lrz

Leibniz Supercomputing Centre

of the Bavarian Academy of Sciences and Humanities

HPC code optimisation workshop

The Roofline Model | 3 November 2021 | Jonathan Coles

Talk Outline

- What are we trying to optimize?
- What limits application performance?
- What is the cache-aware roofline model?
- How does it help identify performance problems?
- What can be done to improve performance?

Choosing a metric

Before we talk about optimization, it is important to decide what metric we want to focus on.

- Time to solution
  - Algorithmic efficiency
    - Is a better algorithm possible $O(n^2) \rightarrow O(n)$?
  - CPU Performance
    - Is the CPU reaching its theoretical peak performance?
    - Is there a bottleneck in memory access or processing?
- Memory requirements
  - Is a different data structure needed?
- Other resources
  - Compute hardware → More cores / socket?
  - Energy → Different CPU or frequency?

This talk will focus here

Note: Both *algorithmic efficiency* and *CPU performance* may improve the *time to solution*, but not necessarily!

What are we trying to optimize?

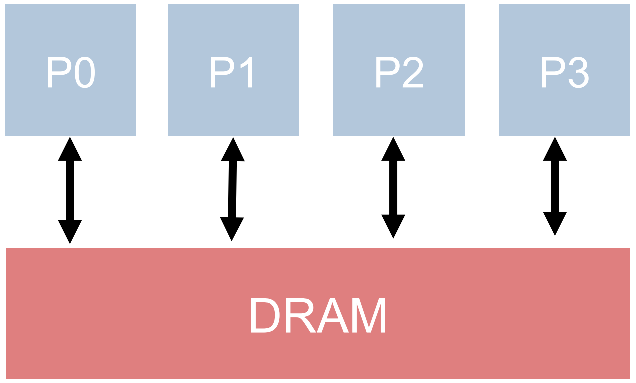
Defining CPU Performance

- For this talk:
  - Fundamental algorithm, data structure, etc., are assumed fixed.
  - Focus is on measuring and improving observed CPU Performance.
  - Performance is defined as FLOP/s.
    - Other definitions are possible to measure different scenarios.

- Measured performance P is limited by:
  - The maximum saturated bandwidth b_s to move data from memory to the CPU (byte/s).
    - Usually bandwidth from DRAM or caches.
  - The intensity of work I for each byte moved (FLOP/byte).
    - How many times we reuse the same float in some set of floating-point operations.
  - The theoretical maximum performance P_peak of the CPU (FLOP/s).
    - Affected by frequency, number of cores, vectorization or instructions like fused multiply-add.

What limits application performance?

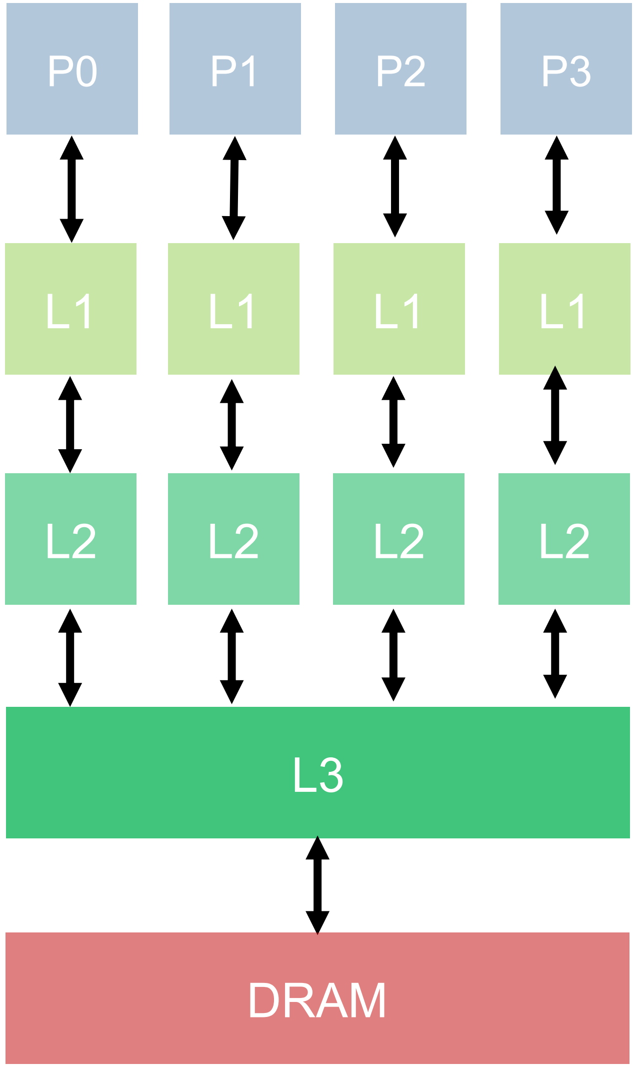
Memory



- Simple view: A large storage area of dynamic random access memory (DRAM), directly connected to the processing cores.
- However fast memory is expensive. To increase size at a reduced cost DRAM is relatively slow. Also located physically far away from the cores in the computer which increases access time.

What limits application performance?

Memory Hierarchy



- One solution is to place several layers of high-speed, limited-space memory known as cache between the cores and DRAM.
- The cache contains a working copy of part of memory.
- When another region of DRAM is needed, cached copies of older regions may be evicted and written back to higher cache levels or DRAM.

- Modern systems usually have three levels: L1, L2, L3.
- At each level closer to the core, the bandwidth to the cores increases, but the size decreases.

What limits application performance?

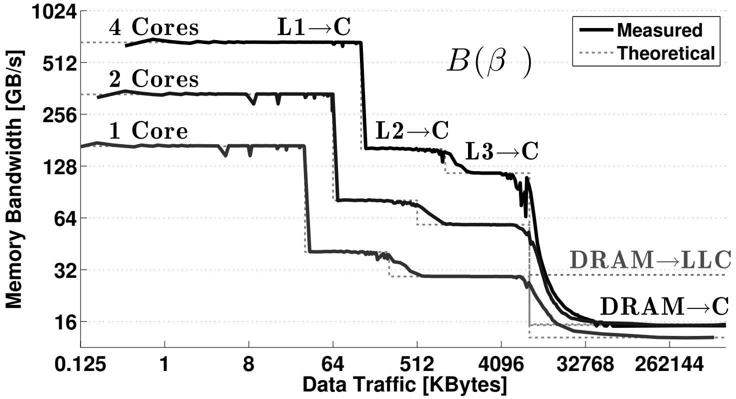
SuperMUC-NG Processor

```
$ likwid-topology
--------------------------------------------------------------------------------
CPU name: Intel(R) Xeon(R) Platinum 8174 CPU @ 3.10GHz
CPU type: Intel Skylake SP processor
CPU stepping:       4
********************************************************************************
Hardware Thread Topology
********************************************************************************
Sockets:            2
Cores per socket:   24
Threads per core:   2
********************************************************************************
Cache Topology
********************************************************************************
Level:                        1
Size:                         32 kB
--------------------------------------------------------------------------------
Level:                        2
Size:                         1 MB
--------------------------------------------------------------------------------
Level:                        3
Size:                         33 MB
--------------------------------------------------------------------------------
********************************************************************************
NUMA Topology
********************************************************************************
NUMA domains:                 2
--------------------------------------------------------------------------------
```

Shared among all cores of a socket!

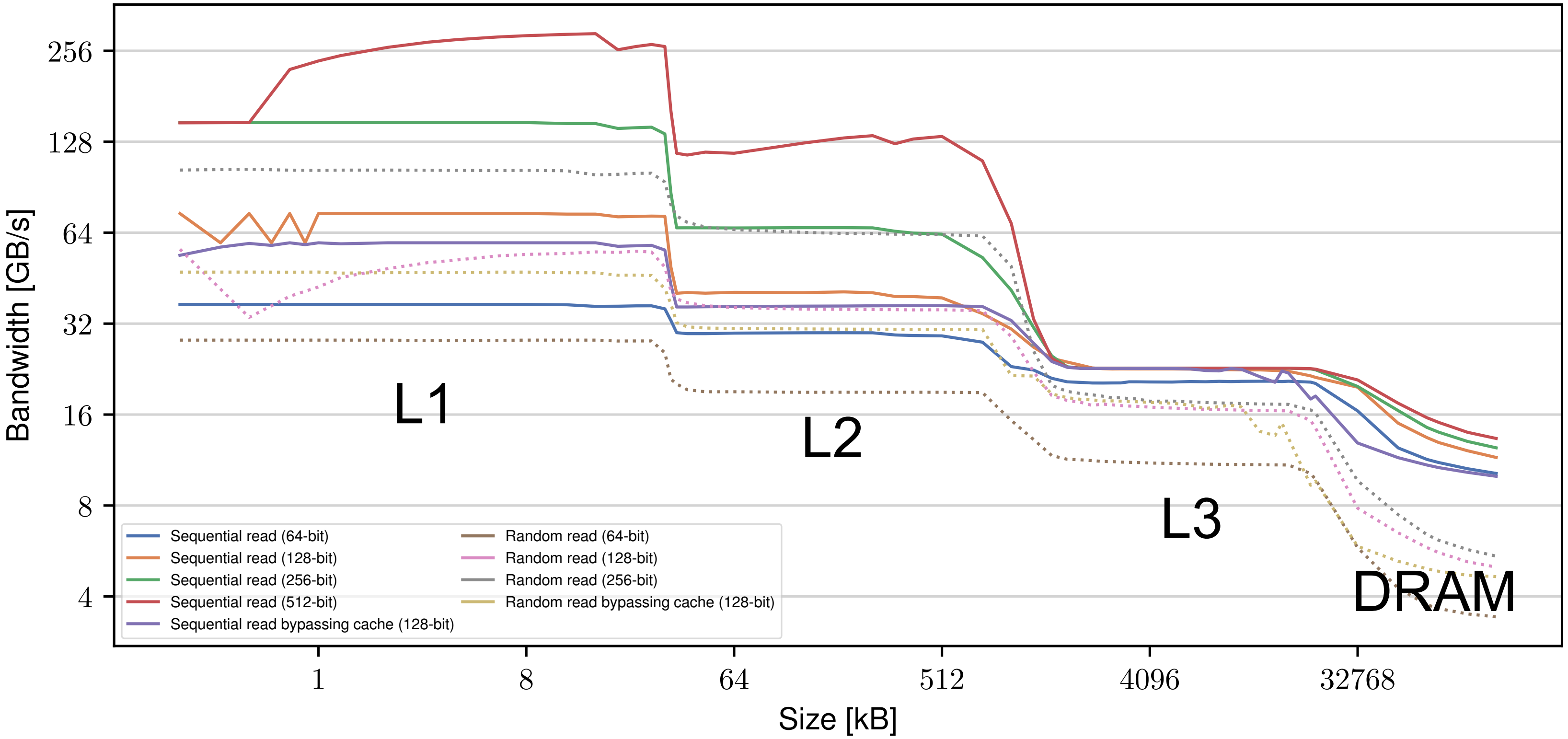
What limits application performance?

Memory Bandwidth



```
Level:                     1
Size:                      32 kB
--------------------------------
Level:                     2
Size:                      1 MB
--------------------------------
Level:                     3
Size:                      33 MB
```

SuperMUC-NG:
Single core of Intel(R) Xeon(R) Platinum 8174 CPU



Cache-aware Roofline model: Upgrading the loft. Ilic et al. (2013).

What limits application performance?

Operational Intensity

I = 1 FLOP / ((2+1)*8 bytes) = 1:24

```
for (i=0; i < N; i++)
    c[i] = a[i] + b[i];
```

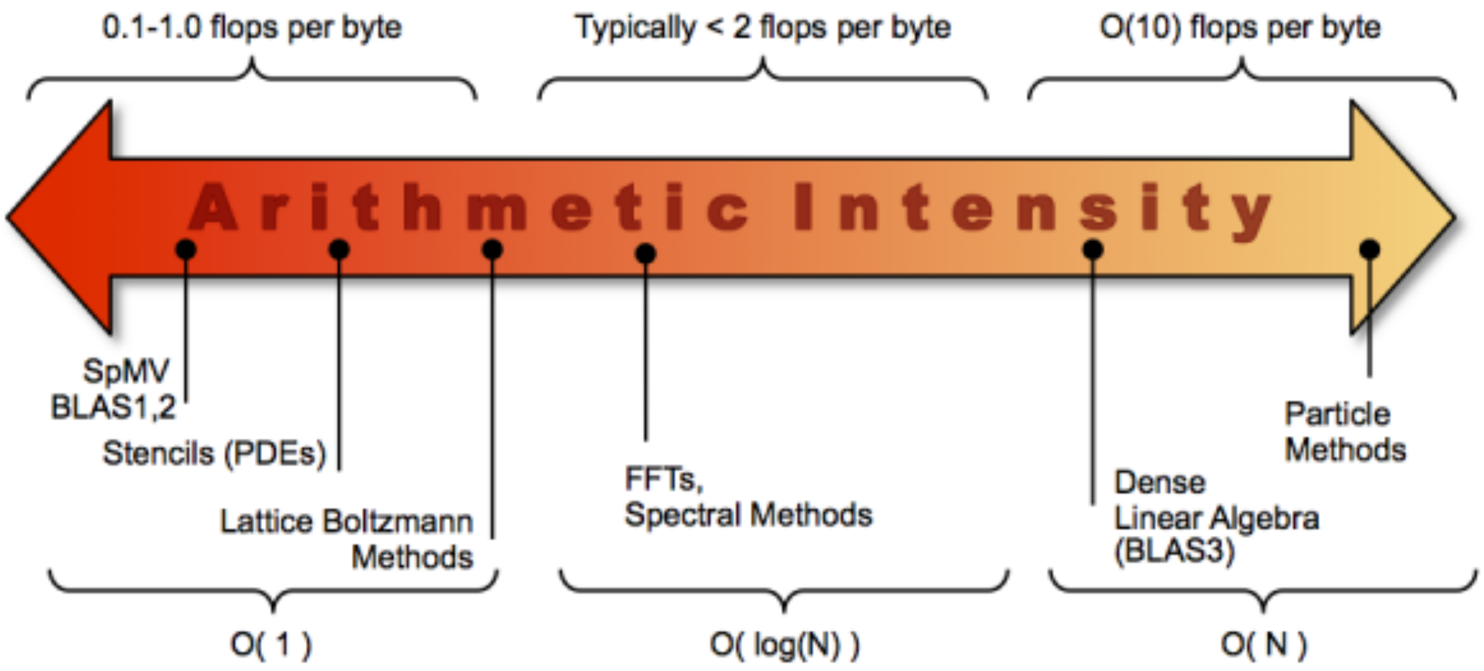
I = 2 FLOP / ((3+1)*8 bytes) = 1:16

Performance benefit from FMA

```
for (i=0; i < N; i++)
    d[i] = a[i] + b[i]*c[i];
```

I = 5 FLOP / ((5+1)*8 bytes) ~ 1:12

```
for (i=1; i < N-1; i++)
for (j=1; j < M-1; j++)
    next[i][j] = -4*prev[i  ][j  ] +
                    prev[i  ][j-1] +
                    prev[i  ][j+1] +
                    prev[i-1][j  ] +
                    prev[i+1][j  ];
```



https://crd.lbl.gov/divisions/amcr/computer-science-amcr/par/research/roofline/introduction

- **Operational Intensity** is more common term.
  May not be interested in only arithmetic in general.
- Calculation includes +1 for store back to memory.

What limits application performance?

Theoretical CPU Performance

P = Freq(GHz) × Cores/Socket × FP/Hz

$$\log P = \log \text{Freq(GHz)} + \log \text{Cores/Socket} + \log \text{FP/Hz}$$

SIMD increases FP/Hz

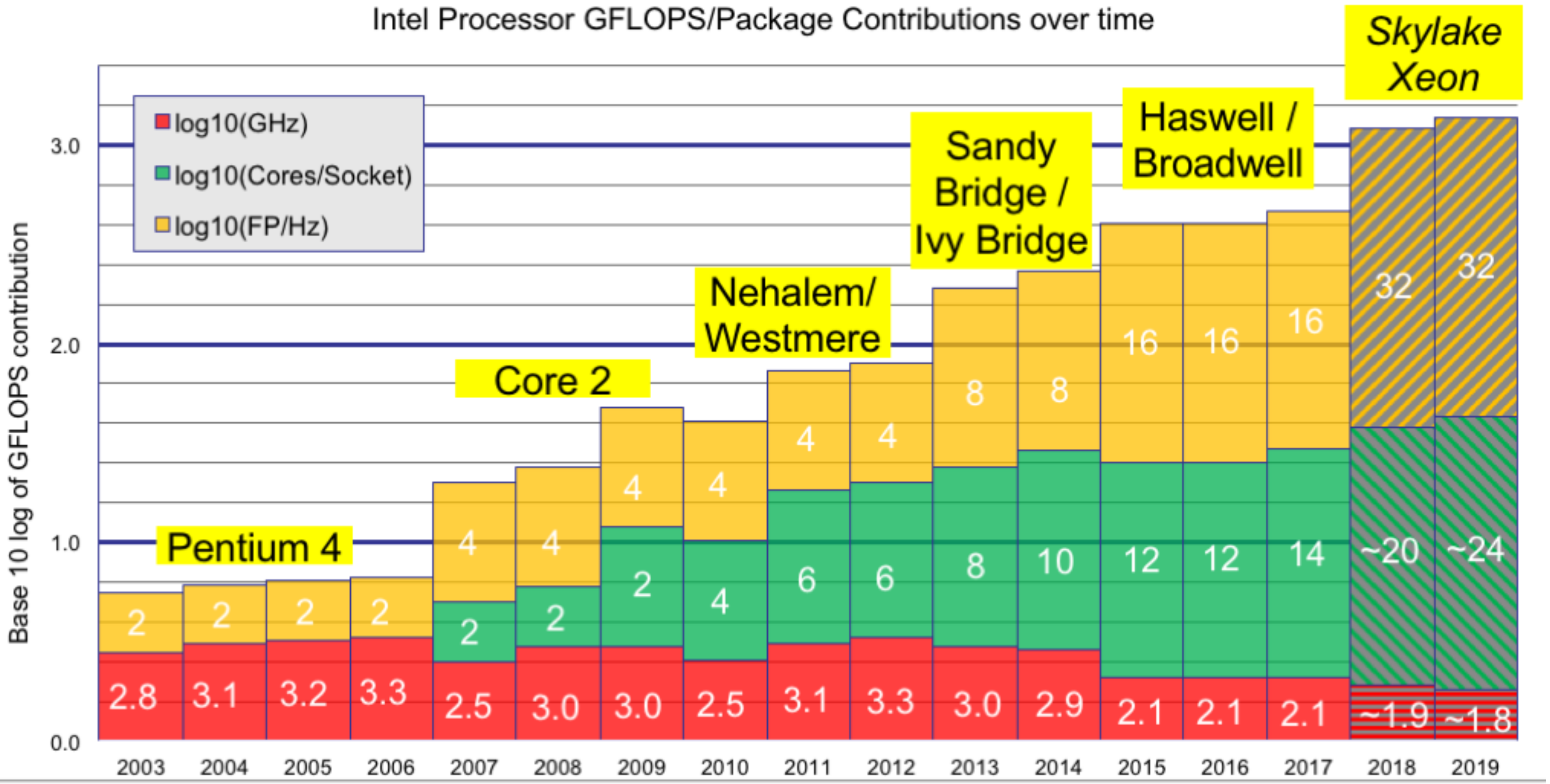
A[i+0]	A[i+1]	A[i+2]	A[i+3]

+

B[i+0]	B[i+1]	B[i+2]	B[i+3]

=

C[i+0]	C[i+1]	C[i+2]	C[i+3]



TACC

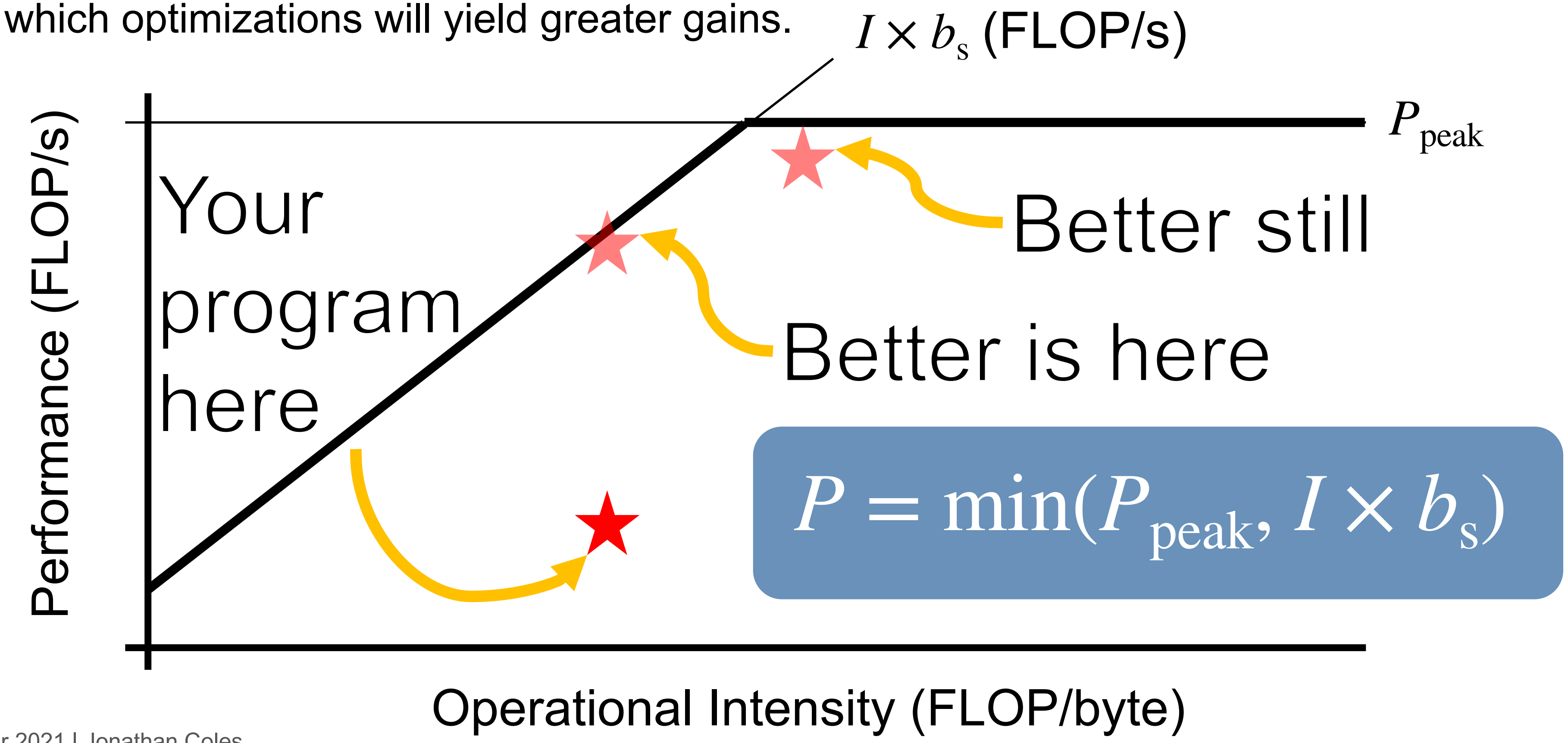
THE UNIVERSITY OF TEXAS AT AUSTIN

TEXAS ADVANCED COMPUTING CENTER

https://sites.utexas.edu/jdm4372/2016/11/22/sc16-invited-talk-memory-bandwidth-and-system-balance-in-hpc-systems/

The Roofline Model

- The roofline model
  - Provides a visual depiction of the factors limiting application performance.
  - Placing application performance into the model can suggest ways to improve performance or show when performance is limited by hardware.
  - May suggest which optimizations will yield greater gains.

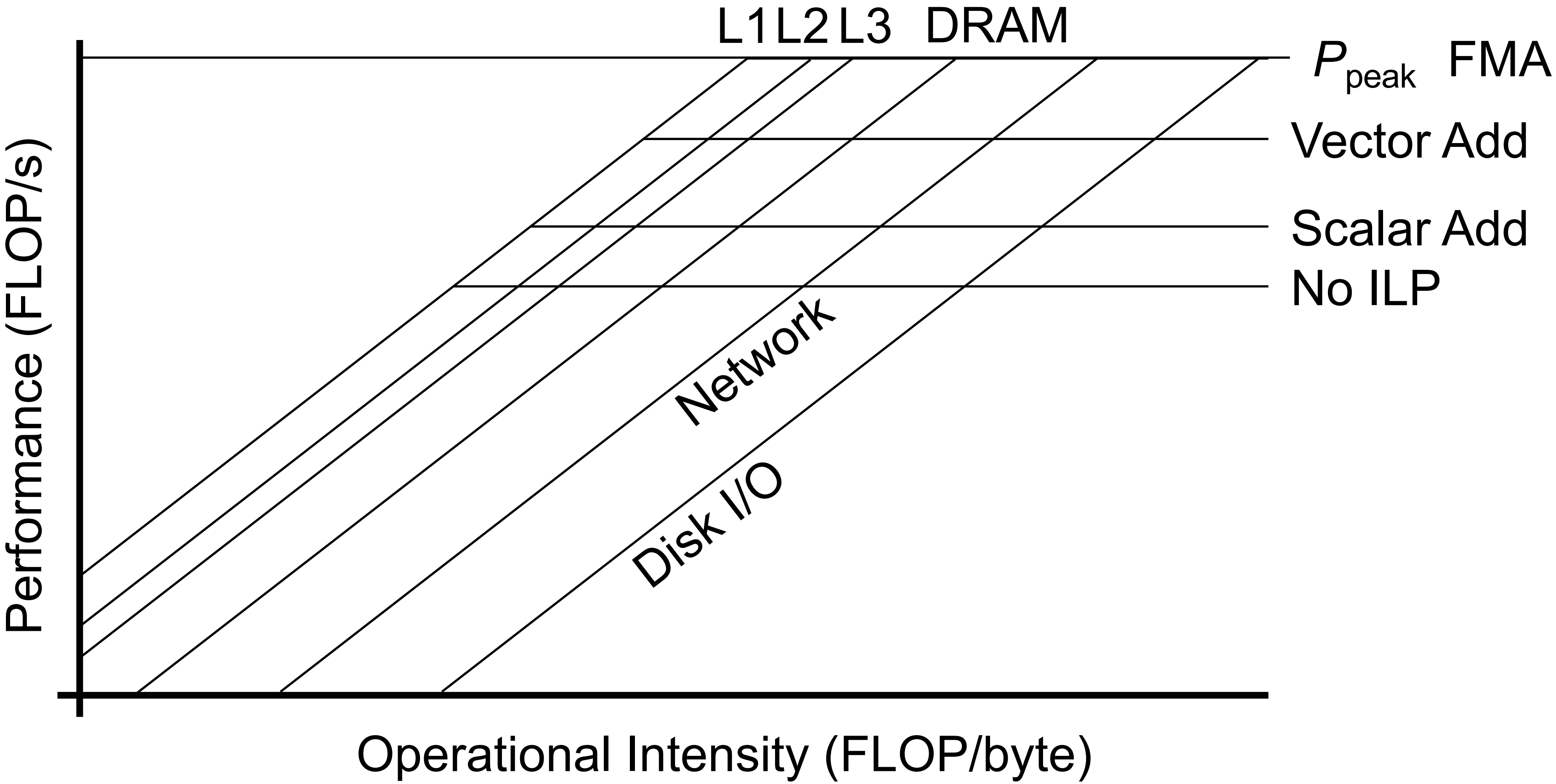


$$P = \min(P_{\text{peak}}, I \times b_s)$$

The cache and performance limits

- The Cache-Aware Roofline Model discussed here measures bandwidth from memory to core.
- The original model measured DRAM to cache bandwidth.

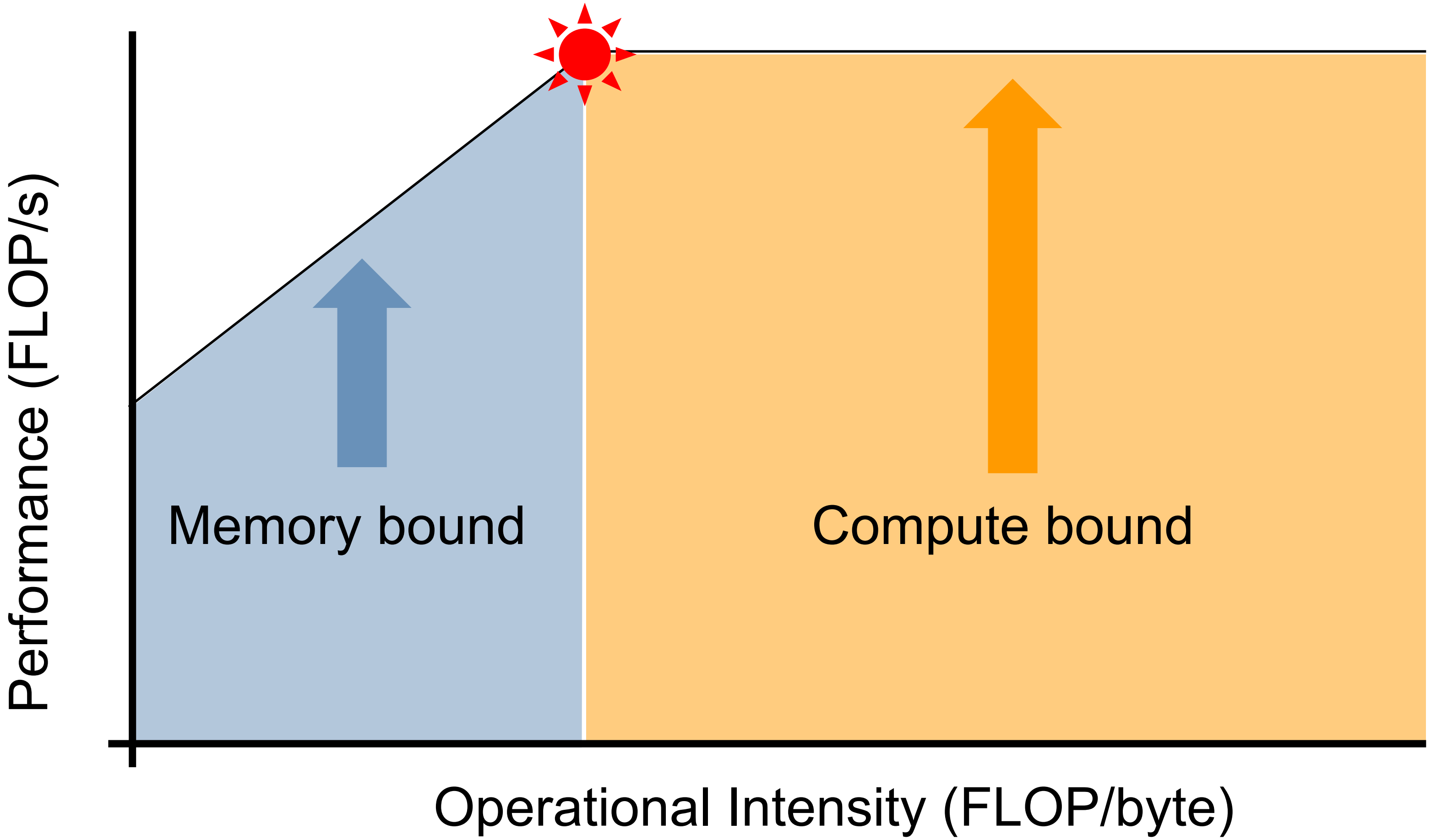
- Faster caches can raise the diagonal roofline.
- Slower components can lower the diagonal roofline.
- Instruction level parallelism (ILP) or vectorization can raise or lower the horizontal roofline.



NOTE:
Axes are log-log.
Different bandwidth (diagonal) bounds are parallel.

How does it help identify performance problems?

- Codes lying under the diagonal are ultimately limited by some form of memory access.
- Codes under the horizontal line are ultimately compute bound.
- Moving higher requires different solutions depending on where an application lands on this plot.
- The “knee” is the least intensity that achieves the best performance given the compute and memory bounds that are intersecting.



How does it help identify performance problems?

Some examples

★ Limited by DRAM. Possibly not fitting into cache.

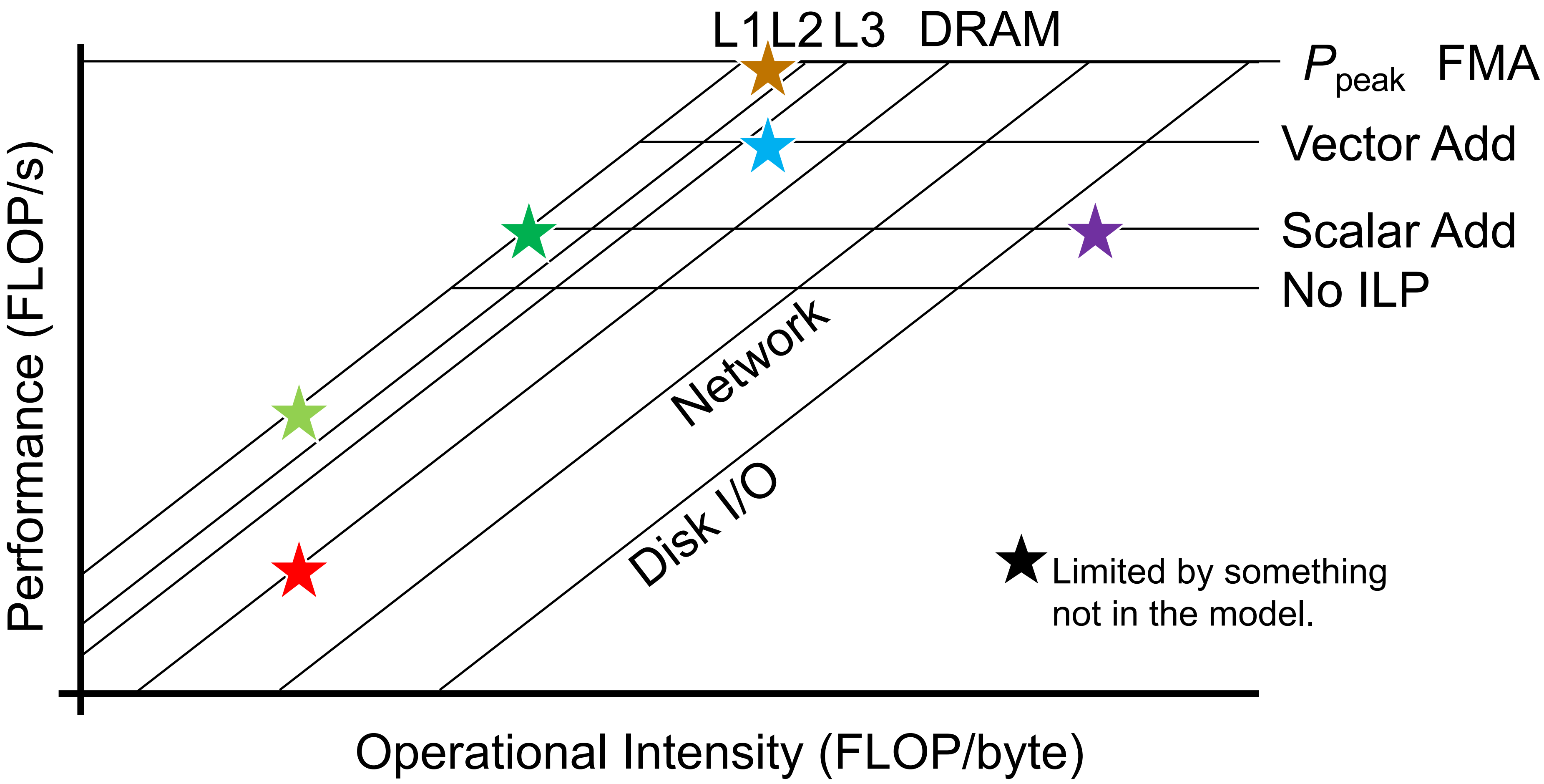
★ Broke through DRAM ceiling with better cache use or reduced data. Now limited by L1. Could possibly benefit from increased intensity.

★ Improved intensity and not limited by L2. Still only using scalar ops.

★ Much higher intensity but no performance benefit. Still limited by scalar ops.

★ Reduced intensity but uses vector ops to break through scalar ceiling.

★ Better cache use and benefits from FMA. Now reaches peak performance.



★ Limited by something not in the model.

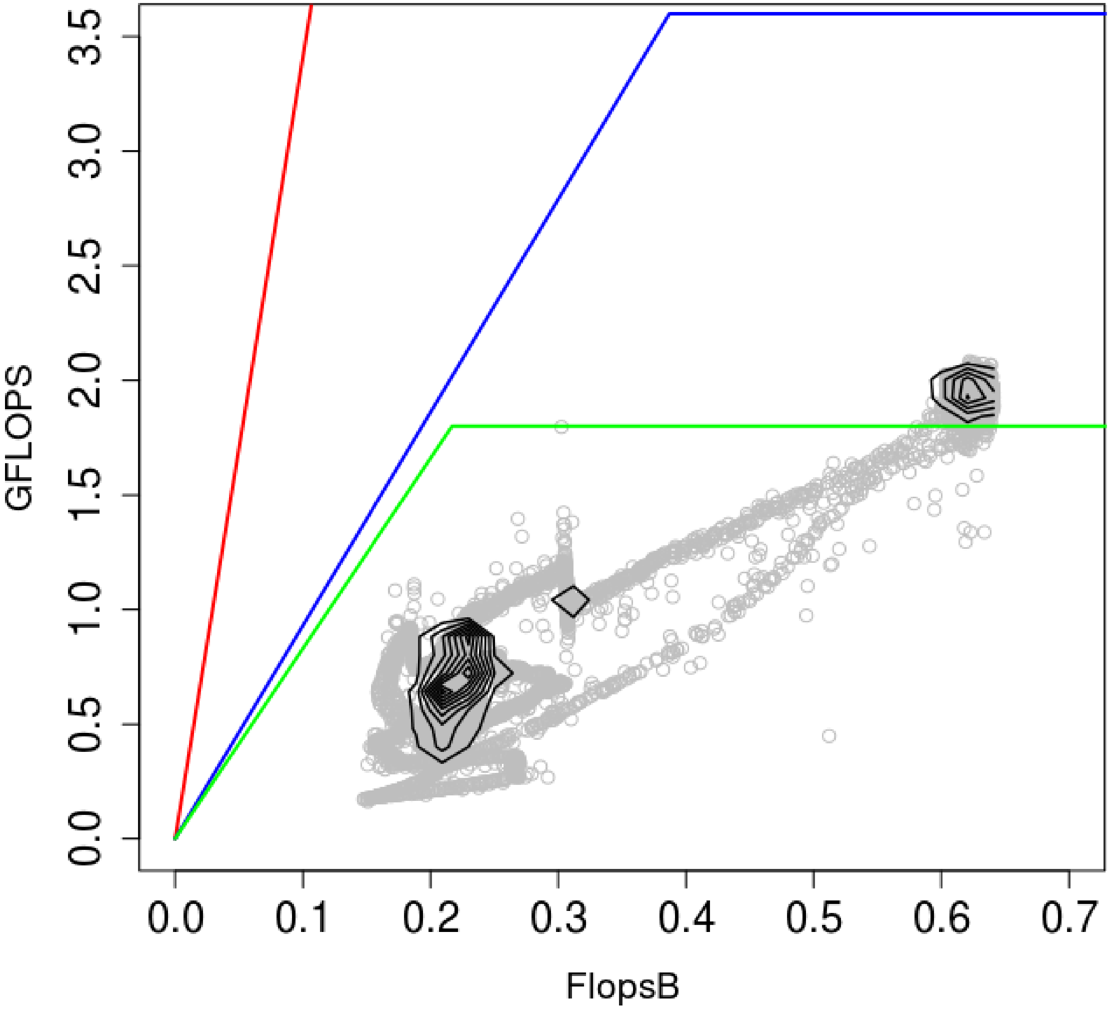
NOTE: Axes are log-log. Different bandwidth (diagonal) bounds are parallel.

How does it help identify performance problems?

Dynamic application behavior

- Be aware that a single data point does not necessarily characterize the entire application.
- In the figure, the NPB SP application has two distinct behaviors with transitions between them.

- NOTE: Linear axes are used here. The colored rooflines represent memory levels.



Lorenzo et al. “Using an extended Roofline Model to understand data and thread affinities on NUMA systems.” (2014).

What can be done to improve performance?

Placing an application in the model

- As with any optimization we need to start by measuring our application.
- Estimating intensity can sometimes be done by hand for relatively simple kernels. More complex kernels will need tools.
- Measuring the CPU performance is harder to do without the use of tools.
- Intel Advisor can generate a roofline model and measure the application.
- LIKWID can provide statistics that can be used to make a roofline model.
- Kerncraft can also be used (based on LIKWID).

What can be done to improve performance?

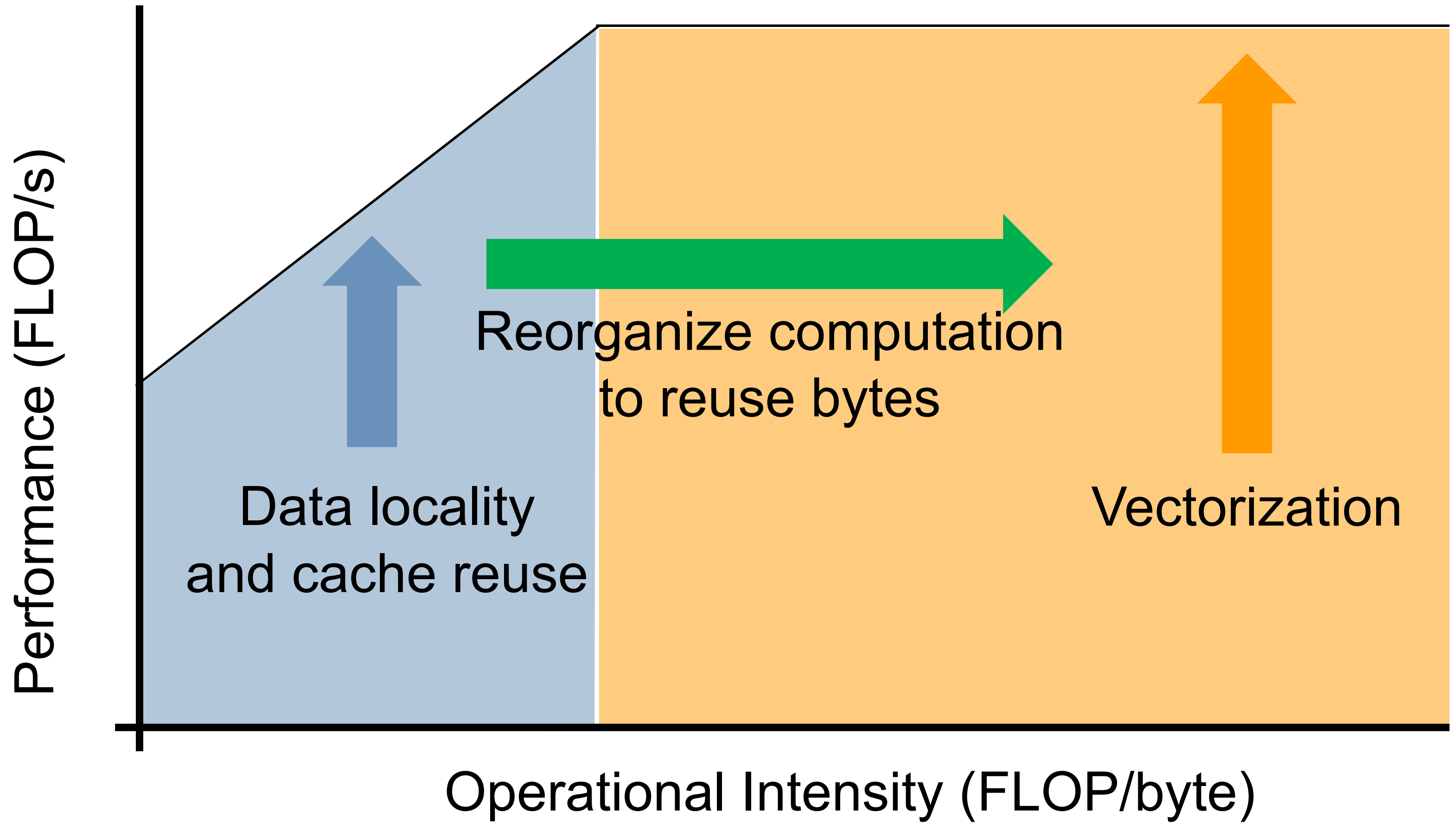
Applying the roofline model

Some of these suggestions the compiler can do itself, but it may need help.
Best to turn on optimization/vectorization reporting when available!

- Break through the horizontal ceiling (increase FLOP/s)
  - FMA, SIMD vectorization / Instruction level parallelism (ILP)
  - Use more cores.
- Break through the bandwidth ceiling (increase bytes/s)
  - Increase data locality. Avoid long-range NUMA accesses. First touch policy.
  - Increase cache reuse. Loop blocking. Long unit-stride loops to use memory prefetcher.
  - Use smaller data structures to keep relevant data in cache. SoA vs. AoS.
  - Use more cores.
- Move to the right to higher intensity (increase FLOP/byte) then up.
  - Reorder code to use the same float multiple times. Loop unrolling. Store in registers.
  - May need algorithmic changes.

What can be done to improve performance?

The take away plot



Data locality and cache reuse

Reorganize computation to reuse bytes

Vectorization