Case Studies: GPP, CoMD and XGC1

Charlene Yang
Lawrence Berkeley National Laboratory
cjyang@lbl.gov
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.
• 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

GPP - Original

Overlapping points at MCDRAM BW

Large gap between DRAM and MCDRAM AI

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

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

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

Look at other aspects (compute)
- Data level parallelism
- Thread level parallelism
CoMD - Better Vectorized

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:**

  - **Gather Fields from Mesh to Ions**
  - **Solve Fields on Mesh**
  - **Ion Push**
  - **Deposit Charge From Particles to Mesh**
  - **Collision Operator**
  - **Electron Push Sub-Cycling**
    - push electrons without updating fields or collisions – only field gather and push
    - ~50x

  *Computation*  
  *Mapping*
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

Marker size $\sim$ CPU time

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
  
  \[ \text{efield}(j, \text{tri}(i, \text{itri}(iv))) \]

- Unaligned access -> align at compile time

- Improved vectorization efficiency

---

LOOP BEGIN at interpolate_aos.F90(67,48)

reference \text{itri}(iv) has unaligned access
reference \text{y}(iv,1) has unaligned access
reference \text{y}(iv,3) has unaligned access
reference \text{evec}(iv,icomp) has unaligned access
reference \text{evec}(iv,icomp) has unaligned access

.....

irregularly indexed load was generated for the variable \text{<grid_mapping_}(1,3,\text{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
ToyPush - Interpolation

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

• Eliminate multiple exits, ‘cycle’, and RAW (read after write) dependency

• Force SIMD vectorization with `omp simd`
**ToyPush: Optimized Profile**

- **Force Calculation**: 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**

**Diagram:**
- Marker size ~ CPU time
- **GFLOPS** vs. **Arithmetic Intensity**
  - Vector FMA Add Peak
  - Vector Add Peak
  - Scalar Add Peak
  - L1 Bandwidth
  - L2 Bandwidth
  - DRAM Bandwidth
XGC1: Merge ToyPush Changes (WIP)

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

Baseline
Optimized

3x
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!