The 44th International Conference on Parallel Processing
(ICPP-2015)

Using Per-Loop CPU Clock Modulation for Energy Efficiency in OpenMP Applications

Wei Wang, University of Delaware
Allan Porterfield, RENaissance Computing Institute
John Cavazos, University of Delaware
Sridutt Bhalachandra, University of North Carolina at Chapel Hill

September 3, 2015

THE UNIVERSITY of NORTH CAROLINA at CHAPEL HILL

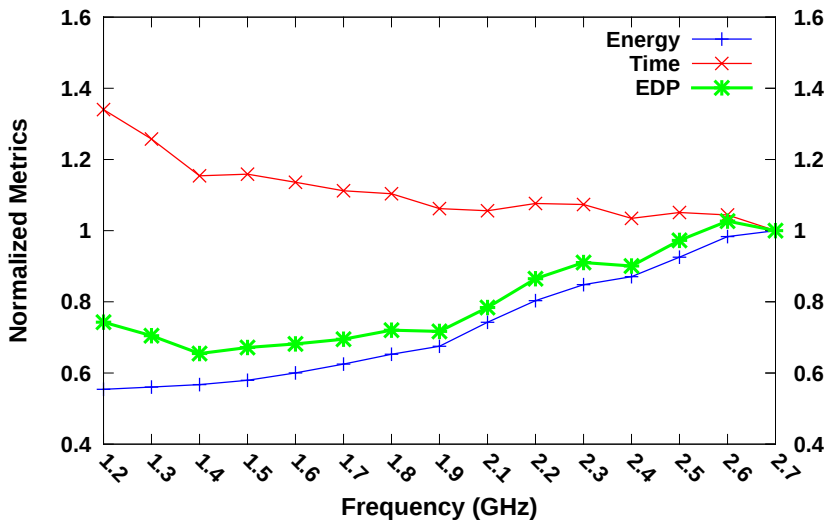
HPC Energy Optimization Challenge

Table: Performance, power, and energy efficiency of top/green500 and exascale systems

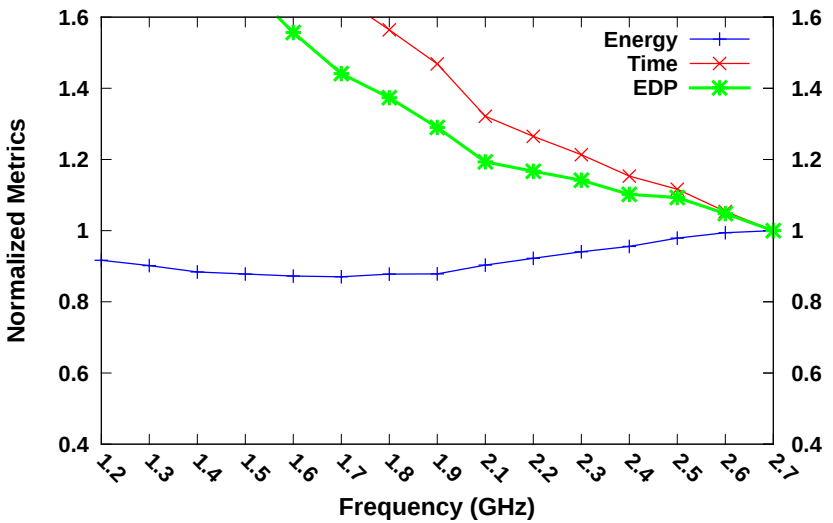
System Name	*Performance (TFLOP/s)*	*Power (KW)*	*GFLOPS/W*
Exascale System	1,000,000	20,000	50
MilkyWay-2	33,862.7	17,808	1.901
Titan	17,590.0	8,209	2.143
Sequoia	17,173.2	7,890	2.176
Shoubu	353.9	50.32	7.032
Suiren Blue	193.3	28.25	6.842
Suiren	202.6	32.59	6.217

Exascale computing requires more than 20$\times$ improvement in GFLOPS/Watts.

Motivation

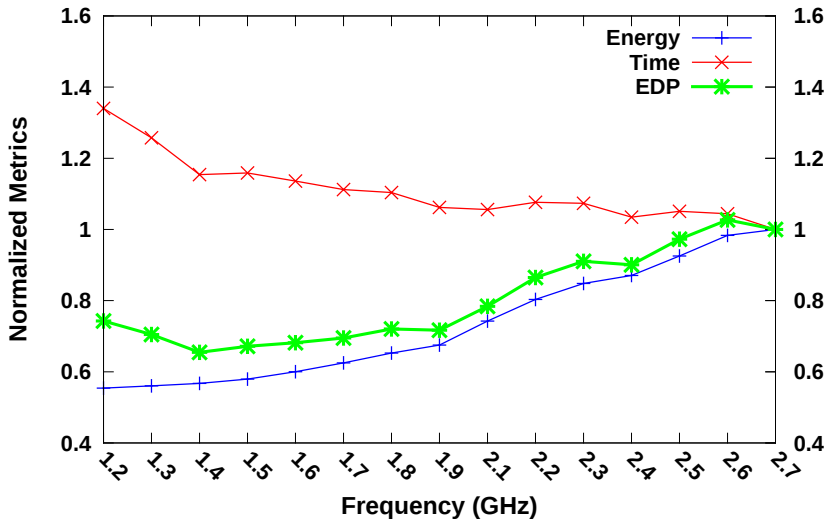


(a) LULESH-MemoryIntensiveLoop: Energy reduced and EDP lowered with low performance impact

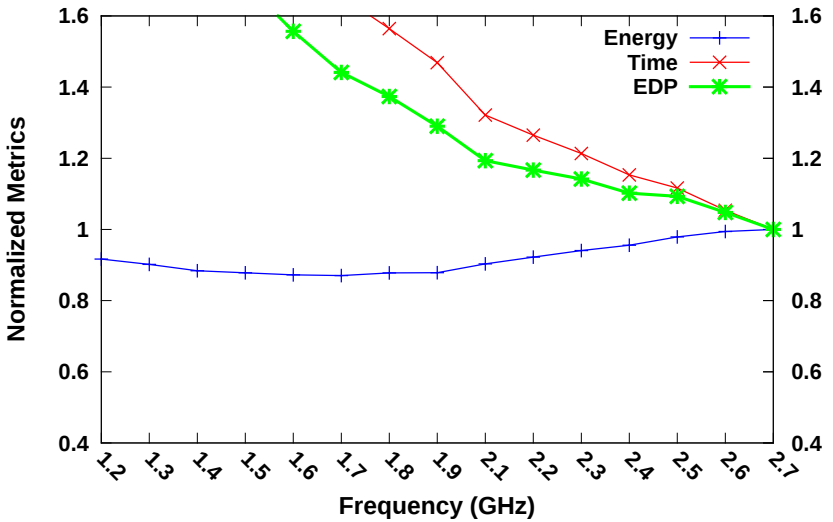


(b) LULESH-ComputeIntensiveLoop: Energy reduced with big performance slowdown and increased EDP

Motivation



(a) LULESH-MemoryIntensiveLoop: Energy reduced and EDP lowered with low performance impact



(b) LULESH-ComputeIntensiveLoop: Energy reduced with big performance slowdown and increased EDP

Loops of the same application prefer different frequencies

Issues

- DVFS mostly applied in coarse-grain cases
- Fine-grained (per-loop) energy control requires faster frequency transition techniques
- Could other power managament technique (e.g. Clock Modulation/Duty Cycle Modulation) help?

CPU Clock Modulation

- Write Specific Value to IA32_CLOCK_MODULATION (0x19a) MSR
- Modify /dev/cpu/cpu{0:15}/msr with root privilege
- Invoke wrmsr inline assembly from applications using added System Call

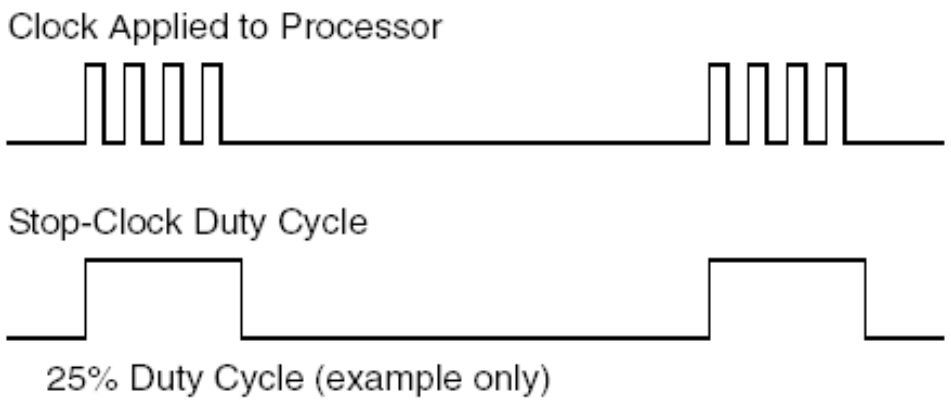


Figure: CPU Clock Modulation. Sample Modulation with 25% Duty Cycle. (Source: IA-32 Intel Architecture Software Developer's Manual, Volume 3: System Programming Guide)

Available Frequencies

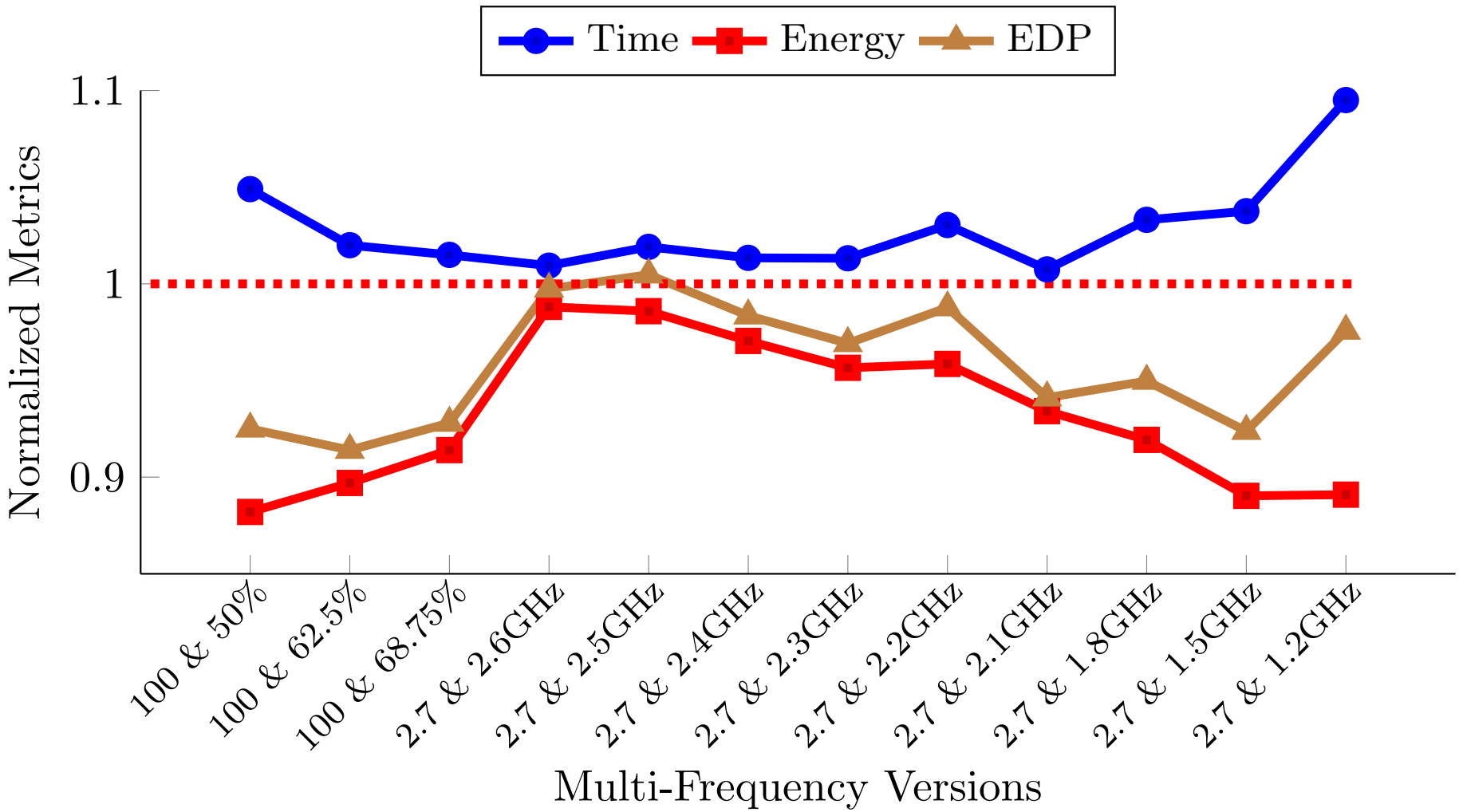
Duty Cycle Level	*Binary*	*Decimal*	*Hexadecimal*	*Effective Frequency*
1	10001B	17	11H	6.25%
2	10010B	18	12H	12.5%
3	10011B	19	13H	18.75%
4	10100B	20	14H	25%
5	10101B	21	15H	31.25%
6	10110B	22	16H	37.5%
7	10011B	23	17H	43.75%
8	11000B	24	18H	50%
9	11001B	25	19H	56.25%
10	11010B	26	1AH	63.5%
11	11011B	27	1BH	69.75%
12	11100B	28	1CH	75%
13	11101B	29	1DH	81.25%
14	11110B	30	1EH	87.5%
15	11111B	31	1FH	93.75%
16	00000B	0	00H	100%

Multi-frequency Execution of Multi-loop Applications

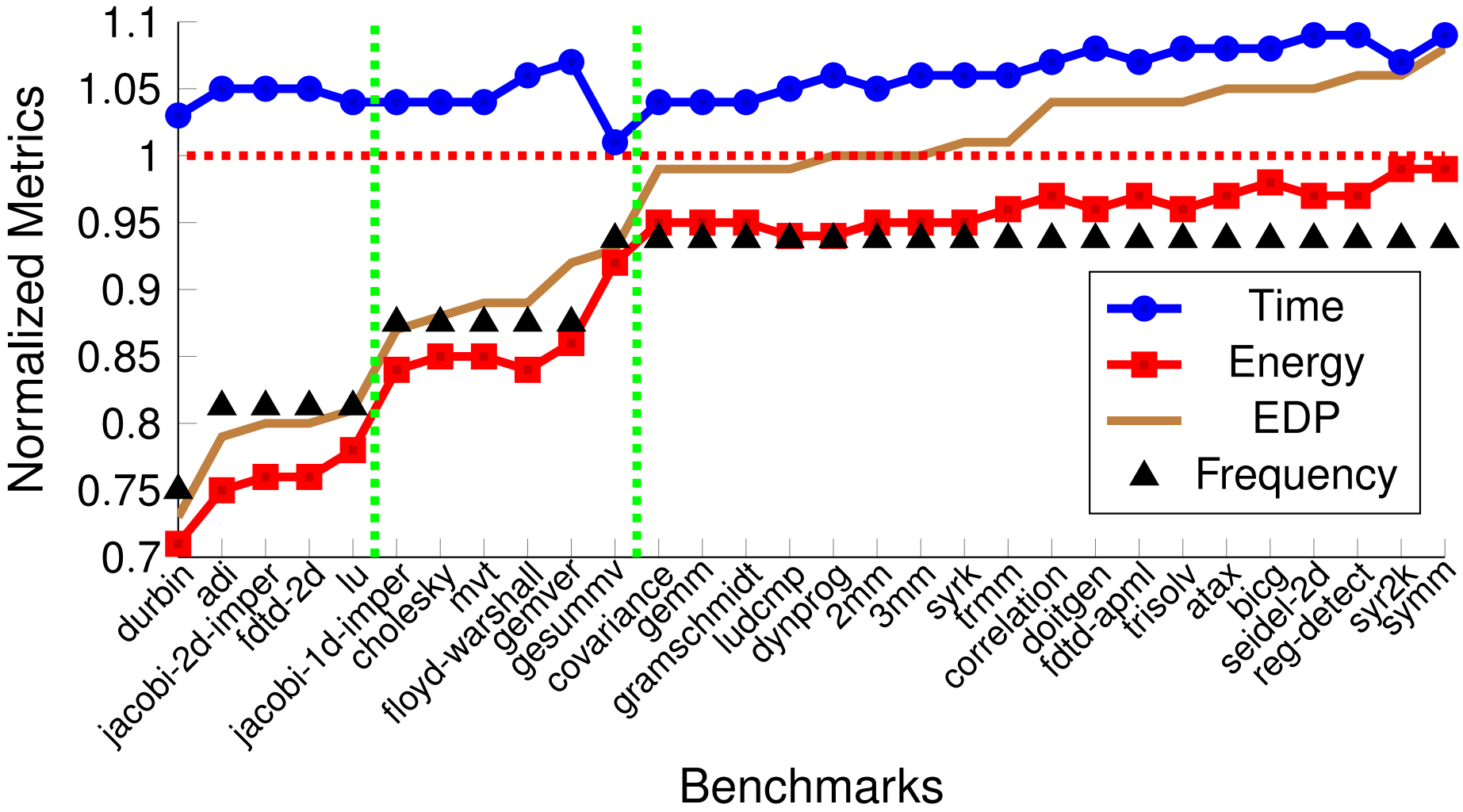
- Adding energy control APIs around loops
- Fine-grain loop regions require fast machine power-state transition to avoid overhead

```
while (condition) {
   ...
   setLowFrequency();
   for (i=0; i<N; i++) {
      ...
   }
   resetFrequency();

   for (j=0; j<N; j++) {
      ...
   }
}
```

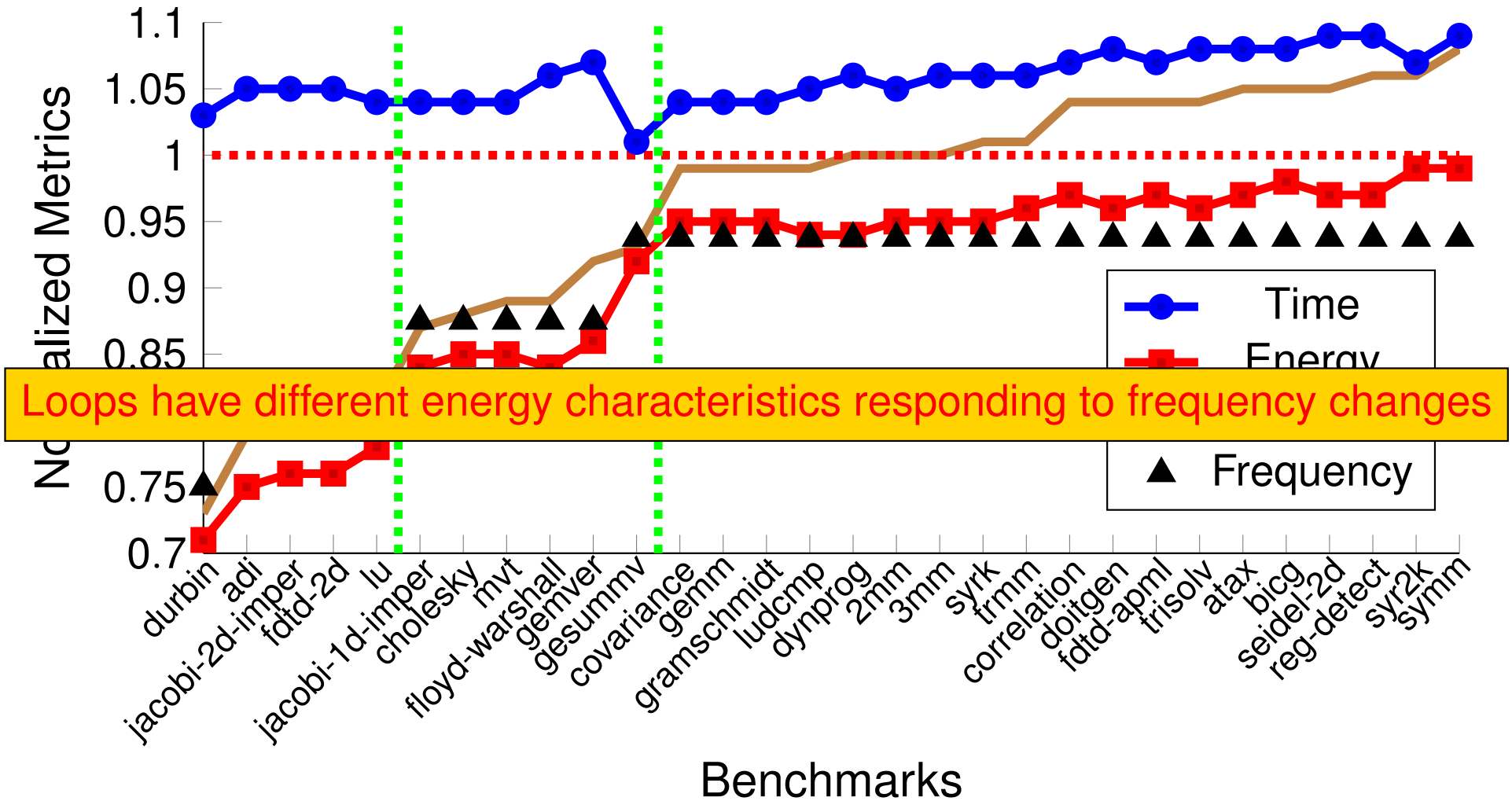
DVFS vs. Clock Modulation (Entire LULESH App)



Polybench Loops



Polybench Loops



Loops have different energy characteristics responding to frequency changes

Multi-frequency Execution: LULESH Results

Table: Comparison of execution time, energy consumption, and EDP for `LULESH`

Version	*Duty Cycle Level*	*Time*	*Energy*	*EDP*
minT	100%	1.000	1.000	1.000
minE	56.25%	1.542	0.743	1.145
minEDP	81.25%	1.157	0.816	0.943
MultiFreq 1	100% & 50%	1.049	0.882	0.925
MultiFreq 2	100% & 62.5%	1.020	0.897	0.914
MultiFreq 3	100% & 68.75%	1.015	0.914	0.928

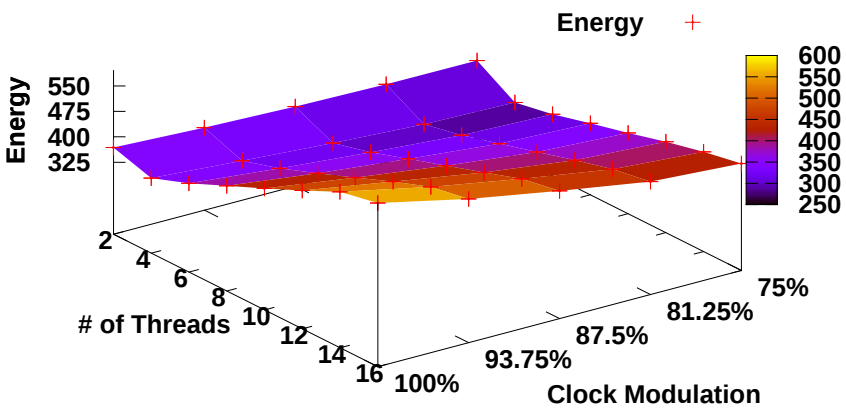
Multi-frequency Execution: miniFE Results

Table: Comparison of execution time, energy consumption, and EDP for `miniFE`

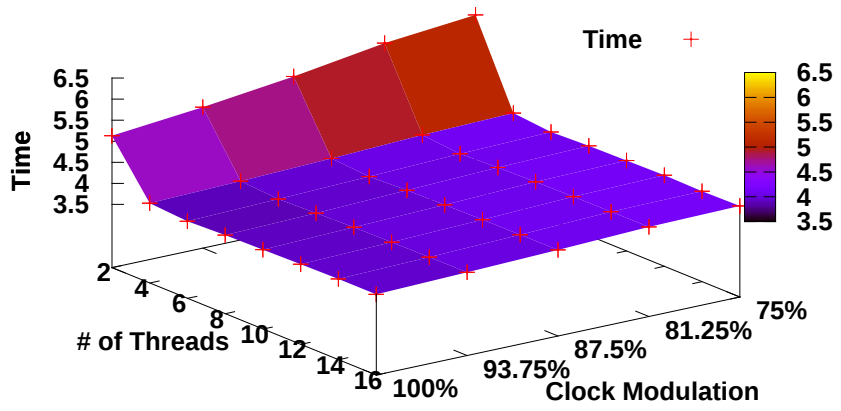
Version	*Duty Cycle Level*	*Time*	*Energy*	*EDP*
minT	100%	1.000	1.000	1.000
minE	62.5%	1.351	0.763	1.031
minEDP	81.25%	1.153	0.819	0.945
MultiFreq 1	100% & 81.25%	1.029	0.893	0.919
MultiFreq 2	100% & 87.5%	1.023	0.923	0.944
MultiFreq 3	100% & 93.75%	1.000	0.954	0.954

Clock Modulation with Concurrency Throttling

- Concurrency Throttling mitigates resource contention
- Clock Modulation reduces idle state power



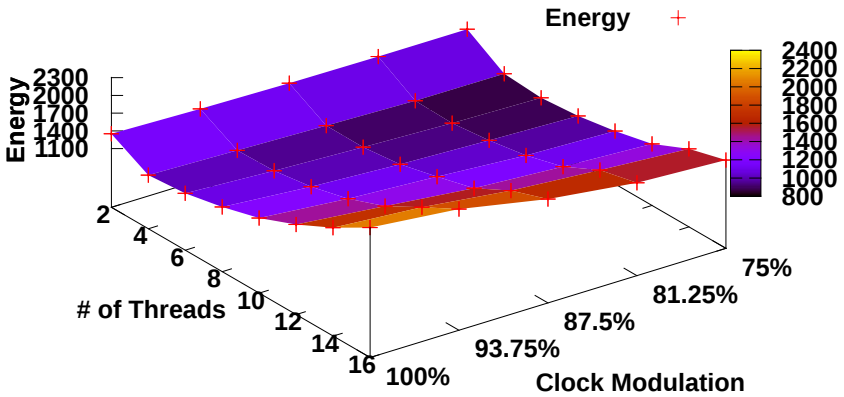
(a) Energy with concurrency throttling and clock modulation. Minimum occurs at (75%, 4)



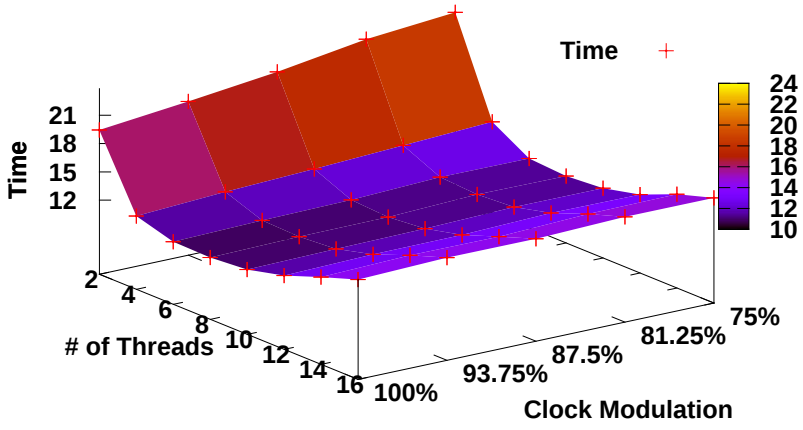
(b) Time with concurrency throttling and clock modulation. Minimum occurs at (100%, 6)

Figure: fdtd-2d Polybench

LULESH



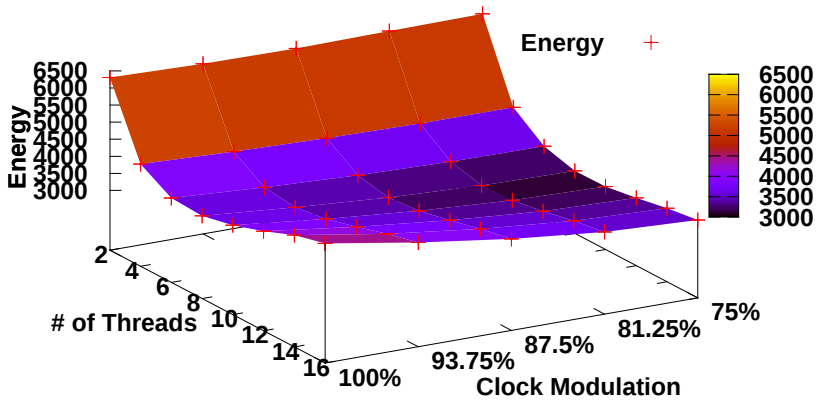
(a) Energy results. Minimum occurs at (75%, 6)



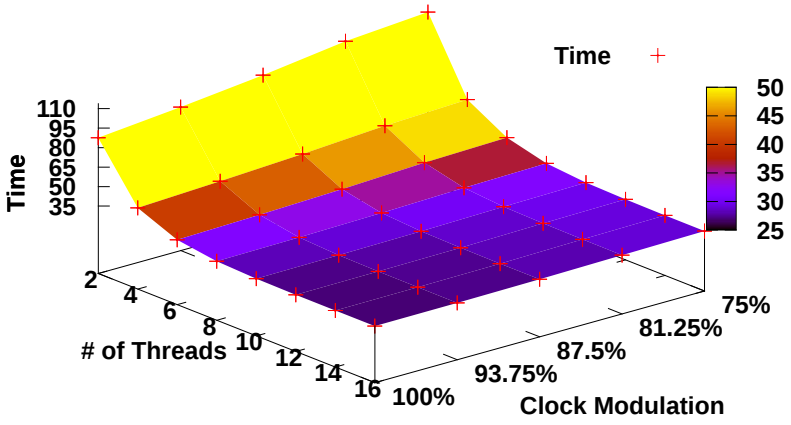
(b) Time results. Minimum occurs at (100%, 8)

Version	*# of Threads*	*Duty Cycle Level*	*Time*	*Energy*	*EDP*
Default	16	100%	1.00	1.00	1.00
CT	6	100%	0.92	0.94	0.87
CT+CM	6	75%	0.95	0.87	0.83

miniFE



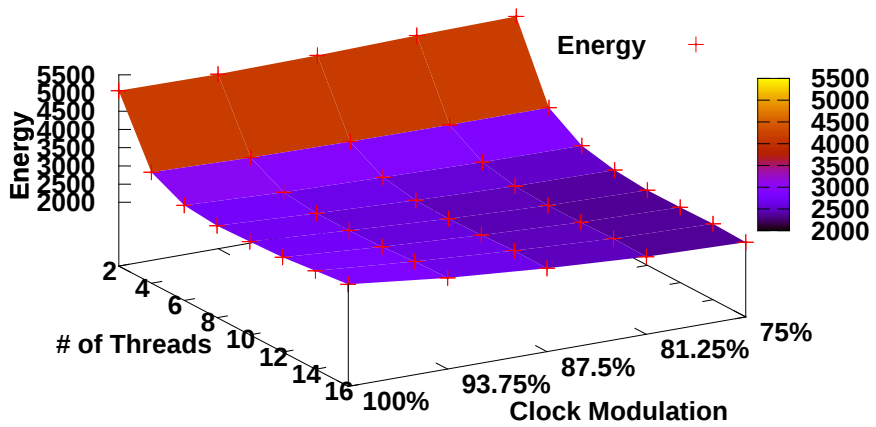
(a) Energy results. Minimum occurs at (75%, 8)



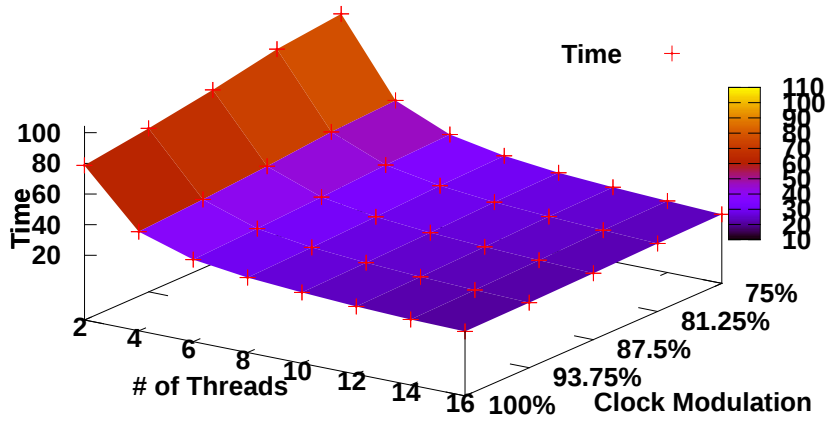
(b) Time results. Minimum occurs at (100%, 14)

Version	*# of Threads*	*Duty Cycle Level*	*Time*	*Energy*	*EDP*
Default	16	100%	1.00	1.00	1.00
CT	10	100%	1.02	0.86	0.88
CT+CM	10	81.25%	1.06	0.79	0.84

When Concurency Throttling is Not Beneficial



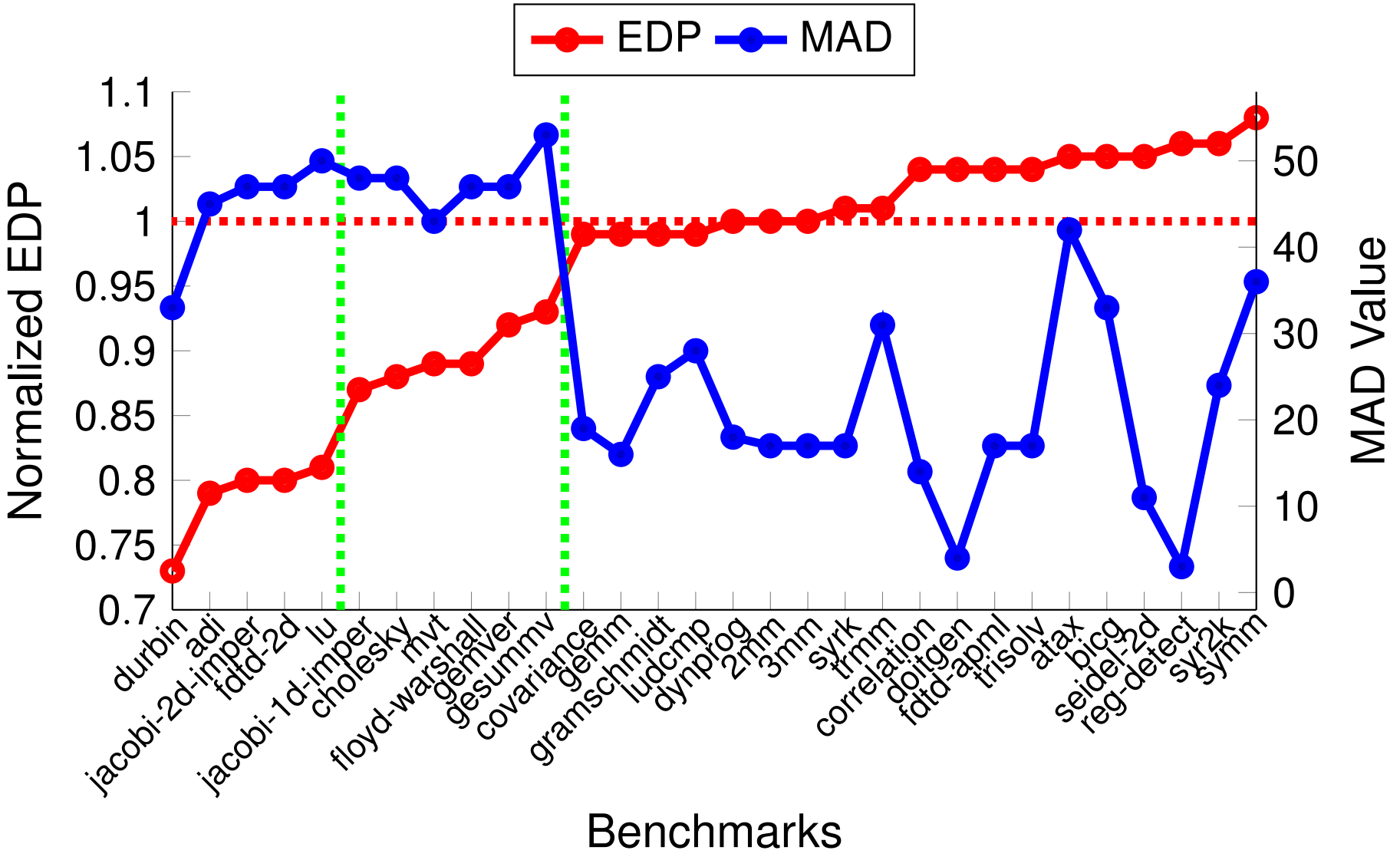
(a) Energy results. Minimum occurs at (75%, 16)



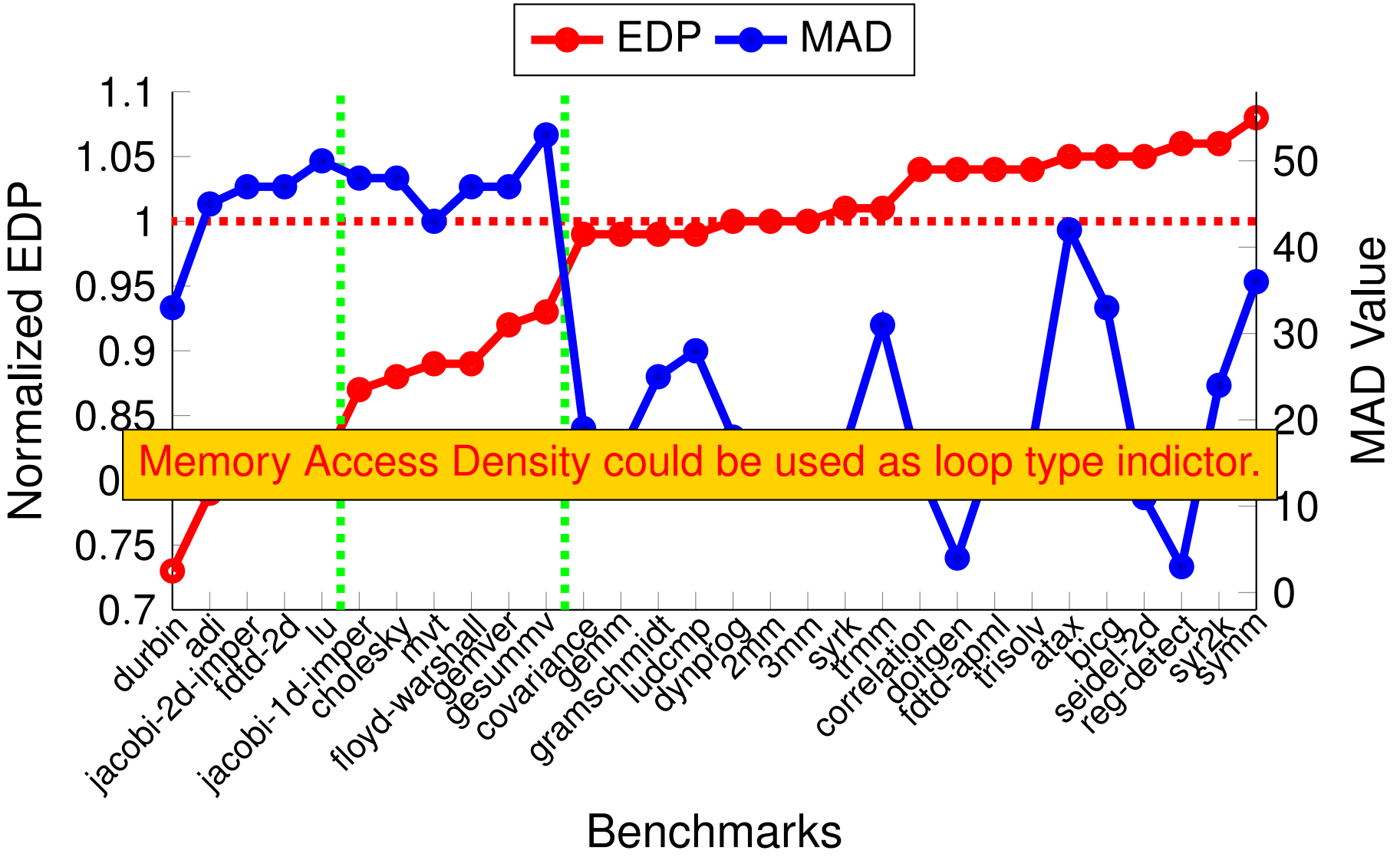
(b) Time results. Minimum occurs at (100%, 16)

Figure: `brdr2d` results when applying both concurrency throttling and clock modulation.

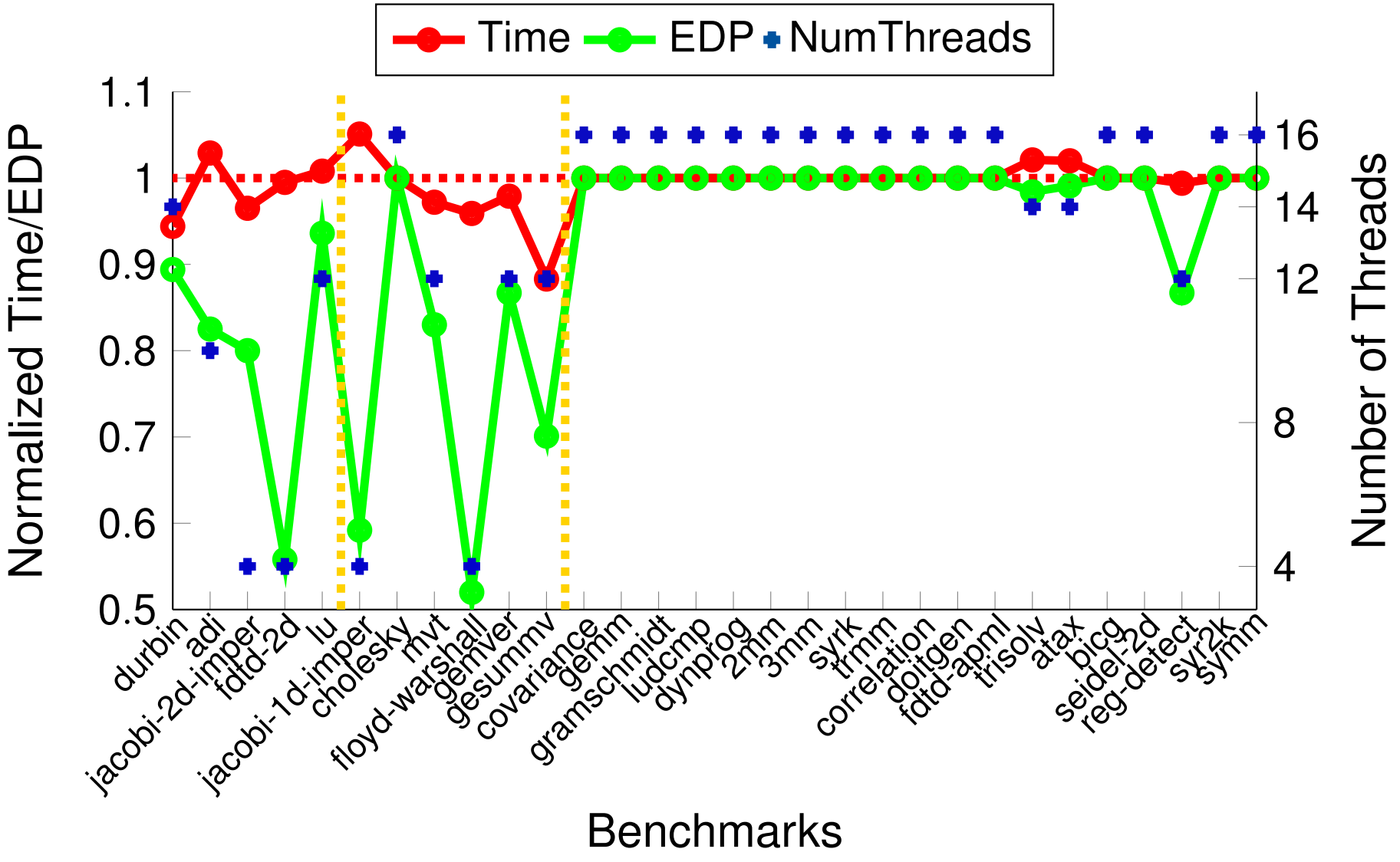
Memory Access Density vs. Three Types of Loops



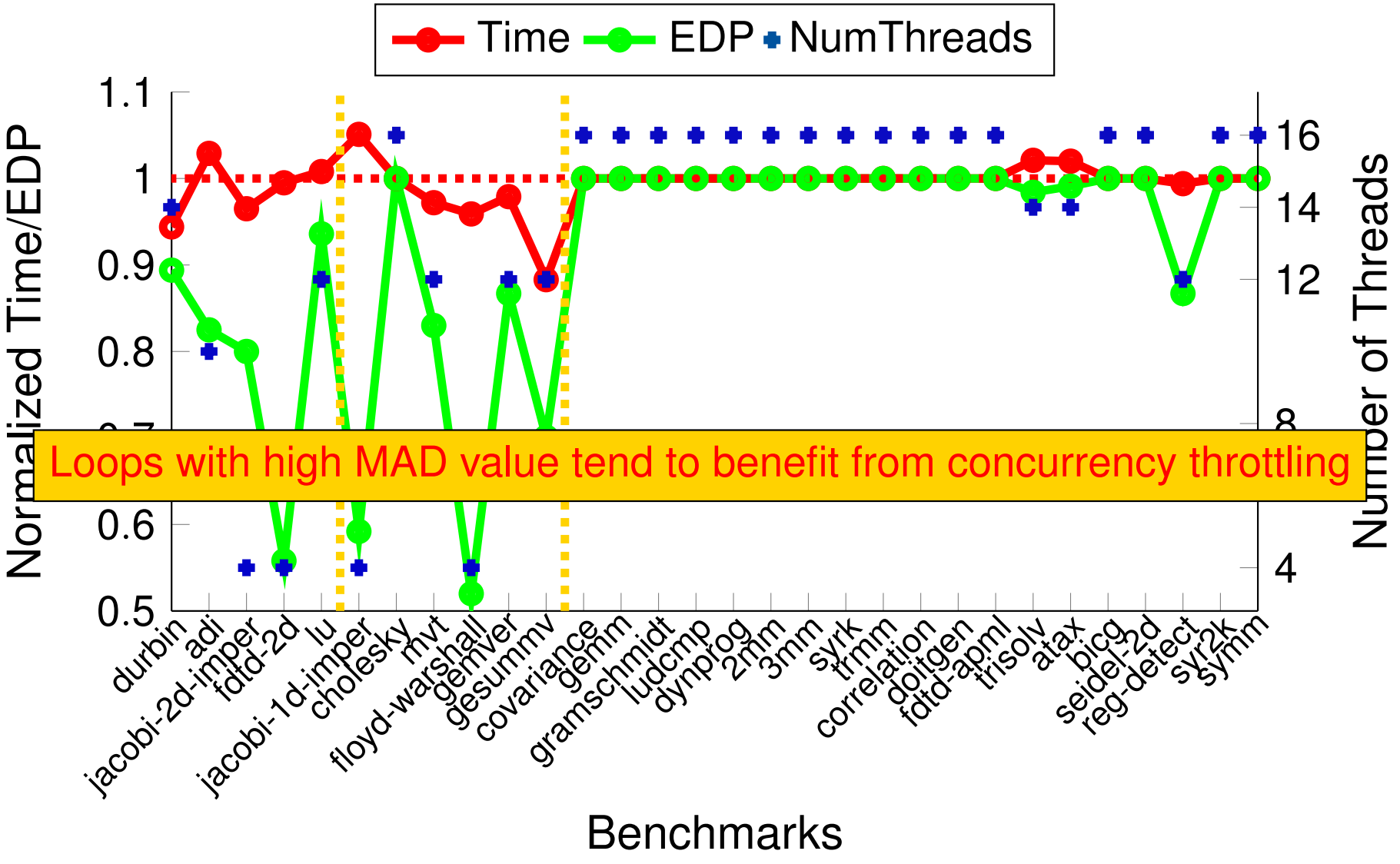
Memory Access Density vs. Three Types of Loops



Concurrency Throttling and Memory Access Density



Concurrency Throttling and Memory Access Density



Conclusion

1. Multi-frequency execution of OpenMP loops with Clock Modulation can achieve better energy efficiency
2. Concurrency throttling can be combined with Clock Modulation to save more energy

Acknowledgment

1. U.S. Department of Defense: ATPER project
2. National Science Foundation: Award Number 1218734

Q & A

Thanks!

Backup Slides: DVFS vs. DCM

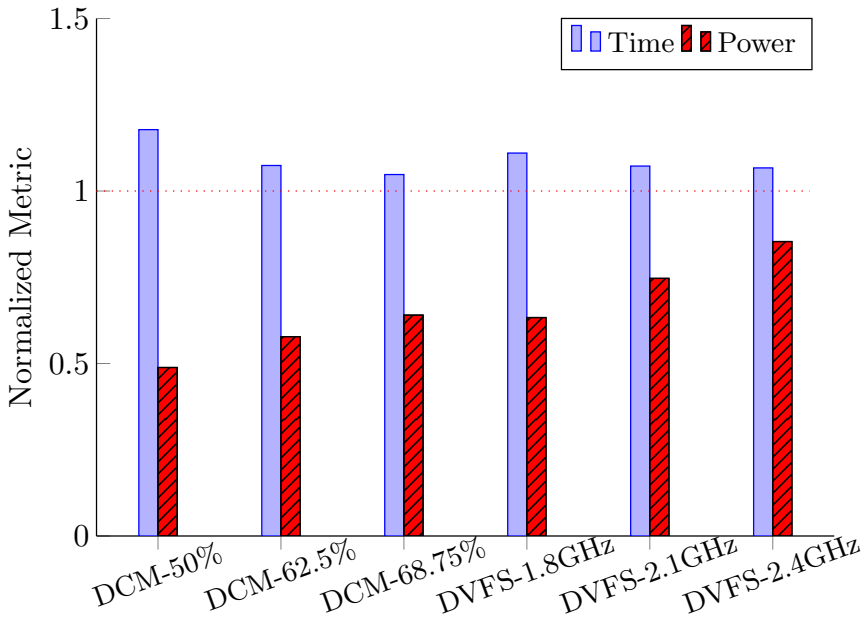
Measure and compare the execution time and power of Loop1 and Loop2 with and without energy control APIs.

```
while (condition)
{
 ...
 //Loop1
 MemLoop();

 //Loop2
 CompLoop();
 OtherLoops();
}
```

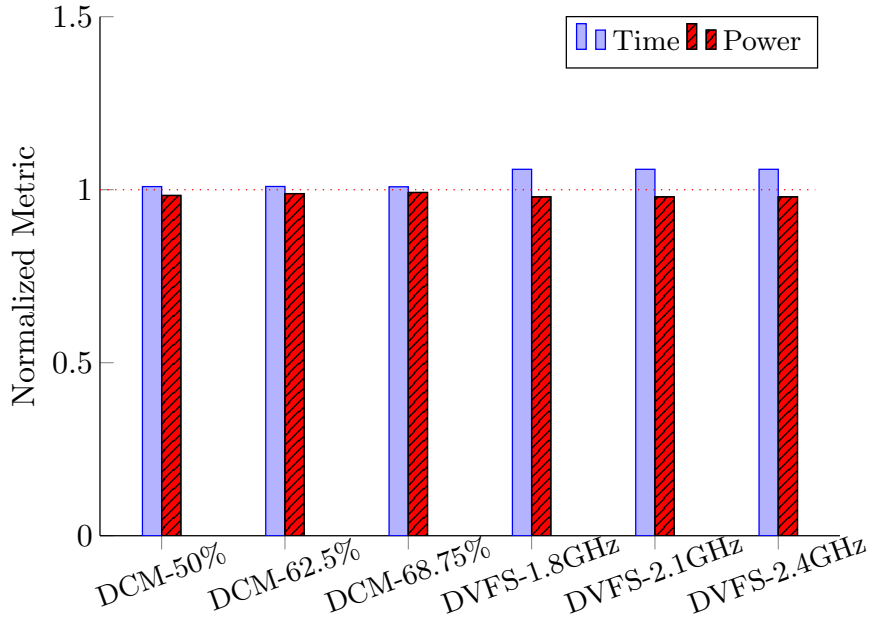
VS.

```
while (condition)
{
 ...

 setFrequency();

 MemLoop();

 resetFrequency();

 CompLoop();
 OtherLoops();
}
```

DVFS vs. Clock Modulation (Loop Analysis)

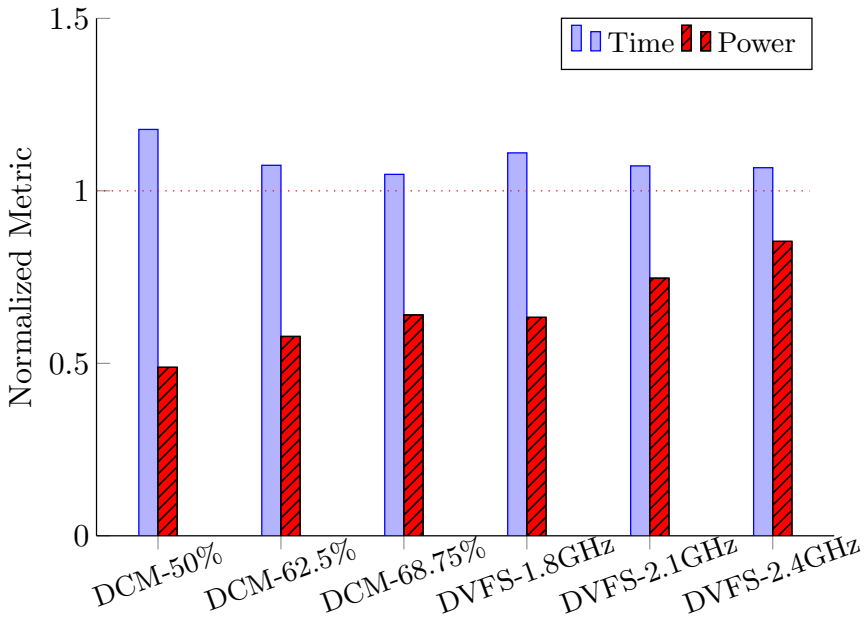


(a) LULESH-MemoryIntensiveLoop

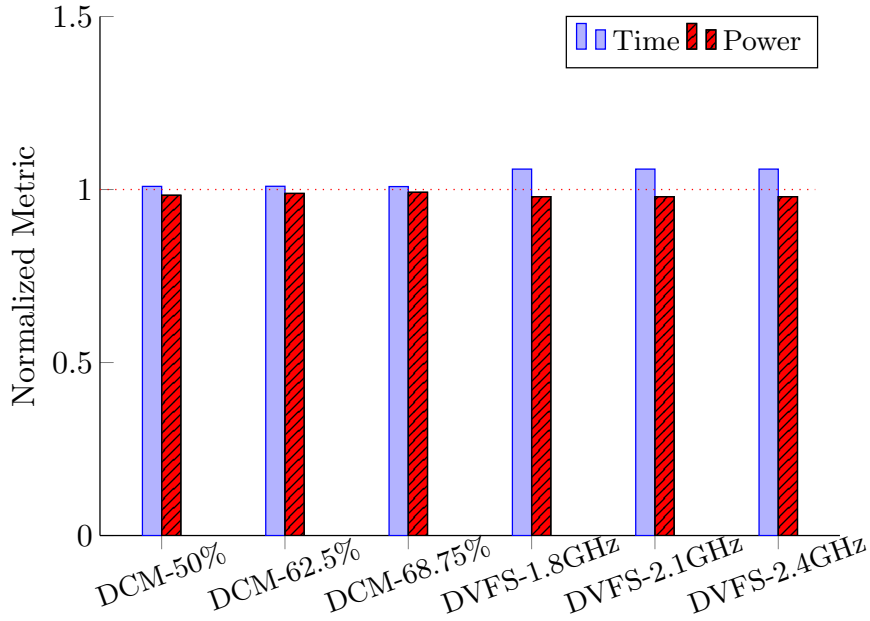


(b) LULESH-ComputeIntensiveLoop

DVFS vs. Clock Modulation (Loop Analysis)



(a) LULESH-MemoryIntensiveLoop



(b) LULESH-ComputeIntensiveLoop

Clock Modulation Performs Slightly Better Than DVFS

Benchmarks and Experimental Setup

1. Benchmarks
   - LULESH : Hydrodynamics
   - miniFE from Mantevo Project: implicit finite-element application
   - brdr2d: 2D Cardiac Wave Propagation Simulation
   - Polybench: 30 Computational Kernels
2. Hardware/Software Setup
   - Intel Xeon E5-2680 (Dual Socket, 8-core processor with 20MB LLC, 2.7GHz)
   - Linux 2.6.32 with ACPI and MSR modules
   - Intel ICC v14.0.2 with -O3