Cellular Irix Plans and Status

Gabriel Broner
Silicon Graphics
655F Lone Oak Dr, Eagan, MN 55121, USA
E-mail: broner@cray.com

CUG Stuttgart, June 1998

Abstract

Cellular Irix is an evolution of Irix that incorporates features available in UNICOS and UNICOS/mk, including increased scalability and fault containment. Cellular Irix is a long-term vision, while at the same time pieces of its technology are being incorporated into Irix to solve today's real-world problems. This paper describes the architecture of Cellular Irix, the near-term Irix deliverables that incorporate its technology, and the status of the project.

Contents

- Introduction: A Single Operating System
- Background: Operating System Requirements
- The Architecture of Cellular Irix
- Cellular Irix Deliverables
- Project Status
- Conclusions

---

1. Introduction: A Single Operating System

After the merger between Cray Research and Silicon Graphics, the two organizations analyzed their needs in operating systems. The groups jointly decided to proceed with a common path. A single operating system for the broader product line leverages the expertise existing on both sides to develop a better product. The joint system, Cellular Irix, offers features from Irix, UNICOS and UNICOS/mk. The system is binary compatible with Irix, provides access to a large set of applications, and is source compatible with applications coming from the UNICOS and UNICOS/mk operating systems that comply with the published Supercomputing API.

The decision to have a single operating system also has an impact in other areas; we will be able to reuse libraries, commands, compilers, etc. across the wider product line. At the same time, we expect a single operating system will simplify things for customers, who will only need to deal with and administer one system, and for ISVs, who would be able to target a wider range of machines with a single application port.

The rest of this paper covers the following: Section 2 talks about requirements for the operating system. Section 3 presents the Cellular Irix architecture. Section 4 describes the various Cellular Irix deliverables. Section 5 covers the status of the project. Section 6 offers the conclusions.

2. Background: Operating System Requirements

An operating system that services the "servers to supercomputers" product line has to address requirements coming from the various market segments it targets.

Requirements from the High End

An operating system for the high end needs to be able to support large machines (typically from 64 to 4,000 CPUs) with good performance. The operating system has to properly scale for technical computing workloads, which typically are large CPU-intensive applications which also perform a large number of big-block I/O requests.

From a usability and ease of administration perspective, a large system (e.g. 1,000 CPUs) needs to present itself as a single coherent system (Single System Image, or SSI) and not as 1,000 independent machines.

On a large system, hardware failures are more likely to occur (for statistical reasons), so it is important for the operating system to be able to tolerate failures, such that any failure does not bring the system down.

There is also a series of features that have unique requirements for the high end. These include Checkpoint/Restart, Limits, Accounting, Scheduling, and Security among others.

Requirements from the Server World

The server world requires an operating system that will support "medium size" machines (from 4 to 64 CPUs.) Applications are typically smaller than those in the supercomputing world, but the demands they impose on the system may be higher, due to a more varied use of system services. Appropriately scaling these general purpose workloads is a challenging requirement for a system in the server world.

From a fault containment point of view, the requirements are also different. In the server world, a system being down may represent money being lost. For example, if a system servicing credit card transactions happens to be unavailable, customers may just switch to use a different credit card. In this example, money lost due to a server being down may never be recovered, and as a result, servers being up and available at all times is clearly a goal.

3. The Architecture of Cellular Irix

Cellular Irix is an operating systems architecture that provides increased *scalability* and *fault containment*.

A *cell* is an Operating System abstraction that represents a "piece" of the machine (a series of CPUs, their associated memories, etc.) Within a cell, the operating system looks pretty much like an SMP operating system (like Irix, or UNICOS). Across cells, the distribution of subsystems offers single system image (like in UNICOS/mk). (Figure 1 shows a multi-cell system.)

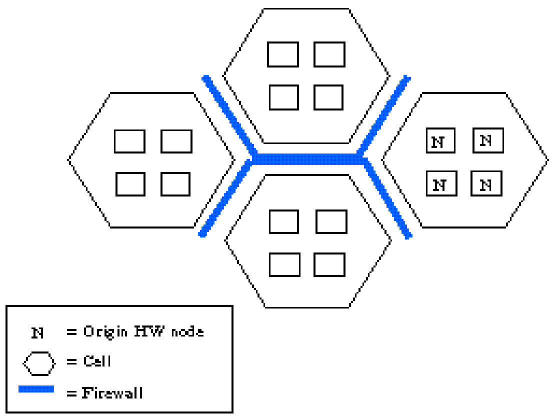

Figure 1 - A multi-cell Cellular Irix system

Cells can be of different sizes, and a particular machine can be configured in more than one way. For example a 128-CPU machine could be configured as 4 cells of 32 CPUs, or 8 cells of 16 CPUs, etc. Cell sizes and the number of cells have implications on fault containment and scalability. From a fault containment point of view, a failure in a cell brings down the complete cell. If cells are large, more of the machine will be lost in a failure. From a scalability point of view, within a cell, the system will scale like an SMP, and across cells, like a distributed OS. If cells are large, lock contention and memory latency within a cell will be high, while there will be less inter-cell traffic due to distributed protocols. On the other hand, if cells are small, SMP lock contention and memory latency will be small, but the increased number of cells will imply higher inter-cell traffic. In general we expect a "medium ground" solution will satisfy the needs of each individual site in terms of fault containment and scalability. Just to give an idea of our current thinking, we imagine a 1,024-CPU machine could be 32 cells of size 32, and a 4,096-CPU machine could be 64 cells of 64 CPUs.

From a performance point of view Cellular Irix behaves similarly to Irix when a system call is serviced locally within a cell. To achieve this level of performance, a series of techniques is being used to keep most operations local to a cell. Examples of these techniques are local caching and a peer-to-peer style of distribution.

4. Cellular Irix Deliverables

Our long-term vision is a multi-cell system where all subsystems are fully distributed, the system is fully single system image, and it tolerates failures on any cell. At the same time, utilizing Cellular Irix technology, we will provide near-term deliverables that address the immediate needs of the various market segments.

Figure 2 shows the deliverables that utilize Cellular Irix technology. In 1998 Irix 6.5 is available as a *single system* to address the needs of the technical compute market, or as a *partitioned system* to address the needs of the server world. In 1999 the *Enterprise* and *Supercomputer* variants of the system represent evolutions with additional capabilities. Future releases will continue to enhance the systems for both markets.

```
        1998                  1999              2000+

    Irix 6.5                Irix 6.X
      Partitioned System      Enterprise      Fut. Cellular Irix
      Single System           Supercomputer

  ----------------------------------------------------------------->
```

Figure 2 - Cellular Irix Deliverables

1998 - Irix 6.5

Irix 6.5 is offered in two flavors: *Single System* and *Partitioned System* .

Single System

The *Single System* variant addresses the needs of the technical computing market, where the main objective is to run large applications across the system. Irix 6.5 single system scales up to 128 CPUs in an SMP manner providing single system image. The system scales well for large applications that perform compute and I/O in large blocks. At the same time, this system does not offer fault containment: hardware or software failures bring the complete system down.

Partitioned System

The *Partitioned System* variant addresses the needs of the server/high availability world. In an Irix 6.5 partitioned system each cell (or partition) is effectively an independent host with its own console, root disk, host name, etc. The partitioned system does not offer single system image. The key feature of this variant is that it offers full fault containment for each cell. If a failure occurs in a given cell, only that cell will be affected while other cells continue to operate. This variant of Irix is intended to run server workloads where full fault containment is the overriding requirement, and single system image is secondary. The Irix partitioned system supports the array/FailSafe software, which offers redundant services for database servers, web servers, and file servers.

As an example, a partitioned Irix 6.5 system would offer a good solution for a credit card transaction database server: a database server would run on one cell. If that cell fails, a second database server would be started on the second cell. Using multi-ported disks, the second cell will have immediate access to the database files. By using IP aliasing, the second database server quickly responds to the internet address of the downed database server. The array/FailSafe software provides the support mentioned here.

1999 - Irix 6.X

Single System *(a.k.a. Supercomputer, HPC, SSI)*

This version utilizes cells to scale up to thousands of cpus to address the needs of the high-end technical market. It offers full *single system image*, which means that for users, applications, and administrators, a large machine running as a single system will look like a single host. To provide single system image, *all subsystems are distributed*. However, not all subsystems are distributed in the same manner:

- The *file system* and *volume (disk) manager* are fully distributed in a peer-to-peer fashion, such that file I/O issued on a given cell be processed on that cell. The new distributed file system is called CXFS (for Cellular XFS, and the new volume manager is call XVM, for eXtended Volume Manager).
- Other components are distributed in a more "basic" fashion. For example networks reside on a special cell, that we call the *golden* cell. Networking requests are forwarded from the originating cell to the golden cell for processing.

From a *fault containment* point of view, a failure on the golden cell is fatal and it brings the system down. A failure in any other cell can be contained to that cell. Note that golden cell failures will be infrequent, compared to failures on any cell. Note also that application recoverability is achieved via Checkpoint/Restart: an application can be checkpointed periodically, and in the event of a failure on a given cell, it can be restarted on a different set of cells.

From an *administration* point of view, this solution allows administering a large machine like a single entity, and not like a large collection of machines.

Note that this solution addresses well the needs of the HPC/Supercomputer world, which is running large technical computing applications that perform lots of computation and I/O in parallel. This solution is not intended to provide scalable services to interactive users or generic server applications.

This system is intended to run parallel applications well across the whole machine (across cells). Parallel jobs using the MPI and shmem programming models will be supported, and they will offer performance similar to MPI and shmem running on an SMP. MPI and shmem utilize shared memory, which is transparently distributed across cells; as a result these applications can exchange messages and data without the operating system being involved. This offers much better performance than cluster solutions. In addition, the MPI and shmem models could be used together. An existing MPI application will directly port to a machine running Cellular Irix, and it will run better than on a cluster due to the MPI shared memory implementation. In addition, a few key routines of that application could be rewritten using shmem, such that the overall performance of the application can be improved without a total rewrite. This is somewhat similar to writing key routines in assembly language in a large C application, to improve its performance without a major rewrite.

As it was mentioned earlier shared memory (system V, *mmap*'ed files, and /dev/zero) is distributed, so multiple cooperating processes can share memory across cells. Individual processes (like a C - Pthreads application) will be confined to use the CPUs of a cell. They can, on the other hand "borrow" memory from other cells, so a single process can effectively use all the memory in the machine.

Partitioned System *(a.k.a. Enterprise, HA, Cluster)*

This version is a direct evolution of the partitioned system described for Irix 6.5. It addresses the needs of the server/high availability market, where the emphasis is on fault containment. The partitioned system looks to its users like a multi-host system. Each cell has a unique host name, network address, root file system, I/O devices, console, etc. From a user or application point of view a cell looks like an independent machine.

The new addition for this release is the distribution of selected subsystems (the file system and volume manager) to provide single system image for those subsystems. The main difference between this release and the previous version of the partitioned system is then that the system will offer a single view of files from all cells (hosts), and applications from multiple cells will be able to access the same file systems (or the same files) simultaneously with local file system levels of performance.

Just as in the previous partitioned systems, the system continues to offer full fault containment for failures occurring on any cell.

2000+

The plan for "after 2000" is to continue to evolve both lines, the single system and the partitioned system. In the single system side of the world we are considering better (peer-to-peer) distribution for additional subsystems, reducing the dependency on the golden cell, and we are analyzing the work involved in multithreaded processes spanning cell boundaries. In the partitioned system, we plan to continue to add high availability features, and also continue to perform distribution of subsystems to provide single system image for additional subsystems.

5. Project Status

The following describes the accomplished and planned milestones for the 1999 release of Irix supporting the Supercomputer world.

Accomplished Milestones

First-multi cell system	Apr 97
(Cells, messaging, CFS)	
Multi-user system	Jul 97
(Remote exec, dist mmap, AIM7/9, VSX)	
Remote fork, proc migration, sockets	Sep 97
Irix 6.5 feature work	Oct 97 - Feb 98
MPP apps on 128p multi-cell	Mar 98
CXFS direct I/O	Apr 98
Cellular Irix open machine time	May 98

Planned Milestones

CXFS buffered I/O	Jul 98
XVM	Aug 98
256p Internal Milestone	Sep 98
Development complete	Mar 99
Alpha	Jun 99
Beta	Aug 99
Release	Oct 99

6. Conclusions

The Cellular Irix project has developed technology for increased scalability and fault containment. Cellular Irix offers us a long term driving vision and at the same time a series of near-term deliverables that address the near term needs of our markets. The project has made great progress and it has met its milestones.