

IBM RS6000 SP

Each node has it's own copy of AIX.

Current nodes support multiple processors per node.

current nodes (03/2000) : 4 cpus, 375 MHz power-3
up to 8 GBytes memory

Nodes are connected by a switch.

Protocol	latency	bandwidth
MPI shared-mem	10 microsec.	270 MB/sec
MPI user-space	24 microsec.	133 MB/sec
MPI ip	165 microsec.	55 MB/sec
PVM	270 microsec.	33 MB/sec
Sockets	132 microsec.	77 MB/sec

TB3MX adapters, 200 MHz power-3 nodes
app-to-app send/recv with pingpong code

Distributed memory architecture.

Shared file-system : GPFS

Useful AIX Commands

AIX operating system version oslevel

list configured devices lscfg, lsdev -C

list device attributes lsattr -EH -l device_name

examples: lsattr -EH -l hdisk0
 lsattr -EH -l mem0
 lsattr -EH -l rmt0
 lsattr -EH -l L2cache0

list IBM software components lslpp -L

examples: lslpp -L | grep -i fortran
 lslpp -L | grep -i compiler
 lslpp -L | more

www.rs6000.ibm.com/resource/aix_resource/Pubs/

home-grown cpu identification qcpu (not from AIX)

```
number of cpus: 2
processor class: POWER_630
L1 Inst Cache Size in KB: 32
L1 Data Cache Size in KB: 64
L1 Data Cache Line Sz (B): 128
L2 Cache Size in KB: 4096
L2 Cache Associativity : 1
apparent freq = 199.8 MHz
```

IBM Compilers (02/2000)

Fortran : xlf version 7.1

supports Fortran 77, Fortran 90, Fortran 95
automatic parallelization for SMP (-qsmp)
directives for SMP parallelization (OpenMP)
installed in : /usr/lpp/xlf/
links in : /usr/bin
invoke with : xlf, f77, xlf90, f90, xlf95, f95
 xlf_r, xlf90_r, xlf95_r
configuration file : /etc/xlf.cfg

C : C for AIX Version 5.0

supports C only
automatic parallelization for SMP (-qsmp)
directives for SMP parallelization (OpenMP)
installed in : /usr/vac/
invoke with : xlc, xlc_r, cc, cc_r
configuration file : /etc/vac.cfg

C and C++ : IBM C and C++ Compilers, Version 3.6.6

supports C and C++
no automatic or directive-based SMP parallelization
installed in : /usr/ibmcxx/
invoke with : xlc, xlc_r, cc, cc_r, xlC, xlC_r
configuration file : /etc/ibmcxx.cfg

Fixdist

Use fixdist to get AIX updates from IBM AIX service.

Can get fixdist by anonymous ftp:

```
ftp service.boulder.ibm.com
cd /aix/tools/fixdist
binary get : fd.tar.Z
```

Install as root : put fd.tar in the root directory (/)
tar xvf fd.tar

Use fixdist as a normal user (not root) by typing : fixdist

Select PTF view or APAR view, point and click to get fixes

Includes automatic transmission of dependent filesets.

IBM Parallel Environment

IBM Software: Parallel Environment, LoadLeveler

www.rs6000.ibm.com/resource/aix_resource/sp_books/

LoadLeveler : installed in /usr/lpp/LoadL

add to PATH : /usr/lpp/LoadL/full/bin

commands : llstatus, llsubmit, llq, llcancel, llclass, llctl

examples: llstatus (standard job info)
llstatus -H (for extended help)
llstatus -f %n %cpu %m (system info)
llstatus -l (detailed, long listing)

complete documentation : /usr/lpp/LoadL/html/index.html

Parallel Environment : installed in /usr/lpp/ppe.poe

/usr/lpp/ppe.poe	mpxfl, mpcc, pdbx, mpi.h, ...
/usr/lpp/ppe.pedocs	complete html documentation
/usr/lpp/ppe.pedb	X-windows parallel debugger
/usr/lpp/ppe.vt	MPI trace visualization tool
/usr/lpp/ppe.xprofiler	profile analyzer

Message-passing libraries: MPI, MPL, LAPI

Language support: Fortran, C, C++

mpxlf, mpxlf_r, mpcc, mpcc_r, mpCC, mpCC_r

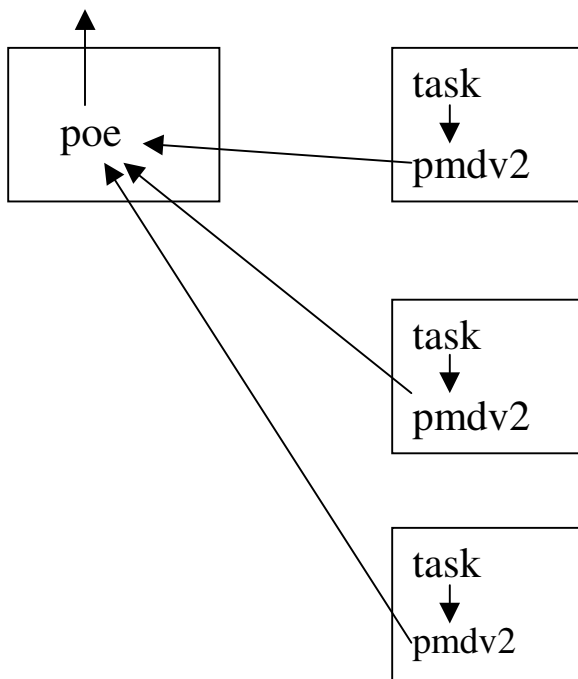
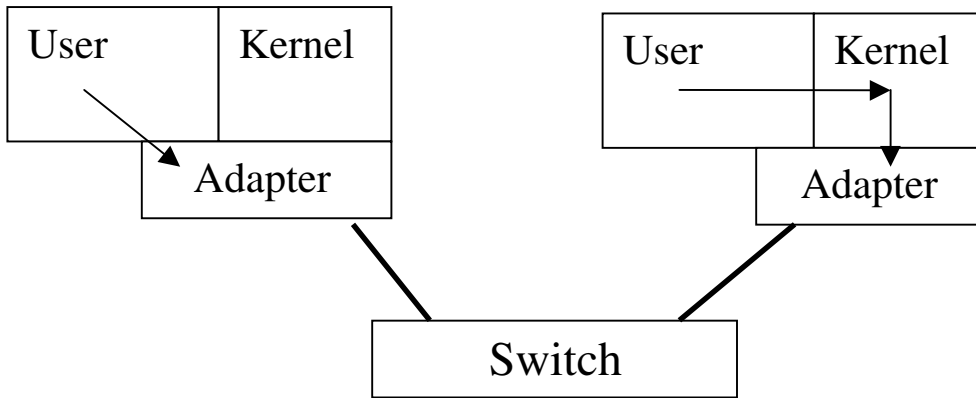
Interactive job submission : poe your.exe -procs 128

Batch job submission : llsubmit your.cmd

IBM SP Communication

User Space

IP Communication



Sample LoadLeveler Command File

```
#!/bin/ksh
#@ output = std.out
#@ error = std.err
#@ notification = never
#@ environment = MP_LABELIO=yes; \
                 MP_INFOLEVEL=0
#@ node = 8
#@ tasks_per_node = 4
#@ class = physics
#@ network.mpi = css0,shared,us
#@ job_type = parallel
#@ checkpoint = no
#@ queue
#-----
# from here the syntax is like any ksh script
#-----
poe pingpong
print "The job ran on nodes:"
for node in $LOADL_PROCESSOR_LIST
do
    echo $node
done
```

One can specify a pool or a class. For a list of job classes, use `llclass`. For a list of node pools, use `llstatus -l` or equivalent.

POE Environment Variables

Always check: `env | grep MP_`

MP_PROCS	8, 16, ...
MP_EUILIB	us, ip
MP_SHARED_MEMORY	yes, no
MP_WAIT_MODE	poll, yield, sleep
MP_HOSTFILE	hostfile
MP_STDOUTMODE	unordered, ordered, 0
MP_STDINMODE	all, 0
MP_LABELIO	yes, no
MP_INFOLEVEL	0, 1, ...
MP_PGMODEL	spmd, mpm
MP_CMDFILE	command file
MP_EUIDEVICE	css0, en0
MP_CSS_INTERRUPT	no, yes
MP_NODES	1, 2, ...
MP_TASKS_PER_NODE	1, 2, ...
MP_SAVEHOSTFILE	nodes_used
MP_RES	no, yes
MP_RMPOOL	0, 1, ...
MP_EAGER_LIMIT	default = 4K, max=64K
MP_BUFFER_MEM	default = 2.8 MB ip, 64 MB us (=max)

www.rs6000.ibm.com/resource/aix_resource/sp_books/

[/usr/lpp/ppe.pedocs/html/pebooks.html](http://usr/lpp/ppe.pedocs/html/pebooks.html)

Simple MPI Program

```
#include <stdio.h>
#include <math.h>
#include <mpi.h>

int main(int argc, char * argv[])
{
    int taskid, ntasks;
    double pi;

    /*-----*/
    /* establish the parallel environment */
    /*-----*/
    MPI_Init(&argc, &argv);
    MPI_Comm_rank(MPI_COMM_WORLD, &taskid);
    MPI_Comm_size(MPI_COMM_WORLD, &ntasks);

    /*-----*/
    /* say hello from each MPI task      */
    /*-----*/
    printf("Hello from task %d.\n", taskid);

    if (taskid == 0)    pi = 4.0*atan(1.0);
    else                pi = 0.0;
    /*-----*/
    /* do a broadcast from node 0 to all */
    /*-----*/
    MPI_Bcast(&pi, 1, MPI_DOUBLE, 0, MPI_COMM_WORLD);
    printf("node %d:  pi = %.10lf\n", taskid, pi);

    MPI_Barrier(MPI_COMM_WORLD);
    MPI_Finalize();

    return(0);
}
```

Tools for the SP

1. System Focus

vmstat, netstat, iostat

monitor/top – <http://www.mesa.nl/pub/monitor>

Performance Toolbox – 3dmon

AIX trace facility

2. Application Focus

xprofiler – function and statement-level profiling

vt - MPI message-passing visualization tool

MPI Trace - measure message-passing time

debuggers - pdbx, totalview

hardware performance monitors

instrumenting your code

Basic Command-Line Tools

virtual memory statistics : vmstat

vmstat [interval [count]]

vmstat -s (summary)

```
[v07n20:/u/user] vmstat 1 5
```

kthr		memory				page				faults				cpu			
r	b	avm	fre	re	pi	po	fr	sr	cy	in	sy	cs	us	sy	id	wa	
0	1	51066	445573	0	0	0	0	0	0	201	763	343	41	3	56	0	
0	2	51066	445573	0	0	0	0	0	0	213	1209	51	0	0	99	0	
0	2	51066	445573	0	0	0	0	0	0	209	71	44	0	0	99	0	
1	2	49902	444016	0	0	0	0	0	0	222	373	76	42	4	54	0	
1	2	49902	444016	0	0	0	0	0	0	213	85	49	50	0	50	0	

r = kernel threads placed on the run queue

b = kernel threads blocked

avm = active virtual memory pages (1 page = 4KBytes)

fre = free memory pages

pi = page-ins from paging space

po = page-outs to paging space

fr = pages freed

sr = pages scanned

cy = scan cycles

in = device interrupts

sy = system calls

cs = context switches

us = user cpu utilization

sy = system cpu utilization

id = cpu idle time

wa = time waiting on I/O

Basic Command-Line Tools

Network statistics : netstat

netstat -i -f inet (lists internet interfaces)
netstat -I css0 (switch status)
netstat -I css0 interval (switch IP traffic)
netstat -D (packet counts)

```
[v01n14:/u/user] netstat -Icss0 1
  input      (css0)      output      input      (Total)      output
 packets  errs  packets  errs  colls  packets  errs  packets  errs  colls
 6254389    0  6364710    0    0  17660924    0  17474352  183221    0
      1    0      1    0    0      4    0      3    0    0
  2127    0    2134    0    0    2139    0    2147    0    0
  1041    0    1117    0    0    1055    0    1130    0    0
      1    0      1    0    0      3    0      3    0    0
```

```
[fr1n12:/u/user] netstat -Icss0
Name  Mtu  Network  Address  Ipkts  Ierrs  Opkts  Oerrs  Coll
css0* 65520 <Link>0.0.0.0.0.0 29869  0  29892  2415  0
css0* 65520 164.122.216 fr1s12 29869  0  29892  2415  0
```

*indicates that the switch is down

I/O statistics : iostat

iostat [interval [count]] (terminal, cpu, and disk activity)
iostat -d [interval [count]] (disk activity only)
iostat -t [interval [count]] (terminal and cpu activity only)

```
[v07n20:/u/user] iostat -d1
Disks:      % tm_act    Kbps    tps    Kb_read  Kb_wrtn
hdisk0      0.2         2.9     0.3   1802140  935998
hdisk1      0.0         0.5     0.0     9546    443556
hdisk0      0.0         0.0     0.0     0        0
hdisk1      0.0         0.0     0.0     0        0
hdisk0      1.0         4.0     1.0     0        4
hdisk1     22.0       1332.0   27.0     0       1332
hdisk0      0.0         0.0     0.0     0        0
hdisk1    100.0       8528.0   78.0     0       8528
hdisk0      0.0         0.0     0.0     0        0
hdisk1     85.0       7840.0   59.0     0       7840
hdisk0      0.0         0.0     0.0     0        0
hdisk1      0.0         0.0     0.0     0        0
```

Performance Toolbox

Easy real-time monitoring of system activity

- cpu utilization
- disk activity
- network activity

Recording and playback feature

Multi-processor 3d monitor : 3dmon

Monitor local or remote systems

Requires installation as a separate lpp

Commands:

- xmpeek (wakes up the xmservd daemon)
- xmperf (starts the graphical user interface)
- 3dmon (3d multi-processor monitor)

```
3dmon -i5 -w0 -ccpu -a"node1 node2 node3"&  
-i = refresh interval  
-w = weight of previous sample  
-c = configuration (cpu, disk, ip)  
-a = node list
```

can customize the 3dmon.cf configuration file

Documentation: IBM SC23-2625-04, /usr/share/man/info

AIX Trace Facility

Provides time-stamps and details for many system events.

Performance Tuning Guide (IBM SC23-2365-04)

General Programming Concepts (IBM SC23-2533-03)

AIX Performance Tuning, Frank Waters, ISBN 0133867072

Example : trace I/O events open, lseek, read, write, close

Generate a trace file:

```
trace -a -o trcfile -j 12e,154,15b,163,19c; cp source target; trcstop
-a asynchronous
-o output file name
-j list of event tags : /usr/include/sys/trckid.h
```

syntax : trace -a -o trcfile [options]; command; trcstop
a particularly useful command is : sleep 10

Produce a “readable” trace report

```
trcrpt -o trace.report -O exec=on,pid=on trcfile
```

Sample trace report:

```
trace -o trcfile -a -j 154,15b,163,19c
```

ID	NAME	PID	ELAPSED_SEC	DELTA_MSEC	SYSCALL	KERNEL	INTERRUPT
001	trace	10474	0.00000	0.0000	TRACE ON	channel	0
15B	--1-	-1	0.01938	4.9035	open fd=3	RONLY	
15B	--1-	-1	0.03131	11.9327	open fd=7	WRONLY CREAT TRUNC	
19C	--1-	-1	0.03172	0.4071	write(1,	F068CC80,2A)	
163	--1-	-1	0.03197	0.2508	read(3,	200007C8,1000)	
163	aixterm	27952	0.03200	0.0321	read(5,	20016A18,1000)	
19C	aixterm	27952	0.03259	0.0746	write(4,	20020AF8,E4)	
19C	--1-	-1	0.04901	16.4125	write(7,	200007C8,1000)	
163	--1-	-1	0.06829	19.2812	read(3,	200007C8,1000)	
19C	--1-	-1	0.07972	11.4312	write(7,	200007C8,1000)	
163	i4lmd	13674	0.08463	4.9084	read(12,	2FF220F8,4)	

AIX Trace Facility

Example: thread management

get event tags from /usr/include/sys/trchkid.h

thread_create, thread_terminate = 465,467

```
trace -a -o trcfile -j 465,467; mmult; trcstop  
trcrpt -o trace.report -O exe=on,pid=on trcfile
```

```
trace -a -o trcfile -j 465,467
```

ID	NAME	PID	ELAPSED_SEC	DELTA_MSEC	SYSCALL	KERNEL	INTERRUPT
001	trace	28696	0.000000000	0.000000	TRACE ON	channel	0
465	-32374-	-1	0.060646077	60.646077	thread_create:		
465	-32374-	-1	0.061048967	0.402890	thread_create:		
465	-32374-	-1	0.061267448	0.218481	thread_create:		
467	--1-	32374	1.209981428	1148.713980	thread_terminate:		
467	--1-	32374	1.214976252	4.994824	thread_terminate:		
467	--1-	32374	1.215281576	0.305324	thread_terminate:		
002	--1-	-1	1.267738409	52.456833	TRACE OFF	channel	0

Example : using trace inside a program

```
#include <sys/trcctl.h>
```

```
#include <trchkid.h>
```

```
...
```

```
trcstart("-ad -o trcfile -j 465,467");
```

```
...
```

```
trcon(0);
```

```
do_work();
```

```
trcstop(0);
```

macros for user-defined events: TRCHKL0T(HKWD_USER1);

macros defined in <sys/trcmacros.h>

Time Sampled Profiling - Xprofiler

1. Compile and link with “-g -pg” flags and optimization.
2. Run the code, a gmon.out file is produced at the end.
3. Analyze the gmon.out file with gprof or xprofiler.

```
syntax: gprof > gprof.report  
        xprofiler your.exe gmon.out
```

4. Important factors

On AIX, the time-sampling interval = 0.01 sec.

Must sample a realistic problem.

PE/MPI codes generate N gmon.out files.

PVM codes require a special hostfile (specify wd).

A notation is made for every function call = overhead.

Sample CM Hostfile for Profiling

```
#host configuration for v08 nodes
#
#executable path for all hosts
*
ep=$PVM_ROOT/bin/$PVM_ARCH:$PVM_ROOT/lib/$PVM_ARCH:
$CM_AHOME/bin:$CM_LAHOME/es:$PVM_ROOT/lib
v08l01 wd=/cfs/profile/01
v08l02 wd=/cfs/profile/02
v08l03 wd=/cfs/profile/03
v08l04 wd=/cfs/profile/04
v08l05 wd=/cfs/profile/05
v08l06 wd=/cfs/profile/06
v08l07 wd=/cfs/profile/07
v08l08 wd=/cfs/profile/08
v08l09 wd=/cfs/profile/09
v08l10 wd=/cfs/profile/10
v08l11 wd=/cfs/profile/11
v08l12 wd=/cfs/profile/12
v08l13 wd=/cfs/profile/13
v08l14 wd=/cfs/profile/14
v08l15 wd=/cfs/profile/15
v08l16 wd=/cfs/profile/16
```

Alternative : add a `chdir()` statement to the procedure code.

Xprofiler Tips

Simplest when gmon.out, binary, and source are in one dir.

Libraries must match across systems.

Select "Report" for the flat profile, click on a routine.

Select "Code Display" for a source-code view.

Use "File ; Set File Search Paths" to set source directory.

Graphical Display:

width of a bar \propto time including called routines

height of a bar \propto time excluding called routines

select : Filter ; Uncluster Functions
then Filter ; Hide All Library Calls

select : Filter ; Filter by CPU Time, or Filter by Call Count

If xprofiler fails with "bad font" error message:

- (1) edit /usr/lib/X11/app-defaults/Xprofiler
replace *narc*font: -ibm-block-...
with *narc*font: fixed
- (2) xrdb -load /usr/lib/X11/app-defaults/Xprofiler

MPI Message-Passing Visualization Tool = VT

VT provides:

message-passing trace visualization (qualitative)

real-time monitoring of system activity (limited)

Generate message-passing trace with “poe”

```
poe your.exe -procs 4 -tlevel 3
```

Specify trace level with `-tlevel` flag or `MP_TRACELEVEL`

EVENT	TRACE LEVEL
application markers	1,2,3,9
kernel statistics	2,9
message-passing	3,9

Visualize the trace with the command : vt

Control trace generation within the code:

```
VT_trc_start_c(3);  
do_work();  
VT_trc_stop_c();
```

Documentation: IBM Parallel Environment SC28-1980-02

http://www.rs6000.ibm.com/resource/aix_resource/sp_books

IBM Internal MPI Trace Library

Quantitative information on MPI message-passing calls.

Not a product.

C/C++ : link with trace_mpi.o

Fortran : link with trace_mpif.o

export TRACE_ALL_PROCS=1

Run the application – must call MPI_Finalize().

output: 1 trace file for each MPI task

MPI message-passing summary for trace_file.80

MPI Function	#calls	Avg Bytes	Time (sec)
MPI_Allreduce:	9355	8.0	3.596
MPI_Barrier:	3	0.0	0.017
MPI_Bcast:	66	5.8	0.013
MPI_Scatter:	31	1008.0	0.088
MPI_Comm_rank:	1	0.0	0.000
MPI_Comm_size:	1	0.0	0.000
MPI_Isend:	43023	2003.7	0.893
MPI_Recv:	43023	2003.7	7.481
MPI_Wait:	43023	2003.7	3.739

Total Communication Information: WALL=15.82, CPU=15.53, MBYTES=258.72
The total amount of wall time = 26.23

xldb Debugger

IBM's X-windows debugger supports Fortran, C, C++

Compile and link with -g, preferably no optimization.

Syntax: `xldb your.exe`

`xldb -co your.exe` to examine a core file

`xldb -I /path1 -I /path2 ... your.exe` - sets source path

xldb is the debugger chosen for PVM/Cube Manager jobs

Hints: `xldb -h > xldb.help` for a detailed help file

or start xldb, and select "Help"

each window is like vi ; type `"/variable_name"`

to search for a variable

can compile with `-g -qfullpath` to provide the full path to the source file (don't need to set a search path)

Totalview Debugger

Multi-platform X-windows debugger from Etnus Inc.

Information on the web : www.etnus.com

Serial or parallel applications, single- or multi-threaded

Point-and-click with the mouse

left button : selects objects (breakpoints, etc.)

middle button : pop-up menu

right button : dives into objects (variables, functions, ...)

Lots of features, easy to use.

```
set LM_LICENSE_FILE, add totalview bin directory to PATH
totalview your.exe -a your_args
```

Supports IBM parallel environment:

```
compile and link with -g, preferably no optimization
```

```
set MP_XXX environment variables for IP communication
```

```
export DISPLAY=your_display:0
```

```
totalview poe -a your.exe [your command arguments]
```

Supports debugging of PVM applications.

Parallel Debugger pdbx - for MPI

character-mode debugger - dbx on the nodes

same env variables and arguments as “poe”

default mode : all tasks execute the same commands

compile and link with -g (preferably no optimization)

```
pdbx your.exe -procs 4 [for 4 MPI tasks]
```

locate segmentation faults:

start the pdbx debugger

continue execution by typing : cont [enter]

after the program halts, type: where [enter]

set breakpoints, examine the value of variables

from the pdbx prompt, type : stop in routine [enter]

c [enter] (continue)

p x, y, z [enter] (print)

or type : stop at line_number [enter]

c [enter] (continue)

p x, y, z [enter] (print)

Parallel Debugger pdbx

identify message-passing deadlock conditions

from the pdbx prompt, type : cont [enter]
after the code hangs, type : control-C
from pdbx subset, type : halt [enter]
from pdbx prompt, type : where [enter]

pdbx supports task-groups and task-specific commands

from the pdbx prompt : on 1 [enter]
on all [enter]
group add even 0 2 [enter]
on even [enter]

attach to all processes : ps -ef | grep poe -> get PID of poe
pdbx -a PID

examples of useful commands:

c	continue execution (not run)
p x	print the value of variable x
on 0 1 210,220	task 0 lists program statements 210-220
stop in routine	sets a breakpoint in routine
stop at 57	sets a breakpoint at line #57
status	lists breakpoints
delete 1	deletes breakpoint #1
return	continues to the next return
where	tells you where you are - most useful!

documentation : </usr/lpp/ppe.pedocs/html/pebooks.html>

www.rs6000.ibm.com/resource/aix_resource/sp_books/

Hardware Performance Monitors

All current processors have registers that count:

Instructions

Cycles

Fixed-unit and floating-point-unit ops

Cache and TLB misses

...

Performance Monitor toolkit - for AIX 4.3.4 (target Q3 2000)

Requires processor-specific kernel extension

Performance API project :

<http://icl.cs.utk.edu/projects/papi/>

IBM internal tools

rs3mon (IBM Montpellier)

HPM library (IBM Yorktown)

Hardware Performance Monitor - Examples

```
[redtail:/home/hpm] mmult  
real*8 matrix multiply ...
```

hpm counter report:

Cycles	=	322227630
FPU0 ops	=	277097561
FPU1 ops	=	276298228
Loads	=	282462125
Stores	=	7105053
FXU0 ops	=	2706392
FXU1 ops	=	1293645
FXU2 ops	=	543444

hpm measured for elapsed-time = 1.656e+00 sec.

n =480, MFlops = 686.7

```
[redtail:/home/hpm] export HPM_EVENT_SET=2  
[redtail:/home/hpm] mmult  
real*8 matrix multiply ...
```

hpm counter report:

Cycles	=	319656204
Instructions	=	882998698
Loads	=	282468430
L1 load misses	=	3932238
Stores	=	7109095
L1 store misses	=	243831
TLB misses	=	166773

hpm measured for elapsed-time = 1.636e+00 sec.

n = 480, MFlops = 688.8

Hardware Performance Monitor - Example

```
[redtail:/home/hpm] slow  
real*8 matrix multiply ...
```

```
hpm counter report:
```

```
Cycles           = 18947744994  
FPU0 ops         = 550581646  
FPU1 ops         = 8199048  
Loads            = 1107074930  
Stores           = 1152379  
FXU0 ops         = 4618818  
FXU1 ops         = 2339300  
FXU2 ops         = 2305621
```

```
hpm measured for elapsed-time = 9.584e+01 sec.
```

```
n = 480    MFlops = 11.6
```

```
[redtail:/home/hpm] export HPM_EVENT_SET=2  
[redtail:/home/hpm] slow  
real*8 matrix multiply ...
```

```
hpm counter report:
```

```
Cycles           = 19662555862  
Instructions     = 1820792567  
Loads           = 1107425018  
L1 load misses  = 572075691  
Stores          = 1153028  
L1 store misses = 935768  
TLB misses      = 523133587
```

```
hpm measured for elapsed-time = 9.889e+01 sec.
```

```
n = 480    MFlops = 11.2
```

Instrumenting your code with timers

getrusage() - fills in a structure with lots of information

part of OSF X/Open specification

user time, system time, memory footprint, context switches

```
#include <sys/resource.h>
struct rusage ru;
double cputime;
int memused;
getrusage(RUSAGE_SELF, &ru);
cputime = ru.ru_utime.tv_sec + ru.ru_utime.tv_usec*1.0e-6;
memused = ru.ru_maxrss;
```

gettimeofday() - elapsed time, OSF X/Open specification

```
#include <sys/time.h>
struct timeval tv;
struct timezone tz;
double elapsed;
gettimeofday(&tv, &tz);
elapsed = (double)tv.tv_sec + ((double)tv.tv_usec)*1.0e-6
```

rtc() - low overhead elapsed-time timer, Fortran library (-lxf90)

```
real*8 time1, time2, rtc
time1 = rtc()
...
time2 = rtc()
elapsed_time = time2 - time1
```

MPI_Wtime - MPI wall-clock timer

```
#include <mpi.h>
double time1;
time1 = MPI_Wtime();
...
```