A Study on
C-group controlled big.LITTLE Architecture

Renesas Electronics Corporation
New Solutions Platform Business Division
Renesas Solutions Corporation
Advanced Software Platform Development Department

Tetsuya Nakagawa
Magnus Damm

2013/5/30 Rev. 1.00

Content

- Introduction
- big.LITTLE Architecture
- Approach 1: A cluster migration using C-group
- Approach 2: A Scalable virtual processor using C-group
- Evaluation
- Conclusion

Introduction

- Renesas has developed big.LITTLE architecture based SoCs and been working on the software solution
  - Existing Renesas SoCs: APE6(Shown in MWC2013), R-CarH2
  - Both of them have CA15 x4 + CA7 x 4, Oct cores

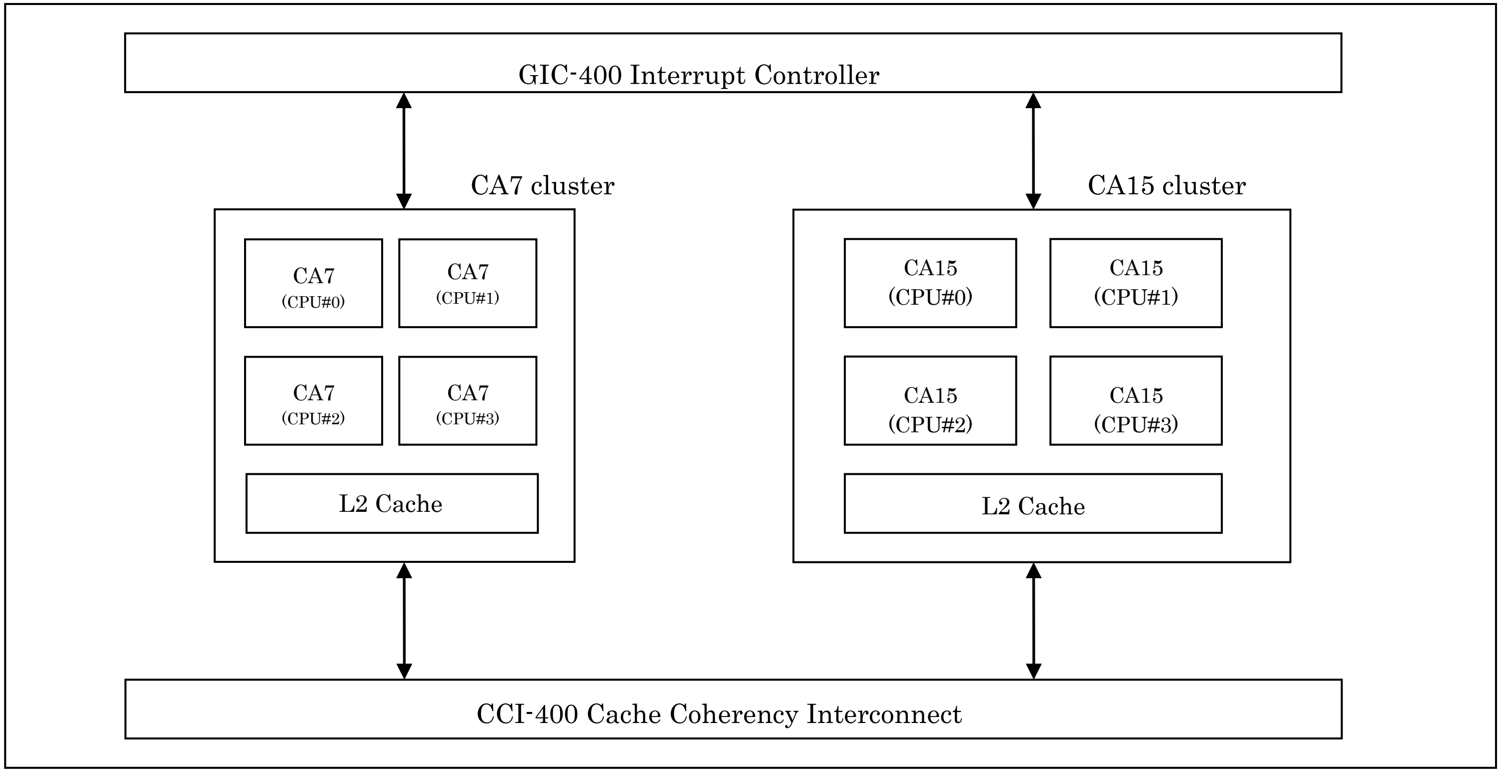
- big.LITTLE is ARM architecture and they proposed 3 use models. But there is no established software solution although ARM and many partners are making much effort
  - Kernel approaches for 2 of 3 ARM use models
  - Some proposal based on existing techniques

- Renesas as an ARM partner propose one powerful solution exploiting existing techniques and give its initial evaluation result on real silicon in this presentation

big.LITTLE Architecture and Solutions

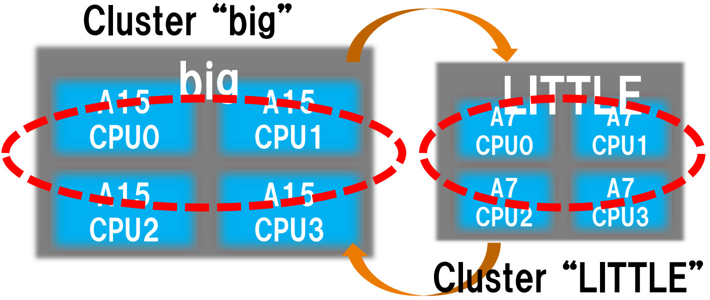
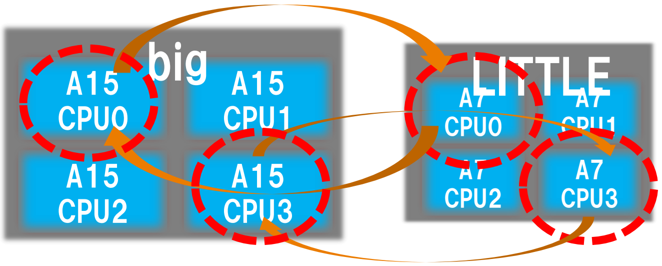
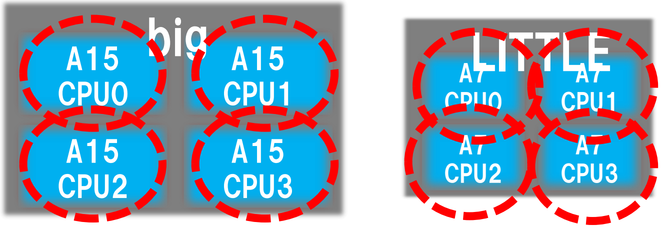
big.LITTLE Architecture and Solutions

- big.LITTLE Architecture
  - Heterogeneous Multi core architecture with performance oriented “big Core” and energy conscious “LITTLE Core” proposed by ARM.



big.LITTLE Architecture and Solutions

- ARM proposes 3 Use models

■ Cluster migration Either one of big (CA15×4) cluster or LITTLE (CA4×4) cluster is active	 Cluster "big" big A15 CPU0 A15 CPU1 A15 CPU2 A15 CPU3 LITTLE A7 CPU0 A7 CPU1 A7 CPU2 A7 CPU3 Cluster "LITTLE"	Pros: Easy to control Cons: Always only the half of all physical cores is active
■ In-Kernel Switcher Switching from big to LITTLE or LITTLE to big in CPU pair-wise (big×1 + LITTLE×1)	 big A15 CPU0 A15 CPU1 A15 CPU2 A15 CPU3 LITTLE A7 CPU0 A7 CPU1 A7 CPU2 A7 CPU3	Pros: Product quality Linux solution exists Cons: Always only the half of all physical cores is active
■ big.LITTLE MP Kernel takes care of heterogeneous multi processors	 big A15 CPU0 A15 CPU1 A15 CPU2 A15 CPU3 LITTLE A7 CPU0 A7 CPU1 A7 CPU2 A7 CPU3	Pros: All the existing physical cores are active if required Cons: Takes time to develop the kernel

Three challenging issues in big.LITTLE MP

- big.LITTLE MP is the most powerful use model which is expected to be the final solution

- Three challenging issues in big.LITTLE MP
  - Issue 1: Optimal process placement
    Dynamically place computationally intensive processes
    on big cores and less intensive ones on LITTLE cores
  - Issue 2: Exploitation of additional input parameters
    Kernel needs to take care of additional input parameters such as
    chip temperature and Performance Index (Performance oriented
    or power conscious) in addition to CPU load.
  - Issue 3: Consolidation with existing Power management framework
    Apply optimal Dynamic Voltage and Frequency Scaling on
    all the big cores and LITTLE cores.

Use Model comparison

- Solving all the 3 Issues at a time is a difficult “multi-dimensional optimization problem” particularly when all physical cores are active.

	Cluster Migration (Original)	In-Kernel Switcher	big.LITTLE MP
Issue 1	✓	✓	✓
Issue 2			✓
Issue 3	✓	✓	✓
All physical cores are active?			✓
Status	Not maintained	Used in a product	Work In Progress (Not in 3.10)

big.LITTLE Architecture and Solutions

- Here we propose two C-group based approaches which overcome all the three issues.

- Approach 1:
  - A Cluster migration using C-group
  - Enhanced comparing to the original Cluster migration use model by exploiting parameters such as Performance Index and Temperature

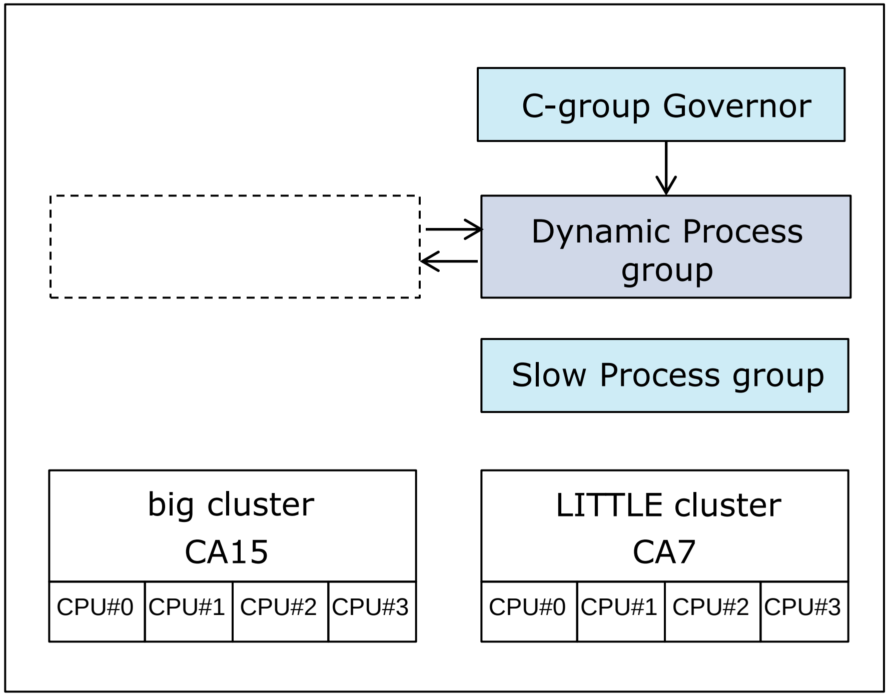
- Approach 2:
  - Based on Approach 1
  - Introduce “a scalable virtual processor” in place of “Cluster migration” to enable the use of all physical cores at the same time

Approach 1: A cluster migration using C-group

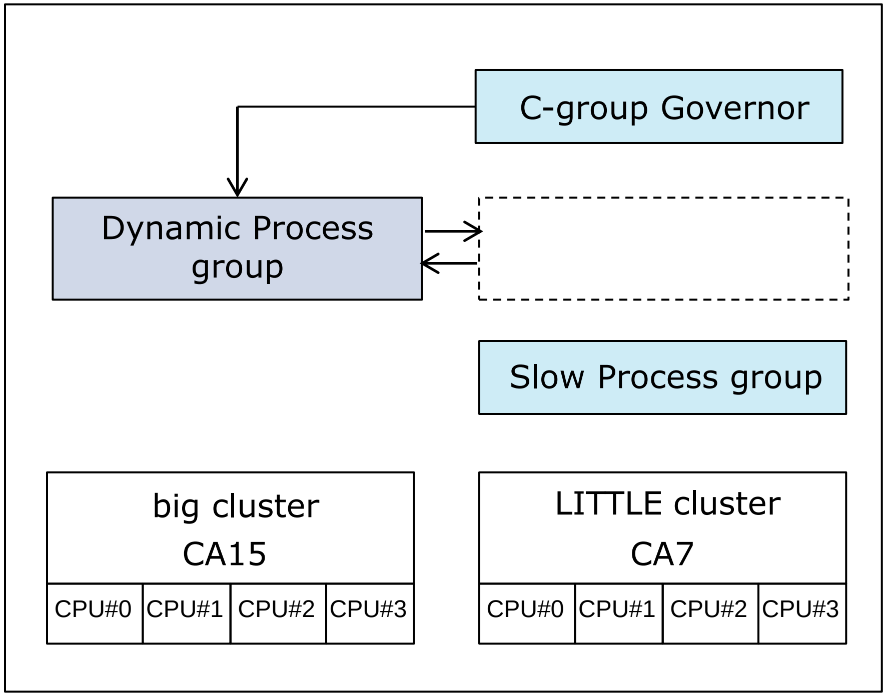
Approach 1: Optimal Process Placement (Issue1)

- C-group assigns "process groups" to pre-defined CPU sets
  - "Slow Process group" is statically assigned to LITTLE cluster
  - User space C-group governor migrates the other processes in "Dynamic Process group" between big and LITTLE clusters

Dynamic Process group on
LITTLE cluster



Dynamic Process group on
big cluster

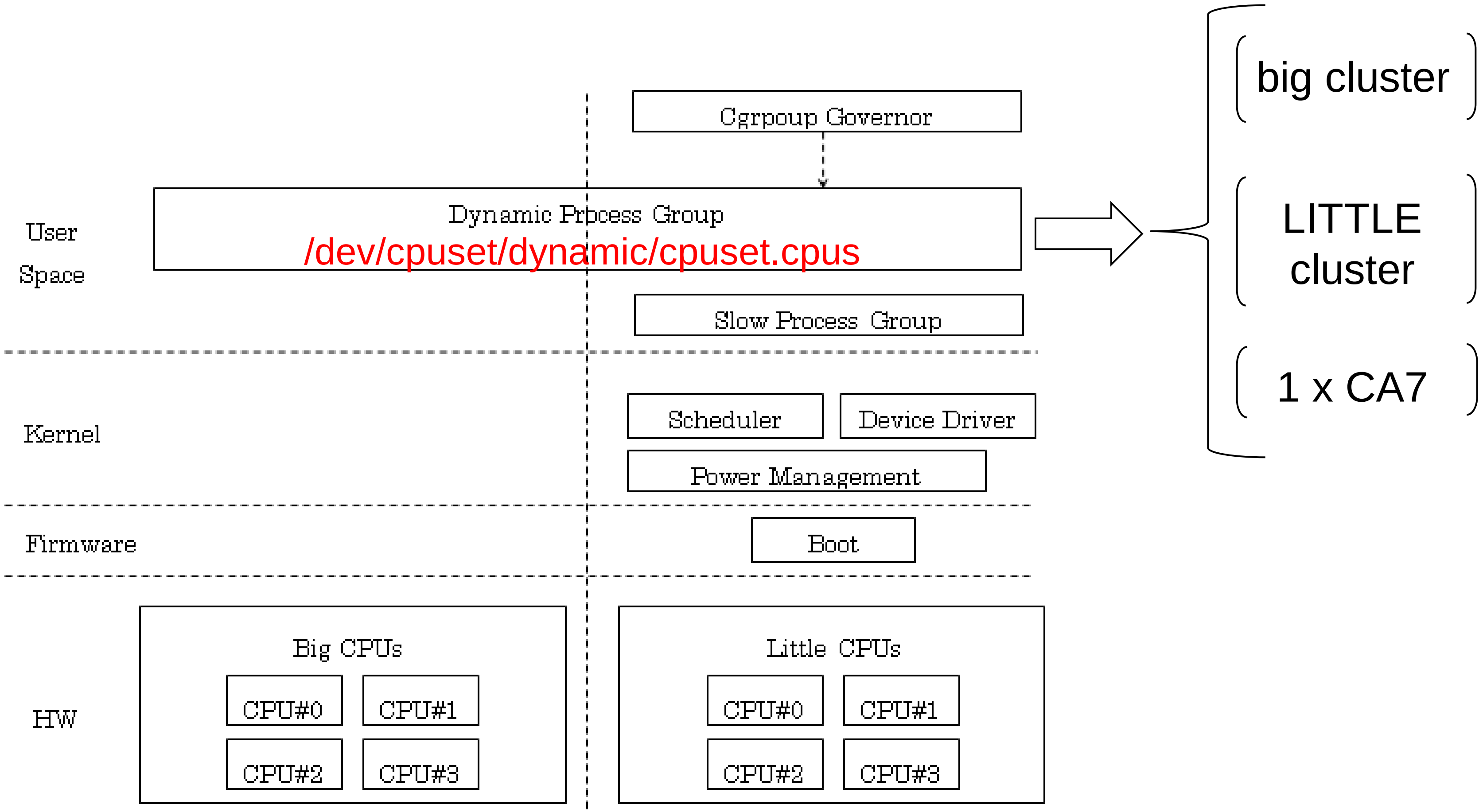


Approach 1: A cluster migration using C-group

- Standard kernel interfaces are used for monitoring and control
  - /proc/cpuinfo is used to detect CA15 and CA7
  - /proc/stat is used to determine per-CPU usage
  - /sys/class/thermal/.. provides temperature information
  - /sys/devices/system/cpu/... can be used with CPU Hotplug
- A “dynamic” C-group cpuset switches between CA15 and CA7
  - /dev/cpuset/dynamic/cpuset.cpus is defined to switch cluster
  - The number of CPU cores is scaled depending on Performance Index
  - Any number of CA15 or any number of CA7 can be used
- C-group governor monitors Thermal sensor state and reduces CA15 usage
  - /dev/cpuset/dynamic/cpuset.cpus is also used stay on CA7

Approach 1: A cluster migration using C-group

- Dynamically assign cpuset.cpus on “Dynamic Process Group” to a cluster



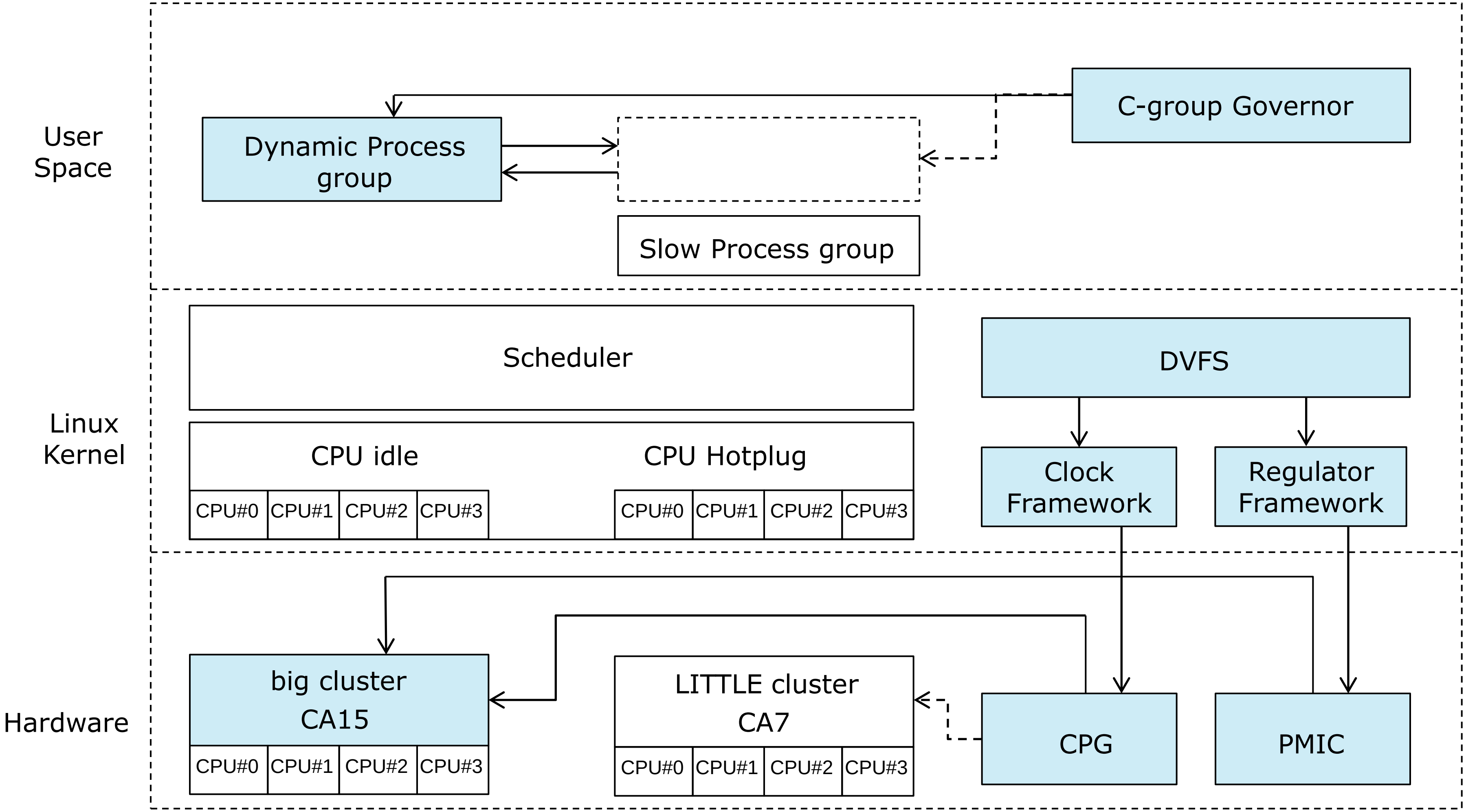
Approach 1: C-group governor algorithm (Issue2)

- C-group governor exploits “Temperature and Performance Index”
- “Temperature and Performance Index” determine “Dynamic process placement” and the number of core

Temperature	Performance Index	Dynamic Process	Slow Process
less than 60 deg C	0% - 20%	CA7 x 1	CA7 x 1
	20% - 30%	CA7 x 2	CA7 x 2
	30% - 40%	CA7 x 3	CA7 x 3
	40% - 50%	CA7 x 4	CA7 x 4
	50% - 60%	CA15 x 1	CA7 x 4
	60% - 70%	CA15 x 2	CA7 x 4
	70% - 80%	CA15 x 3	CA7 x 4
	80% - 100%	CA15 x 4	CA7 x 4
Lager than or equal to 60 deg C	0% - 100%	CA7 x 1	CA7 x 1

Approach 1: Per cluster CPUFreq Scaling(Issue3)

- In current SoCs, CPUs in one cluster share same clock and voltage control and DVFS can be applied in each cluster independently.



Approach 1 Summary

- Approach 1 solves all the three issues,
  - Issue1: Optimal process placement is taken care of by cluster switch of "Dynamic Process Group"
  - Issue2: Additional input parameters such as temperature and Performance Index are exploited by C-Group Governor.
  - Issue3 is solved per cluster base.

	Cluster Migration (Original)	In-Kernel Swither	big.LITTLE MP	Approach 1
Issue 1	✓	✓	✓	✓
Issue 2			✓	✓
Issue 3	✓	✓	✓	✓
All physical cores are active?			✓	Partially Yes
Status	Not maintained	Used in a product	Work In Progress (Not in 3.10)	Available with the current kernel

- But for "Dynamic Process Group", all the physical cores, 8 in this case, can not be assigned.

Approach 2:
A scalable virtual processor using C-group

Approach 2: A scalable virtual processor (Issue1)

- Introduce a scalable virtual processor and map the multi dimensional optimization problem onto one dimensional problem.
  1. Heterogeneous multi core -> a scalable virtual processor
  2. Consolidation with one dimensional CPUfreq scaling

- One example of scalable virtual processor (Vi: i=1-12)

$$\begin{pmatrix} V_1 \\ V_2 \\ V_3 \\ V_4 \\ V_5 \\ V_6 \\ V_7 \\ V_8 \\ V_9 \\ V_{10} \\ V_{11} \\ V_{12} \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} \text{x} \begin{pmatrix} B_1 \\ B_2 \\ B_3 \\ B_4 \\ L_1 \\ L_2 \\ L_3 \\ L_4 \end{pmatrix} = \begin{pmatrix} L_1 \\ L_1 + L_2 \\ L_1 + L_2 + L_3 \\ L_1 + L_2 + L_3 + L_4 \\ B_1 + L_1 + L_2 + L_3 \\ B_1 + L_1 + L_2 + L_3 + L_4 \\ B_1 + B_2 + L_1 + L_2 + L_3 \\ B_1 + B_2 + L_1 + L_2 + L_3 + L_4 \\ B_1 + B_2 + B_3 + L_1 + L_2 + L_3 \\ B_1 + B_2 + B_3 + L_1 + L_2 + L_3 + L_4 \\ B_1 + B_2 + B_3 + B_4 + L_1 + L_2 + L_3 \\ B_1 + B_2 + B_3 + B_4 + L_1 + L_2 + L_3 + L_4 \end{pmatrix}$$

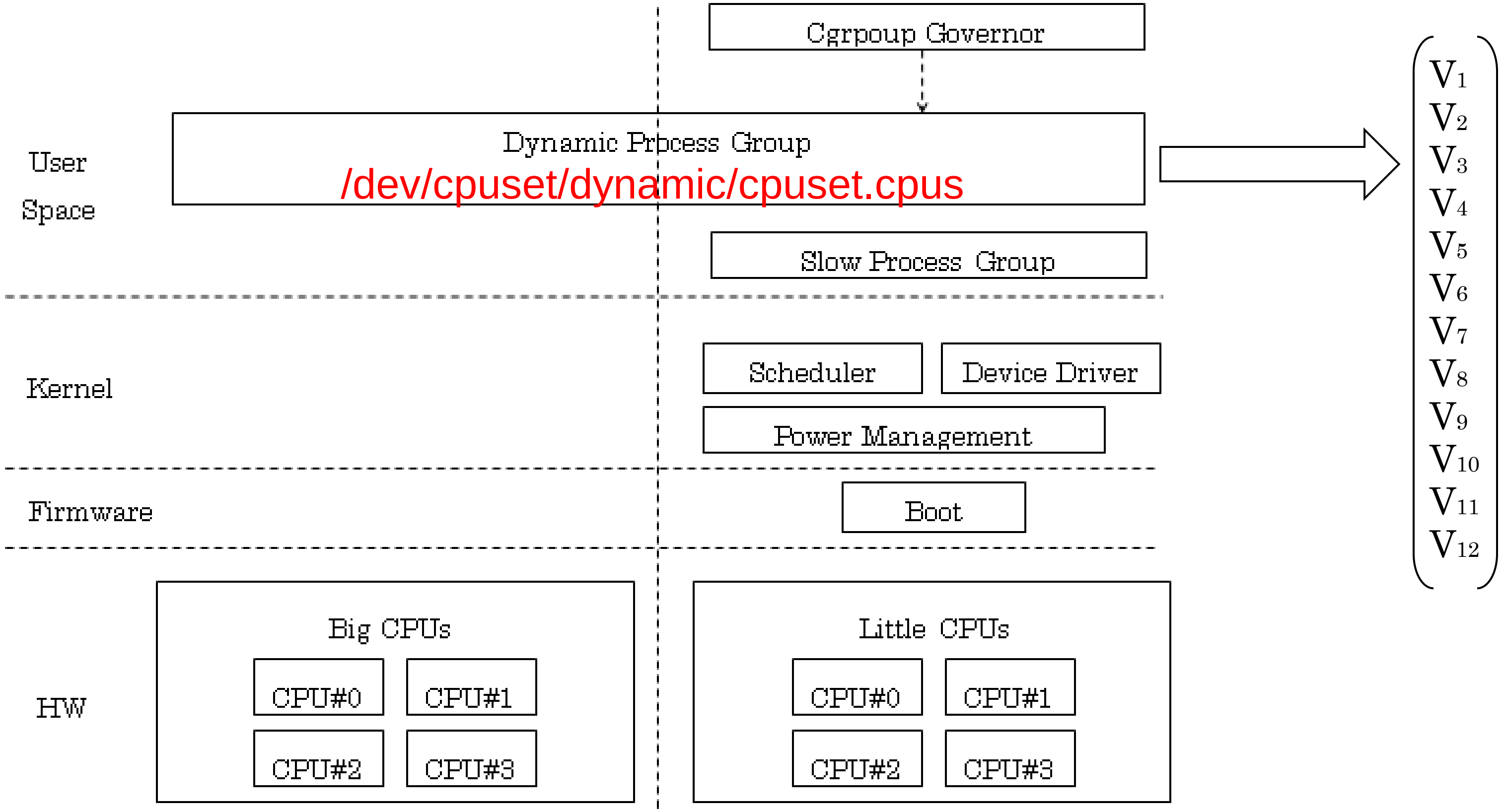
Approach 2: A scalable virtual processor (Issue1)

- Another example of scalable virtual processor (Vi: i=1-8)

$$
\begin{pmatrix} V_1 \\ V_2 \\ V_3 \\ V_4 \\ V_5 \\ V_6 \\ V_7 \\ V_8 \end{pmatrix}
=
\begin{pmatrix}
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\
0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\
1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\
1 & 1 & 0 & 0 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1
\end{pmatrix}
\text{x}
\begin{pmatrix} B_1 \\ B_2 \\ B_3 \\ B_4 \\ L_1 \\ L_2 \\ L_3 \\ L_4 \end{pmatrix}
=
\begin{pmatrix}
L_1 \\
L_1 + L_2 \\
L_1 + L_2 + L_3 \\
L_1 + L_2 + L_3 + L_4 \\
B_1 + L_1 + L_2 + L_3 + L_4 \\
B_1 + B_2 + L_1 + L_2 + L_3 + L_4 \\
B_1 + B_2 + B_3 + L_1 + L_2 + L_3 + L_4 \\
B_1 + B_2 + B_3 + B_4 + L_1 + L_2 + L_3 + L_4
\end{pmatrix}
$$

Scalable virtual processor using C-group

- Dynamically assign cpuset.cpus on “Dynamic Process Group” to an adequate scalable virtual processor state Vi according to its load (hereafter we call “system load”)



Approach 2: C-group governor algorithm (Issue2)

- CPU number scaling is done by selecting a scalable virtual processor state.
- Dynamic process placement is done based on all of temperature, Performance Index and System load.

Temperature	Performance Index	System Load	Scaling Operation
>=60 deg C	-	-	Choose V_1
<60 deg C	>=50%	>=70%	V_i -> V_{i+1}
		30% =< or <70%	NOP
		<30%	V_i -> V_{i-1}
	<50%	30% >=	NOP
		<30%	V_i -> V_{i-1}

CPUfreq consolidation in one dimension (Issue3)

- CPUfreq consolidation is realized using C-group governor also as “CPUfreq User Space Governor”.
- CPUfreq scaling is applied to “Virtual Frequency” of “Scalable Virtual Processor” where its state does not change.
- Standard kernel interfaces can be used for CPUfreq scaling.

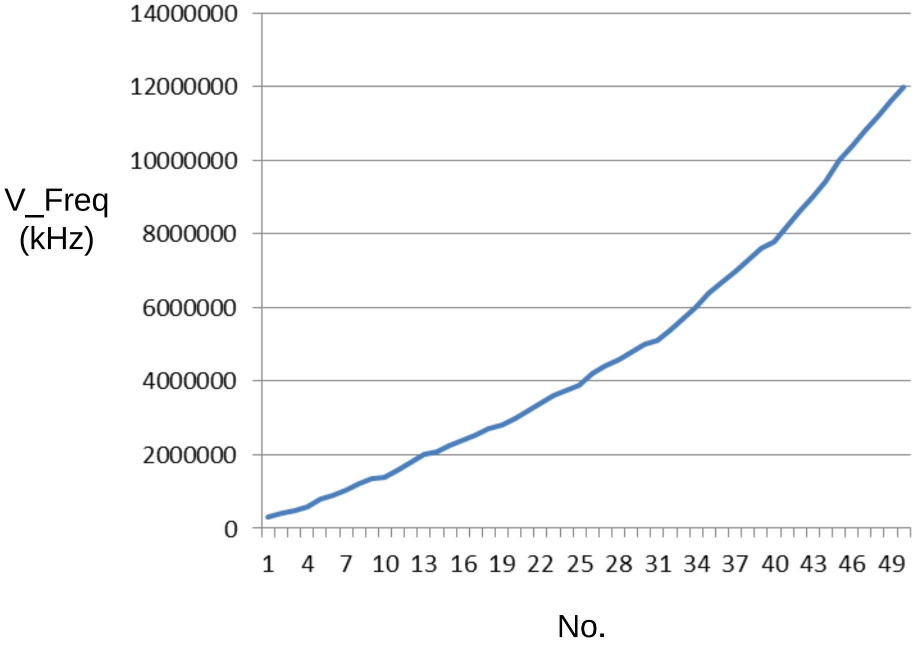
Temperature	Performance Index	System Load	Scaling Operation
>=60 deg C	-	-	Choose V_1
<60 deg C	>=50%	>70%	V_i -> V_{i+1}
		30% =< or < 70%	**CPUfreq scaling on Vi**
		<30%	V_i -> V_{i-1}
	<50%	30% >=	**CPUfreq scaling on Vi**
		<30%	V_i -> V_{i-1}

Virtual Frequency (Issue3)

- Scalable Virtual Processor OPP is mapped to 8 Physical CPU OPPs

		V_Freq	P_OPP	P_OPP	P_OPP	P_OPP	P_OPP	P_OPP	P_OPP	P_OPP	VL	VB
No			(L1)	(L2)	(L3)	(L4)	(B1)	(B2)	(B3)	(B4)		
1	V1	300000	300000								V1	
2		400000	400000								V1	
3		500000	500000								V1	
4	V2	600000	300000	300000							V1	
5		800000	400000	400000							V1	
6		900000	450000	450000							V1	
7	V3	1050000	350000	350000	350000						V1	
8		1200000	400000	400000	400000						V1	
9		1350000	450000	450000	450000						V1	
10	V4	1400000	350000	350000	350000	350000					V1	
11		1600000	400000	400000	400000	400000					V1	
12		1800000	450000	450000	450000	450000					V1	
13		2000000	500000	500000	500000	500000					V1	
14	V5	2100000	500000	500000	500000		600000				V1	V1
15		2250000	550000	550000	550000		600000				V1	V1
16		2400000	600000	600000	600000		600000				V1	V1
17		2550000	650000	650000	650000		600000				V1	V1
18		2700000	700000	700000	700000		600000				V1	V1
19	V6	2800000	550000	550000	550000	550000	600000				V1	V1
20		3000000	600000	600000	600000	600000	600000				V1	V1
21		3200000	650000	650000	650000	650000	600000				V1	V1
22		3400000	700000	700000	700000	700000	600000				V1	V1
23	V7	3600000	800000	800000	800000		600000	600000			V1	V1
24		3750000	850000	850000	850000		600000	600000			V1	V1
25		3900000	900000	900000	900000		600000	600000			V1	V1
26		4200000	1000000	1000000	1000000		600000	600000			V2	V1
27	V8	4400000	800000	800000	800000	800000	600000	600000			V1	V1
28		4600000	850000	850000	850000	850000	600000	600000			V1	V1
29		4800000	900000	900000	900000	900000	600000	600000			V1	V1
30		5000000	950000	950000	950000	950000	600000	600000			V1	V1
31	V9	5100000	1000000	1000000	1000000		700000	700000	700000		V2	V1
32		5400000	1000000	1000000	1000000		800000	800000	800000		V2	V1
33		5700000	1000000	1000000	1000000		900000	900000	900000		V2	V1
34		6000000	1000000	1000000	1000000		1000000	1000000	1000000		V2	V1
35	V10	6400000	1000000	1000000	1000000	1000000	800000	800000	800000		V2	V1
36		6700000	1000000	1000000	1000000	1000000	900000	900000	900000		V2	V1
37		7000000	1000000	1000000	1000000	1000000	1000000	1000000	1000000		V2	V1
38		7300000	1000000	1000000	1000000	1000000	1100000	1100000	1100000		V2	V2
39		7600000	1000000	1000000	1000000	1000000	1200000	1200000	1200000		V2	V2
40	V11	7800000	1000000	1000000	1000000		1200000	1200000	1200000	1200000	V2	V2
41		8200000	1000000	1000000	1000000		1300000	1300000	1300000	1300000	V2	V2
42		8600000	1000000	1000000	1000000		1400000	1400000	1400000	1400000	V2	V2
43		9000000	1000000	1000000	1000000		1500000	1500000	1500000	1500000	V2	V2
44		9400000	1000000	1000000	1000000		1600000	1600000	1600000	1600000	V2	V2
45	V12	10000000	1000000	1000000	1000000	1000000	1500000	1500000	1500000	1500000	V2	V2
46		10400000	1000000	1000000	1000000	1000000	1600000	1600000	1600000	1600000	V2	V2
47		10800000	1000000	1000000	1000000	1000000	1700000	1700000	1700000	1700000	V2	V2
48		11200000	1000000	1000000	1000000	1000000	1800000	1800000	1800000	1800000	V2	V2
49		11600000	1000000	1000000	1000000	1000000	1900000	1900000	1900000	1900000	V2	V2
50		12000000	1000000	1000000	1000000	1000000	2000000	2000000	2000000	2000000	V2	V2

- V_Freq = $\sum_{i=1}^{4}(Freq(Bi) + Freq(Li))$
  - Min(V_Freq) = Min(Freq(L1))
= 300MHz
  - Max(V_Freq) = Max $(\sum_{i=1}^{4}(Freq(Bi) + Freq(Li)))$
= 2GHz x 4 + 1GHz x 4 = 12GHz



Super wide performance dynamic range
300MHz to 12GHz with DVFS (Theoretical Value)
866MHz to 7.4GHz without Frequency scaling

Approach 2 Summary

- Approach 2 solves all the three issues, while All physical cores are active.
  - Issue1: Optimum process placement is taken care of by changing the state of "Virtual Scalable Processor".
  - Issue2: Additional input parameters such as temperature and Performance Index are exploited by C-Group Governor.
  - Issue3: CPUfreq Governor is consolidated with C-Group Governor and Established one dimensional Scaling scheme can be applied.

	Cluster Migration (Original)	In-Kernel Swither	big.LITTLE MP	Approach 1	Approach 2
Issue 1	✓	✓	✓	✓	✓
Issue 2			✓	✓	✓
Issue 3	✓	✓	✓	✓	✓
All physical cores are active?			✓	Partially Yes	✓
Status	Not maintained	Used in a product	Work In Progress (Not in 3.10)	Available with the current kernel	Available with the current kernel

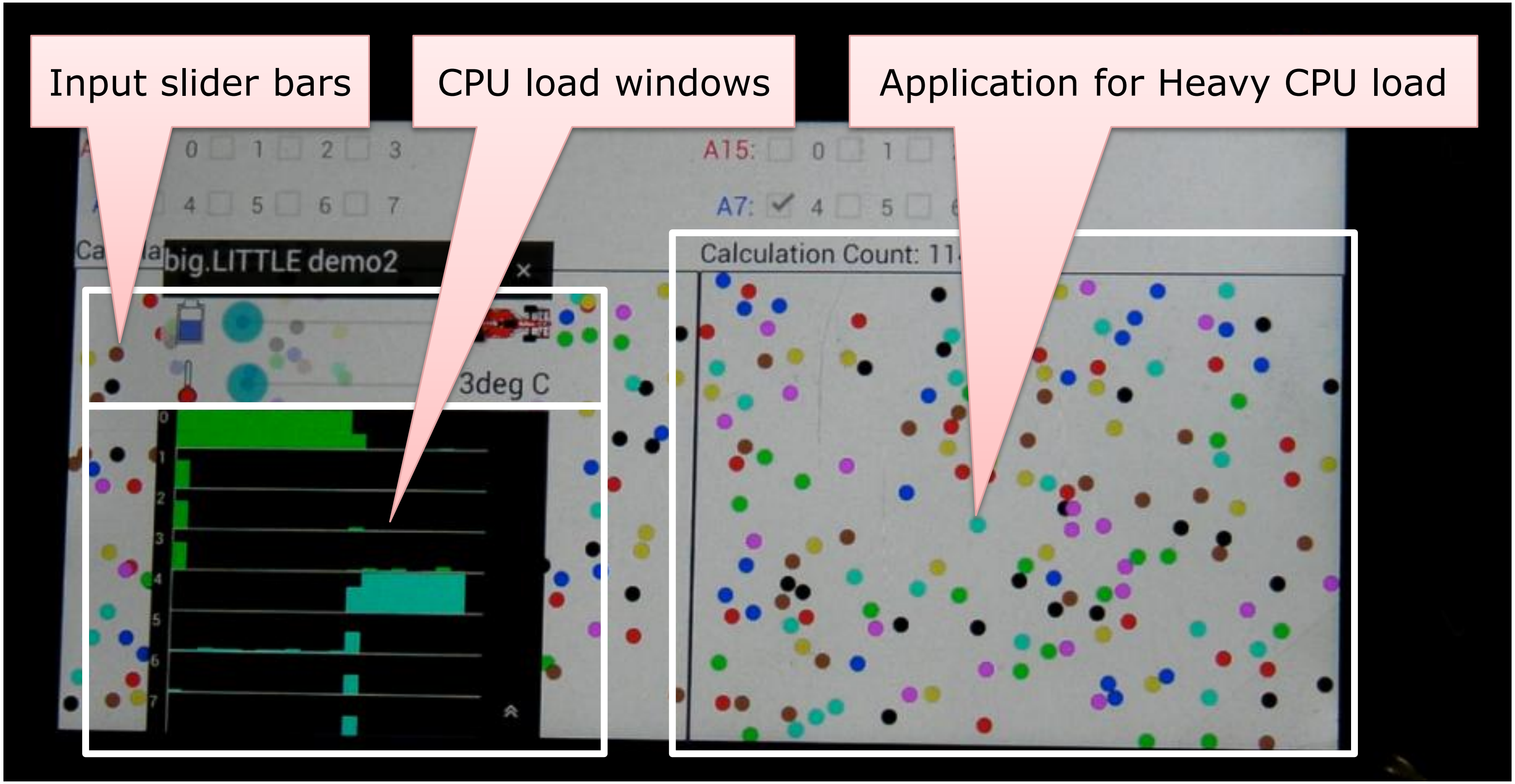
Evaluation

Evaluation on Renesas APE6 test board

- For evaluation a demo is integrated on AOSP Android 4.1.2+Linaro 12.10 kernel 3.6 on APE6 test board.

Evaluation Demo on APE6 test board

- Evaluation Demo consists of 3 components



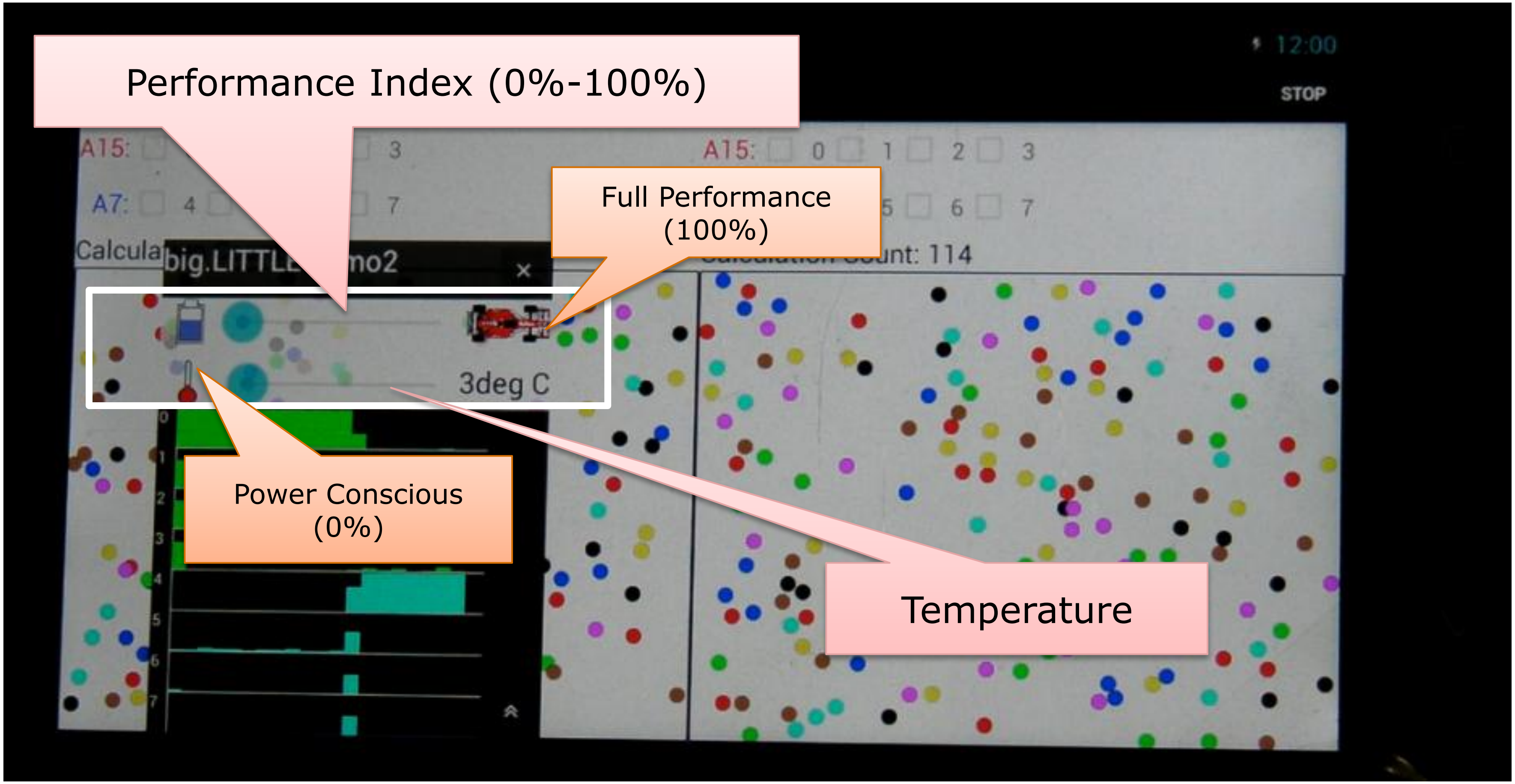
Input slider bars

CPU load windows

Application for Heavy CPU load

Evaluation Demo on APE6 test board

- 2 input slider bars
  - Performance Index and Temperature



Performance Index (0%-100%)

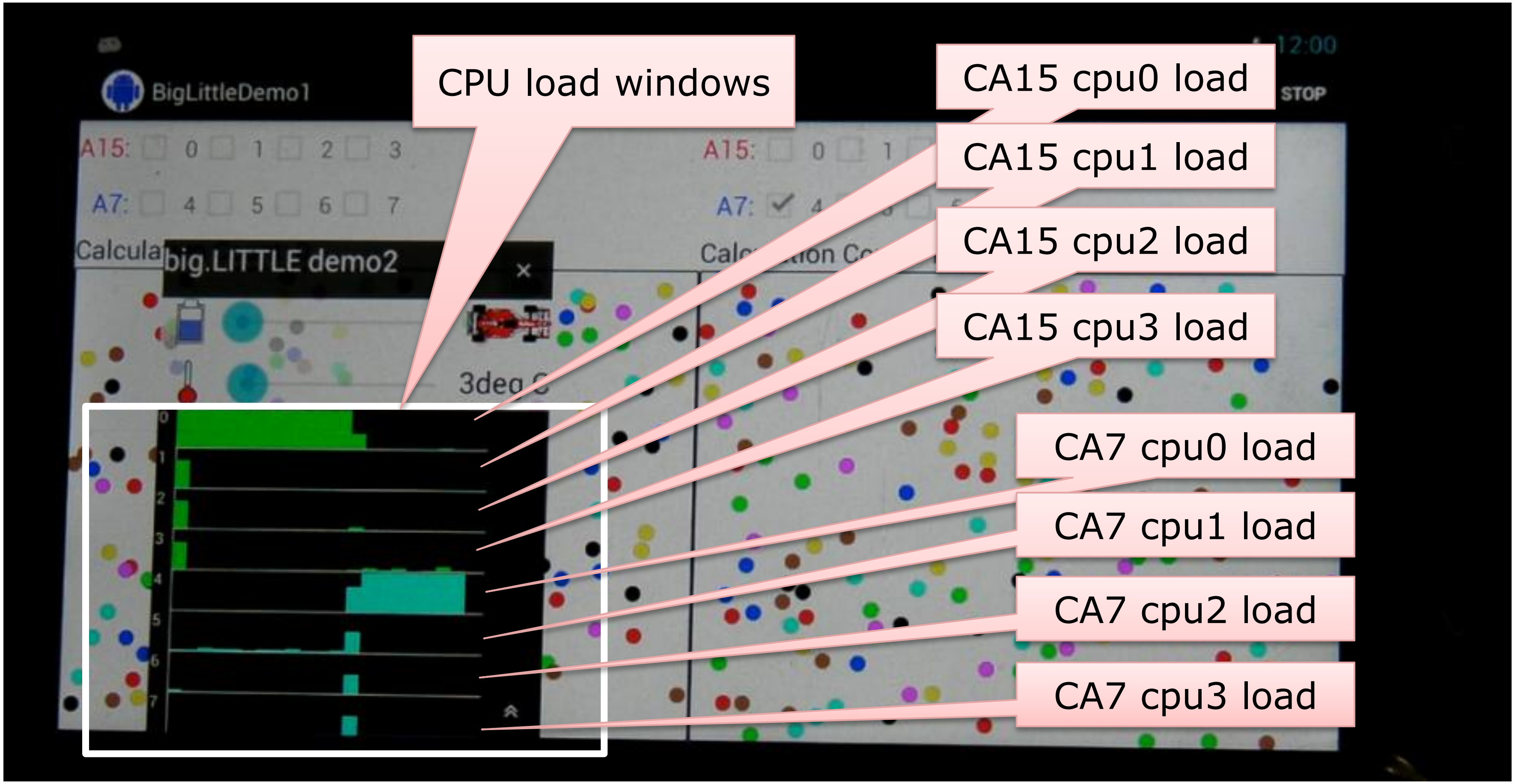
Full Performance (100%)

Power Conscious (0%)

Temperature

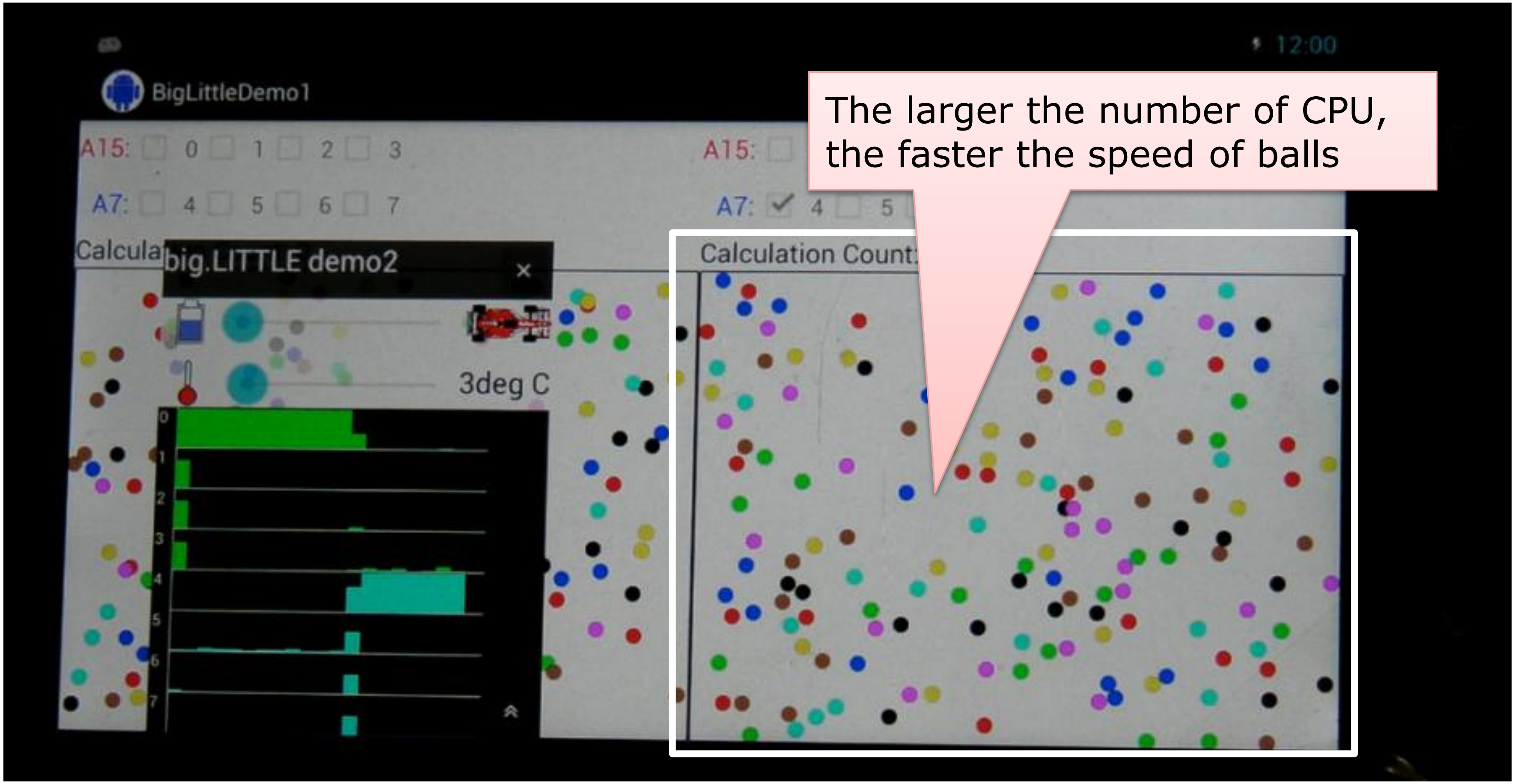
Evaluation Demo on APE6 test board

- CPU load window is prepared for each core
- 8 Sub windows for 8 cores



Evaluation Demo on APE6 test board

- 2D colliding n-body simulation for Heavy CPU load
  - Runs only on the cores which 2 slider bar inputs enable



The larger the number of CPU, the faster the speed of balls

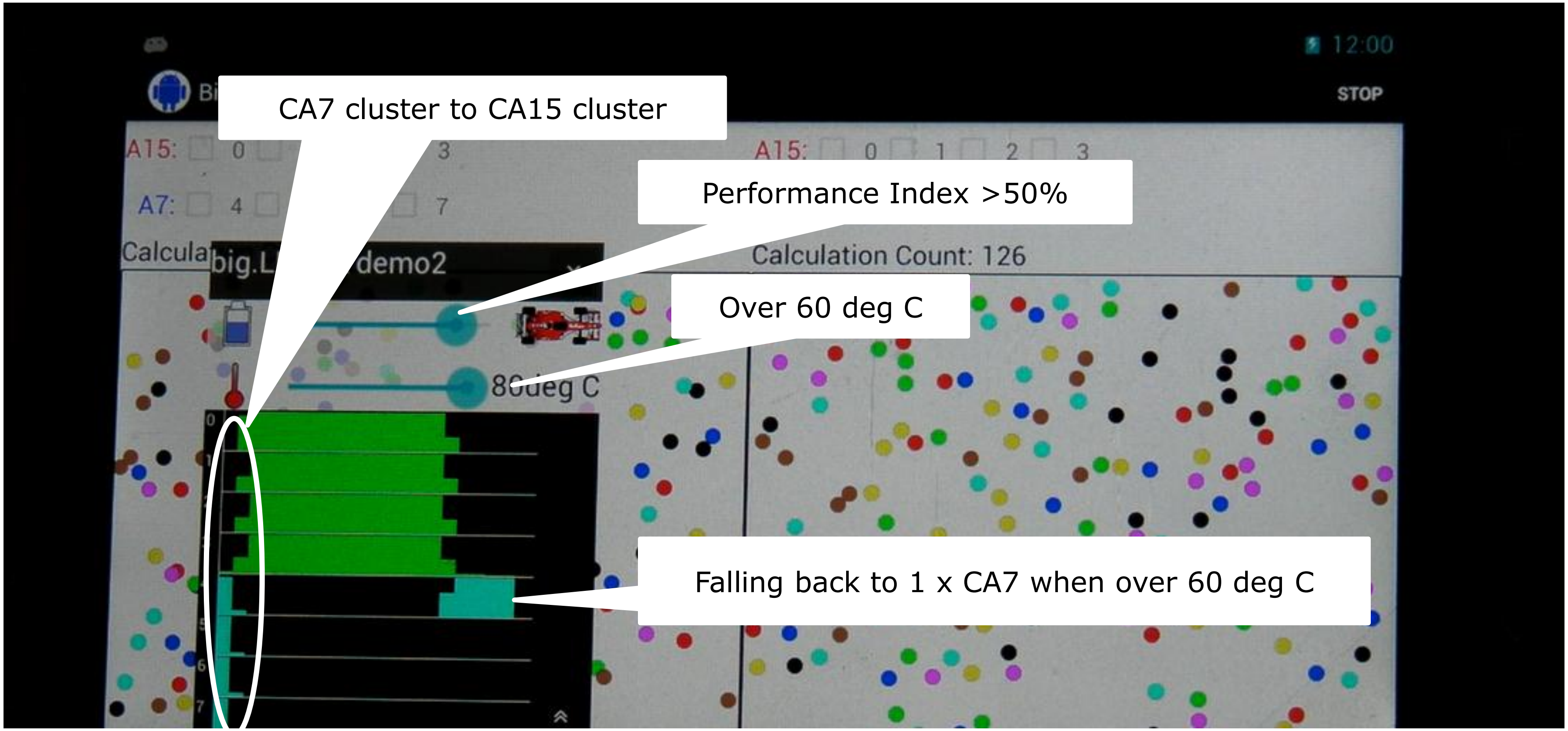
Approach 1 Evaluation

- We evaluate, on demo with Approach 1 C-group governor,
  - Cluster is switched at Performance Index boundary(50%)
  - Number of core is determined by Performance Index
  - Falling back to 1 x CA7 over 60 deg C

Temperature	Performance Index	Dynamic Process
less than 60 deg C	0% - 20%	CA7 x 1
	20% - 30%	CA7 x 2
	30% - 40%	CA7 x 3
	40% - 50%	CA7 x 4
	50% - 60%	CA15 x 1
	60% - 70%	CA15 x 2
	70% - 80%	CA15 x 3
	80% - 100%	CA15 x 4
lager than or equal to 60 deg C	0% - 100%	CA7 x 1

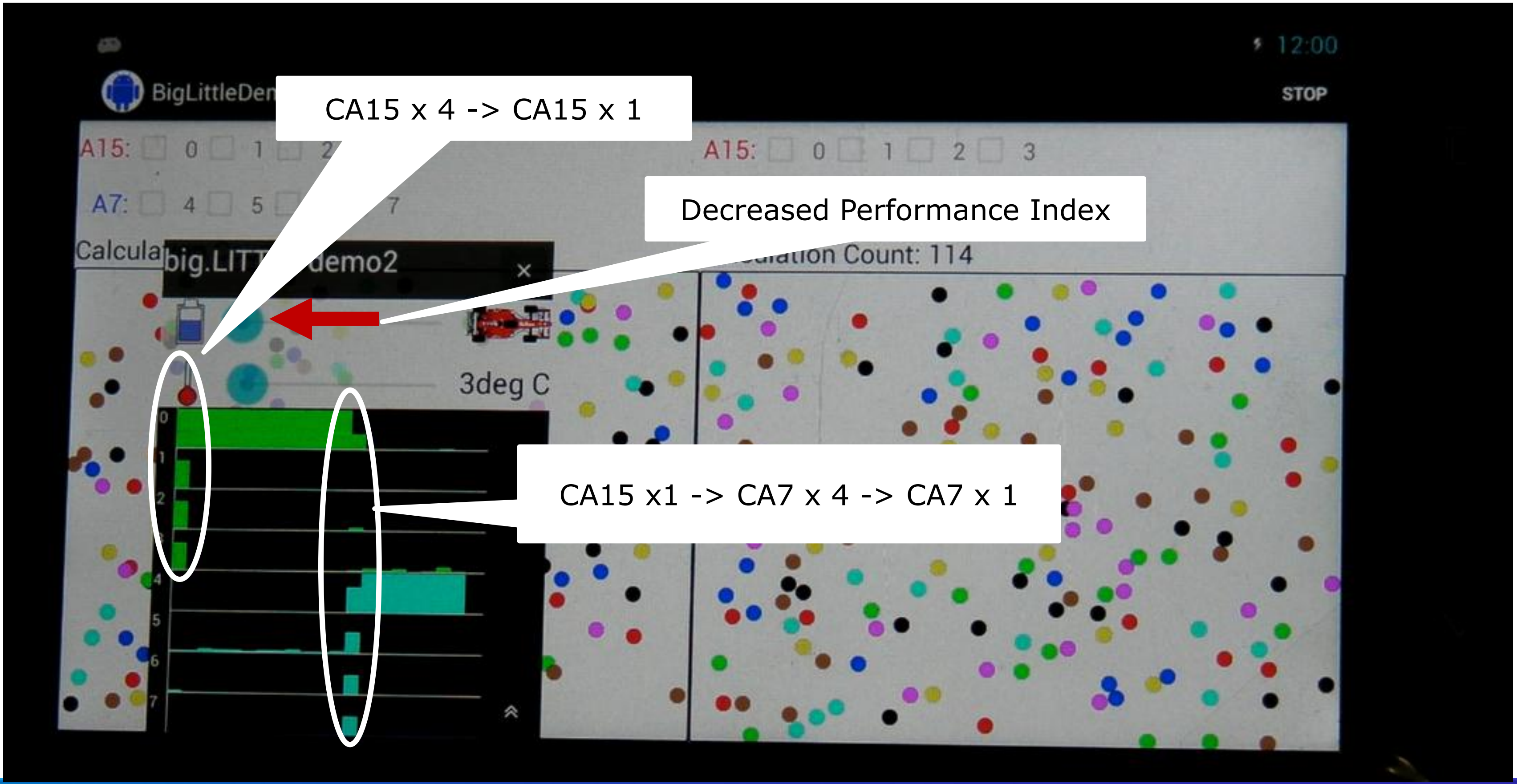
Approach 1 Evaluation Result

- We confirmed, on demo with Approach 1 C-group governor,
  - Cluster is switched at Performance Index boundary(50%)
  - Number of core is determined by Performance Index
  - Falling back to 1 x CA7 when over 60 deg C



Approach 1 Evaluation Result

- We confirmed, on demo with Approach 1 C-group governor,
  - Cluster is switched at Performance Index boundary(50%)
  - Number of core is determined by Performance Index
  - Falling back to 1 x CA7 when over 60 deg C



CA15 x 4 -> CA15 x 1

Decreased Performance Index

CA15 x1 -> CA7 x 4 -> CA7 x 1

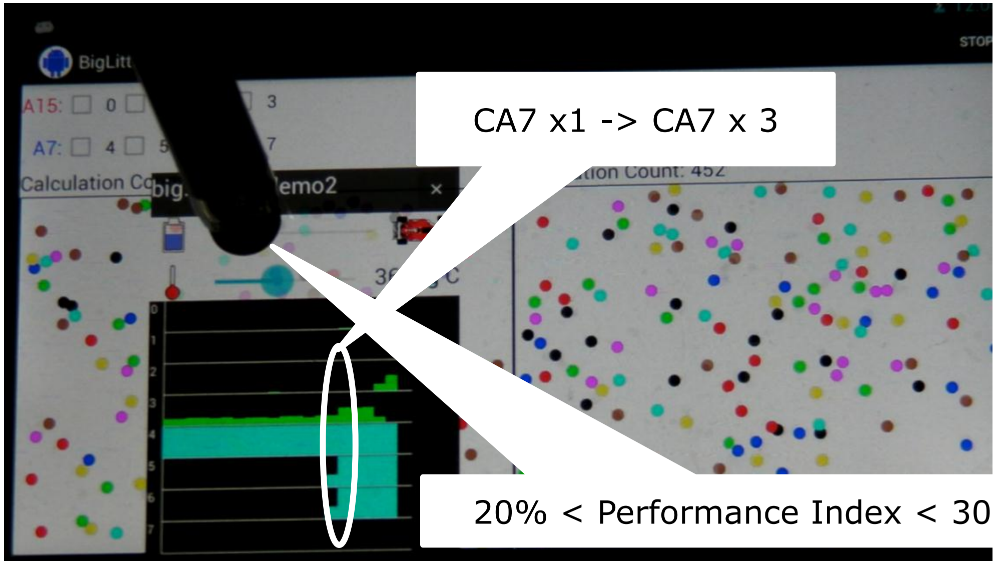
Approach 2 Evaluation

- We evaluate, on demo with Approach 2 C-group governor,
  - Scalable virtual processor is controlled in one dimension by Performance Index.
  - The scalable virtual processor (Vi) evaluated in demo.
    (This is the second example Vi: i=1-8)

Performance Index	Dynamic Process	Virtual Processor
0% - 10%	CA7 x 1	V1
10% - 20%	CA7 x 2	V2
20% - 30%	CA7 x 3	V3
30% - 40%	CA7 x 4	V4
40% - 55%	CA15 x 1 + CA7 x 4	V5
55% - 70%	CA15 x 2 + CA7 x 4	V6
70% - 85%	CA15 x 3 + CA7 x 4	V7
85% - 100%	CA15 x 4 + CA7 x 4	V8

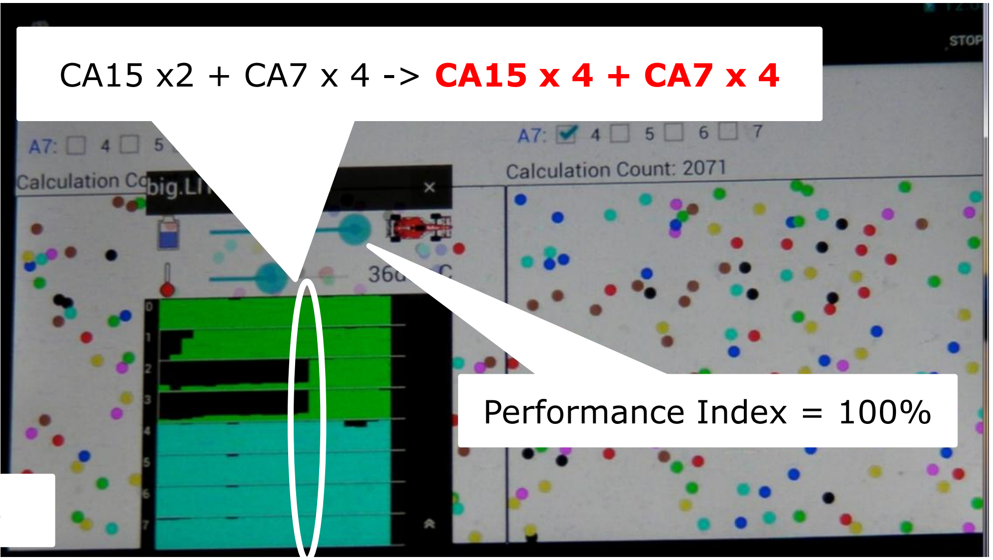
Approach 2 Evaluation Result

- We confirmed, on demo with Approach 2 C-group governor,
  - V1(CA7 x 1) to V8(CA15 x 4 + CA7 x 4) scaling is controlled in one dimension by Performance Index slider bar.
  - At the highest end, this approach enables the use of all physical cores (CA15 x 4 + CA7 x 4) at the same time



CA7 x1 -> CA7 x 3

20% < Performance Index < 30%



CA15 x2 + CA7 x 4 -> **CA15 x 4 + CA7 x 4**

Performance Index = 100%

Not evaluated yet

- CPUfreq consolidation
- Overhead measurement
- CPU hotplug and CPUidle integration

Conclusion

Conclusion

- Two C-group based big.LITTLE solutions are proposed.
- Both Approaches solves all three challenging issues in big.LITTLE MP and can go with the current latest upstream kernel (3.9)
  - **Issue 1: Optimal process placement**

    Optimal process placement is taken care of by "Dynamic Process Group‟ assigned on “CPU cluster” or “Virtual Scalable Processor”
  - **Issue 2: Exploitation of additional input parameters**

    Additional input parameters such as chip temperature and Performance Index are exploited in C-Group governor.
  - **Issue 3: Consolidation with existing Power management framework**

    Established one dimensional Dynamic Voltage and Frequency Scaling scheme can be applied as is.

Conclusion

- In addition, Approach 2
  - enables the use of all physical cores at the same time
  - Provides super wide performance dynamic range
    - 300MHz to 12GHz with DVFS (Theoretical Value)
    - 866MHz to 7.4GHz without Frequency scaling (This demo)

Next Step

- Further evaluation on Approach 2
  - CPUfreq consolidation
  - C-group governor performance overhead measurement
  - CPU hotplug and CPUidle integration

- Study on complementary solution with big.LITTLE MP kernel

- Investigation on other C-group based solutions

Thanks!

Trademarks

All trademarks and registered trademarks are the property of their respective owners.

big.LITTLE and its based trademarks are trademarks of ARM Holdings.
Android and its based trademarks are trademarks of Google Inc.
Linaro® is a registered trademark of Linaro in the U.S. and other countries
Linux® is the registered trademark of Linus Torvalds in the U.S. and other countries

Renesas Electronics Corporation