UNICOS/mk: A Scalable Distributed Operating System

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

ABSTRACT: *UNICOS/mk is a distributed version of the UNICOS operating system, developed to support future parallel Cray architectures, starting with the Cray T3E. A distinguishing characteristic of UNICOS/mk is its scalability; UNICOS/mk effectively supports large parallel machines, while it provides the image of a single system. This paper gives a high-level overview of UNICOS/mk and explains in more depth how operating system scalabilty is accomplished.*

1 Introduction

UNICOS/mk is a *distributed* version of the UNICOS [2] operating system. 'Distributed' means that it is capable of supporting parallel hardware architectures with a single operating system.

The operating system is *scalable*, which means it is capable of increasing its "capacity" as the system size grows.

The ability of the system to present itself as a single coherent operating system, when running on a parallel platform is called *Single System Image (SSI)* (For a more thorough discussion on Single System Image in UNICOS/mk, see [3]).

UNICOS/mk offers, at the same time, scalability and single system image. The result is a system which is able to support large parallel machines, while doing it in a usable manner.

This paper continues as follows: Section 2 contains the design goals behind UNICOS/mk; Section 3 gives a high-level overview of the overall system architecture; Section 4 explains how scalability is achieved, and Section 5 presents the conclusions.

2 Design Goals

The following are the design goals behind UNICOS/mk. These were the reasons to start a new operating system project, and the main guiding principles behind its design.

- **Support new Cray architectures**: support architectures such as the Cray T3E and future Cray machines. This implies supporting parallel architectures with a large number of CPUs.
- **Common Operating System**: the same operating system will support the various Cray architectures.
- **UNICOS compatible**: for compatibility with existing applications, the system will present the familiar UNICOS interface.
- **Modularity**: to improve its reliability and maintainability the operating system will be more modular than traditional "monolithic" operating systems.
- **Single System Image (SSI)**: The multi-node operating system will appear to the user just like one coherent single-node operating system.

Scalability: The operating system's ability to handle requests will grow as the system size grows.

3 The Architecture of UNICOS/mk

Different from traditional "monolithic" operating systems, the UNICOS/mk system is comprised of a *microkernel* and a number of *servers*, which cooperate to provide the complete operating system functionality. The UNICOS/mk operating system has been derived from the Chorus [1] and UNICOS [2] systems. The Chorus operating system has provided the overall serverization scheme, the microkernel (mk), and the basis for the UNICOS/mk Process Manager (PM). Most other servers have been developed from code taken from the UNICOS operating system. Figure 1 shows the overall architecture.

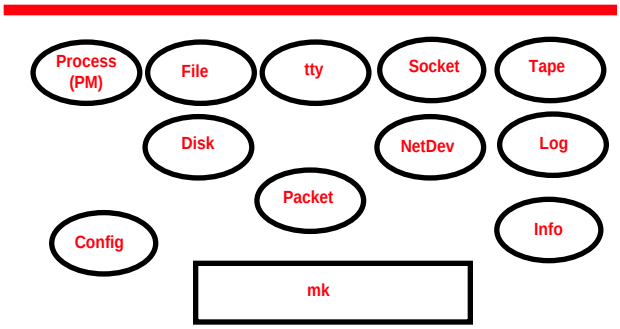

Figure 1: The architecture of UNICOS/mk

- The *microkernel* implements the machine dependent components, memory management, CPU scheduling and inter-process communication (IPC). The microkernel offers

a series of abstractions (actors, threads, ports, regions), which permit implementing the rest of the operating system in a machine-independent fashion.

- The *Process Manager* provides the UNICOS interface. It receives all system calls; it services by itself the process related ones, while it forwards to the appropriate server other (e.g. I/O related) system calls.
- Other servers (in general) contain code directly taken from UNICOS. The UNICOS code is converted into a UNICOS/mk self-contained server by compiling it and linking it with a "wrapper" library. For example, the *File Server* contains the UNICOS file system code, the *Disk Server* contains the disk drivers, the *Socket Server* contains the TCP/IP code, the *Network Device Server* contains the network drivers, etc.

To illustrate how the various servers cooperate to provide the complete operating system functionality, Figure 2 shows `getpid`, as a process-related system call handled completely by the Process Manager, and Figure 3 shows `open` and `read`, as examples of system calls involving multiple servers.

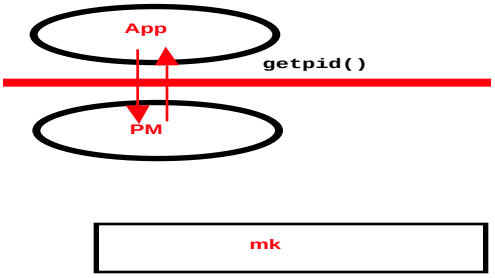

Figure 2: A process-related system call

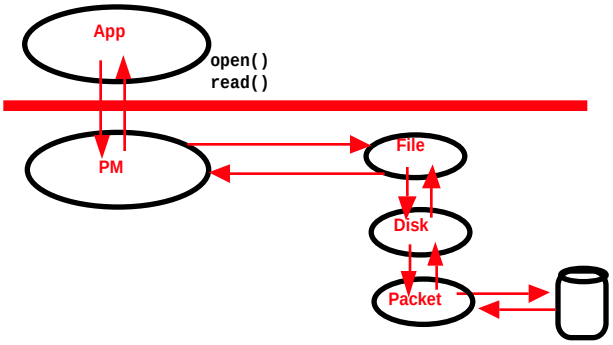

Figure 3: System calls involving multiple servers

UNICOS/mk runs on different hardware platforms. On single CPU or SMP machines, all of the UNICOS/mk components run in the same address space (see Figure 4). When servers are in the same address space, the microkernel is able to optimize the inter-server communication, to achieve performance comparable to that of an integrated kernel.

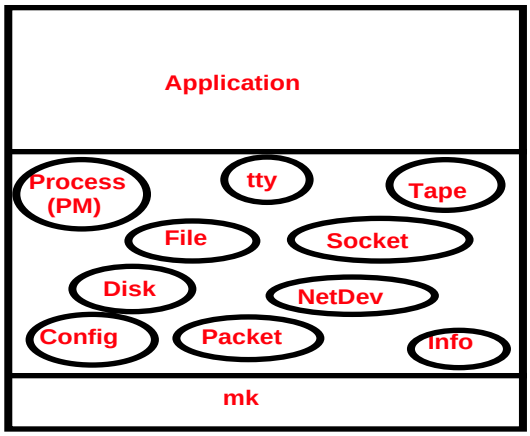

Figure 4: UNICOS/mk on an SMP

When UNICOS/mk runs on an MPP machine, such as the Cray T3E, each node runs only a portion of the operating system servers. Nodes that run applications only need a microkernel and a Process Manager. Other nodes (usually described as operating system nodes from a configuration point of view) run other servers such as the File or Disk servers. Figure 5 shows UNICOS/mk on an MPP system.

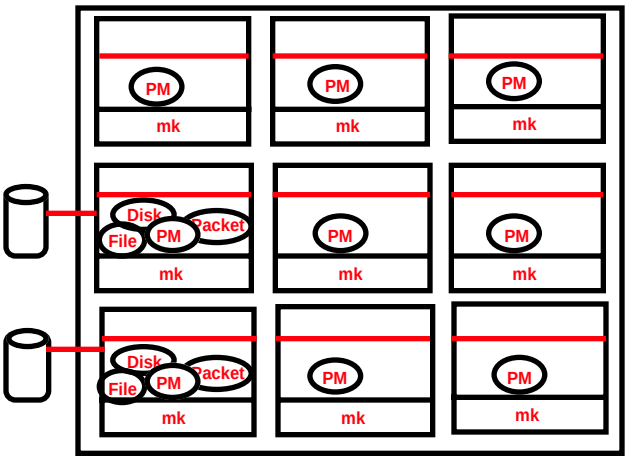

Figure 5: UNICOS/mk on an MPP

UNICOS/mk offers *Single System Image (SSI)*. This means from a user or application developer point of view, the system presents itself as a coherent single system, even when the underlying hardware architecture may have thousands of independent nodes. Users can 'login' to the "system", run applications on it, and never need to be aware of the fact that applications actually run on different nodes of an MPP machine. For a more extensive discussion of Single System Image in UNICOS/mk, the reader is encouraged to read reference [3].

4 Scalability in UNICOS/mk

An important property of the UNICOS/mk operating system is its ability to handle more requests as the system size grows. This ability to grow, or scale, is called *scalability*.

In the UNICOS/mk system, scalability is achieved in a number of ways. *Process-related* functions scale because they are handled by the Process Manager and the microkernel. The larger the machine, the more Process Managers and microkernels it will have. For example, a machine with 16 processing elements (PEs) has 16 Process Managers and 16 microkernels. At the same time, a machine with 2,048 PEs has 2,048 Process Managers and 2,048 microkernels, which are able to handle a proportionally larger load.

For *File-I/O-related* functions, scalability is achieved in a different manner. We have developed a series of solutions which permit scaling the file I/O traffic in different ways. The first solution, shown in Figure 6, is *Remote Mount*: different file servers may service different file systems. The result is files using different file names are accessed via different file servers. This solution permits, for example, separating the traffic to the `/bin` directory, containing applications, from that of the `/tmp` directory, containing temporary files. Programmers can take advantage of this parallelism, but they have to (explicitly) use different files, in different directories.

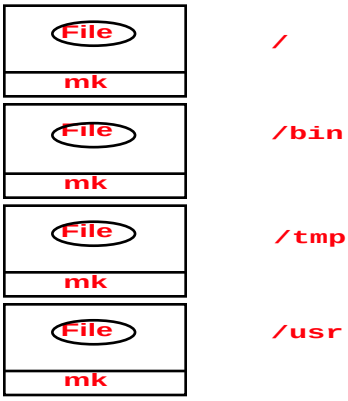

Figure 6: Remote Mount

The second solution for file I/O scalability is the *File Server Assistant*. The File Server Assistant (FSA) is a new server that runs on every (application) node (see Figure 7). The File Server Assistant can be viewed as a "shadow" of the central File Server: it basically contains the same code as the central file server, and it is capable of processing a file server request without central file server intervention. Typically `open` requests go to the central file server, but further `read` and `write` requests can be handled locally by the assistant. Current restrictions are that I/O has to be unbuffered (raw) and well formed (aligned).

The File Server Assistant is a more interesting solution when considered jointly with *Disk Server Parallelization*. A file server can "stripe" a file system across multiple (up to 64) disk

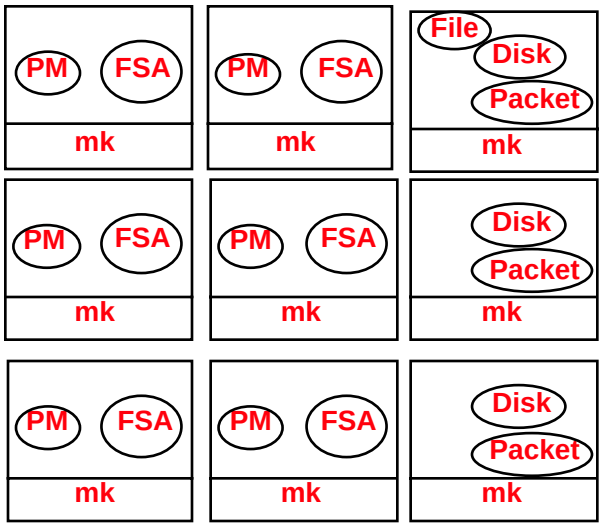

Figure 7: The File Server Assistant

servers (see figure 8). When the File Server Assistant is used (and the central File Server is bypassed) multiple paths exist from each application (or each PE of a multi-PE application) to the same file system, or even the same file. For example, in a 64-PE application every PE could be performing I/O to the same file (different offsets), while each PE takes a different path using a different set of servers. Different paths to disk, without a central point of control, make this solution scalable.

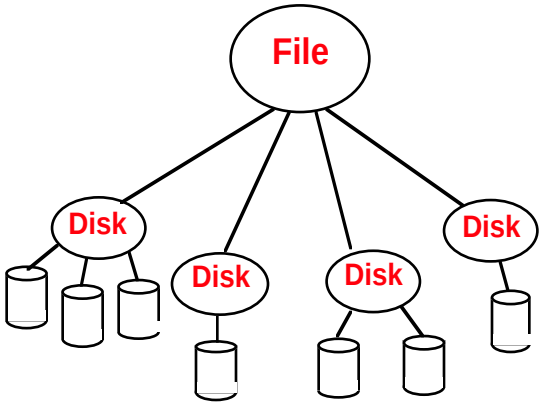

Figure 8: Disk Server Parallelization

A third solution for I/O scalability is *Distio*. Distio is a variation of the `listio` system call. Listio permits specifying multiple read/write operations on a single system call. The Distio variation, permits indicating a destination PE for each I/O request. This permits issuing one system call resulting in multiple actual I/O transfers destined to different PEs (see Figure 9).

All the solutions presented here for I/O scalability can be used together, but at the same time address different problems. Remote Mount addresses the contention due to many programs opening files using the same file server. A programmer can take

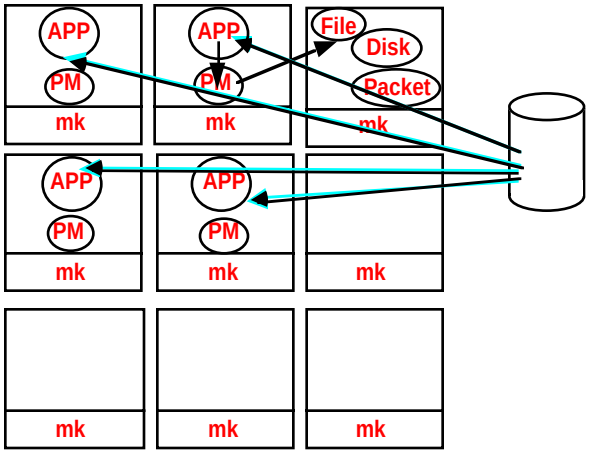

Figure 9: Distio

advantage of this parallelism in an explicit fashion. The File Server Assistant addresses the more common contention when all the elements of a multi-PE application access a single file at different offsets. Finally, Distio provides a scalable way for a single PE to drive all of the I/O for a multi-PE application.

5 Conclusions

UNICOS/mk is a *distributed* operating system as it supports parallel hardware. It is *scalable*, as its ability to handle requests can grow as the system size grows. UNICOS/mk offers *Single System Image (SSI)* when running on a parallel platform, making it more usable for software developers, end users, and system administrators. This paper emphasizes the scalability aspects of UNICOS/mk, but the strength of UNICOS/mk comes from its ability to be a scalable operating system while maintaining the image of a single system.

6 References

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

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

[3] Broner G., *UNICOS/mk: A Distributed Operating System with Single System Image*,CUG 1995 Spring Proceedings.

7 Further Contact

Gabriel Broner can be reached via e-mail at broner@cray.com