

Athene User's Guide

ParTec Cluster Competence Center GmbH

Version 1.1

February 04, 2009

Table of Contents

Compiling and running applications.....	1
Batch system configuration.....	6

Compiling and running applications

All necessary development tools are installed on the login server (*athene1*) as well as on the compute nodes. Usually, software development will take place on the login server, while running will take place on the compute nodes.

In cases where you need to go to the nodes for development, e.g. when special hardware like InfiniBand is needed, you should do so via the batch system. How to do this will be explained in section .

1.1 Available compilers

The gnu compilers, `gcc`, `g++` and `gfortran` (version 4.1.2) are installed as well as the Intel compilers `icc`, `icpc` and `ifort` version 11.0 build 74. There is also the legacy version 10.1 build 21 available.

For documentation on the Intel compilers see

1. `/opt/intel/Compilers/11.0/074/Documentation`
2. `/opt/intel/cce/10.1.021/doc/main_cls`
3. `/opt/intel/fce/10.1.021/doc/main_for`

You may want to use the `w3m` text browser to view the older Intel documentation. The man pages are available after setting up the right environment. Setting the environment will be explained in section .

1.2 Available MPI implementations

The following implementations of MPI are available:

MPI version	Version	Documentation
ParaStation MPI	5.0.10	/opt/parastation/doc
OpenMPI	1.2.6	/usr/mpi/gcc/openmpi-1.2.6/share/openmpi
Mvapich2	1.0.3	/usr/mpi/gcc/mvapch2-1.0.3/www
Intel MPI	3.2.0.011	/opt/intel/impi/3.2/doc

1.3 Intel MKL version 10.1.1.019

The Intel Math Kernel Library offers an implementation of linear algebra routines (BLAS and LAPACK) as well as routines for fast Fourier Transforms and random number generators. Documentation can be found in /opt/intel/mkl/10.1.1.019/doc.

1.4 ATLAS library version 3.8.2

The ATLAS library is available in /opt/lapack/atlas-3.8.2. It has been compiled using the built-in blas routines. This means that only the most commonly used routines are available.

1.5 Selecting a compiler/MPI combination

There are two ways to select a certain compiler/MPI combination, automatically via the *module* command or manually. The former will be very convenient for most purposes, whereas the latter will give you more freedom.

Selecting a compiler/MPI combination using modules

For automatic setup, the following combinations of modules are available:

Module	Description
parastation-gcc	ParaStation MPI using default GCC
openmpi-gcc	OpenMPI using default GCC
mvapich2-gcc	Mvapich2 using default GCC
impi-gcc	Intel MPI using default GCC
parastation-intel	ParaStation MPI using Intel compilers
openmpi-intel	OpenMPI using Intel compilers
mvapich2-intel	Mvapich2 using Intel compilers
impi-intel	Intel MPI using Intel compilers
intel-11.0	Intel compiler (11.0.076) (recommended)
intel-10.1-icc	Intel C/C++ compiler (10.1.021)

Module	Description
intel-10.1-ifort	Intel Fortran compiler (10.1.021)

Note: Loading a *-intel module mandates that the Intel compiler is specified with a intel-* selection before. E.g. when you want the Intel Fortran compiler together with mvapich2, use the following two commands:

```
athene1:~> module load intel-11.0
athene1:~> module load mvapich2-intel
```

To check the current settings, use the following commands:

```
athene1:~> which ifort
/opt/intel/Compiler/11.0/074/bin/intel64/ifort
athene1:~> module load mvapich2-intel
athene1:~> type mpiexec
mpiexec is aliased to `/opt/parastation/bin/mpiexec'
```

To load a module, use the command

```
athene1:~> module load <module_name>
```

The MPI modules are mutually exclusive, you have to unload them before switching over to another:

```
athene1:~> module unload <module_name>
```

To list the available modules, issue

```
athene1:~> module avail
```

For more information, see `man module` or issue the command `module help`.

Selecting a compiler/MPI combination manually

If you want to set your environment for different development tools manually, you can source the corresponding scripts, given here for *sh* family syntax:

- `./opt/intel/Compiler/11.0/074/bin/ifortvars.sh intel64` (also provides C compilers)
- `./opt/intel/Compiler/11.0/074/bin/iccvars.sh intel64` (also provides FORTRAN)
- `./opt/intel/cce/10.1.021/bin/iccvars.sh`
- `./opt/intel/fce/10.1.021/bin/ifortvars.sh`
- `./opt/intel/impi/3.2/bin64/mpivars.sh`
- `./usr/mpi/gcc/openmpi-1.2.6/bin/mpivars.sh`
- `./usr/mpi/gcc/mvapich2-1.0.3/bin/mpivars.sh`

The equivalents for the *csh* shell family syntax are:

- `source /opt/intel/Compiler/11.0/074/bin/ifortvars.csh intel64` (loads C as well)
- `source /opt/intel/Compiler/11.0/074/bin/iccvars.csh intel64` (loads FORTRAN as well)

- `source /opt/intel/cce/10.1.021/bin/iccvars.csh`
- `source /opt/intel/fce/10.1.021/bin/ifortvars.csh`
- `source /opt/intel/impi/3.2/bin64/mpivars.csh`
- `source /usr/mpi/gcc/openmpi-1.2.6/bin/mpivars.csh`
- `source /usr/mpi/gcc/mvapich2-1.0.3/bin/mpivars.csh`

In case of the Intel version 11.0 compilers, it is important to specify the argument *intel64*. Leaving it out will produce an error message like *ERROR: Unknown switch ". Accepted values: ia32, intel64, ia64*.

Be aware however that you are responsible for avoiding conflicts between different versions.

1.6 Using the ParaStation compiler wrappers

To compile a MPI program with ParaStation MPI, it is most convenient to use the wrappers around the compilers as suggested in the MPI standard. These will take care that the right header files are included, the correct libraries are linked, etc. The following wrappers exist for FORTRAN77, FORTRAN90, C and C++:

Wrapper	Switch	Environment variable	Compiler (default)
<code>/opt/parastation/mpi2-ch3/bin/mpif77</code>	<code>-fc</code>	<code>MPICH_F77</code>	<code>gfortran</code>
<code>/opt/parastation/mpi2-ch3/bin/mpif90</code>	<code>-f90</code>	<code>MPICH_F90</code>	<code>gfortran</code>
<code>/opt/parastation/mpi2-ch3/bin/mpicc</code>	<code>-cc</code>	<code>MPICH_CC</code>	<code>gcc</code>
<code>/opt/parastation/mpi2-ch3/bin/mpicxx</code>	<code>-CC</code>	<code>MPICH_CCC</code>	<code>g++</code>

There are two ways to control which actual compiler is invoked by the wrapper: via command line switch or via environment variable. In absence of the environment variables `MPICH_F77`, `MPICH_F90`, `MPICH_CC` and `MPICH_CCC`, the compilers *gfortran*, *gfortran*, *gcc* and *g++* are invoked respectively. Specifying the compiler is thus necessary only if a non-default is chosen. Help on the wrapper can be obtained by using the `-help` command line option. (Using the `-help` command option usually calls the usage information for the actual compiler.)

Should you prefer using the compiler directly and not the wrapper, you can find out the compile and link flags using the `-show` option, e.g.

```
athene1:~> /opt/parastation/mpi2/bin/mpif77 -fc=ifort -show
ifort -I/opt/parastation/mpi2/include -L/opt/parastation/mpi2/lib -Wl,-
rpath -Wl,/opt/parsastation/mpi2/lib -lmpich -lpthread -lrt
```

As an example you can try to compile this little hello world program:

```

/* hello_mpi.c */
#include <stdio.h>
#include <mpi.h>

int main (int argc, charg **argv)
{
    int rank, size;

    MPI_Init (&argc, &argv);          /* starts MPI */
    MPI_Comm_rank (MPI_COMM_WORLD, &rank); /* get current process id */
    MPI_Comm_size (MPI_COMM_WORLD, &size); /* get number of processes */
    printf ( Hello world from process %d of %d\n , rank, size);
    MPI_Finalize ();
    return;
}

```

Compile it with this command:

```
athene1:~> mpicc -o hello_mpi hello_mpi.c
```

1.7 Using the OpenMPI compiler wrappers

The following wrappers are available:

Wrapper	Switch	Environment variable	Compiler (default)
/usr/mpi/gcc/openmpi-1.2.6/bin/mpif77	N/A	OMPI_F77	gfortran
/usr/mpi/gcc/openmpi-1.2.6/bin/mpif90	N/A	OMPI_FC	gfortran
/usr/mpi/gcc/openmpi-1.2.6/bin/mpicc	N/A	OMPI_CC	gcc
/usr/mpi/gcc/openmpi-1.2.6/bin/mpicxx	N/A	OMPI_CXX	g++

Please note that no switches are available for the selection of the compiler and that this will take place via environment variables.

The options `-showme:compile` and `-showme:link` will output the compilation and linking flags in case you prefer a more manual approach to compilation.

Other than that, the usage of these wrappers are pretty much identical to the ParaStation ones.

1.8 Running your code

To run your code, you should use the queuing system in order to make good use of the computing resources. You can use the batch system to start interactive jobs as well as true batch jobs without user intervention.

An interactive job can be started like this:

```
athene1:~> qsub -I -l nodes=4:ppn=8,cput=24:00:00,walltime=24:00:00
```

which would start a shell on the first available node with 8 cores (processors per node, or, ppn for short) and reserves three other machines for your job. Additionally, you ask for 24 hours of wall time here. You then can start your job as you would from any other interactive environment (you have set up the environment to select the appropriate mpiexec before):

```
node0123:~> mpiexec -np 32 ./hello_mpi
```

If you are unsure which mpiexec you are using, you can verify it by typing

```
node0123:~> type mpiexec
mpiexec is aliased to `/opt/parastation/bin/mpiexec'
```

While interactive jobs are convenient for initial tests and debugging, you may want to automatize the process with a script. For the above, you could write the following little script:

```
#!/bin/bash
#PBS -l nodes=4:ppn=8
#PBS -l cput=24:00:00
# select your MPI flavour here, either by using modules
# or by sourcing the appropriate files.
# e.g.
module load parastation-gcc

cd ${PBS_O_WORKDIR}
mpiexec -np 32 ./hello_mpi

# end of this script
```

Assuming this script is called `batch.sh` and that you're using PBS/Torque to provide `PBS_O_WORKDIR`, submit it with

```
athene1:~> qsub batch.sh
```

(you don't need to change the permissions) into the queueing system. Mind that since all the directives are given inside the script via the special `#PBS` comment, you don't need command line parameters here.

For applications built with MPI implementations specifying the execution hosts via a machine file, the batch system provides the environmental variable `PBS_NODEFILE`. This variable points to a file listing all the nodes reserved in this job. A typical use would look like this:

```
mpiexec -np 32 -machinefile ${PBS_NODEFILE} ./hello_mp
```

Note: this is not required using the ParaStation `mpiexec` command, as this version automatically honors the `PBS_NODEFILE` environment variable.

Batch system configuration

The batch system configuration reflects the sponsoring situation in that certain queues are linked to certain machines owned by their project. There is also a default queue which is open to everyone and two maintenance queues. The maintenance queues embed all machines but are accessible to

maintenance users only. Currently, jobs may last 3 days (72 hours) and use up to 16 nodes.

The following queues are available:

Queue	Nodes	Max. wall time	Max # of nodes	Remarks
common	1-108, 144-176	72:00:00	16	general queue for all users
striegnitz	109-122	72:00:00	14	project Striegnitz
spang_i	123-134	72:00:00	12	IB nodes for project Spangler
spang_e	769-776	72:00:00	8	Ethernet nodes for project Spangler
merkl	135-143	72:00:00	9	project Merkl
bocola	777-779	72:00:00	3	project Bocola
parallel	all	180:00:00	176	maintenance, tests
serial	all	180:00:00	1	maintenance, tests

Users which have an `acl_users` entry in the queue definition should get their jobs routed automatically to their queue. They can select the common queue by specifying the `-q common` switch in the `qsub` command line. Normal users however are not allowed to use a project queue, however.