ARQUITETURA BIG.LITTLE

Seminário 1

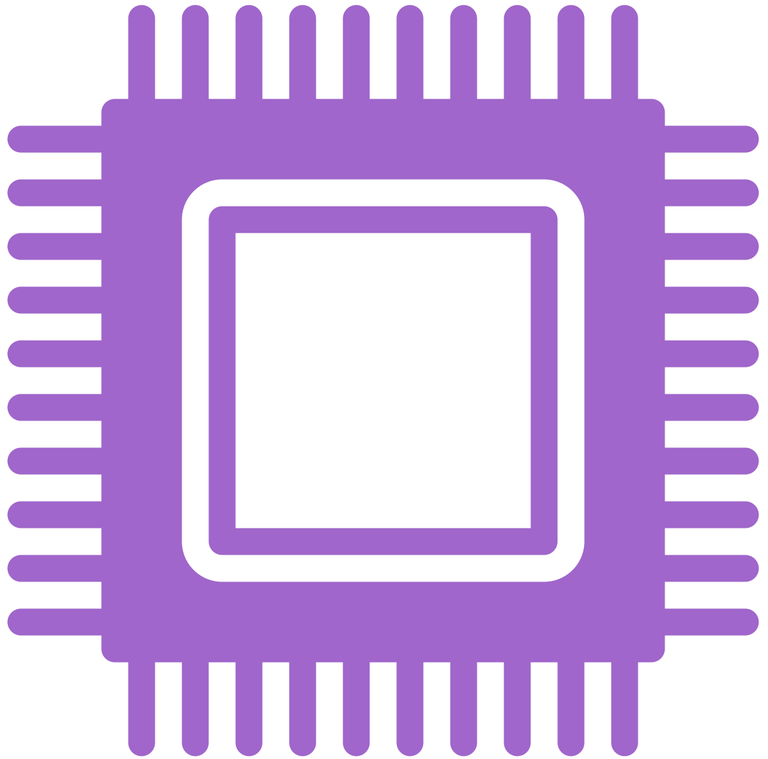
MC504 - Sistemas Operacionais

César Devens Grazioti - RA: 195641
João Miguel De Oliveira Guimarães - RA: 174358
Otavio Anovazzi - RA: 186331
Renan Luis Moraes De Sousa - RA: 243792

Universidade Estadual de Campinas

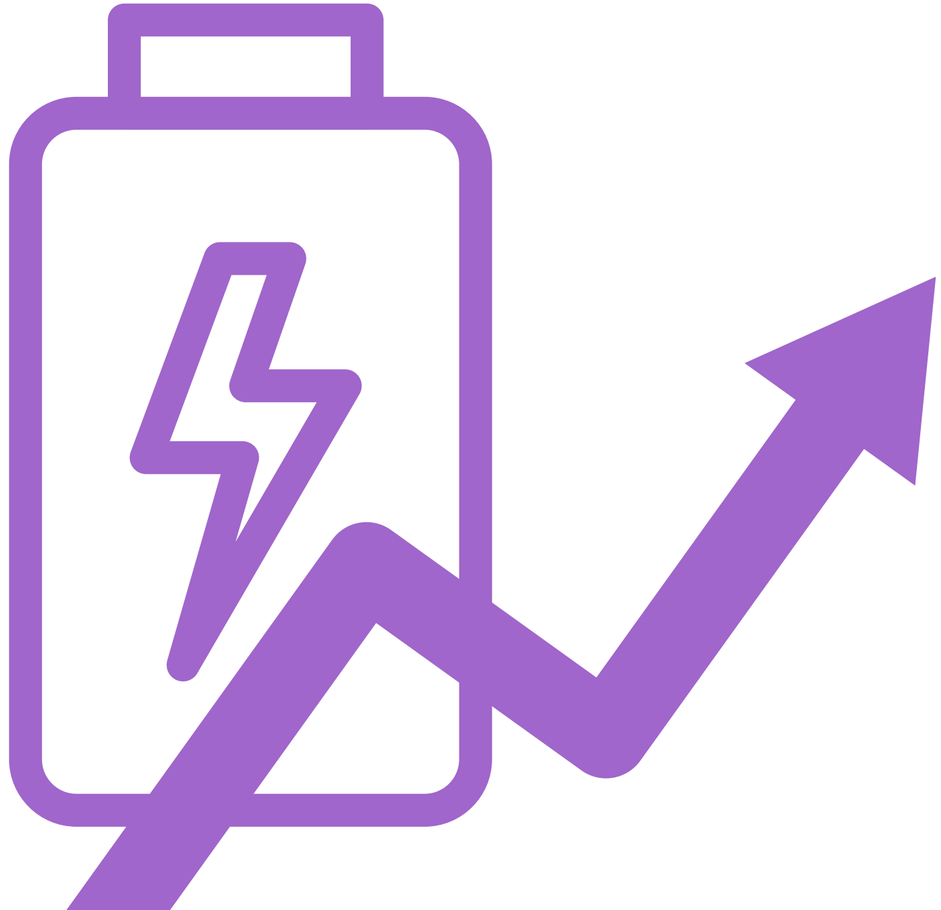
O que é big.LITTLE?

- Arquitetura de processamento heterogênea
- Criada pela ARM
- Construção de processadores multinucleares
- Único chip = Núcleos de alto desempenho + núcleos de alta eficiência energética



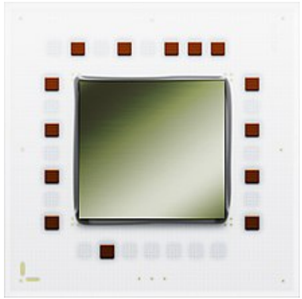
Por que big.LITTLE?

- "big" - Núcleos de alto desempenho: tarefas exigentes
- "LITTLE" - Núcleos de alta eficiência energética: atividades leves.
- Ajusta-se ao padrão de uso dinamicamente
- Otimiza o desempenho e a eficiência energética de processadores multinucleares.
- Maximização do desempenho do dispositivo e prolonga a vida útil da bateria.



Breve histórico

Histórico resumido do lançamento dos processadores multicore



1971

Lançado o primeiro microprocessador: Intel 4004

2001

Primeiro processador multicore do mundo: IBM Power4

2005

Intel e AMD lançam seus processadores multicores: Pentium D e Athlon X64 X2

Hoje

Quase todos as CPUs possuem pelo menos 2 núcleos.

O que havia antes?

- Núcleos idênticos nas CPUs
- Todas as tarefas, grandes ou pequenas, eram distribuídas nesses núcleos
- Frequência de clock fixa
- Consumo de energia constante
- Desempenho fixo

Quais os problemas?

- Ineficiência energética
- Desperdício de recursos de desempenho
- Relação desempenho-eficiência ruim
- Produção de calor
- Estresse dos componentes

Onde é usado o big.LITTLE?

Google Nexus (TV Box)

Chromebook

Smartphones (Galaxy S9, Samsung J4, Samsung J6, etc...)

Raspberry pi

Organização big.LITTLE

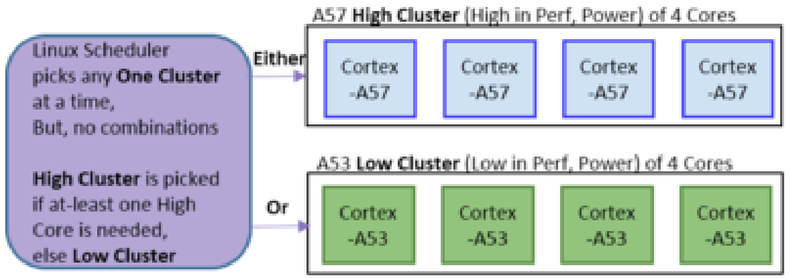
Existem 3 formas de organizar um núcleo big.LITTLE

Clustered Switching (comutação agrupada)

In-kernel switcher (CPU migration)

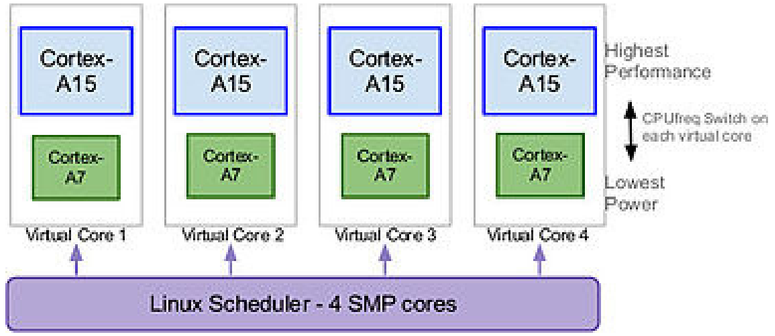
Heterogeneous multi-processing (global task scheduling)

Clustered Switching (comutação agrupada)



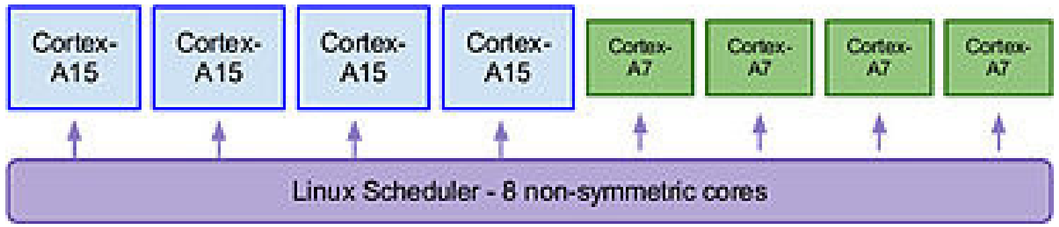
- Implementação mais simples
- Clusters big e LITTLE de mesmo tamanho
- O SO vê um cluster por vez
- Transição de estado entre os cluster baseada na carga no processador

In-kernel switcher (CPU migration)



- Divisão do CPU em núcleos virtuais com um par big e LITTLE
- Somente um núcleo do conjunto virtual está ativo em um dado momento
- Transição é realizada com base na demanda
- Troca entre núcleos utiliza o cpufreq framework [Kernel Linux 3.11]

Heterogeneous multi-processing (global task scheduling)



- Todos os núcleos heterogêneos são acessíveis ao mesmo tempo
- Threads de alta prioridade são alocadas aos core big
- Threads de baixa prioridade e de baixa intensidade computacional (e.g. tarefas de plano de fundo) são alocadas aos core LITTLE

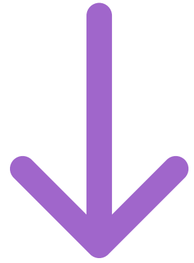
Code Schedulling

A maneira de alocar tarefas é característica de cada organização de um núcleo big.LITTLE

Clustered Switching (comutação agrupada)

In-kernel switcher (CPU migration)

Heterogeneous multi-processing (global task scheduling)



Completely Fair Scheduler

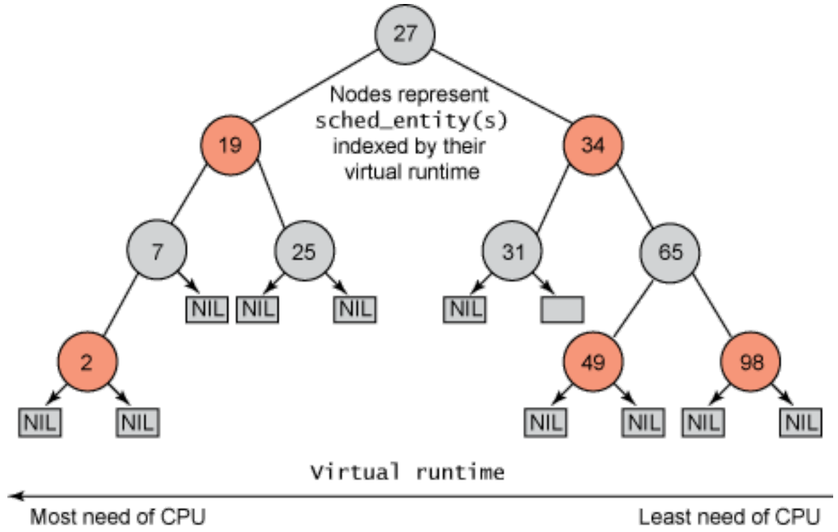
CFS + Energy-Aware Scheduling

Completely Fair Scheduler

De forma simples: Modela uma CPU ideal e precisa em multi-tasking em hardware real.

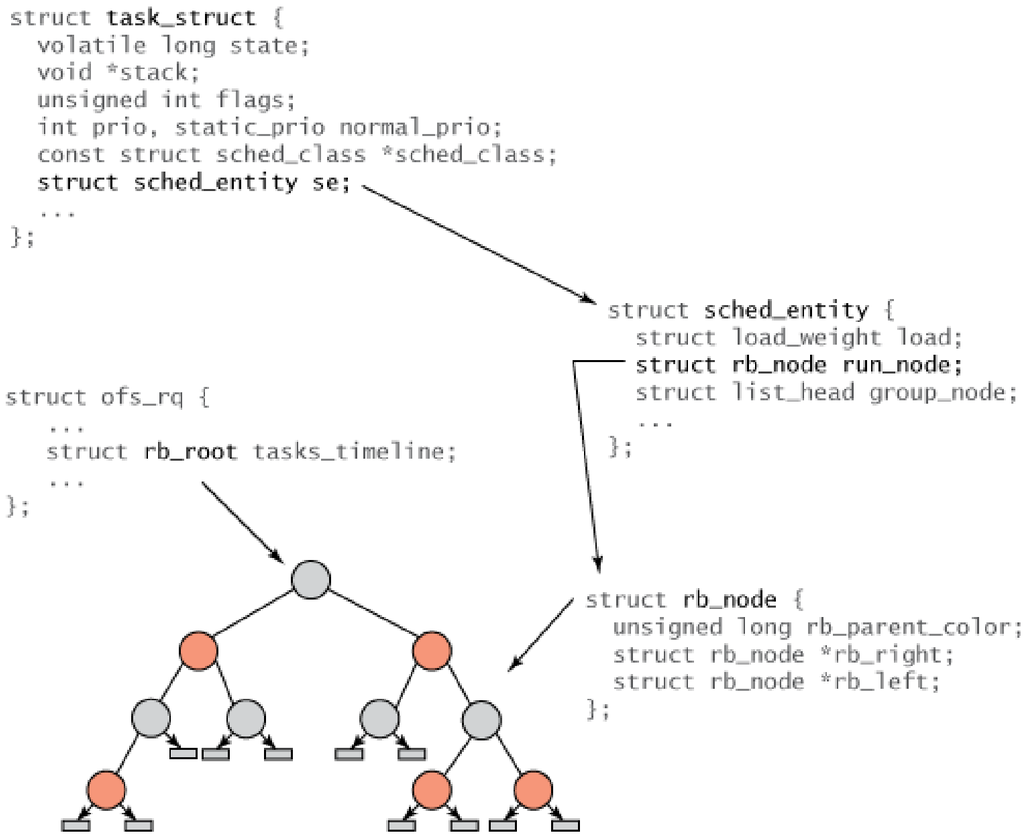
Virtual Runtime: especifica a próxima fração de tempo em que a execução da tarefa inicia na CPU ideal.

CFS sempre tenta dividir o tempo de CPU entre tarefas executaveis o mais próximo possível do hardware multitasking real.



Completely Fair Scheduler

- Implementado com um árvore Rubro Negra ordenada por tempo. Gera um "timeline" da execução das próximas tasks.
- Virtual Runtime: especifica a próxima fração de tempo em que a execução da tarefa inicia na CPU ideal.
- Sempre escolhe a tarefa folha mais a esquerda na árvore e tarefas executadas são movidas mais a direita.



Energy-Aware Scheduling

- Tem como objetivo otimizar a eficiência energética do sistema levando em consideração as características de consumo de energia dos núcleos big e LITTLE.
- Avalia a carga de trabalho e as exigências de desempenho das tarefas em execução.
- Decide qual núcleo é mais adequado para executar cada tarefa, visando minimizar o consumo total de energia.
- Busca maximizar Performance [inst/s] / Power [W]. Equivalente a minimizar Energy [J] / Instruction.

Energy-Aware Scheduling

- Utiliza a Capacidade de uma CPU.

```
CPUs:   0 1 2 3 4 5 6 7 8 9 10 11
PDs:   |--pd0--|--pd4--|---pd8---|
RDs:   |----rd1----|-----rd2-----|
```

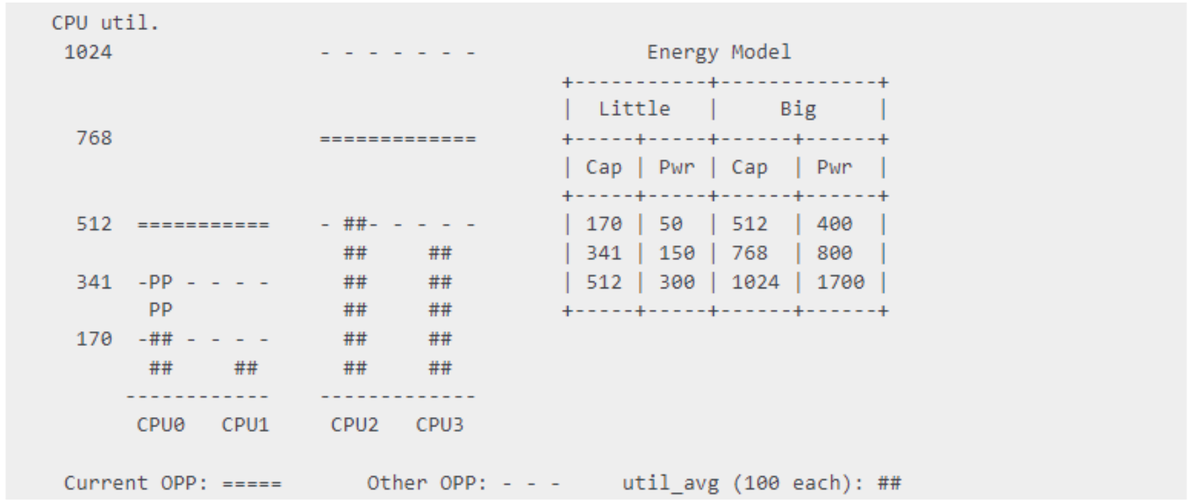
- Capacidade é normalizada em um intervalo de 1024 e calculada pelo Per-Entity Load Tracking (PELT)
- Sobrescreve o balanceamento de "wake-up" do CFS
- Busca a CPU com maior capacidade no domínio disponível em cada dominio de performance. Verifica se há economia de energia comparado ao rodar no último núcleo onde a tarefa foi ativada.

Energy-Aware Scheduling: Exemplo

O escalonador deve determinar a qual CPU atribuir uma tarefa P que custa em média avg_util = 200 e tem prev_cpu = 0

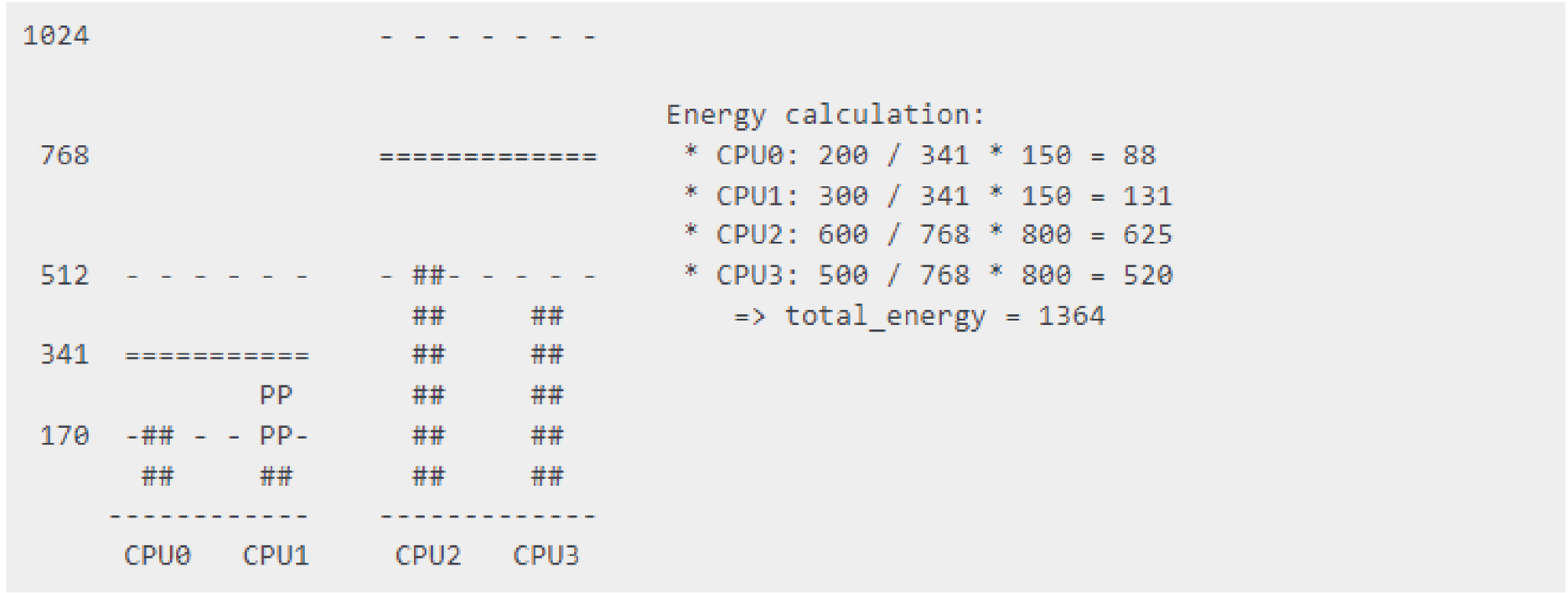
Nesse momento, a carga distribuida entre as CPUs 0-3 é, respectivamente, util_avg 400, 100, 600 e 500.

Além disso, cada domínio de performance tem três Pontos de Operação de Performance.



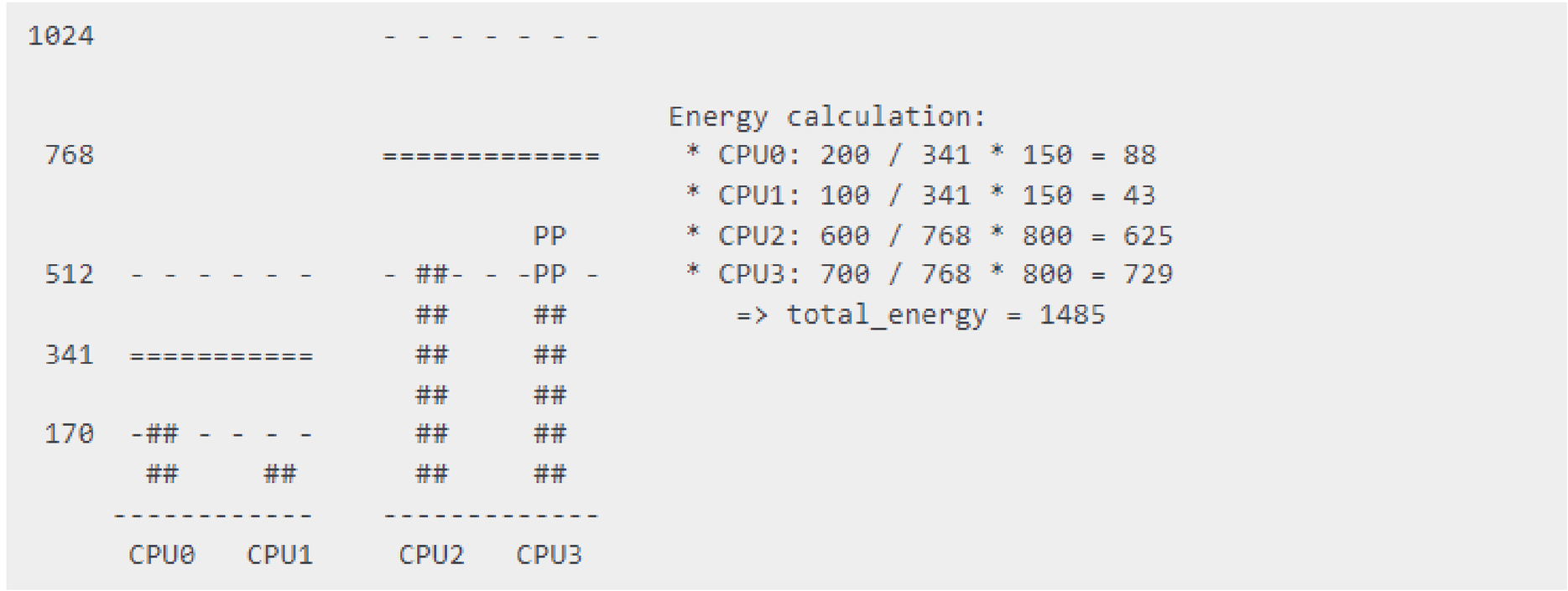
Energy-Aware Scheduling: Exemplo

Caso 1: Tarefa P é migrada para CPU 1



Energy-Aware Scheduling: Exemplo

Caso 2: Tarefa P é migrada para CPU 3



Energy-Aware Scheduling: Exemplo

Caso 3: Tarefa P continua em CPU 0

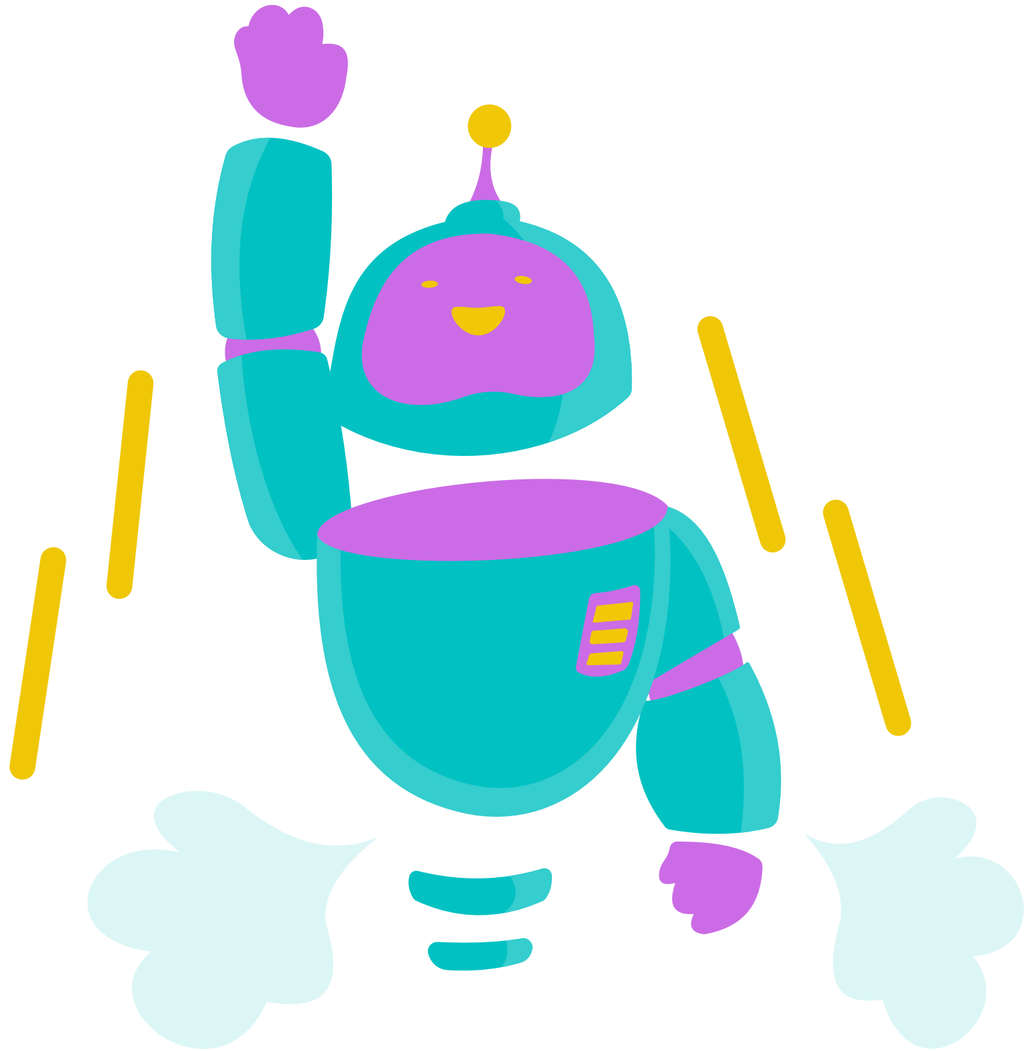
```
1024                - - - - - - -

                                       Energy calculation:
 768                =============       * CPU0: 400 / 512 * 300 = 234
                                        * CPU1: 100 / 512 * 300 = 58
                                        * CPU2: 600 / 768 * 800 = 625
 512  ===========   - ##- - - - -       * CPU3: 500 / 768 * 800 = 520
                      ##     ##            => total_energy = 1437
 341  -PP - - - -     ##     ##
       PP             ##     ##
 170  -## - - - -     ##     ##
       ##     ##      ##     ##
     ------------   -------------
      CPU0   CPU1     CPU2   CPU3
```

DinamIQ: o sucessor do big.LITTLE

- DinamIQ é uma nova arquitetura de processador desenvolvida pela ARM, que substitui o antigo big.LITTLE.

- Permite combinar núcleos de diferentes tipos e desempenhos em um único cluster, em vez de ter dois clusters separados de núcleos big e LITTLE.

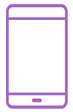
- Permite que os núcleos se comuniquem diretamente entre si, sem passar pelo interconector do chip, o que reduz a latência e o consumo de energia.



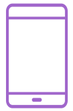
Quais são as vantagens do DinamIQ?

- Possibilidade de personalizar o número e o tipo de núcleos de acordo com as necessidades de cada dispositivo
- DinamIQ oferece mais flexibilidade e escalabilidade para os fabricantes de processadores
- Mais eficiência e desempenho para os usuários finais, que podem aproveitar os benefícios dos três métodos de agendamento.
- Os núcleos se adaptam dinamicamente à carga de trabalho, alternando entre os modos de operação e ajustando a frequência e a tensão

Onde é usado o DinamIQ ?



Kirin 970 da Huawei



Snapdragon 845 da Qualcomm

o Exynos 2100 da Samsung

Membros do grupo

João Miguel

César

Renan

Otavio

Obrigado pela atenção