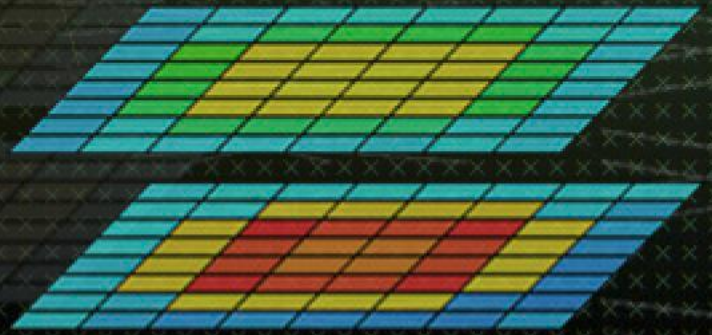


MERIC

Energy consumption and efficiency assessment with MERIC and RADAR

Tomáš Panoc

IT4Innovations, VSB – Technical University of Ostrava, Ostrava, Czech Republic
Contact e-mail: meric@it4i.cz

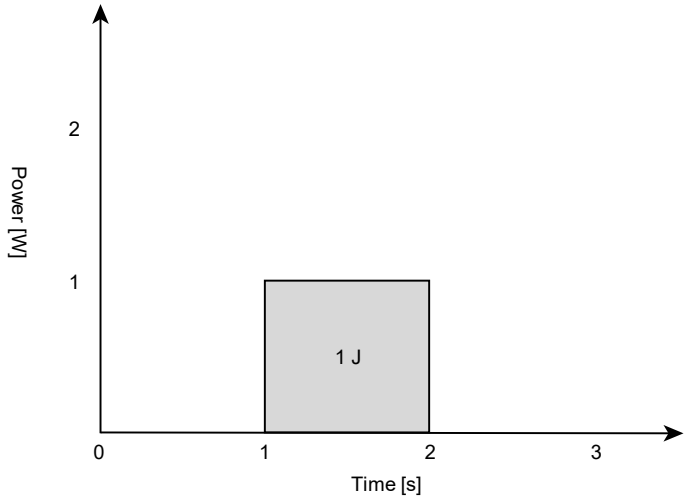
Based on original lectures and slides by Ondřej Vysocký

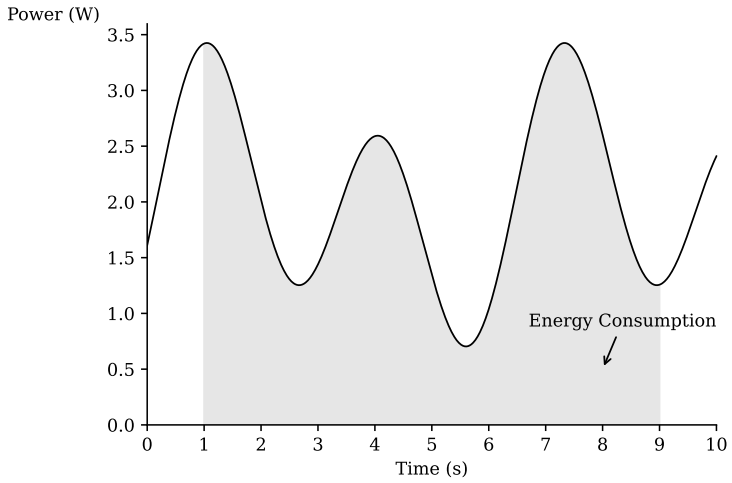
Basic terms

- Energy consumption
- Energy efficiency

Energy consumption

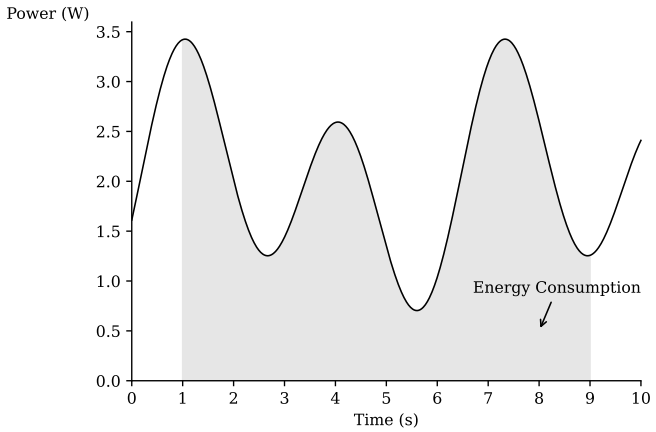
- Energy [J] = Power [W] × Time [s]
- 1 W × 1 s = 1 J
- 1 W × 1 h = 1 Wh = 3600 J

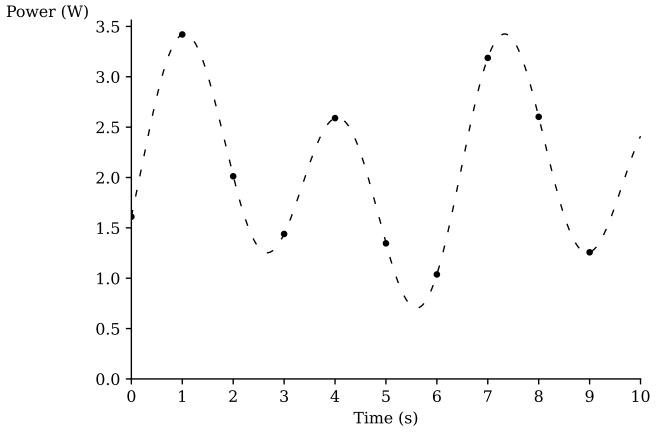




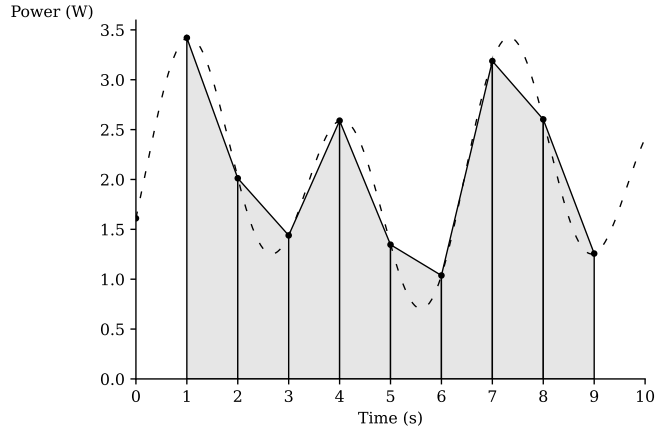
Figures: Luís Cruz, TU Delft

Energy consumption: Measurement





Power sample collection



Total energy consumption calculated from the samples

Approximation with samples and their frequency

$$Energy(t) = \int_0^t Power(x)\,dx \approx \frac{\sum_{i=0}^{n} PowerSample_i}{SamplingFrequency}$$

Figures: Luís Cruz, TU Delft

Energy efficiency

- Performance per watt

- Energy efficiency [FLOPs/Watt] = Performance [FLOPs] / Power [Watt]
  - FLOPs = FLoating-point OPerations per second
  - Energy efficiency [FLOPs/Watt] = Operations [FLOP] / Energy [J]
    - 1 J = 1 W × 1 s

- Performance can be measured in other units depending on the context
  - Lattice updates per second (LUPs) for Lattice-Boltzmann simulations
    - Energy efficiency [LUPs/Watt]

Energy efficiency: Performance

- FLoating-point OPerations per second (FLOPs)
- Measurement of floating-point operations via dedicated counters and APIs
  - perf_events
- Evaluation of FLOPs (Intel CPUs):

event name	**FLOPs/inst**	**#inst.**	**FLOPs**
SCALAR_DOUBLE	1	7240374183451	9164881641636
SCALAR_SINGLE	1	154346452896	154346452896
128B_PACKED_DOUBLE	2	557726129284	1115452258568
128B_PACKED_SINGLE	4	254594513	1018378052
256B_PACKED_DOUBLE	4	2725451737	10901806948
256B_PACKED_SINGLE	8	0	0
512B_PACKED_DOUBLE	8	0	0
512B_PACKED_SINGLE	16	0	0
Runtime [s]	105.38	**Sum [FLOPs]**	8522093079915
		GFLOPs/s	80.87

Note: FLOPs stand only for FLoating-point OPerationS in the table

Energy efficiency: The Green500 list

Rank	TOP500 Rank	System	Cores	Rmax (PFlop/s)	Power (kW)	Energy Efficiency (GFlops/watts)
1	259	**JEDI** - BullSequana XH3000, Grace Hopper Superchip 72C 3GHz, NVIDIA GH200 Superchip, Quad-Rail NVIDIA InfiniBand NDR200, ParTec/EVIDEN EuroHPC/FZJ Germany	19,584	4.50	67	72.733
2	148	**ROMEO-2025** - BullSequana XH3000, Grace Hopper Superchip 72C 3GHz, NVIDIA GH200 Superchip, Quad-Rail NVIDIA InfiniBand NDR200, Red Hat Enterprise Linux, Bull ROMEO HPC Center - Champagne-Ardenne France	47,328	9.86	160	70.912

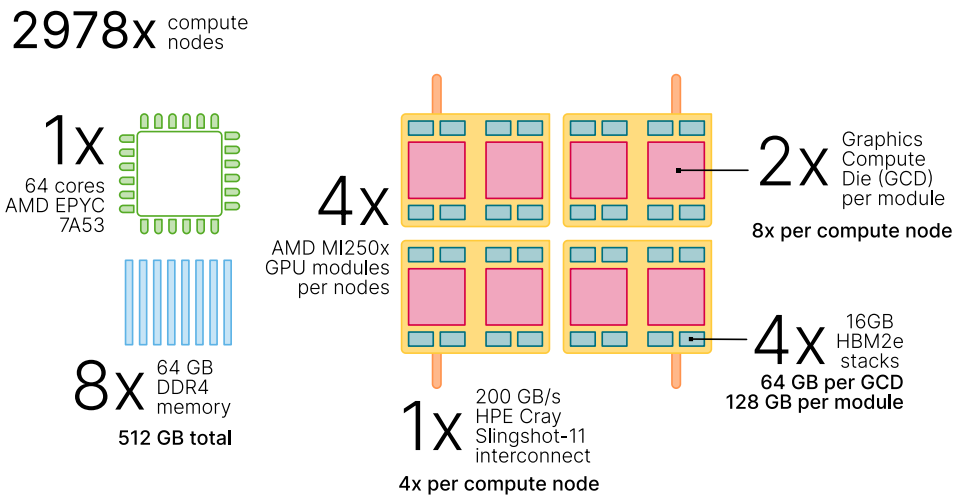
$$Energy\ efficiency = \frac{LINPACK\ performance\ R_{max}}{Average\ power\ consumption\ \bar{P}(R_{max})}$$

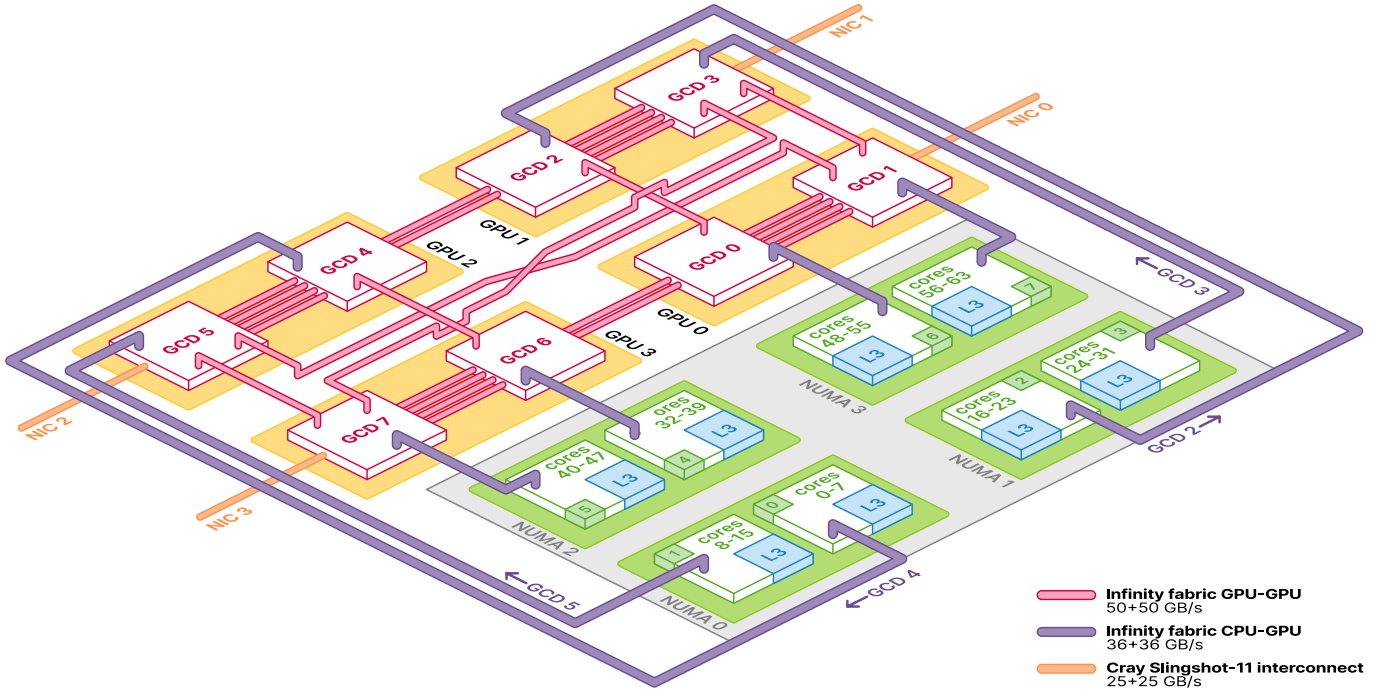
Power measurement tutorial for Green500: https://www.top500.org/files/green500/tutorial.pdf

Power monitoring

Components of a compute node

- CPUs, GPUs, DRAMs, and other on-node components (e.g., NIC)
- Example of an accelerated node of LUMI-G:



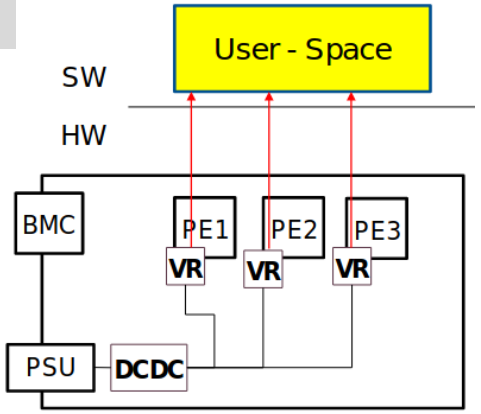


Figures: LUMI supercomputer, LUMI consortium

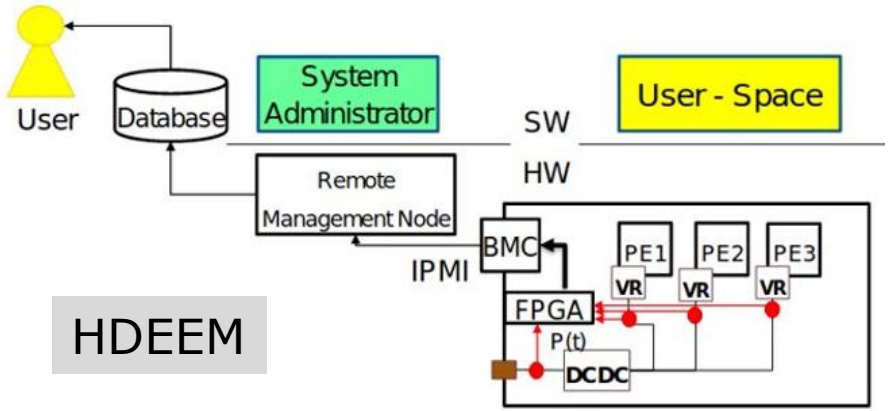
In-band and out-of-band power monitoring

- In-band
  - Part of an internal infrastructure of the given component/device
  - Hardware registers / performance counters
  - Vendor specific
  - Examples: RAPL, NVML, AMD (ROCm) SMI

RAPL



- Out-of-band
  - Dedicated and independent device that monitors other components
  - No overhead
  - Fine-grain power measurement (usually)
  - Custom sensors
  - Examples: HDEEM, HPE iLO (IPMI/Redfish)



HDEEM

Figures: Antonio Libri, UNIBO

Power monitoring systems

Power monitoring system	In-/Out-of-band	Note
Intel RAPL	In-band	Intel CPUs
AMD RAPL	In-band	AMD CPUs
NVML	In-band	NVIDIA GPUs
AMD (ROCm) SMI	In-band	AMD GPUs
HWMON	In-band	Universal, e.g., NVIDIA Grace and GraceHopper
A64FX	In-band	Fujitsu ARM CPUs
HDEEM	Out-of-band	Bull Sequana X1000, B700
IPMI / Redfish	Out-of-band	HPE iLO

Some of the in-band systems above monitor several power domains (e.g., cores, package) including memory

Power monitoring of a compute node and its components

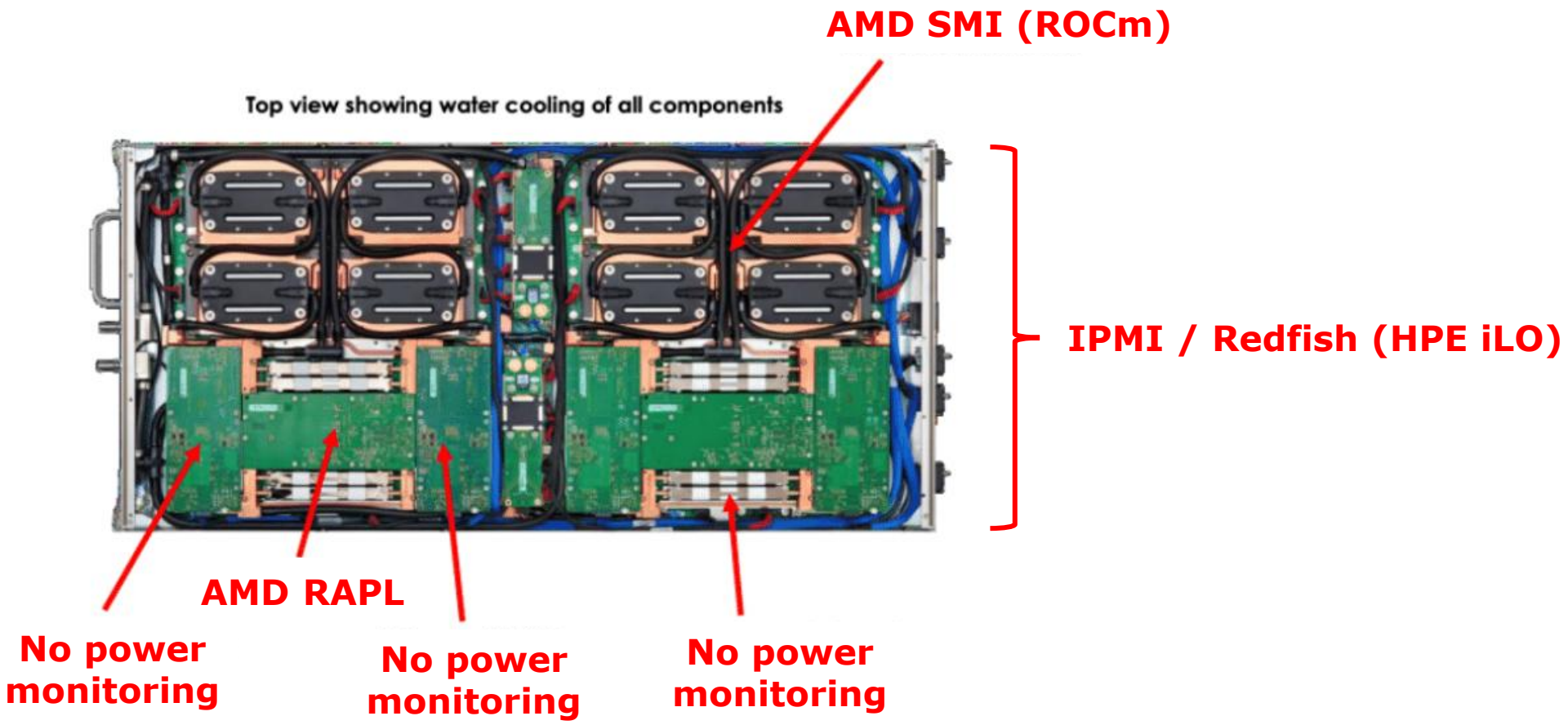


Figure: HPC Cray, OLCF Frontier

Node power baseline

- Power monitoring is not available for every component of a compute node
- Entire node can be monitored (IPMI / Redfish)
  - Low frequency power monitoring (around 0.1 to 1 Hz) in comparison with high frequency systems (e.g., Intel RAPL 1 kHz)
  - Unreliable for short and medium-length runs

- Node power baseline
  - Power of the remaining on-node components with no dedicated power monitoring system
  - System-specific value
  - Estimated from the node power monitoring and available component monitoring (RAPL, NVML, etc.) by loading the node by a uniform workload

- Example of a calculation of the total energy consumption on LUMI-G:
  - Node energy = AMD RAPL (CPUs) + AMD ROCm (GPUs) + (power baseline × time)

Node power baseline: Example from Karolina ACN (GPU load)

Active GPUs	0/8	1/8	2/8	3/8	4/8	5/8	6/8	7/8	8/8
GPU7 [W]	55	54	54	54	54	56	398	397	398
GPU6 [W]	52	52	51	51	51	397	398	396	397
GPU5 [W]	51	51	51	50	50	51	51	55	398
GPU4 [W]	52	52	51	51	51	52	52	398	399
GPU3 [W]	53	60	398	398	398	398	398	398	398
GPU2 [W]	52	398	397	398	398	397	398	397	398
GPU1 [W]	55	54	54	57	396	398	398	398	398
GPU0 [W]	52	52	51	398	398	394	394	393	398
CPU0 [W]	92	94	92	94	96	97	96	99	99
CPU1 [W]	95	95	99	99	98	98	102	100	102
SUM CPU [W]	187	189	191	193	194	195	198	199	201
SUM GPU [W]	422	773	1107	1457	1796	2143	2487	2832	3184
OOB AVG [W]	1129	1490	1842	2210	2570	2930	3290	3650	4010
CPU + GPU [W]	609	962	1298	1650	1990	2338	2685	3031	3385
BASELINE [W]	520	528	544	560	580	592	605	619	625

Idle

Power management

Power management

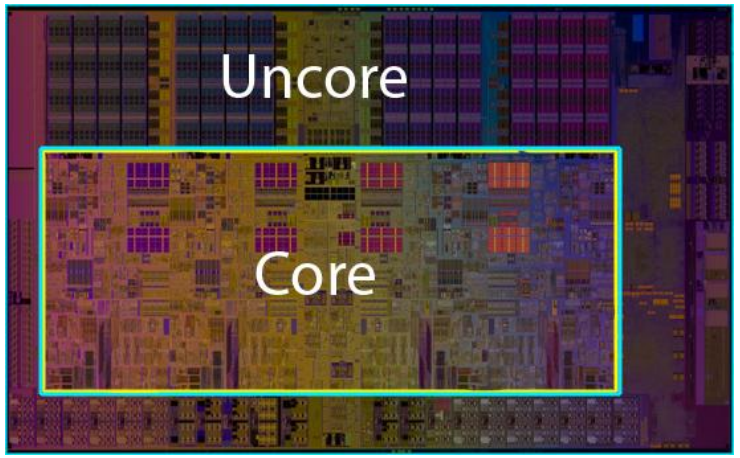
- Control of the power (energy) consumption

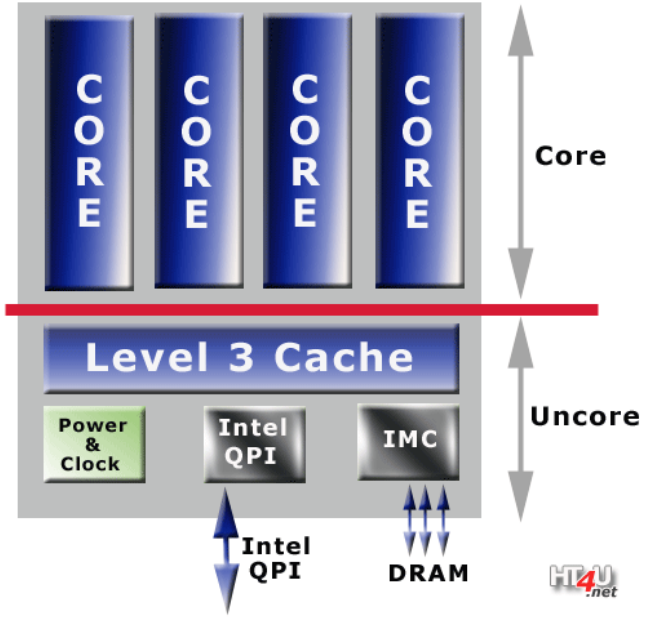
- CPUs
  - Core frequency [Hz]
  - Uncore frequency [Hz] (Intel and AMD)
  - Power limit [W] (power capping)

- GPUs
  - Streaming-multiprocessor frequency [Hz]
  - Memory frequency [Hz]
  - Power limit [W] (power capping)

Intel CPUs: Uncore frequency

- Frequency of subsystems in the CPU package shared by multiple CPU cores
  - Last-level cache, on-chip ring interconnect, integrated memory controllers, etc.
  - The uncore subsystem occupies approximately 30% of the chip area
- Intel Model-Specific Registers (MSRs)
  - MSR 0x620: MSR_UNCORE_RATIO_LIMIT
- AMD – Data-fabric frequency





Uncore frequency: Intel Granite Rapids

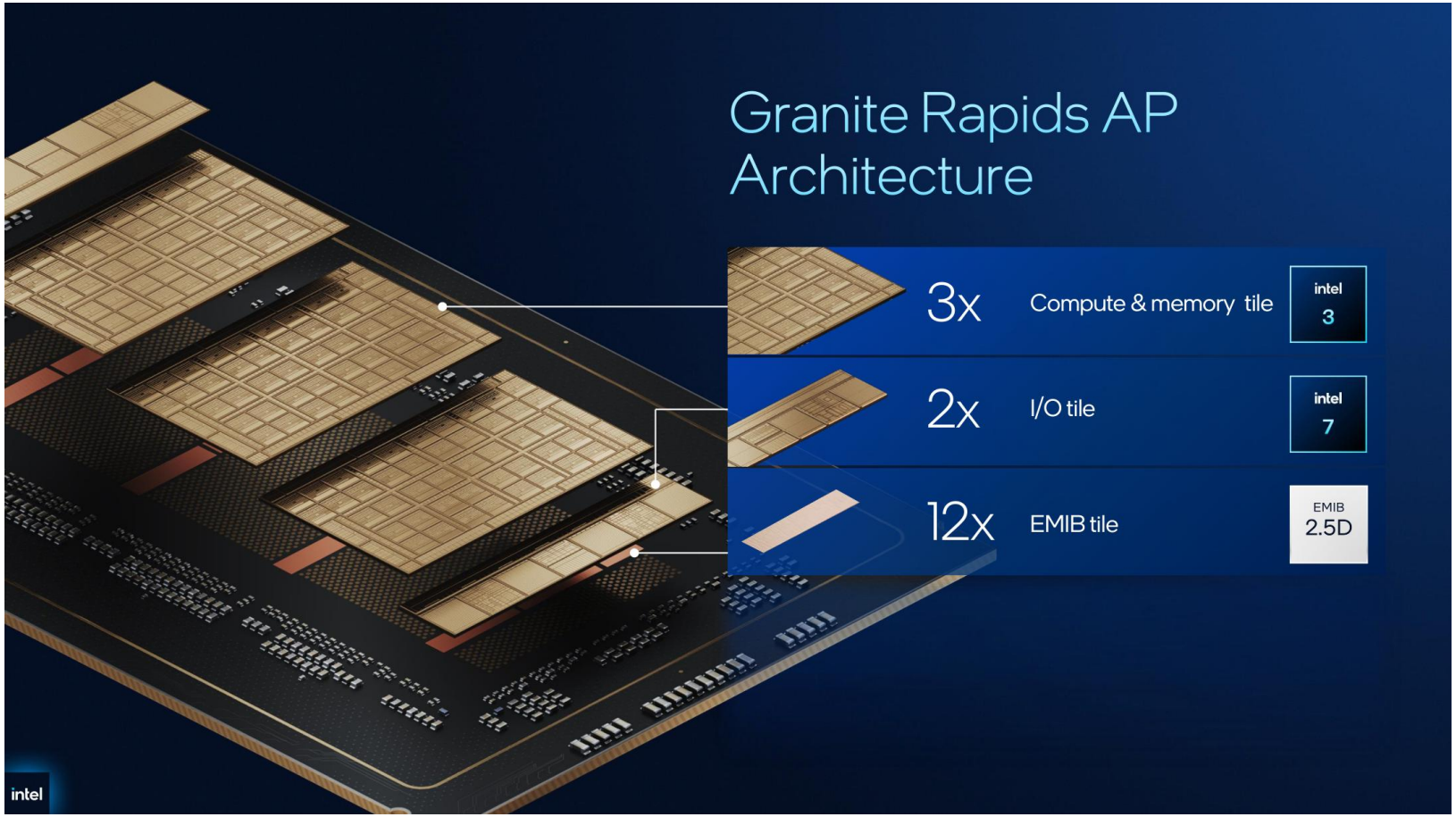
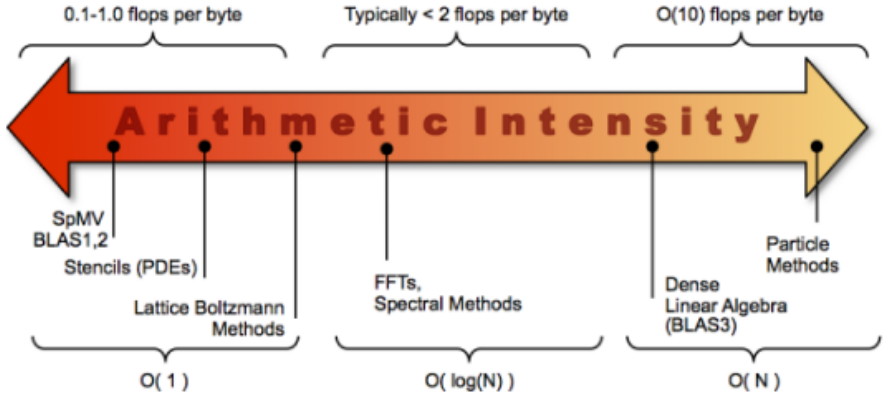


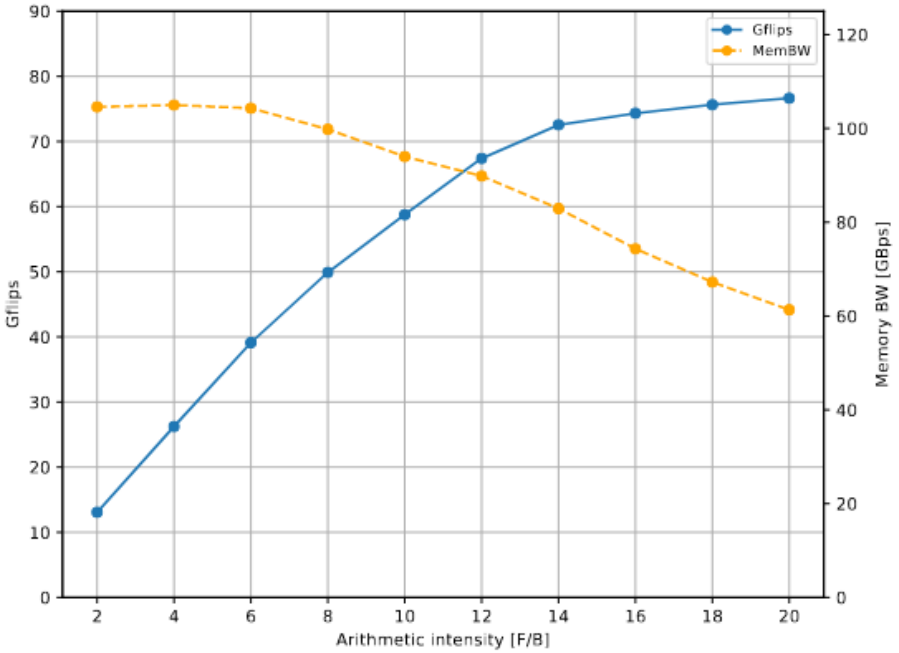
Figure: Intel® Xeon® 6+ Processors, Intel Corporation

Workload boundedness

- Memory-bound, compute-bound, I/O-bound, communication-bound, etc.



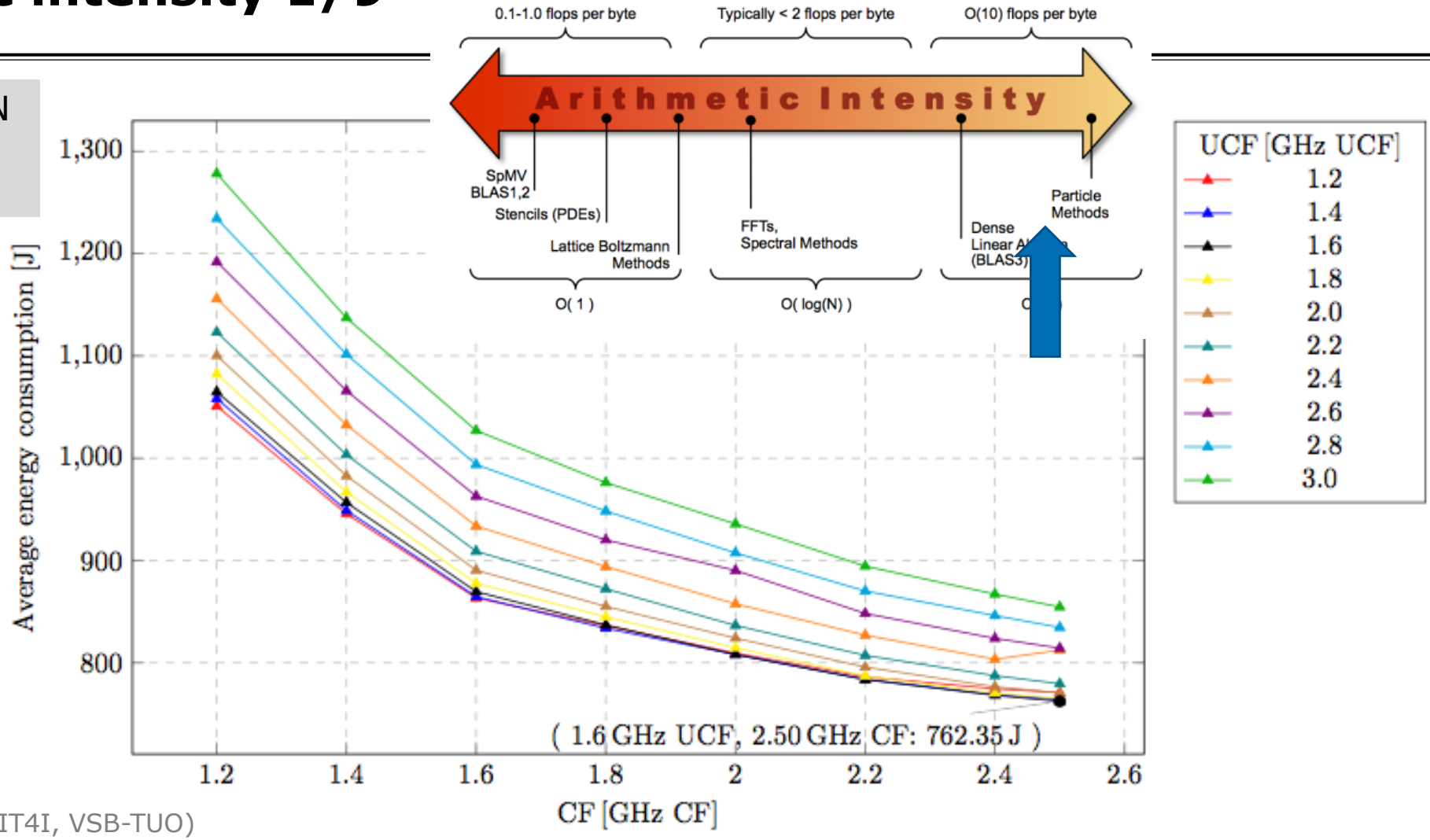
(a) Arrow presenting a range of applications of various arithmetic intensities [54].



(b) Roofline model of the Intel Xeon Gold 6240 processor when executing a workload of AVX-512 instructions.

Arithmetic intensity 1/9

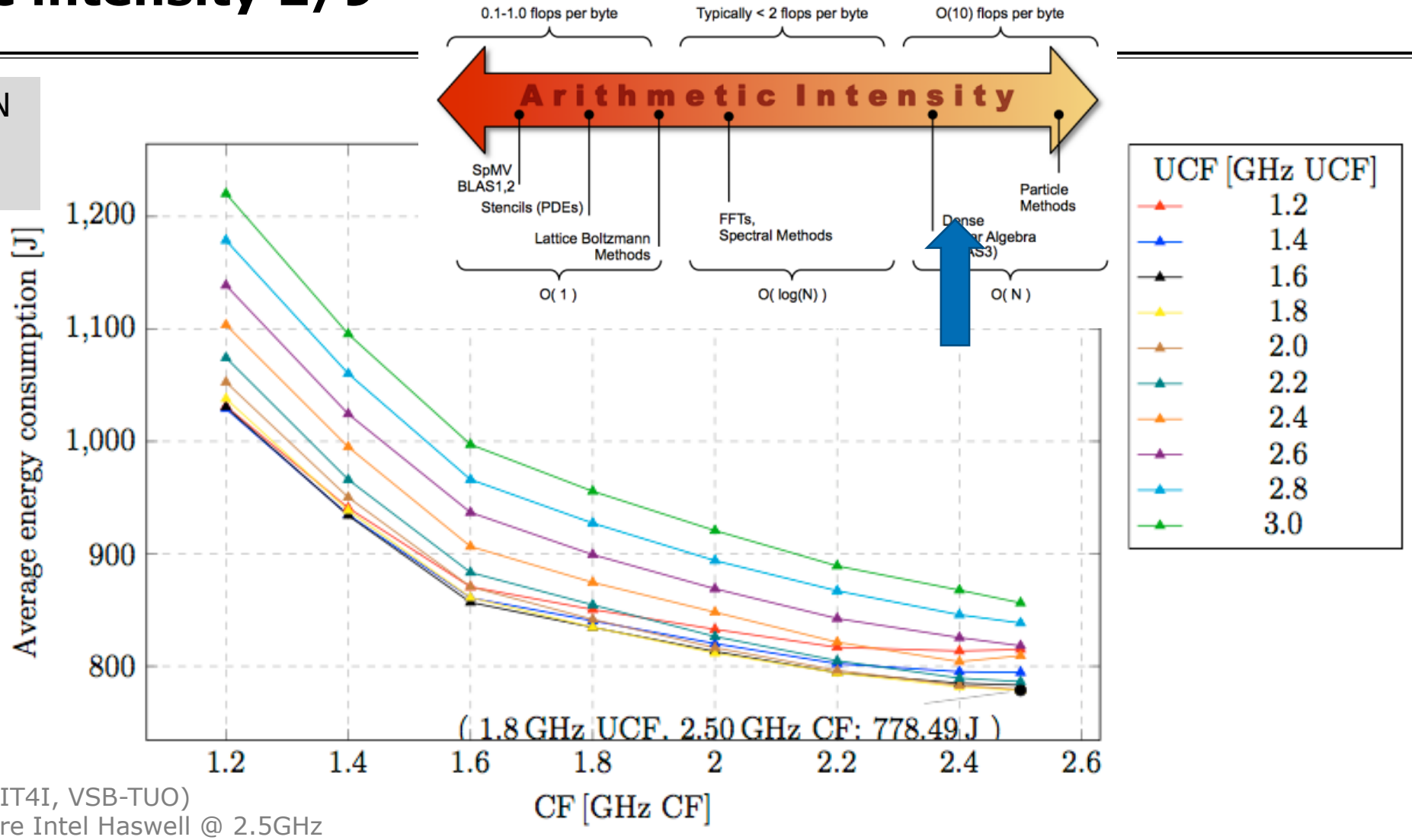
GEMV and TAN operations (ratio 1:9)



Machine: Salomon (IT4I, VSB-TUO)
Processor: 2x 12-core Intel Haswell @ 2.5GHz

Arithmetic intensity 2/9

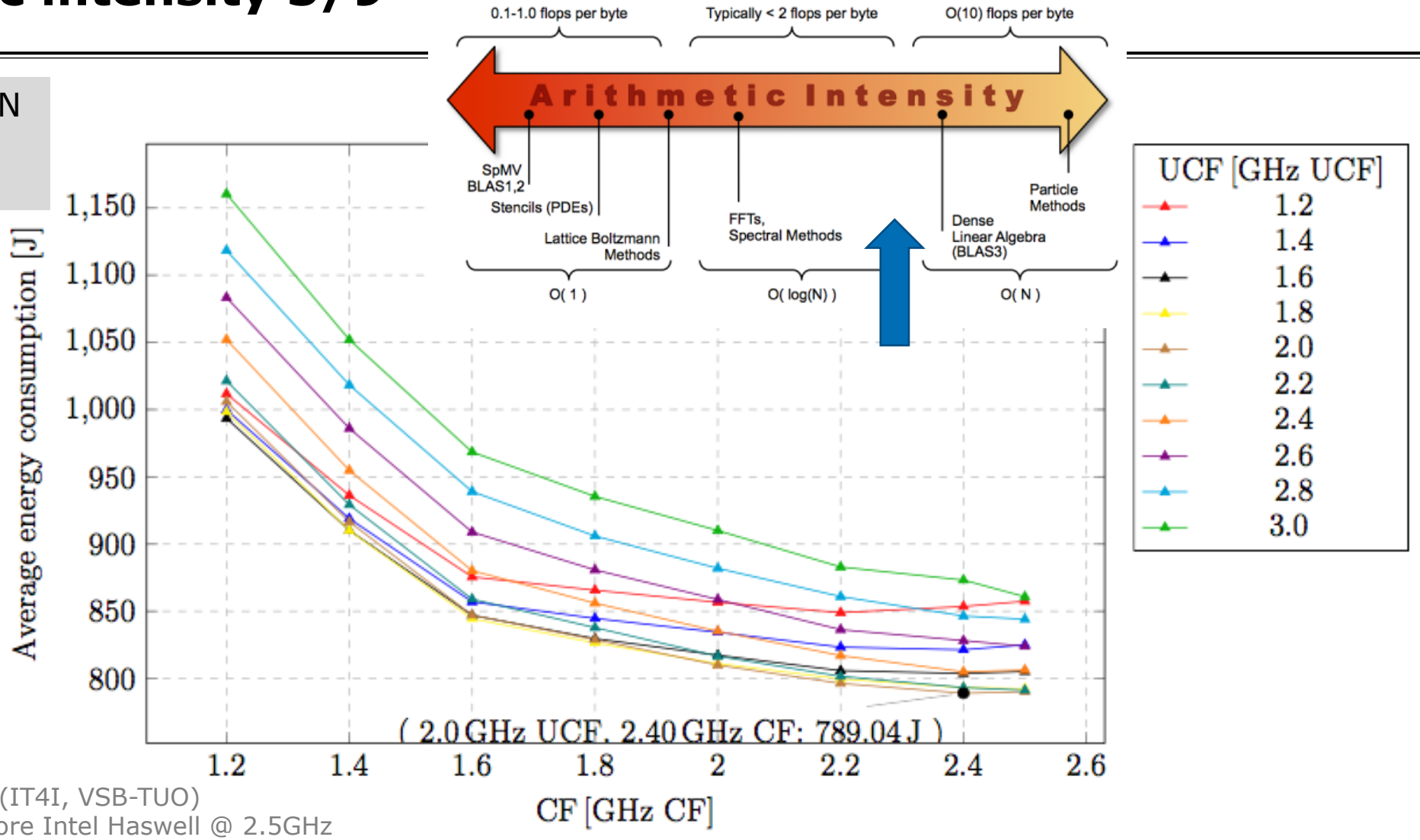
GEMV and TAN operations (ratio 2:8)



Machine: Salomon (IT4I, VSB-TUO)
Processor: 2x 12-core Intel Haswell @ 2.5GHz

Arithmetic intensity 3/9

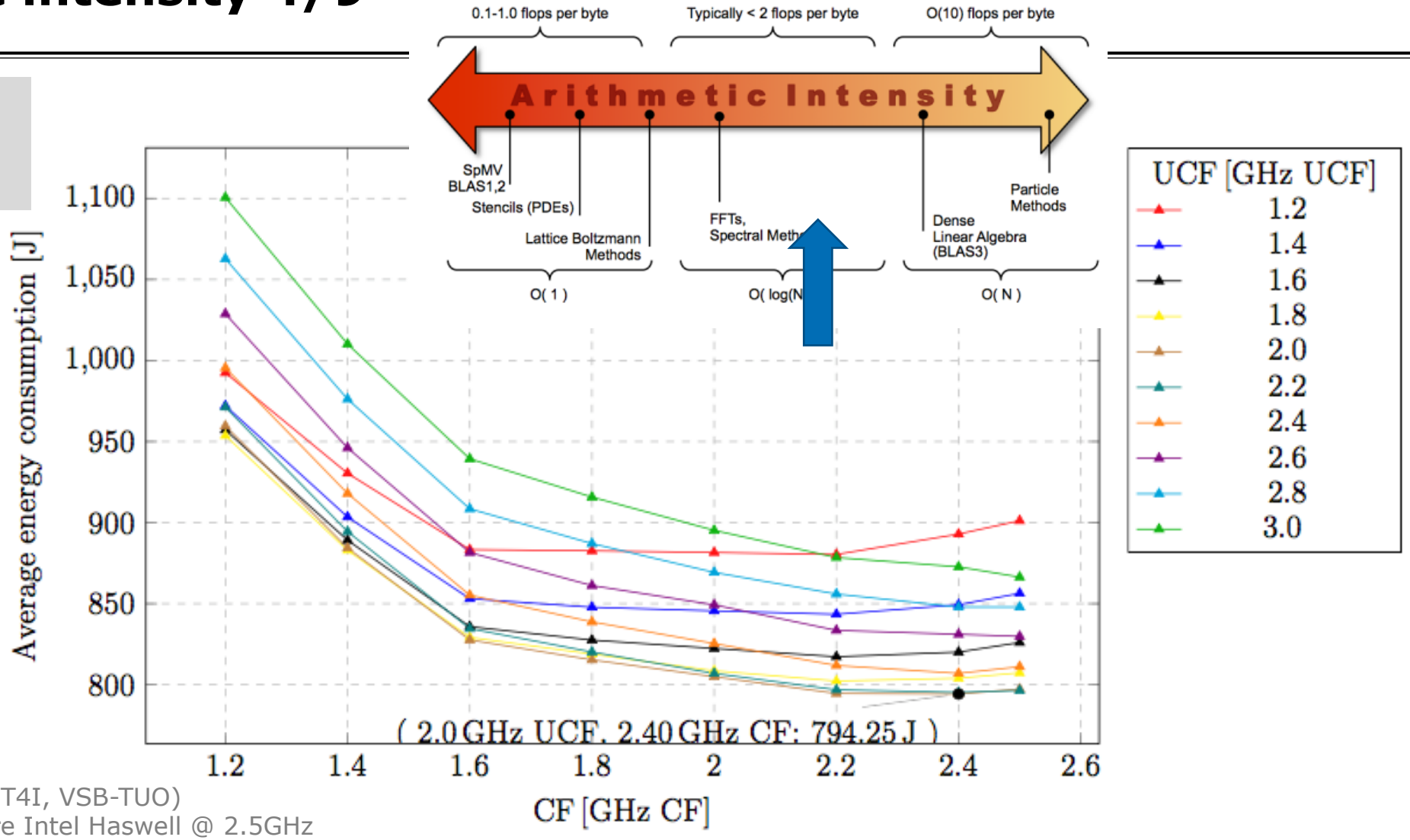
GEMV and TAN operations (ratio 3:7)



Machine: Salomon (IT4I, VSB-TUO)
Processor: 2x 12-core Intel Haswell @ 2.5GHz

Arithmetic intensity 4/9

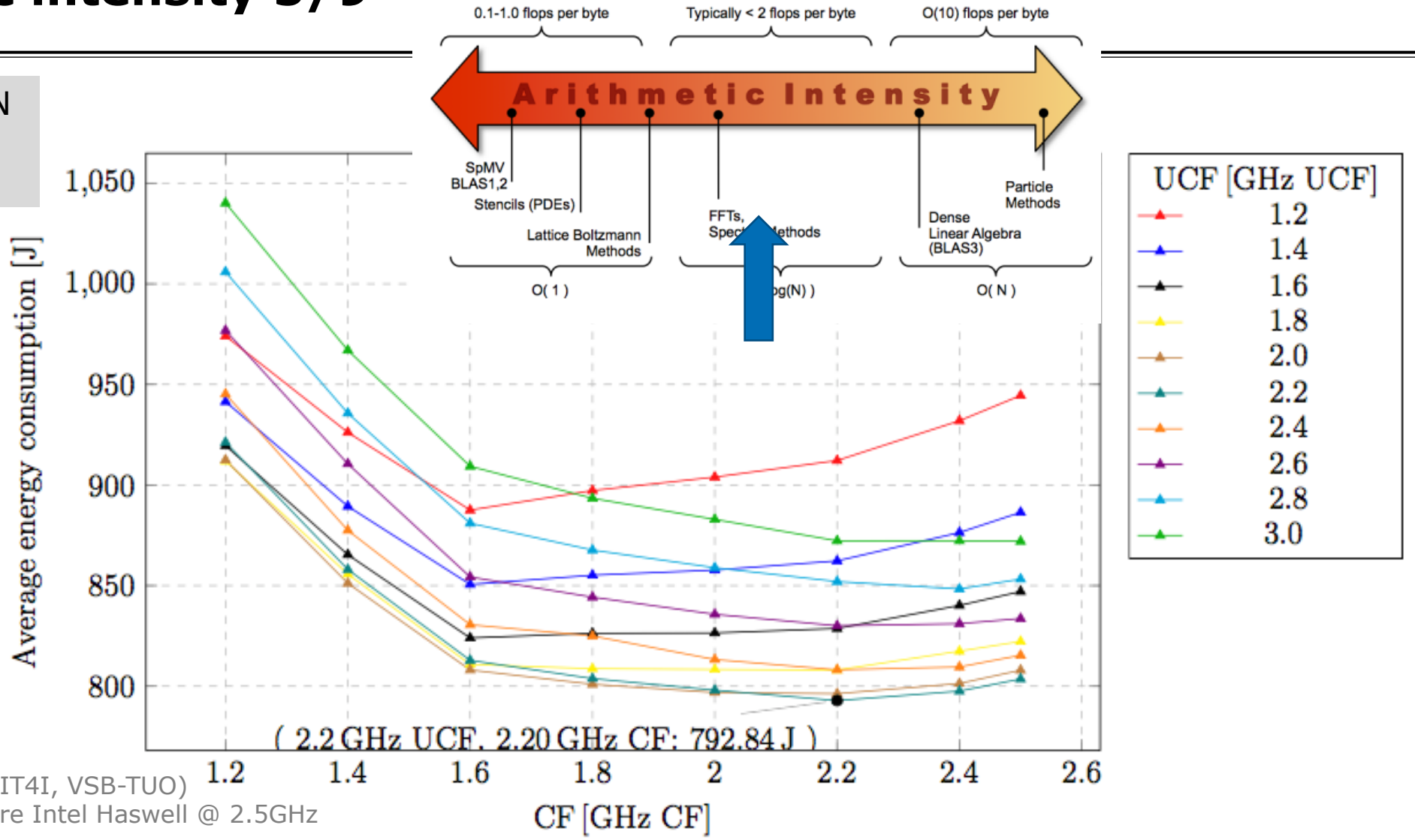
GEMV and TAN operations (ratio 4:6)



Machine: Salomon (IT4I, VSB-TUO)
Processor: 2x 12-core Intel Haswell @ 2.5GHz

Arithmetic intensity 5/9

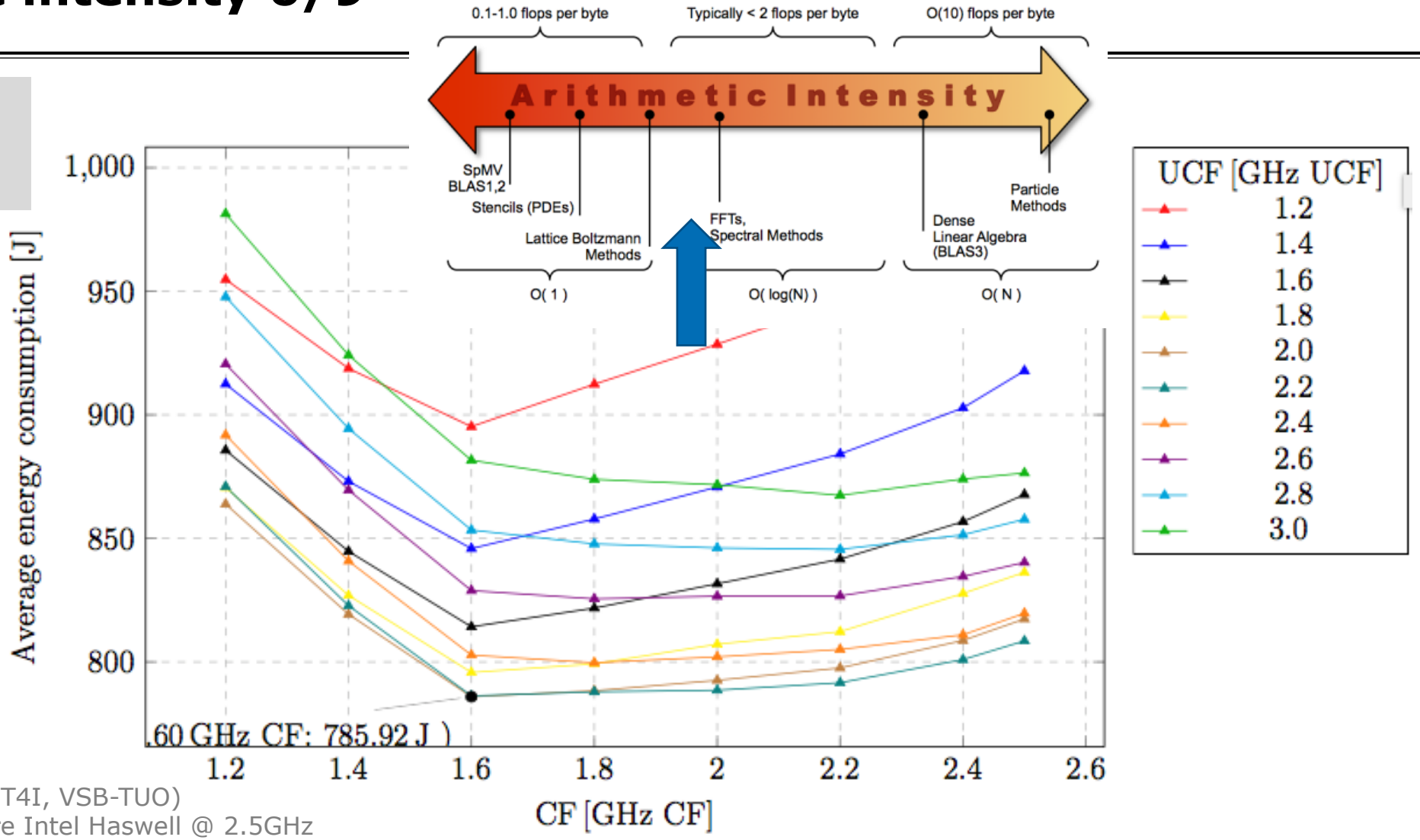
GEMV and TAN operations (ratio 5:5)



Machine: Salomon (IT4I, VSB-TUO)
Processor: 2x 12-core Intel Haswell @ 2.5GHz

Arithmetic intensity 6/9

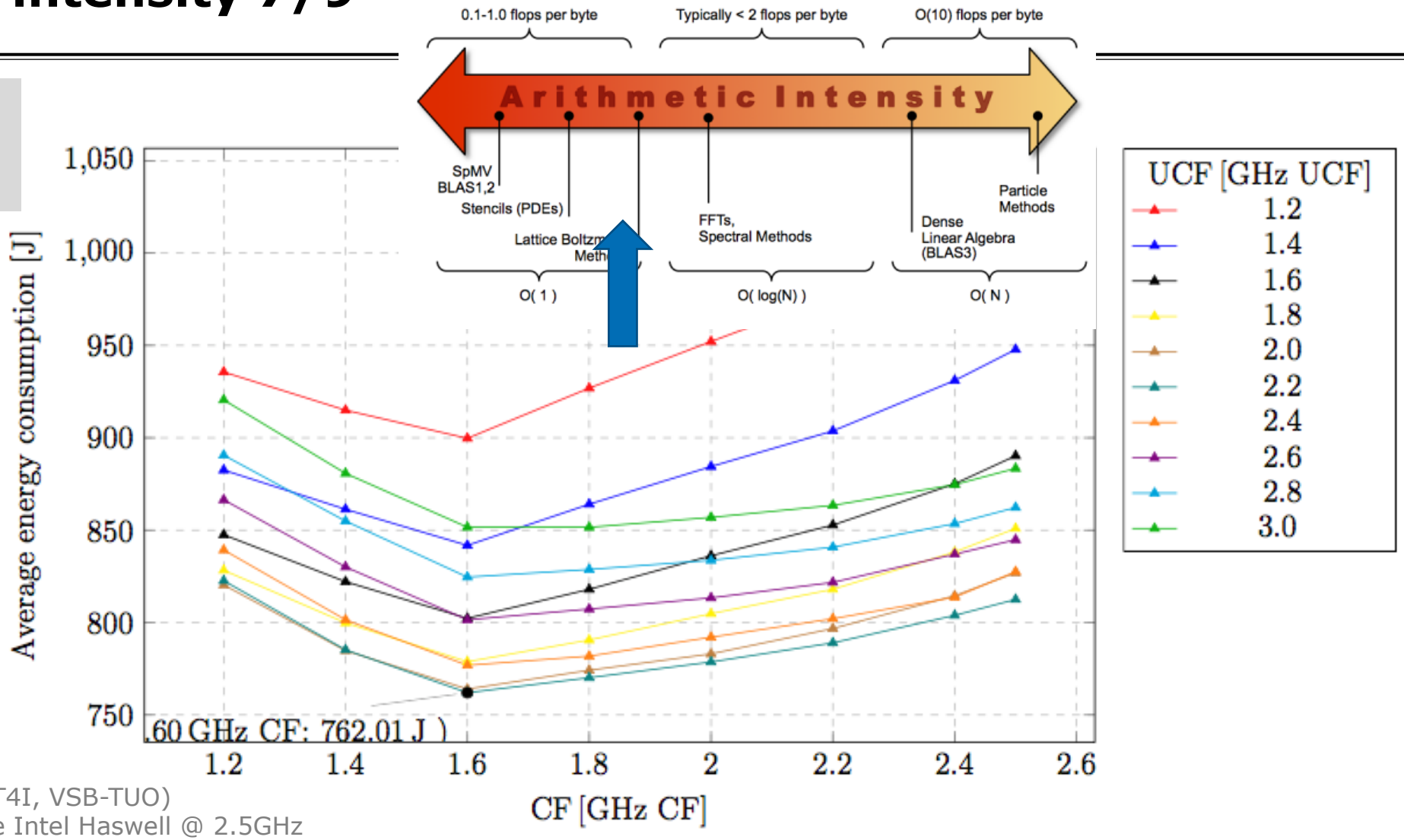
GEMV and TAN operations (ratio 6:4)



Machine: Salomon (IT4I, VSB-TUO)
Processor: 2x 12-core Intel Haswell @ 2.5GHz

Arithmetic intensity 7/9

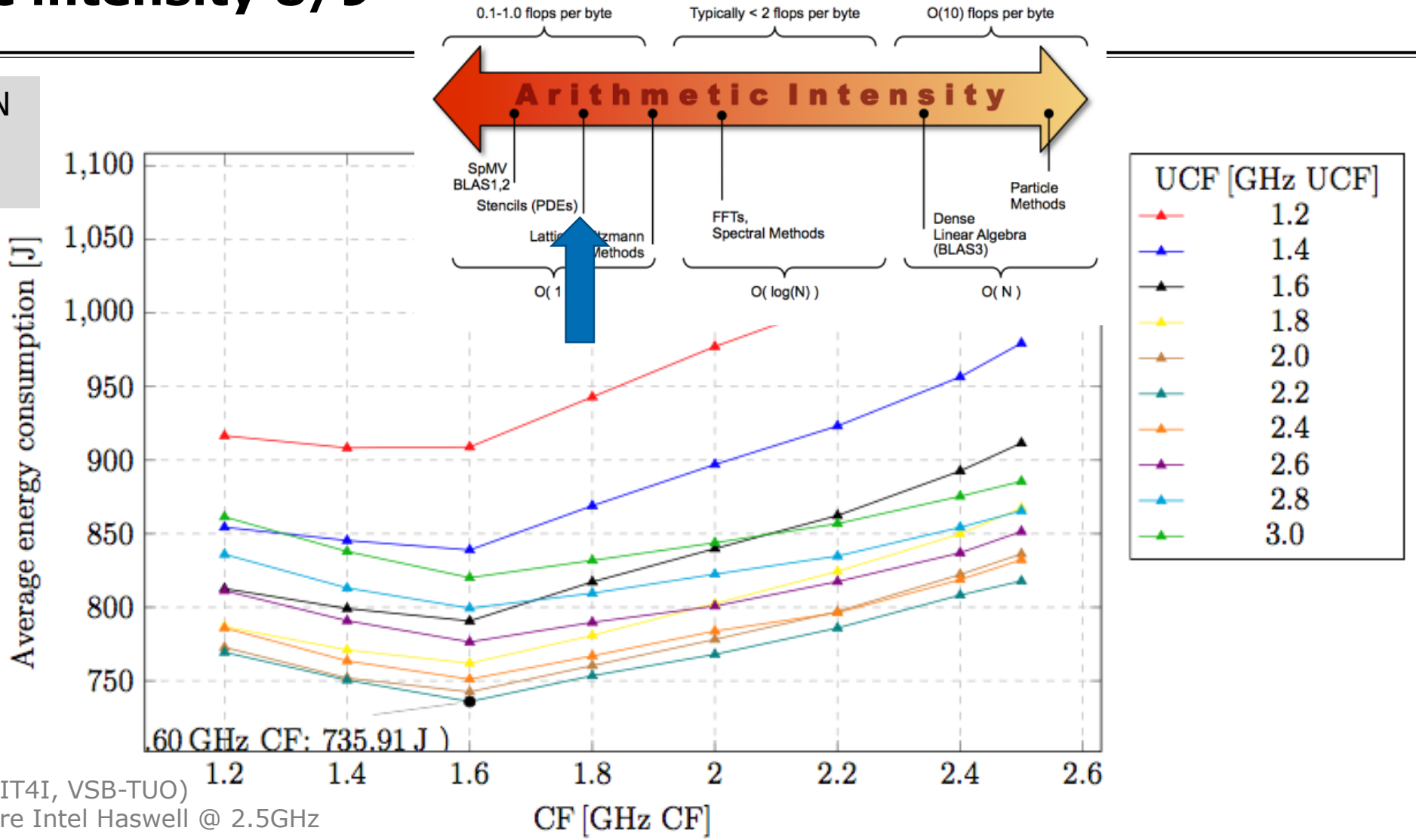
GEMV and TAN operations (ratio 7:3)



Machine: Salomon (IT4I, VSB-TUO)
Processor: 2x 12-core Intel Haswell @ 2.5GHz

Arithmetic intensity 8/9

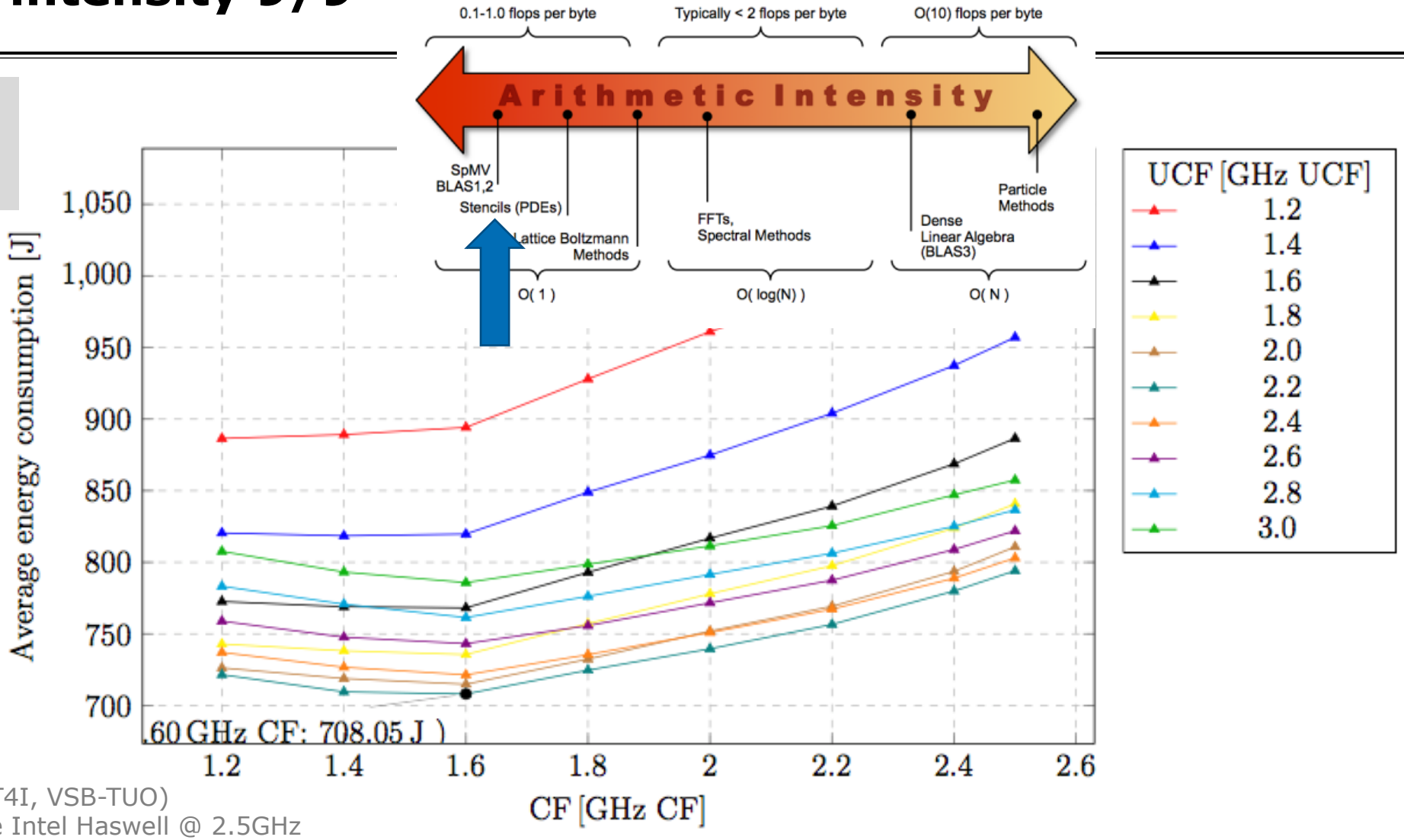
GEMV and TAN operations (ratio 8:2)



Machine: Salomon (IT4I, VSB-TUO)
Processor: 2x 12-core Intel Haswell @ 2.5GHz

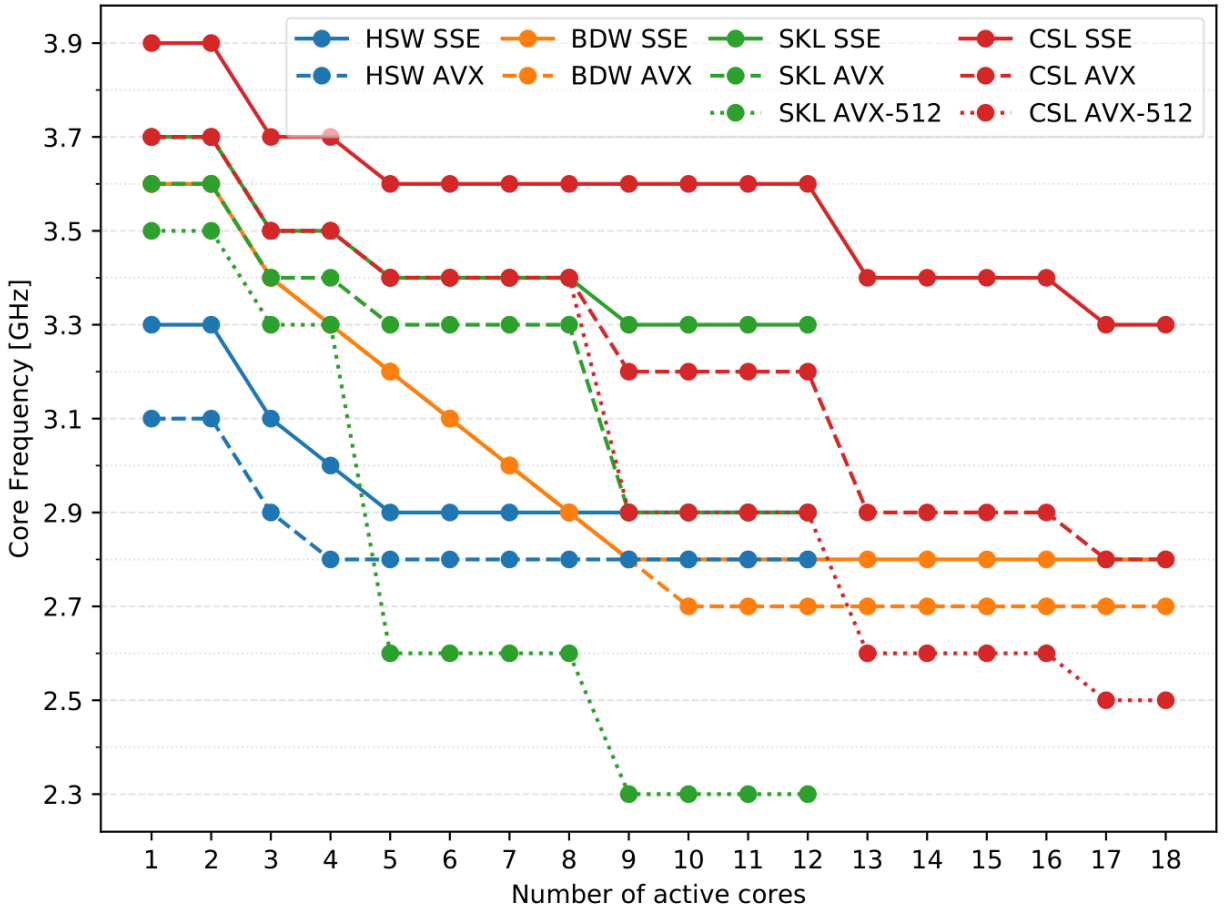
Arithmetic intensity 9/9

GEMV and TAN operations (ratio 9:1)



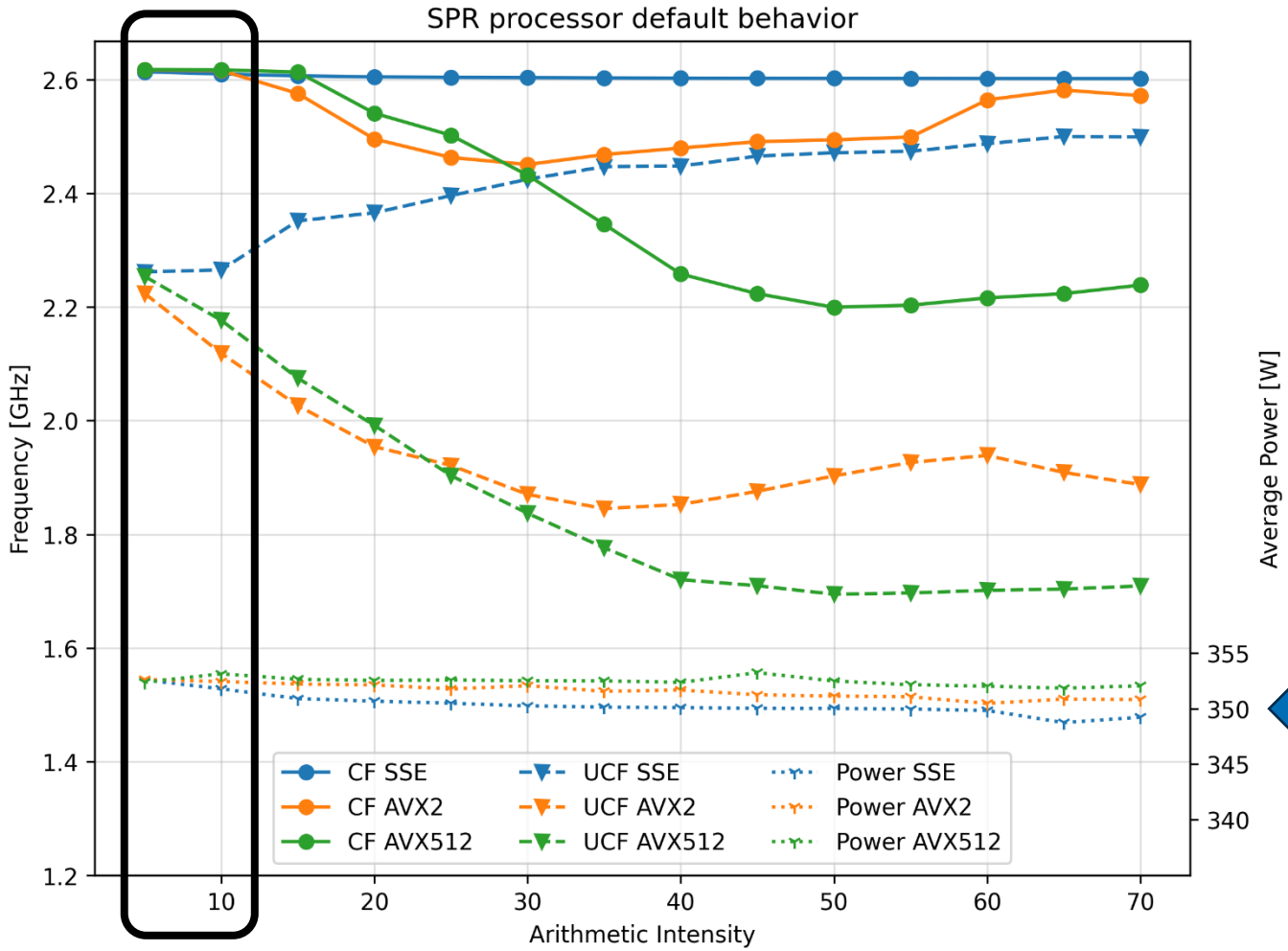
Machine: Salomon (IT4I, VSB-TUO)
Processor: 2x 12-core Intel Haswell @ 2.5GHz

Maximum core frequency, active cores, and vector instructions



HSW - Xeon E5-2680v3
BDW - Xeon E5-2697v4
SKL - Xeon Gold 6126
CSL - Xeon Gold 6240

Default behavior of Sapphire Rapids CPU



- Power capping system downscales CPU core and uncore frequencies to meet power limit
- Intel RAPL does not reflect the arithmetic intensity of the workload
  - Memory-bound codes with a low intensity would benefit from increased UCF and reduced CF
  - Instead, RAPL maintains CF at its maximum and reduces UCF by default

Thermal Design Power

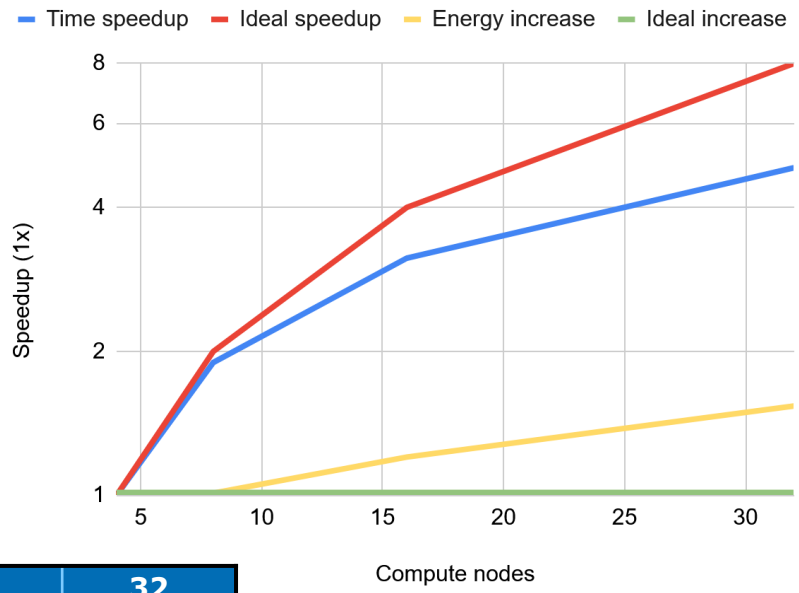
Energy consumption and efficiency assessment

Summary of previous important topics

- Energy consumption
  - Energy [J], Energy [Wh], Power [W]
- Energy efficiency [FLOPs/W]
- Power monitoring
  - RAPL, NVML, ROCm SMI, HDEEM, etc.
  - Node power baseline
- Power management
  - CPU: core and uncore frequencies [Hz], power limit [W]
  - GPU: streaming multiprocessor frequency [Hz], memory frequency [Hz], power limit [W]
  - Energy-efficient configurations for different workload types
    - Compute-bound versus memory-bound

Energy consumption

- Use of power monitoring systems
  - Consumption of the target application and machine baseline
- Energy consumption per component
  - CPUs, GPUs, DRAM => 1 compute node => complete allocation
- Average power consumption per component
- Can help with:
  - Identification of scaling issues
  - Calculation of average power utilization of HW in relation to TDP



Nodes	4	8	16	32
Wall time (s)	42.05	22.18	13.42	8.68
Total (J)	88,908	89,875	106,946	136,698
GPUs (J)	59,066	59,092	70,172	89,656
CPUs (J)	24,778	25,503	30,624	39,136
Average energy consumption				
1 node (J)	22,227	11,234	6,684	4,271
1 GPU (J)	3,691	1,846	1,096	700
1 CPU (J)	6,194	3,187	1,914	1,223
Average power consumption				
1 node (W)	528	506	498	491
1 GPU (W)	87	83	81	80
1 CPU (W)	147	143	142	140

Table: POP3 CoE

Power samples timeline

- Continuous power consumption based on collected power samples
- Analysis of the application's dynamic behavior (computational and idle phases, HW utilization)

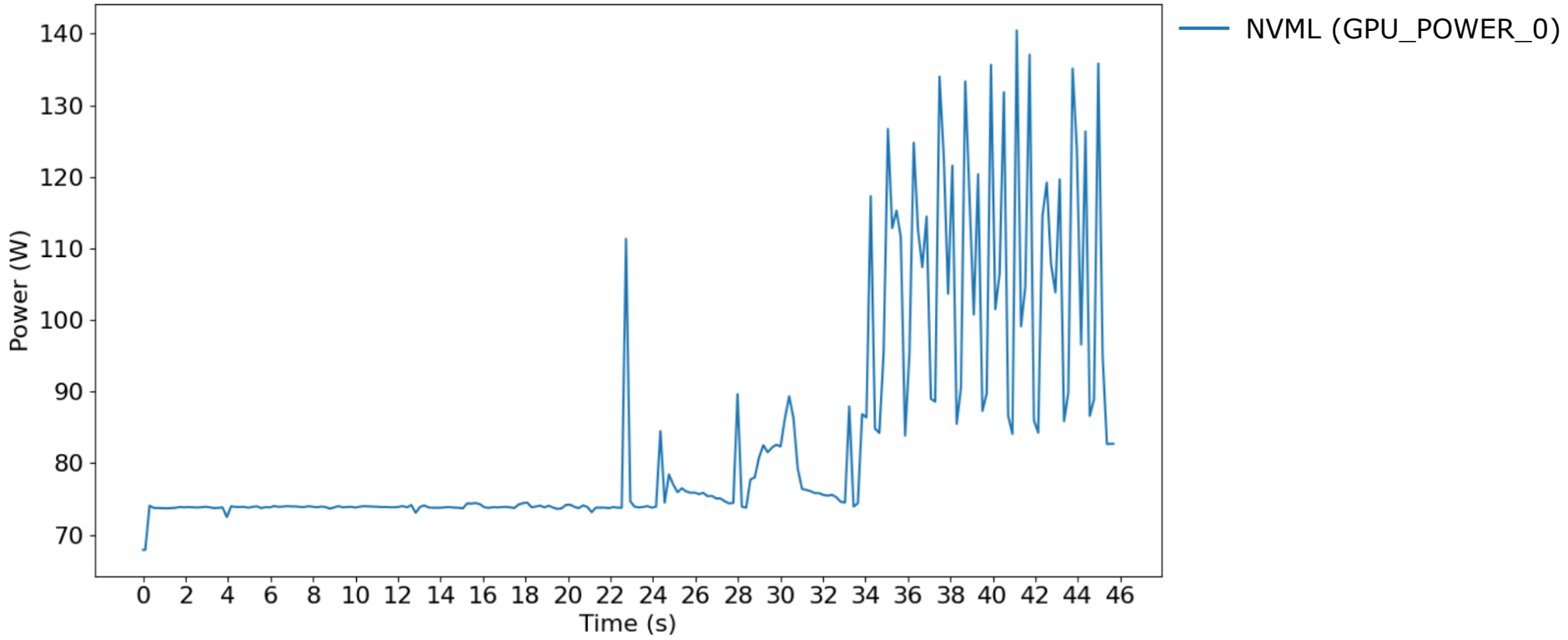


Figure: POP3 CoE

Performance counters (CPUs)

- Available via APIs like perf_events
- Available metrics depend on vendor, CPU generation, and system permissions
- Metrics:
  - CPU frequencies
  - FLOPs and vectorization
  - Temperatures
  - Memory bandwidth

Metric	Unit	Intel ICX	Intel SPR	AMD	Fujitsu A64FX	NVIDIA Grace
CPU_CORE_FREQUENCY	Hz	Core	Core	Core	Core	Core
CPU_UNCORE_FREQUENCY	Hz	Pckg	Pckg	-	-	Pckg
CPU_FLOP	FLOP/s	Core	Core	Core	Core	Core
CPU_AMX_ACTIVE_RATIO	%	-	Core	-	-	-
CPU_INSTRUCTION_SCALAR_FP/FP	%	Core	Core	-	-	-
CPU_INSTRUCTION_SCALAR_FP/ALL	%	Core	Core	-	Core	Core
CPU_INSTRUCTION_VEC_FP/FP	%	Core	Core	-	-	-
CPU_INSTRUCTION_VEC_FP/ALL	%	Core	Core	-	-	-
CPU_INSTRUCTION_VEC_128B_FP/FP	%	Core	Core	-	-	-
CPU_INSTRUCTION_VEC_128B_FP/ALL	%	Core	Core	-	-	-
CPU_INSTRUCTION_VEC_256B_FP/FP	%	Core	Core	-	-	-
CPU_INSTRUCTION_VEC_256B_FP/ALL	%	Core	Core	-	-	-
CPU_INSTRUCTION_VEC_512B_FP/FP	%	Core	Core	-	-	-
CPU_INSTRUCTION_VEC_512B_FP/ALL	%	Core	Core	-	-	-
CPU_INSTRUCTION_VEC_NEON/ALL	%	-	-	-	Core	Core
CPU_INSTRUCTION_VEC_SVE_FP/VEC_SVE	%	-	-	-	Core	-
CPU_INSTRUCTION_VEC_SVE_FP/ALL	%	-	-	-	Core	-
CPU_INSTRUCTION_VEC_SVE/ALL	%	-	-	-	Core	Core
CPU_COMPUTATIONAL_INTENSITY	instr./B	Core	Core	Pckg	Core	Core
CPU_CORE_TEMPERATURE	°C	Core	Core	-	-	-
CPU_PCKG_TEMPERATURE	°C	Pckg	Pckg	-	-	-
CPU_PCKG_PWRAPC_ACTIVE	%	Pckg	Pckg	-	-	-
DRAM_BANDWIDTH_READ	B/s	Pckg	Pckg	-	-	Pckg
DRAM_BANDWIDTH_WRITE	B/s	Pckg	Pckg	-	-	Pckg
DRAM_BANDWIDTH	B/s	Pckg	Pckg	Pkcg	-	Pckg
HBM_BANDWIDTH_READ	B/s	-	Pckg	-	Core	Pckg
HBM_BANDWIDTH_WRITE	B/s	-	Pckg	-	Core	Pckg
HBM_BANDWIDTH	B/s	-	Pckg	-	Core	Pckg

Energy efficiency

- Formulas:
  - Energy efficiency [FLOPs/Watt] = Performance [FLOPs] / Power [Watt]
  - Energy efficiency [FLOPs/Watt] = Operations [FLOP] / Energy [J]

- Comparisons of different hardware architectures and/or applications from one scientific domain

Code	HW platform	Efficiency
BigDFT	SPR+HBM	282 MFLOPs/W
Fleur	SPR+HBM, AMD Zen2	1.82 GFLOPs/W
QE	SPR+HBM	4.56 GFLOPs/W
Siesta	SPR+HBM	2.57 GFLOPs/W
Yambo	SPR+HBM	1,33 GFLOPs/W

HW platform	Energy efficiency
AMD Zen2 (DDR4)	2.95 GFLOPs/W
AMD Zen3 (DDR4)	3.28 GFLOPs/W
Intel Sapphire Rapids (HBM)	4.56 GFLOPs/W
Intel Sapphire Rapids (DDR5)	4.17 GFLOPs/W

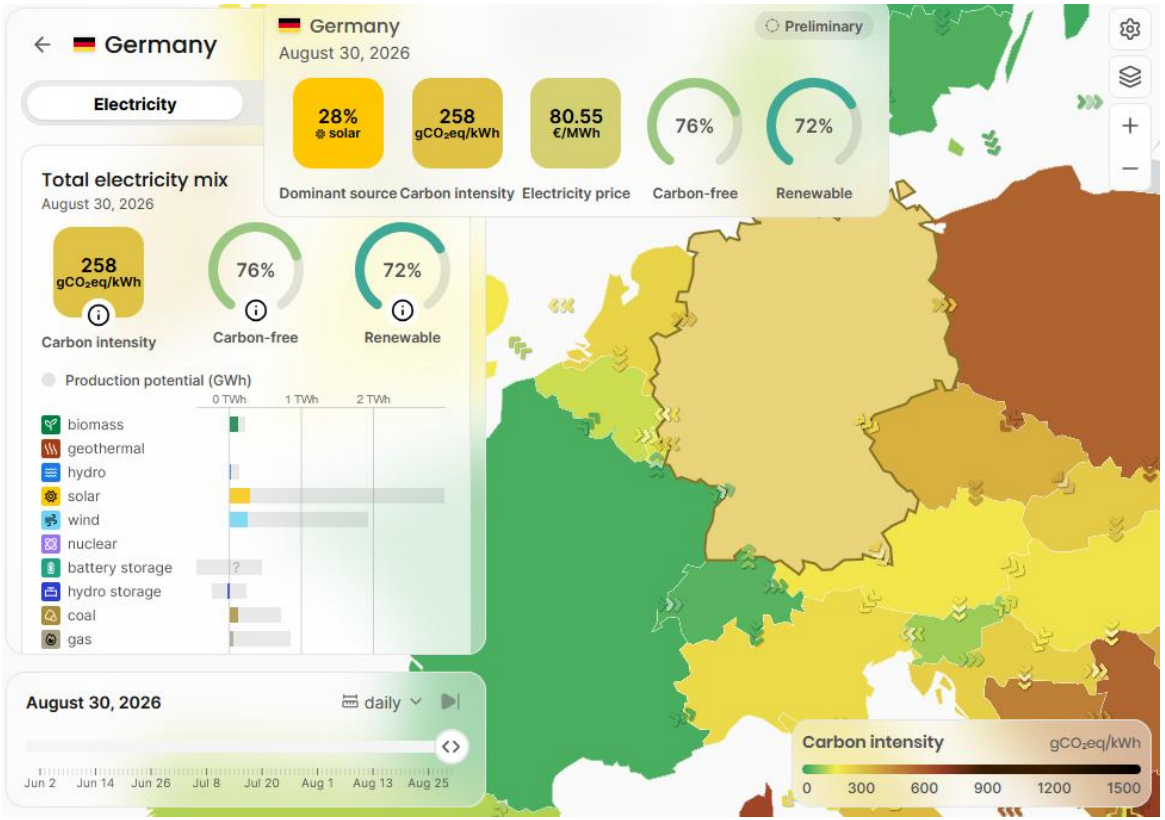
Inst. set	Proportion
Scalar	? %
SSE	? %
AVX/AVX2	? %
AVX512	? %

Tables: MAX3 CoE, D4.2

Carbon intensity and emissions

- Carbon intensity [gCO_2eq/kWh]
  - Number of grams of carbon dioxide equivalent needed to generate 1 kWh of electrical energy
- Carbon emissions [gCO_2eq]
  - = Energy consumption [kWh] × Carbon intensity [gCO_2eq/kWh]
- Approximate calculation for the application run based on country or region data
- It is important to understand the values
  - Kilometers driving a car
    (Toyota Corolla ST 1.8 Hybrid, 103 gCO_2eq/km)
  - Hours of flying
    (Boeing 747-400, 92 $kgCO_2eq$ per passenger per hour)

Electricity Maps (https://app.electricitymaps.com)



Search for an optimal energy-saving configuration

- Based on frequency tuning (CPU or GPU SM)
- Static frequency tuning vs. Dynamic frequency tuning
  - Static = one frequency for the complete application run
  - Dynamic = application divided into regions with a different frequency configured
- Set of selected frequency configurations that is tested
- Interesting configurations worth noting and comparing
  - Default configuration
    - A regular application run with no frequency tuning or any other change/tweak
  - Configuration with maximum energy savings
    - Focus on maximum energy savings no matter the performance penalty
    - Importance of node power baseline
  - Configuration with energy savings but with a small impact on the performance
    - Compromise between performance and energy savings

Results of the search for the energy-saving configurations

Hardware	Energy efficiency	Node energy consumption	HW configuration	Runtime
AMD Zen2 (RAPL + Baseline)	2.88 GFLOPs/W	243.21 kJ	default	442 s (100%)
	2.95 GFLOPs/W	237.53 kJ (**-2%**)	CF 2.7 GHz	100%
	3.11 GFLOPs/W	224.92 kJ (**-8%**)	CF 2.1 GHz	112%
AMD Zen3 (RAPL + Baseline)	3.02 GFLOPs/W	232.00 kJ	default	356 s (100%)
	3.28 GFLOPs/W	213.12 kJ (**-8%**)	CF 2.5 GHz	100%
	3.41 GFLOPs/W	204.98 kJ (**-12%**)	CF 2.1 GHz	110%
Intel Sapphire Rapids + HBM	4.46 GFLOPs/W	204.43 kJ RAPL	default	229 s (100%)
	4.56 GFLOPs/W	202.05 kJ (**-1%**) RAPL	CF default UCF 1.2 GHz	103%
	4.56 GFLOPs/W	202.05 kJ (**-1%**) RAPL	CF default UCF 1.2 GHz	103%
Intel Sapphire Rapids + DDR	4.00 FLOPs/W	227.07 kJ RAPL	default	253 s (100%)
	4.17 FLOPs/W	220.09 J (**-3%**) RAPL	CF 2.5 GHz UCF 1.6 GHz	100%
	4.25 FLOPs/W	218.07 J (**-4%**) RAPL	CF 2.3 GHz UCF 1.4 GHz	103%

Static frequency tuning

Default configuration

Energy savings with a low performance penalty

Maximum energy savings

Table: MAX3 CoE, D4.2

Tested frequency configurations and heatmaps

Intel Sapphire Rapids (HBM)

Static frequency tuning

CF/UnCF [MHz]	0	2500	2400	2200	2000	1800	1600	1400	1200	1000	800
0	229,59	235,37	235,08	235,01	234,83	235,03	234,44	234,49	236,96	245,63	263,56
3500	229,56	235,01	235,00	234,86	234,86	234,67	235,44	235,06	238,19	246,08	263,91
3300	229,76	236,19	236,53	236,07	235,61	235,59	236,70	236,72	238,72	248,27	264,95
3100	231,22	237,11	237,21	237,16	236,08	237,36	238,20	238,29	239,84	249,40	268,22
2900	232,28	238,11	239,87	237,50	238,98	238,73	238,95	239,11	242,74	252,74	271,85
2700	234,29	240,66	241,51	241,63	240,57	241,15	242,19	241,06	245,25	256,18	276,62
2500	236,55	243,15	242,98	243,41	244,39	243,61	243,57	245,14	249,61	260,55	280,34
2300	244,64	253,44	249,95	252,81	250,56	250,72	254,78	251,29	260,48	271,54	293,88
2100	257,28	260,12	260,28	260,52	261,99	264,05	265,58	269,32	281,90	302,30	331,29
1900	316,39	284,93	285,91	290,97	292,45	294,04	298,02	304,27	310,29	333,40	371,00
1700	350,57	322,17	324,36	325,14	326,01	329,76	338,38	341,68	349,28	359,47	382,49
00	418,46	401,02	405,59	406,46	405,55	408,79	414,47	417,73	422,79	433,83	406,65
00	454,31	476,03	444,40	438,32	438,98	436,16	429,50	432,62	433,80	443,95	455,90
00	545,31	537,44	534,32	535,11	537,81	542,13	544,83	548,15	560,26	564,43	587,89
00	733,92	746,16	756,92	747,11	752,42	739,06	719,35	697,62	673,14	661,13	660,78
00	772,98	778,35	753,32	774,73	764,98	783,86	765,46	756,49	767,34	766,42	769,10

Run time [s]

CF/UnCF [MHz]	0	2500	2400	2200	2000	1800	1600	1400	1200	1000	800
0	204,43	210,64	210,31	209,93	208,73	207,34	205,09	202,78	202,05	205,69	214,80
3500	204,43	210,31	210,26	209,83	208,78	207,05	205,92	203,18	202,86	206,07	215,00
3300	204,82	211,36	211,60	210,87	209,42	207,76	206,77	204,34	203,24	207,40	215,67
3100	206,14	212,17	212,16	211,79	209,78	209,20	207,96	205,40	204,01	208,15	217,72
2900	207,09	213,04	214,43	212,10	212,14	210,20	208,43	205,97	205,88	210,37	220,03
2700	208,89	215,23	215,83	215,48	213,36	212,09	210,80	207,38	207,62	212,59	223,09
2500	210,76	217,35	217,03	216,82	216,10	213,45	211,46	209,68	210,04	214,65	224,60
2300	217,17	225,98	222,74	223,37	219,09	216,52	216,54	211,02	213,65	217,84	228,58
2100	227,68	231,43	230,44	227,70	225,56	223,26	220,83	218,97	222,34	231,17	245,02
1900	248,27	251,34	250,10	249,65	245,80	241,44	237,93	236,03	234,64	244,50	262,45
1700	268,51	280,98	279,88	274,07	267,19	260,13	258,24	253,87	252,42	254,04	262,99
1500	284,16	342,42	340,87	328,95	314,69	304,79	299,63	293,77	289,54	290,66	270,71
1300	294,38	397,95	366,84	345,14	330,71	316,23	302,86	296,38	289,82	290,23	291,92
1100	334,07	438,58	424,79	404,05	387,79	374,71	364,80	356,61	354,09	349,90	356,20
900	433,88	581,54	573,04	536,68	515,22	486,05	458,75	433,39	408,85	394,40	386,82
800	444,80	600,96	566,05	550,34	518,51	508,53	481,00	462,41	455,94	445,88	439,06

Energy consumption [kJ]

(RAPL + baseline)

(0,0) Default configuration

Tables: MAX3 CoE, D4.2

Static frequency tuning on GPUs (NVIDIA A100)

Set GPU frequency [MHz]	Runtime [s]	Runtime extension	Average power consumption of the GPU [W]	Average power consumption of the CPU [W]	CPUs + GPUs energy consumption [kJ]	Energy savings for GPUs + CPUs	Node energy consumption [kJ]	Compute node energy savings	Note
1410	112,1	100,0%	163,2	116,5	172,4	0,0%	239,7	0,0%	**Default settings**
1350	112,1	100,0%	153,4	116,5	163,7	5,1%	230,9	3,7%	
1290	112,9	100,7%	141,8	116,6	154,4	10,4%	222,2	7,3%	**No runtime penalty**
1230	117,1	104,5%	131,7	116,7	150,6	12,7%	220,8	7,9%	
1170	119,5	106,7%	123,7	117,9	146,4	15,1%	218,2	9,0%	
1110	124,7	111,3%	114,9	117,7	144,0	16,5%	218,8	8,7%	
1050	124,9	111,5%	110,3	117,9	139,6	19,0%	214,6	10,5%	**Maximum energy savings**
990	128,7	114,9%	107,2	118,0	140,8	18,4%	218,0	9,0%	
930	133,2	118,9%	105,1	117,8	143,3	16,9%	223,2	6,9%	
870	137,1	122,3%	103,4	118,4	145,9	15,4%	228,1	4,8%	
810	144,3	128,8%	100,6	118,4	150,2	12,9%	236,8	1,2%	
750	147,3	131,5%	99,4	116,3	151,4	12,2%	239,8	-0,1%	
690	155,3	138,6%	96,8	117,0	156,5	9,2%	249,7	-4,2%	

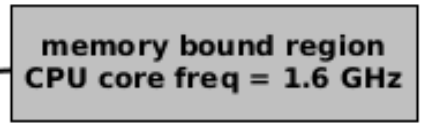
Table: MAX3 CoE, D4.2

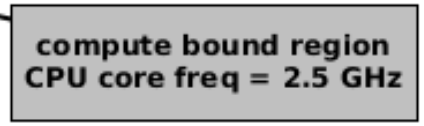
Dynamic frequency tuning

```
void lbm(void) {
  // init application parameters
  // init mass
  for (int iter = 0; iter < NITER; iter++) { // phase region
    update_hallos();  // insignificant region

    propagate();      // significant region

    collide();        // significant region
  }
  // post-processing
  // store output
  // terminate application
}
```

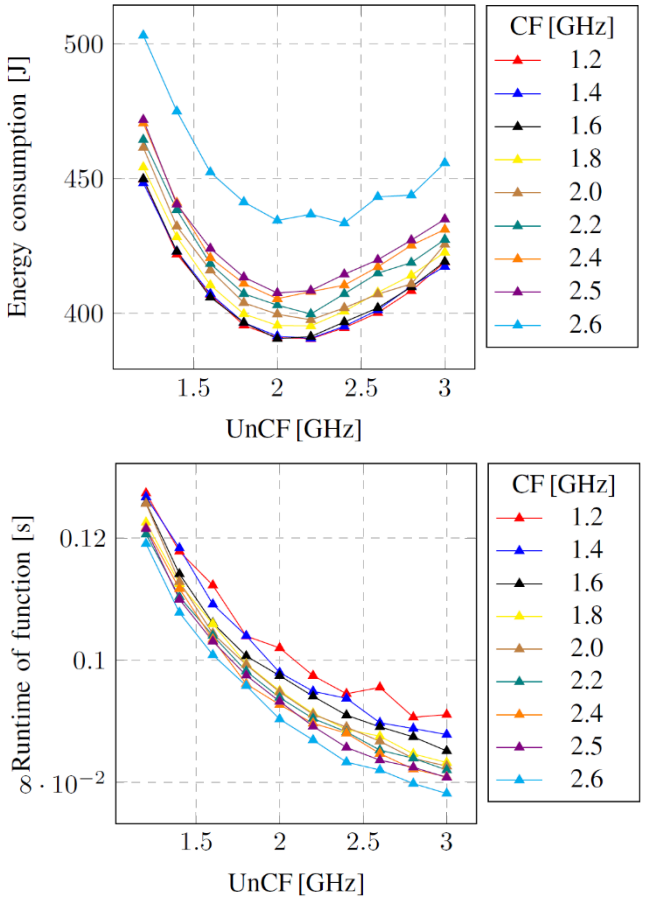




Dynamic frequency tuning: Lattice-Boltzmann benchmark

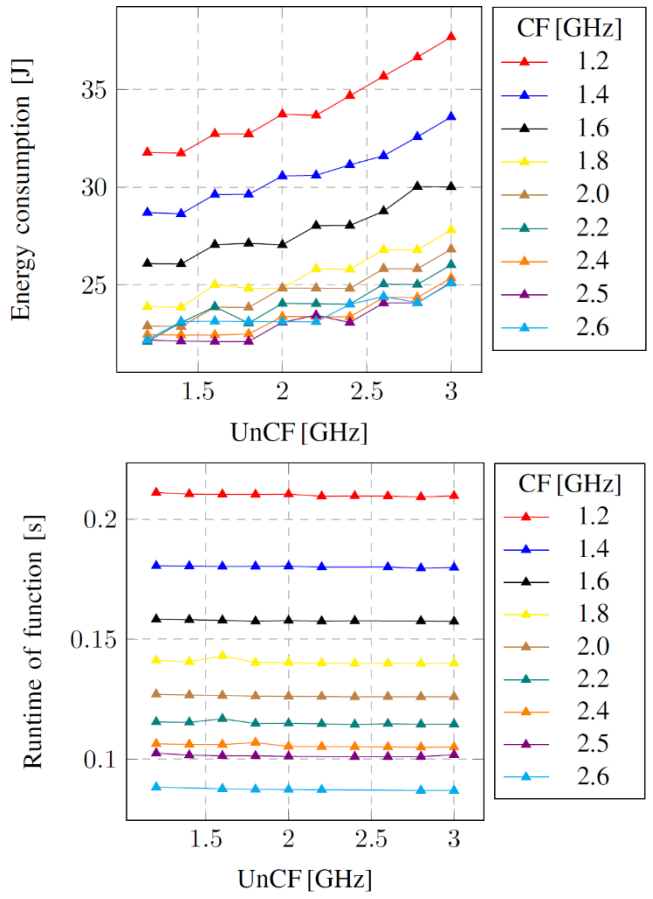
The `propagate()` region

memory-bound



The `collide()` region

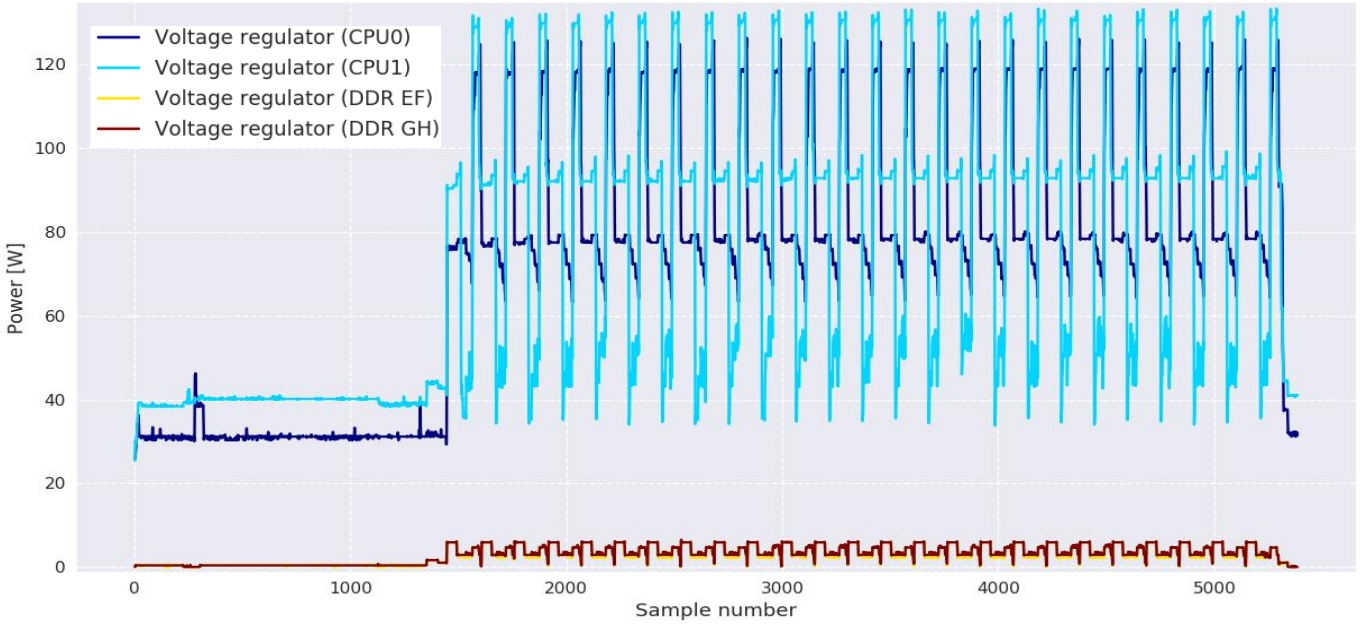
compute-bound



Configuration	Time	Energy
Default	24 s	5.7 kJ
Static freq. tuning	+11.6 %	-7.8 %
Dynamic freq. tuning	**+1.5 %**	**-19.8 %**

Dynamic frequency tuning: READEX methodology

- EU H2020: READEX, 2015 – 2018
- Complex parallel applications
  - Different requirements during execution
  - Possibility: dynamic tuning to save energy without performance penalty
- Goal: a tools-aided methodology for automatic tuning of parallel applications
  - Dynamic adjustment of the system parameters according to actual resource requirements



TECHNISCHE UNIVERSITÄT DRESDEN

NTNU

intel

IT4Innovations national supercomputing center

NUI Galway OÉ Gaillimh

gns

MERIC and RADAR

MERIC Runtime System

MERIC

- Runtime system for dynamic application tuning developed in C++
- Infrastructure Research Lab, IT4Innovations, VSB-TUO, Ostrava, Czech Republic

- Support
  - Parallel paradigms: MPI, OpenMPI, CUDA, HIP
  - Programming languages (APIs): C/C++, Fortran, Python
  - Architectures: x86-64, IBM OpenPower, ARM, NVIDIA and AMD GPUs
  - Power monitoring: RAPL, OCC, HDEEM, NVML, ROCm, HWMON, A64FX

- Important features
  - Energy-consumption measurement of an application
  - Analysis of an application dynamism and energy efficiency
  - Static and dynamic tuning of HW power knobs for energy savings
  - HW & SW power management co-design

MERIC

https://code.it4i.cz/energy-efficiency/meric-suite/meric

Static tool for simple energy measurements

```
# CLI tool mericStatic
$ mericStatic -e RAPL -- <application> [application parameters]
$ mericStatic -- eval
measurement_id : <ID>
Max Runtime [s] = <TIME>
RAPL Energy consumption [J] = <JOULES>

Total Energy consumption [J] = <JOULES>
Total Energy consumption [Wh] = <WATT-HOURS>

# The results are stored in a mericStaticMeasurements folder
# Argument –h can be used to display all options of the tool
# Parallel execution with srun
$ srun mericStatic –e RAPL -- <application> [application parameters]
```

EuroHPC systems and their support of energy measurements

		JUPITER	LUMI	Leonardo	MN5	Vega	MeluXina	Karolina	Discoverer	Deucalion
AMD CPU	intel_rapl kernel module									
	amd_energy kernel module									
	msr_safe kernel module									
	perf									
Intel CPU	intel_rapl kernel module									
	msr_safe kernel module									
	perf									
A64FX	perf									
Nvidia GPU	NVML									
AMD GPU	ROCm									
GraceHopper	HWMON									

legend	AVAILABLE	LIMITED	NO

The systemInfo tool 1/2

```
$ systemInfo
# SYSTEM INFORMATION
        CPU name:          Intel(R) Xeon(R) Platinum 8468
        Sockets per node: 2
        Cores per socket: 48
        Threads per core: 2

# CPU FREQUENCIES
        Current scaling driver:      intel_cpufreq
        Current scaling governor:    performance
        Available governors:         conservative ondemand userspace powersave performance schedutil
        Hardware controlled P-State: available, status unknown

        CPU core frequency scaling (DVFS) NOT available
                 install one the following libraries:
                 - msr_safe / cpufreq / GEOPM
                 grant write access to:
                 - /sys/devices/system/cpu/cpu*/cpufreq/scaling_governor
                 - /sys/devices/system/cpu/cpu*/cpufreq/scaling_setspeed

        Max CPU core frequency:      3800000 kHz
        Nominal CPU core frequency:  2100000 kHz
```

The systemInfo tool 2/2

```
        Min CPU core frequency:      800000 kHz
# CPU uncore frequency NOT available - install libmsr or x86_adapt

# CPU POWER LIMITS
        PKG max power limit #1: 350 W
        PKG max power limit #2: 764 W
        DRAM max power limit:   61 W

        changing RAPL power capping limits NOT available - install msr_safe+libmsr or msr_safe or x86_adapt

# DEFAULT CPU POWER LIMITS
        PKG power capping:  enabled
        PKG power limit #1: 350 W
        PKG time window #1: 0.999424 s
        PKG power limit #2: 420 W
        PKG time window #2: 0.011712 s
        DRAM power capping: disabled
        DRAM power limit:   0 W
        DRAM time window:   0.000976 s

# AVAILABLE POWER MONITORING SYSTEMS
        RAPL
```

API for application instrumentation

```
#include <meric.h>

int main(int argc, char **argv) {
        MERIC_Init(); // In an MPI application, insert this after MPI_Init

        MERIC_MeasureStart("A");
        do_work_A(2);
        MERIC_MeasureStop("A");

        MERIC_MeasureStart("B");
        do_work_B(5);
        MERIC_MeasureStop("B");

        MERIC_Close(); // In an MPI application, insert this before MPI_Finalize
        return 0;
}
```

MERIC library

- libmeric.so/a – for non-MPI codes with OpenMP
- libmericmpi.so/a – for hybrid applications with MPI and OpenMP
- libmericmpionly.so/a – for pure MPI applications
  - MERIC must be compiled with `make noopenmp` to build this library
- Further information at https://code.it4i.cz/energy-efficiency/meric-suite/meric#6-compilation

Configuration of the instrumented run 1/2

```
# Example configuration for a default run of an NVIDIA GPU application
# A subset of all existing settings of MERIC is presented below

# Hardware tuning settings
export MERIC_FREQUENCY=0 # [Hz] no CPU core frequency tuning
export MERIC_UNCORE_FREQUENCY=0 # [Hz] no CPU uncore frequency tuning
export MERIC_NUM_THREADS=0 # Non-OpenMP application
export MERIC_PWRCAP_POWER=0 # [mW] no CPU power capping
export MERIC_PWRCAP_TIME=0 # [ms] no CPU power capping
export MERIC_GPU_FREQUENCY=0 # [Hz] no GPU SM frequency tuning
export MERIC_GPU_MEM_FREQUENCY=0 # [Hz] no GPU memory frequency tuning
export MERIC_PWRCAP_GPU=0 # [mW] no GPU power capping
```

Further information at https://code.it4i.cz/energy-efficiency/meric-suite/meric#7-meric-input-parameters

Configuration of the instrumented run 2/2

```
# Energy and power consumption measurement
export MERIC_MEASURE=RAPL,NVML # Comma-separated list from {NONE, TIME, RAPL, NVML,…}
export MERIC_SAMPLES=1 # Enable power sampling
export MERIC_AGGREGATE=0 # Store data for each node separately (0 required for sampling)
export MERIC_DETAILED=1 # Detailed information about the system

# HW performance counters
export MERIC_COUNTERS=perfevent # {none | msr | perfevent}

# Measurement directory
export MERIC_OUTPUT_DIR=<path-to-measurement-dir>
```

Further information at https://code.it4i.cz/energy-efficiency/meric-suite/meric#7-meric-input-parameters

Analysis without hardware tuning

- Instrumented run with the default system configuration
- Sufficient to assess:
  - Energy and power consumption
  - Dynamic behavior of the application
    - Power samples timeline
  - Energy efficiency and vectorization
    - HW performance counters (perf_events)
  - Carbon emissions

Frequency tuning

- CPU vs. GPU application
  - CPU – Core and uncore frequencies
  - GPU – Streaming-multiprocessor frequency
  - `systemInfo` provides information about available frequency levels

- Static vs. dynamic tuning
  - Static – instrumented runs with varying configurations of global parameters (environmental variables)
    - `MERIC_FREQUENCY`, `MERIC_UNCORE_FREQUENCY`, `MERIC_GPU_FREQUENCY`
  - Dynamic – instrumented application with several regions
    - Configuration of individual regions via a MERIC configuration file
    - More details at https://code.it4i.cz/energy-efficiency/meric-suite/meric#dynamic-tuning

MERICwrapper: Automated frequency tuning

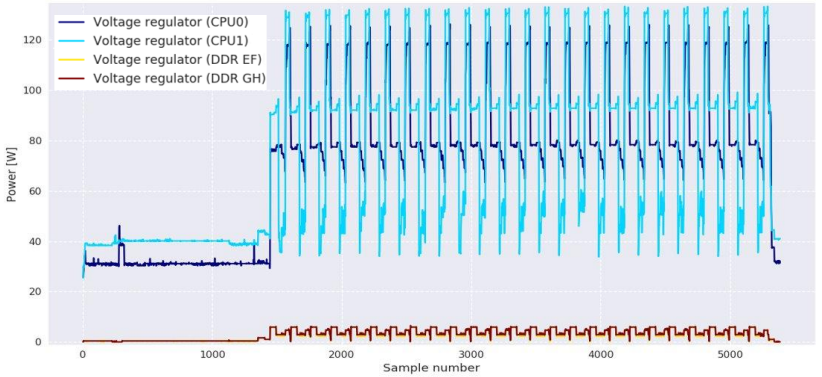
```
$ wrapper -s config.json -- srun <application> [application parameters]
# https://code.it4i.cz/energy-efficiency/meric-suite/meric/-/tree/master/src/tools/wrapper
```

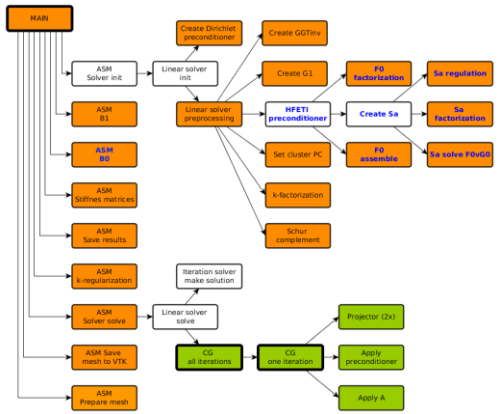
- Automated tuning based on an input JSON configuration file
  - Configuration of MERIC
  - HW configurations to test
- State-space search
- The tuning is applied to all instrumented regions of the application
- Identification of the optimal configuration for each region

RADAR visualizer

- Visualization of the application behavior in various HW configurations

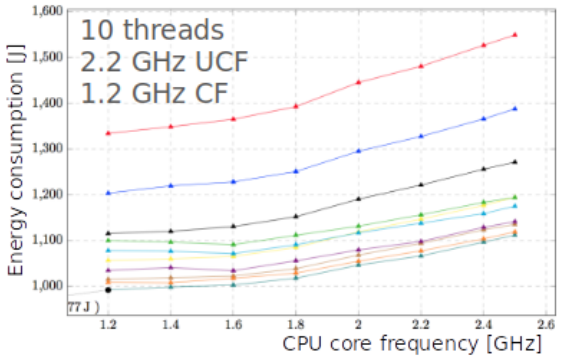
- Overall application evaluation
- Evaluation per region
- Call-path graph
- Plots
- Power samples timeline
- Heatmaps





Uncore freq [GHz] / Core freq [GHz]	1.2	1.4	1.6	1.8	2.0	2.2	2.4	2.6	2.8	3.0
1.2	13,200.02	12,717.1	12,621.78	12,410.62	12,380.68	12,507.38	12,774.16	13,108.6	13,604.2	14,040.8
1.4	13,161.9	12,597.78	12,125.18	12,065.52	12,074.54	12,173.36	12,312.24	12,802.26	13,095.84	13,450.8
1.6	13,320.66	12,640.76	12,256.22	12,033.62	11,966.36	11,992.7	12,372.04	12,579.22	13,126.44	13,370.24
1.8	13,878.04	13,082.66	12,700.92	12,457.08	12,373.86	12,445.98	12,574.6	12,831.82	13,081.62	13,296.04
2	14,218.58	13,327.12	12,902.62	12,544.82	12,456.82	12,494.8	12,680.32	13,038.86	13,207.38	13,474.8
2.2	14,625.62	13,849.58	13,240.14	12,851	12,760.98	12,802.24	12,993.44	13,260.38	13,497.6	13,767.62
2.4	15,083.2	14,412.62	13,568.68	13,447.18	12,973.38	13,238.6	13,332.7	13,388.7	13,777.68	14,030.66
2.5	15,554.96	14,465.2	13,991	13,553.84	13,300.24	13,354.46	13,472.36	14,179.16	14,083.06	14,231.3

	Default settings	Default values	Best static configuration	Static savings	Dynamic savings
Runtime of function [s], Job info - rapl	3.0GHz, 2.5GHz	1.97s	3.0GHz, 2.5GHz	0.00s (0.00%)	0.015s of 1.97s (0.76%)
Energy summary, COUNTERS - rapl:	3.0GHz, 2.5GHz	800.37	2.4GHz, 2.5GHz	19.70 (2.46%)	46.52 of 780.67 (5.96%)
Run-time change with the energy optimal settings	+0.14s (107.04 % of default time)				



Thank You

MERIC

Contact: meric@it4i.cz

MERIC: https://code.it4i.cz/energy-efficiency/meric-suite/meric

RADAR: https://code.it4i.cz/energy-efficiency/meric-suite/radar-visualizer

This work was supported by the POP3 project under grant agreement No 101143931. The project is supported by the European High-Performance Computing Joint Undertaking and its members (including top-up funding by the Ministry of Education, Youth and Sports of the Czech Republic ID: MC2401).