Scheduling of Wavefront Parallelism on Scalable Shared-memory Multiprocessors

Naraig Manjikian and Tarek S. Abdelrahman

Department of Electrical and Computer Engineering
University of Toronto, Ontario, Canada

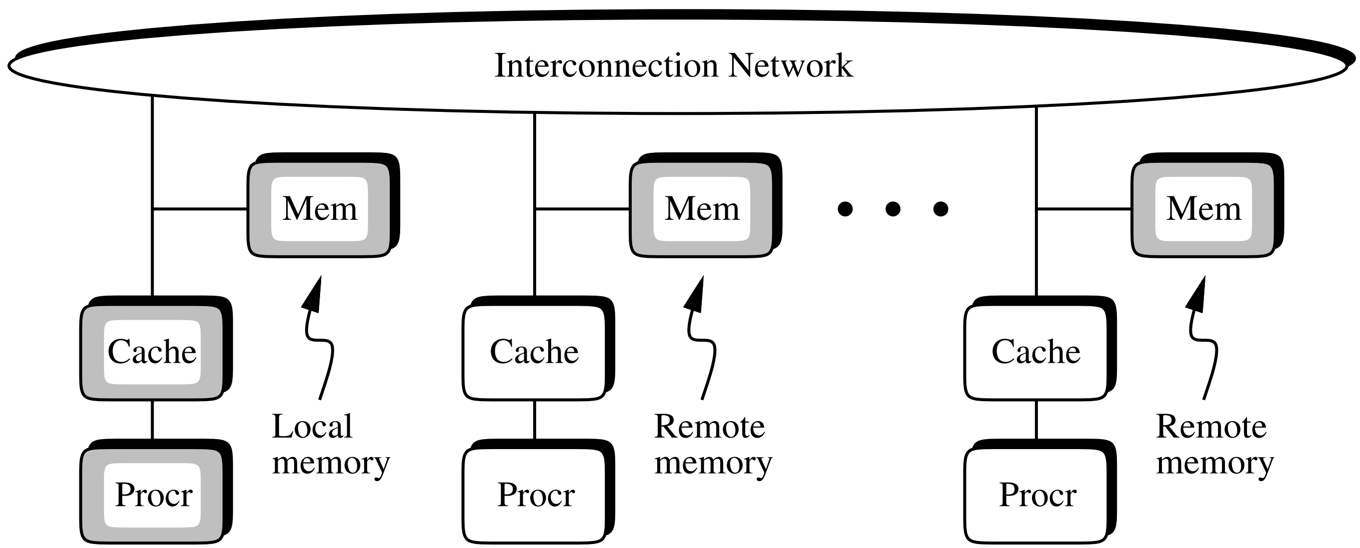
Summary

Tiling of loop nests enhances locality, but transformations to enable tiling lead to *wavefront parallelism* and *intertile data reuse*. Appropriate scheduling of tile execution is required on scalable shared-memory multiprocessors.

We evaluate three scheduling strategies to demonstrate:

- the impact of tile size on parallelism and locality
- the importance of exploiting intertile reuse
- the effect of remote memory access latency

Scalable Shared-memory Multiprocessors



- Production systems: Convex SPP, Cray T3E
- Research systems: Stanford DASH/FLASH, Univ. of Toronto NUMAchine

- Optimize performance by exploiting parallelism *and* enhancing locality

Outline

1. Tiling and wavefront parallelism
2. Data access patterns and reuse
3. Scheduling strategies
4. Experimental evaluation
5. Conclusions

Locality Enhancement with Tiling

- An example of an outer loop which *carries reuse*:

```
do t = 1,T
  do j = 1,N
    do i = 1,N
      a[i,j] = ...
```

- *Tiling* of inner loops to reduce time between uses:

```
do jj = 1,N,B
  do ii = 1,N,B
    do t = 1,T
      do j = jj,min(jj+B-1,N)
        do i = ii,min(ii+B-1,N)
          a[i,j] = ...
```

- Iterations blocked into units of *tiles* identified by (**`jj,ii`**)
- Each tile captures reuse carried by original outer loop

Loop Skewing to Enable Tiling

- SOR loop nest:

```
do t = 1,T
  do j = 2,N-1
    do i = 2,N-1
      a[i,j] = (a[i-1,j] + a[i+1,j] + a[i,j]
            + a[i,j+1] + a[i,j-1]) / 5
```

- Original loop dependences: {(1,0,0), (1,-1,0), (1,0,-1), (0,1,0), (0,0,1)}

- *Skewing* is used to enable tiling:

```
do t = 1,T
  do j = 2+t,N-1+t
    do i = 2+t,N-1+t
      a[i-t,j-t] = ...
```

- Dependences after skewing: {(1,1,1), (1,0,1), (1,1,0), (0,1,0), (0,0,1)}

Loop Skewing to Enable Tiling

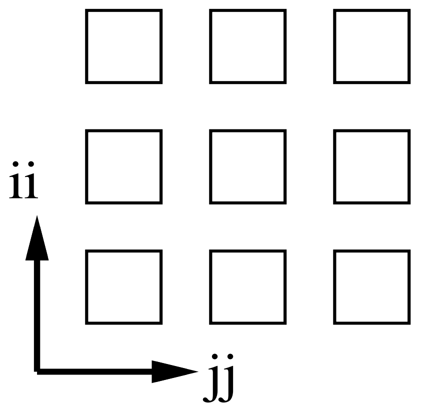
- Legal tiling after loop skewing:

```
do jj = 2,N-1+T,B
  do ii = 2,N-1+T,B
    do t = 1,T
      do j = max(jj,2+t),min(jj+B-1,N-1+t)
        do i = max(ii,2+t),min(ii+B-1,N-1+t)
          a[i-t,j-t] = (a[i-1-t,j-t] + ...
            ... + a[i-t,j-1-t]) / 5
```

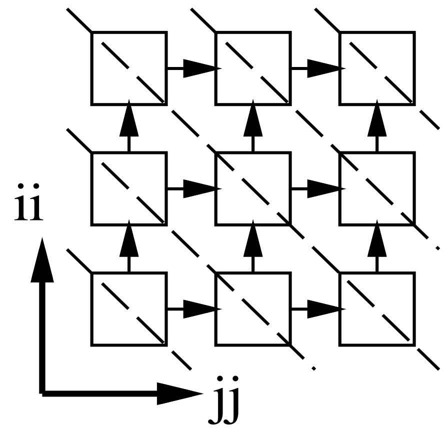
- Effects of loop skewing:
  - dependences in outer loops ⇒ *parallelism limited to wavefronts*
  - changes to data access patterns ⇒ *more reuse between tiles*

Wavefront Parallelism in Tiled Loop Nests

- Skewing introduces dependences in outer loops after tiling
- Dependences dictate degree of parallelism:



Tiles are independent

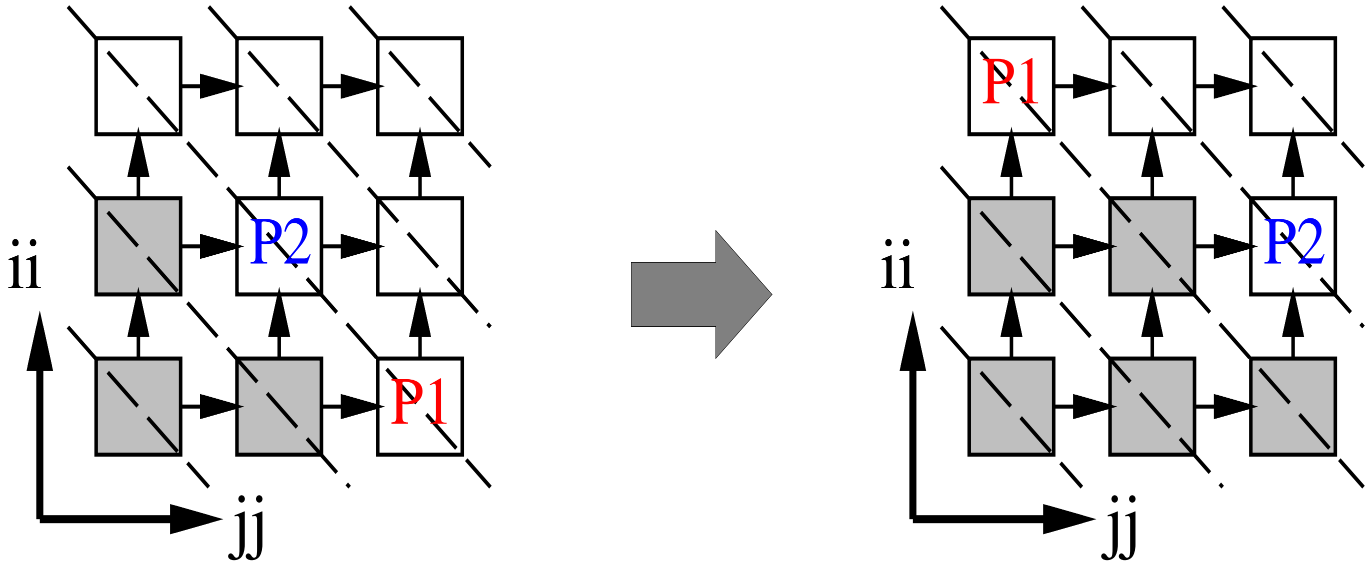


Parallelism limited to wavefronts

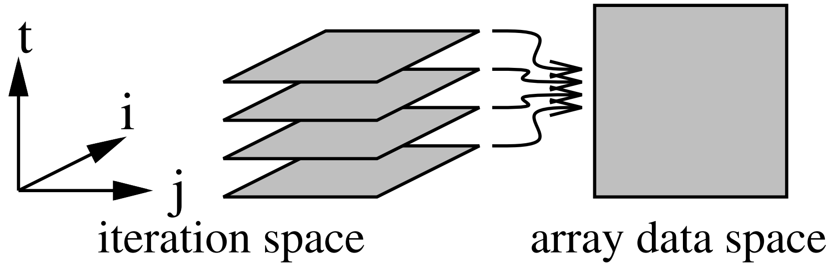
- Execution of tiles must be properly *scheduled*

DOACROSS Scheduling of Tile Execution

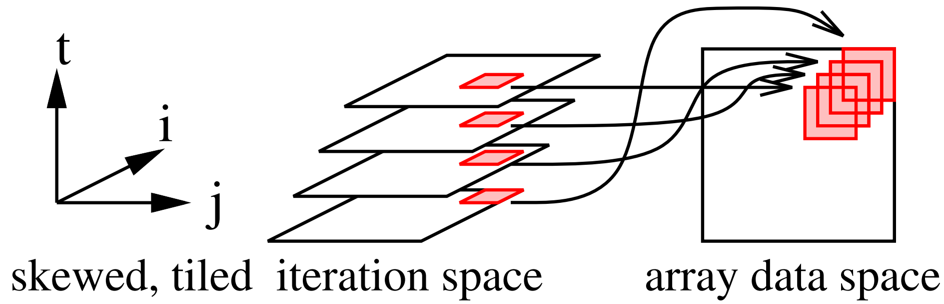
- *Local* synchronization between dependent tiles
- Execute tiles as soon as dependences are satisfied
- Concurrent execution of tiles in *different* wavefronts:



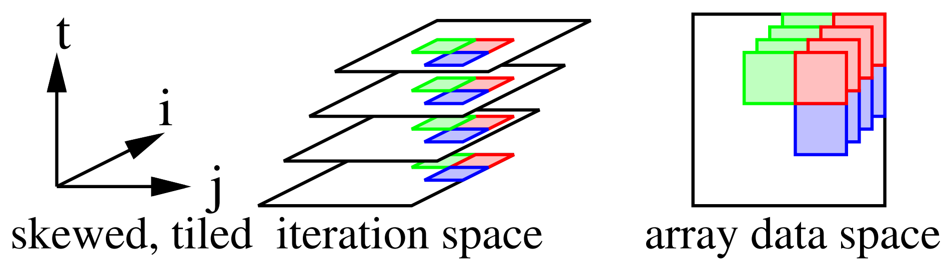
Data Access Patterns after Loop Skewing and Tiling



(a) Original SOR loop nest



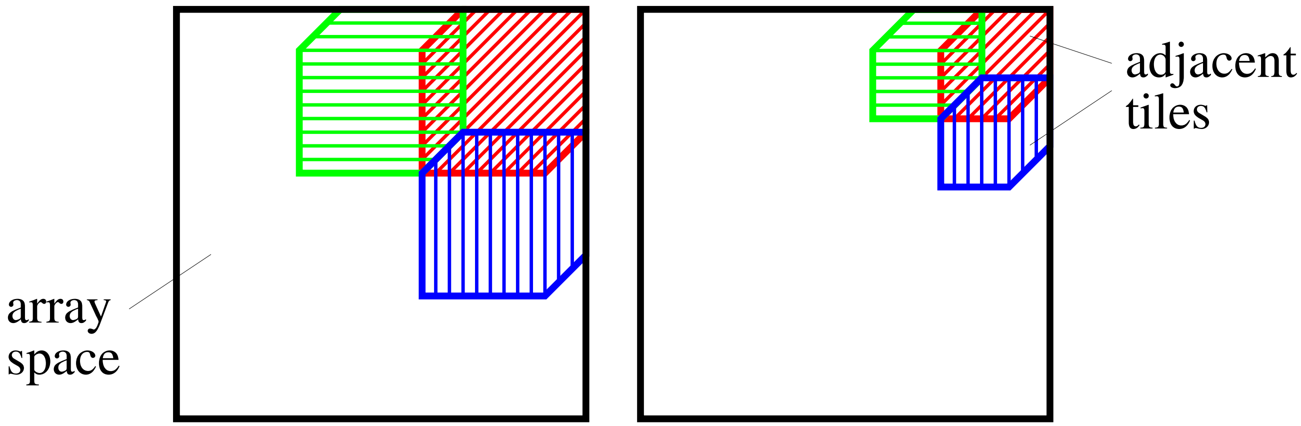
(b) Data access patterns for intratile reuse



(c) Intertile reuse between adjacent tiles

Intratile vs. Intertile Reuse

- Tiling captures original data reuse *within* tiles
- Skewing results in additional reuse *between* tiles
- Tile size determines relative significance of each type of reuse:



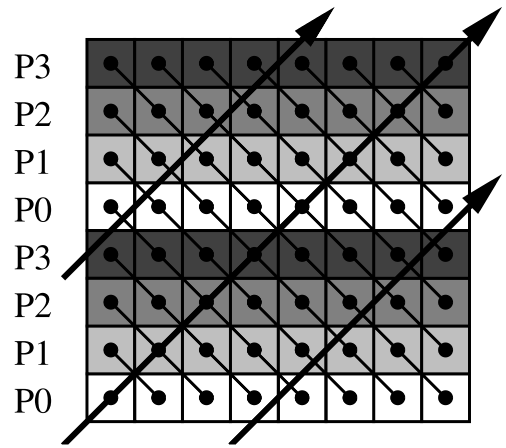
  - large tile size ⇒ *intra*tile reuse is more significant
  - small tile size ⇒ *inter*tile reuse is more significant

Tile Size, Parallelism, and Locality

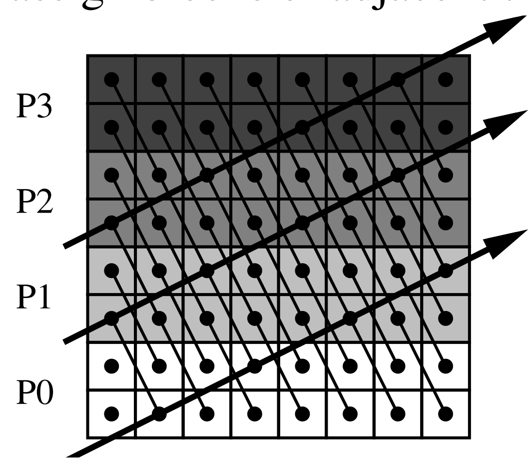
- Large number of processors $\Rightarrow$ greater parallelism $\Rightarrow$ small tiles
  - more independent tiles per wavefront
  - granularity of parallelism decreases
  - synchronization increases
- Small tiles $\Rightarrow$ more *inter*tile reuse
  - must execute adjacent tiles on *same* processor to exploit reuse
- Scheduling strategy should *enhance intertile locality* for small tiles

Scheduling Strategies for Wavefront Parallelism

- *Dynamic self-scheduling*: idle processors obtain tiles at run-time
- *Static cyclic scheduling*: assign rows of tiles cyclically to processors

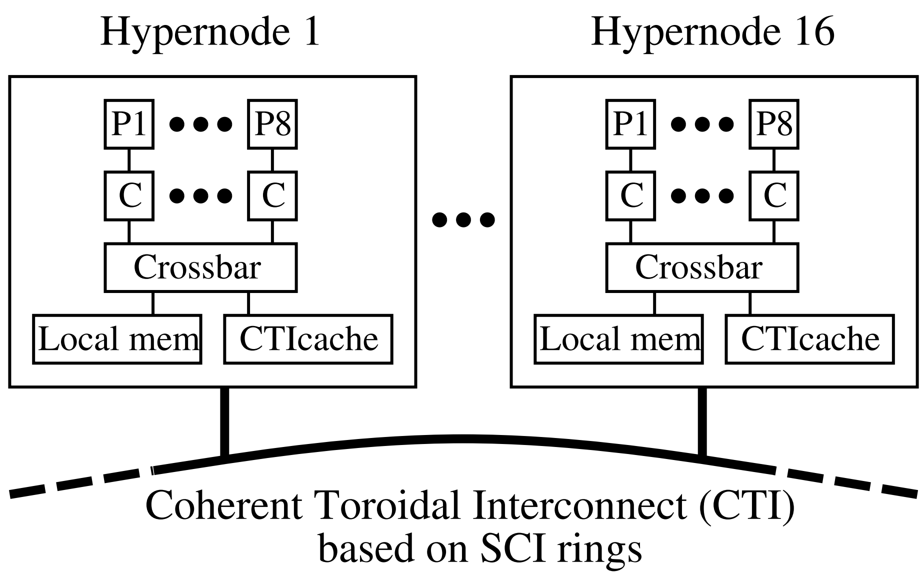


- *Static block scheduling*: assign blocks of adjacent tiles to processors



Experimental Evaluation

- Experiments conducted on Convex SPP1000:

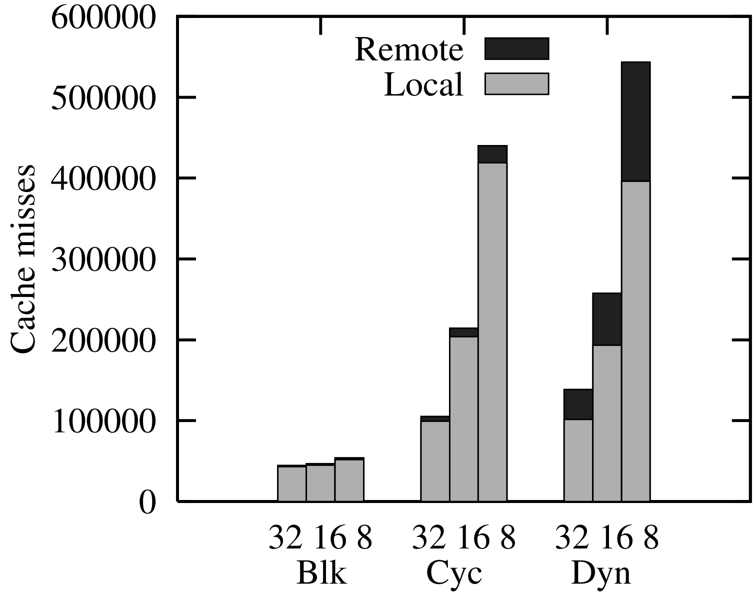


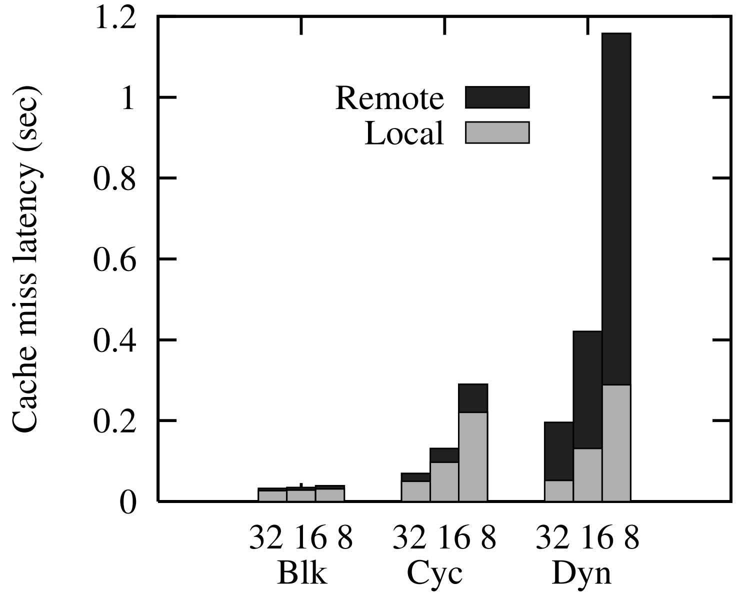
  - CTIcache contains remote data to reduce access latency
  - hardware monitoring measures cache misses, latency

- Compare scheduling strategies using 3 applications on up to 30 processors
  - study impact of varying tile size (32×32, 16×16, 8×8)
  - examine cache behavior and measure performance

Experimental Evaluation: SOR

- Cache misses and miss latencies on 16 processors (1024×1024, *T*=40):

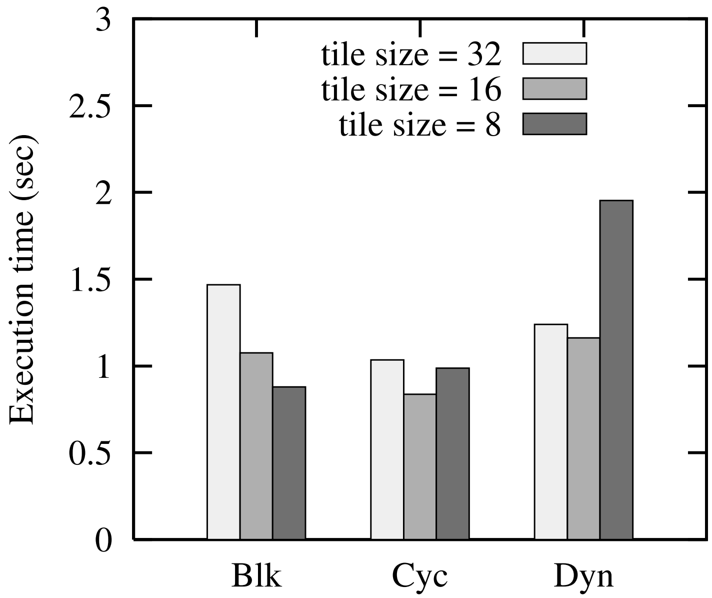


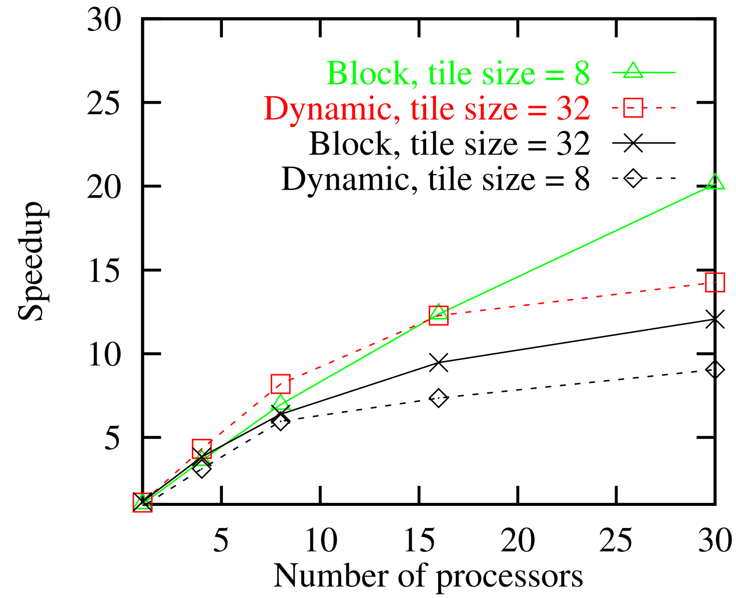


- Impact of remote misses for dynamic scheduling
- Least variation in misses with tile size for block scheduling
  - allows focus on increasing parallelism with small tile size

Experimental Evaluation: SOR

- Execution times on 30 processors and speedup:





- Consistent improvement with smaller tile size for block scheduling
- Importance of static scheduling for large number of processors

Experimental Evaluation: Jacobi

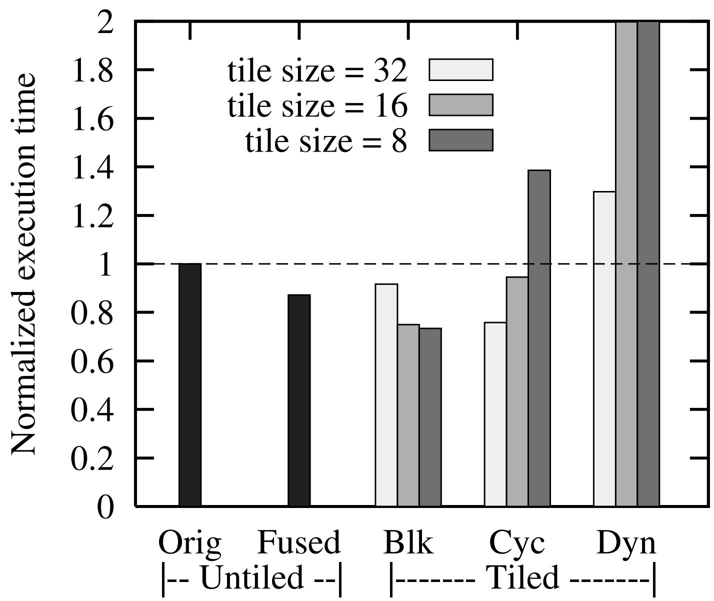
- The Jacobi loop nest sequence:

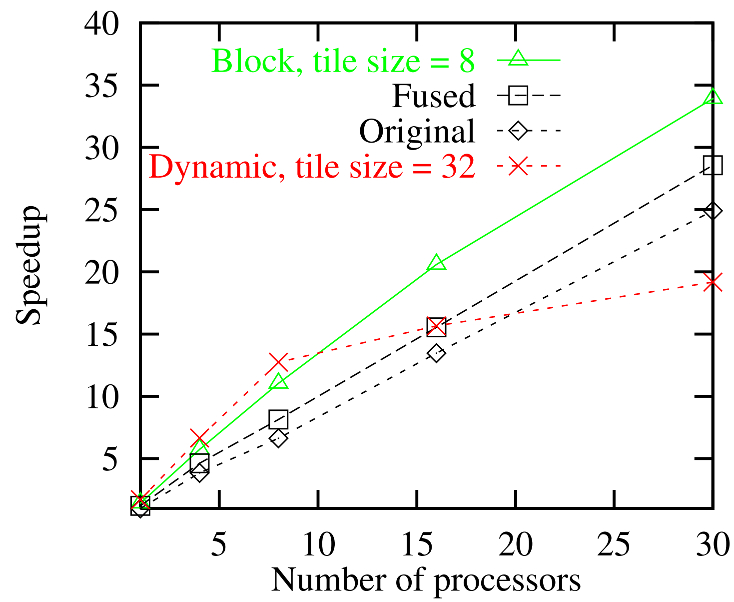
```
do t = 1,T
  doall j = 2,N-1
    doall i = 2,N-1
      b[i,j] = (a[i,j-1] + a[i,j+1]
              + a[i-1,j] + a[i+1,j]) / 4
  doall j = 2,N-1
    doall i = 2,N-1
      a[i,j] = b[i,j]
```

- *Fuse* inner loops using *shift-and-peel* (Manjikian/Abdelrahman,ICPP95)
- Skewing required to enable tiling
- Parallelism in final loop nest is limited to wavefronts
- Measure parallel performance of *original* code, *fused* code, and *tiled* code

Experimental Evaluation: Jacobi

- Cache misses and latencies similar to SOR
- Normalized times for 30 processors and speedup (2048×2048, T=10):

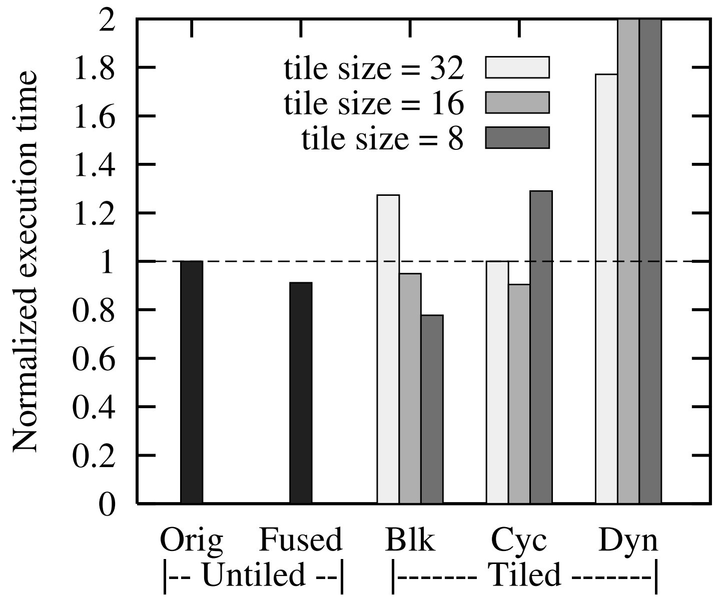


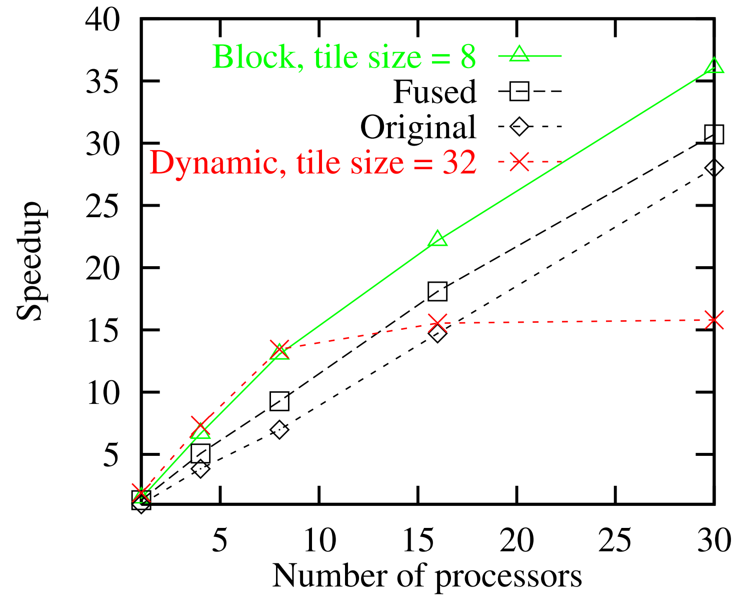


- Importance of exploiting *all* reuse with fusion and tiling

Experimental Evaluation: LL18

- LL18 requires fusion of 3 inner loops and skewing to enable tiling
- Cache misses and latencies similar to previous results
- Normalized times for 30 processors and speedup (1024×1024, *T*=10):





Conclusions

- Scheduling of tiled loop nests on scalable shared-memory multiprocessors
- Wavefront parallelism from skewing requires proper scheduling
  - exploiting intertile *vs.* intratile reuse
  - tile size, parallelism, and locality
- Static scheduling outperforms dynamic scheduling with remote misses
- Block scheduling gives least variation in cache misses with tile size
  - exploits all reuse within blocks of tiles
  - allows focus on maximizing parallelism with smaller tile size