Introduction to Distributed Systems

Tran Giang Son, tran-giang.son@usth.edu.vn

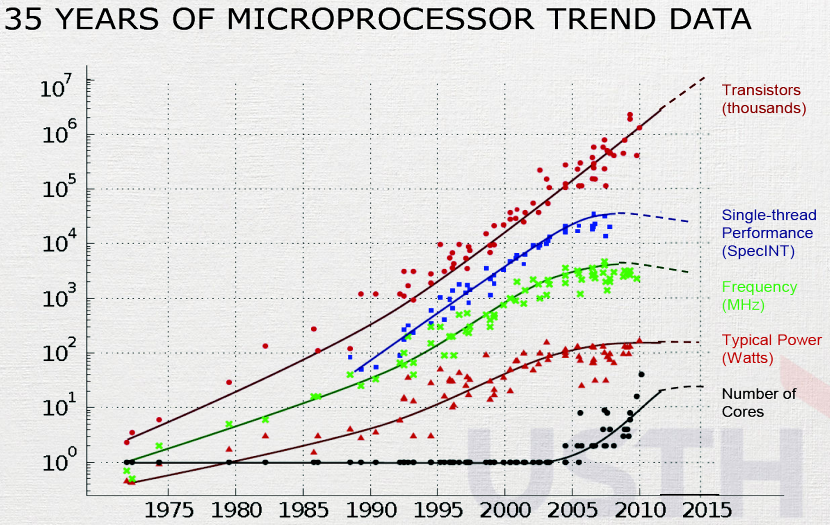
ICT Department, USTH

> *A distributed system is one in which the failure of a computer you didn't even know existed can render your own computer unusable*
>
> \- Leslie Lamport, 1987

Distributed Architecture

Centralized Computing

- One big, fast guy



Distributed Systems

- Hardware and software of a collection of independent computers
- Cooperate to implement some functionality

Distributed Systems

- Abstraction of distribution
  - Storage
  - Communication
  - Computation

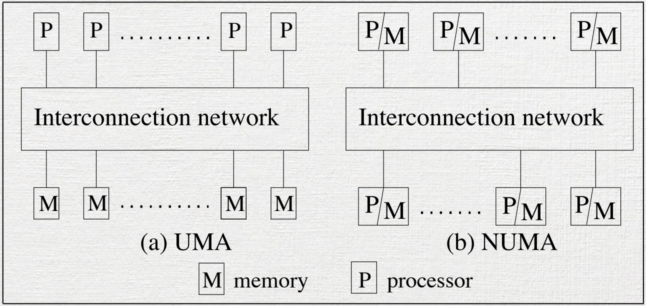
Distributed Systems : Examples

- Multi-tier web apps: Facebook, Twitter...
- Cloud-based systems: GCP, Azure, EC2, Drive, Flickr, ...
- Scientific applications: SETI@Home, Folding@Home

Parallel System?

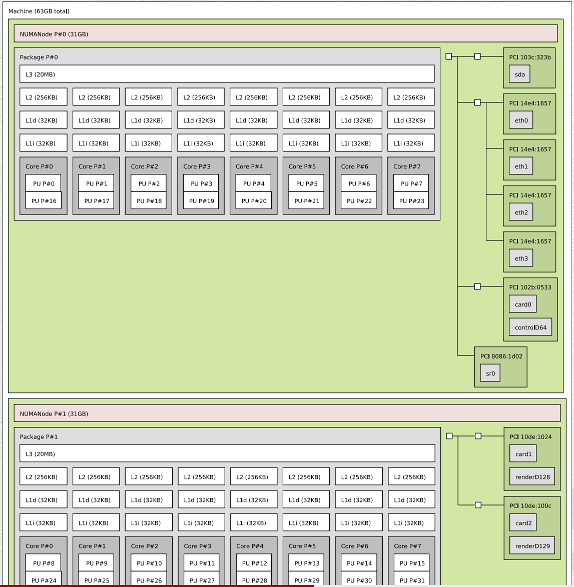
- Multiprocessor systems
  - Direct access to shared memory, UMA
  - Interconnection network
- Multicomputer parallel systems
  - No direct access to shared memory, NUMA
  - Easier scalability

Parallel System

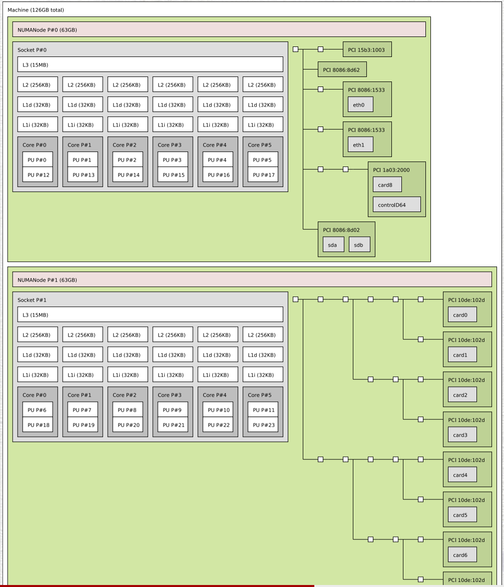


UMA vs NUMA

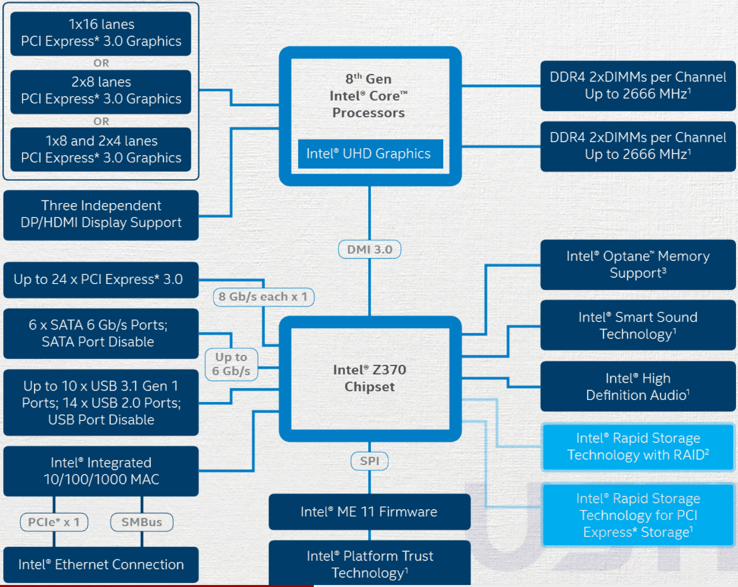
Parallel System: ICTLab's ICT2 NUMA example



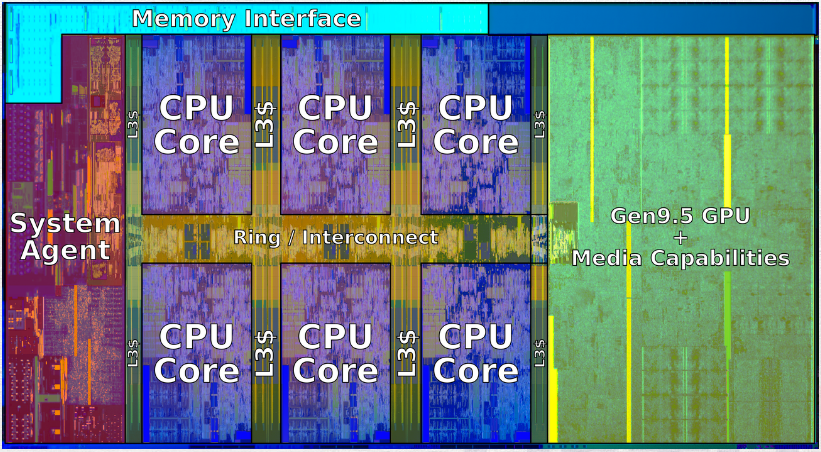
Parallel System: ICTLab's ICT5 NUMA example



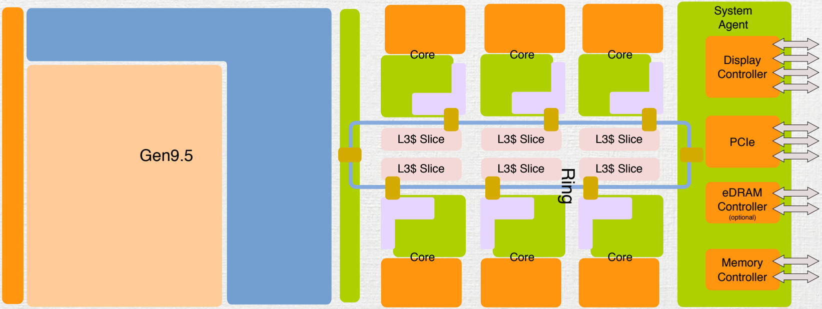
Parallel System: Intel Coffee Lake



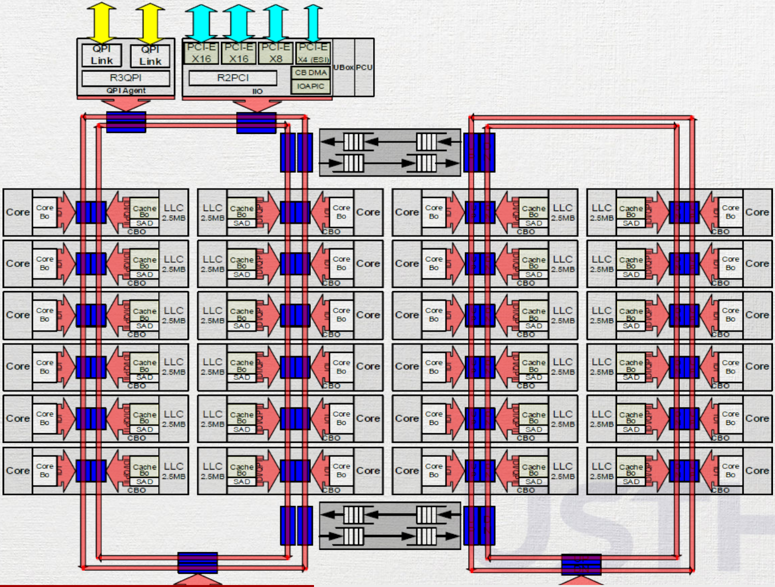
Parallel System: Intel Coffee Lake 8700K



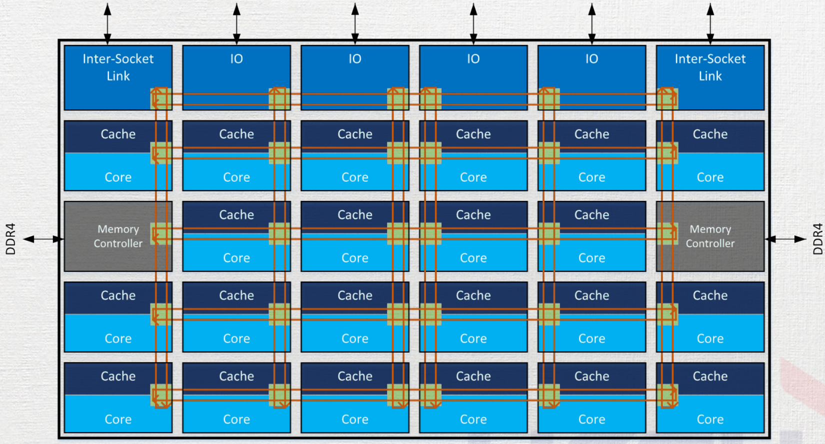
Parallel System: Intel Coffee Lake 8700K



Parallel System: Intel Broadwell EP Xeons

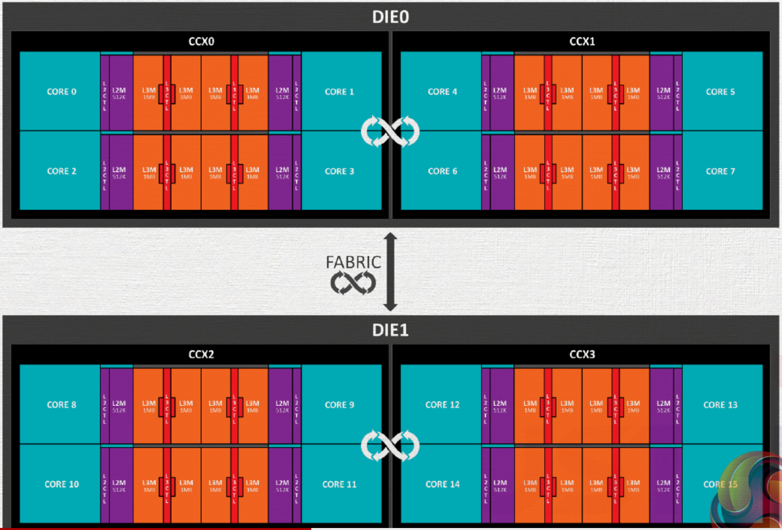


Parallel System: Intel Skylake SP Xeons



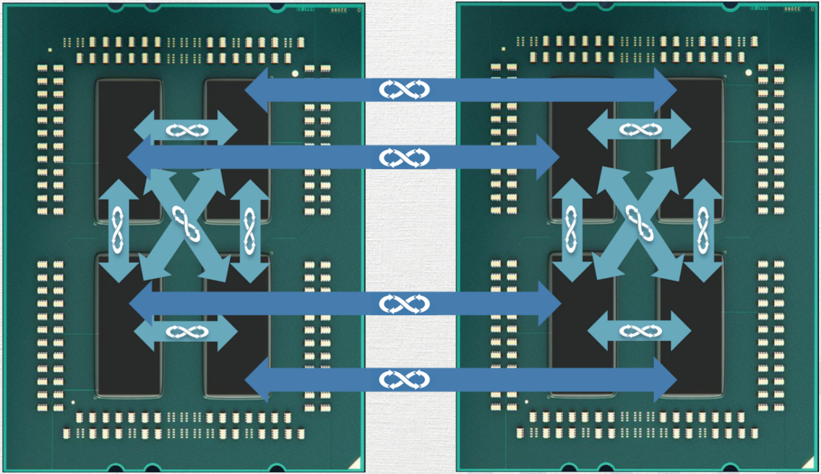
Parallel System: AMD ThreadRipper

- Intra-Socket



Parallel System: AMD Epyc

- Inter-Socket



Why?

- Performance
- Scalability
- Reliability
    - Availability, fault-tolerance
- Modularity
- Resource sharing
- Efficiency

Why not?

- Communication
- Synchronization
- Fault-tolerance
- Security
- Scalability

Distributed System: Common mistakes

- The network is reliable
- Latency is zero
- Bandwidth is infinite
- The network is secure
- Topology doesn't change
- There is one administrator
- The network is homogeneous

Distributed System: Common mistakes - Network

- Hardware failures: switch/router failure, cable failure, power failure
- Solutions
  - Hardware: redundancy
  - Software: reliable messaging. Retries. Ordering. Integrity.

Distributed System: Common mistakes - Latency

- Latency?

Distributed System: Common mistakes - Latency

- Latency?
  - Not bandwidth
  - Time for data to move from one place to another (sec, ms)

Distributed System: Common mistakes - Latency

- Latency?
  - Not bandwidth
  - Time for data to move from one place to another (sec, ms)
- Limit: speed of light

Distributed System: Common mistakes - Latency

- Latency?
  - Not bandwidth
  - Time for data to move from one place to another (sec, ms)
- Limit: speed of light :
  - 300,000 km/s
  - Ping Hanoi to Washington DC?

Distributed System: Common mistakes - Latency

- Latency?
  - Not bandwidth
  - Time for data to move from one place to another (sec, ms)
- Limit: speed of light :
  - 300,000 km/s
  - Ping Hanoi to Washington DC?
  - 13359 km

Distributed System: Common mistakes - Latency

- Latency?
  - Not bandwidth
  - Time for data to move from one place to another (sec, ms)
- Limit: speed of light :
  - 300,000 km/s
  - Ping Hanoi to Washington DC?
  - 13359 km
  - 44.53ms

Distributed System: Common mistakes - Latency

- Latency?
  - Not bandwidth
  - Time for data to move from one place to another (sec, ms)
- Limit: speed of light :
  - 300,000 km/s
  - Ping Hanoi to Washington DC?
  - 13359 km
  - 44.53ms
  - 89ms roundtrip

Distributed System: Common mistakes - Latency

- Latency?
  - Not bandwidth
  - Time for data to move from one place to another (sec, ms)
- Limit: speed of light :
  - 300,000 km/s
  - Ping Hanoi to Washington DC?
  - 13359 km
  - 44.53ms
  - 89ms roundtrip

$\Rightarrow$ Think about latency.

Distributed System: Common mistakes - Bandwidth

- Bandwidth: how much data can be transferred (bit/sec)
- Faster and faster
- Packet loss

$\Rightarrow$ Use compression, if possible

Distributed System: Common mistakes - Secured Network

- Intermediate nodes between hosts
- Packet sniffing, eavesdropping on routers, switches
- Unsecured WiFi

$\Rightarrow$ Think about network security since day 1

Distributed System: Common mistakes - Topology

- Myth: the network topology doesn't change

Distributed System: Common mistakes - Topology

- Myth: the network topology doesn't change
- Reality:
  - Adding servers
  - Laptops and phones connect to network day by day
  - Topology changes

Distributed System: Common mistakes - Topology

- Myth: the network topology doesn't change
- Reality:
  - Adding servers
  - Laptops and phones connect to network day by day
  - Topology changes

$\Rightarrow$ Do not rely on specific endpoints or routes $\Rightarrow$ Use DNS.

Distributed System: Common mistakes - Administrators

- ICTLab : 1 administator
- USTH : 2 administrators
- Netnam : N administrators
- FPT : M administrators
- Different degrees of expertise
- Difficult to locate problems

Distributed System: Common mistakes - Homogeneous

- Different types of machines
  - ict1 : 6C12T / 24GB
  - ict2 : 16C32T / 64GB
  - ict3/4 : 6C12T / 32GB
  - ict5/6 : 12C24T / 128GB
  - ict7/8/9 : 8C16T / 24GB

Distributed System: Common mistakes - Homogeneous

- Different types of machines
  - ict1 : 6C12T / 24GB
  - ict2 : 16C32T / 64GB
  - ict3/4 : 6C12T / 32GB
  - ict5/6 : 12C24T / 128GB
  - ict7/8/9 : 8C16T / 24GB
- Different types of networks
  - WiFi: a, b, g, abg, n, ac
  - LAN: 10/100Mbps, 1Gbps
  - Internet: Fiber 24Mbps, 48Mbps, Leased lines 80Mbps

Distributed System: Common mistakes - Homogeneous

- Different types of machines
  - ict1 : 6C12T / 24GB
  - ict2 : 16C32T / 64GB
  - ict3/4 : 6C12T / 32GB
  - ict5/6 : 12C24T / 128GB
  - ict7/8/9 : 8C16T / 24GB
- Different types of networks
  - WiFi: a, b, g, abg, n, ac
  - LAN: 10/100Mbps, 1Gbps
  - Internet: Fiber 24Mbps, 48Mbps, Leased lines 80Mbps
- Different OSes
  - Windows

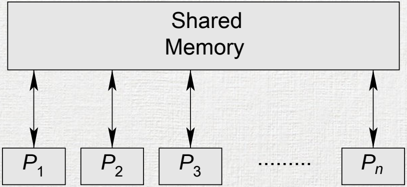
Distributed System: Common mistakes - Homogeneous

- Different types of machines
  - ict1 : 6C12T / 24GB
  - ict2 : 16C32T / 64GB
  - ict3/4 : 6C12T / 32GB
  - ict5/6 : 12C24T / 128GB
  - ict7/8/9 : 8C16T / 24GB
- Different types of networks
  - WiFi: a, b, g, abg, n, ac
  - LAN: 10/100Mbps, 1Gbps
  - Internet: Fiber 24Mbps, 48Mbps, Leased lines 80Mbps
- Different OSes
  - Windows

Parallel Models

PRAM

- Parallel Random Access Machine
- Shared memory
- Multiple processing units



PRAM

Read/write conflicts

- EREW: **E**xclusive **R**ead **E**xclusive **W**rite
- CREW: **C**oncurrent **R**ead **E**xclusive **W**rite
- ERCW: **E**xclusive **R**ead **C**oncurrent **W**rite
- CRCW: **C**oncurrent **R**ead **C**oncurrent **W**rite

Flynn's Taxonomy

- Classification of computer architecture by Michael J. Flynn in 1966
  - SISD: **S**ingle **I**nstruction **S**ingle **D**ata
  - SIMD: **S**ingle **I**nstruction **M**ultiple **D**ata
  - MISD: **M**ultiple **I**nstruction **S**ingle **D**ata
  - MIMD: **M**ultiple **I**nstruction **M**ultiple **D**ata

Flynn's Taxonomy

- Example:

[1] Instruction-level parallelism with pipelining

Flynn's Taxonomy

- Example:
  - SISD: Old school single core (scalar or superscalar[1]) CPUs

[1] Instruction-level parallelism with pipelining

Flynn's Taxonomy

- Example:
  - SISD: Old school single core (scalar or superscalar[1]) CPUs
  - SIMD: GPUs

[1] Instruction-level parallelism with pipelining

Flynn's Taxonomy

- Example:
  - SISD: Old school single core (scalar or superscalar[1]) CPUs
  - SIMD: GPUs
  - MISD: Highly fault tolerance system

[1] Instruction-level parallelism with pipelining

Flynn's Taxonomy

- Example:
  - SISD: Old school single core (scalar or superscalar[1]) CPUs
  - SIMD: GPUs
  - MISD: Highly fault tolerance system
  - MIMD: Modern multi-core CPUs

[1] Instruction-level parallelism with pipelining

Scalability

What?

- The ability to scale
  - N servers = N times better performance
  - Parallel CPU, disk, network
- More load == more computers

Why - Why not?

- Why?
    - More brains think faster
    - Many problems cannot be solved by one system
    - Many bigger problems cannot be solved by many systems

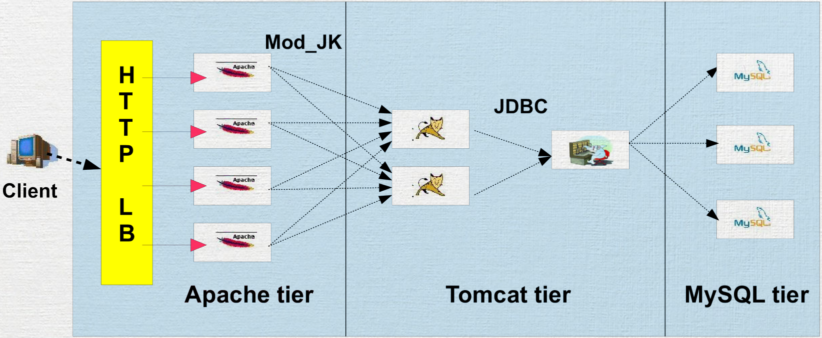
Why - Why not?

- Why?
  - More brains think faster
  - Many problems cannot be solved by one system
  - Many bigger problems cannot be solved by many systems
- Why not?
  - Load imbalance
  - Non parallelizable code
    - Initialization
    - Interaction
    - Dependencies
  - Bottleneck from shared resources

How?

- Duplicate of machines
- Each machine performs a part of the big work
- Combine the result together

Example

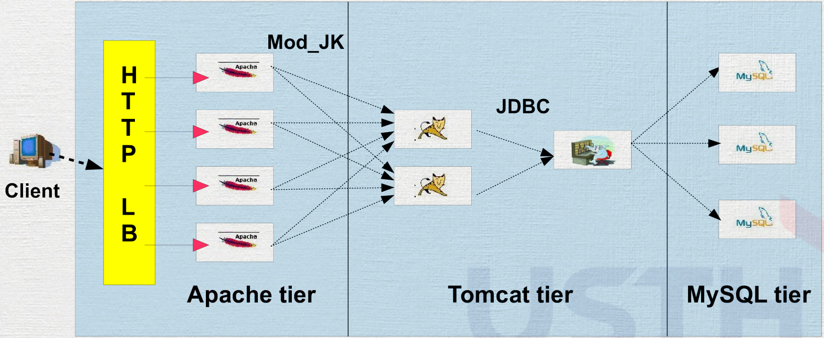


Classical J2EE Multi-tier application

Consistency

What?

- Well-defined behavior
  - Get(k) yields the value from the most recent Put(k,v)
- Example
  - `UPDATE` / `SELECT` from replicated MySQL



Why?

- Transparency of distributed systems
- Expectation of behavior from application
- Data quality

Why not?

- Hard to achieve good behavior
  - “Replica” is hard to keep identical
  - Client crashes during content updates
  - Server crashes
  - Unreliable network
- Think of distributed semaphore / mutex
- Anti-performance
  - Consistency requires communication, e.g. latest Put()
  - Slow

How?

- Strong consistency
  - Wait until all replicas update content
  - All further updates are pending
  - High latency

How?

- Balance consistency vs performance
- Weak consistency
  - Update once
  - Propagate updates
  - Low latency
  - Different Get() may return different values

Fault-tolerance

What?

- The ability to keep the system up and running with failure
  - Hardware
  - Software
- Hide failure from application

What?

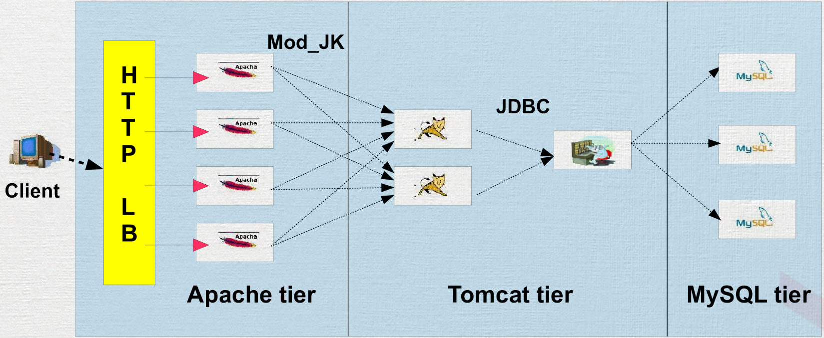
- The ability to keep the system up and running with failure
  - Hardware
  - Software
- Hide failure from application
- Availability
  - Processes can keep running and use their data even when failure
- Durability
  - App's data will come back when failures are repaired

Why?

- Always a chance for failure
  - Hard drive
  - Motherboard
  - Memory
  - CPU
- Probability of failure occurence increases with amount of servers
- Example: ICTLab's NAS RAID5 had 2 3TB HDDs died almost at the same time

How?

- Replication, as before
- If one server crashes, others can still take the work

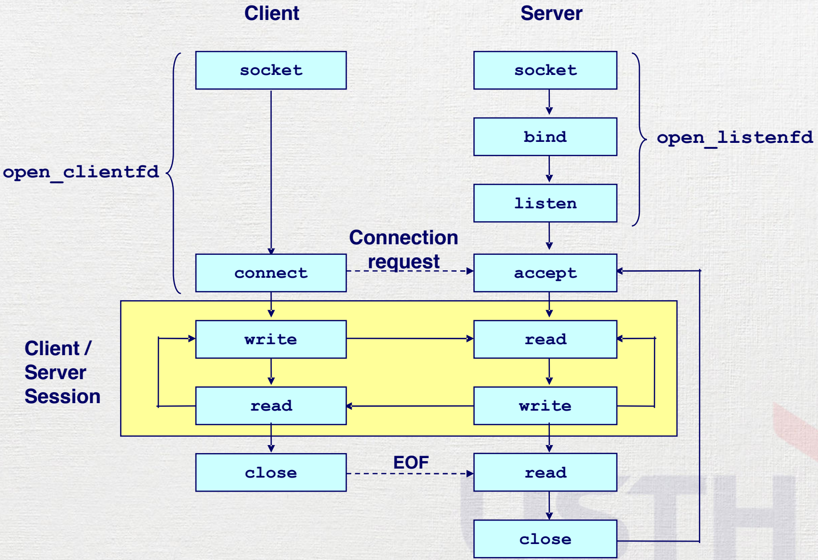


Classical J2EE Multi-tier application

Practical Work 1: TCP File transfer

- Goal: 1-1 File transfer over TCP/IP in CLI, based on the provided chat system
  - One server
  - One client
  - Using socket
- Write a short report in LaTeX:
  - Name it « 01.tcp.file.transfer.tex »
  - How you design your protocol. Figure.
  - How you organize your system. Figure.
  - How you implement the file transfer. Code snippet.
  - Who does what
- Work in your group, in parallel

Practical Work 1: USTH Master Spoiler Alert!



Practical Work 1: USTH Master Spoiler Alert!

- `socket()`: Creates a socket, a communications endpoint
- `setsockopt()`: Set options on a socket
- `bind()`: Associate a socket with an address
- `gethostbyname()`: Get the the address of the machine with a given name
- `listen()`: Listen for machines trying to connect to this machine
- `connect()`: Establish a connection with another machine
- `accept()`: Accept a connection
- `send()`: Send data over a connection
- `recv()`: Read data from a connection