Future File System: An Evaluation

Brian Gaffey and *Daniel J. Messer*, Cray Research, Inc., Eagan, Minnesota, USA

ABSTRACT: *Cray Research's file system, NC1, is based on an early System V technology. Cray has made a significant investment in NC1's performance and reliability. Recent improvements in file system technologies and new customer requirements drove us to undertake an evaluation of available file systems. This paper presents the results of our evaluation.*

1 Introduction

With advances in storage technology and access to large memory, Cray supercomputers can now solve data intensive problems that were not conceivable a few years ago. On the storage technology side, the amount of disk space is increasing rapidly because of inexpensive technology such as SCSI and RAID disks. High capacity disks are widely available and reliability has increased. It is not uncommon to see large file systems of tens or hundreds of Gigabytes spread across tens of disks. In terms of memory, the Operating system must be able to perform fast I/Os to transfer large amounts of data between storage devices and large memories.

File systems and storage management mechanisms are now held responsible to manage complex disks configurations. Modern file systems must meet six major requirements:

- performance
- easy administration and maintenance
- system availability
- resiliency
- security
- standards compliance.

The CRAY native file system NC1, combined with the Logical Device Driver (LDD) have been developed and optimized to run on CRAY's high-end products which solve data intensive or I/O intensive problems. Cray is committed to high performance and Cray's file system proves it. Cray is also committed to offer a file system solution that scales well with no impact on administrability, system availability and resiliency. Cray conducted a study to determine the best approach to offer efficient scaling: develop and implement new capabilities internally or find an existing product on the market to integrate into the Cray system.

This paper describes the process to evaluate the file systems available and how they compare to Cray NC1 file system. At this point in our evaluation, we've completed our feature and performance comparisons. Our next step is to obtain the source of the file systems and compare the implementations.

2 Requirements and Features

Our first step of the evaluation was to define the requirements for the future file system and compare all the features currently available in different products that fulfill these requirements.

2.1 Resiliency

Resiliency defines the stability and data coherency of the file system when recovering from hardware or system failure. It also defines the time to recover.

The feature which significantly improves the scaling of recovery in modern file systems is called "journaling" or "logging". Non journal-based file systems such as NC1 must be checked at boot time because unscheduled system downtime can interrupt file system updates. Checking consistency for large file systems can be time consuming. With journaling, file systems do not have to be checked because changes from unfinished system calls are discarded. The log only records updates that modify the file system structure. When the file system recovers the log is played back which considerably reduces the time for recovery. Journaling is ranked as the most important feature required in a new file system. Therefore, only file systems with logging capability have been evaluated.

2.2 Performance

Cray is committed to continue to deliver performance. File system performance is usually influenced by two factors: how the file system handles the transfer of data from user space to disk and how the device is accessed.

2.2.1 Transfer Of Data

There are several ways to transfer data between disk and user space. The most effective features are:

- Raw disk I/O,
- Buffered I/O
- Asynchronous I/O.

Raw disk I/O consists of direct data transfer between disk and the user supplied buffer for reads and writes. It bypasses the file system layer in the system. This is especially useful for database applications. Among the file systems evaluated, only Cray has implemented this feature.

Buffered I/O consists of a delayed write, in which the data is put into a buffer at the system level but not written to disk before the write returns to the caller. The user application is free to continue processing. The system however must complete the transfer to disk before processing any other task. In the same manner, data is read ahead from disk to system buffer. All file systems evaluated use buffering, also called cache, to achieve performance.

Asynchronous I/O is similar in concept to Buffered I/O. With Asynchronous I/O, however, the system is also free to do something else before the data is written on disk. It is an important distinction because there is often confusion between Buffered and Asynchronous I/O. Only Cray NC1 can claim support of Asynchronous I/O.

2.2.2 Physical Access

Extent-based allocation, space pre-allocation, explicit file system alignment, primary/secondary allocation area and contiguous space allocation are the most important methods known to improve access to a disk.

A disk is partitioned into physical sectors. File system architecture usually partitions the disk into logical blocks which contain a fixed number of sectors. In the case of **Extent-based** file systems, an extent is defined as one or more adjacent blocks of data within the file system. It means that disk I/O to and from a file can be done in units of multiple blocks instead of one block at a time.

Pre-allocation associates space with the file when it is created rather than as data is written into the file. The space cannot be allocated to other files in the file system. This prevents an unexpected out-of-space condition on the file system.

Explicit file system alignment allows one to make a file system with all the allocation units aligned to physical boundaries. It helps applications that are tuned to the specific disk they reside on, doing I/O in track sizes or cylinder sizes.

Primary/secondary allocation area splits the data blocks into two areas. One that will contain small files, the other large files.

With **Contiguous space allocation**, when a reservation is made, a specification can be included in the request that the allocation of the reservation be contiguous. Maximum contiguity of a file optimizes its I/O characteristics.

All these features are common in other file systems. Cray has implemented all the features except contiguous space allocation.This feature analysis indicates that none of the file systems evaluated can achieve Cray native file system performance.

2.3 Administrability and Maintainability

Administrability and maintainability define how easy it is to use a file system from an administration point of view. This requirement is increasingly important because, as file systems spread across a large number of disks and as the number of file systems to administer increases, the administration should remain simple and the amount of time spent for maintenance should remain constant. Several features are available in order to facilitate file system administration such as file system creation, graphical interface and on-line defragmentation.

Although all file systems follow a similar interface to create a file system, some are better suited for creation of large file systems. Depending on the architecture, the time to create a file system can be proportional to the size of the file system or not. In an environment where large file systems are required the later technology will be more appropriate.

Graphical interfaces provide an icon-based interface for representation of system resources. Combined with an **auto layout** capability, it allows the administrator to have a clear representation of the hardware and file system structures. It is useful when tens of disks are involved.

Visual management allows an administrator to manage system resources using a graphical interface instead of using text commands.

On-line defragmentation allows gaps to be removed between areas that are in use on a disk without unmounting the file system that is being cleaned. Space is allocated or freed on disk when files are created and remove. Free space is reused but it is unlikely to be totally reallocated which creates holes. Fragmentation impacts performance especially when the file system becomes full.

2.4 System Availability

System availability defines how disruptive administrative operations are to the user. On-line operations allow one to significantly reduce down time for file systems. On-line disk movement, resizing and backup are the most popular features.

On-line disk movement allows one to add, remove or replace failed disks without disrupting the system or file systems not affected by the disk failure.

On-line resizing allows one to resize a file system, grow or shrink, without unmounting the file system or interrupting users' productivity.

On-line backup provides the capability to take a snapshot image of a mounted file system without disrupting the file system. These features are implemented in some of the products evaluated.

2.5 Standards

In a heterogenous environment it is important that the future file system conforms to standards such as **POSIX** but also to standard protocols in order to offer functionality in storage management such as distributed computing and Hierarchical Storage Management (HSM).

2.5.1 Distributed Computing

In terms of distributed computing, **NFS** and **DFS** support are important. NFS and DFS support is in recognition of the general industry trend towards client/server computing within a heterogeneous environment.

2.5.2 Hierarchical Storage Management

The current level of integration between **DMF** [Cray HSM solution] and NC1 is seen to be effective, however, it is not a requirement that the HSM solution is tied so tightly to the file system if this would allow a more open solution. Using the **DMIG** interface to interact between the HSM and the file system would be a preferable approach.

2.6 Conclusion

This analysis provides a list of features that would answer the requirements of a new file system. Among the file systems evaluated, some provide most of the feature which could improve NC1 in terms of usability, system availability and resiliency. However, in terms of performance, the evaluation shows that Cray's file system is the strongest product.

3 Testing

The feature analysis points out that there are competitive products on the market which could be good candidate to integrate into Cray's system to strengthen Cray's file system solution. The following step of our evaluation was to conduct testing on these file systems in order to determine the best candidate.

Three kinds of tests have been conducted:

- Functionality,
- stress
- performance tests.

Functionality tests verify that the products evaluated do what they claim they do and provide a good feeling of their administrability and maintainability.

Stress tests determine the file system limits such as the maximum number of files. It also simulates how the file system would react in a real multi-users environment where fragmentation occurs.

Performance tests are not relevant in terms of rates; they are a good source of information to determine how the different file systems have been optimized and how they compare against the others. For example, a file system can be better optimized for a large number of small I/Os while another is better suited for a small number of large I/Os.

3.1 File Systems Tested

We evaluated only **journal-based** file system since it seems the best approach to improve the scalability of our file system.

Among many journal-based file systems available, three were chosen for this evaluation because they answer most of the requirements previously defined. Unfortunately, their names and the name of the companies that developed them can not be disclosed at this time because Cray is under negotiation with the chosen candidate. Therefore, they will be described as File Systems A, B and C.

3.2 Testing Environment

Two Solaris platforms have been used for testing because all file systems evaluated were running in this environment.

A Sun IPC workstation running Solaris 2.4 was used for functionality and stress testing. It contains 1 controller and 5 disks (200 MB, 1.3GB, 1GB, 200MB, 200MB). Two 200MB disks were fully available for the file system evaluation. 6 partitions (900MB) were available on the 1GB disk. The 7th partition was reserved.

A Cray CS6400 with 32 processors and 4Gbytes of memory running Solaris 2.4 has been used for stress and performance testing. 13GB (one partition per disk) were available, spread across 8 controllers and 13 DD5S disks (Seagate Elite-3 ST43401N) with 1 to 3 disks per controller.

3.3 Functionality Testing

Functionality testing focused mainly on administrability and resiliency. In order for the tests to be valid, the features tested were available on at least two of the candidate products.

The features tested are:

- file system creation
- on-line resizing
- journaling
- on-line backup

Because those features are processed through user commands, depend on human intervention and don't produce any valuable numbers, there is no objective way to evaluate them. However, there are criteria that facilitate the comparison such as the number of user commands necessary to implement or use the functionality, the number of administrative files to modify and the timing to execute a feature. The best implementation should be one command, no administrative files to modify and the fastest execution.

This paper only presents the **journaling** functionality test. It consisted of creating a 150MBytes file system, perform a first test with logging turned off and a second test with logging. In both cases the test consisted of mounting and populating the file system, crash and reboot the system, and time the file system check operation.

Table 1 presents the results for three file systems.

The interesting number in this test is to compare the time to check the file system using the logging feature. Logging brings a clear performance improvement in all cases. The table shows, however, that the time to recover varies among the systems tested. In this case, File System B is clearly the best candidate. Further testing would also prove that the time to recover using a logging technology is constant and is not proportional to the size of the file system.

TABLE 1.

Operation	File System A	File System B	File System C
fsck without logging(minutes)	1:00.12	0:56.38	N/A
setting up logging	0:4.25	0:1.12	N/A
number of commands	2	1	0
fsck with logging (minutes)	0:19.63	0:6.54	0:27.01
status	PASSED	PASSED	PASSED

3.4 Performance Testing

Performance tests consisted mainly of timing throughput operations and basic file systems functionality such as opening and closing a file system.

The tool used during the evaluation measured throughputs for multiple transfer sizes. For each transfer size, the program calculated the number of transfers required to write an entire file to disk, write the entire file on disk with as many write() system calls as necessary, read the entire file and rewrite the same file (preallocate).

For each of these operations, it recorded the real elapsed time for the entire transfer and the high resolution real time of each individual write() and read() system call. The program reported the environment information, write rates, write - preallocated rates, read rates and system call times.

The figures compare read, write and pre-allocated write performances for three file systems evaluated.

The **Read performances** show that File System A has excellent read performance which is probably due to a very good job of read-ahead caching. Read ahead caching delivers roughly a factor of 6 performance boost (compared with disk speed which is 1 MBytes/Second).

File System B also shows a nice curve but is about 15% slower than File System A. Although it also performs read-ahead caching, the algorithm is not as well optimized. In both cases they do a good job to reach maximum performance fast.

File System C has read performance equivalent to File System B for large transfers but starts very slow (even slower than disk speed for 512 bytes transfers).

Write performance is very good for File System B. It has a curve that one can expect. It can achieve this kind of performance using write-behind caching. Compared with the read performance on the previous graph, write performance is just slightly below read performance. It probably means that File System B uses the same caching technology for both read and write operations.

Write performance for File System A is at about disk speed which shows than it does write through without caching.

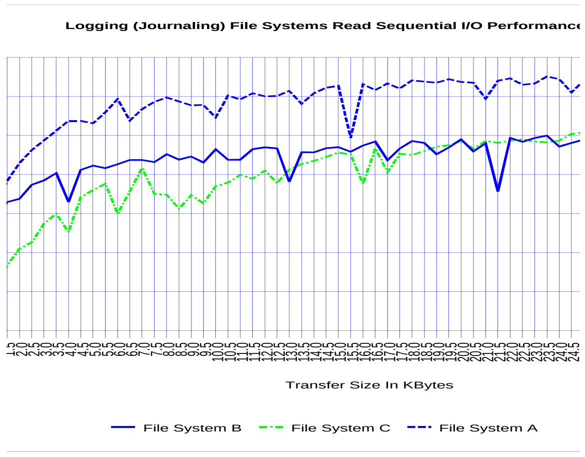


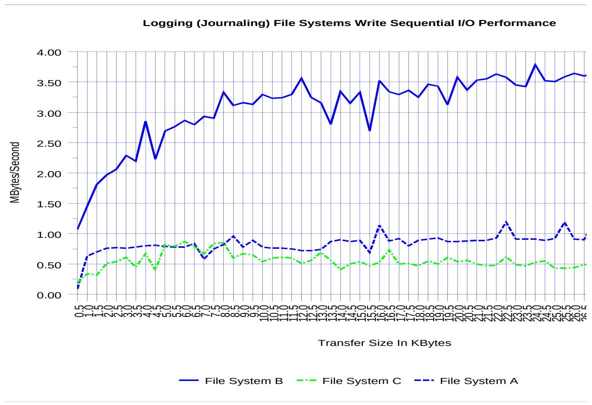
Figure 3 shows that File system B has good **pre-allocated write performance** for the same reason as mentioned earlier. Compared to write performance, pre-allocated write performance is a little bit better. The difference is probably due to the file system overhead to allocate extents in the first case.

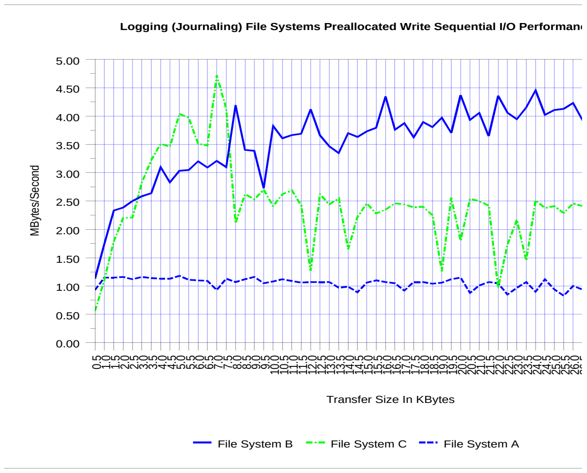
File System C takes very good advantage of pre-allocated writes compared to writes in the previous graph, especially for small size transfers (less than 8K bytes). It even outperforms File System B performance which is excellent. However, performance drops for larger transfer sizes with a lot of fluctuations.

3.5 Stress Testing

Stress tests were composed of a transaction test and a workload generation test.

The **transaction test** determines the file system behavior in a multi-user environment. The AIM Multi-users benchmark, Suite



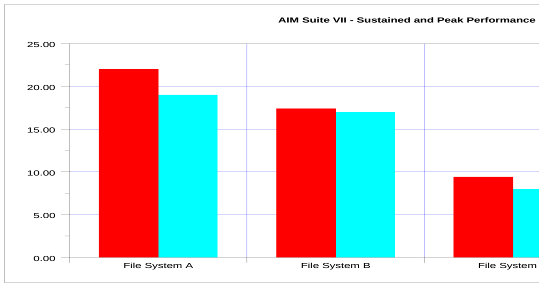


VII, has been used for this test. It is designed to mimic the activities of a specified system, such as a file server. In this case a workload contains sync writes, async reads, and block copies as well other file server's operations.

The program starts an increasing number of processes (task) which perform file server operations pseudo randomly. A weight is associated with each operation; the heavier the weight is, the most frequent the operation will happen. The benchmark runs a series of jobs to determine the system's cross-over, the point where the system's performance becomes less than 1 job per minute per AIM fileserver load. Peak performance is the maximum jobs/minute achieved. Sustained performance is the number of jobs at cross over.

These bars show that File system A can handle a little bit more transactions than File system B, especially in terms of peak performance. However, in terms of sustained performance the difference is minimal between the two file systems. File System B is better optimized as sustained and peak performance are the same.

The **workload generation test** populated a directory with either a normal, gamma or flat distribution of files. The goal is to fill a file system with as many files as possible. The objective was to be able to fill a file system with more than 1 million files.



The test was conducted on two file systems (File System A and File System B). The file system was 4GBytes striped across 4 disks. The maximum number of inodes was 1.98 millions inodes for File System A and 1.9 millions inodes for File System B. The test didn't complete in either cases because of a lack of available time. About 900,000 files were created in 23:35' hours for File System A and about 850.000 inodes allocated in 21:14' hours for File System B.

Although more inodes were allocated for File System A, File System B seems to allocate inodes faster. The goal of 1 million inodes would probably have been achieved if more time was available.

4 Conclusion

Whether it is from a feature availability point of view or from a technical point of view, one must conclude that there is no product that solves all the problems. Each file system evaluated has strong points and weakness. Each product has been developed with some priorities in mind and does a good job fulfilling these requirements.

The next step toward a future file system is to try to integrate the outside vendor technology into Cray's environment. Cray recognizes the strength of its native file system in terms of performance and is committed to deliver I/O performance.