

# Self Calibrating Processor Speed: A New Feedback Loop For Dynamic Voltage Scaling Control

PhD Final Defense

May 14, 2007

Vasanth Venkatachalam

Commitee Members: Michael Franz (Chair),  
Nikil Dutt, Tony Givargis, Christian Probst

# Broad Research Area

- **system-level** approaches to managing power and energy consumption
- **very important** issue that affects electronic devices of all shapes and sizes
- **every percent** of power saved is good

# Dynamic voltage scaling (DVS)

- processor with discrete number of “power settings”
  - Intel Centrino, AMD Athlon
- the setting controls the CPU speed and voltage
- adjust the setting through software
  - OS, compiler. VM..

# Why does DVS work?

- power **quadratic** in voltage and **linear** in clock frequency
- **cubic** power reduction ideally
- **how to minimize performance loss?**

# Memory Boundedness

- **memory (disk) bounded** programs less performance loss than **CPU bounded** programs
- typically **memory boundedness** inferred from hardware counter events
  - cache misses, memory bus transactions...

# Limitations of Previous Approaches

- events measured may be **statistically** correlated but **not perfectly** so
- execution time depends on **everything** that is happening in the system

# My Approach

# My Approach

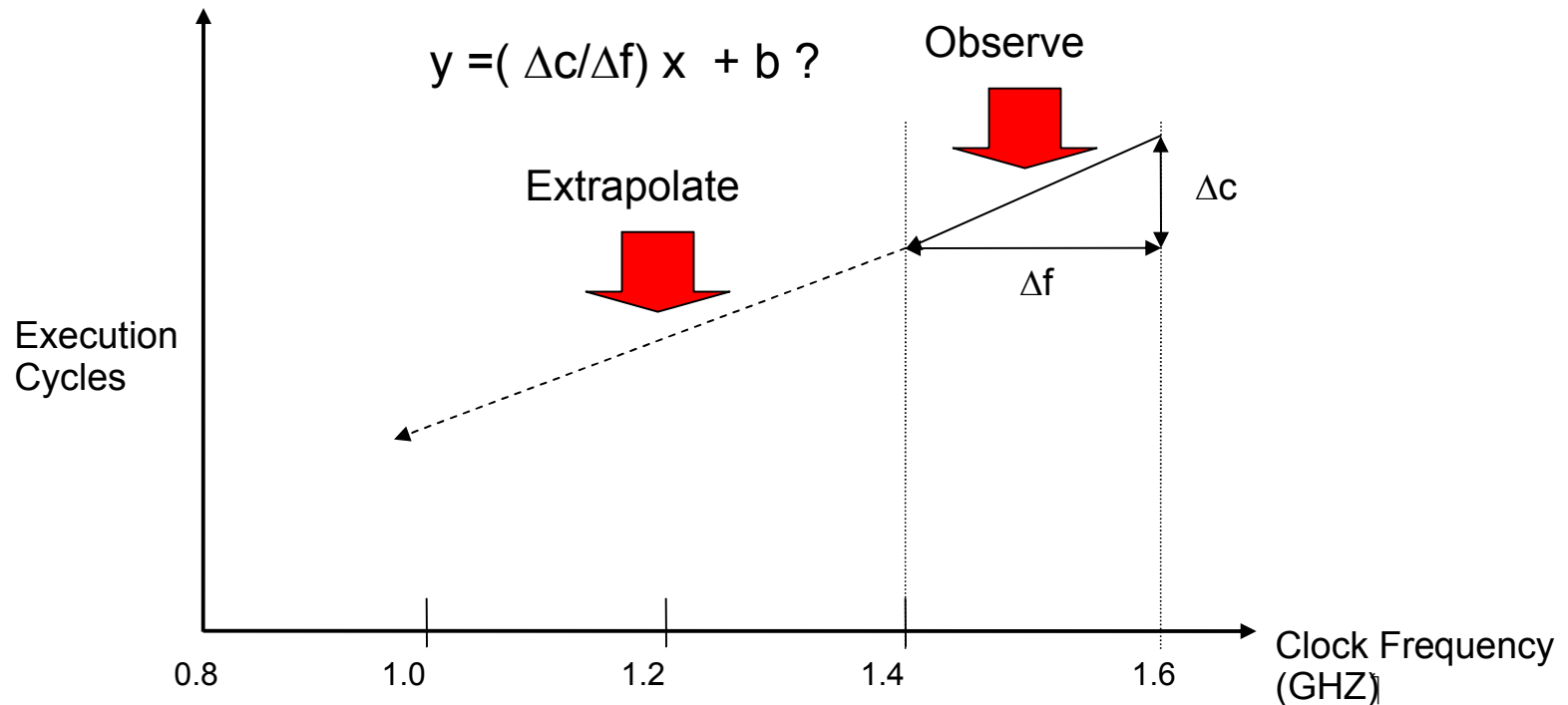
- adjust the CPU speed by a **tiny** amount and **observe** the impact on execution cycles
- **extrapolate** workload behavior for all other speeds
- the **two point** hypothesis

# Why this Approach is Useful

- when CPU speed adjusted, observable impact takes integrates all the "hard to measure" **system effects**
- **minimal runtime overhead**
  - adjusting by a single notch
  - custom hardware makes this seamless

# The Two Point Hypothesis

- measuring execution cycles at the **two top** clock frequencies enough for predicting the execution time at **any other** clock frequency?



# Motivation for Two Point Hypothesis

- clock cycles as function of clock frequency  
~ linear or ~ quadratic
  - degree of CPU and memory overlap
- if linear, can be extrapolated with two points
- if quadratic, its slope is linear

# Thesis Statements

- adjusting CPU speed by a **single notch** suffices for calculating the performance at any speed.
- a notch can be **significantly smaller** than it typically is
- can **save substantial energy** using this approach

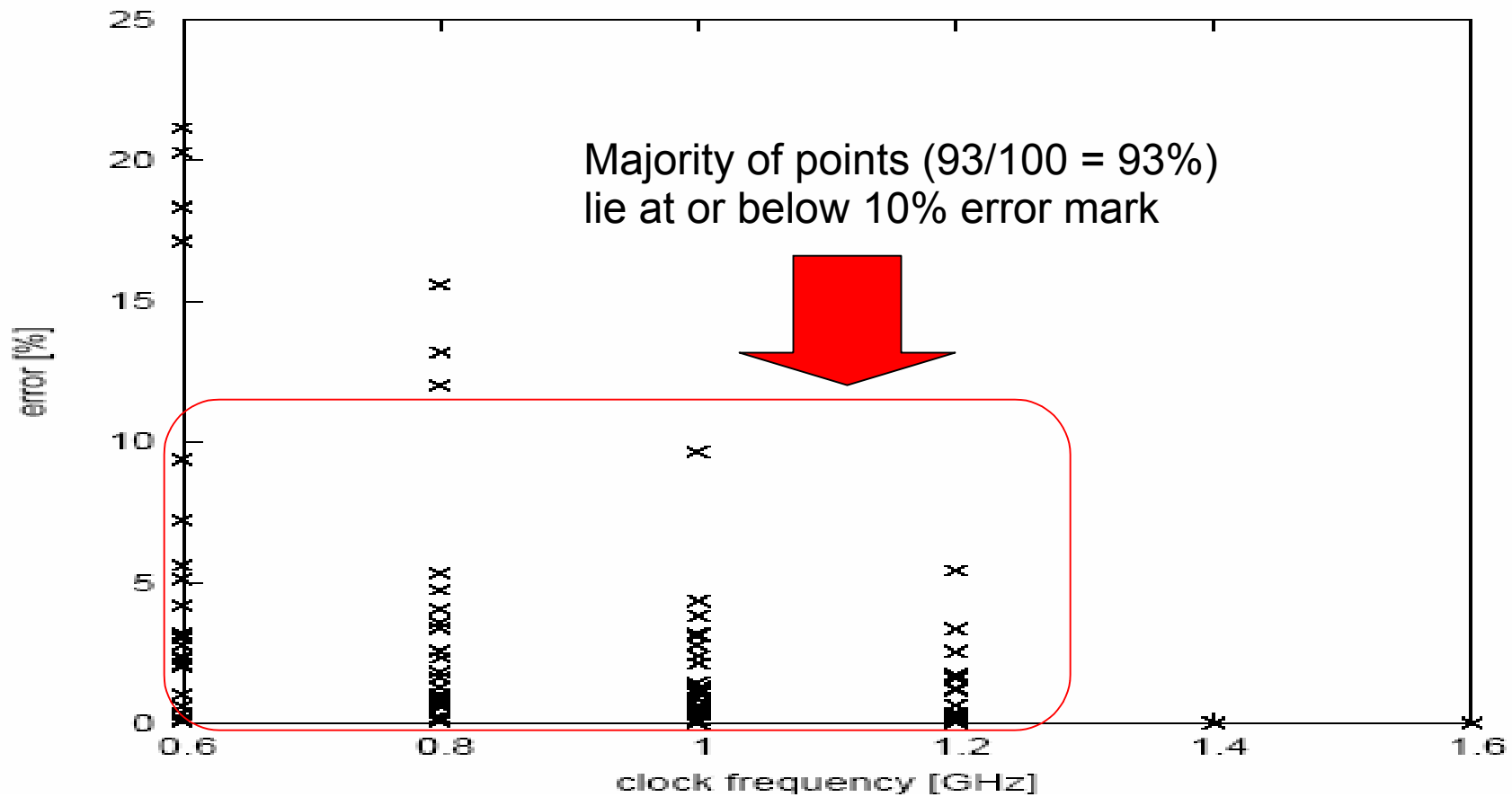
# Testing the Two Point Hypothesis

- tested “two point hypothesis” on 25 applications from SpecJVM, Javagrande and DaCapo Suites
- run each benchmark over every supported clock frequency
- very time consuming process

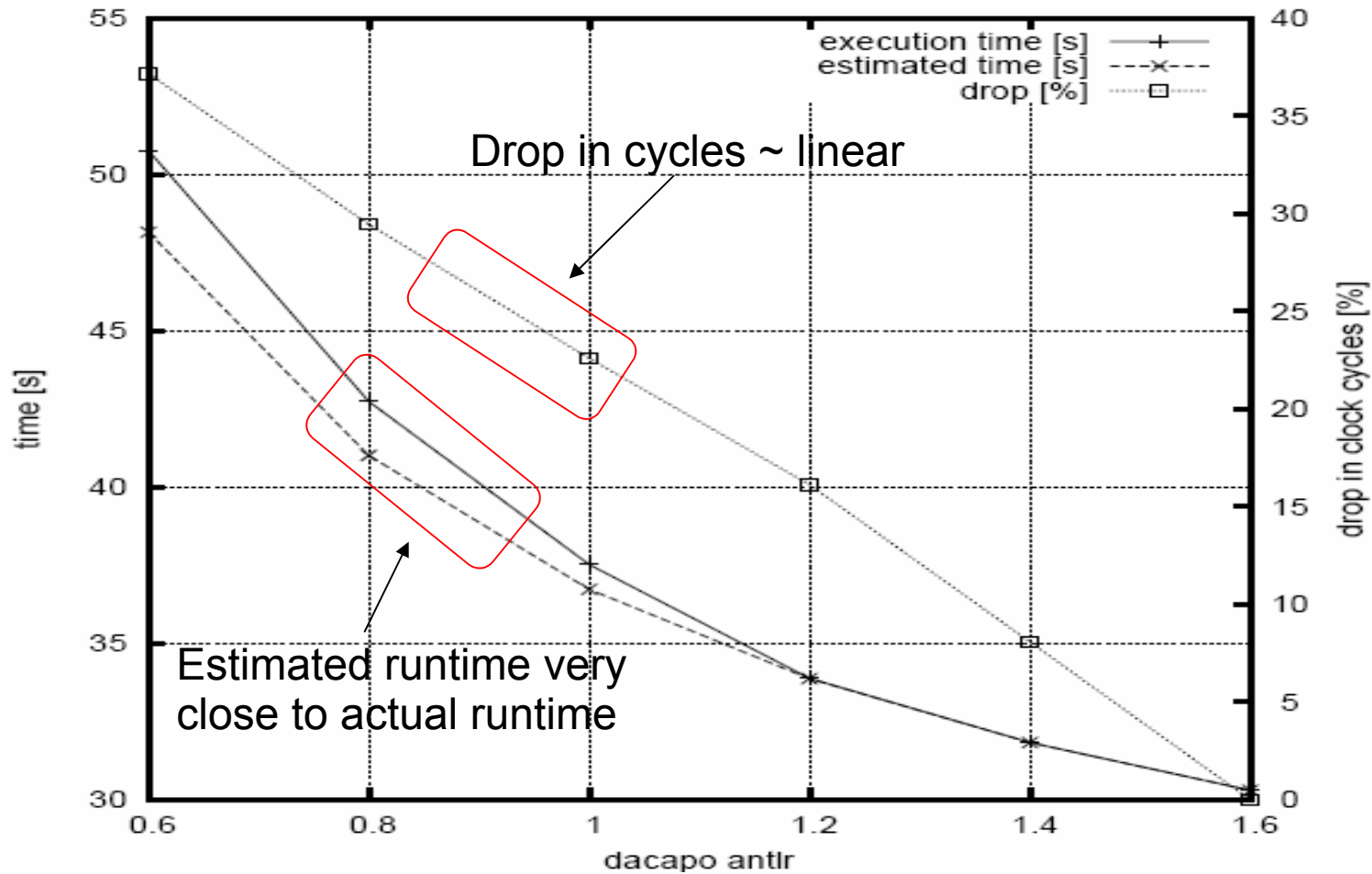
# Testing the Two Point Hypothesis

- record for each benchmark and every clock frequency
  - **actual execution time** for that clock frequency
  - **estimated execution time** under “two-point hypothesis”
  - **estimation error** as a percentage
- similar results for single user mode and multi user mode

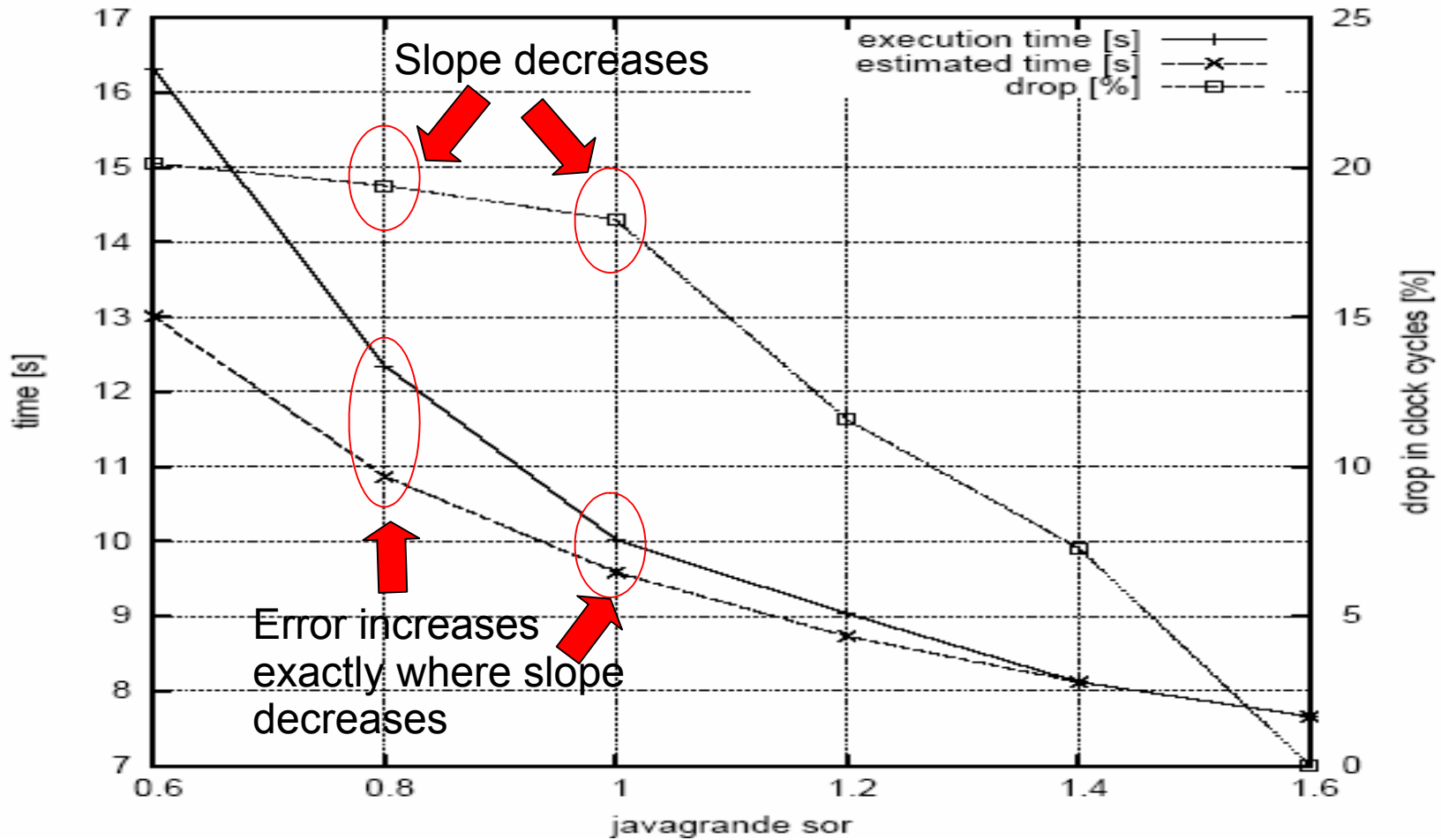
# Overall Estimation Error



# Why Most Results Are Good



# Why A Few Results Are Worse

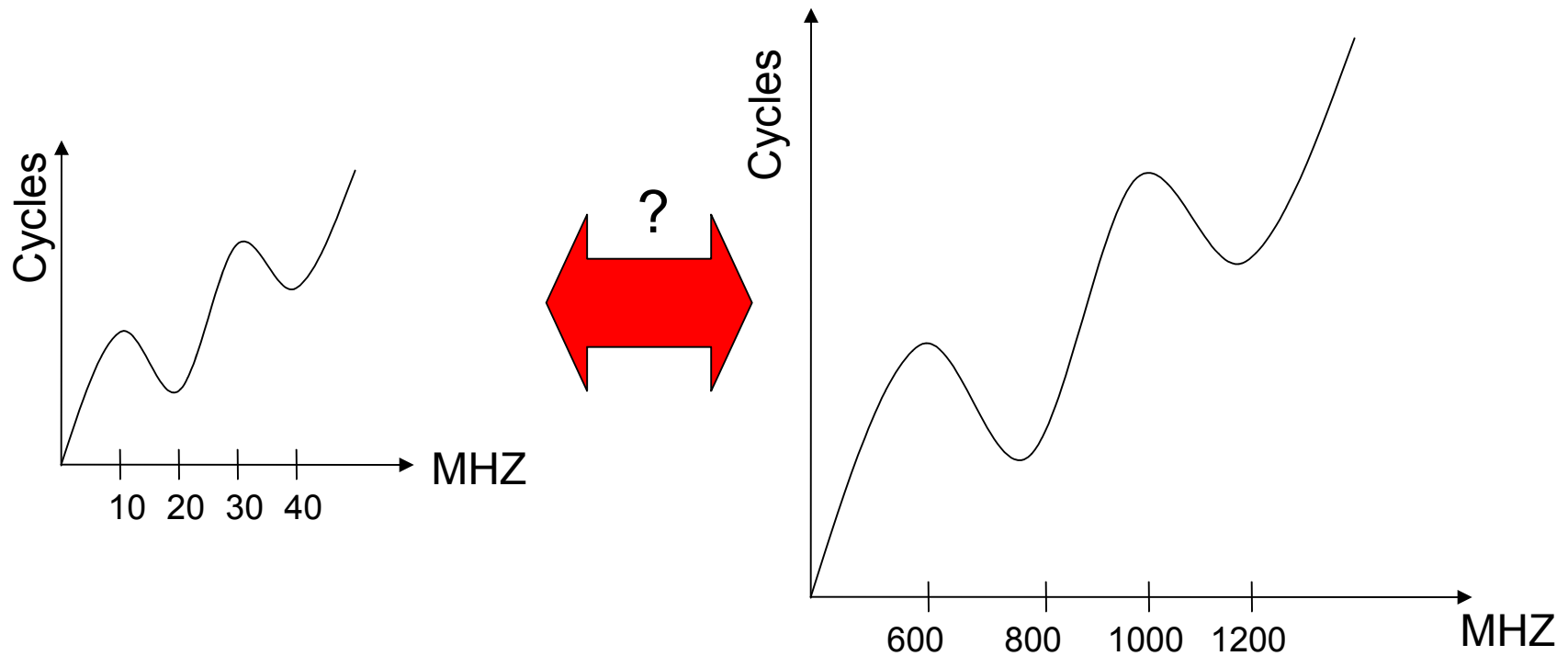


# Extending the Two-Point Approach

- distance between consecutive frequency steps **hundreds** of megahertz on typical laptops
- speed adjustments still noticeable
- would the two-point approach work if the distance were **tens** of megahertz?

# Larger Research Issue

- what can behavior at **microscopic** level tell us about behavior at **macroscopic** level?



# Methodology

- modified SimpleScalar to simulate a DVS-enabled out-of-order processor
- run benchmarks from Spec 2000 suite in simulator and on a typical laptop
- **argument by transitivity**

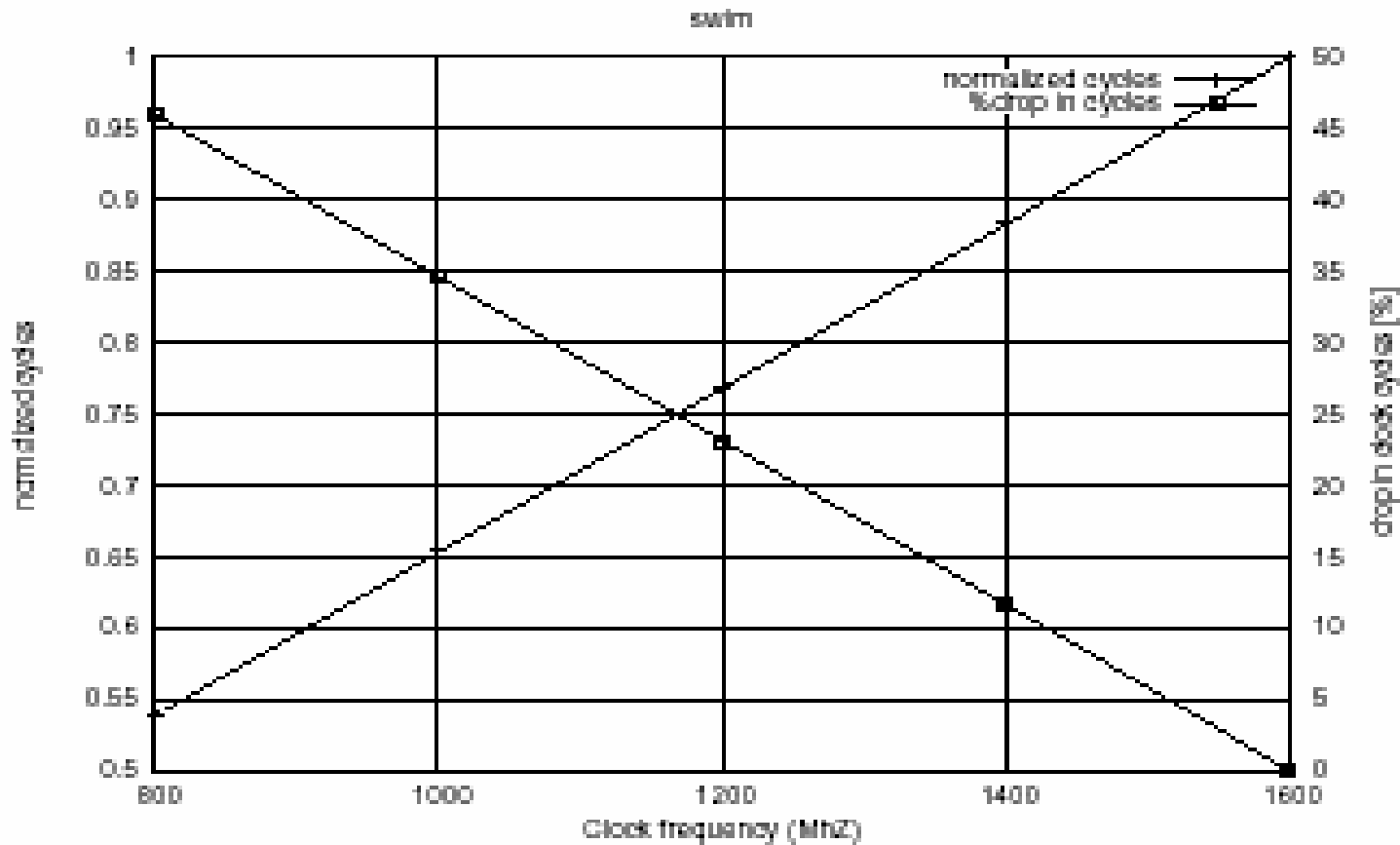
# Methodology

- isomorphism between workload behavior in **simulations** and in **real hardware**
- isomorphism between workload behavior in **simulations** at different granularities
- **by transitivity**, isomorphism between workload behavior in **real hardware** at different granularities

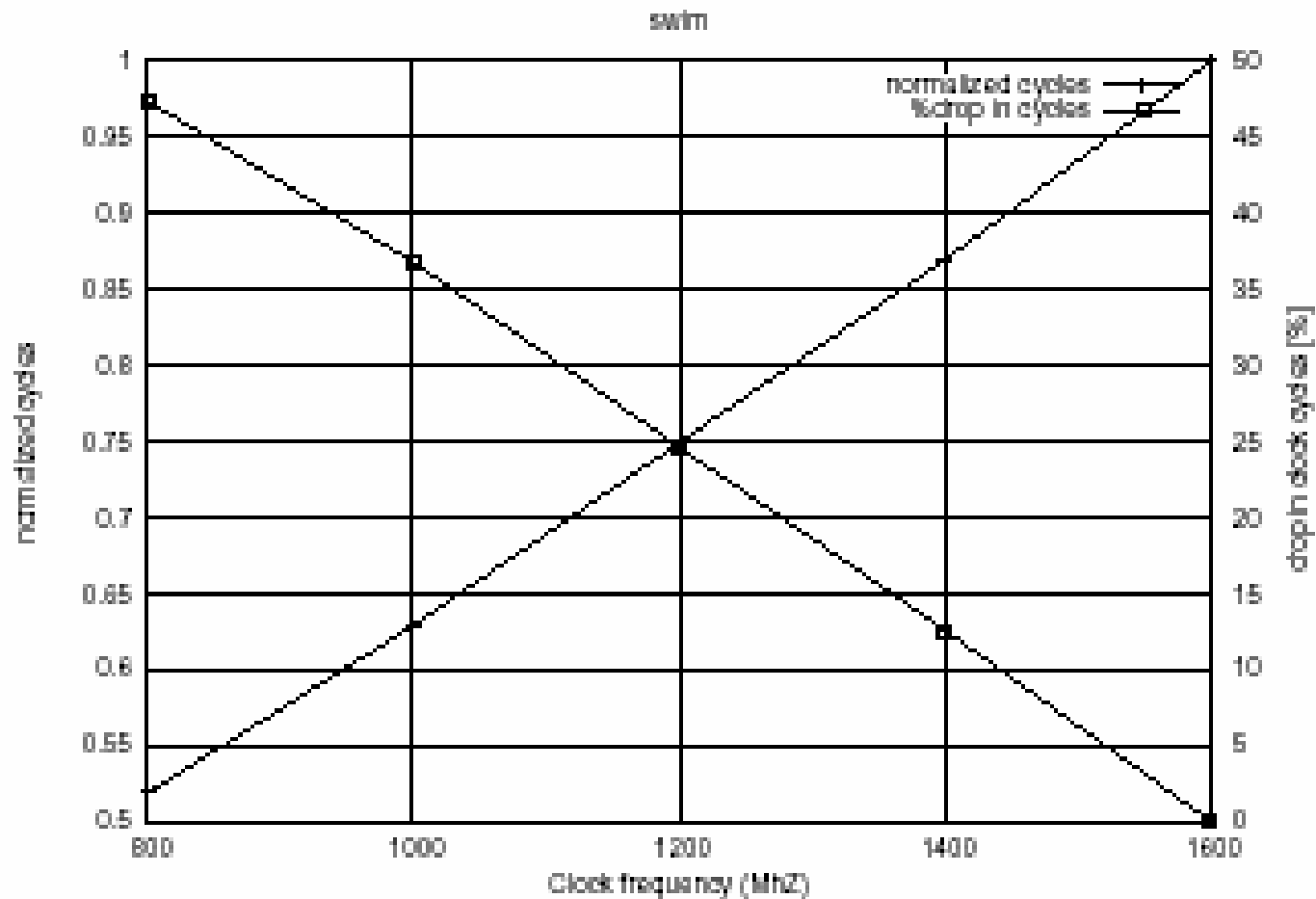
# Isomorphism One

- run each benchmark on a typical laptop at all the supported clock frequencies
- run each benchmark in simulator at the same clock frequencies
- record execution cycles at each clock frequency
- compare the plots

# Workload Behavior in Simulator (swim)



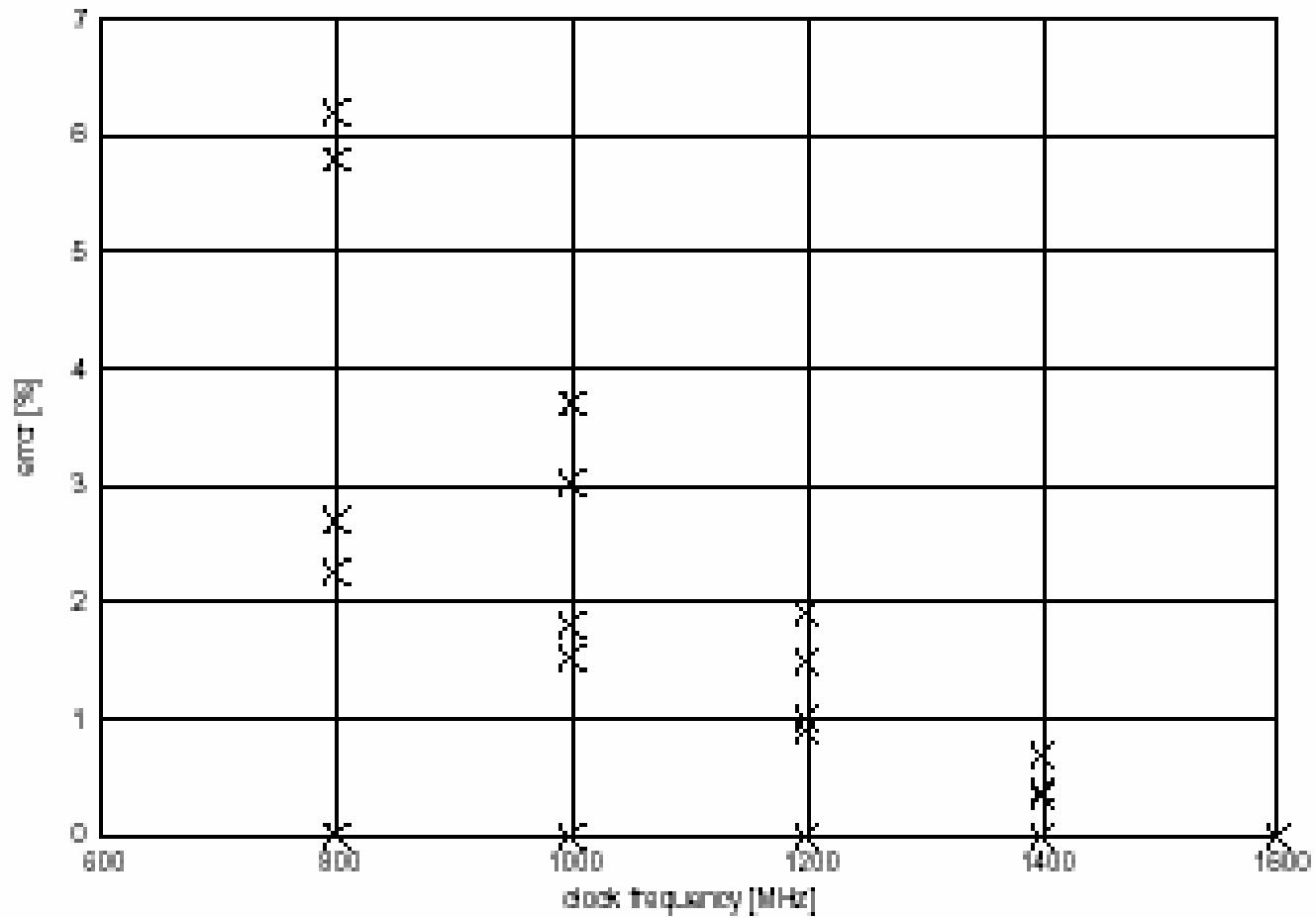
# Workload Behavior on Real Hardware (swim)



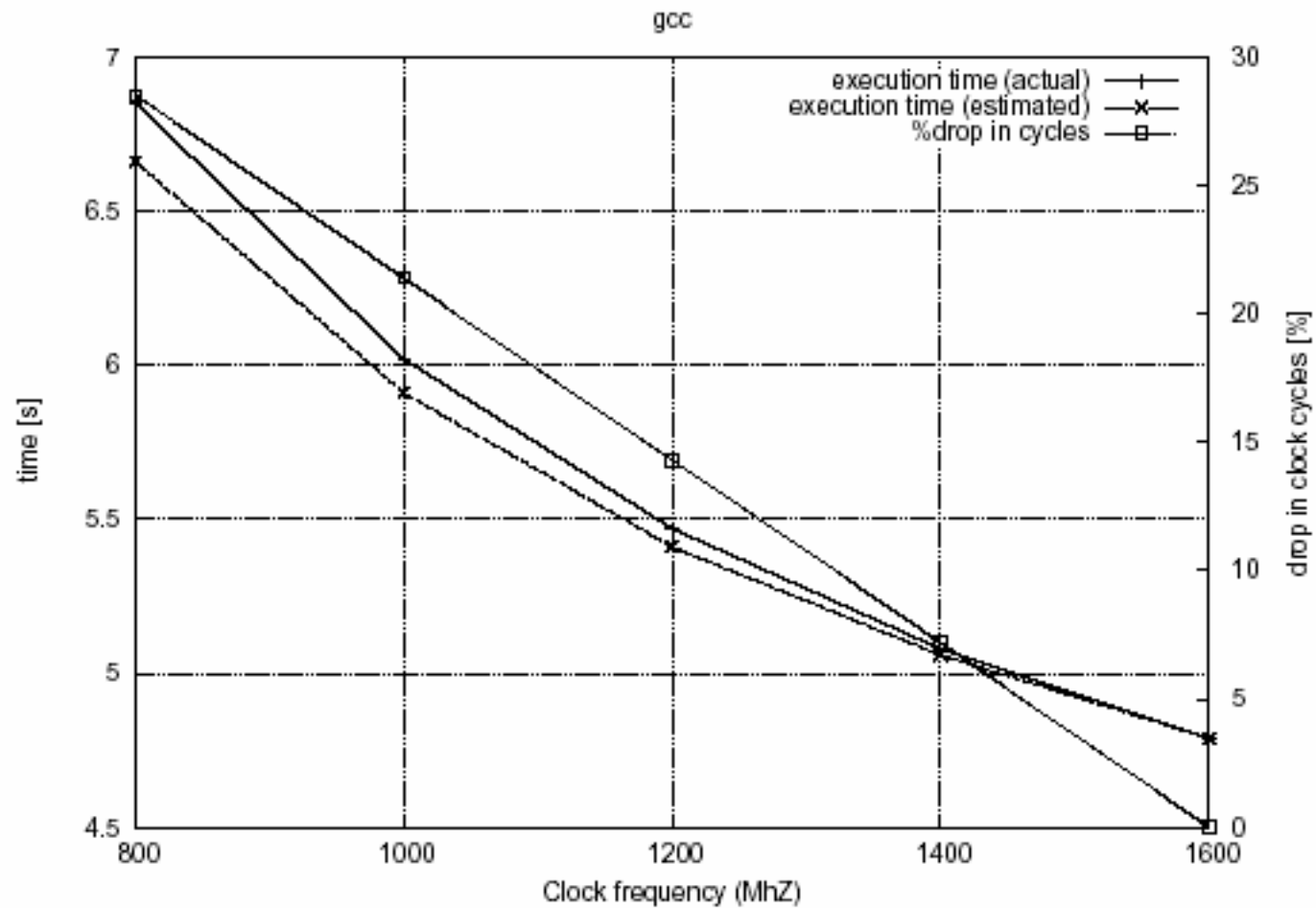
# Isomorphism Two

- run each benchmark in simulator 10 MHz below the top clock frequency
- extrapolate execution times for other clock frequencies
- compare with actual simulated execution times

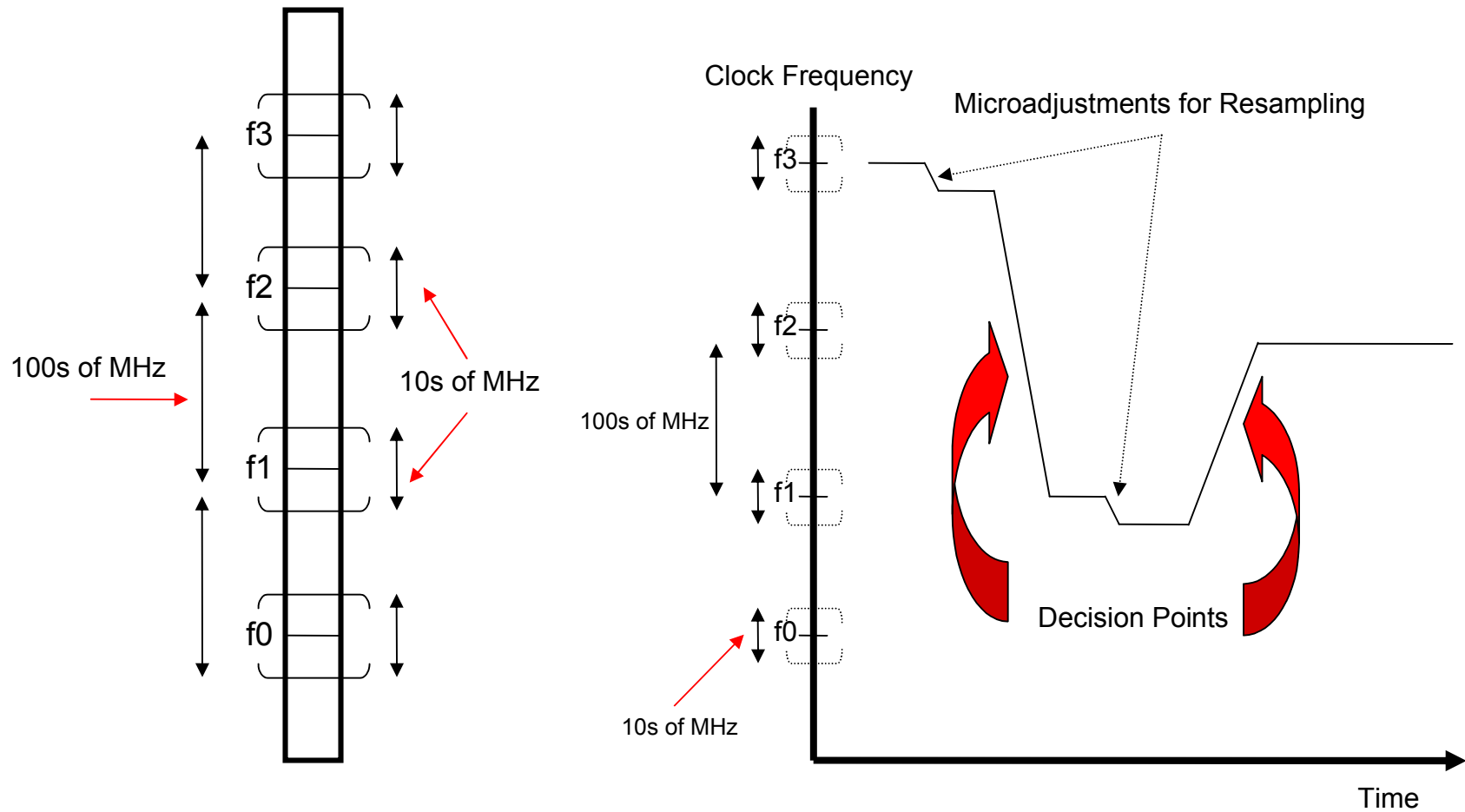
# Overall Estimation Errors



# Estimated vs. Actual Runtimes (GCC)



# Self-Calibrating Feedback Loop



# Prototype Implementation

- prototype implementations in Sun Hotspot Virtual Machine and Intel Pin toolkit
  - method and loop granularities
- 14 benchmarks from Javagrande, Spec 2000 and Spec 95 suites
- energy measurements on a typical laptop

# Results

	Benchmark	Energy Saved %	Runtime Overhead %
Memory Bound	jgf_sparse	26	1.5
	171.swim	23.9	11.9
	jgf_FFT	22	2.6
	102.swim	19.19	11.9
Borderline	104.hydro2d	16.58	17.89
	jgf_LUFact	16.49	18.37
	101.tomcatv	13.05	19.72
CPU Bound	125.turbo3d	0.63	0.11
	jgf_crypt	-1.06	5
	jgf_series	-1.3	5.2

# Limitations

- difficult to track input or phase changes without relying on other hardware events
  - use hints from hardware counters?
  - continuous adaptation may amortize errors
- limitations of existing hardware

# Future Work

- multicore processors
- extend two-point approach to a multipoint approach
- OS-level implementation

# Want More Information?

- V. Venkatachalam, C. Probst and M. Franz, A New Way of Estimating Compute Boundedness and Its Application To Dynamic Voltage Scaling, *International Journal Of Embedded Systems*, Vol. 1, Nos. 1/2/3. p.64-74
- V. Venkatachalam, C. Probst and M. Franz, Self-Calibrating Processors: A New Feedback Loop For Dynamic Voltage Scaling Control, Technical Report No. 06-15, School of Information and Computer Science, University of California, Irvine, December 2006
- <http://www.ics.uci.edu/~vvenkata>

# Want more Information?

- Pollack, New Microarchitecture Challenges in the Coming Generations of CMOS Process Technologies, *Proc. Of the 32<sup>nd</sup> Annual Symposium on ACM/IEEE International Symposium on Microarchitecture*, 1999