Open Source Software

EAS in Android Common Kernel

Linaro Connect

Hong Kong 2018

Chris Redpath

Quentin Perret

EAS in Android Common Kernel

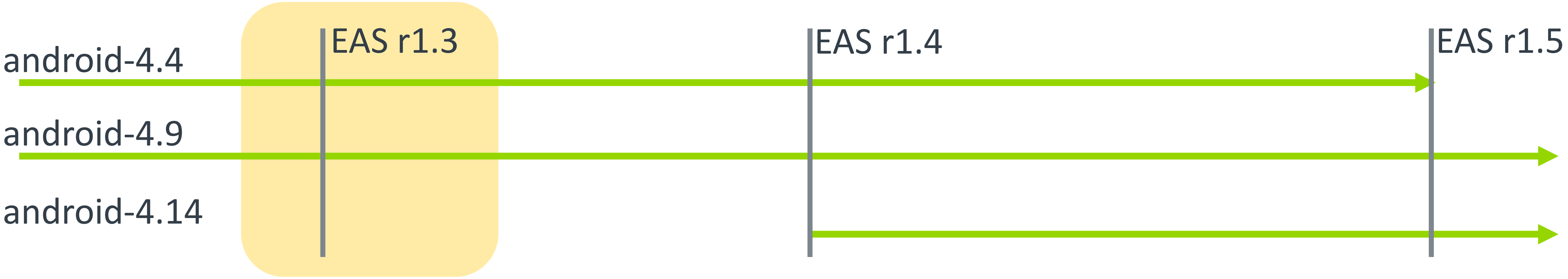
AOSP Common Kernel Update

EAS Mainline Strategy

EAS Upstreaming

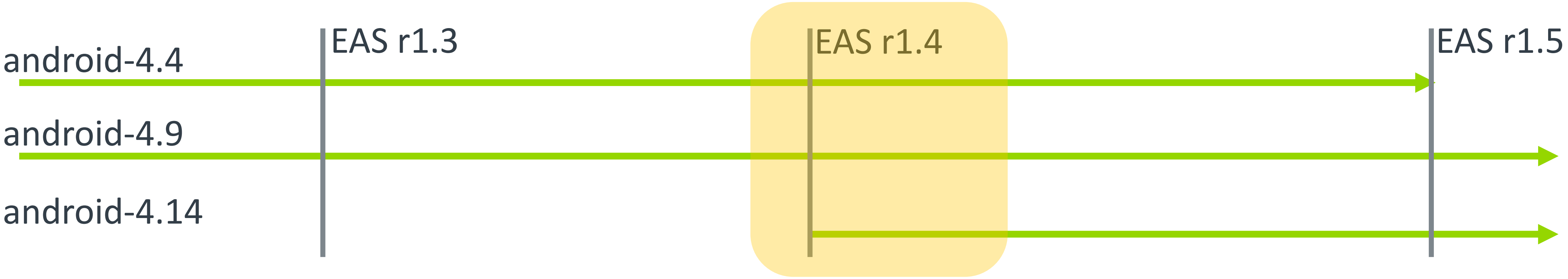
AOSP Common Kernel Update

AOSP Common Kernel Update



- EAS r1.3, July 2017
  - android-4.4, android-4.9
  - Default cpufreq governor switched to schedutil, sched-freq removed
  - Backports of upstream schedutil changes
  - Upstream backports of relevant scheduler features

AOSP Common Kernel Update



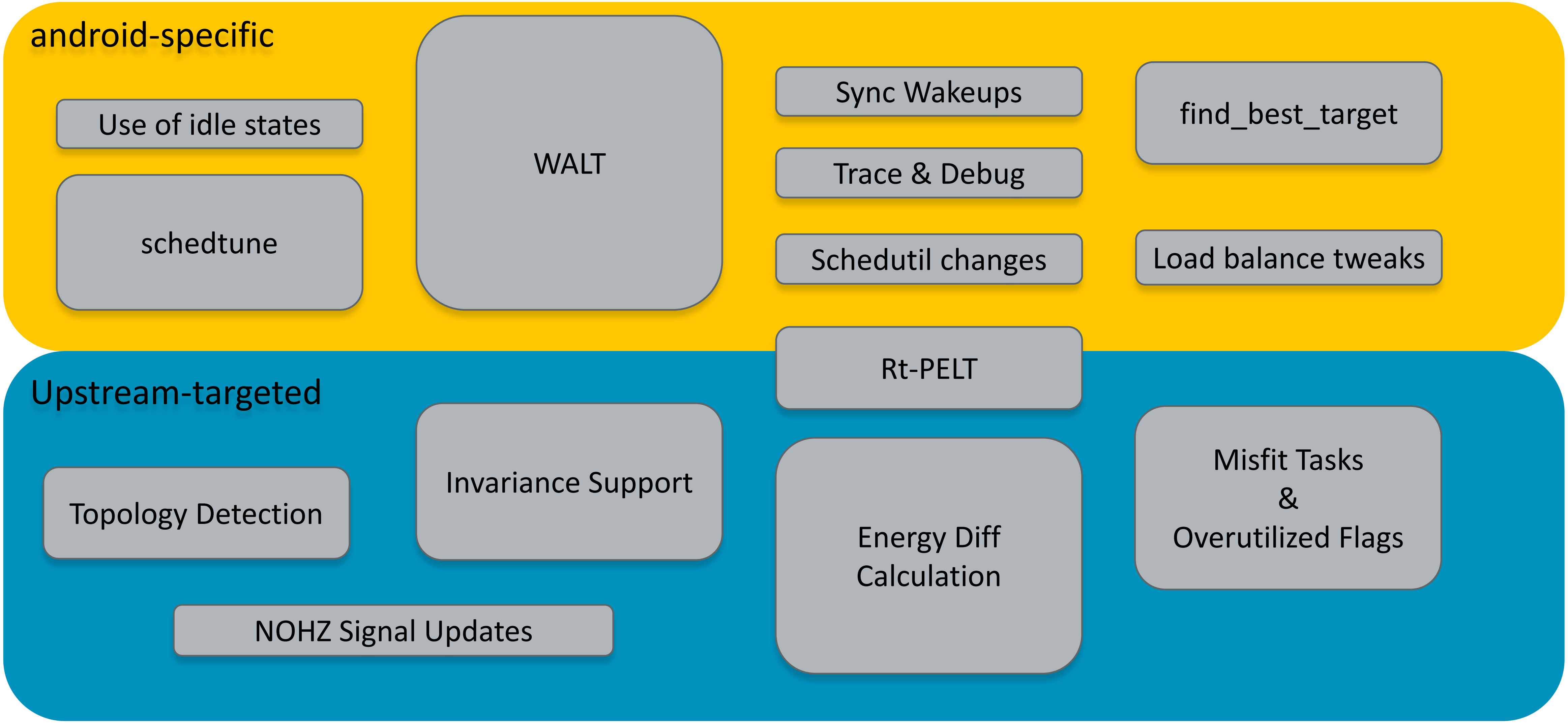
- EAS r1.4, November 2017
  - android-4.4, android-4.9
  - Upstream backports of more scheduler and schedutil patches
  - Energy diff improvements & fixes
  - android-4.14 EAS released including 1.4 & most 1.5 functionality

EAS in android-4.14

A new set of patches implementing EAS rather than forward-porting

- Based upon our latest mainline-focussed integration branch
- Refactored latest android-eas on top to build clean set of patches
- More Experimental features placed behind sched_features
  - Feature configuration matches android-4.9
- Produced during linux-4.14 rc phase, ready 2-weeks after linux-4.14

EAS in android-4.14



android-specific

Use of idle states

schedtune

WALT

Sync Wakeups

Trace & Debug

Schedutil changes

find_best_target

Load balance tweaks

Rt-PELT

Upstream-targeted

Topology Detection

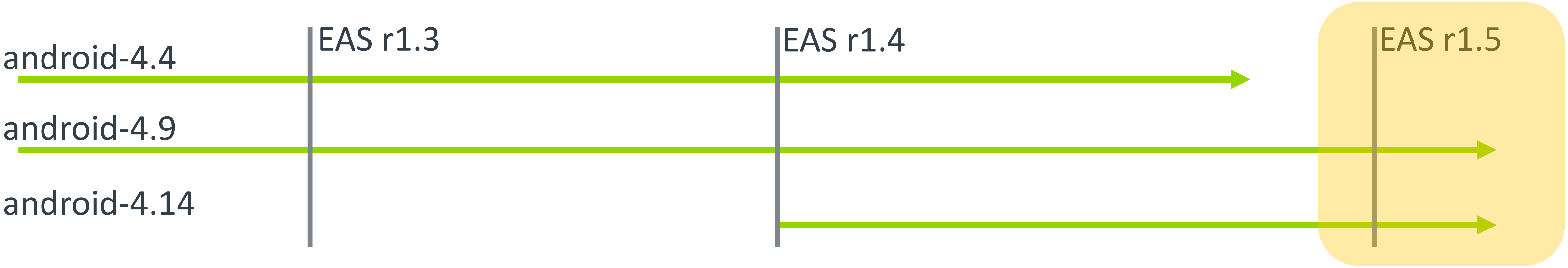
Invariance Support

NOHZ Signal Updates

Energy Diff Calculation

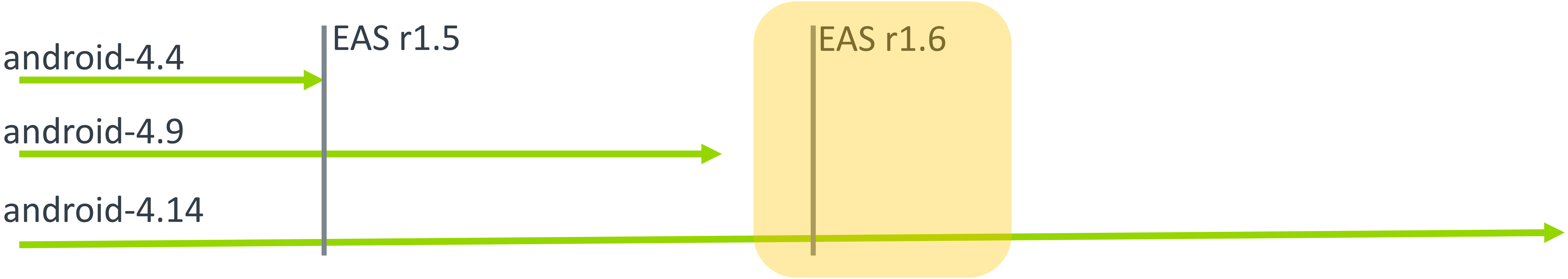
Misfit Tasks & Overutilized Flags

AOSP Common Kernel Update



- EAS r1.5, Feb 2018 (eas-dev), merging to android-4.9 soon
  - android-4.9 only, most changes already in android-4.14
  - Refactored energy diff to make calculation more efficient
  - Further refinement of EAS CPU pre-selection (find_best_target)
    - Thanks for excellent contributions from Qualcomm, Spreadtrum, Mstar, Linaro
  - Aggressive up-migrate of Misfit tasks & WALT updates from CodeAurora

AOSP Common Kernel Update



- EAS r1.6, eas-dev starting April 2018
  - Moving to android-4.14
  - Adding back Schedtune PE space filtering
  - Util_est backport, with PELT decay rate changes
  - Use mainline wakeup code for prefer_idle tasks
  - Remove ordering dependency in find_best_target
    - (better tri-gear support when using find_best_target)

AOSP Common Kernel Update

Branches:

- android-4.4, android-4.9 & android-4.14
  - Common kernel upstream for device kernels
  - Only post against this for bugfixes
  - People merge these into device kernels, so need to be selective about changes

AOSP Common Kernel Update

More branches:

- android-4.9-eas-dev (soon android-4.14-eas-dev)
  - This is where in-development patches should be posted
  - Arm power team usually post patches at RFC stage to stimulate discussion
  - Changes picked or merged back to common
- android-4.4-eas-test (android-4.9-eas-test later)
  - Test branch is against android common for the latest well-supported public device
  - Intended to hold backports of EAS patches which merged into the active common branch, but did not get back to the branch we test with

AOSP Common Kernel Update

There have been some consistent themes in EAS development over the last year or so:

AOSP Common Kernel Update

There have been some consistent themes in EAS development over the last year or so:

- Reducing delta with mainline

AOSP Common Kernel Update

There have been some consistent themes in EAS development over the last year or so:

- Reducing delta with mainline
- Refactoring to improve maintainability and predictability

AOSP Common Kernel Update

There have been some consistent themes in EAS development over the last year or so:

- Reducing delta with mainline
- Refactoring to improve maintainability and predictability
- New features where necessary

AOSP Common Kernel Update

There have been some consistent themes in EAS development over the last year or so:

- Reducing delta with mainline
- Refactoring to improve maintainability and predictability
- New features where necessary
- Open, collaborative development

AOSP Common Kernel Update

Open Development

- Patches for AOSP are reviewed on AOSP Gerrit
  - https://android-review.googlesource.com
  - We always try to justify patches with performance & energy numbers – use wltests for this
  - Wltests is part of LISA https://www.github.com/arm-software/lisa

- Discussion of other topics and announcements are on Linaro's eas-dev list
  - https://lists.linaro.org/mailman/listinfo/eas-dev

EAS Mainline Strategy

EAS Mainline Strategy

- EAS is a large, complex piece of functionality

- EAS being in AOSP helps a lot of users but not all

- Upstream development results in better code

EAS Mainline Strategy

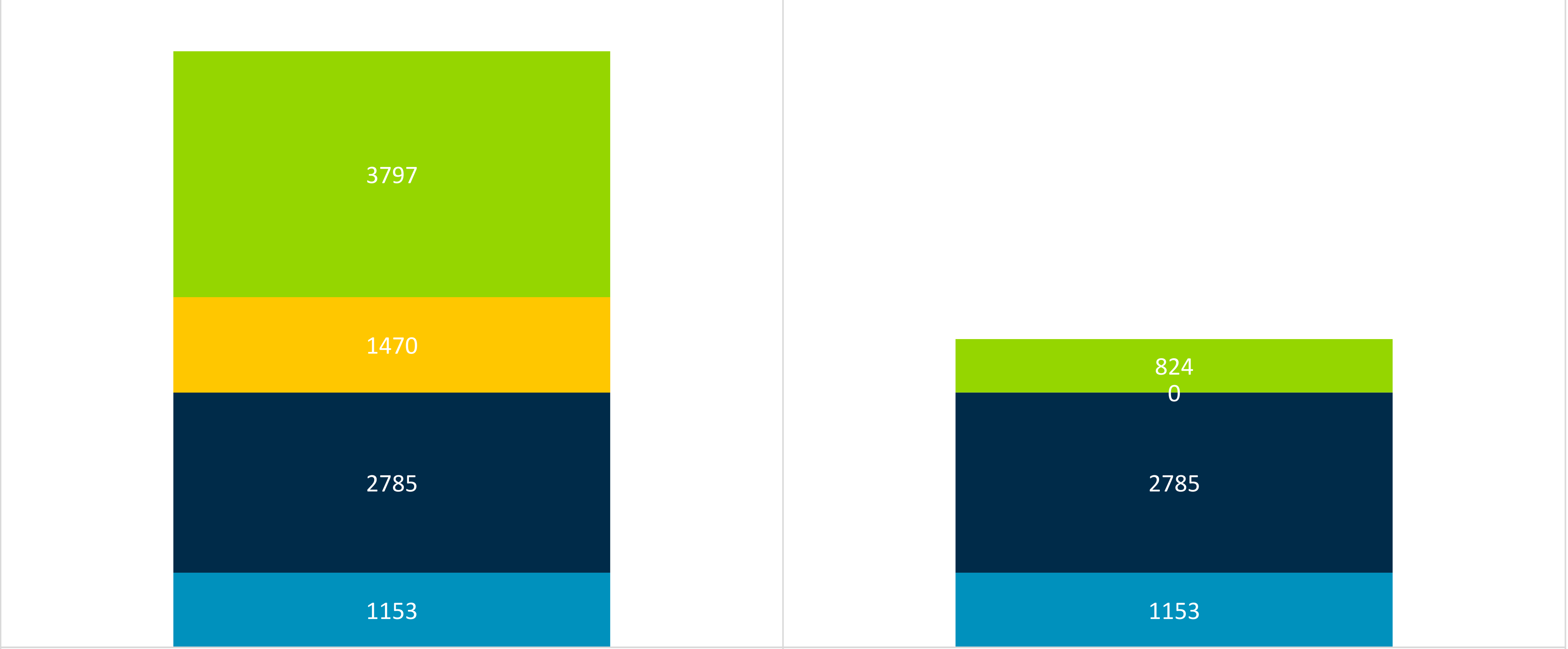
- We make regular bi-weekly integrations of all our upstream-focussed code
  - Available on linux-arm.org & announced on eas-dev
  - Allows us to more easily see when changes impact us and work to resolve as soon as possible

- Have been identifying suitable code we already have
  - Working on getting them into acceptable shape
  - Pushing when we think they are good enough for a review
  - Hoping to upstream quite a lot of EAS this year

EAS Mainline Strategy

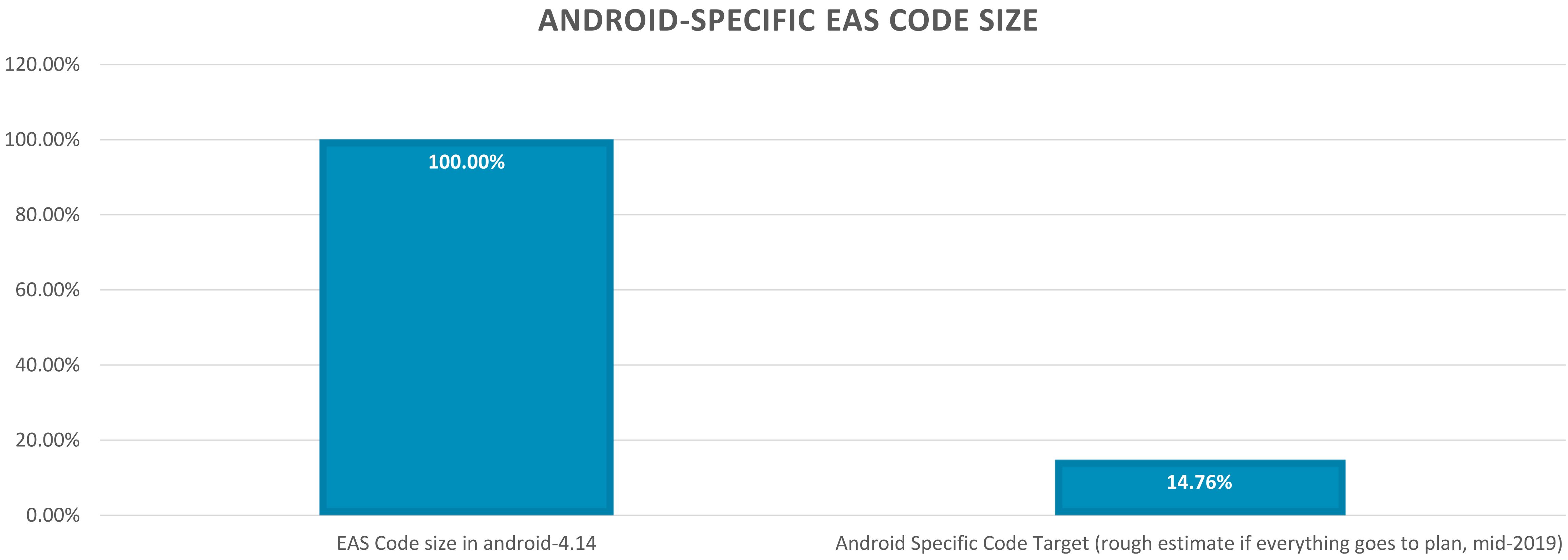
- Also working upstream where we can and backporting to Android
  - schedutil fixes
  - cpu signal updates
  - any fix/change applicable and potentially useful elsewhere
  - participating in reviews and testing

EAS in AOSP



	EAS Code Size by Category	Target EAS Code Size by Category
🟩 Android-specific	3797	824
🟨 WALT	1470	0
⬛ Upstreamable Features	2785	2785
🟦 Documentation	1153	1153

EAS Size


ANDROID-SPECIFIC EAS CODE SIZE
120.00%
100.00%
80.00%
60.00%
40.00%
20.00%
0.00%
100.00%
14.76%
EAS Code size in android-4.14
Android Specific Code Target (rough estimate if everything goes to plan, mid-2019)

Bringing EAS in AOSP Closer to Mainline

1. **Reach performance/energy parity with WALT**
   - WALT is great for mobile but not popular upstream
   - It’s also 1.5k LoC
   - Touches many parts of the scheduler we want to change upstream, which makes backporting harder

Bringing EAS in AOSP Closer to Mainline

1. **Reach performance/energy parity with WALT**
   - Disable WALT by default in android-common when ready
2. **Push better support for big.LITTLE into mainline scheduler**
   - Push out-of-tree wakeup and periodic balance changes upstream
   - Push energy diff calculations upstream

Bringing EAS in AOSP Closer to Mainline

1. **Reach performance/energy parity with WALT**
   - Disable WALT by default in android-common when ready
2. **Push better support for big.LITTLE into mainline scheduler**
   - Push out-of-tree wakeup and periodic balance changes upstream
   - Push energy diff calculations upstream
3. **Expect to continue to carry mobile-specific changes in AOSP**
   - Schedutil up/down throttle split
   - Rt-rq signals
   - Performance/Energy task classification

EAS Upstreaming

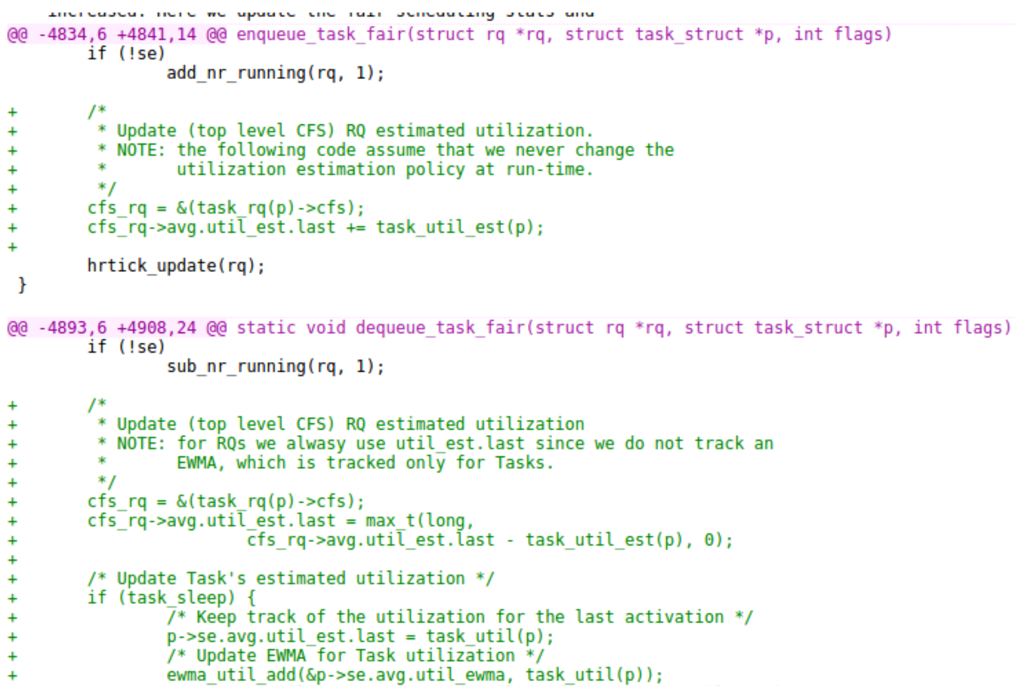
EAS Upstreaming

7 areas identified for upstreaming.

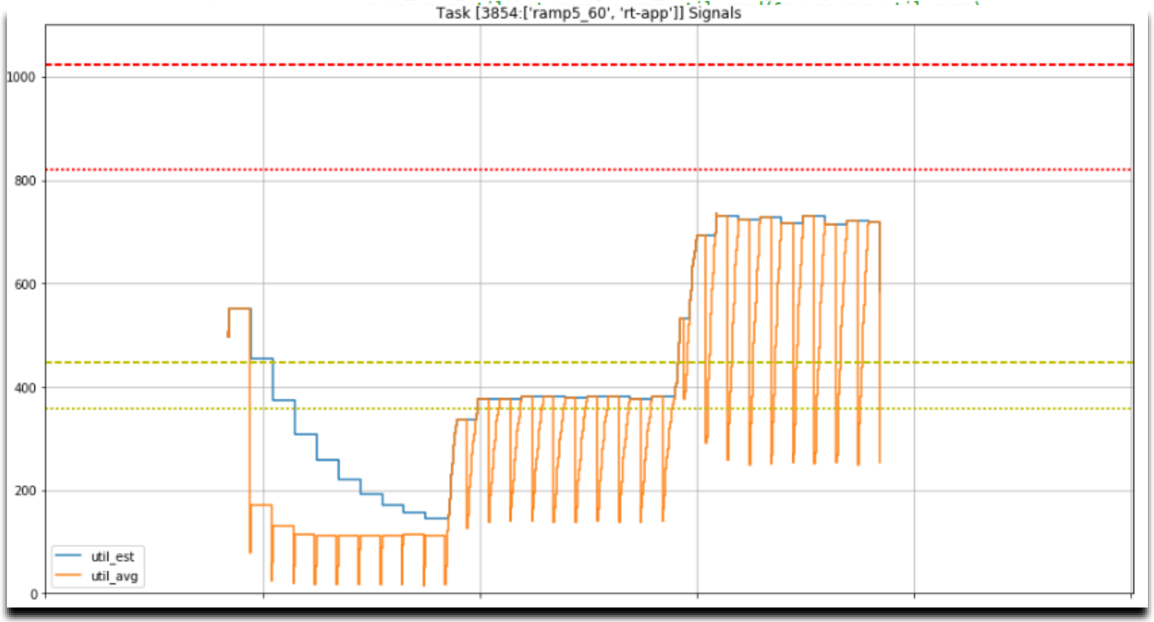
Feature	Status
Energy Model	On LKML (v1 March 2018, during Connect!)
Frequency and Cpu Invariant Engines (FIE/CIE)	Merged in v4.15
Idle Cpu PELT update (Remote status update)	Merged in tip/sched/core
Util Est	Merged in tip/sched/core (during Connect!)
Util Clamp	Almost ready (v1 on LKML April 2018)
Misfit Task	On LKML (v2 March 2018)
Dynamic Topology Flag Detection	In development, many scenarios to cover

EAS Upstreaming

- Util-Est
  - Add an **aggregator** on top of the PELT **estimator**
    - keep track of what “we learned” about task’s previous activations
    - generate a “new” signal on top of PELT
  - Build a **low-overhead statistic** for SEs and CPUs
    - Tasks at dequeue time
    - Root RQs at task enqueue/dequeue
- Lots of detail at last year’s OSPM Summit and lkml
  - Patches merged into upstream tip/sched/core branch

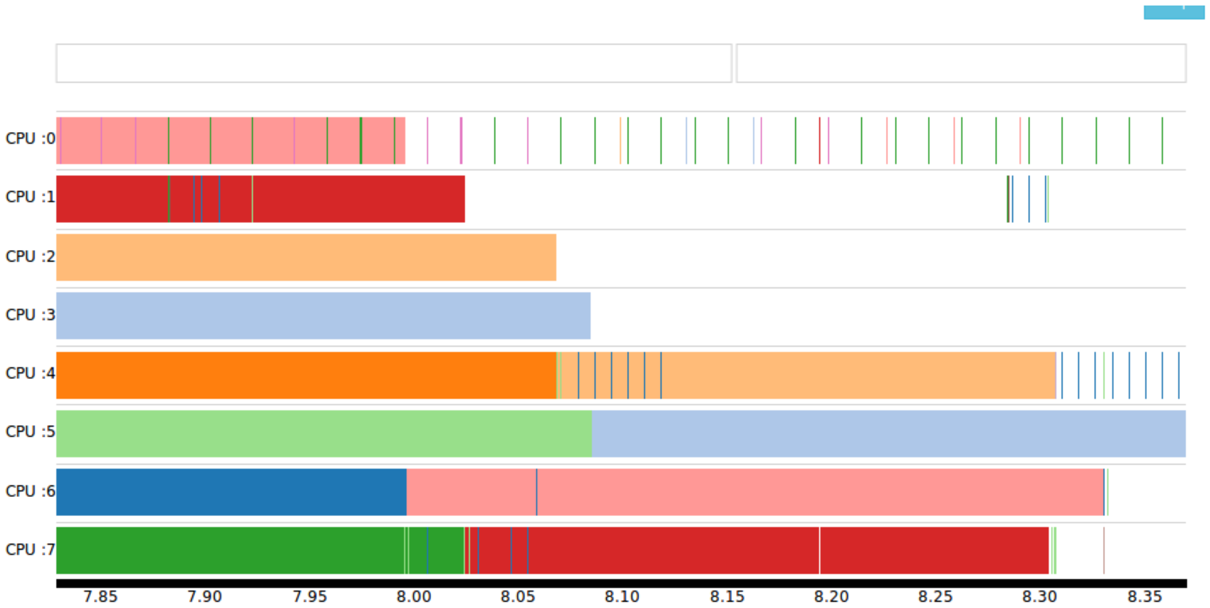


```
    increased: here we update the fair scheduling stats and
@@ -4834,6 +4841,14 @@ enqueue_task_fair(struct rq *rq, struct task_struct *p, int flags)
        if (!se)
                add_nr_running(rq, 1);

+       /*
+        * Update (top level CFS) RQ estimated utilization.
+        * NOTE: the following code assume that we never change the
+        *       utilization estimation policy at run-time.
+        */
+       cfs_rq = &(task_rq(p)->cfs);
+       cfs_rq->avg.util_est.last += task_util_est(p);
+
        hrtick_update(rq);
 }

@@ -4893,6 +4908,24 @@ static void dequeue_task_fair(struct rq *rq, struct task_struct *p, int flags)
        if (!se)
                sub_nr_running(rq, 1);

+       /*
+        * Update (top level CFS) RQ estimated utilization
+        * NOTE: for RQs we alwasy use util_est.last since we do not track an
+        *       EWMA, which is tracked only for Tasks.
+        */
+       cfs_rq = &(task_rq(p)->cfs);
+       cfs_rq->avg.util_est.last = max_t(long,
+                       cfs_rq->avg.util_est.last - task_util_est(p), 0);
+
+       /* Update Task's estimated utilization */
+       if (task_sleep) {
+               /* Keep track of the utilization for the last activation */
+               p->se.avg.util_est.last = task_util(p);
+               /* Update EWMA for Task utilization */
+               ewma_util_add(&p->se.avg.util_ewma, task_util(p));
```



EAS Upstreaming

- Misfit Tasks
  - Promote long-running tasks to most capable Cpus
  - Key to achieving consistent performance in heterogenous systems
  - Tasks which don't sleep need active migration



arm

A Simplified Energy Model for EAS

An Energy Model: why ?

- Power/perf. characteristics differ between different SoCs
- Heuristics don’t perform well on many platforms
- The Energy Model enables the design of a platform-agnostic algorithm in the scheduler
- Designed for mainline

Summary

1. Today’s Energy Model
2. Which simplified Energy Model ?
3. Mainline implementation
4. Conclusion

Summary

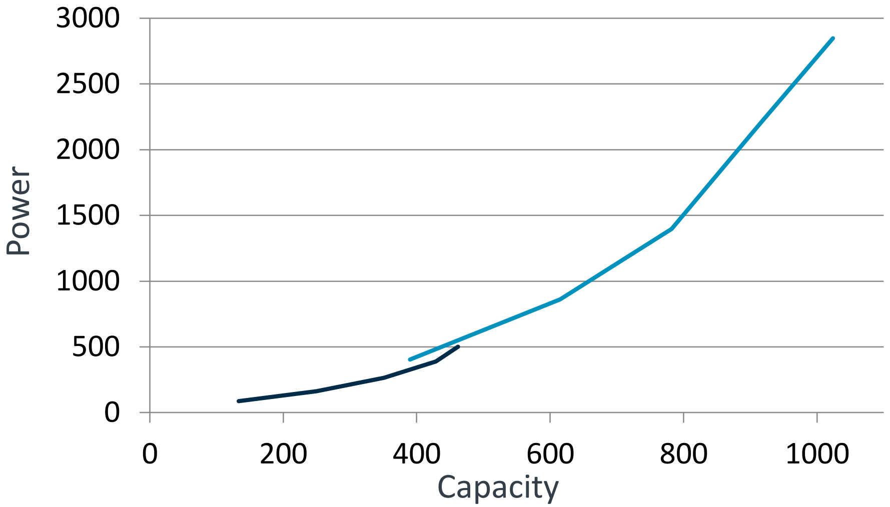
1. **Today’s Energy Model**
2. Which simplified Energy Model ?
3. Mainline implementation
4. Conclusion

The Energy Model in Android / Hikey960

The Energy Model in Android / Hikey960

CPU LEVEL

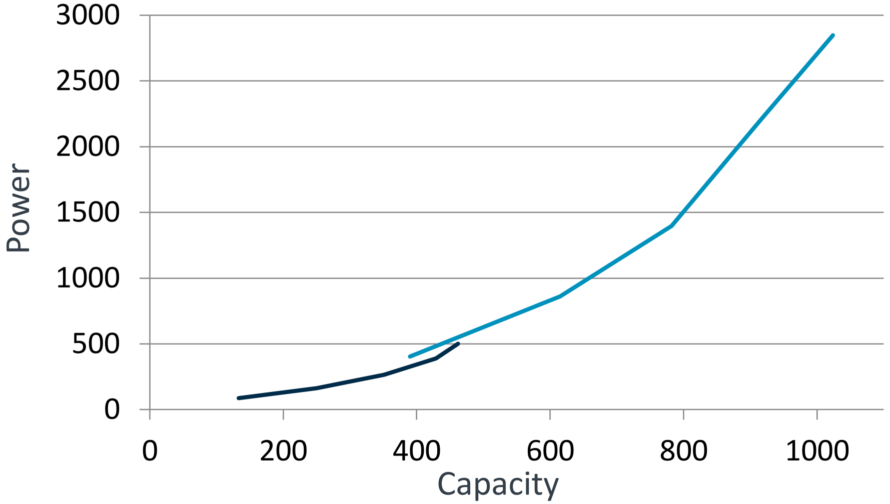
LITTLE			big		
MHz	*Cap.*	*Cost*	*MHz*	*Cap.*	*Cost*
533	133	87	**903**	390	404
999	250	167	**1421**	615	861
1402	351	265	**1805**	782	1398
1709	429	388	**2112**	915	2200
1844	462	502	**2362**	1024	2848



The Energy Model in Android / Hikey960

CPU LEVEL

LITTLE			big		
MHz	*Cap.*	*Cost*	*MHz*	*Cap.*	*Cost*
533	133	87	**903**	390	404
999	250	167	**1421**	615	861
1402	351	265	**1805**	782	1398
1709	429	388	**2112**	915	2200
1844	462	502	**2362**	1024	2848

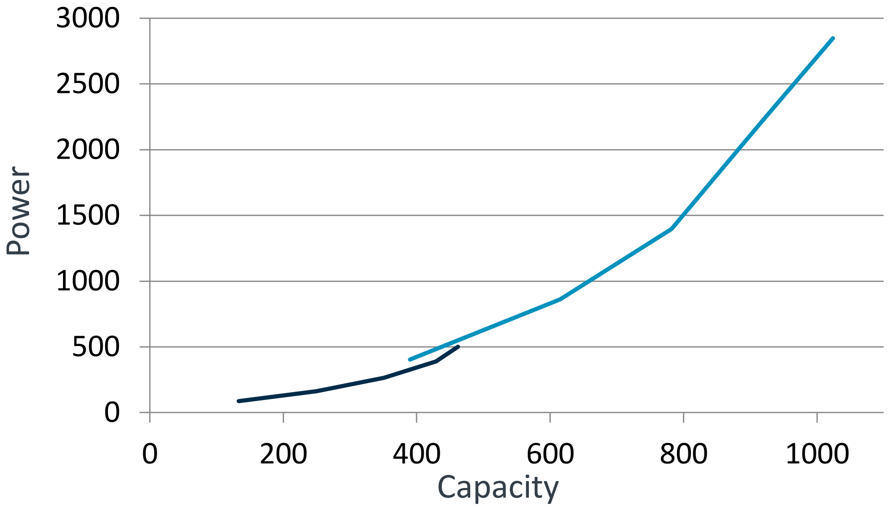


LITTLE	big
5	18
5	18
0	0
0	0

The Energy Model in Android / Hikey960

CPU LEVEL

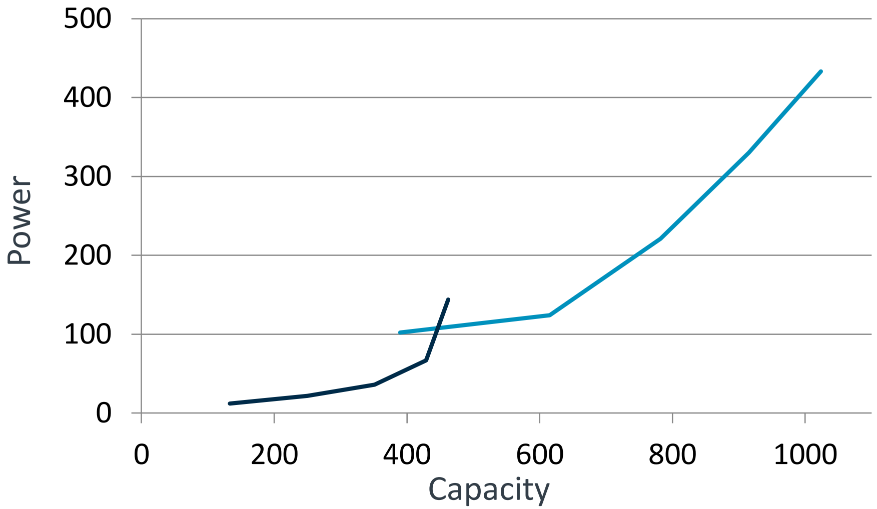
LITTLE			big		
MHz	*Cap.*	*Cost*	*MHz*	*Cap.*	*Cost*
533	133	87	**903**	390	404
999	250	167	**1421**	615	861
1402	351	265	**1805**	782	1398
1709	429	388	**2112**	915	2200
1844	462	502	**2362**	1024	2848



LITTLE	big
5	18
5	18
0	0
0	0

CLUSTER LEVEL

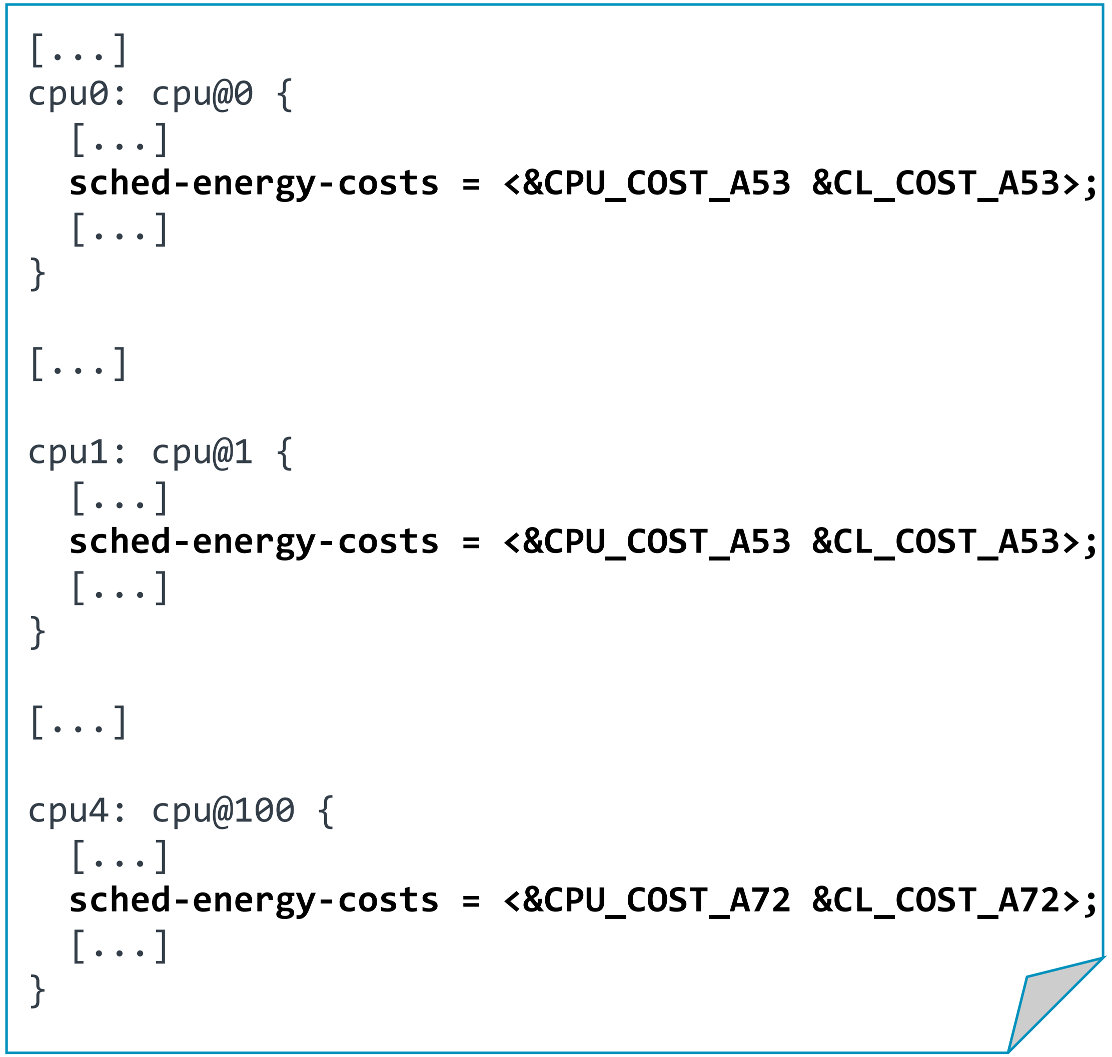
LITTLE			big		
MHz	*Cap.*	*Cost*	*MHz*	*Cap.*	*Cost*
533	133	12	**903**	390	102
999	250	22	**1421**	615	124
1402	351	36	**1805**	782	221
1709	429	67	**2112**	915	330
1844	462	144	2362	1024	433



LITTLE	big
12	102
12	102
12	102
0	0

Device-tree bindings

Device-tree bindings



```
[...]
cpu0: cpu@0 {
  [...]
  sched-energy-costs = <&CPU_COST_A53 &CL_COST_A53>;
  [...]
}

[...]

cpu1: cpu@1 {
  [...]
  sched-energy-costs = <&CPU_COST_A53 &CL_COST_A53>;
  [...]
}

[...]

cpu4: cpu@100 {
  [...]
  sched-energy-costs = <&CPU_COST_A72 &CL_COST_A72>;
  [...]
}
```

arch/arm64/boot/dts/hisilicon/hi3660.dtsi

Device-tree bindings

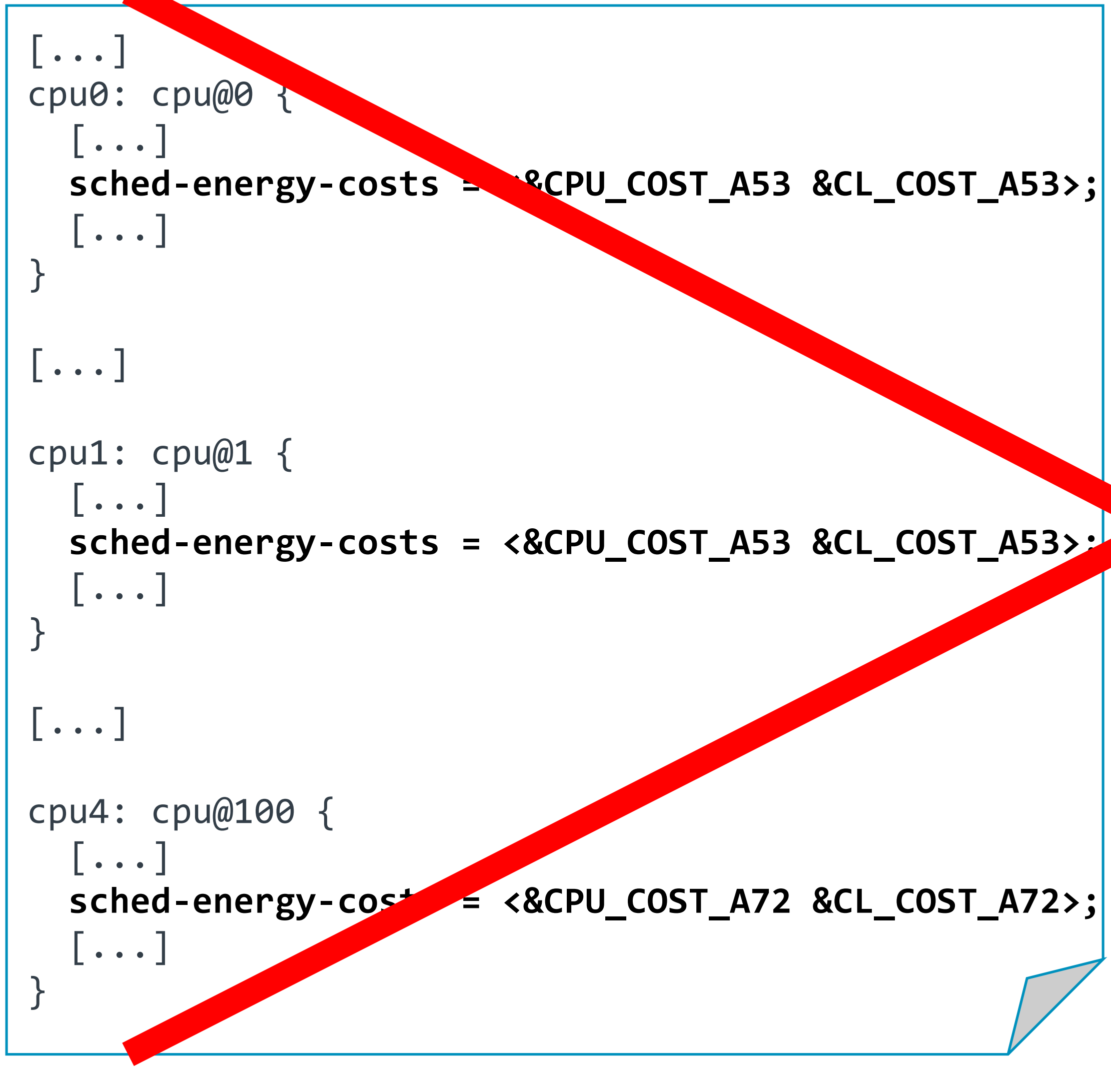
```
[...]
cpu0: cpu@0 {
  [...]
  sched-energy-costs = <&CPU_COST_A53 &CL_COST_A53>;
  [...]
}

[...]

cpu1: cpu@1 {
  [...]
  sched-energy-costs = <&CPU_COST_A53 &CL_COST_A53>;
  [...]
}

[...]

cpu4: cpu@100 {
  [...]
  sched-energy-costs = <&CPU_COST_A72 &CL_COST_A72>;
  [...]
}
```

arch/arm64/boot/dts/hisilicon/hi3660.dtsi

```
CPU_COST_A72: core-cost0 {
    busy-cost-data = <
        390    404
        615    861
        782    1398
        915    2200
        1024   2848 >;
    idle-cost-data = < 18 18 0 0 >;
};
CPU_COST_A53: core-cost1 {
    busy-cost-data = <
        133    87
        250    164
        351    265
        429    388
        462    502 >;
    idle-cost-data = < 5 5 0 0 >;
};
CLUSTER_COST_A72: cluster-cost0 {
    busy-cost-data = <

  [...]
```

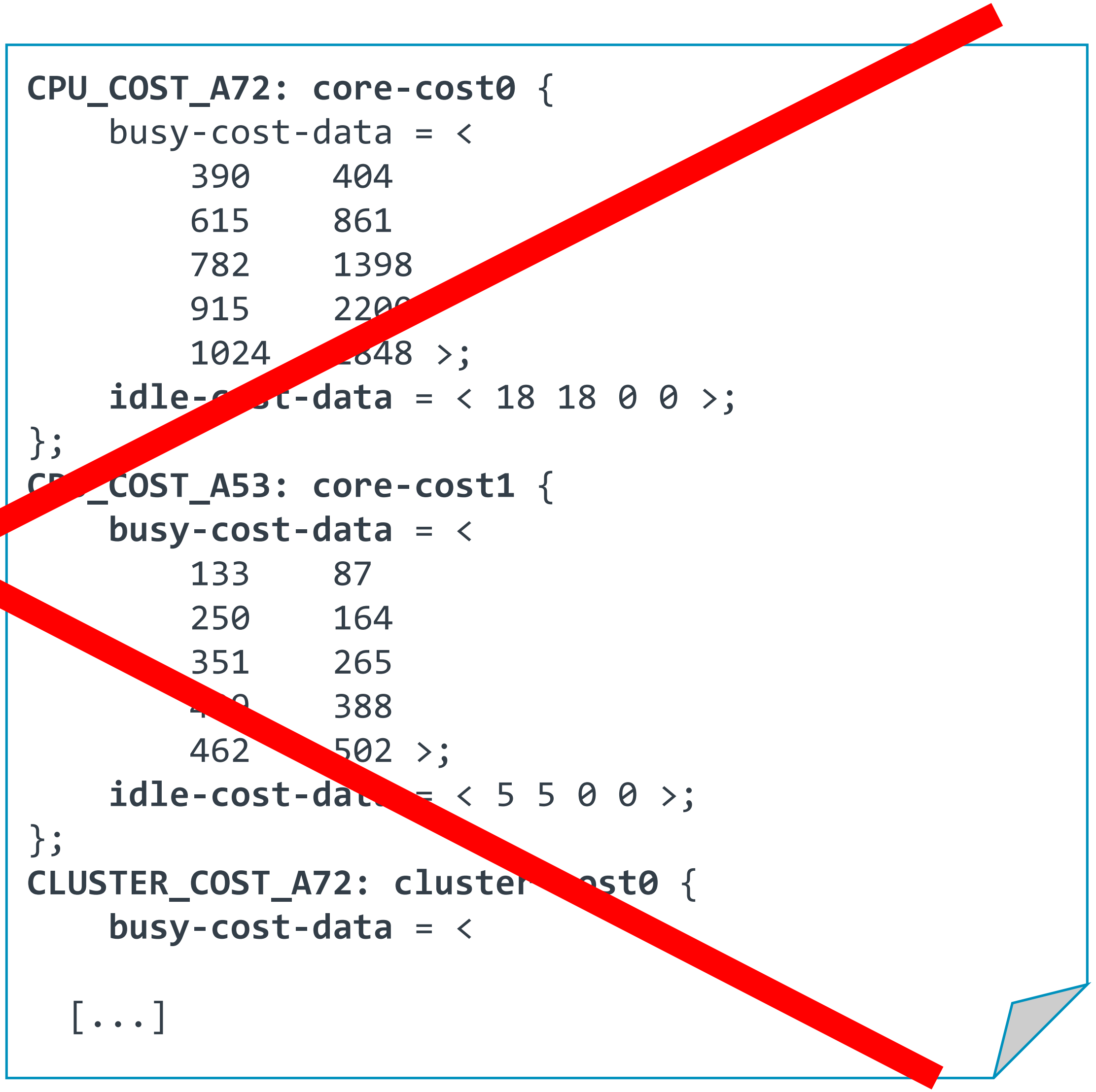
arch/arm64/boot/dts/hisilicon/hi3660-sched-energy.dtsi

Device-tree bindings



```
[...]
cpu0: cpu@0 {
  [...]
  sched-energy-costs = <&CPU_COST_A53 &CL_COST_A53>;
  [...]
}

[...]

cpu1: cpu@1 {
  [...]
  sched-energy-costs = <&CPU_COST_A53 &CL_COST_A53>;
  [...]
}

[...]

cpu4: cpu@100 {
  [...]
  sched-energy-costs = <&CPU_COST_A72 &CL_COST_A72>;
  [...]
}
```

arch/arm64/boot/dts/hisilicon/hi3660.dtsi

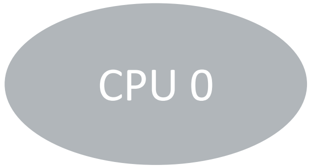


```
CPU_COST_A72: core-cost0 {
    busy-cost-data = <
        390    404
        615    861
        782    1398
        915    2200
        1024   2848 >;
    idle-cost-data = < 18 18 0 0 >;
};
CPU_COST_A53: core-cost1 {
    busy-cost-data = <
        133    87
        250    164
        351    265
        429    388
        462    502 >;
    idle-cost-data = < 5 5 0 0 >;
};
CLUSTER_COST_A72: cluster-cost0 {
    busy-cost-data = <

  [...]
```

arch/arm64/boot/dts/hisilicon/hi3660-sched-energy.dtsi

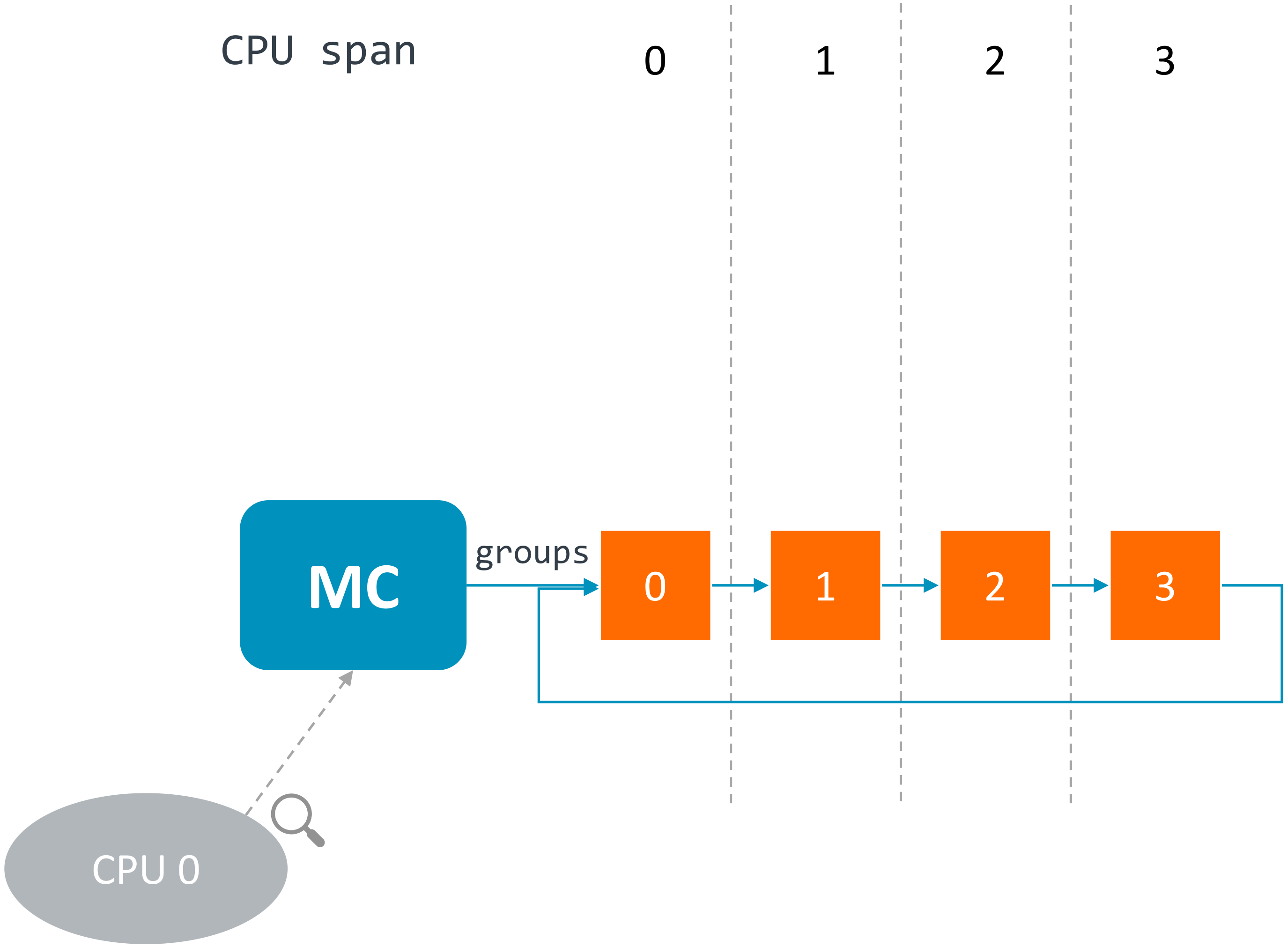
Sched domains

Sched domains

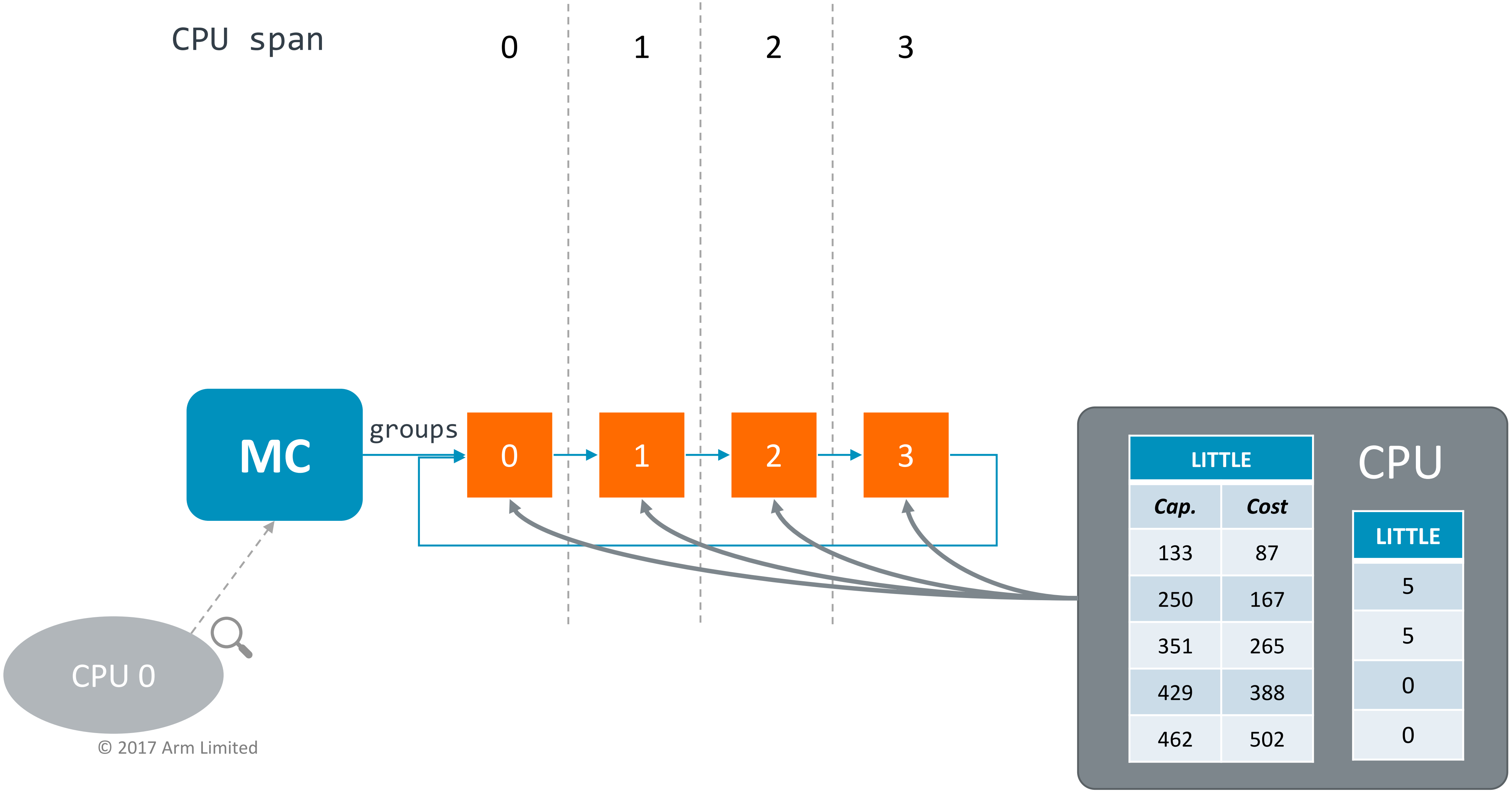


CPU 0

Sched domains



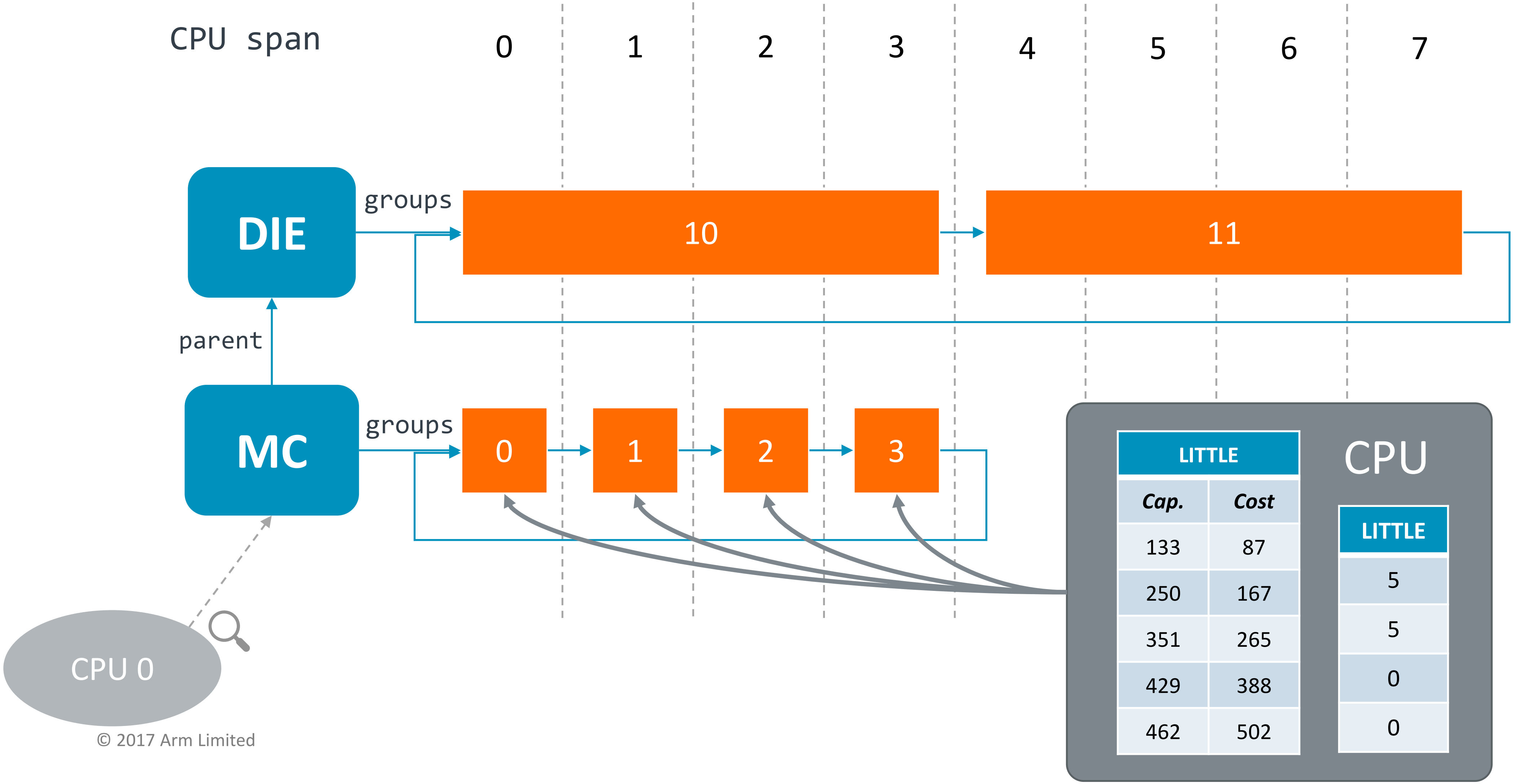
Sched domains



LITTLE	
Cap.	***Cost***
133	87
250	167
351	265
429	388
462	502

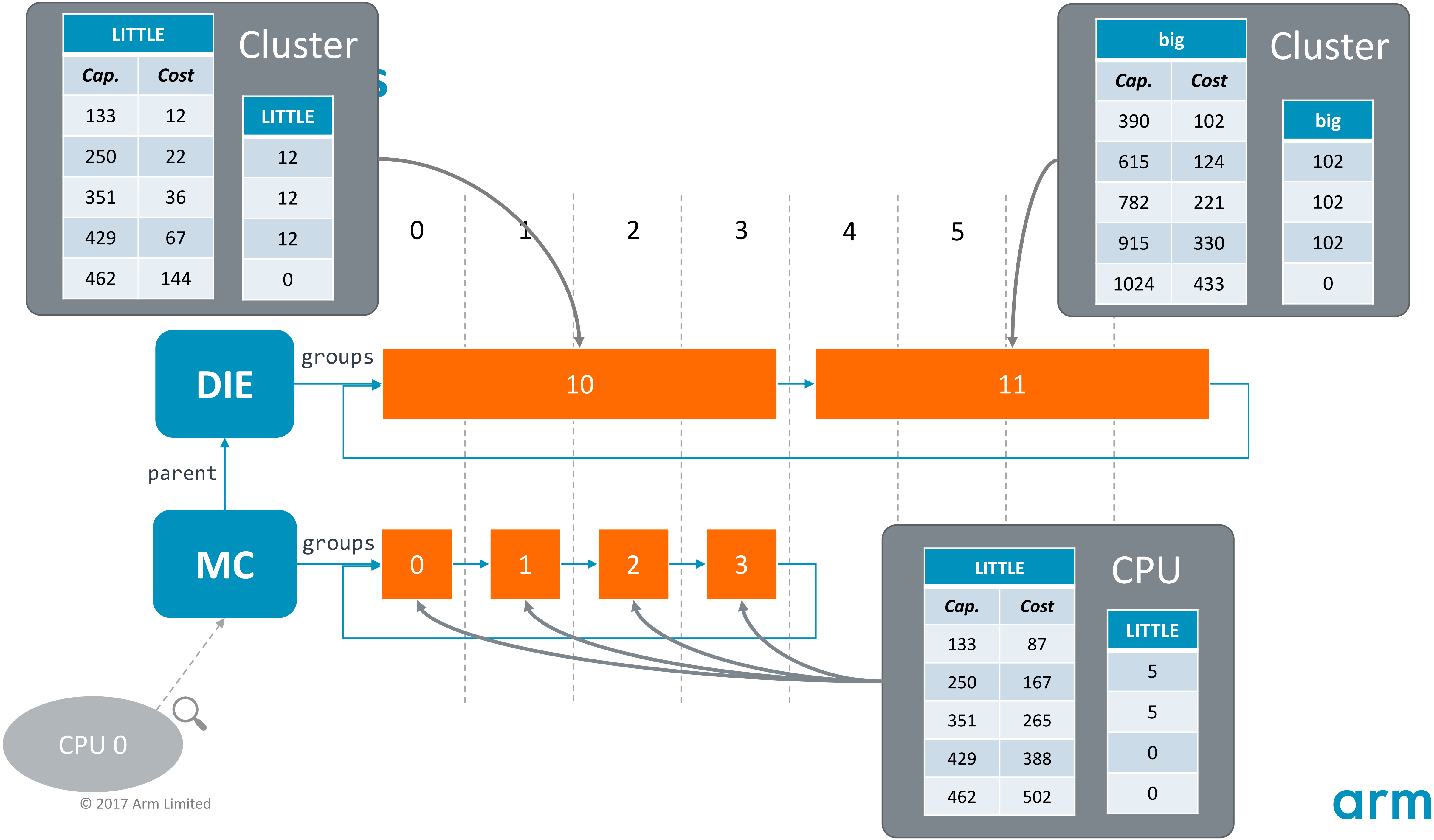
LITTLE
5
5
0
0

Sched domains



LITTLE	
Cap.	***Cost***
133	87
250	167
351	265
429	388
462	502

LITTLE
5
5
0
0



Cluster

LITTLE	
Cap.	*Cost*
133	12
250	22
351	36
429	67
462	144

LITTLE
12
12
12
0

Cluster

big	
Cap.	*Cost*
390	102
615	124
782	221
915	330
1024	433

big
102
102
102
0

CPU

LITTLE	
Cap.	*Cost*
133	87
250	167
351	265
429	388
462	502

LITTLE
5
5
0
0

Need for simplification

- Comprehensive Energy Model, but …

Need for simplification

- Comprehensive Energy Model, but …
- Complex to measure for new platforms

Need for simplification

- Comprehensive Energy Model, but …
- Complex to measure for new platforms
- Computationally expensive scheduling decisions

Need for simplification

- Comprehensive Energy Model, but …
- Complex to measure for new platforms
- Computationally expensive scheduling decisions
- Existing code relies only on out-of-tree bindings

Need for simplification

- Comprehensive Energy Model, but …
- Complex to measure for new platforms
- Computationally expensive scheduling decisions
- Existing code relies only on out-of-tree bindings
- Inaccurate assumptions for future platforms

Summary

1. Today’s Energy Model
2. **Which simplified Energy Model ?**
3. Mainline implementation
4. Conclusion

Which simplified EM ?

Which simplified EM ?

Name	CPU Level		Cluster Level	
	Active costs	***Idle costs***	***Active costs***	***Idle costs***

Which simplified EM ?

Name	CPU Level		Cluster Level	
	Active costs	***Idle costs***	***Active costs***	***Idle costs***
FULL	YES	YES	YES	YES

Which simplified EM ?

Name	CPU Level		Cluster Level	
	Active costs	*Idle costs*	*Active costs*	*Idle costs*
FULL	YES	YES	YES	YES
NOIDLE	YES	NO	YES	NO

Which simplified EM ?

Name	CPU Level		Cluster Level	
	Active costs	*Idle costs*	*Active costs*	*Idle costs*
FULL	YES	YES	YES	YES
NOIDLE	YES	NO	YES	NO
NOCLUSTER	YES	YES	NO	NO

Which simplified EM ?

Name	CPU Level		Cluster Level	
	Active costs	*Idle costs*	*Active costs*	*Idle costs*
FULL	YES	YES	YES	YES
NOIDLE	YES	NO	YES	NO
NOCLUSTER	YES	YES	NO	NO
NOCLUSTER_NOIDLE	YES	NO	NO	NO

Which simplified EM ?

Name	CPU Level		Cluster Level	
	Active costs	*Idle costs*	*Active costs*	*Idle costs*
FULL	YES	YES	YES	YES
NOIDLE	YES	NO	YES	NO
NOCLUSTER	YES	YES	NO	NO
NOCLUSTER_NOIDLE	YES	NO	NO	NO
NO_EAS	NO	NO	NO	NO

Which simplified EM ?

Name	CPU Level		Cluster Level	
	Active costs	*Idle costs*	*Active costs*	*Idle costs*
FULL	YES	YES	YES	YES
NOIDLE	YES	NO	YES	NO
NOCLUSTER	YES	YES	NO	NO
NOCLUSTER_NOIDLE	YES	NO	NO	NO
NO_EAS	NO	NO	NO	NO

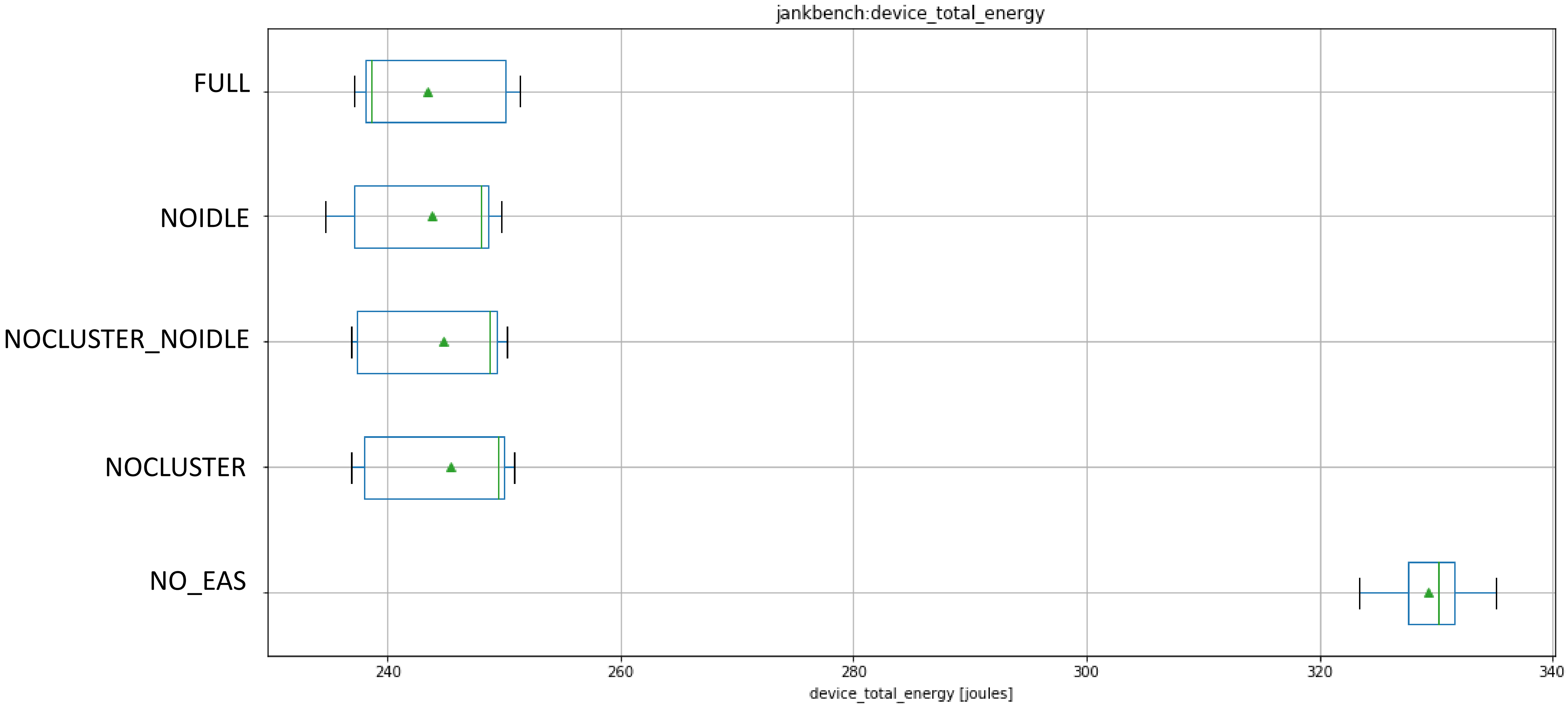
- Tested on Android-4.4: Hikey960, Pixel2, Hikey620

Which simplified EM ?

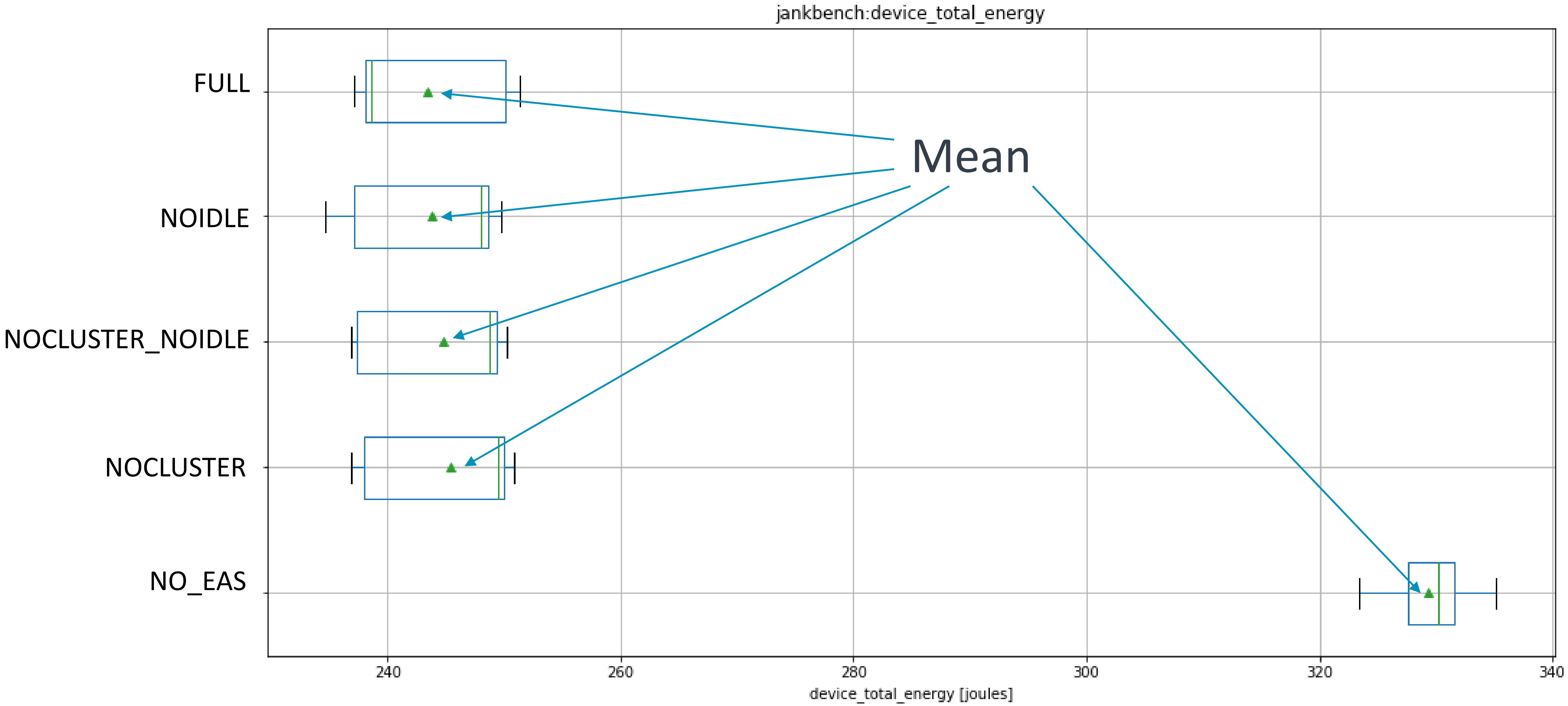
Name	CPU Level		Cluster Level	
	Active costs	*Idle costs*	*Active costs*	*Idle costs*
FULL	YES	YES	YES	YES
NOIDLE	YES	NO	YES	NO
NOCLUSTER	YES	YES	NO	NO
NOCLUSTER_NOIDLE	YES	NO	NO	NO
NO_EAS	NO	NO	NO	NO

- Tested on Android-4.4: Hikey960, Pixel2, Hikey620
- SchedTune disabled, no cpusets

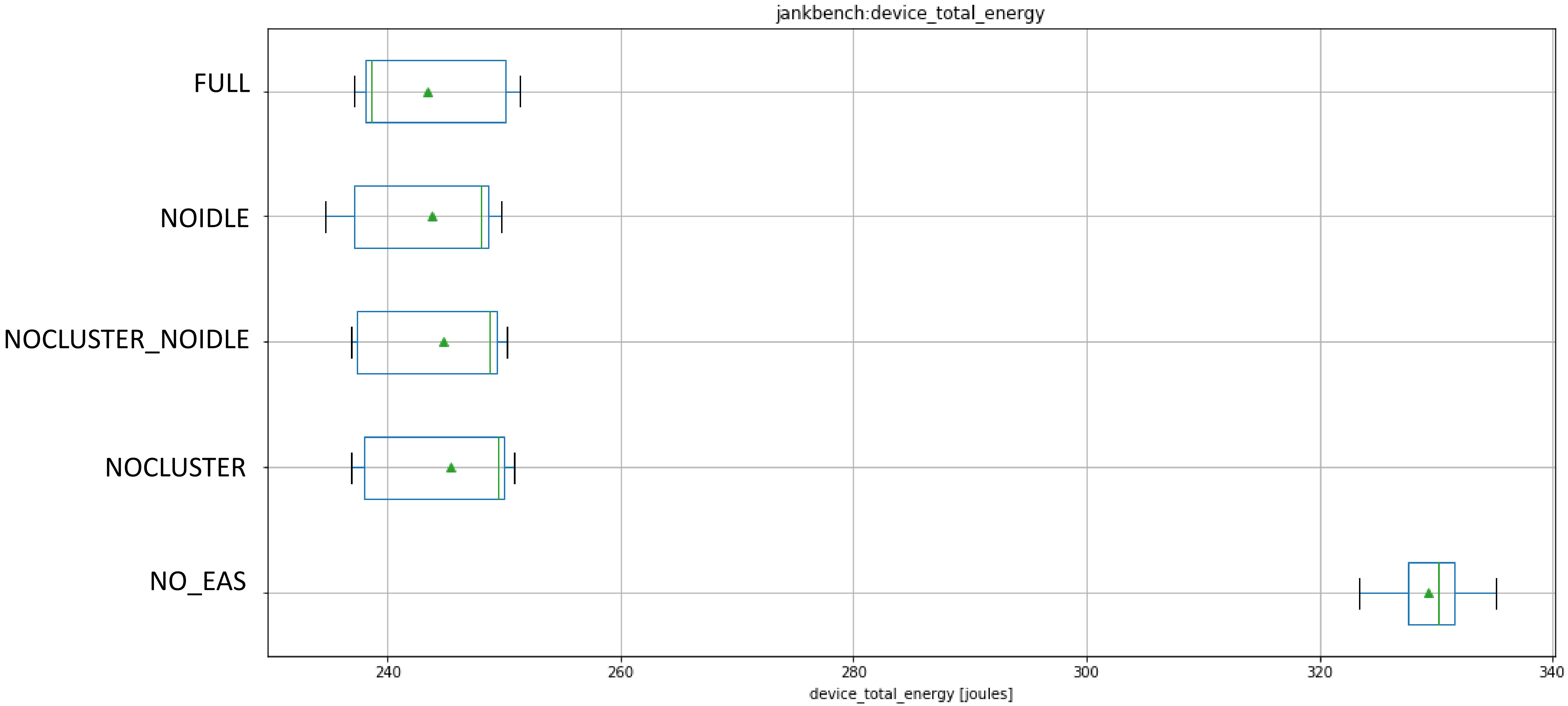
Jankbench / list_view - Hikey960



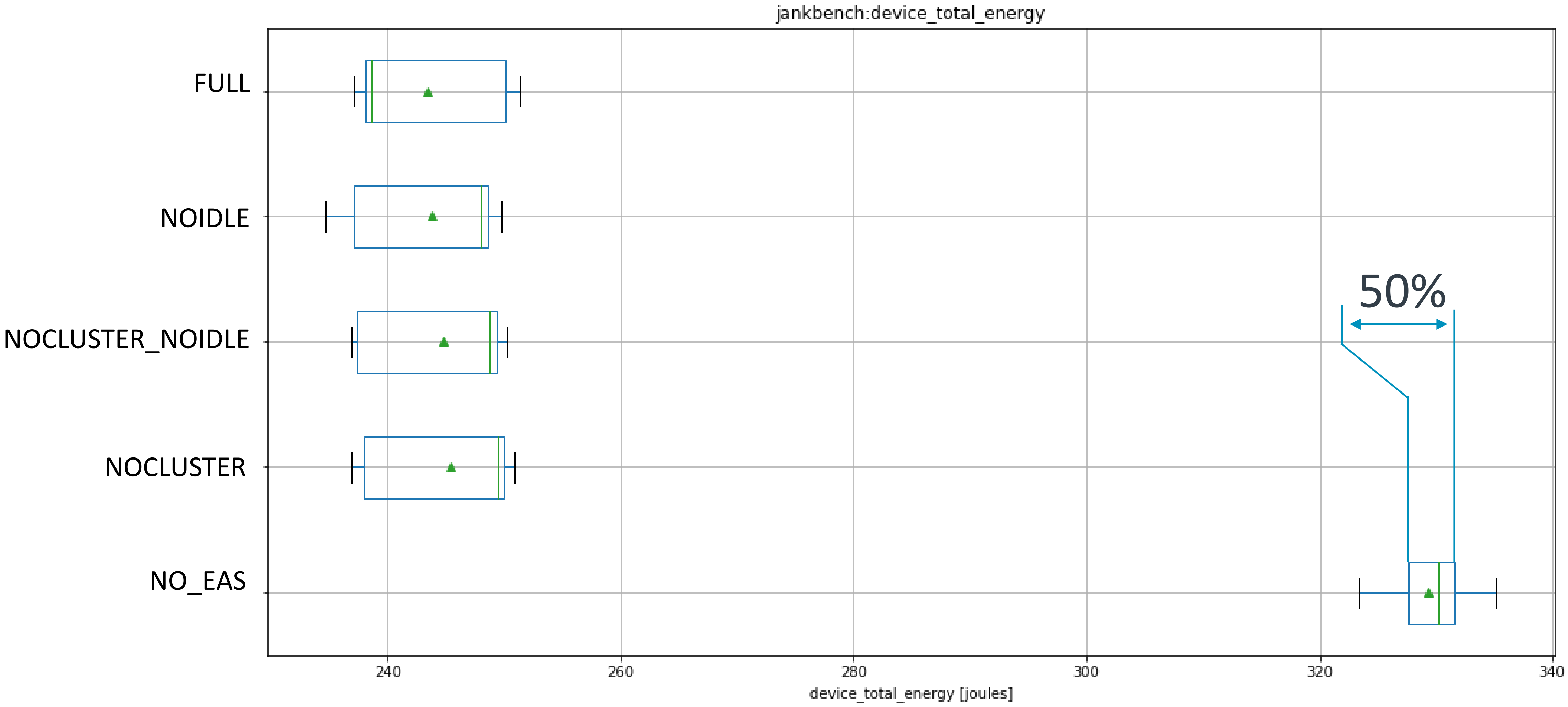
Jankbench / list_view - Hikey960



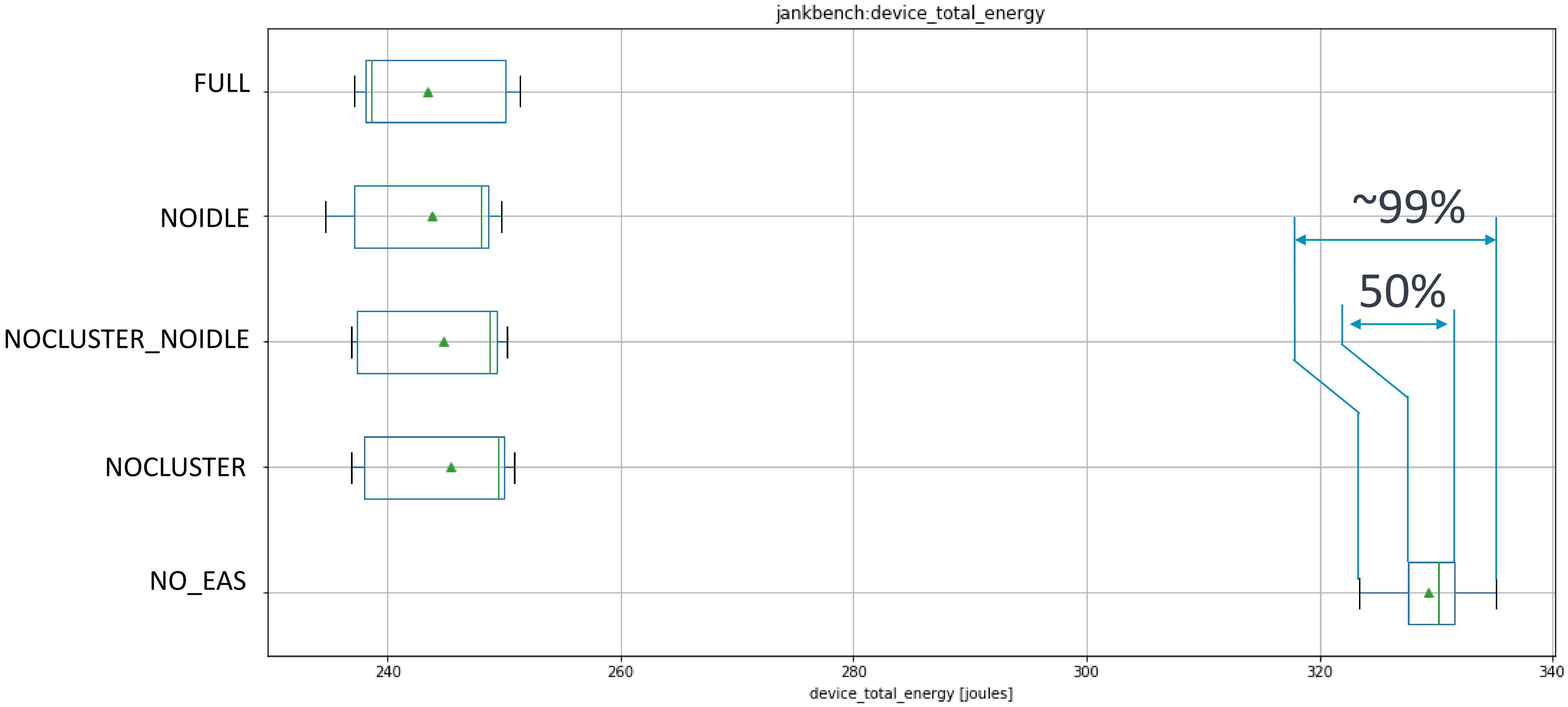
Jankbench / list_view - Hikey960



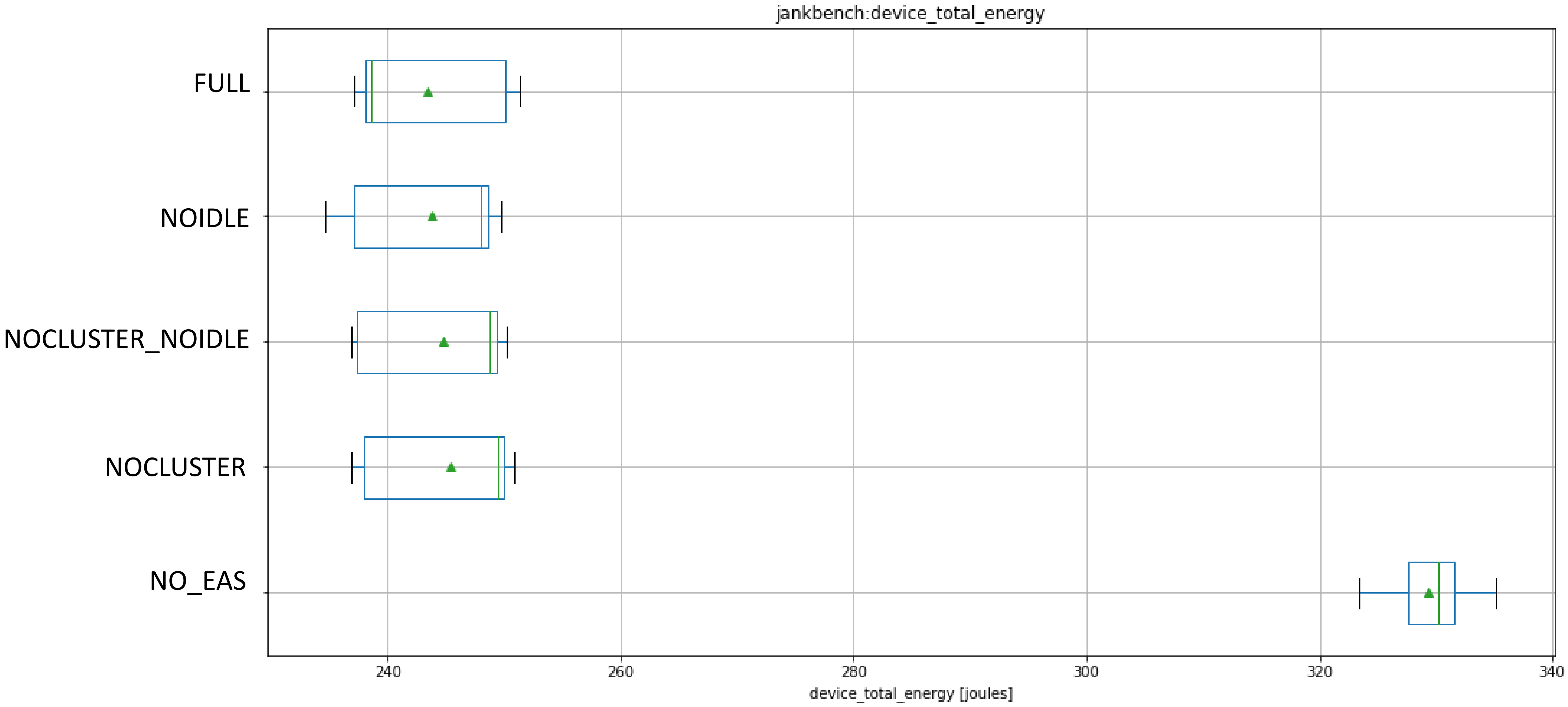
Jankbench / list_view - Hikey960



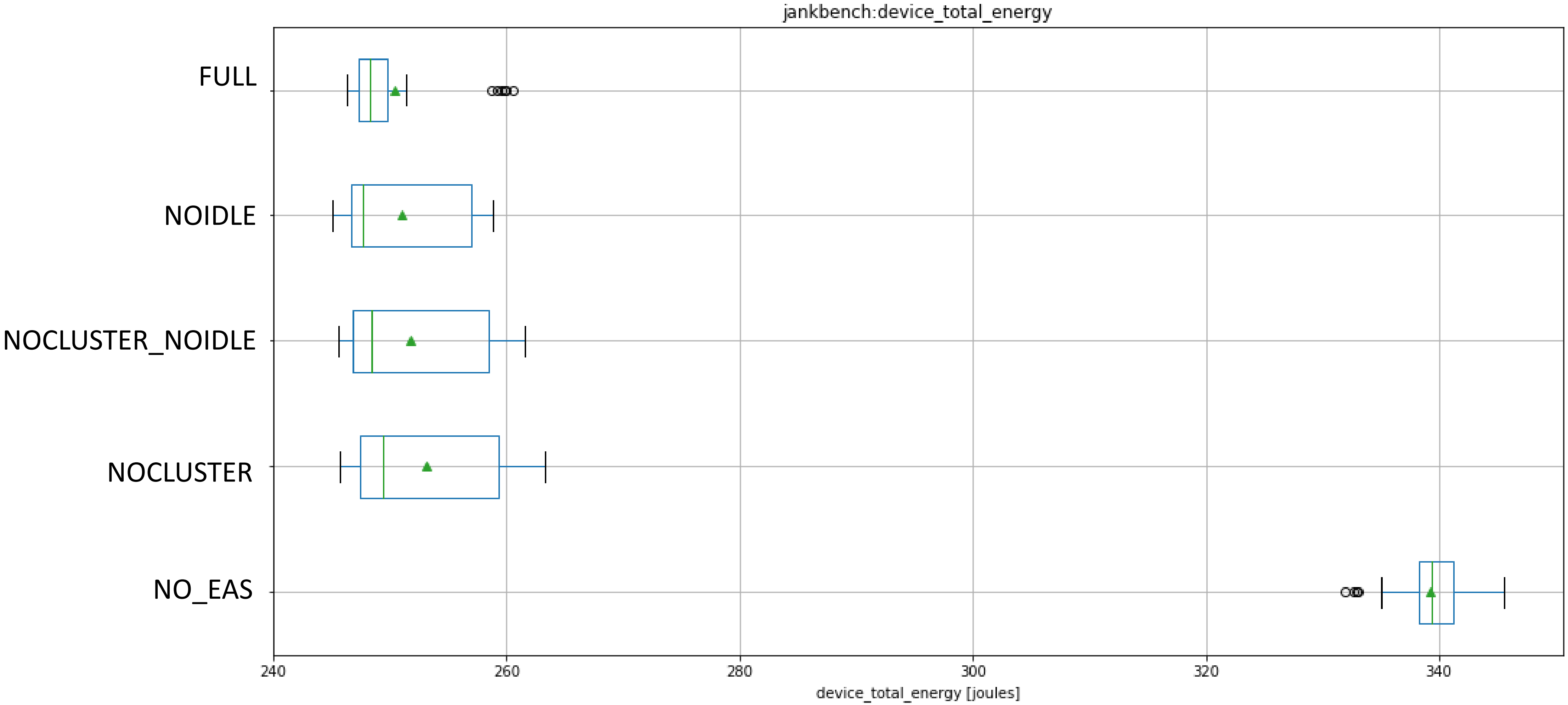
Jankbench / list_view - Hikey960



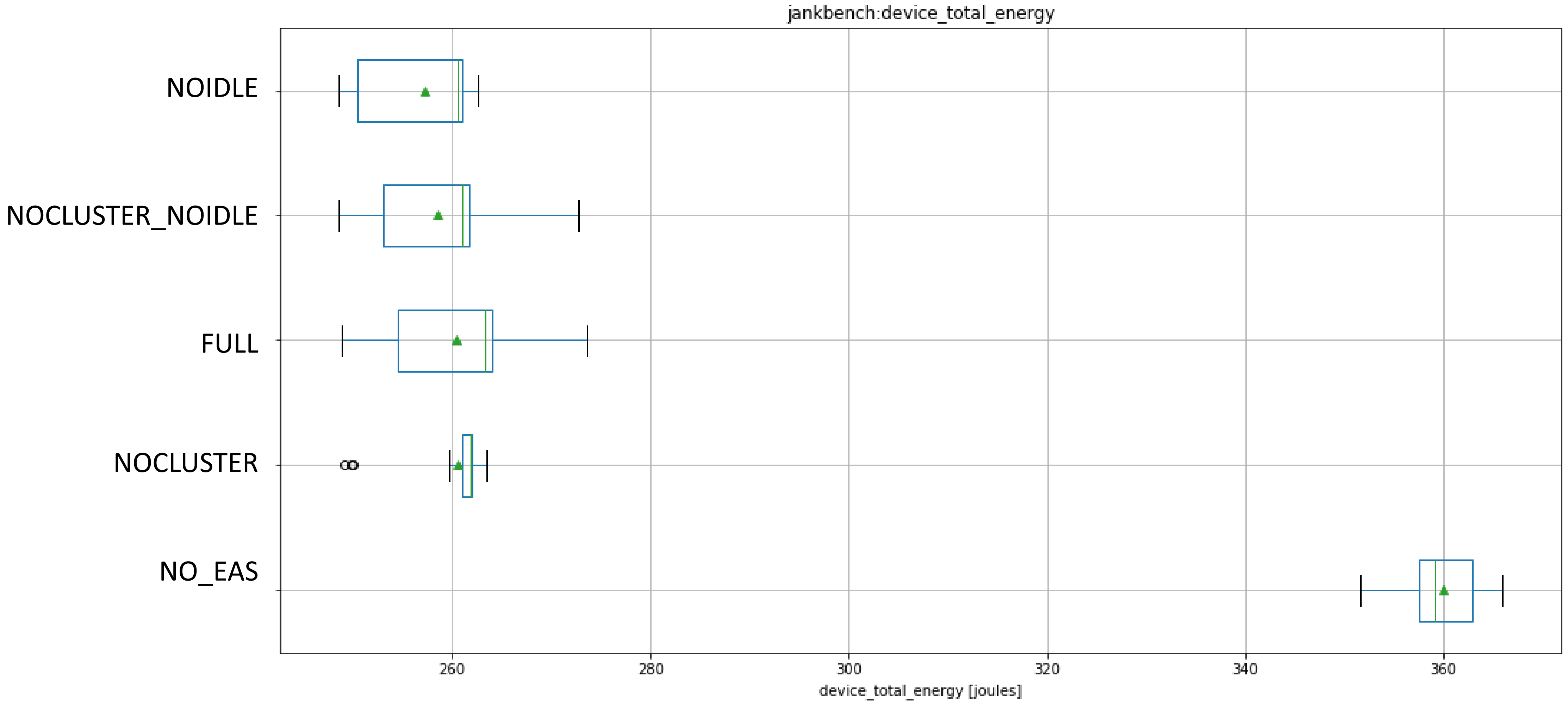
Jankbench / list_view - Hikey960



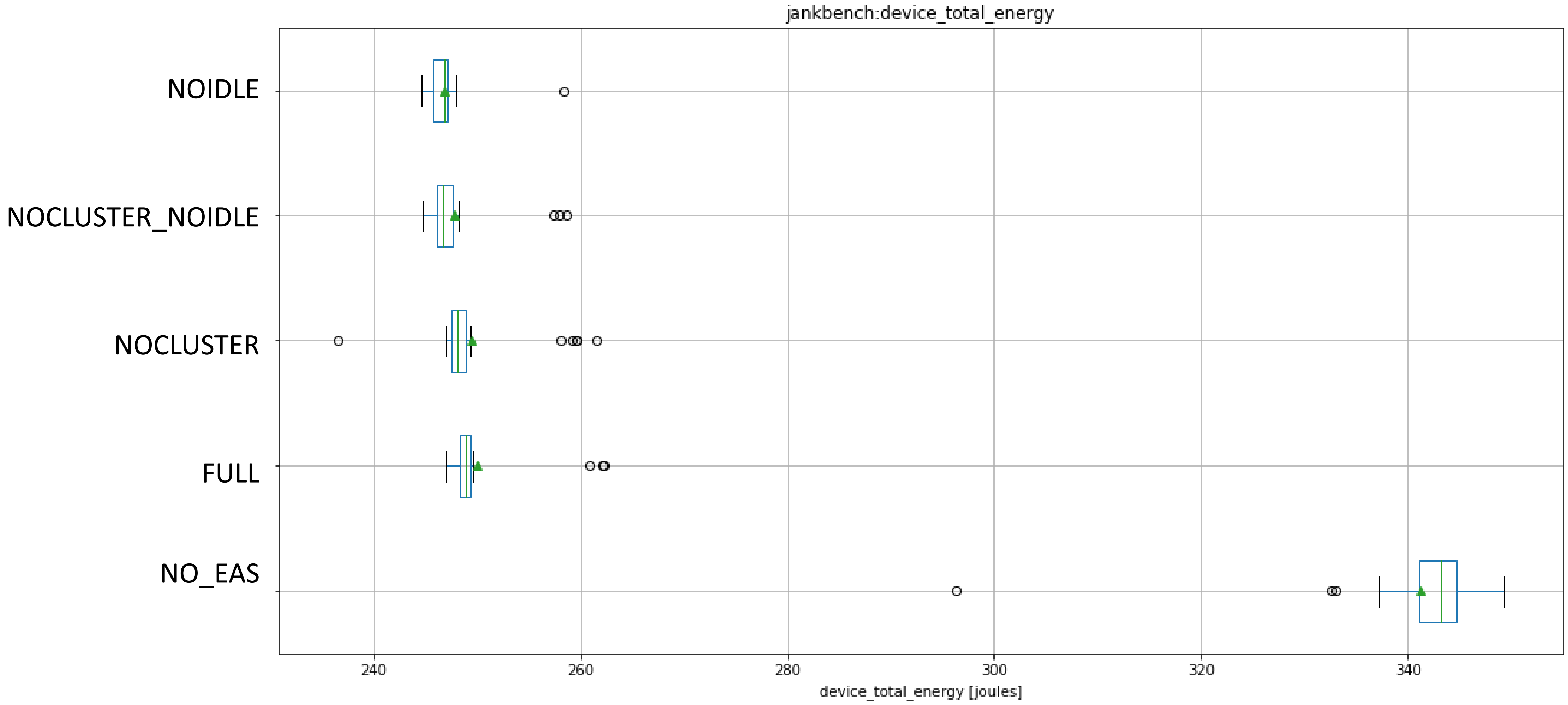
Jankbench / image_list_view - Hikey960



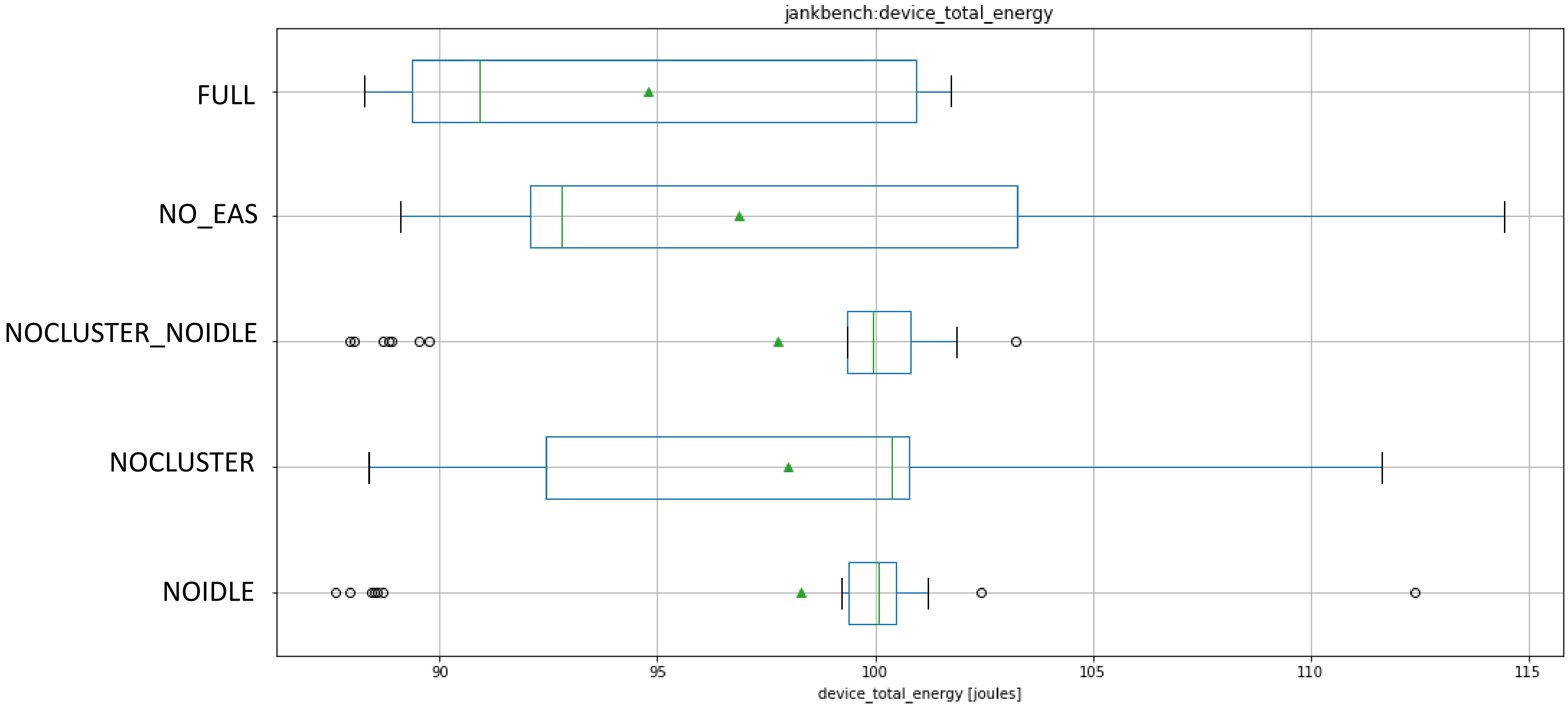
Jankbench / low_hitrate_text - Hikey960



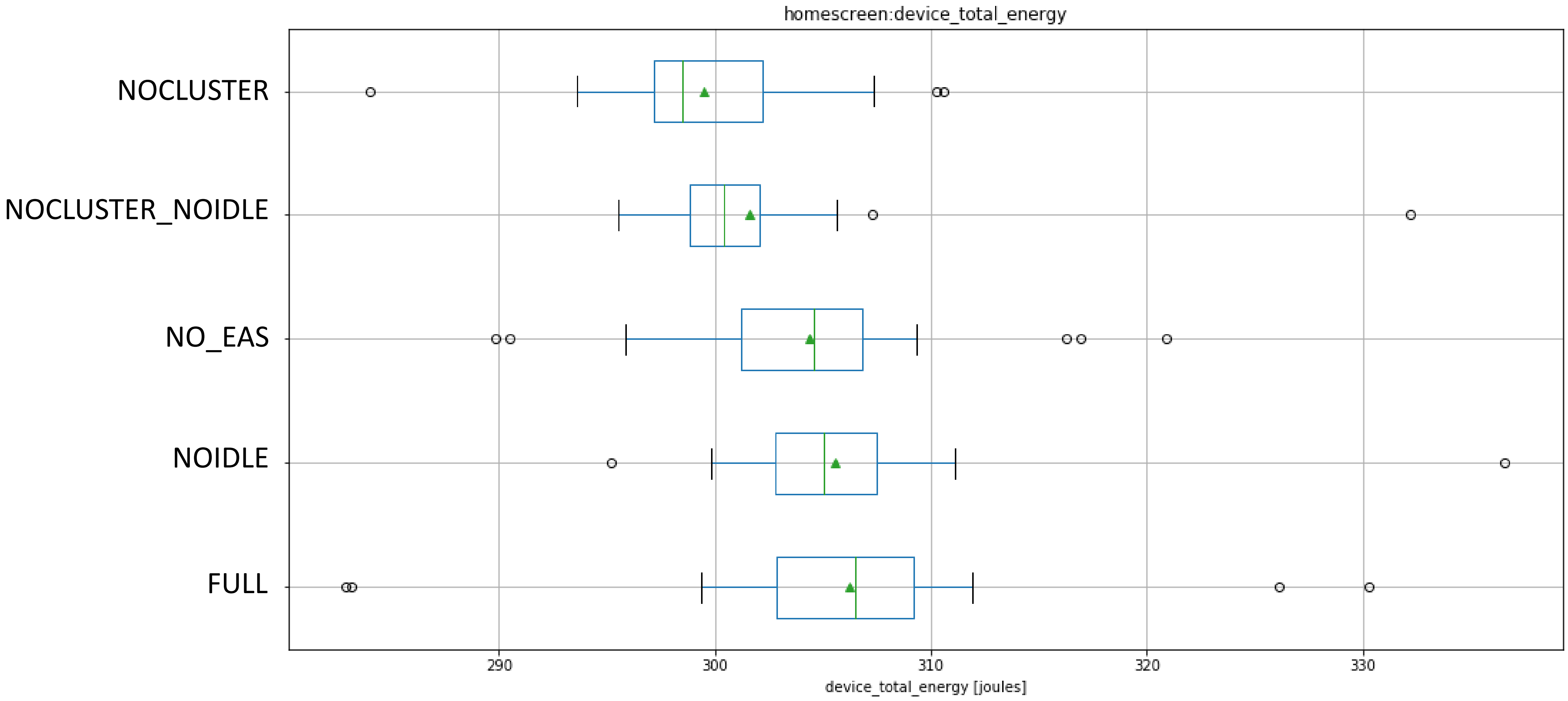
Jankbench / shadow_grid - Hikey960



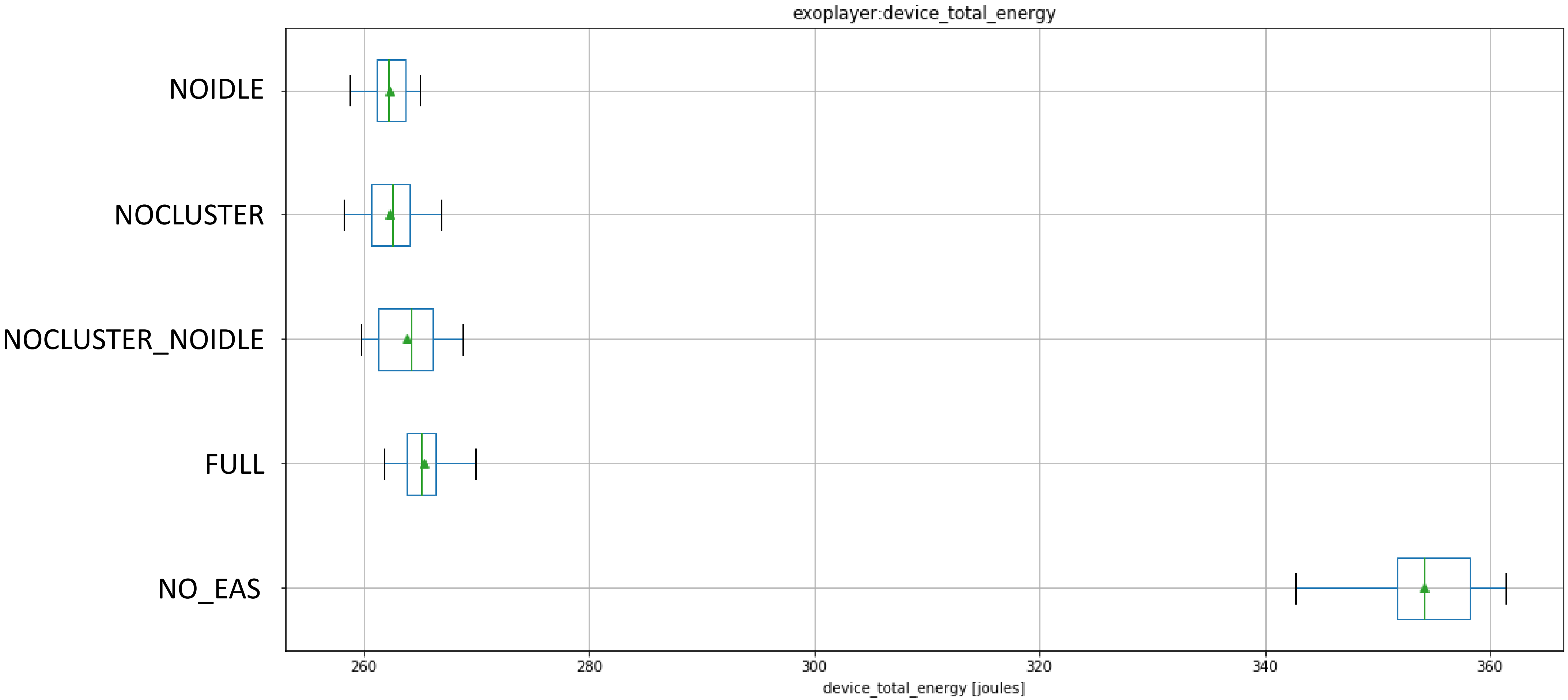
Jankbench / edit_text - Hikey960



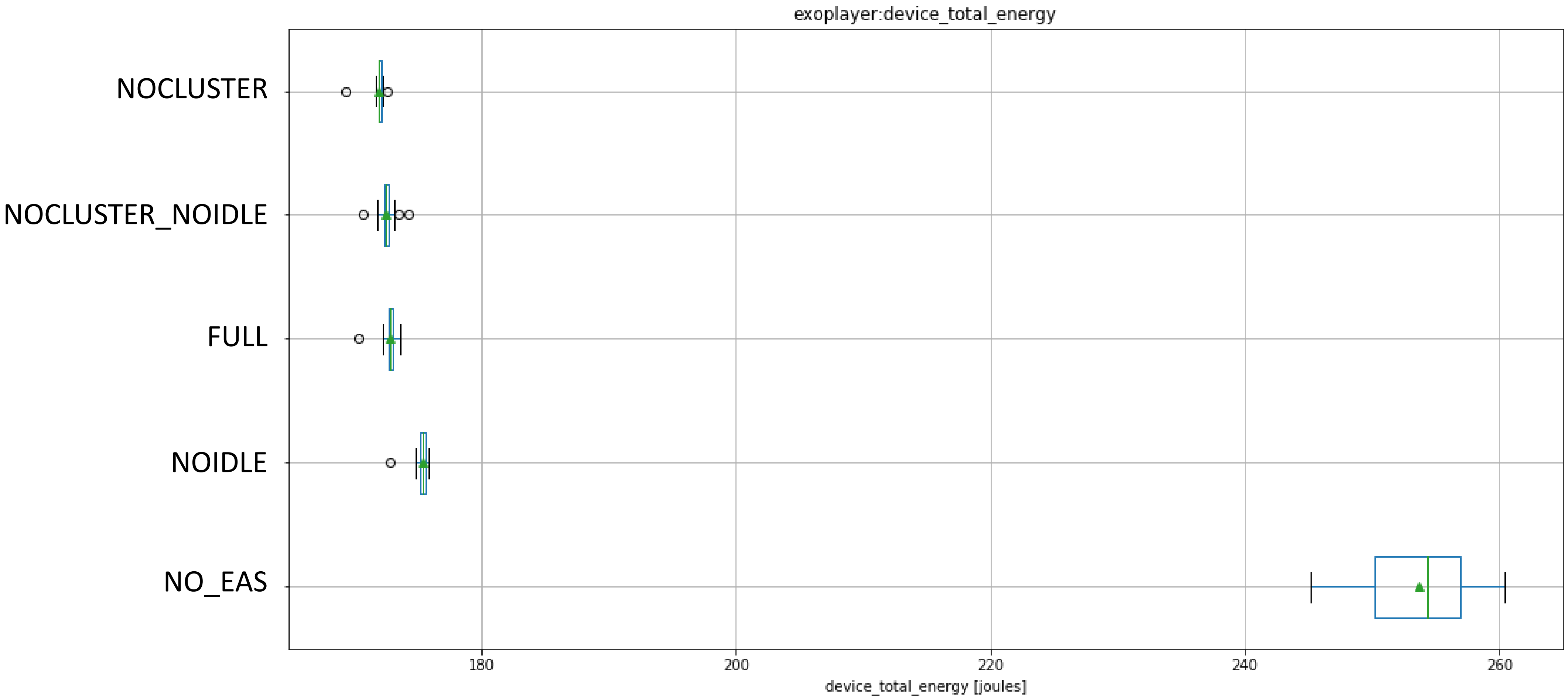
Homescreen / Hikey960



ExoPlayer Video / Hikey960



ExoPlayer Audio / Hikey960



Results of experiments

- Hikey960: all energy models show comparable energy savings
- Pixel2: same conclusions with smaller savings (up to 13%, screen on)
- Hikey620 (*SMP*): No significant savings

Results of experiments

- Hikey960: all energy models show comparable energy savings
- Pixel2: same conclusions with smaller savings (up to 13%, screen on)
- Hikey620 (*SMP*): No significant savings

The simplest EM *(noidle_nocluster)* is
a reasonable option for modern platforms

Summary

1. Today’s Energy Model
2. Which simplified Energy Model ?
3. **Mainline implementation**
4. Conclusion

Dynamic power model

Dynamic power model

$$P = C * \quad V\ 2 * \ f$$

$$2\ 2\ f$$

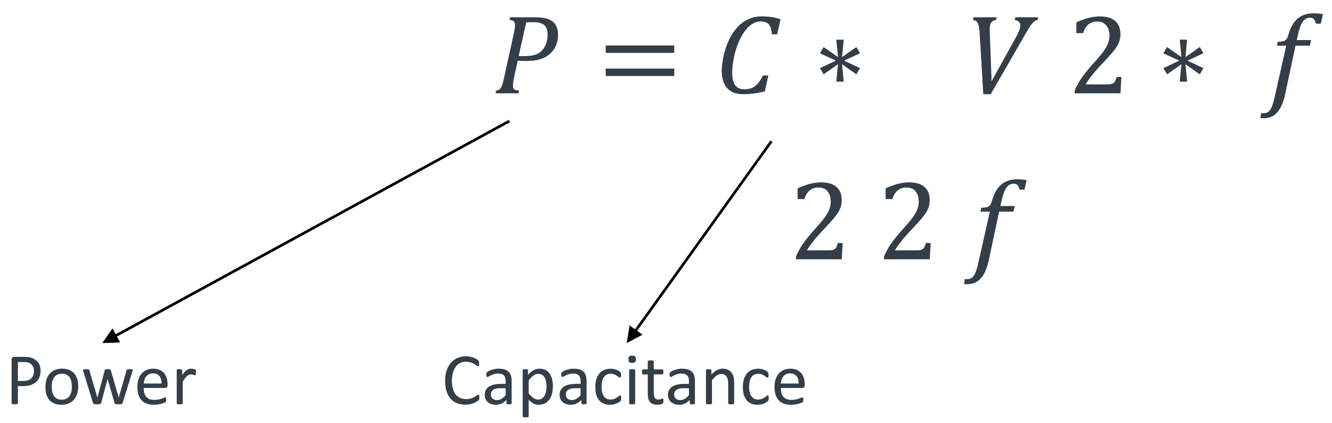
Dynamic power model

$$P = C * \quad V\ 2 * \ f$$

$$2\ 2\ f$$

Power

Dynamic power model



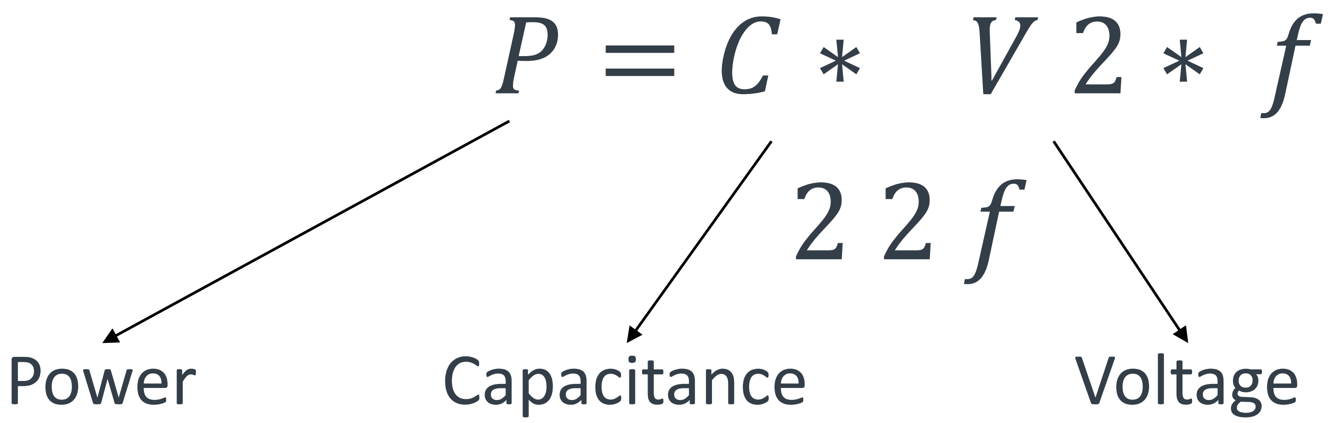
$$P = C * \quad V\,2 * \; f$$

$$2\,2\,f$$

Power

Capacitance

Dynamic power model



$$P = C * \quad V\ 2 * \ f$$

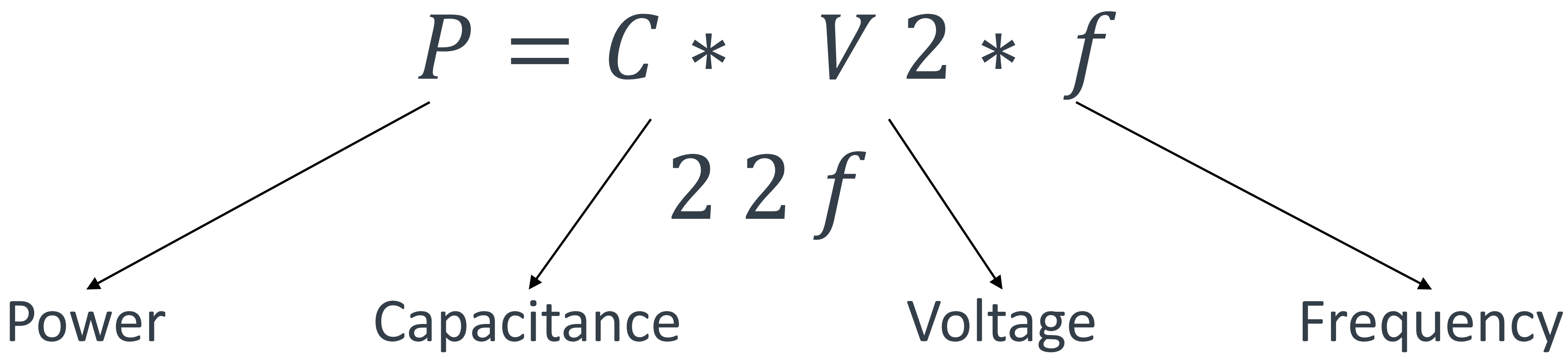
$$2\ 2\ f$$

Power

Capacitance

Voltage

Dynamic power model



$$P = C * \quad V\ 2 * \ f$$

$$2\ 2\ f$$

Power

Capacitance

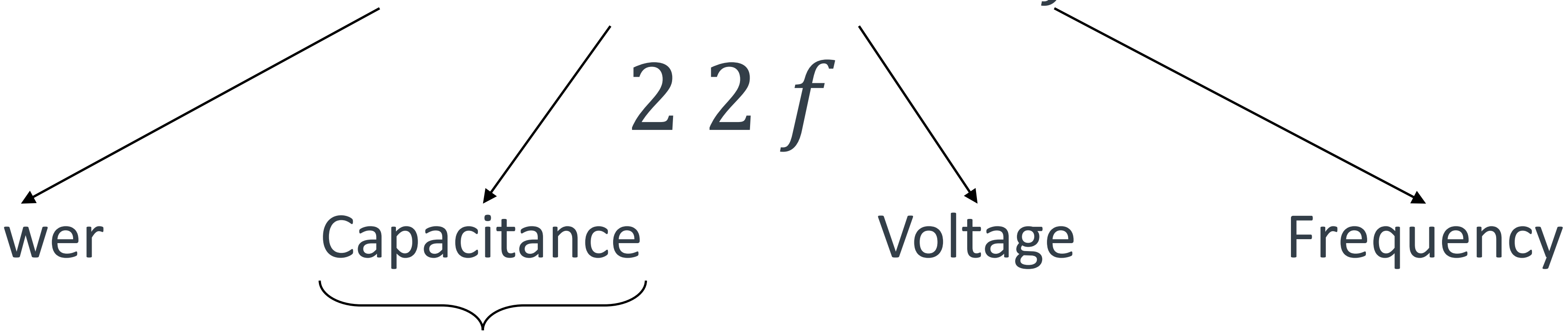
Voltage

Frequency

Dynamic power model

$$P = C * \ V 2 * f$$

$$2 2 f$$



Power

Capacitance

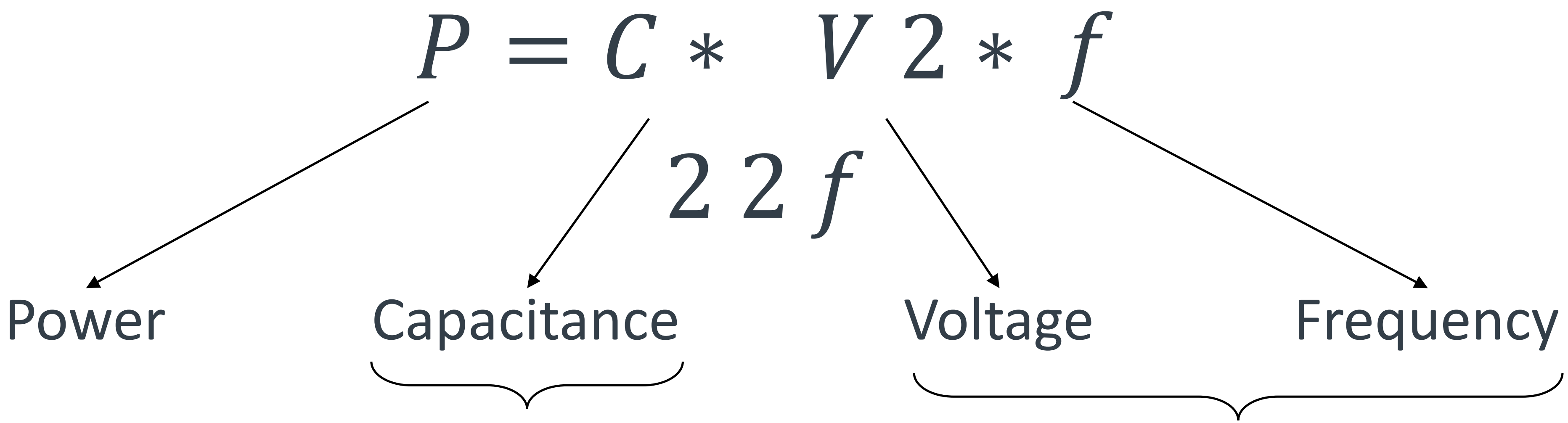
Voltage

Frequency

Mainline DT binding:

`dynamic-power-coefficient`

Dynamic power model



$$P = C * \ V 2 * f$$

$$2 \ 2 \ f$$

Power

Capacitance

Voltage

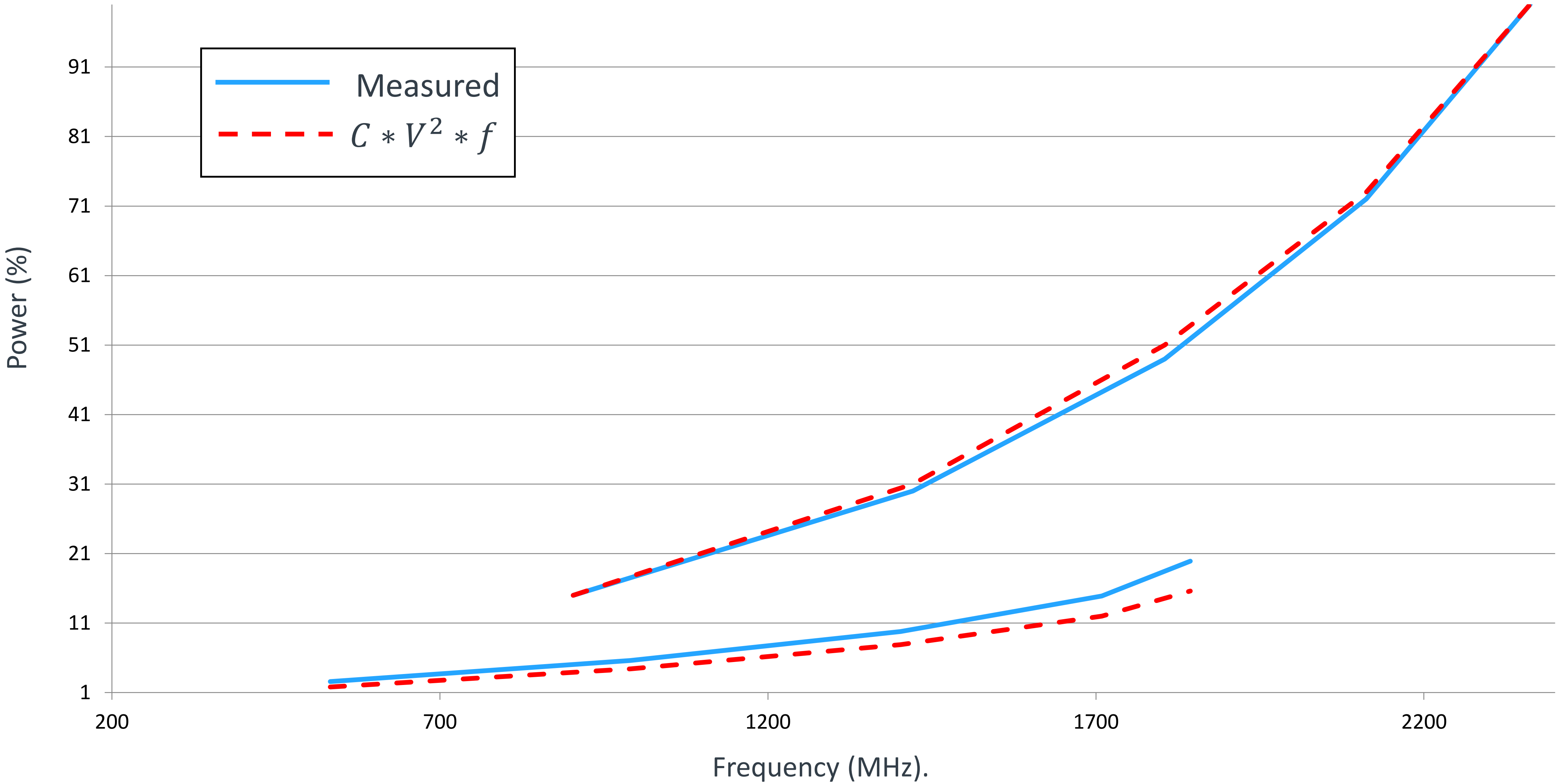
Frequency

Mainline DT binding:

`dynamic-power-coefficient`

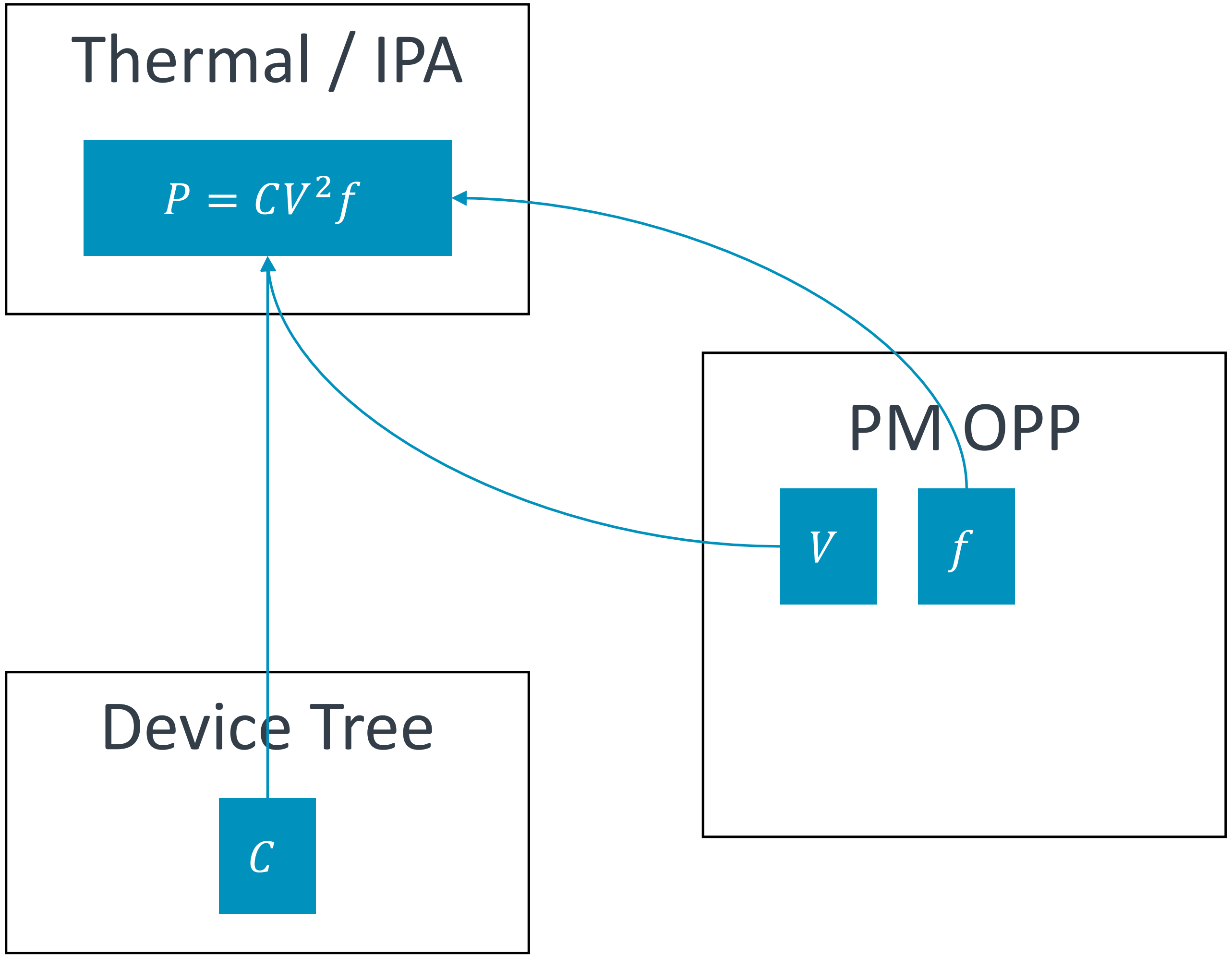
Managed by CPUFreq / OPP

Energy Model Comparison / Hikey960

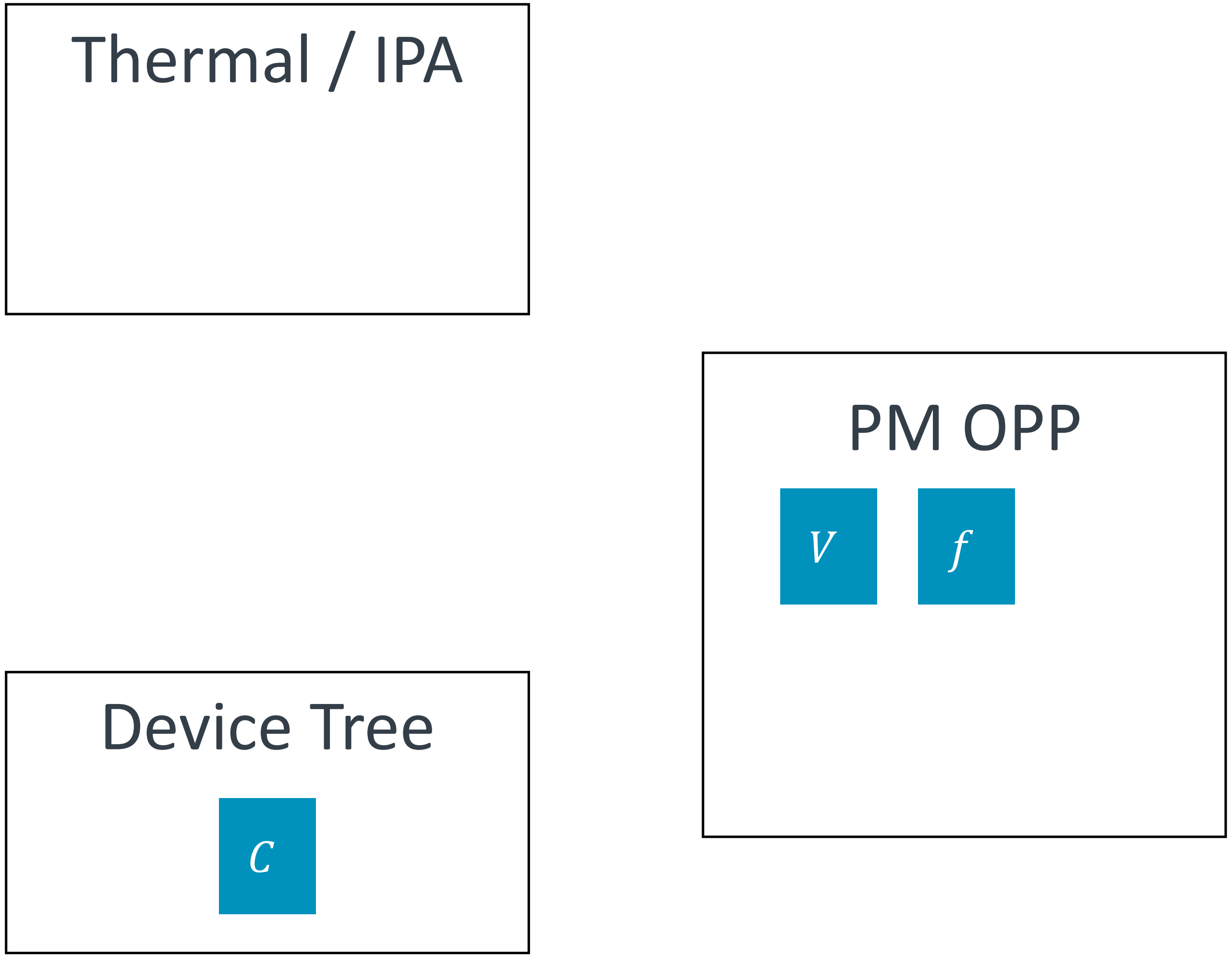


Architecture

Architecture



Architecture



Thermal / IPA

PM OPP

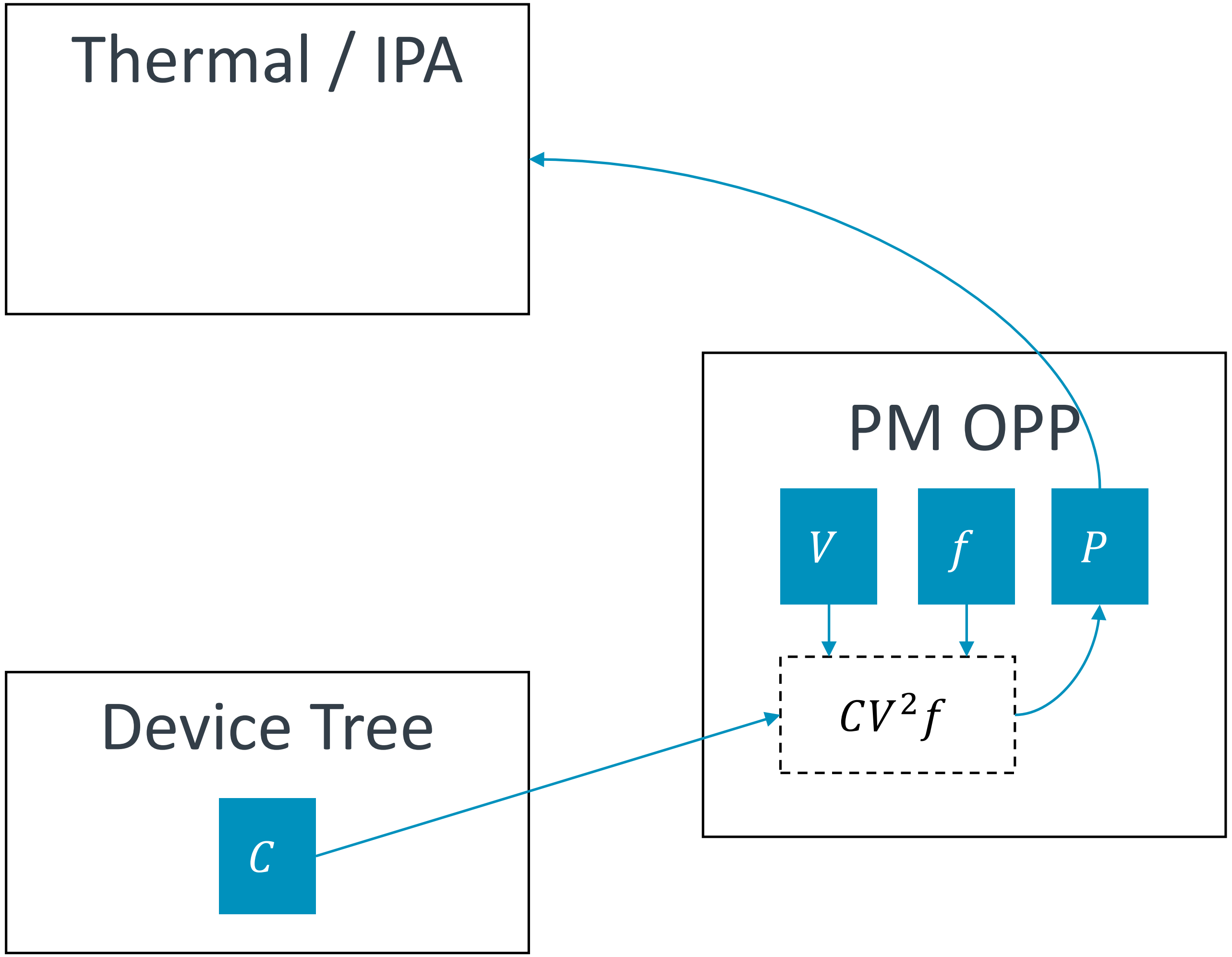
V

f

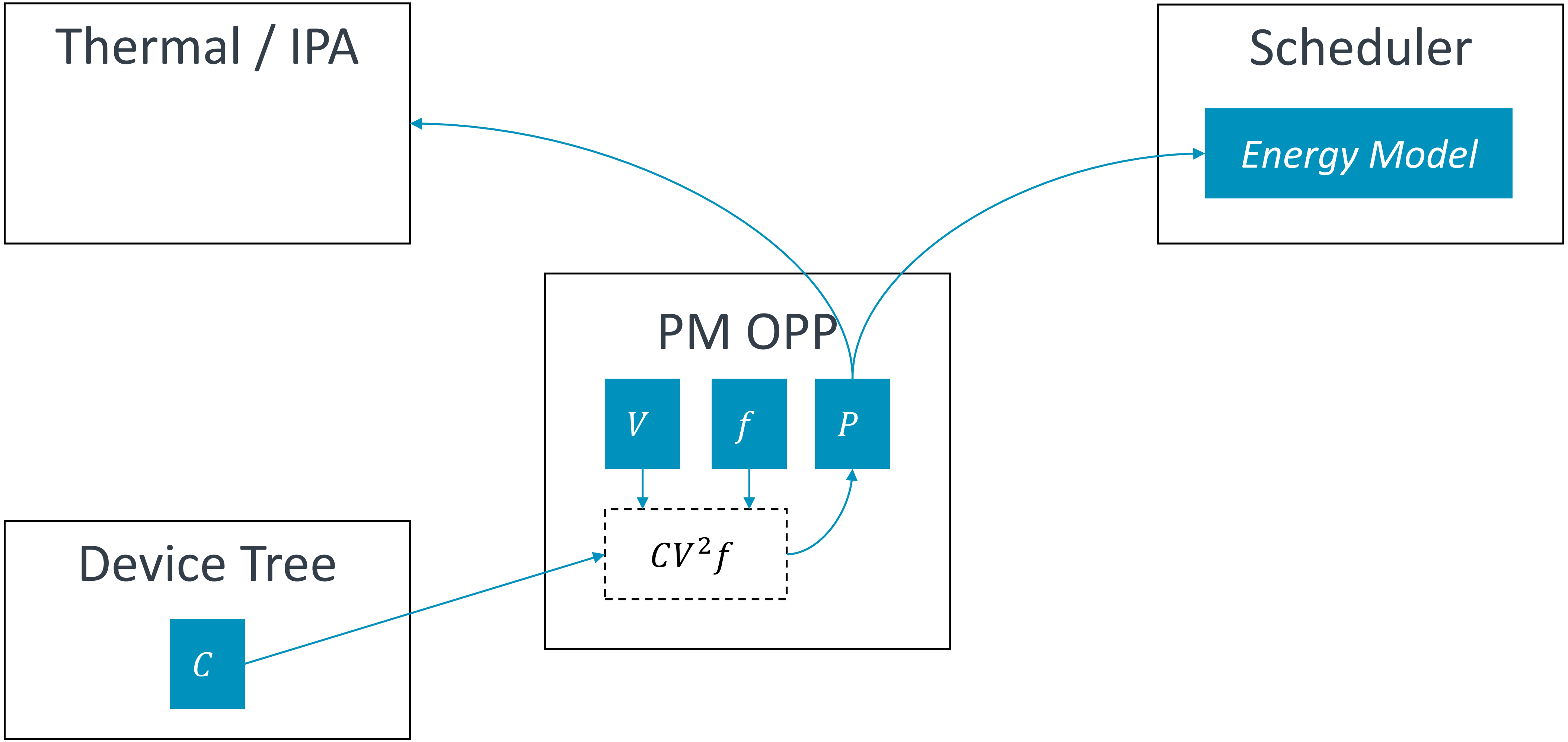
Device Tree

C

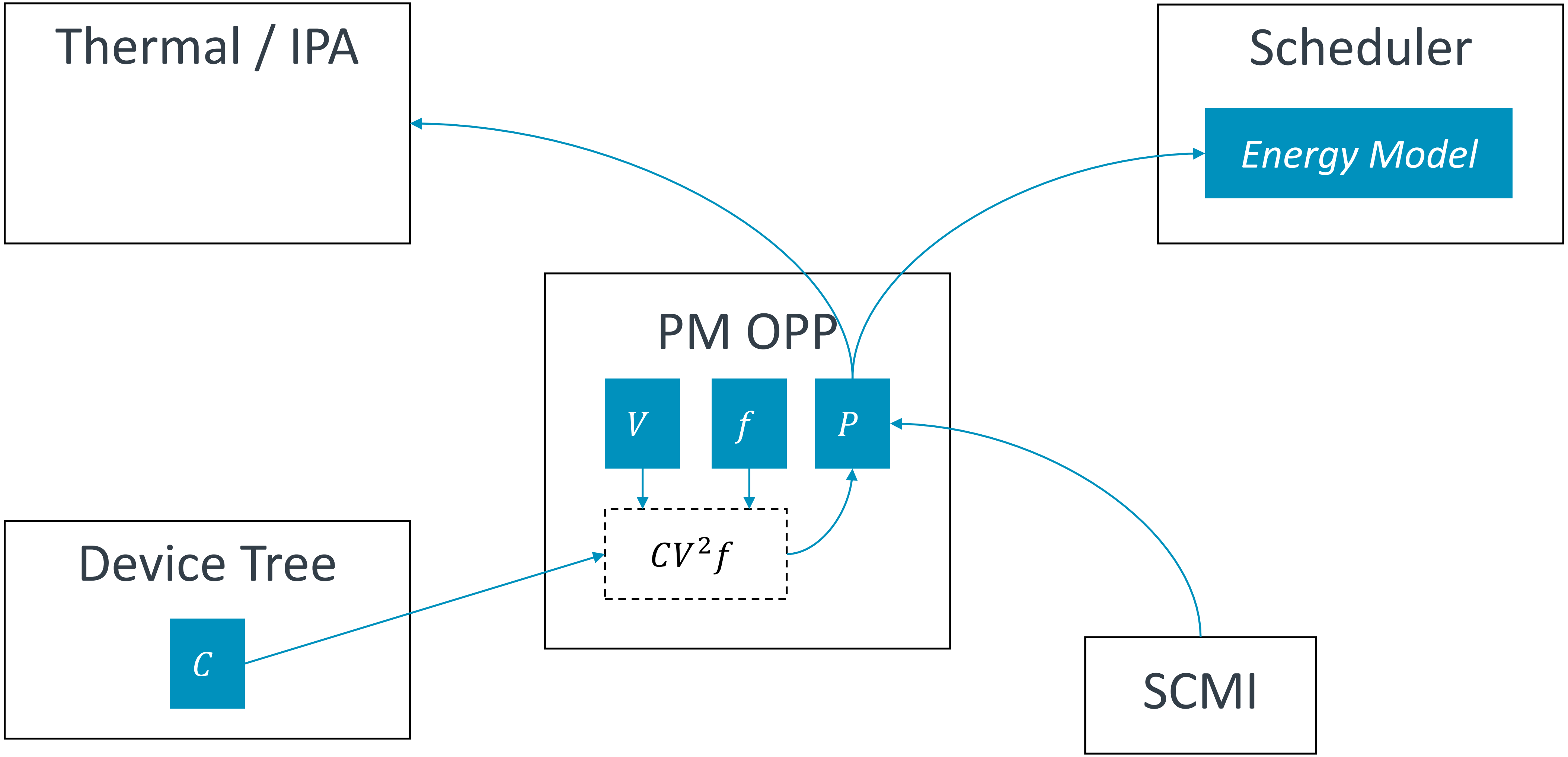
Architecture



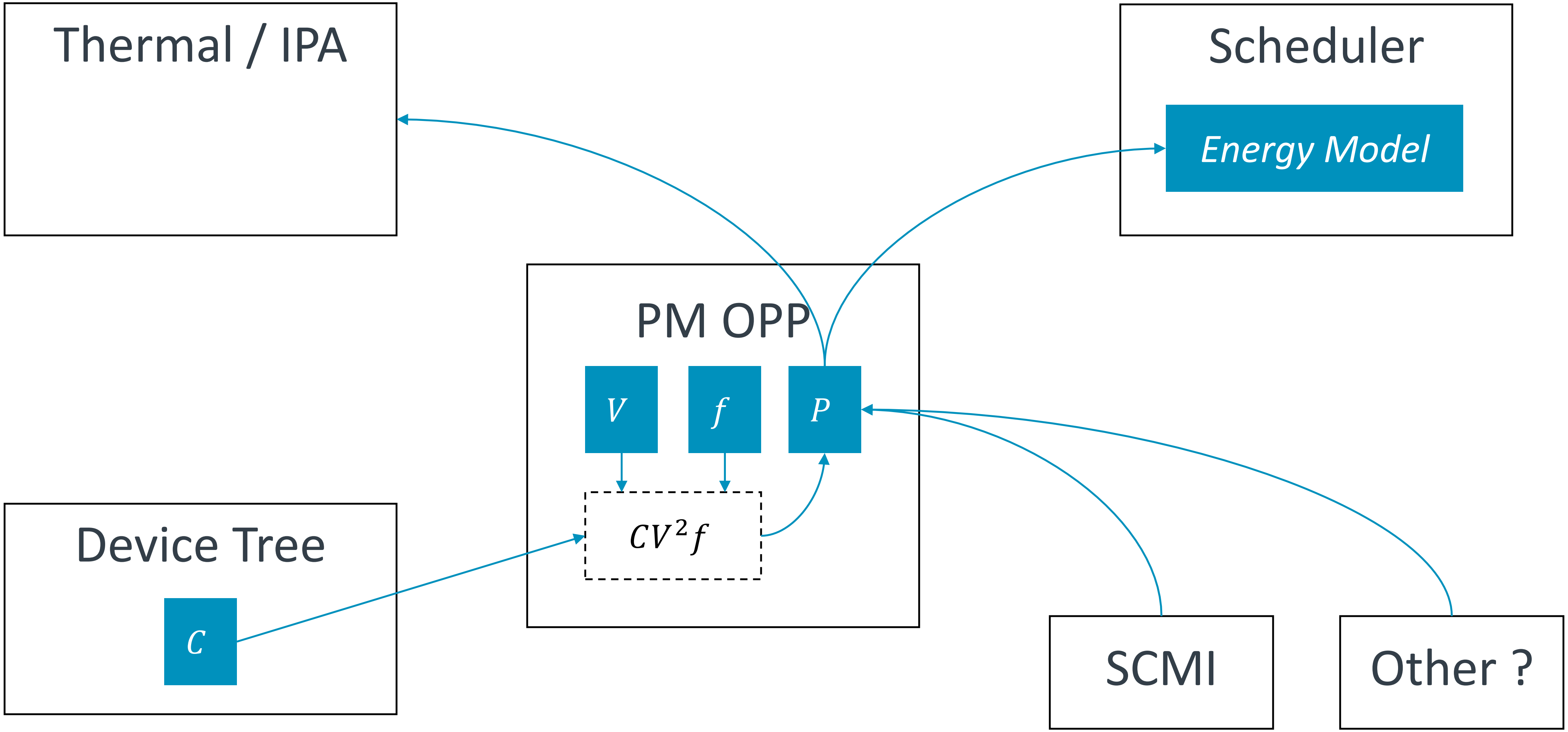
Architecture



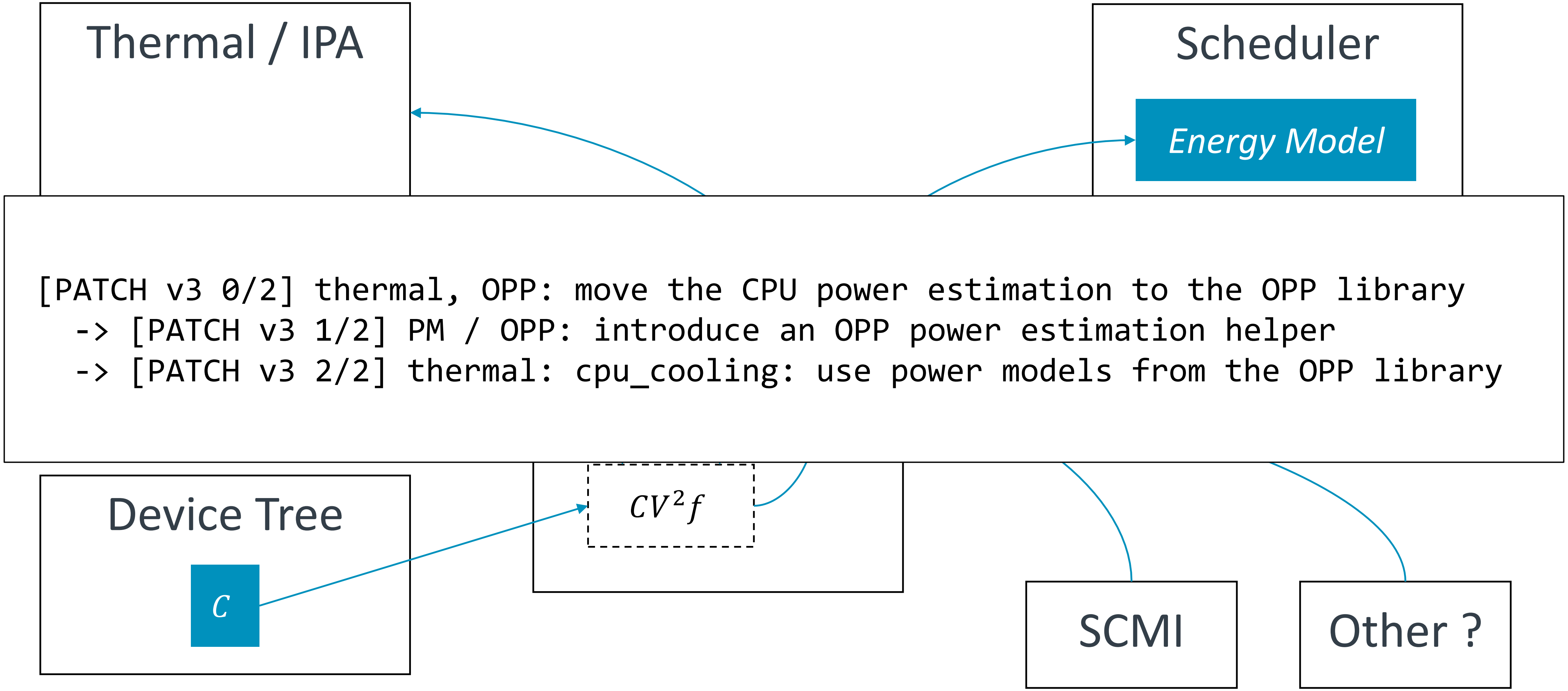
Architecture



Architecture



Architecture



```
[PATCH v3 0/2] thermal, OPP: move the CPU power estimation to the OPP library
  -> [PATCH v3 1/2] PM / OPP: introduce an OPP power estimation helper
  -> [PATCH v3 2/2] thermal: cpu_cooling: use power models from the OPP library
```

Implementation

- No hierarchical data, no need to use the scheduling domains
- Data structures:
  - Loaded from PM / OPP at boot time, after CPUfreq is setup
  - Energy models are stored in a flat per-cpu array
  - Frequency domains are stored in cpu-masks

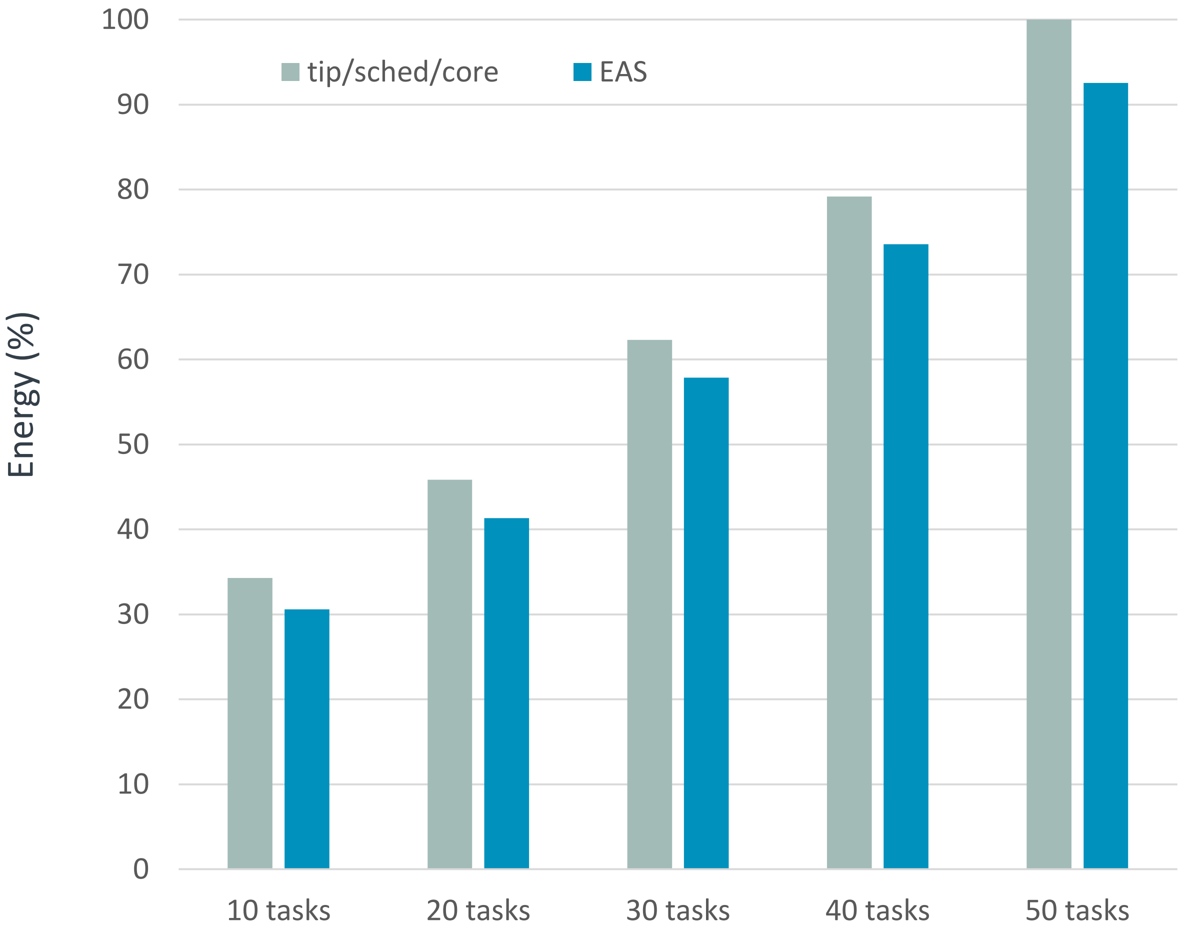
Assumptions

- All CPUs in a freq. domain share capacity states
    - All CPUs in a freq. domain have the same micro-architecture
    - Possible to relax this if needed, but higher computational cost
- EAS enabled for asymmetric platforms only (SD_ASYM_CPUCAPACITY flag set)
    - EAS shines on heterogeneous platforms
    - Avoid “conflicts” for purely perf-oriented platforms (servers, …)

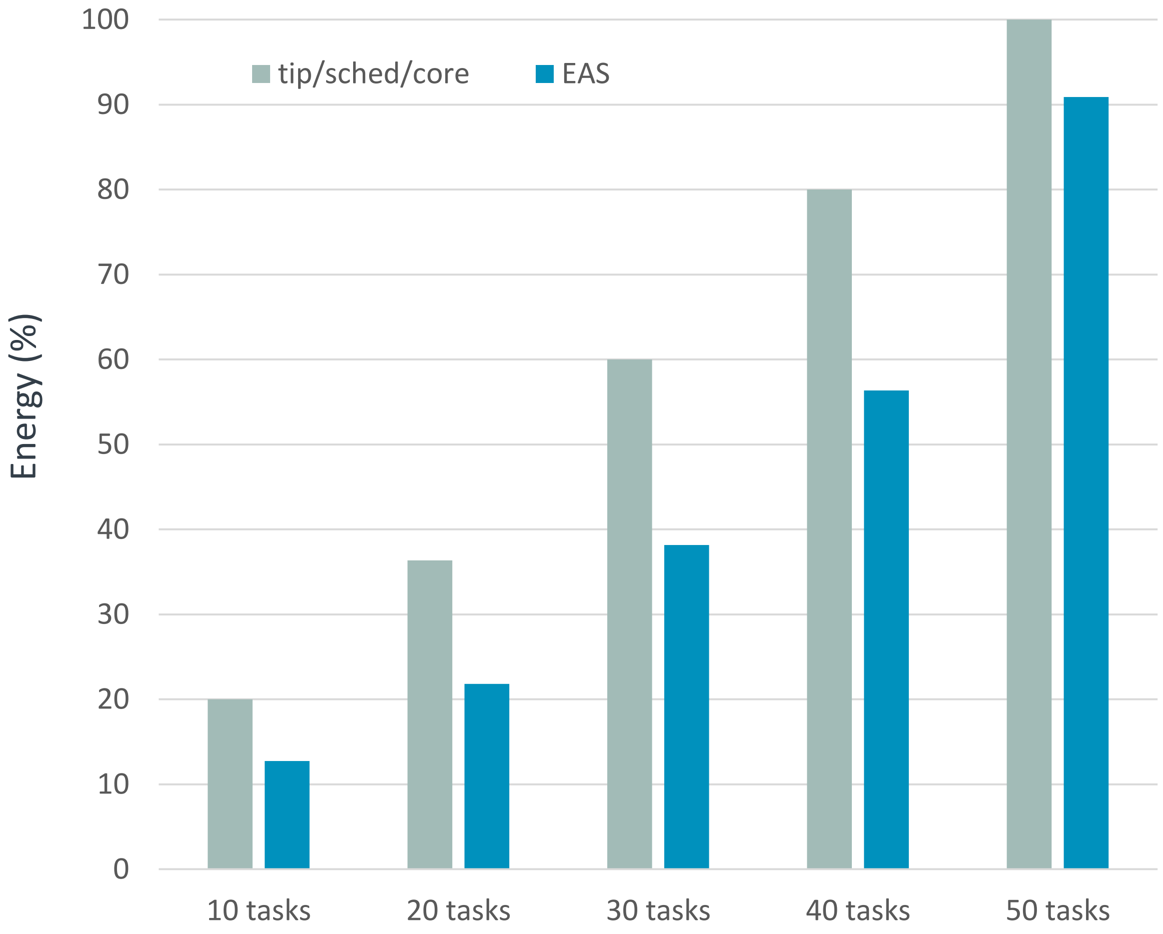
Tests against mainline

- Test setup:
  - Platform: Hikey960 and Juno r0
  - Debian userspace
  - Base kernel: tip/sched/core – 4.16-rc2
- Test cases:
  - **Energy:** “X” RTApp tasks, 16ms period, 5% duty cycle, 30 seconds
  - **Performance:** `perf bench sched messaging -pipe -thread -group X -loop 50000`

Tests against mainline / Energy

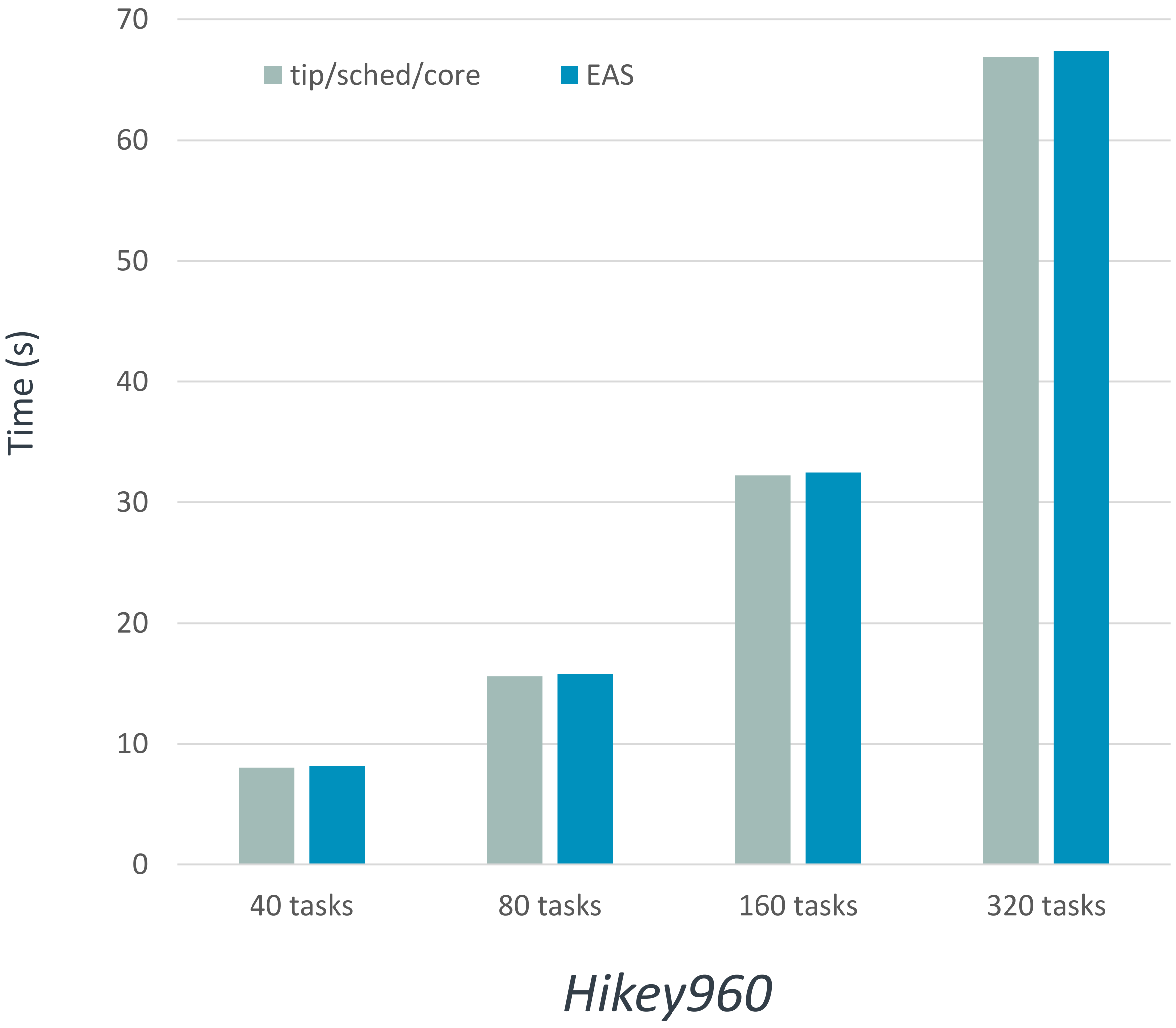


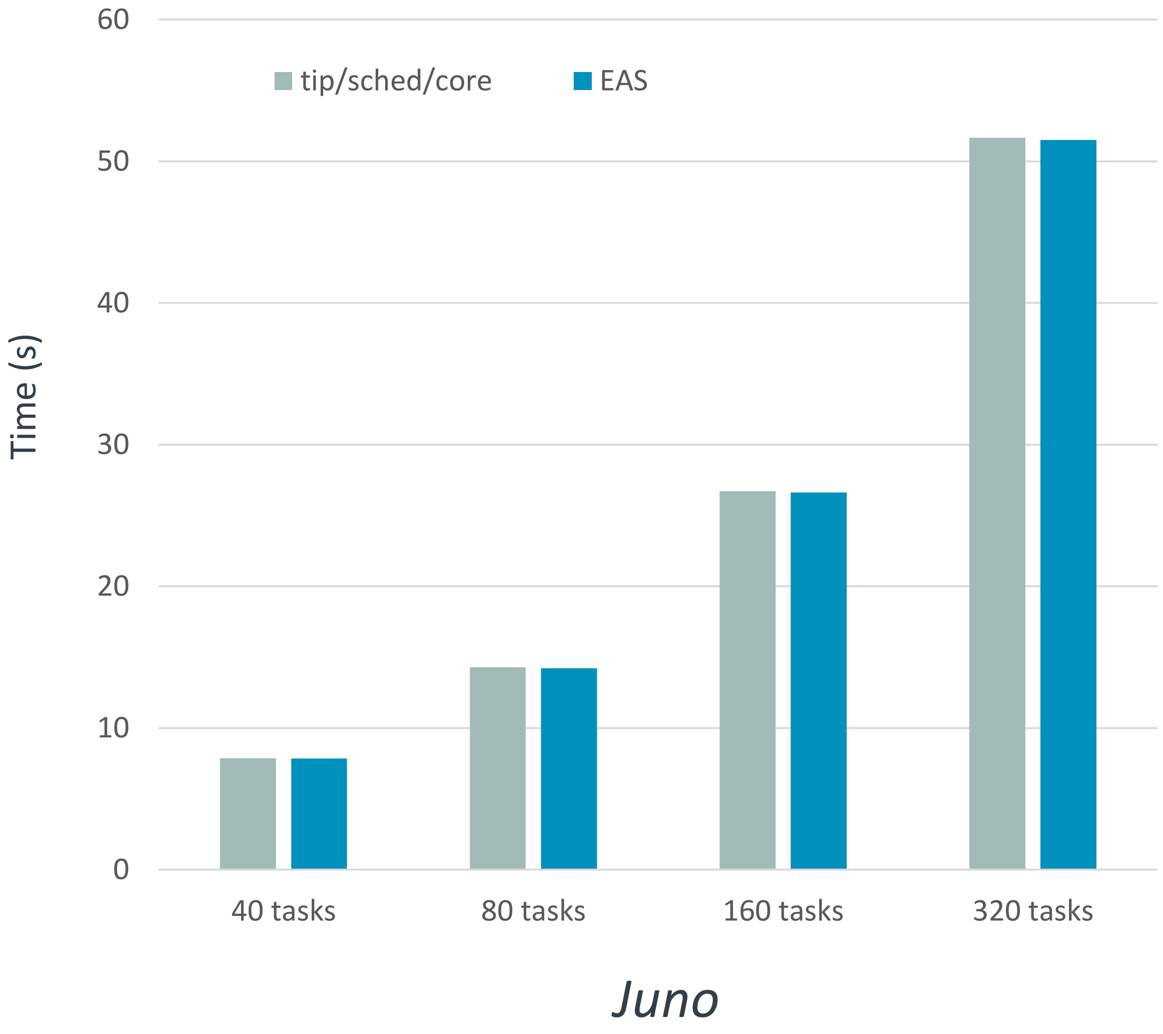
Hikey960 (ACME / full SoC + memory)



Juno (HW monitor / b.L CPUs only)

Tests against mainline / Perf.





Posted to LKML this week

[RFC PATCH 0/6] Energy Aware Scheduling

[RFC PATCH 1/6] sched/fair: Create util_fits_capacity()

[RFC PATCH 2/6] sched: Introduce energy models of CPUs

[RFC PATCH 3/6] sched: Add over-utilization/tipping point indicator

[RFC PATCH 4/6] sched/fair: Introduce an energy estimation helper …

[RFC PATCH 5/6] sched/fair: Select an energy-efficient CPU on task …

[RFC PATCH 6/6] drivers: base: arch_topology.c: Enable EAS for arm/…

Summary

1. Today’s Energy Model
2. Which simplified Energy Model ?
3. Mainline implementation
4. Conclusion

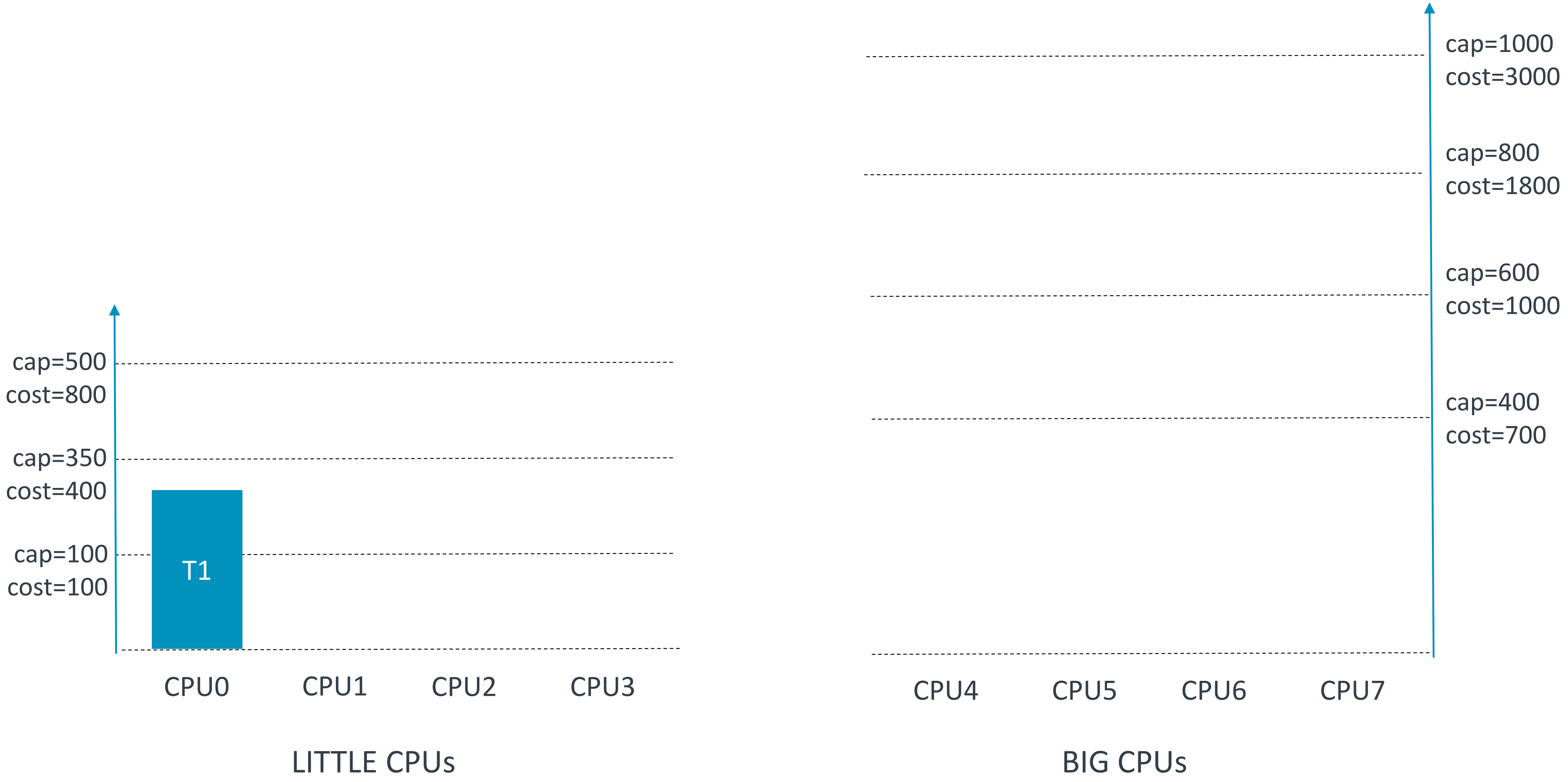
Next steps

- Ideal scenario: simplified EM goes in the next LTS (4.19 ?)
- Test & assessment on android-4.14
- In case of gaps with the full EM, they will be filled in product

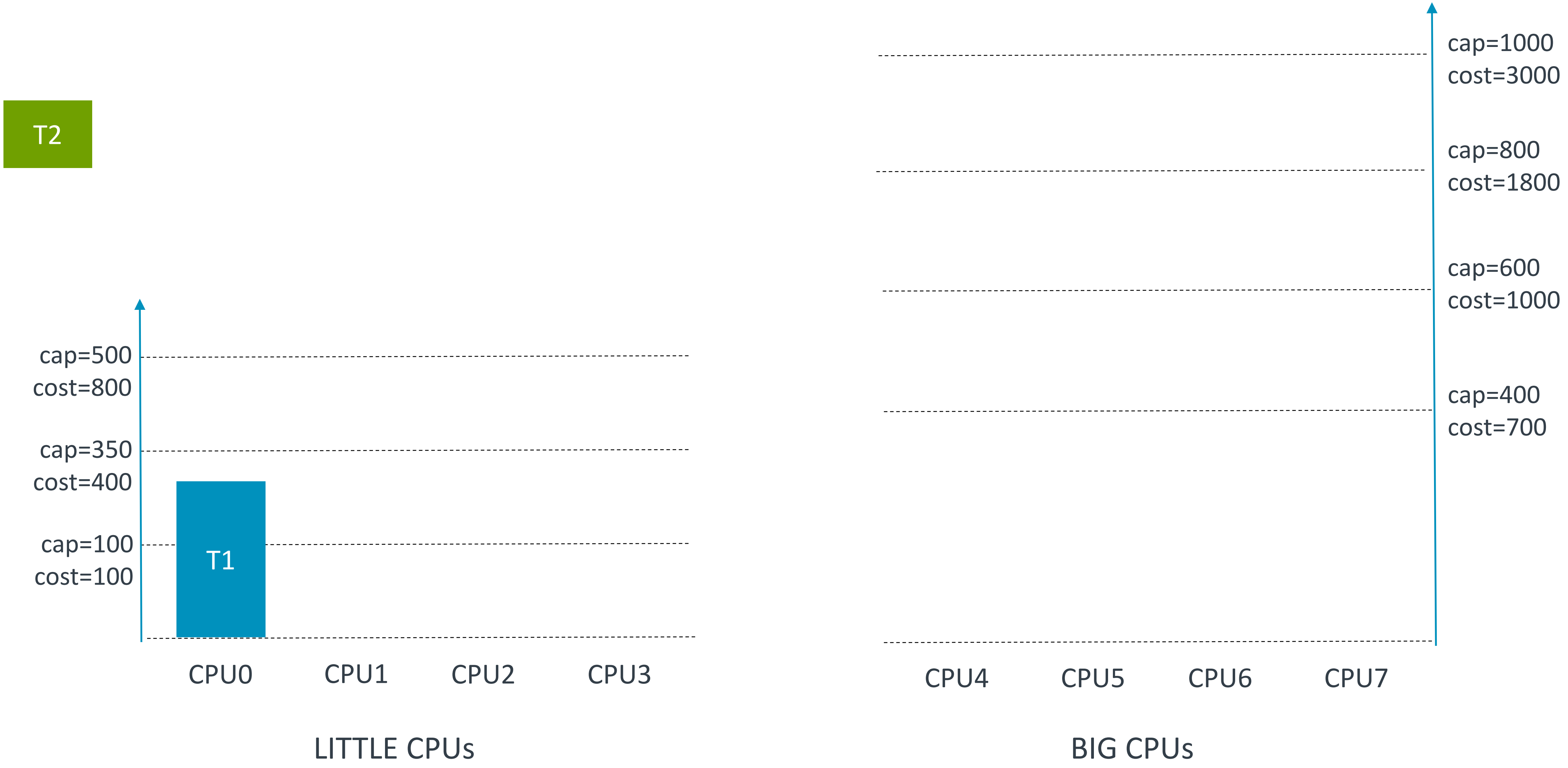
Thanks.

Any questions ?

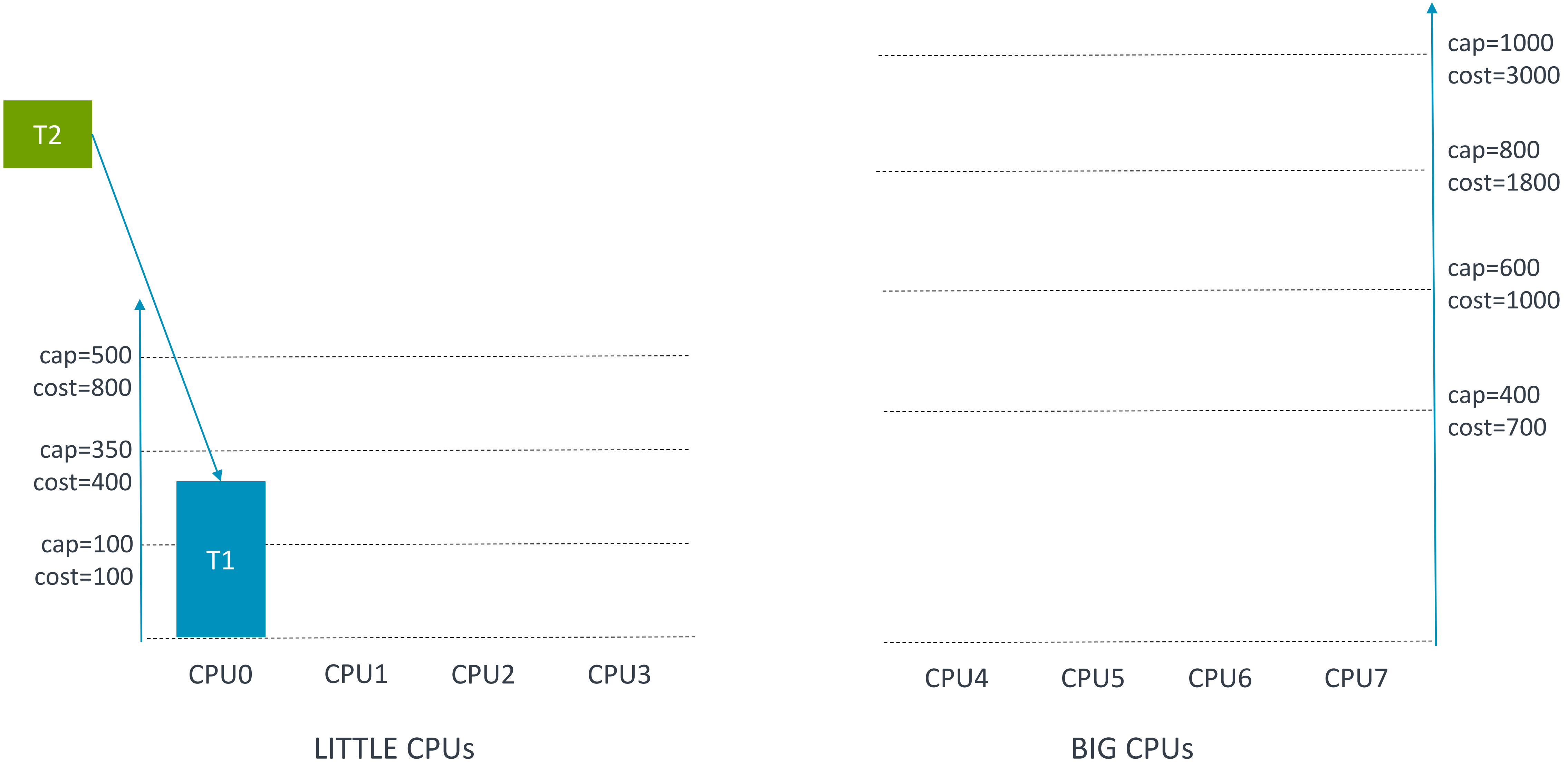
Algorithm complexity



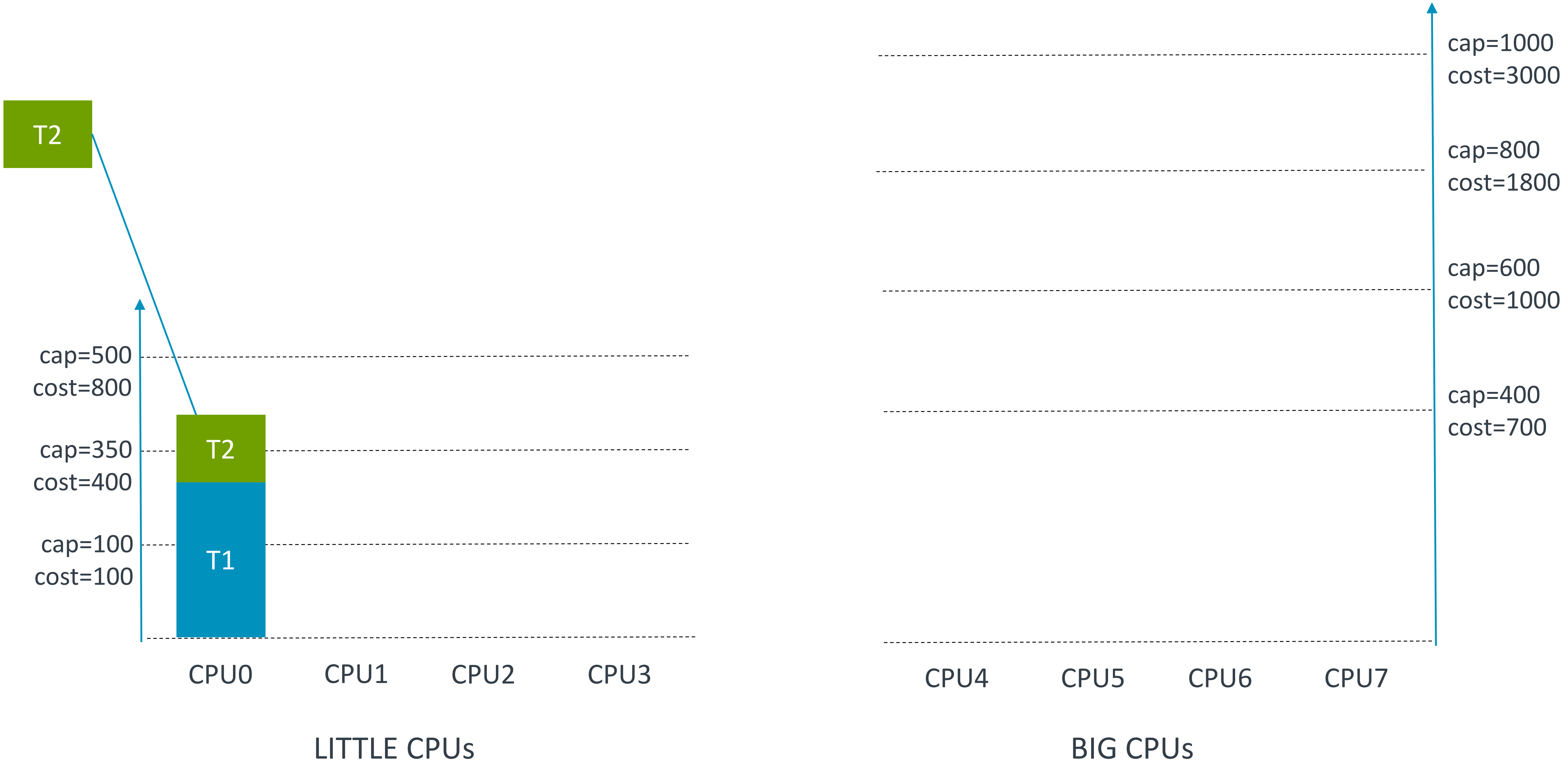
Algorithm complexity



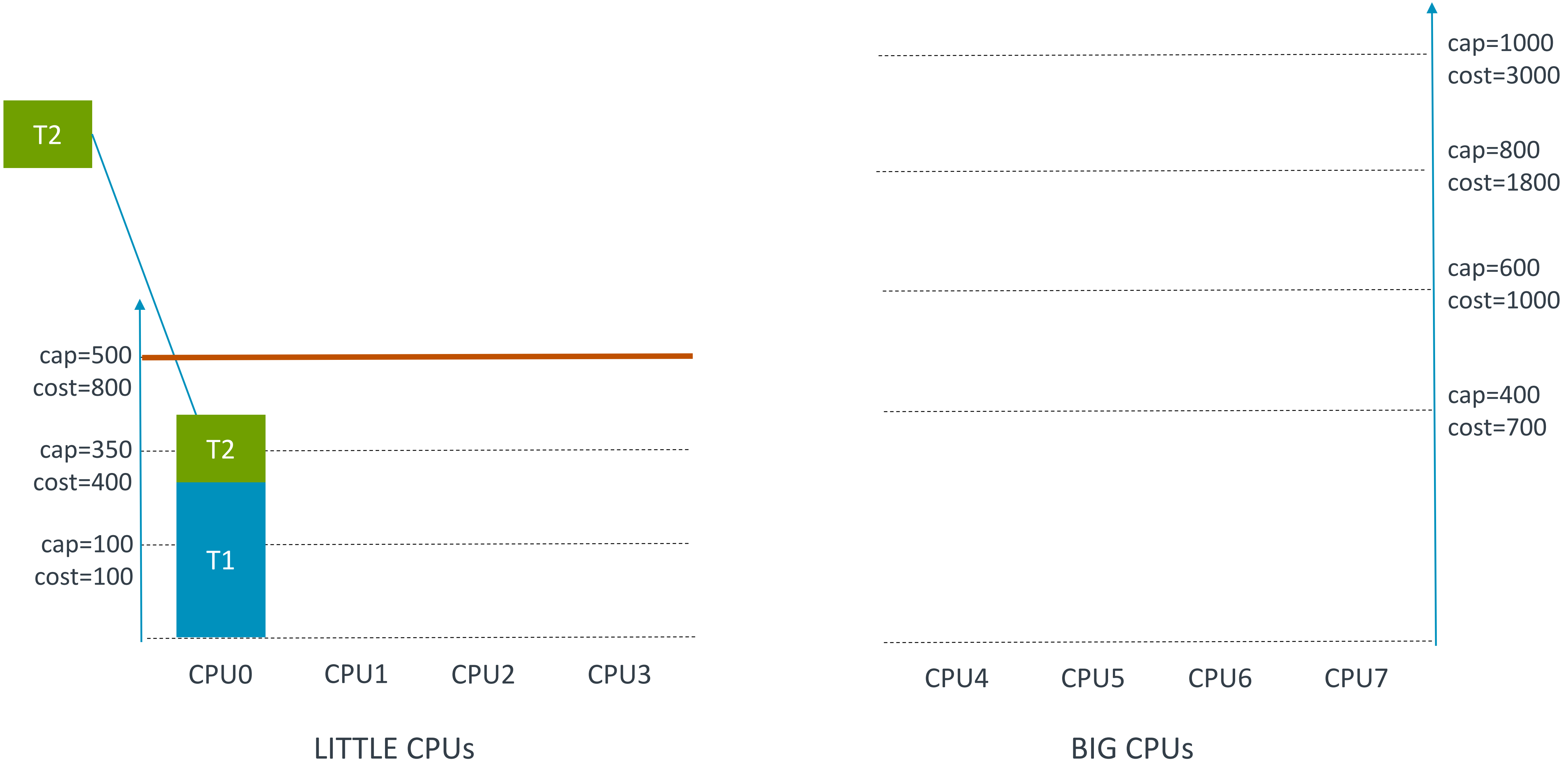
Algorithm complexity



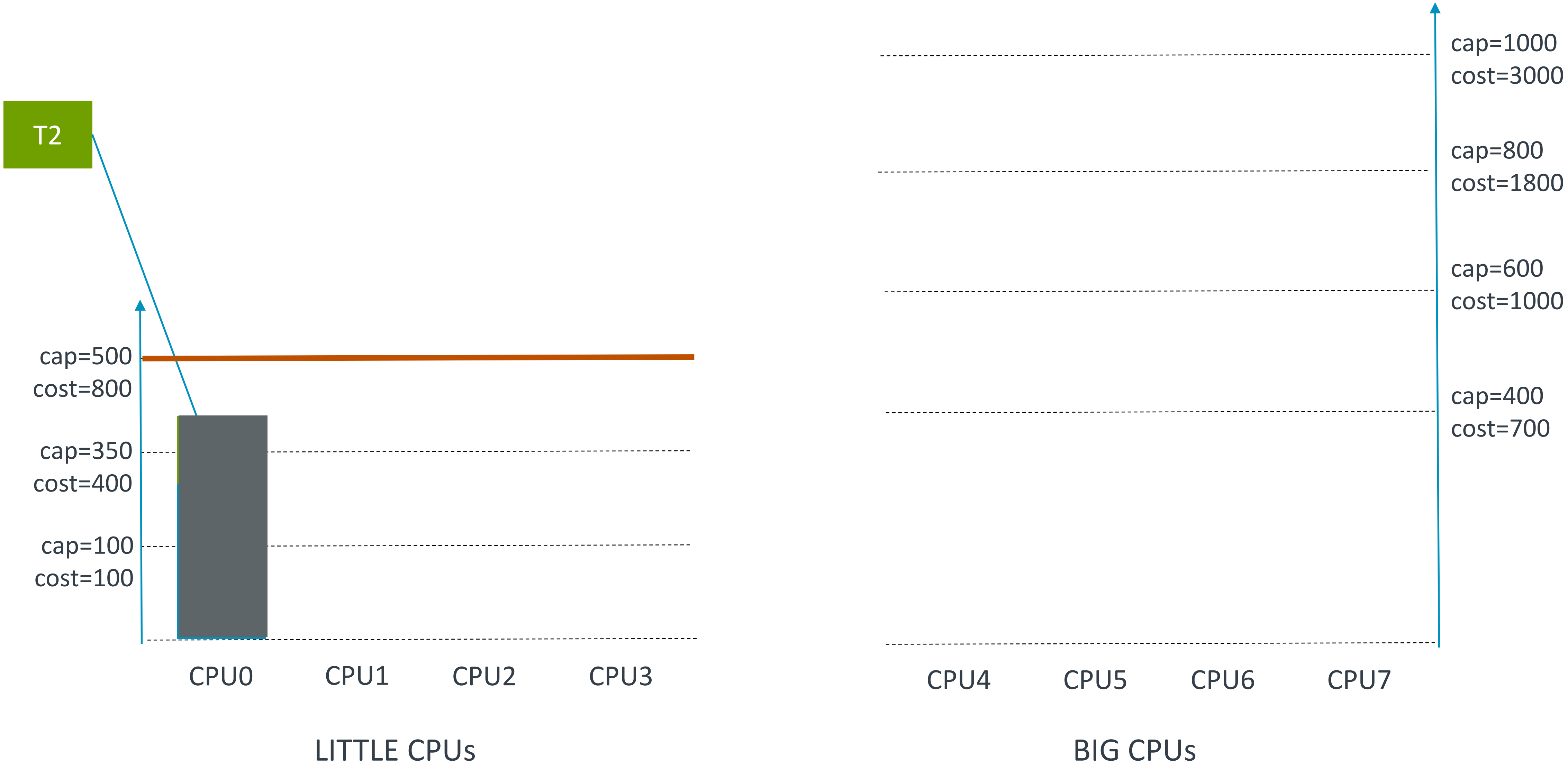
Algorithm complexity



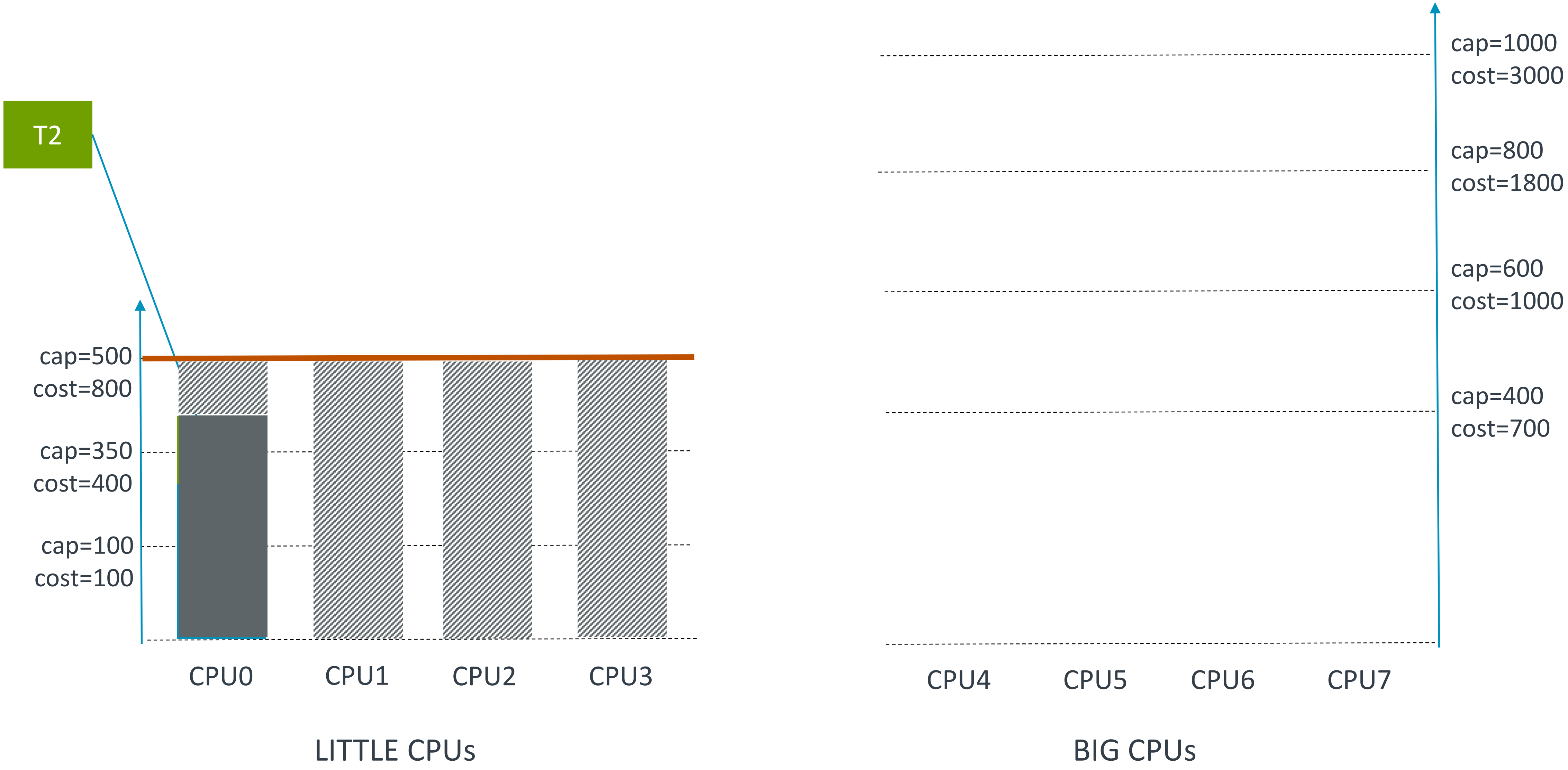
Algorithm complexity



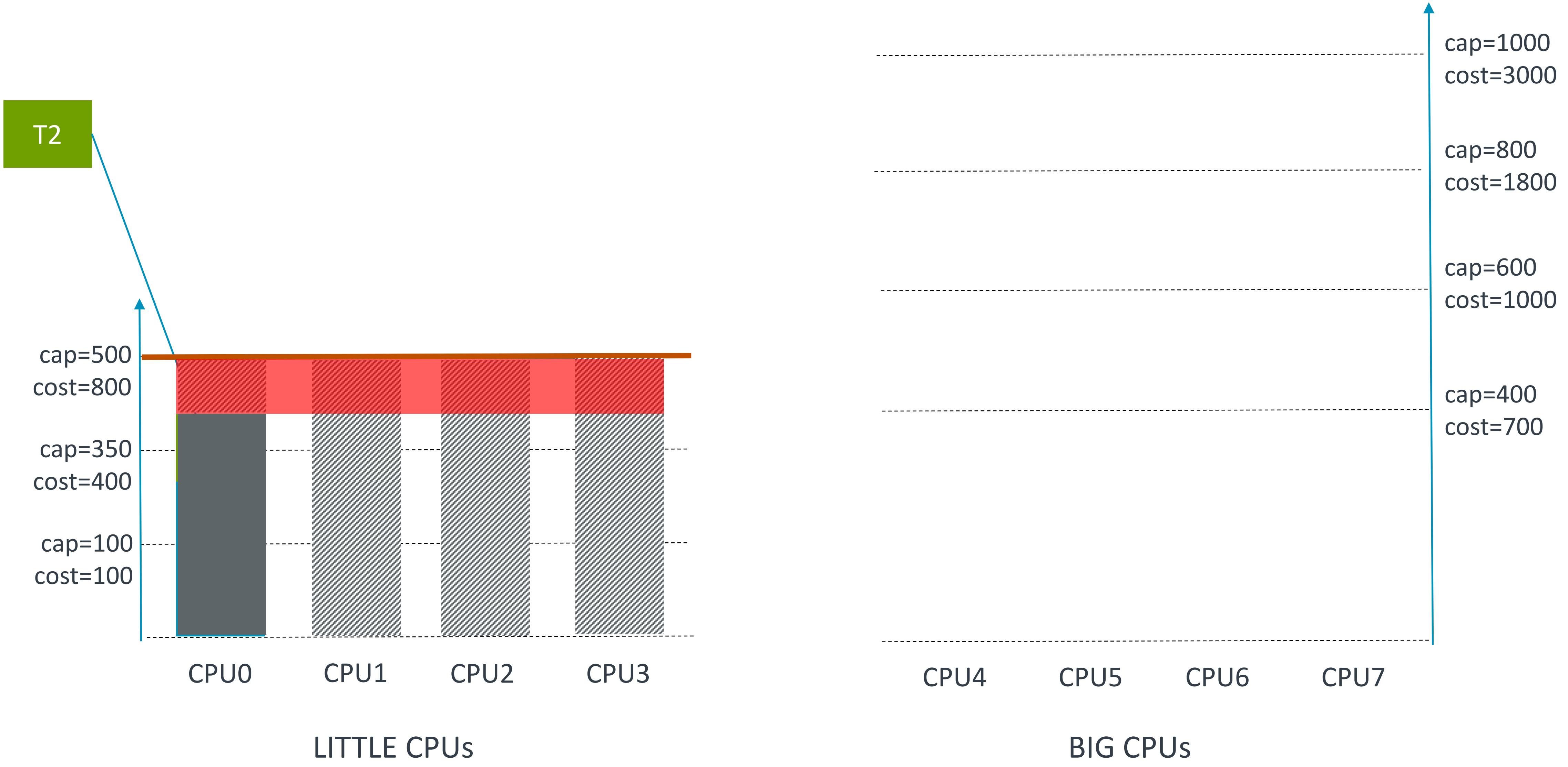
Algorithm complexity



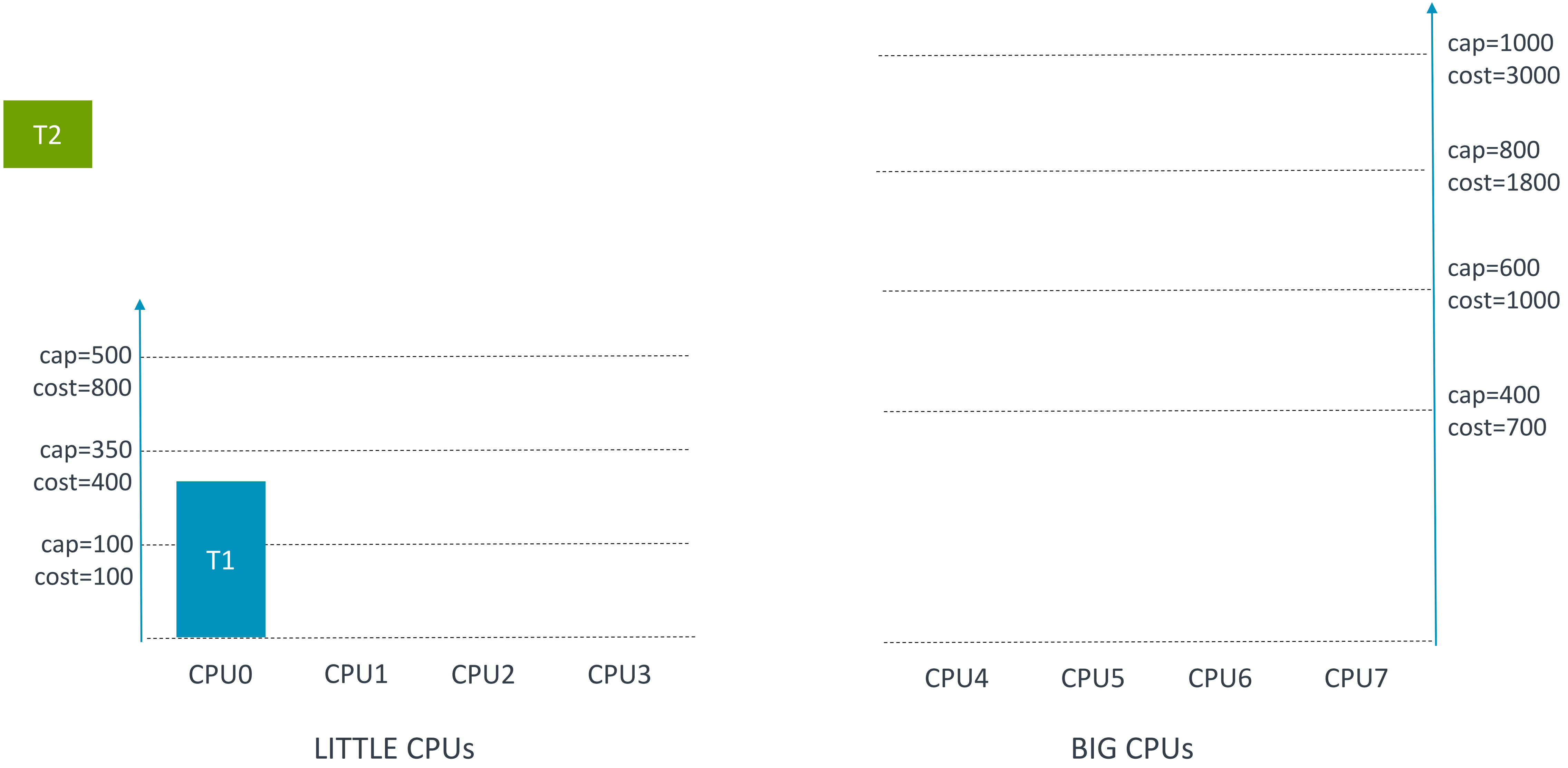
Algorithm complexity



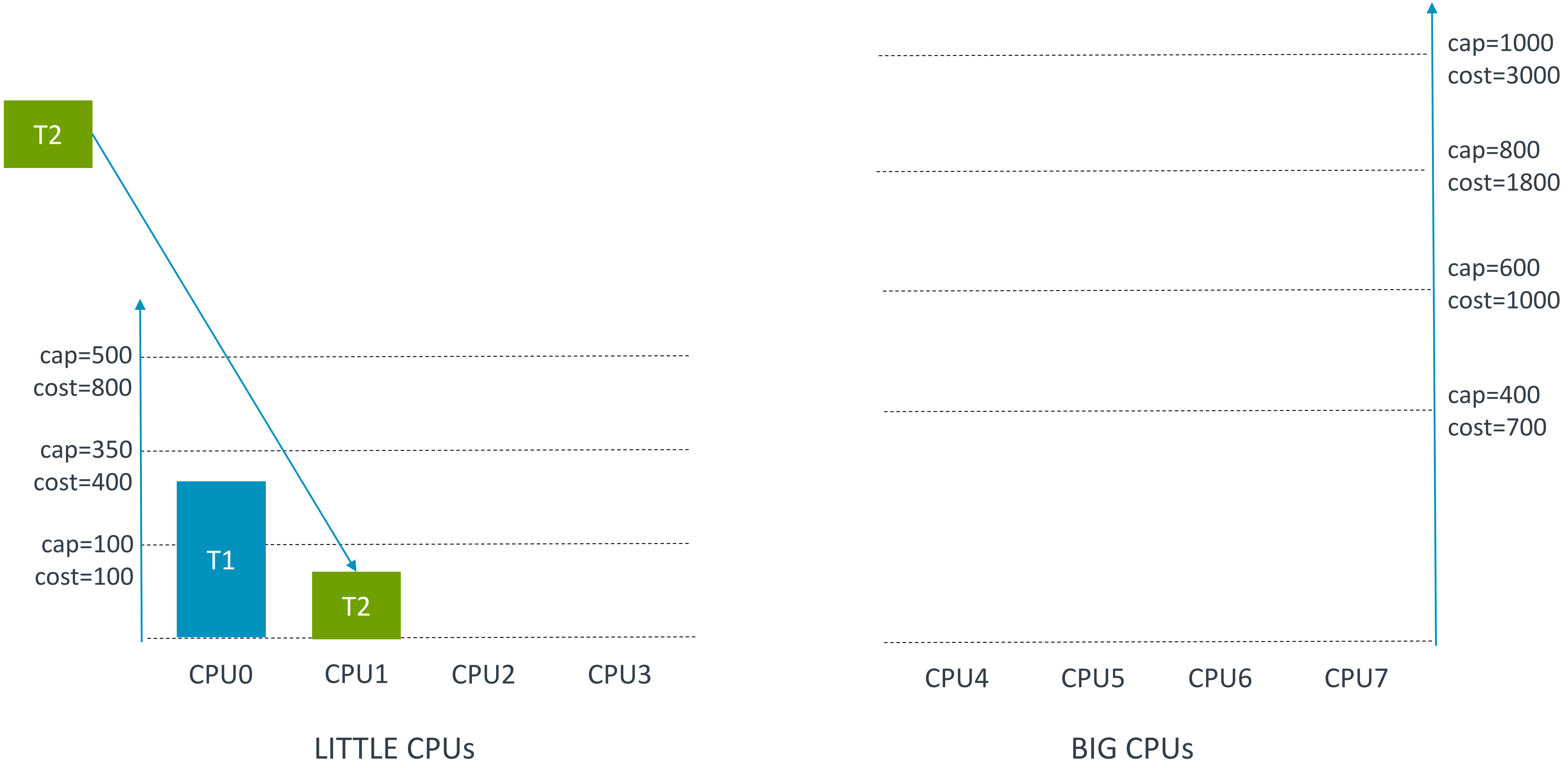
Algorithm complexity



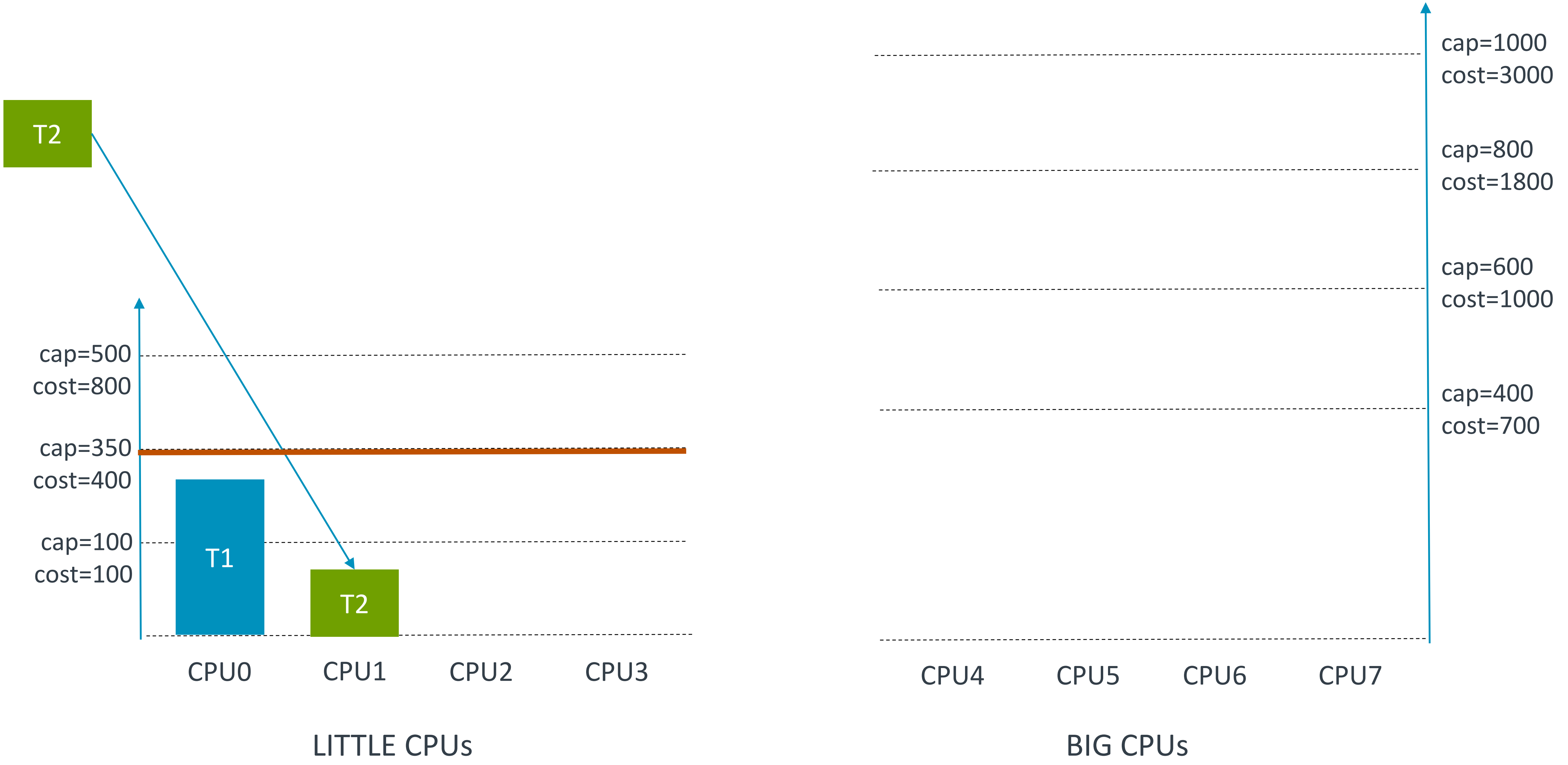
Algorithm complexity



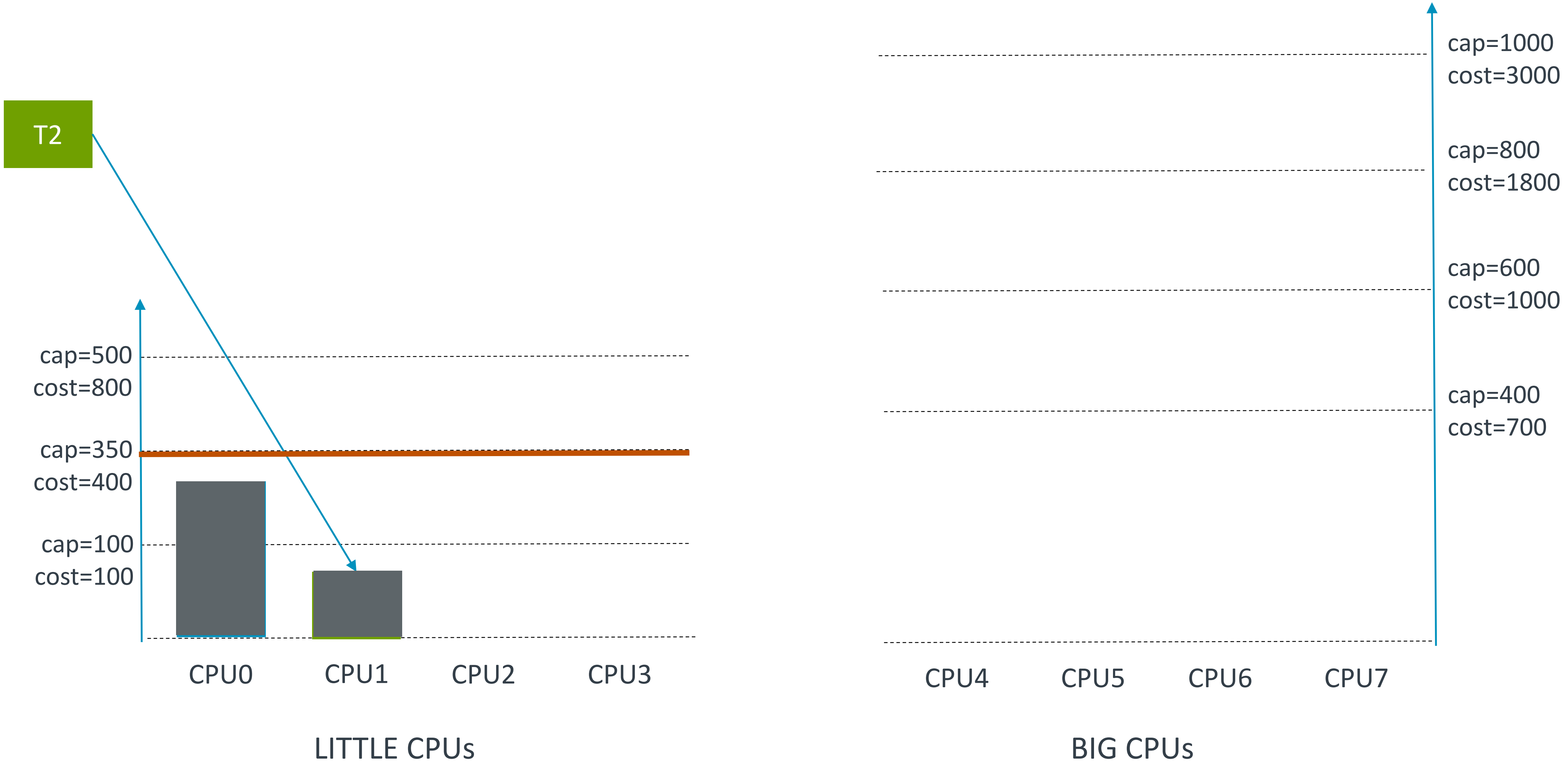
Algorithm complexity



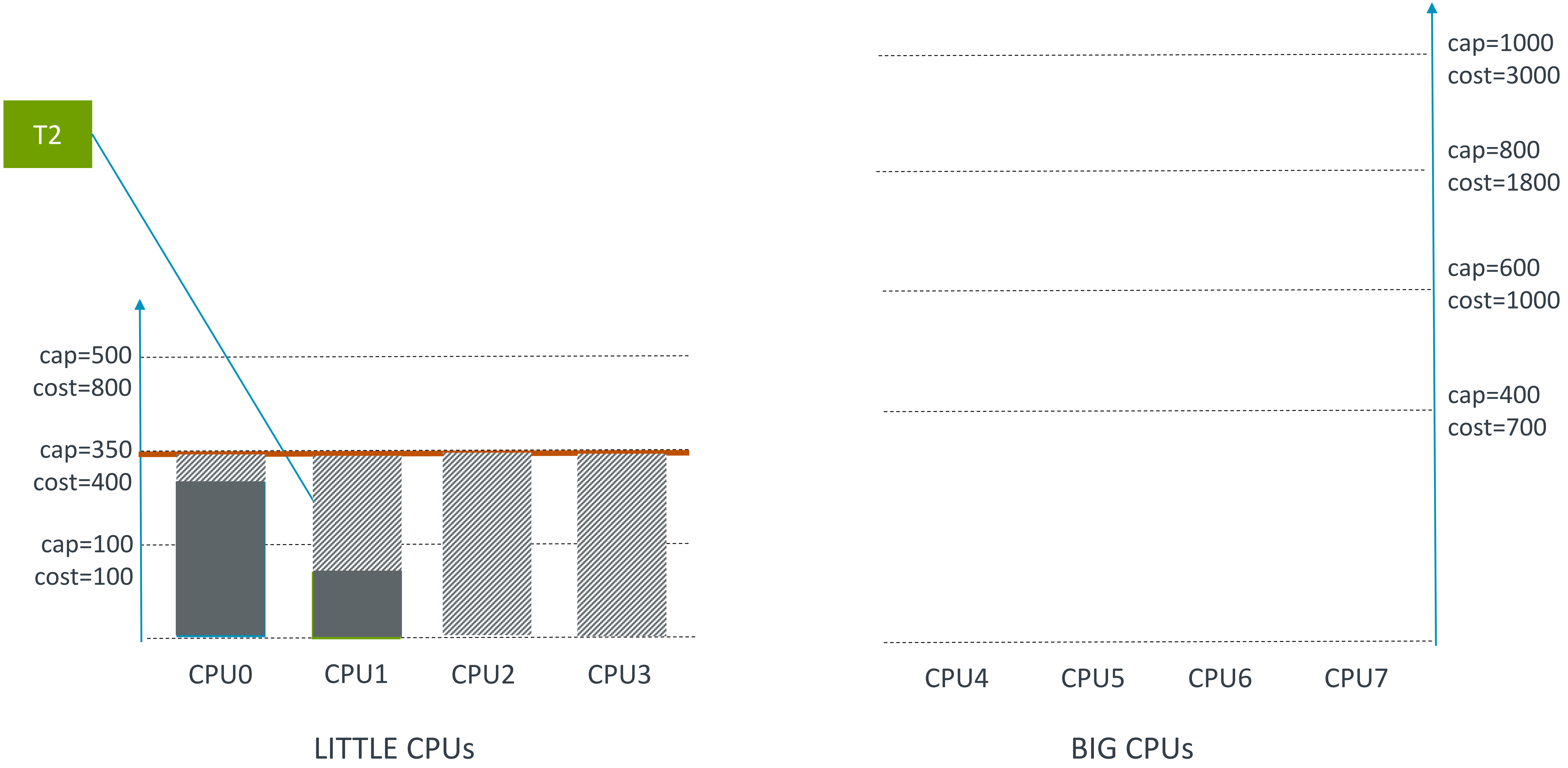
Algorithm complexity



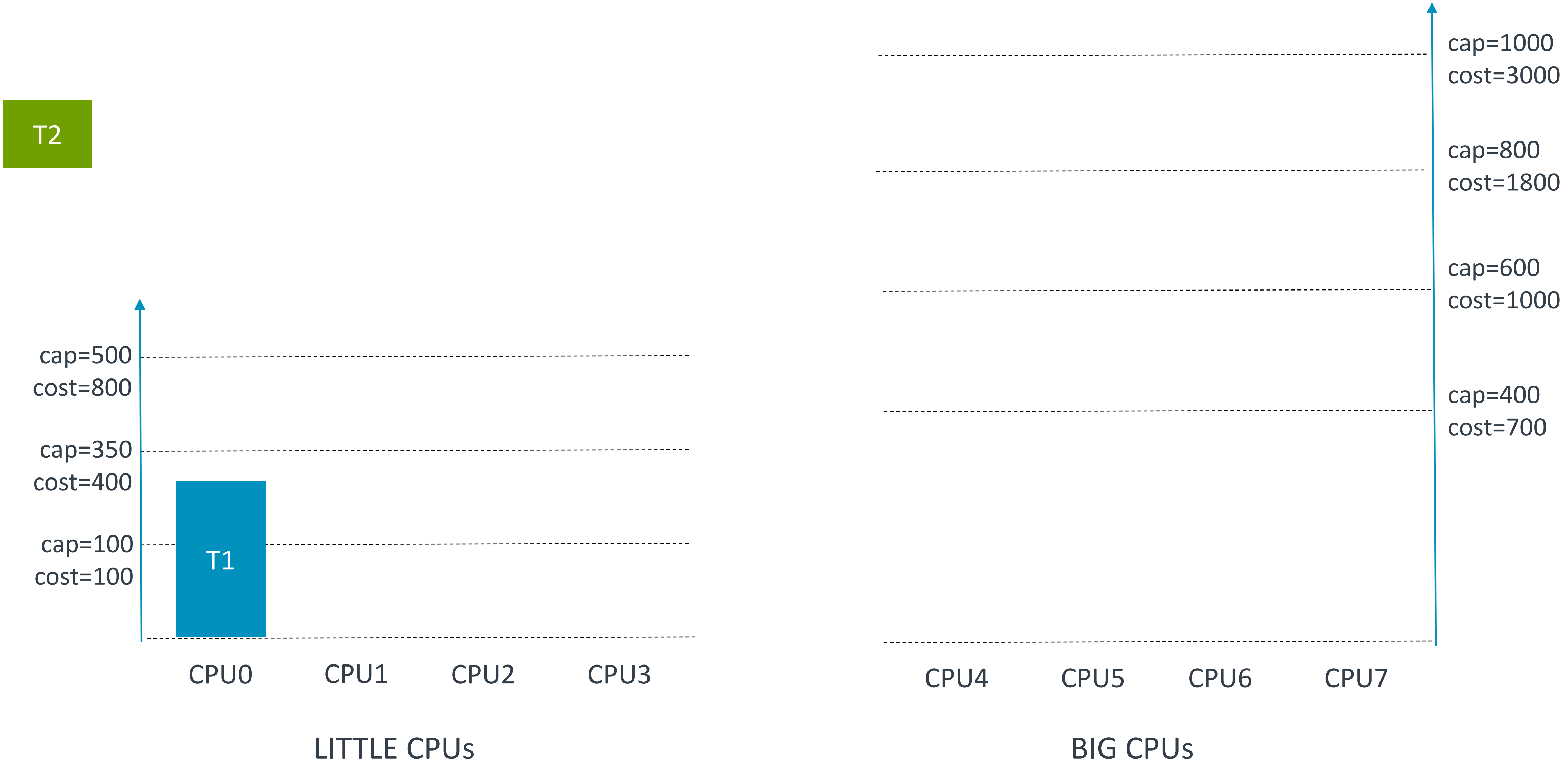
Algorithm complexity



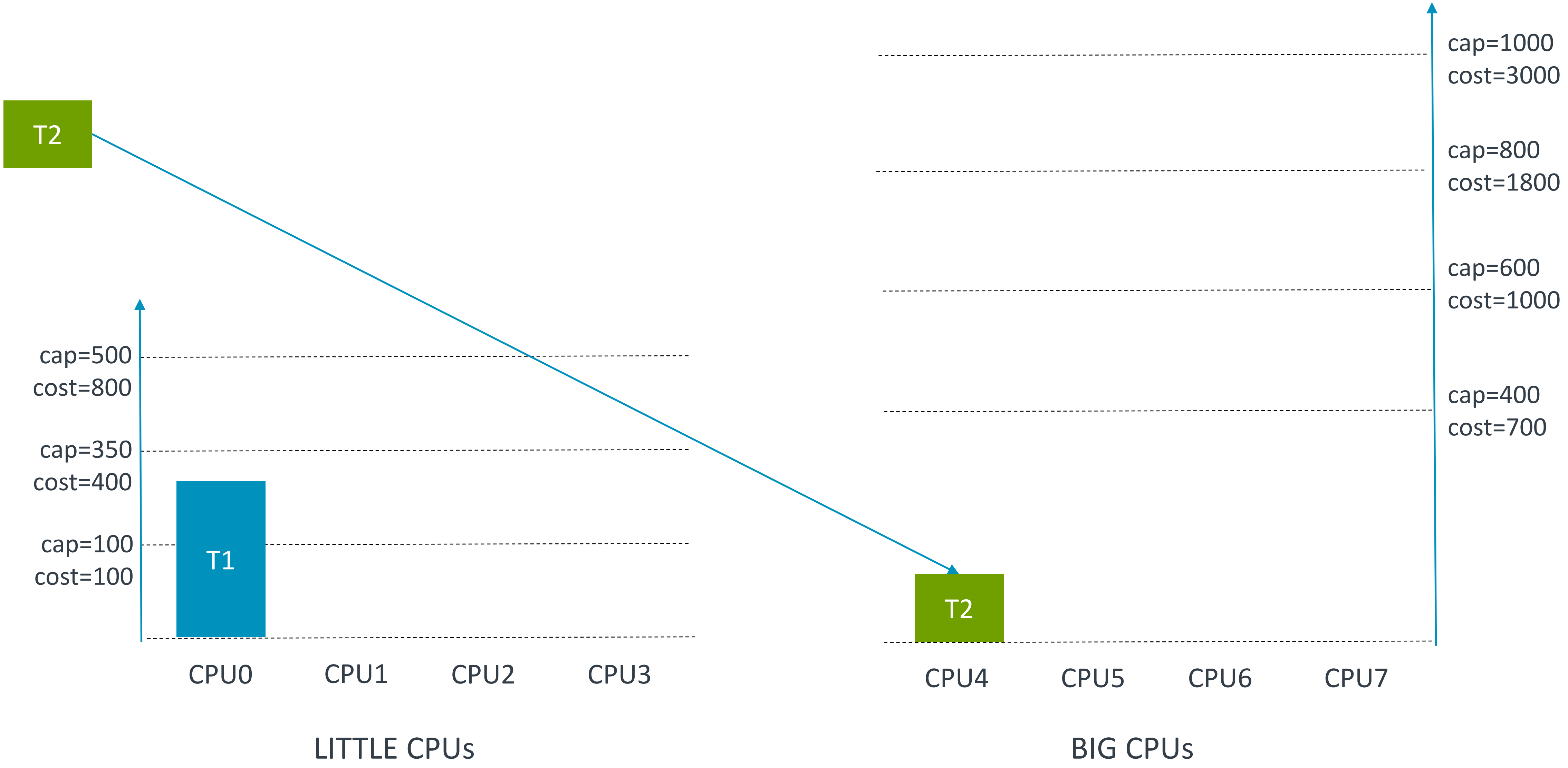
Algorithm complexity



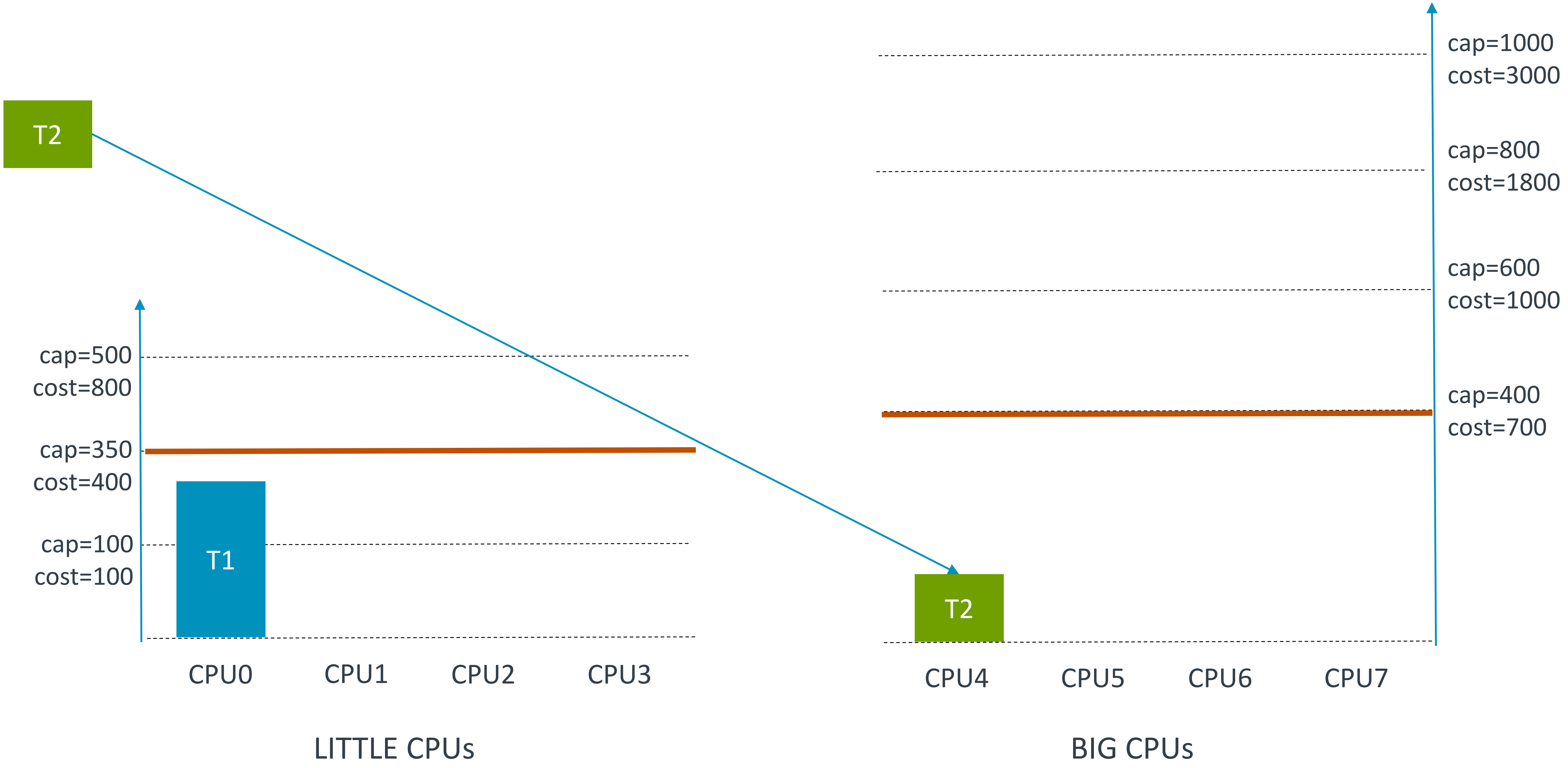
Algorithm complexity



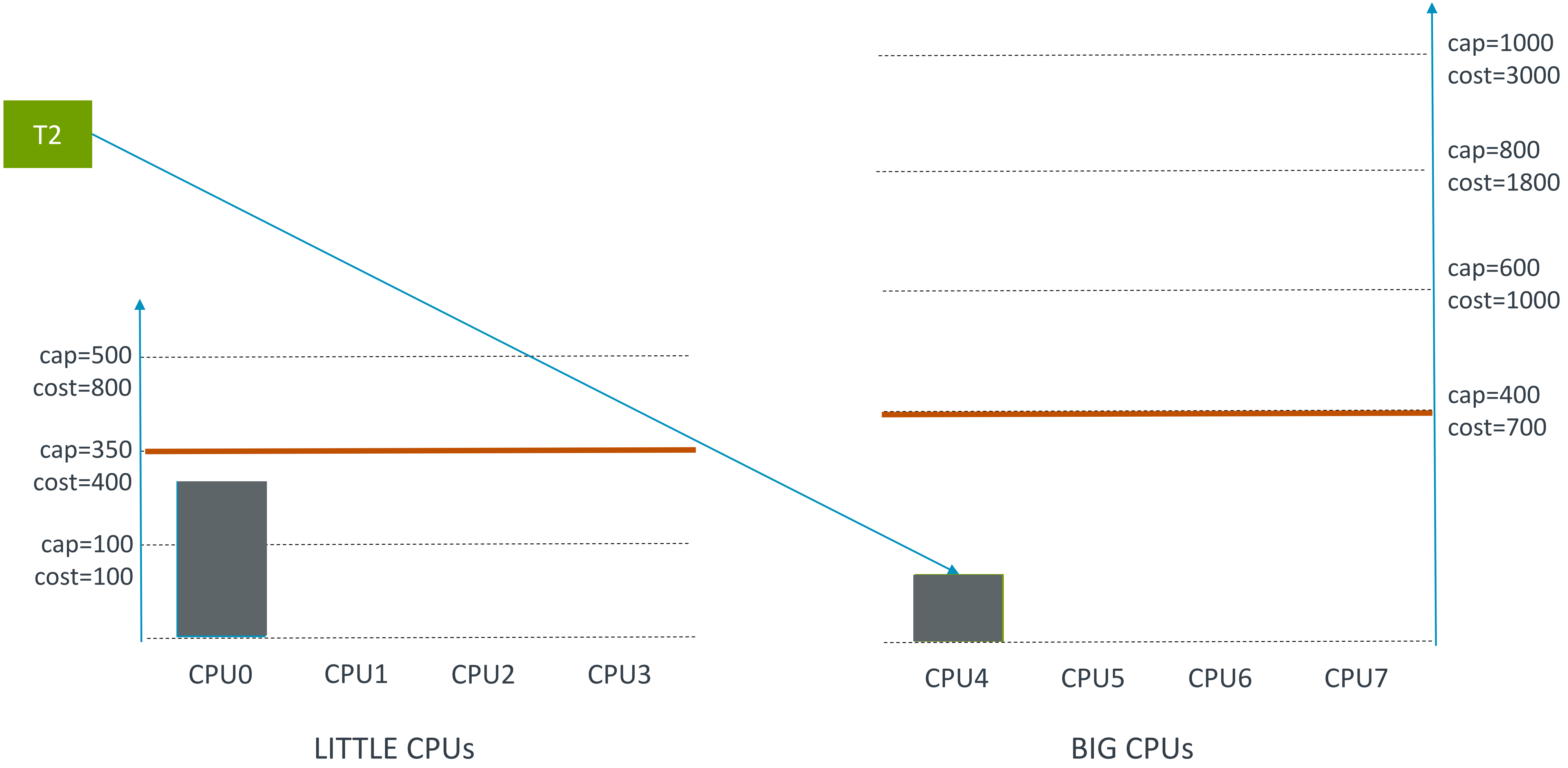
Algorithm complexity



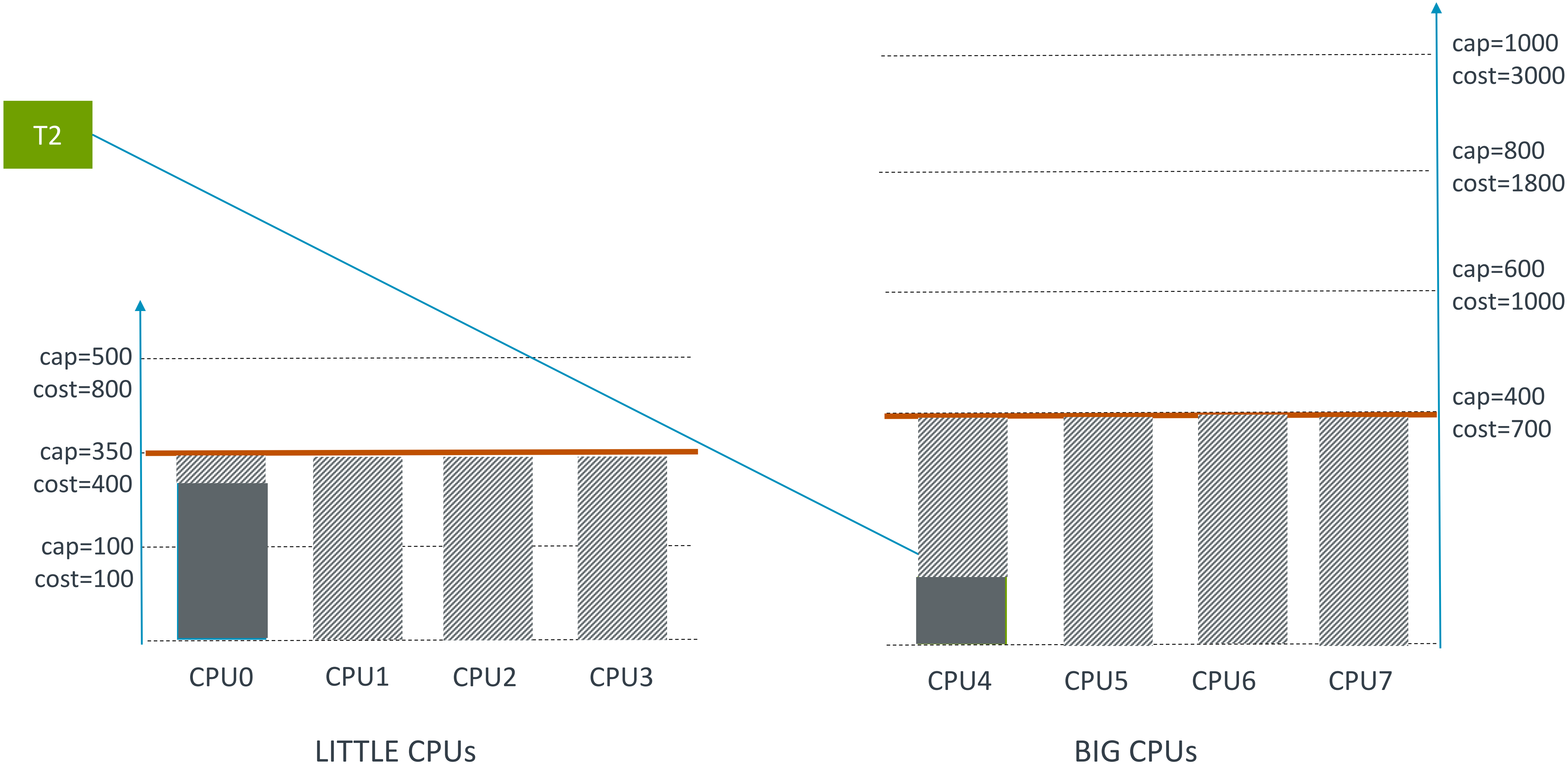
Algorithm complexity



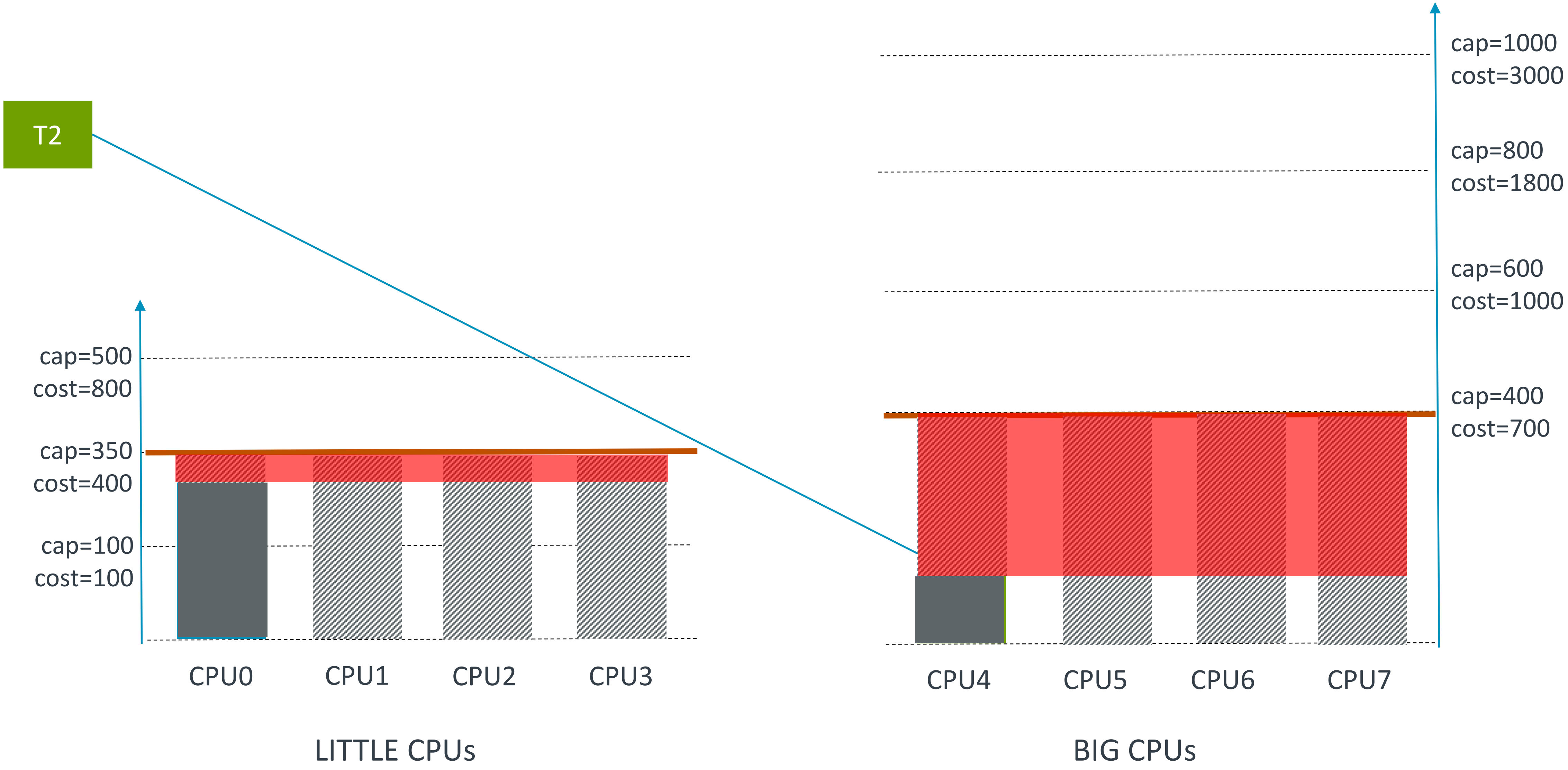
Algorithm complexity



Algorithm complexity



Algorithm complexity


T2
cap=1000
cost=3000
cap=800
cost=1800
cap=600
cost=1000
cap=400
cost=700
cap=500
cost=800
cap=350
cost=400
cap=100
cost=100
CPU0
CPU1
CPU2
CPU3
CPU4
CPU5
CPU6
CPU7
LITTLE CPUs
BIG CPUs