# Best Practice mini-guide "Hydra"

IBM iDataPlex HPC System at RZG

Content by:
RZG staff

Compiled from: http://www.rzg.mpg.de/computing

Editor: Maciej Szpindler
ICM University of Warsaw

For updates please check the RZG web site

March 2013

# Table of Contents

# 1. Introduction

In September 2012, the first phase of the new Intel-based IBM iDataPlex supercomputer was installed at the RZG. This first phase consists of an Intel Sandy Bridge-EP based cluster with 610 nodes and a fast InfiniBand FDR14 interconnect. The main system (a PFlop/s class system) will be installed by mid 2013.

Each Sandy Bridge compute node has 16 cores (32 logical cpus in SMT/Hyperthreading mode) and 64 GB of main memory. 20 nodes have a main memory of 128 GB. Thus, in total there are 9792 cores with a main memory of 40 TB and a peak performance of about 200 TFlop/s.

Additionally, there are 26 I/O nodes to serve the 5 PetaByte of disk storage.

The name of the IBM iDataPlex HPC system is *Hydra*.

**Figure 1. Hydra compute racks, source: Rechenzentrum Garching**

# 2. System architecture and configuration

## 2.1. System configuration

The current system configuration includes:

- 610 compute nodes
- 2 nodes for login and application development
- 26 I/O nodes
- 5 PB of online disk space
- fast InfiniBand FDR14 (4 x 14 Gb/s) network to connect the nodes

Node-specific information:

- Processor type: Intel Xeon Sandy Bridge-EP
- Processor clock: 2.6 GHz
- Peak performance per core: 20.5 GFlop/s
- Cores per node: 16 (each with 2 hyperthreads, so there are 32 logical CPUs per node)
- Main memory of the compute nodes: 590 x 64 GB, 20 x 128 GB
- Main memory of the login and development nodes: 2 x 192 GB

## Operating system

Operating system on Hydra is SuSe Enterprise Linux (SLES) 11.

# 2.2. Filesystems

`$HOME` Your home directory is in the GPFS file system `/u` (see below).

`AFS` AFS is only available on the login node `hydra` (`hydra01i`) and on `hydra02i` in order to access software that is distributed by AFS. If you don't get automatically an AFS token on `hydra` and `hydra02` during login, you can get an AFS token with the command **/afs/ipp/@sys/bin/klog**. There is no AFS on the compute nodes, so you have to avoid any dependencies on AFS in your job script.

If you get the message:

```
-------------------------------------------------------
LoadL_starter: 2512-906 Cannot set user credentials.
LoadL_starter: AFS token(s) were captured for the job step,
however AFS was not found running on this node
-------------------------------------------------------
```

in your job log files, you can ignore it, it's not an error message but only for information.

`GPFS` There are two global, parallel file systems of type GPFS, symmetrically accessible from all *Hydra* cluster nodes:

`/u` (a symbolic link to `/hydra/u`)

for permanent user data (source files, config files, etc.)

0.75 PB, RAID 6, no system backups

Your home directory is in `/u`. If you need a private shell profile, you have to provide it by yourself.

The default disk quota in `/u` is 2.5 TB. You can check your disk quota in `/u` with the command: **/usr/lpp/mmfs/bin/mmlsquota u**

`/ptmp` (a symbolic link to `/hydra/ptmp`) for batch job I/O.

3.8 PB, RAID 6, no system backups

Files in `/ptmp` that have been not accessed for more than 12 weeks will be removed automatically. The period of 12 weeks may be reduced if necessary. For interactive data analysis and visualization the "viz" cluster should be used on which the file system is mounted as `/hydra/ptmp`.

As a current policy, no quotas are applied on `/ptmp`. This gives users the freedom to manage their data according to their actual needs without administrative overhead. This liberal policy presumes a fair usage of the common file space. So, please do a regular housekeeping of your data and archive/remove files that are not used actually.

Archiving data from the GPFS file systems to tape can be done with ADSM/TSM. Moreover, you can use the migrating file system `/r` (see below) to save your files that were created on the HPC machines at RZG.

`/r` (a symbolic link to `/ghi/r`)

for migrated data, available only on the login node `hydra` (`hydra01i`) and on `hydra02i`.

Each user has a subdirectory `/r/<initial>/<userid>` to store his/her data. Files should be packed to tar, cpio or zip files (with a size of 1 - 500 GByte) before archiving them in /r. When the file system gets full above a certain value, files will be transferred from disk to tape, beginning with the largest files which have been unused the longest time.

If you access a file which has been migrated to tape, the file will automatically be transferred back from tape to disk. This of course implies a delay. You can manually force the recall of a migrated file by using any command which opens the file. You can recall in advance all files needed by some job with a command like **file myfiles/***

You can see which files are resident on disk and which ones have been migrated to tape with the command **ghi_ls** (located in `/usr/local/bin`), optionally with the option `-l`. Here is a sample output:

```
hydra01% ghi_ls -l
G -rw-r--r--   1  ifw  rzs           22 Nov 21 15:12 a1
H -rw-------   1  ifw  rzs  138958551040 Sep 18 22:22 abc.tar
H -rw-r--r--   1  ifw  rzs    1073741312 May 06 2009  core
G -rw-r--r--   1  ifw  rzs             0 Jun 20 2008  dsmerror.log
B -rw-r--r--   1  ifw  rzs    1079040000 Aug 03 2010  dummyz3
```

The first column states where the file resides: a '`G`' means the file is resident on disk; a '`H`' means the file has been transferred to the underlying HPSS archiving system, probably on tape; a '`B`' means premigrated to tape (the file has already been copied to HPSS but is still present on disk and can be removed immediately if the system needs to free disk space).

### Please note

If you want to "tar" files that are alreaddy located in `/r`, please carefully check the contents of the resulting TAR file if all migrated files were correctly retrieved and included into the TAR file. Don't use "gzip" or "compress" on files that are already located in `/r`. It's not necessary because all files are automatically compressed by hardware as soon as they are written to tape.

### Further reading

For more information about `/r` see:

- HSM (HPSS) for users of the IBM High-Performance Computer [http://www.rzg.mpg.de/datastorage/tsm/adsm_PSI_qa.html]

`/tmp` Please, *DON'T* use the file system `/tmp` for your scratch data. Instead, use `/ptmp` which is accessible from all *Hydra* cluster nodes, and set the environment variable `TMPDIR` to `/ptmp/<userid>` in your job scripts.

## Further details

For more information related to Section 2, “System architecture and configuration” please refer to the RZG website:

- Configuration of the IBM iDataPlex HPC system (Hydra) [http://www.rzg.mpg.de/computing/hardware/Hydra/configuration]

- File systems on the IBM iDataPlex HPC system (Hydra) [http://www.rzg.mpg.de/computing/hardware/Hydra/filesystems]

# 3. System Access

## 3.1. Remote access

For security reasons, direct login to the IBM iDataPlex HPC cluster *Hydra* is allowed only from within the MPG networks. Users from other locations have to login to `gate.rzg.mpg.de` first. Use ssh to connect to Hydra:

**ssh hydra.rzg.mpg.de**

You always have to provide your (Kerberos) password on the Hydra login nodes, SSH keys are not allowed.

Secure copy (scp) can be used to transfer data to or from `hydra.rzg.mpg.de`.

The SSH key fingerprints for `hydra.rzg.mpg.de` are:

```
1024 8d:3c:4b:a2:9e:15:fa:15:3e:ae:b2:9b:dc:99:f7:91 (RSA)
1024 6d:c4:11:9d:b5:8c:53:79:8f:a4:dd:3b:7a:9a:ba:02 (DSA)
```

## 3.2. Using compute resources on Hydra

The login node `hydra.rzg.mpg.de` is mainly intended for editing, compiling and submitting your parallel programs. Interactive usage of the Parallel Operating Environment (POE) on the login node is not enabled. Test or production jobs have to be submitted to the LoadLeveler batch system which reserves and allocates the resources (e.g. compute nodes) required for your job. Short test jobs (runtime less than 15 min) which are requiring 1, 2, 4 or 8 cores will run on a dedicated node with short turn around times.

## 3.3. Interactive debug runs

If you need to debug your program code you may login to the node `hydra02i.rzg.mpg.de` and run your code interactively (2 hours at most).

But please, take care that the machine does not become overloaded. Don't occupy all the 16 cores and please do not request more than 30 GB of main memory. Neglecting these recommendations may cause a system crash or hangup!

### Further details

For more information related to Section 3, “System Access” please refer to the RZG website:

- Access to the IBM iDataPlex HPC system (Hydra) [http://www.rzg.mpg.de/computing/hardware/Hydra/access-to-the-ibm-HPC-system]

# 4. User environment and programming

## 4.1. Batch System

The login node `hydra.rzg.mpg.de` of the iDataPlex HPC system is intended mainly for editing and compiling your parallel programs. Interactive usage of **poe/mpirun** is not allowed on the login node `hydra.rzg.mpg.de`. To run test or production jobs, submit them to the LoadLeveler batch system, which will find and allocate the resources required for your job (e.g. the compute nodes to run your job on).

Short test jobs ( shorter than 15 min) with 2, 4 or 8 cores will run on a dedicated node with short turn around times.

By default, the job run limit is set to 3 on *Hydra*. If your batch jobs can't run independently from each other, please use job steps or contact the helpdesk on the RZG web page.

In principle, you can run your old job scripts from the Power6 cluster without major changes, except that you have to omit any statement like

```
# @ requirements = (Arch == "Power6") && (OpSys >= "AIX53")".
```

This is not valid on the Intel/Linux platform of *Hydra*.

The Intel processors support the "Hyperthreading / Simultaneous Multithreading (SMT)" mode which might increase the performance of your application ny up to 20%. With hyperthreading, you have to increase the number of `tasks_per_node` from 16 to 32 in your job script. Please be aware that with 32 `tasks_per_node` each process gets only half of the memory by default. If you need more memory per process you have to specify it in the variable `ConsumableMemory`. In the *Hydra* cluster, there are 20 compute nodes available with 128 GB of real memory (120 GB for the application). So, on these 20 nodes, you can specify `ConsumableMemory(3800mb)` for MPI jobs with hyperthreading or `ConsumableMemory(7600mb)` for MPI jobs without hyperthreading.

The default Parallel Environment on *Hydra* is POE with the IBM MPI. But you may use Intel MPI as well. You can use executables that were built with Intel MPI in the poe call in your job script. Or, you can use a pure Intel MPI environment in your job. However, we recommend to use IBM's MPI/POE because it shows somewhat better performance than Intel MPI.

Since the upgrade to LoadLeveler 4.1 the graphical user interface **xloadl** is no longer supported by IBM.

### Further reading

- For more information on general LoadLeveler usage please refer to the PRACE Generic x86 Best Practice Guide [http://www.prace-ri.eu/IMG/pdf/Best-Practice-Guide-Generic-x86.pdf].
- For detailed information about LoadLeveler, please see IBM's manual about Using and Administering IBM LoadLeveler for Linux [http://publib.boulder.ibm.com/infocenter/clresctr/vxrx/index.jsp].

## 4.2. Compilers

The Intel Fortan (`ifort`) and C/C++ (`icc, icpc`) compilers are the default compilers on the HPC cluster *Hydra* and are provided automatically at login (see the output of **module list** for details on the version).

To compile and link MPI codes using Intel compilers, use the commands `mpiifort, mpiicc or mpiicpc`, respectively.

The GNU compiler collection (`gcc, g++, gfortran`) is available as well. A default version comes with the operating system (SLES 11). More recent versions can be accessed via environment modules. To compile and link MPI codes using GNU compilers, use the commands `mpicc, mpic++, mpif77 or mpif90`, respectively.

Invoke the command **module avail** to get an overview on all compilers and versions available on Hydra.

### Modules environment

- For more information on general *Modules* usage please refer to the PRACE Generic x86 Best Practice Guide [http://www.prace-ri.eu/IMG/pdf/Best-Practice-Guide-Generic-x86.pdf].

## 4.3. Parallel Programming

On the Hydra HPC cluster two basic parallelization methods are available.

MPI

The Message Passing Interface provides a maximum of portability for distributed memory parallelization over a large numer of nodes.

| | |
|---|---|
| OpenMP | OpenMP is a standardized set of compiler directives for shared memory parallelism. On Hydra a pure OpenMP code is restricted to run on a single node. |

It is possible to mix MPI and OpenMP parallelism within the same code in order to achieve large scale parallelism.

# 4.4. Parallel MPI applications

By default, the Intel compilers for Fortran/C/C++ and the IBM MPI implementation are available on Hydra.

The MPI wrapper executables are mpiifort, mpiicc and mpiicpc. These wrappers pass include and link information for MPI together with compiler-specific command line flags down to the Intel compiler.

# IBM Parallel Operating Environment with IBM MPI

Hydra is using IBM's Parallel Operating Environment (POE) for executing and managing MPI jobs.

To run a MPI program interactively (for debug purposes) please follow these steps:

- Compile your program, e.g.

  **mpiifort myprog.f -o myprog**

- Create a file named `host.list` with a line "localhost" for each processor you intend to use. To use, say, 4 processors of a node, there are four lines with "localhost" (without quotes) in `host.list`.

- Run your program on the node `hydra02` using the command

  **poe ./myprog -procs 4**

### Note

Remember that poe's options must appear after the name of your binary.

## Batch jobs

For production runs it is necessary to run the MPI program as a batch job. Please see Section 4.1, “Batch System” for further information.

## Environment variables

There are many environment variables and command line flags to the 'poe' command which influence the operation of the PE tools and the execution of parallel programs. Useful defaults are set on Hydra. A complete list of these environment variables can be found in the poe documentation.

### Further reading

For more detailed information on POE environment please refer to the poe documentation [http://publib.boulder.ibm.com/epubs/pdf/c2366673.pdf].

# 4.5. Multithreaded (OpenMP or hybrid MPI/OpenMP) applications

To compile and link OpenMP applications pass the flag -openmp to the Intel compiler.

In some cases it is necessary to increase the private stack size of the treads at runtime, e.g. when threaded applications exit with segmentation faults. On Hydra the thread private stack size is set via the environment variable

KMP_STACKSIZE. The default value is 4 megabytes (4m). For example, to request a stack size of 128 megabytes on Hydra, set KMP_STACKSIZE=128m.

### Note

For information on compiling applications which use pthreads please consult the Intel compiler documentation [http://software.intel.com/en-us/articles/intel-c-composer-xe-documentation/#lin].

# 4.6. Using MKL mathematical library on Hydra

## Intel Math Kernel Library overview

The Intel Math Kernel Library (MKL) is provided on Hydra. For users migrating their code from POWER6 it replaces IBM ESSL and pESSL. MKL provides highly optimized implementations of (among others)

- LAPACK/BLAS routines,
- direct and iterative solvers,
- FFT routines,
- ScaLAPACK.

Parts of the library support thread or distributed memory parallelism.

### Note

Extensive information on the features and the usage of MKL is provided by the official Intel MKL documentation [http://software.intel.com/en-us/articles/intel-math-kernel-library-documentation/].

## Linking programs with MKL

By default, an MKL environment module is already loaded. The module set the environment variables `MKL_HOME` and `MKLROOT` to the installation directory of MKL. These variables can then be used in makefiles and scripts.

The Intel MKL Link Line Advisor is often useful to obtain information on how to link programs with MKL. For example, to link statically with the threaded version of MKL/10.3 on Hydra (Linux, Intel64) using standard 32 bit integers, pass the following command line arguments to the Intel compiler:

```
-Wl,--start-group
$(MKLROOT)/lib/intel64/libmkl_intel_lp64.a
$(MKLROOT)/lib/intel64/libmkl_intel_thread.a
$(MKLROOT)/lib/intel64/libmkl_core.a
-Wl,--end-group -lpthread -lm -openmp
```

Select MPICH2 in the Link Line Advisor in case you intend to use MPI parallel routines from MKL on Hydra using IBM MPI.

# 4.7. Numerical Libraries

## NAG

Collection of numerical algorithms for HPC. Different sequential versions of the NAG C, Fortran77, and Fortran90 library for various compilers are available. See module avail, module help nag_clib and module help nagf90lib for available versions, documentation and detailed usage instructions.

### WSMP

Watson Sparse Matrix Package - A high performance shared- and distributed-memory parallel sparse linear equation solver. See **module avail**, **module help wsmp**.

### FFTW

Aka the "Fastest Fourier transforms in the West/World". FFTW is a C library for computing the discrete Fourier transform (DFT) in one or more dimensions, of arbitrary input size, and of both real and complex data (as well as of even/odd data, i.e. the discrete cosine/sine transforms or DCT/DST). Latest versions of FFTW2 and FFTW3 are available as a module.

### PETSc

A suite of data structures and routines for the scalable (MPI parallel) solution of scientific applications modeled by partial differential equations. Available as a module.

### SLEPc

Library for the solution of large scale sparse eigenvalue problems on parallel computers. Available as a module.

### GSL

The GNU Scientific Library (GSL) is a numerical library for C and C++ programmers (the FGSL FORTRAN addon interface is installed). GSL provides a wide range of mathematical routines such as random number generators, special functions and least-squares fitting. Available as a module.

## 4.8. I/O Libraries

### HDF5

HDF5 is a data model, library, and file format for storing and managing data. It supports an large variety of datatypes, and is designed for flexible and efficient I/O and for high volume and complex data. HDF5 is portable and is extensible, allowing applications to evolve in their use of HDF5. The HDF5 Technology suite includes tools and applications for managing, manipulating, viewing, and analyzing data in the HDF5 format. Serial (to use the tools like h5ls, h5dump interactively on the login nodes) and parallel (to build applications using the library API) variants of HDF5 are available as a module.

### NetCDF

NetCDF is a set of software libraries and self-describing, machine-independent data formats that support the creation, access, and sharing of array-oriented scientific data. Available as a module.

## 4.9. Miscellaneous Libraries

- Boost: The Boost C++ libraries provide many classes and routines for various applications.
- TBB: Intel Threading Building Blocks. TBB enables the C++ programmer to integrate (shared memory) parallel capability into the code.
- IPP: Intel Integrated Performance Primitives (IPP). The IPP API contains highly optimized primitive operations used for digital filtering, audio and image processing.
- PAPI: The Performance Application Programming Interface is a library for reading performance event counters in a portable way.
- Perflib: A simple library deleveloped by RZG for manual performance instrumentation.

## 4.10. Further hints for porting

### Environment modules

- Some names of some environment modules have changed. Please use **module avail** on Hydra to obtain a comprehensive list of environment modules.
- Only the latest/current version (as of November 2012) of software packages are offered via the module environment.

### Default Fortran data type size

Some Fortran programmers use e.g. 32 bit floats in the source code (`REAL`) and map these to 64 bit floats (`REAL*8`) at compile time. The corresponding compiler flag for the Intel Fortran compiler is `-r8` (which replaces the XL compiler flag `-qautodbl=dbl4` known to POWER6 users).

### Further details

For more information related to Section 4, “User environment and programming” please refer to the RZG website:

- Getting started on Hydra [http://www.rzg.mpg.de/computing/hardware/Hydra/migration-and-porting-hints]
- Batch system on Hydra [http://www.rzg.mpg.de/computing/hardware/Hydra/batch-system]
- Software available on Hydra [http://www.rzg.mpg.de/computing/hardware/Hydra/software]
- Parallel programming on Hydra [http://www.rzg.mpg.de/computing/hardware/Hydra/parallel.html]
- Libraries available on Hydra [http://www.rzg.mpg.de/computing/hardware/Hydra/libraries.html]

# 5. Programming Tools

To access the software described below please use the module command (**module avail, module load**).

## 5.1. Debugging

- Compiler options: Compilers usually have some debugging features which allow e.g. to check violations of array boundaries. Please consult the compiler's manual pages and documentation for details.
- Forcheck is a tool for the static analysis of Fortran programs.
- gdb, the GNU debugger.
- Totalview is a tool for debugging parallel applications.
- Intel Inspector enables the debugging of threaded applications.
- Marmot is a tool for debugging MPI communication.

## 5.2. Profiling and Performance Analysis

- perflib is a library developed at RZG which provides a simple API for performing time measurements of code regions.
- Intel VTune/Amplifier is a powerful tool for analyzing the single core performance of a code.

- gprof, the GNU profiler.
- Intel Trace Analyzer and Collector is a tool for profiling MPI communication.
- Scalasca enables the analysis of MPI/OpenMP/hybrid codes.

## Further reading

For more information related to Section 5, “Programming Tools” please refer to the RZG website:

- http://www.rzg.mpg.de/computing/hardware/Hydra/tools.html