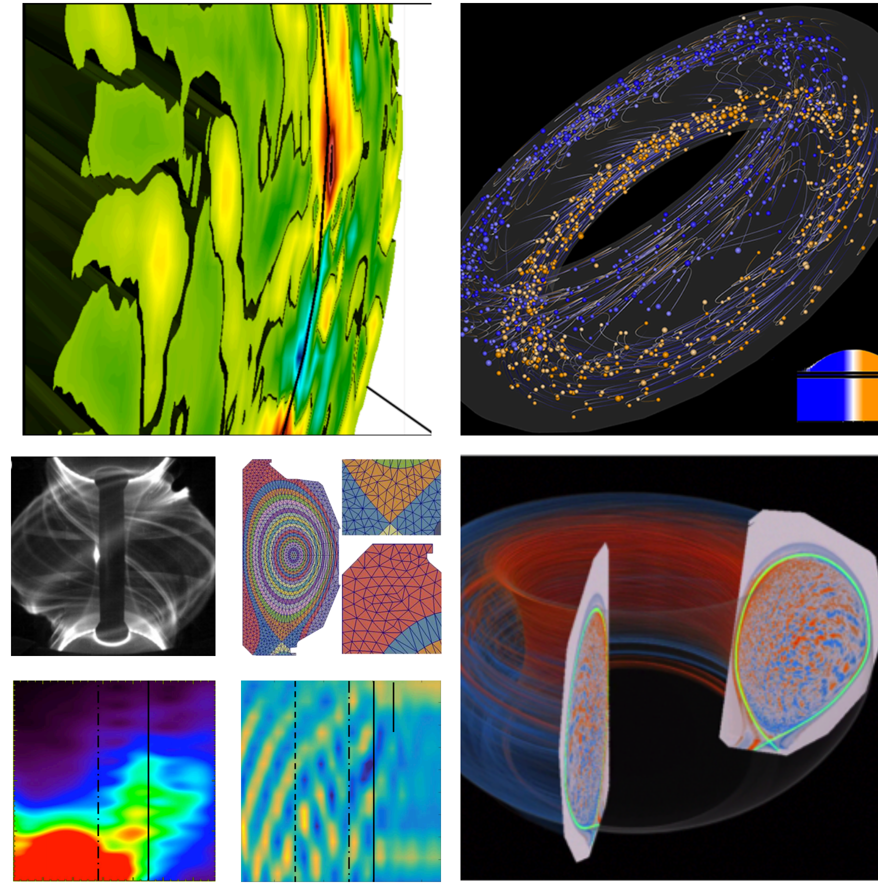
Case Study: Fusion PIC Code XGC1 on Cori KNL



Charlene Yang

Application Performance Specialist
NERSC, LBNL

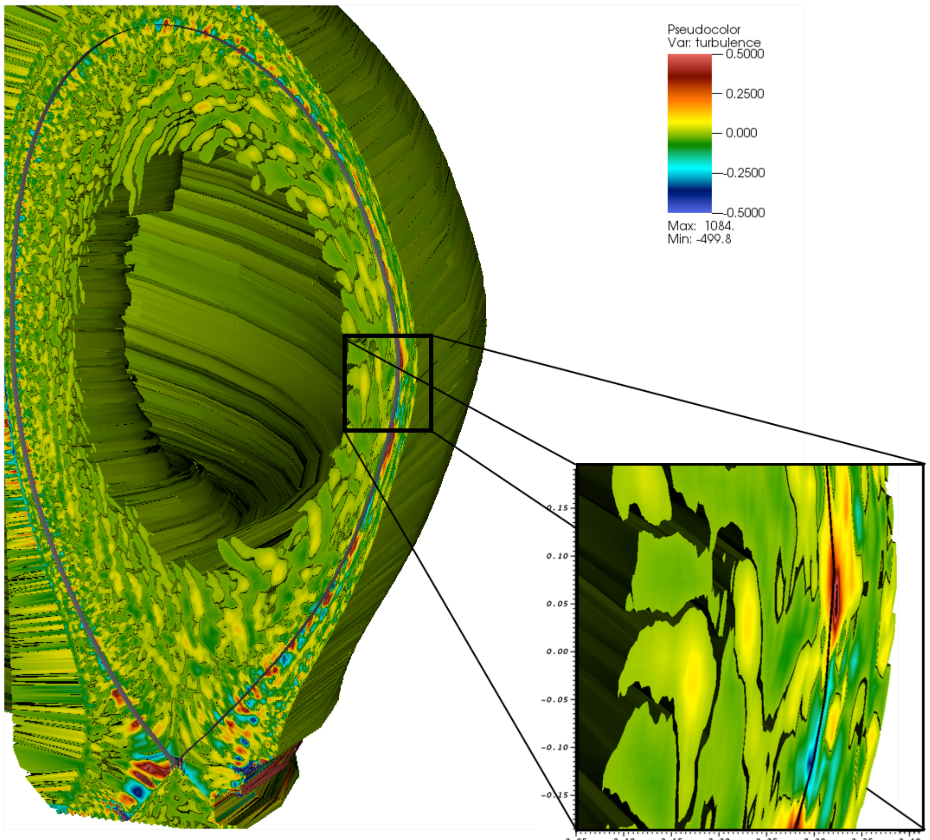
cjyang@lbl.gov

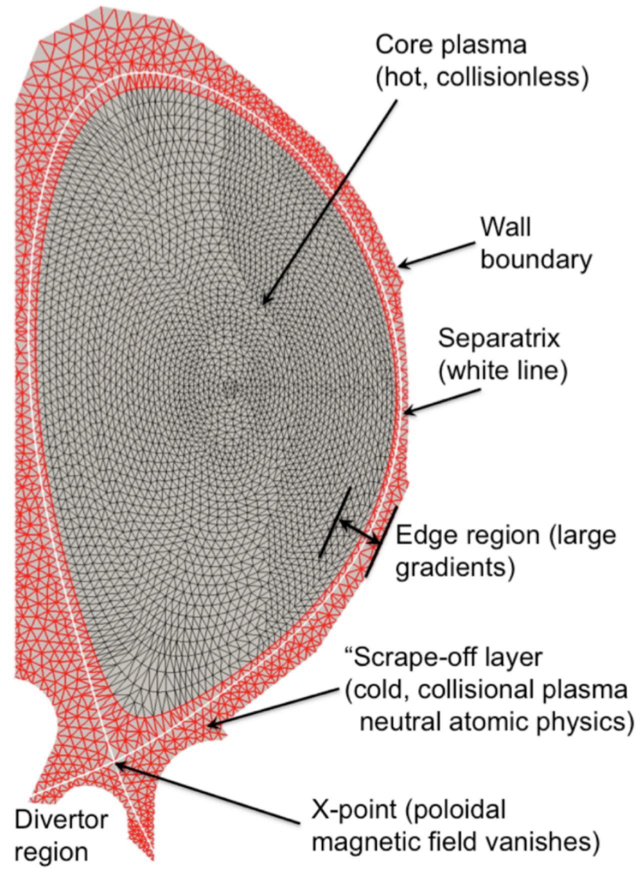
NERSC

U.S. DEPARTMENT OF ENERGY | Office of Science

BERKELEY LAB
Lawrence Berkeley National Laboratory

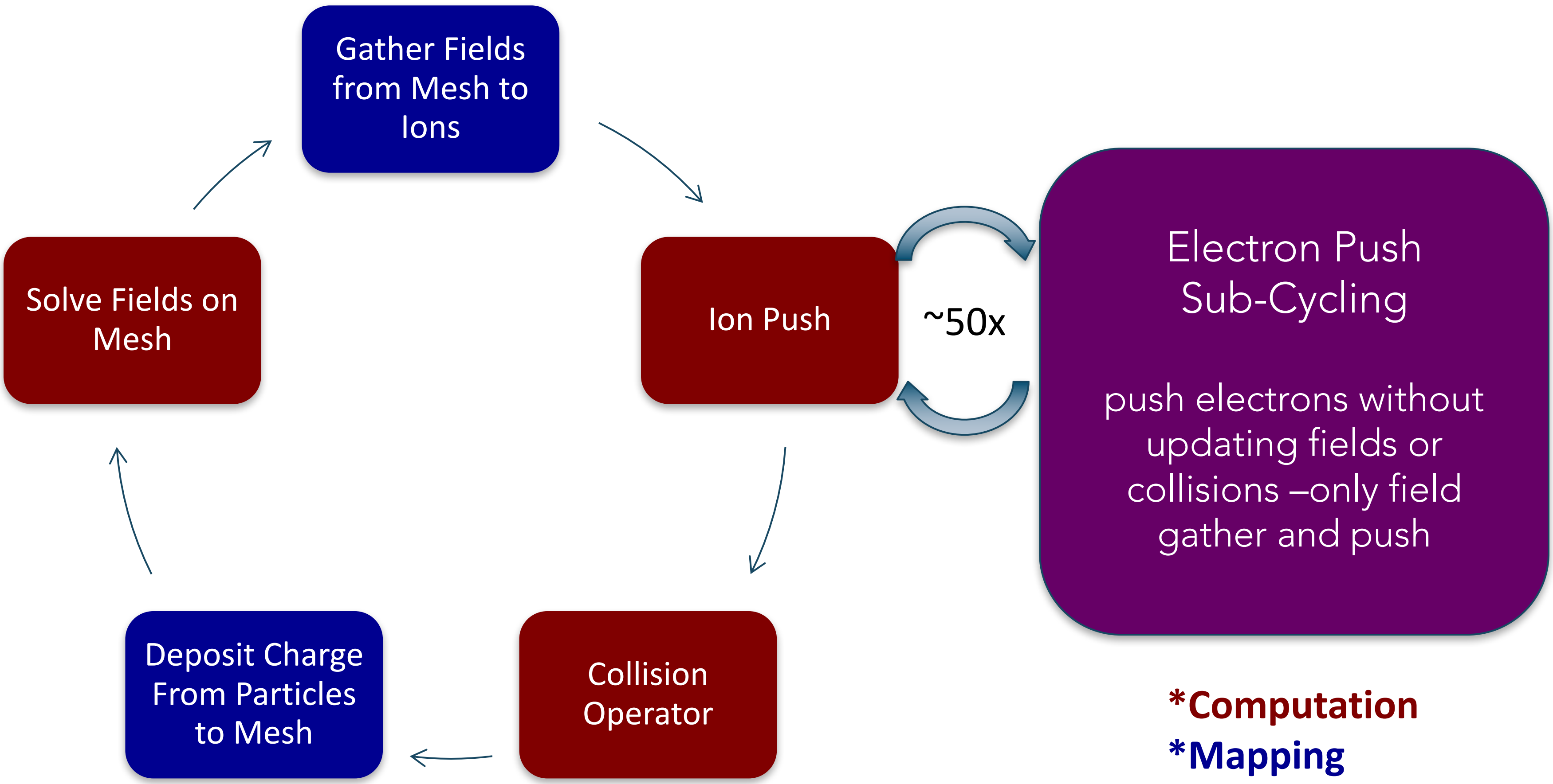
XGC1: Particle-In-Cell Simulation





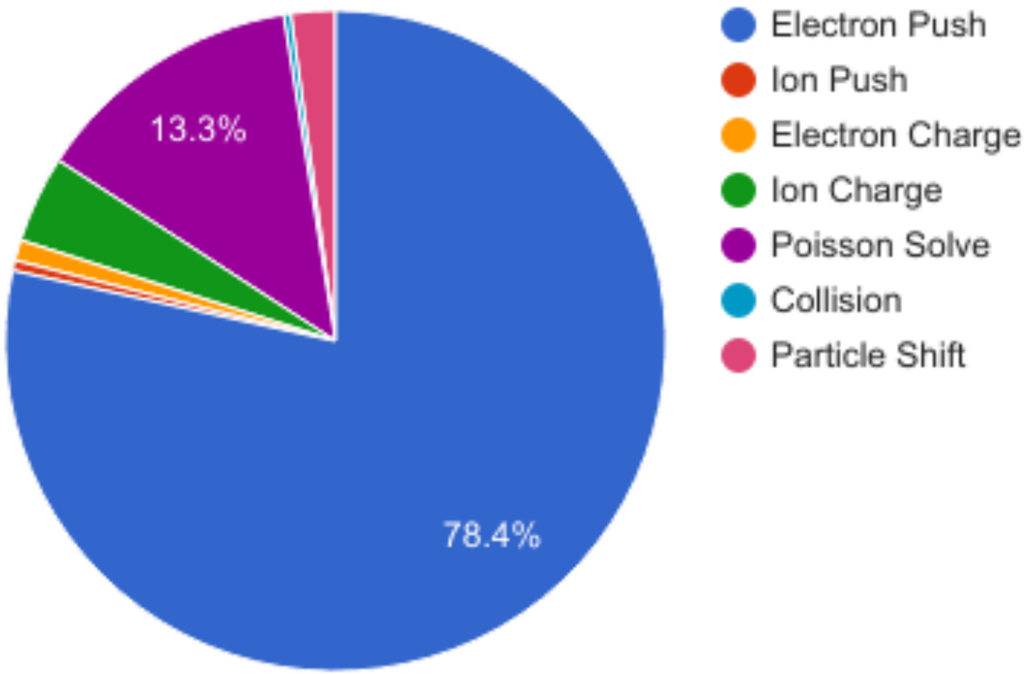
PI: CS Chang (PPPL) | ECP: High-Fidelity Whole Device Modeling of Magnetically Confined Fusion Plasma

XGC1: Code Flowchart



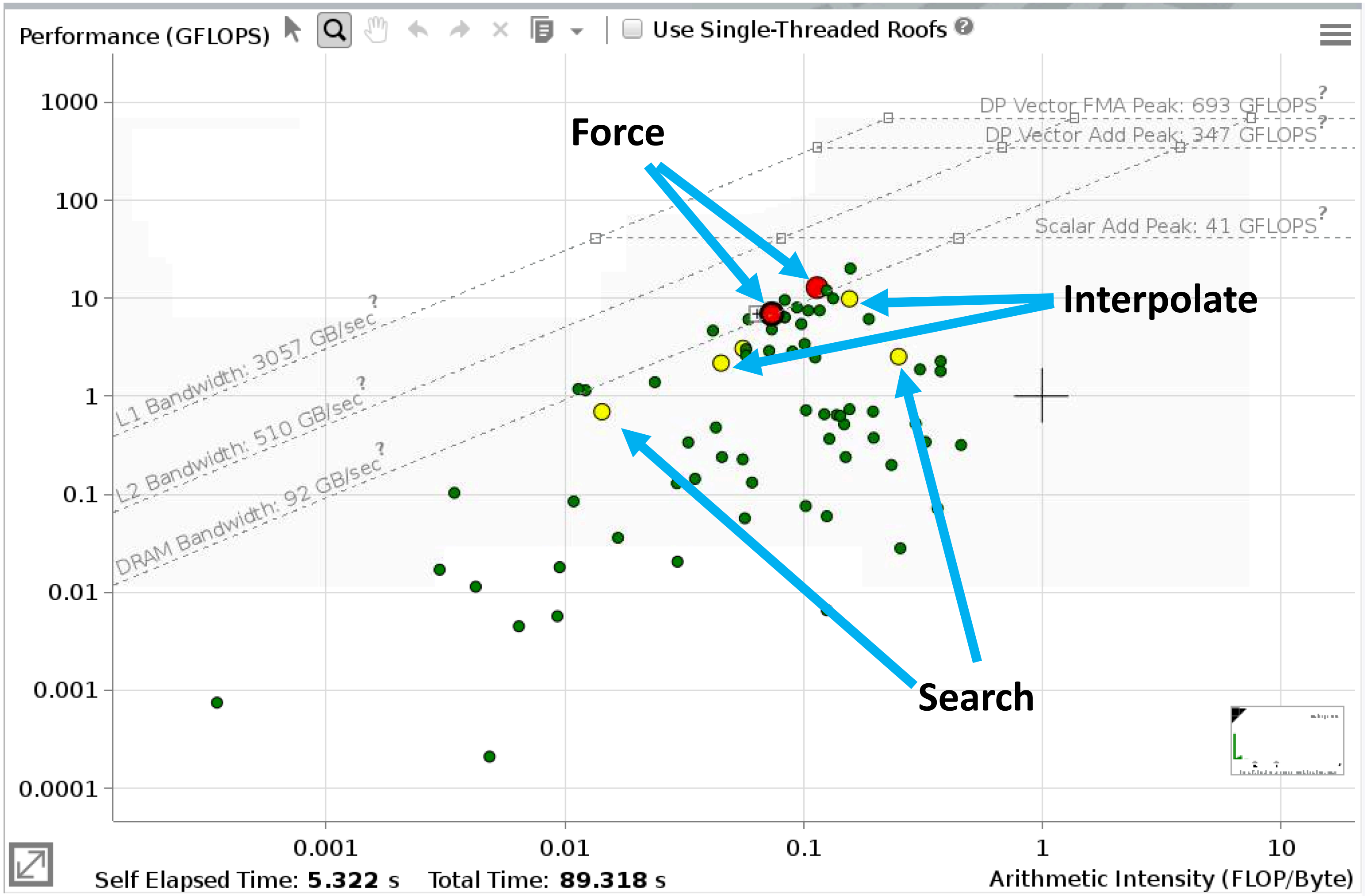
***Computation**
***Mapping**

XGC1: Code Timings

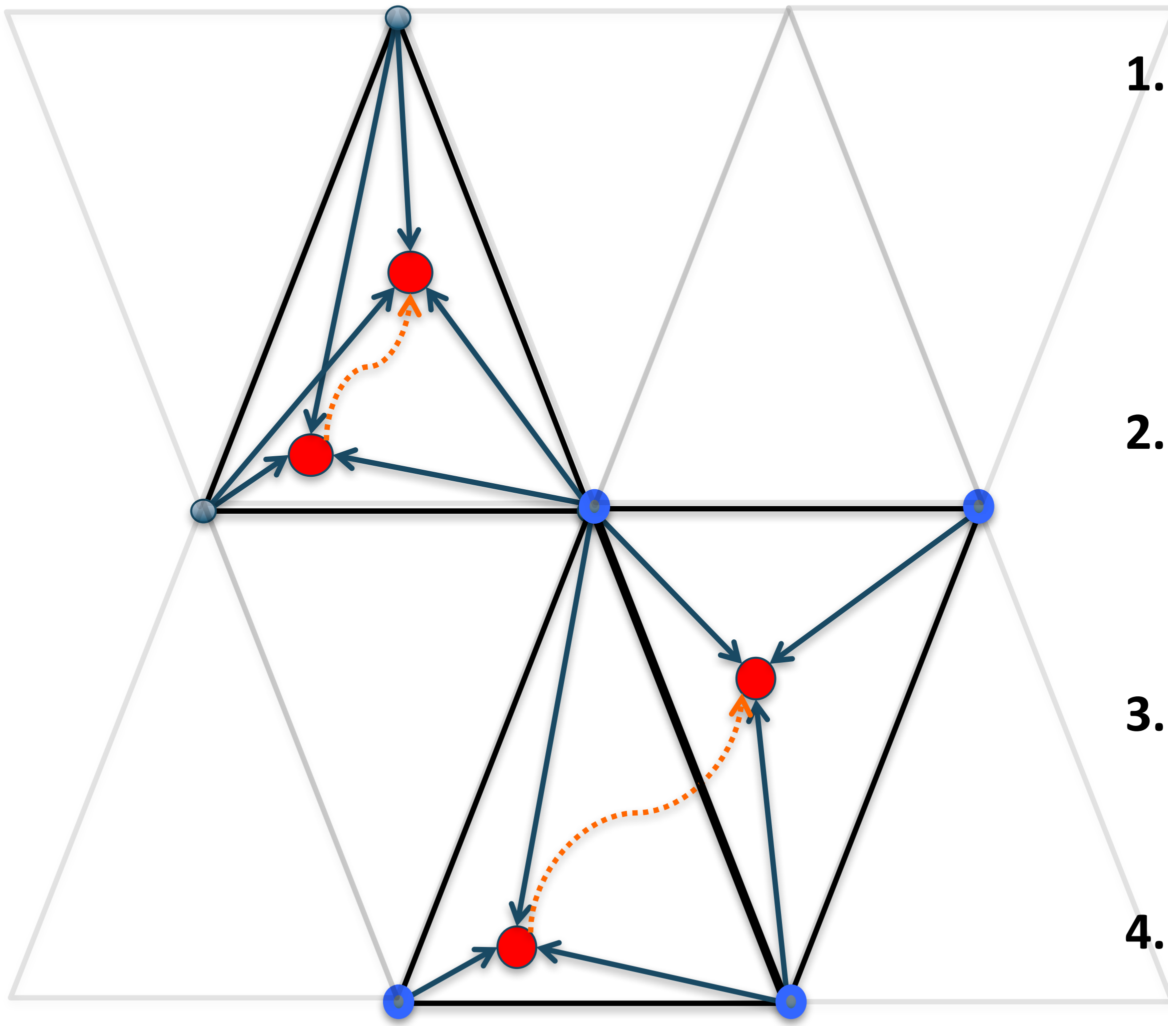


Unoptimized XGC1 Timings on 1024 Cori KNL nodes in Quad-Flat mode

XGC1: Code Profile on Roofline



ToyPush: Mini-App for Electron Sub-Cycling



1. Search for nearest 3 mesh nodes to the particle position
   (for multi-mesh refinement)

2. Interpolate fields from 3 mesh points to particle position

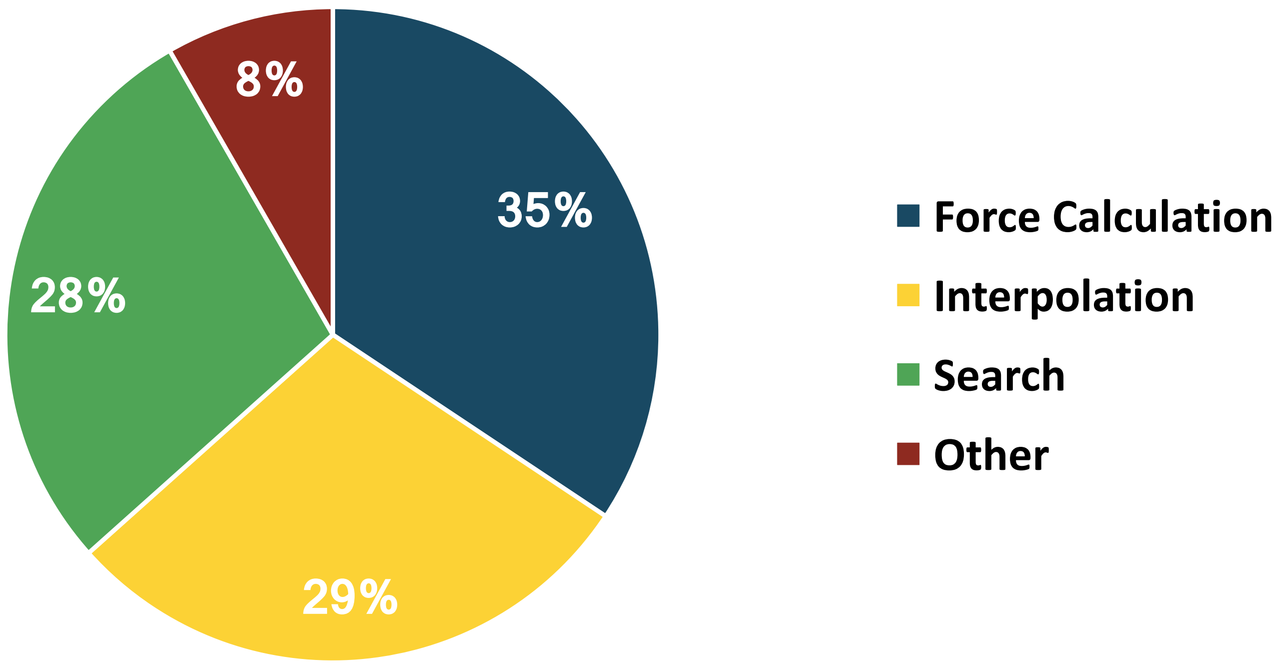
3. Calculate force on particle from fields

4. Push particle for time step dt

U.S. DEPARTMENT OF ENERGY | Office of Science

BERKELEY LAB

ToyPush: Baseline Profile - Timings

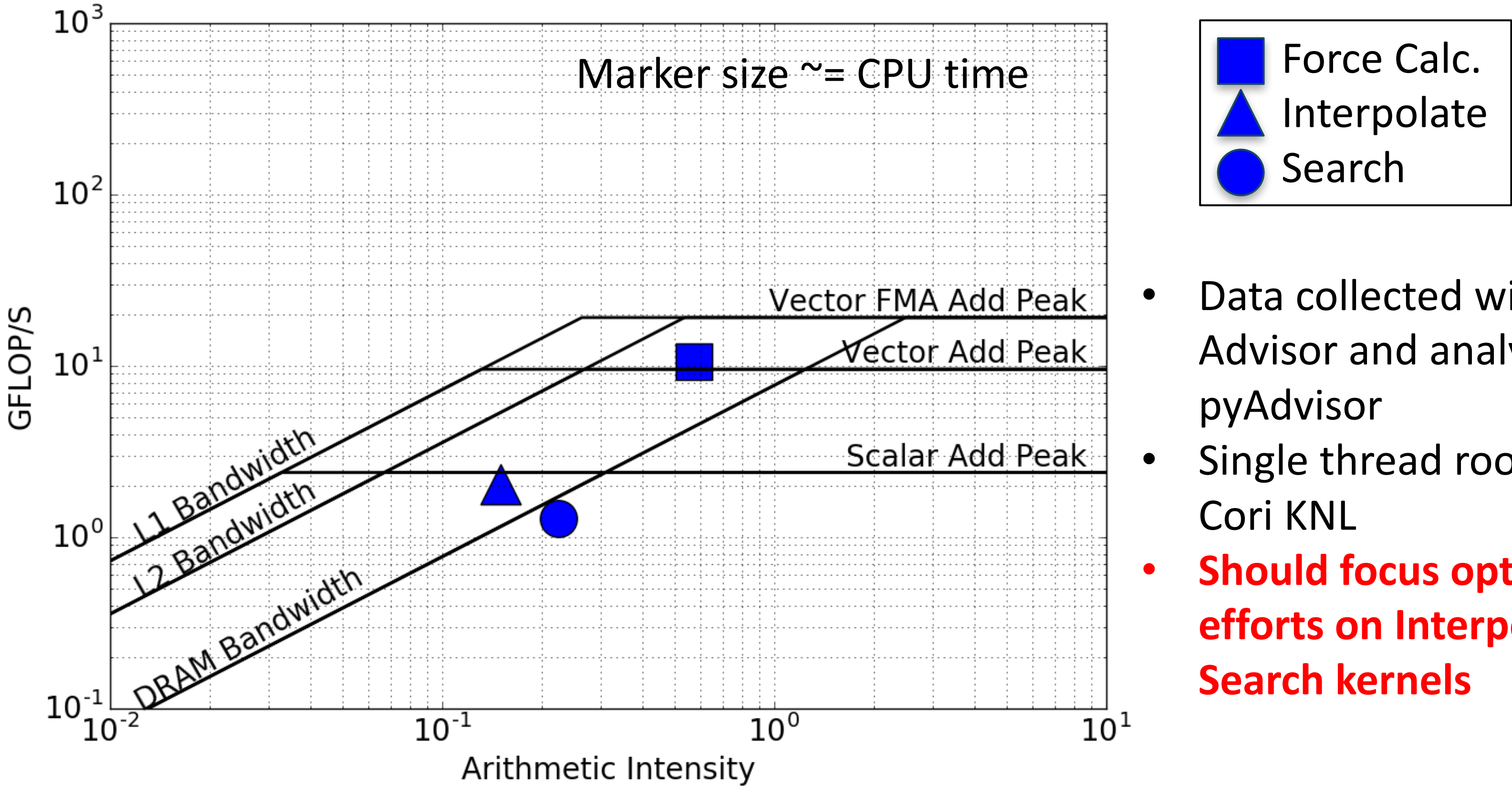


Unoptimized ToyPush Timings on Cori KNL in Quad-Cache mode

ToyPush: Baseline Profile - Roofline

Roofline Model shows information timings alone can not show

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



- Data collected with Intel Advisor and analyzed with pyAdvisor
- Single thread rooflines on Cori KNL
- **Should focus optimization efforts on Interpolate and Search kernels**

Interpolation: Vectorization – L1 Blocking

- **veclength optimizations**
  - Baseline: 2^(9)

Low L1 Hit Rate, L2 Hit Bound

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%

  - Optimized: 2^(6)

High L1 Hit Rate

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%

~1.5x improvement (MCDRAM Flat); ~2x improvement (DDR Flat)

Interpolation: Vectorization - Memory Access

Problems:

- **Field data is stored on grid nodes, particles access nearest 3 grid nodes indirectly via triangle index.**
  efield(j, tri(i, itri(iv)))

- **Interpolation loop is vectorized but inefficiently because of gather loads**

18 Gathers per loop iteration
(3 nodes x 3 components x 2)

Intel Compiler Vectorization Report

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
.....

Interpolation: Vectorization - Memory Access

Optimizations:

- Group particles that access the same triangle together, access grid nodes directly with a scalar index
- Single mesh: Trivial
- Multiple mesh: Feasible for number of particles >> number of grid nodes
- Align arrays during compile time.

Intel Compiler Vectorization Report

LOOP BEGIN at interpolate_aos.F90(72,51)
reference y(iv,1) has aligned access
reference y(iv,3) has aligned access
reference evec(iv, icomp) has aligned access
.....
SIMD LOOP WAS VECTORIZED
.....
unmasked aligned unit stride loads: 5
unmasked aligned unit stride stores: 3
....

~1.6x improvement

Interpolation: Vectorization – memset

Problem:

- **Initialization of large arrays with avx512_memset at every time step before entering vector loop becomes memory bandwidth bound.**

Intel Compiler Vectorization Report

LOOP BEGIN at interpolate_aos.F90(57,5)
memset generated
loop was not vectorized:
loop was transformed to memset or memcpy
LOOP END

Optimizations:

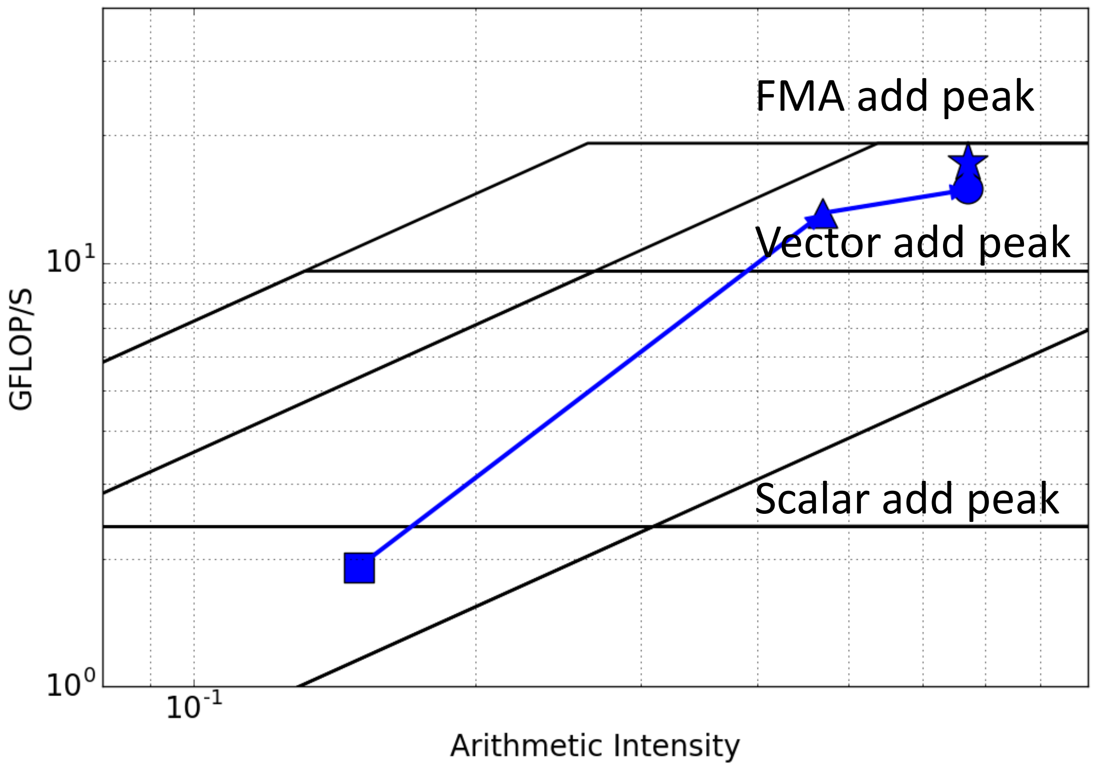
- **Initialize array inside the vector loop (if you can)**
- **Use threads for initialization**

~5% improvement
Higher if no. of particle increases

U.S. DEPARTMENT OF ENERGY | Office of Science

BERKELEY LAB

Interpolation: Optimization Path on Roofline



- Baseline Case (w/ Indirect access)
- Replace Indirect Access with Scalar Access
- Optimize Vector Length
- Access Grid Data in Scalar Chunks

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

U.S. DEPARTMENT OF ENERGY | Office of Science

BERKELEY LAB

Search: Vectorization – 'cycle' + SIMD

Problems:

- **Multiple exits due to 'cycle' statement prevents vectorization**
- **Assumed read after write (RAW) dependency prevent vectorization**

Optimization:

- **Replace exit condition with a logical mask**
- **Vectorize with omp simd directive, declare private arrays simd private**

Intel Compiler Vectorization Report

LOOP BEGIN at search.F90(62,8)
loop was not vectorized: loop with multiple exits cannot be vectorized unless it meets search loop idiom criteria

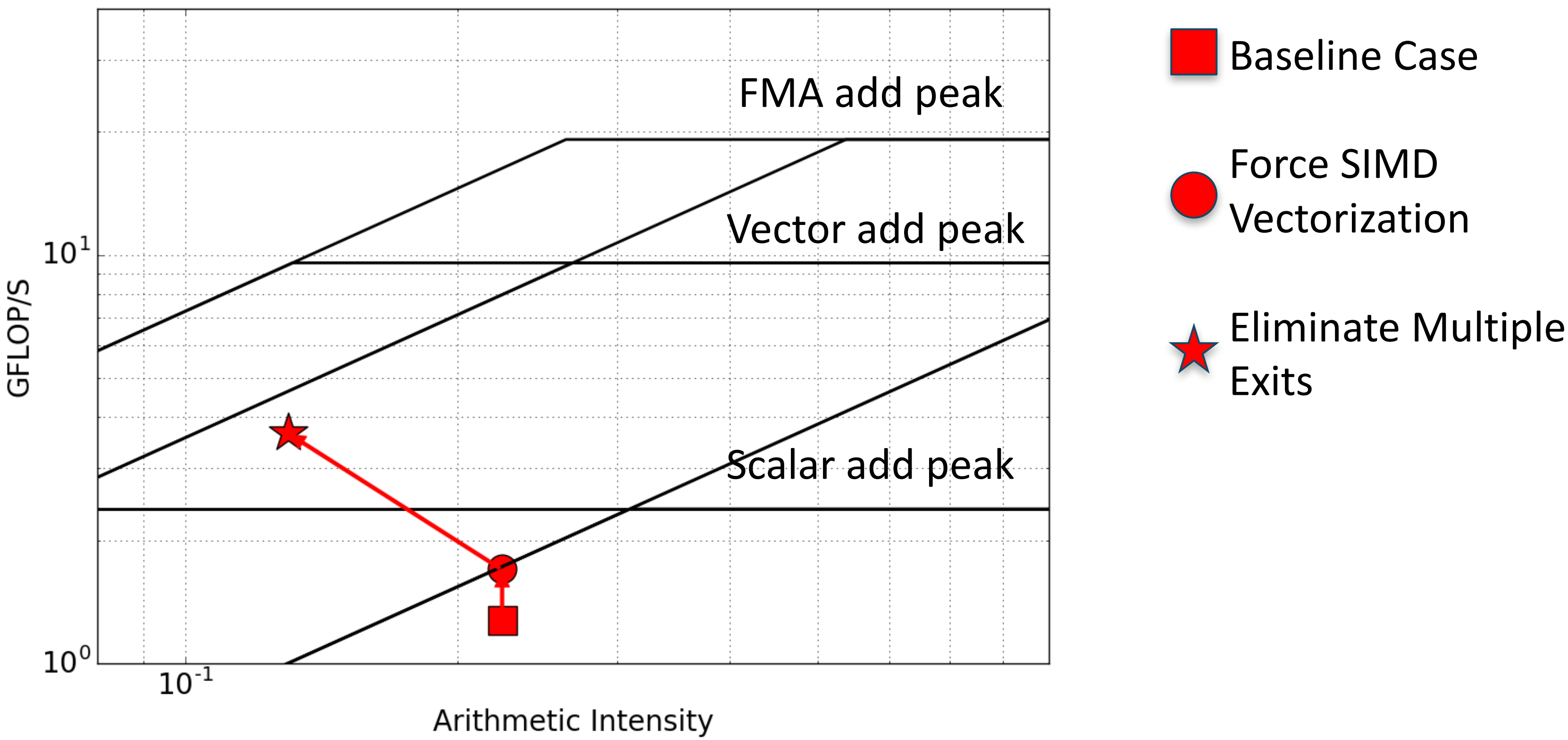
LOOP BEGIN at search.F90(66,8)
reference y(iv,1) has aligned access
reference y(iv,3) has aligned access
reference id(iv) has aligned access
reference continue_search(iv) has aligned access
data layout of a private variable bc_coords was optimized, converted to SoA
OpenMP SIMD LOOP WAS VECTORIZED
unmasked aligned unit stride loads: 4
unmasked aligned unit stride stores: 1

1.5x improvement

U.S. DEPARTMENT OF ENERGY | Office of Science

BERKELEY LAB

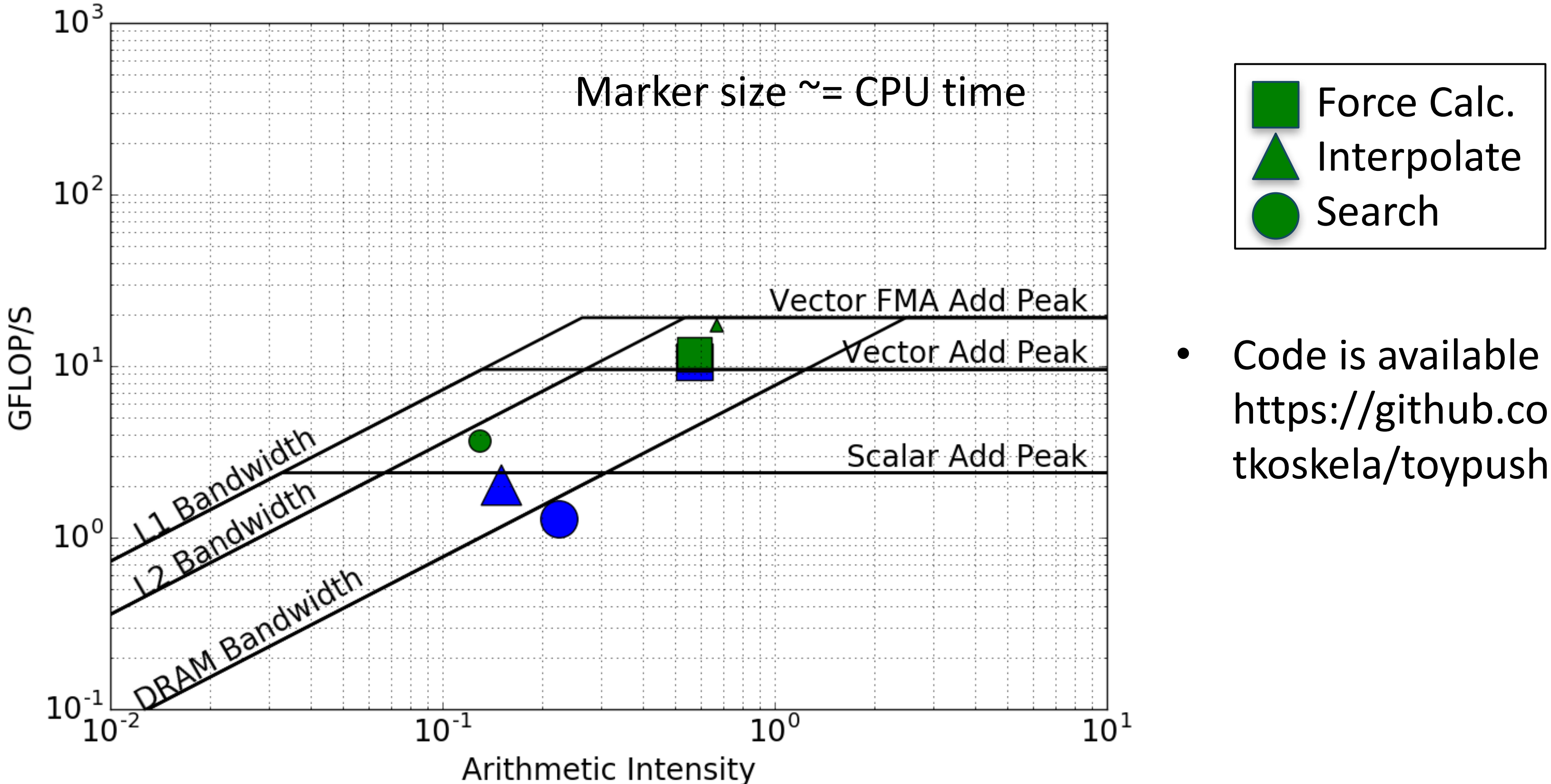
Search: Optimization Path on Roofline



- Forcing SIMD vectorization doesn’t work initially due to multiple exits
- Once exits are eliminated, code vectorizes

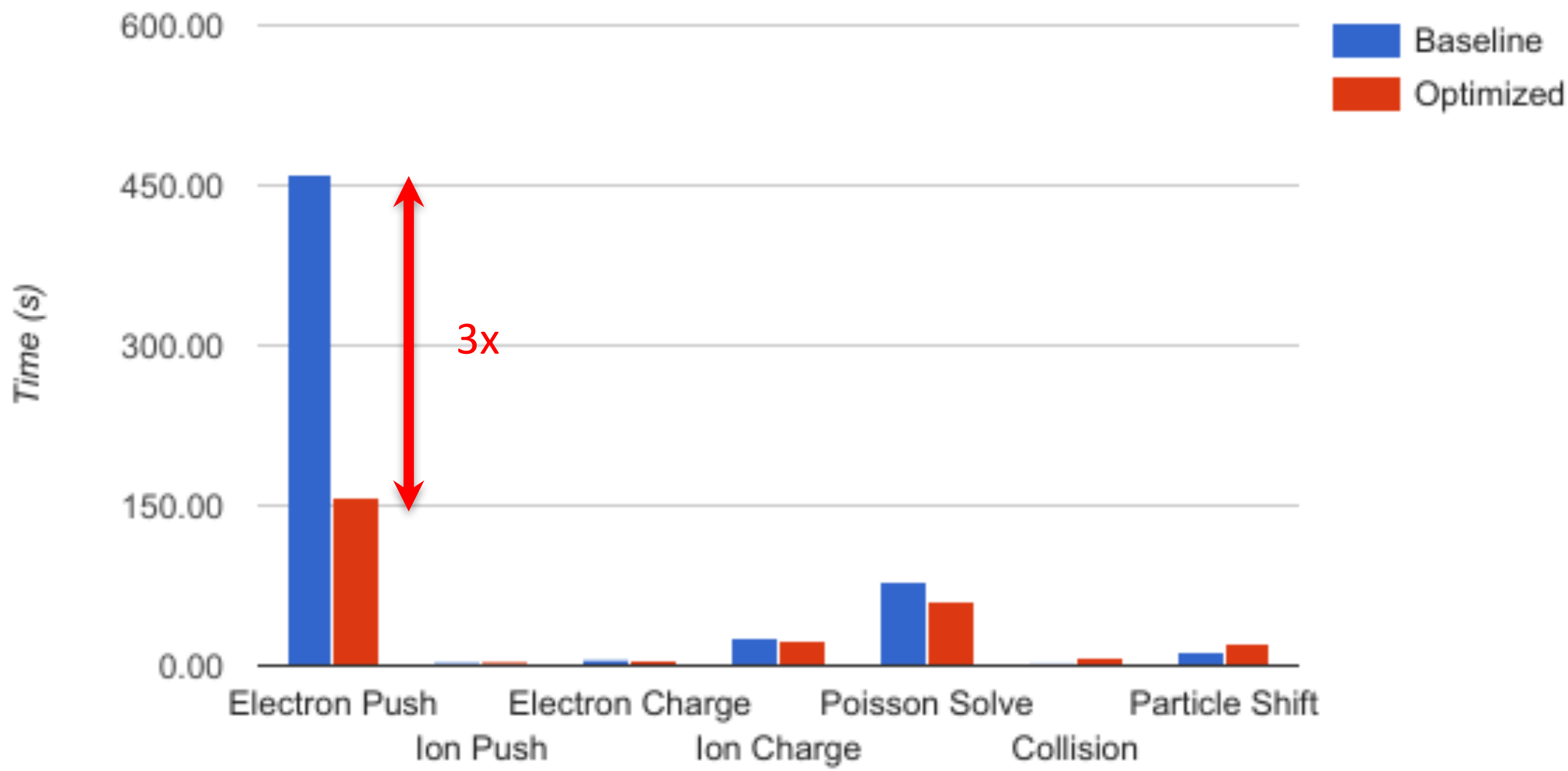
ToyPush: Optimized Performance

- **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



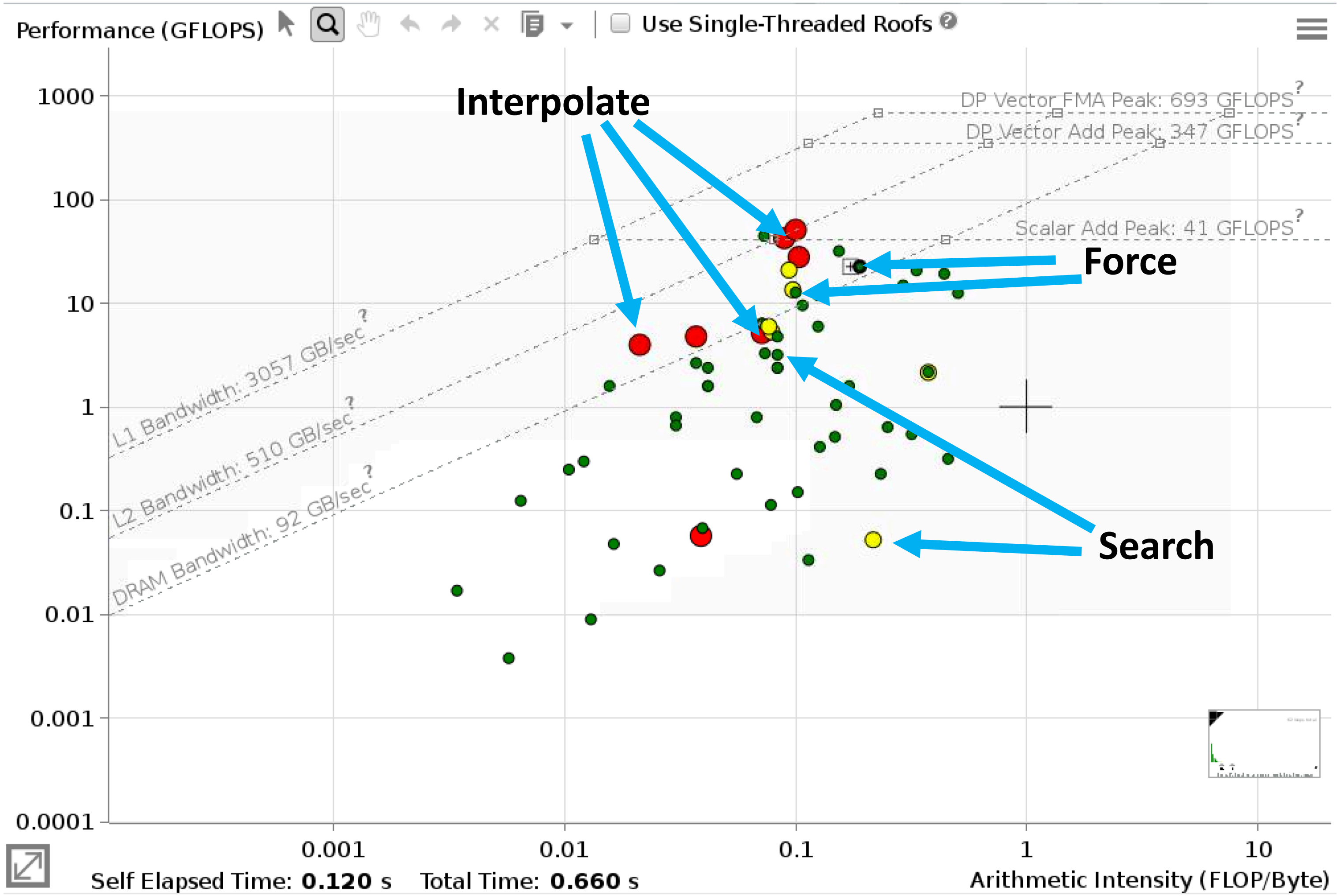
- Code is available at https://github.com/tkoskela/toypush

XGC1: Optimization Speedups (WIP)

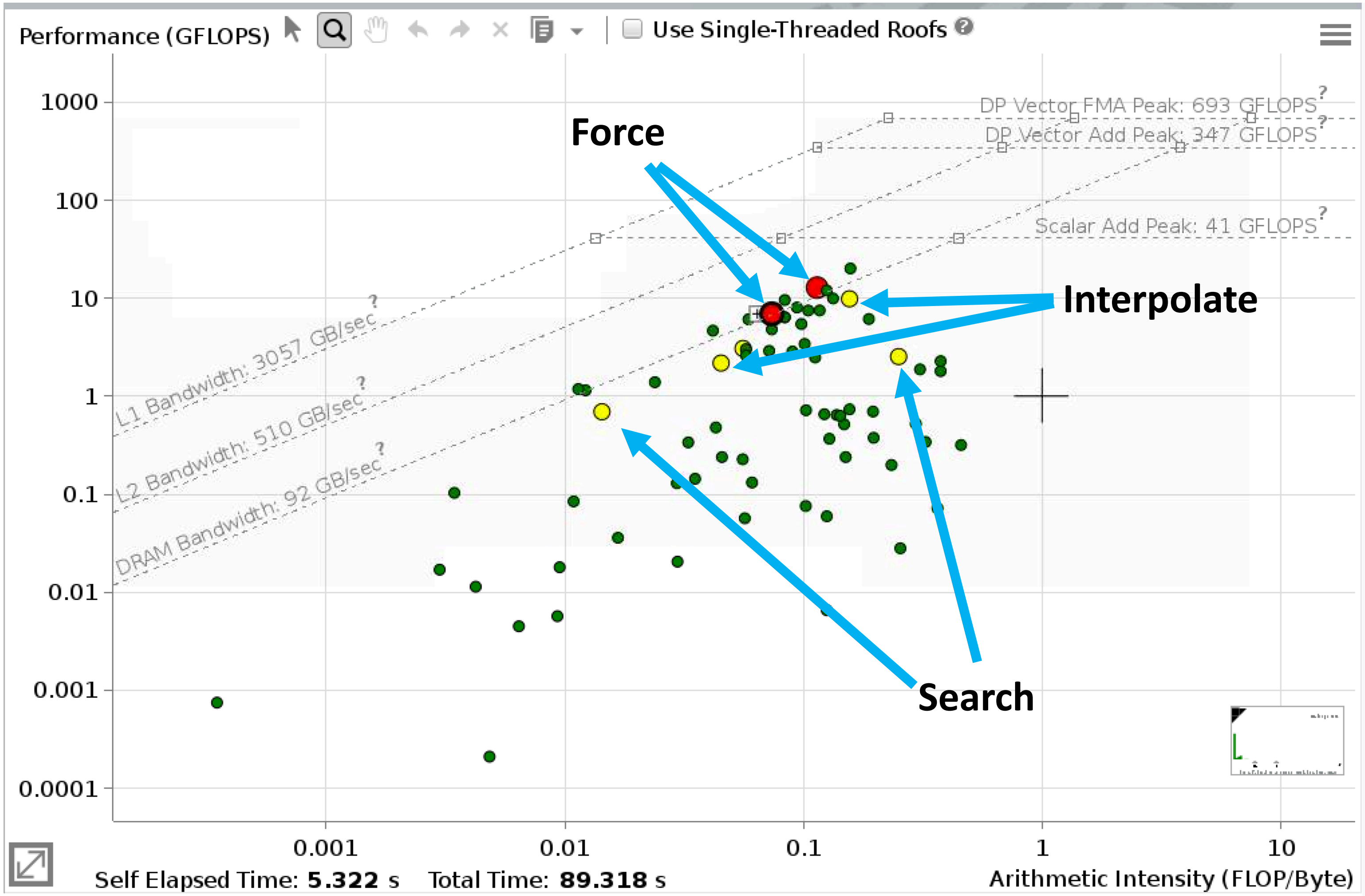


XGC1 Timings on 1024 Cori KNL nodes in Quad-Flat mode

XGC1: Optimized Performance on Roofline



XGC1: Code Profile on Roofline



Summary

- **XGC1 -> ToyPush -> XGC1**

- **Roofline Model can help**
  - Identify performance bottlenecks (compute, bandwidth, latency, *etc*)
  - Prioritize optimization efforts (routines, vectorization, memory access, *etc)*
  - Tell when to stop (realistic achievable performance, distance to roofs)

- **Intel Advisor can take care of the rest!**
  - Integrated compiler reports, static binary analysis (instruction set, data types, *etc*) and dynamic analysis (CPU sampling)
  - FLOPS/trip counts, vectorization efficiency, dependency and memory access pattern
  - Roofline charts of various flavors ☺
    - Original DRAM-based Roofline (DRAM <-> Core)
    - Cache-aware Roofline (L1 <-> Core) ✓
    - Cache-simulator based Roofline (L1, L2, LLC, MCDRAM and DRAM <-> Core) ✓