Scal-Tool: Pinpointing Scalability Bottlenecks

Yan Solihin, Vinh Lam, Josep Torrellas
Department of Computer Science
University of Illinois at Urbana Champaign
solihin, lam, torrella @cs.uiuc.edu
http://iacoma.cs.uiuc.edu

Problems in Programming DSMs

- Parallelizing compilers do not produce highly-scalable codes
- Programmers still need to hand-tune their code
- Performance tools are:
  - Cumbersome to use
  - Require large space
  - Unable to capture some high level info

Goals of Study

- Develop a model that pinpoints scalability bottlenecks:
  - Insufficient L2 cache size
  - Synchronization
  - Load imbalance

Proc / Size	1	2	4	…	32
So	X	X	X	…	X
So/2	X				
So/4	X				
…	…				

- The model should use few resources and be fast

Testing Platform

- SGI Origin 2000:
  - 32 - 250 MHz R10000 processors
  - 32 KB L1 I-Cache + 32 KB L1 D-Cache
  - 4 MB L2 Cache
  - Hardware event counters in the processor
- Parallelism model: MP + PCF
- Applications: t3dheat (Los Alamos), swim (SPEC95), hydro2d (SPEC95)

Basic Formulae

- 3 types of instructions:
  - Instr that access L2 cache (fraction = *h2*)
  - Instr that access main memory (fraction = *hm*)
  - the rest (fraction = *1-h2-hm*)

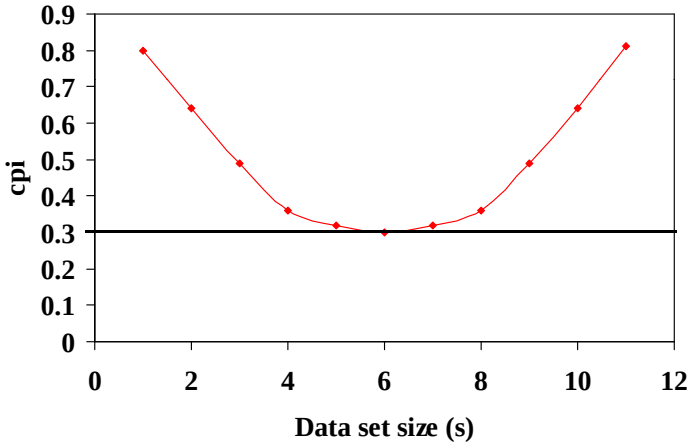
$$cpi = (1-h2-hm)\ cpi0 + h2\ t2 + hm\ tm$$

- *cpi*, *h2*, *hm* are measured directly with performance counters.
- *cpi0*, *t2*, *tm* have to be estimated

Parameter Estimation: *cpi0*

- Initial estimate = *cpi0**

  obtained by measuring cpi of a small data set size



cpi0*

- adjust the initial estimate:

$$cpi0 = (1 - h2 - hm)\, cpi0^* - h2\, \hat{t2} - hm\, \hat{tm}$$

- assume cpi0 constant for all applications

Parameter Estimation: *t2* and *tm*

- Assume t2 constant, tm = f(n) for an application
- Solve linear equation system:

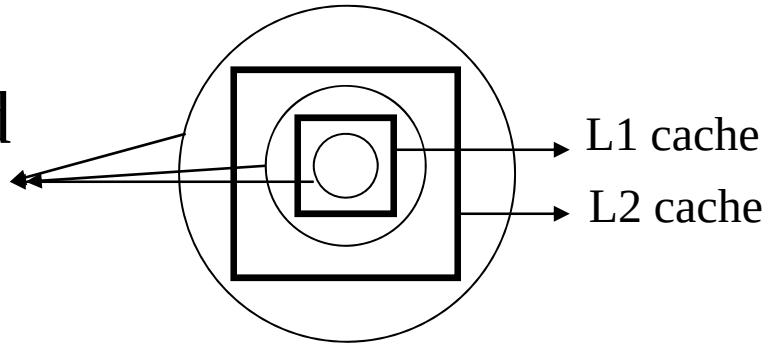
$$cpi^1 = (1 - h2^1 - hm^1)\, cpi0 * + h2^1\, t2 + hm^1\, tm$$

$$cpi^2 = (1 - h2^2 - hm^2)\, cpi0 * + h2^2\, t2 + hm^2\, tm$$

$$\ldots$$

$$cpi^n = (1 - h2^n - hm^n)\, cpi0 * + h2^n\, t2 + hm^n\, tm$$

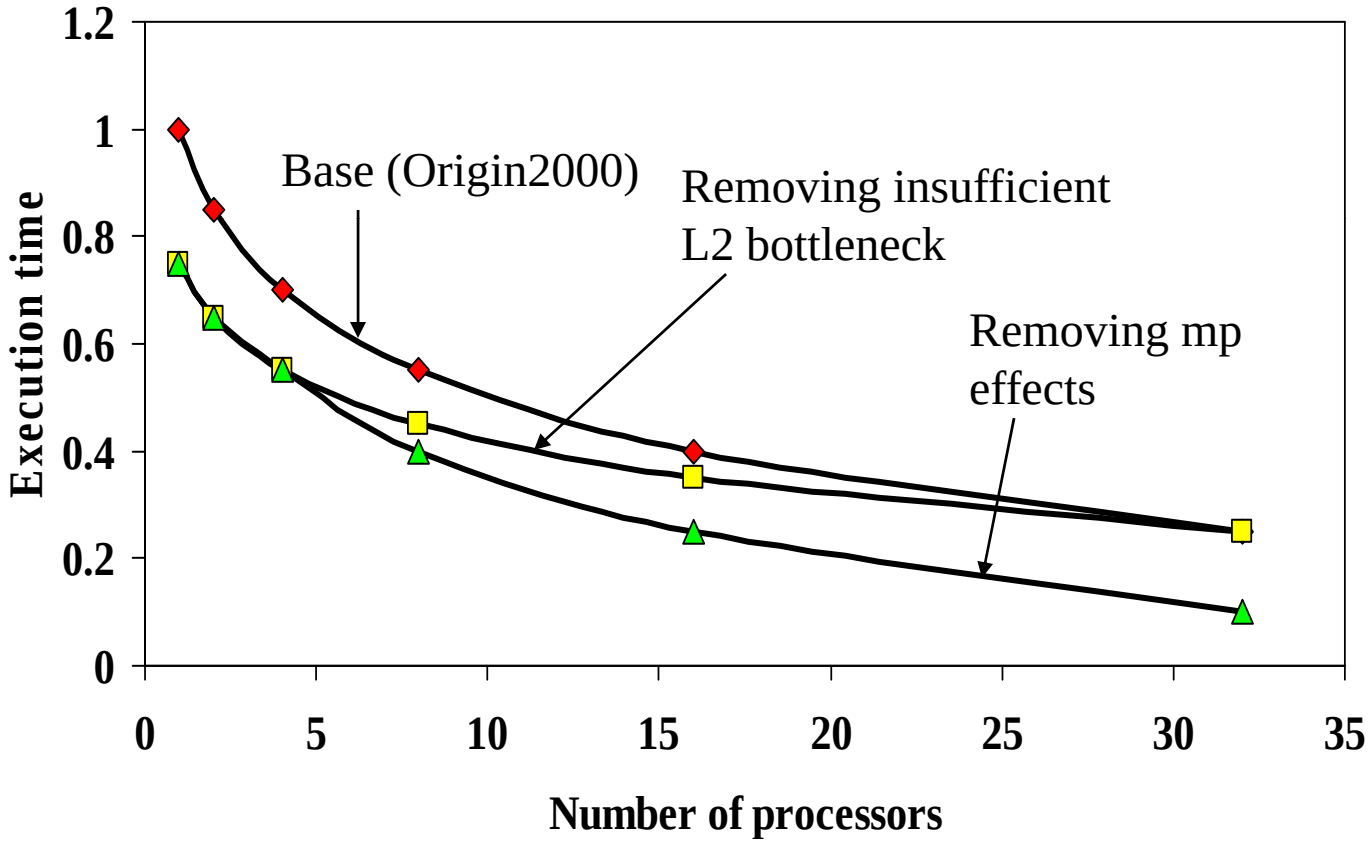
- Data points are obtained by varying data set size



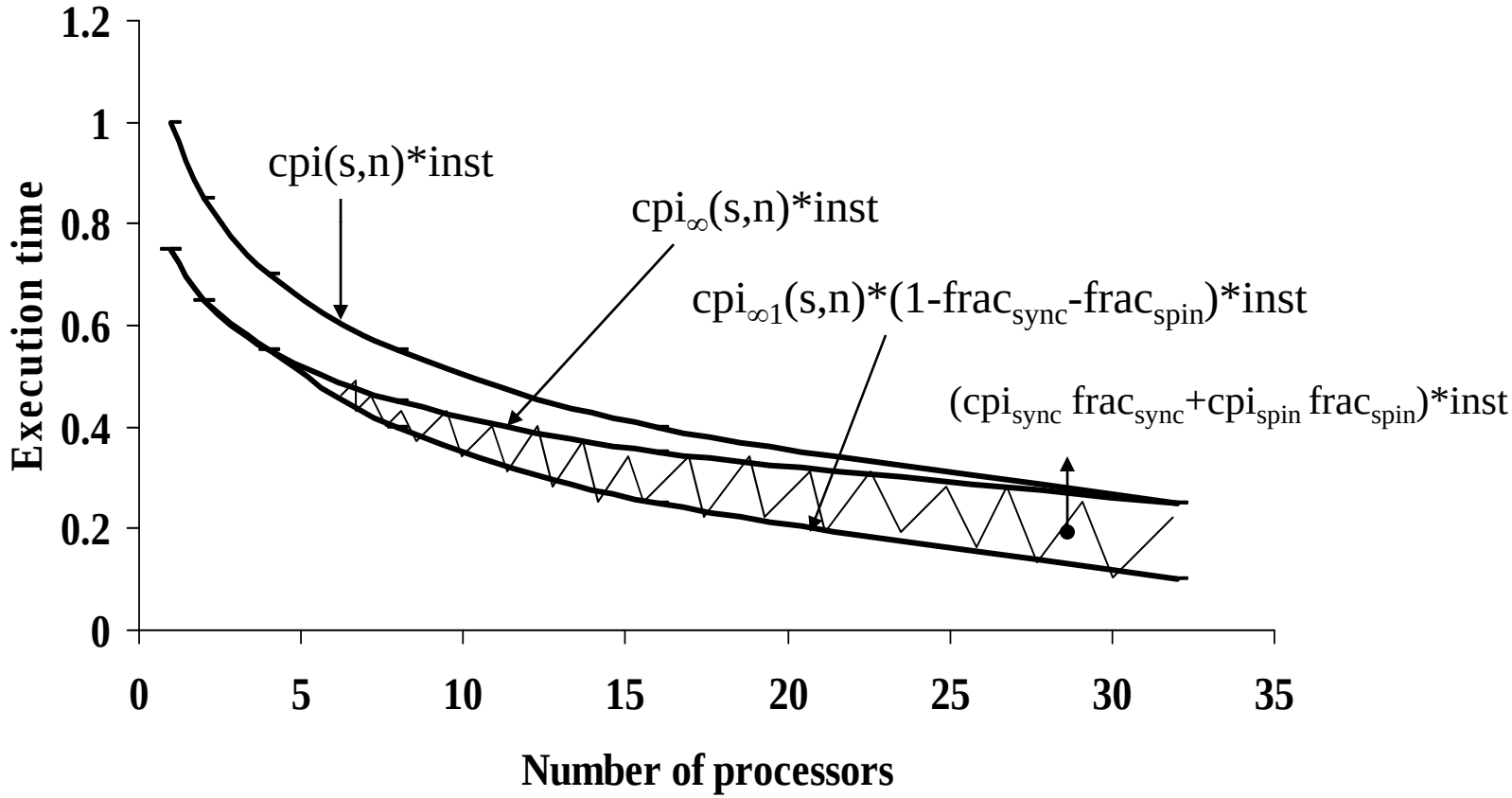
Bottlenecks

Bottlenecks	Results
1. *Insufficient L2 cache size*	Conflict misses
2. *MP Effects*	
- Synchronization	Coherence Misses + extra instructions
- Load imbalance	Extra instructions

Execution Time



Execution Time



Cpi and L2 Hit Rate

- where

$$h2(s,n) = (1 - L1hitr(s,n))\, m(s,n)\, L2hitr(s,n)$$

$$hm(s,n) = (1 - L1hitr(s,n))\, m(s,n)\, (1 - L2hitr(s,n))$$

$$m(s,n) = \frac{load + store}{inst}$$

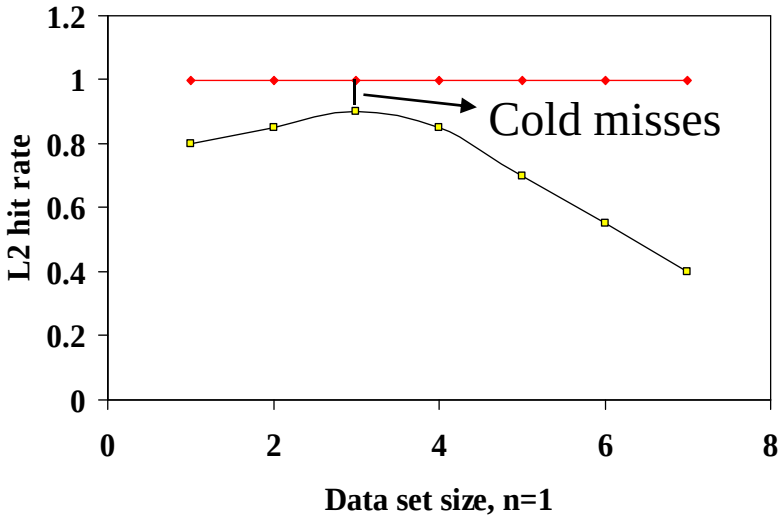
- Thus

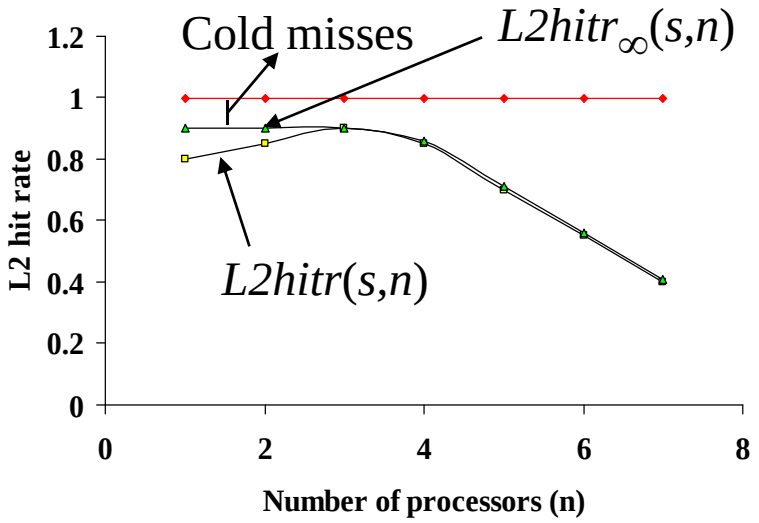
$$cpi(s,n) = f(L2hitr(s,n), tm(n))$$

Estimating $cpi_{\infty}(s,n)$

$$cpi(s,n) = f(L2hitr(s,n), tm(n))$$

- $cpi_{\infty}(s,n)$ needs $L2hitr_{\infty}(s,n)$

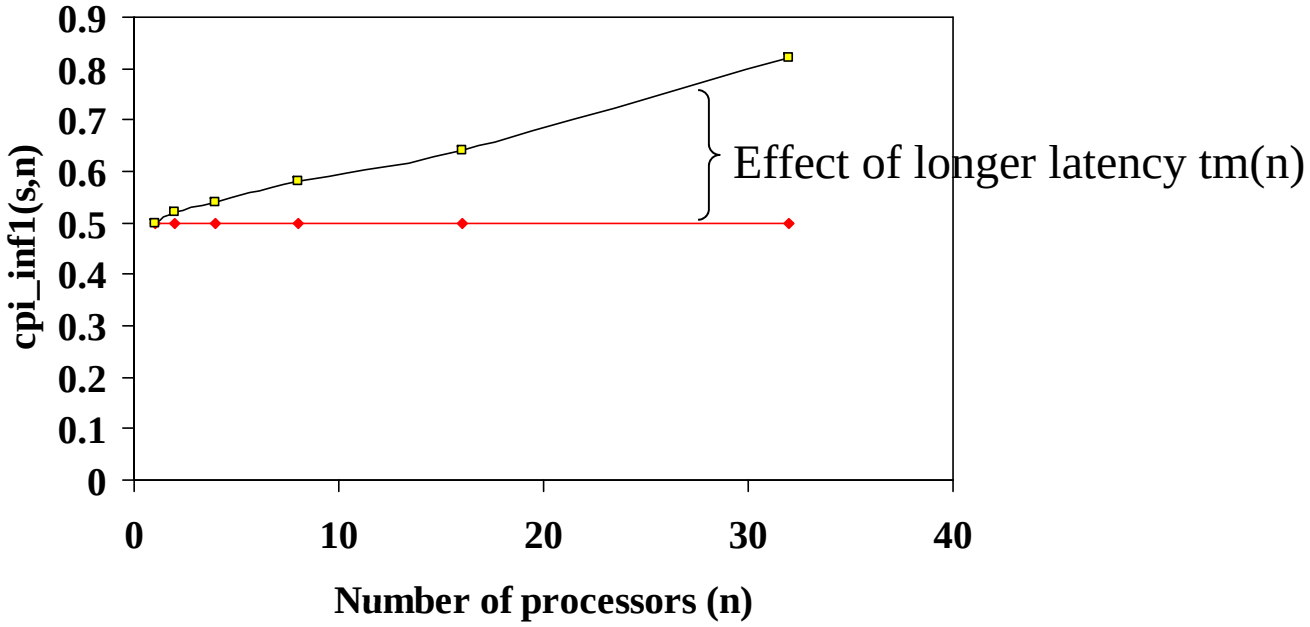




Estimating $cpi_{\infty 1}(s,n)$

$$cpi(s,n) = f(L2hitr(s,n), tm(n))$$

- $cpi_{\infty 1}(s,n)$ needs $L2hitr_{\infty 1}(s,n)$



Estimating $frac_{syn}$, $frac_{spin}$

$$cpi_{\infty}(s,n) = cpi_{\infty 1}(s,n)\,(1\text{-}frac_{syn}\text{-}frac_{spin}) + cpi_{syn} * frac_{syn} + cpi_{spin} * frac_{spin}$$

- cpi_{syn}, cpi_{spin}: measured kernel
- $frac_{syn}$, $frac_{spin}$: unknown, can be measured