Current Solutions and Open Challenges

Presenter: Bjorn Andersson

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8702-15-D-0002 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

DM18-0756

Outline

- Mechanisms
  - Mechanisms for a single resource
    - Cache coloring
    - Bank coloring
    - Memory bus guard
    - Cache locking
  - Mechanisms for more than a single resource
    - Coordinated cache and bank coloring

- Analysis
  - Work-conserving bus
  - Analysis with bank sharing and more advanced model
  - Co-runners

- Comparing solutions

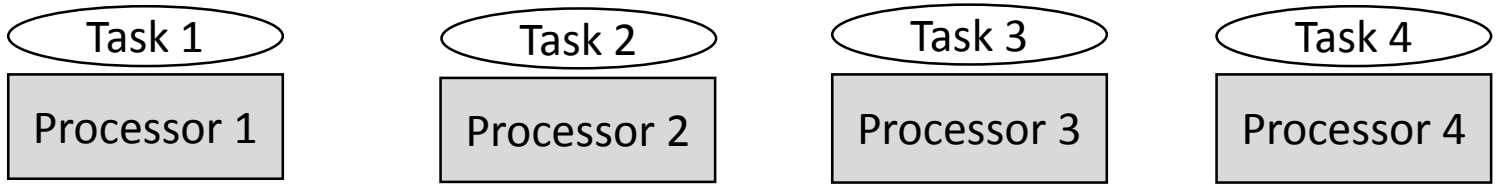
- Unsolved challenges

Mechanisms, Mechanisms for a single resource

Cache coloring

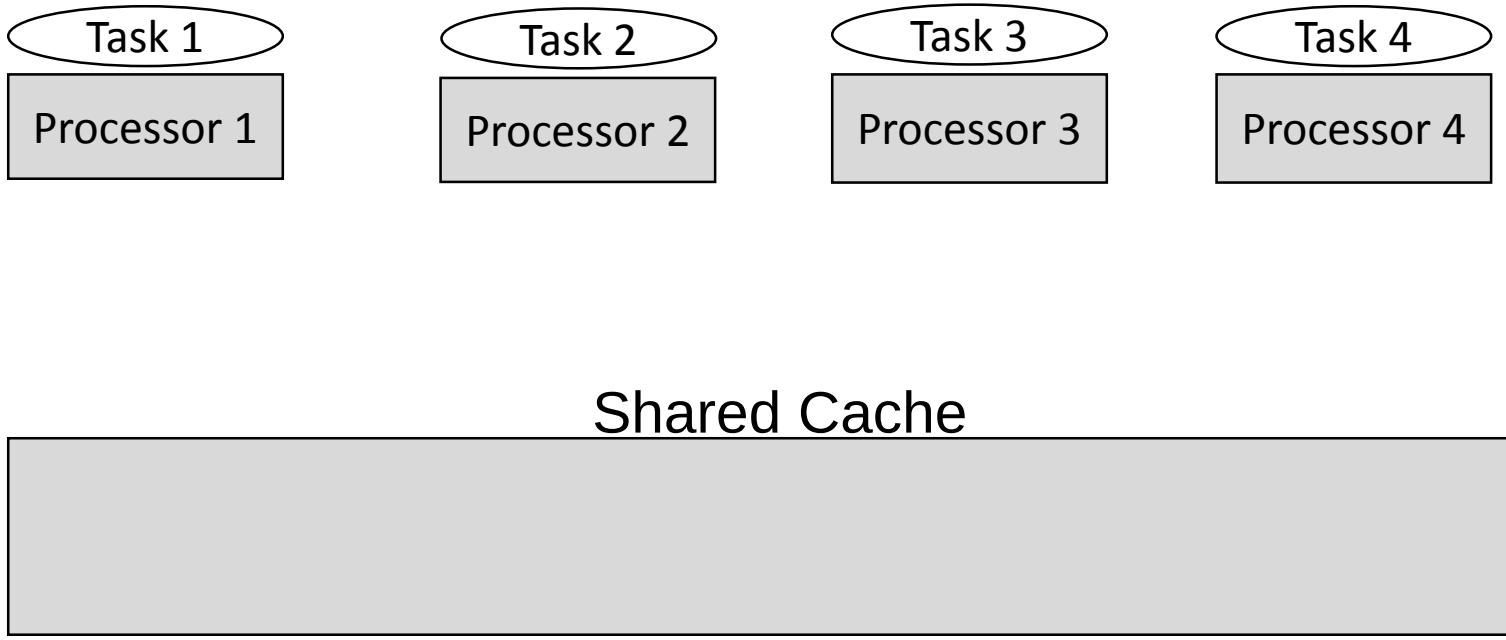
Mechanisms, Mechanisms for a single resource

Cache coloring



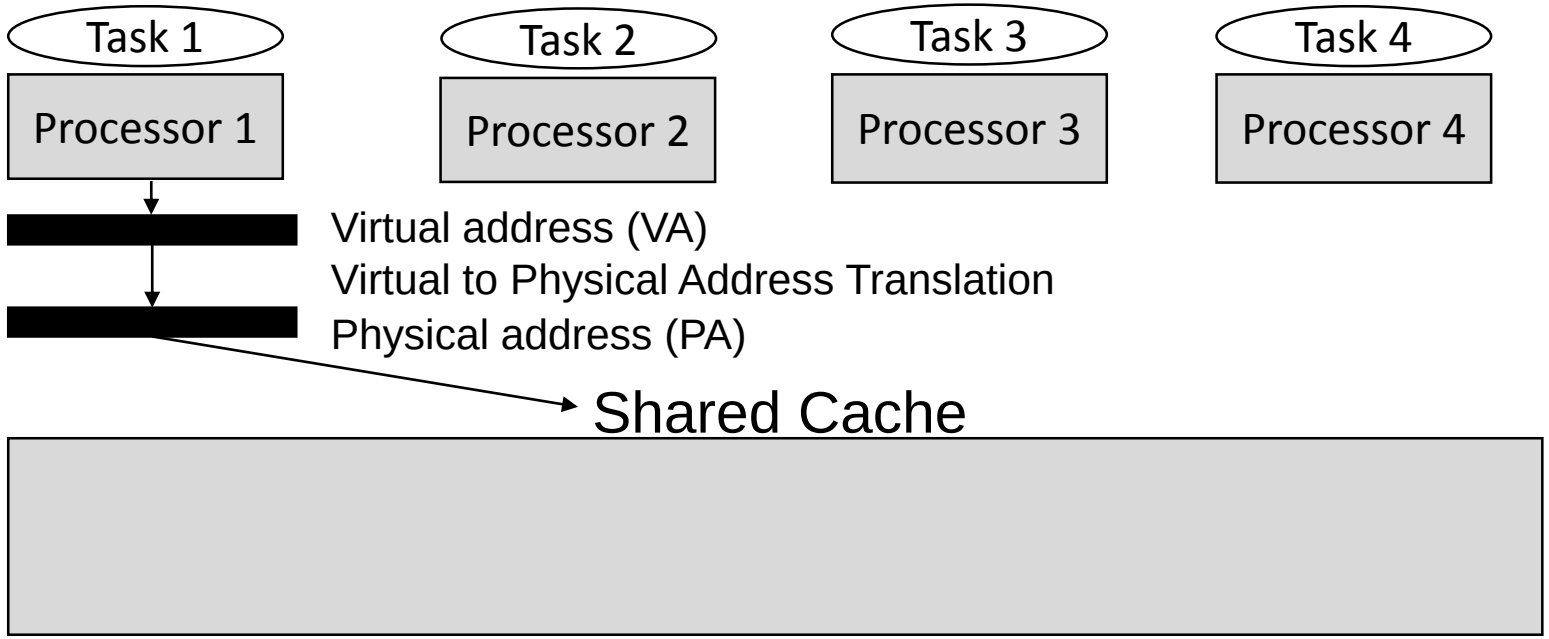
Mechanisms, Mechanisms for a single resource

Cache coloring



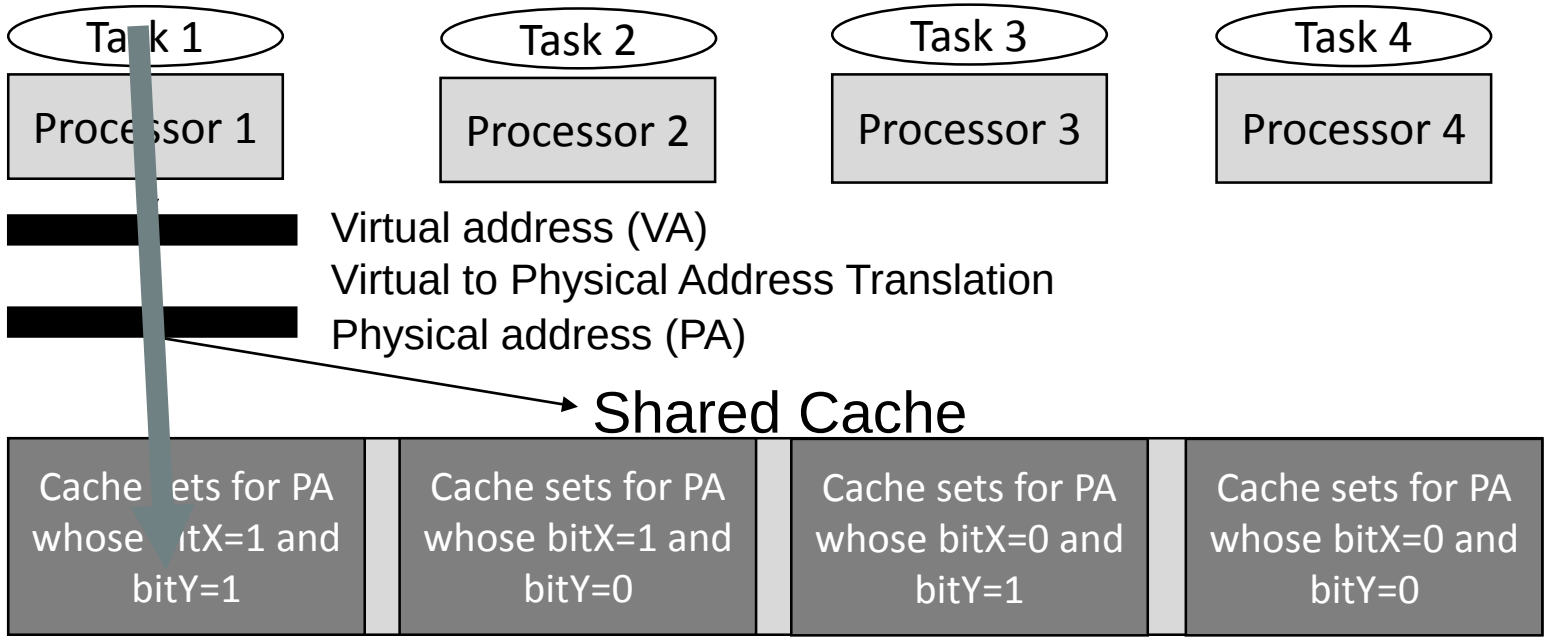
Mechanisms, Mechanisms for a single resource

Cache coloring



Mechanisms, Mechanisms for a single resource

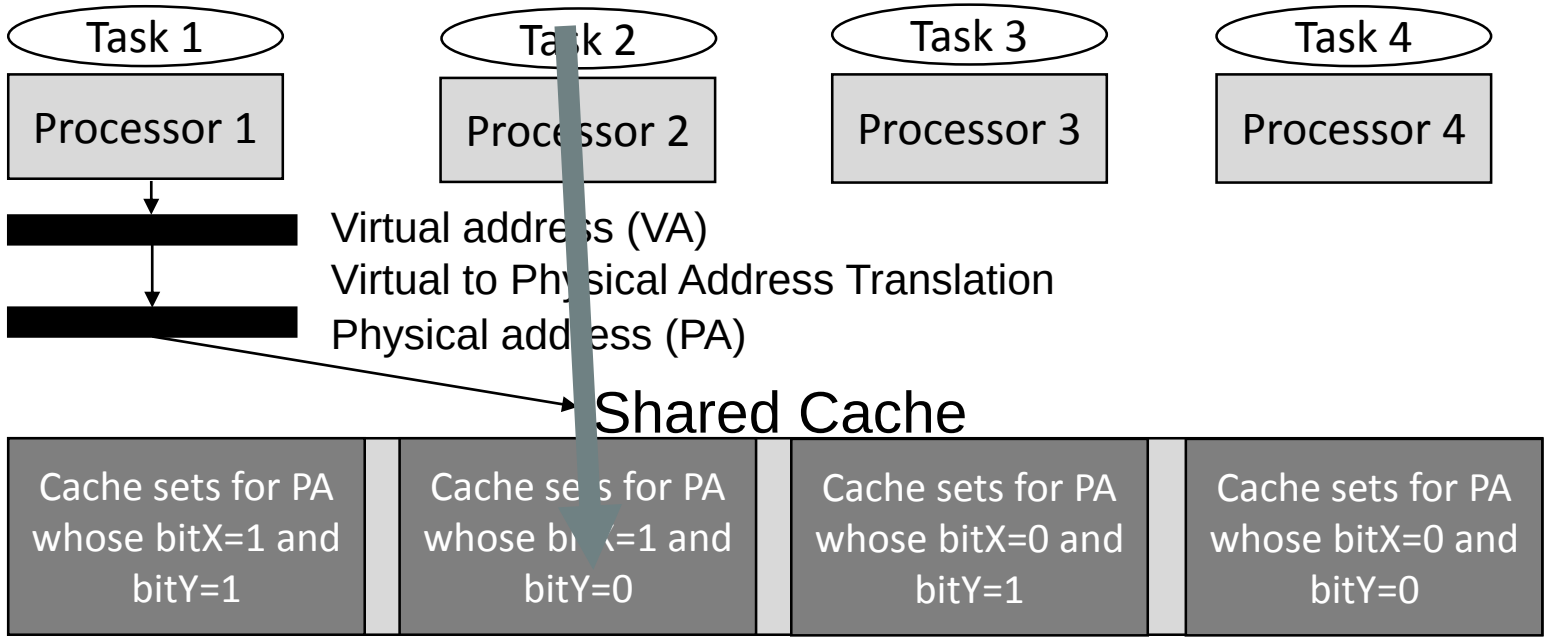
Cache coloring



Goal: Make sure that all memory accesses from task 1 go to leftmost cache sets and to the leftmost cache sets.

Mechanisms, Mechanisms for a single resource

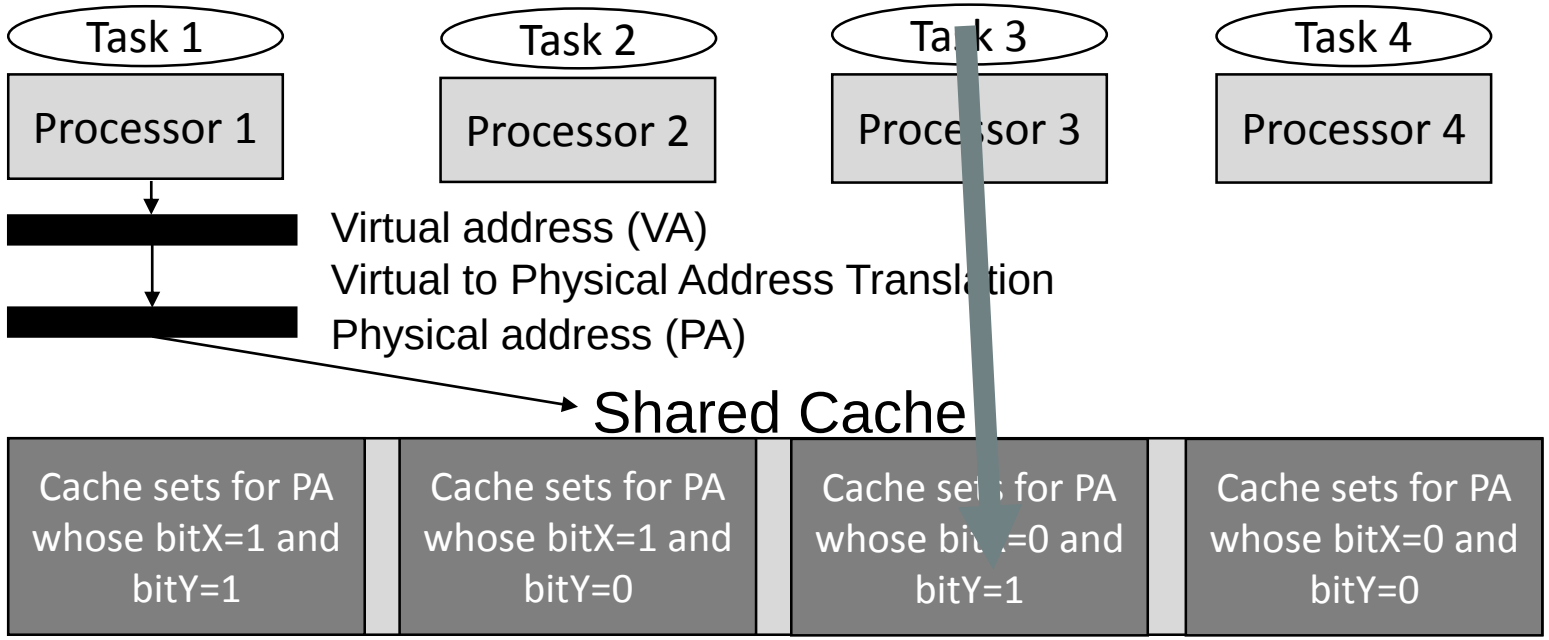
Cache coloring



Goal: Make sure that all memory accesses from task 2 go to leftmost cache sets and to 2nd leftmost cache sets.

Mechanisms, Mechanisms for a single resource

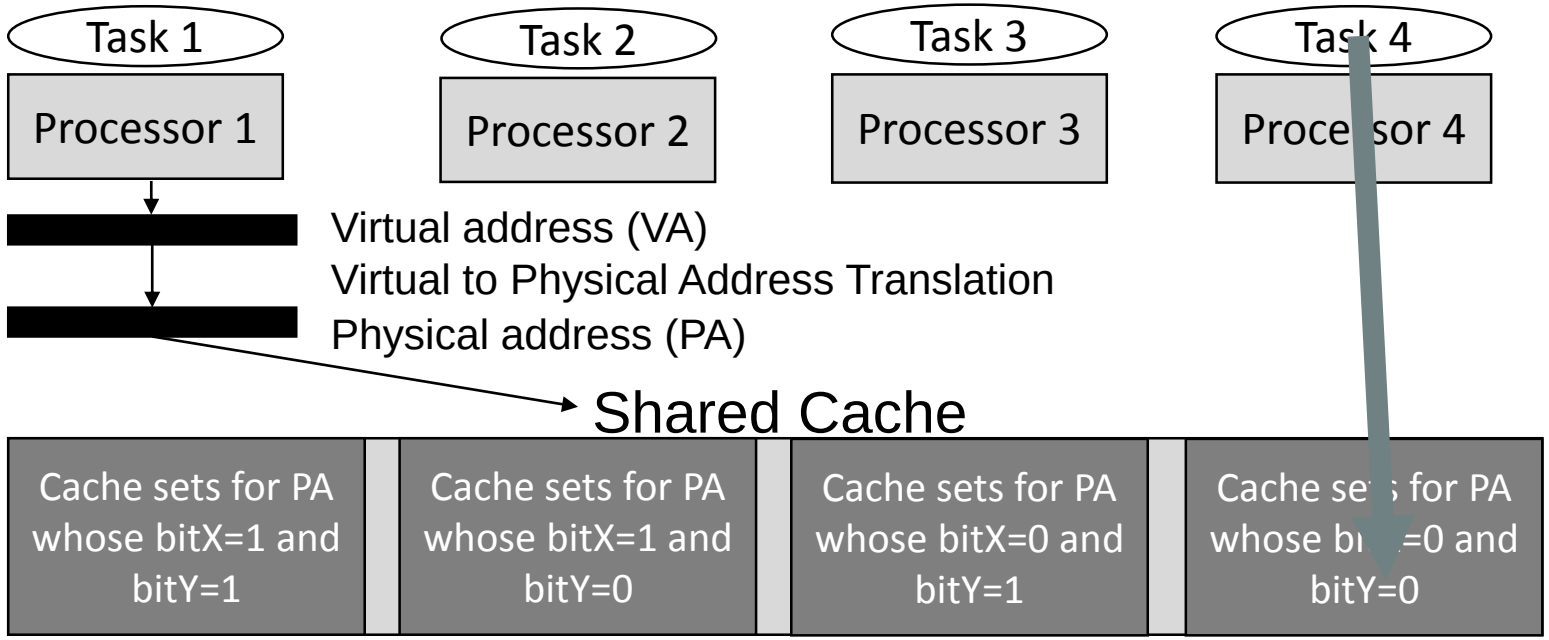
Cache coloring



Goal: Make sure that all memory accesses from task 3 go to leftmost cache sets and to 3rd leftmost cache sets.

Mechanisms, Mechanisms for a single resource

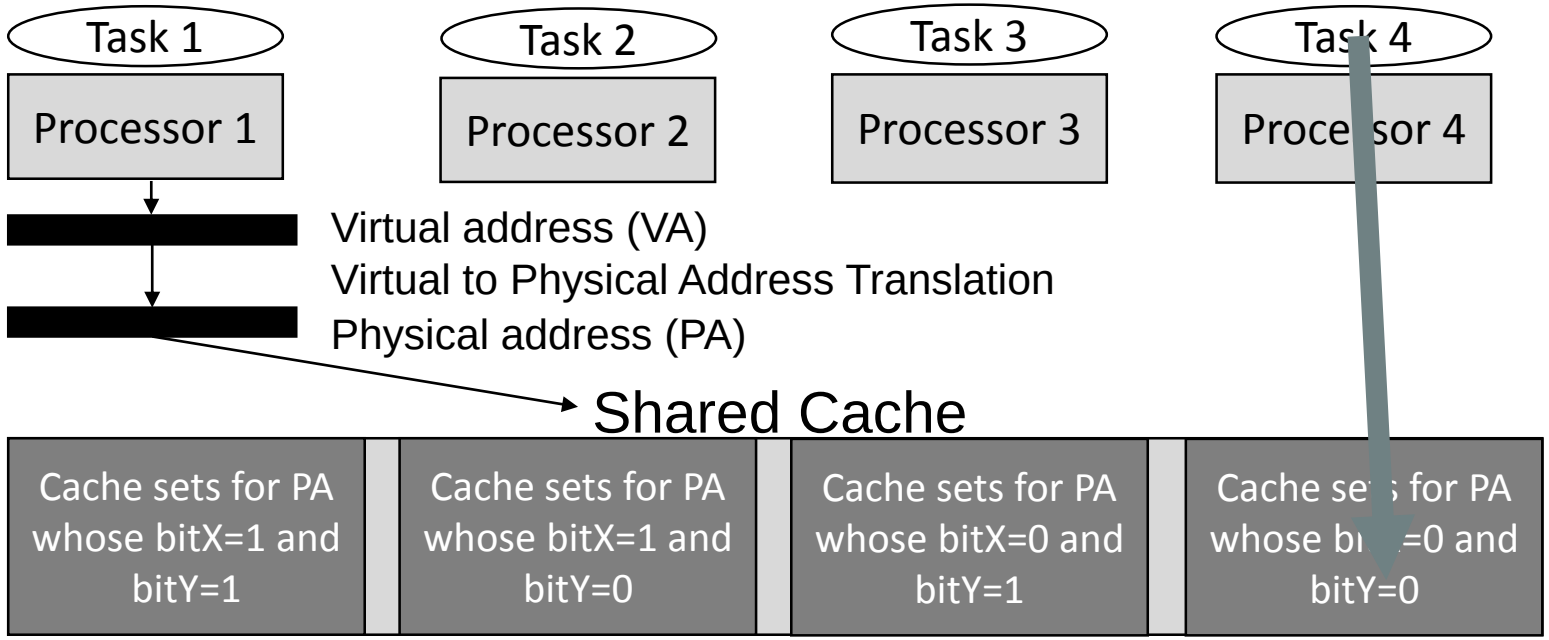
Cache coloring



Goal: Make sure that all memory accesses from task 4 go to leftmost cache sets and to the 4th leftmost cache sets.

Mechanisms, Mechanisms for a single resource

Cache coloring



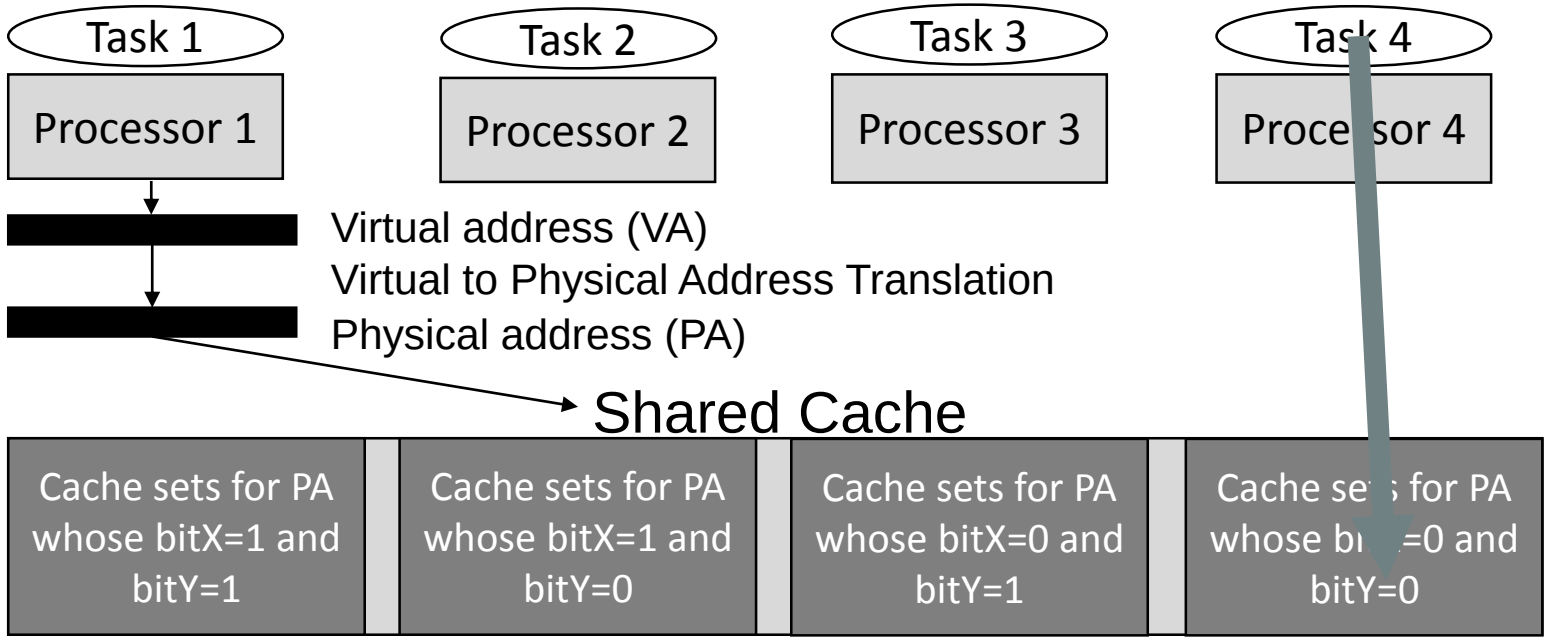
We can achieve these goals by setting up the virtual-to-physical translation mechanism. I.e., setting up the page table in the OS properly can be used to achieve cache isolation.

Mechanisms, Mechanisms for a single resource

Cache locking

Mechanisms, Mechanisms for a single resource

Cache locking

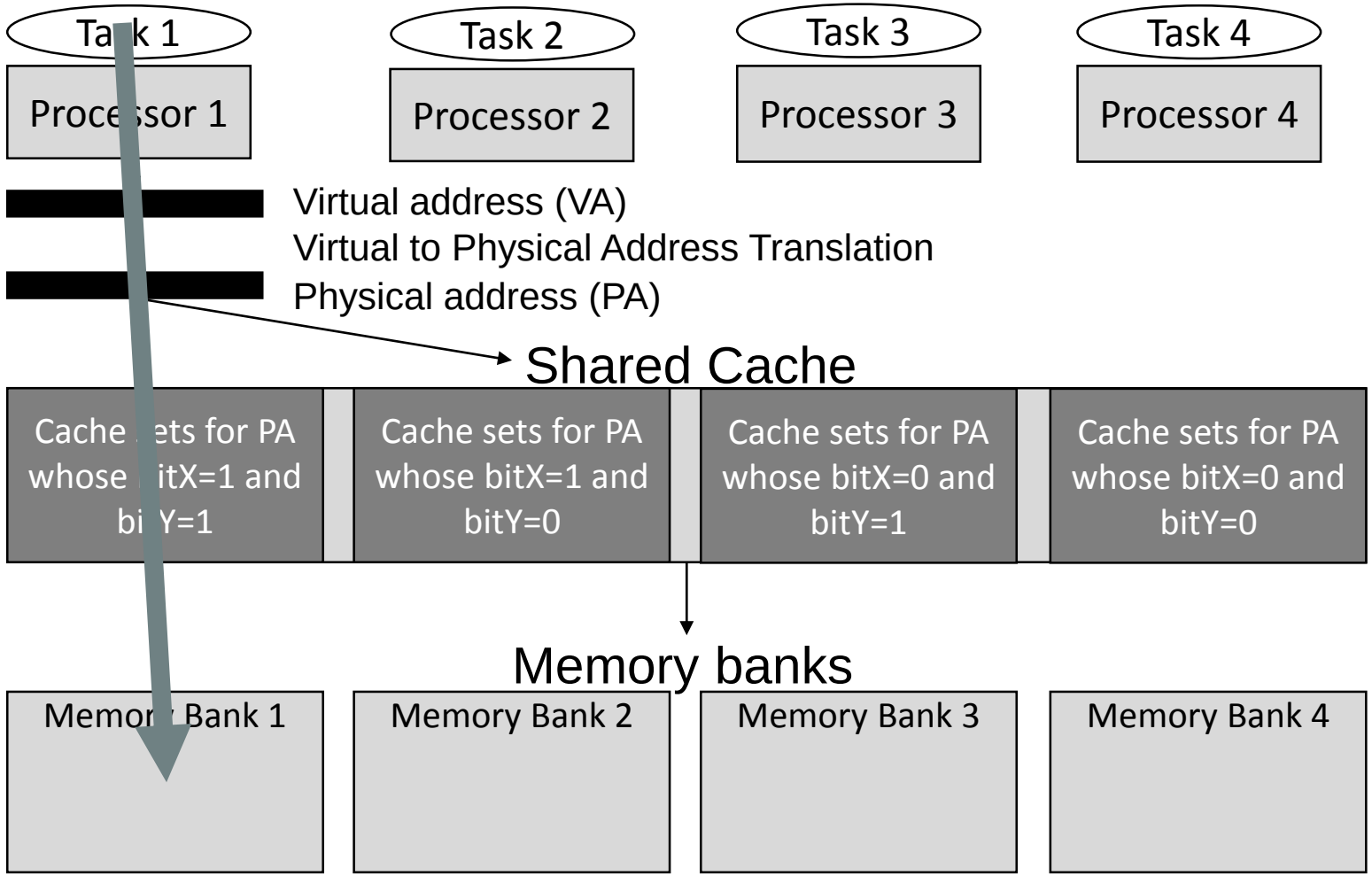


Execute a special instruction that ensures that cache blocks currently in the cache cannot be evicted.

Bank coloring

Mechanisms, Mechanisms for a single resource

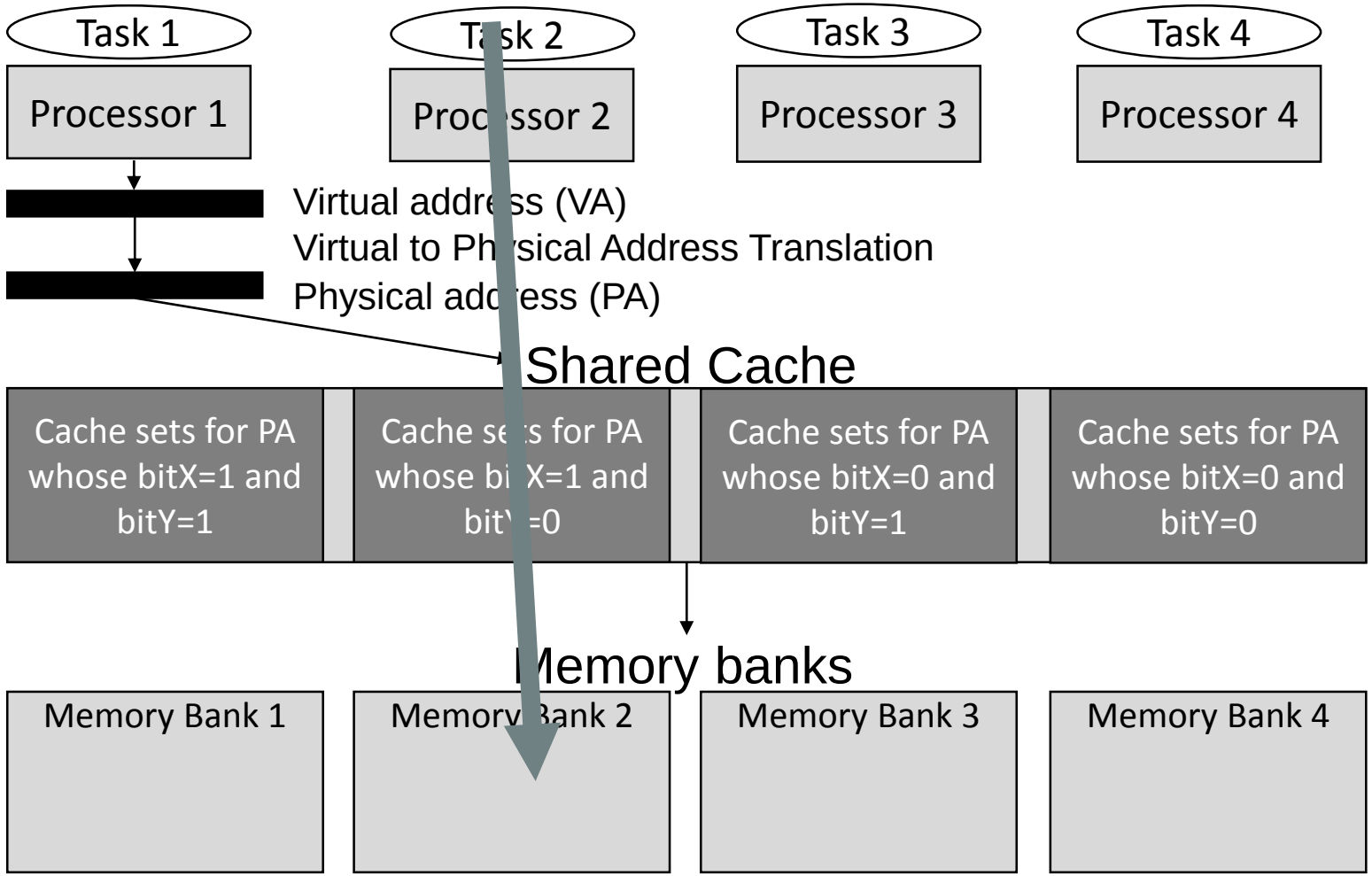
Bank coloring



Goal: Make sure that all memory accesses from task 1 go to leftmost cache sets and to memory bank 1.

Mechanisms, Mechanisms for a single resource

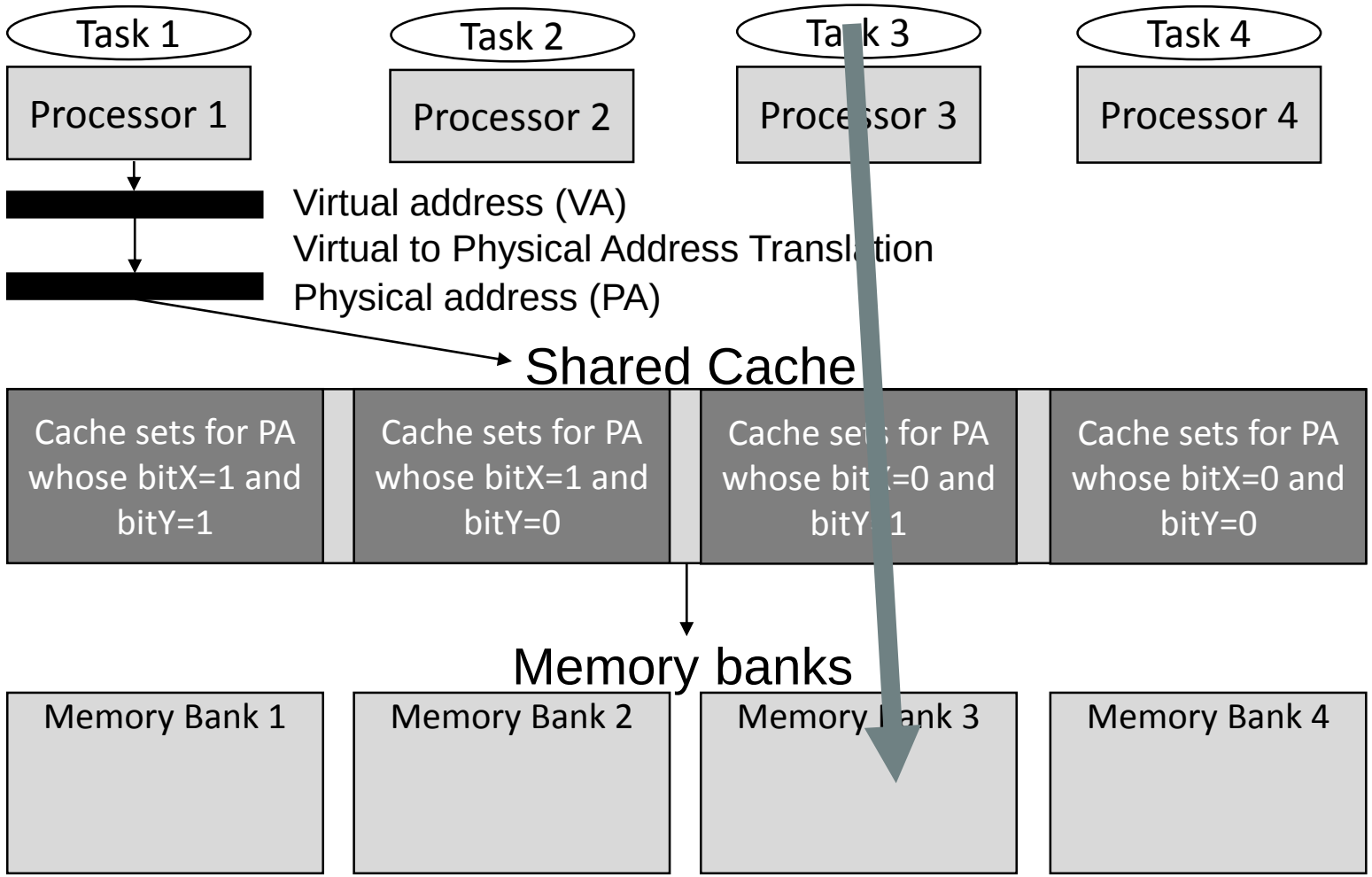
Bank coloring



Goal: Make sure that all memory accesses from task 2 go to leftmost cache sets and to memory bank 2.

Mechanisms, Mechanisms for a single resource

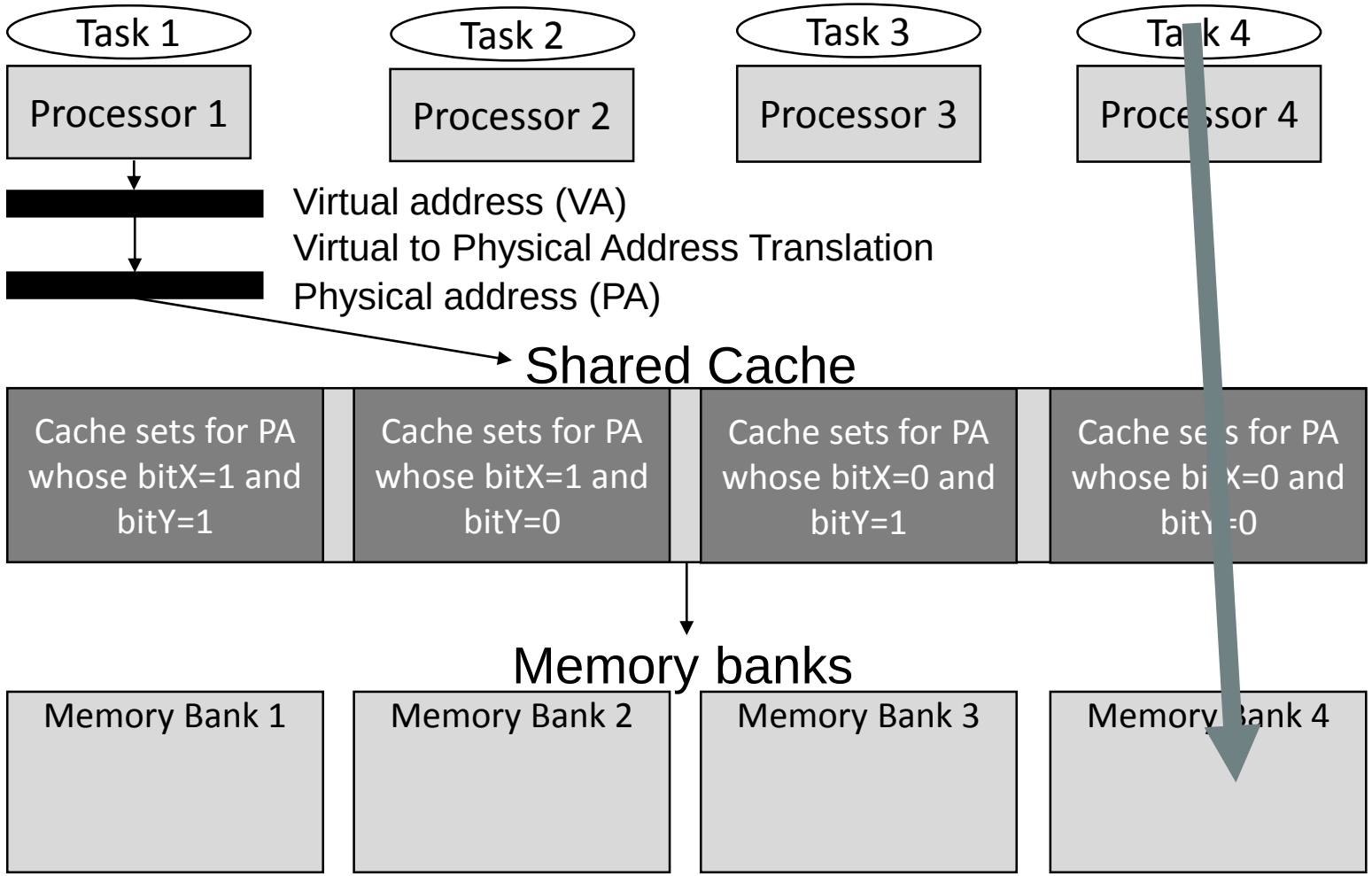
Bank coloring



Goal: Make sure that all memory accesses from task 3 go to leftmost cache sets and to memory bank 3.

Mechanisms, Mechanisms for a single resource

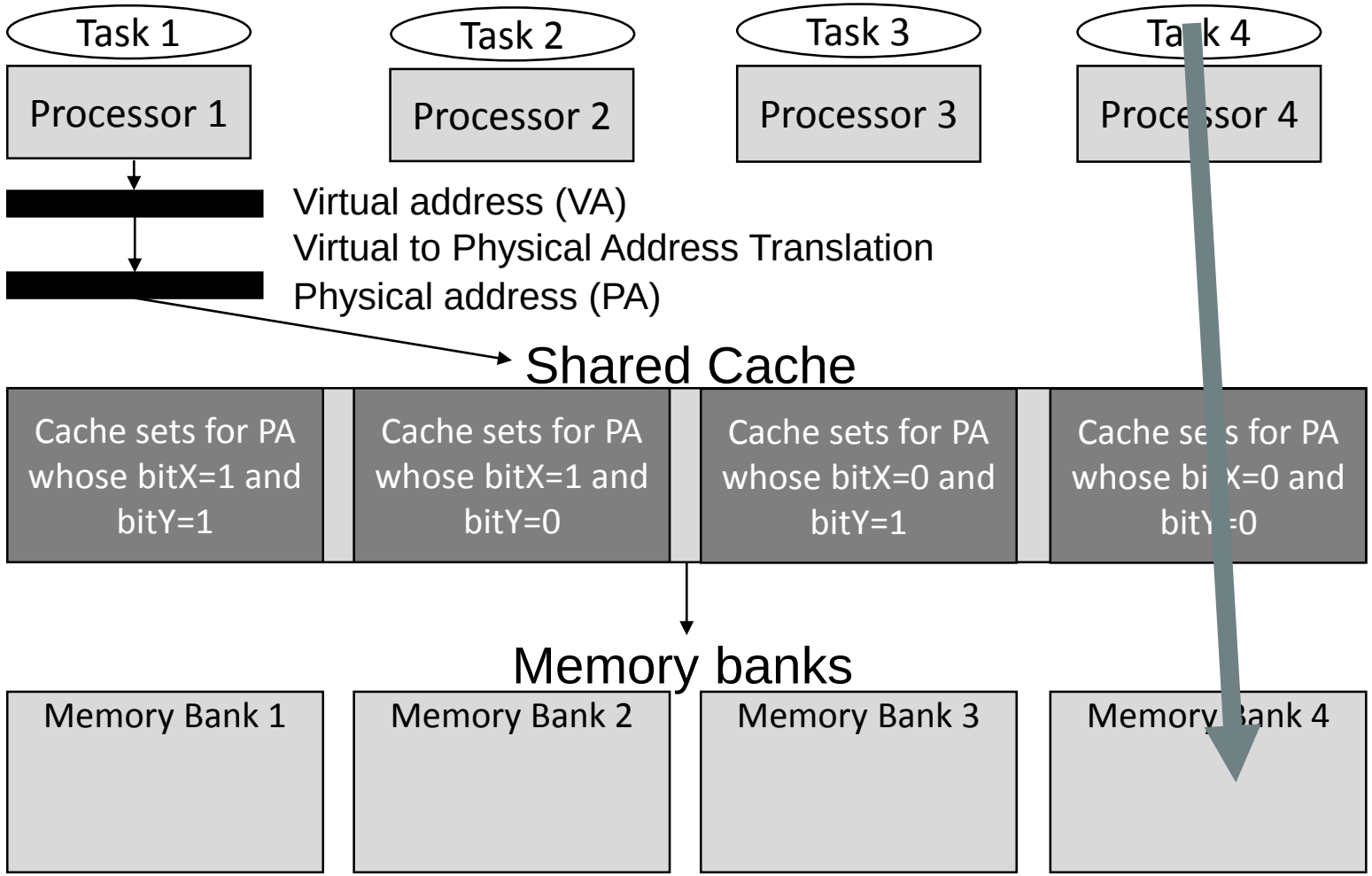
Bank coloring



Goal: Make sure that all memory accesses from task 4 go to leftmost cache sets and to memory bank 4.

Mechanisms, Mechanisms for a single resource

Bank coloring

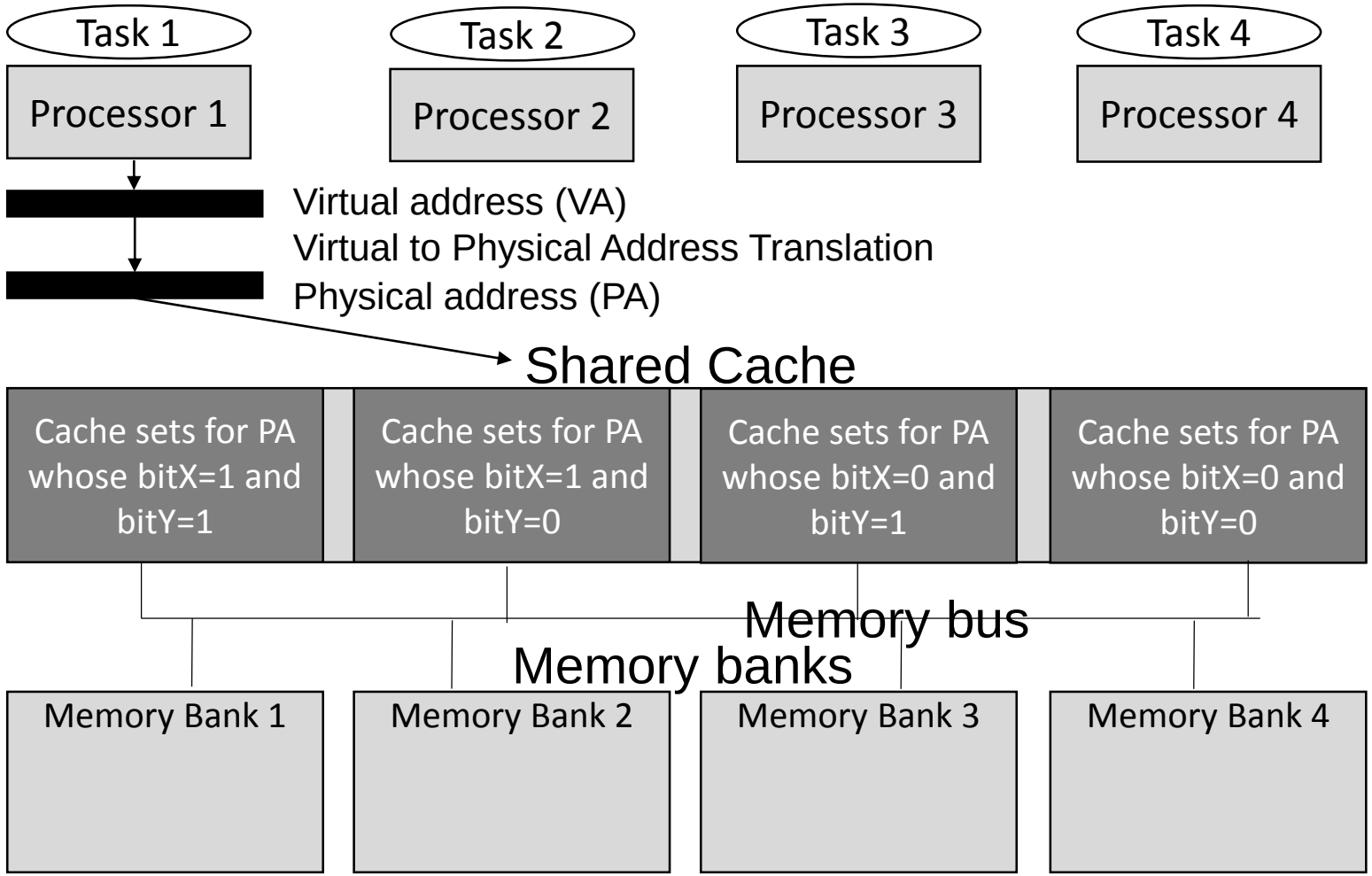


We can achieve these goals by setting up the virtual-to-physical translation mechanism. I.e., setting up the page table in the OS properly can be used to achieve cache isolation.

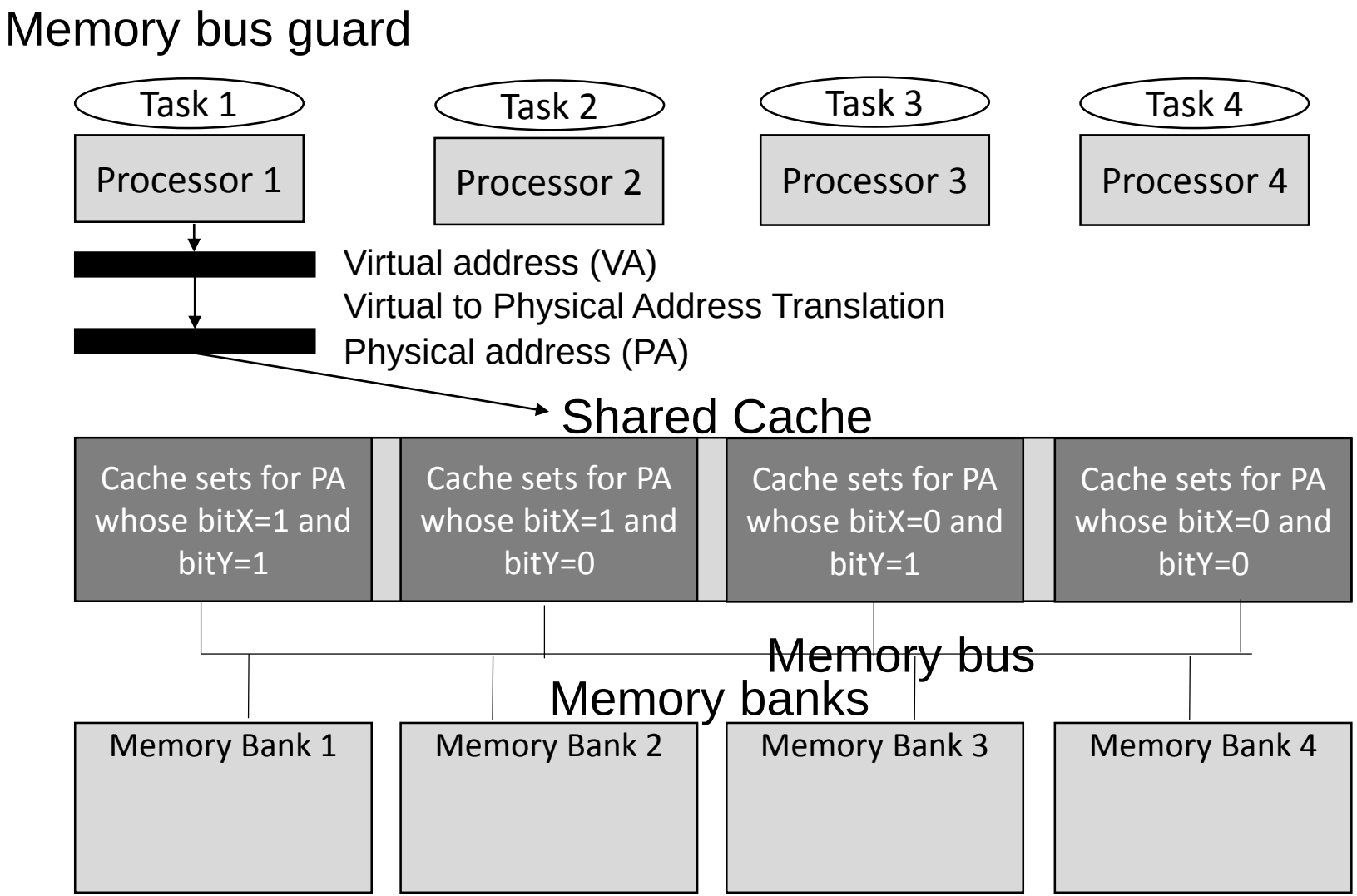
Memory bus guard

Mechanisms, Mechanisms for a single resource

Memory bus guard



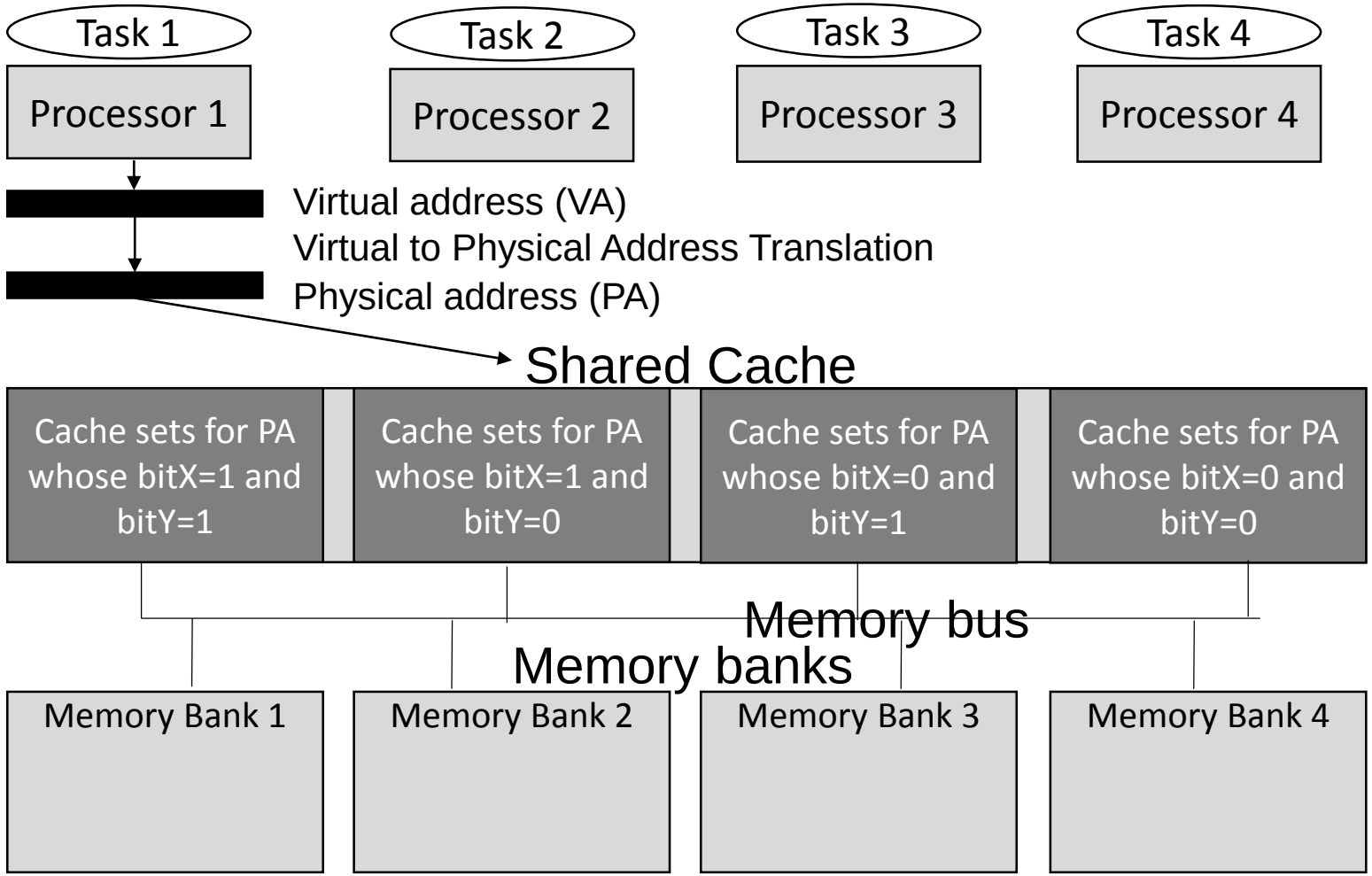
Mechanisms, Mechanisms for a single resource



We can assign, for each processor, a budget that specifies how many memory accesses it is permitted to do over the memory bus in a time interval of a given duration.

Mechanisms, Mechanisms for a single resource

Memory bus guard

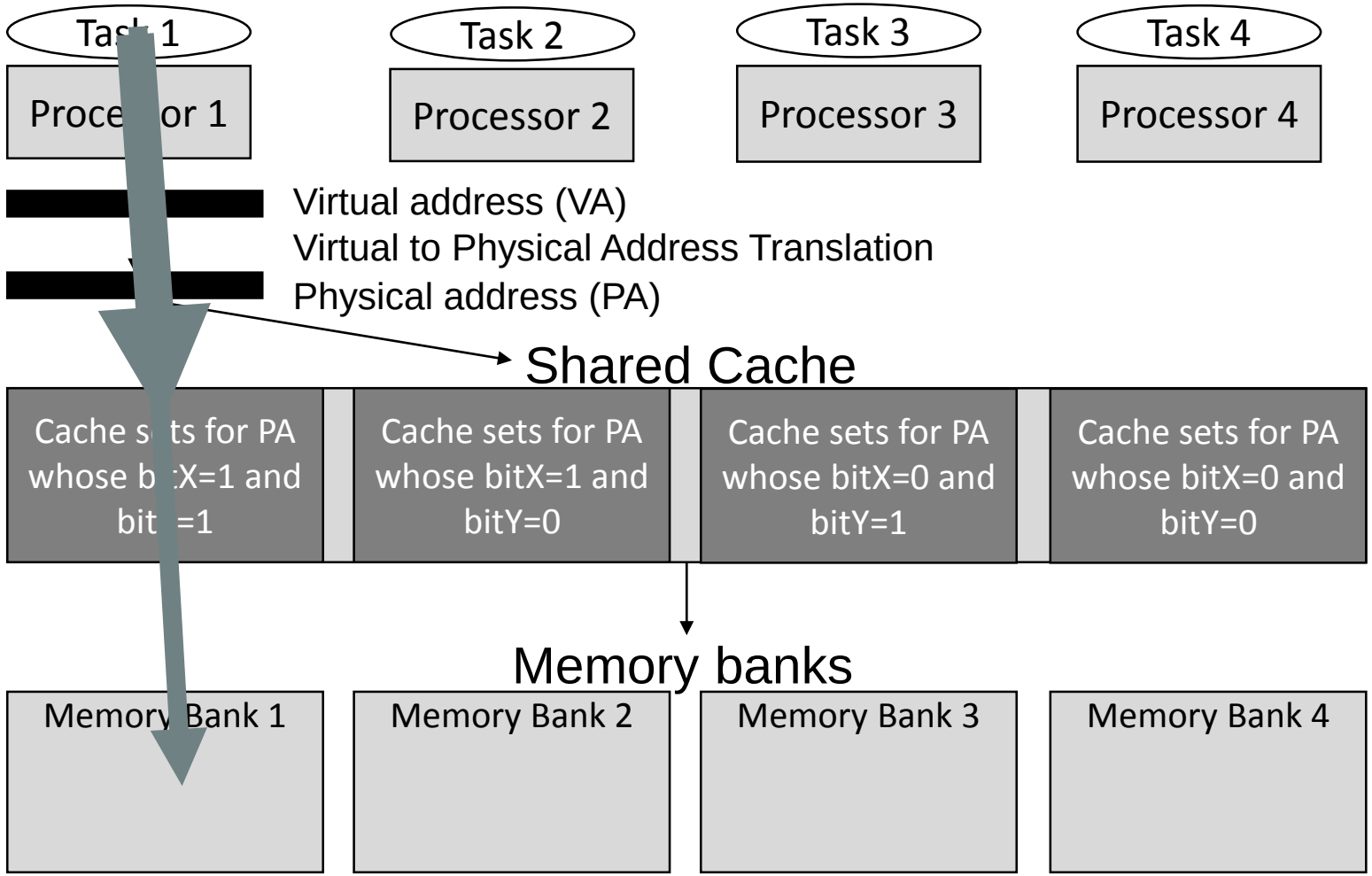


If a processor requests more, then suspend the task.

Coordinated cache and bank coloring

Mechanisms, Mechanisms for a many resources

Coordinated cache and bank coloring



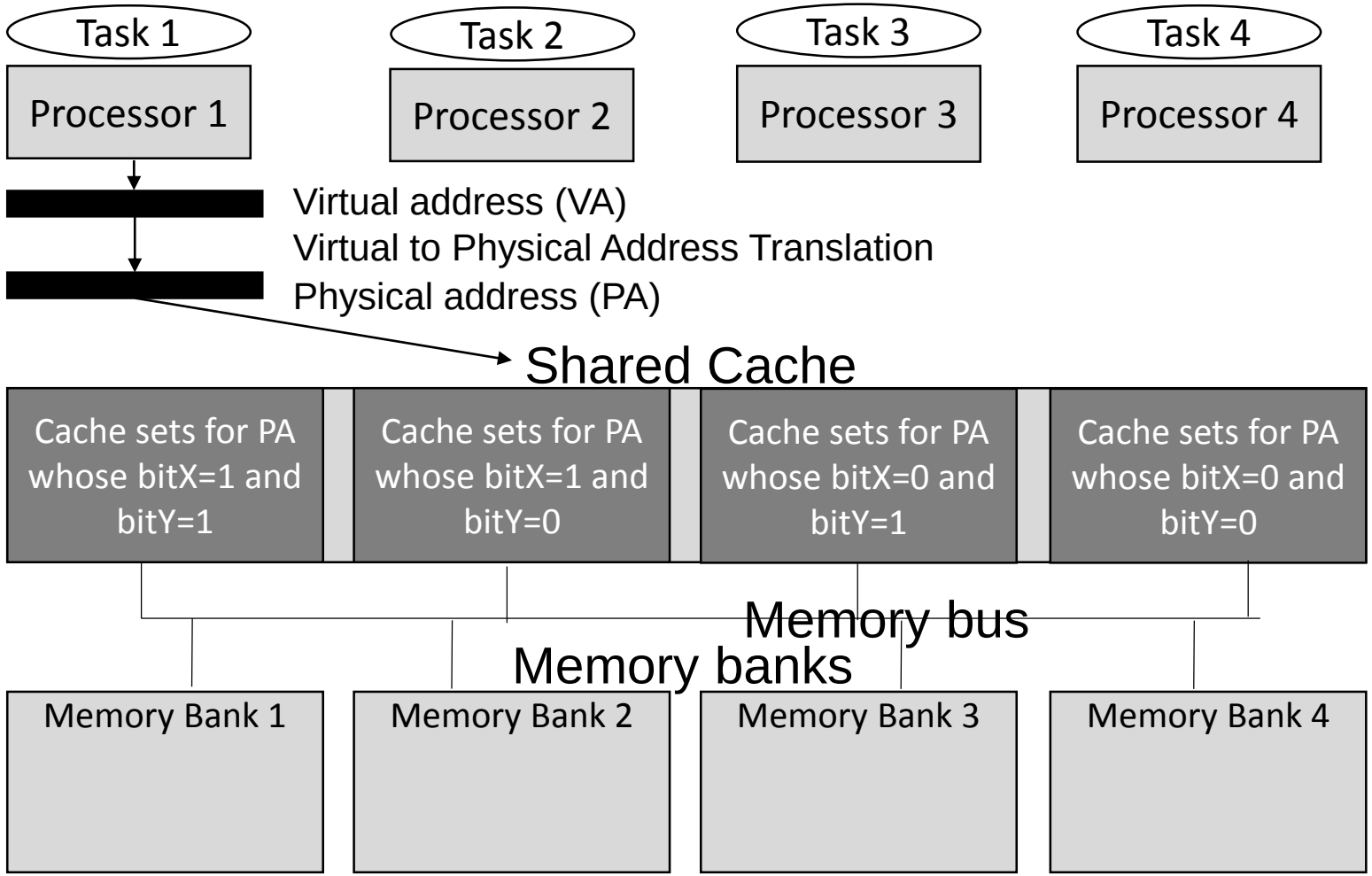
Set up the virtual-to-physical translation mechanism so that:
1. All memory accesses from task 1 goes to the 1st leftmost cache set and if it results in a cache miss, then it uses memory bank 1.

Analogous for 2,3, and 4.

Analysis, Work-conserving bus

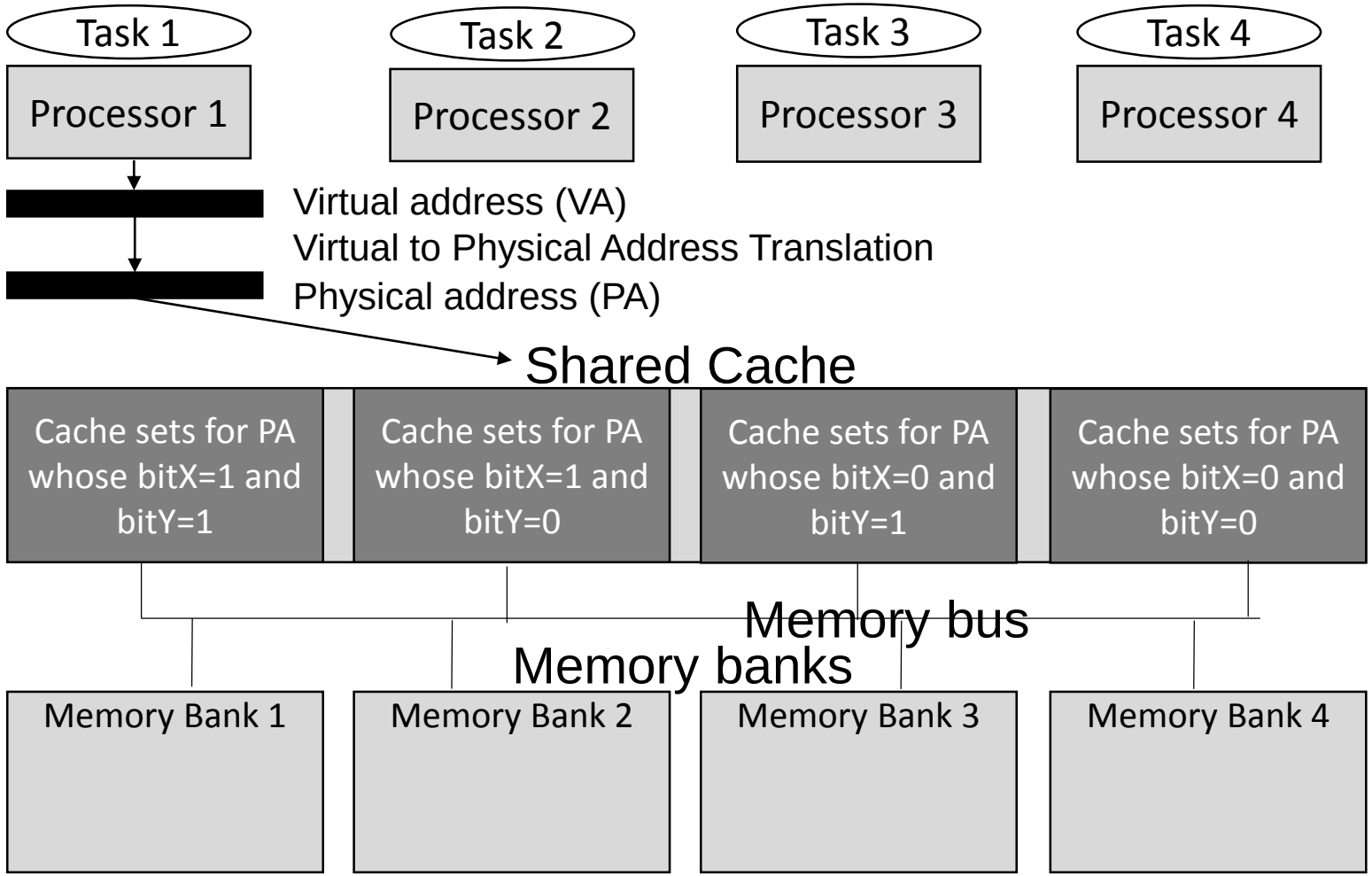
Analysis, Work-conserving bus

Memory bus guard



Analysis, Work-conserving bus

Memory bus guard



If we know an upper bound on the number of memory accesses from each processor and if we know that the memory bus is work conserving, then we can compute an upper bound on the total stalling time that a processor can experience due to contention for the memory bus.

Analysis, Analysis with bank sharing and more advanced model

We can model the details of the DRAM memory, the memory controller and compute the response times of tasks considering these effects.

$$R_i^{k+1} = C_i + \sum_{\tau_j \in hp(\tau_i)} \left\lceil \frac{R_i^k}{T_j} \right\rceil \cdot C_j$$

$$+ \min \left\{ H_i \cdot RD_p + \sum_{\tau_j \in hp(\tau_i)} \left\lceil \frac{R_i^k}{T_j} \right\rceil \cdot H_j \cdot RD_p,\ JD_p(R_i^k) \right\}$$

Analysis, Analysis with bank sharing and more advanced model

We can model the details of the DRAM memory, the memory controller and compute the response times of tasks considering these effects.

The tasks own execution

$$R_i^{k+1} = C_i + \sum_{\tau_j \in hp(\tau_i)} \left\lceil \frac{R_i^k}{T_j} \right\rceil \cdot C_j$$

$$+ \min \left\{ H_i \cdot RD_p + \sum_{\tau_j \in hp(\tau_i)} \left\lceil \frac{R_i^k}{T_j} \right\rceil \cdot H_j \cdot RD_p,\ JD_p(R_i^k) \right\}$$

Analysis, Analysis with bank sharing and more advanced model

We can model the details of the DRAM memory, the memory controller and compute the response times of tasks considering these effects.

Preemptions from higher priority tasks

$$R_i^{k+1} = C_i + \sum_{\tau_j \in hp(\tau_i)} \left\lceil \frac{R_i^k}{T_j} \right\rceil \cdot C_j$$

$$+ \min \left\{ H_i \cdot RD_p + \sum_{\tau_j \in hp(\tau_i)} \left\lceil \frac{R_i^k}{T_j} \right\rceil \cdot H_j \cdot RD_p,\ JD_p(R_i^k) \right\}$$

Analysis, Analysis with bank sharing and more advanced model

We can model the details of the DRAM memory, the memory controller and compute the response times of tasks considering these effects.

$$R_i^{k+1} = C_i + \sum_{\tau_j \in hp(\tau_i)} \left\lceil \frac{R_i^k}{T_j} \right\rceil \cdot C_j$$

$$+ \min \left\{ H_i \cdot RD_p + \sum_{\tau_j \in hp(\tau_i)} \left\lceil \frac{R_i^k}{T_j} \right\rceil \cdot H_j \cdot RD_p,\ JD_p(R_i^k) \right\}$$

Delay caused by the memory system

Analysis, Co-Runners

There are many other resources in the memory system (e.g., MSHR) and the details of those resources are typically not publicly known. Hence, we need a method to analyze timing despite of that.

Analysis, Co-Runners

We can create a model where the execution speed of a task at a given time depends on which other tasks execute in parallel with it at this given time. For example, task 1 executes with speed 1 normally but task 1 executes with speed ½ when it executes in parallel with task 2 (on another processor)

Analysis, Co-Runners

We can obtain parameters of such a model through measurements.

Analysis, Co-Runners

We can then compute response times.

1. Compute the amount of duration that a processor can be busy from requests arriving in a time interval of duration *t*.

$$\text{maximize} \quad \sum_{i' \in \text{hep}(\tau,\Pi,i)} \sum_{v_{i'}^{k'} \in V_{i'}} \sum_{s \in S(\tau,\Pi,i',k')} \text{du}_s$$

subject to

$$\forall i' \in \text{hep}(\tau,\Pi,i), \forall v_{i'}^{k'} \in V_{i'}, \sum_{s \in S(\tau,\Pi,i',k')} \text{pw}_{i',s\setminus\{\langle i',k'\rangle\}}^{k'} \times \text{du}_s \leq \lceil \frac{t}{T_{i'}} \rceil \times C_{i'}^{k'}$$

$$\forall i' \in \text{top}(\tau,\Pi,\text{i}), \forall v_{i'}^{k'} \in V_{i'}, \sum_{\substack{s \in S(\tau,\Pi,i',k') \wedge \\ (\exists \langle i'',k''\rangle \text{ s.t. } (i'' \in \text{hep}(\tau,\Pi,\text{i})) \wedge (\langle i'',k''\rangle \in s))}} \text{pw}_{i',s\setminus\{\langle i',k'\rangle\}}^{k'} \times \text{du}_s \leq \text{xUB}(\tau,\Pi,i',k',t)$$

$$\forall s \in S(\tau,\Pi) \text{ s.t. } (\exists \langle i'',k''\rangle \text{ s.t. } (i'' \in \text{hep}(\tau,\Pi,\text{i})) \wedge (\langle i'',k''\rangle \in s)), \text{du}_s \in \mathbb{R}_{\geq 0}$$

2. Use the result in 1. to compute R_i.

Unsolved Challenges

- Combining mechanisms
  - Many different levels of cache + TLB + memory bank
  - Dealing with L1 cache (cache coloring does not work)
- Combining allocation and analysis
  - Assign tasks to processors such that some schedulability analysis (e.g., co-runner analysis) succeeds
- Obtaining WCET on processors architectures whose internals are undocumented (e.g., x86)
- Port contention (e.g., shared cache), undocumented.
- Run-time monitoring and enforcement
- Managing critical sections without killing parallelism