BRUDEN-OSSG
On-Shore Systems Group

The Performance Cookbook

October 2006

Bruce Ellis & Guy Peleg
BRUDEN-OSSG
bruce.ellis@bruden.com
guy.peleg@bruden.com

Agenda

- O/S
- Applications
- RMS
- System management
- Troubleshooting tools
- Simulators

Disclaimer

“Si vous n’aimez pas ma conduite, vous n’avez que descendre du trottoir.”

-anonymous

The Golden Rules

Source: OpenVMS Information Desk – October 2004

The best performing code is the code not being executed

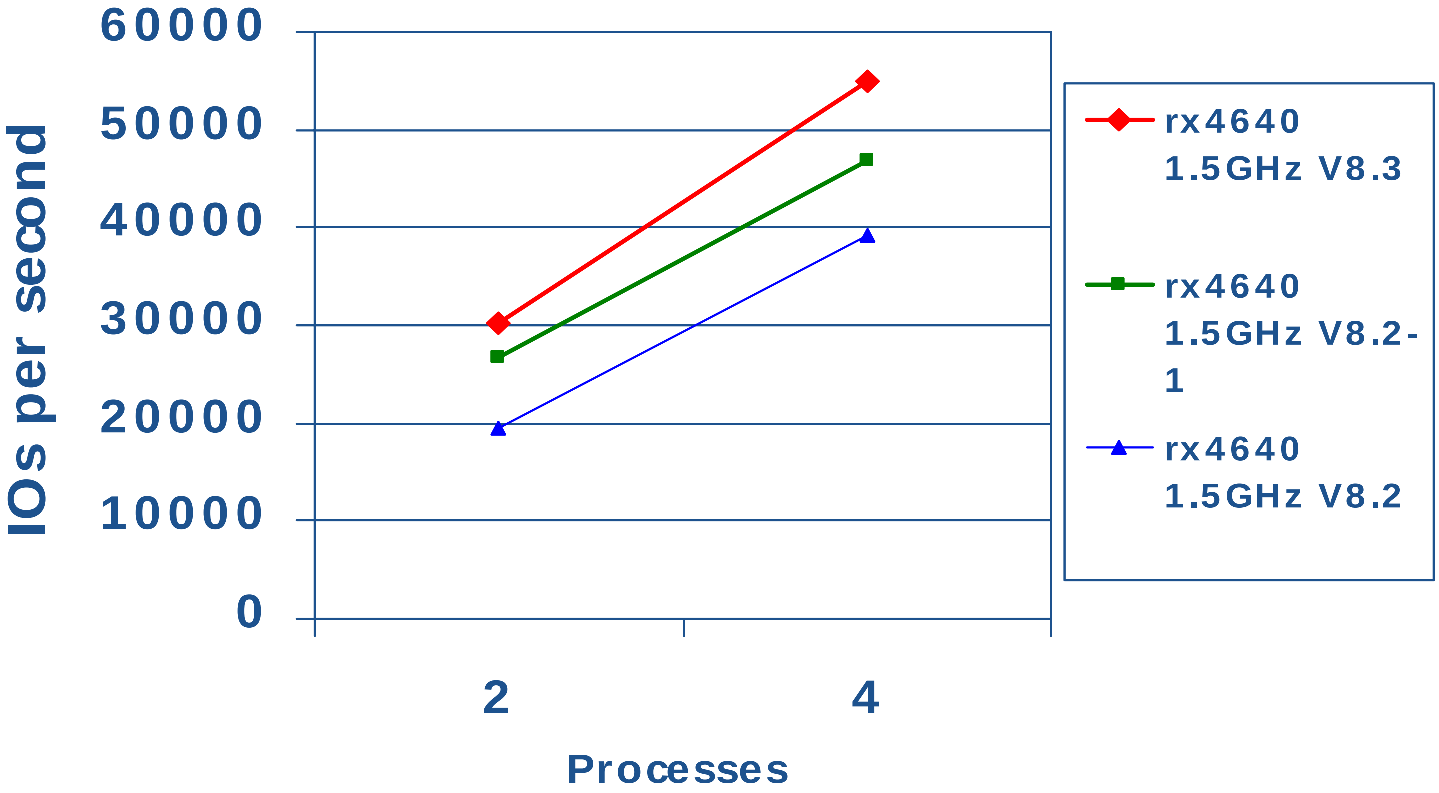
The fastest I/Os are those avoided

Idle CPUs are the fastest CPUs

Upgrade

- V8.2
  - IPF, Fast UCB create/delete, MONITOR, TCPIP, large lock value blocks
- V8.2-1
  - Scaling, alignment fault reductions, $SETSTK_64, Unwind data binary search
- V8.3
  - AST delivery, Scheduling, $SETSTK/$SETSTK_64, Faster Deadlock Detection, Unit Number Increases, PEDRIVER Data Compression, RMS Global Buffers in P2 Space, S2 Code GH Region, alignment fault reductions

RMS1 (Ramdisk) OpenVMS Improvements by version



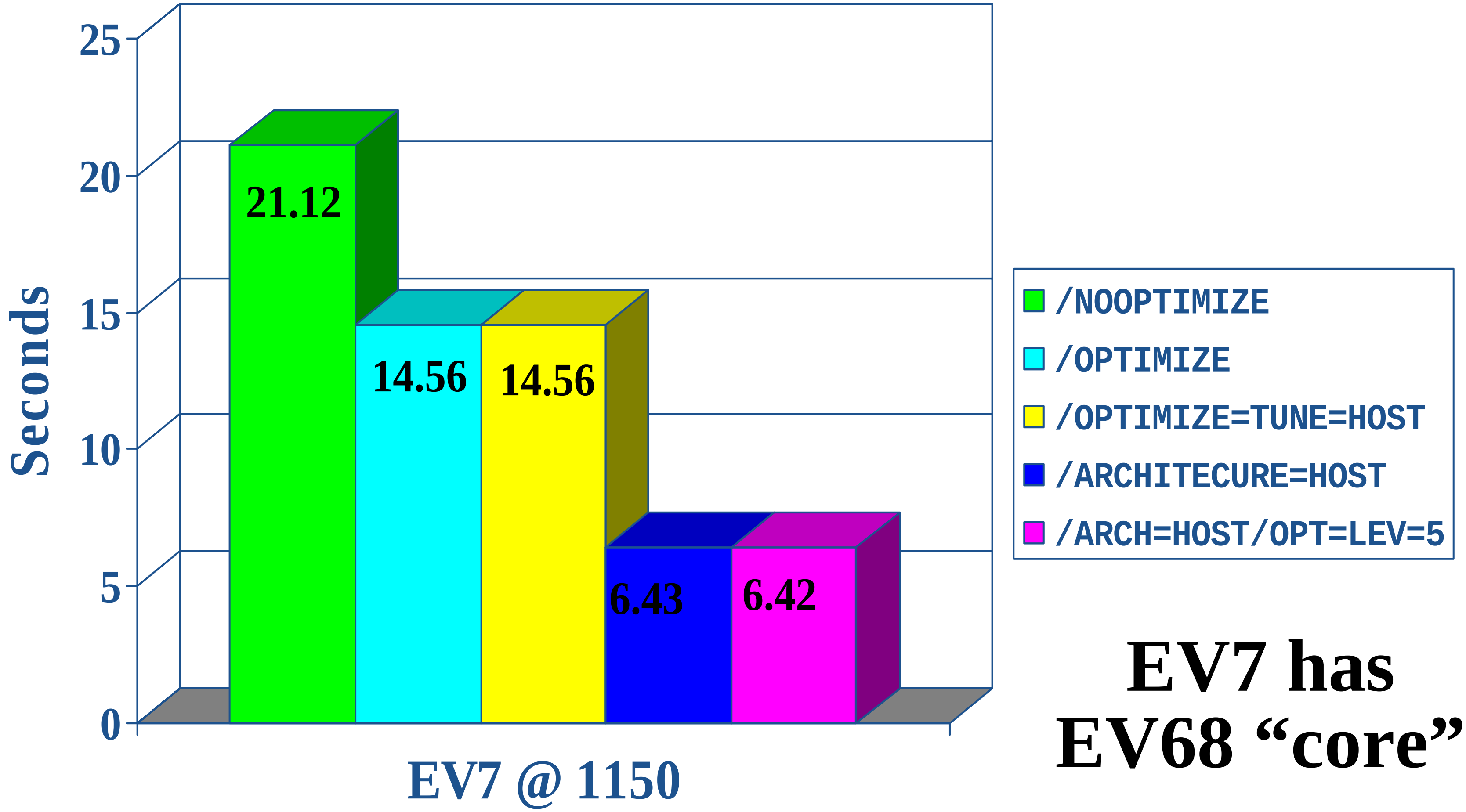
More is better

Performance enhancements to the application hold the greatest potential for improving performance

Examples of …TUNE & /ARCHITECURE

- **/OPTIMIZE=TUNE=EV56**
  - Execute on all Alpha generations
  - Biased towards EV56
- **/OPTIMIZE=TUNE=EV6 /ARCHITECTURE=EV56**
  - Execute on EV56 and later (Byte/Word instructions)
  - Biased for EV6 (quad issue)
- **/ARCHITECTURE=EV6**
  - Execute on EV6 and later (Integer-Floating conversion, Byte/Word & Quad-issue scheduling)
- **/ARCHITECTURE=HOST**
  - Code intended to run on processors the same type as host computer
  - Eexecute on that processor type and higher

Generating Primes GS1280 7/1150



EV7 has EV68 "core"

Initializing Structures - which is fastest/efficient?

- Initializing structures in BLISS....

    …..Wait a second, how many people around here use BLISS….☺

    …… Let’s try again…..

Initializing Structures - which is fastest/efficient?

```
void foo1 (){
    char array[512]={0};
    printf("array=%x",&array);}

void foo2 (){
    char array[512];
    for (int i=0;i<512;i++) array[i]=0;
    printf("array=%x",&array);}

void foo3 (){
    char array[512];
    memset (array, 0, sizeof(array));
    printf("array=%x",&array);}
```

setjmp

```
main(char **av, int ac)
{ time_t tm = time(0);
  int i, env, nosetjmp = 0;

  if ((ac == 2) && (*av[1] == '-')) {
    printf("No setjmp\n");
    nosetjmp = 1; }

  lib$init_timer();

  for (i = 0; i++ < 1000000;) {
    if (nosetjmp) env = i;
    else {
      env = setjmp(g_jmpbuf);
      if (env) printf("Jumped\n"); } }
lib$show_timer(); }
```

setjmp

- Takes 45 seconds to execute this program on 8P Superdome (1.5GHZ)

- Compiled with **`/define=__FAST_SETJMP`** program takes only 0.05 seconds

LIB$FIND_IMAGE_SYMBOL

- LIB$FIS searches for translated image if lookup failed
- Not using translated images?
  - Set LIB$M_FIS_TV (Alpha)
  - Set LIB$M_FIS_TV_AV (IA64)

- Watch out for new Binary Translator (V2) with several performance improvements
  - *Don’t get too excited, TI are still slow*

Application Temporary Files

- Frequently create/delete small temp files?
  - Consider caching in virtual memory instead
  - “Spill” to disk file if needed after some threshold (1mb?)

- Don’t be afraid of P2 virtual address space
  - Keep an eye out for excessive page faulting

Parallel Compilation

- PIPE spawns a sub-process for each pipe segment
  - Easy multithreaded build
  - No need for SUBMIT & SYNCHRONIZE
- Some compilers allow several source modules to be specified at once

Example – compiling 3 modules

- ***Serial compilation***

```
Accounting information:
      Buffered I/O count:              353      Peak working set size:      23584
      Direct I/O count:                214      Peak virtual size:         221680
      Page faults:                    4227      Mounted volumes:                0
      Charged CPU time:      0 00:00:00.90      Elapsed time:       0 00:00:02.30
```

- ***Parallel compilation using PIPE***

```
  Accounting information:
      Buffered I/O count:              104      Peak working set size:       4400
      Direct I/O count:                 27      Peak virtual size:         177120
      Page faults:                     319      Mounted volumes:                0
      Charged CPU time:      0 00:00:00.04      Elapsed time:       0 00:00:01.23
```

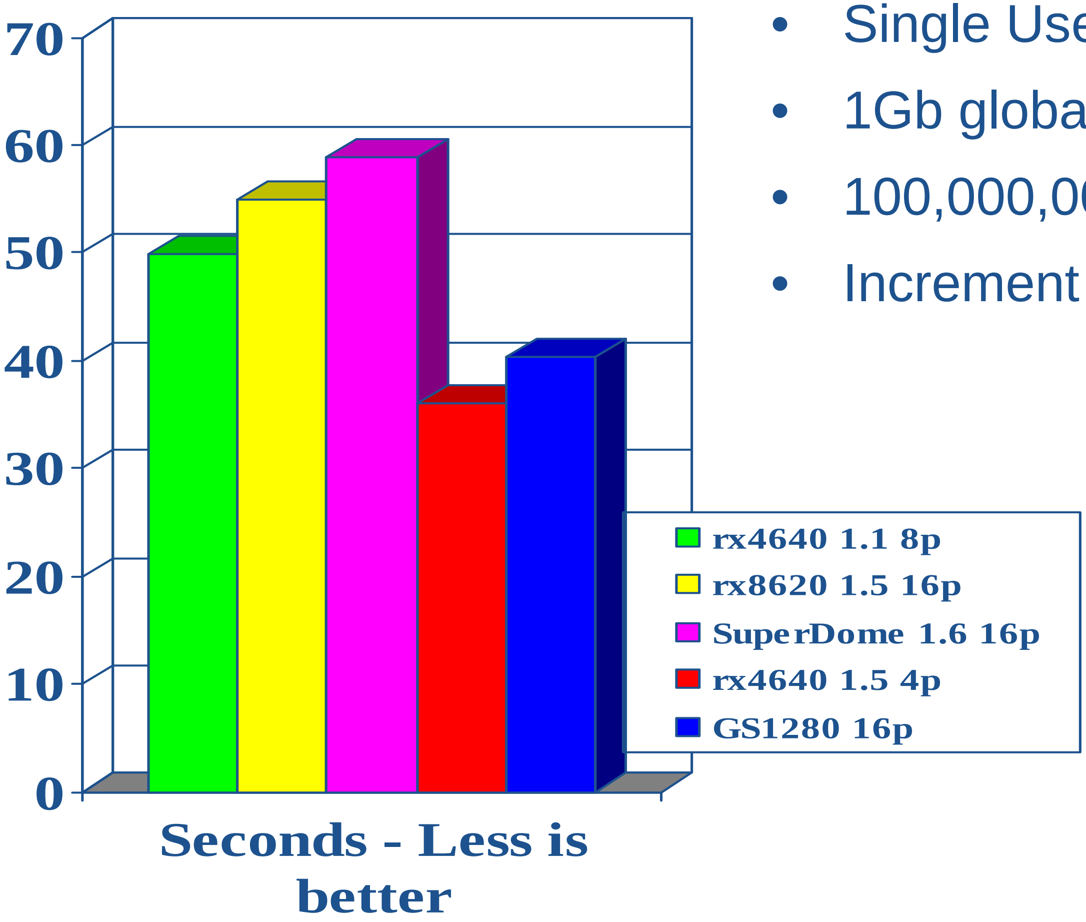
- ***Single command***

```
 Accounting information:
  Buffered I/O count:              265      Peak working set size:      25600
  Direct I/O count:                175      Peak virtual size:         221840
  Page faults:                    3044      Mounted volumes:                0
  Charged CPU time:      0 00:00:00.70      Elapsed time:       0 00:00:01.85
```

FLT - Alignment Fault Tracing

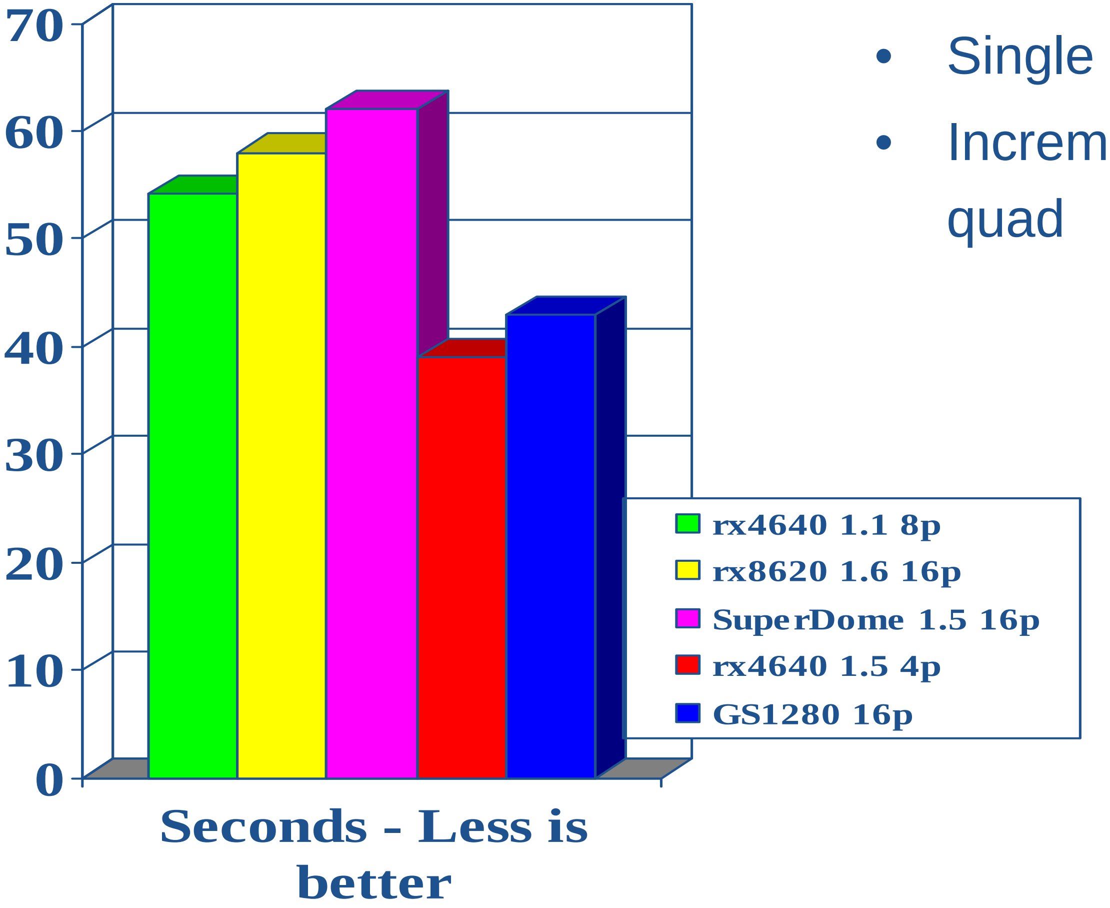
- Ideal is no alignment faults at all!
  - Poor code & unaligned data structures do exist
- **Faults on I64 vastly slower than Alpha & impact all processes on system**
- Alignment fault summary…
  - `SDA> FLT START TRACE`
  - `SDA> FLT SHOW TRACE /SUMMARY`
  - `flt_summary.txt`
- Alignment fault trace...
  - `SDA> FLT START TRACE [/CALL]`
  - `SDA> FLT SHOW TRACE`
  - `flt_trace.txt`

Random Memory Read/Update Performance Comparison



- Single User
- 1Gb global section
- 100,000,000 Loops
- Increment a random quad

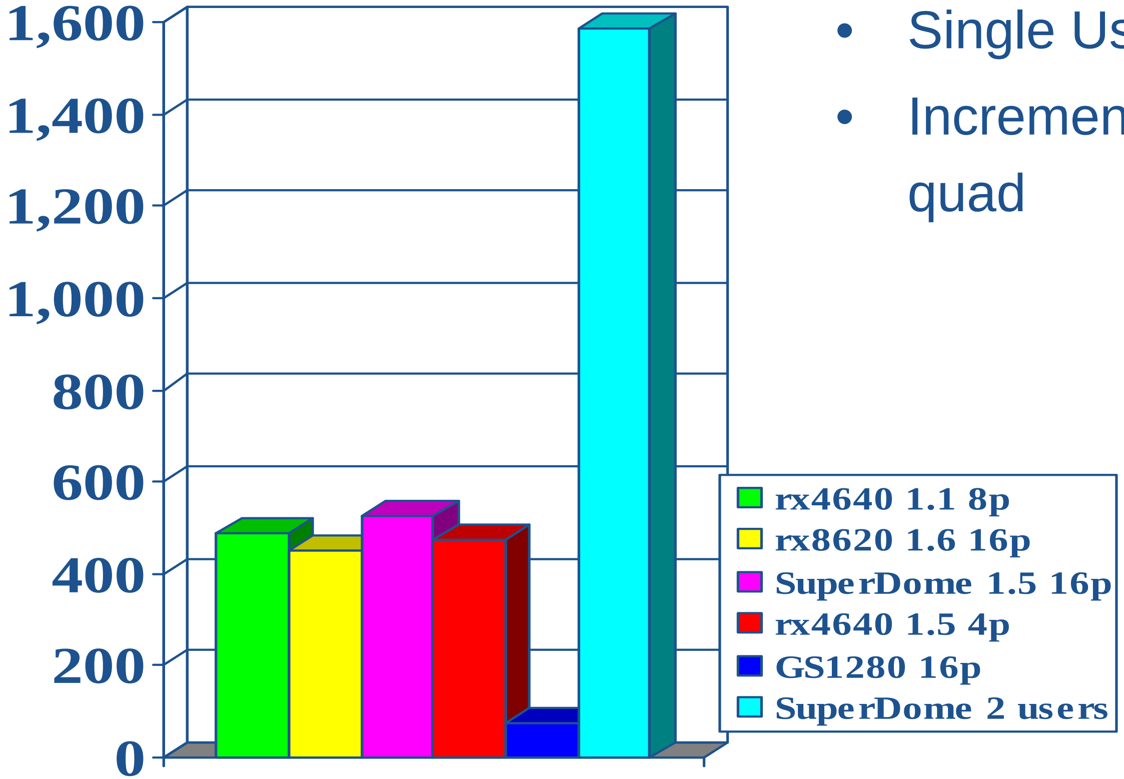
Expected Unaligned Memory Read/Update



- Single User
- Increment an expectedly unaligned quad

Unexpected Unaligned Memory Read/Update

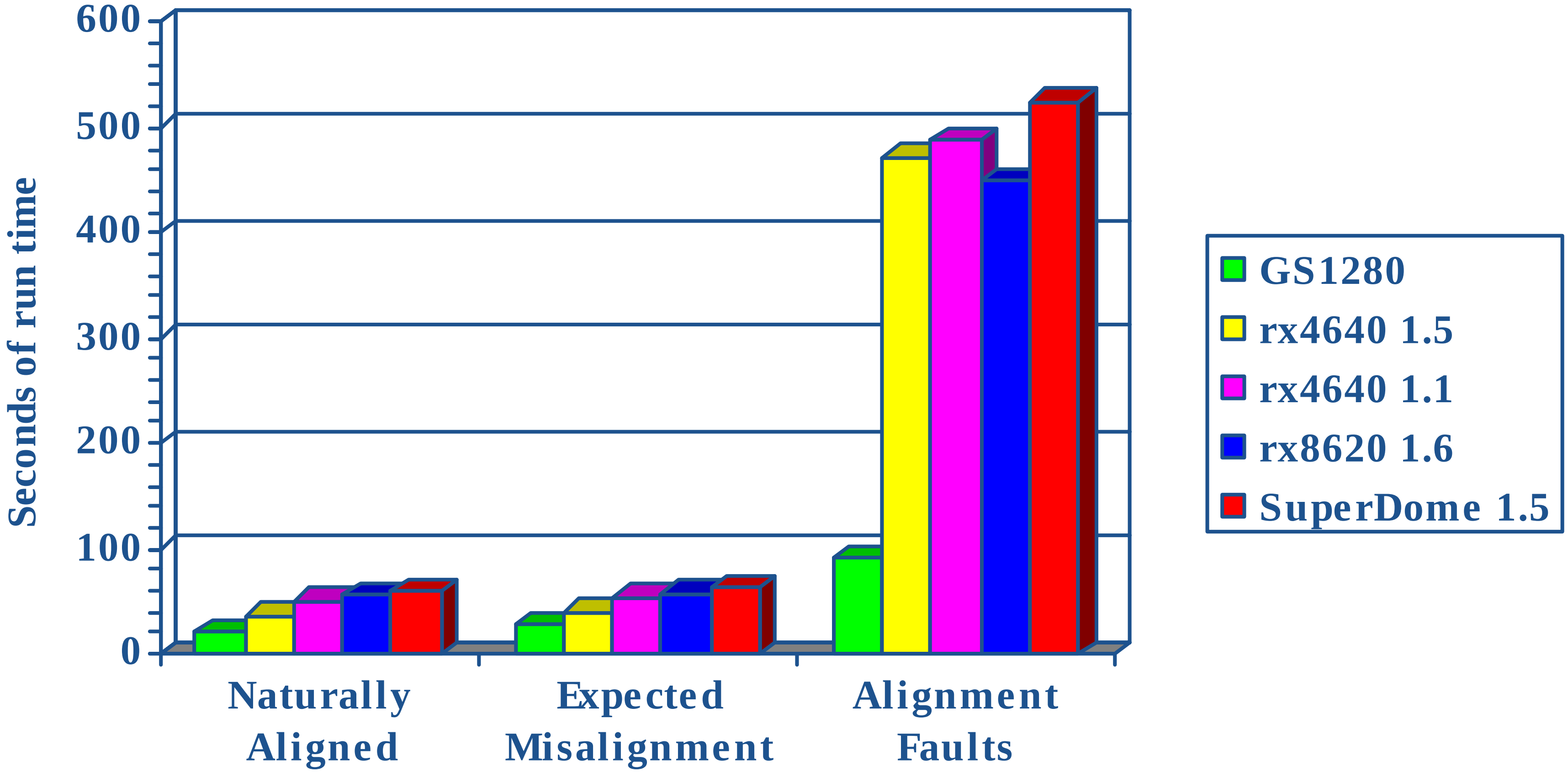
- Single User
- Increment an unexpectedly unaligned quad



Seconds - Less is better

Alignment faults on IPF are much more expensive than on Alpha & impact all processes on the system

Alignment Faults – Avoid them



RMS

Remember slide 7?

We lied….

RMS

- SYSGEN> SET RMS_SEQFILE_WBH 1
- SET FILE /STATISTICS
  - MONITOR RMS
- After Image Journaling for data protection
  - RMSJNLSNAP freeware tool

RMS

- Use larger buffers & more of ‘em
- FAB/RAB parameters:
  - `ASY, RAH, WBH, DFW, SQO`
  - `ALQ & DEQ`
  - `MBC & MBF`
  - `NOSHR, NQL, NLK`
- `SET RMS …`
  - `/SYSTEM`
  - `/BUFFER_COUNT=n`
  - `/BLOCK_COUNT=n`

RMS Hints

Watch out for NULL Keys!

FDL: NULL_KEY yes

FDL: NULL_VALUE "*char*"/value

```
$ run cidx_short
Time to add record: 0.0017268440000seconds
Time to add record: 0.2398654220000seconds
Time to add record: 0.2417297160000seconds
Time to add record: 0.0017836680000seconds
...
```

Copy to DECram/Convert from DECram back to Disk

- Sample1 DECram ANALYZE/RMS/FDL and CONVERT took

  7:59.44 vs. 12:00.01 on the HSG disks.

- Sample 2 DECram ANALYZE/RMS/FDL and CONVERT took

  7:38.12 vs. 3:54:50.56 on HSG disks!

More RMS Hints

- Use FDL to create "shell" files

Tests using HSG mirrorset.

```
$ @frag_test
```

Elapsed time is 40.31 seconds, with 10787 direct I/Os.

```
$ show status
  Status on   2-JUN-2003 11:14:11.22     Elapsed CPU :   0 00:00:00.91
  Buff. I/O :      2012     Cur. ws. :    3632     Open files :          1
  Dir. I/O :        630     Phys. Mem. :  1472     Page Faults :      4253
$ run frag
$ show status
  Status on   2-JUN-2003 11:14:51.53     Elapsed CPU :   0 00:00:02.82
  Buff. I/O :      4122     Cur. ws. :    3632     Open files :          1
  Dir. I/O :      11417     Phys. Mem. :  1536     Page Faults :      4318
```

Create the three shell files.

```
$ create/fdl=nofrag.fdl file1.dat
$ create/fdl=nofrag.fdl file2.dat
```

Elapsed time is now 3.99 seconds, with 4697 direct I/Os.

```
$ show status
  Status on   2-JUN-2003 11:37:20.85     Elapsed CPU :   0 00:00:10.70
  Buff. I/O :     12437     Cur. ws. :    3632     Open files :          1
  Dir. I/O :      49407     Phys. Mem. :  1584     Page Faults :      9361
$ run frag
$ show status
  Status on   2-JUN-2003 11:37:24.84     Elapsed CPU :   0 00:00:11.45
  Buff. I/O :     12465     Cur. ws. :    3632     Open files :          1
  Dir. I/O :      54104     Phys. Mem. :  1584     Page Faults :      9421
$
```

System Management Tips

"Experience is that marvelous thing that enables you to recognize a mistake when you make it again."

- Franklin P. Jones

IO vs CPU

- Advertised:
  - “OpteronX @ 2GHz”
  - “64-bit PCI-X @33Mhz”

- I/O performance is combination of I/O bus type (PCI, PCI-X, etc.), bus speed, bus data path and/or command width, etc.
- Many times perception that system is "running slow" is more function of I/O contention than CPU overload

EVA/XP Storage

- Initialize disks with cluster size multiple of 4
  - Brian Allison suggests 32 is good value
- Perform sequential write I/O on RAID5 groups…
  - Multiple of 4 block transfers
  - Starting on multiple of 4 block VBN
  - COPY/BLOCK_SIZE (V8.2)
  - Avoid excessive async sequential access I/O queues
    - Throttle your IO load

XP storage

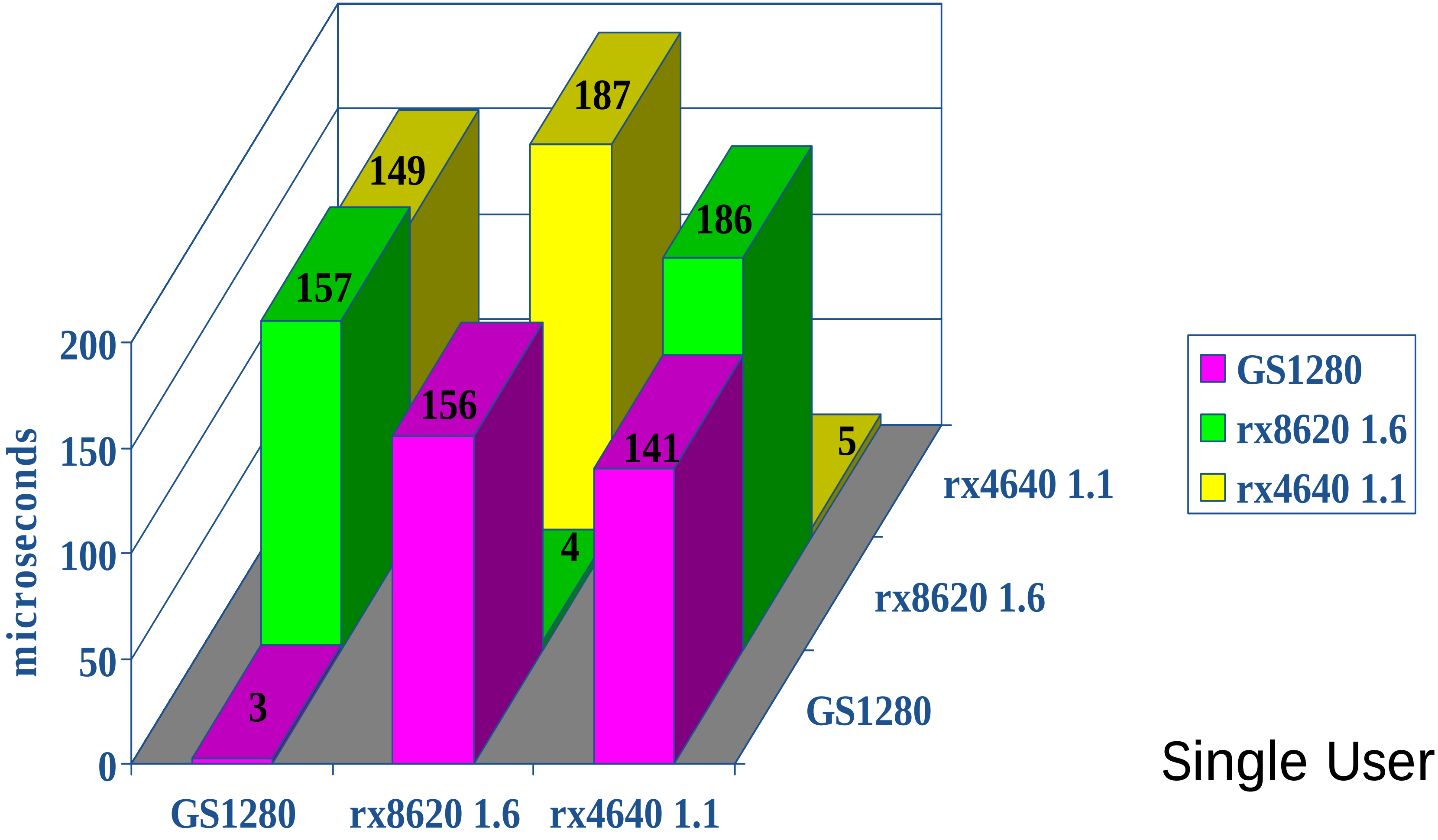
- Best if 8 I/Os per LUN are presented by host
- OpenVMS methods that can help
  - BACKUP
    - Lower values for DIOLM and PQL_MDIOLM
    - Redesigned to work with modern controllers
      - VMS732_BACKUP_V0600 (/IO_LOAD)
  - WWID throttle IO descriptor to limit the total number of I/Os per FC port
    - V7.3-2 FIBRE_SCSI-V0400 and later
    - **`SDA> FC SET WTID /WWID=target_wwid /CAP=cap_value`**
    - **`V8.3 MC SANCP`**

MSCP Disk Serving

- Alpha & I64 MSCP server does not do dynamic balancing
  - **`SET PREFERRED /HOST=<node>/FORCE <dev>`**
- **`MSCP_CREDITS >=`** 64 for busy/big servers
- **`MSCP_BUFFER >=`** 2048
  - **`127 * MSCP_CREDITS`** when using host-based shadowing

- V8.3 - PE data compression

Cluster Lock-Request Latencies



Single User

The Tech Commandments

- *Thou shalt backup, backup, BACKUP!*
- *Thou shalt not make thy password be "password".*
- *Thou shalt not adopt early or install thy version 1.0.*
- *Thou shalt not steal thy neighbor's bandwidth.*
- *Thou shalt not covet thy neighbor's toys. Instead, buy a newer model.*
- *Thou shalt not open unknown email attachments nor reply to SPAM.*
- *Thou shalt use a firewall.*
- *Remember the Slackith days. Six days thou shalt slack and do all thy surfing.*
- *Don't be Evil.*
- *Thou shalt not curse at thy computer when thy problem lies with its user.*

QUANTUM

- SYSGEN parameter
- Maximum processor time before passing control to another process
  - Units - 10 Ms
- Prior to V8.3 default value is set to 20
  - This means only 5 processes may be scheduled in a second
- Consider lowering the value to 5
  - Decrease throughput & Improve response time
  - Schedule up to 20 processes per second
  - More adequate value for modern (fast) processors

TCP/IP & DECnet

- TCP/IP V5.4 or later
  - Scaleable Kernel (logical name **`TCPIP$STARTUP_CPU_IMAGES`**)
  - Default as of TCPIP V5.5

- Increase default buffer size → reduce BIO
  - **`sysconfig -r inet tcp_mssdflt=1500`**

- **`SET RMS /SYSTEM /NETWORK = 127`**

Fibre Channel & Fastpath

- V8.3
  - Removal of IOLOCK8 spinlock usage for fibre channel drivers
- Previously
  - Fastpath allows concurrency during I/O initiate
  - Distributed interrupts allows concurrency during I/O complete
  - However, ISR (interrupt service routine) takes global IOLOCK8... Yikes...
  - Workaround: assign FGx adapters to same fastpath CPU

SHOW FASTPATH

```
Ryerox> show fastpath
Fast Path preferred CPUs on RYEROX 19-APR-2006 14:29:42.81
hp AlphaServer GS1280 7/1150 with 16 CPUs

Device:                  Fastpath CPU:
EWA0                            1
EIA0                            1
EIB0                            1
EWB0                            8
FGA0                            1
FGB0                            8
PEA0                            2
PKA0                            1
PKB0                            1
PKC0                            1

OpenVMS TCP/IP is currently running on CPU 3
OpenVMS Lock Manager is currently running on CPU 4
Ryerox>
```

Virtual Terminals

- Avoid process deletion at network disconnect (PC crash?)

Add to system startup:

```
$ ! ENABLE VIRTUAL TERMINALS
$ MCR SYSMAN IO CONNECT /NOADAPT VTA0 -
    /DRIVER=SYS$LOADABLE_IMAGES:SYS$TTDRIVER
$ DEFINE/SYSTEM/EXECUTIVE TCPIP$TELNET_VTA TRUE
```

POOL

- **NPAG_GENTLE=NPAG_AGGRESSIVE=100** to disable pool reclamation – Current VMS default

- Leave **NPAG_GENTLE** and **NPAG_AGGRESSIVE** out of MODPARAMS

Large Sequential Files

- Rarely read?
  - Create in directory marked `/CACHE=NOCACHE`

- Perhaps for…
  - Backup savesets, unload data, log files, .MAP files, etc

- Avoids polluting XFC cache

- SHOW MEMORY/CACHE

Global Sections

- Memory resident
  - Shared page tables
  - Granularity hints (when registered)
- P2 virtual address space
- Per-RAD sections on Wildfire

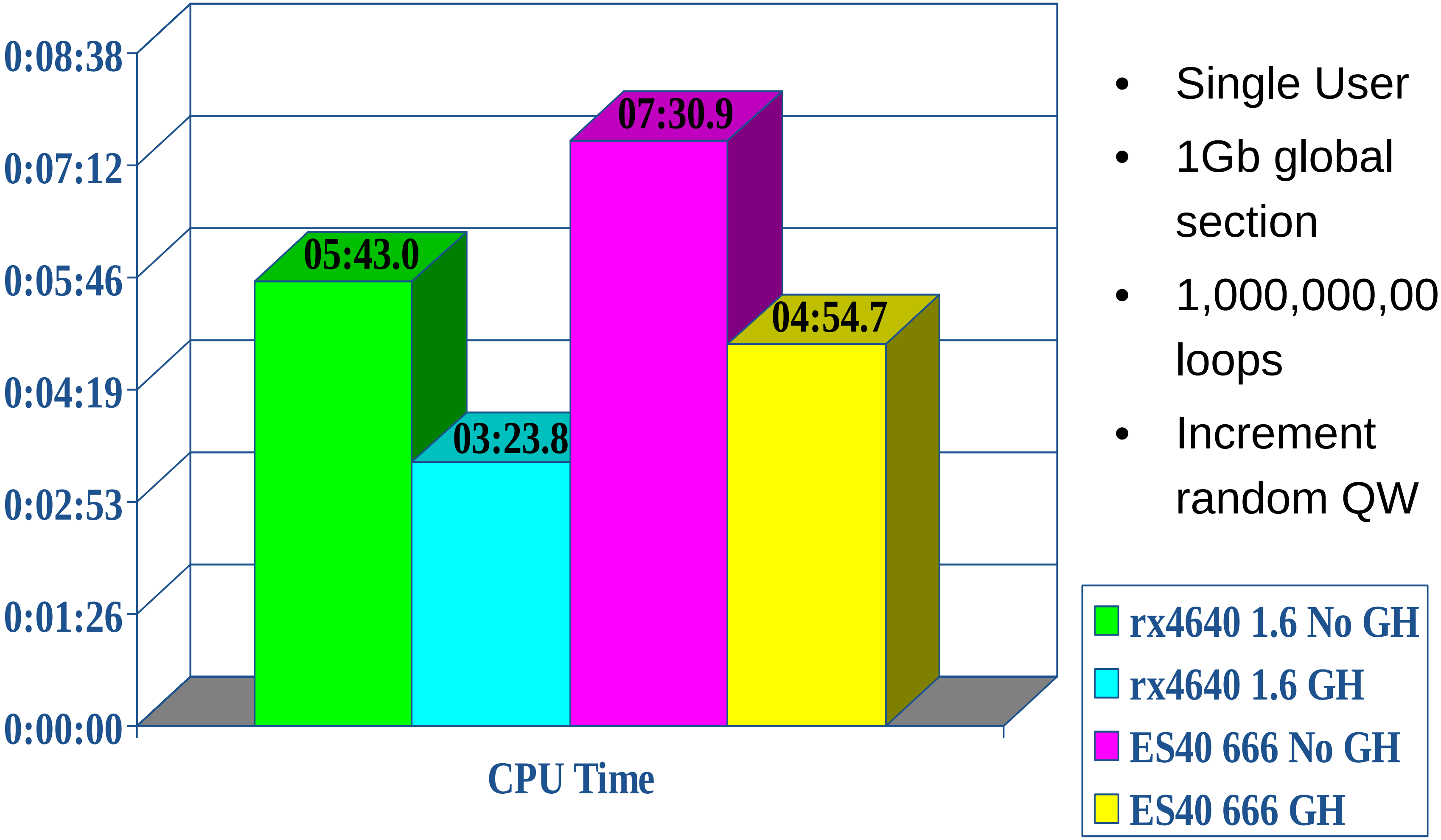
Granularity Hint Regions

- Use less CPU translation buffer entries
  - Each maps many pages; reduces TB misses
- Resident images & global sections with reserved memory
  - V8.3 maps/loads resident images into S2 space

```
Wells TNA27:> MCR SYSMAN RESERVED_MEMORY ADD NJL$SHARED_MEMORY -
              /PAGE_TABLES /SIZE=1100 /ALLOCATE

Wells TNA3:> SHOW MEMORY /RESERVE
Memory Reservations (pages):  Group    Reserved   In Use        Type
  NJL$SHARED_MEMORY          SYSGBL        138        0   Page Table
  NJL$SHARED_MEMORY          SYSGBL     131072        0    Allocated
  NJL$SHARED_MEMORY          SYSGBL       8192        0    Allocated
  NJL$SHARED_MEMORY          SYSGBL       1536        0    Allocated
  Total (1.07 GBytes reserved)               140938       0
```

Using GH Regions



- Single User
- 1Gb global section
- 1,000,000,000 loops
- Increment random QW

XFC

- It isn’t 1980 any longer…
  - Historically I/O sizes maxed at 127 blocks.
  - Today, utilities are doing I/Os up to 256 blocks at a time

- Set **VCC_MAX_IO_SIZE** to 256

- **MCR SYSMAN RESERVED_MEMORY ADD VCC$MIN_CACHE_SIZE /SIZE=xxx /ALLOCATE /NOGLOBAL /NOZERO**

DECram

- Create virtual disk from system memory
- When temp/work files can not be avoided
- Integrated with VMS V8.2
- May be shadowed with physical disk
  - Shadowing smart enough to read from memory

“No one really listens to anyone else, and if you try it for a while you'll see why.”

- *Mignon McLaughlin*

“An inventor is simply a fellow who doesn't take his education too seriously.”

- *Charles F. Kettering*

CRC

- Significant performance enhancements
  - LIB$CRC
  - CRC macro instruction

```
$ r crc2
500 buffers of size = 32768
 lib$crc latency    228.6628 msec
Total bytes processed = 16384000
 Rate =     68.3321 Mbytes/sec

$ r crc2
500 buffers of size = 32768
 lib$crc latency    152.2836 msec
Total bytes processed = 16384000
 Rate =    102.6046 Mbytes/sec
```

Disk Volumes

- SET VOLUME
  - /NOHIGHWATER
  - /EXTEND=big?
  - /CLUSTER=<multiple-of-4-or-16>
  - /LIMIT

Data Encryption

- VMS Encryption kit ships with VMS V8.2
  - – V8.3 adds additional algorithms
    - Encrypt integrated into the base O/S

- **`BACKUP /ENCRYPT`**
  - – Encryption increases CPU utilization ! Duh? You'd be surprised….

- Roll your own encryption functions

BACKUP Performance?

- Focus on *total* **restore & recovery** performance…
  - Locate media, transport media, mount it, etc
  - Zero TPS when the system is down

However…if you do care about performance…

BACKUP

- Enabling media compaction increases throughput

- **`SET RMS…`**
  - **`/BLOCK_COUNT = 127 (or 124)`**
  - **`/BUFFER_COUNT = 4 (?)`**
  - **`/EXTENDED_QUANTITY = 65535 (or 65532)`**

- Compression

Troubleshooting Tools

Analyze High MPSYNC Time

```
   sda> spl start trace/buff=5000
            .
            .
   sda> spl stop trace
   sda> spl analyze/usag=hold=1


                  OR


SYS$EXAMPLE:SPL.COM
```

Analyze High Locking Rate

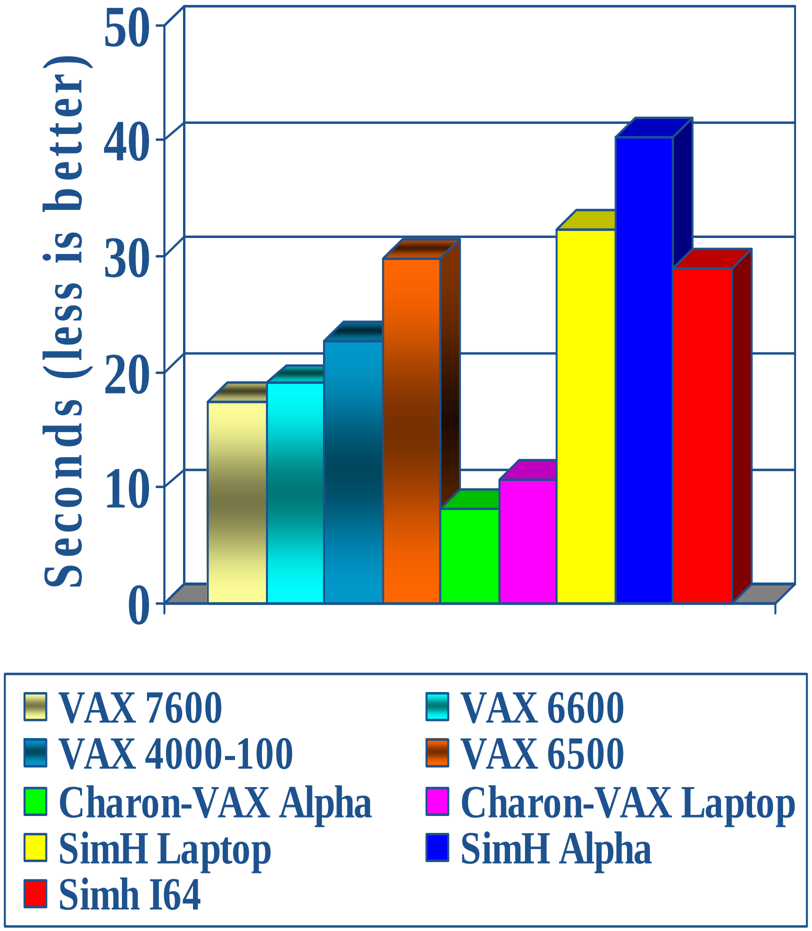
```
   sda> lck show active ! which files, volumes
or
   sda> rdb show active ! which Rdb db's
or
   sda> lck start trace ! which processes
   sda> lck start collect/process
        .
        .
   sda> lck show collect
```

Analyze High IO Rate

```
   sda> io start trace
   sda> io start collect/device
or
   sda> io start collect/process
        .
        .
   sda> io show collect /full
```

Simulators

Real & Simulated VAXen Performance



- Prime number generation
  - C program from Internet
  - Single-user
  - CPU intensive
- Charon-VAX
  - Intel Laptop 2ghz
  - …at 37,000 feet
- SimH machines
  - GS1280/1.15 32p
  - rx4640/1.5/6mb
  - Intel Laptop 2ghz

We started with applications

and will finish with programmers

Real Programmers…

- … don't write specs. Users should consider themselves lucky to get any programs at all and take what they get.
- … don't comment their code. If it was hard to write, it should be hard to read.
- … never work 9 to 5. If any real programmers are around at 9 am, it's because they were up all night.
- … don't read manuals. Reliance on a reference is a hallmark of the novice and the coward.

Credits & Special Thanks

- Norm Lastovica
- Christian Moser
- Sue Skonetski
- Greg Jordan

Questions?

BRUDEN-OSSG thanks you for attending this session.

See us at www.BRUDENOSSG.com for:

- *Performance analysis*
    - *(Performance Results Or No Expense)*
- *Porting assistance*
- *Special OPS (OpenVMS Programming Services)*