Review1. Characterization of Distributed Systems

2019年5月29日 11:09

Basics of Internet and cluster

- Satellite link, backbone, ISP (Internet service provider), intranet, router, firewall.
- HPC (high performance computing): ubiquitous of reliable and scalable, autonomic of dynamic and discovery, composable of QoS and SLA and so on.
- Cluster architecture: servers, I/O devices, disk arrays, networks (SAN, LAN, NAS: Ethernet, Myrinet, InfiniBand) to Internet.
- Computational Grid (data grid): computing utility, data and information services.
  - Standards and middleware: OGSA, Globus, IBM Grid Toolbox.
- P2P system: mapping a physical IP network to an overlay network with virtual links.
  - Categories of P2P: system features, distributed file sharing, collaborative platform, distributed P2P computing, P2P platform.
- Clouds and Internet of things: HTC (high throughput computing) through P2P, HPC through cluster or MPPs (massively parallel processors) both in computational and data grids, and then goes to services, clouds, or IoT (Internet of things, RFID and sensors).
  - Opportunities of IoT: any time, place, thing connection.

Distributed parallel computing systems

- Definition: a system in which components located in networked computers communicate and coordinate their actions only by passing massages.
  - Motivation: sharing of resources.
  - Layers: distributed systems, computer networks, data transmission.
  - Computer network: a collection of interconnected autonomous computers.
    - Generality: built from general purpose hardware.
- Classification: multicomputer clusters, P2P networks, data / computational grids, cloud platforms.
  - Programming: MPI (C or Fortran), MapReduce (web), Hadoop (software library).
  - Middleware layers: resource broker, secure access, task analyzer and scheduler, communication service, information service, reliability control.
- ISO (international standards organization) OSI architecture: physical, data link, network, transport, session, presentation, application.
  - OSI session layer: establish and maintain a connection or session between 2 end-users.
  - *OSI presentation & application layer: presentation - communication, conversion, transmission, compression; application - software.*
- Internet architecture (TCP/IP): host-to-network layer, Internet layer, transport layer, application layer.
- Challenges: heterogeneity, openness, security, scalability, failure, concurrency, transparency.
  - *Transparencies: access, location, concurrency, replication, failure, mobility, performance, scaling.*

Review2. System Models

2019年5月29日 11:09

Tier architecture

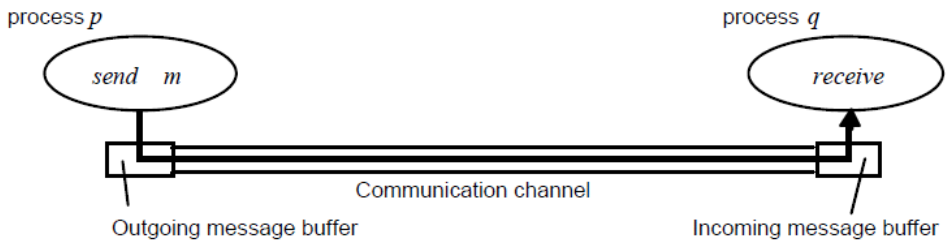
- Architecture: structure in terms of separate specified component.
  - Presentation layer: user submits operations and get responses.
  - Application logic layer (ALL / business logic): data processing.
  - *Data layer and user layer.*
- Tier architecture: architecture into n components.
  - Single tier: dumb terminal.
  - 2-tier architecture: user & P layer, ALL & data layer.
    - Fat client (work without network) and thin client.
  - 3-tier: user & P layer, ALL, (middleware), data layer.
    - Middleware: supports the development of ALL.
  - N-tier: many 3-tier distributed systems interacting (tiers are physically separated).
- Service layers: $\boldsymbol{platform = hardware + network + OS}$

Architecture model

- Distributed systems: functions of components, placement of components (data, workload) across a network, and inter-relationships between components.
- System architectures:
  - Client-server model: server - accept, process, respond; client: invoke server.
  - Proxy servers and caches: cache - store of recent used data; proxy - shared cache of web resource, or access remote web servers through a firewall.
  - Peer processes: cooperate to perform distributed activity.
  - *Variations of C-S model: Applets, thin clients.*

Design of distributed systems

- Requirements:
  - Performance: responsiveness, throughput, load balancing.
  - Quality of Service (QoS): deadline properties (hard & soft), adaptability.
  - Dependability: correctness, fault-tolerance, security.
- Fundamental models: specify assumptions, make generalisations.
  - Interaction model: distributed algorithms (including communication, non-deterministic), important factors (performance, timing events), synchronous.
  - Failure: process (send/receive using outgoing/incoming message buffer) and communication channel.



    - Omission and arbitrary failures: process (fail-stop, crash, send/receive-omission, arbitrary) and channel (omission, arbitrary) failure.
    - Timing failures: process (clock, performance) and channel (performance).

Review3. Interprocess Communication

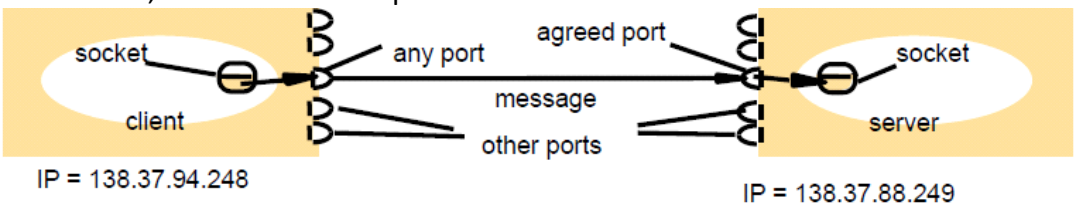
2019年5月29日 11:09

Network architecture

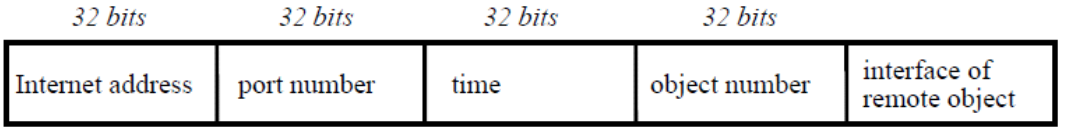
- Software: protocols and services, hardware: transmission technology, media and devices (scale: LANs, MANs, and WANs), topology.
  - Subnet: connected by router, with LAN and host.
- *Layered architecture & protocol.*
- Accessing transport layer: request-reply protocols.

Inter-process communication

- Message passing model: send-receive.
  - Synchronous communications (blocking) and asynchronous communications (blocking receive and non-blocking send).
  - Reliability: validity and integrity.
  - Ordering: message send and delivery.
- Sockets and ports: sockets - endpoint for inter-process communication, port - message destination, socket bounds to port.



- Remote object reference: identifier for an object (unique, may passed as argument - object number).



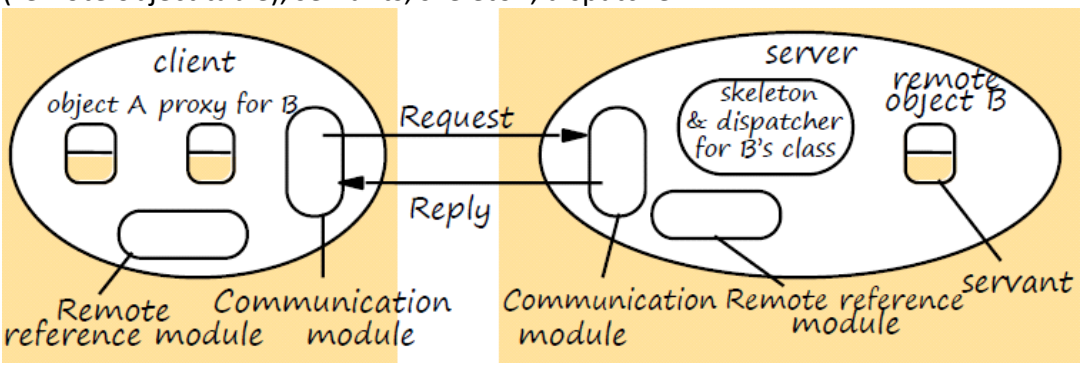
- The request-reply protocol: processes communicate using a protocol.
  - *Issues: message identifiers, failure model, timeouts, duplicate request messages, lost replay messages, history for retransmissions.*
- UDP sockets: non-blocking send & blocking receive, receive from any.
  - ***DatagramPacket*** & ***DatagramSocket***
- TCP sockets: each socket is both for input and output.
  - ***ServerSocket*** & ***Socket***
- Group communication: multicast.

Review4. Distributed Objects and Remote Invocation

2019年5月29日 11:09

Distributed applications programming

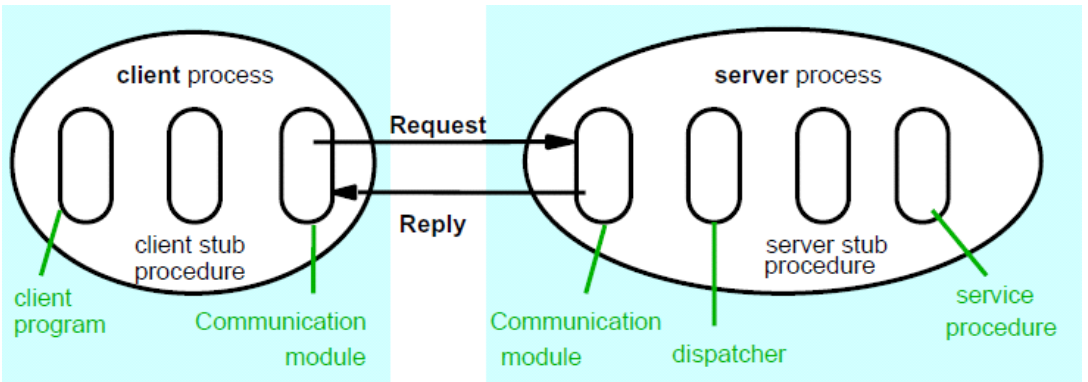
- Objects: data (attributes) and operations (methods).
  - Interact via interfaces: define types of arguments and exceptions.
- *The object (local) model: OO programs, interfaces, actions, garbage collection.*
- Distribute object model: remote object reference, remote interfaces, remote method invocation (RMI).
- Implementation of RMI: communication module, proxies, remote reference module (remote object table), servants, skeleton, dispatcher.



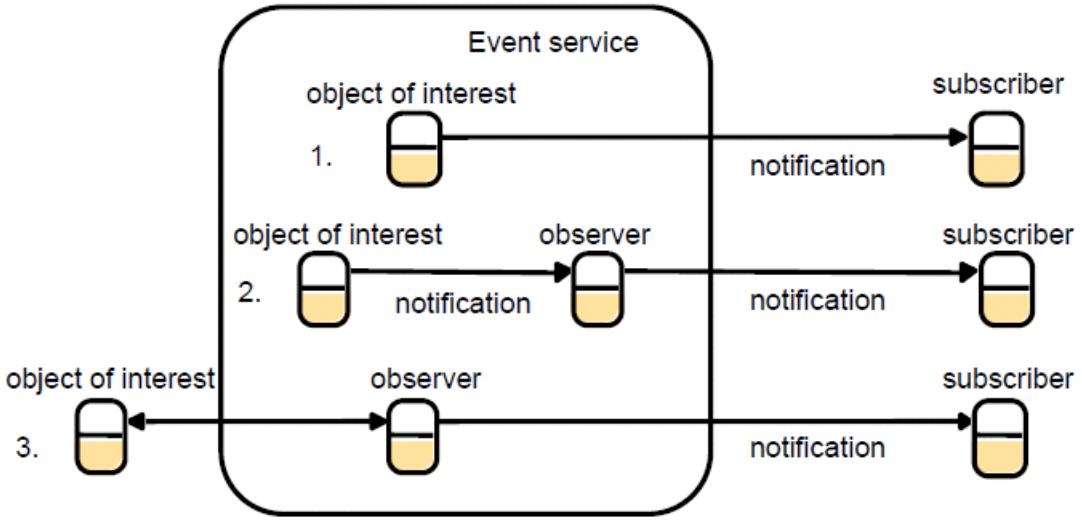
- Binding and activation: binder (look-up service), activation, activator.
- *Object location issues: persistent object stores, object migration, location service.*

Products

- Remote procedure call (RPC): communication, dispatcher, and stub in place of proxy / skeleton.



- Events and notifications:



- Java RMI: $\boldsymbol{RemoteObject > RemoteServer > Activatable\ or\ UnicastRemoteObject > ServantClass}$

Review5. Indirect Communications

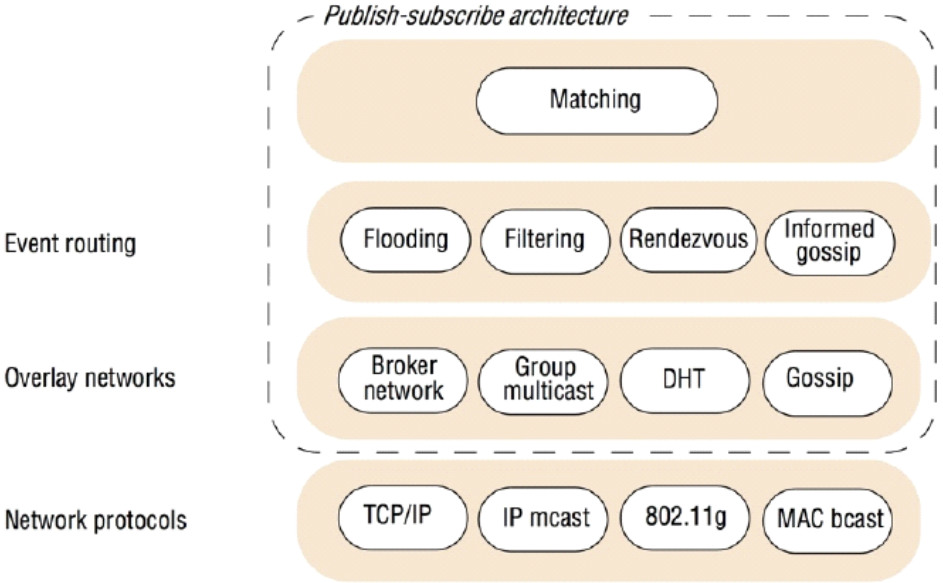
2019年5月29日 11:09

Indirect communication basis

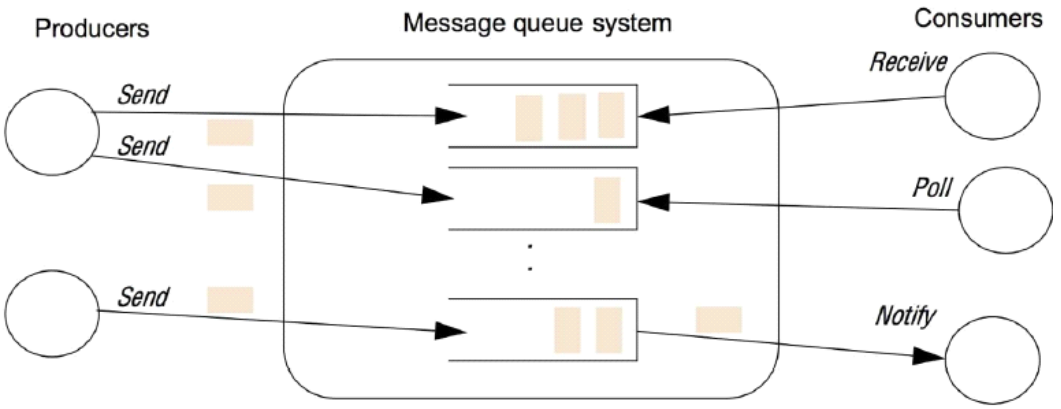
- Space coupling / uncoupling and time-coupled / time-uncoupled.
- Open and closed groups.
- Reliability and ordering: integrity, validity, agreement.
  - *Group management: group membership changes, failure detection, notifying, address expansion.*

Indirect communication styles

- Publish-subscribe paradigm: heterogeneity, asynchronicity, channel based, topic based, content based, type based.
  - *Implementation issues: centralized or distributed.*
  - Architecture: matching, event routing, overlay networks.



    - *Filtering-based routing, rendezvous-based routing.*
- Message queue paradigm:



  - *Example networked topology in WebSphere MQ:* $client \rightarrow proxy \rightarrow client\ channel \rightarrow stub \rightarrow queue\ manager \rightarrow services$
- JMS programming model: connection factory, connection, session, message producer / message consumer, message.
- Distributed shared memory: physical memory mapping to DSM, process accessing DSM, and DSM appears as memory in address space of process.
- Tuple space abstraction: write / read entries as tuples.

Review6. Operating Systems

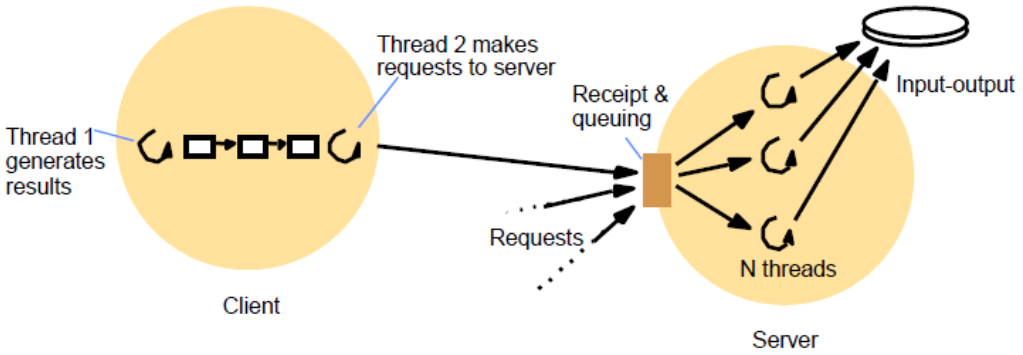
2019年5月29日 11:09

Operating system

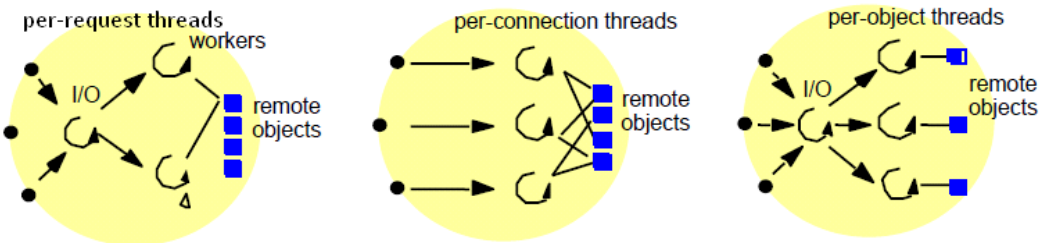
- System layers: applications / services, middleware, OS (kernel, libraries & services), computer & network hardware.
  - $\boldsymbol{Platform = OS + hardware}$
- Core OS functionality: process (thread & space) manage, communication manager (sockets), thread manager (create, synchronisation, scheduling), memory manager, supervisor.
  - Process: a program that is currently executing.
    - $\boldsymbol{Process = execution\ environment + thread(s)}$
  - Thread: lightweight process, OS abstraction of an activity.
  - *Execution environment: address space, thread synchronisation mechanism, communication interface (socket), high level resources (file & window).*
    - Address space: text, stack, heap, region & contents.

Process and thread

- Process: concurrently, CPU sharing, scheduling requires switching of environment.
- Thread: share execution environment, create / destroy dynamically.
  - Advantages: process context switching, threads within a process.
- Role of threads: single CPU system - logically decompose problem; distributed system - remote invocations, disk access, better speed up.



  - *Threads: separate, pass data via buffer, run concurrently, improved speed.*
- Multi-threaded server architectures: worker pool, thread-per-request, thread-per-connect, thread-per-object, physical parallelism.



Review7. Time Global States

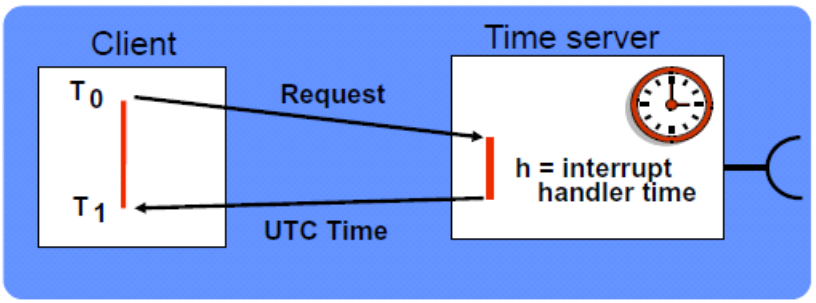
2019年5月29日 11:09

Time service

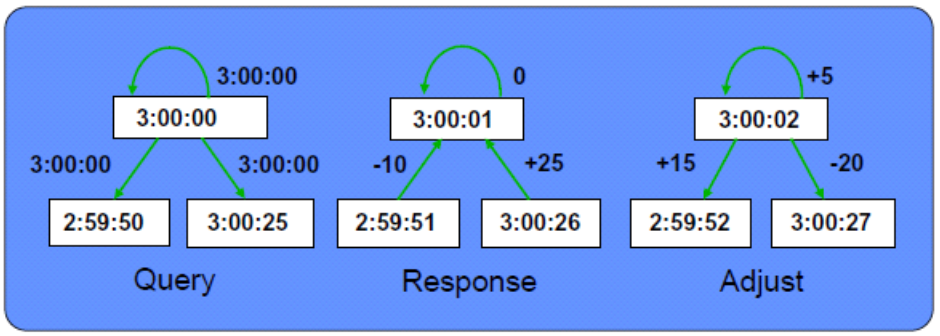
- *Requirements: measure delays, synchronisation, ordering, identify & authenticate.*
- Internal hardware clock: clock skew, clock drift.
  - Source of time: International Atomic Time, Universal Coordinated Time (UTC), Global Positioning System (GPS).
- Clock synchronisation: external & internal, time must never run backwards.

Clock synchronisation algorithms

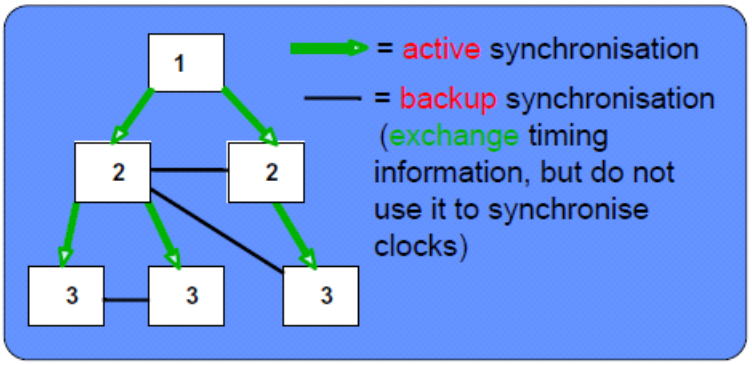
- Synchronisation methods: synchronous systems - known time bounds; asynchronous systems - intranets (Cristian's algorithm, Berkeley algorithm), Internet (the Network Time Protocol).
- Synchronous systems: message delay bounds MAX and MIN, p2 set time to $t + \frac{MAX+MIN}{2}$
  - Clock skew at most $\frac{MAX-MIN}{2}$
- Cristian's algorithm: time server (accurate UTC time)



  - Synchronisation: round-trip shorter than required accuracy, transmission time close to minimum.
  - Agreement protocol: $N > 3f$, f numbers of faulty clocks.
- Berkeley algorithm: master coordinator periodically polls slaves.



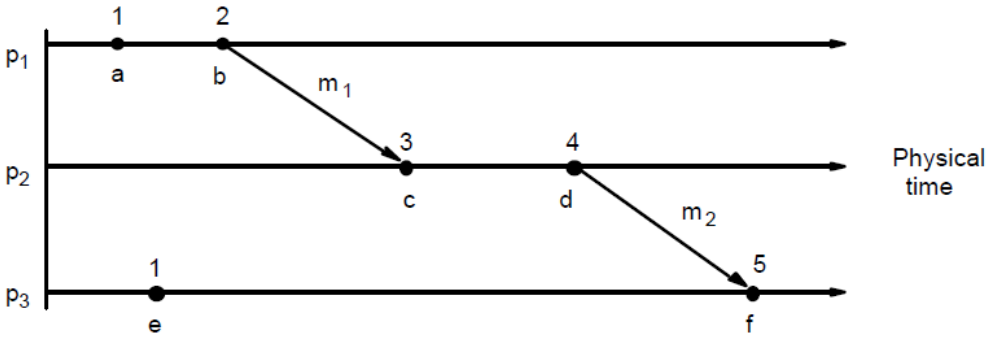
  - Accuracy depends on the round-trip time.
  - Fault-tolerant average: eliminates readings of faulty clocks.
- Network Time Protocol (NTP): multiple time servers across the Internet.



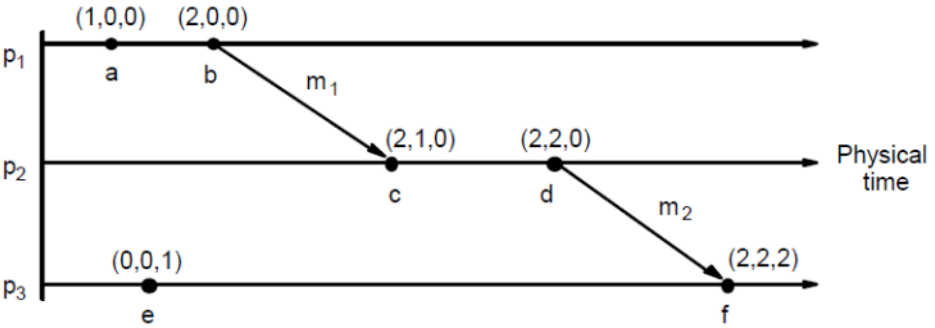
- Primary: directly connected to UTC receivers.
  - *Secondary: synchronise with primaries.*
  - *Tertiary: synchronise with secondary, etc.*
- NTP synchronisation modes: multicast (LAN), procedure call mode, symmetric protocol (pairwise synchronisation).

Logical clocks

- Logical time: causal relationships, agree on same time.
- Event ordering: a occurs before b (in same process or send a receive b).
  - $\boldsymbol{Neither\ a \rightarrow b\ nor\ b \rightarrow a, then\ a \parallel b}$
- Logical clocks (Lamport): monotonically increasing <u>software counter</u>.



- Vector clocks: <u>totally ordered</u> logical clocks.



- VT(b) < VT(c), hence b → c
- neither VT(b) < VT(e), nor VT(b) < VT(e), hence b || e

Review8. Coordination Agreement

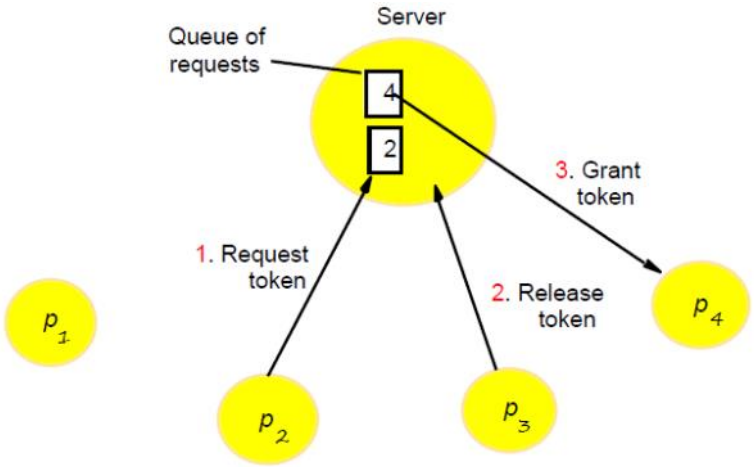
2019年5月29日 11:09

Coordination in distributed systems

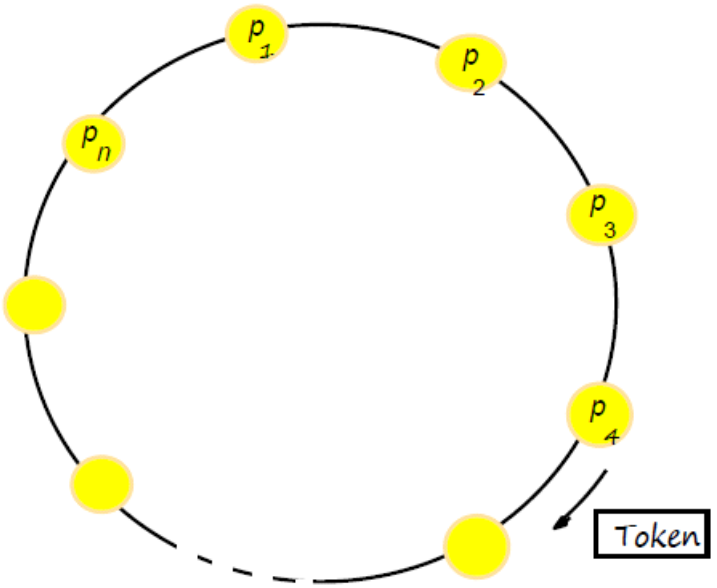
- Resource sharing, agree on actions, dynamically re-assign the role of master.
- *Difficulties: centralised solutions, fixed master-slave arrangements, varying network topologies, failures must be tolerated, impossibility results, mutual exclusion, leader elections, consensus (agreement).*
    - Failure assumptions: reliable links and process crashes, failure detection service, observations of failures (suspected, unsuspected, failed).

Mutual exclusion

- Problem: in asynchronous processes and reliable links, to execute critical section (CS) including enter, resource access, exit.
    - Requirements: one process each time, enter and exit granted, (optional) enter granted at causality order.
- Centralised mutual exclusion:



    - *Doesn't respect causality order.*
- Ring-based algorithm:



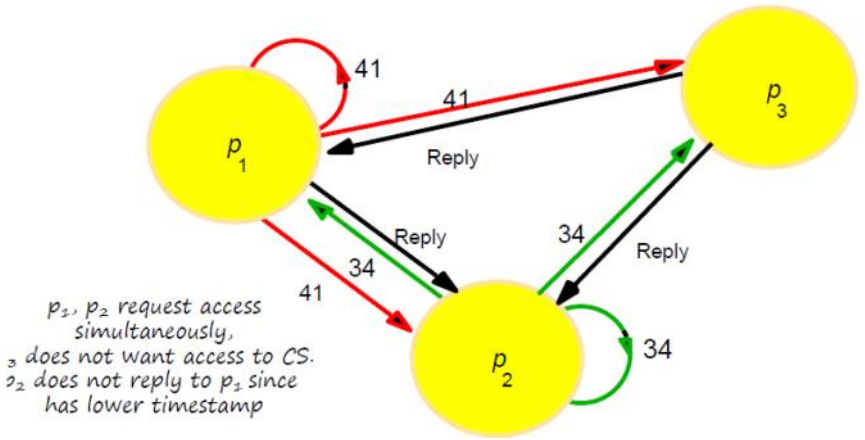
    - *No master server, doesn't respect causality order.*
- Ricart-Agrawala algorithm:

On initialization

On initialization
state := RELEASED;
To enter the critical section
state := WANTED;
Multicast *request* to all processes;request processing deferred here
T := request's timestamp;
Wait until (number of replies received = (*N* – 1));
state := HELD;

On receipt of a request $<T_i, p_i>$ *at* p_j $(i \neq j)$
if (*state* = HELD) or ((*state* = WANTED) *and* $((T, p_j) < (T_i, p_i))$
then
queue *request* from p_i without replying;
else
reply immediately to p_i;
end if
To exit the critical section
state := RELEASED;
reply to any queued requests;

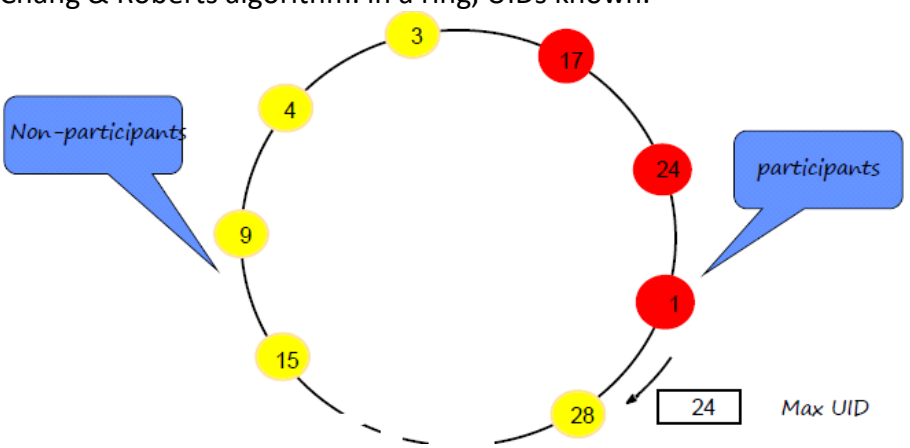
- Based on multicast communication: inter (id, Lamport's logical clock) > multicast to critical section (timestamped) > entry granted.
- *Satisfy causality order, only one message to enter if hardware support multicast.*
- Diagram of multicast mutual exclusion:



- *Summary: performance, fault tolerance, sufficient to obtain agreement.*

Leader election

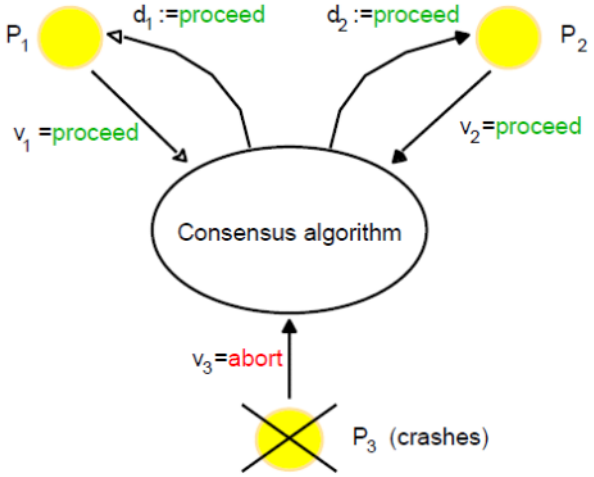
- Requirements: Every process identify P of leader or P is yet undefined, all processes participate and eventually identify leader of maximum UID.
- Chang & Roberts algorithm: in a ring, UIDs known.



- Election: initially each process non-participant, determine leader (election message), announce winner (elected message).
  - *No failure toleration.*

Agreement

- Agreement (consensus) problem: Byzantine generals.
- Consensus algorithm, works in certain failures.



  - Initially - processes in undecided state and an initial value from set D; then - communicate & exchange values, and decide (can't change decision value in decided state).
  - Requirements: termination, agreement, integrity.
- *Byzantine generals: agree to attack or retreat, one commander issues the order, other lieutenants decide, but one or more generals are treacherous.*
  - Requirements: termination and agreement as before, integrity.
  - In synchronous system: basic multicast, admits process crash failures.
  - In asynchronous system: no guaranteed solution.

Review9. Cloud Fundamentals

2019年5月29日 11:09

Cloud computing

- *Based on parallel and distributed systems.*
- *Cloud computing: emerging paradigm, dynamic provisioning, service orientation and virtualisation, lack of standardisation.*
- Basic concept of Internet clouds: submit requests and paid services through clouds (hardware, software, storage, network, service).

Review10. Cloud Architecture

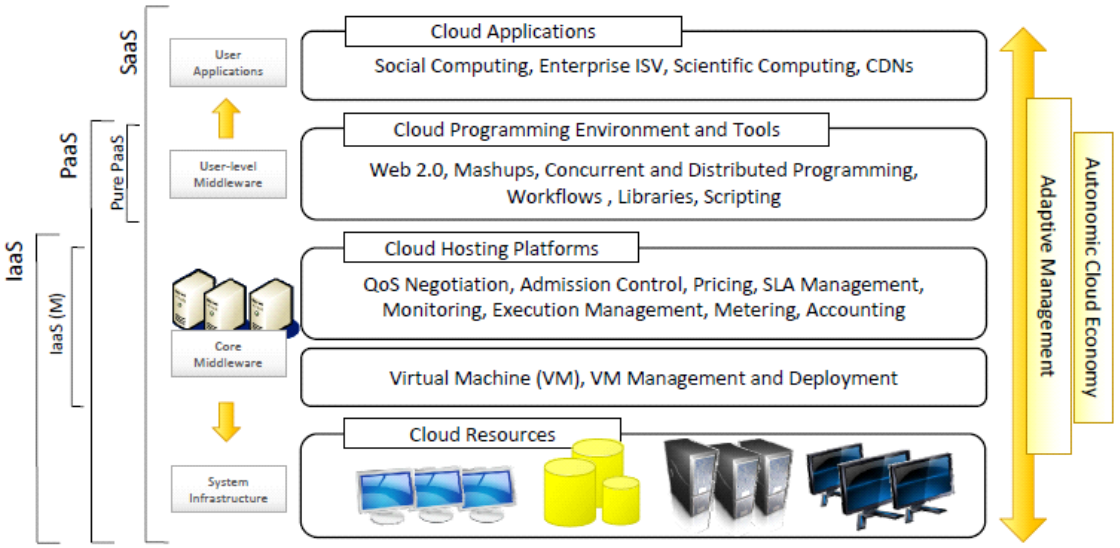
2019年5月29日 11:09

Service oriented computing

- Service: explicit boundaries, autonomous, share schema and contracts, compatibility based on policy.
- Service oriented computing (SOA): architecting several software systems, interoperability, standards, service contracts.
- Web services: prominent technology for implementing SOA.
  - *Web service flow, service discovery, service publication, service description, XML-based messaging, network.*

Cloud computing architecture

- Service oriented architecture:



  - IaaS: infrastructure as a service.
  - PaaS: platform as a service.
  - SaaS: software as a service.
    - Private cloud: hardware and software stack.
    - Hybrid / heterogenous cloud and community cloud.

Review11. Cloud Virtualization

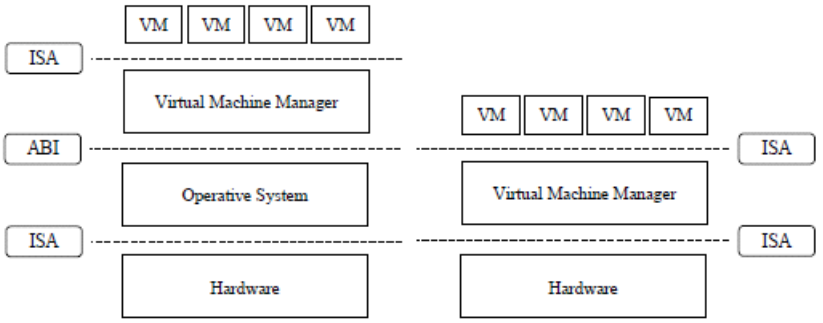
2019年5月29日 11:09

The von Neumann Architecture

- The von Neumann bottleneck and Harvard Architecture.
- Computer components: CPU, memory, I/O devices, buses.
  - Instruction execution: the fetch-decode-execute cycles.
- Multilevel machines: digital logic design and gate level > micro-architecture level > instruction set architecture level (ISA).

Virtualisation

- Host > virtualization layer > guest.
  - Managed execution: sharing, aggregation, emulation, isolation.
- VM-driven infrastructure and cloud-driven infrastructure.
  - Hardware level virtualisation reference model.
  - Hypervisors: type I (native virtual machine) - run directly on top of the hardware; type II (hosted virtual machine)



    - Equivalence, resource control, and efficiency.

Review12. Cloud Advanced Topics

2019年5月29日 11:09

Advanced topics

- Issues: security privacy and trust, massive use of virtualisation.
  - Private clouds and public clouds.
- Security: traditional IT systems relies on service provider, cloud computing environment divides between provider and customer.
- Market-based management.