TOBIAS GYSI, TOBIAS GROSSER, LAURIN BRANDNER, AND TORSTEN HOEFLER

A Fast Analytical Model of Fully Associative Caches

SPCL

The Cost of Data Movement Depends on Global State and Does Not Compose

```
int N = 1000;
for(int i = 0; i < N; i++) {
  for(int j = 0; j < i; j++) {
    for(int k = 0; k < j; k++) {
      A[i][j] -= A[i][k] * A[j][k];
    }
    A[i][j] /= A[j][j];
  }
  for(int k = 0; k < i; k++) {
    A[i][i] -= A[i][k] * A[i][k];
  }
  A[i][i] = sqrt(A[i][i]);
}
```

Cholesky kernel from https://sourceforge.net/projects/polybench/

percentage of cache misses?

L1 cache **1.6%**

L2 cache **1.4%**

most expensive memory access?

`A[j][k]`

amount of compulsory and capacity misses?

# compulsory misses	**31,752**
# capacity misses	**10,630,620**

HayStack Output for Cholesky Factorization

parameters:

- cache sizes (32k and 512k)
- cacheline size (64B)

relative number of cache misses (statement)

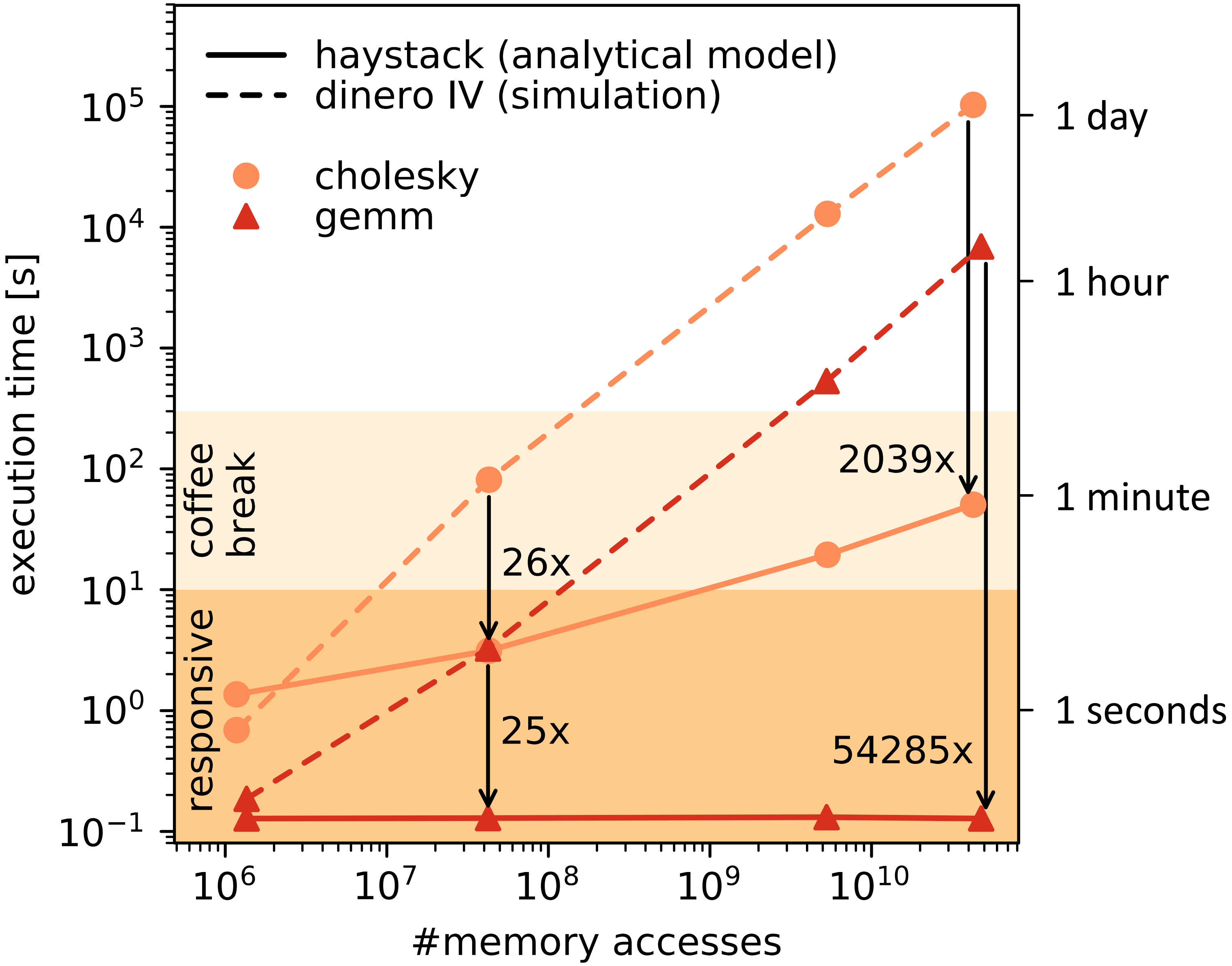
```
5   for (int i = 0; i < N; i++) {
6     for (int j = 0; j < i; j++) {
7       for (int k = 0; k < j; k++) {
8         A[i][j] -= A[i][k] * A[j][k];
```

ref	type	comp[%]	L1[%]	L2[%]	tot[%]	reuse[ln]
A[i][j]	rd	**0.00459**	0.00000	0.00000	24.86910	8,10
A[i][k]	rd	0.00000	0.00000	0.00000	24.86910	8,10
A[j][k]	rd	0.00000	**1.58635**	**1.38213**	24.86910	8,10,13,15
A[i][j]	wr	0.00000	0.00000	0.00000	24.86910	8

absolute number of cache misses (program)

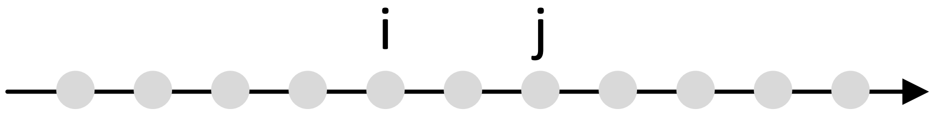
```
compulsory:                  31'752
capacity (L1):           10'630'620
capacity (L2):            9'258'460
total:                  668'166'500
```

Comparison to Simulation

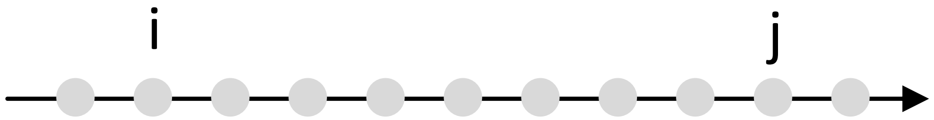


Symbolic Counting Avoids the Explicit Enumeration

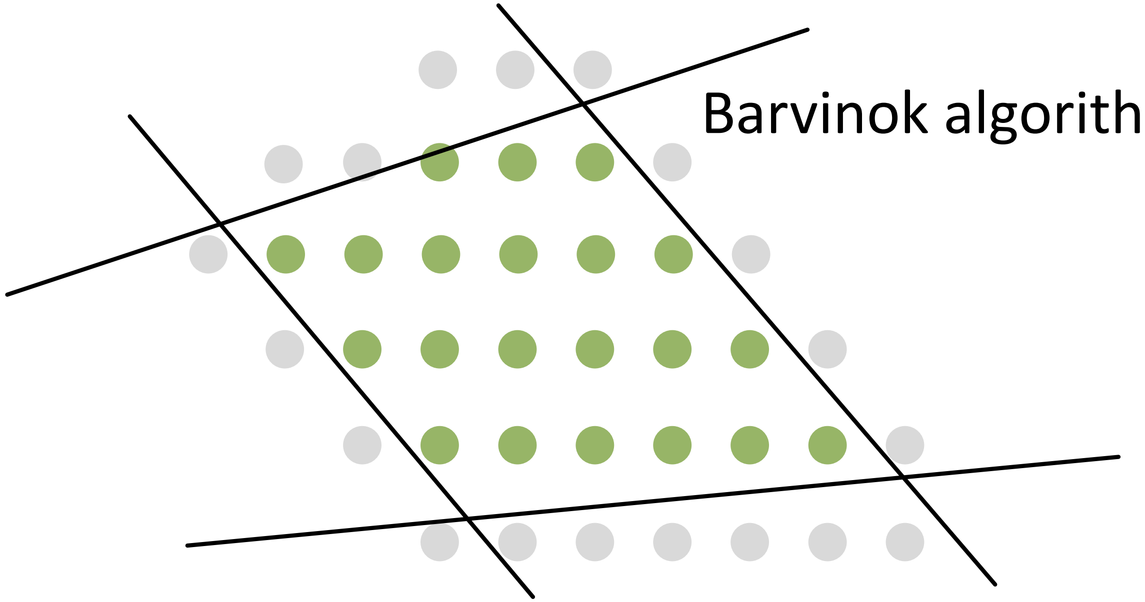
1d illustration



enumeration	#points = 3
symbolic	#points = j-i+1 = 3



enumeration	#points = 9
symbolic	#points = j-i+1 = 9



Barvinok algorithm

Alexander I. Barvinok, *A Polynomial Time Algorithm for Counting Integral Points in Polyhedra When the Dimension is Fixed*. 1994.

The LRU Stack Distance Allows Us to Model Fully Associative Caches

example

```
    int sum = 0;
    for(int i=0; i<4; ++i)
S0:   M[i] = i;
    for(int j=0; j<4; ++j)
S1:   sum += M[3-j];
```

deliberately **generic model**

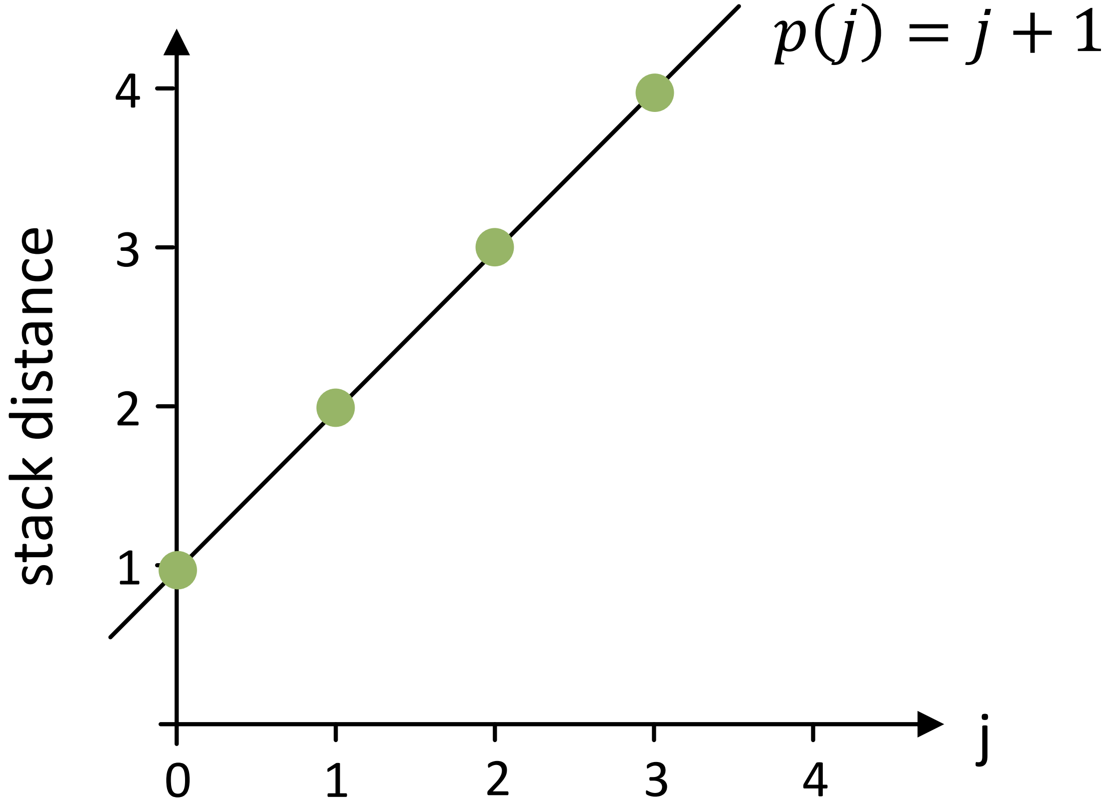
Richard L Mattson, Jan Gecsei, Donald R Slutz, and Irving L Traiger, *Evaluation techniques for storage hierarchies.* 1970.

Compute the LRU Stack Distance

example

```
     int sum = 0;
     for(int i=0; i<4; ++i)
S0:    M[i] = i;
     for(int j=0; j<4; ++j)
S1:    sum += M[3-j];
```

apply **symbolic counting** once



Kristof Beyls and Erik H D'Hollander, *Generating cache hints for improved program efficiency*. 2005.

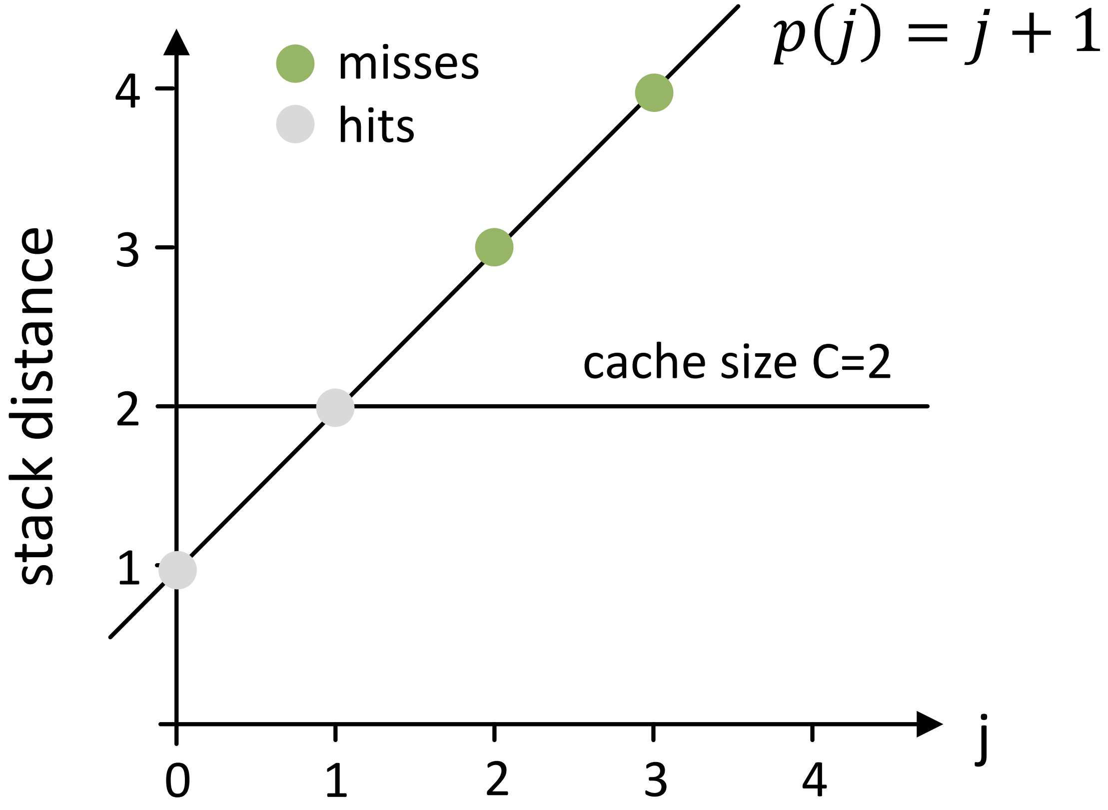
Count the Cache Misses Given the LRU Stack Distance

example

```
     int sum = 0;
     for(int i=0; i<4; ++i)
S0:    M[i] = i;
     for(int j=0; j<4; ++j)
S1:    sum += M[3-j];
```

apply **symbolic counting** twice

many **different** pieces and sometimes **non-affine** polynomials



$$|\{j : p(j) > C \wedge 0 \leq j < 4\}| = 2$$

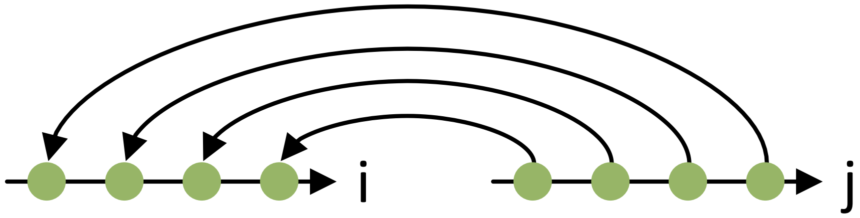
Some Access Patterns Result in Non-Linearities

example

```
     int sum = 0;
     for(int t=0; t<4; ++t) {
       for(int i=0; i<4; ++i)
S0:      M[i] = i;
       for(int m=0; m<t; ++m)
         for(int n=0; n<t; ++n)
           N[m][n] = t;
       for(int j=0; j<4; ++j)
S1:      sum += M[3-j];
     }
```

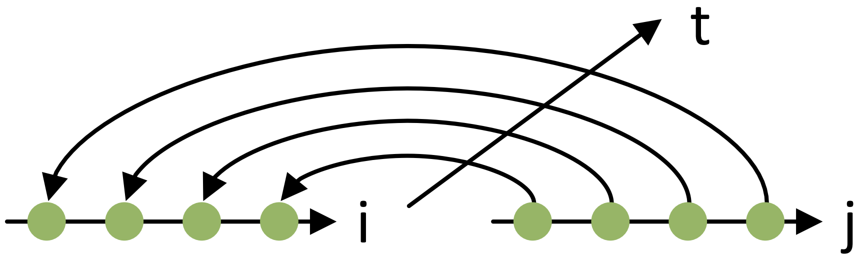
partial enumeration

original



$$p(j) = j + 1$$

additional **time loop**



$$p(j,t) = \boldsymbol{t^2} + j + 1$$

Enumerate the Non-Affine Dimensions

$p(j,t) = \mathbf{t^2} + j + 1$

(0,0)	(1,0)	(2,0)
(0,1)	(1,1)	(2,1)
(0,2)	(1,2)	(2,2)

$0 \leq (j,t) < 3$

π_t →

$0 \leq t < 3$

0
1
2

$p_{t=0}(j) = \mathbf{0} + j + 1$

0	1	2

$p_{t=1}(j) = \mathbf{1} + j + 1$

0	1	2

$p_{t=2}(j) = \mathbf{4} + j + 1$

0	1	2

12.4x speedup
due to **partial enumeration**

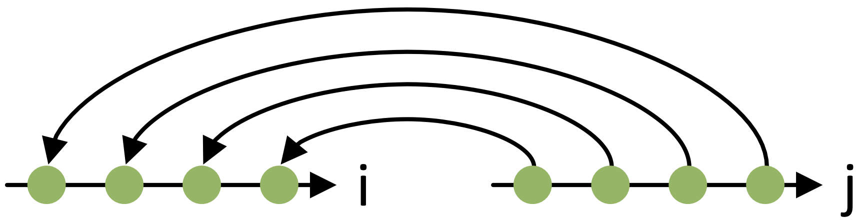
Modelling Cache Lines Introduces Floor Terms

example

```
     int sum = 0;
     for(int i=0; i<4; ++i)
S0:    M[i] = i;
     for(int j=0; j<4; ++j)
S1:    sum += M[3-j];
```

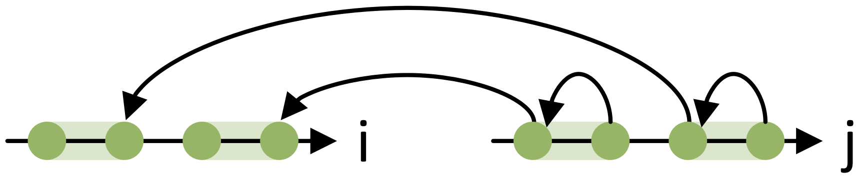
equalization and **rasterization**

original



$$p(j) = j + 1$$

modelling cache lines



$$p(j) = \frac{j}{2}\left(\left\lfloor\frac{j}{2}\right\rfloor - \left\lfloor\frac{j-1}{2}\right\rfloor\right) + 1$$

Split the Domain to Eliminate Floor Terms

$$p(j) = \frac{j}{2}\left(\left\lfloor \frac{j}{2} \right\rfloor - \left\lfloor \frac{j-1}{2} \right\rfloor\right) + 1$$

0 | 1 | 2 | 3

$$0 \leq j < 4$$

$$p(j)_{j\%2=0} = \frac{j}{2}\mathbf{1} + 1$$

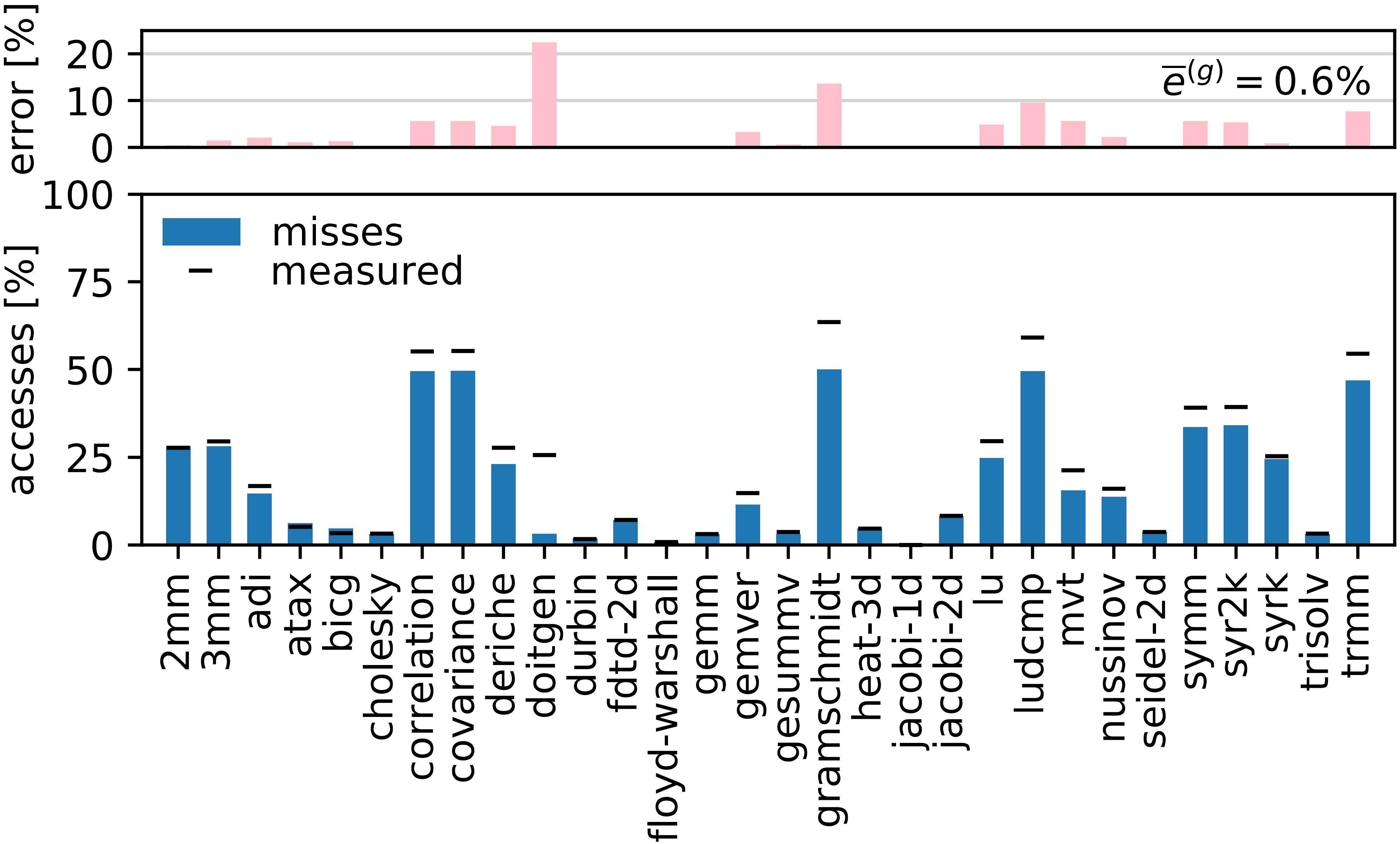
0 | 2

$$p(j)_{j\%2>0} = \frac{j}{2}\mathbf{0} + 1$$

1 | 3

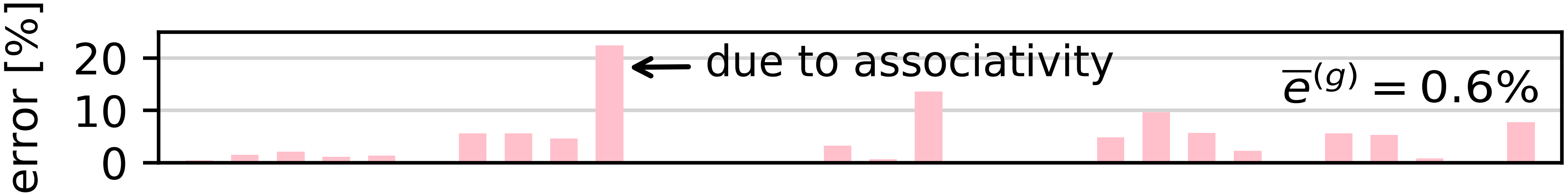
1.9x speedup
due to **equalization**

Accuracy of HayStack for the L1 Cache of Our Test System

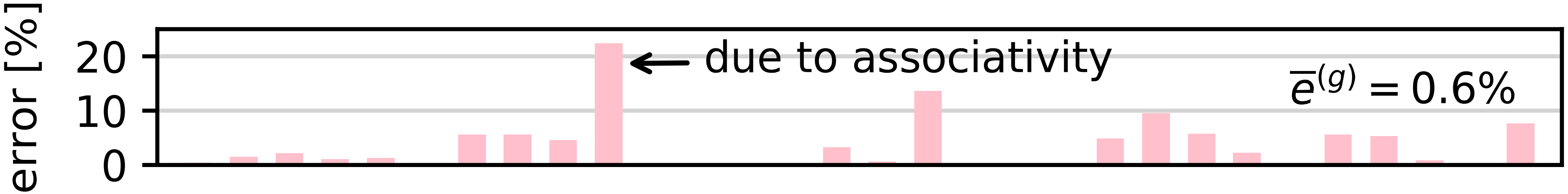


Error of HayStack Compared to Simulation (Dinero IV)

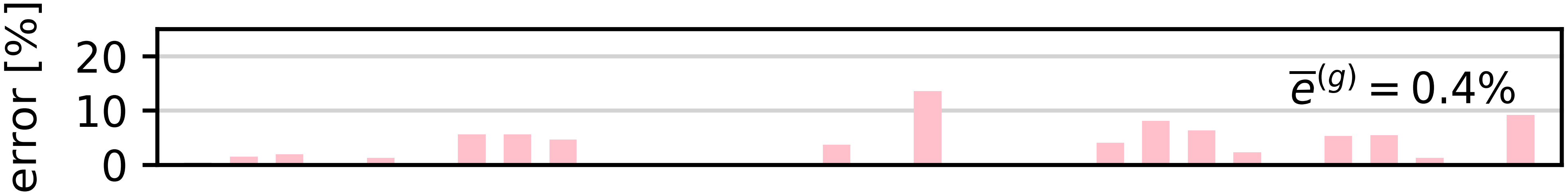
HayStack
(fully associative)



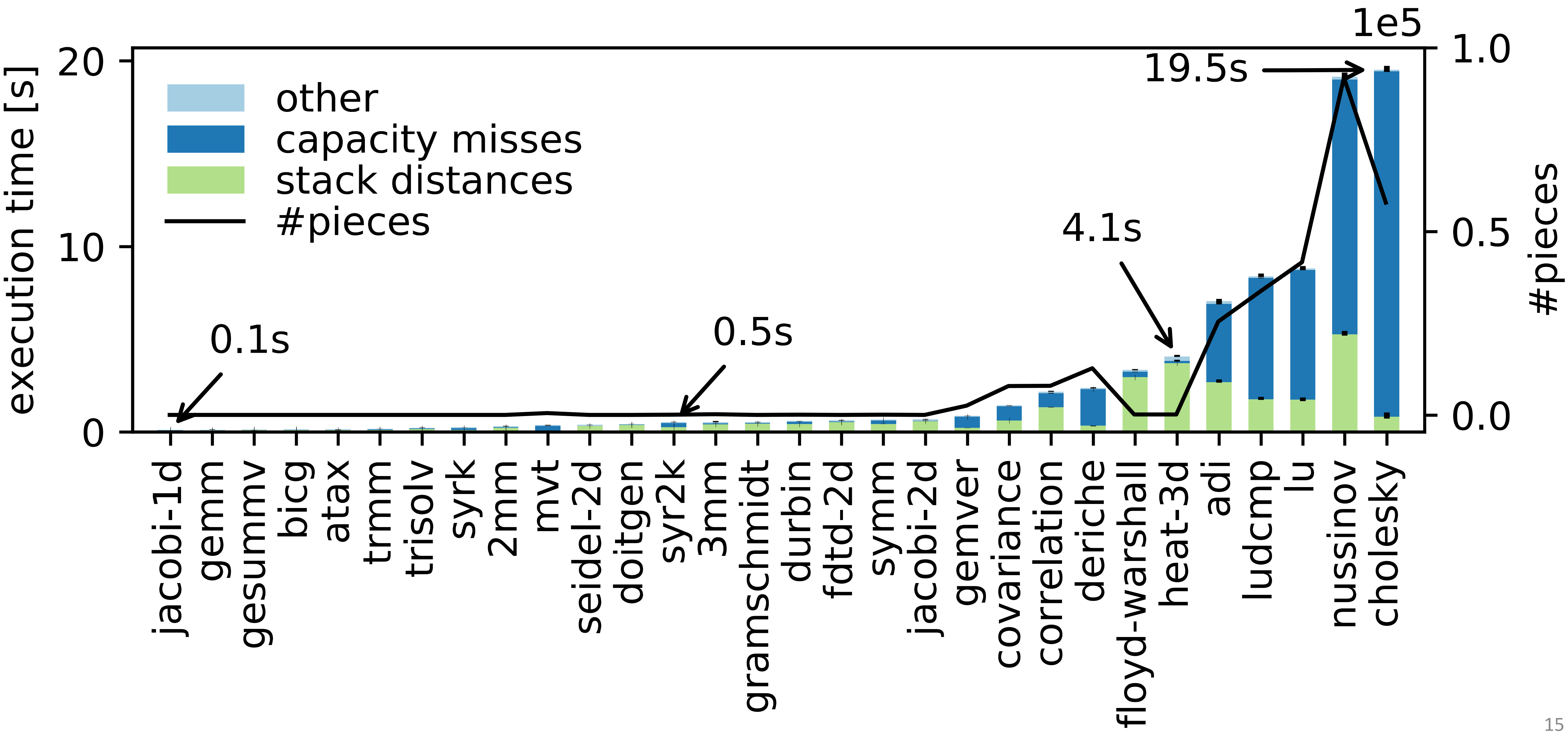
Dinero IV
(fully associative)



Dinero IV
(8-way associative)



Performance of HayStack for the Large Problem Size of PolyBench

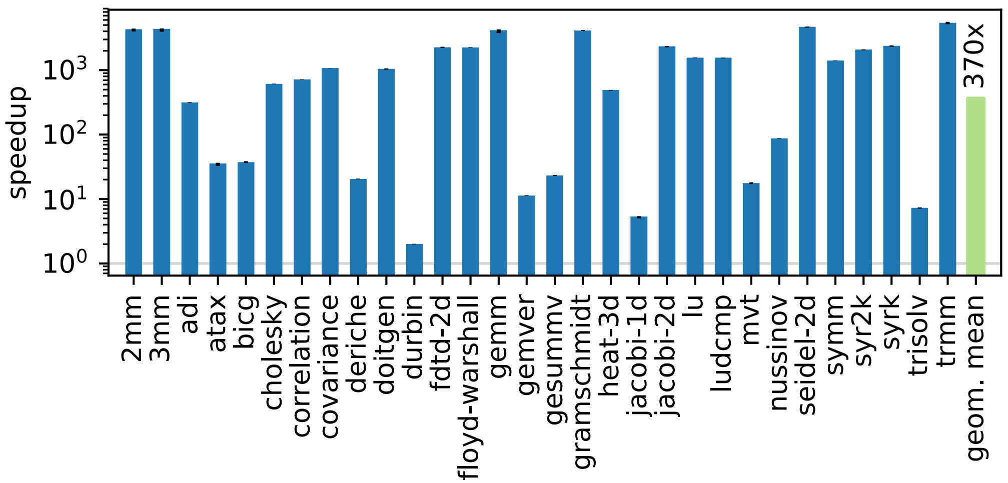


Performance of HayStack Compared to PolyCache and Dinero

Dinero IV

- simulator
- setup to simulate full associativity
- problem size dependent performance

370x speedup

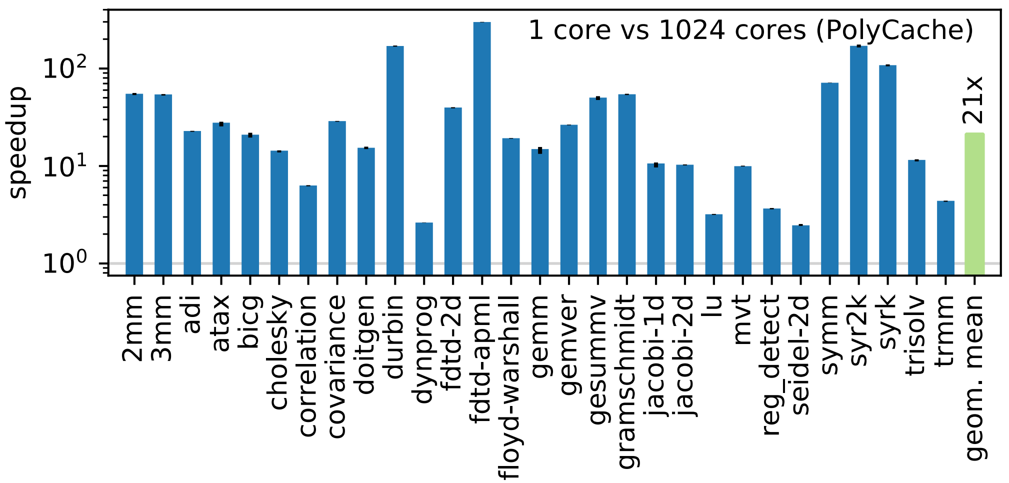


Jan Elder and Mark D. Hill, *Dinero IV Trace-Driven Uniprocessor Cache Simulator*. 2003.

PolyCache

- analytical cache model
- models set associativity
- one core per cache set

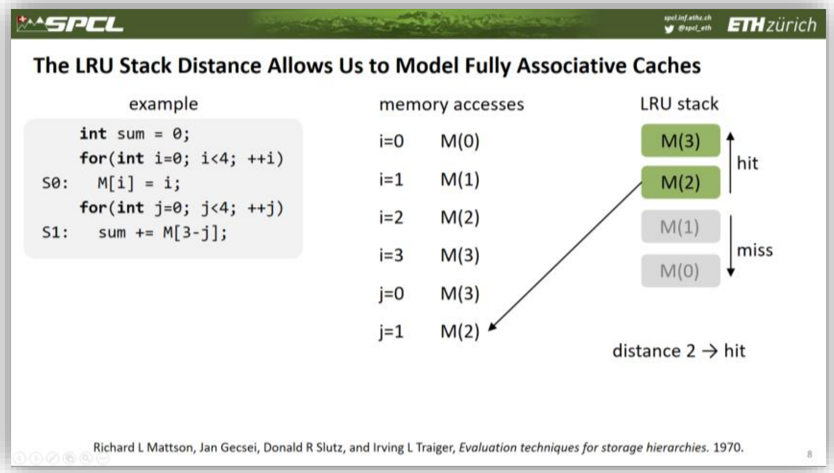
21x speedup



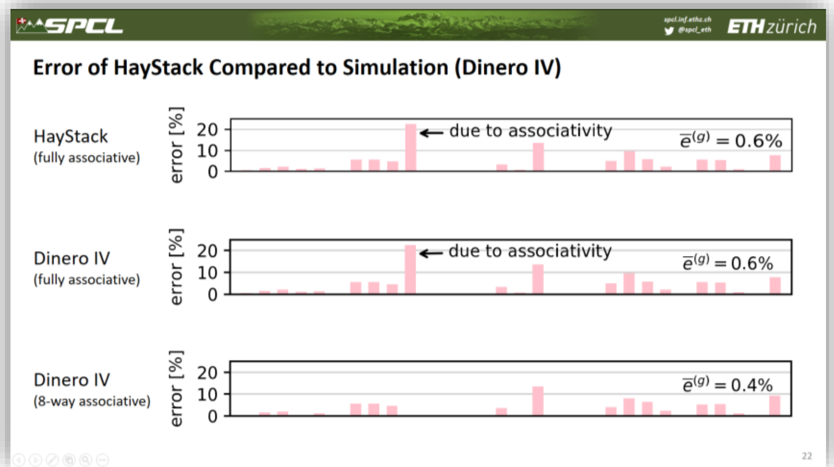
Wenlei Bao, Sriram Krishnamoorthy, Louis-Noel Pouchet, and P Sadayappan, *Analytical modeling of cache behavior for affine programs.* 2017.

Conclusion

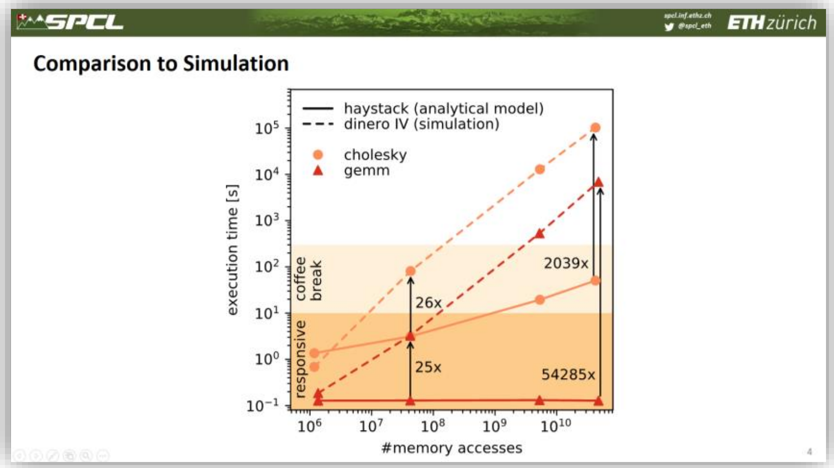
generic model of **fully associative caches**



accurate results compared to measurements



fast enough to provide **interactive feedback**



excellent performance compared to **alternatives**