UNIVERSITY OF
CALGARY

Dynamic Speed Scaling: Theory and Practice

Carey Williamson

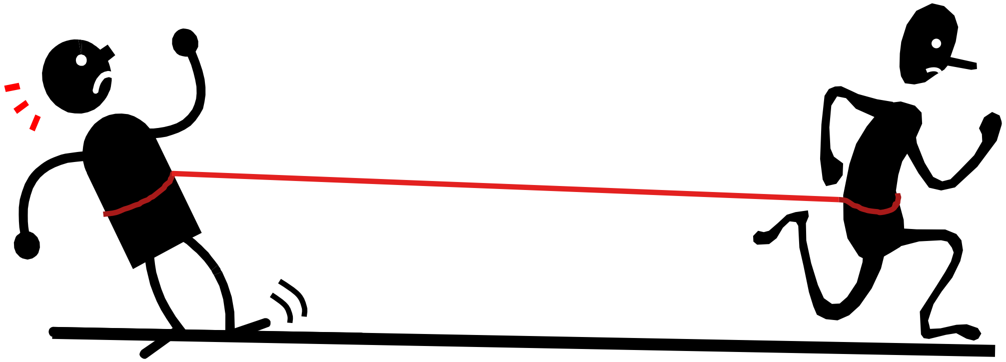
Department of Computer Science

Introduction and Motivation

- The ICT ecosystem is responsible for 10% of the world's energy consumption [Mills 2013]
- Data centers account for roughly 2% of global energy consumption (and still growing at a rate of approximately 6% per annum)
- The most energy-intensive component of any computer is its processor [Skrenes 2016]
  - 90% of energy usage when active (72W/80W)
  - 48% of energy usage when idle (3.1W/6.4W)
- Need for more energy-efficient computing

Speed Scaling: Inherent Tradeoffs

Dynamic Speed Scaling: adapt service rate to the current state of the system to balance energy consumption and performance.



Run slower: less ***energy***

Run faster: less ***delay***

- Minimize power consumption ***P***
  - Minimize energy cost $\boldsymbol{\varepsilon}$
  - Minimize heat, wear, etc.
- Minimize response time ***T***
  - Minimize delay
- Maximize job throughput

Main Messages (preview)

- There is broad and diverse literature on speed scaling systems for the past 20+ years
- There is a dichotomy between theoretical work and systems work on speed scaling
- Modern processors provide surprisingly rich functionality for speed scaling that is not yet well exploited by systems software
- There are many interesting tradeoffs to explore in dynamic speed scaling systems

Talk Outline

- Introduction and Motivation
- Background and Literature Review
- Summary of Key Results and Insights
- Recent Results and Contributions
  - Practice: Experimental Measurements
  - Theory: Autoscaling Effects
- Conclusions and Future Directions

Background: Theory and Systems

Theoretical Research

- Goal: optimality
- Domains: CPU, parallel systems
- Methods: proofs, complexity, competitive analysis, queueing theory, Markov chains, worst case, asymptotics, simulation
- Metrics: E[T], E[ε], combo, slowdown, competitive ratio
- Power: $P = s^{\alpha}$ $(1 \leq \alpha \leq 3)$
- Schedulers: PS, SRPT, FSP, YDS
- Speed scalers: job-count-based, continuous and unbounded speeds
- Venues: SIGMETRICS, PEVA, Performance, INFOCOM, OR

Systems Research

- Goal: practicality
- Domains: CPU, disk, network
- Methods: DVFS, power meter, measurement, benchmarking, simulation, power gating, over-clocking, simulation
- Metrics: response time, energy, heat, utilization
- Power: $P = a\, C_{eff}\, V^2\, f$
- Schedulers: FCFS, RR, FB
- Speed scalers: threshold-based, discrete and finite speeds
- Venues: SIGMETRICS, SOSP, OSDI, ISCA, MASCOTS, TOCS

Literature #1: The Classics

- [Kelly 1979] Reversibility and Stochastic Networks, Wiley
- [Kleinrock 1975] Queueing Systems, Volume 1: Theory, Wiley
- [Schrage 1968] “A Proof of the Optimality of the SRPT Discipline”, Operations Research
- [Weiser et al. 1994] “Scheduling for Reduced CPU Energy”, OSDI (and Mobile Computing)
- ★ [Yao, Demers, Shenker 1995] “A Scheduling Model for Reduced CPU Energy”, FOCS

Literature #2: Scheduling

- [Bansal and Harchol-Balter 2001] "Analysis of SRPT Scheduling: Investigating Unfairness", SIGMETRICS
- ★ [Friedman and Henderson 2003] "Fairness and Efficiency in Web Server Protocols", SIGMETRICS
- [Harchol-Balter et al. 2002] "Asymptotic Convergence of Scheduling Policies with Respect to Slowdown", IFIP Performance
- [Rai et al. 2003] "Analysis of LAS Scheduling for Job Size Distributions with High Variance", SIGMETRICS
- [Wierman and Harchol-Balter 2003] "Classifying Scheduling Policies with Respect to Unfairness in an M/GI/1", SIGMETRICS

Literature #3: Speed Scaling

- [Albers 2010] “Energy-Efficient Algorithms”, CACM
- [Albers et al. 2014] “Speed Scaling with Parallel Processors”, Algorithmica
- [Bansal et al. 2007] “Speed Scaling to Manage Energy and Temperature”, JACM
- [Bansal et al. 2009a] “Speed Scaling with an Arbitrary Power Function”, SIAM
- [Bansal et al. 2009b] “Speed Scaling for Weighted Flow Time”, SIAM
- ★ [Andrew, Lin, Wierman 2010] “Optimality, Fairness, and Robustness in Speed Scaling Designs”, SIGMETRICS
- [Elahi et al. 2012] “Decoupled Speed Scaling: Analysis and Evaluation”, QEST (PEVA 2014)
- [Elahi et al. 2014] “Turbo-charged Speed Scaling: Analysis and Evaluation”, MASCOTS
- [Wierman et al. 2009] “Power-Aware Speed Scaling in Processor Sharing Systems”, IEEE INFOCOM

Literature #4: Inexact Job Sizes

- [Dell’Amico et al. 2014] “Revisiting Size-based Scheduling with Estimated Job Sizes”, MASCOTS
- ★ [Dell’Amico et al. 2016] “PSBS: Practical Size-Based Scheduling”, IEEE Trans. on Computers
- [Lu et al. 2004] “Size-based Scheduling Policies with Inaccurate Scheduling Information”, MASCOTS
- [Rai et al. 2003] “Analysis of LAS Scheduling for Job Size Distributions with High Variance”, SIGMETRICS
- [Wierman et al. 2008] “Scheduling Despite Inexact Job Size Information”, SIGMETRICS

Literature #5: Systems

- [Hahnel et al. 2012] “Measuring Energy Consumption for Short Code Paths Using RAPL”, PER
- ★ [Meisner et al. 2009] “PowerNap: Eliminating Server Idle Power”, ASPLOS
- [Schroeder et al. 2006] “Web Servers Under Overload: How Scheduling Can Help”, TOIT
- [Skrenes and Williamson 2016] “Experimental Calibration and Validation of a Speed Scaling Simulator”, MASCOTS
- [Snowdon et al. 2009] “Koala: A Platform for OS-level Power Management”, EuroSys
- [Snowdon et al. 2007] “Accurate Online Prediction of Processor and Memory Energy Usage under Voltage Scaling”, Embedded Software
- [Spiliopoulos 2012] “Power-Sleuth: A Tool for Investigating Your Program’s Power Behaviour”, MASCOTS

Talk Outline

- Introduction and Motivation
- Background and Literature Review
- Summary of Key Results and Insights
- Recent Results and Contributions
    - Practice: Experimental Measurements
    - Theory: Autoscaling Effects
- Conclusions and Future Directions

Key Results: Single-Speed World

- PS is the gold standard for fairness
- Asymptotic convergence of slowdown for all work-conserving scheduling policies
- SRPT is “Sometimes Unfair”
- YDS is optimal for energy consumption
- FSP dominates PS for response time

Key Results: Speed Scaling World

- No policy can be optimal, robust, and fair
- Speed scaling exacerbates unfairness
- Asymptotic convergence of slowdown property no longer holds
- FSP’s dominance of PS breaks under coupled speed scaling
- FSP’s dominance of PS is restored under decoupled speed scaling

Talk Outline

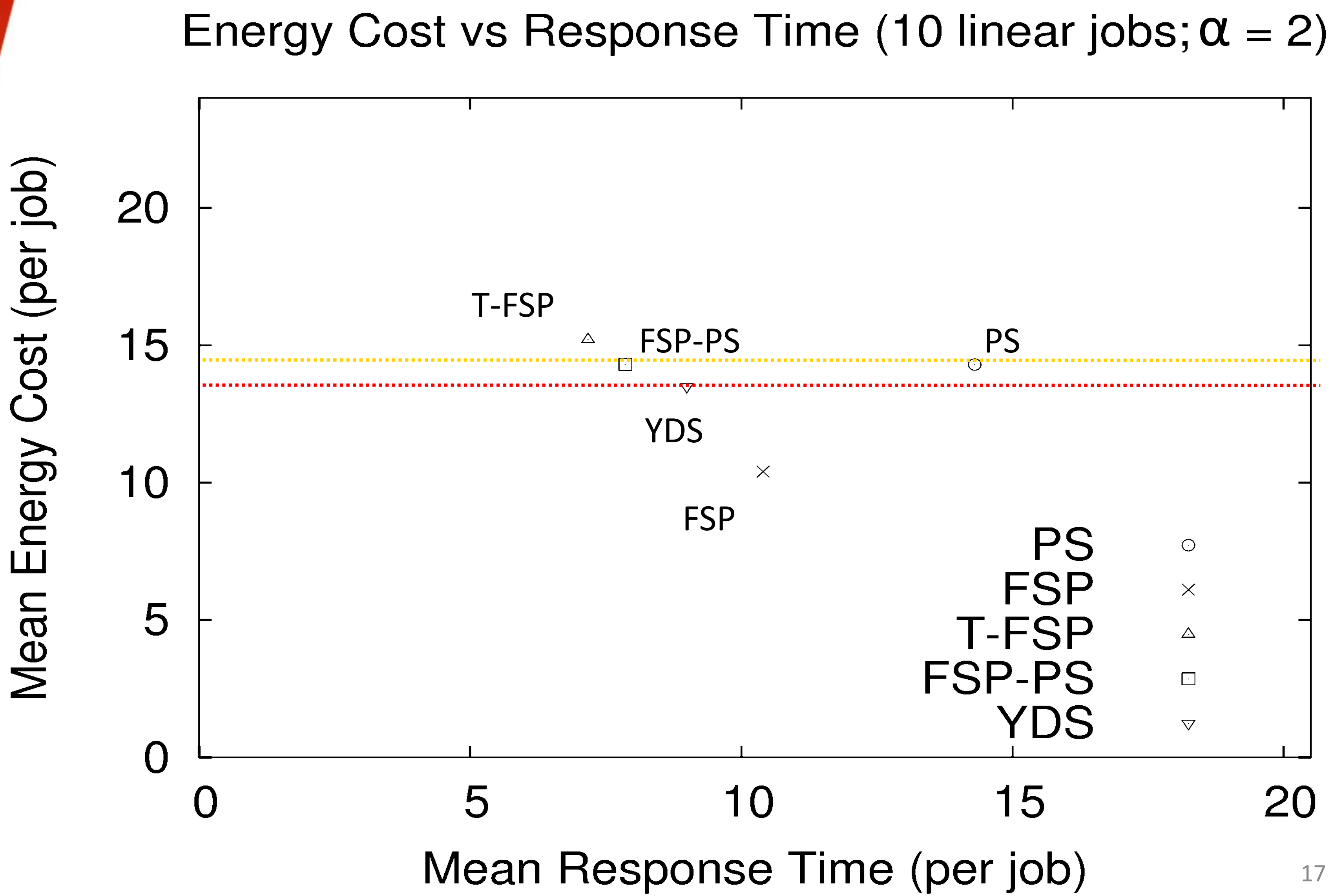
- Introduction and Motivation
- Background and Literature Review
- Summary of Key Results and Insights
- Recent Results and Contributions
  - Practice: Experimental Measurements
  - Theory: Autoscaling Effects
- Conclusions and Future Directions

IEEE MASCOTS 2016

Experimental Calibration and Validation of a Speed Scaling Simulator

Arsham Skrenes
Carey Williamson
Department of Computer Science
UNIVERSITY OF CALGARY

Example Simulation Results: IEEE MASCOTS 2014



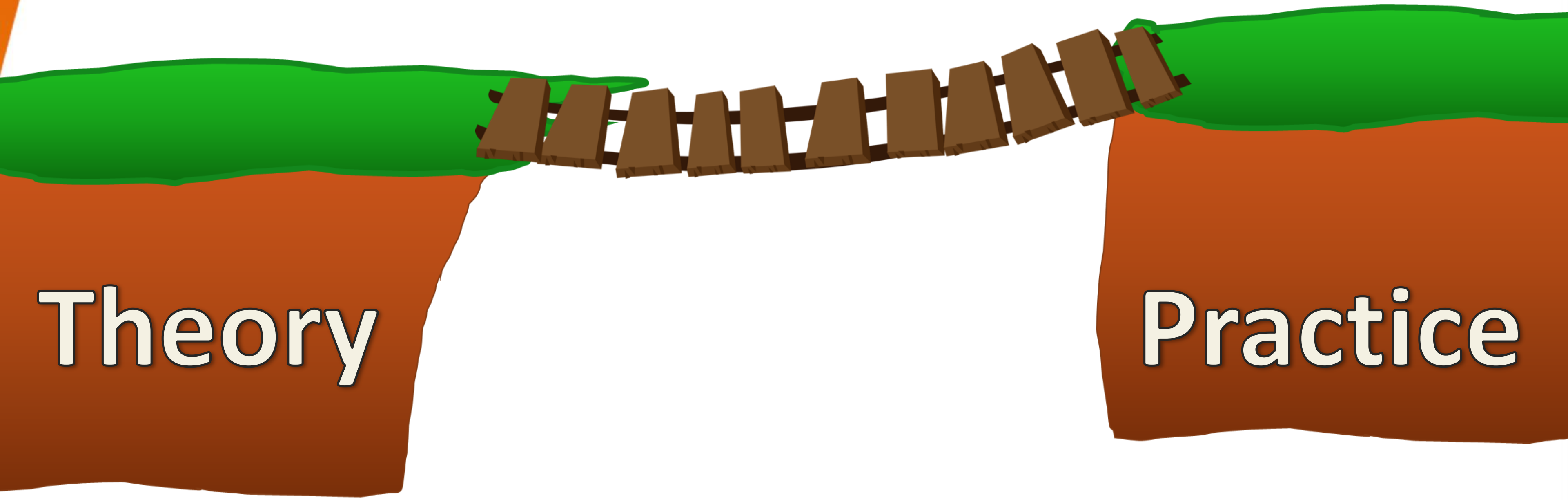
Typical Modeling Assumptions

- Single-server queue for CPU service
- Single batch of n jobs arrive at time 0
- Job sizes known in advance
- Dynamic speed scaling with $s = f(n)$
- Power consumption $P = s^{\alpha}$ where $1 \leq \alpha \leq 3$
- Maximum system speed is unbounded
- System speeds are continuous (not discrete)
- Context switches are free (i.e., zero cost)
- Speed changes are free (i.e., zero cost)

Question: How would they perform on real systems?

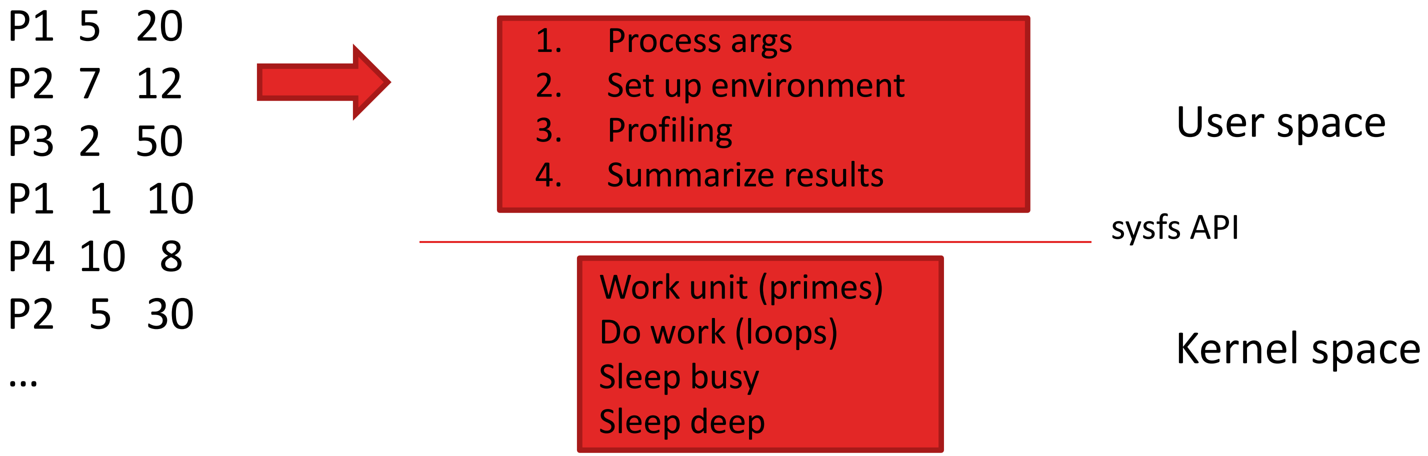
Bridging Theory and Practice

- Profilo enables all scheduling and speed scaling algorithms to be analyzed on real systems.



Profilo Design [Skrenes 2016]

- Flexible framework for the experimental evaluation of arbitrary scheduling and speed scaling policies
- Hybrid user-mode and kernel-mode implementation
- User space: CSV file input to specify workload
- Kernel space: carefully-controlled job execution, timing, and energy measurement using RAPL MSRs



P1 5 20
P2 7 12
P3 2 50
P1 1 10
P4 10 8
P2 5 30
…

1. Process args
2. Set up environment
3. Profiling
4. Summarize results

User space

sysfs API

Work unit (primes)
Do work (loops)
Sleep busy
Sleep deep

Kernel space

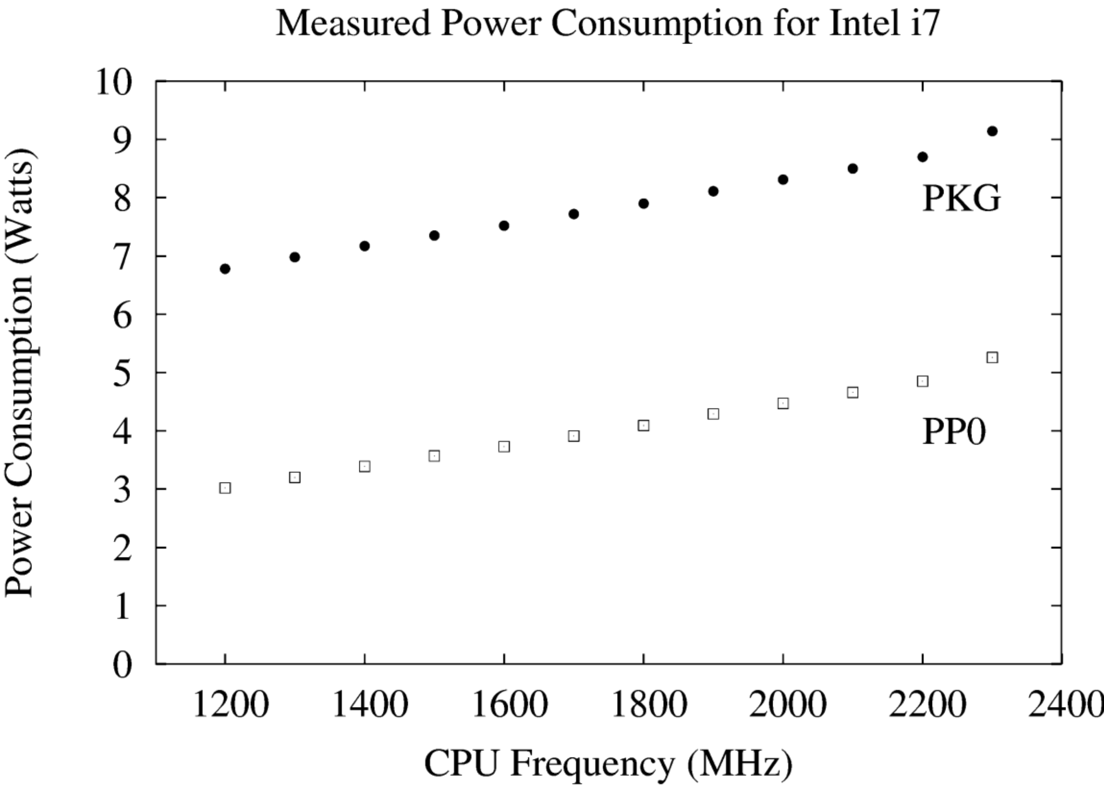
Running Average Power Limit (RAPL)

- Non-architectural model specific registers (MSRs)
- Accurate power meters for each of the domains (independently found to match actual power measurements)
- Four domains (three for any given CPU)
  - Power Plane 0 (PP0)
  - Power Plane 1 (PP1) – Consumer Packages Only
  - DRAM [8], [15] – Server Packages Only
  - Package (PKG)

Measurement Results

Frequency (MHz)	PP0 (W)	PKG (W)
2301 (3300)	11.5	15.3
2300	5.4	9.2
2200	5.0	8.9
2100	4.8	8.6
2000	4.6	8.4
1900	4.5	8.3
1800	4.3	8.0
1700	4.1	7.9
1600	3.9	7.6
1500	3.7	7.5
1400	3.5	7.3
1300	3.3	7.1
1200	3.1	6.9

Quite unpredictable and uncontrollable!



Highly linear throughout most of range!

Plus multiple sleep and idle modes (not shown here)

Measurement Results

Frequency (MHz)	PP0 (W)	PKG (W)	Context Switch (us)
2301 (3300)	11.5	15.3	1.140
2300	5.4	9.2	1.634
2200	5.0	8.9	1.708
2100	4.8	8.6	1.808
2000	4.6	8.4	1.898
1900	4.5	8.3	1.999
1800	4.3	8.0	2.118
1700	4.1	7.9	2.213
1600	3.9	7.6	2.369
1500	3.7	7.5	2.526
1400	3.5	7.3	2.709
1300	3.3	7.1	2.886
1200	3.1	6.9	3.167

Measurement Results

Frequency (MHz)	PP0 (W)	PKG (W)	Context Switch (us)	Speed Switch (us)
2301 (3300)	11.5	15.3	1.140	0.76
2300	5.4	9.2	1.634	1.09
2200	5.0	8.9	1.708	1.14
2100	4.8	8.6	1.808	1.20
2000	4.6	8.4	1.898	1.26
1900	4.5	8.3	1.999	1.32
1800	4.3	8.0	2.118	1.38
1700	4.1	7.9	2.213	1.47
1600	3.9	7.6	2.369	1.56
1500	3.7	7.5	2.526	1.67
1400	3.5	7.3	2.709	1.81
1300	3.3	7.1	2.886	1.93
1200	3.1	6.9	3.167	2.09

Measurement Results

Frequency (MHz)	PP0 (W)	PKG (W)	Context Switch (us)	Speed Switch (us)	Mode Switch (ns)
2301 (3300)	11.5	15.3	1.140	0.76	44.8
2300	5.4	9.2	1.634	1.09	64.2
2200	5.0	8.9	1.708	1.14	67.0
2100	4.8	8.6	1.808	1.20	70.2
2000	4.6	8.4	1.898	1.26	73.7
1900	4.5	8.3	1.999	1.32	78.3
1800	4.3	8.0	2.118	1.38	81.9
1700	4.1	7.9	2.213	1.47	86.7
1600	3.9	7.6	2.369	1.56	92.1
1500	3.7	7.5	2.526	1.67	98.6
1400	3.5	7.3	2.709	1.81	105.3
1300	3.3	7.1	2.886	1.93	113.4
1200	3.1	6.9	3.167	2.09	123.1

Experimental Evaluation Setup

- Three workloads (each with batch of 12 jobs)
  1. Homogenous
  2. Additive (arithmetic progression)
  3. Multiplicative (factors of 2)

- Three algorithms
  1. PS (epitomizes fairness)
  2. YDS (minimizes power consumption)
  3. FSP-PS (decoupled speed scaling; improves mean response time while retaining fairness)

Experimental Evaluation Results

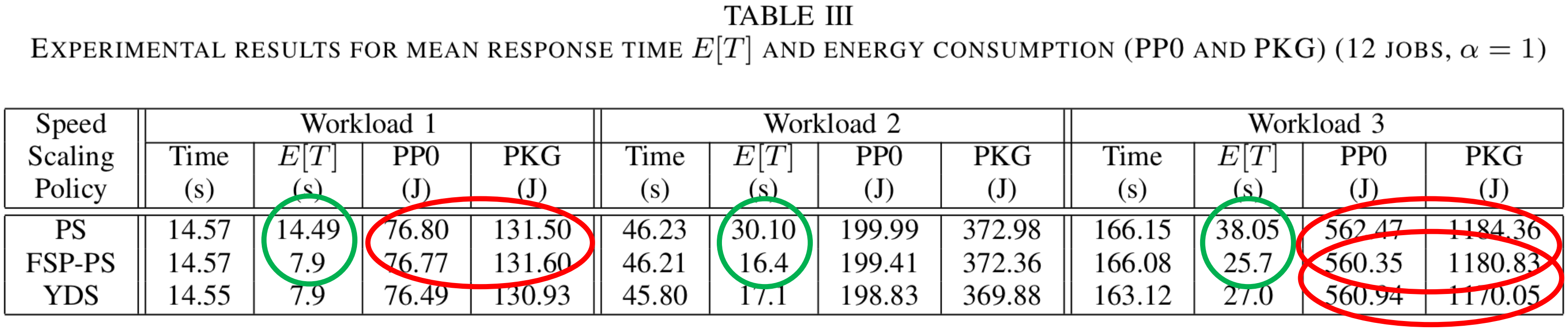


TABLE III

EXPERIMENTAL RESULTS FOR MEAN RESPONSE TIME $E[T]$ AND ENERGY CONSUMPTION (PP0 AND PKG) (12 JOBS, $\alpha = 1$)

Speed Scaling Policy	Workload 1				Workload 2				Workload 3			
	Time (s)	$E[T]$ (s)	PP0 (J)	PKG (J)	Time (s)	$E[T]$ (s)	PP0 (J)	PKG (J)	Time (s)	$E[T]$ (s)	PP0 (J)	PKG (J)
PS	14.57	14.49	76.80	131.50	46.23	30.10	199.99	372.98	166.15	38.05	562.47	1184.36
FSP-PS	14.57	7.9	76.77	131.60	46.21	16.4	199.41	372.36	166.08	25.7	560.35	1180.83
YDS	14.55	7.9	76.49	130.93	45.80	17.1	198.83	369.88	163.12	27.0	560.94	1170.05

- Observation 1: Decoupled speed scaling (FSP-PS) provides a significant response time advantage over PS, for the “same” energy costs
- Observation 2: The response time advantage of FSP-PS decreases as job size variability increases
- Observation 3: FSP-PS has a slight energy advantage over PS because of fewer context switches between jobs
- Observation 4: YDS has the lowest energy consumption among these policies (even better than expected due to discretization effect, and no speed changes)

Simulation Results

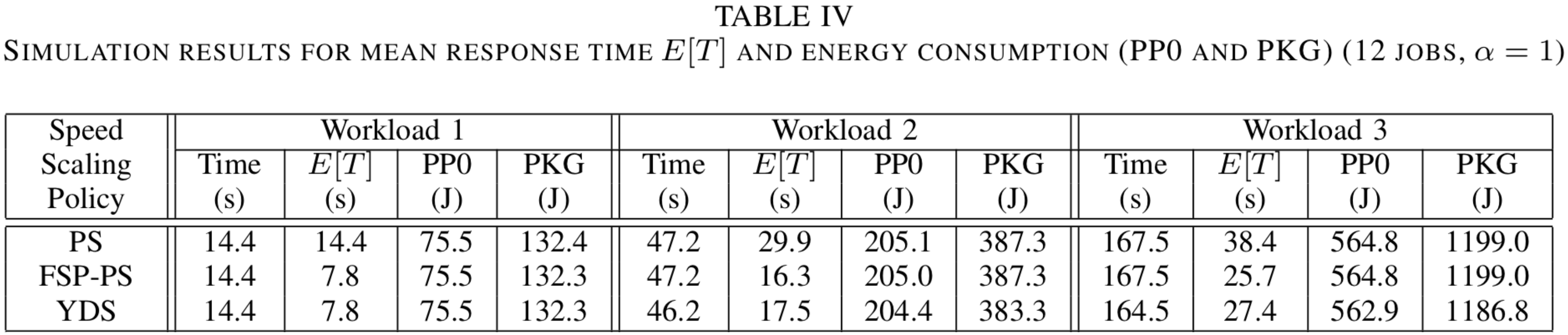


TABLE IV

SIMULATION RESULTS FOR MEAN RESPONSE TIME $E[T]$ AND ENERGY CONSUMPTION (PP0 AND PKG) (12 JOBS, $\alpha = 1$)

Speed Scaling Policy	Workload 1				Workload 2				Workload 3			
	Time (s)	$E[T]$ (s)	PP0 (J)	PKG (J)	Time (s)	$E[T]$ (s)	PP0 (J)	PKG (J)	Time (s)	$E[T]$ (s)	PP0 (J)	PKG (J)
PS	14.4	14.4	75.5	132.4	47.2	29.9	205.1	387.3	167.5	38.4	564.8	1199.0
FSP-PS	14.4	7.8	75.5	132.3	47.2	16.3	205.0	387.3	167.5	25.7	564.8	1199.0
YDS	14.4	7.8	75.5	132.3	46.2	17.5	204.4	383.3	164.5	27.4	562.9	1186.8

Summary: Practice

- Designed and implemented a novel experimental platform (Profilo) for fine-grain energy measurements
  - Hybrid user-space/kernel-space using RAPL and *hrtimers*
  - Flexible platform to quantify tradeoffs between different scheduling and speed scaling strategies
- Used this experimental platform to do the following:
  - Micro-benchmark a modern Intel processor to measure system costs and power consumption
  - Calibrate/validate a discrete-event simulator for dynamic speed scaling systems
  - Compare and evaluate three different speed scaling strategies from the literature: PS, FSP-PS, and YDS
- Gained new insights into practical aspects of dynamic speed scaling systems

Talk Outline

- Introduction and Motivation
- Background and Literature Review
- Summary of Key Results and Insights
- Recent Results and Contributions
  - Practice: Experimental Measurements
  - Theory: Autoscaling Effects
- Conclusions and Future Directions

IEEE MASCOTS 2016

Autoscaling Effects in Speed Scaling Systems

Maryam Elahi
Carey Williamson
Department of Computer Science

Introduction

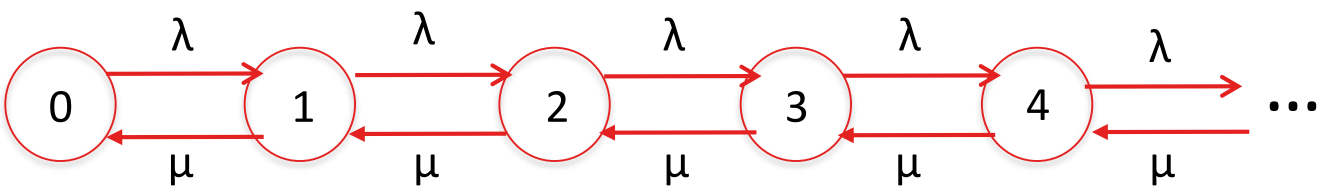
- Dynamic CPU speed scaling systems
- Service rate adjusted based on offered load
- Classic tradeoff:
  - Faster speed → lower response time, higher energy usage
- Two key design choices:
  - Scheduler: which job to run? (FCFS, <u>PS</u>, FSP, <u>SRPT</u>, LRPT)
  - Speed scaler: how fast to run? (static, <u>coupled</u>, decoupled)
- Research questions:
  - What are the “autoscaling” properties of coupled (i.e., job-count based) speed scaling systems under heavy load?
  - In what ways are PS and SRPT similar or different?

System Model (1 of 4)

Review: Birth-death Markov chain model of classic M/M/1 queue

Fixed arrival rate λ

Fixed service rate μ



Mean system occupancy: $N = \rho / (1 - \rho)$

Ergodicity requirement: $\rho = \lambda/\mu < 1$

$p_n = p_0 (\lambda/\mu)^n$

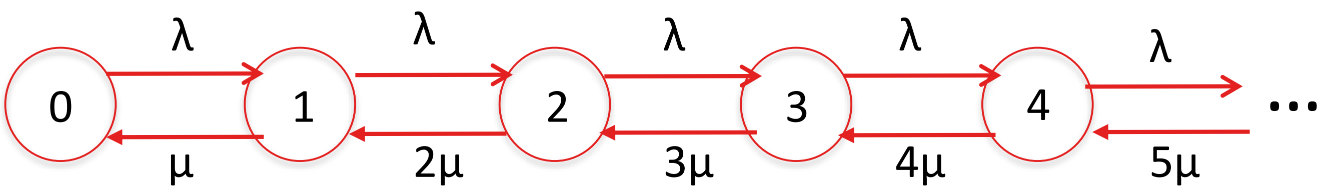
$U = 1 - p_0 = \rho$

System Model (2 of 4)

Birth-death Markov chain model of classic M/M/∞ queue

Fixed arrival rate λ

Service rate scales linearly with system occupancy (α = 1)



Mean system occupancy: $N = \rho = \lambda/\mu$

System occupancy has Poisson distribution

Ergodicity requirement: $\rho = \lambda/\mu < \infty$

$$p_n = p_0 \prod_{i=0}^{n-1} (\lambda/(i+1)\mu)$$

$$U = 1 - p_0 \neq \rho$$

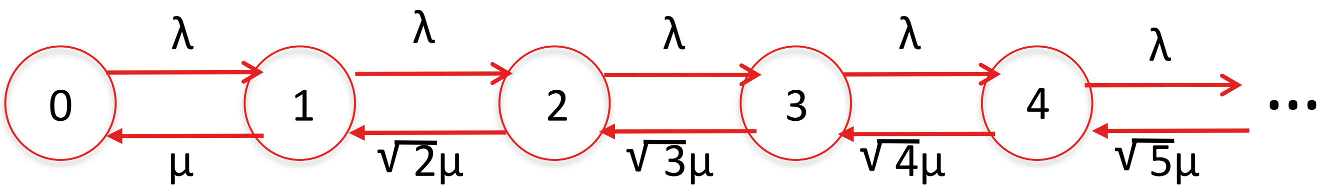
FCFS = PS ≠ SRPT

System Model (3 of 4)

Birth-death Markov chain model of dynamic speed scaling system

Fixed arrival rate λ

Service rate scales sub-linearly with system occupancy ($\alpha = 2$)



Mean system occupancy: $N = \rho^2 = (\lambda/\mu)^2$ $\qquad p_n = p_0 \prod_{i=0}^{n-1} (\lambda/(\sqrt{i+1})\mu)$

System occupancy has higher variance than Poisson distribution

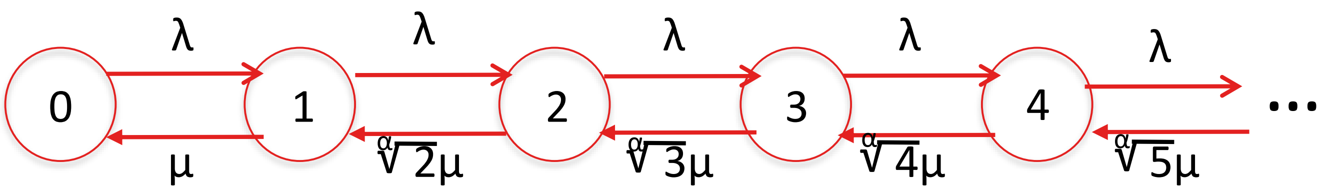
Ergodicity requirement: $\rho = \lambda/\mu < \infty$

System Model (4 of 4)

Birth-death Markov chain model of dynamic speed scaling system

Fixed arrival rate λ

Service rate scales sub-linearly with system occupancy ($\alpha > 1$)



Mean system occupancy: $N = \rho^\alpha = (\lambda/\mu)^\alpha$ $\quad p_n = p_0 \prod_{i=0}^{n-1} (\lambda/(\sqrt[\alpha]{i+1})\mu)$

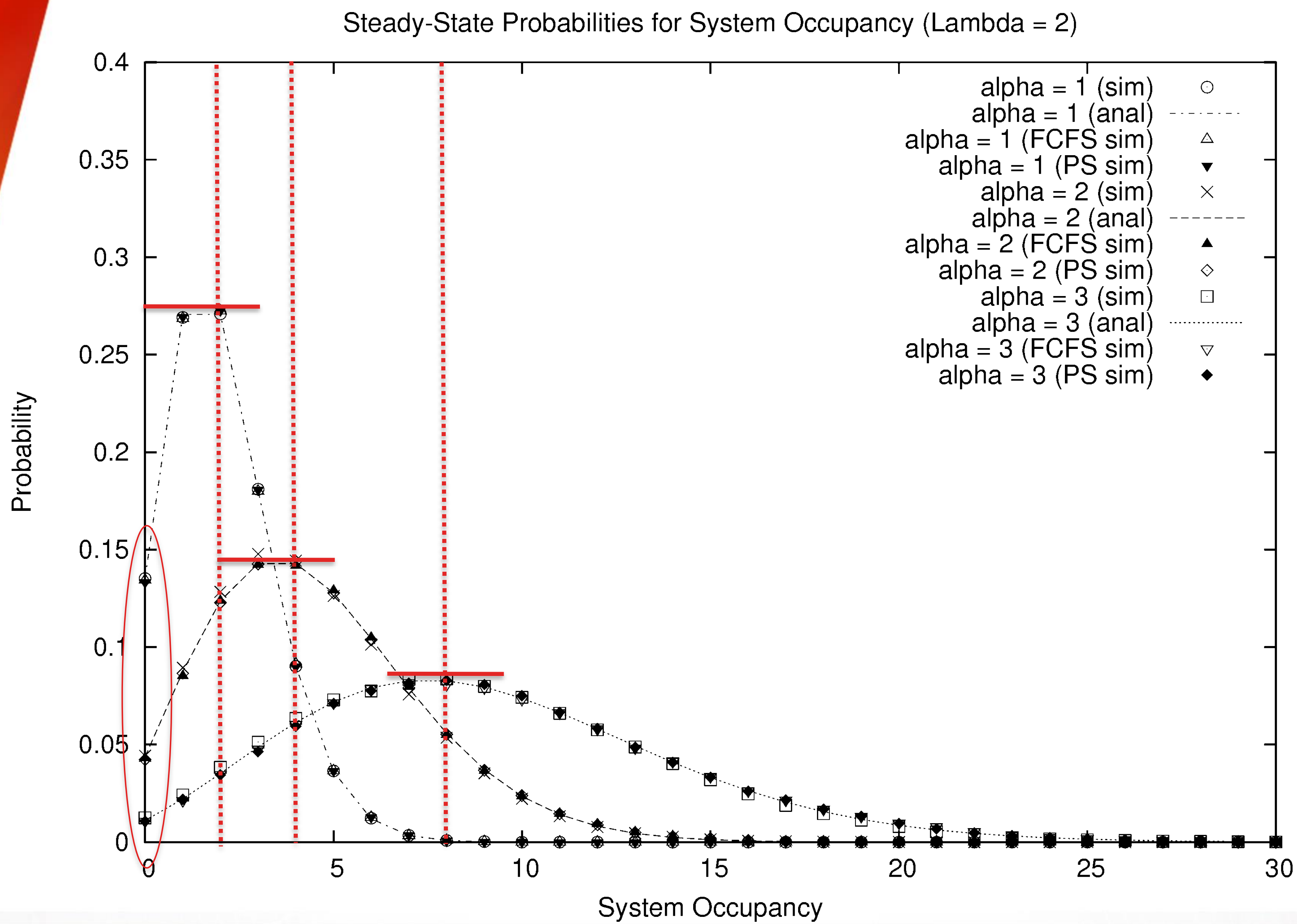
System occupancy has higher variance than Poisson distribution

Ergodicity requirement: $\rho = \lambda/\mu < \infty$

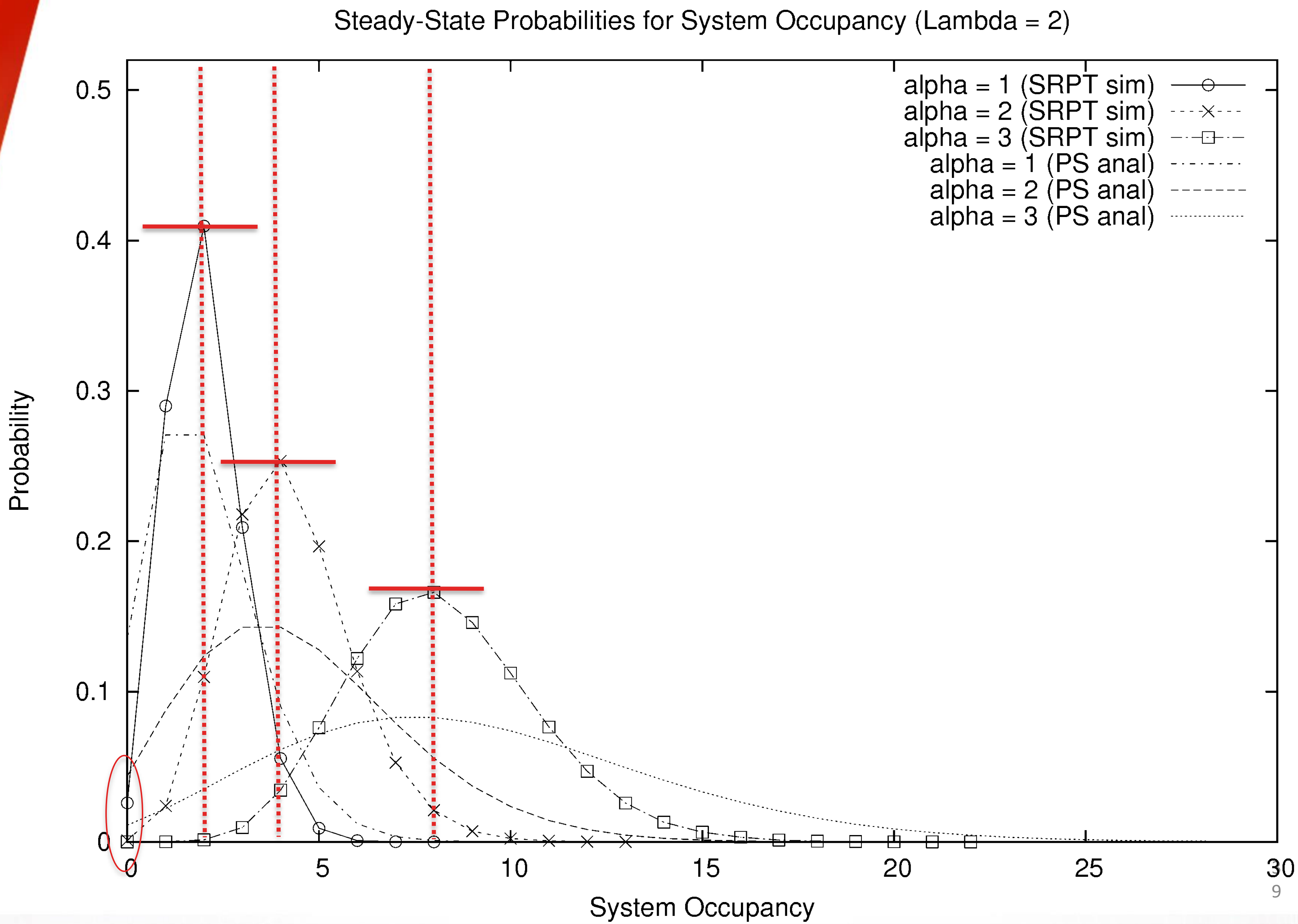
Analytical Insights and Observations

- In speed scaling systems, ρ and U differ
- Speed scaling systems stabilize even when $\rho > 1$
- In stable speed scaling systems, $s = \rho$ (an invariant)
- PS is amenable to analysis; SRPT is not
- PS with linear speed scaling behaves like M/M/∞, which has Poisson distribution for system occupancy
- Increasing α changes the Poisson structure of PS
- At high load, $N \rightarrow \rho^{\alpha}$ (another invariant property)

PS Modeling Results



SRPT Simulation Results

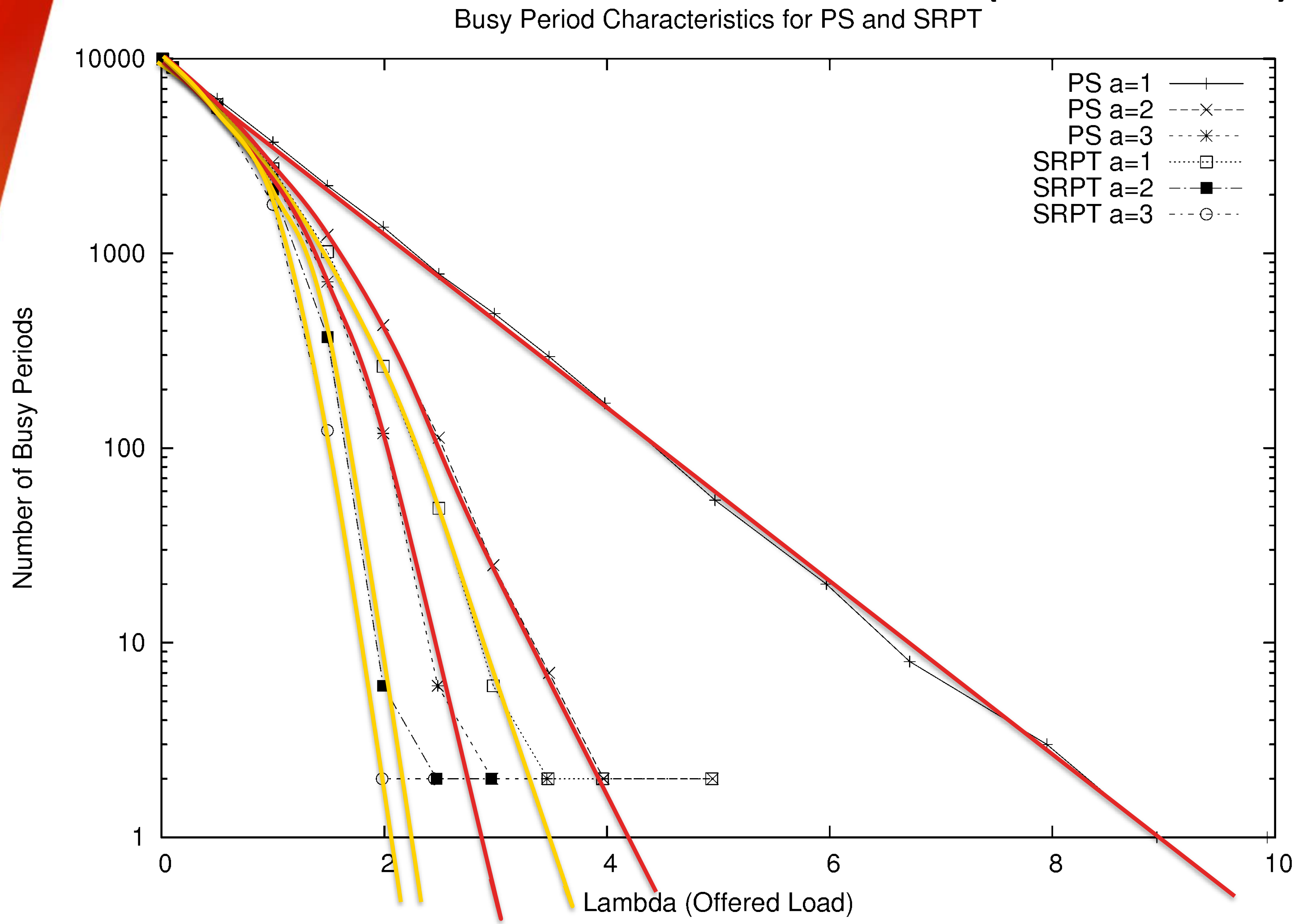


Comparing PS and SRPT

- Similarities:
  - Mean system speed (invariant property)
  - Mean system occupancy (invariant property)
  - Effect of α (i.e., the shift, the squish, and the squeeze)

- Differences:
  - Variance of system occupancy (SRPT is lower)
  - Mean response time (SRPT is lower)
  - Variance of response time (SRPT is higher)
  - PS is always fair; SRPT is unfair (esp. with speed scaling!)
  - Compensation effect in PS
  - Procrastination/starvation effect in SRPT

Busy Period Structure for PS and SRPT (simulation)



Simulation Insights and Observations

- Under heavy load, busy periods coalesce and U → 1
- Saturation points for PS and SRPT are <u>different</u>
  - Different “overload regimes” for PS and SRPT
  - Gap always exists between them
  - Gap shrinks as α increases
  - Limiting case ($\alpha = \infty$) requires $\rho < 1$ (i.e., fixed rate)
- SRPT suffers from starvation under very high load
- “Job count” stability and “work” stability differ

Summary: Theory

- The autoscaling properties of dynamic speed scaling systems are many, varied, and interesting!
    - Autoscaling effect: stable even at very high offered load ($s = \rho$)
    - Saturation effect: $U \rightarrow 1$ at heavy load, with $N \rightarrow \rho^\alpha$
    - The α effect: the shift, the squish, and the squeeze
- Invariant properties are helpful for analysis
- Differences exist between PS and SRPT
    - Variance of system occupancy; mean/variance of response time
    - Saturation points for PS and SRPT are different
    - SRPT suffers from starvation under very high load
- Our results suggest that PS becomes superior to SRPT for coupled speed scaling, if the load is high enough

Talk Outline

- Introduction and Motivation
- Background and Literature Review
- Summary of Key Results and Insights
- Recent Results and Contributions
  - Practice: Experimental Measurements
  - Theory: Autoscaling Effects
- Conclusions and Future Directions

Concluding Remarks

- There is broad and diverse literature on speed scaling systems for the past 20+ years
- There is a dichotomy between theoretical work and systems work on speed scaling
- Modern processors provide surprisingly rich functionality for speed scaling that is not yet well exploited by systems software
- There are many interesting tradeoffs to explore in dynamic speed scaling systems

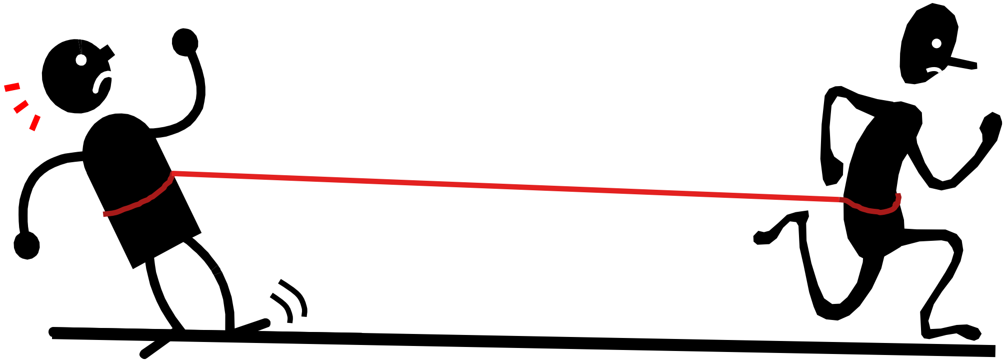
Future Directions

- Cost function for speed scaling optimization
- Redefining the benchmark for fairness
- Stability (or quasi-stability) in overload regimes
- Extending PSBS to speed scaling scenario
- Practical schedulers and speed scalers for modern operating systems that better exploit the available hardware features
- Speed scaling policies on multi-core systems

The End

- Thank you!

- Questions?



- For more info: carey@cpsc.ucalgary.ca