From **imagination** to **impact**

Australian Government
Department of Communications, Information Technology and the Arts
Australian Research Council

NICTA Members

NICTA Partners

Saturday, 4 April 2009

NICTA

From **imagination** to **impact**

Australian Government
Department of Communications, Information Technology and the Arts
Australian Research Council

NICTA Members

ANU

UNSW

Department of State and Regional Development

The University of Sydney

Queensland Government

Griffith University

QUT

The University of Queensland

NICTA Partners

David Snowdon, Etienne Le Sueur, Stefan Petters and Gernot Heiser

Koala

A platform for Operating-System Level Power-Management

* This is a talk about Koala -- a platform which forgets heuristic-based power management techniques, and uses empirical models to allow real trade-offs between reduced performance and energy savings. It solves a serious problems facing power management researchers -- that platforms don't behave the way they're supposed to!

Talk outline

- Hardware is really complicated
  - over-simplifying assumptions

Koala is workload-aware, uses realistic models and has practical policies.

Need to say that energy is different to power. You need to save energy, but you need to manage the power. Energy = Power x Time.

Need to say that Koala manages CPU and memory energy.

Talk outline

1. Energy is **really** important!

Saturday, 4 April 2009

- Hardware is really complicated
  - over-simplifying assumptions

Koala is workload-aware, uses realistic models and has practical policies.

Need to say that energy is different to power. You need to save energy, but you need to manage the power. Energy = Power x Time.

Need to say that Koala manages CPU and memory energy.

Talk outline

1. Energy is **really** important!

2. PM is **really** hard.

Saturday, 4 April 2009

- Hardware is really complicated
  - over-simplifying assumptions

Koala is workload-aware, uses realistic models and has practical policies.

Need to say that energy is different to power. You need to save energy, but you need to manage the power. Energy = Power x Time.

Need to say that Koala manages CPU and memory energy.

Talk outline

1. Energy is **really** important!
2. PM is **really** hard.
3. Koala helps... how?

- ☑ Workload-aware
- ☑ Realistic models
- ☑ Practical policies

- Hardware is really complicated
  - over-simplifying assumptions

Koala is workload-aware, uses realistic models and has practical policies.

Need to say that energy is different to power. You need to save energy, but you need to manage the power. Energy = Power x Time.

Need to say that Koala manages CPU and memory energy.

The importance of energy...

- Energy efficiency is really important!
- Each of you probably has a mobile phone in your pocket, and in this crowd, they're probably smart phones.
  - Mobile devices are energy-conscious for two reasons
    - –Thermal dissiption -- the devices are small and don't have space for heatsinks/fans.
    - –Battery lifetime -- power limits the number of operations that can be performed, which limits potential applications.

What about the cost of energy?
- Using energy has both an environmental and a monetary impact.
- A server has about the same CO2 emissions as 1.5 cars! (\cite[Reduce Energy Costs and Go Green With VMware Green IT Solutions]
–Energy-star compliance has become a big issue.
–VMWARE: In the United States alone, datacenters consumed $4.5 billion worth of electricity in 2006.
– VMWARE: 4 Tons of CO2 per server per year.

For all of these reasons we consider energy efficiency to be one of the premier problems in computer science and engineering.

The importance of energy...

NICTA

Saturday, 4 April 2009

- Energy efficiency is really important!
- Each of you probably has a mobile phone in your pocket, and in this crowd, they're probably smart phones.
  - Mobile devices are energy-conscious for two reasons
    - –Thermal dissiption -- the devices are small and don't have space for heatsinks/fans.
    - –Battery lifetime -- power limits the number of operations that can be performed, which limits potential applications.

What about the cost of energy?

- Using energy has both an environmental and a monetary impact.
- A server has about the same CO2 emissions as 1.5 cars! (\cite[Reduce Energy Costs and Go Green With VMware Green IT Solutions]
- –Energy-star compliance has become a big issue.
- –VMWARE: In the United States alone, datacenters consumed $4.5 billion worth of electricity in 2006.
- VMWARE: 4 Tons of CO2 per server per year.

For all of these reasons we consider energy efficiency to be one of the premier problems in computer science and engineering.

The importance of energy...

- Energy efficiency is really important!
- Each of you probably has a mobile phone in your pocket, and in this crowd, they're probably smart phones.
  - Mobile devices are energy-conscious for two reasons
    - –Thermal dissiption -- the devices are small and don't have space for heatsinks/fans.
    - –Battery lifetime -- power limits the number of operations that can be performed, which limits potential applications.

What about the cost of energy?
- Using energy has both an environmental and a monetary impact.
- A server has about the same CO_2 emissions as 1.5 cars! (\cite[Reduce Energy Costs and Go Green With VMware Green IT Solutions]
- –Energy-star compliance has become a big issue.
- –VMWARE: In the United States alone, datacenters consumed $4.5 billion worth of electricity in 2006.
- VMWARE: 4 Tons of CO_2 per server per year.

For all of these reasons we consider energy efficiency to be one of the premier problems in computer science and engineering.

Power Management

NICTA

Saturday, 4 April 2009

•Power management is really all about controlling power-related hardware knobs in order to achieve some goal.

•Some of those knobs are... (list knobs).

•These knobs trade performance against power.

• To limit our scope, we're looking at one of these hardware controlled knobs -- DVFS -- but there's no reason that, in the future, this approach couldn't be applied to other knobs which affect power/performance.

•These knobs are normally controlled in naiive ways: in Linux for example, there are two main CPU power management schemes -- ondemand is applies to DVFS. In academic terms, this is based on Mark Weiser's 1994 OSDI paper. That work was good, and applied well to systems at the time, but modern computers don't work in the same way.

•But there is so much academic research!? Why doesn't it ever get used? Answer: it could be, it just needs to be made practical. The answer is Koala. Koala bridges the gap between the real world and the academic world.

•Why are we using 1994 technology to run computers in 2009? They're simply not the same devices that they were.

Power Management

- Power management:
  - Controlling hardware *knobs*

Saturday, 4 April 2009

•Power management is really all about controlling power-related hardware knobs in order to achieve some goal.

•Some of those knobs are... (list knobs).

•These knobs trade performance against power.

• To limit our scope, we're looking at one of these hardware controlled knobs -- DVFS -- but there's no reason that, in the future, this approach couldn't be applied to other knobs which affect power/performance.

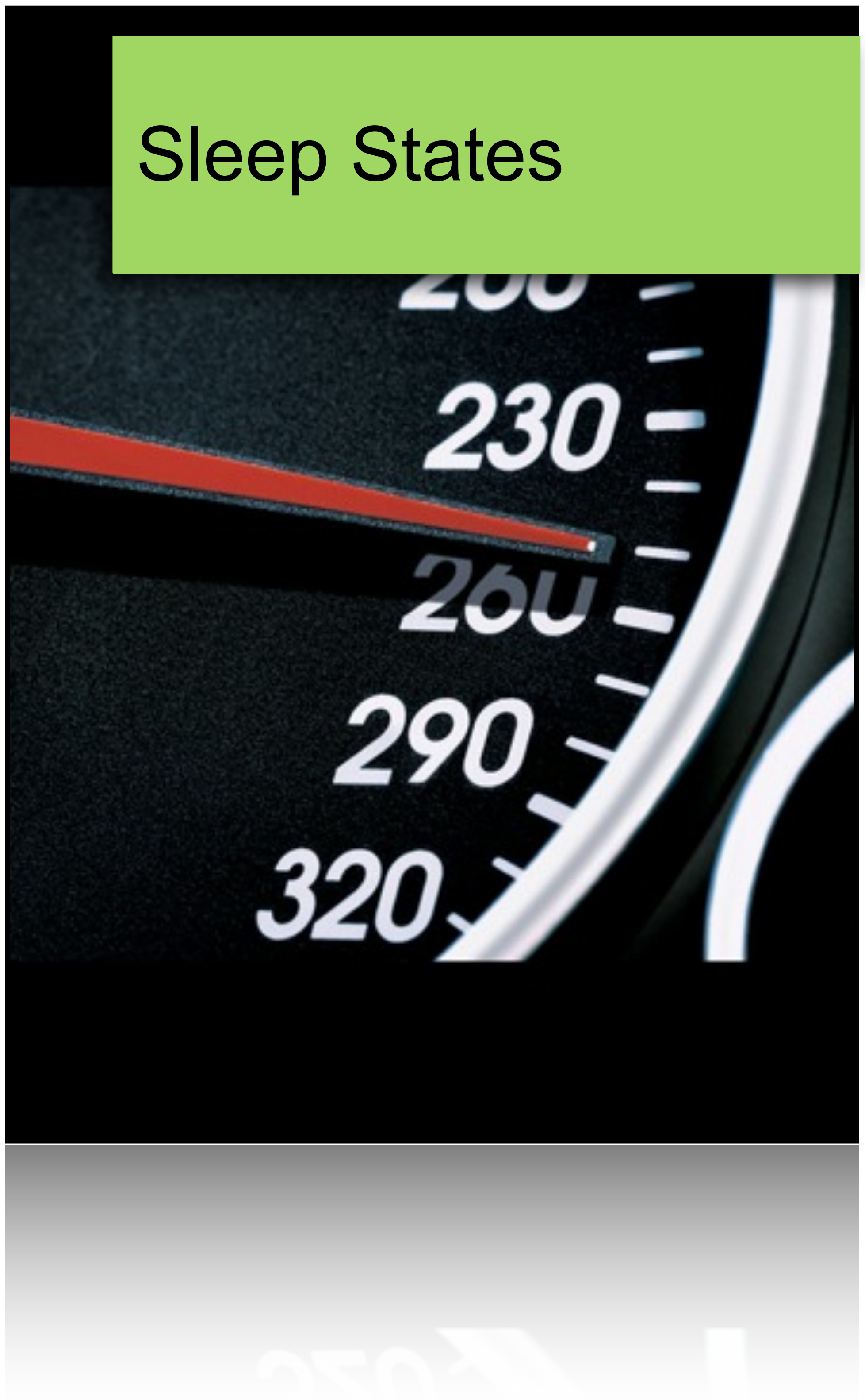
•These knobs are normally controlled in naiive ways: in Linux for example, there are two main CPU power management schemes -- ondemand is applies to DVFS. In academic terms, this is based on Mark Weiser's 1994 OSDI paper. That work was good, and applied well to systems at the time, but modern computers don't work in the same way.

•But there is so much academic research!? Why doesn't it ever get used? Answer: it could be, it just needs to be made practical. The answer is Koala. Koala bridges the gap between the real world and the academic world.

•Why are we using 1994 technology to run computers in 2009? They're simply not the same devices that they were.

Power Management

- Power management:
  - Controlling hardware *knobs*



Saturday, 4 April 2009

•Power management is really all about controlling power-related hardware knobs in order to achieve some goal.

•Some of those knobs are... (list knobs).

•These knobs trade performance against power.

• To limit our scope, we're looking at one of these hardware controlled knobs -- DVFS -- but there's no reason that, in the future, this approach couldn't be applied to other knobs which affect power/performance.

•These knobs are normally controlled in naiive ways: in Linux for example, there are two main CPU power management schemes -- ondemand is applies to DVFS. In academic terms, this is based on Mark Weiser's 1994 OSDI paper. That work was good, and applied well to systems at the time, but modern computers don't work in the same way.

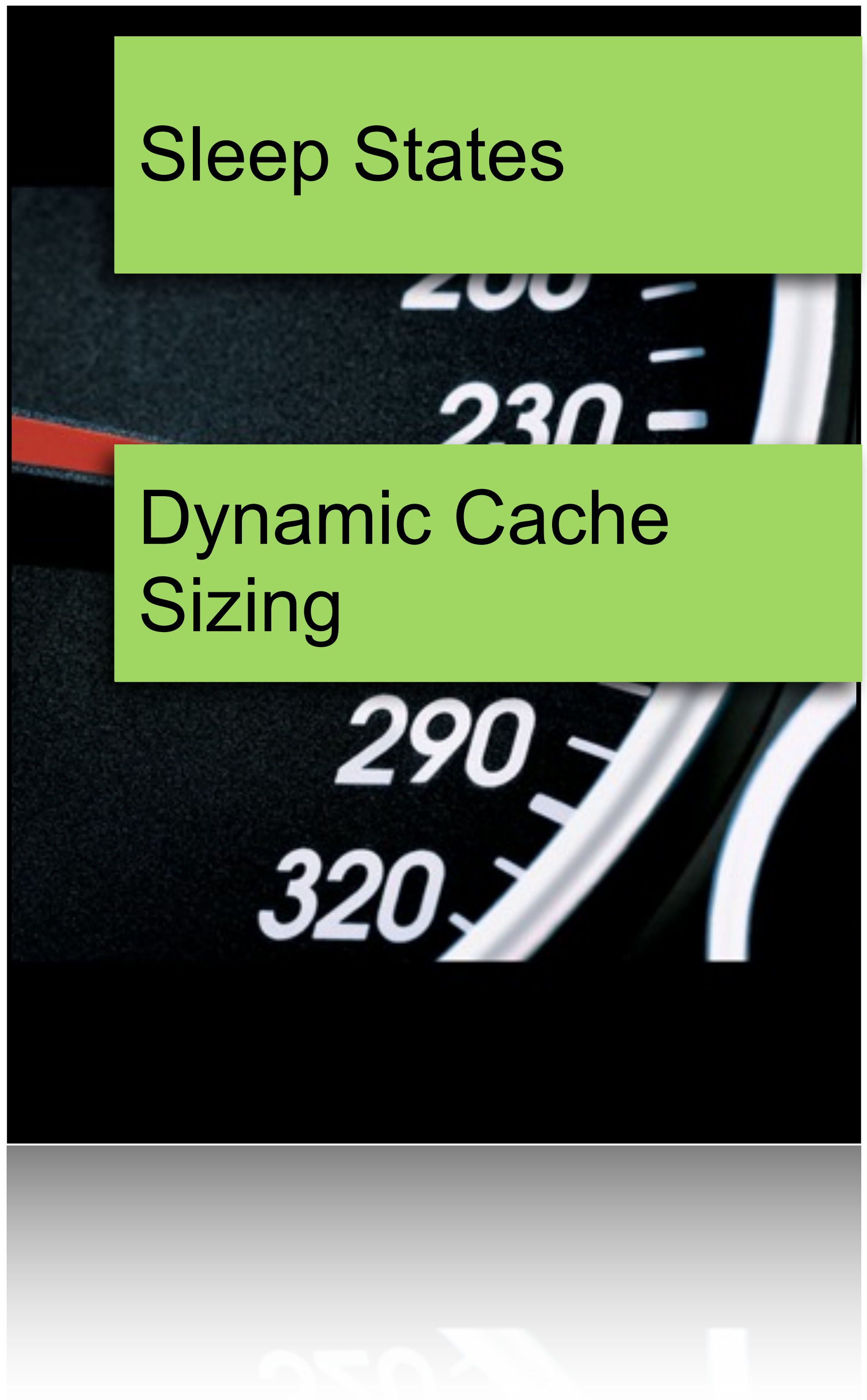
•But there is so much academic research!? Why doesn't it ever get used? Answer: it could be, it just needs to be made practical. The answer is Koala. Koala bridges the gap between the real world and the academic world.

•Why are we using 1994 technology to run computers in 2009? They're simply not the same devices that they were.

Power Management

NICTA

- Power management:
  - Controlling hardware *knobs*



●Power management is really all about controlling power-related hardware knobs in order to achieve some goal.

●Some of those knobs are... (list knobs).

●These knobs trade performance against power.

● To limit our scope, we're looking at one of these hardware controlled knobs -- DVFS -- but there's no reason that, in the future, this approach couldn't be applied to other knobs which affect power/performance.

●These knobs are normally controlled in naiive ways: in Linux for example, there are two main CPU power management schemes -- ondemand is applies to DVFS. In academic terms, this is based on Mark Weiser's 1994 OSDI paper. That work was good, and applied well to systems at the time, but modern computers don't work in the same way.

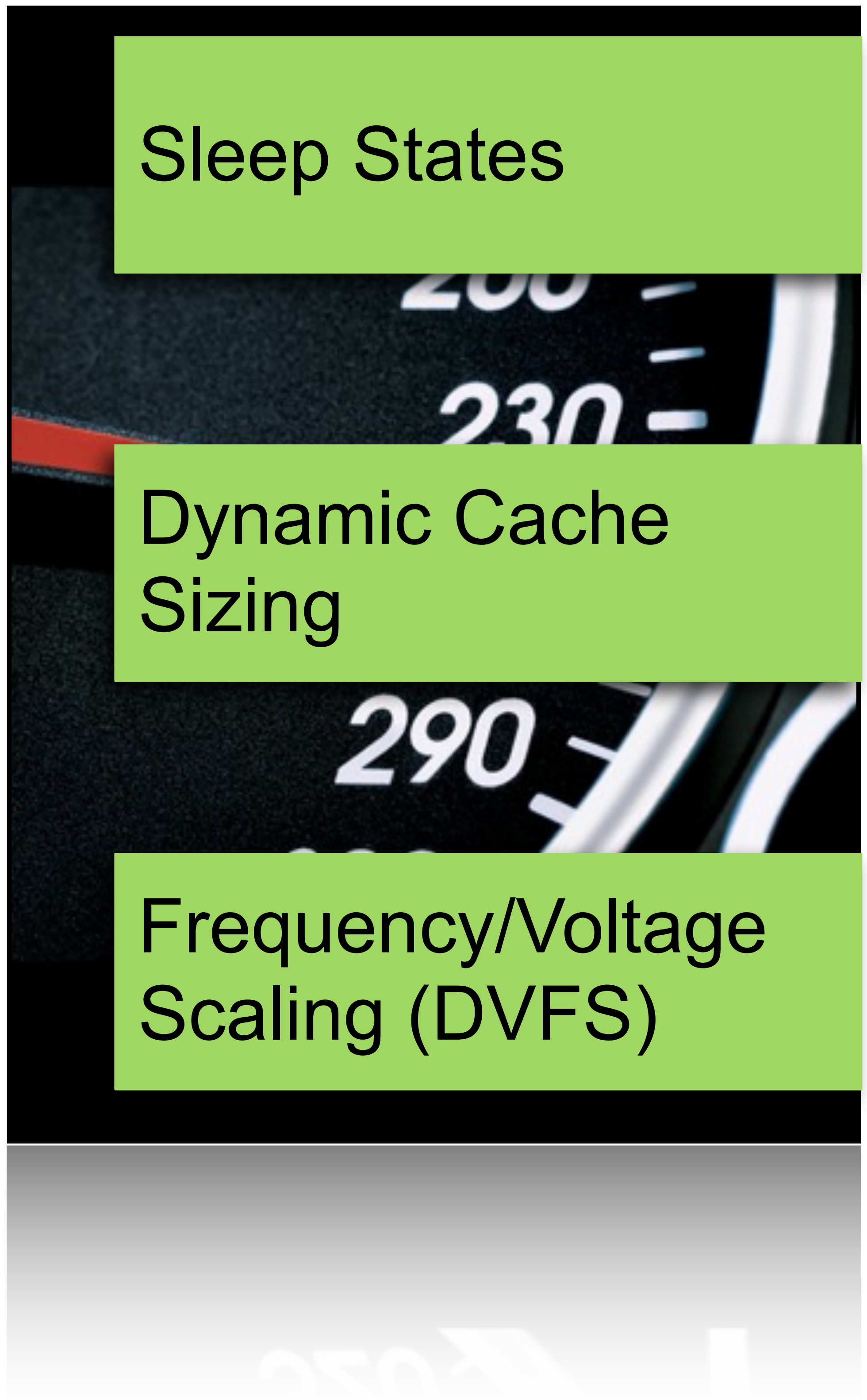
●But there is so much academic research!? Why doesn't it ever get used? Answer: it could be, it just needs to be made practical. The answer is Koala. Koala bridges the gap between the real world and the academic world.

●Why are we using 1994 technology to run computers in 2009? They're simply not the same devices that they were.

Power Management

NICTA

- Power management:
  - Controlling hardware *knobs*



- Power management is really all about controlling power-related hardware knobs in order to achieve some goal.

- Some of those knobs are... (list knobs).

- These knobs trade performance against power.

- To limit our scope, we're looking at one of these hardware controlled knobs -- DVFS -- but there's no reason that, in the future, this approach couldn't be applied to other knobs which affect power/performance.

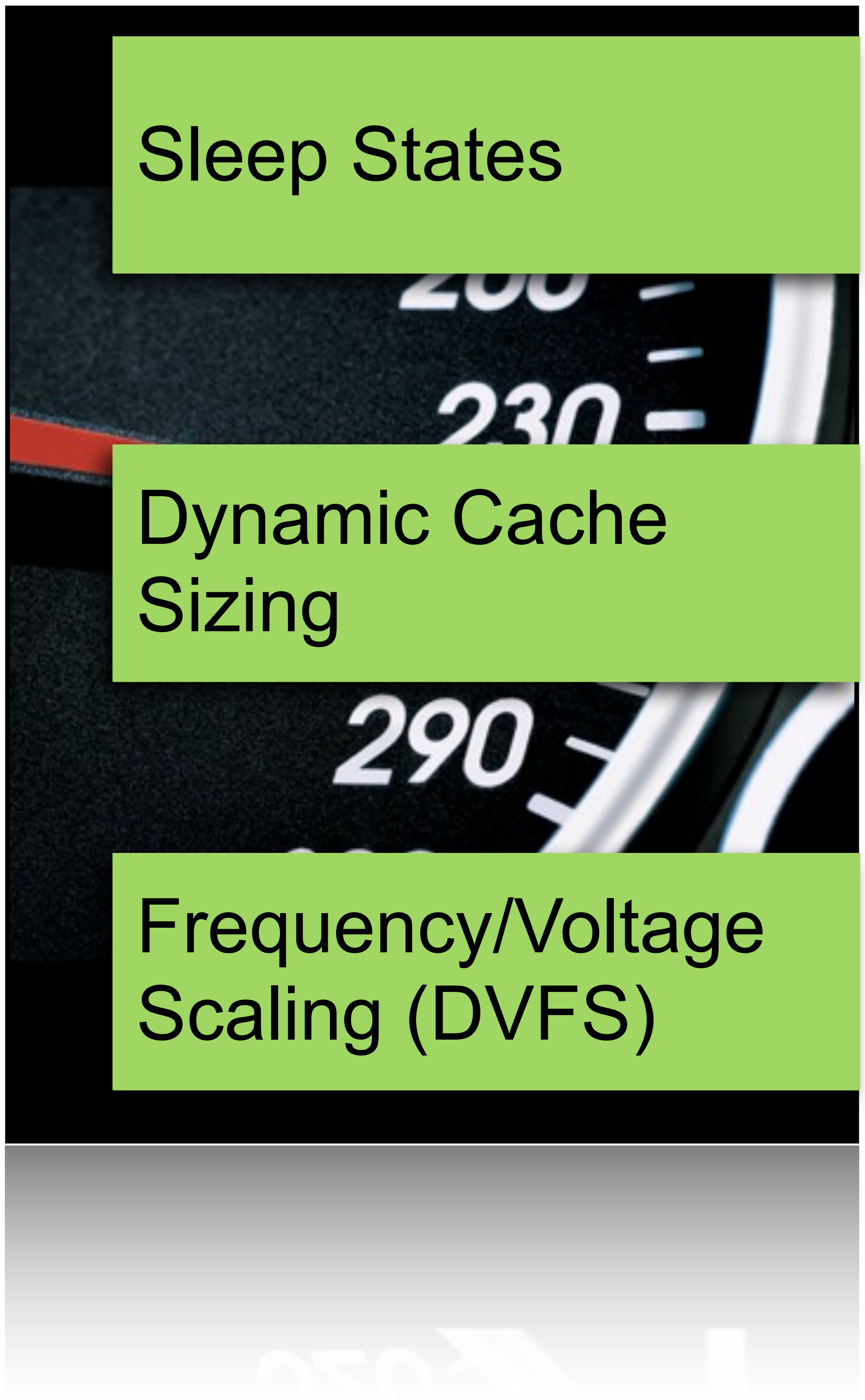
- These knobs are normally controlled in naiive ways: in Linux for example, there are two main CPU power management schemes -- ondemand is applies to DVFS. In academic terms, this is based on Mark Weiser's 1994 OSDI paper. That work was good, and applied well to systems at the time, but modern computers don't work in the same way.

- But there is so much academic research!? Why doesn't it ever get used? Answer: it could be, it just needs to be made practical. The answer is Koala. Koala bridges the gap between the real world and the academic world.

- Why are we using 1994 technology to run computers in 2009? They're simply not the same devices that they were.

Power Management

- Power management:
  - – Controlling hardware *knobs*
  - ➡ Performance vs. power



- Power management is really all about controlling power-related hardware knobs in order to achieve some goal.

- Some of those knobs are... (list knobs).

- These knobs trade performance against power.

- To limit our scope, we're looking at one of these hardware controlled knobs -- DVFS -- but there's no reason that, in the future, this approach couldn't be applied to other knobs which affect power/performance.

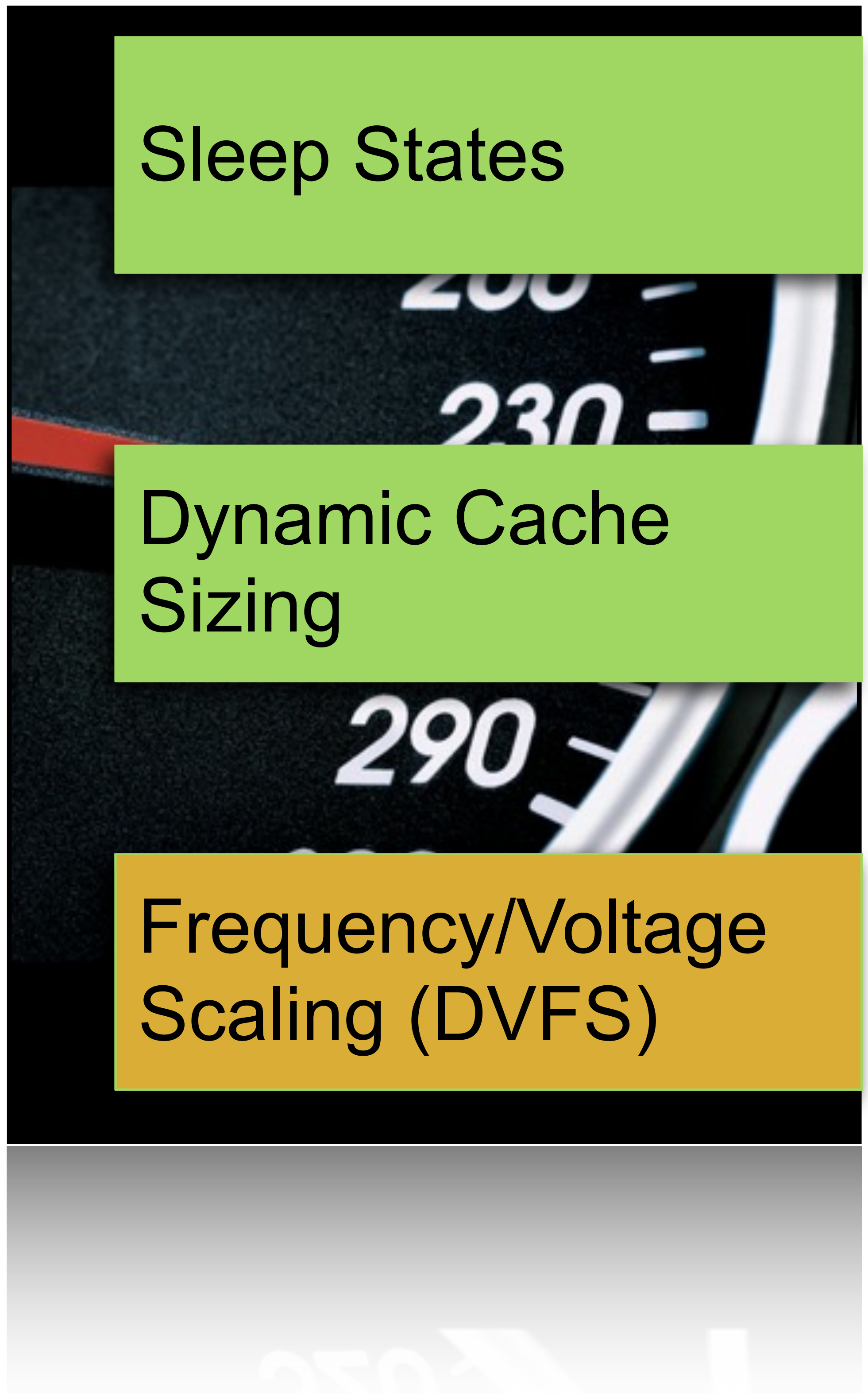
- These knobs are normally controlled in naiive ways: in Linux for example, there are two main CPU power management schemes -- ondemand is applies to DVFS. In academic terms, this is based on Mark Weiser's 1994 OSDI paper. That work was good, and applied well to systems at the time, but modern computers don't work in the same way.

- But there is so much academic research!? Why doesn't it ever get used? Answer: it could be, it just needs to be made practical. The answer is Koala. Koala bridges the gap between the real world and the academic world.

- Why are we using 1994 technology to run computers in 2009? They're simply not the same devices that they were.

Power Management

- Power management:
  - – Controlling hardware *knobs*
  - ➡ Performance vs. power



- Power management is really all about controlling power-related hardware knobs in order to achieve some goal.

- Some of those knobs are... (list knobs).

- These knobs trade performance against power.

- To limit our scope, we're looking at one of these hardware controlled knobs -- DVFS -- but there's no reason that, in the future, this approach couldn't be applied to other knobs which affect power/performance.

- These knobs are normally controlled in naiive ways: in Linux for example, there are two main CPU power management schemes -- ondemand is applies to DVFS. In academic terms, this is based on Mark Weiser's 1994 OSDI paper. That work was good, and applied well to systems at the time, but modern computers don't work in the same way.

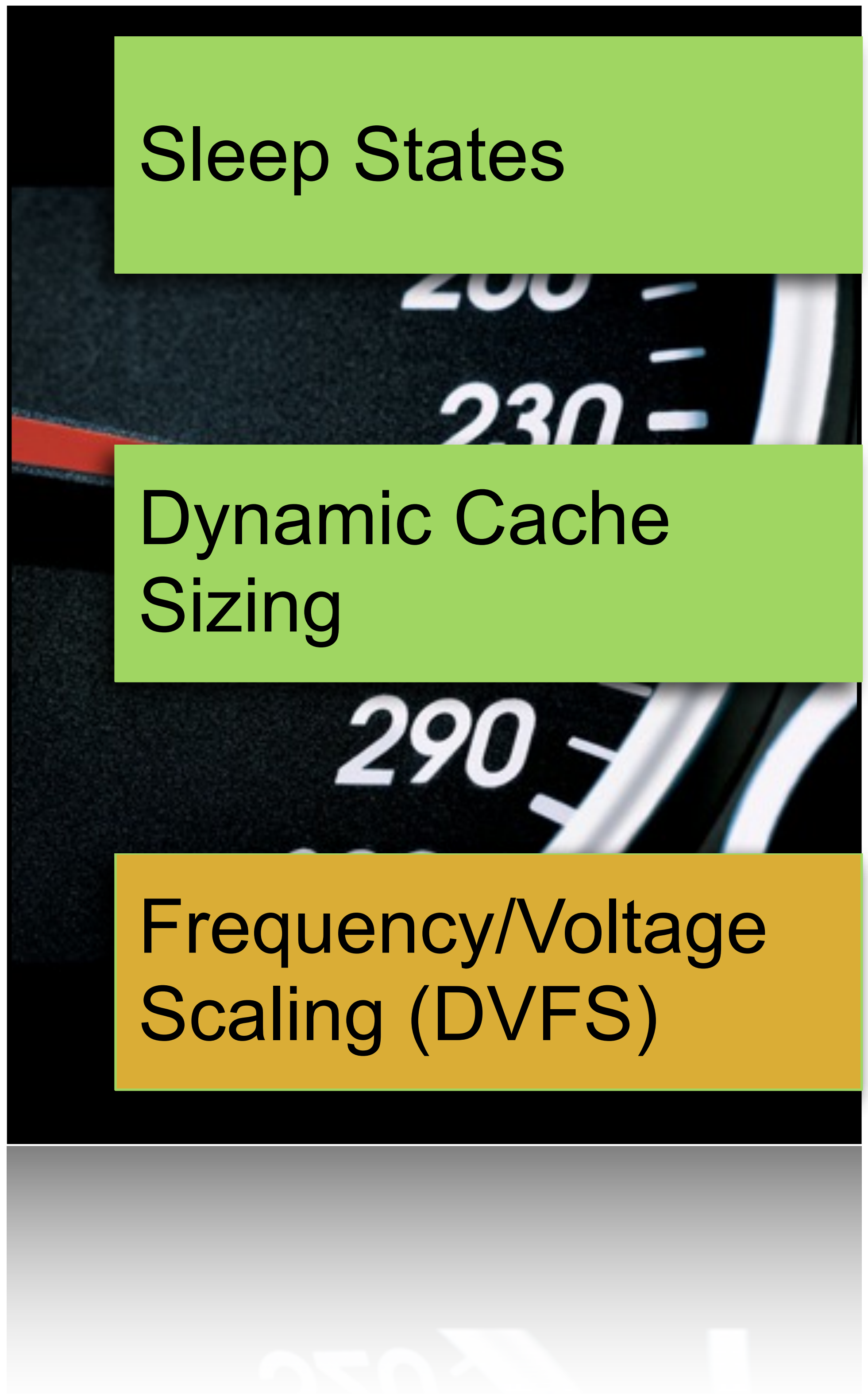
- But there is so much academic research!? Why doesn't it ever get used? Answer: it could be, it just needs to be made practical. The answer is Koala. Koala bridges the gap between the real world and the academic world.

- Why are we using 1994 technology to run computers in 2009? They're simply not the same devices that they were.

Power Management

- Power management:
  – Controlling hardware *knobs*
  ➡ Performance vs. power

- Power management practice
  – Linux ondemand: keep utilisation high, but not too high.
  – cpuidle menu: choose progressively lower sleep states.



- Power management is really all about controlling power-related hardware knobs in order to achieve some goal.

- Some of those knobs are... (list knobs).

- These knobs trade performance against power.

- To limit our scope, we're looking at one of these hardware controlled knobs -- DVFS -- but there's no reason that, in the future, this approach couldn't be applied to other knobs which affect power/performance.

- These knobs are normally controlled in naiive ways: in Linux for example, there are two main CPU power management schemes -- ondemand is applies to DVFS. In academic terms, this is based on Mark Weiser's 1994 OSDI paper. That work was good, and applied well to systems at the time, but modern computers don't work in the same way.

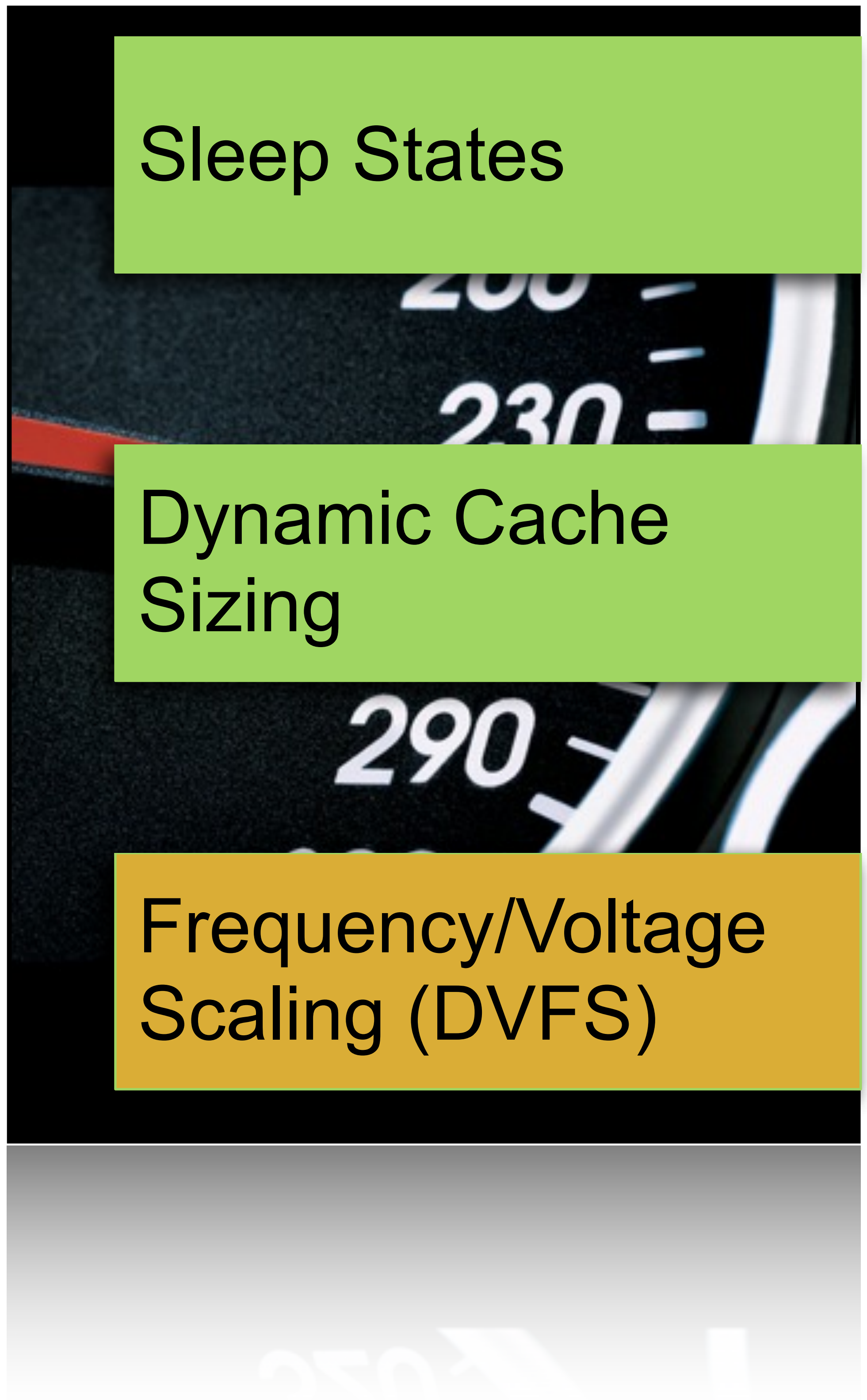
- But there is so much academic research!? Why doesn't it ever get used? Answer: it could be, it just needs to be made practical. The answer is Koala. Koala bridges the gap between the real world and the academic world.

- Why are we using 1994 technology to run computers in 2009? They're simply not the same devices that they were.

Power Management

- Power management:
  - Controlling hardware *knobs*
  - ➡ Performance vs. power

- Power management theory
  - Martin: battery nonlinearities.
  - Optimal scheduling
  - Highly refined speed setting.



Saturday, 4 April 2009

- Power management is really all about controlling power-related hardware knobs in order to achieve some goal.

- Some of those knobs are... (list knobs).

- These knobs trade performance against power.

- To limit our scope, we're looking at one of these hardware controlled knobs -- DVFS -- but there's no reason that, in the future, this approach couldn't be applied to other knobs which affect power/performance.

- These knobs are normally controlled in naiive ways: in Linux for example, there are two main CPU power management schemes -- ondemand is applies to DVFS. In academic terms, this is based on Mark Weiser's 1994 OSDI paper. That work was good, and applied well to systems at the time, but modern computers don't work in the same way.

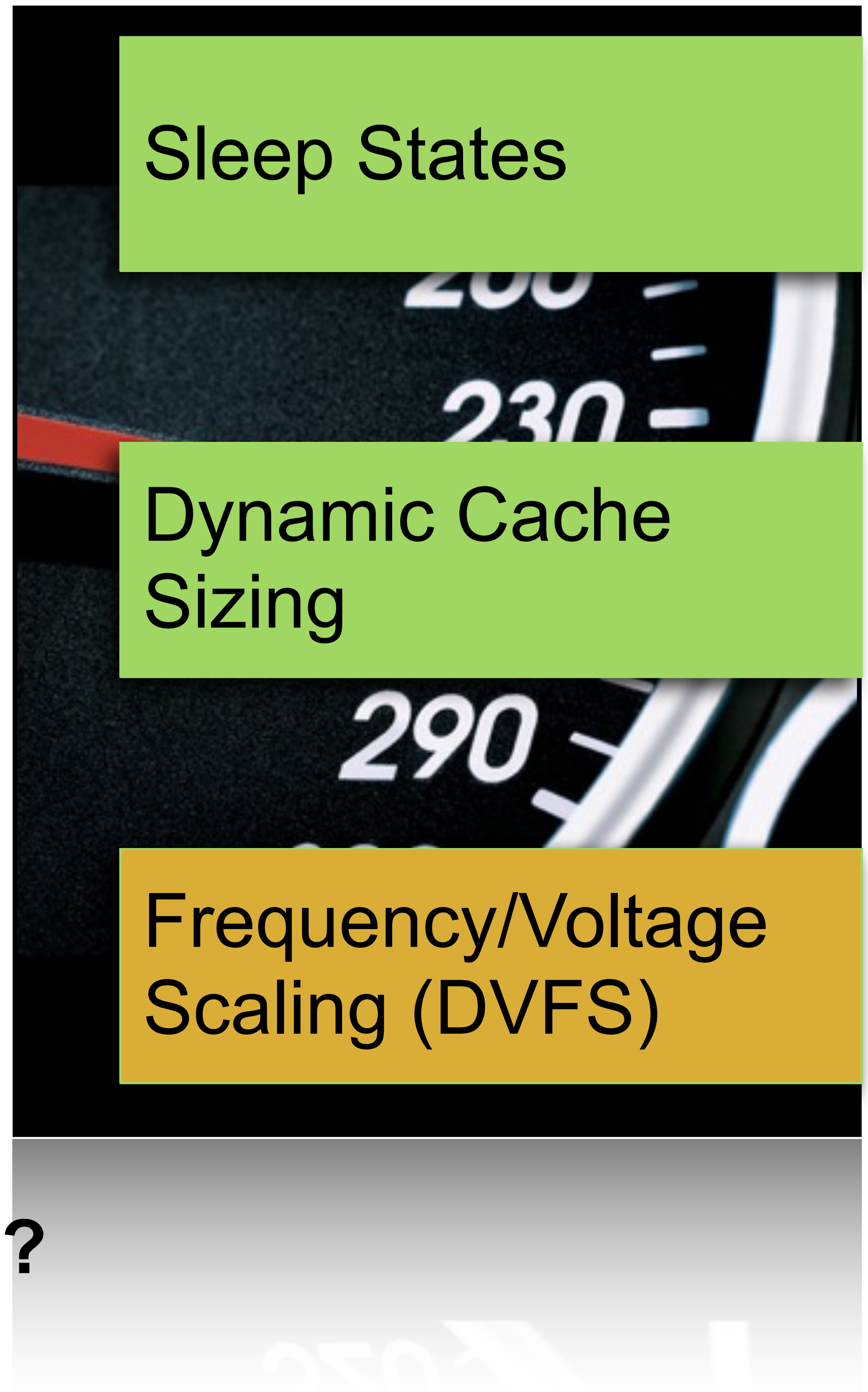
- But there is so much academic research!? Why doesn't it ever get used? Answer: it could be, it just needs to be made practical. The answer is Koala. Koala bridges the gap between the real world and the academic world.

- Why are we using 1994 technology to run computers in 2009? They're simply not the same devices that they were.

Power Management

- Power management:
  – Controlling hardware *knobs*
  ➡ Performance vs. power

- Power management theory
  – Martin: battery nonlinearities.
  – Optimal scheduling
  – Highly refined speed setting.



Why aren't these techniques used?

- Power management is really all about controlling power-related hardware knobs in order to achieve some goal.

- Some of those knobs are... (list knobs).

- These knobs trade performance against power.

- To limit our scope, we're looking at one of these hardware controlled knobs -- DVFS -- but there's no reason that, in the future, this approach couldn't be applied to other knobs which affect power/performance.

- These knobs are normally controlled in naiive ways: in Linux for example, there are two main CPU power management schemes -- ondemand is applies to DVFS. In academic terms, this is based on Mark Weiser's 1994 OSDI paper. That work was good, and applied well to systems at the time, but modern computers don't work in the same way.

- But there is so much academic research!? Why doesn't it ever get used? Answer: it could be, it just needs to be made practical. The answer is Koala. Koala bridges the gap between the real world and the academic world.

- Why are we using 1994 technology to run computers in 2009? They're simply not the same devices that they were.

Dynamic Voltage and Frequency Scaling

NICTA

- Reduce performance, lower the power
- Assumption: constant cycles
- Assumption: $P \propto fV^2$

$$T \propto \frac{1}{f}$$

$$P \propto fV^2$$

$$V_{min} \propto f$$

$$E \propto f^2$$

Saturday, 4 April 2009

* Both real-world, and lots of research, **assume some fairly simple models**.

* These assumptions are good on a gate-level, but **don't work** for complex systems where, on each cycle, the gates perform different tasks. They ignore static power, memory, other effects

* **Koala allows** you to manage modern systems which have more complicated models.

* I'm going to **show a summary** of the experiments we ran to investigate these assumptions. For real detail, see the paper.

Need to be explicit that T is Time, not temperature.

Dynamic Voltage and Frequency Scaling

NICTA

- Reduce performance, lower the power
- Assumption: constant cycles
- Assumption: $P \propto fV^2$

Saturday, 4 April 2009

* Both real-world, and lots of research, **assume some fairly simple models**.

* These assumptions are good on a gate-level, but **don't work** for complex systems where, on each cycle, the gates perform different tasks. They ignore static power, memory, other effects

* **Koala allows** you to manage modern systems which have more complicated models.

* I'm going to **show a summary** of the experiments we ran to investigate these assumptions. For real detail, see the paper.

Need to be explicit that T is Time, not temperature.

Performance

+ Assumed

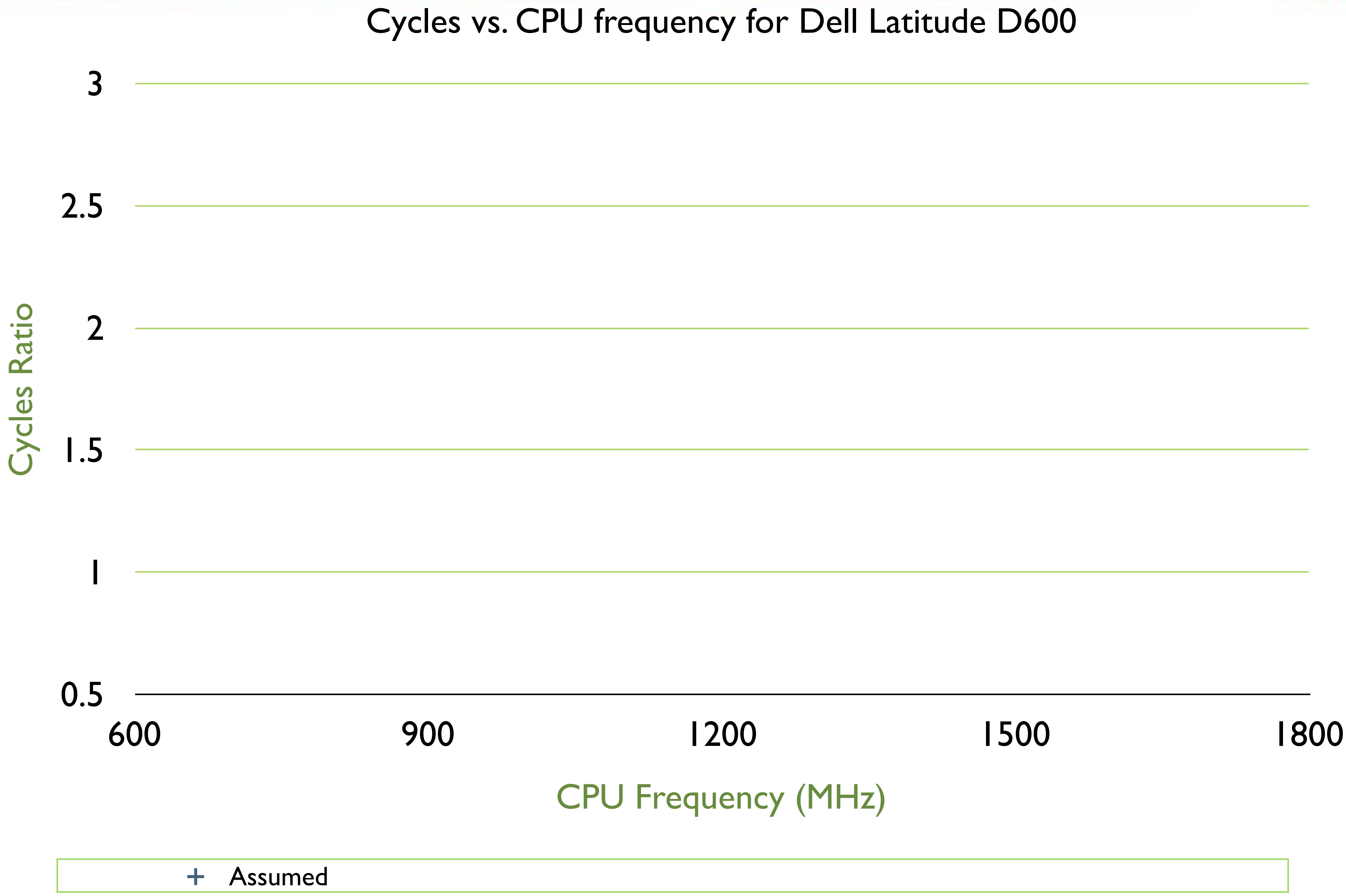
* Lets look at performance. The commonly assumed models suggest that the number of CPU cycles for a workload is constant across frequency changes -- doubling the CPU frequency halves the execution time.

* Looking at a CPU bound benchmark, this is indeed the case! The number of cycles stays nearly constant as the CPU frequency is increased. So far so good.

* But let's look at a memory bound program. The performance of memory isn't improved by increased CPU frequency, so a memory-bound workload doesn't really benefit from increased frequency. Therefore the number of cycles increases as the CPU clock runs faster.

* Those cycles use extra energy, and the CPU voltage must be increased to support the higher frequency.

Performance



Saturday, 4 April 2009

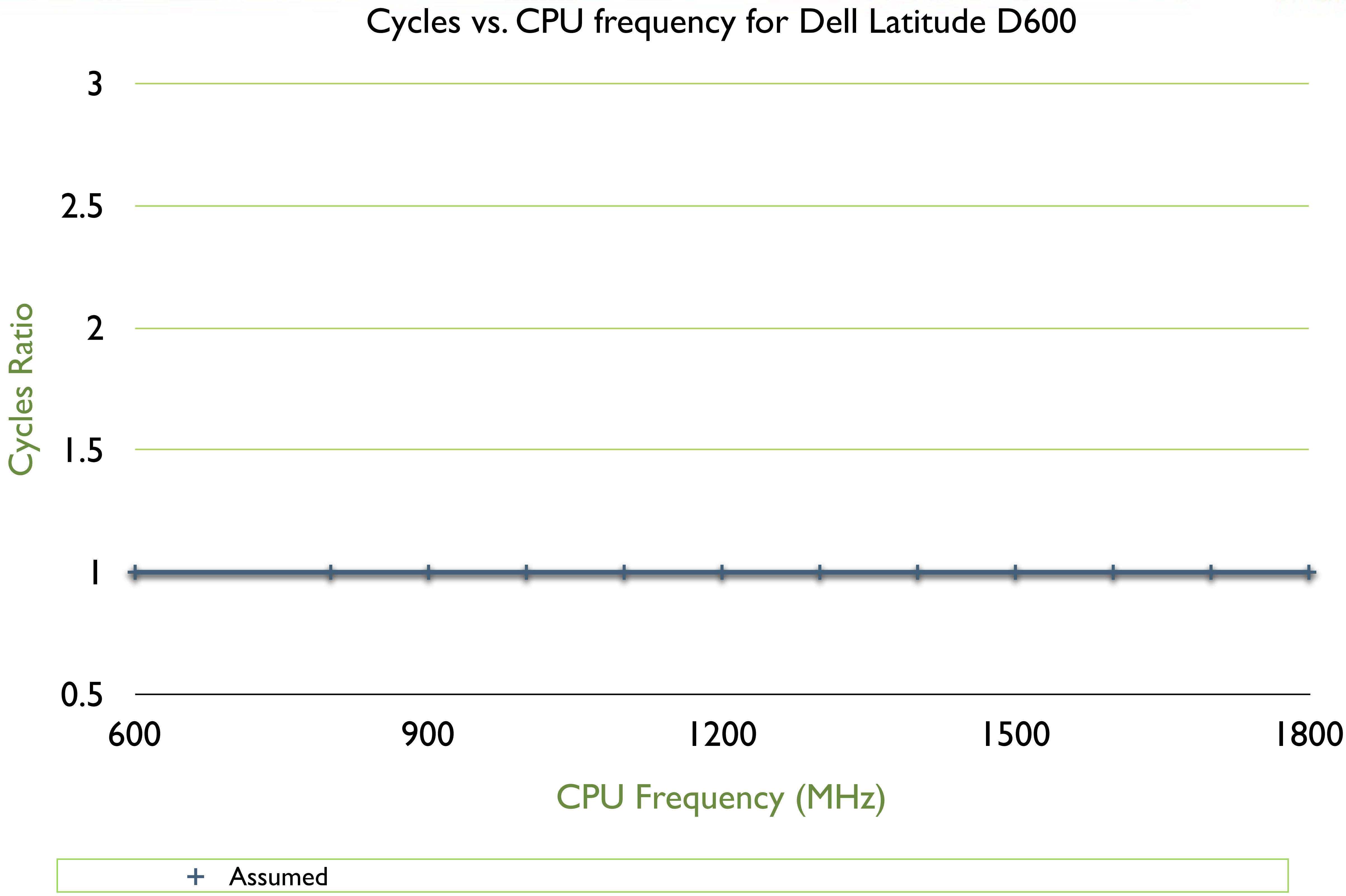
* Lets look at performance. The commonly assumed models suggest that the number of CPU cycles for a workload is constant across frequency changes -- doubling the CPU frequency halves the execution time.

* Looking at a CPU bound benchmark, this is indeed the case! The number of cycles stays nearly constant as the CPU frequency is increased. So far so good.

* But let's look at a memory bound program. The performance of memory isn't improved by increased CPU frequency, so a memory-bound workload doesn't really benefit from increased frequency. Therefore the number of cycles increases as the CPU clock runs faster.

* Those cycles use extra energy, and the CPU voltage must be increased to support the higher frequency.

Performance



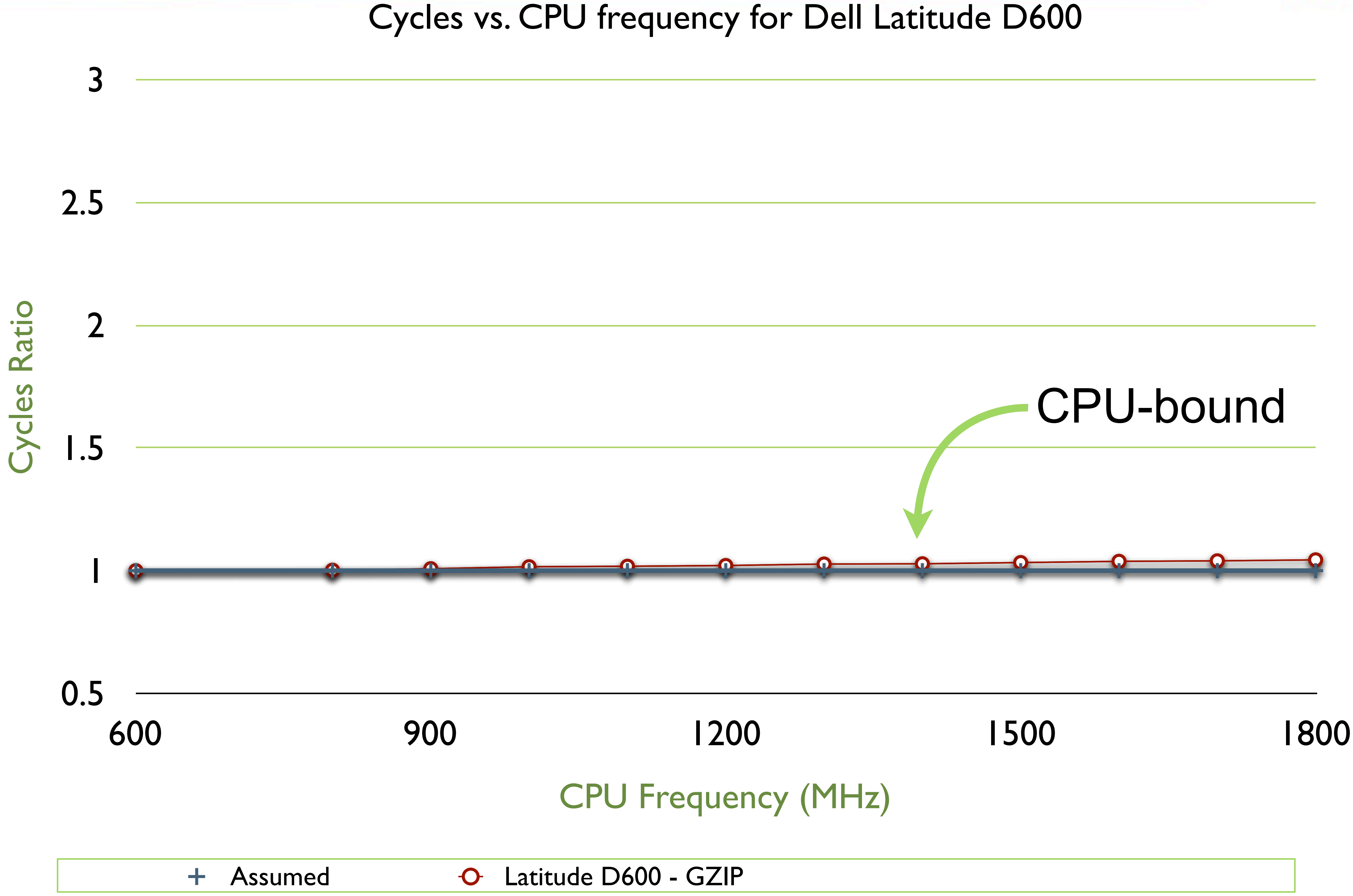
* Lets look at performance. The commonly assumed models suggest that the number of CPU cycles for a workload is constant across frequency changes -- doubling the CPU frequency halves the execution time.

* Looking at a CPU bound benchmark, this is indeed the case! The number of cycles stays nearly constant as the CPU frequency is increased. So far so good.

* But let's look at a memory bound program. The performance of memory isn't improved by increased CPU frequency, so a memory-bound workload doesn't really benefit from increased frequency. Therefore the number of cycles increases as the CPU clock runs faster.

* Those cycles use extra energy, and the CPU voltage must be increased to support the higher frequency.

Performance



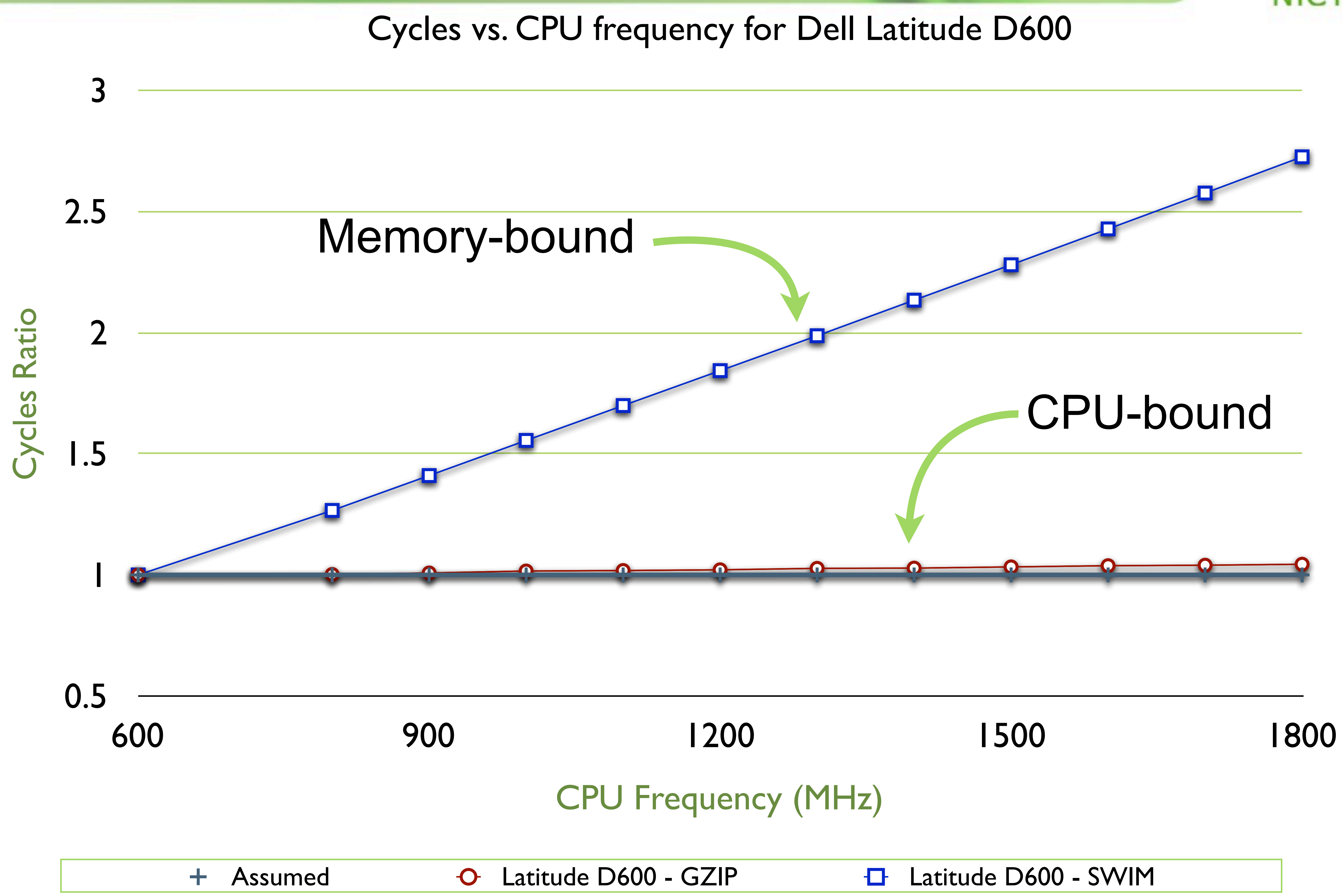
* Lets look at performance. The commonly assumed models suggest that the number of CPU cycles for a workload is constant across frequency changes -- doubling the CPU frequency halves the execution time.

* Looking at a CPU bound benchmark, this is indeed the case! The number of cycles stays nearly constant as the CPU frequency is increased. So far so good.

* But let's look at a memory bound program. The performance of memory isn't improved by increased CPU frequency, so a memory-bound workload doesn't really benefit from increased frequency. Therefore the number of cycles increases as the CPU clock runs faster.

* Those cycles use extra energy, and the CPU voltage must be increased to support the higher frequency.

Performance



Saturday, 4 April 2009

* Lets look at performance. The commonly assumed models suggest that the number of CPU cycles for a workload is constant across frequency changes -- doubling the CPU frequency halves the execution time.

* Looking at a CPU bound benchmark, this is indeed the case! The number of cycles stays nearly constant as the CPU frequency is increased. So far so good.

* But let's look at a memory bound program. The performance of memory isn't improved by increased CPU frequency, so a memory-bound workload doesn't really benefit from increased frequency. Therefore the number of cycles increases as the CPU clock runs faster.

* Those cycles use extra energy, and the CPU voltage must be increased to support the higher frequency.

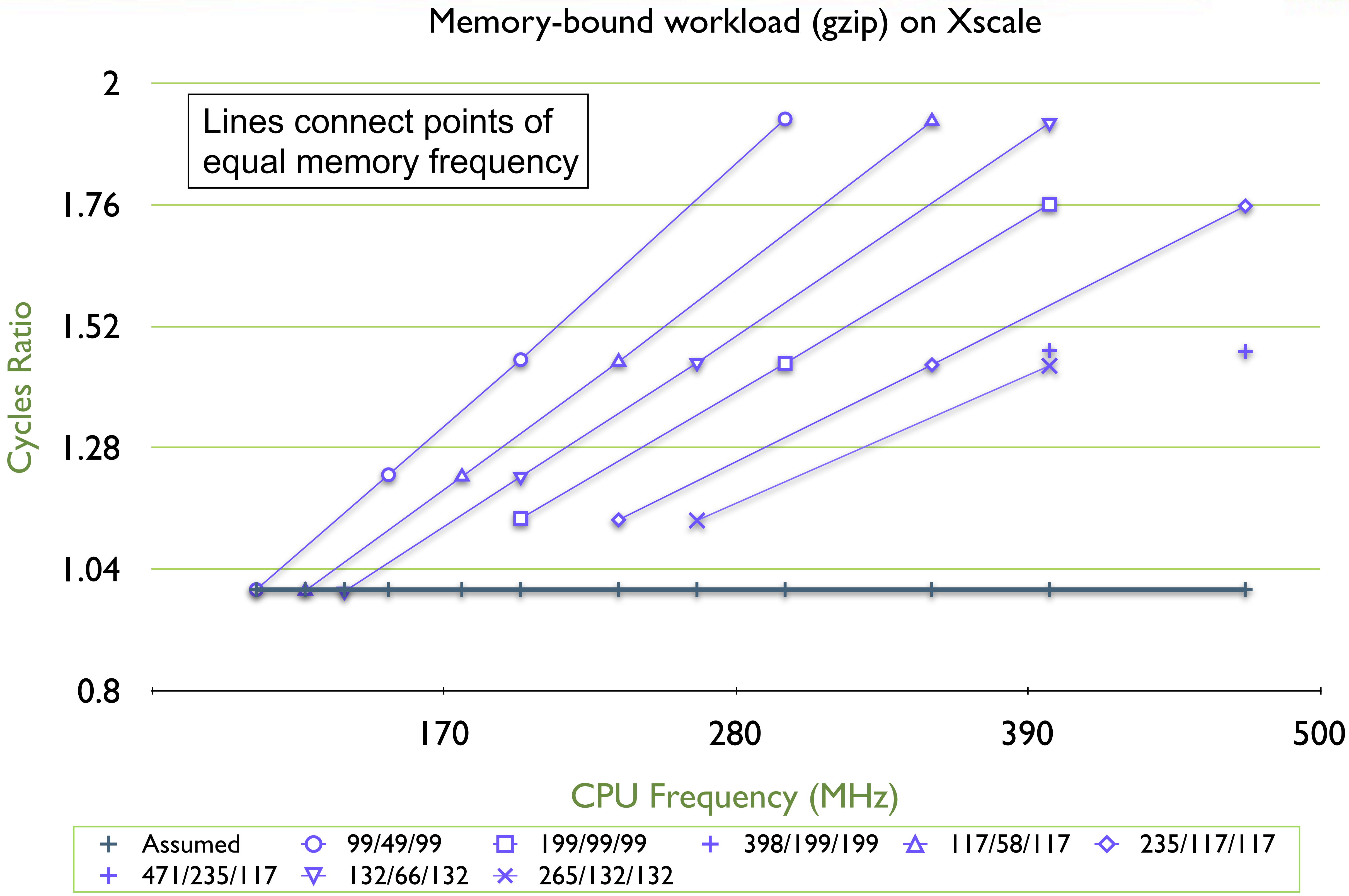
Multiple Frequencies

NICTA

+ Assumed
○ 99/49/99
□ 199/99/99
+ 398/199/199
△ 117/58/117
◇ 235/117/117
+ 471/235/117
▽ 132/66/132
× 265/132/132

Things get even more complicated when we start modifying the memory frequency -- on this XScale based platform, we can't easily modify the CPU frequency without modifying the memory and bus frequency.

Multiple Frequencies



Saturday, 4 April 2009

Things get even more complicated when we start modifying the memory frequency -- on this XScale based platform, we can't easily modify the CPU frequency without modifying the memory and bus frequency.

Energy

NICTA

+ Assumed   swim   gzip

* Now let’s look at energy. The simplest models would suggest a quadratic relationship between energy consumption and energy use -- the lowest frequency is always the most energy efficient.

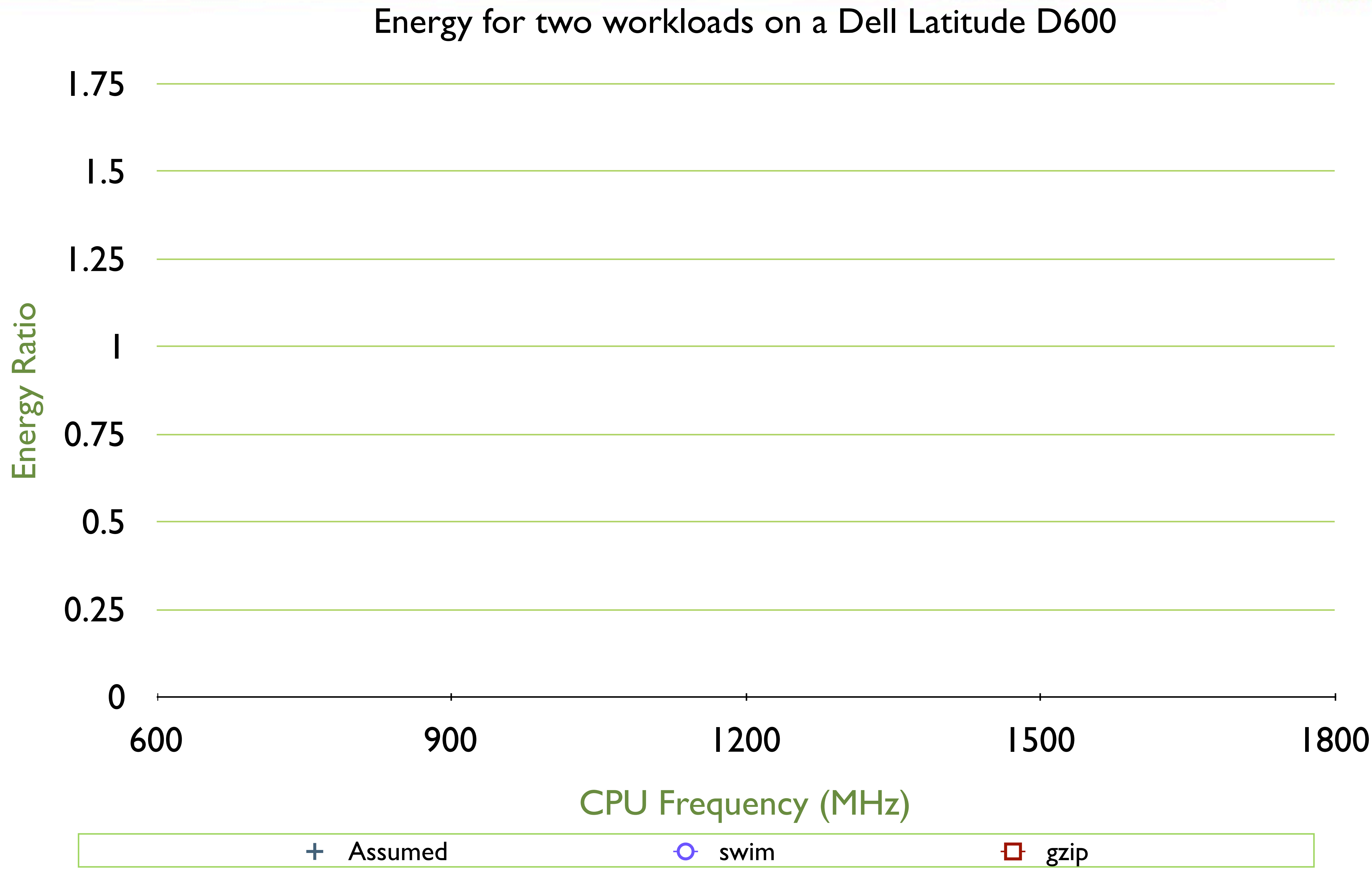
* For the memory bound benchmark, where the execution time remains nearly constant, this is the case! The workload takes more energy as the CPU frequency increases, although not nearly so much as the assumed model suggests.

* But if we look at the CPU bound benchmark, the energy used is **reduced** when we increase the frequency! What’s going on? How can we be so wrong?

* Well... While the power for both benchmarks is definitely increased at higher frequencies, the CPU-bound benchmark runs for a much shorter time at the higher frequecies. Since the CPU-bound benchmark runs for a much shorter amount of time, it uses less energy over-all.

Energy

NICTA



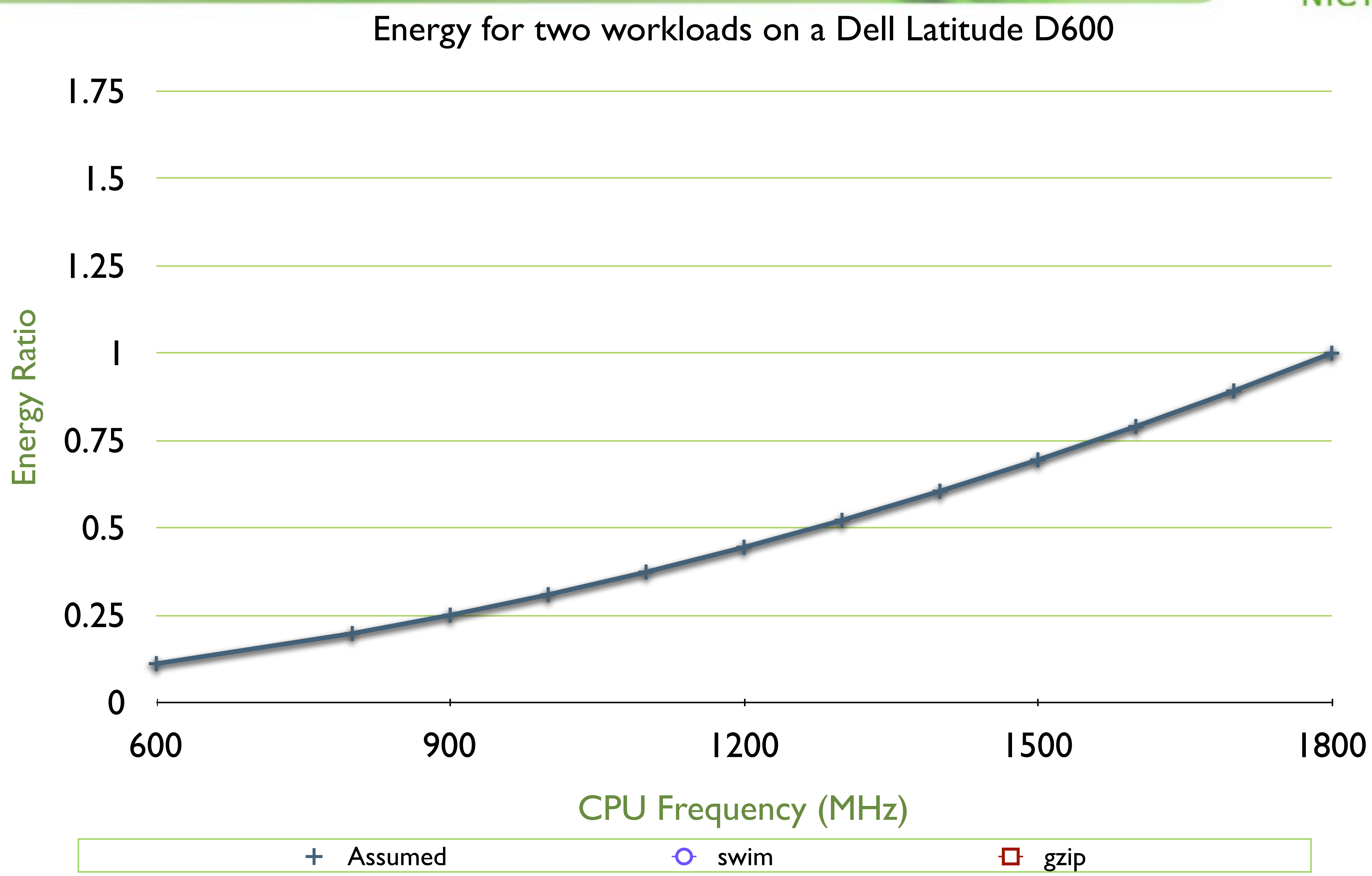
* Now let's look at energy. The simplest models would suggest a quadratic relationship between energy consumption and energy use -- the lowest frequency is always the most energy efficient.

* For the memory bound benchmark, where the execution time remains nearly constant, this is the case! The workload takes more energy as the CPU frequency increases, although not nearly so much as the assumed model suggests.

* But if we look at the CPU bound benchmark, the energy used is **reduced** when we increase the frequency! What's going on? How can we be so wrong?

* Well... While the power for both benchmarks is definitely increased at higher frequencies, the CPU-bound benchmark runs for a much shorter time at the higher frequecies. Since the CPU-bound benchmark runs for a much shorter amount of time, it uses less energy over-all.

Energy



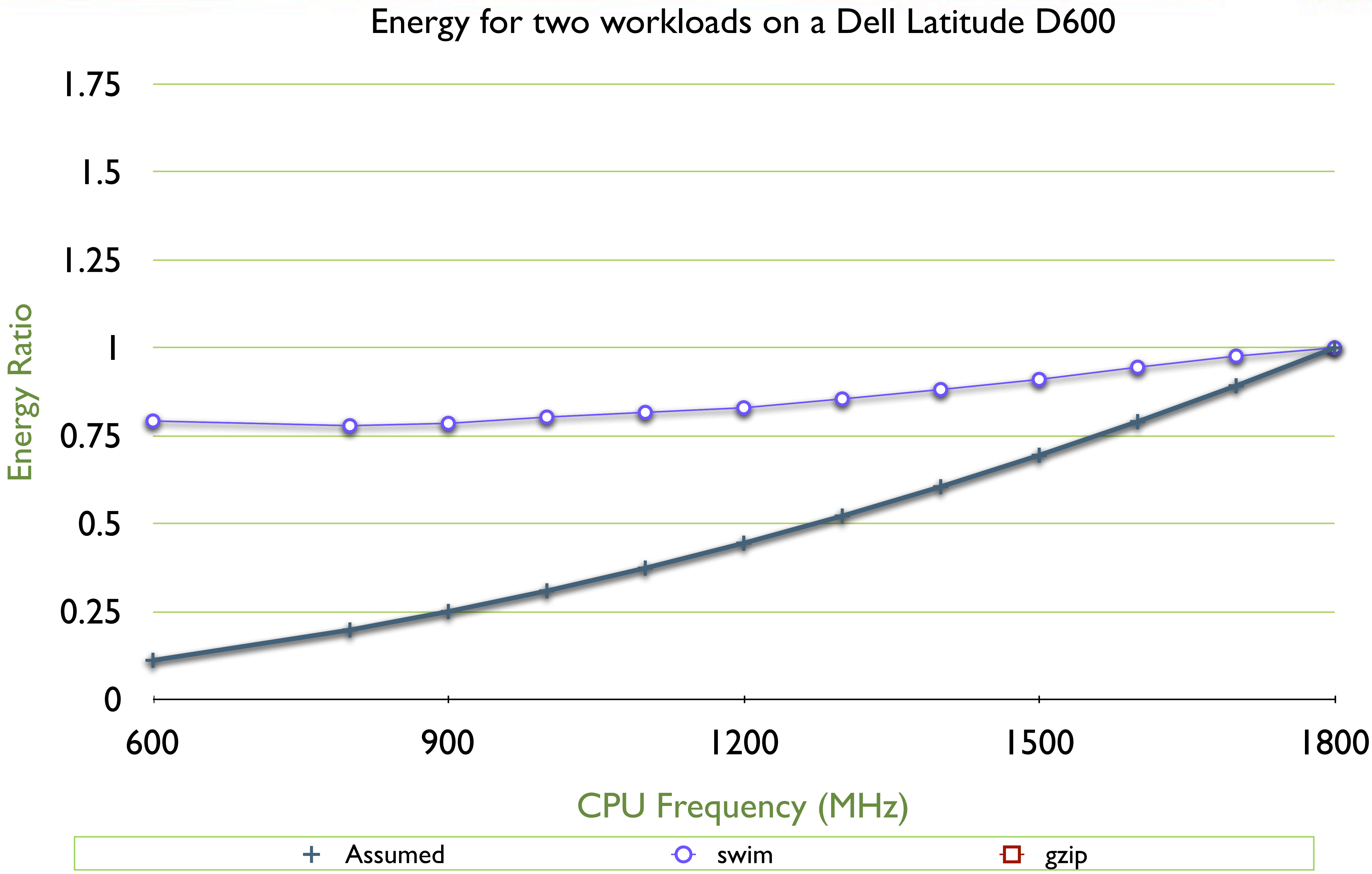
* Now let’s look at energy. The simplest models would suggest a quadratic relationship between energy consumption and energy use -- the lowest frequency is always the most energy efficient.

* For the memory bound benchmark, where the execution time remains nearly constant, this is the case! The workload takes more energy as the CPU frequency increases, although not nearly so much as the assumed model suggests.

* But if we look at the CPU bound benchmark, the energy used is **reduced** when we increase the frequency! What’s going on? How can we be so wrong?

* Well... While the power for both benchmarks is definitely increased at higher frequencies, the CPU-bound benchmark runs for a much shorter time at the higher frequecies. Since the CPU-bound benchmark runs for a much shorter amount of time, it uses less energy over-all.

Energy



Saturday, 4 April 2009

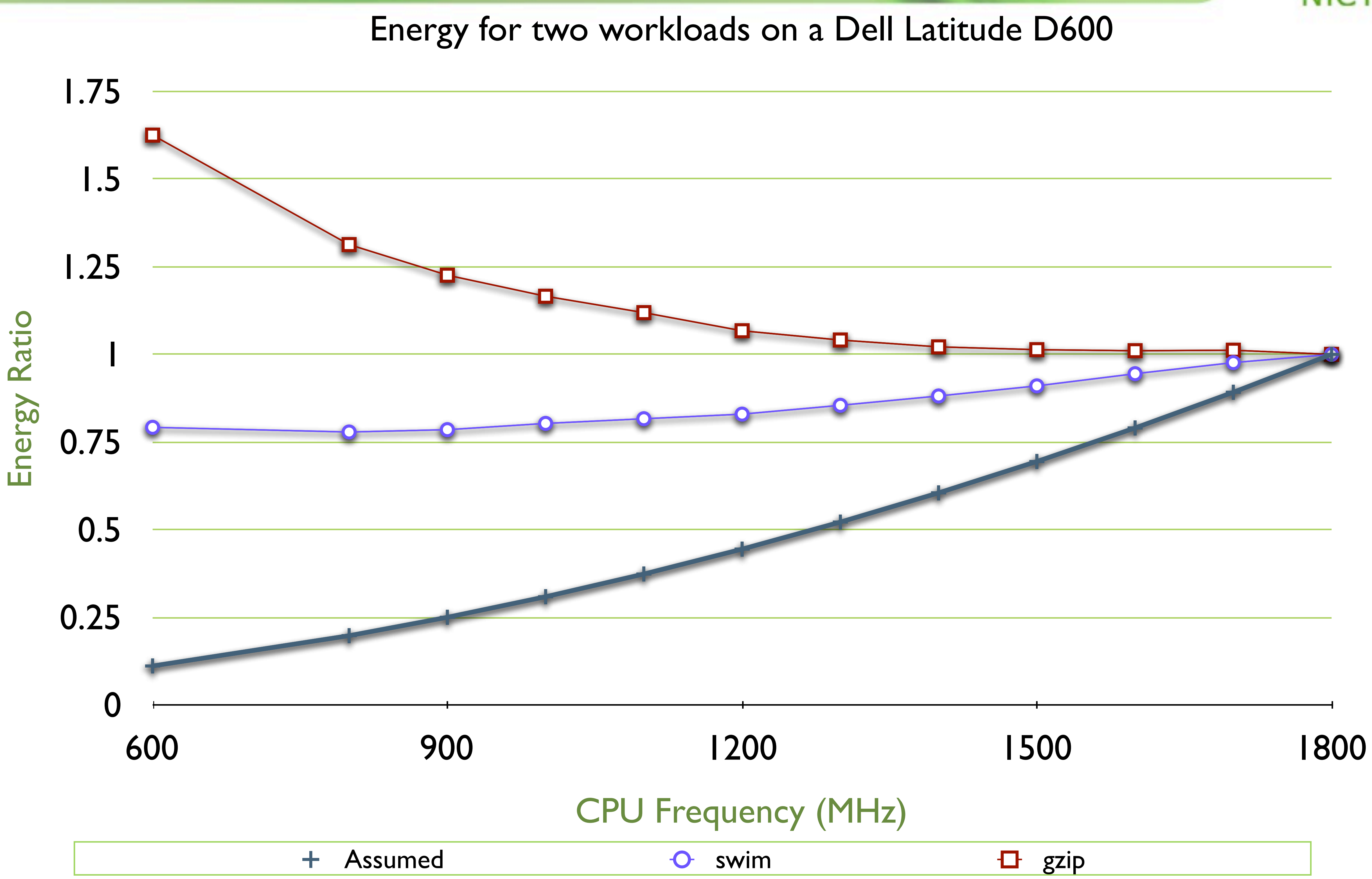
* Now let’s look at energy. The simplest models would suggest a quadratic relationship between energy consumption and energy use -- the lowest frequency is always the most energy efficient.

* For the memory bound benchmark, where the execution time remains nearly constant, this is the case! The workload takes more energy as the CPU frequency increases, although not nearly so much as the assumed model suggests.

* But if we look at the CPU bound benchmark, the energy used is **reduced** when we increase the frequency! What’s going on? How can we be so wrong?

* Well... While the power for both benchmarks is definitely increased at higher frequencies, the CPU-bound benchmark runs for a much shorter time at the higher frequecies. Since the CPU-bound benchmark runs for a much shorter amount of time, it uses less energy over-all.

Energy



Saturday, 4 April 2009

* Now let's look at energy. The simplest models would suggest a quadratic relationship between energy consumption and energy use -- the lowest frequency is always the most energy efficient.

* For the memory bound benchmark, where the execution time remains nearly constant, this is the case! The workload takes more energy as the CPU frequency increases, although not nearly so much as the assumed model suggests.

* But if we look at the CPU bound benchmark, the energy used is **reduced** when we increase the frequency! What's going on? How can we be so wrong?

* Well... While the power for both benchmarks is definitely increased at higher frequencies, the CPU-bound benchmark runs for a much shorter time at the higher freqecies. Since the CPU-bound benchmark runs for a much shorter amount of time, it uses less energy over-all.

Sleep States

NICTA

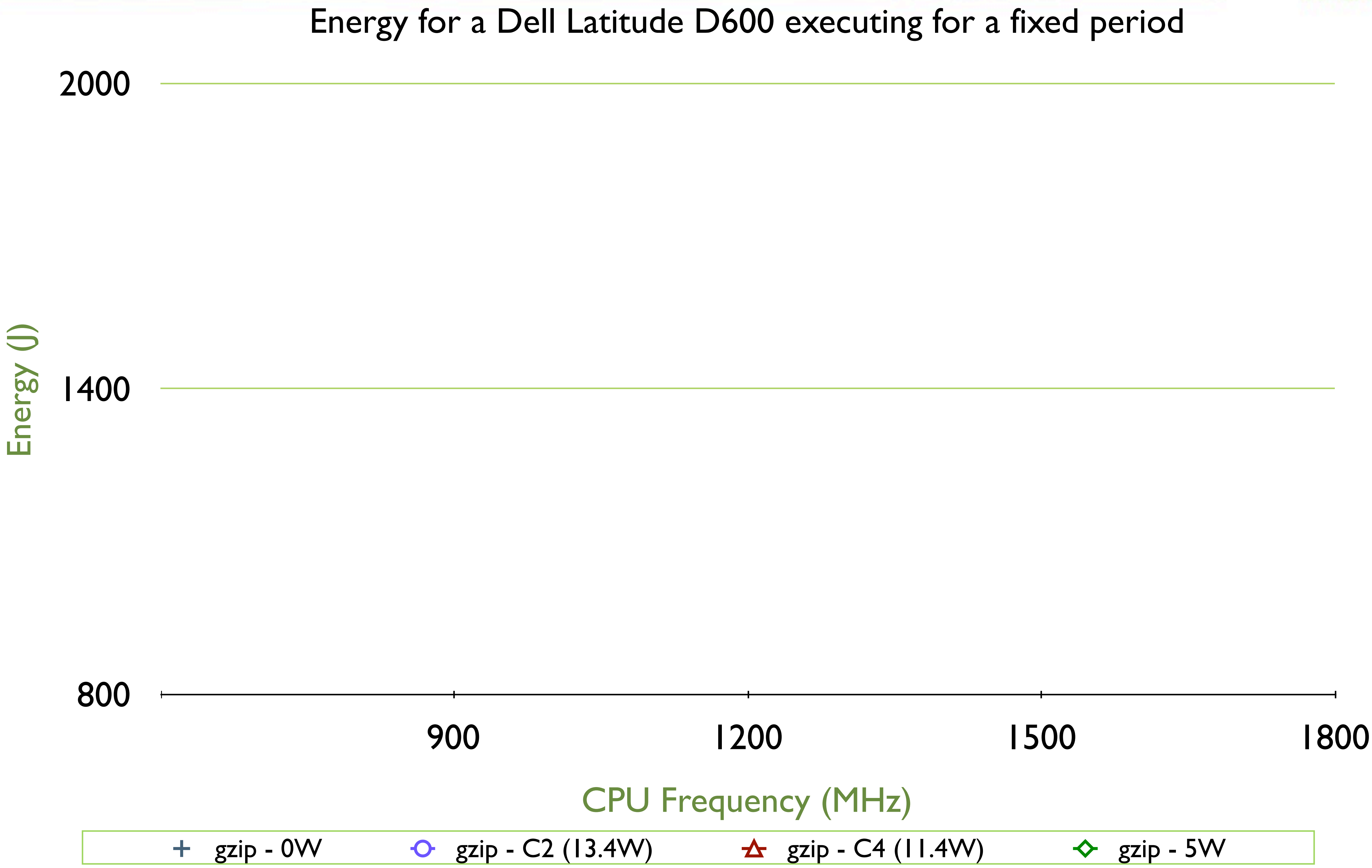
gzip - 0W   gzip - C2 (13.4W)   gzip - C4 (11.4W)   gzip - 5W

This assumes that we either use the extra time created by running fast, or we shut the system down. But what if we don't have anything useful to do with that extra time?

The system goes idle... And there are different idle modes. This graph shows what would happen for four different idle states when we execute a particular benchmark (gzip -- CPU bound).

Note that the lowest energy frequency to run at is dependent on which sleep state we'll enter. If we're going into a higher-power state, we should run at the lowest frequency, and if we're going to end up in a low-power state, we need to run at a high frequency.

Sleep States

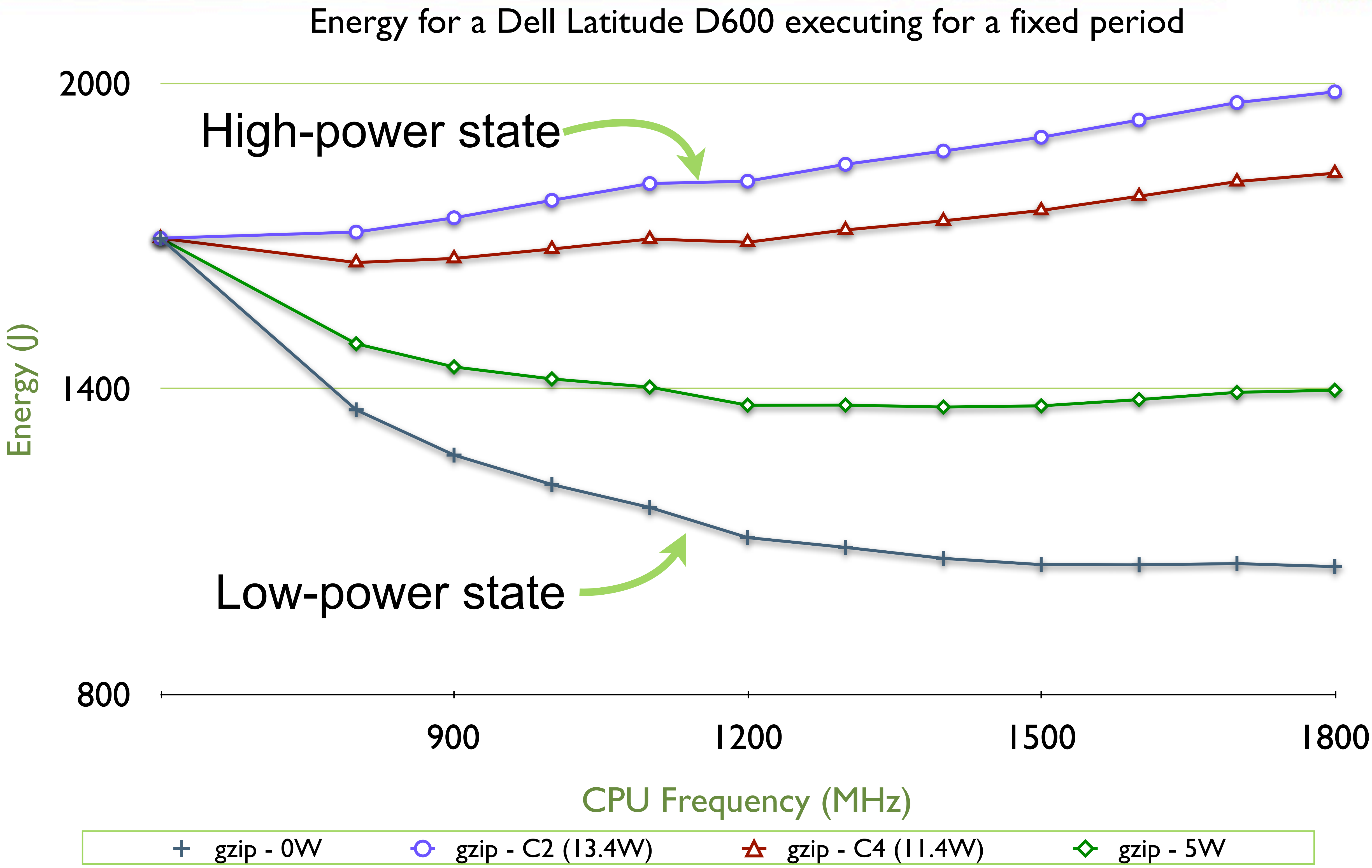


This assumes that we either use the extra time created by running fast, or we shut the system down. But what if we don't have anything useful to do with that extra time?

The system goes idle... And there are different idle modes. This graph shows what would happen for four different idle states when we execute a particular benchmark (gzip -- CPU bound).

Note that the lowest energy frequency to run at is dependent on which sleep state we'll enter. If we're going into a higher-power state, we should run at the lowest frequency, and if we're going to end up in a low-power state, we need to run at a high frequency.

Sleep States



Saturday, 4 April 2009

This assumes that we either use the extra time created by running fast, or we shut the system down. But what if we don't have anything useful to do with that extra time?

The system goes idle... And there are different idle modes. This graph shows what would happen for four different idle states when we execute a particular benchmark (gzip -- CPU bound).

Note that the lowest energy frequency to run at is dependent on which sleep state we'll enter. If we're going into a higher-power state, we should run at the lowest frequency, and if we're going to end up in a low-power state, we need to run at a high frequency.

Temperature

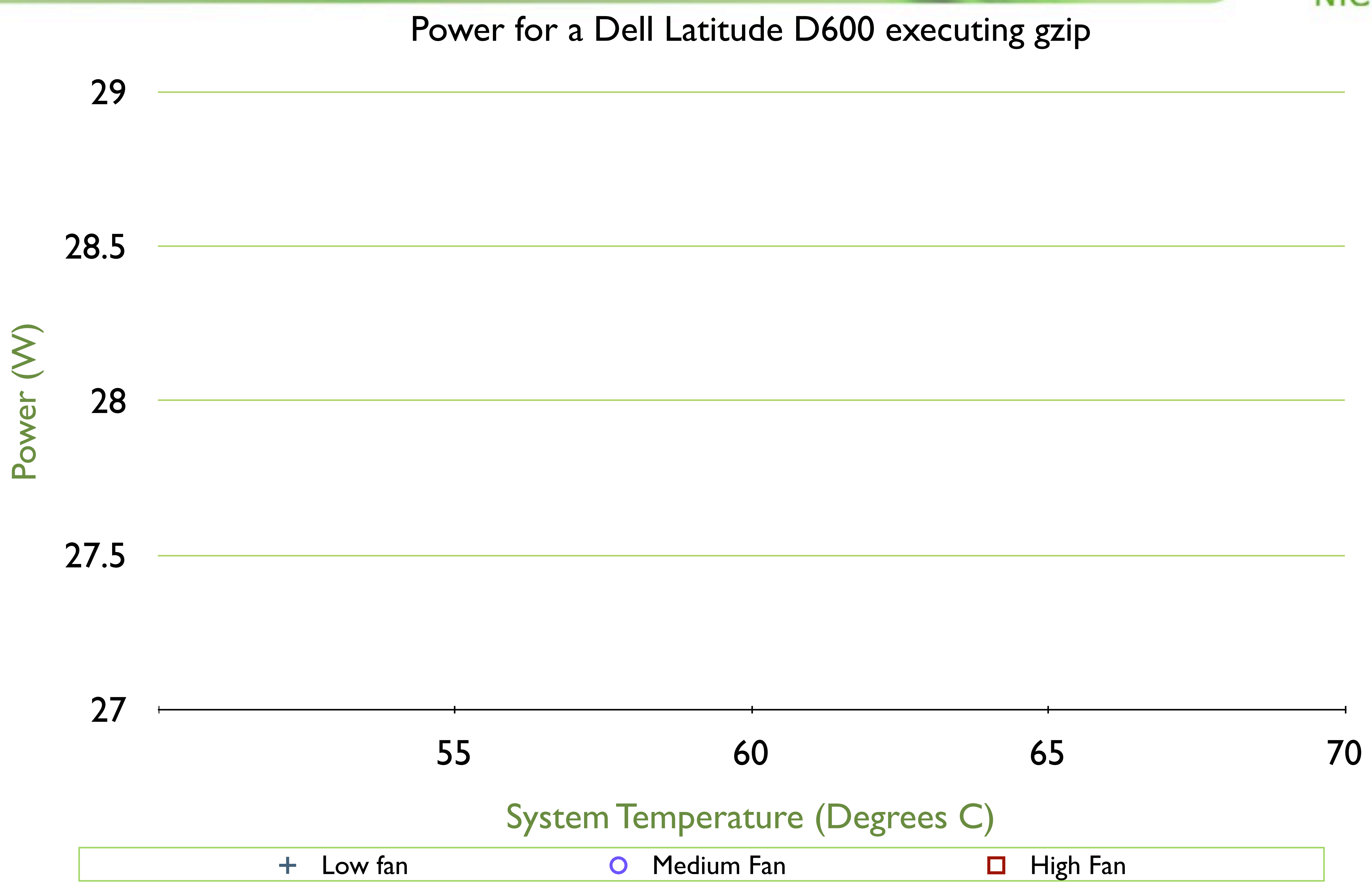
NICTA

+ Low fan   ○ Medium Fan   □ High Fan

There are lots more of these DVFS “gotchas” discussed in the paper. The DVFS behaviour of real systems just doesn’t fit the model. We looked at the effect of temperature and CPU fan speed... As the system warms up, it uses more and more power...

And when the fan kicks in to cool the system it uses even more power.

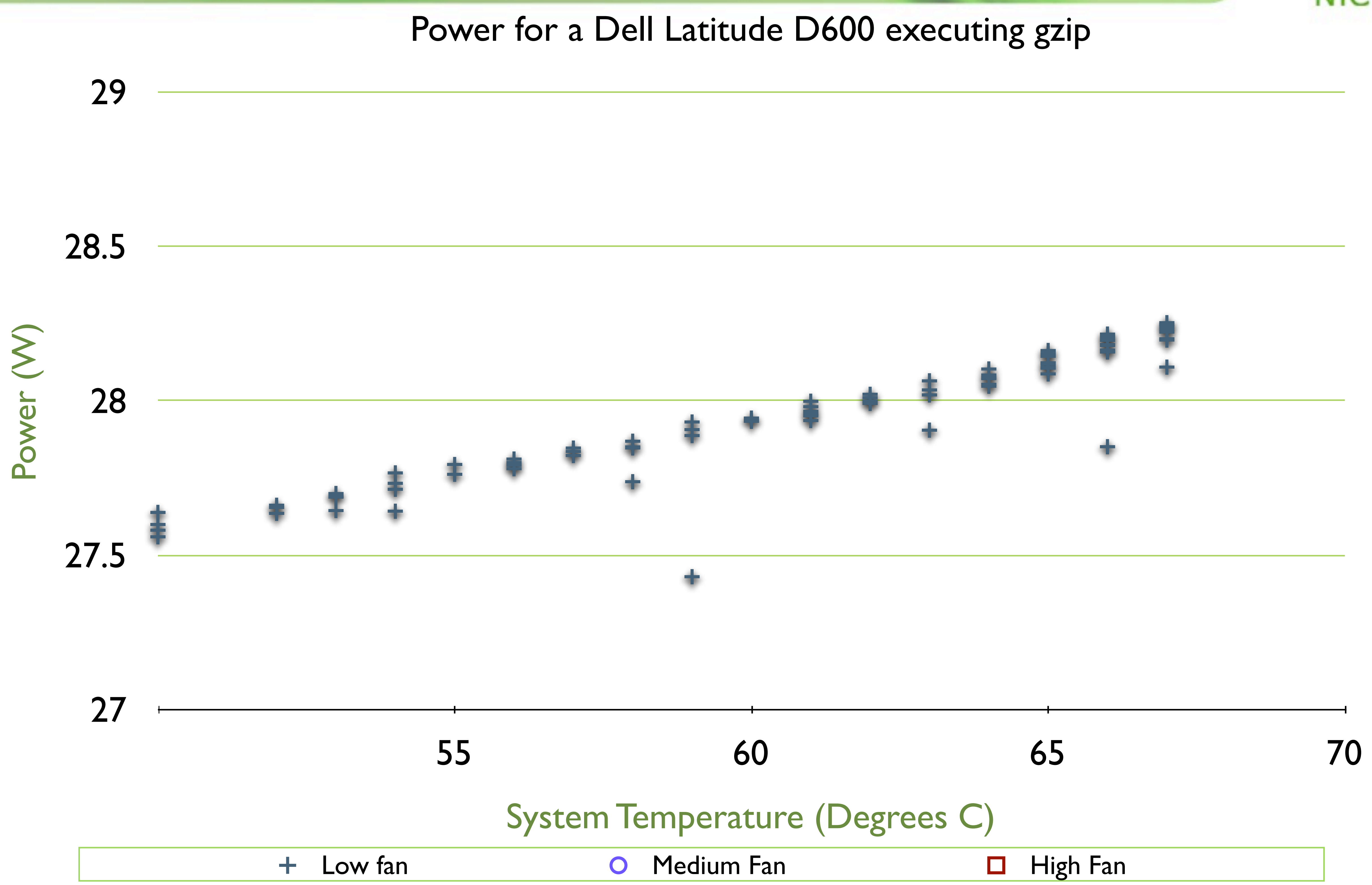
Temperature



There are lots more of these DVFS “gotchas” discussed in the paper. The DVFS behaviour of real systems just doesn’t fit the model. We looked at the effect of temperature and CPU fan speed... As the system warms up, it uses more and more power...

And when the fan kicks in to cool the system it uses even more power.

Temperature

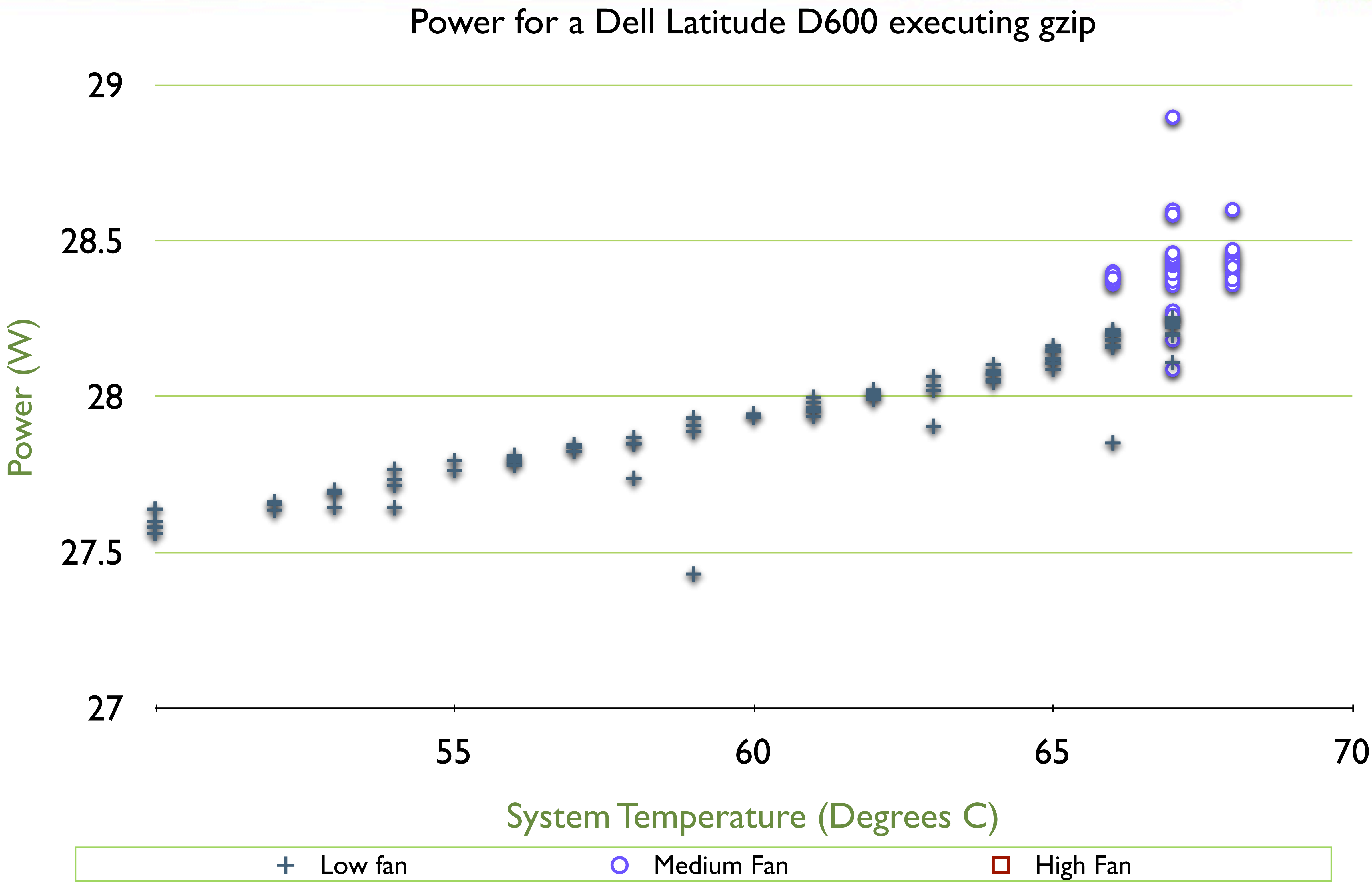


Saturday, 4 April 2009

There are lots more of these DVFS “gotchas” discussed in the paper. The DVFS behaviour of real systems just doesn’t fit the model. We looked at the effect of temperature and CPU fan speed... As the system warms up, it uses more and more power...

And when the fan kicks in to cool the system it uses even more power.

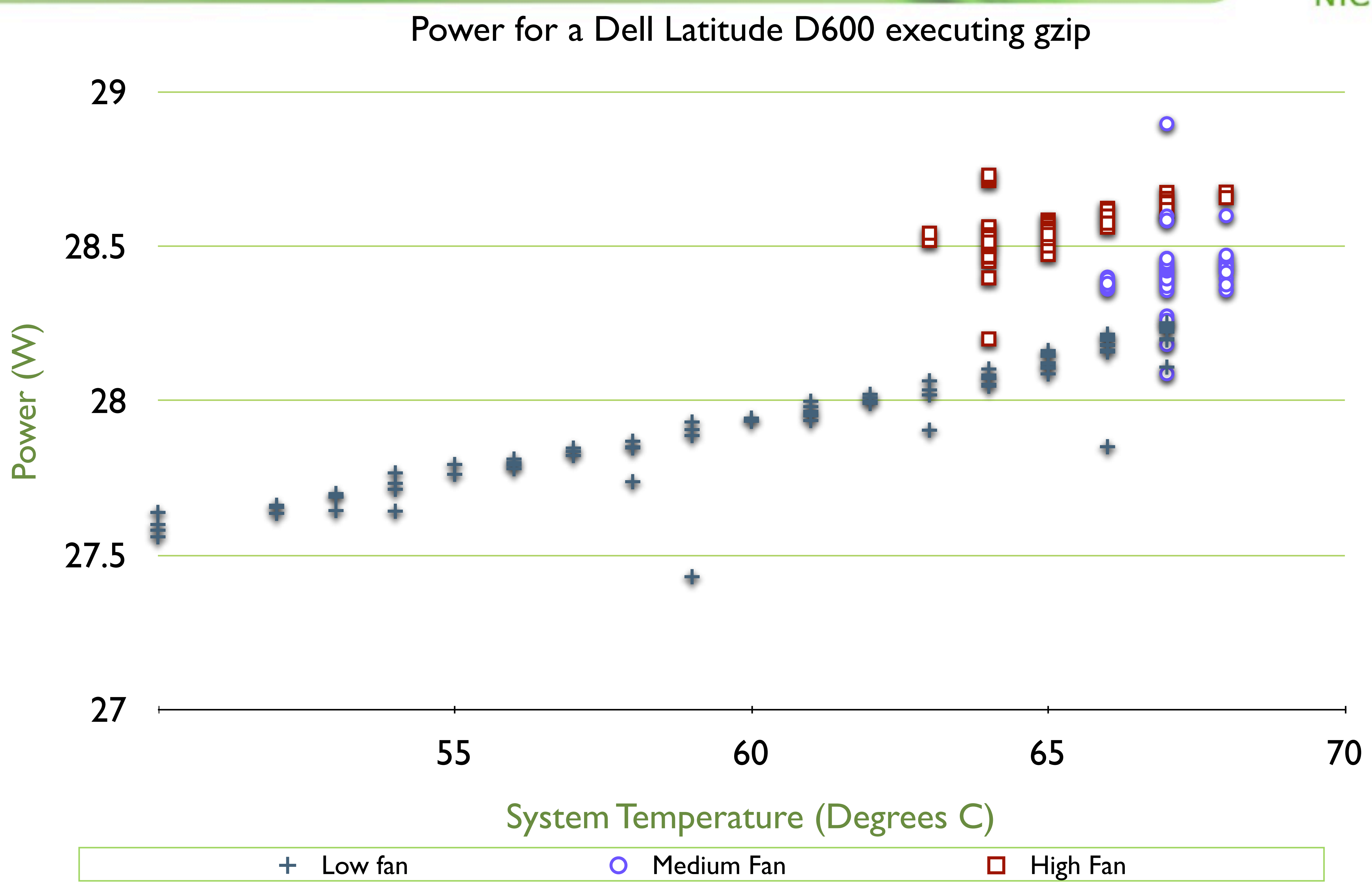
Temperature



There are lots more of these DVFS “gotchas” discussed in the paper. The DVFS behaviour of real systems just doesn’t fit the model. We looked at the effect of temperature and CPU fan speed... As the system warms up, it uses more and more power...

And when the fan kicks in to cool the system it uses even more power.

Temperature



Saturday, 4 April 2009

There are lots more of these DVFS "gotchas" discussed in the paper. The DVFS behaviour of real systems just doesn't fit the model. We looked at the effect of temperature and CPU fan speed... As the system warms up, it uses more and more power...

And when the fan kicks in to cool the system it uses even more power.

Voltage Regulator Efficiency

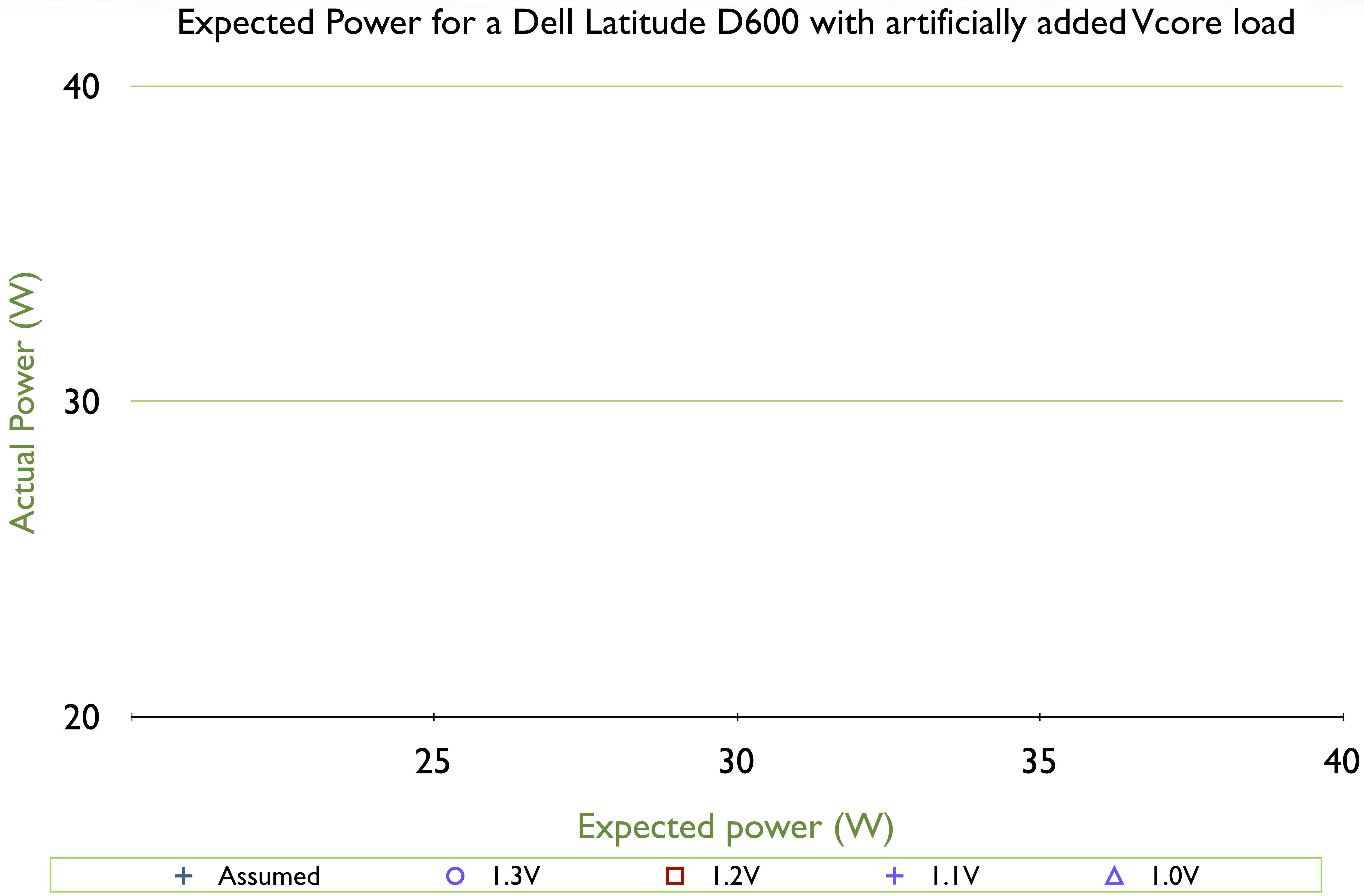
NICTA

+ Assumed   ○ 1.3V   □ 1.2V   + 1.1V   △ 1.0V

Another one that really had us flummoxed for a while was the efficiency of the system's voltage regulators. In the Dell Latitude D600, the main core regulator's efficiency is highly dependent on the amount of power running through it as well as the input voltage.

It meant that, at a particular temperature, the system power actually _increased_ when changing down from 1300MHz to 1200MHz.

Voltage Regulator Efficiency

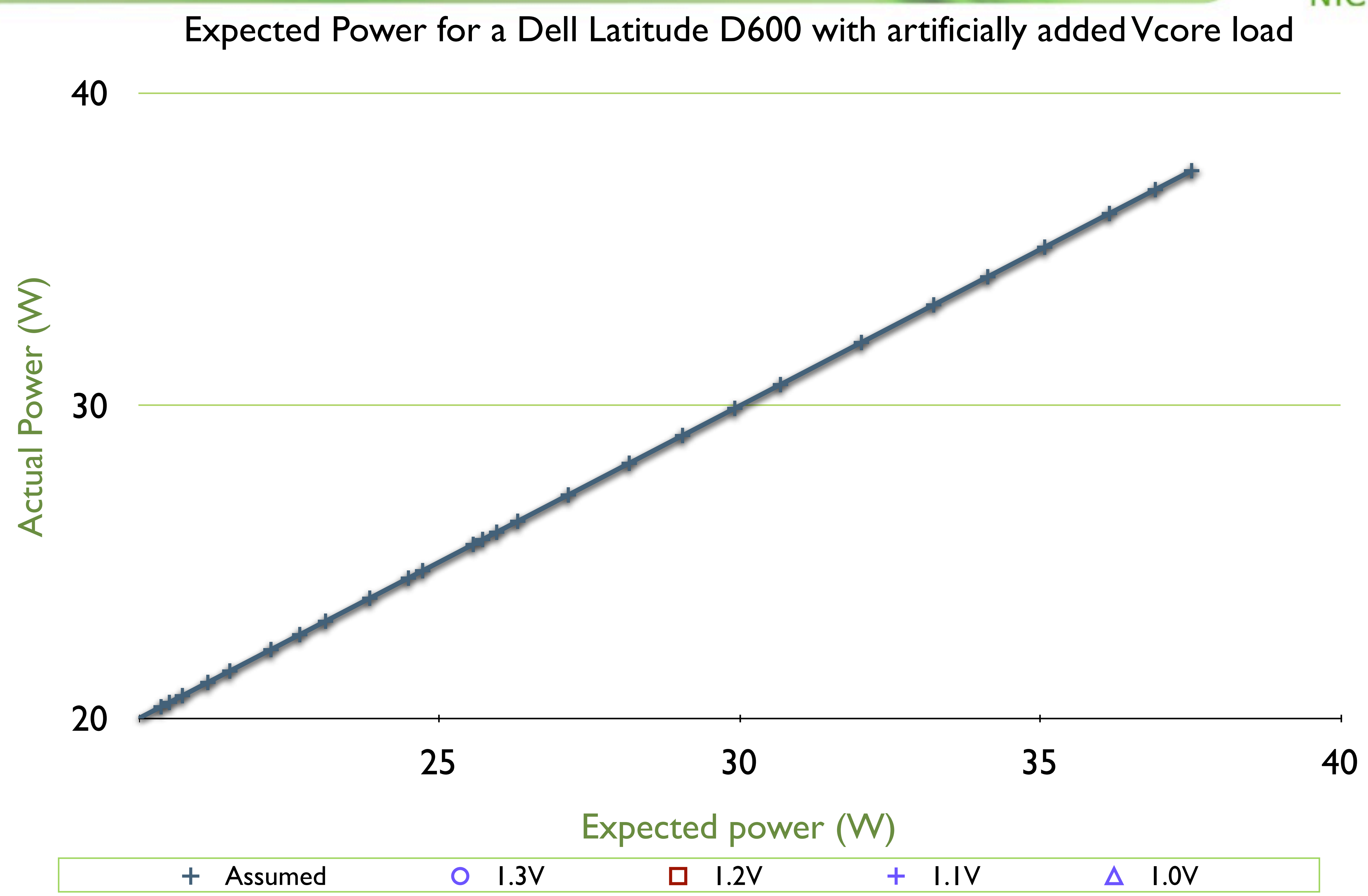


Saturday, 4 April 2009

Another one that really had us flummoxed for a while was the efficiency of the system’s voltage regulators. In the Dell Latitude D600, the main core regulator’s efficiency is highly dependent on the amount of power running through it as well as the input voltage.

It meant that, at a particular temperature, the system power actually _increased_ when changing down from 1300MHz to 1200MHz.

Voltage Regulator Efficiency

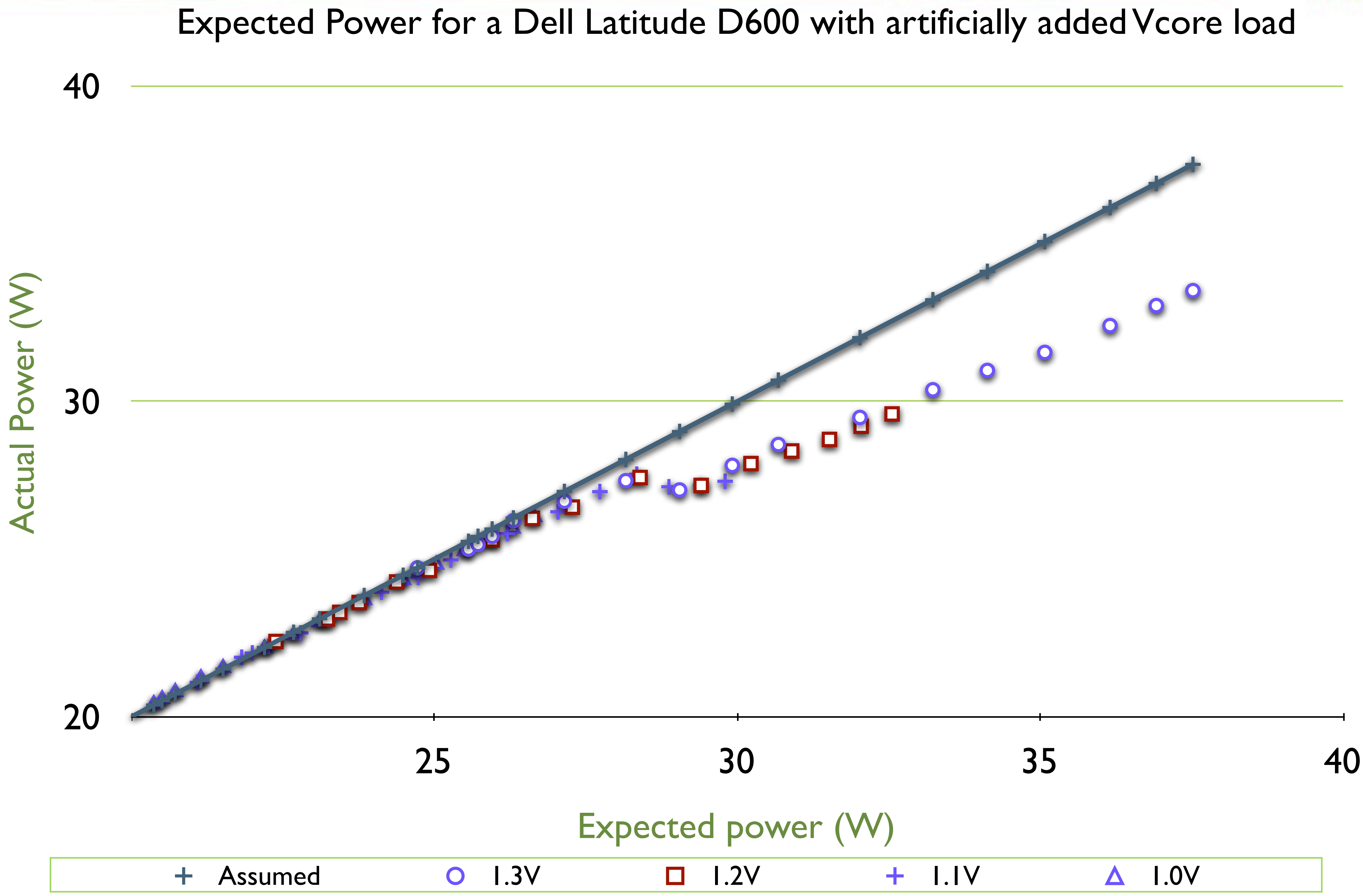


Another one that really had us flummoxed for a while was the efficiency of the system's voltage regulators. In the Dell Latitude D600, the main core regulator's efficiency is highly dependent on the amount of power running through it as well as the input voltage.

It meant that, at a particular temperature, the system power actually _increased_ when changing down from 1300MHz to 1200MHz.

Voltage Regulator Efficiency

NICTA



Another one that really had us flummoxed for a while was the efficiency of the system's voltage regulators. In the Dell Latitude D600, the main core regulator's efficiency is highly dependent on the amount of power running through it as well as the input voltage.

It meant that, at a particular temperature, the system power actually _increased_ when changing down from 1300MHz to 1200MHz.

And so many more...

- Temperature
- Fan power
- Power supply efficiency
- Memory performance variation
- Real-time dependencies
- Frequency switch overheads
- Changing hardware configurations
- Manufacturing variation

There are lots of other quirks discussed in the paper. It means that the traditional assumptions can actually cause power management schemes to use **more energy**, not less. This will become increasingly true as we see more and more hardware power management features.

Hardware platforms behave differently to each other and workloads behave differently to each other on them. You need to build a model that reflects the actual hardware you're dealing with, and you need to scale workloads independently. And it's not good enough to set the settings system wide -- in a multi-tasking workload, serious gains can be made by customising the system settings for individual workloads. To do this, we need a more realistic model.

And so many more...

- Temperature
- Fan power
- Power supply efficiency
- Memory performance variation
- Real-time dependencies
- Frequency switch overheads
- Changing hardware configurations
- Manufacturing variation

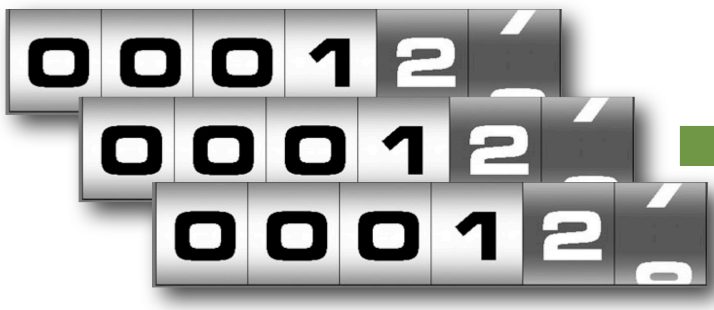
We need a realistic model!

There are lots of other quirks discussed in the paper. It means that the traditional assumptions can actually cause power management schemes to use **more energy**, not less. This will become increasingly true as we see more and more hardware power management features.

Hardware platforms behave differently to each other and workloads behave differently to each other on them. You need to build a model that reflects the actual hardware you're dealing with, and you need to scale workloads independently. And it's not good enough to set the settings system wide -- in a multi-tasking workload, serious gains can be made by customising the system settings for individual workloads. To do this, we need a more realistic model.

Koala overview

α



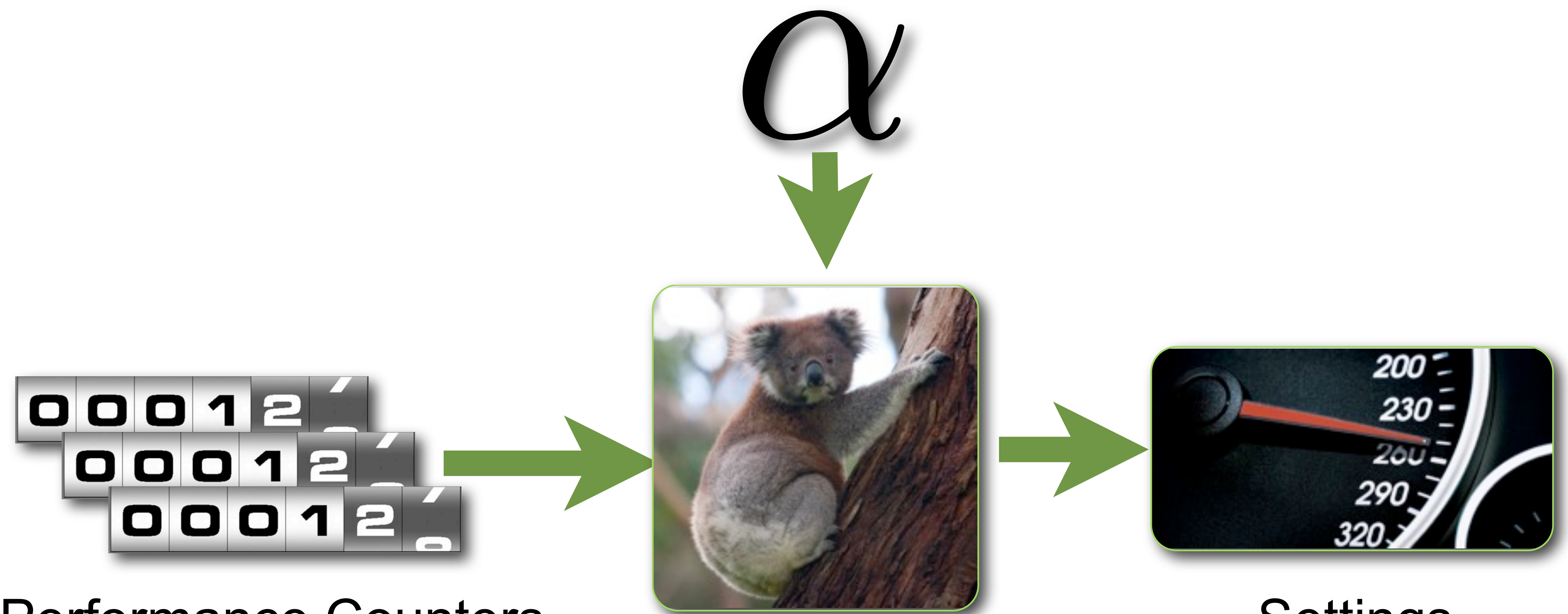
Performance Counters

Settings

Koala elegantly deals with real hardware and real platforms by using models to represent the system. We use:
* CPU performance counters to measure the properties of running workloads;
* A workload-agnostic system tuning knob -- alpha.

And we can select the right combination of settings for the system's scaling knobs.

Koala overview

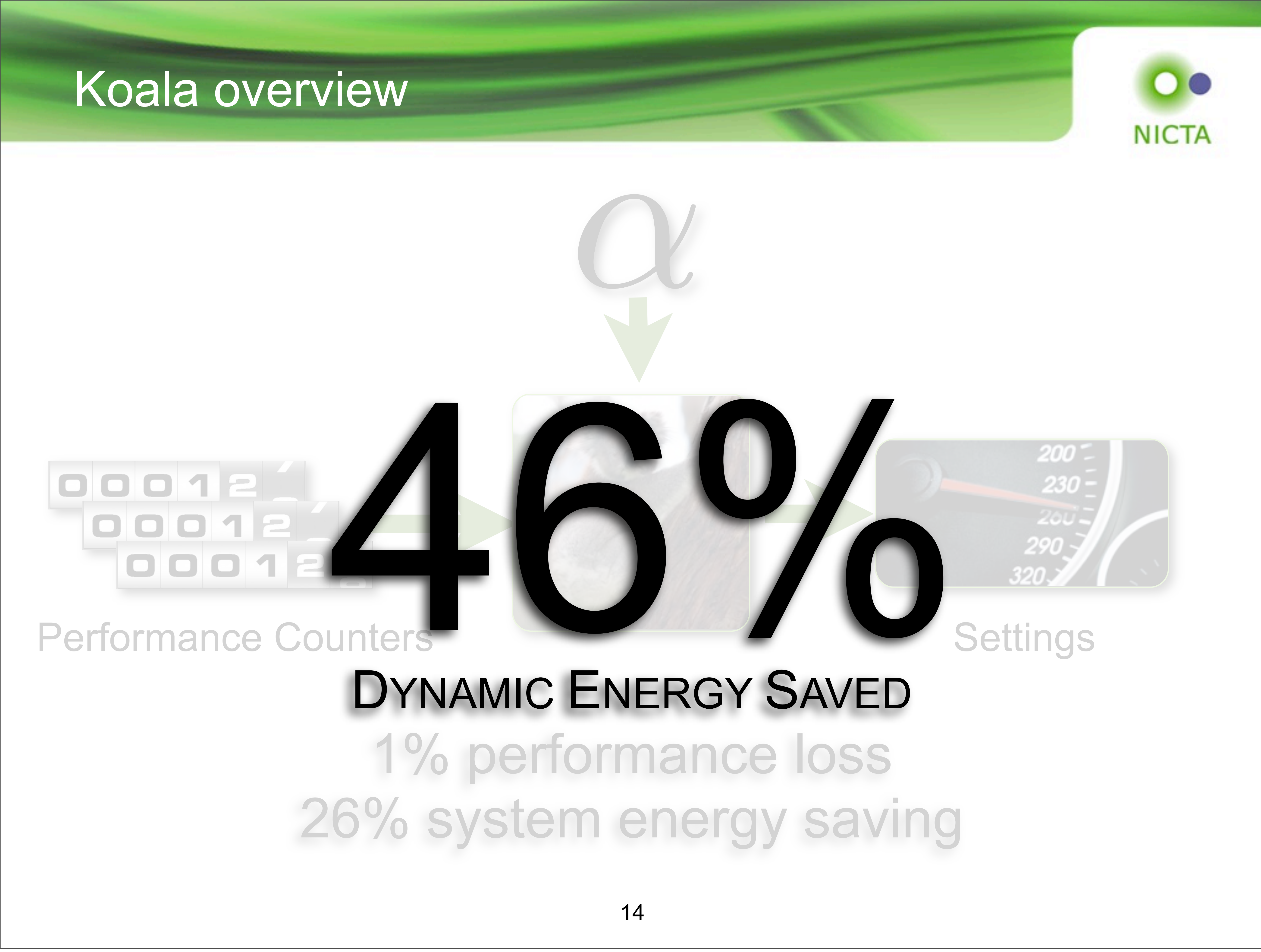


1% performance loss
26% system energy saving

Saturday, 4 April 2009

Koala elegantly deals with real hardware and real platforms by using models to represent the system. We use:
* CPU performance counters to measure the properties of running workloads;
* A workload-agnostic system tuning knob -- alpha.

And we can select the right combination of settings for the system’s scaling knobs.

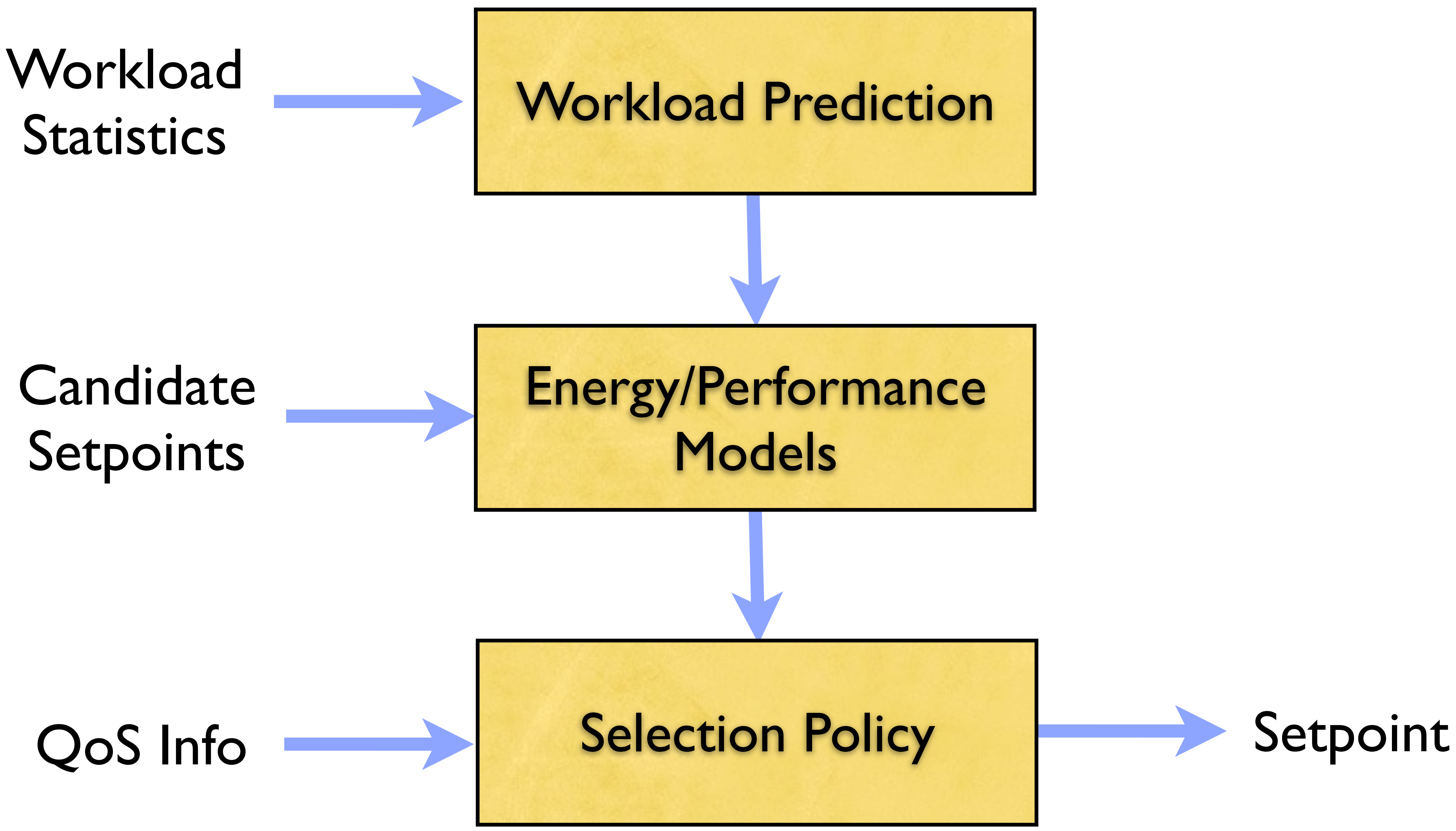


Saturday, 4 April 2009

Koala elegantly deals with real hardware and real platforms by using models to represent the system. We use:
* CPU performance counters to measure the properties of running workloads;
* A workload-agnostic system tuning knob -- alpha.

And we can select the right combination of settings for the system's scaling knobs.

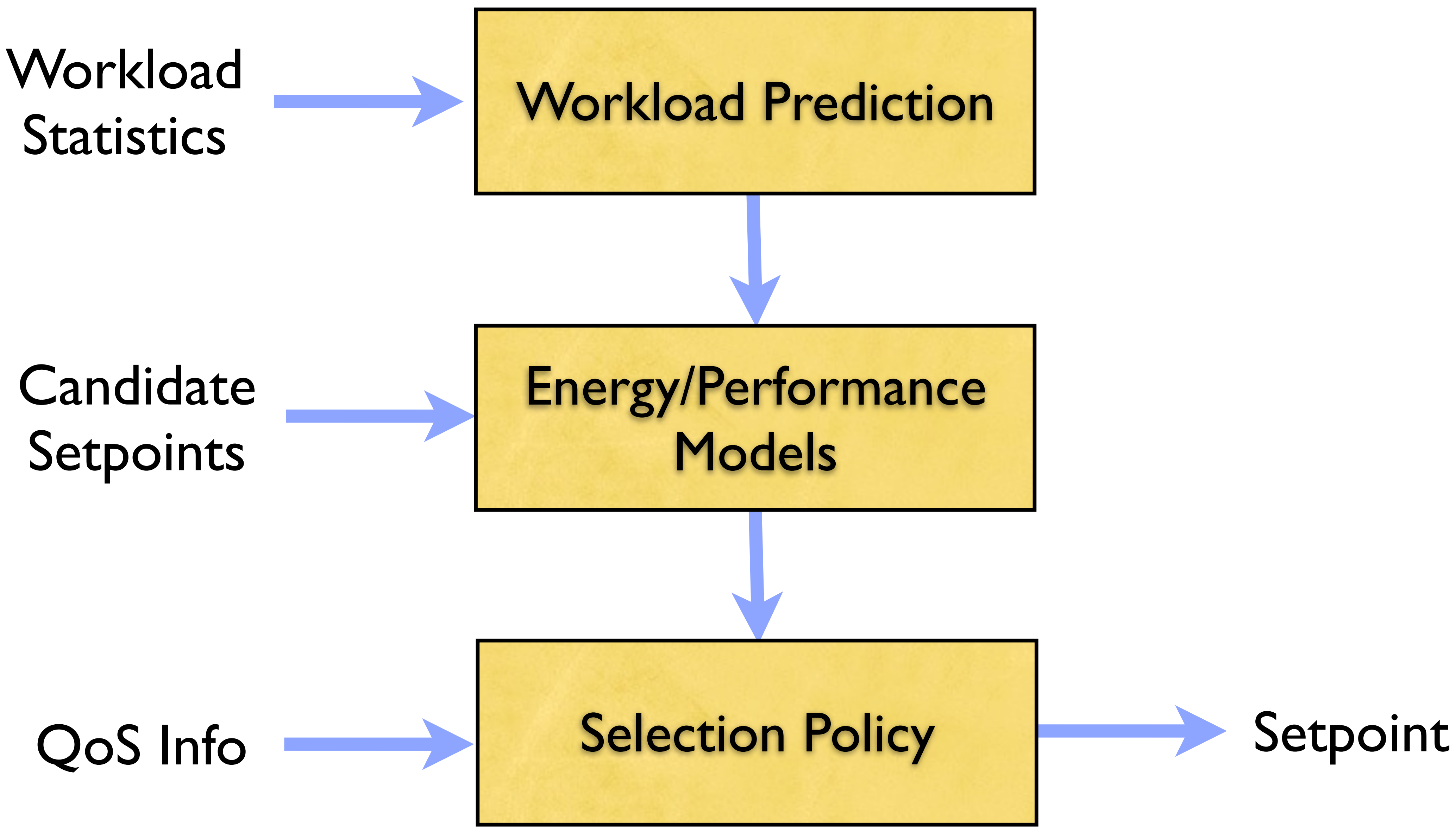
The Koala Approach



Saturday, 4 April 2009

- First, we look at which workloads we are going to run, and predict some characteristics about those workloads based on what they've been doing recently (i.e. we argue temporal locality).
- Second, we use the information from the workload prediction to estimate what the performance and energy would be for various candidate setpoints.
- Third, we use a selection policy to choose from the candidate setpoints, based on quality of service constraints.

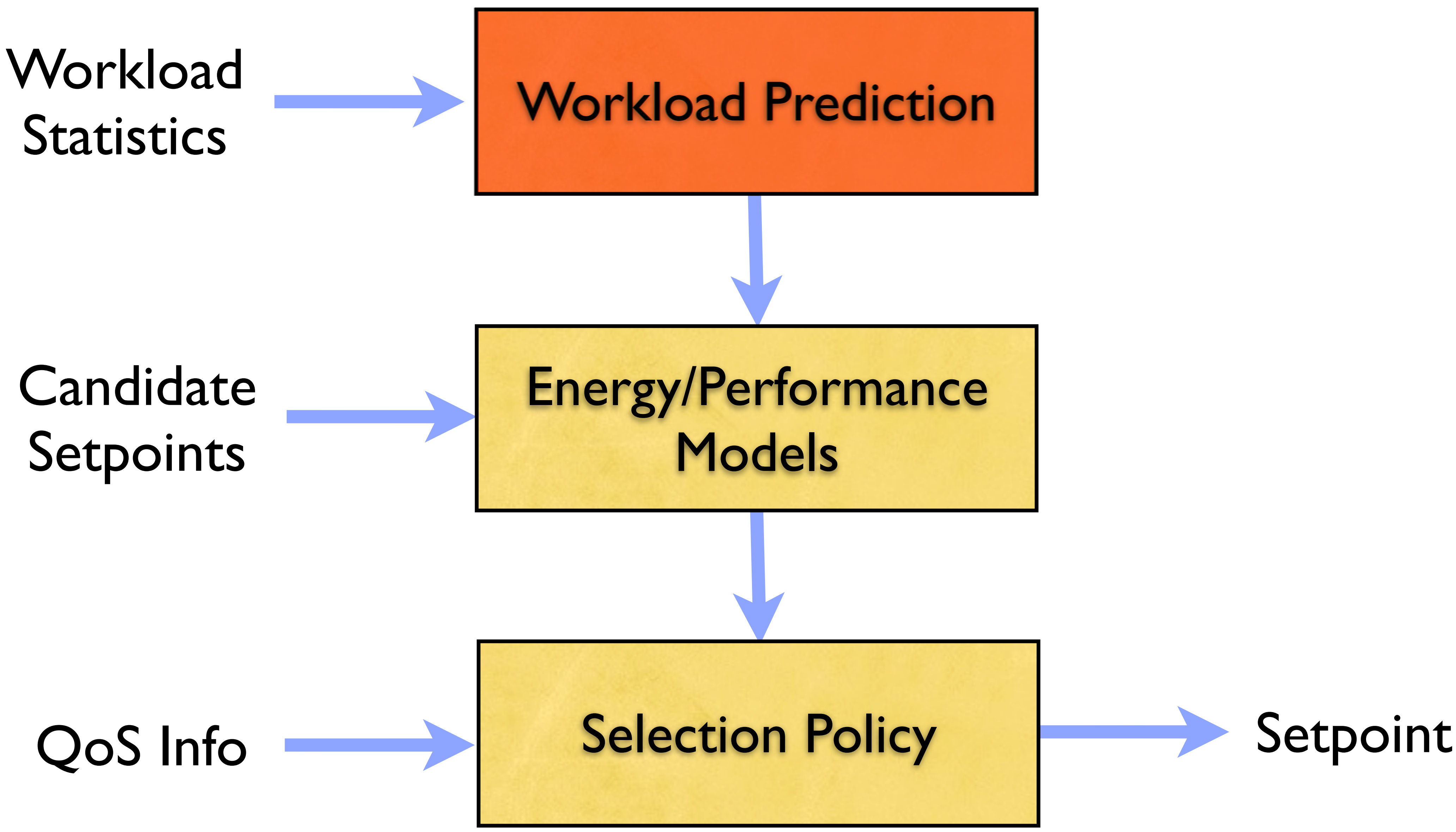
Workload Prediction



Our workload predictor is, at present, very simple. There is lots of work around on how this can be done much better, but our present method is to assume locality -- workloads will continue to do what they've been doing -- we assume that the next time slice will have the same properties as the previous timeslice.

Multi-tasking -- if you're running multiple workloads, the settings need to be appropriate for the particular application.

Workload Prediction

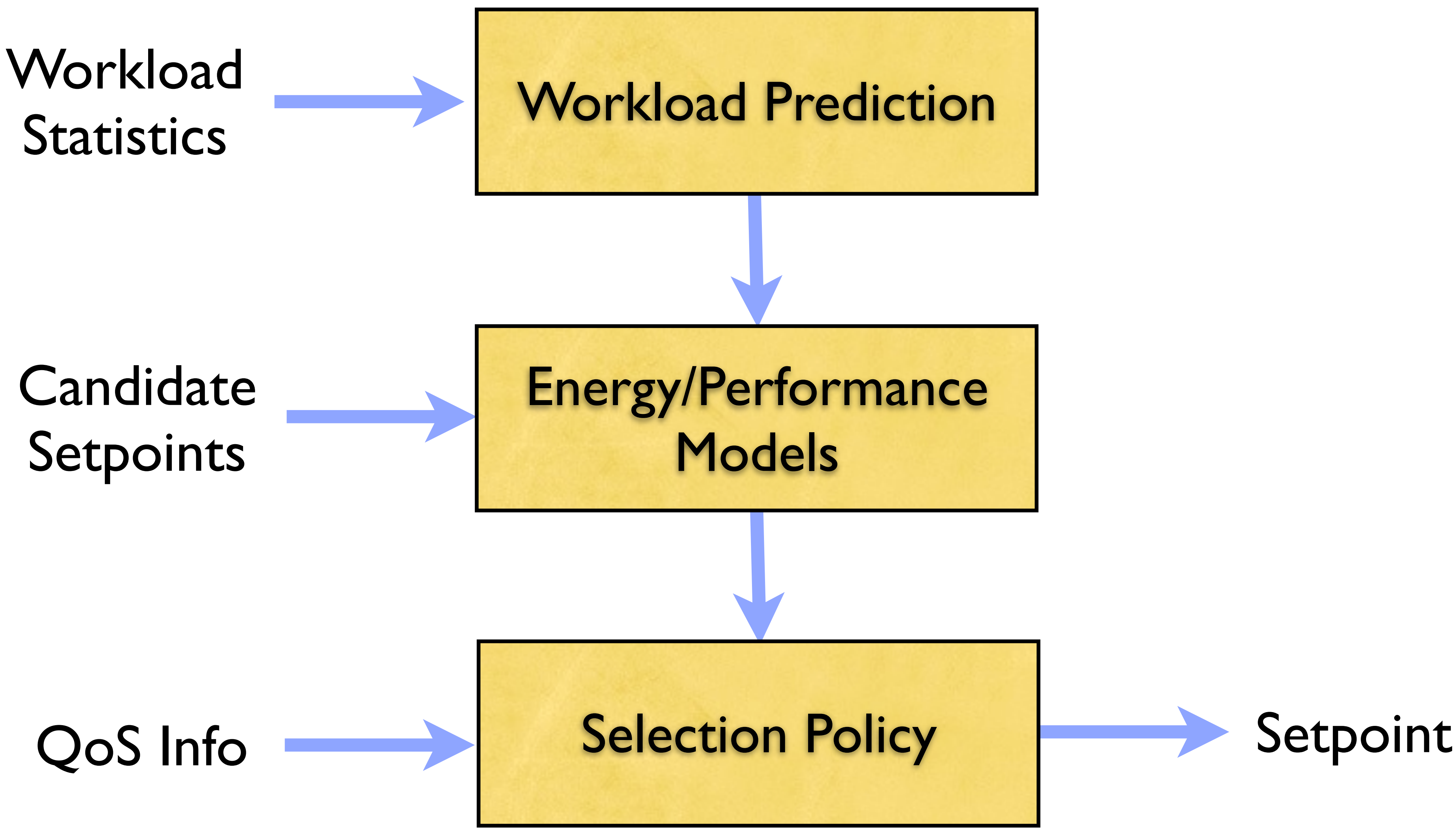


Saturday, 4 April 2009

Our workload predictor is, at present, very simple. There is lots of work around on how this can be done much better, but our present method is to assume locality -- workloads will continue to do what they've been doing -- we assume that the next time slice will have the same properties as the previous timeslice.

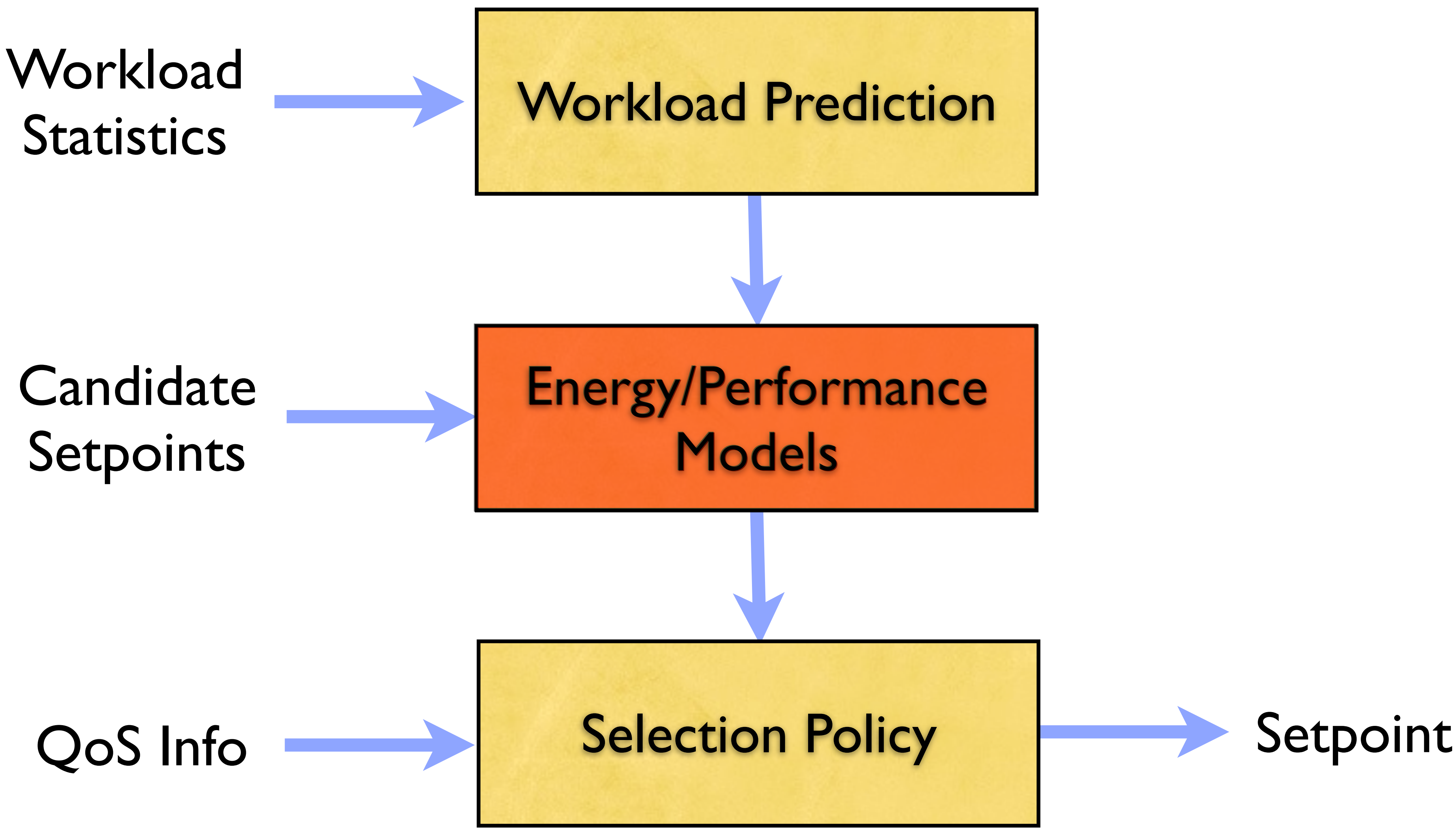
Multi-tasking -- if you're running multiple workloads, the settings need to be appropriate for the particular application.

Energy and performance models



* Next, I'll talk about a major component of Koala -- energy models. These are built and characterised off-line for use at run-time.

Energy and performance models



Saturday, 4 April 2009

* Next, I'll talk about a major component of Koala -- energy models. These are built and characterised off-line for use at run-time.

Energy and performance models

Previously published: performance counter based performance and energy models.

* The models we use here are similar to those which we've discussed in previous work. Our performance model calculates the ratio between the number of cycles at a target frequency and the number of cycles at the sampled frequency. The model which you see here is for a single adjustable frequency, but more generic models are possible and discussed in the papers.

* The energy model we use is also based on previous works -- it is based on the number of events that occur in both voltage scaled and static voltage domains. These events might be as simple as CPU cycles, but can include other events like external bus accesses and particular types of instructions which use more energy than others.

* We select the appropriate performance counters, and characterise the models off-line for each platform. Note though, that these models encapsulate all of the platform-specificity in Koala -- if you can build a model for your platform, Koala can make power management decisions.

* We presented a couple of extra things in this paper -- a method for building the empirical models in a scientific way, modelling idle mode power, switching overheads, temperature, fans, etc.

-- Talk about the macro-level workflow. Characterising off-line. Systematic way of choosing performance counters, etc.

Energy and performance models

NICTA

Previously published: performance counter based performance and energy models.

Performance

$$\frac{C'}{C} = 1 + \beta_0 (f'_{cpu} - f_{cpu}) \frac{PMC_0}{C} + \ldots$$

* The models we use here are similar to those which we've discussed in previous work. Our performance model calculates the ratio between the number of cycles at a target frequency and the number of cycles at the sampled frequency. The model which you see here is for a single adjustable frequency, but more generic models are possible and discussed in the papers.

* The energy model we use is also based on previous works -- it is based on the number of events that occur in both voltage scaled and static voltage domains. These events might be as simple as CPU cycles, but can include other events like external bus accesses and particular types of instructions which use more energy than others.

* We select the appropriate performance counters, and characterise the models off-line for each platform. Note though, that these models encapsulate all of the platform-specificity in Koala -- if you can build a model for your platform, Koala can make power management decisions.

* We presented a couple of extra things in this paper -- a method for building the empirical models in a scientific way, modelling idle mode power, switching overheads, temperature, fans, etc.

-- Talk about the macro-level workflow. Characterising off-line. Systematic way of choosing performance counters, etc.

Energy and performance models

Previously published: performance counter based performance and energy models.

Performance

$$\frac{C'}{C} = 1 + \beta_0 (f'_{cpu} - f_{cpu}) \frac{PMC_0}{C} + \ldots$$

Energy

$$\begin{aligned} E' = V'^2_{cpu}(\alpha_0 C' + \alpha_1 PMC_0 + \ldots) + \\ (\gamma_0 PMC_0 + \ldots) + \\ P_{static} T' \end{aligned}$$

* The models we use here are similar to those which we've discussed in previous work. Our performance model calculates the ratio between the number of cycles at a target frequency and the number of cycles at the sampled frequency. The model which you see here is for a single adjustable frequency, but more generic models are possible and discussed in the papers.

* The energy model we use is also based on previous works -- it is based on the number of events that occur in both voltage scaled and static voltage domains. These events might be as simple as CPU cycles, but can include other events like external bus accesses and particular types of instructions which use more energy than others.

* We select the appropriate performance counters, and characterise the models off-line for each platform. Note though, that these models encapsulate all of the platform-specificity in Koala -- if you can build a model for your platform, Koala can make power management decisions.

* We presented a couple of extra things in this paper -- a method for building the empirical models in a scientific way, modelling idle mode power, switching overheads, temperature, fans, etc.

-- Talk about the macro-level workflow. Characterising off-line. Systematic way of choosing performance counters, etc.

Energy and performance models

Previously published: performance counter based performance and energy models.

Performance

$$\frac{C'}{C} = 1 + \beta_0 (f'_{cpu} - f_{cpu}) \frac{PMC_0}{C} + \ldots$$

Energy

$$\begin{aligned} E' = V'^2_{cpu}(\alpha_0 C' + \alpha_1 PMC_0 + \ldots) + \\ (\gamma_0 PMC_0 + \ldots) + \\ P_{static} T' \end{aligned}$$

Added extras:

- Empirical model building techniques
- Idle power, switching overheads, temperature

* The models we use here are similar to those which we've discussed in previous work. Our performance model calculates the ratio between the number of cycles at a target frequency and the number of cycles at the sampled frequency. The model which you see here is for a single adjustable frequency, but more generic models are possible and discussed in the papers.

* The energy model we use is also based on previous works -- it is based on the number of events that occur in both voltage scaled and static voltage domains. These events might be as simple as CPU cycles, but can include other events like external bus accesses and particular types of instructions which use more energy than others.

* We select the appropriate performance counters, and characterise the models off-line for each platform. Note though, that these models encapsulate all of the platform-specificity in Koala -- if you can build a model for your platform, Koala can make power management decisions.

* We presented a couple of extra things in this paper -- a method for building the empirical models in a scientific way, modelling idle mode power, switching overheads, temperature, fans, etc.

-- Talk about the macro-level workflow. Characterising off-line. Systematic way of choosing performance counters, etc.

Modelling in action

Sample data:

fcpu	Cycles	PMC0	PMC1
1862.0	7445578	56285	134734

Lets look at how these models are used in Koala.

First, we take a sample of the workload and assume that the next timeslice of the workload is going to behave in a very similar way.

Then, for several candidate setpoints, the models predict what the percentage performance and energy will be. Note that the model can be arbitrarily accurate depending on the hardware available -- it can take into account as many of the hardware quirks as possible.

Modelling in action

Sample data:

fcpu	Cycles	PMC0	PMC1
1862.0	7445578	56285	134734

Via the models:

fcpu	Vcpu	Performance	Energy
798	0.988		
1064	1.068		
1197	1.100		
1330	1.148		
1463	1.180		
1596	1.228		
1729	1.260		
1862	1.308		

Lets look at how these models are used in Koala.

First, we take a sample of the workload and assume that the next timeslice of the workload is going to behave in a very similar way.

Then, for several candidate setpoints, the models predict what the percentage performance and energy will be. Note that the model can be arbitrarily accurate depending on the hardware available -- it can take into account as many of the hardware quirks as possible.

Modelling in action

Sample data:

fcpu	Cycles	PMC0	PMC1
1862.0	7445578	56285	134734

Via the models:

fcpu	Vcpu	Performance	Energy
798	0.988	75.5%	93.7%
1064	1.068	84.6%	89.8%
1197	1.100	88.1%	89.2%
1330	1.148	91.1%	90.3%
1463	1.180	93.8%	91.3%
1596	1.228	96.1%	93.9%
1729	1.260	98.1%	96.0%
1862	1.308	100.0%	100%

Lets look at how these models are used in Koala.

First, we take a sample of the workload and assume that the next timeslice of the workload is going to behave in a very similar way.

Then, for several candidate setpoints, the models predict what the percentage performance and energy will be. Note that the model can be arbitrarily accurate depending on the hardware available -- it can take into account as many of the hardware quirks as possible.

Selection Policy

Performance	Energy
75.5%	93.7%
84.6%	89.8%
88.1%	89.2%
91.1%	90.3%
93.8%	91.3%
96.1%	93.9%
98.1%	96.0%
100.0%	100%

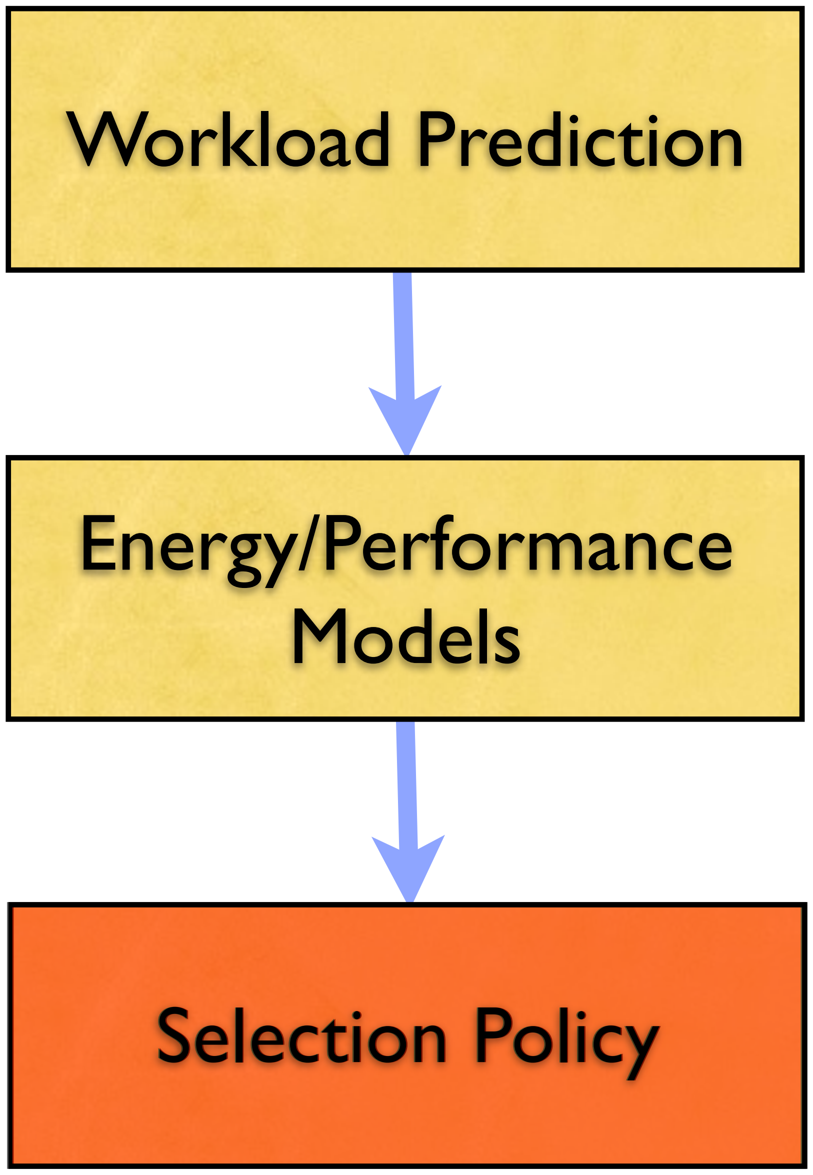
Now that we have some information about the performance and energy used by the workload at various frequencies, we can try to choose a setting based on our needs.

We could choose the minimum energy setting if we really cared about energy, or the minimum time (max performance) setting if we really cared about that. If we had problems with thermal dissipation in the system we could choose the minimum power setting.

One issue with choosing the minimum energy setting is that we might be getting a large performance hit for very little energy savings. This is addressed by a Minimum Energy * Delay policy, since you can expect at least an equal energy saving for any performance hit.

A policy that we've seen in the literature (Frank, and others) that we can easily implement with Koala is a Bounded Performance Degradation policy. Here we bound the performance degradation at 90%, and so we choose setting 4.

Selection Policy



Performance	Energy
75.5%	93.7%
84.6%	89.8%
88.1%	89.2%
91.1%	90.3%
93.8%	91.3%
96.1%	93.9%
98.1%	96.0%
100.0%	100%

Now that we have some information about the performance and energy used by the workload at various frequencies, we can try to choose a setting based on our needs.

We could choose the minimum energy setting if we really cared about energy, or the minimum time (max performance) setting if we really cared about that. If we had problems with thermal dissipation in the system we could choose the minimum power setting.

One issue with choosing the minimum energy setting is that we might be getting a large performance hit for very little energy savings. This is addressed by a Minimum Energy * Delay policy, since you can expect at least an equal energy saving for any performance hit.

A policy that we've seen in the literature (Frank, and others) that we can easily implement with Koala is a Bounded Performance Degradation policy. Here we bound the performance degradation at 90%, and so we choose setting 4.

Selection Policy

NICTA

	Performance	Energy
1	75.5%	93.7%
2	84.6%	89.8%
3	88.1%	89.2%
4	91.1%	90.3%
5	93.8%	91.3%
6	96.1%	93.9%
7	98.1%	96.0%
8	100.0%	100%

Now that we have some information about the performance and energy used by the workload at various frequencies, we can try to choose a setting based on our needs.

We could choose the minimum energy setting if we really cared about energy, or the minimum time (max performance) setting if we really cared about that. If we had problems with thermal dissipation in the system we could choose the minimum power setting.

One issue with choosing the minimum energy setting is that we might be getting a large performance hit for very little energy savings. This is addressed by a Minimum Energy * Delay policy, since you can expect at least an equal energy saving for any performance hit.

A policy that we've seen in the literature (Frank, and others) that we can easily implement with Koala is a Bounded Performance Degradation policy. Here we bound the performance degradation at 90%, and so we choose setting 4.

Selection Policy

- Minimum Energy

	Performance	Energy
1	75.5%	93.7%
2	84.6%	89.8%
3	88.1%	89.2%
4	91.1%	90.3%
5	93.8%	91.3%
6	96.1%	93.9%
7	98.1%	96.0%
8	100.0%	100%

Now that we have some information about the performance and energy used by the workload at various frequencies, we can try to choose a setting based on our needs.

We could choose the minimum energy setting if we really cared about energy, or the minimum time (max performance) setting if we really cared about that. If we had problems with thermal dissipation in the system we could choose the minimum power setting.

One issue with choosing the minimum energy setting is that we might be getting a large performance hit for very little energy savings. This is addressed by a Minimum Energy * Delay policy, since you can expect at least an equal energy saving for any performance hit.

A policy that we've seen in the literature (Frank, and others) that we can easily implement with Koala is a Bounded Performance Degradation policy. Here we bound the performance degradation at 90%, and so we choose setting 4.

Selection Policy

NICTA

- Minimum Energy
- Maximum Performance

	Performance	Energy
1	75.5%	93.7%
2	84.6%	89.8%
3	88.1%	89.2%
4	91.1%	90.3%
5	93.8%	91.3%
6	96.1%	93.9%
7	98.1%	96.0%
8	100.0%	100%

Now that we have some information about the performance and energy used by the workload at various frequencies, we can try to choose a setting based on our needs.

We could choose the minimum energy setting if we really cared about energy, or the minimum time (max performance) setting if we really cared about that. If we had problems with thermal dissipation in the system we could choose the minimum power setting.

One issue with choosing the minimum energy setting is that we might be getting a large performance hit for very little energy savings. This is addressed by a Minimum Energy * Delay policy, since you can expect at least an equal energy saving for any performance hit.

A policy that we've seen in the literature (Frank, and others) that we can easily implement with Koala is a Bounded Performance Degradation policy. Here we bound the performance degradation at 90%, and so we choose setting 4.

Selection Policy

NICTA

- Minimum Energy
- Maximum Performance
- Minimum Power

	Performance	Energy
1	75.5%	93.7%
2	84.6%	89.8%
3	88.1%	89.2%
4	91.1%	90.3%
5	93.8%	91.3%
6	96.1%	93.9%
7	98.1%	96.0%
8	100.0%	100%

Now that we have some information about the performance and energy used by the workload at various frequencies, we can try to choose a setting based on our needs.

We could choose the minimum energy setting if we really cared about energy, or the minimum time (max performance) setting if we really cared about that. If we had problems with thermal dissipation in the system we could choose the minimum power setting.

One issue with choosing the minimum energy setting is that we might be getting a large performance hit for very little energy savings. This is addressed by a Minimum Energy * Delay policy, since you can expect at least an equal energy saving for any performance hit.

A policy that we've seen in the literature (Frank, and others) that we can easily implement with Koala is a Bounded Performance Degradation policy. Here we bound the performance degradation at 90%, and so we choose setting 4.

Selection Policy

- Minimum Energy
- Maximum Performance
- Minimum Power
- Minimum E*D product

	Performance	Energy
1	75.5%	93.7%
2	84.6%	89.8%
3	88.1%	89.2%
4	91.1%	90.3%
5	93.8%	91.3%
6	96.1%	93.9%
7	98.1%	96.0%
8	100.0%	100%

Now that we have some information about the performance and energy used by the workload at various frequencies, we can try to choose a setting based on our needs.

We could choose the minimum energy setting if we really cared about energy, or the minimum time (max performance) setting if we really cared about that. If we had problems with thermal dissipation in the system we could choose the minimum power setting.

One issue with choosing the minimum energy setting is that we might be getting a large performance hit for very little energy savings. This is addressed by a Minimum Energy * Delay policy, since you can expect at least an equal energy saving for any performance hit.

A policy that we've seen in the literature (Frank, and others) that we can easily implement with Koala is a Bounded Performance Degradation policy. Here we bound the performance degradation at 90%, and so we choose setting 4.

Selection Policy

NICTA

- Minimum Energy
- Maximum Performance
- Minimum Power
- Minimum E*D product
- Bounded Performance Degradation -- 90%

	Performance	Energy
1	75.5%	93.7%
2	84.6%	89.8%
3	88.1%	89.2%
4	91.1%	90.3%
5	93.8%	91.3%
6	96.1%	93.9%
7	98.1%	96.0%
8	100.0%	100%

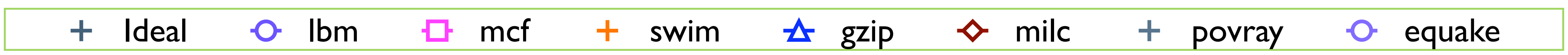
Now that we have some information about the performance and energy used by the workload at various frequencies, we can try to choose a setting based on our needs.

We could choose the minimum energy setting if we really cared about energy, or the minimum time (max performance) setting if we really cared about that. If we had problems with thermal dissipation in the system we could choose the minimum power setting.

One issue with choosing the minimum energy setting is that we might be getting a large performance hit for very little energy savings. This is addressed by a Minimum Energy * Delay policy, since you can expect at least an equal energy saving for any performance hit.

A policy that we’ve seen in the literature (Frank, and others) that we can easily implement with Koala is a Bounded Performance Degradation policy. Here we bound the performance degradation at 90%, and so we choose setting 4.

Bounded performance degradation



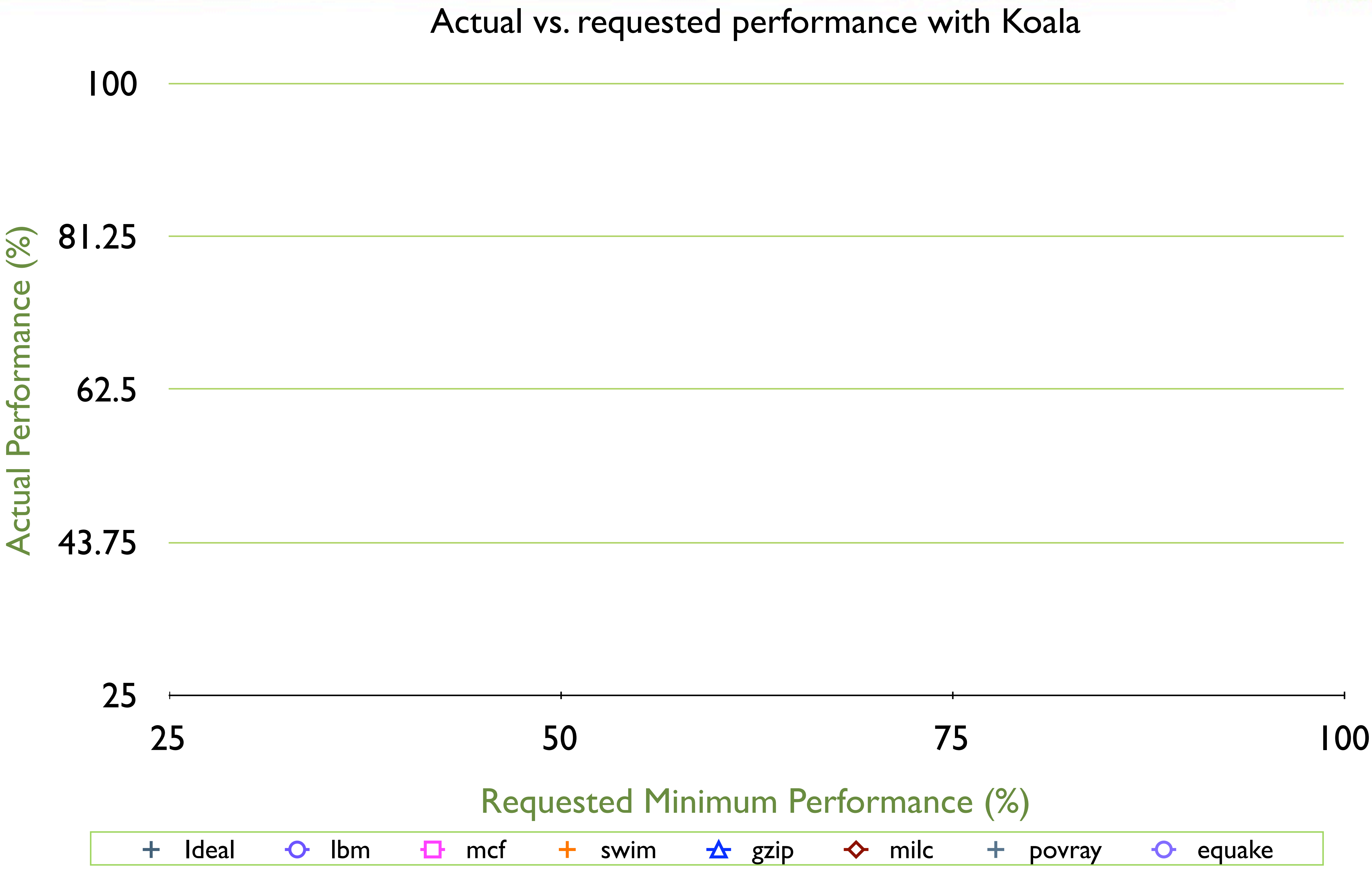
Saturday, 4 April 2009

This plot represents real Koala data -- benchmarks running with the bounded performance degradation policy. The CPU bound benchmarks achieve the requested minimum performance, but the memory bound benchmarks can't be degraded that far because even if you ran at the lowest frequency, the performance wouldn't really change that much. Koala makes the best effort.

The thing to take away from this graph is that if we ask for a performance degradation, the CPU bound benchmarks will definitely do it. Set the performance to 90%, and you'll get 90% of the performance for a CPU bound benchmark.

Bounded performance degradation

NICTA

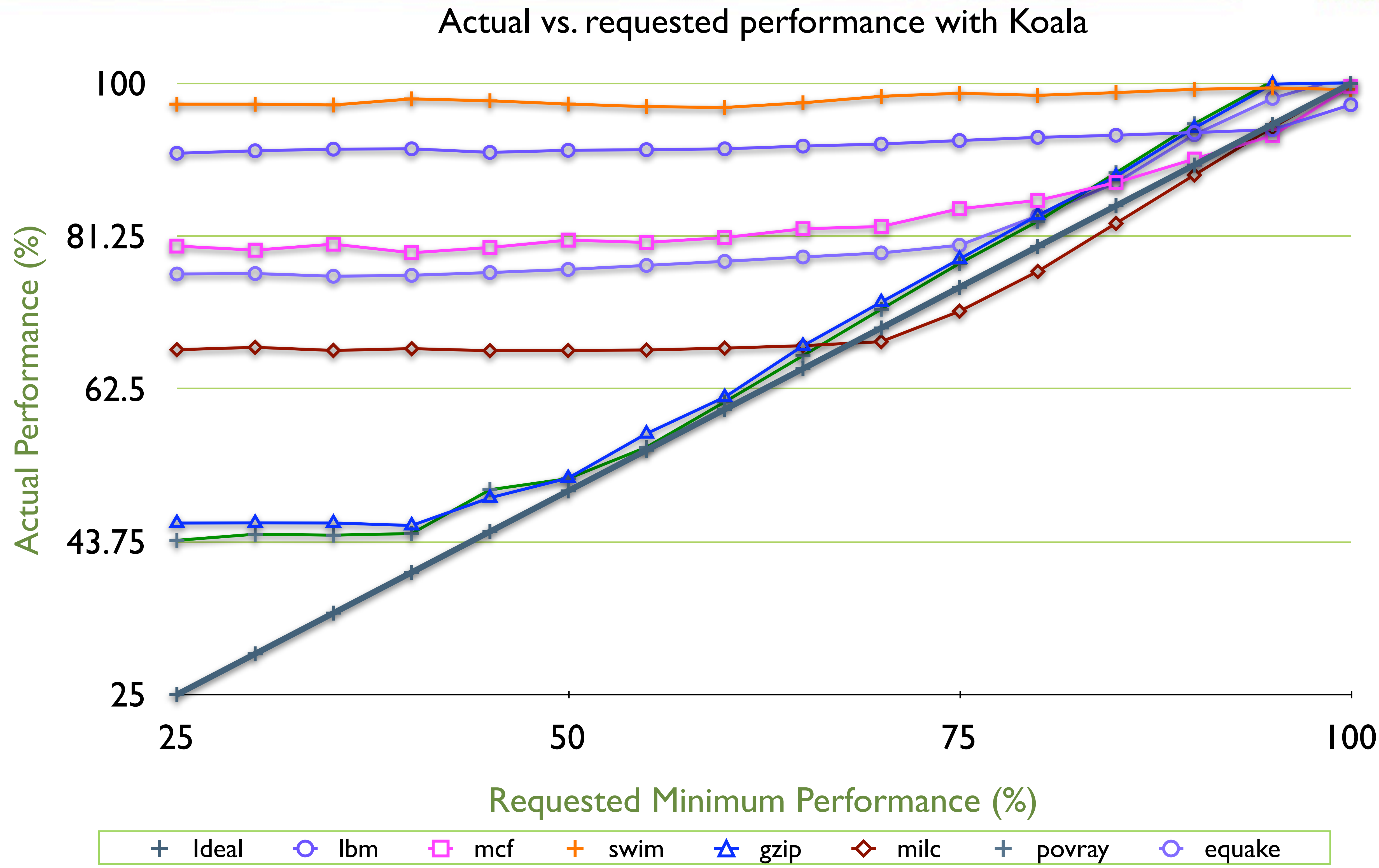


Saturday, 4 April 2009

This plot represents real Koala data -- benchmarks running with the bounded performance degradation policy. The CPU bound benchmarks achieve the requested minimum performance, but the memory bound benchmarks can't be degraded that far because even if you ran at the lowest frequency, the performance wouldn't really change that much. Koala makes the best effort.

The thing to take away from this graph is that if we ask for a performance degradation, the CPU bound benchmarks will definitely do it. Set the performance to 90%, and you'll get 90% of the performance for a CPU bound benchmark.

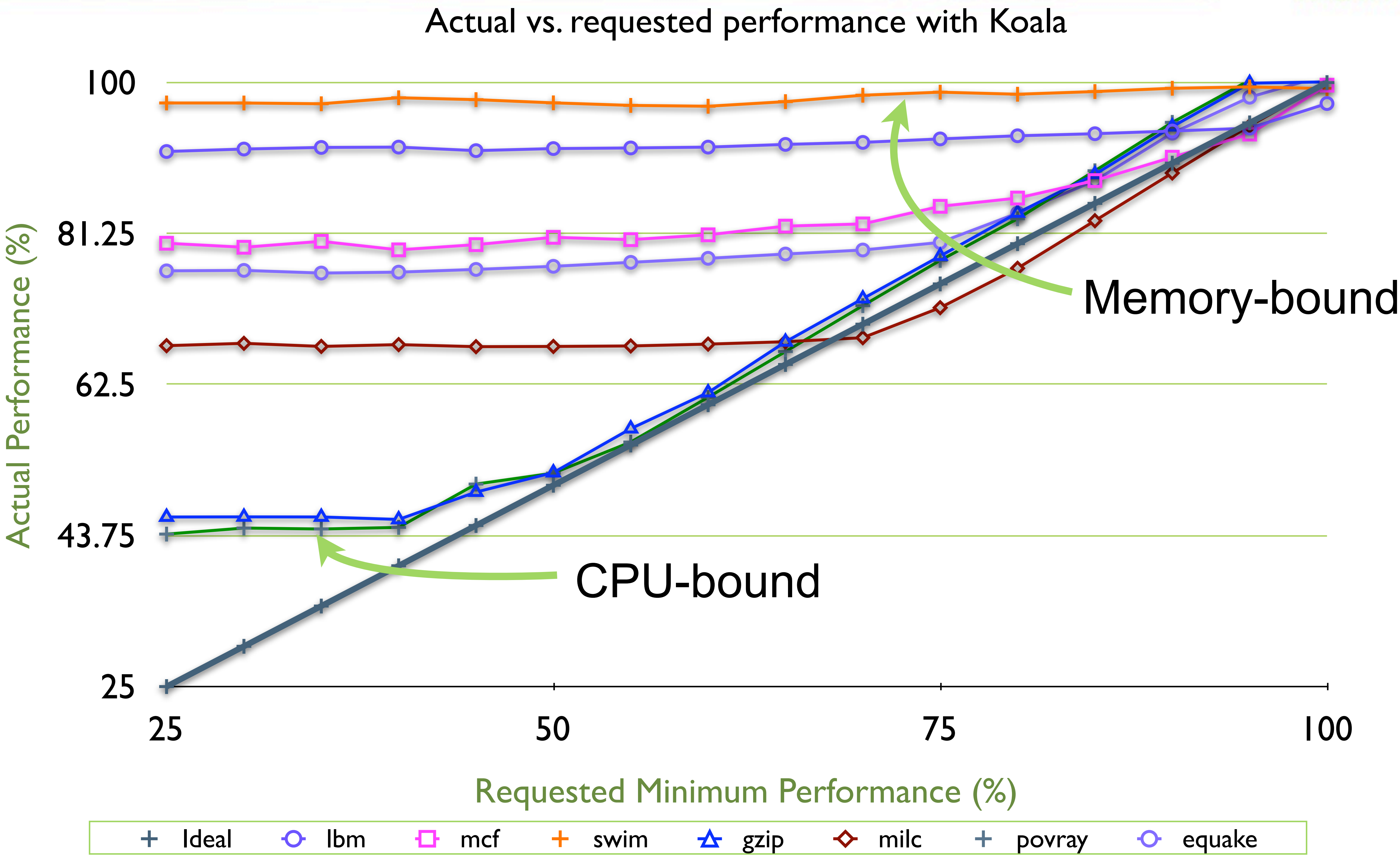
Bounded performance degradation



This plot represents real Koala data -- benchmarks running with the bounded performance degradation policy. The CPU bound benchmarks achieve the requested minimum performance, but the memory bound benchmarks can't be degraded that far because even if you ran at the lowest frequency, the performance wouldn't really change that much. Koala makes the best effort.

The thing to take away from this graph is that if we ask for a performance degradation, the CPU bound benchmarks will definitely do it. Set the performance to 90%, and you'll get 90% of the performance for a CPU bound benchmark.

Bounded performance degradation



Saturday, 4 April 2009

This plot represents real Koala data -- benchmarks running with the bounded performance degradation policy. The CPU bound benchmarks achieve the requested minimum performance, but the memory bound benchmarks can't be degraded that far because even if you ran at the lowest frequency, the performance wouldn't really change that much. Koala makes the best effort.

The thing to take away from this graph is that if we ask for a performance degradation, the CPU bound benchmarks will definitely do it. Set the performance to 90%, and you'll get 90% of the performance for a CPU bound benchmark.

Bounded performance degradation

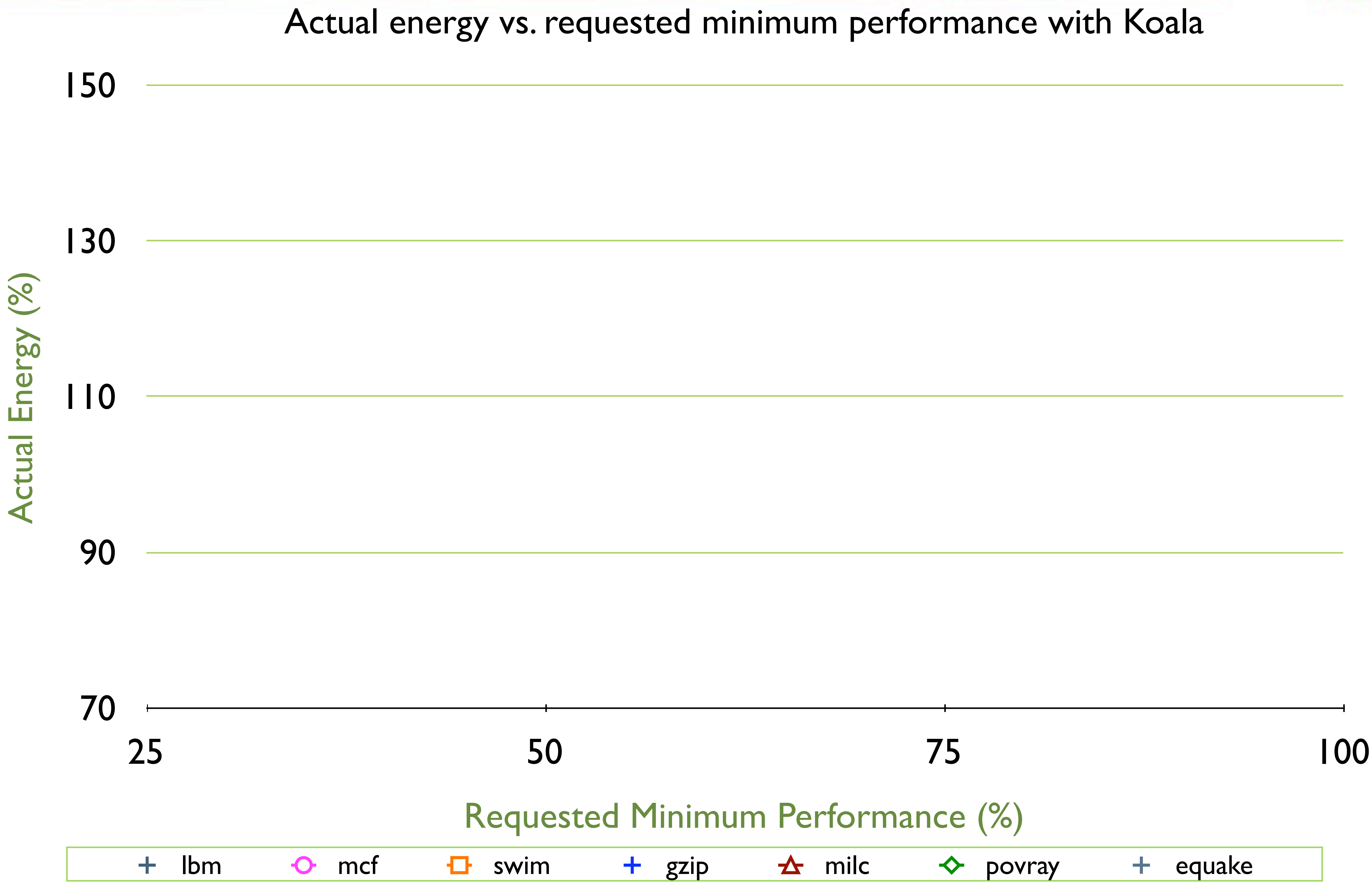
NICTA

Looking at the energy, we see that the CPU bound benchmarks also behave differently to memory bound ones. For any value of minimum performance les than 100%, the CPU bound benchmarks use _more_ energy. The memory bound benchmarks use less. A value found empirically is around about 90%, but that is sub-optimal for both the CPU bound and memory-bound benchmarks. Moreover -- we lose 10% of the performance on the CPU-bound benchmarks... for an energy INCREASE!

It means that the performance setting really isn't a globally applicable metric for how much we want to scale.

We need a policy that is entirely workload agnostic!

Bounded performance degradation



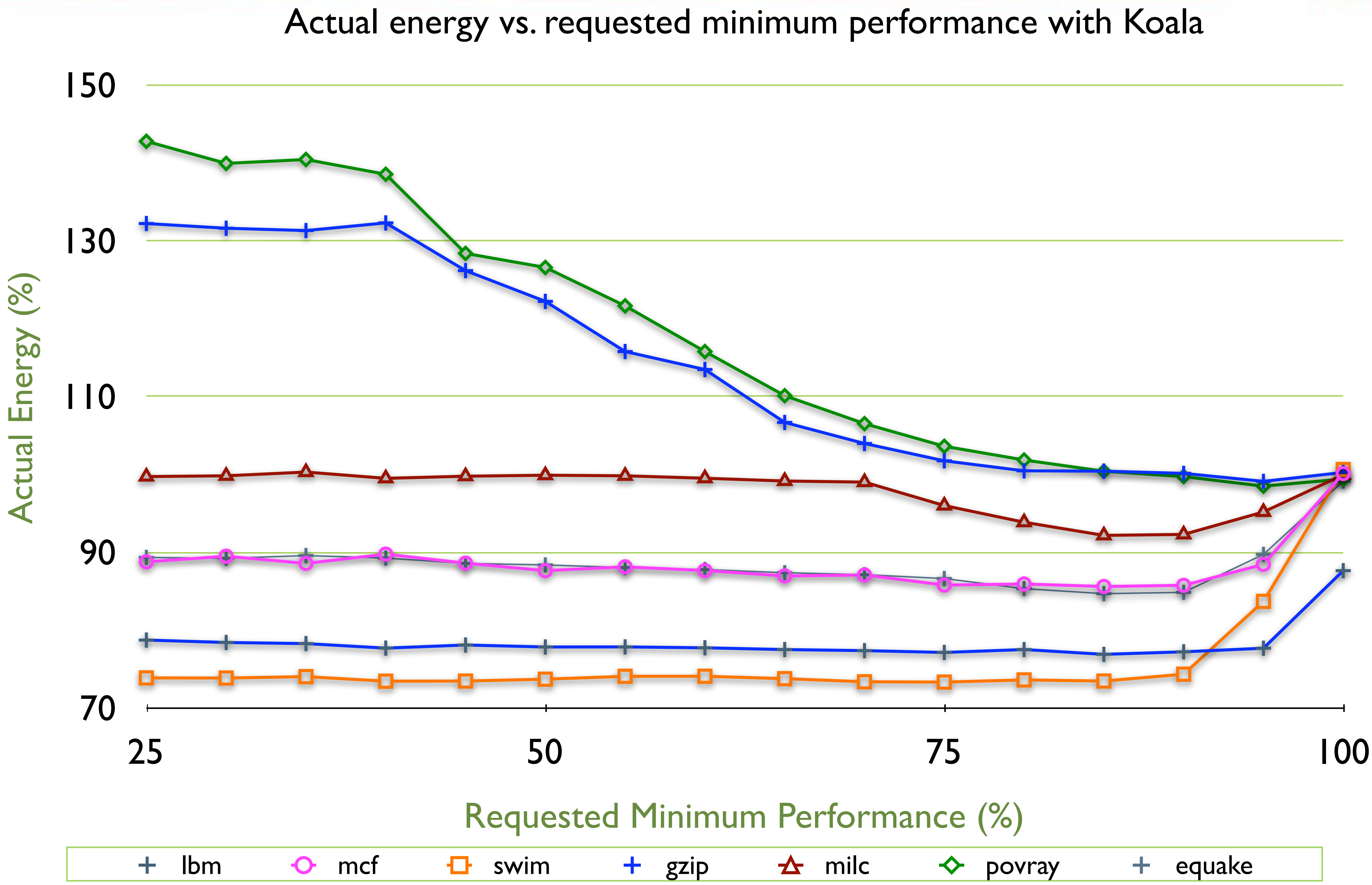
Saturday, 4 April 2009

Looking at the energy, we see that the CPU bound benchmarks also behave differently to memory bound ones. For any value of minimum performance les than 100%, the CPU bound benchmarks use _more_ energy. The memory bound benchmarks use less. A value found empirically is around about 90%, but that is sub-optimal for both the CPU bound and memory-bound benchmarks. Moreover -- we lose 10% of the performance on the CPU-bound benchmarks... for an energy INCREASE!

It means that the performance setting really isn't a globally applicable metric for how much we want to scale.

We need a policy that is entirely workload agnostic!

Bounded performance degradation



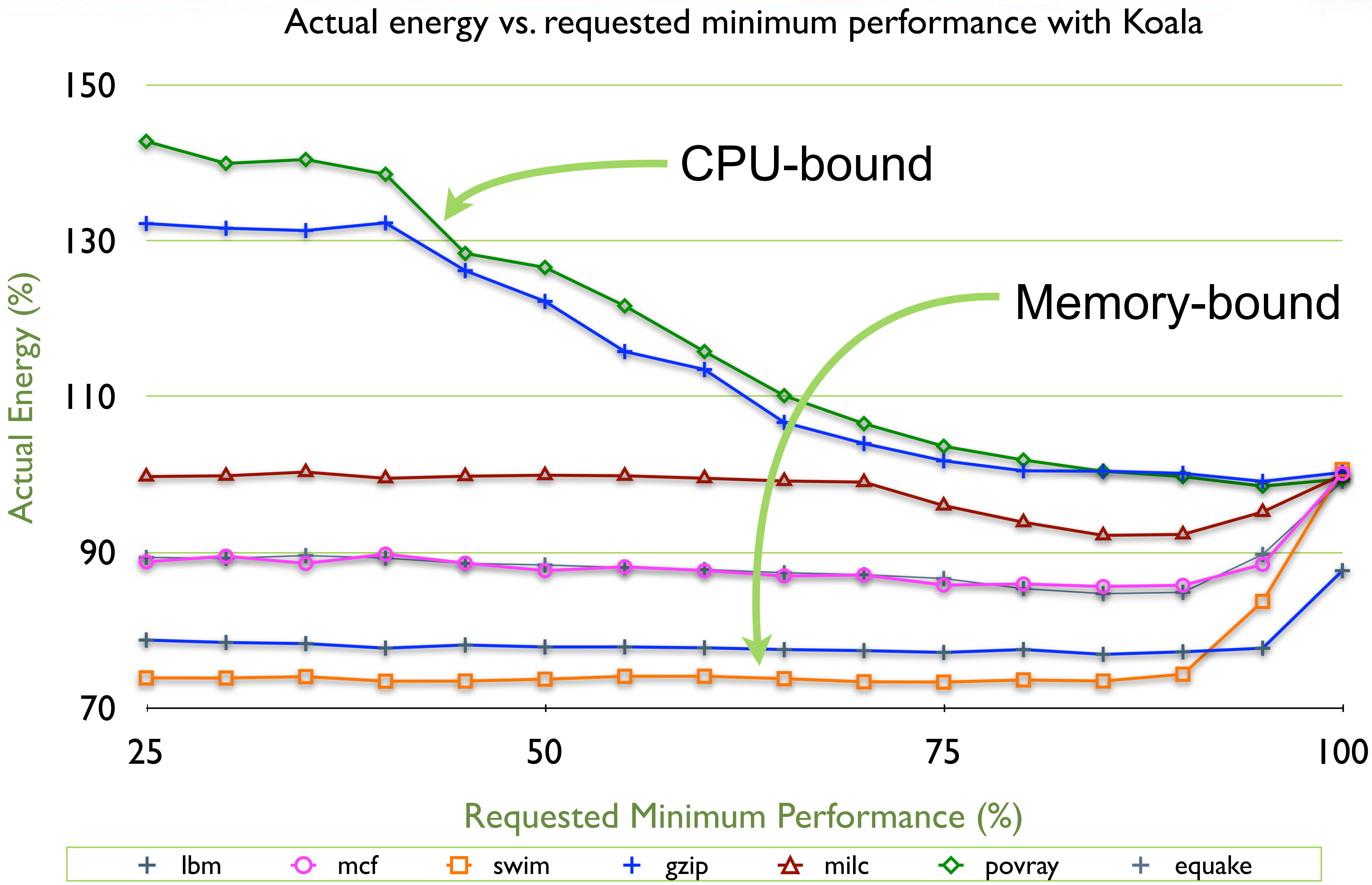
Saturday, 4 April 2009

Looking at the energy, we see that the CPU bound benchmarks also behave differently to memory bound ones. For any value of minimum performance les than 100%, the CPU bound benchmarks use _more_ energy. The memory bound benchmarks use less. A value found empirically is around about 90%, but that is sub-optimal for both the CPU bound and memory-bound benchmarks. Moreover -- we lose 10% of the performance on the CPU-bound benchmarks... for an energy INCREASE!

It means that the performance setting really isn't a globally applicable metric for how much we want to scale.

We need a policy that is entirely workload agnostic!

Bounded performance degradation



Looking at the energy, we see that the CPU bound benchmarks also behave differently to memory bound ones. For any value of minimum performance les than 100%, the CPU bound benchmarks use _more_ energy. The memory bound benchmarks use less. A value found empirically is around about 90%, but that is sub-optimal for both the CPU bound and memory-bound benchmarks. Moreover -- we lose 10% of the performance on the CPU-bound benchmarks... for an energy INCREASE!

It means that the performance setting really isn't a globally applicable metric for how much we want to scale.

We need a policy that is entirely workload agnostic!

Generalised Energy-Delay Policy

$$\eta = P^{(1-\alpha)} T^{(1+\alpha)}$$

- What if, instead of minimising energy, or time, or power, we minimised some function which gave us a good trade-off.
- We came up with such a function, and call the resulting policy generalised E*D, or Alpha.
- By using various different values of alpha, we can express the full spectrum of policies, including Minimum Energy, Minimum Time (max performance), Minimum Energy and, for thermal throttling, minimum power.

Generalised Energy-Delay Policy

$$\eta = PT = E$$

0: Minimum Energy

Saturday, 4 April 2009

- What if, instead of minimising energy, or time, or power, we minimised some function which gave us a good trade-off.
- We came up with such a function, and call the resulting policy generalised E*D, or Alpha.
- By using various different values of alpha, we can express the full spectrum of policies, including Minimum Energy, Minimum Time (max performance), Minimum Energy and, for thermal throttling, minimum power.

Generalised Energy-Delay Policy

$$\eta = P^{(1-\alpha)} T^{(1+\alpha)}$$

0: Minimum Energy

- What if, instead of minimising energy, or time, or power, we minimised some function which gave us a good trade-off.
- We came up with such a function, and call the resulting policy generalised E*D, or Alpha.
- By using various different values of alpha, we can express the full spectrum of policies, including Minimum Energy, Minimum Time (max performance), Minimum Energy and, for thermal throttling, minimum power.

Generalised Energy-Delay Policy

NICTA

$$\eta = T^2$$

0: Minimum Energy

1: Maximum Performance

- What if, instead of minimising energy, or time, or power, we minimised some function which gave us a good trade-off.
- We came up with such a function, and call the resulting policy generalised E*D, or Alpha.
- By using various different values of alpha, we can express the full spectrum of policies, including Minimum Energy, Minimum Time (max performance), Minimum Energy and, for thermal throttling, minimum power.

Generalised Energy-Delay Policy

NICTA

$$\eta = P^{(1-\alpha)} T^{(1+\alpha)}$$

0: **Minimum Energy**

1: **Maximum Performance**

- What if, instead of minimising energy, or time, or power, we minimised some function which gave us a good trade-off.
- We came up with such a function, and call the resulting policy generalised E*D, or Alpha.
- By using various different values of alpha, we can express the full spectrum of policies, including Minimum Energy, Minimum Time (max performance), Minimum Energy and, for thermal throttling, minimum power.

Generalised Energy-Delay Policy

$$\eta = (ET)^{\frac{2}{3}}$$

0: Minimum Energy

0.33: Minimum Energy-Delay

1: Maximum Performance

- What if, instead of minimising energy, or time, or power, we minimised some function which gave us a good trade-off.
- We came up with such a function, and call the resulting policy generalised E*D, or Alpha.
- By using various different values of alpha, we can express the full spectrum of policies, including Minimum Energy, Minimum Time (max performance), Minimum Energy and, for thermal throttling, minimum power.

Generalised Energy-Delay Policy

NICTA

$$\eta = P^{(1-\alpha)}T^{(1+\alpha)}$$

0: **Minimum Energy**

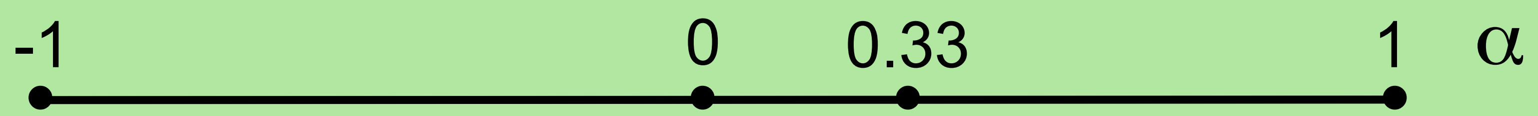
0.33: **Minimum Energy-Delay**

1: **Maximum Performance**

- What if, instead of minimising energy, or time, or power, we minimised some function which gave us a good trade-off.
- We came up with such a function, and call the resulting policy generalised E*D, or Alpha.
- By using various different values of alpha, we can express the full spectrum of policies, including Minimum Energy, Minimum Time (max performance), Minimum Energy and, for thermal throttling, minimum power.

Generalised Energy-Delay Policy

$$\eta = P^2$$



-1: Minimum Power

0: Minimum Energy

0.33: Minimum Energy-Delay

1: Maximum Performance

- What if, instead of minimising energy, or time, or power, we minimised some function which gave us a good trade-off.
- We came up with such a function, and call the resulting policy generalised E*D, or Alpha.
- By using various different values of alpha, we can express the full spectrum of policies, including Minimum Energy, Minimum Time (max performance), Minimum Energy and, for thermal throttling, minimum power.

Generalised Energy-Delay Policy

NICTA

$$\eta = P^{(1-\alpha)}T^{(1+\alpha)}$$

-1 0 0.33 1 α

-1: **Minimum Power**

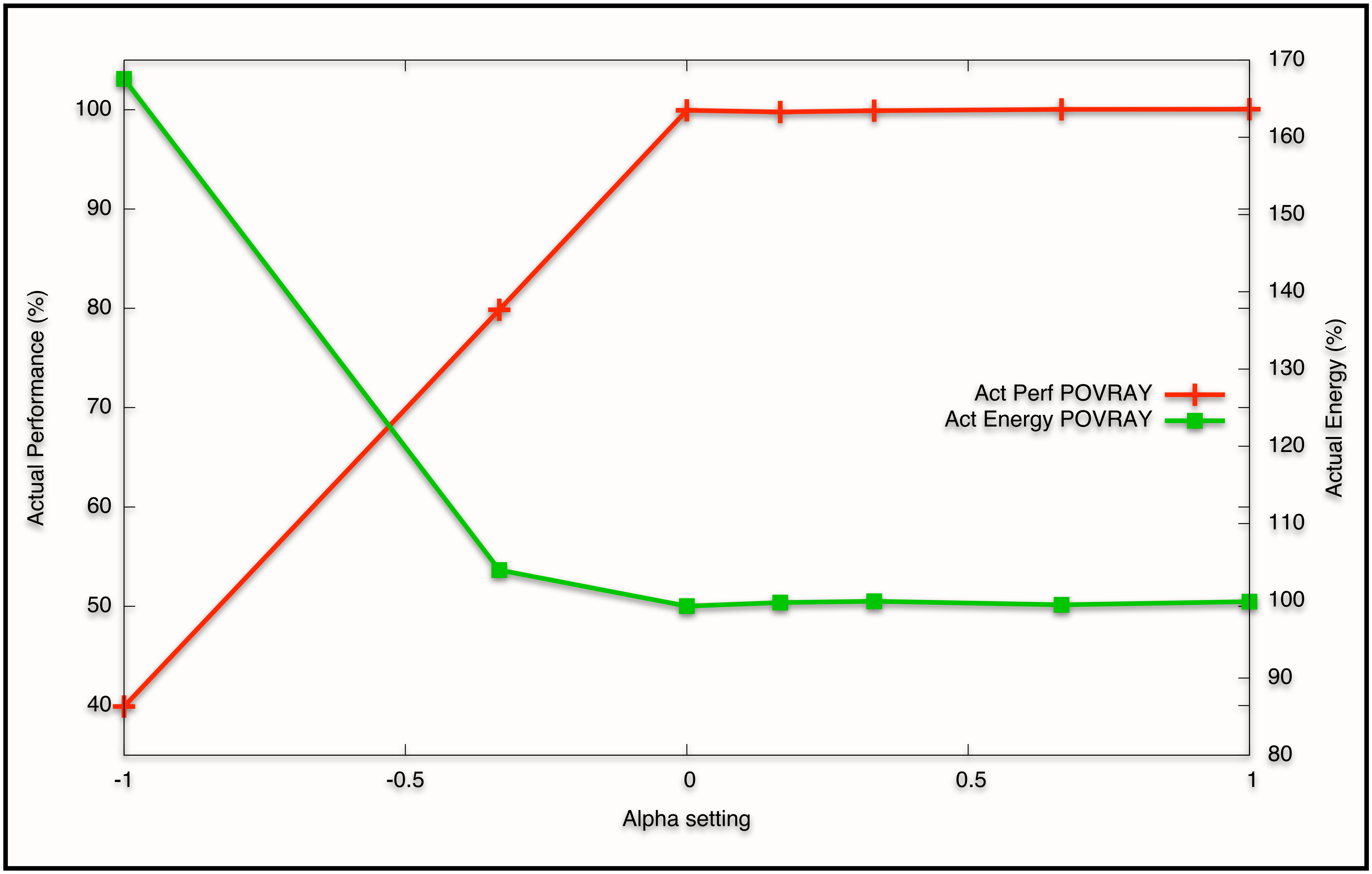
0: **Minimum Energy**

0.33: **Minimum Energy-Delay**

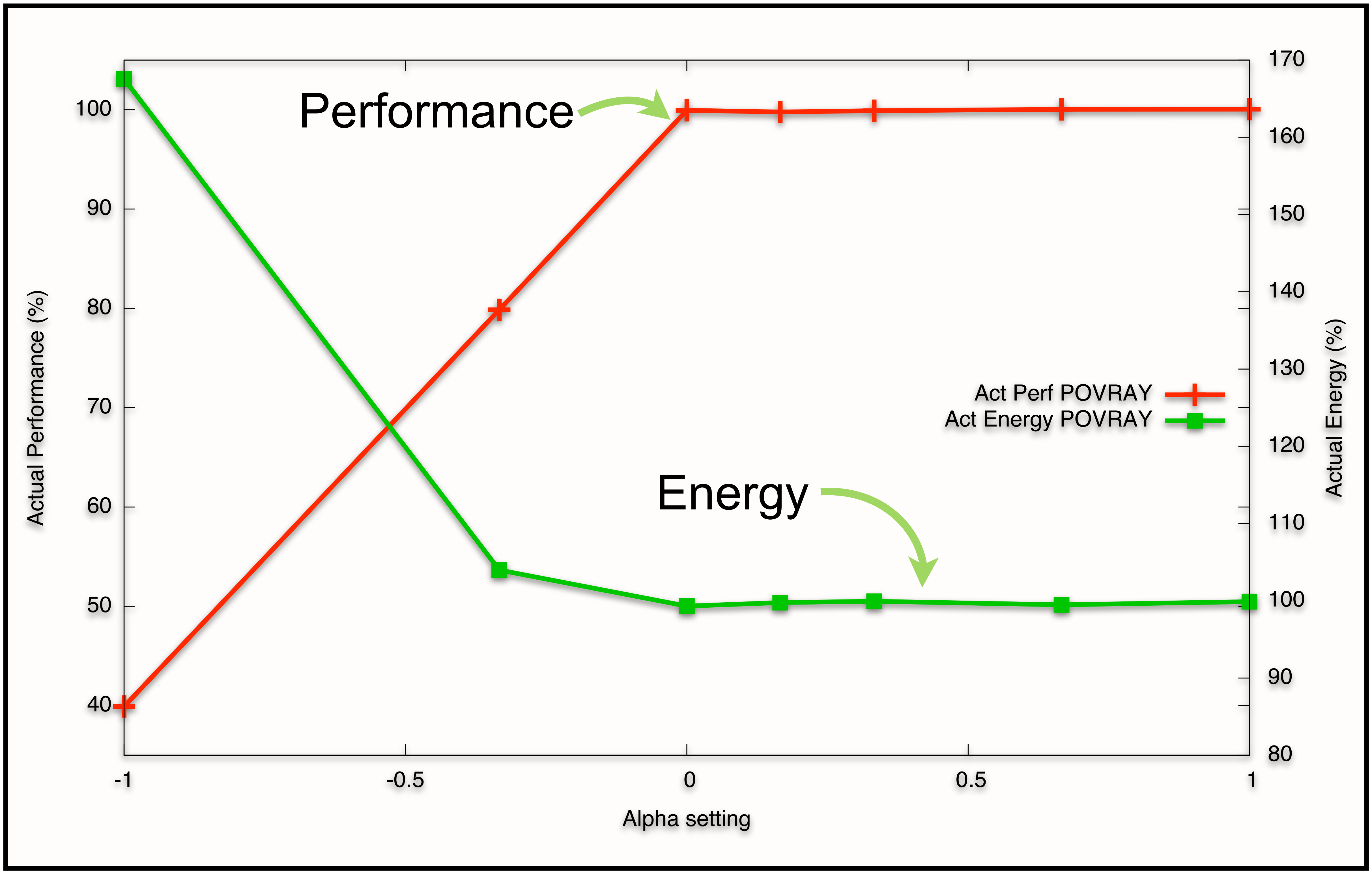
1: **Maximum Performance**

- What if, instead of minimising energy, or time, or power, we minimised some function which gave us a good trade-off.
- We came up with such a function, and call the resulting policy generalised E*D, or Alpha.
- By using various different values of alpha, we can express the full spectrum of policies, including Minimum Energy, Minimum Time (max performance), Minimum Energy and, for thermal throttling, minimum power.

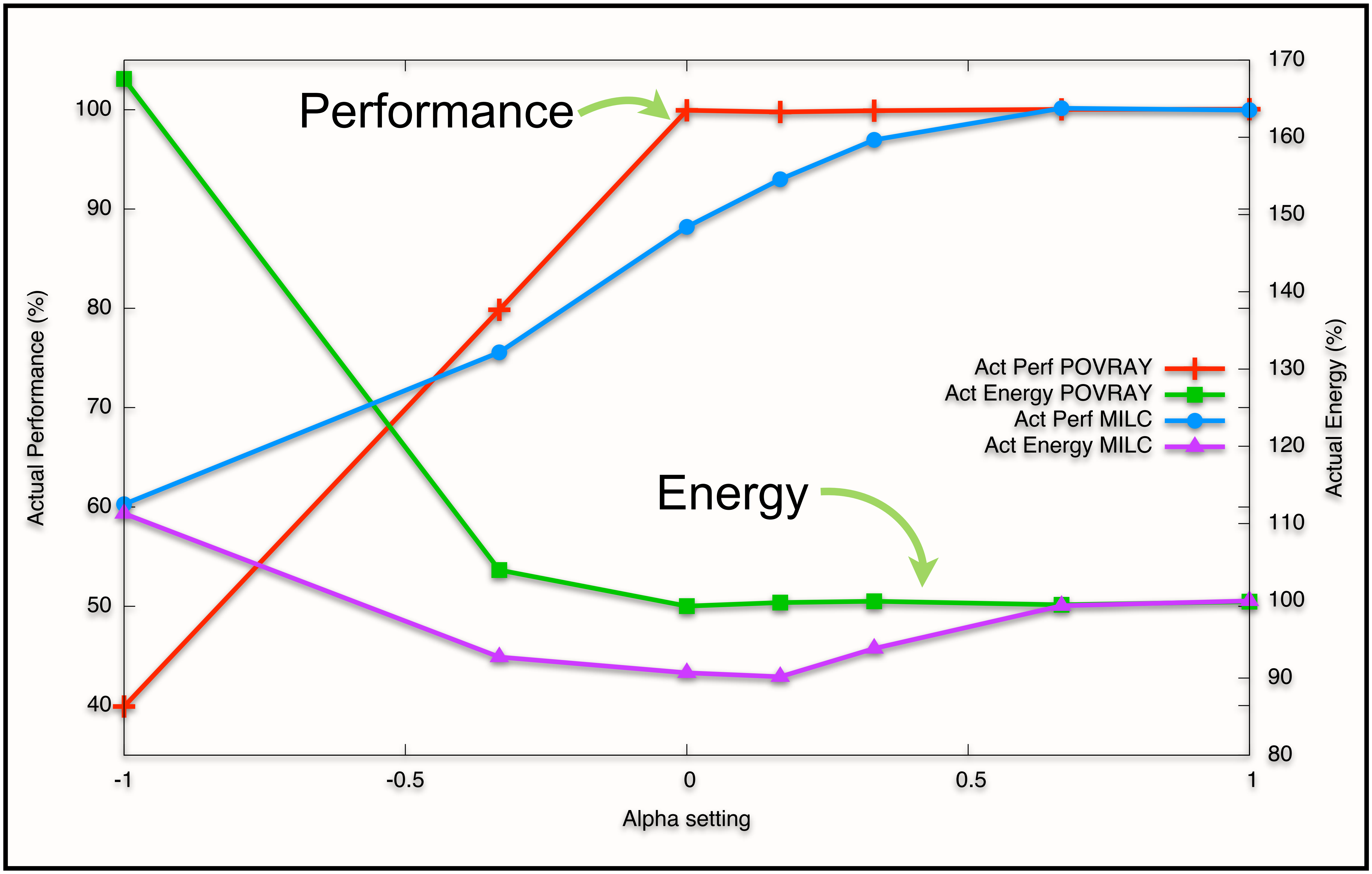
Generalised Energy-Delay Policy



Generalised Energy-Delay Policy



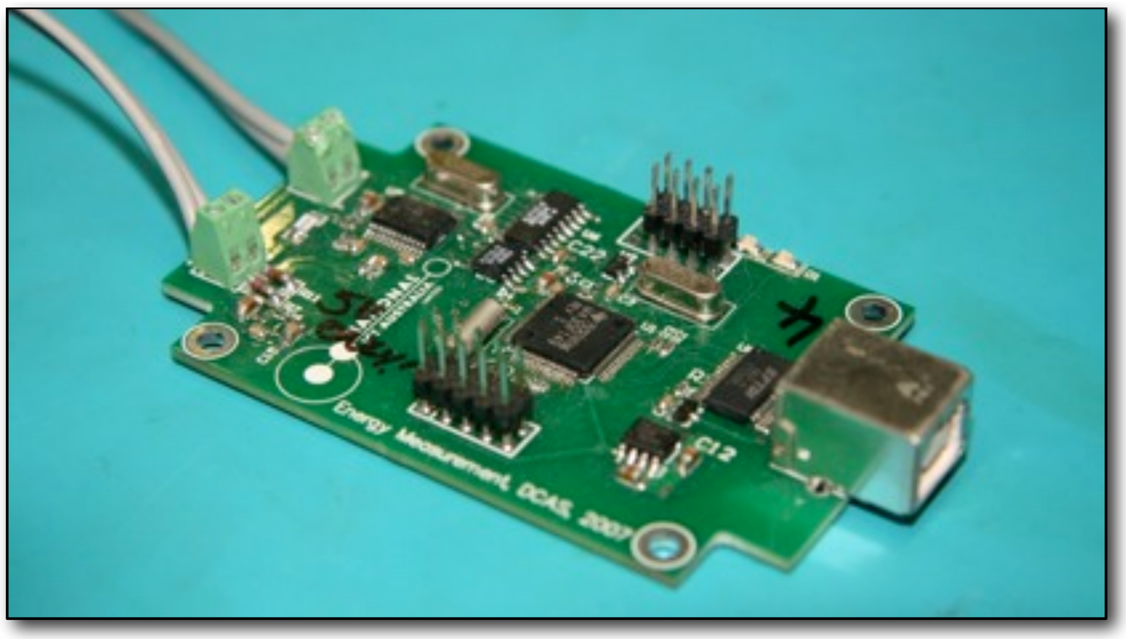
Generalised Energy-Delay Policy



Saturday, 4 April 2009

Implementation

- Implemented in Linux 2.6.24.
- Characterised using SPEC2000.
- Validated using SPEC2006.
- Measured using a custom-built data logger.



Platforms

1. Dell Latitude D600
2. IBM T41
3. AMD Opteron Server
4. Intel XEON Server
5. Gumstix
6. UNSW PLEB2
7. NICTA Ibox
8. Menlow
9. Asus EEEPC 901
10. Phycore iMX31

Ten more reasons to read the paper.

NICTA

- More hardware quirks.
- Empirical data from several platforms
- Parameter and model selection
- Experimental details
- Implementation details
- Multi tasking
- Frequency switch overheads
- Calculation overheads
- Higher level policies
- Practicality issues

Koala

- The commonly assumed models are **wrong**.
- Use **empirical models** to manage power.
- Use workload-agnostic policies.
- Characterised, tested and evaluated on lots of **real hardware**.

http://ertos.nicta.com.au

David.Snowdon@nicta.com.au

The idea with Koala is that if you can model how a system is likely to behave in various conditions, you can control it.

If you can build a model for your particular platform, Koala can control it. If that model encompasses the quirks of your platform, Koala will avoid the pitfalls and take advantage of the opportunities. You just need to build the model.

From **imagination** to **impact**

Saturday, 4 April 2009