Best Practice mini-guide "Avitohol"

HP

Content by:
IICT-BAS staff

Compiled from: http://www.hpc.acad.bg/documentation

Editor: Emanouil Atanassov
IICT-BAS

For updates please check http://www.hpc.acad.bg
March 2016

Table of Contents

1. Introduction

In June 2015, the new supercomputer was deployed at IICT-BAS. The cluster consists of 150 HP Cluster Platform SL250S GEN8 servers.
Each compute node had 2 Intel Xeon CPU E5-2650 v2 @ 2.6 GHz (8 cores each), 64GB RAM and 2 Intel Xeon Phi 7120P coprocessors (with 61 cores each). Thus, in total there are 20700 cores.

Additionally, there are 4 I/O nodes to serve the 96 TB of disk storage.
The name of the system, deployed by HP, is *Avitohol*.

Avitohol cluster currently has the following capabilities: theoretical peak performance of 412.32 TFlop/s, total memory 9600 GB and total disk storage of 96 TB.

It is ranked at 389th place on the November 2015 Top 500 list (http://www.top500.org) with LINPACK performance of 264.2 TFlop/s.

Figure 1. Avitohol HPC system

2. System architecture and configuration

2.1. System configuration

The current system configuration includes:

- 150 dual-socket nodes HP ProLiant SL250S Gen8
- 1 of them reserved for login/job submission and application development
- 4 I/O nodes HP ProLiant DL380p Gen8 with 2 Intel Xeon CPU E5-2650 v2, 64GB RAM, Fibre Channel cards
- 2 management nodes HP ProLiant DL380p Gen8 with 2 Intel Xeon CPU E5-2650 v2, 64GB RAM
- 96 TB of online disk space, connected with Fibre Channel with the I/O nodes.
- Fully non-blocking 56Gbps FDR InfiniBand network interconnecting all the nodes above

Configuration of the computing nodes:

- Processors: dual Intel Xeon 8-core CPU E5-2650 v2 @ 2.6 GHz,
- Coprocessors: two Intel Xeon Phi 7120P coprocessors, 16 GB RAM and 61 cores each one
- Main memory of the compute nodes: 64 GB (9.6 TB in total)
- Memory of the accelerators: 16 GB (4.8 TB in total)

Operating system

Operating system on Avitohol is Red Hat Enterprise Linux release 6.7.

The coprocessors run MPSS version 3.6-1.

2.2. Filesystems

The Avitohol system is relatively new. That is why currently the only filesystem that is read-write for all users is the `/home` filesystem, which is of type Lustre. Groups of users may request to have a shared directory so that they can share data between them.

Some software that is made available to all users is available under `/opt` and is typically deployed using environment modules. This filesystem is shared via NFS and is read-only for users.

Currently the Lustre filesystem is provided from only two OSTs. This effectively means that one big file may be served from two storage servers, if this file is created in appropriate way.

To convert a directory to “stripe” files in this way one can do the following:

```
lfs setstripe <filename|dirname> --count 2
```

to set file or directory to use striping with 2 OSTs in parallel.

Currently there only setting the count to two may be advisable if high I/O rates are expected for files in a given directory.

The Lustre filesystem is optimized for processing of large files. This means that creating large amount of small files may be relatively slow process. However, one can use the `/dev/shm` filesystem on each node, which is in the RAM, if fast processing of high number of small files is necessary. This filesystem is not shared between nodes and does not survive restarts! For most users its use is not needed.

Further reading

For more information about the Lustre filesystem, consult Lustre documentation. See, e.g., https://wiki.hpdd.intel.com/display/PUB/HPDD+Wiki+Front+Page

Further details

Applications with high I/O requirements can request dedicated storage.

3. System Access

3.1. Remote access

The gateway node for accessing *Avitohol* is the node **gw.avitohol.acad.bg**. The key fingerprints are currently:

1024 cd:97:16:41:2c:cd:3d:59:c4:f0:d2:67:ce:1c:5e:62 /etc/ssh/ssh_host_dsa_key.pub (DSA)

2048 b9:e3:6f:f5:a2:ea:8f:95:0a:96:5b:55:03:0c:bf:e9 /etc/ssh/ssh_host_rsa_key.pub (RSA)

For security reasons it is advisable for users to use keys with at least 2048 bits.

Access can be done with ssh protocol.
The gateway node has identical hardware and software configuration (as much as possible) with the execution nodes.
Only this node can be used for launching jobs.
Once inside the cluster, the user can login with ssh to any of the execution nodes where he or she has a running job or to the Xeon Phi accelerators that are associated with such nodes.

3.2. Using compute resources on Avitohol

The login node **gw.avitohol.acad.bg** is mainly intended for editing, compiling and submitting compute-intensive programs. Some lightweight testing of parallel programs is allowed. In case of doubt, users are encouraged to submit a job requesting one whole compute node and use it for debugging and development.
The execution nodes are named **sl001-sl150.**

3.3. Interactive debug runs

If you need to debug your program code you may login to the node **gw.avitohol.acad.bg** and run your code interactively. This node has two CPUs and two Xeon Phi coprocessors in the same way as the execution nodes.

Please monitor resource utilization and in case the system becomes overloaded (as can be seen from running **top**) cancel the debug runs and perform them via job submission on an execution node, where you will get dedicated access. Users that cause system crashes of the node or system crashes or hangs on the Xeon Phi coprocessors should try to resolve the issues with their codes using execution nodes allocated by the batch system. If the problems persist they are encouraged to contact system administrators.
The support email is:
avitohol-support@parallel.bas.bg.

Further details

There are other ways of accessing storage from the system, which do not concern most users. If your storage needs exceed one terabyte, please describe them fully in the access form. Access to different interfaces can be provided based on evaluation of these needs.

The Xeon Phi coprocessors are visible as **sl001-mic0** to **sl150-mic0** and **sl001-mic1** to **sl150-mic1**. Users are expected to use them only after obtaining exclusive access to the corresponding execution node.

4. User environment and programming

4.1. Batch System

The login node **gw.avitohol.acad.bg** of the Avitohol HPC system is intended mainly for editing and compiling of parallel programs. Interactive usage of **mpirun/mpiexec** is allowed on the login node and its coprocessors, although it should not be necessary. It is advisable to switch to using a dedicated node obtained through the batch system if such parallel runs take more than a few minutes or use lots of memory.

To run test or production jobs, submit them to the torque 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 with short turn-around times. However, the use of partial nodes (i.e., less than 16 cores from one node) is discouraged. The use of multiple partial nodes, e.g., by requesting 10 nodes with 4 cores, is strongly discouraged.

By default, the job run limit is set to 24 hours on Avitohol. If your batch jobs can't run independently from each other, please use job steps or contact the system administrators helpdesk using the email address
avitohol-support@parallel.bas.bg.

The Intel processors support the "Hyperthreading / Simultaneous Multithreading (SMT)" mode which might increase the performance of your application by up to 20%. However, we leave to the users to decide whether to use Hyperthreading or not. Currently the execution nodes are declared to have 16 CPU cores in torque, although the Linux OS sees 32 (logical) cores.
It is encouraged for users to request multiples of full nodes, e.g., by adding

```
#PBS -l nodes=100:ppn=16
```

in the PBS script you can ask for 100 dedicated nodes.

With hyperthreading, you have to increase the number of actual MPI processes or OpenMP threads to 32 instead of 16 for each node.
However, it is best to first measure the performance on a realistic benchmark problem to see if hyperthreading is actually useful. In both cases, with and without hyperthreading, the request for nodes to PBS will be the same, but the mpirun command line would change.
For example, one can compare the performance of the same executable **testprogram** with and without hyperthreading on 100 nodes measuring the execution times of:

```
mpirun –f hostfile -np 1600 -ppn 16 ./testprogram
```
vs

```
mpirun –f hostfile -np 3200 -ppn 32 ./testprogram
```

The file with name **hostfile** can should contain the list of all nodes in the job. This can be achieved by doing

```
cat $PBS_NODEFILE| sort |uniq > hostfile
```

Please be aware that with 32 tasks-per-node each process gets only half of the memory by default. In the Avitohol cluster, there are 150 compute nodes available with 64 GB of real memory (some is taken by the OS). So, on these 150 nodes, you can expect to have 2 GB per core for MPI jobs with hyperthreading or 4 GB per core for MPI jobs without hyperthreading.

The default Parallel Environment is Intel MPI. You can use executables that were built with other MPI libraries. For example, openmpi is also available system-wide.

For each execution node there are two associated Xeon Phi coprocessors. For example, on node **sl039** the coprocessors are available via ssh or for mpiexec with the names **sl039-mic0** and **sl039-mic1**.

Each coprocessor is running its own operating system, which means that it appears as separate machine.

For each execution node there are alternative names that correspond to alternative network interfaces, associated with the InfiniBand cards. For example, **ibsl039** corresponds to the IP-over-InfiniBand network interface ib0 on host **sl039**.

The same type of IP-over-InfiniBand interfaces are available on the coprocessors, for example **ibsl039-mic0** and **ibsl039-mic1**.

Although these interfaces offer higher bandwidth than the standard Ethernet interfaces, they are not necessary for normal use, because MPI jobs are expected by default to use native InfiniBand and not IP-over-InfiniBand.

The Lustre filesystem already uses native InfiniBand transport and since the files are shared between nodes and it is even available on the coprocessors, necessity for data movement between nodes via the IP-over-InfiniBand interface may be indication for sub-optimal use of the system.

InfiniBand is also the default transport for MPI when using the coprocessors via MPI.

Once a given user has obtained exclusive access to an execution node he or she can also use the corresponding Xeon Phi coprocessors. Inside a job script one may read the contents of the file pointed by the environment variable PBS_NODEFILE and obtain the names of the corresponding Xeon Phi nodes. The access to the Xeon Phi coprocessors is not managed separately.

However, users are not allowed to login with ssh to nodes where they do not have running jobs. To determine which nodes have been allocated to a given job, the user can run

```
qstat –na <jobnumber>
```

Users are not expected to use Xeon Phi coprocessors without first requesting their host node from the batch system.

Further reading

- For more information about using **torque** batch system and **moab** scheduler please see the documentation from the website of the software provider: **www.adaptivecomputing.com**.

4.2. Compilers

The Intel Fortan (`ifort`) and C/C++ (`icc, icpc`) compilers are the default compilers on the HPC cluster Avitohol 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. Version 4.4.7 is installed on Avitohol system. More recent versions can be accessed via environment modules. To compile and link MPI codes using GNU compilers, use the commands `mpicc, mpic++/mpicxx, mpif77 or mpif90`, respectively.

The compilation for the Xeon Phi is usually done at the host, i.e., using cross-compilation.

To load the development environment for the Xeon Phi one can issue

```
source /opt/mpss/3.6/environment-setup-k1om-mpss-linux
```

Then the version of the compiler and other development tools can be accessed with something like:

```
k1om-mpss-linux-gcc

k1om-mpss-linux-g++

k1om-mpss-linux-nm
```

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

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].
- For more information about using MPSS consult the documentation from Intel [https://software.intel.com/en-us/articles/intel-xeon-phi-coprocessor-developers-quick-start-guide]

4.3. Parallel Programming

On the Avitohol 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 Avitohol a pure OpenMP code is restricted to run on a single node (whether the host or the coprocessor system).

It is possible to mix MPI and OpenMP parallelism within the same code in order to achieve large scale parallelism.
Because of the presence of Xeon Phi accelerators, they are also available on the coprocessors, and all the modes of using coprocessors - native, symmetric and offload, are available.
Each Xeon Phi coprocessor can run OpenMP programs in native mode. The number of physical cores on a coprocessor is 61. It is advisable to reserve one core for the operating system use and to use at most 60 cores. However, the number of logical cores on the coprocessor is 4 times the number of physical cores, thus 244. Experience has shown that using number of threads higher than the number of physical cores can be beneficial. Users can try to find which setting of the number of threads for OpenMP via the variable OMP_NUM_THREADS (in native mode) or MIC_OMP_NUM_THREADS (in offload mode) is acceptable. In many cases 120 seems to be a good compromise.
For the programs that run on the host the number of threads may be set equal to the number of CPU cores (16) or number of logical cores (32) if only OpenMP is used. If a hybrid MPI/OpenMP program is run, one must take into account how many MPI processes are running on one node. For example, if two MPI processes are run on each node, then OMP_NUM_THREADS should be set to 8 to use all physical cores or 16 to use all logical cores.

4.4. Parallel MPI applications

By default, the Intel compilers for Fortran/C/C++ are available on Avitohol.

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.
The parallel program can be started with `mpiifort`. We remind that environmental variable can be passed via options. For example, to set the variable MKL_MIC_ENABLE to 1, thus enabling automatic offload for all MPI processes, one can add

-genv MKL_MIC_ENABLE 1

to the mpiexec command line.

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

Some environment variables may give information about software that does not come directly from the operating system distribution. Examples are some build tools like ant/maven, which are provided.

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

To compile and link OpenMP applications pass the flag –qopenmp (which replaces the former option –openmp, which is now deprecated) to the Intel compiler, or –fopenmp to the gcc 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 Avitohol the thread private stack size is set via the environment variable KMP_STACKSIZE. The default value is 4 megabytes (4MB). For example, to request a stack size of 128 megabytes on Avitohol, set KMP_STACKSIZE=128m.
Beware that on the coprocessor the practical limit for number of threads is 244 (61 hardware threads times 4 due to hyperthreading). Thus using high numbers for KMP_STACKSIZE may crush the application.

Note

Applications occasionally may use swap memory. In general such application behavior is undesirable and strongly discouraged. It is presumed to be a result of application mis-configuration or error in the input. Users are expected to cancel jobs that they observe to be swapping and try and correct the errors. If they believe that this is normal behavior, they should discuss this with system administrators.

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 Avitohol

Intel Math Kernel Library overview

The Intel Math Kernel Library (MKL) is provided on Avitohol. 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.
By setting the environment variable MKL_MIC_ENABLE to 1, e.g., by

```
MKL_MIC_ENABLE=1
```

in bash shell, one can enable the automatic offload of some library routines to the Xeon Phi coprocessors that are available. For using this option with MPI, see above.

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/].

About the automatic offload one can consult:

https://software.intel.com/sites/default/files/11MIC42_How_to_Use_MKL_Automatic_Offload_0.pdf

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 on Avitohol (Linux, Intel64) using standard 32 bit integers, something like these must be added to the Makefile when invoking 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 –qopenmp
```

(in one line). If compiling directly in bash, one must use ${MKLROOT} instead of $(MKLROOT).

4.7. Numerical Libraries

FFTW

The so-called "Fastest Fourier transforms in the West/World" software is installed. Currently the version of FFTW 3 that comes with the OS is 3.2.3 and is available in the standard locations for development software.

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 part of the OS distribution.

Python modules

The **Python** version from the OS distribution is of the 2.6 series. The latest current 2.7 series version is available as a module. The **tensorflow** module is available as part of the 2.7 installation.

4.8. I/O Libraries

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 (version 4).

4.9. Miscellaneous Libraries

1. Boost: The Boost C++ libraries provide many classes and routines for various applications. The version of boost provided by the OS is rather old. Newer version is available under `/opt/soft/` and are available through the environmental modules system.
2. TBB: Intel Threading Building Blocks. TBB enables the C++ programmer to integrate (shared memory) parallel capability into the code. Available as part of the Intel Compiler Suite.
3. IPP: Intel Integrated Performance Primitives (IPP). The IPP API contains highly optimized primitive operations used for digital filtering, audio and image processing.
4. PAPI: The Performance Application Programming Interface is a library for reading performance event counters in a portable way.

4.10. Other software

Other popular packages are available system-wide. For example – R, cmake, gnuplot.

Environment modules

- Please use **module avail** on Avitohol to obtain a comprehensive list of environment modules.
- In most cases only the latest/current production version (as of March 2016) 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`.

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.
- gdb, the GNU debugger.
- Intel Inspector enables the debugging of threaded applications.

- If debugging features will not be used, the executable can be stripped of debugging information by running **strip.**

5.2. Profiling and Performance Analysis

- 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.
- Optimised executables can be obtained by first compiling for profile generation (option -prof-gen for Intel, --fprofile-generate for gcc) and then running the executable and using the generated profile with options like
    - -prof_use -prof_dir ./profdir for Intel
    - '-fprofile-use - for gcc, sometimes it is necessary to add -fprofile-correction or -flto-partition=none

Further reading

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

• http://www.hpc.acad.bg/systems/avitohol/index.php/system-1

6. Conclusion

The Avitohol system is relatively new, but is already equipped with a rich set of development tools, libraries and software. Users that require additional tools or software to be installed or upgrades to existing software should contact the support team at **avitohol-support@parallel.bas.bg.**