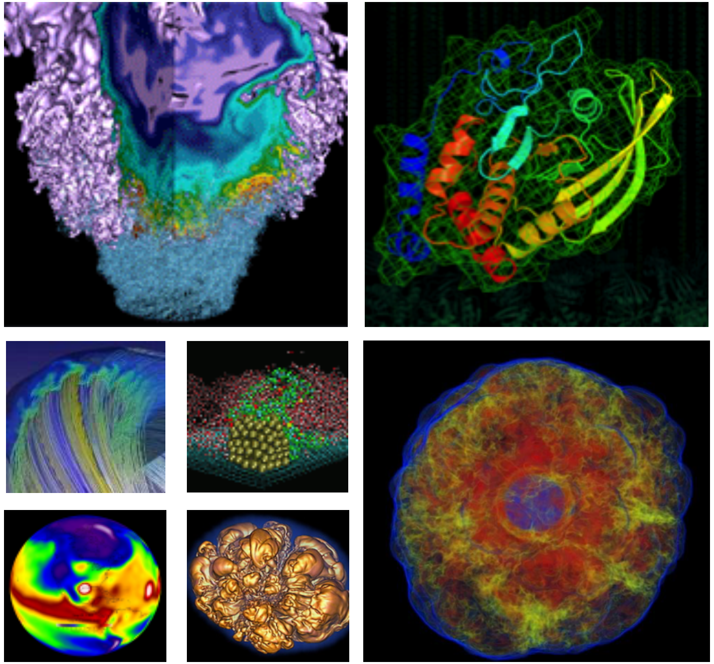
Case Studies: GPP, CoMD and XGC1



NERSC

Charlene Yang

Lawrence Berkeley National Laboratory
cjyang@lbl.gov

U.S. DEPARTMENT OF ENERGY | Office of Science

BERKELEY LAB

Outline

- **Roofline helps identify the performance bottleneck of a code**
  - GPP from BerkeleyGW -- Bandwidth-bound
  - CoMD from ddcMD and SPaSM -- Compute-bound

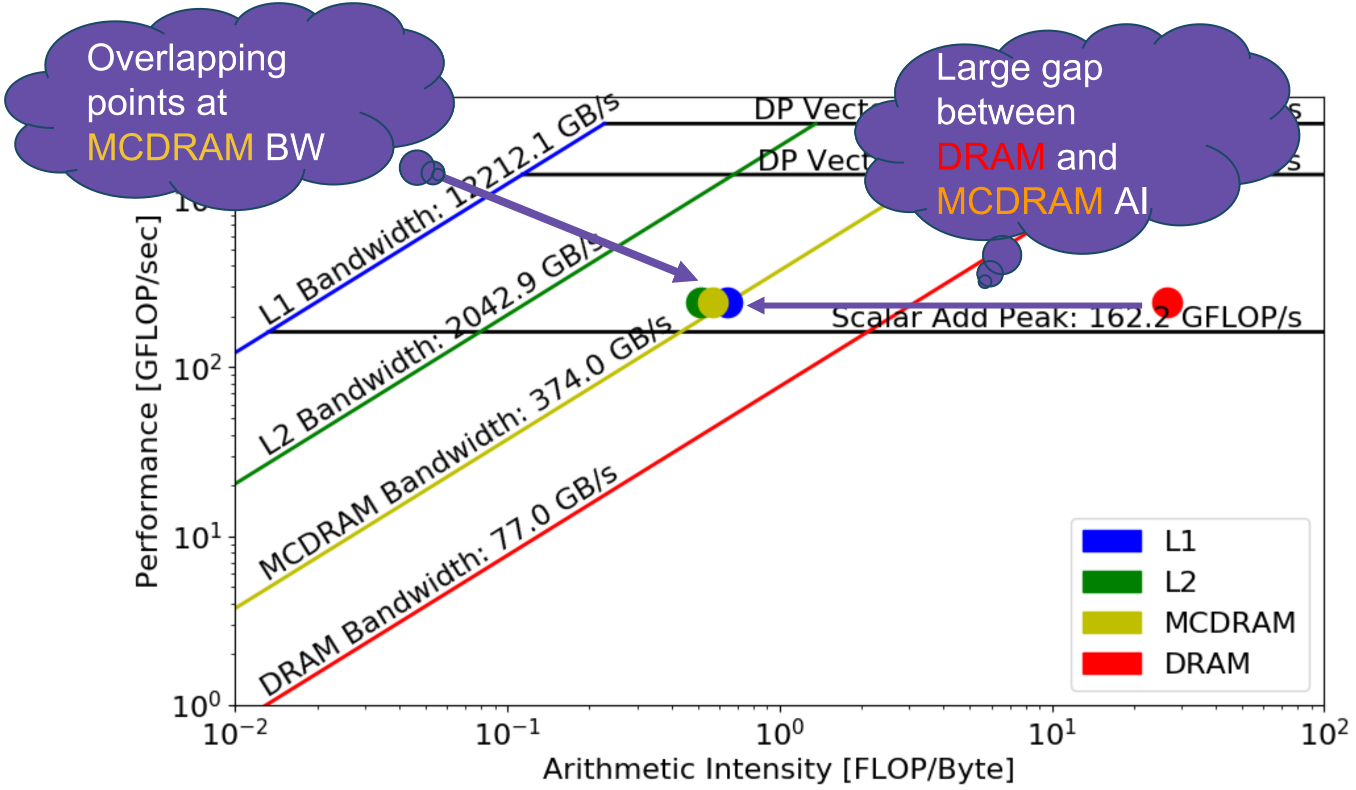
- **Roofline helps guide optimization efforts**
  - XGC1 from high energy physics -- Electron push module, ToyPush -- XGC1
  - A complete optimization process together with its results
  - How Roofline is used in the optimization process, together with other ad-hoc analysis, such as general code analysis, hotspot analysis, compiler report analysis on vectorization, dependency and memory access pattern, and instruction set analysis.

GPP: Material Science

- BerkeleyGW [1-3] is a material science application that predicts the excited-state properties of materials
  - https://berkeleygw.org
- GPP is a proxy code for BerkeleyGW, used for optimization efforts
  - Represents the work on a single MPI rank in a large BGW computation
  - https://github.com/rahulgayatri23/BGW-Kernels
- Performs a tensor-contraction like operation, where pre-computed complex double-precision arrays are multiplied/summed and collapsed into a 3x3 matrix
- Fortran/C++, OpenMP, ~500 LOC

[1] M. S. Hybertsen, S. G. Louie, Phys. Rev. B 34, 5390 (1986)
[2] M. Rohlfing, S. G. Louie, Phys. Rev. B 62, 4927 (2000)
[3] J. Deslippe, et. al., Comput. Phys. Commun. 183, 1269 (2012)

GPP - Original



KNL

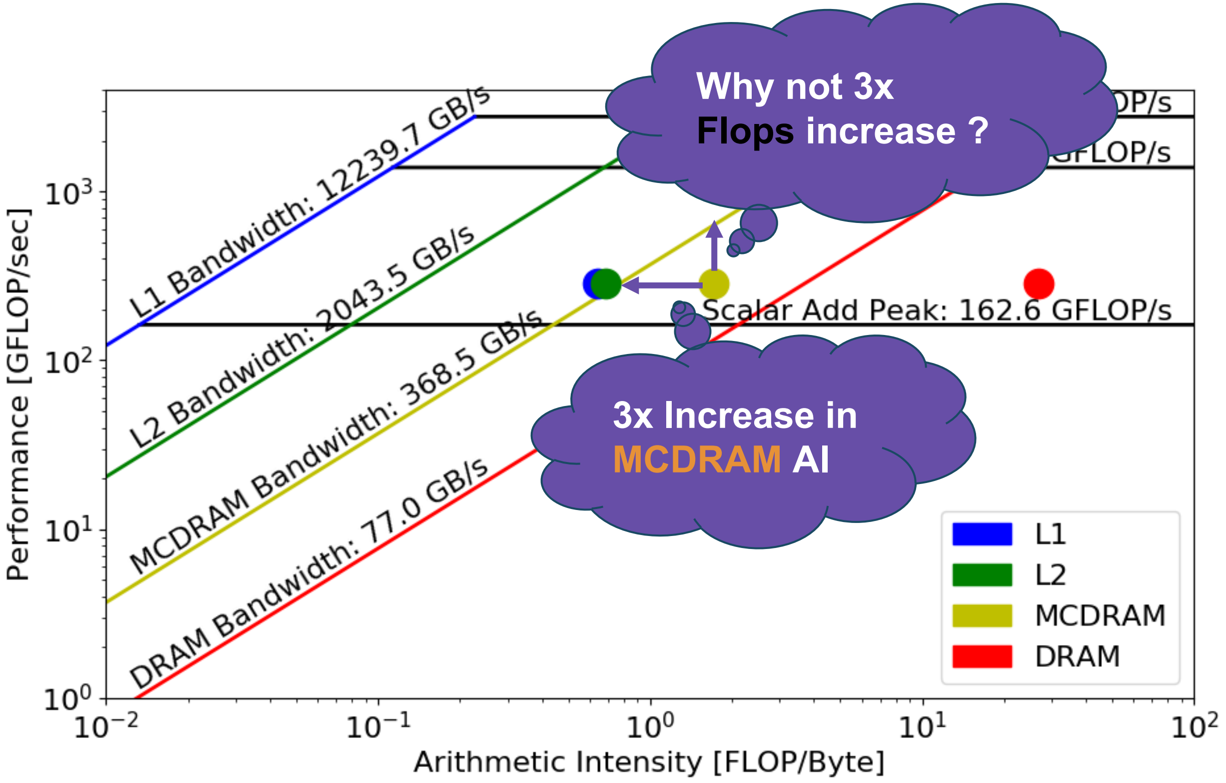
242 GFLOP/s, **Bound by MCDRAM Bandwidth**

Read/Write 2MB of data per inner loop iteration

➤ No reuse of data in L1/L2, shown by overlapping points at MCDRAM bandwidth

➤ Bandwidth bound. Increase MCDRAM AI by improving cache locality

GPP - With L2 Cache Blocking



KNL

Cache blocking implemented to achieve L2 data reuse. Notice the gap between L2 and MCDRAM dots.

Performance increased from 242 to 287 GFLOP/s (+18%)

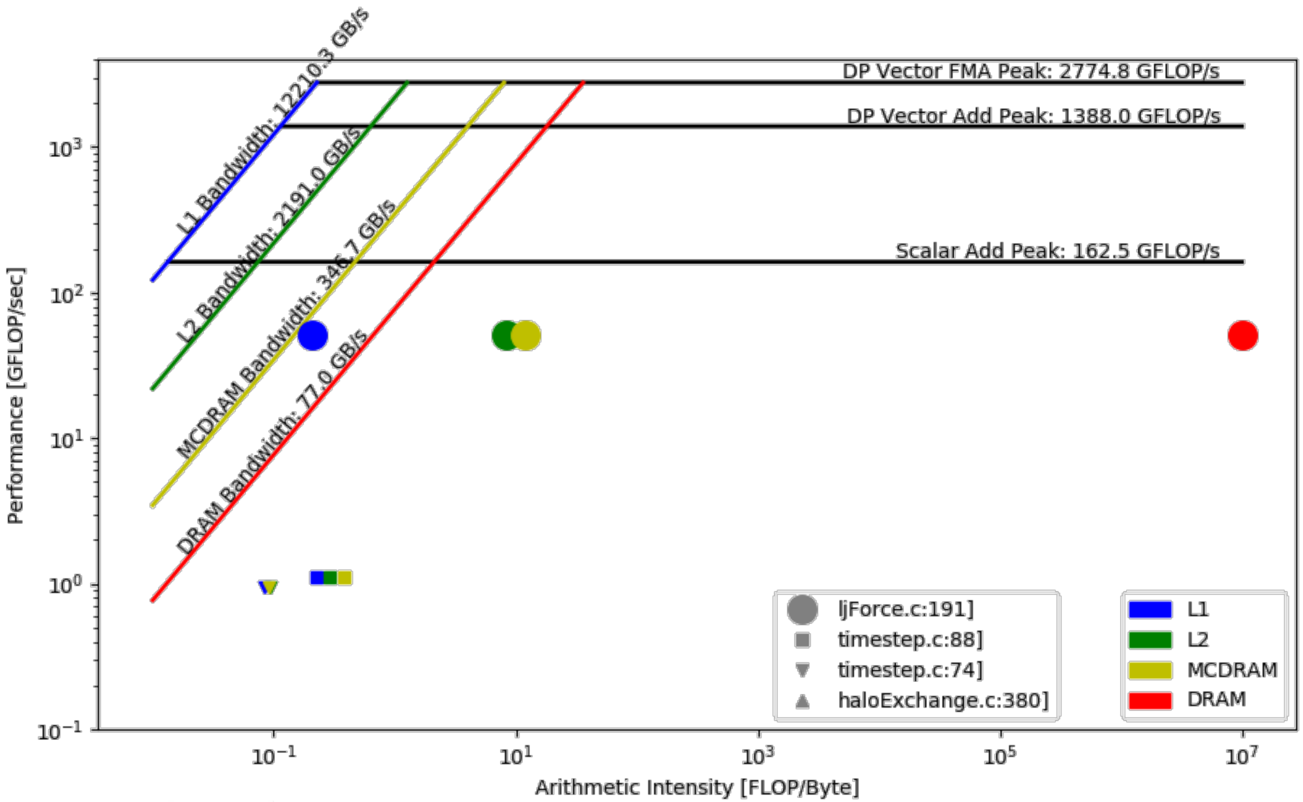
But why not 3x increase in FLOP/s?

CoMD: Molecular Dynamics

- CoMD [4-6] is a proxy code for ddcMD and SPaSM in molecular dynamics
- Molecular dynamics are usually N-body problems with $O(n^2)$ complexity
- Two types of force calculation, Leonard Jones (LJ) and Embedded Atom Model (EAM), and we focus on the LJ kernels
- CoMD is implemented in C with MPI and OpenMP, ~4k LOC

[4] J. Mohd-Yusof, CoDesign Molecular Dynamics (CoMD) Proxy App, LA-UR-12-21782, Los Alamos National Lab, 2012.
[5] P. Cicotti, et. al., An Evaluation of Threaded Models for a Classical MD Proxy Application, Hardware-Software Co-Design for High Performance Computing, New Orleans, LA, 2014, pp. 41-48.
[6] A. Adedoyin, A Case Study on Software Modernization using CoMD, LA-UR-17-22676, Los Alamos National Lab, 2017.

CoMD - Original



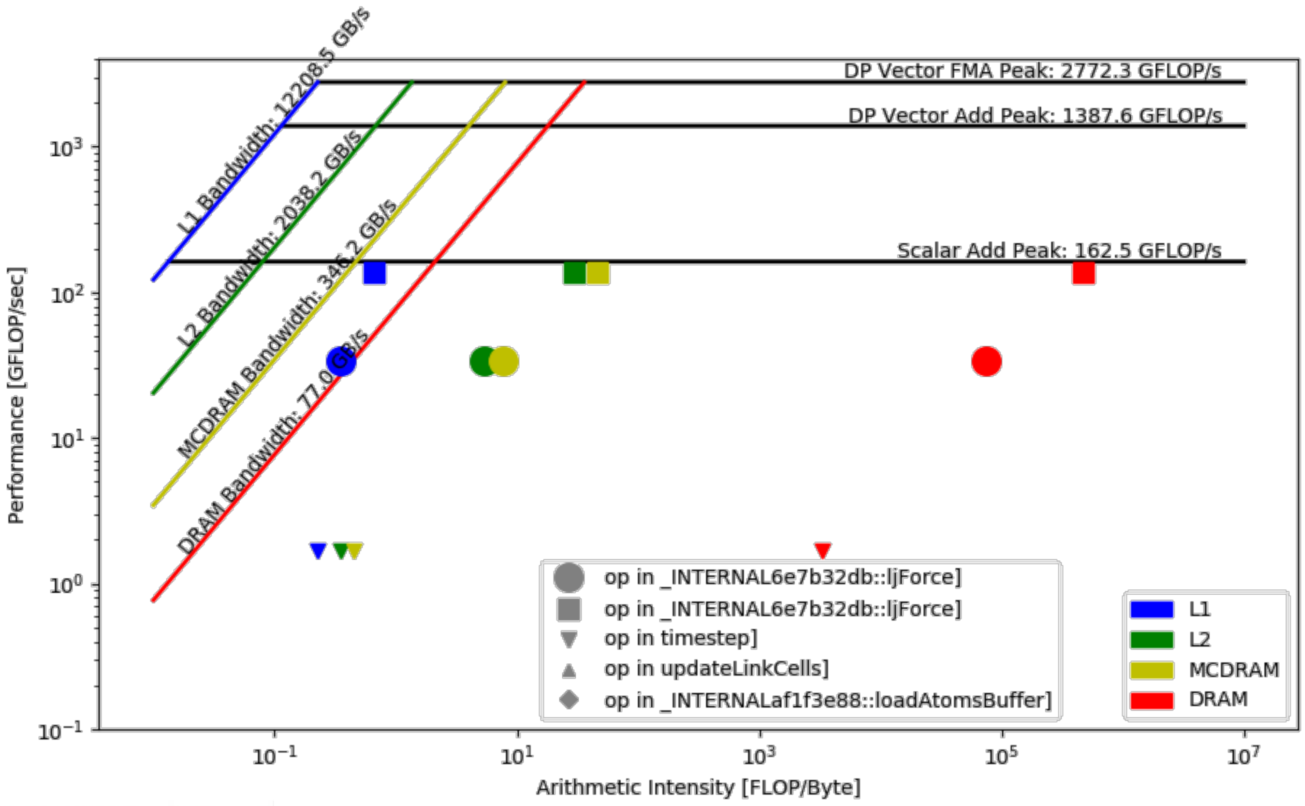
KNL

Good L1 and MCDRAM locality, **not really bandwidth bound**.

Look at other aspects (**compute**)

- Data level parallelism
- Thread level parallelism

CoMD - Better Vectorized



KNL

30% improvement

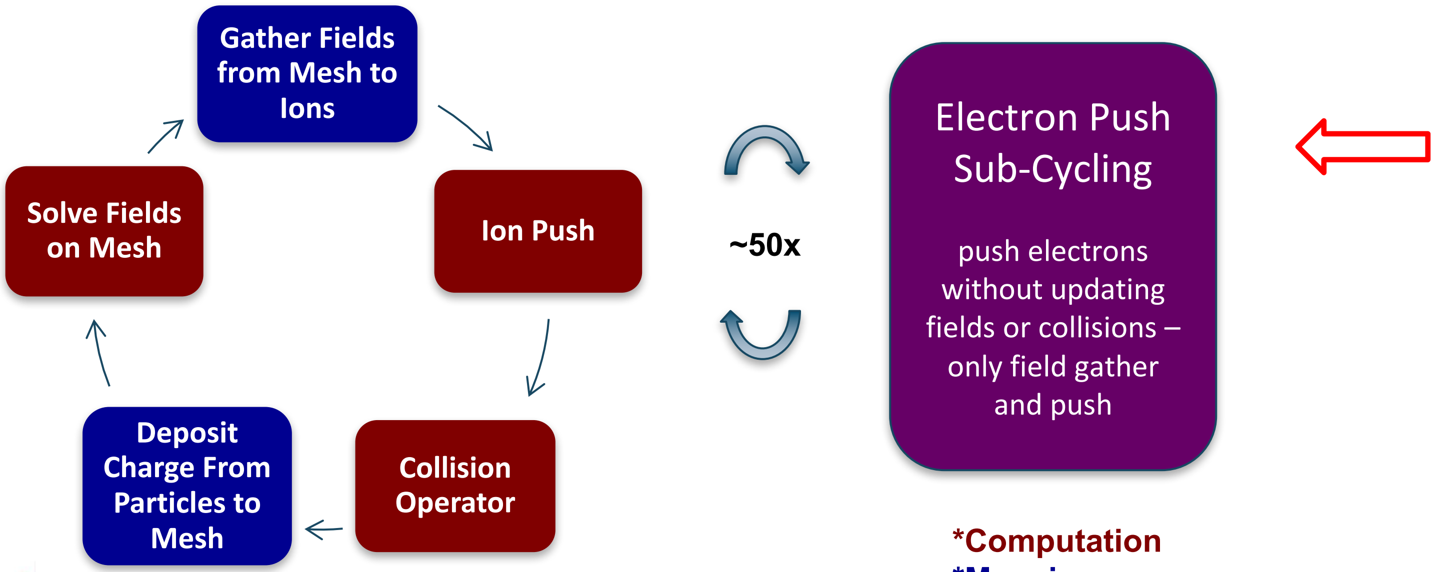
➤ Vectorization. Data alignment, compiler hints, data structure transformations.

➤ Work remains to be done, given the gap between LJ and the Vector Peak.

Roofline needs to be used together with other analysis/tools, e.g. compiler report.

XGC1: Particle-In-Cell (PIC)

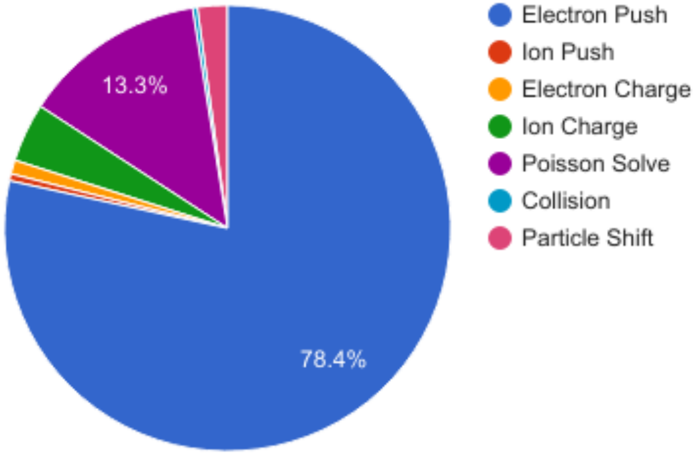
- PIC code to simulate edge plasmas for Tokamak fusion reactor
- Written in Fortran 90, parallelized with MPI and OpenMP, ~100k LOC
- **Code analysis:**

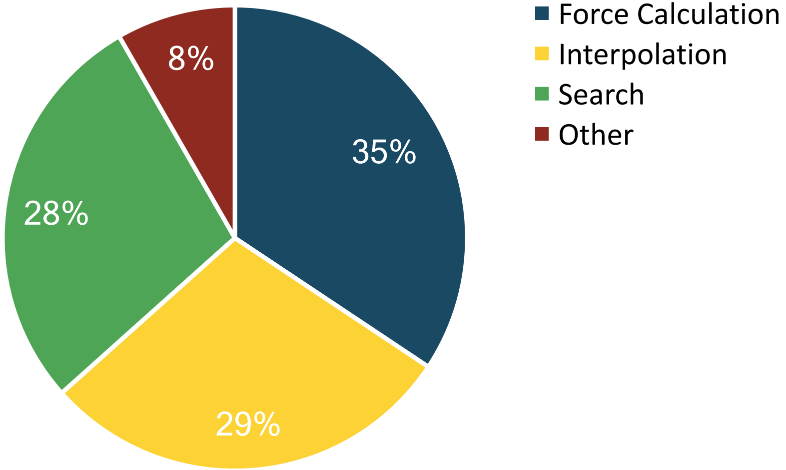


***Computation**
***Mapping**

XGC1 - ToyPush

- **Hotspot analysis:**

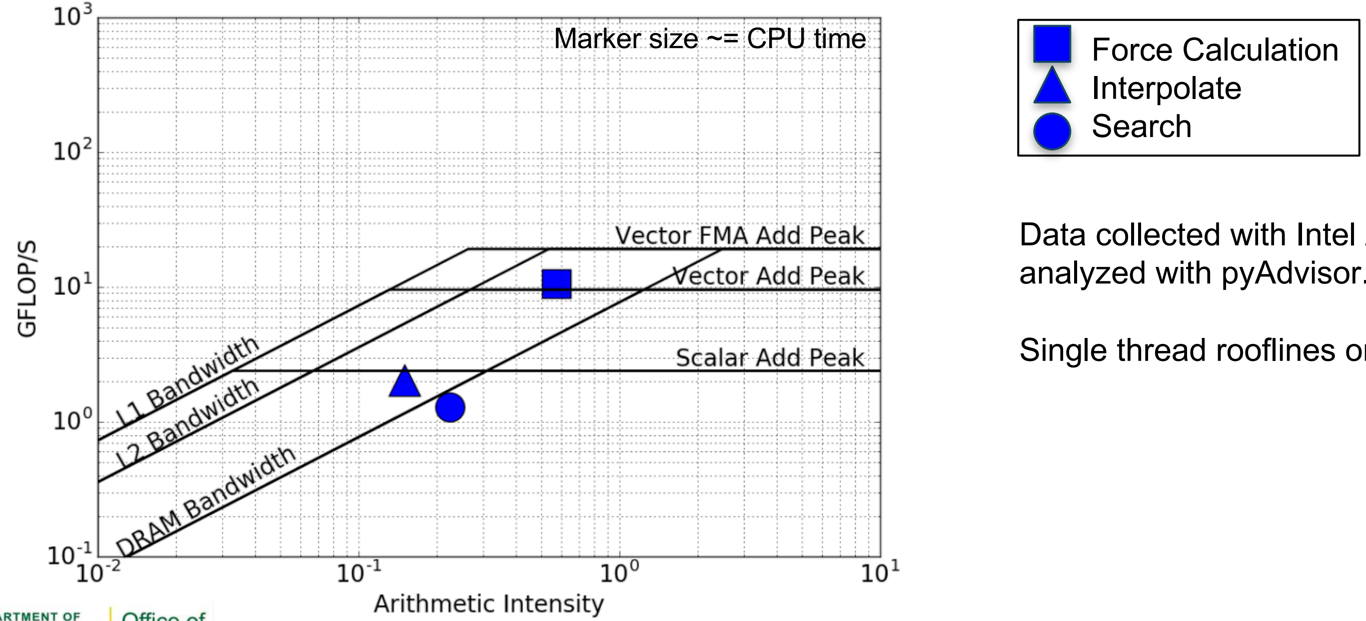




Left: Unoptimized XGC1 timings on 1024 Cori KNL nodes in Quad-Flat mode
Right: Unoptimized ToyPush timings on Cori KNL in Quad-Cache mode
***ToyPush is the proxy app for electron push part of XGC1.**

ToyPush: Baseline Profile

- Force Calculation: close to vector peak
- **Interpolate** and **Search**: less than scalar peak



Data collected with Intel Advisor and analyzed with pyAdvisor.

Single thread rooflines on Cori KNL.

ToyPush - Interpolation

- **Compiler vectorization report**
- Indirect access/gathers -> group particles together that access the same triangle

```
efield(j,tri(i,itri(iv)))
```

- Unaligned access -> align at compile time
- Improved vectorization efficiency

```
LOOP BEGIN at interpolate_aos.F90(67,48)
reference itri(iv) has unaligned access
reference y(iv,1) has unaligned access
reference y(iv,3) has unaligned access
reference evec(iv,icomp) has unaligned access
reference evec(iv,icomp) has unaligned access
.....
irregularly indexed load was generated for the
variable <grid_mapping_(1,3,itri(iv))>, 64-bit
indexed, part of index is read from memory
.....

LOOP WAS VECTORIZED
unmasked unaligned unit stride loads: 6
unmasked unaligned unit stride stores: 3
unmasked indexed (or gather) loads: 18
.....
```

ToyPush - Interpolation

- **Use Advisor to examine cache behavior**
- L1 hit rate low -> shorten veclength from 2^9 to 2^6 to achieve L1 blocking

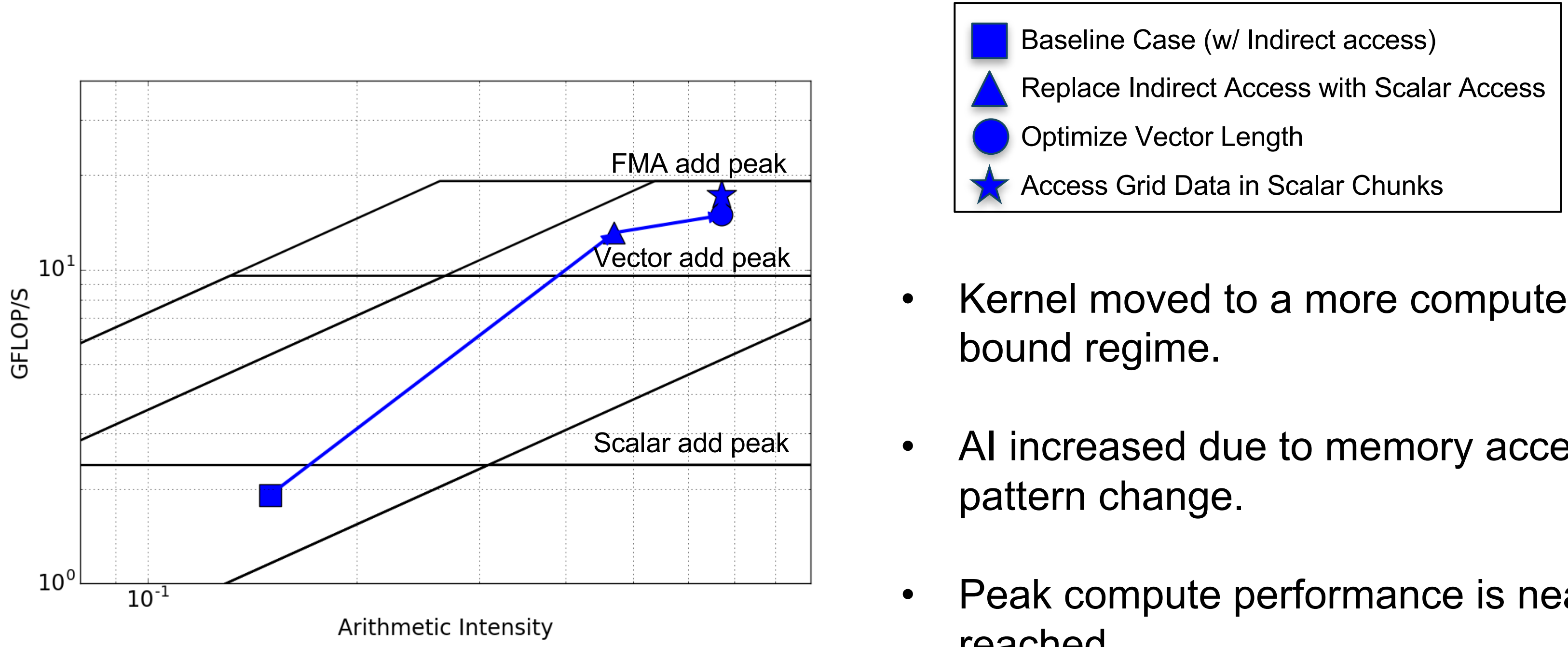
Grouping: Function / Call Stack

Function / Call Stack	Clockticks ▼	Instructions Retired	L1 Hit Rate	L2 Hit Rate	L2 Hit Bound	L2 Miss Bound
e_interpol_tri	105,271,600,000	64,954,400,000	80.8%	94.4%	36.7%	29.5%
eom_eval	73,858,400,000	65,283,400,000	67.3%	99.9%	100.0%	0.8%
b_interpol_analytic	60,141,200,000	23,109,800,000	90.3%	100.0%	4.2%	0.0%
__intel_mic_avx512f_memset	35,288,400,000	3,441,200,000	42.1%	100.0%	0.8%	0.0%
rk4_push	20,528,200,000	14,898,800,000	31.9%	100.0%	100.0%	0.0%

Grouping: Function / Call Stack

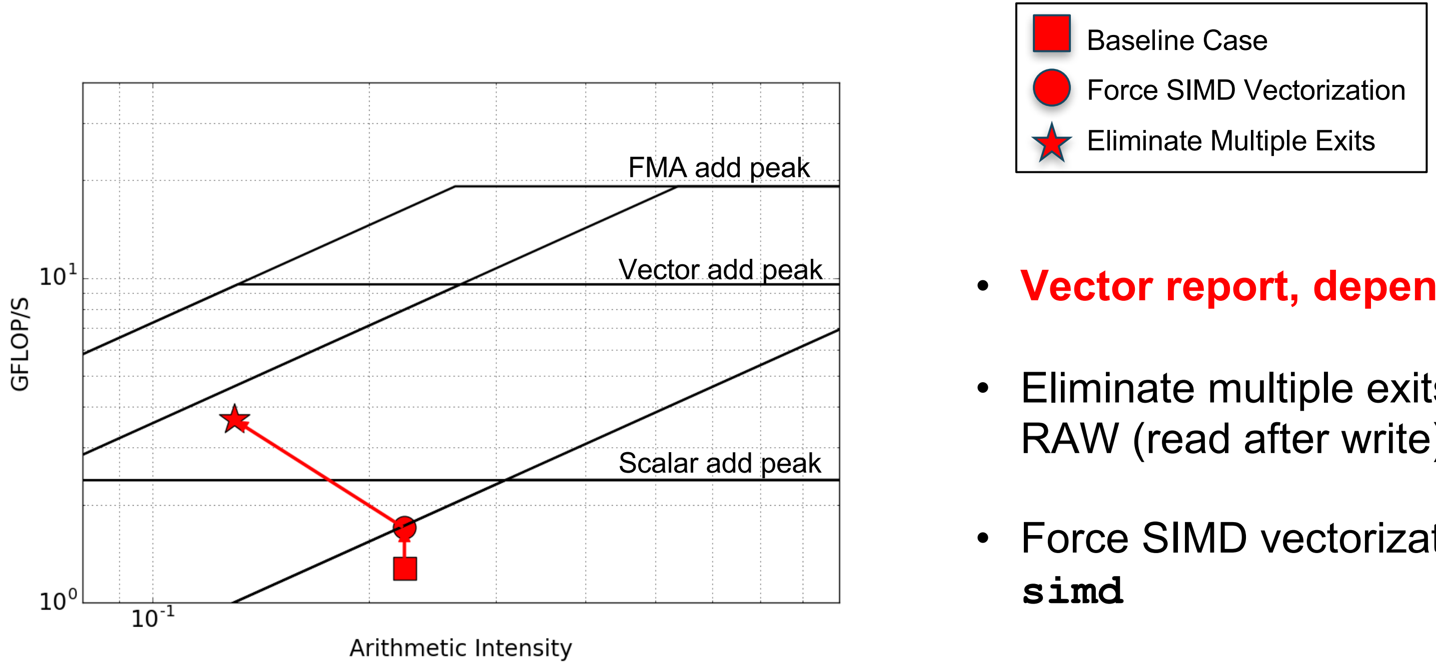
Function / Call Stack	Clockticks ▼	Instructions Retired	L1 Hit Rate	L2 Hit Rate	L2 Hit Bound	L2 Miss Bound
e_interpol_tri	97,042,400,000	76,687,800,000	99.4%	100.0%	0.9%	0.0%
eom_eval	66,556,000,000	67,110,400,000	99.0%	100.0%	3.3%	0.0%
b_interpol_analytic	16,360,400,000	23,641,800,000	99.3%	100.0%	0.3%	0.0%
proc_reg_read	14,984,200,000	75,600,000	100.0%	0.0%	0.0%	0.0%
rk4_push	14,954,800,000	19,702,200,000	98.5%	100.0%	24.8%	0.0%

ToyPush - Interpolation



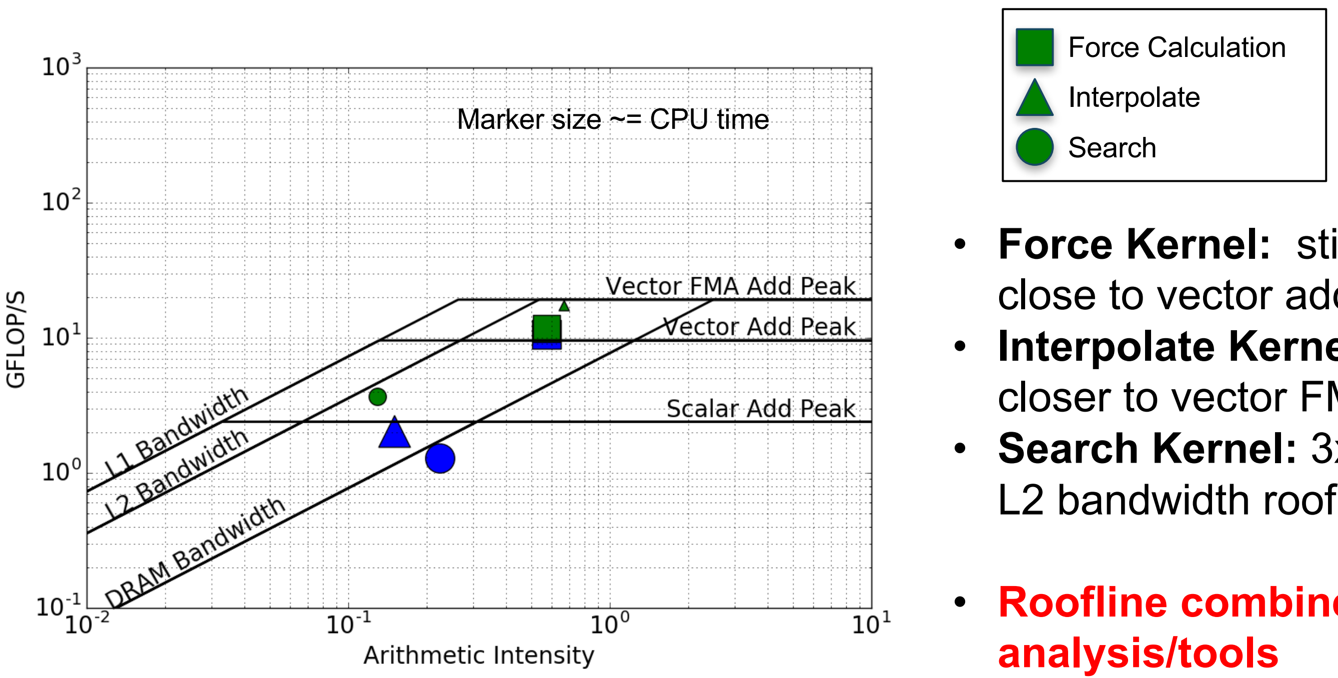
- Kernel moved to a more compute bound regime.
- AI increased due to memory access pattern change.
- Peak compute performance is nearly reached.

ToyPush - Search



- **Vector report, dependency report**
- Eliminate multiple exits, ‘cycle’, and RAW (read after write) dependency
- Force SIMD vectorization with **`omp simd`**

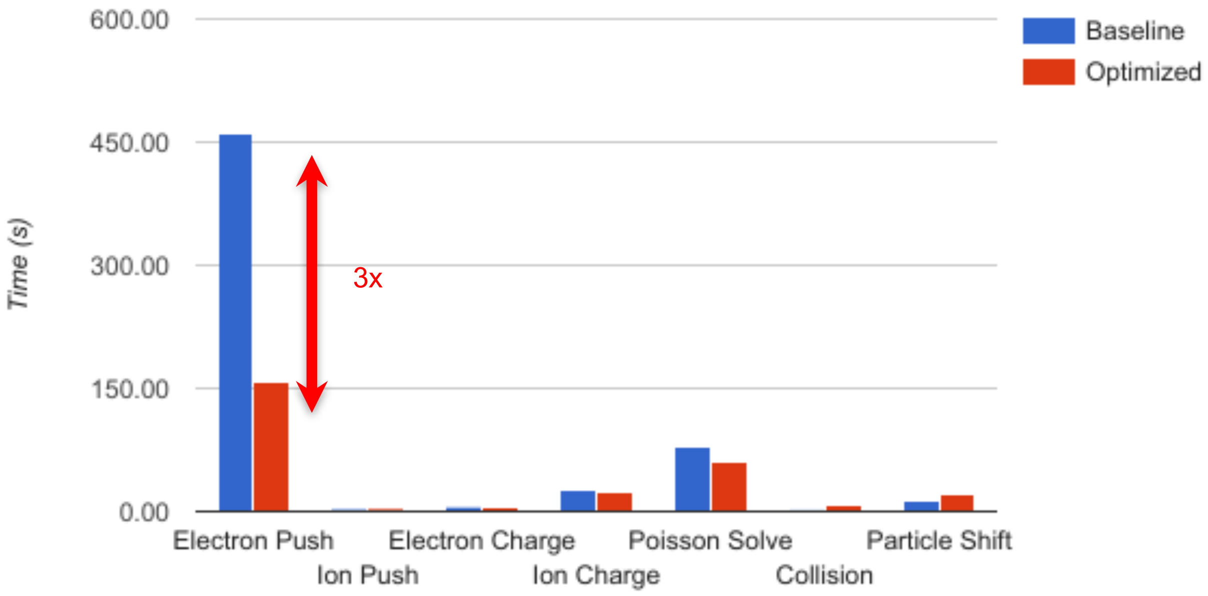
ToyPush: Optimized Profile



- **Force Kernel:** still good performance, close to vector add peak
- **Interpolate Kernel:** 10x speedup, closer to vector FMA peak
- **Search Kernel:** 3x speedup, closer to L2 bandwidth roof

- **Roofline combined with other analysis/tools**

XGC1: Merge ToyPush Changes (WIP)



XGC1 Timings on 1024 Cori KNL nodes in Quad-Flat mode

Summary

- Showcased three scientific applications, and their performance analysis and/or optimization process: GPP from BerkeleyGW, CoMD from ddcMD and SPaSM, and XGC1.

- Roofline model can help identify performance bottlenecks, prioritize optimization efforts (e.g. routines, vectorization, memory access), and tell when to stop (e.g. attainable performance, distance to roofs).

- Complement Roofline with generic code analysis, compiler reports, binary analysis to confirm details and ways to implement optimizations.
  - vectorization, dependency, memory access pattern, cache locality, cache hit rate, instruction mix

- Tools such as Intel Advisor, Intel VTune are very useful!

Thank You!

http://bit.ly/sc18-eval