HP-SEE

Advanced Application Porting and Optimization

www.hp-see.eu

Emanouil Atanassov
WP5 leader
Institute of Information and Communication Technologies
Bulgarian Academy of Science
emanouil@parallel.bas.bg

HP-SEE

High-Performance Computing Infrastructure for South East Europe's Research Communities

The HP-SEE initiative is co-funded by the European Commission under the FP7 Research Infrastructures contract no. 261499

OUTLINE

HP-SEE
High-Performance Computing Infrastructure for South East Europe's Research Communities

- Goals
- Application porting
- Profiling and optimization
- Conclusions

Goals

HP-SEE
High-Performance Computing Infrastructure for South East Europe's Research Communities

- The main goal is to be able to perform the desired computer simulations/computations
- The goal of the application porting phase is to produce and verify executables for the target architecture, with acceptable performance
- The further optimization of the application aims to improve scalability, shorten execution time, reduce load on the infrastructure, etc.

Application porting

HP-SEE
High-Performance Computing Infrastructure for South East Europe's Research Communities

- Two main use cases are considered:
  - Application developed mainly by the developers' team
  - Application that uses established open source or commercial tools, with or without access to actual source code

Issues during application porting

HP-SEE
High-Performance Computing Infrastructure for South East Europe's Research Communities

- Missing libraries
  - Operation teams would usually install libraries that are distributed as rpms in the popular repositories
  - Non-standard libraries must be compiled and installed by the developers themselves (set LD_LIBRARY_PATH appropriately).
- Problems with interpreted languages (java, python, perl, etc.)
  - Usually the installed version would be different from the desired, or some modules would be missing
  - In most cases – install your own version under your user account
  - Example: s3curl on bg-fen (the front end node of Blue Gene P) required some perl modules, that on their own required newer version of perl. After that commands like:
    - perl -MCPAN -e "install Digest::HMAC_SHA1"

Issues during application porting

HP-SEE
High-Performance Computing Infrastructure for South East Europe's Research Communities

- Top problems with compilers
  - Default versions of compilers available on SL 5 are rather outdated – you can use gcc44 or you can deploy your own version (takes a long time to compile a compiler, so do this only if really necessary)
  - Compilation for CUDA may be tricky, sometimes codes that compiled fine with previous version may start showing problems with newer version. You have no control over the driver module that is installed, but you can use your own toolkit/SDK version.
  - On Blue Gene P the compiler to use is a cross compiler thus running the configure scripts requires extra care.

Compiler options used on Blue Gene P

HP-SEE
High-Performance Computing Infrastructure for South East Europe's Research Communities

- CC="xlc_r"
- CPP=xlc_r -E
- CXX=xlC_r
- CXXFLAGS=-qarch=450d -qtune=450 -O3 #-qstrict
- CXXCPP=xlC_r -E
- F77=xlf_r
- FC=xlf_r
- F90=xlf90_r
- FFLAGS="-O3 -qstrict"
- MPICXX=/bgsys/drivers/ppcfloor/comm/bin/mpixlcxx_r
- MPICC=/bgsys/drivers/ppcfloor/comm/bin/mpixlc_r
- MPIF77=/bgsys/drivers/ppcfloor/comm/bin/mpixlf90_r

Issues with static/dynamic libraries

HP-SEE
High-Performance Computing Infrastructure for South East Europe's Research Communities

- Compiling with –static or similar options has performance and to some extent portability advantages, however:
  - Some widely used libraries have no static version in a default installation
  - The resulting executable may be dependant on using the same version of the standard C library, which may not be the case if system is upgraded
  - Always make sure LD_LIBRARY_PATH has the right content
  - Do not mix compilers, if you can avoid it. Always compile NETCDF with the same compiler like the rest of the program.

Compiler flags

HP-SEE
High-Performance Computing Infrastructure for South East Europe's Research Communities

- First goal is to achieve correctness of the application execution. Conservative compilation flags include:
  - -O0 or –O2, -g for gcc
  - -fp-model strict for icc or similar
  - -qstrict for IBM Blue Gene compilers

Test cases are used to detect wrong compilation results. Many open-source applications have visible mistakes in the code related to the move to 64-bit architectures (a pointer is not the same size as an int, etc.). Correct these as last resort.

Compiler flags

HP-SEE
High-Performance Computing Infrastructure for South East Europe's Research Communities

- More aggressive optimization flags can be tried once we see the application is working correctly.
- For Intel/AMD based clusters, it may be useful to have a look on spec.org and see what kind of flags were used for the benchmark cases submitted there, since these are supposedly optimal. They differ between applications.
- Using profiling information is beneficial, i.e. two phases of compilation are recommended. For Intel compiler, the relevant options are –prof_gen and -prof_use. May take a long time.

Improving scalability

HP-SEE
High-Performance Computing Infrastructure for South East Europe's Research Communities

- Application profiling is useful for improving scalability of your own code. Using non-blocking MPI communications, trying to optimize cache use, overlapping communications and computations are usual techniques.
- Avoid using swap!
- For other people's code our options are mainly:
  - Choice of parallelization strategy
  - Choice of compiler, mpi library, lapack/blas and other libraries
  - Choice of compiler flags
  - Choice of number of nodes, number of CPUs/Cores per node to actually use vs the maximum, options to tune MPI usage

Improving scalability

HP-SEE
High-Performance Computing Infrastructure for South East Europe's Research Communities

- Choice of parallelization strategy:
  - Some codes offer choice of parallelization strategy –
    - pure MPI
    - hybrid MPI+OpenMP (or MPI+shared memory)
    - Custom sockets (example – GAMESS) or ibverbs (example – NAMD)
- Choice of compiler, mpi library, lapack/blas and other libraries
- There are many examples showing that difference between compilers is not so big, but still relevant.
- Contrary to the suggestions in documentation, we have found that pure MPI outperforms the other parallelization strategies (your mileage may vary).

Improving scalability

HP-SEE
High-Performance Computing Infrastructure for South East Europe's Research Communities

- The behaviour of the MPI executable can be changed at runtime via command line options or environment variables. Example of such tuning include:
  - For NAMD, openmpi (suggested combination):
    - -gmca mpi_paffinity_alone 1 -gmca btl_openib_eager_limit 32767
  - For GAMESS, Intel MPI:
    - setenv I_MPI_WAIT_MODE disable (important, contrary to suggestion in doc)
    - setenv I_MPI_DAT_LIBRARY libdat2.so (as suggested in doc)
  - To PIN or not to PIN (processes to cores)

Detecting bottlenecks

HP-SEE
High-Performance Computing Infrastructure for South East Europe's Research Communities

- When the application is running, we can determine:
  - Does it use swap – leads to extremely bad performance in most cases?
  - Are the processes using close to 100% CPU time or idling?
  - Is disk I/O time important?
- Using 100% CPU does not guarantee good performance!
- How to see that – login to the node and run top
- How to see what the process is doing – strace –p xxxx
- You can attach gdb to a particular process
- Use MPI tracing and/or profiling tools
  - MPE for tracing (jumpshot for visualisation)
  - mpiP for profiling
- Find optimal number of nodes / number of CPU cores per node
  - Using the hyper-threading may be useful or useless – use appropriate options to launch less processes than PBS gives you.

Demo

HP-SEE
High-Performance Computing Infrastructure for South East Europe's Research Communities

- Demonstration of application porting and optimisation