HP XC cluster
HP-MPI

workshop at CSC
11-15 june 2007

Patrick DEMICHEL patrick.demichel@hp.com

Architect HPC EMEA

Proprietary Reminder

The information contained in this presentation is proprietary to Hewlett-Packard (HP) and is offered in confidence, subject to the terms and conditions of a Non-Disclosure Agreement.

HP makes no warranties regarding the accuracy of this information. HP does not warrant or represent that it will introduce any product to which the information relates. It is presented for evaluation by the recipient and to assist HP in defining product direction.

Agenda

- HP-MPI
- Debugging
- Oprofile
- SFS

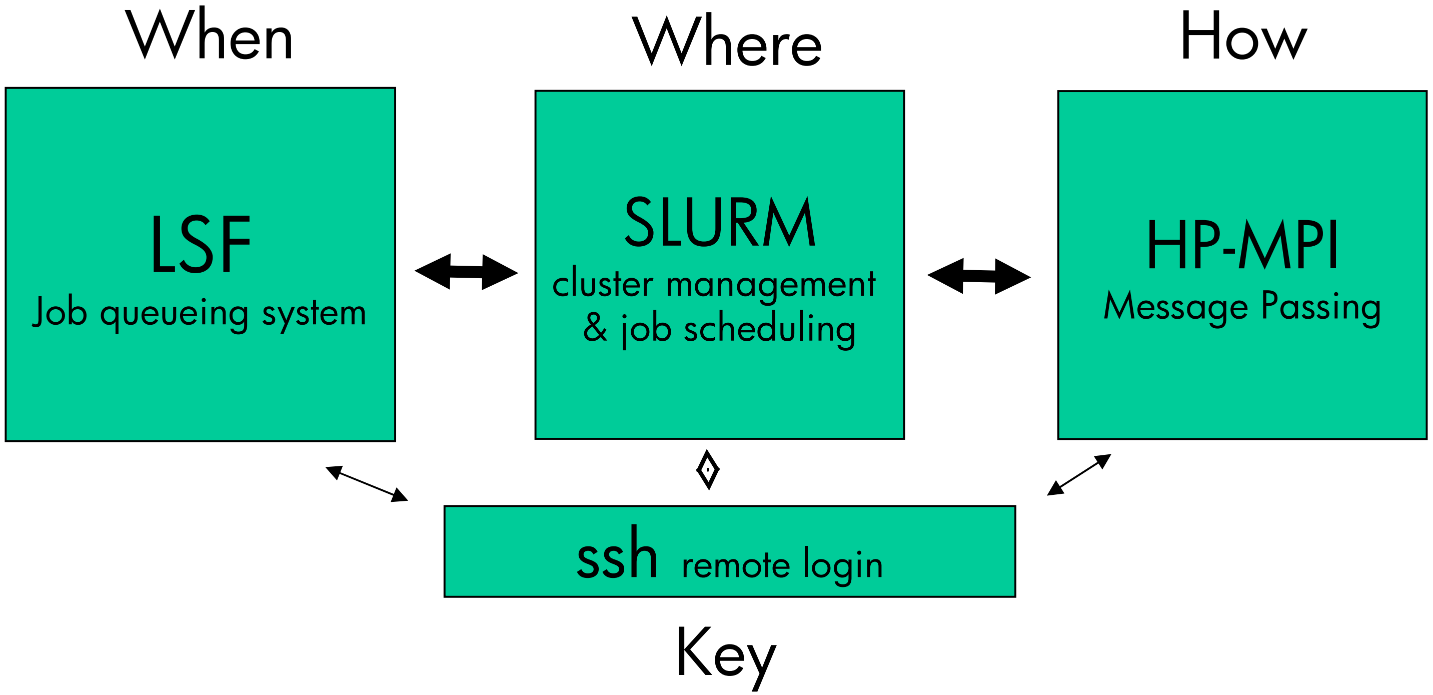
HP-MPI

June 6, 2007

HP-MPI 2.2 and XC 3.0

- Usability
  - Xc jobs, srun, lustre, ssh, 32 bit mode,
- Debuggability and Profiling
  - Message Profiling
  - Message validation Library
- Communication and Cluster Health
  - MPI Communication
  - Interconnect health check
- Scaleout
  - rank to core binding
  - Startup, message buffers, licensing
- Performance Improvements
  - InfiniBand, Ethernet

XC Job Control



LSF, SLURM, HP-MPI are tightly coupled, built to interact with a remote login program.

LSF determine WHEN the job will run LSF talks with SLURM to determine WHICH resources will be used.

SLURM - Determines WHERE the job runs. It controls things like which host each rank runs on. SLURM also starts the executables on each host as requested by HP-MPI's mpirun

HP-MPI - Determines HOW the job runs, part of the application, handles communication. Can also pinpoint the processor on which each rank runs.

SSH/rsh - The KEY that opens up remote hosts.

HP-MPI mpirun

Useful options:

- **-prot** Prints the communication protocol
- **-np *#*** - Number of processors to use
- **-h *host*** - Set host to use
- **-e *<var>[=<val>]*** - Set environment variable
- **-d** - Debug mode
- **-v** - Verbose
- **-i *file*** - Write profile of MPI functions
- **-T** - Prints user and system times for each MPI rank.
- **-srun** - Use SLURM
- **-mpi32** - Use 32-bit interconnect libraries on X86-64
- **-mpi64** - Use 64-bit interconnect libraries on X86-64 (default)
- **-f *appfile*** - Parallelism directed from instructions in appfile

SLURM srun utility

srun – SLURM utility to run parallel jobs

srun usage on XC:

- hpmpi option
    - Use as: **-srun *options exe args***
- hpmpi implied srun mode
    - Use as: **export MPI_USESRUN 1**
    - Set options by: **export MPI_SRUNOPTIONS *options***

32- and 64-bit selection

- Options have been added to indicate the bitness of the application so the proper interconnect library can be invoked.
- Use –mpi32 or –mpi64 on the mpirun command line for AMD64 and EM64T.
- Default is –mpi64.
- Mellanox only provides a 64-bit IB driver.
  - 32-bit apps are not supported for IB on AMD64 & EM64T systems.

HP-MPI Parallel Compiler Options

Useful options:

- **-mpi32** - build 32-bit

Useful environment variables:

- **setenv MPI_CC *cc*** - set C compiler
- **setenv MPI_CXX *C++*** - set C++ compiler
- **setenv MPI_F90 *f90*** - set Fortran compiler
- **setenv MPI_ROOT *dir*** - useful when MPI not installed in /opt/[hpmpi|mpi]

Problematic Compiler Options

INTEL	PGI	Description
-static	-Bstatic	Link static – does not allow HP-MPI to determine interconnect
-i8	-i8	If you compile with this, be sure to link with it. Intel and AMD math libraries do not support Integer*8.

HP-MPI Debugging

June 6, 2007

Debugging Scripts: Use hello_world Test case

```
#include <stdio.h>
#include <mpi.h>
main(int argc,char ** argv)
 {
            int         rank, size, len;
            char     name[MPI_MAX_PROCESSOR_NAME];

            MPI_Init(&argc, &argv);
            MPI_Comm_rank(MPI_COMM_WORLD, &rank);
            MPI_Comm_size(MPI_COMM_WORLD, &size);
            MPI_Get_processor_name(name, &len);
            printf ("Hello world! I'm %d of %d on %s\n", rank, size, name);
            MPI_Finalize();
            exit(0);
}
```

How to debug HP-MPI applications with a single-process debugger

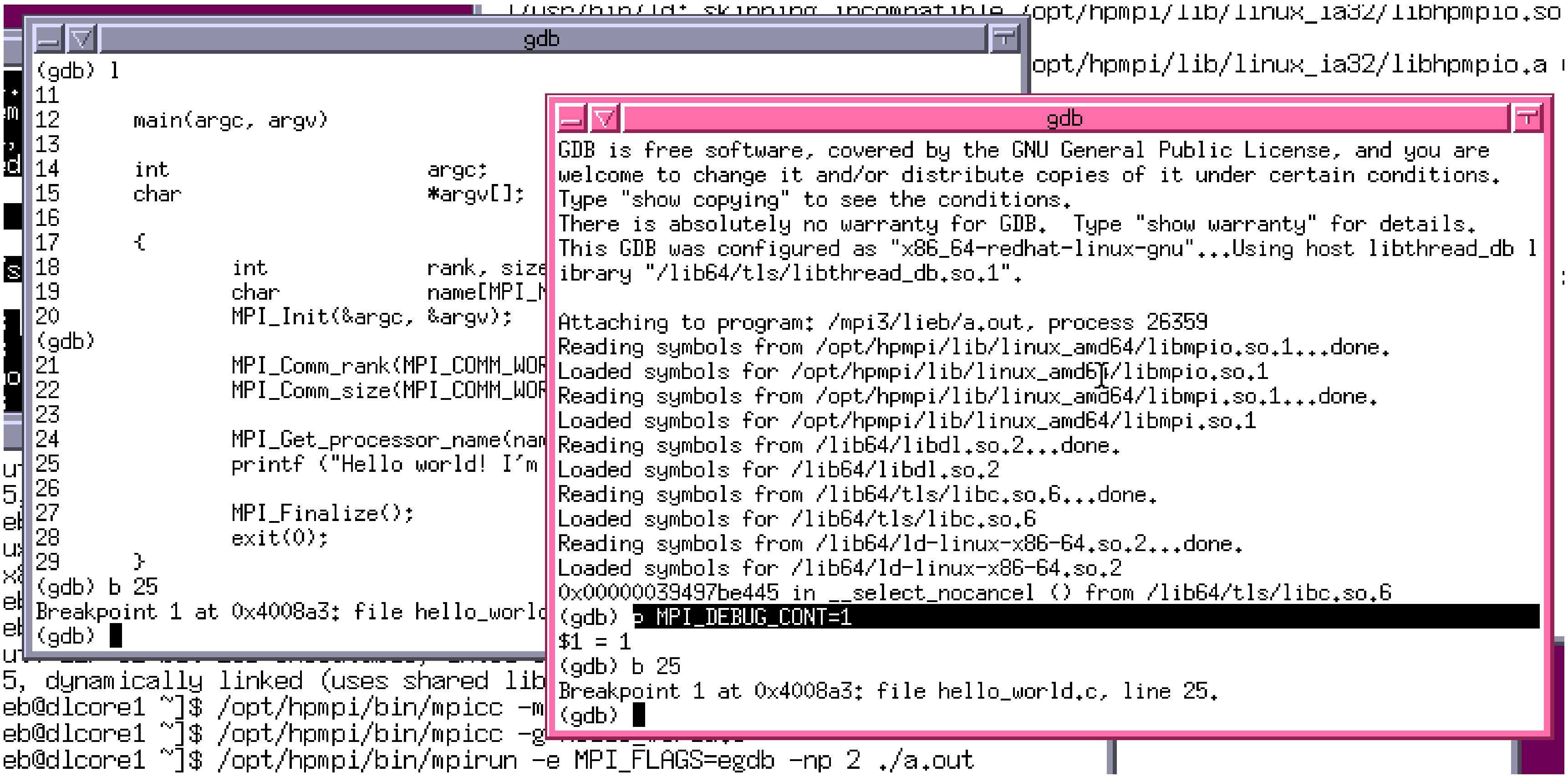
- **export MPI_DEBUG_CONT=1**
- Set the **MPI_FLAGS** environment variable to choose debugger. Values are:
  - **eadb** – Start under adb
  - **exdb** – Start under xdb
  - **edde** – Start under dde
  - **ewdb** – Start under wdb
  - **egdb** – Start under gdb
- Set **DISPLAY** to point to your console with ssh -X

Attaching Debuggers to HP-MPI Applications

- HP-MPI conceptually creates processes in MPI_Init, and each process instantiates a debugger session.
- Each debugger session in turn attaches to the process that created it.
- HP-MPI provides **MPI_DEBUG_CONT** to control the point at which debugger attachment occurs via breakpoint.
- **MPI_DEBUG_CONT** is a variable that HP-MPI uses to temporarily spin the processes awaiting the user to allow execution to proceed via debugger commands.
- By default, **MPI_DEBUG_CONT** is set to 0 and you must set it to 1 to allow the debug session to continue past this ‘spin barrier’ in MPI_Init.

Debugging HP-MPI apps cont:

-



```
/usr/bin/ld: skipping incompatible /opt/hpmpi/lib/linux_ia32/libhpmpio.so
opt/hpmpi/lib/linux_ia32/libhpmpio.a
```

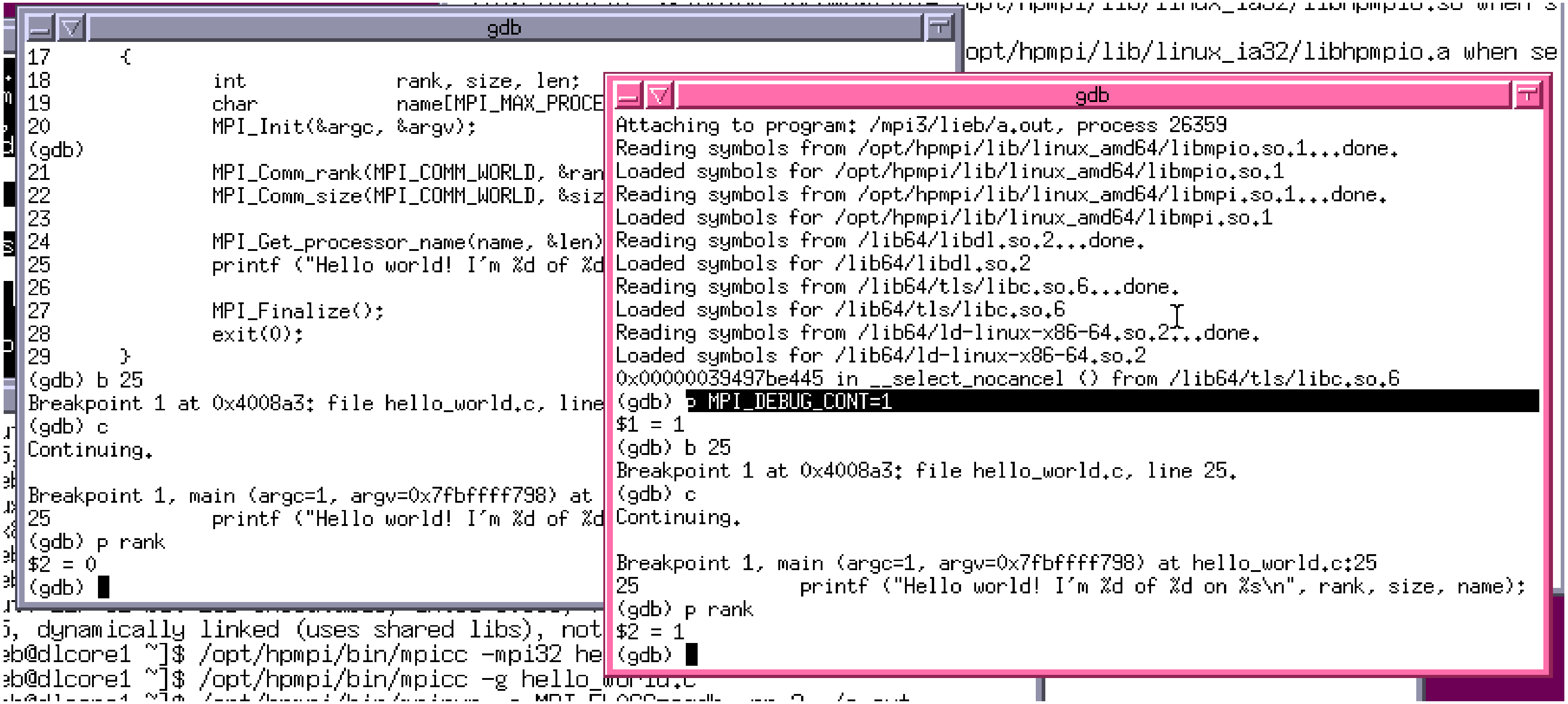
```
(gdb) l
11
12      main(argc, argv)
13
14      int                     argc;
15      char                    *argv[];
16
17      {
18              int             rank, size
19              char            name[MPI_M
20              MPI_Init(&argc, &argv);
(gdb)
21              MPI_Comm_rank(MPI_COMM_WOR
22              MPI_Comm_size(MPI_COMM_WOR
23
24              MPI_Get_processor_name(nam
25              printf ("Hello world! I'm
26
27              MPI_Finalize();
28              exit(0);
29      }
(gdb) b 25
Breakpoint 1 at 0x4008a3: file hello_world
(gdb) █
```

```
GDB is free software, covered by the GNU General Public License, and you are
welcome to change it and/or distribute copies of it under certain conditions.
Type "show copying" to see the conditions.
There is absolutely no warranty for GDB.  Type "show warranty" for details.
This GDB was configured as "x86_64-redhat-linux-gnu"...Using host libthread_db l
ibrary "/lib64/tls/libthread_db.so.1".

Attaching to program: /mpi3/lieb/a.out, process 26359
Reading symbols from /opt/hpmpi/lib/linux_amd64/libmpio.so.1...done.
Loaded symbols for /opt/hpmpi/lib/linux_amd64/libmpio.so.1
Reading symbols from /opt/hpmpi/lib/linux_amd64/libmpi.so.1...done.
Loaded symbols for /opt/hpmpi/lib/linux_amd64/libmpi.so.1
Reading symbols from /lib64/libdl.so.2...done.
Loaded symbols for /lib64/libdl.so.2
Reading symbols from /lib64/tls/libc.so.6...done.
Loaded symbols for /lib64/tls/libc.so.6
Reading symbols from /lib64/ld-linux-x86-64.so.2...done.
Loaded symbols for /lib64/ld-linux-x86-64.so.2
0x00000039497be445 in __select_nocancel () from /lib64/tls/libc.so.6
(gdb) p MPI_DEBUG_CONT=1
$1 = 1
(gdb) b 25
Breakpoint 1 at 0x4008a3: file hello_world.c, line 25.
(gdb) █
```

```
5, dynamically linked (uses shared lib
eb@dlcore1 ~]$ /opt/hpmpi/bin/mpicc -m
eb@dlcore1 ~]$ /opt/hpmpi/bin/mpicc -g
eb@dlcore1 ~]$ /opt/hpmpi/bin/mpirun -e MPI_FLAGS=egdb -np 2 ./a.out
```

Debugging HP-MPI apps cont:

-



```
opt/hpmpi/lib/linux_ia32/libhpmpio.a when se
```

```
17      {
18              int             rank, size, len;
19              char            name[MPI_MAX_PROCE
20              MPI_Init(&argc, &argv);
(gdb)
21              MPI_Comm_rank(MPI_COMM_WORLD, &ran
22              MPI_Comm_size(MPI_COMM_WORLD, &siz
23
24              MPI_Get_processor_name(name, &len)
25              printf ("Hello world! I'm %d of %d
26
27              MPI_Finalize();
28              exit(0);
29      }
(gdb) b 25
Breakpoint 1 at 0x4008a3: file hello_world.c, line
(gdb) c
Continuing.

Breakpoint 1, main (argc=1, argv=0x7fbffff798) at
25              printf ("Hello world! I'm %d of %d
(gdb) p rank
$2 = 0
(gdb)
```

```
Attaching to program: /mpi3/lieb/a.out, process 26359
Reading symbols from /opt/hpmpi/lib/linux_amd64/libmpio.so.1...done.
Loaded symbols for /opt/hpmpi/lib/linux_amd64/libmpio.so.1
Reading symbols from /opt/hpmpi/lib/linux_amd64/libmpi.so.1...done.
Loaded symbols for /opt/hpmpi/lib/linux_amd64/libmpi.so.1
Reading symbols from /lib64/libdl.so.2...done.
Loaded symbols for /lib64/libdl.so.2
Reading symbols from /lib64/tls/libc.so.6...done.
Loaded symbols for /lib64/tls/libc.so.6
Reading symbols from /lib64/ld-linux-x86-64.so.2...done.
Loaded symbols for /lib64/ld-linux-x86-64.so.2
0x00000039497be445 in __select_nocancel () from /lib64/tls/libc.so.6
(gdb) p MPI_DEBUG_CONT=1
$1 = 1
(gdb) b 25
Breakpoint 1 at 0x4008a3: file hello_world.c, line 25.
(gdb) c
Continuing.

Breakpoint 1, main (argc=1, argv=0x7fbffff798) at hello_world.c:25
25              printf ("Hello world! I'm %d of %d on %s\n", rank, size, name);
(gdb) p rank
$2 = 1
(gdb)
```

```
5, dynamically linked (uses shared libs), not
eb@dlcore1 ~]$ /opt/hpmpi/bin/mpicc -mpi32 he
eb@dlcore1 ~]$ /opt/hpmpi/bin/mpicc -g hello_world.c
```

HP-MPI Profiling

June 6, 2007

Profiling

- **Instrumentation**
  - Lightweight method for cumulative runtime statistics
  - Profiles for applications linked with standard HP-MPI library
  - Profiles for applications linked with the thread-compliant library

HP-MPI instrumentation profile:

-i <myfile>[:*opt*] - produces a rank by rank summary of where MPI spends its time and places result in file name **myfile.trace**

```
bsub –I –n4 mpirun –i myfile  -srun ./a.out
```

```
Application Summary by Rank (second):

Rank      Proc CPU Time        User Portion              System Portion
-----------------------------------------------------------------------------
 0        0.040000         0.030000( 75.00%)         0.010000( 25.00%)
 1        0.050000         0.040000( 80.00%)         0.010000( 20.00%)
 2        0.050000         0.040000( 80.00%)         0.010000( 20.00%)
 3        0.050000         0.040000( 80.00%)         0.010000( 20.00%)
```

HP-MPI instrumentation continued

- Routine Summary by Rank:

```
Rank  Routine        Statistic      Calls   Overhead(ms)   Blocking(ms)
------------------------------------------------------------------------
  0
     MPI_Bcast                        4       7.127285       0.000000
                      min                     0.033140       0.000000
                      max                     5.244017       0.000000
                      avg                     1.781821       0.000000
     MPI_Finalize                     1       0.034094       0.000000
     MPI_Init                         1 1080.793858       0.000000
     MPI_Recv                      2010       3.236055       0.000000
```

HP-MPI instrumentation continued

- Message Summary by Rank Pair:

```
SRank   DRank     Messages       (minsize,maxsize)/[bin]      Totalbytes
-------------------------------------------------------------------------
  0
          1         1005              (0, 0)            0
                    1005             [0..64]            0

          3         1005              (0, 0)            0
                    1005             [0..64]            0
```

Diagnostic Library

- Advanced run time error checking and analysis
- Message signature analysis detects type mismatches
- Object-space corruption detects attempts to write into objects
- Detects operations that causes MPI to write to a user buffer more than once

HP-MPI Diagnostic Library

- Link with –ldmpi to enable diagnostic library, or use
- ld_preload on an existing pre-linked application (shared libs)
  - This will dynamically insert diagnostic lib
    - mpirun -e LD_PRELOAD=libdmpi.so:libmpi.so -srun ./a.out
  - This will also dump message formats (could be REALLY Large)
    - mpirun -e LD_PRELOAD=libdmpi.so:libmpi.so -e MPI_DLIB_FLAGS=dump:foof -srun ./a.out
- See “MPI_DLIB_FLAGS” on page 46 of Users Guide or man mpienv for more information on controlling features.

Oprofile

June 6, 2007

OPROFILE Profiling example

- oprofile configured in XC, but not enabled
- Need to be root to enable on a node

```
# opcontrol --no-vmlinux
# opcontrol --start
Using default event: GLOBAL_POWER_EVENTS:100000:1:1:1
Using 2.6+ OProfile kernel interface.
Using log file /var/lib/oprofile/oprofiled.log
Daemon started.
Profiler running.
```

Clear out old performance data.

```
# opcontrol --reset
Signalling daemon... done
```

OPROFILE Profiling example cont.

- Run your application

```
# bsub -I -n4 -ext "SLURM[nodelist=xcg14]" ./run_linux_amd_intel 4 121 test
```

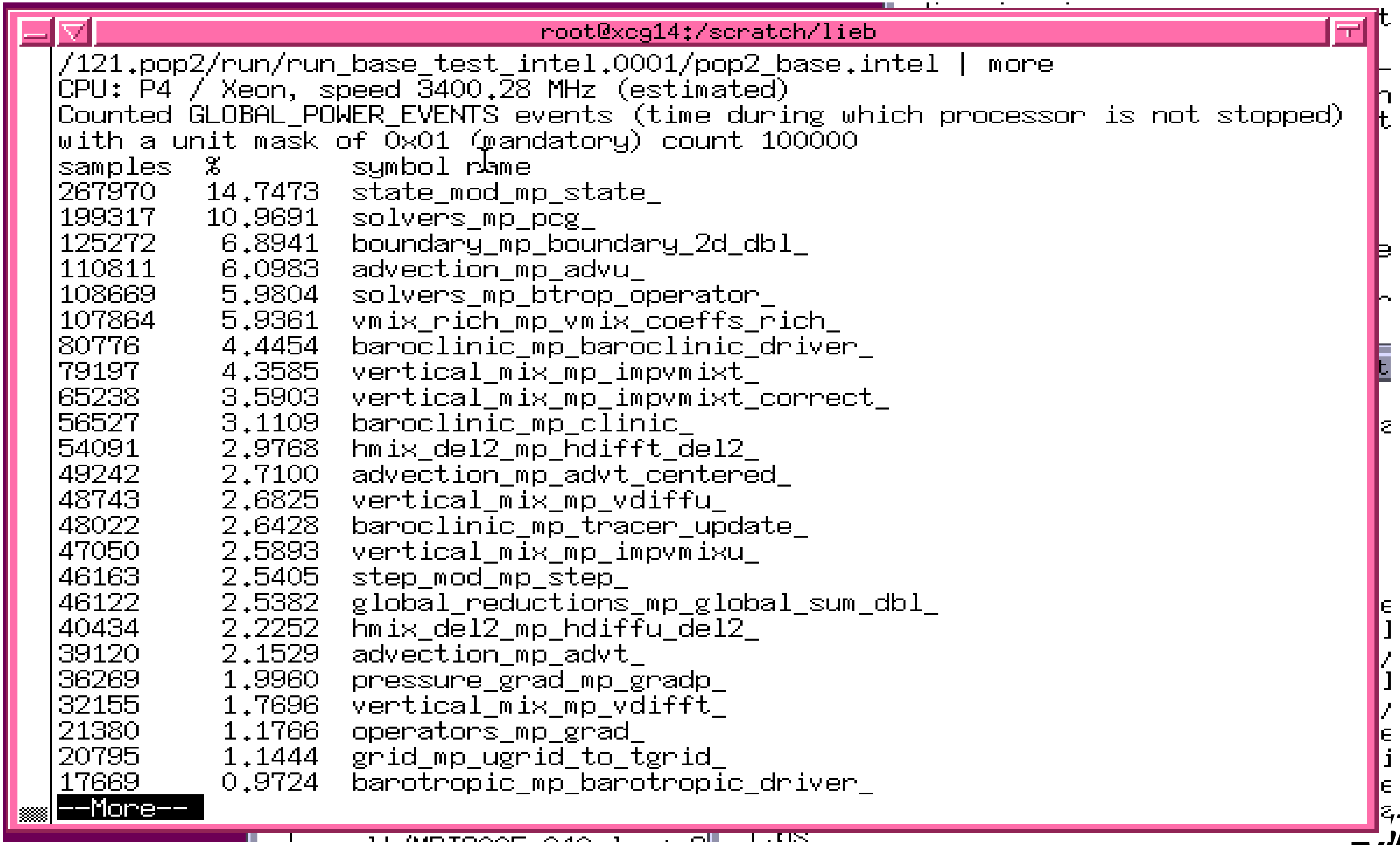
- find the name of your executable

```
# opreport --long-filenames
```

- Generate a report for that executable image

```
# opreport -l /mlibscratch/lieb/mpi2005.kit23/benchspec/MPI2005/121.pop2/run/run_base_test_intel.0001/pop2_base.intel | more
```

OPROFILE Profiling example cont.



```
/121.pop2/run/run_base_test_intel.0001/pop2_base.intel | more
CPU: P4 / Xeon, speed 3400.28 MHz (estimated)
Counted GLOBAL_POWER_EVENTS events (time during which processor is not stopped)
with a unit mask of 0x01 (mandatory) count 100000
samples  %        symbol name
267970   14.7473  state_mod_mp_state_
199317   10.9691  solvers_mp_pcg_
125272    6.8941  boundary_mp_boundary_2d_dbl_
110811    6.0983  advection_mp_advu_
108669    5.9804  solvers_mp_btrop_operator_
107864    5.9361  vmix_rich_mp_vmix_coeffs_rich_
80776     4.4454  baroclinic_mp_baroclinic_driver_
79197     4.3585  vertical_mix_mp_impvmixt_
65238     3.5903  vertical_mix_mp_impvmixt_correct_
56527     3.1109  baroclinic_mp_clinic_
54091     2.9768  hmix_del2_mp_hdifft_del2_
49242     2.7100  advection_mp_advt_centered_
48743     2.6825  vertical_mix_mp_vdiffu_
48022     2.6428  baroclinic_mp_tracer_update_
47050     2.5893  vertical_mix_mp_impvmixu_
46163     2.5405  step_mod_mp_step_
46122     2.5382  global_reductions_mp_global_sum_dbl_
40434     2.2252  hmix_del2_mp_hdiffu_del2_
39120     2.1529  advection_mp_advt_
36269     1.9960  pressure_grad_mp_gradp_
32155     1.7696  vertical_mix_mp_vdifft_
21380     1.1766  operators_mp_grad_
20795     1.1444  grid_mp_ugrid_to_tgrid_
17669     0.9724  barotropic_mp_barotropic_driver_
--More--
```

OPROFILE Profiling kernel symbols

The actual version of the rpm may change

- **The vmlinux file is contained in the kernel debug RPM:**
  - **kernel-debuginfo-2.6.9-11.4hp.XC.x86_64.rpm**
- **Kernel symbols file is installed in:**
  - **/usr/lib/debug/lib/modules/2.6.9-11.4hp.XCsmp/vmlinux**
- **opcontrol --vmlinux=\**
  - **/usr/lib/debug/lib/modules/2.6.9-11.4hp.XCsmp/vmlinux**

HP-MPI Communication

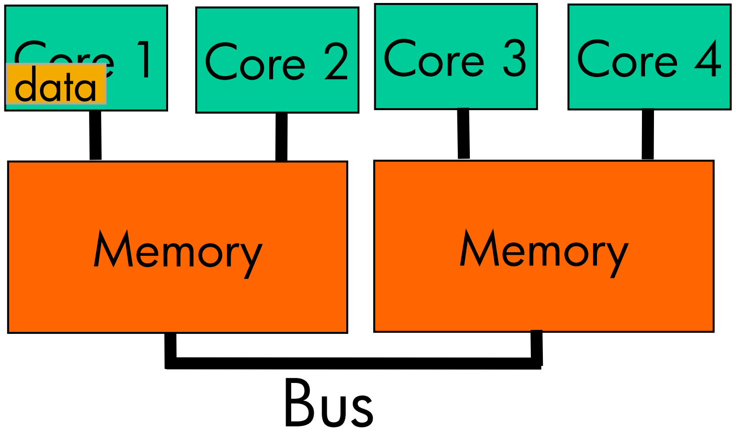
June 6, 2007

HP-MPI Communication

Movement of data depends on relative location of destination and interconnect. Paths are:

- Communication within a Node (shared memory)
- Communication from Node to Node over TCP/IP
- Communication from Node to Node over high speed interconnects InfiniBand, Quadrics, Myrinet

HP-MPI Communication within a Node



To Send data from Core 1 to Core 4:

Core 1 -> Core 1 Local Memory

Core 1 Local Memory* -> System Shared Memory**

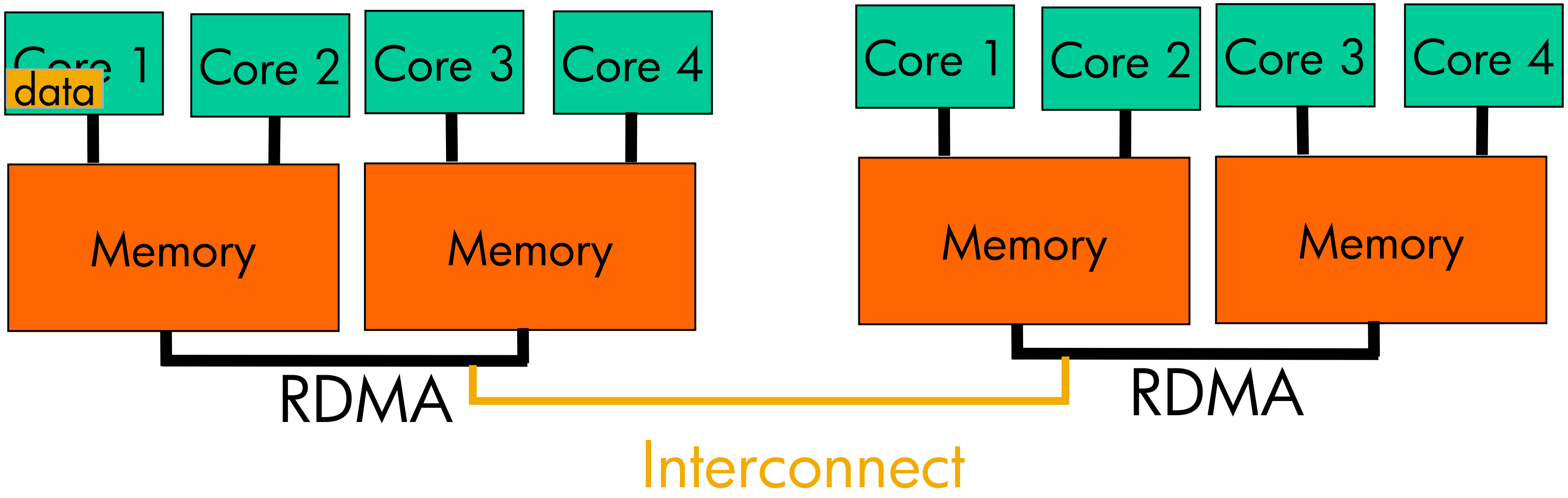
System Shared Memory -> Core 4 Local Memory

Core 4 Local Memory -> Core 4

*The operating system makes Local Memory available to a single process

**The operating system makes Shared Memory available to multiple processes

HP-MPI Communication to another Node via other Interconnects



To Send data from Core 1, Node 1 to Core 1, Node 2:

- Core 1, Node 1 -> Core 1, Node 1 Local Memory
- Core 1, Node 1 Local Memory -> Node 1 Shared Memory
- Node 1 Shared Memory -> Interconnect
- Interconnect -> Node 2 Shared Memory
- Node 2 Shared Memory -> Core 1, Node 2 Local Memory
- Core 1, Node 2 Local Memory -> Core 1, Node 2

X86-64: 32-bit versus 64-bit Interconnect Support

- Supported 64-bit interconnects:
  - TCP/IP
  - GigE
  - InfiniBand
  - Elan
  - Myrinet
- Supported 32-bit interconnects:
  - TCP/IP
  - Myrinet
  - InfiniBand (but not 32 bit mode on 64 bit architectures)

Cluster Interconnect Status

- '-prot' displays the protocol in use
  - possibilities: VAPI SHM UDPL GM MX IT ELAN
  - mpirun **–prot** –srun ./hello.x
- Measure bandwidth between pairs of nodes using ping_pong_ring.c
  - copy shipped in /opt/hpmpi/help/ping_pong_ring.c –o ppring.x
  - bsub –I –n12 -ext "SLURM[nodes=12]" /opt/hpmpi/bin/mpirun
    –srun ./ppring.x 300000
- Exclude "suspect" nodes explicitly
  - bsub –ext "SLURM[nodes=12;exclude=n[1-4]]"
- Include "suspect" nodes explicitly
  - bsub –ext "SLURM[nodes=12;include=n[1-4]]"

HP-MPI Affinity Control

June 6, 2007

HP-MPI support for Process binding

- distributes ranks across nodes
  - mpirun -cpu_bind=[v,][policy[:maplist]] -srun a.out
  - [v] requests info on what binding is performed
- Policy is one of
  - LL|RANK|LDOM|RR|RR_LL|CYCLIC|FILL|FILL_LL|
  - BLOCK|MAP_CPU|MAP_LDOM|PACKED|HELP
  - MAP_CPU and MAP_LDOM list of cpu#s
- Example: bsub –I –n8 mpirun -cpu_bind=v,MAP_CPU:0,2,1,3 –srun ./a.out

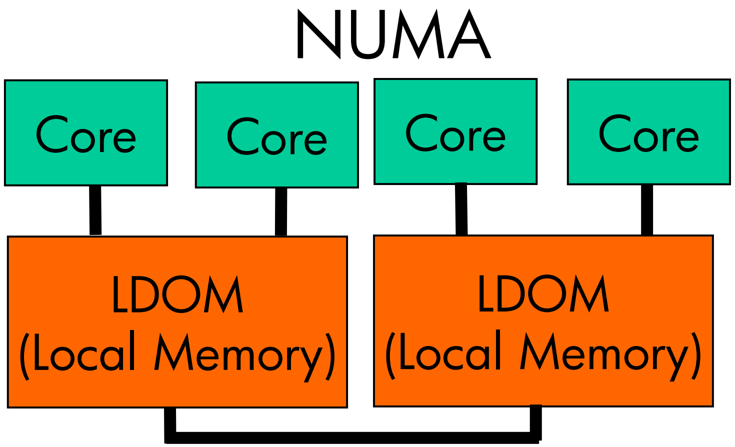
… **This is the map info for the 2nd node**

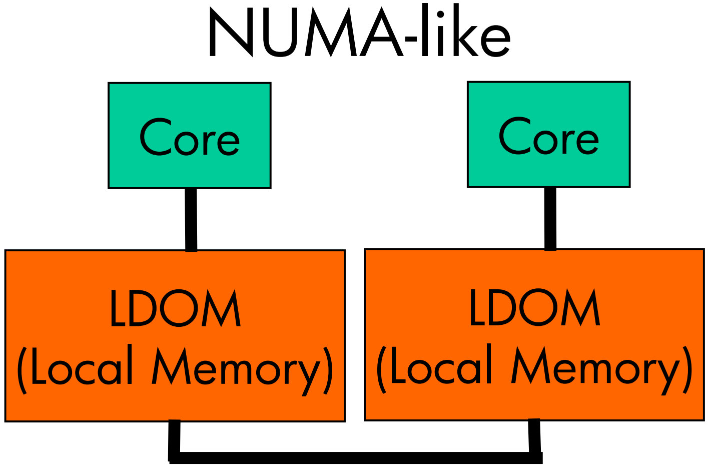
```
MPI_CPU_AFFINITY set to RANK, setting affinity of rank 4 pid 7156 on host dlcore1.rsn.hp.com to cpu 0

MPI_CPU_AFFINITY set to RANK, setting affinity of rank 5 pid 7159 on host dlcore1.rsn.hp.com to cpu 2

MPI_CPU_AFFINITY set to RANK, setting affinity of rank 6 pid 7157 on host dlcore1.rsn.hp.com to cpu 1

MPI_CPU_AFFINITY set to RANK, setting affinity of rank 7 pid 7158 on host dlcore1.rsn.hp.com to cpu 3

…
```

HP-MPI support for Process binding

```
$MPI_ROOT/bin/mpirun -cpu_bind=help ./a.out
-cpu_binding help info
  cpu binding methods available:
    rank      - schedule ranks on cpus according to packed rank id
    map_cpu   - schedule ranks on cpus in cycle thru MAP variable
    mask_cpu  - schedule ranks on cpu masks in cycle thru MAP variable
    ll        - bind each rank to cpu each is currently running on
  for numa based systems the following are also available:
    ldom      - schedule ranks on ldoms according to packed rank id
    cyclic    - cyclic dist on each ldom according to packed rank id
    block     - block dist on each ldom according to packed rank id
    rr        - same as cyclic, but consider ldom load avg.
    fill      - same as block, but consider ldom load avg.
    packed    - bind all ranks to the same ldom as lowest rank
    slurm     - slurm binding
    ll        - bind each rank to ldom each is currently running on
    map_ldom  - schedule ranks on ldoms in cycle thru MAP variable
```

Memory Models





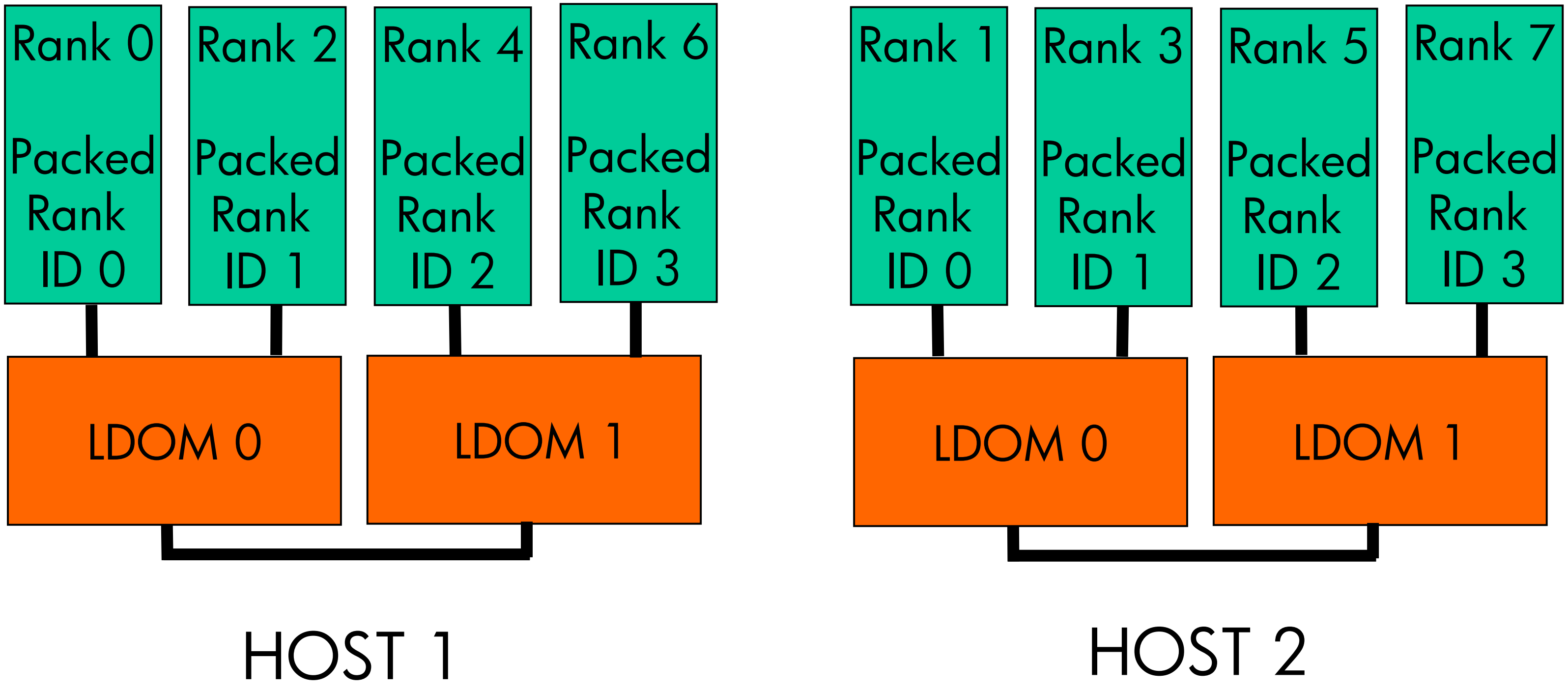
Examples of NUMA or NUMA-like systems:

- Dual-core Opteron has (in effect) local and remote memories, is considered a NUMA
- Single-core Opteron with memory controller is considered as a NUMA-like system
- Cell-based Itanium SMP system, is considered a NUMA system.

Example of Rank and LDOM distributions

mpirun –np 8 –srun -m=*cyclic*

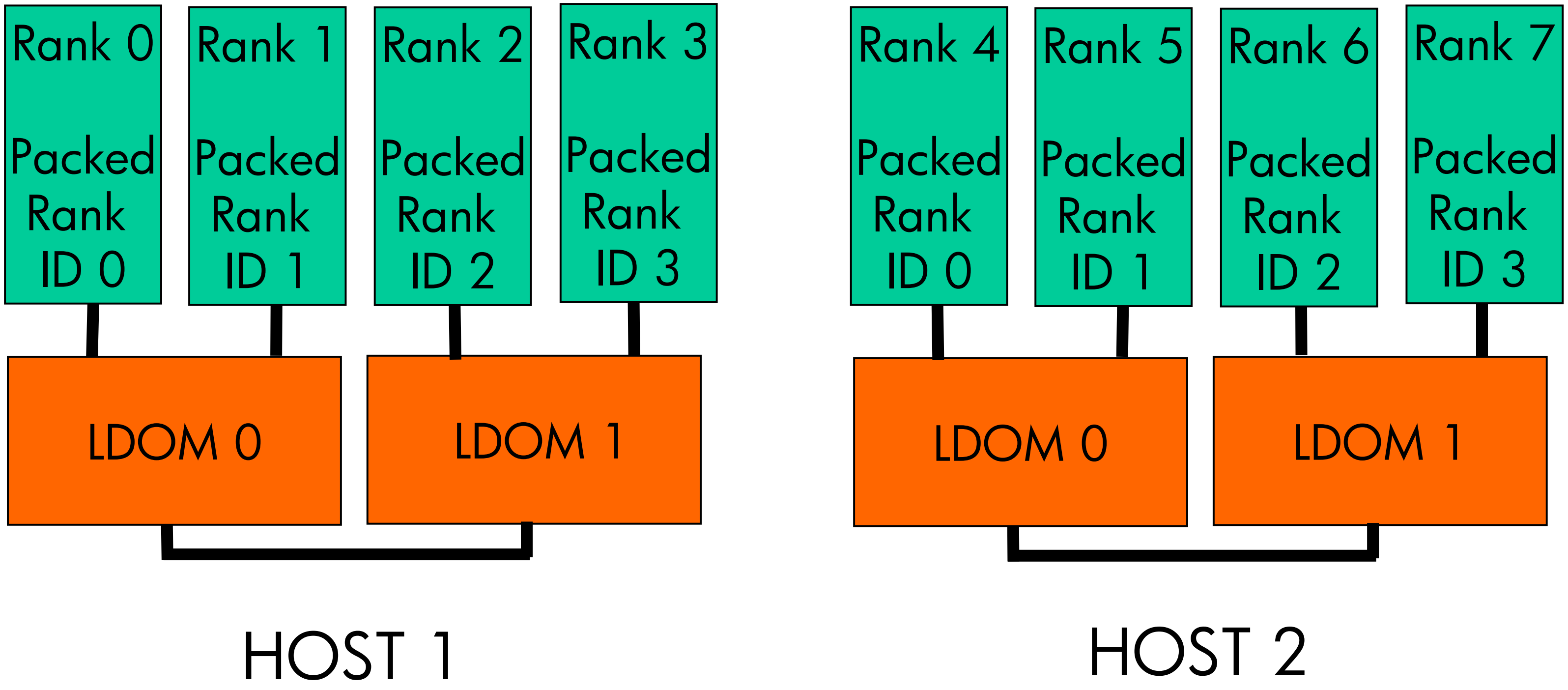
causes ranks and Packed Rank IDs to be distributed across 2 4-Core hosts as:



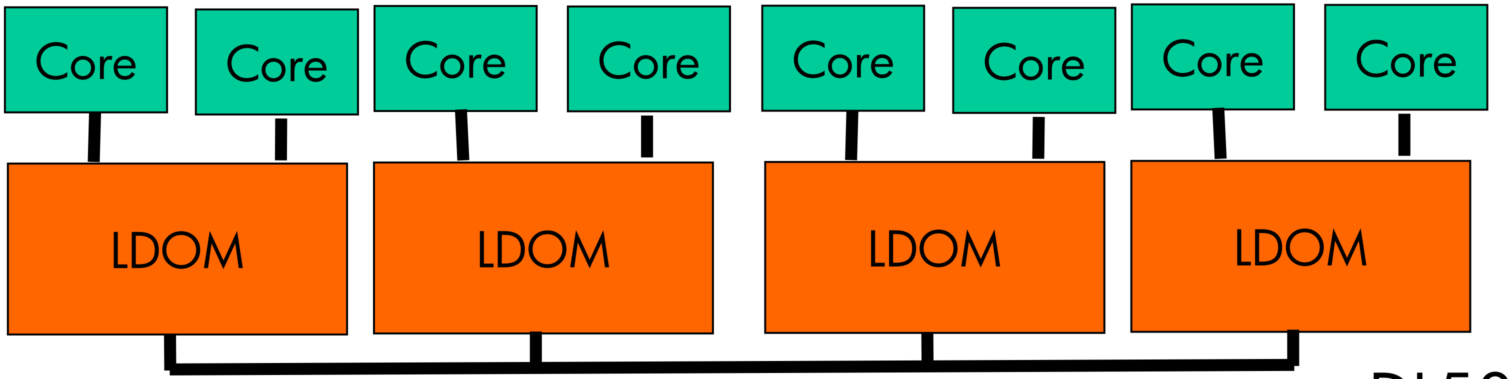
Another Example of Rank and LDOM distributions

mpirun –np 8 –srun -m=*block*

causes ranks and Packed Rank IDs to be distributed across 2 4-Core hosts as:



ccNUMA and I/O buffer-cache Interaction



Core Core Core Core Core Core Core Core

LDOM LDOM LDOM LDOM

DL585/4p8c

- On Opteron systems, memory can either be 100% interleaved among processors or 100% processor-local
  - For best performance, we use processor-local memory
- Linux can use all available memory for IO buffering
- When a user process requests local memory and the local memory is in use for IO buffering,
- LINUX assigns the memory on another processor → *worst-case latency*
- Given user demand for local memory, LINUX frees the IO buffers over time – at which point the best runtime is achieved

HP-MPI Scaleout

June 6, 2007

HP-MPI Scaleout Challenges

- Scalable process startup
  - reducing number of open sockets
  - Tree structure of MPI Daemons
  - now handles > **256 MPI ranks (srun and appfile)**
- Scalable teardown of processes
- Scalable Licensing
  - rank 0 checks for an N rank license.
- Scalable setup data
  - reduced Init4 Message size by 96%
- Managing IB Buffer requirements
  - physical memory pinning
- 1-sided lock/unlock now over IB if using VAPI

Managing IB Buffer requirements

- Two modes: RDMA and Shared-Receive-Queue

- The amount of memory pinned (locked in physical memory)

  - 1) memory which is always pinned (base)
  - 2) memory that may be pinned depending on communication. (dynamic)

- maximum_dynamic_pinned_memory = min(2 * max_messages * chunksize), (physical_memory / local_ranks) * pin_percentage);

  - max_messages is 3 * remote connections and chunksize varies depending on the protocol.
    - for IB it is 4MB and for GM it is 1MB.
  - maximum_dynamic_pinned_memory <= MPI_PIN_PERCENTAGE of rank's portion of physical memory. For large clusters, the limit will generally be based on the pin_percentage as 2*max_messages*chunksize gets large for even moderate clusters.
  - MPI_PIN_PERCENTAGE is 20% by default, but can be changed by the user.

Managing IB Buffer reqs cont

- Default is -rdma from 1 to 1024 ranks.
- Default is -srq mode for 1025 ranks or larger.
- "base" memory is based on the number of off-host connections.
- Without –srq (aka -rdma):
  - base_pinned_memory = envelopes * 2 * shortlen * N
- With -srq:
  - base_pinned_memory = min(N * 8 , 2048) * 2 * shortlen
- envelopes = # of envelopes for each connection, default is 8 (can be changed by the user)
- shortlen = short message length, default is 16K for infiniband (uDAPL and VAPI).
-

Managing IB Buffer reqs cont

- For a 2048 CPU job (memory per rank):

8 * 2 * 16K * 2047 = 524,032K (WITHOUT srq)

2048 * 2 * 16K = 65,536K (WITH srq)

- If we have two ranks on a node, then the total pre-pinned memory will be
  - around 1G without srq and 128MB with srq.
- For 4 ranks per node (still 2048 CPU's total)
  - 2048 ranks --> roughly 2GB without SRQ and 256MB with SRQ.

Shared-Receive-Queue model for Dynamic Message Buffer

- HP-MPI default mode for more than 1024 ranks
- Also triggered with –**srq** option for mpirun
- Shared-Receive-Queue
  - A single shared memory communication queue on each node
    - Other processes write directly to this buffer.
    - Buffer is in shared memory
  - Size of queue grows with the number of ranks in the job up to maximum size at 1024 ranks

$$SRQ\_dynamic\_memory = \min(Nranks, 1024) * 4 * shortlen * RanksPerNode$$

- *shortlen* = short message length. Determined by interconnect
- *Nranks* = Number of MPI ranks in the job
- *RanksPerNode* = Number of ranks per node

Effect of PIN Percentage on Buffer Memory

Change PIN Percentage to **increase amount of usable** base memory

Problem:

- a.out: Rank 0:23: MPI_Init: ERROR: The total amount of memory that may be pinned (210583540 bytes), is insufficient to support even minimal rdma network transfers. This value was derived by taking 20% of physical memory (2105835520 bytes) and dividing by the number of local ranks (2). A minimum of 253882484 bytes must be able to be pinned.

Solution:

- These values can be changed by setting environment variables
  - **MPI_PIN_PERCENTAGE**
  - **MPI_PHYSICAL_MEMORY** (Mbytes).
- In this case, 210583540 bytes is about 83% of the 253882484 bytes required.
- Increasing the MPI_PIN_PERCENTAGE from the default of 20% to 24% is sufficient to allow the application to run. Here is how to set to 30%:

```
$MPI_ROOT/bin/mpirun -e MPI_PIN_PERCENTAGE=30 –srun ./a.out
```

Managing InfiniBand Message Buffer Example

1200 ranks over InfiniBand used for this example

RDMA Mode

Memory footprint measured with ‘top’

```
PID   USER      PR  NI   VIRT  RES  SHR S %CPU %MEM    TIME
```

MPI_RDMA_NENVELOPE=8 gives optimum performance at a reasonable memory footprint

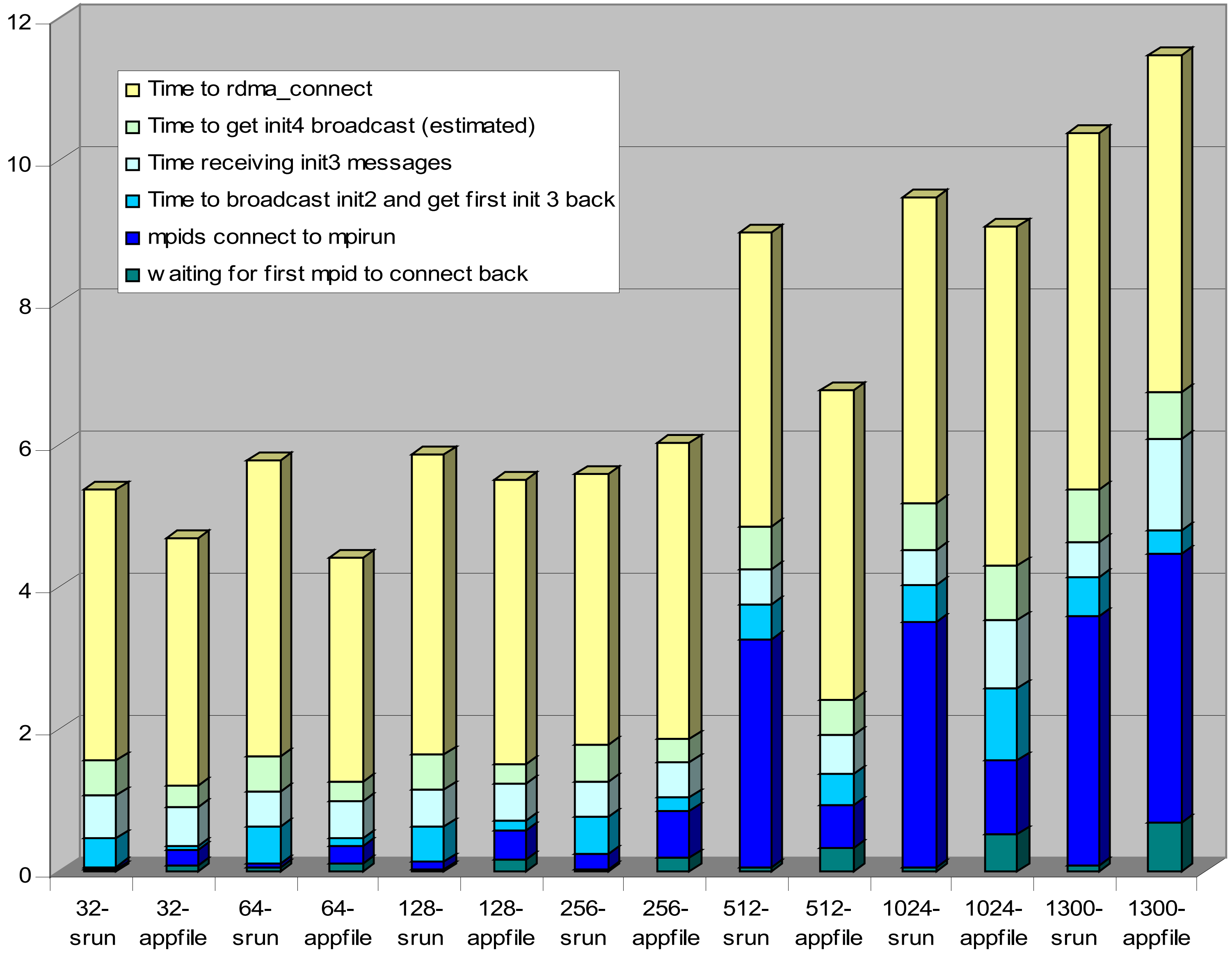
MPI_RDMA_NENVELOPE value	**Memory footprint (MB)**	**CPU Time Sec**
2	**201**	***BAD IDEA !***
4	**279**	27
6	**356**	25
8	**432**	21.4
10	**508**	25.4

Managing IB Buffer reqs cont

- Latency for RDMA vs SRQ

	rdma	srq
0 byte latency :	3.97us	7.09us
4M bandwidth:	903.61	902.63

Startup Performance Data

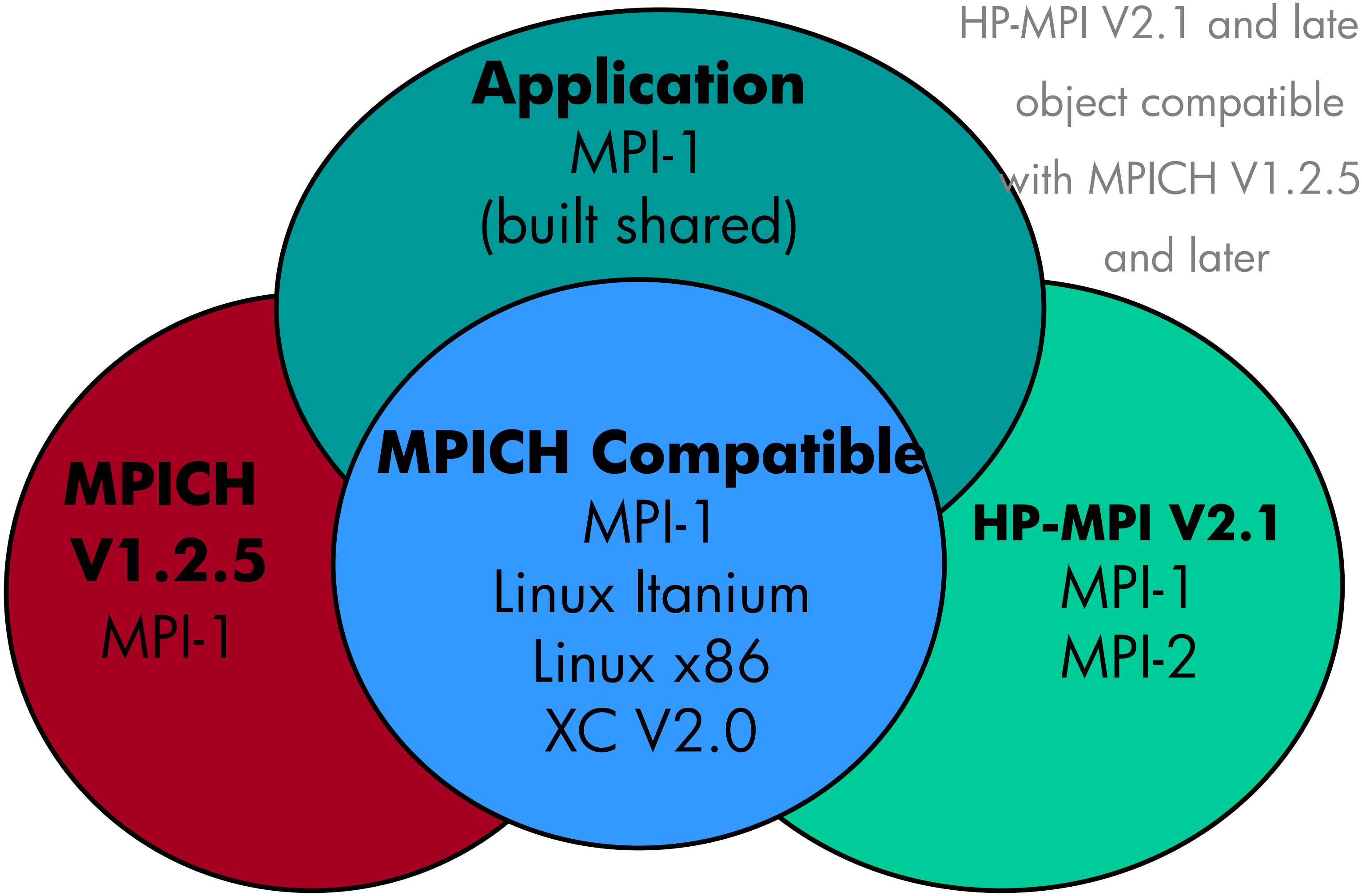


References

- HP-MPI User’s Guide
- XC User’s Guide

HP-MPI Object Compatibility

HP-MPI V2.1 and later is object compatible with MPICH V1.2.5 and later



Application
MPI-1
(built shared)

MPICH V1.2.5
MPI-1

MPICH Compatible
MPI-1
Linux Itanium
Linux x86
XC V2.0

HP-MPI V2.1
MPI-1
MPI-2

A compatibility is documented in the MPI V2.1 & later Release Note

SFS

June 6, 2007

Lustre support for SFS for XC

- Lustre allows individual files to be striped over multiple OSTs (Object Storage Targets) to improve overall throughput
- “striping_unit” = <value>
  - Specifies number of consecutive bytes of a file that are stored on a particular IO device as part of a stripe set
- “striping_factor” = <value>
  - Specifies the number of IO devices over which the file is striped. Cannot exceed the maximum defined by the system administrator
- “start_iodevice” = <value>
  - Specifies the IO device from which striping will begin

Lustre support for SFS for XC - cont

- These need to be defined prior to file creation so that the call to MPI_File_open can access them:

```
/* set new info values. */
value = randomize_start();
MPI_Info_create(&info);
MPI_Info_set(info, "striping_factor", "16");
 MPI_Info_set(info, "striping_unit", "131072");
MPI_Info_set(info, "start_iodevice", value );
/* open the file and set new info */
MPI_File_open(MPI_COMM_WORLD, filename,
  MPI_MODE_CREATE | MPI_MODE_RDWR, info, &fh);
```

Questions?

Thanks

HP logo white on blue

June 6, 2007