NEW GPU FUNCTIONALITY IN VASP WITH OPENACC AND CUDA LIBRARIES

Stefan Maintz, 2019/12/18
AGENDA

Introduction to VASP

GPU Acceleration in VASP 5

Prioritizing Use Cases for New Porting Efforts

OpenACC in VASP 6 and Supported Features

Comparative Benchmarking
INTRODUCTION TO VASP

Scientific Background

A leading electronic structure program for solids, surfaces and interfaces

Used to study chemical and physical properties, reactions paths, etc.

Atomic scale materials modeling from first principles from 1 to 1000s atoms

Liquids, crystals, magnetism, semiconductors/insulators, surfaces, catalysts

Solves many-body Schrödinger equation

\[ H \Psi = i \hbar \frac{\partial}{\partial t} \Psi \]
\[ H \Psi = E \Psi \]

\[ H = -\sum_{i} \frac{\hbar^2}{2m_e} \Delta_i + \sum_{i} \sum_{i' > i} \frac{1}{4\pi \varepsilon_0} \frac{e^2}{|\vec{r}_i - \vec{r}_{i'}|} - \sum_{i} \sum_{a} \frac{1}{4\pi \varepsilon_0} \frac{Z_a e^2}{|\vec{r}_i - \vec{R}_a|} V_{\text{ext}}(\vec{r}_i) V_{\text{ext}} \]
Density Functional Theory (DFT)

\[ n(\vec{r}_1) = N_e \int \cdots \int |\Psi(\vec{r}_1, \vec{r}_2, \ldots, \vec{r}_{N_e})|^2 \, d\vec{r}_2 \cdots d\vec{r}_{N_e} \]

Enables solving sets of Kohn-Sham equations

\[ \left( -\frac{1}{2} \Delta_n + v_{\text{eff}}(\vec{r}_n) \right) \psi_n(\vec{r}) = e_n^{\text{KS}} \psi_n(\vec{r}) \]

In a plane-wave based framework (PAW)

\[ \psi_n(\vec{k}, \vec{r}) = \frac{1}{\sqrt{\Omega}} \sum_{\vec{G}}^{N_\vec{G}} C_{\vec{G},n}(\vec{k}) e^{i(\vec{k}+\vec{G}) \cdot \vec{r}} \]

Hybrid DFT adding (parts of) exact exchange (Hartree-Fock) and VASP can go even beyond!
INTRODUCTION TO VASP

12-25% of CPU cycles at supercomputing centers

Academia

Material Sciences
Chemical Engineering
Physics & Physical Chemistry

Companies

Large semiconductor companies
Oil & gas
Chemicals - bulk or fine
Materials - glass, rubber, ceramic, alloys, polymers and metals

### Top 5 HPC Applications

<table>
<thead>
<tr>
<th>Rank</th>
<th>Application</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>GROMACS</td>
</tr>
<tr>
<td>2</td>
<td>ANSYS Fluent</td>
</tr>
<tr>
<td>3</td>
<td>Gaussian</td>
</tr>
<tr>
<td>4</td>
<td>VASP</td>
</tr>
<tr>
<td>5</td>
<td>NAMD</td>
</tr>
</tbody>
</table>

Source: Intersect360 2017 Site Census Mentions
INTRODUCTION TO VASP

Details on the code

Developed by Prof. Kresse’s group at University of Vienna (and external contributors)

Under development/refactoring for about 25 years

460K lines of Fortran 90, some FORTRAN 77

MPI parallel, OpenMP recently added for multicore

First endeavors on GPU acceleration date back to <2011 timeframe with CUDA C
INTRODUCTION TO VASP

Computational characteristics

Lots of small Fast-Fourier-Transformation (about 100x100x100 nodes)
Matrix-Matrix and Matrix-Vector multiplications
Matrix diagonalizations
AllToAll communications
And of course some custom kernels
AGENDA

Introduction to VASP

GPU Acceleration in VASP 5

Prioritizing Use Cases for New Porting Efforts

OpenACC in VASP 6 and Supported Features

Comparative Benchmarking
COLLABORATION ON CUDA C PORT OF VASP 5

Collaborators

CUDA Port Project Scope

Minimization algorithms to calculate electronic ground state:
  Blocked Davidson (ALGO = Normal & Fast) and RMM-DIIS (ALGO = VeryFast & Fast)
Parallelization over k-points
Exact-exchange calculations

Earlier Work

*Speeding up plane-wave electronic-structure calculations using graphics-processing units* (Maintz, Eck, Dronskowski)

*VASP on a GPU: Application to exact-exchange calculations of the stability of elemental boron* (Hutchinson, Widom)

*Accelerating VASP Electronic Structure Calculations Using Graphic Processing Units* (Hacene, Anciaux-Sedrakian, Rozanska, Klahr, Guignon, Fleurat-Lessard)
INSTRUCTIONS TO COMPILE AND RUN VASP 5 ON GPUS

NVIDIA offers step-by-step instructions to compile and run VASP 5 with the CUDA C port:

https://www.nvidia.cn/data-center/gpu-accelerated-applications/vasp/

Exemplary benchmarks to test against expected performance

Hints on tuning important run-time parameters

English version at https://www.nvidia.com/vasp
CUDA SOURCE INTEGRATION IN VASP 5.4.4

Original source tree (Fortran)

Accelerated call tree, drop-in replacements (Fortran)

Custom kernels and support code (CUDA C)
CUDA C ACCELERATED VERSION OF VASP

Available today on NVIDIA Tesla GPUs with VASP 5.4.4

All GPU acceleration with CUDA C

Only some cases are ported to GPUs

Different source trees for Fortran vs CUDA C

CPU code gets continuously updated and enhanced, required for various platforms

Challenge to keep CUDA C sources up-to-date

Long development cycles to port new features
CUDA C ACCELERATED VERSION OF VASP

Available today on NVIDIA Tesla GPUs with VASP 5.4.4

Source code duplication for CUDA C in VASP led to:

• increased maintenance cost
• improvements in CPU code need replication
• long development cycles to port new solvers

Explore OpenACC as an improvement for GPU acceleration
AGENDA

Introduction to VASP

GPU Acceleration in VASP 5

Prioritizing Use Cases for New Porting Efforts

OpenACC in VASP 6 and Supported Features

Comparative Benchmarking
## FEATURES AVAILABLE AND ACCELERATED IN VASP 5

### LEVELS OF THEORY
- Standard DFT
- Hybrid DFT (exact exchange)
- RPA (ACFDT, GW)
- Bethe-Salpeter Equations (BSE)
- ...

### SOLVERS / MAIN ALGORITHM
- Davidson
- RMM-DIIS
- Davidson+RMM-DIIS
- Direct optimizers (Damped, All)
- Linear response
- ...

### PROJECTION SCHEME
- Real space
- Real space (automatic optimization)
- Reciprocal space

### EXECUTABLE FLAVORS
- Standard variant
- Gamma-point simplification variant
- Non-collinear spin variant

*Light green: GPU accelerated in VASP 5, black: not accelerated in VASP 5*
EXAMPLE BENCHMARK: CuC\textsubscript{vdW}

- **Standard DFT** level of theory
  - Davidson solver
  - RMM-DIIS solver
  - Dav.+RMM-DIIS solver
  - Damped solver

- **Hybrid DFT** level of theory
  - Real space proj. scheme
  - Reciprocal proj. scheme
  - Automatic proj. scheme

- **RPA** level of theory
  - Gamma-point exec. flavor
  - Non-collinear exec. flavor

- **BSE** level of theory
  - Standard exec. flavor
  - KPAR, NSIM, NCORE parallelization options
PARALLELIZATION LAYERS IN VASP

Wavefunction

Spins

$k$-points

Bands/Orbitals

Plane-wave coefficients

Physical quantities

\[ \Psi \]

\[ \uparrow \quad \downarrow \]

\[ k_1 \quad k_2 \quad k_3 \quad \ldots \]

\[ n_1 \quad n_2 \quad \ldots \quad n_1 \quad n_2 \quad \ldots \]

\[ C_1 \quad C_2 \quad C_3 \quad \ldots \quad C_1 \quad C_2 \quad C_3 \quad \ldots \]

\[ \ldots \quad \ldots \quad \ldots \quad \ldots \quad \ldots \quad \ldots \quad \ldots \quad \ldots \quad \ldots \quad \ldots \quad \ldots \quad \ldots \quad \ldots \]

KPAR>1

Default

NCORE>1

Parallelization feature
## PARALLELIZATION OPTIONS

### KPAR
Distributes k-points

Highest level parallelism, more or less embarrassingly parallel

Can help for smaller systems

Not always possible

### NSIM
Blocking of orbitals

Grouping (no distributing) influences caching and communication

Ideal value differs between CPU and GPU

Needs to be tuned

### NCORE
Distributes plane waves

Lowest level parallelism, needs parallel 3D FFT, inserts lots of MPI msgs

Can help with load balancing problems

*No support in CUDA port*
POSSIBLE USE CASES IN VASP
Each with a different computational profile

Supports a plethora of run-time options that define the workload (use case)

Those methodological options can be grouped into categories

Some, but not all are combinable

Combination determines if GPU acceleration is supported and also how well

Benchmarking the complete situation is tremendously complex
WHERE TO START

You cannot accelerate everything (at least soon)

Ideally every use case would be ported

Standard and Hybrid DFT alone give 72 use cases (ignoring parallelization options)!

Need to select most important use-cases

Selection should be based on real-world or supercomputing-facility scenarios
Zhengji Zhao (NERSC) collected such data (INCAR) for 30397 VASP jobs over nearly 2 months

Data is based on job count, but has no timing information

Includes 130 unique users on Edison (CPU-only system)

No 1:1-mapping of parameters possible, expect large error margins

Data does not include calculation sizes, but it’s a great start
VASP FEATURE USAGE AT NERSC
Levels of theory and main algorithms based on job count

Source: based on data provided by Zhengji Zhao, NERSC, 2014
SUMMARY

Where to start

Start with standard DFT, to accelerate most jobs

RMM-DIIS and Davidson nearly equally important, share a lot of routines anyway

Real-space projection scheme more important for large setups

Gamma-point executable flavor as important as standard, so start with general one

Support as many parallelization options as possible (KPAR, NSIM, NCORE)

Communication is important, but scaling to large node counts is low priority
(62% fit into 4 nodes, 95% used ≤12 nodes)
AGENDA

- Introduction to VASP
- GPU Acceleration in VASP 5
- Prioritizing Use Cases for New Porting Efforts
- OpenACC in VASP 6 and Supported Features
- Comparative Benchmarking
OPENACC DIRECTIVES

Data directives are designed to be optional

Manage Data Movement

Initiate Parallel Execution

Optimize Loop Mappings

 !$acc data copyin(a,b) copyout(c)

 ... !$acc parallel

 !$acc loop gang vector

 do i=1, n
  
 c(i) = a(i) + b(i)

 ... 

 enddo 

 !$acc end parallel

 ...

 !$acc end data
DERIVED TYPES IN OPENACC

1. allocates device memory for var_dyn
2. copies m (H2D)
3. copies host pointer for var_dyn%r!
   -> device ptr invalid

1. allocates device memory for r
2. copies r (H2D)
3. attaches the device copy’s pointer var_dyn%r to the device copy of r

1. copies r (D2H)
2. deallocates device memory for r
3. detaches var_dyn%r on the device, i.e. overwrites r with its host value!
   -> device ptr invalid

1. copies m (D2H)
2. copies var_dyn%r -> host pointer intact!
3. deallocates device memory for var_dyn
MANAGING VASP AGGREGATE DATA STRUCTURES

OpenACC + Unified Memory not an option today, some aggregates have static members

OpenACC 2.6 manual deepcopy was key

Requires large numbers of directives in some cases, but well encapsulated (107 lines for COPYIN)

Future spec of OpenACC might add true deep copy, require far fewer data directives

When CUDA Unified Memory + HMM supports all classes of data, potential for a VASP port with no data directives at all

Manual deepcopy allowed to port VASP
INTERFACING NVIDIA CUDA LIBRARIES
Subtitle Optional

VASP 6 leverages: cuBLAS, cuFFT, cuSolver, NCCL and CUDA-aware MPI

transparently by encapsulated routines

```c
#ifdef _OPENACC
#define myZGEMM ZGEMM
#else
#define myZGEMM ACC_ZGEMM
#endif

CALL myZGEMM(‘N’,’N’,M,N,K,ALPHA,A,& LDA,B,LDB,BETA,C,LDC)

SUBROUTINE ACC_ZGEMM(OPA,OPB,M,N,K,ALPHA,A,& LDA,B,LDB,BETA,C,LDC)
...
IF (ACC_ACTIVE) THEN
    !$ACC HOST_DATA USE_DEVICE(A,B,C)
    CALL cublasZGEMM(OPA,OPB,M,N,K,ALPHA,A,& LDA,B,LDB,BETA,C,LDC)
    !$ACC END HOST_DATA
ELSE
    CALL ZGEMM(OPA,OPB,M,N,K,ALPHA,A,& LDA,B,LDB,BETA,C,LDC)
ENDIF
END SUBROUTINE ACC_ZGEMM
```
NVIDIA Devtech collaborating with VASP developers to migrate to OpenACC + CUDA Libraries

Development uses the PGI Compiler (Community Edition is free-of-charge)

Developers own/maintain the GPU code in Fortran, single source

Targeting a much wider set of VASP features and improved performance

Will be part of VASP 6 as announced at SC19 to be released before Christmas 2019
GPU ACCELERATED FEATURES IN VASP 6

LEVELS OF THEORY
- Standard DFT
- Hybrid DFT (exact exch., double buffered)
- Cubic-scaling RPA (ACFDT, GW)
- Bethe-Salpeter Equations (BSE)
- Davidson (+Adaptively Compressed Exch.)
- RMM-DIIS
- Davidson+RMM-DIIS
- Direct optimizers (Damped, All)
- Linear response
- ...
AGENDA

Introduction to VASP

GPU Acceleration in VASP 5

Prioritizing Use Cases for New Porting Efforts

OpenACC in VASP 6 and Supported Features

Comparative Benchmarking
DETAILS ON DATASET

\textbf{CuC\_vdw}

Cellsize: 10.3x10.3x31.5 Å³
Atoms: 96 Cu, 2 C (98 total)
5 k-points, 638 bands, 400 eV cutoff Energy, 52,405 PWs
Standard DFT (GGA: PBE)

Algo=VeryFast (RMM-DIIS)
Real-space projection scheme
BENCHMARK RESULTS CuC_vDW

- VASP 5
  - 2x E5-2698 v4: 1.0
  - 1 V100: 1.7
  - 2 V100: 2.2
  - 4 V100: 2.9
  - 8 V100: 3.3

- VASP 6RC
  - 2x E5-2698 v4: 1.0
  - 1 V100: 2.3
  - 2 V100: 3.3
  - 4 V100: 3.3
  - 8 V100: 5.4

- VASP 6+
  - 2x E5-2698 v4: 1.0
  - 1 V100: 2.5
  - 2 V100: 3.7
  - 4 V100: 4.7
  - 8 V100: 6.6
DETAILS ON DATASET

Si-Huge

Cellsize: 15.4x30.7x30.7 Å³
Atoms: 512 Si
14 k-points, 1281 bands, 245.4 eV cutoff Energy, 89,614 PWs
Standard DFT (GGA: PW91)
Algo=Normal (Davidson)
Real-space projection scheme
BENCHMARK RESULTS Si-HUGE

- 2x E5-2698 v4
- 1 V100
- 2 V100
- 4 V100
- 8 V100

*: 32GB V100

<table>
<thead>
<tr>
<th></th>
<th>VASP 5</th>
<th>VASP 6RC</th>
<th>VASP 6+</th>
</tr>
</thead>
<tbody>
<tr>
<td>Speedup</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td></td>
<td>2.6</td>
<td>3.1*</td>
<td>3.1*</td>
</tr>
<tr>
<td></td>
<td>2.9</td>
<td>5.5*</td>
<td>5.5*</td>
</tr>
<tr>
<td></td>
<td>3.8</td>
<td>7.0</td>
<td>6.3</td>
</tr>
<tr>
<td></td>
<td>4.7</td>
<td>10.5</td>
<td>10.7</td>
</tr>
</tbody>
</table>
DETAILS ON DATASET

GaAsBi_512

Cellszie: 22.6x22.6x22.6 Å³
Atoms: 256 Ga, 255 As, 1 Bi (512 total)
4 k-points, 1536 bands, 313 eV cutoff Energy, 145,484 PWs
Standard DFT (GGA: PBE)
Algo=Fast (Davidson + RMM-DIIS)
Real-space projection scheme
BENCHMARK RESULTS GaAsBi_512

- **VASP 5**:
  - 2x E5-2698 v4: 1.0
  - 1 V100: 2.8
  - 2 V100: 4.5
  - 4 V100: 7.0
  - 8 V100: 9.8

- **VASP 6RC**:
  - 2x E5-2698 v4: 1.0
  - 1 V100: 3.8*
  - 2 V100: 6.2
  - 4 V100: 6.2
  - 8 V100: 10.7

- **VASP 6+**:
  - 2x E5-2698 v4: 1.0
  - 1 V100: 3.6*
  - 2 V100: 6.2
  - 4 V100: 6.2
  - 8 V100: 10.6

*: 32GB V100
DETAILS ON DATASET

Si256_VJT_PBE0

Cellsize: 18.9x18.9x18.9 Å³

Atoms: 256 Si

1 k-point (Γ), 640 bands, 250 eV cutoff Energy, 23,589 PWs

Hybrid DFT (PBE0)

Algo=Damped (direct minimizer)

Real-space projection scheme
BENCHMARK RESULTS SI256_VJT_PBE0

VASP 5

not yet GPU accelerated in VASP 5

VASP 6RC

2x E5-2698 v4
1 V100
2 V100
4 V100
8 V100

VASP 6+

2x E5-2698 v4
1 V100
2 V100
4 V100
8 V100

Speedup vs CPU
Benchmarked with experimental, pre-release versions of PGI compilers, CUDA toolkit, libraries and drivers. Performance is subject to change.
“For VASP, OpenACC is the way forward for GPU acceleration. Performance is similar and in some cases better than CUDA C, and OpenACC dramatically decreases GPU development and maintenance efforts.”

Prof. Georg Kresse
Computational Materials Physics
University of Vienna,
CEO of VASP Software GmbH