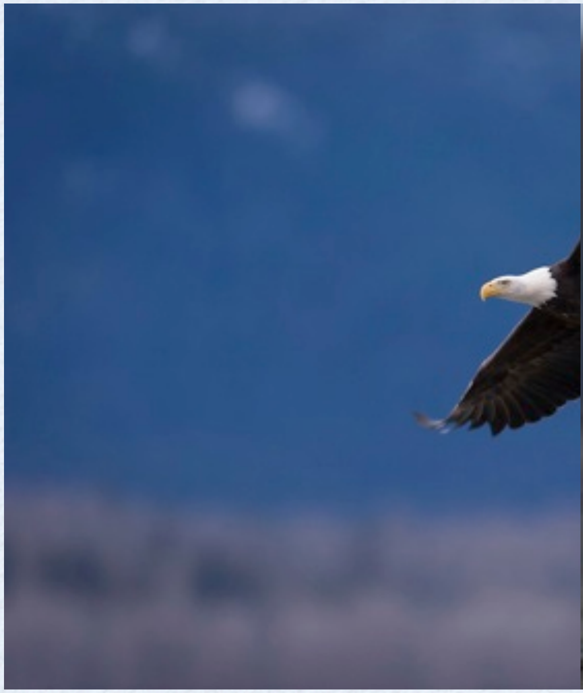
Defensive Loop Tiling for Shared Cache

Bin Bao
Adobe Systems
Chen Ding
University of Rochester

Bird and Program

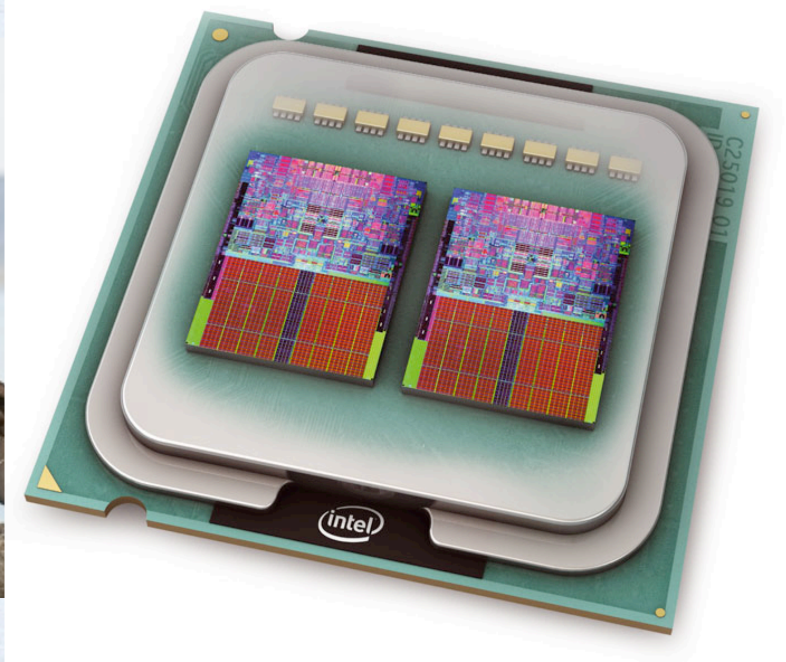
- “Unlike a bird, which can learn to fly better and better, existing programs are sort of dumb---the one millionth run of a program is typically not a bit better than the first-time run.” --- Professor Xipeng Shen @ W&M

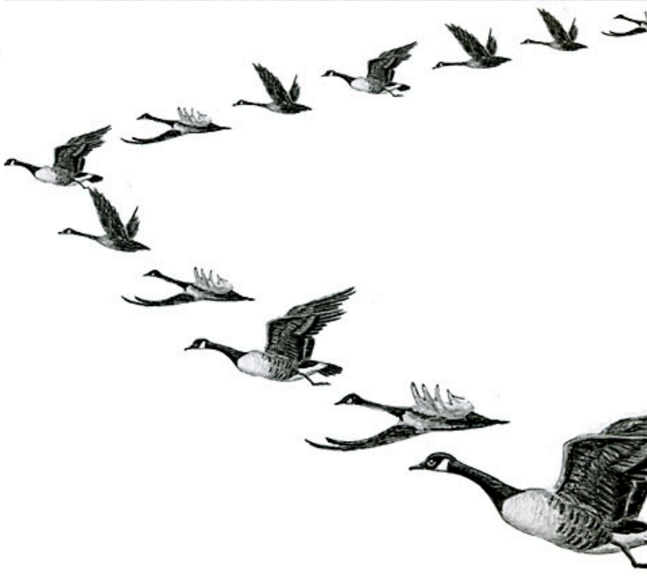


Peer Interaction

- Interfering
  - Limited resources

- Collaborative
  - Parallel tasks





- Peers: threads, tasks, and independent programs

Co-Run Program Optimization

- Existing shared-cache optimization
  - Cache partitioning
  - Job scheduling
  - Task throttling
- Compiler optimization?

Loop Tiling --- A Matrix Multiplication Example

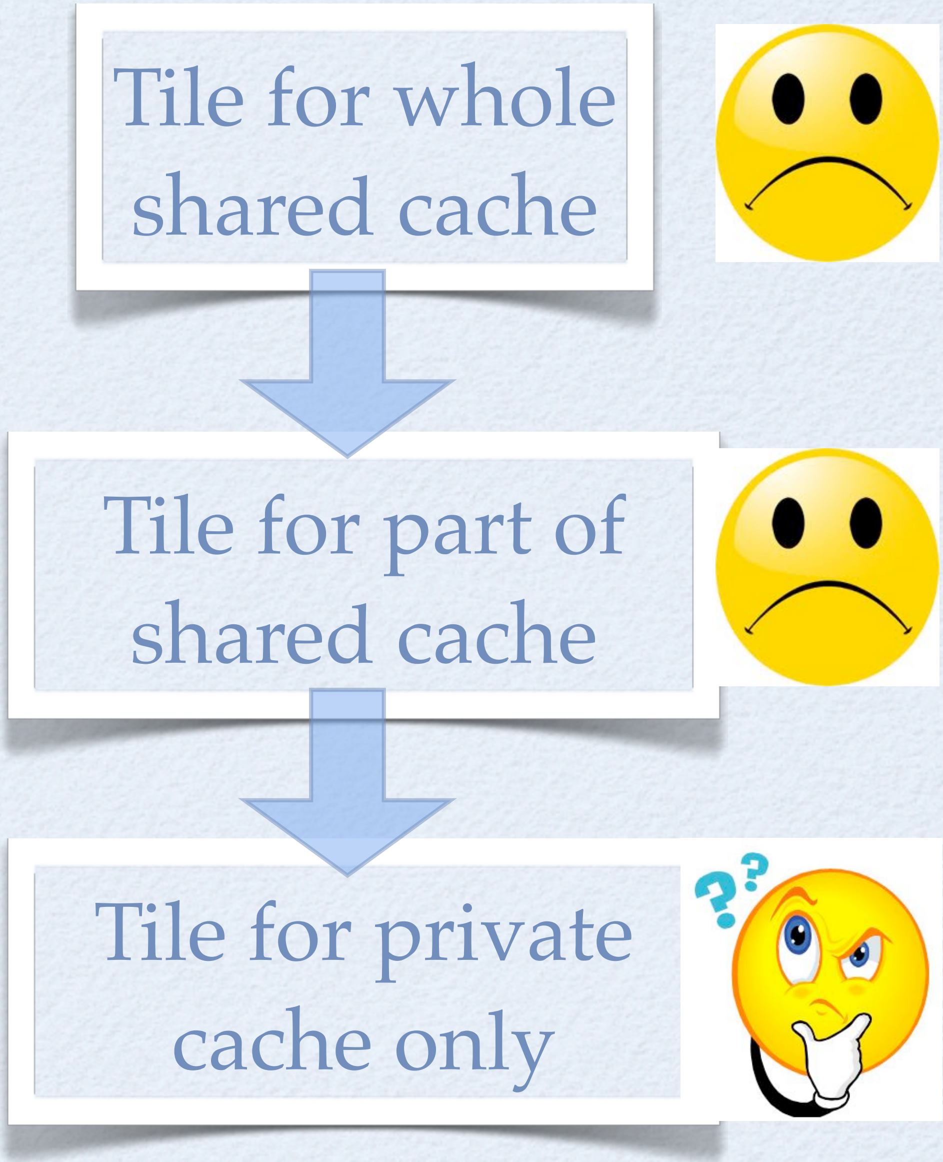
```
$for(i = 0; i < N; i = i + 1)$
 $for(j = 0; j < N; j = j + 1)$
  $for(k = 0; k < N; k = k + 1)$
   $C[i][j] = beta * C[i][j] + alpha * A[i][k] * B[k][j];$
```

(a) Original code

```
$for(jj = 0; jj < N; jj = jj + B_j)$
 $for(kk = 0; kk < N; kk = kk + B_k)$
  $for(i = 0; i < N; i = i + 1)$
   $for(j = jj; j < min(jj + B_j, N); j = j + 1)$
    $for(k = kk; k < min(kk + B_k, N); k = k + 1)$
     $C[i][j] = beta * C[i][j] + alpha * A[i][k] * B[k][j];$
```

(b) Tiled code

Tiling Strategy for Shared Cache



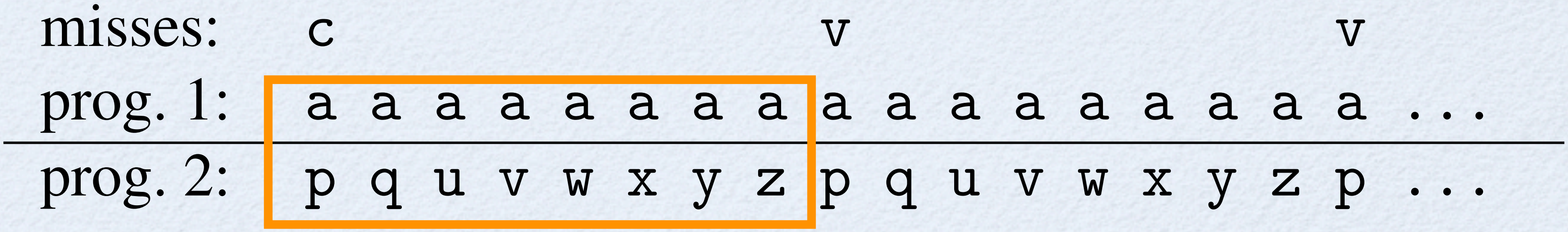
Tile for whole shared cache

Tile for part of shared cache

Tile for private cache only

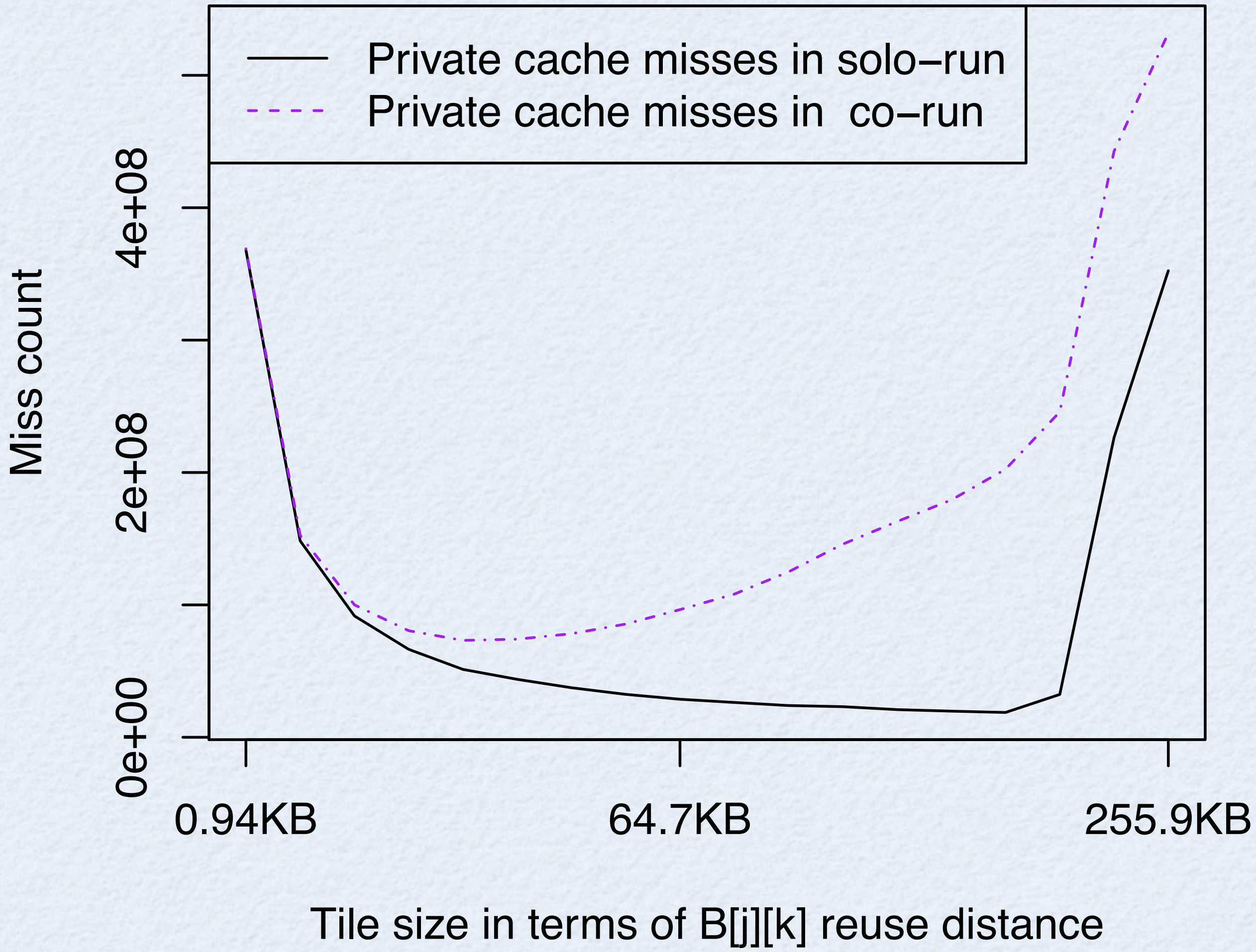
Inclusion Victim Misses

- Inclusive cache
  - E.g. L3 cache in Intel Nehalem processor
- Inclusive victim [Jaleel et al. MICRO’10]
  - A toy example: L1 cache size 2; L2 cache size 8



```
misses:   c               v               v
prog. 1:  a a a a a a a a a a a a a a a a a . . .
prog. 2:  p q u v w x y z p q u v w x y z p . . .
```

Matrix Multiplication Results on a Cache Simulator

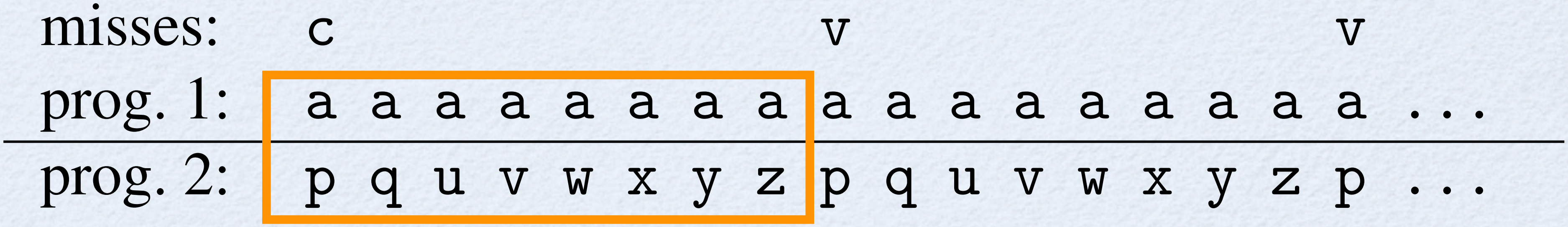


- 2 cores
- Private 256KB L1 cache
- Shared 2MB L2 cache
- Matmul and streaming

Inclusion Victim Modeling

- Data usage
  - Reused data, active period
- Cache interference
  - Survival window

$$iv(p_1) = \frac{ap(p_1)}{sw(p_1 + p_2)} * reuse(p_1)$$



```
misses:   c                       v                       v
prog. 1:  a  a  a  a  a  a  a  a  a  a  a  a  a  a  a  a  a  ...
prog. 2:  p  q  u  v  w  x  y  z  p  q  u  v  w  x  y  z  p  ...
```

Implementation in Open64 Compiler

- Wolf, Maydan, and Chen. IJPP, 26(4):479–503, 1998.
  - A cache cost function
- Example: matrix multiplication
  - Footprint

$$F_i = 8 * (N * B_k + B_j * B_k + N * B_j)$$

$$F_j = 8 * (B_k + B_j * B_k + B_j)$$

  - Reuse

$$reuse_j = F_j - (F_i - F_j)/N$$

Implementation in Open64 Compiler (cont.)

- Original cache miss equation

$$CM_j = \frac{F_i}{N} + (\alpha * \frac{R_i}{ecsz} + \beta * \frac{|R_i - ecsz|^+}{ecsz}) * reuse_j$$

- Cache misses caused by inclusion victim

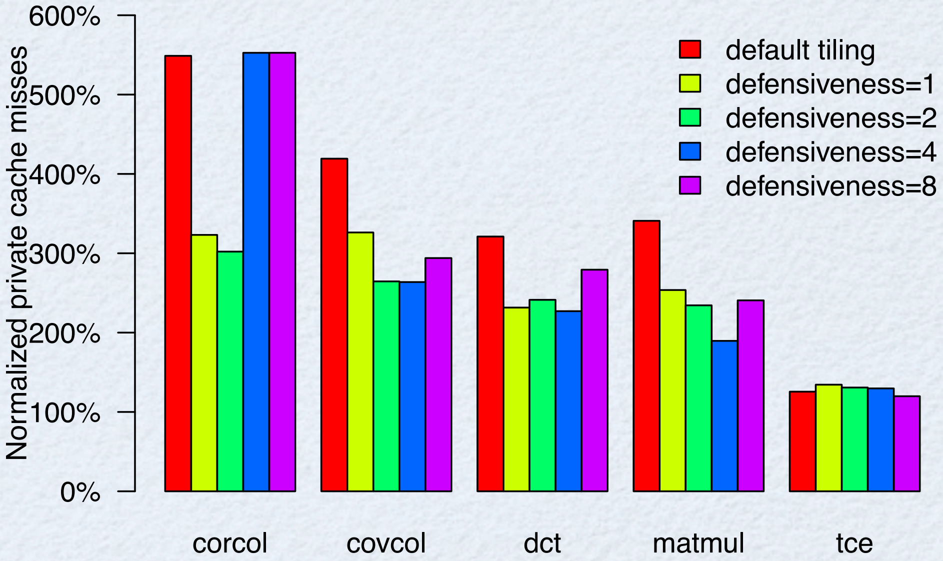
$$\boxed{IV_j = \frac{F_i}{scsz/\gamma} * reuse_j}$$

- γ is the defensiveness parameter

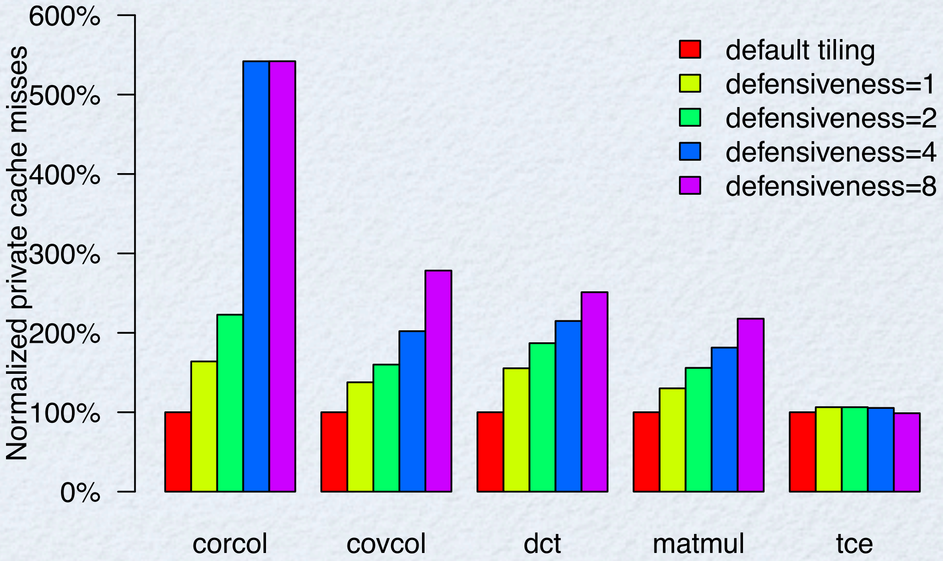
Experimental Results

- PLUTO benchmarks
- Pin-based cache simulator
  - 256KB private L1, 2MB shared L2
- Intel Nehalem processor
  - private 32KB L1 and 256KB L2, shared 8MB L3
- Co-run peers
  - STREAM benchmark, in addition to PLUTO

- Effect on private cache miss
- Baseline: default tiling on solo-run
- 4 defensiveness values



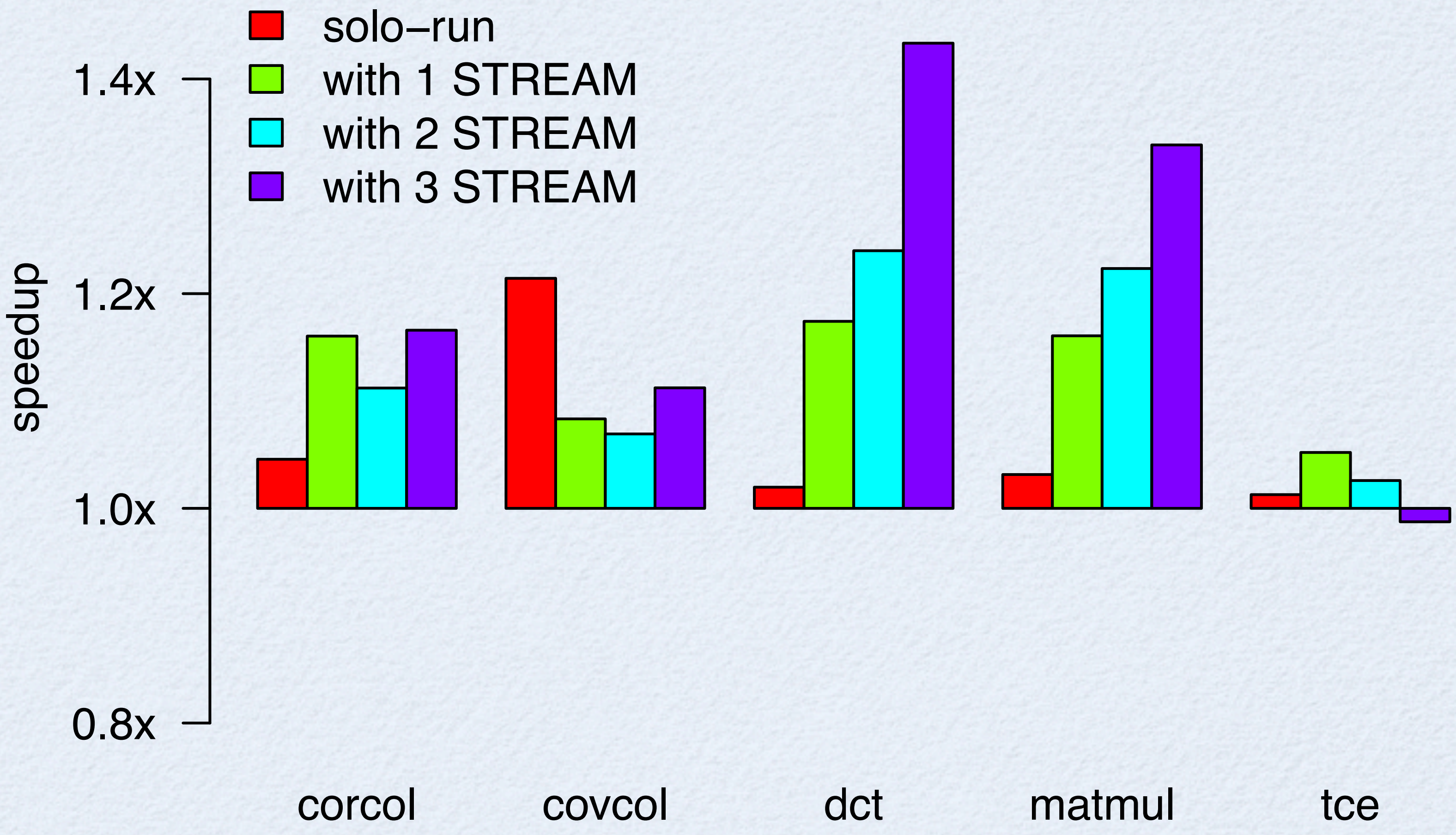
(a) Co-run simulation result



(b) Solo-run simulation result

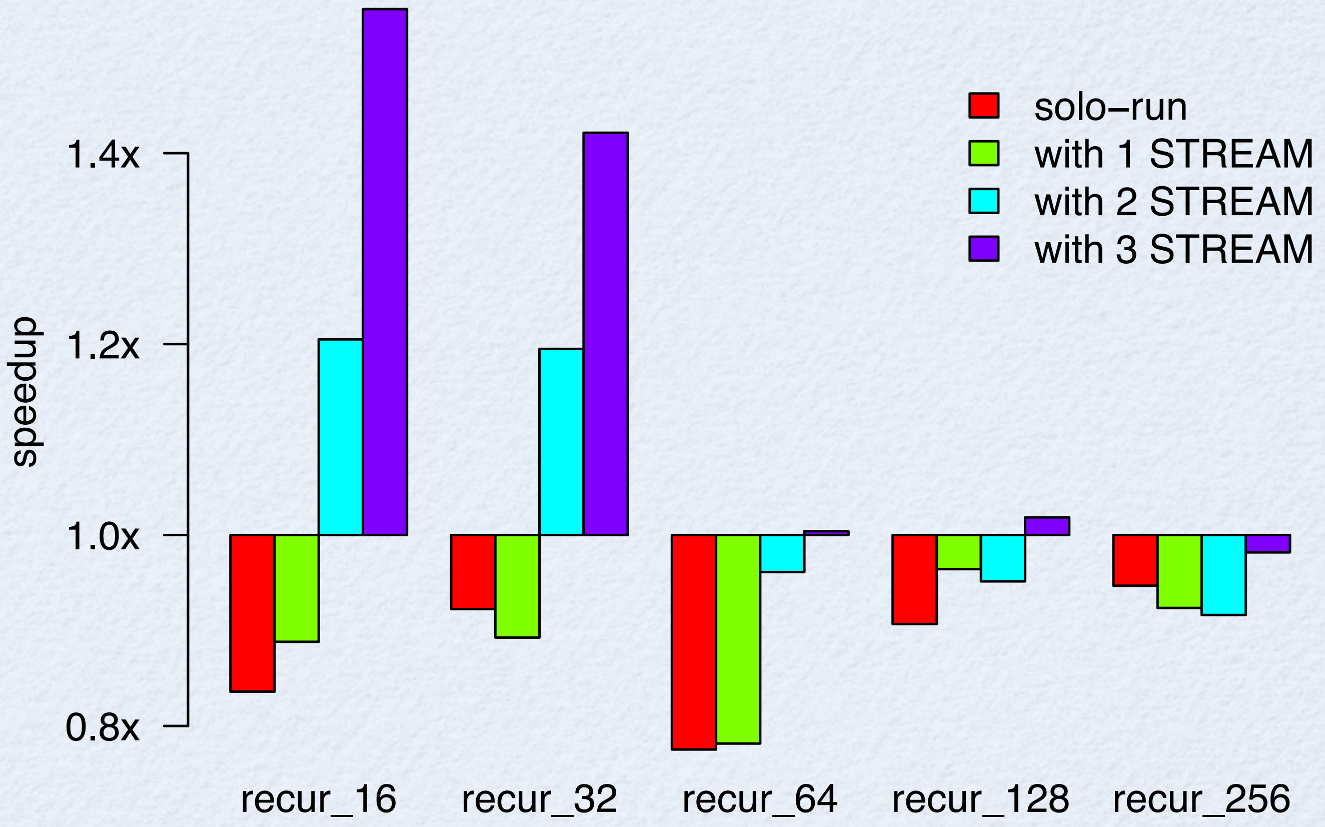
Real Machine Performance

- Defensiveness parameter $\gamma = 4$



Comparison with Cache Oblivious Algorithm

- Recursive version matrix multiplication [Qing et al. PLDI 2000]



Summary and Future Work

- Defensive tiling
  - Self-aware -> Peer-aware
  - Reduce interference
- Currently investigating
  - Co-run with other programs, add adaptivity
  - Use the shared cache model to direct compiler optimization