Intel Xeon Phi Coprocessor on Helios Cluster

C.Berthelot
Christophe.Berthelot@bull.net

Copyright ©Bull, an Atos compagny 2016
©Bull. All rights reserved

✔ Users Restricted Rights - Use, duplication or disclosure restricted.
✔ Any copy of these documents should keep all copyright, logos and other proprietary notices contained herein.
✔ This publication may include technical inaccuracies or typographical errors.
✔ This publication is provided "AS IS" without any warranty either expressed or implied including but not limited to the implied warranties of merchantabilities or fitness of the described product.
✔ Course Material Licensing Terms : No sublicensing rights.
✔ For other licensing needs, please contact Bull
Agenda

- Introduction
- Helios and Intel Xeon Phi
- Benchmark MIPS
- Future
- Bibliography
Introduction

Broader Approach

ITER

B.A.

JT-60SA

IFERC

IFMIF-EVEDA
Introduction

The Satellite Tokamak Programme (STP): JT-60SA

JT-60SA is a fusion experiment designed to support the operation of ITER and to investigate how best to optimise the operation of fusion power plants that are built after ITER.
IFMIF, the International Fusion Materials Irradiation Facility, is an accelerator-based neutron source that will use Li(d,xn) reactions to generate a flux of neutrons with a broad peak at 14 MeV equivalent to the conditions of the Deuterium-Tritium reactions in a fusion power plant. IFMIF is conceived for fusion materials testing.
Introduction

International Fusion Energy Research Centre (IFERC)

The mission of International Fusion Energy Research Centre (IFERC) is to complement ITER and to contribute to an early realization of DEMO reactor.
Helios and Intel Xeon Phi

- Introduction
- Helios and Intel Xeon Phi
  - Hardware
  - Software
  - Programming Models
- Benchmark MIPS
- Future
- Bibliography
Hardware

- 180 Intel Xeon Phi Coprocessor

Performance Linpack: 225TF

<table>
<thead>
<tr>
<th>T/V</th>
<th>N</th>
<th>NB</th>
<th>P</th>
<th>Q</th>
<th>Time</th>
<th>Gflops</th>
</tr>
</thead>
<tbody>
<tr>
<td>WC01C2R4</td>
<td>940800</td>
<td>960</td>
<td>20</td>
<td>18</td>
<td>2465.87</td>
<td>2.25129e+05</td>
</tr>
</tbody>
</table>

HPL_pdgesv() start time Tue Jan 28 15:47:36 2014
HPL_pdgesv() end time Tue Jan 28 16:28:42 2014

||Ax-b||_oo/(eps*(||A||_oo*||x||_oo+||b||_oo)*N)= 0.0013965 ...... PASSED
Helios and Intel Xeon Phi

Hardware

Features of an Individual Core 1/2

- Up to 61 in-order cores (Ring interconnect) [1]
- 64-bit addressing
- Two pipelines:
  - Intel Pentium processor family-based scalar units
  - Pipelined one-per-clock scalar throughput (4 clock latency, hidden by round-robin scheduling of threads)
- 4 hardware threads per core

Features of an Individual Core 2/2

- Optimized (single and double precision)
- All new vector unit
  - 512-bit SIMD Instructions - not Intel SSE, MMX or Intel AVX
  - 32 512-bit wide vector registers
- Fully-coherent L1 and L2 caches
Coprocessor Overview - Cache

- **L1 Cache**
  - 32K I-cache per core
  - 32K D-cache per core
  - 3 cycles access
  - Up to 8 outstanding requests
  - Fully coherent

- **L2 Cache**
  - 512K Unified per core
  - Sum across all cores
  - 11 cycles best access
  - Up to 32 outstanding requests
  - Fully coherent

Coprocessor Overview - Memory and Alignment

- **Alignment**
  - Based upon the number of elements, elements size and vector load and store instruction
  - 64B alignment for 4B (float) data elements for 16 to 16 vector loads

- **Memory**
  - GDDR5
  - 16 Memory channels (up to 5.5Gbit each)
  - 8GB up to 16 GB
  - 300ns access
  - Memory Bandwidth 320GB/s
Helios and Intel Xeon Phi

Software Architecture: Two Modes
Helios and Intel Xeon Phi

Software Architecture: Two Modes
Helios and Intel Xeon Phi

Intel Development Environment

- Intel compilers
- OpenMP
- Intel MPI Library support the Intel Phi Coprocessor as an MPI node
- Intel Parallel Building Blocks (Intel Cilk Plus)
- Intel Trace Analyzer
- Intel Performance Libraries (Intel MKL), 3 version host-only, coprocessor-only, heterogeneous
- Intel Vtune Amplifier XE
- Standard runtime libraries, even pthreads, Gnu tools also supported (gcc as cross compiler of the host, gdb... GCC could be natively compiled on the target)
Helios and Intel Xeon Phi

MKL: DGEMM

![Graph showing performance of DGEMM for different sizes and configurations]
Benchmark MIPS

- Introduction
- Helios and Intel Xeon Phi
- Benchmark MIPS
  - Scalability
  - Profiling on Host
  - Profiling on Intel Xeon Phi Coprocessor
  - Profiling Itac
  - Remarks
  - To Sum up
- Future
- Bibliography
Benchmark MIPS

Benchmark MIPS

- MIPS: MHD (magnetohydrodynamic) Infrastructure for Plasma Simulation

\[ \frac{\partial \rho}{\partial t} = -\nabla \cdot (\rho v) \]

\[ \rho \frac{\partial}{\partial t} V = -\rho w \times V - \rho \nabla \left( \frac{v^2}{2} \right) - \nabla p + j \times B + \frac{4}{3} \nabla [\nu \rho (\nabla \cdot v)] - \nabla \times [\nu \rho w] \]

\[ \frac{\partial B}{\partial t} = -\nabla \times E \]

\[ \frac{\partial p}{\partial t} = -\nabla . (p V) - (\gamma - 1) \rho \nabla . v + (\gamma - 1) \left[ \nu \rho v^2 + \frac{4}{3} \nu \rho (\nabla \cdot v)^2 + \eta j \cdot (j - j_{eq}) \right] \]

\[ E = -v \times B + \eta (j - j_{rq}) \]

\[ j = \frac{1}{\mu_0} \nabla \times B \]

\[ w = \nabla \times v \]
Benchmark MIPS

- MIPS : MPI for parallelization
- Fortran code : 6000 lignes
- Fukuoka Daichi (Support Team) added OpenMP parallelization

Work

- First : Scalability
- Second : Profiling with Intel tools (Vtune and Itac)
- Third: Expertise with Thomas Guillet (Exascale Labs), Romain Dolbeau (CEPP)
Benchmark MIPS

Behavior without Intel Xeon Phi Coprocessor

- Test case name: 64EQ, with 16MPI
- Use between 1 and 8 nodes
- Goal test memory bandwidth
- Without OpenMP (red) and with OpenMP (green)
Benchmark MIPS

How Many Thread by MIC?

- Test case name 64EQ, with 16 MPI tasks, only one node
- Test between 10 and 240 threads by MPI task
- No more 4 MPI Task/MIC (to use 16 MPI task/Node)
Benchmark MIPS

Increase Number of MPI task

- Test case 64EQ
- Between 16 and 128 MPI tasks
- Host: 16 cores, red line
- Host+MIC: 8 MPI, 2 OpenMP/MPI + 2(4 MPI, 20 OpenMP/MPI)), green line
### Benchmark MIPS

- **CPI Rate**: 2.98 very high rate
- **A lot of trouble on Memory Replacement**

### Metrics

<table>
<thead>
<tr>
<th>Metric</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Elapsed Time</td>
<td>60.060s</td>
</tr>
<tr>
<td>Clockticks</td>
<td>1,889,758,954,834</td>
</tr>
<tr>
<td>Instructions Retired</td>
<td>634,046,951,966</td>
</tr>
<tr>
<td>CPI Rate</td>
<td>2.980</td>
</tr>
</tbody>
</table>

The CPI may be too high. This could be caused by issues such as memory stalls, instruction throttle, branch misprediction, or long latency instructions. Explore the other hardware-related metrics to identify what is causing high CPI.

- **Latency**: 1.000
- **Pipe Time**: 0s

#### Filled Pipeline Slots
- **Retiring**: 0.106
- **Bad Speculation**: 0.001

#### Unfilled Pipeline Slots (Stalls)
- **Back-end Bound**: 0.066

A significant proportion of pipeline stalls are remaining empty. When operations take too long, the back-end, they introduce hiccups in the pipeline that ultimately cause fewer pipeline stalls controlling useful work to be retired per cycle than the machine is capable of supporting. This opportunity cost results in lower execution. Long-latency operations like branches and memory operations can cause this, as too many operations being directed to a single execution port (for example, more multiply operations arriving in the back-end per cycle than the execution unit can support).

#### Memory Latency
- LLC Load Misses Serviced By Remote DRAM: 0.013
- LLC Miss: 0.027
- LLC Hit: 0.030
- DRB Overhead: 0.055
- Contained Accesses: 0.000
-LD miss: 0.000

#### Memory Replacements
- LLC Replacement (Percent): 1.000

This row is responsible for memory of all L1 cache replacements. Some replacements are unavoidable, and slight levels of replacements may not indicate a problem. Consider this metric only when looking for the source of a significant number of L1 cache misses for a particular grouping. If these replacements are marked as a problem, try rearranging data.

#### Memory Reuses
- Level2 Booted From Store Forwardings: 0.031
- Prefetch: 0.034
- RFO Store: 0.034
- Load Reuse: 0.014
- A significant proportion of cycles are spent handling split loads. Consider aligning your data to the 64-byte cache line granularity.
<table>
<thead>
<tr>
<th>Benchmark MIPS</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Function / Call Stack</th>
<th>Clockticks</th>
<th>Instructions Retired</th>
<th>CPI Rate</th>
<th>MUX Reliability</th>
<th>Filled Pipeline Slots</th>
<th>Unfilled Pipeline Slots (Stats)</th>
<th>Module</th>
</tr>
</thead>
<tbody>
<tr>
<td>mhib</td>
<td>368,912,553,368</td>
<td>48,524,072,786</td>
<td>7.603</td>
<td></td>
<td></td>
<td></td>
<td>mips</td>
</tr>
<tr>
<td>nalu15.......osmp@parallel@24</td>
<td>319,970,479,955</td>
<td>106,282,159,423</td>
<td>3.011</td>
<td></td>
<td></td>
<td></td>
<td>mips</td>
</tr>
<tr>
<td>dpixindex4...osmp@parallel@23</td>
<td>241,280,361,920</td>
<td>70,584,105,876</td>
<td>3.418</td>
<td></td>
<td></td>
<td></td>
<td>mips</td>
</tr>
<tr>
<td>dixion5...osmp@parallel@20</td>
<td>202,010,303,015</td>
<td>60,666,090,999</td>
<td>3.330</td>
<td></td>
<td></td>
<td></td>
<td>mips</td>
</tr>
<tr>
<td>n.1_field......osmp@parallel@18</td>
<td>137,346,206,019</td>
<td>16,512,024,768</td>
<td>8.318</td>
<td></td>
<td></td>
<td></td>
<td>mips</td>
</tr>
<tr>
<td>dixation3...osmp@parallel@25</td>
<td>135,838,203,757</td>
<td>111,698,167,547</td>
<td>1.216</td>
<td></td>
<td></td>
<td></td>
<td>mips</td>
</tr>
<tr>
<td>dixion3...osmp@parallel@22</td>
<td>115,990,173,985</td>
<td>11,936,179,094</td>
<td>6.971</td>
<td></td>
<td></td>
<td></td>
<td>mips</td>
</tr>
<tr>
<td>mhib_boundry</td>
<td>75,314,112,973</td>
<td>2,996,004,494</td>
<td>25.138</td>
<td></td>
<td></td>
<td></td>
<td>mips</td>
</tr>
<tr>
<td>dixation3...osmp@parallel@23</td>
<td>72,232,108,348</td>
<td>21,168,031,752</td>
<td>3.417</td>
<td></td>
<td></td>
<td></td>
<td>mips</td>
</tr>
<tr>
<td>dixion3...osmp@parallel@28</td>
<td>68,330,302,495</td>
<td>3,290,004,935</td>
<td>20.769</td>
<td></td>
<td></td>
<td></td>
<td>mips</td>
</tr>
<tr>
<td>dixion3...osmp@parallel@28</td>
<td>56,584,084,876</td>
<td>2,880,004,320</td>
<td>19.647</td>
<td></td>
<td></td>
<td></td>
<td>mips</td>
</tr>
<tr>
<td>MPID...wem...shmp...p01</td>
<td>35,290,052,935</td>
<td>50,158,075,237</td>
<td>6.704</td>
<td></td>
<td></td>
<td></td>
<td>mips</td>
</tr>
<tr>
<td>[D]etails any known module</td>
<td>21,670,032,505</td>
<td>3,430,005,145</td>
<td>6.318</td>
<td></td>
<td></td>
<td></td>
<td>mips</td>
</tr>
<tr>
<td>implements generic</td>
<td>9,962,014,943</td>
<td>18,000,027</td>
<td>553,444</td>
<td></td>
<td></td>
<td></td>
<td>mips</td>
</tr>
<tr>
<td>implements generic</td>
<td>5,704,006,556</td>
<td>12,000,018</td>
<td>475,333</td>
<td></td>
<td></td>
<td></td>
<td>mips</td>
</tr>
<tr>
<td>implements generic</td>
<td>5,138,007,977</td>
<td>8,328,012,492</td>
<td>6.369</td>
<td></td>
<td></td>
<td></td>
<td>mips</td>
</tr>
<tr>
<td>finfתח WEIGHTS</td>
<td>3,920,005,880</td>
<td>46,000,069</td>
<td>85.217</td>
<td></td>
<td></td>
<td></td>
<td>mips</td>
</tr>
<tr>
<td>Decon0.2...full</td>
<td>1,770,006,656</td>
<td>2,680,006,647</td>
<td>6.656</td>
<td></td>
<td></td>
<td></td>
<td>mips</td>
</tr>
</tbody>
</table>

Selected module: C.Berthelot Christophe.Berthelot@bull.net (C)Atos
Benchmark MIPS

Elapsed Time: 60.019s

- CPU Time: 4798.424s
- Clocks: 5,045,837,568,745
- Instructions Retired: 688,700,000,000

CPI: 7.327

The CPI may be too high. This could be caused by issues such as memory stalls, instruction starvation, branch misprediction or long latency instructions. Explore the other hardware-related metrics to identify what is causing high CPI.

- Vectorization Usage: 0.000
- L1 Usage: 0.000
- Pseudo Time: 0s

Cache Usage:

- L1 Hits: 21,926,750,000
- L1 Hit Rate: 0.878

The L1 cache hit ratio should be as close to 1 as possible. A low value for this ratio may mean that the application does not use the cache effectively.

- Estimated Latency Impact: 194.584

Estimated Latency Impact value is high, which likely indicates that the majority of L1 data cache misses are not being serviced by the L2 cache. Software prefetching is one strategy for improving this on the Intel Xeon Phi processor. Data reorganization or traditional techniques to increase data locality (such as cache blocking, using streaming stores, and data alignment) are other...

Hardware Event Counts:

- L2.DAT.A.READ.MISS.CACHE.Fill: 197,000,000
- L2.DAT.A.WRITE.MISS.CACHE.Fill: 3,941,500,000
- L2.DAT.A.READ.MISS.MEM.Fill: 9,348,500,000
- L2.DAT.A.WRITE.MISS.MEM.Fill: 7,549,000,000

Logical CPU Balance

This histogram represents a breakdown of overall CPU time by logical CPUs where it was spent. Explore to understand the overall balance of program execution over available logical CPUs.
<table>
<thead>
<tr>
<th>Function / Call Stack</th>
<th>CPU Time</th>
<th>Clockticks</th>
<th>Instructions Retired</th>
<th>CPI Rate</th>
<th>L1 Misses</th>
<th>L1 Hit Ratio</th>
<th>Estimated Latency Impact</th>
</tr>
</thead>
<tbody>
<tr>
<td>rotation3_fomp@parallel@25</td>
<td>895.619s</td>
<td>942,191,413,285</td>
<td>117,600,000,000</td>
<td>8.012</td>
<td>2,320,000,000</td>
<td>0.933</td>
<td>345,393</td>
</tr>
<tr>
<td>divergence_fomp@parallel@22</td>
<td>561.607s</td>
<td>590,810,866,215</td>
<td>147,300,000,000</td>
<td>4.011</td>
<td>1,880,000,000</td>
<td>0.932</td>
<td>234,352</td>
</tr>
<tr>
<td>mxn_fomp@parallel@04</td>
<td>545.410s</td>
<td>573,779,860,055</td>
<td>31,250,000,000</td>
<td>18.361</td>
<td>4,425,000,000</td>
<td>0.298</td>
<td>120,920</td>
</tr>
<tr>
<td>divergence4_fomp@parallel@21</td>
<td>508.451s</td>
<td>534,899,682,335</td>
<td>50,000,000,000</td>
<td>10.698</td>
<td>3,925,000,000</td>
<td>0.728</td>
<td>120,600</td>
</tr>
<tr>
<td>func@0x4f29</td>
<td>431.579s</td>
<td>454,020,681,030</td>
<td>86,000,000,000</td>
<td>5.228</td>
<td>35,000,000</td>
<td>0.999</td>
<td>10,156,162</td>
</tr>
<tr>
<td>dec_density_comp@parallel@24</td>
<td>370.514s</td>
<td>389,780,584,670</td>
<td>92,600,000,000</td>
<td>4.209</td>
<td>355,750,000</td>
<td>0.980</td>
<td>919,446</td>
</tr>
<tr>
<td>rotation1_routine@omp@parallel@24</td>
<td>282.481s</td>
<td>297,170,445,755</td>
<td>42,000,000,000</td>
<td>6.968</td>
<td>2,267,250,000</td>
<td>0.764</td>
<td>111,918</td>
</tr>
<tr>
<td>laplacian5_fomp@parallel@20</td>
<td>258.299s</td>
<td>271,730,407,595</td>
<td>38,700,000,000</td>
<td>7.021</td>
<td>2,263,750,000</td>
<td>0.763</td>
<td>99,708</td>
</tr>
<tr>
<td>a_field1_fomp@parallel@28</td>
<td>154.810s</td>
<td>162,860,244,290</td>
<td>18,250,000,000</td>
<td>8.924</td>
<td>1,445,000,000</td>
<td>0.219</td>
<td>99,004</td>
</tr>
<tr>
<td>varinux</td>
<td>130.751s</td>
<td>137,550,206,325</td>
<td>11,950,000,000</td>
<td>11.510</td>
<td>185,000,000</td>
<td>0.966</td>
<td>551,866</td>
</tr>
<tr>
<td>periodic_field_mlt8_fomp@parallel@28</td>
<td>129.734s</td>
<td>136,480,204,720</td>
<td>2,600,000,000</td>
<td>52.492</td>
<td>600,500,000</td>
<td>0.469</td>
<td>222,084</td>
</tr>
<tr>
<td>periodic_field_mlt7_fomp@parallel@28</td>
<td>114.088s</td>
<td>120,020,180,030</td>
<td>2,650,000,000</td>
<td>45.291</td>
<td>545,250,000</td>
<td>0.559</td>
<td>214,835</td>
</tr>
<tr>
<td>mnd_boundary_fomp@parallel@57</td>
<td>100.532s</td>
<td>105,760,158,540</td>
<td>3,400,000,000</td>
<td>31.106</td>
<td>305,000,000</td>
<td>0.647</td>
<td>339,566</td>
</tr>
<tr>
<td>gradient3_fomp@parallel@23</td>
<td>84.154s</td>
<td>88,510,132,795</td>
<td>10,500,000,000</td>
<td>8.431</td>
<td>1,031,250,000</td>
<td>0.689</td>
<td>74,205</td>
</tr>
<tr>
<td>func@0x4c950</td>
<td>80.418s</td>
<td>84,600,126,900</td>
<td>16,050,000,000</td>
<td>5.271</td>
<td>5,000,000</td>
<td>0.999</td>
<td>13,169,025</td>
</tr>
<tr>
<td>mnd_boundary_fomp@parallel@13</td>
<td>36.017s</td>
<td>37,899,056,835</td>
<td>650,000,000</td>
<td>58.292</td>
<td>145,000,000</td>
<td>0.256</td>
<td>257,897</td>
</tr>
<tr>
<td>__imp.yield</td>
<td>20.036s</td>
<td>21,080,031,205</td>
<td>4,050,000,000</td>
<td>5.205</td>
<td>1,000</td>
<td>1.000</td>
<td>0.000</td>
</tr>
<tr>
<td>func@0x4c160</td>
<td>19.866s</td>
<td>20,710,031,050</td>
<td>3,250,000,000</td>
<td>6.372</td>
<td>35,000,000</td>
<td>0.969</td>
<td>499,287</td>
</tr>
<tr>
<td>__imp.barrier</td>
<td>15.960s</td>
<td>16,790,025,185</td>
<td>900,000,000</td>
<td>18.656</td>
<td>55,000,000</td>
<td>0.843</td>
<td>289,819</td>
</tr>
<tr>
<td>__imp_forstatic_init.A</td>
<td>6.046s</td>
<td>6,360,009,540</td>
<td>450,000,000</td>
<td>14.133</td>
<td>5,000,000</td>
<td>0.964</td>
<td>1,124,002</td>
</tr>
</tbody>
</table>
Benchmark MIPS

Hybrid test

- Test on 16 MPI, 1 Node, 8 MPI with 2 OpenMP/MPI on host and 4 MPI on Intel Xeon Phi Coprocessor with 20 OpenMP/MPI
- Good load balance
- Some MPI.Wait (normal)
- Difference at the beginning and end (normal)
Benchmark MIPS

Loop

- No time locality, only potential space locality
- Boundary 1r will increase with input test case size
- Tests/remarks:
  - Try to re write with intrinsic (1 load againsts 2 loads) no gain
  - Try to split loop : no gain
  - Very close to “Stream” trouble

```plaintext
<table>
<thead>
<tr>
<th>No.</th>
<th>Line</th>
<th>Source</th>
<th>CPU Time</th>
<th>Cycles</th>
<th>Instructions Retired</th>
<th>CPI Rate</th>
<th>Cache Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>22</td>
<td>$smp parallel</td>
<td>0.1990 s</td>
<td>420,000,930</td>
<td>0</td>
<td>0</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>23</td>
<td>$smp do private(i,j,k,10,11,12,13)</td>
<td>0.0670 s</td>
<td>70,000,105</td>
<td>0</td>
<td>0</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>24</td>
<td>do k = 3, lph-2</td>
<td>0.0100 s</td>
<td>10,000,015</td>
<td>0</td>
<td>0</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>25</td>
<td>do j = 1, l1x</td>
<td>0.0100 s</td>
<td>10,000,015</td>
<td>0</td>
<td>0</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>26</td>
<td>do i = 3, l1r-2</td>
<td>0.1240 s</td>
<td>130,000,195</td>
<td>0</td>
<td>0</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>27</td>
<td>l1r = 1 + 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>l2r = i + 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>al = i + 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>z0 = i + 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>dvi(i,j,k) = ccDmp(i,j,k)gpr[i,j,k] &amp;</td>
<td>1.7440 s</td>
<td>18,540,275,310</td>
<td>45,600,000,000</td>
<td>4.025</td>
<td>230,000,000</td>
<td>0.960</td>
</tr>
<tr>
<td>33</td>
<td>&lt;t(I,j,k) = ccDmp(i,j,k)gpr[i,j,k] &amp;</td>
<td>12.522 s</td>
<td>13,180,019,770</td>
<td>4,450,000,000</td>
<td>2.962</td>
<td>130,000,000</td>
<td>0.910</td>
</tr>
<tr>
<td>34</td>
<td>&lt;cDmp(i,j,k)gpr[i,j,k] &amp;</td>
<td>18.088 s</td>
<td>16,010,028,530</td>
<td>5,300,000,000</td>
<td>3.550</td>
<td>150,000,000</td>
<td>0.967</td>
</tr>
<tr>
<td>35</td>
<td>&lt;cDmp(i,j,k)gpr[i,j,k] &amp;</td>
<td>7.814 s</td>
<td>8,220,012,330</td>
<td>1,000,000,000</td>
<td>2.740</td>
<td>5,000,000,000</td>
<td>0.983</td>
</tr>
<tr>
<td>36</td>
<td>dy(i,j,k) = dvi(i,j,k) &amp;</td>
<td>8.850 s</td>
<td>9,310,019,385</td>
<td>3,250,000,000</td>
<td>2.805</td>
<td>0,1,000,000</td>
<td>0.980</td>
</tr>
<tr>
<td>37</td>
<td>end do</td>
<td>4.795 s</td>
<td>4,560,006,840</td>
<td>1,450,000,000</td>
<td>3.145</td>
<td>35,000,000</td>
<td>30,857</td>
</tr>
<tr>
<td>38</td>
<td>end do</td>
<td>0.314 s</td>
<td>330,000,495</td>
<td>100,000,000</td>
<td>1.300</td>
<td>0.1,000</td>
<td>0.000</td>
</tr>
<tr>
<td>39</td>
<td>end do</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>$smp end do</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>41</td>
<td>$smp do private(j,k)</td>
<td>0.0080 s</td>
<td>70,000,105</td>
<td>0</td>
<td>0</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>42</td>
<td>do k = 3, lph-2</td>
<td>0.0090 s</td>
<td>100,000,150</td>
<td>0</td>
<td>0</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>43</td>
<td>do j = 1, l1r</td>
<td>0.0290 s</td>
<td>30,000,045</td>
<td>0</td>
<td>0</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
</tbody>
</table>
```
Benchmark MIPS

On first result

- Intel tools work fine on Intel Xeon Phi Coprocessor to find troubles (vtune/ltač)
- Good behavior for MIPS on Intel Xeon Phi Coprocessor
- Good load balance
- Bottleneck :
  - usage of static allocaton
  - behavior close to stream benchmark

What's Next ?

- Work on memory allocation. Goal : try to use less than 2.2G/MPI for real test case
- Work on algorithm to decrease pressure on memory bandwidth
Future

- Introduction
- Helios and Intel Xeon Phi
- Benchmark MIPS
- Future
- Bibliography
Genc: "Cellule de veille Technologique GENCI"

- 5 experts to support Workshop activity: 2 from Atos, 2 from Intel, 1 from Genci
- Be ready for Exascale next cluster
- Follow new HPC architectures
- Provide small cluster POC to test new architectures
- New team from CEPP at Montpellier
- Application at least 1 developper by application:
  - GYSELA, SMILEI and two codes from IN2P3
  - DYNAMICO and MesoNH
  - CFD: YALES2 et TRIO-CFD
  - PATMOS
  - SPECFEM3D
  - Astrophysique: RAMSES-GPU and hydro
  - Metawalls
  - BigDFT et QMC=Chem
  - Maths: Sparse multifrontal QR et NT2
Future

And Now?
Thanks for your attention
Bibliography

James Reinders James L Jeffers :
*Intel Xeon Phi Coprocessor High-Performance Programming.*