The Ondemand Governor

Venkatesh Pallipadi
Alexey Starikovskiy
Presented by: Len Brown

Intel Open Source Technology Center

July 19, 2006

The Authors

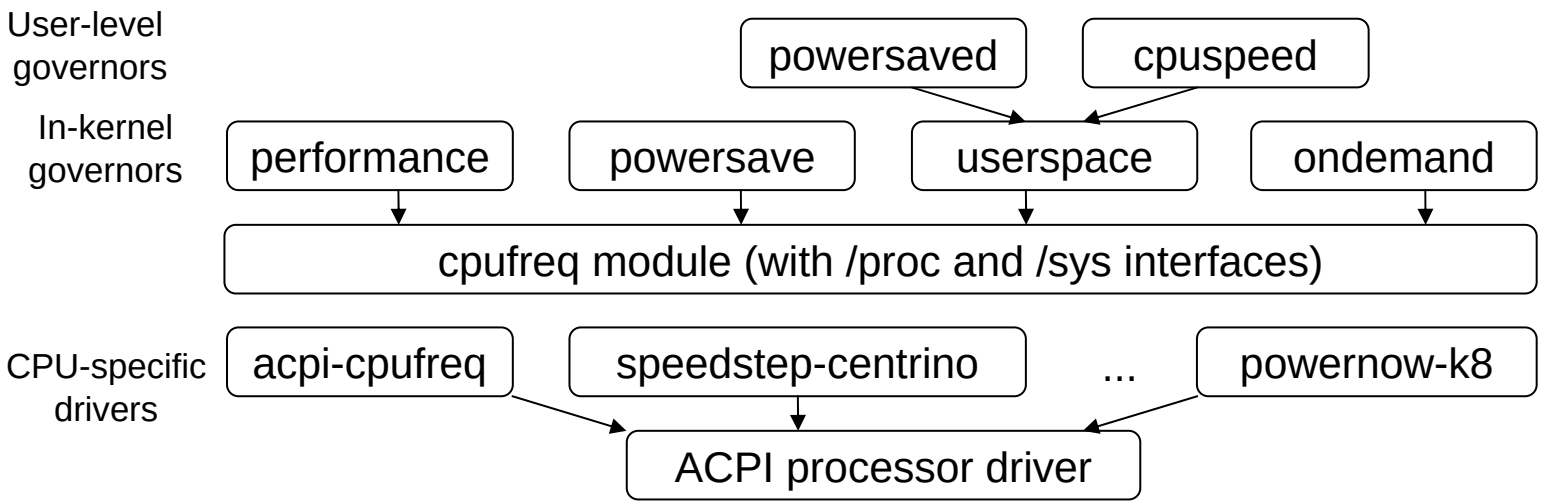
Venkatesh Pallipadi

Alexey Starikovskiy

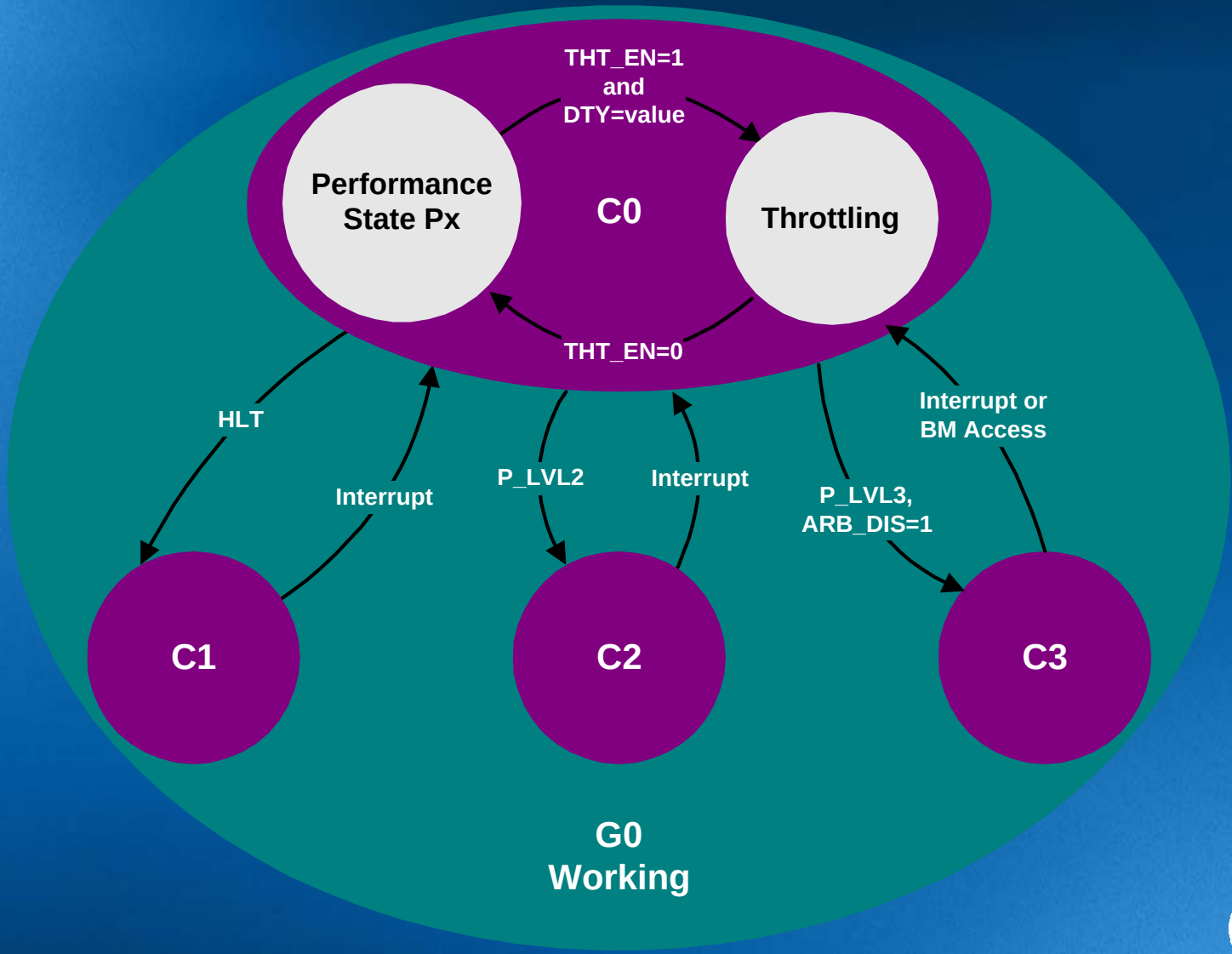
Agenda

- Background: states, cpufreq & ondemand
- Electricity
- 2.6.9 baseline results
- Newer Results

Cpufreq Implementation Architecture



ACPI Processor Power States



Key points on processor states

- C0: instructions are executed
- C1, C2...Cn: instructions are NOT executed

- Within C0
  - P0: Maximum MHz, Maximum Power
  - Pn: Minimum MHz, Minimum Power

  - T0: Maximum MHz Maximum Power
  - Tn: Minimum MHZ, Minimum Power

Performance States (ACPI P-States)

- Varies voltage and frequency
- Power varies with V^2
- Much more efficient than T-states
  - P0: 2000 MHz (HFM) Vcc=1.3V
  - P1: 1667 MHz
  - P2: 1333 MHz
  - P3: 1000 MHz (LFM) Vcc=1.0V
- LFM is maximum performance/power
  - Reducing further with T-states just delays how long it takes to get into power saving Cx state

P-States in Linux

- /sys/devices/system/cpu/cpu*/cpufreq/
- governors:
  - performance: fixed at P0
  - powersave: fixed at Pn
  - userspace: typically varied once/sec
  - ondemand: fully automatic
- ondemand is almost always preferred
  - Can react to short burst in load
  - can still set it's min/max from user-space if you really want additional user control

Frequency Scaling without Voltage Scaling saves no Energy

- Power [Watts]= f(cycles/time) [MHz]
- Energy = Power * time
- Energy [KWHr] = f(cycles)

- eg. cut frequency in half, cuts performance by half, but takes same energy to complete the task.
- ie. Do not use p4clockmod for anything unless the goal is to make it run slowly.

Clock Throttling (T-states)

- 8 steps available.
  - eg 2000/1750/1500/1250/1000/750/500/250
- Linear reduction in Processor Power
- Linear reduction in Processor Performance

- Generally Throttling is NOT what you want
  - Unless infinite non-idle workload that you don't care how slow it runs

Userspace Governor

- Historical
- Reacts well to steady-state workloads
- Reacts poorly to bursty-workloads
  - Response time of interactive workloads severely impacted by userspace governor.

ondemand governor

- in-kernel
- reacts quickly to changes in workload
- initial implementation in 2.6.9

  For all CPUs

  if (> 80% busy) then P0

  if (< 20% busy) then down by 20%
- Multiple improvements since 2.6.9

Hardware Coordination

- HW_ALL
  - Register Accesses made on all cores
  - Hardware implements MAX(core0, core1)
    - Raise: any core can raise with local MSR
    - Lower: max core must lower w/ local MSR
  - If central decision thread, cross-interrupts needed.
  - If distributed decision threads, local MSR accesses are sufficient

  HW_ALL is Supported by Intel® Core™ Duo

BIOS Coordination

- Transparent to the OS
- Behaves like HW_ALL
- Implemented by BIOS SMM
- Used on HT systems
- inefficient

Software Coordination

- SW_ALL
  - OS must issue requests on all cores
  - Potential issues with off-line cores
  - Appropriate for distributed per-core threads
- SW_ANY
  - OS tracks MAX() function in software and issues a single request for entire package.
    - Minimal register access
    - No cross-processor interrupts necessary
    - Appropriate for central, or per-package thread

How to fool a Governor today

Single Thread Migration on SMP

- Linux thread affinity seems to handle this case

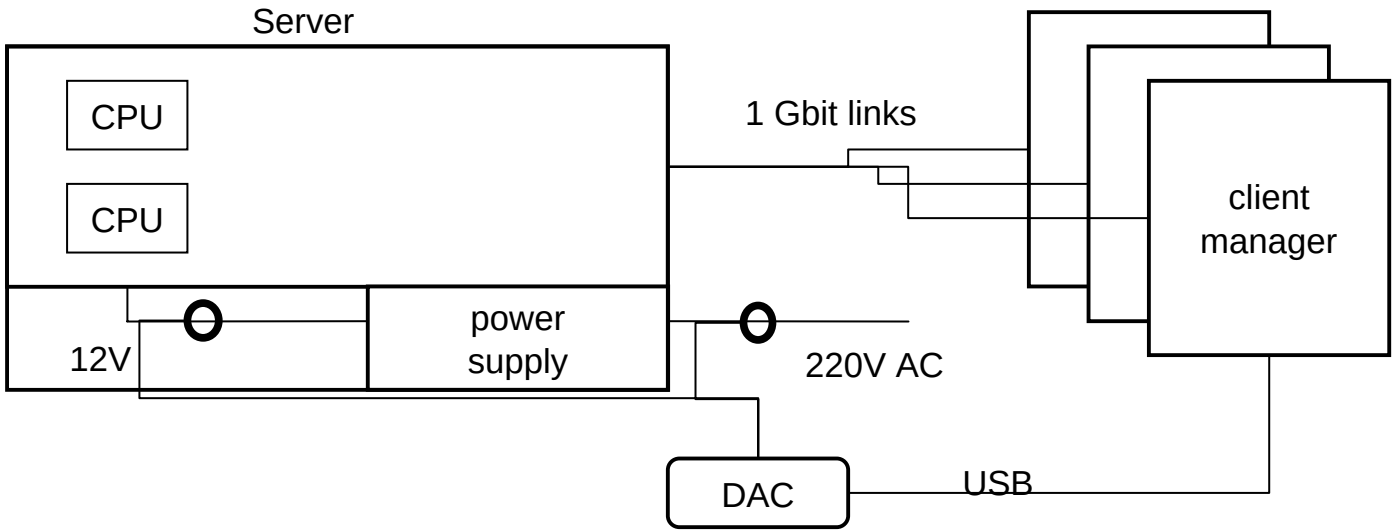
make –j1 the Linux Kernel on an SMP

- userspace will run it slower on SMP than on UP
- ondemand will will deliver UP performance

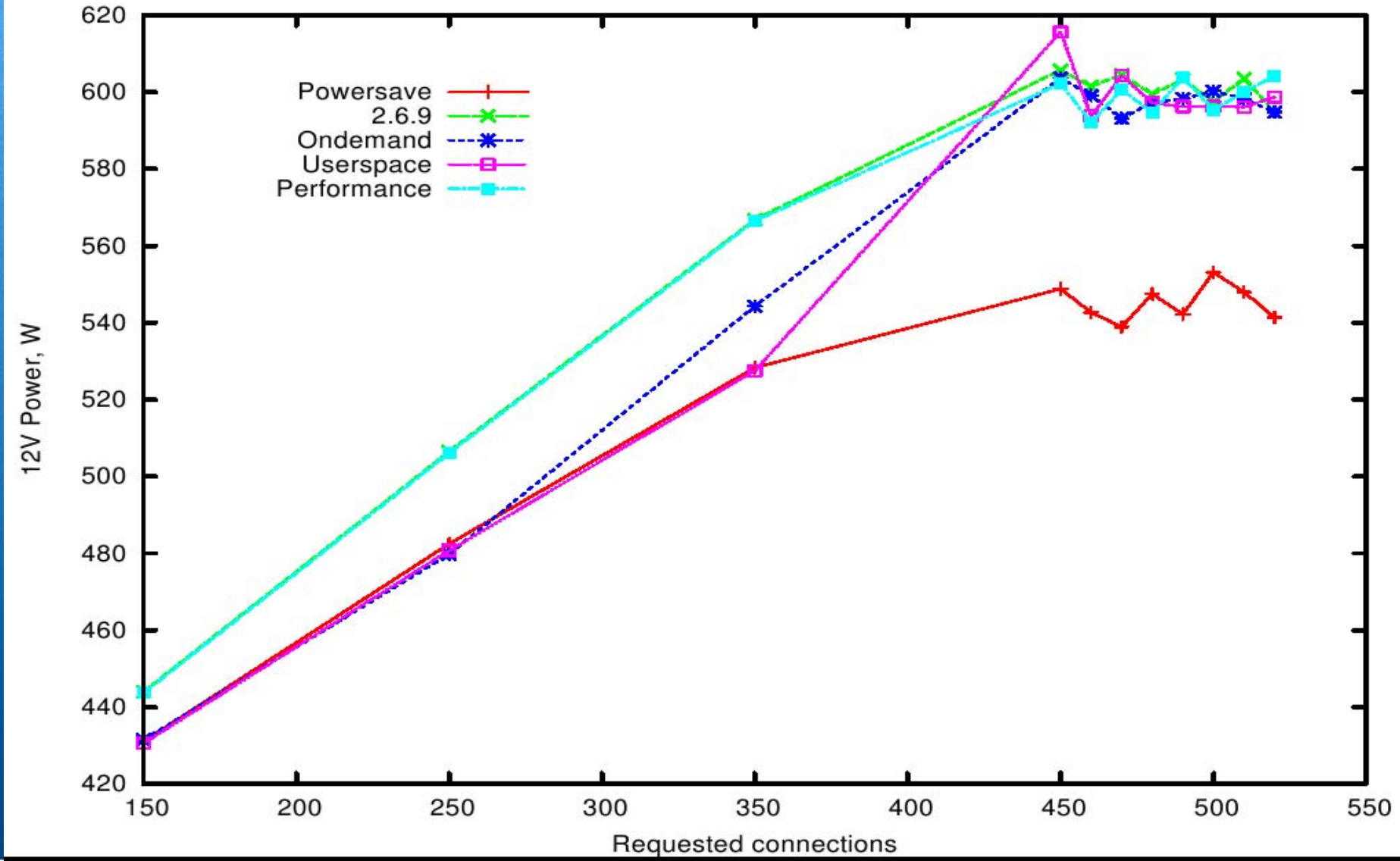
Improvements

- Automatic down-scaling
  - Jump directly to MHz that keeps 80% busy
- Coordination of dependent CPUs
- Unify up-scaling and down-scaling paths
  - "clean"
- Parallel calculation of utilization
  - Each CPU makes own decisions, lockless
- Dedicated workqueue (used to be keventd)

Measurement Test Bed



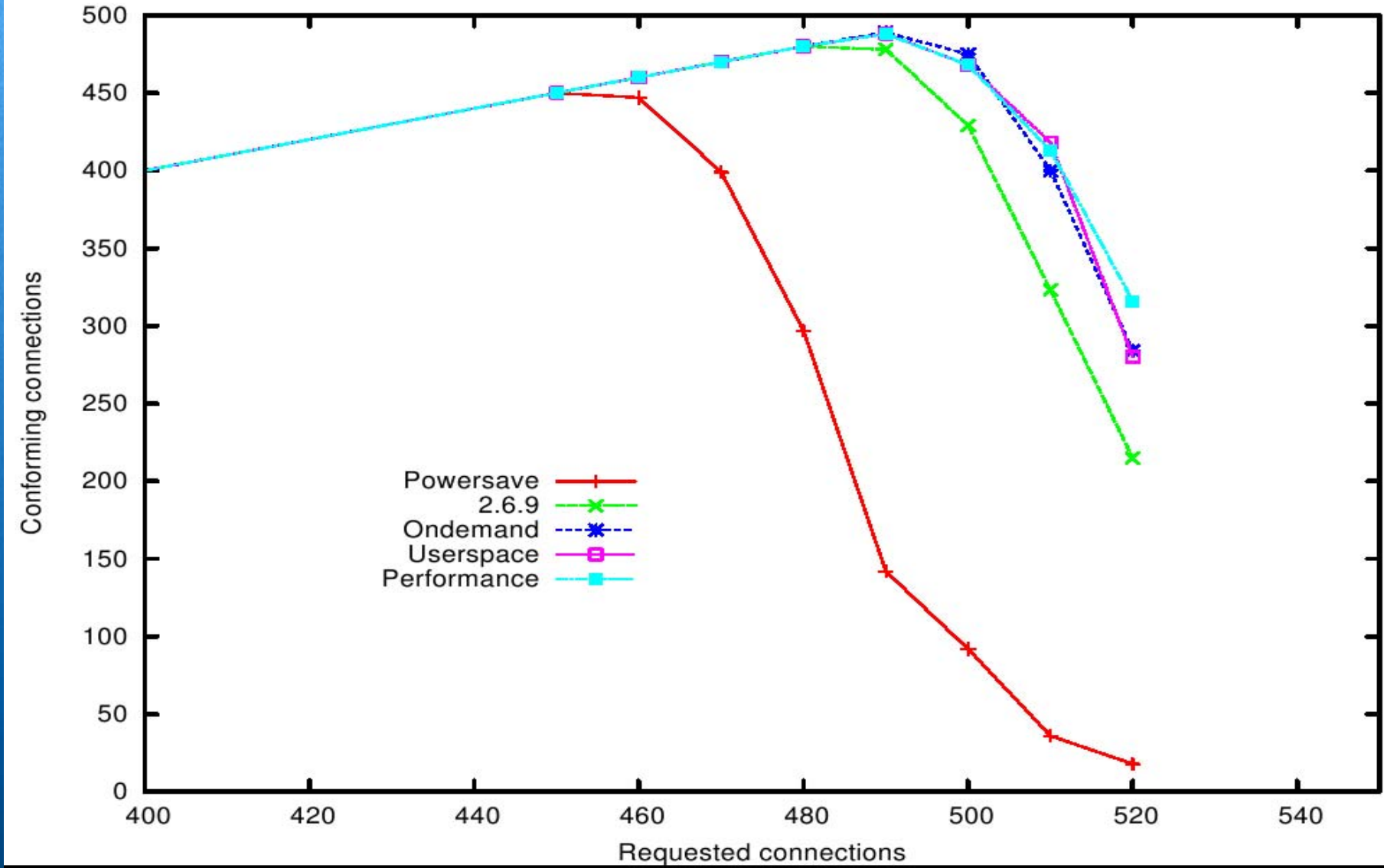
Baseline Power



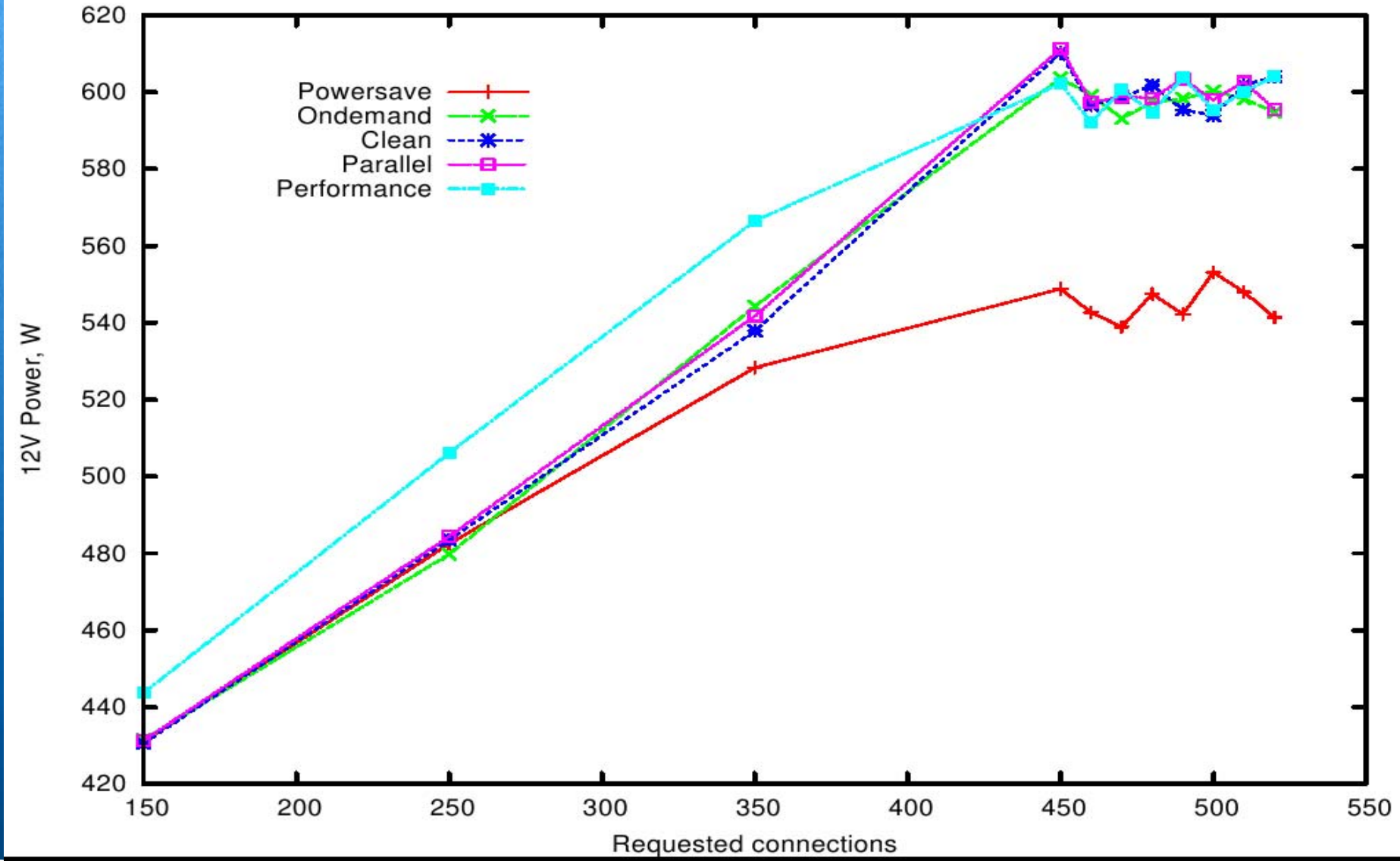
Server Workload

- Use SPECWeb99 and SPECWeb99-SSL
- Not a “tuned” benchmark run
  - use apache, not zeus
  - no tux acceleration
  - SPECWeb99 obsolete by ’05 anyway, but it is easy to run.
- Simply a convenient reproducible server workload.

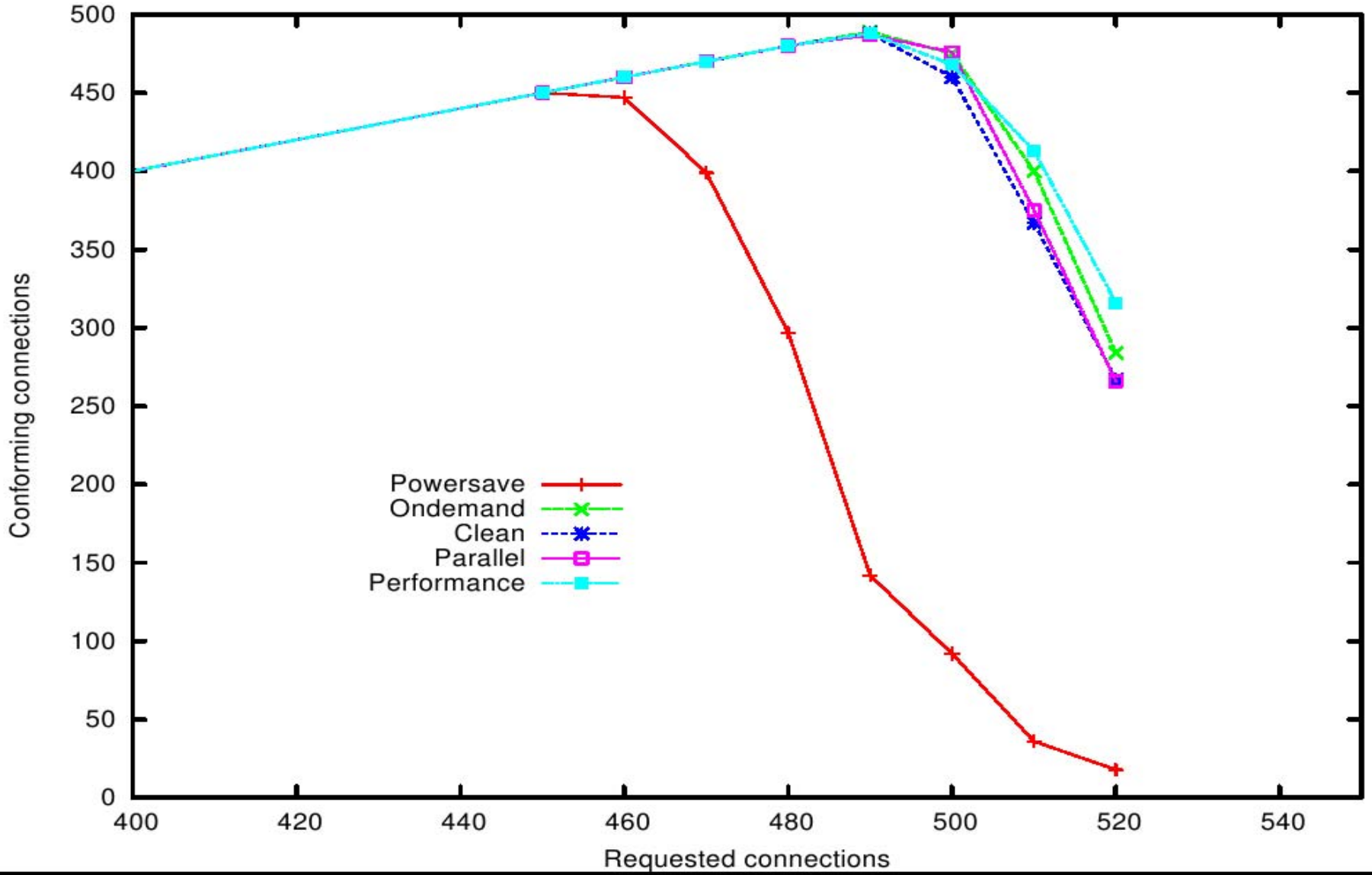
Baseline Performance



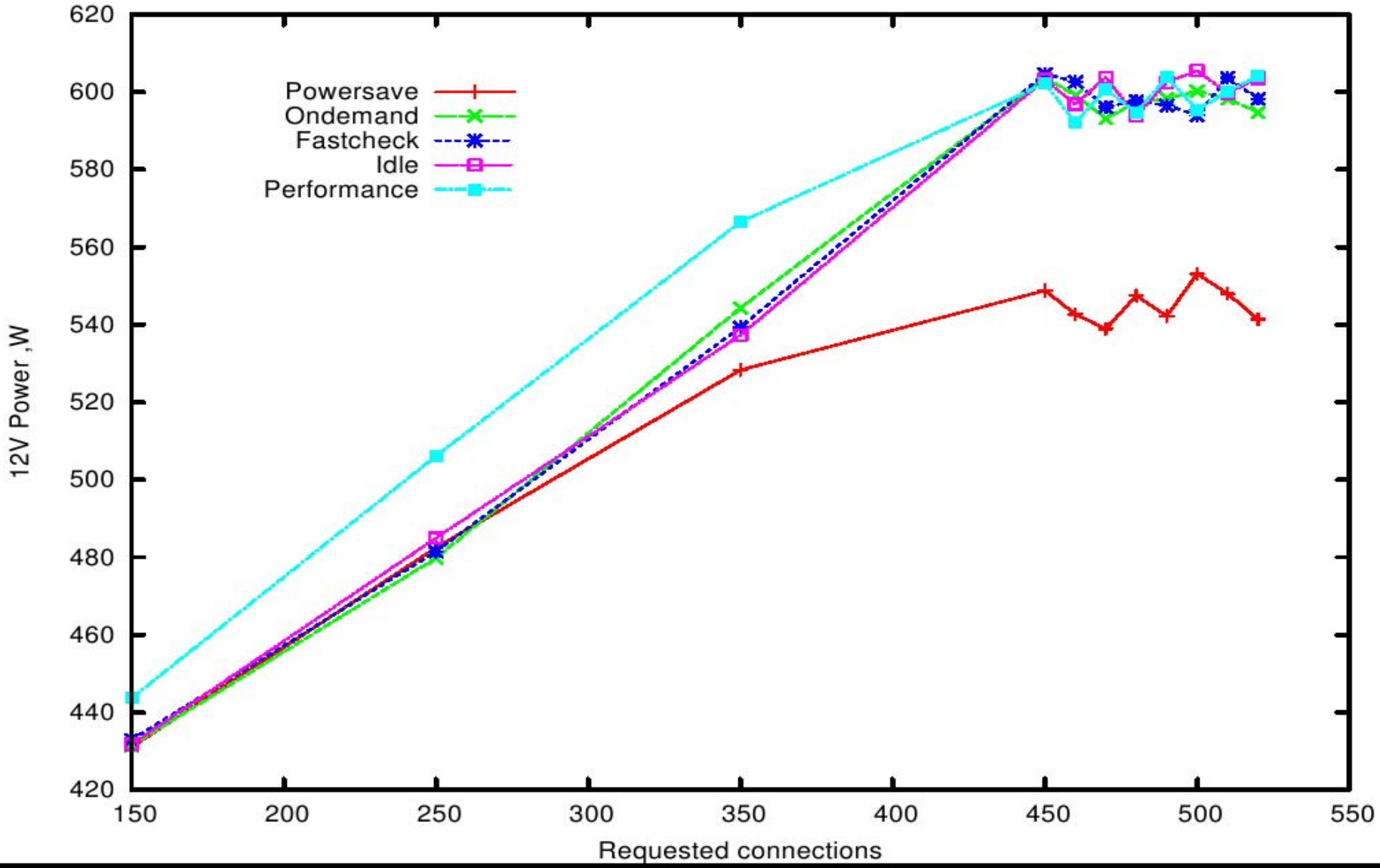
New Power


Powersave
Ondemand
Clean
Parallel
Performance
12V Power, W
Requested connections
620
600
580
560
540
520
500
480
460
440
420
150
200
250
300
350
400
450
500
550

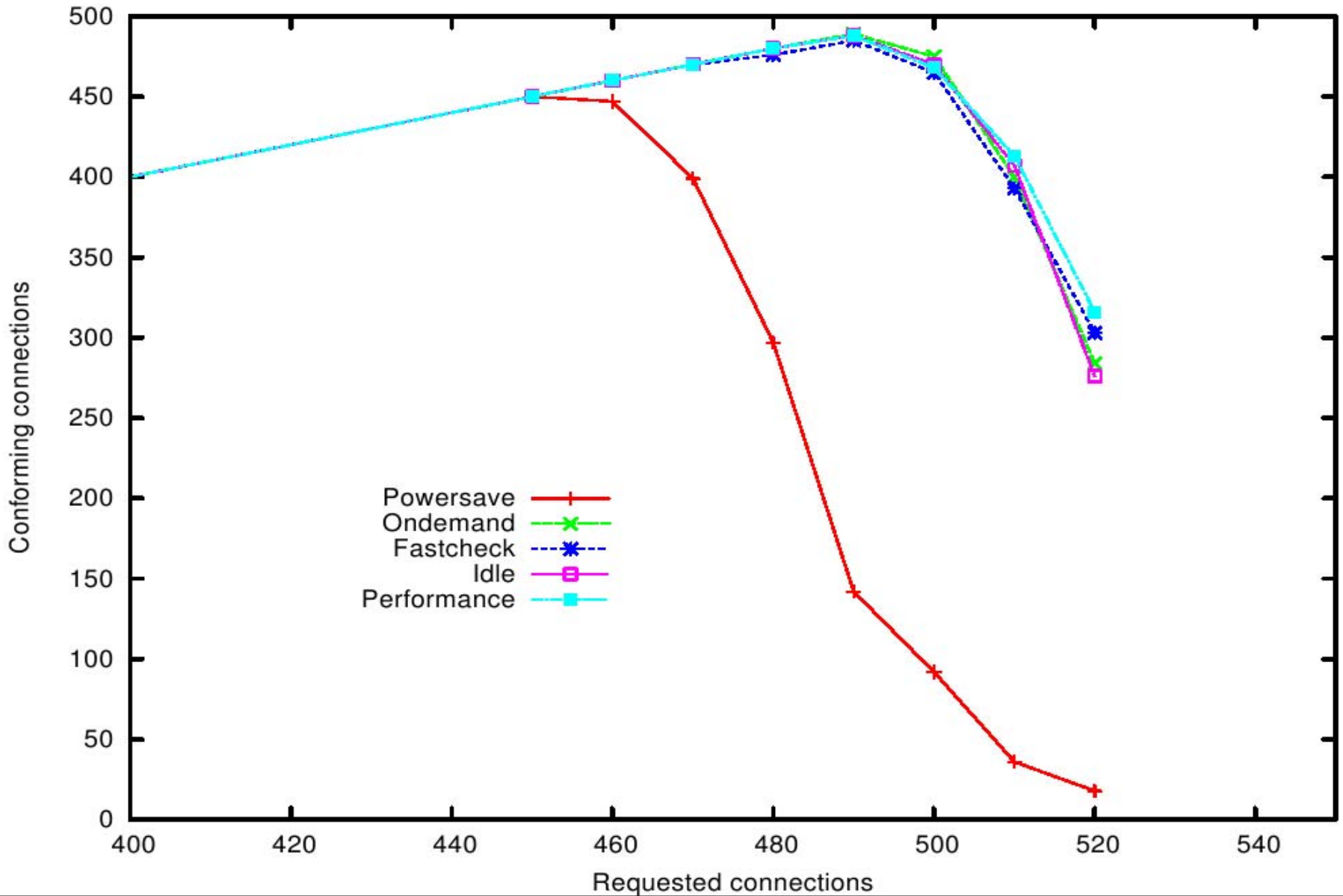
New Performance



Newer Power


Powersave
Ondemand
Fastcheck
Idle
Performance
12V Power ,W
Requested connections
620
600
580
560
540
520
500
480
460
440
420
150
200
250
300
350
400
450
500
550

Newer Performance



powersave_bias

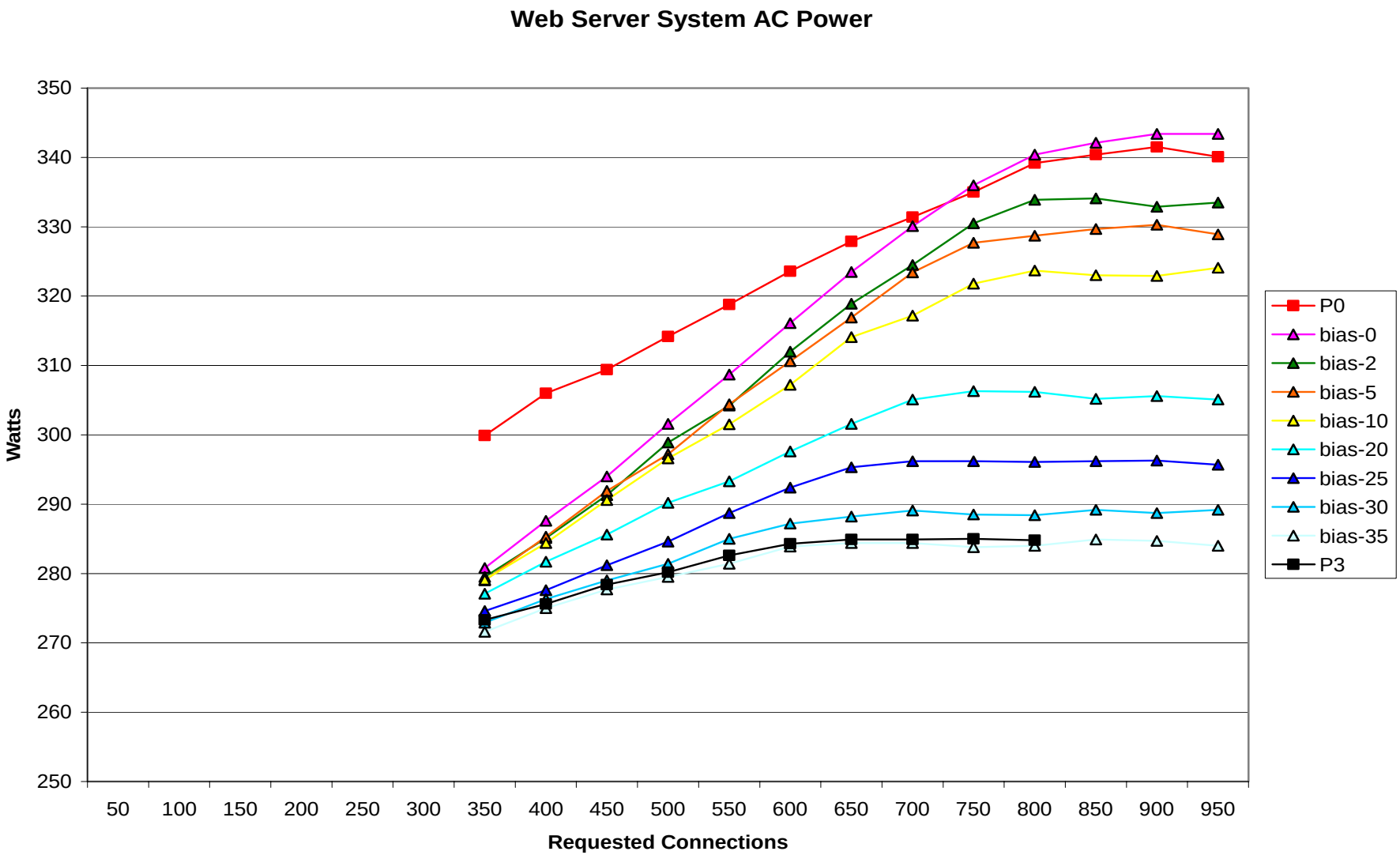
- ondemand attempts to retire workload at lowest MHz that results in zero idle time
- But power/performance increases as one approaches P0
- powersave_bias reduces ondemand target MHz by specified % (units are 0.1%)

powersave_bias Example: if ondemand target = 1.5 GHz

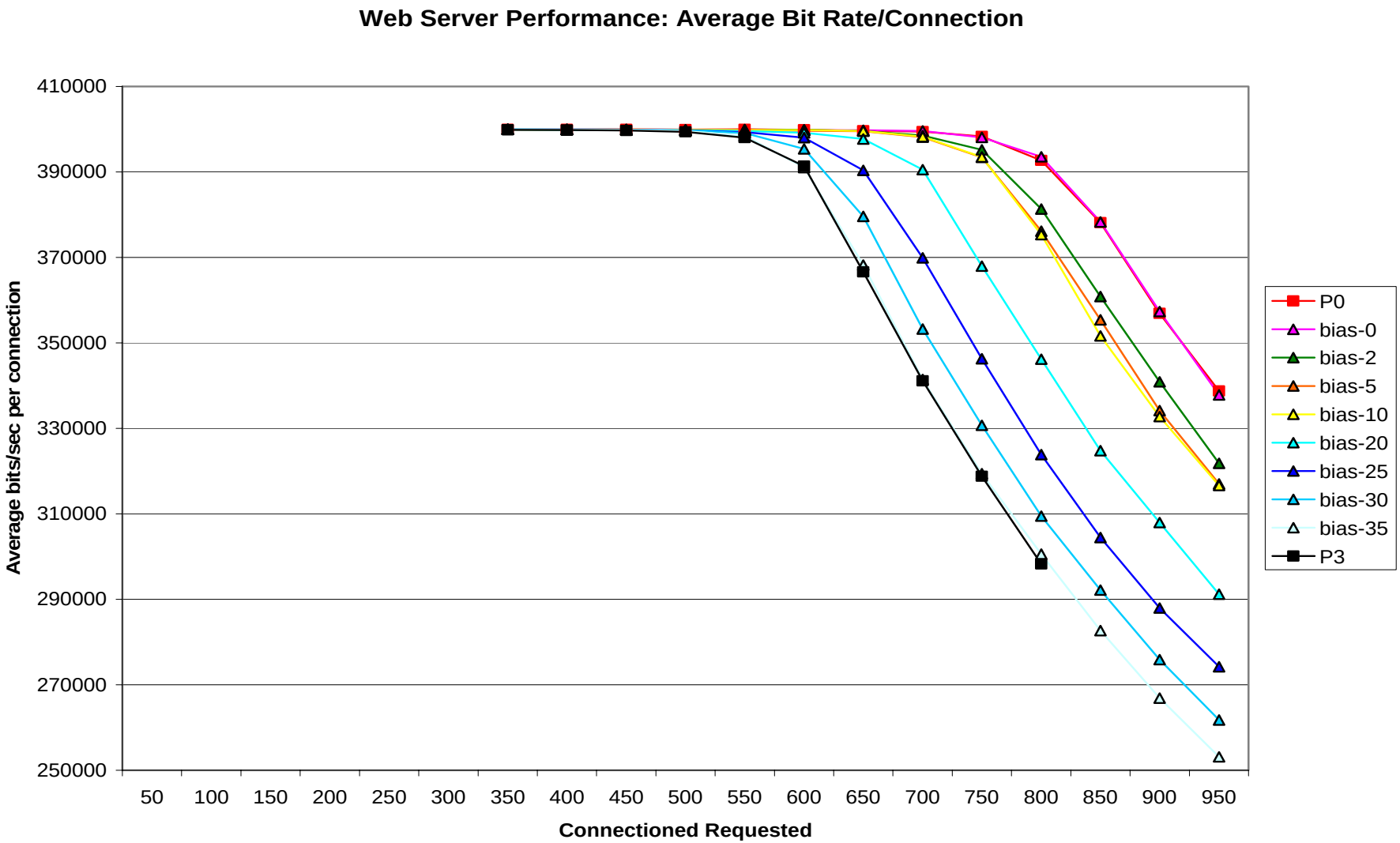
	Bias = 0% target 1.5	Bias = 2% target 1.47	Bias = 5% target 1.43	Bias = 25% target 1.13
P0 2.0 GHz				
P1 1.6 GHz	100%*	57%	43%	
P2 1.3 GHz		43%	57%	42%
P3 1.0 GHz				58%

*powersave_bias=0 (default) has no effect

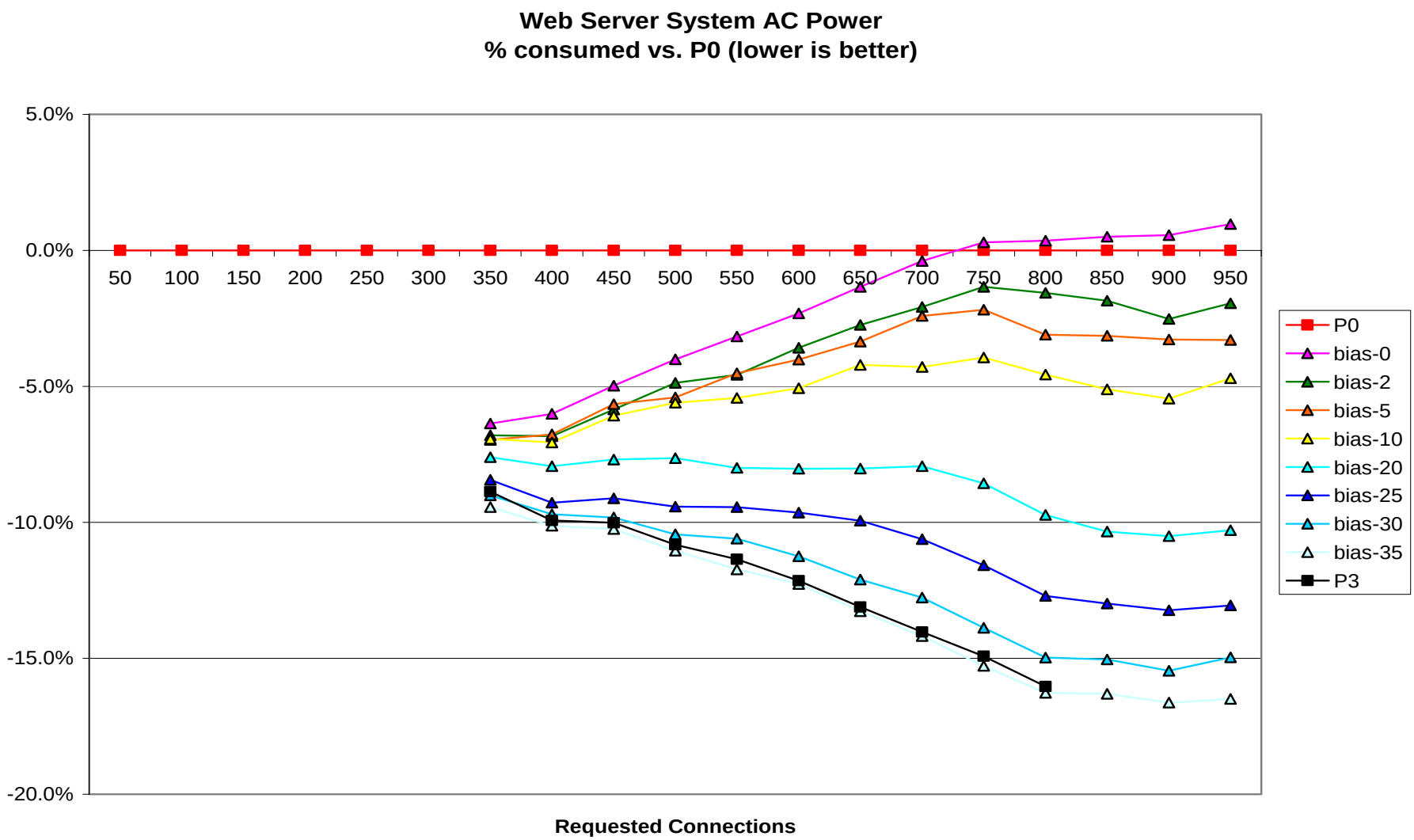
powersave_bias: Power



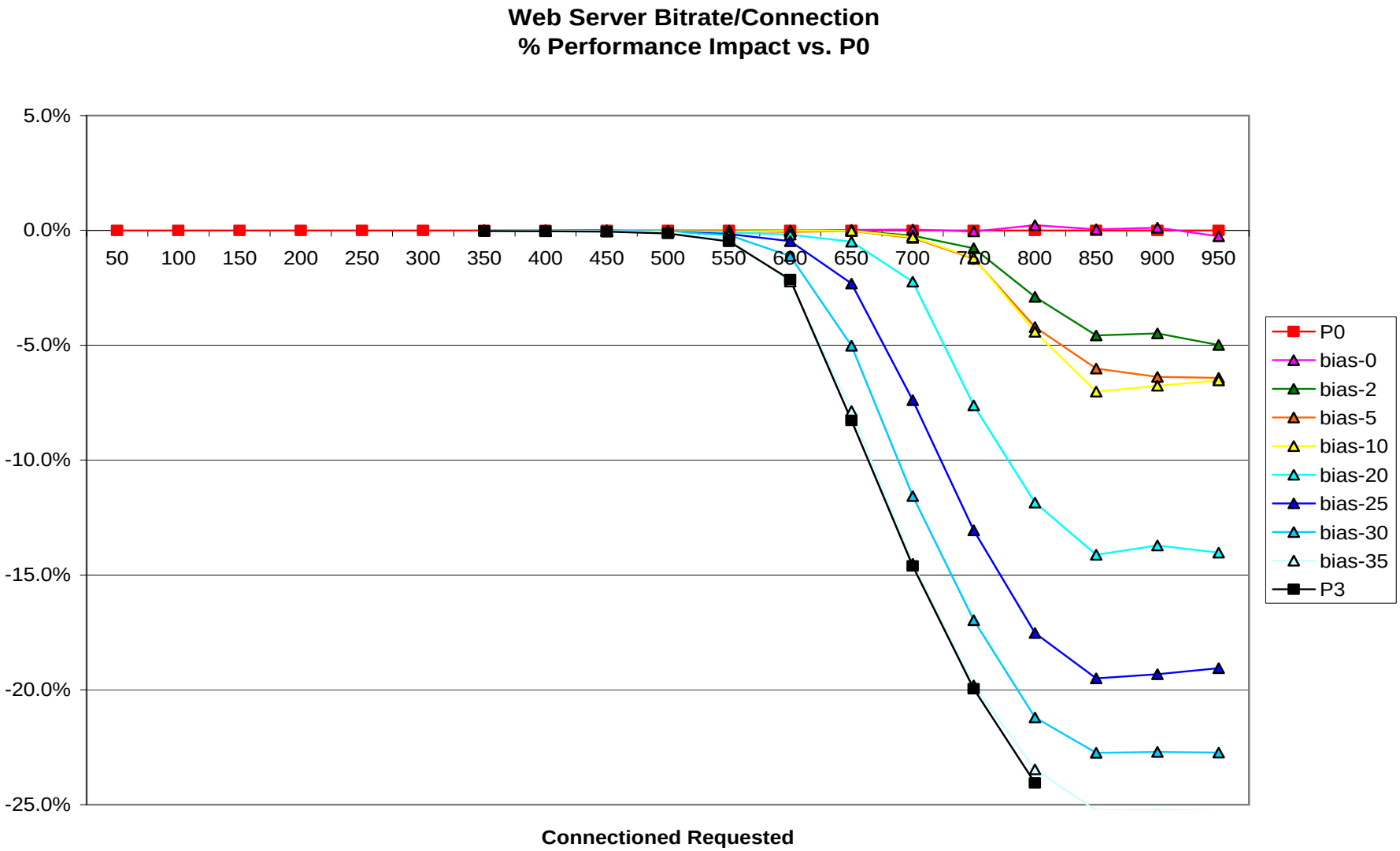
powersave_bias: Performance



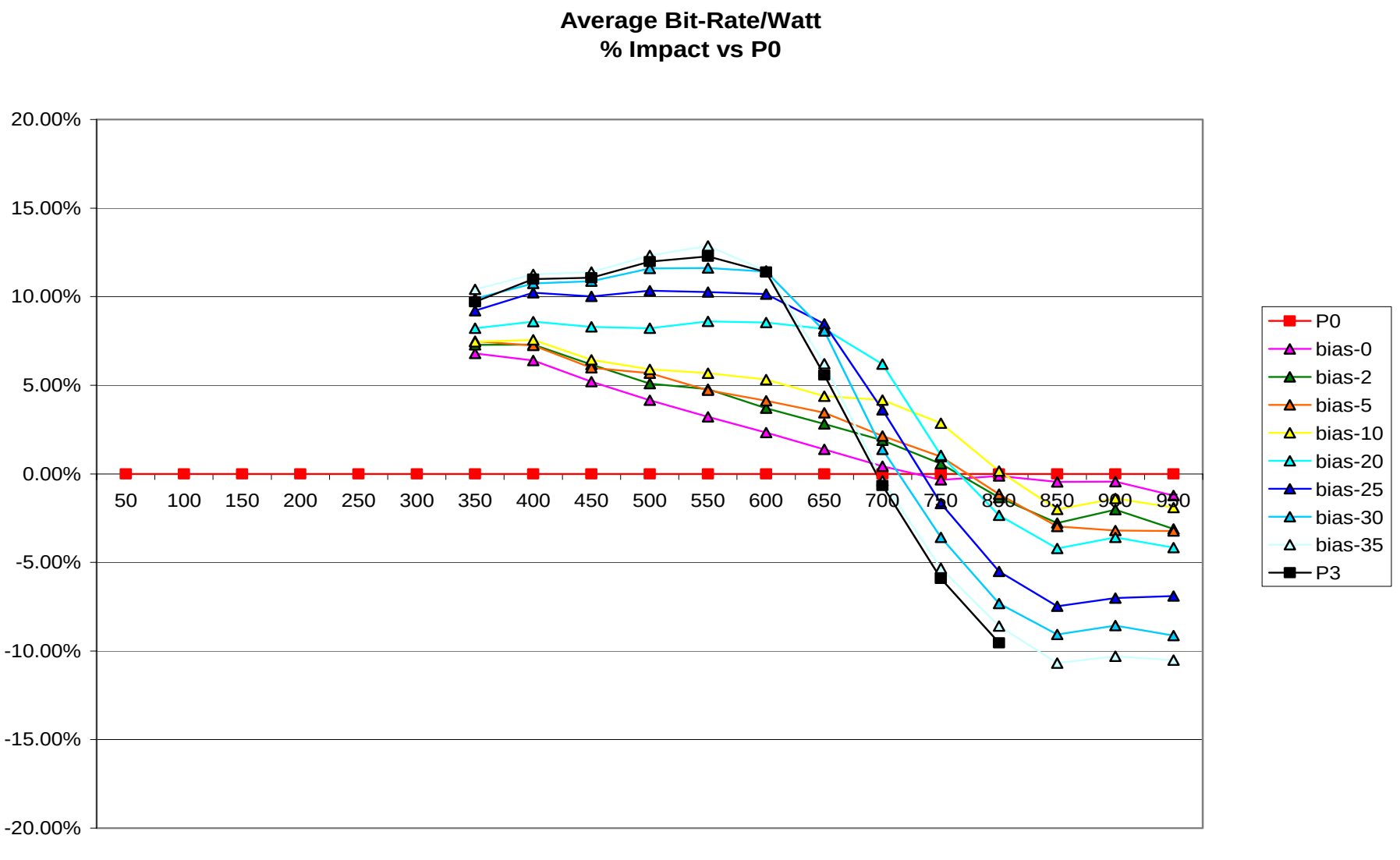
powersave_bias: Power %



powersave_bias: performance %



powersave_bias efficiency %



Future: Measuring Idle

- Input into ondemand algorithm
- Depends on HZ sampling of idle/busy
- “microstate” idle accounting expected to make ondemand smarter

Future: Power-Aware Scheduler

- Scheduler assumes all CPUs running at same speed
- Scheduler is currently optimized for max performance on HT, which is sub-optimal power on multi-socket multi-core.

Potential RT conflict

- RT could starve ondemand and prevent it from increasing MHz, which is needed for RT
- Needs coordination

Thank You

Intel Core Duo Processor SV

	Name	Vcc	Watt
C0	High Frequencey Mode (P0)	1.3	31
C0	Low Frequency Mode (Pn)	1.0	
C1	Auto Halt Stop Grant (HFM)		15.8
C1E	Enhanced Halt (LFM)		4.8
C2	Stop Clock (HFM)		15.5
C2E	Enhanced Stop Clock (LFM)		4.7
C3	Deep Sleep (HFM)		10.5
C3E	Enhanced Deep Sleep (LFM)		3.4
C4	Intel Deeper Sleep	0.85	2.2
DC4	Intel Enhanced Deeper Sleep	0.80	1.8