GPU-Accelerated Real Space Electronic Structure Theory on HPC Resources

William P. Huhn[1], Björn Lange[1], Victor Yu[1], Seyong Lee[2], Mina Yoon[3], Volker Blum[1]
1. Department of Mechanical Engineering and Materials Science, Duke University
2. Computer Science and Math Division, Oak Ridge National Laboratory
3. Center for Nanophase Materials Sciences, Oak Ridge National Laboratory

MolSSI Workshop/ELSI Conference 2018
August 17, 2018, 2:40 - 3:10 EST

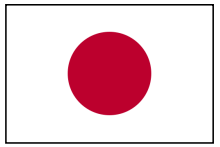
OAK RIDGE National Laboratory | OAK RIDGE LEADERSHIP COMPUTING FACILITY
ELSI
Duke UNIVERSITY | PRATT SCHOOL of ENGINEERING

Introduction

Rank	System	Cores	(TFlop/s)	(TFlop/s)	(kW)
1	**Summit** - IBM Power System AC922, IBM POWER9 22C 3.07GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband , IBM DOE/SC/Oak Ridge National Laboratory United States	2,282,544	122,300.0	187,659.3	8,806
2	**Sunway TaihuLight** - Sunway MPP, Sunway SW26010 260C 1.45GHz, Sunway , NRCPC National Supercomputing Center in Wuxi China	10,649,600	93,014.6	125,435.9	15,371
3	**Sierra** - IBM Power System S922LC, IBM POWER9 22C 3.1GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband , IBM DOE/NNSA/LLNL United States	1,572,480	71,610.0	119,193.6	
4	**Tianhe-2A** - TH-IVB-FEP Cluster, Intel Xeon E5-2692v2 12C 2.2GHz, TH Express-2, Matrix-2000 , NUDT National Super Computer Center in Guangzhou China	4,981,760	61,444.5	100,678.7	18,482
5	**AI Bridging Cloud Infrastructure (ABCI)** - PRIMERGY CX2550 M4, Xeon Gold 6148 20C 2.4GHz, NVIDIA Tesla V100 SXM2, Infiniband EDR , Fujitsu National Institute of Advanced Industrial Science and Technology (AIST) Japan	391,680	19,880.0	32,576.6	1,649
6	**Piz Daint** - Cray XC50, Xeon E5-2690v3 12C 2.6GHz, Aries interconnect , NVIDIA Tesla P100 , Cray Inc. Swiss National Supercomputing Centre (CSCS) Switzerland	361,760	19,590.0	25,326.3	2,272







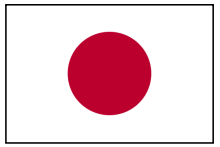

https://www.top500.org/list/2018/06/, accessed 25 June 2018

Introduction

Rank	System	Cores	(TFlop/s)	(TFlop/s)	(kW)
1	**Summit** - IBM Power System AC922, IBM POWER9 22C 3.07GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband , IBM DOE/SC/Oak Ridge National Laboratory United States	2,282,544	122,300.0	187,659.3	8,806
2	**Sunway TaihuLight** - Sunway MPP, Sunway SW26010 260C 1.45GHz, Sunway , NRCPC National Supercomputing Center in Wuxi China	10,649,600	93,014.6	125,435.9	15,371
3	**Sierra** - IBM Power System S922LC, IBM POWER9 22C 3.1GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband , IBM DOE/NNSA/LLNL United States	1,572,480	71,610.0	119,193.6	
4	**Tianhe-2A** - TH-IVB-FEP Cluster, Intel Xeon E5-2692v2 12C 2.2GHz, TH Express-2, Matrix-2000 , NUDT National Super Computer Center in Guangzhou China	4,981,760	61,444.5	100,678.7	18,482
5	**AI Bridging Cloud Infrastructure (ABCI)** - PRIMERGY CX2550 M4, Xeon Gold 6148 20C 2.4GHz, NVIDIA Tesla V100 SXM2, Infiniband EDR , Fujitsu National Institute of Advanced Industrial Science and Technology (AIST) Japan	391,680	19,880.0	32,576.6	1,649
6	**Piz Daint** - Cray XC50, Xeon E5-2690v3 12C 2.6GHz, Aries interconnect NVIDIA Tesla P100 Cray Inc. Swiss National Supercomputing Centre (CSCS) Switzerland	361,760	19,590.0	25,326.3	2,272








https://www.top500.org/list/2018/06/, accessed 25 June 2018

Introduction

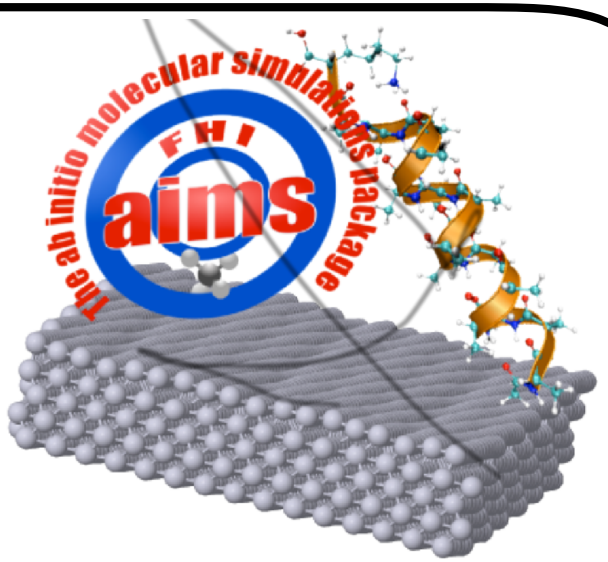
Supporting general-purpose GPU (GPGPU) acceleration is critical for running electronic structure theory on current leading HPC platforms!

Introduction

Supporting general-purpose GPU (GPGPU) acceleration is critical for running electronic structure theory on current leading HPC platforms!

Code targeted: FHI-aims

- All-Electron, Full-Potential KS-DFT
- Localized basis sets: Numeric atom-centered orbitals (NAOs)
- Developed by an active, globally-distributed academic community of 100+ developers

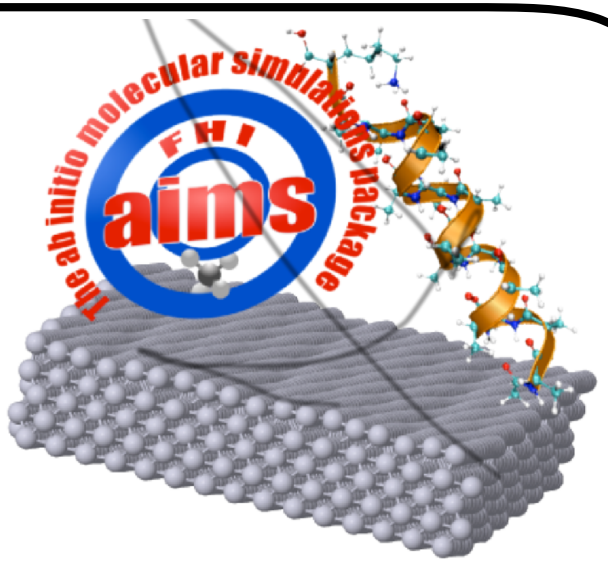


Introduction

Supporting general-purpose GPU (GPGPU) acceleration is critical for running electronic structure theory on current leading HPC platforms!

Code targeted: FHI-aims

- All-Electron, Full-Potential KS-DFT
- Localized basis sets: Numeric atom-centered orbitals (NAOs)
- Developed by an active, globally-distributed academic community of 100+ developers



Δ-Project Places FHI-aims on a Similar Accuracy Level to LAPW

Code	Version	Basis	Δ-value
WIEN2k	13.1	LAPW/APW+lo	0 meV/atom
FHI-aims	081213	tier2 numerical orbitals	0.2 meV/atom

https://molmod.ugent.be/deltacodesdft

What is Needed for GPGPU-Accelerated KS-DFT?

For (non-hybrid) KS-DFT using real-space operations, the rate-limiting steps are:

Integration of the Hamiltonian Matrix Elements

$$H_{mn} = \int \varphi_m^*(\boldsymbol{r}) \hat{h} \varphi_n(\boldsymbol{r}) d\boldsymbol{r}$$

Electron Density Calculation

$$\rho(\boldsymbol{r}) = \sum_{mn} \varphi_m^*(\boldsymbol{r}) D_{mn} \varphi_n(\boldsymbol{r})$$

Solution (or circumvention) of the KS-DFT Equation

$$\left[\hat{t} + \hat{v}_{ES} + \hat{v}_{XC}\right] \psi_m(\boldsymbol{r}) = \epsilon \psi_m(\boldsymbol{r})$$

What is Needed for GPGPU-Accelerated KS-DFT?

For (non-hybrid) KS-DFT using real-space operations, the rate-limiting steps are:

Small/Mid-Scale Calculations

Integration of the Hamiltonian Matrix Elements

$$H_{mn} = \int \varphi_m^*(\boldsymbol{r}) \hat{h} \varphi_n(\boldsymbol{r}) d\boldsymbol{r}$$

Electron Density Calculation

$$\rho(\boldsymbol{r}) = \sum_{mn} \varphi_m^*(\boldsymbol{r}) D_{mn} \varphi_n(\boldsymbol{r})$$

Large-Scale Calculations

Solution (or circumvention) of the KS-DFT Equation

$$\left[\hat{t} + \hat{v}_{ES} + \hat{v}_{XC}\right] \psi_m(\boldsymbol{r}) = \epsilon \psi_m(\boldsymbol{r})$$

What is Needed for GPGPU-Accelerated KS-DFT?

For (non-hybrid) KS-DFT using real-space operations, the rate-limiting steps are:

Integration of the Hamiltonian Matrix Elements

$$H_{mn} = \int \varphi_m^*(\boldsymbol{r}) \hat{h} \varphi_n(\boldsymbol{r}) d\boldsymbol{r}$$

Electron Density Calculation

$$\rho(\boldsymbol{r}) = \sum_{mn} \varphi_m^*(\boldsymbol{r}) D_{mn} \varphi_n(\boldsymbol{r})$$

Small/Mid-Scale Calculations

$O(N_{atom})$ Batch Integration+ GPGPUs

Solution (or circumvention) of the KS-DFT Equation

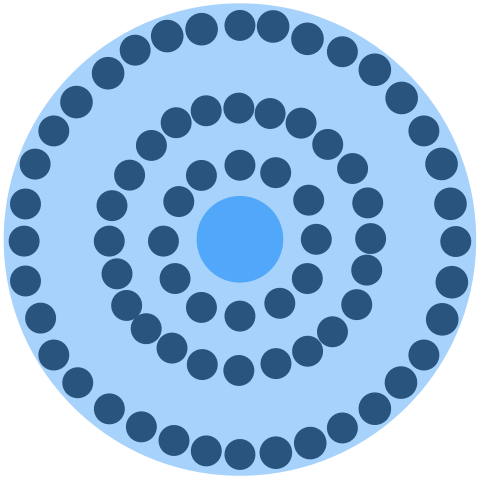
$$\left[\hat{t} + \hat{v}_{ES} + \hat{v}_{XC}\right] \psi_m(\boldsymbol{r}) = \epsilon \psi_m(\boldsymbol{r})$$

Large-Scale Calculations

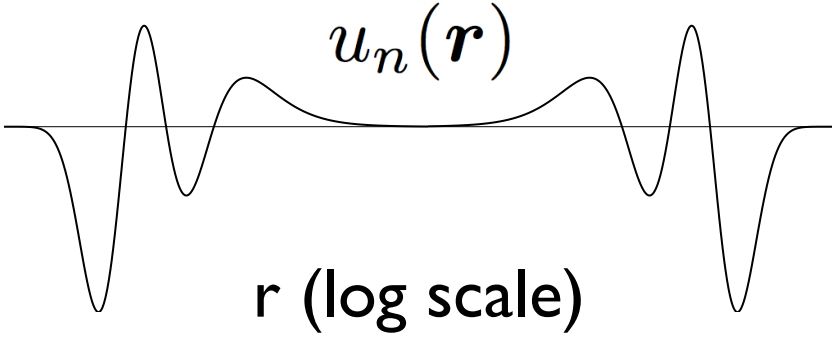
ELSI

elsi-interchange.org

Numeric Atom-Centered Orbitals



$$\varphi_{nlm}(\boldsymbol{r}) = \frac{u_n(\boldsymbol{r})}{\boldsymbol{r}} Y_{lm}(\boldsymbol{r})$$



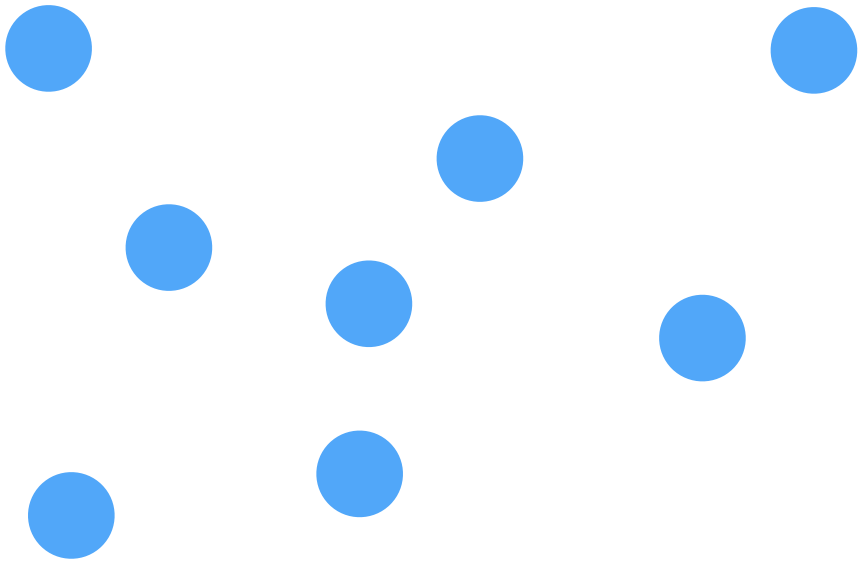
- **Integration points** distributed on radial grids around atoms
- Basis elements localized in space give O(N) real-space operation
- Capture core region "wiggles"; naturally all-electron
- Pre-constructed "tiers" of basis elements suitable for DFT:

Basis Set	H	C	Si
Minimum	1s	1s2s2p	1s2s2p3s3p
Tier 1	+sp	+pds	+dpfs
Tier 2	+spsd	+fpsgd	+dgps
…	…	…	…

Blum et al., Comp. Phys. Comm. 2009

Grid Adapted Cut-Plane Partitioning

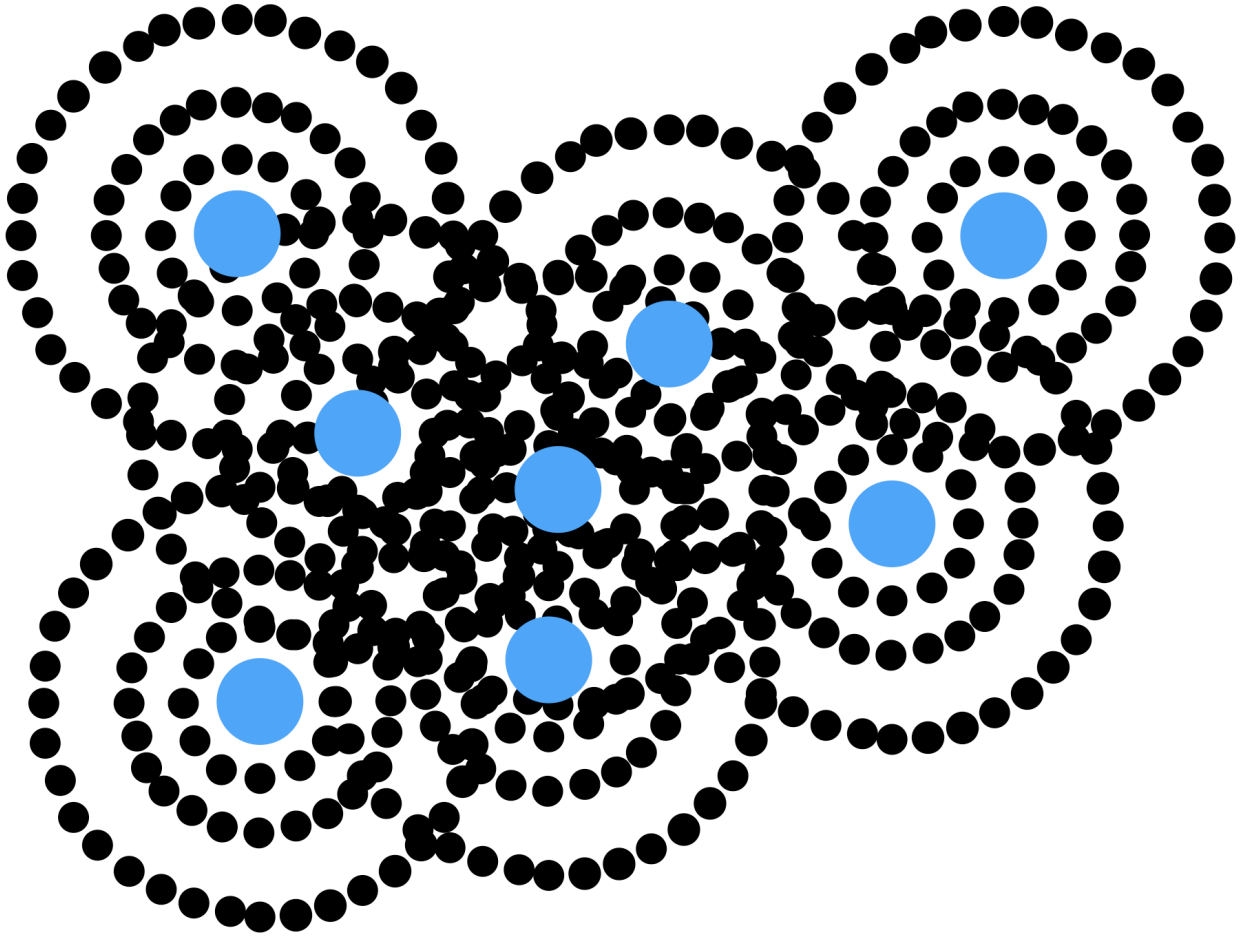
Divide integration grid into compact sets of points (**batches**) via cut planes iteratively



Becke, J. Phys. Chem 1988; Delley, J.Phys. Chem.1990; Havu, J Comp. Phys 2009

Grid Adapted Cut-Plane Partitioning

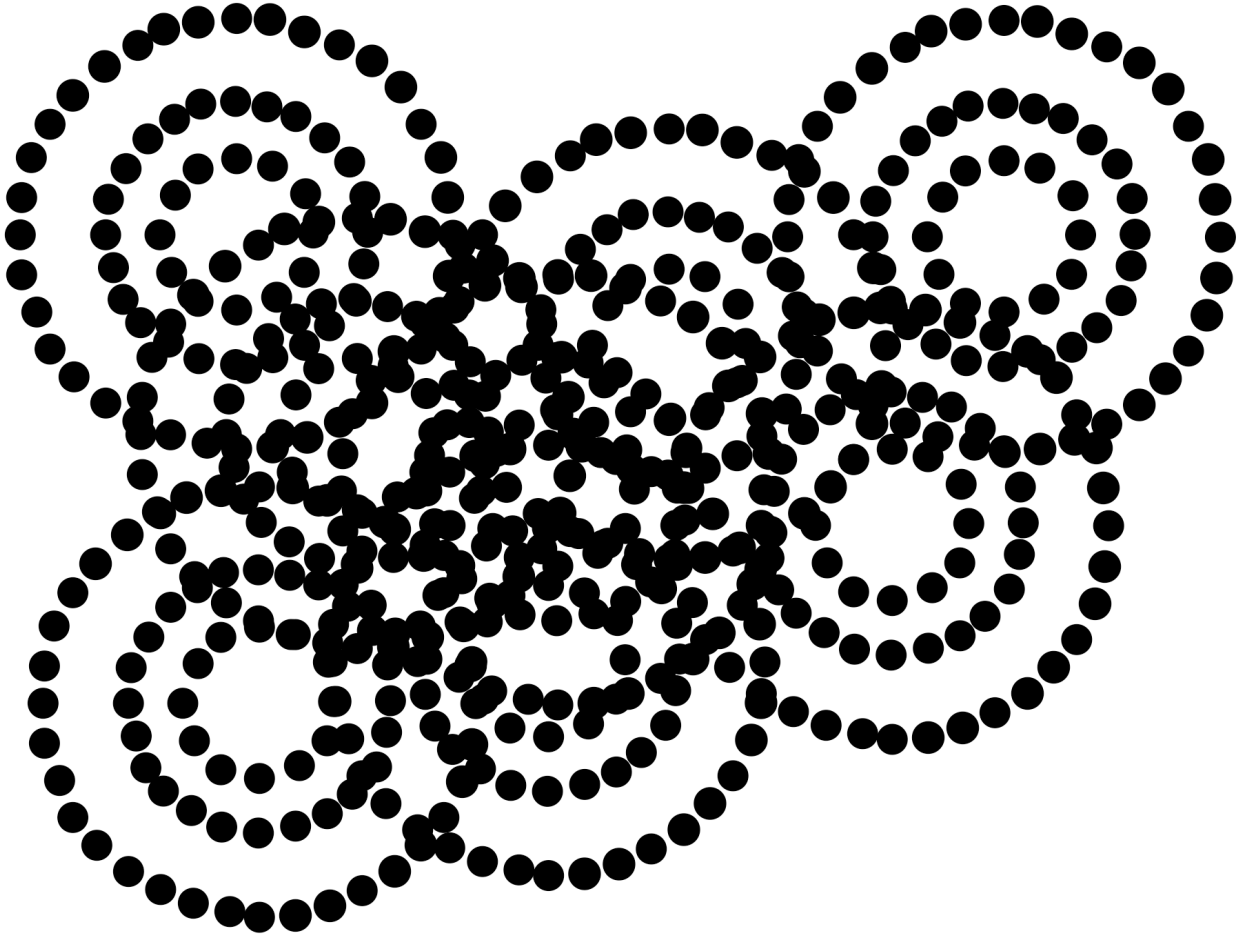
Divide integration grid into compact sets of points (**batches**) via cut planes iteratively



Becke, J. Phys. Chem 1988; Delley, J.Phys. Chem.1990; Havu, J Comp. Phys 2009

Grid Adapted Cut-Plane Partitioning

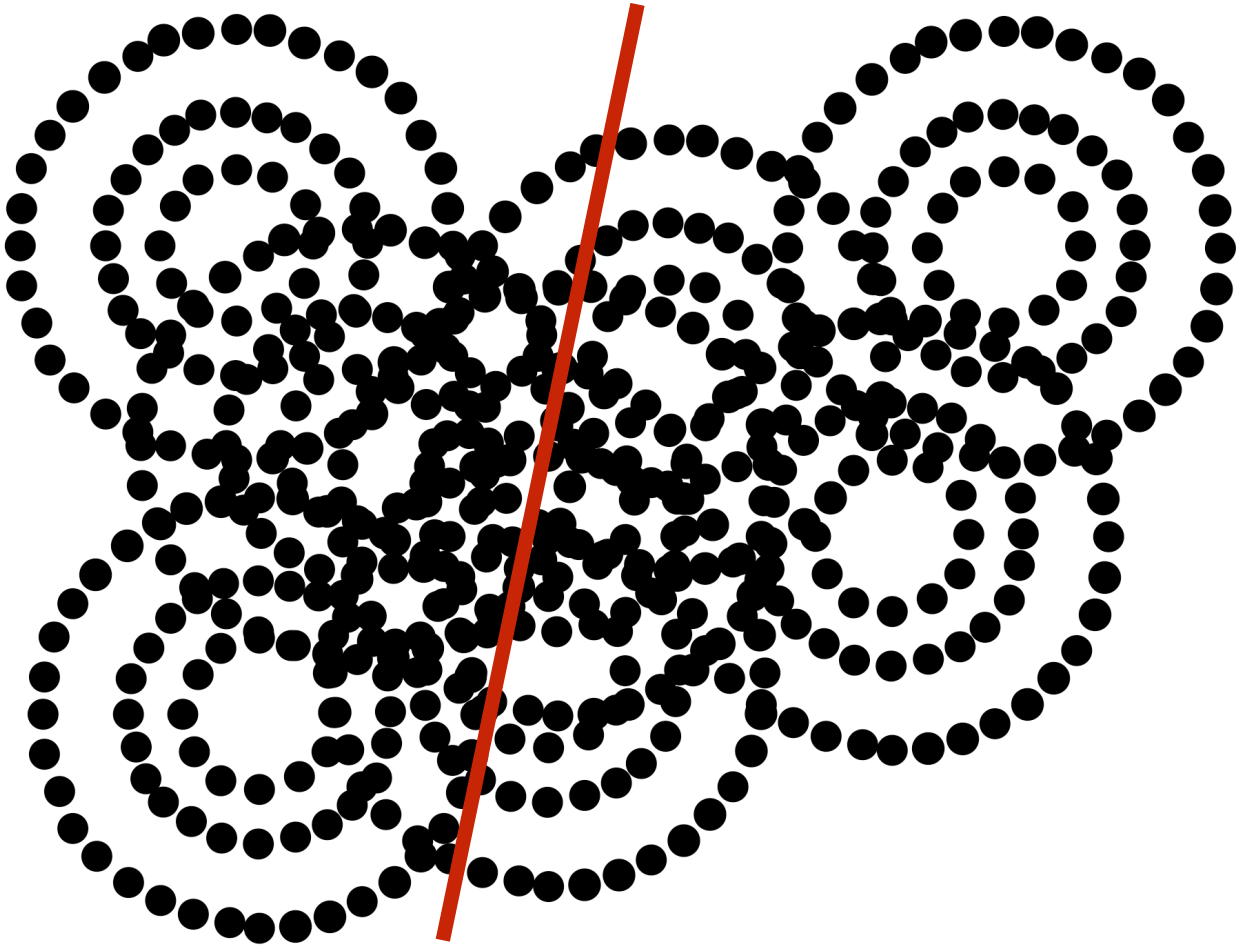
Divide integration grid into compact sets of points (**batches**) via cut planes iteratively



Becke, J. Phys. Chem 1988; Delley, J.Phys. Chem.1990; Havu, J Comp. Phys 2009

Grid Adapted Cut-Plane Partitioning

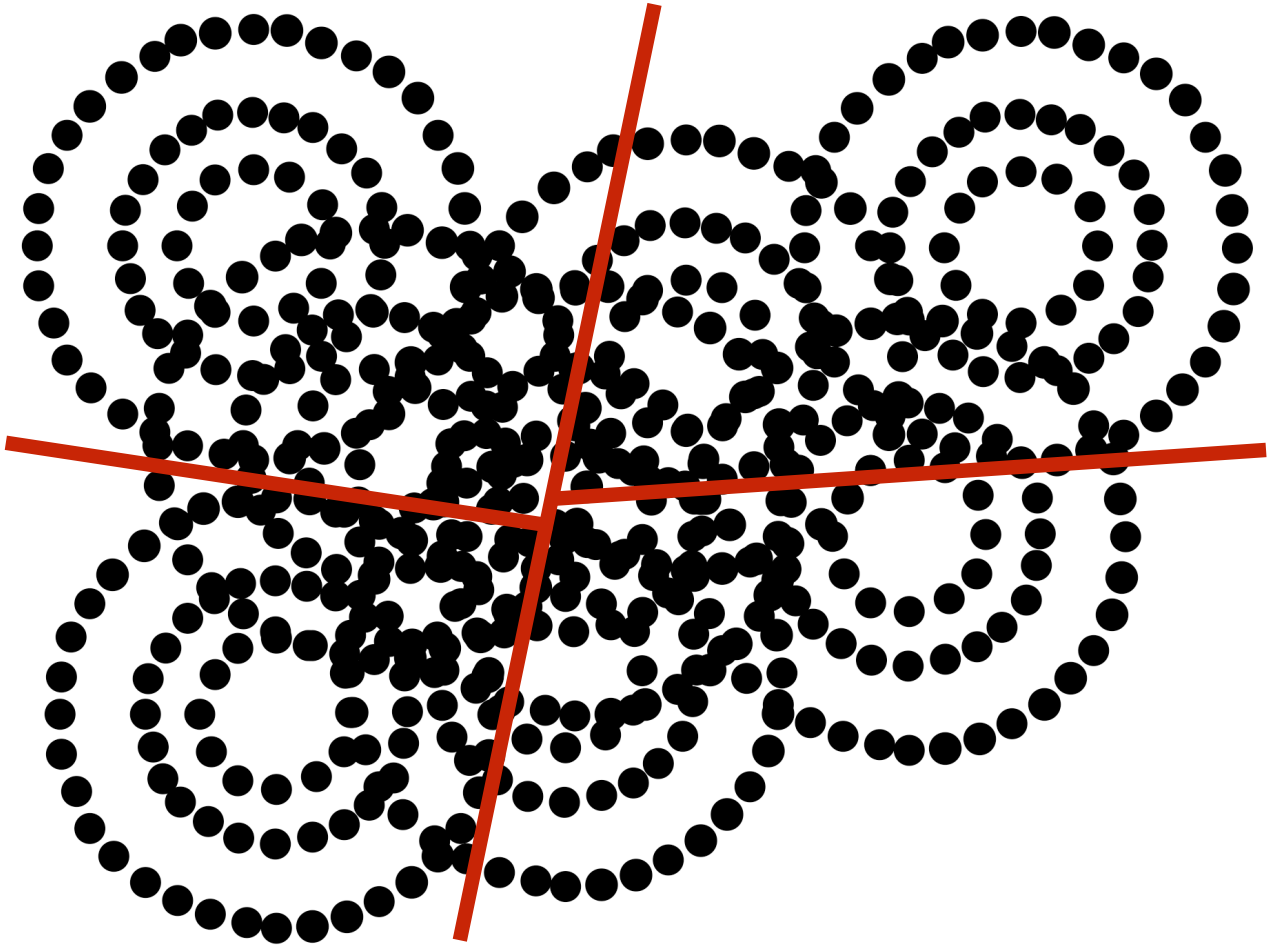
Divide integration grid into compact sets of points (**batches**) via cut planes iteratively



Becke, J. Phys. Chem 1988; Delley, J.Phys. Chem.1990; Havu, J Comp. Phys 2009

Grid Adapted Cut-Plane Partitioning

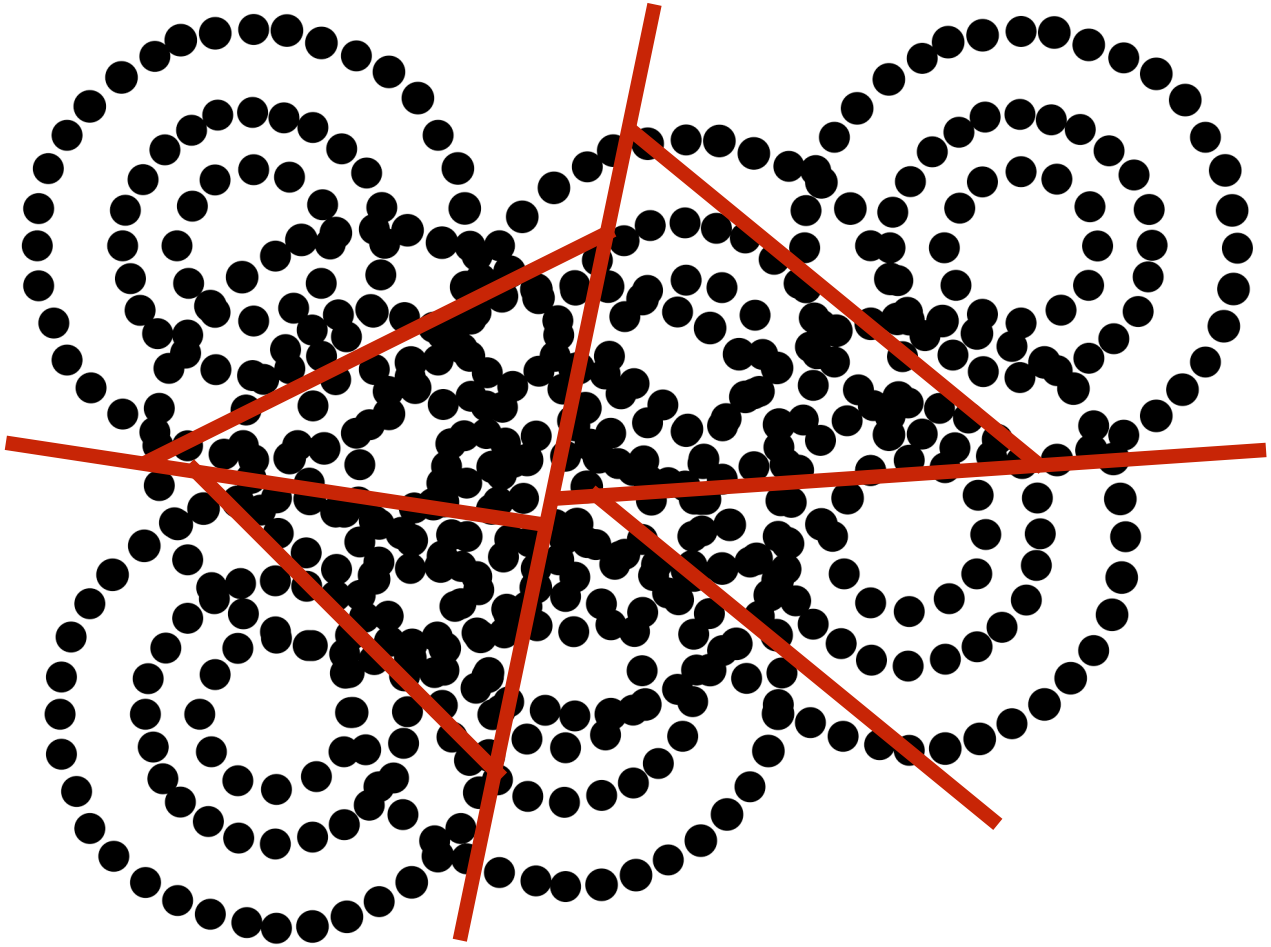
Divide integration grid into compact sets of points (**batches**) via cut planes iteratively



Becke, J. Phys. Chem 1988; Delley, J.Phys. Chem.1990; Havu, J Comp. Phys 2009

Grid Adapted Cut-Plane Partitioning

Divide integration grid into compact sets of points (**batches**) via cut planes iteratively

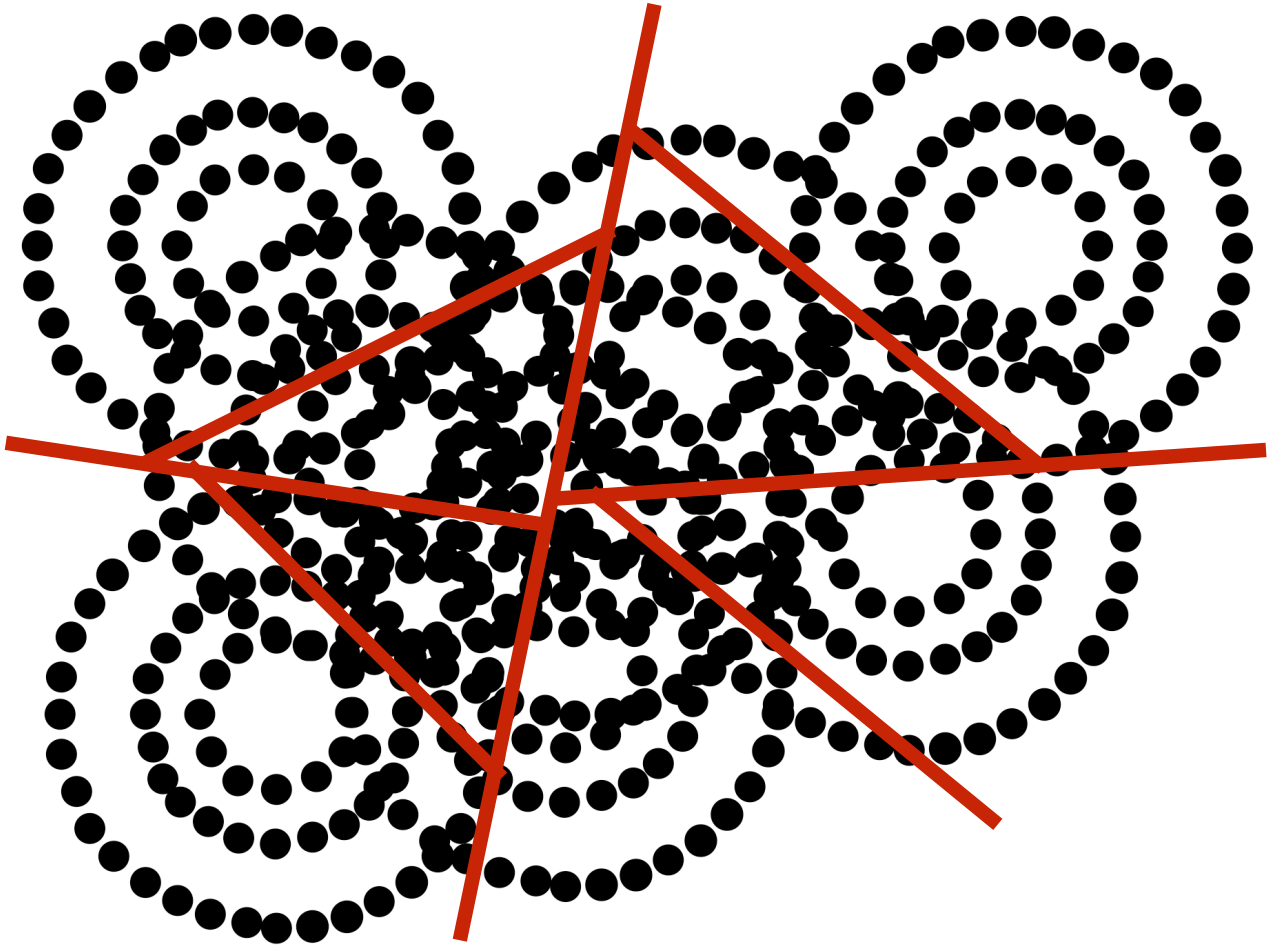


Becke, J. Phys. Chem 1988; Delley, J.Phys. Chem.1990; Havu, J Comp. Phys 2009

Grid Adapted Cut-Plane Partitioning

Divide integration grid into compact sets of points (**batches**) via cut planes iteratively

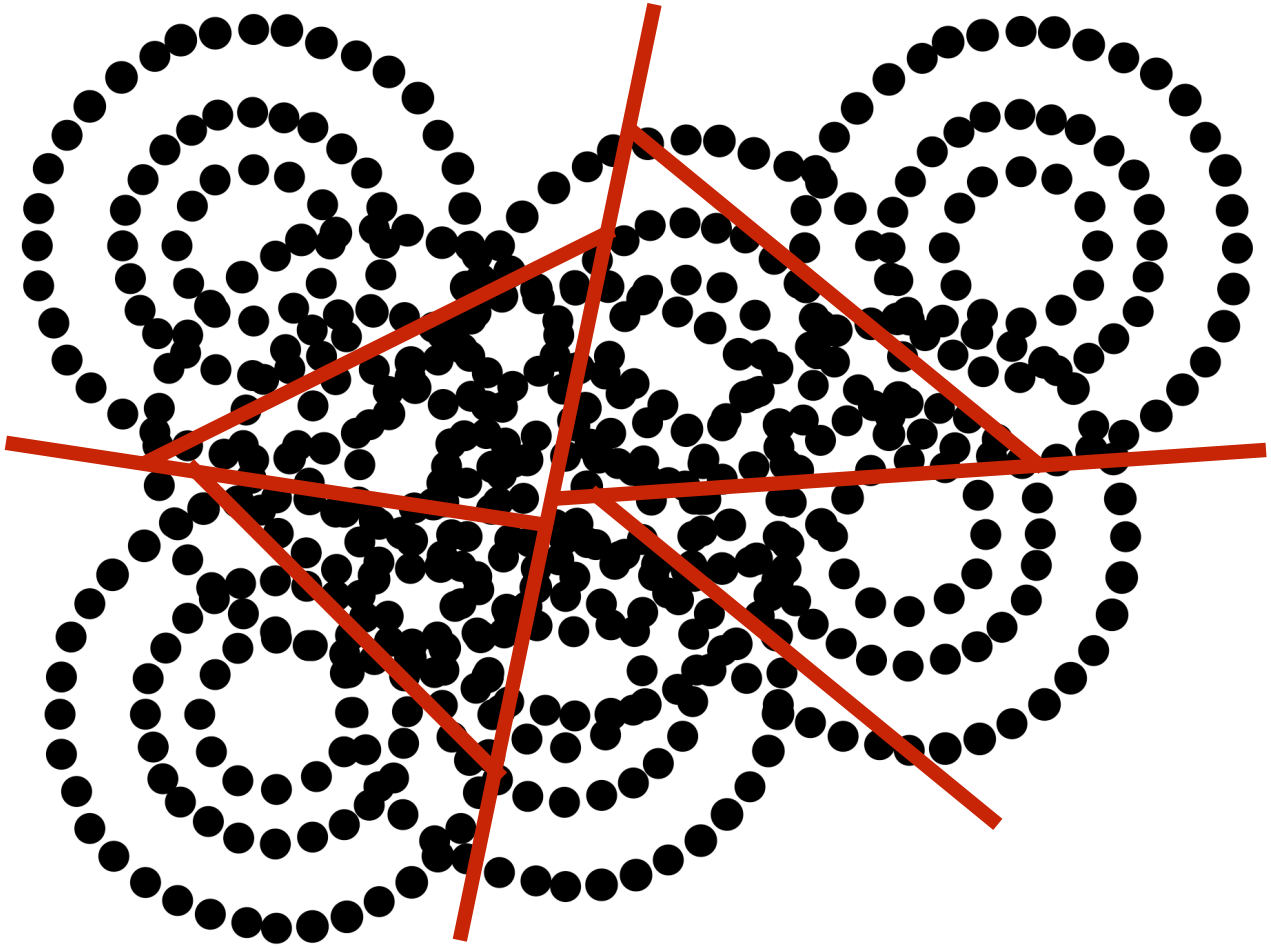
As system size grows, number of batches grows linearly



Becke, J. Phys. Chem 1988; Delley, J.Phys. Chem.1990; Havu, J Comp. Phys 2009

Grid Adapted Cut-Plane Partitioning

Divide integration grid into compact sets of points (**batches**) via cut planes iteratively



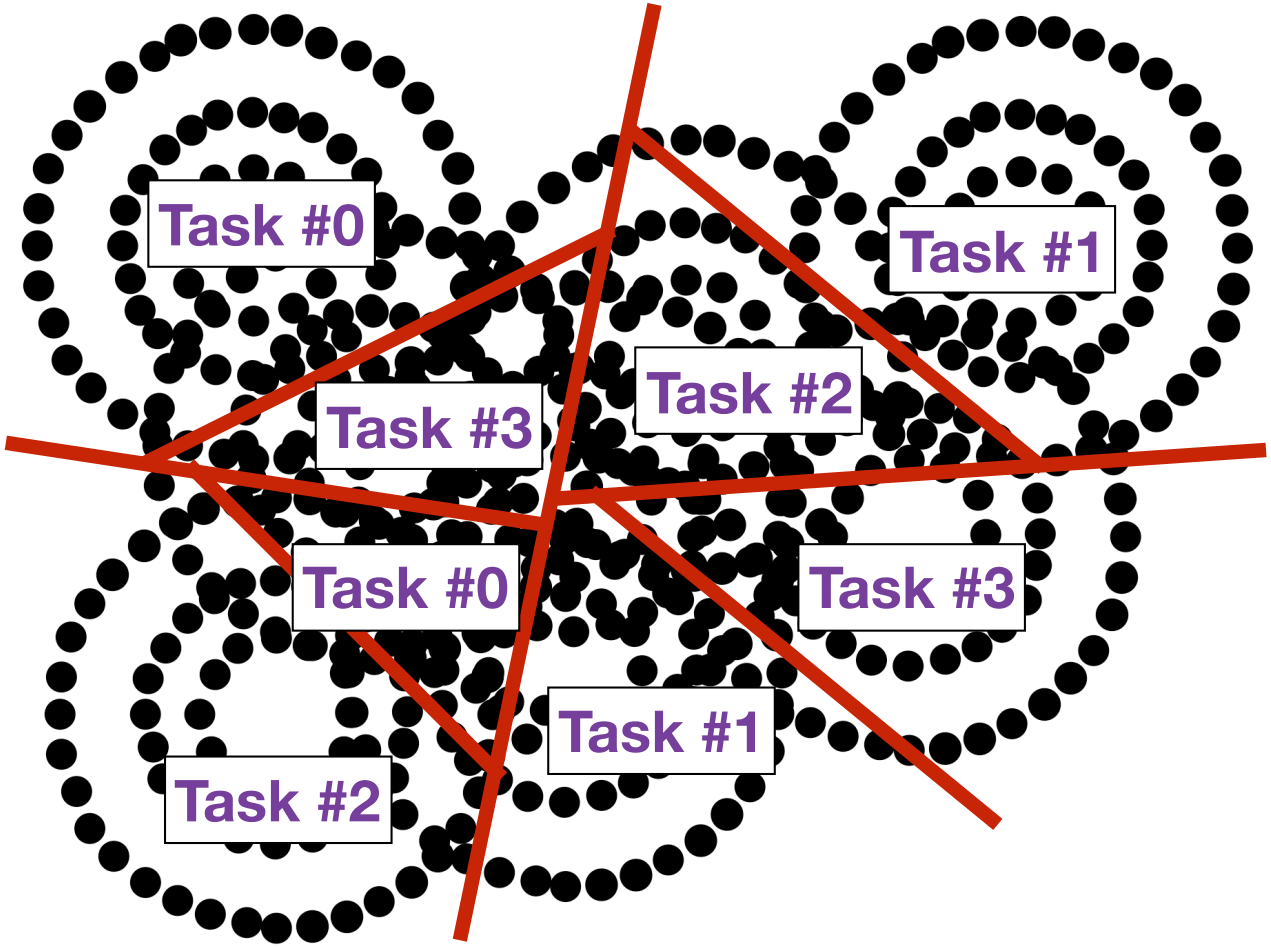
As system size grows, number of batches grows linearly

Batches are embarrassingly parallel over tasks

Becke, J. Phys. Chem 1988; Delley, J.Phys. Chem.1990; Havu, J Comp. Phys 2009

Grid Adapted Cut-Plane Partitioning

Divide integration grid into compact sets of points (**batches**) via cut planes iteratively



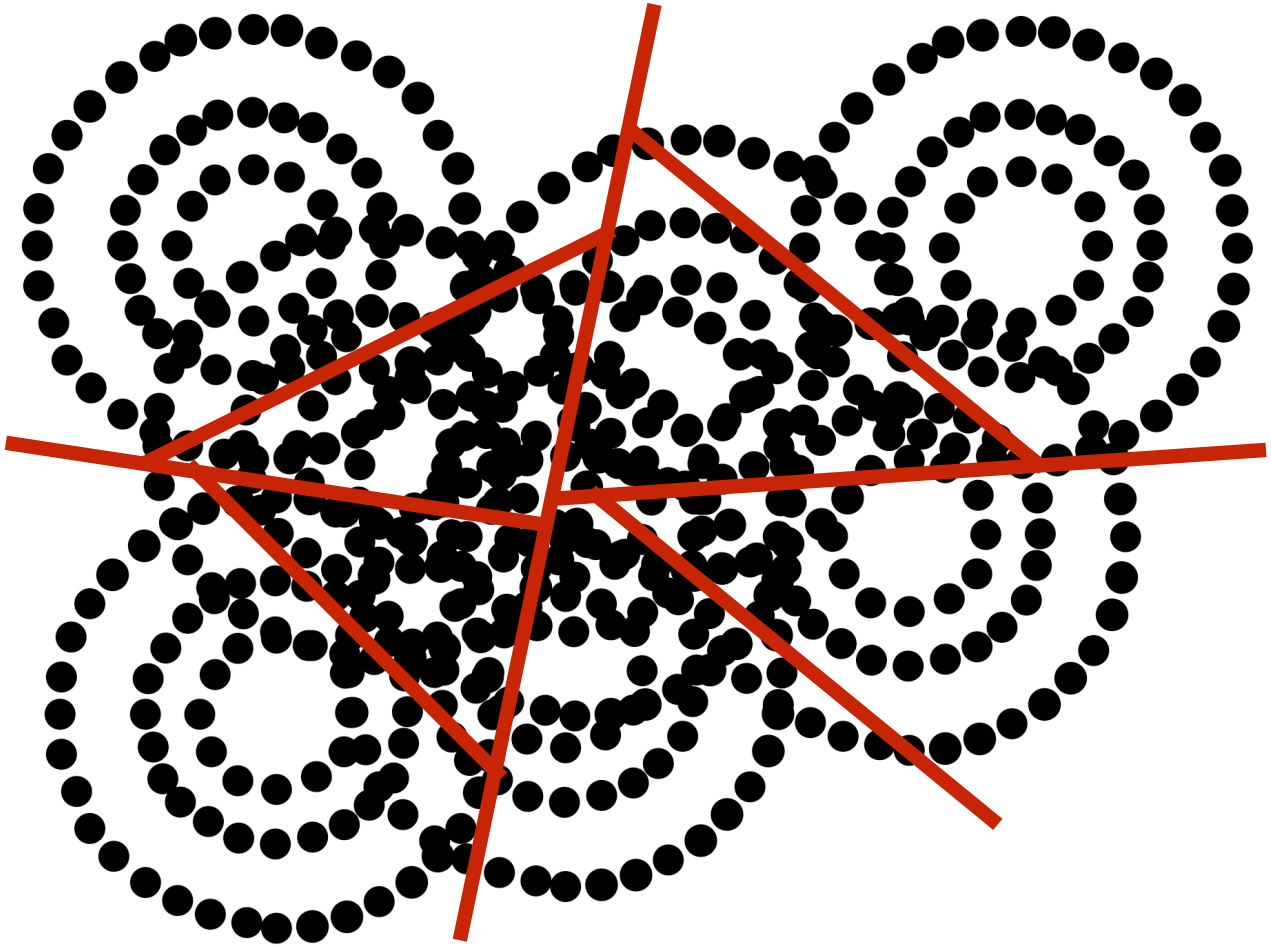
As system size grows, number of batches grows linearly

Batches are embarrassingly parallel over tasks

Becke, J. Phys. Chem 1988; Delley, J.Phys. Chem.1990; Havu, J Comp. Phys 2009

Grid Adapted Cut-Plane Partitioning

Divide integration grid into compact sets of points (**batches**) via cut planes iteratively



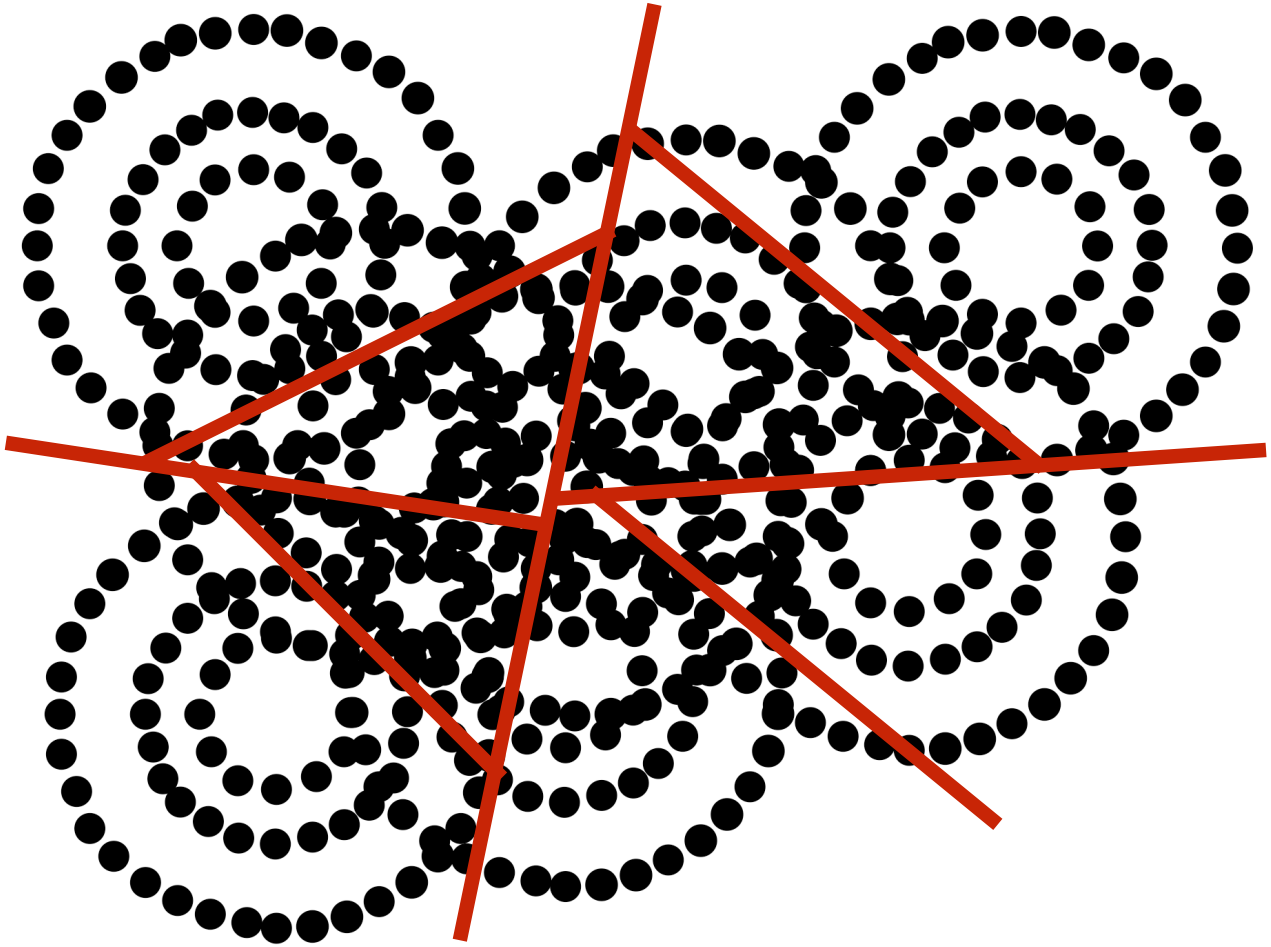
As system size grows, number of batches grows linearly

Batches are embarrassingly parallel over tasks

Becke, J. Phys. Chem 1988; Delley, J.Phys. Chem.1990; Havu, J Comp. Phys 2009

Grid Adapted Cut-Plane Partitioning

Divide integration grid into compact sets of points (**batches**) via cut planes iteratively



As system size grows, number of batches grows linearly

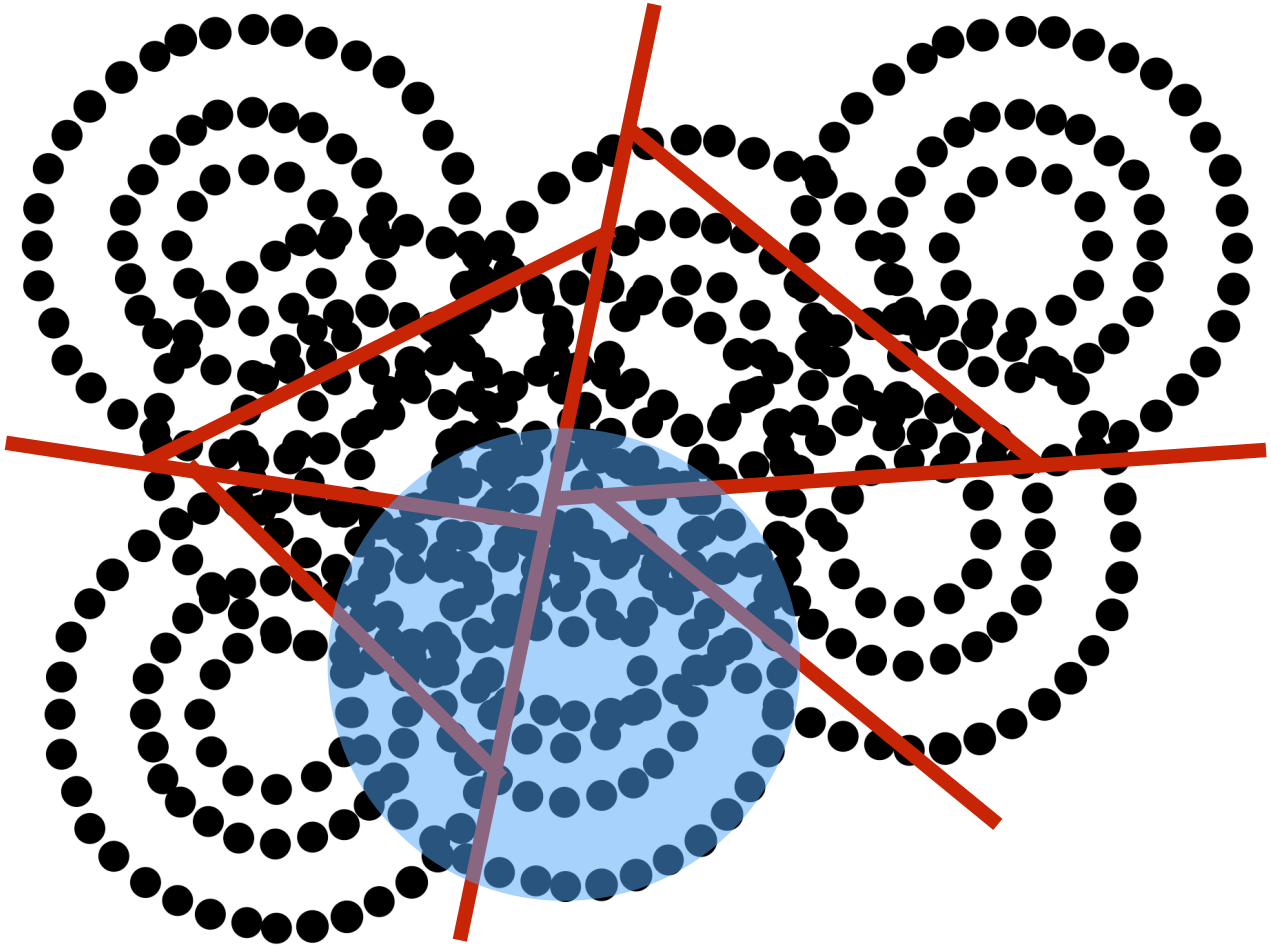
Batches are embarrassingly parallel over tasks

Only basis elements touching a batch contribute to work done for batch

Becke, J. Phys. Chem 1988; Delley, J.Phys. Chem.1990; Havu, J Comp. Phys 2009

Grid Adapted Cut-Plane Partitioning

Divide integration grid into compact sets of points (**batches**) via cut planes iteratively



As system size grows, number of batches grows linearly

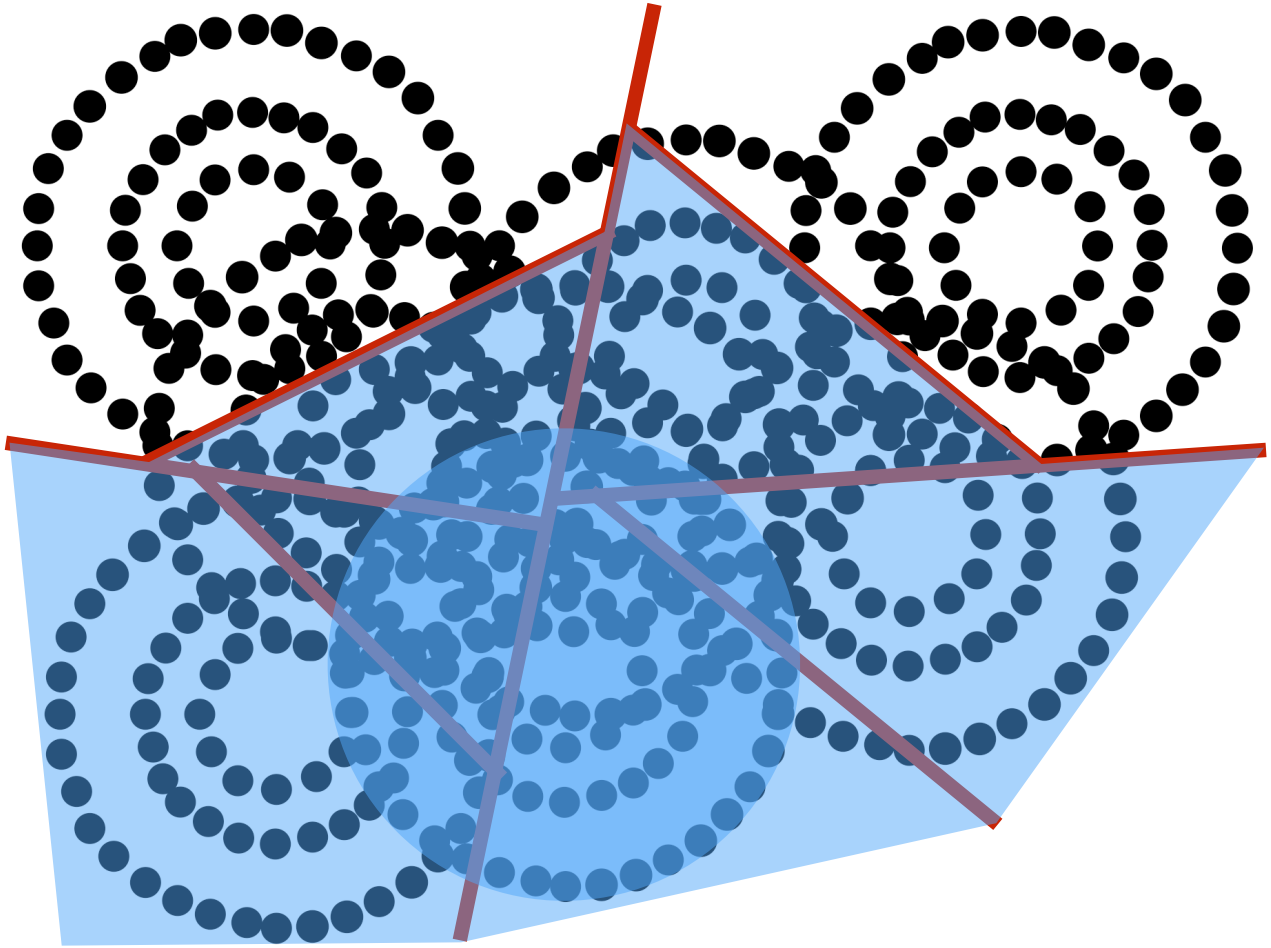
Batches are embarrassingly parallel over tasks

Only basis elements touching a batch contribute to work done for batch

Becke, J. Phys. Chem 1988; Delley, J.Phys. Chem.1990; Havu, J Comp. Phys 2009

Grid Adapted Cut-Plane Partitioning

Divide integration grid into compact sets of points (**batches**) via cut planes iteratively



As system size grows, number of batches grows linearly

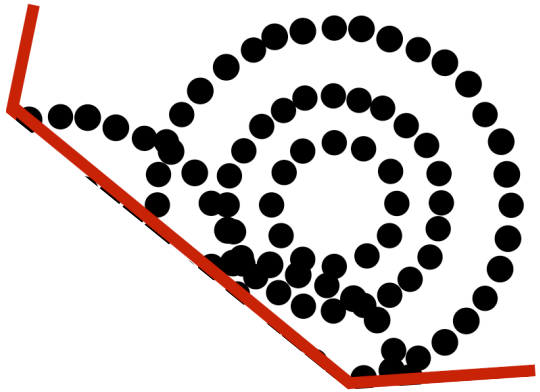
Batches are embarrassingly parallel over tasks

Only basis elements touching a batch contribute to work done for batch

Becke, J. Phys. Chem 1988; Delley, J.Phys. Chem.1990; Havu, J Comp. Phys 2009

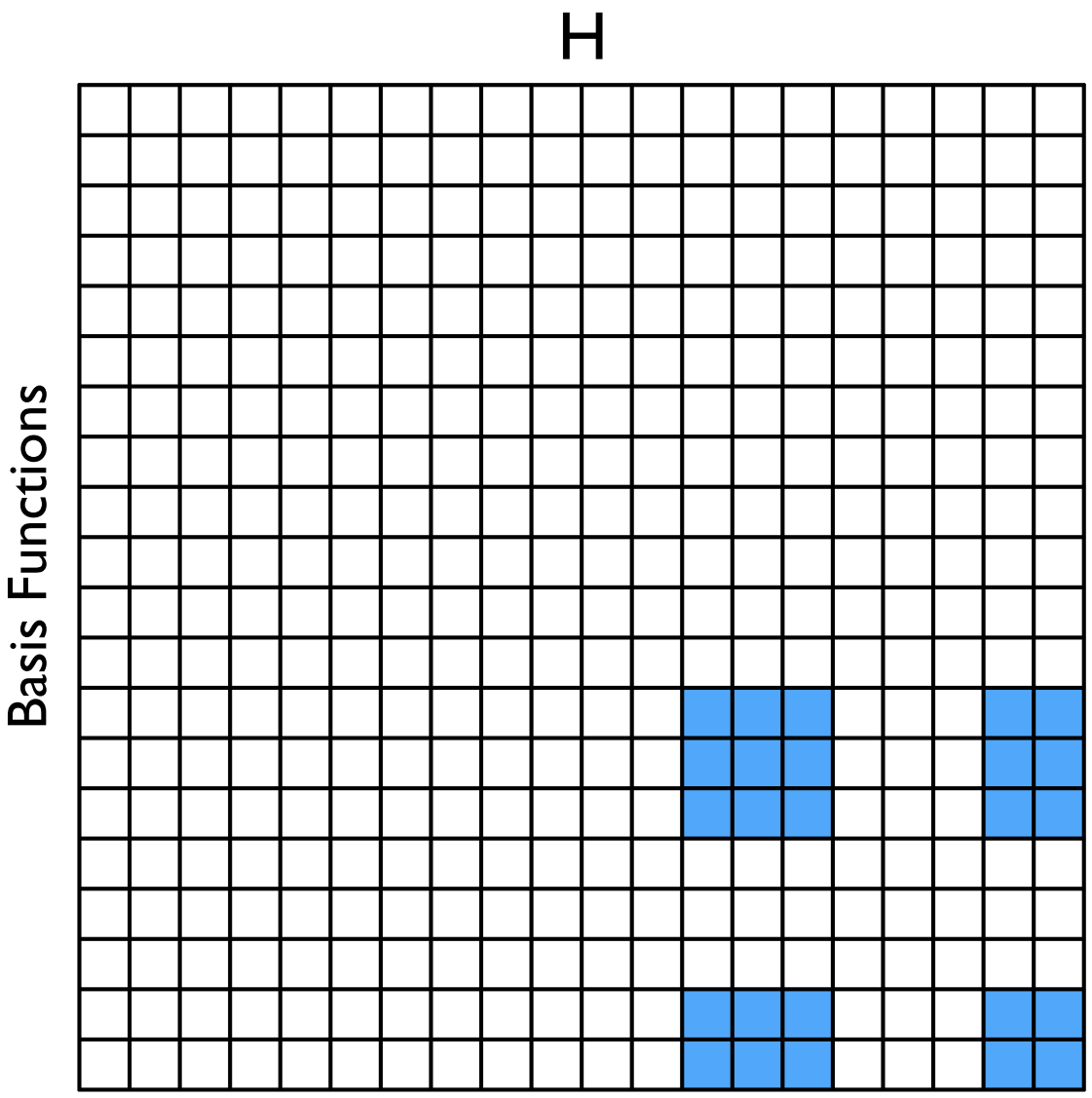
Divide-And-Conquer on the Real-Space Matrix

Each batch only contributes to a small number of real-space matrix elements…

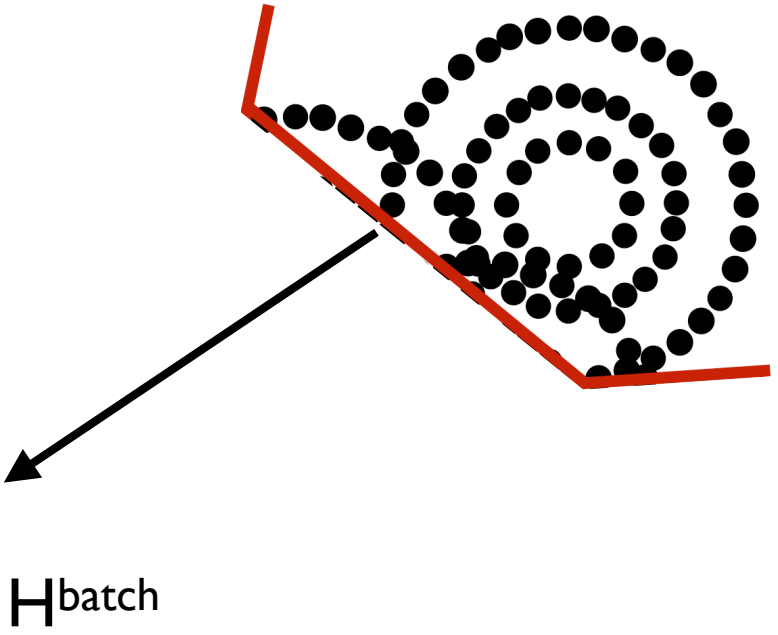


Becke, J. Phys. Chem 1988; Delley, J.Phys. Chem.1990; Havu, J Comp. Phys 2009

Divide-And-Conquer on the Real-Space Matrix



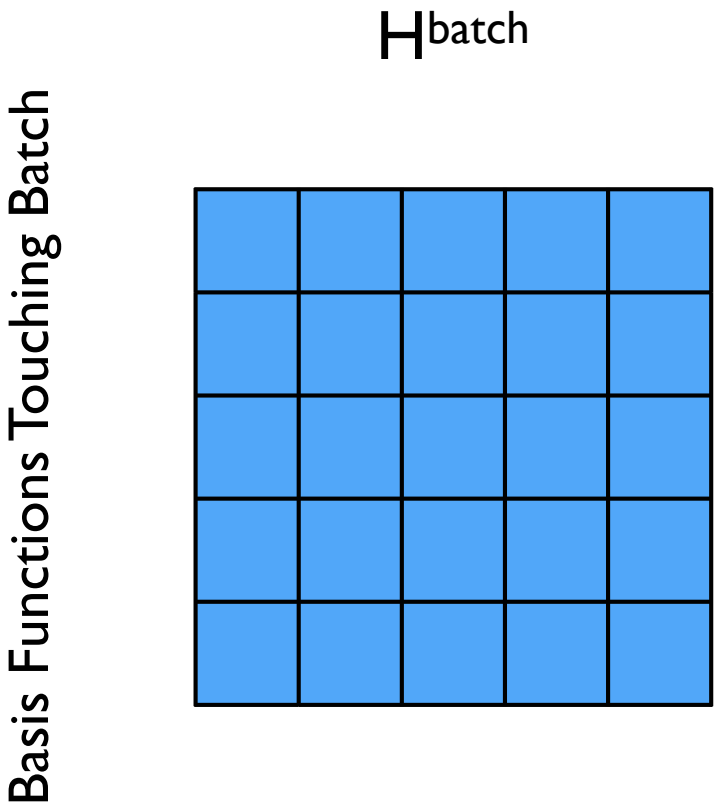
Each batch only contributes to a small number of real-space matrix elements…



H^{batch}

Becke, J. Phys. Chem 1988; Delley, J.Phys. Chem.1990; Havu, J Comp. Phys 2009

Divide-And-Conquer on the Real-Space Matrix

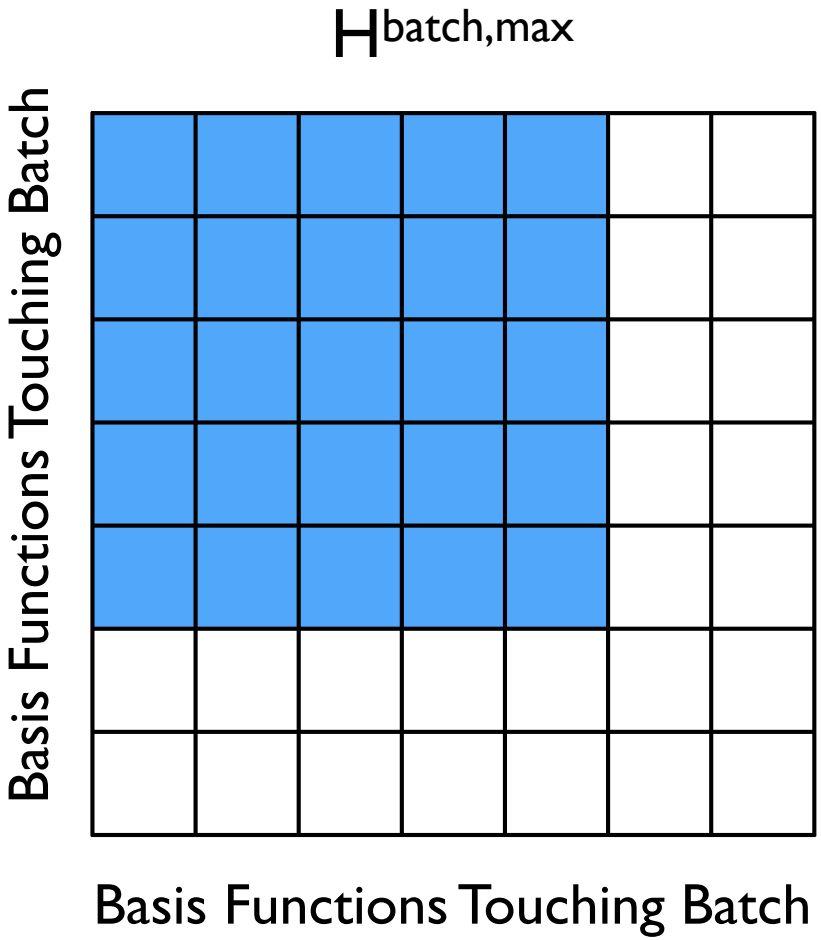


Each batch only contributes to a small number of real-space matrix elements…

… so for each batch, we only calculate a reduced H^{batch} on all basis elements touching that batch

Becke, J. Phys. Chem 1988; Delley, J.Phys. Chem.1990; Havu, J Comp. Phys 2009

Divide-And-Conquer on the Real-Space Matrix



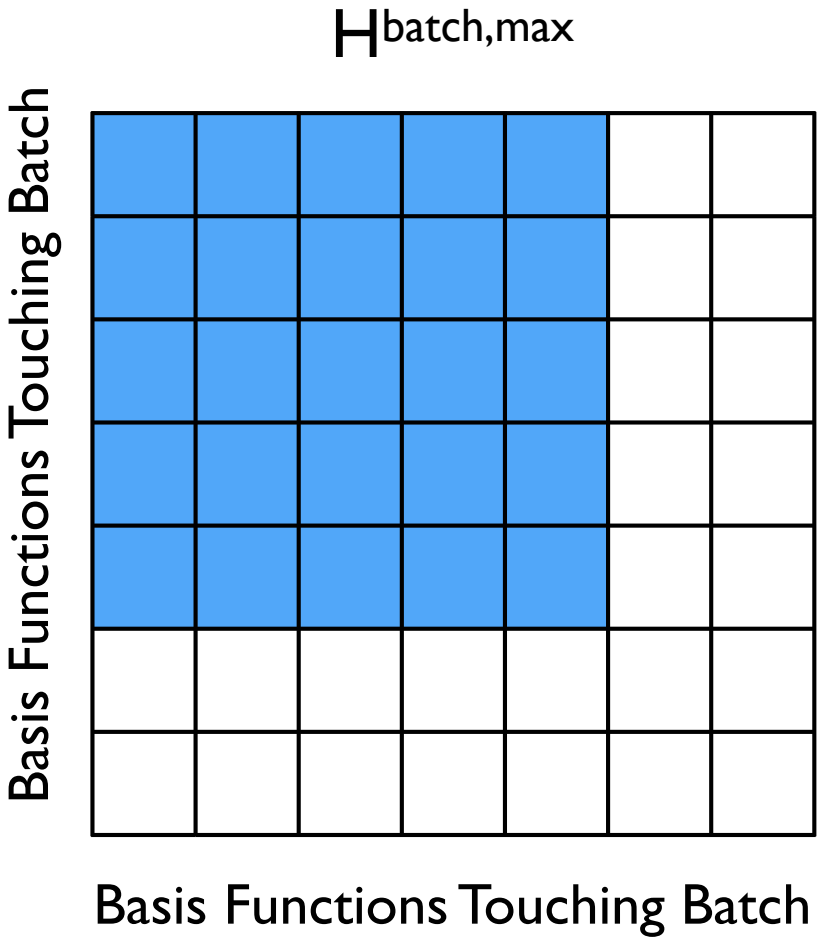
Each batch only contributes to a small number of real-space matrix elements…

… so for each batch, we only calculate a reduced H^{batch} on all basis elements touching that batch

Since basis elements are localized, there is an maximum limit on the size of H^{batch}!

Becke, J. Phys. Chem 1988; Delley, J.Phys. Chem.1990; Havu, J Comp. Phys 2009

Divide-And-Conquer on the Real-Space Matrix



Each batch only contributes to a small number of real-space matrix elements…

… so for each batch, we only calculate a reduced H^{batch} on all basis elements touching that batch

Since basis elements are localized, there is an maximum limit on the size of H^{batch}!

Fixed max work done per batch * $O(N_{atom})$ number of batches = **$O(N_{atom})$ overall work**

Becke, J. Phys. Chem 1988; Delley, J.Phys. Chem.1990; Havu, J Comp. Phys 2009

Example Real-Space Operation: Hamiltonian Integration

Hamiltonian Matrix Calculation

$$H_{mn} = \int \varphi_m^*(\boldsymbol{r}) \hat{h} \varphi_n(\boldsymbol{r}) d\boldsymbol{r}$$

CPU-Only

```
do i_batch = 1, number_of_batches
  calculate_w( i_batch )
  calculate_psi( i_batch )
  calculate_H_times_psi( i_batch )


  psi(:) = w . psi(:)
  DSYR2K( H_times_psi, psi, H_batch )

  index_H( H_batch, hamiltonian )
end do
```

Example Real-Space Operation: Hamiltonian Integration

Hamiltonian Matrix Calculation

$$H_{mn} = \int \varphi_m^*(\boldsymbol{r}) \hat{h} \varphi_n(\boldsymbol{r}) d\boldsymbol{r}$$

CPU-Only

```
do i_batch = 1, number_of_batches
  calculate_w( i_batch )
  calculate_psi( i_batch )
  calculate_H_times_psi( i_batch )


  psi(:) = w . psi(:)
  DSYR2K( H_times_psi, psi, H_batch )

  index_H( H_batch, hamiltonian )
end do
```

Setting up batch quantities
(tedious but cheap on CPU)

Example Real-Space Operation: Hamiltonian Integration

Hamiltonian Matrix Calculation

$$H_{mn} = \int \varphi_m^*(\boldsymbol{r}) \hat{h} \varphi_n(\boldsymbol{r}) d\boldsymbol{r}$$

CPU-Only

```
do i_batch = 1, number_of_batches
  calculate_w( i_batch )
  calculate_psi( i_batch )
  calculate_H_times_psi( i_batch )


  psi(:) = w . psi(:)
  DSYR2K( H_times_psi, psi, H_batch )

  index_H( H_batch, hamiltonian )
end do
```

Setting up batch quantities
(tedious but cheap on CPU)

BLAS3 calls and vectorized ops
(computationally expensive part)

Example Real-Space Operation: Hamiltonian Integration

Hamiltonian Matrix Calculation

$$H_{mn} = \int \varphi_m^*(\boldsymbol{r}) \hat{h} \varphi_n(\boldsymbol{r}) d\boldsymbol{r}$$

CPU-Only

```
do i_batch = 1, number_of_batches
  calculate_w( i_batch )
  calculate_psi( i_batch )
  calculate_H_times_psi( i_batch )


  psi(:) = w . psi(:)
  DSYR2K( H_times_psi, psi, H_batch )

  index_H( H_batch, hamiltonian )
end do
```

Setting up batch quantities (tedious but cheap on CPU)

BLAS3 calls and vectorized ops (computationally expensive part)

Indexing from batch matrix back to accumulated matrix

Example Real-Space Operation: Hamiltonian Integration

Hamiltonian Matrix Calculation

$$H_{mn} = \int \varphi_m^*(\boldsymbol{r}) \hat{h} \varphi_n(\boldsymbol{r}) d\boldsymbol{r}$$

CPU-Only

```
do i_batch = 1, number_of_batches
  calculate_w( i_batch )
  calculate_psi( i_batch )
  calculate_H_times_psi( i_batch )


  psi(:) = w . psi(:)
  DSYR2K( H_times_psi, psi, H_batch )

  index_H( H_batch, hamiltonian )
end do
```

GPGPU-Accelerated

```
do i_batch = 1, number_of_batches
  calculate_w( i_batch )
  calculate_psi( i_batch )
  calculate_H_times_psi( i_batch )

  copy_cpu_data_to_gpu( w, psi, H_times_psi )
  cublasDDGMM( psi, w, psi )
  cublasDSYR2K( H_times_psi, psi, H_batch )

  index_H_on_gpu( H_batch, hamiltonian )
end do

copy_gpu_data_to_cpu( hamiltonian )
```

Example Real-Space Operation: Hamiltonian Integration

Hamiltonian Matrix Calculation

$$H_{mn} = \int \varphi_m^*(\boldsymbol{r}) \hat{h} \varphi_n(\boldsymbol{r}) d\boldsymbol{r}$$

CPU-Only

```
do i_batch = 1, number_of_batches
  calculate_w( i_batch )
  calculate_psi( i_batch )
  calculate_H_times_psi( i_batch )

  psi(:) = w . psi(:)
  DSYR2K( H_times_psi, psi, H_batch )

  index_H( H_batch, hamiltonian )
end do
```

GPGPU-Accelerated

```
do i_batch = 1, number_of_batches
  calculate_w( i_batch )
  calculate_psi( i_batch )
  calculate_H_times_psi( i_batch )

  copy_cpu_data_to_gpu( w, psi, H_times_psi )
  cublasDDGMM( psi, w, psi )
  cublasDSYR2K( H_times_psi, psi, H_batch )

  index_H_on_gpu( H_batch, hamiltonian )
end do

copy_gpu_data_to_cpu( hamiltonian )
```

BLAS3 calls become cuBLAS calls

Compute-time-intensive portion

Example Real-Space Operation: Hamiltonian Integration

Hamiltonian Matrix Calculation

$$H_{mn} = \int \varphi_m^*(\boldsymbol{r}) \hat{h} \varphi_n(\boldsymbol{r}) d\boldsymbol{r}$$

CPU-Only

```
do i_batch = 1, number_of_batches
  calculate_w( i_batch )
  calculate_psi( i_batch )
  calculate_H_times_psi( i_batch )

  psi(:) = w . psi(:)
  DSYR2K( H_times_psi, psi, H_batch )

  index_H( H_batch, hamiltonian )
end do
```

GPGPU-Accelerated

```
do i_batch = 1, number_of_batches
  calculate_w( i_batch )
  calculate_psi( i_batch )
  calculate_H_times_psi( i_batch )

  copy_cpu_data_to_gpu( w, psi, H_times_psi )
  cublasDDGMM( psi, w, psi )
  cublasDSYR2K( H_times_psi, psi, H_batch )

  index_H_on_gpu( H_batch, hamiltonian )
end do

copy_gpu_data_to_cpu( hamiltonian )
```

BLAS3 calls become cuBLAS calls

Compute-time-intensive portion

GPGPU handles indexing of matrix…
allowing CPU to prepare next batch
GPGPU-to-CPU transfer occurs once
Programmer-time-intensive portion

Systems Used for Benchmarking

Two clusters used:

- timewarp: Development cluster of aims group at Duke University
- PSG: Benchmarking cluster for Tesla GPGPUs at NVIDIA

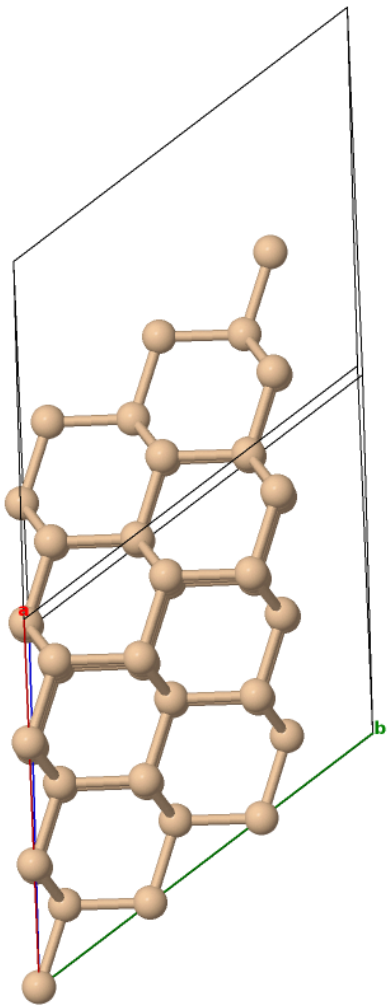
Three types of nodes used:

Node	Ivy Bridge/GP100	Haswell/P100	Skylake/V100
Cluster	timewarp	PSG	PSG
CPU	2x Intel Xeon E5-2670v2 (20 cores*, Ivy Bridge)	2x Intel Xeon E5-2698v3 (32 cores, Haswell)	2x Intel Xeon Gold 6148 (20 cores, Skylake)
GPGPU**	1x Quadro GP100 (Pascal)	4x Tesla P100 (Pascal)	4x Tesla V100 (Volta)
MPI Tasks/GPGPUs	16/1	32/4	20/4
Compilers/Libraries	ifort 14.0, MKL 11.1.1, IMPI 4.1.3, CUDA 8.0	ifort 17.0, MKL 11.3.3, IMPI 5.0.3, CUDA 9.1	ifort 17.0, MKL 11.3.3, IMPI 5.0.3, CUDA 9.1

* Due to usage of MPS, only 16 MPI tasks used for Ivy Bridge/GP100

** All GPGPUs are PCI-E models

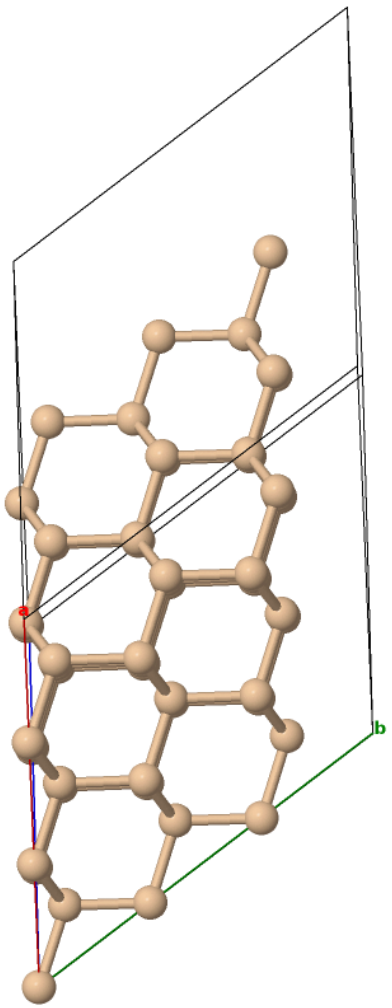
A Complete All-Electron Calculation: Si (3x3x3)



Diamond Si
3x3x3 supercell
(54 atoms)

We perform a complete all-electron calculation including forces and analytical stress tensor for a 3x3x3 supercell of Si

A Complete All-Electron Calculation: Si (3x3x3)



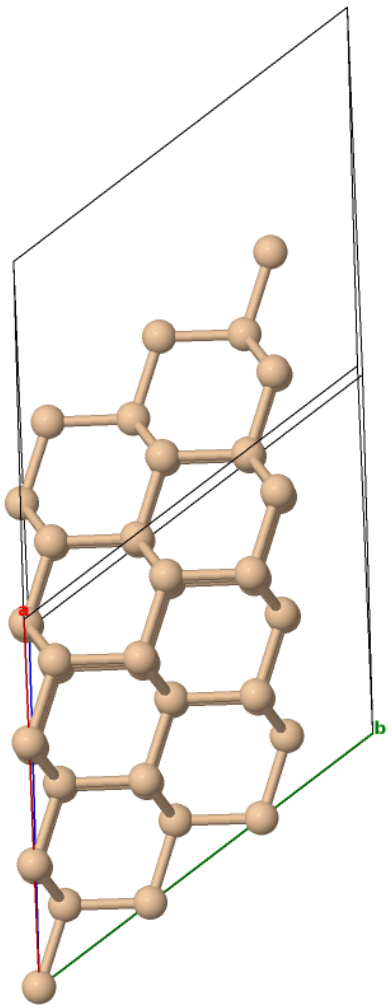
Diamond Si
3x3x3 supercell
(54 atoms)

We perform a complete all-electron calculation including forces and analytical stress tensor for a 3x3x3 supercell of Si

This calculation is representative of a single iteration of a geometric optimization calculation, containing

- Initialization (not GPGPU accelerated)
- 12 "Normal" SCF Iterations
- 1 SCF + Forces + Analytical Stress Tensor Iteration

A Complete All-Electron Calculation: Si (3x3x3)



Diamond Si
3x3x3 supercell
(54 atoms)

We perform a complete all-electron calculation including forces and analytical stress tensor for a 3x3x3 supercell of Si

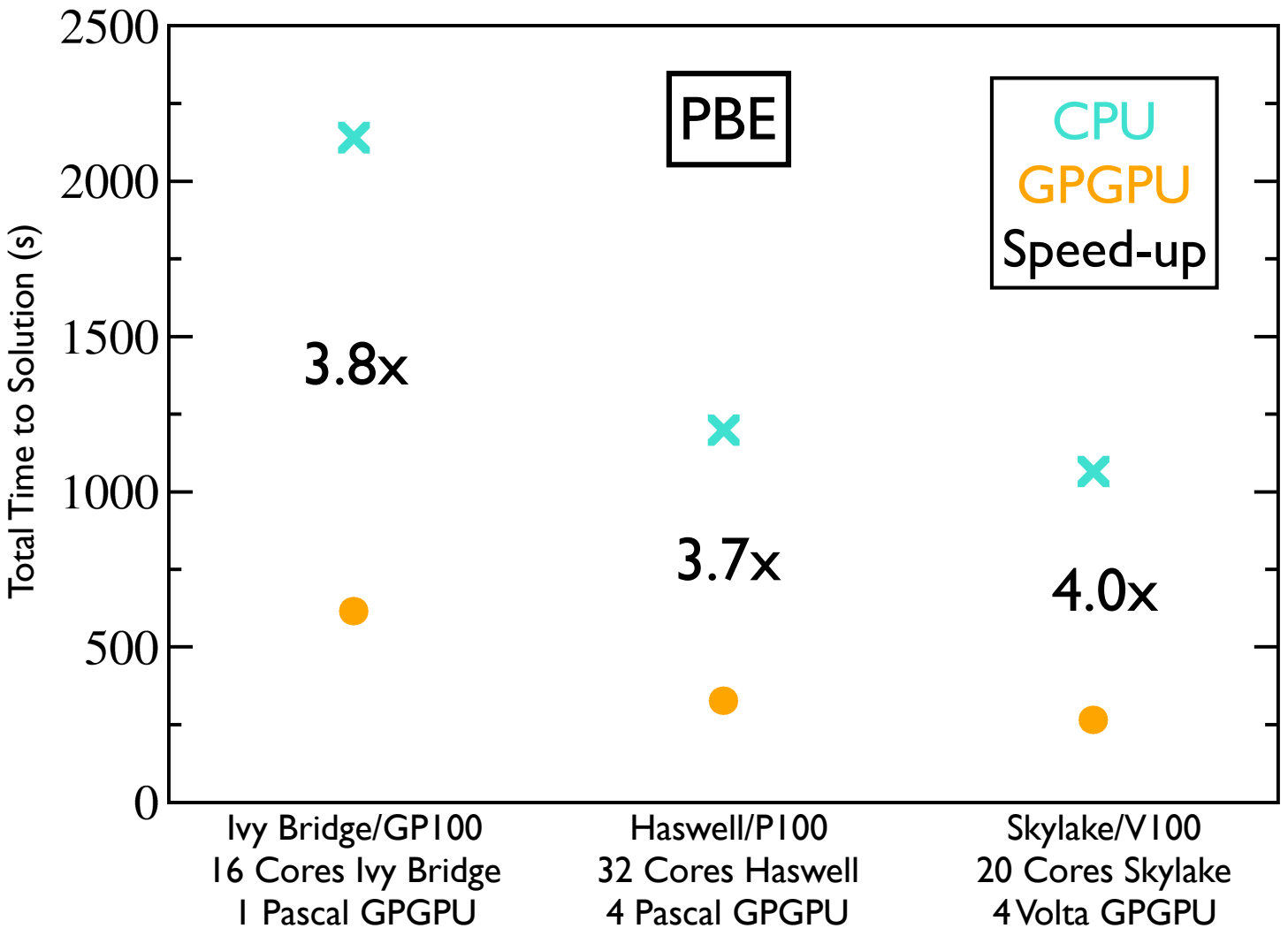
This calculation is representative of a single iteration of a geometric optimization calculation, containing

- Initialization (not GPGPU accelerated)
- 12 "Normal" SCF Iterations
- 1 SCF + Forces + Analytical Stress Tensor Iteration

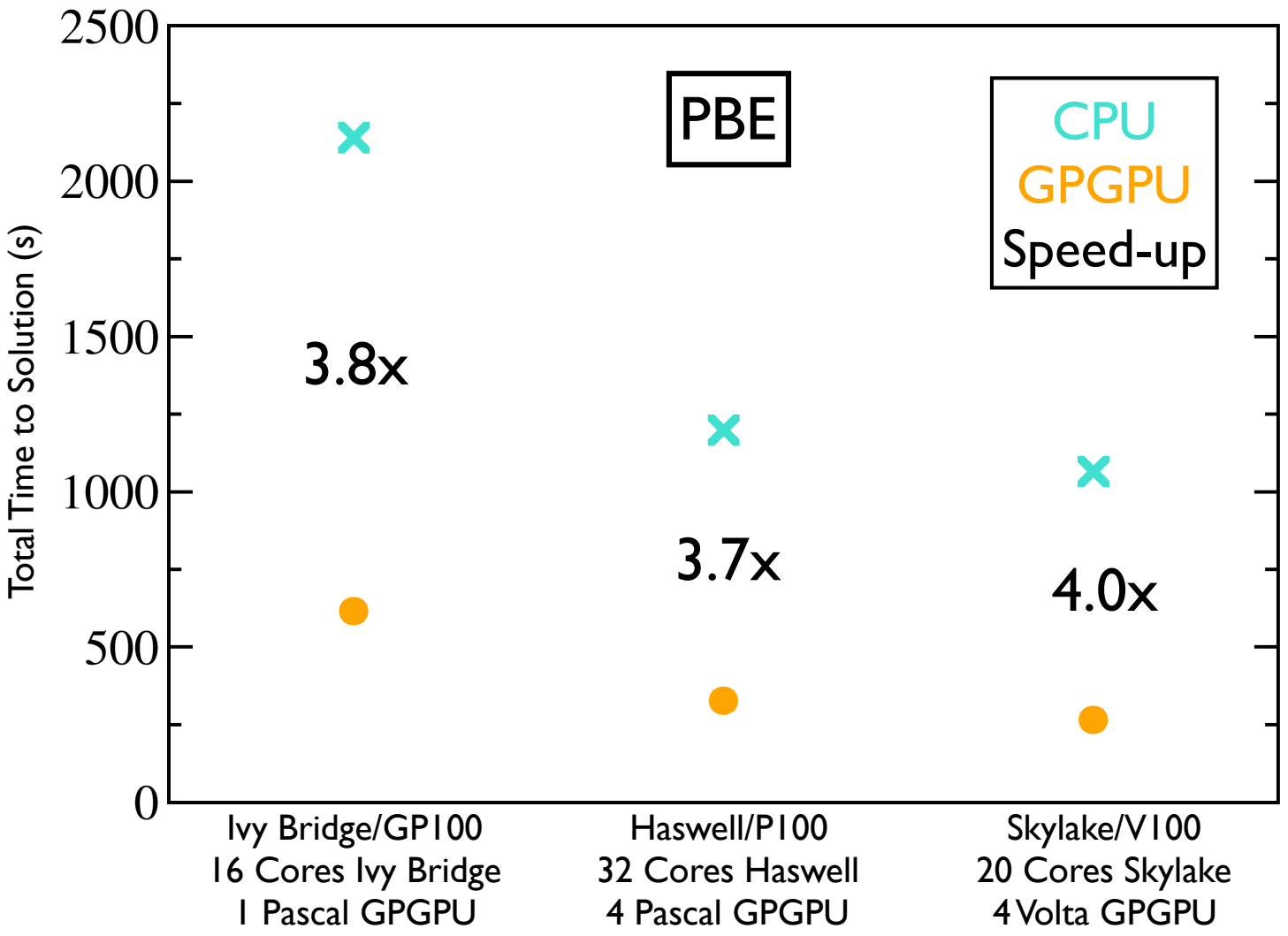
Computational Details:

- PBE functional
- Tight integration settings
- Tight basis sets
- SCF k-grid: 1x1x1
- Load balancing for GPGPU, CSR for CPU

Total Time for **Entire Calculation**: Si (3x3x3)

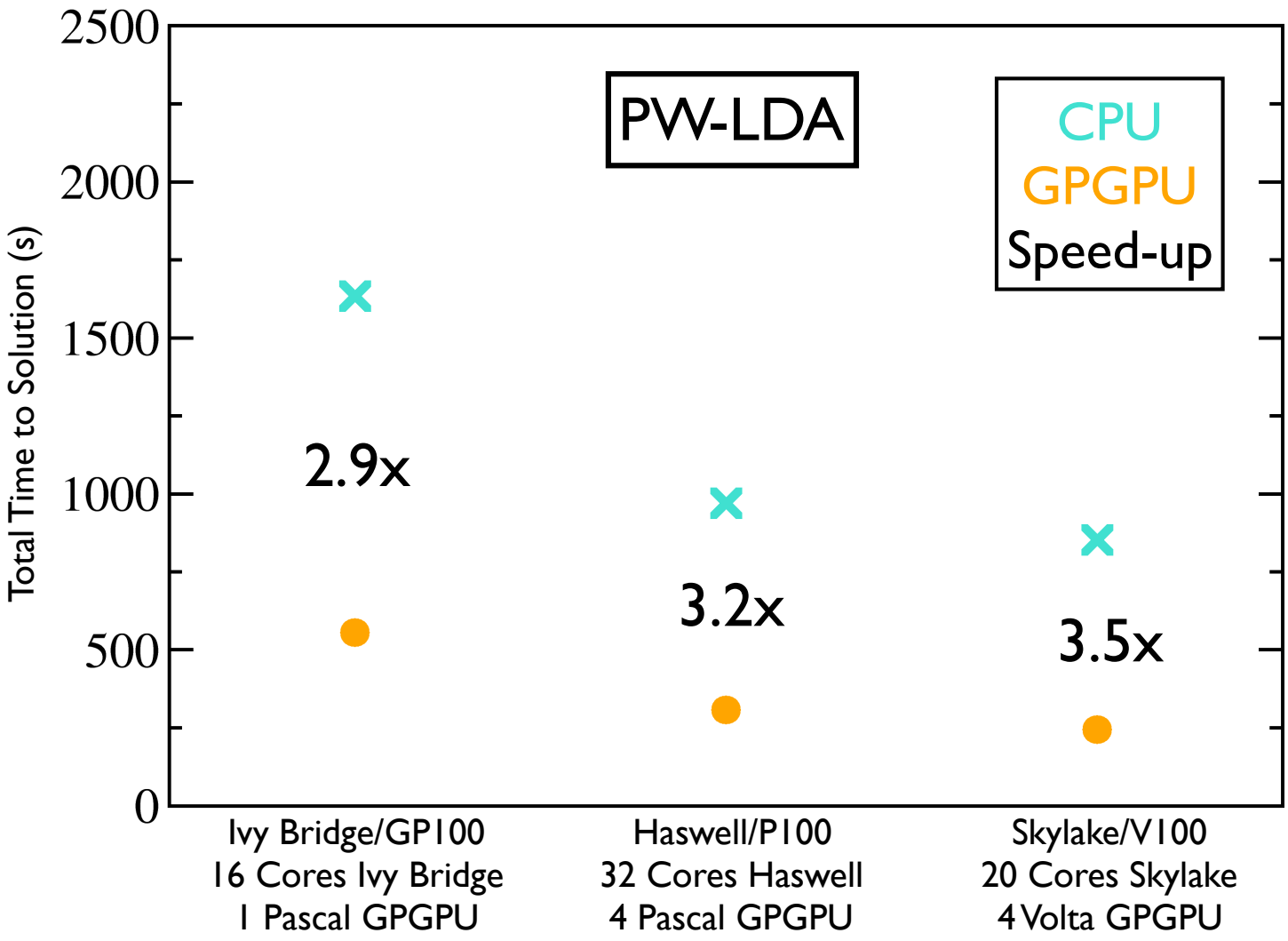


Total Time for **Entire Calculation**: Si (3x3x3)

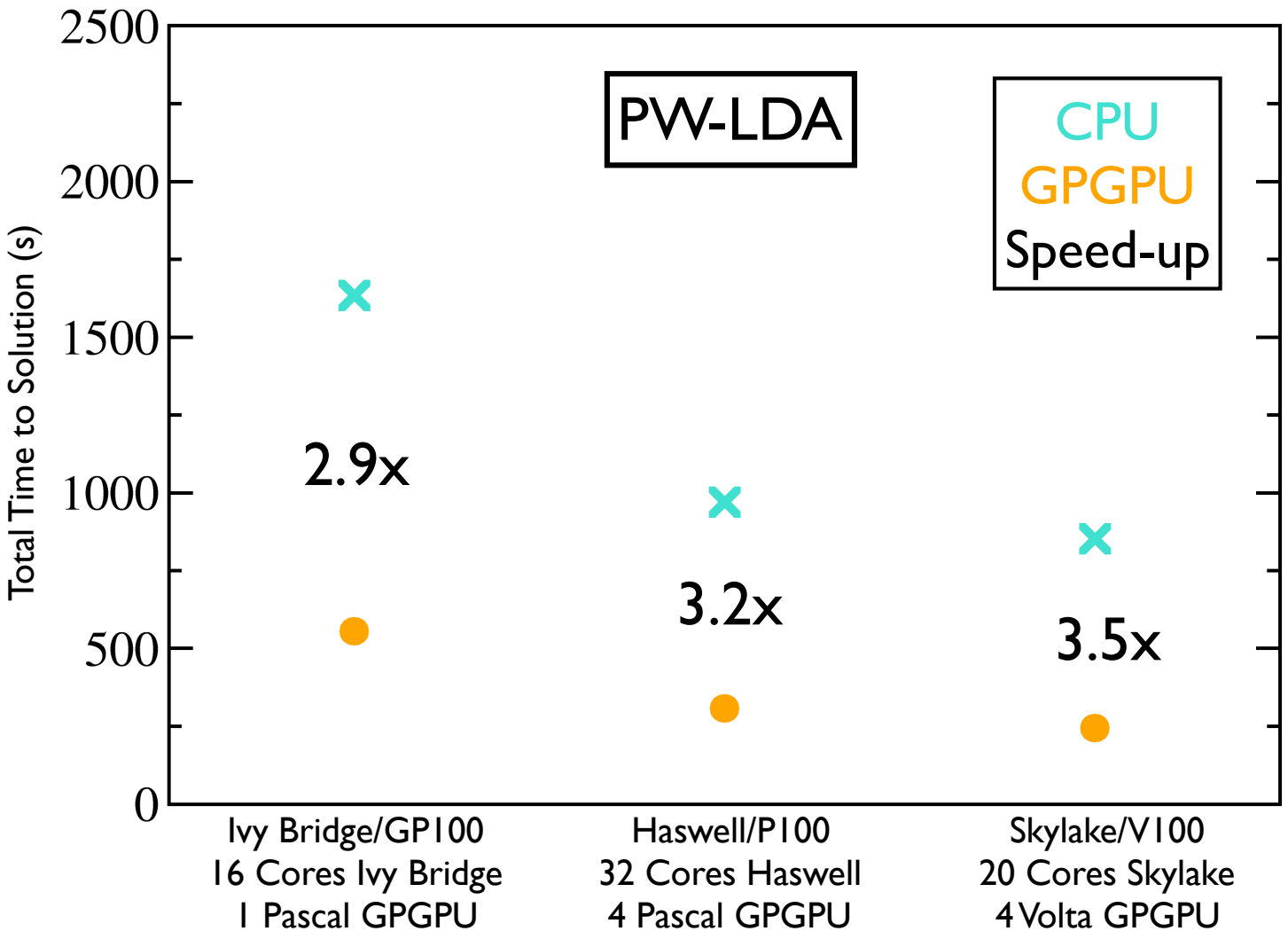


Do these results extend to other functionals?

Total Time for **Entire Calculation**: Si (3x3x3)



Total Time for **Entire Calculation**: Si (3x3x3)



Yes, but less speedup for LDA due to less linear algebra

Back to PBE…

Speedups for Diamond Si (3x3x3)

	Ivy Bridge/GP100	Haswell/P100	Skylake/V100
SCF Iteration	2.9x	2.5x	3.1x
SCF Iteration + Forces	4.4x	5.8x	5.7x
SCF Iteration + Forces + AS	5.2x	6.7x	7.8x
Entire Calculation	**3.8x**	**3.7x**	**4.0x**

Back to PBE…

Speedups for Diamond Si (3x3x3)

	Ivy Bridge/GP100	Haswell/P100	Skylake/V100
SCF Iteration	2.9x	2.5x	3.1x
SCF Iteration + Forces	4.4x	5.8x	5.7x
SCF Iteration + Forces + AS	5.2x	6.7x	7.8x
Entire Calculation	**3.8x**	**3.7x**	**4.0x**

Do these results generalize to other materials?

103 Compound Material Benchmark Set

To systematically verify the performance of the GPGPU code, we use the 103 compound material test set from Huhn and Blum:

PHYSICAL REVIEW MATERIALS 1, 033803 (2017)

One-hundred-three compound band-structure benchmark of post-self-consistent spin-orbit coupling treatments in density functional theory

William P. Huhn* and Volker Blum†
Department of Mechanical Engineering and Materials Science, Duke University, Durham, North Carolina 27708, USA
(Received 4 May 2017; published 30 August 2017)

Family	No. materials	Materials
Elementals	45	Be, C [GRA], Ne, Mg, Al, Si, Ca, Sc, Ti, V, Cr, Mn, Fe, Co, Ni, Cu, Zn, Ge, Sr, Y, Zr, Nb, Mo, Tc, Ru, Rh, Pd, Ag, Cd, Sn, Xe, Ba, Lu, Hf, Ta, W, Re, Os, Ir, Pt, Au, Tl, Pb, Bi, Po
Compound semiconductors	37	C [DIA], MgO, AlN [WUR], AlN [ZB], SiC, BP, AlP, MgS, ZnO, ZnS [WUR], ZnS [ZB], GaN [WUR], GaN [ZB], GaP, AlAs, BAs, GaAs, MgSe, ZnSe, CdS [WUR], CdS [ZB], CdSe [WUR], CdSe [ZB], InN, InP, InAs, AlSb, GaSb, InSb, ZnTe, CdTe, HgS, HgSe, HgTe, PbS, PbSe, PbTe
Alkali halides	21	LiF, NaF, LiCl, NaCl, KF, KCl, LiBr, NaBr, KBr, RbF, RbCl, RbBr, LiI, NaI, KI, RbI, CsF, CsCl [CSCL], CsCl [RS], CsBr, CsI

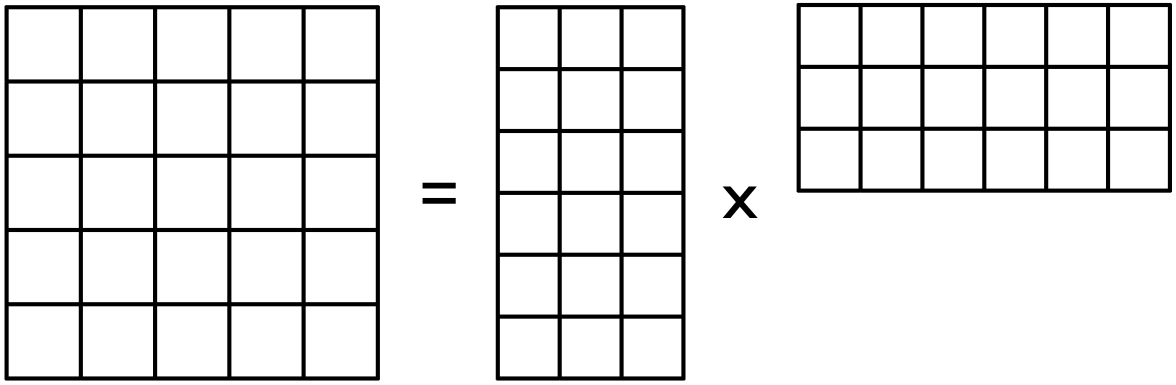
3x3x3 supercells of the primitive cell for a given material was used, yielding unit cell sizes ranging from 27, 54, and 108 atoms

Computational Details:

- PBE functional
- Tight integration settings
- Tight basis sets
- SCF k-grid: 1x1x1
- Load balancing was used (critical!)

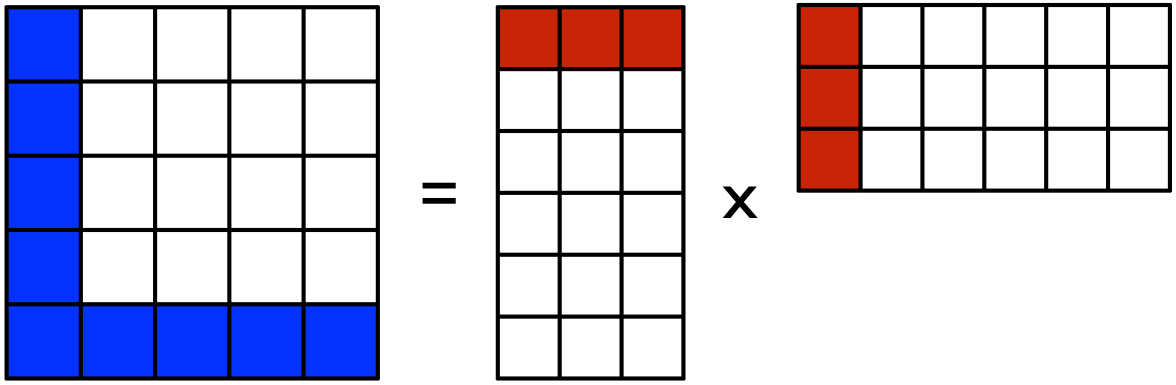
Quantities Used in 103 Compound Benchmark

(Crude Estimate!) of Problem Size



Quantities Used in 103 Compound Benchmark

(Crude Estimate!) of Problem Size



basis fns
x
basis fns

=

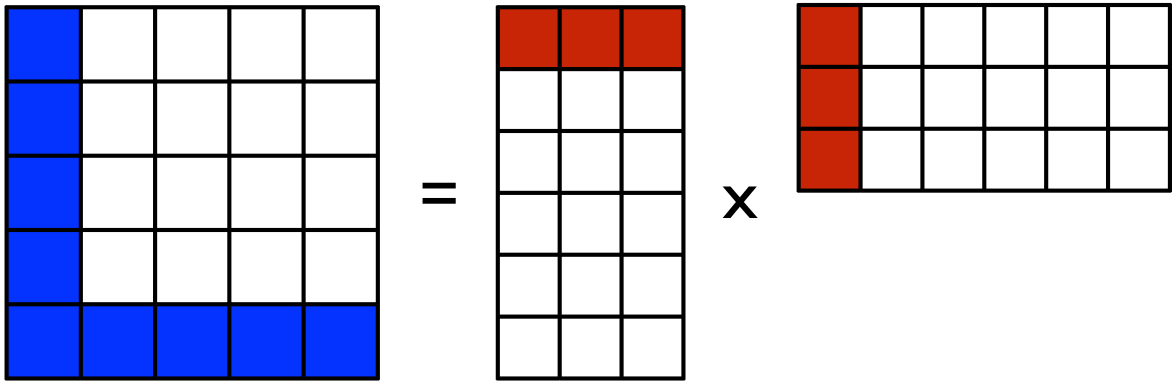
basis fns
x
points

x

points
x
basis fns

Quantities Used in 103 Compound Benchmark

(Crude Estimate!) of Problem Size



basis fns
x
basis fns

=

basis fns
x
points

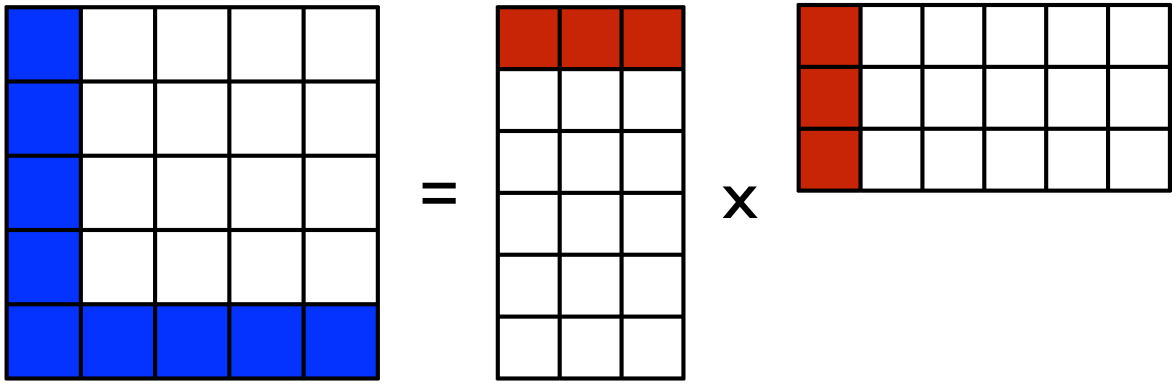
x

points
x
basis fns

DGEMM ≈ (# basis fn)^2 * (# points)

Quantities Used in 103 Compound Benchmark

(Crude Estimate!) of Problem Size



basis fns
x
basis fns

=

basis fns
x
points

x

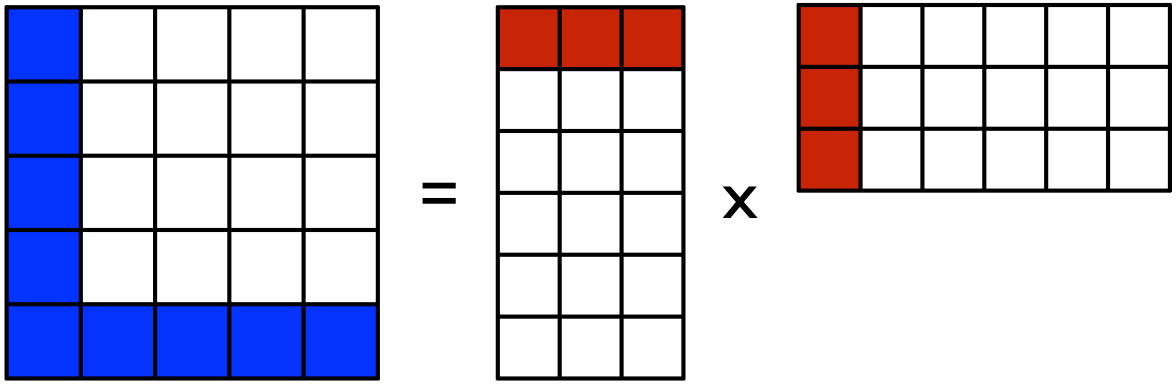
points
x
basis fns

DGEMM ≈ (# basis fn)^2 * (# points)

If we did one DGEMM per batch…

Quantities Used in 103 Compound Benchmark

(Crude Estimate!) of Problem Size



# basis fns x # basis fns	# basis fns x # points	# points x # basis fns

DGEMM ≈ (# basis fn)^2 * (# points)

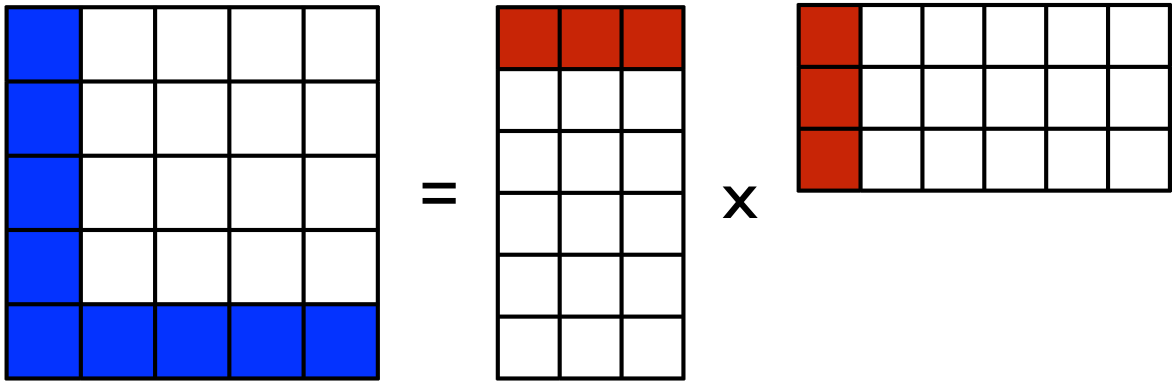
If we did one DGEMM per batch…

ops ≈ # batchs * <# basis fns>^{2} * <# points>

where <…> denotes average over batches

Quantities Used in 103 Compound Benchmark

(Crude Estimate!) of Problem Size



# basis fns x # basis fns	=	# basis fns x # points	x	# points x # basis fns

DGEMM ≈ (# basis fn)^2 * (# points)

If we did one DGEMM per batch…

ops ≈ # batchs * <# basis fns>^{2} * <# points>

where <…> denotes average over batches

Speedup

To compute speedups, we use timings for all calculations:

$$T_{CPU} = \sum_{materials} t_{CPU}$$

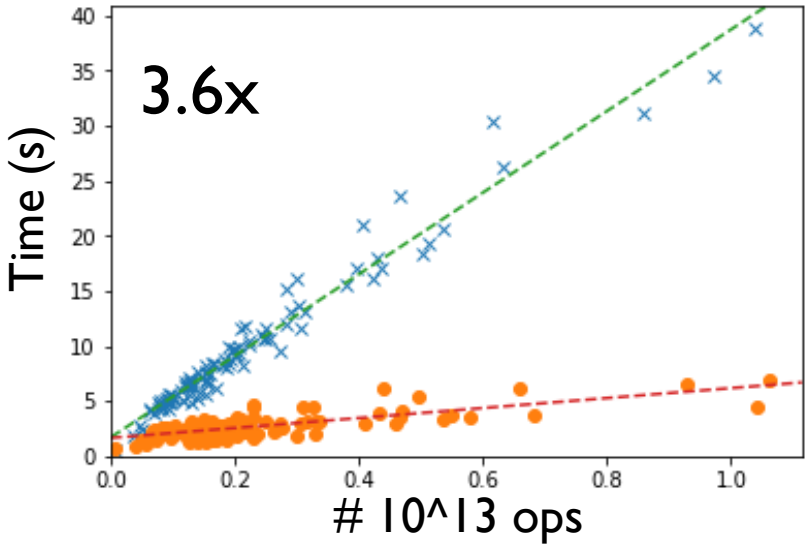
$$T_{GPGPU} = \sum_{materials} t_{GPGPU}$$

$$Speedup = T_{CPU}/T_{GPGPU}$$

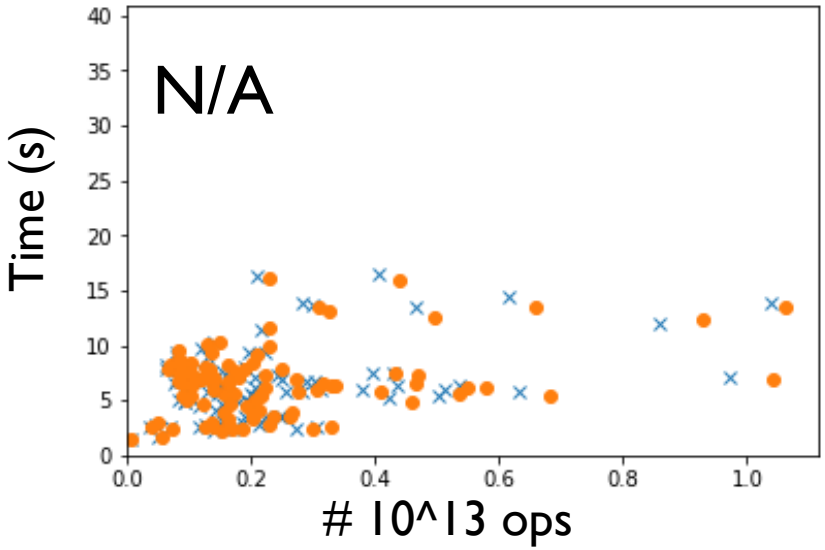
This is commonly reported as #x, where 1.0x corresponds to no speedup

SCF Iteration for 103 Compounds: Skylake/V100

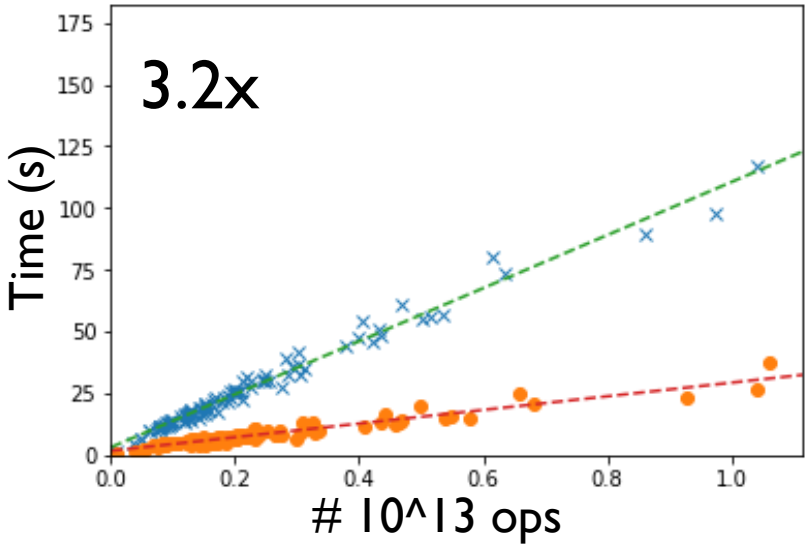
Hamiltonian Integration



Hartree Multipole Summation

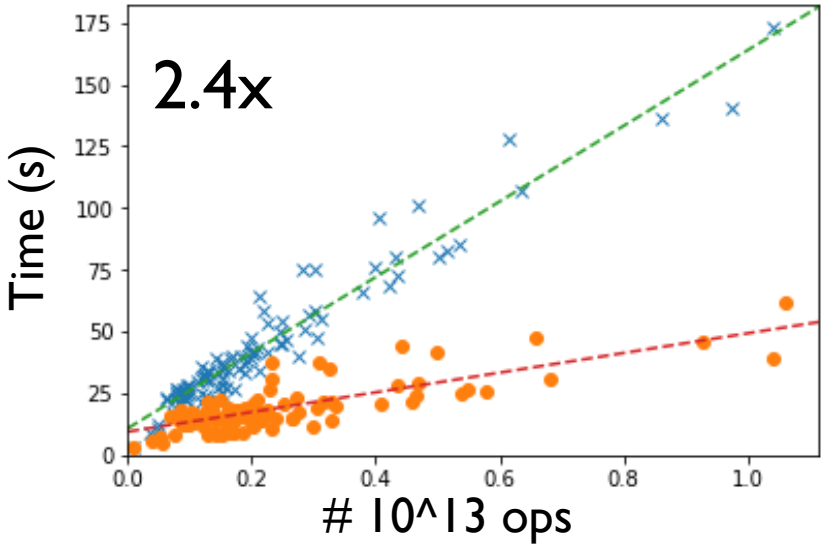


Density



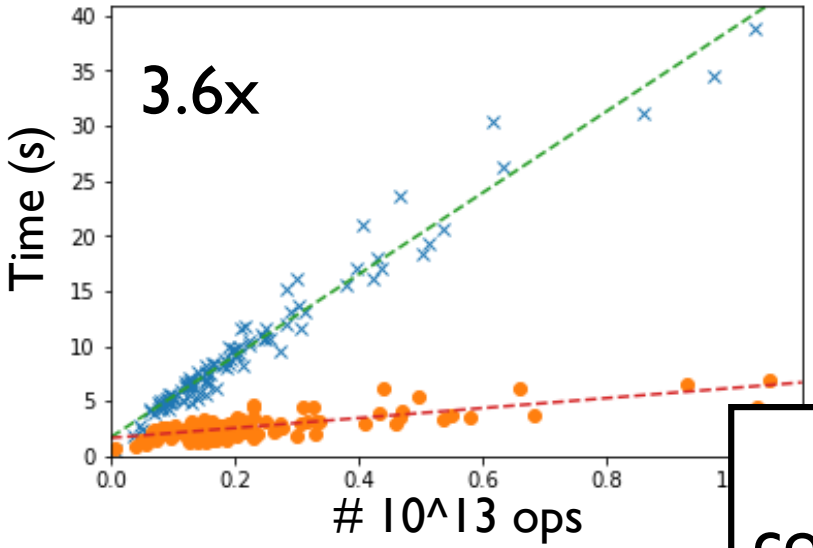
CPU
GPGPU
Speed-up

Total Time for Iteration

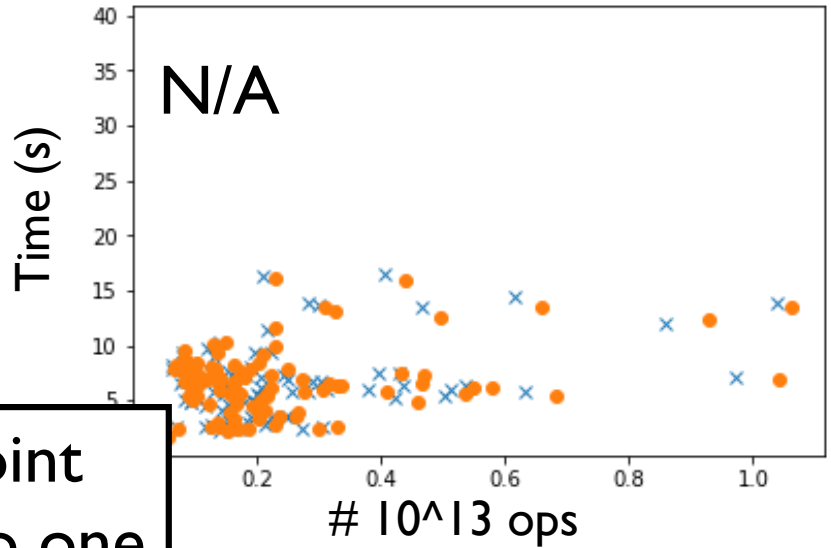


SCF Iteration for 103 Compounds: Skylake/V100

Hamiltonian Integration

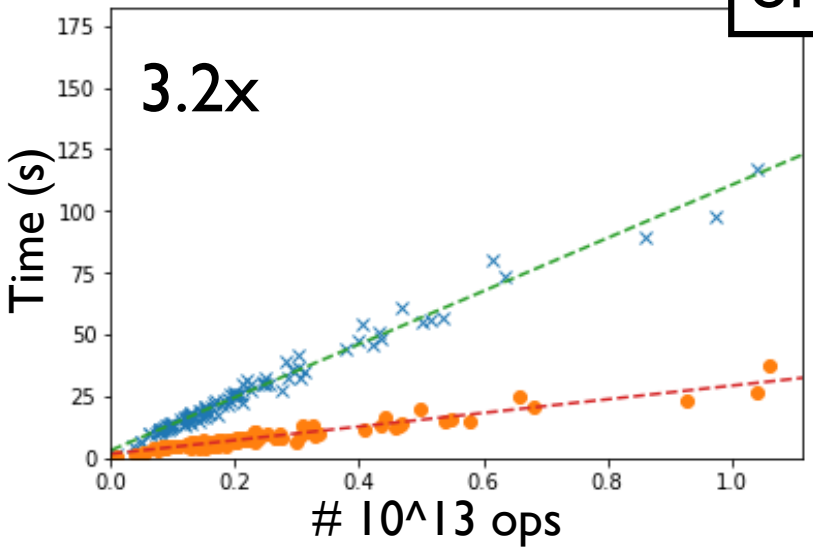


Hartree Multipole Summation

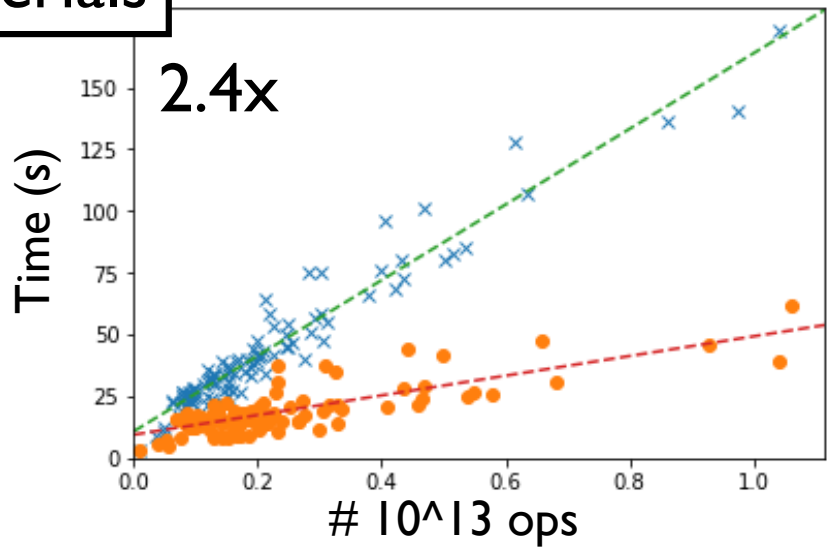


Each data point corresponds to one of the 103 materials

Density



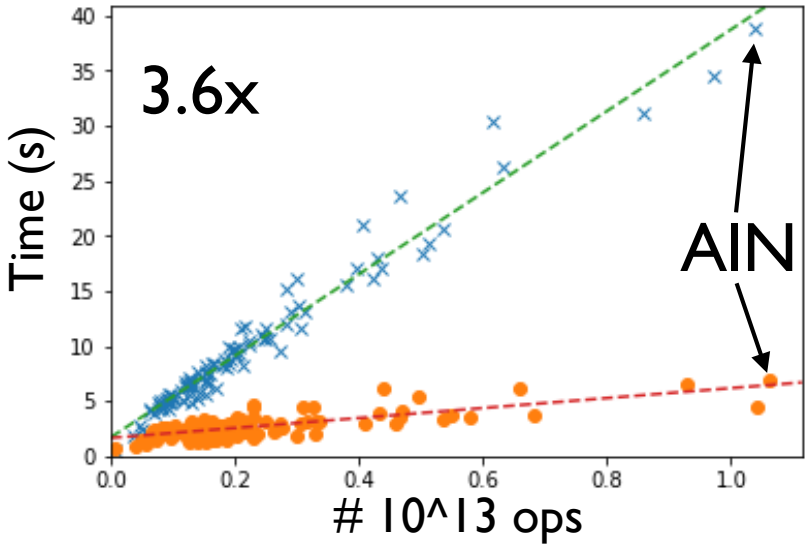
Total Time for Iteration



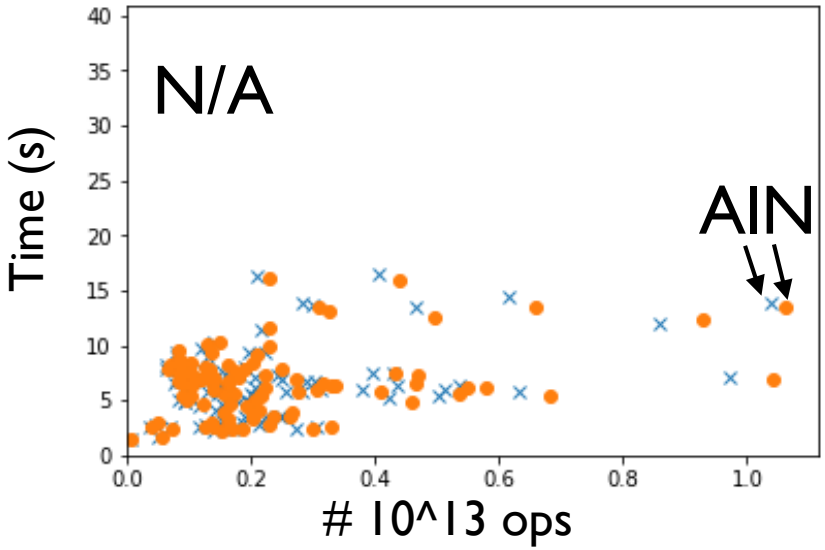
CPU
GPGPU
Speed-up

SCF Iteration for 103 Compounds: Skylake/V100

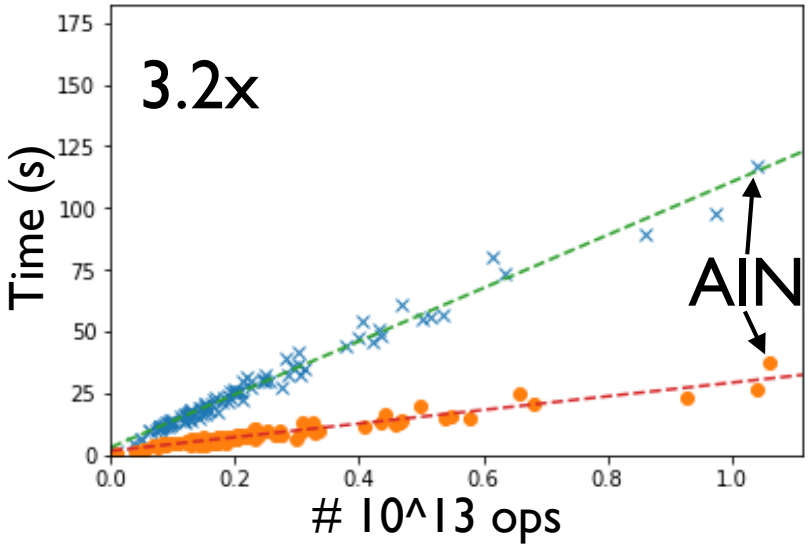
Hamiltonian Integration



Hartree Multipole Summation

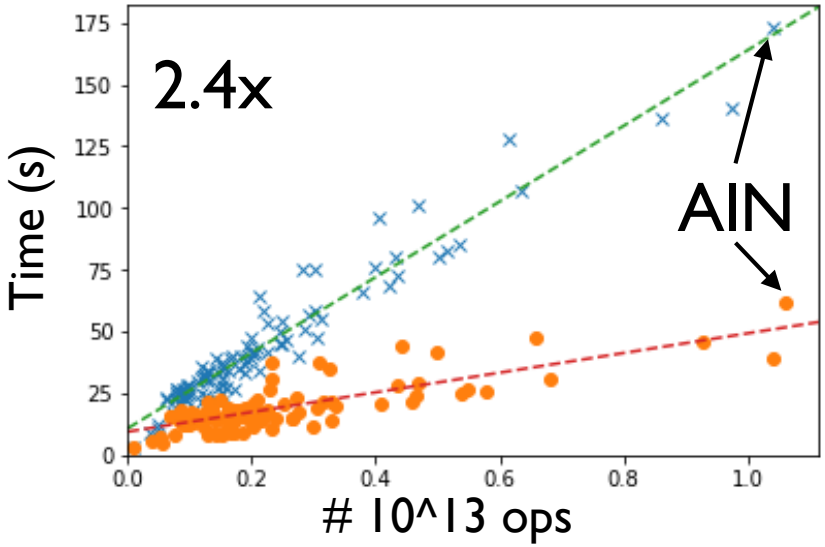


Density



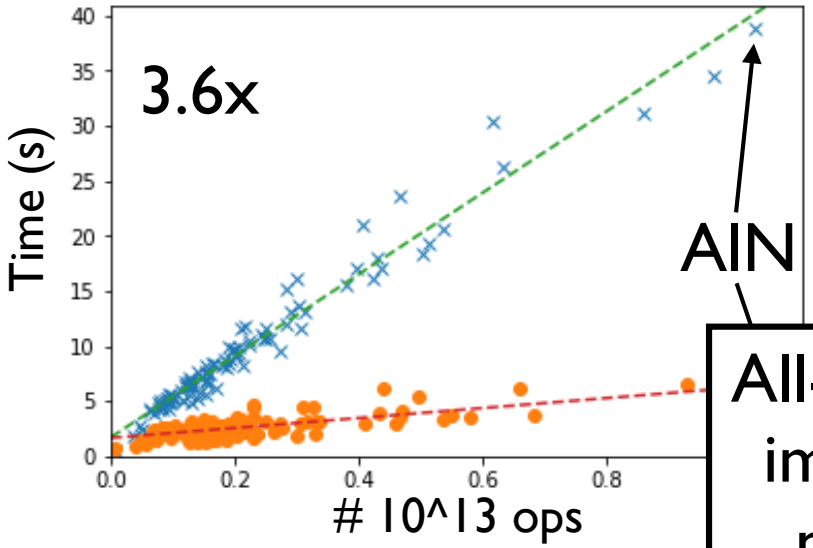
CPU
GPGPU
Speed-up

Total Time for Iteration

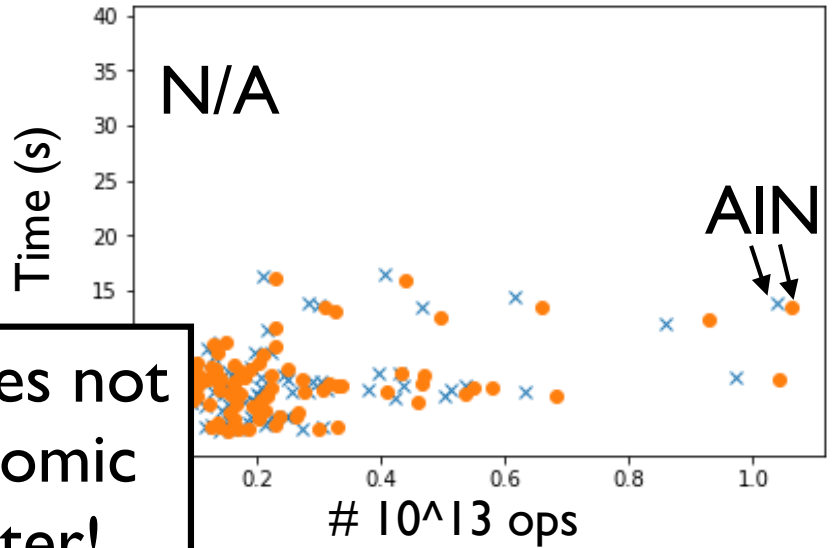


SCF Iteration for 103 Compounds: Skylake/V100

Hamiltonian Integration

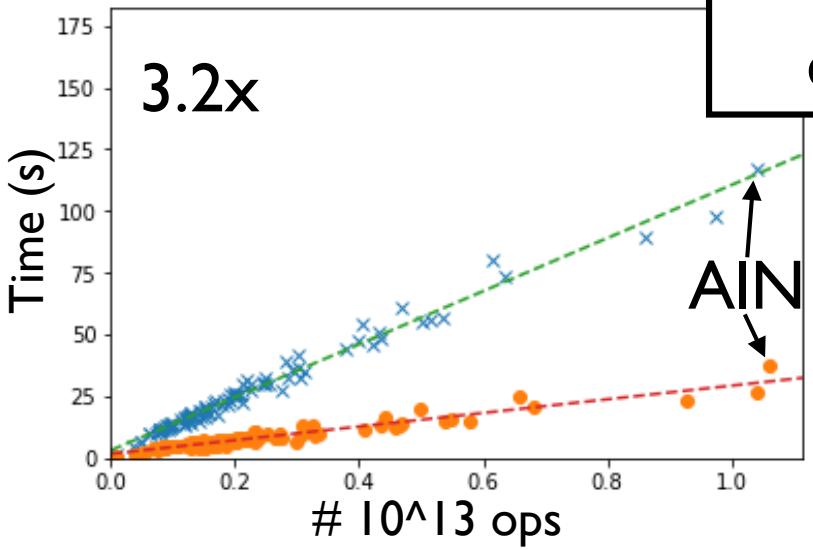


Hartree Multipole Summation

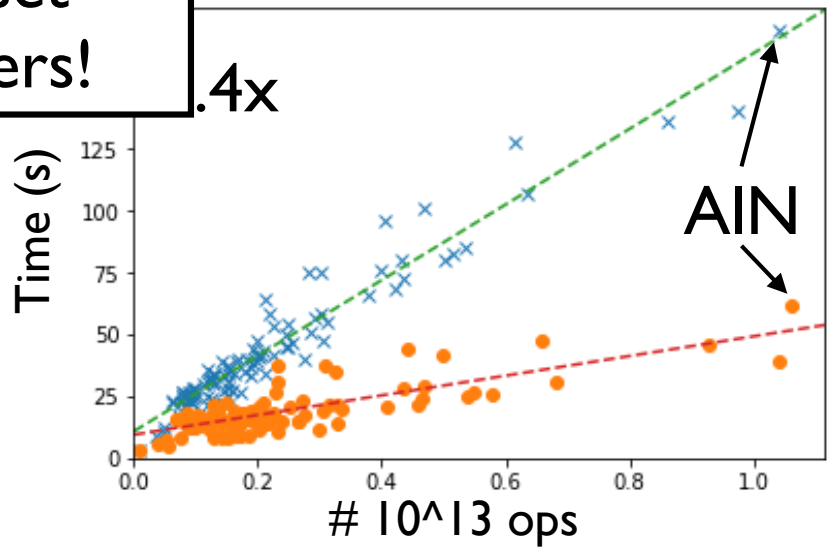


All-electron does not imply lower atomic number = faster! Local basis set density matters!

Density



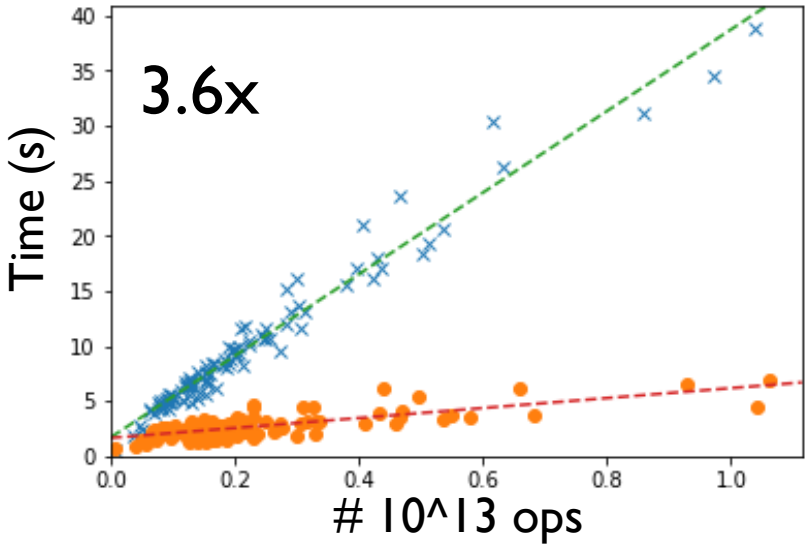
tal Time for Iteration



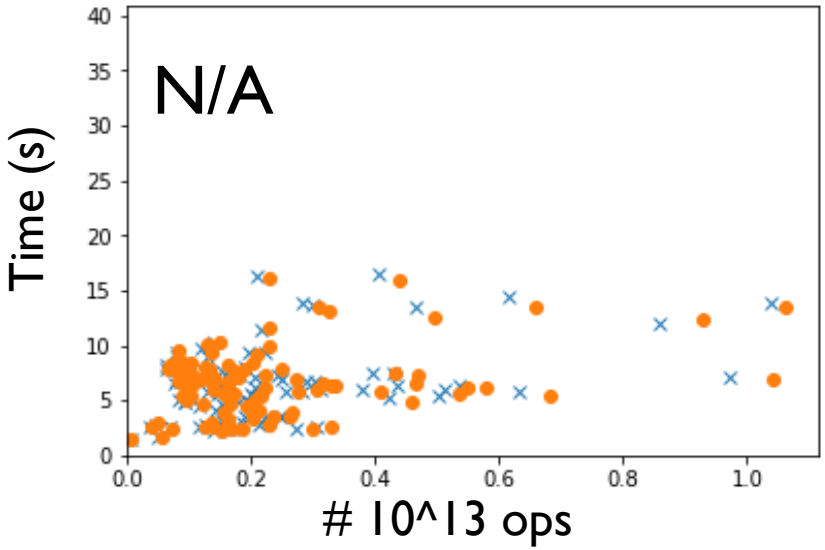
CPU
GPGPU
Speed-up

SCF Iteration for 103 Compounds: Skylake/V100

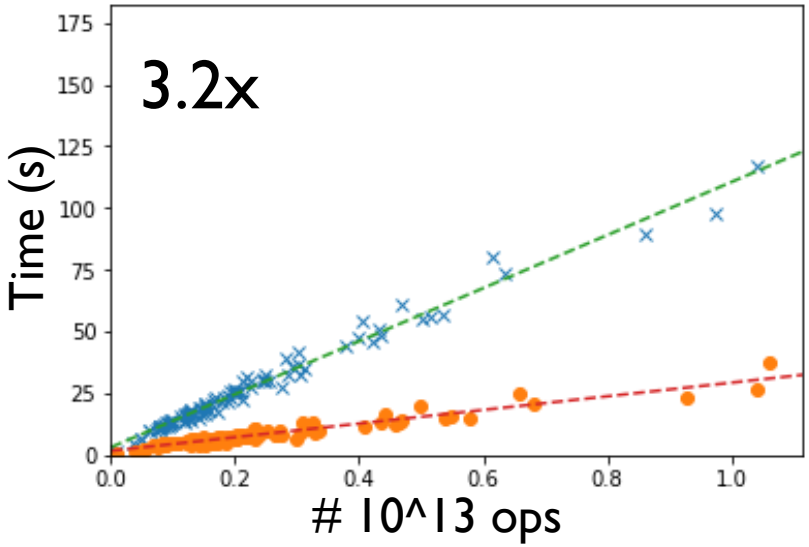
Hamiltonian Integration



Hartree Multipole Summation

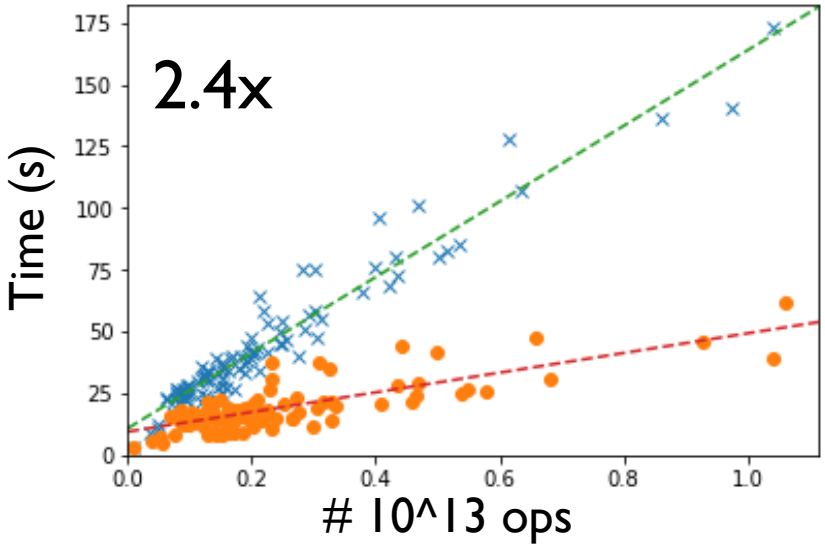


Density



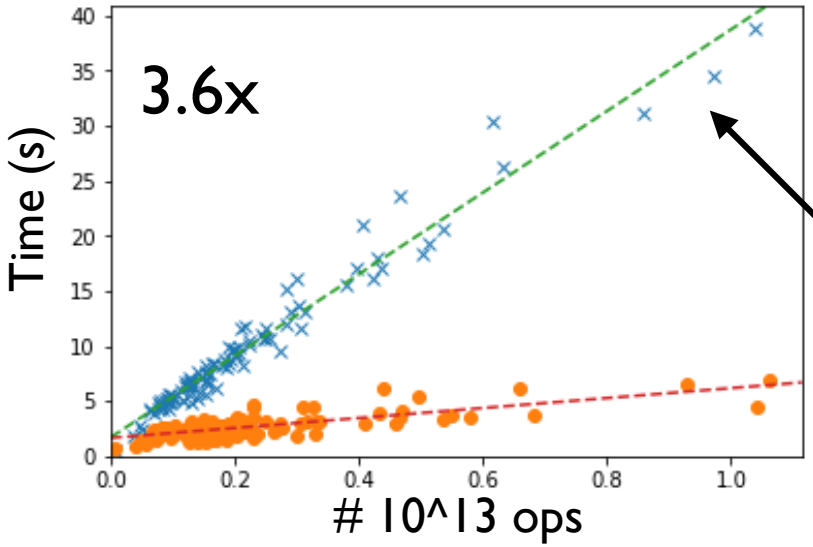
CPU
GPGPU
Speed-up

Total Time for Iteration

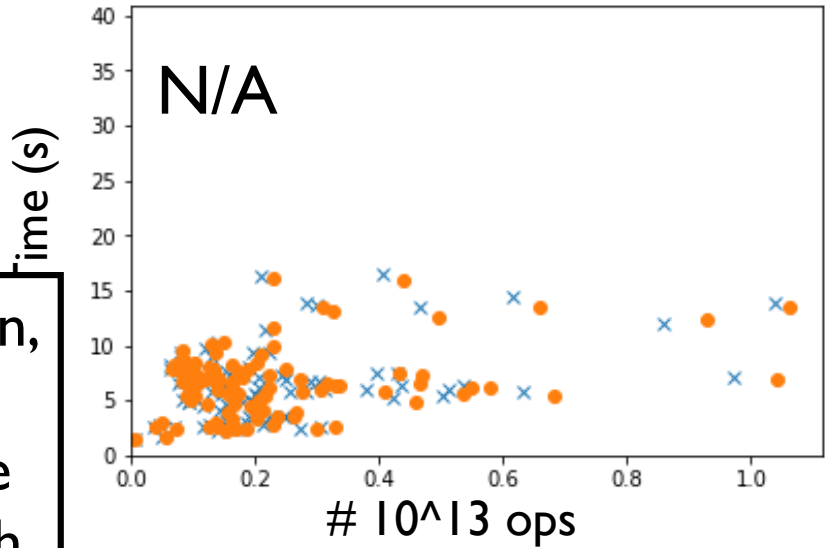


SCF Iteration for 103 Compounds: Skylake/V100

Hamiltonian Integration

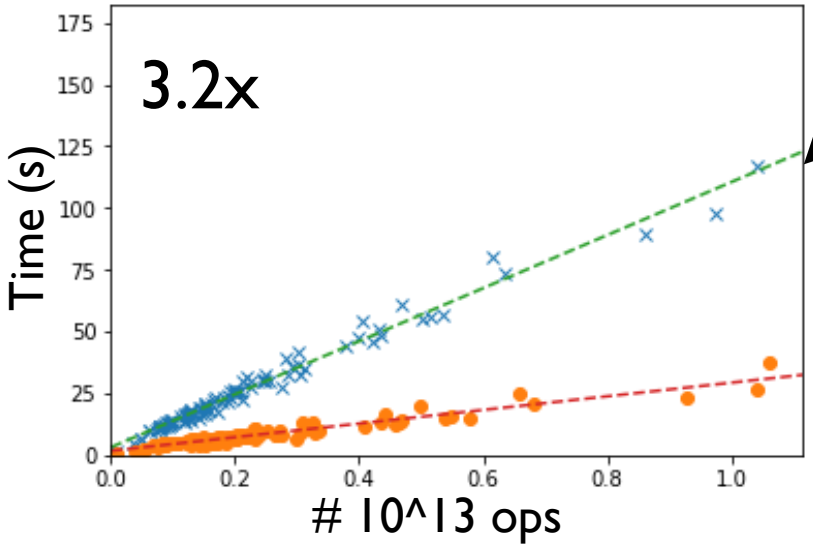


Hartree Multipole Summation

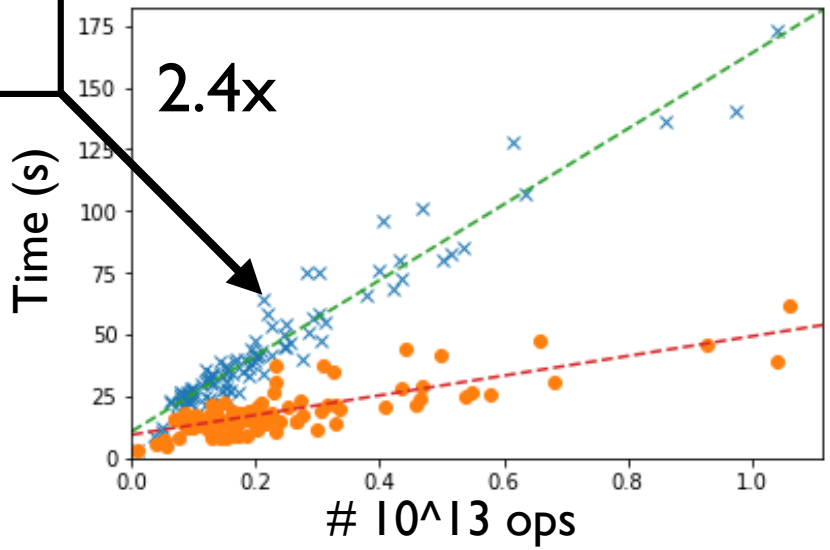


Hamiltonian, Density, Total Time agree with DGEMM estimate

Density



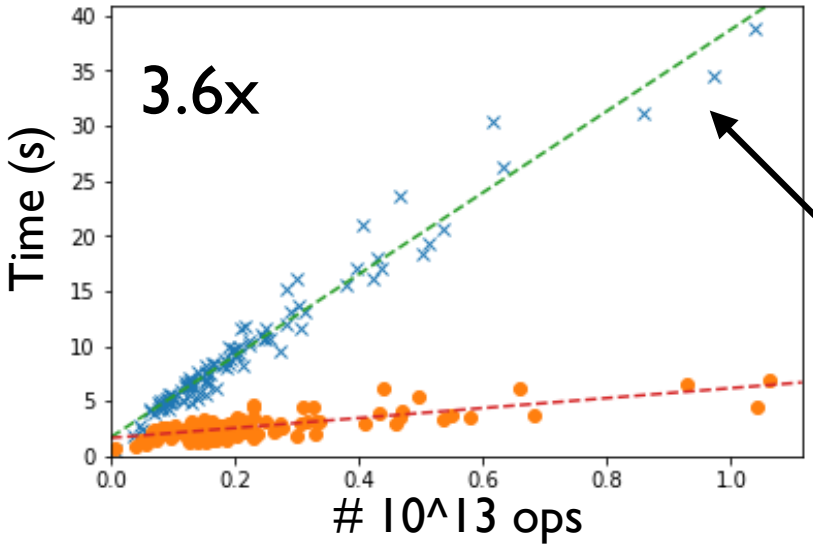
Total Time for Iteration



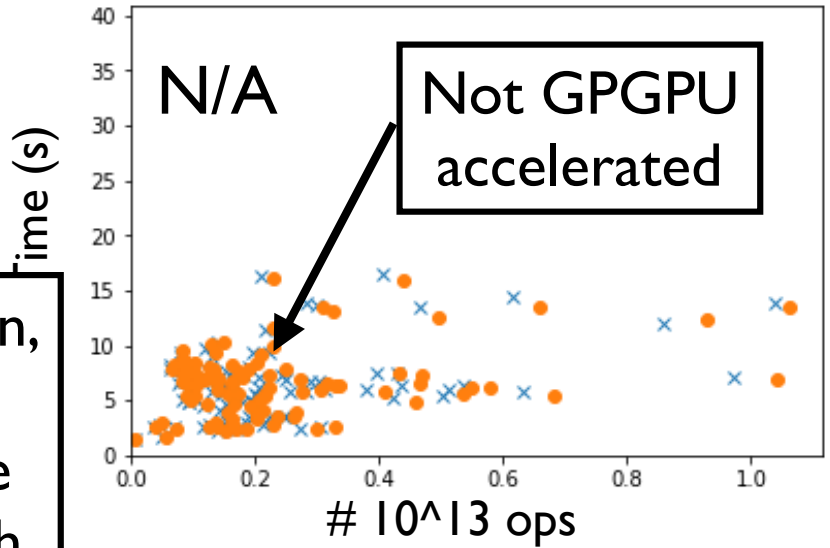
CPU
GPGPU
Speed-up

SCF Iteration for 103 Compounds: Skylake/V100

Hamiltonian Integration

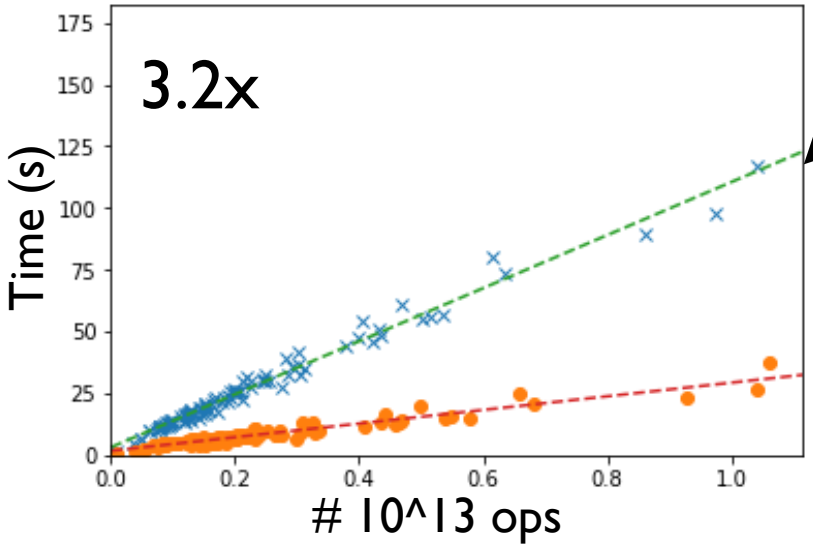


Hartree Multipole Summation

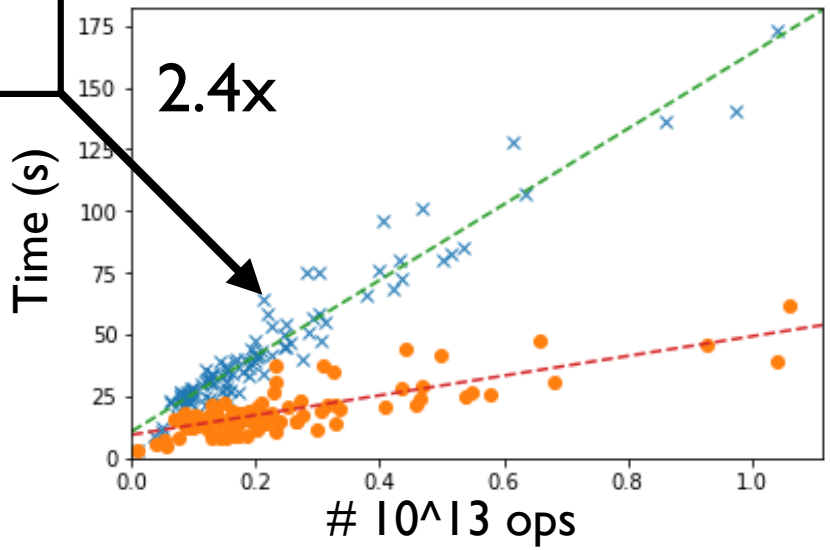


Hamiltonian, Density, Total Time agree with DGEMM estimate

Density



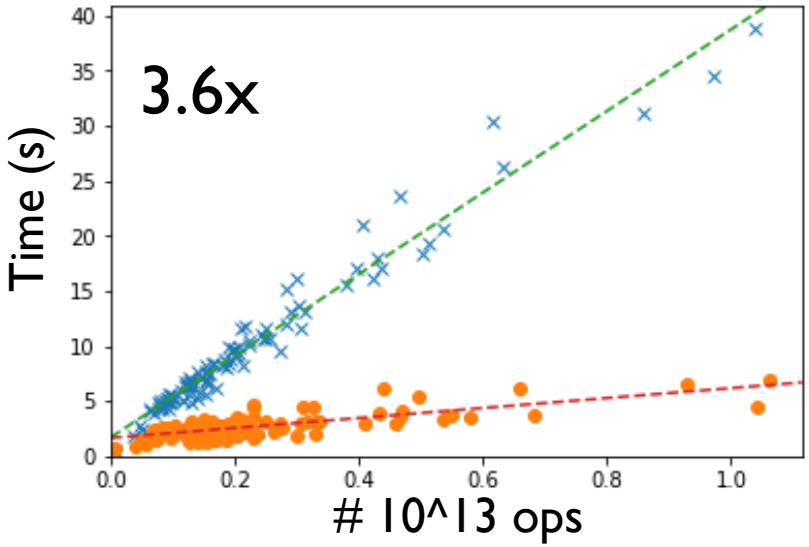
Total Time for Iteration



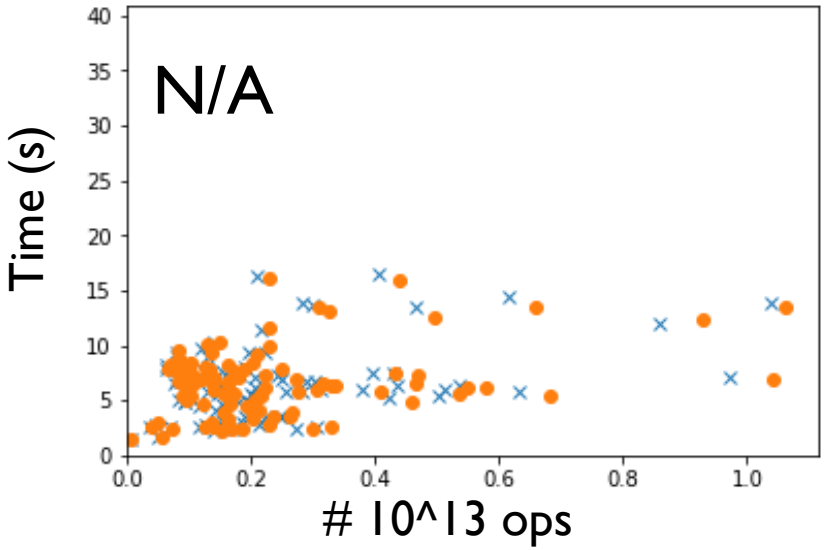
CPU
GPGPU
Speed-up

SCF Iteration for 103 Compounds: Skylake/V100

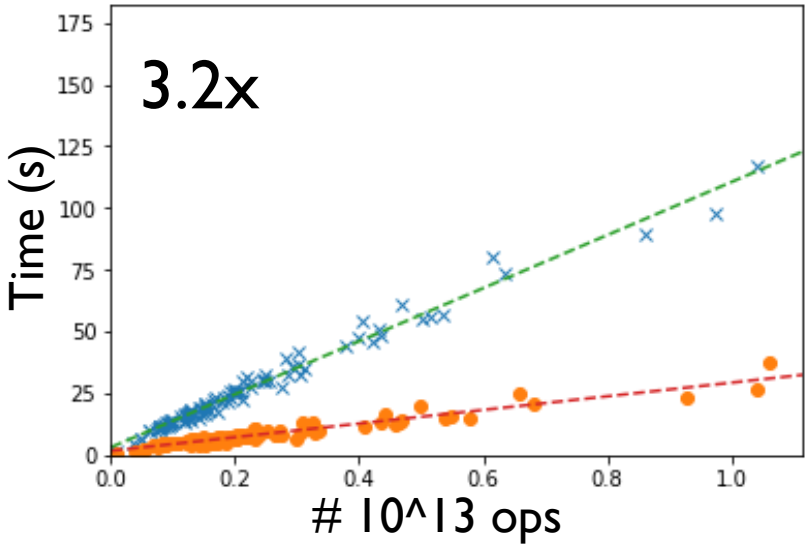
Hamiltonian Integration



Hartree Multipole Summation

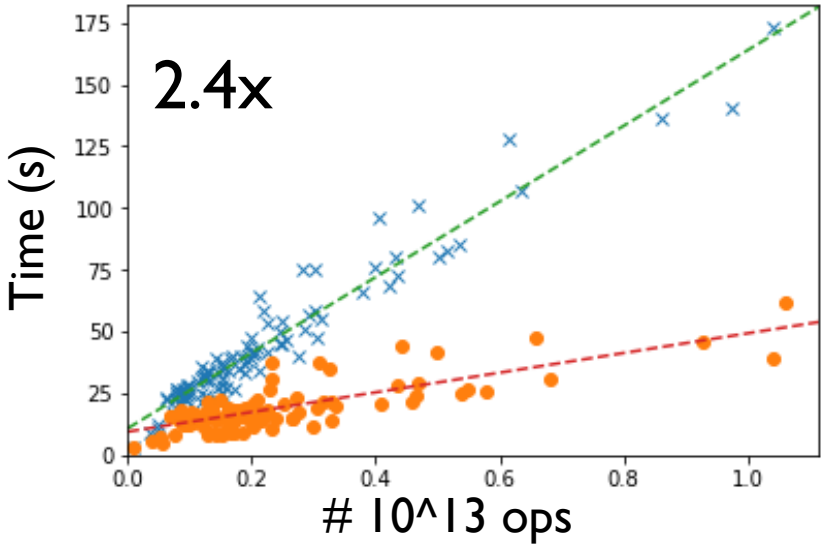


Density



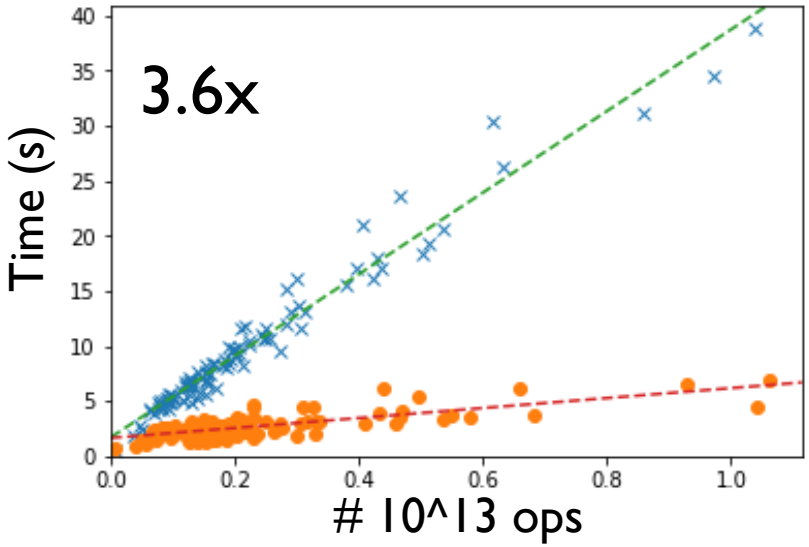
CPU
GPGPU
Speed-up

Total Time for Iteration

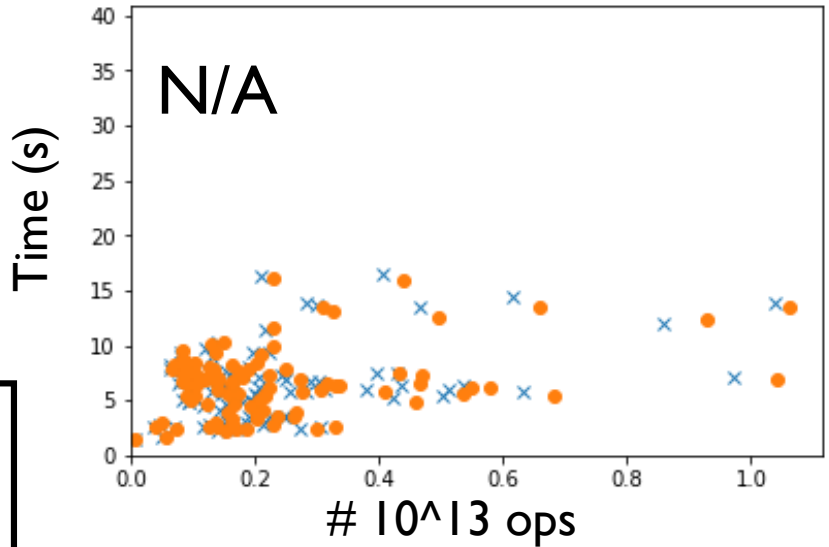


SCF Iteration for 103 Compounds: Skylake/V100

Hamiltonian Integration

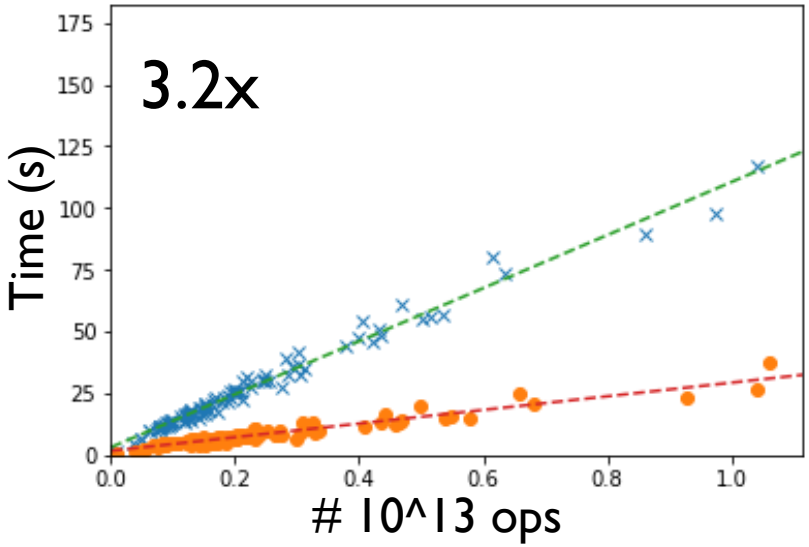


Hartree Multipole Summation

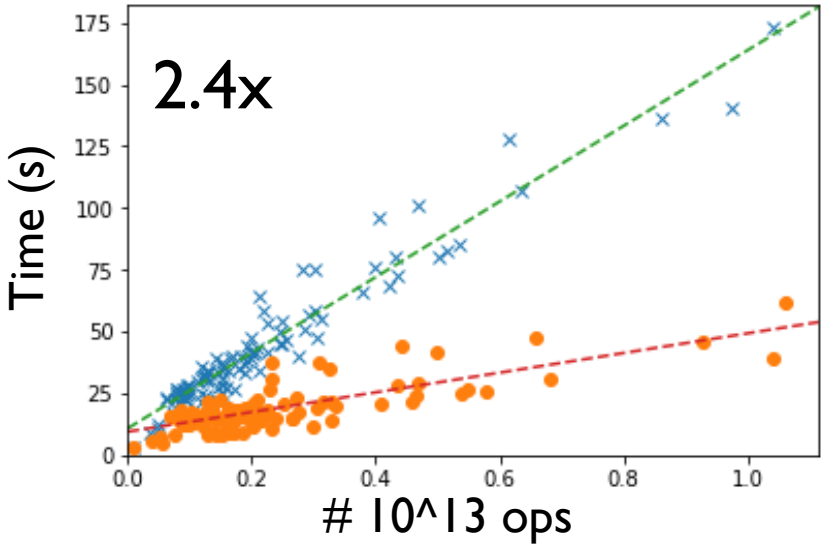


What if we calculate forces?

Density



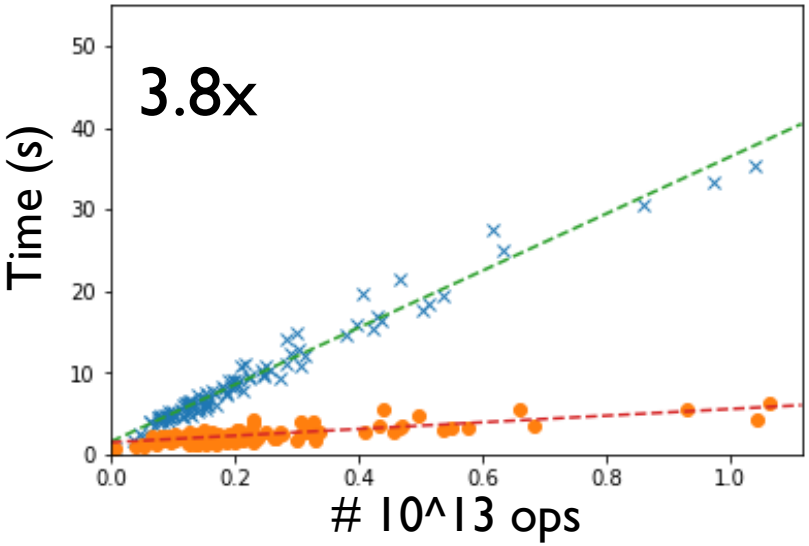
Total Time for Iteration



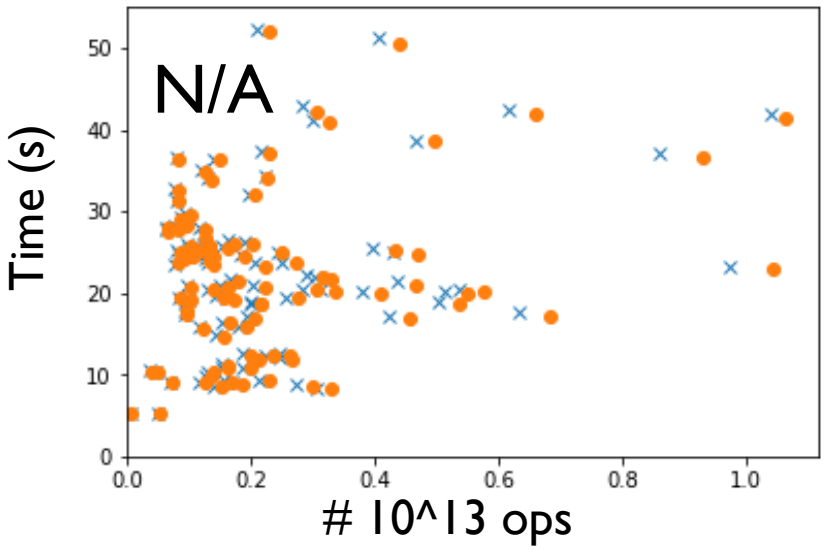
CPU
GPGPU
Speed-up

SCF Iteration + Forces for 103 Compounds: Skylake/V100

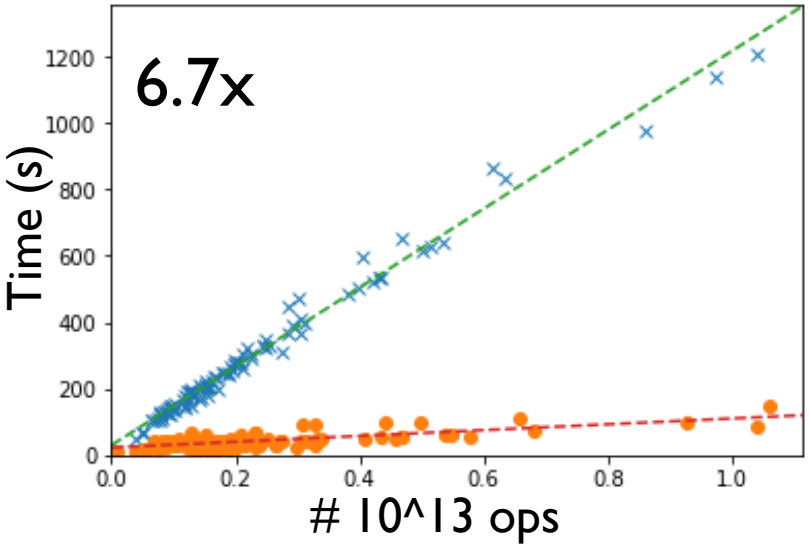
Hamiltonian Integration



Hartree Multipole Summation

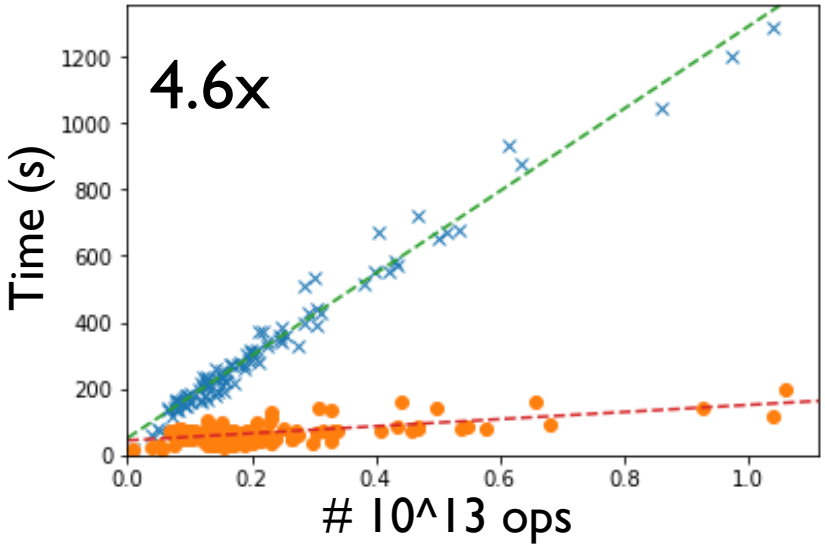


Density + Forces



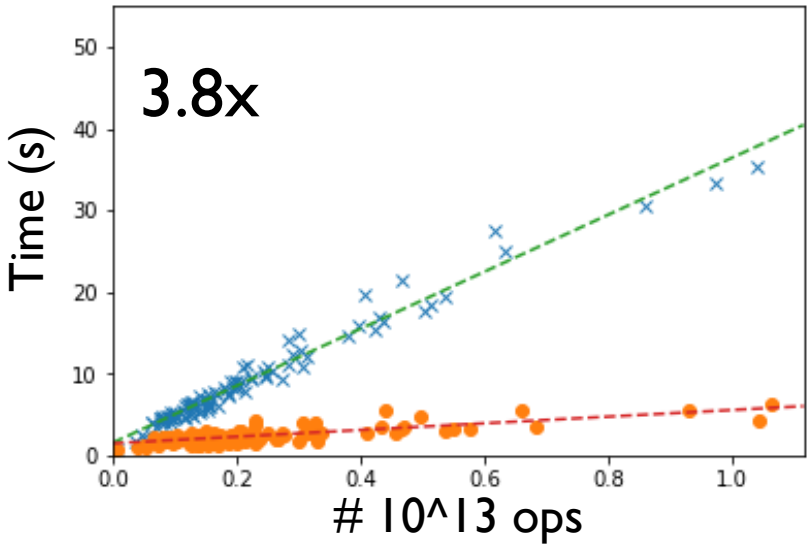
CPU
GPGPU
Speed-up

Total Time for Iteration

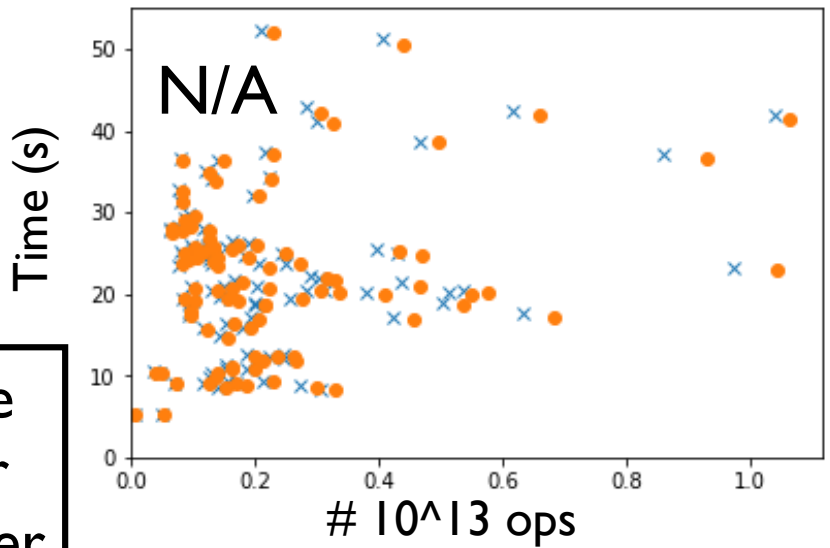


SCF Iteration + Forces for 103 Compounds: Skylake/V100

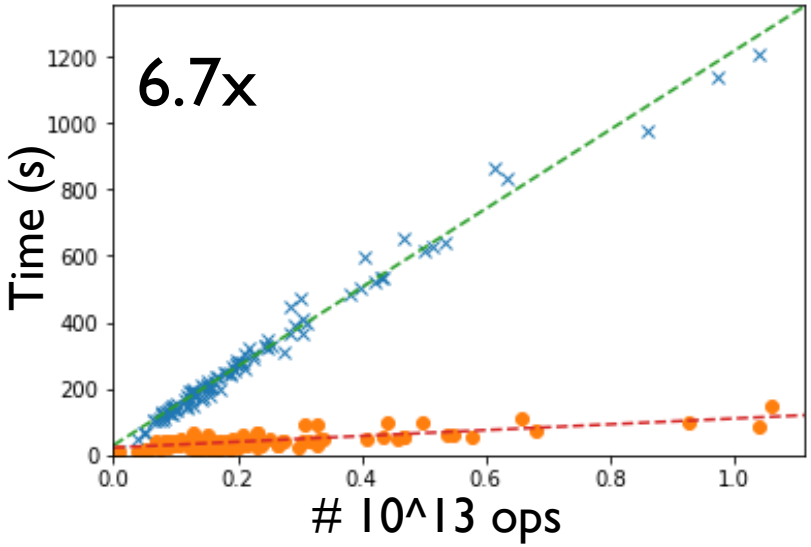
Hamiltonian Integration



Hartree Multipole Summation



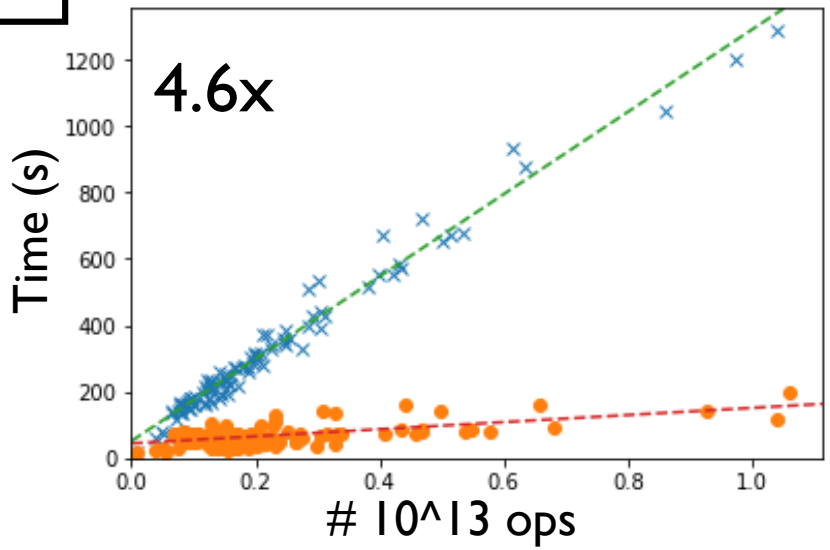
Density + Forces



More dense serial linear algebra, better speedups

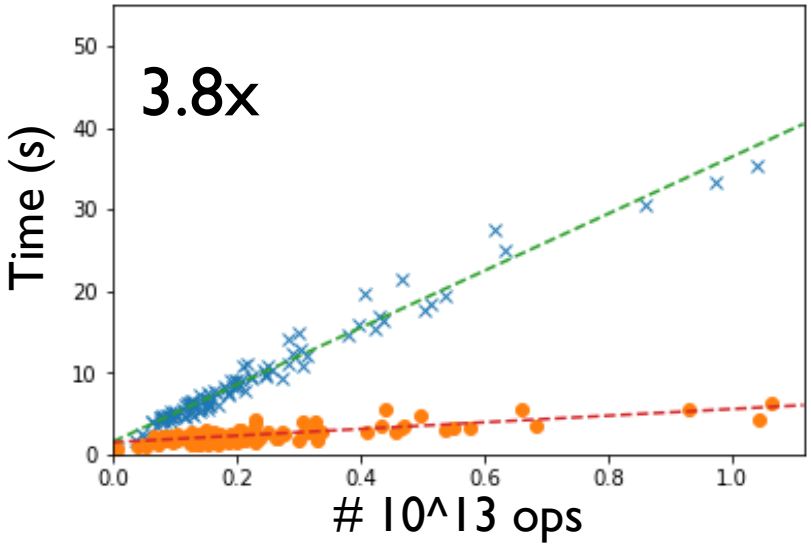
CPU
GPGPU
Speed-up

Total Time for Iteration

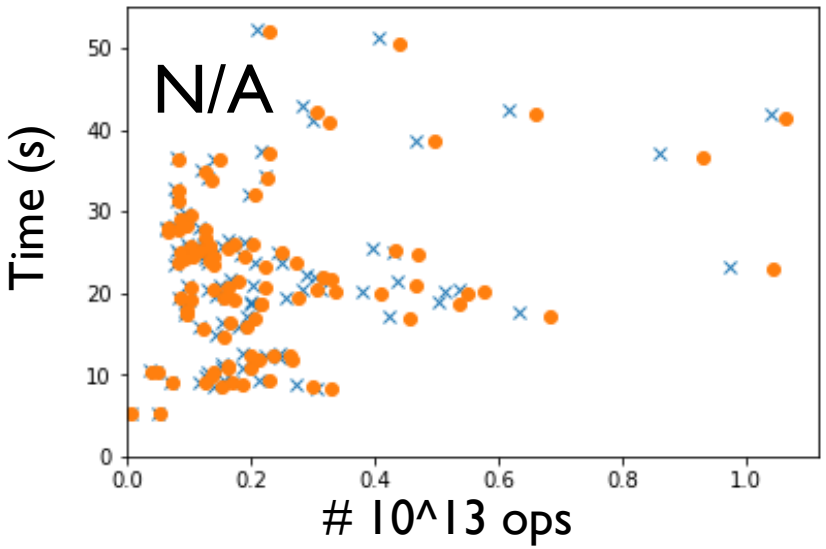


SCF Iteration + Forces for 103 Compounds: Skylake/V100

Hamiltonian Integration

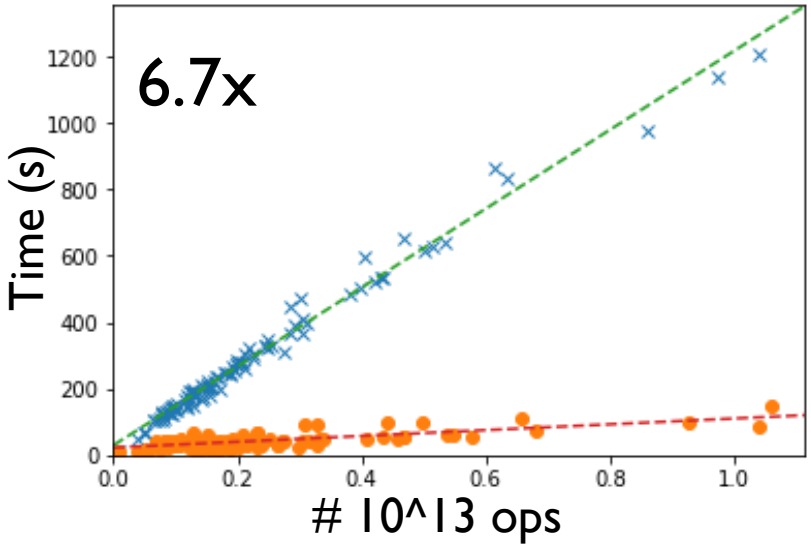


Hartree Multipole Summation

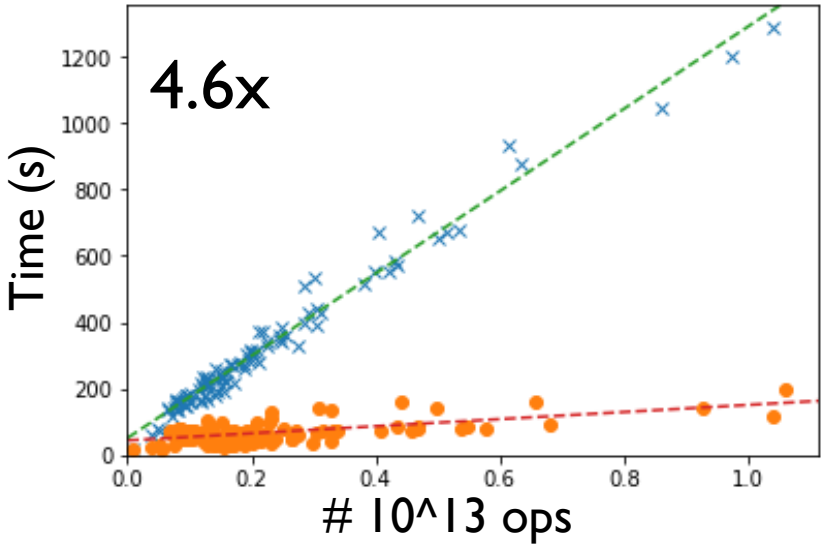


Now let's add analytical stress…

Density + Forces



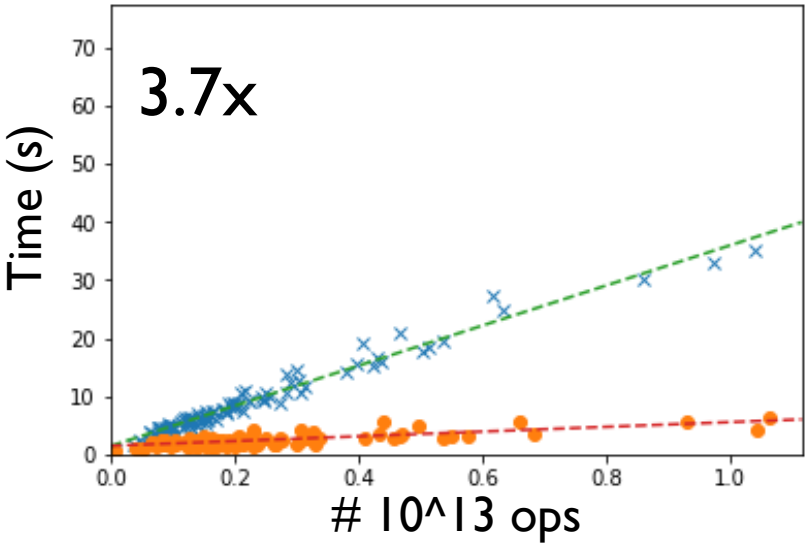
Total Time for Iteration



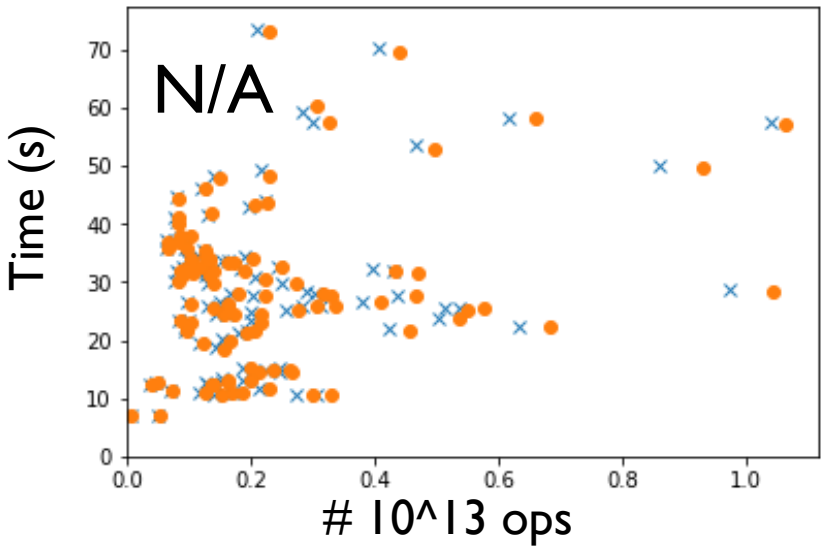
CPU
GPGPU
Speed-up

SCF Iteration + Forces + Stress for 103 Compounds: Skylake/V100

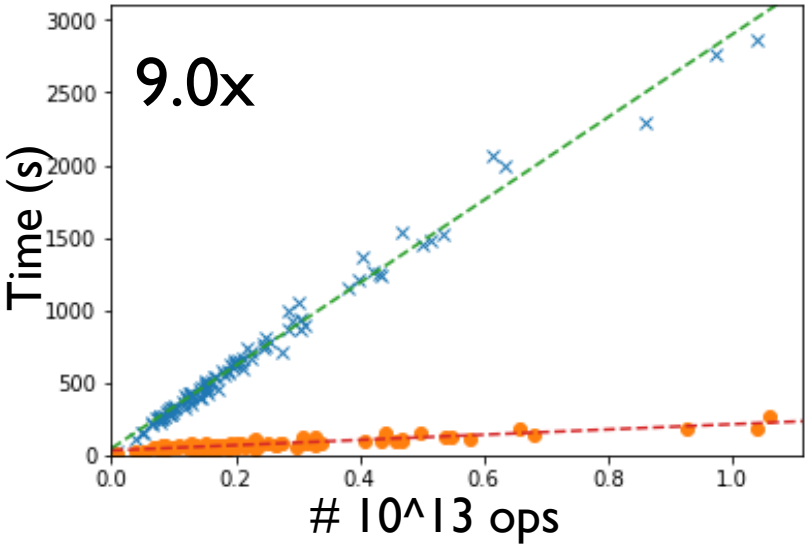
Hamiltonian Integration



Hartree Multipole Summation

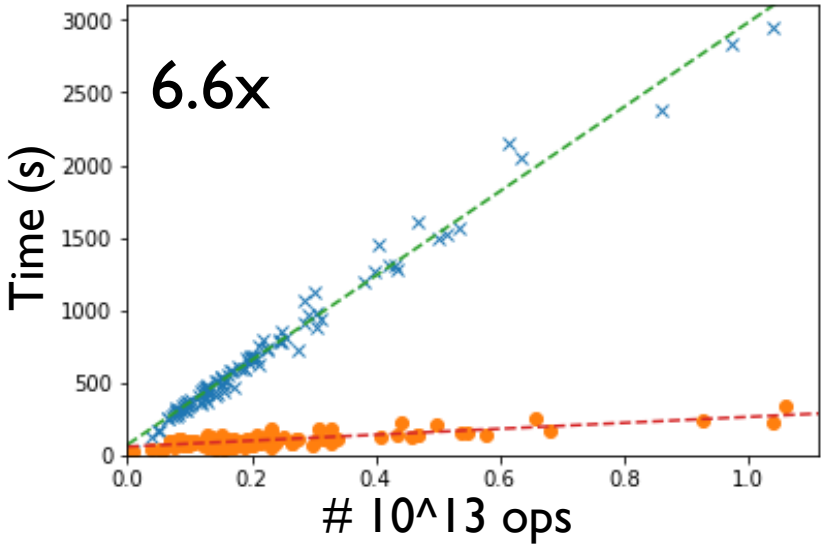


Density + Forces + Stress



CPU
GPGPU
Speed-up

Total Time for Iteration



TL; DR

Speedups for Diamond Si (3x3x3)

	Ivy Bridge/GP100	Haswell/P100	Skylake/V100
SCF Iteration	2.9x	2.5x	3.1x
SCF Iteration + Forces	4.4x	5.8x	5.7x
SCF Iteration + Forces + AS	5.2x	6.7x	7.8x
Entire Calculation	**3.8x**	**3.7x**	**4.0x**

Speedups for 103 Compound Test Set

SCF Iteration	2.4x	2.1x	2.4x
SCF Iteration + Forces	3.9x	4.5x	4.6x
SCF Iteration + Forces + AS	4.5x	6.6x	6.6x

TL; DR

Speedups for Diamond Si (3x3x3)

	Ivy Bridge/GP100	Haswell/P100	Skylake/V100
SCF Iteration	2.9x	2.5x	3.1x
SCF Iteration + Forces	4.4x	5.8x	5.7x
SCF Iteration + Forces + AS	5.2x	6.7x	7.8x
Entire Calculation	**3.8x**	**3.7x**	**4.0x**

Speedups for 103 Compound Test Set

SCF Iteration	2.4x	2.1x	2.4x
SCF Iteration + Forces	3.9x	4.5x	4.6x
SCF Iteration + Forces + AS	4.5x	6.6x	6.6x

TL; DR

Speedups for Diamond Si (3x3x3)

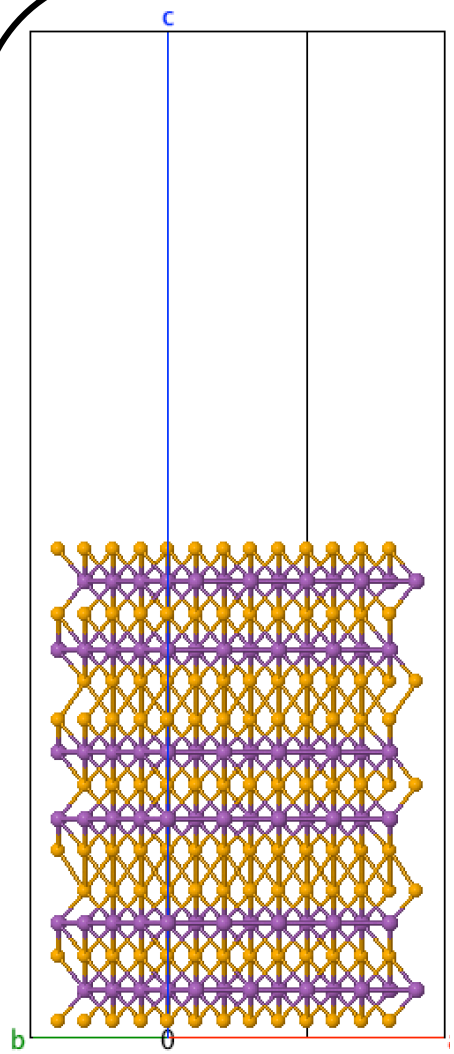
	Ivy Bridge/GP100	Haswell/P100	Skylake/V100
SCF Iteration	2.9x	2.5x	3.1x
SCF Iteration + Forces	4.4x	5.8x	5.7x
SCF Iteration + Forces + AS	5.2x	6.7x	7.8x
Entire Calculation	**3.8x**	**3.7x**	**4.0x**

But does it scale?

Speedups for 103 Compound Test Set

SCF Iteration	2.4x	2.1x	2.4x
SCF Iteration + Forces	3.9x	4.5x	4.6x
SCF Iteration + Forces + AS	4.5x	6.6x	6.6x

Strong Scaling for GPGPU Calculations: 375 Atom Bi_2Se_3 Slab

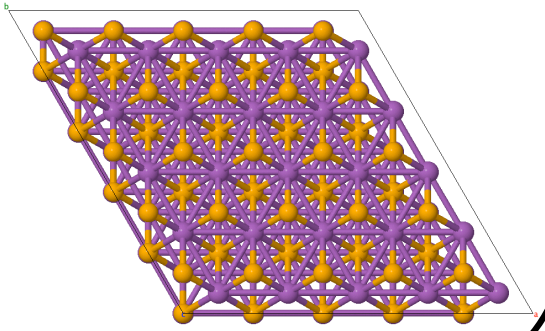


Bi_2Se_3 Slab
5x5x1 Supercell
375 atoms

25 Å slab,
32 Å vacuum

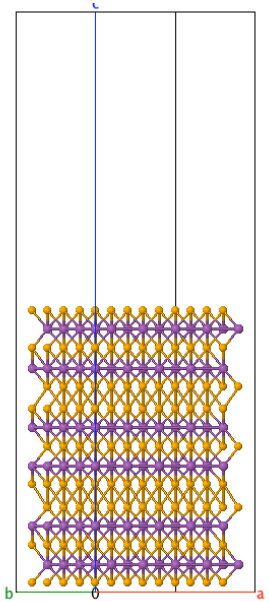
FHI-aims, tight settings (47.6 basis functions/atom)

PBE functional



Node	Opteron/K20X
Cluster	Titan Cray XK7
CPU	1x AMD Opteron 6274 (16 cores, Bulldozer)
GPGPU	1x Tesla K20X (Kepler)
MPI Tasks/GPGPU	16/1
Compilers/Libraries	PGI 17.9.0, Cray LibSci 16.11.1, Cray MPICH 7.6.3, CUDA 7.5

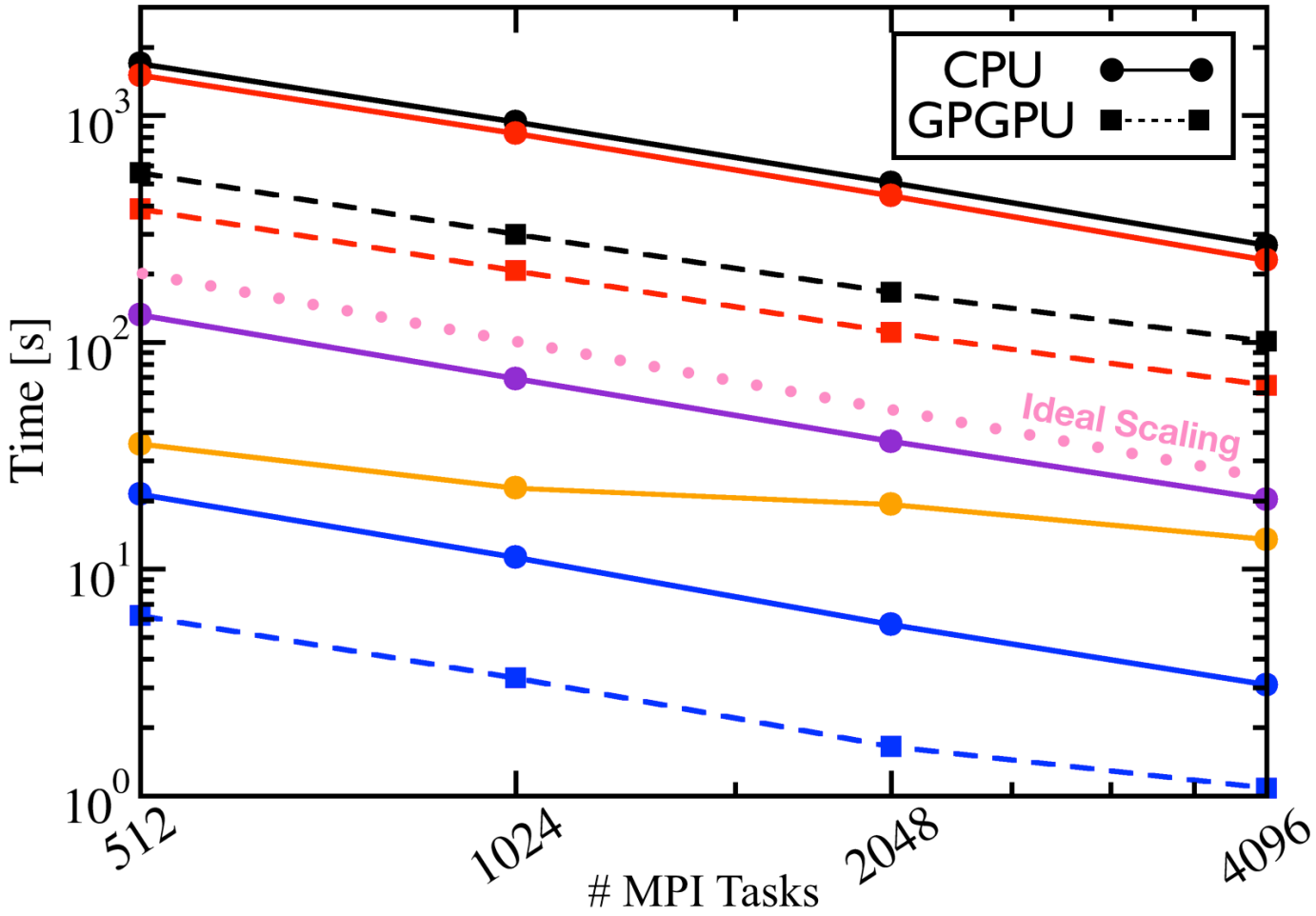
SCF Iteration + Forces + Stress on Titan: 375 Atom Bi_2Se_3 Slab



Bi_2Se_3 Slab
5x5x1 Supercell
375 atoms

FHI-aims, tight settings, PBE

47.6 basis functions per atom (17850 total)



Total Time for Iteration: 2.6x - 3.1x

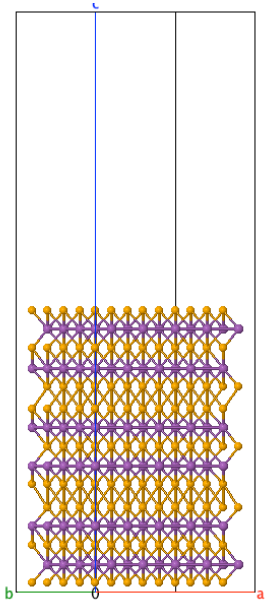
Density + Stress: 3.6x - 4.0x

KS Solution: N/A

Hamiltonian: 2.9x - 3.5x

Hartree Sum: N/A

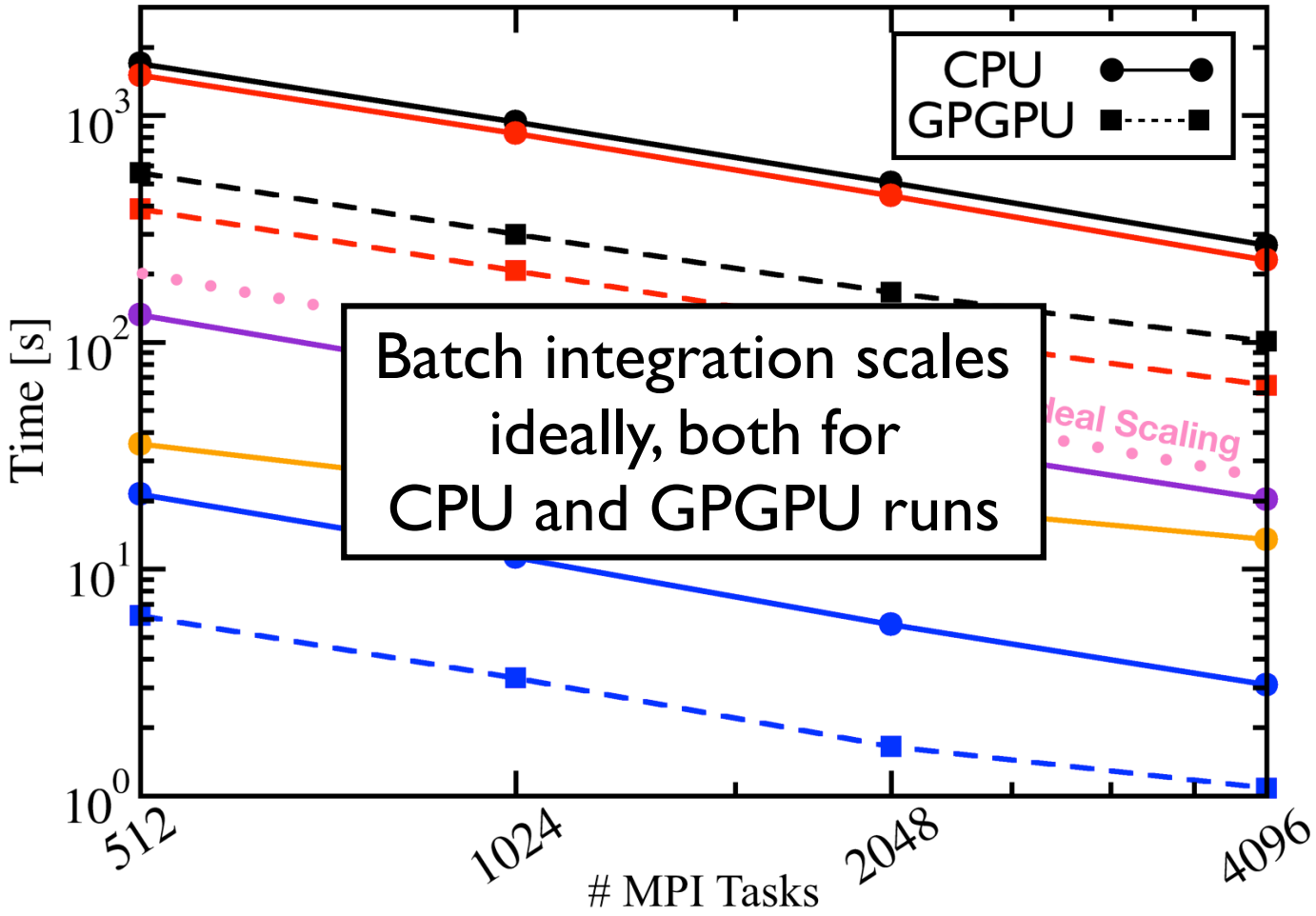
SCF Iteration + Forces + Stress on Titan: 375 Atom Bi_2Se_3 Slab



Bi_2Se_3 Slab
5x5x1 Supercell
375 atoms

FHI-aims, tight settings, PBE

47.6 basis functions per atom (17850 total)



Total Time for Iteration: 2.6x - 3.1x

Density + Stress: 3.6x - 4.0x

KS Solution: N/A

Hamiltonian: 2.9x - 3.5x

Hartree Sum: N/A

Conclusion and Acknowledgements

All-electron real-space DFT can be effectively GPGPU-accelerated, with ideal scaling on HPC resources, using domain decomposition

For realistic (non-hybrid) calculations with geometry relaxation, can expect ≈3x-4x speedup with FHI-aims on modern architectures

GPGPU acceleration of real-space operations shown in this talk available on mainline FHI-aims git repo, production-ready

Work supported by the LDRD Program of ORNL managed by UT-Battle, LLC, for the U.S. DOE and by the Oak Ridge Leadership Computing Facility, which is a DOE Office of Science User Facility supported under Contract DE-AC05-00OR22725.

We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Quadro GP100 and Titan V used for local development, as well as access to their PSG cluster.

OAK RIDGE National Laboratory | OAK RIDGE LEADERSHIP COMPUTING FACILITY

ELSI

NVIDIA

Duke UNIVERSITY | PRATT SCHOOL of ENGINEERING