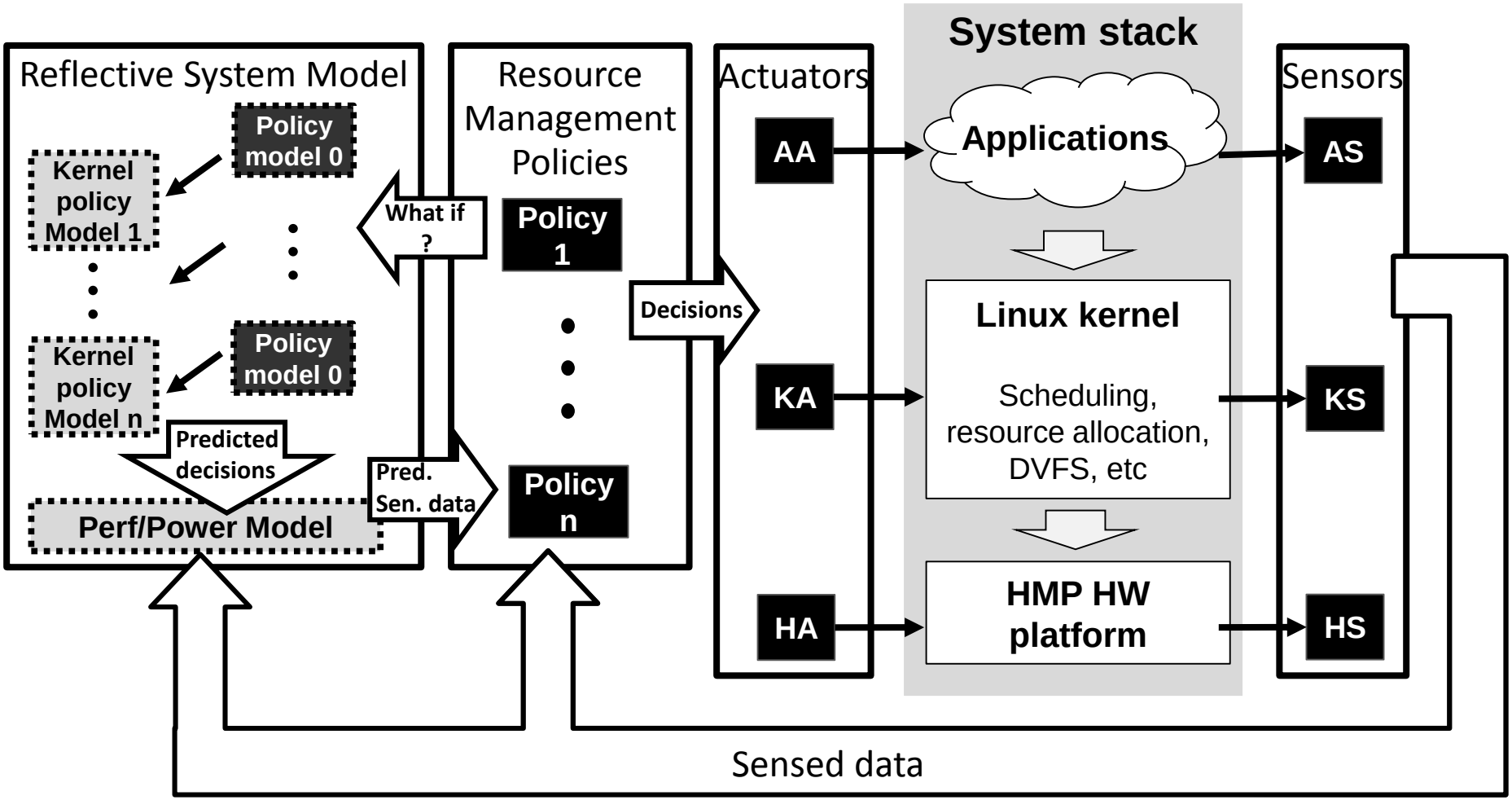
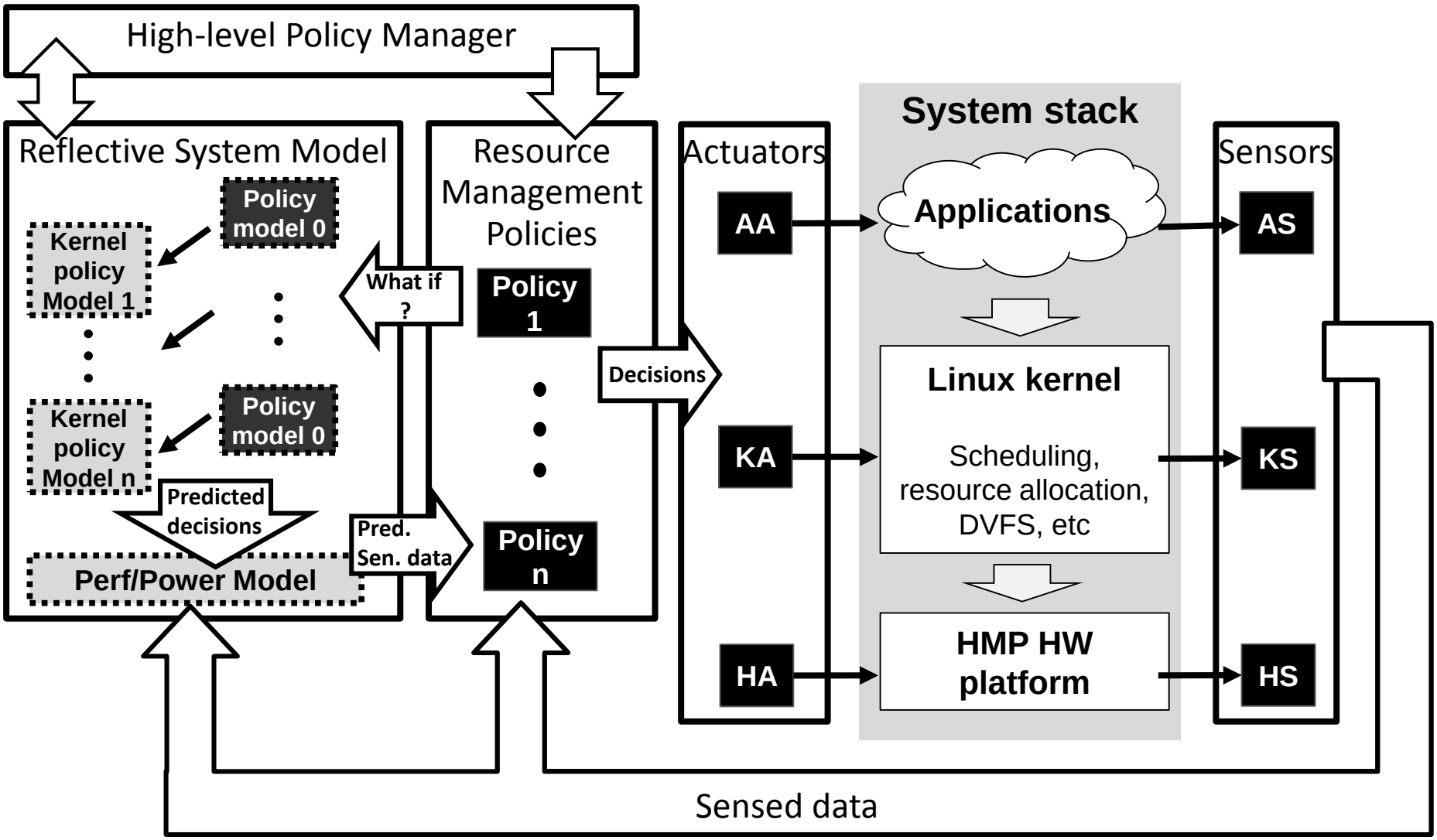
MARS Framework

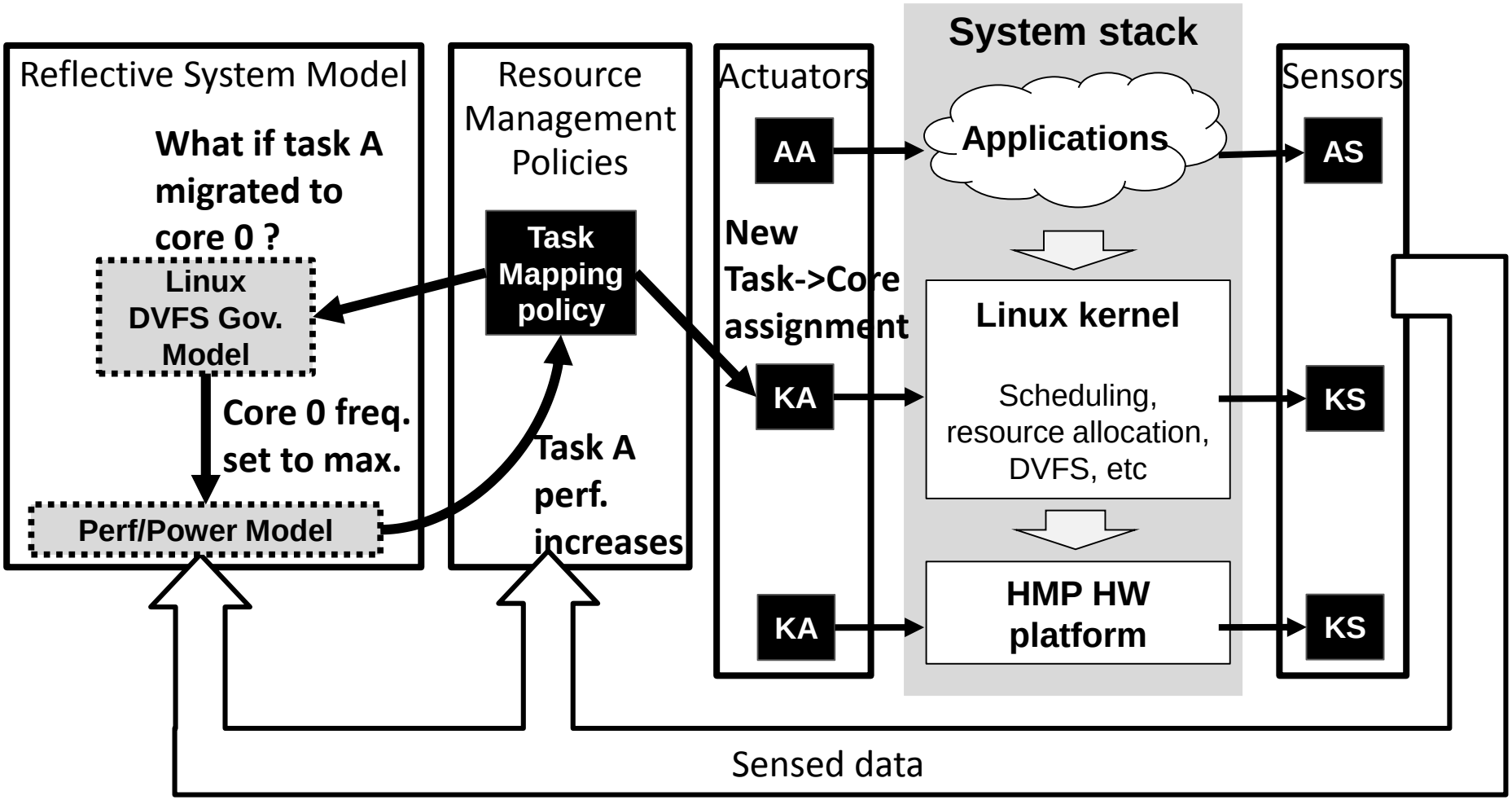
A Framework for resource management



A Framework for resource management



Example: reflective task mapping



MARS usage

- Crating a resource manager with two policies:

```
class Simple_Map_Policy : public Policy_Manager {
  void setup() {
     registerPolicy (new Simple_Map_Policy()), 150 );
     registerPolicy (new Simple_DVFS_Policy(), 50 );
  }
};
```

MARS interface to user policies

- ***sense*** - *r*eturns sensed data for the latest execution period.

  E.g.: for a policy that executes every 50ms

  sense<SEN_INSTR_TOTAL>(task)

  returns the total number of instructions in the last 50ms

- ***actuate*** - sets an actuation knob

  E.g.: changing a task mapping:

  actuate<ACT_TASK_MAP>(task,core.id);

- ***senseIf*** - returns predicted sensed data for the next execution period.
- ***tryActuate*** - "What if I perform an actuation ?"
  - tryActuate affects senseIf results but it does not actually sets physical actuation values

Simple_Map_Policy implementation

- If a task performance drops below stablished goal, searches for a new task mapping

```
class Simple_Map_Policy : public Policy {
  void execute(){
    auto task = sys.info().tasks[0];
    double ips = sense<SEN_INSTR_TOTAL>(task) / sense<SEN_TIME_TOTAL_S>(task);
    if (ips < GOAL)
      for(auto core : sys.info().cores) {
        tryActuate<ACT_TASK_MAP>(task,core.id);
        ips = senseIf<SEN_INSTR_TOTAL>(task) / senseIf<SEN_TIME_TOTAL_S>(task);
        if (ips >= GOAL){
          actuate<ACT_TASK_MAP>(task,core.id);
          break;
        }
      }
  }
};
```

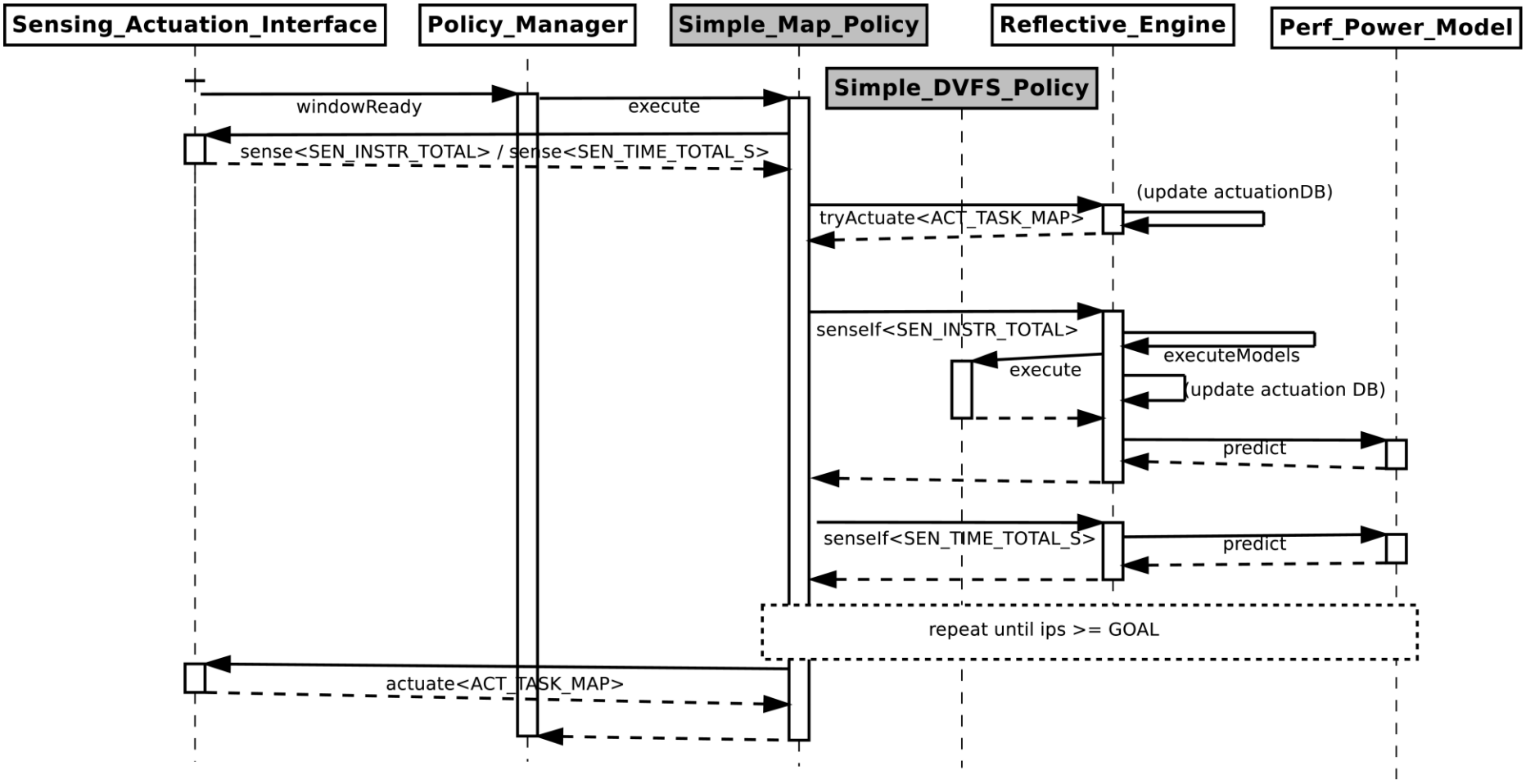
Abstractions for system entities

Reading performance

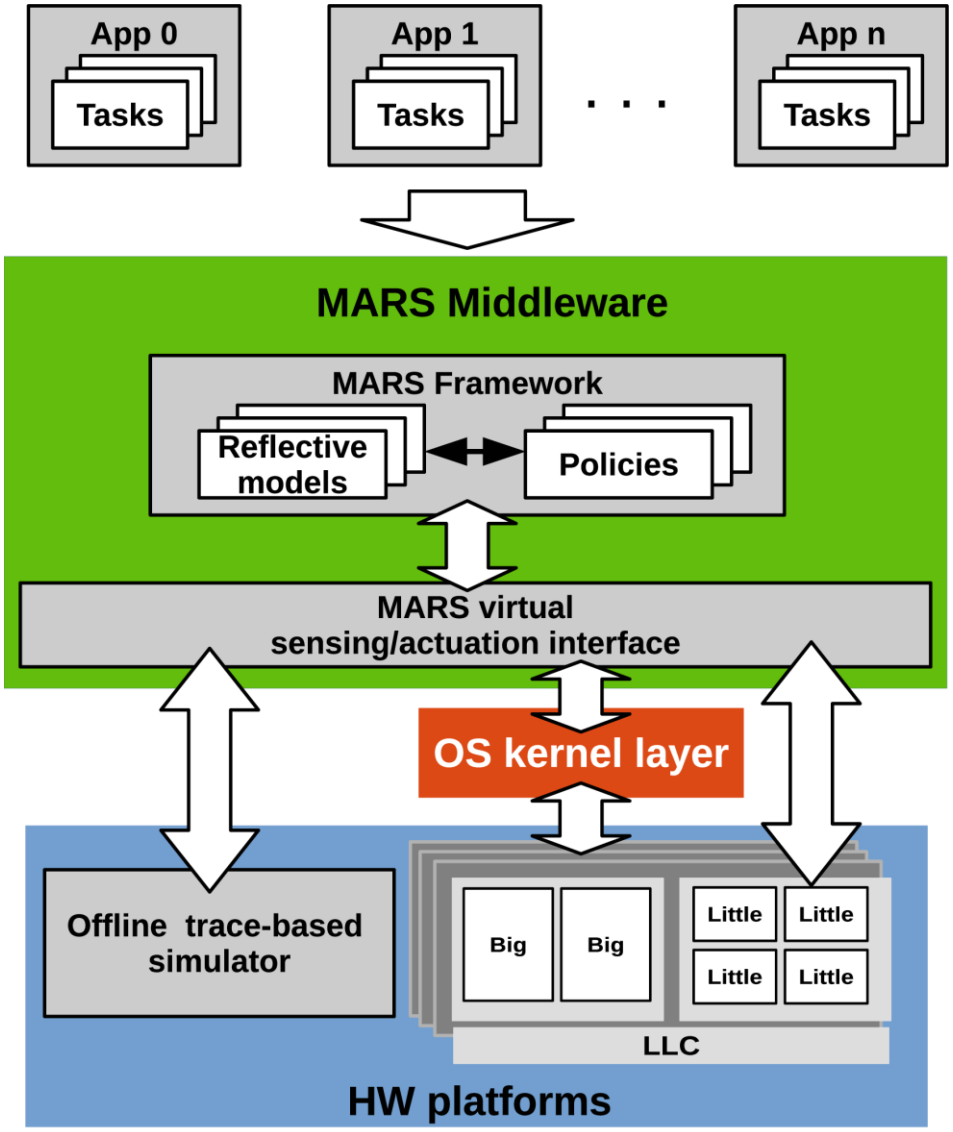
What if the mapping changes ?

Decision based on predicted result

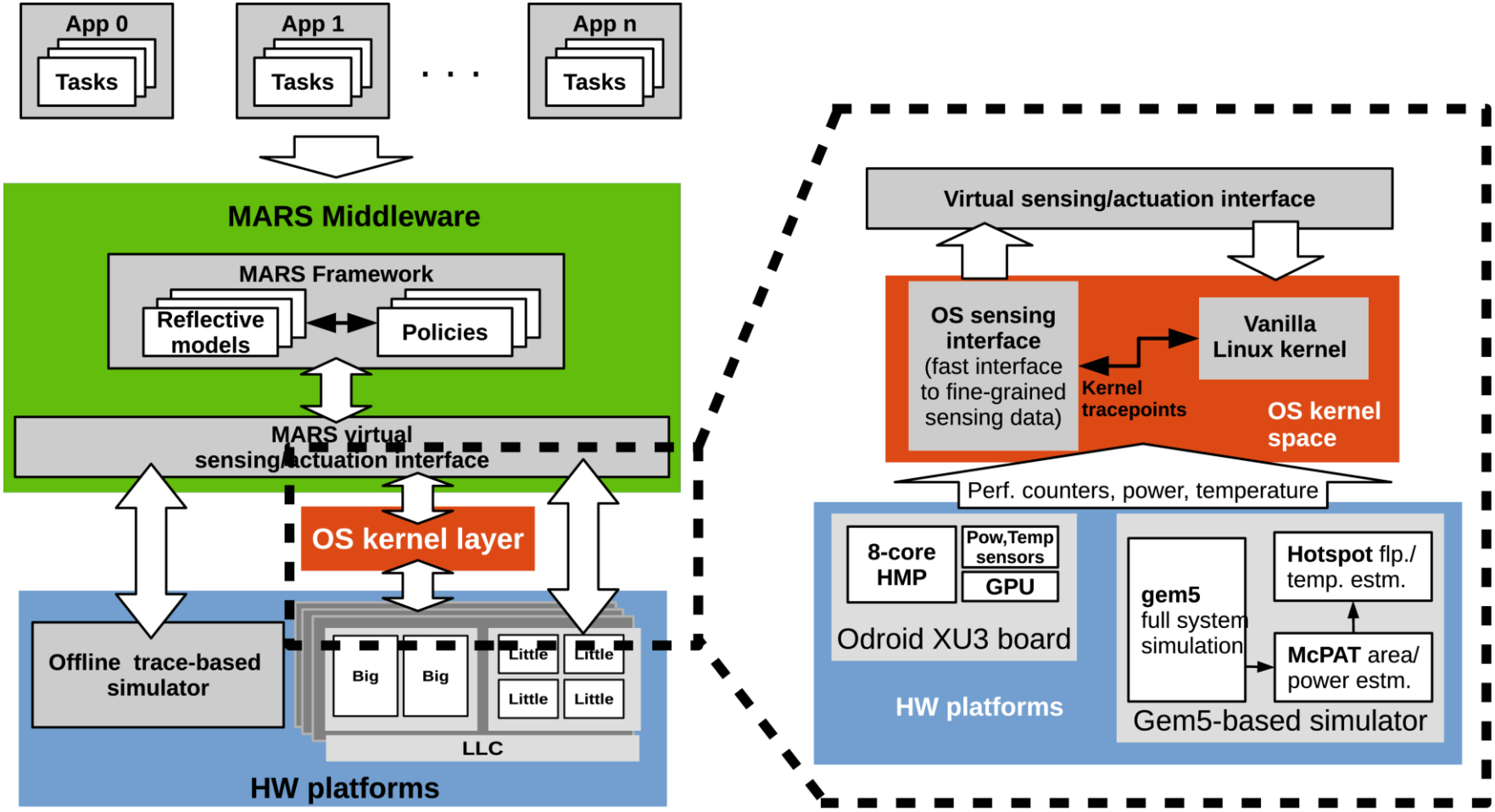
Reflection process



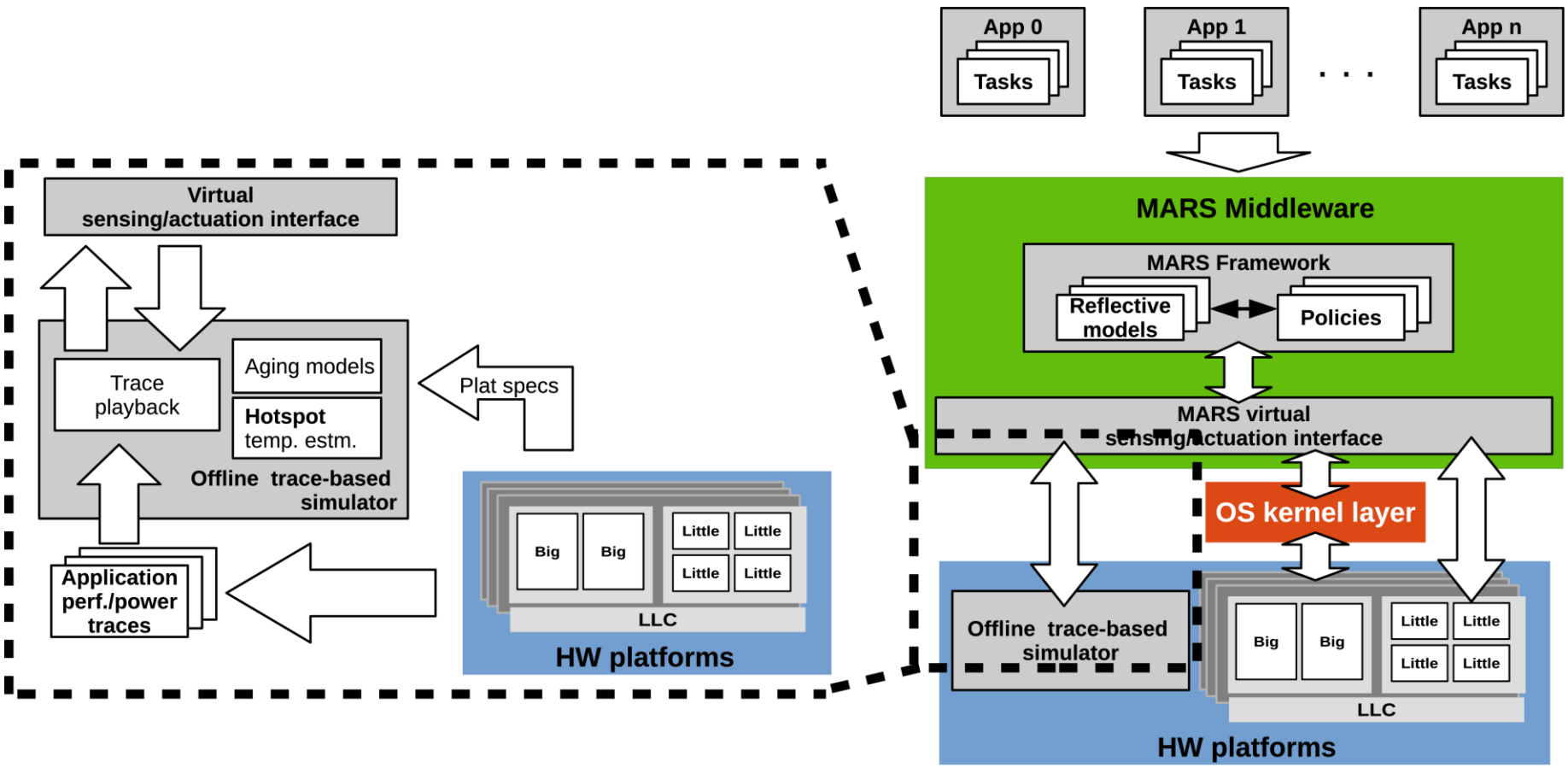
MARS deployment



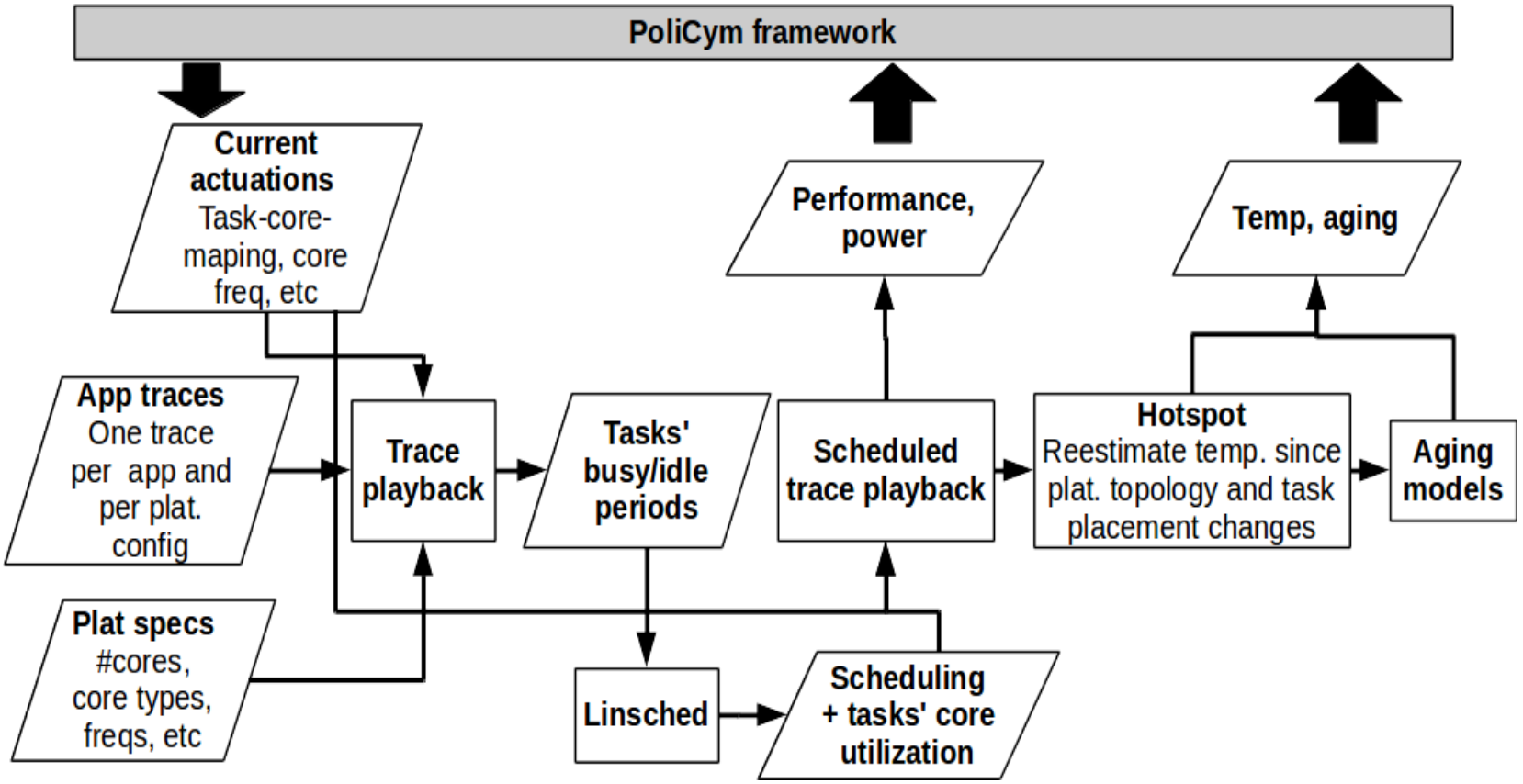
Implementation on Linux



Offline implementation



Offline simulation flow



Case studies

- **Energy efficient task mapping for HMPs**
- Managing chip wear-out on mobile platforms
- Design space exploration of HMPs

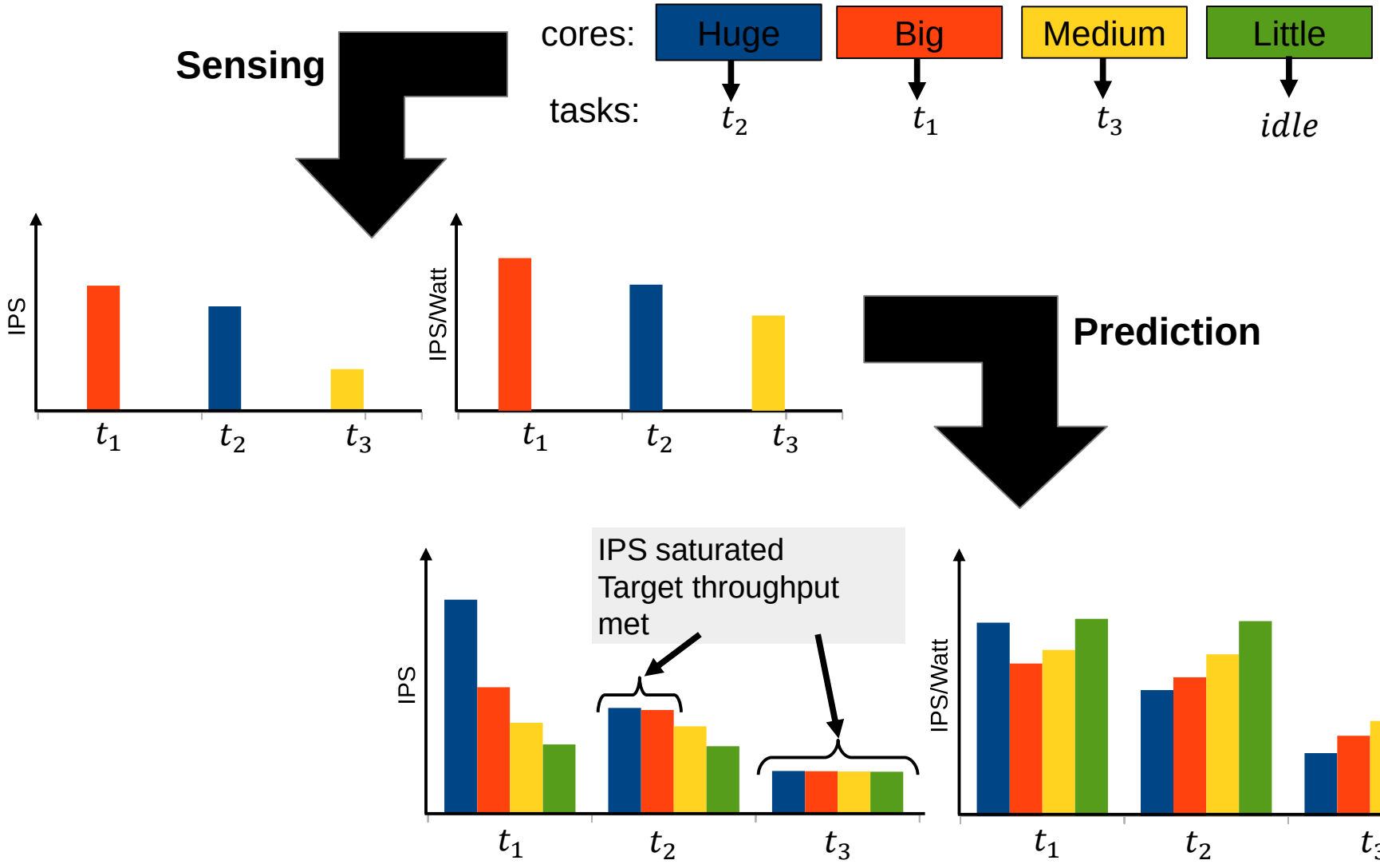
Case study: energy efficient task mapping

- **SPARTA**: Runtime Task Allocation for Energy Efficient Heterogeneous Many-cores
  - Light-weight QoS-aware task mapping
    - maximize the energy efficiency given performance constraints
  - Orthogonal to other policies
    - Performance/Power predictive models take in to account the behavior of the Linux's **DVFS governor** and **CFS policy**.

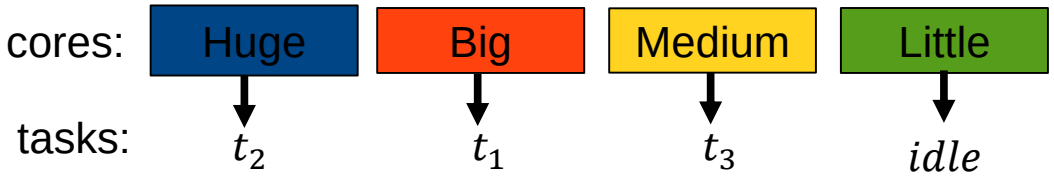
Muck, T., Sarma, S., Dutt, N. (2015). **Run-DMC: Runtime dynamic heterogeneous multicore performance and power estimation for energy efficiency**. CODES+ISSS '15

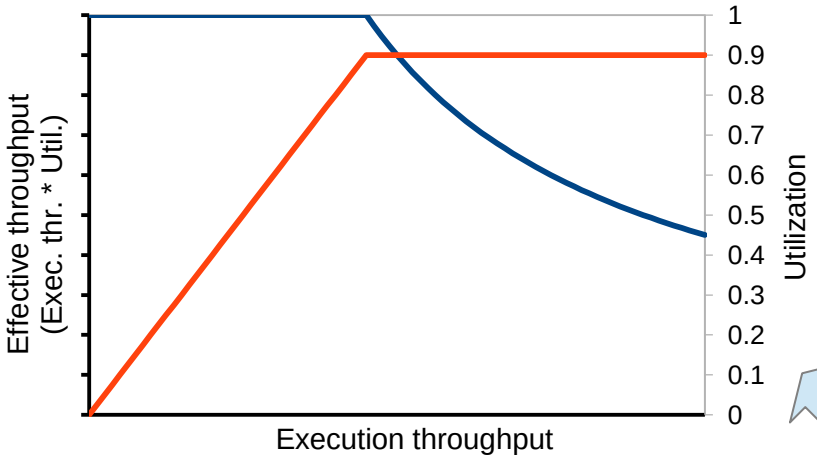
Donyanavard, B., Mück, T., Sarma, S., Dutt, N. (2016). **SPARTA: Runtime Task Allocation for Energy Efficient Heterogeneous Many-cores**. CODES+ISSS '16

SPARTA Overview

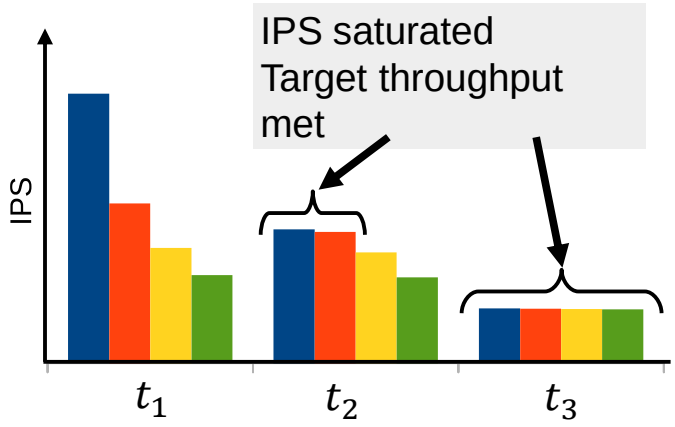


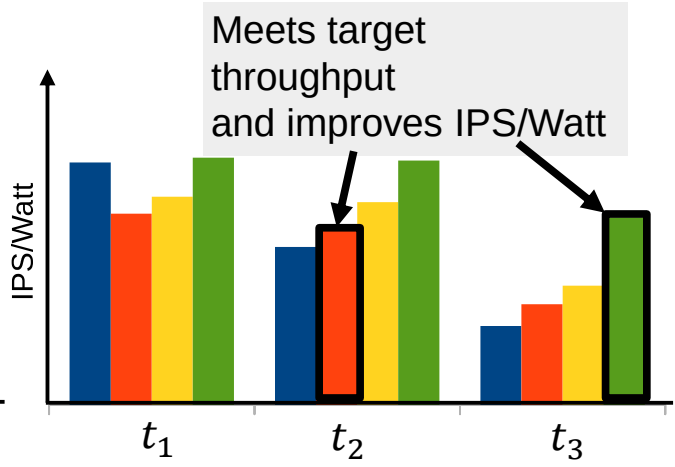
SPARTA Overview



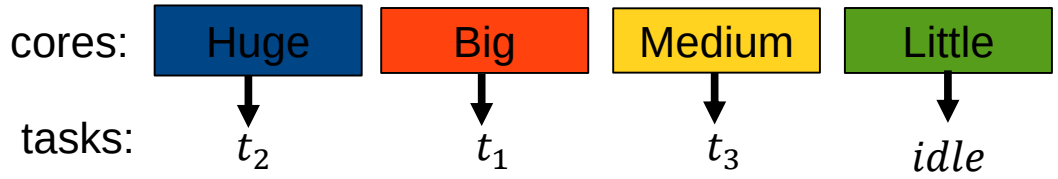


- Effective throughput eventually saturated unless the task is 100% CPU bound
- Opportunity to move task to a slower core and save energy

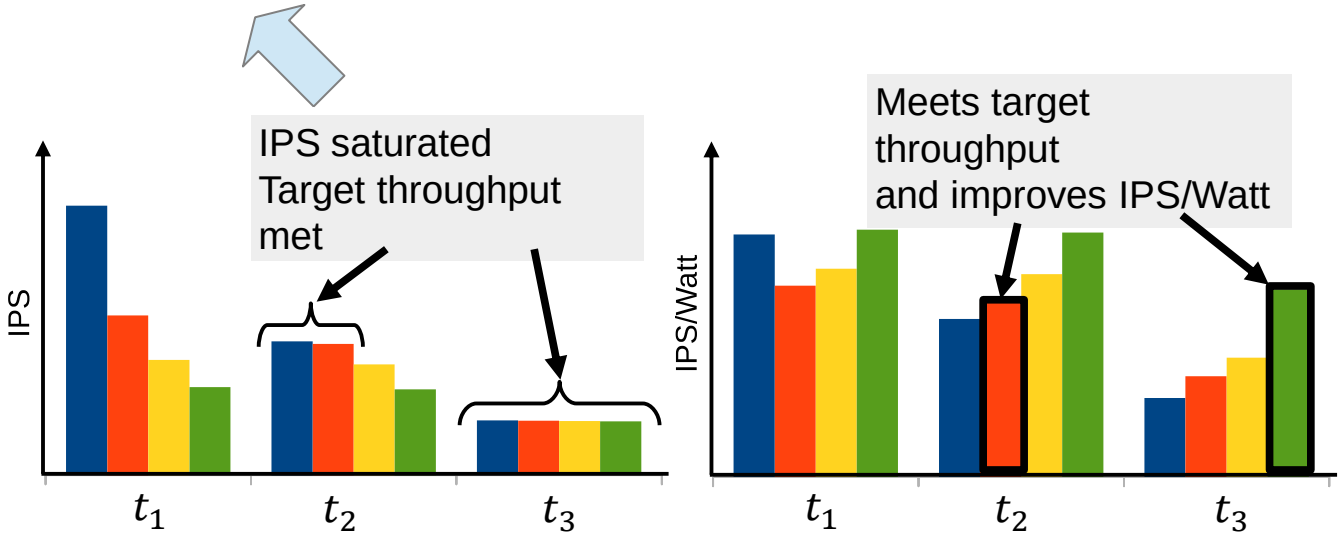




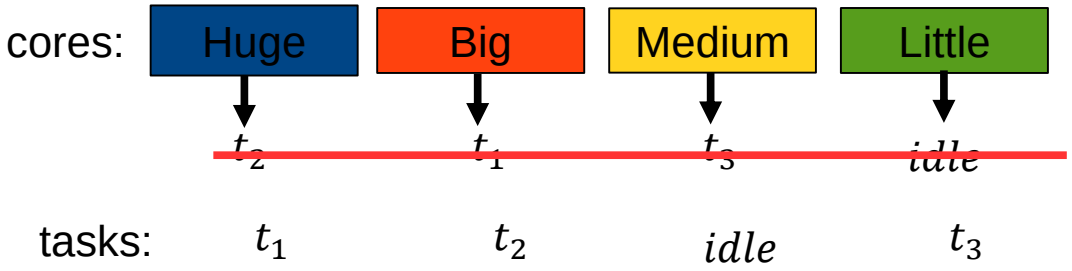
SPARTA Overview

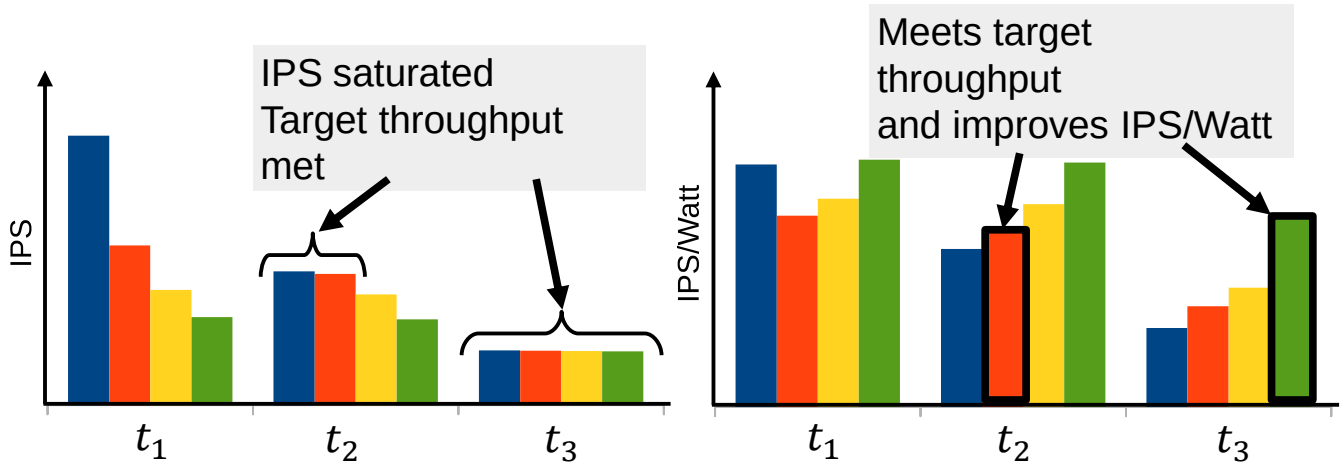


- Due to workloads not being 100% cpu bound (i.e. interactive apps)
- Memory latency or throughput bound workloads
- Opportunity to move task to a slower core and save energy



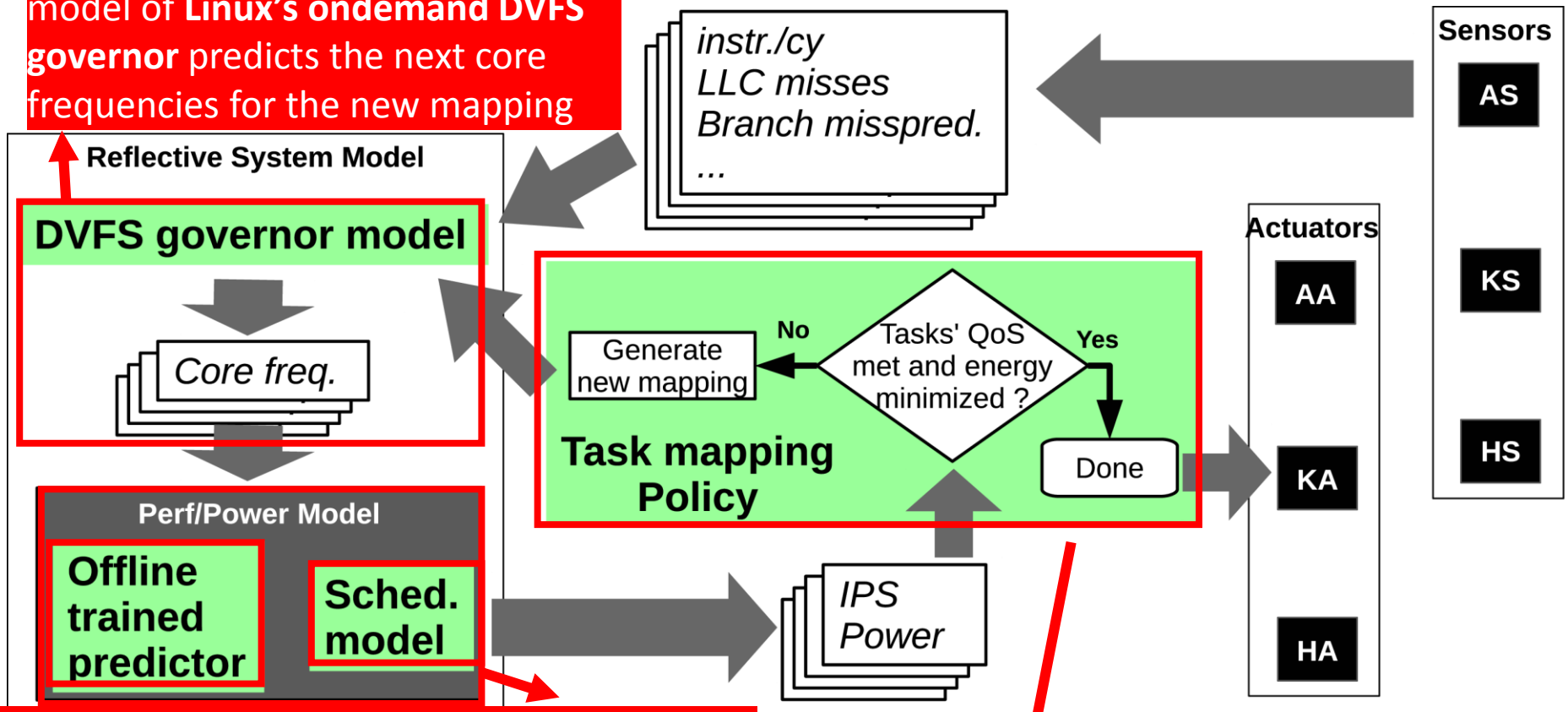
SPARTA Overview





SPARTA on MARS framework

When *tryActuate/senseIf* is used, a model of **Linux's ondemand DVFS governor** predicts the next core frequencies for the new mapping



Performance/Power model predicts IPS and power given new mapping and frequencies

- *Offline trained model predicts per-task IPS/Power*
- *Scheduler model predicts resource utilization*

List scheduling heuristic to explore new mappings

- **tryActuate/senseIf** to find which cores meet QoS with highest energy efficiency
- Map tasks in order of energy efficiency