NRG-Loops: Adjusting Power from within Applications

Melanie Kambadur*+, Martha Kim*

*Columbia University, New York, NY USA

+Oscar Health Insurance, New York, NY USA

Power efficiency evolution

Once, power/performance tradeoffs were set at HW design time…

- Less power
- Slower runtime

+ More power
+ Faster runtime

Power efficiency evolution

Once, power/performance tradeoffs were set at HW design time…

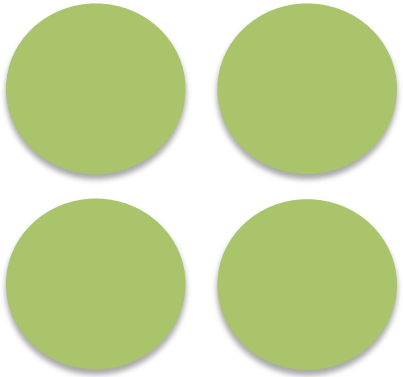
- Less power
- Slower runtime

+ More power
+ Faster runtime

+ Med. power
+ Faster runtime

Power efficiency evolution

Once, power/performance tradeoffs were set at HW design time…



- Less power
- Slower runtime

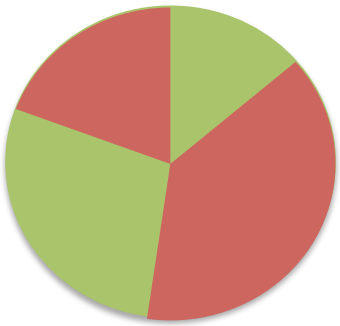
+ More power
+ Faster runtime

+ Med. power
+ Faster runtime

-- Power
++ Runtime

Power efficiency evolution

The next big thing was tunable "knobs"

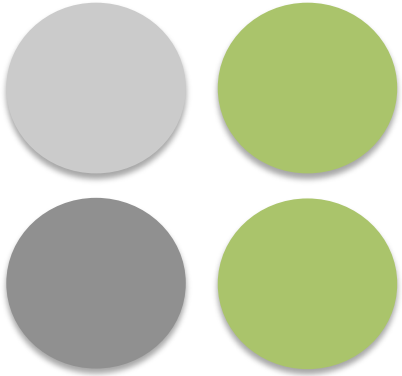


Dynamic Frequency Tuning (DFS/ DVFS)

Power efficiency evolution

The next big thing was tunable "knobs"

Dynamic Frequency Tuning (DFS/ DVFS)

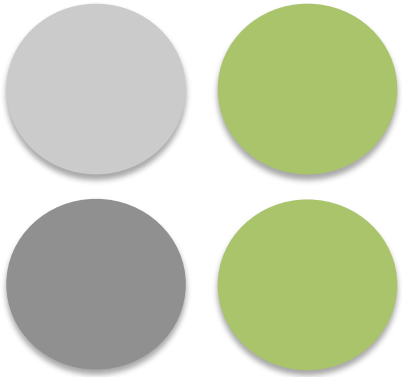


CPU Idle Modes

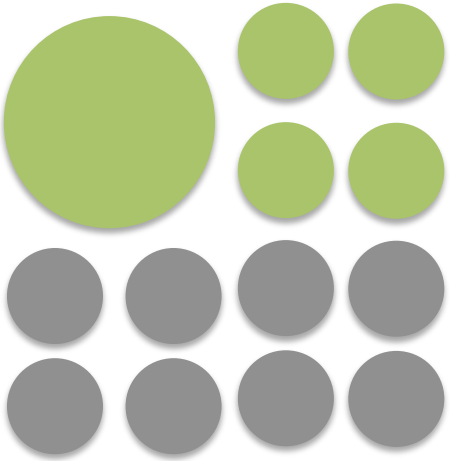
Power efficiency evolution

The next big thing was tunable "knobs"

Dynamic Frequency Tuning (DFS/ DVFS)



CPU Idle Modes

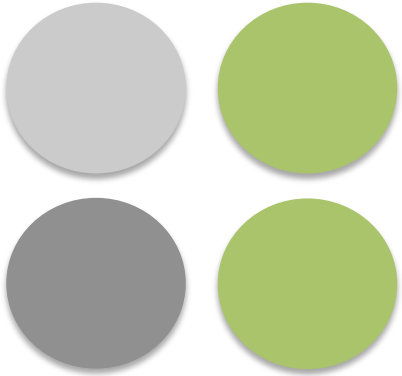


Asymmetric Multicores

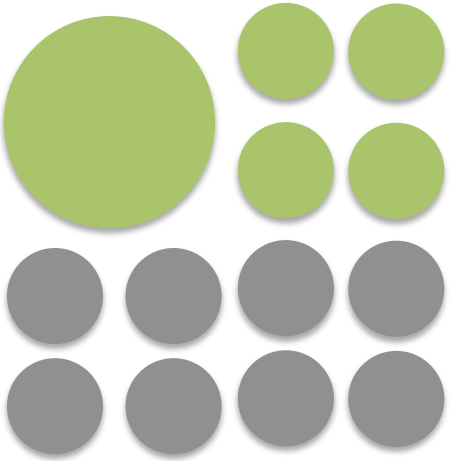
Moving power efficiency up the stack

How do we use these HW knobs for SW power & energy efficiency?

Dynamic Frequency Tuning (DFS/ DVFS)



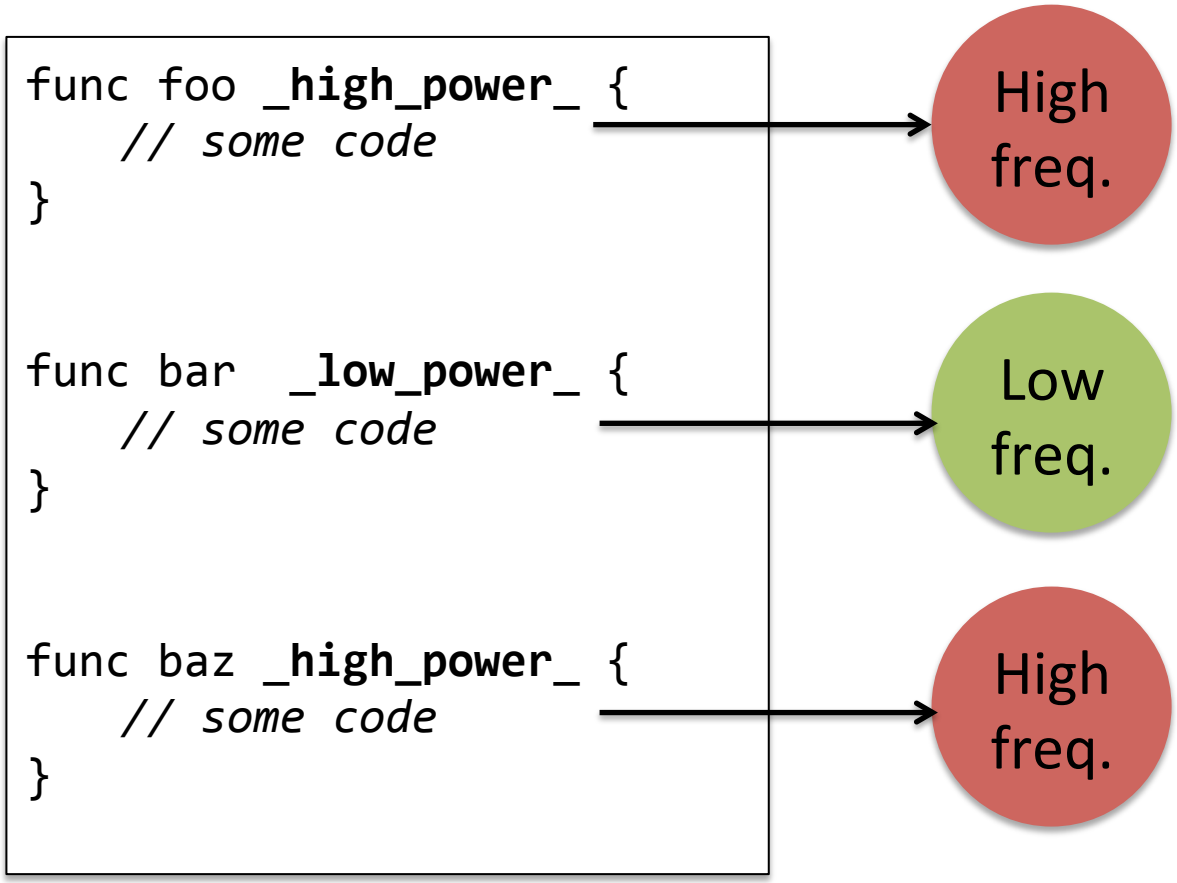
CPU Idle Modes



Asymmetric Multicores

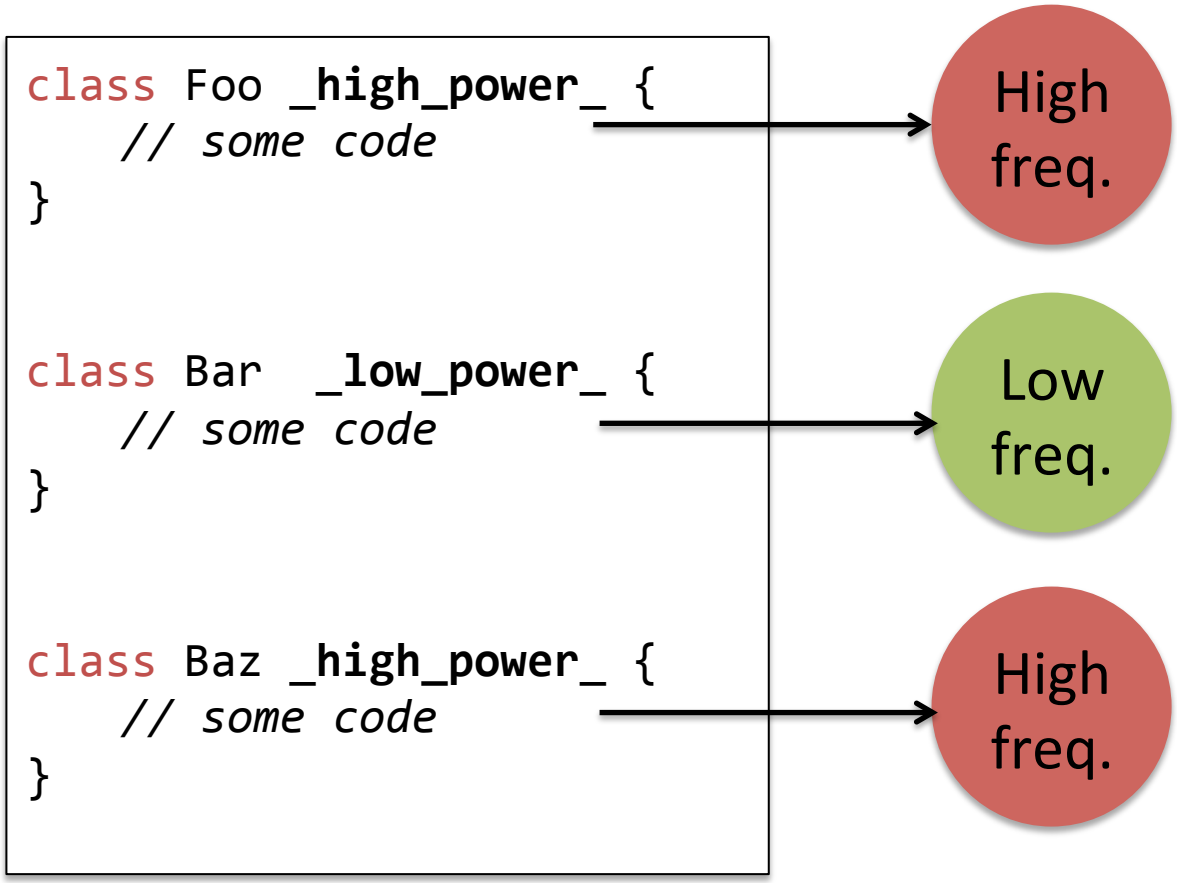
Using HW knobs for SW energy efficiency

Most SW energy efficiency solutions expose “hints” to OS, which then tunes HW knobs.



Using HW knobs for SW energy efficiency

Most SW energy efficiency solutions expose “hints” to OS, which then tunes HW knobs.



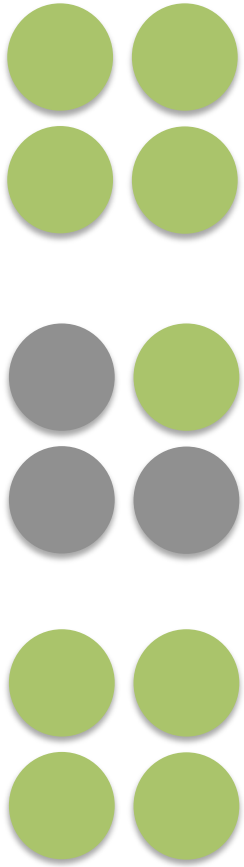
Using HW knobs for SW energy efficiency

Most SW energy efficiency solutions expose “hints” to OS, which then tunes HW knobs.

```
class Foo _high_power_ {
    // some code
}

class Bar  _low_power_ {
    // some code
}

class Baz _high_power_ {
    // some code
}
```



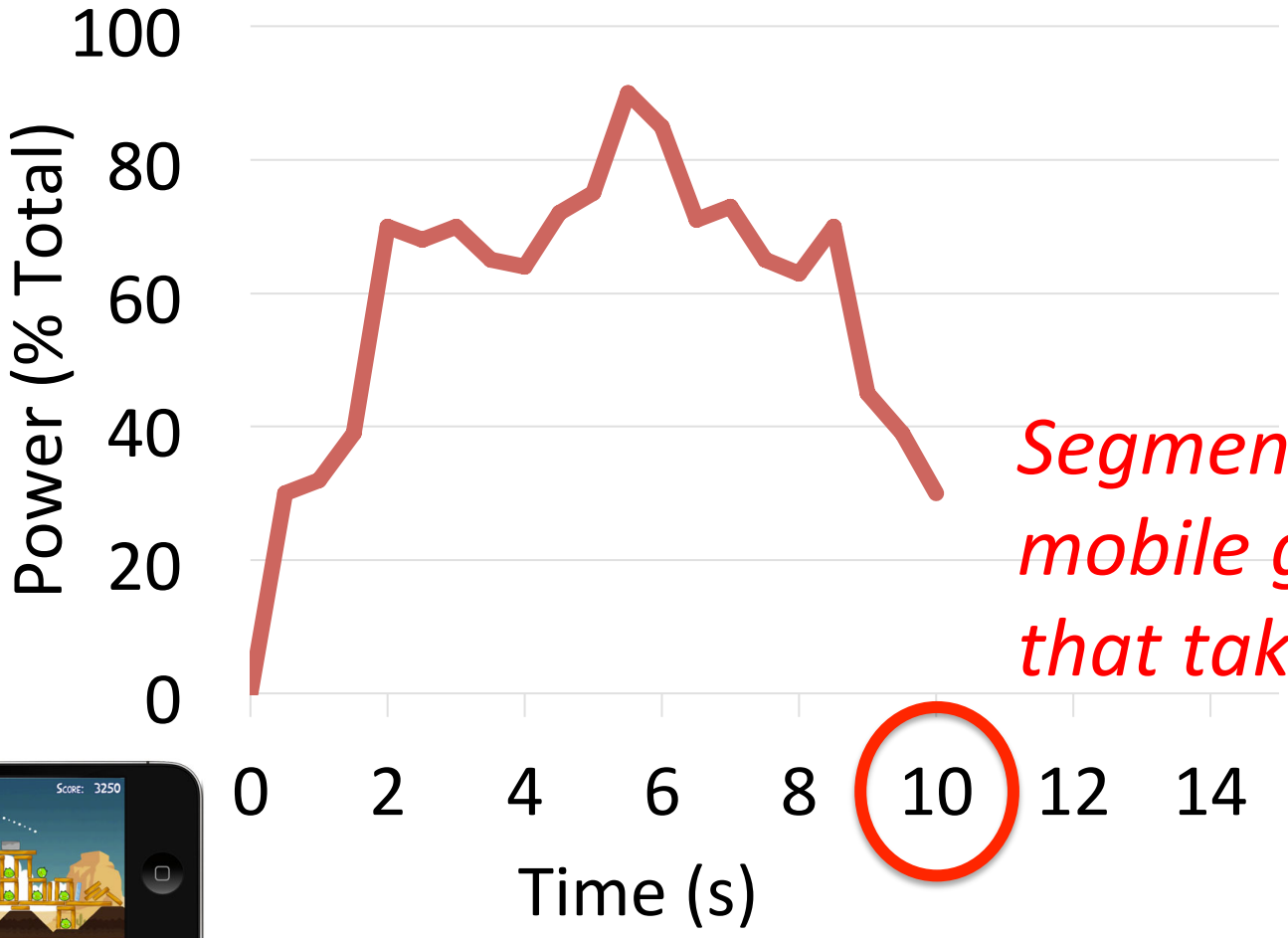
Idle some cores

STOP using HW knobs for SW energy efficiency

~~Most SW energy efficiency solutions expose "hints" to OS, which then tunes HW knobs.~~

- Hard to manage HW power when multiple programs give hints simultaneously
- HW can predict idle periods better→ sub-cycle DVFS tuning?
- In practice, most HW tuning increases runtime to save power, so can't save energy during SW active periods.

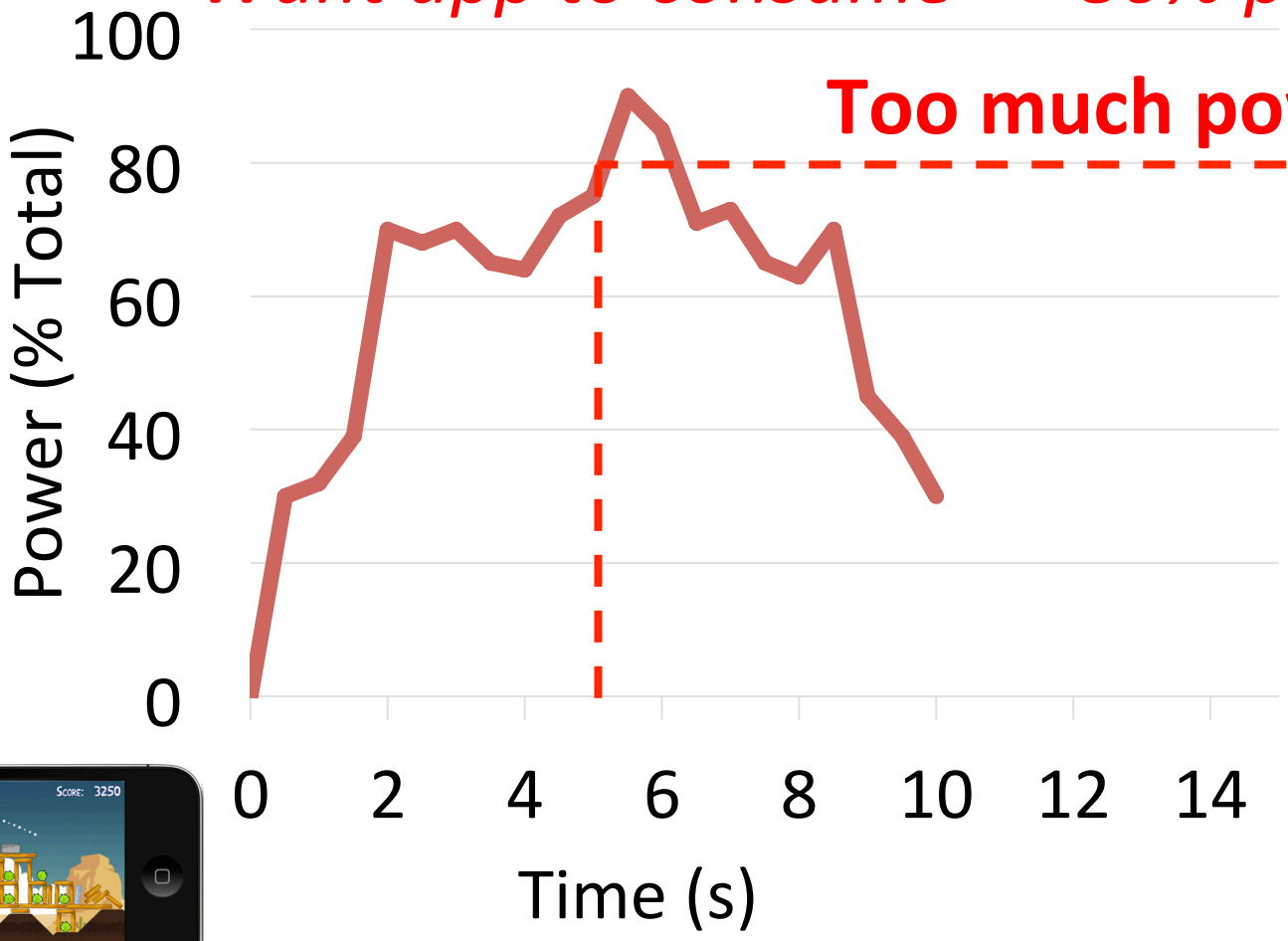
Mobile app example



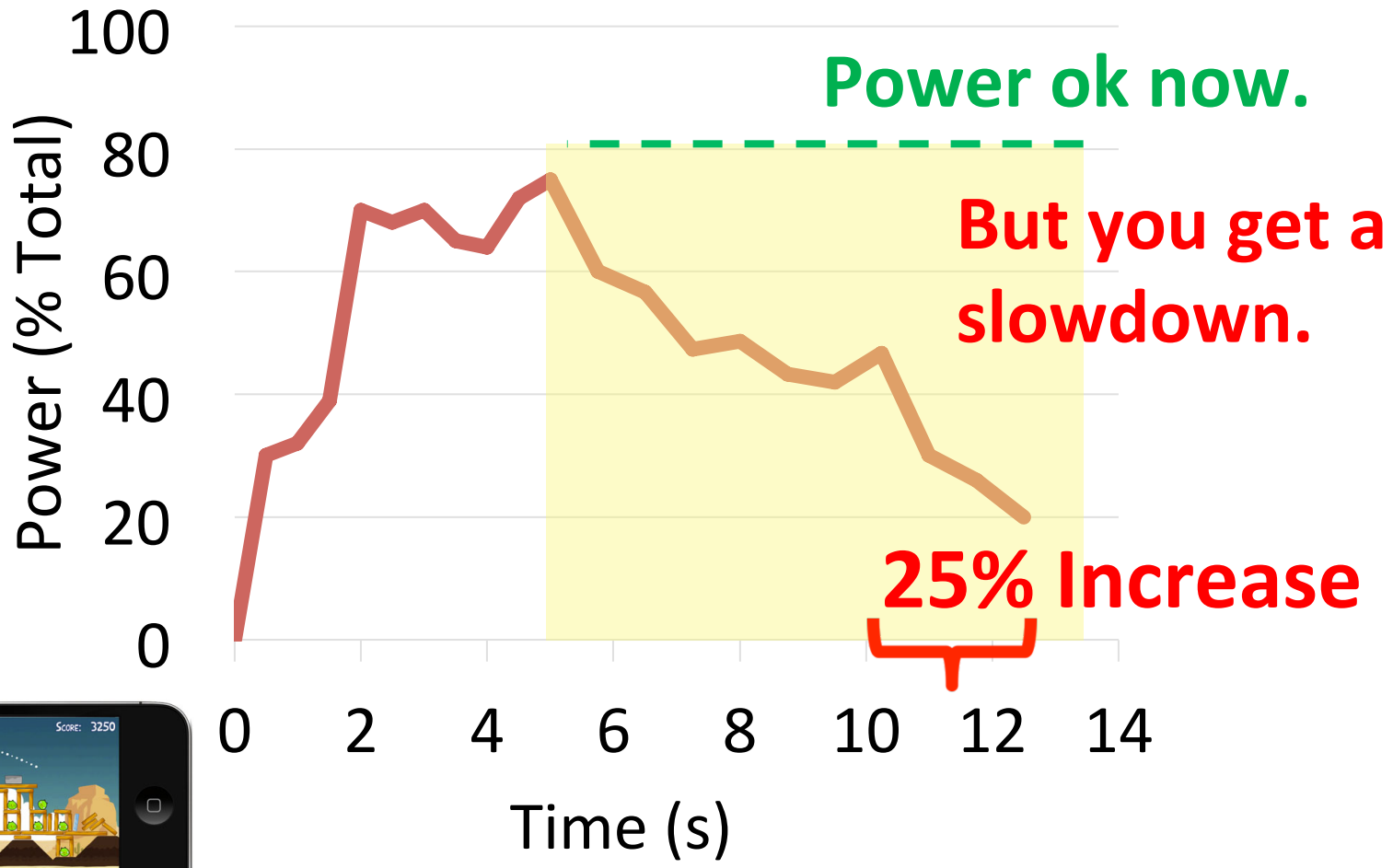
Segment of mobile game that takes 10s.

Mobile app example

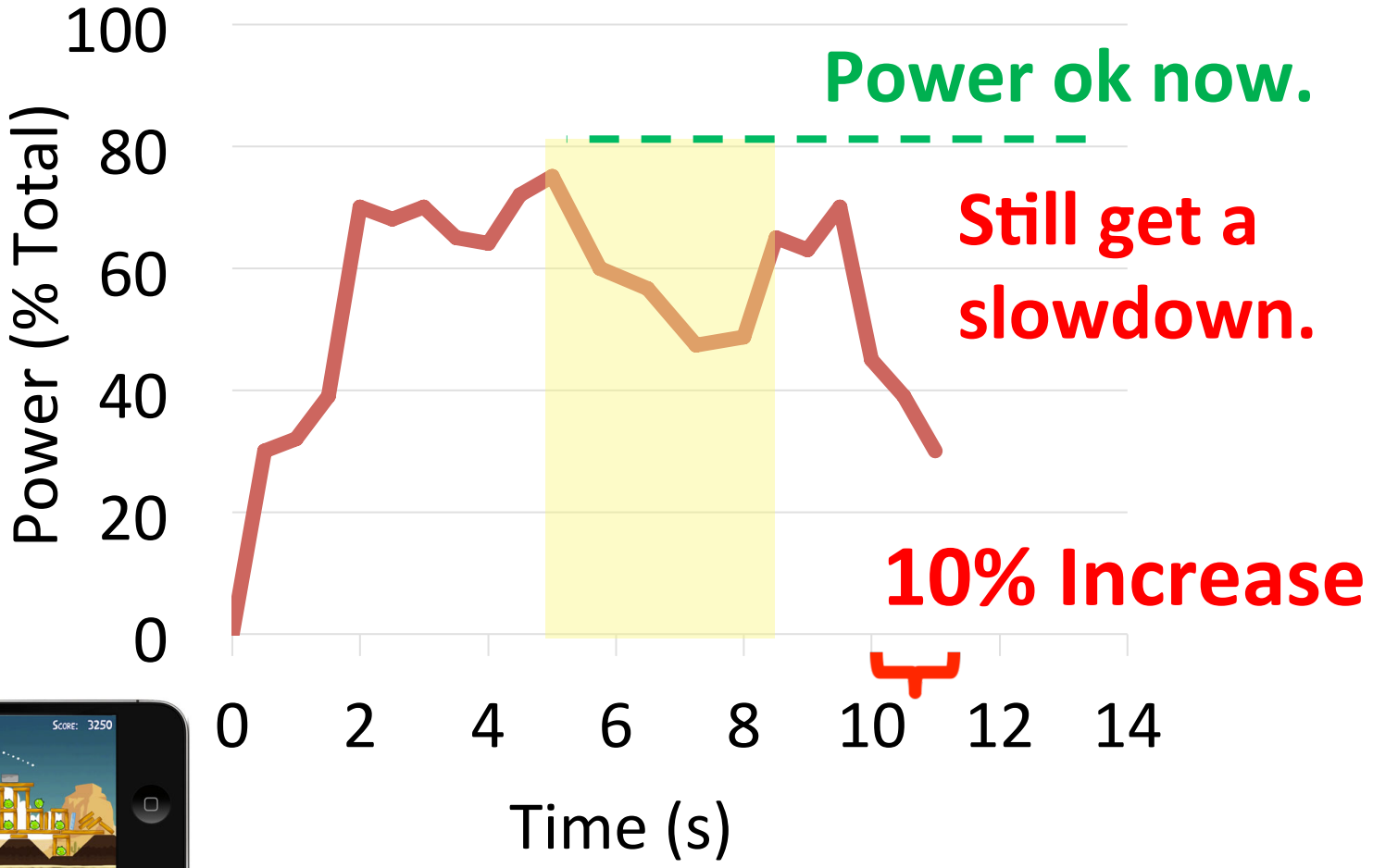
Want app to consume <= 80% power.



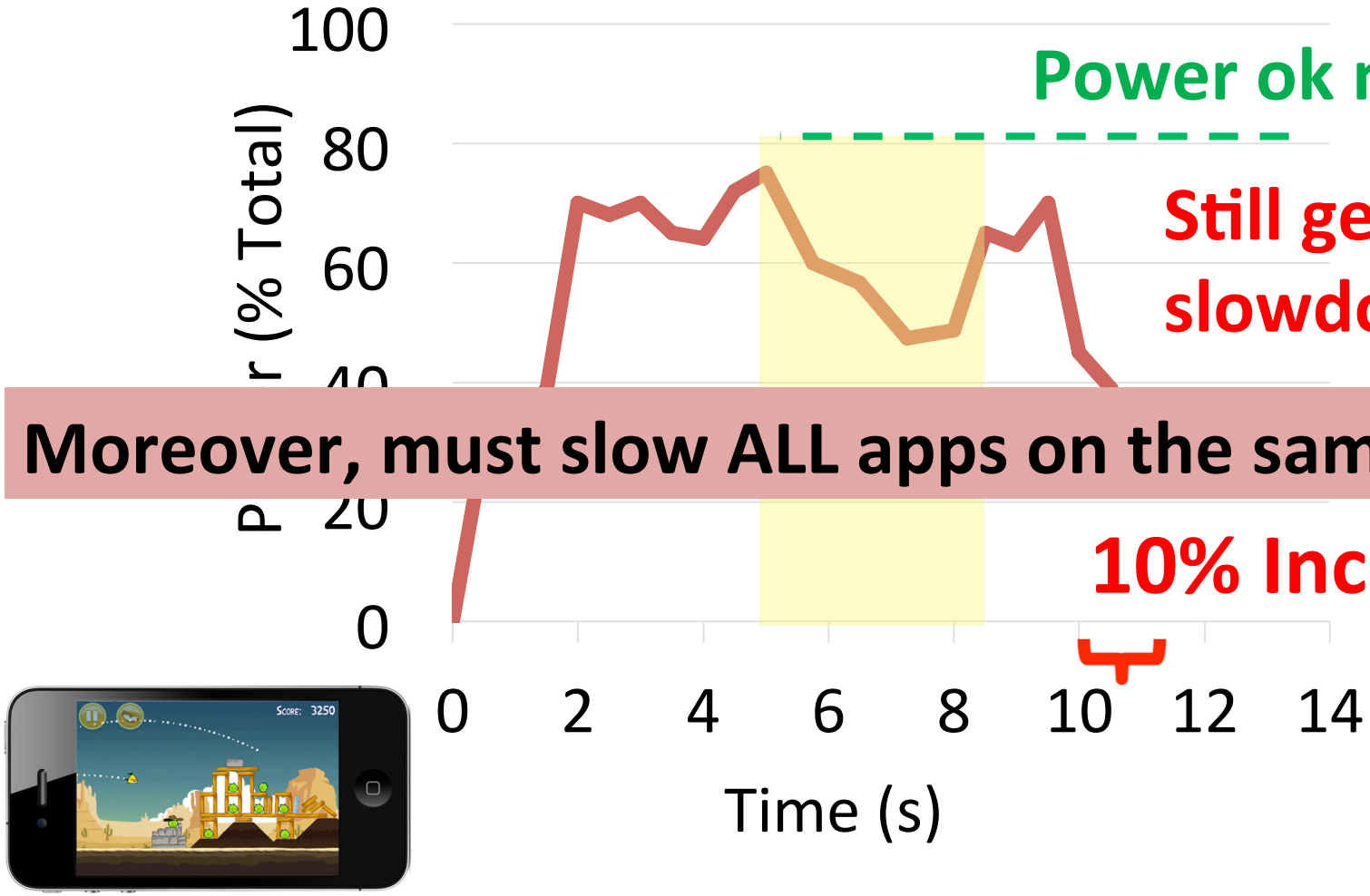
Option 1: Let HW handle with DVFS



Option 2: Compiler/Language Smart DVFS

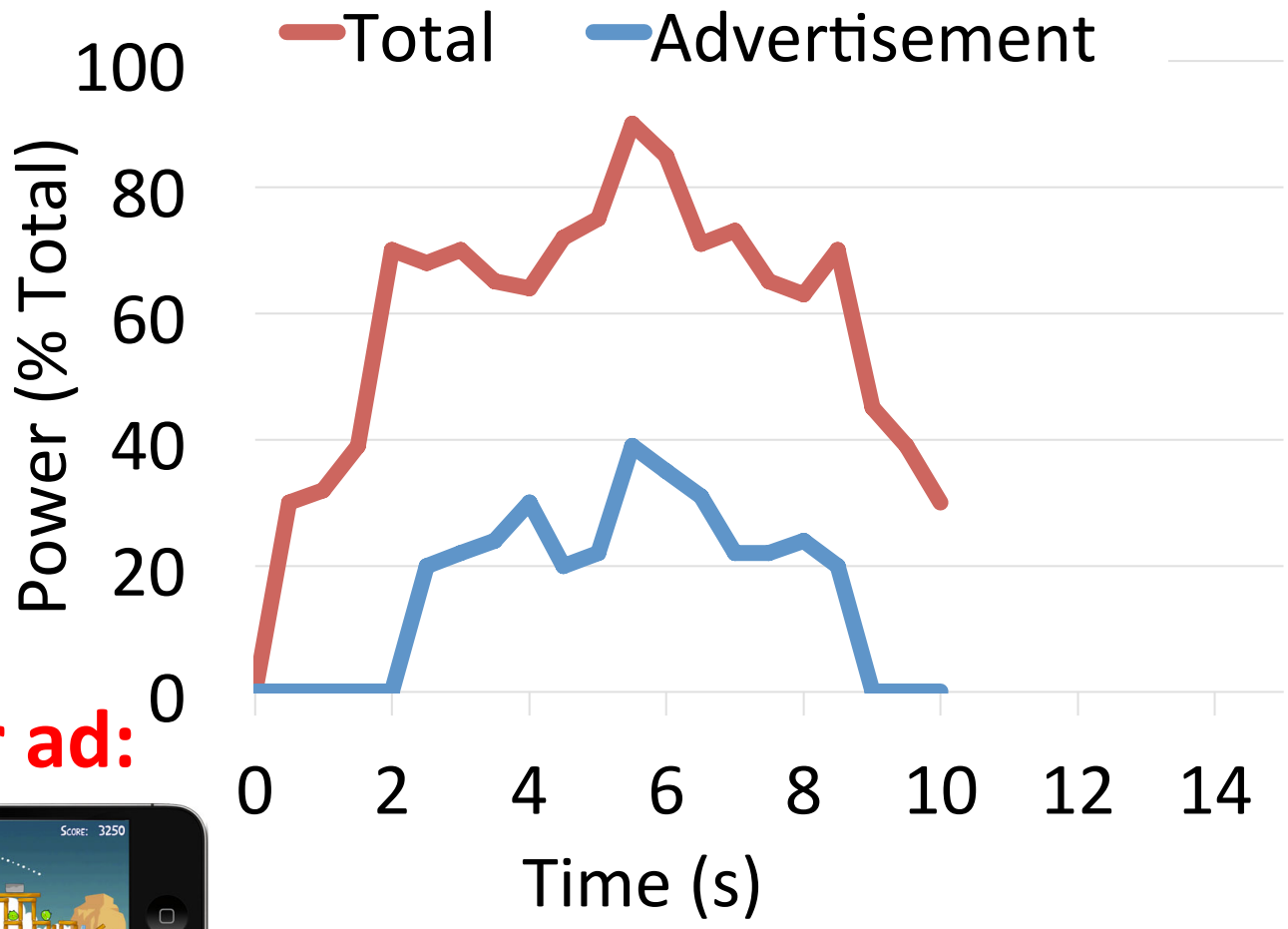


Option 2: Compiler/Language Smart DVFS



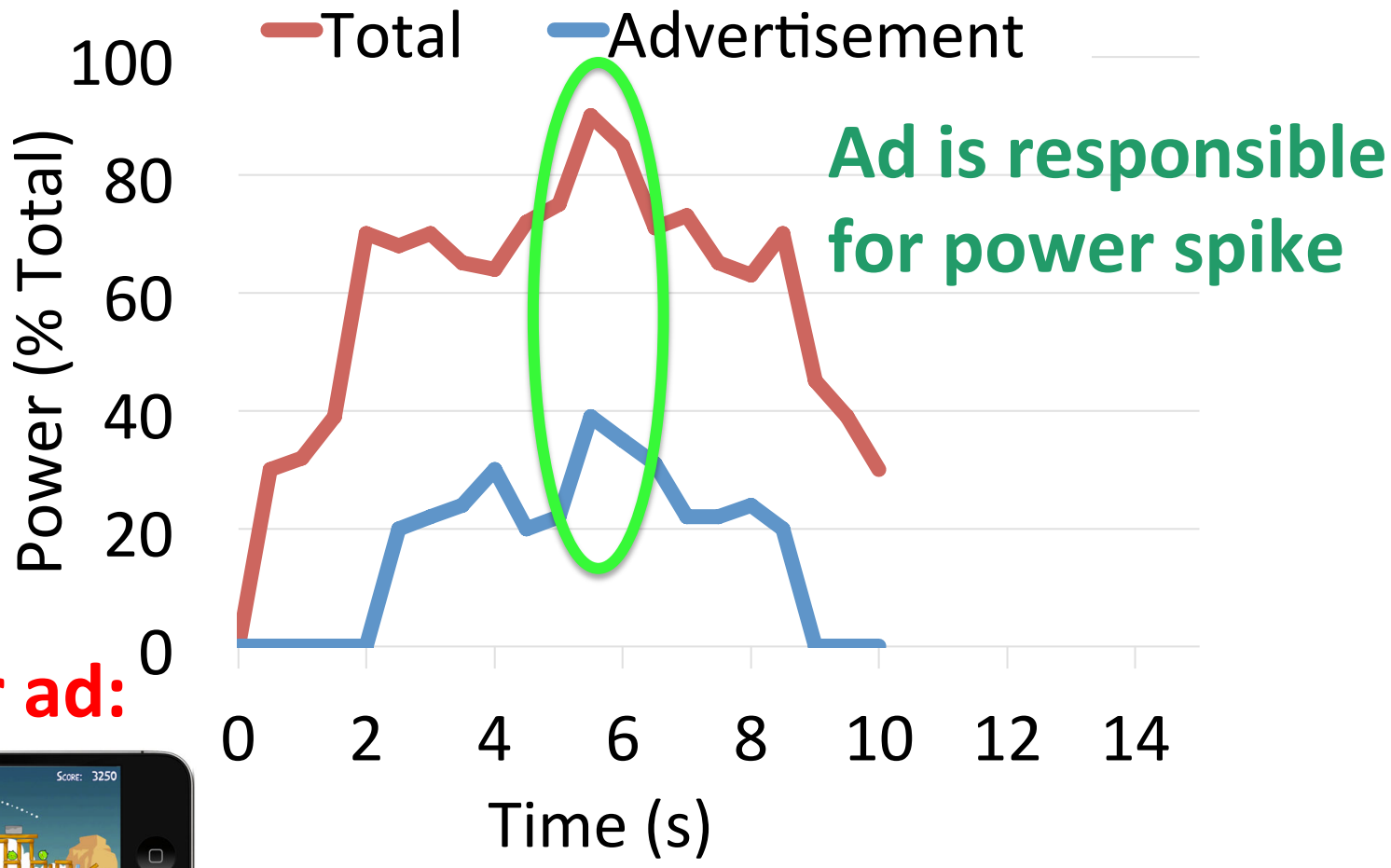
Moreover, must slow ALL apps on the same core.

Option 3: Trade functionality for power



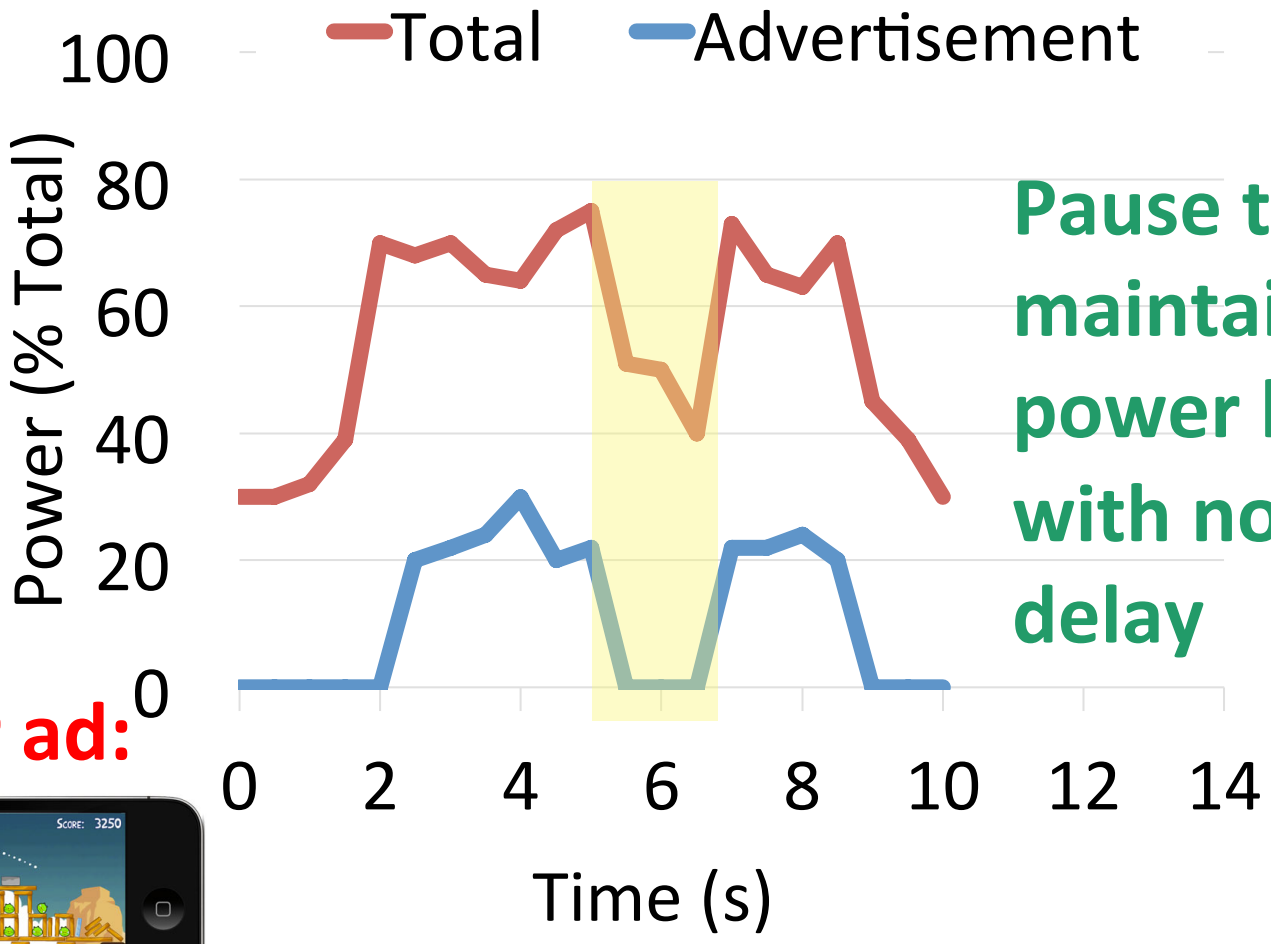
Banner ad:

Option 3: Trade functionality for power



Banner ad:

Option 3: Trade functionality for power



Pause the ad, maintain power budget with no time delay

Banner ad:

NRG-Loops: SW-Only Power Management

- Instead of having software manage power via hardware knobs, have software manage power via software knobs.

NRG-Loops: SW-Only Power Management

- Instead of having software manage power via hardware knobs, have software manage power via software knobs.

HW Knobs	SW Knobs
• DVFS • Idle cores • Asymmetric multicore	• Adjust caching strategy • Reduce thread count • Estimate mathematical function • Stop computation early and dump memory

NRG-Loops: SW-Only Power Management

- C++ Language Extension to tune SW power through SW knobs.
- Measures hardware power + energy and enables programs to trade functionality or accuracy ONLY when runtime power or energy budgets are exceeded.
- Can work concurrently with HW power solutions.

NRG-Loops ADAPT

Concise syntax adds only a few lines of code to existing programs

NRG-Loops ADAPT

Concise syntax adds only a few lines of code to existing programs

Loop pragma

```
NRG_ADAPT_FOR ( int i=0; i<MAX_ADS ; ++i &&
                    NRG_AVG_P<=POWER_LIMIT ) {
      // run ad normally
} NRG_ALTERNATE {
      usleep ( PAUSE_TIME );
}
```

NRG-Loops ADAPT

Concise syntax adds only a few lines of code to existing programs

Original loop bounds

```
NRG_ADAPT_FOR ( int i=0; i<MAX_ADS ; ++i &&
                NRG_AVG_P<=POWER_LIMIT ) {
    // run ad normally
} NRG_ALTERNATE {
    usleep ( PAUSE_TIME );
}
```

NRG-Loops ADAPT

Concise syntax adds only a few lines of code to existing programs

```
NRG_ADAPT_FOR ( int i=0; i<MAX_ADS ; ++i &&
                NRG_AVG_P<=POWER_LIMIT ) {
      // run ad normally
} NRG_ALTERNATE {
      usleep ( PAUSE_TIME );
}
```

Concatenate power/ energy goals

NRG-Loops ADAPT

Concise syntax adds only a few lines of code to existing programs

```
NRG_ADAPT_FOR ( int i=0; i<MAX_ADS ; ++i &&
                NRG_AVG_P<=POWER_LIMIT ) {
     // run ad normally
} NRG_ALTERNATE {
     usleep ( PAUSE_TIME );
}
```

Original loop body

NRG-Loops ADAPT

Concise syntax adds only a few lines of code to existing programs

```
NRG_ADAPT_FOR ( int i=0; i<MAX_ADS ; ++i &&
                NRG_AVG_P<=POWER_LIMIT ) {
      // run ad normally
} NRG_ALTERNATE {
      usleep ( PAUSE_TIME );
}
```

Enter if power budget exceeded

NRG-Loops ADAPT

Concise syntax adds only a few lines of code to existing programs

```
NRG_ADAPT_FOR ( int i=0; i<MAX_ADS ; ++i &&
                NRG_AVG_P<=POWER_LIMIT ) {
    // run ad normally
} NRG_ALTERNATE {
    usleep ( PAUSE_TIME );
}
```

Alternate, low-power loop body

Other types of NRG-Loops

```
NRG_TRUNCATE_FOR (int i=0; i<N; ++i &&
                  NRG_TOT_E <= FOO_ENERGY) {
  // original loop body
}
```

Do work until NRG Condition is met

Other types of NRG-Loops

```
NRG_TRUNCATE_FOR (int i=0; i<N; ++i &&
                  NRG_TOT_E <= FOO_ENERGY) {
  // original loop body
}

NRG_PROB_PERF_FOR (int i=0; i<N; ++i &&
                   NRG_TOT_E <= FOO_ENERGY;
                   PROB_SKIP=0.1) {
   // original loop body
}
```

Once condition met, do work 9/10 times

Other types of NRG-Loops

```
NRG_TRUNCATE_FOR (int i=0; i<N; ++i &&
                  NRG_TOT_E <= FOO_ENERGY) {
  // original loop body
}

NRG_PROB_PERF_FOR (int i=0; i<N; ++i &
                  NRG_TOT_E <= FOO_EN
                  PROB_SKIP=0.1) {
   // original loop body
}

NRG_AUTO_PERF_FOR (int i=0; i<N; ++i &&
                   NRG_TOT_E <= FOO_ENERGY) {
   // original loop body
}
```

Do work, skipping an estimated number of iterations to exactly match FOO_ENERGY

NRG-Loop Helpers

```
NRG_AUDIT {
    foo()
    // any code here
} NRG_USAGE (NRG_USAGE_INFO* foo_usage);

float foo_energy = foo_usage->energy;
float foo_average_power = foo_usage->average_power;
float foo_wall_time = foo_usage->wall_time;
```

Capture the energy, power, and runtime use of any code.

NRG-Loop Helpers

```
NRG_AUDIT {
    foo()
    // any code here
} NRG_USAGE (NRG_USAGE_INFO* foo_usage);

float foo_energy = foo_usage->energy;
float foo_average_power = foo_usage->average_power;
float foo_wall_time = foo_usag

NRG_AVG_P <= 50.0  // Watts
NRG_TOT_E <= foo_energy // Relative to foo()
NRG_AVG_P <= 0.5*SYS_MAX_POWER // Relative to TDP
```

Different options to set energy/power budgets.

NRG-Loops Implementation

- No adjustments to the O/S, used commodity HW with built in power *meters* (Intel RAPL)
  - Measure CPU + cache power + estimate DRAM power

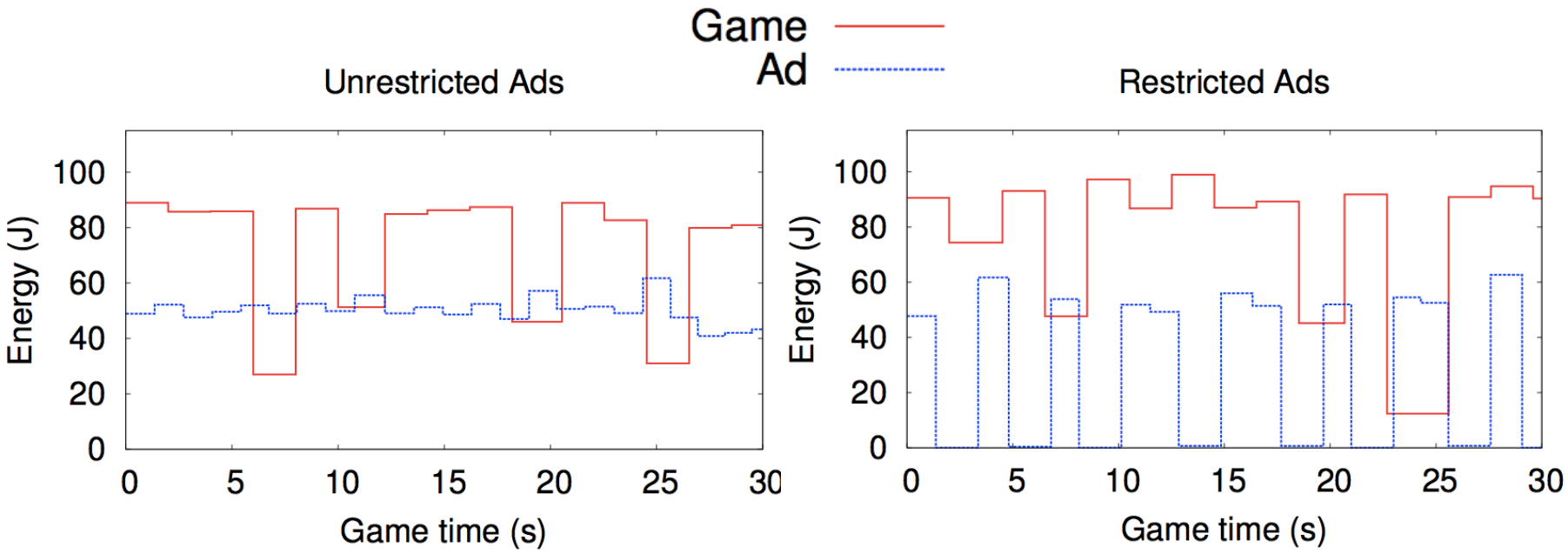
NRG-Loops Implementation

- No adjustments to the O/S, used commodity HW with built in power *meters* (Intel RAPL)
  - Measure CPU + cache power + estimate DRAM power
- Implementation *should have* been trivial, but unfortunately wasn't
  - Socket level power meters make attributing power to processes tricky
  - Small, overflowing counters
  - To minimize monitoring overhead we have one monitoring thread even for multithreaded programs

NRG-Loops Implementation

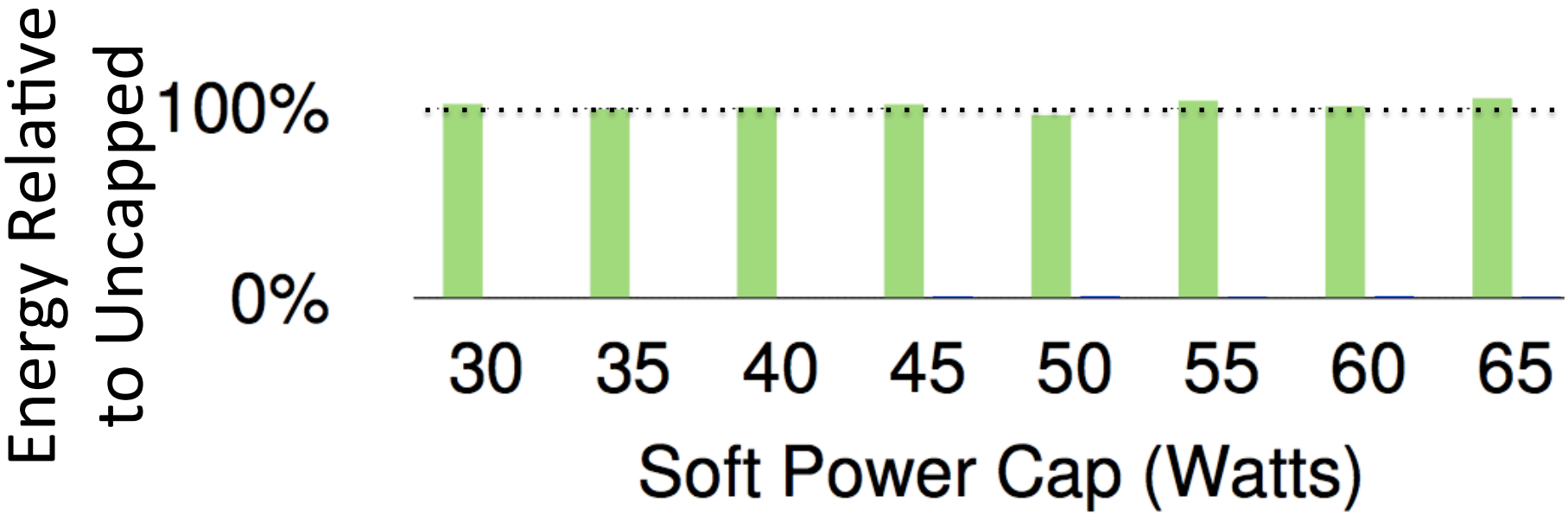
- No adjustments to the O/S, used commodity HW with built in power *meters* (Intel RAPL)
  - Measure CPU + cache power + estimate DRAM power
- Implementation *should have* been trivial, but unfortunately wasn't
  - Socket level power meters make attributing power to processes tricky
  - Small, overflowing counters
  - To minimize monitoring overhead we have one monitoring thread even for multithreaded programs
- Profiling power & energy goals adds <1% overhead.

Results: NRG_ADAPT Minesweeper



When advertisements are consuming too much energy, force them to occasionally pause, decreasing net game plus ad energy.

Results: NRG_ADAPT Parallel Programs



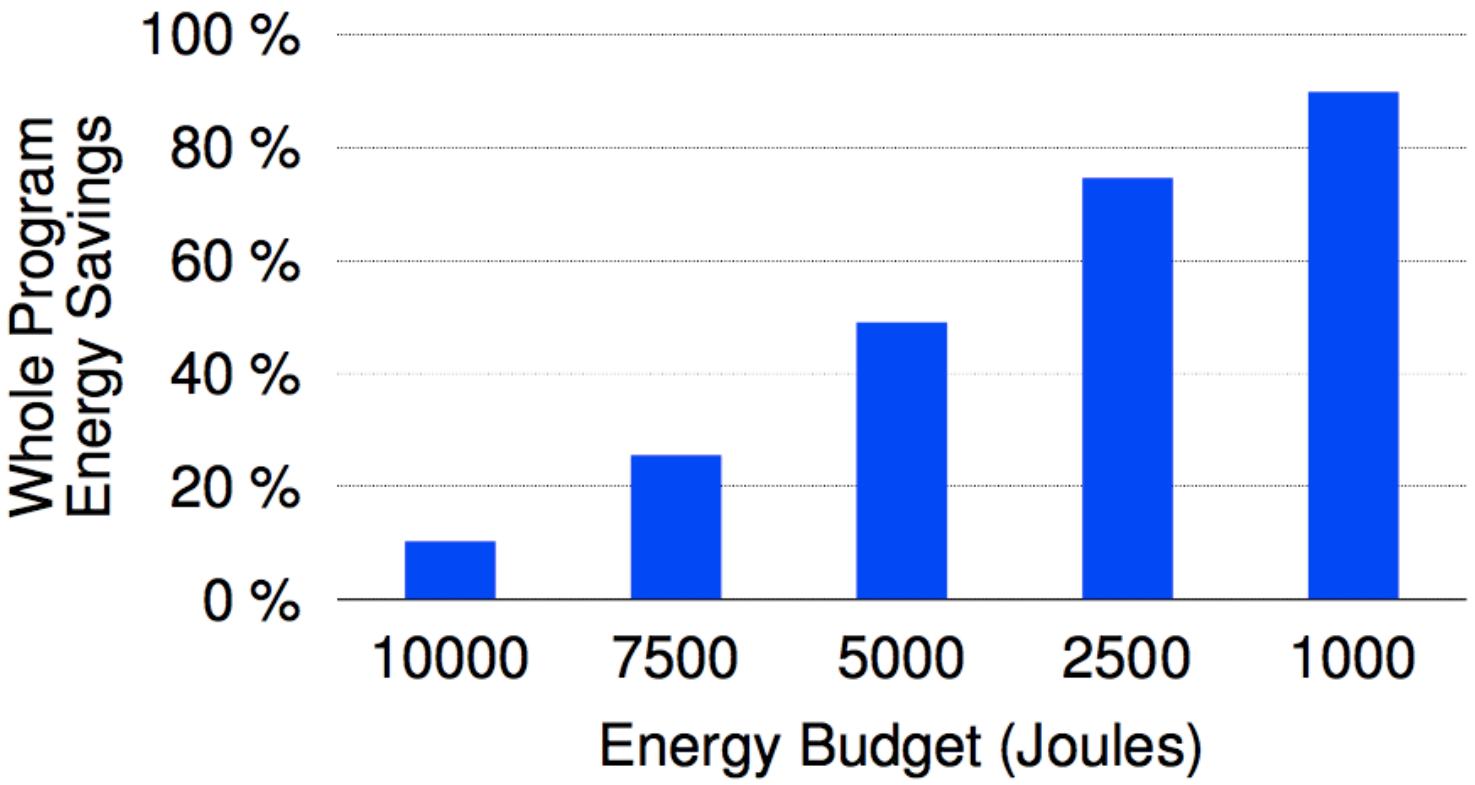
Reduce software thread count to keep within a power budget.

Questions?

- Melanie Kambadur (melaniekambadur@gmail.com)
- Martha Kim (martha@cs.columbia.edu)

NRG-Loops: Adjusting Power from within Applications

Results: NRG_PERFORATE



Set a target energy budget and drop frames to meet it.