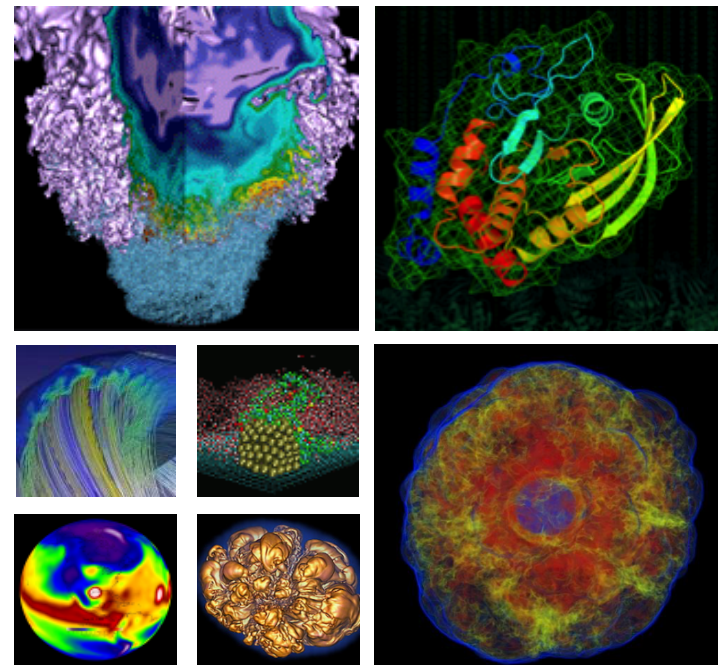
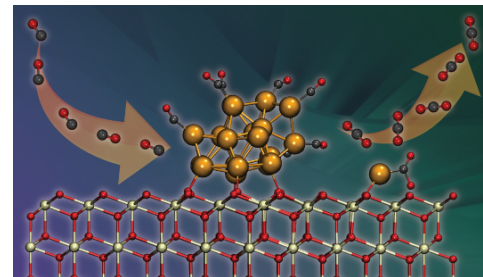
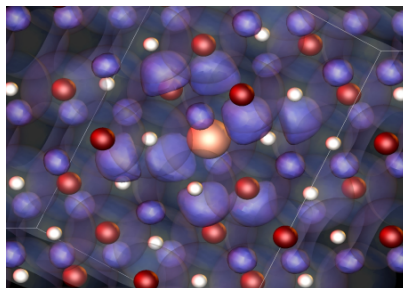
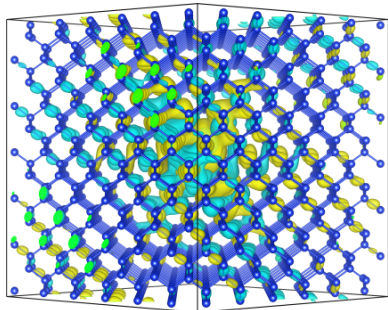


Case Study: Roofline Analysis on GPUs



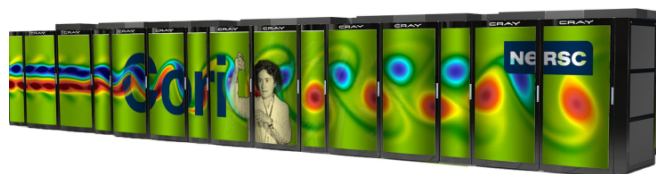
Charlene Yang
Lawrence Berkeley National Laboratory
SC 2019, Denver

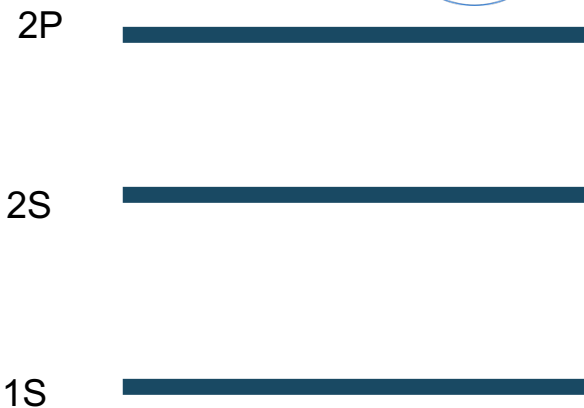
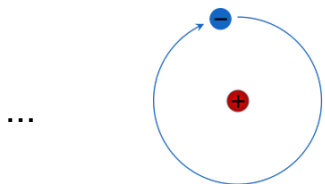
Material Science/Chemistry at Exascale



Mat. Sci & Chem apps like **VASP**, **Quantum ESPRESSO**, **NWChem**, **GAMESS**, **QMCPACK**, **BerkeleyGW**, and **CP2K** are some of the most heavily used apps at DOE facilities.

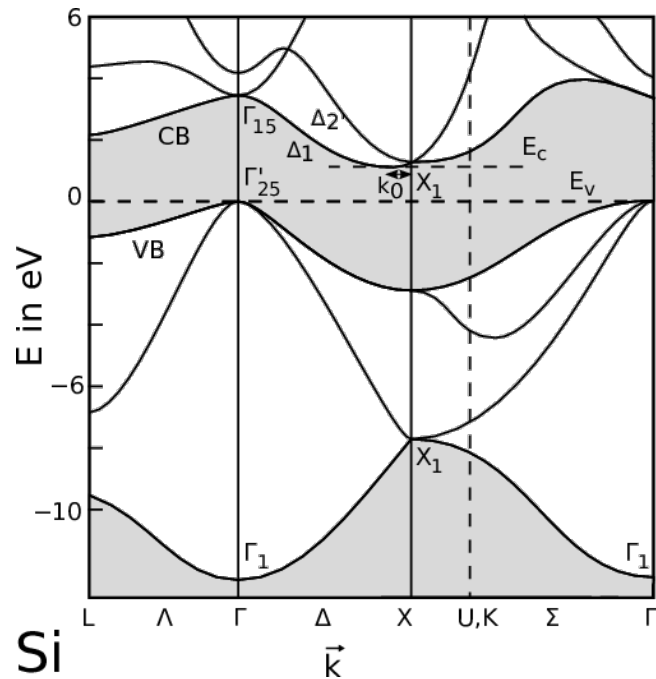
They are being used to design and understand the fundamental components of **Quantum Computers**, **Solar Cells**, **OLEDs**, **Batteries**, **Catalysts**, **Bio-Energy**, **Semiconductors**, **Sensors**, **Hydrogen Storage**, **Carbon Sequestration**





In crystalline materials, electrons are allowed to occupy only specific energy states or “bands” $E(k)$.

These bands (and gaps) determine material properties.



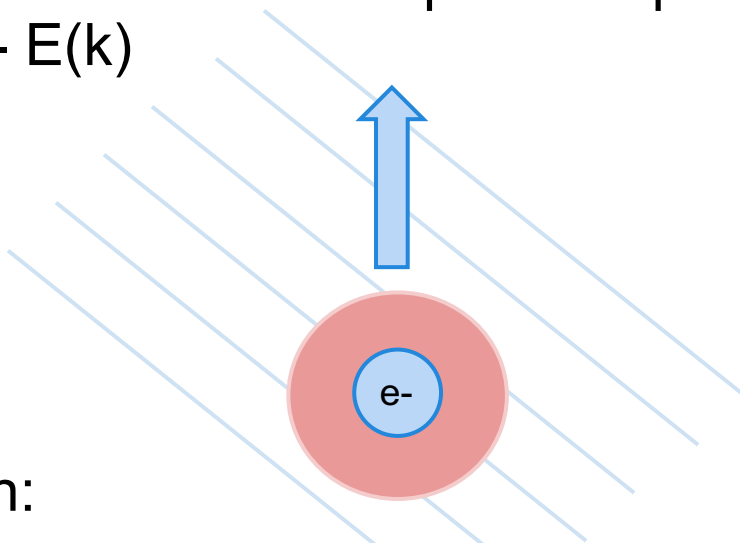
$$\left[-\frac{\nabla^2}{2} + V_{ion} + V_{Hartree} + V_{xc}^{KS} \right] \psi_{nk} = E_{nk}^{KS} \psi_{nk}$$

DFT “Kohn-Sham” Equations for $E(k)$

Equation of Motion for an Electron



DFT is deficient in computing $E(\mathbf{k})$ for many systems, where excitations take the form of an independent particle with effective complex energies - $E(\mathbf{k})$



The Dyson equation:

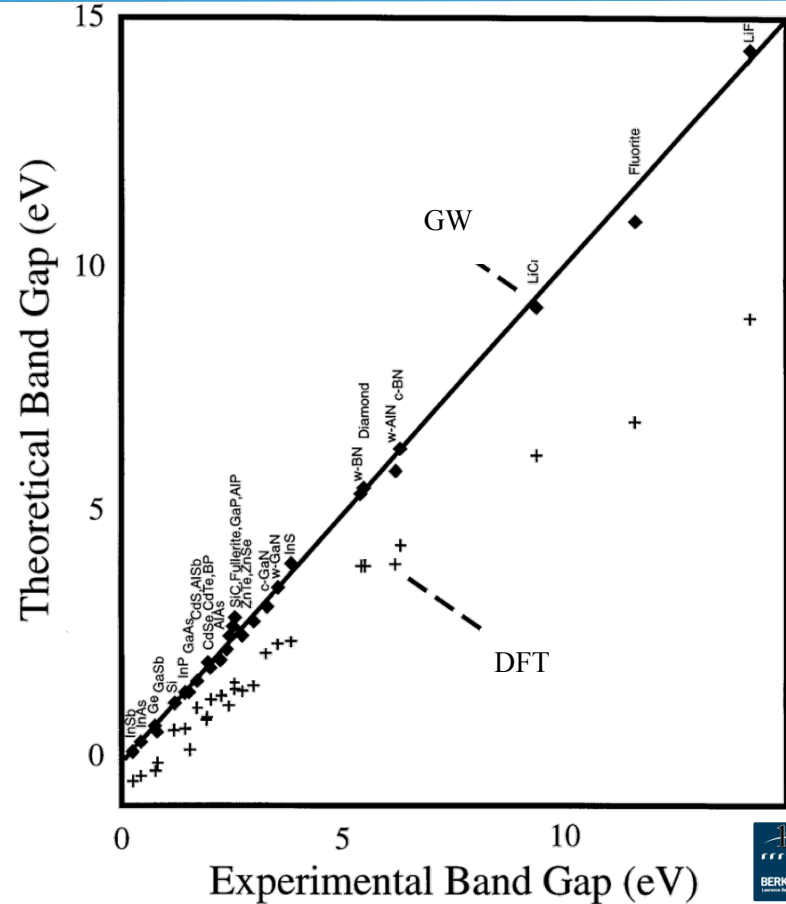
$$[E_{n\mathbf{k}} - H_0(\mathbf{r}) - V_H(\mathbf{r})] \Psi_{n\mathbf{k}}(\mathbf{r}) - \int \Sigma(\mathbf{r}, \mathbf{r}', E_{n,\mathbf{k}}) \Psi_{n\mathbf{k}}(\mathbf{r}') d\mathbf{r}' = 0$$

What is GW



The “GW” method is an **accurate** approach for simulating the “excited state” properties of materials.

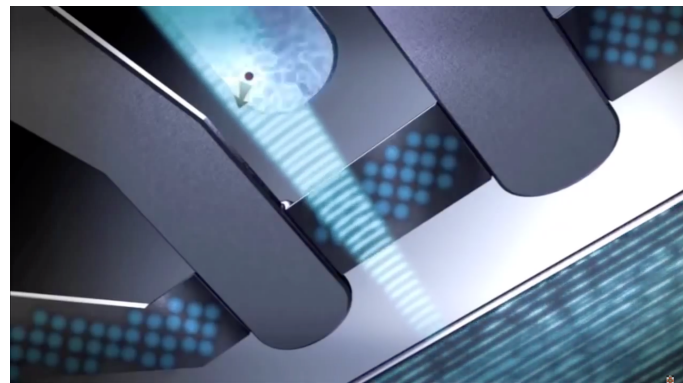
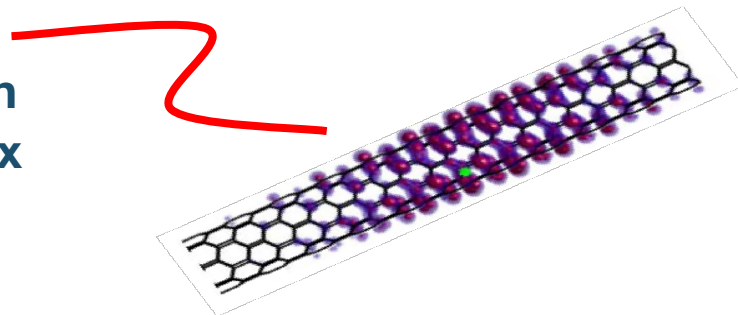
- What happens when you add or remove an electron from a system?
- How do electrons behave when you apply a voltage?
- How does the system respond to light or X-rays?



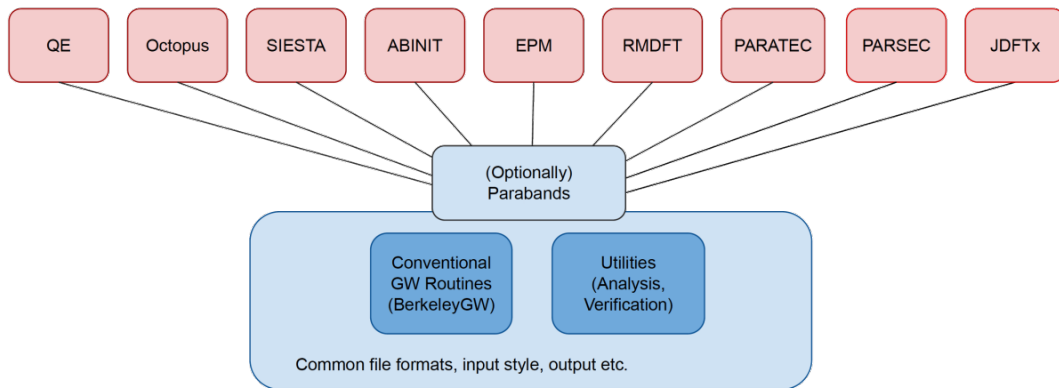
What is GW

Many-body effects extremely important in **Excited-State properties** of Complex Materials for Devices:

- Photovoltaics
- LEDs
- Quantum Computers
- Junctions / Interfaces
- Defect Energy Levels
-



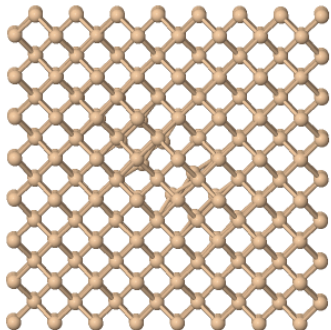
- A massively parallel package for GW calculations
- Sits on top of DFT codes
- Computational motifs
 - FFTs
 - Dense linear algebra
 - Large reductions



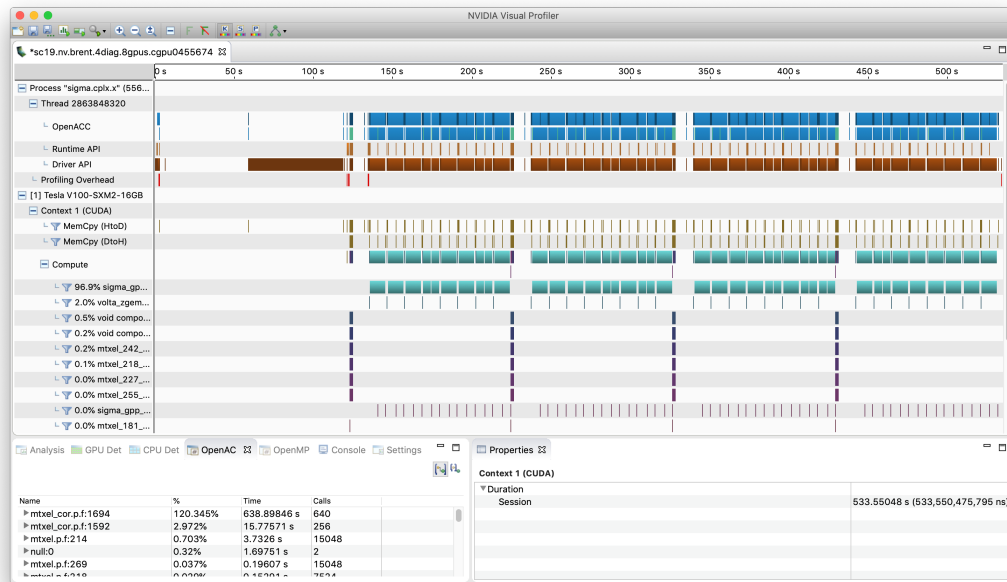
BerkeleyGW



- Large reductions in Sigma-GPP module
- Study of the divacancy effect in Silicon (prototype for qubits)



- Si510 system
- **>75% after optimization!**



- Calculate interacting electron energy

$$\Sigma_n = \sum_{n'} \sum_{\mathbf{G}\mathbf{G}'} M_{n'n}^*(-\mathbf{G}) M_{n'n}(\mathbf{G}') \frac{\Omega_{\mathbf{G}\mathbf{G}'}^2}{\tilde{\omega}_{\mathbf{G}\mathbf{G}'} (E - E_{n'} - \tilde{\omega}_{\mathbf{G}\mathbf{G}'})} v(\mathbf{G}')$$

- $\tilde{\omega}$ and Ω are complex double precision arrays over \mathbf{G}, \mathbf{G}' derived from the polarizability
- M are complex double-precision arrays representing transition probabilities
- E_n are DFT orbital energies
- E is an array of “response” energies.
- v is the coulomb interaction in plane-wave basis \mathbf{G}

Pseudo Code

```
do n1 = 1, nbands           n' e.g. 2763
  do igp = 1, ngpown        G' e.g. 6633
    do ig = 1, ncouls       G e.g. 26529
      do iw = 1, nw         E e.g. 3
        compute:
          1. mixed data types
             e.g. complex double, double, integer
          2. various memory access patterns
             e.g. (ig,igp) (ig,n1) (igp,n1) (iw,n1) (n1)
          3. complex number divisions
          4. nw is very small, will be unrolled

        reduction:
          1. complex numbers
          2. all top 3 loops, billions of iterations
```

V1. Naïve Implementation



- Collapse the first 3 loops to gain parallelism

```
!$ACC PARALLEL LOOP COLLAPSE(3) REDUCTION(+: )
do n1 = 1, nbands
  do igp = 1, ngpown
    do ig = 1, ncouls
      do iw = 1, nw          #unrolled
        compute and reduction
      end do
    end do
  end do
end do
```


| | TFLOPs | Time (sec) | TFLOP/s |
|--------------|--------|------------|---------|
| v1.collapse3 | 3.71 | 1.63 | 2.27 |

V2. More Compute Per Thread



- Move n' loop in, and collapse the first 2 loops

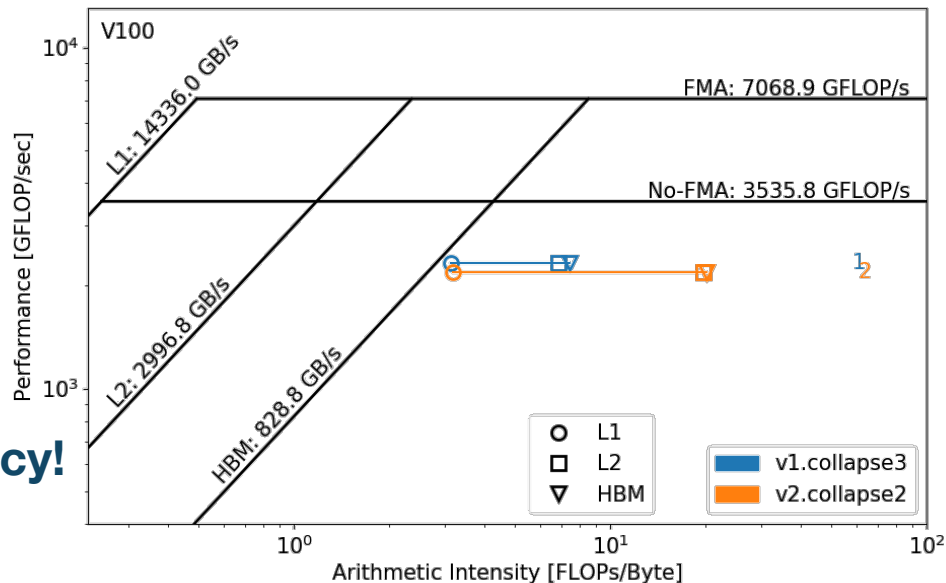
```
!$ACC PARALLEL LOOP COLLAPSE(2) REDUCTION(+: )  
do igp = 1, ngpown  
  do ig = 1, ncouls  
    do n1 = 1, nbands          #unrolled too!  
      do iw = 1, nw          #unrolled  
        compute and reduction
```

| | TFLOPs | Time | TFLOP/s |
|--------------|--------|--|---------|
| v1.collapse3 | 3.71 | 1.63  | 2.27 |
| v2.collapse2 | 3.71 | 1.73 | 2.15 |

V2. More Compute Per Thread



- L2/HBM AI increases!
- Register count at 186
 - Very low occupancy
 - 8 warps per SM
- Need more warps to hide latency!



Active Warps Per Scheduler [warp]
Eligible Warps Per Scheduler [warp]
Issued Warp Per Scheduler

2.00 Instructions Per Active Issue Slot [inst/cycle]
0.35 No Eligible [%]
0.30 One or More Eligible [%]

1
70.01
29.99

V3. Increase Threadblock Size



- Force threadblock size to be 512, instead of the default 128

```
!$ACC PARALLEL LOOP COLLAPSE(2) VECTOR_LENGTH(512) REDUCTION(+: )
```

- Register spills but performance may not be bad!

0 bytes stack frame, 0 bytes spill stores, 0 bytes spill loads

ptxas info : Used **186 registers**, 624 bytes cmem[0], 32 bytes cmem[2]

104 bytes stack frame, 188 bytes spill stores, 168 bytes spill loads

ptxas info : Used **128 registers**, 624 bytes cmem[0], 32 bytes cmem[2]

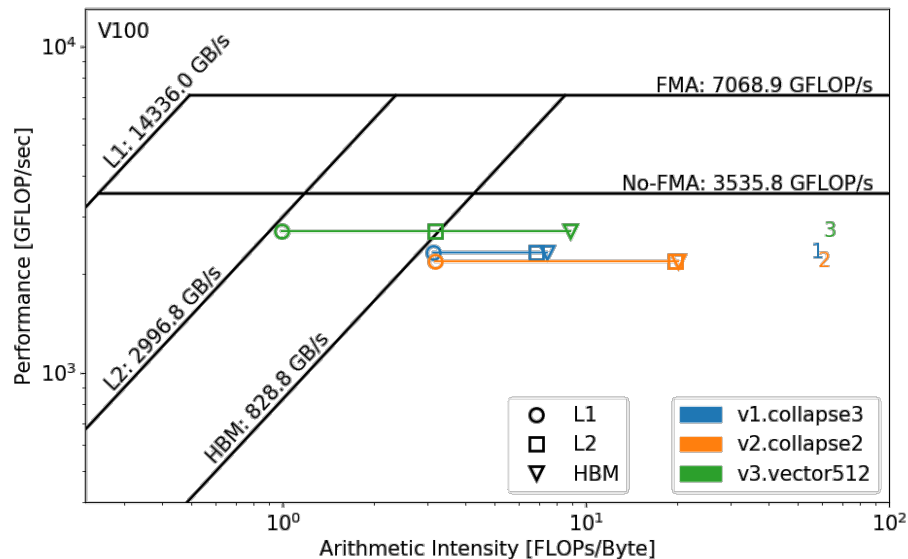


V3. Increase Threadblock Size



- More bandwidth bound now but latency hiding is successful!

| | TFLOPs | Time | TFLOP/s |
|--------------|--------|------|---------|
| v2.collapse2 | 3.71 | 1.73 | 2.15 |
| v3.vector512 | 3.71 | 1.40 | 2.65 |



| | | | |
|-------------------------------------|------|---|-------|
| Active Warps Per Scheduler [warp] | 4.00 | Instructions Per Active Issue Slot [inst/cycle] | 1 |
| Eligible Warps Per Scheduler [warp] | 0.67 | No Eligible [%] | 58.21 |
| Issued Warp Per Scheduler | 0.42 | One or More Eligible [%] | 41.79 |

V4. Reduce Branching



- Bring iw loop outside of the kernel

```
do iw = 1, nw                #reduce branching
!$ACC PARALLEL LOOP COLLAPSE(2) VECTOR_LENGTH(512) REDUCTION(+: )
do igp = 1, ngpown
  do ig = 1, ncouls
    do n1 = 1, nbands        #unrolled
      compute and reduction
```

- Fewer variables to be reduced -> lower register pressure

0 bytes stack frame, 0 bytes spill stores, 0 bytes spill loads

ptxas info : Used 122 registers, 600 bytes cmem[0], 32 bytes cmem[2]

V4. Reduce Branching



- Aggregated data for all kernels

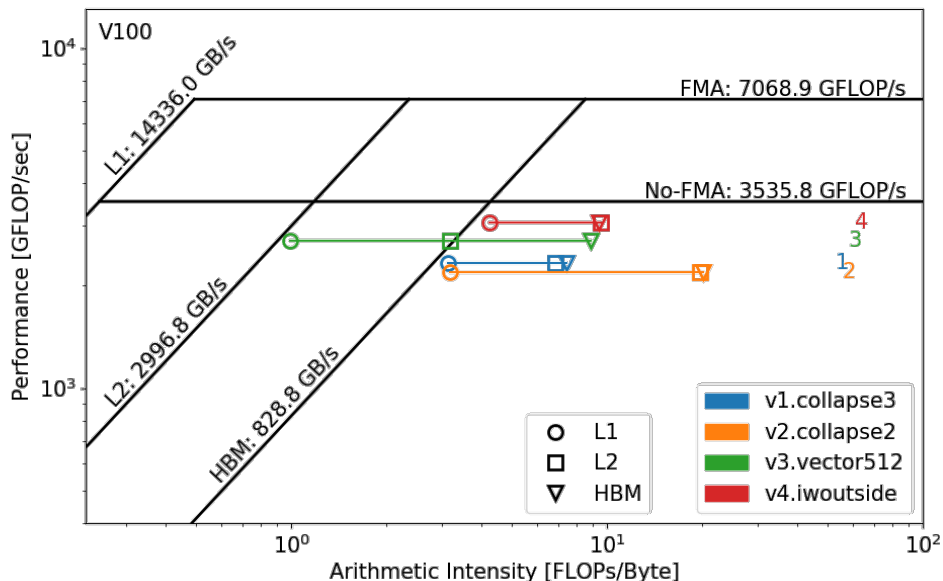
- BRA instruction count

14,278,897,053

5,975,051,812 x 2

➤ 16%

| | TFLOPs | Time | TFLOP/s |
|--------------|--------|------|---------|
| v3.vector512 | 3.71 | 1.40 | 2.65 |
| v4.iwoutside | 3.52 | 1.17 | 3.00 |

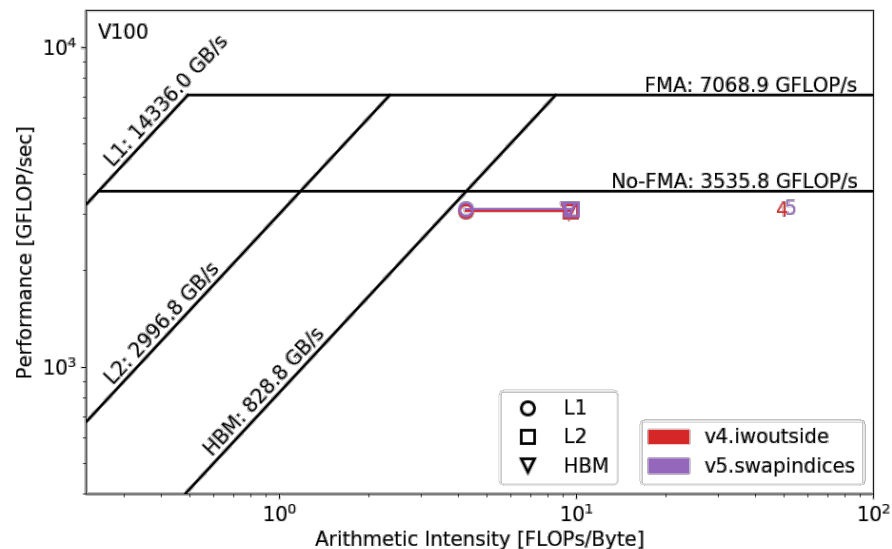


V5. Swap Indices



```
do iw = 1, nw
!$ACC PARALLEL LOOP
do igp = 1, ngpown
do ig = 1, ncouls
do n1 = 1, nbands
wx_array(iw,n1) to (n1,iw)
```

| | TFLOPs | Time | TFLOP/s |
|----------------|--------|------|---------|
| v4.iwoutside | 3.52 | 1.17 | 3.00 |
| v5.swapindices | 3.52 | 1.16 | 3.03 |

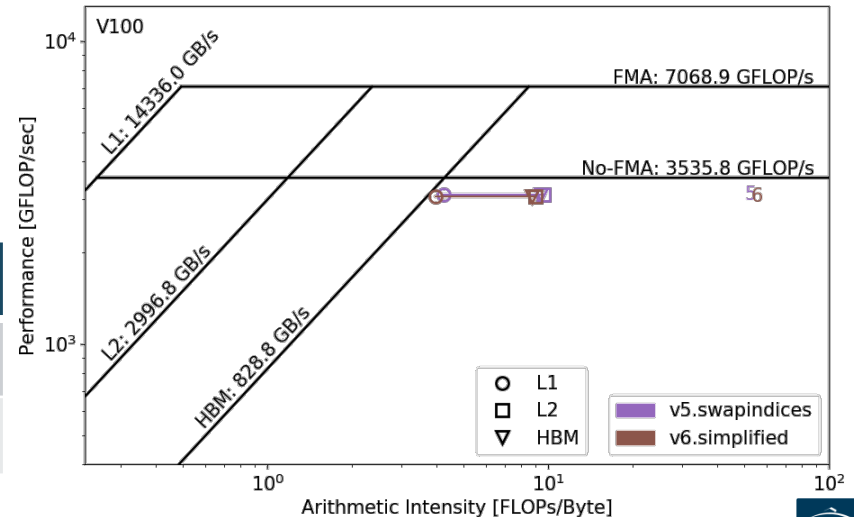


V6. Simplify Code



- Fewer instructions -> less work
 - Pull repeated instructions outside the loop
 - Use temporary variables to hold intermediate values for reuse
- Less branches -> better programming
 - 3 branches is more than 1 branch worse than 2 branches!

| | TFLOPs | Time | TFLOP/s |
|----------------|--------|------|---------|
| v5.swapindices | 3.52 | 1.16 | 3.03 |
| v6.simplify | 3.30 | 1.10 | 3.00 |

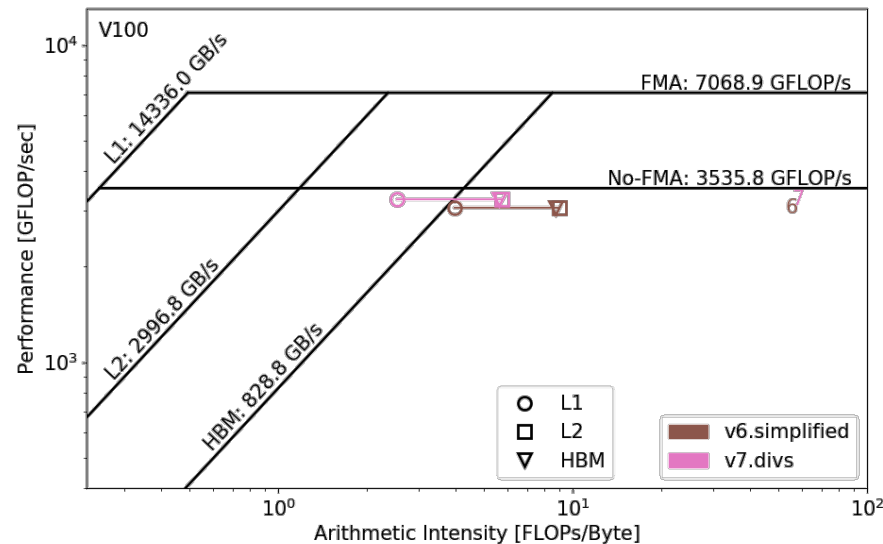


V7. Replace Divides



- Replace (complex) div. with (double) rcp. and (complex) mul.
- Lower instruction count: 40%
- More bandwidth bound now!

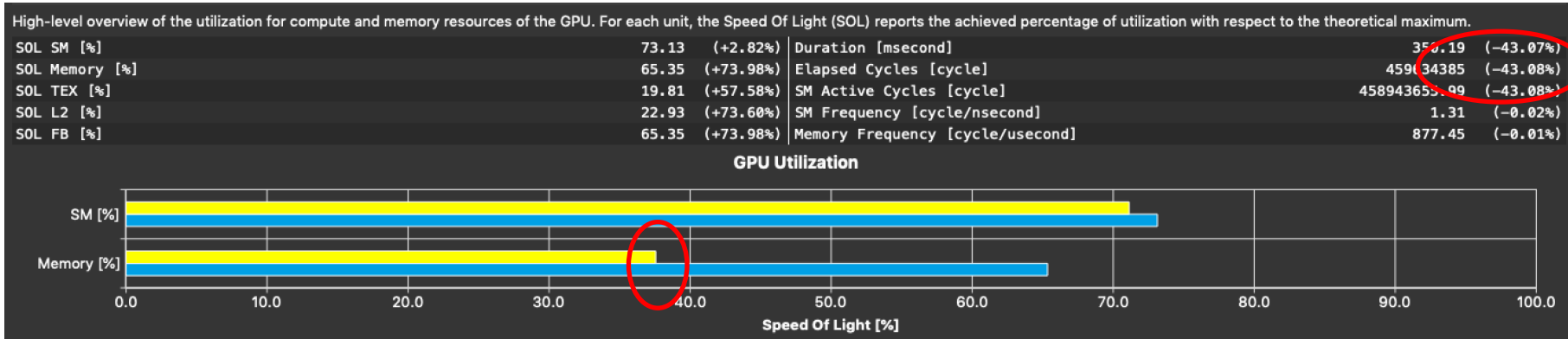
| | TFLOPs | Time | TFLOP/s |
|-------------|--------|------|---------|
| v6.simplify | 3.30 | 1.10 | 3.00 |
| v7.divs | 2.09 | 0.66 | 3.18 |



V7. Replace Divides



- Can be confirmed by Nsight Compute profiles



V8. Replace $\text{abs}(x)$ with x^{**2}



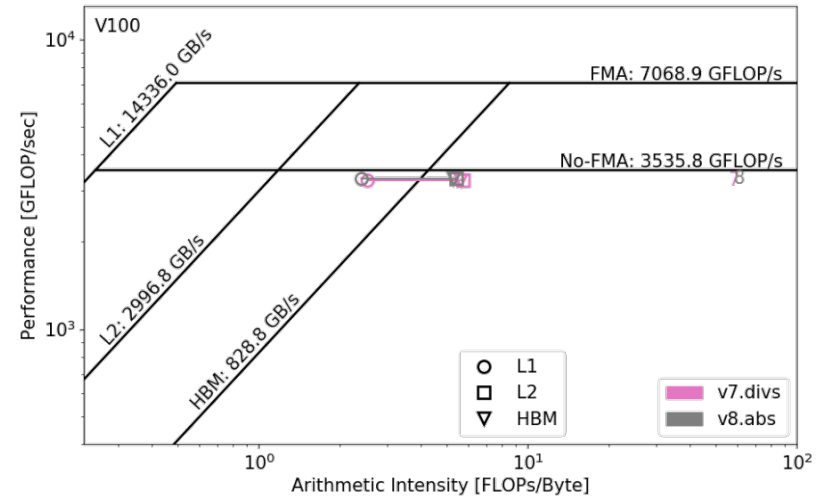
```
complex(DP) ssx  
if (abs(ssx) .le. ssxcutoff) then
```



```
real(DP) ssxpower  
if (ssxpower .le. ssxcutoff **2) then
```

- $\text{sqrt}(\text{complex})$ vs power of 2
- Causing pipeline to wait

| | TFLOPs | Time | TFLOP/s |
|---------|--------|------|---------|
| v7.divs | 2.09 | 0.66 | 3.18 |
| v8.abs | 1.99 | 0.62 | 3.23 |



V8. Replace $\text{abs}(x)$ with x^{**2}



Before:

- **Wait:** warp was stalled waiting on a fixed latency execution dependency

| # | Source | Sampling Data (All) | Sampling Data (Not Issued) | Instructions Executed | Predicated-On T |
|-----|--|---------------------|----------------------------|-----------------------|-----------------|
| 308 | rden = 1D0 / rden | 31,172 | 17,748 | 857,269,170 | |
| 309 | ssx = -Omega2 * conjg(cden) * rden * delw | 44,670 | 27,041 | 1,200,176,838 | |
| 310 | ! ssx = -Omega2 * delw / cden | 0 | 0 | | |
| 311 | endif | 0 | 0 | | |
| 312 | upd1 = 0.0d0 | 0 | 0 | | |
| 313 | if (abs(ssx) .le. ssxcutoff .or. wxt .ge. 0.0d0) then | 467,330 | 225,479 | 10,351,659,971 | |
| 314 | upd1 = vcoulxocc * ssx * matngmatmgrp | | 90,336 | 2,300,832,000 | |
| 315 | endif | | | | |
| 316 | ! if (abs(ssx) .gt. ssxcutoff .and. wxt .lt. 0.0d0) then | | | | |
| 317 | ! upd1 = 0.0d0 | | | | |
| 318 | else | | | | |
| 319 | ! upd1 = vcoulxocc * ssx * matngmatmgrp | | | | |
| 320 | ! end if | | | | |
| 321 | upd2 = vcoulx * sch * matngmatmgrp * 0.5d0 | 186,963 | 98,916 | 2,684,304,000 | |
| 322 | ssx_array_3 = ssx_array_3 + upd1 | 46,678 | 25,214 | 766,944,000 | |
| 323 | sch_array_3 = sch_array_3 + upd2 | 47,767 | 24,797 | 766,944,000 | |
| 324 | enddo ! loop over n1_loc | 0 | 0 | | |
| 325 | enddo ! loop over g | 125,175 | 109,449 | 149,675,164 | |

Total Sample Count: 467330
Dispatch Stall: 10296 (2.2%)
lmc Miss: 59 (0.0%)
Math Pipe Throttle: 92769 (19.9%)
Misc: 518 (0.1%)
No Instructions: 25849 (5.5%)
Not Selected: 42378 (9.1%)
Selected: 67526 (14.4%)
Short Scoreboard: 10195 (2.2%)
Wait: 217938 (46.6%)

V8. Replace abs(x) with x**2



After:

- Wait: 46.6% -> 23.7%

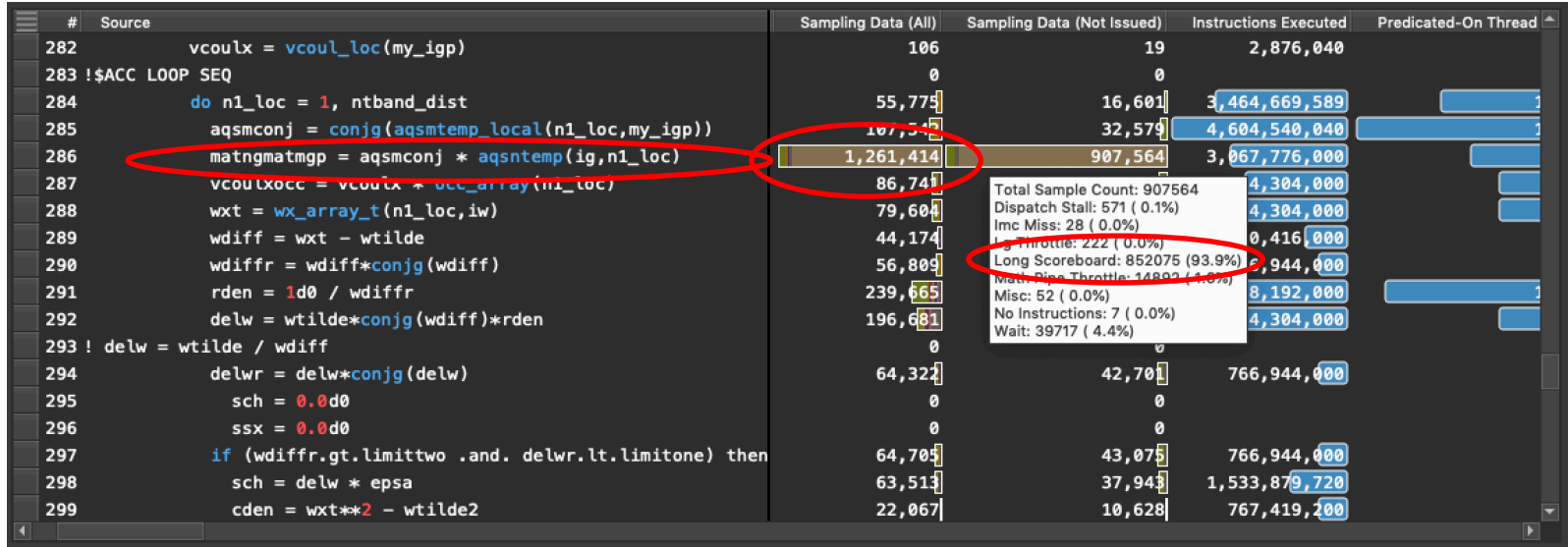
| # | Source | Sampling Data (All) | Sampling Data (Not Issued) | Instructions Executed | Predicated-On Thread Ir |
|-----|--|---------------------|----------------------------|-----------------------|-------------------------|
| 309 | wdiff = -Omega2 * conjg(cden) | 16,491 | 10,128 | 342,907,668 | |
| 310 | ssx = rden * delw * wdiff | 23,965 | 14,655 | 685,815,336 | |
| 311 | ! ssx = -Omega2 * delw / cden | 0 | 0 | | |
| 312 | endif | 0 | 0 | | |
| 313 | upd1 = 0.0d0 | 0 | 0 | | |
| 314 | rden = ssx * conjg(ssx) | 50,014 | 28,089 | 766,944,000 | |
| 315 | if (rden .le. ssxcutoff .or. wxt .ge. 0.0d0) then | 45,380 | 25,259 | 1,150,416,000 | |
| 316 | ! upd1 = vcoulxocc * ssx * matngmatmp | 111,372 | | 2,300,832,000 | |
| 317 | endif | | | | |
| 318 | ! if (abs(ssx) .gt. ssxcutoff .and. wxt .lt. 0.0d0) then | | | | |
| 319 | ! upd1 = 0.0d0 | | | | |
| 320 | ! else | | | | |
| 321 | ! upd1 = vcoulxocc * ssx * matngmatmp | | | | |
| 322 | ! end if | | | | |
| 323 | ! upd2 = vcoulx * sch * matngmatmp * 0.5d0 | 232,746 | 141,586 | 3,067,776,000 | |
| 324 | ! ssx_array_3 = ssx_array_3 + upd1 | 27,603 | 12,011 | 766,944,000 | |
| 325 | ! sch_array_3 = sch_array_3 + upd2 | 79,874 | 45,705 | 766,944,000 | |
| 326 | ! enddo ! loop over n1_loc | 0 | 0 | | |

Total Sample Count: 45380
Dispatch Stall: 929 (2.0%)
Math Pipe Throttle: 19231 (42.4%)
Misc: 37 (0.1%)
Not Selected: 6591 (14.5%)
Selected: 7853 (17.3%)
Wait: 10739 (23.7%)

V9. Cache Blocking



- Non-coalesced memory access for aqsn temp
- Causing Long Scoreboard Warp State
 - Warp stalled waiting for L1TEX (local, global, surface, tex) memory operation



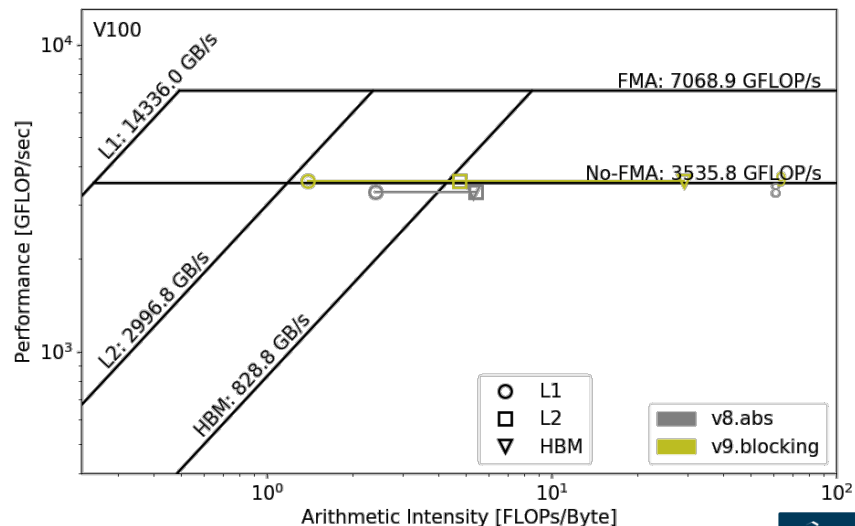
V9. Cache Blocking



- Break loops into chunks and reuse data across threadblocks
- Increase L2 hit rate

```
!$ACC LOOP GANG VECTOR
do ig_blk = 1, ig_blksize
!$ACC LOOP SEQ
  do ig = ig_blk, ncouls, ig_blksize
```

| | TFLOPs | Time | TFLOP/s |
|----------|--------|------|---------|
| v8.abs | 1.99 | 0.62 | 3.23 |
| v9.block | 2.00 | 0.57 | 3.50 |



V9. Cache Blocking



- Less coalescence and higher L2/L1 hit rate
- Not much more could be done; arrays of various dimensions



Memory Workload Analysis

Detailed analysis of the memory resources of the GPU. Memory can become a limiting factor for the overall kernel performance when fully utilizing the involved hardware units (Mem Busy), exhausting the available communication bandwidth between those units (Max Bandwidth), or by reaching the maximum throughput of issuing memory instructions (Mem Pipes Busy).

| | | | |
|----------------------------------|-------------------|--------------------|-----------------|
| Memory Throughput [Gbyte/second] | 131.85 (-78.99%) | Mem Busy [%] | 31.10 (+29.96%) |
| L1 Hit Rate [%] | 71.38 (+27.27%) | Max Bandwidth [%] | 28.44 (-59.32%) |
| L2 Hit Rate [%] | 88.89 (+3272.88%) | Mem Pipes Busy [%] | 12.84 (+59.44%) |

8 Steps to Optimize Sigma-GPP

1. Collapse n', G', and G loops
2. Bring n' loop in; collapse only G' and G
3. Adjust threadblock size
4. Reduce branching; pull iw loop outside
5. Swap indices to suite parallelisation
6. Simplify code
7. Replace div. with rcp. and mul.
8. Replace abs with power of 2
9. Cache blocking

```
!$ACC PARALLEL LOOP REDUCTION(+: )
do n1 = 1, nbands
  do igp = 1, ngpown
    do ig = 1, ncouls
      do iw = 1, nw
        compute and reduction
      end do
    end do
  end do
end do
```

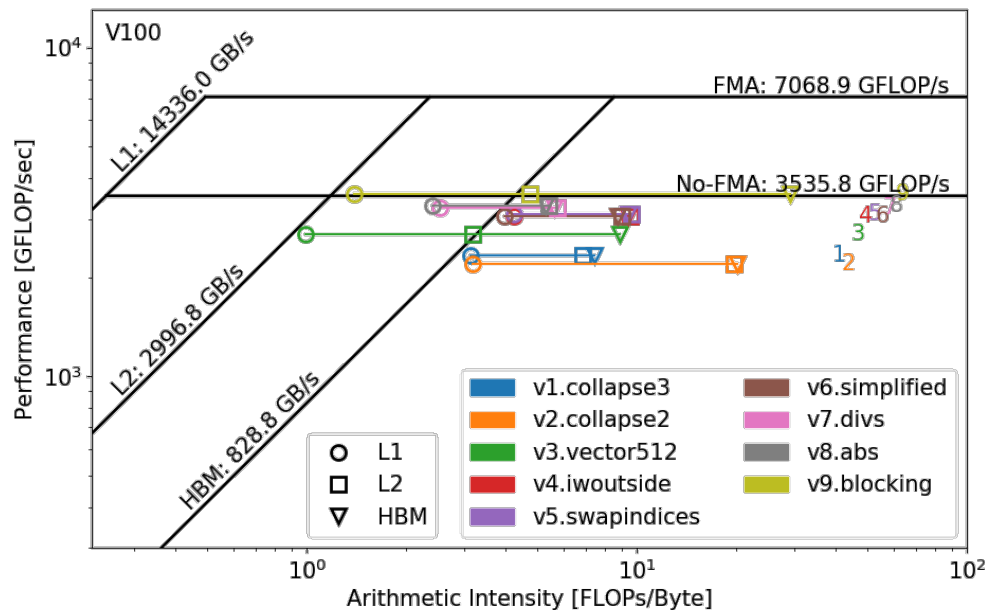
| | TFLOPs | Time | TFLOP/s |
|--------------|--------|------|---------|
| v1.collapse3 | 3.71 | 1.63 | 2.27 |
| v9.block | 2.00 | 0.57 | 3.50 |

3x !!

Summary



- Code is still bandwidth and latency bound
 - shared memory
 - lower register count
 - improve FMA ratio
- Together with profilers, Roofline provides the complete solution for your performance analysis and optimization needs!



Acknowledgement



- This material is based upon work supported by the Advanced Scientific Computing Research Program in the U.S. Department of Energy, Office of Science, under Award Number DE-AC02-05CH11231.
- This material is based upon work supported by the DOE RAPIDS SciDAC Institute.
- This research used resources of the National Energy Research Scientific Computing Center (NERSC), which is supported by the Office of Science of the U.S. Department of Energy under contract DE-AC02-05CH11231.