



Abel welcome users

Ole W. Saastad, Dr.Scient
USIT / SUF / VD



What have changed ?

Changes visible for users.....

Outline

- Hardware overview
 - Compute nodes
 - Interconnect
 - Storage – local and permanent
- Compilers
- Performance libraries
- OpenMPI, MPIS
- Queue system



Hardware overview



Universitetets senter for informasjonsteknologi

Installed compute hardware

630 Supermicro nodes

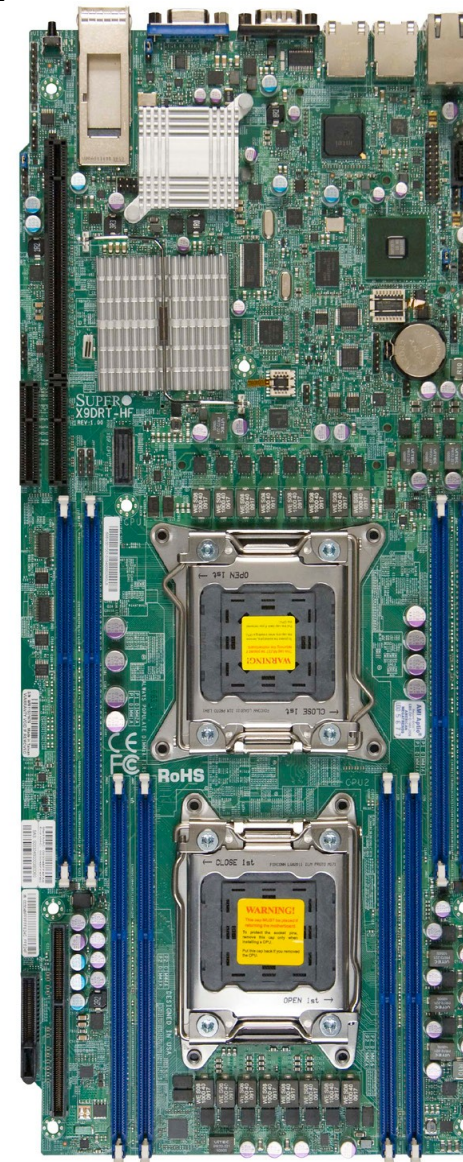
Two socket

Intel E5-2670

2.6 GHz octa core

64 GiB memory

FDR InfiniBand



Compute nodes - performance

- CPU performance
 - 332 Gflops/s Theoretical
 - 318 Gflops/s Practical – HPL (top500)
- Memory bandwidth
 - 63 GiB/s Practical (streams)
- Memory latency
 - 115 μ seconds (random access double)

Node performance

High Performance Linpack performance (top500 test) :

T/V	N	NB	P	Q	Time	Gflops
WR11R2R4	87500	180	4	4	1404.46	3.180e+02

||Ax-b||_oo/(eps*(||A||_oo*||x||_oo+||b||_oo)*N) = 0.0033152 PASSED

=====

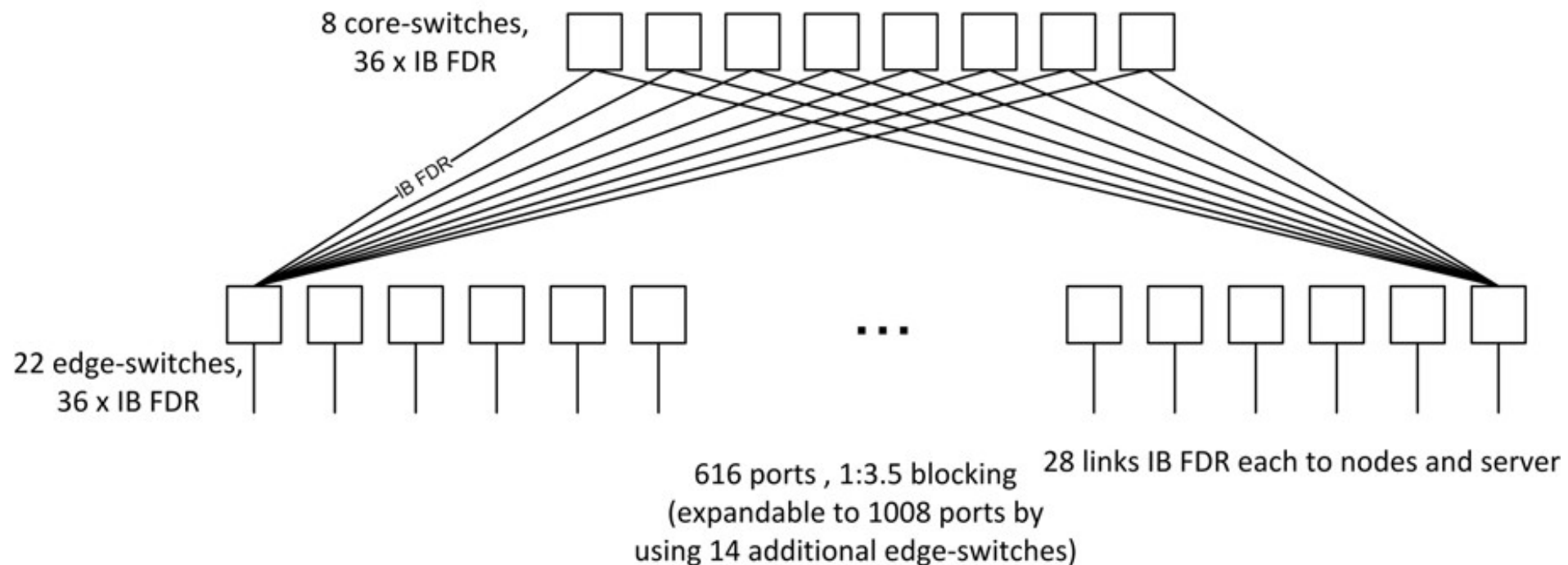
```
[olews@login-0-0 hpl]$ ./xhpl-max.sh HPL.single.node.log
```

```
No clock freq given, setting it to 2.6 GHz
```

```
High perf. linpack results:
```

Params	size	block	nxm	time	Tflops	%peak
WR11R2R4	87500	180	4x4	1404.46	0.318	95.6
WR11R2R4	87500	200	4x4	1408.52	0.317	95.3
WR11R2R4	85000	200	4x4	1301.79	0.315	94.5

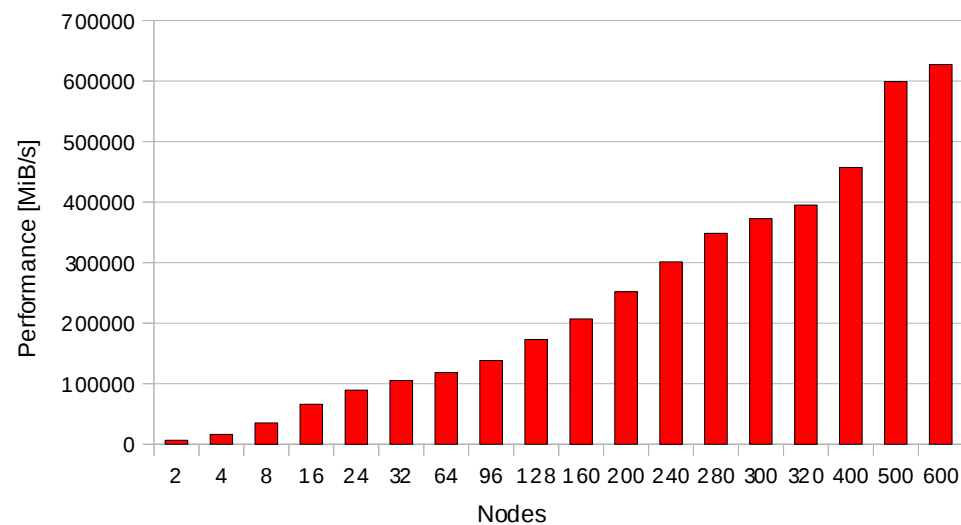
All nodes have InfiniBand



All nodes have InfiniBand



All2all - bisection bandwidth
OpenMPI, 1ppn



InfiniBand Basics

- FDR – Fourteen Data Rate / signaling rate
 - Single data rate equal 2.5 Gbits/s
 - Quad data rate equal 10 Gbits/s
 - Fourteen data rate is 14 Gbits
 - Octa data rate equal 20 Gbits/s soon
- Each connection is 4x
 - Four links in a single cable
 - 14 Gbits/s times 4 yield 56 Gbits/s

InfiniBand Basics

- FDR – 56 Gbits/s signaling rate
 - Encoding 64/66
 - Effective data rate 54.3 Gbits/s
 - Numbers are 10 based, $G=10^9$
 - FDR bandwidth 6.32 GiB/second
 - However – 16 cores / node
 - Only 405 MiB/s per core or MPI rank

InfiniBand Basics

- FDR – latency – the hard part
 - Ping-pong Latency key for performance
 - Intra rack : 0.95 μ second
 - Inter rack : 1.40 μ second
 - Ping-pong Bandwidth
 - 6.14 GiB/s
 - All numbers measured under full production using OpenMPI

InfiniBand Basics

- TCP/IP over InfiniBand – IPoIB on all node
- Both GbE and IB interfaces, eth0 and ib0 named
- Example
 - Node compute-x-y has two interfaces
 - cx-y is the eth0 interface
 - ib-x-y is the ib0 interface
- All tcp/ip communication works over ib0
 - Example: applications that uses tcp communication between nodes

Semi permanent storage

/work

Common scratch area

Size 218 TiB ($234325432320/2^{30}$)



/cluster - also accomodate /cluster/home

Cluster wide storage for programs, libraries etc and
user data /cluster/home

Size 218 TiB ($234325432320/2^{30}$)

Semi permanent storage

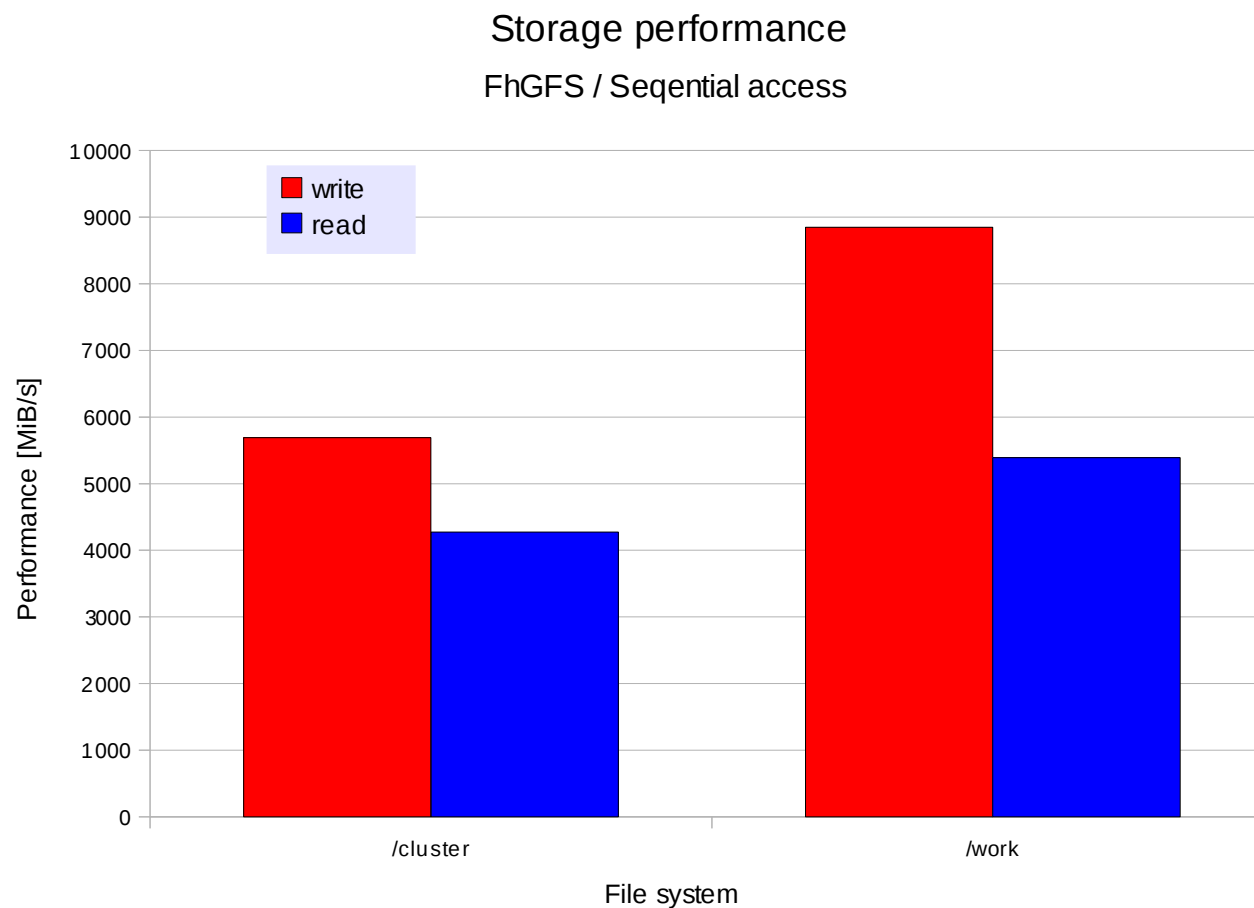
Storage – Supermicro / LSI / Toshiba

10 IO nodes dual socket Intel E5-2609, 64 GiB memory,
dual LSI-9265-8i controller, 33 Toshiba SAS 2TB disks
3 RAID 6 (8+2) sets + 3 hot spare
2 redundant meta data servers, SSD for meta data

Fraunhofer global parallel file system, FhGFS
All nodes connected via FDR InfiniBand



Semi permanent storage



Permanent storage

Only **/cluster/home** is backed up

Backup is quite expensive !

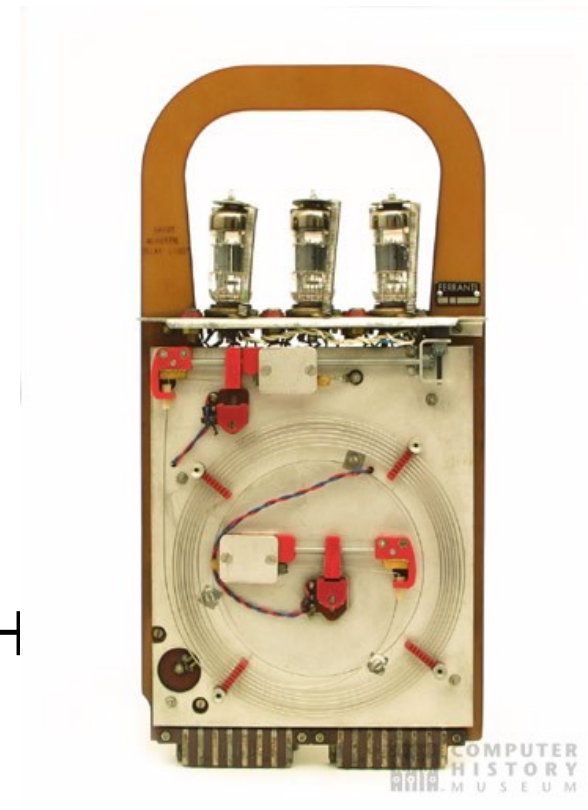
All users have a directort called
nobackup files here are not backed up

Backup cannot handle millions of files, tar them together
and backup the tarball, put the files in a nobackup
directory nor does it like that files vanished....



More on Shared Scratch storage

- Shared scratch space /work
- Global file system FhGFS
 - 200 TiB space
 - High performance
 - Tuned for high concurrency
- No need to spread files
 - A simple copy file to \$SCRATCH
 - See later.....



Scratch space

- Bandwidth
 - Approx 8 GiB/sec sequential
- Metadata
 - 3635 create, fill, touch and delete files per second, metadata benchmark
 - Run metadata benchmark faster than expected
- Cleaned every night, orphan files deleted

Scratch space

- The directory \$SCRATCH is cleared and removed when SLURM job terminates
- The command “savework” no longer exist, all is gone when job is done !
- To copy back files from \$SCRATCH automatically use the script “chkfile” or a copy back to your home dir.
- **Do not run jobs that uses file IO on /cluster (\$HOME)**

Scratch space

- Environment variable SCRATCH still points to a SLURM job unique directory
- The directory \$SCRATCH is now shared and common to all nodes allocated the job
- Performance of \$SCRATCH is far higher than the local SATA disk before
- Do use \$SCRATCH as your work area !

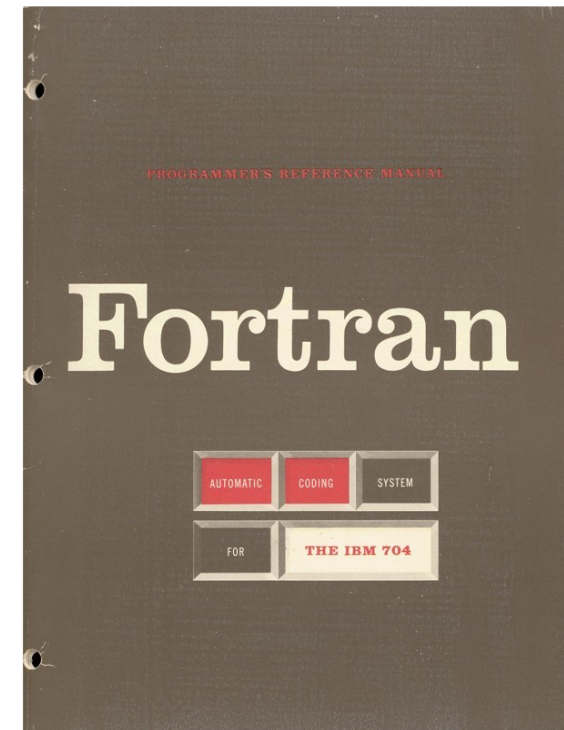
Compilers

Intel compilers Fortran, C and C++

Gnu C, C++ and Fortran compilers

Portland Group C, C++ and Fortran compilers

Open64 C, C++ and Fortran compilers



Compilers - common to all

- Module sets programming environment, example intel
 - F90, F77, FC, FORTRAN = ifort
 - CC = icc
 - CXX = icpc
 - CPATH, FPATH points to include files
 - CXXFLAGS, CFLAGS, FFLAGS to “-O2 -xAVX -mavx”
 - LD_LIBRARY_PATH to library files

Intel set of compilers

Versions 2011, 2013 currently installed

Modules available for all versions

`ifort -help` gives overview of options

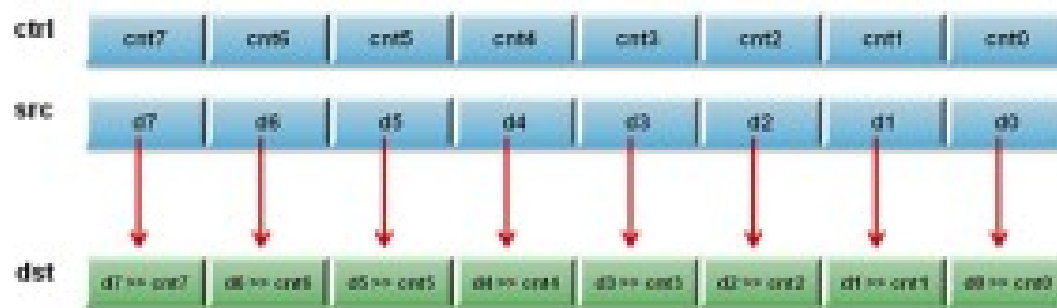
Documentation found at :

<http://software.intel.com/>



Intel compiler

New vector unit in Sandy Bridge

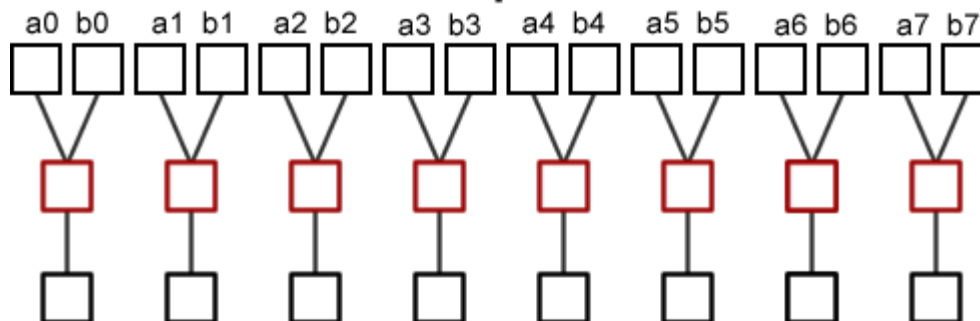


Element with	<code>==</code>	<code>==</code> (Logical)	<code>==</code> (Arithmetic)
DWORD	VPSLLVD	VPSRLVD	VPSRAVD
QWORD	VPSLLVQ	VPSRLVQ	(not implemented)

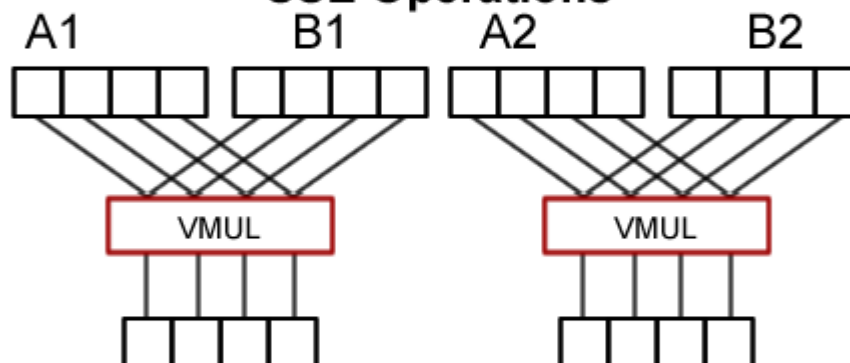
Options for in core parallelization – SIMD

Default by module **-xAVX** and **-mavx**

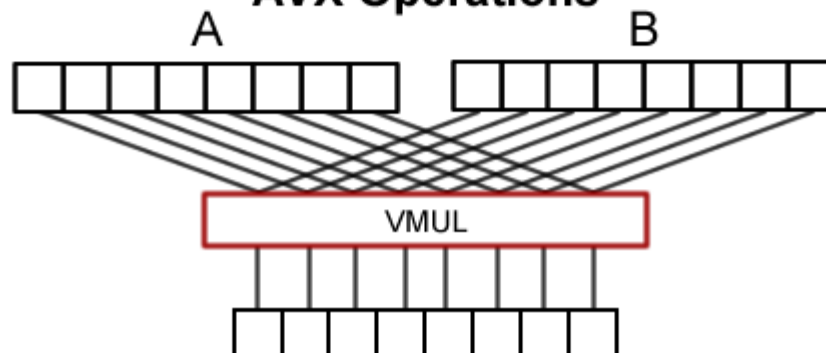
Scalar Operations



SSE Operations



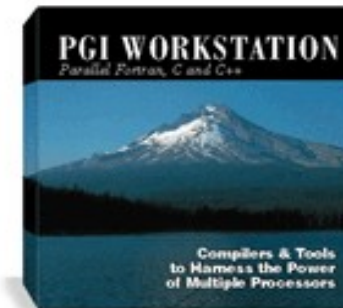
AVX Operations



Portland set of compilers

Version 11.10 currently installed.

Modules sets PGI programming environment
Including C compiler, pgcc



Documentation found at :

<http://www.pgroup.com/resources/docs.htm>



Portland compiler

Intel Sandy Bridge supported with :
-tp=sandybridge-64

pgf90 -help gives overview of flags

Open64 set of compilers

Version 5.0 currently installed

Modules available, open64

`openf90 -help` overview of options

Documentation found at :

<http://www.open64.net/documentation>



Performance libraries

High performance libraries tuned for performance for Intel Sandy Bridge architecture

MATH : Intel libimf, libmv, IPP etc

BLAS : MKL

LAPACK : MKL

SCALAPACK: MKL (also i8 version)

FFT : MKL

Some applications use libm heavily, check what your are using

Performance libraries

- Intel libraries comes often in threaded and sequential versions.
- Be careful when building MPI jobs
 - Do you want a multi or single core lib ?
 - Hybrid model ?
 - OMP_NUM_THREADS controls # cores
- Read the documentation !

MATH – Performance primitives

Intel Integrated Performance Primitives (IPP) is an extensive library of multicore-ready, highly optimized software functions for digital media and data-processing applications.

Intel IPP offers thousands of optimized functions covering frequently-used fundamental algorithms.

Intel IPP functions are designed to deliver performance beyond what optimized compilers alone can deliver.

Performance libraries – gain ?

- IPP exmples:
 - bzip2 compression 2 GiB tar file.
 - Native bzip2 : 213 sec
 - IPP bzip2 : 176 sec
 - gzip compression
 - Native gzip : 467 sec
 - IPP gzip : 231 sec
- module load hpczip to get access

BLAS

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \dots & a_{3n} \\ \vdots & \vdots & \vdots & & \vdots \\ a_{m1} & a_{m2} & a_{m3} & \dots & a_{mn} \end{bmatrix}$$

← second row

↑
third column

→ Intel MKL

Intel Math Kernel Library

`$MKLPATH` and `$MKLROOT`

Often -mkl is enough, single or multi thread ?

→ BLAS source

<http://www.netlib.org/blas/> Fortran 77 code for reference

Now the fun begins – can you beat MKL ?

FFT - libraries

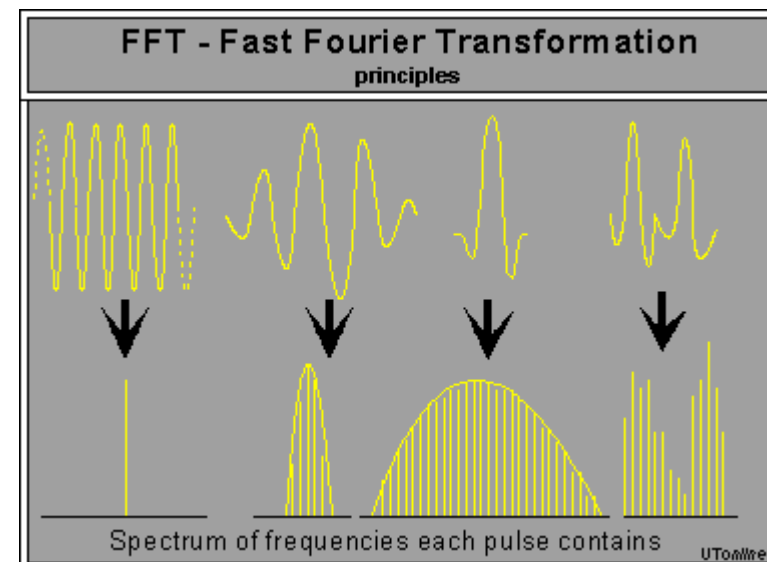
→ FFTW

Commonly compiled by individual users, not installed

→ MKL

MKL contains a range of FFT routines, 1d, 2d and 3
Intel MKL contains interfaces for FFTW

→ See Abel documentation software / libraries



Using OpenMPI

InfiniBand is standard for OpenMPI

Module load openmpi.<fortran compiler>/<version>

Example: `module load openmpi.intel/1.6.1`

Run with mpirun

Example: `mpirun <your executable>`

SLURM queue system

- SLURM batch queue system as before
- Only minor changes to the run scripts required
 - `source /cluster/bin/jobsetup`
 - Max time one week –partition=long for longer jobs, up to 4 weeks
 - `--tmp` removed, /work common, 200TiB
 - `squeue` : `-n` now means name, `-w` nodes, but `–nodes=` is as before