ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

CSCS
Swiss National Supercomputing Centre

DAY 2: Cray Programming Environment for MPI and OpenMP Code Development

Multi-threaded Programming, Tuning and Optimization on Multi-core MPP Platforms

15-17 February 2011

CSCS, Manno

The Cray Programming Environment

Roberto Ansaloni
roberto.ansaloni@cray.com

~course02/slides/day1/CrayPE.pdf

Agenda

- Programming Environment Overview
  - Modules
- Compilers
  - PGI, Cray, GNU, Intel, Pathscale
- Programming considerations
- MPI Communications
- Running an application

Programming Environment Overview

Cray XE6 programming environment overview

- Several compilers available
  - PGI, Cray, GNU, Intel, (Pathscale)
- Optimized libraries
  - More on this tomorrow...
- Aprun command to launch jobs; similar to mpirun command
- PBSPro batch system by default
  - On CSCS palu SLURM is installed
- Performance tools: CrayPat, Apprentice2

Cray XE6 programming environment is SIMPLE

- Edit and compile the program (no need to specify include files or libraries)

```
$ vi mysrc.f90
$ ftn -o myexe mysrc.f90
```

- Edit the batch job file (myjob.job) (SLURM batch)

```
#SBATCH --job-name="myjob"
#SBATCH --nodes=10
aprun -n 240 ./myexe
```

- Run the job

```
$ qsub myjob.job
```

The module tool on the Cray XE6

- How can we get appropriate Compiler and Libraries to work with?
- module tool used on XE6 to handle different versions of packages (compiler, tools,...):

  e.g.: module load compiler1

  e.g.: module swap compiler1 compiler2

  e.g.: module load perftools
- taking care of changing of PATH, MANPATH, LM_LICENSE_FILE,.... environment.
- users should not set those environment variable in their shell startup files, makefiles,....
- keep things flexible to other package versions

Cray XE6 PE: module list

```
palu> module list
Currently Loaded Modulefiles:
  1) modules
  2) nodestat/2.2-1.0301.22648.3.3.gem
  3) sdb/1.0-1.0301.22744.3.24.gem
  4) MySQL/5.0.64-1.0301.2899.20.4.gem
  5) lustre-cray_gem_s/1.8.2_2.6.27.48_0.1.1_1.0301.5475.7.1-1.0301.23312.0.0
  6) udreg/1.3-1.0301.2236.3.6.gem
  7) ugni/2.0-1.0301.2365.3.6.gem
  8) gni-headers/2.0-1.0301.2497.4.1.gem
  9) dmapp/2.2-1.0301.2427.3.8.gem
 10) xpmem/0.1-2.0301.22550.3.6.gem
 11) slurm
 12) Base-opts/1.0.2-1.0301.21771.3.3.gem
```

- No pre-loaded programming environment module
- Should load a specific module to produce code optimized for the AMD Magny-Cours processor (xtpe-mc12)

Cray XE6 PE: module avail

```
palu> module avail cce

cce/7.2.4           cce/7.2.5           cce/7.2.6           cce/7.2.7
cce/7.2.8(default)  cce/7.3.0.145
```

- Which modules are available ?

Useful module commands

- Basic: load PGI compiler and Magny-Cours specific

  module load PrgEnv-pgi

  module load xtpe-mc12
- Change environment (GNU)

  module swap PrgEnv-pgi PrgEnv-gnu
- Load Cray environment and use a specific version of the compiler

  module swap PrgEnv-pgi PrgEnv-cray

  module swap cce cce/7.3.0.145
- Load the new MPICH2 environment

  module unload xt-mpt

  module load xt-mpich2

Compilers

Compiler drivers to create CLE executables

- When the PrgEnv is loaded the compiler drivers are also loaded
  - the compiler drivers also take care of loading appropriate libraries (-lmpich, -lsci, -lacml, -lpapi)

- Available drivers (also for linking of MPI applications):
  - Fortran 90/95 programs: ftn
  - Fortran 77 programs: f77
  - C programs: cc
  - C++ programs: CC

- Cross compiling environment
  - Compiling on a Linux service node
  - Generating an executable for a CLE compute node
  - Do not use standard compler names (pgf90, gcc, mpicc,...) unless you want a Linux executable for the service node

PGI programming environment (PrgEnv-pgi)

- **Overall Options**
  - -Mlist creates a listing file
  - -Minfo info about optimizations performed
  - -Mneginfo why certain optimizations are not performed
- **Preprocessor Options**
  - -Mpreprocess runs the preprocessor on Fortran files
- **Optimisation Options**
  - -fast chooses generally optimal flags for the target platform
  - -Mipa=fast,inline Inter Procedural Analysis
  - -Minline=levels:n number of levels of inlining

PGI programming environment

- **Language Options**
  - -Mfree process Fortran source using freeform specifications
  - -Mnomain useful for using the ftn driver to link programs with the main program written in C or C++ and one or more subroutines written in Fortran
  - -i8, -r8 treat INTEGER and REAL variables as 64-bit
  - -Mbyteswapio big-endian files in Fortran; XE6 is little endian
- **Parallelization Options**
  - -mp recognize OpenMP directives
  - -Mconcur automatic parallelization

man pages: pgf90, pgcc, pgCC

PGI User's Guide (Chapter 2) http://www.pgroup.com/doc/pgiug.pdf

Cray Compiler Environment (CCE): (PrgEnv-cray)

- Cray has a long tradition of high performance compilers
  - Vectorization for vector architectures
- In 2008 decided to move forward with Cray X86 compiler
  - Vector is back ? (SSE, AVX…)
  - CCE 7.0 released in December 2008
  - CCE 7.3 about to be released
- Still young but interesting…
  - FORTRAN 2008 Coarray support
  - Strongly supported by Cray

Cray compiler flags

- **Overall Options**
  - -ra creates a listing file with optimization info
- **Preprocessor Options**
  - -eZ runs the preprocessor on Fortran files
  - -F enables macro expansion throughout the source file
- **Optimisation Options**
  - -O2 optimal flags [enabled by default]
  - -O3 aggressive optimization
  - -O ipa<n> inlining, n=0-5

Cray compiler flags

- **Language Options**
  - -f free — process Fortran source using freeform
  - -s real64 — treat REAL variables as 64-bit
  - -s integer64 — treat INTEGER variables as 64-bit
  - -hbyteswapio — big-endian files in Fortran
XE6 is little endian (this is a link time option)
- **Parallelization Options**
  - -O omp — Recognize OpenMP directives [default]
  - -O thread<n> — n=0-3, aggressive parallelization, default n=2

man pages: crayftn, intro_directives, intro_pragmas, assign

http://docs.cray.com/cgi-bin/craydoc.cgi?mode=View;id=S-3901-71;idx=books_search;this_sort=;q=3901;type=books;title=Cray%20Fortran%20Reference%20Manual

Cray programming environment: assign

- Assigns options for library file open processing

  assign [assign options] assign_object

- **Interesting assign options**
  - -R removes all assign options for assign_object
  - -N <numcon> specifies foreign numeric conversion

    swap_endian the endianess of data is swapped during unformatted input and output.
- **assign object** used to specify the object of assign options
  - f:<filename> applies to filename
  - u:<unit> applies to Fortran unit number
  - g:su applies to all Fortran sequential unform. files

How to handle byte-swapped files with CCE

- Explicit usage of assign
  - Can control which files are byte-swapped

```
export FILENV=.assign
assign -R
assign -N swap_endian f:aof
aprun a.out
```

- Link the application with -hbyteswapio
  - All unformatted Fortran I/O are byte-swapped
  - This is equivalent to set

```
assign -N swap_endian g:su
assign -N swap_endian g:du
```

Cray programming environment: explain

- Displays the explanation for an error message: compile, run time
- Compiler error

```
cft90: llvm/lib/VMCore/Instructions.cpp:328: void
   llvm::CallInst::init(llvm::Value*, llvm::Value* const*, unsigned int):
Assertion `(i >= FTy->getNumParams() || FTy->getParamType(i) == Params[i]-
   >getType()) && "Calling a function with a bad signature!"' failed.
ftn-2116 crayftn: INTERNAL
```

```
rosa> explain ftn-2116
Internal : "Program" was terminated due to receipt of signal signal.
The process spawned by the command to run the specified program was
terminated by a system signal.  See the signal(2) man page for more
information about this signal.
This message does not indicate a problem with your code, and you may be able
to change your program so that this error does not occur.
Please notify your product support representative with the text of this
message, the source code being compiled, the version of the compiler, and the
Command line options in effect.
```

Cray programming environment: explain

- I/O runtime error

```
Open IOSTAT=5016

rosa> explain lib-5016
An EOF or EOD has been encountered unexpectedly.
The file terminated unexpectedly, perhaps without the proper end-of-file
record or end-of-data control information.  The file specification may be
incorrect, or the file may be corrupted.
Ensure that the file specification is correct.  If the file is corrupted,
create the data again, if possible.
See the man pages for assign(1) and asgcmd(1).
The error class is UNRECOVERABLE (issued by the run time library).
```

Pathscale programming env (PrgEnv-pathscale)

- **Overall Options**
  - -LIST:=ON creates a listing file
  - -LNO:simd_verbose=ON info about vectorizations
- **Preprocessor Options**
  - -cpp -ftpp runs the preprocessor on C or FORTRAN files
- **Optimisation Options**
  - -O3 –OPT:Ofast chooses generally optimal flags
  - -Ofast aggressive optimization
  - -ipa Inter Procedural Analysis

Pathscale programming environment

- **Language Options**
  - -freeform process Fortran source using free form specifications
  - -i8, -r8 treat INTEGER and REAL variables as 64-bit
  - -byteswapio big-endian files in Fortran; XE6 is little endian
- **Parallelization Options**
  - -mp recognize OpenMP directives
  - -apo automatic parallelization

man pages: eko, pathf90, pathcc

Cray XE compilers Rosetta Stone

Feature	PGI	Pathscale	Cray
Listing	-Mlist	-LIST:=ON	-ra
Diagnostic	-Minfo -Mneginfo	-LNO:simd_verbose=ON	(produced by -ra)
Free format	-Mfree	-freeform	-f free
Preprocessing	-Mpreprocess	-cpp -ftpp	-eZ -F
Suggested Optimization	-fast	-O3 –OPT:Ofast	(default)
Aggressive Optimization	-Mipa=fast,inline	-Ofast	-O3, fp3
Variables size	-r8 –i8	-r8 -i8	-s real64 –s integer64
Byte swap	-byteswapio	-byteswapio	-h byteswapio
OpenMP recognition	-mp=nonuma	-mp	(default)
Automatic parallelization	-Mconcur	-apo	-h autothread

Other programming environments

- GNU (PrgEnv-gnu)
  - Suggested options: -O3 –ffast-math –funroll-loops
  - Compiler feedback: -ftree-vectorizer-verbose=2
  - OpenMP: -fopenmp
  - Man pages: gcc, gfortran, g++

- Intel (PrgEnv-intel)
  - Suggested options: -O3
  - Aggressive options: -ffast-math -funroll-loops -msse3 -ftree-vectorize
  - OpenMP: -openmp=on
  - An extra control thread is spawn: issues when pinning threads to cores
  - Man pages: ifort, icc

Some Programming Considerations

Memory allocation

- Linux provides some environment variables to control how malloc behaves (Equivalent to using the mallopt system call)
- MALLOC_MMAP_MAX_
  - Number of 'internal' non heap, mmap regions (default 64)
  - Using MMAP regions turns out to be very costly compared to using the heap
  - They exist to allow a program to return unused memory back to the system more easily so it may be used by other processes on the node: no need for it on the XE6
  - Suggested value: `export MALLOC_MMAP_MAX_=0`

Memory allocation

- MALLOC_TRIM_THRESHOLD_
  - Amount of free space at the top of the heap after a free() that needs to exist before malloc will return the memory to the OS
  - Returning memory to the OS is costly. The default setting of 128 KBytes is much too low for a node with 4 GBytes of memory and one application.
  - Suggested value:

```
export MALLOC_TRIM_THRESHOLD_=536870912
```

Huge pages - description

- The AMD Opteron supports multiple page sizes
  - The base page size is 4 Kbytes; the default huge page size is 2 Mbytes.
  - Use HUGETLB_DEFAULT_PAGE_SIZE to modify huge page size
  - Available sizes: 128KB, 512KB, 2MB, 8MB,16MB, 64MB.
- Huge pages can provide better performance by reducing the number of TLB misses and by enforcing larger sequential physical memory inside each page
- Useful man pages: aprun, intro_hugepages

Huge pages

- Memory fragmentation issue: the number of huge pages available on a node declines over time: this is affected by all applications that have run on the node since it was last booted.
- Huge pages are important in PGAS codes to determine remote memory mapping

Huge pages - howto

- Link with the correct library: `-lhugetlbfs`
- Activate the library at run time: `export HUGETLB_MORECORE=yes`
- Launch the program wirh aprun preallocating huge pages
  - request <size> MBytes per PE `-m<size>h` (advisory mode)
  - request <size> MBytes per PE `-m<size>hs` (required mode)
  - What if the request can't be satisfied ? Slow or crash ?
- Example: `aprun -m700hs -N2 -n8 ./my_app`
  - Requires 1400 MBytes of huge page memory on each node

MPI Communications on the Cray XE6

CRAY THE SUPERCOMPUTER COMPANY

CSCS Swiss National Supercomputing Centre

HP2C

MPICH2 environment variables

- Several environment variables are available to control MPI features (man mpi or intro_mpi)
- MPICH_ENV_DISPLAY
  - If set, causes rank 0 to display all MPICH environment variables
- MPICH_CPUMASK_DISPLAY
  - If set, causes each MPI rank in the job to display its CPU affinity bitmask
- MPICH_MAX_THREAD_SAFETY
  - Specifies thread-safety level
  - MPI_THREAD_MULTIPLE requires a specific library: link to -lmpich_threadm

MPICH2 GNI Netmod Message Protocols

- Eager Protocol
  - For a message that can fit in a GNI SMSG mailbox (E0)
  - For a message that can't fit into a mailbox but is less than MPICH_GNI_MAX_EAGER_MSG_SIZE in length (E1)
- Rendezvous protocol (LMT)
  - RDMA Get protocol – up to 512 KB size messages by defalt
  - RDMA Put protocol – above 512 KB

Maximum message size for E0 varies with Job Size

- Protocol for messages that can fit into a GNI SMSG mailbox
- The default varies with job size, although this can be tuned by the user to some extent

ranks in job	maximum bytes of user data
<= 1024	984
>1024 && <=16384	472
> 16384	216

MPICH_GNI_MAX_VSHORT_MSG_SIZE

- Can be used to control the maximum size message that can go through the private SMSG mailbox protocol (E0 *eager* path).
- Default varies with job size.
- Maximum size is 1024 bytes. Minimum is 80 bytes.
- If you are trying to demonstrate an MPI_Alltoall at very high count, with smallest possible memory usage, may be good to set this as low as possible.
- If you know your app has a scalable communication pattern, and the performance drops at one of the edges shown on the table, you may want to set this environment variable.
- Pre-posting receives for this protocol avoids a potential extra memcpy at the receiver.

MPICH_GNI_MAX_EAGER_MSG_SIZE

- Default is 8192 bytes
- Maximum size message that go through the *eager* (E1) protocol
- May help for applications sending medium size messages
- Maximum allowable setting is 131072 bytes
- Pre-posting receives can avoid potential double memcpy at the receiver.
- Note that a 40-byte Nemesis header is included in account for the message size.

MPICH_GNI_RDMA_THRESHOLD

- Default is now 1024 bytes
- Controls the threshold at which the GNI netmod switches from using FMA for RDMA read/write operations to using the BTE.
- Since BTE is managed in the kernel, BTE initiated RDMA requests can progress even if the applications isn't in MPI.
- But using the BTE may lead to more interrupts being generated

MPICH_GNI_NDREG_LAZYMEM

- Default is enabled. To disable
  export MPICH_GNI_NDREG_LAZYMEM=disabled
- Controls whether or not to use a lazy memory deregistration policy inside UDREG. Memory registration is expensive so this is usually a good idea.
- Only important for those applications using the LMT (large message transfer) path, i.e. messages greater than MPICH_GNI_MAX_EAGER_MSG_SIZE.
- Disabling may be a workaround for some UDREG issues
- However, disabling results in a significant drop in measured bandwidth for large transfers ~40-50 %.

MPICH_GNI_DYNAMIC_CONN

- Enabled by default
- Normally want to leave enabled so mailbox resources (memory, NIC resources) are allocated only when the application needs them
- If application does all-to-all or many-to-one/few, may as well disable dynamic connections. This will result in significant startup/shutdown costs though.
- Recent bugs have been worked around by disabling dynamic connections.
- Syntax for disabling:
  export MPICH_GNI_DYNAMIC_CONN=disabled

MPI_Allgather / MPI_Allgatherv

- With MPT 5.1 switched to using Seastar-style algorithm where for short transfers/rank: use MPI_Gather(v)/MPI_Bcast rather than ANL algorithm
- Switchover from Cray algorithm to ANL algorithm can be controlled by the MPICH_ALLGATHER_VSHORT_MSG and MPICH_ALLGATHERV_VSHORT_MSG environment variables. By default enabled for transfers/rank of 1024 bytes or less
- The Cray algorithm can be deactivated by setting

export MPICH_COLL_OPT_OFF=mpi_allgather
export MPICH_COLL_OPT_OFF=mpi_allgatherv

MPI_Alltoall

- Optimizations added in MPT 5.1
- Switchover from ANL's implementation of Bruck algorithm (IEEE TPDS, Nov. 1997) is controllable via the MPICH_ALLTOALL_SHORT_MSG environment variable. Defaults are

ranks in communicator	
<= 512	2048
>512 && <=1024	1024
> 1024	128

- New algorithm can be disabled by **export MPICH_COLL_OPT_OFF=mpi_alltoall**

MPI_Allreduce/MPI_Reduce

- The ANL smp-aware MPI_Allreduce/MPI_Reduce algorithms can cause issues with bitwise reproducibility. To address this Cray MPICH2 has two new environment variables starting with MPT 5.1 -
- **MPI_ALLREDUCE_NO_SMP**
  - disables use of smp-aware MPI_Allreduce
- **MPI_REDUCE_NO_SMP**
  - disables use of smp-aware MPI_Reduce

MPI_Bcast

- Starting with MPT 5.1, all ANL algorithms except for binomial tree are disabled since the others perform poorly for communicators with 512 or more ranks
- To force the tree algorithm to be used for all cases set the MPICH_BCAST_ONLY_TREE environment variable to 0, i.e.

  export MPICH_BCAST_ONLY_TREE=0

MPICH_SMP_SINGLE_COPY_SIZE

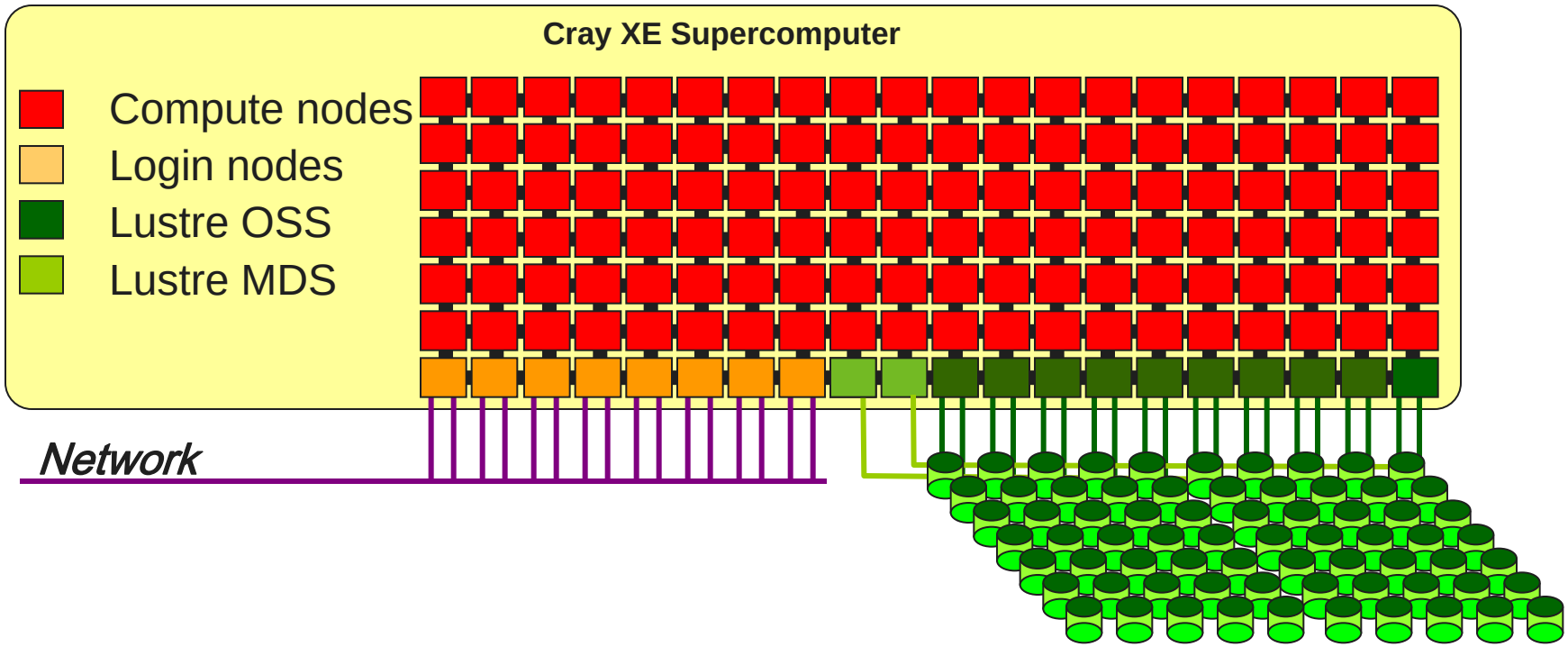
- Default is 8192 bytes
- Specifies threshold at which the Nemesis shared memory channel switches to a single-copy, XPMEM based protocol for intra-node messages

MPICH_SMP_SINGLE_COPY_OFF

- In MPT 5.1 the default is enabled
- Specifies whether or not to use a XPMEM-based single-copy protocol for intra-node messaes of size MPICH_SMP_SINGLE_COPY_SIZE bytes or larger
- May need to set this environment variable if
  - Finding XPMEM is kernel OOPses (check the console on the SMW)
  - Sometimes helps if hitting UDREG problems. XPMEM goes kind of crazy with Linux mmu notifiers and causes lots of UDREG invalidations (at least the way MPICH2 uses XPMEM).

I/O on the Cray XE6

The Storage Environment



Lustre

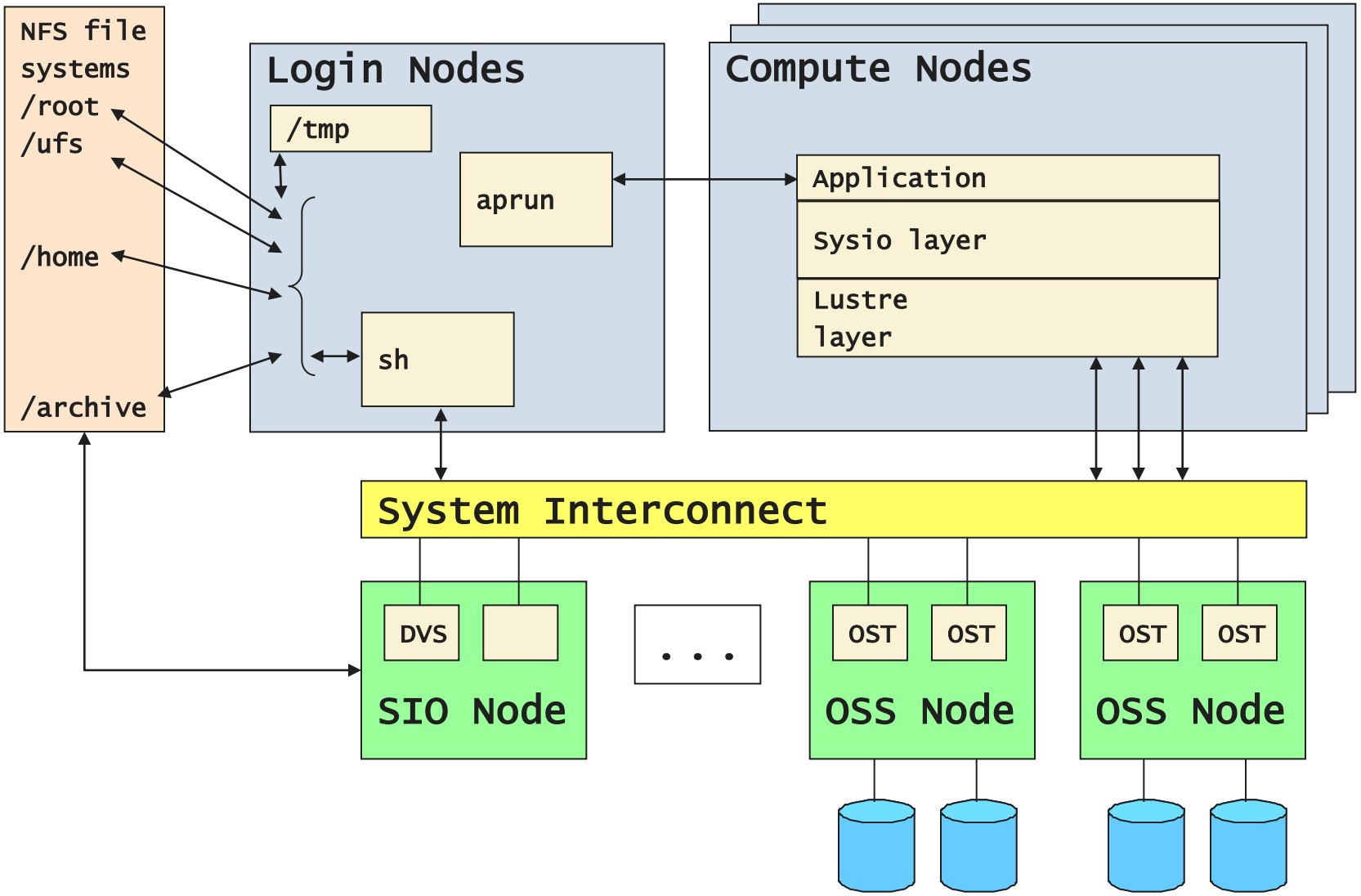
high performance

parallel filesystem

Cray XE I/O architecture

- All I/O is offloaded to service nodes
- Lustre
  - High performance parallel I/O file system
  - Direct data transfer between compute nodes and files
- DVS
  - Virtualization service
  - Allows compute nodes to access NFS mounted on service node
  - Applications must execute on file systems mounted on compute nodes
- No local disks
- /tmp is a MEMORY file system, on each login node

Cray XT I/O architecture



Lustre

- A scalable cluster file system for Linux
  - Developed by Cluster File Systems, Inc.
  - Name derives from “Linux Cluster”
  - The Lustre file system consists of software subsystems, storage, and an associated network
- **MDS** – metadata server
  - Handles information about files and directories
- **OSS** – Object Storage Server
  - The hardware entity
  - The server node
  - Support multiple OSTs
- **OST** – Object Storage Target
  - The software entity
  - This is the software interface to the backend volume

Lustre File Striping

- Stripes defines the number of OSTs to write the file across
  - Can be set on a per file or directory basis

- CRAY recommends that the default be set to
  - not striping across all OSTs, but
  - set default stripe count of one to four

- But not always the best for application performance.
  As a general rule of thumbs :
  - If you have one large file: stripe over all OSTs
  - If you have a large number of files (~2 times #OSTs): turn off striping

Lustre lfs command

- lfs is a lustre utility that can be used to create a file with a specific striping pattern, displays file striping patterns, and find file locations
- The most used options are :
  - setstripe
  - getstripe
  - df
- For help execute lfs without any arguments

```
$ lfs
lfs > help
Available commands are:
        setstripe
        find
        getstripe
        check
```

Lustre striping hints

- For maximum aggregate performance: Keep all OSTs occupied
- Many clients, many files: Don't stripe
  - If number of clients and/or number of files >> number of OSTs: Better to put each object (file) on only a single OST.
- Many clients, one file: Do stripe
  - When multiple processes are all accessing one large file: Better to stripe that single file over all of the available OSTs.
- Some clients, few large files: Do stripe
  - When a few processes access large files in large chunks: Stripe over enough OSTs to keep the OSTs busy on both write and read paths.

Running an application on the Cray XE6

Running an application on the Cray XE

- ALPS : Application Level Placement Scheduler
- aprun is the ALPS application launcher
  - It must be used to run application on the XT compute nodes
  - If aprun is not used, the application is launched on the login node (and likely fails)
  - aprun man page contains several useful examples
- aprun has (at least) 3 important parameters to control:
  - The total number of MPI tasks (PEs): -n
  - The number of MPI tasks per node: -N
  - The number of OpenMP threads: -d

Running an application on the Cray XE6

- Assuming a XE6 system (24 cores per node)
- Pure MPI application, using all the available cores in a node

```
$ aprun –n <npes>
```

- Pure MPI application, using only 1 core per node
  - npes MPI tasks, 24*npes cores allocated, npes nodes allocated
  - Can be done to increase the available memory for the MPI tasks

```
$ aprun –N 1 –n <npes>
```

- Hybrid MPI/OpenMP application, 4 MPI ranks per node
  - npes MPI tasks, 6 OpenMP threads each
  - need to set OMP_NUM_THREADS

```
$ export OMP_NUM_THREADS=6
$ aprun –N 4 –d 6 –n <npes>
```

aprun CPU Affinity control

- CPU affinity options enable to bind a PE or thread to a particular CPU or a subset of CPUs on a node
- CNL can dynamically distribute work by allowing PEs and threads to migrate from one CPU to another within a node
- In some cases, moving PEs or threads from CPU to CPU increases cache and translation lookaside buffer (TLB) misses and therefore reduces performance
- aprun CPU affinity option:
  - -cc cpu_list | keyword
  - suggested (default) settings: -cc cpu
  - The cpu keyword (the default) binds each PE to a CPU within the assigned NUMA node

aprun CPU Affinity control (Pathscale)

- Pathscale compiler provide its own control of cpu affinity: this should be disabled to avoid interference with ALPS
  - export PSC_OMP_AFFINITY=FALSE

Further aprun affinity control

- Cray XE6 systems use dual-socket 24-core compute nodes
  - Each die (6 cores) is considered a NUMA-node
- Remote-NUMA-node memory references, can adversely affect performance.
- aprun memory affinity options:
  - -S pes_per_numa_node #PEs to allocate per NUMA node
  - -sl list_of_numa_nodes list of NUMA nodes to use [0,3]
  - -sn numa_nodes_per_node #NUMA nodes to consider
  - -ss strict memory containment per NUMA node; a PE can allocate only the memory local to its assigned NUMA node

Running an application on the Cray XT - MPMD

- aprun supports MPMD – Multiple Program Multiple Data
- Launching several executables on the same MPI_COMM_WORLD

```
$ aprun -n 128 exe1 : -n 64 exe2 : -n 64 exe3
```

Core specialization

- System ‘noise’ on compute nodes may significantly degrade scalability for some applications
- Core Specialization can mitigate this problem
  - 1 core per node will be dedicated for system work (service core)
  - As many system interrupts as possible will be forced to execute on the service core
  - The application will not run on the service core
- Use aprun-r to get core specialization

```
$ aprun –r –n 100 a.out
```

- apcount provided to compute total number of cores required

```
$ qsub -l mppwidth=$(apcount -r 1 1024 24)job
  aprun -n 1024 -r 1 a.out
```

Running a batch application with SLURM

- The number of required nodes can be specified in the job header
- The job is submitted by the qsub command
- At the end of the exection output and error files are returned to submission directory
- Environment variables are inherited
- The job starts in the directory from which the job has been submitted

Hybrid MPI + OpenMP

```
#!/bin/bash
#SBATCH --job-name="hybrid"
#SBATCH --time=00:10:00
#SBATCH --nodes=8

export OMP_NUM_THREADS=6
aprun -n32 -d6 a.out
```

Starting an interactive session with SLURM

- An interactive job can be started by the SLURM **salloc** command

- Example: allocate 8 nodes

```
$ salloc –N 8
```

Further SLURM info available from CSCS web page: www.cscs.ch
User Entry Point / How to Run a Batch Job / Palu - Cray XE6

Watching a launched job on the Cray XE

- xtnodestat
  - Shows XE nodes allocation and aprun processes
  - Both interactive and PBS
- apstat
  - Shows aprun processes status
  - apstat overview
  - apstat –a[apid]info about all the applications or a specific one
  - apstat –n info about the status of the nodes
- Batch qstat command
  - shows batch jobs

Documentation

- Cray docs site

  http://docs.cray.com

- Starting point for Cray XE info

http://docs.cray.com/cgi-bin/craydoc.cgi?mode=SiteMap;f=xe_sitemap

- Twitter ?!? 

  http://twitter.com/craydocs

Thank you !