The Use of Prediction for Accelerating Upgrade Misses in cc-NUMA Multiprocessors

Manuel E. Acacio[1], José González[2], José M. García[1] and José Duato[3]

[1] Dept. Ingeniería y Tecnología de Computadores
Universidad de Murcia (SPAIN)

[2] Intel Barcelona Research Center
Intel Labs, Barcelona (SPAIN)

[3] Dept. Informàtica de Sistemes i Computadors
Universitat Politècnica de València (SPAIN)

e-mail: meacacio@ditec.um.es

GACOP

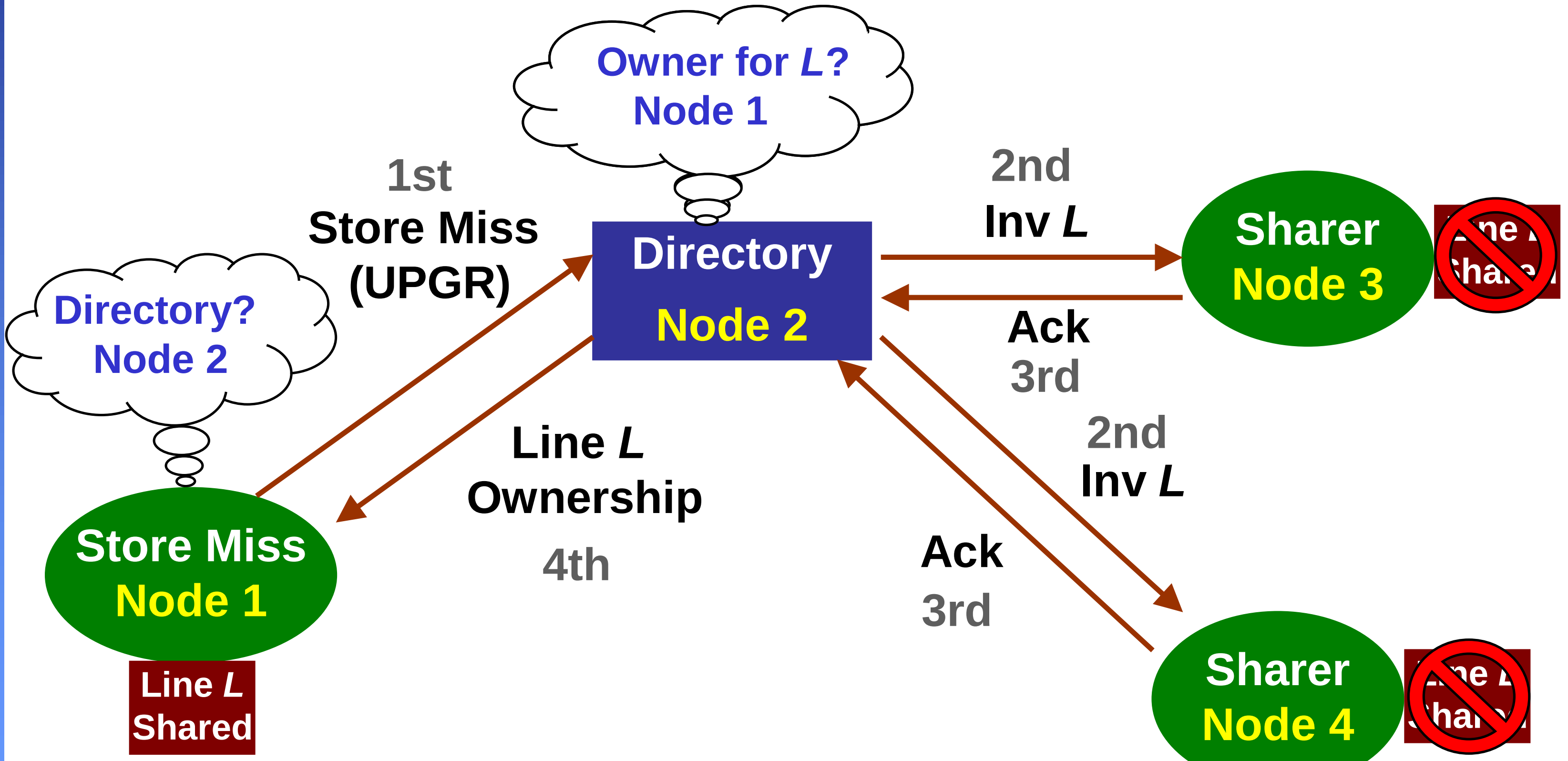
Introduction

- **Scalable shared-memory multiprocessors**
  - Based on the use of directories
  - Known as cc-NUMA architectures
- **Long L2 miss latencies**
  - Mainly caused by the indirection introduced by the access to the directory information
    - *Network* latency
    - *Directory* latency
- ***Upgrade* *misses***
  - Important fraction of the L2 miss rate (> 40%)
  - Store instruction for which a read-only copy of the line is found in the local L2 cache
  - Exclusive ownership is required

GACOP

Introduction

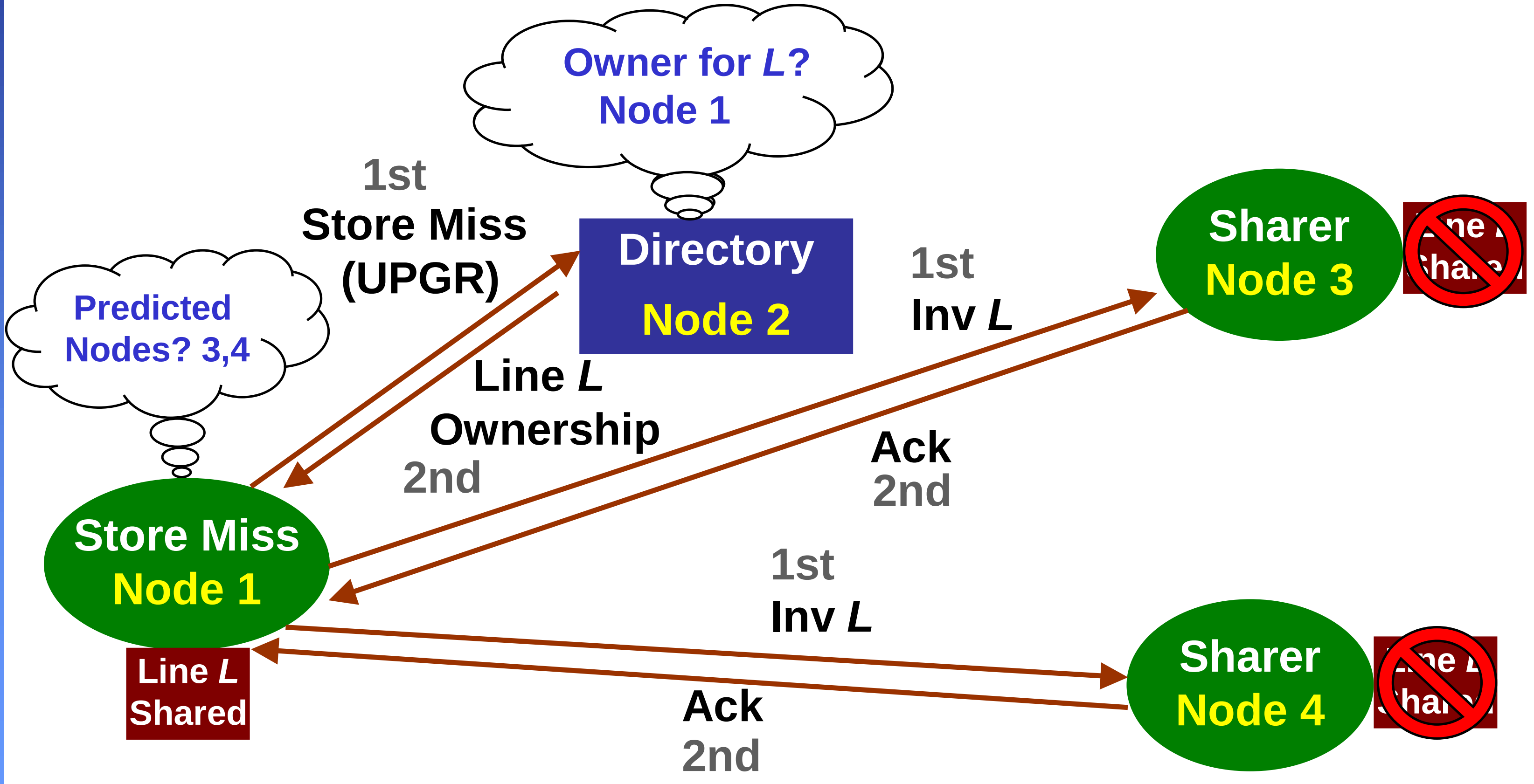
- *Upgrade misses* in a conventional cc-NUMA



GACOP

Introduction

- *Upgrade misses* using prediction



GACOP

Introduction

- **Two key observations motivate our work:**
  - Repetitive behavior found for *upgrade misses*
  - Small number of invalidations sent on an *upgrade miss*
- **Two main elements must be developed:**
  - An effective *prediction engine*
    - Accessed on an *upgrade* miss
    - Provides a list of the sharers
  - A *coherence protocol*
    - Properly extended to support the use of prediction

GACOP

Outline

- Introduction
- Predictor Design for Upgrade Misses
- Extensions to a MESI Coherence Protocol
- Performance Evaluation
- Conclusions

GACOP

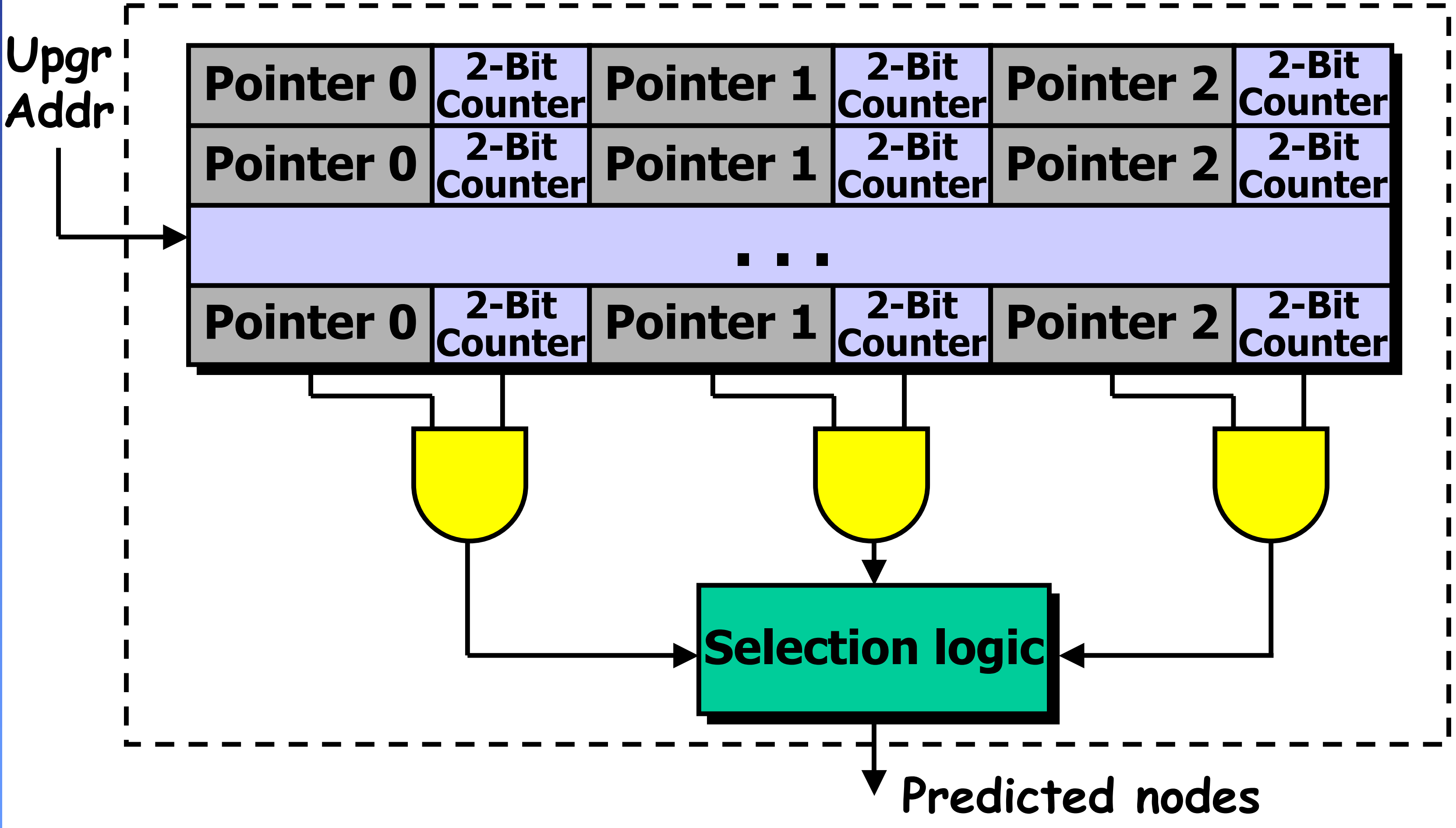
Predictor Design for Upgrade…

- **Predictor characteristics:**
  - *Address-based* predictor
    - Accessed using the effective address of the line
  - 3 pointers per entry
    - Small number of sharers per line
    - Addition of confidence bits per each pointer
    - (3 x $\log_2 N$ + 6) bits per entry
  - Implemented as a *non-tagged table*
    - Initially, all 2-bit counters store 0
    - Predictor is probed on each upgrade miss
      - Miss predicted when confidence
    - Predictor is updated in two situations:
      - On the reply from the directory
      - On a load miss serviced with a $-to-$ transfer (*Migratory Data*)

GACOP

Predictor Design for Upgrade...

- Predictor Anatomy



GACOP

Outline

GACOP

Extensions to a MESI Protocol

- **Changes to Requesting node, sharer nodes, home directory**
- **Requesting Node Operation**
  - **On suffering a predicted UPGRADE MISS**

$$❶\left\{\begin{array}{l}\text{– Create \& send invalidation messages to predicted nodes}\\ \quad\text{❍ Put message } \textit{Predicted} \text{ bit to 1}\\ \text{– Send miss to the directory}\end{array}\right.$$

❍ Put message *Predicted* bit to 1 and include the list of predicted nodes

❷ – Collect *directory reply* and *ACK* **/** *NACK* from predicted nodes:

❍ Re-invalidate those *real* sharers that replied *NACK* (if any)

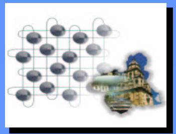
❸ – Gain *exclusive ownership*

GACOP

Extensions to a MESI Protocol

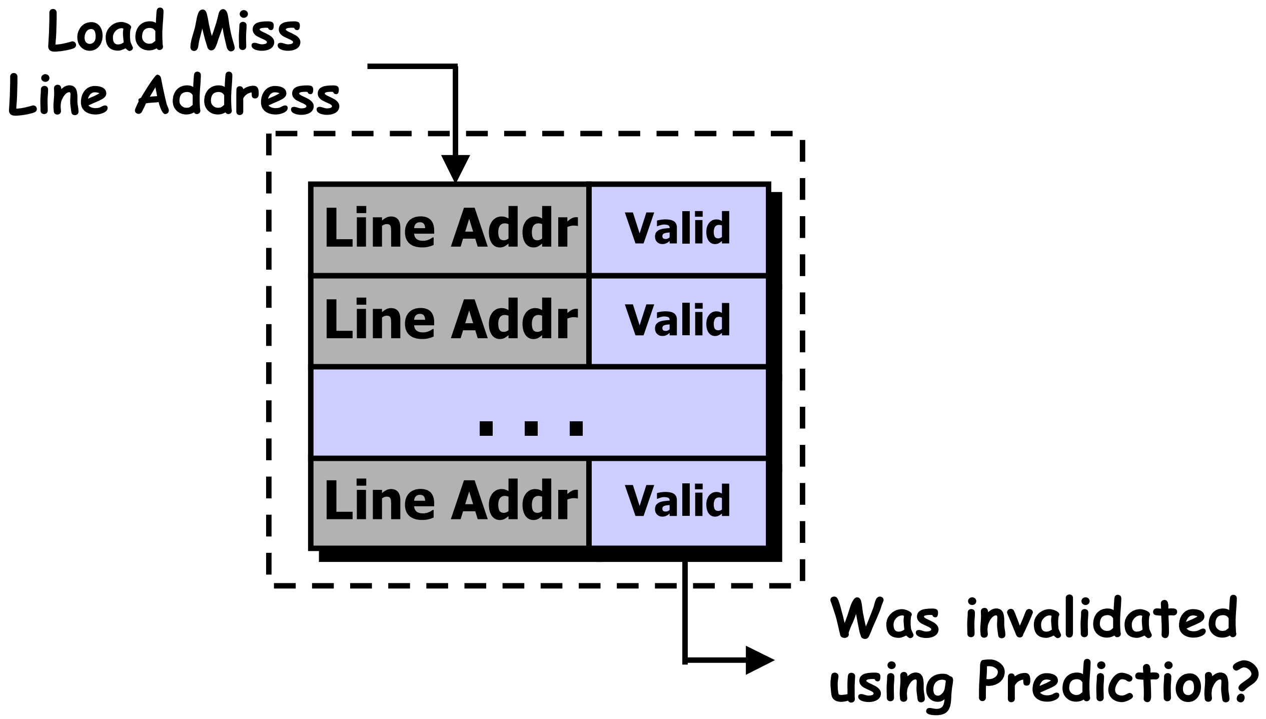
Sharer Node Operation

- **On receiving a predicted INVALIDATION message and**
  - Pending *Load* Miss: store invalidation and return *NACK*
  - Pending *UPGR* Miss (line in the *Shared* state):
    - Directory reply not received: return *ACK* and invalidate line
    - Directory reply previously received: return *NACK*
  - Not pending UPGR Miss and line in the *Shared* state:
    - Return *ACK* and invalidate
    - Insert tag in Invalidated Lines Table (*ILT*)
  - Otherwise, return *NACK* message
- **On suffering a Load Miss**
  - If entry found in the *ILT*, put message *Invalidated* bit to 1


GACOP

Extensions to a MESI Protocol

- Predictor + *ILT* added to each node
- Anatomy of the *Invalidated Lines Table* (ILT)



GACOP

Extensions to a MESI Protocol

Directory Node Operation

- **On receiving a predicted UPGRAGE MISS**
  - If line is in the *Shared* state
    - All sharers predicted ➲ send reply (**TOTAL HIT**)
    - Some actual sharers not predicted (**PARTIAL HIT**) or none correctly predicted (**TOTAL MISS**) ➲ Invalidate and send reply
  - Otherwise, process as usually (**NOT INV**)

- **On receiving a Load Miss**
  - If message *Invalidated* bit is set **&&** requesting node present in sharing code ➲ wait until UPGR to complete!
  - Otherwise, process as usually

GACOP

Outline

- Introduction
- Predictor Design for Upgrade Misses
- Extensions to a MESI Coherence Protocol
- Performance Evaluation
- Conclusions

GACOP

Performance Evaluation

- **Performance Evaluation**
  - RSIM multiprocessor simulator

Number of nodes	16
Processor Speed	1 GHz
L1	32 KB Direct Mapped (2 cycles hit)
L2	1 MB 4-way associative (15 cycles hit)
Main Memory	4 banks, 70-cycle latency
Network	2-dimensional mesh
Router speed	250 Mhz
Channel speed	500 Mhz

  - We assume that predictors do not add any cycle
  - Benchmarks
    - Applications with more than 25% *upgrade* misses covering a variety of patterns
      - EM3D, FFT, MP3D, Ocean and Unstructured

GACOP

Performance Evaluation

Experimental Framework

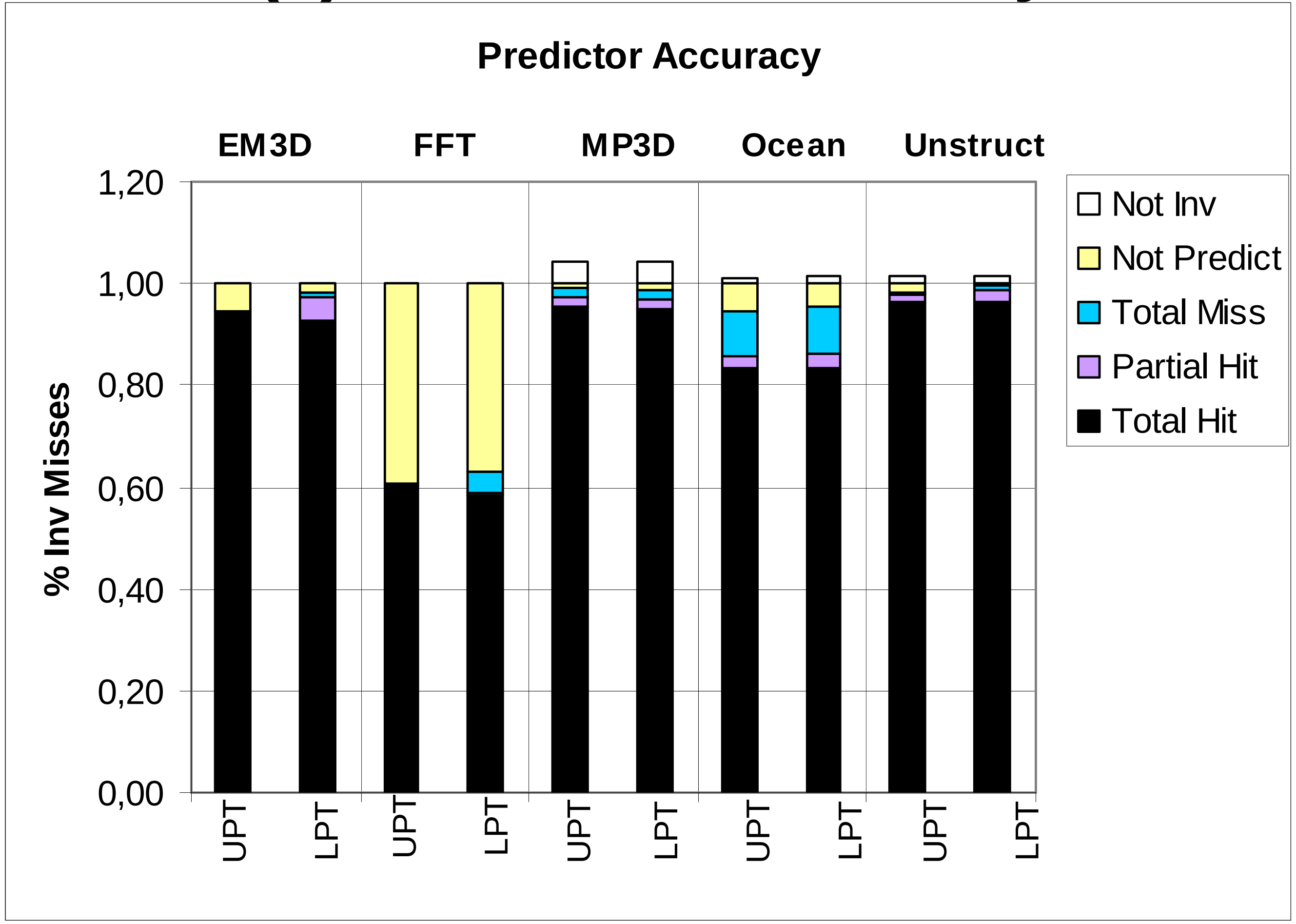
- Compared systems:
  - *Base*: Traditional cc-NUMA using a bit-vector directory
  - *UPT*: Added unlimited Prediction Table and *ILT*
  - *LPT*: Added a "realistic" Prediction Table and *ILT*
    - *Prediction Table*: 16K entries (non-tagged)
    - *ILT*: 128 entries (totally associative)
    - Total size less than 48 KB (1 MB L2 caches)

- We study:
  - Predictor accuracy
  - Impact on latency of *upgrade* misses
  - Impact on latency of *load* & *store* misses
  - Impact on execution time

GACOP

Performance Evaluation

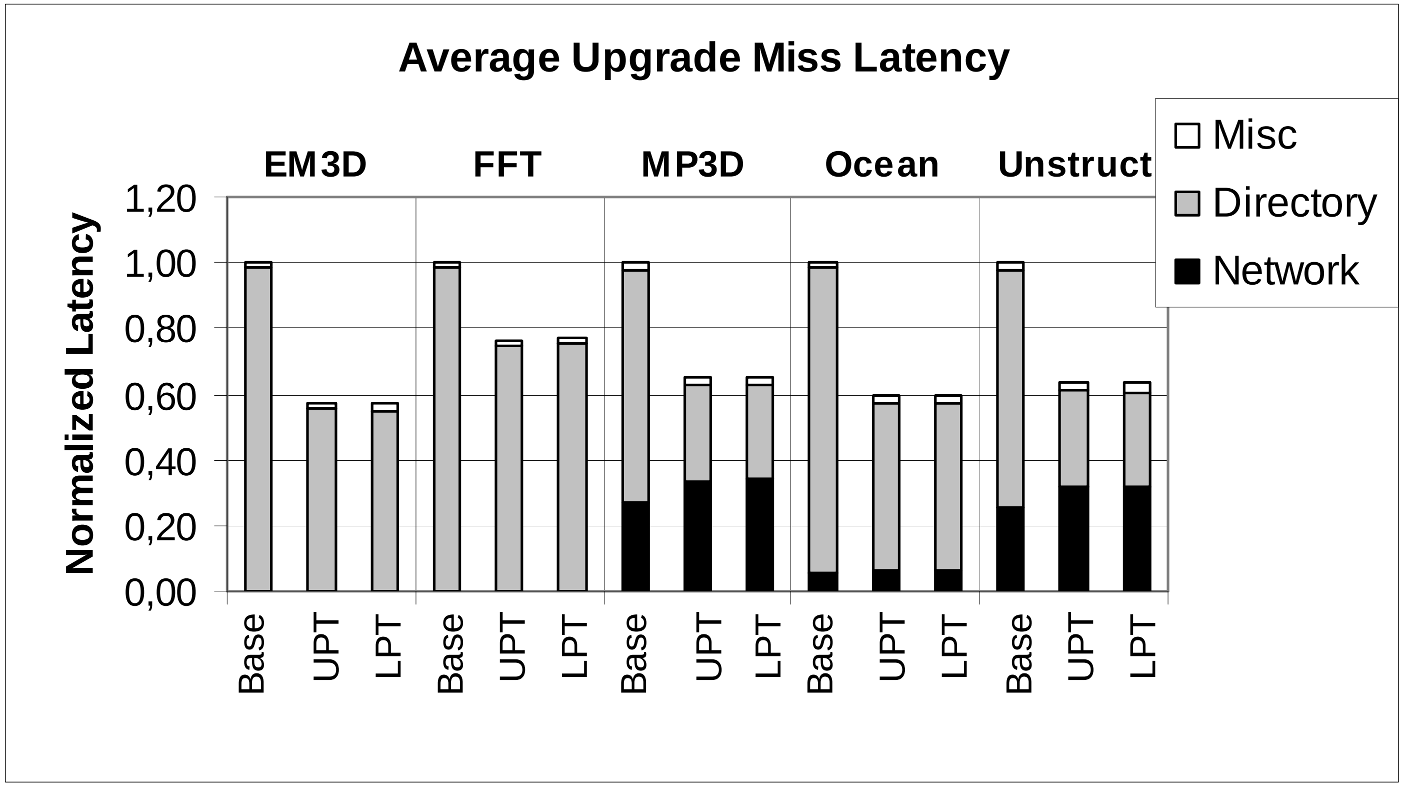
- **Results(1). Predictor Accuracy**



GACOP

Performance Evaluation

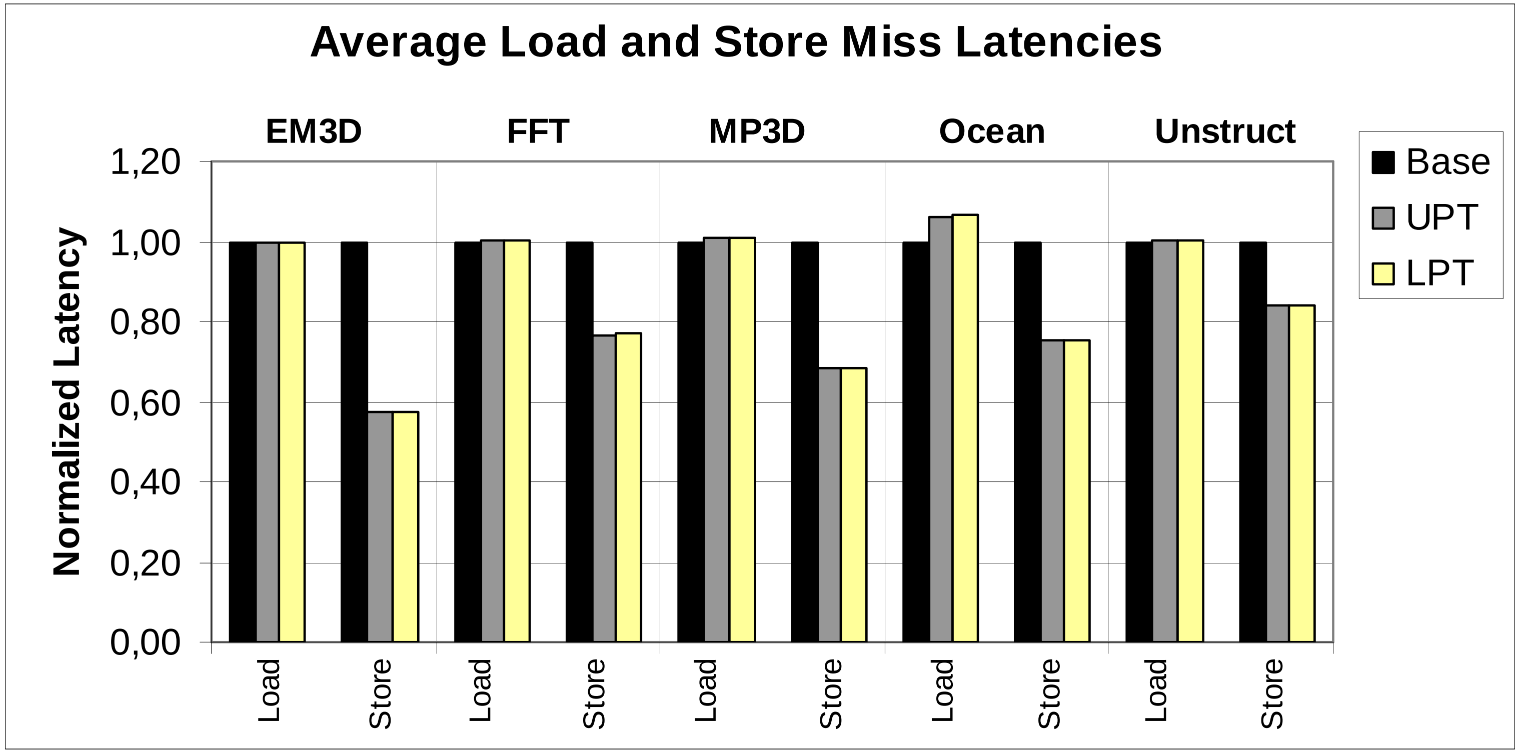
- Results(2). Average Upgrade Miss Latency



GACOP

Performance Evaluation

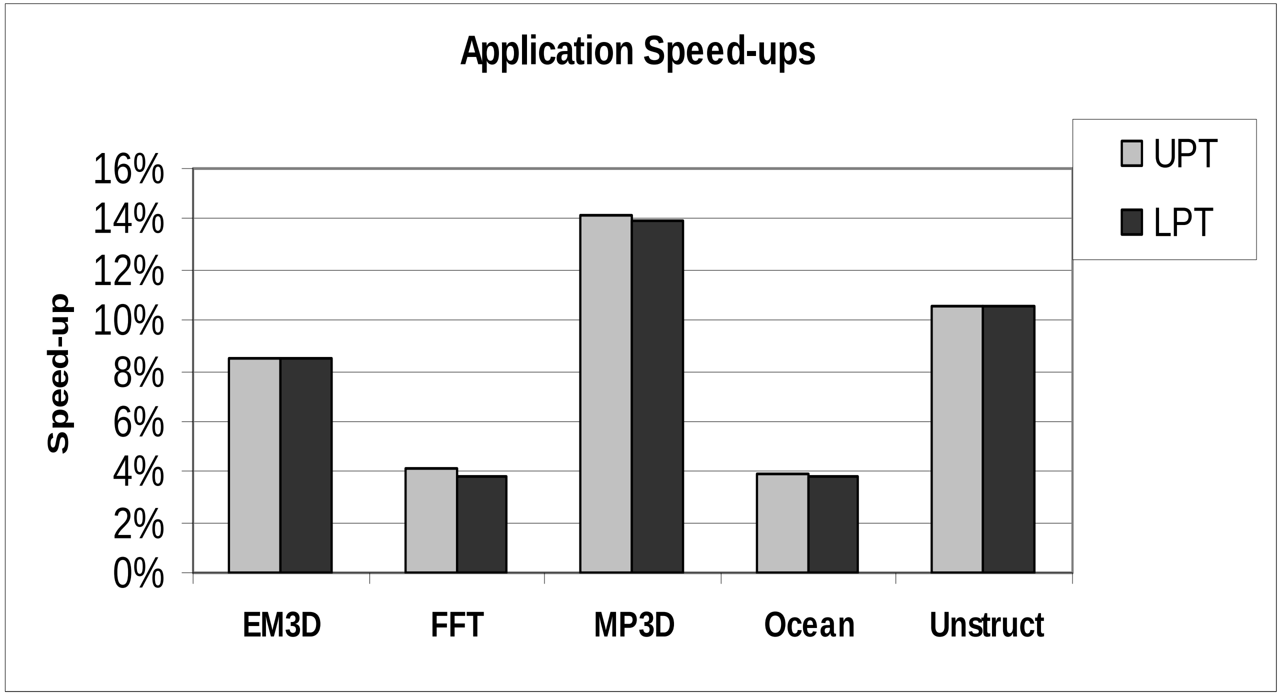
- **Results(3). Average Load/Store Miss Latency**



GACOP

Performance Evaluation

- **Results(4). Application Speed-ups**



GACOP

Outline

- Introduction
- Predictor Design for Upgrade Misses
- Extensions to a MESI Coherence Protocol
- Performance Evaluation
- Conclusions

GACOP

Conclusions

Conclusions (1)

- *Upgrade* misses are caused by a store instruction when a read-only copy is found:
  - Message sent to directory
  - Directory lookup
  - Invalidations sent to sharers
  - Replies to the invalidations sent back
  - Ownership message returned
- Account for an important fraction of the L2 miss rate (>40%)
- We propose use of *prediction* for accelerating them
  - On an *upgrade* miss: predict sharers and invalidate them in parallel with the access to the directory
  - Based on:
    - Repetitive behavior
    - Small number of invalidations per *upgrade* miss

GACOP

Conclusions

Conclusions (2)

- Results:
  - Great fraction of upgrade misses successfully predicted
  - Reductions > 40% on average upgrade miss latency
  - Load miss latencies are not affected in most cases
  - Speed-ups on application execution time up to 14%

GACOP

The Use of Prediction for Accelerating Upgrade Misses in cc-NUMA Multiprocessors

Manuel E. Acacio[1], José González[2], José M. García[1] and José Duato[3]

[1] Dept. Ingeniería y Tecnología de Computadores
Universidad de Murcia (SPAIN)

[2] Intel Barcelona Research Center
Intel Labs, Barcelona (SPAIN)

[3] Dept. Informàtica de Sistemes i Computadors
Universitat Politècnica de València (SPAIN)

e-mail: meacacio@ditec.um.es

GACOP