Modeling Communication in Cache-Coherent SMP Systems

A Case-Study with Xeon Phi

Sabela Ramos[1] (sramos@udc.es)
Torsten Hoefler[2] (htor@inf.ethz.ch)

[1]Computer Architecture Group, University of A Coruña (Spain)
[2]Scalable Parallel Computing Lab, ETH Zurich (Switzerland)

22nd ACM Intl. Symp. on HPDC, New York City, 2013

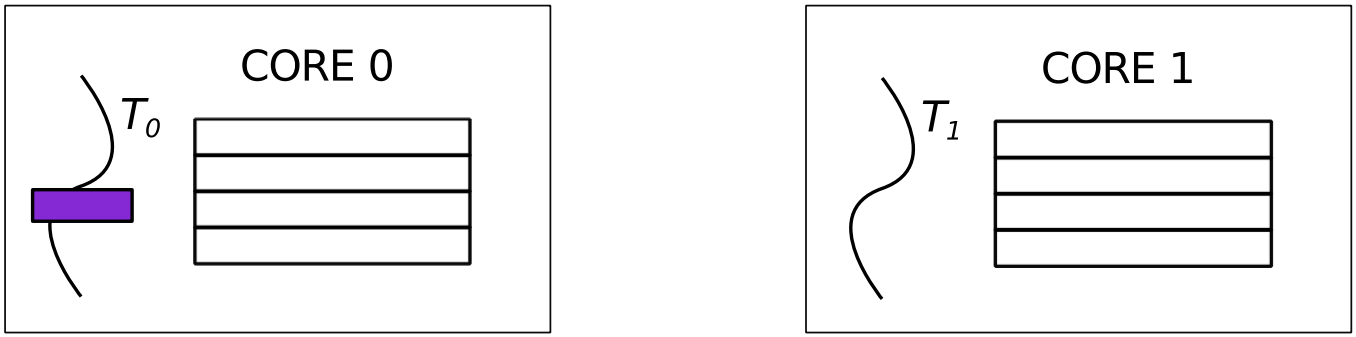
Outline

Motivation

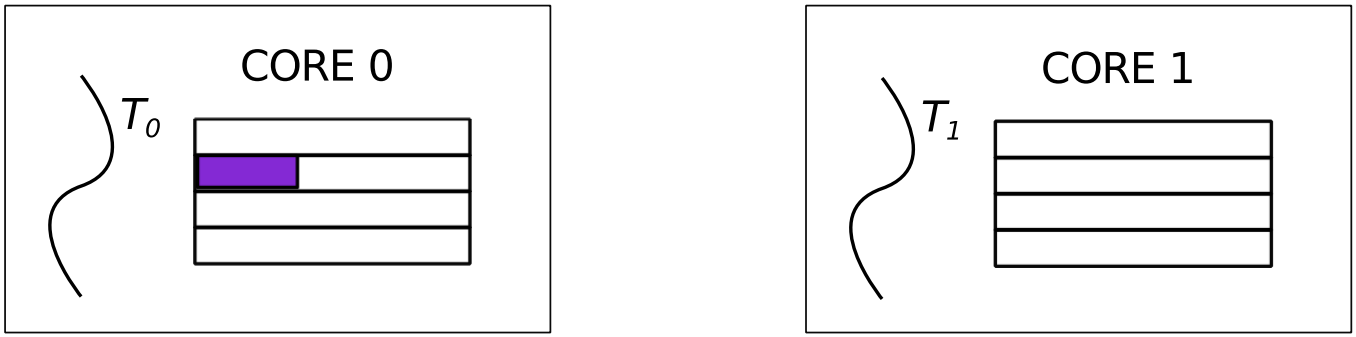
- Increase in the number of cores per processor.
- `x86` processors offer Cache Coherency (Xeon Phi).
  - Cache-coherency as the only means of communication between cores
  - CC protocols implemented using a state machine.

- PROPOSAL:
  - Model of the transition cost in the state machine.
  - Simplification of the full model.
  - Application of the model to algorithm optimization.

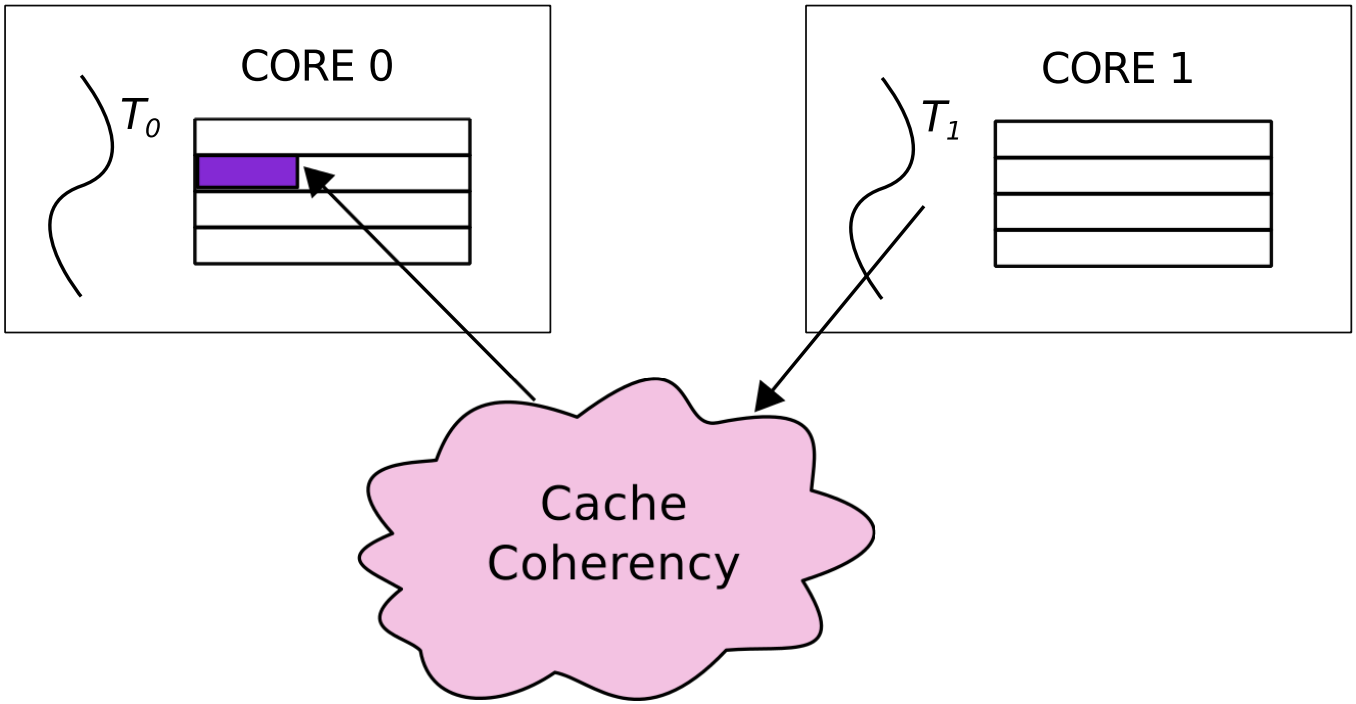
Communication in Cache-coherent Systems



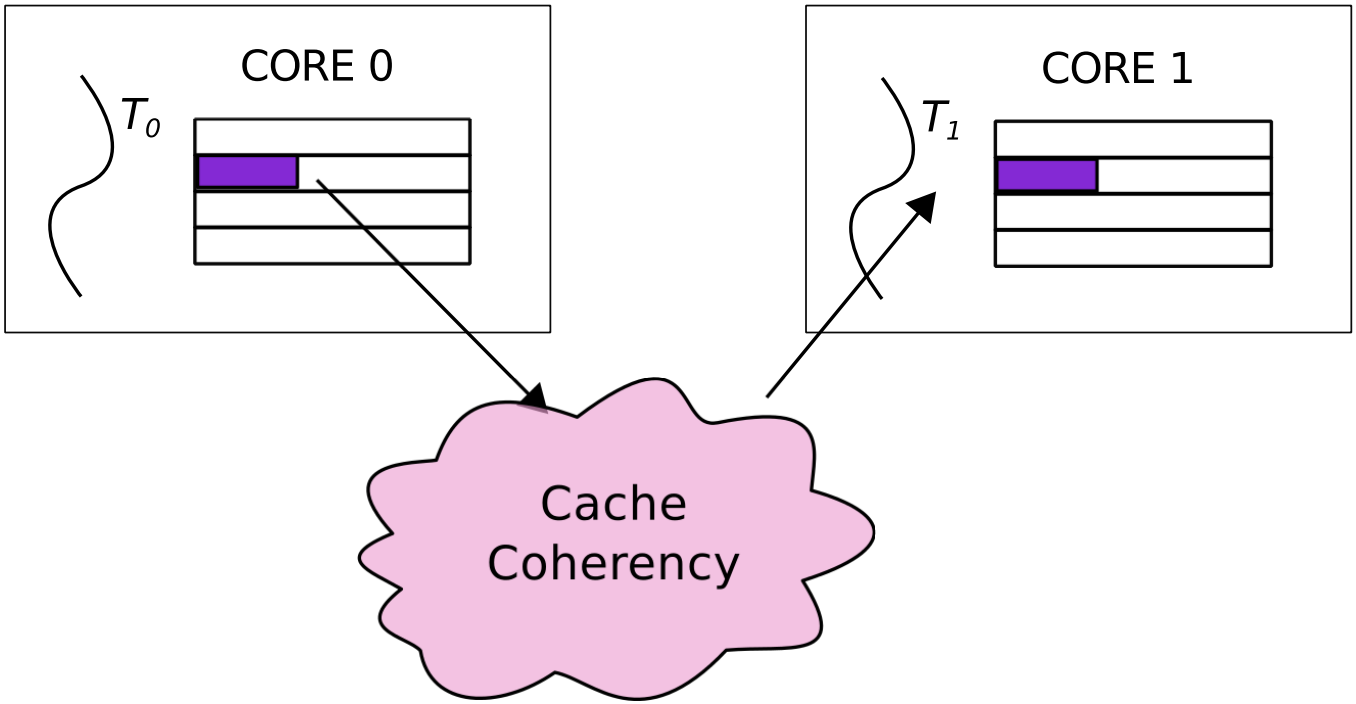
Communication in Cache-coherent Systems



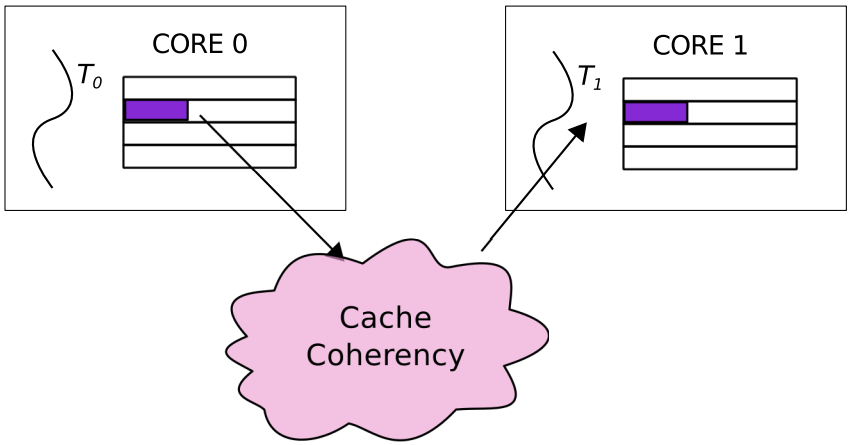
Communication in Cache-coherent Systems



Communication in Cache-coherent Systems



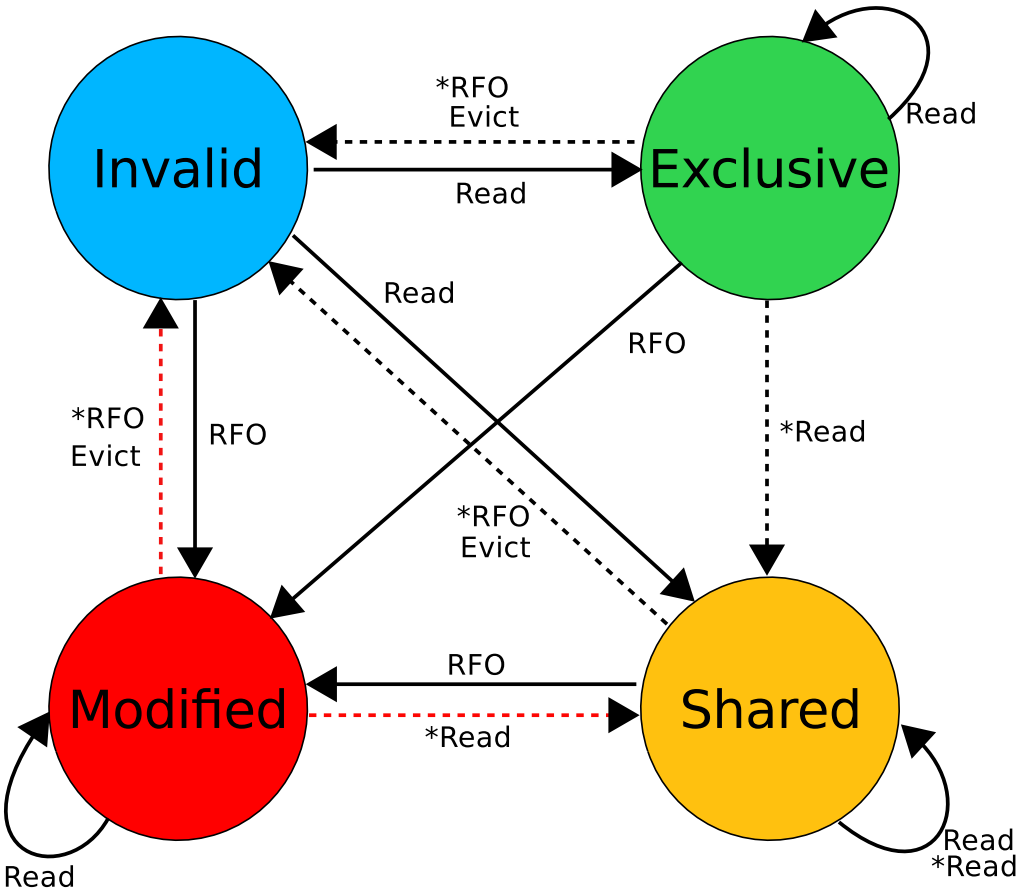
Communication in Cache-coherent Systems



- Where is the line located?
- Which is the state of the line? Is it modified?
- Do I have to fetch it from memory?

Cache-coherency protocols: MESI

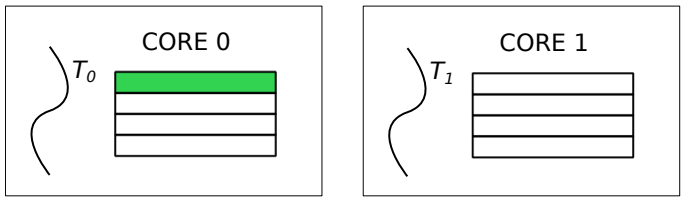
There are others like MOESI, MESIF or extended MESI[1].



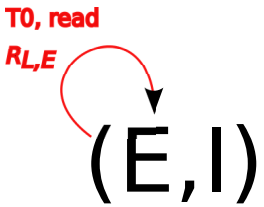
[1] M state can be extended to allow sharing of modified lines.

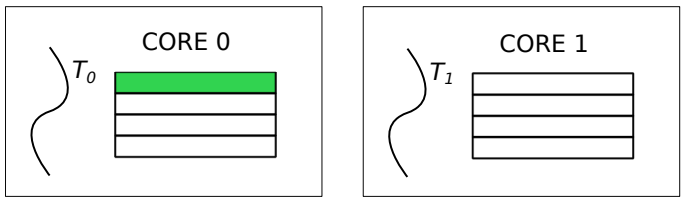
Transition diagram for extended MESI

(E,I)

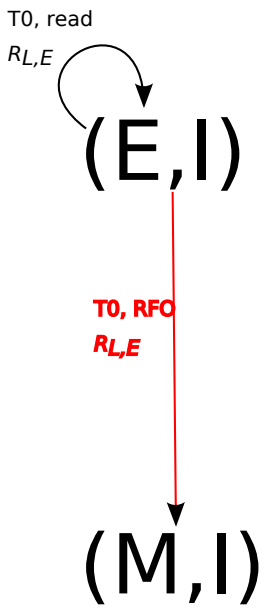


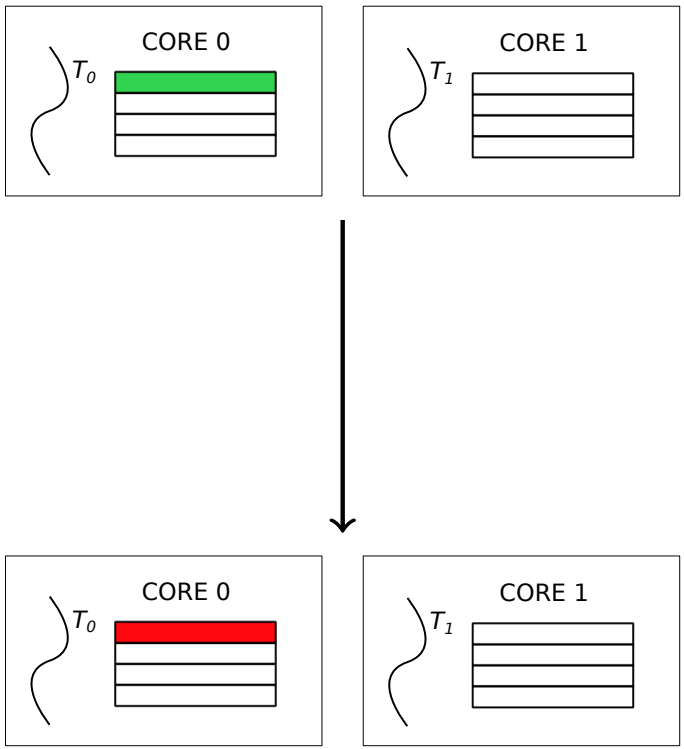
Transition diagram for extended MESI



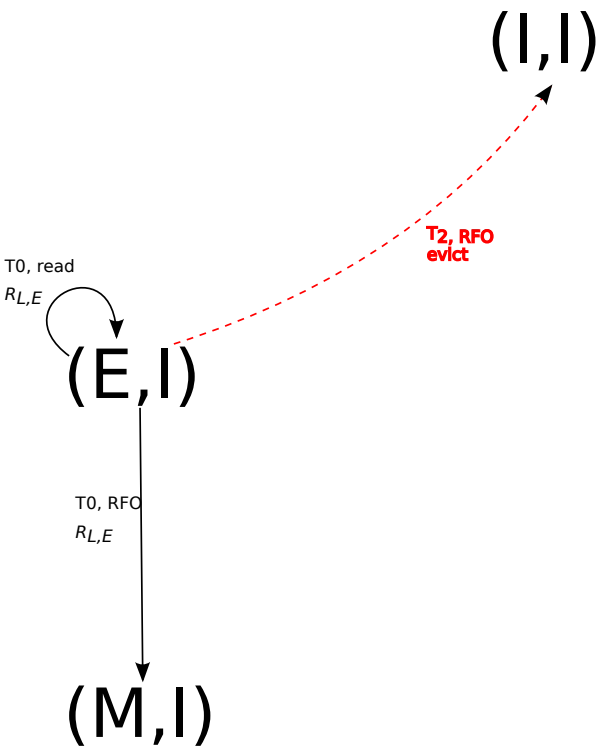


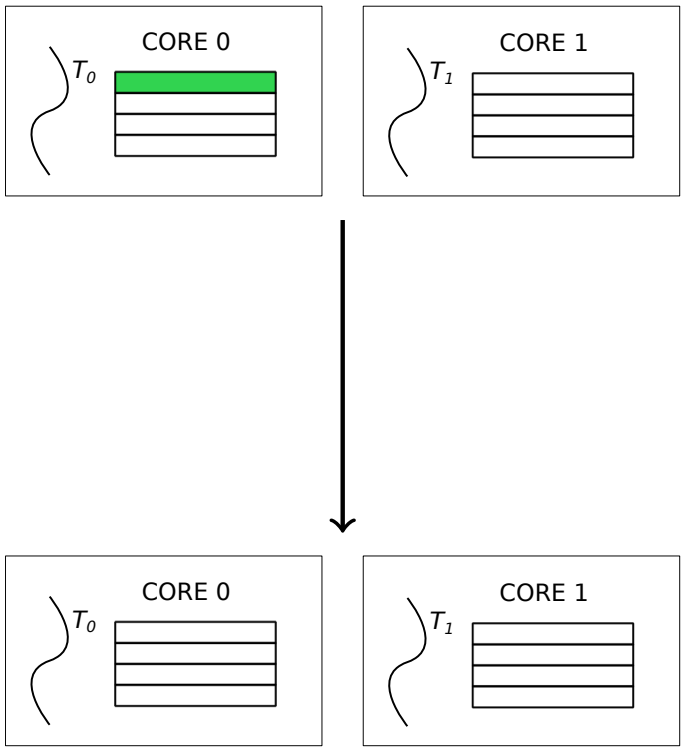
Transition diagram for extended MESI



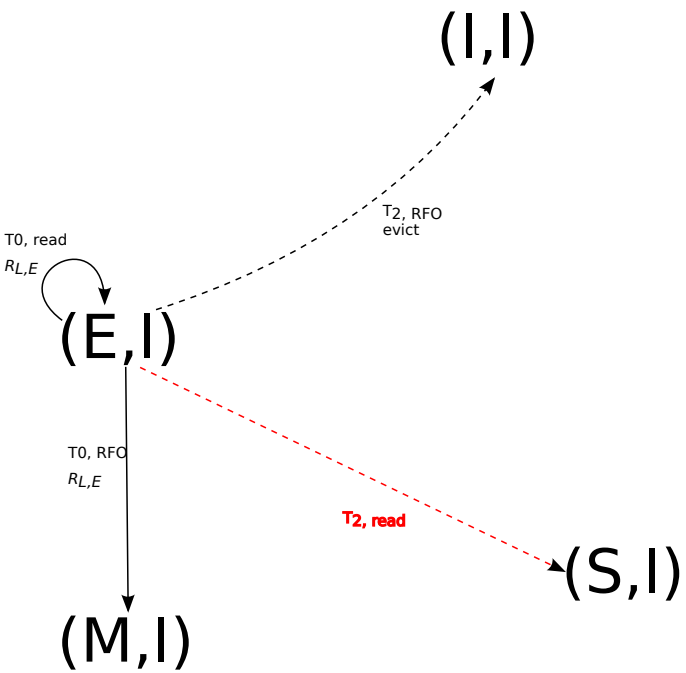


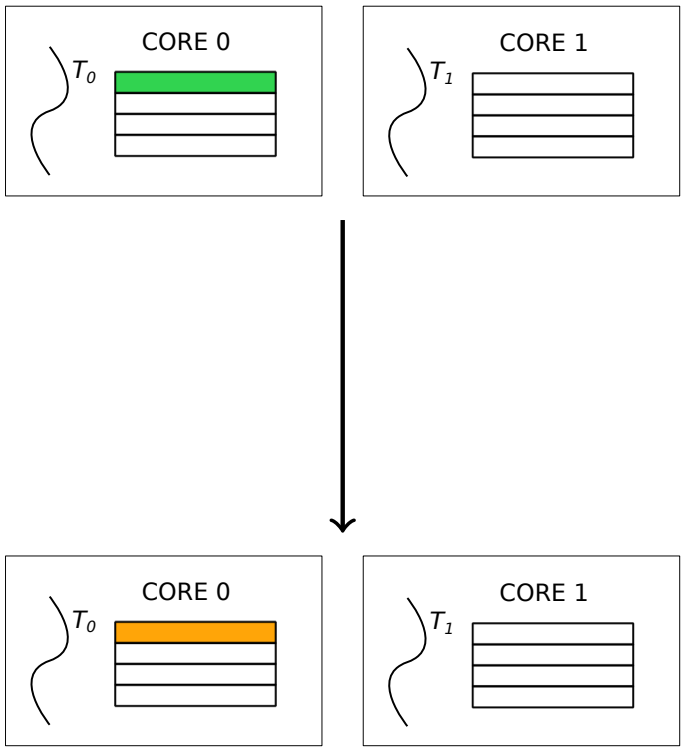
Transition diagram for extended MESI



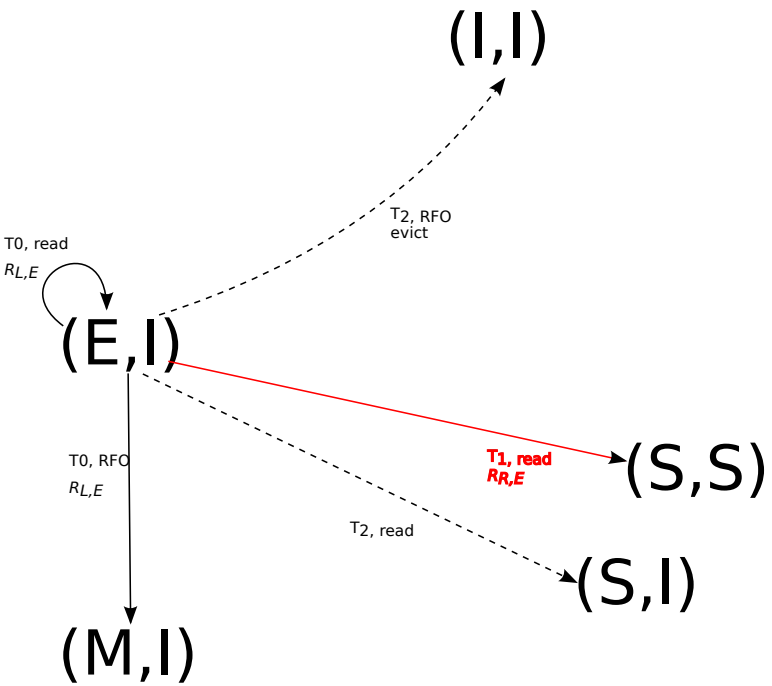


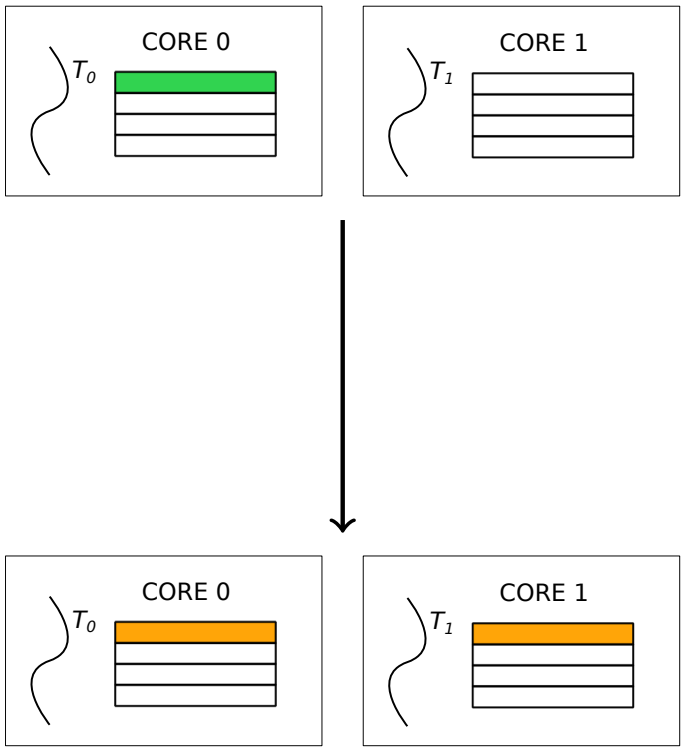
Transition diagram for extended MESI



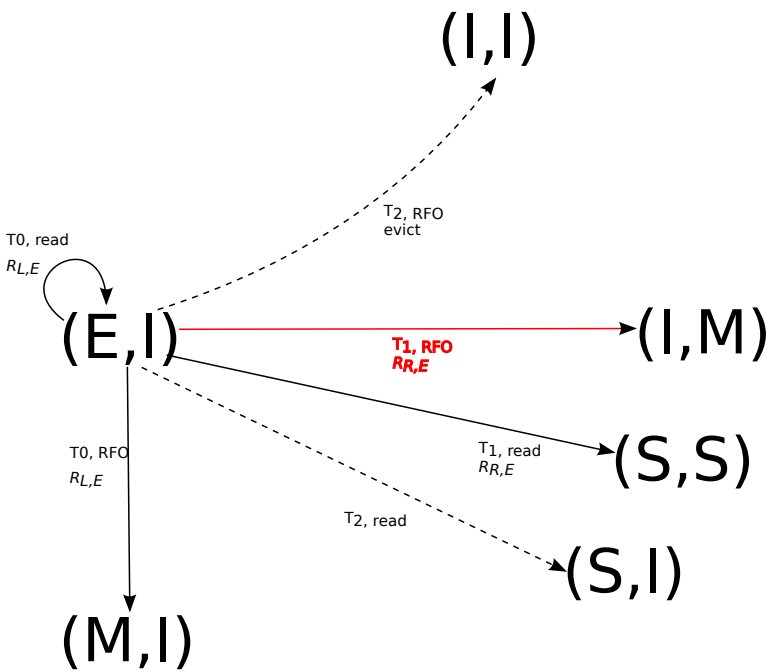


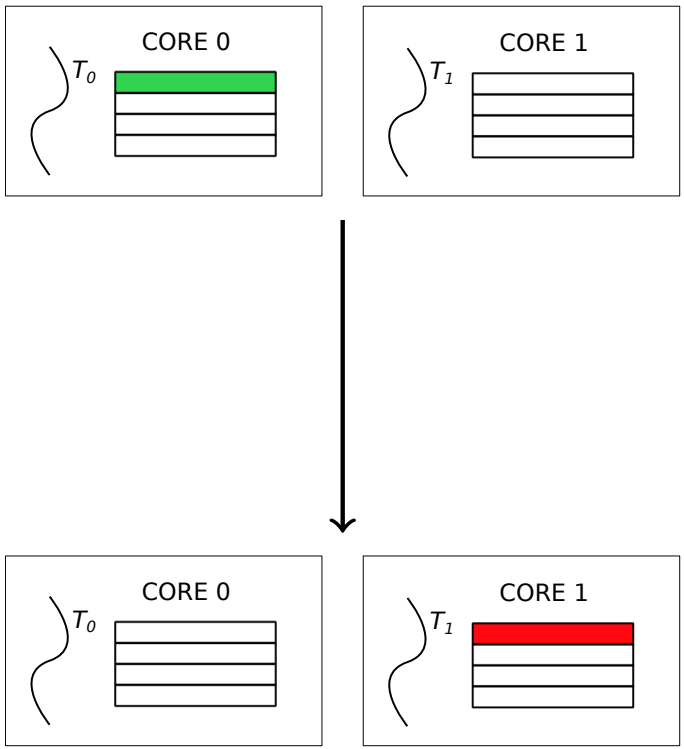
Transition diagram for extended MESI



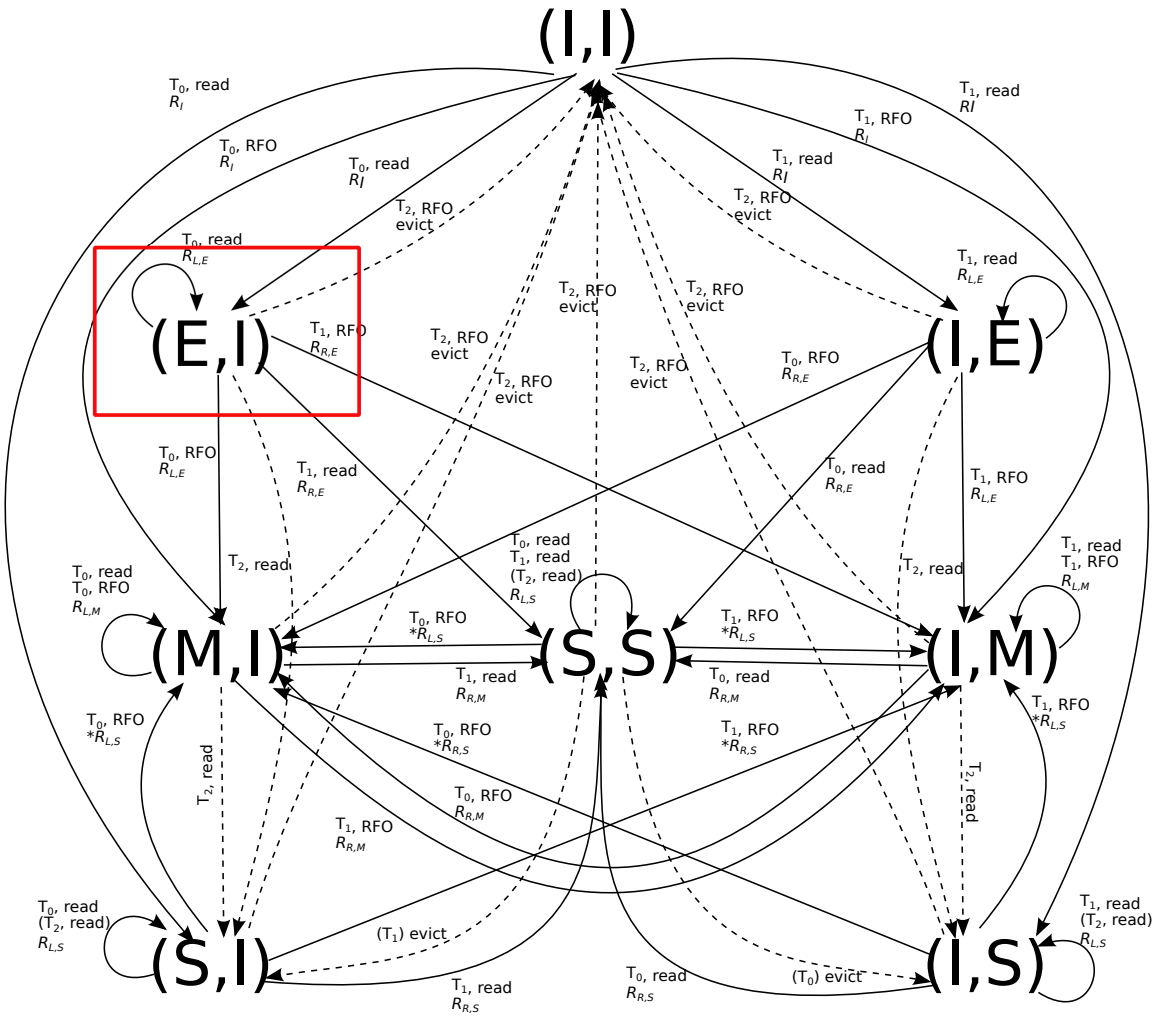


Transition diagram for extended MESI

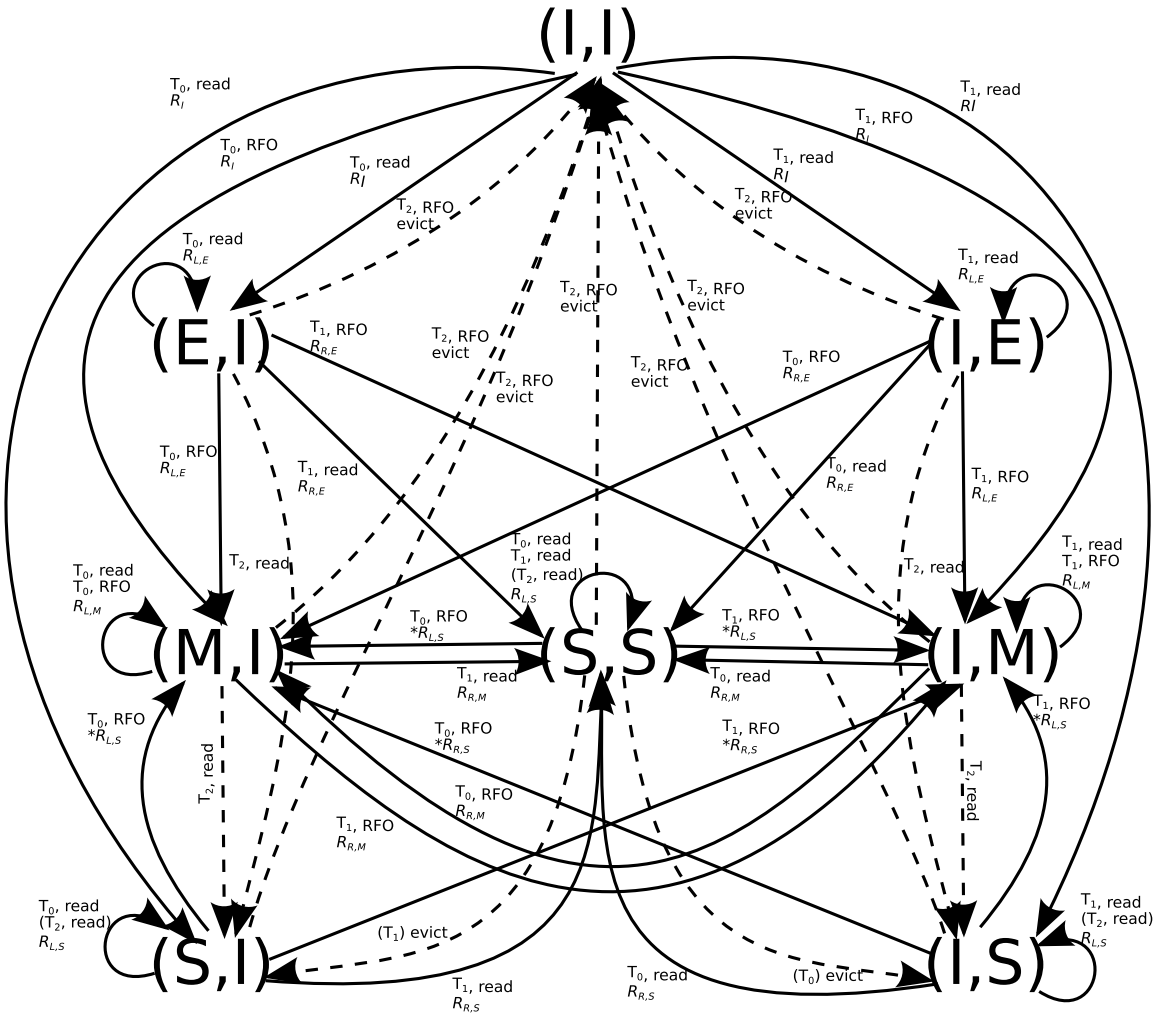




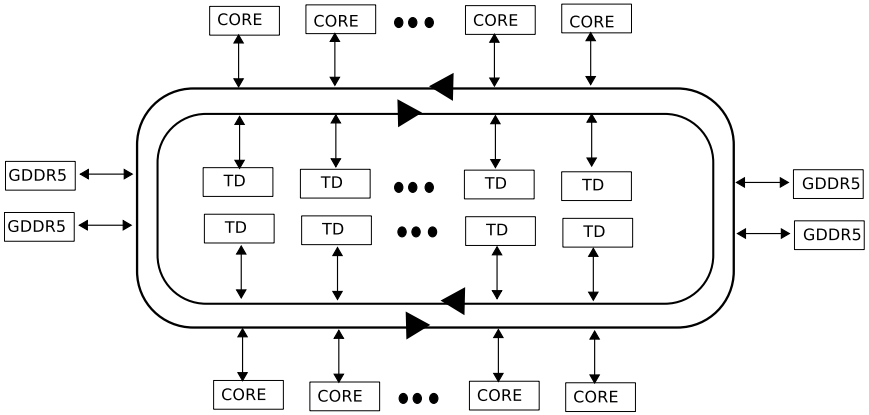
Transition diagram for extended MESI

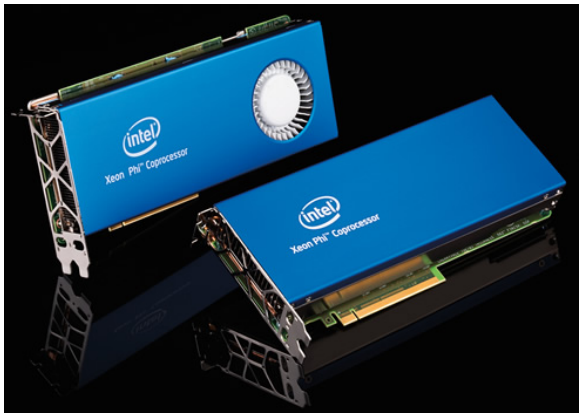


Transition diagram for extended MESI



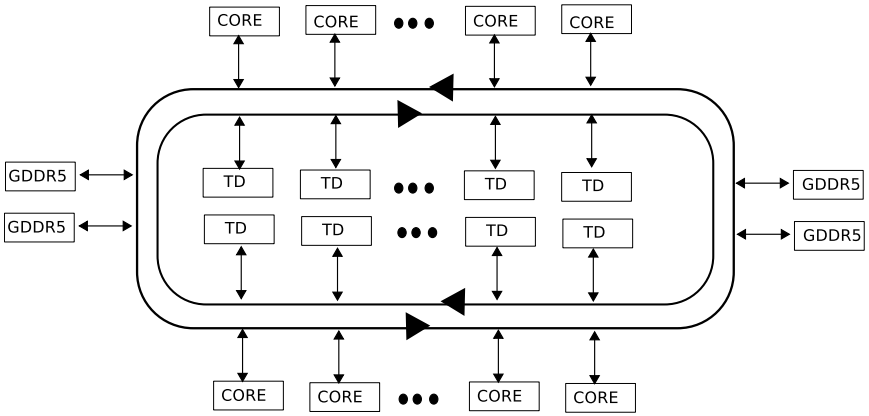
Intel Xeon Phi Architecture

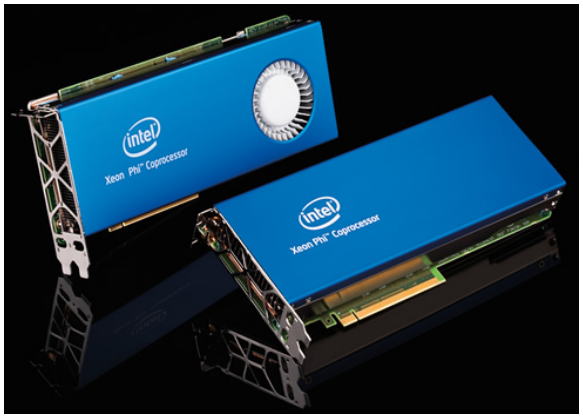




- Intel Xeon Phi 5110P, 60 cores at 1056 MHz (4 threads per core).
- Vector Processing Unit with 64 byte registers.
- L1: 32 kb Data + 32 kb Instructions
- L2: 512 kb unified

Intel Xeon Phi-Cache Coherency Protocol

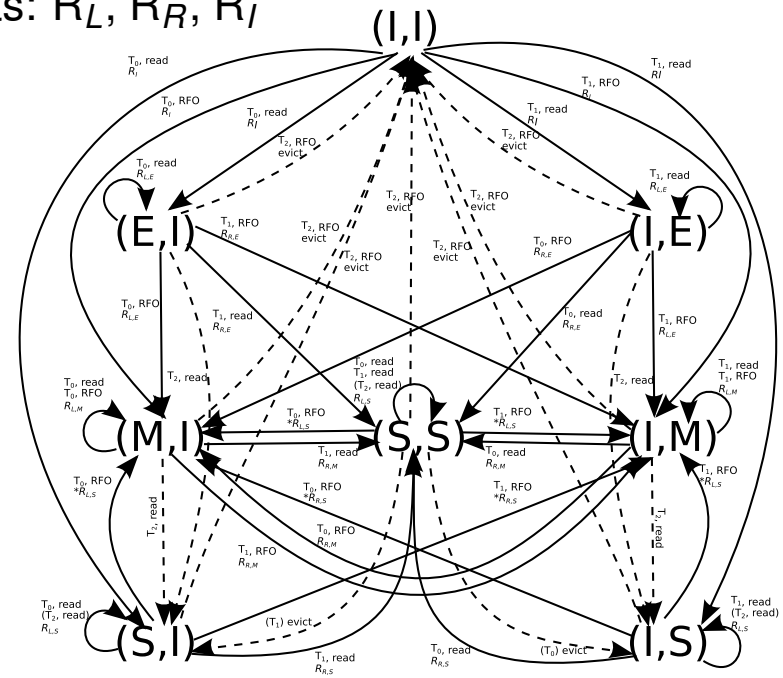




- Distributed Tag Directories.
  - GOLS coherency state.
  - Lines assigned by hash function based on the address.
  - Even load distribution but no locality of the network.
  - Benchmarking needs randomization in the accesses to avoid bias

Parametrization

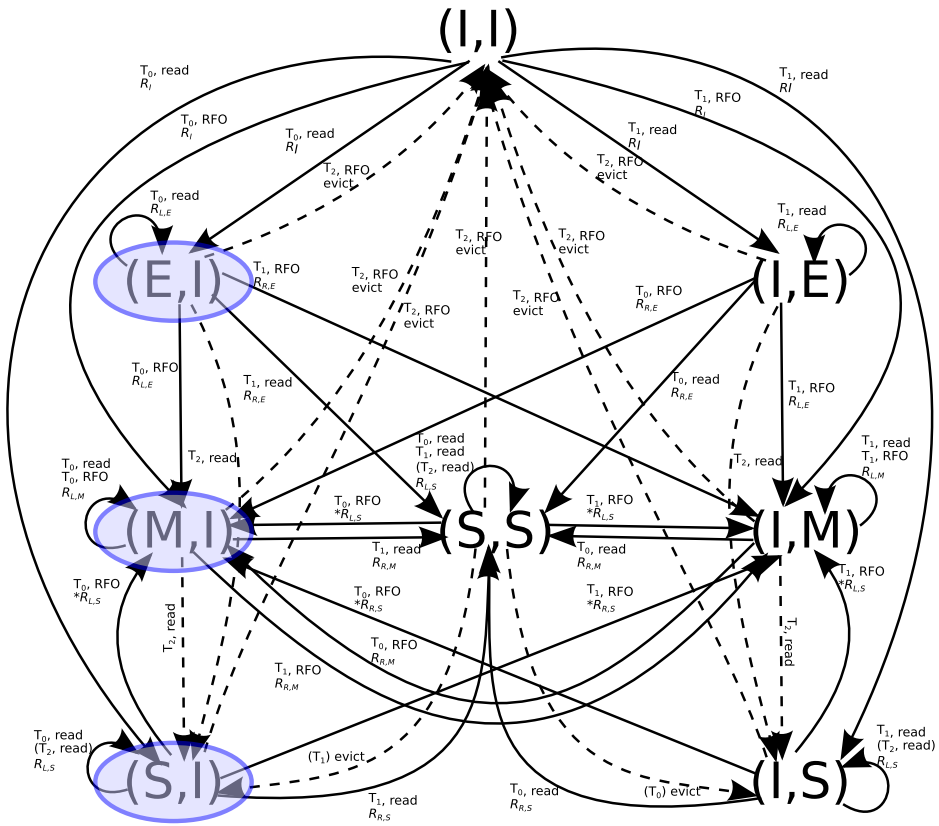
- BenchIT[2] to measure cache line transfer latencies.
  - No significant difference among cached states (S, M, E).
  - Distance between cores is nearly irrelevant, less than 5% (due to DTDs).
  - Three costs: R_L, R_R, R_I



[2] D. Molka, D. Hackenberg, R. Schoene and M. S. Mueller. Memory Performance and Cache Coherency Effects on an Intel Nehalem Multiprocessor System. In Proc. 18th Intl. Conf. on Parallel Architectures and Compilation Techniques (PACT'09), pages 261–270, Raleigh, NC, USA, 2009.

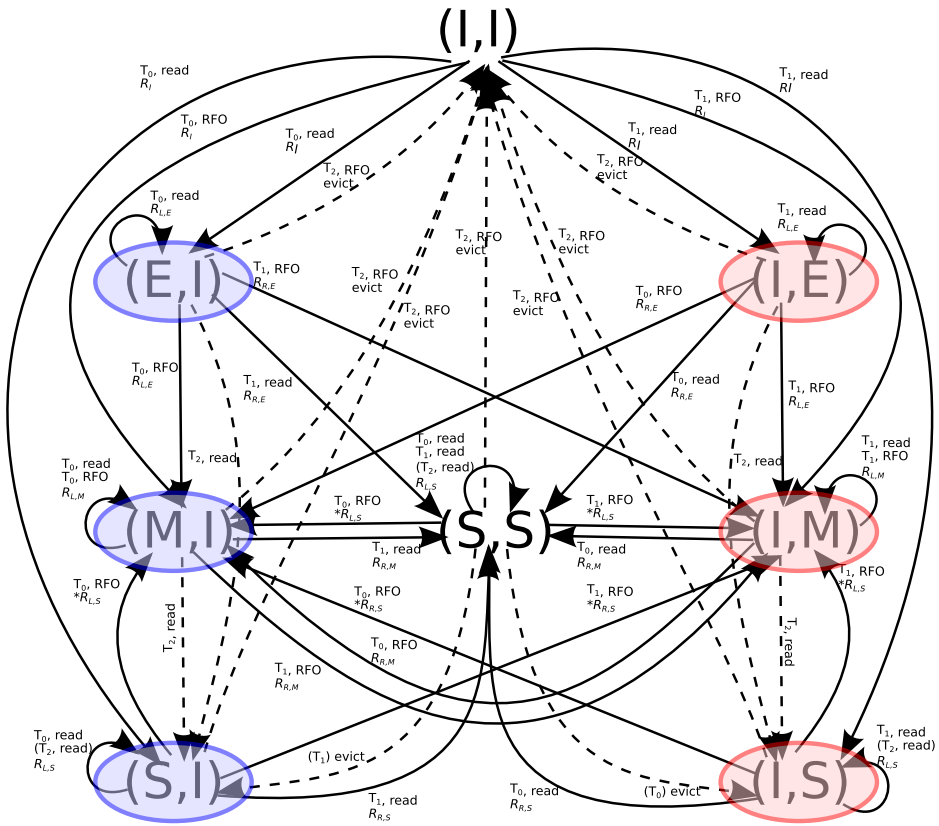
Parametrization

- No significant difference among cached states (S, M, E).
- Distance between cores is nearly irrelevant, less than 5% (due to DTDs).
- Three costs: R_L, R_R, R_I



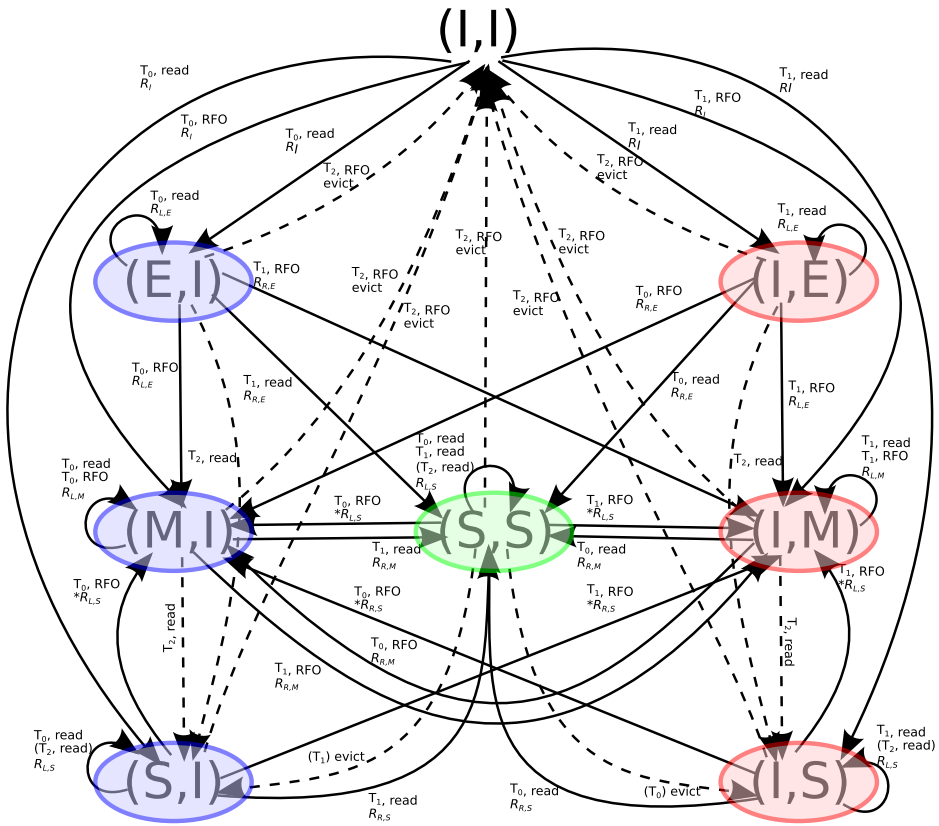
Parametrization

- No significant difference among cached states (S, M, E).
- Distance between cores is nearly irrelevant, less than 5% (due to DTDs).
- Three costs: R_L, R_R, R_I



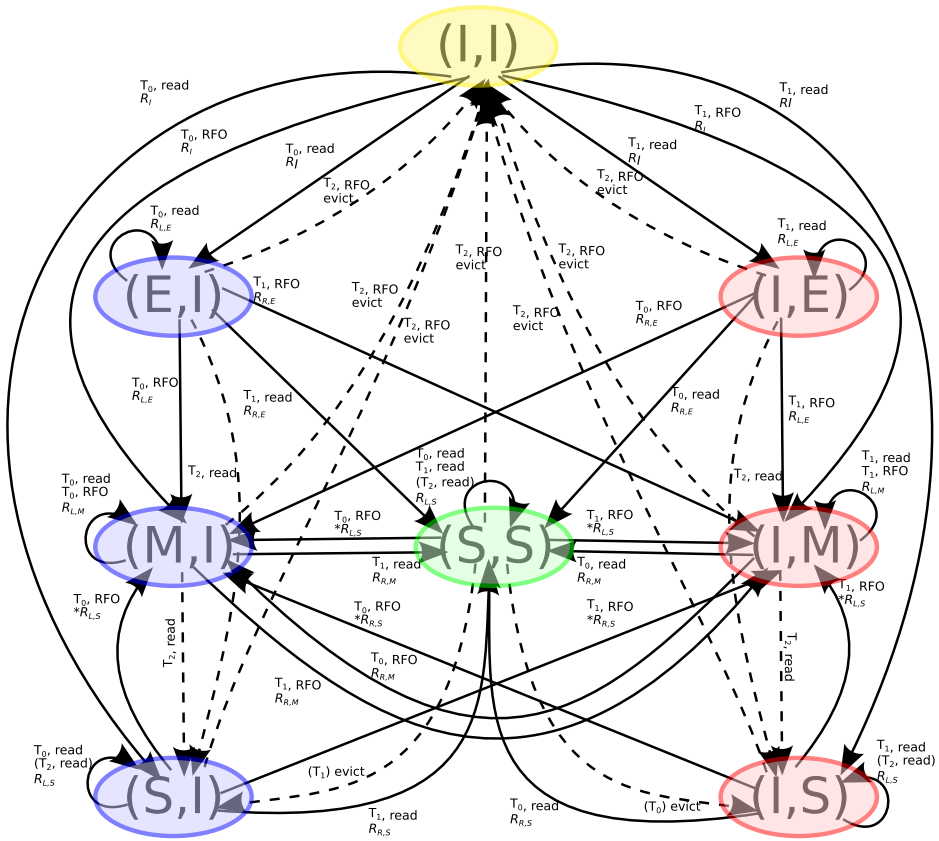
Parametrization

- No significant difference among cached states (S, M, E).
- Distance between cores is nearly irrelevant, less than 5% (due to DTDs).
- Three costs: R_L, R_R, R_I



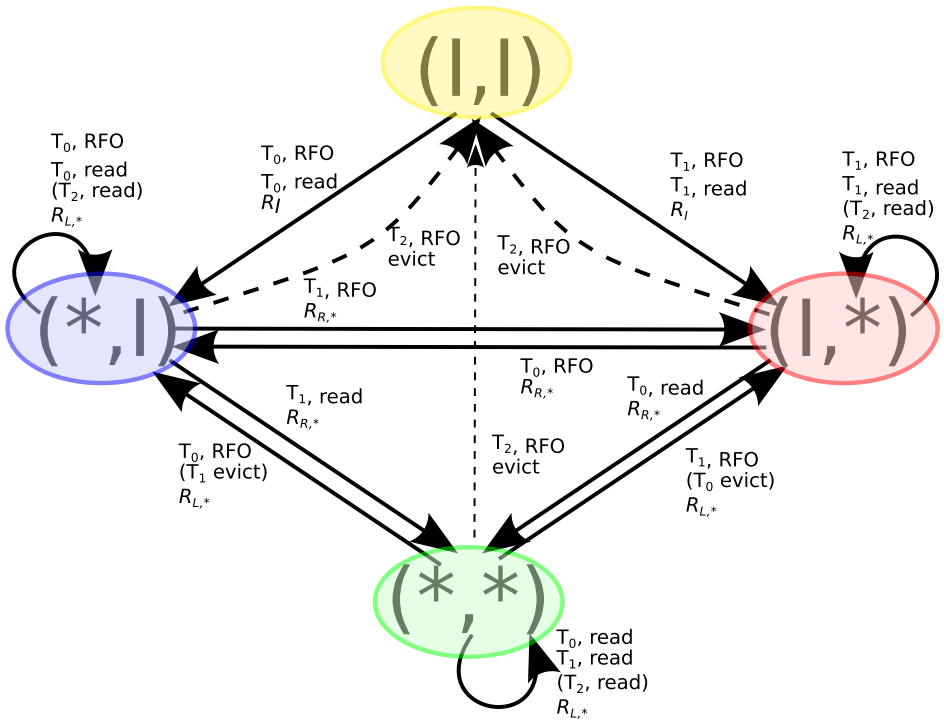
Parametrization

- No significant difference among cached states (S, M, E).
- Distance between cores is nearly irrelevant, less than 5% (due to DTDs).
- Three costs: R_L, R_R, R_I

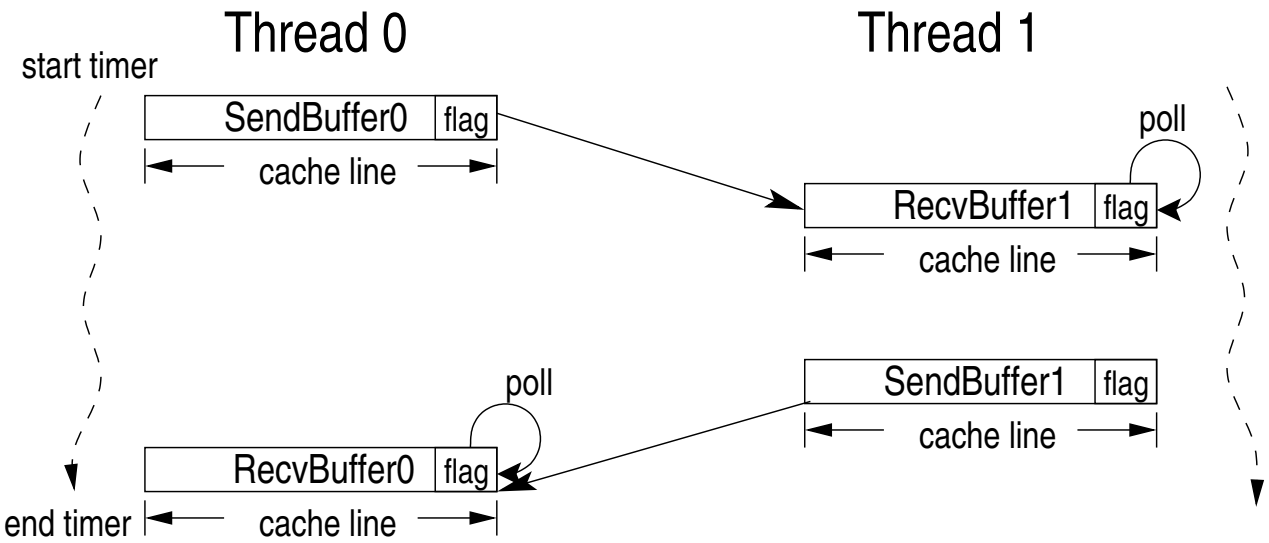


Parametrization

- No significant difference among cached states (S, M, E).
- Distance between cores is nearly irrelevant, less than 5% (due to DTDs).
- Three costs: R_L, R_R, R_I

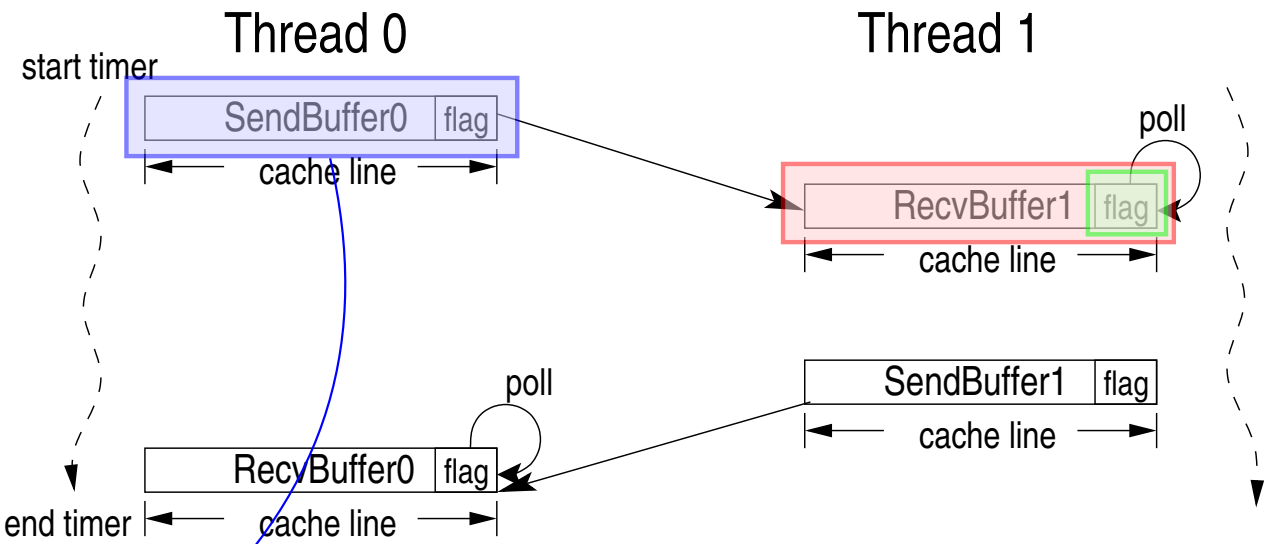


The Single-line Ping-Pong Model



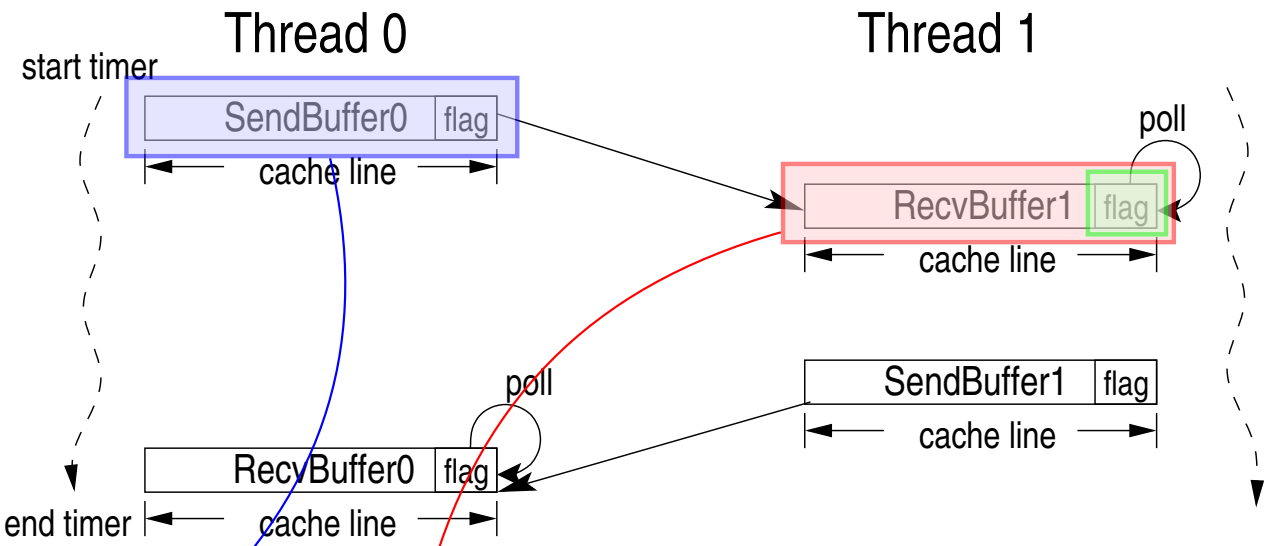
$$\mathcal{T}_1 = R_{L,\mathcal{S}_s} + R_{R,\mathcal{S}_r} + R_{R,M} + O = R_L + 2R_R + O$$

The Single-line Ping-Pong Model



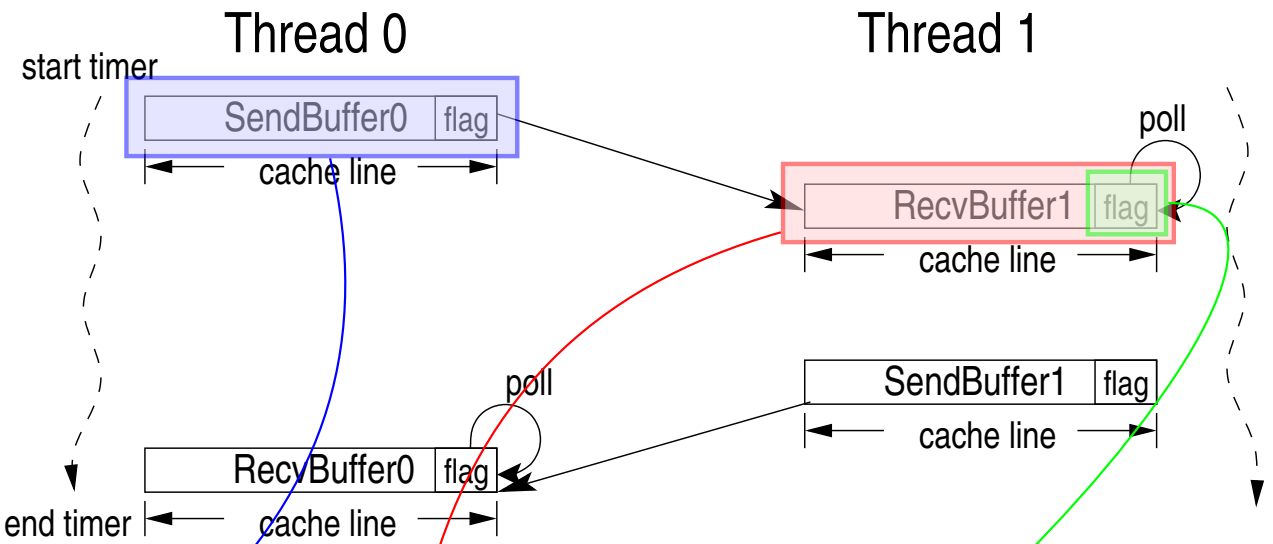
$$\mathcal{T}_1 = R_{L,\mathcal{S}_s} + R_{R,\mathcal{S}_r} + R_{R,M} + O = R_L + 2R_R + O$$

The Single-line Ping-Pong Model



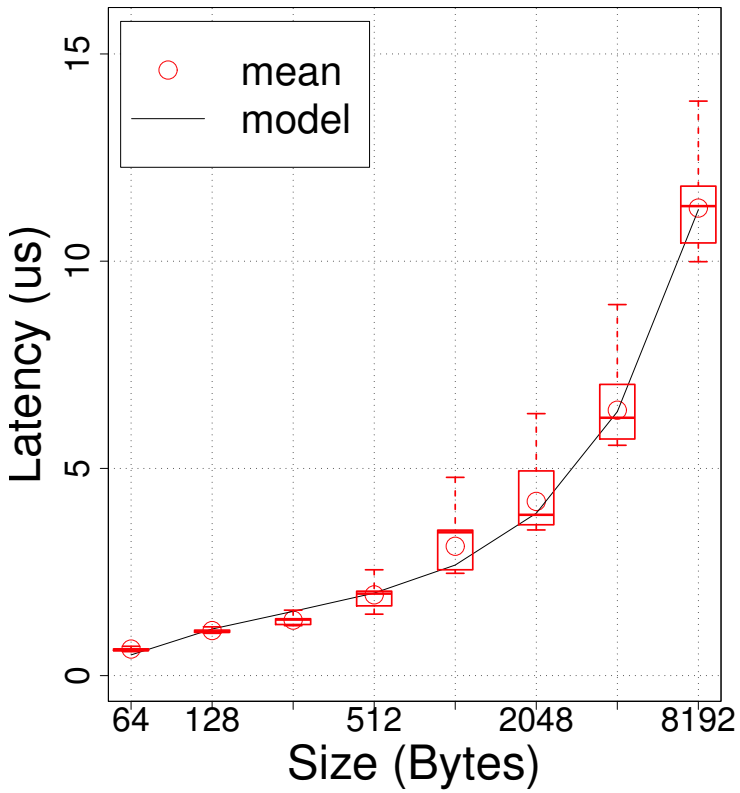
$$\mathcal{T}_1 = R_{L,\mathcal{S}_s} + R_{R,\mathcal{S}_r} + R_{R,M} + O = R_L + 2R_R + O$$

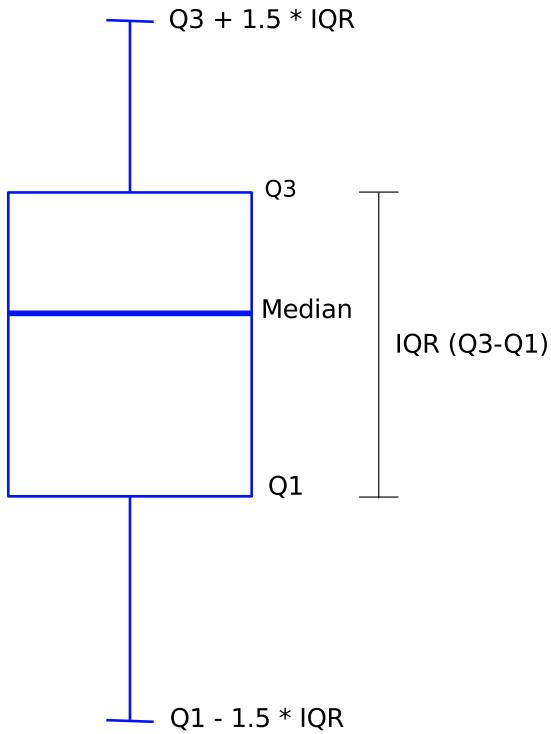
The Single-line Ping-Pong Model



$$\mathcal{T}_1 = R_{L,\mathcal{S}_s} + R_{R,\mathcal{S}_r} + R_{R,M} + O = R_L + 2R_R + O$$

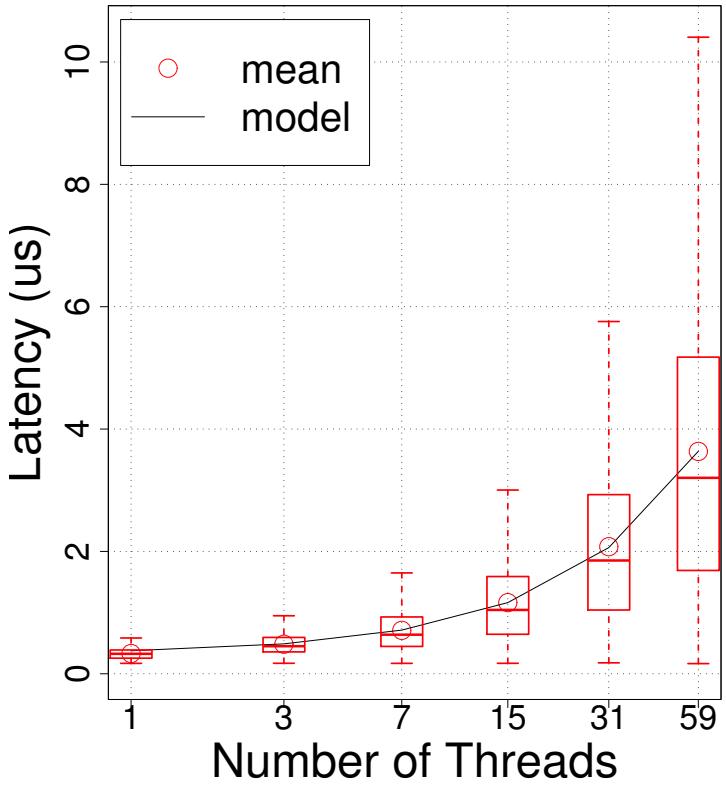
The Multi-line Ping-Pong Model

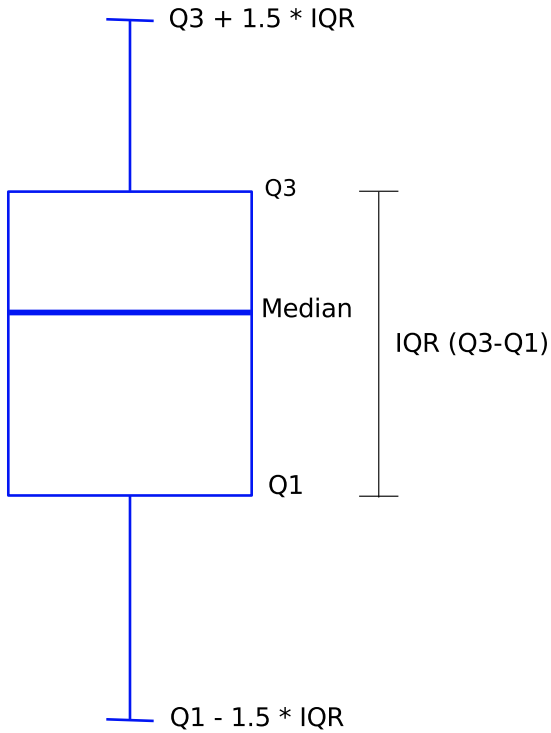




$$\mathcal{T}_N = o \cdot N + q - \frac{p}{N}$$

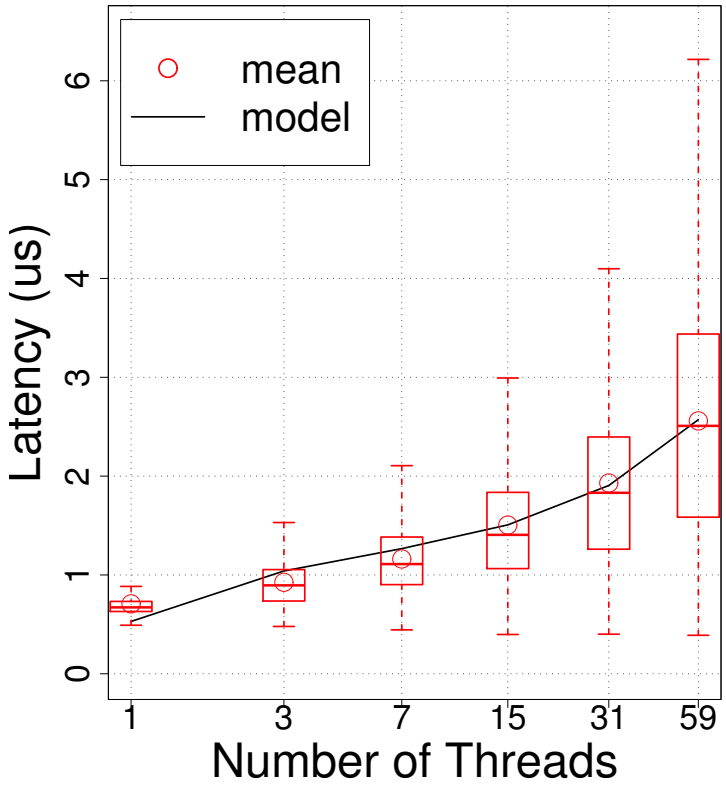
DTD Contention Model

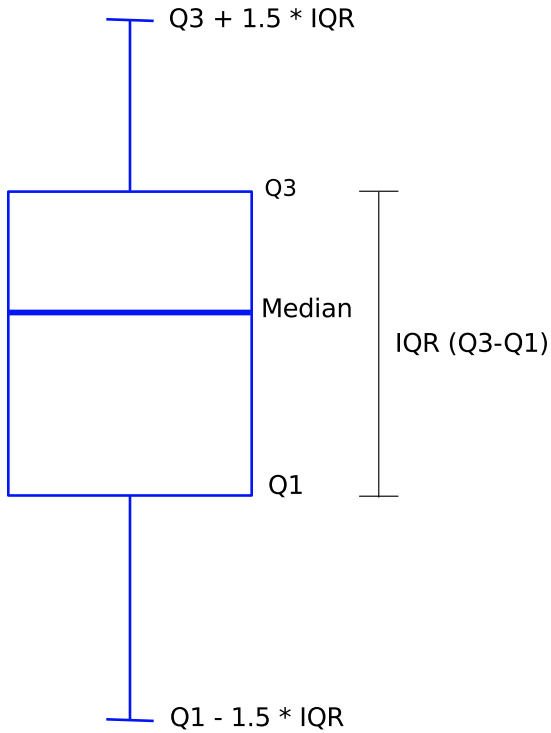




$$\mathcal{T}_C(n_{th}) = R_L + R_R + c \cdot (n_{th} - 1) = b + c \cdot n_{th}$$

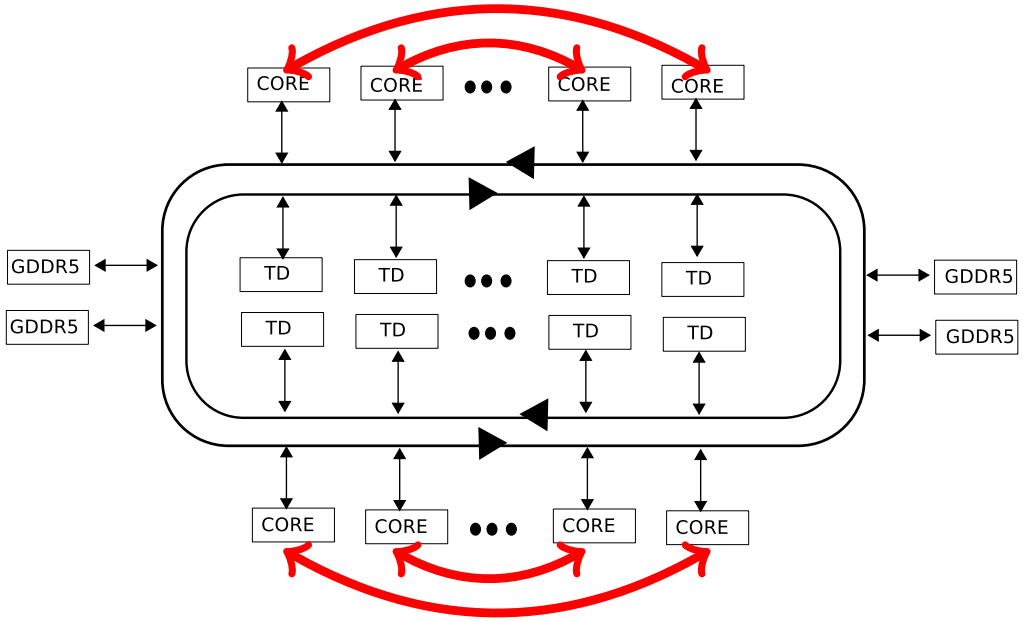
DTD Contention Model





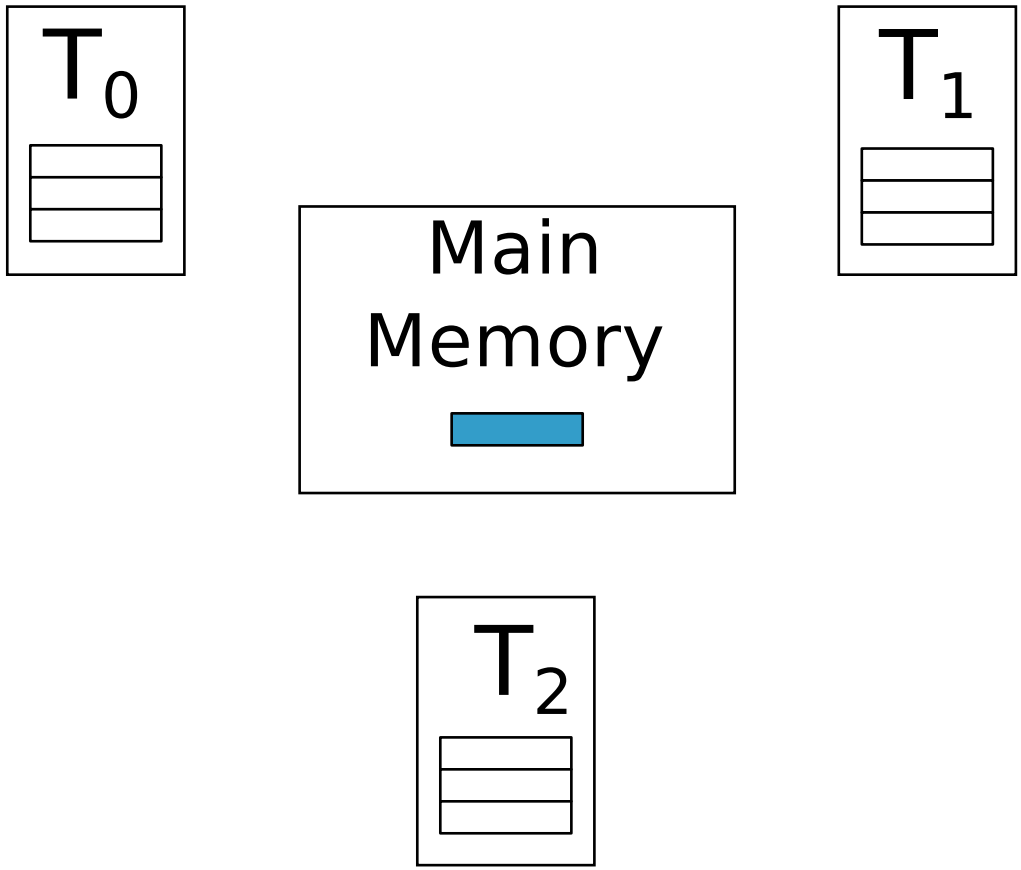
$$\mathcal{T}_C(n_{th}) = c \cdot n_{th} + b - \frac{a}{n_{th}}$$

Ring Congestion

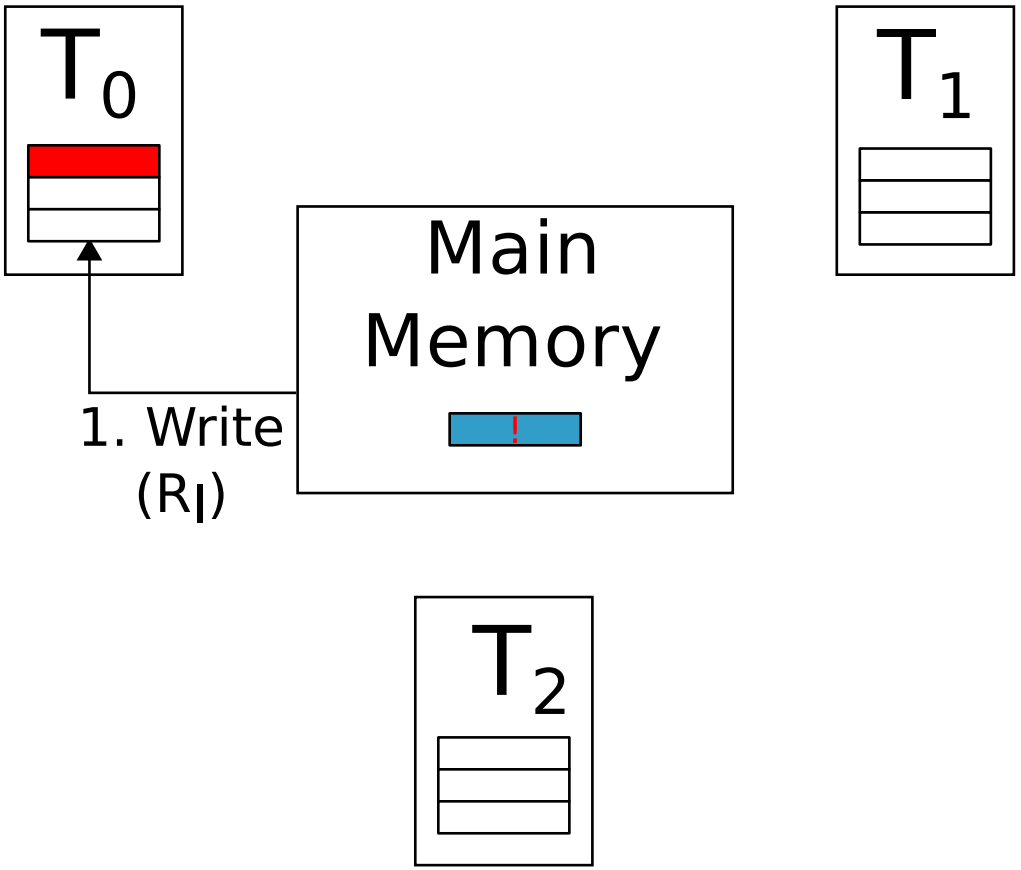


- Benchmarking using several interleaved ping-pongs.
- Our results showed no congestion derived from having several communicating pairs accessing diferent memory addresses.

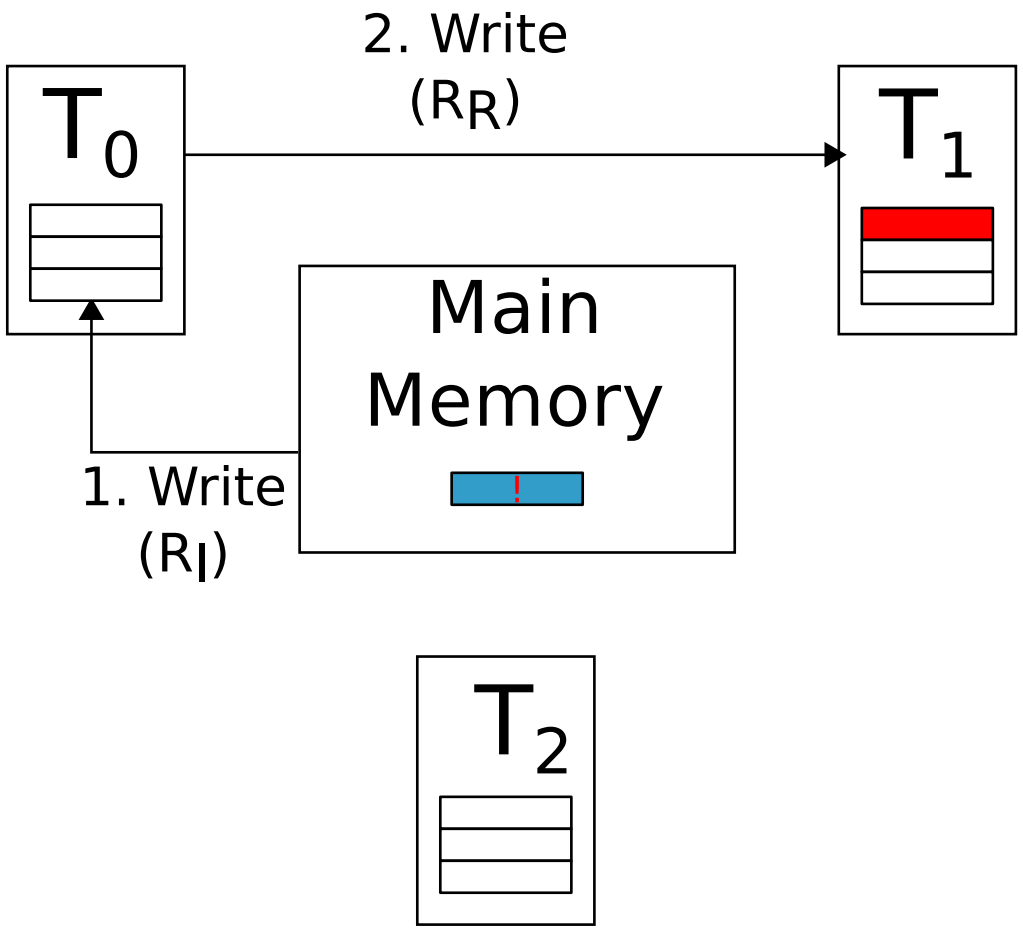
Thread Interference



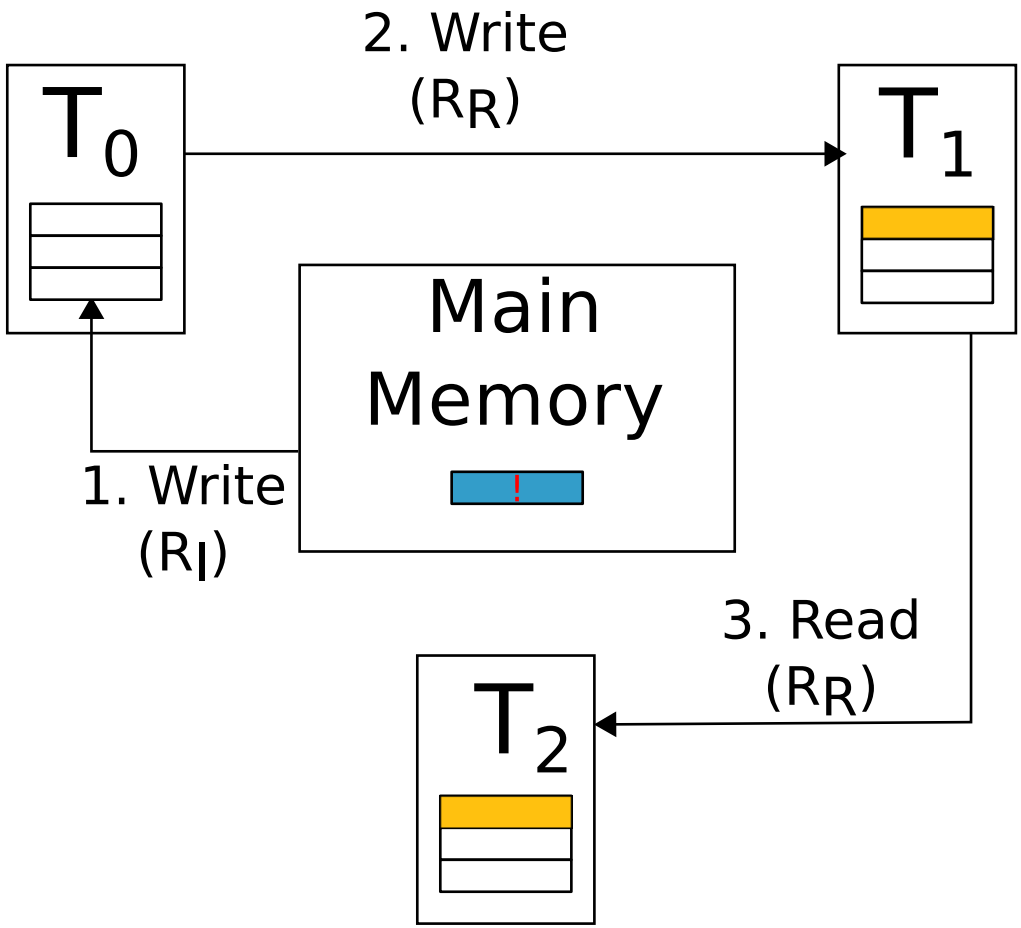
Thread Interference



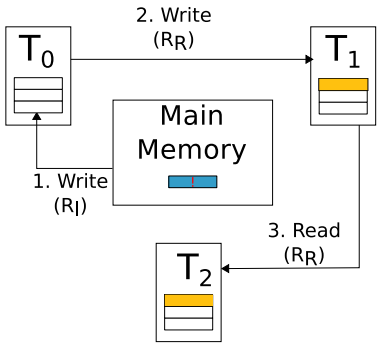
Thread Interference

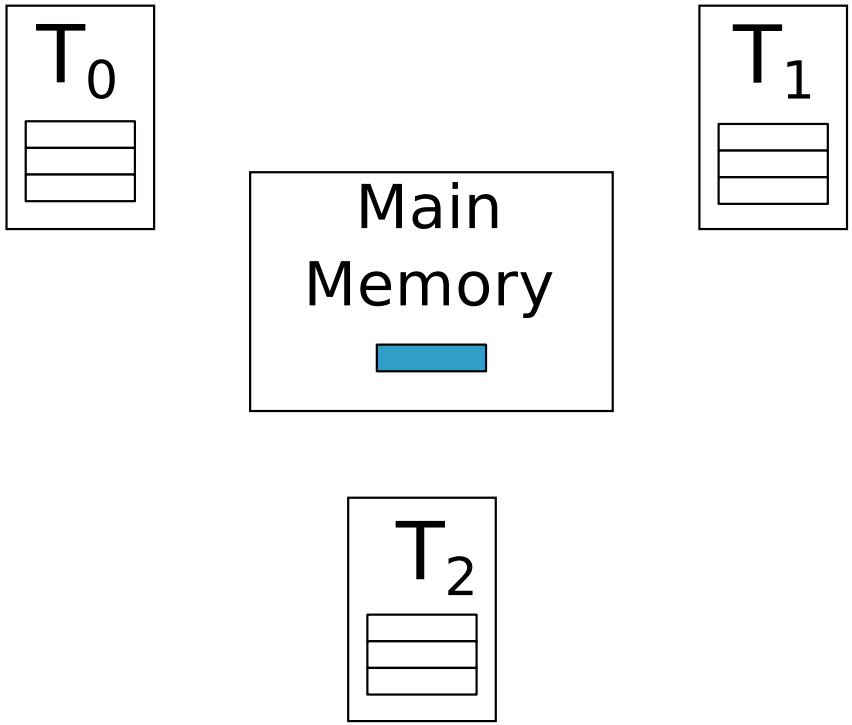


Thread Interference

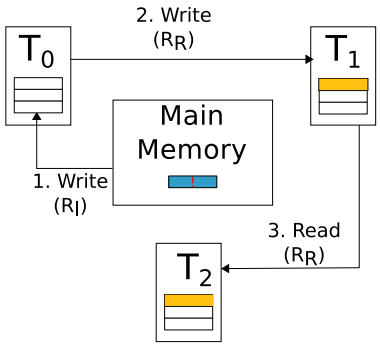


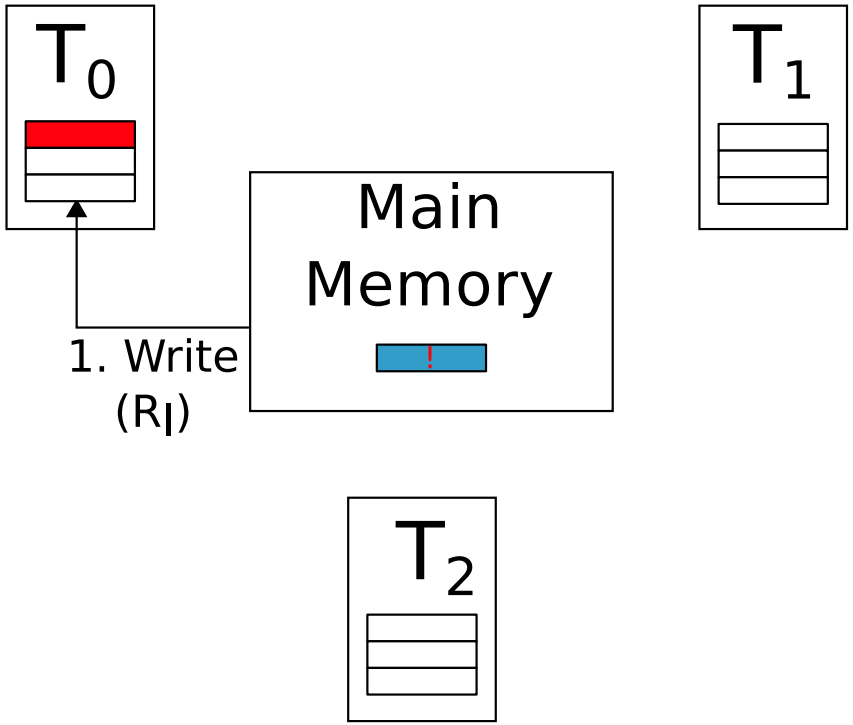
Thread Interference



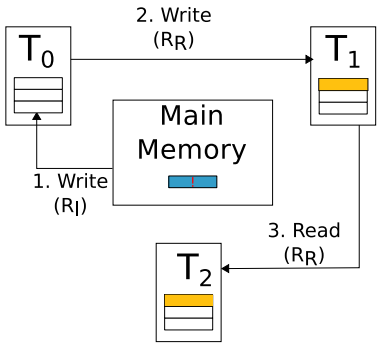


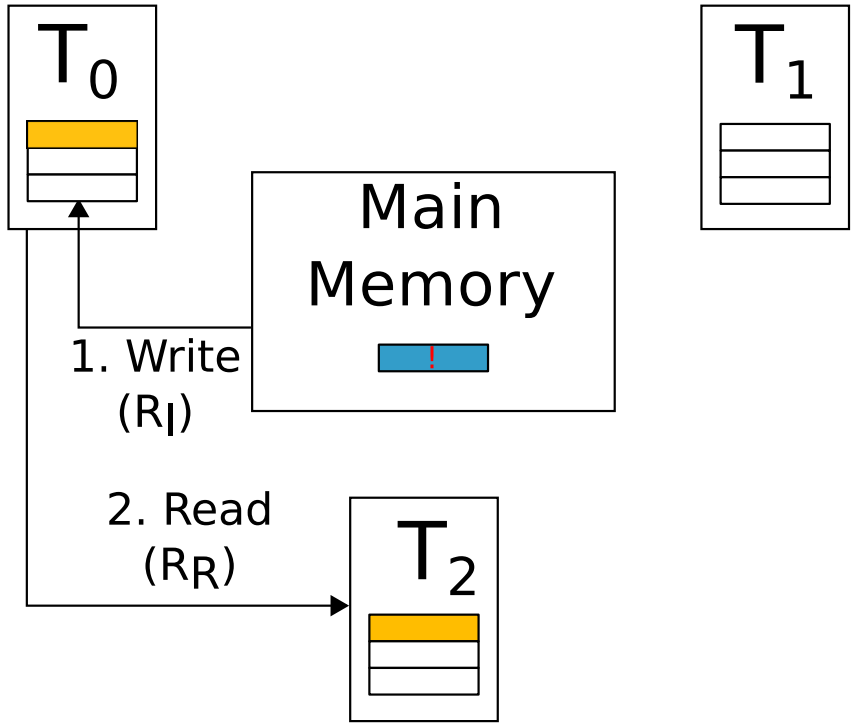
Thread Interference



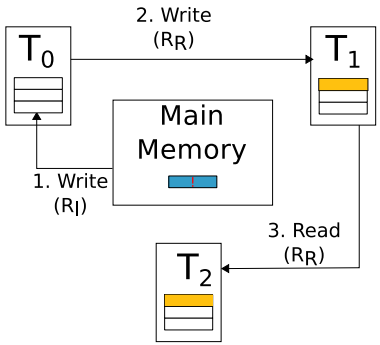


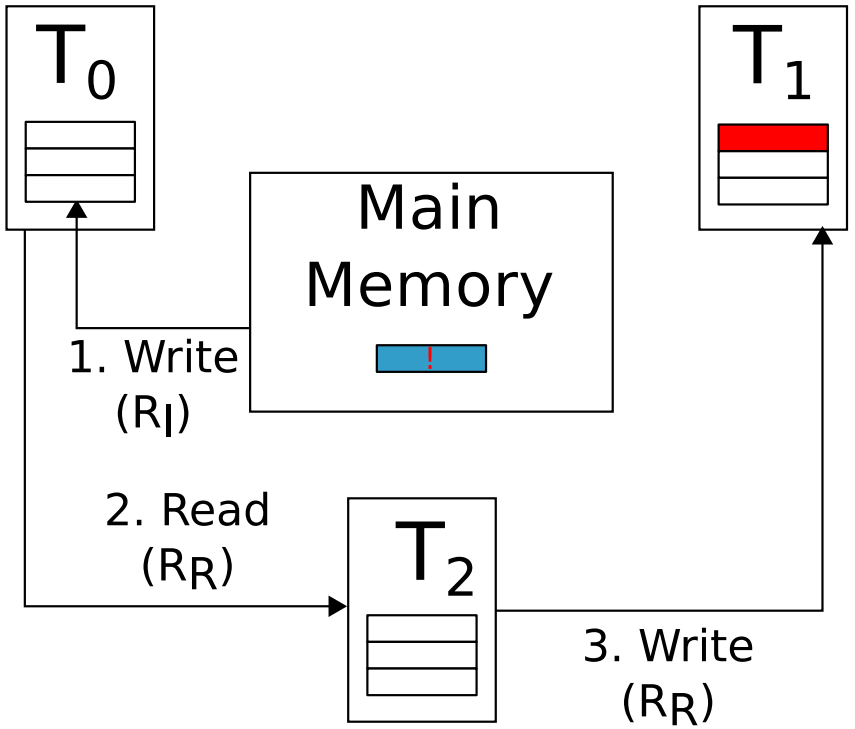
Thread Interference



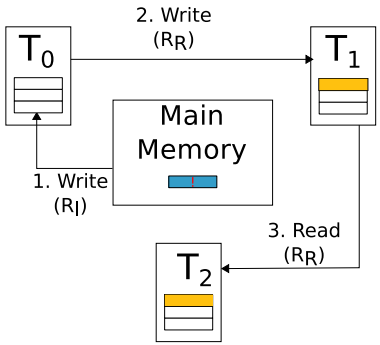


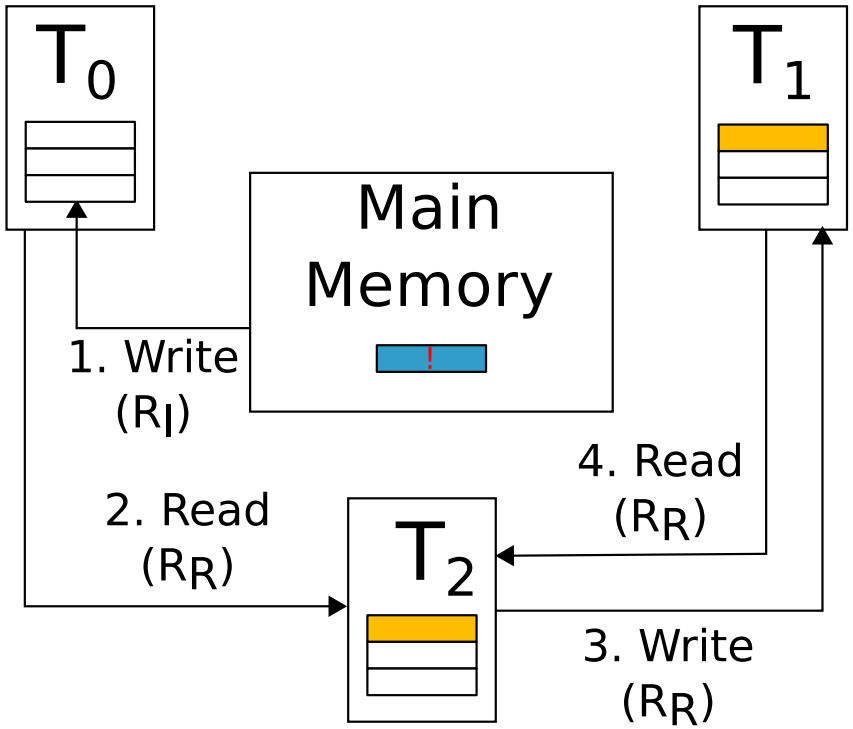
Thread Interference



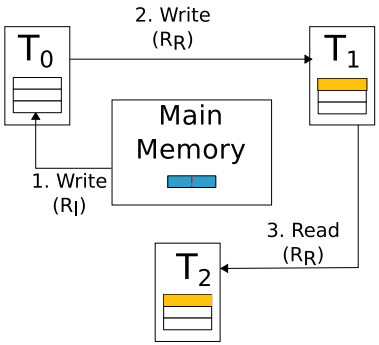


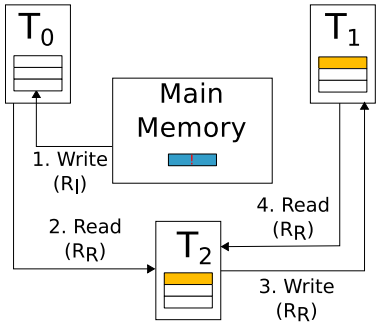
Thread Interference

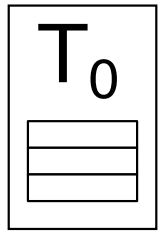


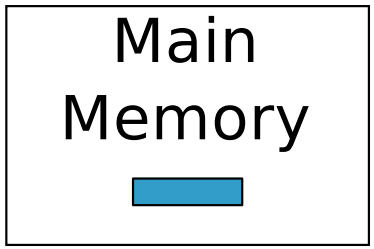


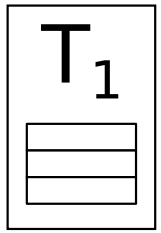
Thread Interference

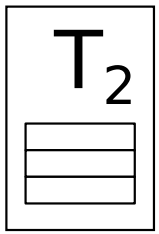




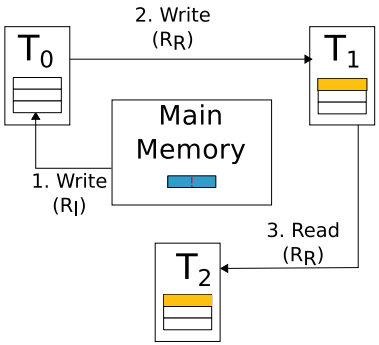


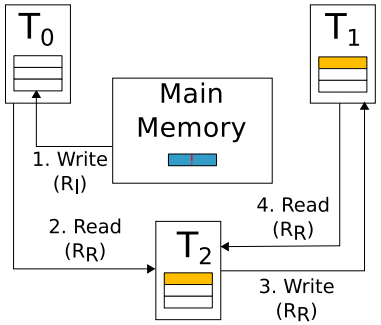


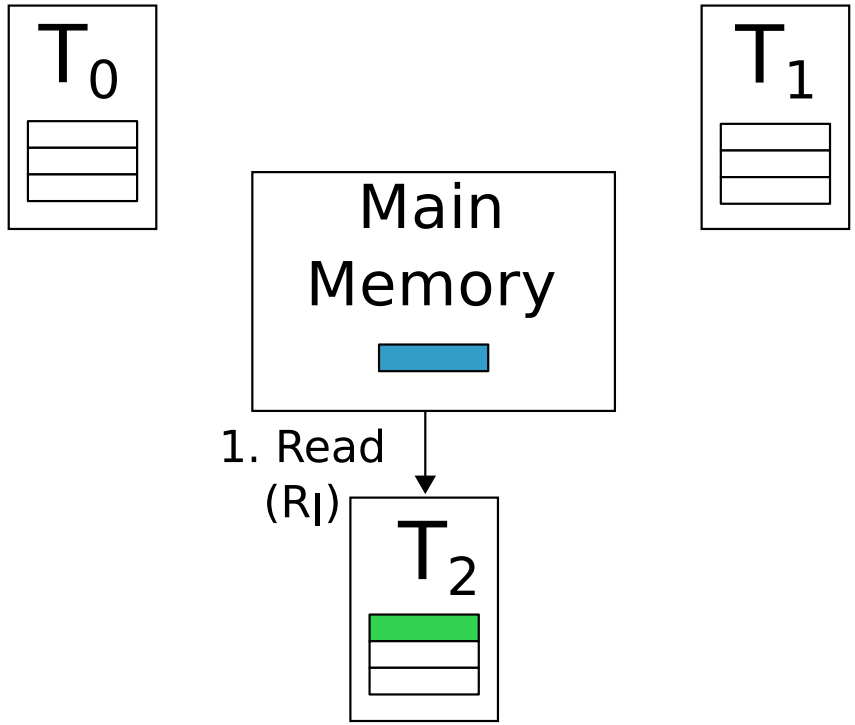




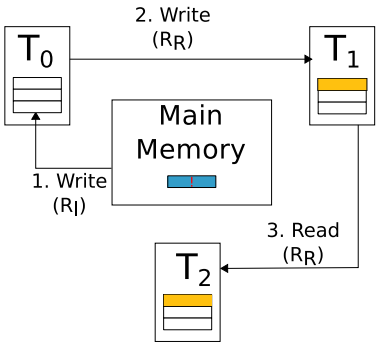
Thread Interference

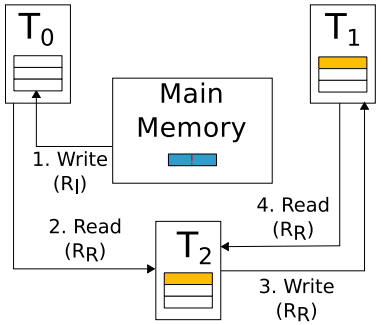


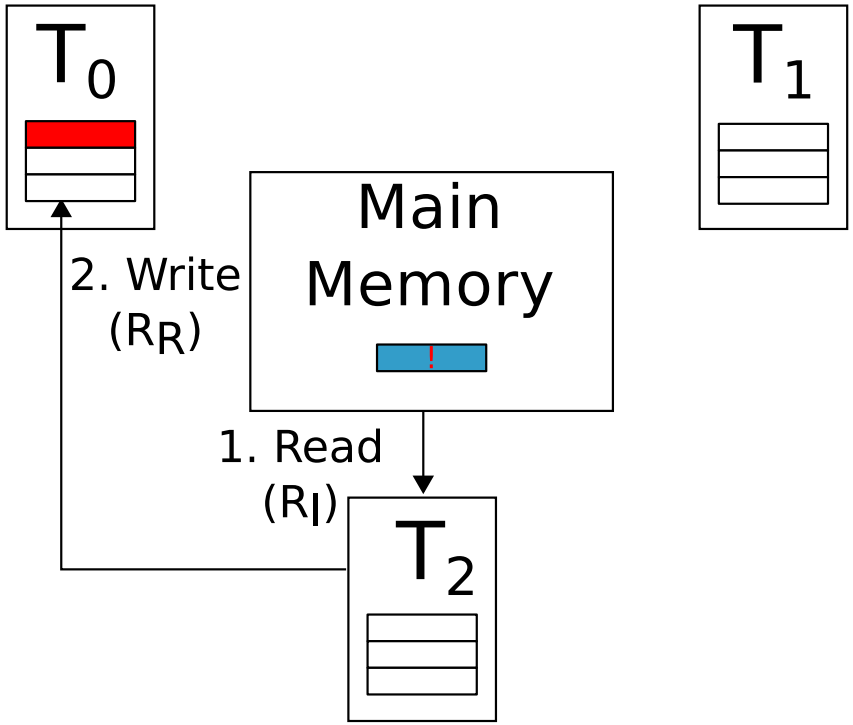




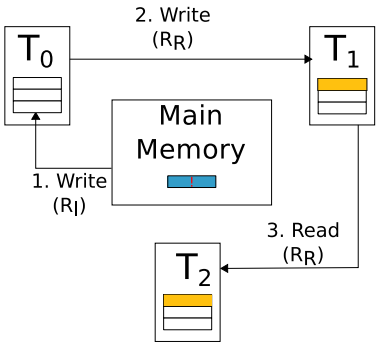
Thread Interference

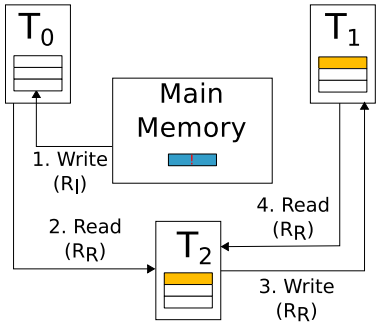


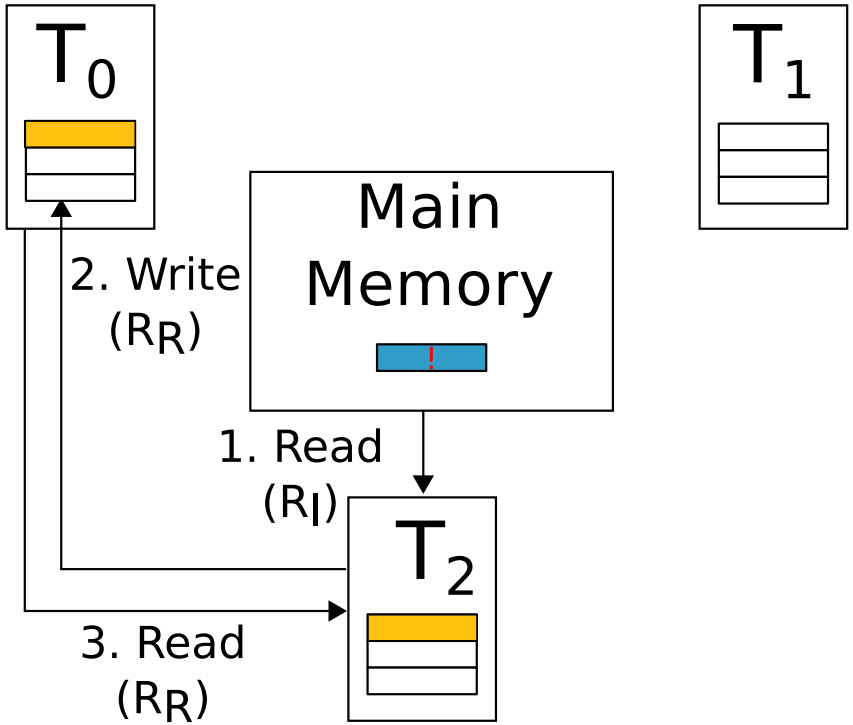




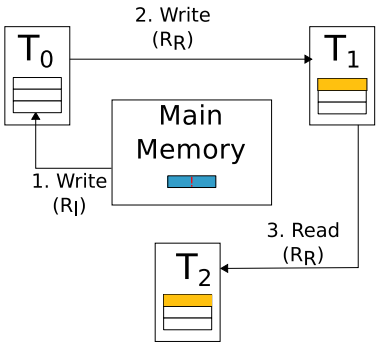
Thread Interference

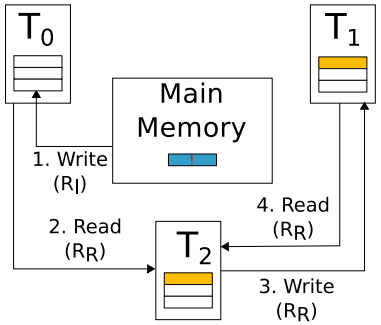


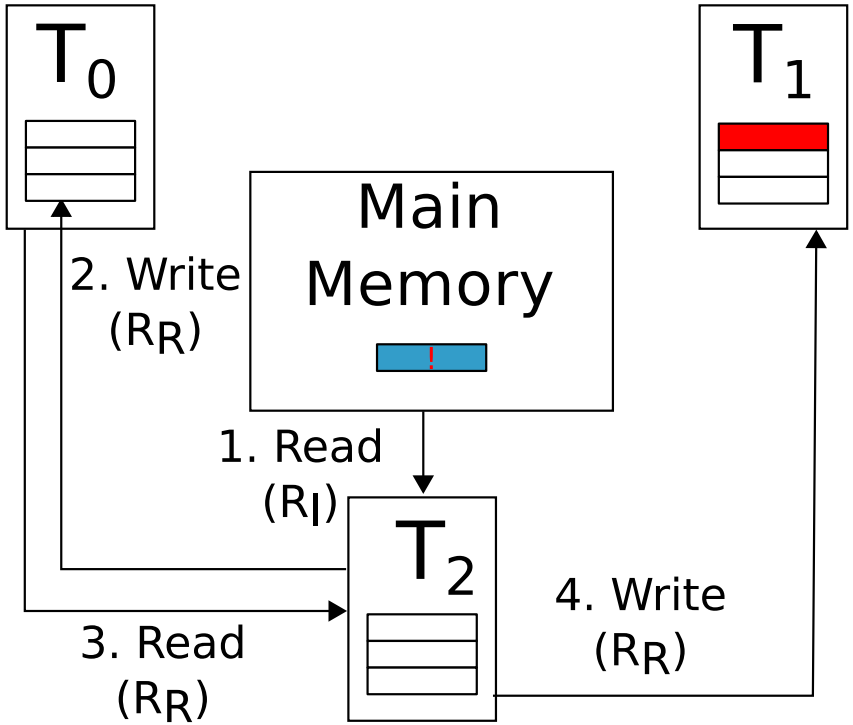




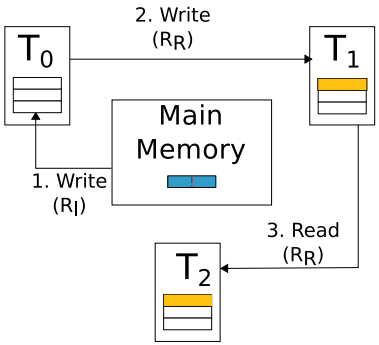
Thread Interference

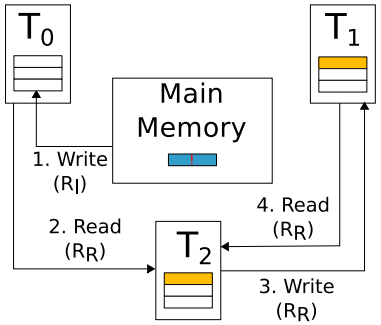


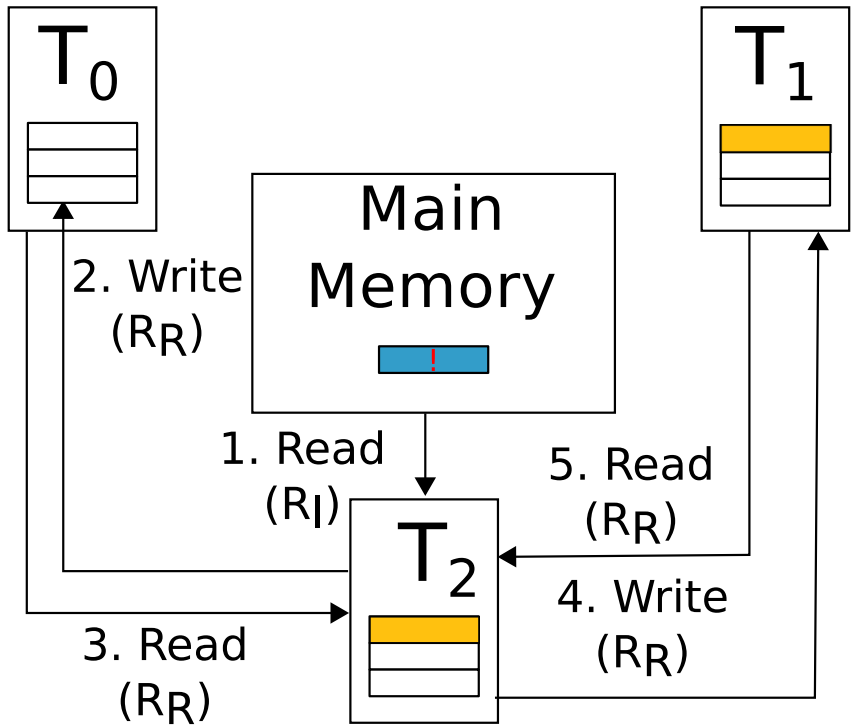




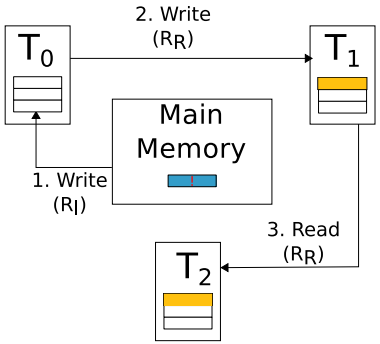
Thread Interference

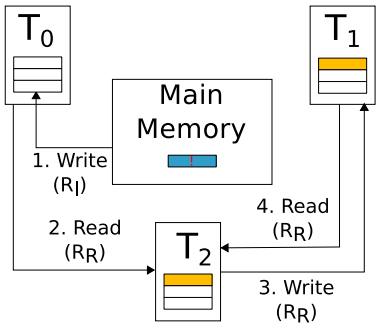


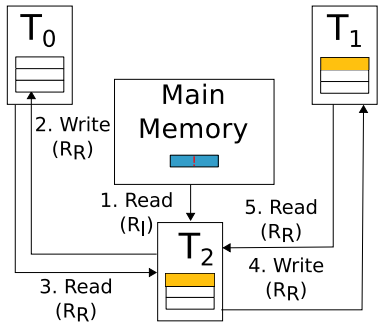




Thread Interference



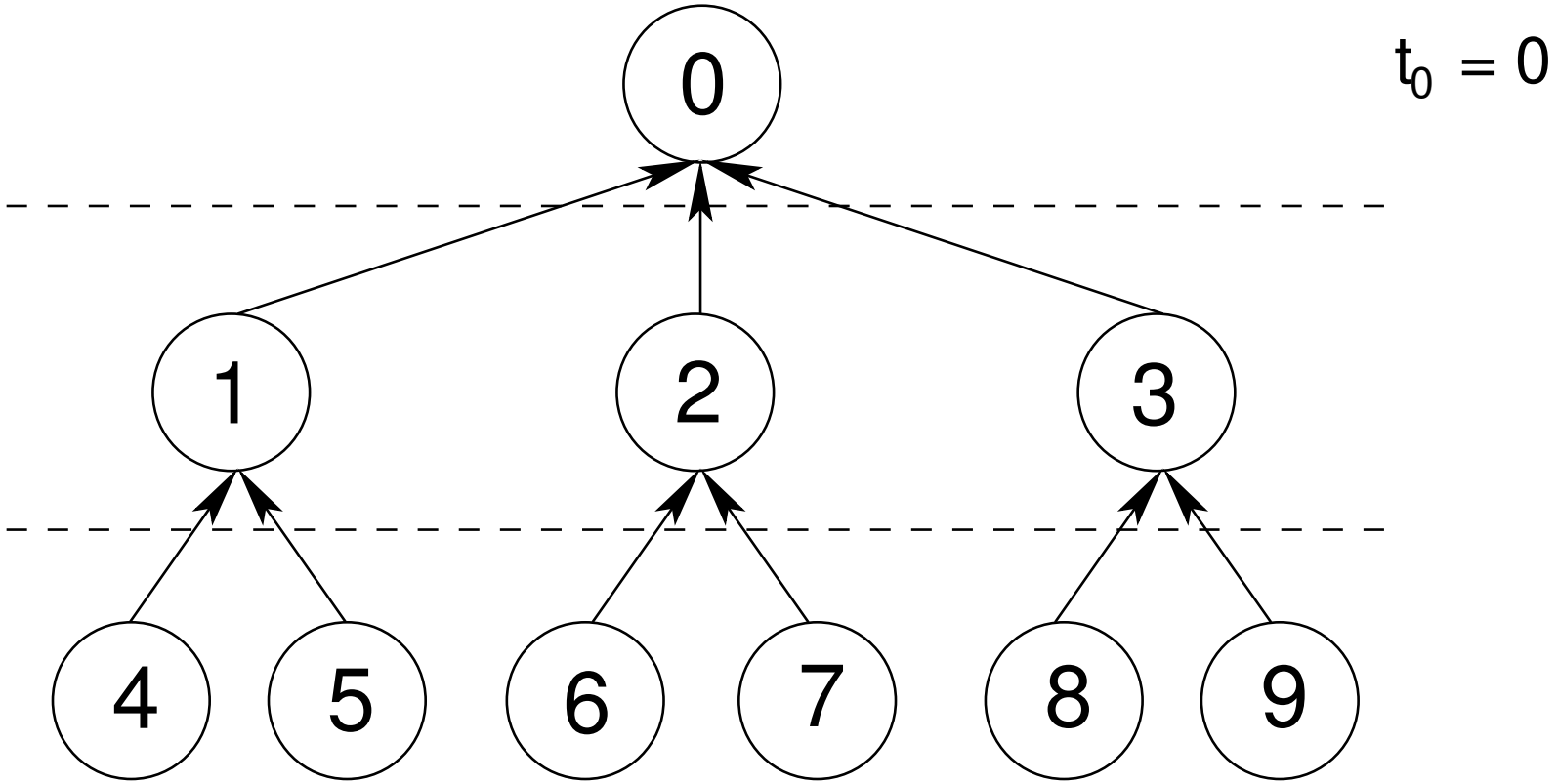




SOLUTION: Min-Max Models.

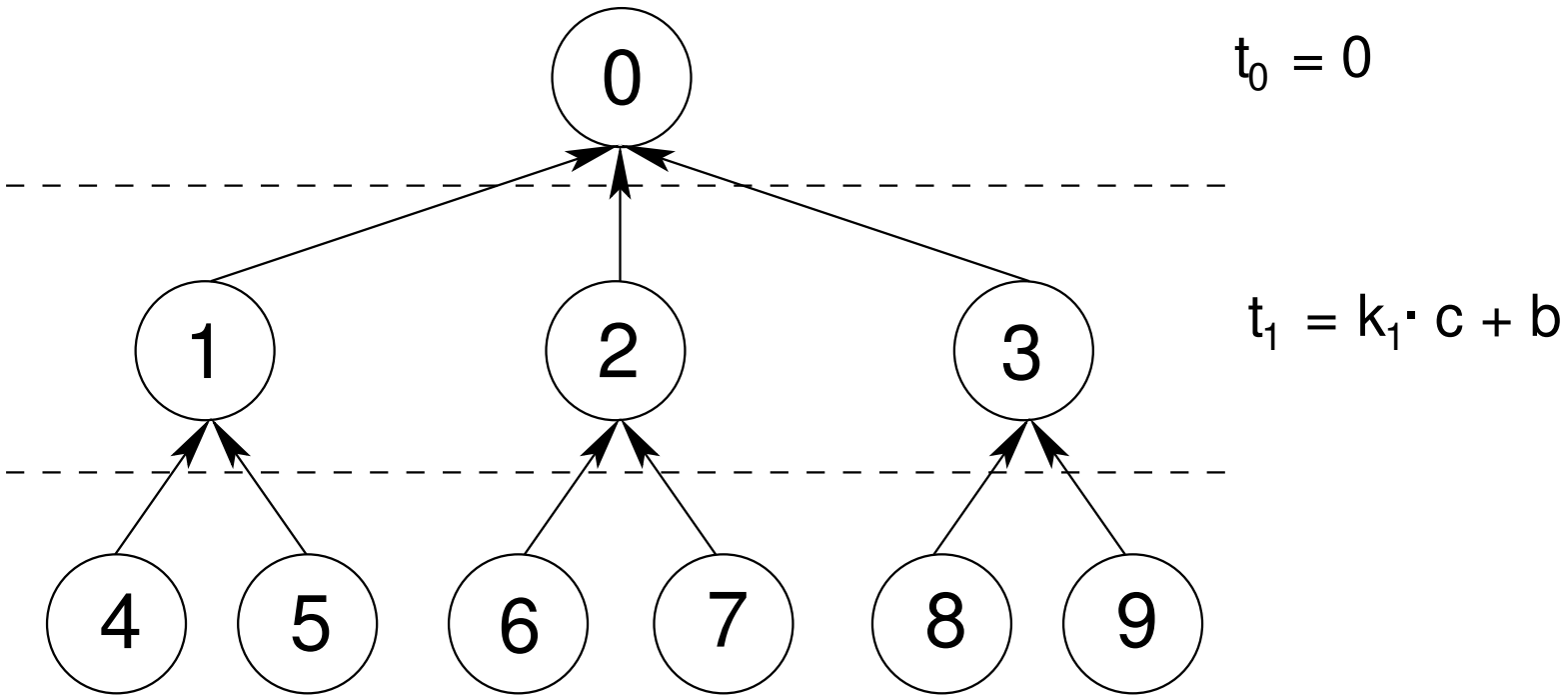
Small Broadcast

- Receiver-driven approach
- Parameters: $k_1 = 3$, $k_2 = 2$, $d = 2$



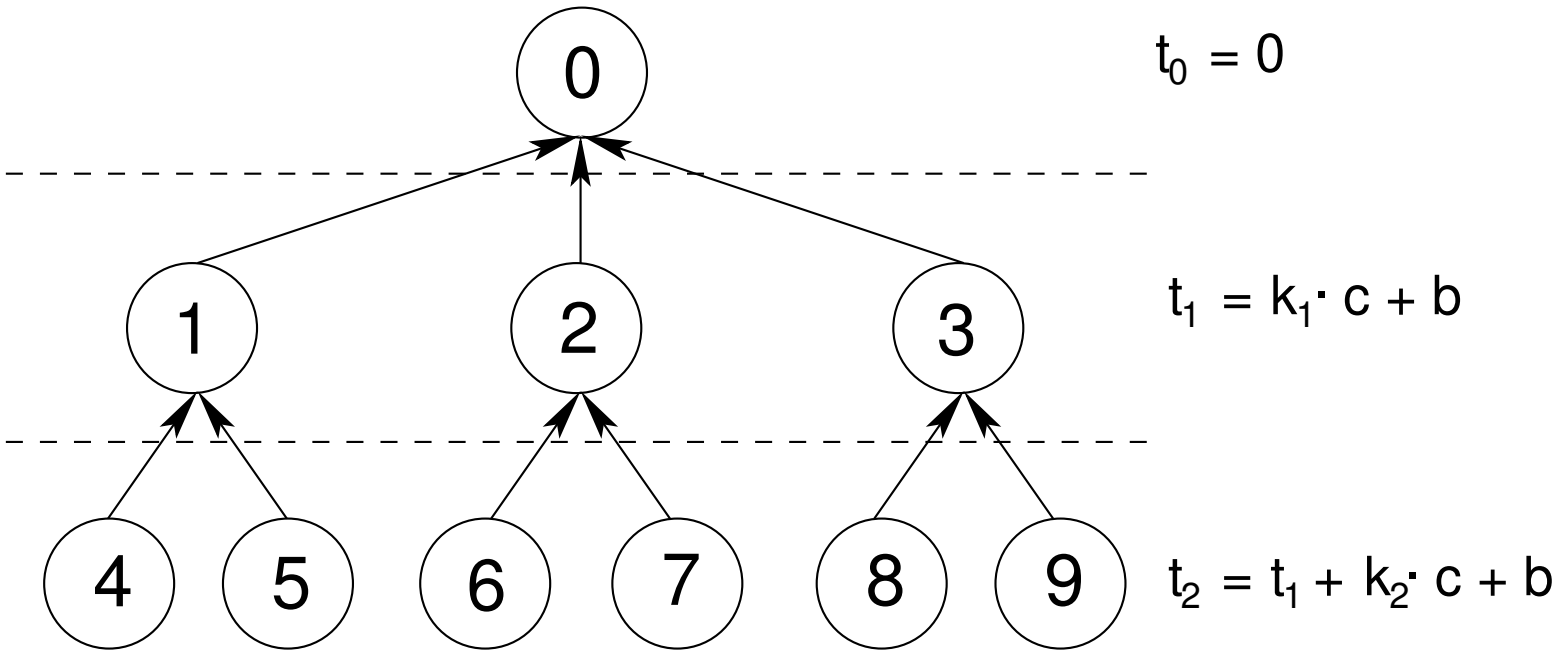
Small Broadcast

- Receiver-driven approach
- Parameters: $k_1 = 3$, $k_2 = 2$, $d = 2$



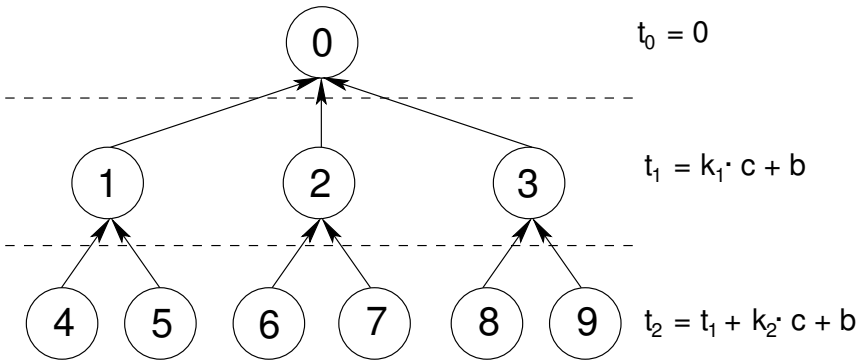
Small Broadcast

- Receiver-driven approach
- Parameters: $k_1 = 3$, $k_2 = 2$, $d = 2$



Small Broadcast

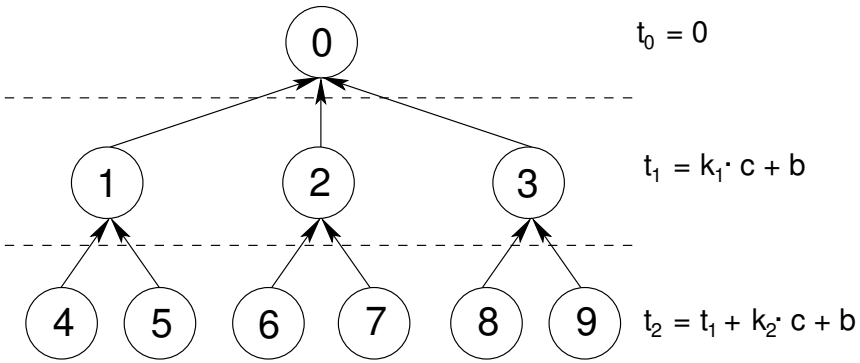
- Receiver-driven approach
- Parameters: $k_1 = 3$, $k_2 = 2$, $d = 2$



$$\mathcal{T}_{tree} = \sum_{i=1}^{d} \mathcal{T}_C(k_i) = \sum_{i=1}^{d} (k_i \cdot c + b)$$

Small Broadcast

- Receiver-driven approach
- Parameters: $k_1 = 3$, $k_2 = 2$, $d = 2$



$$\mathcal{T}_{tree} = \sum_{i=1}^{d} \mathcal{T}_C(k_i) = \sum_{i=1}^{d} (k_i \cdot c + b)$$

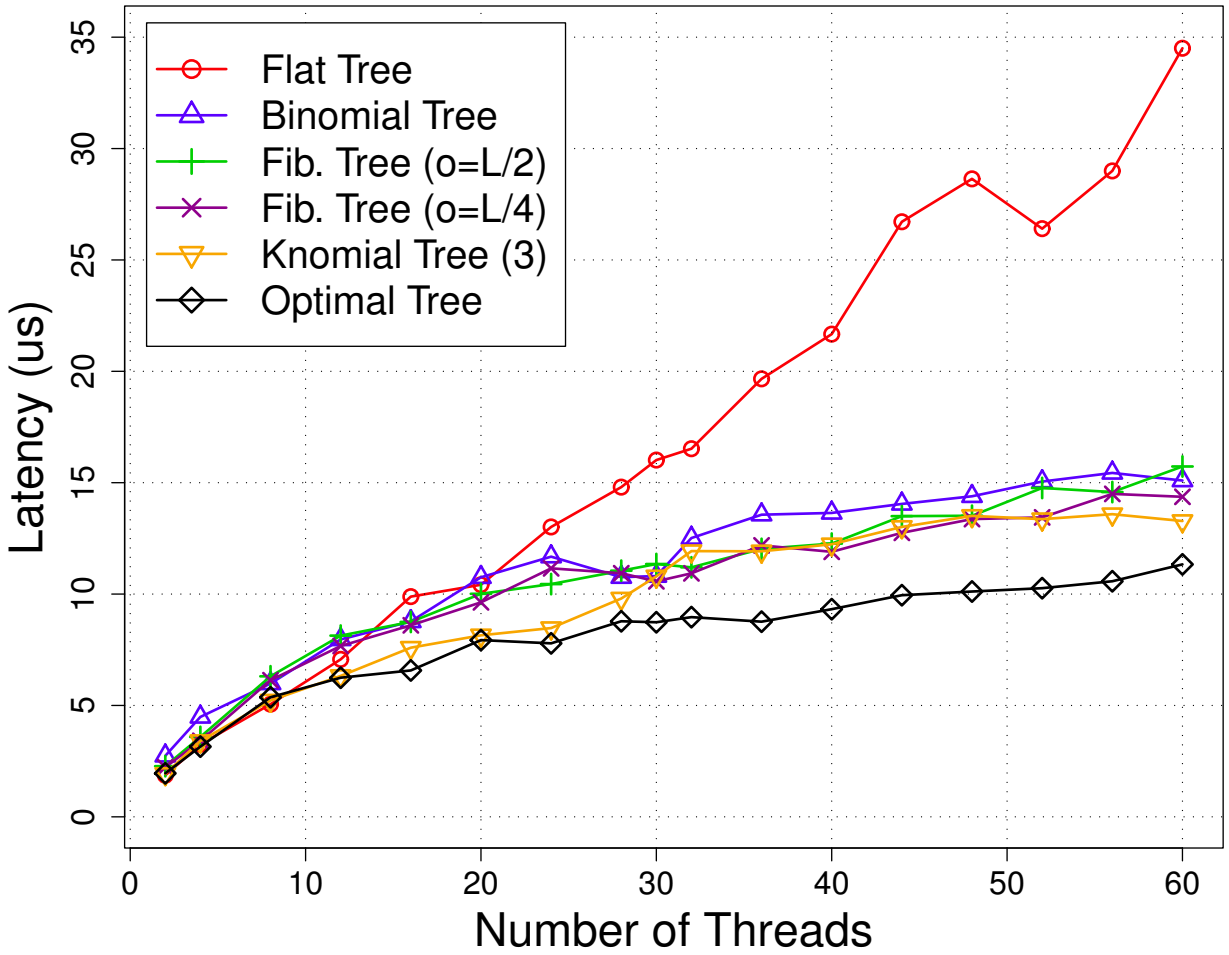
$$\mathcal{T}_{sbcast} = \min_{d,k_i} \left(\mathcal{T}_{fw} + \mathcal{T}_{tree} + \sum_{i=1}^{d} \mathcal{T}_{nb}(k_i + 1) \right)$$

$$N \leq 1 + \sum_{i=1}^{d} \prod_{j=1}^{i} k_j, \quad \forall i < j, k_i \leq k_j$$

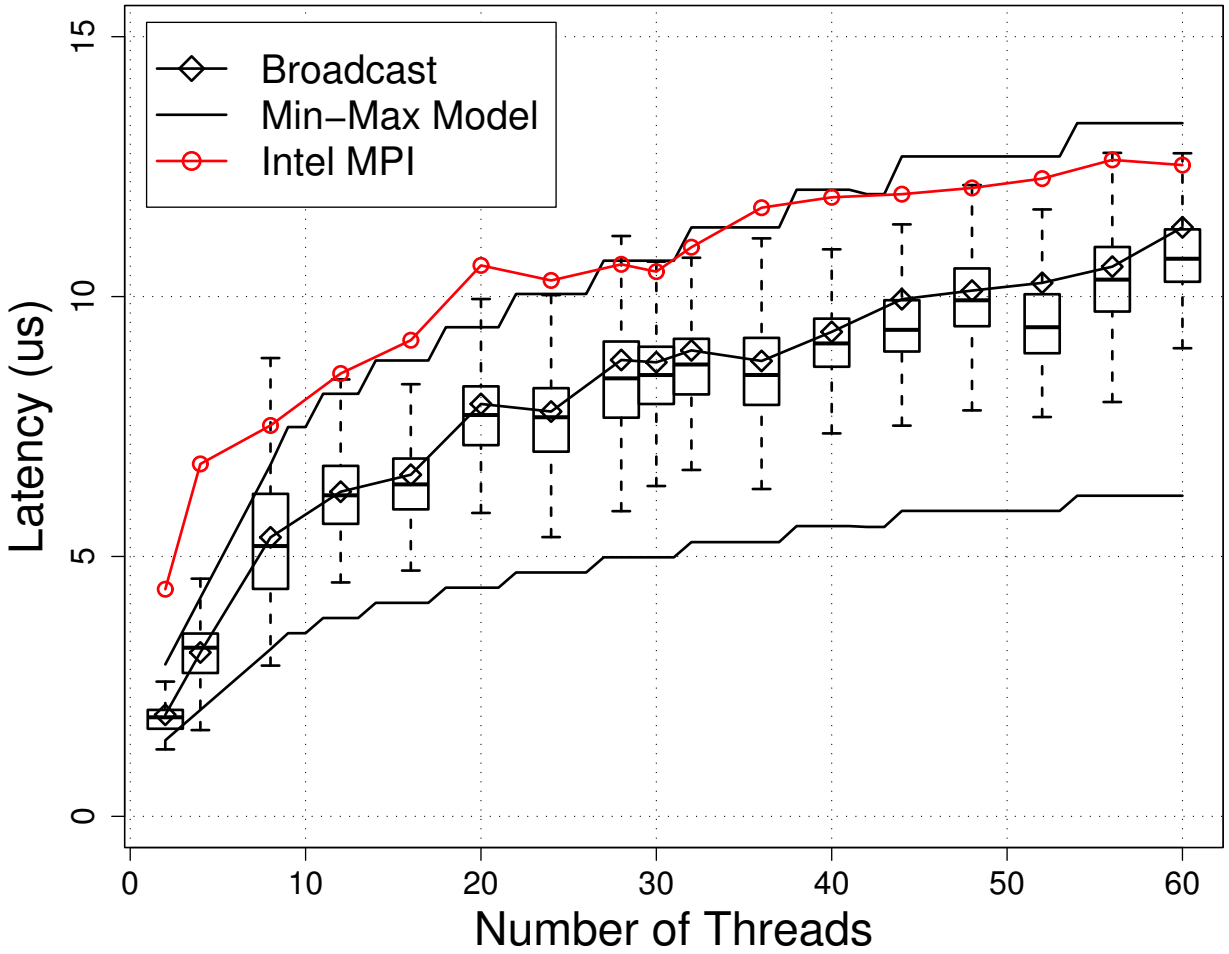
Other Operations

- Large Broadcast
  - Pipelined tree.
  - Flat Tree.
- Barrier Synchronization
  - Dissemination barrier.
- Small Reduction

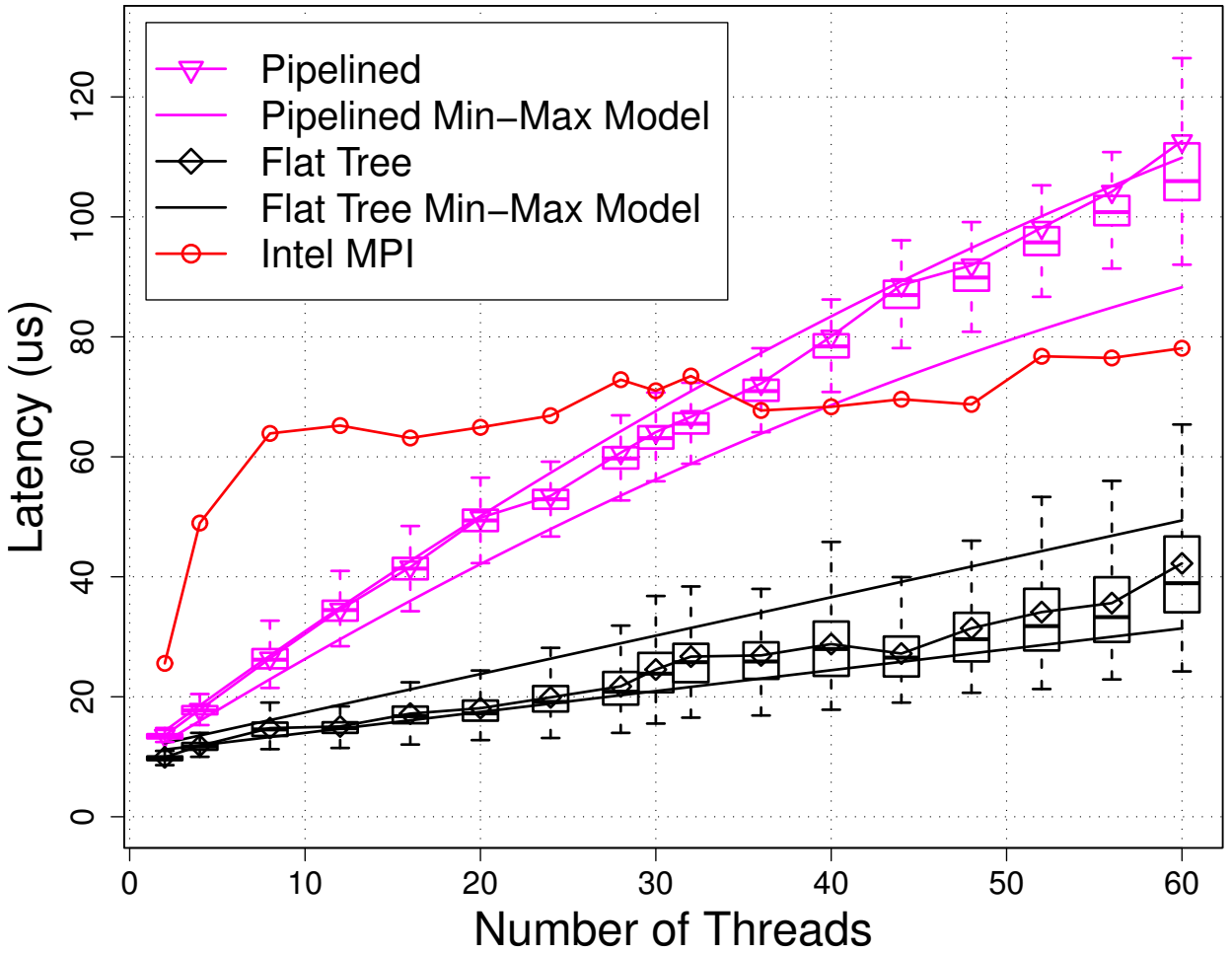
Small Broadcast Performance



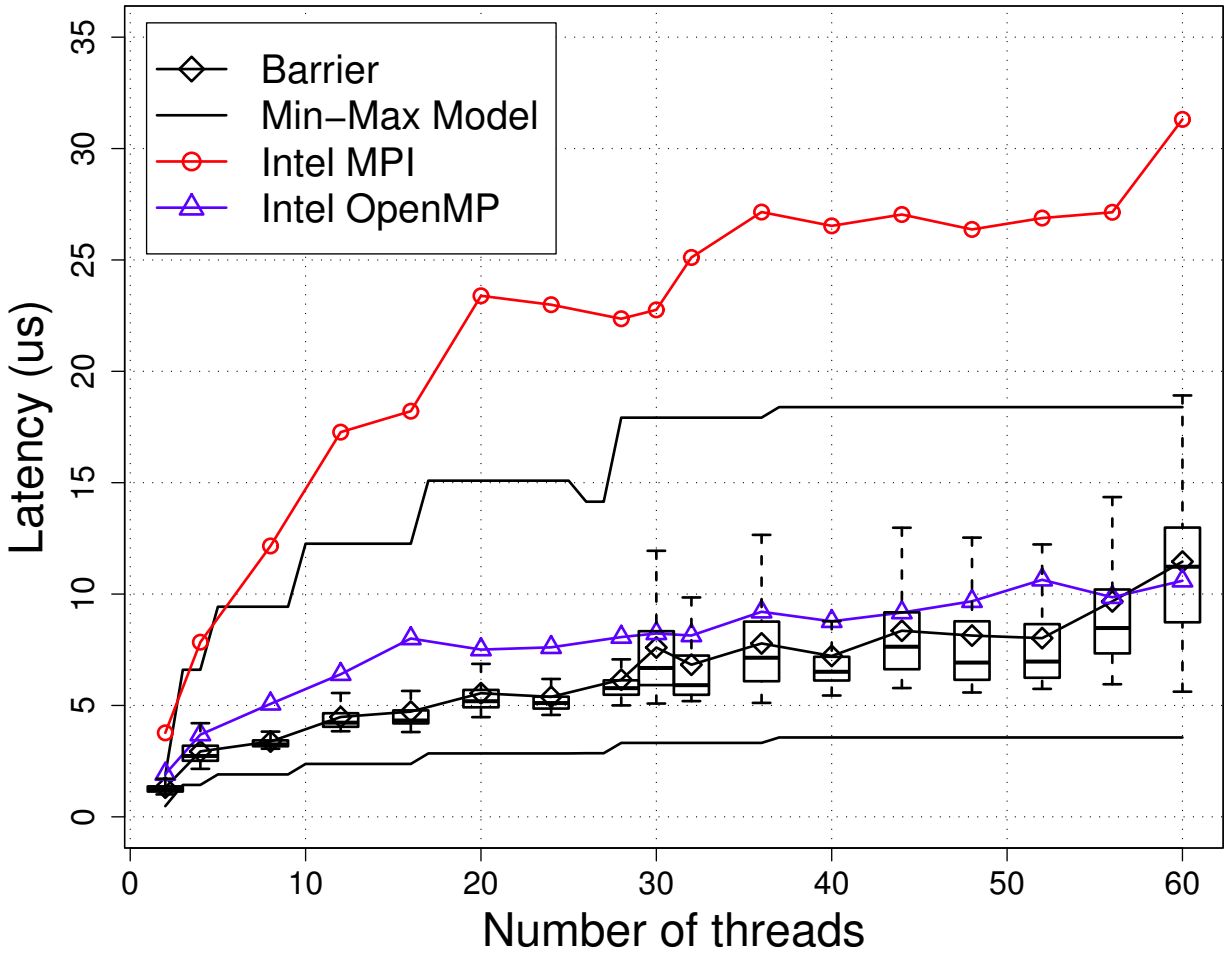
Small Broadcast Model



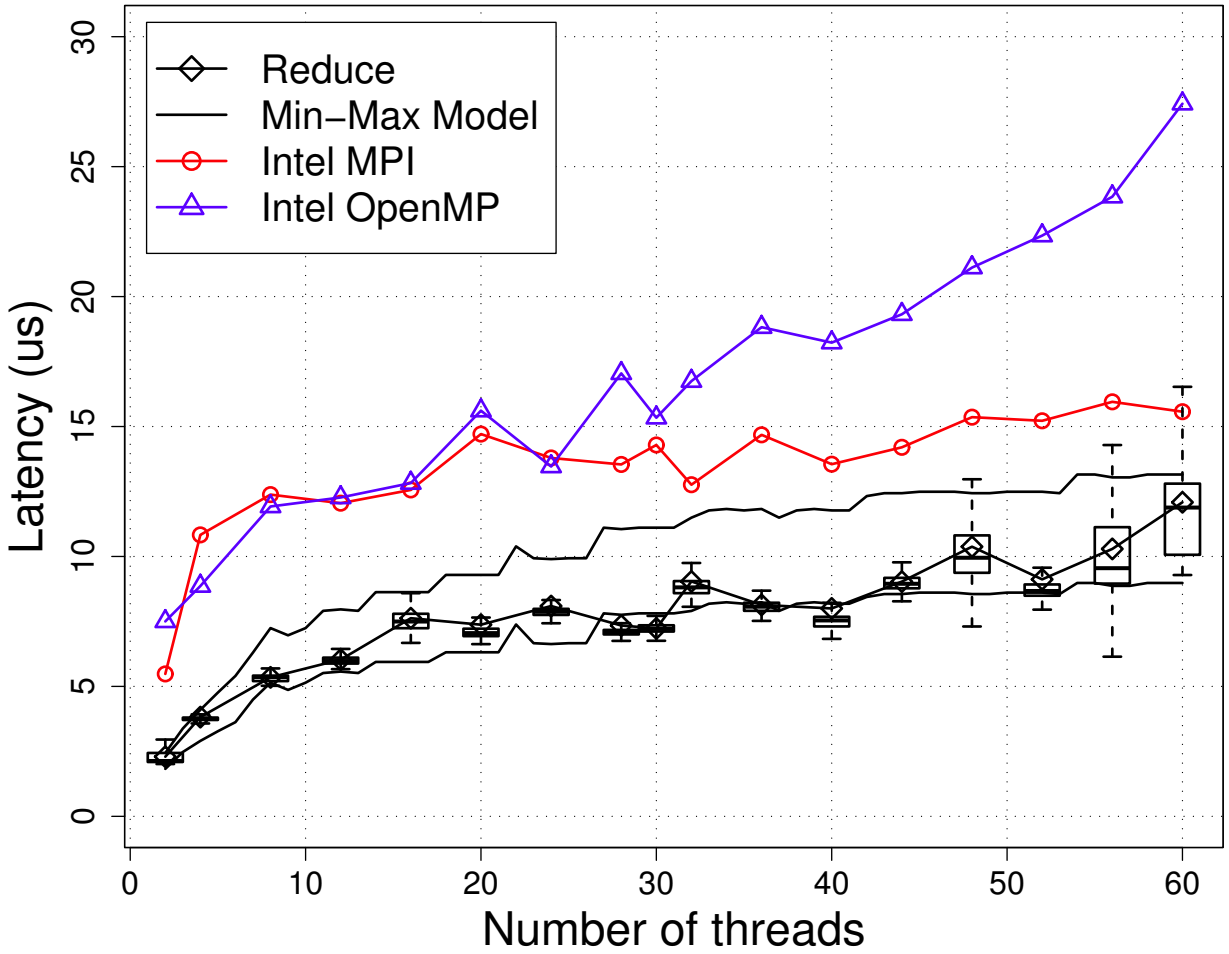
Large Broadcast Model



Barrier Synchronization



Small Reduction



- Optimizing for cache-coherency protocols is hard.
- Impact of interference caused by polling: min-max models. Is DRCA the solution?
- The model is able to guide algorithm design and development.
  - It does not provide a precise prediction but a range of possible performance.
  - The algorithms developed are up to 4.3 times faster than Intel MPI and OpenMP libraries.

Questions?

MODELING COMMUNICATION IN CACHE-COHERENT SMP SYSTEMS
A CASE STUDY WITH XEON PHI

HPDC 2013

Sabela Ramos
sramos@udc.es