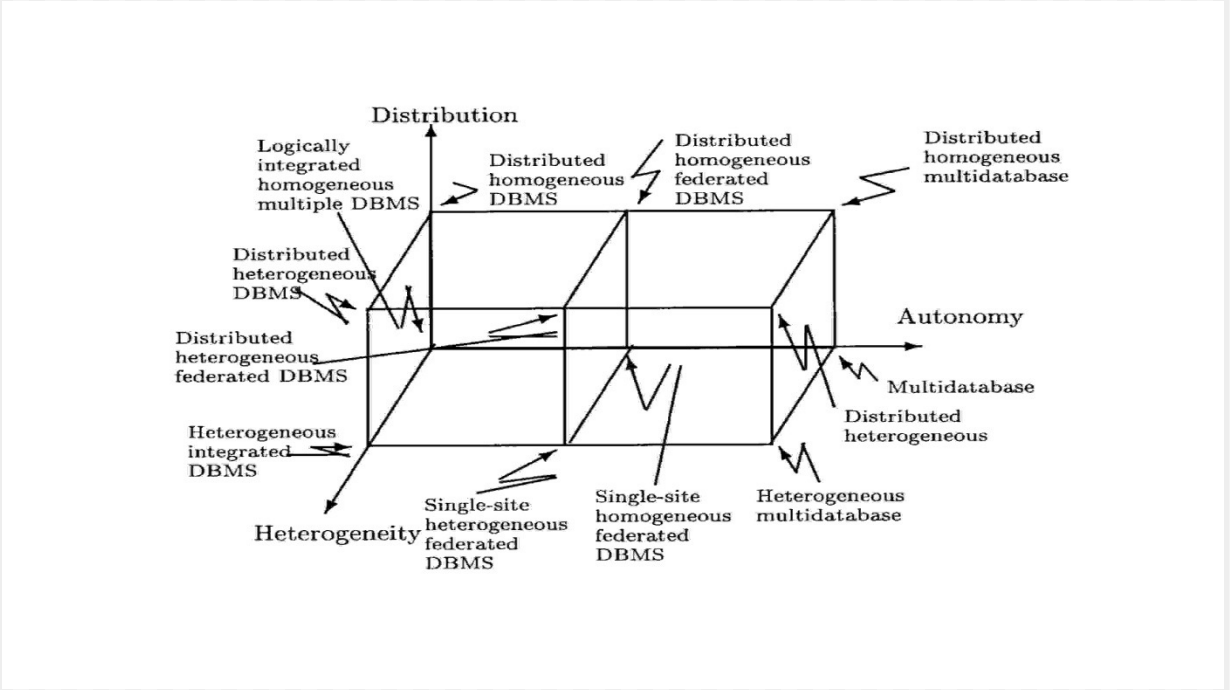
Focus 24 (Cont'd)

- **Main Issues**:
  - Structure and granularity
  - Coherence semantics
  - Scalability
  - Heterogeneity
- **Several research prototypes**:
  - Dash, Ivy, Munin, and Clouds
  - Alewife (MIT), Treadmarks (Rice), Coherent Virtual machine (Maryland), and Millipede (Israel Inst. Tech.)

Distributed Database Systems

A **distributed database** is a collection of multiple, logically interrelated databases distributed over a computer network.

- **Possible design alternatives**:
  - Autonomy
  - Distribution
  - Heterogeneity



Essentials of Distributed Database Systems

- Local autonomy
- No reliance on a central site
- Continuous operation
- Location independence
- Fragment independence
- Replication independence
- Distributed query processing
- Distributed transaction management
- Hardware independence
- Operating system independence
- Network independence
- Data independence

Open Research Problems

- Network scaling problem
- Distributed query processing
- Integration with distributed operating systems
- Heterogeneity
- Concurrency control
- Security
- Next-generation database systems:
  - Object-oriented database management systems
  - Knowledge base management systems

Prototypes and Products

- Research Prototypes
  - ADDS (Amocha Distributed Database Systems)
  - JDDBS (Japanese Distributed Database Systems)
  - Ingres/star
  - SWIFT (Society for Worldwide Interbank Financial Telecomm)
  - System R, MYRIAD, MULTIBASE, and MERMAID
- Commercial products (XML, NewSQL, and NoSQL)
  - Blockchain (popularized by bitcoin)
  - Aerispike, Cassandra, Clusterpoint, Druid (open-source data store)
  - ArangoDB, ClustrixDB, Couchbase, FoundationDB, NueDB, and OrientDB

Heterogeneous Processing

- Tuned use of diverse processing hardware to meet distinct computational needs.
  - **Mixed-machine systems**. Different execution modes by inter-connecting several different machine models.
  - **Mixed-mode systems**. Different execution modes by reconfigurable parallel architecture obtained by interconnecting the same processors.

Classifications

- Single Execution Mode/Single Machine Model (SESM)
- Single Execution Mode/Multiple Machine Models (SEMM)
- Multiple Execution Modes/Single Machine Model (MESM)
- Multiple Execution Modes/Multiple Machine Models (MEMM)

Focus 25: Optimization

An optimization problem that minimizes

$$\sum t_{i,j}$$

such that

$$\sum c_j \leq C$$

where $t_{i,j}$ equals the time for machine i on code segment j, c_i equals the cost for machine i, C equals the specified overall cost constraint.

Table of Contents

- Introduction and Motivation
- Theoretical Foundations
- Distributed Programming Languages
- Distributed Operating Systems
- Distributed Communication
- Distributed Data Management
- Reliability
- Applications
- Conclusions
- Appendix

"Killer" Applications

- Distributed Object-based Systems: CORBA
- Distributed Document-based Systems: WWW
- Distributed Coordination-based Systems: JINI
- Distributed Multimedia Systems: QoS requirements
- Distributed Currency and Transactions: Bitcoin and blockchain

Other Applications: MapReduce

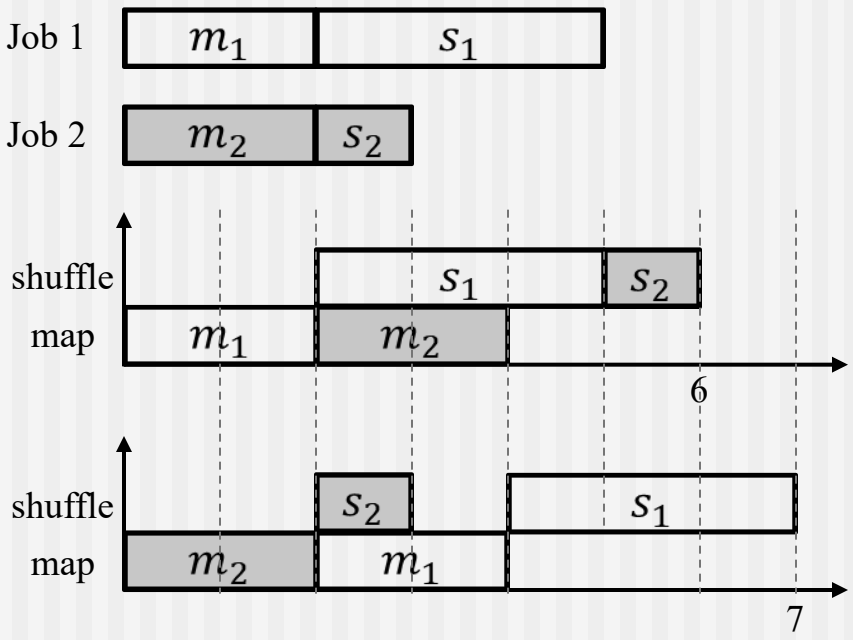
- A framework for processing highly distributable problems across huge datasets on a file system or a database.
  - Map: In a recursive way, the master node takes the input, divides it into smaller sub-problems, and distributes them to worker nodes.
  - Reduce: The master node then collects the answers to all the sub-problems and combines them in some way to form the output. The shuffle is used collect all data through I/O. Usually, shuffle dominates the actual reduce phase.
- Apache Hadoop: a software framework that supports data-intensive distributed applications under a free license. It was derived from Google's MapReduce and Google File System (GFS).
- Spark for Big Data and beyond

Minimize Last Job: Flow Shop

Minimize last job completion time with two-stage pipeline (with no overlapping between two stages)

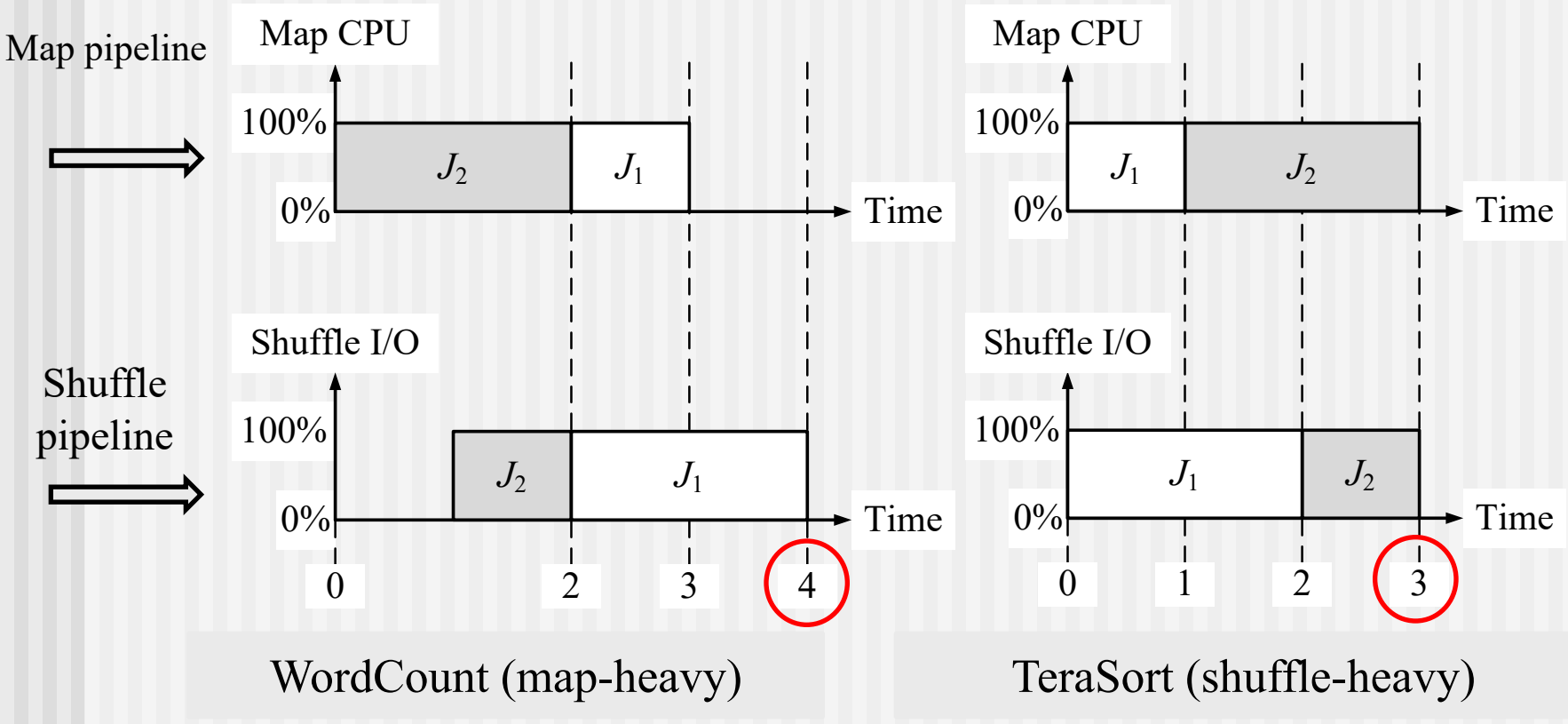
Johnson's rule:

- Select a job with shortest time in one stage.
  - Run the job first if it is in the first stage
  - Run the job last if it is in the second stage
- Remove the job and repeat the above process



Pipeline: Map followed by Shuffle

Impact of overlapping map and shuffle: Johnson's rule still hold.



Minimize Average Jobs: Strong Pair

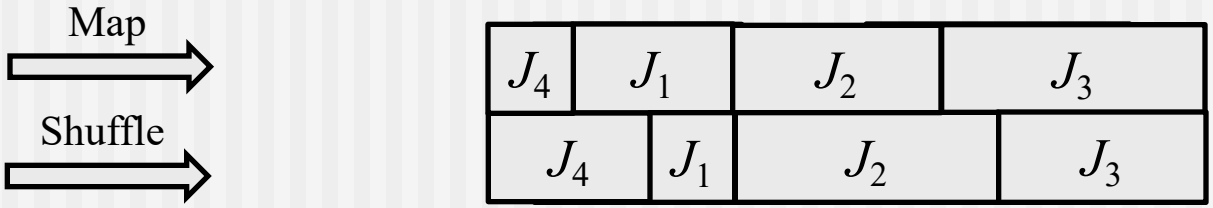
Minimize average job completion time, NP-hard in general

Special case one: strong pair

- J_1 and J_2 are a strong pair if $m_1 = s_2$ and $s_1 = m_2$ (m: map, s: shuffle)

Optimal schedule: jobs are strong pairs

Pair jobs and rank pairs by total workloads in a non-decreasing order



1 and 4 form a pair and 2 and 3 form another pair.

H. Zheng, Z. Wan, and J. Wu, Optimizing MapReduce framework through joint scheduling of overlapping phases, *Proc. of IEEE ICCCN*, 2016.

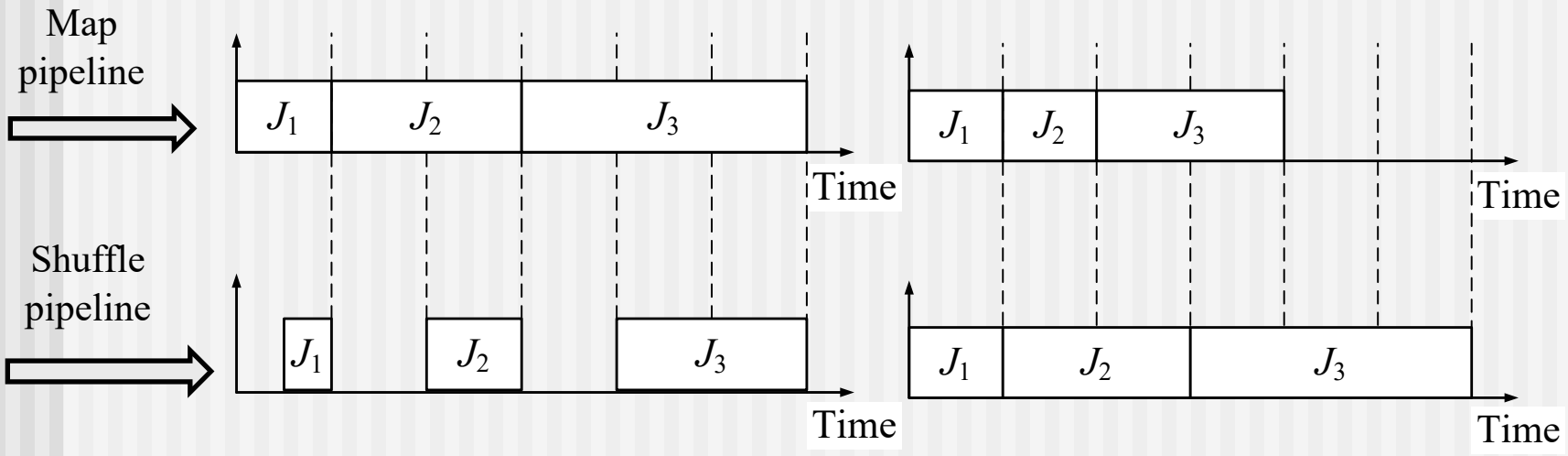
Minimize Average Jobs: Dominant Load

Special case two: when all jobs are map-heavy or shuffle-heavy

Optimal schedule:

Sort jobs ascendingly by dominant workload max{m, s}

Execute smaller jobs first



Finishing times J_1, J_2, J_3: 1, 3, 6 vs. J_3, J_2, J_1: 3, 5, 6

Other Applications: Crowdsourcing

How Much Data?

- Facebook: 40 B photos; 30 B pieces of content shared every month
- WeChat: 846 M users and 20 B message per day
- Global Internet traffic: quadrupled from 2010 to 2015, reaching 966 EB (10^{18}) per year

  (All human knowledge created from the dawn of man to 2003 is totaled 5 EB)

Big Data Era

- "In information technology, big data consists of datasets that grow so large that they become awkward to work with using on-hand database management tools."
- Computers are not efficient in processing or creating certain things: pattern recognition, complex communication, and ideation.

- Crowdsourcing: coordinating a crowd (a large group of people online) to do microwork (small jobs) that solves problems (that software or one user cannot easily do)
- Crowdsourcing: crowd + outsourcing (through Internet)

The Benefits of Crowdsourcing

- Performance
  - Inexpensive and fast
  - The whole is greater than the sum of its parts
- Human Processing Unit (HPU)
  - More effective than CPU (for some apps)
    - Verification and validation: Image labeling
    - Interpretation and analysis: language translation
    - Surveys: Social network survey
  - Future: mix of HPU and CPU
- High adoption in business (85% of the top global brands) based on eYeka

Basic Components of Crowdsourcing

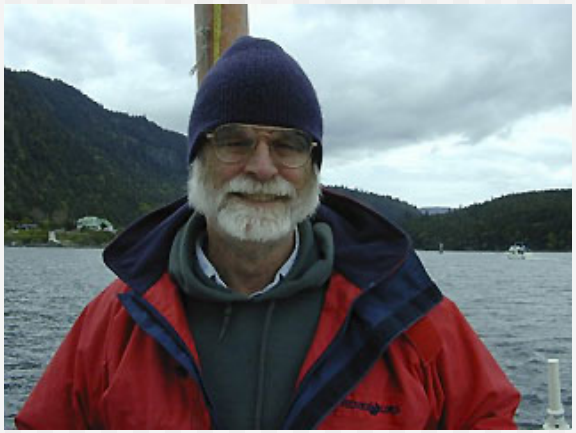
- Requester
  - People submit jobs (microwork)
  - Human Intelligence Tasks (HITs)
- Worker
  - People work on jobs
- Platform
  - Job management

Amazon Mechanical Turk (MTurk): 18th century chess playing robot with a human inside

Requester

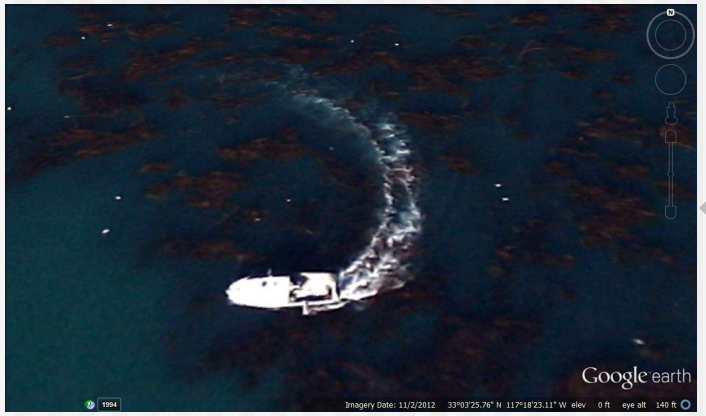
Worker Pool

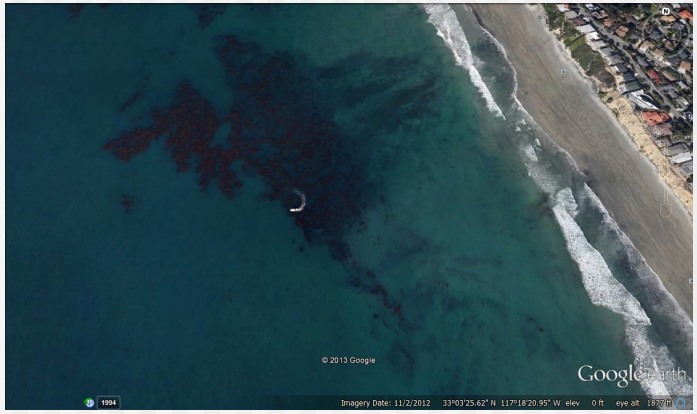
Help Find Jim Gray



- Jim Gray, Turing Award winner and inventor of 2PC, went missing with his sailboat outside San Francisco Bay in January 2007.

- Use satellite image to search for his sailboat.





Malaysia Airlines Flight MH 370

- DigitalGlobe
  - Crowdsourcing volunteers combed through satellite photos for Malaysia Airlines jet
- March 11, 2014 (from CSU prof. email)

I just saw on our local Denver Fox news (KDVR.com) that a local company, DigitalGlobe, has reoriented their satellites to take high-res images in the area where the plane may have crashed. Crowdsourcing efforts are on to have people scan these images and find signs of debris. I was reminded of Jie Wu's talk earlier this month.

DARPA Network Challenges

- Problem (2009): $40,000 challenge award for the first team to find 10 balloons.
- MIT team won under 9 hours.
- Winning strategy
  - $2,000 per balloon to the first person to send the correct location
  - $1,000 to the person who invited the winner
  - $500 to whoever invited the inviter
  - … (or to charity) …

Tag Challenges

- Problem (March 31, 2012): Find five suspects in Washington, D.C., New York, London, Stockholm, and Bratislava.
- Winner from UCSD CrowdScanner: located 3 of the 5 suspects.
- Winner's strategy: same as MIT. Also, recruiters of the first 2,000 eat gets $1.

Washington DC

New York City

Bratislava

Smarter Than You Think

- Who is smarter
  - Human (HPU) or computer (CPU)?
- AI will redefine
  - What it means to be human
  - ChartGPT

- 1997 (Chess)
  - Kasparov vs. Deep Blue
- 1998
  - Kasparov vs. Topalov: 4:0
  - Kasparov + machine vs. Topalov + machine: 3:3
- 2005 (freestyle tournament)
  - Grand-master (>2,500)
  - Machine (Hydra)
  - Grand-master + machine
  - Amateurs (>1,500) + machine *
- 2016 (Go game)
  - AlphaGo vs. Lee Sedol: 4:1
  - AlphaGo vs. Jie Ke: 3:0 (May 2017)

Other Applications: PageRank

A link analysis algorithm

- PR(E): Rage Rank of E
- likelihood that a person randomly clicking on links will arrive at any particular page

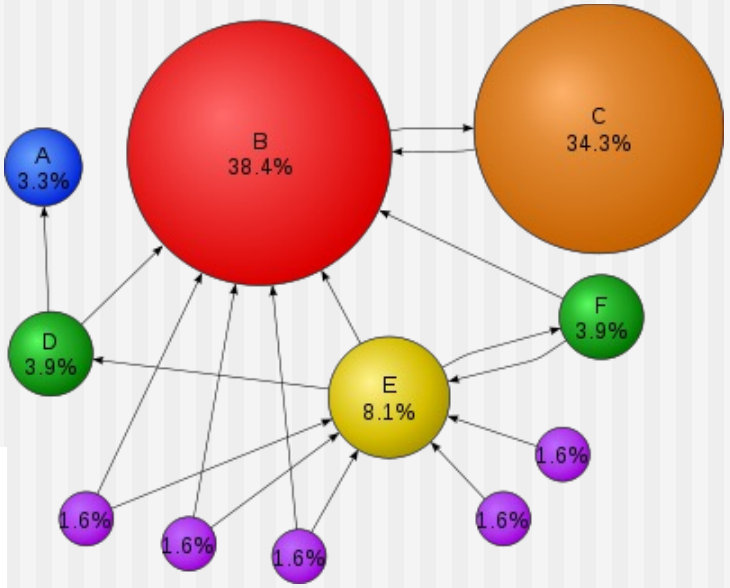
$$PR(p_i) = \frac{1-d}{N} + d \sum_{p_j \in M(p_i)} \frac{PR(p_j)}{L(p_j)}$$

$M(p_j)$: set of pages link to p_i,

$L(p_j)$: the number of outgoing links on p_j,

d: dumping factor

N: total number of pages.



HITS (Jon Kleinberg): each node has two values in a mutual recursion

- Authority: the sum of the *Hub Scores* of each node that points to it.
- Hub: the sum of the *Authority Scores* of each node that it points to.

Other Applications: P2P Systems

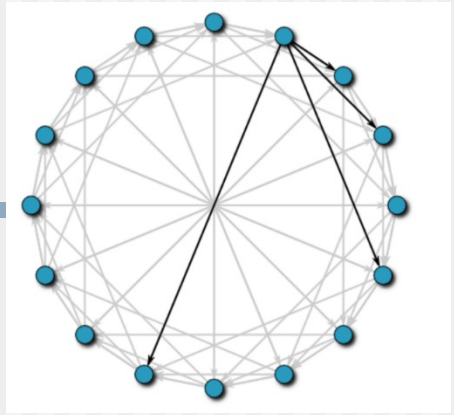
- Central server limitations: scalability, single point of failure, etc.
- P2P network
  - An overlay network no centralized control, e.g., Blockchain
  - It is dynamic: nodes join and leave
- Unstructured P2P
  - Napster: share music, server stores index (legal issues)
  - Gnutella: no server, query flooding through TTL control
  - Kazza: supernodes to improve scalability over Napster
  - BitTorrent: Tit-for-Tat to avoid *free-raiders*
- Structured P2P: *distributed hash table* (DHT): map key to value
  - Chord: n-node ring, table size: log(n), search time: log(n)
  - CAN: n-node d-dimension mesh, table size: d, search time: d $n^{1/d}$
  - Others based on different graphs: De Brujin, Butterfly, and Kautz graphs

P2P Systems: Search

- Unstructured P2P
  - BFS and variations (e.g. expanding rings and directed BFS)
  - Probabilistic search and variations (e.g., random walk)
  - Indices-based search and variations (e.g., dominating-set and Bloom filter)
- Structured P2P
  - Based on a regular network topology
- An Overview of Structured P2P Overlay Networks

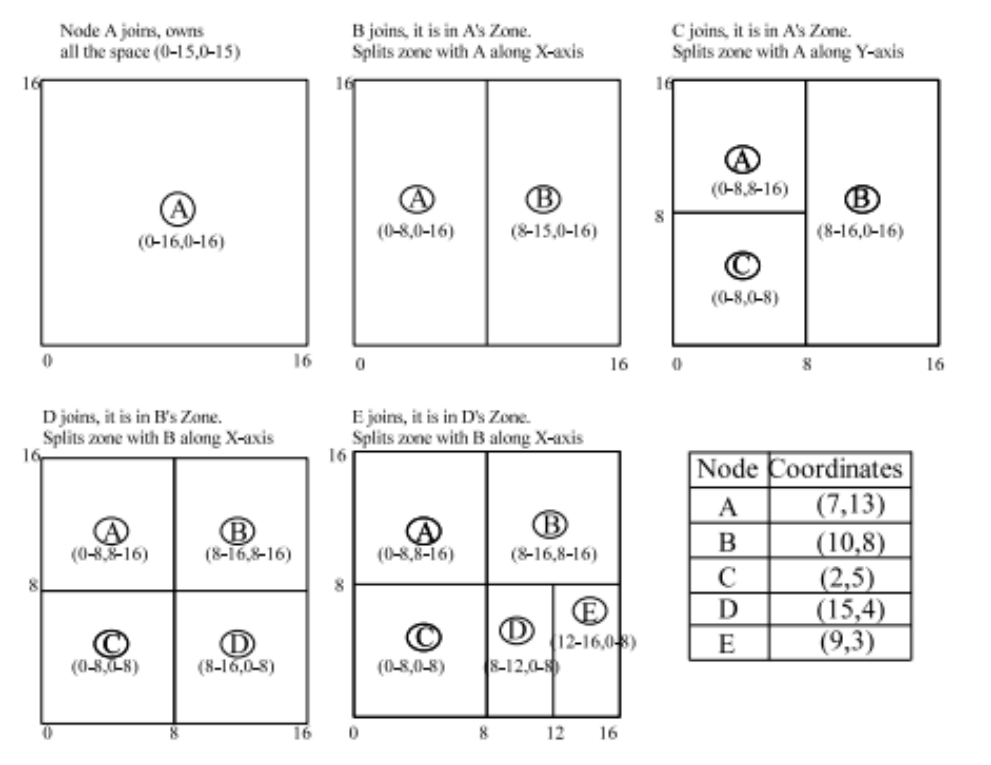
  (S. El-Ansary and S. Haridi)

Structured P2P: Chord



- A *circular identifier space* of size n
- A node with ID u has a pointer to the first node following it clockwise on the ID space (Succ(u)) as well as the first node preceding it (Pred(u))
- In addition, each node keeps log n pointers called *fingers*. Its structure is exactly like a hypercube.
- Graceful node *joins* and *leaves* are done like insertion/deletion in a linked list using Succ ad Pred information

CAN



- CAN (2-D mesh, with *adjacent blocks* (squares or rectangles) split (because a node join) or merge (a node leave)
- Zones are split along the x axis first then along the y axis for a new node to join. Upon a split, the new node learns its neighbors from the previous owner. Neighbors of the new node are neighbors of the previous owner plus the owner.
- The leave process is the reverse, a node informs its neighbors of its leaving and merges its zone with a neighbor to produce a valid zone.

Emerging Systems

- **Wireless networks** and **mobile computing: mobile agents**
  - Move the computation to the data rather than the data to the computations.
- **Grid**
  - TeraGrid: 13.6 teraflops of Linux Cluster computing power distributed at the four TeraGrid sites.
  - Open Grid Services Architecture (OGSA): delivering tighter integration between grid computing networks and Web services technologies.
  - Grid Computing (via OGSA) as the basis of evolution of Internet computing in the future.

Distributed Grid

TeraGrid (U. of Chicago, funded by NSF, 2004-2011)

- Integrated high-performance computers, data resources, and tools
- A petaflops (10^{15}) of computing capability
- 30 petaflops of online an d archival data storage
- 100 discipline-specific database
- Eleven partner sites

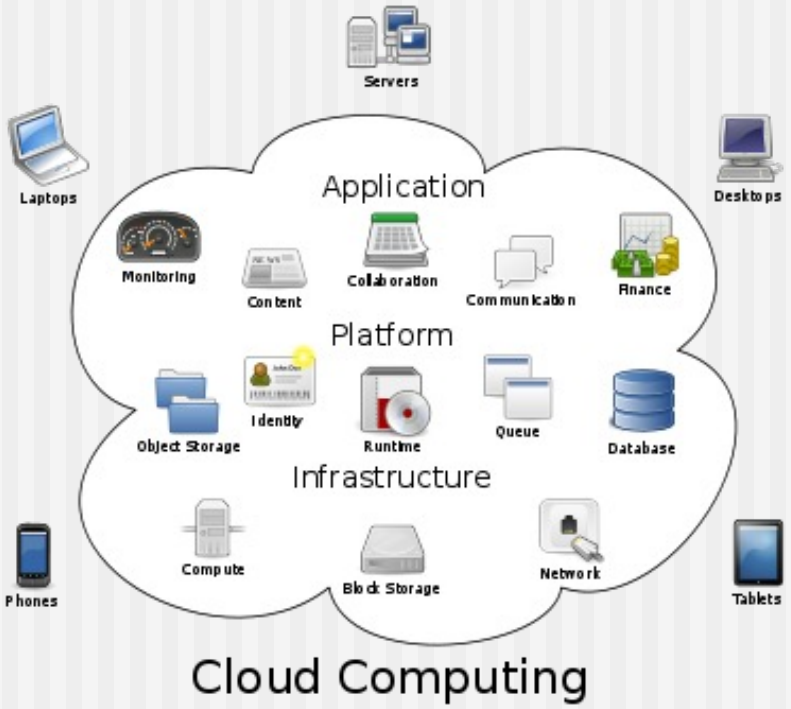
E-Science (UK Research Councils, 2001)

- Large-scale science carried out through distributed global collaboration enabled by networks, requiring access to very large data collaborations, very large-scale computing resources, and high-performance visualization.

Cloud (edge, fog) Computing

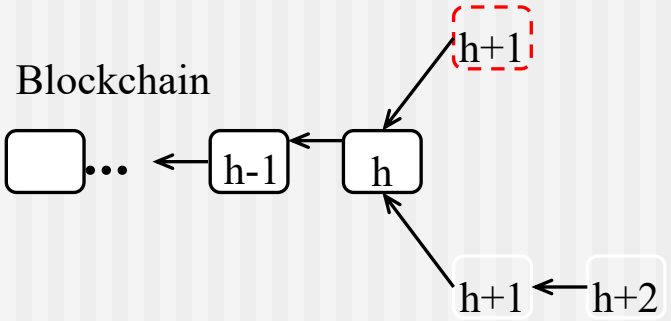
Sharing of resources to achieve coherence and economies of scale similar to utility (e.g. electricity grid) over a network (e.g. Internet)

- Characteristics
  - Agility, API, cost, device, virtualization, multi-tenancy, reliability, scalability, performance, security, maintenance
- Service
  - Infrastructure as a Service (LaaS)
  - Platform as a Service (PaaS)
  - Software as a Service (SaaS)

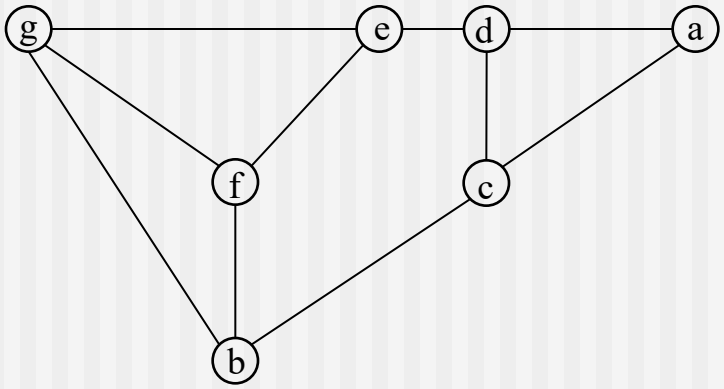


Blockchain: Probabilistic Consistency

- Network is P2P with each node having up to 125 neighbors.
- Each node maintain a blockchain.
- When a node mined a block, it is broadcast to all nodes.
- The version with a *large sum of PoW* (Proof of Work, i.e., ability to mine a block) will likely be the main branch



(a) Blockchain



(b) Network topology.

Cyberinfrastructure

Infrastructure composed of “cyber” elements

- Includes high-end computing (Supercomputing), grid computing, distributed computing, etc

Working definition

- An integrated system of interconnected computation, communication, or information element that supports a range of applications
- Another term: Cloud-and-network convergence

Cyberinfrastructure (con't)

Cyberinfrastructure consists of

- Computational engines (supercomputers, clusters, workstations, small processors, …)
- Mass storage (disk drives, tapes, …)
- Networking (including wireless, distributed, ubiquitous)
- Digital libraries/data bases
- Sensors/effectors
- Software (OS, middleware, domain specific tools)
- Services (education, training, consulting, user assistance)

Cyberinfrastructure (con't)

Characteristics

- Built on broadly accessible, highly capable network: 100's of terabits backbones down to intermittent, wireless connectivity at very low speeds;
- Contains significant and varied computing resources: 100's of petaflops at high ends, with capacity to support most scientific work;
- Contains significant storage capacity: exabyte collections common; high-degree of DB confederation possible;
- Allows wide range of of sensors/effectors to be connected: sensor nets of millions of elements attached;
- Contains a broad variety of intelligent visualization, search, database, programming and other services that are fitted to specific disciplines.

Cyberinfrastructure (con't)

Technical Challenges

- Computer Science and Engineering broadly
- How to build the components
- Networks, processors, storage devices, sensors, software
- How to shape the technical architecture
- Pervasive, many cyberinfrastructures, constantly evolving/changing capabilities
- How to customize cyberinfrastructures to particular S&E domains

Vision of the Field

Convergence of Multiple Disciplines

- Parallel processing, distributed systems, and network computing
- Distributed computing as an important component in Cyberinfrastructure
  - Upper and middle layers in integrated cyberinfrastructure
- Ultimate Cyberinfrastructure
  - petascale (10^{15}) computing, exabyte (10^{18}) storage, and terabit (10^{12}) networks to exascale
  - Fastest supercomputer: Frontier (Oak Ridge) over 1 exaFLOPs
  - Top 500: Home - | TOP500

Vision of the Field (Con't.)

Diverse aspects (ICDCS'22 areas)

- Cloud Computing and Data Centers
- Distributed Algorithms and Theory
- Distributed Big Data Systems and Analytics
- Distributed Fault Tolerance and Dependability
- Distributed Operating Systems and Middleware
- Edge Computing
- Internet of Things and Cyber-Physical Systems
- Mobile and Wireless Computing
- Security, Privacy, and Trust in Distributed Systems
- Blockchains
- Machine Learning on or for Distributed Systems
- Insights from Industrial Experience

Future of Distributed Computing

- **Theoretical aspects**, including global state, logical/physical clock, synchronization, and algorithm verification.
- **Fault tolerance** and crash recovery through software fault tolerance.
- **Tools** and **languages** are badly needs.
- **High-performance systems** that connects 10^5 or more processors.
- **Real-time distributed systems** used for automated manufacturing, remote sensing and control, and other time-critical missions.
- **Actual distributed systems (e.g., blockchain)** with significantly improved fault tolerance, resource sharing, and communications. These systems will function as single, coherent, powerful, virtual machines providing transparent user access to network-wide resources.

“old” Challenges

- **Transparency**
- **Scalability**
- The need to meet various **goals**
  - Real time constraints
  - Fault tolerance
  - Security
  - Other quality/performance related objectives
- **Cross-field interaction**

“New” Challenges

Supernetworks: networks are faster than the computers attached to them

- Endpoints scale to bandwidth-match the network with multiple-10Gbps lambdas
- Models that simplify distributed application programming
- Middleware for bandwidth-matching distributed resources

High-speed, parallel-access storage abstractions

Adaptive and peer-to-peer data access

Real-time object models enabling predictable performance

Fault tolerance and security protocols and models

“New” Challenges: Green Computing

- Also green IT and ICT Sustainability
- Article: Harnessing Green IT: Principles and Practices
- Government
  - EPA’s Energy Star program
- Industry
  - Climate Savers Computing Initiative
  - The Green Grid
  - Green500
  - Green Comm Challenge

- Algorithmic efficiency
- Resource allocation
- Virtualization
- Terminal servers (thin client)
- Power management
- Data center power
- Operating system support
- Power supply
- Storage
- Video card
- Display
- Materials recycling
- Telecommuting

Table of Contents

- Introduction and Motivation
- Theoretical Foundations
- Distributed Programming Languages
- Distributed Operating Systems
- Distributed Communication
- Distributed Data Management
- Reliability
- Applications
- Conclusions
- Appendix

A List of Common Symbols in DCDL

□	Alternate
*	Repetition
||	Parallel
→	Condition
;	Sequence
Send	Output
:=	Assignment
Receive	Input
::	Definition

[	Begin
]	End
∀	For all (universal quantifier)
∃	There exist (existential quantifier)
=	Equal
≠	Unequal
∨	OR
∧	AND
¬	NOT

Exercise 8

1. Discuss the relative advantages and disadvantages of the four algorithms based on Stumm and Zhou's classification: central-server algorithm, migration algorithm, read-replication algorithm, and full replication algorithms.

2. Consider a 9x9 2-D CAN with two opposite corners: (0,0) and (8,8). Initially, there is only one user at (1,1). The subsequent join and leave functions are the following: (1) (1,5) joins, (2) (3, 7) joins, (3) (7,8) joins, (4) (7,3) joins, (5) (6,5) joins, and (6) (1,1) leaves. Find the corresponding 2D maps and corresponding interconnection networks.

3. Given a set of six map-reduce tasks with (map, shuttle) loads: (1, 3), (4, 1), (4, 3), (3, 4), (2, 3), (2, 2). (1) Suppose map-shuttle is non-overlapping and the objective is to minimize the last job completion time, use Johnson's rule to get the optimal schedule. (2) Suppose map-shuttle is overlapping and the objective is to minimize the average completion time, find an optimal schedule.