Detecting communication in multi-cores

Andreas Sandberg[1] <andreas.sandberg@it.uu.se>
Stefanos Kaxiras[2] <kaxiras@ece.upatras.gr>

[1]Uppsala University

[2]University of Patras

2009-11-26

UPPSALA UNIVERSITET

Modeling cache states in multi-cores
Why?

Overall goals:

- Model all caches in cache coherent system
  - Mainly interested in communication
- Find communication hot-spots in multi-threaded applications
- Low on-line overhead

UPPSALA UNIVERSITET

Cache modeling
Approaches

- On-line simulation
  - Cachegrind (Valgrind)
  - GEMS (Simics)
  - CMP$im (Pin-based)
  - ...
- Off-line simulation
  - Trace driven
- Statistical models
  - StatCache
  - StatCache MP

UPPSALA UNIVERSITET

Cache modeling
Why statistical models?

- Does not require a full trace of the target
  - Low overhead for on-line phase (data acquisition)
  - Low storage requirements
  - Fast analysis
- Flexible
  - Cache parameters can be changed off-line

UPPSALA UNIVERSITET

What is StatCache?

StatCache is:

- A statistical model to calculate miss ratios in full-scale *single-threaded* applications.
- Based on memory reuse distances.
  - Reuse distances can be sampled.
  - Measuring reuse distances is easy (no need to keep track of unique accesses).

UPPSALA UNIVERSITET

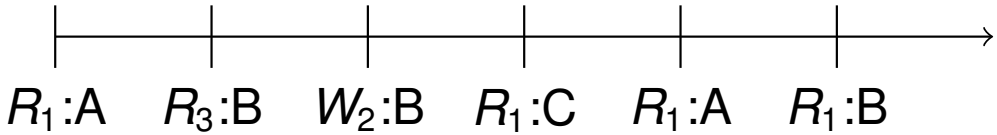
What is StatCache?
Sampling

The reuse distance characteristics of an application can be sampled. A sample consists of access pairs to the same cache line.

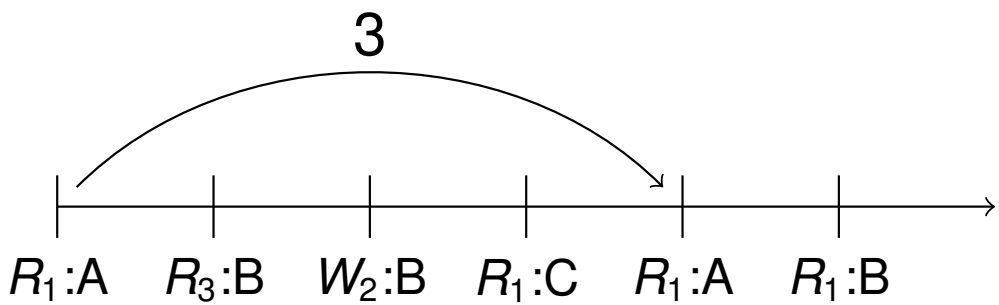
Each access pair is annotated with:

- The instruction that started the pair (PC1)
- The instruction that terminated the pair (PC2)
- The distance between the accesses (reuse distance)

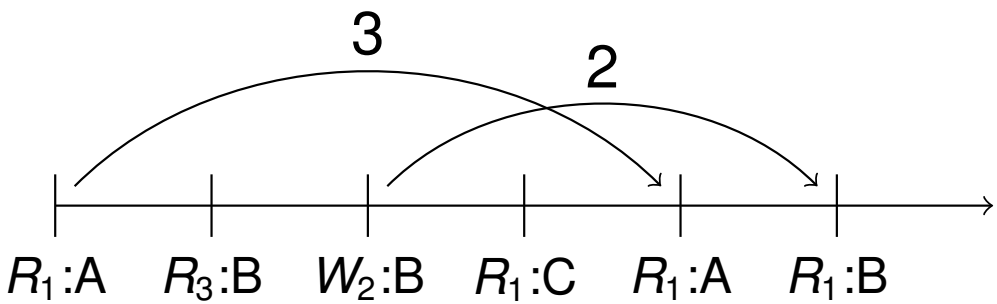
What is StatCache?
Sampling



What is StatCache?
Sampling



What is StatCache?
Sampling



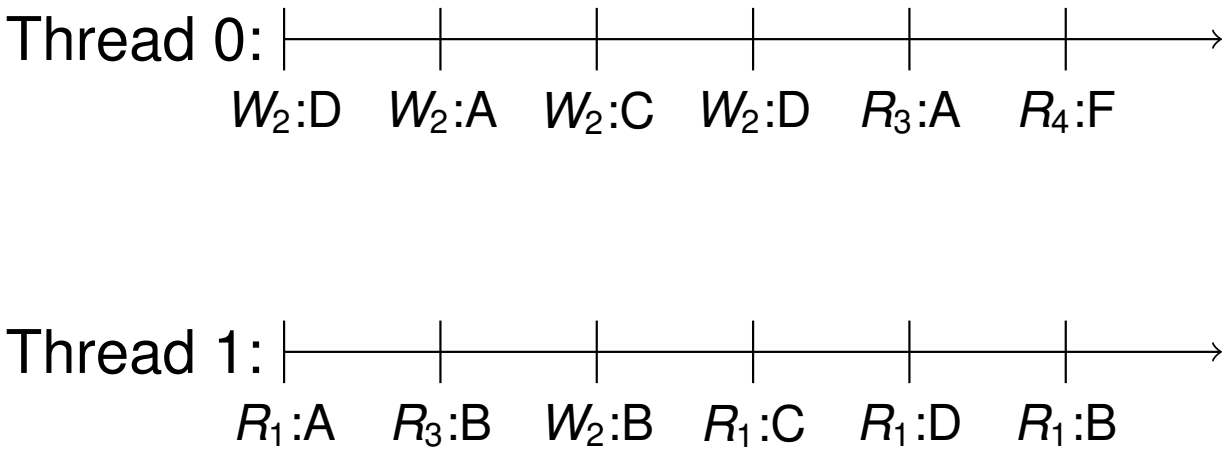
UPPSALA UNIVERSITET

Modeling cache states in multi-cores
Why?

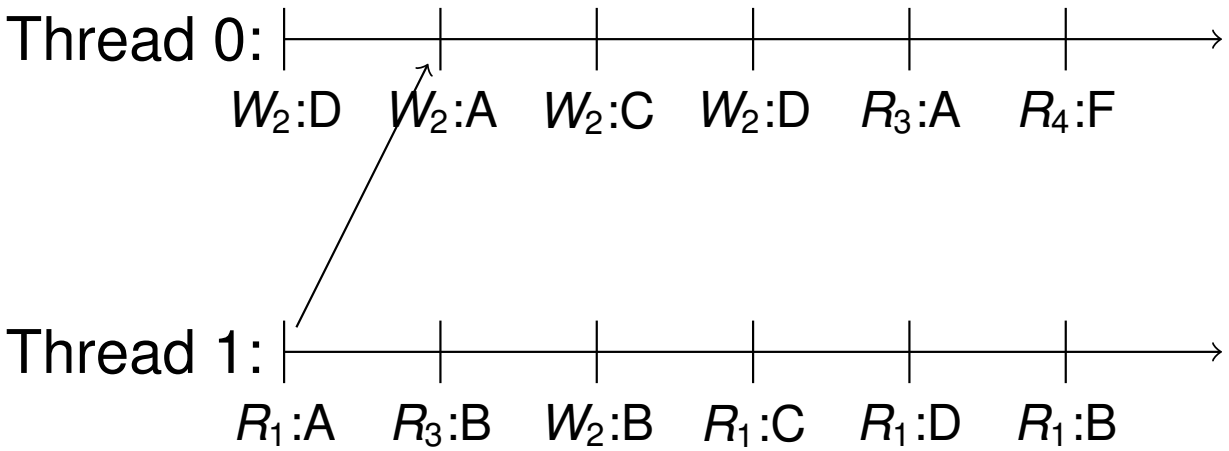
Overall goals:

- Model all caches in cache coherent system
  - Mainly interested in communication
- Find communication hot-spots in multi-threaded applications
- Low on-line overhead

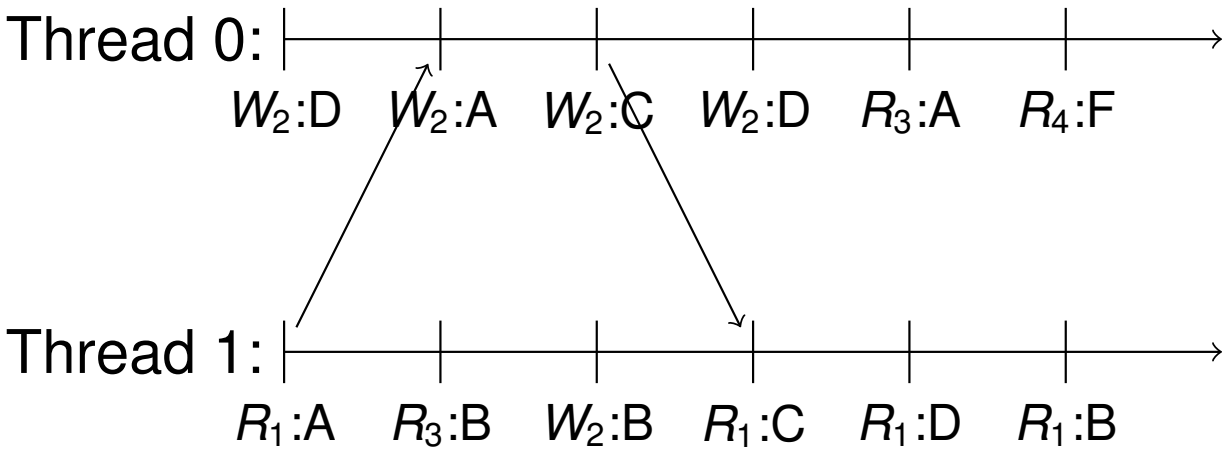
Modeling cache states in multi-cores
Sampling



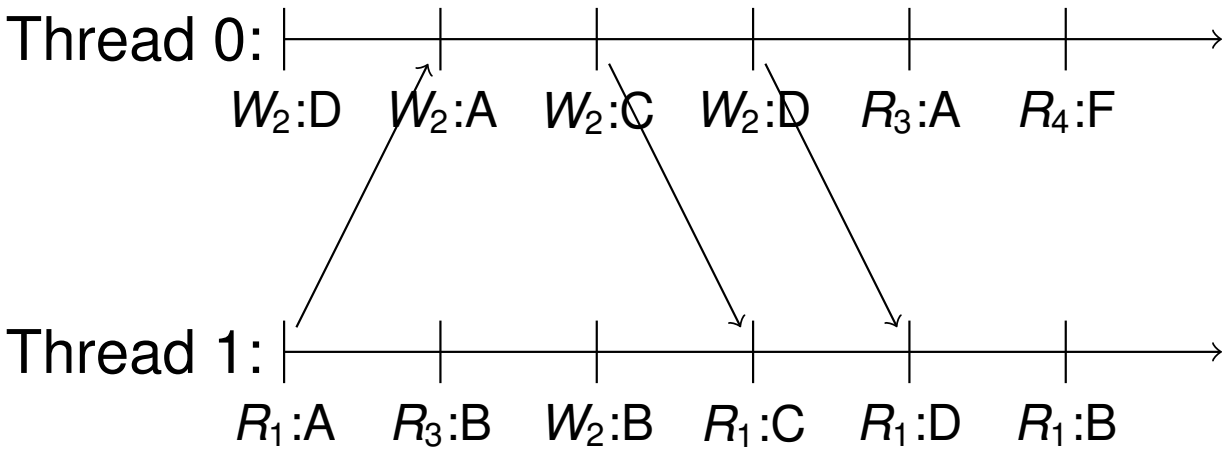
Modeling cache states in multi-cores
Sampling



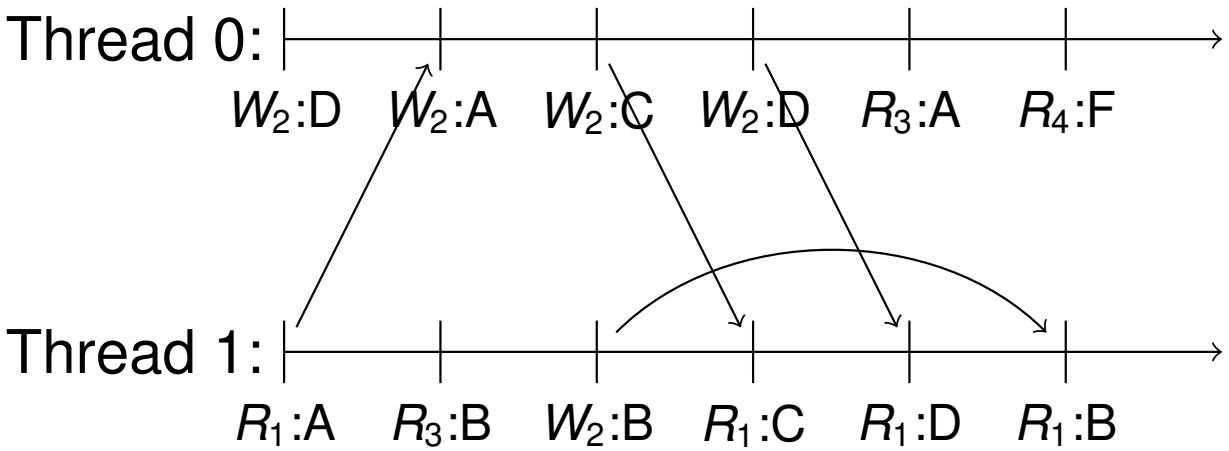
Modeling cache states in multi-cores
Sampling



Modeling cache states in multi-cores
Sampling



Modeling cache states in multi-cores
Sampling



Modeling cache states in multi-cores

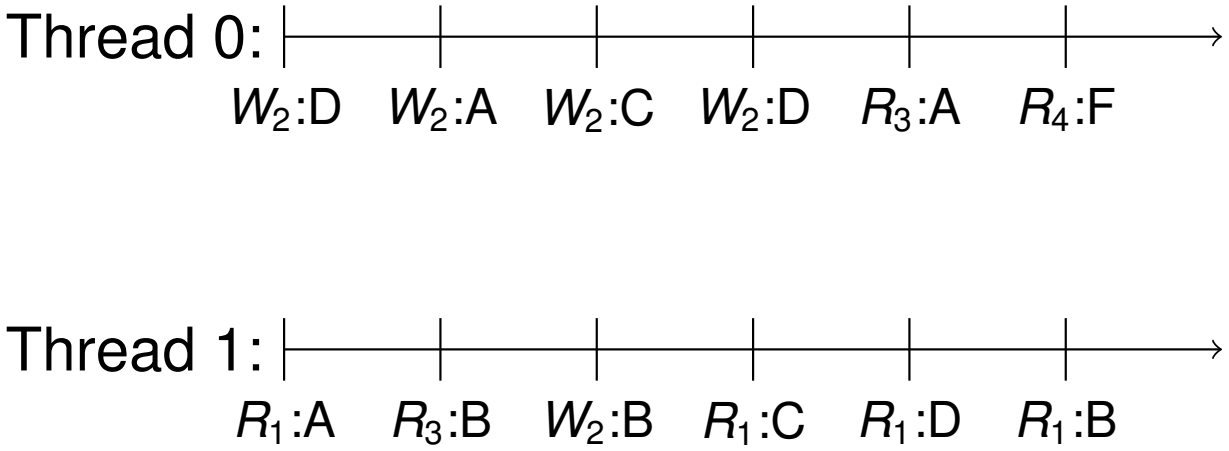
Modeling data flow

Using the samples we build a weighted directed graph consisting of access pairs.

1. Each (instruction, thread) pair is a vertex
2. The edges correspond to the access pairs in the memory access sample
3. The weight of an edge is the number of times an access pair has been sampled

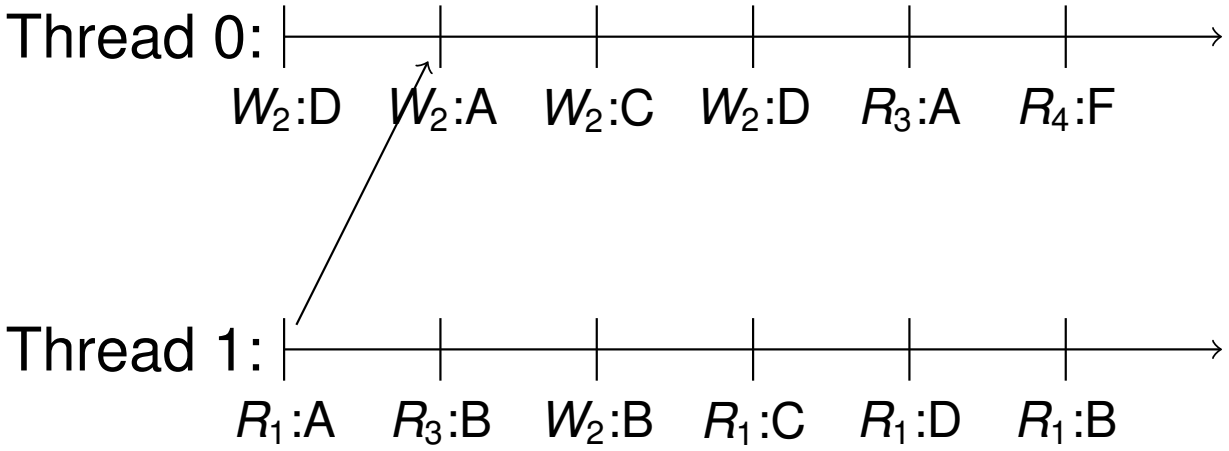
Modeling cache states in multi-cores

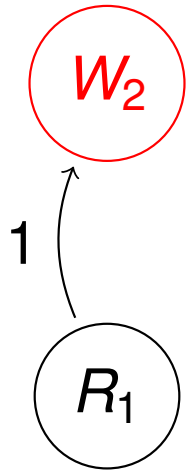
Graph building



Modeling cache states in multi-cores

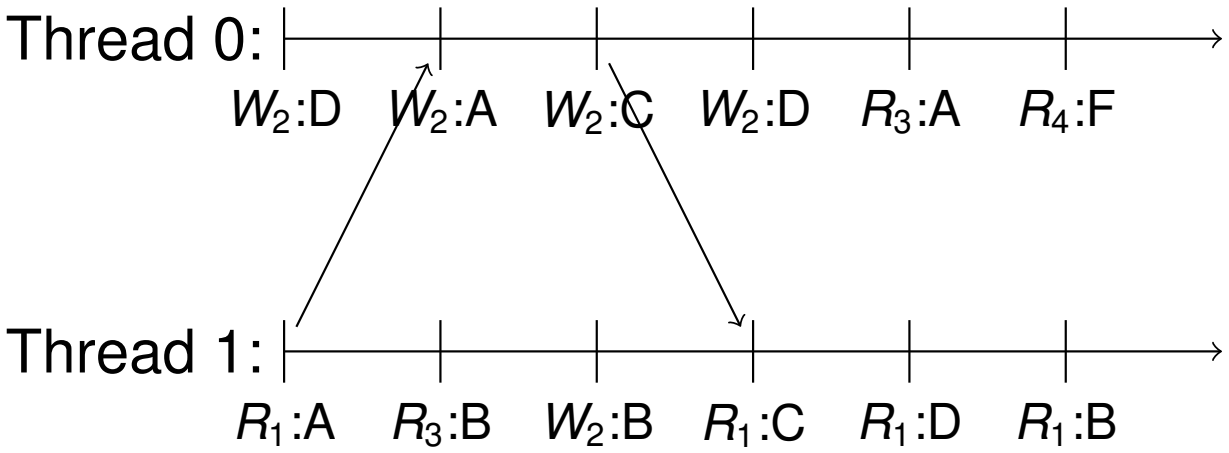
Graph building

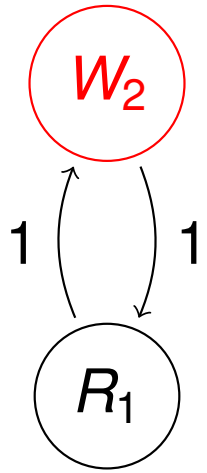




Modeling cache states in multi-cores

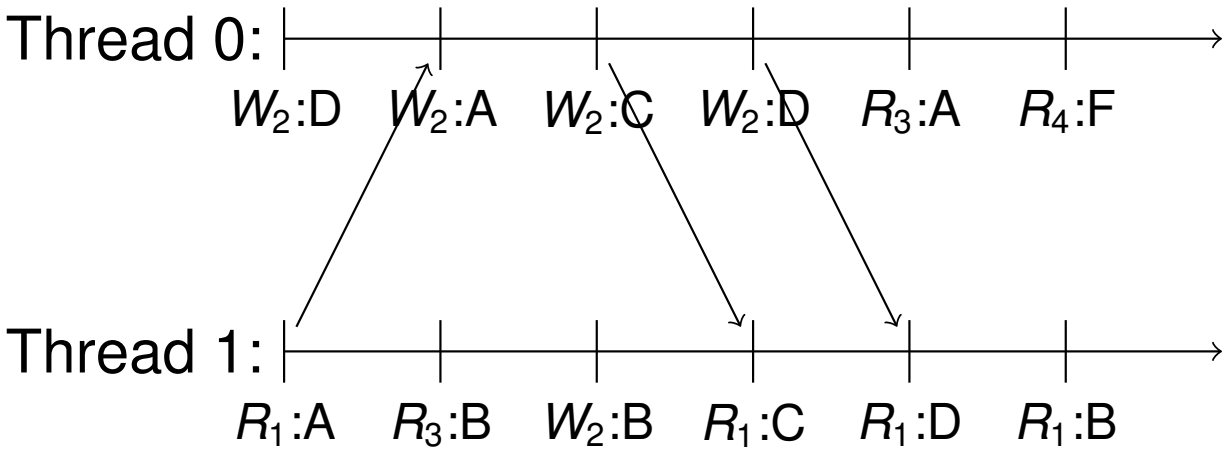
Graph building

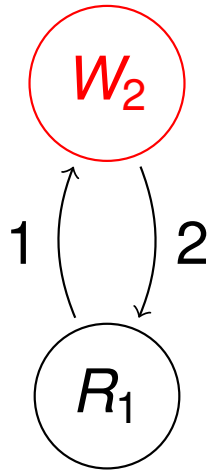




Modeling cache states in multi-cores

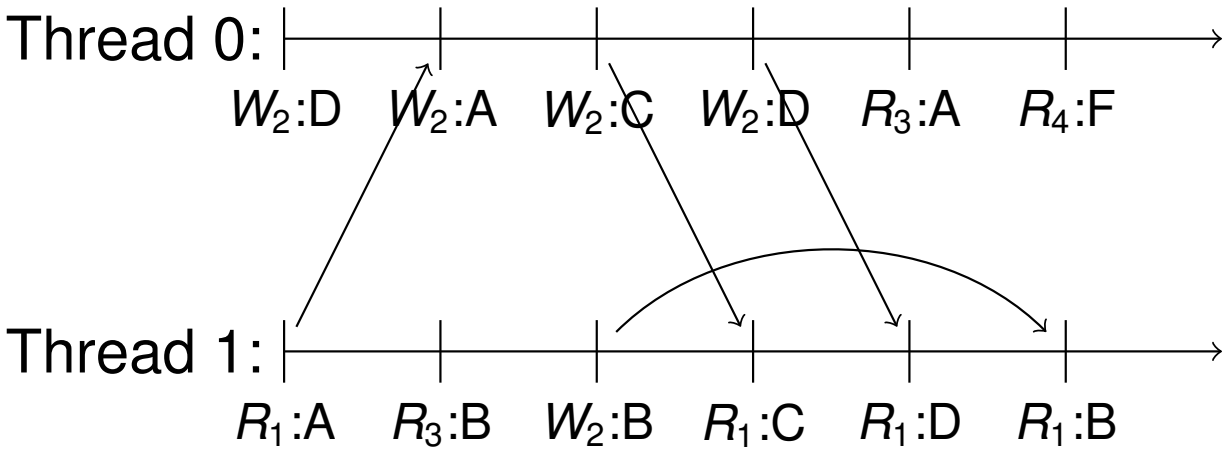
Graph building

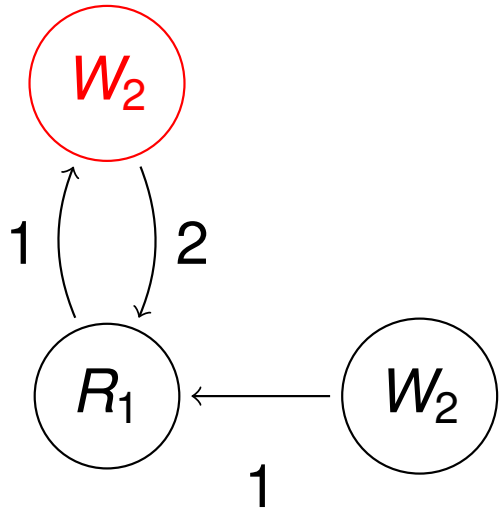




Modeling cache states in multi-cores

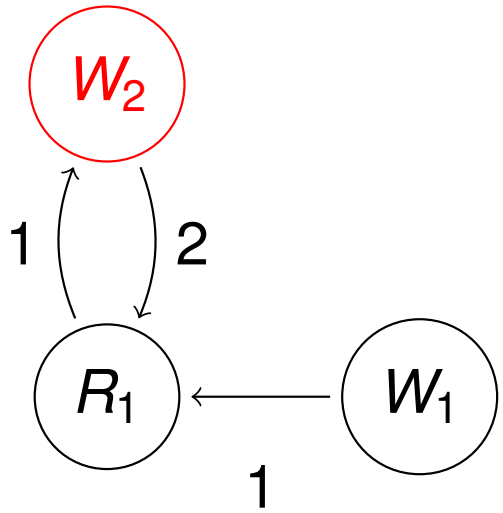
Graph building





Modeling cache states in multi-cores
Modeling cache state

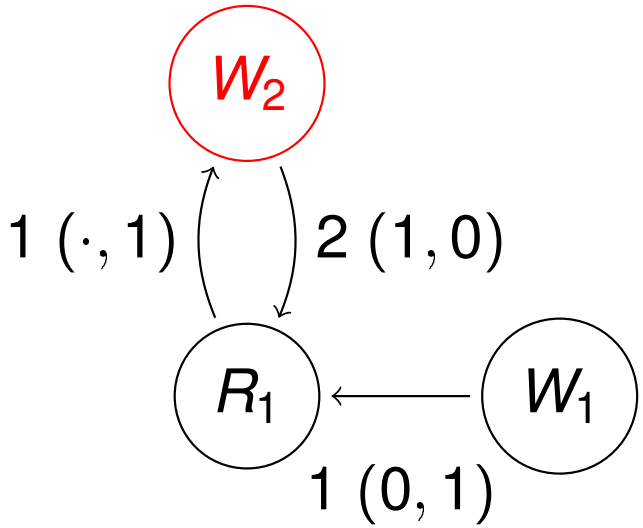
- Using the weights of the edges we can probability for a transition along an edge.
- We need to know the state of the system when an instruction executes and after it has executed.
  - The state after a write is known for all caches.
  - Reads don't affect the state of foreign caches.
- We can form an equation system to resolve unknown states.



Modeling cache states in multi-cores

Modeling cache state

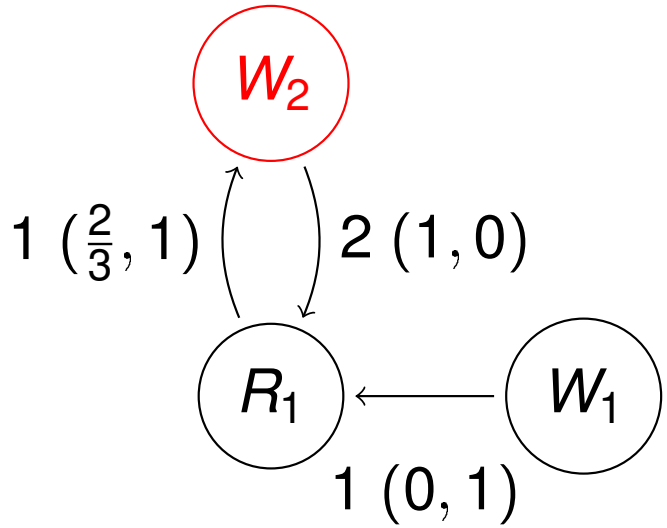
- Using the weights of the edges we can probability for a transition along an edge.
- We need to know the state of the system when an instruction executes and after it has executed.
  - The state after a write is known for all caches.
  - Reads don't affect the state of foreign caches.
- We can form an equation system to resolve unknown states.



Modeling cache states in multi-cores

Modeling cache state

- Using the weights of the edges we can probability for a transition along an edge.
- We need to know the state of the system when an instruction executes and after it has executed.
  - The state after a write is known for all caches.
  - Reads don't affect the state of foreign caches.
- We can form an equation system to resolve unknown states.



Evaluation
Methodology

Reference:

- Cache simulator simulating 2 CPUs
- *Private* 4MB caches
- Uses a trace with $\approx$ 200 000 000 accesses

Model:

- Uses a sample of the access trace
- Sample period 100
- $\approx$ 1000 times fast than the simulator.

Instructions with more than 50 000 invalidation misses were classified as hot-spots.

Evaluation
Results

Benchmark	Reference	Model	Missing hot-spots
GS	5	8	0
Raytrace	6	11	0
BT	13	39	1

Evaluation
Results

Benchmark	Reference	Model	Missing hot-spots
GS	5	8	0
Raytrace	6	11	0
BT	13	39	1

Evaluation
Results

Benchmark	Reference	Model	Missing hot-spots
GS	5	8	0
Raytrace	6	11	0
BT	13	39	1

Evaluation
Results

Benchmark	Reference	Model	Missing hot-spots
GS	5	8	0
Raytrace	6	11	0
BT	13	39	1

UPPSALA
UNIVERSITET

Summary

- We have a working prototype that can analyze real applications
- We managed to find the majority ($\approx$ 95%) of the communication hot-spots in real applications
- Several orders of magnitude faster than a simulator