Power saving scheduling for heterogeneous architectures like ARM big.LITTLE

Agenda

- **Introduction**
  - Scheduling Goals
  - Naive Scheduling
  - Idea of Energy Efficient Scheduling
  - The big.LITTLE Architecture
- **ARM HMP Scheduler**
- **Queue Based Scheduling**
- **POET: A Portable Approach to Minimizing Energy**
- **Model Based Scheduling**
- **Summary**

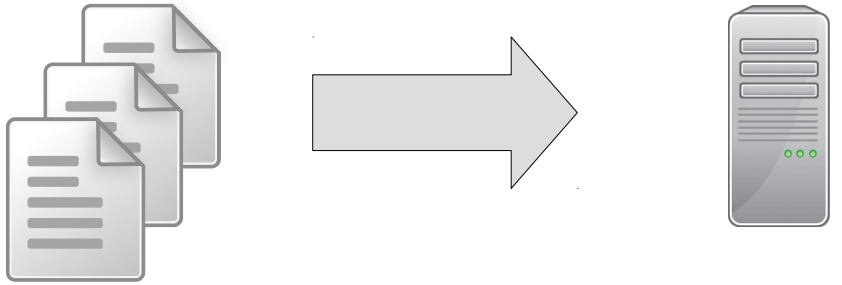
Scheduling Goals

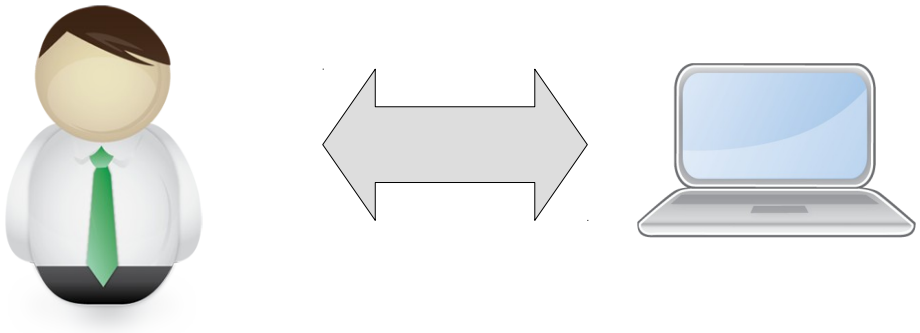
Common scheduling goals:

- Fairness
- Load Balancing

Environment dependant:

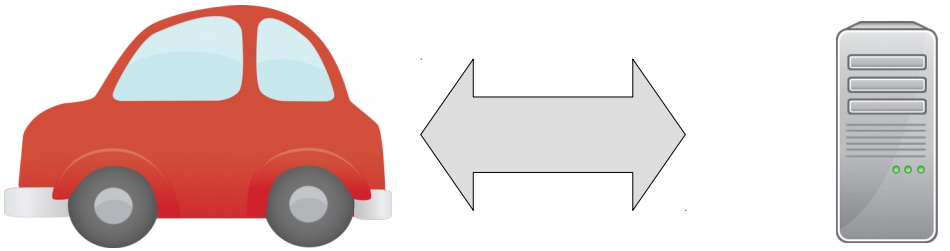
- Throughput - batch
- Latency – interactive
- Deadlines - real-time





Energy efficient scheduling goals:

- Reduce energy consumption
- Energy efficiency (e.g. tasks per Joule)



Naive Scheduling

What would naive scheduling do?

All cores are considered equal and the load would be balanced

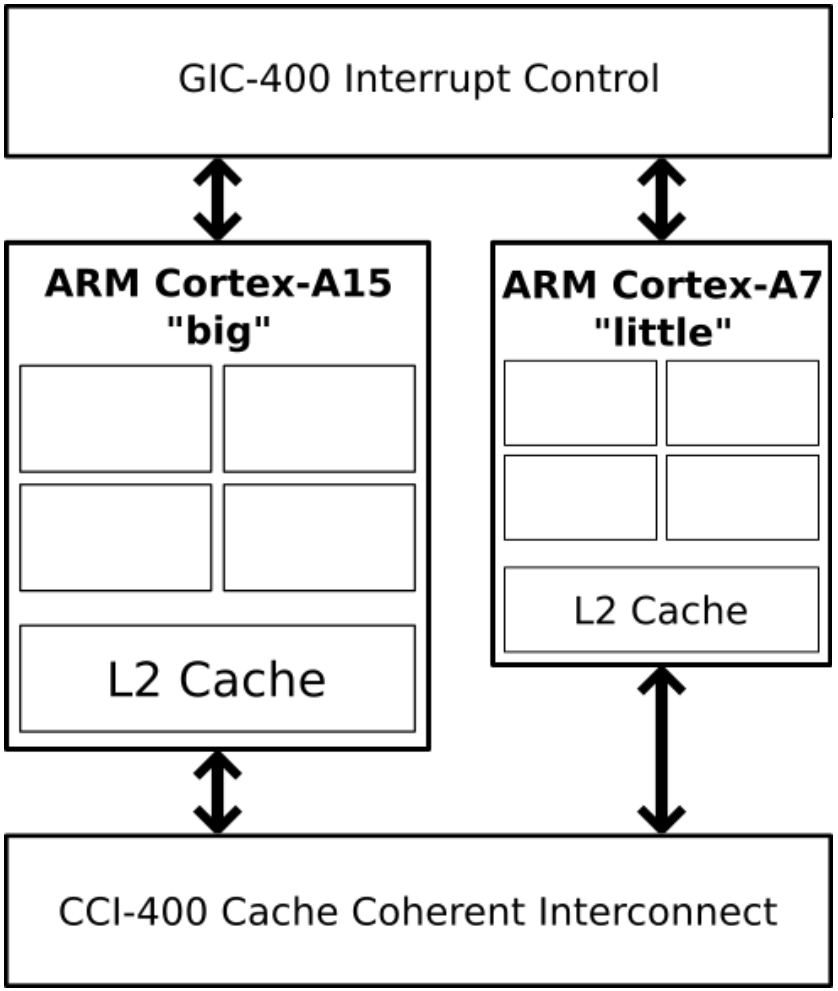
- No low energy operating mode where only little cores are used
- Heavy tasks might not be executed on a big core
- Gang scheduling (used for collaborative threads) would waste performance

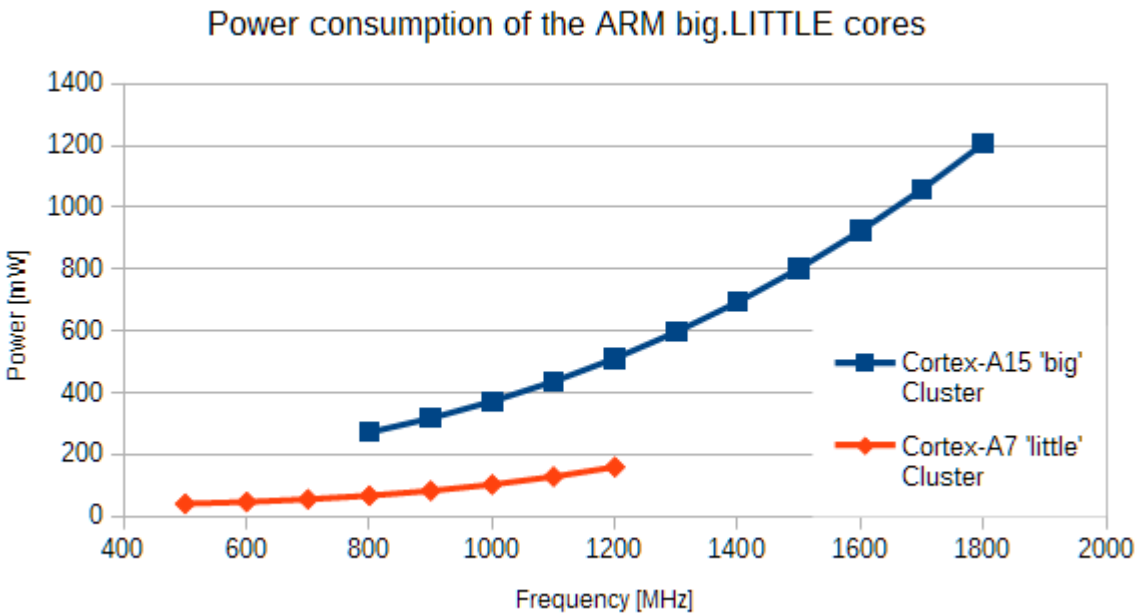
Idea of Energy Efficient Scheduling

Idea:

- There are different kinds of cores which have a different power model and energy efficiency
- Focus on thermal budget and energy consumption instead of performance only
- Use the little cores for lightweight task and the big ones for computational demanding tasks
- Prevent big cores from throttling down

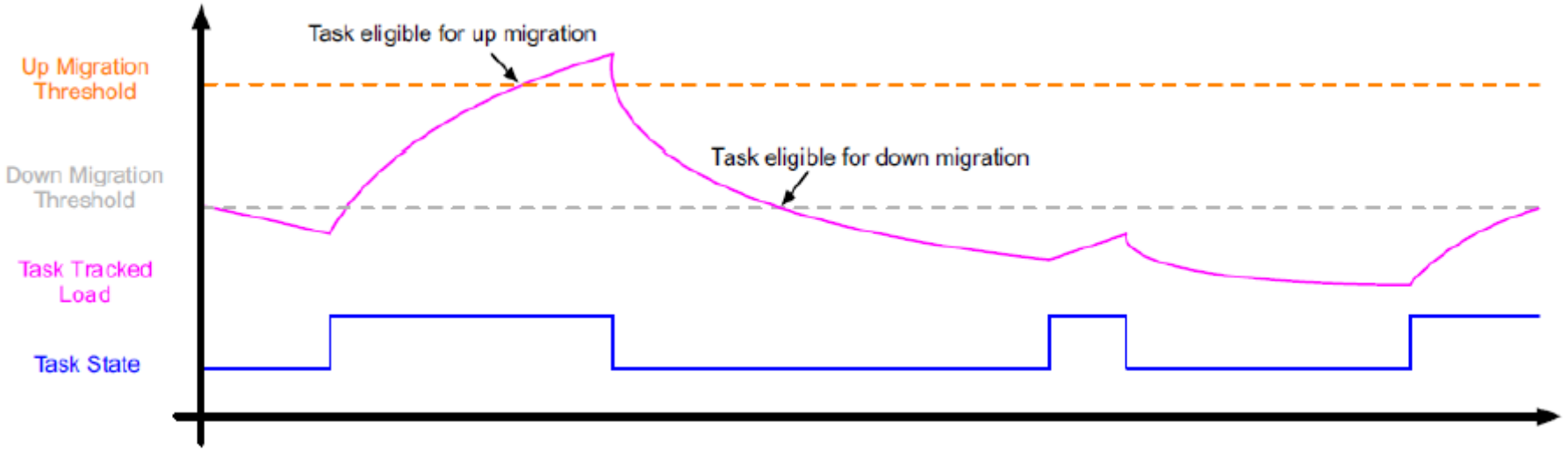
The big.LITTLE Architecture



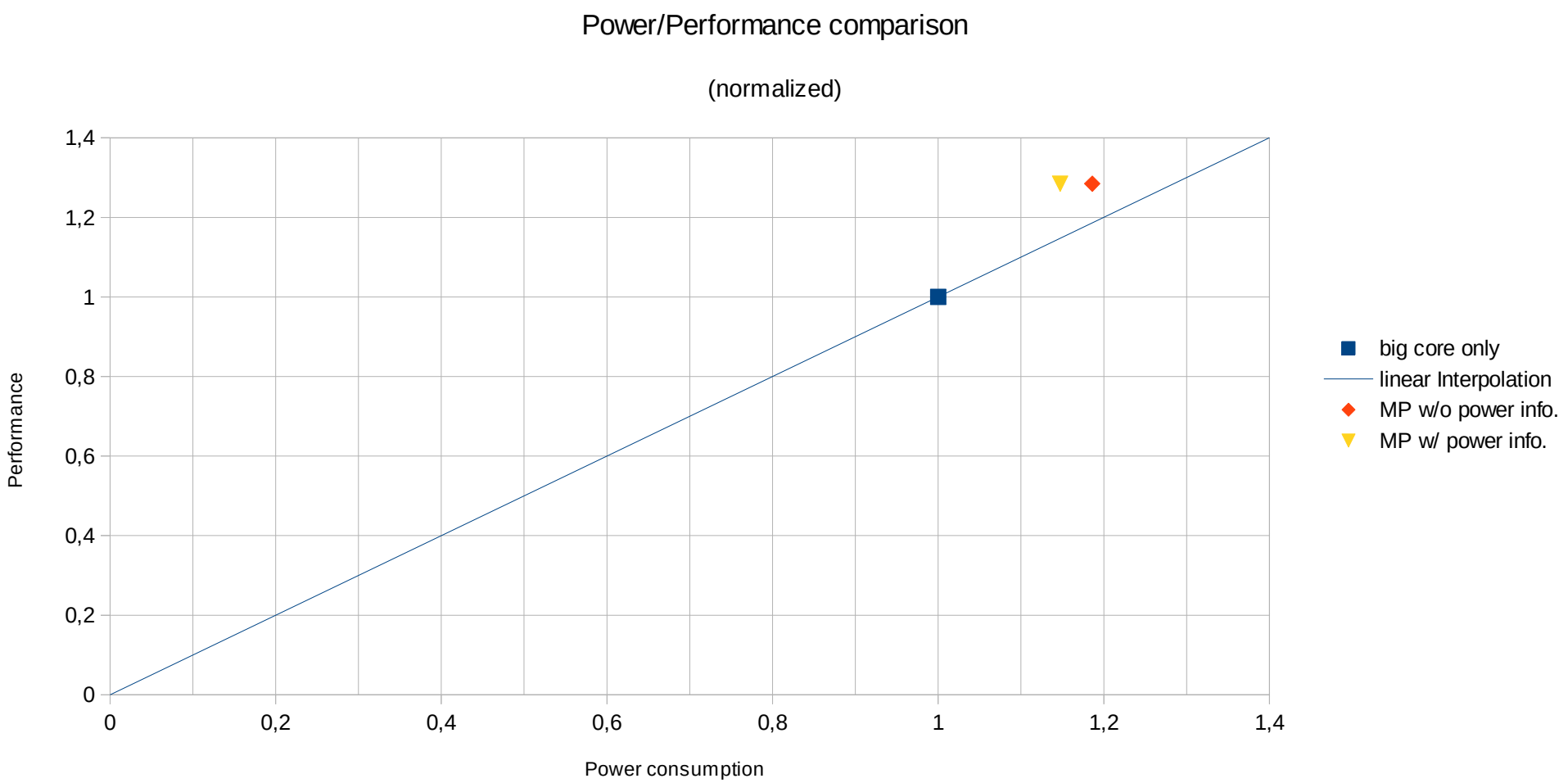


ARM HMP Scheduler

- Based on Completely Fair Scheduler
- Tasks are moved up or down if the load after a scheduling period reaches a certain threshold
- Load balancing within the clusters
- Load tracking considers CPU frequency → DVFS compatibility



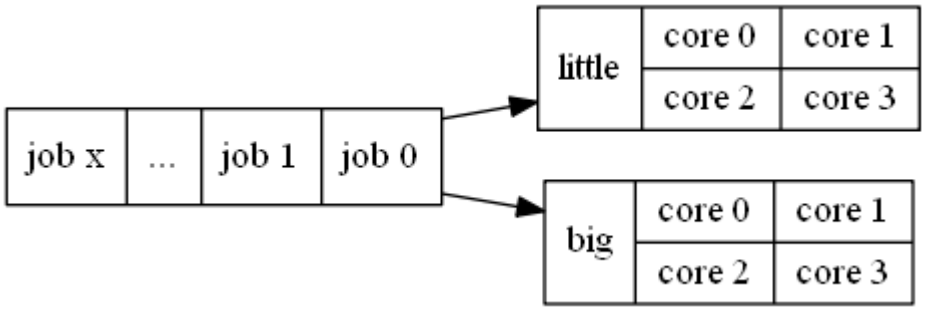
ARM HMP Scheduler - Performance



Queue Based Scheduling

System like web servers have a queue with outstanding requests

- Each request shall be processed within a certain service time
- In order to save energy use the big cores only when there is a certain load

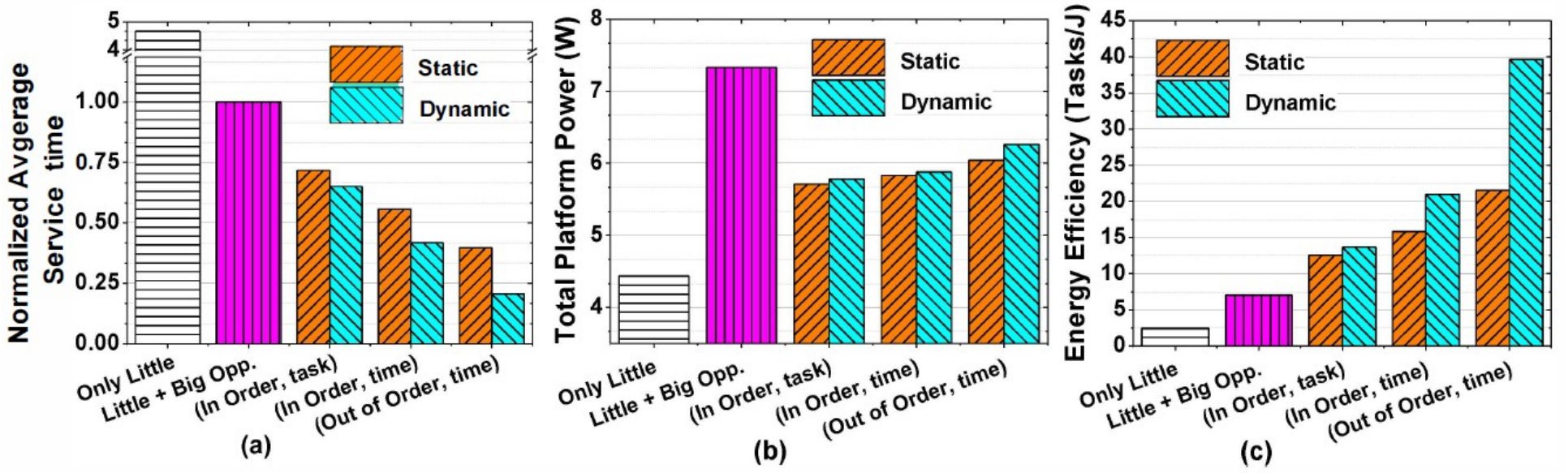


Queue Based Scheduling in Detail

```
procedure SCHEDULETASK, Input: PreferredServer
    PreferredServer ← idle
    NonPreferredServer ← idle
    while TaskQueue is not empty do
        if PreferredServer is idle then
            Schedule the next job to the PreferredServer
        if (TaskQueueSize >= Threshold) AND
            (NonPreferredServer is idle) AND
            (There is no thermal violation) then
                Schedule the next job to the NonPreferredServer
```

Threshold can be static, or adapted dynamically!

Queue Based Scheduling - Performance



Following techniques improve the performance even more:

- Execution time prediction
- Out of Order execution

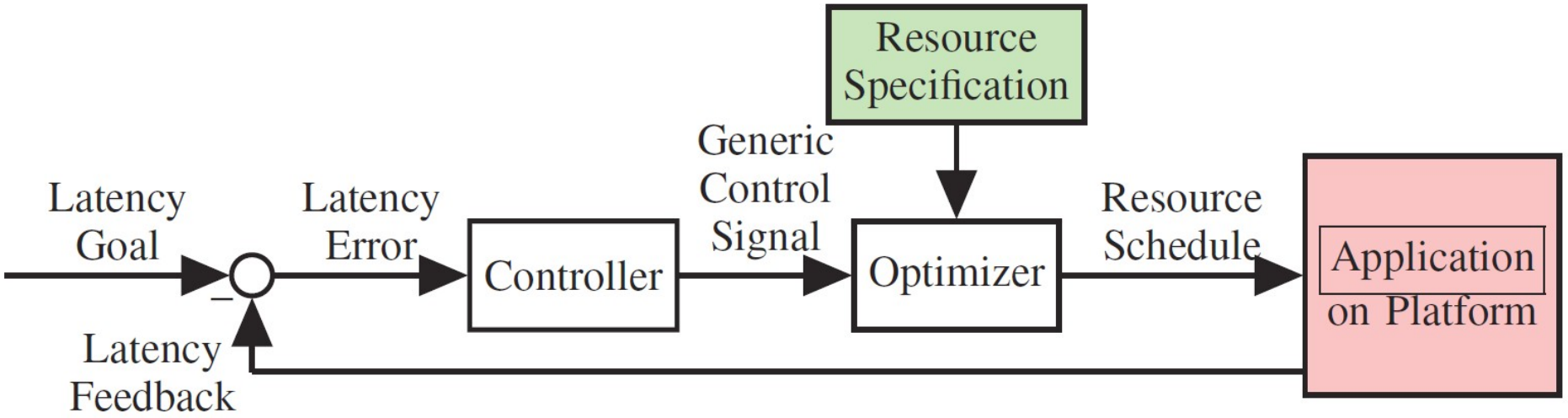
POET: A Portable Approach to Minimizing Energy

POET (Performance with Optimal Energy Toolkit) is a portable c-library to minimize energy consumption under soft real-time constraints.

- User provides a model with different core configurations
- Digital control is used to control the speed-up of the application
- So called “optimizer” dispatches the task onto the resources
- Optimization only for one application because dispatching of the tasks would get too complicated.

Goal: meet the deadline, with the minimal amount of energy

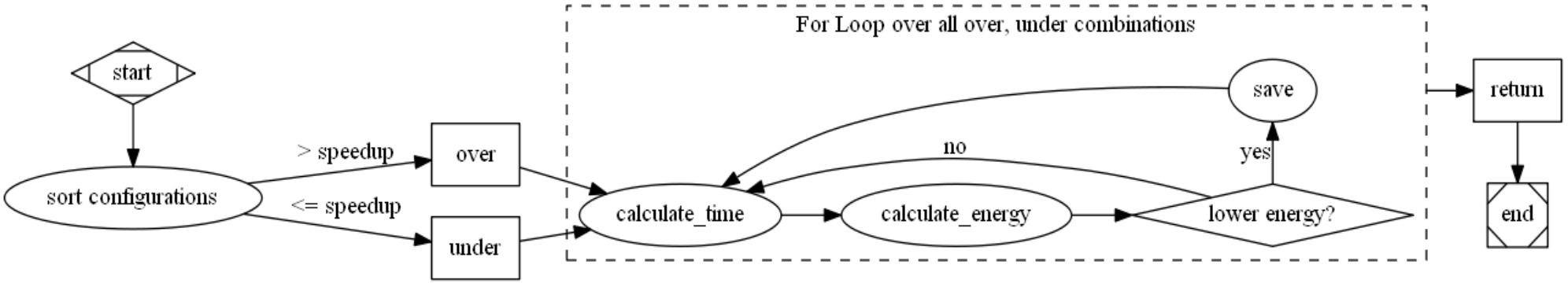
POET in Detail



#id	speedup	powerup
0	1	1
1	1.20	1.09
2	1.40	1.16
3	1.60	1.30
4	2.12	1.35
5	2.53	1.50
6	2.88	1.64
7	3.18	1.69

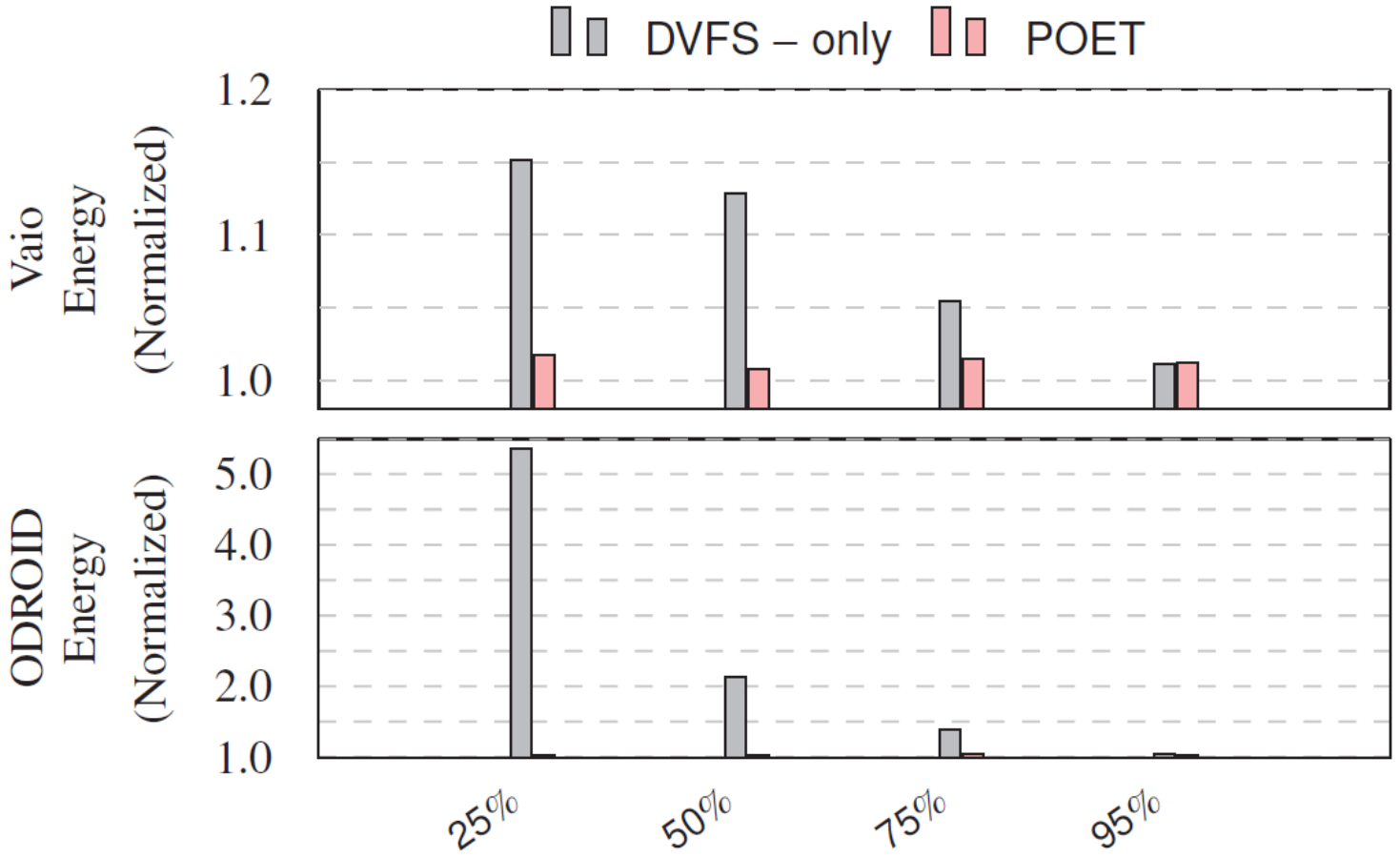
#id	frequency	cores
0	250000	0
1	300000	0
2	350000	0
3	400000	0
4	250000	1
5	300000	1
6	350000	1
7	250000	2

POET Optimizer



- over: all configurations which provide a higher speed-up then required
- under: all configurations which provide a lower speed-up then required
- calculate_time: determine how much time is spend in each configuration
- calculate_energy: calculate energy consumption

POET - Performance



Model Based Scheduling – Real-Time Applications on Heterogeneous Processors

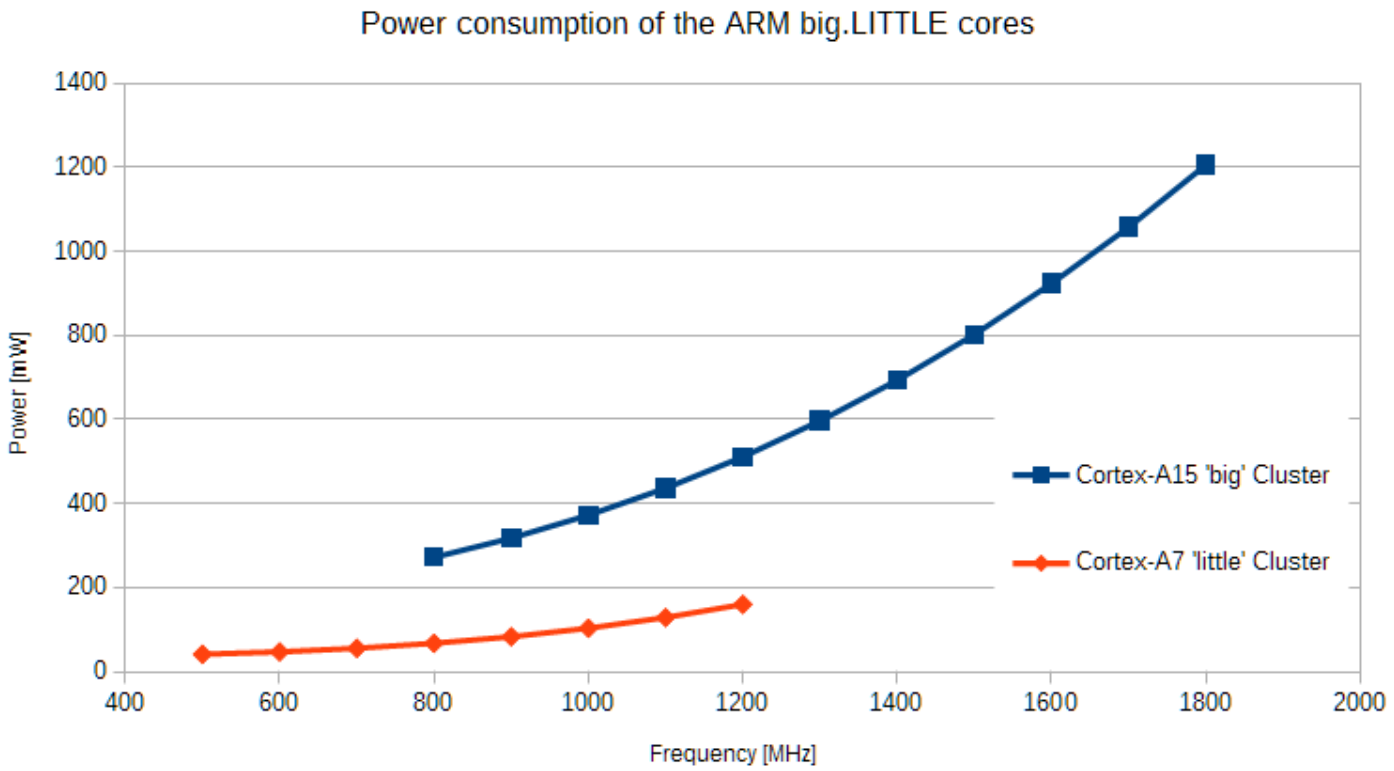
- Computational load with deadlines and available processors can be modelled
- Focus is on real-time applications with sets of periodic tasks

→ optimize the task partition for energy consumption with deadlines as constraints

But problem is NP-hard!

→ Use heuristics to get feasible algorithms

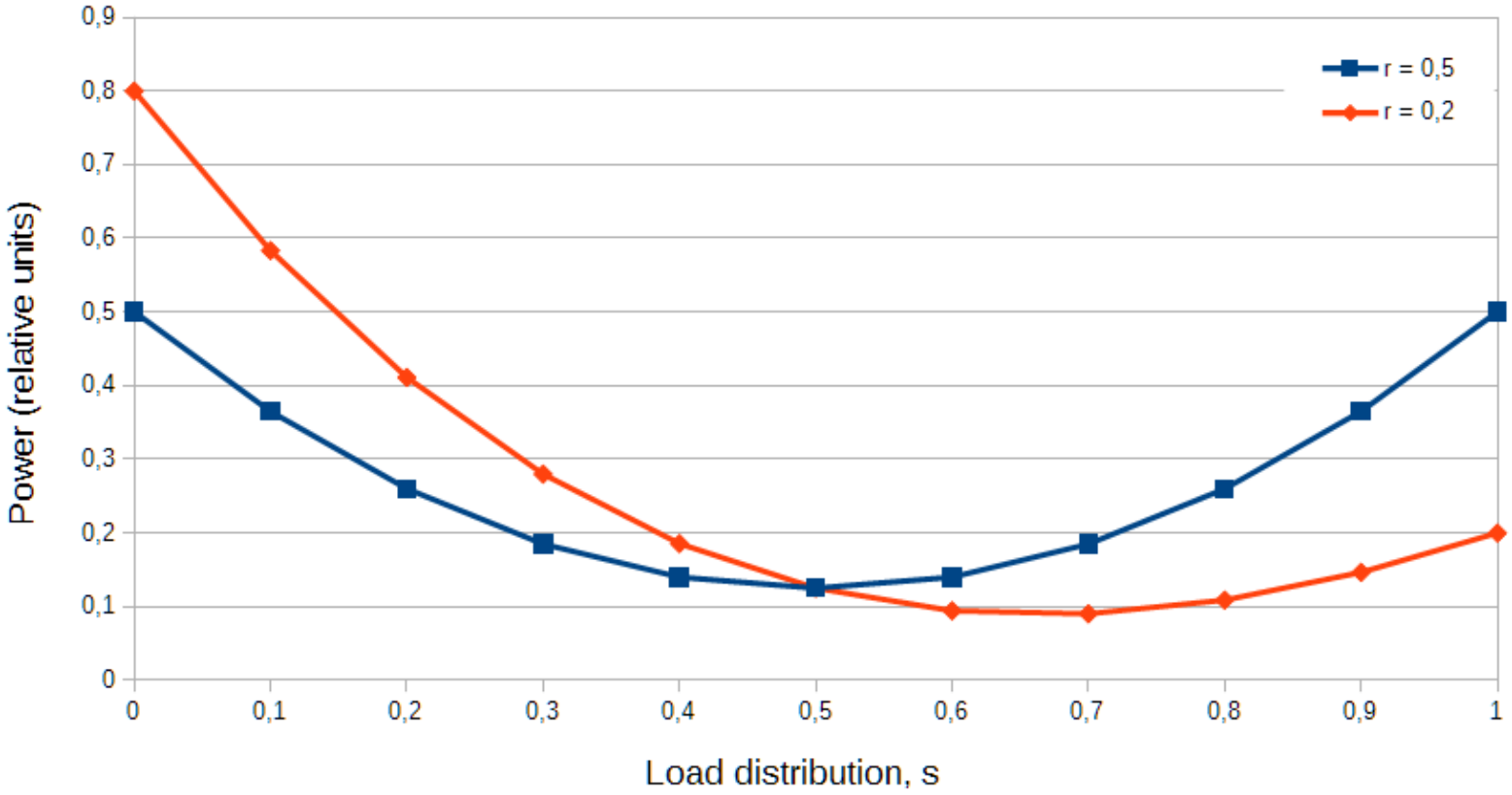
Heterogeneous Processor Energy Modell



$$P(f) = \kappa f^{\alpha} + \beta$$

Core	κ	α	β
A-7	1.00E-8	3.28	34.24
A-15	2.91E-6	2.63	146.49

Load Distribution on Heterogeneous Processors



$$r = \frac{\kappa_1}{\kappa_1 + \kappa_2}$$

$$s = \frac{U_1}{U_1 + U_2}$$

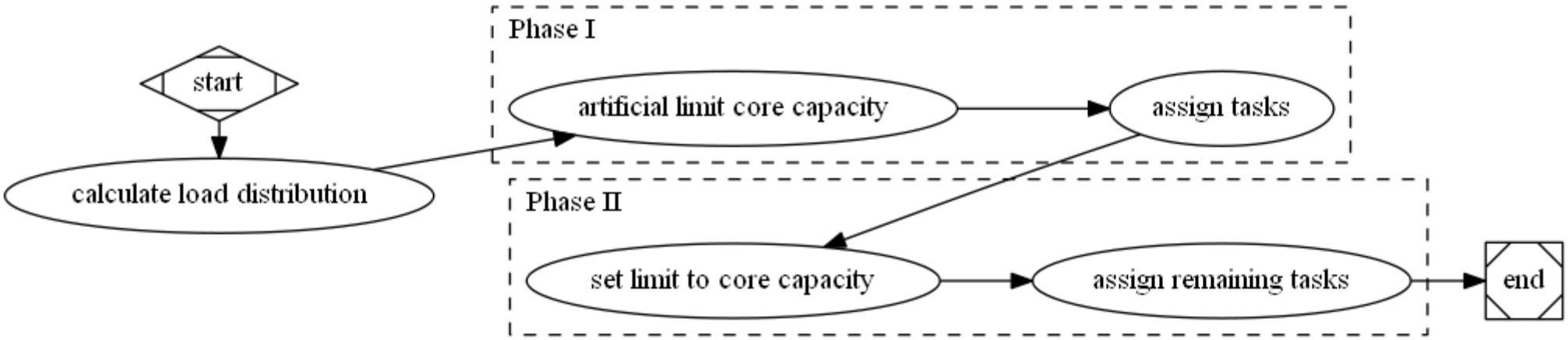
Real-Time Applications on Heterogeneous Processors – Scheduling Heuristics

Naive (Load Balancing): Sort tasks descending by computational demand and assign each task to the processing unit (PE) with the least load at that point

Marginal Power (M-PWR): Sort tasks descending by computational demand and assign it to the PE where is will have the least power consumption

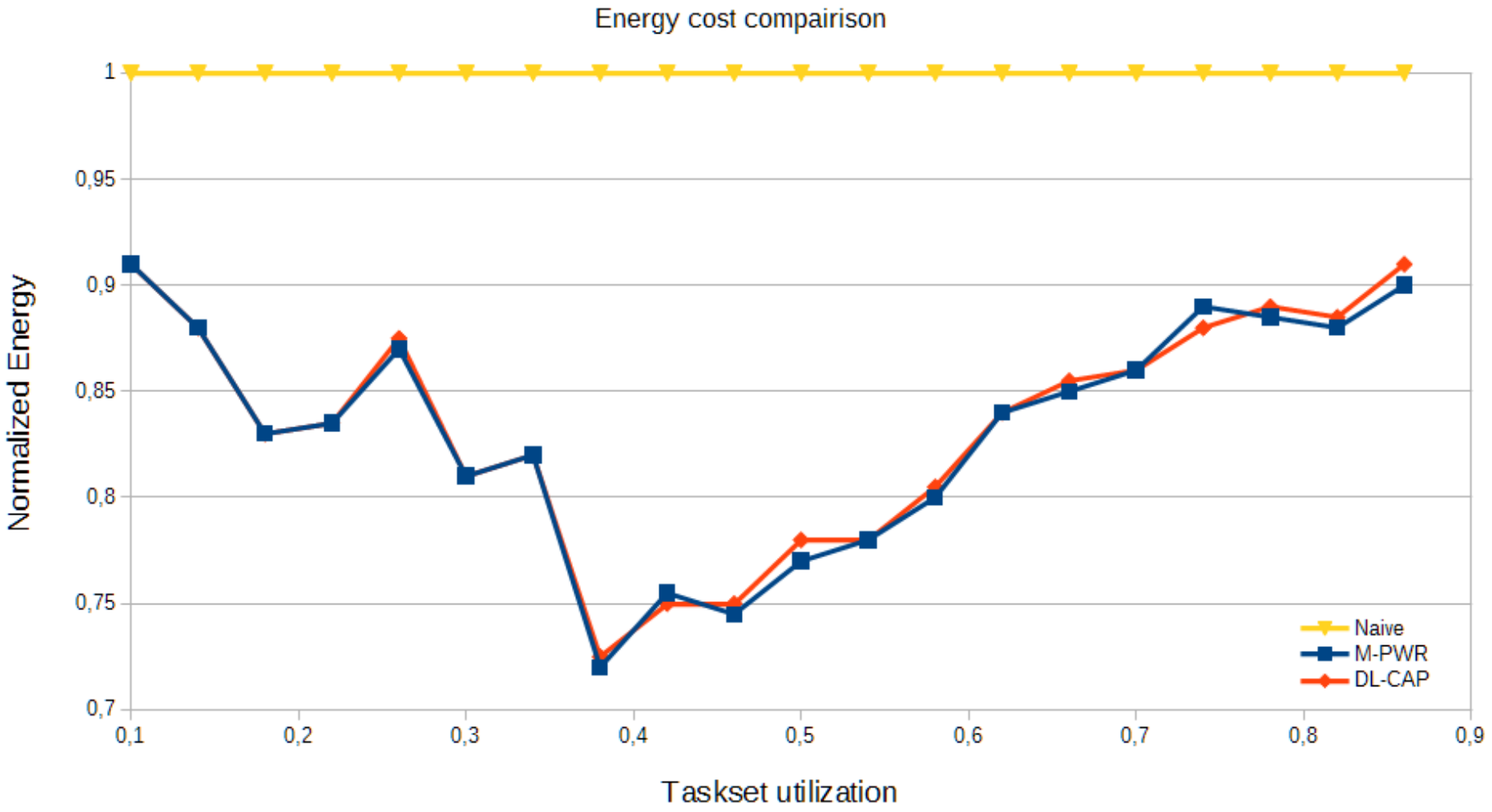
Real-Time Applications on Heterogeneous Processors – Scheduling Heuristics

DL-CAP:



- Assigning of tasks can be done with load balancing or marginal power algorithm
- Using marginal power in phase II leads to optimal results

Real-Time Applications on Heterogeneous Processors - Performance



Summary

Scheduler	Applications	Portable	Effort	Real-Time Support	Improvement (Energy Consumption)
ARM HMP Scheduler	any	yes	least	no	~ 5%
Queue Based Scheduler	applications with indepedent tasks of similar kind	no	moderate	soft real-time	~15%
POET	any (only single applications)	yes	little	soft real-time	~81%
Heterogeneous Load Distribution	any	no	high	hard real-time	~38%

The End

Thank you for your attention!

References:

[1] Kisoo Yu et al; Power-aware task scheduling for big.LITTLE mobile processor

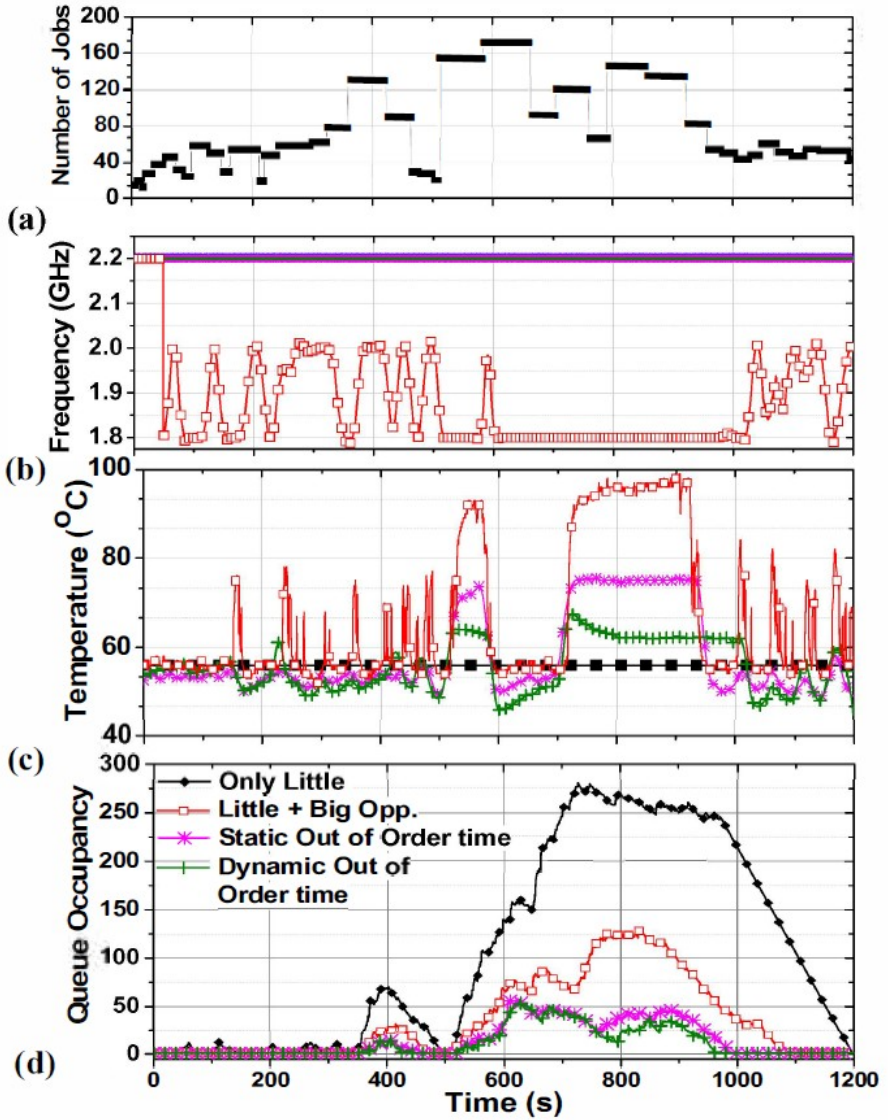
[2] Colin, A. et al.; Energy-efficient allocation of real-time applications onto Heterogeneous Processors

[3] Imes, C. et al.; POET: a portable approach to minimizing energy under soft real-time constraints

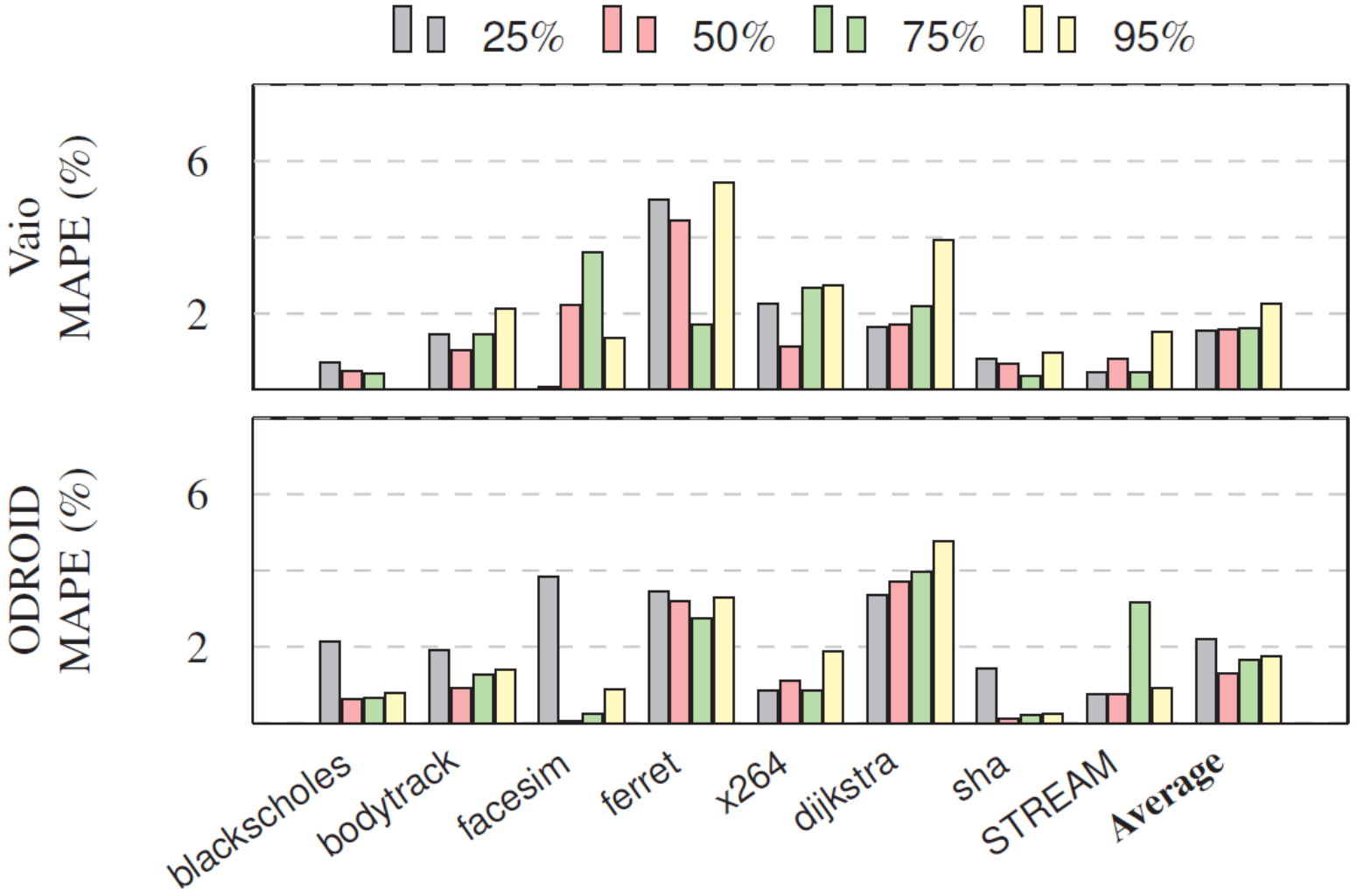
[4] Jain, S. et al.; Energy efficient scheduling for web search on heterogeneous microservers

[5] ARM Whitepaper; big.LITTLE Technology: The Future of Mobile

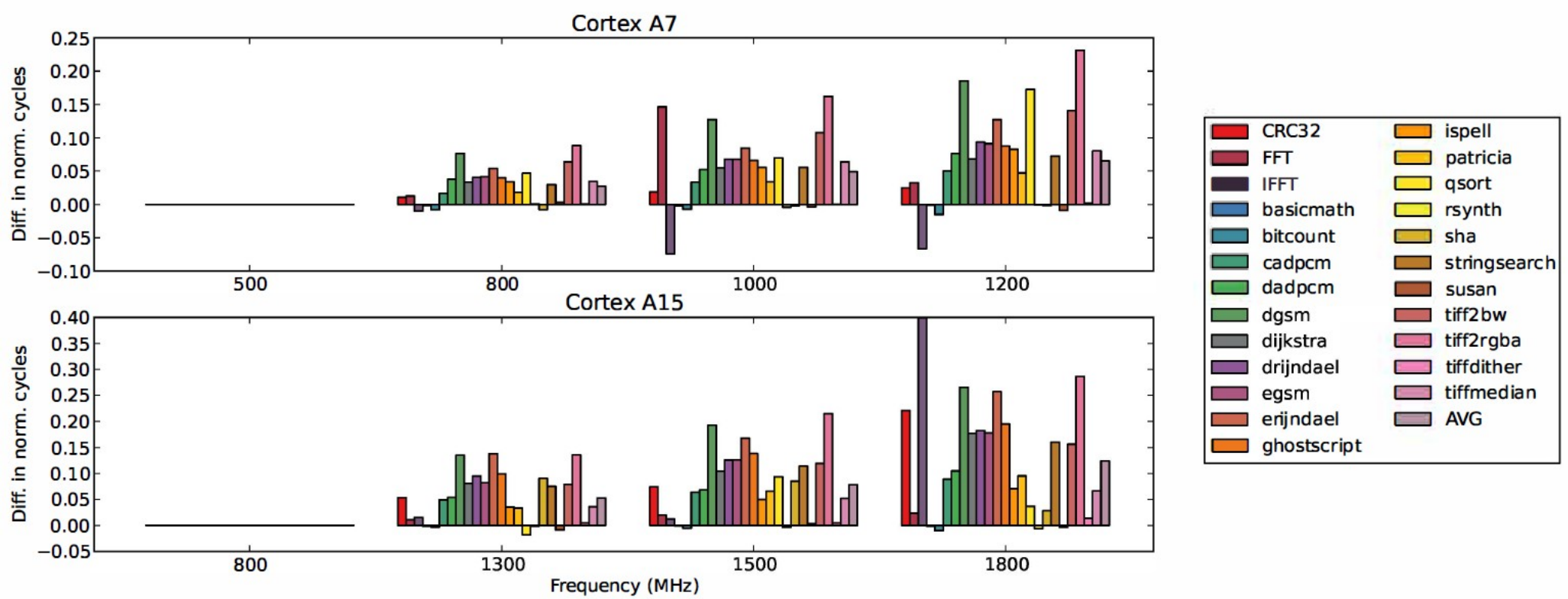
Bonus Slide – Queue Scheduling



Bonus Slide - POET



Bonus Slide – Computational Load



Bonus Slide – Marginal Power