Valéry Weber, A. Cristiano I. Malossi, Ivano Tavernelli, Teodoro Laino, Costas Bekas, Manish Modani, Nina Wilner, Tom Heller, and Alessandro Curioni

18th October 2016

First Experiences with ab initio Molecular Dynamics on OpenPOWER: the case of CPMD

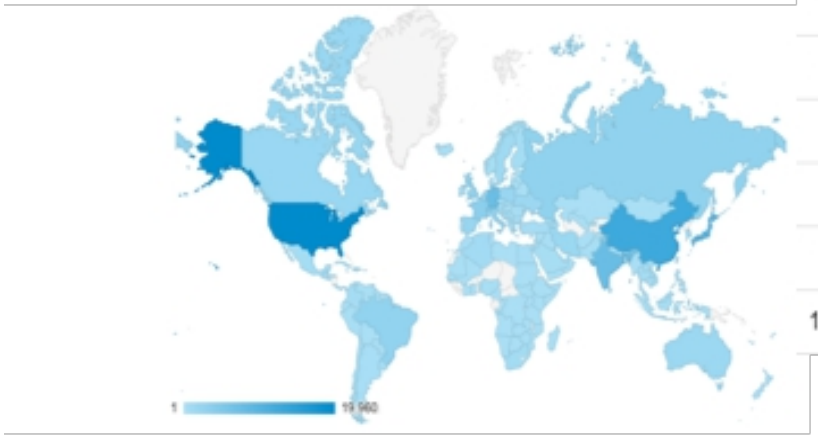
Teodoro Laino, IBM Research - Zurich

© 2016 IBM Corporation

CPMD

more than 2500 (on average) visitors per month (~ 100 visitor/day - on average the site is visited 6 hours per day): excluding bots and crawlers

© 2016 IBM Corporation



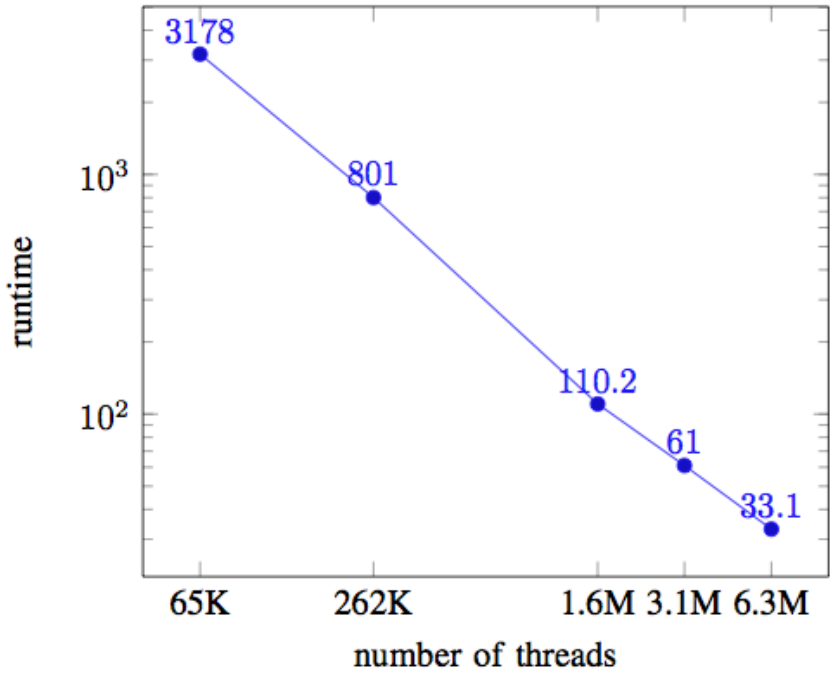
Country	Acquisition		
	Sessions ↓	% New Sessions	New Users
	112,095 % of Total: 100.00% (112,095)	50.78% Avg for View: 50.72% (0.12%)	56,921 % of Total: 100.12% (56,851)
1. United States	**19,960** (17.81%)	63.93%	12,760 (22.42%)
2. China	**12,926** (11.53%)	66.70%	8,622 (15.15%)
3. Japan	**10,164** (9.07%)	23.94%	2,433 (4.27%)
4. India	**7,706** (6.87%)	50.79%	3,914 (6.88%)
5. Germany	**7,667** (6.84%)	42.77%	3,279 (5.76%)
6. Switzerland	**5,042** (4.50%)	25.66%	1,294 (2.27%)
7. France	**3,584** (3.20%)	50.00%	1,792 (3.15%)
8. Italy	**3,055** (2.73%)	49.49%	1,512 (2.66%)
9. Russia	**3,038** (2.71%)	35.91%	1,091 (1.92%)
10. United Kingdom	**2,980** (2.66%)	66.07%	1,969 (3.46%)

since March 2013.

© 2016 IBM Corporation

Success in Petascale computing: BG/Q Results

Implementing Exact-Exchange in CPMD
>99% Parallel Efficiency to over 6.2M threads
Studying Li-Air Batteries, 1736 atoms, 70Ry cutoff



© 2016 IBM Corporation

Motivations

- More and more architectures are now available with GPUs.

- CPU/GPU hybrid systems are needed to reach Exascale Ambitions.

Thus there is a need to port computational science codes to hybrid systems. This requires algorithm rethinking and code reengineering in order to fully exploit next generation of heterogeneous architectures.

In this presentation we focus on the electronic structure code CPMD.

© 2016 IBM Corporation

Outlook and Goal

- Introduction: Kohn–Sham equations.
- Parallelization: Distributed Memory and 3D FFT.
- GPU Porting: Electronic density, applying the potential to the wavefunctions and orthogonalization.
- Benchmark
- Summary

Goal: Porting 3 expensive kernels to GPU.

© 2016 IBM Corporation

Introduction: Kohn–Sham equations

$$\left[-\frac{1}{2}\nabla_i^2 + V_{\text{eff}}[\rho]\right]\phi_i(\mathbf{r}) = \epsilon_i\phi_i(\mathbf{r}),$$

$$\rho(\mathbf{r}) = \sum_i^N |\phi_i(\mathbf{r})|^2$$

$$\int \phi_i(\mathbf{r})\phi_j(\mathbf{r})d^3r = \delta_{ij}$$

Observation:

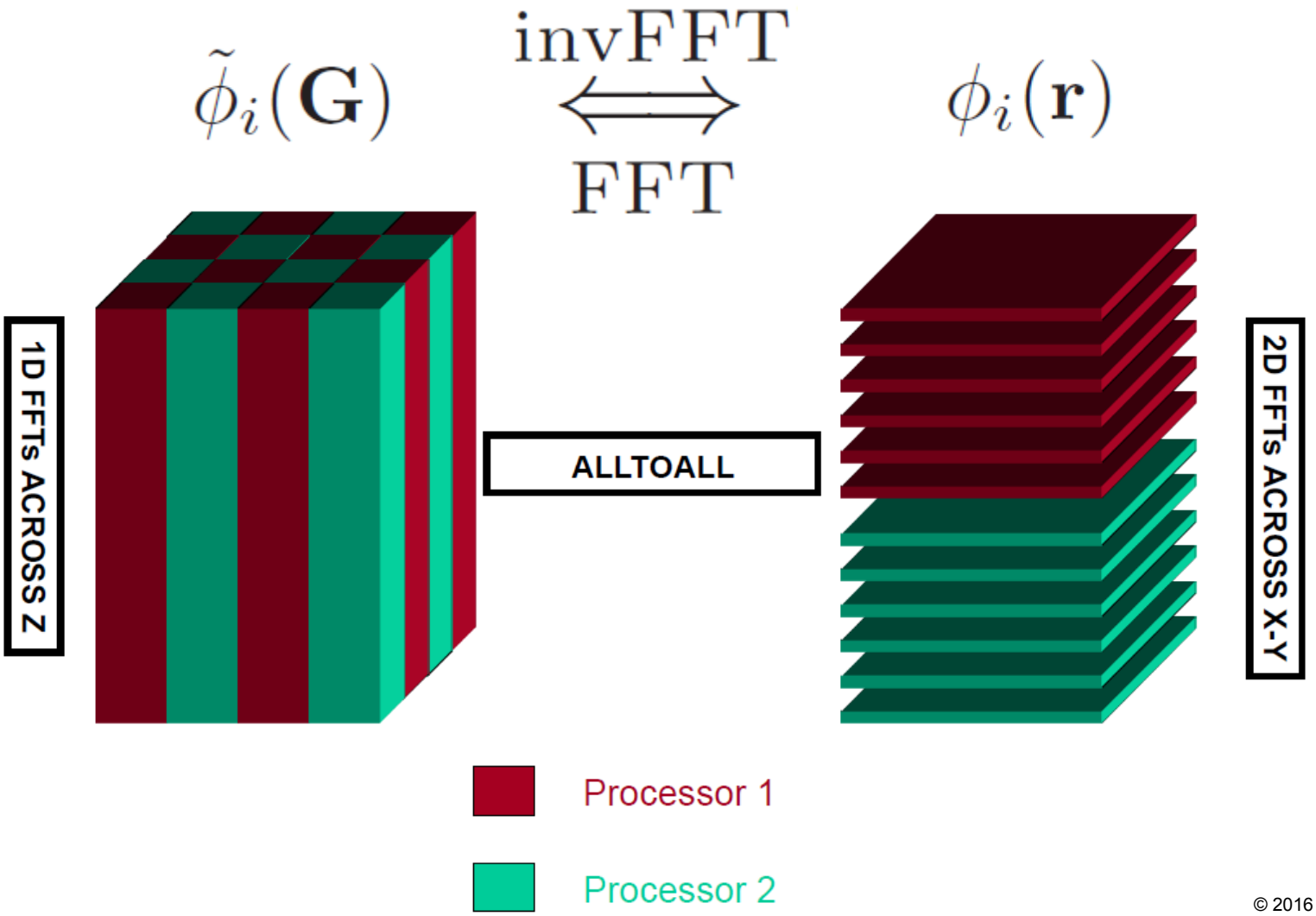
- Each SCF iteration step require at least N x 3D FFT (inverse/forward)

We focused on:

- Construction of the electronic density
- Applying the potential to the wavefunctions
- Orthogonalization of the wavefunctions

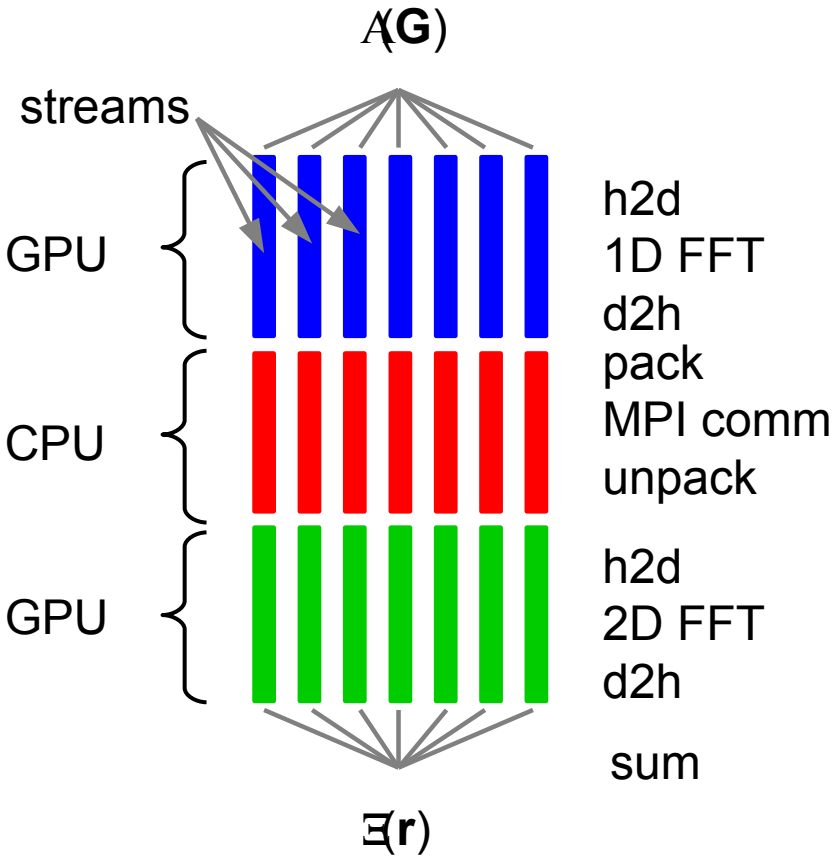
© 2016 IBM Corporation

Parallelization: Distributed Memory and 3D FFT



© 2016 IBM Corporation

GPU Porting: Construction of the electronic density



$$\phi_i(\mathbf{r}) = \mathrm{invFFT}(\tilde{\phi}_i(\mathbf{G}))$$

$$\rho(\mathbf{r}) = \sum_i^N |\phi_i(\mathbf{r})|^2$$

- The reverse Fourier transform of the N states $A(\mathbf{G})$ is distributed over the N_s streams that work concurrently.
- Each stream is assigned to a CPU thread.
- Each stream transforms a state $A(\mathbf{G})$ to the corresponding density (1D FFT – all2all – 2D FFT).

© 2016 IBM Corporation

GPU Porting: Applying the potential to the wavefunctions

$$\phi_i(\mathbf{r}) = \mathrm{invFFT}(\tilde{\phi}_i(\mathbf{G}))$$

$$V(\mathbf{r})\phi_i(\mathbf{r})$$

$$\widetilde{(V\phi_i)}(\mathbf{G}) = \mathrm{FFT}((V\phi_i)(\mathbf{r}))$$

- The reverse and forward Fourier transforms as well as the application of the potential V to the N states are distributed over N_S streams that work concurrently.
- Each stream is assigned to a CPU thread.
- Each stream transforms a state $A(\mathbf{G})$ to $A(\mathbf{r})$ (1D FFT – all2all – 2D FFT). The potential is applied and the result back transformed (2D FFT – all2all – 1D FFT).

© 2016 IBM Corporation

GPU Porting: Orthogonalization of the wavefunctions via block Gram-Schmidt

We seek the orthogonalized coefficient matrix

$$\tilde{C} = \mathrm{ortho}(C)$$

$$C = [C_1, C_2, \ldots, C_n]$$

$$[\tilde{C}_1, \ldots, \tilde{C}_{i-1}, C_i, \ldots, C_n]$$

$$\tilde{C}_i = \mathrm{ortho}((I - \sum_{j=1}^{i-1} \tilde{C}_j \tilde{C}_j^T) C_i)$$

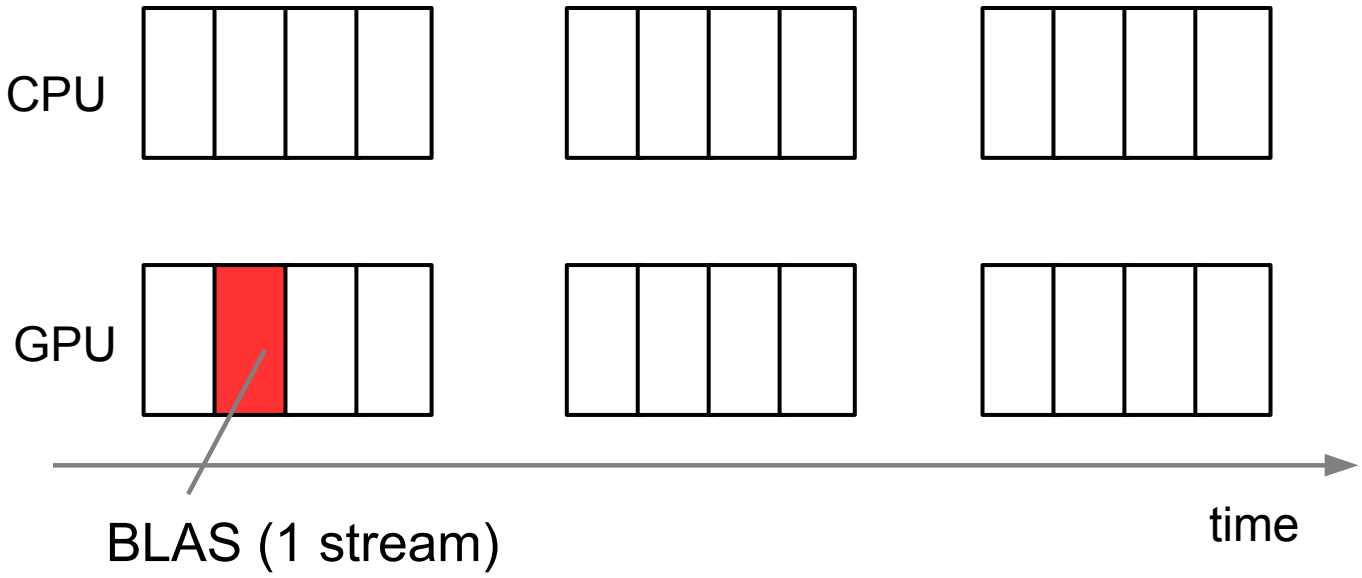
- the coefficients of the expansion of $A(\mathbf{G})$ on the plane-wave basis is block-partitioned column-wise into n blocks of size b.
- the block Gram–Schmidt scheme loops over the n blocks C_i and orthogonalizes them one after the other.

© 2016 IBM Corporation

GPU Porting: Orthogonalization of the wavefunctions via block Gram-Schmidt

$$[\tilde{C}_1, \ldots, \tilde{C}_{i-1}, C_i, \ldots, C_n]$$

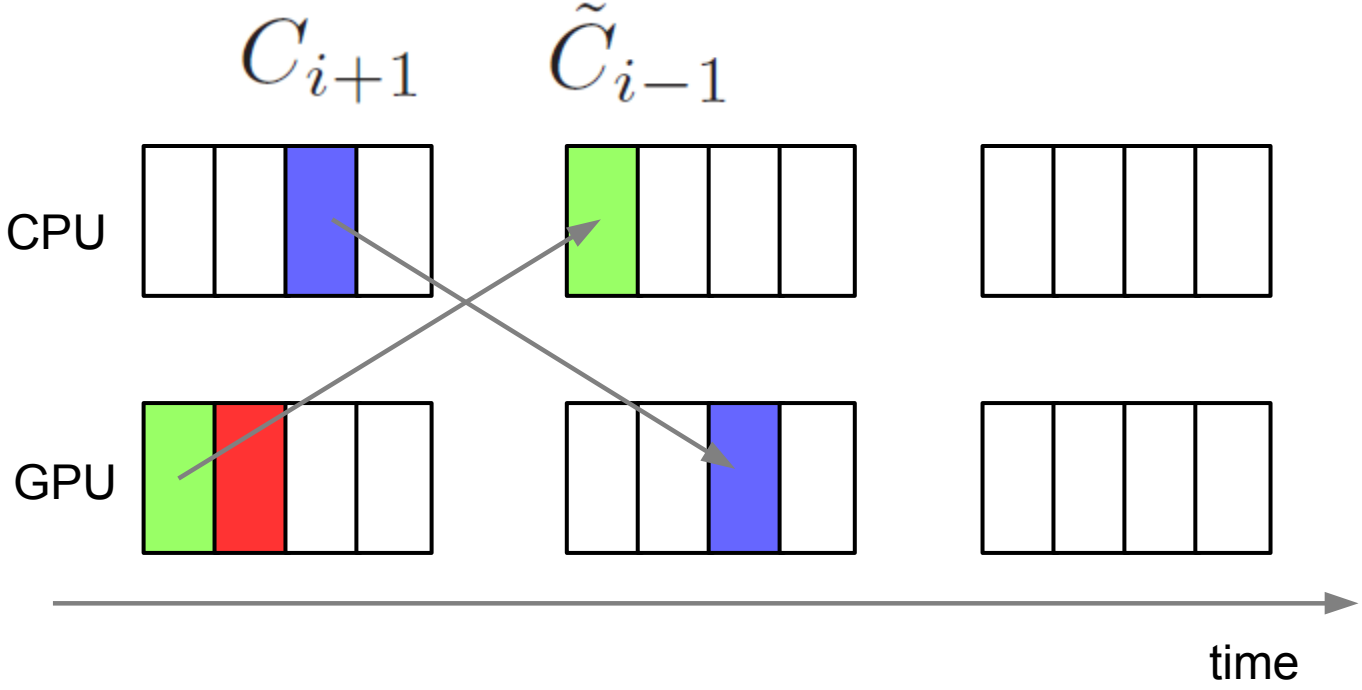
$$\tilde{C}_i = \mathrm{ortho}((I - \sum_{j=1}^{i-1} \tilde{C}_j \tilde{C}_j^T) C_i)$$



© 2016 IBM Corporation

GPU Porting: Orthogonalization of the wavefunctions via block Gram-Schmidt

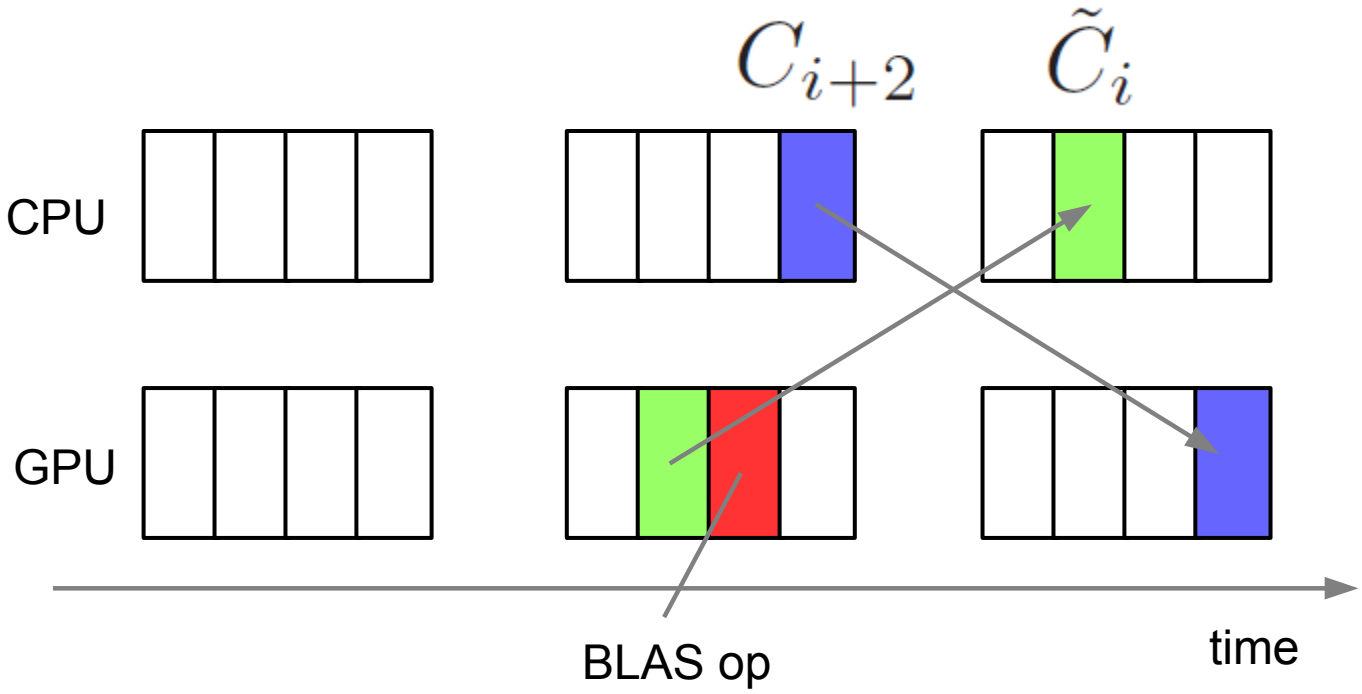
Two streams take care of D2H and H2D communication, respectively.



© 2016 IBM Corporation

GPU Porting: Orthogonalization of the wavefunctions via block Gram-Schmidt

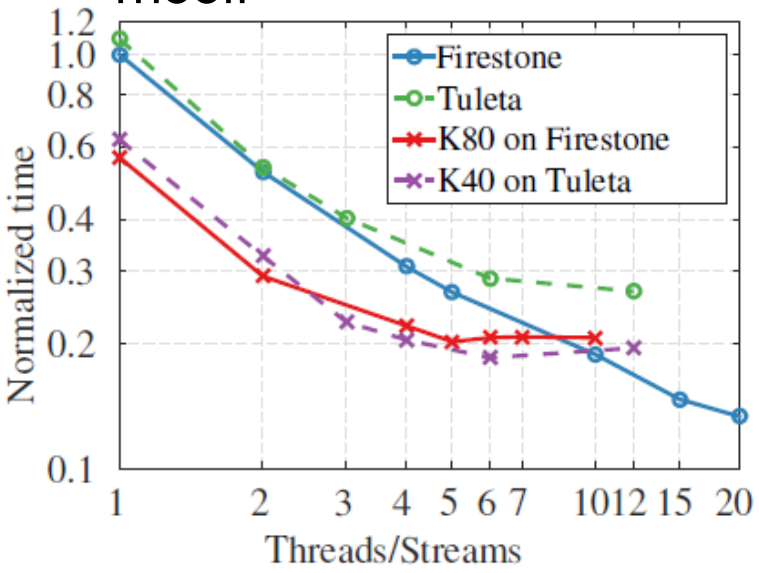
$$\tilde{C}_{i+1} = \mathrm{ortho}((I - \sum_{j=1}^{i} \tilde{C}_j \tilde{C}_j^T) C_{i+1})$$



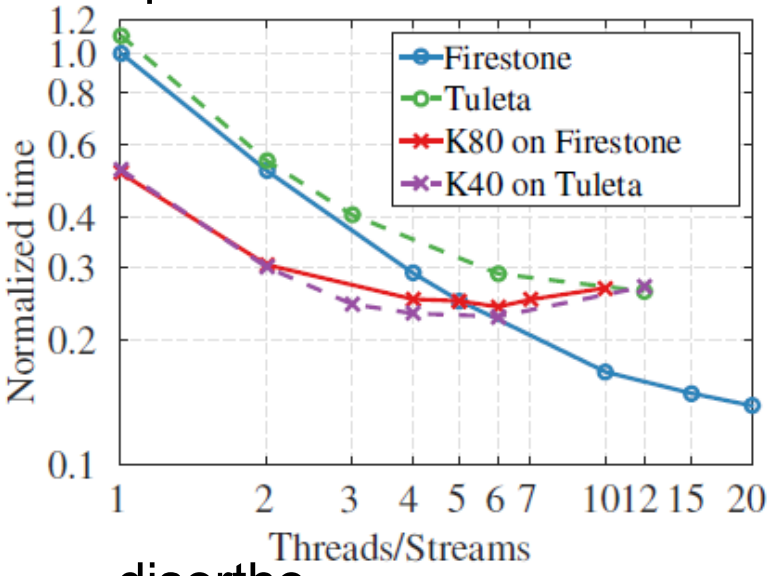
© 2016 IBM Corporation

Benchmark

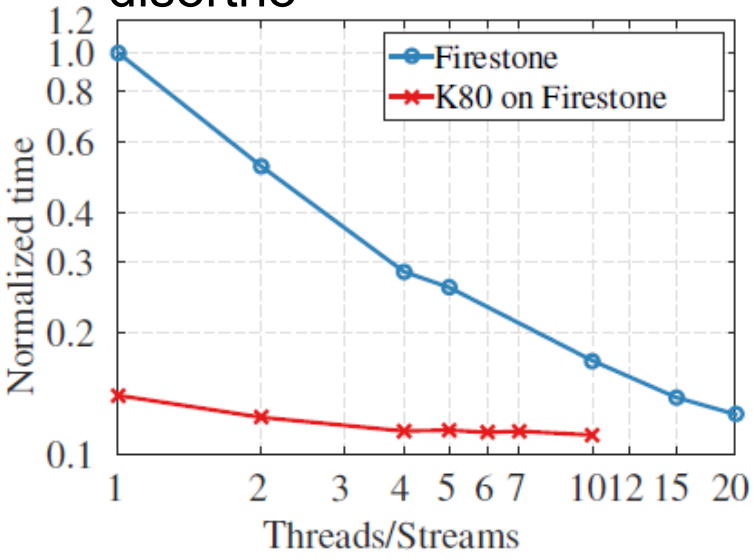
rhoofr



vpsi



disortho

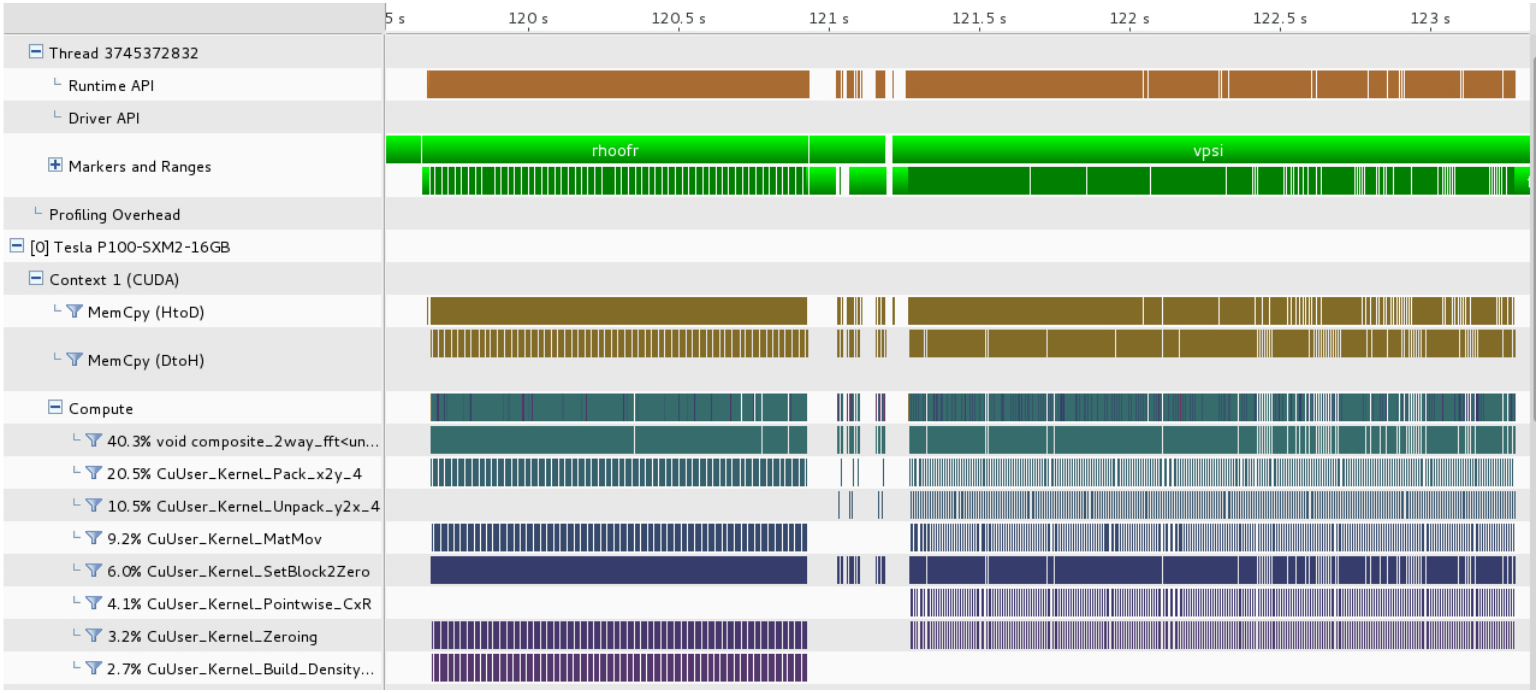


Firestone (20-cores / K80)
Tuleta (12-cores / K40)

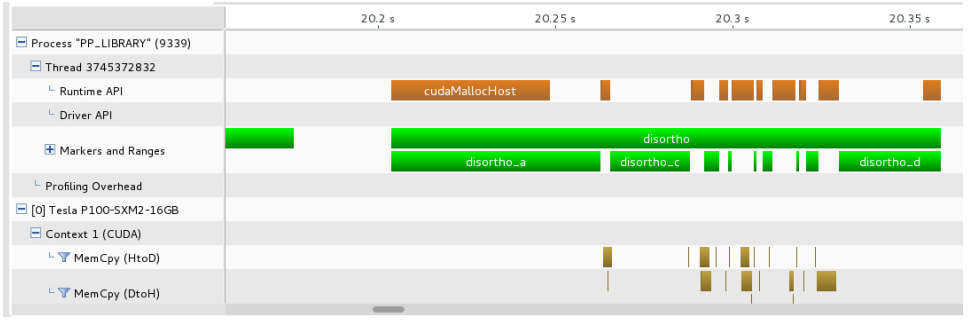
128 water box, 100Ry, GTH

Benchmark

rhoofr and vpsi



ortho



© 2016 IBM Corporation

Benchmark

Table 1: Time percentage spent in computation and memory copy from device to host (D2H) and host to device (H2D) for constructing the electronic density and applying the potential to the wavefunctions. Computation and memory copies are performed asynchronously.

Kernel	Computation [%]	D2H [%]	H2D [%]
Electronic density	27	95	76
Applying potential	30	92	89

© 2016 IBM Corporation

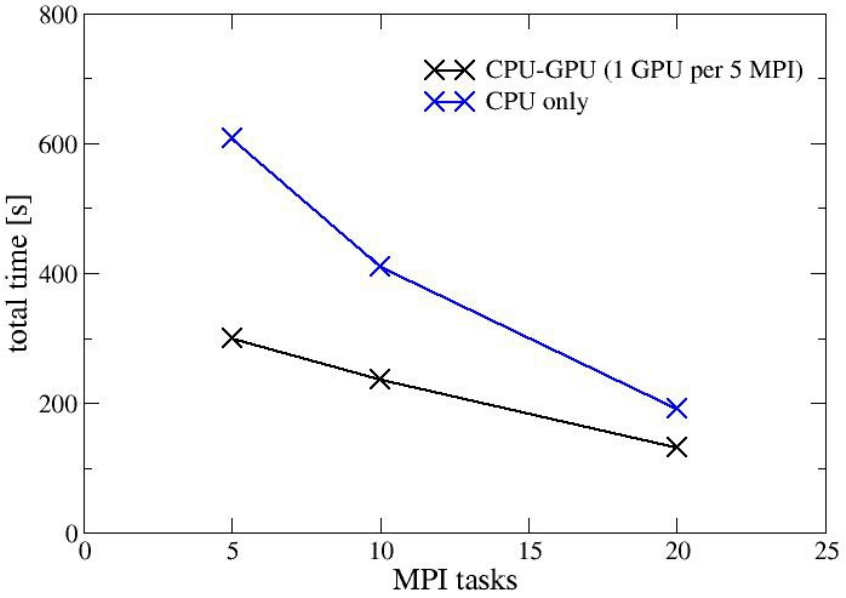
Summary and Future Directions

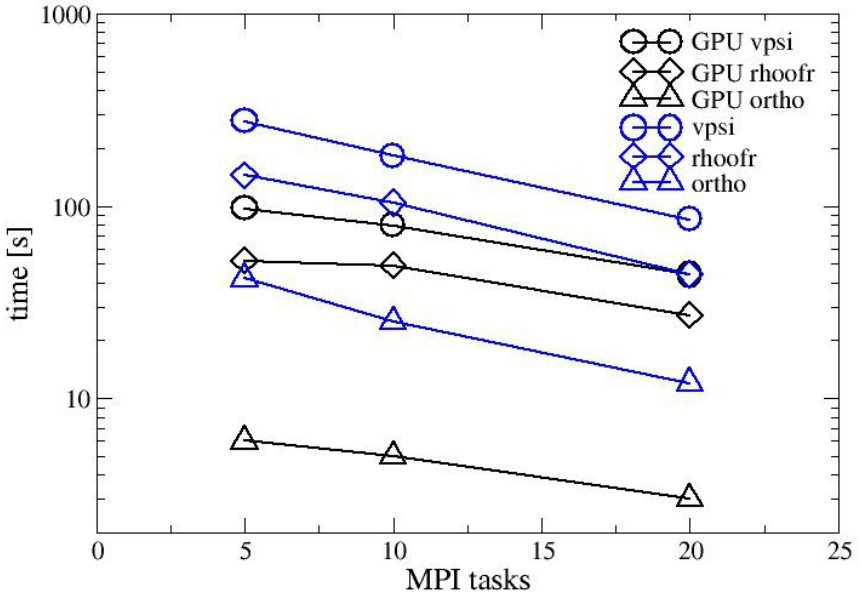
- Our initial porting phase of CPMD to OpenPOWER architectures highlights the negative impact of a limited PCI-E bandwidth between the CPU and the GPU.

- Working on reducing communication volume.
- Exploring benefit of using CUDA-aware MPI functions.
- Porting more kernels to GPU.

© 2016 IBM Corporation

Benchmark





Minsky (20-cores / 4-P100)
128 water box, 100Ry, GTH

© 2016 IBM Corporation