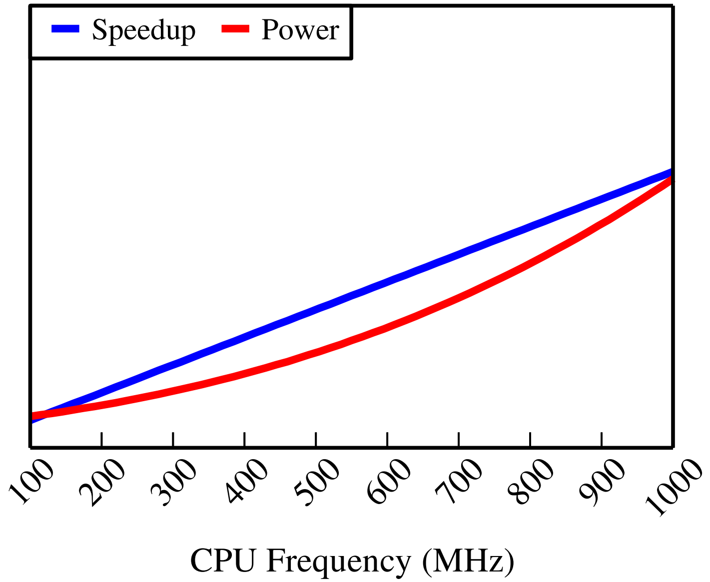
Energy-Performance Trade-offs on Energy-Constrained Devices with Multi-Component DVFS

Rizwana Begum, David Werner, Mark Hempstead
Drexel University

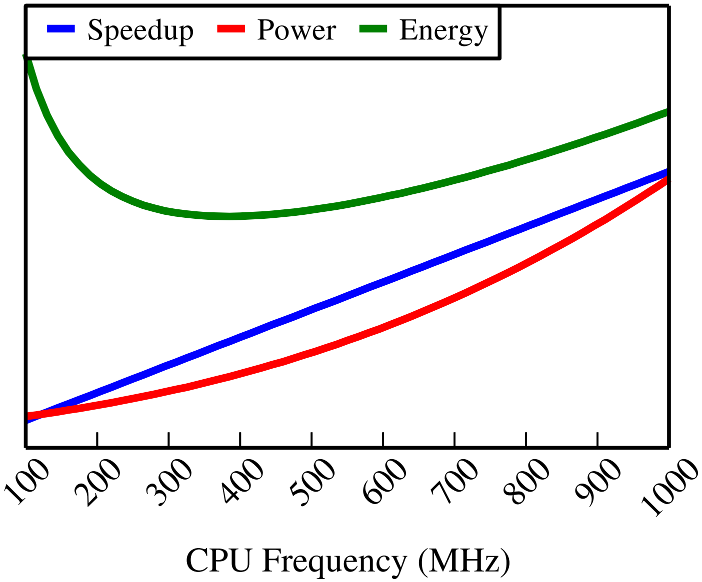
Guru Prasad, Geoffrey Challen
University at Buffalo

Oct 5, 2015

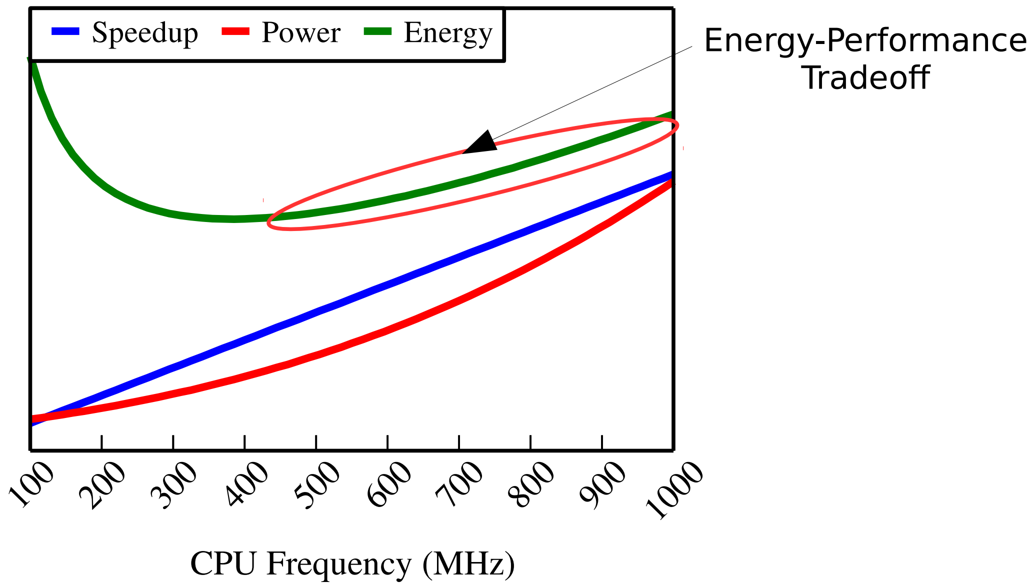
Dynamic Voltage and Frequency Scaling



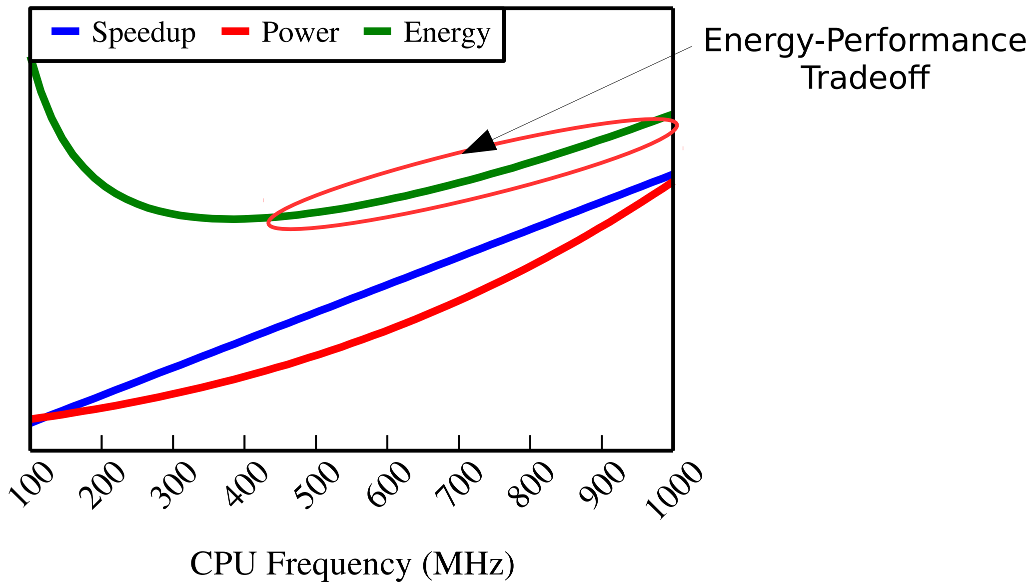
Dynamic Voltage and Frequency Scaling



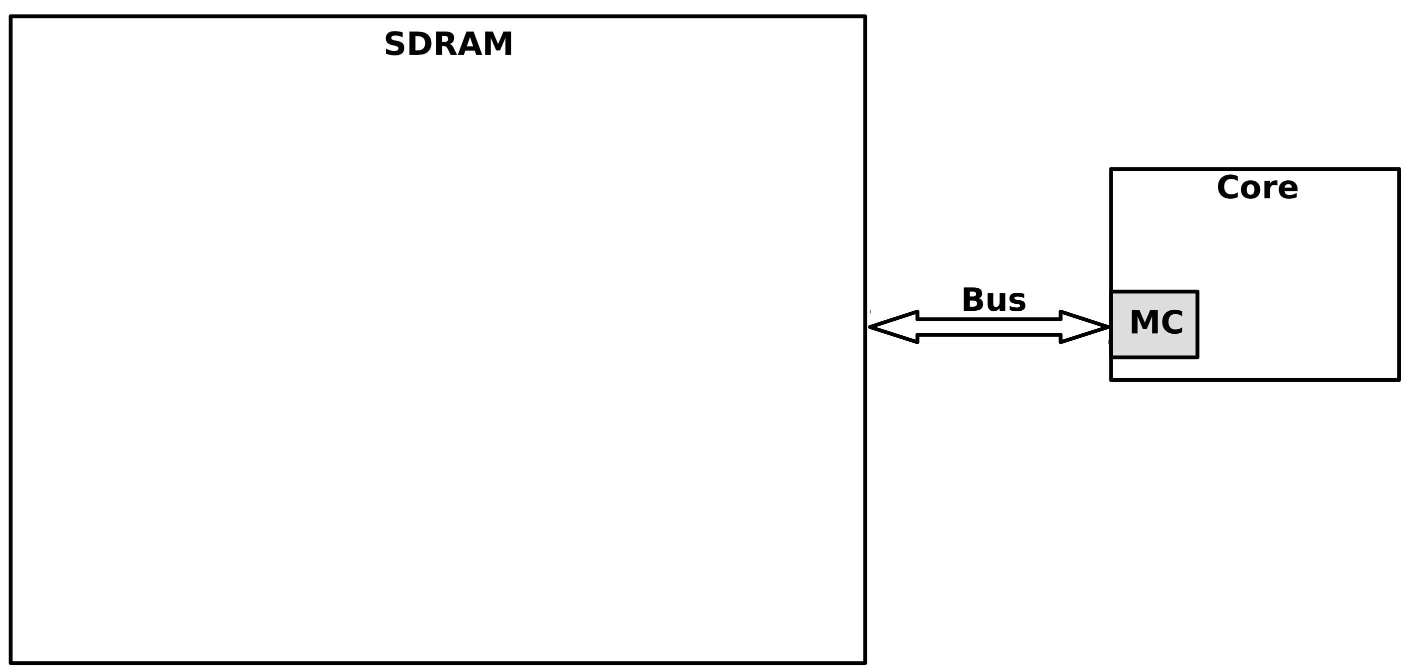
Dynamic Voltage and Frequency Scaling



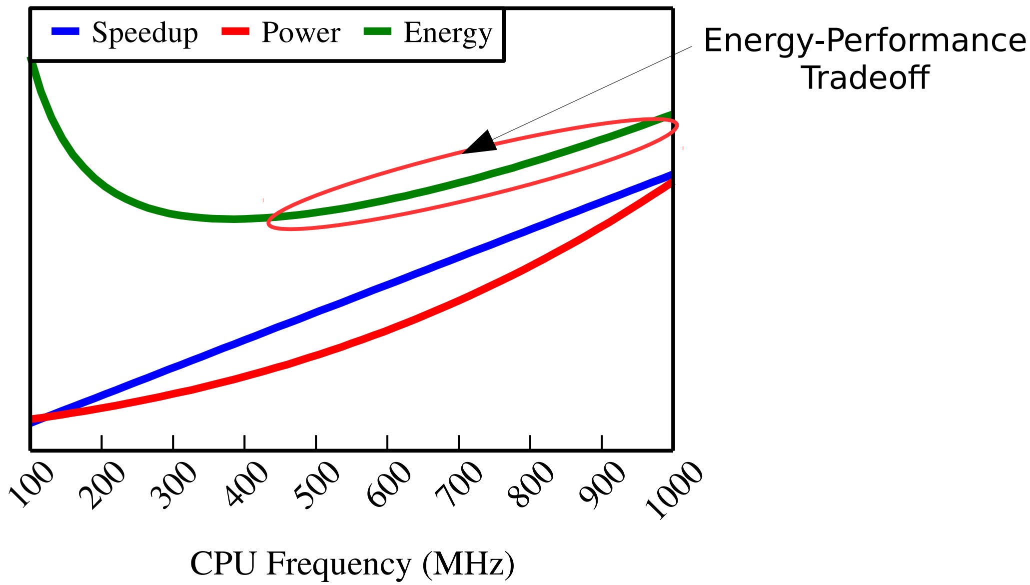
Dynamic Voltage and Frequency Scaling



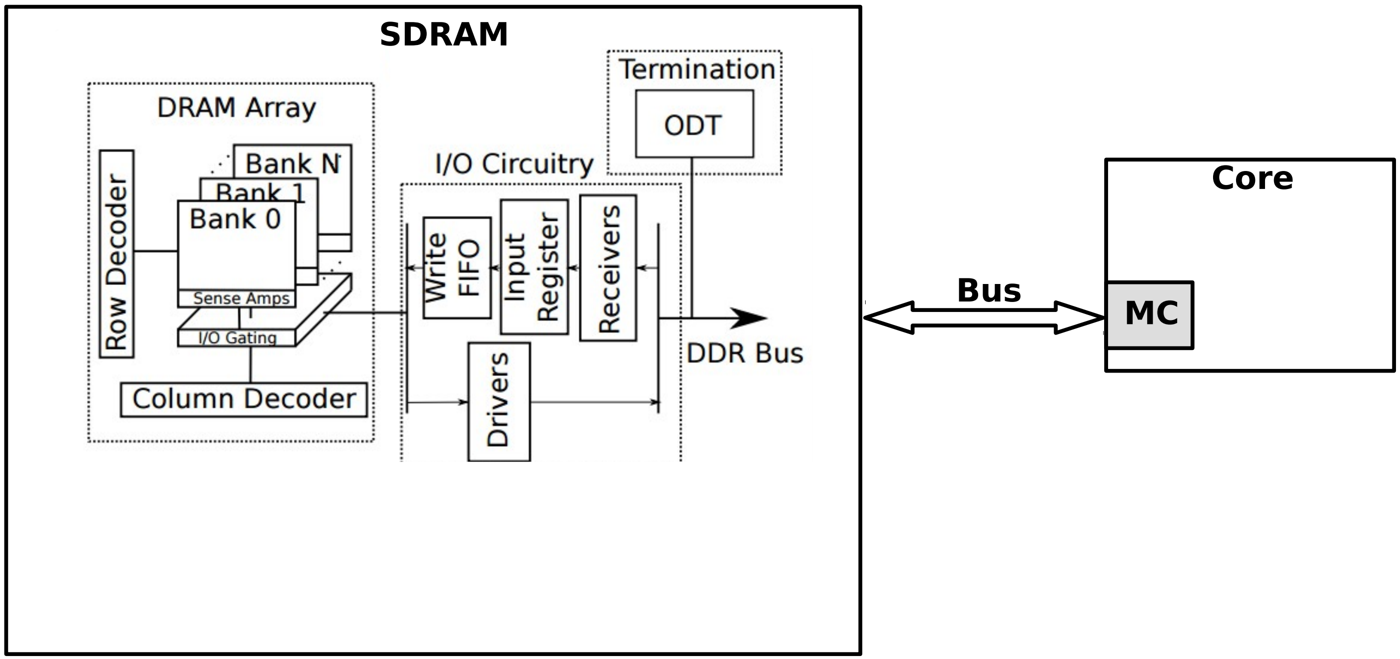
SDRAM Frequency Scaling



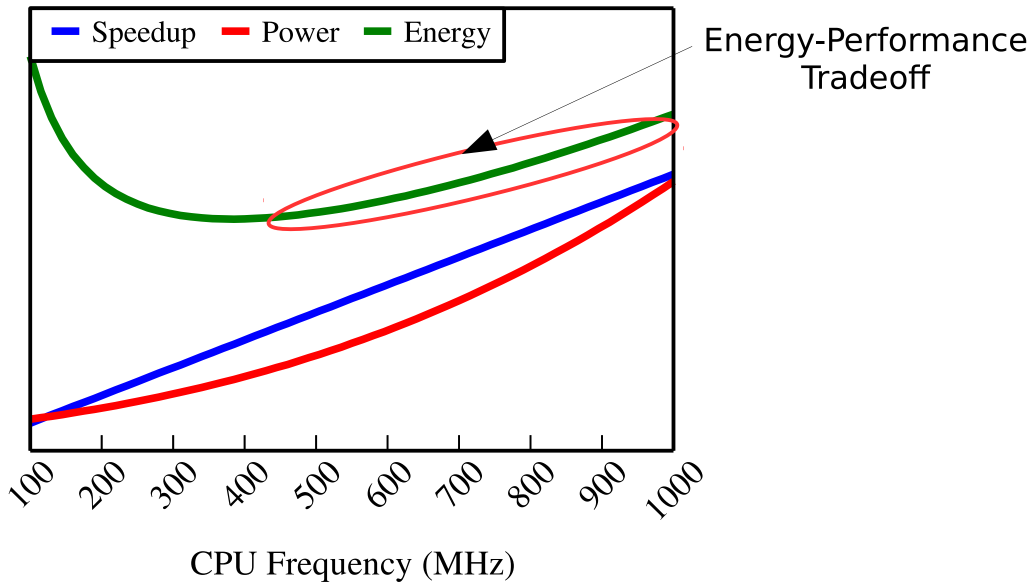
Dynamic Voltage and Frequency Scaling



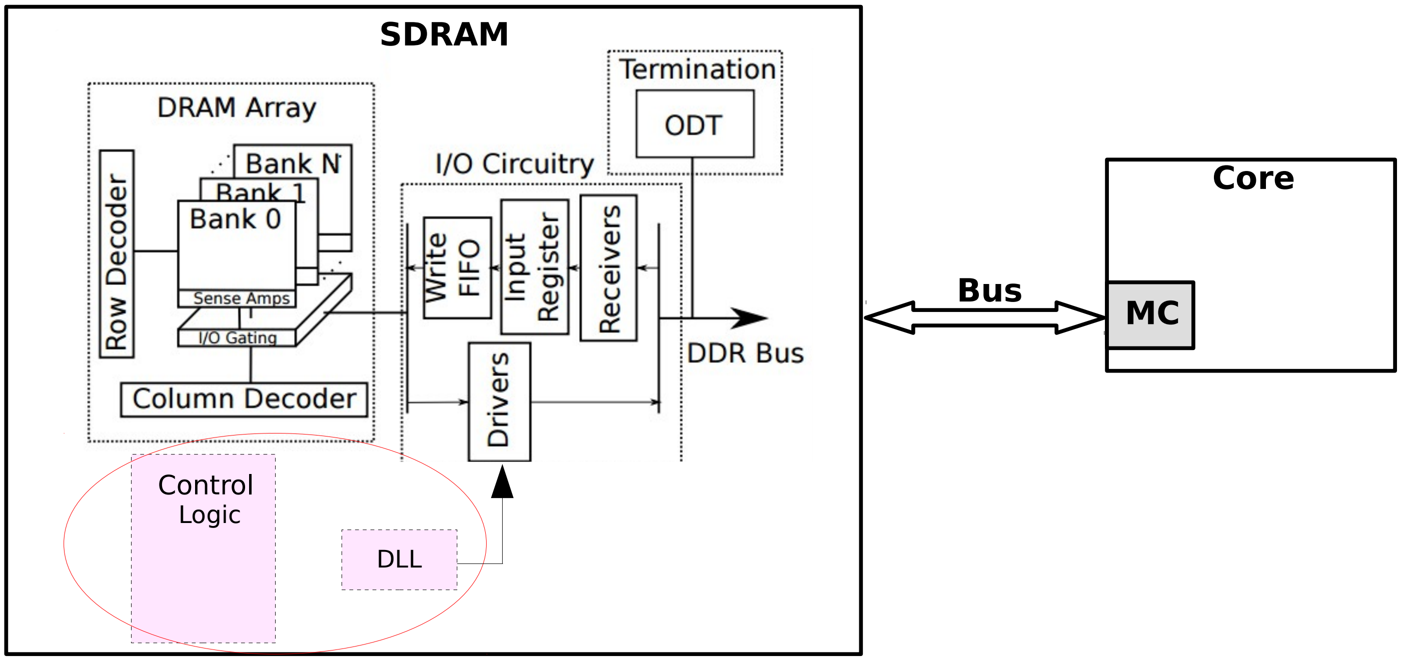
SDRAM Frequency Scaling



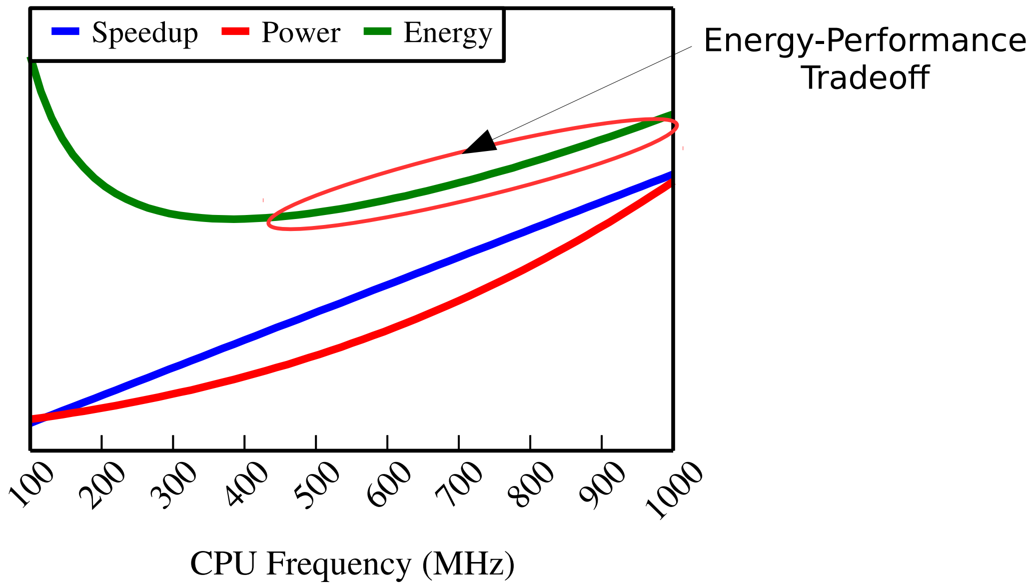
Dynamic Voltage and Frequency Scaling



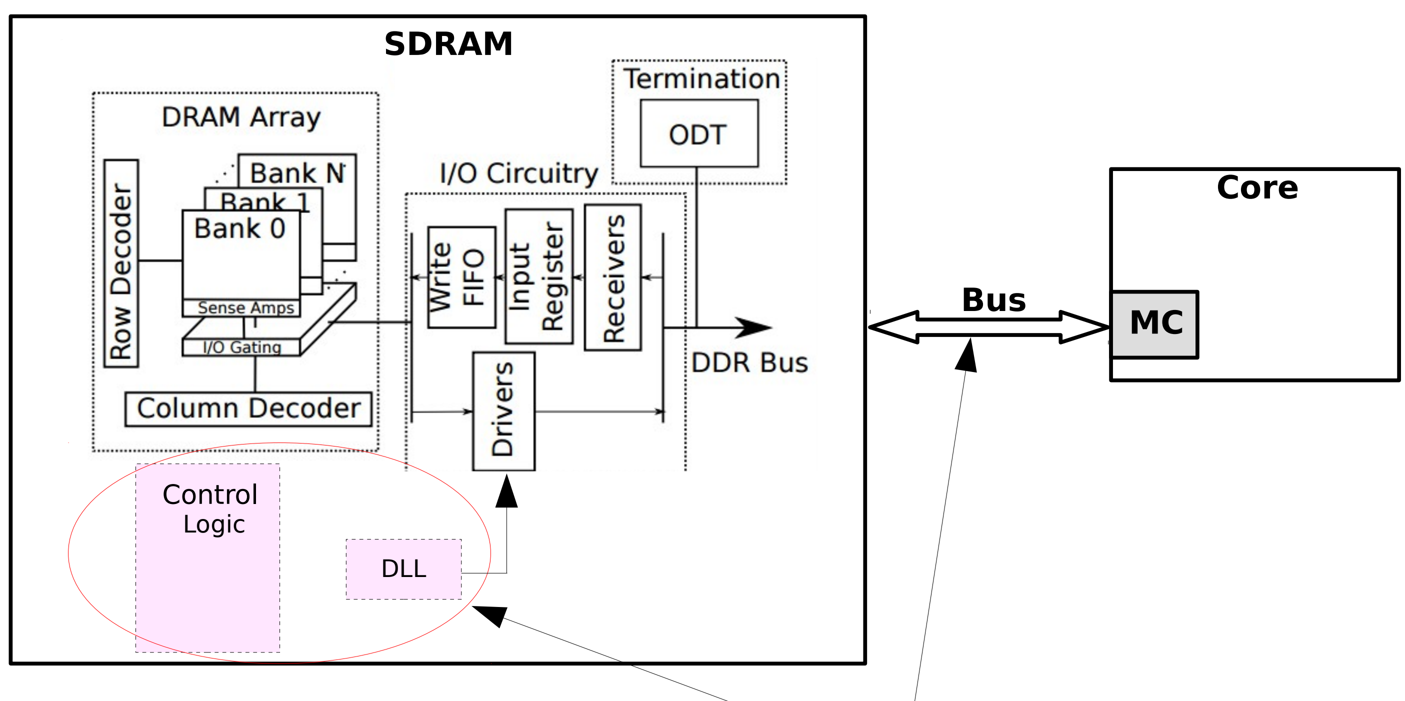
SDRAM Frequency Scaling



Dynamic Voltage and Frequency Scaling



SDRAM Frequency Scaling



DRAM DFS

- Only Frequency Scaling
- Performance and power are proportional to DRAM frequency
- Increase in energy with DRAM frequency is due to scalable and non-scalable parts of DRAM

David, Howard, et al. international conference on Autonomic computing. ACM, 2011.

CPU DVFS and Memory DFS

- Managing Systems - a challenging task

CPU DVFS and Memory DFS

- Managing Systems - a challenging task

CPU DVFS and Memory DFS

- Managing Systems - a challenging task

- CPU intensive applications – higher CPU frequency
- Interplay of performance and energy of CPU and memory frequency scaling is complex

Performance vs. Energy Constraints

- Previous efforts explored DVFS under performance constraints

Performance vs. Energy Constraints

- Previous efforts explored DVFS under performance constraints
- Servers --- working under performance constraints is imperative

Performance vs. Energy Constraints

- Previous efforts explored DVFS under performance constraints
- Servers --- working under performance constraints is imperative
- Mobile systems –-- operating under energy constraints is fitting

Performance vs. Energy Constraints

- Previous efforts explored DVFS under performance constraints
- Servers --- working under performance constraints is imperative
- Mobile systems –-- operating under energy constraints is fitting
- Absolute energy or rate of energy consumption as energy constraints --- application and device dependent

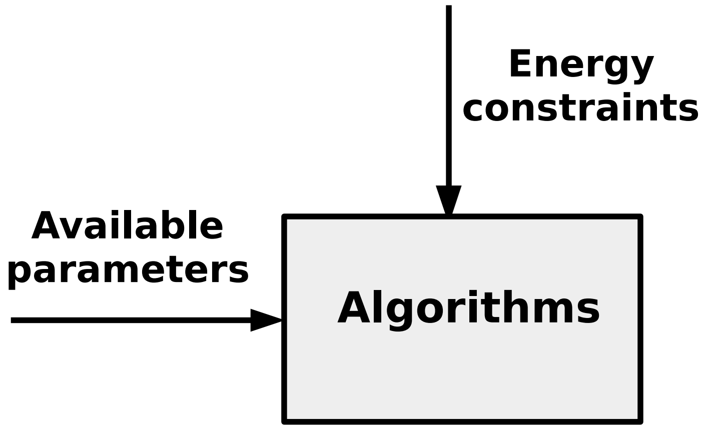
Performance vs. Energy Constraints

- Previous efforts explored DVFS under performance constraints
- Servers --- working under performance constraints is imperative
- Mobile systems –-- operating under energy constraints is fitting
- Absolute energy or rate of energy consumption as energy constraints --- application and device dependent
- Need for a new metric –-- ***Inefficiency***

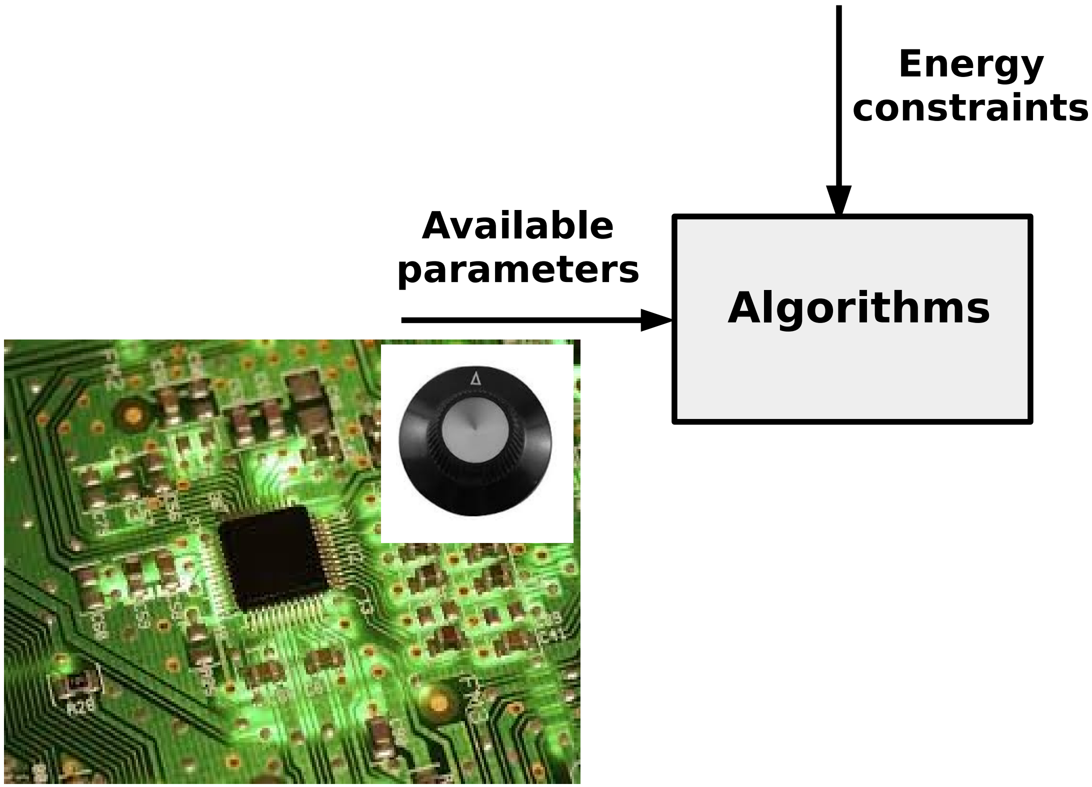
Outline

- Inefficiency
- Inefficiency vs. Speedup
- Characteristics of Optimal Frequency Settings
- Performance Clusters and Stable Regions
- Conclusions and Future Work

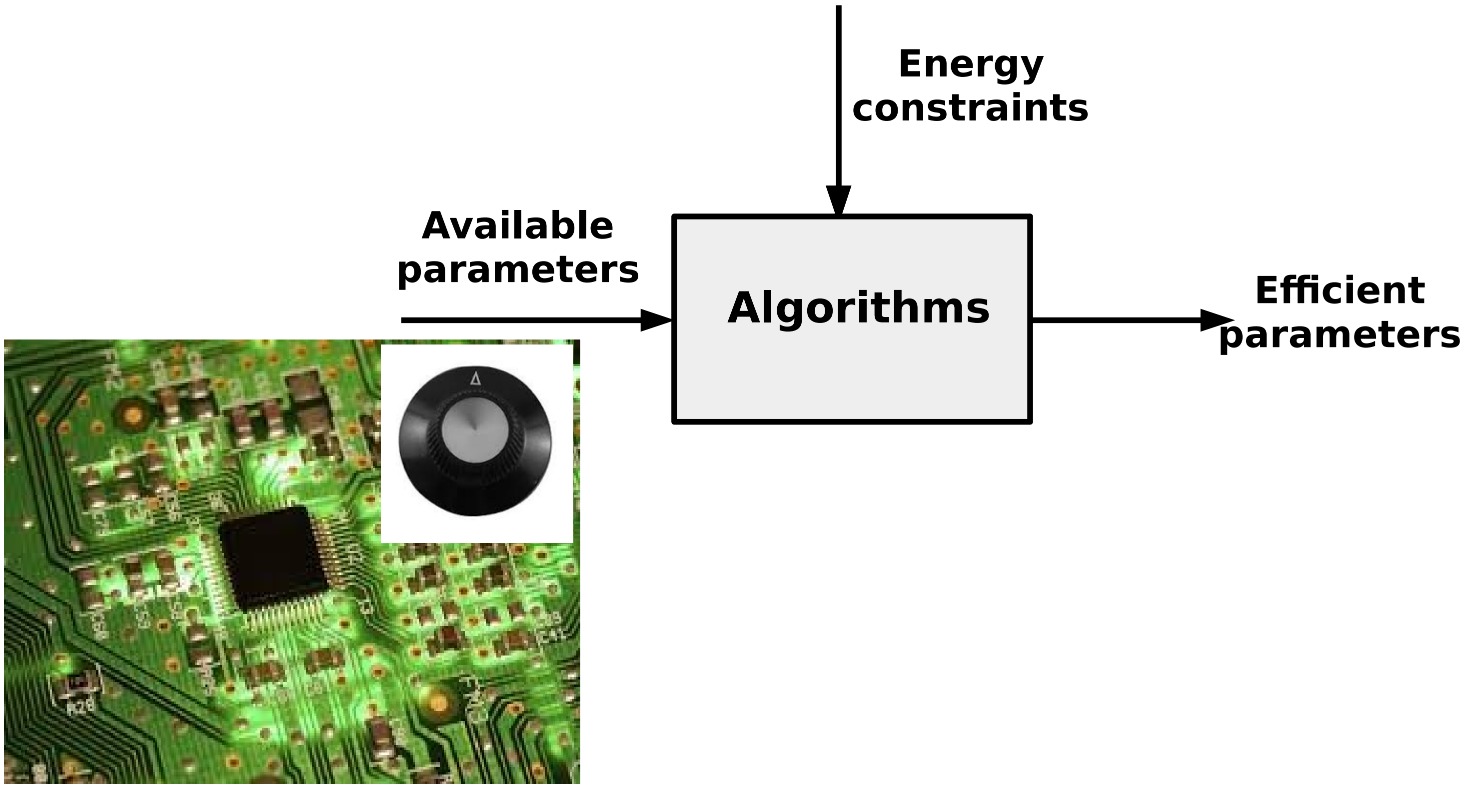
Energy Management Algorithms



Energy Management Algorithms



Energy Management Algorithms



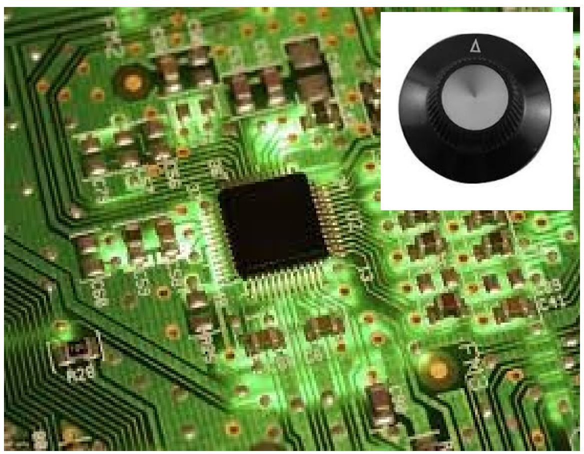
Energy Management Algorithms

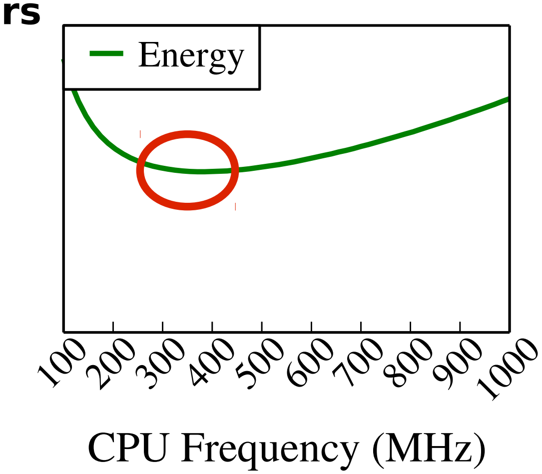
Energy constraints

Available parameters

Algorithms

Efficient parameters



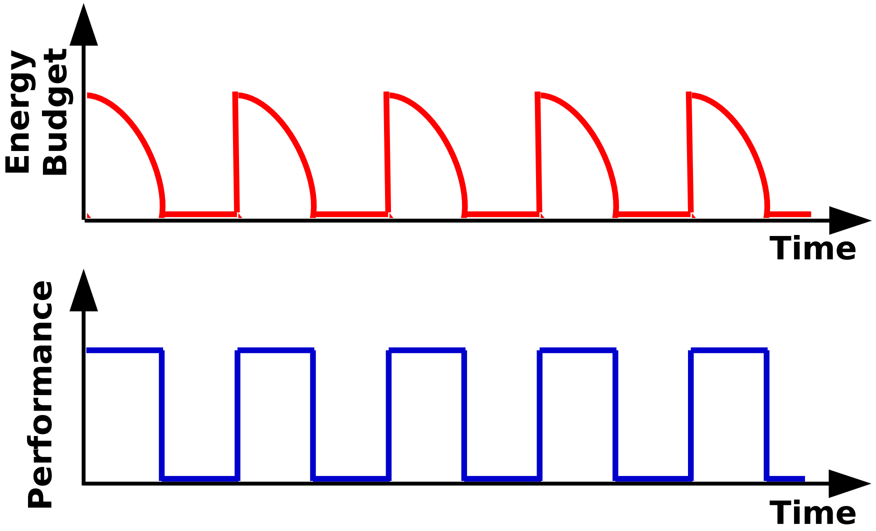


Energy Management Algorithms

- Rate limiting

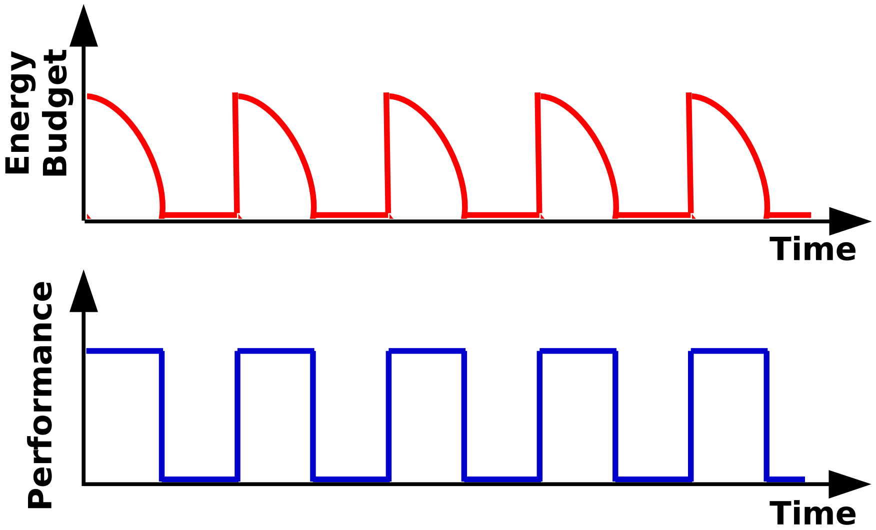
Energy Efficient Algorithms

- Rate limiting
  - Impacts both performance and energy



Energy Efficient Algorithms

- Rate limiting
  - Impacts both performance and energy



- Energy–Delay products – EDP, ED^{2}P etc.

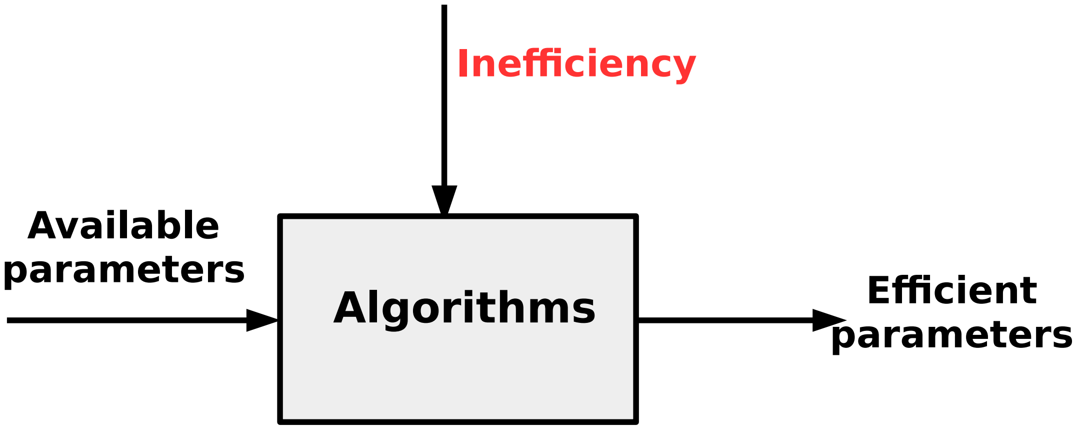
Inefficiency

Inefficiency

- Inefficiency: Additional energy that can be used by the *application* to improve performance

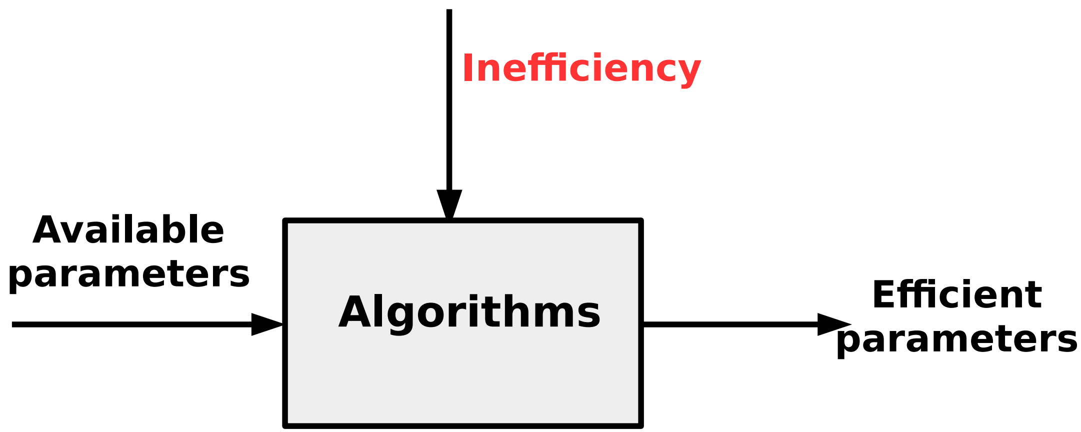
Inefficiency

- Inefficiency: Additional energy that can be used by the *application* to improve performance



Inefficiency

- Inefficiency: Additional energy that can be used by the *application* to improve performance

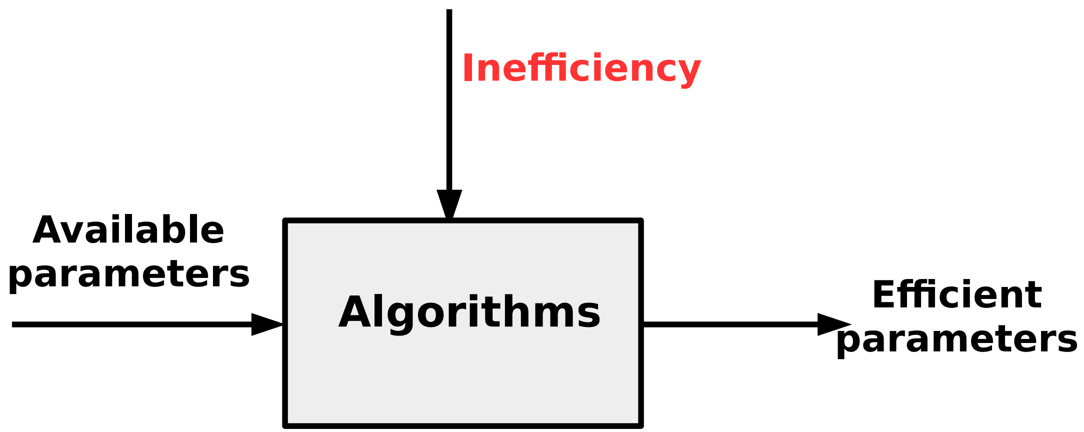


- Inefficiency:

$$\textit{Inefficiency} = \frac{E_{total}}{E_{min}}$$

Inefficiency

- Inefficiency: Additional energy that can be used by the *application* to improve performance



- Inefficiency:

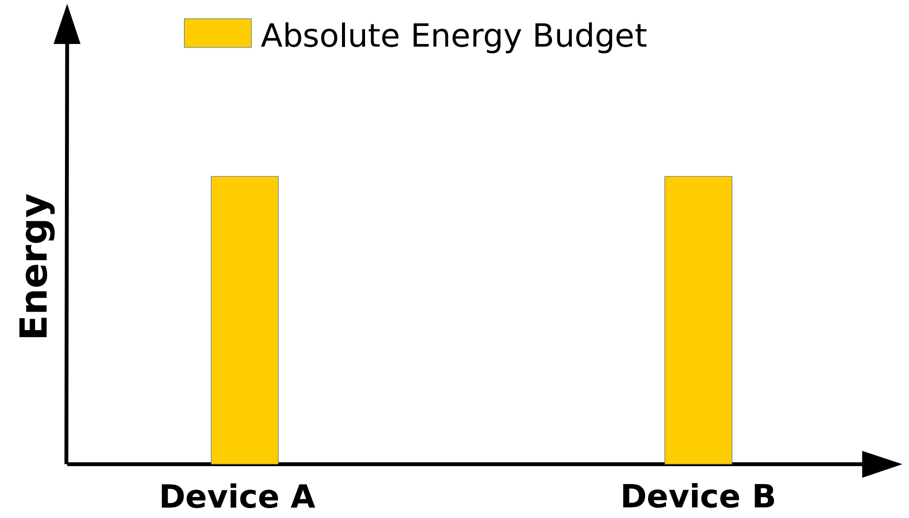
$$\textit{Inefficiency} = \frac{E_{total}}{E_{min}}$$

E_{min} – Minimum energy application could have consumed on the same device

E_{total} – Additional energy application can use to improve performance

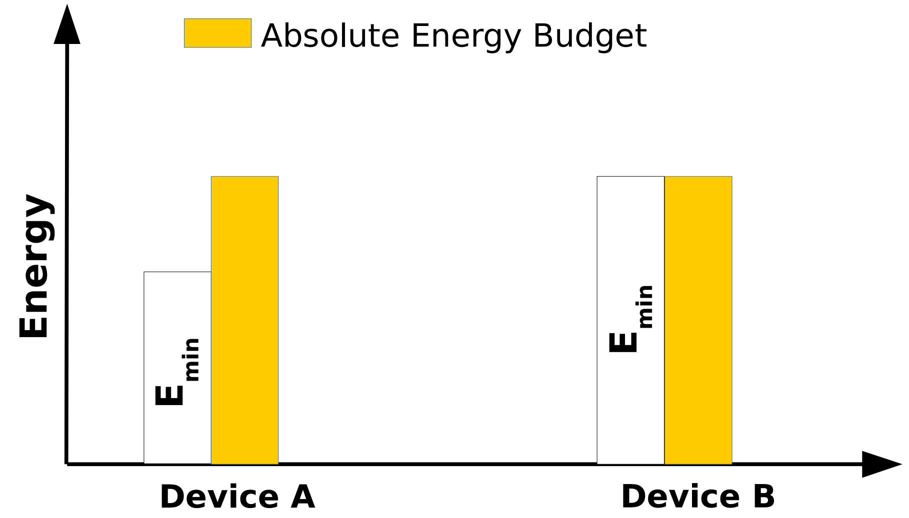
Inefficiency as a System Resource

- Agnostic to Devices



Inefficiency as a System Resource

- Agnostic to Devices



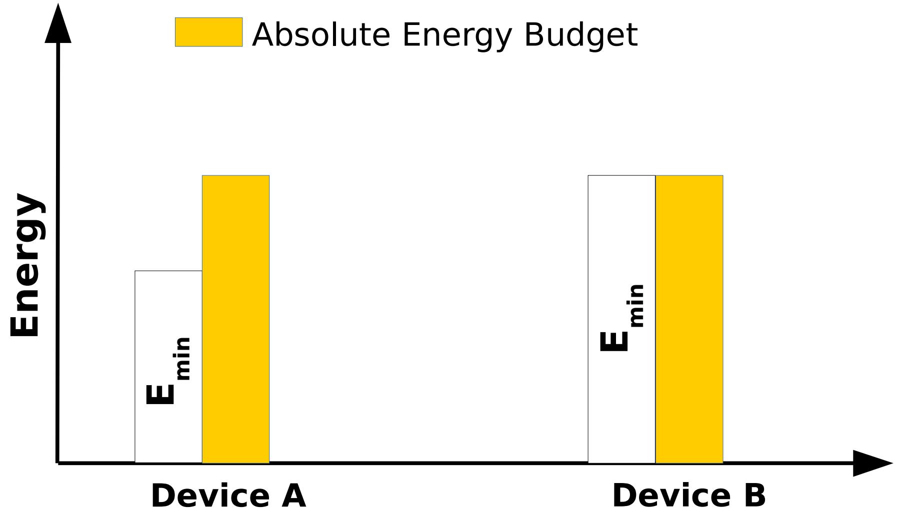
Inefficiency as a System Resource

- Agnostic to Devices

$$Inefficiency = \frac{E_{total}}{E_{min}}$$

$$\Downarrow$$

$$E_{total} = Inefficiency \times E_{min}$$

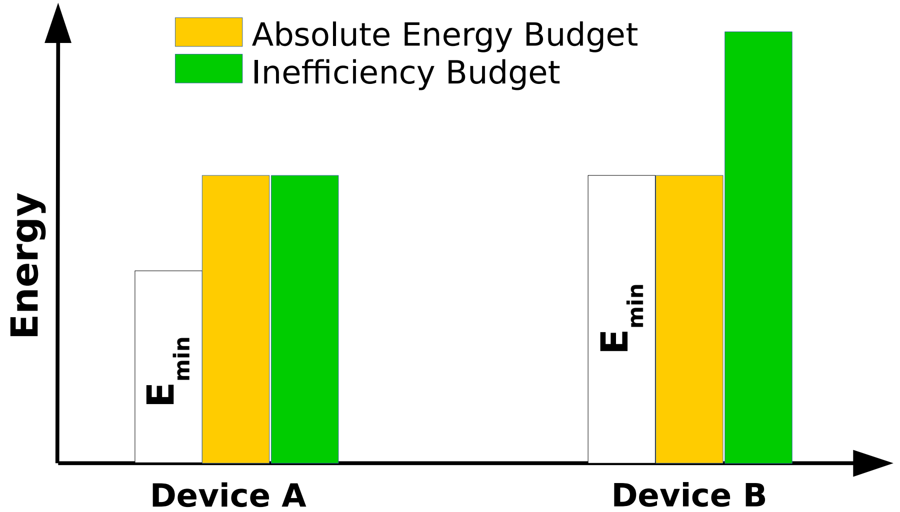


Inefficiency as a System Resource

- Agnostic to Devices

$$\textit{Inefficiency} = \frac{E_{total}}{E_{min}}$$

$$E_{total} = \textit{Inefficiency} \times E_{min}$$

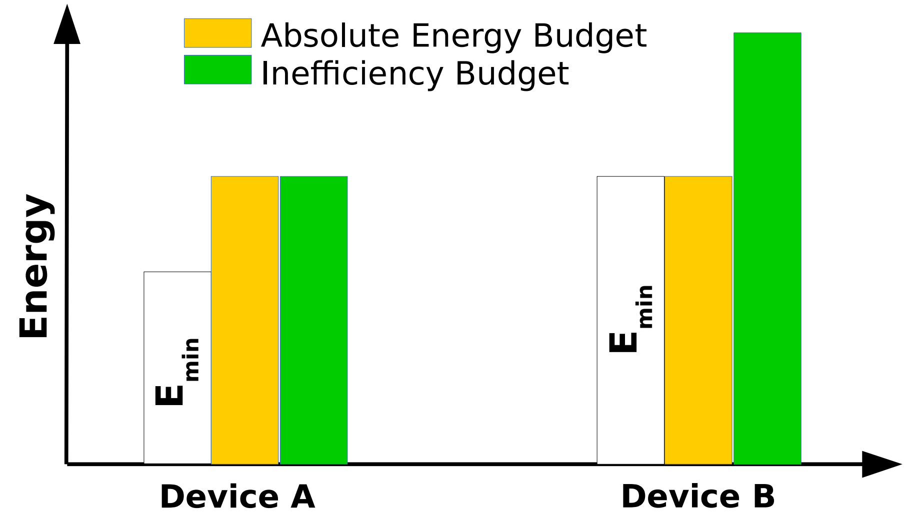


Inefficiency as a System Resource

- Agnostic to Devices

$$\textit{Inefficiency} = \frac{E_{total}}{E_{min}}$$

$$E_{total} = \textit{Inefficiency} \times E_{min}$$



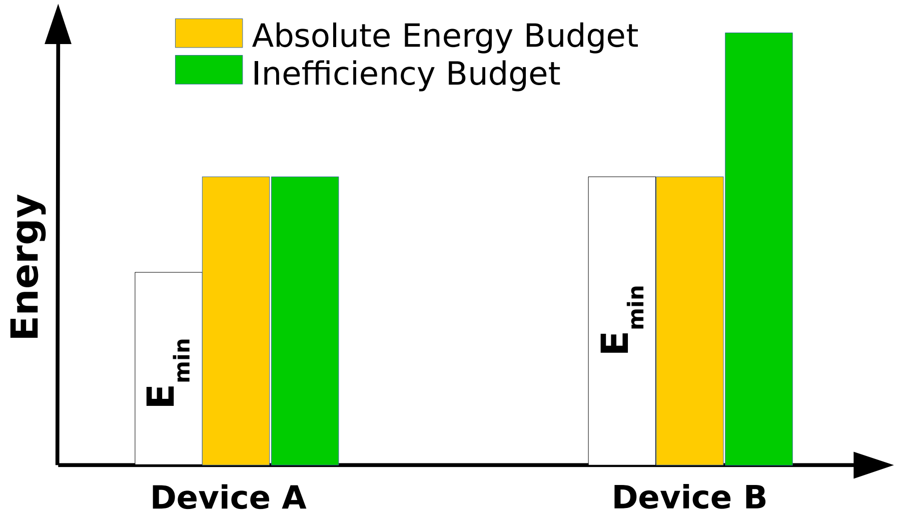
- Relative to inherent energy needs of the application
  - Inefficiency tied to priority of the applications

Inefficiency as a System Resource

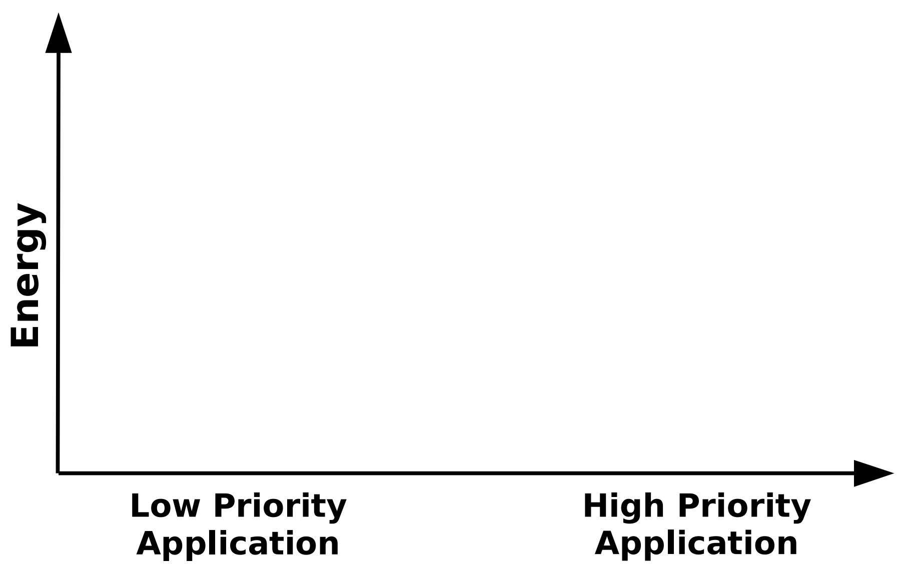
- Agnostic to Devices

$$Inefficiency = \frac{E_{total}}{E_{min}}$$

$$E_{total} = Inefficiency \; X \; E_{min}$$



- Relative to inherent energy needs of the application
  - Inefficiency tied to priority of the applications

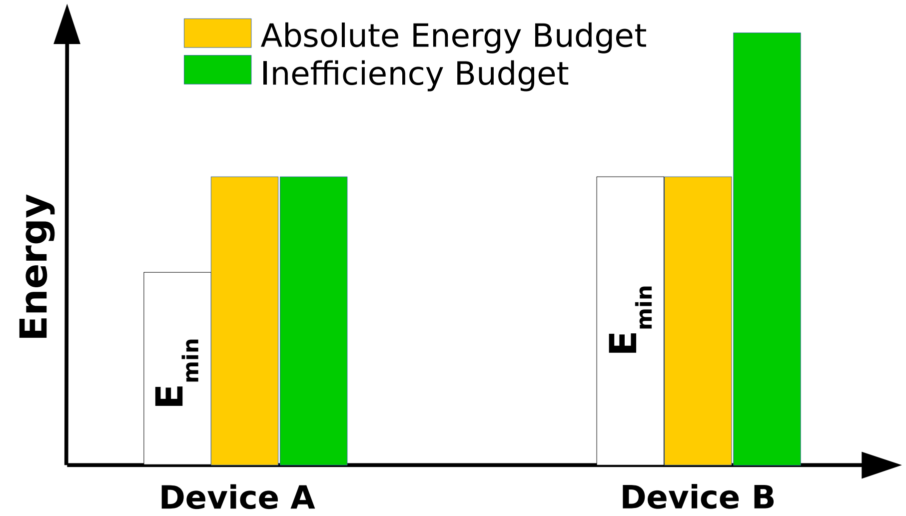


Inefficiency as a System Resource

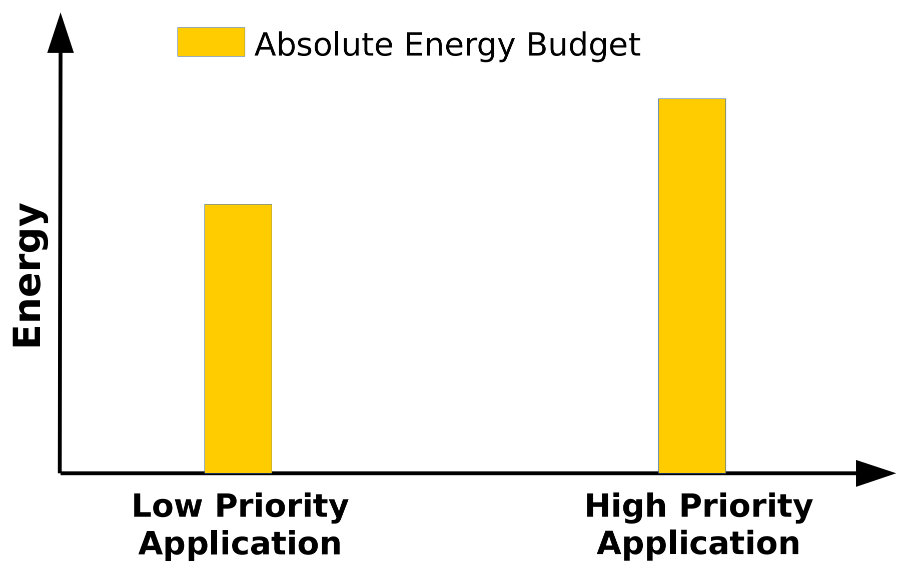
- Agnostic to Devices

$$Inefficiency = \frac{E_{total}}{E_{min}}$$

$$E_{total} = Inefficiency \times E_{min}$$



- Relative to inherent energy needs of the application
  - Inefficiency tied to priority of the applications



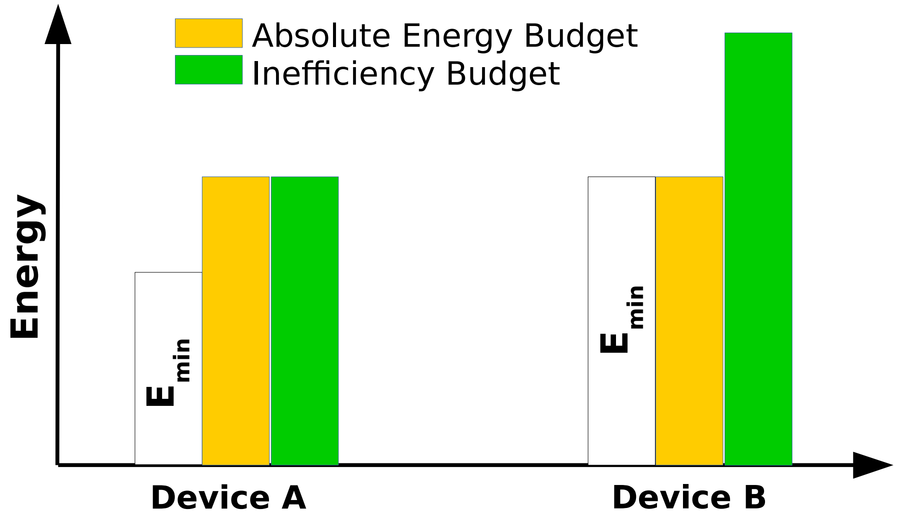
Inefficiency as a System Resource

- Agnostic to Devices

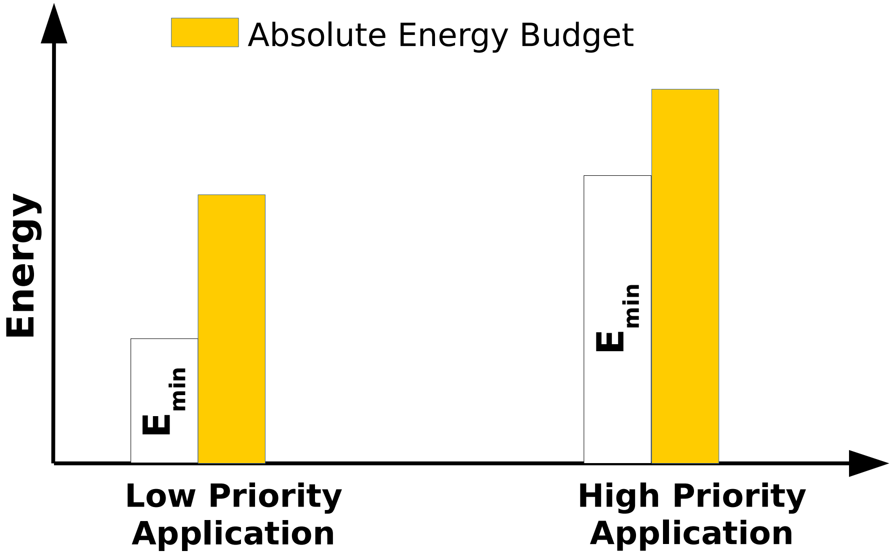
$$Inefficiency = \frac{E_{total}}{E_{min}}$$

$$\Downarrow$$

$$E_{total} = Inefficiency \times E_{min}$$



- Relative to inherent energy needs of the application
  - Inefficiency tied to priority of the applications

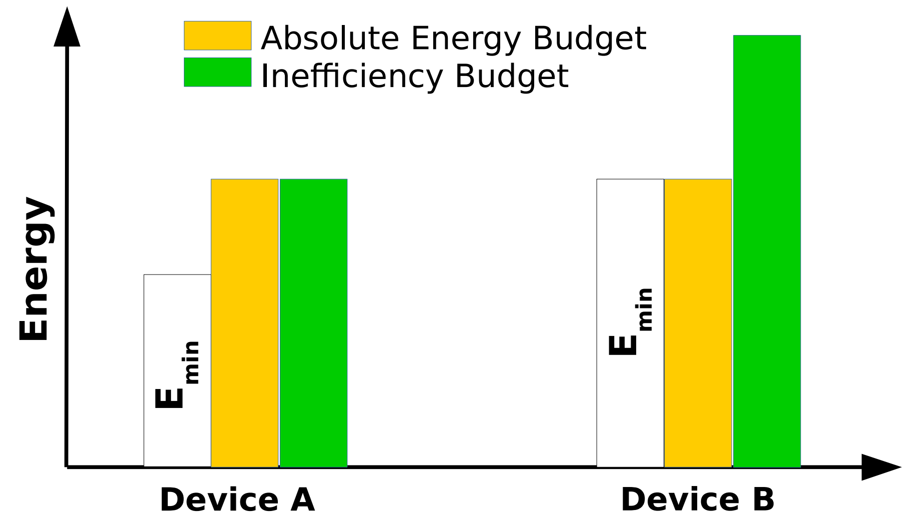


Inefficiency as a System Resource

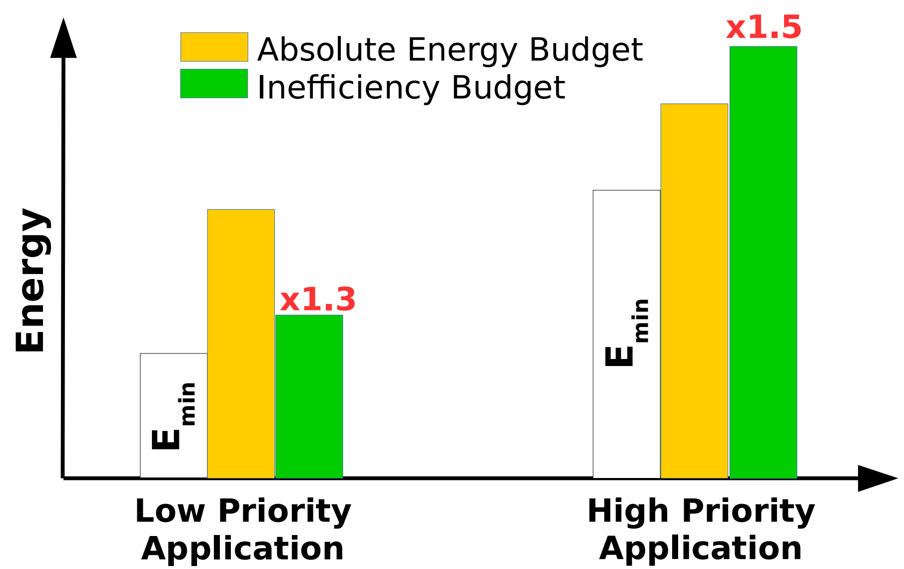
- Agnostic to Devices

$$\textit{Inefficiency} = \frac{E_{total}}{E_{min}}$$

$$E_{total} = \textit{Inefficiency} \times E_{min}$$



- Relative to inherent energy needs of the application
  - Inefficiency tied to priority of the applications



Inefficiency

- What are bounds of inefficiency?

Inefficiency

- What are bounds of inefficiency?
  - 1 and I_{max} --- I_{max} is device dependent

Inefficiency

- What are bounds of inefficiency?
  - 1 and I_{max} --- I_{max} is device dependent and is irrelevant

- How is inefficiency estimated?

Inefficiency

- What are bounds of inefficiency?
  - 1 and I_{max} --- I_{max} is device dependent and is irrelevant

- How is inefficiency estimated?
  - E_{total} computation is straightforward --- energy counters and research tools

Inefficiency

- What are bounds of inefficiency?
  - 1 and I_{max} --- I_{max} is device dependent and is irrelevant
- How is inefficiency estimated?
  - E_{total} computation is straightforward --- energy counters and research tools
  - E_{min} computation is challenging --- developed cross-component energy models

Inefficiency

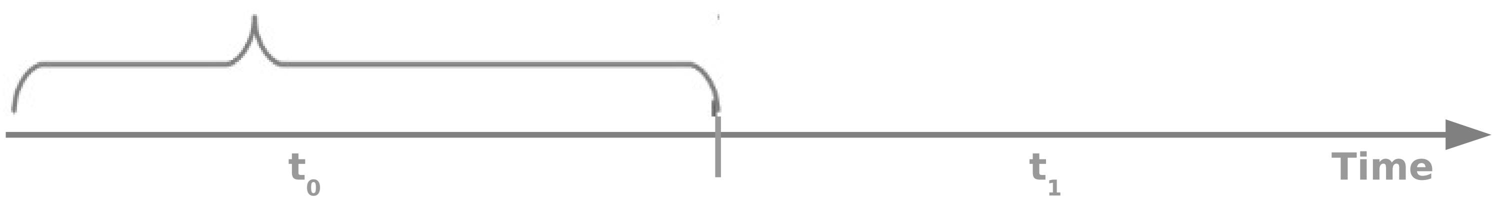
- What are bounds of inefficiency?
  - 1 and I_{max} --- I_{max} is device dependent and is irrelevant

- How is inefficiency estimated?
  - E_{total} computation is straightforward --- energy counters and research tools
  - E_{min} computation is challenging --- developed cross-component energy model

- How should systems stay within the inefficiency budget

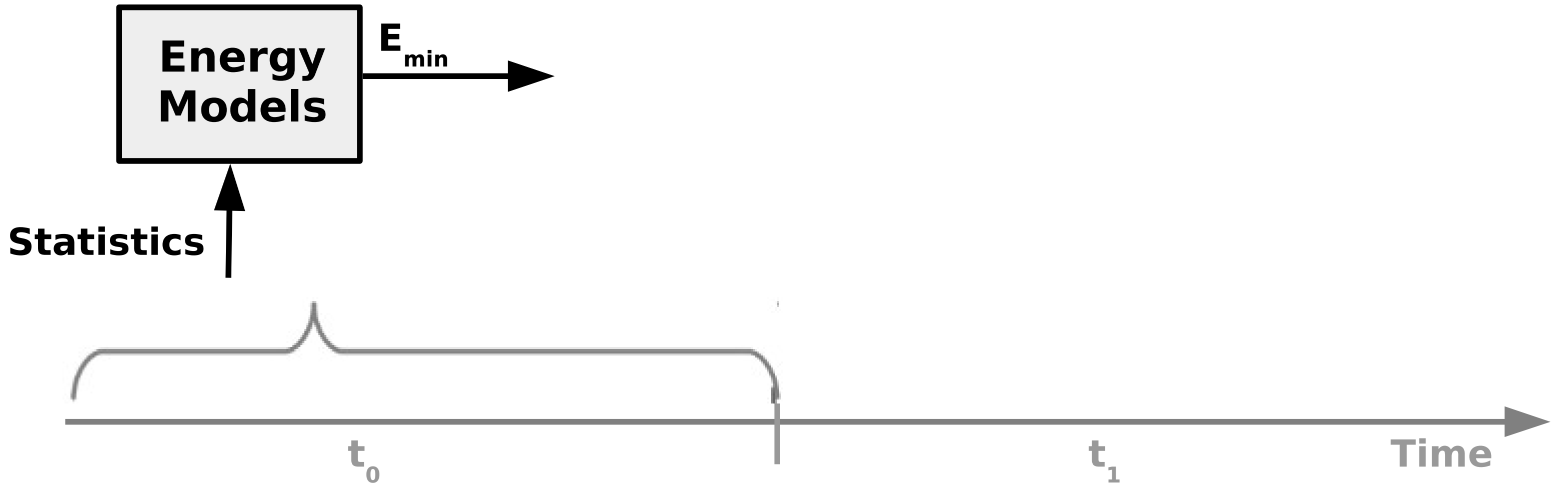
Inefficiency

- How should systems stay within the inefficiency budget



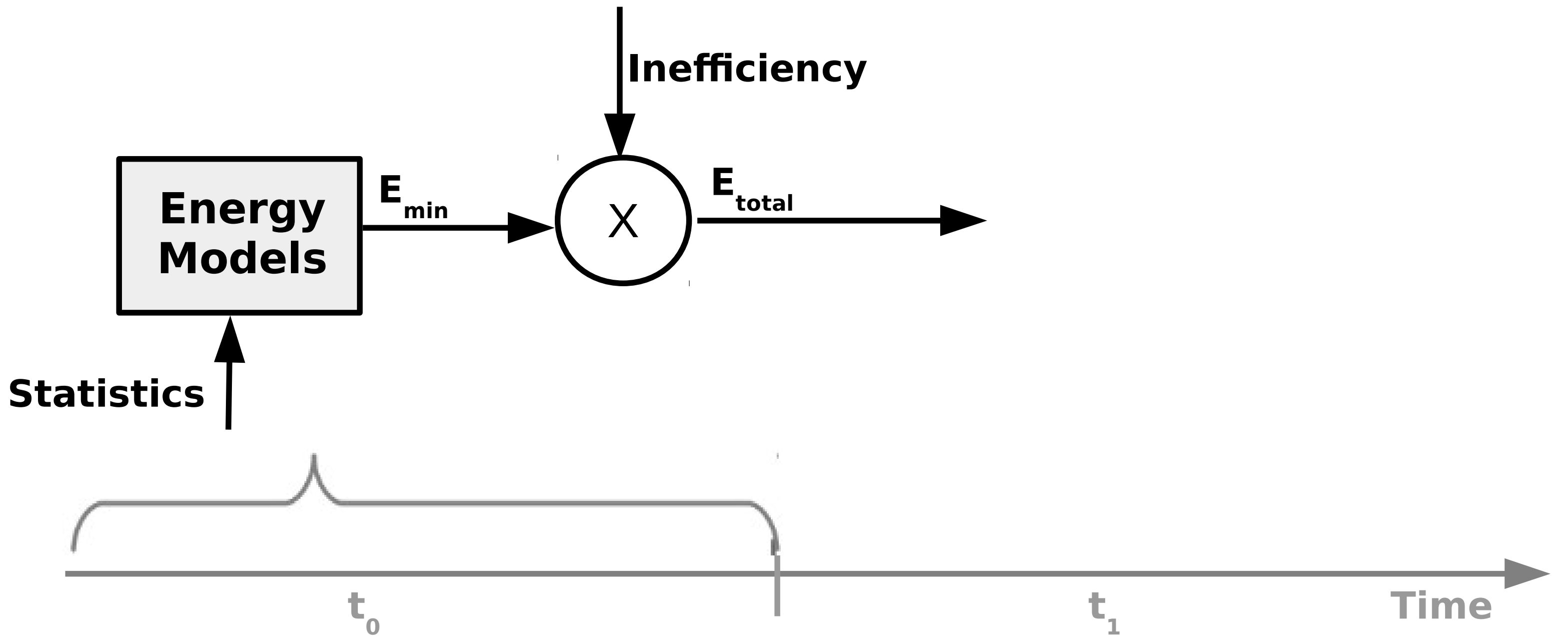
Inefficiency

➢ How should systems stay within the inefficiency budget



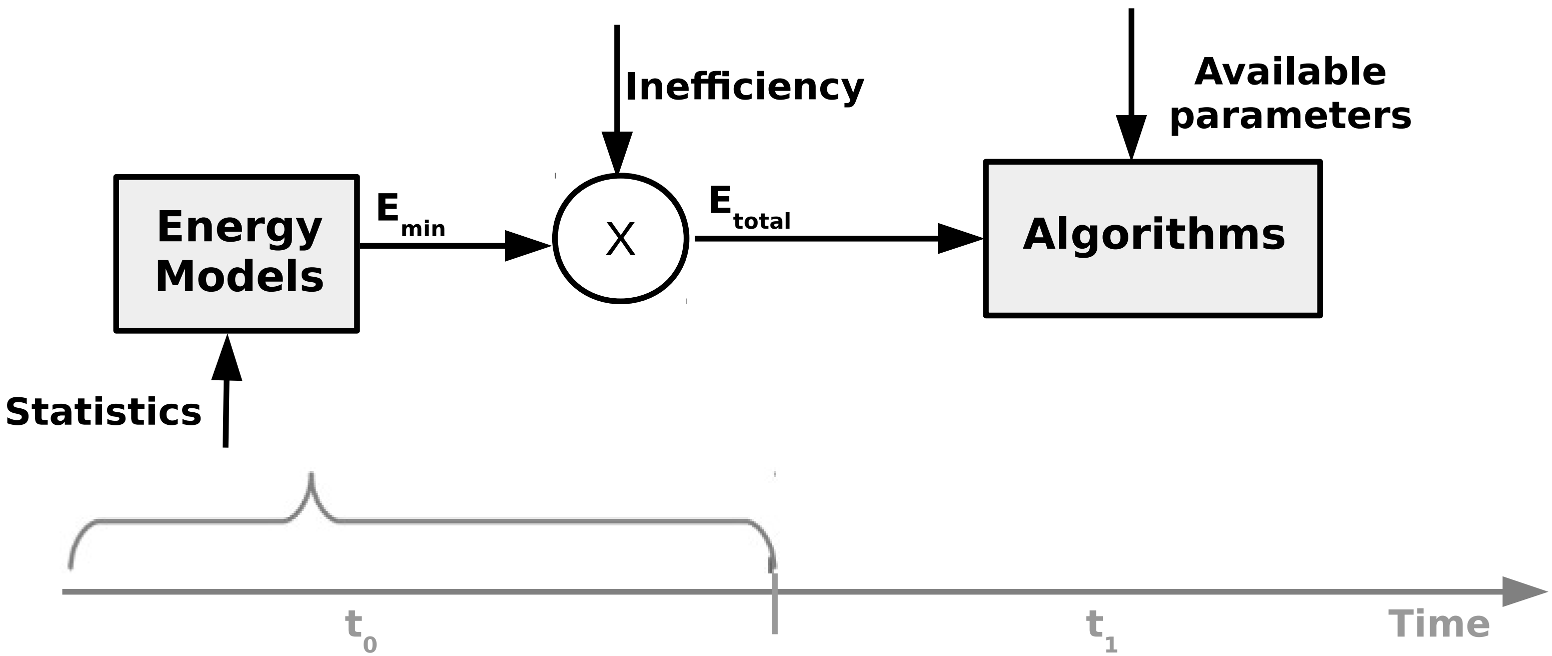
Inefficiency

- How should systems stay within the inefficiency budget



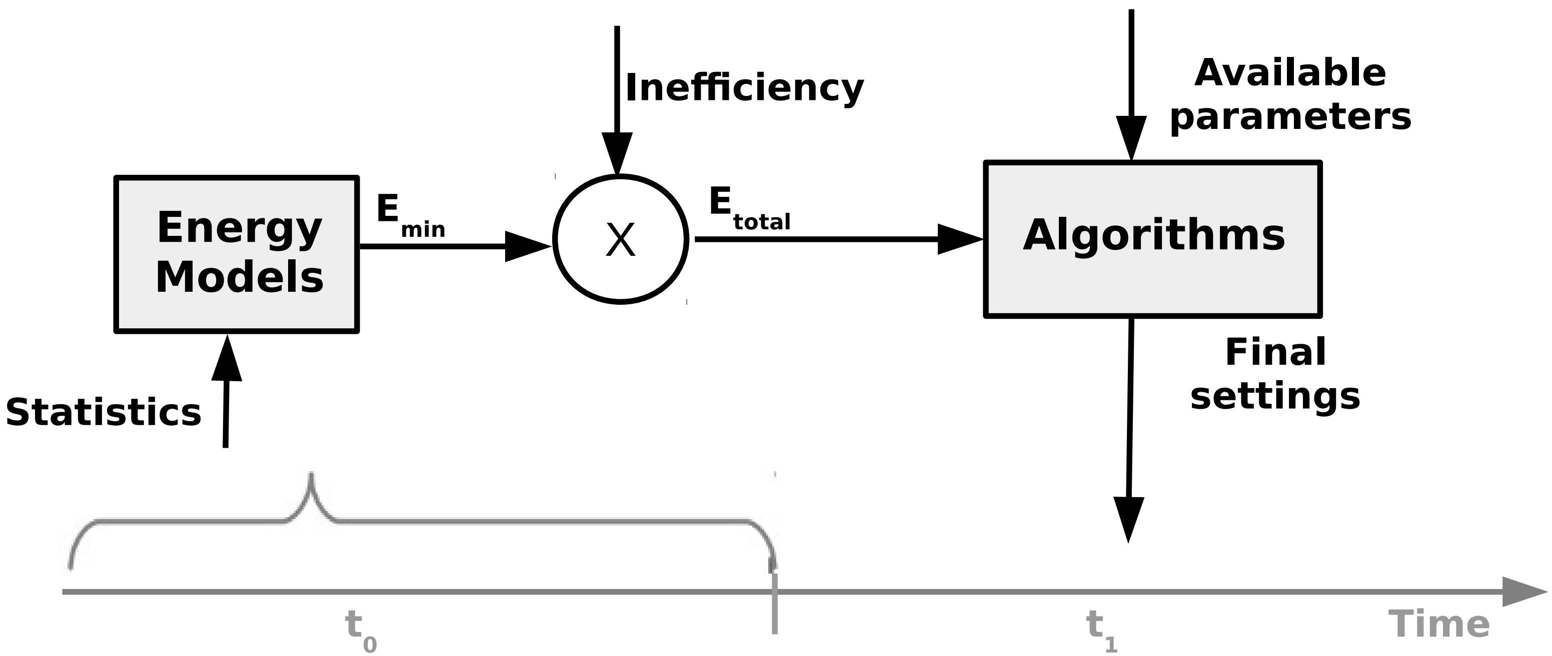
Inefficiency

- How should systems stay within the inefficiency budget



Inefficiency

- How should systems stay within the inefficiency budget



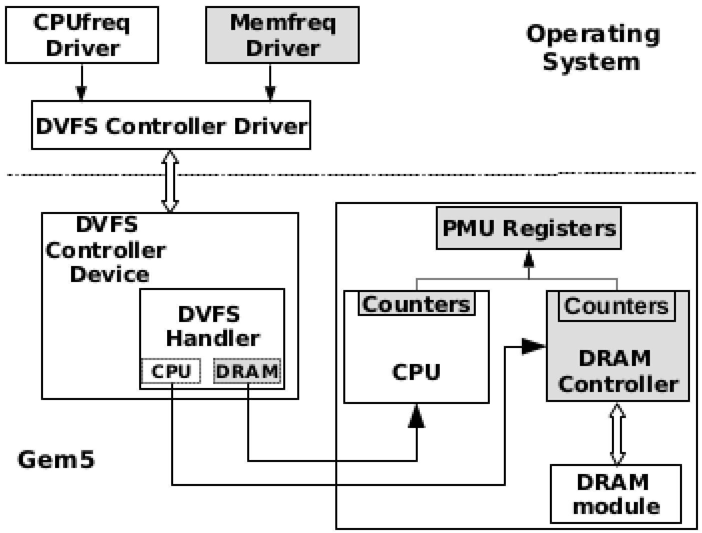
Outline

- Inefficiency
- **Inefficiency vs. Speedup**
- Characteristics of Optimal Frequency Settings
- Performance Clusters and Stable Regions
- Conclusions and Future Work

Inefficiency vs. Speedup

Inefficiency vs. Speedup

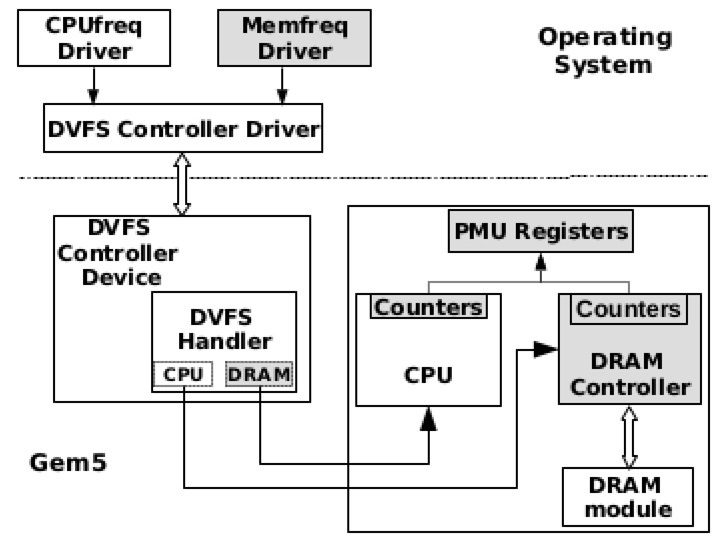
- Methodology
  - Gem5
  - DVFS controller driver



Inefficiency vs. Speedup

- Methodology
  - Gem5
  - DVFS controller driver
  - Android 4.1.1 Jelly Bean
  - CPU : 100 – 1000 MHz,

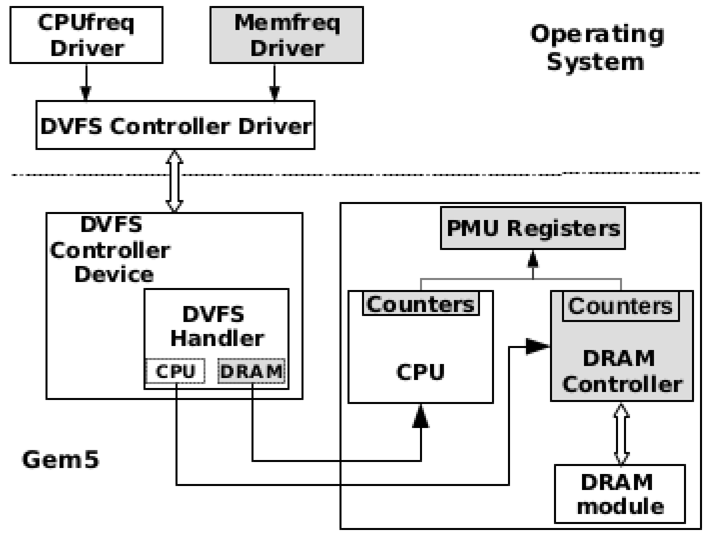
    0.65 – 1.25 V
  - DRAM : 200 – 800 MHz, 1.2V



Inefficiency vs. Speedup

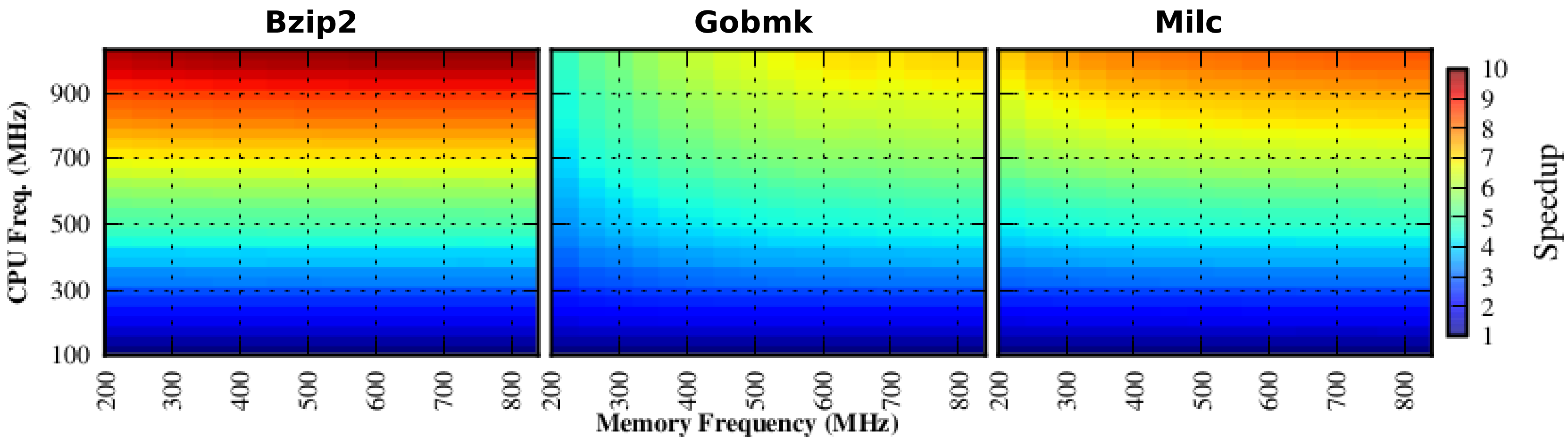
- Methodology
  - Gem5
  - DVFS controller driver
  - Android 4.1.1 Jelly Bean
  - CPU : 100 – 1000 MHz,

    0.65 – 1.25 V
  - DRAM : 200 – 800 MHz, 1.2V

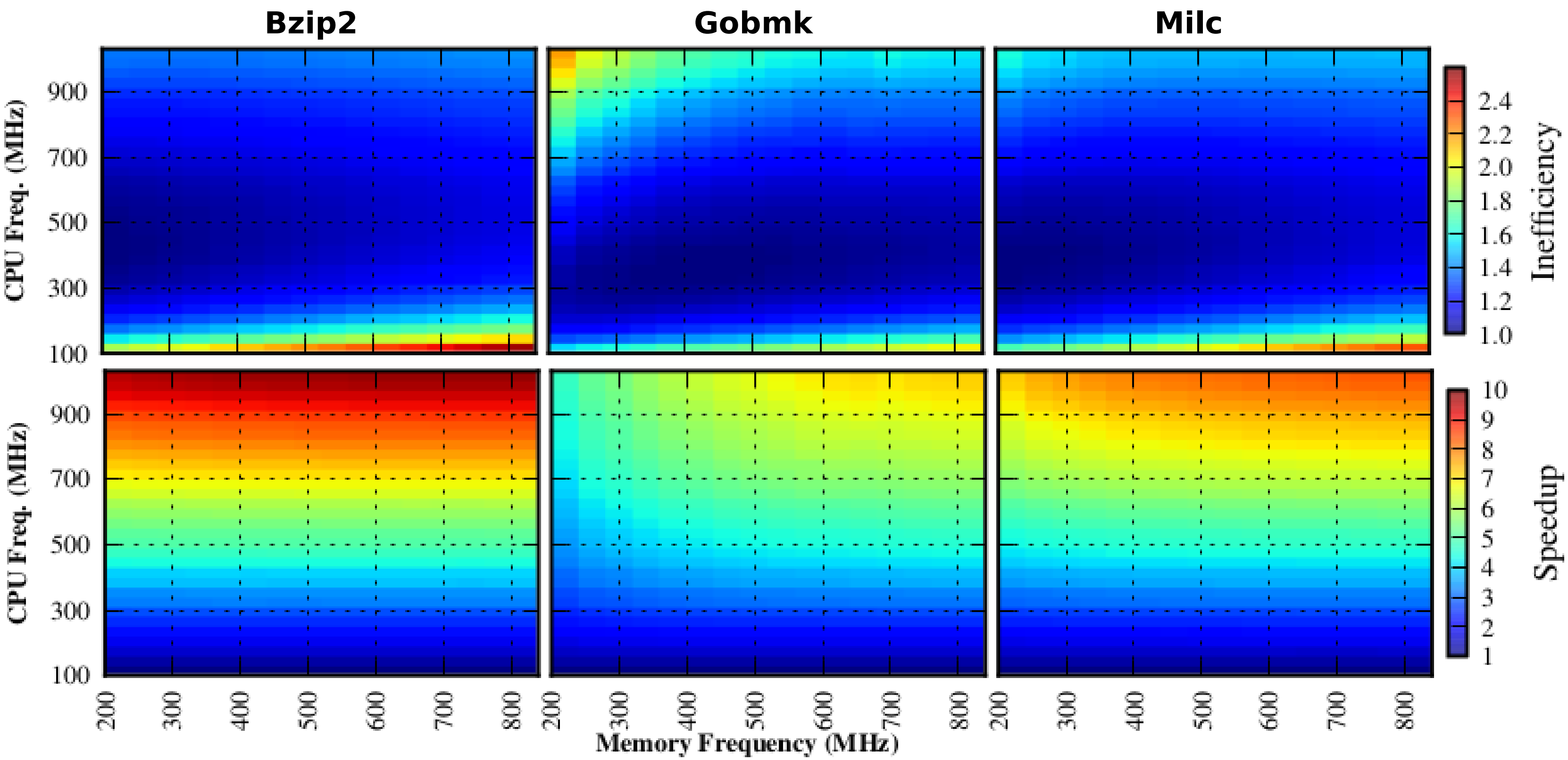


- Energy Models
  - Cortex – A9, Pandaboard
  - Micron power model --- extended to incorporate frequency scaling

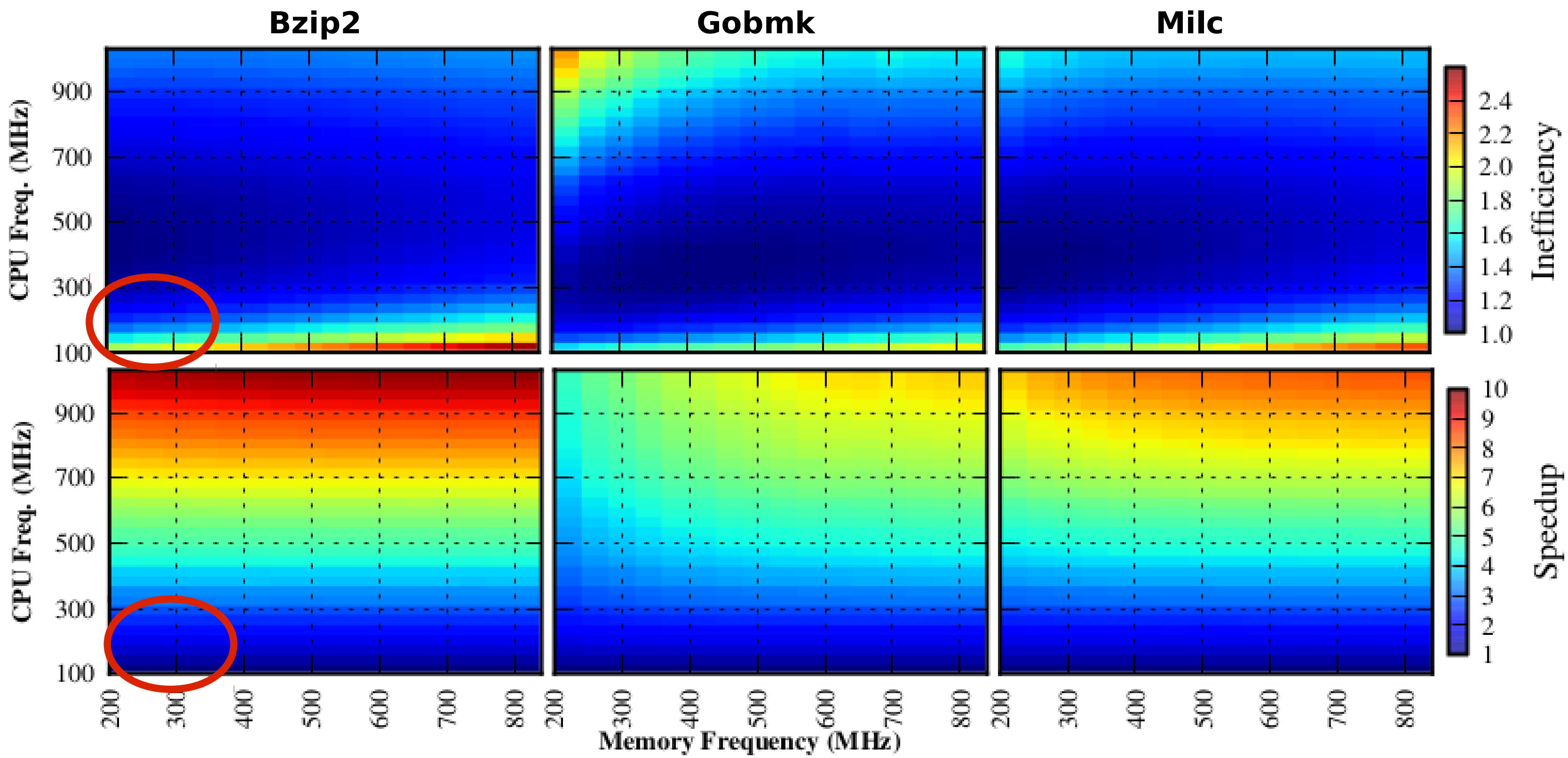
Inefficiency vs. Speedup



Inefficiency vs. Speedup

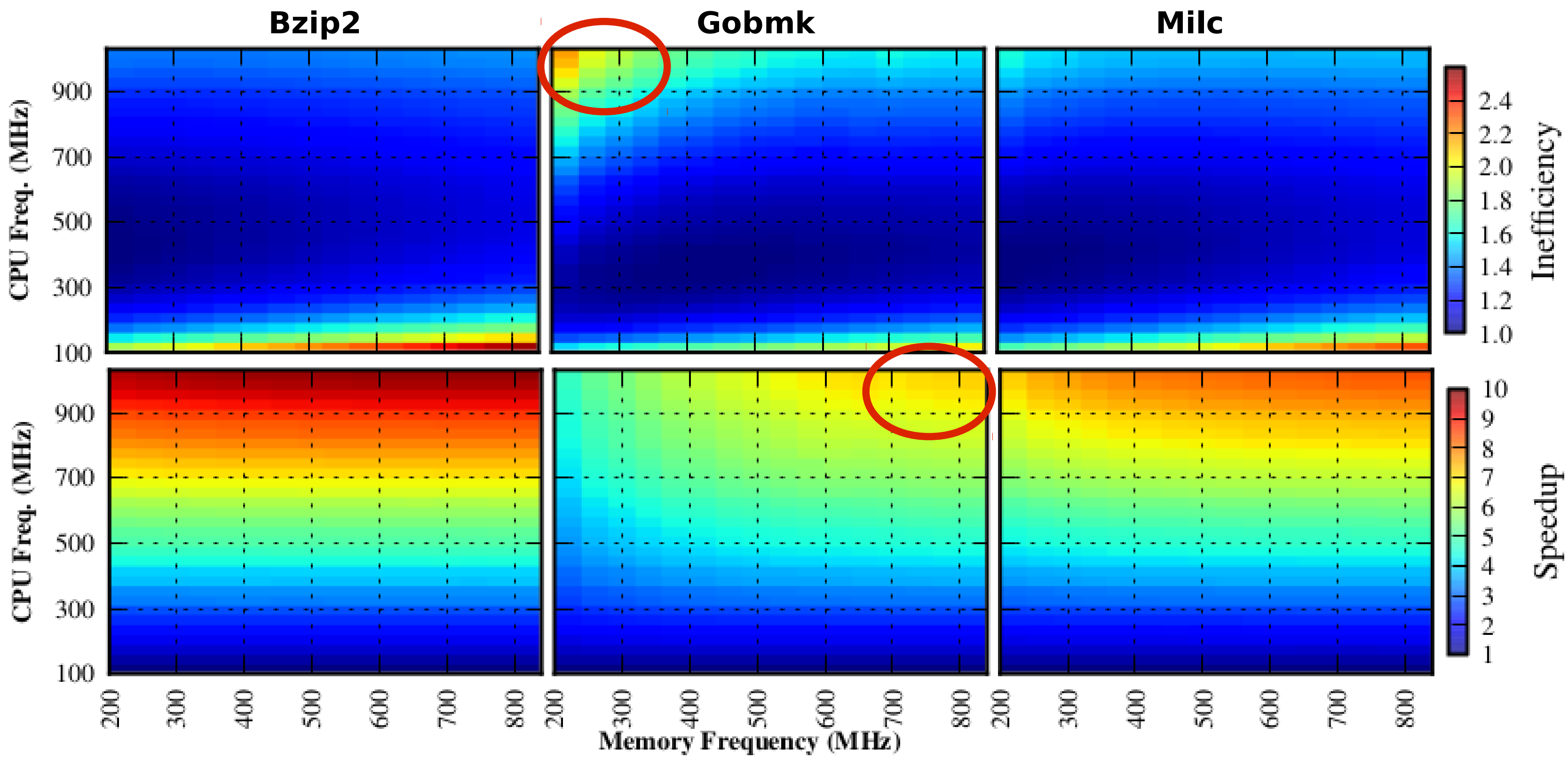


Inefficiency vs. Speedup



➢ Running slower doesn't mean system is running efficiently

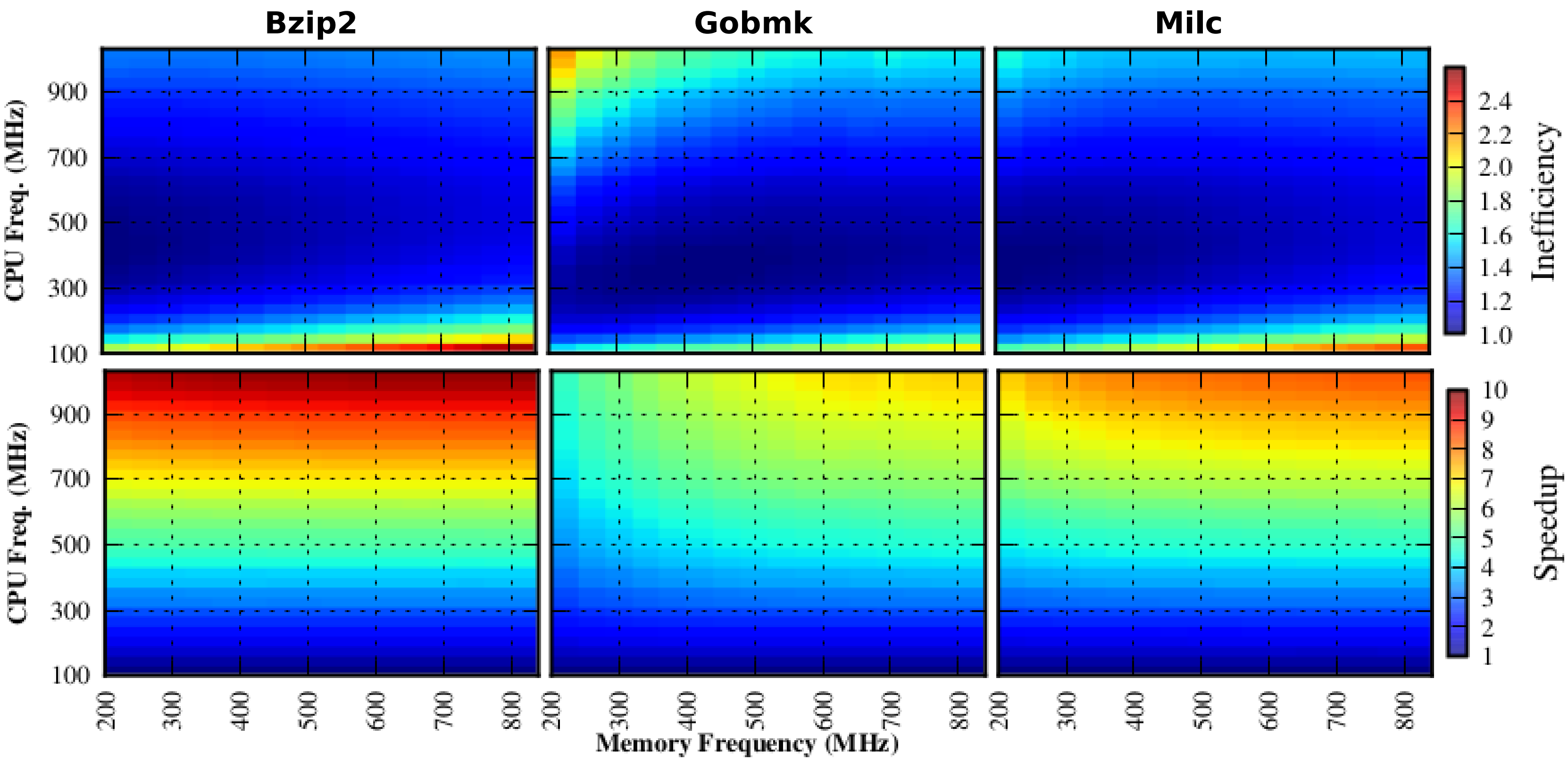
Inefficiency vs. Speedup



➢ Running slower doesn't mean system is running efficiently

➢ Higher inefficiency doesn't always result in higher performance

Inefficiency vs. Speedup



- Running slower doesn't mean system is running efficiently
- Higher inefficiency doesn't always result in higher performance
- Smart algorithms should search for optimal frequency settings ***under*** the inefficiency constraint and ***not*** just ***at*** the inefficiency constraint

Outline

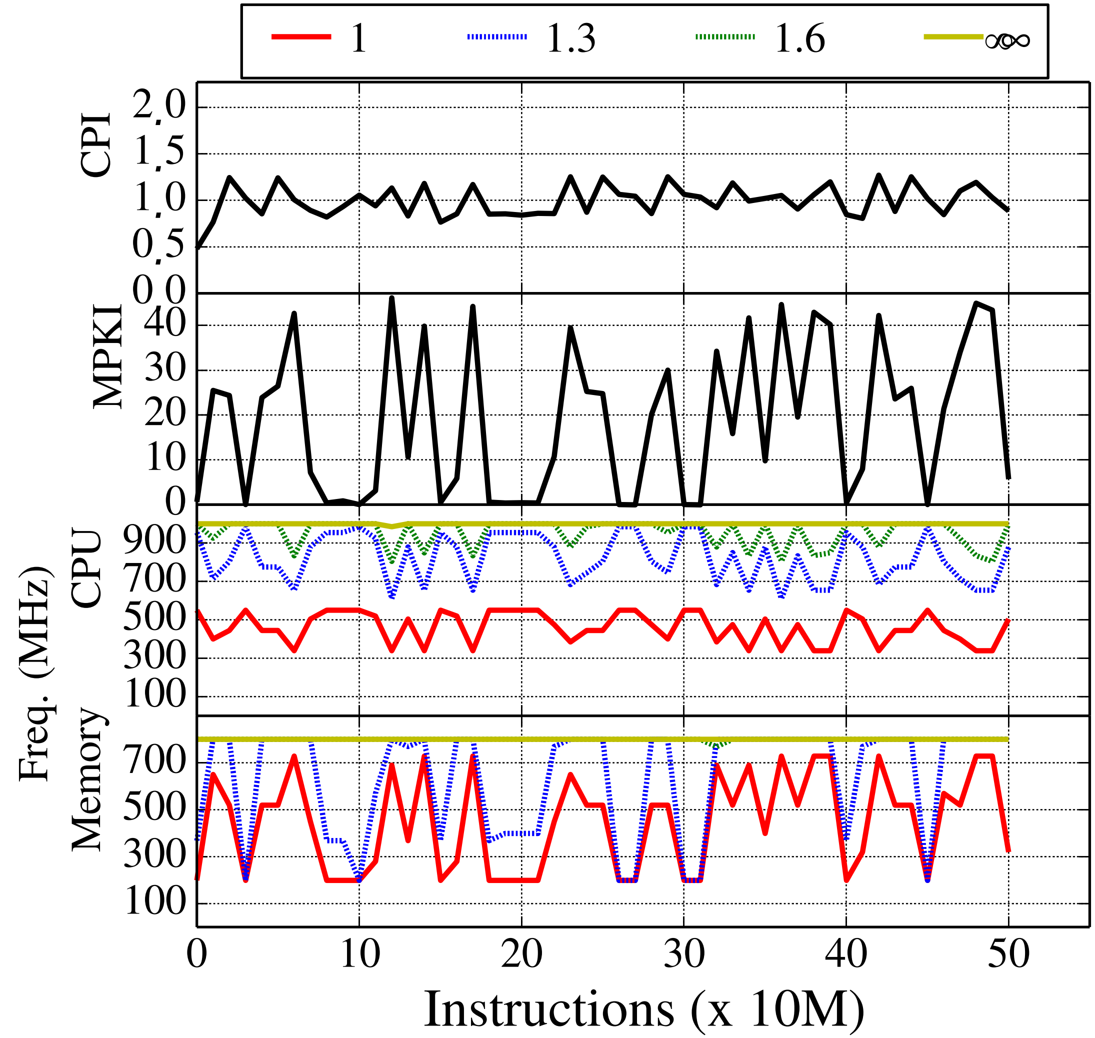
- Inefficiency
- Inefficiency vs. Speedup
- **Characteristics of Optimal Frequency Settings**
- Performance Clusters and Stable Regions
- Conclusions and Future Work

Optimal Frequency Settings

- Deliver best performance under given inefficiency budget

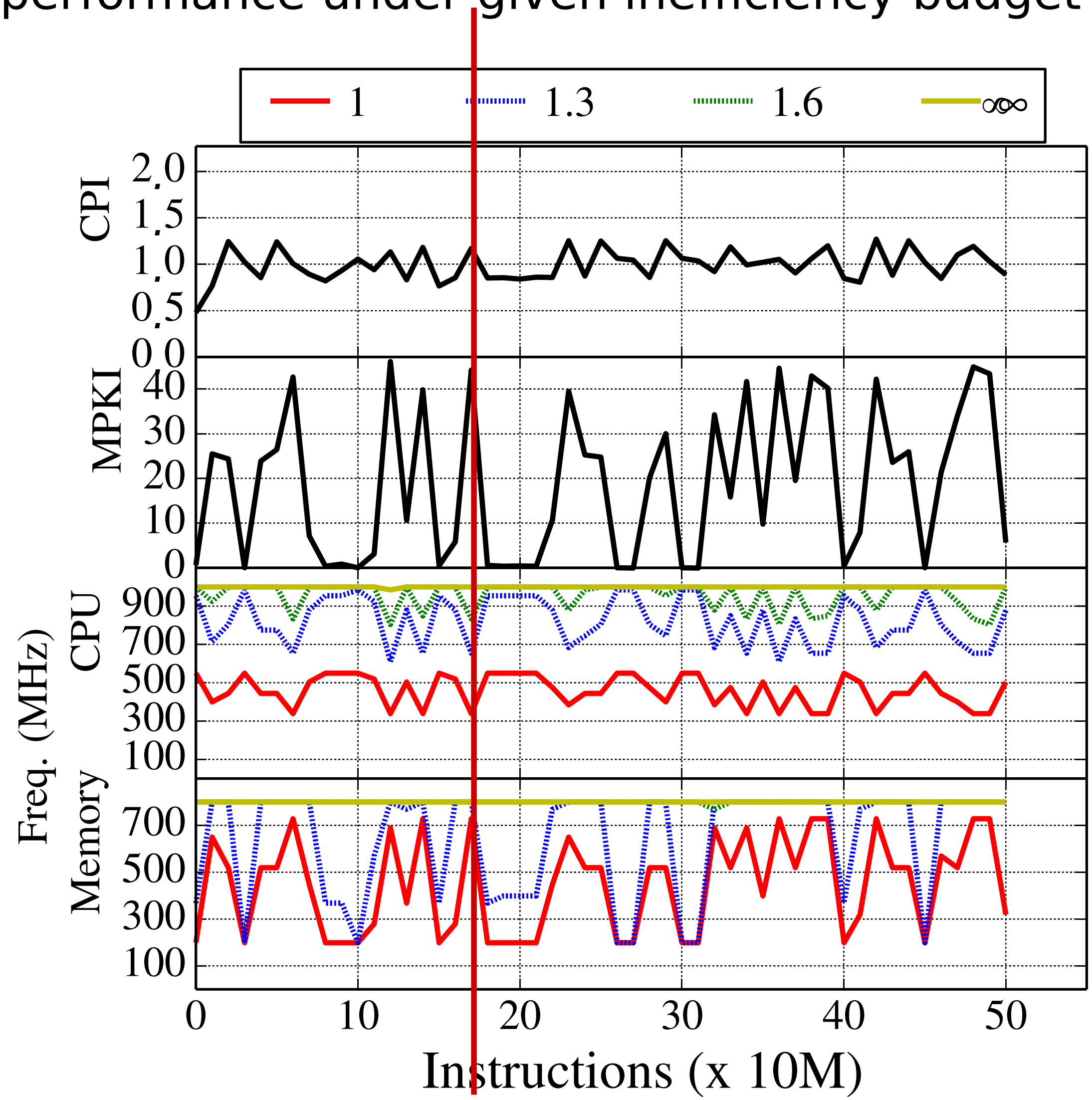
Optimal Frequency Settings

- Deliver best performance under given inefficiency budget



Optimal Frequency Settings

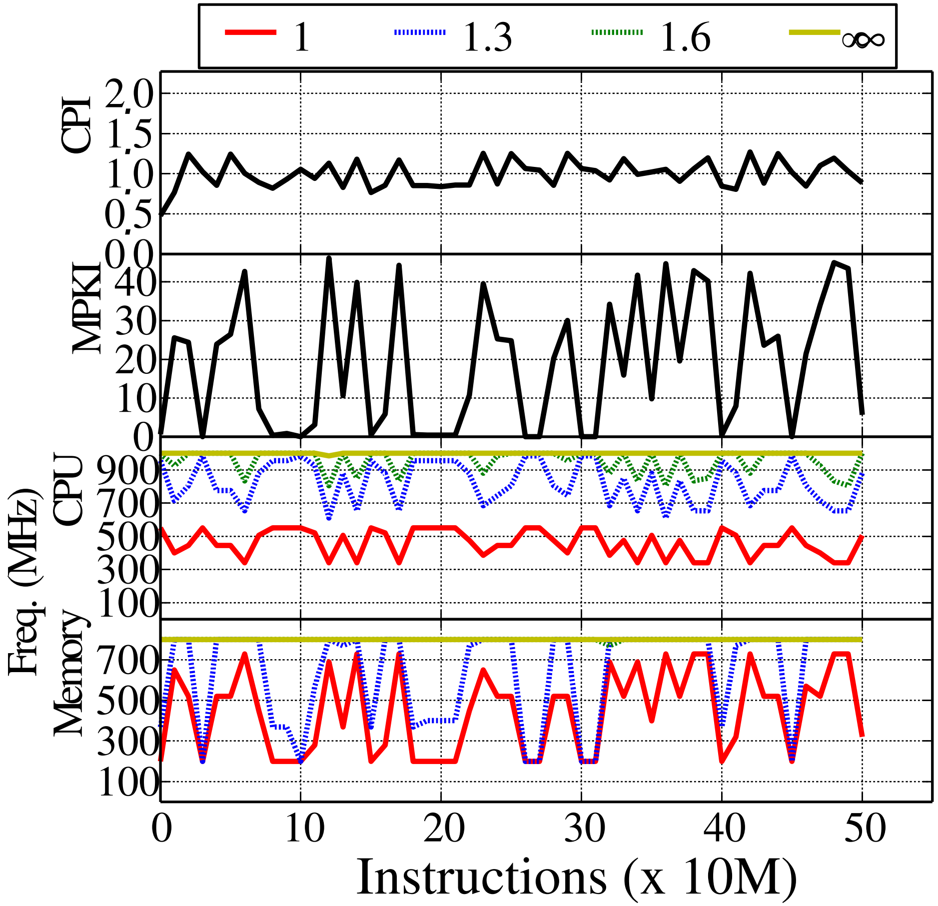
- Deliver best performance under given inefficiency budget



- Higher CPI results in higher memory frequency and lower CPU frequency

Optimal Frequency Settings

- Deliver best performance under given inefficiency budget



- It is expensive
- Limited energy performance trade-off options

Outline

- Inefficiency
- Inefficiency vs. Speedup
- Characteristics of Optimal Frequency Settings
- **Performance Clusters and Stable Regions**
- Conclusions and Future Work

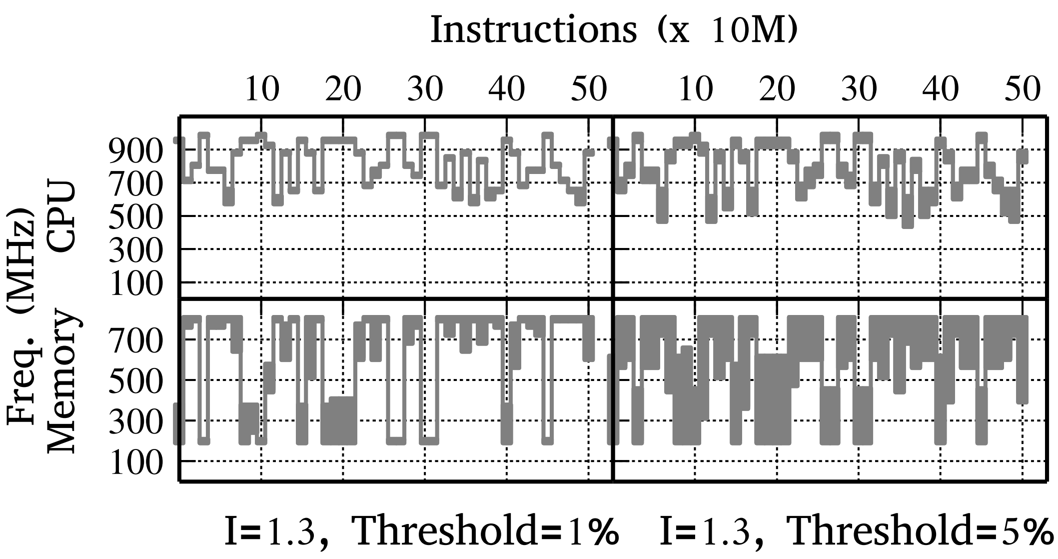
Performance Clusters

Performance Clusters

- *Performance cluster:* Set of frequency settings that have performance within a performance degradation threshold – *cluster threshold* – compared to the optimal performance for a given inefficiency budget.

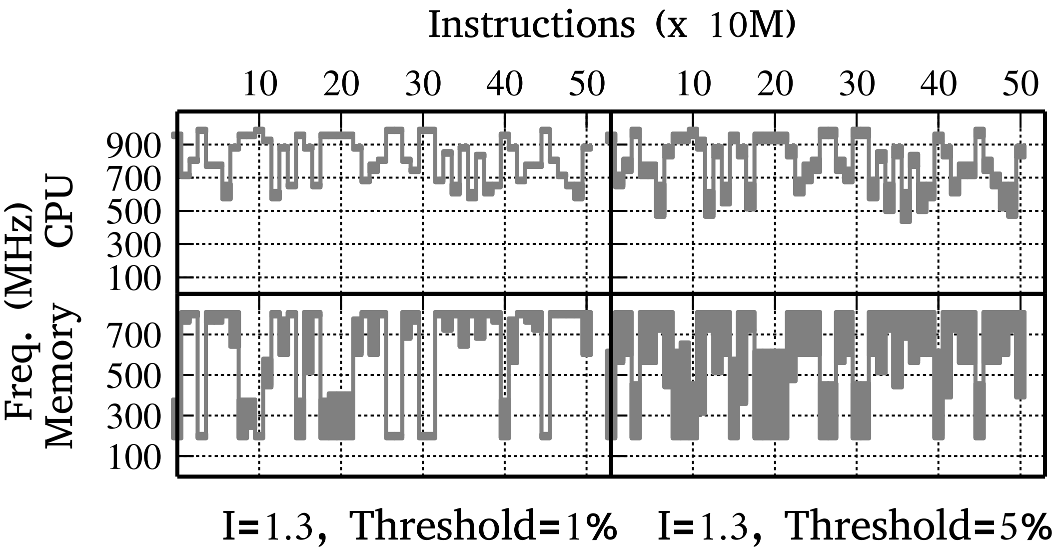
Performance Clusters

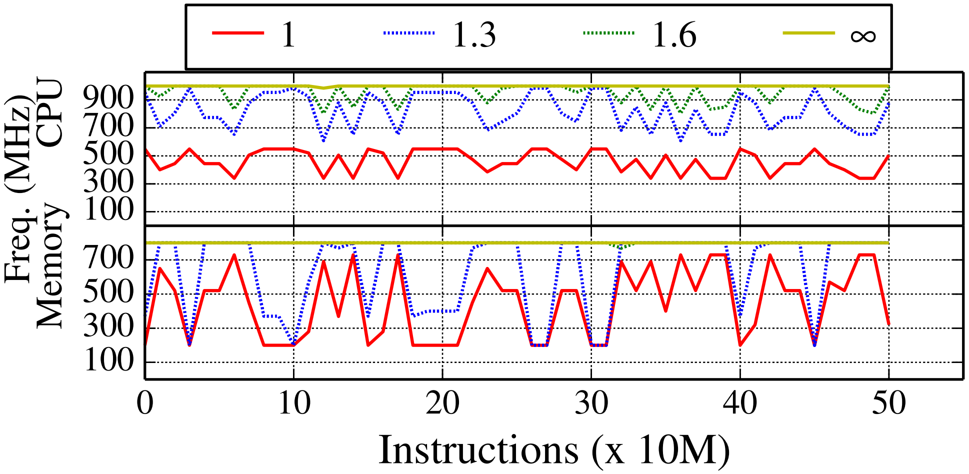
➢ *Performance cluster:* Set of frequency settings that have performance within a performance degradation threshold – *cluster threshold* – compared to the optimal performance for a given inefficiency budget.



Performance Clusters

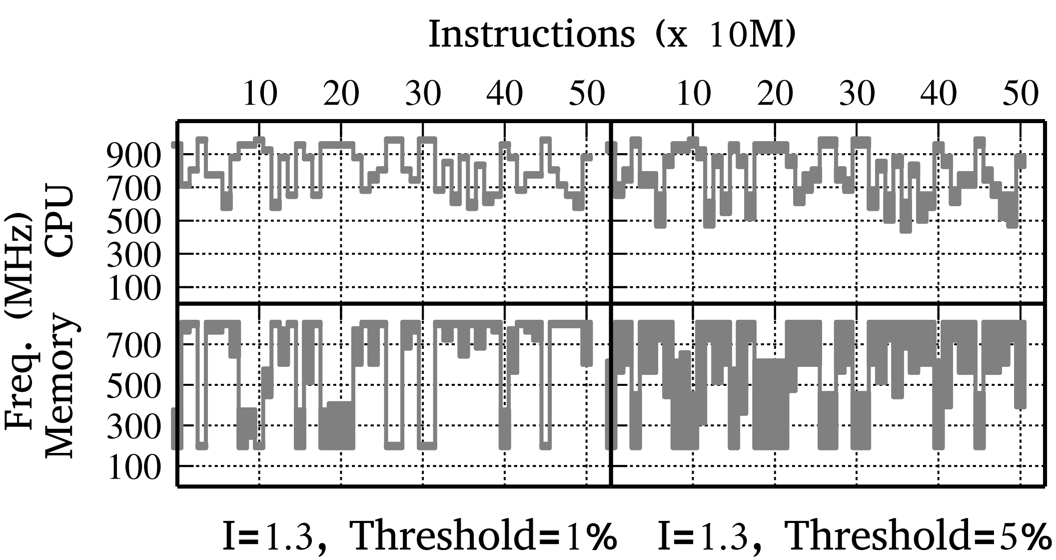
➢ *Performance cluster:* Set of frequency settings that have performance within a performance degradation threshold – *cluster threshold* – compared to the optimal performance for a given inefficiency budget.

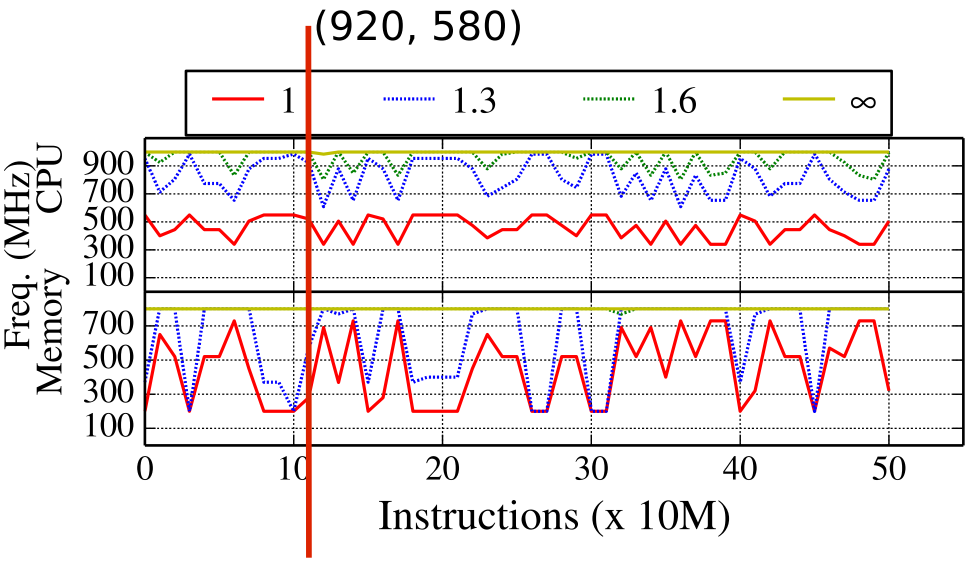




Performance Clusters

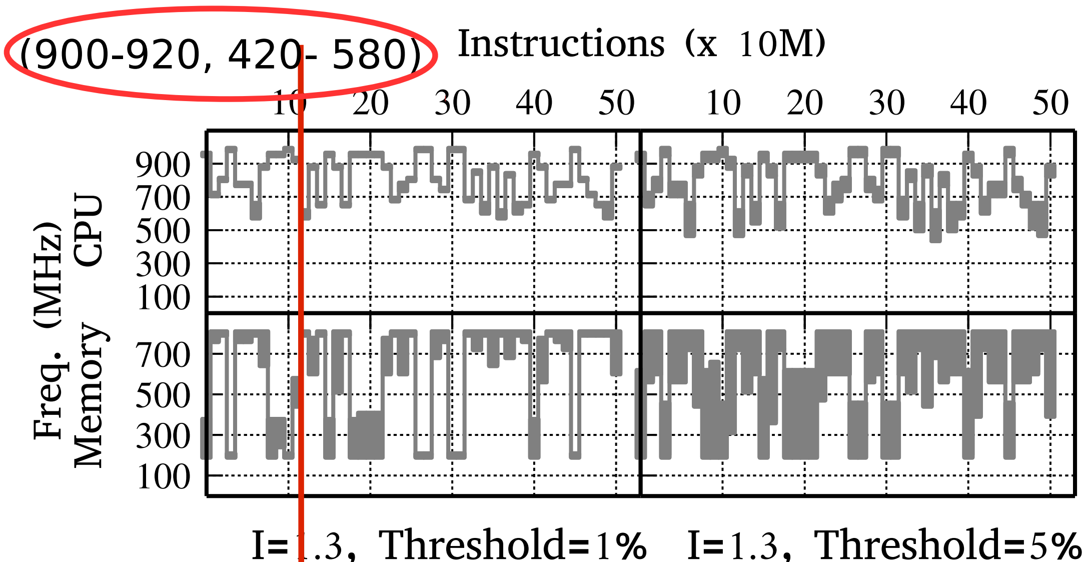
- *Performance cluster:* Set of frequency settings that have performance within a performance degradation threshold – *cluster threshold* – compared to the optimal performance for a given inefficiency budget.

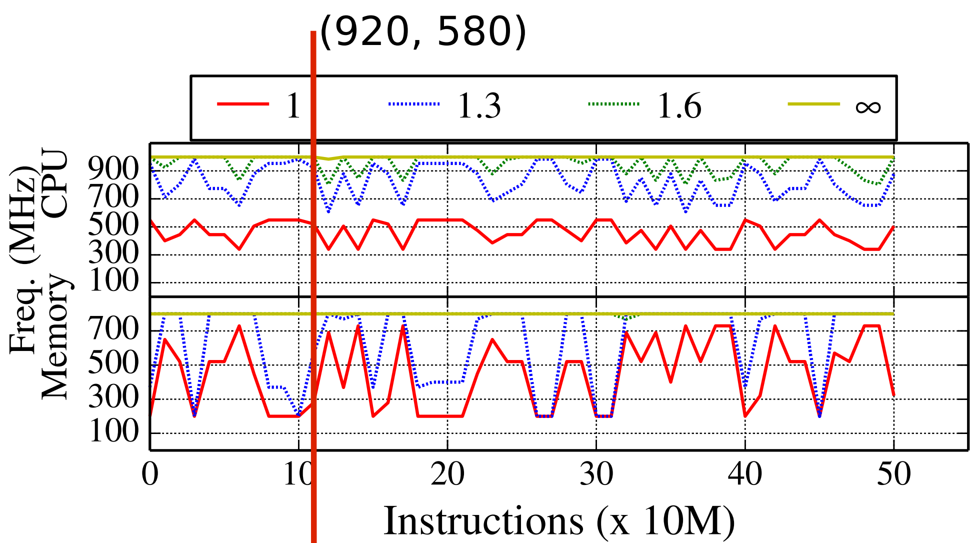




Performance Clusters

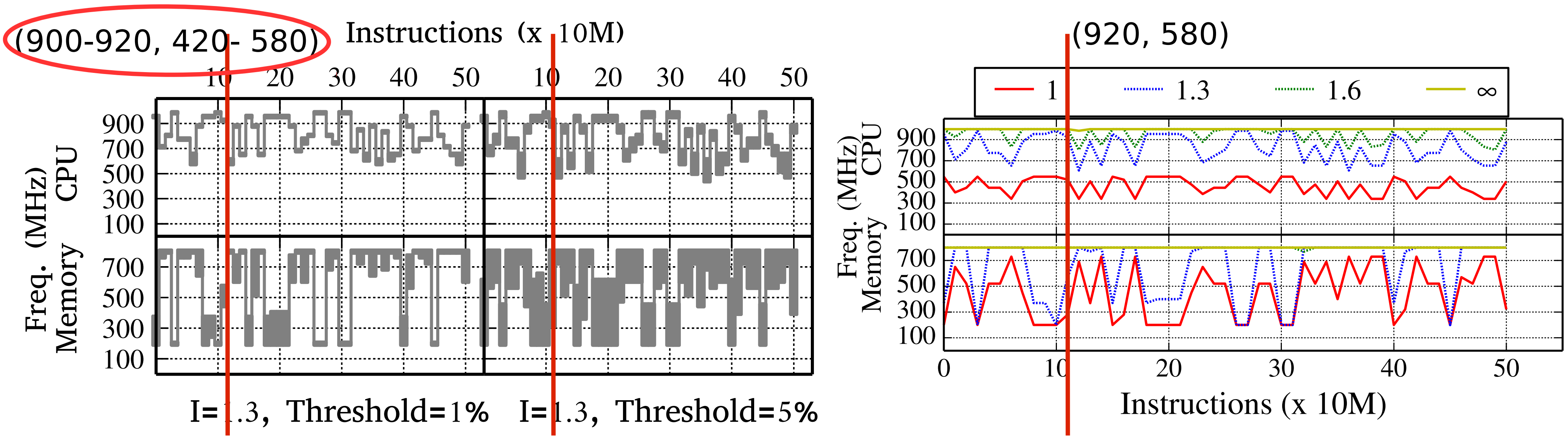
- *Performance cluster:* Set of frequency settings that have performance within a performance degradation threshold – *cluster threshold* – compared to the optimal performance for a given inefficiency budget.





Performance Clusters

- *Performance cluster:* Set of frequency settings that have performance within a performance degradation threshold – *cluster threshold* – compared to the optimal performance for a given inefficiency budget.



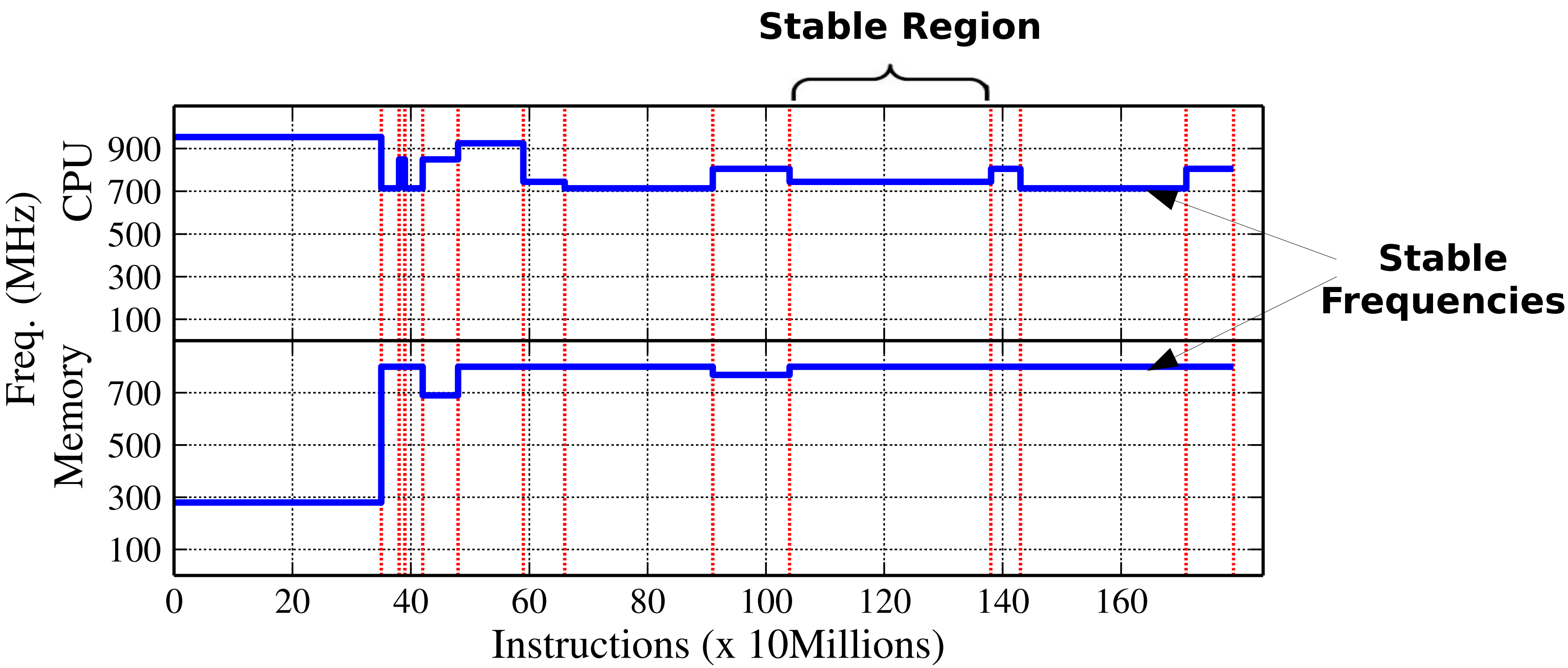
- Higher cluster thresholds result in higher range of available settings

Stable Regions

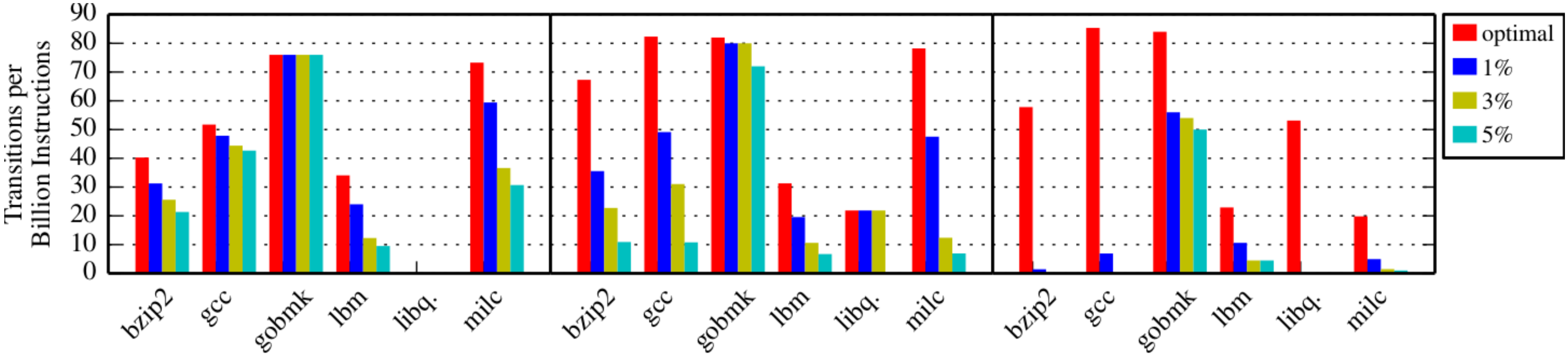
- *Stable regions:* Regions in which atleast one pair of CPU and memory frequency settings is common for all given samples of the region

Stable Regions

- *Stable regions:* Regions in which atleast one pair of CPU and memory frequency settings is common for all given samples of the region

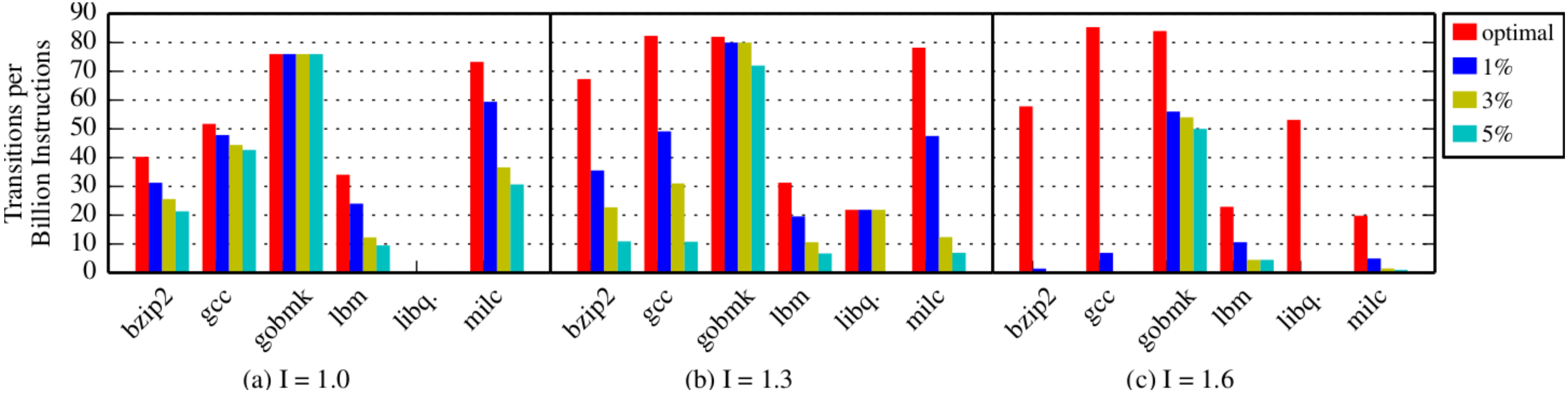


Stable Regions



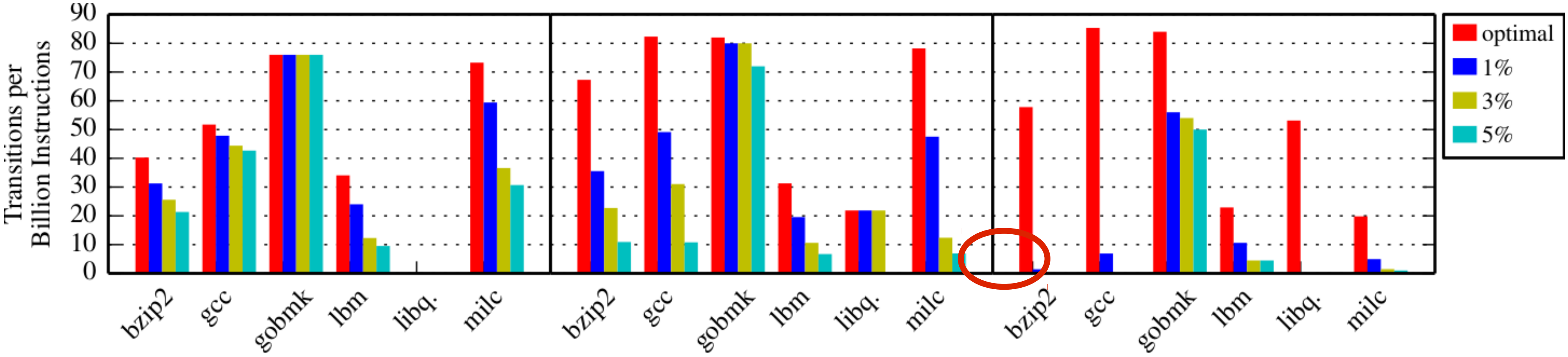
(a) I = 1.0 (b) I = 1.3 (c) I = 1.6

Stable Regions

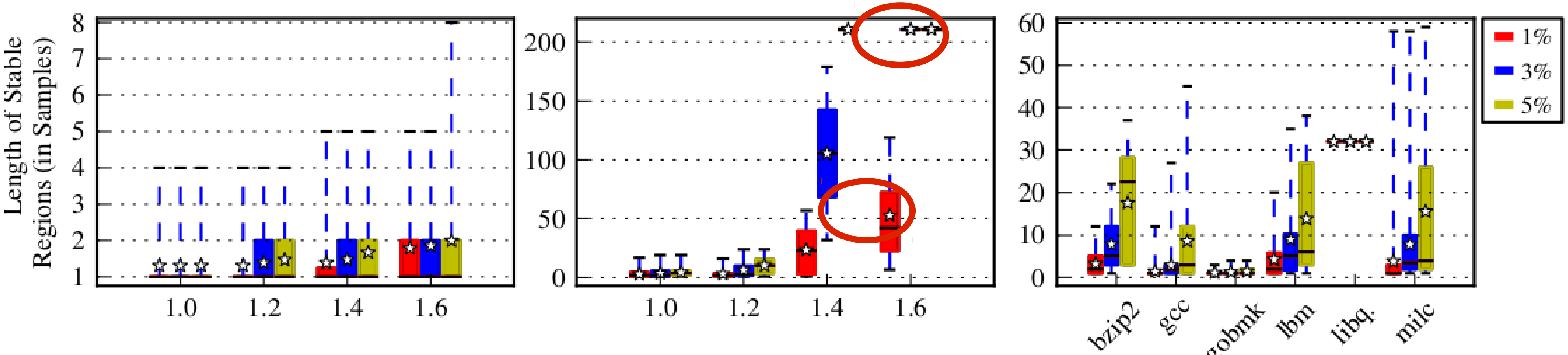


- Number of transition decrease with in increase in cluster thresholds

Stable Regions



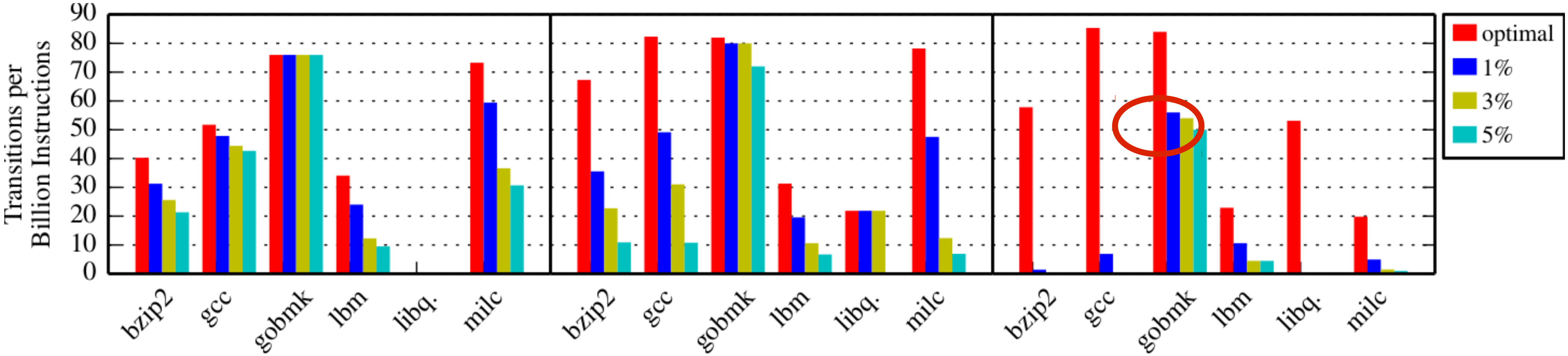
(a) I = 1.0 (b) I = 1.3 (c) I = 1.6



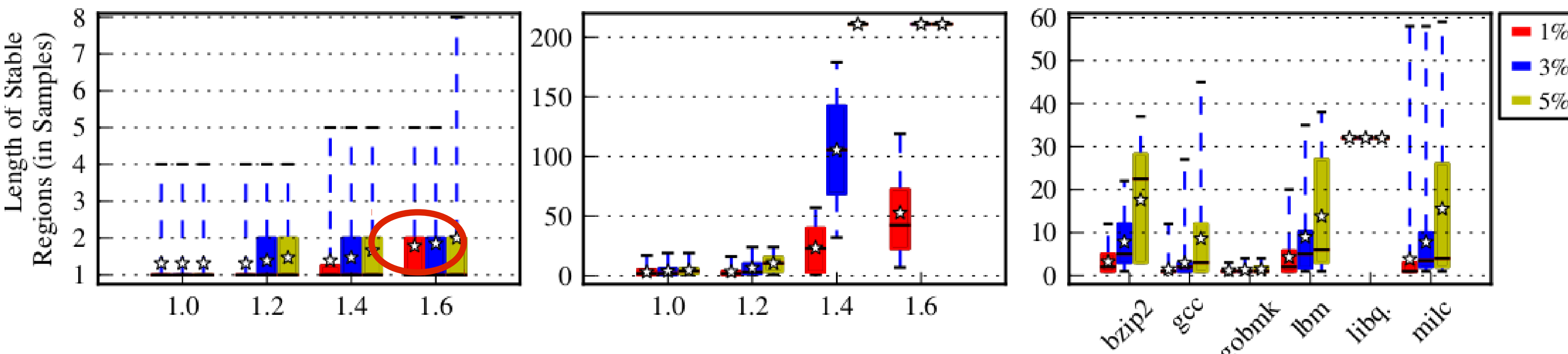
(a) Gobmk (b) Bzip2 (c)

- Number of transition decrease with in increase in cluster thresholds
- For *bzip2,* number of transitions are zero at higher inefficiencies

Stable Regions



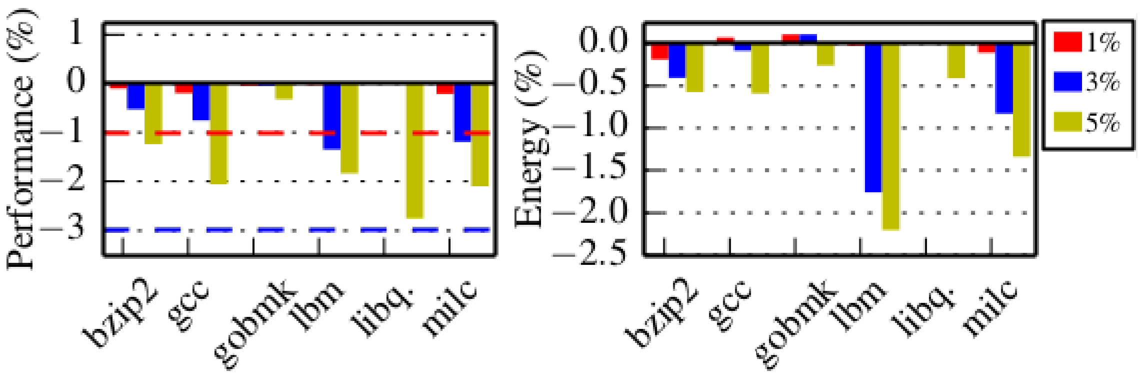
(a) I = 1.0 (b) I = 1.3 (c) I = 1.6



(a) Gobmk (b) Bzip2 (c)

- Number of transition decrease with in increase in cluster thresholds
- For *bzip2,* number of transitions are zero at higher inefficiencies
- *Rapidly changing phases of gobmk* result in only a slight decrease in number of transitions with cluster threshold

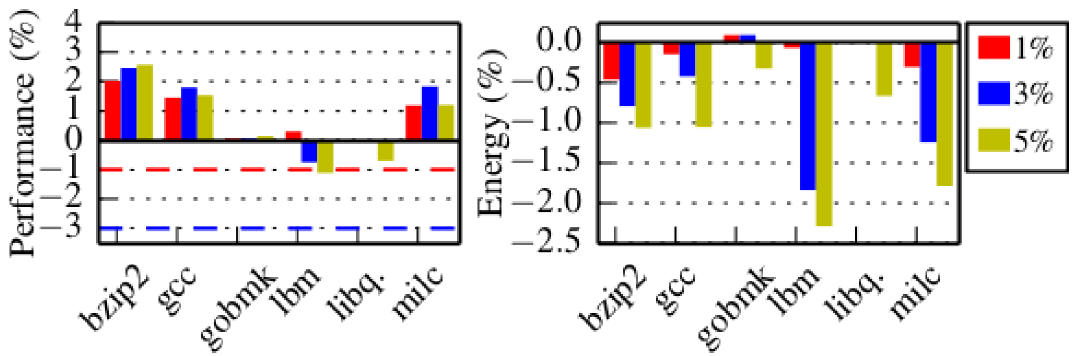
Energy, Performance Results



No Tuning Overhead

- Performance drop is within bounds --- always
- Energy consumption decreases with increase in cluster threshold --- lower frequency settings are selected

Energy, Performance Results



With Tuning Overhead

- Performance improves when tuning overhead is included --- due to decrease in number of transitions

Outline

- Inefficiency
- Inefficiency vs. Speedup
- Characteristics of Optimal Frequency Settings
- Performance Clusters and Stable Regions
- **Conclusions and Future Work**

Conclusion & Future Work

- Inefficiency

Conclusion & Future Work

- Inefficiency
- Inefficiency and performance trade-offs ---CPU DVFS and memory DFS

Conclusion & Future Work

- Inefficiency
- Inefficiency and performance trade-offs ---CPU DVFS and memory DFS
- Tracking optimal frequency settings is expensive

Conclusion & Future Work

- Inefficiency
- Inefficiency and performance trade-offs ---CPU DVFS and memory DFS
- Tracking optimal frequency settings is expensive
- Performance clusters and stable regions help reduce the cost of frequent tuning

Conclusion & Future Work

- Inefficiency
- Inefficiency and performance trade-offs ---CPU DVFS and memory DFS
- Tracking optimal frequency settings is expensive
- Performance clusters and stable regions help reduce the cost of frequent tuning
- We are building a system that is capable of tuning multiple components simultaneously while executing applications using the models and analysis of the performance clusters.

Questions?

References

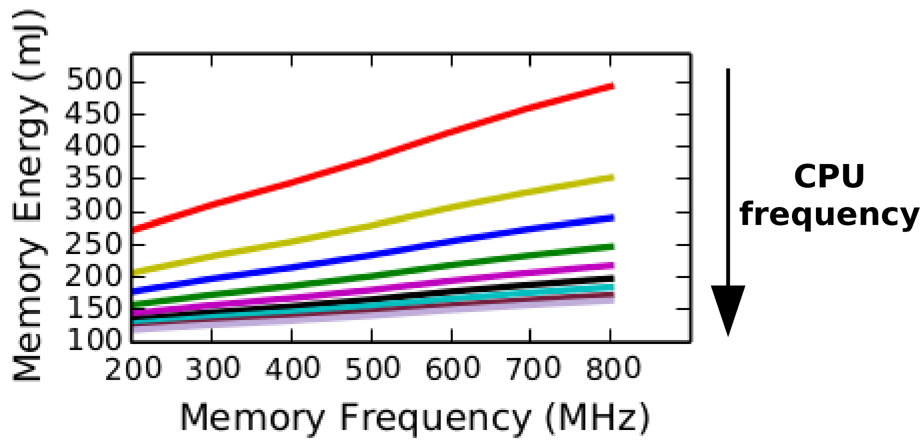
1) Nachiappan, Nachiappan Chidambaram, et al. "Domain knowledge based energy management in handhelds." High Performance Computer Architecture (HPCA), 2015 IEEE 21st International Symposium on. IEEE, 2015.
2) Deng, Qingyuan, et al. "Memscale: active low-power modes for main memory." ACM SIGARCH Computer Architecture News 39.1 (2011): 225-238.
3) David, Howard, et al. "Memory power management via dynamic voltage/frequency scaling." Proceedings of the 8th ACM international conference on Autonomic computing. ACM, 2011.
4) Deng, Qingyuan, et al. "Coscale: Coordinating cpu and memory system dvfs in server systems." Microarchitecture (MICRO), 2012 45th Annual IEEE/ACM International Symposium on. IEEE, 2012.
5) Rumble, Stephen M., et al. "Apprehending joule thieves with cinder." ACM SIGCOMM Computer Communication Review 40.1 (2010): 106-111.
6) Binkert, Nathan, et al. "The gem5 simulator." ACM SIGARCH Computer Architecture News 39.2 (2011): 1-7.
7) Pandaboard, http://pandaboard.org/content/platform.
8) Micron, “Calculating Memory System Power for LPDDR2, May 2013.”
9) Isci, Canturk, Alper Buyuktosunoglu, and Margaret Martonosi. "Long-term workload phases: Duration predictions and applications to DVFS." Micro, IEEE 25.5 (2005): 39-51.
10) Lau, Jeremy, Erez Perelman, and Brad Calder. "Selecting software phase markers with code structure analysis." Proceedings of the International Symposium on Code Generation and Optimization. IEEE Computer Society, 2006.

CPU DVFS and DRAM DFS

- The interplay of performance and energy consumption of CPU and DRAM frequency scaling is complex.

CPU DVFS and DRAM DFS

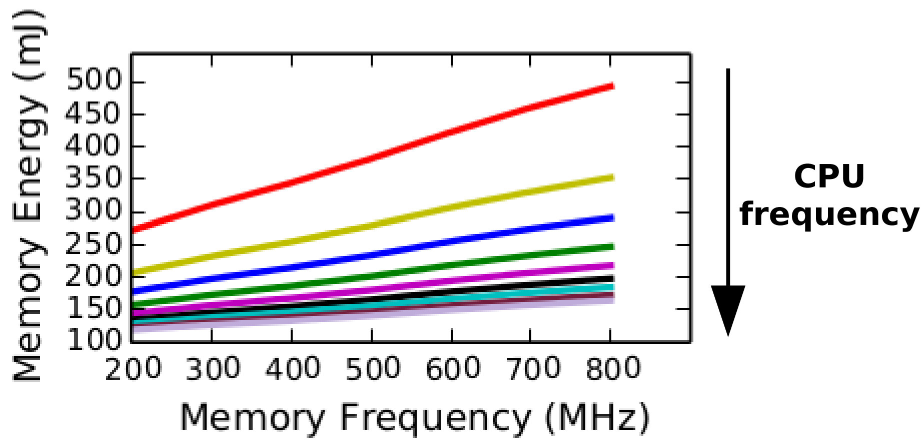
- The interplay of performance and energy consumption of CPU and DRAM frequency scaling is complex.



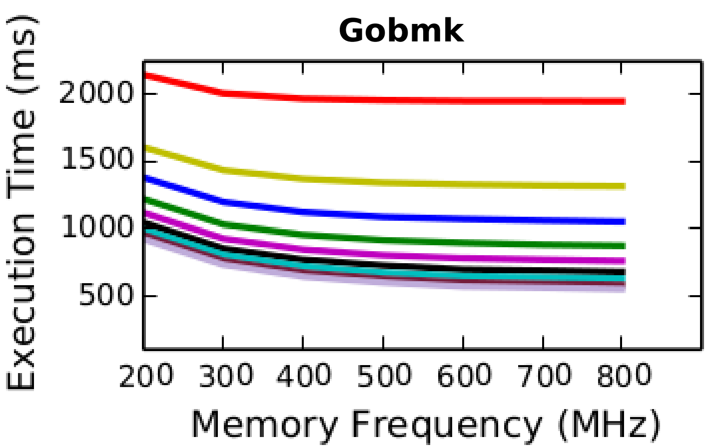
- Increase in DRAM energy is a function of CPU frequency.

CPU DVFS and DRAM DFS

- The interplay of performance and energy consumption of CPU and DRAM frequency scaling is complex.

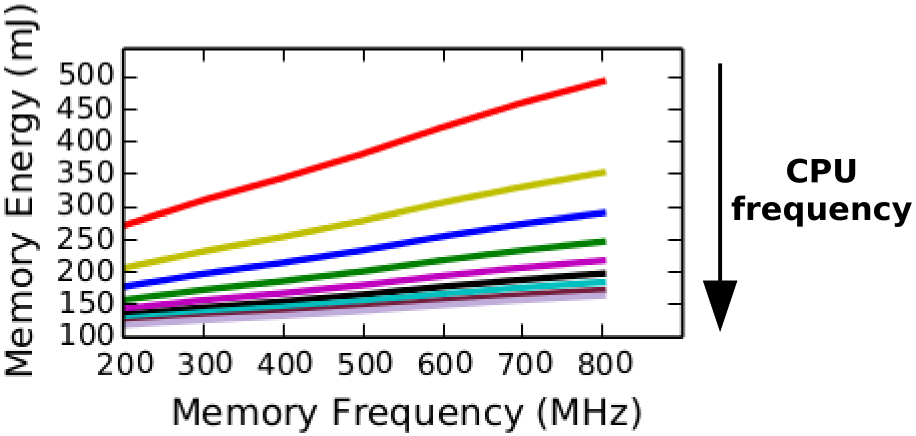


- Increase in DRAM energy is a function of CPU frequency.

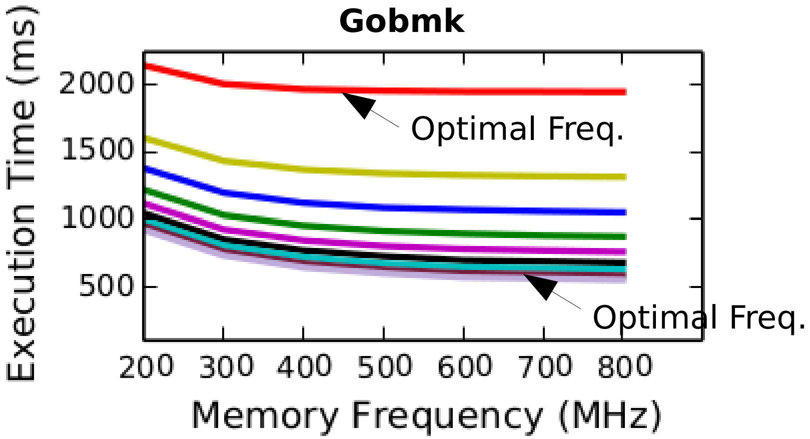


CPU DVFS and DRAM DFS

- The interplay of performance and energy consumption of CPU and DRAM frequency scaling is complex.

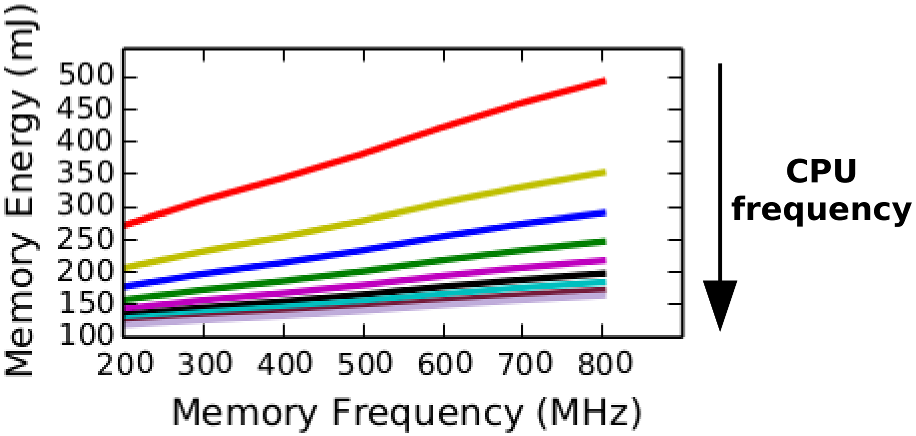


- Increase in DRAM energy is a function of CPU frequency.

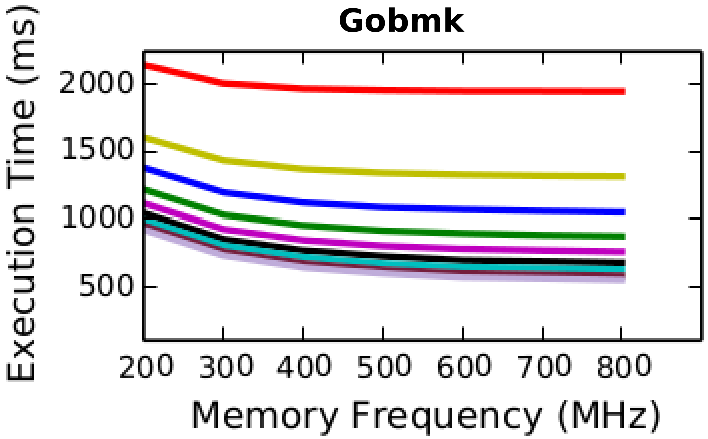


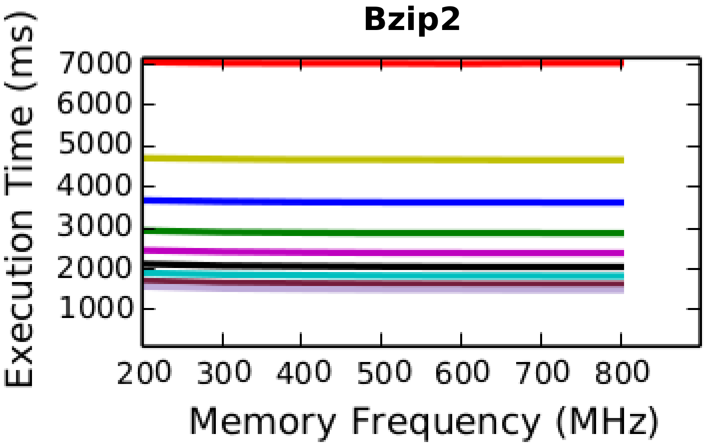
CPU DVFS and DRAM DFS

- The interplay of performance and energy consumption of CPU and DRAM frequency scaling is complex.



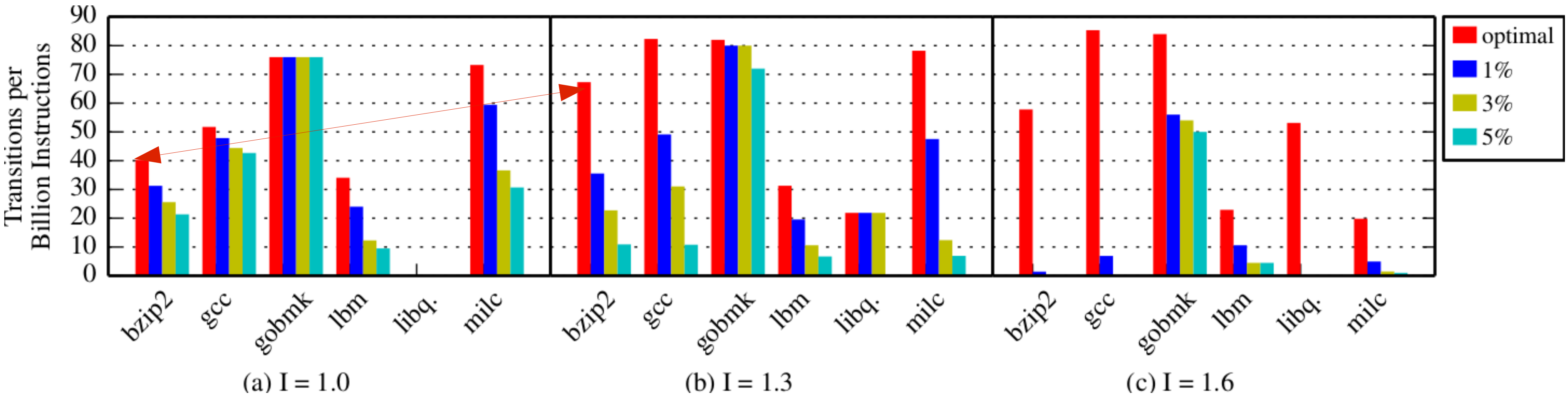
- Increase in DRAM energy is a function of CPU frequency.





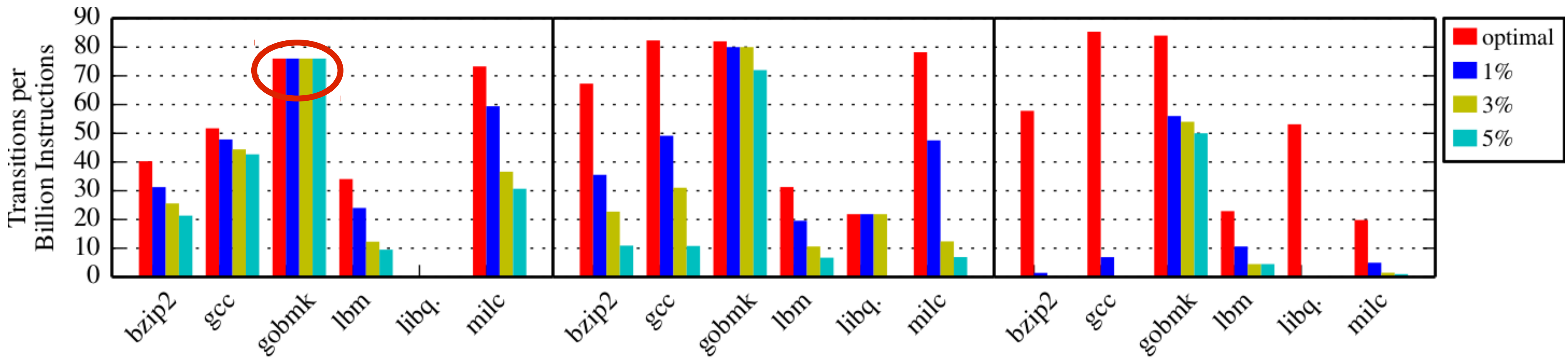
- Performance improvement with DRAM frequency varies across applications.

Stable Regions

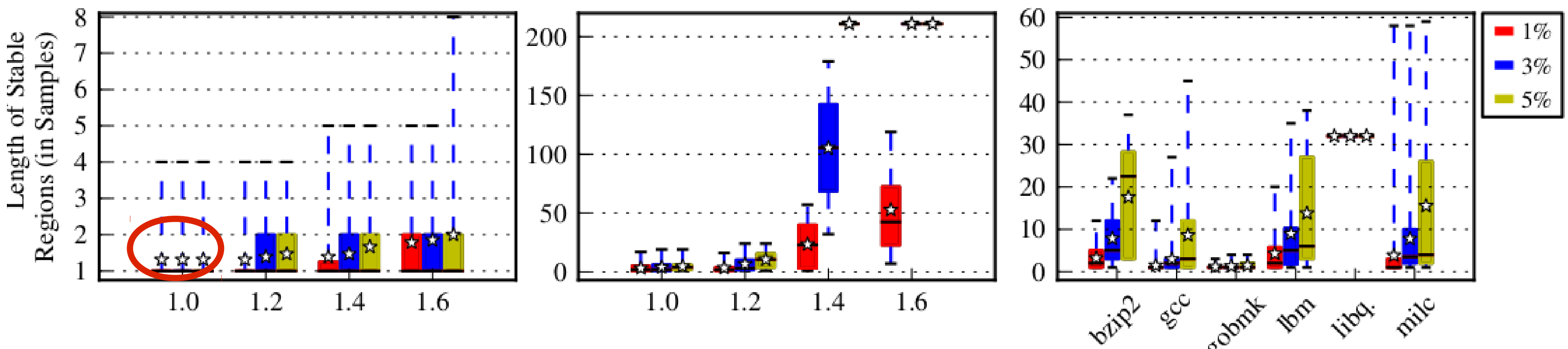


- Number of transition decrease with in increase in cluster thresholds
- For *bzip2,* number of transitions to track optimal settings increase with inefficiency

Stable Regions



(a) I = 1.0 (b) I = 1.3 (c) I = 1.6

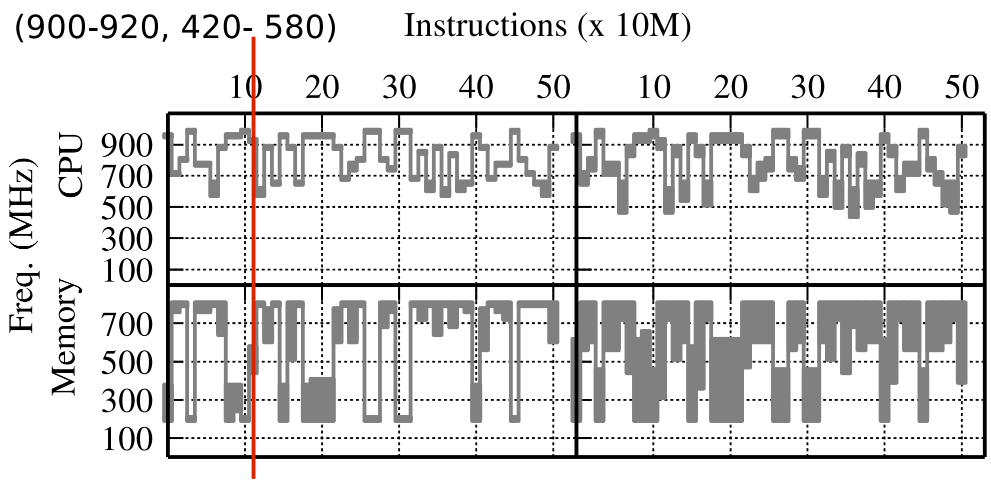


(a) Gobmk (b) Bzip2 (c)

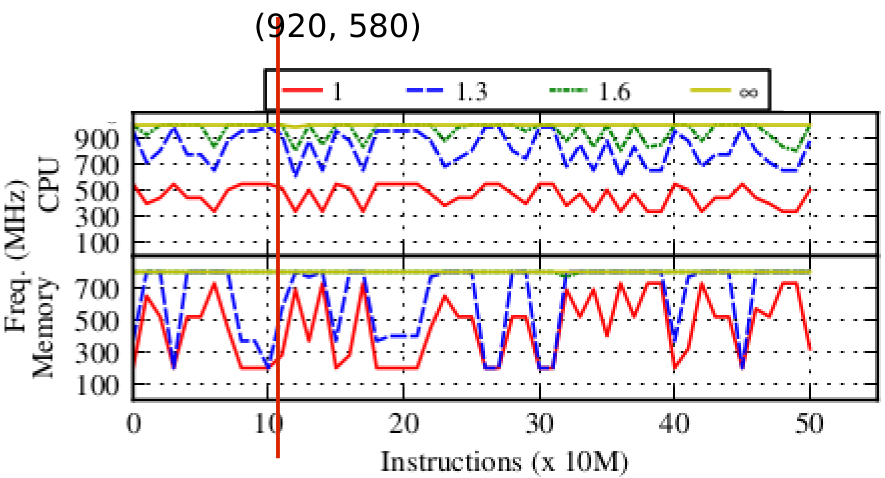
- Number of transition decrease with in increase in cluster thresholds
- For *bzip2*, number of transitions to track optimal settings increase with inefficiency

Performance Clusters

➢ *Performance cluster:* Set of frequency settings that have performance within a performance degradation threshold – *cluster threshold* – compared to the optimal performance for a given inefficiency budget.

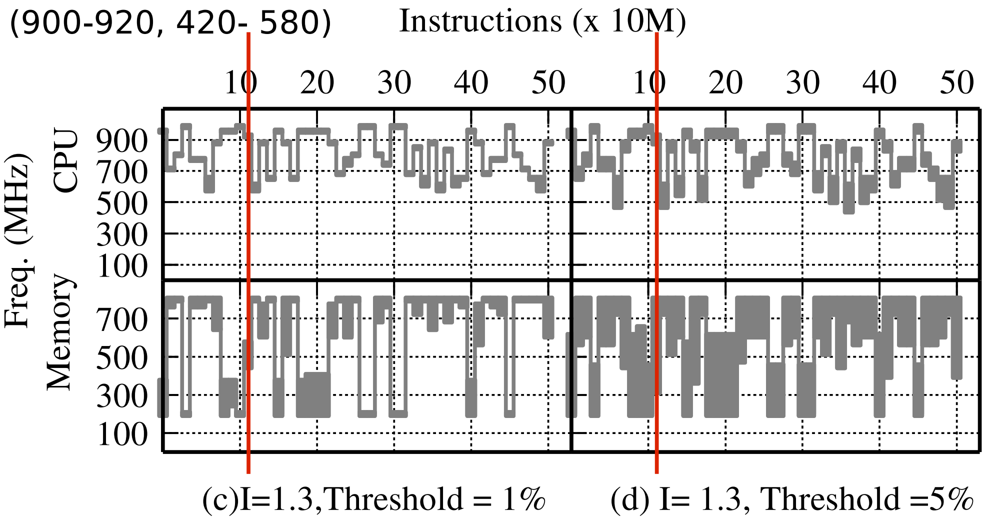


(c)I=1.3,Threshold = 1% (d) I= 1.3, Threshold =5%

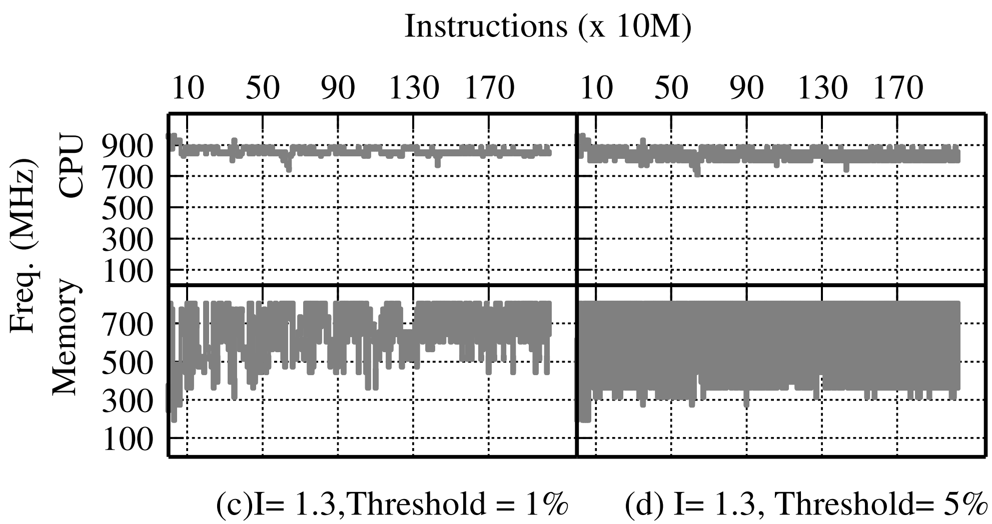


Performance Clusters

➢ *Performance cluster:* Set of frequency settings that have performance within a performance degradation threshold – *cluster threshold* – compared to the optimal performance for a given inefficiency budget.

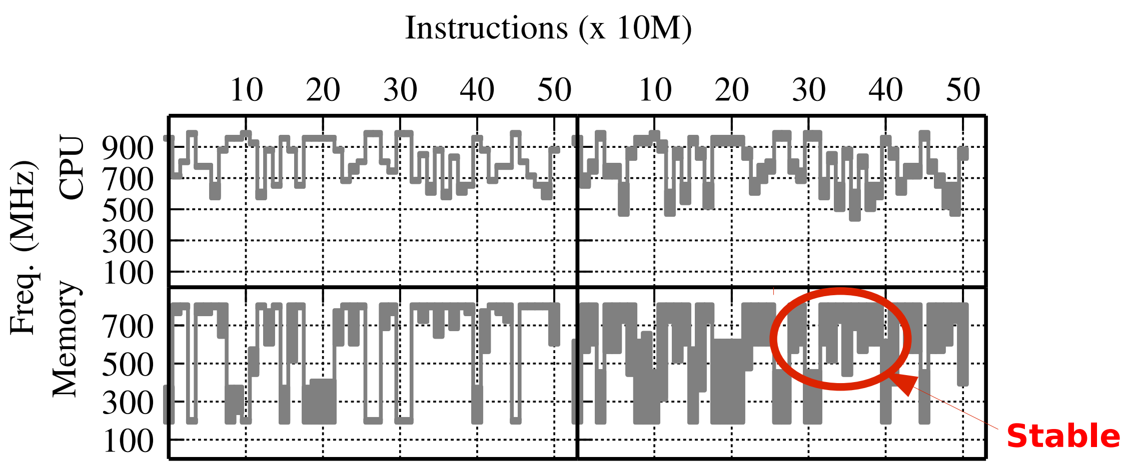


Gobmk



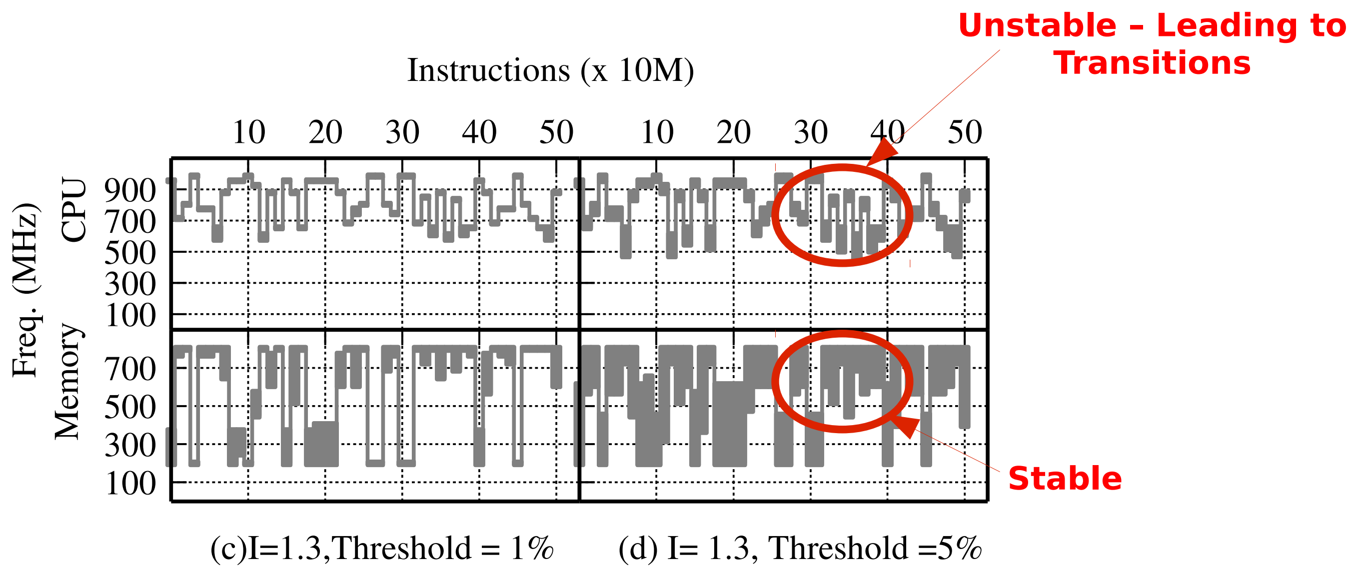
Milc

Performance Clusters



(c)I=1.3,Threshold = 1%　　(d) I= 1.3, Threshold =5%

Performance Clusters



➢ System doesn't need to transition to new set of frequencies only when ***both*** CPU and DRAM frequencies are stable.

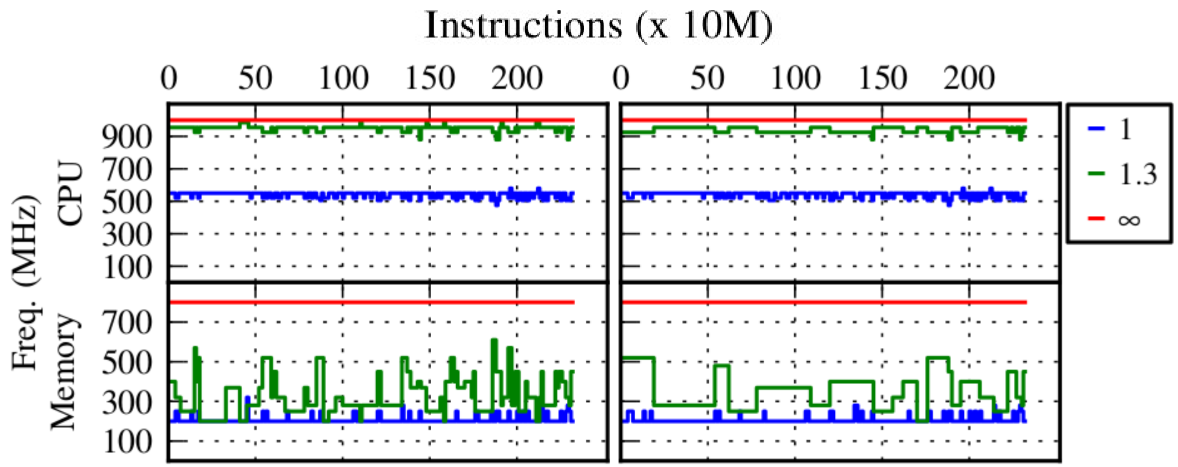
Algorithm Implications

- How do higher cluster thresholds and stable regions help energy management algorithms?
  - Algorithms with no knowledge of length of stable regions run periodically to find optimal settings --- high overhead

Algorithm Implications

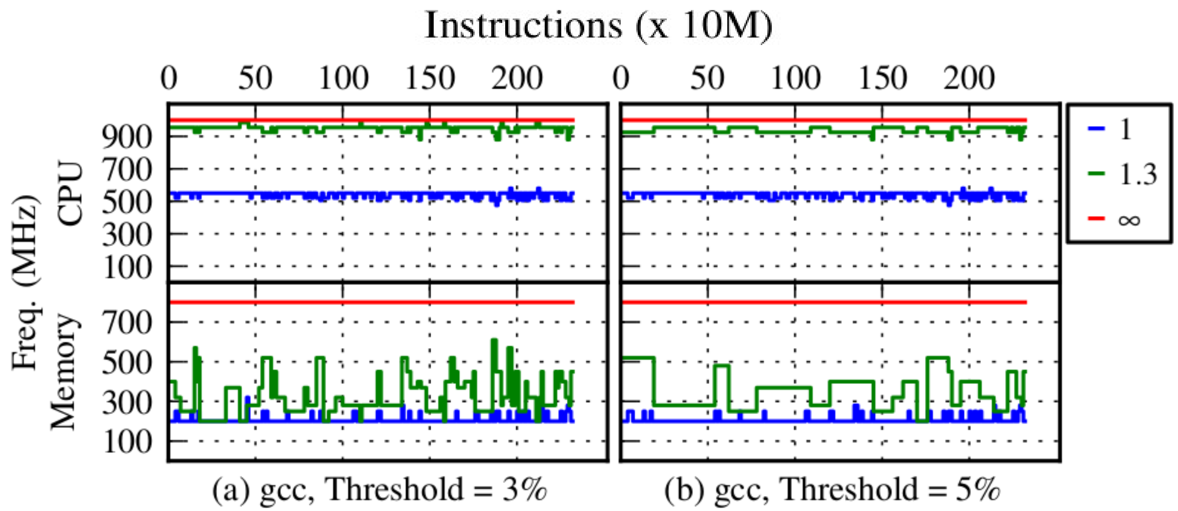
- How do higher cluster thresholds and stable regions help energy management algorithms?
  - Algorithms with no knowledge of length of stable regions run periodically to find optimal settings --- high overhead
  - Smart algorithms may choose to tune less often by predicting the length of stable regions --- simple prediction mechanisms as proposed by Isci et. al[9] can be used

Stable Regions



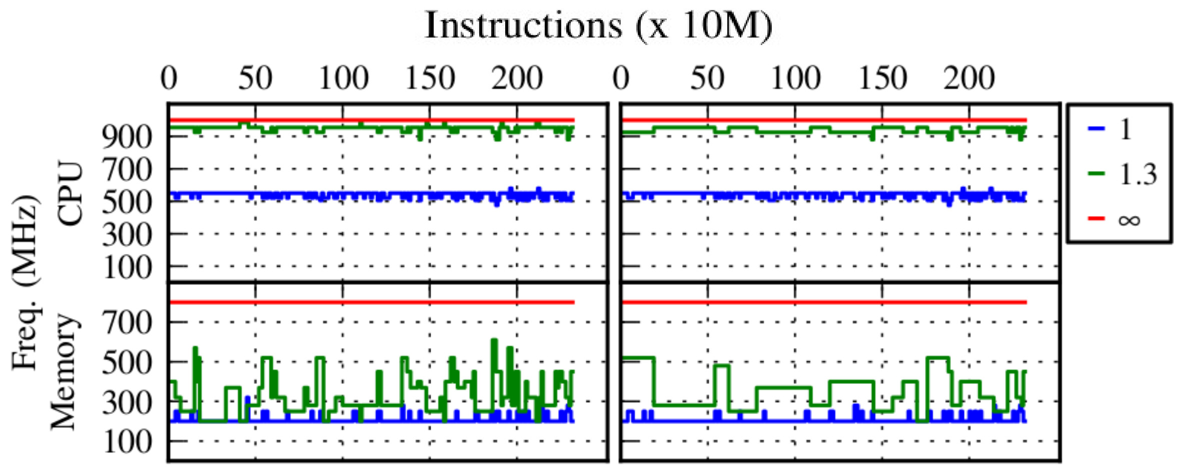
(a) gcc, Threshold = 3%
(b) gcc, Threshold = 5%

Stable Regions



- Number of transition made by gcc drop significantly at lower inefficiencies --- Higher inefficiencies allow the system to choose max-max always

Stable Regions



(a) gcc, Threshold = 3% (b) gcc, Threshold = 5%

- Number of transition made by gcc drop significantly at lower inefficiencies --- Higher inefficiencies allow the system to choose max-max always
- Increase in inefficiency also decreases the transitions --- function of application characteristics

Outline

- Inefficiency
- Inefficiency vs. Speedup
- Characteristics of Optimal Frequency Settings
- Performance Clusters and Stable Regions
- **Algorithm Implications**
- Conclusions and Future Work

Algorithm Implications

- How do higher cluster thresholds and stable regions help energy management algorithms?

Algorithm Implications

- How do higher cluster thresholds and stable regions help energy management algorithms?
  - Algorithms with no knowledge of length of stable regions run periodically to find optimal settings --- high overhead
  - Smart algorithms may choose to tune less often by predicting the length of stable regions --- simple prediction mechanisms as proposed by Isci et. al[1] can be used
  - Offline profiling of applications[2] helps in pre determining the length and position of stable regions --- can be extended to other applications with similar phases

1) Isci et. al Micro 2005
2) Lau et. al International Symposium on Code Generation and Optimization. 2006.