Roofline Analysis at NERSC

Accelerating Large-Scale Excited-State Studies in Materials Science

International Conference for High Performance Computing, Networking, Storage, and Analysis 2020 (SC'20)

Charlene Yang
Application Performance Specialist
Nov 10, 2020

Outline

- *M. Del Ben, C. Yang, Z. Li, F. H. da Jornada, S. G. Louie and J. Deslippe, "Accelerating Large-Scale Excited-State GW Calculations on Leadership HPC Systems", ACM Gordon Bell Finalist 2020*

- Performance: **105.9 PFLOP/s** in double precision on full Summit
- Optimization: Roofline analysis, Nsight Compute/Systems

COMPUTATIONAL RESEARCH DIVISION

Berkeley UNIVERSITY OF CALIFORNIA

C2SEPEM

Center for Computational Study of Excited-State Phenomena in Energy Materials

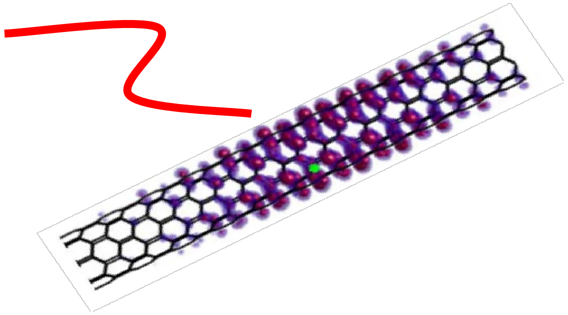
Stanford University

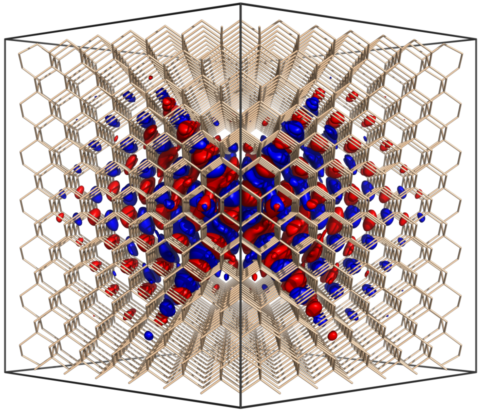
BERKELEY LAB Bringing Science Solutions to the World

U.S. DEPARTMENT OF ENERGY Office of Science

GW Calculations

- *G* for Green's function, *W* for screened Coulomb interaction
- Used to study excited-state properties of electronic structures
- More accurate than DFT (density-functional theory) methods





BerkeleyGW

- A massively parallel package for GW calculations
- Sits on top of DFT codes such as Quantum Espresso
- 4 modules: Epsilon, **Sigma**, Kernel, and Absorption

- Computational characteristics:
  - dense linear algebra
  - FFTs
  - large **low-rank reductions**
  - eigenvalue problems
  - matrix inversion

BerkeleyGW

https://berkeleygw.org

General Plasmon Pole (GPP) Kernel

- Sigma module calculates self-energy matrix elements

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

- GPP kernel
  - dominating kernel in Sigma
  - 1000s of invocations per GPU

```
for band = 1, nbands     # O(1,000)
  for igp = 1, ngpown    # O(10,000)
    for ig = 1, ncouls   # O(100,000)
      for iw = 1, nw     # small, <10
       complex arithmetic, divs, sqrts…
reduction to arrays[iw]
```

Benchmark System

Parameters	Si-214	Si-510	Si-998	SiC-998	Si-2742
N_{spin}	1	1	1	2 (↑/↓)	1
N_G^{ψ}	31,463	74,653	145,837	422,789	363,477
N_G	11,075	26,529	51,627	149,397	141,505
N_b	6,397	15,045	29,346	16,153	80,694
N_v	428	1,020	1,996	1,997/1,995	5,484
N_c	5,969	14,025	27,350	14,156/14,158	75,210
N_{Σ}	Variable, up to 128 per spin				
Epsilon PFLOPs	2.5	80.5	1164	10,091	66,070
Epsilon Memory (TB)	0.45	6.07	45.1	135	934
Sigma PFLOPs	0.127	1.71	12.6	58.2	260.7
Sigma Memory (GB)	6.19	34.3	133.8	791.4	1006

Silicon or silicon carbide systems with divacancy defects used for prototyping quantum information devices

- Si-2742 with ~11k electrons
- For each quasi-particle
  - compute: 260 PFLOPs
  - memory: 1 TB
- Our Gordon Bell results:
  - 256 quasi-particles
- This talk:
  - **1 quasi-particle**
  - **108 GPUs on Summit**

Computational Characteristics

- Tensor contraction
  - low arithmetic intensity, bandwidth bound
- Complex double data type, long kernel
  - high register and shared memory usage, low occupancy
- Mixed memory access pattern
  - multiple 2D/3D arrays
  - hard to ensure coalesced or contiguous access for all

Computational Characteristics

- Long-latency instructions
  - complex number arithmetic, divides, square roots
- FMA ratio at 51%
  - measured with Nsight Compute, FMA/total FP64 instructions
- Low-rank global reductions
  - low effective usage of threads
  - warp level (bisection), thread block level (thread 0 in each warp)
  - synchronization barriers

Optimization of GPP

	Optimization Path	Time (s)	Speedup
v1	baseline *with retrospectively optimized parameters	1557	1
v2	replace divides with reciprocals	1389	1.12x
v3	replace square roots with power of 2	1061	1.47x
v4	replace divides and square roots	943	1.65x
v5	loop reordering to gain arithmetic intensity	671	2.32x
v6	further increase occupancy	600	2.60x
v7	cache blocking	571	2.73x
v8	cache more arrays in shared memory	549	2.84x

Optimization of GPP

	Optimization Path	Time (s)	Speedup
v1	baseline *with retrospectively optimized parameters	1557	1
v2	replace divides with reciprocals	1389	1.12x
v3	replace square roots with power of 2	1061	1.47x
v4	replace divides and square roots	943	1.65x
v5	loop reordering to gain arithmetic intensity	671	2.32x
v6	further increase occupancy	600	2.60x
v7	cache blocking	571	2.73x
v8	cache more arrays in shared memory	549	2.84x

Optimization of GPP

	Optimization Path	Time (s)	Speedup
v1	baseline *with retrospectively optimized parameters	1557	1
v2	replace divides with reciprocals	1389	1.12x
v3	replace square roots with power of 2	1061	1.47x
v4	replace divides and square roots	943	1.65x
v5	loop reordering to gain arithmetic intensity	671	2.32x
v6	further increase occupancy	600	2.60x
v7	cache blocking	571	2.73x
v8	cache more arrays in shared memory	549	2.84x

Optimization of GPP

	Optimization Path	Time (s)	Speedup
v1	baseline *with retrospectively optimized parameters	1557	1
v2	replace divides with reciprocals	1389	1.12x
v3	replace square roots with power of 2	1061	1.47x
v4	replace divides and square roots	943	1.65x
v5	loop reordering to gain arithmetic intensity	671	2.32x
v6	further increase occupancy	600	2.60x
v7	cache blocking	571	2.73x
v8	cache more arrays in shared memory	549	2.84x

Reduce Execution Latency (v1 - v4)

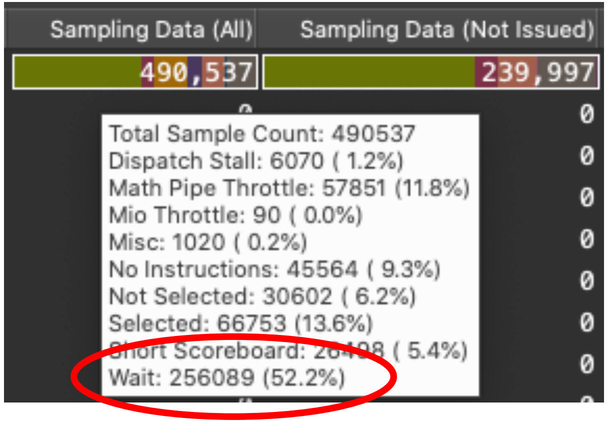
Reduce Execution Latency (v1 - v4)

- Replace complex divides by reciprocals

  $$(a+bi)/(c+di) = ((ac+bd)+(bc-ad)i)/(c^2+d^2)$$

- Replace $abs(a+bi)>c$ by $(a^2+b^2)>c^2$

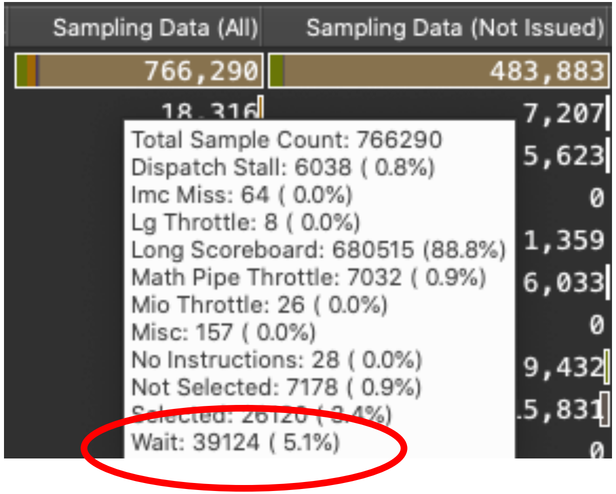
- High warp stalls:
  - *waiting on a fixed latency execution dependency*

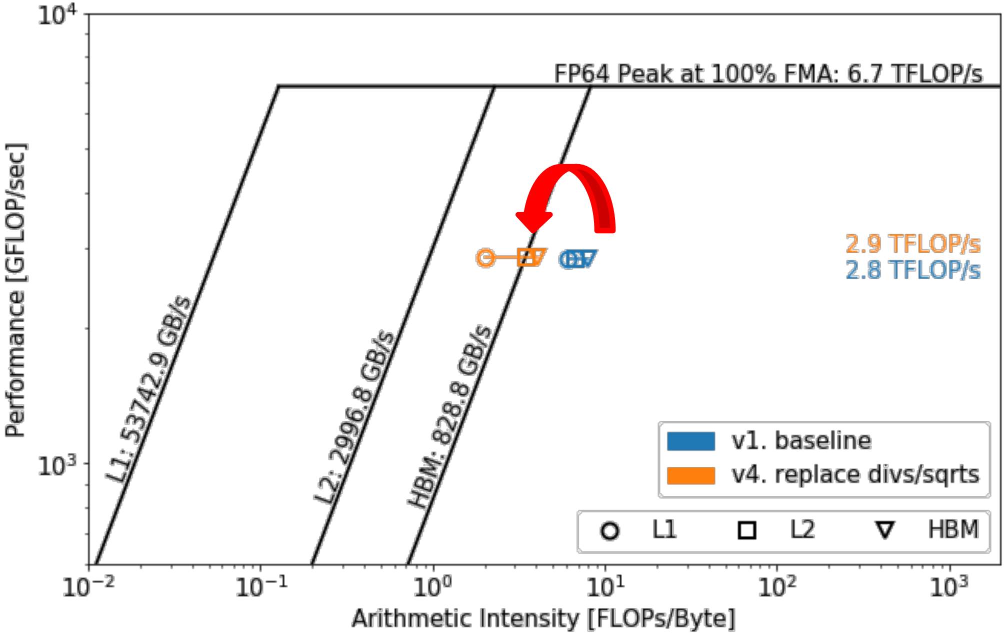


https://docs.nvidia.com/nsight-compute/ProfilingGuide/index.html#statistical-sampler

Reduce Execution Latency (v1 - v4)

- After this optimization:





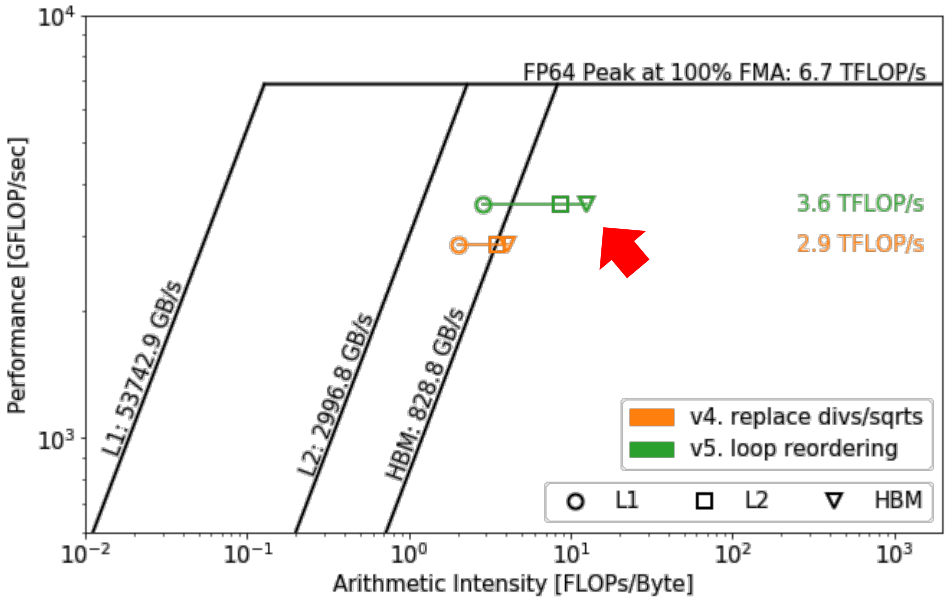
Gain Arithmetic Intensity (v5)

Gain Arithmetic Intensity (v5)

```
# before (v4)
for band = 1, nbands    # O(1,000)
  for igp = 1, ngpown   # O(10,000)
    for ig = 1, ncouls # O(100,000) # threads
    …
# after (v5)
for igp = 1, ngpown   # O(10,000)
  for ig = 1, ncouls # O(100,000)    # threads
    for band = 1, nbands    # O(1,000)
    …
```

Gain Arithmetic Intensity (v5)

- Less data movement -> higher arithmetic intensity



V100 GPU

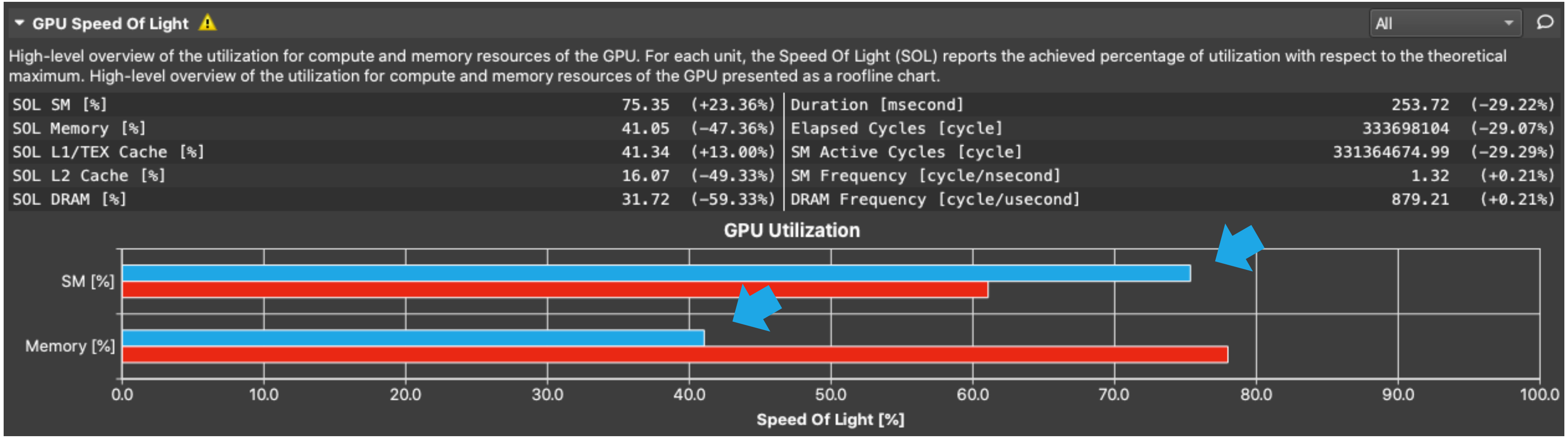
- 6.7 TFLOP/s vs 7.8 TFLOP/s
- 1312 MHz vs 1530 MHz

$$80 \times 32 \times 2 \times 1312e6 = 6.7 \text{ TFLOP/s}$$

Confirmed with Nsight Compute!

Gain Arithmetic Intensity (v5)

- Less data movement -> higher arithmetic intensity
- Increased SM utilization and decreased memory utilization



Hide Memory Latency (v6 - v8)

More Compute Resources

- GPU computing is all about **latency hiding** !

- Adjust kernel launch parameters

- Experiment with `maxregcount`
  - trade register spill for higher occupancy
  - do this when the code is stable (register usage might change)

V100 GPU:

- 88 registers per threads
  -> 16 warps per SM
- 84 registers per threads
  -> 24 warps per SM

More Compute Resources (v6)

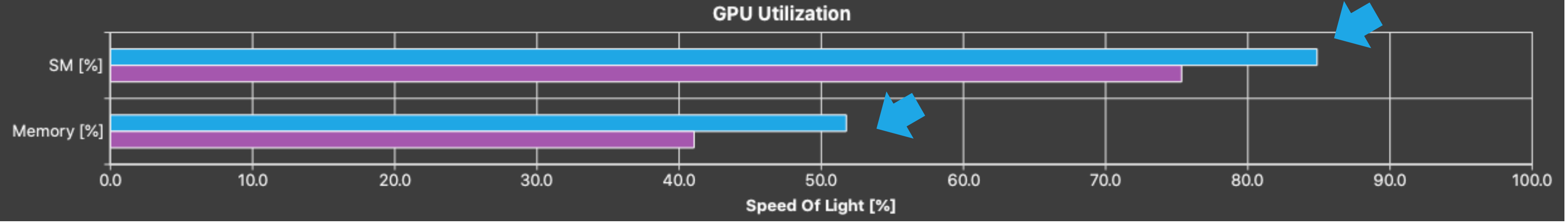
- Both SM and memory utilization are increased !

GPU Speed Of Light

All

High-level overview of the utilization for compute and memory resources of the GPU. For each unit, the Speed Of Light (SOL) reports the achieved percentage of utilization with respect to the theoretical maximum. High-level overview of the utilization for compute and memory resources of the GPU presented as a roofline chart.

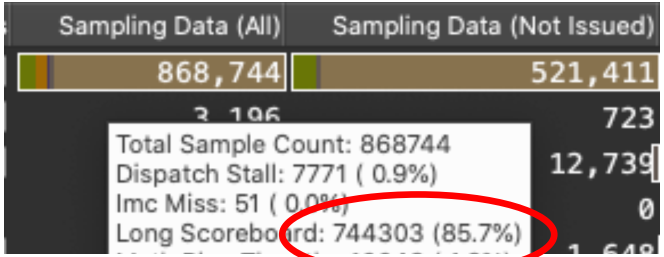
SOL SM [%]	84.87	(+12.64%)	Duration [msecond]	225.81	(-11.00%)
SOL Memory [%]	51.76	(+26.08%)	Elapsed Cycles [cycle]	296275442	(-11.21%)
SOL L1/TEX Cache [%]	45.97	(+11.20%)	SM Active Cycles [cycle]	295555702.77	(-10.81%)
SOL L2 Cache [%]	20.19	(+25.67%)	SM Frequency [cycle/nsecond]	1.31	(-0.25%)
SOL DRAM [%]	51.76	(+63.19%)	DRAM Frequency [cycle/usecond]	877.12	(-0.24%)



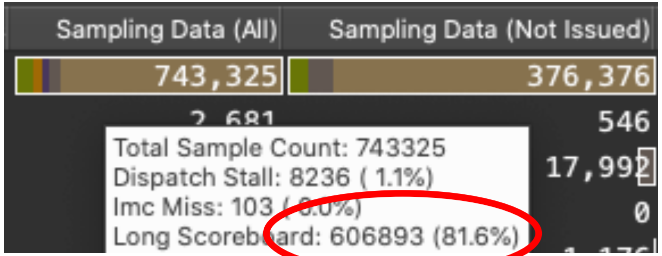
Reduce Memory Latency

- Squeezing more threads onto the SM has helped
- But can we do more?
    - We have a lot of 'long scoreboard' warp stalls

v5



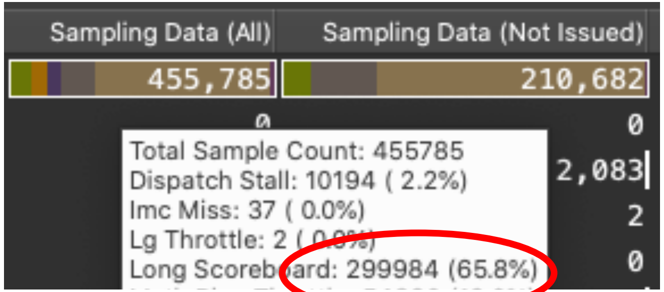
v6



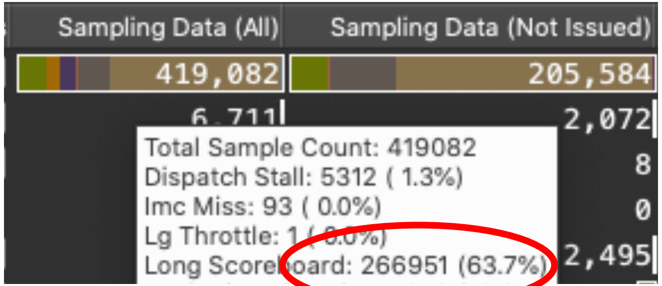
Reduce Memory Latency (v7 - v8)

- v7. cache blocking
    - careful design and selection of block sizes
- v8. move more arrays into shared memory
    - limited resource, only store the most impactful arrays

v7



v8



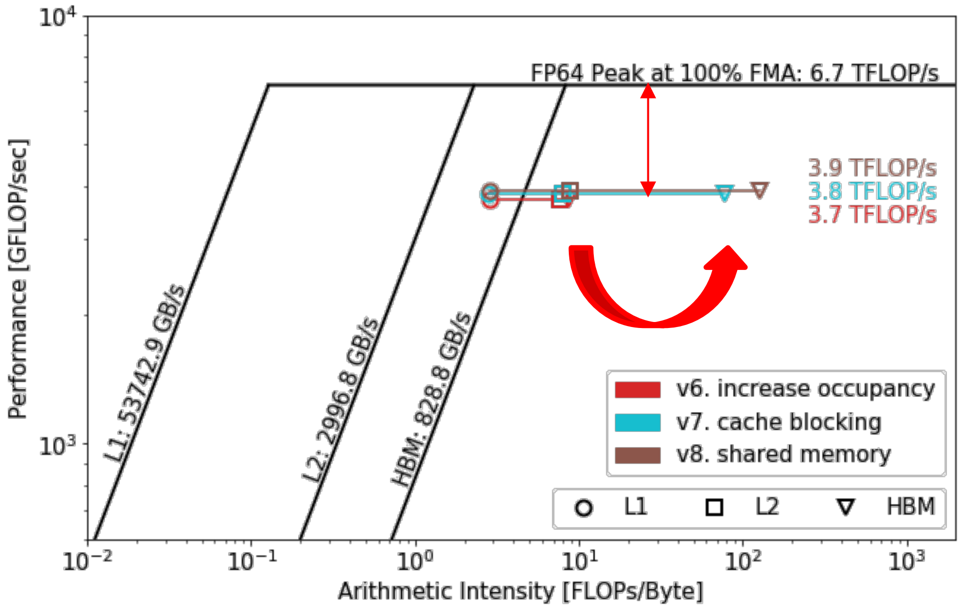
Reduce Memory Latency (v7 - v8)

- HBM data movement has dramatically reduced !

v5	Memory Throughput [Gbyte/second]	285.57
	L1/TEX Hit Rate [%]	67.07
	L2 Hit Rate [%]	35.37
v6	Memory Throughput [Gbyte/second]	464.92
	L1/TEX Hit Rate [%]	63.23
	L2 Hit Rate [%]	14.89
v7	Memory Throughput [Gbyte/second]	46.65
	L1/TEX Hit Rate [%]	65.00
	L2 Hit Rate [%]	89.99
v8	Memory Throughput [Gbyte/second]	32.96
	L1/TEX Hit Rate [%]	67.58
	L2 Hit Rate [%]	92.58

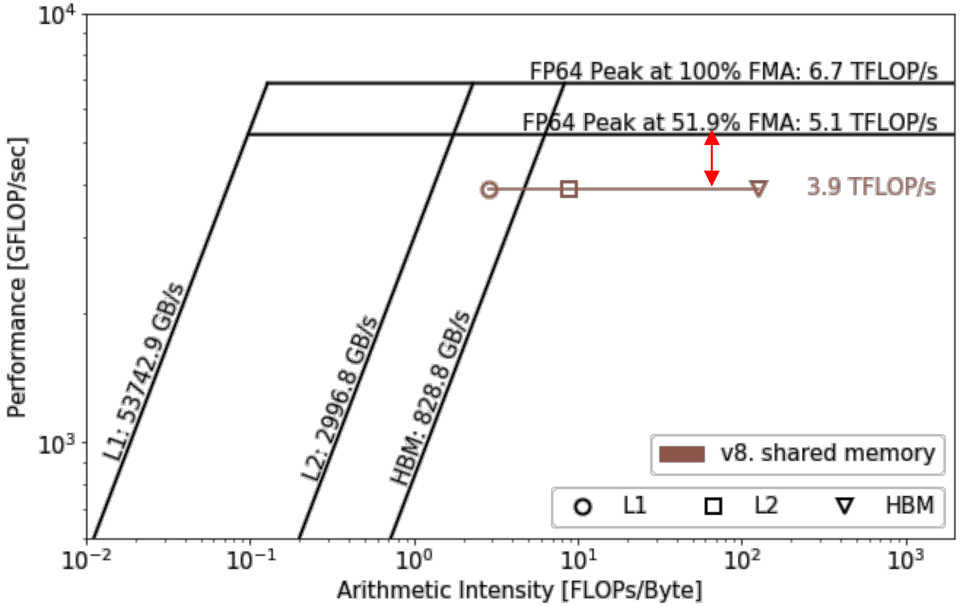
Reduce Memory Latency (v7 - v8)

- HBM data movement has dramatically reduced !



- Overall, we have achieved a 3.9 TFLOP/s performance in double precision
- Compared to the theoretical peak 6.7 TFLOP/s, we are at **58.4%** !

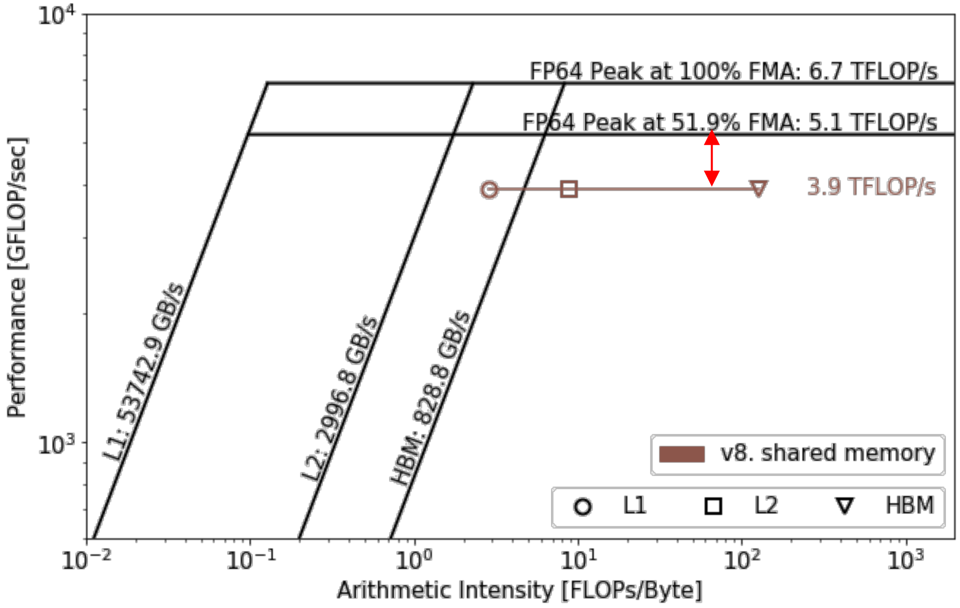
Final Results for GPP



- Measured with Nsight Compute, our FMA ratio is

$$\alpha = \frac{\text{FP64 FMA instructions}}{\text{FP64 instructions}} = 51.9\%$$

Final Results for GPP



- Given our FMA ratio, the more customized attainable peak is 5.1 TFLOP/s [1]

$$\frac{2\alpha + 1 - \alpha}{2} = 76\%$$

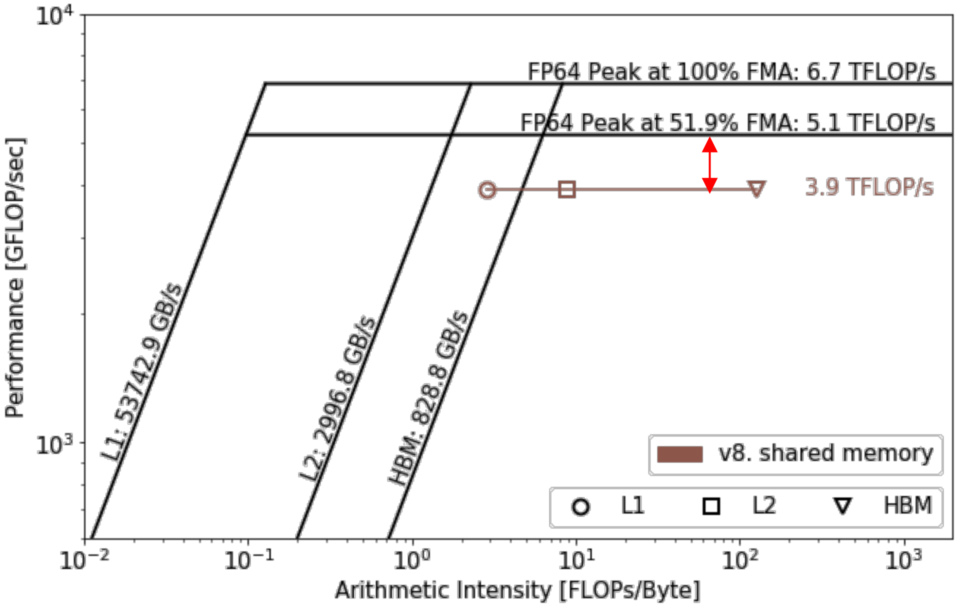
$$76\% \times 6.7\ \mathrm{TFLOP/s} = 5.1\ \mathrm{TFLOP/s}$$

- We are at **76.9%** of that peak!

[1] C. Yang, T. Kurth, and S. Williams, "Hierarchical Roofline Analysis for GPUs: Accelerating Performance Optimization for the NERSC-9 Perlmutter System", *Concurrency and Computation: Practice and Experience, DOI: 10.1002/cpe.5547*

Summary

For this complex scientific kernel: **3.9 TFLOP/s !**



	Time (s)	Speedup
v1	1557	1
v2	1389	1.12x
v3	1061	1.47x
v4	943	1.65x
v5	671	2.32x
v6	600	2.60x
v7	571	2.73x
v8	549	2.84x

Summary

For this complex scientific application, **105.9 PFLOP/s !!**

Application	BerkeleyGW
Benchmark	Si-2742
# of GPUs	27,648
Compute Time	592 s
I/O Time	39 s
Throughput	105.9 PFLOP/s (double precision)
% of R_{max}	71.3% of 148.60 PFLOP/s
% of R_{peak}	52.7% of 200.79 PFLOP/s

Roofline Analysis and other tricks!

Acknowledgement

- This research used resources at the National Energy Research Scientific Computing Center (NERSC), which is supported by the U.S. Department of Energy Office of Science under contract DE-AC02-05CH11231.

- This research used resources at the Oak Ridge Leadership Computing Facility (OLCF) through the Innovative and Novel Computational Impact on Theory and Experiment (INCITE) program, which is supported by the U.S. Department of Energy Office of Science under Contract No. DE-AC05-00OR22725.

- This work was supported by the Center for Computational Study of Excited-State Phenomena in Energy Materials (C2SEPEM), funded by the U.S. Department of Energy Office of Science under Contract No. DEAC02-05CH11231.

References

- S. Williams, A. Waterman, and D. Patterson, “Roofline: An Insightful Visual Performance Model for Multicore Architectures,” *Commun. ACM*, vol. 52, no. 4, 2009.

- C. Yang, T. Kurth, and S. Williams, "Hierarchical Roofline Analysis for GPUs: Accelerating Performance Optimization for the NERSC-9 Perlmutter System", *Concurrency and Computation: Practice and Experience, DOI: 10.1002/cpe.5547*

- https://gitlab.com/NERSC/roofline-on-nvidia-gpus
- https://docs.nvidia.com/nsight-compute/2020.1/ProfilingGuide/index.html#roofline

Thank You!