Performance and Numerical Accuracy Evaluation of Heterogeneous Multicore Systems for Krylov Orthogonal Basis Computation

J. Dubois

CEA/DEN/DM2S/SERMA – Université De Lille – LIFL/MAP

C. Calvin

CEA/DEN/DM2S/SERMA

S. Petiton

Université de Lille - LIFL/MAP

Vecpar10 - Berkeley, California

22-25 June 2010

Outline

- Motivations
- IEEE norm and parallelism
- Accelerators IEEE handling
- Numerical method
- Experiments

Motivations

- Emerging computing hardware offer:
  - high peak performance, ~Tflops in SP, 100s of Gflops in DP
  - high on-device memory bandwidth, 100s of GB/s
  - high level of fine grain parallelism : 100s of cores
  - high Gflops / $ ratio
- They are in used in several supercomputers : US Roadrunner (Cell), China Nebulae(GPU), France Titane(GPU)
- They actually speed-up scientific computations : we could accelerate our neutronics solvers by a 5x to 20x speed-up.
- But they have slightly different IEEE compliance

What is the influence of this non full IEEE compliance on actual computations?

IEEE norm and parallelism

- IEEE norm defines :
  - The maximum level of expected error for basic floating operations
  - The way floating point numbers are represented
  - Over/underflow errors
  - …

- Because of finite precision, we do not have:
  - Associativity: (a + b) + c not necessarily equals a + (b + c)
    - This can impact parallelism results
  - Exact representation : 0.1 is actually represented by 0.0999…
    - This can impact hardware truncating numbers during operations

Accelerators IEEE handling

- Cell : fully IEEE compliant PowerPC core + 8 non fully IEEE compliant cores :
  - Only round-towards-zero rounding mode (vs round-to-even)
  - Different underflow/overflow handling
  - « The programmer should be aware that, in some cases, the computation results will not be identical to IEEE Standard 754 ones »
  - For **double precision**, the Cell processor is fully IEEE-compliant

- GPU : non fully IEEE compliant computing cores (1.0 to 1.3)
  - Single multiply-add instruction (FMAD) truncates the intermediate result of the multiplication (**real influence**);
  - Division is implemented in a non-standard-compliant way;
  - Square root is implemented via the reciprocal square root in a non-standard-compliant way;
  - For addition and multiplication, only round-to-nearest-even and round-towards-zero are suuported

Numerical method

- Arnoldi reduction based on Gram-Schmidt orthogonalization
  - Classical, modified and with reorthogonalization
- Commonly used in Krylov subspace methods :
  - Linear solvers
  - Eigensolvers

- Orthogonality may impact numerical results
- Method sensitive to numerical pertubations:
  - CGS vs MGS orthogonality
- What if the hardware is not IEEE?

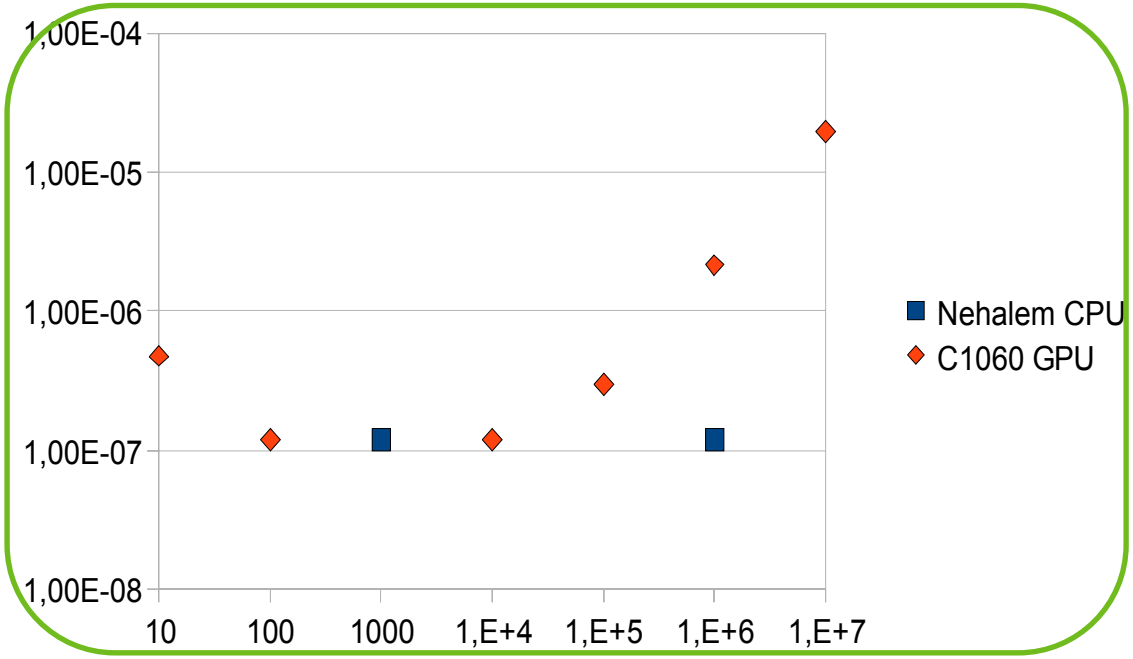
Experiments

- We test if different IEEE handling influences simple vector operations
- We used a simple algorithm based on BLAS operations

1. Take a vector v
2. $w = v + v$ (axpy kernel)
3. $n_2 = ||w||_2$ (norm2 kernel)
4. $w = w / n_2$ (scal kernel)
5. result = $||w||_2$ (norm2 kernel)

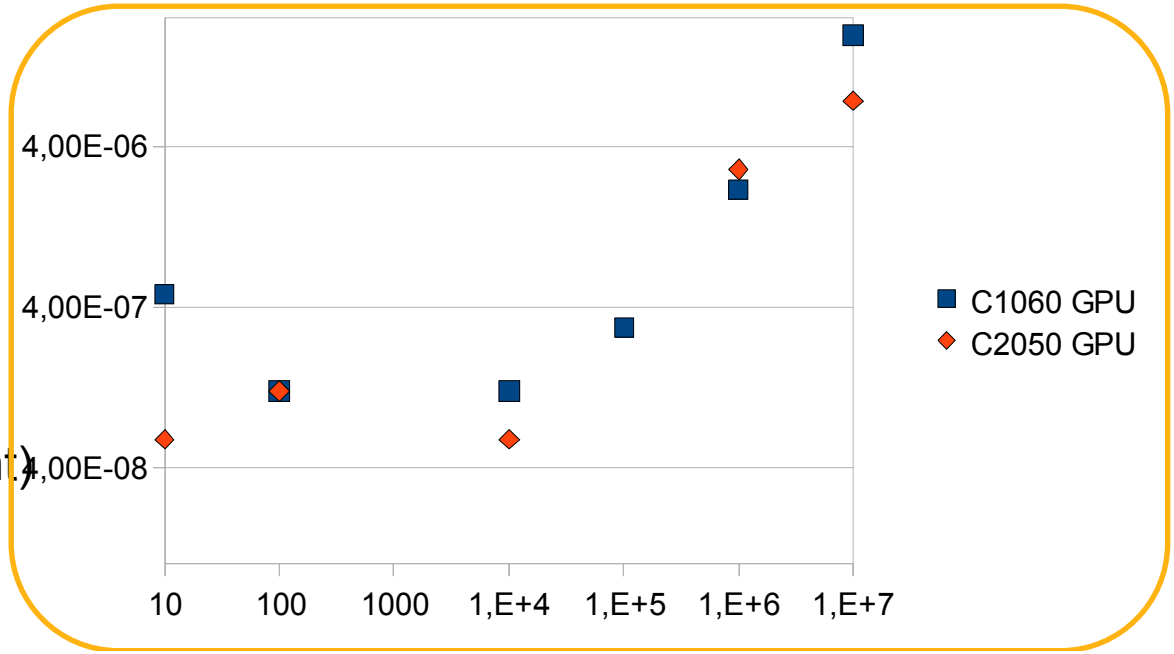
Actual result should be **equal to 1**.

Experiments : vector operations



CPU/GPU 32 bits results :
- Y axis : | 1 - actual result |
- X axis : vector size
- no dot=> « exact solution »

GPU/GPU 32 bits results :
- same data
- C1060 vs C2050
  (less vs more IEEE compliant)



Experiments : vector operations

Double precision results

Vector size	C1060	C2050
1.0e+6	9,99E-15	9,99E-15
1.0e+7	4,00E-14	1,59E-14

Temporary conclusions

1. Results actually differ even for simple vector operations
2. The more IEEE compliant, the more accurate the results

Arnoldi Reduction

Arnoldi Reduction CGS algorithm

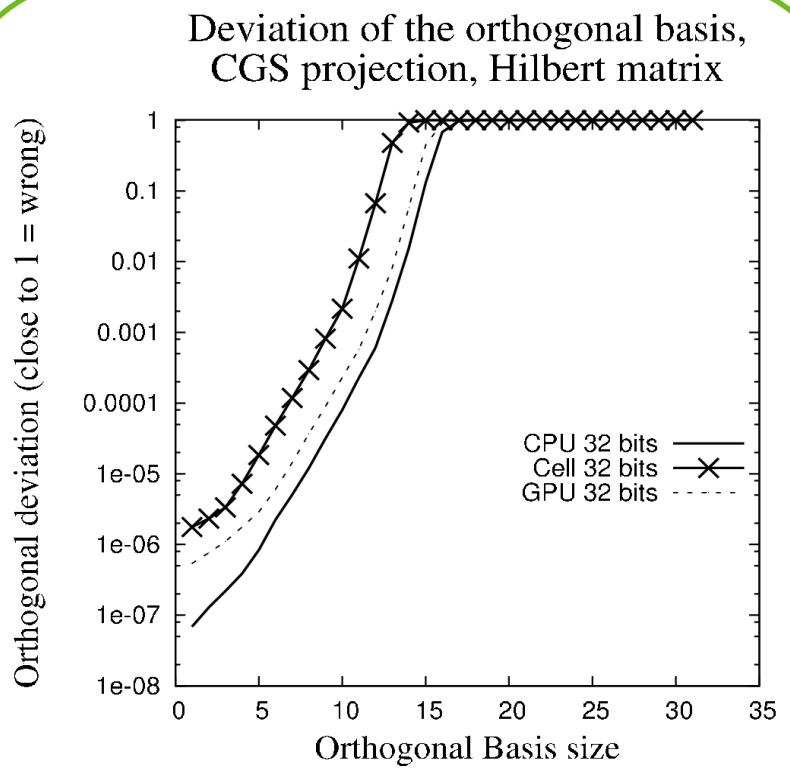
```
for i from 1 to p                                          CGS
do
   res = matrix_vector_product(A,V_{i-1})                  ← mv
   for k from 1 to i
      H_{k,i} = dot_product(res , V_k)                     ← dot
   for k from 1 to i
      res = res + H_{k,i} * V_i                            ← axpy
   H_{i+1,i} = norm2(res)                                  ←norm2
   if H_{i+1,i} = 0 then stop
   V_i = res / H_{i+1,i}                                   ← scal
done
```

Arnoldi Reduction orthogonal error computation

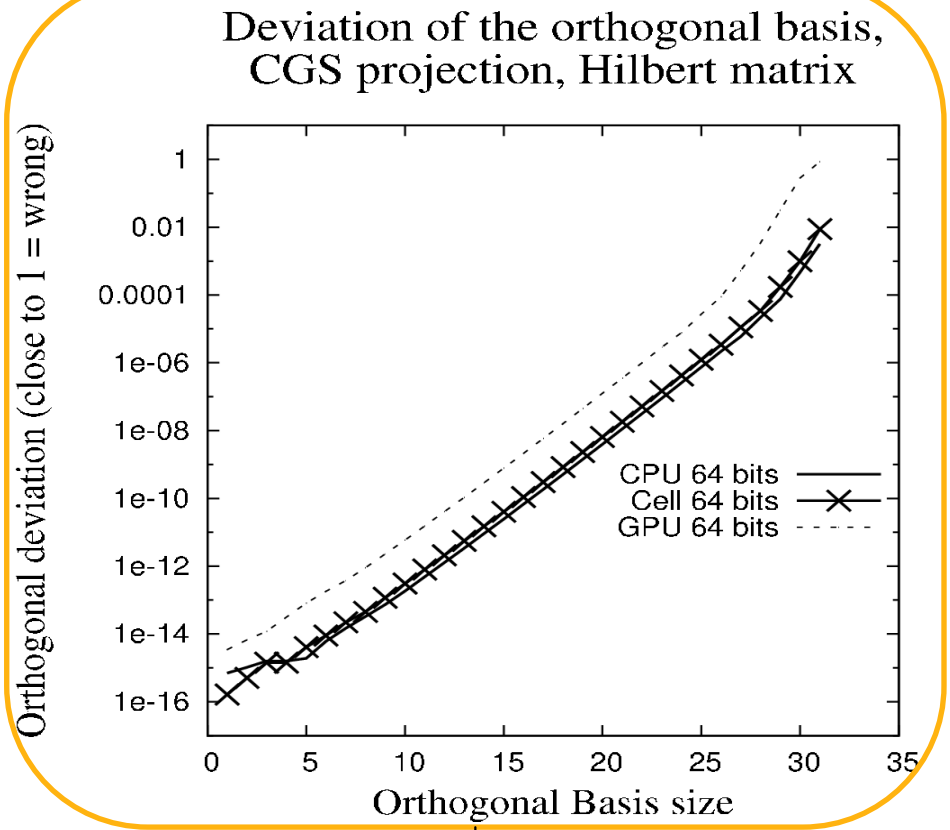
Error = max | V_i, V_j |, with i ≠ j

Experiments : Arnoldi reduction

•32 bits dense CGS orthogonalization
• IEEE limit ~1e-08



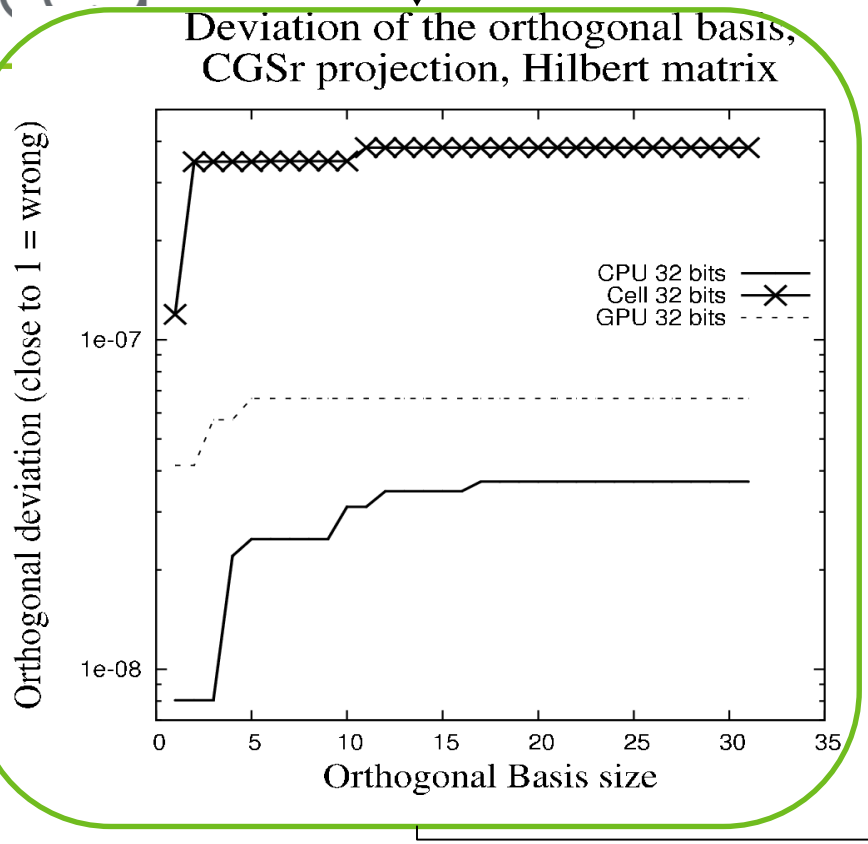
•64 bits dense CGS orthogonalization :
•IEEE limit ~1e-16



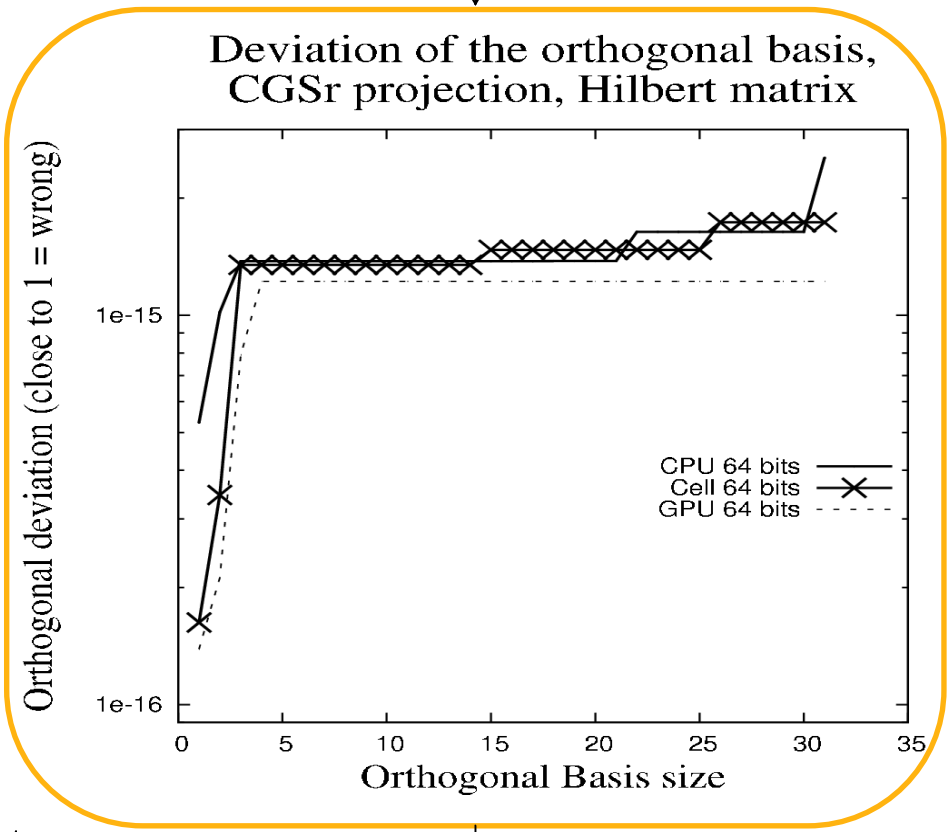
•32 bits accuracy: Cell < GPU < multicore
•64 bits accuracy: GPU < Cell <= multicore

Experiments : Arnoldi reduction

•32 bits dense CGS reorthogonalization
•IEEE limit ~1e-08



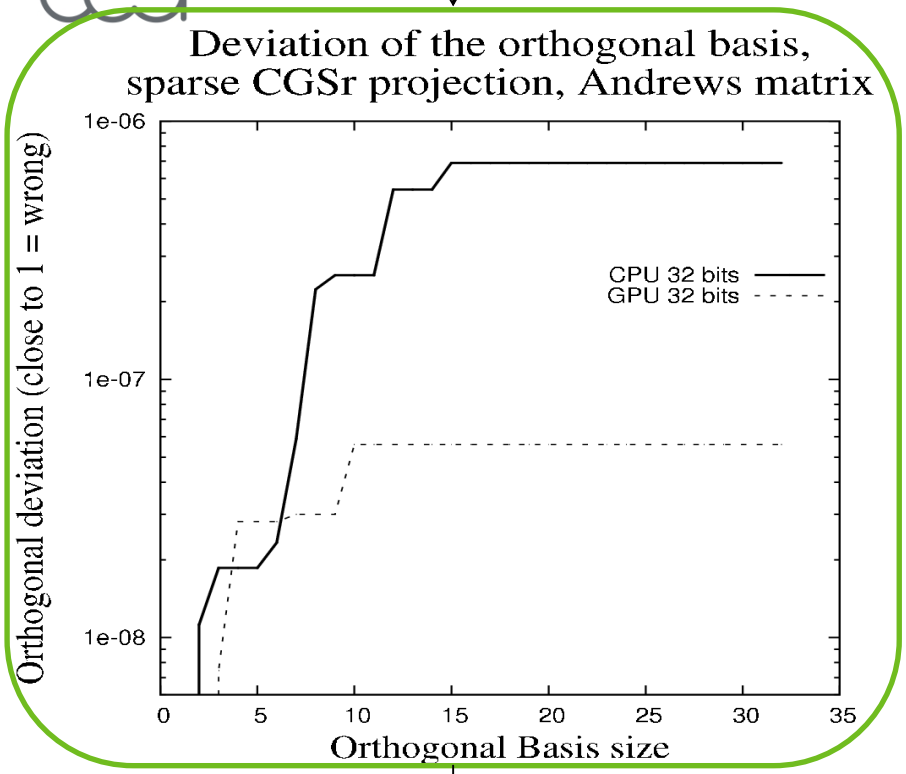
•64 bits dense CGS reorthogonalization
•IEEE limit ~1e-16

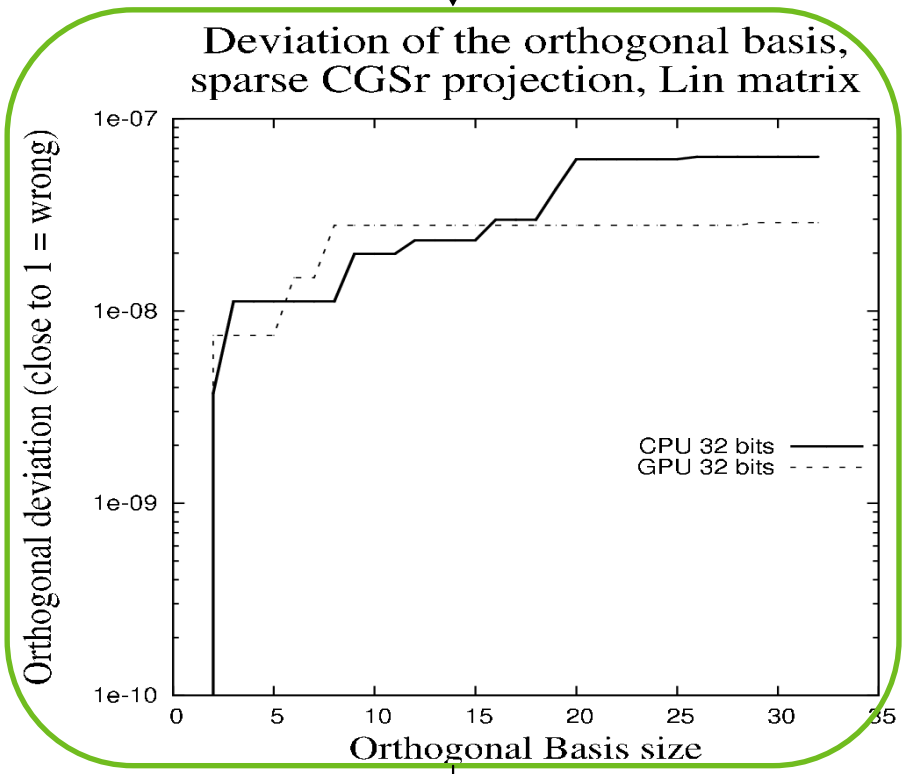


•32 bits accuracy: Cell < GPU < multicore
•64 bits accuracy: Cell ≤ multicore ≤ GPU

Experiments : Arnoldi reduction

- 32 bits sparse CGSr computations
- IEEE accuracy ~1.0^{e}-8

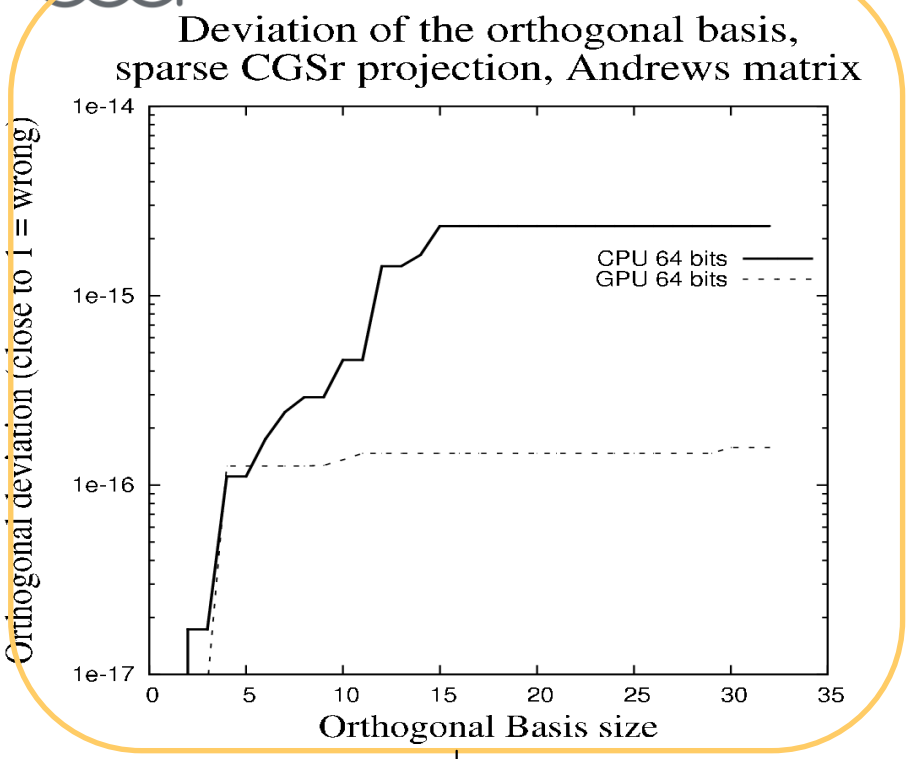


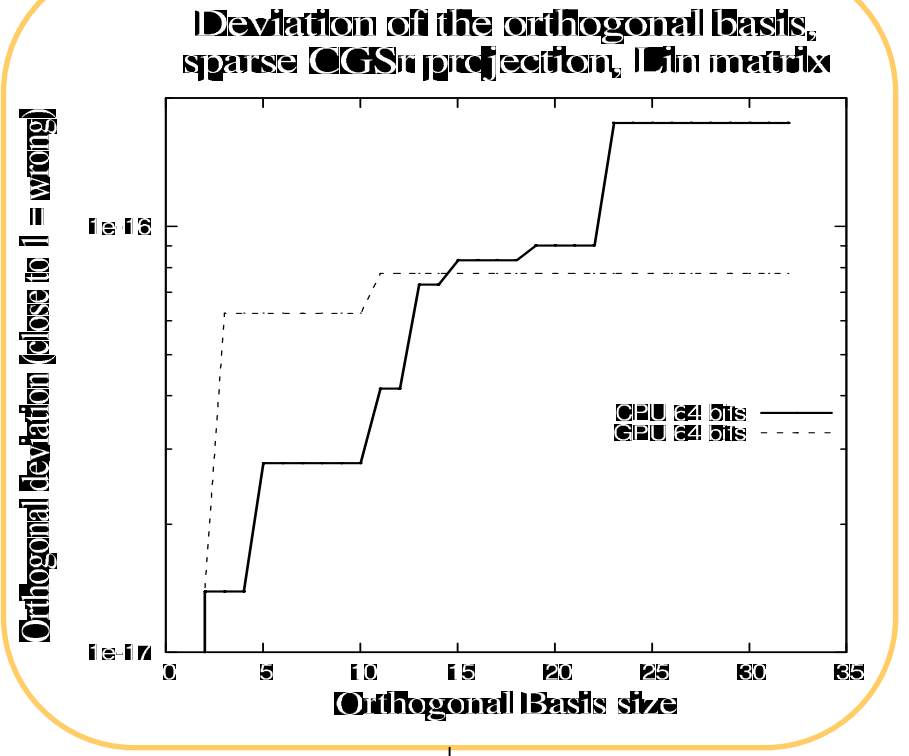


- 32 bits accuracy: multicore ?≤? GPU

Experiments : Arnoldi reduction

- 64 bits sparse CGSr computations
- IEEE accuracy ~1.0^{e}-16





- 64 bits accuracy: multicore ?≤? GPU

Performance

- Performance is bandwidth bound:

Architecure	bandwidth	Exp. Perf.
Nehalem	25.6 GB/s	6.4 Gflops
Cell	25.6 GB/s	6.4 Gflops
GPU	70 GB/s	17.5 Gflops

- Performance achieved:

Arch.	Dense	Speed-up	Sparse	Speed-up
Nehalem	3	/	1.3	/
Cell	6	2x	N/A	N/A
GPU	17 (22)	5.7x (7.3x)	10	7.7x

Summary

- For a simple algorithm involving BLAS1 operations:
  - IEEE compliance influences actual result
  - Stronger influence in SP than DP
  - The more IEEE, the more accurate (old vs new Tesla)

- For a more complex algorithm, Arnoldi Reduction:
  - Dense results show:
    - With a method sensitive to round-off error, IEEE very important
    - With a less sensitive one, hierarchy is differs:
      - SP accuracy => Cell < GPU < multicore
      - DP accuracy => GPU ?<? Cell < multicore ?<? GPU
  - Sparse results show:
    - GPU seems to give better accuracy in both SP/DP
    - Not really in fact: due to cancellation of many elements

Conclusion & Future Works

- GPU and Cell are good accelerators : 2x to 7x faster (4-21x vs unicore)
- IEEE compliance differs, influencing results in various way
- Next generation of GPUs is more IEEE compliant
- It also sports caches, ECC, and lots of CPU-like features

- Future works include
  - Actual influence of the IEEE compliance on a full Arnoldi Process
  - Extending the parallelization on a cluster of GPUs
  - Adding one level of parallelism: Multiple Restarted Arnoldi Process
  - Experiment on Fermi Tesla

Thanks for your attention!

And time for questions.