Towards Automated Characterization of the Data Movement Complexity of Affine Programs

Venmugil Elango	Ohio State University
Louis-Noel Pouchet	Ohio State University
Fabrice Rastello	INRIA, Grenoble
J. (Ram) Ramanujam	Louisiana State University
Saday Sadayappan	Ohio State University

Computational vs. Data Movement Complexity

```
for (i=1; i<N-1; i++)
  for (j=1;j<N-1; j++)
    A[i][j] = A[i][j-1] + A[i-1][j];
```

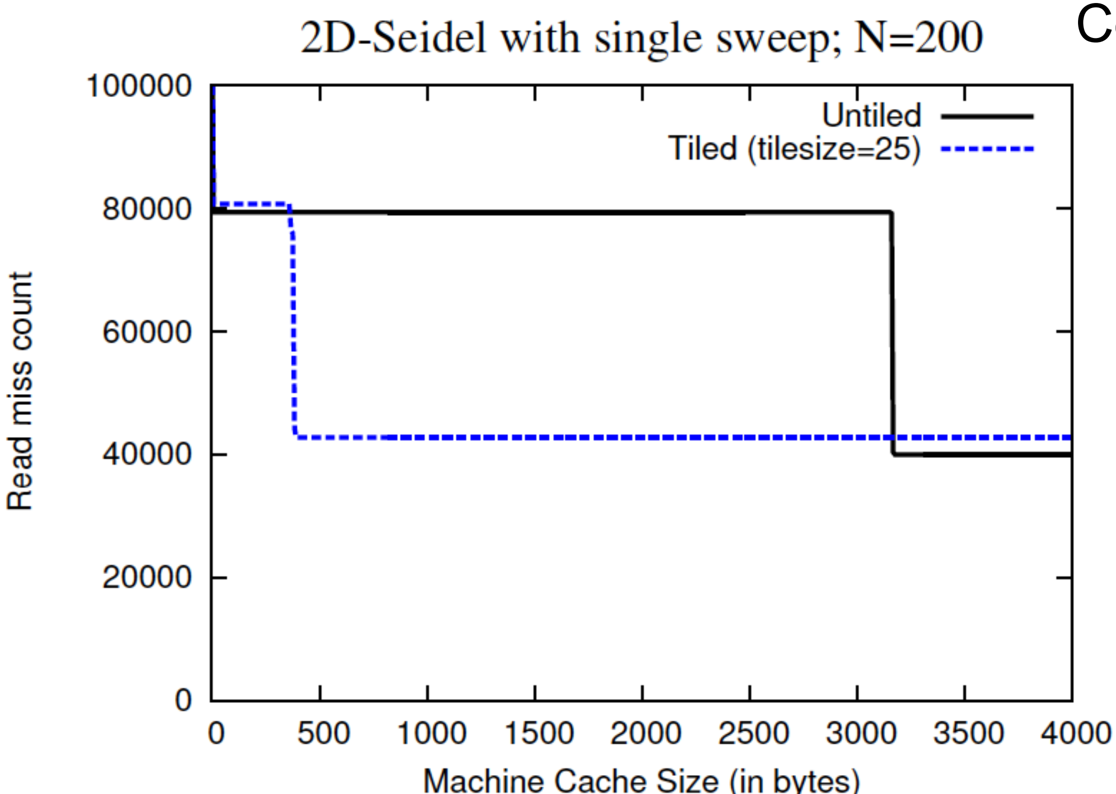
Untiled version

Comp. complexity: $(N-1)^2$ Ops

```
for(it = 1; it<N−1; it +=B)
  for(jt = 1; jt<N−1; jt +=B)
    for(i = it; i < min(it+B, N−1); i++)
      for(j = jt; j < min(jt+B, N−1); j++)
        A[i][j] = A[i−1][j] + A[i][j−1];
```

Tiled Version

Comp. complexity: $(N-1)^2$ Ops



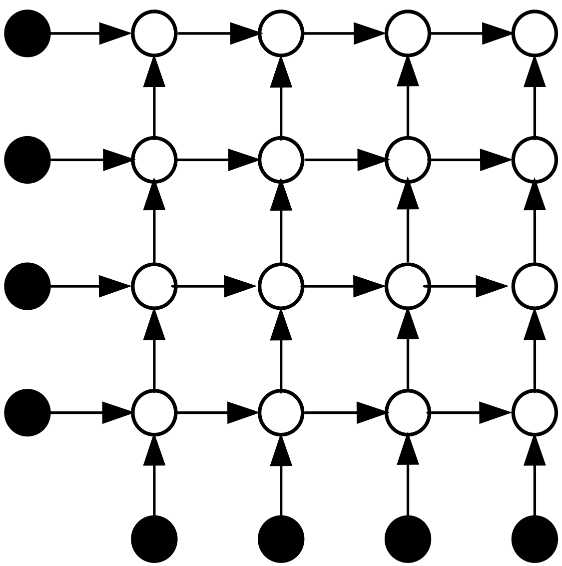
- Data movement cost different for two versions
- Also depends on cache size

Question: Can we achieve lower cache misses than this tiled version? How can we know when to stop, i.e. further improvement is not possible?

Question: What is the lowest achievable data movement cost among all possible equivalent versions of the computation?

Modeling Data Movement Complexity: CDAG

```
for (i=1; i<N-1; i++)
  for (j=1;j<N-1; j++)
    A[i][j] = A[i][j-1] + A[i-1][j];
```



CDAG for N=6

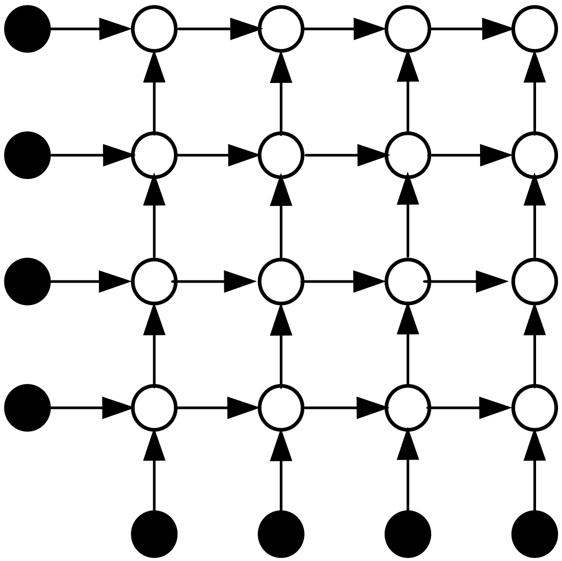
```
for(it = 1; it<N−1; it +=B)
  for(jt = 1; jt<N−1; jt +=B)
    for(i = it; i < min(it+B, N−1); i++)
      for(j = jt; j < min(jt+B, N−1); j++)
        A[i][j] = A[i−1][j] + A[i][j−1];
```

- CDAG abstraction:
  - Vertiex = operation, edges = data dep.
- 2-level memory hierarchy with S fast mem locs. & infinite slow mem. locs.
  - To compute a vertex, predecessor vertices must hold values in fast mem.
  - Limited fast memory => computed values may need to be temporarily stored in slow memory and reloaded
- Inherent data movement complexity of CDAG: Minimal #loads+#stores among all possible valid schedules

Modeling Data Movement Complexity: CDAG

```
for (i=1; i<N-1; i++)
  for (j=1;j<N-1; j++)
    A[i][j] = A[i][j-1] + A[i-1][j];
```

```
for(it = 1; it<N−1; it +=B)
  for(jt = 1; jt<N−1; jt +=B)
    for(i = it; i < min(it+B, N−1); i++)
      for(j = jt; j < min(jt+B, N−1); j++)
        A[i][j] = A[i−1][j] + A[i][j−1];
```



CDAG for N=6

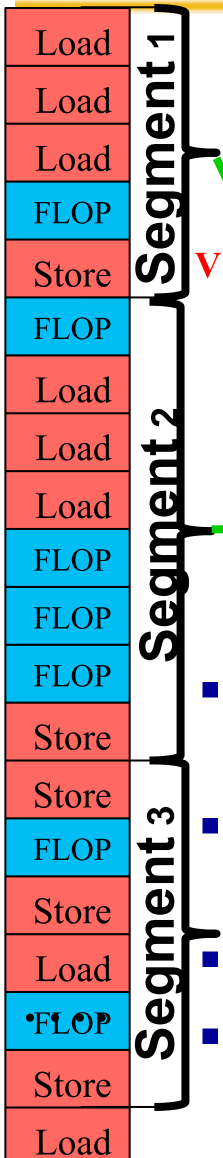
Develop upper bounds on min-cost

Minimum possible data movement cost?

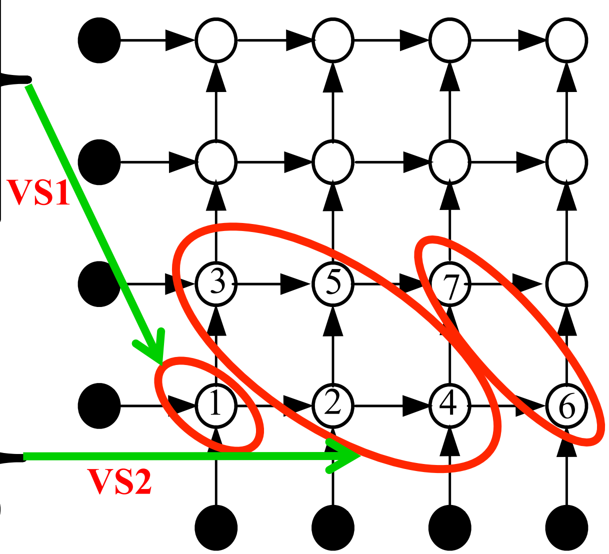
No known effective solution to problem

Develop lower bounds on min-cost

Prior Work on Lower Bounds Modeling

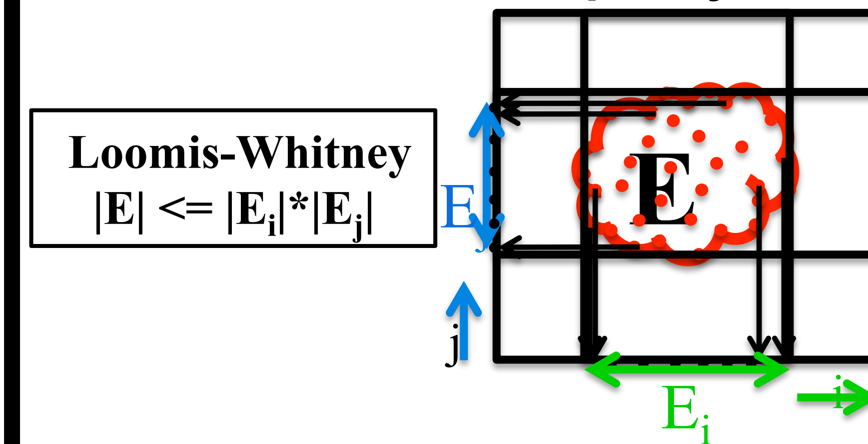


S-partition (Hong&Kung)



- Association between schedule and special kind of graph partition of CDAG
- Reason about valid 2S-partitions of graph instead of all valid schedules
- (+) Generality
- (-) Manual CDAG-specific reasoning => challenge to automate

Geometric Inequality



- Association between iteration space and data foot-print; use geometric inequality
- Christ et al. (2013): Automation, based on generalized geometric inequality (Holder-Brascamp-Lieb)
- (+) Automated bounds, e.g., $O(N^3/\mathrm{sqrt}(S))$ for NxN matrix-mult
- (-) Restricted computational model: 1) probs. multi-statement programs; 2) weakness of bound: ignore deps.

Our work: Static analysis using geometric reasoning to automate lower bounds for affine codes with CDAG model

Lower Bounds: Recent Developments

Theory & Models

1) Alternate lower bounds approach (graph min-cut based)
2) Modeling vertical + horizontal data movement bounds for scalable parallel systems **[SPAA '14]**

Tools

1) Automated lower bounds for arbitrary explicit CDAGs
2) **Automated parametric lower bounds for affine programs**

[HiPEAC poster; POPL '15]

Applications

1) Comparative analysis of algorithms via lower bounds
2) Assessment of compiler effectiveness
3) **Algorithm/architecture co-design space exploration [HiPEAC Paper, Session 12]**