Coarse-grained Simulation Method for Performance Evaluation of a Shared Memory System

R. Kawahara, K. Nakamura, K. Ono, T. Nakada
(IBM Research – Tokyo, Japan)

Y. Sakamoto
(Global Business Services, IBM Japan, Japan)

Abstract

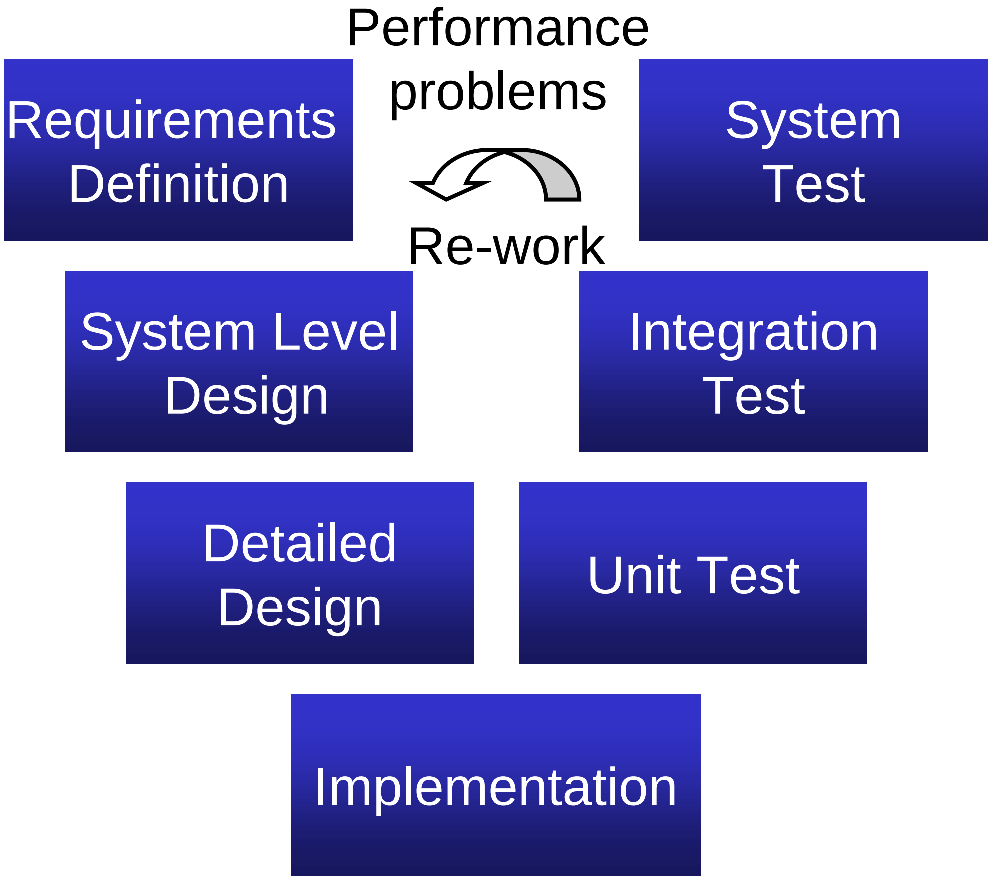
- Purpose
  - Performance evaluation at early stage of development
- Subject
  - Evaluation of memory access contention without knowing precise memory access timings
- Our method
  - Exploit a statistical approximation which assumes that memory accesses are random and uncorrelated
- Comparison with preliminary experiment
  - Error of order of 3% on the execution time

Introduction: Application-level performance estimation at early stage of development

UML is used to clarify software specification

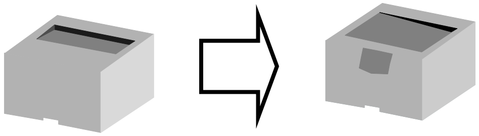
⇕

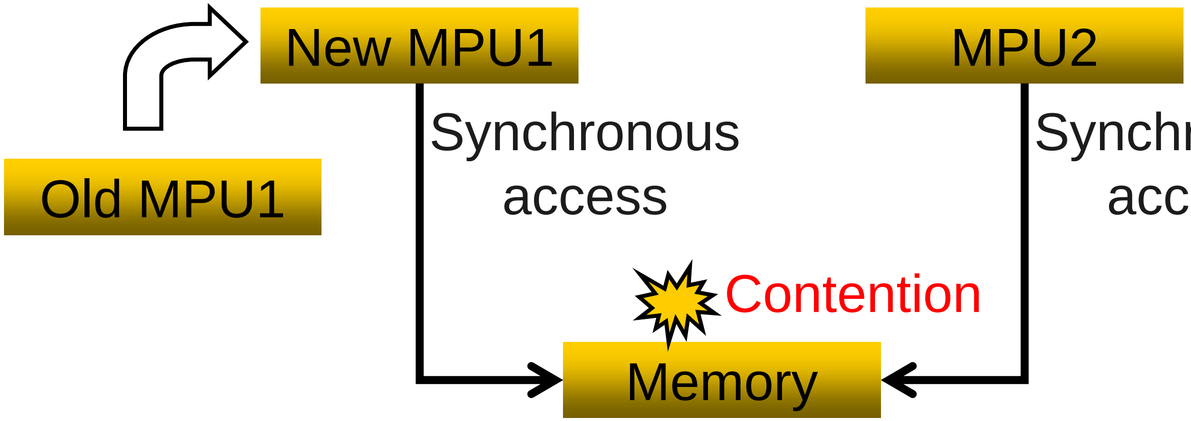
It is difficult to evaluate application performance because it requires platform resources to be taken into account



Introduction: Memory Access Contention

- Context
  - Improvement of performance for next-generation product
- Metric of performance
  - execution time of application





Old MPU1

New MPU1

MPU2

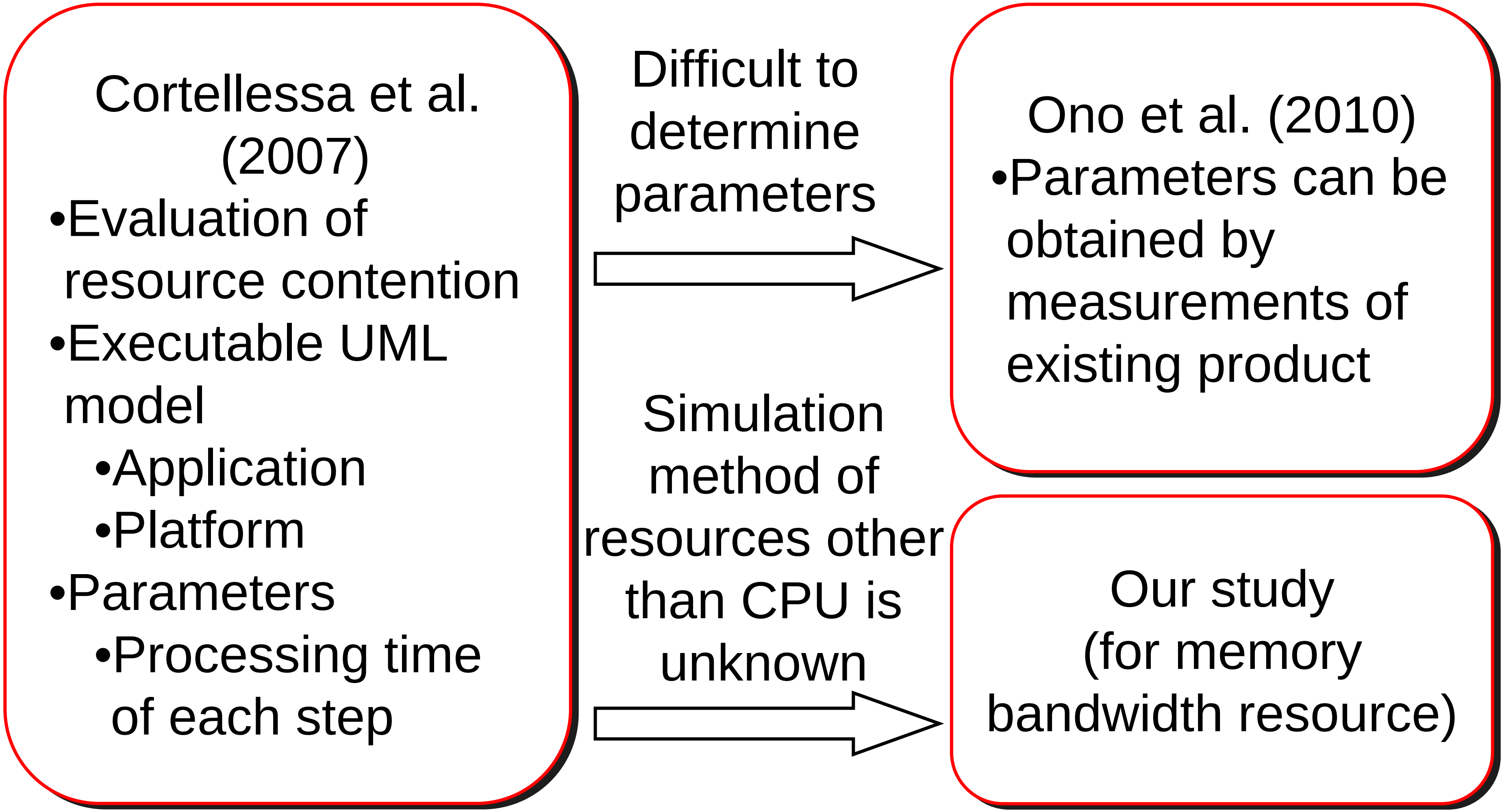
Synchronous access

Synchronous access

Contention

Memory

Related works: UML-based simulation



Cortellessa et al. (2007)

- Evaluation of resource contention
- Executable UML model
  - Application
  - Platform
- Parameters
  - Processing time of each step

Difficult to determine parameters

Ono et al. (2010)

- Parameters can be obtained by measurements of existing product

Simulation method of resources other than CPU is unknown

Our study (for memory bandwidth resource)

Related works: Simulation methods of memory access contention

Dev. phase	Methods	Input	Output
System-level analysis	Queuing Network	System requirements	Statistics
UML-level design	?	Architecture (HW / SW), Step time	Execution time, Step trace
ESL-level design	SystemC TLM	Detailed design, Transaction timings	Execution time, Transaction trace
Implementation	ISS	Binary code	All

Abstract (fast)

Detailed (slow)

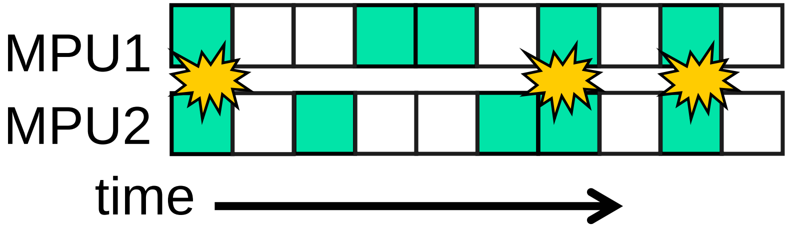
TLM: Transaction-Level Modeling, ISS: Instruction Set Simulator

Question: How to evaluate memory access contention without access timing information?

Method: Main Idea

Approximations on memory access timings:

- Random within a simulation step
- Uncorrelated between processors



MPU1

MPU2

time

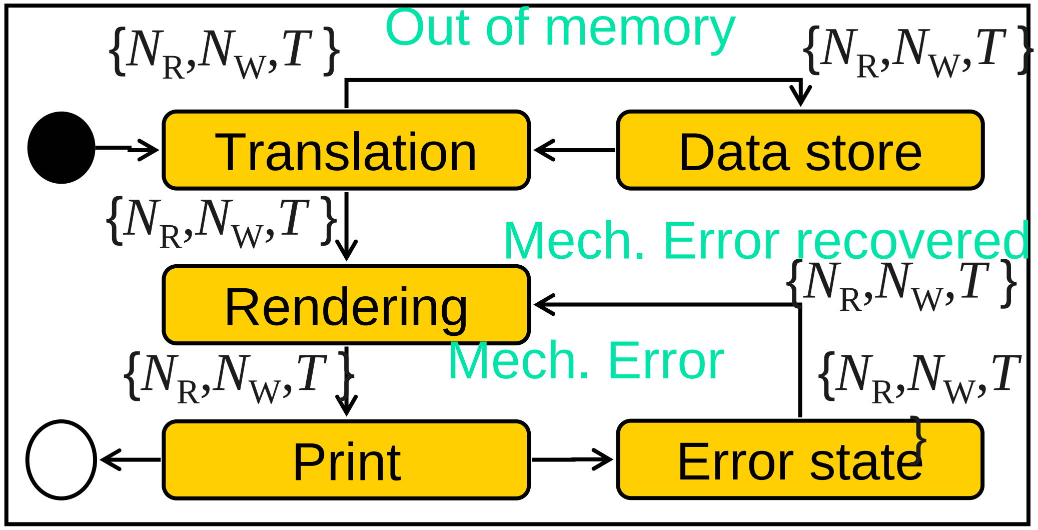
🟩 Memory access

□ Other instructions

Example:

- Bandwidth utilization:
  - $U_1 = 5/10,\ U_2 = 5/10$
- Collision probability:
  - $P_{12}(U_1, U_2) = U_1\ U_2\ = 0.25$
- For round-robin arbitration:
  - $A = 1*(1/2) + 2*(1/2) = 1.5$
- Increase of step time:
  - $T'/T = 1 + (A - 1)*P_{12} = 1.12$

Method: Model and Parameter



$\{N_R, N_W, T\}$

Out of memory

$\{N_R, N_W, T\}$

Translation

Data store

$\{N_R, N_W, T\}$

Mech. Error recovered

$\{N_R, N_W, T\}$

Rendering

$\{N_R, N_W, T\}$

Mech. Error

$\{N_R, N_W, T\}$

Print

Error state

Input model
(application behavior
with parameters)

$$U_i = \frac{N_{Ri}}{T_i W_R} + \frac{N_{Wi}}{T_i W_W}$$

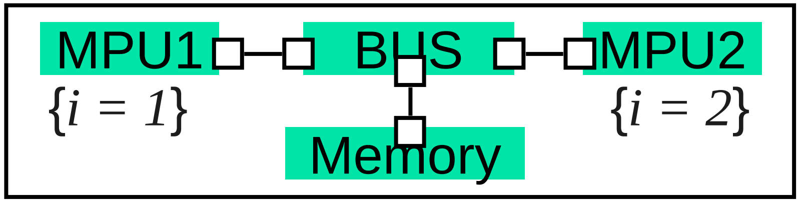
N_R : Number of read access

N_W : Number of write access

W_R : Read throughput (times/s)

W_W : Write throughput (times/s)

Method: Overview of our method



Input model
(platform architecture)

$$T_i'/T_i = 1 + \sum_{c=0}^{2^M-1} (A_{c,i} - 1) P_c (U_1, \cdots, U_M)$$

$$U_i'/U_i = T_i / T_i'$$

P_c : collision probability

T : step processing time (w/o contention)

T' : step processing time (w/ contention)

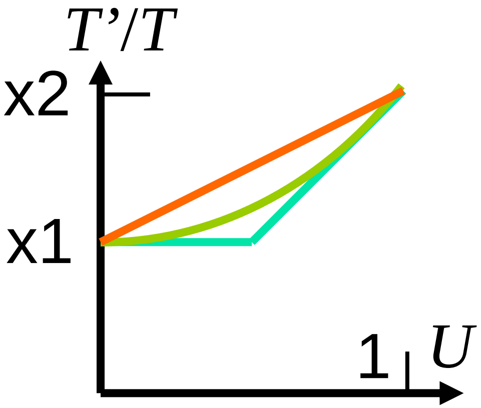
M : number of processors (i=1, …, M)

c : collision pattern index (2^M patterns)

$A_{c,i}$: ratio of access time at collision to original access time

Method: Numerical characteristics

- Numerical improvements in calculation of $A_{c,i}$ and P_c
  - $A_{c,i}$ → $A_{c,i}(U_i)$,
    - U dependence in round-robin arbitration
  - $P_c(U_1,\ldots,U_M)$ → $P_c(U_1^*,\ldots,U_M^*)$, where $U^* = \frac{(T'-T)+TU}{T'}$
    - Increase of bandwidth utilization due to arbitration
- Complexity: $O(M^2\, 2^M)$
- Effects of correlation
  - Positive correlation → upper bound 🟧
  - No correlation → our method 🟩
  - Negative correlation → lower bound 🟦



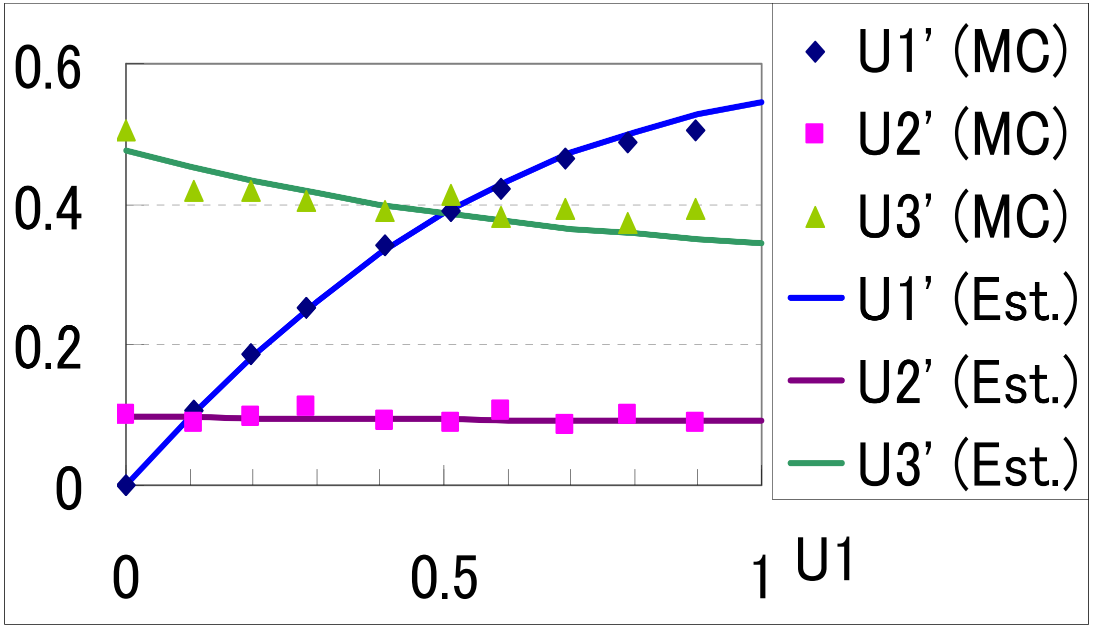
Method: Comparison with MC Simulation

Monte-Carlo (MC) simulation of Round-robin arbiter

Assumptions:

- Memory access occurs randomly
- No correlations between processors

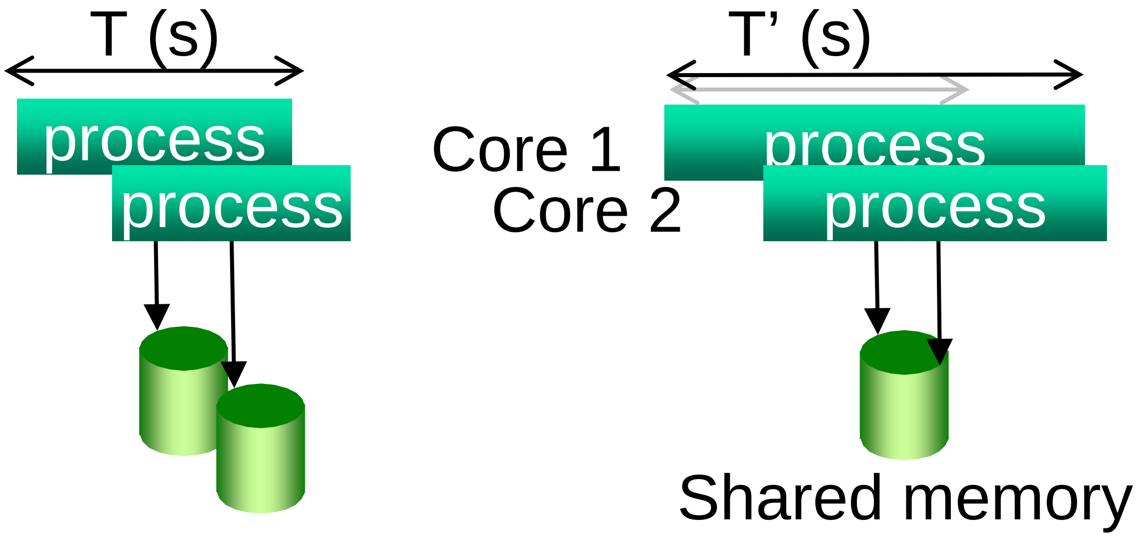
Assigned bandwidth utilization Ui'



Simulation Conditions

Number of processors	3
Request of utilization	U_1 = [0, 1], U_2 = 0.1, U_3 = 0.5
MC iterations	1000

Experiment: Overview



T (s)

process

process

Core 1

Core 2

T' (s)

process

process

Shared memory

Execute two processes

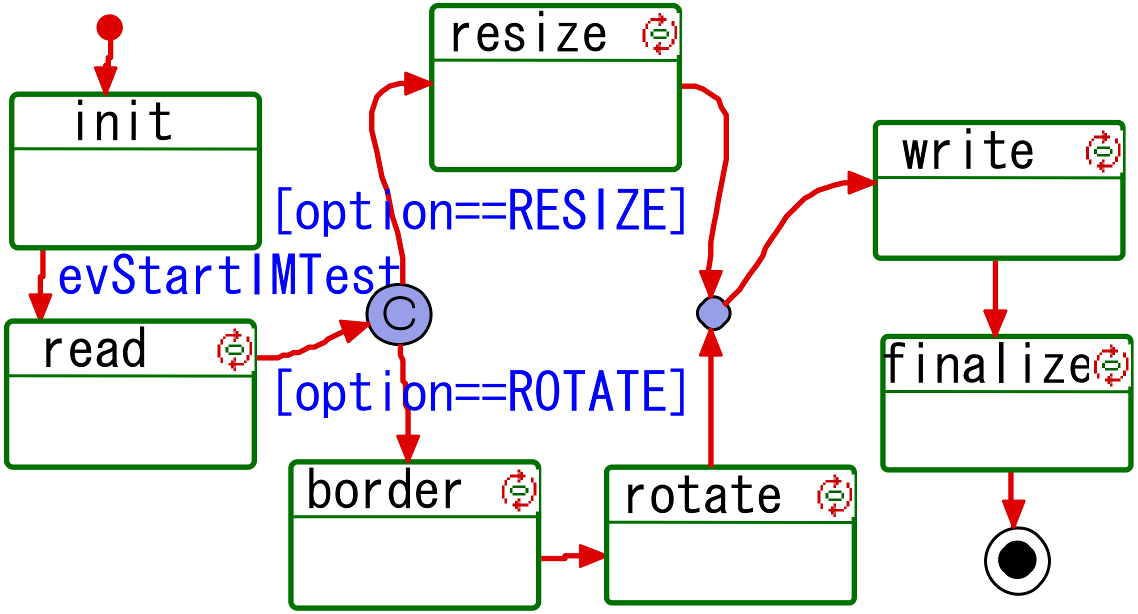
⇧

Memory access contention makes the execution time longer

Processor	Intel® Xeon™ 3.60GHz x 2
Main mem.	PC2-5300 DDR2 SDRAM, 2GB
L1 cache	16KB, 32byte block (I), 16KB, 64byte block (D)
L2 cache	1MB x 2, 64byte block

Experiment: Model

- Benchmarks are two image processing commands in ImageMagick software
  - Resize: enlarge a JPEG file by 150%
  - Rotate: rotate a JPEG image by 45 degrees



Experiment: Parameters

Program		N_R (x10^6)	N_W (x10^6)	T (s)	U
-resize	Read	0.001	1.252	0.56 +- 0.02	0.255
	Resize	23.542	7.936	6.49 +- 0.04	0.324
	Write	4.124	1.409	2.26 +- 0.03	0.166
-rotate	Read	0.001	1.252	0.56 +- 0.02	0.255
	Border	1.248	10.214	1.19 +- 0.04	1.03
	Rotate	15.645	6.157	16.5 +- 0.2	0.091
	Write	3.828	1.284	1.61 +- 0.02	0.212

- Throughputs: read=19.56+-0.07, write=8.76+-0.03 (x 10^6/s)
- Number of memory accesses is measured with Valgrind tool
  - Cache hits are subtracted from the above values of Ns

Experiment: Result

Program		Inc. (%)	T’(Exp.) (s)	T’(Est.) (s)	Error (%)
-resize	Read	-0.3	0.56+-0.02	0.59	5.2
	Resize	11.7	7.25+-0.2	7.00	-3.4
	Write	3.2	2.33+-0.05	2.32	-0.7
	(Total)	8.3	10.14+-0.3	9.91	-2.3
-rotate	Read	2.7	0.57+-0.01	0.59	3.1
	Border	8.2	1.28+-0.05	1.71	33.6
	Rotate	7.3	17.73+-0.2	16.74	-5.6
	Write	-0.2	1.61+-0.02	1.61	0.2
	(Total)	6.8	21.31+-0.2	20.66	-3.0

Discussion

- The accuracy can be improved by choosing an appropriate throughput parameter
  - DRAM throughput of accesses to sequential addresses is higher than that of accesses to random addresses
    - We adopt throughputs of random access in the simulation
    - However ‘Border’ step consists of sequential accesses
  - Improved estimate: T’ = 1.284s (error: 0.3%, U=0.316)

Parameters		N_R (x10^6)	N_W (x10^6)	T (s)	U
-rotate	Border	1.248	10.214	1.19 +- 0.04	1.03

Result		Inc. (%)	T’(Exp.) (s)	T’(Est.) (s)	Error (%)
-rotate	Border	8.2	1.28+-0.05	1.71	33.6

Summary

- We propose a method for the evaluation of the memory access contention
  - Access timings are approximated as random and uncorrelated
  - The method can be used in event-driven simulations
- Error of order of 3% is found in comparison of our estimate with experimental result
  - Error becomes larger if there are sequential accesses
- Future works
  - More comparison with actual embedded systems
  - Support for cache memory and out-of-order execution

Logos and trademarks

- Following terms are registered trademarks of International Business Machines Corporation in United States, other countries, or both:
    - IBM, Rational, and Rhapsody.
- Intel and Intel Xeon are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.
- Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.
- Other company, product, or service names may be trademarks or service marks of others.

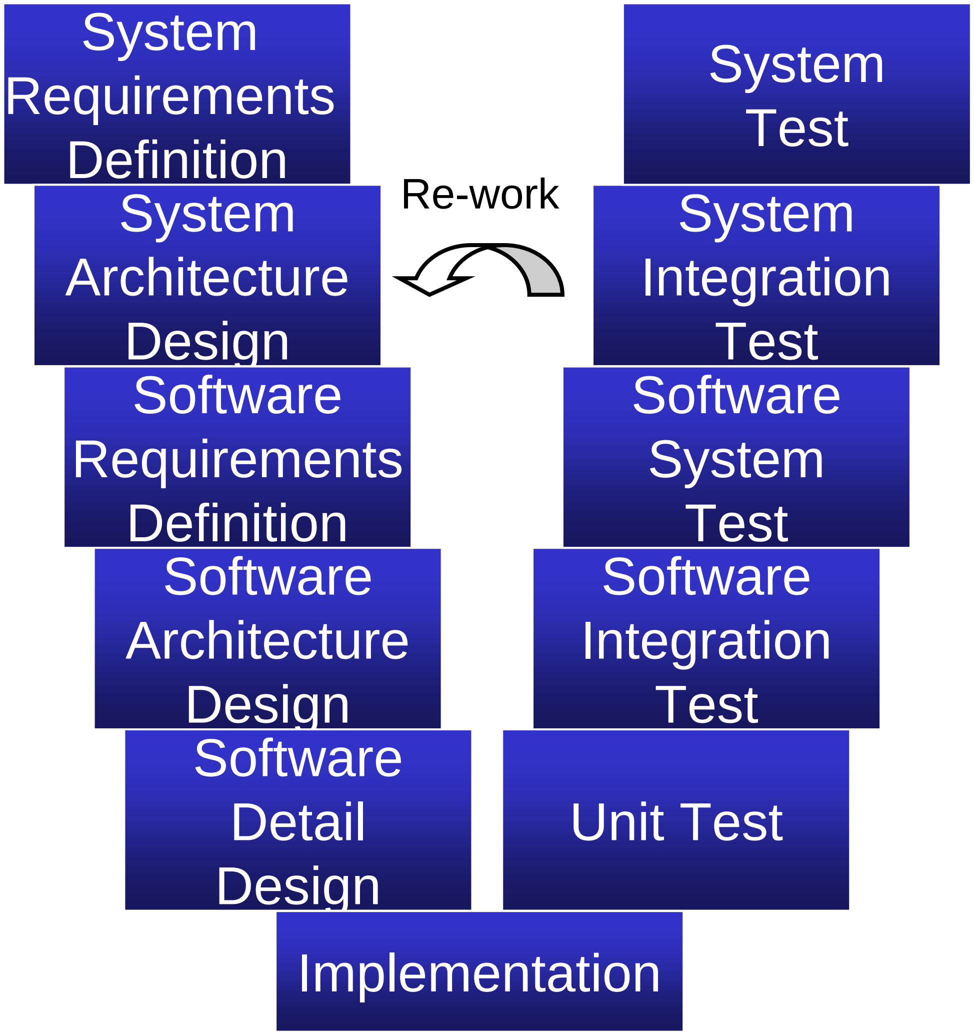
Table of Contents

- Introduction:
  - Application-level performance estimation at early stage of development
  - Memory Access Contention
- Related works:
  - simulation methods of memory access contention
  - UML-based simulation
- Method:
  - Main Idea
  - Model and parameter
  - Waiting time by arbitration
- Method: (continued)
  - Estimation of collision probability
  - Comparison with MC Simulation
- Experiment:
  - Overview
  - Model
  - Parameters
  - Result
- Discussion
- Summary

Introduction: V-Model and Modeling

UML is used to clarify software specification

It is difficult to evaluate application performance because it requires hardware resources to be taken into account



System Requirements Definition

System Architecture Design

Software Requirements Definition

Software Architecture Design

Software Detail Design

Implementation

Unit Test

Software Integration Test

Software System Test

System Integration Test

System Test

Re-work

Method: waiting time by arbitration

Generalization:

$$A_{c,i}(U_i) = \sum_{n=0}^{m(c)-2} E_n r_n(U_i) + E_{m(c)-1}(1-U_i)^{m(c)-1},$$

$$m(c) =$$

# of simultaneous access in pattern c

$$E_n = (m(c)-1)\frac{m(c)-n-1}{m(c)} + (n+1)\frac{2m(c)-n}{2m(c)}$$

$$r_n(U) = (1-U)^n U$$

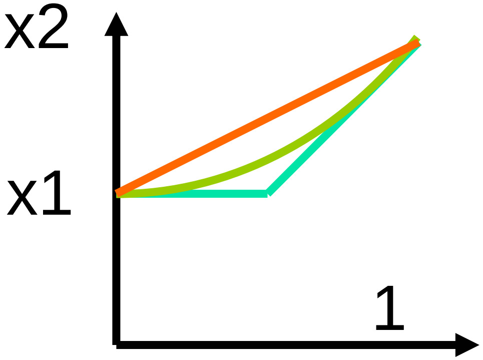
Prob. of access N cycles after current cycle

Relative access time at collision and successive access after n cycles

Method: Correlation of memory access

- Error due to correlation has upper and lower bounds

Relative increase of execution time T’/T

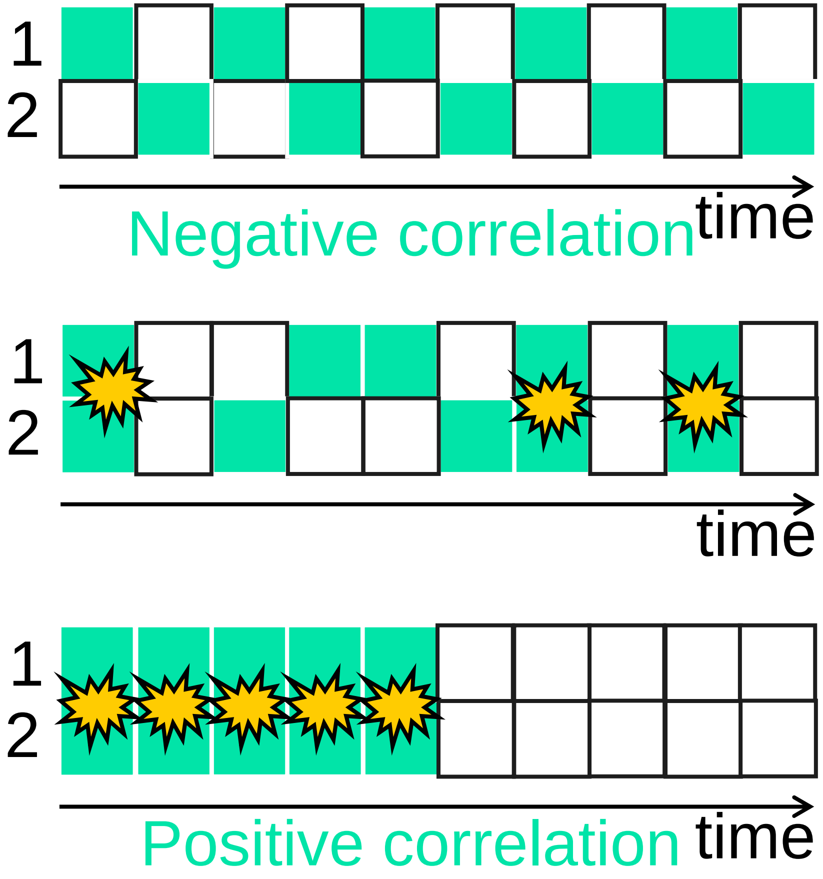


Utilization request U

Periodic access

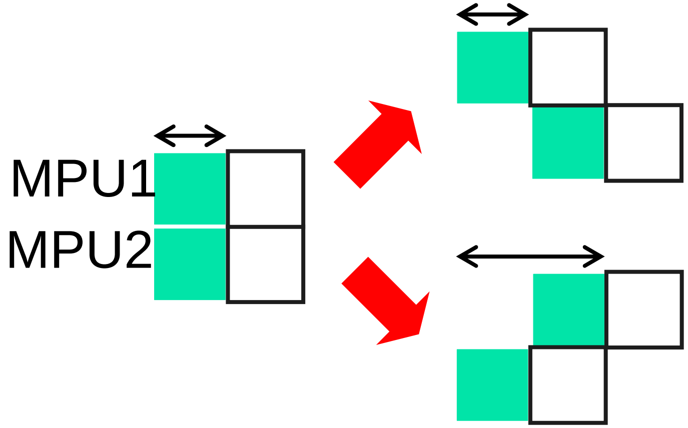
Random access

Cluster access



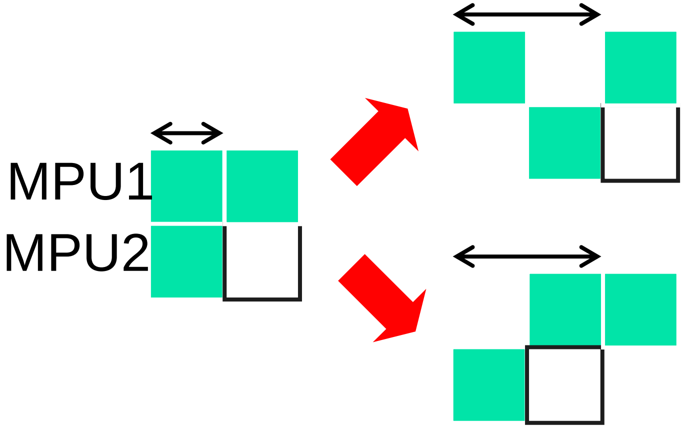
Method: waiting time by arbitration

(a) Without successive access



Average access time:
x 3/2

(b) With successive access



Average access time:
x 2

⇒ $$A = \frac{3}{2}(1 - U_1) + 2U_1$$ ⇐

(for round-robin arbitration case)

Method: Estimation of collision probability

- Simple estimation underestimates the increase of collisions due to arbitration

$$P_c(U_1,\cdots U_M)=\prod_{q=1}^{M}(1-U_q)^{n(q,c)}U_q{}^{b(q,c)}$$

$\Big($ $b(q,c)=1$ if q-th processor accesses in pattern c

$n(q,c)=1-b(q,c)$ $\Big)$

- Improved estimation uses iterative calculation ($O(M^2\,2^M)$)

$U^*=\{(T'-T)+TU\}/T'$: Bandwidth utilization taking into account the accesses waiting for arbitration

$$P_c(U_1,\cdots U_M)=\prod_{q=1}^{M}(1-U^*{}_q)^{n(q,c)}U^*{}_q{}^{b(q,c)}$$

$$T_i'/T_i=1+\sum_{c=0}^{2^M-1}(A_{c,i}-1)P_c(U_1,\cdots,U_M)$$