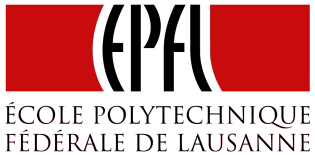
TM^{2}C: a Software Transactional Memory for Many-Cores

Vasileios Trigonakis



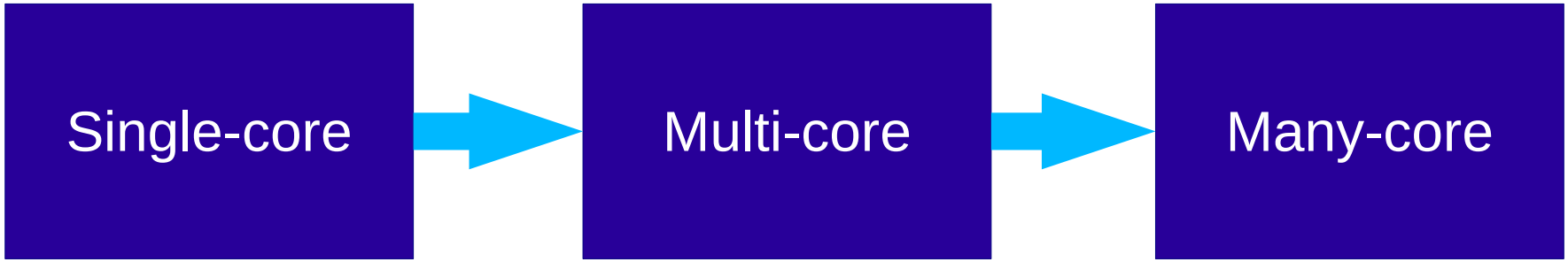
Joint work with:
Dr. Vincent Gramoli
Prof. Rachid Guerraoui

Outline

- Introduction
  - many-cores
  - (D)TM
- TM^2C
- Evaluation
- Portability
- Conclusions

Introduction

Processors



- 1 core
- Moore's law
- “Free lunch”

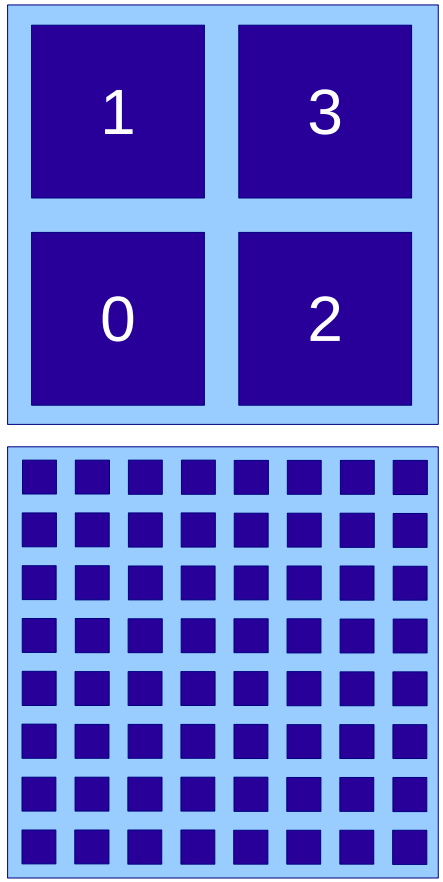
- > 1 cores
- Complex cores
- Cache-coherence

- **???**

Many-core

- More cores
- Simper cores
- Limited or no hardware cache-coherence
- **Message passing**

- … a Distributed System



(Distributed) Transactional Memory

- Ease of programming
- Avoids
  - deadlocks
  - priority inversion
  - lock convoying
  - …
- Simple interface
  - hides message passing

Transactional Memories

DTMs

- DMV [PPoPP'06]
- Cluster-TM [PPoPP'08]
- D^2STM/ALC [Middleware'10]
- Snake-DSTM [SSS'11]
- ...

STMs

- HTM [ISCA'93]
- STM [PODC'95]
- TL2 [DISC'06]
- McRT-STM [PPoPP'06]
- E-STM [DISC'09]
- ...

Transactional Memories

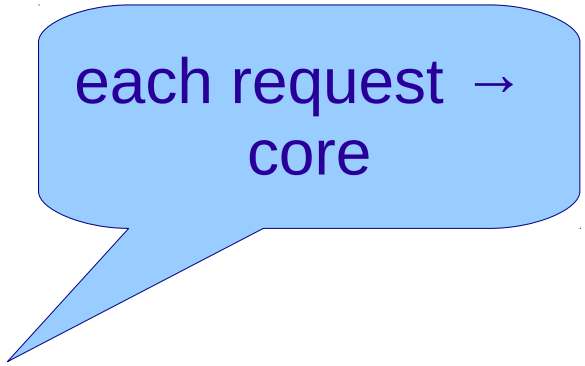
DTMs	STMs
Non-terminating transactions	**Assume cache-coherence**

Liveness

- Obstruction-freedom
  - one process → it completes requests
- Livelock-freedom
  - any process → **some** process completes requests
- Starvation-freedom
  - any process → **this** process completes requests

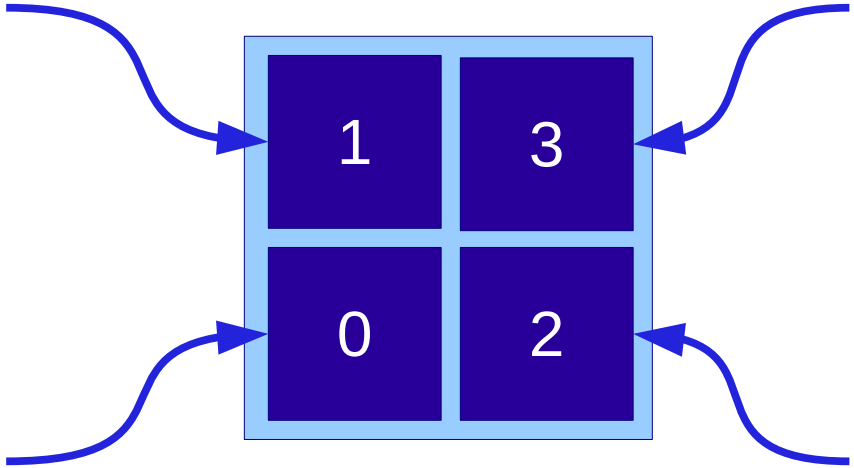
Liveness

- **Which one should we aim for?**
  - depends on
    - the application
    - what is “possible”
- Many-cores
  - application:
    - cloud computing
    - server applications
  - what is “possible”
    - starvation-freedom



Starvation

- Client 1
  - tx 1
  - tx 2
  - tx 3
  - ...
- Client 0
  - tx 1
  - tx 2
  - tx 3
  - ...



- Client 3
  - tx 1
  - tx 2
  - tx 3
  - ...
- Client 2

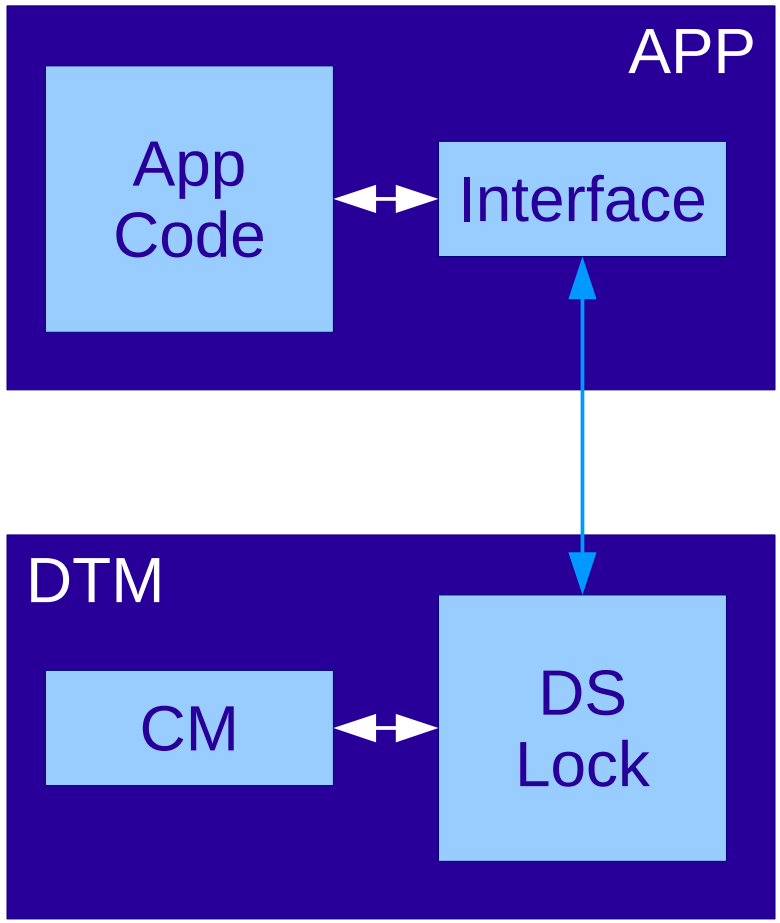
Starvation

TM^2C

TM^{2}C

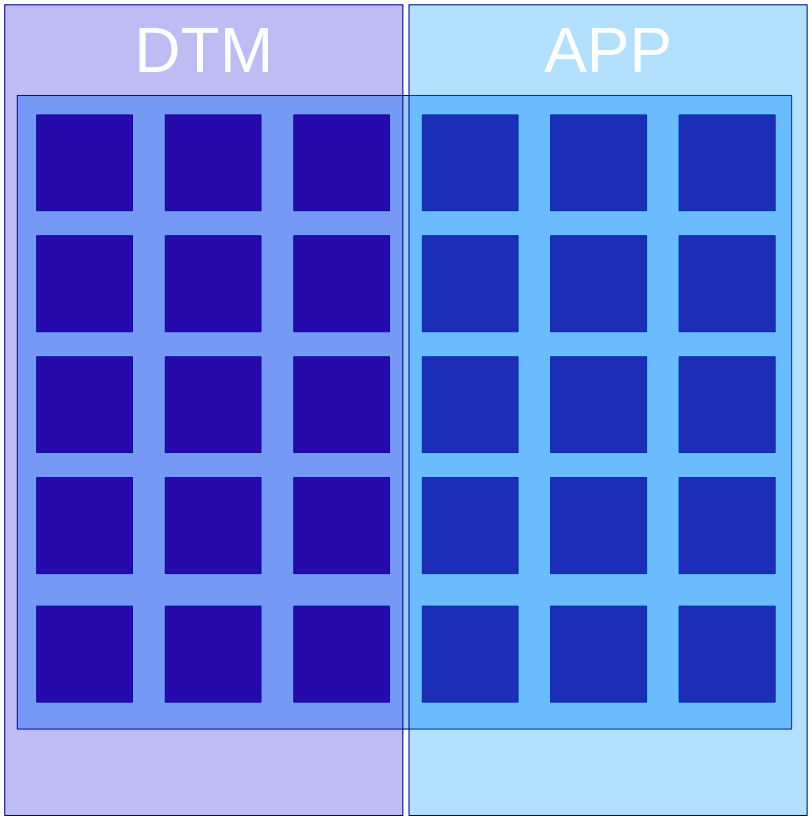
- **T**ransactional **M**emory for **M**any-**C**ores
  - 1st TM for many-cores
  - 1st starvation-free DCM
    - FairCM

TM^{2}C: Components



- App code
- Interface
  - tx_load(...)
  - tx_store(...)
  - ...
- DS Lock
  - MR/SW locks
- CM
  - conflict resolution

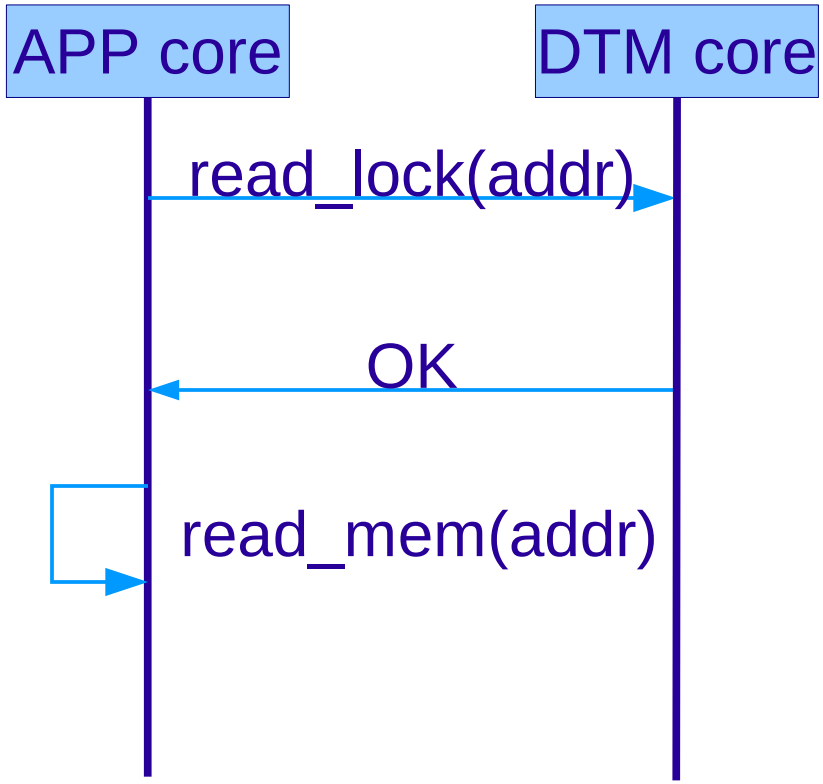
TM^{2}C: Deployment



- DTM / APP
  - different cores
- Benefits
  - decoupling
  - messaging
  - flexibility

Transactional Read

- Visible reads
  - eager lock acquisition
- Why?
  - low latency
  - eager detection



Transactional Write

- Deferred writes
  - buffered
  - lazy lock acquisition
- Why?
  - minimize conflicts

Distributed Contention Management

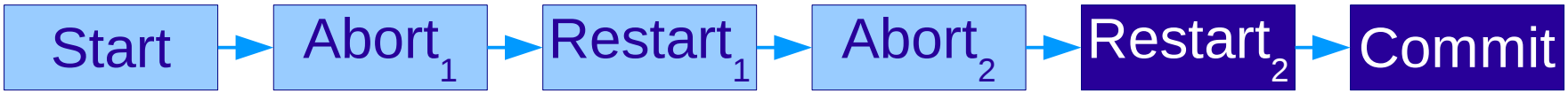
- Similar to the STMs
  - but, fully decentralized
- Conflicts:
  - RAW
  - WAR
  - WAW

Contention Managers

- Offset-Greedy
  - adaptation of Greedy [PODC'05]
  - time offsets
- Wholly
  - # of committed txs

FairCM

- Effective transactional time

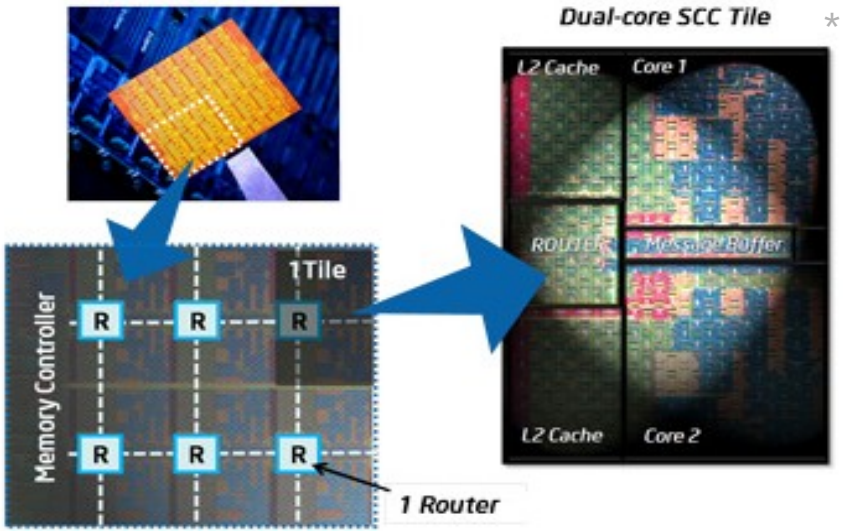


  - short transactions cheaper

- Fairness
  - time consumed by each core

Evaluation

Intel's Single-chip Cloud Computer (SCC)



- 48 cores
- Network on Chip
- **No Hardware Cache–coherence**
- Software-based Message Passing

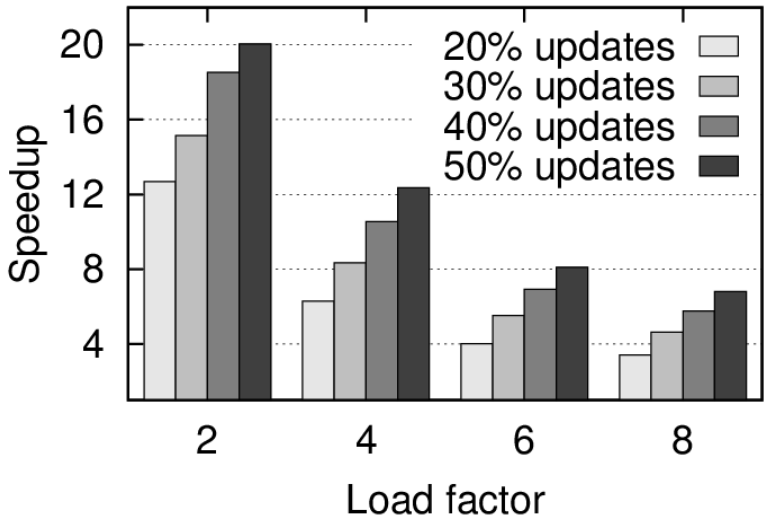
* http://techresearch.intel.com/spaw2/uploads/images/Scc-Diagram1b.jpg

Settings

- SCC
  - slowest performance mode
- TM^{2}C Deployment
  - ½ DTM cores
  - ½ APP cores

Evaluation: Hash table

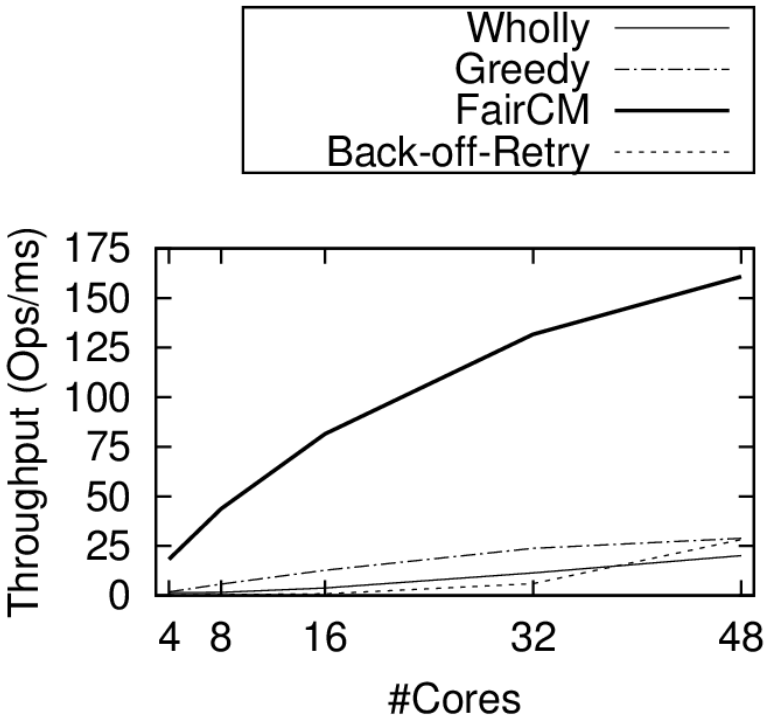
- Improvement over sequential
- 48 cores



- Updates
  - insert
  - remove
- Writes more expensive
- ↑writes
  - better MC usage

Evaluation: Bank

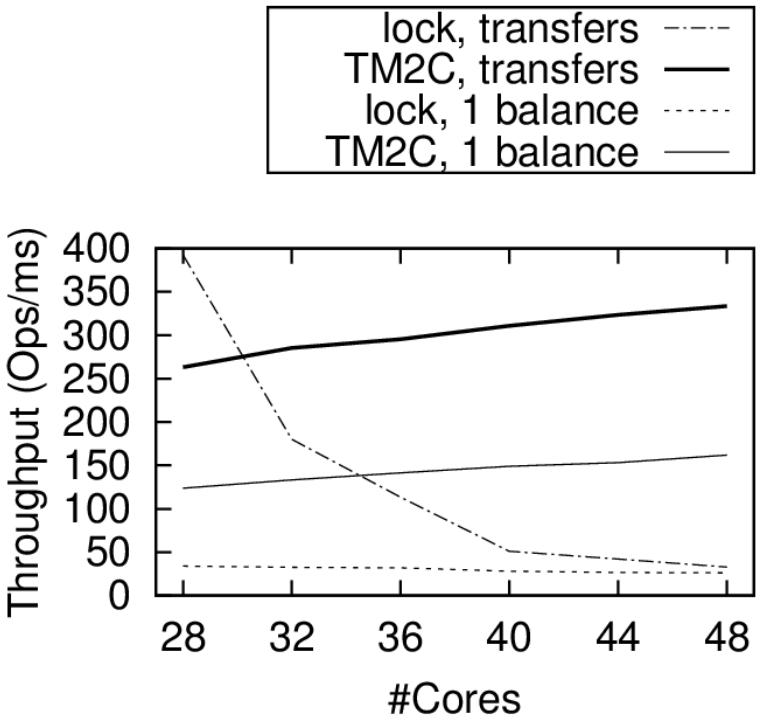
- balance by 1 core
- 100% transfer



- FairCM
  - balance more expensive
  - < 10% abort rate
- Trade-off
  - O-Greedy: 81 bals/s
  - FairCM: 44 bals/s

Evaluation: Bank

- Improvement over lock
- 2 workloads



- Global lock
  - bottleneck
  - doesn't scale

Portability

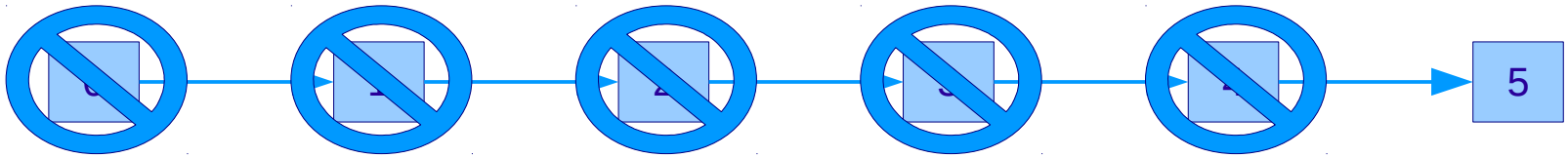
- TM^{2}C needs
  - message passing
  - shared memory → PGAS
- Ported
  - multi-core
    - AMD Opteron 48-core
  - cluster

Conclusions

- On a cluster/large-scale DS
  - try to hide expensive communication
- On a many-core
  - take advantage of cheap communication
- V. Gramoli, R. Guerraoui, V. Trigonakis, *TM2C: a Software Transactional Memory for Many-Cores*, EuroSys 2012

Elastic Linked List (1/2)

- Linked list
  - sequential accesses
  - are all locks necessary?



?

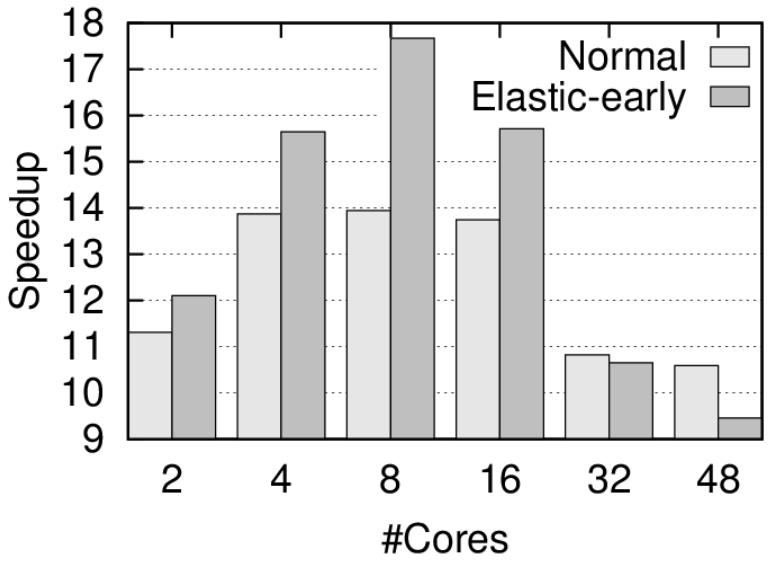
Elastic Linked List (2/2)

- Elastic opacity [DISC'09]

- Implementations
  - elastic-early
    - using explicit release
  - elastic-read
    - using read validation

Evaluation: Linked list

- 2048 elements
- 20% updates



- Improvement of elastic-read
- SCC
  - memory access cheaper than messaging