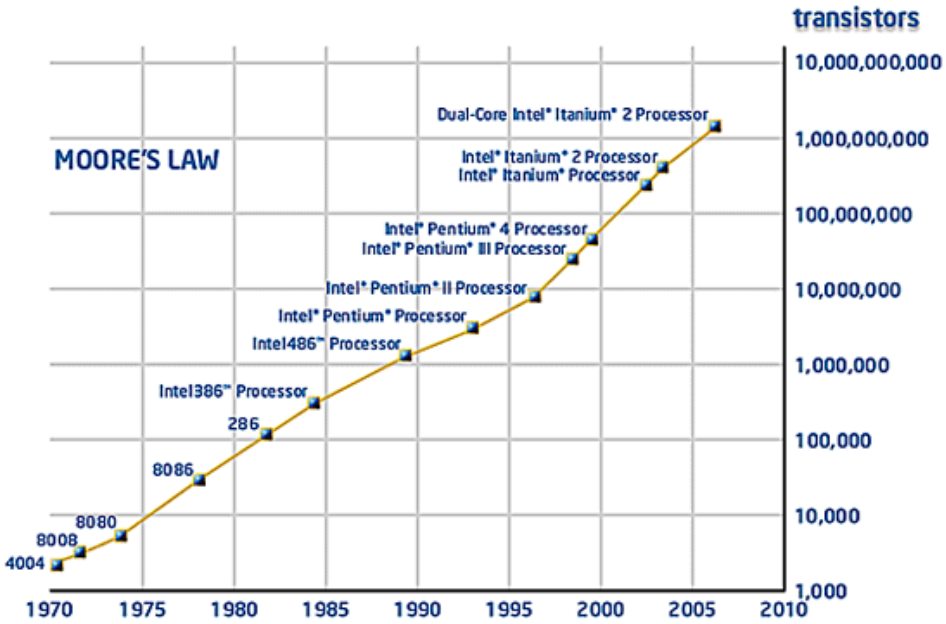
Operativsystem

Multiprocessorsystem och distribuerade system

(kapitel 8)

Varför flere processorer?

- Moores lag: Antalet transistorer på en chip fördubblas inom ca 2 år
- Energiförbrukningen är beroende av i huvudsak två saker
  - Kopplingsström (switching current)
  - Läckström (leakage current)
- Totaleffekten är beroende av
  - Antalet transistorer och klockfrekvens (och spänning)
- Idag är det svårt att öka beräkningskapacitet genom att öka klockfrekvens
  - → ökar antalet transistorer och parallelliserar



seminar.ppt

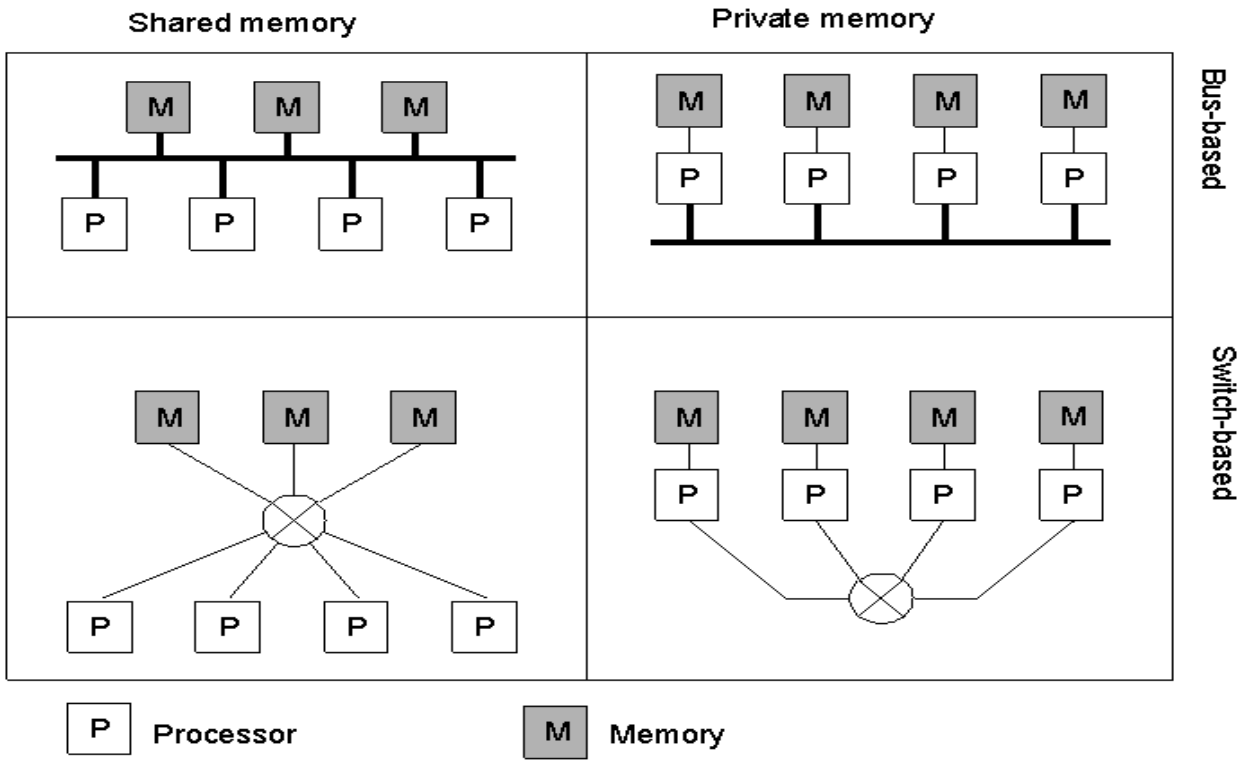
Hardware Concepts

- Computers share memory
  - Multi-processors
- Computer do not share memory
  - Multi-computers
- Architecture of interconnect network
  - Bus-based (eg: cable network)
  - Switched (eg: phone network)
- Multi-computers
  - homogeneous
  - heterogeneous

Multiprocessorsystem

- Nära kopplade (tightly coupled) – någon form av delat minne
  - Lika minnesaccess (uniform memory access – UMA)
  - Icke-lika minnessaccess (non-uniform memory access – NUMA)
- Löst kopplade (loosy coupled) – processorerna har eget minne, meddelande skickas via en meddelandeöverföringskanal
  - Distribuerade system
  - Cluster-beräkningssystem

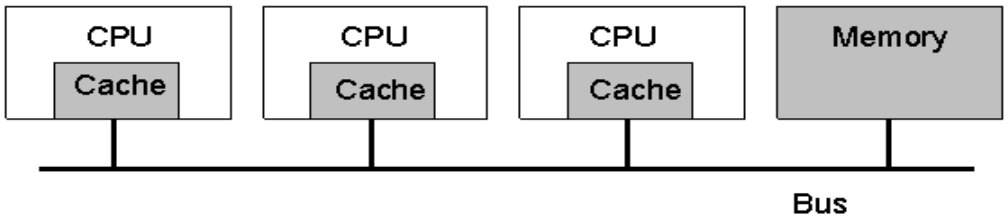
Multiprocessor and Multicomputer



Different basic organizations and memories in distributed computer systems

Multiprocessors

- All CPUs have direct access to shared memory
- Advantage: provides coherency
  - multiple CPUs writing/reading from same address
- Problem: bus may get overloaded
- Solution: place a cache with each CPU
- Problem with cache: coherency



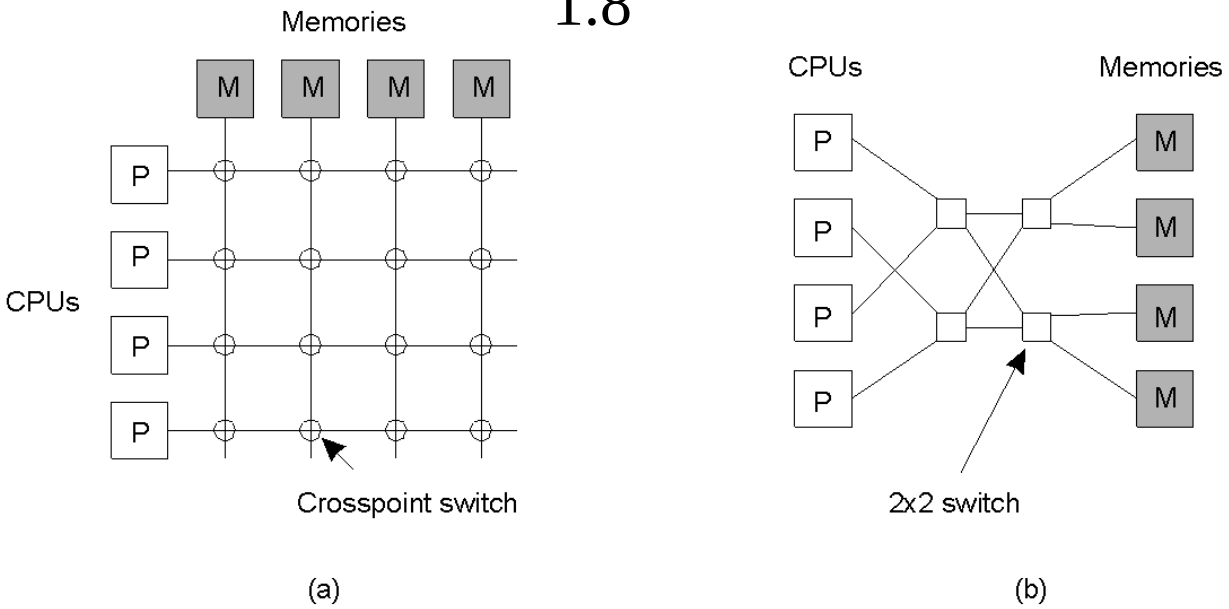
Multiprocessors (2)

Solutions to Cache coherence

a) A crossbar switch

b) An omega switching network

1.8



DOS

- Distributed Operating System
  - tightly coupled
  - for
    - multiprocessors and
    - homogeneous multi-computers
  - Same functionality as a uni-processor OS, except that it runs on multiple CPUs
  - managed as if it were a single system

Multiprocessor OS

- SMP är det populäraste idag
- OS måste beakta
  - Exekveringsenheter (trådar)
  - Synkronisering
  - CPU skedulering
  - Minneshantering
  - Säkerhet och feltolerans

Synkronisering

- Avstängning av avbrott är inte längre tillräckligt
- Mjukvarulösningar för att skapa synkronisering
- För effektiv synkronisering krävs hårdvarustöd (typ test-and-set instruktioner)

Processorskedulering

- Skedulera på process- eller på trådnivå?
- Vilka processorer kan en skedulerbar enhet köras på
  - Hur cache-minneshierarkin ser ut spelar en viktig roll
  - Kan OS göra bra beslut utan hjäl av applikationerna
    - Applikationerna vet hur minne används, OS känner ej till
  - Vilka skeduleringspolicyn kommer OS att erbjuda

Coherence and Consistency

- Coherence: The synchronization of data in multiple caches such that reading a memory location via any cache will return the most recent data written to that location via any (other) cache.
- Consistency Problem: Replication of a resource will result in one copy being different from the others.

More Terminology

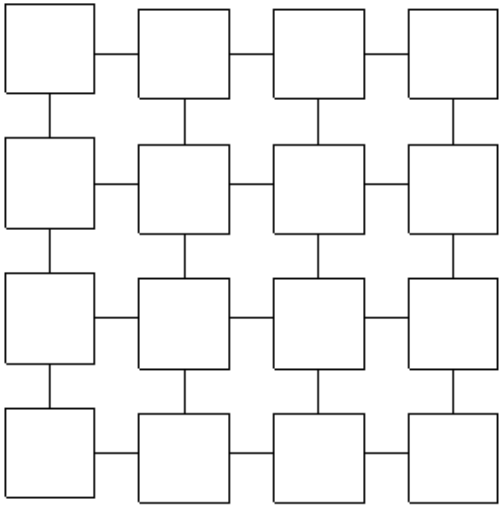
- Multi-user: Many users can run programs on the computer at the same time
- Multi-tasking: More than one program can run at the same time
- Multi-processing: A single program can run on multiple CPUs
- Multi-threading: Different parts of the same program can concurrently run

Homogeneous Multicomputers

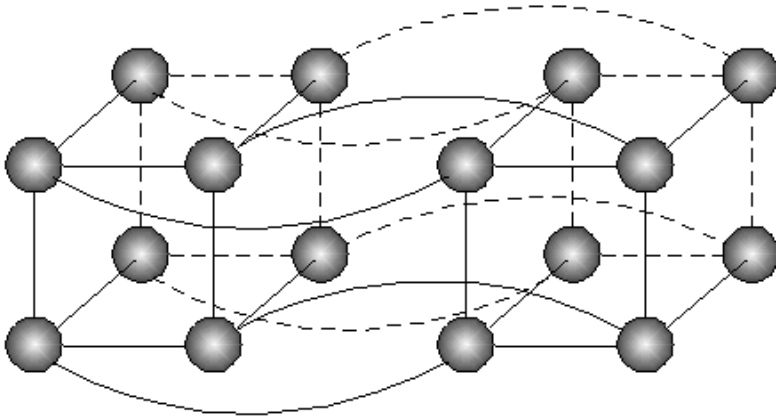
- Messages between processors are routed via a special interconnect network
  - not a broadcast as in a bus based system
- Massively Parallel Processors (MPPs)
  - multi-million dollar supercomputers
  - 1000s of CPUs
  - high performance proprietary interconnection n/w
  - low-latency how bandwidth and fault-tolerance
- Cluster of Workstations (COWs)
  - standard PCs, off-the-shelf interconnects (Myrinet)
  - simple and cheap

Homogeneous Multicomputer Systems

a) Grid
b) Hypercube



(a)



(b)

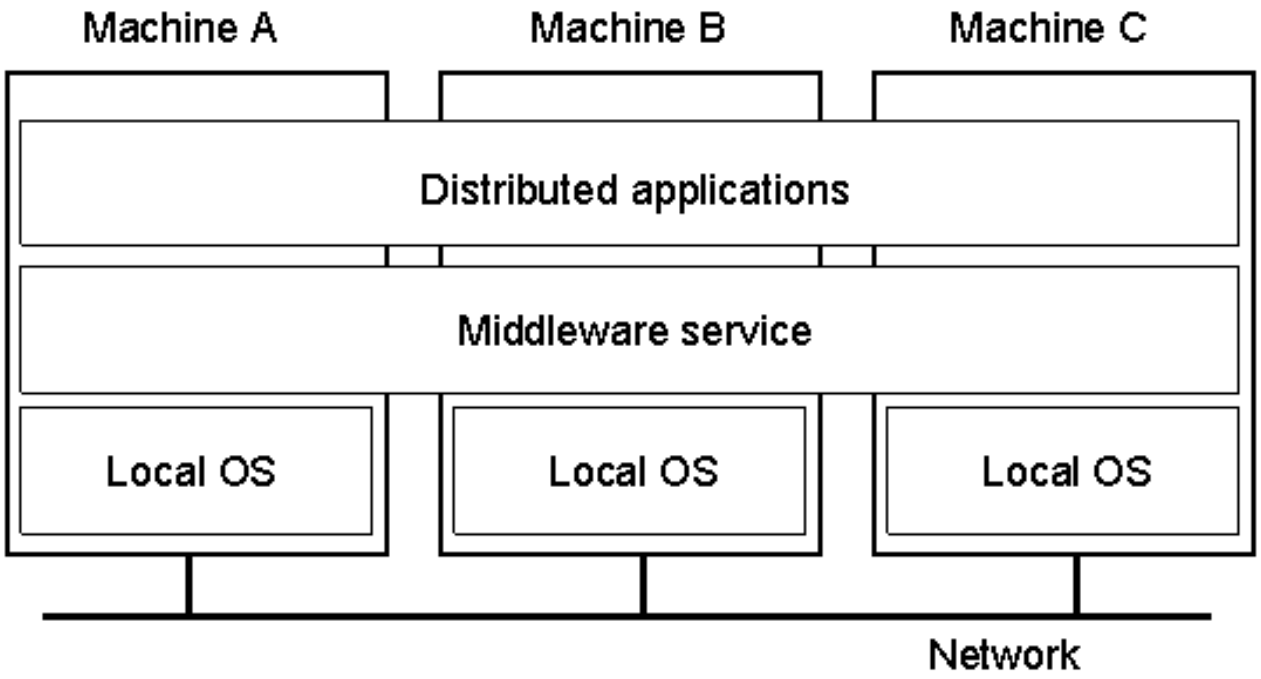
Heterogeneous Multicomputer Systems

- Most distributed systems are built this way
  - computers with different processors, memory sizes and I/O bandwidths
- Eg: each dept has a LAN. All LANs are connected via a backbone
- Large scale heterogeneous multicomputer
  - Grid
  - lacks global system view (can’t expect same performance every-where)
- Need sophisticated s/w on top of heterogeneous multicomputers
- Cloud computing
  - Virtualised computing services, typically over the Internet

Distribuerade system

- Ett distribuerat system är
  - En mängd *oberoende datorer* som som ur användarperspektiv *ser ut* som ett koherent system
- Kommunikationen mellan datorerna är gömd
- Tillåter interaktion på ett konsistent och lika sätt
- Användaren skall inte märka att delar har slutat fungera / blivit utbytta

Distribuerade system



A distributed system organized as middleware.
Note that the middleware layer extends over multiple machines.

Example

- World Wide Web
  - consistent and uniform model for distributed documents
  - just need a URL (Uniform Resource Locator)
    - refers to a local file on the server
- Is the WWW a distributed system?
  - Do users know that documents are located at different locations?
  - What if documents move?

Connect users and resources

- Examples of shared (and remote) resources
  - printers, computers, storage facilities, files, …
- Why?
  - Economics: too expensive to allocate a printer for each user
  - Foster collaboration
    - easily exchange information
- Problems
  - security: eavesdropping, passwords in clear text
  - privacy: tracking preference profiles of online users

Transparency

- Access, Location, Migration, Relocation, Replication, Concurrency, Failure, Persistence
- Transparent distributed system
  - A distributed system that presents itself to users and applications as if it were a single computer
- Access transparency
  - hide byte-ordering representations
  - hide different file naming schemes in various Operating Systems
- Location transparency
  - don’t reveal where the resource is physically located
    - use logical names
    - eg;
    - Does not indicate where the server is running

Transparency: contd

- **Migration transparency**: the way the resource is accessed doesn't change even if the resource is moved
- **Relocation transparency**: doesn't change the way it is accessed even while the resource is being moved
  - Example?
- **Replication transparency**: hide the fact that several copies exist. What are the advantages of replication?

Transparency: contd

- **Concurrency transparency**: shared access to data.
  - Need locking mechanisms to maintain consistency
- **Failure transparency**
  - mask failures of modules
  - "You know you have a distributed system when the crash of a computer you've never heard of stops you from getting any work done." *Leslie Lamport*
- **Persistence transparency**: Is the data in memory or disk?

Transparency: is it always good?

- Request Binghamton newspaper to appear at 8.00 AM local time in your mailbox
  - you are on travel to other part of the globe
- Sending messages between different continents
  - can't make performance same as communication between local machines
- Tradeoff between transparency and performance
  - Accessing a web page that is taking a long time
    - Is the server down? Is the server far-away?

Openness

- Offer services according to standard rules to describe syntax and semantics
  - use an IDL (Interface Definition Language)
- Interoperability
  - two implementations (from different people) can work together by relying on each other's service based on the specification of a common standard
- Portability
  - application developed for a distributed system *A* can be executed without changes on a different system *B* that implements the same interfaces as *A*

Transparency in a Distributed System

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 may be shared by several competitive users
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

Different forms of transparency in a distributed system.

Distributed Algorithms

- Avoid algorithms that
  - collect information from different machines
  - and send to a single machine to be processed
  - disseminate the results back
- Ideally
  - no machine should have the entire state information
  - machines should execute based on local information
  - failure of a machine shouldn’t bring the system down
  - do not assume clocks are synchronized

Scalability Problems

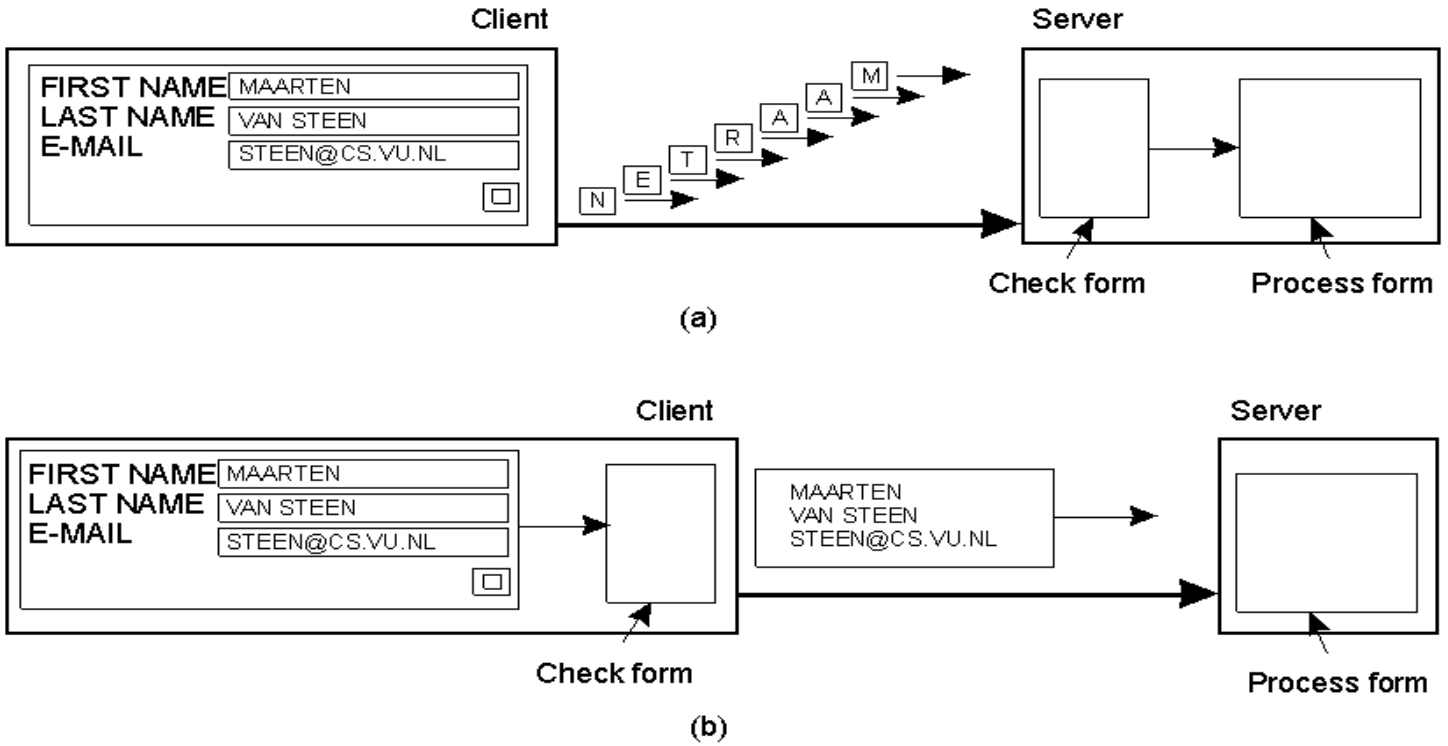
Concept	**Example**
Centralized services	A single server for all users Bottleneck?
Centralized data	A single on-line telephone book Performance if there are millions of numbers to store?
Centralized algorithms	Doing routing based on complete information

Examples of scalability limitations.

Scalability: Improvements

- Buy a bigger machine?
- Asynchronous communication
  - hide communication latency
  - When is asynchronous communication not possible?
- Replication
- Distribute data or algorithms
  - DNS: distributed across several servers
  - no single server to handle all requests
- Send code instead of data (agent technology)
- Cache

Scaling Techniques (1)



The difference between letting:

<1> a server or <2> a client check forms as they are being filled

Software Concepts

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

Multiprocessor Operating Systems

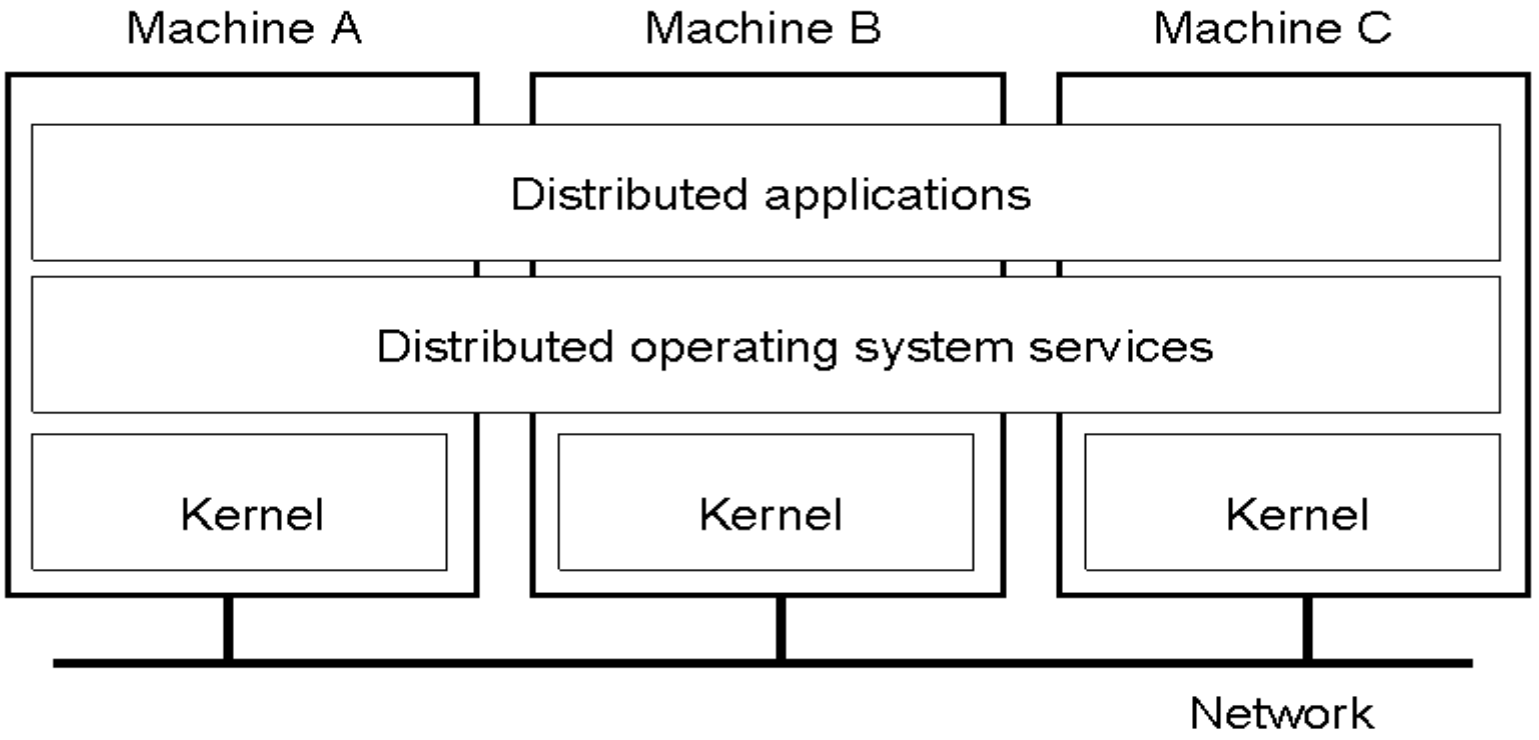
- Shared memory access for multiple processors
- Goal: should be transparent to the user that multiple-CPUs are being used
- Protect data using:
  - semaphores
  - monitors

Multicomputer Operating Systems

- For homogeneous multicomputers
- Data structures for system-wide resource management is no longer shared
- Communication is via message-passing
- Each node has its own kernel
  - and modules for inter-process communication, assigning tasks to processors, masking hardware failures
- Software implementation of shared-memory

Multicomputer Operating Systems (1)

- General structure of a multicomputer operating



Distributed Shared Memory

- Programming with multi-computers requires message passing techniques
- Programming multi-computers is hard
  - compared to multi-processors
  - as can’t use semaphores and monitors
- Solution: emulate shared memory on multi-computers
- Use virtual memory capabilities of each machine to support a large virtual address space
  - Page based Distributed Shared Memory (DSM)

Multiprocessor operativsystem

- Skilld ”supervisor” konfiguration
  - Vanlig i kluster
  - Delar endast begränsade resurser
- Master-slave konfiguration
  - En CPU kör OS, resten kör program
  - OS CPU kan bli flaskhals, eller kan krascha
- Symmetrisk konfiguration (SMP)
  - OS körs på alla processorer
  - Varje processor kan göra alla operationer