CRAY T3D Experiments with I/O

R. Kent Koeninger, Cray Research, Inc., 655F Lone Oak Drive, Eagan, Minnesota 55121

ABSTRACT*: This paper shows I/O techniques and results on CRAY T3D systems.*

1 Introduction

In this paper I review the three phases of CRAY T3D I/O and give likely results of mixing phases I and II. I give general ruses for optimizing CRAY T3D I/O and show actual results from several sources.

2 Three Phases of I/O

All CRAY T3D systems in the field, as of this CUG, use phase-I I/O exclusively. Some sites will begin using phase II in the first half of 1995.

Phase I is the most general-purpose and easiest-to-use method of I/O. Most sites should use phase I, when they can, reserving phases II and III for configurations where phase I will not work.

Phase I copies all data through the host. This buffering in the host supports using any buffer size and any address for the I/O.

Phase II is useful with two processor hosts and with four processor hosts attached to MPPs with 256 or more processors. In these cases, one cannot use general-purpose phase I for every available MPP I/O gateway.

Phase II sends the data directly between the I/O subsystems and the MPP, bypassing the data buffering in the host. This improves the individual packet transfer performance, but not necessarily the overall system I/O performance.

Phase II requires special application code to align the data buffers on 8 word boundaries and requires all buffer lengths to be multiples of the disk sector size. This is called "well-formed" I/O.

With phase II configured, well-formed I/O will use the phase-II path and other I/O will use the phase-I path. Splitting the load between these two I/O paths is likely to lower the overall bandwidth, when compared to using phase I exclusively on the same number of channels.

The peak bandwidth of phases I and II are similar: about 130 megabytes per second (MB/s). For medium size transfers (8 sectors at 4096 bytes per sector), phase II transfers are about 50% faster: 20 MB/s for phase I and 30 MB/s for phase II. In theory, by carefully mixing phases I and II, one can see bandwidths higher than using the same number of gateways for phase I only. In practice, maintaining this balance is rare and using phase I exclusively usually yields higher overall system bandwidth.

On the other hand, if a host will support only two phase-I gateways, and the MPP will support four total gateways, configuring the last two gateways for phase-II I/O will increase the aggregate I/O bandwidth for the system.

Phase III is used only when there are more I/O gateways on the MPP than can be connected with phases I & II. It allows a connection to an I/O subsystem that has no physical connection to the host. This is useful for very large MPP installations or for MPPs with two-processor hosts.

3 Optimizing CRAY T3D I/O

3.1 Minimize System Calls

Each MPP I/O system call requires over 2 milliseconds of latency before the physical I/O starts and causes significant system overhead on the host. Bundling up I/O transfers into large blocks can significantly increase the bandwidths and reduce the system overhead. The FFIO libraries can be used to buffer data in individual processing elements (PEs), turning many small logical I/Os into fewer large physical I/Os.

3.2 Disk Configurations

System level striped file systems provide higher bandwidth for the general case where ASSIGN is not used, but are susceptible to disk contention for parallel I/O because each I/O touches every disk in the striped group. User level striping allows greater control to avoid contention, but provides lower bandwidth by default (without an ASSIGN statement).

System striped file systems increase the transfer rate on individual I/O requests. Using system striped file systems, a single PE can sustain about 50 MB/s. If many PEs attempt I/O to a striped file system, the resulting contention can significantly reduce the bandwidth.

The ASSIGN statement and user level striping allow the programmer more control over how the data is blocked on across multiple disks in a single file systems. The programmer can match the block size on the disk to the buffer sizes in the application. With care, each I/O will touch data on a single physical disk. This is useful for reducing disk contention when multiple PEs perform I/O in parallel.

Users generally don't specify user-level striping for a system-level striped file system. User level striping works only on multiple-disk, multiple-partition file systems. If one uses a unstriped file system without an ASSIGN statement, all the file system access will be sequential with each file generally laid out on one disk.

3.3 Parallel I/O

A simple I/O coding method is to use one PE for I/O and let it gather and distribute data to the other PEs using "gets" and "puts." With this technique and a fast file system, one can sustain 25 to 50 MB/s of bandwidth.

If one uses every PE for I/O in parallel, the contention on the disks is likely to lower the bandwidth to below that of using only a single PE.

If one uses one PE per physical disk, and each I/O touches only one disk, then one can achieve near-linear speedups in bandwidth as more disks are included in the parallel I/O. Using more than two PEs per disk tends to increase disk contention and lower the overall bandwidth.

3.4 Host I/O Considerations

CRAY T3D I/O is UNICOS host I/O. UNICOS I/O can be optimized by using LDcached file systems and striped I/O. File systems taking advantage of these features tend to have fast transfer rates.

3.5 Agent Tuning

For each partition on the CRAY T3D, the host runs a process called the Agent. This Agent processes all I/O requests for all PEs in the partition. The Agent will automatically expand its size to meet I/O demands, but it works best when initialized with sufficient memory. The MPP_AGENT_IO_MEM_MIN environment variable controls the initial size of the Agent.

The MPP_AGENT_PLOCK environment variable controls how the process will be held in the host's memory and how it will be relocated and swapped to disk. The default is "delay_shuffle", which locks the process in memory, but allows the host to determine when to move the Agent to one end of memory to avoid memory contention. Another option is "immediate_shuffle", which will relocate and lock the agent immediately. Some sites prefer to run with the agent not locked in memory. They find the memory load from locking Agents to be too large.

The MPP_AGENT_IOPATH environment variable allows one to specify the I/O should be direct (not buffered) which increases the I/O bandwidth. Direct I/O generally should be used with the "immediate_shuffle" option. By default, MPP_AGENT_IOPATH uses buffered I/O.

4 Results from Paul Helvig

Paul Helvig used user-level striping techniques and one or two PEs per physical disk for optimum I/O bandwidth. Using a single PE for I/O, he measured over 50 MB/s. Using 4 PEs for I/O he sustained over 130 MB/s. Using all 64 PEs for I/O, the bandwidth dropped to 15 MB per second. These rates were for write operations; read operations showed similar bandwidths.

Paul used asynchronous I/O and multiple buffers per PE to maximize the I/O rate per PE. He used direct I/O through the Agent.

5 Results from David O. Rich

David Rich reported that at the Los Alamos National Laboratory they sustained 50 MB/s to a HIPPI frame buffer using a single PE for the I/O. With every PE performing I/O, he sustained 20 MB/s and caused a heavy load on the host. He recommends using a single PE for I/O.

6 Results from Kah-Song Cho

Kah-Song Cho works for Cray Research at Los Alamos. He wrote an I/O library for the Parallel Ocean Program (POP). It gathers data into a CRAFT shared array, compressing out unneeded data (approximately a 50% compression). Each PE with data then performs I/O for the portion residing on that PE. Using Fortran direct access I/O he sustained about 25 MB/s.

7 Summary

I/O on the CRAY T3D can be optimized to deliver hundreds of megabytes per second of I/O. (Most applications do not have access to the number of parallel disks needed to sustain this very high bandwidth.) Many applications can use system level striped file systems and sustain 25 to 50 MB/s while using a single PE for I/O. With a little more coding effort and careful layout of the data on disk, one can sustain over 100 MB/s using 4 or 8 PEs for I/O.

The key factors to keep in mind are minimizing the number of system calls and minimizing the physical disk contention.

I owe thanks to Paul Helvig, Tim Gass, David Rich, and Kah-Song Cho for their excellent performance results.