D7.5 Deliverable contribution guidelines for Subtask 7.5.D

Chapter D: "Multi-core/many-core systems"

10-15 pages total; Contributors: all

Contribution from SNIC-LIU :

On modern clusters with many-core / multi-core nodes there are large differences in bandwidth and latency between cores in the same node and cores across different nodes. This leads to a non-uniform network topology for MPI processes w.r.t. Bandwidth and latency. Moreover in many cluster configurations a node has single interconnect cable going out for connecting with other nodes. It creates a blocking configuration for MPI communication. As an example for a 32 core node when all the MPI ranks try to communicate outside node through a single interconnect cable it creates a 1:32 blocking situation. This might impact the data transfer rate and consequently the scalability of codes. On such systems topology unaware MPI programs can create network congestion. Thus it is important to write topology aware algorithms which exploit intra-node bandwidth / latency more and less reliant on inter-node bandwidth / latency. Although it is generally difficult to introduce network awareness in application programs, for MPI collective operations it is possible to rewrite the collectives in a topology aware way. In the project - “Improving MPI communication latency on Euroben kernels” we have demonstrated the impact of topology aware MPI_Alltoall routine on the performance of a kernel called mod2f.

D.1 Introduction and summary

3 pages; Contributors: Antun Balaz, Vladimir Slavnic

D.2.Improving MPI communication latency on Euroben kernels

D.2.Improving MPI communication latency on Euroben kernels.1 : Project summary

Project name: Improving MPI communication latency on Euroben kernels

Project authors and their contacts (institutions, e-mails): Chandan Basu, Linköping University, Sweden, cbasu@nsc.liu.se

WP7 contributors and their contacts (institutions, e-mails): Chandan Basu, Linköping University, Sweden, cbasu@nsc.liu.se

Effort (pms): 0.8

White Paper, Technical report or Scientific publications

- Improving MPI communication latency on Euroben kernels - Chandan Basu, Linköping University, Sweden
- https://bscw.zam.kfa-juelich.de/bscw/bscw.cgi/d709869/mod2f_tuned_whitepaper.pdf

Hardware platform(s): Curie system

Programming language(s): c, MPI

Brief project description:

The MPI communication latency is an important factor in scalability and performance of parallel programs. Many parallel programs are heavily dependent on MPI collective calls. The performance of MPI collective calls can be improved if these collectives are topology aware. Normally MPI installations are not topology aware. However it is possible to add topology information through some wrapper routines around MPI calls. To demonstrate this we have chosen mod2f kernel from euroben benchmark suite. Euroben benchmark suite has many synthetic benchmark programs. The C-MPI version of mod2f program is suitable for testing our approach as it is communication intensive. We have tested the scalability and performance of this modified version of mod2f w.r.t the performance original mod2f for different data sizes with encouraging results.

Applications area(s): Parallel programs, collective operations

D.2.Improving MPI communication latency on Euroben kernels.2 : Description of work

The mod2f routine is a 1-D Fast Fourier Transform (FFT). The C, MPI version of mod2f program does FFT of distributed arrays. The main communication overhead

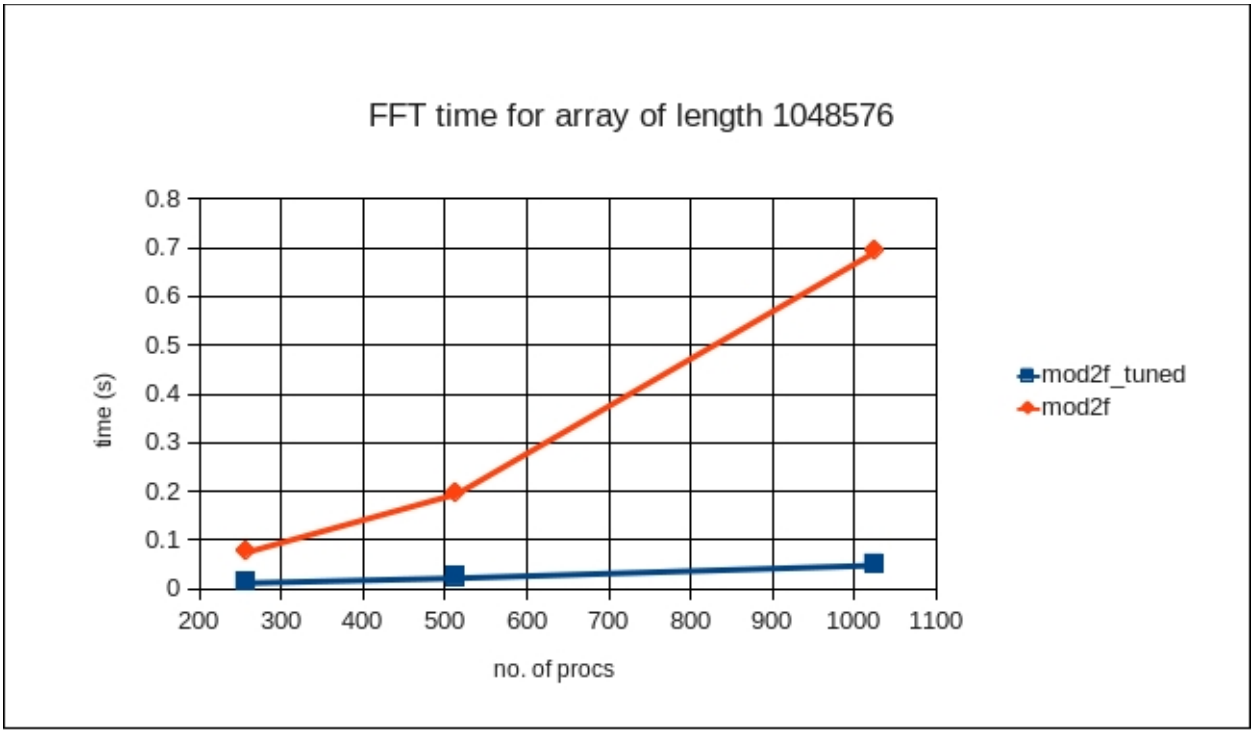
comes from a routine called gtrans. This routine does a global transposition of arrays. To achieve global transpose gtrans calls MPI_Alltoallv routine. As is well known MPI_Alltoall / MPI_Alltoallv is the very communication intensive MPI call and consequently does not scale well. The code while running prints out communication overhead which shows that most of the time is spent in communication routines.

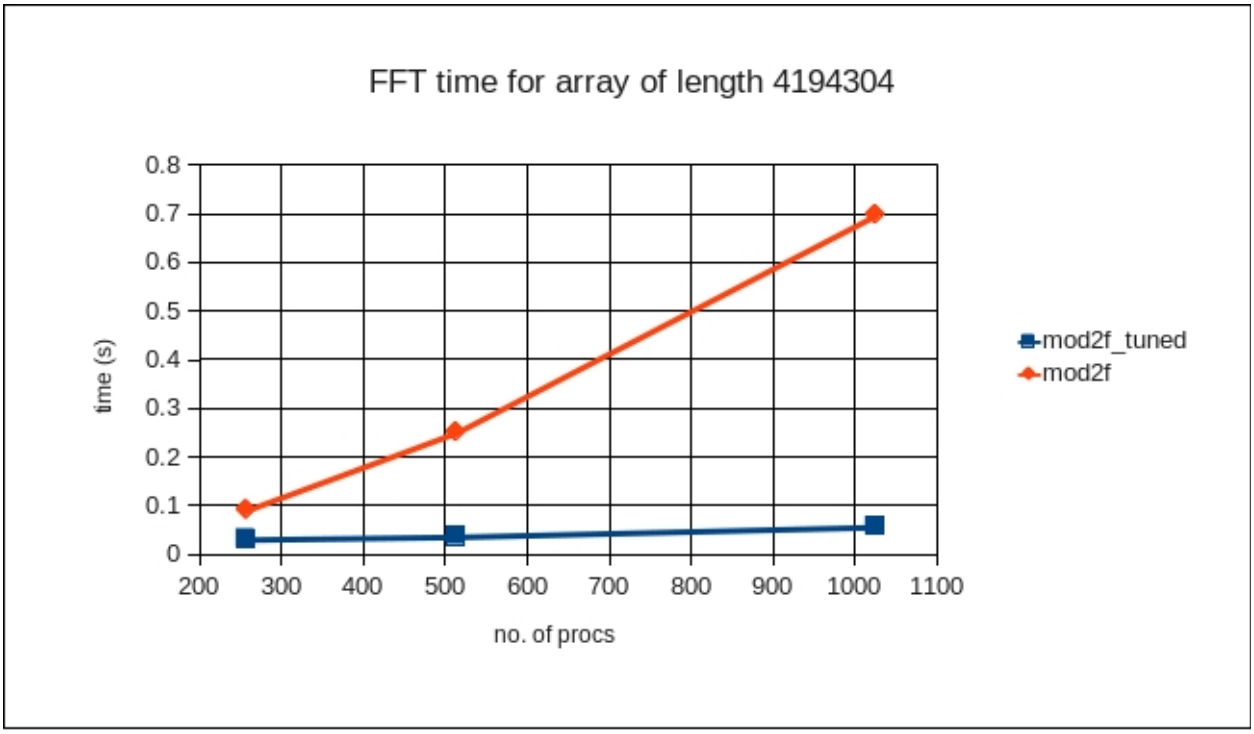
We have developed a topology aware version of MPI_Alltoallv called MPI_Alltoallv_tuned. In mod2f we replace the MPI_Alltoallv calls with MPI_Alltoallv_tuned calls. However the interface of MPI_Alltoallv_tuned is not exactly same as standard MPI_Alltoallv. Some data structures passed to MPI_Alltoallv_tuned are different from MPI_Alltoallv. Those data structures are additionally defined and initialized at appropriate places.

We needed to do some structural changes in the code for improving efficiency. In the original version of mod2f send / receive / displacement counts for MPI_Alltoallv are computed within gtrans. However these counts do not change for a particular FFT operation. It is enough to compute the counts only once outside the iterative loop. We have modified the code accordingly. This modification is suitable for our approach. Because in our approach send / receive / displacement counts are passed as structures and their computation is more complex than send / receive / displacement counts for MPI_Alltoallv.

In the modified version of the code the original version or the tuned versions are activated by preprocessor flags. As we are interested in seeing the scaling effects on large number of cores / nodes we choose relatively large sizes of the problems. We have done our tests on Curie system. We show our results in the following graph. The results show that mod2f_tuned times are much better than the standard mod2f version.

The current MPI_Alltoallv_tuned is working for MPI_COMM_WORLD. Also the total number of MPI processors should be divisible by number of cores / node. There are few ad-hock things in the current version of MPI_Alltoallv_tuned, e.g., statically allocated work buffers etc. These are however limitations of current implementation and can be removed.


FFT time for array of length 1048576
time (s)
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
200
300
400
500
600
700
800
900
1000
1100
no. of procs
mod2f_tuned
mod2f


FFT time for array of length 4194304
time (s)
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
200
300
400
500
600
700
800
900
1000
1100
no. of procs
mod2f_tuned
mod2f

D.2.Petascaling enabling and support for EC-Earth3

One section D.2.X per project X, 1-1.5 pages per project, Contributor: project owner

Contents:

D.2.Petascaling enabling and support for EC-Earth3.1 : Project Summary

Project name: Petascaling enabling and support for EC-Earth3

Project authors and their contacts : John Donners, SARA Amsterdam, The Netherlands, john.donners@sara.nl

WP7 contributors and their contacts : Chandan Basu, John Donners (WP7.2), et.al

Effort (pms): 3.2

White Paper, Technical report or Scientific publications

- Petascaling enabling and support for EC-Earth3 (under preparation)

Hardware platform(s): Curie system

Programming language(s): c, FORTRAN, MPI

Profiling tools: TAU profiling tool, strace

Libraries: Netcdf, Intel MKL

Brief project description :

The EC-EARTH is an earth system modeling application. The three main components of EC-EARTH are IFS (for atmosphere), NEMO (for ocean) and OASIS (for coupling). We have ported, benchmarked and analysed a high resolution version of EC-EARTH configuration. The ocean component used for this configuration is ORCA025 which is a 1/4° global configuration. The atmosphere component uses T799 resolution. The system used for our scaling studies is Curie. As EC-EARTH is a large and complex program it's porting on a new machine is a considerable challenge. The goal is to test the scalability of this model and to figure out the bottlenecks.

Applications area(s): Meteo-climatology

D.2.Petascaling enabling and support for EC-Earth3.2 : Description of Work

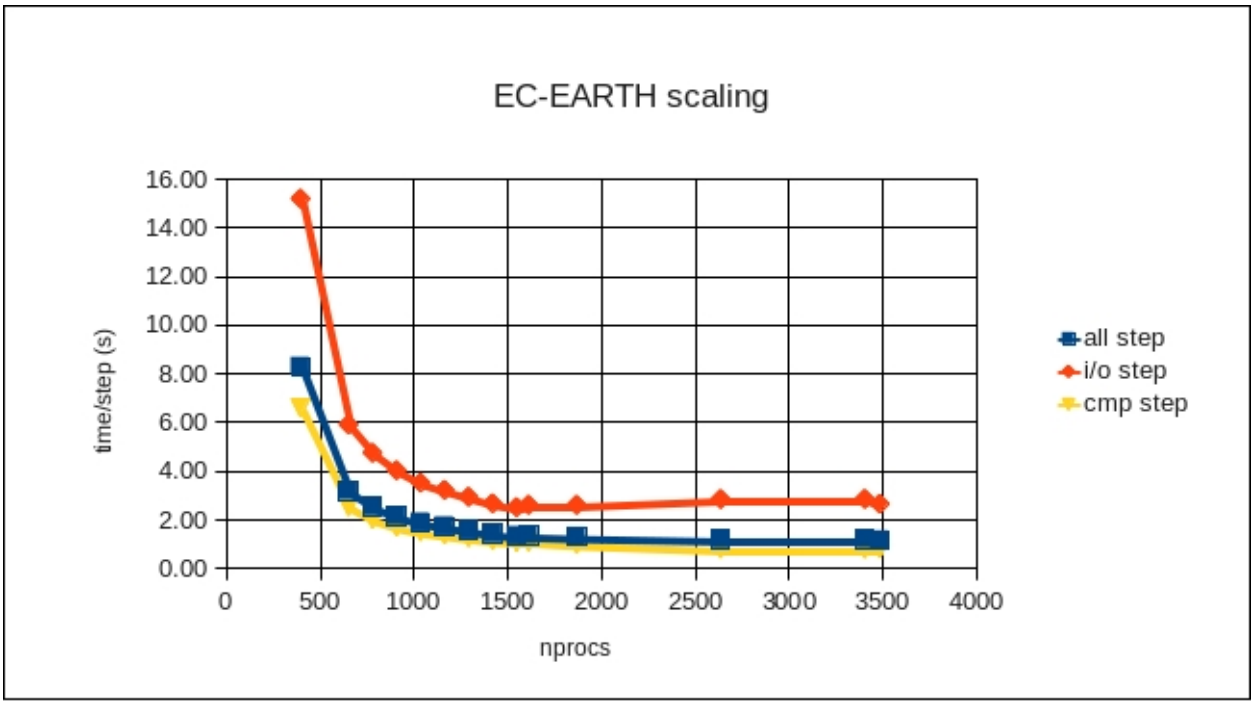
EC-EARTH is made up of three different components. These components have quite different build environments. The EC-EARTH package provides a semi-automatic utility called eexcon. The important thing in the EC-EARTH compilation is to set up

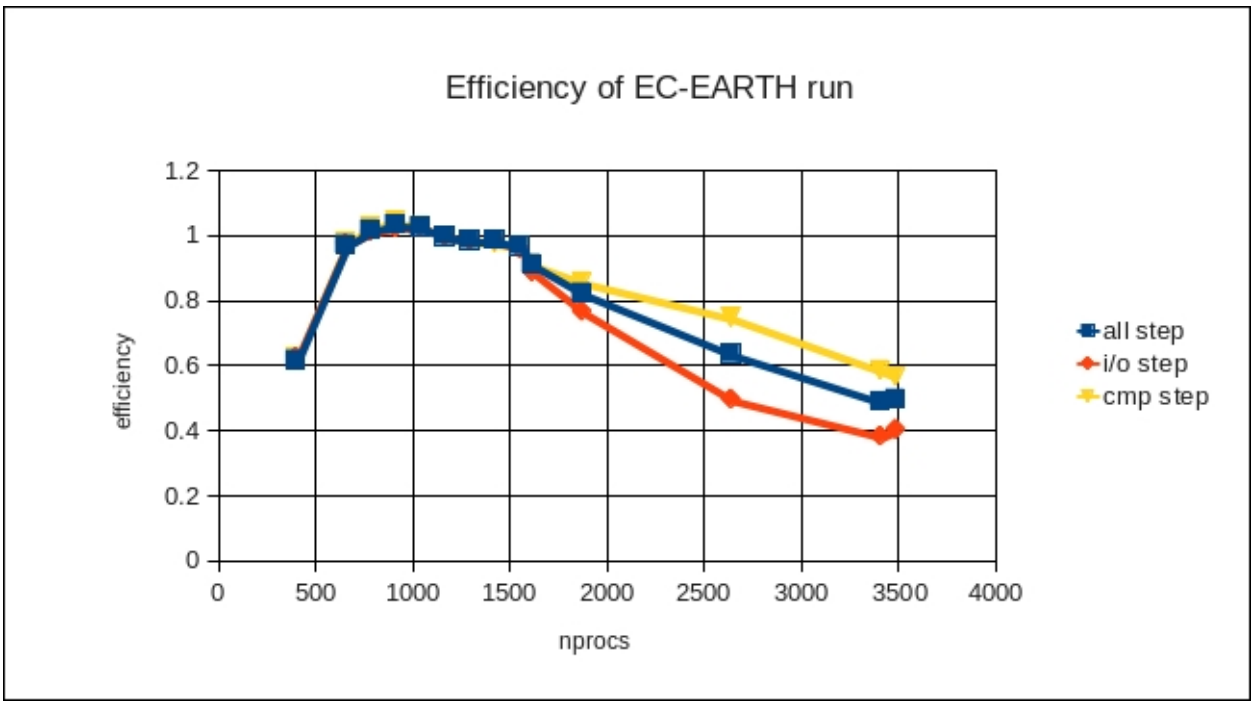
an appropriate xml file with description of system specific settings for package paths, compilers, flags, libaries etc. We have used intel comipler version 12.1.7.256 bullxmpi version 1.1.8.1 and netcdf version 4.1.1 in our compilation.

EC-EARTH compilation produces multiple binaries, one each for NEMO, OASIS and IFS. The bullxmpi on curie is capable of launching MPMD runs. For efficient runs it is needed that IFS, NEMO and OASIS processes are not mixed up in the same node. We have created specialized scripts for efficiently submitting jobs on targetted nodes.

We have run high resolution EC-EARTH upto 3500 MPI processes. At this scale the efficiency is ~50% of the most efficient run. The most efficient run is obtained at ~1034 MPI processes. When EC-EARTH is run with less than 1034 MPI processes the efficiency is lower. This is because for less than 1034 processes NEMO processes finish earlier and wait for IFS processes.

We have profiled EC-EARTH with TAU. TAU instruments the source code with light weight timers and then compiles and links. Both the MPI functions and user functions are profiled with TAU. The TAU profiling has negligible overhead. We have also observed the i/o access pattern of EC-EARTH through strace. This shows that few small files are frequently opened and read.





D.3 Conclusions

Contributors: Antun Balaz, Vladimir Slavnic