Mtech thesis presentation

Stage-1

Presented by: Shikhar Jain
CS22MTECH02002
Advisor: Dr. Ramakrishna Upadrasta

CONTENT

- *Dynamic voltage and frequency scaling (DVFS)*
  - Motivation
  - Related Work
  - Research questions & results
    - Platform & Tools
    - Framework Schema & Challenges faced
    - Results
      - Intel-Rocketlake
      - ARM+External Power Meter
  - Key Observations
  - Future Work
- *IR2Vec Pip package*
- *GeMS*
- *Contributions Summary*

DVFS
(Dynamic Voltage and Frequency Scaling)

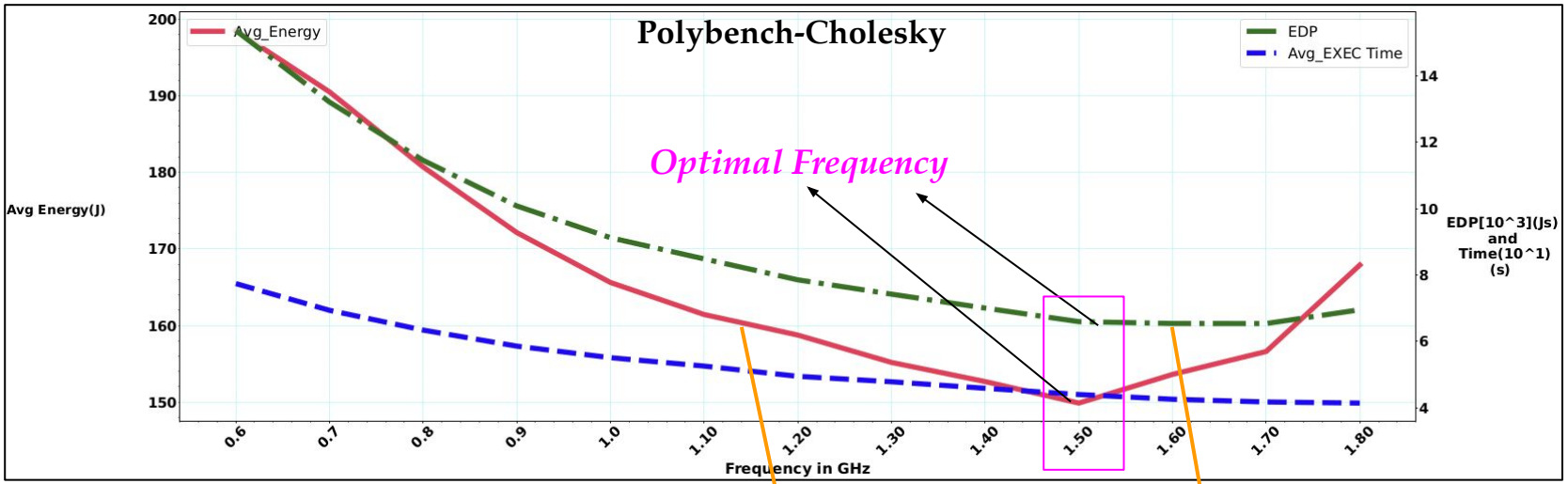
Motivation: DVFS as a solution for Green Computing

$$P = c \cdot V^2 \cdot f + P_S$$

Core Voltage

Core Frequency

But Voltage can't be controlled by user



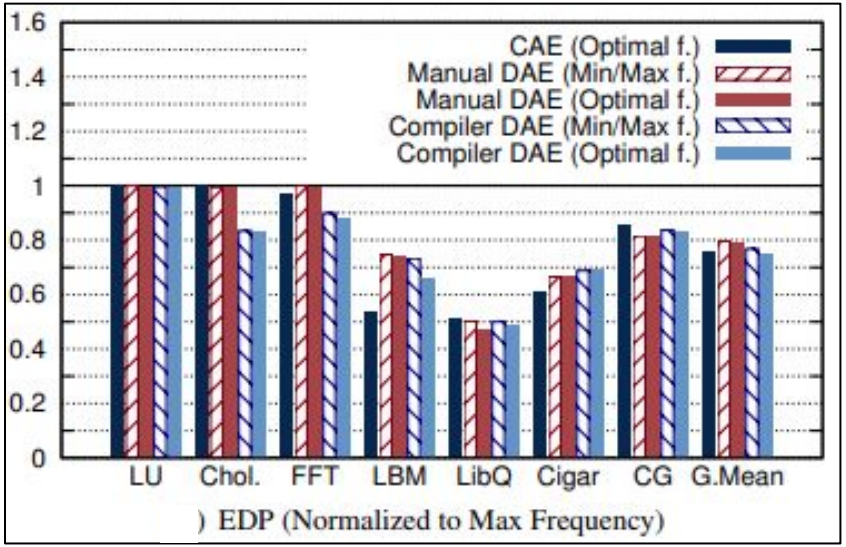
Energy Consumed : E = P*T
F(Core-frequency)

Objective to Optimize for EDP = E*T

RELATED WORK

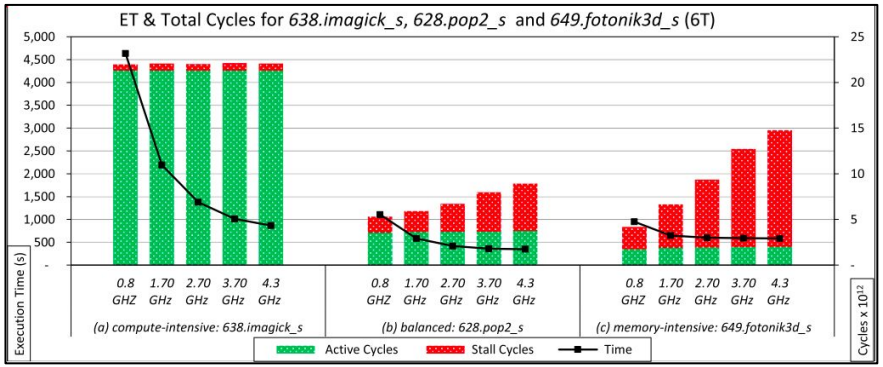
Fix the Code Don't tweak hardware [CGO'14]

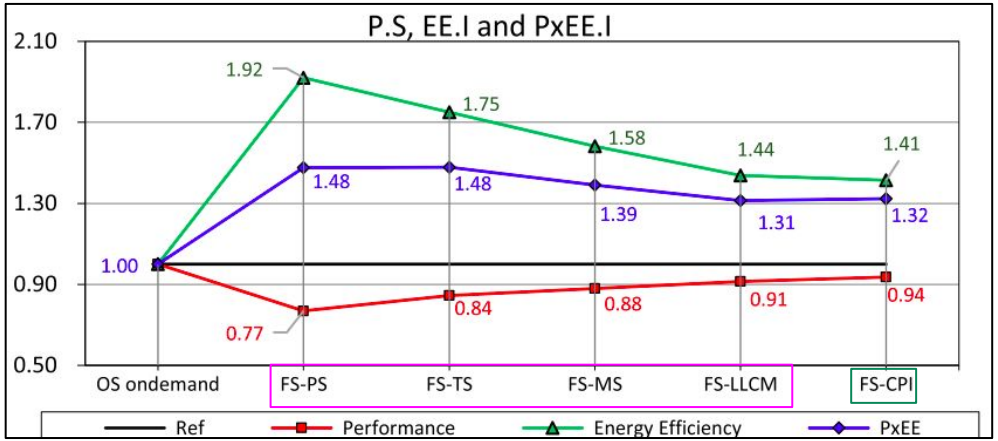
- Study decoupling of execute (compute) and access (mem) phases
  - Target: affine (polyhedral) and non-affine code
  - Improvements
    - EDP reduced upto 25%
    - Performance
- Platform: Intel Haswell
- Limitations
  - Software prefetching
  - Optimal frequency selection by profiling



Alexandra Jimborean, Konstantinos Koukos, Vasileios Spiliopoulos, David Black-Schaffer, and Stefanos Kaxiras. 2018. Fix the code. Don't tweak the hardware: A new compiler approach to Voltage-Frequency scaling. In Proceedings of Annual IEEE/ACM International Symposium on Code Generation and Optimization (CGO '14). Association for Computing Machinery, New York, NY, USA

PMU-Events-Driven DVFS [ACM TOMPECS ‘22]





- Platform used: Coffee-Lake (Core i7-8700K)
- Limitations
  - Heuristics, Not program property guided
  - Algorithm invocation overhead

Ranjan Hebbar and Aleksandar Milenković. 2022. PMU-Events-Driven DVFS Techniques for Improving Energy Efficiency of Modern Processors. ACM Trans. Model. Perform. Eval. Comput. Syst. (March 2022)

RAPL in Action [ACM TMPEC'18]

- Low performance overhead
- Correlation(Package power, Core temperature)

- MSR overflow
- Per Core measurement not supported

Kashif Nizam Khan, Mikael Hirki, Tapio Niemi, Jukka K. Nurminen, and Zhonghong Ou. 2018. RAPL in Action: Experiences in Using RAPL for Power Measurements. ACM Trans. Model. Perform. Eval. Comput. Syst.

Research questions & Results

Some Research Questions and Solutions

Research Questions:

1. Study DVFS for new architectures
2. Study mutual effects of frequency and temperature
    - Frequency
        - EDP, Core-temperature
        - For Arm - Power, Current, Voltage
    - Temperature
        - Core-frequency modulation
        - Kernel energy consumption
3. Validate Experimental Results
    - Across machines
    - Across softwares (Intel RAPL)
    - Vendor solutions (Intel P_state)

Some Solutions:

- [ARM] Implement an automated server client framework
- [Intel] Implement an experimental framework

Experimental Setup

Platforms & Tools

Intel

- Intel Rocketlake (11th Gen) i5 11400 **x2**

Arm

- Quad core Cortex-A72 (ARM v8) 64-bit SoC **x2**
- **UM25C** External Power Meter

Intel

- Performance monitoring
  - PAPI-7.2
- Core-temperature
  - lm-sensors
- Prefetcher Control
  - Likwid-features
- Msr tools

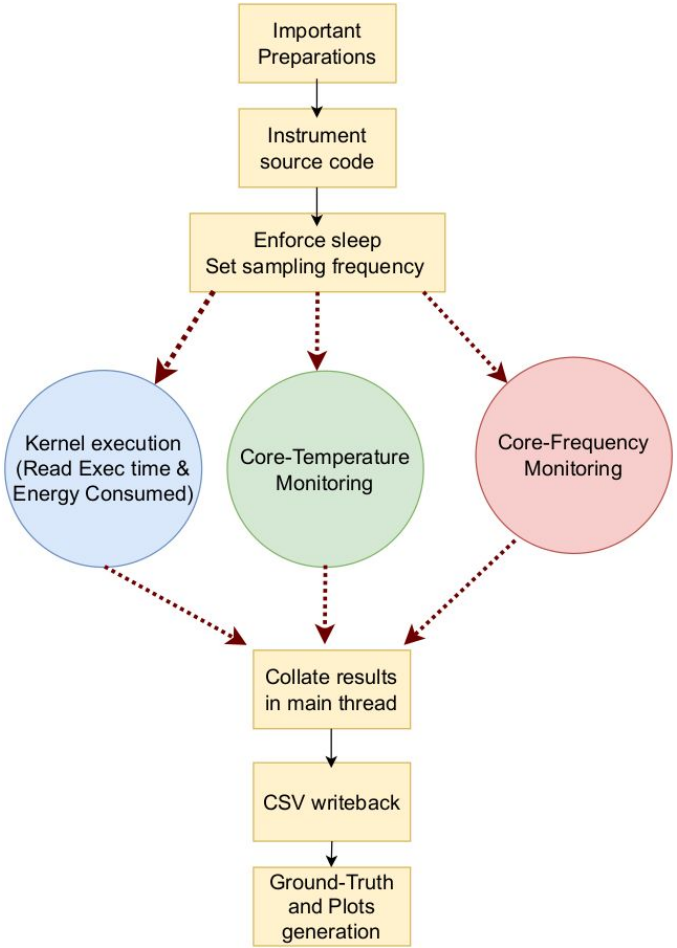
Common

- Perf
- Core isolation/affinity
  - Isolcpus, taskset
- Power governors
  - Cpupower/Cpufreq-utils

Arm

- Communication
  - Pybluez
  - Sockets
- Core- temperature
  - Gpizero package

Common framework schema



Problems & solutions

MSR overflow detection and correction

- MSR Range 0 - 262143 unit
- Examples
  - `3mm` , ~68k iterations, 8.8 hrs
  - `jacobi-2d` , ~46k iterations, 11.7 hrs

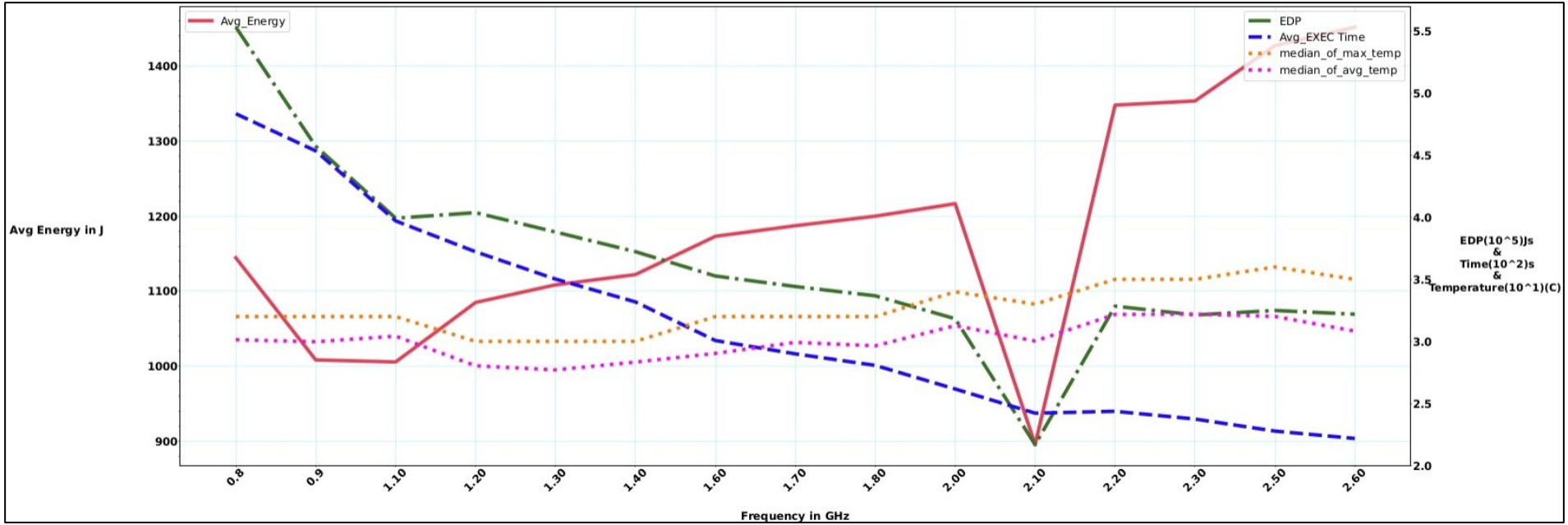
(N*262143+(MSR value))*energy unit; N=no.of overflows

Loss of statistical soundness

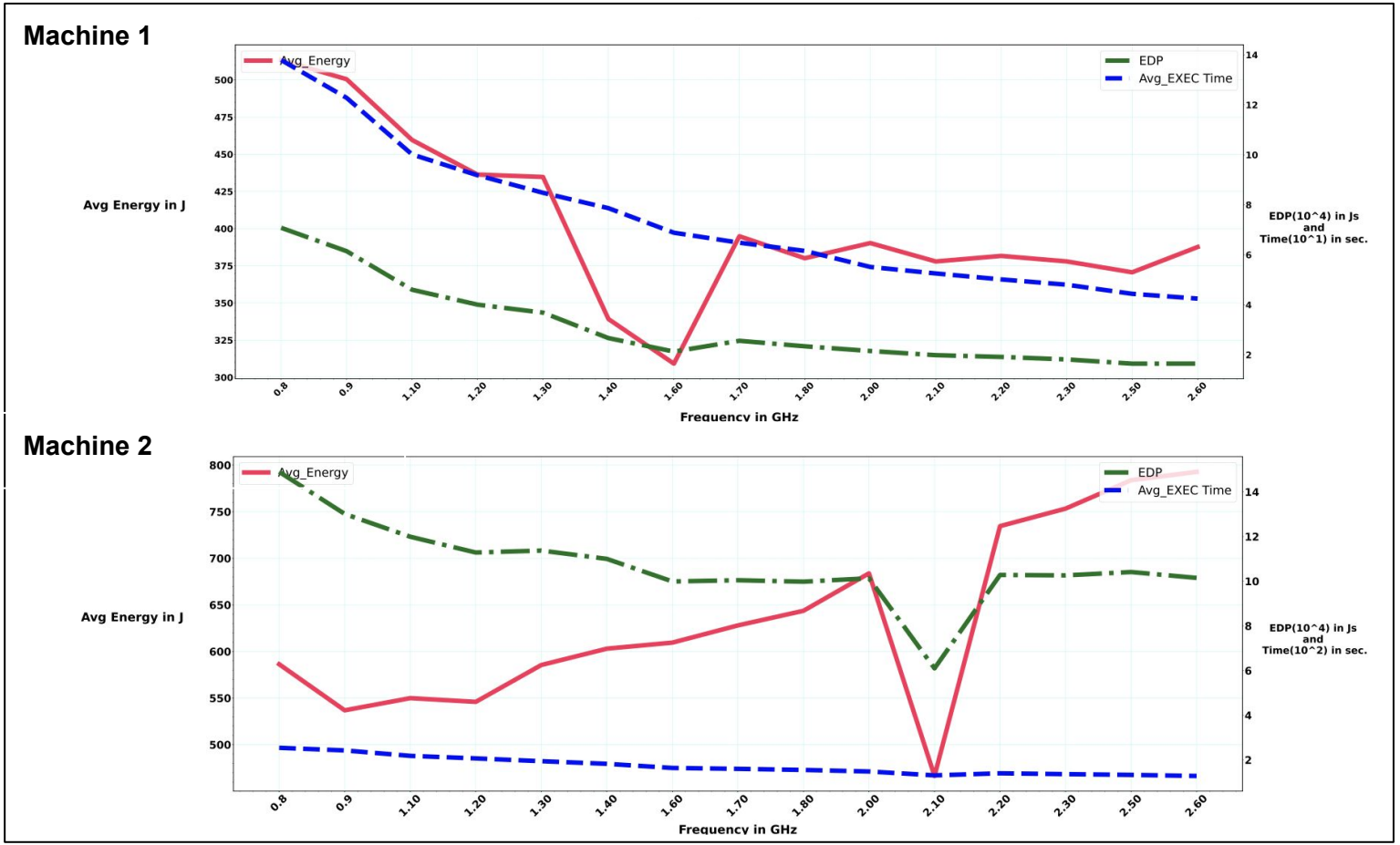
- Data Cache Flushing
- Disable prefetchers
- Usage of standalone threads
- Prevent core overheating
- Ensure correct bios configuration

Results for Intel (x86)

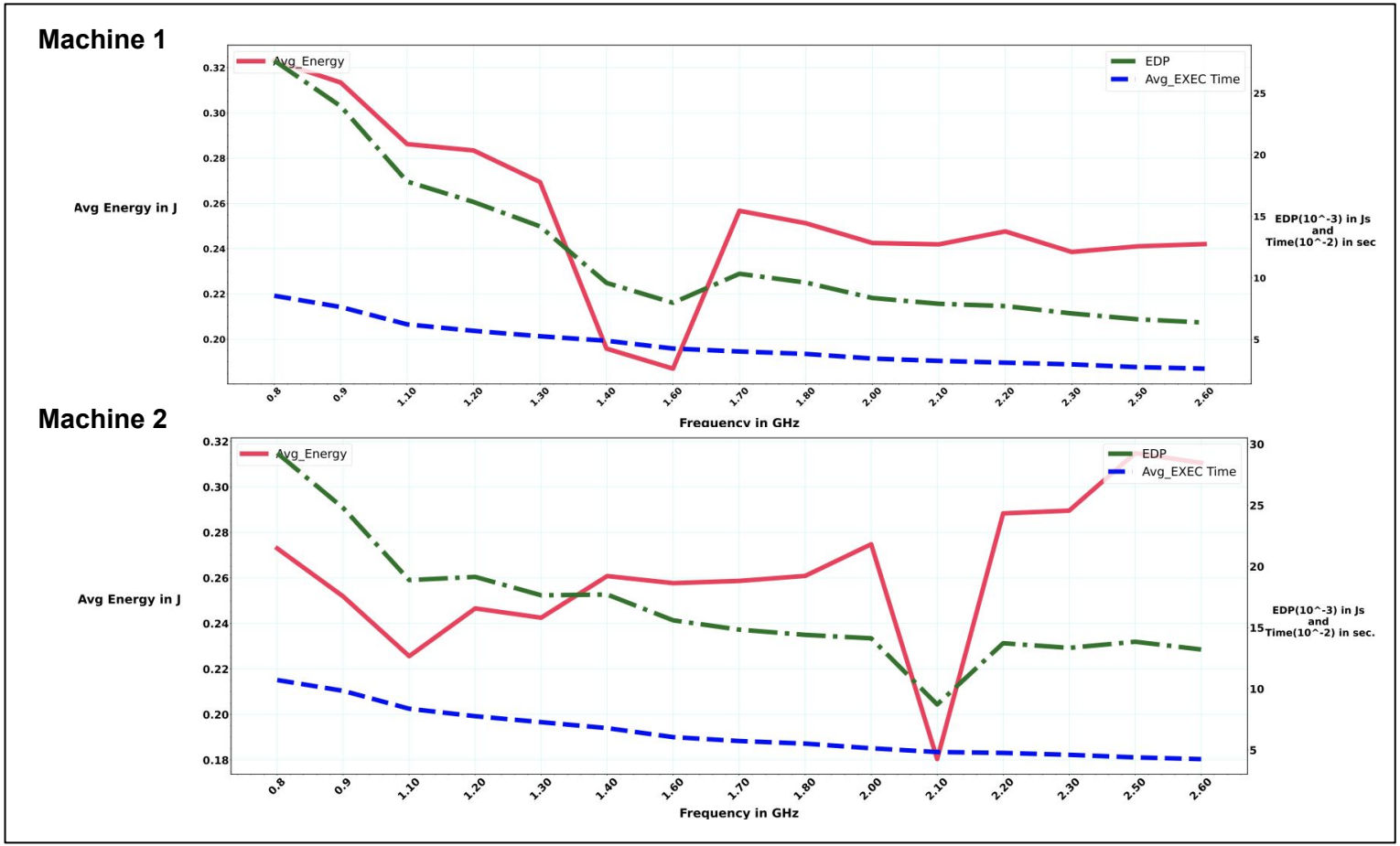
Results - Energy, EDP and Execution time on PolyBench: Jacobi-2d



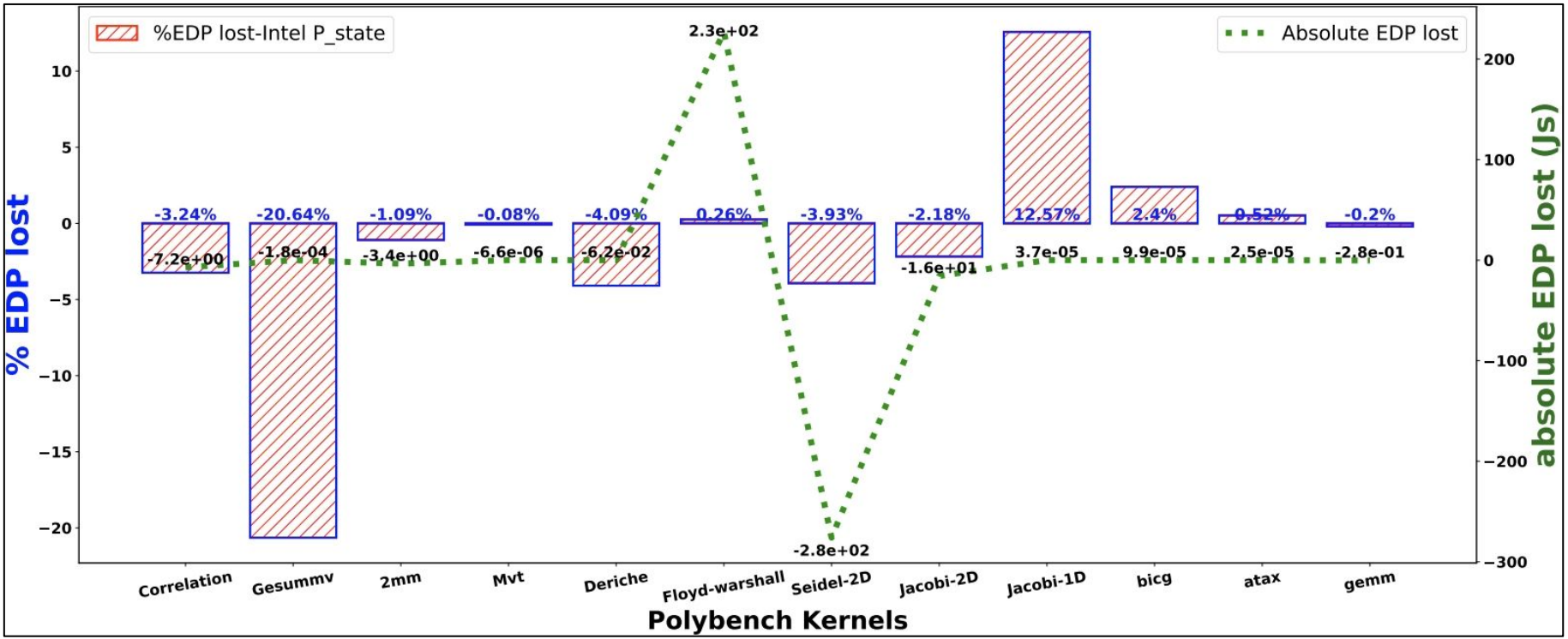
Challenge - Dissimilarity between same machines: gemm



Challenge - Dissimilarity between same machines: atax



Static frequency selection vs Intel P_state



Results for ARM-64

Power Measurement: Problems & proposed solutions

TARGET

- Instantaneous Volt, Current
- Average Instantaneous Power
- Execution time

Architecture	S/W interface	Internal Powermeter
x86 (Rocketlake)	✓ (RAPL)	✗
ARM (Raspberry-pi4)	✗	✗

Raspberry-Pi4 + External Powermeter (UM25C)

Reasons to use

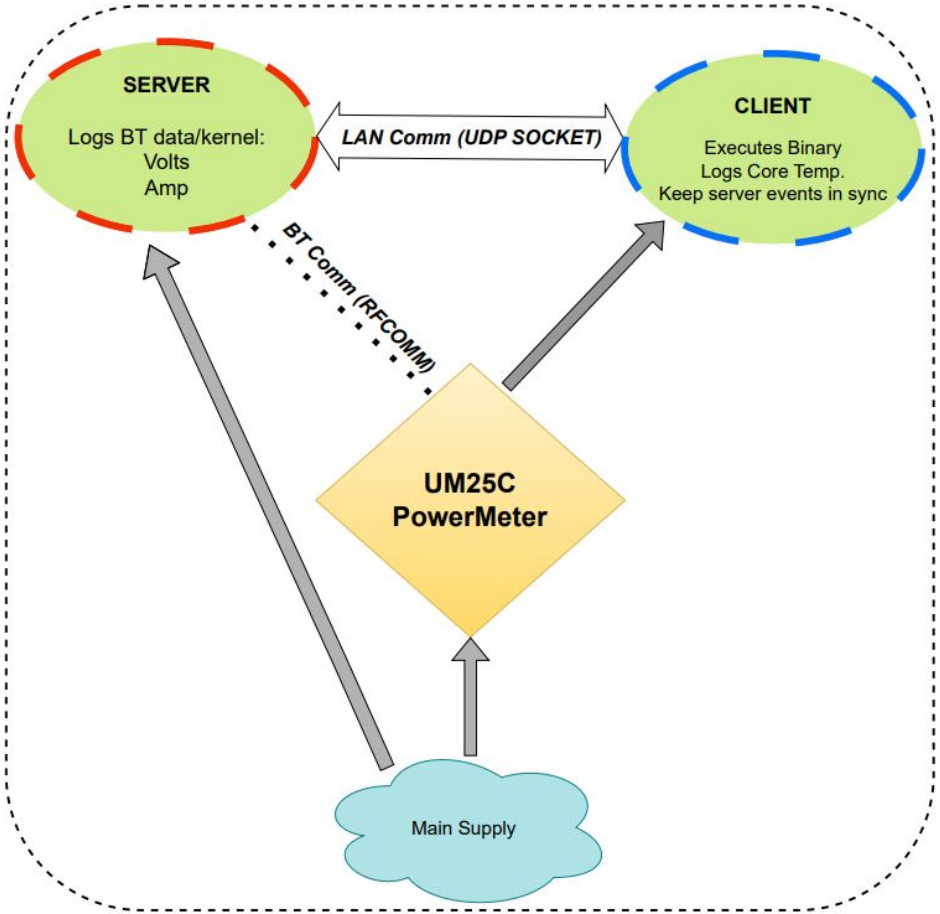
- BT packet encoding info available*
- Measures whole system power

Difficulties in setup

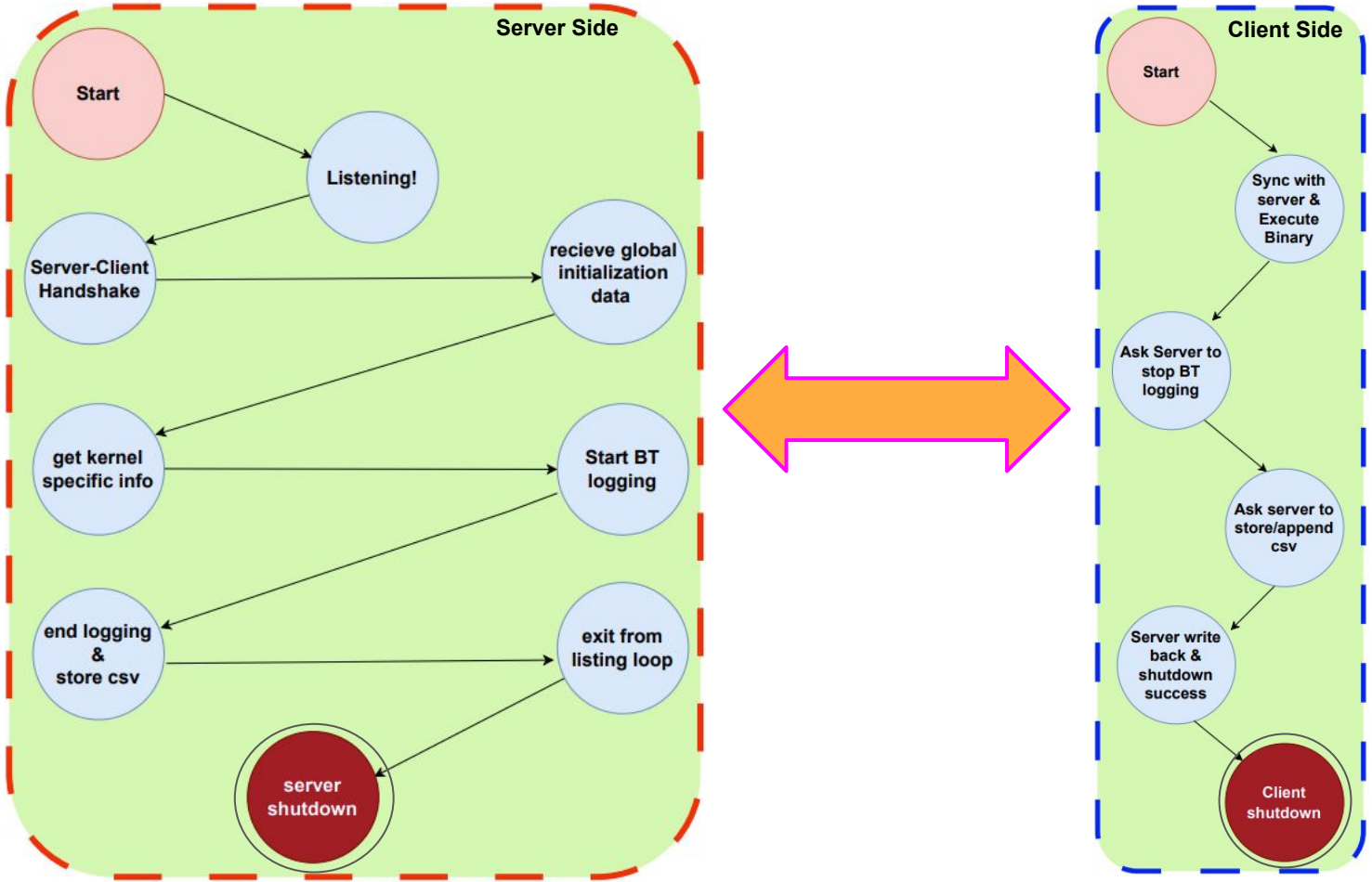
- Connect using LAN & Bluetooth
- In-sync communication
- Automated & efficient sampling

*https://github.com/anfractuosity/usbmeter

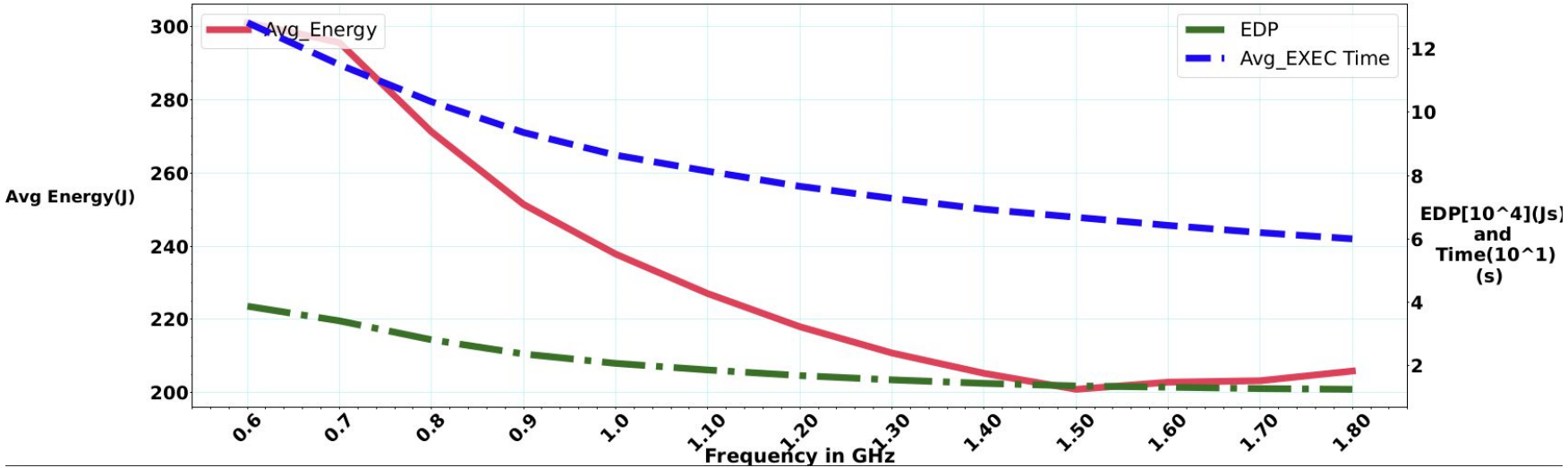
Server-Client & Power Meter Communication Framework

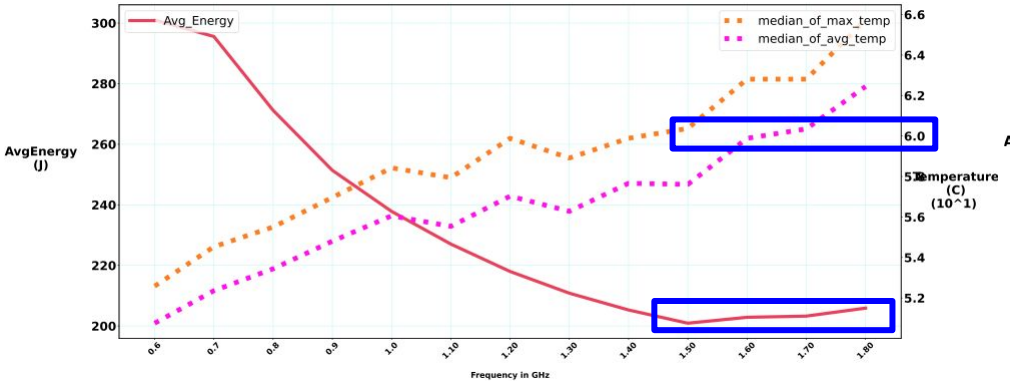


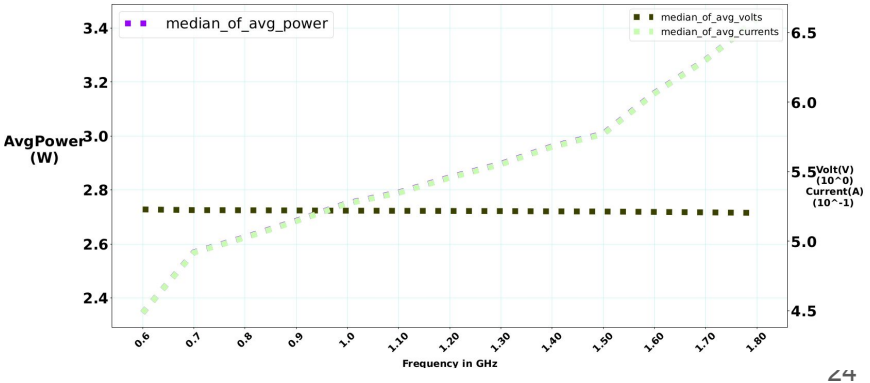
Server & Client State Diagram



Results- Energy, EDP and Execution time : Jacobi-2D







Key Observations

- Core-frequency effects
  - EDP, Energy Consumption, Execution Time, Core-temperature
- No modulation in assigned core frequency
- “Buckets” of optimal frequency, different across similar machines
- Intel
  - HWP is comparable to static frequency selection
  - Validation across similar machine isn’t feasible
- ARM
  - Core temperature effects optimal frequency
  - Volts $\not\propto$ core-frequency, Current $\propto$ core-frequency, Power $\propto$ Current

Future Work

- **Idea: DVFS over uncore & off-core domains**
  - Research Question:
    - *“Proposing Compiler driven approach for uncore DVFS”*
  - Related Work: [1] SysScale

- **Idea: Don’t “conquer” HWP or intel pstate, add software solutions on top**
  - Research Question:
    - *“Explore relevant alternative power saving techniques”*
  - Related Work: [2] CoPPer

[1] Jawad Haj-Yahya, Mohammed Alser, Jeremie Kim, A. Giray Yağlıkçı, Nandita Vijaykumar, Efraim Rotem, and Onur Mutlu. 2020. SysScale: exploiting multi-domain dynamic voltage and frequency scaling for energy efficient mobile processors. In Proceedings of the ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA '20)

[2] C. Imes, H. Zhang, K. Zhao and H. Hoffmann, "CoPPer: Soft Real-Time Application Performance Using Hardware Power Capping," *2019 IEEE International Conference on Autonomic Computing (ICAC)*

IR2Vec Python Package

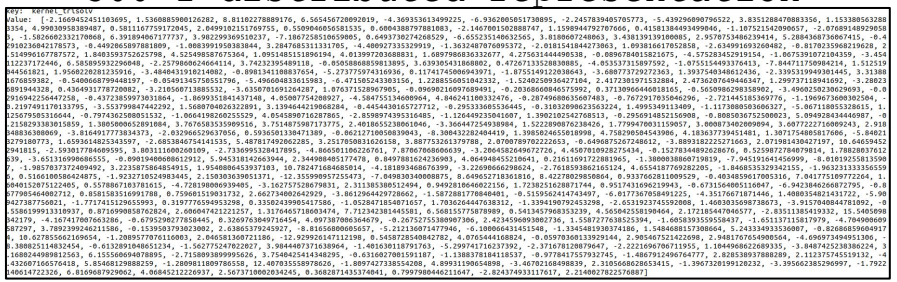
Highlights

- **Aim:** To deploy **IR2Vec** as a pip installable package
- Output
  - Many-linux 2014 compliant wheel files (Python 3.6-3.11)
  - Package Source-Code
- **Tools:** `Cpython, LLVM-12, Docker, Many-linux 2014`
- Artifacts were pushed to **IR2Vec** repo.

Python-3.10**

```
import IR2Vec_pkg as ir2vec
d = ir2vec.generateEmbeddings("/home/shikharj/trisolv.ll", "fa", "f")
if d["Status"]:
    for x in d["Function_Dict"]:
        print("key: ", x)
        print("Value: ", d["Function_Dict"][x])
    print("\n\n")
    print(d["Instruction_Dict"].keys())
else:
    print(d["Message"])
```

300*1 distributed representation



**Commit to IITH-Compilers/IR2Vec (Main Branch)

GeMS(Generating Million Scopes)

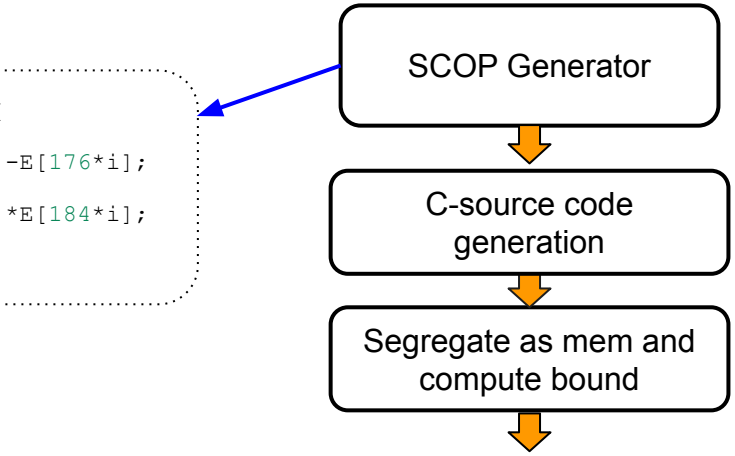
Presented at: IMPACT 2023 (13th Workshop)

Venkatakeerthy S, Nilesh Shah, Anilava Kundu, Shikhar Jain and Ramakrishna Upadrasta.

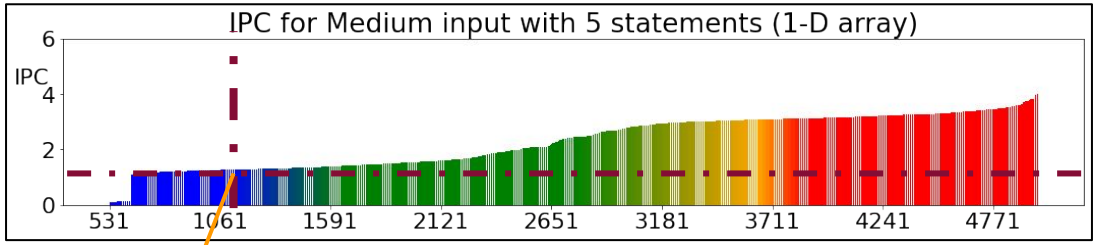
GeMS : Beyond SCOP generation

```
for(int i = 0;i<1000;i++){
C[184*i]=A[184*i]*B[184*i]-E[176*i];
D[184*i]=D[192*i]+E[192*i]*E[184*i];
        }
```

A Generated SCOP



- **Reuse** is "control" knob
- IPC ∝ Reuse
- ⬇ Reuse ⬆ Mem Bound
- ⬆ Reuse ⬇ Compute Bound
- Enforcing Compulsory misses [Spatial Locality]



IPC < 1.0 (or Machine Balance)

Contributions & Future Work

- Contributions
  - DVFS
    - Study of
      - New architectures, Intel's proprietary p_state governor
      - Polybench-suite on intel & arm
      - Emperical relationship among core temperature, frequency, etc.
    - Tools/framework
      - PAPI 7.2 native events for intel 11th gen (Rocketlake)
      - Server client communication framework, automated setup
  - IR2Vec Manylinux2014 compliant python package
- Publication

  GeMS: Towards Generating Millions of SCoPs.

  Accepted in IMPACT 2023 Workshop, Jan 2023.

  *Authors: Venkatakeerthy S, Nilesh Shah, Anilava Kundu, **Shikhar Jain** and Ramakrishna Upadrasta.*
- Future Work
  - Alternative power saving techniques
  - Formulation of static model guided by compiler , LLVM support

THANK YOU

Motivation : Feasibility of DVFS towards energy conservation

Power ∝ Performance (~MIPS,FLOPS)

$P = c \cdot V^2 \cdot f + P_S$

Core Voltage

Core Frequency

$P_{static} \sim N * V * e^{-Vt}$

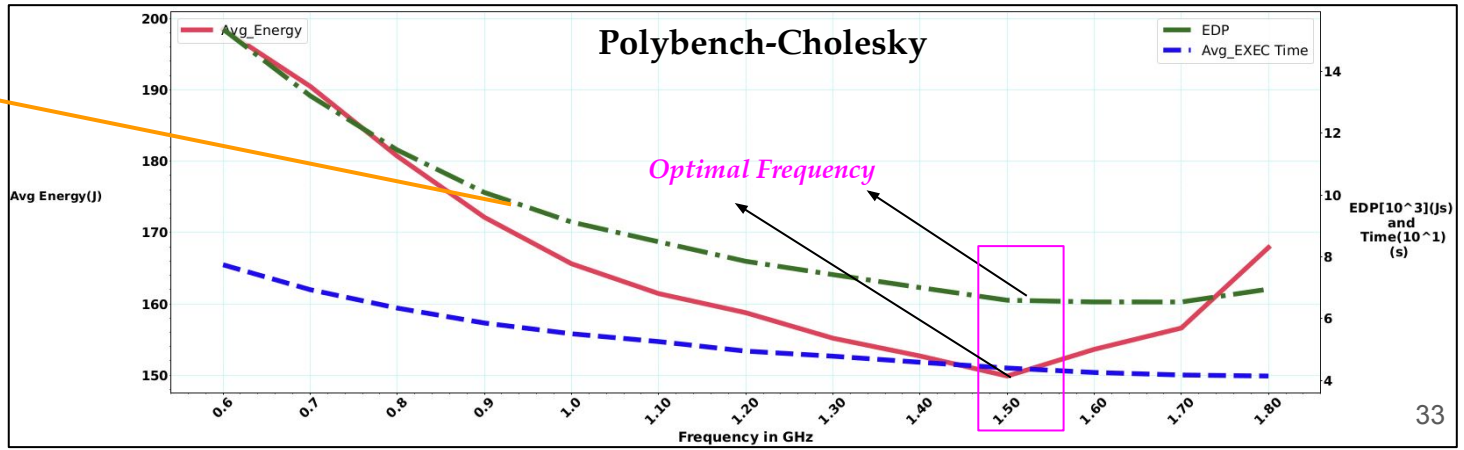
$P_{dynamic} \sim N * C * V^2 * f * A$

But Voltage can't be controlled by user

Execution Time ∝ Core-frequency

Energy Consumed during kernel Execution: P*T → Energy ∝F(Core-frequency)

Objective to Optimize for EDP=E*T



Points to be understood

- Study DVFS for :
  - New architectures like intel 11th gen & cortex-A72
  - Vendor specific solutions (Intel P_state)
  - Results validation across similar machines
- Core frequency effects on properties :
  - EDP, Core-temperature
  - For Arm - Power,Current,Volts
- Effects of high core temperature on :
  - Core-frequency modulation
  - Kernel’s energy consumption
- Intel RAPL as “internal powermeter”

- Implement Missing tools/frameworks for both Intel & Arm

AIM

- Study of recent (SOTA) static & dynamic techniques.
- Prospects of DVFS across
  - Multiple architectures (Workstation [Intel] & Embedded [ARM])
  - Newer architectures (ex Intel 11th gen -Rocketlake)
- Correlation of Core frequency with
  - Execution time, Energy consumed
  - EDP, Temperature
  - Power, Volt, current (ARM)
- Vendor provided solutions vs Static DFS (Intel)
- Parameters effecting actual kernel execution
  - Temperature
  - Architecture constraints (Intel)
- Implement Missing tools/frameworks (ARM)

AIM

- Study DVFS
  - Vendor provided solutions vs Static DFS (Intel)
  - Across diverse & recent architectures
- Correlate core frequency with
  - Execution time, Energy consumed
  - EDP, Temperature
  - Power, Volt, current (ARM)
- Parameters effecting actual kernel execution
  - Temperature
  - Architecture constraints (Intel)
- Implement Missing tools/frameworks (ARM)

AIM

- Prospects of DVFS?
  - Vendor provided solutions vs Static DFS (Intel)
- Correlate core frequency with
  - Execution time, Energy consumed
  - EDP, Temperature
  - Power, Volt, current (ARM)
- Parameters effecting actual kernel execution
  - Temperature
  - Architecture constraints (Intel)
- Implement Missing tools/frameworks (ARM)

Final Steps

1. Check drop in assigned core freq. due to high temperature

Correct Bios Configuration

- Race to Halt: off
- Intel Turbo Boost: off
- Intel Hyper-Threading: off

2. Result Collation
3. Groundtruth generation (Min-EDP)
4. Plots Generation

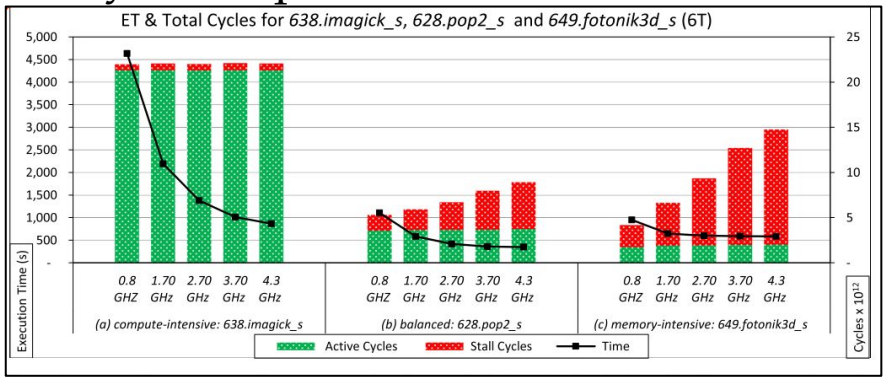
Platforms description

Intel Workstation

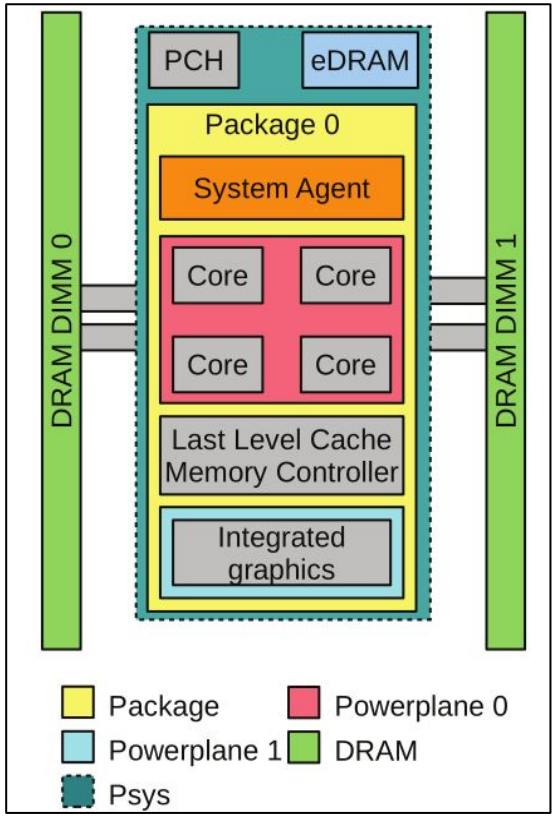
- Intel Rocketlake (11th Gen): i5 11400 **x2**
- **Tools:** cpupower, taskset, isolcpus, Likwid, lm-sensors, msr-tools, perf, papi

ARM

- **ARM-64** (Raspberry-Pi4) **x2**
- **UM25C** External Power Meter
- **Tools:** cpupower, gpiozero, PyBluez, perf



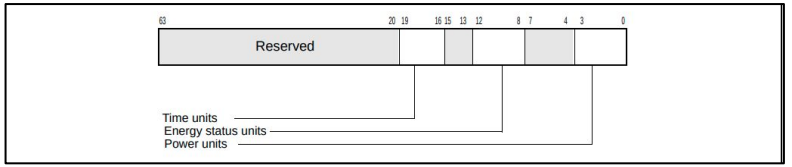
Intel RAPL (Running Average power Limit)



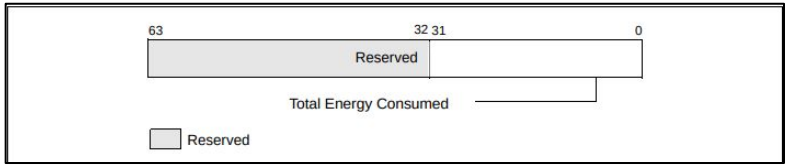
RAPL's logical topology

Model Specific Registers

- Control registers provided by Processor vendors
- Get or manipulate Core/Off-core features



RAPL POWER UNIT MSR



RAPL PP0/PP1 ENERGY STATUS MSR

RAPL in Action: Experiences in Using RAPL for Power Measurements

https://www.intel.com/content/dam/www/public/us/en/documents/manuals/64-ia-32-architectures-software-developer-vol-3b-part-2-manual.pdf

RAPL in Action

- Low performance overhead
- Correlation between Package power and Core temperature
- Fixed refresh rate(~1ms)

- Model specific register(MSR) overflow
- Non-Atomic MSR updates
- Individual core measurement not supported

How to read MSRs in C

```
uint64_t msr_value;
/* Haswell: units of 61 microjoules */
/* MSR_PKG_ENERGY_STATUS is at address 0x611 */

double energy_units = pow(0.5, 14);
int fd = open("/dev/cpu/0/msr", O_RDONLY);
if (fd < 0) {
        perror("open");
        return -1;
}
if (pread(fd, &msr_value, 8, 0x611) < 0) {
        perror("pread");
        return -1;
}
double energy = msr_value * energy_units;
printf("%f\n", energy);
```

Kashif Nizam Khan, Mikael Hirki, Tapio Niemi, Jukka K. Nurminen, and Zhonghong Ou. 2018. RAPL in Action: Experiences in Using RAPL for Power Measurements. ACM Trans. Model. Perform. Eval. Comput. Syst.

Observations & solutions

MSR overflow detection and correction

- Energy Status MSR
- Range 0 - 262143 unit
- Examples
  - `3mm` ,~68k iterations ,8.8 hrs
  - `jacobi-2d` , ~46k iterations, 11.7 hrs

Loss of statistical soundness

- Data Cache Flushing
- Disable prefetchers
- Usage of standalone threads
- Isolcpus, taskset

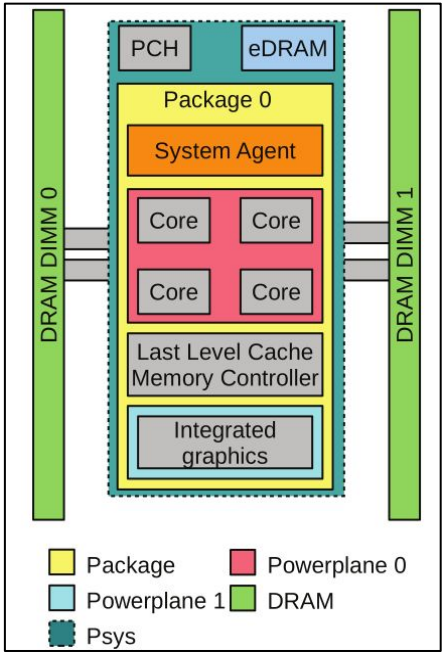
Some precautions

- **Uniform Core temperature**
  - Low sampling Frequency
  - Low core-utilization
- **Correct Bios Configuration**
  - Race to Halt: off
  - Intel Turbo Boost: off
  - Intel Hyper-Threading: off

RAPL in Action

- Low performance overhead
- Correlation between Package power & Core temperature
- Fixed refresh rate(~1ms)

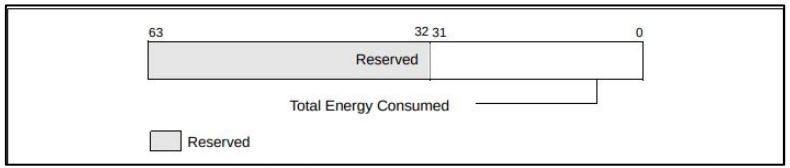
- Model specific register(MSR) overflow
- Individual core measurement not supported



How to read MSRs in C

```
uint64_t msr_value;
/* Haswell: units of 61 microjoules */
/* MSR_PKG_ENERGY_STATUS is at address 0x611 */

double energy_units = pow(0.5, 14);
int fd = open("/dev/cpu/0/msr", O_RDONLY);
if (fd < 0) {
        perror("open");
        return -1;
}
if (pread(fd, &msr_value, 8, 0x611) < 0) {
        perror("pread");
        return -1;
}
double energy = msr_value * energy_units;
printf("%f\n", energy);
```



RAPL PP0/PP1 ENERGY STATUS MSR

Kashif Nizam Khan, Mikael Hirki, Tapio Niemi, Jukka K. Nurminen, and Zhonghong Ou. 2018. RAPL in Action: Experiences in Using RAPL for Power Measurements. ACM Trans. Model. Perform. Eval. Comput. Syst.

Important Tools & Interfaces

Performance Monitoring
- Perf (Arm & Intel)
- PAPI-7.2 (Intel)

Communication
- Pybluez package (Arm)
- Sockets package (Arm)

Core Temperature Measurement
- lm-sensors(Intel) &
- Gpizero package(Arm)

Core Isolation and affinity
- isolcpus
- Taskset

Miscellaneous
- Likwid-features (Intel)
- Cpupower/Cpufrequtils
- Msr tools(Intel)

Important Tools & Interfaces

Performance Monitoring
- Perf (Arm & Intel)
- PAPI-7.2 (Intel)

Communication
- Pybluez package (Arm)
- Sockets package (Arm)

Core Temperature Measurement
- lm-sensors(Intel) &
- Gpizero package(Arm)

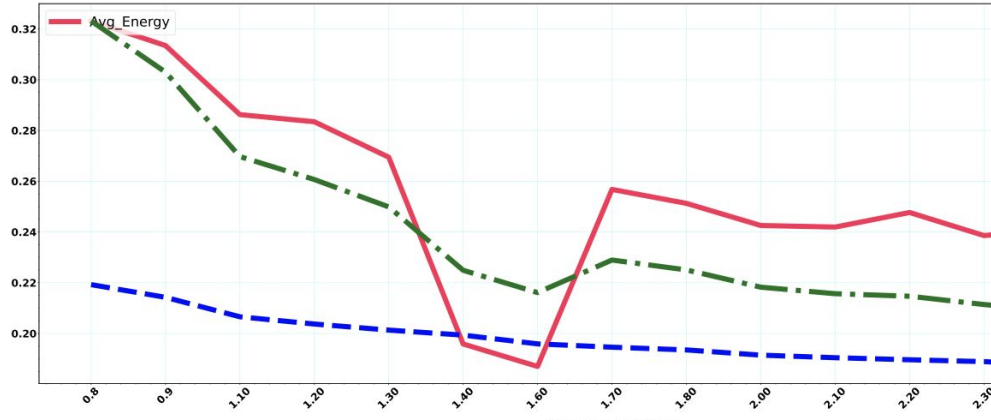
Core Isolation and affinity
- isolcpus
- Taskset

Get IPC

Miscellaneous
- Likwid-features (Intel)
- Cpupower/Cpufrequtils
- Msr tools(Intel)

BACKUP


Avg Energy in J
%EDP lost-Intel P_state
2.3e+0
-3.24%
-20.64%
-1.09%
-0.08%
-4.09%
0.26%
-3.93%
-2.18%
12.57%
2.4%
0.52%
-0.2%


Avg_Energy
0.32
0.30
0.28
0.26
0.24
0.22
0.20
0.8
0.9
1.10
1.20
1.30
1.40
1.60
1.70
1.80
2.00
2.10
2.20
2.30
Frequency in GHz

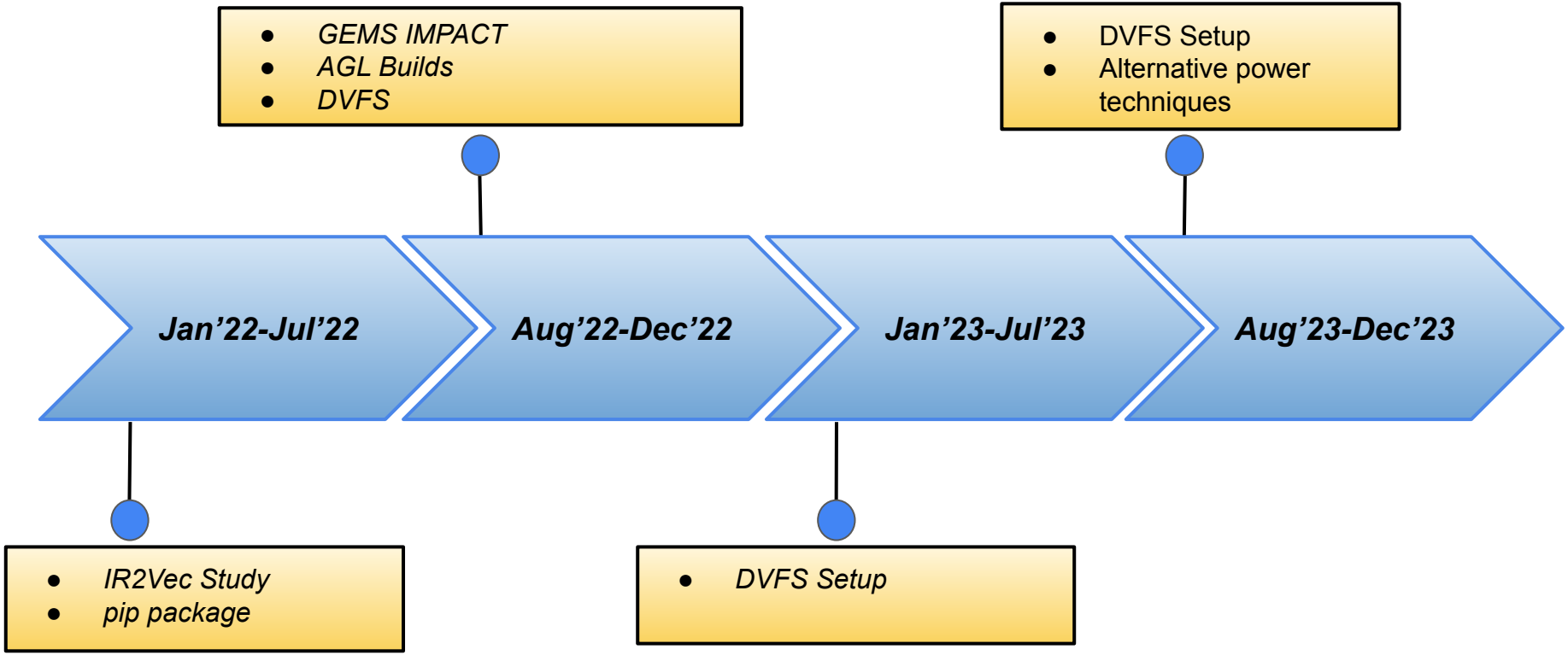
Experiences with Experiments

- MSR overflow detection and correction
  - Max 262143 units can be measured
  -
- Correct Bios Configuration
  - Race to Halt : off
  - Intel EIST : off/on
  - Intel Turbo Boost : off
  - Intel Hyper-Threading: off
- Maintain Similar core temp for each kernel run
  - Enforcing Sleep
  - Low sampling Frequency ; Helps in low core-utilization
  - Better kernel execution schedule
- Ensuring minimal overhead of context-switch
  - IsolCpus
  - Taskset
- For Better sampling accuracy
  - Data Cache Flushing
  - Disable prefetchers
  - Kernel-execution, sampling [Core-Freq ,Core-temp] as standalone threads

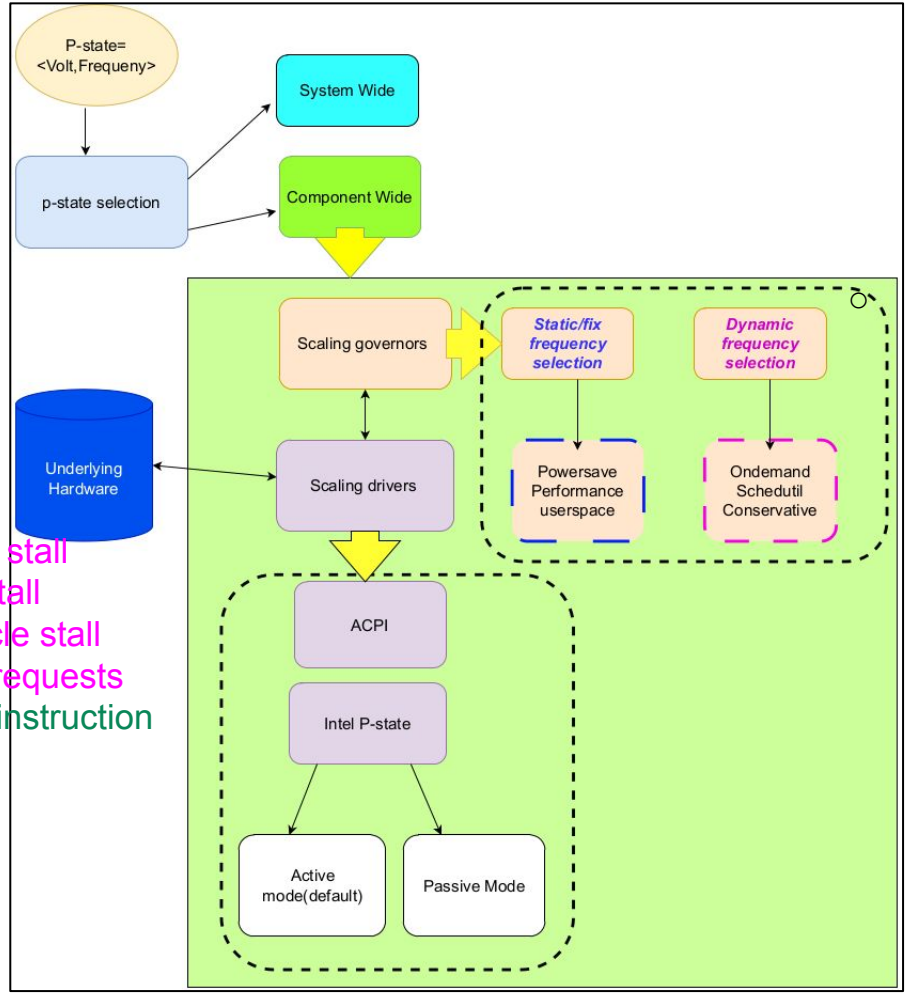
Key Observations

- DFS as a software solution for DVFS on Intel Rocketlake
  - Intel Rocketlake implements Hardware P-state based DVFS
  - Only software DFS doesn’t lead to significant EDP improvements
- DFS approach on ARM leads to fixed bins of core frequency
  - Higher Core-frequency better the EDP, Energy Consumption, Execution Time
  - Core temperature plays significant role in deciding optimal frequency

TIMELINE


GEMS IMPACT
AGL Builds
DVFS
DVFS Setup
Alternative power techniques
Jan'22-Jul'22
Aug'22-Dec'22
Jan'23-Jul'23
Aug'23-Dec'23
IR2Vec Study
pip package
DVFS Setup

CPU Power Management for linux systems



- FS-PS : Pipeline slot stall
- FS-TS : Total cycle stall
- FS-MS : Memory cycle stall
- FS-LLCM : Off-core requests
- FS-CPI : Cycles per instruction

Seperate analysis and optimizations for access & execute phase

https://www.kernel.org/doc/html/v4.19/index.html

Observations

MSR overflow detection and correction

- Energy Status MSR
- Max measurable energy (uJ) 262143 unit
- Example
  - `3mm` ,~68k iterations ,8.8 hrs
  - `jacobi-2d` , ~46k iterations, 11.7 hrs

Correct Bios Configuration

- Race to Halt: off
- Intel Turbo Boost: off
- Intel Hyper-Threading: off
- Speed Step: on/off

Minimal overhead of context-switch

- Isolcpus
- Taskset

Uniform core temperature

- Enforcing Sleep
- Low sampling Frequency, low core-utilization
- Better kernel execution schedule

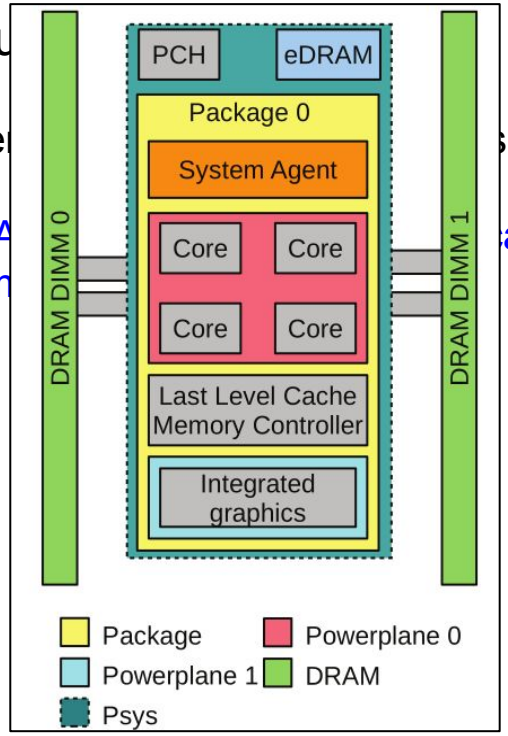
For Better sampling accuracy

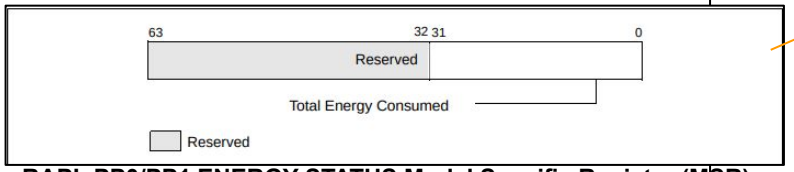
- Data Cache Flushing
- Disable prefetchers
- Kernel-execution, Core-Freq ,Core-temp as standalone threads

Tasks Performed

Contribu

- Ene
- An A … cation fram

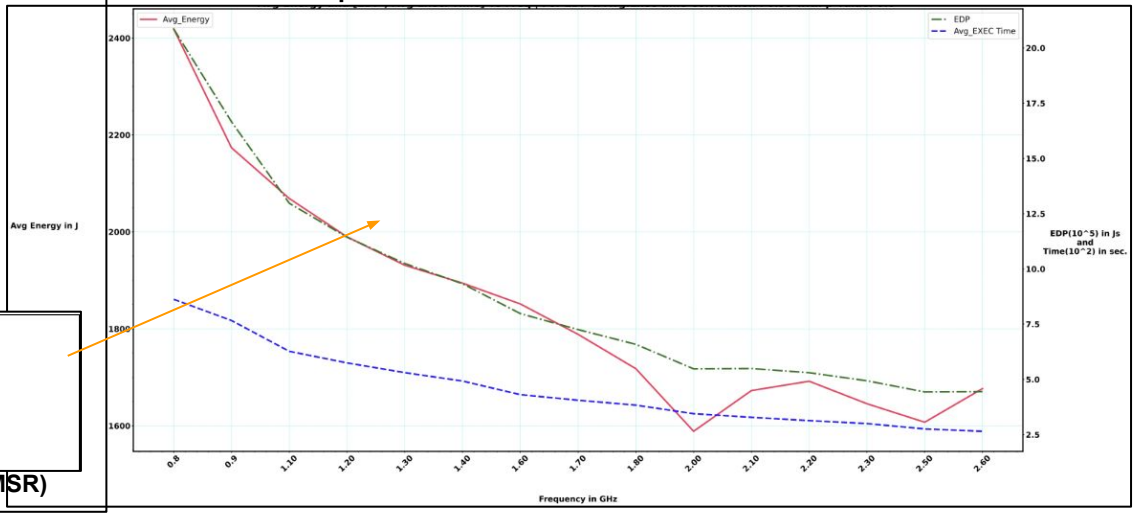




RAPL PP0/PP1 ENERGY STATUS Model Specific Register (MSR)

New Contributions

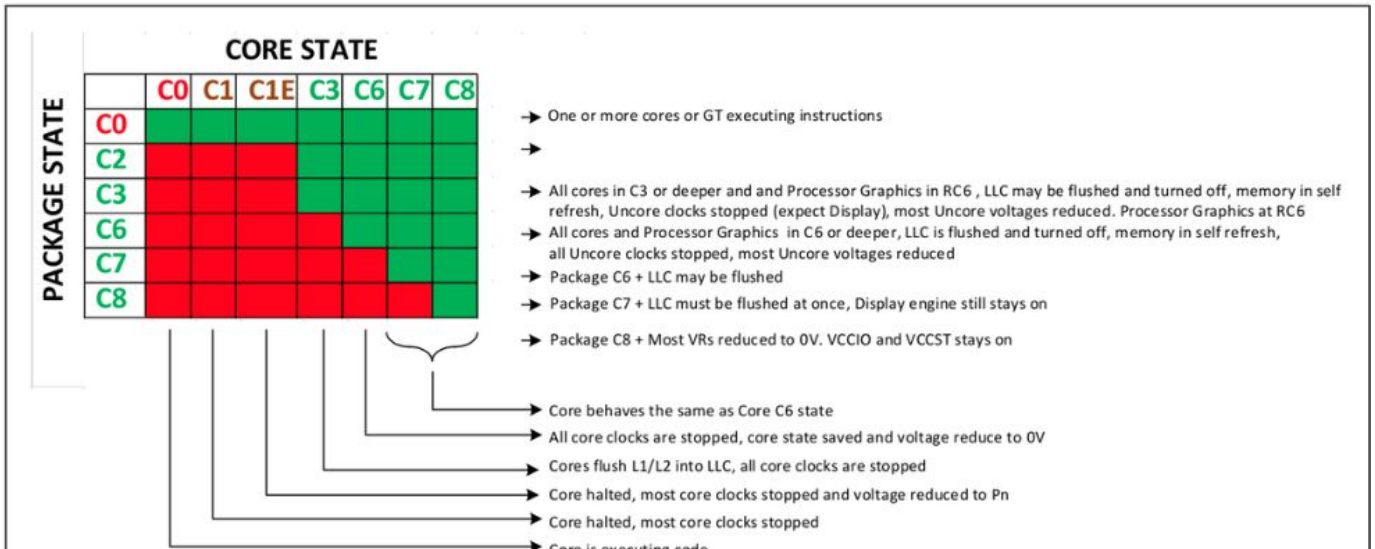
- Enabled PAPI 7.2 native events support on rocketlake
- DVFS study on Intel Rocketlake
- Fully Automated framework for ARM experiments

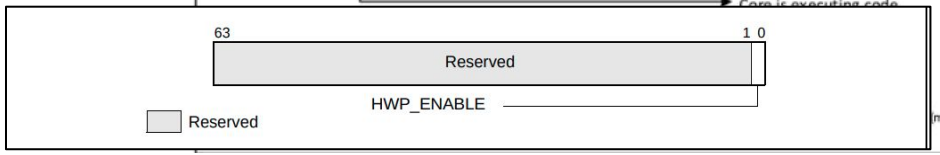


Important Tools & Interfaces

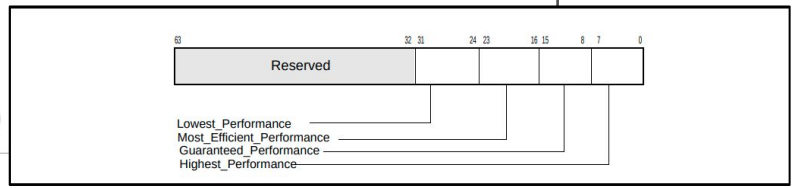
- Likwid-features
  - Disable Prefetchers HW_PREFETCHER,CL_PREFETCHER,DCU_PREFETCHER,IP_PREFETCHER
- Cpupower/Cpufrequtils
  - To change core power governor and policy
- Msr-tools
  - read/write to model specific registers as per there address & design
- Performance monitoring
  - Perf : cli tool to get various HW stats(eg LLC miss, CPU-Cycle Stalls) for an application
  - PAPI: Provide library support to instrument the code.
  - Enabled native events support on Rocketlake for PAPI 7.2
- Lm-sensors & gpiozero package(ARM)
  - To read Core temperature
- Taskset & Isolcpus
  - To isolate physical cores from OS scheduling(except kernel interfence)
  - Set and retrieve CPU affinity of a running process
- PyBluez
  - To handle Bluetooth sockets and associated data transfer

An idea about C-states





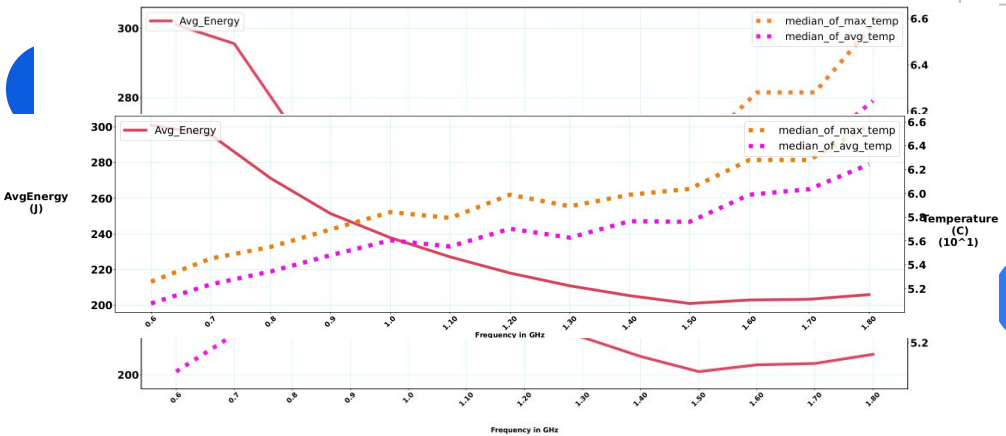
IA32_PM_ENABLE (770H)



IA32_HWP_CAPABILITIES (771H)

https://www.thomas-krenn.com/en/wiki/File:Processor-Package-and-Core-C-States.png

CONTENT

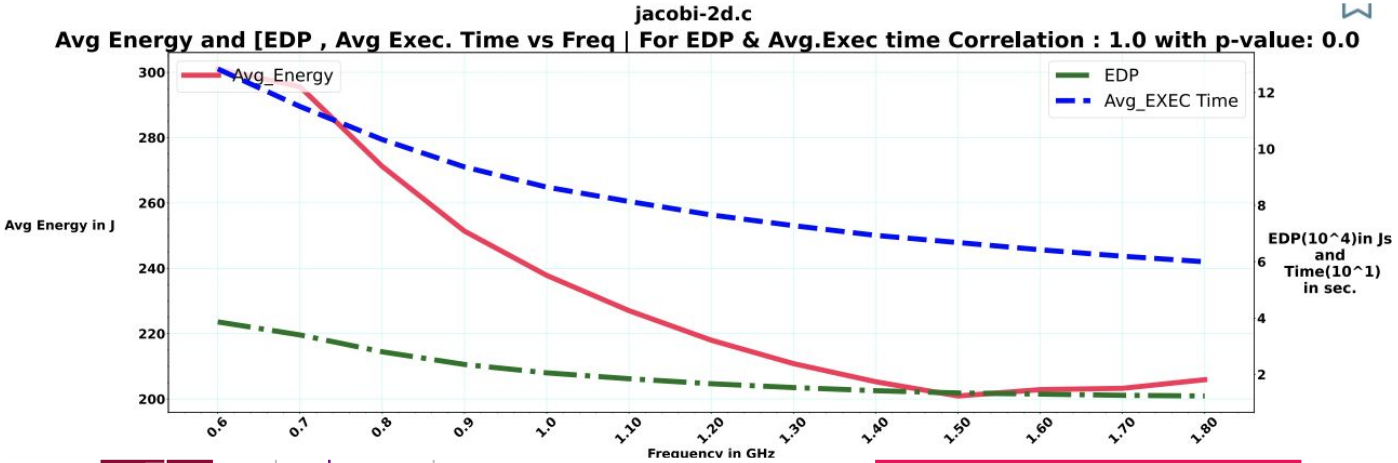


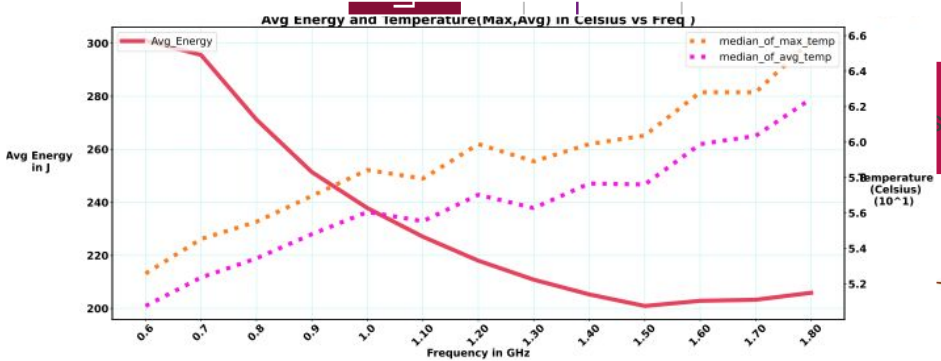
Lorem Ipsum

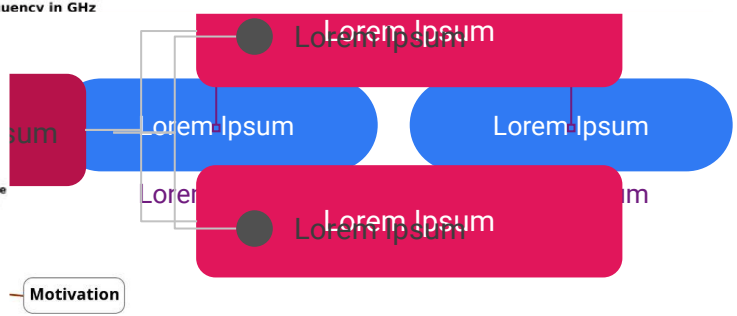
Lorem Ipsum

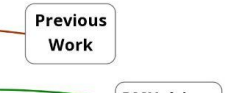
Lorem Ipsum

Lorem Ipsum

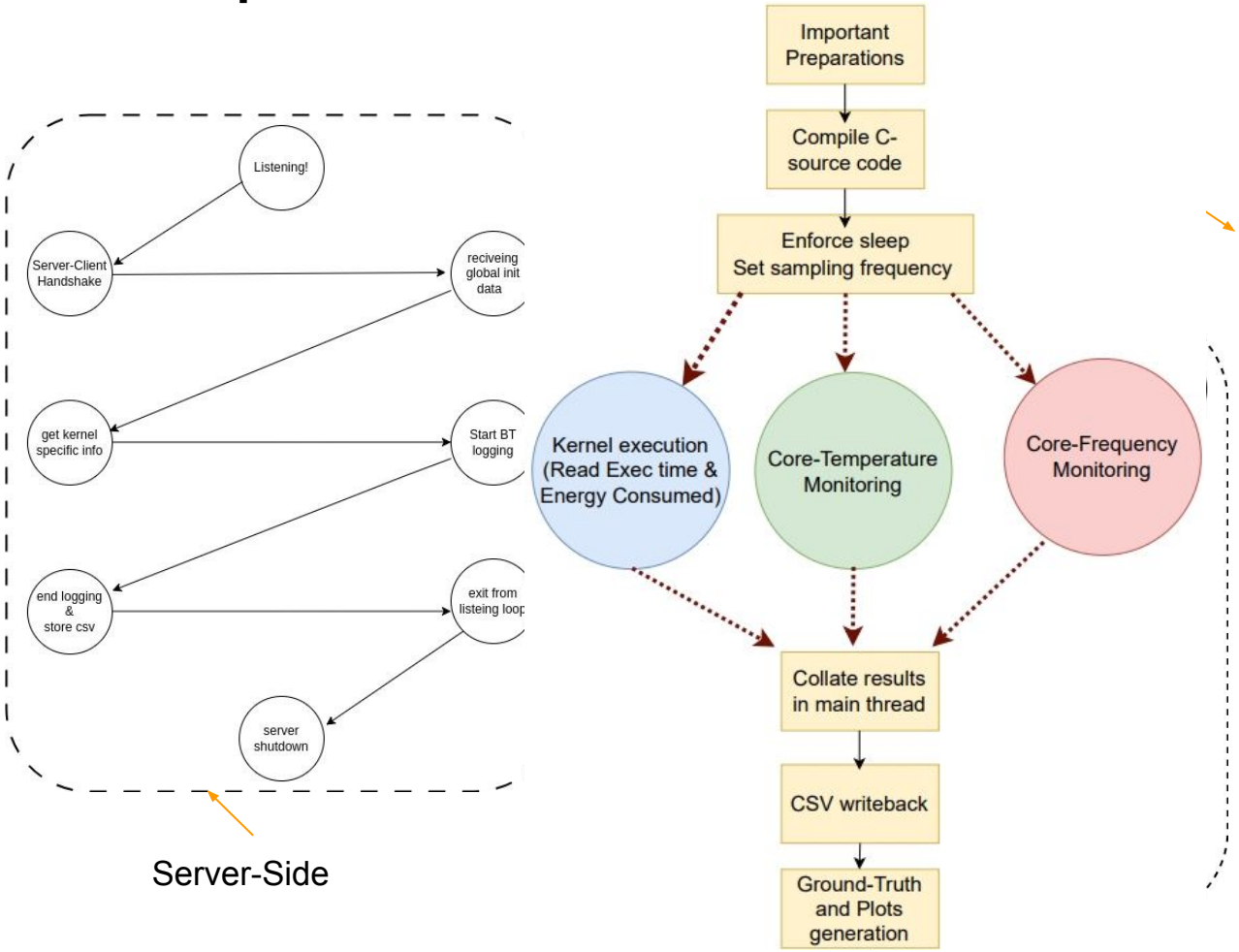

jacobi-2d.c
Avg Energy and [EDP , Avg Exec. Time vs Freq | For EDP & Avg.Exec time Correlation : 1.0 with p-value: 0.0
Avg_Energy
EDP
Avg_EXEC Time
Avg Energy in J
EDP(10^4)in Js and Time(10^1) in sec.
Frequency in GHz


Avg Energy and Temperature(Max,Avg) in Celsius vs Freq)
Avg_Energy
median_of_max_temp
median_of_avg_temp
Avg Energy in J
Temperature (Celsius) (10^1)
Frequency in GHz


Lorem Ipsum
Lorem Ipsum
Lorem Ipsum
Lorem Ipsum
Lorem Ipsum
Motivation


Previous Work

ARM Setup: Server-Client & Power Meter Communication Framework



- Check drop in assigned core freq. due to high temperature
- Result Collation
- Groundtruth generation (Min-EDP)
- Plots Generation

Intel P states imp links

https://vstinner.github.io/intel-cpus.html

https://www.kernel.org/doc/Documentation/cpu-

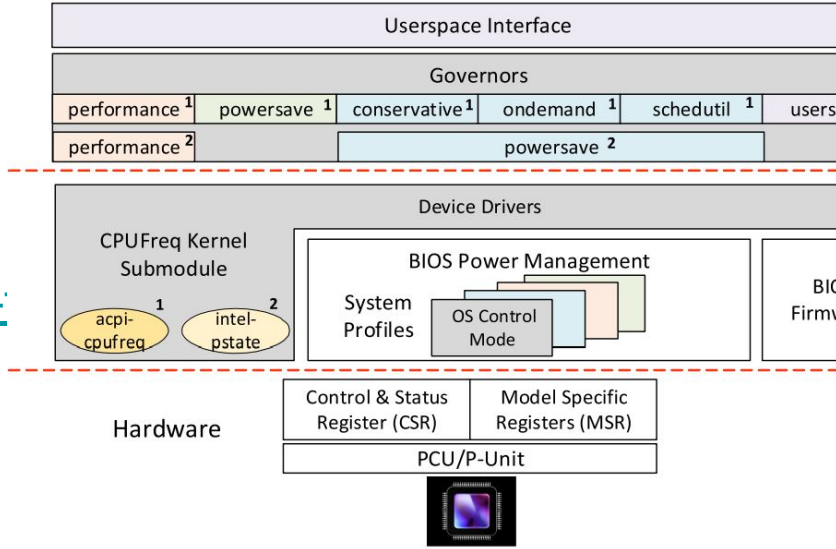
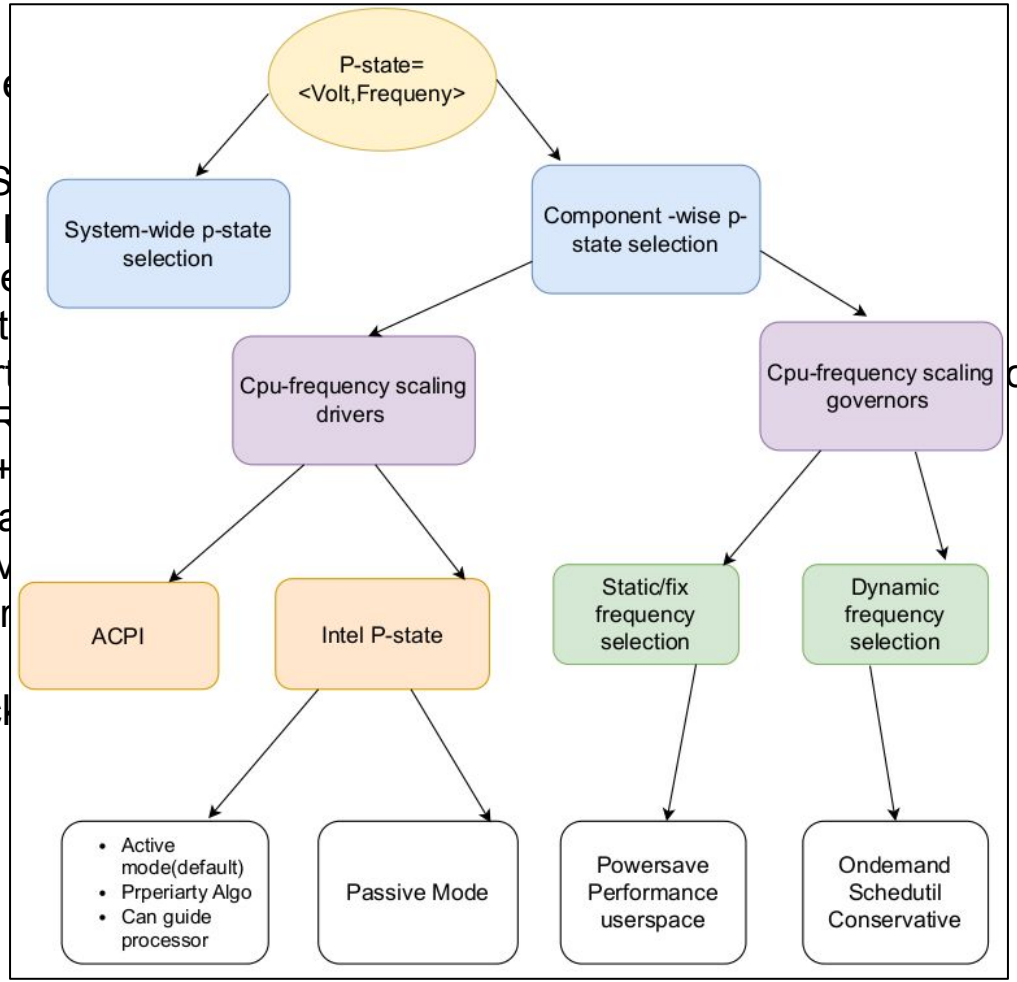


Fig. 2. Hierarchy of power management components.

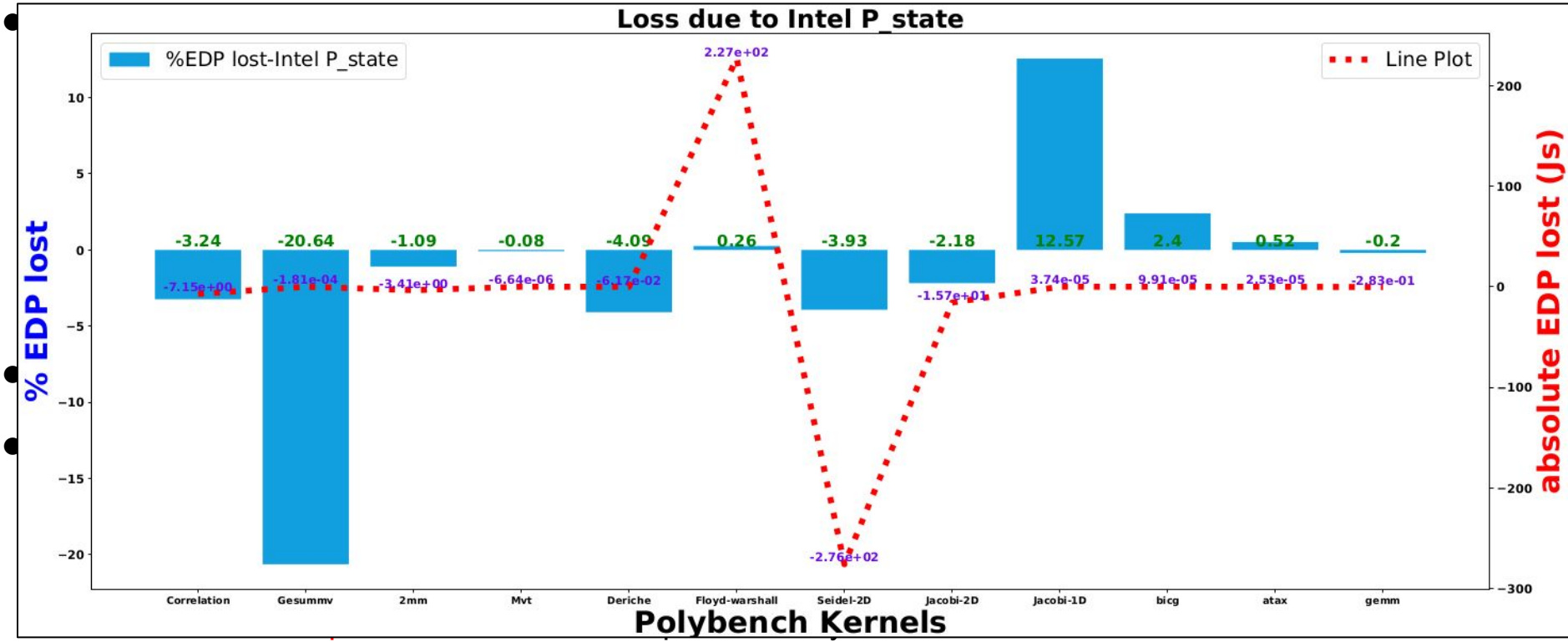
CONTENT

- Dynamic voltage
  - Motivation
  - Literature S
    - PMU
    - Fix the
  - Experiment
    - Import
    - Intel-F
    - ARM+
  - Problems fa
  - Key Observ
  - Possible dir
- IR2Vec Pip pack
- GeMS



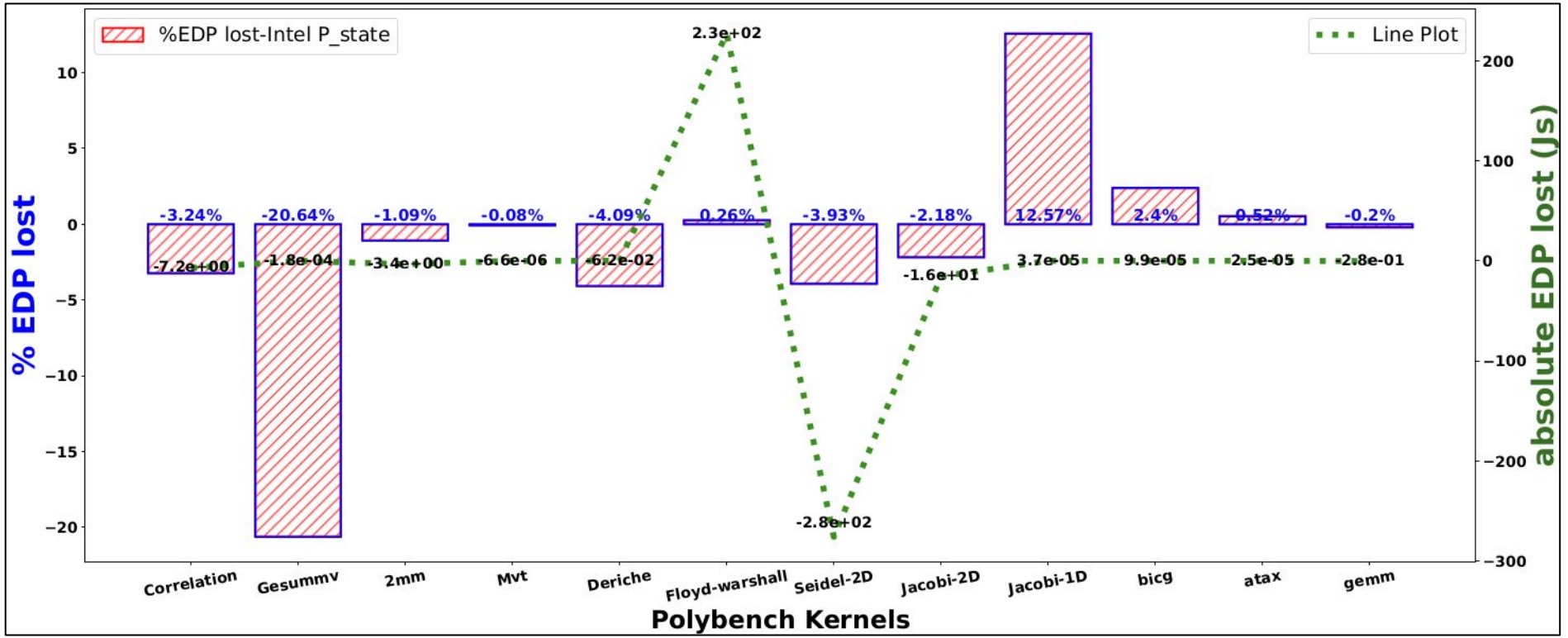
ction)

Clairvoyance [CGO ‘17]

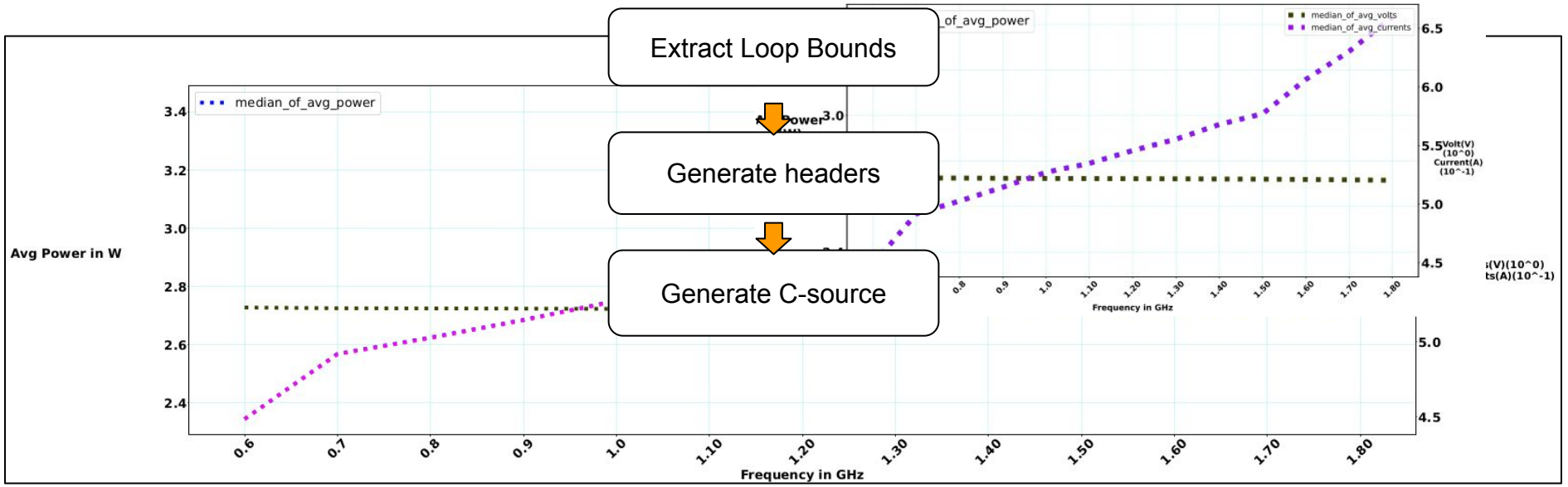


K. -A. Tran *et al*., "Clairvoyance: Look-ahead compile-time scheduling," *2017 IEEE/ACM International Symposium on Code Generation and Optimization (CGO)*, Austin, TX, USA, 2017

Results- Max & Avg Temperature vs Core frequency



Results-Power, Volts, Current vs Core frequency: ARM



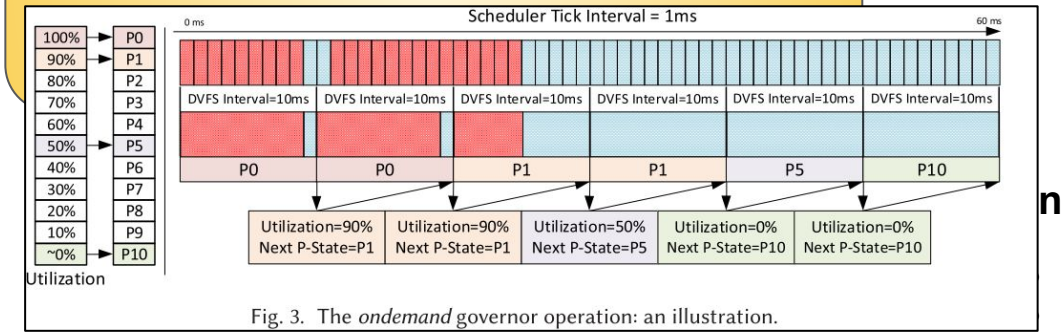


Fig. 3. The *ondemand* governor operation: an illustration.

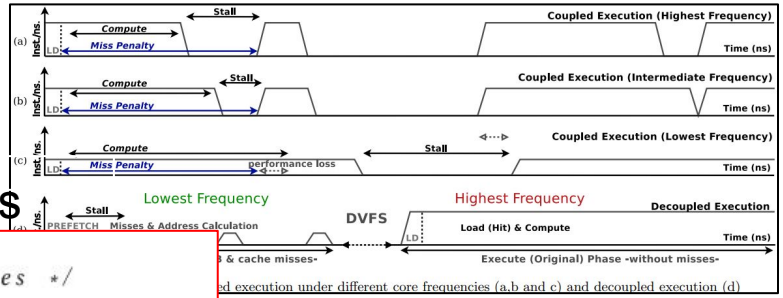
nimal overhead of context-switch

Isolcpus
Taskset

- Intel Hyper-Threading: off
- Speed Step: on/off

Related Works: HW and SW techniques

- **H/W:** Edinburg paper etc.
  - Intel P-state
  - ARM?
- **S/W:** DAE, Fix th



How to read MSRs

```
uint64_t msr_value;
/* Haswell: units of 61 microjoules */
/* MSR_PKG_ENERGY_STATUS is at address 0x611 */

double energy_units = pow(0.5, 14);
int fd = open("/dev/cpu/0/msr", O_RDONLY);
if (fd < 0) {
        perror("open");
        return -1;
}
if (pread(fd, &msr_value, 8, 0x611) < 0) {
        perror("pread");
        return -1;
}
double energy = msr_value * energy_units;
printf("%f\n", energy);
```

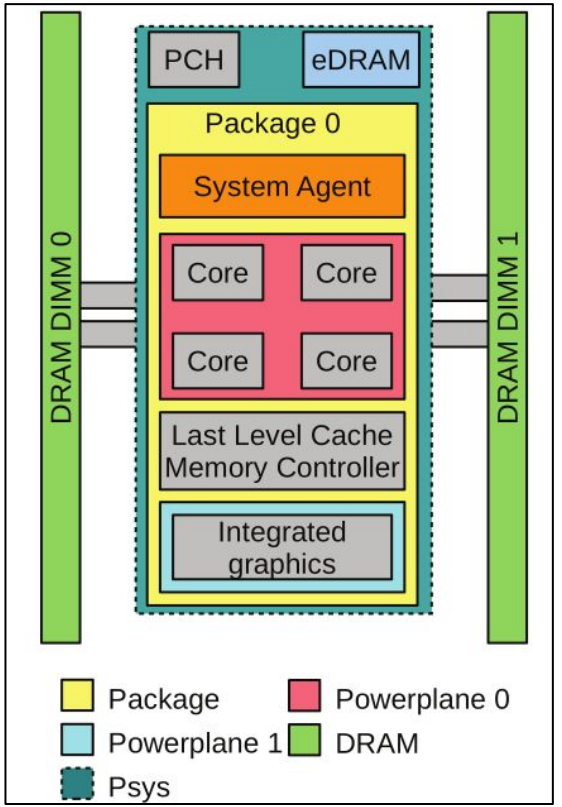
DAE architectures discuss ??
First paper in this series to tackle dvfs w
DAE{compiler-side}
Decoupling the access and execute pha
is better than running in coupled way . a
DAE under min-max freq selection or
optimal EDP corresponding to min edp
better than coupled

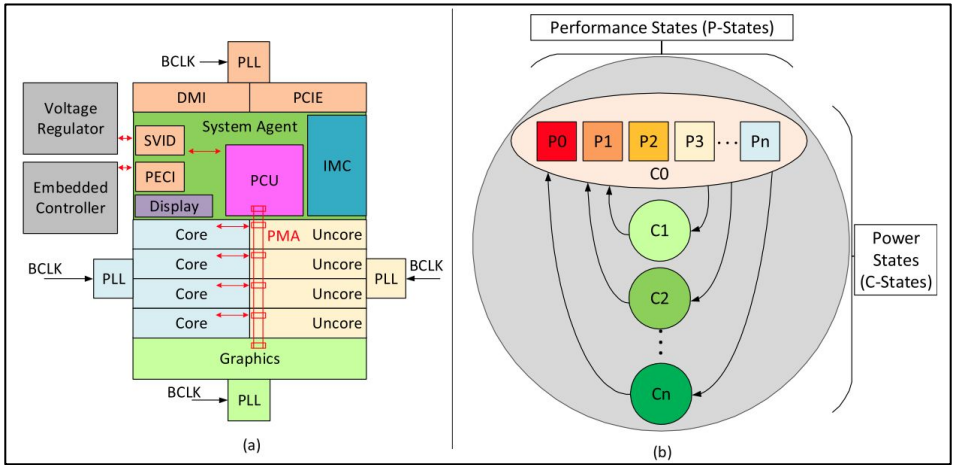
To explain the meaning of DAE

Turning off/on HW prefetcer actually eff
results
Lesser pipeline stalls in execute phase
Sometimes total stalls are lesser as
compared to CAE

Important Tools

- Tools for performance analysis
  - Perf
  - PAPI
  - Likwid
  - CpuFreq and Cpupower
- Hardware interfaces
  - Intel P-state
  - Intel RAPL

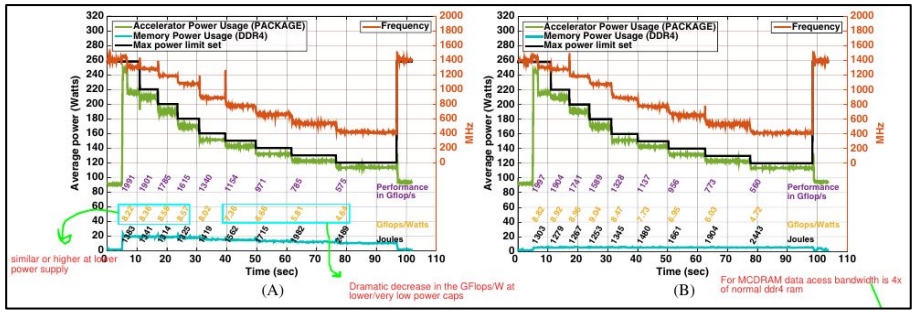




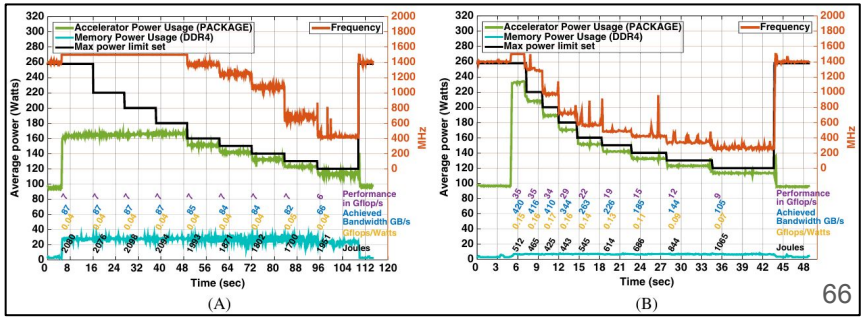
Energy Efficient Compilation

- Explain plots from jack dongra's paper
- This will help in explaining the aim/relevance of this problem
- Plots clearly highlights need of energy conservation methods for mem-bound kernels
- Also they show effect of core frequency accordingly

Compute Bound(dgemm)



Mem Bound(dgemm)



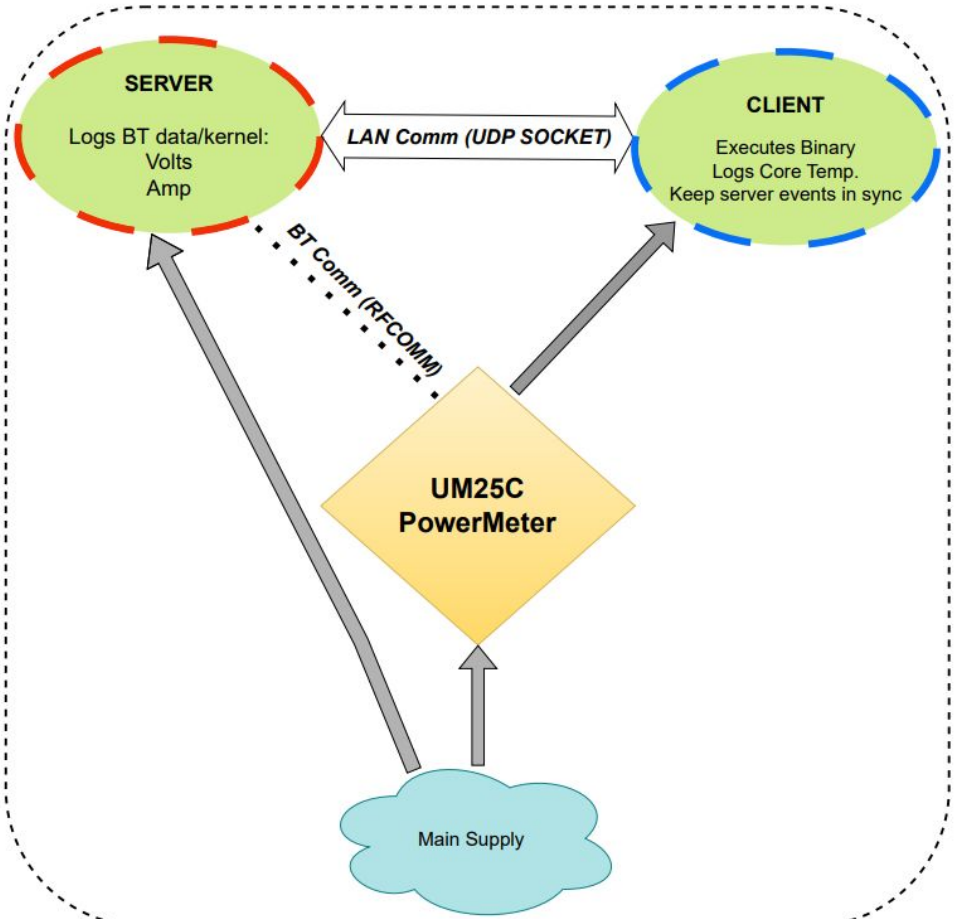
Experimental Setup

Workstation/Des

- Intel Rocketla
- Two desktops
  - 6 physical,
- Tools: cpupov -tools
- Made PAPI-7

Embedded Boar

- **2 ARM-64** (R
- Quad core Cc
- Tools: cpupov prefetchers)



Arm stubs setup done by Anilava I

Power Plane-0 Energy MSR Overflow Detection:

- Both Intel
  - Has e
  - Can m
  - MSR
- Rocketlake
- Rocketlake

amples per sec

```
5   262142 , 262142,1679819196
5   262143 , 262143,1679819196
    262143 , 262143,1679819196
    0 , 0,1679819196
    0 , 0,1679819196
```

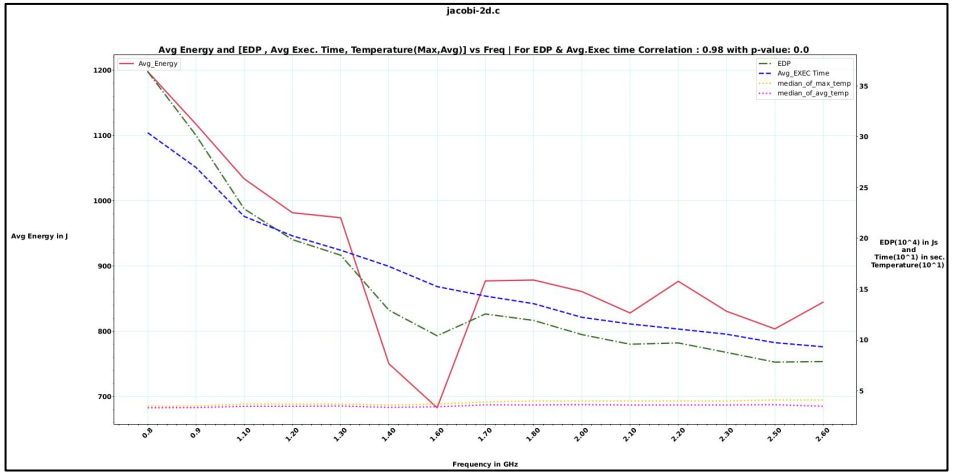
me stamps

```
262143,262143,1679833190
262143,262143,1679833190
0,0,1679833190
0,0,1679833190
```

Can mention methodology used to capture this fact?

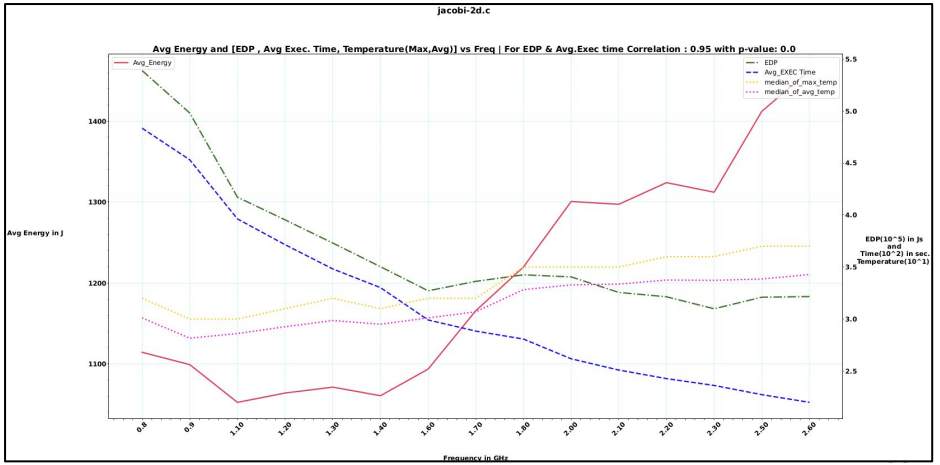
Experiments: Rocket Lake

- Two same configuration machines are not comparable for energy measurements



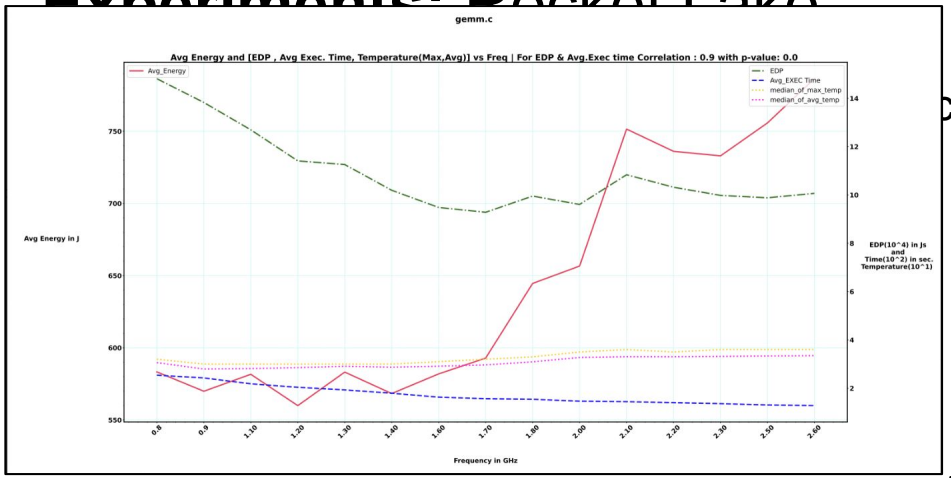
Dvfstest1

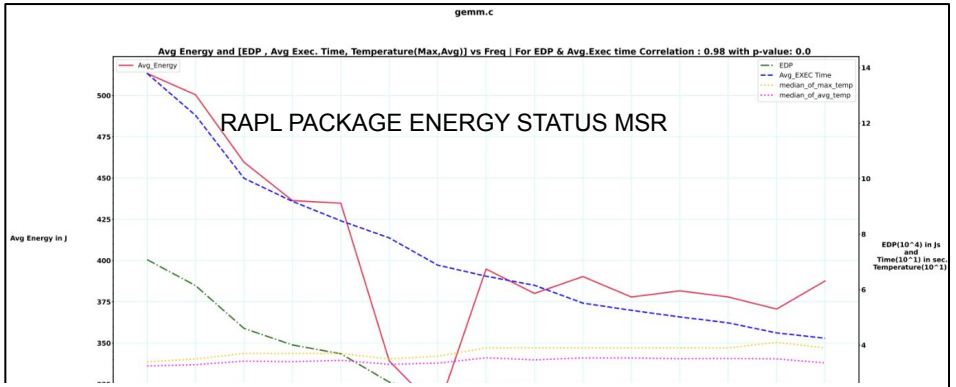
Dvfstest2



Experiments: Rocket Lake

c} DVFS techniques



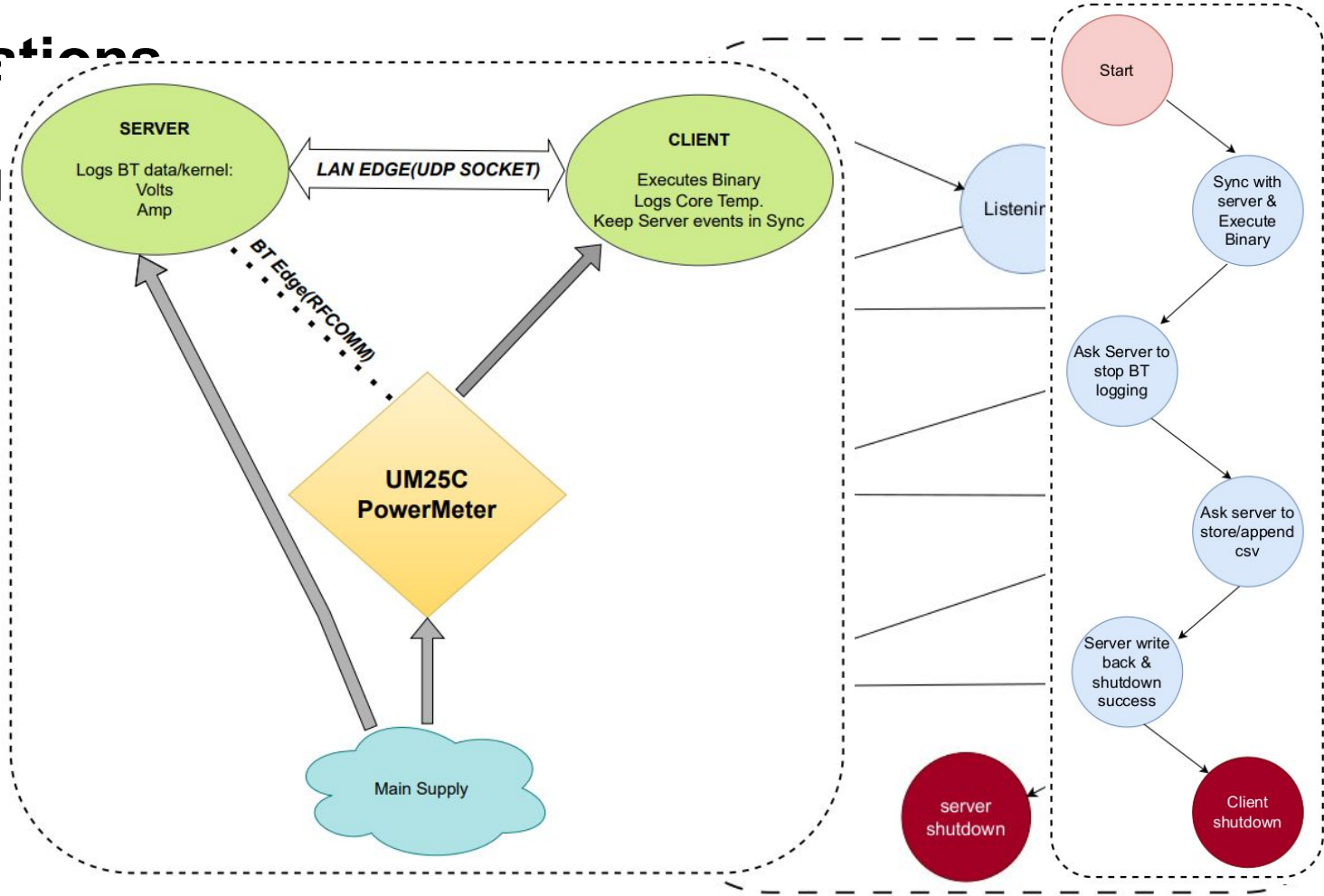


Precautions taken for experiments

- Race to Halt, Intel Speed Shift, Thermal Efficiency Boost disabled
- Enforcing Sleep to maintain similar Temp.
- Better Schedule {Interleaving of Kernel Execution}
- Data Cache Flushing
- IsolCpus{kernel} & taskset
- Disable H/W prefetchers
- Kernel-execution, sampling [Core-Freq ,Core-temp] as standalone threads
- Low sampling frequency to keep core-utilization low

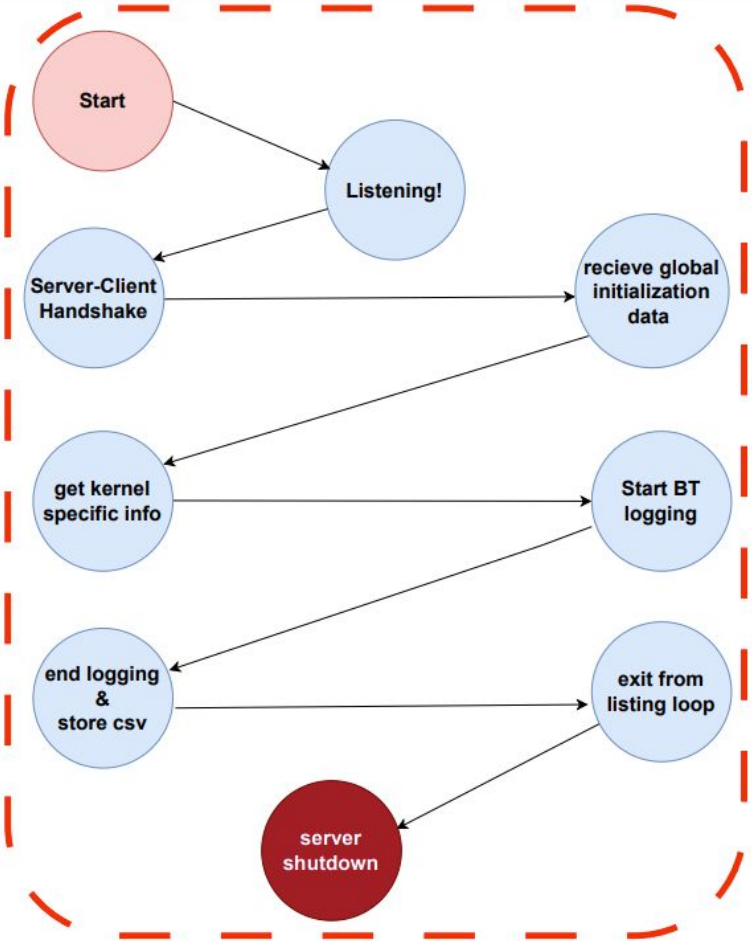
Key Observations

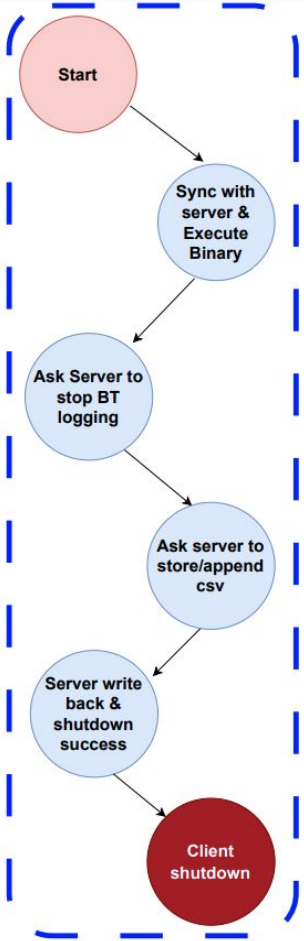
- DVFS modell
  -



Pos

-
-


Start
Listening!
Server-Client Handshake
recieve global initialization data
get kernel specific info
Start BT logging
end logging & store csv
exit from listing loop
server shutdown


Start
Sync with server & Execute Binary
Ask Server to stop BT logging
Ask server to store/append csv
Server write back & shutdown success
Client shutdown

IR2Vec as a Pip-installable **Python** Package

Contributions & Future Work

- DVFS using compilers and hardware techniques
  - Related work: H/W, S/W and related power optimizations
  - Experimental Setup:
    - Intel
    - ARM
  - MSR overflow detection
  - Core-temperature relation with Core-frequency
  - Study of Intel P-state w.r.t static *dvfs* techniques
- IR2Vec as a pip installable package
- GeMS: Presented in IMPACT 2023
- Future Work
  - Exploration of techniques apart from DVFS towards Energy optimizations

PolyBench-Kernel	Optimal EDP Userspace (Js)	EDP HWP Enabled (Js)
Jacobi-2d	216727.53	216190.41
Seidel-2d	641133.54	624895.51
gemm	60970.99	60644.51
atax	0.0087	0.0086
bicg	0.013	0.014
jacobi-1d	0.0046	0.0044

THANKS

MONTH/TASK	JAN	FEB	MARCH	APRIL	MAY
Literacy Survey	CARTAD, ROOFLINE MODEL	CARM, INTEL ADVISOR	RAPL IN ACTION, PMU Driven EVENTS	ENERGY AWARE ROOFLINE, COPPer	
Benchmark	Polybench	Polybench	Polybench	Polybench	mibench
Architecture	Intel I5 11th Gen	Intel I5 11th Gen	Intel I5 11th Gen	ARM Cortex A72	ARM Cortex A72
Custom Tools	Intel Setup	Intel Experiments rounds	ARM SETUP	Overflow-detection,Temperature Affects etc.	LLVM Pass
Tools	ADVISOR,SDE,VTUNE	ADVISOR,SDE,VTUNE	PERF	PERF	PERF, PAPI
Results	Exploration beyond IPC, ROOFLINE+OI	Data collection,understanding, improved ways of experiments Measured current,voltage,Energy,EDP	ARM vs Intel	Temperature measurement & its effects . Understanding "joint play" of Arch. & Compiler Techn. Role of Power-Capping and specific "problem" on intel DVFS via "OI" .	Static OI estimation using FLops LLVM PASS & BULLSEYE, Issues with PAPI and perf, Discussion of modelling technique & accordingly studying few more papers

ARM (64-Raspberry-Pi4)

- Discuss Communication framework which is used to get the plots
- Maybe explain it using DFA

AGL Builds (Comparative Study)

Timeline