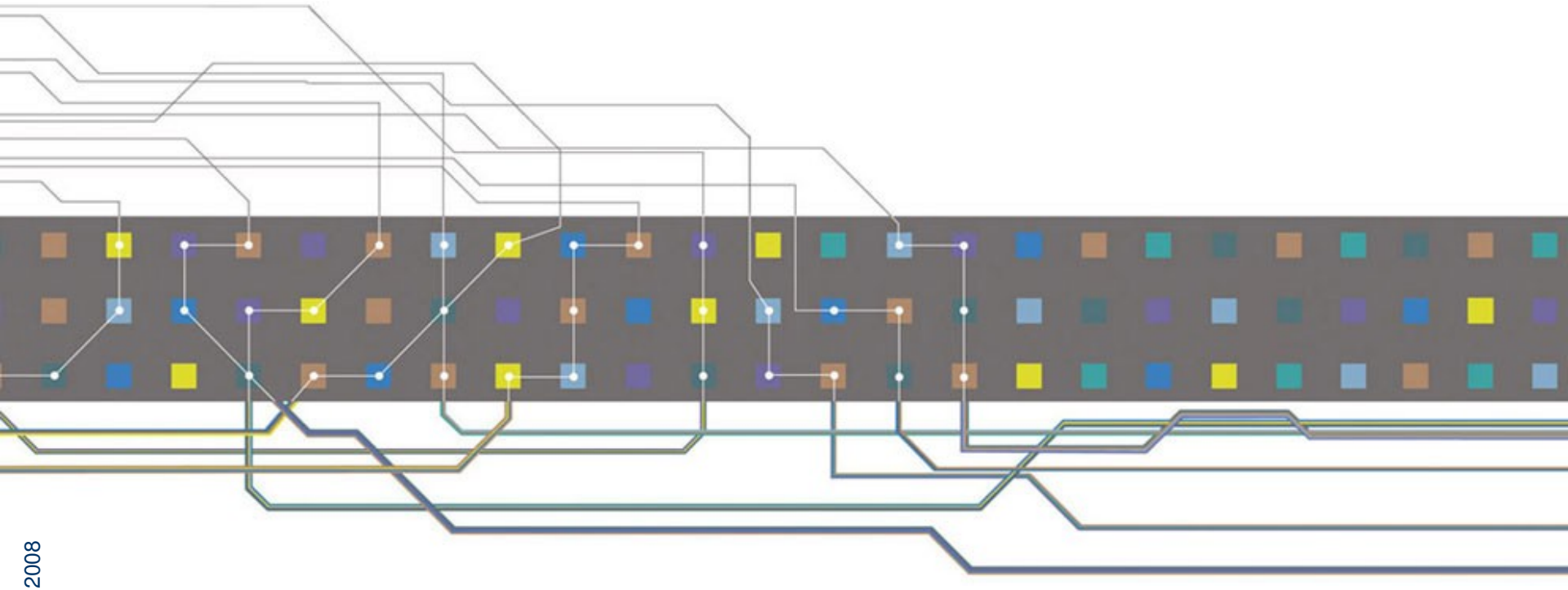
THALES

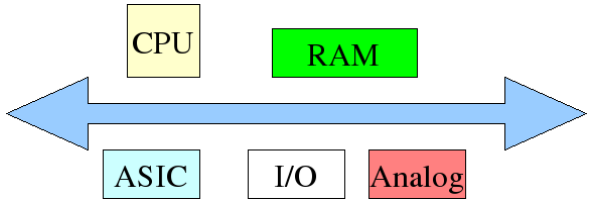


MPSoC Architecture low-power optimisation under Linux

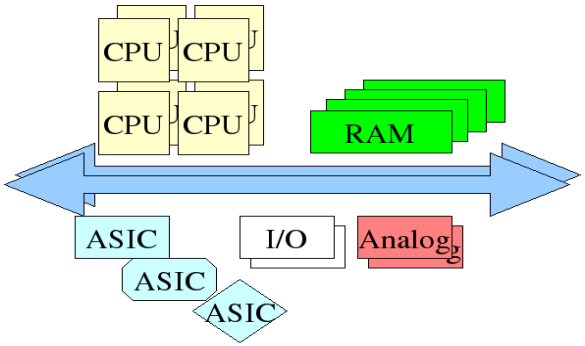
Dominique Ragot - THALES

Power-awareness in MPSoC

- Many hardware capabilities available at component level (CPU, RAM, peripherals, dedicated HW units)
  - Sleep/hibernation/activity modes
  - DVFS & DPCT
    - Stepwise
    - Continuous

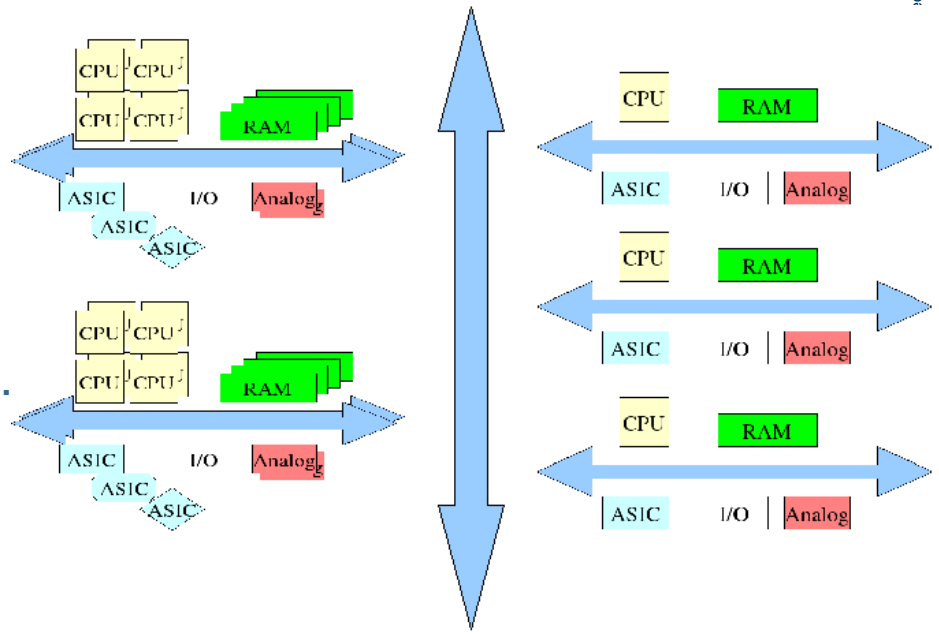


- How take advantage of them for applications on MPSoC ?
  - System level / component level
    - Power-consumption models
  - Heterogeneity
    - HW capabilities vs power
    - power modes of components
  - Dynamicity
    - Load balancing
    - Time constraints

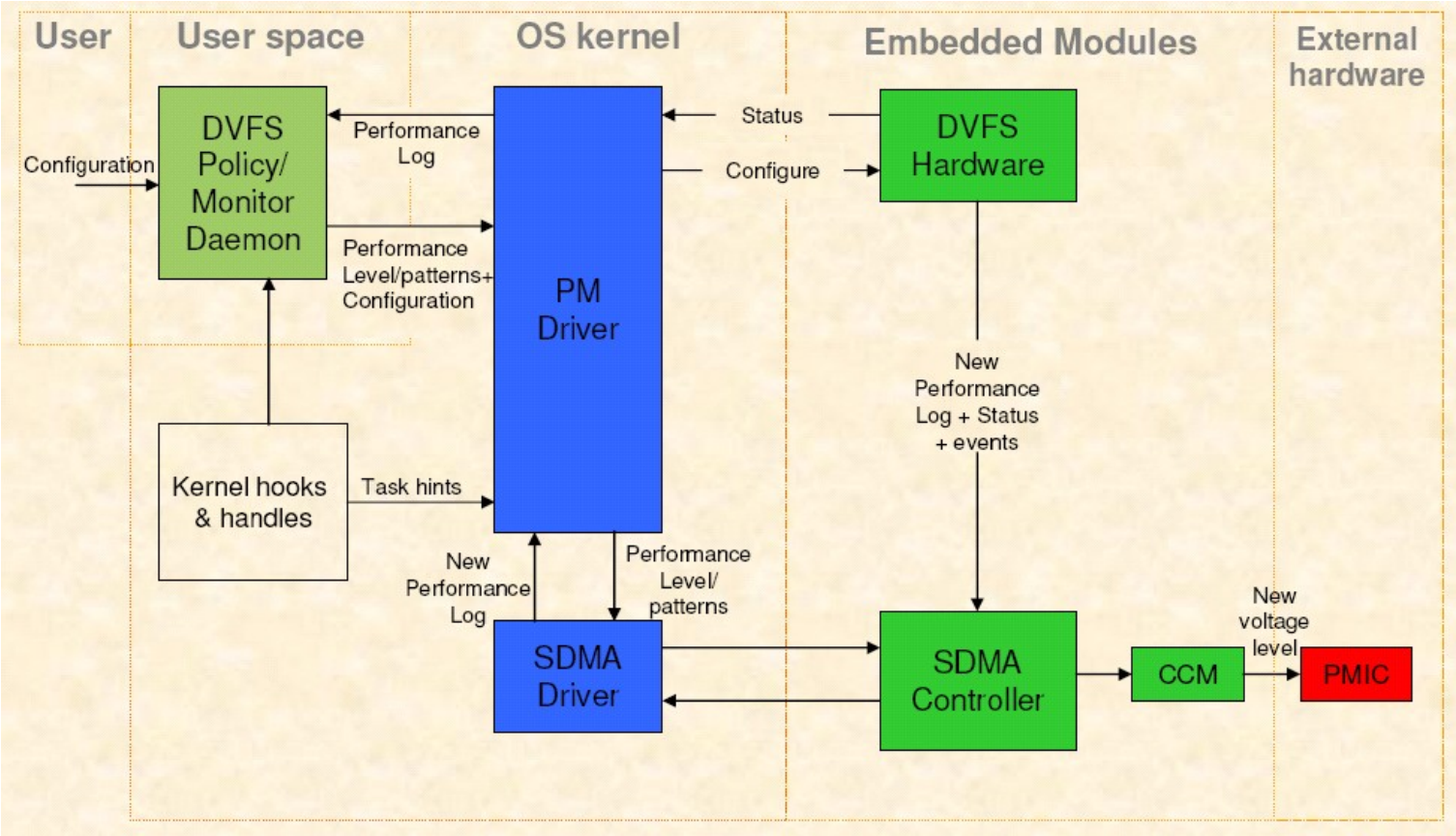


Using Linux in MPSoC

- More and more an embedded OS
  - Many initiatives: CELF, mobile Linux, Limo …
  - Footprint is no longer THE issue
    - Versatility vs. Applications and services
    - Evolutivity within product lines
    - Overall cost
- Many examples in industry
  - SoC-based products
    - Smart phones, Set-top boxes, ...
  - MPSoC evaluation boards
    - PowerPC, ARM

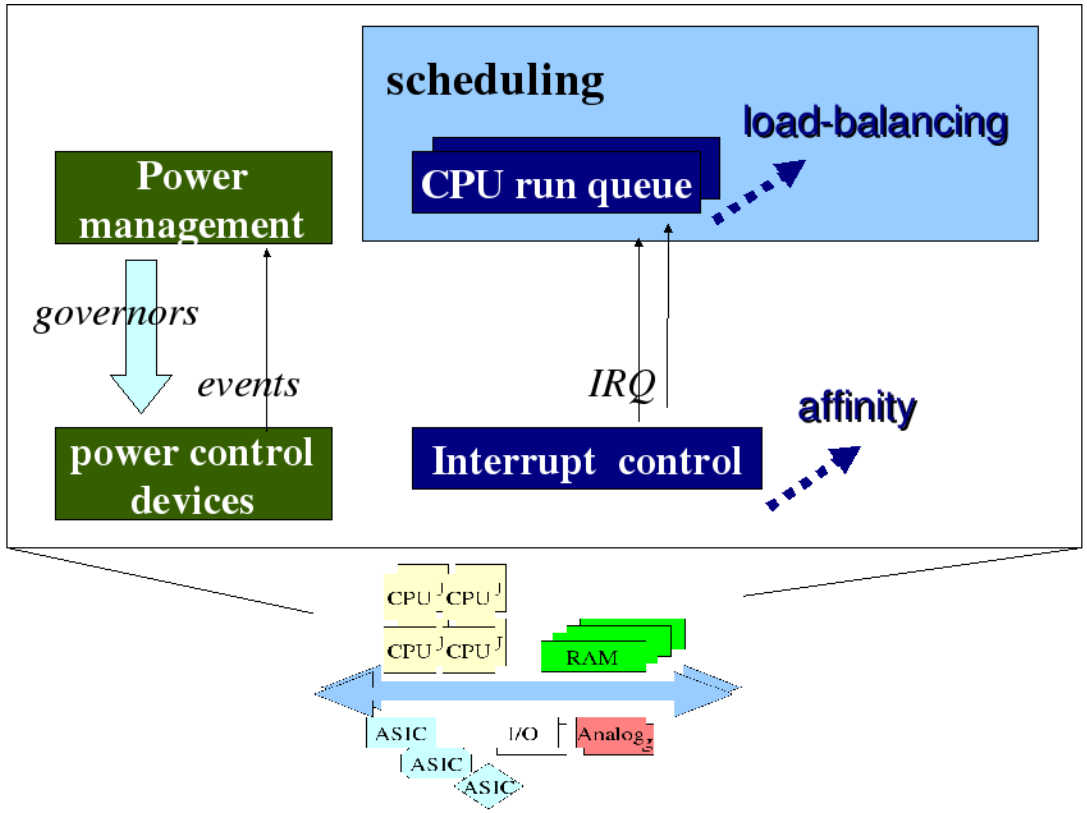


A Typical Hardware Performance Monitor Architecture



Linux and power consumption: actual techniques

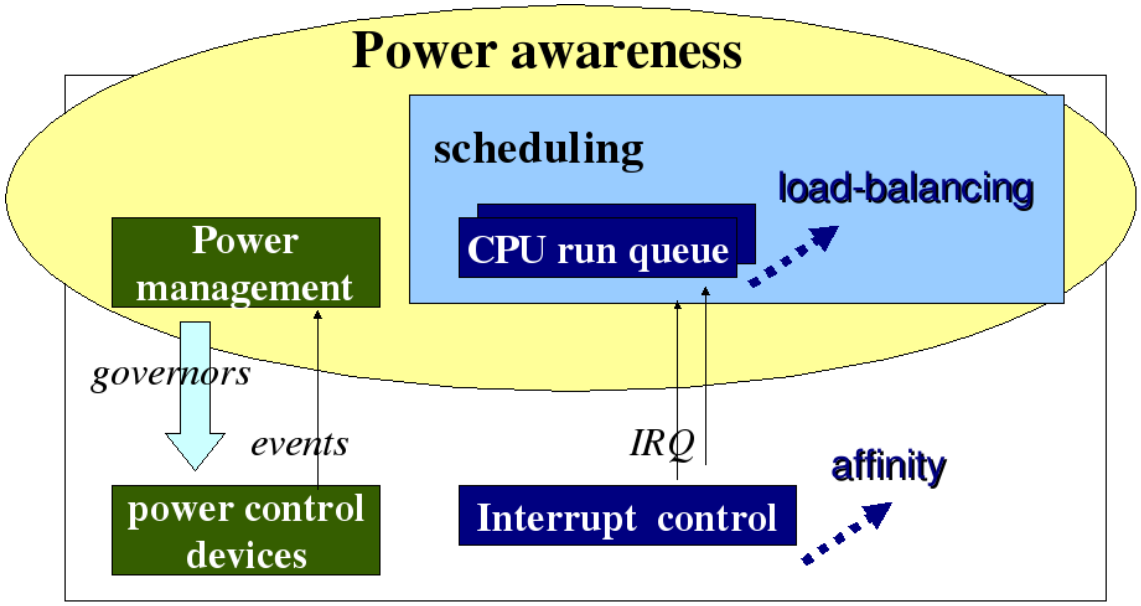
- Existing Technologies
  - Well-known APM and ACPI
  - Low-level DVFS/DPCT drivers
- Limitations
  - No real power-aware scheduling yet
  - No MPSoC power-aware dynamicity management



Power-awareness: impact on scheduling

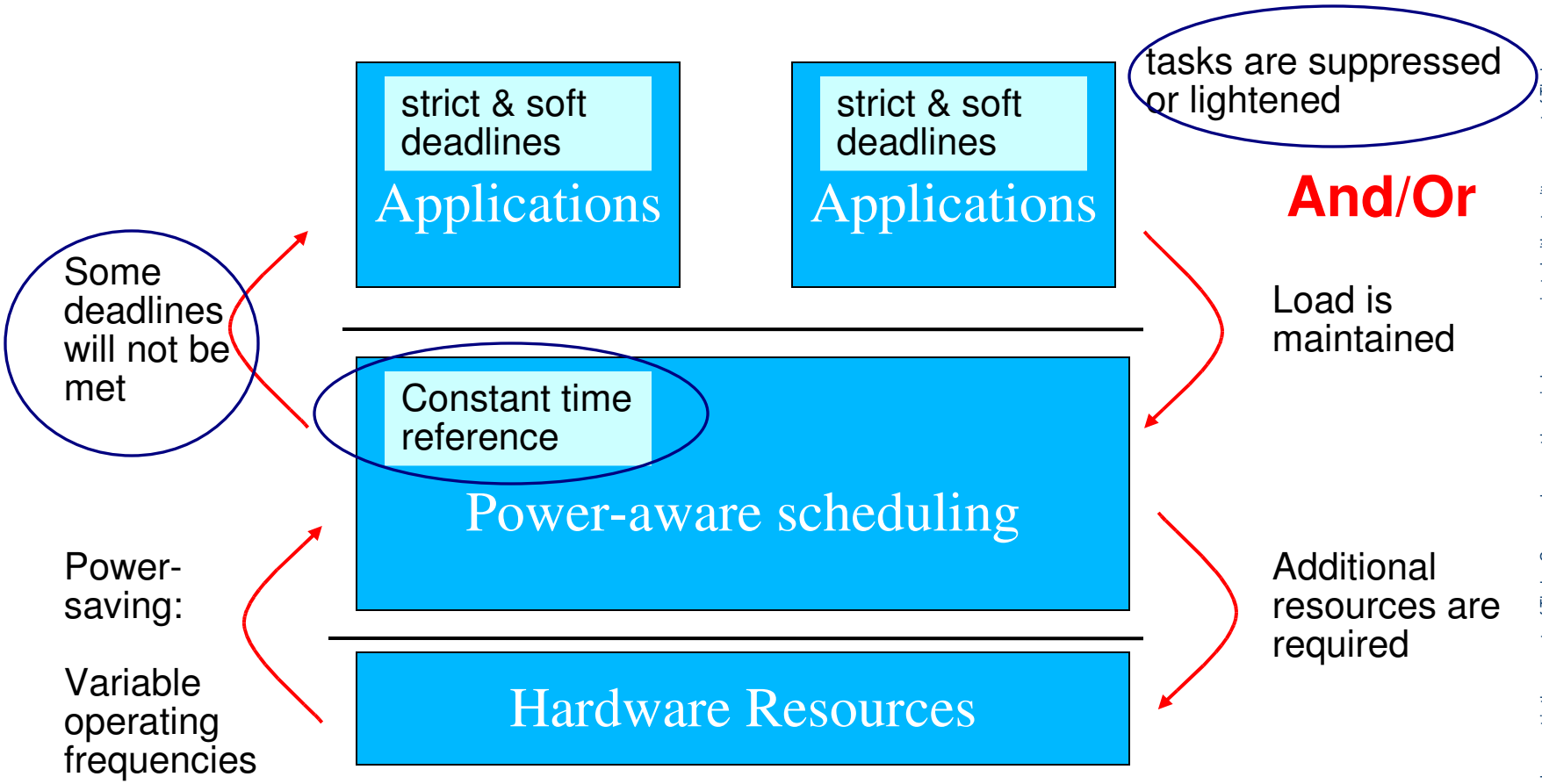
- Operating frequencies of components may change
  - Bus
  - CPU
  - Hardware IPs
  - RAM
  - Peripherals
- Need to support heterogeneity and dynamicity
  - Provide Application-level control over dynamicity
    - Adapt the load to available resources (power-driven)
    - Adapt the resources to the load (performance-driven)
- Real-time additional constraints
  - Maintain a persistent time reference
    - Insensitive to frequency scaling

Linux and power consumption: trends



- New techniques, useful for power-awareness:
  - Generalized hotplug
  - High-resolution timers
  - CPU groups scheduling (even in RT)
    - Resource control
  - Modular scheduler infrastructure
    - Power-aware scheduling
  - RT additional capabilities
    - In-kernel and exo-kernel
  - Virtualization

Example of interactions

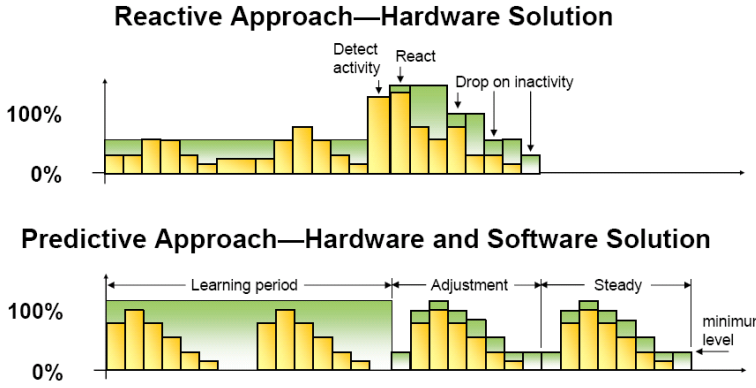


Power-aware scheduling: load analysis

- Maximize resources usage + minimize power (+ honor deadlines)
  - User controlled: load tracking module is disabled
  - Hardware Controlled: Load tracking and sets performance level according to predefined user parameters
  - Hardware supported load analysis: DVFS monitor returns a buffer to the system, which exploits it into policies
  - Hardware controlled with kernel hooks

Reactive Approach: controlled by hardware

Predictive Approach: load analysis



  - Load analysis via Kernel hooks: rely on OS performance analysis
  - Task specific power control: Task gives requirements to the OS
  - Combination of the previous modes

Power-aware scheduling: load-balancing

- Load balancing can be an answer to power-awareness
  - Despite the adverse effect: migration cost
    - has to be quite well known (requires prediction)
  - Many nodes in low freq modes are be more power-efficient than 1 node at highest frequency
    - Data processing has to be parallelizable
    - Memory and cache subsystem must be suited to this
      - Low Contention and/or fast, inexpensive data replication
- Strategies for load-balancing
  - Which nodes to activate/suspend ?
  - What time frame is realistic ?
  - Impact on system behaviour
    - Testability and reproductibility
    - Overall stability
- Real-time
  - Same questions apply with respect to interrupt dispatching

Thank you for your attention

Any questions ?