

Energy analysis of plasma physics algorithms

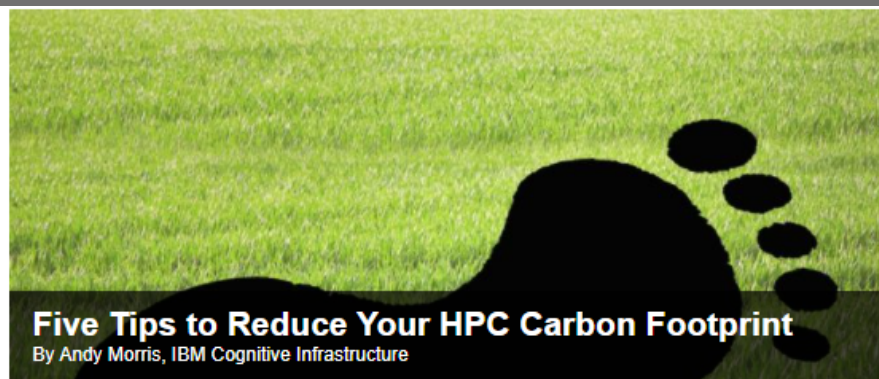
I. Chernykh, I. Kulikov, M. Boronina,
A. Efimova, V. Vshivkov, D. Weins

**Institute of Computational Mathematics
and Mathematical Geophysics SB RAS,
Siberian Supercomputer Center SB RAS**

Carbon dioxide emissions are the primary driver of global climate change. It's widely recognized that to avoid the worst impacts of climate change, the world needs to urgently reduce emissions. But, how this responsibility is shared between regions, countries, and individuals has been an endless point of contention in international discussions. This debate arises from the various ways in which emissions are compared: as annual emissions by country; emissions per person; historical contributions; and whether they adjust for traded goods and services. These metrics can tell very different stories

<https://ourworldindata.org/co2-emissions>

- ▼ Home
- ▼ Technologies
- ▼ Sectors
- ▼ COVID-19
- ▼ AI/ML/DL
- ▼ Exascale
- ▼ Specials
- ▼ Resource Library
- ▼ Podcast
- ▼ Events
- ▼ Job Bank
- ▼ About
- ▼ Our Authors
- ▼ Solution Channels
- ▼ Subscribe



Five Tips to Reduce Your HPC Carbon Footprint

By Andy Morris, IBM Cognitive Infrastructure

August 6, 2019

PROVIDED BY IBM

Faced with more extreme weather events and dangerous heat episodes concerns about global warming and carbon emissions are on the rise. Corporate and cloud data centers are, unfortunately, part of the problem. Today, global data centers account for as almost as much CO₂ emissions as the entire airline industry and consume more electricity than the country of Iran⁽¹⁾.

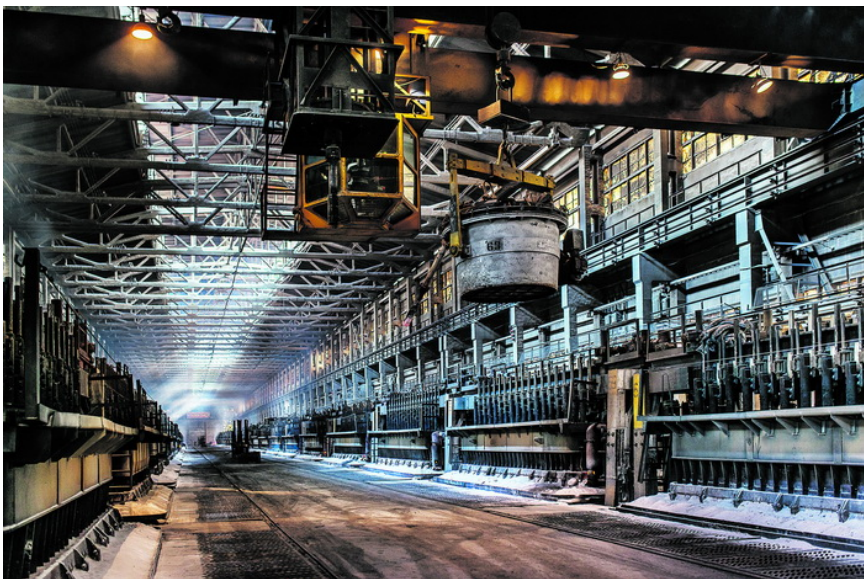
What's most concerning is the rate of growth. Fueled by growing internet traffic, AI and IoT, some worrisome models predict that information and communications technology may grow to 20% of global electricity consumption by 2030 from just 2% today⁽²⁾. Getting to carbon-neutral data centers will be essential to combat global climate change.

Read also: [HPC Brings the Heat to Impacts of Global Warming](#)

Waste in the cloud adds to the problem

Gartner projects that spending on cloud IaaS services will grow 27.6% to reach \$39.5 billion in 2019⁽³⁾, and according to InfoWorld, as much as 35% of cloud spending will be wasted⁽⁴⁾. The ease with which cloud services can be consumed is part of the problem. Users are frequently tempted to use cloud resources even when capacity exists on-premises. It's not uncommon for users to over-provision compute instances and storage or start cloud services and forget them.

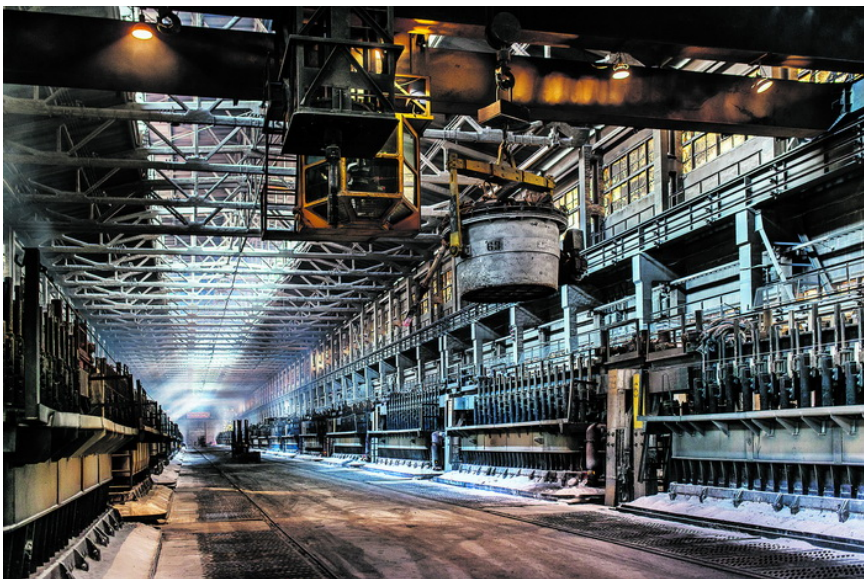
Toward a carbon-neutral data center



Production of
1 tonne of aluminum = 17mW



Fugaku supercomputer = 29,9 mW



Production of
1 tonne of aluminum = 17mW



Fugaku supercomputer = 29,9 mW

Typical AI production
plant – 1 000 000 tones per
year

Most of computing's carbon emissions are coming from manufacturing and infrastructure



Carole-Jean Wu, Udit Gupta

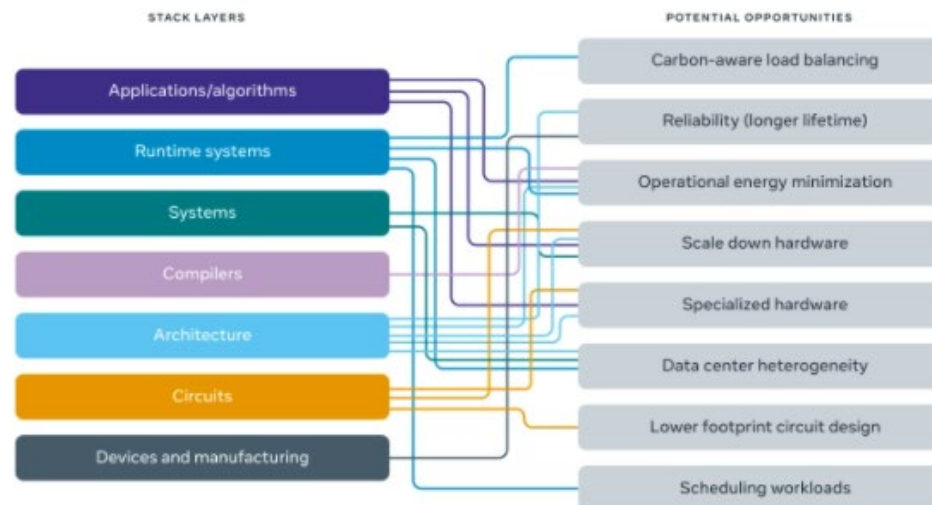
March 2, 2021

Reducing computing's carbon footprint isn't all about optimizing hardware and software.

[XPRIZE: The Future of Sustainable Computing Infrastructure](#) | [Apple](#) | [Google](#) | [Microsoft](#) | [Amazon](#) | [Facebook](#) | [Twitter](#) | [LinkedIn](#) | [YouTube](#) | [Instagram](#) | [TikTok](#) | [Snapchat](#) | [WhatsApp](#) | [Telegram](#) | [Signal](#) | [Proton](#) | [Nextcloud](#) | [ownCloud](#) | [Synology](#) | [QNAP](#) | [Asustor](#) | [Toshiba](#) | [Seagate](#) | [Western Digital](#) | [Samsung](#) | [HGST](#) | [Micron](#) | [SK Hynix](#) | [Kioxia](#) | [Toshiba](#) | [Seagate](#) | [Western Digital](#) | [Samsung](#) | [HGST](#) | [Micron](#) | [SK Hynix](#) | [Kioxia](#)

Recommended Reading

5 factors that will change how we work in 2021 and beyond



[nature](#) > [nature astronomy](#) > [comment](#) > [article](#)

[Comment](#) | [Published: 10 September 2020](#)

The ecological impact of high-performance computing in astrophysics

[Simon Portegies Zwart](#) 

Nature Astronomy **4**, 819–822 (2020) | [Cite this article](#)

1908 Accesses | **10** Citations | **552** Altmetric | [Metrics](#)

Computer use in astronomy continues to increase, and so also its impact on the environment. To minimize the effects, astronomers should avoid interpreted scripting languages such as Python, and favour the optimal use of energy-efficient workstations.

Green Algorithms

How green are your computations?

Details about your algorithm

To understand how each parameter impacts your carbon footprint, check out the formula below and the [methods article](#)

Runtime (HH:MM)

Type of cores

Number of cores

Model

Memory available (in GB)

Select the platform used for the computations

Select location

Do you know the real usage factor of your CPU?
☐ Yes ☒ No

Do you know the Power Usage Efficiency (PUE) of your local data centre?
☐ Yes ☒ No

Do you want to use a Pragmatic Scaling Factor?
☐ Yes ☒ No

[Reset](#)

303.10 g CO2e
Carbon footprint

2.28 kWh
Energy needed

0.33 tree-months
Carbon sequestration

1.73 km
in a passenger car

1 %
of a flight Paris-Lc

Share your results with [this link!](#)

Computing cores VS Memory

Category	Percentage
CPU	70.1%
Memory	29.9%

How the location impacts footprint

Location	Emissions (gCO2e)
Switzerland	~100
Sweden	~200
France	~300
Canada	~400
your algorithm	~500
United Kingdom	~1000
USA	~1200
China	~1400
India	~1600

<http://www.green-algorithms.org/>

<https://github.com/GreenAlgorithms/green-algorithms-tool>

<https://www.advancedsciencenews.com/measuring-computers-carbon-footprint-with-green-algorithms/>

The formula

The carbon footprint is calculated by estimating the energy draw of the algorithm and the carbon intensity of producing this energy at a given location:

$$\text{carbon footprint} = \text{energy needed} * \text{carbon intensity}$$

Where the energy needed is:

$$\text{runtime} * (\text{power draw for cores} * \text{usage} + \text{power draw for memory}) * \text{PUE} * \text{PSF}$$

The power draw for the computing cores depends on the model and number of cores, while the memory power draw only depends on the size of memory available. The usage factor corrects for the real core usage (default is 1, i.e. full usage). The PUE (Power Usage Effectiveness) measures how much extra energy is needed to operate the data centre (cooling, lighting etc.). The PSF (Pragmatic Scaling Factor) is used to take into account multiple identical runs (e.g. for testing or optimisation).

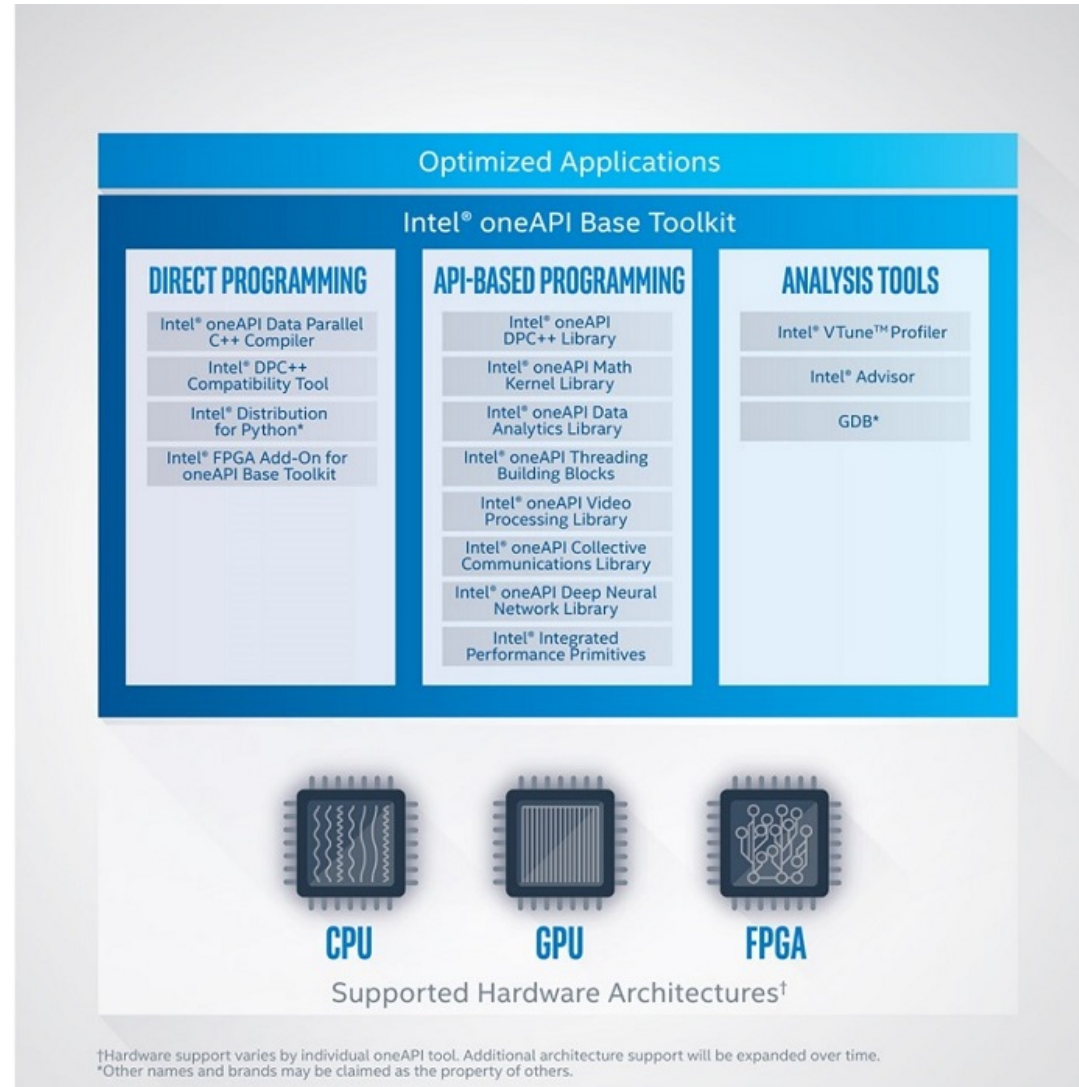
The Carbon Intensity depends on the location and the technologies used to produce electricity. But note that **the "energy needed" indicated at the top of this page is independent of the location.**

Applications need to be optimized for efficient execution on a given system hardware. Although some optimization techniques exist that aim particularly at improving energy efficiency (such as choosing algorithms that yield the same results with less energy), any optimization that reduces the runtime of an algorithm (time to solution) will ultimately result in lower energy consumption (energy to solution). Therefore, classical and well-known performance optimization techniques can be applied to achieve this goal.

Reducing runtime of an algorithm = Effective usage of CPU or GPU or any other hardware.

Max FLOPS/Watts

Intel oneAPI



1. Survey collection by command line with advisor:

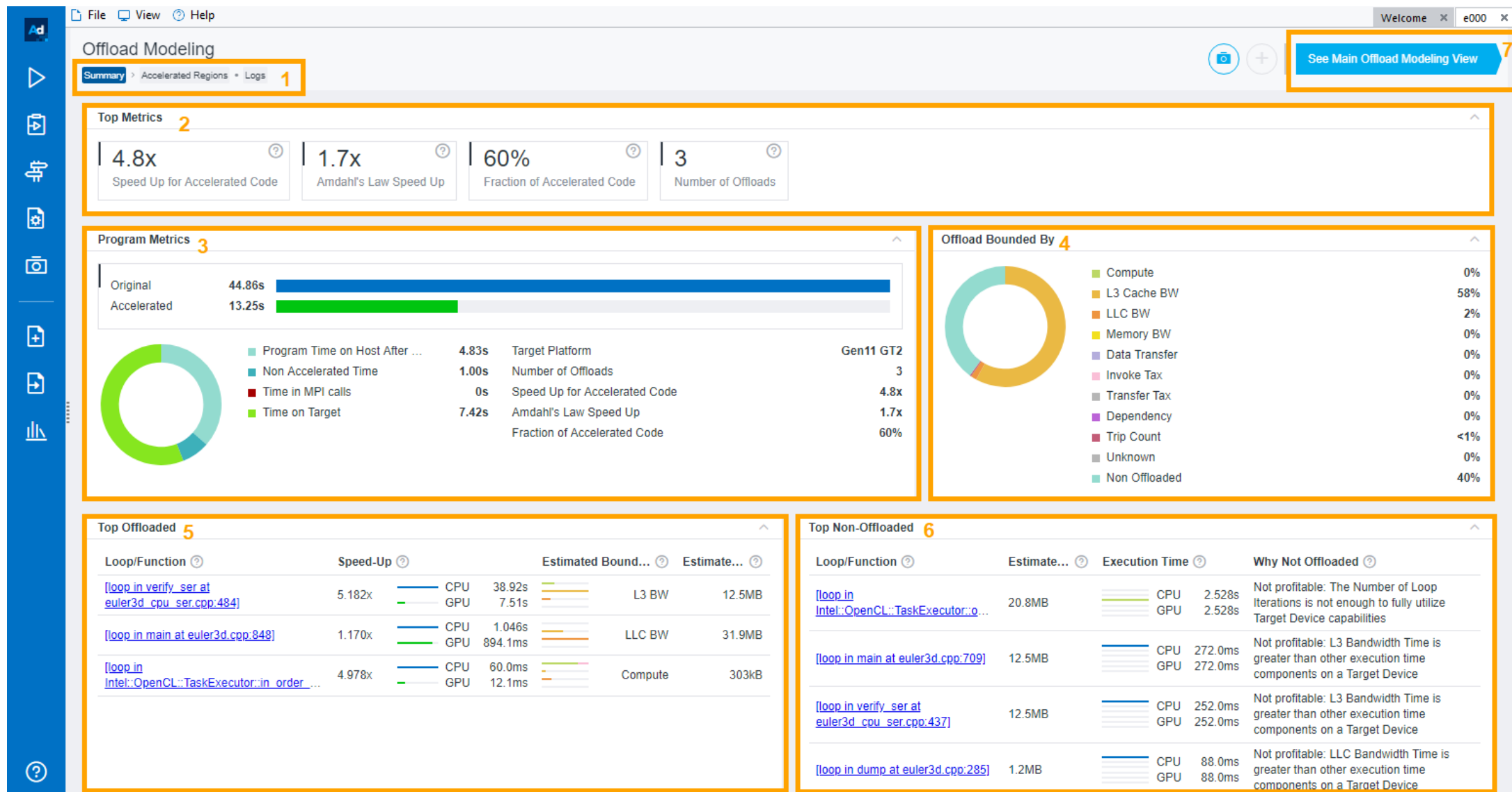
```
mpirun -n <number of nodes> advixe-cl -collect survey --trace-mpi --  
./<app_name>
```

2. Trip count collection by command line with advisor:

```
mpirun -n <number of nodes> advixe-cl -collect tripcounts -flops-and-  
masks --trace-mpi -- ./<app_name>
```

3. Extraction of the data in a report:

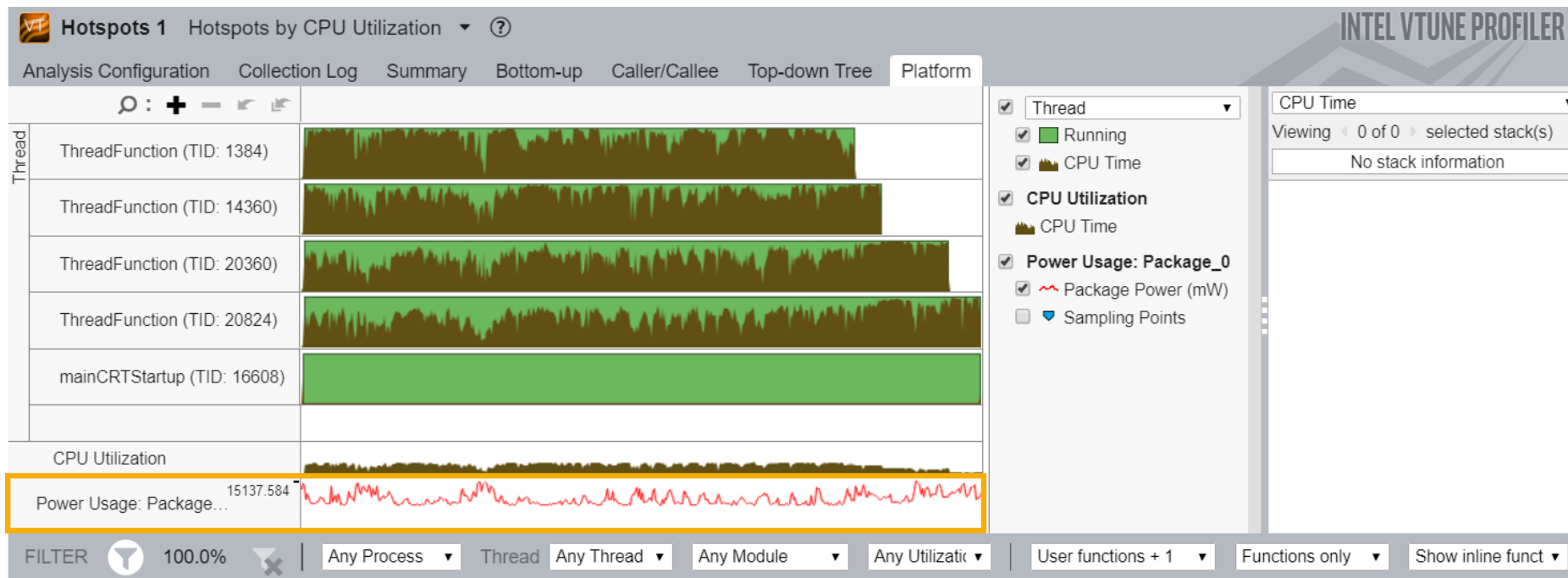
```
advixe-cl -report survey -show-all-columns --format=text -- report-  
output report.txt
```

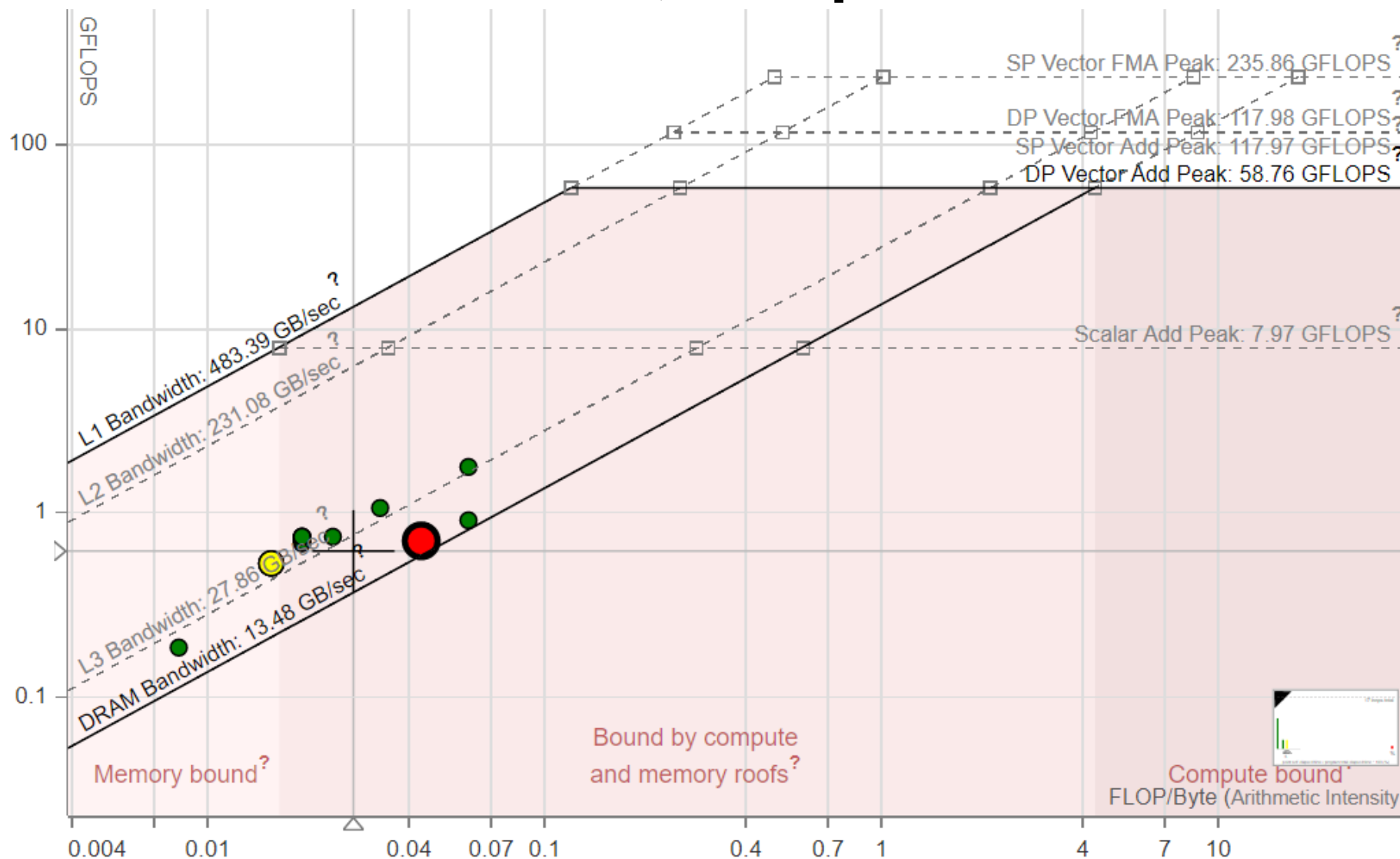
1. Run Intel SoC Watch for collecting data

```
socwatch -t 30 -f sys -r vtune -m -o run1
```

2. Read with Intel Vtune:

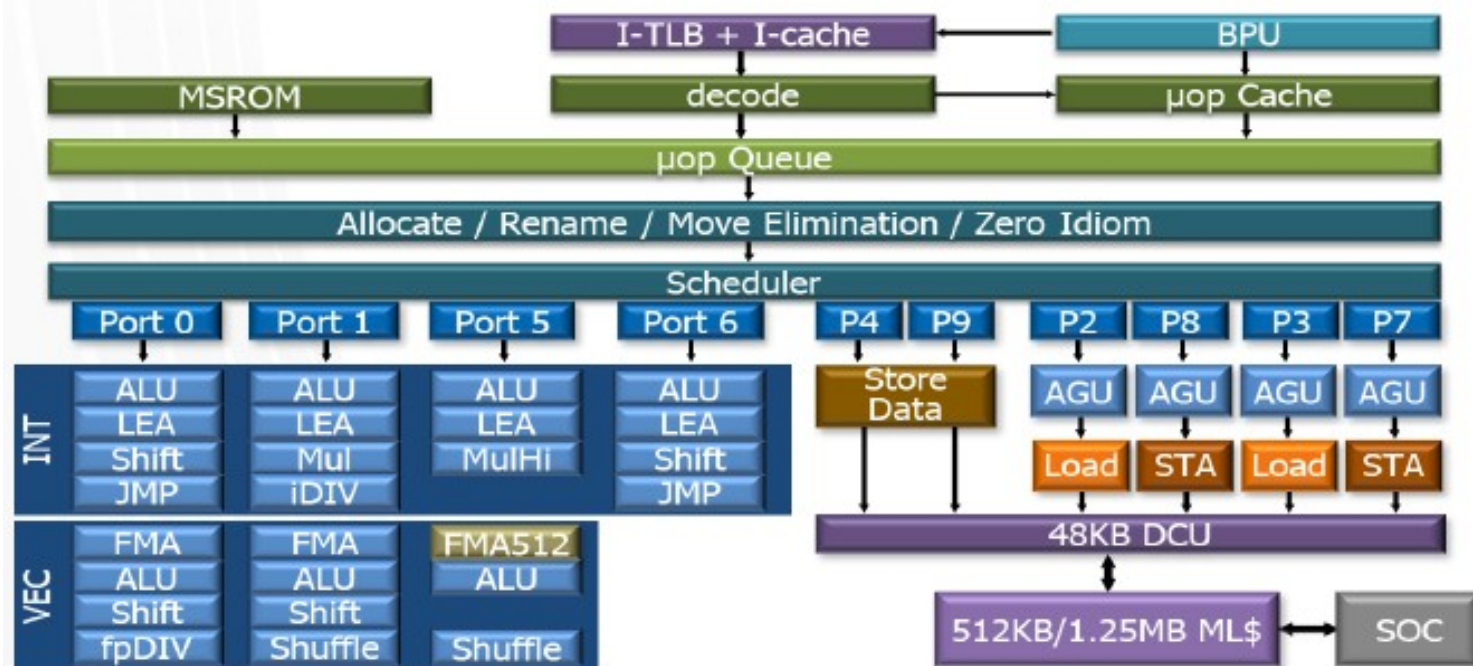


Poisson solver (27 point stencil approximation), 6248R, 1 core, no opt



6248R (1 core, no optimizations) - 0.6 GFLOPS/64 Watts =
0.009 GFLOPS/Watt

6248R (24 cores, no optimizations) – 10 GFLOPS/ 120 Watts =
0.08 GFLOPS/Watt



	Cascade Lake (per core)	Ice Lake (per core)
Out-of-order Window	224	384
In-flight Loads + Stores	72 + 56	128 + 72
Scheduler Entries	97	160
Register Files – Integer + FP	180 + 168	280 + 224
Allocation Queue	64/thread	70/thread 140/1 thread
L1D Cache (KB)	32	48
L1D BW (B/Cyc) – Load + Store	128 + 64	128 + 64
L2 Unified TLB	1.5K	2K
Mid-level Cache (MB)	1	1.25

- Improved Front-end: higher capacity and improved branch predictor
- Wider and deeper machine: wider allocation and execution resources + larger structures
- Enhancements in TLBs, single thread execution, prefetching
- Server enhancements – larger Mid-level Cache (L2) + second FMA

~18% Increase In IPC On Existing SPECcpu2017(est) Integer Rate Binaries

```

do while(Residual > EPS .and. NumIter < 1000)
  psn_q = 0.d0                                ! q = Az

  forall(i = 2:N+1, k = 2:N+1, l = 2:N+1)
    psn_q(i,k,l) = -38.d0/9.d0 * psn_z(i,k,l) +
      4.d0/9.d0 * ( psn_z(i+1,k,l) + psn_z(i-1,k,l) +
        psn_z(i,k+1,l) + psn_z(i,k-1,l) +
        psn_z(i,k,l+1) + psn_z(i,k,l-1) ) +
      1.d0/9.d0 * ( psn_z(i,k+1,l+1) + psn_z(i,k-1,l+1) +
        psn_z(i,k+1,l-1) + psn_z(i,k-1,l-1) +
        psn_z(i+1,k,l+1) + psn_z(i-1,k,l+1) +
        psn_z(i+1,k,l-1) + psn_z(i-1,k,l-1) +
        psn_z(i+1,k+1,l) + psn_z(i-1,k+1,l) +
        psn_z(i+1,k-1,l) + psn_z(i-1,k-1,l) ) +
      1.d0/36.d0 * ( psn_z(i+1,k+1,l+1) + psn_z(i-1,k+1,l+1) +
        psn_z(i+1,k+1,l-1) + psn_z(i-1,k+1,l-1) +
        psn_z(i+1,k-1,l+1) + psn_z(i-1,k-1,l+1) +
        psn_z(i+1,k-1,l-1) + psn_z(i-1,k-1,l-1) )

  end forall

  alphazn = sum(psn_q * psn_z)                ! alzn = (q,z)
  alpha = alphach/alphazn                    ! al = alch/alzn
  psn_x = psn_x + alpha * psn_z              ! x = x + al * z
  psn_r = psn_r - alpha * psn_q              ! r = r - al * q
  psn_q = psn_r                               ! q = r
  betach = sum(psn_q * psn_r)                ! btch = (q,r)
  betazn = alphach                           ! btzn = alch
  alphach = betach                           ! alch = betach
  beta = betach/betazn                       ! bt = btch/btzn
  psn_z = psn_q + beta * psn_z               ! z = q + bt * z
  NormR = dsqrt(sum(psn_r * psn_r))          ! |r|

  Residual = NormR/NormRight
  NumIter = NumIter + 1
  print *,NumIter,Residual

enddo

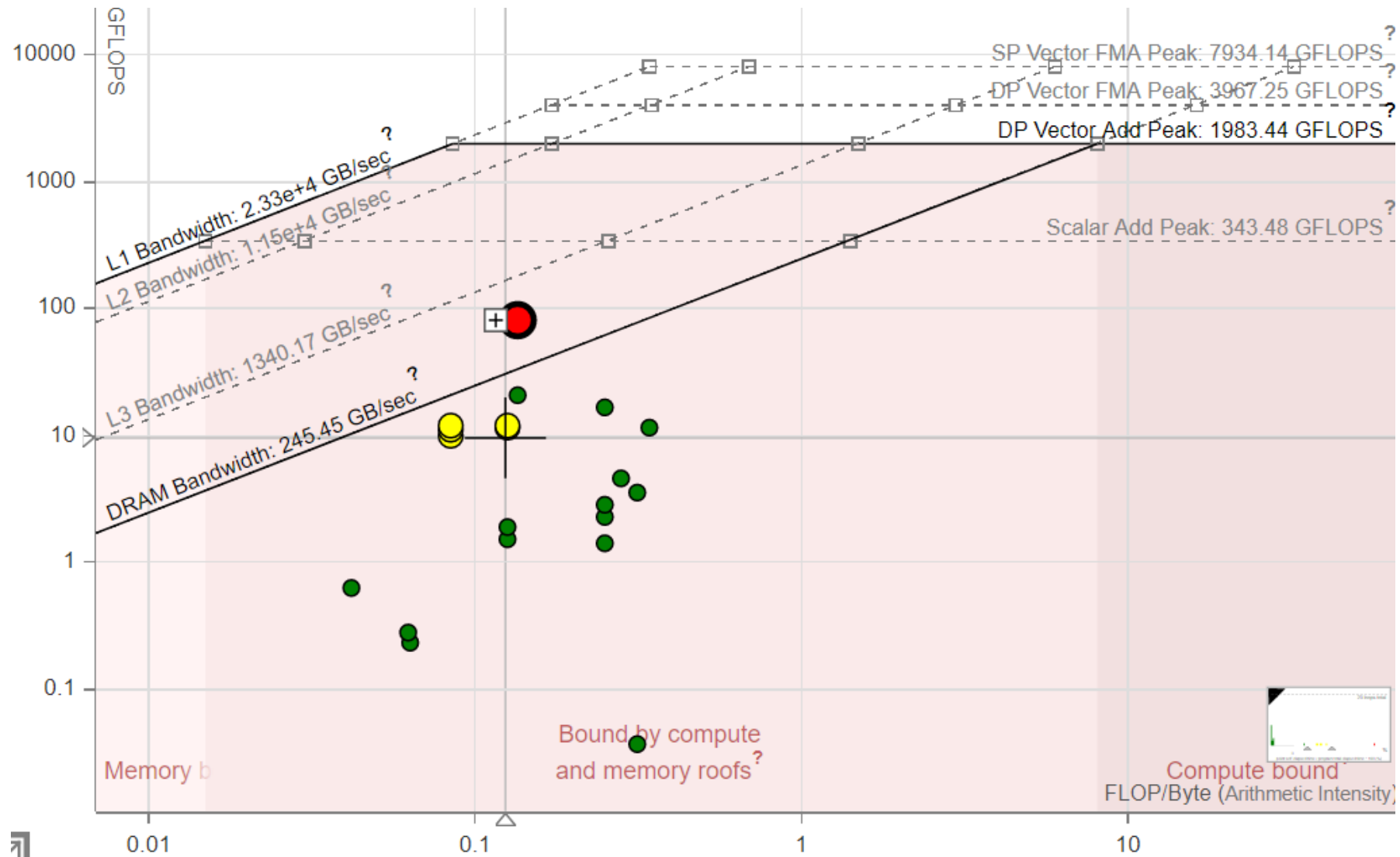
```

How to maximize the performance?

1. The loop count can't change once the loop starts by the code inside the loop.
2. Control flow shouldn't be changed by special statements.
3. There shouldn't be data dependencies with other indexes of the loop due to the memory conflict problem.
4. Conditionals sentences (if/else) can be used if they don't change the control flow, and are only used to conditionally load A or B values into a C variable. In fact, the best performance can be achieved only without conditional sentences inside loop.

Mode	Intel Xeon Gold 6144 (8 cores)	Intel Xeon Gold 6150 (18 cores)	Intel Xeon Gold 6154 (18 cores)
Base (without/with AVX-512) frequency	3.5GHz/2.2GHz	2.7GHz/1.9GHz	3GHz/2.1GHz
Normal turbo	4.1 GHz	3.4 GHz	3.7 GHz
AVX-512 turbo	2.8 GHz	2.5 GHz	2.7 GHz

Poisson solver (27 point stencil approximation), 6248R, 24 cores, AVX512, autoparallel



6248R (1 core, no optimizations) - 0.6 GFLOPS/64 Watts =
0.009 GFLOPS/Watt

6248R (24 cores, no optimizations, autoparallel) – 10
GFLOPS/ 120 Watts = 0.08 GFLOPS/Watt

6248R (24 cores, auto AVX512, autoparallel) – 95
GFLOPS/ 180 Watts = 0.52 GFLOPS/Watt



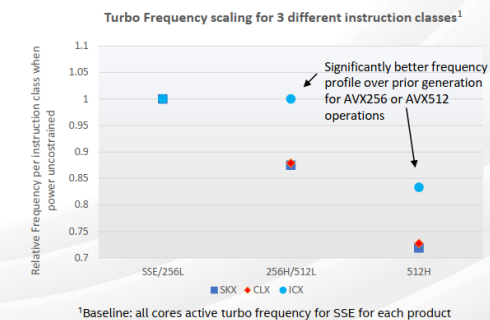
AVX Frequency Improvements

Goal: Minimize frequency impact on AVX 512 bit operations when not bounded by physical limits


- Not all AVX512 instructions consume high power
 - 512-bit loads, 512-bit stores, 256-bit FP, integer multiply are a few examples
 - Smarter mapping between instructions and specific power levels

Provides software writers greater latitude when using these instructions to optimize their code for performance

Power level	Class of instructions	Instruction types per class (not the full list)
0	SSE/256L	all 64b and 128bit
	256H	FP Mul, INT Mul, VNNI, FMA 256b
1	512L	VPCLMUL, VAES, VBMI, Ld, St 512b
2	512H	FP Mul, INT Mul, VNNI, FMA, VPMADD52 512b



8368Q (38 cores, AVX512, autoparallel) – 130 GFLOPS/ 210 Watts = 0.61 GFLOPS/Watt



2	Summit - IBM Power System AC922, IBM POWER9 22C 3.07GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband, IBM DOE/SC/Oak Ridge National Laboratory United States	2,414,592	148,600.0	200,794.9	10,096
3	Sierra - IBM Power System AC922, IBM POWER9 22C 3.1GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband, IBM / NVIDIA / Mellanox DOE/NNSA/LLNL United States	1,572,480	94,640.0	125,712.0	7,438

6 GPUs per node!!!

Thank you for your attention

Vectorization

```
void hist(const float * samples, int * const hist, const float group_width)
{ float group_width_rec = 1.0f / group_width;
#pragma vector aligned
for (int sample_index = 0; sample_index < 1024; sample_index++)
{ const int bin = (int) (samples[sample_index] * group_width_rec);
hist[bin]++; }
}
```

```
double A[vec_width], B[vec_width];
//this loop will be automatically vectorized
for(int i = 0; i < vec_width; i++)
A[i]+=B[i];
```

```
double A[vec_width], B[vec_width];
__m512d A_vec = _mm512_load_pd(A);
__m512d B_vec = _mm512_load_pd(B);
A_vec = _mm512_add_pd(A_vec,B_vec);
_mm512_store_pd(A,A_vec);
```