Centro Euro-Mediterraneo sui Cambiamenti Climatici

Research Papers
Issue RP0208
December 2013

ANS - Numerical Applications and Scenarios Division

Performance characteristics of the PELAGOS coupled model

By **James Brunson**
Centro Euro-Mediterraneo sui Cambiamenti Climatici (CMCC)

Tomas Lovato
Centro Euro-Mediterraneo sui Cambiamenti Climatici (CMCC),
tomas.lovato@cmcc.it

Marcello Vichi
Centro Euro-Mediterraneo sui Cambiamenti Climatici (CMCC), Istituto Nazionale di Geofisica e Vulcanologia (INGV)
marcello.vichi@bo.ingv.it

William McKiver
Centro Euro-Mediterraneo sui Cambiamenti Climatici (CMCC),
william.mckiver@cmcc.it

Esteban Damian Gutierrez Mlot
Centro Euro-Mediterraneo sui Cambiamenti Climatici (CMCC),
esteban.gutierrez@cmcc.it

SUMMARY This report describes and assesses the computational performance of the PELAGOS coupled model, a global ocean coupling of NEMO and BFM. The first part is dedicated to a comparison of CMCC architectures on which the model is running, and the second part offers insight regarding focal points pertaining to near and long-term optimization possibilities.

Keywords: HPC, PELAGOS, biogeochemical modelling, performance, code optimization

This work was funded by the EU FP7 project GreenSeas and by the Italian Ministry of Education, University and Research and the Italian Ministry of Environment, Land and Sea under the GEMINA project.The authors wish to thank Italo Epicoco, Silvia Mocavero, Giovanni Aloisio and all the members of the SCO division for their support.

02 Introduction

The scope of this project was to analyze performance characteristics of the PELAGOS coupled model, to compare and contrast CMCC architectures on which the model is running, and to offer insight regarding focal points pertaining to near and long-term optimization possibilities.

PELAGOS is a coupled model consisting of two components; the “blue ocean”, represented by NEMO, and the “green ocean” as modelled by BFM (Biogeochemical Flux Model).

PELAGOS is currently being run on the following CMCC HPC clusters:

IBM P575 cluster

The P575 cluster at CMCC is a power6-based system, with the following characteristics:

- 2-core socket @ 4.7 GHz
  - 64KB 4-way L1 cache, instruction + data per core
  - 4MB “semi-private” L2 cache
  - 32MB L3 (shared)
  - 18.8 GFlops peak performance per core
  - Support for 2 Symmetric MultiThreading instances per core, 64 threads per node
- 32 core (4 quad x 4) per node
- 2 memory controllers/socket
- 128 GB memory/node (*4GB/core*), DDR2 (533 MHz?)
- 4x Infiniband DDR interconnect on PCI-X (20 Gb/s = 2.5 GB/s unidirectional b/w)

IBM iDataPlex dx 360 m4 cluster

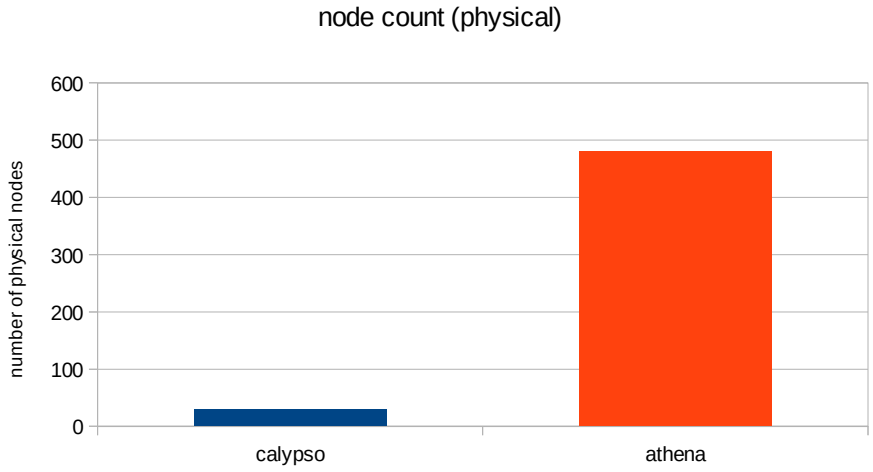
The idataplex cluster at CMCC is an Intel Xeon (Sandy Bridge E5-2670) system with the following characteristics:

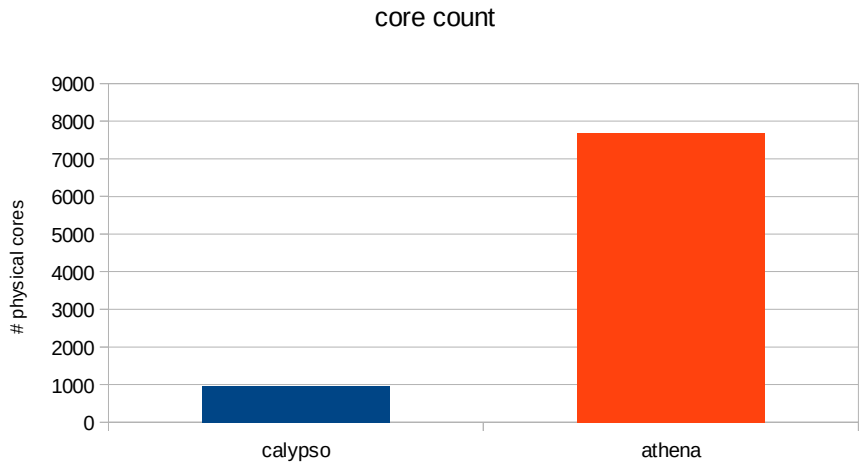
- 8-core socket @ 2.6 GHz (with 3.0/3.3 turbo modes)
  - 32KB 8-way L1 cache, instruction + data per core
  - 256kb L2 cache per core
  - 20 MB L3 cache (shared)
  - 20.8 GFlops peak performance per core
  - Support for 2 HyperThreading instances per core, 32 hyperthreads per node
- 2 sockets per node, 16 core per node
- 4 memory controllers/socket
- 51.2 GB/s memory bandwidth per socket
- 64 GB memory/node (*4GB/core*), DDR3 1600MHz

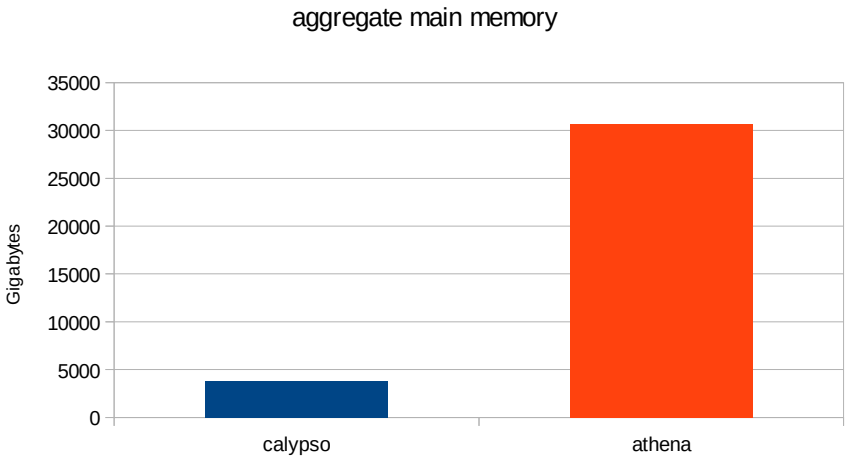
- 4x Infiniband FDR10 interconnect on PCIe3 (40 Gb/s = 5 GB/s unidirectional b/w)

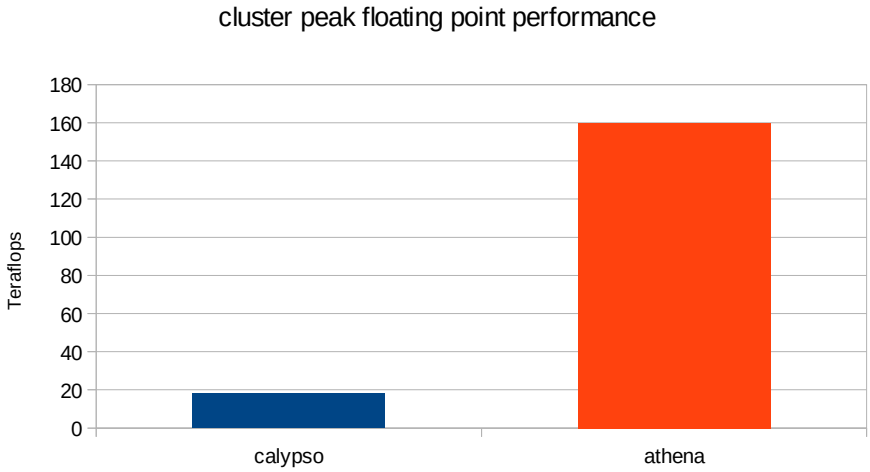
Comparing the two clusters

The charts below illustrate the increase in core count/memory/peak floating point performance from the calypso cluster, installed in 2009, to athena, installed in early 2012.





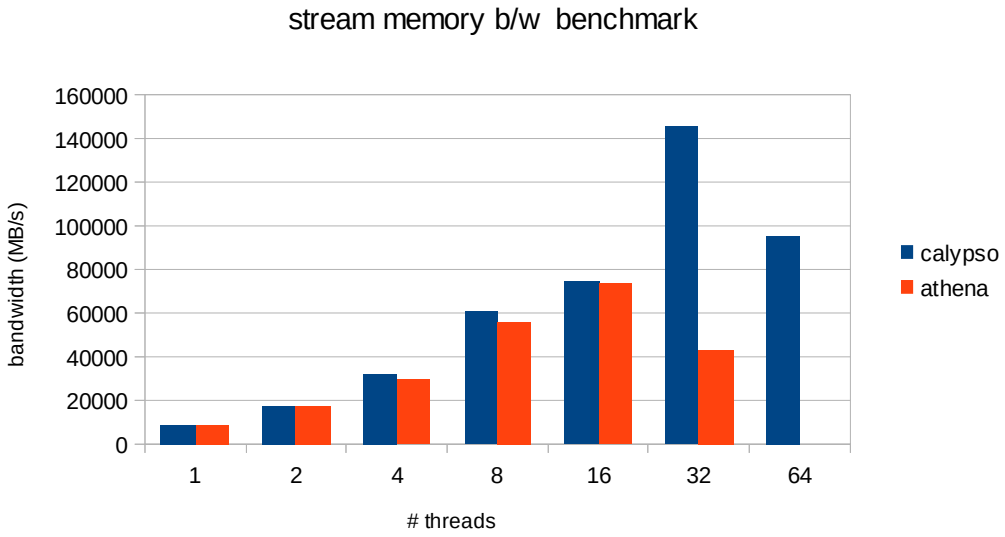




Memory Bandwidth

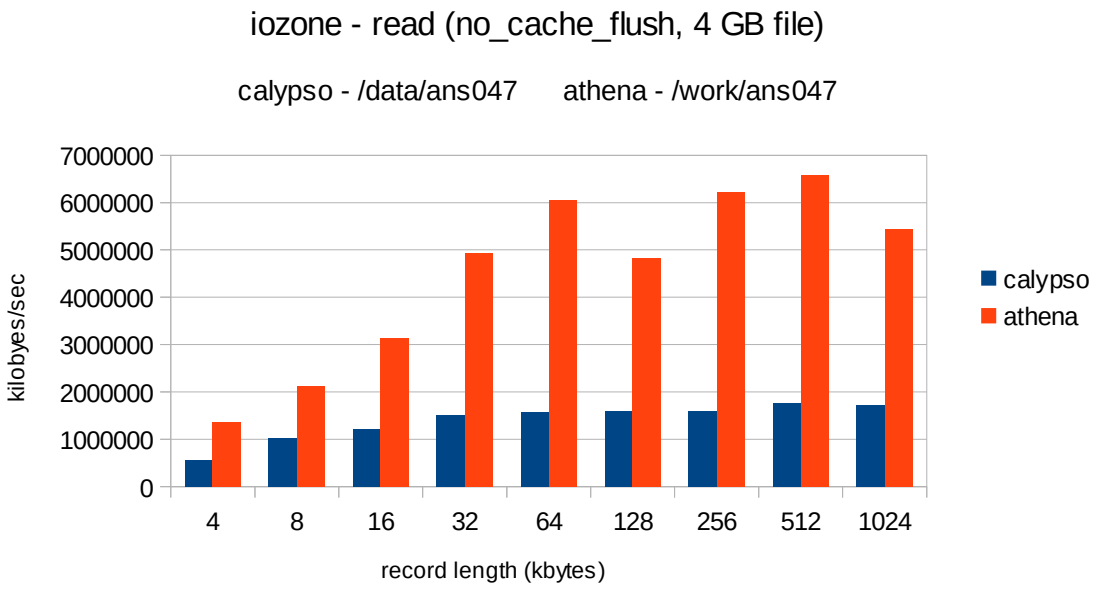
Both power6 and x86 nodes of the respective CMCC clusters offer nearly identical memory bandwidth. The table below illustrates Triad numbers achieved on each cluster at CMCC; at 32 OpenMP threads, the Sandy Bridge cluster is running 2 hyperthreads per core, while the power6 cluster is running on 32 physical cores. 64-thread measurement is possible only on the power6 system.

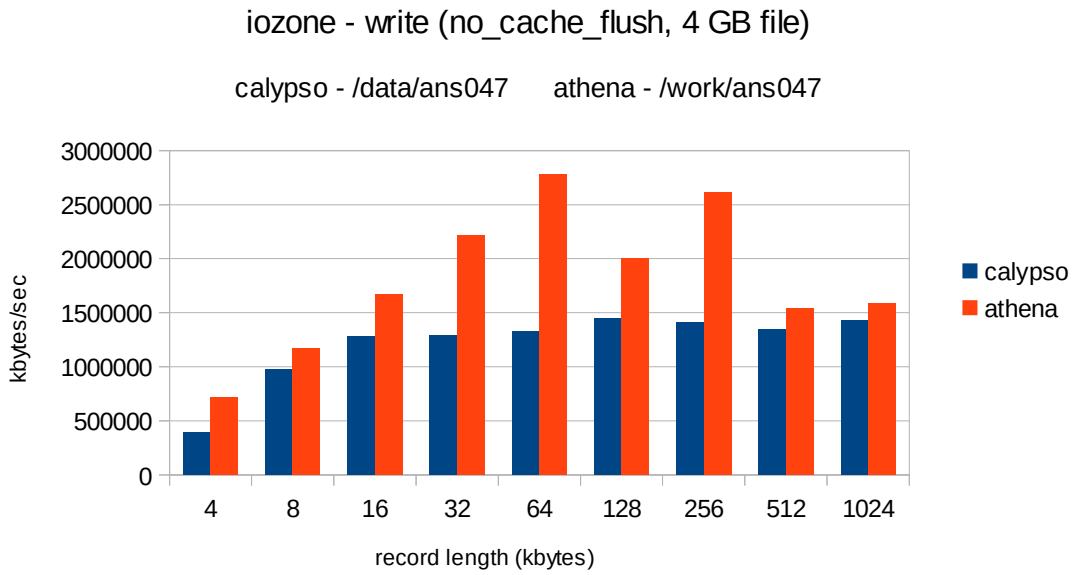
The systems share very similar memory bandwidth characteristics, highlighting 1) one of the architectural highlights of the power6 architecture as far back as 2006, that is to say its memory bandwidth, and 2) the importance of the collective performance improvements gained from the Intel Xeon Sandy Bridge processor architecture, interconnect, and software development environment. There is a dropoff in memory bandwidth for both systems when running two threads per core (at 32 threads for the x86 system, 64 threads for the power6).



Parallel Filesystem

Both calypso and athena utilize the parallel global filesystem GPFS. Tables below show performance measurements taken during production operation of both clusters; these benchmarks would ideally have been performed on a quiescent filesystem with caching disabled at both the filesystem, operating system and hardware levels, and are therefore included solely for the benefit of reference.





Code Performance

As-is runs (no modifications to source code) on both the P575 and iDataplex clusters returned the following representative timings (these timings are generated through the "timing function compiled/invoked as part of PELAGOS):

calypso - total timing (sum)

---

elapsed time (s) CPU time (s)

44535.508 **43450.050**

section	elap. time(%)	CPU time(s)	CPU time(%)	freq
trc_trp_bfm	38.18	133.29	39.04	150
tra_adv_muscl	32.89	116.36	34.08	7500
trc_stp	10.15	35.55	10.41	150
tra_ldf_iso	3.50	11.89	3.48	7500
sbc	2.41	4.61	1.35	150
tra_adv_eiv	2.05	7.19	2.31	150

athena - total timing (sum)

---

elapsed time (s) CPU time (s)

13226.180 **12757.744**

section	elap. time(%)	CPU time(s)	CPU time(%)	freq
trc_trp_bfm	31.08	31.63	32.01	150
tra_adv_muscl	30.66	31.35	31.72	7500
trc_stp	5.41	5.59	5.65	150
tra_sbc	4.13	4.22	4.27	150
sbc	4.06	2.31	2.34	150
sol_pcg	3.41	3.48	3.52	150
tra_ldf_iso	2.73	2.78	2.81	7500

A move to the x86 ecosystem, the Sandy Bridge architecture in particular, accounting for only latest-generation hardware and software tuned for the new hardware, yields a more than **3x** speedup in performance compared to the power6 architecture, with a clock frequency which is roughly 45% (with caveat regarding effects of turbo mode) lower than that of the power6 processor.

Instrumentation

Measuring application performance/behaviour ideally involves two types of measurements: profiling of the application itself, and instrumentation of identified problem spots within the code. Each has its own advantages and disadvantages, and an iterative approach to find the ideal mix is normally required.

Profiling/instrumentation of executables on calypso is limited by a lack of tools; the user-written module “timing.F90” can be called from within other model modules, and provides detailed profiling information; there is otherwise currently only one performance suite available on calypso; this is the IBM High Performance Computing Toolkit (HPCT). HPCT is a suite of tools and libraries used in application tuning.

There are two methods to access hardware performance counter information of the HPM; by way of the “hpmcount” utility, which measures performance metrics at the application itself, and through the use of “libhpm”, which allows for the instrumentation of selected code sections. There are various event “sets” which can be specified to collect different metrics, but attention should be paid to both the number of sets invoked concurrently as well as, in the case of libhpm calls, where system calls are placed, in order to avoid erroneous data due issues such as multiplexing. The command “pmlist –S -1” on calypso gives a listing of supported event sets.

Example 1:

The “hpmcount” utility is invoked in the following manner, in this case from the job submission script:

```
time mpirun.lsf hpmcount -s 0 -o hpmcount_32x128_dedi ${ocemod}.xx -np $NB_PROC  || {
```

The example below is the output generated from one of 128 processes generated from a run of PELAGOS:

```
Execution time (wall clock time): 384.105285 seconds

########  Resource Usage Statistics  ########

Total amount of time in user mode: 355.559891 seconds
Total amount of time in system mode: 1.192089 seconds
Maximum resident set size: 282268 Kbytes
Average shared memory use in text segment: 2547232 Kbytes*sec
Average unshared memory use in data segment: 93397543 Kbytes*sec
Number of page faults without I/O activity: 4654
Number of page faults with I/O activity :127
Number of times process was swapped out : 0
Number of times file system performed INPUT: 0
Number of times file system performed OUTPUT: 0
Number of IPC messages sent: 0
Number of IPC messages received: 0
```

Number of signals delivered: 0
Number of voluntary context switches: 89254
Number of involuntary context switches: 840

####### End of Resource Statistics ########

Set: 1
Counting duration: 30.010960527 seconds

PM_FPU_1FLOP (FPU executed one flop instruction)	:	1231508306
PM_FPU_FMA (FPU executed multiply-add instruction)	:	182748975
PM_FPU_FSQRT_FDIV (FPU executed FSQRT or FDIV instruction)	:	112884527
PM_CYC (Processor cycles)	:	139904224355
PM_RUN_INST_CMPL (Run instructions completed)	:	87389847860
PM_RUN_CYC (Run cycles)	:	141145144842

Set: 2
Counting duration: 30.027090136 seconds

PM_INST_CMPL (Instructions completed)	:	88299460563
PM_LSU_LDF (LSU executed Floating Point load instruction)	:	1672571191
PM_FPU_STF (FPU executed store instruction)	:	683154948
PM_CYC (Processor cycles)	:	139968028241
PM_RUN_INST_CMPL (Run instructions completed)	:	88513924021
PM_RUN_CYC (Run cycles)	:	141220917518

Set: 3
Counting duration: 30.037697480 seconds

PM_CYC (Processor cycles)	:	140560116883
PM_LD_MISS_L1 (L1 D cache load misses)	:	119044746
PM_ST_MISS_L1 (L1 D cache store misses)	:	494188860
PM_INST_CMPL (Instructions completed)	:	88256703289
PM_RUN_INST_CMPL (Run instructions completed)	:	88378678245
PM_RUN_CYC (Run cycles)	:	141271328153

Set: 4
Counting duration: 30.031874269 seconds

PM_INST_CMPL (Instructions completed)	:	87699690382
PM_INST_DISP (Instructions dispatched)	:	101736824111
PM_LD_REF_L1 (L1 D cache load references)	:	9586889722
PM_ST_REF_L1 (L1 D cache store references)	:	13188070899
PM_RUN_INST_CMPL (Run instructions completed)	:	87772625303
PM_RUN_CYC (Run cycles): 141241247215		

Set: 5
Counting duration: 29.951386654 seconds

PM_ST_REF_L1 (L1 D cache store references)	:	13420463189
PM_LD_REF_L1 (L1 D cache load references)	:	27052375400
PM_ST_MISS_L1 (L1 D cache store misses)	:	495255564
PM_LD_MISS_L1 (L1 D cache load misses)	:	115515286
PM_RUN_INST_CMPL (Run instructions completed)	:	89719329012
PM_RUN_CYC (Run cycles): 140865412174		

Set: 6
Counting duration: 29.880046169 seconds

PM_DATA_FROM_MEM_DP (Data loaded from double pump memory)	:	0
PM_DATA_FROM_DMEM (Data loaded from distant memory)	:	0
PM_DATA_FROM_RMEM (Data loaded from remote memory)	:	2763
PM_DATA_FROM_LMEM (Data loaded from local memory)	:	89476051
PM_RUN_INST_CMPL (Run instructions completed)	:	88103107152
PM_RUN_CYC (Run cycles)	:	140529722525

Set: 7
Counting duration: 29.912201435 seconds

PM_FPU_FIN (FPU produced a result)	:	1656243877
PM_CYC (Processor cycles)	:	140017198890
PM_FXU0_FIN (FXU0 produced a result)	:	13720084367
PM_FXU1_FIN (FXU1 produced a result)	:	25501346088
PM_RUN_INST_CMPL (Run instructions completed)	:	88120704398
PM_RUN_CYC (Run cycles)	:	140681203102

Set: 8
Counting duration: 29.899893833 seconds

PM_DATA_FROM_L2 (Data loaded from L2)	:	21045047
PM_DATA_FROM_L21 (Data loaded from private L2 other core)	:	7168
PM_DATA_FROM_L25_MOD (Data loaded from L2.5 modified)	:	97875
PM_DATA_FROM_L25_SHR (Data loaded from L2.5 shared)	:	55533
PM_RUN_INST_CMPL (Run instructions completed)	:	88028458225
PM_RUN_CYC (Run cycles)	:	140623390625

Set: 9
Counting duration: 29.930968257 seconds

PM_DATA_FROM_L35_MOD (Data loaded from L3.5 modified)	:	0
PM_DATA_FROM_L35_SHR (Data loaded from L3.5 shared)	:	264
PM_DATA_FROM_L3 (Data loaded from L3)	:	6359047
PM_CYC (Processor cycles)	:	140110826303
PM_RUN_INST_CMPL (Run instructions completed)	:	87112976895
PM_RUN_CYC (Run cycles)	:	140768513241

Set: 10
Counting duration: 29.923744150 seconds

PM_FPU_1FLOP (FPU executed one flop instruction)	:	1259747290
PM_FPU_FMA (FPU executed multiply-add instruction)	:	185093545
PM_FPU_STF (FPU executed store instruction)	:	677842029
PM_LD_MISS_L1 (L1 D cache load misses)	:	120537847
PM_RUN_INST_CMPL (Run instructions completed)	:	87071752606
PM_RUN_CYC (Run cycles)	:	140736377563

Set: 11
Counting duration: 29.885686261 seconds

PM_LD_MISS_L1 (L1 D cache load misses)	:	119696533
PM_CYC (Processor cycles)	:	140096286433
PM_LSU_LDF (LSU executed Floating Point load instruction)	:	1671834225
PM_ST_MISS_L1 (L1 D cache store misses)	:	528484077
PM_RUN_INST_CMPL (Run instructions completed)	:	88006555828
PM_RUN_CYC (Run cycles)	:	140555637540

Set: 12
Counting duration: 29.873944193 seconds

PM_INST_CMPL (Instructions completed)	:	87093949993
PM_L2_MISS (L2 cache misses)	:	502575352
PM_INST_FROM_L3MISS (Instruction fetched missed L3)	:	5186
PM_DATA_FROM_L3MISS (Data loaded from private L3 miss)	:	89911282
PM_RUN_INST_CMPL (Run instructions completed)	:	87202322737
PM_RUN_CYC (Run cycles)	:	140500112064

PM_FPU_1FLOP (FPU executed one flop instruction)	:	14937444026
PM_FPU_FMA (FPU executed multiply-add instruction)	:	2205565363
PM_FPU_FSQRT_FDIV (FPU executed FSQRT or FDIV instruction)	:	1351732934
PM_CYC (Processor cycles)	:	1680173900294
PM_RUN_INST_CMPL (Run instructions completed)	:	1055420282282
PM_RUN_CYC (Run cycles)	:	1690139006562
PM_INST_CMPL (Instructions completed)	:	1052449427967
PM_LSU_LDF (LSU executed Floating Point load instruction)	:	20060227126
PM_FPU_STF (FPU executed store instruction)	:	8158274291
PM_LD_MISS_L1 (L1 D cache load misses)	:	1424264138
PM_ST_MISS_L1 (L1 D cache store misses)	:	6069457782
PM_INST_DISP (Instructions dispatched)	:	1217396678693
PM_LD_REF_L1 (L1 D cache load references)	:	219509366188
PM_ST_REF_L1 (L1 D cache store references)	:	159414290472
PM_DATA_FROM_MEM_DP (Data loaded from double pump memory)	:	0
PM_DATA_FROM_DMEM (Data loaded from distant memory)	:	0
PM_DATA_FROM_RMEM (Data loaded from remote memory)	:	33230
PM_DATA_FROM_LMEM (Data loaded from local memory)	:	1076123009
PM_FPU_FIN (FPU produced a result)	:	19898130844
PM_FXU0_FIN (FXU0 produced a result)	:	164833233630
PM_FXU1_FIN (FXU1 produced a result)	:	306373432202
PM_DATA_FROM_L2 (Data loaded from L2)	:	252939483
PM_DATA_FROM_L21 (Data loaded from private L2 other core)	:	86151
PM_DATA_FROM_L25_MOD (Data loaded from L2.5 modified)	:	1176355

PM_DATA_FROM_L25_SHR (Data loaded from L2.5 shared)	:	667448
PM_DATA_FROM_L35_MOD (Data loaded from L3.5 modified)	:	0
PM_DATA_FROM_L35_SHR (Data loaded from L3.5 shared)	:	3169
PM_DATA_FROM_L3 (Data loaded from L3)	:	76349753
PM_L2_MISS (L2 cache misses)	:	6045677737
PM_INST_FROM_L3MISS (Instruction fetched missed L3)	:	62384
PM_DATA_FROM_L3MISS (Data loaded from private L3 miss)	:	1081578381

Utilization rate	:	92.990 %
MIPS	:	2740.002 MIPS
Instructions per cycle	:	0.626
Flop	:	24755.506 Mflop
Flop rate (flops / WCT)	:	64.450 Mflop/s
Flops / user time	:	69.308 Mflop/s
Floating point load and store operations	:	28218.501 M
Instructions per floating point load/store	:	37.296
% Instructions dispatched that completed	:	86.451 %
Total L2 data cache accesses	:	7493.722 M
% accesses from L2 per cycle	:	0.446 %
L2 traffic	:	914760.977 MBytes
L2 bandwidth per processor	:	2381.537 MBytes/s
Total load and store operations	:	378923.657 M
Instructions per load/store	:	2.777
Number of stores per store miss	:	26.265
Number of loads per load miss	:	154.121
Memory load traffic	:	6814941.234 MBytes
Memory load bandwidth per processor	:	43776.907 MBytes/s
Number of loads from local memory per loads from remote memory	:	32384.081
Number of loads from local memory per loads from remote and distant memory:		32384.081
% loads from memory per cycle	:	0.064 %
Total HW floating point instructions	:	19898.131
HW floating point instructions per Cycle	:	0.012
HW floating point instructions / user time	:	55.709 M HWflops/s
HW floating point rate (HW Flops / WCT)	:	51.804 M HWflops/s
Number of load/stores dispatched per L1 miss	:	50.565
L1 cache hit rate	:	98.022 %
Total Fixed point operations	:	471206.666
Fixed point operations per Cycle	:	0.280
Total Loads from L2	:	254.869 M
% loads from L2 per cycle	:	0.015 %
L2 load traffic	:	31111.992 MBytes/s
L2 load bandwidth per processor	:	80.999 MBytes/s
Total loads from L3	:	76.353 M
% loads from L3 per cycle	:	0.005 %
L3 load traffic	:	9320.425 MBytes/s
L3 load bandwidth per processor	:	24.265 MBytes/s
FMA percentage	:	25.731 %
Floating point Computation intensity	:	0.608
L2 load miss rate	:	80.677 %
L3 load miss rate	:	17.891 %

Example 2:

When utilizing “libphm” instrumented code must be compiled/linked with the libraries libhpm, libpmapi, and libm:

%USER_LIB %NCDF_LIB -L/usr/lib -lhpm -lpmapi –lm

Additionally, the following calls are utilized within the code itself to initialize/initiate sampling:


```
#include "f_hpm.h"
call f_hpminit(task_id, program_tag)
          call f_hpmstart(task, tag)
          call f_hpmstop(task)
call f_hpmterminate(task_id)
```

In this particular instance, the hpminit() and hpmterminate are invoked within nemo.f90, while hpmstart() and hpmstop() are inserted into every section of trc_bfm.F90 where instrumentation is desired. (see code below).

The following example illustrates usage of the libhpm library and associated calls, as well as a memory bottleneck within the trc_trp_bfm routine of PELAGOS (highlighted in blue).

```
#if defined D1SOURCE
        dummy(:) = D3SOURCE(m,:)
#else
call f_hpmstart(10, "trc_trp_bfm-bfm_tracers")
        do k=1,NO_D3_BOX_STATES
          do n=1,NO_BOXES
#ifdef ONESOURCE
            dummy(n) = dummy(n) + D3SOURCE(m,k,n)
#else
            dummy(n) = dummy(n) + D3SOURCE(m,k,n) - D3SINK(m,k,n)
#endif
          end do
        end do
call f_hpmstop(10)
#endif

         ! Add biogeochemical trends
         tra(:,:,:,1) = unpack(dummy,SEAmask,ZEROS)
#ifdef DEBUG
        LEVEL2 'trc_trp_bfm at',kt
        CALL prxy( LOGUNIT, 'trn for '//trim(var_names(m)),trn(:,:,1,1), jpi, 1, jpj, 1, ZERO)
        CALL prxy( LOGUNIT, 'tra:BIO',tra(:,:,1,1), jpi, 1, jpj, 1, ZERO)
```

The following is a report from one of the 128 processes run in this instance (in a libhpm instrumentation run, each process generates its own report).

Total execution time of instrumented code (wall time) : 351.919338 seconds

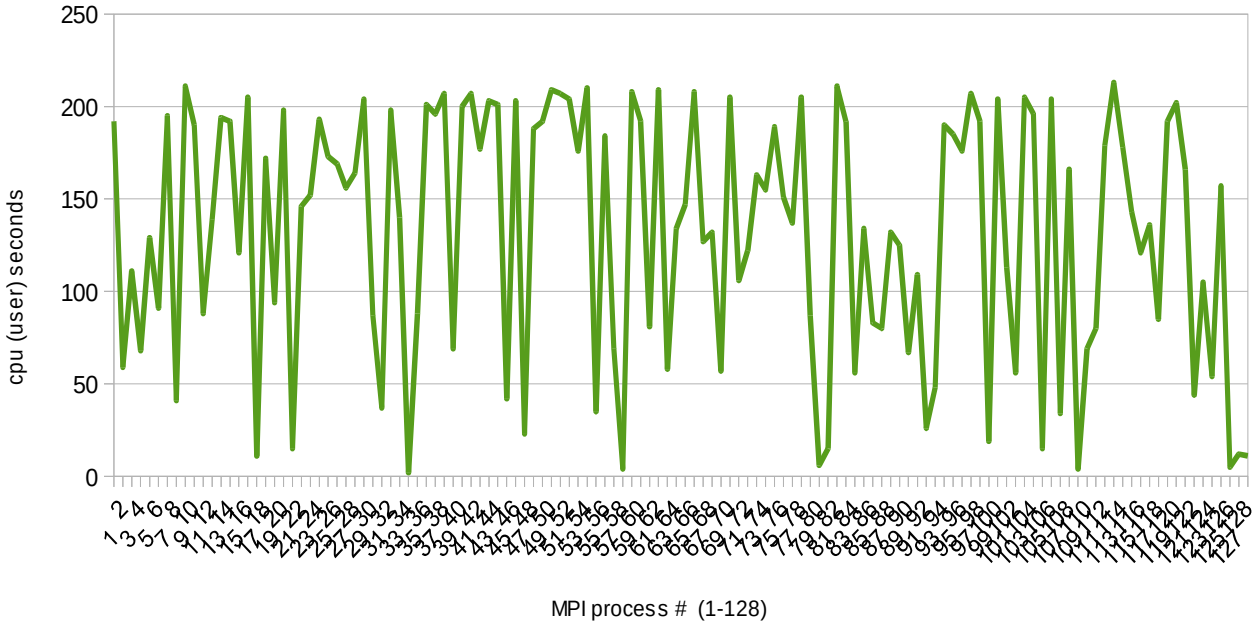
######## Resource Usage Statistics ########

Total amount of time in user mode	: 341.300645 seconds
Total amount of time in system mode	: 0.938681 seconds
Maximum resident set size	: 392168 Kbytes
Average shared memory use in text segment	: 2514068 Kbytes*sec
Average unshared memory use in data segment	: 127422823 Kbytes*sec
Number of page faults without I/O activity	: 6572
Number of page faults with I/O activity	: 157
Number of times process was swapped out	: 0
Number of times file system performed INPUT	: 0
Number of times file system performed OUTPUT	: 0
Number of IPC messages sent	: 0
Number of IPC messages received	: 0
Number of signals delivered	: 0
Number of voluntary context switches	: 9727
Number of involuntary context switches	: 467

```
#######  End of Resource Statistics  ########

Instrumented section: 10 - Label: trc_trp_bfm-bfm_tracers – process:   1
file: trc_trp_bfm.F90, lines: 109 <--> 119
 Count                                          : 7350
 Wall Clock Time                                : 193.112493  seconds
 Total time in user mode                        : 192.904707738946  seconds
 Average duration                               : 0.0262738
 Standard deviation                             : 5.24414e-315

Set: 5
Counting duration:   0.021848128  seconds
 PM_ST_REF_L1 (L1 D cache store references)              :    2863152531
 PM_LD_REF_L1 (L1 D cache load references)               :    5726792176
 PM_ST_MISS_L1 (L1 D cache store misses)                 :     329346
 PM_LD_MISS_L1 (L1 D cache load misses)                  :    2146220731
 PM_RUN_INST_CMPL (Run instructions completed)           :   16565879367
 PM_RUN_CYC (Run cycles)                                 :  907423745204

 Total L2 data cache accesses                   :      2146.550  M
 L2 traffic                                     :    262030.039  MBytes
 L2 bandwidth per processor                     :      1356.878  MBytes/s
 Total load and store operations                :      8589.945  M
 Number of stores per store miss                :      8693.449
 Number of loads per load miss                  :         2.668
 Number of load/stores dispatched per L1 miss   :         4.002
 L1 cache hit rate                              :        75.011  %
```

Of particular note for these 10 lines of code within trc_trp_bfm is the very low instructions per cpu-cycle (IPC) which may be an indication of a significant number of stalls due to memory misses, and the very low L1 cache hit rate of 75% (highlighted in red). This may also explain the additional drop in performance seen during runs with 2 SMT threads per core (SMT enabled); with two threads contending for cache on each core, the poor cache hit efficiency is magnified.

Depending on the computation being performed by each of the 128 processes, this particular section of code within trc_trp_bfm consumes anywhere from 5 to 207 seconds



Example 3:

This example illustrates usage of the libhpm library and associated calls, as well as floating point performance of a section of the traadv_muscl routine:

```
      !                                !-- Slopes of tracer
      zslpx(:,:,1) = 0.e0                        ! surface values
!call f_hpmstart(70, "muscl_70")
      DO jk = 2, jpkm1                           ! interior value
        DO jj = 1, jpj
          DO ji = 1, jpi
            zslpx(ji,jj,jk) =                ( zwx(ji,jj,jk) + zwx(ji,jj,jk+1) )   &
              &          * ( 0.25 + SIGN( 0.25, zwx(ji,jj,jk) * zwx(ji,jj,jk+1) ) )
          END DO
        END DO
      END DO
!call f_hpmstop(70)
      !                                !-- Slopes limitation
call f_hpmstart(80, "muscl_80")
      DO jk = 2, jpkm1                           ! interior values
        DO jj = 1, jpj
          DO ji = 1, jpi
            zslpx(ji,jj,jk) = SIGN( 1., zslpx(ji,jj,jk) ) * MIN(    ABS( zslpx(ji,jj,jk
) ),   &
              &                                 2.*ABS( zwx  (ji,jj,jk+1) ),   &
              &                                 2.*ABS( zwx  (ji,jj,jk
) ) )
          END DO
        END DO
      END DO
call f_hpmstop(80)
      !                                !-- vertical advective flux
      !                                     ! surface values  (bottom already set to zero)
      IF( lk_vvl ) THEN    ;   zwx(:,:, 1 ) = 0.e0                 !  variable volume
      ELSE               ;   zwx(:,:, 1 ) = pwn(:,:,1) * ptb(:,:,1,jn)   ! linear free surface
      ENDIF
      !
call f_hpmstart(90, "muscl_90")
      DO jk = 1, jpkm1                           ! interior values
        zdt  = p2dt(jk)
        DO jj = 2, jpjm1
          DO ji = fs_2, fs_jpim1   ! vector opt.
            zbtr = 1. / ( e1t(ji,jj) * e2t(ji,jj) * fse3w(ji,jj,jk+1) )
            z0w = SIGN( 0.5, pwn(ji,jj,jk+1) )
            zalpha = 0.5 + z0w
            zw  = z0w - 0.5 * pwn(ji,jj,jk+1) * zdt * zbtr
            zzwx = ptb(ji,jj,jk+1,jn) + zw * zslpx(ji,jj,jk+1)
            zzwy = ptb(ji,jj,jk  ,jn) + zw * zslpx(ji,jj,jk  )
            zwx(ji,jj,jk+1) = pwn(ji,jj,jk+1) * ( zalpha * zzwx + (1.-zalpha) * zzwy )
          END DO
        END DO
      END DO
call f_hpmstop(90)
```

The following is a report from one of the 128 processes run in this instance (as previously described, in this particular libhpm instrumentation run, each process generates its own report).

Total execution time of instrumented code (wall time): 366.823533 seconds

######## Resource Usage Statistics ########

Total amount of time in user mode	: 351.006301 seconds
Total amount of time in system mode	: 1.153869 seconds
Maximum resident set size	: 275432 Kbytes
Average shared memory use in text segment	: 2578012 Kbytes*sec
Average unshared memory use in data segment	: 89722782 Kbytes*sec
Number of page faults without I/O activity	: 4709
Number of page faults with I/O activity	: 138
Number of times process was swapped out	: 0
Number of times file system performed INPUT	: 0
Number of times file system performed OUTPUT	: 0
Number of IPC messages sent	: 0
Number of IPC messages received	: 0
Number of signals delivered	: 0
Number of voluntary context switches	: 13103
Number of involuntary context switches	: 576

####### End of Resource Statistics ########

Instrumented section: 80 - Label: muscl_80 - process: 1
file: traadv_muscl.F90, lines: 226 <--> 236
Count: 7650
Wall Clock Time: 0.880342000000009 seconds
Total time in user mode: 0.68550468494898 seconds
Average duration: 0.000115077
Standard deviation: 5.21641e-315

Set: 1
Counting duration: 0.700275212 seconds

PM_FPU_1FLOP (FPU executed one flop instruction)	:	475202700
PM_FPU_FMA (FPU executed multiply-add instruction)	:	0
PM_FPU_FSQRT_FDIV (FPU executed FSQRT or FDIV instruction)	:	0
PM_CYC (Processor cycles)	:	3224614038
PM_RUN_INST_CMPL (Run instructions completed)	:	1433126284
PM_RUN_CYC (Run cycles)	:	3237461386

Utilization rate	:	77.868 %
Flop	:	475.203 Mflop
Flop rate (flops / WCT)	:	539.793 Mflop/s
Flops / user time	:	693.216 Mflop/s
FMA percentage	:	0.000 %

Instrumented section: 90 - Label: muscl_90 - process: 1
file: traadv_muscl.F90, lines: 243 <--> 258
Count: 7650
Wall Clock Time: 0.732532000000016 seconds
Total time in user mode: 0.624671786564626 seconds
Average duration: 9.57558e-05
Standard deviation: 5.21384e-315

Set: 1
Counting duration: 0.639162550 seconds

PM_FPU_1FLOP (FPU executed one flop instruction)	:	433755000
PM_FPU_FMA (FPU executed multiply-add instruction)	:	192780000
PM_FPU_FSQRT_FDIV (FPU executed FSQRT or FDIV instruction)	:	48195000
PM_CYC (Processor cycles)	:	2938456084
PM_RUN_INST_CMPL (Run instructions completed)	:	2116578224
PM_RUN_CYC (Run cycles)	:	2950027259

Utilization rate	:	85.276 %
Flop	:	1012.095 Mflop
Flop rate (flops / WCT)	:	1381.639 Mflop/s
Flops / user time	:	1620.203 Mflop/s
FMA percentage	:	61.538 %

Floating point performance in this computationally intensive module varies, with upwards of around 10% of peak floating point performance (power6 processor) being the high boundary.

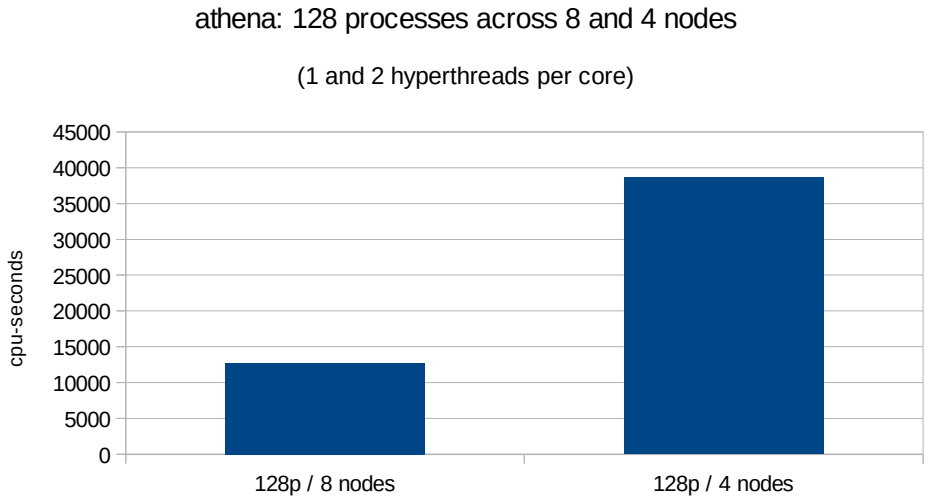
Unfortunately, at the time of this writing, no instrumentation on a per-routine level is available for runs on athena.

Hyperthreading

The E5-2670 supports HyperThreading (HT), akin to symmetric multithreading (SMT) of the power6. Each Sandy Bridge core can support 2 hyperthreads, which means up to 16 virtual cores are available per physical socket, and 32 hyperthreads per physical athena node.

Hyperthreading is of value primarily from a throughput standpoint; it may be possible to run heterogeneous workloads concurrently (in parallel on the same core) so that system throughput is improved. Hyperthreading should not be used if the primary performance concern is performance on a core basis.

Preliminary tests running PELAGOS in hyperthreaded mode (32 processes per physical node, ie 2 hyperthreads per core) did not show encouraging results While it is likely that with additional tuning the rate can be reduced, unless it can be reduced to no more than a 2:1 ratio, enabling hyperthreading when running multiple PELAGOS instances will not be of value. It is not clear whether more heterogeneous hyperthreaded workloads, where the applications exercise various units of the processor in different manners, would be worth considering.



Vectorization

Dedicated vector processors seem to have lost favor, but programming to optimize vector performance remains an important consideration in application performance.

The Sandy Bridge processor evolves Intel vector capability on x86 processors, incorporating a new instruction set and 256-bit vector registers.

Various loops within routines such as traadv_muscl (see example 3 above) demonstrate upwards of 10% of peak floating point on the power6 processor of calypso; unfortunately at this time there are no similar performance numbers for Sandy Bridge on athena. However, attempts at autovectorization through the compiler results in significant inability to vectorize loops, because of loop dependencies, unsupported loop structures and other constraints.

An attempt at loop reordering in order to further increase vectorization efficiency could be undertaken, but at the moment it is unclear as to how much of an effort this would be, and how much benefit would actually be derived.

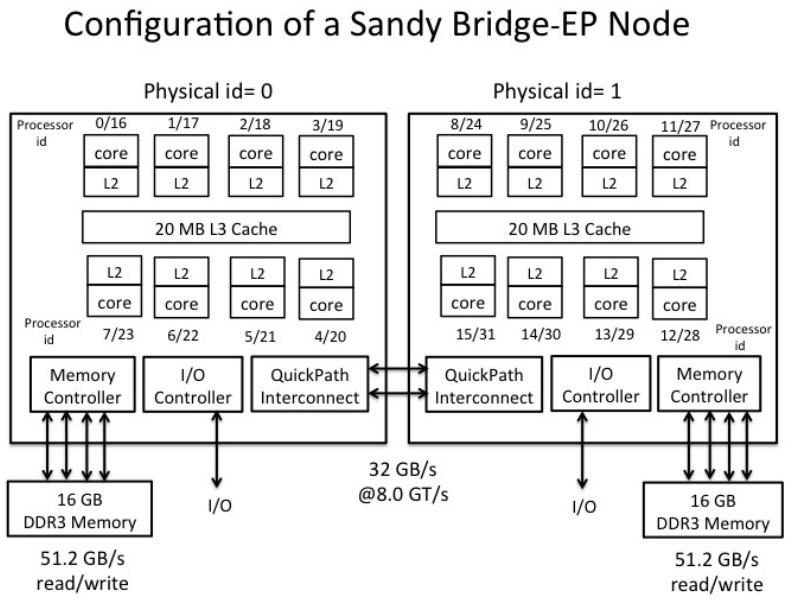
Additional architectural considerations

The significant improvement in application performance observed between the power6 and xeon-based clusters at CMCC can be attributed to general improvements in processor, memory, interconnect and software development environments, with particular consideration to improvements in the processor architecture. The Intel Xeon E5-2670 processor utilized in athena, also known by the Intel codename “Sandy Bridge”, benefits from improvements in out-of-order instruction execution, improved floating point arithmetic, but last but not least an improved and efficient cache management system.

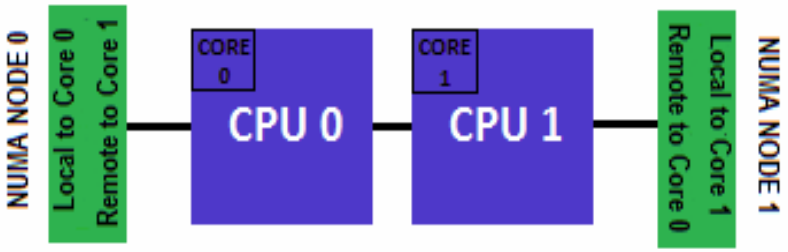
The Intel Sandy Bridge processor populating athena nodes is specifically designed for high-performance server workloads. It has several features optimizing processor performance, including

- Highly efficient out-of-order instruction execution and branch prediction - the processor used on the athena cluster is able to efficiently dispatch instructions as soon as the operands they need are ready, minimizing wasted processor cycles. This occurs for all types of instructions, whereas the power6 processor supports out-of-order processing for only a subset of (floating point) instructions.
- Improved caching and memory subsystems – Sandy Bridge offers improved memory load/cycle performance. There is also a ring-type interconnect linking last-level cache, memory controllers, and cores and offering high bandwidth.
- Dual QuickPath Interconnects (QPI) between sockets on a node
- On-die PCIe3
- Support for 1600 MHz memory
- Improved vector instruction set (AVX)
- Turbo Boost – a form of automatic, temporary overclocking ability, which based on several system parameters can for a time increase the processor clock speed. Depending on application type, this can result in a 10-12% performance improvement in single-thread performance per core.

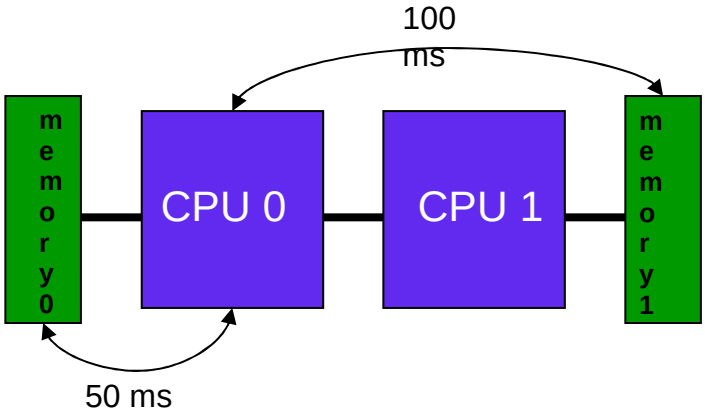
Below is a schematic overview of a 2-socket Sandy Bridge configuration, as found on a single physical athena compute node:



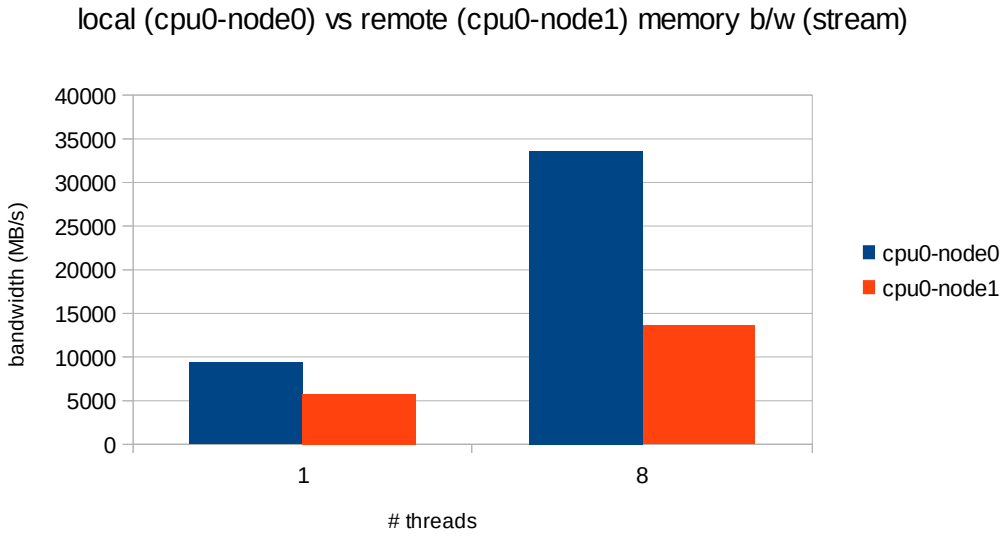
Each "physical" athena node is comprised of 2 non-uniform memory access (NUMA) nodes (also sometimes referred to as domains); all 16 cores (8 cores per socket, or CPU) have access to all memory on the physical node, but memory access is non-uniform.



This means that accessing memory outside of a local NUMA node involves increased latencies:



The stream memory bandwidth test illustrated this latency. In the chart below "node" is defined as the memory of the NUMA node, so that by cpu0-node0 it is intended a core accessing memory local to it, while cpu0-node1 means a core accessing memory from the other socket (across the QPI, the link connecting the two sockets):



Memory affinity/pinning and its consequences (positive or negative) should therefore not be overlooked/underestimated during program execution.

Final considerations

There are 3 possible ways to approach performance improvements on the PELAGOS model:

1. coarse-grained optimization/parallelization at the application level (message passing between nodes)
2. medium-grained shared memory optimization/parallelism at the node level
3. fine-grained optimization/parallelism improvements (vectorization)

Point #1, scaling up the number of processors, is likely the area in which the greatest performance can be achieved. This coarse-grained parallelism is beyond the scope of this exercise, and there are already works underway with collective vs point-to-point operations, differing MPI distributions, etc.

Point #2 may possibly be improved through the following means:

- rewriting of BFM to better utilize cache-based memory hierarchies; instrumentation runs on the power6 indicate inefficient use of the memory hierarchy by some BFM routines, in particular trc_trp_bfm. While performance improvements on athena indicate that performance improvement is as much about newer generation hardware and software

(compilers) as it is about effectively-written code, if the effort in man-hours needed to perform a rewrite of BFM modules to better take advantage of the memory hierarchy of cache-based processors is not great, it would be worth the effort to attempt a rewrite of code in order to maximize memory hierarchy efficiency. It is however almost a prerequisite to have knowledge of the processor architecture as well as of the code in order to extract maximum benefit from this endeavor.

- Hyperthreading on the Sandy Bridge (and future Intel processors) may lend itself to the integration of OpenMP into the model, or to other programming models beyond strict-MPI (see “Futures” below).

Point #3, vectorization improvements, may be the targeted optimization effort with the least tangible benefits, but activities such as loop rewriting for reordering (in order to more effectively leverage the vector capability of Sandy Bridge and future Intel/non processors) and expanded use of scientific libraries where applicable (to indirectly leverage the vector capability) are worth investigating.

Futures

Two additional items which can also be considered regarding PELAGOS computation platforms going forward might be:

1. programming model
2. accelerators

Programming model: With regards to programming models, if code rewrites are implemented, it might be instructive to investigate the possibilities of moving PELAGOS to a new programming paradigm beyond its current SPMD/MPI model, in particular the partitioned global address space (PGAS) model. PGAS is a parallel programming model employing a combination of physical local and logical globally addressable memory space. It is supposedly well-suited to the SPMD programming style employed by PELAGOS, incorporating advantages from both MPI and shared-memory programming systems. PGAS constructs are already incorporated into compilers such as the Intel Fortran Compiler currently installed on athena, but it should be underlined that PGAS is not yet considered mature.

Accelerators: Computational accelerators are another possible avenue of future performance enhancement. The most well-known and established accelerator at the moment is the GPGPU. Getting codes to run effectively on GPGPU means not only optimizing (yet again?) for the architecture of the GPGPU, but generating GPU code (OpenCL? CUDA?) from existing code. This can cause portability issues which need to be considered, especially within communities such as those running climate simulations. An alternative to the GPGPU is the Intel Many Integrated Core (MIC) architecture, known as Xeon Phi. While also likely requiring code restructuring to effectively leverage its many x86-core architecture, it is possible to run Fortran and C, MPI and OpenMP codes on the Xeon Phi. It may take a generation or two before the Xeon Phi architecture achieves its full potential, but it would be useful, and relatively cost-effective, to have a small test-bed cluster of Xeon Phi-equipped nodes (perhaps even just a couple) available for evaluation purposes (should there be an interest).

20

Centro Euro-Mediterraneo sui Cambiamenti Climatici

Visit **www.cmcc.it** for information on our activities and publications.

The Euro-Mediteranean Centre on Climate Change is a Ltd Company with its registered office and administration in Lecce and local units in Bologna, Venice, Capua, Sassari, Viterbo, Benevento and Milan. The society doesn't pursue profitable ends and aims to realize and manage the Centre, its promotion, and research coordination and different scientific and applied activities in the field of climate change study.

cmcc
Centro Euro-Mediterraneo
sui Cambiamenti Climatici