1) The classic Vestal model

- In each mode, tasks of a certain criticality or higher execute.
- Different WCET estimates for the same task in different modes.
  - By techniques with corresponding confidence levels.
- When a task would overrun its WCET estimate for that mode, a mode change occurs (e.g., $L \rightarrow H$, with two modes).
- Essentially, the processor resources intended for the $L$-tasks are repurposed for the $H$-tasks upon mode change.

2) Extending the principle to additional resource types

- **Idea**: Reclaim more kinds of resources at mode change!
- Explored by our paper at the ECRTS 2017 main track, for the shared last-level cache of a multicore.
  - Per-task cache partitions reconfigured at mode change.
  - This work-in-progress explores a more refined arrangement:
    - Partial use of $H$-task partitions for locked hot pages.
    - The rest of the cache partitions is populated dynamically (e.g., LRU).
- WCET estimates become functions of:
  - Analysis technique for each mode ($L$, $H$);
  - Size ($\sigma$) of task partition for locked hottest pages;
  - Size ($\pi$) of task partition used dynamically.

3) Illustration of the arrangement and reconfiguration

4) Lockdown vs dynamic use

- Locking of hot pages in cache: more predictable task execution.
- Dynamic cache partition use: *might* lower the *actual* WCET but makes analysis more complicated – possibly requiring pessimistic simplifying assumptions.
- Different tradeoffs for static WCET analysis ($H$-mode estimates) vs probabilistic measurement-based ($L$-mode estimates).
- No page locking in $L$-task cache partitions, in order to minimise reconfiguration overheads at mode change.

5) Some challenges

- Accurate and tractable parametric WCET estimation for many points ($\sigma, \pi$) per task in the design space.
- Identification of good heuristics for partitioning the cache in the two modes.
- Estimation of reconfiguration overhead from the task partition parameters and incorporation to schedulability test for EDF with deadline-scaling.