Cray X1 System Performance

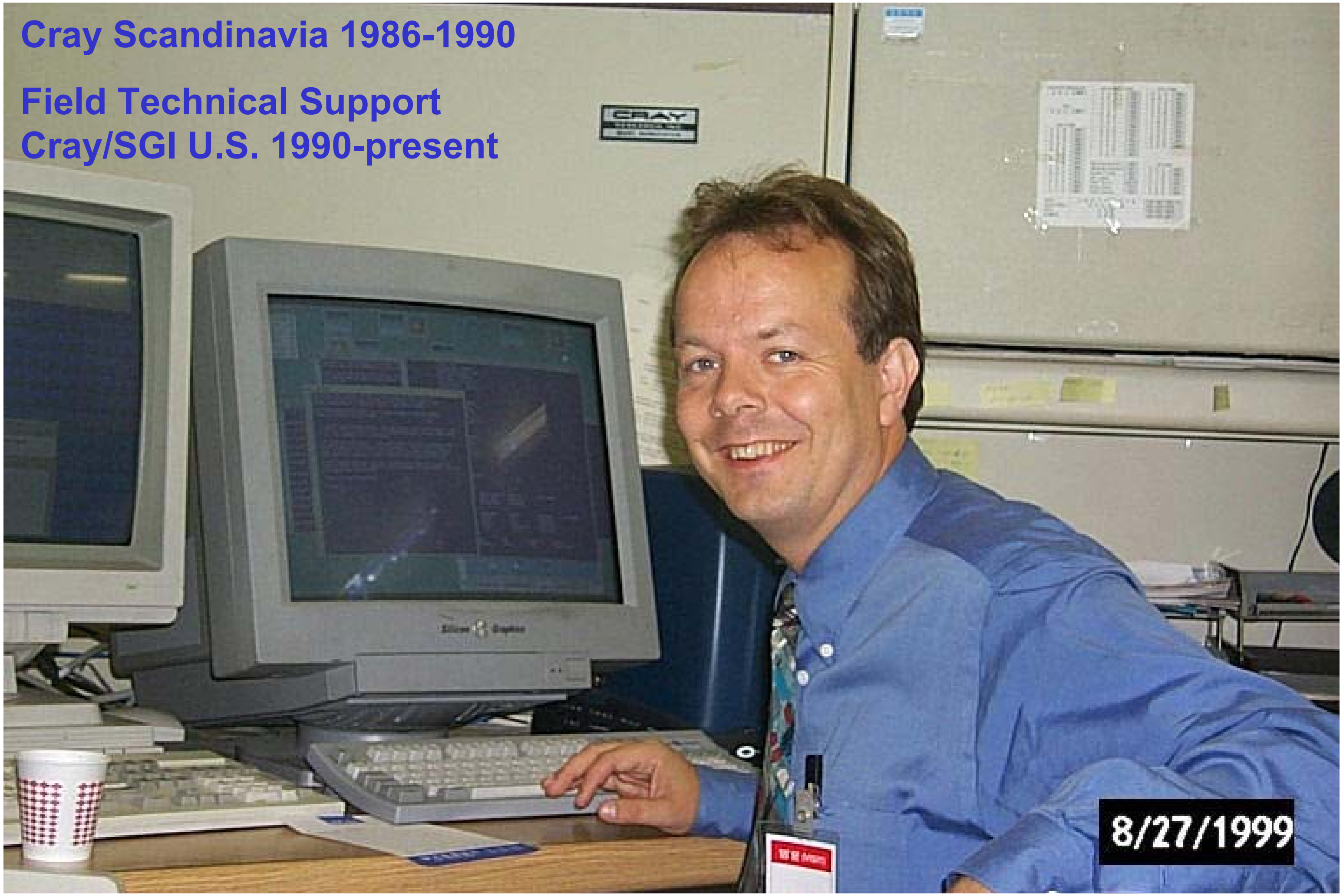
Frithjov Iversen
Field Technical Support
Cray, Inc.

Cray X1 System Performance

About the speaker

Cray Scandinavia 1986-1990

Field Technical Support
Cray/SGI U.S. 1990-present



Cray X1 System Performance

FTS – Field Technical Support

- Software support generalists
- Preinstallation/installation support
- Travel to support Cray Field Service
- Escalations

SPS – Software Product Support

- Software support specialists
- Preinstallation/installation support
- SPR processing, SWDEV interaction
- Escalations

Part of Cray Customer Service

CUG 2004

Classic System Performance considerations

individual job runtime

application performance

workload throughput

user/system/idle time

transfer speeds

user code performance, optimization

scheduling configuration (psched, PBS Pro, limits)

I/O configuration and use

tunable kernel parameters

CUG 2004

Cray X1 I/O

Configuration

- I/O hardware configuration
- RAID configuration
- Partitioning
- XLV/XFS configuration

Usage

- User I/O
- System I/O

Monitoring

CUG 2004

Uncovered topics

- Monitoring details
- Swapping
  - OS node
  - APP nodes
- Oversubscription
  - CPU oversubscription
  - Memory oversubscription
- SAN configuration, performance concerns
- Networking

CUG 2004

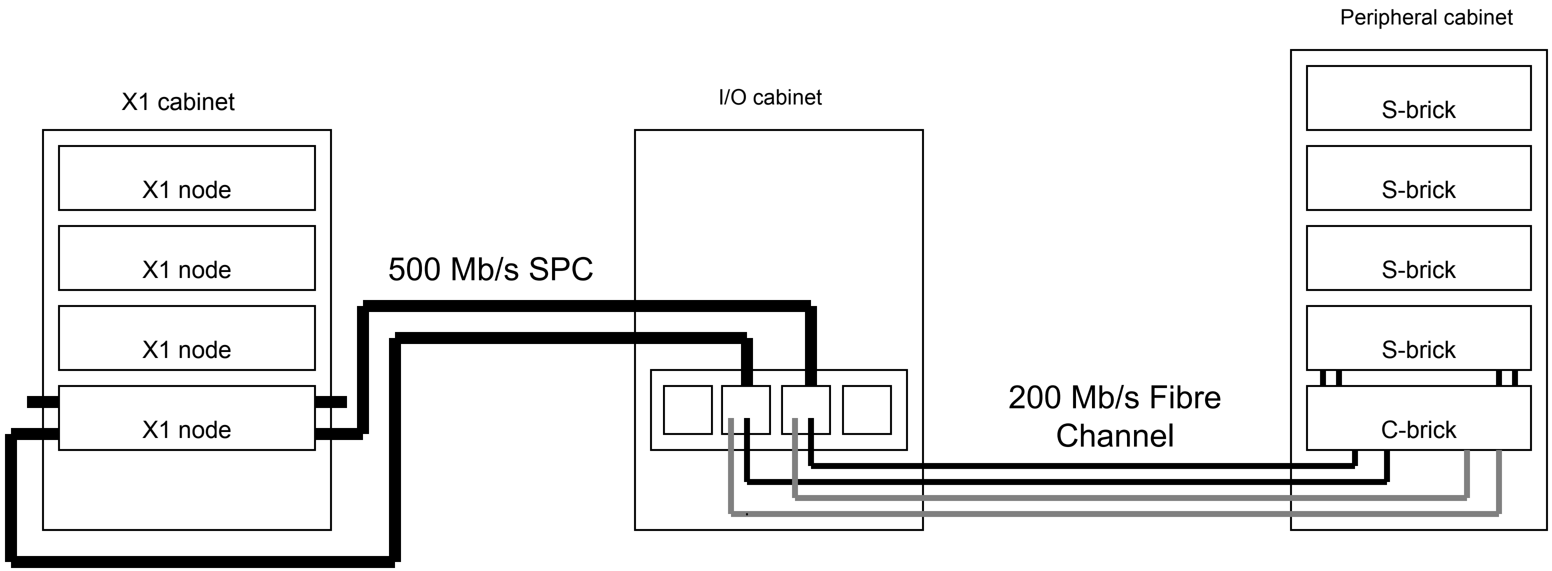
Cray X1 I/O

General principles for I/O performance

- Alignment of I/O requests and file allocations
- Avoid Unix buffer cache for large files
- Large transfer sizes
  - 1Mb or more
- Utilize parallelism:
  - asynchronous I/O
  - parallel I/O streams
  - balanced distribution of I/O across available hardware

CUG 2004

Disk cabling



Configuration determined by space, bandwidth requirements, price

Some cabling conventions and restrictions

Some bandwidth restrictions

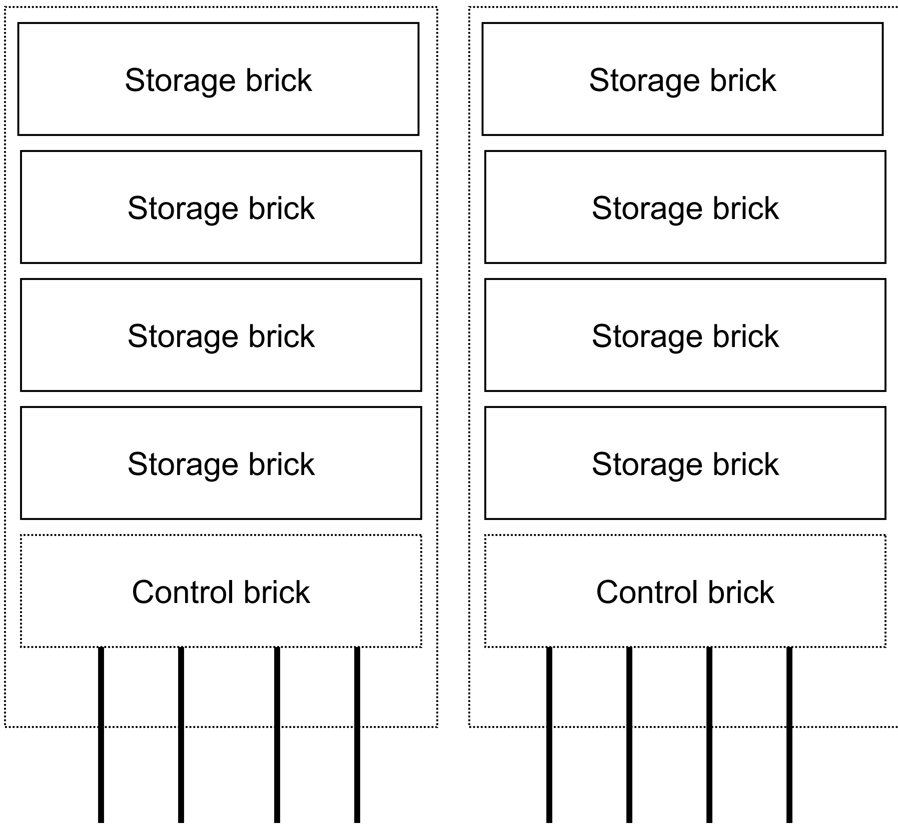
Reflected in X1 hardware configuration file

Shipping configurations are generally balanced, symmetric

CUG 2004

RAID configuration

Cray Storage Management (CSM)



Storage brick = 14 drives ("tray", "disk")

Each storage brick is assigned a RAID Group layout and split into LUNs

Using CSM commands for configuration rather than vendor tools

CUG 2004

Cray Storage Management (CSM)

Why CSM?

- **Standard interface that stays the same if the underlying RAID hardware/vendor changes**
- **Configuration control**
  - **Optimize performance for “Cray I/O”**
  - **Limited set of configurations to support/test**
  - **Control low level RAID controller parameters**
- **Maintain naming and numbering conventions**
- **Command line interface easier for remote support**
- *csmtune command initially planned for finer control*
  - *Change LUN write caching, segment sizes*

CUG 2004

RAID configuration

Cray Storage Management (CSM)

Prior to configuring CSM

- Collect known space requirements
- Collect known performance requirements

CUG 2004

RAID configuration

Cray Storage Management (CSM)

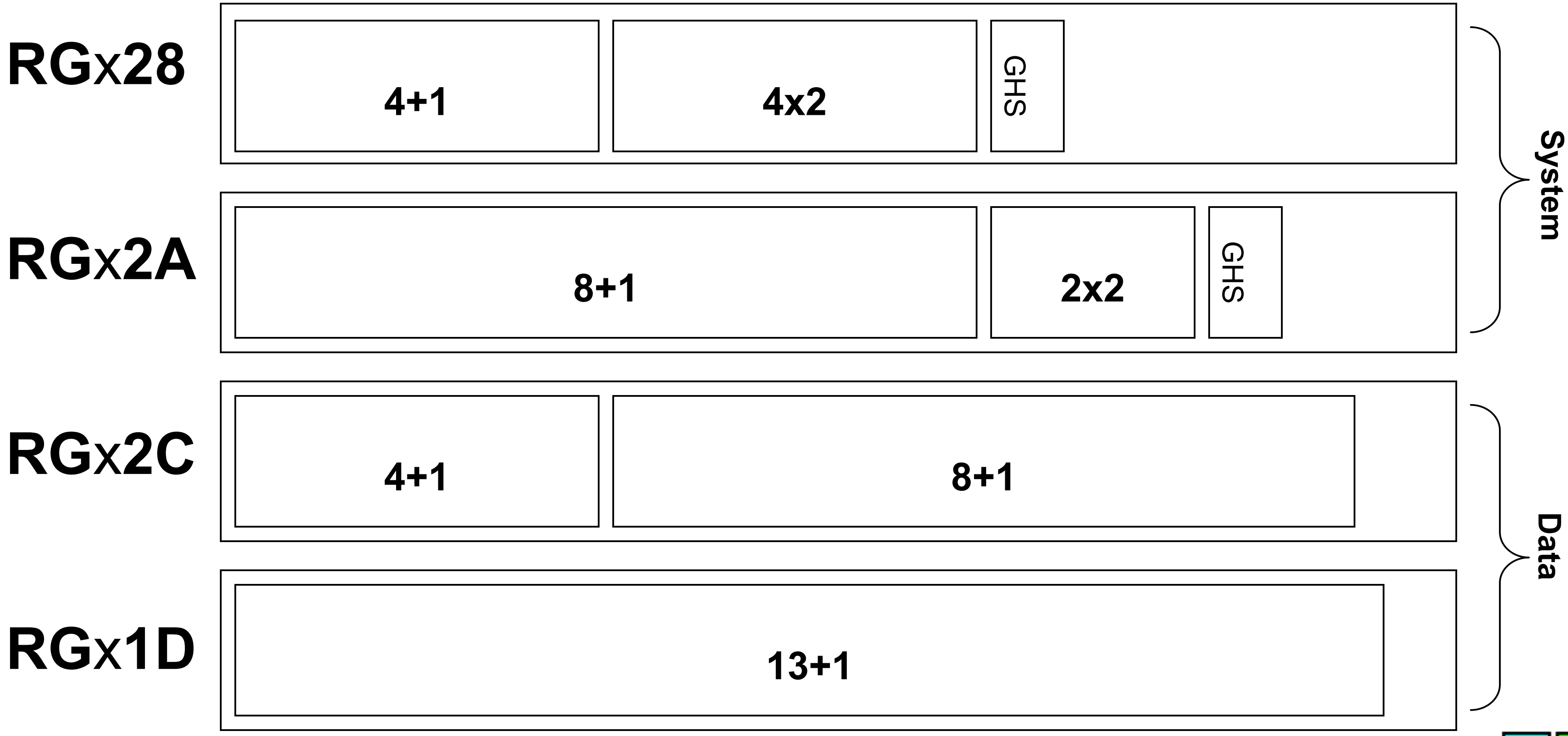
- CSM uses 'decimal' storage capacity denominations by default
  - 1 Gb = 1000 Mb
  - 1 Mb = 1000000 bytes
- Use -G option on commands to use power-of-2 values
  - 1 Gb = 1024 Mb
  - 1 Mb = 1048576 bytes

CUG 2004

RAID configuration

Cray Storage Management (CSM)

RAID Group Layouts



CUG 2004

RAID configuration

Cray Storage Management (CSM)

RAID Group Layout – segment/stripe sizes

geometry	width	segment size	stripe size	512b blocks
"transaction" LUNs				
2x2	2	128Kb	256Kb	512
4x2	4	128Kb	512Kb	1024
4+1	4	128Kb	512Kb	1024
"bandwidth" LUNs				
4+1	4	256Kb	1Mb	2048
8+1	8	256Kb	2Mb	4096
"capacity" LUNs				
13+1	13	128Kb	1664Kb	3328

CUG 2004

RAID configuration

Cray Storage Management (CSM)

RAID Group Layout – segment/stripe sizes

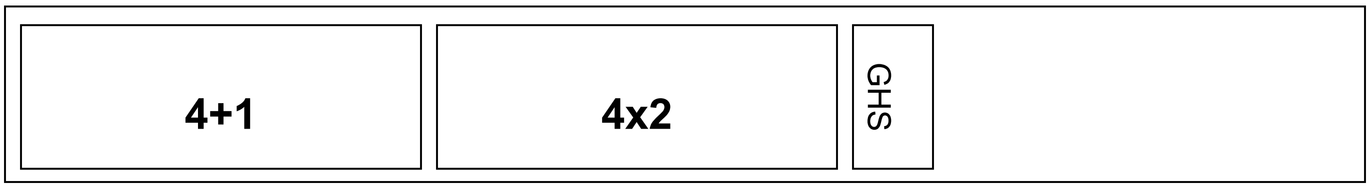
```
cws% csmreport
                                 …
 01d01  - RS200 starting at Chassis:  1   Slot:  1       Profile(01d01.prf)
        1 CB200 Cbrick, with 2 RC200 RAID Controllers
        4 SB201 Sbricks ( 56  73G spindles)
                                 …
 CTLR-A (top) 10.0.117.24          Optimal    Active
   Host chn:  Tid:  0 alpa: 0xEF   Optimal    Label: CTLR-A/1
              Tid:  2 alpa: 0xE4   Optimal    Label: CTLR-A/2
   LUNs:      LUN:  0 01d02_A0_L00 Optimal    RAID-5/4+1/128KB   80GB
              LUN:  2 01d02_A0_L02 Optimal    RAID-5/4+1/128KB  211GB
              LUN:  4 01d02_A2_L04 Optimal    RAID-1/4x2/128KB   80GB
              LUN:  6 01d02_A2_L06 Optimal    RAID-1/4x2/128KB  211GB
              LUN:  8 01d04_A0_L08 Optimal    RAID-5/4+1/256KB  291GB
              LUN: 10 01d04_A2_L10 Optimal    RAID-5/8+1/256KB  583GB
              LUN: 31 UTM-A        Optimal    PseudoLUN
```

CUG 2004

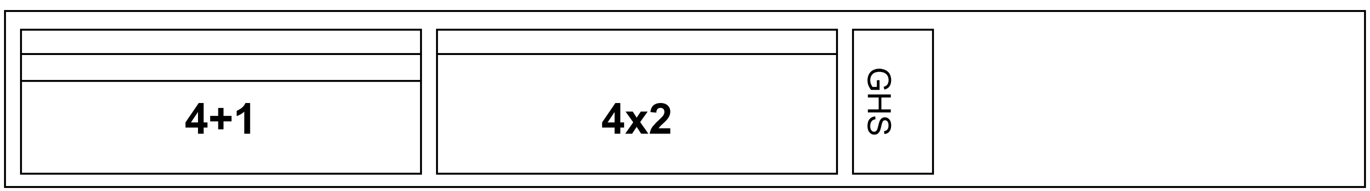
RAID configuration

Cray Storage Management (CSM)

Creating/remaking LUNs




```
cws% csmdelete 1 2 SBRICK
cws% csmadd 1 2 RG128 40:40:0 40:0
```



RAID configuration

Seeing LUNs and paths from the X1:

```
X1# /sbin/pm
pm4d7L0
        path  port    state  read blks  write blks  errs  MB/Sec
        ----  ----    -----  ---------  ----------  ----  ------
     fc2d0L0   pri   active     866071      812707     0    0.00
     fc4d2L0   pri   active     865884      817213     0    0.00
     fc1d1L0   alt  standby          0           0     0    0.00
     fc3d3L0   alt  standby          0           0     0    0.00

pm4d7L1
        path  port    state  read blks  write blks  errs  MB/Sec
        ----  ----    -----  ---------  ----------  ----  ------
     fc1d1L1   pri   active       2048           0     0    0.00
     fc3d3L1   pri   active       2048           0     0    0.00
     fc2d0L1   alt  standby          0           0     0    0.00
     fc4d2L1   alt  standby          0           0     0    0.00
```

CUG 2004

RAID configuration

Seeing LUNs and paths – details

```
X1# /sbin/pm -v tab
```

pm_nblks **LUN size**

pm_maxor **CTQ depth**

```
X1# /sbin/pm -v ustat
pm2d1L10           %rd  maxor  nor  wait   avg_rd  avg_wrt     sz/GBs
                  92.4      8    0     0   2991.2   3988.0      583.0

         path  port state      read blks      write blks err  MB/Sec
         ----  ---- -----      ---------      ---------- ---  ------
    fc20d0L10   pri   act      425839567        35239510   0    0.00
    fc22d2L10   pri   act      425929449        35049703   0    0.00
    fc21d1L10   alt  stby              0               0   0    0.00
    fc23d3L10   alt  stby              0               0   0    0.00
```

CUG 2004

RAID configuration

Partitioning LUNs with ‘parts’

Unicos/mp replacement for IRIX ‘fx’ tool

Volume header (disk label) at start of LUN

Slices aligned on 1Mb boundaries by default

```
X1# cat /etc/parts/root.cf
#
#    Partno      type       size
#
part    0                   25%
part    1        -t raw     15%
part    2                   30%
part    3                   30%
```

s0	root
s1	swap
s2	opt
s3	home

CUG 2004

RAID configuration

Partitioning LUNs with ‘parts’

- **Setting CTQ depth for device**
  - **CTQ default depth is 8**
- **‘modulo’ parameter for other alignments**
  - **2Mb for 8+1**
  - **13Mb for 13+1**

```
X1# cat option-RG11D.cf
#
#    Partno      type       size
#
modulo   13M
part     11                 100%
X1# /sbin/parts -w -q 8 -v \
   -c option-RG11D.cf pm0d6L4
```

Filesystem configuration

XFS filesystems

mkfs /dev/dsk/pm1d1L8s5

Creates an XFS filesystem with default parameters

Works, but does not give optimal I/O performance

Filesystem configuration

XFS filesystems

mkfs –d sunit=2048,swidth=2048 /dev/dsk/pm1d1L8s5

sunit	preferred alignment unit for allocation
swidth	preferred I/O size

Cheat sheet for sunit/swidth calculations

total stripe width	512b disk blocks	4k filesystem blocks
256K	512	64
512K	1024	128
1M	2048	256
2M	4096	512

Filesystem configuration

Creating XFS filesystems

X1# /sbin/mkfs –d sunit=2048,swidth=2048 /dev/dsk/pm1d1L8s11

```
meta-data=/dev/dsk/pm1d1L8s11    isize=256    agcount=272, agsize=261632 blks
data     =                       bsize=4096   blocks=71162368, imaxpct=25
         =                       sunit=256    swidth=256 blks, unwritten=1
naming   =version 2              bsize=4096
log      =internal log           bsize=4096   blocks=4096
realtime =none                   extsz=65536  blocks=0, rtextents=0
```

Output of XFS commands generally uses filesystem blocks (4k bytes)

This example is for a 4+1 LUN

mkfs with no options will show 0 for sunit/swidth fields

CUG 2004

Filesystem configuration

Inspecting mkfs parameters for XFS filesystems

X1# /sbin/mount /dev/dsk/pm1d1L8S11 /mnt

X1# /usr/sbin/xfs_growfs -n /mnt

```
meta-data=/dev/dsk/pm1d1L8s11    isize=256     agcount=272, agsize=261632 blks
data     =                       bsize=4096    blocks=71162368, imaxpct=25
         =                       sunit=256     swidth=256 blks, unwritten=1
naming   =version 2              bsize=4096
log      =internal log           bsize=4096    blocks=4096
realtime =none                   extsz=65536   blocks=0, rtextents=0
```

For filesystems mkfs'ed with no options

xfs_growfs will display 0 for sunit/swidth fields

CUG 2004

Filesystem configuration

Creating XLV volumes

```
X1# /usr/sbin/xlv_make
xlv_make>vol xlv0
xlv_make>log
xlv_make>plex
xlv_make>ve pm1d6L6s4
xlv_make>end
xlv_make>data
xlv_make>plex
xlv_make>ve -stripe -stripe_unit 4096
    pm1d1L1s11 pm1d6L1s11
    pm1d1L7s11 pm1d6L7s11
    pm1d1L10s11 pm1d6L10s11
xlv_make>end
```

8+1 pm1d1L10s11	**8+1** pm1d6L10s11
8+1 pm1d1L7S11	**8+1** pm1d6L7s11
8+1 pm1d1L1s11	**8+1** pm1d6L1s11

Use slices of same size, bandwidth, (RAID geometry)

Specify stripe unit in XLV configuration, using 512-byte blocks

Round-robin on controllers and IOCAs

mkfs should pick up sunit/swidth and log info from striped XLV devices

If using an external XFS log, 4x2 or 2x2 devices best for performance

Filesystem configuration

Displaying attributes of existing XLV volumes

```
X1# /usr/sbin/xlv_mgr
xlv_mgr> show -long xlv0
VOL xlv0 (complete)                (node=mfeg10)
VE xlv0.log.0.0  [active]
     start=0, end=262143, (cat)grp_size=1
     /hw/disk/pm1d6L6s4 (262144 blks)
VE xlv0.data.0.0 [active]
     start=0, end=6831611903, (stripe)grp_size=6, stripe_unit_size=4096
     /hw/disk/pm1d1L1s11 (1138601984 blks)
     /hw/disk/pm1d6L1s11 (1138601984 blks)
     /hw/disk/pm1d1L7s11 (1138601984 blks)
     /hw/disk/pm1d6L7s11 (1138601984 blks)
     /hw/disk/pm1d1L10s11 (1138601984 blks)
     /hw/disk/pm1d6L10s11 (1138601984 blks)
xlv_mgr> quit
```

CUG 2004

Filesystem configuration

XFS filesystem on striped XLV volume

```
X1# /usr/sbin/xfs_growfs -n /large
meta-data=/large                 isize=256    agcount=3258, agsize=262144 blks
data     =                       bsize=4096   blocks=853951488, imaxpct=25
         =                       sunit=512    swidth=3072 blks, unwritten=1
naming   =version 2              bsize=4096
log      =external               bsize=4096   blocks=32768
realtime =none                   extsz=65536  blocks=0, rtextents=0
```

Default internal log file size is 4000 filesystem blocks

Not sufficient for very fast filesystems > 300 Mbyte/s

CUG 2004

Filesystem configuration

XFS filesystem parameters – stripe unit, stripe width

stripe unit — alignment size for
- inode allocations
- internal log
- large files written through buffer cache

set to RAID stripe width
- RAID segment size * data width

this is *not* the "allocation unit"

stripe width — preferred I/O size
- st_blksize returned with stat(2)
- for use by libraries and commands

CUG 2004

Filesystem configuration

XFS filesystem parameters – log size

- internal log size defaults to 4000 filesystem blocks
- appears to limit performance to ~350 Mbytes/s
- make larger for striped XLV volumes
  - mkfs -l size=32768b internal log
  - or add external log to XLV device
- maximum usable log size is 128 Mbytes
  - or 32768 4096-byte blocks

CUG 2004

Filesystem configuration

XFS filesystem parameters – filesystem block size

Default is 4096 bytes

Corresponds to –P and –S for Unicos nc1fs

Change is supported but not recommended

- not proven to provide performance enhancements
- maximum of 64k bytes smaller than RAID segment size
- Increases the minimum size for all data in filesystem, including metadata, and all I/O transfers

CUG 2004

Filesystem configuration

XFS filesystem parameters – allocation groups

- Filesystems are divided into allocation groups
  - Default appears to be around 1 Gb per allocation group
  - Maximum allocation group size is a little less than 4 Gb
- New directories are allocated round-robin to different allocation groups
- Files stay in the allocation group of the directory
  - except when full
- Files can span multiple allocation groups
- Allocation groups allow for parallelism of f/s operations

CUG 2004

Filesystem configuration

XFS filesystem parameters – allocation groups

- **When fewer allocation groups might be needed**
  - Very large filesystems
  - Allocation group size limits size of extents
  - Large files get fragmented
  - System CPU time increases when filesystem gets full
- **When more allocation groups might be needed**
  - Small filesystems with many simultaneous transactions

CUG 2004

Filesystem configuration

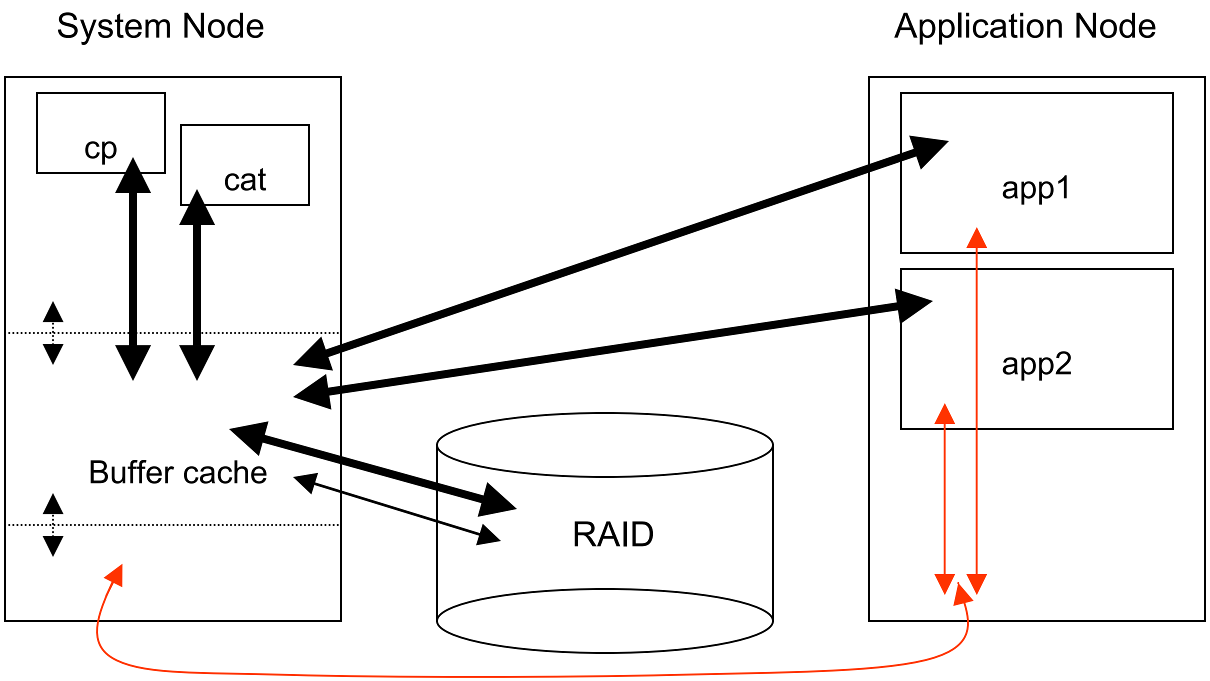
XFS filesystem parameters summary

stripe unit	**-d sunit=***blocks*	
stripe width	**-d swidth=***blocks*	
internal log	**-l size=***fs_blocks***b**	**4000**
filesystem block size	**-b** *bytes*	**4096**
allocation groups	**-d agcount=***number*	

CUG 2004

Cray X1 I/O overview

Buffered I/O



I/O system calls from application nodes are migrated to OS node.

Some additional cost.

Standard Unix I/O buffer cache

- **read/write system call I/O can be any size**
- **provides blocking for block devices, cache, readahead and writebehind**

Consumes memory from X1 OS node(s)

CUG 2004

Cray X1 I/O overview

Buffered I/O summary

- Differences from Unicos and previous Cray machines
  - dynamic size, lots of memory
  - faster memory-to-memory transfers
  - mature buffer cache management
- Provides
  - read-ahead, write-behind
    - write-behind important since disk controller cache is disabled
  - xfs allocation ‘sunit’ alignment
- Good for small I/O (up to single digit Gb files)

CUG 2004

Cray X1 I/O overview

Buffered I/O – how it works

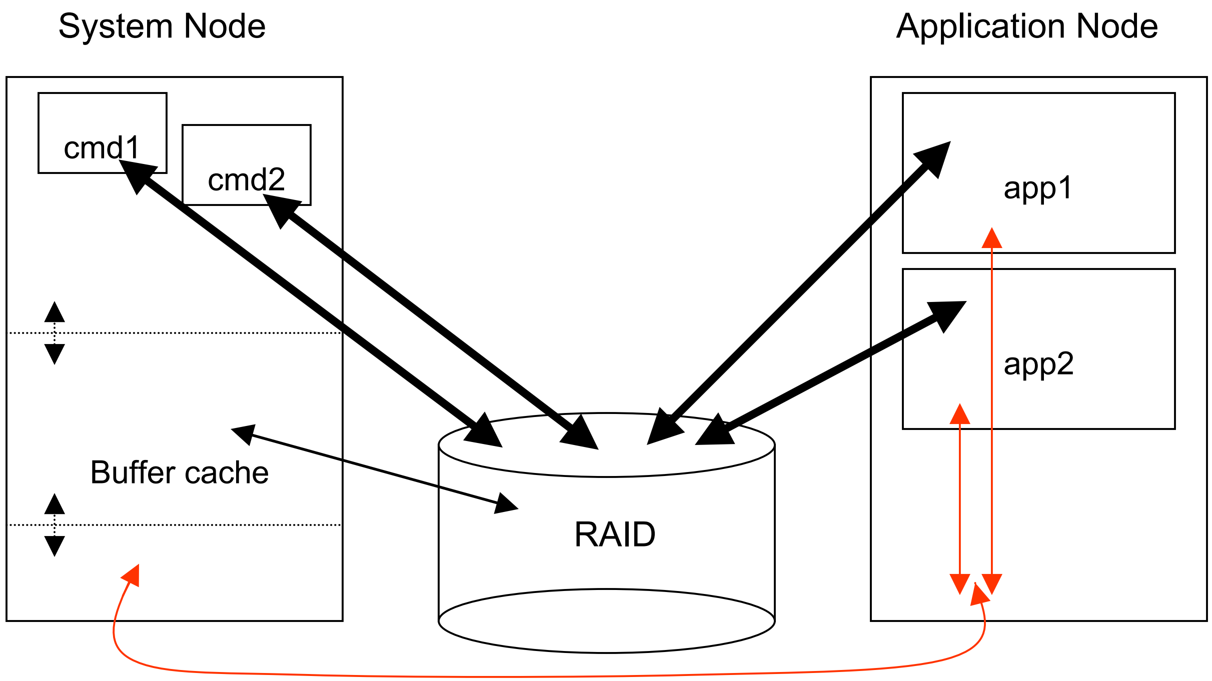
systune parameters for flushing dirty buffers illustrate this

- nbuf=256k (buffer headers = osmem / pagesz)
- bdflushr=5 (examine 1/5 of buffers every time)
- bdflush_interval=100 (1 sec.)
- dwcluster=64 (4 Mbytes, collect this much before writing)

Note: little experience with changes other than for ‘nbuf’

CUG 2004

Direct I/O



I/O system calls from application nodes are migrated to OS node.

Some additional cost.

Data flows directly between user space and disk

- **alignment and transfer size restrictions**

Filesystem metadata still flows through buffer cache

Fewer OS node resources (memory, system CPU time) needed

Cray X1 I/O overview

Direct I/O details – how it works

- Data transfers bypass buffer cache
- Writes are synchronous
- XFS allocations are immediate
- User data size and alignment requirements
- Maximum transfer size is 16Mb
  - Higher transfer sizes would require disk driver change

CUG 2004

How to use direct I/O

From C programs:

```
int fd; long datasize;
struct dioattr d;
          .
          .
fd = open("/tmp/outfile",O_DIRECT|…,…);
          .
          .
ret = fcntl(fd, F_DIOINFO, &d);
buf = (unsigned long long *) memalign(d.d_mem, datasize);
          .
          .
write(fd,buf,datasize);   # datasize must be d.d_miniosz multiple
```

CUG 2004

Cray X1 I/O overview

How to use direct I/O with assign and FFIO cachea layer

With FORTRAN programs

With C programs that use ffopen()/ffread()/ffwrite()

```
module load PrgEenv
export FILENV=$HOME/.assign
assign -R
assign -B on -F cachea:4096:8:8 u:21
./my_prog
```

'-B on' enables O_DIRECT open flag

-F selects FFIO layers and options:

cachea asynchronous, cached I/O layer

Buffer size is 4096 512-byte blocks (2Mb)

8 buffers total – allocated with proper alignment for O_DIRECT

8 buffers max. readahead

'u:21' applies to FORTRAN unit 21

CUG 2004

How to use direct I/O – preallocation

Preallocation from C programs

```
struct flock fl;
        …
fl.l_whence = SEEK_SET;
fl.l_start = 0LL;
fl.l_len = (long long) filesize;
fcntl(fd, F_RESVSP, &fl)
```

Preallocation with assign/FFIO

“allocate ahead” feature within cachea layer
planned for Programming Environment 5.2 Update 1 or 2

CUG 2004

Cray X1 I/O overview

Direct I/O summary

- Differences from Unicos O_RAW
  - Not automatic with well formed large I/O
  - Proper alignment of memory buffers required
- Need asynchronous I/O for performance
- xfs allocation/alignment concerns - preallocate
- Consider for large files (> a few Gb)
- Example: Checkpoint/Restart
  - cpr uses direct I/O for checkpoint files as of U/mp 2.3

CUG 2004

Cray X1 I/O overview

I/O from System node vs. Application node

- Some cost/delay associated with migrating I/O system calls from APP nodes to OS node
- #syscalls/second capacity – write(2) example

from OS node	5500	calls/sec.
from APP node	500	calls/sec.

- Alleviate with
  - large I/O sizes, library buffers
  - asynchronous I/O

CUG 2004

Cray X1 I/O overview

Kernel tuning for I/O

- systune changes generally not required
- some sites have reduced ‘nbuf’ to prevent hangs

CUG 2004

Cray X1 Filesystems

“transaction” LUNs – 2x2, 4x2, 4+1

- root
- /opt
- /var
- /var/…
- /tmp
- home filesystem(s)
- swap
- external XFS log devices

“bandwidth” LUNs – 8+1, 4+1

- very fast/large filesystems
- checkpoint filesystem
- dump filesystem

“capacity” LUNs – 13+1

- very large filesystems

CUG 2004

Cray X1 Filesystems

root

- **3-4 copies for shipment**

/var

- **can be separated from root to reduce writes and control root size**

/var/adm

- **Can be separated to contain accounting data**

/var/spool

- **Can be separated to contain PBS spooled data**

CUG 2004

Cray X1 Filesystems

/opt

Could be a small slice on “transaction device” if not used for anything else

/tmp

Could be a small slice on “transaction device” if not used for anything else

CUG 2004

Cray X1 Filesystems

Home filesystems

- NFS-export to CPES

Swap slices

- First swap slice configured in nvram
- Additional slices configured in /etc/fstab

External XFS log devices

- 128Mb appears sufficent
- Recommended with striped XLV devices

CUG 2004

Cray X1 Filesystems

Very fast/large filesystems

- XLV-stripe LUNs on alternate RAID controllers, IOCAs
- External XFS log is recommended
- Communicate best I/O sizes and access methods to users
- Monitor for large files

System dump directory

- Can reside in checkpoint filesystem or on other fast/large device
- Dump files written with FFIO, buffer sizes up to 8Mb
- Typical sizes are 500 Mb per OS node, 55 Mb per APP node

Checkpoint filesystem

- ‘cpr’ uses O_DIRECT and cachea with 8x16Mb buffers
- PBS Pro requires permission bits 0700 on checkpoint directory
  - and 0755 on parent directories within filesystem

I/O tests

Internal Cray Service I/O test

- Used by FTS/SPS as basic I/O diagnostic
- System calls
- Varying I/O sizes
- Buffered or direct I/O

Streams

- General benchmark for user I/O
- Highly configurable
- Supports asynchronous I/O, preallocation

FORTRAN I/O

I/O tests

How to read the graphs

Primary measurements

Write Mb/sec.

Read Mb/sec.

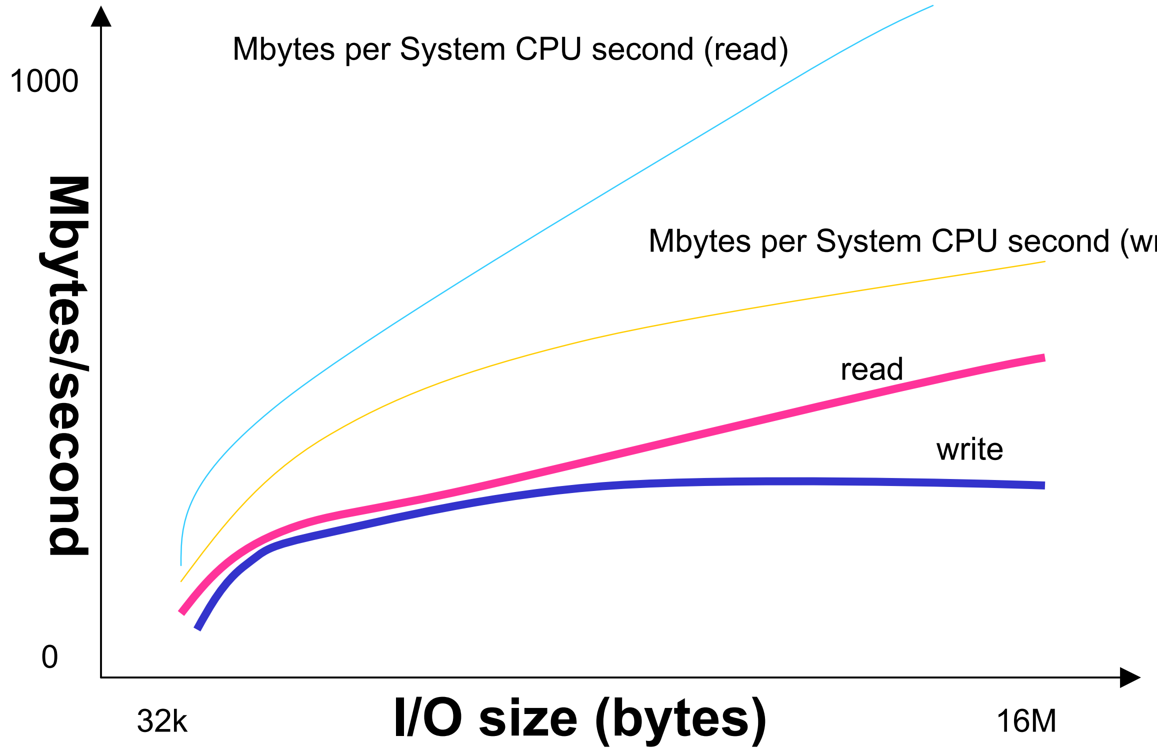
Scenarios shown in next 4 slides

Buffered I/O – OS node

Buffered I/O – APP node

Direct I/O – OS node

Direct I/O – APP node



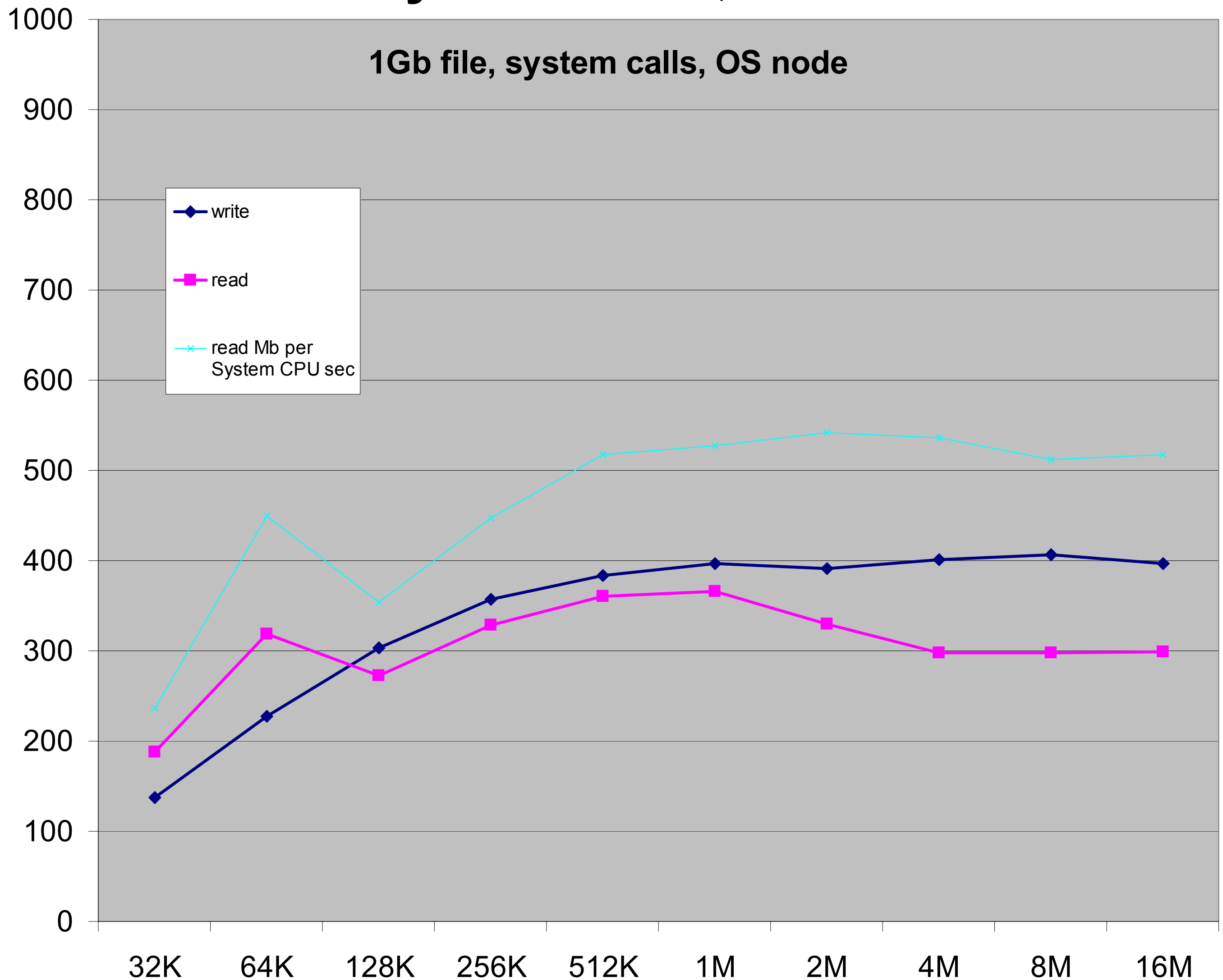
Mbytes per System CPU second as an unscientific metric for system resources needed to move data

using numbers from 'time' command

CUG 2004

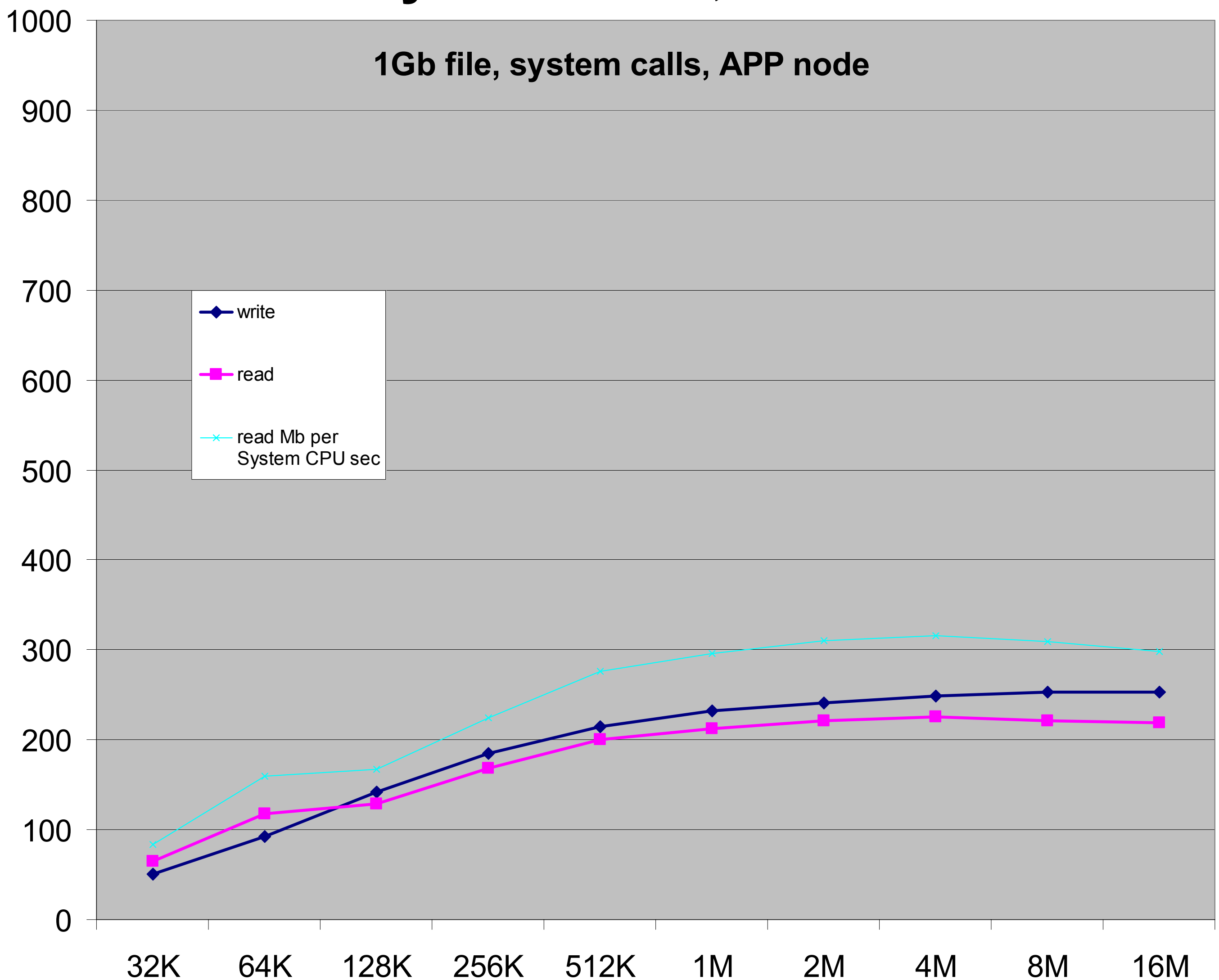
I/O tests

Buffer cache I/O – system calls, OS node



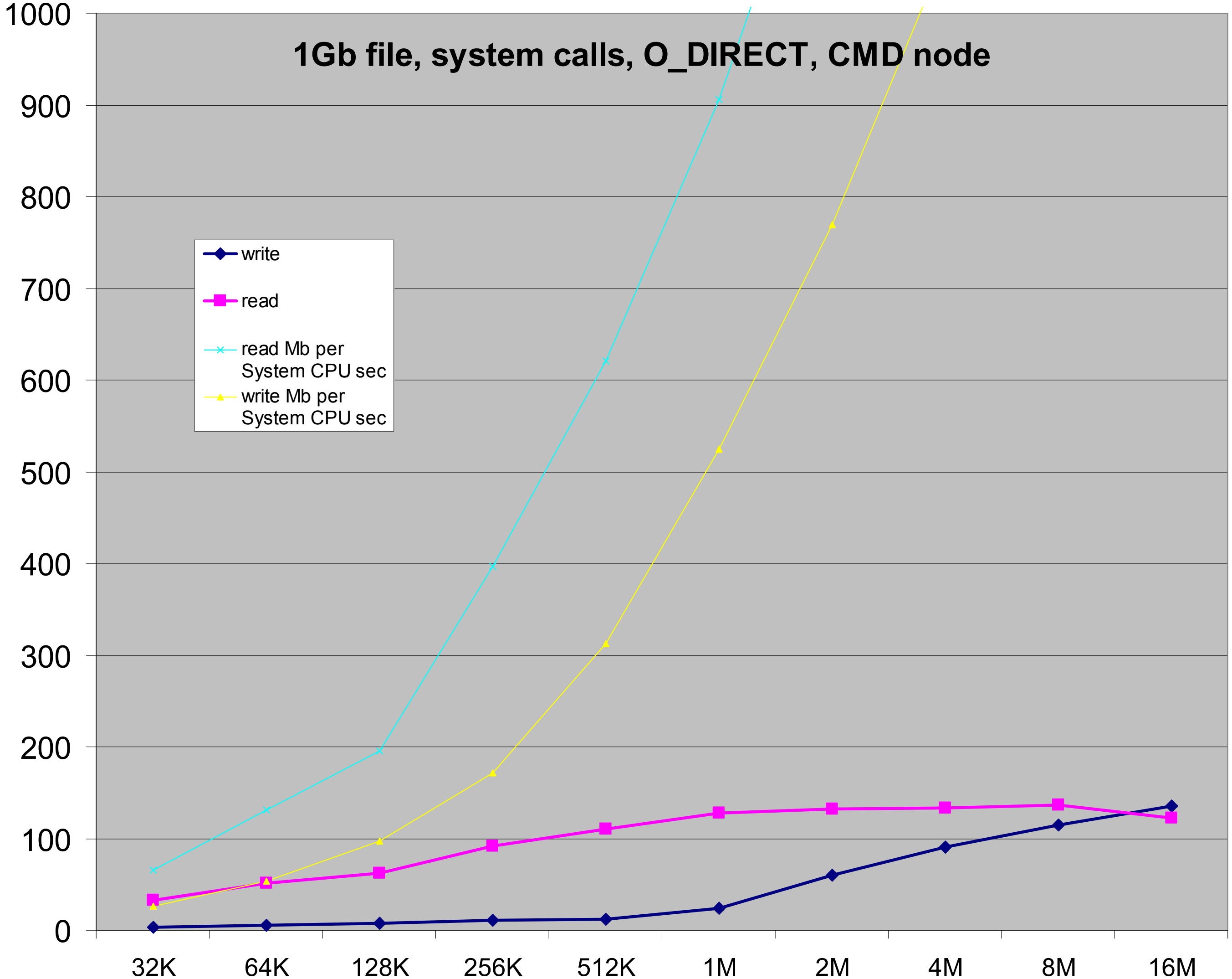
CUG 2004

Buffer cache I/O – system calls, APP node



I/O tests

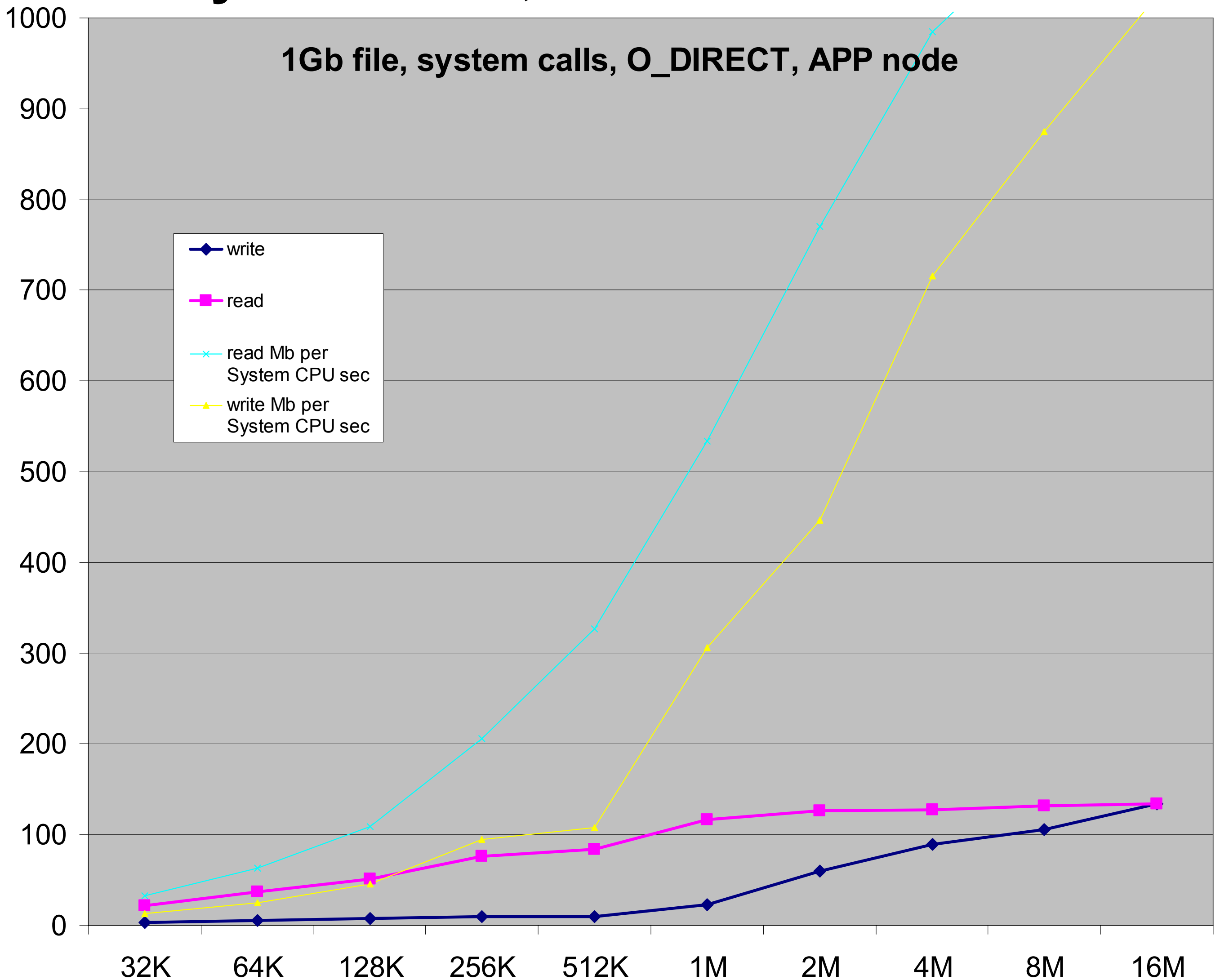
Direct I/O – system calls, OS node



CUG 2004

I/O tests

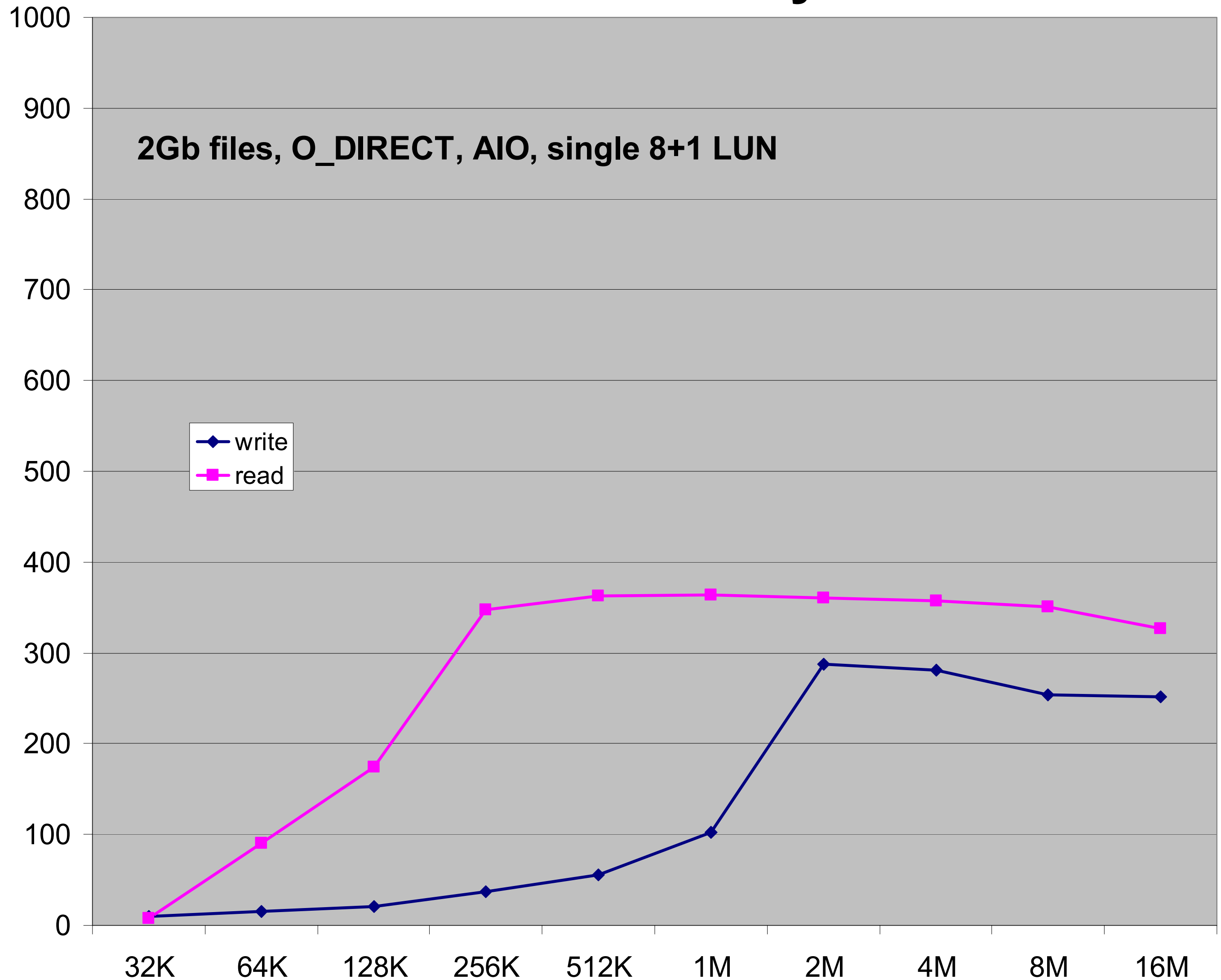
Direct I/O – system calls, APP node



CUG 2004

I/O tests

Direct I/O – STREAMS with 16x asynchronous I/O



CUG 2004

I/O tests

STREAMS benchmark – RAID stripe boundary alignment

8+1 LUN (256kb segment size * 8 = 2Mb optimal alignment)

mkfs –s sunit=4096,swidth=4096

Aligned I/O:	**271 Mb/s**	**write**

- 2 Gb file size
- preallocated, contiguous space
- 2Mb I/O transfer size
- async I/O, 8 buffers
- Direct I/O

Non-aligned I/O:	**136 Mb/s**	**write**

- As 1, with offset 256Kb into file

CUG 2004

I/O tests

STREAMS benchmark – large XLV volume

6-wide 8+1 XLV volume, across two RAID controllers

10Gb file size

RAID stripe aligned I/O (2Mb)

Preallocated space

Asynchronous direct I/O, 16 buffers

2Mb I/O size:	362 Mb/s	write
	891 Mb/s	read
12 Mb I/O size:	847 Mb/s	write
	1011 Mb/s	read

CUG 2004

I/O tests

FORTRAN WRITE example

```
      program io
      parameter (n=10000)
      integer a(n)

      do 5 i=1,10000
5     a(i) = i+1

      do 10 i=1,1000              ! 1000 = 40Gb file, 100 = 4Gb file
      do 10 j=1,1000
10     write(21) n, a

       close(21)

      end
```

CUG 2004

I/O tests

FORTRAN WRITE example – results

```
module load PrgEnv
ftn -h command -o io1 io1.f
rm -f fort.21
export FILENV=/users/fi/.filenv
assign -R
assign … u:21
timex ./io1
```

4Gb file	real	user	sys	
Single 8+1 LUN	34.8	5.8	19.0	as-is
32Gb OS node	10.8	6.2	10.0	assign –F cachea:4096:8:8
	19.5	6.1	2.9	assign –B on –F cachea:4096:8:8
40 Gb file	**real**	**user**	**sys**	
Single 8+1 LUN	361	57	228	as-is
32Gb OS node	328	60	134	assign –F cachea:4096:8:8
	184	60	67	assign –B on –F cachea:4096:8:8

CUG 2004

System Administration tasks

- Setting up ‘sar’
- Setting up accounting
- Checking/repairing/reorganizing filesystems (fsr)
- Configuring psched, PBS, limits to manage workload and memory subscription
- Remaking filesystems – moving/shuffling data

CUG 2004

X1 I/O monitoring tools

sar

-d/-D	I/O rates per LUN
-b	Buffer cache vs. direct I/O
-T	I/O totals per LUN, use with –D and -b

bufview

monitoring buffer cache I/O

acctcom

Locating processes doing large I/O, high system CPU time

xfs_db

Allocation/fragmentation details for XFS files

xfs_bmap

scheduled for Unicos/mp 2.5

pm

monitoring I/O paths

CUG 2004

Conclusions

General principles for I/O performance

- Alignment of I/O requests and file allocations
- Avoid Unix buffer cache for large files
- Large transfer sizes
  - 1Mb or more
- Utilize parallelism:
  - asynchronous I/O
  - parallel I/O streams
  - balanced distribution of I/O across available hardware

CUG 2004

Cray X1 System Performance

Frithjov Iversen
Field Technical Support
Cray, Inc.
fi@cray.com