Energy issues of GPU computing clusters

Stéphane Vialle
SUPELEC– UMI GT-CNRS 2958 & AlGorille INRIA Project Team

EJC 19-20/11/2012
Lyon, France
What means « using a GPU cluster » ?

Programming a cluster of « CPU+GPU » nodes
- Implementing message passing + multithreading + vectorization
- Long and difficult code development and maintenance

→ How many software engineers can do it ?

Computing nodes requiring more electrical power (Watt)
- CPU + (powerful) GPU dissipate more electrical power than CPU
- Can lead to improve the electrical network and subscription

→ Can generate some extra-costs !

But we expect :
- To run faster
  and / or
- To save energy (Watt.Hours)
1 - First experiment:

« hapilly parallel » application

- Asian option pricer (independant Monte Carlo simulations)
- Rigorous parallel random number generation

Lokman Abas-Turki
Stephane Vialle
Bernard Lapeyre

2008
1 - Happily parallel application

Application:
« Asian option pricer »:
Independent Monte Carlo trajectory computations

Coarse grain parallelism on the cluster:
• Distribution of data on each computing node
• Local and independent computations on each node
• Collect of partial results and small final computation

Fine grain parallelism on each node:
• Local data transfer on GPU memory
• Local parallel computation on the GPU
• Local result transfer from the GPU to the CPU memory

→ Coarse and fine grain parallel codes can be developed separately (nice!)
1 - Happily parallel application

Long work to design rigorous parallel random number generation on the GPUs

1 - Input data reading on P₀

2 - Input data broadcast from P₀

Coarse grain

3 - Parallel and independent RNG initialization

Coarse grain and fine grain

4 - Parallel and independent Monte-Carlo computations

Coarse grain and fine grain

5 - Parallel and independent partial results computation

Coarse grain

6 - Partial results reduction on P₀ and final price computation

7 - Print results and perfs
1 - Happily parallel application

Comparison to a **multi-core** CPU cluster (using all CPU cores):

16 GPU nodes run **2.83** times faster than 256 CPU nodes

---

256 INTEL dual-core nodes
1 CISCO 256-ports switch, Gigabit-eth

16 INTEL dual-core nodes
1 GPU (GeForce 8800 GT) / node
1 DELL 24-ports switch, Gigabit-eth
1 - Happily parallel application

Comparison to a **multi-core** CPU cluster (using all CPU cores):

- **16 GPU nodes consume 28.3 times less than 256 CPU nodes**

GPU cluster is 2.83x28.3 = **80.1 times more efficient**
2 – « Real parallel » code experiments:
Parallel codes including communications

• 2D relaxation (frontier exchange)
• 3D transport PDE solver

Sylvain Contassot-Vivier
Stephane Vialle
Thomas Jost
Wilfried Kirschenmann

2009
2 – Parallel codes including comms.

These algorithms remain synchronous and deterministic
But coarse and fine grained parallel codes have to be jointly designed

→ Developments become more complex

<table>
<thead>
<tr>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Internode CPU communications</td>
</tr>
<tr>
<td>Local CPU → GPU data transfers</td>
</tr>
<tr>
<td>Local GPU computations</td>
</tr>
<tr>
<td>Local GPU → CPU partial result transfers</td>
</tr>
<tr>
<td>Local CPU computations (not adapted to GPU processing)</td>
</tr>
<tr>
<td>Internode CPU communications</td>
</tr>
<tr>
<td>Local CPU → GPU data transfers</td>
</tr>
<tr>
<td>...</td>
</tr>
</tbody>
</table>

More synchronization issues between CPU and GPU tasks

More complex buffer and indexes management:

One data has a global index, node cpu-buffer index, node gpu-buffer index, a fast-shared-memory index in a sub-part of the GPU...
Developments become (really) more complex

- Less software engineers can develop and maintain parallel code including communications on a GPU cluster

GPU accelerate only some parts of the code

GPU requires more data transfer overheads (CPU $\leftrightarrow$ GPU)

- Is it possible to speedup on a GPU cluster?
- Is it possible to speedup enough to save energy?
2 – Parallel codes including comms.
2 – Parallel codes including comms.

Rmk: Which comparison? Which reference?

You have a GPU cluster
→ you have a CPU cluster!

You succeed to program a GPU cluster
→ you can program a CPU cluster!

Compare a GPU cluster to a CPU cluster (not to one CPU core...) when possible
Comparison will be really different
2 – Parallel codes including comms.

Temporal gain (speedup) & Energy Gain of GPU cluster vs CPU cluster:

Up to 16 nodes this GPU cluster is more interesting than our CPU cluster, **but its interest decreases**...
## 2 – Parallel codes including comms.

<table>
<thead>
<tr>
<th></th>
<th>CPU cluster</th>
<th>GPU cluster</th>
</tr>
</thead>
<tbody>
<tr>
<td>Computations</td>
<td>T-calc-CPU</td>
<td>If algorithm is adapted to GPU architecture: T-comput-GPU (\ll) T-compu-CPU else: do not use GPUs!</td>
</tr>
<tr>
<td>Communications</td>
<td>T-comm-CPU =</td>
<td>T-comm-GPU = T-transfert-GPUtoCPU + T-comm-MPI + T-transfert-CPUtoGPU</td>
</tr>
<tr>
<td></td>
<td>T-comm-MPI</td>
<td>T-comm-GPU (\geq) T-comm-CPU</td>
</tr>
<tr>
<td>Total time</td>
<td>T-CPUcluster</td>
<td>T-GPUcluster (&lt;\ ? &gt;) T-CPUcluster</td>
</tr>
</tbody>
</table>

→ For a given pb on a GPU cluster: T-comm becomes strongly dominant and GPU cluster interest decreases
3 – Asynchronous parallel code experiments:
(asynchronous algorithm & asynchronous implementation)

• 3D transport PDE solver

Sylvain Contassot-Vivier
Stephane Vialle

2009-2010
3 - Async. parallel codes on GPU cluster

Asynchronous algo. provide implicit overlapping of communications and computations, and communications are important into GPU clusters.

→ Asynchronous code **should** improve execution on GPU clusters specially on heterogeneous GPU cluster

**BUT:**

• Only some iterative algorithms can be turned into asynchronous algorithms

• The convergence detection of the algorithm is more complex and requires more communications (than with synchronous algo)

• Some extra iterations are required to achieve the same accuracy.
3 - Async. parallel codes on GPU cluster

Rmk: asynchronous code on GPU cluster has awful complexity

Available synchronous PDE solver on GPU cluster (previous work)

- 2 senior researchers in parallel computing
- 1 year work

The most complex debug we have achieved!

... how to « validate » the code?

Operational asynchronous PDE solver on GPU cluster
3 - Async. parallel codes on GPU cluster

Execution time using 2 GPU clusters of Supelec:

• 17 nodes Xeon dual-core + GT8800
• 16 nodes Nehalem quad-core + GT285

• 2 interconnected Gibagit switches
3 - Async. parallel codes on GPU cluster

**Speedup vs 1 GPU:**
- asynchronous version achieves more regular speedup
- asynchronous version achieves better speedup on high nb of nodes
3 - Async. parallel codes on GPU cluster

Energy consumption:
- sync. and async. energy consumption curves are (just) different
3 - Async. parallel codes on GPU cluster

Energy overhead factor vs 1 GPU (overhead to minimize):
- overhead curves are (just) « differents »
  → no more global attractive solution!

GPU cluster & synchronous vs 1 GPU

GPU cluster & asynchronous vs 1 GPU
Async vs sync speedup and async vs sync energy gain

- Can be used to choose the version to run
- But region frontiers are complex: need a fine model to predict

---

**Speedup**

<table>
<thead>
<tr>
<th>1,1-1,2</th>
<th>1,0-1,1</th>
<th>0,9-1,0</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td>12</td>
<td>10</td>
</tr>
</tbody>
</table>

**Energy gain**

<table>
<thead>
<tr>
<th>0,8-0,9</th>
<th>0,9-1,0</th>
<th>1,0-1,1</th>
<th>1,1-1,2</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td>12</td>
<td>10</td>
<td>8</td>
</tr>
</tbody>
</table>

Async. better
Overview of asynchronous code experiments:

Can lead to better performances on heterogeneous GPU clusters

But:
  • Very hard to develop
  • Difficult to identify when it is better than a synchronous code

→ Not the « magical solution » to improve performances on GPU clusters

We are investigating new asynchronous approaches ...
4 – Synchronous parallel application including communications and designed for GPU clusters

• American option pricer

Lokman Abbas-Turki
Stephane Vialle

Supélec

UNIVERSITÉ PARIS EST

2010-2011-2012
American option pricing:

- Non linear PDE problem
- Many solutions does not require too much computations  
  BUT:
  - Have limited accuracy
  - Are not parallel (bad scaling when parallelized)

Our solution:

- New mathematic approach based on Maillavin calculus
- Use Monte Carlo computation: we have efficient solution on GPU
- Design a BSP-like parallel algorithm: separated big computing steps and communication steps

To get:
- high quality results
- GPU efficient code
- scalable parallel code on GPU cluster
4 – Sync. code designed for GPUs

Comparison on one node : CPU vs GPU
• 1 INTEL 4-core hyperthreaded (« Nehalem »)
• 1 NVIDIA GTX480 (« Fermi »)

The parallelization seems well adapted to GPU
(it has been designed for this architecture)
4 – Sync. code designed for GPUs

Good scalability of parallel code on GPU cluster

Energy consumption remains constant when using more nodes

And results have high quality!

Missing experiments on multicore CPU clusters... (long to measure...)

---

**Pricer AmerMal - 5Stk**

- T-256Ktt-5Stk
- T-128Ktt-5Stk
- T-64Ktt-5Stk

**Pricer AmerMal – 5Stk**

- 256Ktt-5Stk
- 128Ktt-5Stk
- 64Ktt-5Stk
4 – Sync. code designed for GPUs

After some years of experience in « option pricing on GPU clusters »

• Redesign of the mathematic approach
• Identification of a solution accurate and adapted to GPU clusters
• Many debug steps, many benchmarks-perf analysis-optimization
• Good results and performances!

• Still a bottleneck in the code limits the full scalability…
  … under improvement.

• Has required long development times
  1-2 years (part time)
  1 mathematician, with strong knowledge in GPGPU
  1 computer scientist in parallel computing (and GPGPU)
5 – Can GPU clusters decrease the energy consumption?
5 – GPU cluster energy consumption

In all our experiments: Te decreases ⇒ Energy decreases

« Te decreases stronger than P increases »

But sometimes Speedup and Energy Gain are low (< 5)!

Warning! Electrical Power increase can require some changes:

- Improve the electrical network!
- Increase of the electrical subscription!
- Improve the maximal electrical production ...
5 – GPU cluster energy consumption

Different use-cases when you add GPUs in a PC cluster:

- **Limited amount of computations to run** (unsaturated machines)
  - During computations: $P \uparrow$, $T_e \downarrow$, Flops/Watt $\uparrow$ ...... and $E \downarrow$
  - When machine is unused and switched on: $P \uparrow$ ...... and $E \uparrow$
  - An unused and switched on GPU cluster wastes a lot of energy (under improvement ?)

- **Add GPUs and reduce the nb of nodes:** total Flops unchanged
  - $P \downarrow$, $T_e \downarrow$, Flops/Watt $\uparrow$ ...... and $E \downarrow$
  - But applications not adapted to GPU will run slowly!
5 – GPU cluster energy consumption

Different use-cases when you add GPUs in a PC cluster:

• Add GPU in each node: increase the total Flops

  If unlimited amount of computations to run
  (saturated machines)

  $P \uparrow$, $Te \downarrow$, Flop/Watt $\uparrow$ …… but $E \uparrow$

  Each computation is faster and less consuming
  But more and more computations are run
Conclusion

GPU and GPU-clusters remain complex to program to achieve high performances:
  • Re-design mathematic solutions
  • Optimize code for GPU clusters
  • Compare to multicore CPU clusters

Add GPUs in a PC cluster increase the electrical power dissipation:
  • Poor usage of GPUs will waste (a lot of) energy

GPU is a « vector co-processor » with high impact:
  • Can speedup application and reduce the energy consumption and satisfy users
  • Can be not adapted to a code and can increase the energy consumption ... and make users angry!

→ Analyse the requirements and knowledge of users before to install (actual) GPUs
Energy issues of GPU computing clusters

Questions ?