Distributed Operating Systems

Prof. Chi-Sheng Shih
Graduate Institute of Networking and Multimedia
Department of Computer Science and Information Engineering
National Taiwan University

High Performance Computing

Cloud Computing

Are they the same?

Cloud Computing = High Performance Computing?

- Shared properties:
  - Large amount of computation resources are interconnected to provide coherence services.
  - Located in a server room/data center and connected via networks.

Workloads

- Real-time Weather forecast
- Nuclear fusion research
- Stock trading
- Facebook
- Online Gaming

Workloads

- Real-time Weather forecast:
  - Large amount of data, short latency (< 10s)
- Nuclear fusion research
  - Low latency, generating large amount of data during and after the workload,
- Stock trading
  - Real-time response (< 10^{-2}s), 10^{6} requests per second, guaranteed ordering.

Performance

- Facebook
  - 10^{3} of participants per message, long latency, guaranteed ordering, and number of messages increase over time.
- Online Gaming
  - Short latency (< 1s), 10^{3} of players per games, number of games increase over time.

Scalability

HPC vs. Cloud Computing

- High performance computing:
  - The majority of the workloads are computation intensive and can only tolerate short latency among sub-workloads.
  - The systems are built with high performance processors, and high bandwidth bus. However, it is not easy to add additional computation resources.
- Cloud Computing:
  - The majority of the workloads can be partitioned and conducted independently.
  - The systems are built with low cost processors, and computer networks. However, it is designed to add/remove computation resources at any time.
  - The performance are improved by adding more computation resources.
- HPC and Cloud Computing are distributed computing in general.

Why distributed computing systems?

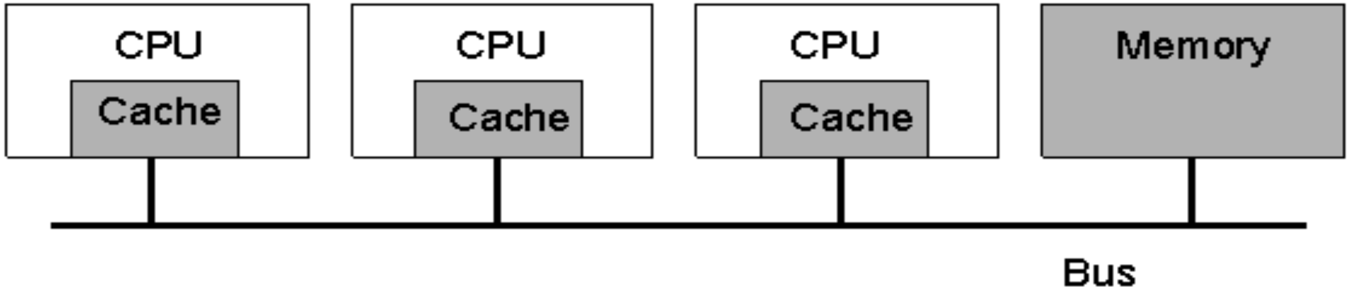
- Personal computers are cheap and powerful.
- Why bother to use distributed computing systems?
  - (Do you use peer-to-peer file sharing/streaming?)
  - Broadband connection is becoming popular and costs a little
  - Inherently distributed applications
  - Communication and resource sharing are possible
  - Economics – price-performance ratio
  - Higher reliability
  - Scalability
  - Potential for incremental growth
- What should be done to make it possible/better?
  - Distribution-aware platforms, operating systems and applications.
  - Security and privacy.

Distributed Computing System Models

- A distributed system:
  - Multiple connected processors/computing devices working together.
  - A collection of independent computers that appear to its users as a single coherent system
- Examples of distributed computing system models:
  - Minicomputer Model
  - Workstation Model
  - Workstation-server Model
  - Processor-pool Model
  - Hybrid Model

Hardware Concepts: Multiprocessors (1)

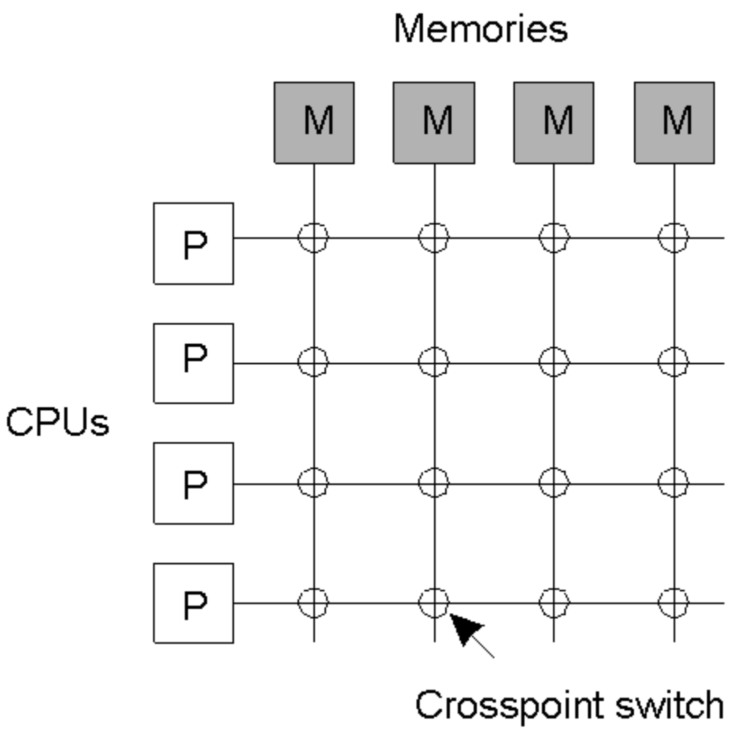
- Multiprocessor dimensions
  - Memory: could be shared or be private to each CPU
  - Interconnect: could be shared (bus-based) or switched
- A bus-based multiprocessor.



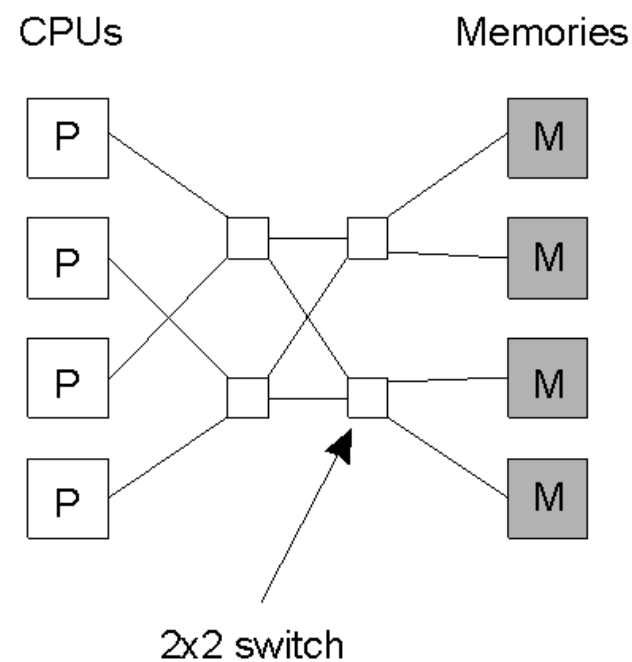
- Q: What are the potential problem for bus-based multiprocessor?

Multiprocessors (2)

a) A crossbar switch b) An omega switching network



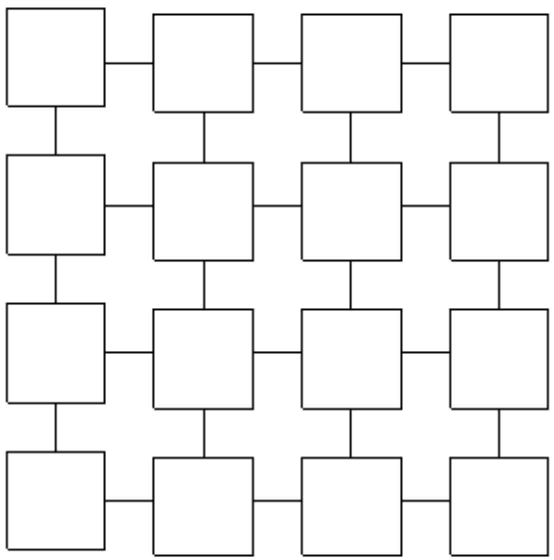
(a)



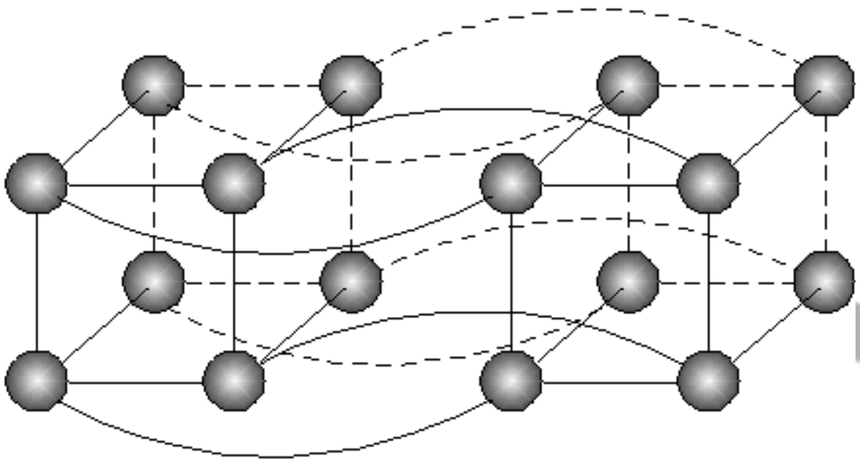
(b)

Homogeneous Multicomputer Systems

a) Grid



b) Hypercube



Distributed Systems Models

- Minicomputer model (e.g., early networks)
  - Each user has local machine
  - Local processing but can fetch remote data (files, databases)
- Workstation model (e.g., Sprite)
  - Processing can also migrate
- Client (workstation)- server Model (e.g., V system, world wide web)
  - Each user has local workstation
  - Powerful workstations serve as servers (file, print, DB servers)
- Processor pool model (e.g., Amoeba, Plan 9)
  - Terminals are Xterms or diskless terminals
  - Pool of backend processors handle processing
- Cloud Computing
  - Relationship between client and servers changes over time.
  - Computation capacity on servers are dynamically adjustable.

Distributed Operating Systems

- What's an operating system?
  - To present users with a virtual environment that is easier to program than the underlying hardware.
  - To manage the various resources of the system.

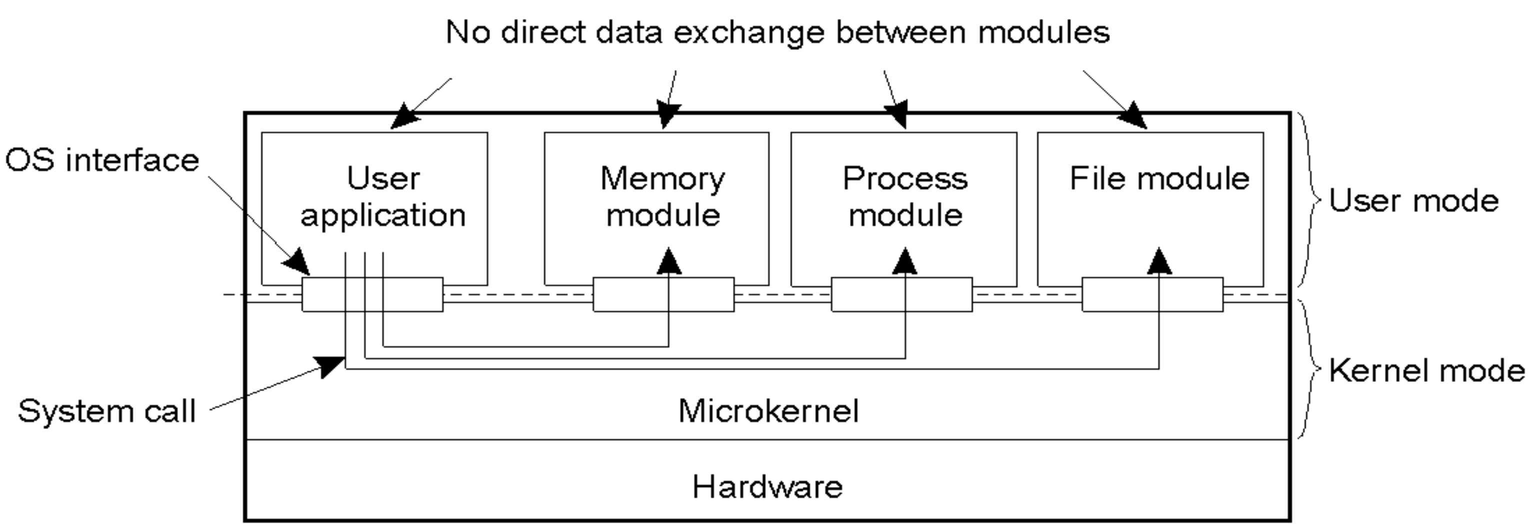
Uniprocessor Operating Systems

- An OS acts as a resource manager or an arbitrator
  - Manages CPU, I/O devices, memory
- OS provides a virtual interface that is easier to use than hardware
- Structure of uniprocessor operating systems
  - Monolithic (e.g., MS-DOS, early UNIX)
    - One large kernel that handles everything
  - Layered design
    - Functionality is decomposed into N layers
    - Each layer uses services of layer N-1 and implements new service(s) for layer N+1

Uniprocessor Operating Systems

Microkernel architecture

- Small kernel
- user-level servers implement additional functionality



Distributed Operating Systems

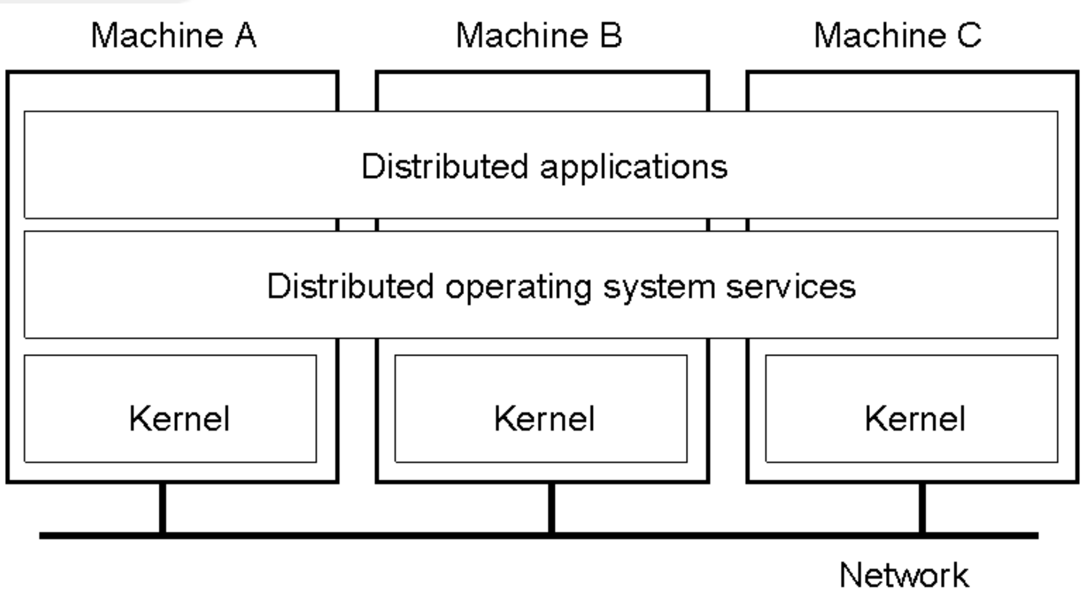
- The operating system for distributed computing systems:
  - Network operating systems
  - Distributed operating systems

Types of OSs for distributed systems

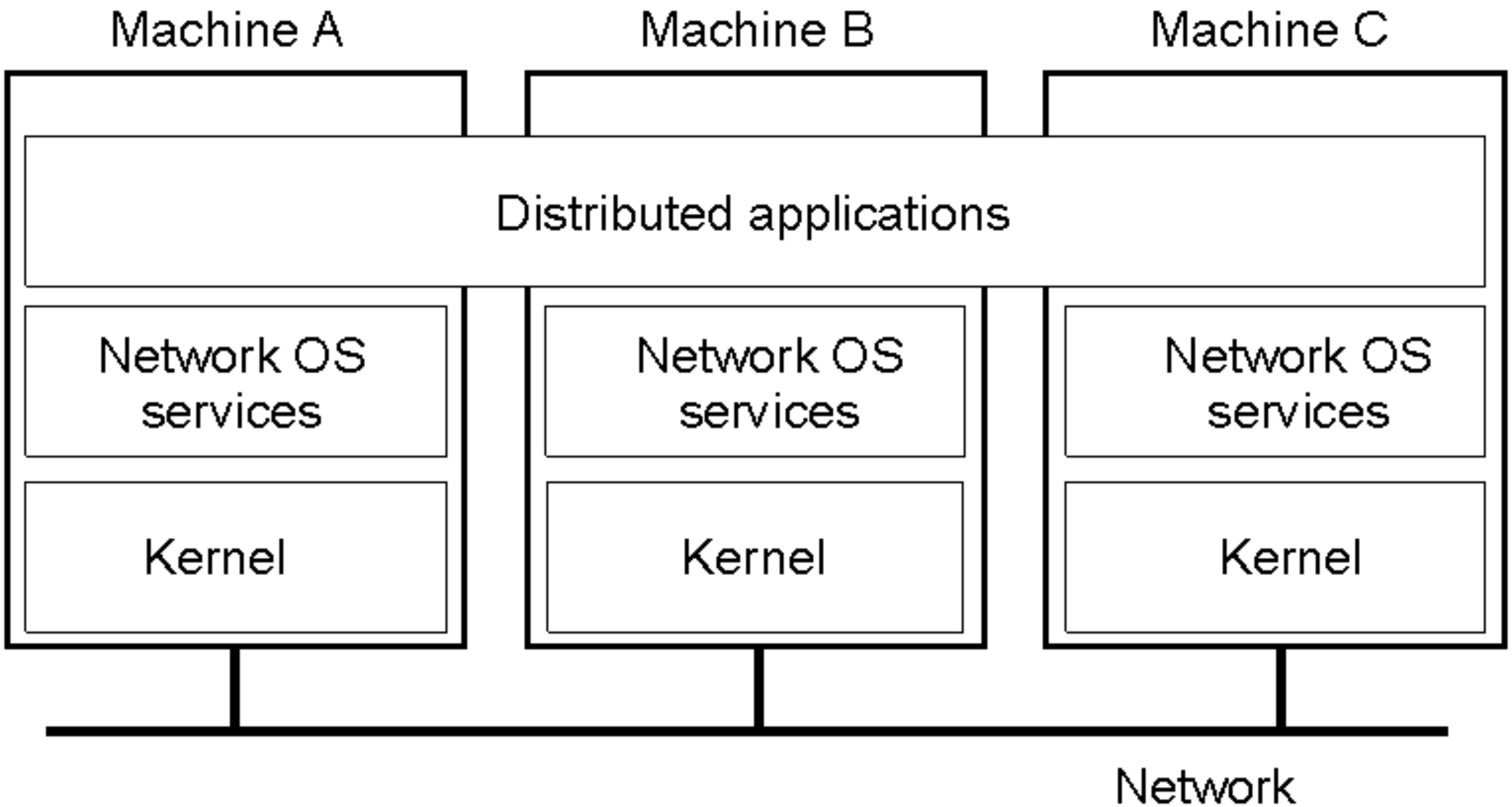
System	Description	Main Goal
DOS	Tightly-coupled operating system for multi-processors and homogeneous multicomputers	Hide and manage hardware resources
NOS	Loosely-coupled operating system for heterogeneous multicomputers (LAN and WAN)	Offer local services to remote clients
Middleware	Additional layer atop of NOS implementing general-purpose services	Provide distribution transparency

Distributed Operating Systems

- Each computing unit has its own hardware (including memory, CPU, and IO devices)
- A distributed operating system
  - looks like a uni-processor operating system but operates on multiple independent computing units
  - manages multiple computing units transparently to the user

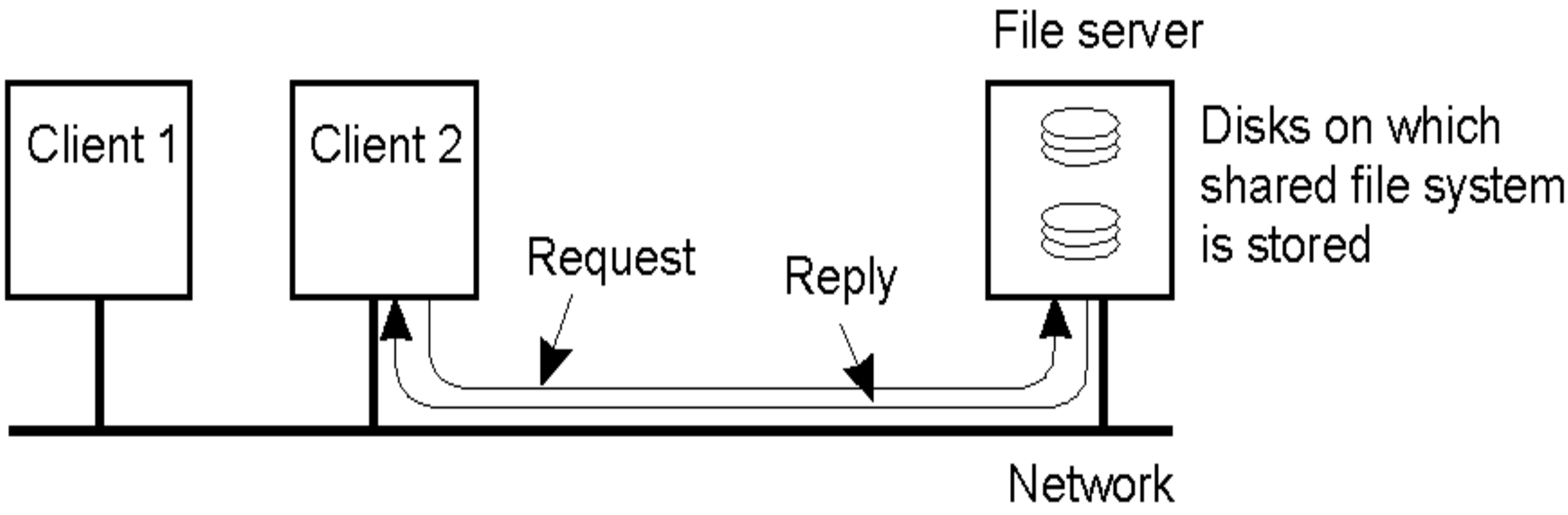


Network Operating System



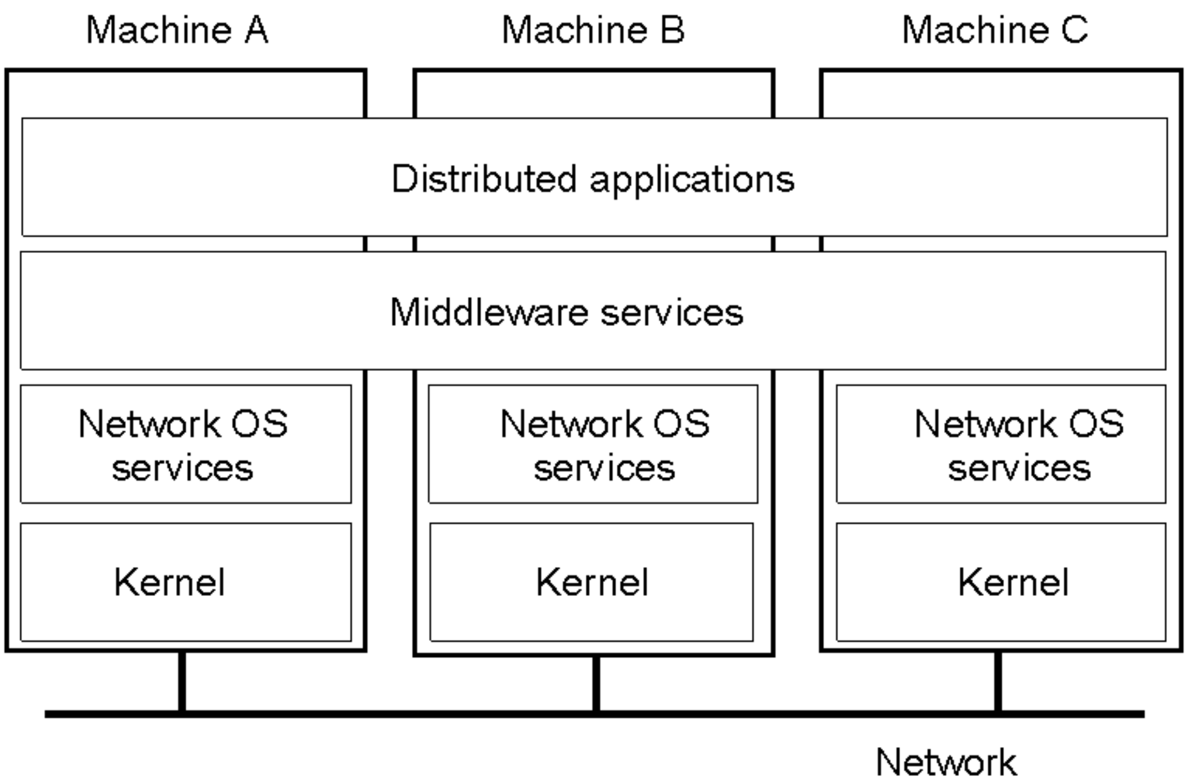
Network Operating System

- Employs a client-server model
  - Minimal OS kernel
  - Additional functionality as user processes



Middleware-based Systems

- General structure of a distributed system as middleware.



Network OS vs. Distributed OS

	Network OSs	Distributed OSs
System Image		
Autonomy		
Fault tolerance capability		

Goals of developing distributed systems

- Connecting Users and Resources
- Transparency
- Openness
- Scalability
- Flexibility
- Reliability
- Performance

Transparency in Distributed Systems

Transparency	**Description**
Access	
Location	
Migration	
Relocation	
Replication	
Concurrency	
Failure	
Persistence	

Transparency in Distributed Systems

Transparency	**Description**
Access	Hide differences in data representation and how a resource is accessed
Location	Hide where a resource is located
Migration	Hide that a resource may move to another location
Relocation	Hide that a resource may be moved to another location while in use
Replication	Hide that a resource is replicated.
Concurrency	Hide that a resource may be shared by several competitive users
Failure	Hide the failure and recovery of a resource
Persistence	Hide whether a (software) resource is in memory or on disk

Openness

- An open distributed system offers services according to standard rules.
- Services are generally specified through interfaces in Interface Definition Language.
  - Completeness
  - Neutrality
- Distributed services should have
  - Interoperability and
  - Portability

Reliability

- A fault in a system may cause system failure. Multiple resources may not be able to increase the system reliability.
  - Fail-stop failure
  - Byzantine failure
- Fault-handling mechanisms:
  - Fault Avoidance
  - Fault Tolerance
    - Redundancy technique
    - Distributed control
  - Fault Detection and Recovery
    - Atomic transaction
    - Stateless servers
    - Acknowledgements and timeout-based retransmission of messages.

Byzantine failure

- A Byzantine Fault is an incorrect operation (algorithm) that occurs in a distributed system that can be classified as:
  - Omission Failure – a failure of not being present such as failing to respond to a request or not receiving a request.
  - Execution Failure or Lying – a failure due to sending incorrect or inconsistent data, corrupting the local state or responding to a request incorrectly.
  - Examples
    - Round off errors passed from one function to another and then another, etc.
    - Corrupted system databases where the error is not detected Compiler errors
    - An undetected bit flip producing a bad message.
- This is a worse case model since the Byzantine Fault can generate misleading information causing a maximum of confusion.

Flexibility

- Why flexibility is an important feature of a distributed operating system?
  - Ease of modification
  - Ease of enhancement
- A flexible distributed system should be organized as a collection of relatively small and easily replaceable or adaptable components.
- Separating policy from mechanism
  - Web caching
    - Policy=?
    - Mechanism=?

Performance

- Distributed systems vs. centralize systems
  - Google search

L Barroso, J Dean, U Hoezle, Web Search for a Planet: The Architecture of the Google Cluster, - IEEE Micro, 2003.

How fast can you sort?

- Sort is a fundamental function for data analysis.
- Minute Sort:
  - Amount of data that can be sorted in 60.00 seconds or less.
- Gray Sort:
  - How much time taken to sort 100TB data?
- Joule Sort: Amount of energy to sort 10^{10} (10 giga) records.

Latest Results for Sort Benchmark

2019 competition submission was closed on September 1, 2019.

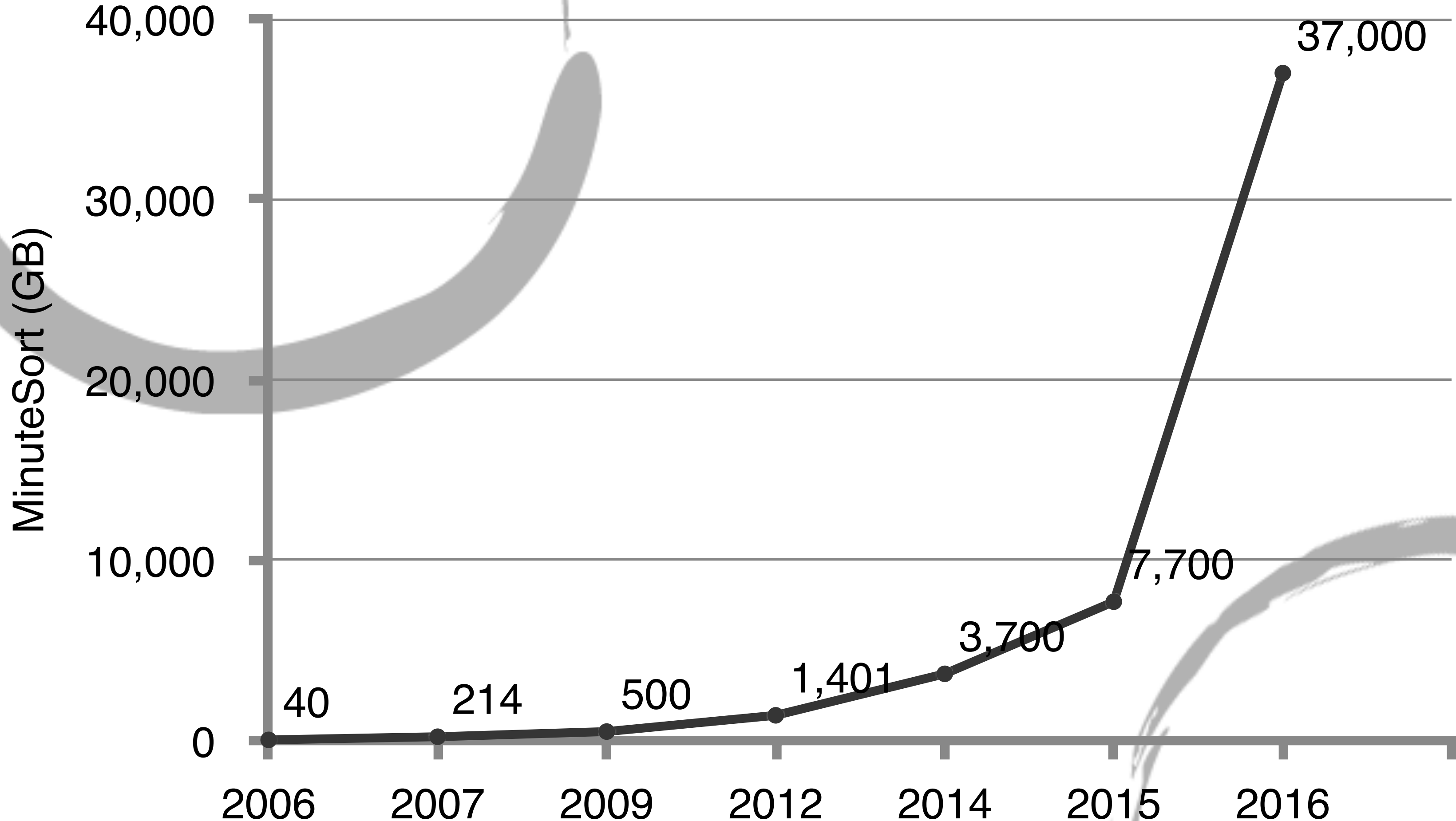
Minute Sort: How much data one can sort in one minute?

Minute	2016, 37 TB **Tencent Sort** 512 nodes x (2 OpenPOWER 10-core POWER8 2.926 GHz, 512 GB memory, 4x Huawei ES3600P V3 1.2TB NVMe SSD, 100Gb Mellanox ConnectX4-EN) Jie Jiang, Lixiong Zheng, Junfeng Pu, Xiong Cheng, Chongqing Zhao Tencent Corporation Mark R. Nutter, Jeremy D. Schaub	2016, 55 TB **Tencent Sort** 512 nodes x (2 OpenPOWER 10-core POWER8 2.926 GHz, 512 GB memory, 4x Huawei ES3600P V3 1.2TB NVMe SSD, 100Gb Mellanox ConnectX4-EN) Jie Jiang, Lixiong Zheng, Junfeng Pu, Xiong Cheng, Chongqing Zhao Tencent Corporation Mark R. Nutter, Jeremy D. Schaub

Gray: How much time it takes to sort 100TB?

	Daytona	Indy
Gray	2016, 44.8 TB/min **Tencent Sort** 100 TB in 134 Seconds 512 nodes x (2 OpenPOWER 10-core POWER8 2.926 GHz, 512 GB memory, 4x Huawei ES3600P V3 1.2TB NVMe SSD, 100Gb Mellanox ConnectX4-EN) Jie Jiang, Lixiong Zheng, Junfeng Pu, Xiong Cheng, Chongqing Zhao Tencent Corporation Mark R. Nutter, Jeremy D. Schaub	2016, 60.7 TB/min **Tencent Sort** 100 TB in 98.8 Seconds 512 nodes x (2 OpenPOWER 10-core POWER8 2.926 GHz, 512 GB memory, 4x Huawei ES3600P V3 1.2TB NVMe SSD, 100Gb Mellanox ConnectX4-EN) Jie Jiang, Lixiong Zheng, Junfeng Pu, Xiong Cheng, Chongqing Zhao Tencent Corporation Mark R. Nutter, Jeremy D. Schaub

Trend of Minute Sort



Status of Minute Sort

- 2016:
  - Winner: Tencent Corporation, China
  - 37 TB TB/min using 512 nodes x (2 OpenPOWER 10-core POWER8 2.926 GHz, 512 GB memory, 4x Huawei ES3600P V3 1.2TB NVMe SSD,
- 2015:
  - Winner: FuxiSort by AliBaba
  - 7.7 TB using 3,134 nodes x (2 Xeon E5-2630 2.30Ghz, 96 GB memory, 12x2 TB SATA HD, 10 Gb/s Ethernet) + 243 nodes x (2 Xeon E5-2650v2 2.60Ghz, 128 GB memory, 12x2 TB SATA HD, 10 Gb/s Ethernet)
- 2014:
  - Winner: DeepSort by Zheng Li, Juhan Lee, Samsung.
  - 3.7 TB using 384 nodes of 2x2.1GHz Intel Xeon, 64GB memory, and 8 HDs
- 2012:
  - Winner: Flat Datacenter Storage from Microsoft Research
  - 1,401 GB using 256 nodes
- 2009:
  - Winner: Hadoop from Yahoo
  - 500GB using 1406 nodes x (2 quad core Xeons, 8GB memory, 4 SATA HDs)
- 2007: 214BGB
- 2006: 40GB
- 2004: 34GB

Latest Results for Sort Benchmark

Joule 10^{10} recs	2-way tie: 2019, 163 KJoules **TaichiSort** 61 K records sorted / joule Intel i7-9700, 32GB RAM, Nsort, Ubuntu 16.04.3 LTS, 2 Intel DC 3600 series PCIe NVMe SSD (1.2 TB), 1 Intel DC 3600 series PCIe NVMe SSD (2.0 TB) Ming Liu, Kaiyuan Zhang, Arvind Krishnamurthy University of Washington Simon Peter University of Texas at Austin 2013, 168 KJoules **NTOSort** 59 K records sorted / joule Intel i7-3770K, 16GB RAM, Nsort, Windows 8, 16 Samsung 840 Pro 256GB SSDs, 1 Samsung 840 Pro 128GB SSD Andreas Ebert Microsoft	2019, 89 KJoules **KioxiaSort** 112 K records sorted / joule Intel i9-9900K, 64GB RAM, Ubuntu 19.04 Server, 8 CFD CSSD-M2B1TPG3VNF (1TB), 1 Toshiba XG5-P KXG50PNV2T04 (2TB) Shintaro Sano, Tomoya Suzuki Kioxia Corporation Zaid Mahmoud Princess Sumaya University for Technology

http://sortbenchmark.org/

Performance

- Distributed systems vs. centralize systems
  - Google search
- How can the performance of a distributed system be as good as a centralized system?
  - Batch if possible
  - Cache whenever possible
  - Minimize copying of data
  - Minimize network traffic
  - Take advantage of fine-grain parallelism for multiprocessing

L Barroso, J Dean, U Hoezle, Web Search for a Planet: The Architecture of the Google Cluster, - IEEE Micro, 2003.

Scalability Problems

Concept	Example
Centralized services	A single server for all users
Centralized data	A single on-line telephone book
Centralized algorithms	Doing routing based on complete information

Examples of scalability limitations.

Scalability Problems

- Decentralized algorithms should be used.
  - No machine has complete information about the system state.
  - Machines make decisions based only on local information.
  - Failure of one machine does not ruin the algorithm.
  - There is no implicit assumption that a global clock exists.
- Geographical scalability:
  - LAN vs. WAN.

Comparison between Systems

Item	Distributed OS		Network OS	Middleware-based OS
	Multiproc	Multicomp		
Degree of transparency	Very High	High	Low	High
Same OS on all nodes	Yes	Yes	No	No
Number of copies of OS	1	N	N	N
Basis for communication	Shared memory	Messages	Files	Model specific
Resource management	Global, central	Global, distributed	Per node	Per node
Scalability	No	Moderately	Yes	Varies
Openness	Closed	Closed	Open	Open

Additional Readings

- David L. Cohn, William P. Delaney, Karen M. Tracey, ARCADE: A Platform for Heterogeneous Distributed Operating Systems, *Proceedings of the Symposium on Experiences with Distributed and Multiprocessor Systems, 1989, 373-390.*
- Douglis, F., Ousterhout, J.K., Kaashoek, M.F., and Tanenbaum, A.S., *Comparison of Two Distributed Systems: Amoeba and Sprite, Computing Systems Journal 4(Fall), 1991, 353-384.*
- L Barroso, J Dean, U Hoezle, *Web Search for a Planet: The Architecture of the Google Cluster*, IEEE Micro, Volume: 23, Issue: 2, March-April, 2003, 22- 28**.**
- Thain, D., Tannenbaum, T. and Livny, M. (2005), Distributed computing in practice: the Condor experience. Concurrency and Computation: Practice and Experience, 17: 323–356. doi: 10.1002/cpe.938
- B Hayes, *Cloud Computing*, Communication of ACM, Vol. 51, Issues 7, July 2008.
- Anil Madhavapeddy, Richard Mortier, Charalampos Rotsos, David Scott, Balraj Singh, Thomas Gazagnaire, Steven Smith, Steven Hand, and Jon Crowcroft. 2013. Unikernels: library operating systems for the cloud. SIGARCH Comput. Archit. News 41, 1 (March 2013), 461-472.
- Jiamang Wang, Yongjun Wu, Hua Cai, Zhipeng Tang, Zhiqiang Lv, Bin Lu, Yangyu Tao, Chao Li, Jingren Zhou, and Hong Tang, FuxiSort, file: http://sortbenchmark.org/FuxiSort2015.pdf

Before next class

- Reading assignment:
  - **Anil Madhavapeddy, Richard Mortier, Charalampos Rotsos, David Scott, Balraj Singh, Thomas Gazagnaire, Steven Smith, Steven Hand, and Jon Crowcroft. 2013. Unikernels: library operating systems for the cloud. SIGARCH Comput. Archit. News 41, 1 (March 2013), 461-472.**
  - **Brian N. Bershad, Thomas E. Anderson, Edward D. Lazowska, and Henry M. Levy. 1990. Lightweight remote procedure call. ACM Trans. Comput. Syst. 8, 1 (February 1990), 37-55.**