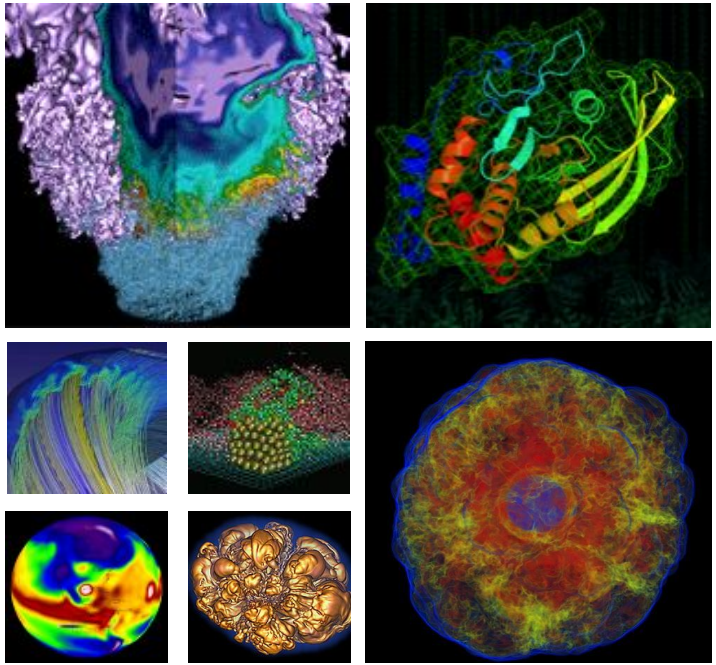
Porting Quantum ESPRESSO Hybrid Functional DFT to GPUs using CUDA Fortran



Thorsten Kurth, Brandon Cook, Rahul Gayatri, Zhengji Zhao, Jack Deslippe (NERSC/ LBNL)
Joshua Romero, Everett Phillips, Massimiliano Fatica (NVIDIA)

NERSC

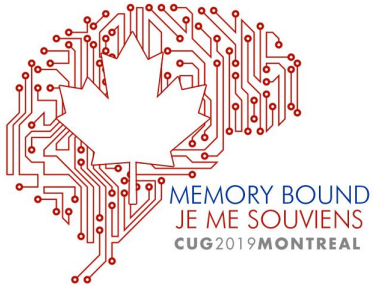
MEMORY BOUND
JE ME SOUVIENS
CUG2019MONTREAL

U.S. DEPARTMENT OF ENERGY | Office of Science

BERKELEY LAB

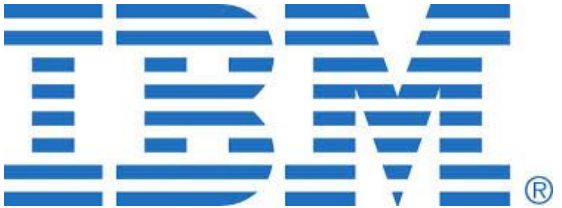
Outline

- CUDA Fortran
- Quantum ESPRESSO
  - hybrid DFT
- porting to GPU
  - Performance
- Mixed precision



CUDA Fortran

- Extensions to Fortran
  - Device data management
  - GPU subroutines + functions
  - Invocation of kernels
  - ACC-like loop directives
  - Interfaces for CUDA C & libraries
- Support
  - Primary: PGI
  - Other: IBM XL



CUDA Fortran

- CUDA C style code

```
! Kernel definition
attributes(global) subroutine ksaxpy( n, a, x, y )

   real, dimension(*) :: x,y
   real, value :: a
   integer, value :: n, i
   i = (blockidx%x-1) * blockdim%x + threadidx%x
   if( i <= n ) y(i) = a * x(i) + y(i)

end subroutine

! Host subroutine
subroutine solve( n, a, x, y )

   real, device, dimension(*) :: x, y
   real :: a
   integer :: n
   ! call the kernel

   call ksaxpy<<<n/64, 64>>>( n, a, x, y )
end subroutine
```

CUDA Fortran

```
COMPLEX(DP), ALLOCATABLE :: A(:)              !host array
COMPLEX(DP), ALLOCATABLE, DEVICE :: A_D(:)    !device array

ALLOCATE(A(N))
CALL FILL_A_HOST(A)

ALLOCATE(A_D, source=A) ! allocate on device and copy from host

A = A_D     !copy from device to host
```

CUDA Fortran

```
COMPLEX(DP), ALLOCATABLE :: A(:)              !host array
COMPLEX(DP), ALLOCATABLE, DEVICE :: A_D(:)    !device array

#ifdef _CUDA
associate(A=>A_D)
#endif

!$cuf kernel do(1)
do i=1,N                          !same loop for cpu & gpu
    A(i) = f(i)
end do

#ifdef _CUDA
end associate
#endif
```

CUDA Fortran

!$cuf kernel do[(n)] <<< grid, block [optional stream] >>>

- OpenACC style directives
- very convenient

```
!$cuf kernel do(2) <<< (*,*), (32,4) >>>
do j = 1, m
  do i = 1, n
    a(i,j) = b(i,j) + c(i,j)
  end do
end do
```

Quantum ESPRESSO overview

- Open source Integrated suite for electronic structure and materials modeling
  - density functional theory
  - plane waves basis set
  - pseudopotentials
  - generates input for many other codes
- Large HPC workload
- Wide range of capabilities
  - 50+ binaries
  - Fortran
  - Most™ time in Linear Algebra + FFTs

Quantum ESPRESSO hybrid DFT

Quantum Many Body Problem

$$\hat{H} = \underbrace{-\frac{\hbar^2}{2}\sum_{\alpha}\frac{\nabla_{\alpha}^2}{M_{\alpha}}}_{\hat{T}_n} \underbrace{-\frac{\hbar^2}{2m_e}\sum_{i}\nabla_{i}^2}_{\hat{T}_e} + \underbrace{\underbrace{\sum_{\alpha>\beta}\frac{e^2 Z_{\alpha} Z_{\beta}}{4\pi\epsilon_0 R_{\alpha\beta}}}_{\hat{V}_{nn}} \underbrace{-\sum_{\alpha,i}\frac{e^2 Z_{\alpha}}{4\pi\epsilon_0 R_{\alpha i}}}_{\hat{V}_{en}} + \underbrace{\sum_{i>j}\frac{e^2}{4\pi\epsilon_0 r_{ij}}}_{\hat{V}_{ee}}}_{\hat{V}}$$

Wave function methods

$$E_e = \frac{\langle \psi_e \mid \hat{H}_e \mid \psi_e \rangle}{\langle \psi_e \mid \psi_e \rangle} \qquad \phi_i = \sum_{\mu=1}^{N} c_{\mu i}\chi_{\mu}$$

$$\psi_e = \frac{1}{\sqrt{N!}}\begin{vmatrix} \phi_1(1) & \phi_2(1) & \cdots & \phi_N(1) \\ \phi_1(2) & \phi_2(2) & \cdots & \phi_N(2) \\ \cdots & \cdots & \cdots & \cdots \\ \phi_1(N) & \phi_2(N) & \cdots & \phi_N(N) \end{vmatrix}$$

density functional methods

$$\left(-\frac{1}{2}\nabla^2 + v_{\text{eff}}(r)\right)\phi_i(r) = \varepsilon_i\phi_i(r) \qquad \rho(r) = \sum_{i}^{N}|\phi_i(r)|^2$$

$$E[\rho] = T_s[\rho] + \int dr\, v_{\text{ext}}(r)\rho(r) + V_H[\rho] + E_{\text{xc}}[\rho]$$

$$v_{\text{eff}}(r) = v_{\text{ext}}(r) + \int \frac{\rho(r')}{|r - r'|}dr' + \frac{\delta E_{\text{xc}}[\rho]}{\delta\rho(r)}$$

hybrid DFT - why?

- wavefunction methods are expensive
  - could be O(N^7) or worse
- density methods are cheap
  - O(N) - O(N^3)
- current density functionals miss important physics in some materials

Quantum ESPRESSO hybrid DFT

Quantum Many Body Problem

$$\hat{H} = \underbrace{-\frac{\hbar^2}{2}\sum_{\alpha}\frac{\nabla_{\alpha}^2}{M_{\alpha}}}_{\hat{T}_n} \underbrace{-\frac{\hbar^2}{2m_e}\sum_{i}\nabla_{i}^2}_{\hat{T}_e} + \underbrace{\sum_{\alpha>\beta}\frac{e^2 Z_{\alpha} Z_{\beta}}{4\pi\epsilon_0 R_{\alpha\beta}}}_{\hat{V}_{nn}} \underbrace{-\sum_{\alpha,i}\frac{e^2 Z_{\alpha}}{4\pi\epsilon_0 R_{\alpha i}}}_{\hat{V}_{en}} + \underbrace{\sum_{i>j}\frac{e^2}{4\pi\epsilon_0 r_{ij}}}_{\hat{V}_{ee}}$$

$\hat{V}$

Wave funct

tional methods

$$E_e = \frac{\langle \psi_e \mid \hat{H}_e \mid \psi_e \rangle}{\langle \psi_e \mid \psi_e \rangle}$$

$$\psi_e = \frac{1}{\sqrt{N!}}\begin{vmatrix} \phi_1(1) & \phi_2(1) & \cdots & \phi_N(1) \\ \phi_1(2) & \phi_2(2) & \cdots & \phi_N(2) \\ \cdots & \cdots & \cdots & \cdots \\ \phi_1(N) & \phi_2(N) & \cdots & \phi_N(N) \end{vmatrix}$$

$$\rho(r) = \sum_{i}^{N} |\phi_i(r)|^2$$

$$E[\rho] = T_s[\rho] + \int dr\, v_{\text{ext}}(r)\rho(\quad V_H[\rho] + E_{\text{xc}}[\rho]$$

$$v_{\text{eff}}(r) = v_{\text{ext}}(r) + \int \frac{\rho(r')}{|r - r'|} d' + \frac{\delta E_{\text{xc}}[\rho]}{\delta\rho(r)}$$

Many-body physics is "swept under the rug" here

hybrid DFT

Add a wavefunction based "exchange" term to the electron density based Hamiltonian

$$\hat{K}[\psi_i](\mathbf{r}) = -\sum_{j=1}^{n} \psi_j(\mathbf{r}) \int \mathrm{d}^3 r' \frac{\psi_j^*(\mathbf{r}')\psi_i(\mathbf{r}')}{|\mathbf{r}-\mathbf{r}'|}$$

hybrid DFT

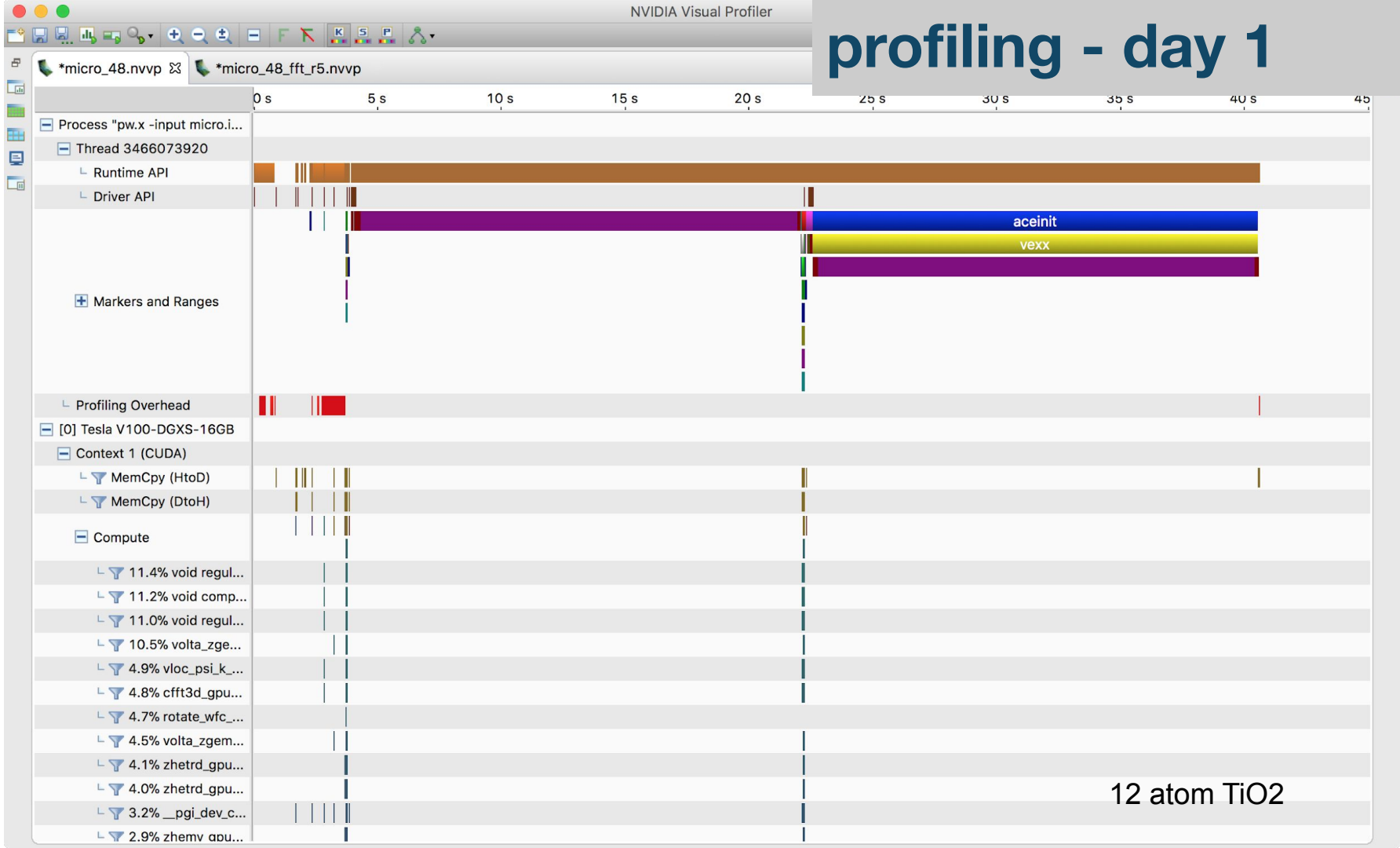
```
procedure vexx
  ...
  for i in 1:n do
    ...
    c(g)=FFT( 1./abs(r-rp) )
    for j in 1:n do
      rho(i,j,rp) = conjg(psi(j,rp))psi(i,rp)
      rho(i,j,g) = FFT( rho(i,j,rp) )
      v(i,j,g) = c(g) * rho(i,j,g)
      v(i,j,r) = invFFT( v(i,j,g) )
      Kpsi(i,r) += psi(j,r) * v(i,j,r)
```

hybrid DFT

- FFT's = cuFFT
- cuBLAS when possible
- the rest
  - !$cuf kernel do

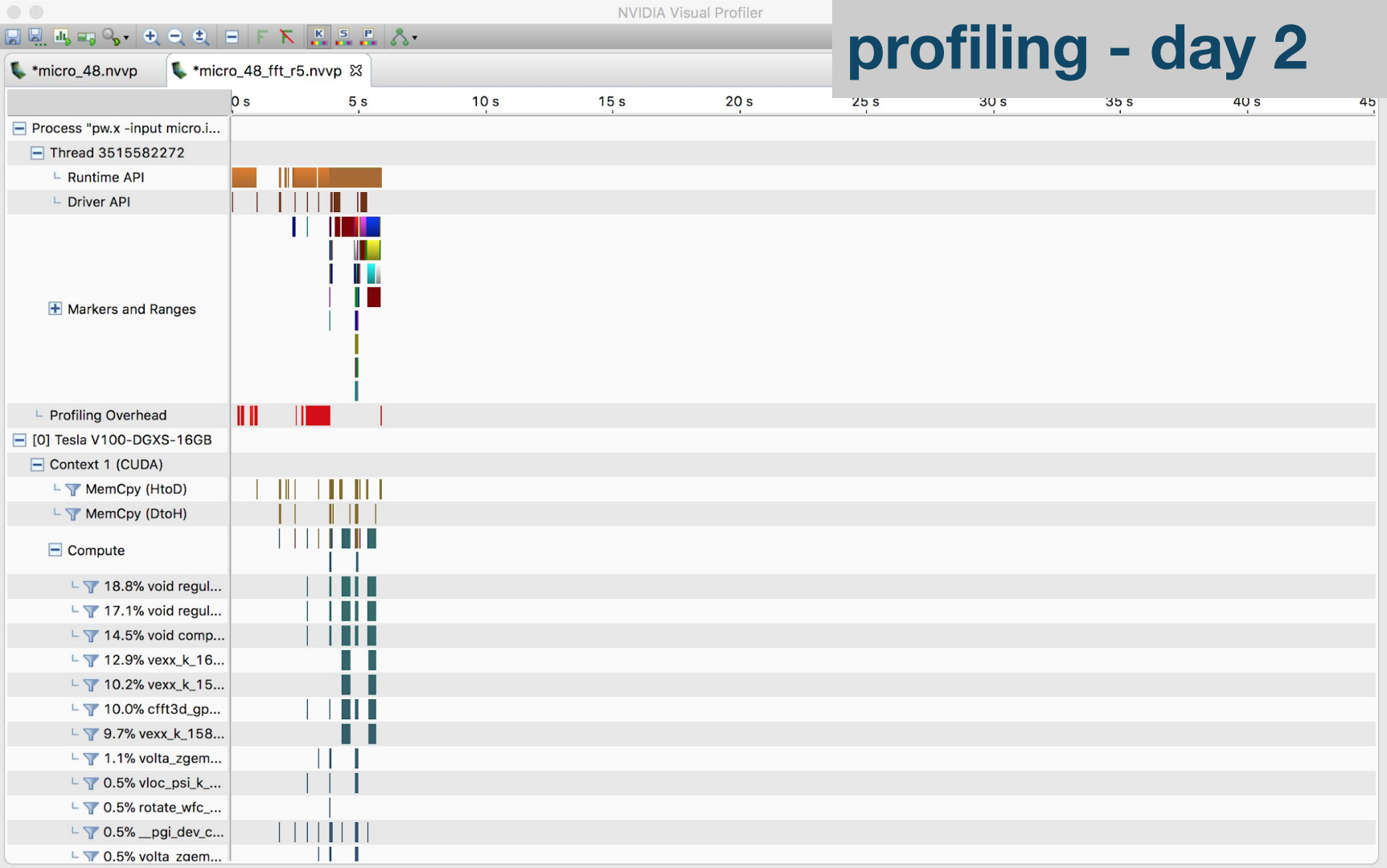
```
procedure vexx
  ...
  for i in 1:n do
    ...
    c(g)=FFT( 1./abs(r-rp) )
    for j in 1:n do
      rho(i,j,rp) = conjg(psi(j,rp))psi(i,rp)
      rho(i,j,g) = FFT( rho(i,j,rp) )
      v(i,j,g) = c(g) * rho(i,j,g)
      v(i,j,r) = invFFT( v(i,j,g) )
      Kpsi(i,r) += psi(j,r) * v(i,j,r)
```

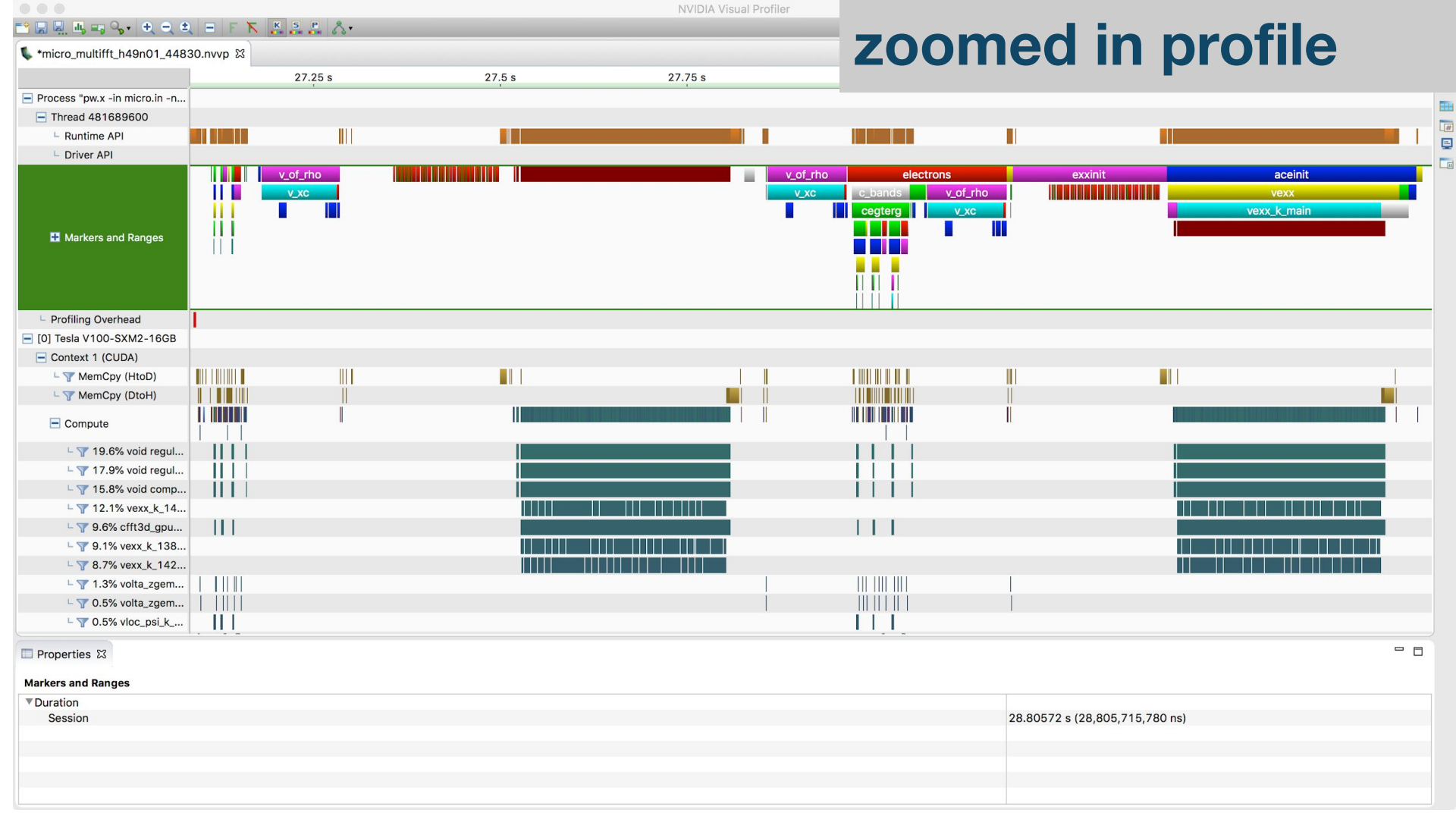
profiling - day 1



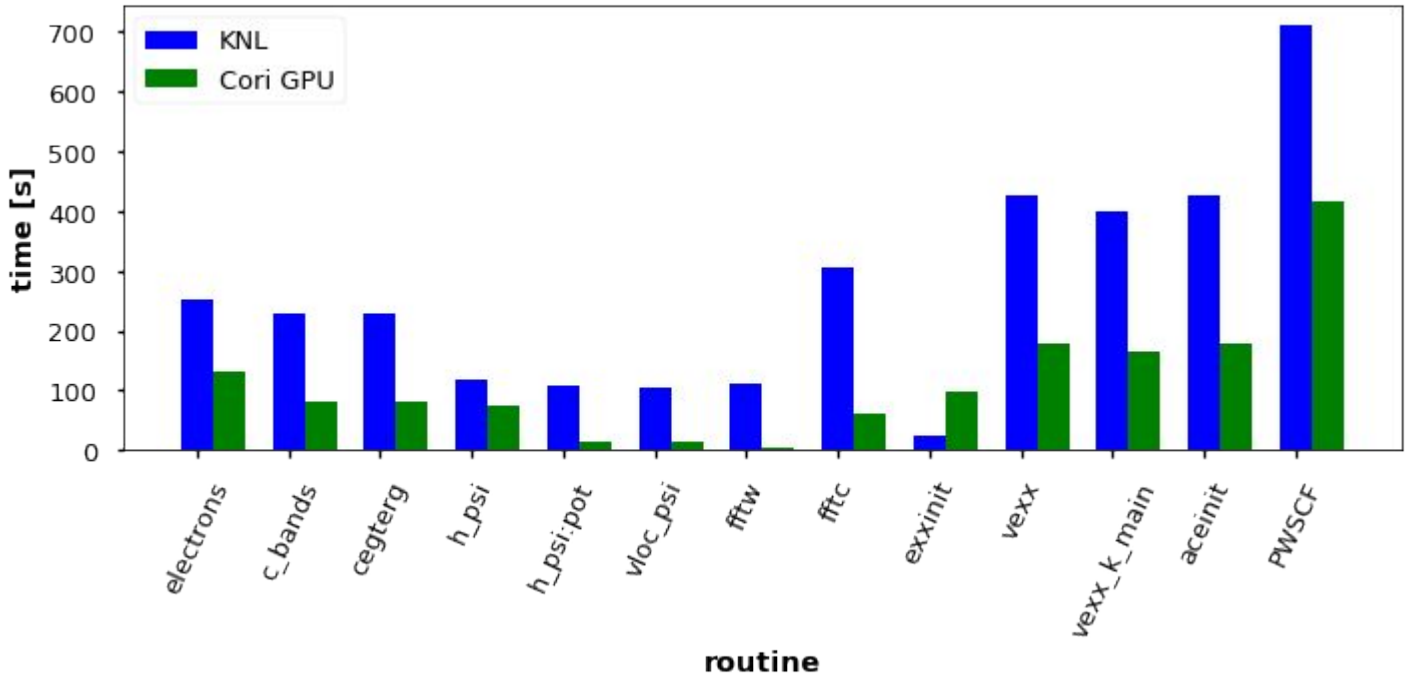
12 atom TiO2

profiling - day 2




NVIDIA Visual Profiler
zoomed in profile
*micro_multifft_h49n01_44830.nvvp
27.25 s
27.5 s
27.75 s
Process "pw.x -in micro.in -n...
Thread 481689600
Runtime API
Driver API
Markers and Ranges
v_of_rho
v_xc
v_of_rho
electrons
exxinit
aceinit
v_xc
c_bands
v_of_rho
vexx
cegterg
v_xc
vexx_k_main
Profiling Overhead
[0] Tesla V100-SXM2-16GB
Context 1 (CUDA)
MemCpy (HtoD)
MemCpy (DtoH)
Compute
19.6% void regul...
17.9% void regul...
15.8% void comp...
12.1% vexx_k_14...
9.6% cfft3d_gpu...
9.1% vexx_k_138...
8.7% vexx_k_142...
1.3% volta_zgem...
0.5% volta_zgem...
0.5% vloc_psi_k_...
Properties
Markers and Ranges
Duration
Session
28.80572 s (28,805,715,780 ns)

profiling + performance results



48 atoms TiO2

mixed precision

QE uses an inner + outer loop method for hybrid DFT

- outer SCF loop - iterate Fock operator
- inner SCF loop - Fock operator basis fixed

current techniques to improve performance

- Adaptively compressed exchange (ACE)
  - 3-6x speedup
  - only need correct answer on occupied manifold
- reduce FFT cutoff (smaller grid)
  - 1-8x speedup depending on accuracy tradeoff

mixed precision approach

- compute most FFT's in fp32 vs fp64
- switch to fp64 when "close" to convergence
- ~1.8x total application speedup

future work

- batched FFTs
- Fast Multipole Method approximate FFT
- dynamic FFT grid size

takeaways

- CUDA Fortran
  - pro: good porting experience
  - con: lack of portability/ compiler diversity
- iterate to convergence?
  - mixed precision worth a try
- nvprof interface is good for developers
- Consider integrating nvtx markers into your timing framework
- GPU enabled EXX in QE coming soon
  - more code paths coming soon (cp.x, etc)

Thank You