UNICOS/mk: A Distributed Operating System with Single System Image

Gabriel Broner, Cray Research, Inc., 655F Lone Oak Dr., Eagan, MN 55121

ABSTRACT: *UNICOS/mk is a modular distributed operating system intended to support future Cray distributed architectures, including MPP systems, clusters, and hybrid systems. A feature of UNICOS/mk is that it offers both users and applications the "view" of a single system (Single System Image or SSI). At the same time, the complexity of dealing with a parallel architecture is hidden inside the operating system.*

1 Introduction

This paper offers a brief high level overview of the architecture of the UNICOS/mk operating system, emphasizing one aspect of it: its Single System Image. Single System Image (or SSI) is discussed in three "planes": the user view, the application view, and the system view.

The rest of this paper continues as follows: Section 2 contains the overview of the UNICOS/mk system, Section 3 discusses Single System Image and Section 4 offers the conclusions.

2 The UNICOS/mk Operating System

UNICOS/mk is the new operating system developed at Cray Research. It is a modular distributed operating system, designed to support the new distributed Cray architectures, including future Massively Parallel machines and clusters of machines.

The UNICOS/mk system is compatible with the UNICOS system [UNIC95], so users and applications need not be aware of the system's internal changes.

Internally, different from the "monolithic" UNICOS system, the UNICOS/mk system has been structured as a number of smaller components or "servers". Each server implements a well-defined portion of the operating system, which allows for simpler development and maintenance, and permits to support distribution in a more natural way. The structure of the UNICOS/mk system can be viewed in Figure 1.

The *microkernel* permits abstracting machine dependencies. It implements memory management, cpu scheduling and inter-process communication (IPC).

The *Process Manager (PM)* implements the UNICOS interface. It traps all system calls, it services (by itself) the process-related ones, and it forwards I/O-related and other system calls to the appropriate server.

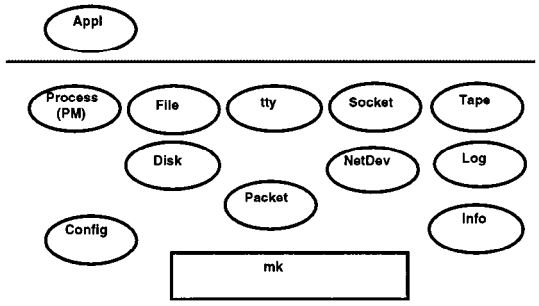

Figure 1: UNICOS/mk Architecture.

The microkernel and the Process Manager have been originally adopted from the CHORUS MiX system [CHOR92]. The Process Manager has been extensively modified to offer the UNICOS interface (as opposed to SVR4) and to support the new large-scale distributed Cray architectures (MPPs).

Most other servers have been adopted from the UNICOS system. For example the File Server contains the UNICOS file system code. Additional code, which we call the "wrapper", turns the file system code into a self-contained server.

In a similar way, the Disk Server contains the UNICOS disk drivers; the Packet Server, the UNICOS IOS/packet communication; the Socket Server, the UNICOS TCP/IP code; and the Network Device Server, the networking drivers.

To illustrate how the servers "work together", let's cover a few UNICOS/mk system calls:

- `getpid`, `fork`, `wait` and `exit` are handled by the Process Manager (See Figure 2).

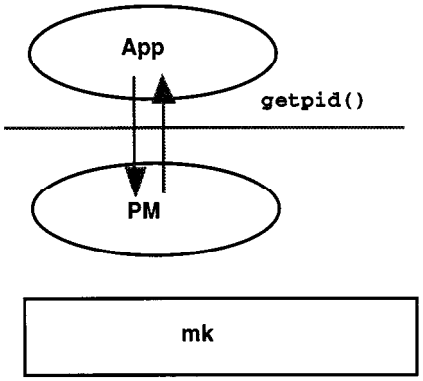

Figure 2: Example System Call: `getpid`

- `sbreak` is handled by the Process Manager in conjunction with the microkernel (See Figure 3).

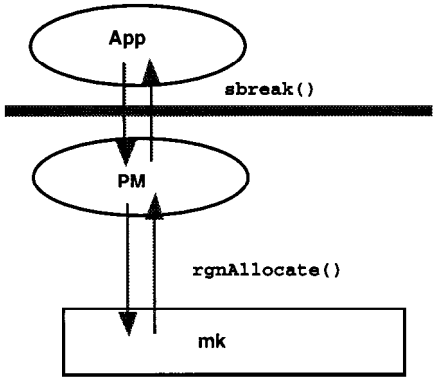

Figure 3: Example System Call: `sbreak`

- A disk `read` is received by the Process Manager, sent to the File Server, from there to the Disk Server, "down" to the Packet Server and "out" to the disk in the I/O Subsystem (See Figure 4).
- A network `send` goes from the Process Manager to the Socket Server, the Network Device Server, the Packet Server, and out to the network via the IOS (See Figure 5).

The UNICOS/mk operating system will run on a variety of Cray platforms. Figures 6, 7, 8 and 9 show how the system is laid on a single node system, an MPP, a cluster, and a "hybrid" system.

3 Single System Image

Single System Image, or SSI, refers to the ability of a Distributed Operating System to appear to users and applications as a single-node operating system. The UNICOS/mk operating system offers the image of a single system to users and applications, hiding the complexity of the distributed architecture within the operating system.

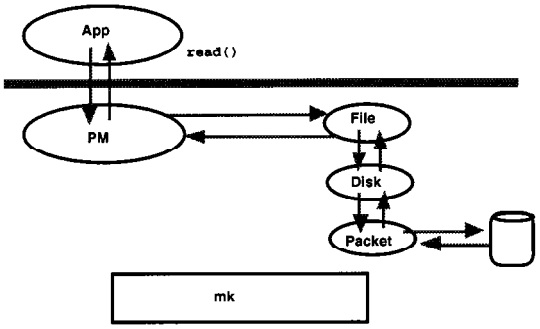

Figure 4: Example system Call: `read`

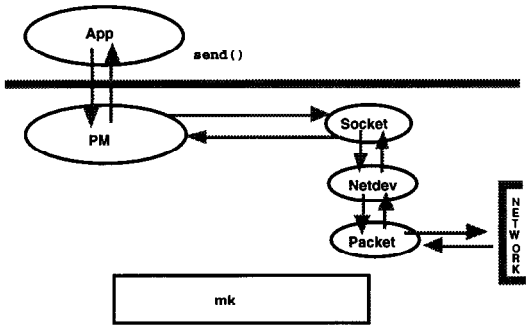

Figure 5: Example system Call: `send`

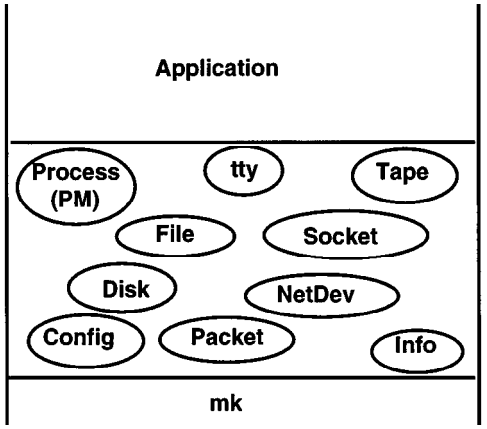

Figure 6: UNICOS/mk on a Single Node System

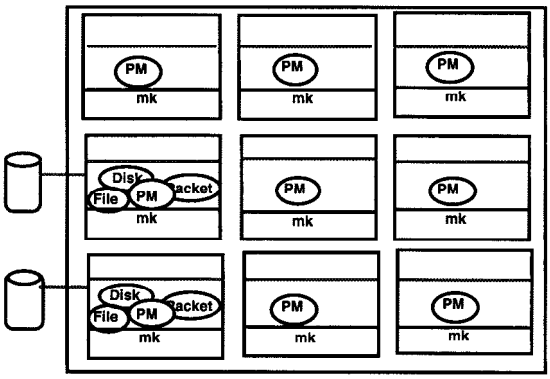

Figure 7: UNICOS/mk on an MPP System

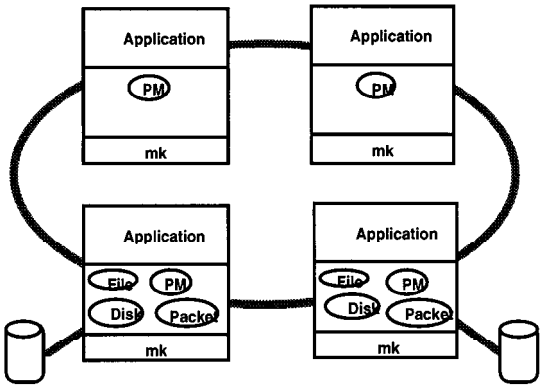

Figure 8: UNICOS/mk on a Cluster

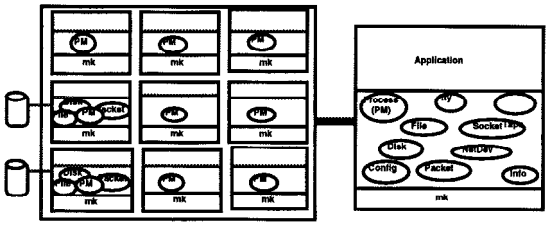

Figure 9: UNICOS/mk on a Hybrid System

3.1 Single System Image: The user view

From a user perspective, a distributed hardware platform running UNICOS/mk appears no different than a single-node system running UNICOS. Users login to the "system", obtain a shell and start applications. In reality, these applications can be running "transparently" on different nodes of an MPP system. (See Figure 10).

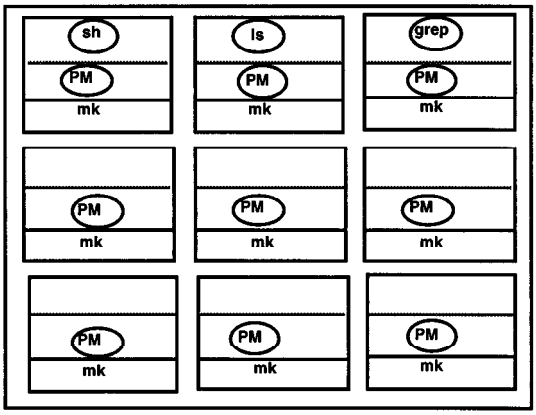

Figure 10: Application Running on UNICOS/mk

For example if a user types:

```
ls | grep x
```

`ls` could run on one node and `grep` on another one. The output will come to the screen. If the user now types CTL-C, both processes will be "killed", and if the user types CTL-Z, both processes will be put in background. Again, from a user perspective, this is no different than a single-node system.

3.2 Single System Image: The application view

From an application point of view, the UNICOS/mk system appears no different than the UNICOS system. Applications running on any of the nodes of the distributed system have the same view of processes, files, pipes, sockets, etc.

For example, a process can create (`fork`) a child, communicate with it using pipes, access the same files, and send signals to it, just like if they were on the same node. In actuality the two processes can be running on different nodes.

Even applications that were not intended to work on a distributed environment can benefit from Single System Image. For example 'nmake' is the UNICOS utility used to build applications. Typically, nmake starts a number of compilations in parallel to build a large application. Nmake on an MPP system running UNICOS/mk can potentially start each compilation on a separate MPP node, taking immediate advantage of the MPP parallelism. (See Figure 11).

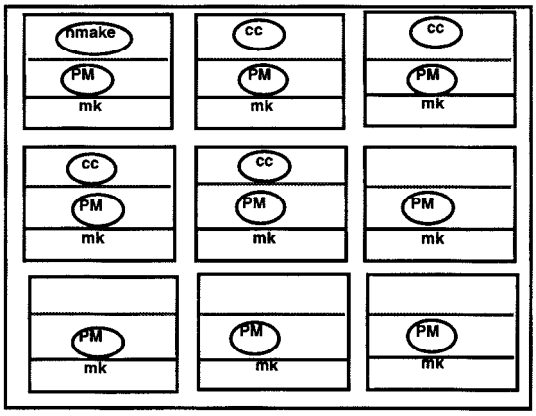

Figure 11: Nmake on UNICOS/mk

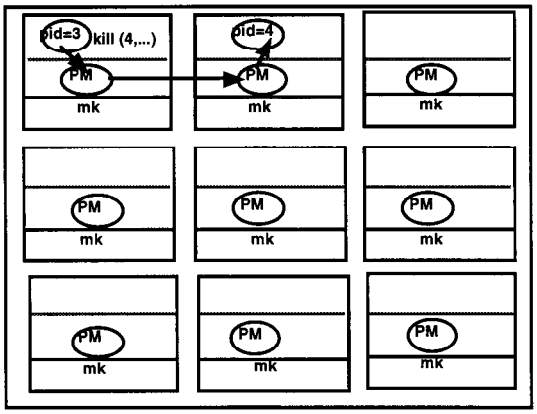

Figure 12: Single System Image Implementation of `kill`

3.3 Single System Image: The system view

The purpose of Single System Image is to make things simpler for users and applications. It is inside the Operating System where the complexity of providing the illusion of running on a single-node system resides.

Single System Image is created by the Process Managers. From a microkernel perspective, each node is independent of each other. The Process Managers, on the other hand, cooperate to create the "illusion" of a single system.

For example, an `exec` system call can be turned into a "remote exec" by the Process Manager starting a new process on a different node. Later, a `kill` system call to send a signal to the new process will be turned into a "remote kill" by the Process Managers. (See Figure 12).

Besides process-related functions, the UNICOS/mk system offers a unified view of files, pipes, devices and sockets. The way this is implemented is that all Process Managers talk to the same root File Server, which implements the "/" file system and pipes. The root File Server "points" to the same subsequent file and device servers, providing a common file and device view. In a similar way, a single Socket Server provides a common socket view. Any future server distribution (like parallelizing the socket server) will be done while preserving the image of a single system.

4 Conclusions

The UNICOS/mk Operating System will support the new parallel Cray architectures including MPPs, clusters and hybrid systems.

It is a long-term direction for Cray to provide operating systems with Single System Image. Single System Image permits users and applications to take advantage of the new parallelism without new user procedures or application changes, as the increased complexity is managed by the operating system.

References

[CHOR92] Rozier M. et al, *Overview of the CHORUS Distributed Operating System*, USENIX workshop on Microkernels and Other Kernel Architectures, April 1992.

[UNIC95] *UNICOS*, Cray Research Publication MCPF-2580394, 1995.