HP-SEE

Application case studies

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

- Stages of application porting
- Application case study: own application
- Application case study: established open source applications
- Conclusions and directions for future work

Stages of application porting

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

- Obtain sources, resolve dependencies
- Choose initial version of MPI to use
- ./configure
- Make
- Run tests, eliminate errors
- Read the instructions again
- Re-configure, change compiler options etc.
- More extensive testing, study scalability with real data
- Wash, rinse, repeat!

Application case study: own codes

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

- Example application: SET
- Codes are under our control
- Main dependencies: parallel random number generators (SPRNG), parallel low-discrepancy sequences generators (own codes), optional inclusion of Genetic Algorithms codes (galib)
- Issues when porting to Blue Gene P:
  - OS is CNK, not linux (no fork)
  - We had a small dependency on SSL, removed it
  - The codes we used for parallel random number generation gave incorrect results, most probably because of Little Endian vs Big Endian issues. Replaced with previously used SPRNG codes.
  - Tricky compilation of galib (had to modify some codes, Makefile)
  - Excellent scalability, once everything is working.

Application case study: established applications

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

- Example application: GAMESS
- Input data: Jose Kaneti from IOCCP-BAS
- Issues with his own scripts: high memory usage, causes swapping, system instability, long execution times (runs days on more than 10 nodes).
- Initial approach:
  - recompile application for MPI instead of using sockets
  - Vary number of cores per node to determine optimal number – found that using just one core for computations and one for communications is faster than using more cores (removes swap).
  - How can we use less than one core? Can we use memory from one node on another one? Yes!
  - Some interesting things (bugs) we found: DDI_Id vs DDI_Id()

Application case study: established applications

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

- After reading the instructions again:
  - Memory settings can be made such that the application fits in physical RAM, even with more cores. In this way we do not have the paradoxical result that one core is better than two or more.
  - Good scalability with up to 4 cores for computations plus 4 for communications, with more than 10 nodes.
  - Total time decreased to several hours (4-5 hours).
  - Using more cores decreases performance
  - The options -prof_gen and –prof_use were used to recompile for the target architecture
  - Varying the MPI options we concluded that I_MPI_WAIT_MODE should be disabled
- Additional efforts that did not produce better results -
  - Returning to the sockets implementation

Application case study: established applications

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

- Example: NAMD
- Various parallelization strategies available: SMP, CUDA, MPI
- Initial attempts:
  - The CUDA version was successfully compiled and run
  - Required some tweaking in the options of the test case in order to run it
  - Runs using more than one CUDA device in parallel (we have two on each CUDA node).
  - The iverbs version was succesfully compiled and run on the Infiniband cluster. This should be 10% faster than MPI version
  - Hit a performance problem when trying with more than 4 cores per node.

Application case study: established applications

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

- After reading the instructions:
  - Attempt with openmpi and gcc
  - Tricky compilation – had to change manually some options in the automatically generated configuration files
  - Overcomes the scalability issue: up to 8 cores per node can be used, using all 16 threads per node yields small improvement.
- Additional efforts:
  - Suggested options for openmpi yield another small improvement.

Application case study: established applications

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

- NEURON is an application from the domain of neuro-science.
- Has a parallel version
- Extremely tricky compilation in case of GCC, easier with ICC.
- Main reason for this – the application uses flex and bison. One of these was causing a problem by having a second definition of the same function. Manual changes in scripts were needed.
- Succesfully tested for correctness running in parallel, we still do not have realistic test case to test scalability.

Conclusions and future work

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

- When dealing with our own codes we have flexibility to resolve porting and/or scalability issues
- When dealing with other people's codes we are limited in our options. These codes tend to be complex and have rather strange portability or performance issues. Following suggestions from the documentation is a hit or miss.
- It is always important to concentrate on test cases that are close to our real data.
- In the hands-on session we will try and follow part of the process that we described here.

DEMO

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

- Demonstration of application porting