Chapter 6
OS Support

Gandeva Bayu Satrya, ST., MT.
Telematics Labz.
Informatics Department
Telkom University
@2014

Outline Today

Chapter 6 – Operating System Support

- OS Layer
- Processes and Threads
- Comm & Invocation
- OS Architecture
- Virtualization at OS Level

Introduction

- Chapter 5 explained the main types of remote invocation found in middleware, such as Java RMI and CORBA,
- with Chapter 6 exploring alternative indirect styles of communication.
- In this chapter we shall focus on support for such remote communication, without real-time guarantees.
- Below the middleware layer is the Operating System (OS) Layer
- The task of any operating system is to provide problem-oriented abstractions of the underlying physical resources – **the processors, memory, networks,** and **storage media**.

OS Support in DS

solaris

Powered By
GNU/Linux

OS X Server
The server for everyone.

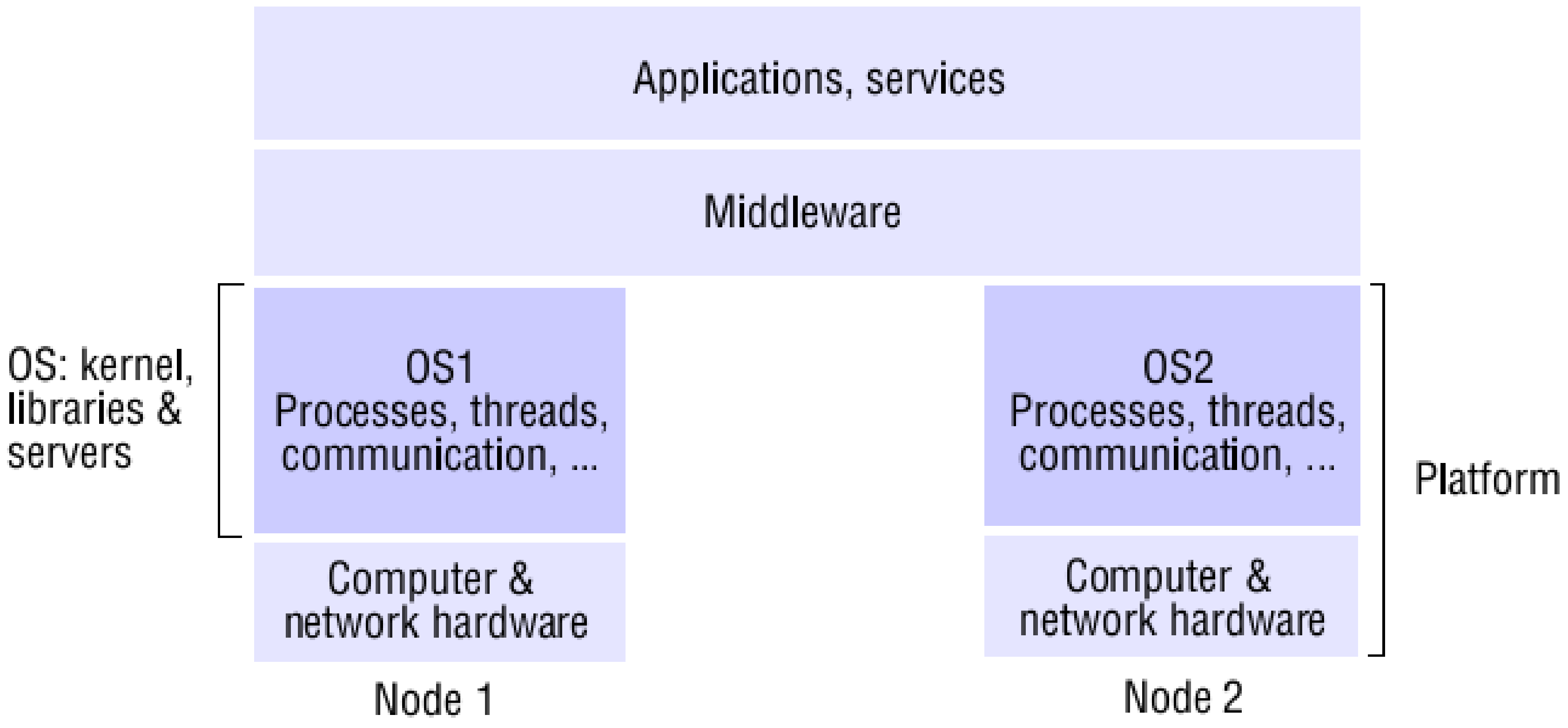
Mac OS X

Macintosh

Windows Server

Windows Server 2012

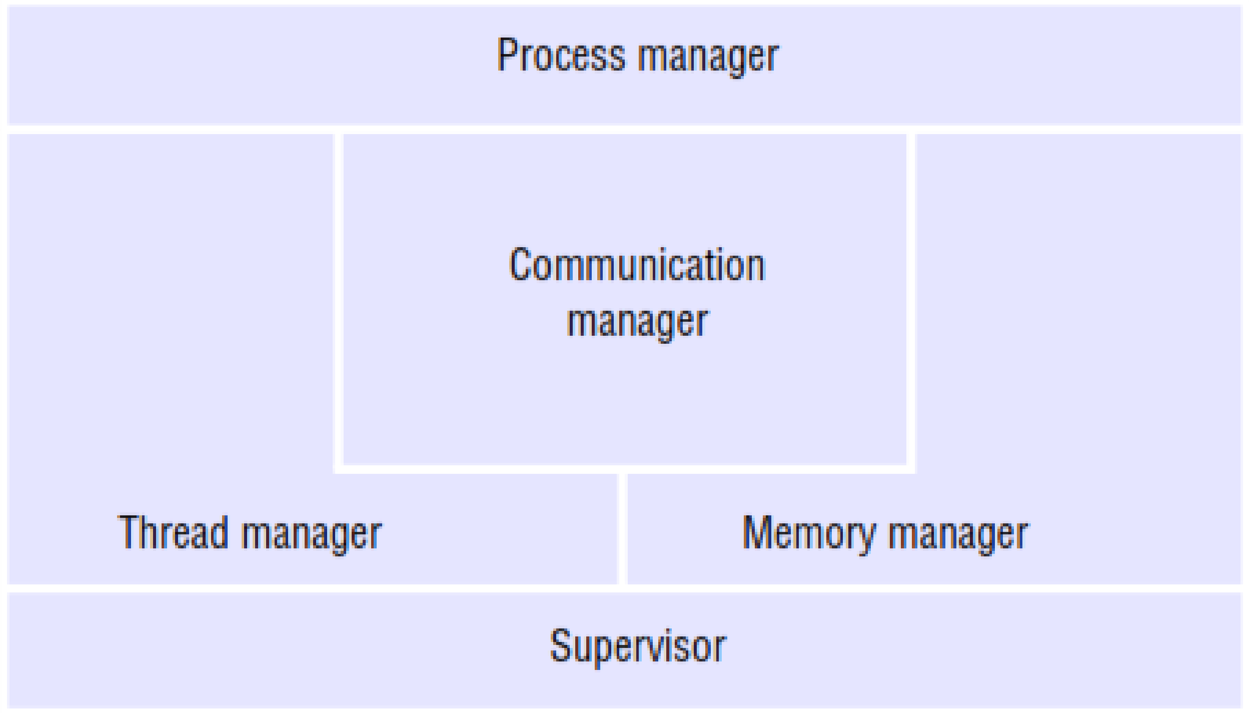
1. OS Layer


Applications, services
Middleware
OS: kernel, libraries & servers
OS1 Processes, threads, communication, ...
OS2 Processes, threads, communication, ...
Platform
Computer & network hardware
Computer & network hardware
Node 1
Node 2

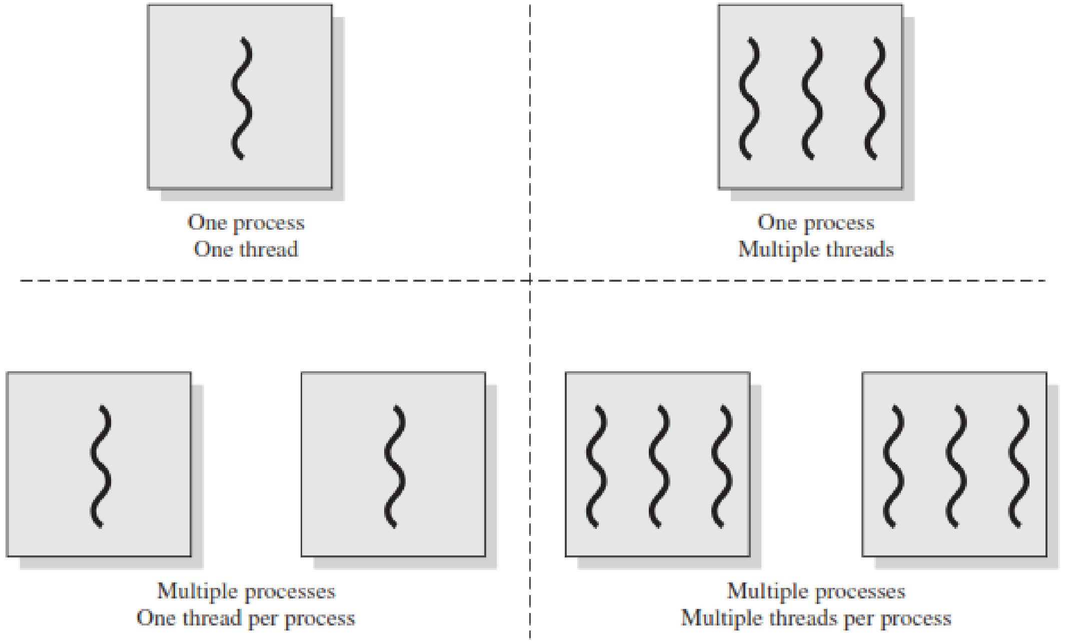
1. OS Layer (con't)

- Our goal in this chapter is to examine the impact of particular OS mechanisms on middleware's ability to deliver distributed resource sharing to users.
- Kernels and the client & server processes that execute upon them are the chief architectural components that concern us.
- Kernels and server processes are the components that manage resources and present clients with **an interface to the resources**.
- Require at least the following of them:
    a) *Encapsulation*: a set of operations that meet their clients' needs.
    b) *Protection*: being read by users without read permissions.
    c) *Concurrent processing*: share resources and access them concurrently.

Core OS Functionality



2. Processes & Threads



Process : A program in execution or An instance of a program running on a computer

Thread : lightweight process

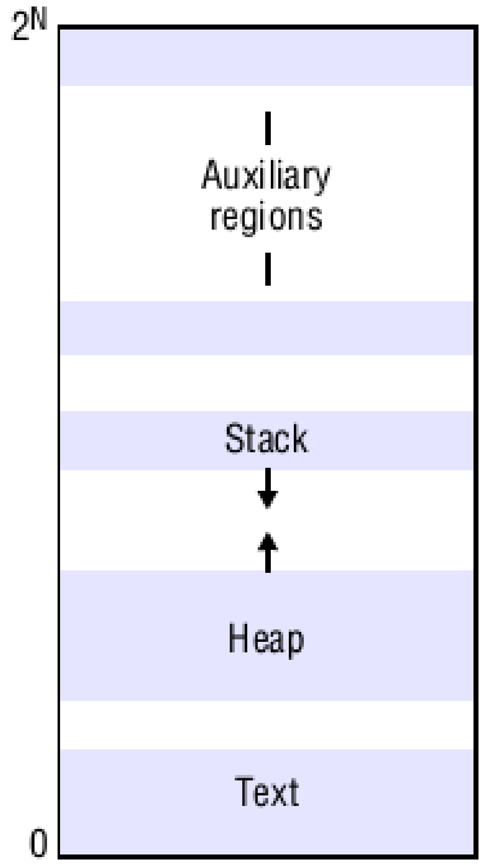
2. Processes & Threads (con't)

An execution environment **is** the unit of resource management: a collection of local kernel-managed resources to which its threads have access.

An **execution environment** primarily consists of:

- ✓ an address space;
- ✓ thread synchronization and communication resources such as semaphores and communication interfaces (for example, sockets);
- ✓ higher-level resources such as open files and windows.

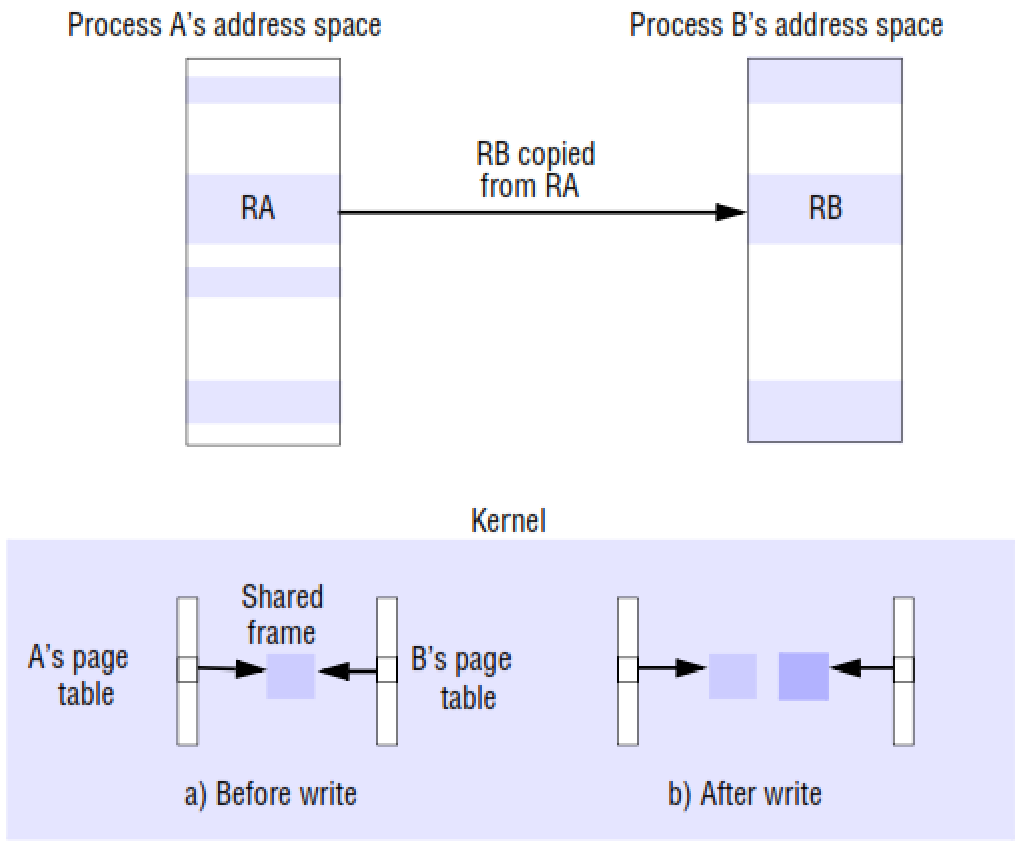
a. Address Spaces



- A region **is** an area of contiguous virtual memory that is accessible by the threads of the owning process.
- Regions **do not** overlap.

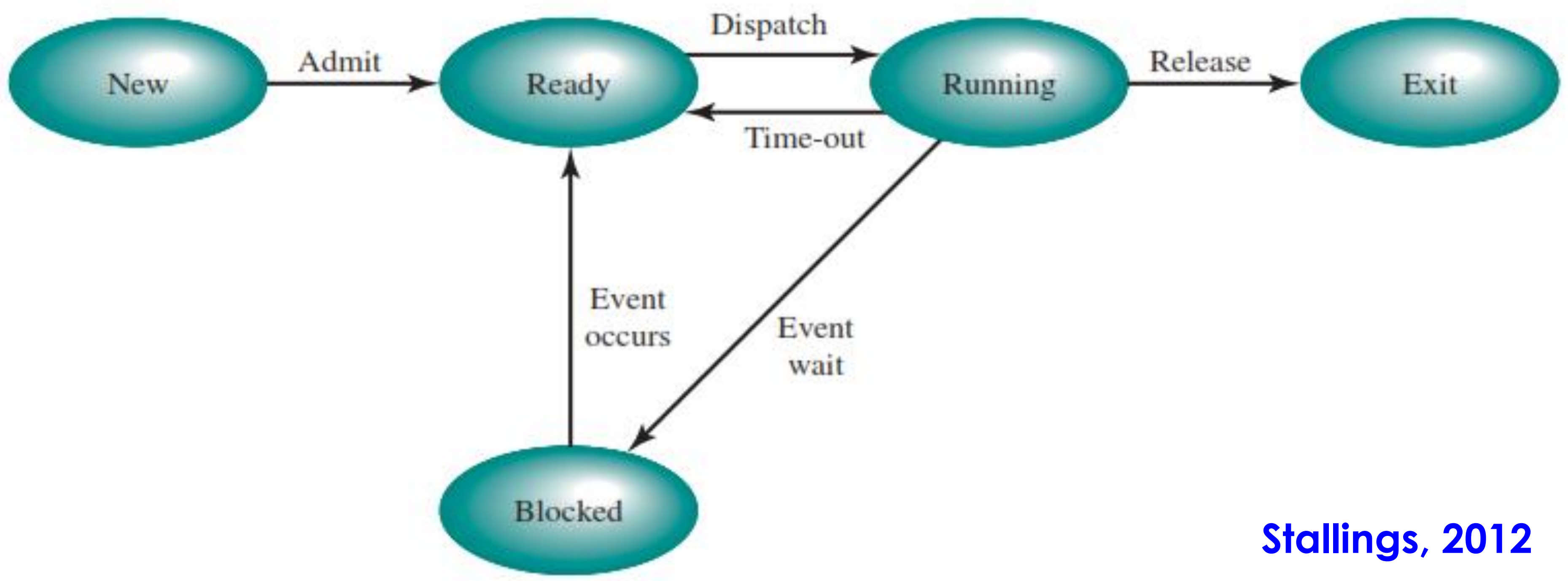
- **Each region** is specified by the following properties:
  a) its extent (lowest virtual address and size);
  b) read/write/execute permissions for the process's threads;
  c) whether it can be grown upwards or downwards.

b. Creation of a New Process



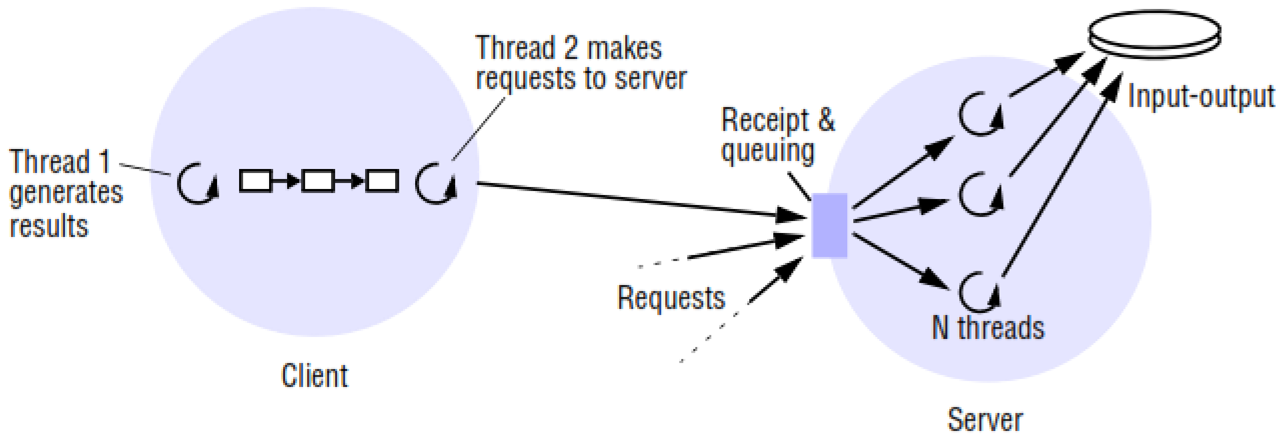
- For a distributed system, the design of the process-creation mechanism has to take into account the utilization of multiple computers
- The creation of a new process can be separated into
  - a) the choice of a target host,
  - b) the creation of an execution environment

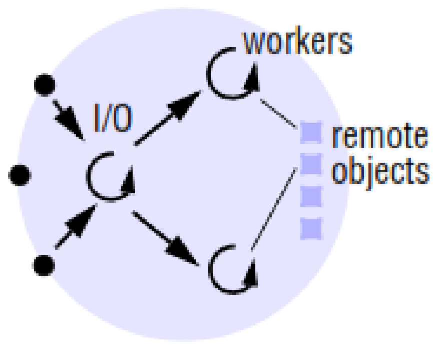
Five-State Process Model



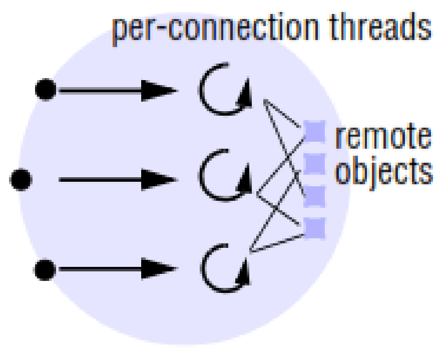
Stallings, 2012

c. Threads

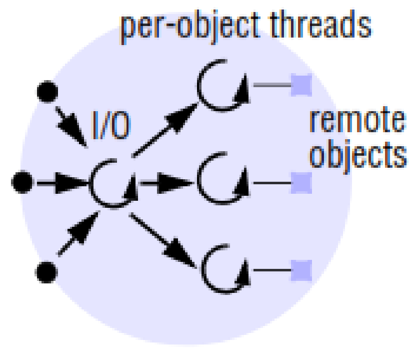




a. Thread-per-request

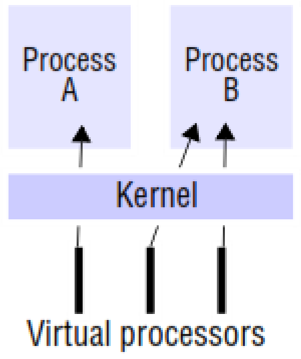


b. Thread-per-connection

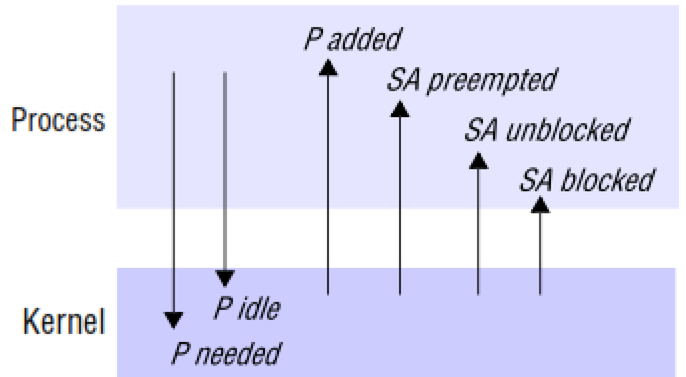


c. Thread-per-object

- ✓ **Architectures for multi-threaded servers** : the number of requests processed per second
- ✓ **Threads within clients** : to be passed and to continue computing further results
- ✓ **Threads versus multiple processes** : threads are cheaper to create and manage than processes



A. Assignment of virtual processors to processes



B. Events between user-level scheduler & kernel
Key: P = processor; SA = scheduler activation

- ✓ Threads programming
- ✓ Thread lifetimes
- ✓ Thread synchronization
- ✓ Thread scheduling
- ✓ Threads implementation

3. Comm & Invocation

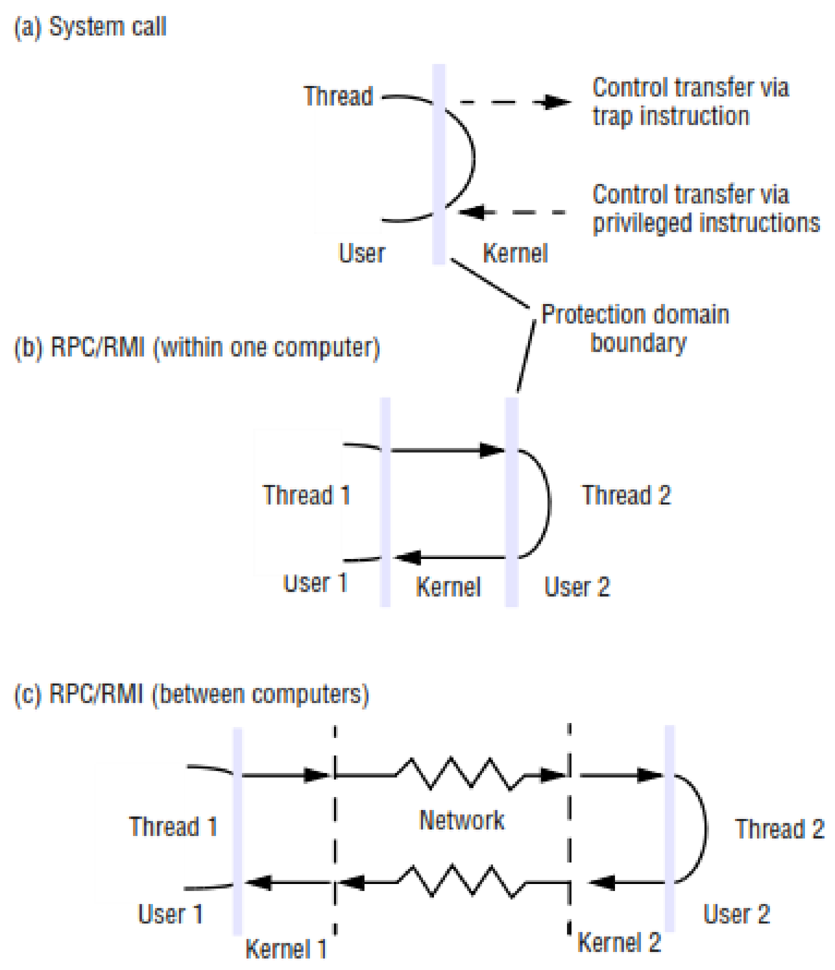
Communication primitives :

- Some kernels designed for distributed systems have provided communication primitives tailored to the types of invocation that Chapter 5 described.

Protocols and openness :

- One of the main requirements of the operating system is to provide standard protocols that enable interworking between middleware implementations on different platforms.
- notably Amoeba RPC [van Renesse 1989], VMTP [Cheriton 1986] and Sprite RPC [Ousterhout 1988]

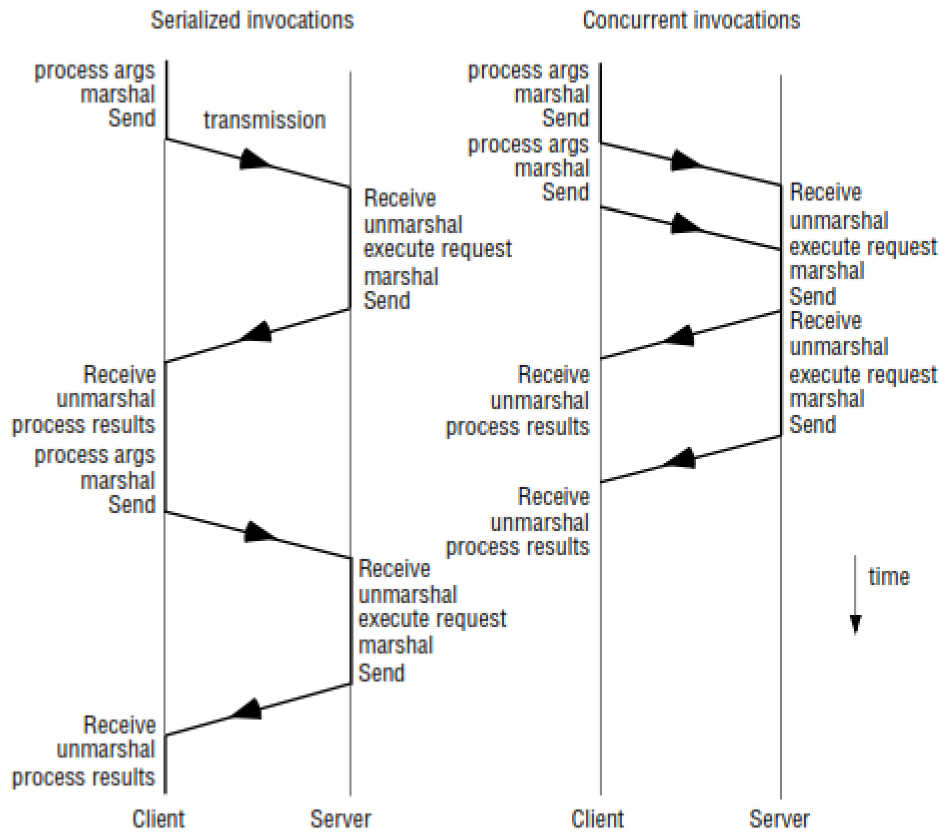
a. Invocation Performance


(a) System call
Thread
Control transfer via trap instruction
Control transfer via privileged instructions
User
Kernel
Protection domain boundary
(b) RPC/RMI (within one computer)
Thread 1
Thread 2
User 1
Kernel
User 2
(c) RPC/RMI (between computers)
Thread 1
Network
Thread 2
User 1
Kernel 1
Kernel 2
User 2

- **Invocation performance** is a critical factor in distributed system design.
- The more designers separate functionality between **address spaces, the more remote invocations** are required.

- **Invocation costs** : making a system call, sending a message, remote procedure calling and remote method invocation
- **Invocation over the network** : A null RPC (and similarly, a null RMI) is defined as an RPC without parameters that executes a null procedure and returns no values.

b. Asynchronous Operation



- In **the Internet environment**, the effects of relatively high latencies, low throughput and high server loads may outweigh any benefits that the OS can provide.

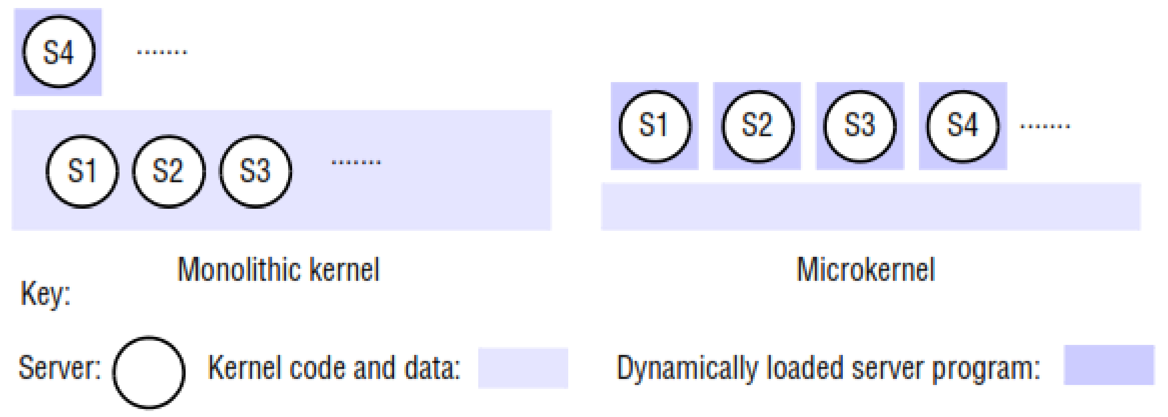
- A common technique to defeat high latencies is **asynchronous operation**, which arises in two programming models: **concurrent invocations** and **asynchronous invocations.**

4. OS Architecture

- We adopt a first-principles approach of starting with the requirement of **openness** and examining the major **kernel architectures** that have been proposed

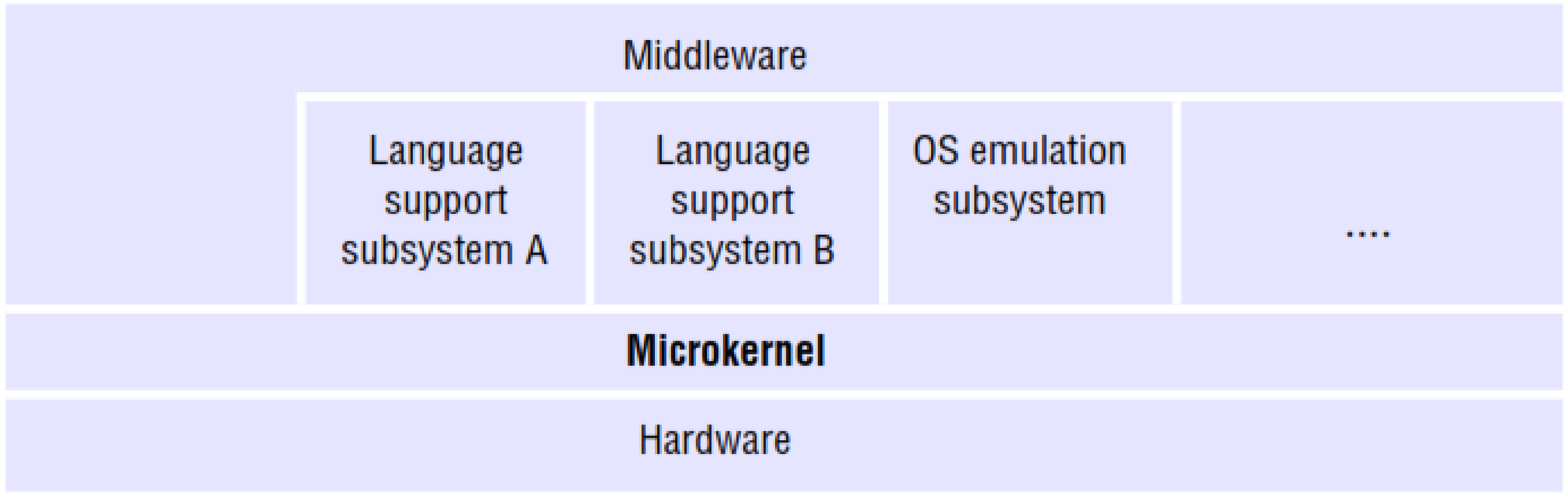
- An open distributed system should make it possible to:
  a) run only that system software at each computer that is necessary for it to carry out its particular role in the system architecture
  b) allow the software (and the computer) implementing any particular service to be changed independently of other facilities;
  c) allow for alternatives of the same service to be provided, when this is required to suit different users or applications;
  d) introduce new services without harming the integrity of existing ones.

a. Monolithic Kernels and Microkernels



- The chief advantages of a **microkernel-based** operating system are its extensibility and its ability to enforce modularity behind memory protection boundaries.
- The advantage of a **monolithic design** is the relative efficiency with which operations can be invoked. System calls may be more expensive than conventional procedures, but even using the techniques we examined in the previous section, an invocation to a separate user-level address space on the same node is more costly still.

The Role of The Microkernel



The microkernel supports middleware via subsystems

b. Some Hybrid Approaches

- Two of the original microkernels, Mach [Acetta 1986] and Chorus [Rozier 1990], began their developmental life running servers only as user processes
- **The SPIN operating system design** [Bershad 1995] finesses the problem of trading off efficiency for protection by employing language facilities for protection.
- **Operating systems such as Nemesis** [Leslie 1996] exploit the fact that, even at the hardware level, an address space is not necessarily also a single protection domain.
- Some **later Kernel Designs**, such as L4 [Härtig 1997] and the Exokernel [Kaashoek 1997], L4 is a 'second-generation' microkernel design that forces dynamically loaded system modules to execute in user level address spaces, but optimizes interprocess communication to offset the costs of doing so.

5. Virtualization

- Virtualization is an important concept in distributed systems.
- Virtualization is also applied in the context of operating systems.

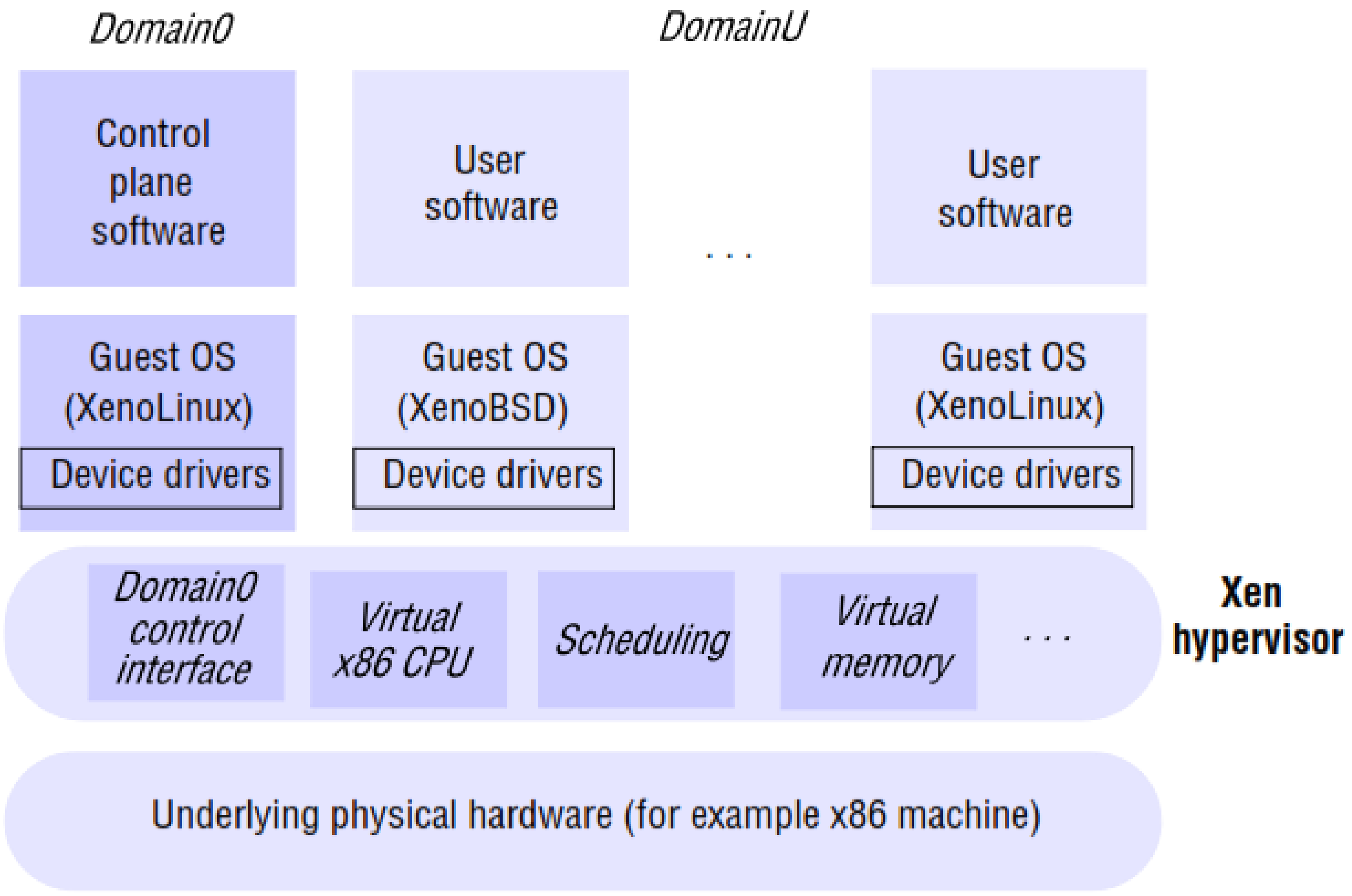
- Separation between having a single CPU and being able to pretend there are more can be extended to other resources as well, leading to what is known as resource virtualization. [**TAN'07**]

- The virtualization of resources plays a key role in achieving the required degree of adaptability. Therefore, the term virtualization is heard in many areas, including the virtualization of servers, applications, storage devices, security appliances, and, not surprisingly, the network infrastructure. [**MOR'06**]

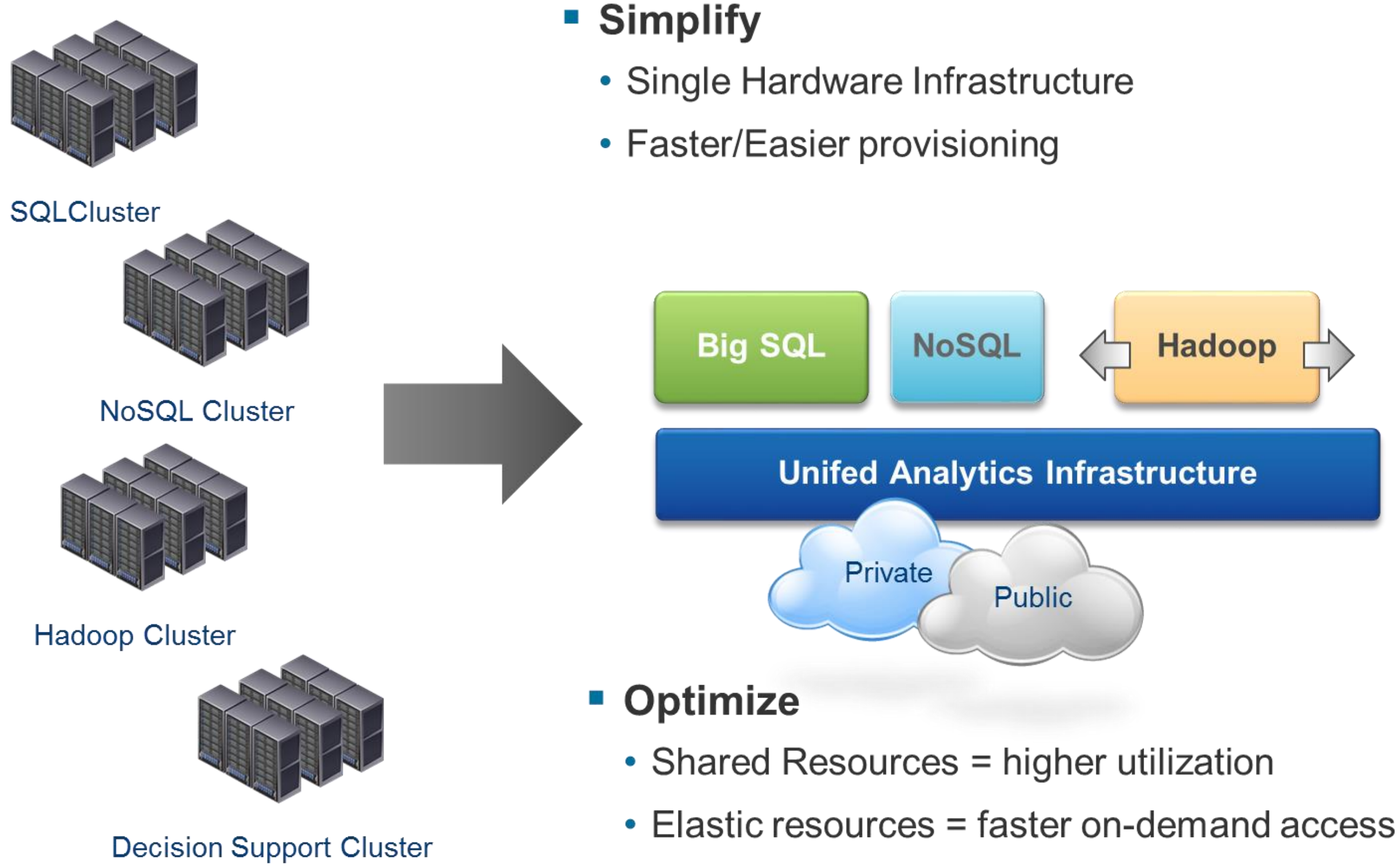
a. System Virtualization

- The Goal of system virtualization is to provide multiple virtual machines (virtual hardware images) over the underlying physical machine architecture.
- Multiple instances of the same operating system can run on the virtual machines or a range of different operating systems can be supported.
- The virtualization system allocates the physical processor(s) and other resources of a physical machine between all virtual machines that it supports.
- System virtualization is implemented by a thin layer of software on top of the underlying physical machine architecture; this layer is referred to as a ***virtual machine monitor*** *or* ***hypervisor***.

b. The Xen

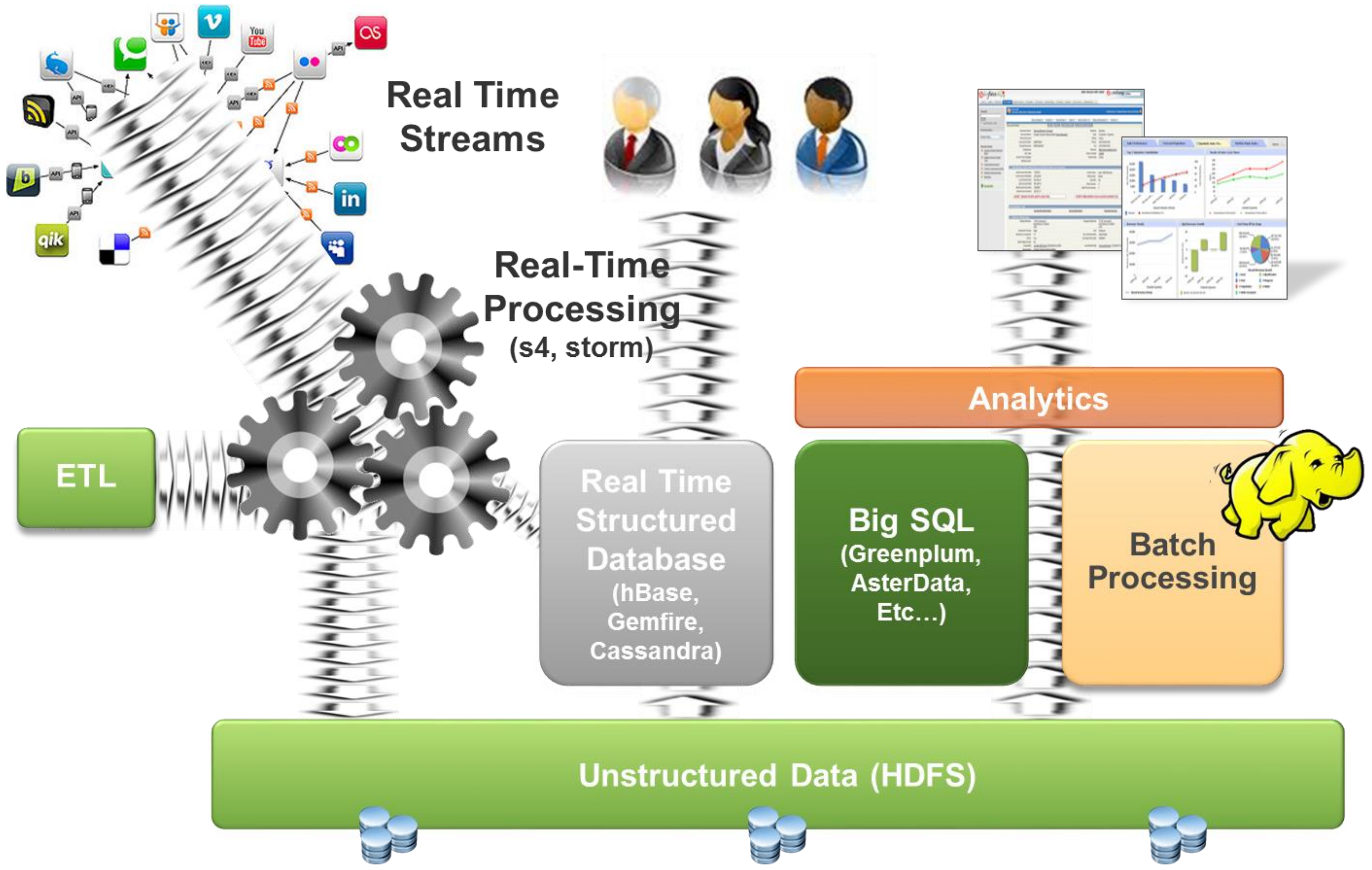

Domain0
DomainU
Control plane software
User software
. . .
User software
Guest OS (XenoLinux)
Device drivers
Guest OS (XenoBSD)
Device drivers
Guest OS (XenoLinux)
Device drivers
Domain0 control interface
Virtual x86 CPU
Scheduling
Virtual memory
. . .
Xen hypervisor
Underlying physical hardware (for example x86 machine)

c. HDFS (Hadoop Distributed File System)



- **Simplify**
  - Single Hardware Infrastructure
  - Faster/Easier provisioning
- **Optimize**
  - Shared Resources = higher utilization
  - Elastic resources = faster on-demand access

A Holistic View of a Big Data System


Real Time Streams
Real-Time Processing (s4, storm)
ETL
Real Time Structured Database (hBase, Gemfire, Cassandra)
Analytics
Big SQL (Greenplum, AsterData, Etc…)
Batch Processing
Unstructured Data (HDFS)

References

[COU'12]	Coulouris, G. Dollimore, J., Kindberg, T., Blair, G., DISTRIBUTED SYSTEMS : Concepts and Design Fifth Edition, Pearson Education, Inc., United States of America, 2012.
[MOR'06]	Moreno, V., Network Virtualization, Cisco Press, USA, July 19 2006.
[STA'12]	Stallings, W., OPERATING SYSTEMS INTERNALS AND DESIGN PRINCIPLES Seventh Edition, Pearson Education, Inc., New Jersey, 2012.
[TAN'07]	Tanenbaum, A.S., Steen, M.V., DISTRIBUTED SYSTEMS : Principles and Paradigms Second Edition, Pearson Education, Inc., United States of America, 2007.

Thank You

Gandeva Bayu Satrya, ST., MT.
Telematics Labz.
Informatics Department
Telkom University
@2014