

Providing a Single System Image: The GENESIS Approach

Andrzej M. Goscinski

{ang@deakin.edu.au}

School of Information Technology

Deakin University

Overview

- Clusters
- Parallel processing on clusters
- Single System Image
- How to Achieve SSI?
- Toward SSI clusters
- Genesis provides a SSI
- What is Offered by the Best Known Systems?
- Conclusion

Clusters

They are everywhere!

Definition of a Cluster

- A cluster is a type of parallel system that:
 - Consists of a collection of interconnected whole computers
 - We would like to use it as a single unified computing resource
- A “whole computer” could be a uni-processor or a SMP
- Trends that promote clusters
 - Very high-performance microprocessors, i.e. the need for massive parallelism is decreasing
 - The communication technology is improving rapidly, e.g. fiber channels, Gbit networks etc.
 - Standard tools and protocols for distributed computing, e.g. TCP/IP

Reasons for Using Clusters

- Performance
- Non-stop operation (also called availability)
- Price/performance – clusters consist of standard computers and networks
- Incremental growth – one can incrementally extend the system by adding more computers
- Scaling – there is no upper limit on the number of computers in cluster, as opposed to the maximum number of processors in an SMP
- Scavenging – turn the idle time on organization's computers into something useful

Clusters of Computers - Disadvantages

- Distribution of resources (CPUs and peripherals)
- Poor and difficult to use software (operating systems & runtime systems)
- Unreliable!
 - Clusters are usually shared
 - Low cost components can fail
 - External influence easy

Parallel Processing on Clusters

Researchers and manufacturers mainly
concentrate on execution performance

Parallel Processing - Problems

- Application developers are discouraged as they have to
 - program many activities of an operating system nature
 - deal with distribution of resources as their location is visible
 - deal with process instantiation and their placement
 - use one single communication/programming paradigm
 - deal with computer/parallel process failure (even to restart the application)
 - deal with adding, removing computers to/from clusters
 - overcome poor and difficult to use software (operating systems and runtime systems)
 - face user unfriendliness and the lack of ease of use
 - limited number of software products currently support clusters

Moving Parallel Computing on Clusters to the Computing Mainstream

- Clusters could easily support coarse grain parallelism
 - Data parallelism
 - Functional parallelism
- High performance is only one part of a story
- The second part is created by users who do not wish to deal with all the problems
- **The problem with clusters is not hardware; it is software**
- A need of a major breakthrough like the Web for the Internet
- **Question:** what to do to overcome these problems and move parallel computing to mainstream computing?
- **Answer:** provide a Single System Image (SSI)

Single System Image

Make life of application developers easy

Single System Image (SSI)

- SSI is the illusion, created by software or hardware, that presents a collection of computing resources as one, powerful resource
- SSI makes the cluster appear like a single machine to the user, to applications, and to the network
- The provision of a SSI means that
 - application developers see the whole cluster as a single powerful and reliable computer rather than as a set of personal computers connected by a network, and
 - where resources are added removed or reconfigured automatically

Benefits of SSI

- Use of system resources transparent
- Transparent process migration and load balancing across all cluster computers
- Improved reliability and high availability
- Improved system response time and performance
- Simplified system management
 - Single file hierarchy, memory space, job management,
- Reduction in the risk of operator errors
- No need to be aware of the underlying system architecture to use these machines effectively

SSI Levels

- SSI levels of abstractions:

Application and Subsystem Level

Operating System Kernel Level

Hardware Level

- Every SSI has a boundary
- Single system support can exist at different levels within a system, one able to be build on another

Software-based Single System Image

- A SSI cluster can be achieved if there is a SSI operating system and/or runtime system
- They should provide:
 - Transparency
 - High availability
 - Parallelism management
 - Fast IPC
 - Communication / programming paradigms
 - Fault tolerance

How to *Achieve* a SSI?

Transparency

- Users should see a cluster as a single powerful computer
- Dimensions of parallel processing transparency
 - Location transparency
 - Process relation transparency
 - Execution transparency
 - Device transparency
 - Failure transparency

High Availability

- There is a need to handle addition, removal and reorganization of system resources transparently
- These features can be achieved if a virtual parallel computer is established automatically and dynamically
 - Provide resource discovery service
 - Identify computers and peripheral devices automatically
 - Identify and record lightly loaded and idle computers and devices
 - Identify and record faulty computers and peripheral devices
 - Record events (e.g., process completion, additional load)
 - Provide availability service
 - Collect information from all computers about computation and communication pattern and volume
 - Use information about application requirements
 - Pass information to the Global Scheduler

Parallelism Management

- Present operating systems that manage clusters are not built to support parallel processing
- **Reason:** these operating systems do not provide services to manage parallelism
- Parallelism management is the management of parallel processes and computational resources
 - Achieve high performance
 - Use computational resources efficiently
 - Make programming and use of parallel systems easy

Services for Parallelism Management on Clusters

- Establishment of a virtual machine
- Mapping of processes to computers
- Parallel processes instantiation
- Data (including shared) distribution
- Initialisation of synchronization variables
- Coordination of parallel processes
- Dynamic load balancing

IPC - Communication Paradigms

- Message Passing – Explicit communication among processes of a parallel application
 - Fast
 - Difficult to use for programmers
- Distributed Shared Memory – Implicit communication among processes of a parallel application through shared memory objects
 - Easy to use
 - Demonstrates reduced performance
- **Claim:** MP and DSM should be provided as a part of a cluster operating system as they manage system resources

Group Communication in Clusters

- Communication sometimes involves more processes
 - Replicated services
 - Parallel applications
 - Cluster operating system management
- A group is a collection of processes that act together in some system or user-specified way
- Group communication is the problem of how to communicate with, and within, a group of processes
- Rich semantics to support programmers
 - Group addressing
 - Reliable vs. unreliable communication
 - Ordering, delivery and response semantics
 - Closed vs. open and overlapping groups
 - Static vs. dynamic group membership

Some Options for Fault Tolerance

- Backward or Forward Error Recovery
- Operating system service or middleware?
- Transparent or user-driven?
- High failure-free execution overhead or high overhead during recovery?
 - Both? Neither?
- Process Replication or Fault Tolerance?
- Transactions?

Checkpointing

- Checkpointing a single process
 - Requires that a complete image of the process must be saved
 - Recovery involves the total restoration of this image to an active state
- Checkpointing a parallel application
 - Multiple processes with interdependent states
 - Checkpointing and recovery must effectively capture/re-establish the distributed state of the parallel application

Towards SSI Clusters

Cluster Middleware / Underware

Applications

PVM / MPI / DSM

Middleware – application level
Underware – kernel level



Hardware/OS

Middleware - Advantages

- Middleware makes the system quickly portable, tracks vendor software upgrades, reduces development time
- New systems can be built quickly by mapping new services onto the functionality provided by the layer beneath
- Middleware allows programmers
 - to develop parallel application
 - execute parallel applications on clusters
 - employ shared memory based programming
 - achieve good execution performance
 - take advantage of portability

Middleware - Disadvantages

- Middleware
 - does not offer complete transparency
 - reduces potential execution performance (services are duplicated)
 - forces programmers to be involved in many time consuming and error prone activities that are of the operating system nature
- Conclusion: to provide SSI (parallelism management, transparency, make programming and use of a system easy) there is a need for services at the operating system level

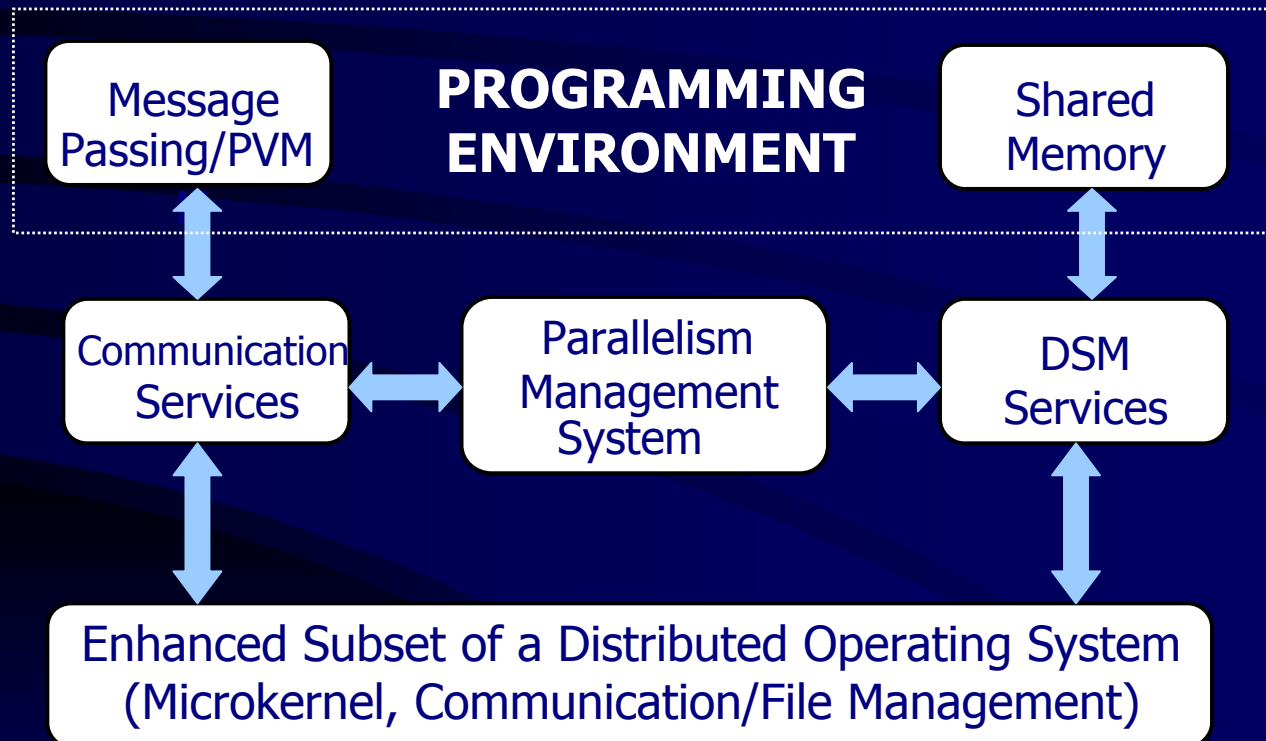
SSI via Kernel Path!

- Design and develop new SSI cluster operating systems supporting parallel processing
- Build SSI at kernel level
 - True Cluster OS
 - Good, but can't leverage of OS improvements by vendor
- Services of cluster operating systems:
 - Distributed services for transparent communication and management of basic system resources
 - Services for parallelism management and transparency
 - Services for high availability and fault tolerance
 - Services for enhanced programming environment

Cluster Operating Systems

- Cluster is a special kind of a distributed system
- Cluster operating system supporting parallel processing should
 - possess the features of a distributed operating system to deal with distributed resources and their management and hide distribution
 - exploit additional services to manage parallelism for application and offer complete transparency
 - provide an enhanced programming environment
- Three logical levels of a cluster operating system
 - Basic distributed operating system
 - Parallelism management and transparency system
 - Programming environment

Logical Architecture of a Cluster Operating System



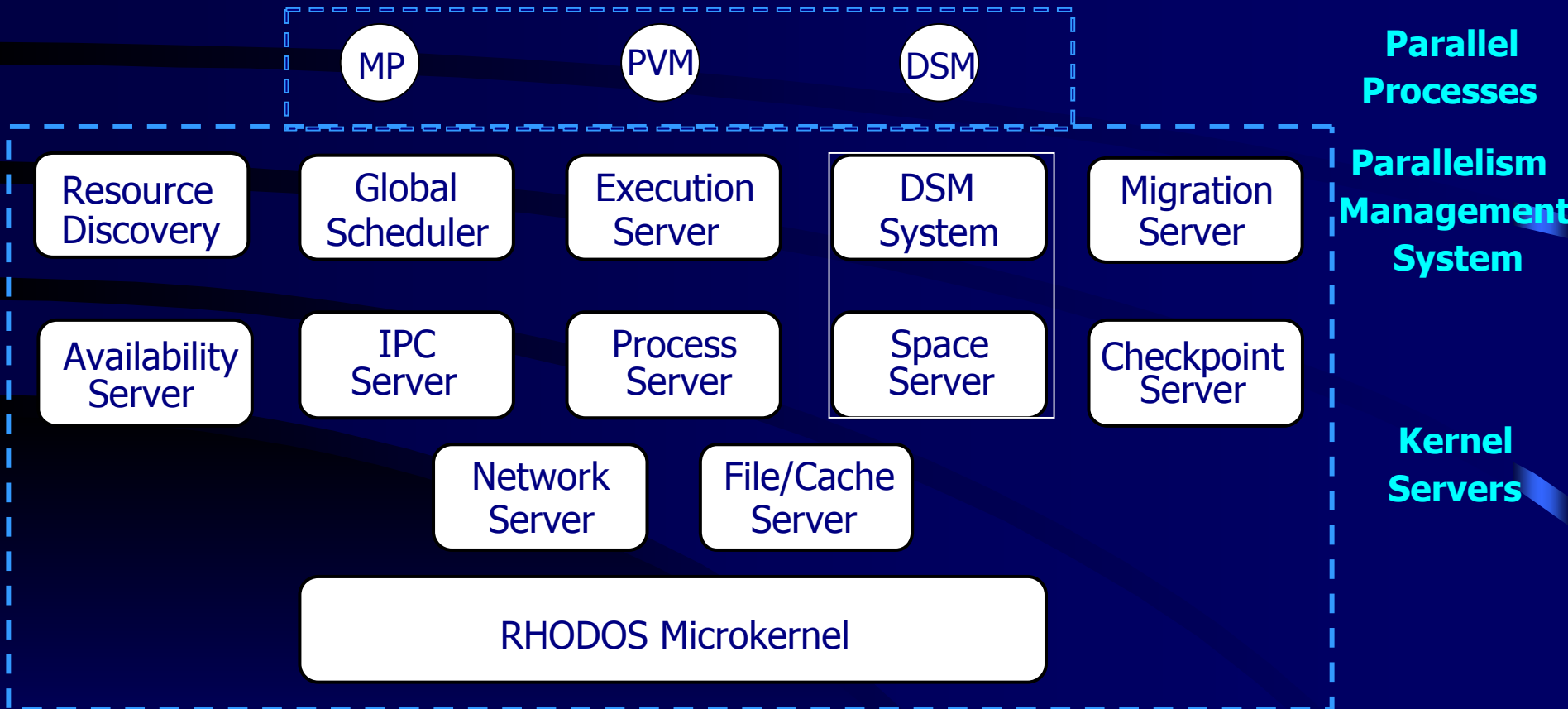
Genesis Provides a SSI

GENESIS

Cluster Operating System

- “Proof of concept”
- Client-server model and microkernel approach
- All basic resources: processor, main memory, network, IPC, files are managed by relevant servers
- IPC - Message passing services
 - Basic communication paradigm
 - Cornerstone of the architecture
 - Provided by IPC Manager and local IPC component of microkernel
- IPC placement and relationship with other services designed to achieve high performance and transparency
- DSM provided by Space (memory) and IPC Managers

The GENESIS Architecture



What is Offered by the Best Known Systems?

SSI Systems and Tools

- OS level SSI:
 - SCO NSC UnixWare
 - Solaris-MC
 - MOSIX
 - Kerrighed
 - GENESIS
- Middleware level SSI:
 - PVM, TreadMarks (DSM), Glunix, Condor, Codine, Nimrod
- Application level SSI
 - PARMON, Parallel Oracle

SSI Systems and Tools

- All systems but MOSIX, Kerrighed, GENESIS are based on middleware
- The solutions are performance driven – little work has been done on making them application developer friendly by providing SSI
- MOSIX, Kerrighed, GENESIS are offering many services that form a SSI

Conclusion

- SSI is a desirable feature
 - Can improve performance, reliability and availability
 - Offers transparency
 - Relieve programmers from activities that are of the operating system nature
- SSI could be provided by hardware, an OS kernel and/or at the application level
- SSI can / will move parallel computing on clusters to the computing mainstream
- The very next stage in the development of cluster operating system is building services supporting Autonomic Computing