Dynamic Tuning of HPC Applications

VSB TECHNICAL UNIVERSITY OF OSTRAVA | IT4INNOVATIONS NATIONAL SUPERCOMPUTING CENTER

Overview

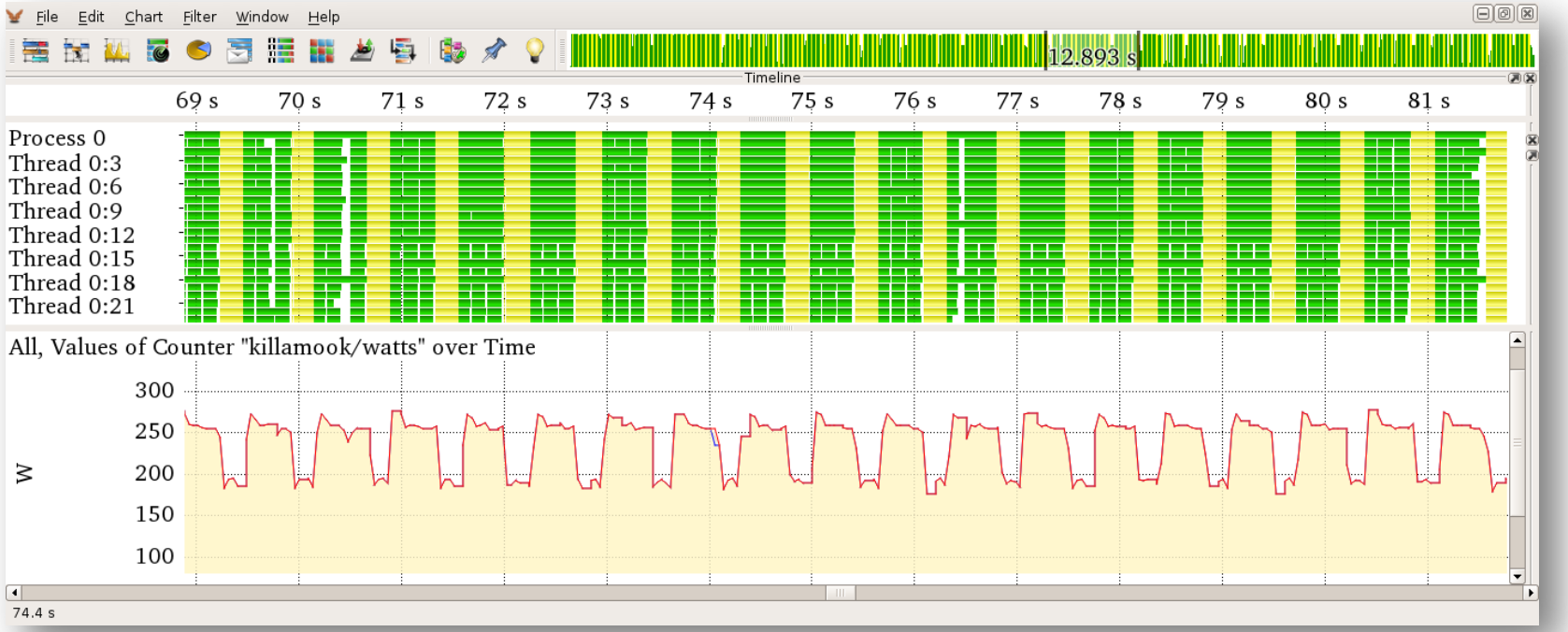
Methodology

- Motivation and Introduction
  - READEX project overview
  - What can you achieve with static and dynamic tuning
- Tuning of the hardware parameters
  - Effect on the energy consumption
- Effect of hardware parameter tuning on kernels with various arithmetic intensity
- Evaluation of complex HPC applications
  - BEM4I
  - OpenFOAM
  - Scalability tests with ESPRESO
- Tuning of the application parameters

Motivation and Introduction

READEX Project & Motivation

- Energy efficiency is critical to current and future systems
- Applications exhibit dynamic behavior
  - Changing resource requirements
  - Computational characteristics
  - Changing load on processors over time



TECHNISCHE UNIVERSITÄT DRESDEN

NTNU

VSB TECHNICAL UNIVERSITY OF OSTRAVA | IT4INNOVATIONS NATIONAL SUPERCOMPUTING CENTER

NUI Galway
OÉ Gaillimh

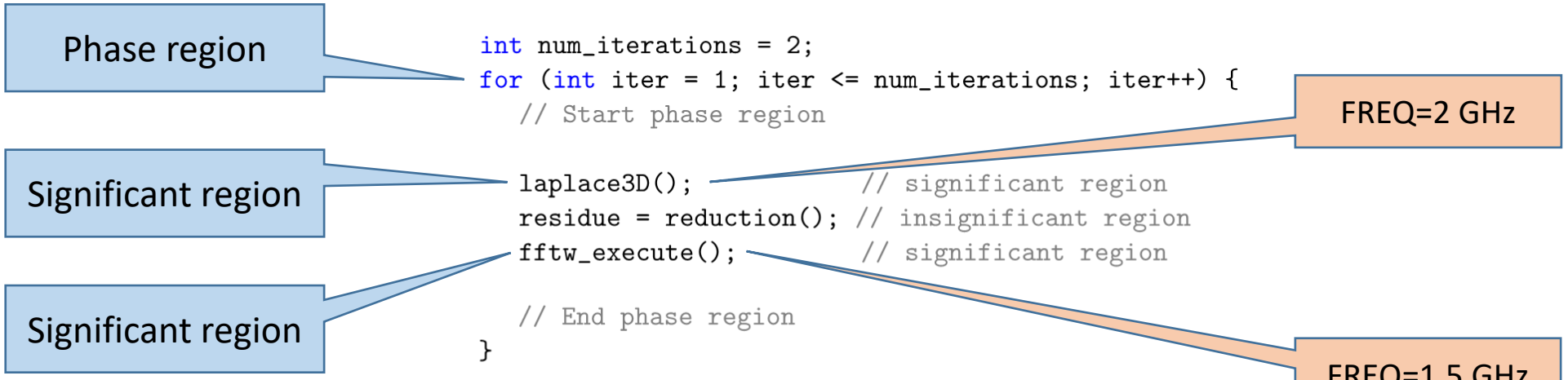
intel

gns

Goal was to create a tools-aided methodology for automatic tuning of parallel applications.

Dynamically adjust system parameters to actual resource requirements

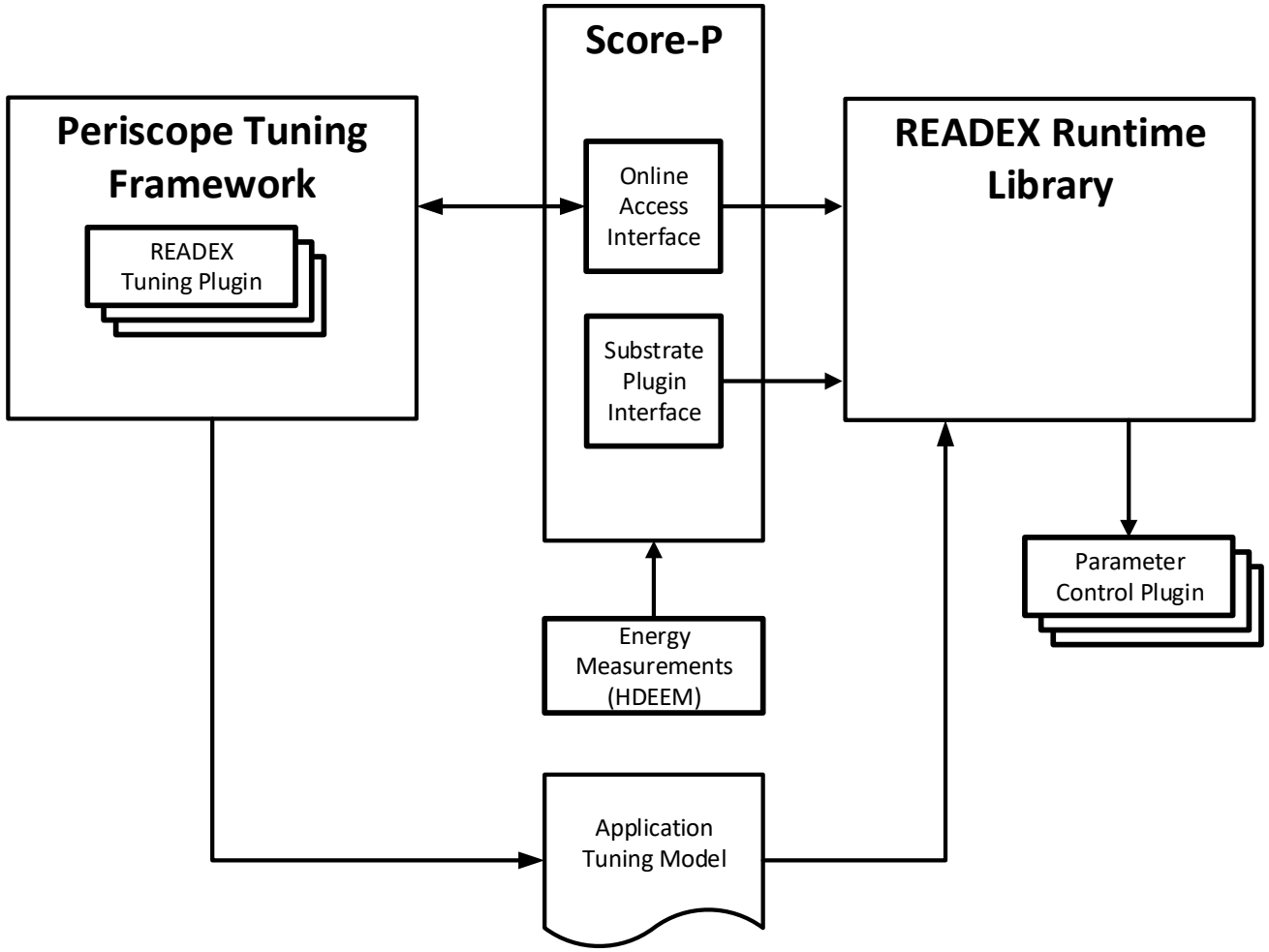
What is dynamic tuning




```
int main(void) {

    // Initialize application
    // Initialize experiment variables

    int num_iterations = 2;
    for (int iter = 1; iter <= num_iterations; iter++) {
       // Start phase region

       laplace3D();           // significant region
       residue = reduction(); // insignificant region
       fftw_execute();        // significant region

       // End phase region
    }

    // Post-processing:
    // Write noise matrices to disk for visualization
    // Terminate application

    MPI_Finalize();
    return 0;
}
```

READEX Tool Suite

1. Instrument application
   - Score-P provides different kinds of instrumentation
2. Detect dynamism
   - Check whether runtime situations could benefit from tuning
3. Detect energy saving potential and configurations (DTA)
   - Use tuning plugin and power measurement infrastructure to search for optimal configuration
   - Create tuning model
4. Runtime application tuning (RAT)
   - Apply tuning model, use optimal configuration



READEX Tool Suite

READEX Test Suite

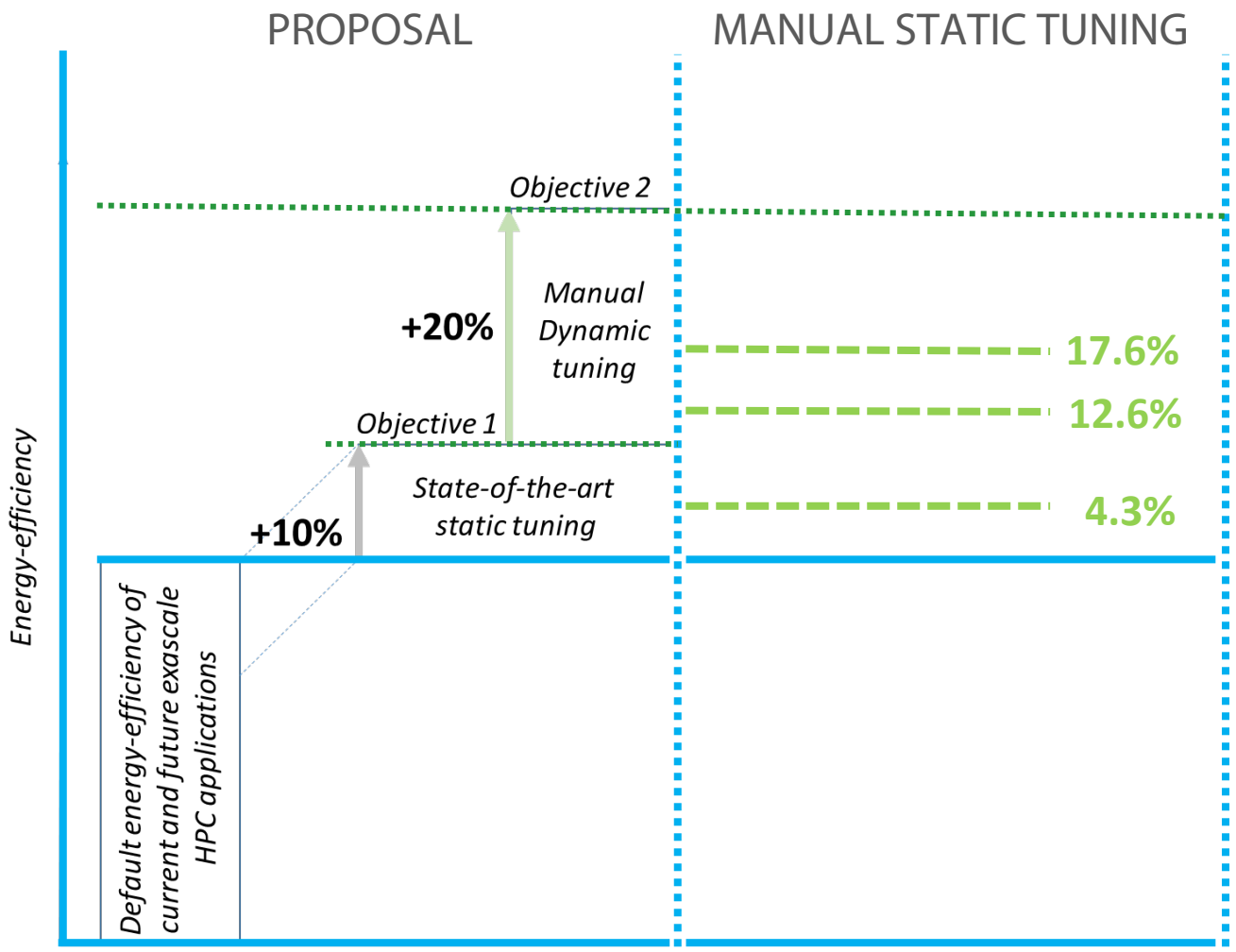
Consists of benchmarks, proxy apps and complex production applications

Key features:

- Full set of scripts allows reproducibility of experiments on
  - TUD Taurus HSW (HDEEM) and BDW partitions
  - IT4I Salomon machine (RAPL)
- Support for Slurm and PBS schedulers
- Automatic savings evaluation
- Performs evaluation of
  - hardware and system parameter tuning
  - application parameter tuning
- Contains manual instrumentation of significant regions
  - using header file → can be adopted to test other tools

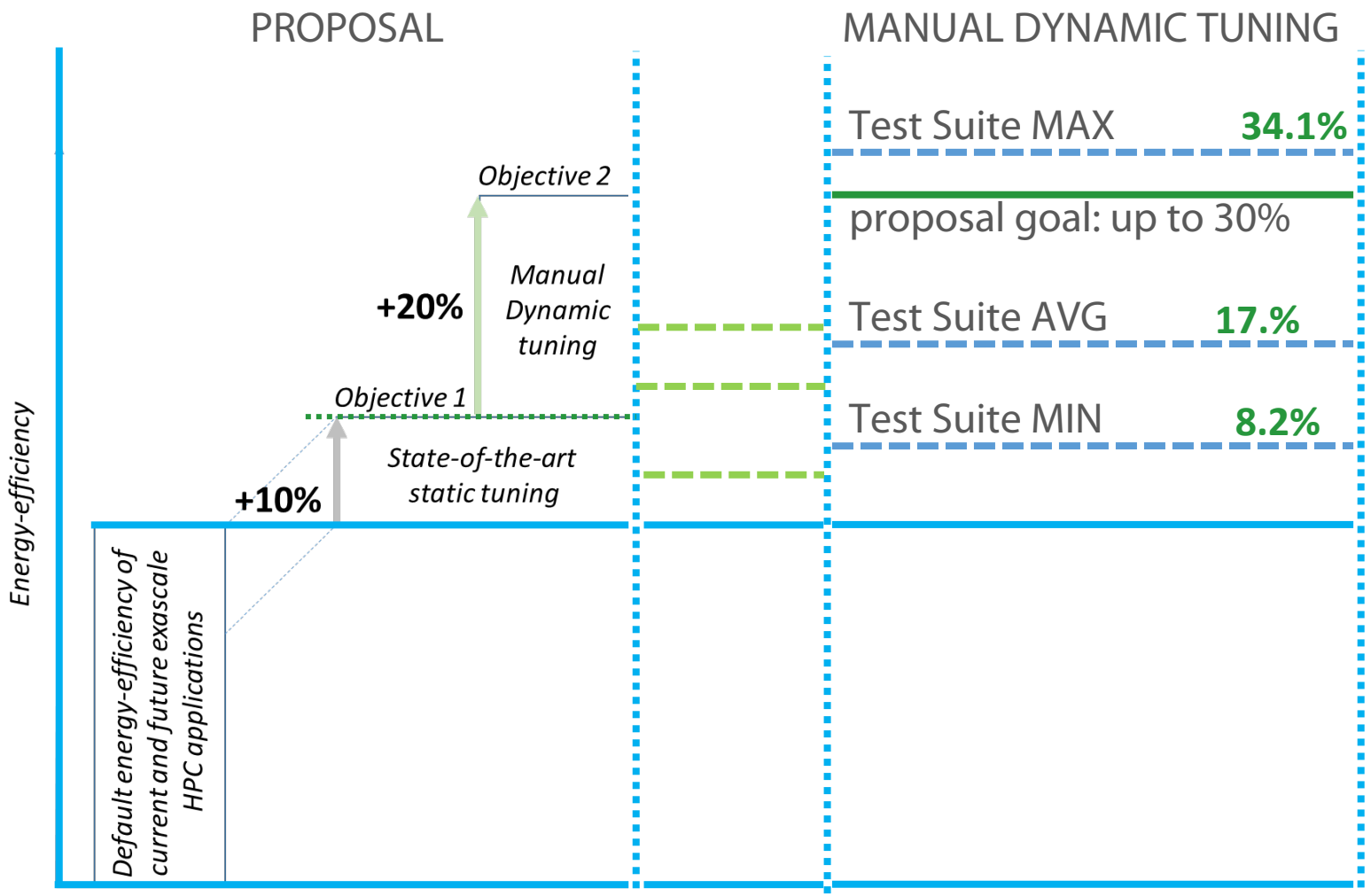
Application type	Application name
benchmarks or proxy apps	AMG2013
	Blasbench
	Kripke
	Lulesh
	NPB3.3
production applications	BEM4I
	ESPRESO
	INDEED
	OpenFOAM

What can you expect from static tuning



Software	Static tuning savings
AMG2013	12.5 %
Blasbench	7.4 %
Kripke	11.5 %
Lulesh	17.6 %
NPB3.3	11.0 %
BEM4I	15.7 %
INDEED	17.6 %
ESPRESO	4.3 %
OpenFOAM	15.9 %
Average	**12.6 %**

What can you expect from dynamic tuning



Software	Dynamic tuning savings
AMG2013	12.5 %
Blasbench	15.3 %
Kripke	18.5 %
Lulesh	18.7 %
NPB3.3	11.0%
BEM4I	34.1 %
INDEED	19.5 %
ESPRESO	8.2 %
OpenFOAM	20.1%
Average	**17.5 %**

Energy savings achieved by static and dynamic tuning

Application (default is Intel compiler) (* uses GCC compiler)	HW parameters	Static tuning saving node energy / time	Dynamic tuning savings node energy/time
AMG2013	CF, UCF, threads	12.5% / –0.9%	N/A
Blasbench	CF, UCF, threads	7.4% / –0.9%	15.3% / –18.1%
Kripke	CF, UCF	11.5% / –28.3%	18.8% / – 18.7%
Lulesh	CF, UCF, threads	17.6% / –8.9%	18.7% / –11.7%
NPB3.3-BT-MZ	CF, UCF, threads	11% / –11.3%	N/A
BEM4I	CF, UCF, threads	15.7% / –6.2%	34.1% / 10.9%
INDEED	CF, UCF, threads	17.6% / –12.8%	19.5% / –14.2%
ESPRESO	CF, UCF, threads	4.3% / –8.9%	8.2% / –10.1%
OpenFOAM	CF, UCF	15.9% / –10.5%	20.1% / 11.5%

Evaluation of READEX Tool Suite on TUD Taurus Haswell system with HDEEM energy measurements

Key findings:

- Best savings achieved with BEM4I application – up to 34% for energy and 11% for runtime
- In general energy savings are ”paid” by extra runtime

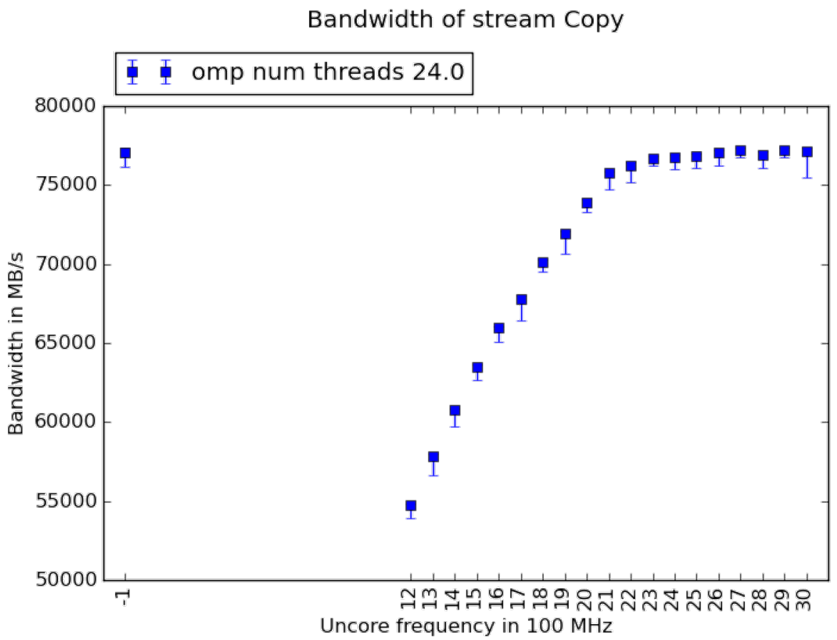
Tuning of the hardware parameters

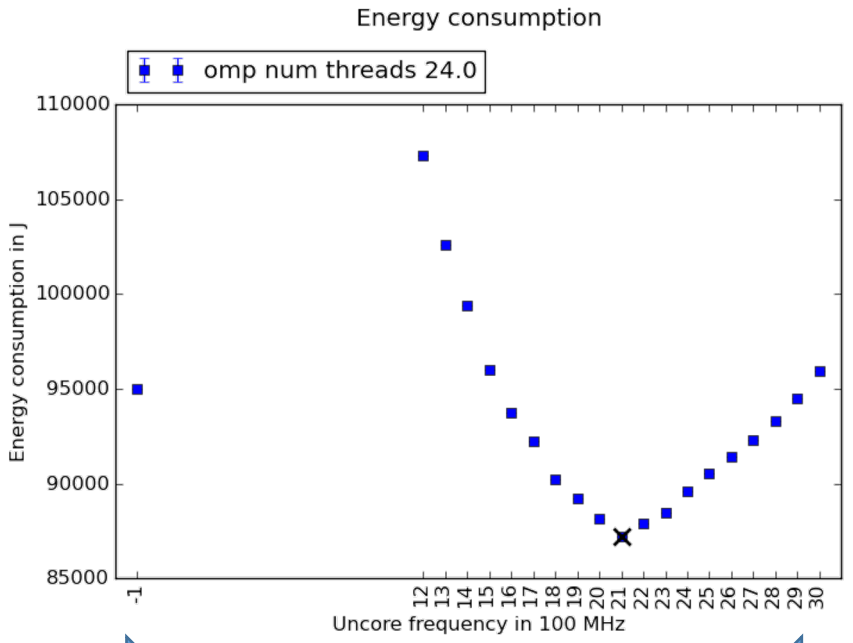
Hardware parameter tuning

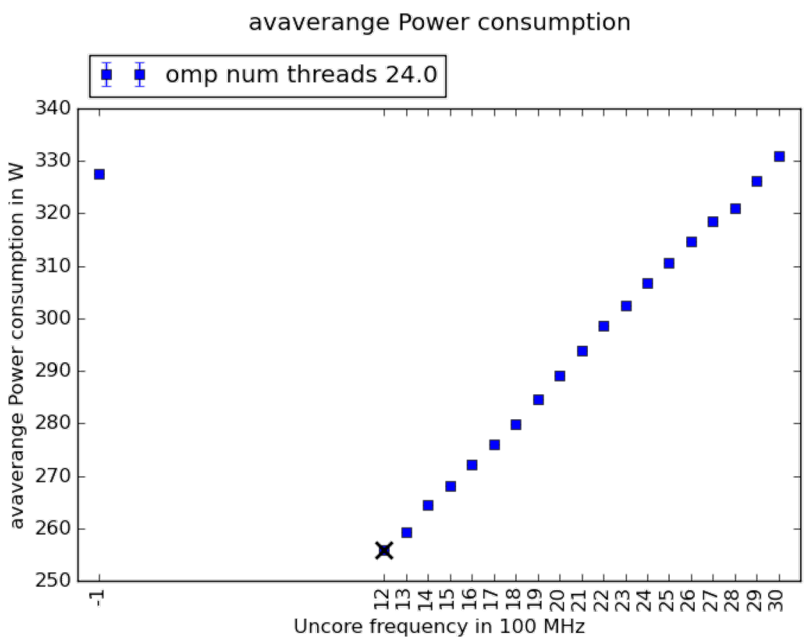
Investigation of impact of **CPU uncore frequency** tuning on memory bound code:

- Optimal frequency, with low energy consumption, and a small performance impact

Evaluation using STREAM Copy benchmark





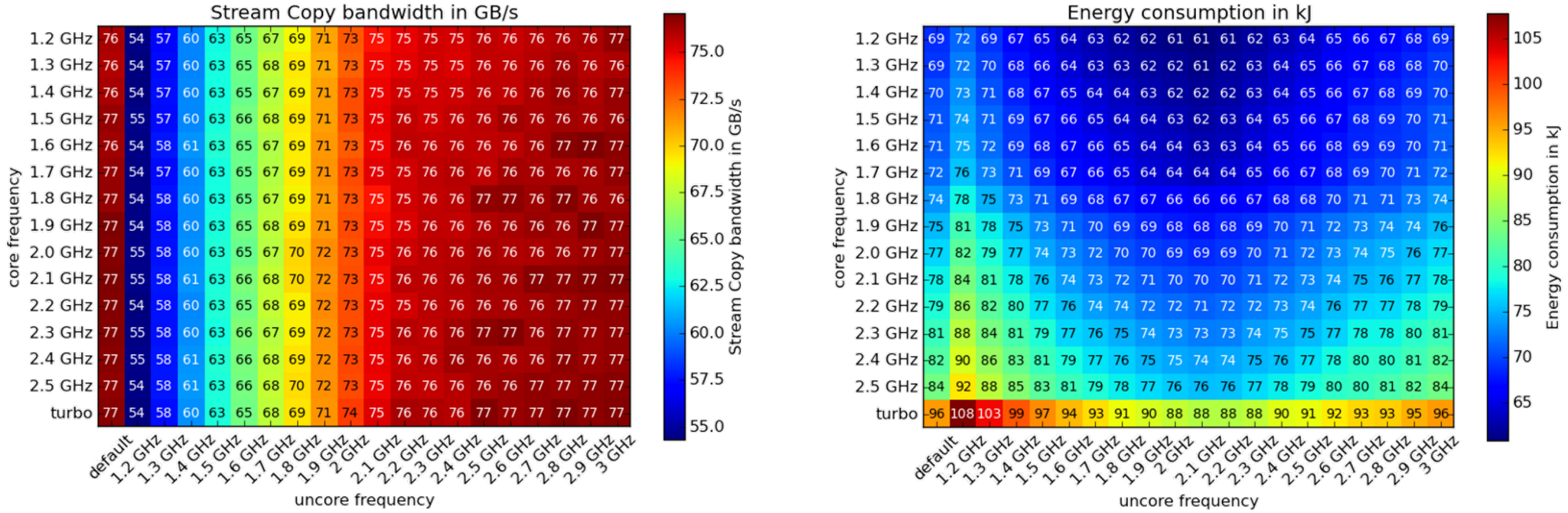


Results by TU Dresden under READEX project

Hardware parameter tuning

Effect of changing **core frequencies** on **uncore performance** using memory bound code

Evaluation using STREAM Copy benchmark



Heatmap of the energy consumption of a stream benchmark for different core and uncore frequencies.

The data array does not fit in the processor's L3 processor cache

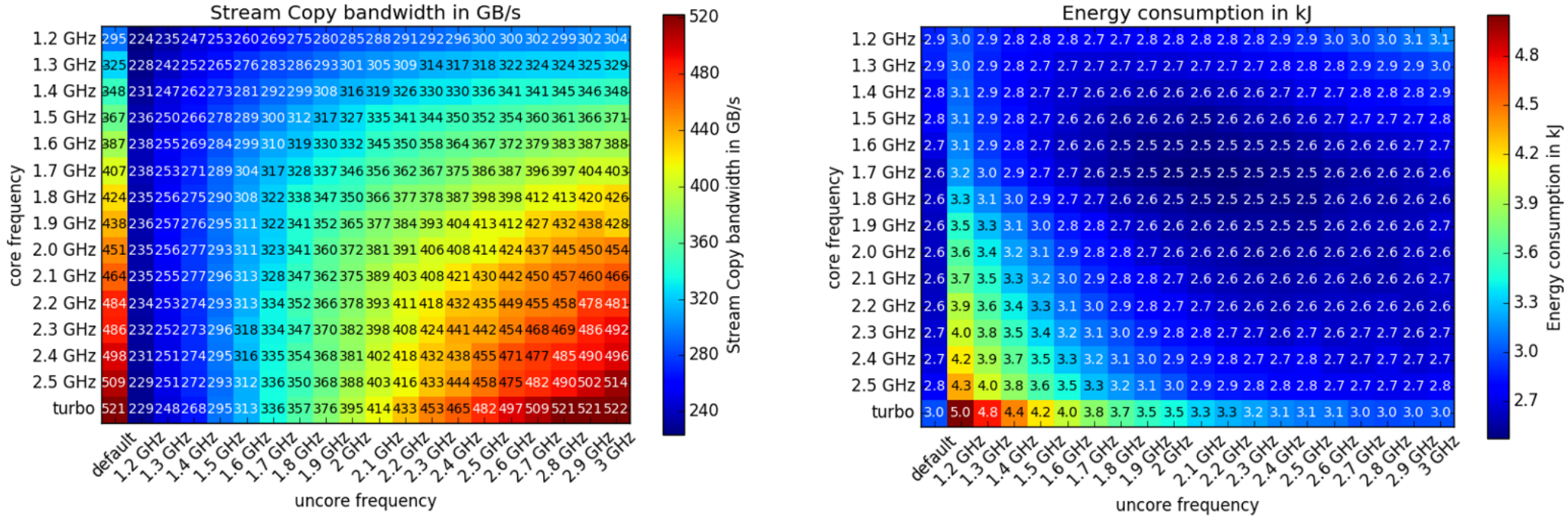
Results by TU Dresden under READEX project

Hardware parameter tuning

L3 Cache Energy efficiency and Bandwidth:

- Different optimal uncore frequencies

Evaluation using STREAM Copy benchmark



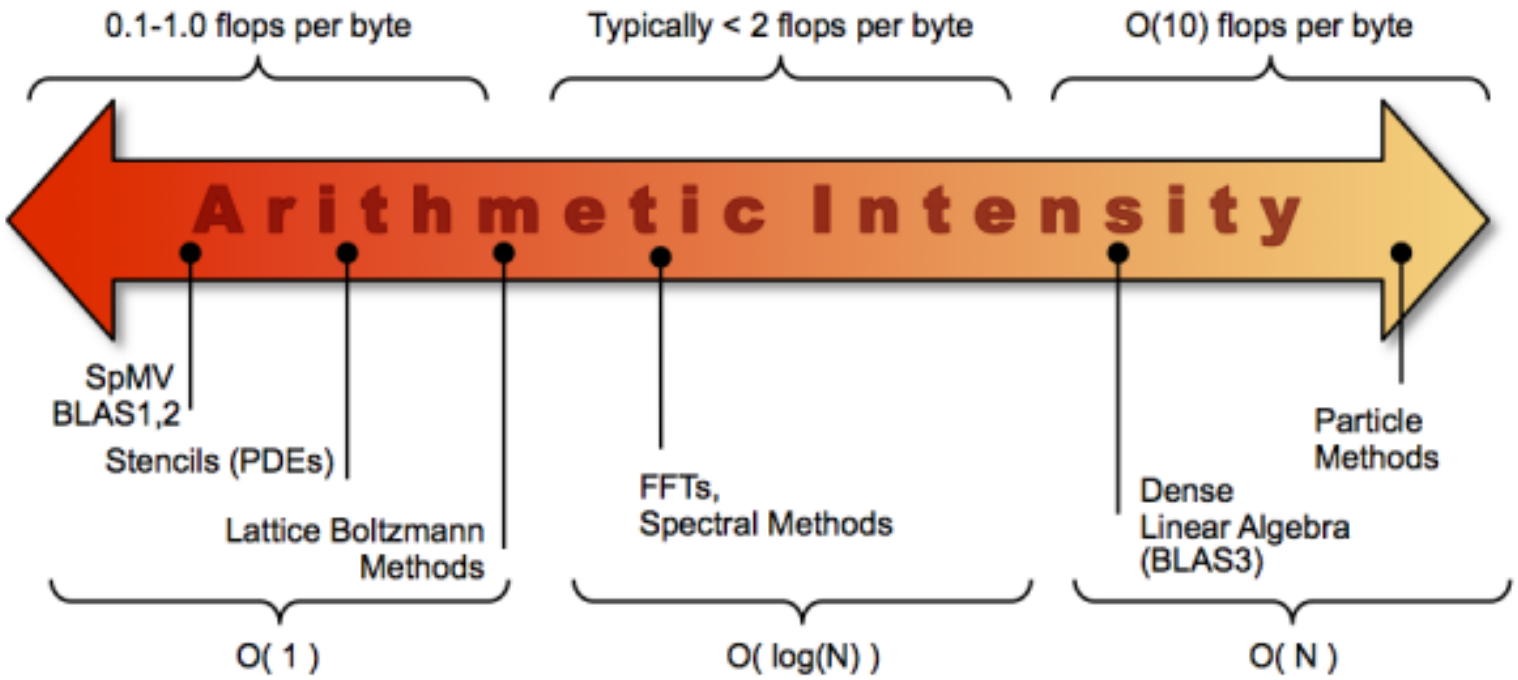
Heatmap of the energy consumption of a stream benchmark for different core and uncore frequencies.

The data array does fit in the processor's L3 processor cache

Results by TU Dresden under READEX project

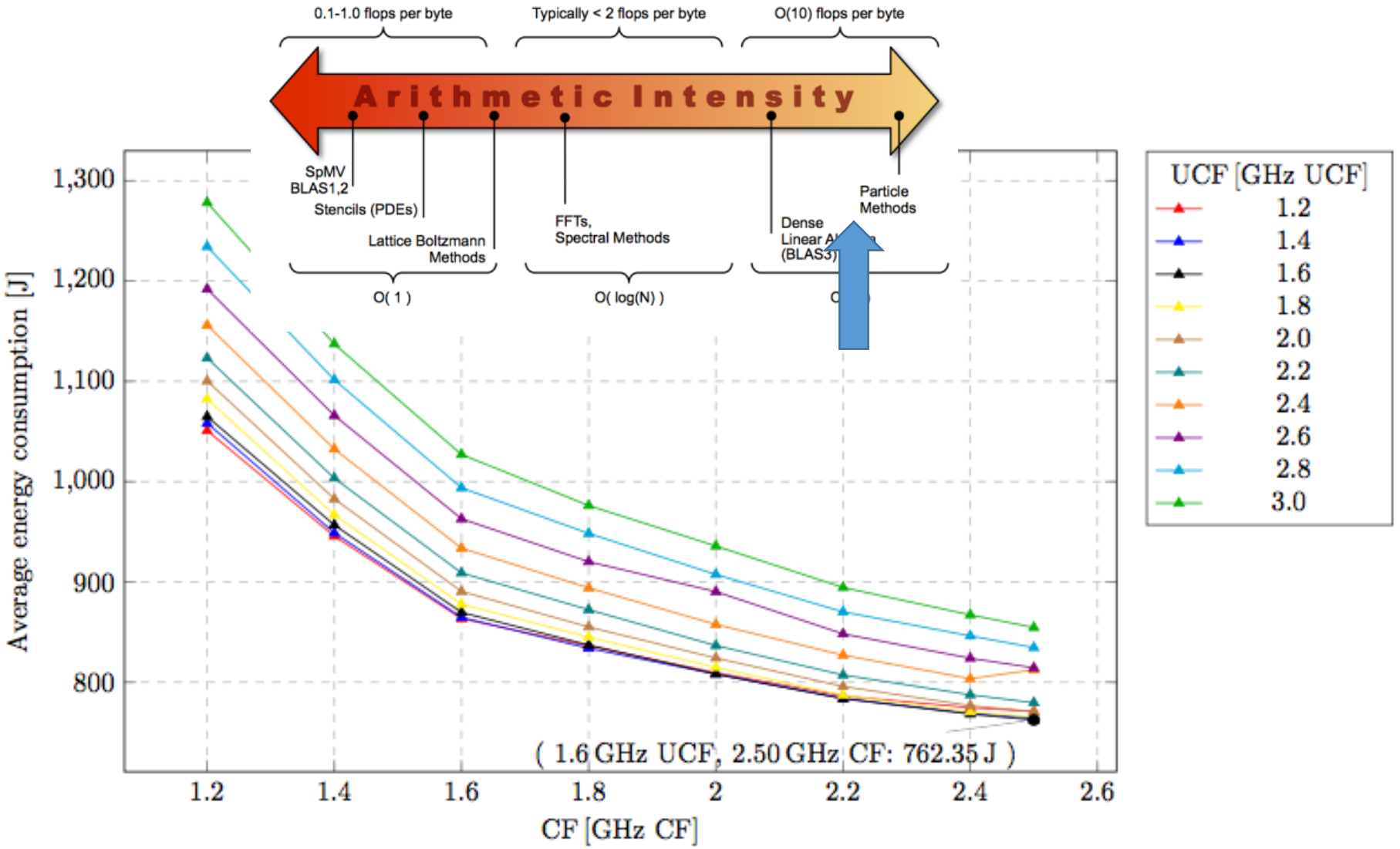
Effect of hardware parameter tuning on kernels with various arithmetic intensity

Arithmetic intensity



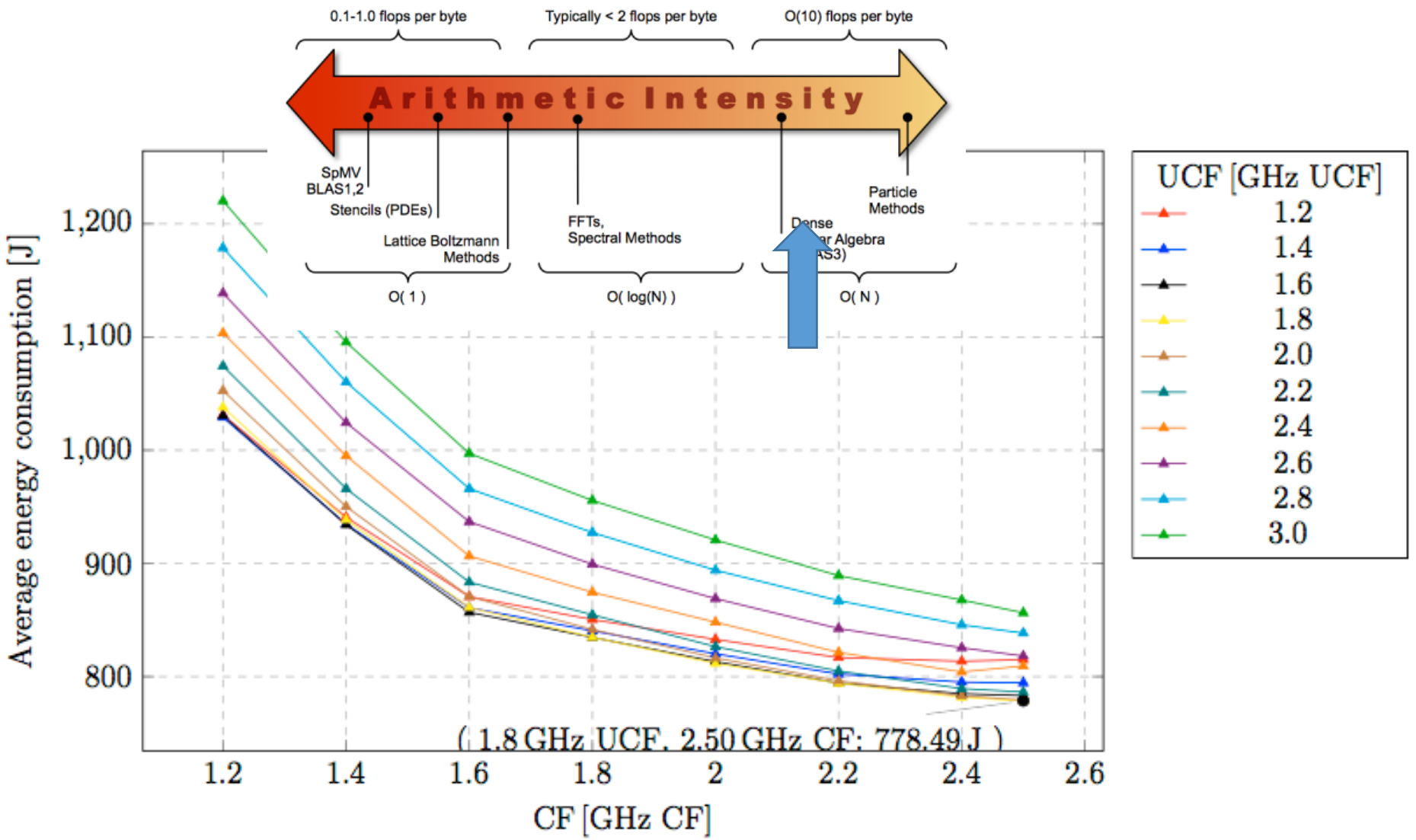
Static tuning for various arithmetic intensity

Ratio from 1:9



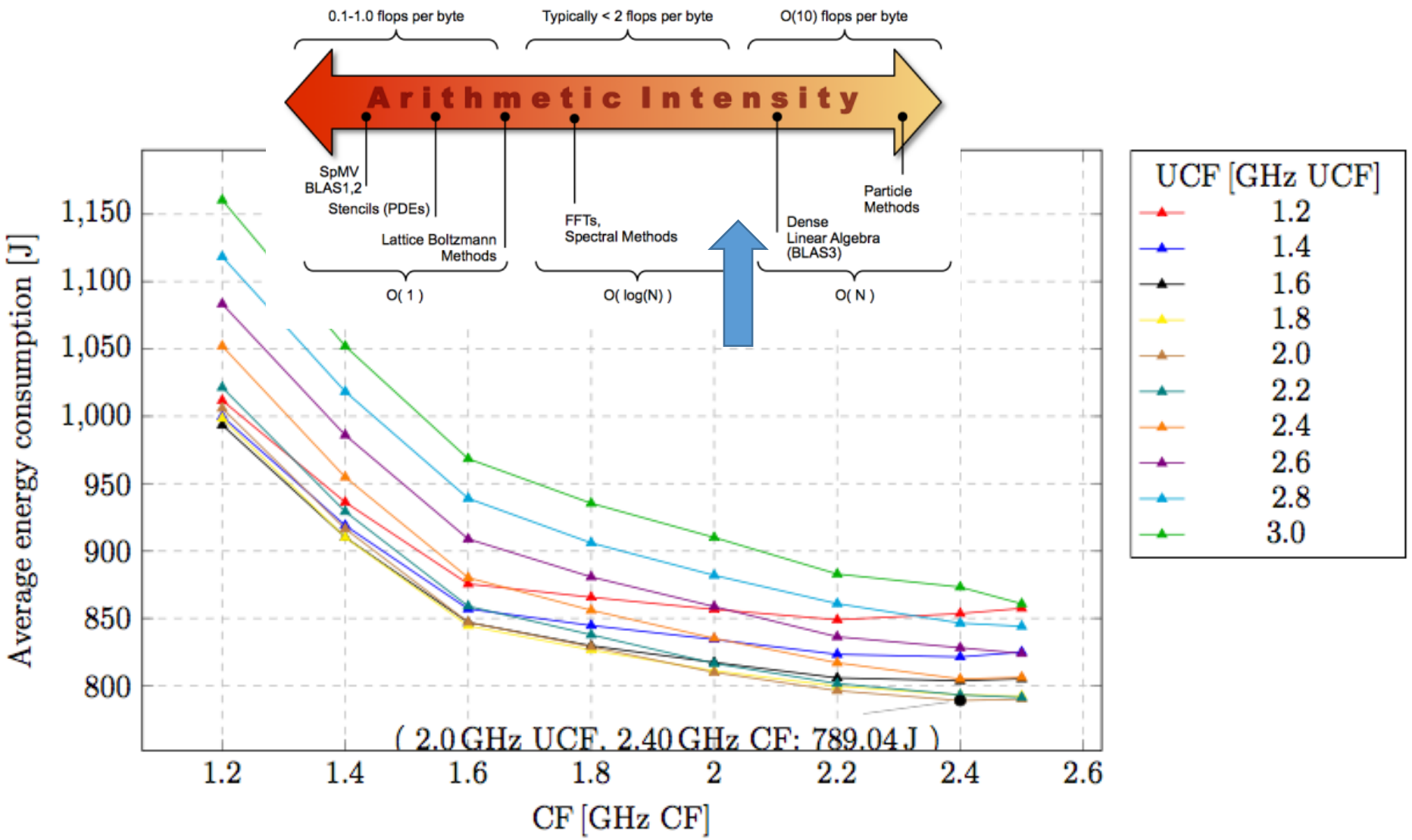
Static tuning for various arithmetic intensity

Ratio from 2:8



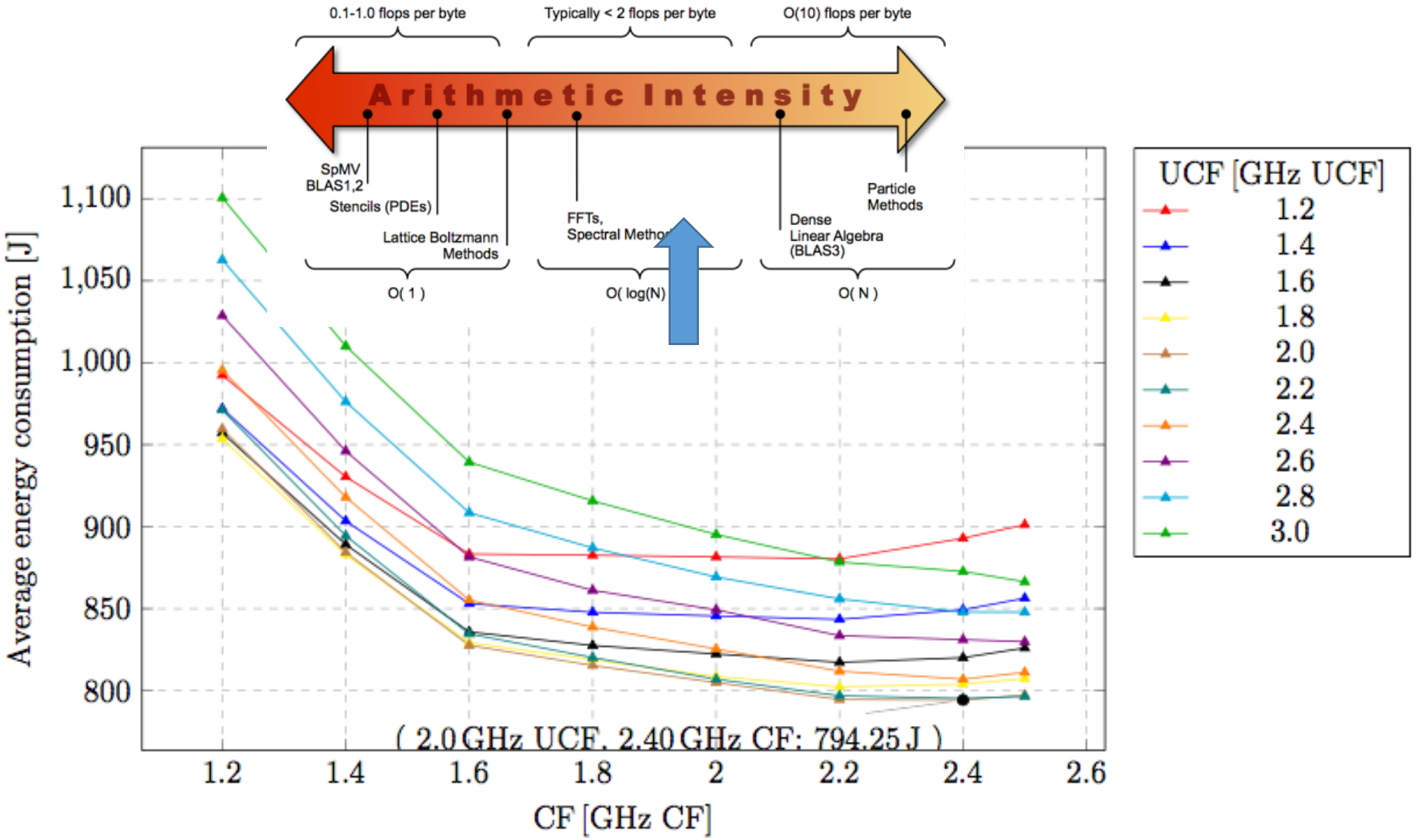
Static tuning for various arithmetic intensity

Ratio from 3:7



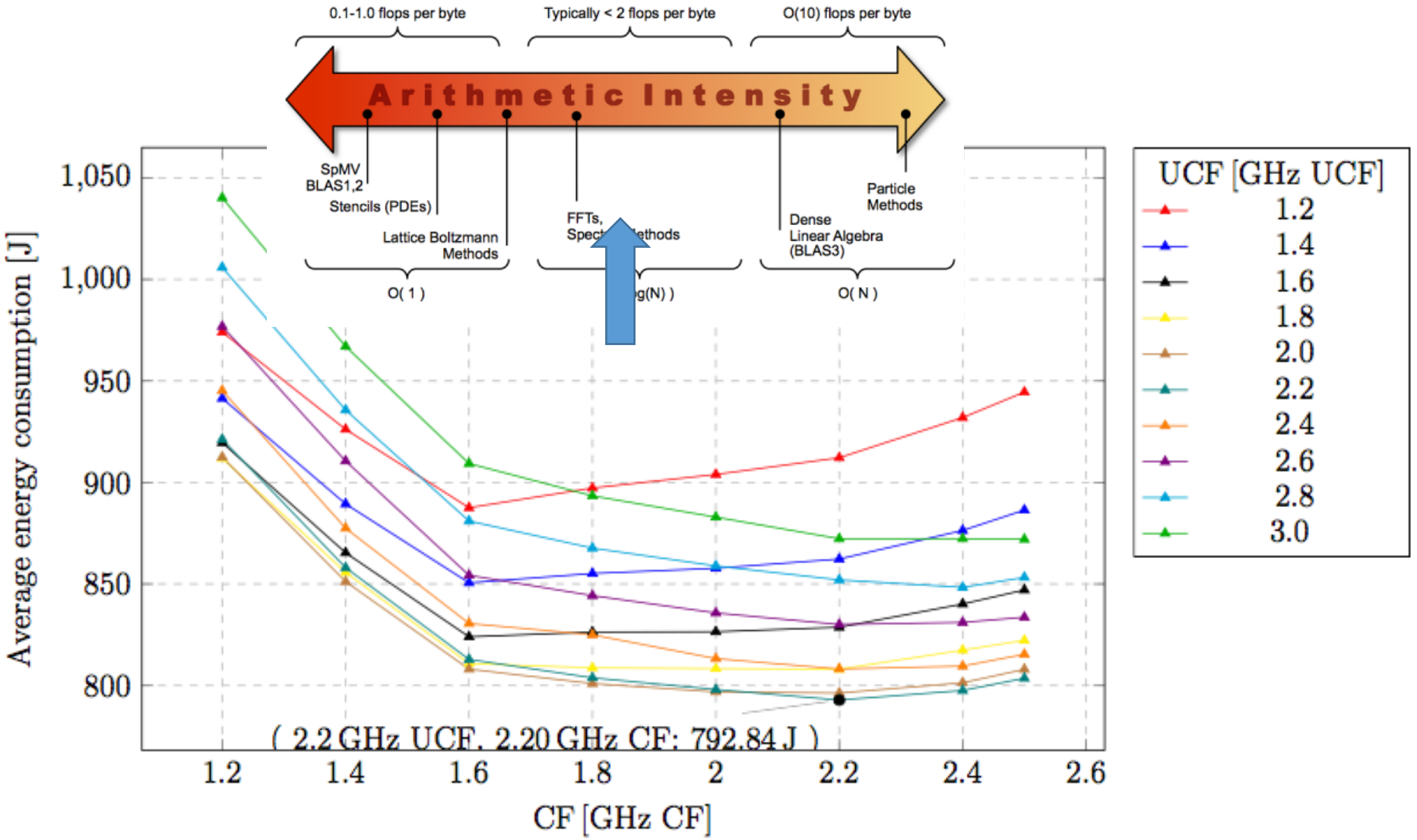
Static tuning for various arithmetic intensity

Ratio from 4:6



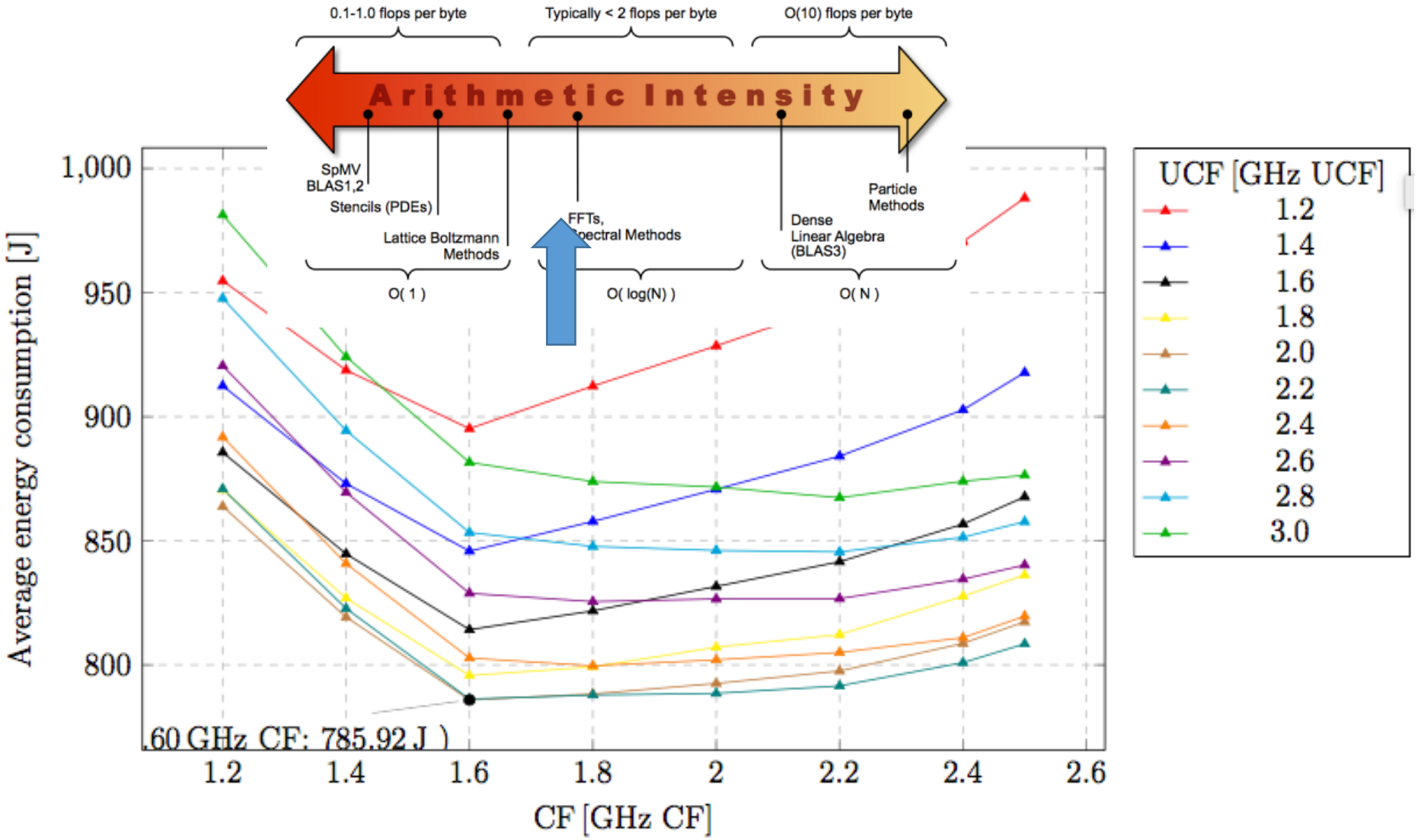
Static tuning for various arithmetic intensity

Ratio from 5:5



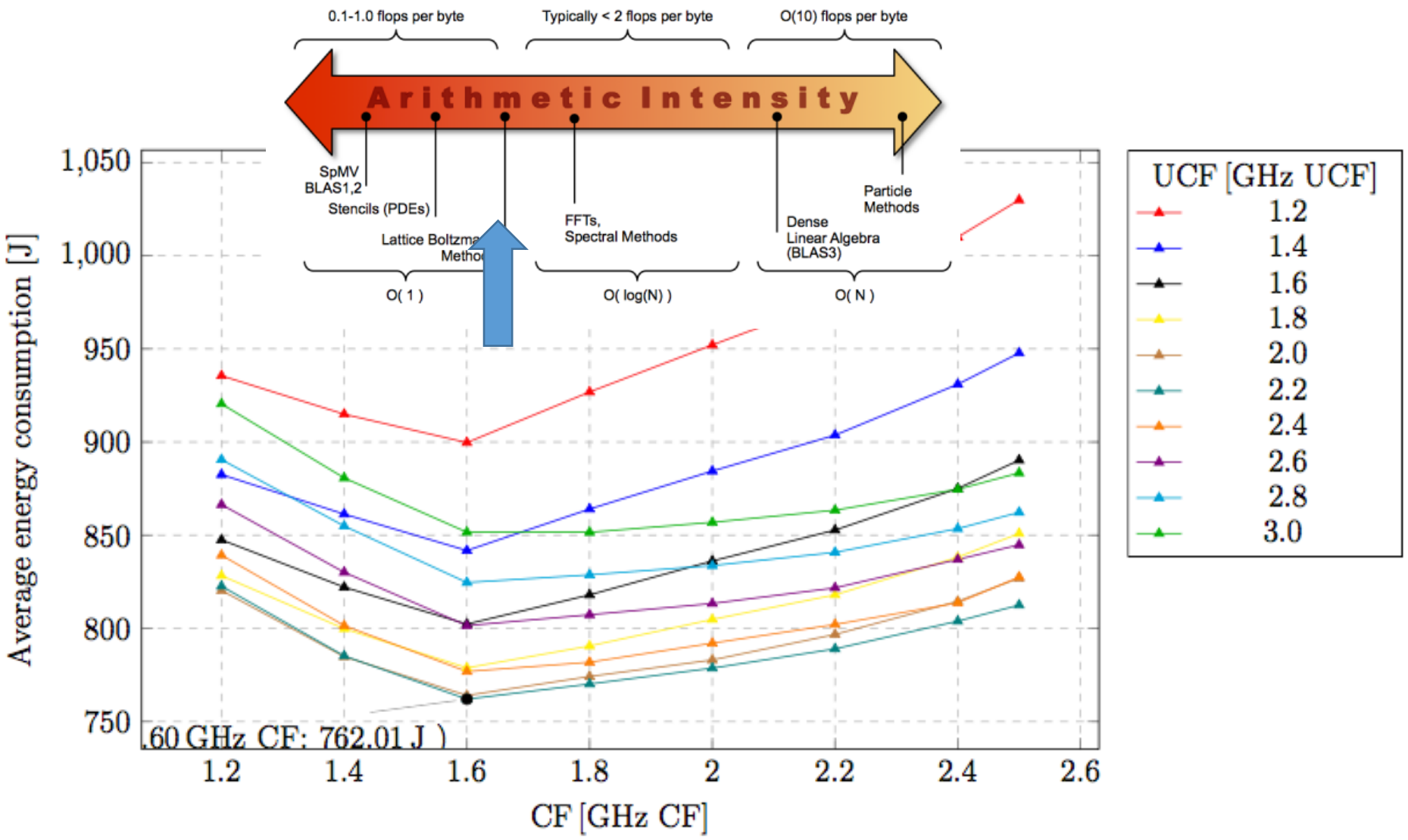
Static tuning for various arithmetic intensity

Ratio from 6:4



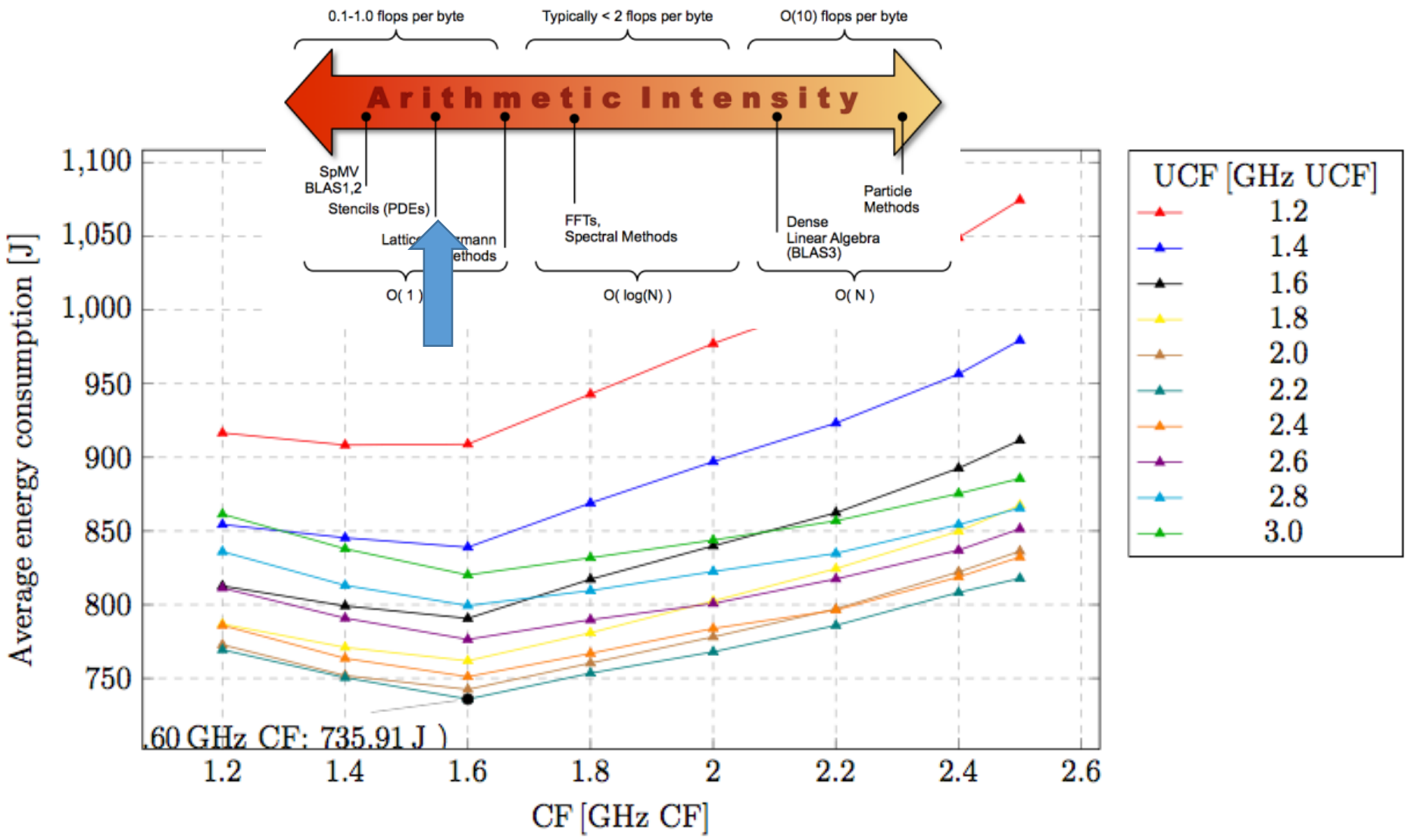
Static tuning for various arithmetic intensity

Ratio from 7:3



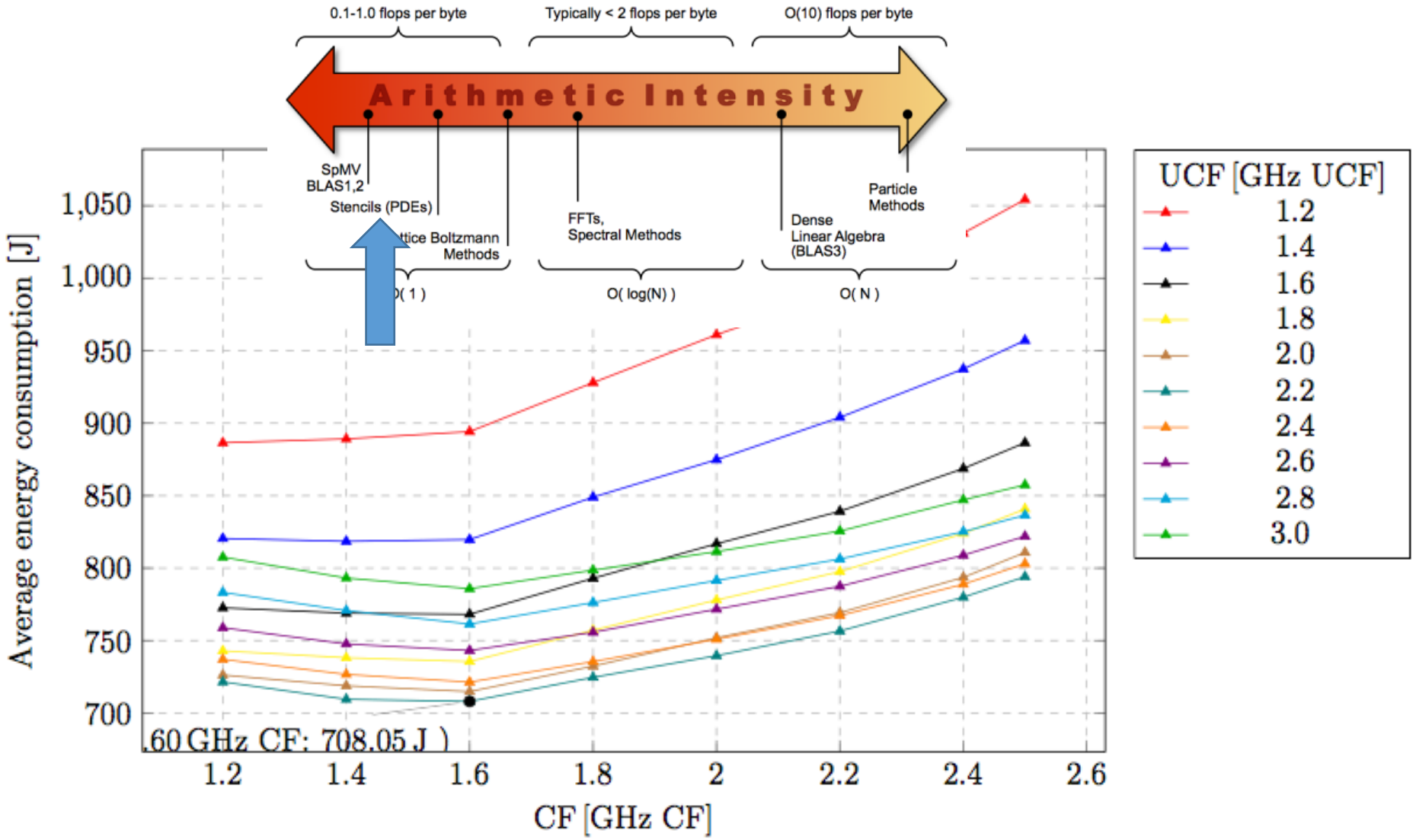
Static tuning for various arithmetic intensity

Ratio from 8:2



Static tuning for various arithmetic intensity

Ratio from 9:1



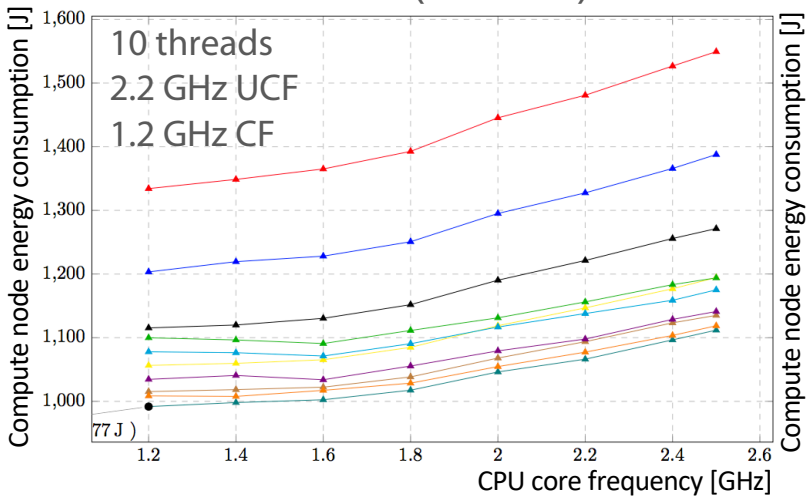
Hardware parameter tuning

Behavior of the simple application with two kernels

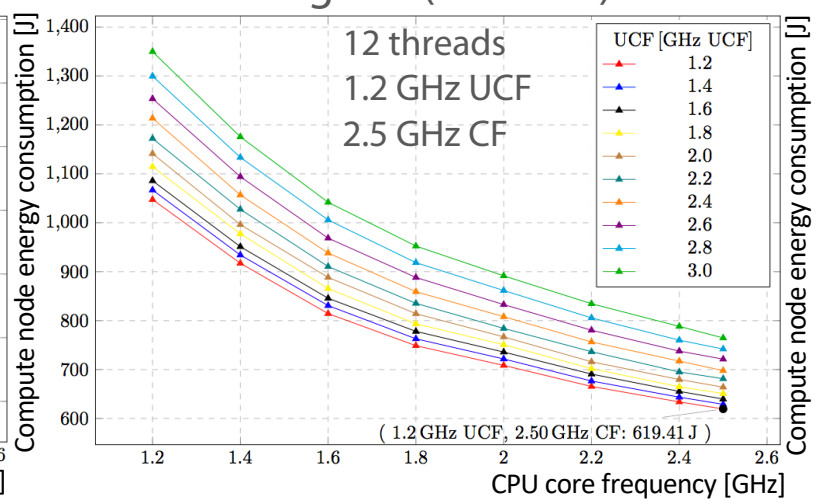
- Low computational intensity – DGEMV
- High computational intensity – DGEMM
- Tuning of three parameters
  - Core frequency
  - Uncore frequency
  - Number of OpenMP threads
- Visualized by RADAR

Two kernels with 1:1 workload ratio	Energy consumption	Energy savings	
Default settings	2017J	-	-
Static optimal	1833J	179J	9%
Dynamic optimal	1612J	221J	12%
Total savings	-	400J	20%

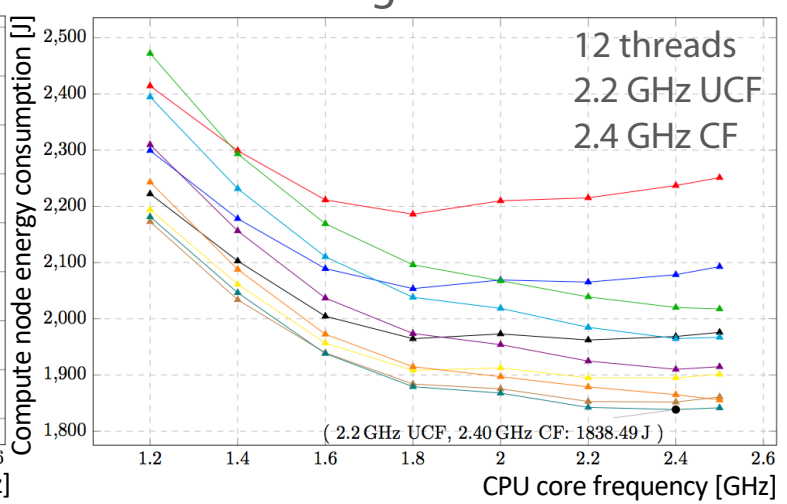
Low CI (DGEMV)



High CI (DGEMM)



Static tuning for both kernels



Note: runtime of both kernels was equal for default settings

Core and uncore frequency tuning under power cap

Experiments description and testbed parameters

Experiment number	Powercapping	DVFS (core freq. tuning)	Uncore freq. tuning	Description
0	-	-	-	default CPU behavior
1	x	-	-	default CPU behavior under powercap
2	-	x	-	default CPU behavior under DVFS tuning
3	-	-	x	default CPU behavior under uncore freq. tuning
4	-	x	x	READEX tuning approach - DVFS & uncore freq.
5	x	x	-	DVFS tuning under powercap; uncore freq. unset
6	x	-	x	uncore freq. under powercap; DVFS unset
7	x	x	x	DVFS and uncore freq. tuning under powercap

A set of experiments performed on Intel Broadwell Architecture

Testbed: Broadwell partition of the Galileo supercomputer in CINECA

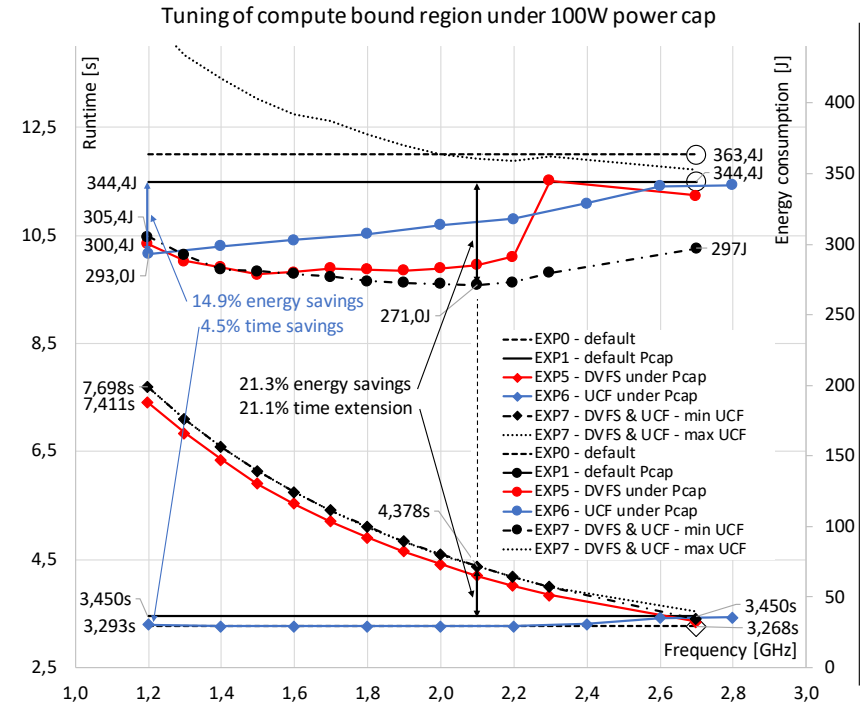
- dual socket server
- two 18-core Intel Xeon E5-2697v4 processor
- 2.3 GHz nominal frequency.
- 2.7 GHz turbo frequency when all 18 cores are utilized
- 145W TDP

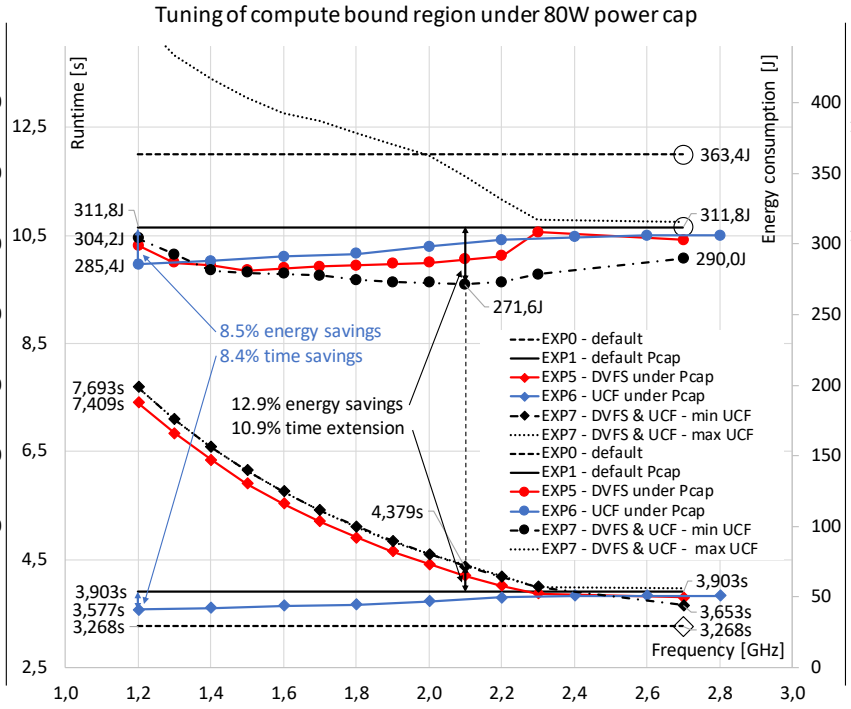
	nominal value	minimal value	maximal value	minimal step
CPU core frequency (DVFS)	2.3 GHz	1.2 GHz	2.8 GHz turbo	100 MHz
CPU uncore frequency	-	1.2 GHz	2.8 GHz	100 MHz
Power capping	145 W[3]	33 W	145 W	0.125 W

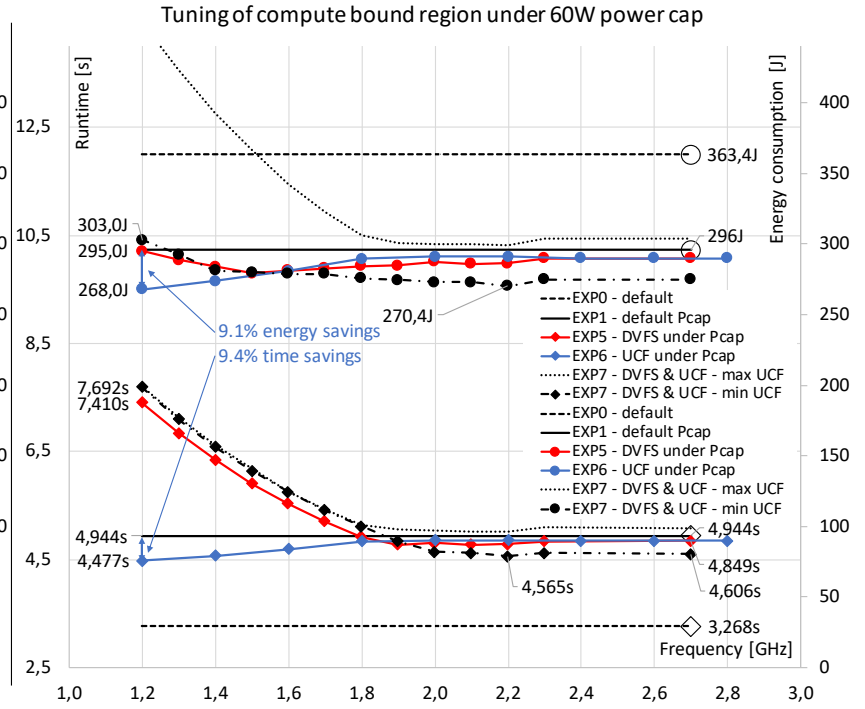
Key tunable parameters of the 18-core Intel Xeon E5-2697v4 processor and their respective ranges and steps.

Tuning of COMPUTE bound workload

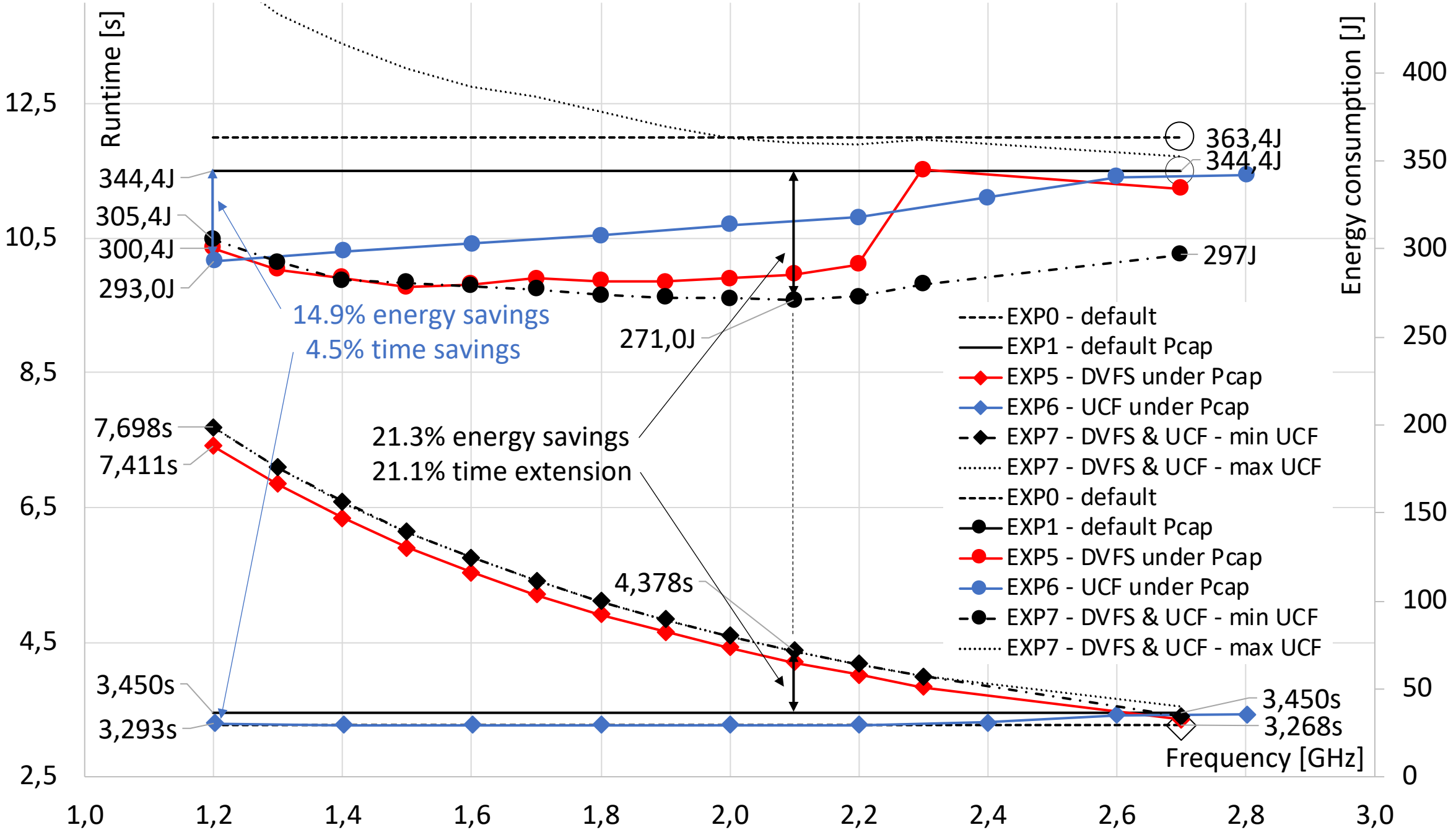
- behavior of the platform when running memory bound workload
  - under 145 W (TDP level, no power cap)
  - three different power cap levels 100 W, 80 W and 60 W.



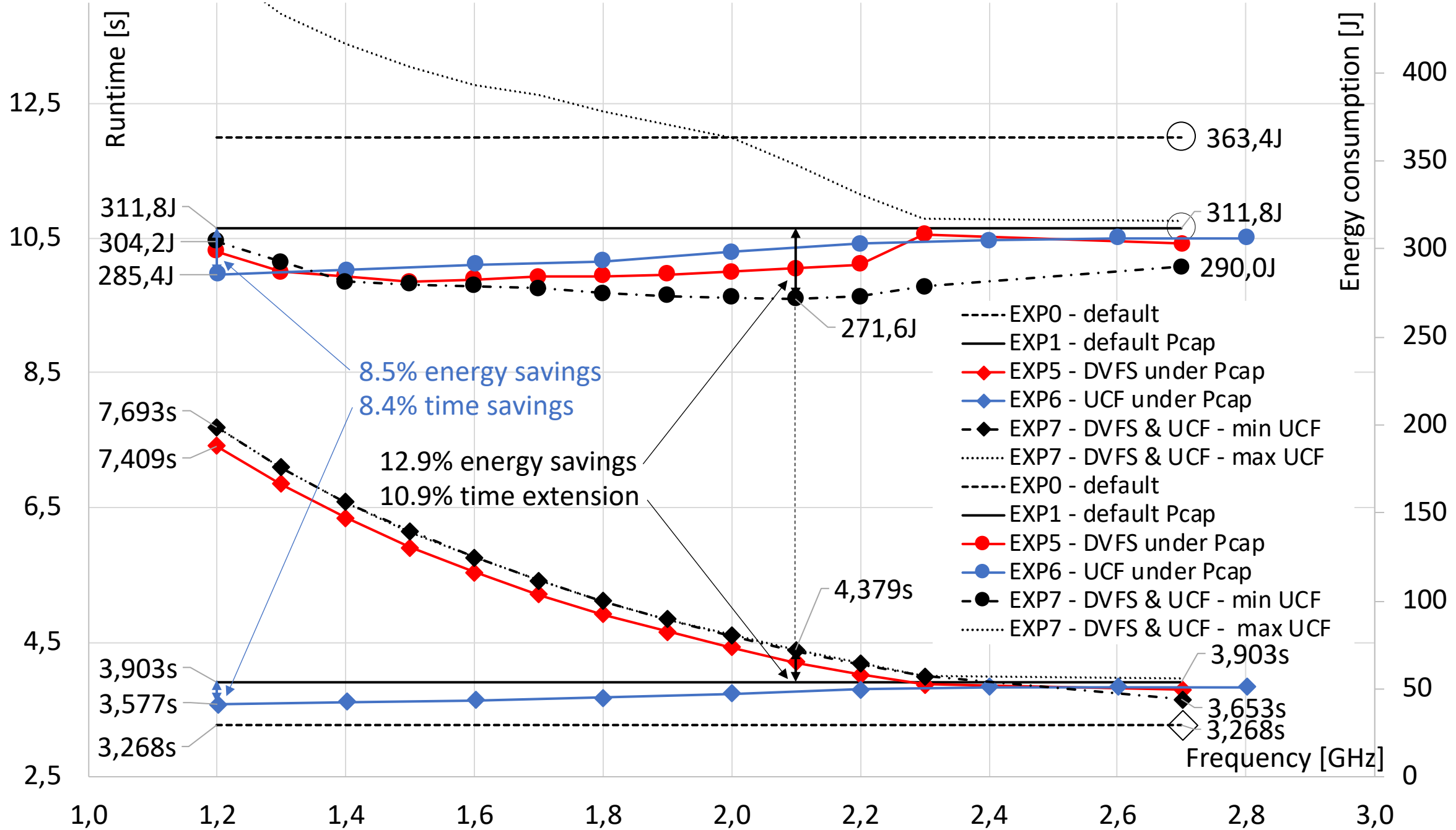




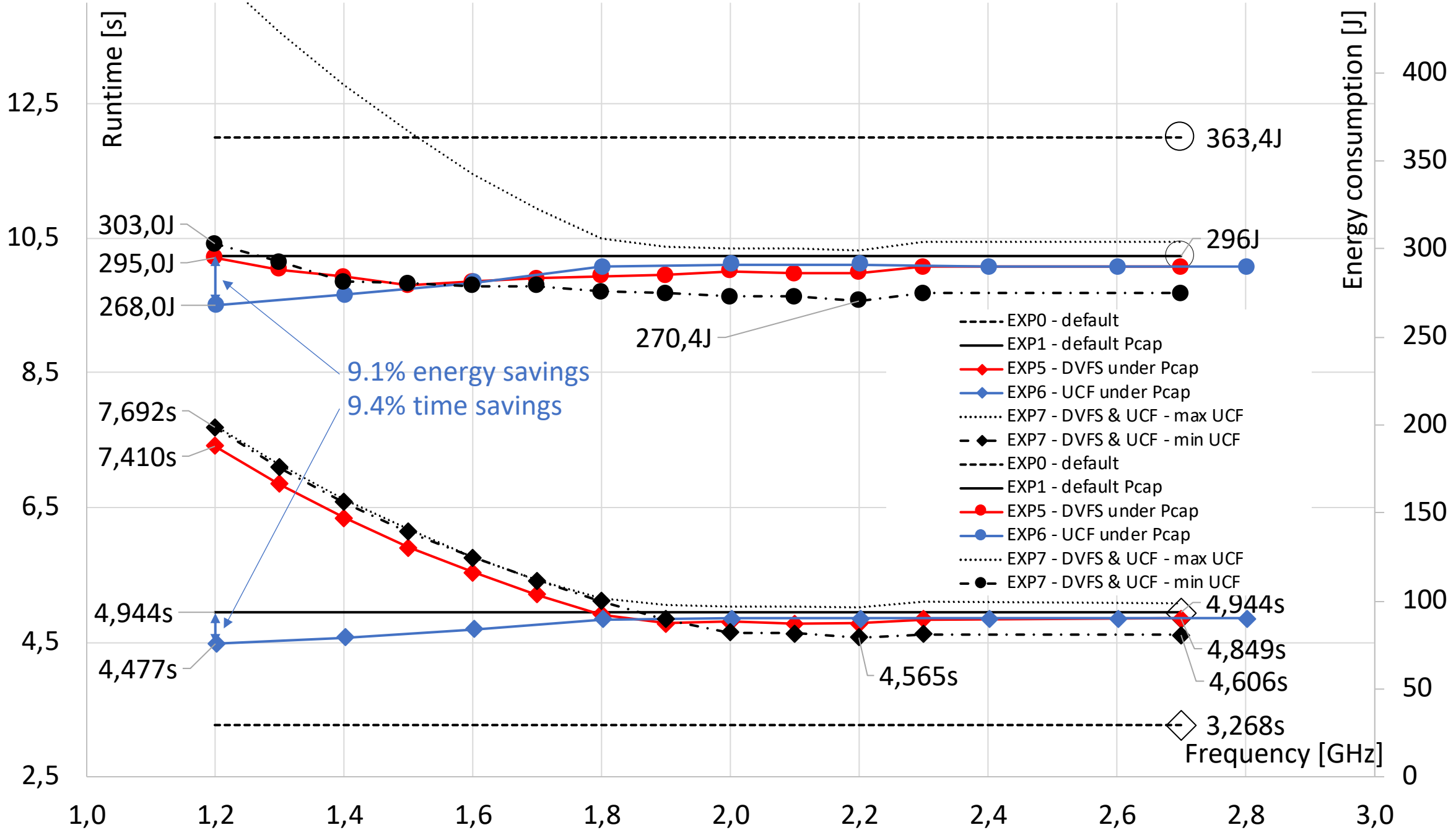
Tuning of COMPUTE bound workload under 100W power cap



Tuning of COMPUTE bound workload under 80W power cap



Tuning of COMPUTE bound workload under 60W power cap



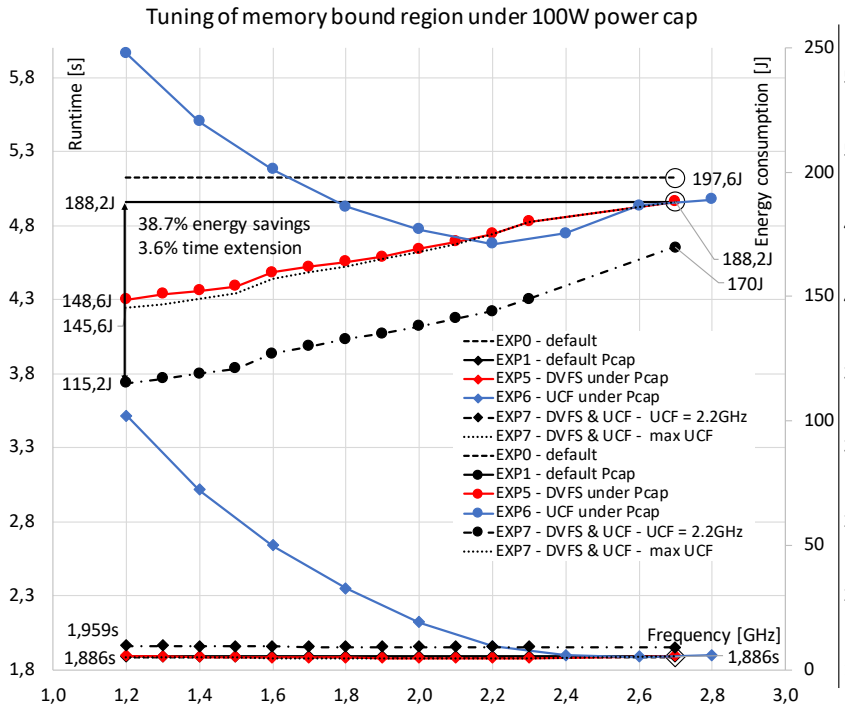
Observations for COMPUTE bound workload

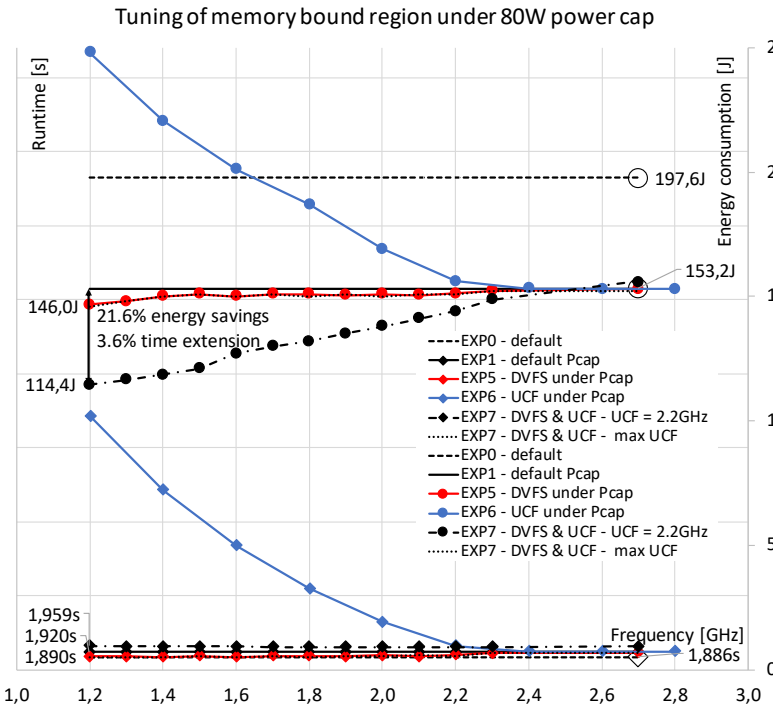
- To achieve the best possible performance
  - **the uncore frequency must be reduces to minimum**
    - **9.4 % performance gain up to and**
    - **14.9 % lower energy consumption**

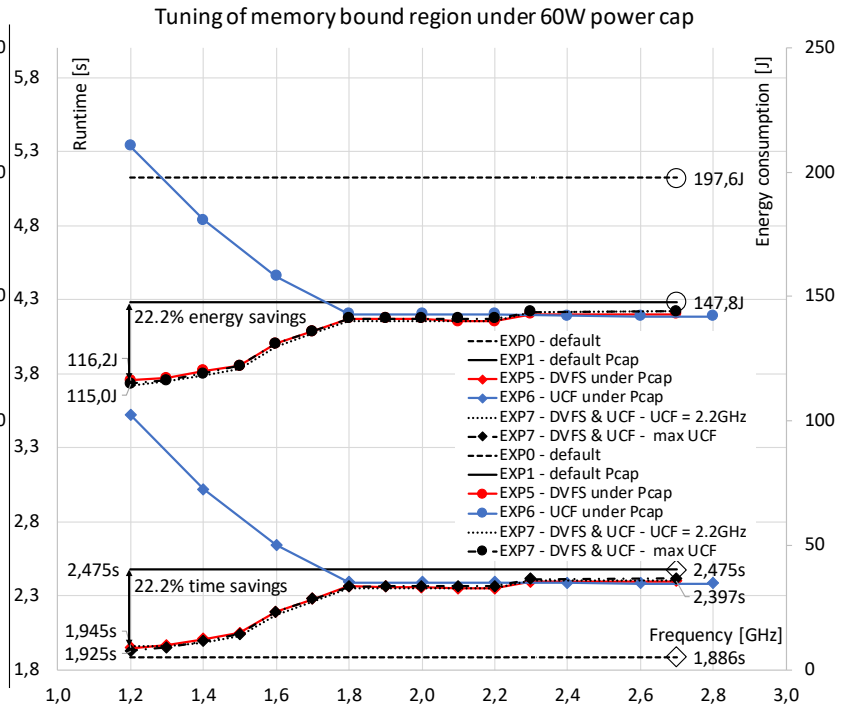
- If further energy savings are required – use DVFS and lower the core freq.
  - up to 21 % of energy savings
  - up to 21 % penalty in runtime
  - this effect is more visible for higher powercap levels

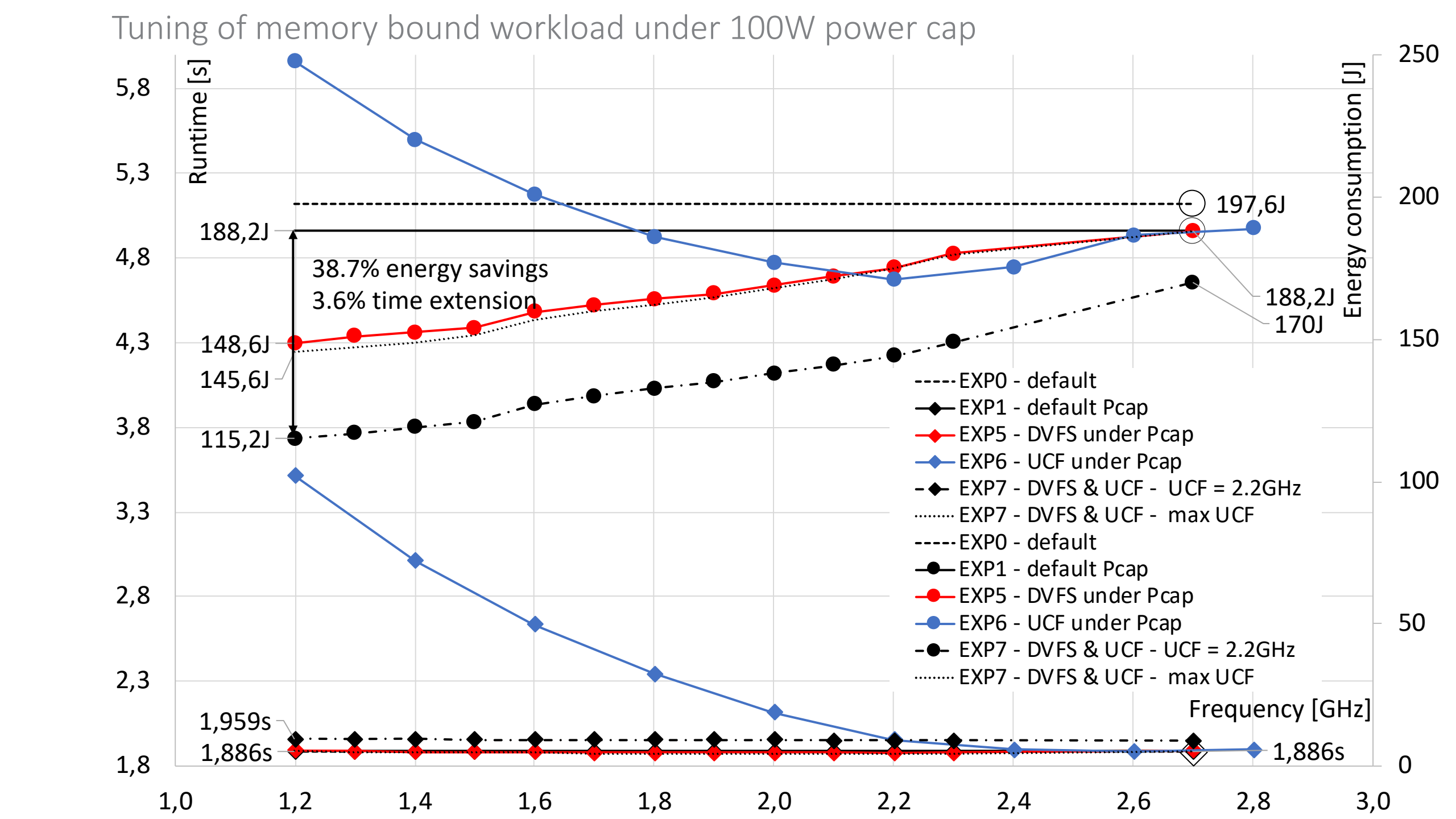
Tuning of memory bound workload

- behavior of the platform when running memory bound workload
  - under 145 W (TDP level, no power cap)
  - three different power cap levels 100 W, 80 W and 60 W.

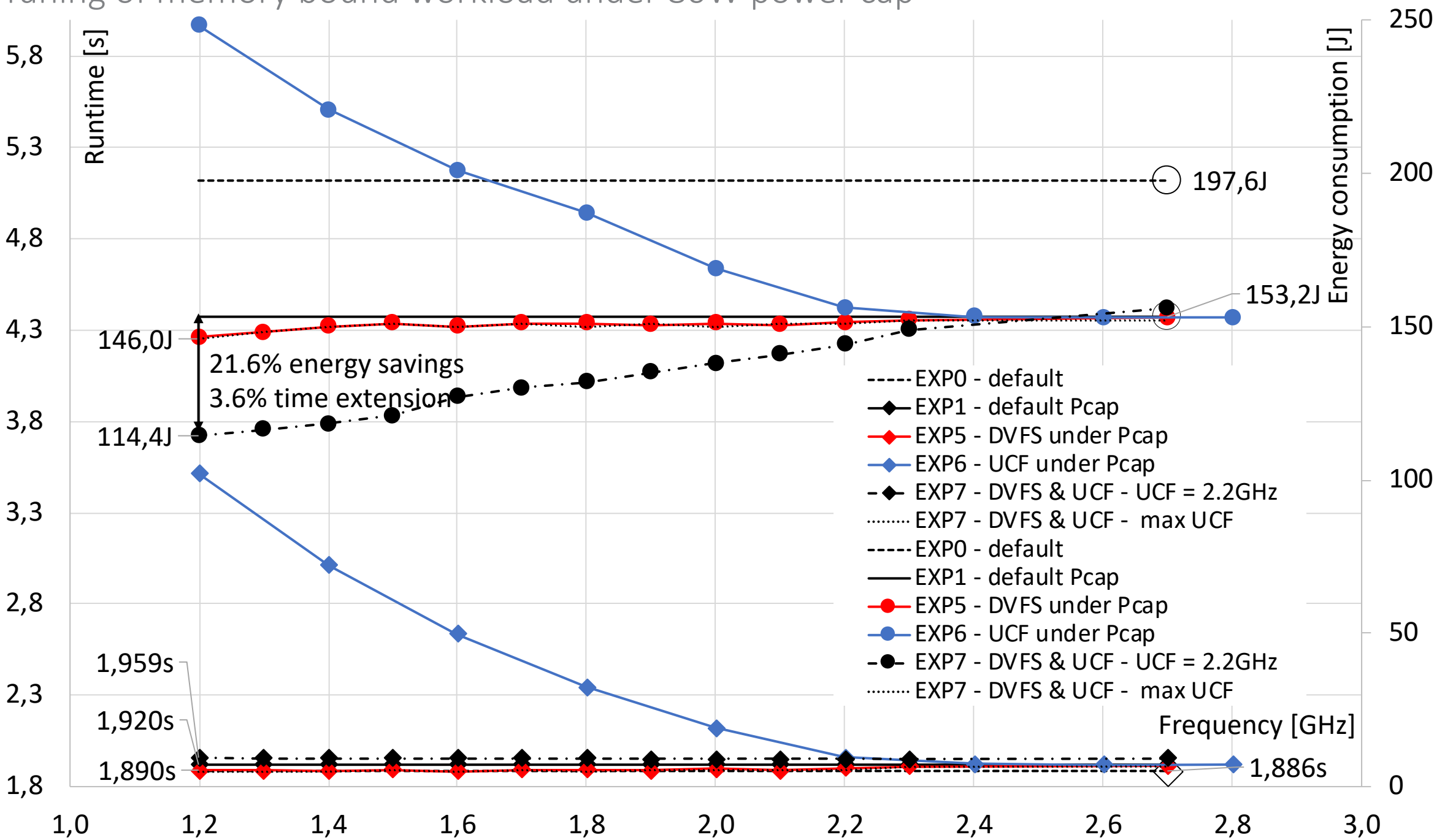




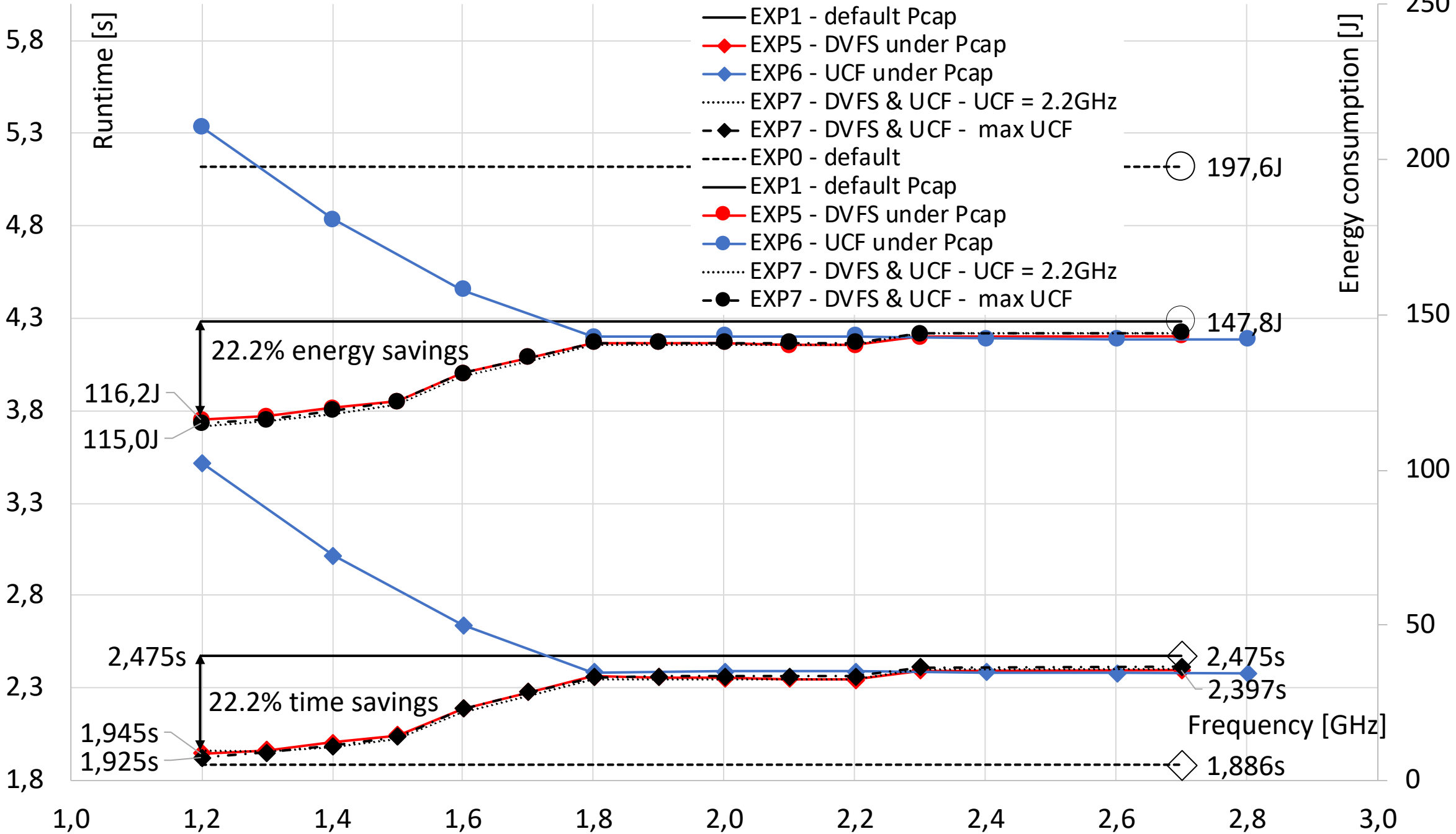



Tuning of memory bound workload under 100W power cap
Runtime [s]
Energy consumption [J]
Frequency [GHz]
5,8
5,3
4,8
4,3
3,8
3,3
2,8
2,3
1,8
250
200
150
100
50
0
1,0
1,2
1,4
1,6
1,8
2,0
2,2
2,4
2,6
2,8
3,0
197,6J
188,2J
38.7% energy savings
3.6% time extension
148,6J
145,6J
115,2J
188,2J
170J
1,959s
1,886s
1,886s
EXP0 - default
EXP1 - default Pcap
EXP5 - DVFS under Pcap
EXP6 - UCF under Pcap
EXP7 - DVFS & UCF - UCF = 2.2GHz
EXP7 - DVFS & UCF - max UCF
EXP0 - default
EXP1 - default Pcap
EXP5 - DVFS under Pcap
EXP6 - UCF under Pcap
EXP7 - DVFS & UCF - UCF = 2.2GHz
EXP7 - DVFS & UCF - max UCF

Tuning of memory bound workload under 80W power cap



Tuning of memory bound workload under 60W power cap



Observations for both workloads

Observations for memory bound workload

- Under the power budget lower that 80 W
  - **DVFS should be set to minimum value**
    - **boost the performance of the uncore part by 22%.**
- Tuning of the uncore frequency
  - **has low effect on the performance**
  - **but a major effect on energy consumption**
    - between 21% (60 W and 80W) to 38% (100W)

Observations for compute bound workload

- To achieve the best possible performance
  - **the uncore frequency must be reduces to minimum**
    - **9.4 % performance gain up to and**
    - **14.9 % lower energy consumption**
- If further energy savings are required – use DVFS and lower the core freq.
  - up to 21 % of energy savings
  - up to 21 % penalty in runtime
  - this effect is more visible for higher powercap levels

Evaluation of complex HPC applications

BEM4I Application

Application runtime	assemble_k [s]	assemble_v [s]	gmres_solve [s]	print_vtu [s]	main [s]
default runtime	5.4	5.9	10.2	5.6	27.3
static tuning runtime	9.8	10.6	6.1	2.4	29.0
dynamic tuning runtime	7.0	7.2	7.9	2.1	24.3
static savings [%]	-82.3%	-79.1%	40.5%	56.8%	-6.2%
dynamic savings [%]	-30.6%	-20.9%	23.2%	62.9%	10.9%

```
"static": {
    "FREQUENCY": "25",          <--------- 2.5 GHz
    "NUM_THREADS": "12",        <--------- 12 OpenMP threads
    "UNCORE_FREQUENCY": "22" }  <--------- 2.2 GHz
```

```
"assemble_k": {
   "FREQUENCY": "23",
   "NUM_THREADS": "24",
   "UNCORE_FREQUENCY": "16"
},

"assemble_v": {
   "FREQUENCY": "25",
   "NUM_THREADS": "24",
   "UNCORE_FREQUENCY": "14"
},

"gmres_solve": {
   "FREQUENCY": "17",
   "NUM_THREADS": "8",
   "UNCORE_FREQUENCY": "22"
},

"print_vtu": {
   "FREQUENCY": "25",
   "NUM_THREADS": "6",
   "UNCORE_FREQUENCY": "24"
}
```

Hardware: dual socket system with 2x12 CPU cores – "*standard HW*" in HPC centres

Region description:

- **assemble_k** and **assemble_v** – high utilization of vector units, extreme level of optimization – fully compute bound great utilization of both sockets and all cores
- **gmres_solve** – uses DGEMV from MKL – memory bound, suffers on NUMA effect; this routine is more efficient on single socket
- **print_vtu** – single threaded I/O and network bound region why stores data to a file on LUSTRE system

BEM4I Application

Compute node energy	assemble_k [J]	assemble_v [J]	gmres_solve [J]	print_vtu [J]	main [J]
default energy	1476	1484	2733	1142	6872
static tuning energy	1962	2015	1366	420	5792
dynamic tuning energy	1467	1462	1259	293	4531
static savings [%]	-33.8%	-35.8%	50.0%	63.2%	15.7%
dynamic savings [%]	0.6%	1.5%	53.9%	74.3%	34.1%

```
"static": {
    "FREQUENCY": "25",          <--------- 2.5 GHz
    "NUM_THREADS": "12",        <--------- 12 OpenMP threads
    "UNCORE_FREQUENCY": "22" }  <--------- 2.2 GHz
```

```
"assemble_k": {
   "FREQUENCY": "23",
   "NUM_THREADS": "24",
   "UNCORE_FREQUENCY": "16"
},

"assemble_v": {
   "FREQUENCY": "25",
   "NUM_THREADS": "24",
   "UNCORE_FREQUENCY": "14"
},

"gmres_solve": {
   "FREQUENCY": "17",
   "NUM_THREADS": "8",
   "UNCORE_FREQUENCY": "22"
},

"print_vtu": {
   "FREQUENCY": "25",
   "NUM_THREADS": "6",
   "UNCORE_FREQUENCY": "24"
}
```

Large energy savings is combination of optimal HW settings and runtime savings due to mitigation of NUMA effect by optimal settings of OpenMP threading

- Without savings in runtime caused by similar application will
  - Energy savings approx. 15 – 20%
  - Runtime savings approx. -15%

OpenFOAM Application

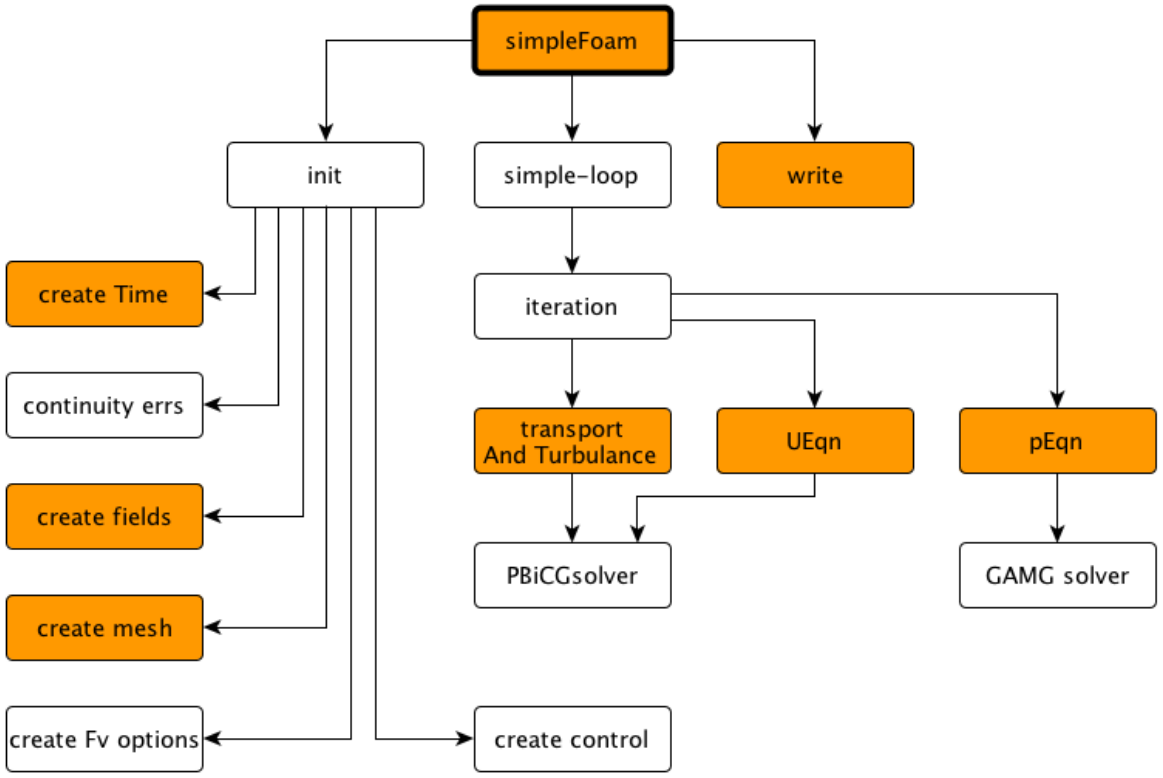
- Computational fluid dynamics
- Finite volume + multigrid solver

OpenFOAM	Energy consumption	Energy savings	
Default settings	14 231J	-	-
Static tuning	12 264J	2 264J	15.9%
Dynamic tuning			
Total savings			

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

OpenFOAM Application

- Computational fluid dynamics
- Finite volume + multigrid solver



OpenFOAM	Energy consumption	Energy savings	
Default settings	14 231J	-	-
Static tuning	12 264J	2 264J	15.9%
Dynamic tuning	11 370J	597J	4.8%
Total savings		2 861J	20.1%

Region	% of 1 phase	Best dynamic configuration	Value	Dynamic savings
init-createTime	0.03	1.4 GHz UCF, 1.4 GHz CF	2.64 J	0.71 J (21.06%)
init-createFields	4.28	2.4 GHz UCF, 2.0 GHz CF	474.80 J	32.11 J (6.33%)
init-createMesh	2.26	1.4 GHz UCF, 1.4 GHz CF	194.38 J	72.96 J (27.29%)
UEqn	40.71	2.2 GHz UCF, 1.6 GHz CF	4810.03 J	10.80 J (0.22%)
pEqn	19.15	2.0 GHz UCF, 1.6 GHz CF	2268.19 J	0.00 J (0.00%)
transportAndTurbulence	25.70	2.0 GHz UCF, 1.6 GHz CF	3042.91 J	0.00 J (0.00%)
write	7.88	1.2 GHz UCF, 1.4 GHz CF	841.62 J	90.97 J (9.75%)

ESPRESO Application

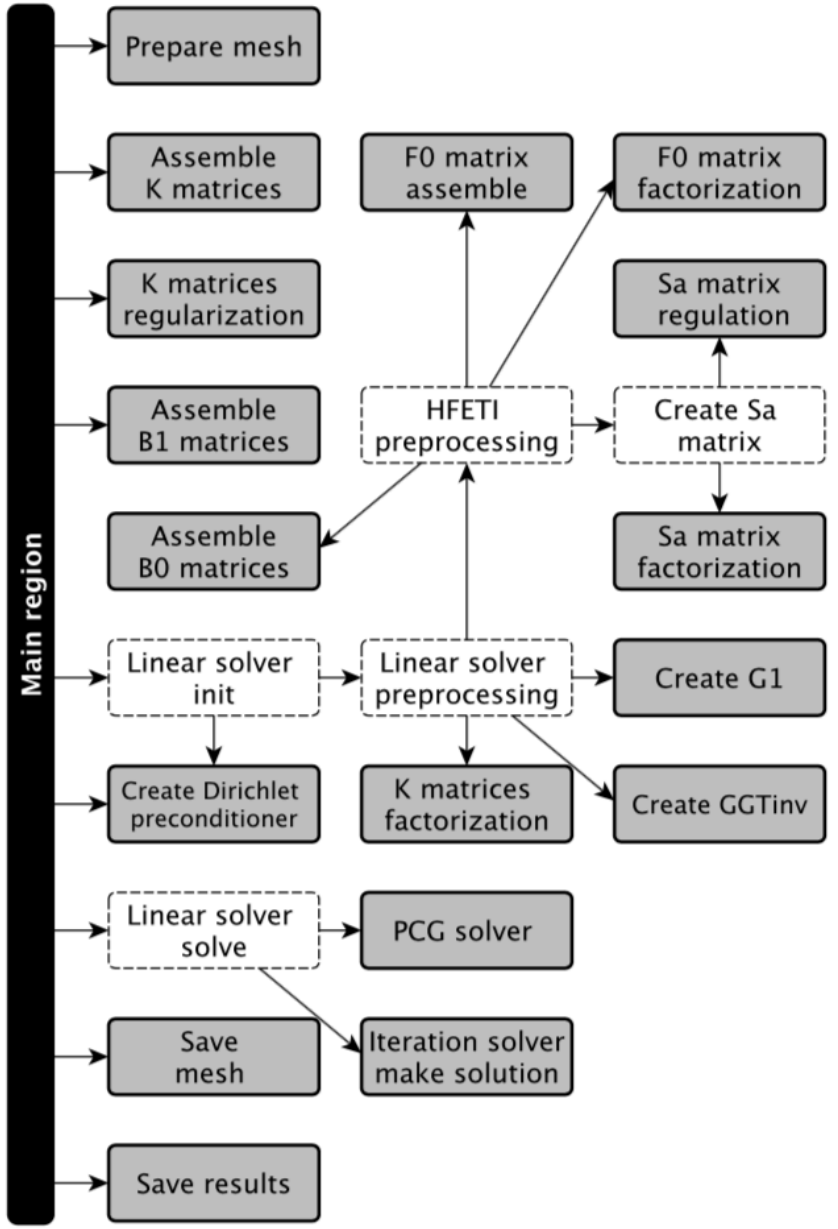
33% of energy savings

22% of time savings and improved strong scalability

- Structural mechanics code
- Finite element + sparse FETI solver
- Different tuning models for different # of nodes is needed for strong scalability – workload per node is varies
- Includes dynamic switching overheads

	no tuning		dynamic tuning		total savings	
number of nodes	time [s]	energy [kJ]	time [s]	energy [kJ]	time [%]	energy [%]
1	129.3	37.2	143.7	34.3	-11.1	**8.0**
2	68.6	39.8	75.5	36.5	-10.1	**8.2**
4	33.2	38.0	35.6	34.3	-7.2	**9.8**
8	21.5	49.6	22.9	44.7	-6.8	**9.9**
16	13.4	60.8	14.3	53.5	-6.3	**12.1**
32	7.7	62.2	7.2	50.6	6.1	**18.7**
64	4.0	69.9	3.6	52.4	9.3	**25.0**
128	3.6	119.6	2.8	80.1	22.2	**33.0**

Energy savings analysis for the strong scalability test of the ESPRESO library when running the cube benchmark



Application parameters tuning

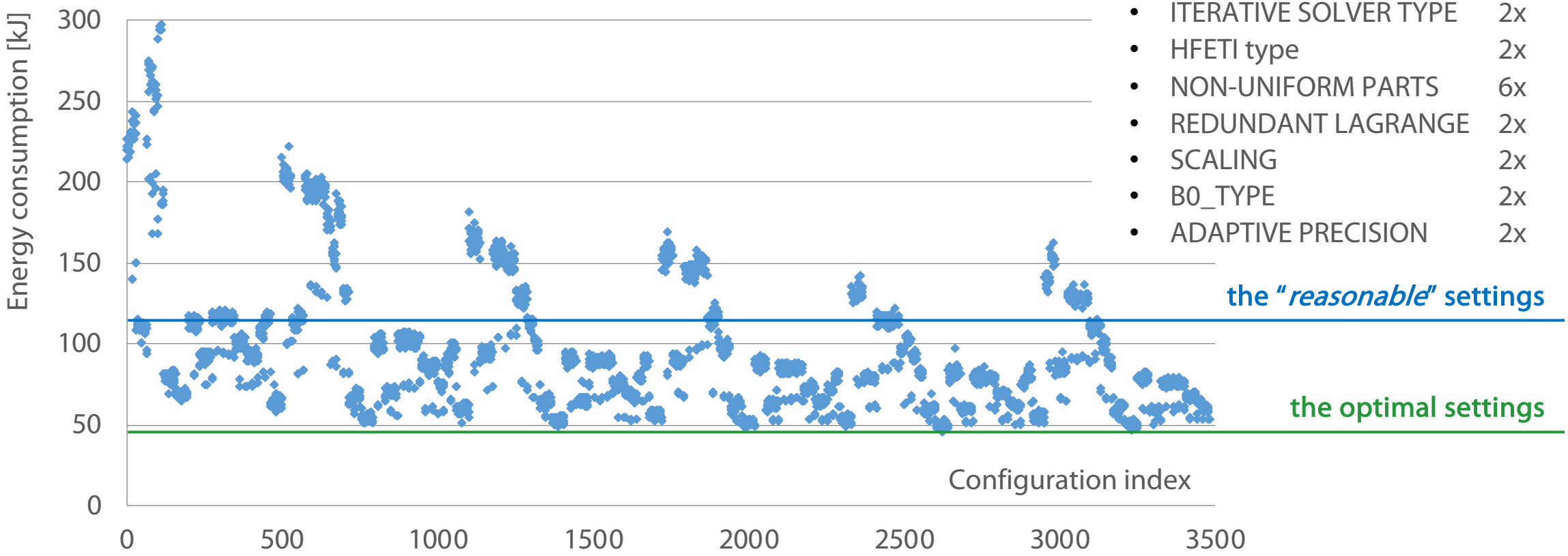
Application parameters tuning of the ESPRESO

50% - 66% against ”reasonable” settings

86% against the worst case

9 parameters
3840 combinations

- FETI METHOD 2x
- PRECONDITIONER 5x
- ITERATIVE SOLVER TYPE 2x
- HFETI type 2x
- NON-UNIFORM PARTS 6x
- REDUNDANT LAGRANGE 2x
- SCALING 2x
- B0_TYPE 2x
- ADAPTIVE PRECISION 2x



Application parameters tuning

Application parameter tuning parameters is very promising

- application configuration parameters are given in the input file
- each setting requires an individual start of the application
  - tool performs automatic search of application parameter space

Application	number of parameters tested / total number of options	Energy savings compared to the worst settings	Energy savings compared to default or reasonable settings
ESPRESO	9 / 3840	86%	50 – 66%
ELMER	1 / 40	97%	50 – 75%
OpenFOAM	2 / 12	24%	8%
INDEED	3 / 12	35%	25%

Thank you

VSB TECHNICAL UNIVERSITY OF OSTRAVA | IT4INNOVATIONS NATIONAL SUPERCOMPUTING CENTER