HiPEAC'15 1/19/2015

Towards Automated Characterization of the Data Movement Complexity of Affine Programs

V. Elango†, F. Rastello§, L-N Pouchet†, J. Ramanujam‡, P. Sadayappan†
†The Ohio State University §INRIA ‡Louisiana State University

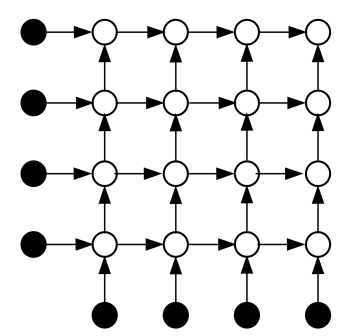
Modeling Data Movement Complexity

```
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];
```

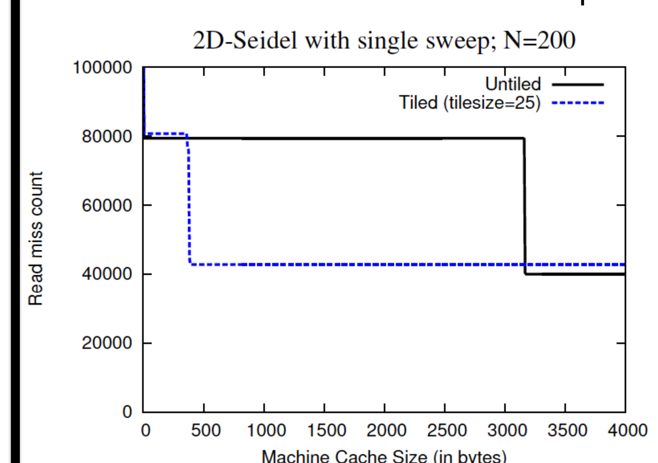
Untiled version
Comp. complexity: $(N-1)^2$ Ops

```
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); i++)
        A[i][j] = A[i−1][j] + A[i][j−1];
```

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


CDAG for N=6

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



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

Question: Can we do better? How do we know when no further improvement possible?

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

Develop upper bounds on min-cost

Minimum possible data movement cost? No known effective solution to problem

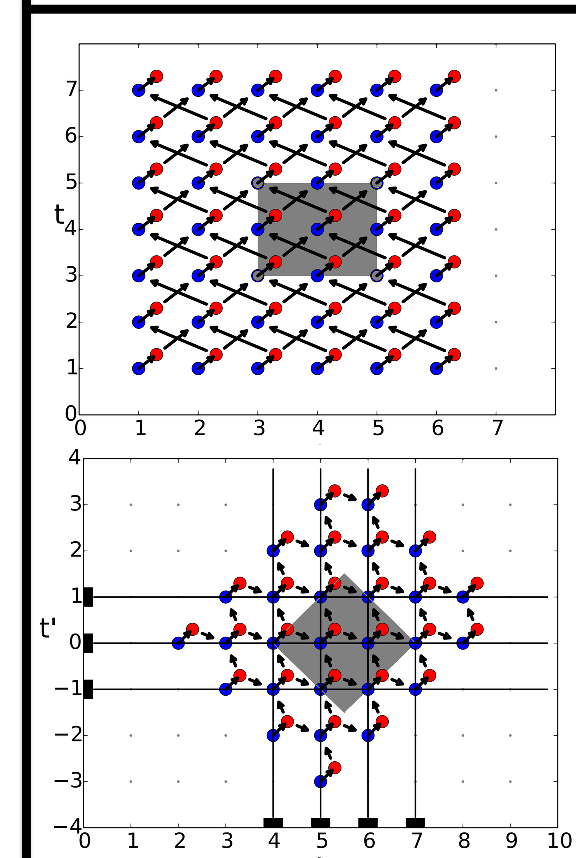
Develop lower bounds on min-cost

Prior Work: Data Movement Lower Bounds

- Arbitrary CDAGs:
  - Hong & Kung (1981): strong relation between:1) Data movement cost for a CDAG schedule, and 2) Number of vertex-sets in "2S-partition" of CDAG
  - Change from reasoning about all valid schedules to all valid 2S-partitions of graph
  - (+) Generality
  - (-) Manual CDAG-specific reasoning => challenge to automate
- Linear-Algebra-like algorithms:
  - Irony et al. (2004) and Ballard et al. (2011): Geometric approach based on geometric inequality
  - Christ et al. (2013): Automation, based on generalized geometric HBL inequality (Holder-Brascamp-Lieb)
  - (+) Automated asymptotic parametric lower bound expressions, e.g., $O(N^3/\mathrm{sqrt}(S))$ for NxN mat-mult
  - (-) Restricted computational model: weakness of bounds or inapplicability

Our work: Static analysis to automate asymptotic parametric lower bounds analysis of affine codes for CDAG model

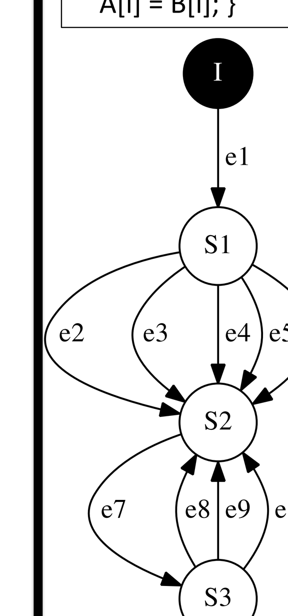
Lower Bounds for CDAGs: Geometric Reasoning



```
for (t=1; t<T; t++) {
  for (i=1; i<N-1; i++)
    B[i] = A[i-1]+A[i]+A[i+1];
  for (i=1; i<N-1; i++)
    A[i] = B[i];
}
```

- How to upper-bound $|VS_i|$ for any valid 2S-partition?
- Key Idea: Use geometric inequality, but relate iteration points to $In(VS_i)$ and not data footprint
  - Find rays corresponding to dependence chains => projections
  - Projected points from VS_i must be subset of $In(VS_i)$
  - Transform iteration space so that rays are along coordinate axes

```
Parameters: N, T
Inputs: In[N]; Outputs: A[N]
for (i=0; i<N; i++)
  A[i] = In[i];                      S1
for (t=0;t<T;t++) {
  for(i=1;i<N-1;i++)
    B[i] = A[i-1]+A[i]+A[i+1];       S2
  for(i=1;i<N-1;i++)
    A[i] = B[i]; }                   S3
```



- Use ISL to find all "must" data flow dependences
- Cycles data dep. graph == "rays" in the CDAG
- Generalized geom. inequality allows different dimensional orthogonal projections
  - Parametric exponents in inequality: sum of weighted ranks of projected subspaces must exceed rank of full iteration space
  - solve a linear program to find optimal weights
  - => asymptotic parametric I/O lower bound for affine program

$|E| \le \prod_{j=1}^{d} |\phi_j(E)|^{1/(d-1)} \rightsquigarrow |E| \le \prod_{j=1}^{m} |\phi_j(E)|^{s_j}$ s.t., $\forall i,\ 1 \le \sum_{j=1}^{m} s_j \delta_{i,j}$

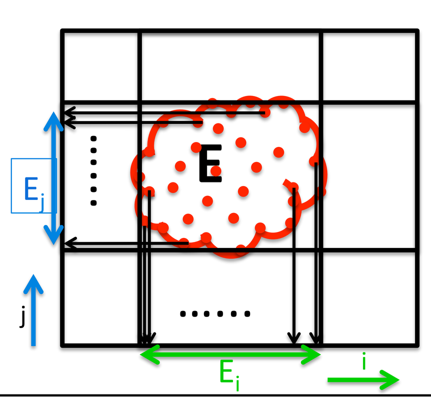
where,
$(s_1,\dots,s_m) \in [0,1]^m$
$\phi_j : \mathbb{R}^d \to \mathbb{R}^{d_j}$ are orthogonal projections
$\delta_{i,j} : \dim(\phi_j(\mathrm{span}(\vec{e_i})))$ – where $\vec{e_i}$, i-th canonical vector.

Lower Bounds: Geometric Reasoning with Data Footprints

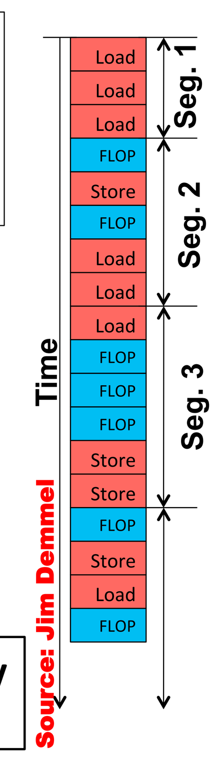
- Loomis-Whitney inequality (2D): bounds #points in a set by product of # projected points on coordinate axes
- Prior work: Uses Loomis-Whitney inequality & generalization (Holder-Brascamp-Lieb) for lower bounds for linear-algebra-like computations
  - Projections of iteration-space points <==> Data footprint
  - Geometric inequality: Bound max. #of ops for a given # of data moves

$$|E| \le \prod_{j=1}^{d} |\phi_j(E)|^{1/(d-1)}$$

```
for (i=0; i<N; i++)
  for (j=0;j<N;j++)
    if (i <> j) force[i] +=
      func(pos[i],pos[j])
```



2D Loomis-Whitney Inequality
$|E| \le |E_i| * |E_j|$



- Divide execution trace into segments with S load/stores (3 in ex.)
- Within each segment, #distinct elements of pos[] <= 2S (up to S coming into segment in scratchpad and another S explicitly loaded)
- For code example, projection of Stmt(i,j) onto i-axis maps to data element pos[i]; similarly for j-axis Max. # distinct elts of pos[i] or pos[j] read in any segment <= 2S
- By Loomis-Whitney, max. # iteration points in any segment, |P| <= 2S*2S
- Min. #segments >= $N^2/4S^2$; each seg. (but last) has S load/stores
- #load/stores >= $(N^2/4S^2-1)*S = \Omega(N^2/S)$

Geometric Reasoning with Data Footprints: Limitations

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

```
for (i=0; i<N; i++)
  for (j=0;j<N;j++)
    for (k=0;k<N;k++) {
      C[i][j] += 1;
      A[i][k] += 1;
      B[k][j] += 1;
    }
```

Loop Distribution

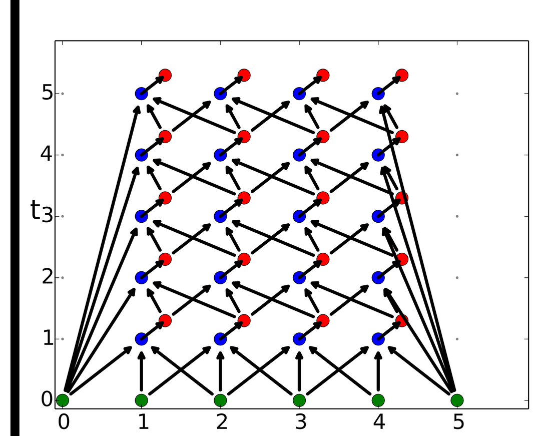
```
for (i,j,k) C[i][j] += 1;
for (i,j,k) A[i][k] += 1;
for (i,j,k) B[k][j] += 1;
```

Same access functions => same analysis result LB = $\Omega(N^3/\sqrt{S})$

Semantically equivalent code after loop distribution: but different IO lower bound LB = $\Omega(N^2)$

- Cannot handle multi-statement programs
  - Computations with very different data mvmt. Rqmts. but same array access footprint => same LB
  - Semantics preserving loop transformations can result in change to lower bound

```
for (t=1; t<T; t++) {
  for (i=1; i<N-1; i++)
    B[i] = A[i-1]+A[i]+A[i+1];
  for (i=1; i<N-1; i++)
    A[i] = B[i];
}
```

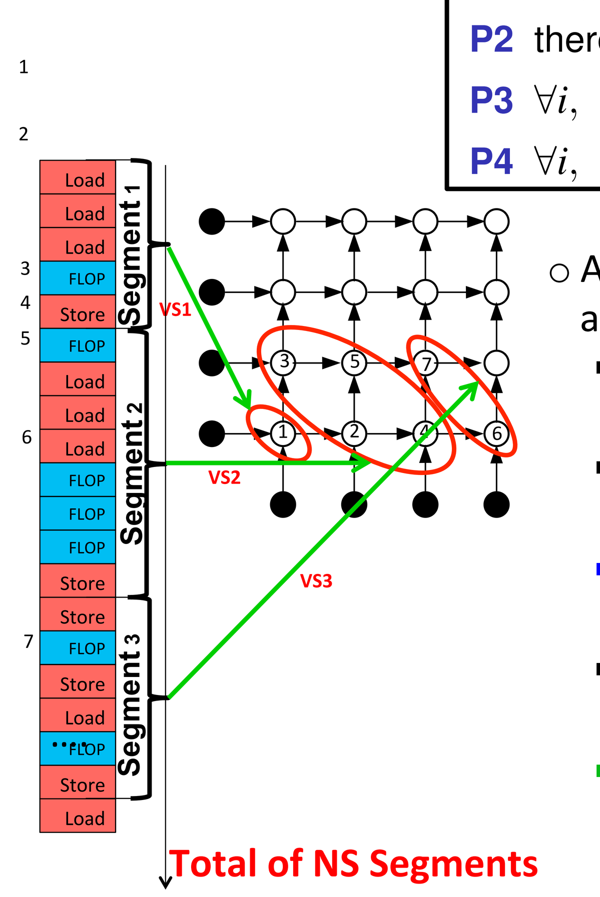


- Cannot model effect of data dependences
  - Dependences may impose constraints => higher data movement cost than footprint analysis reveals
  - Example: 1D Jacobi – footprint based geometric analysis cannot derive known LB of $\Omega(NT/S)$

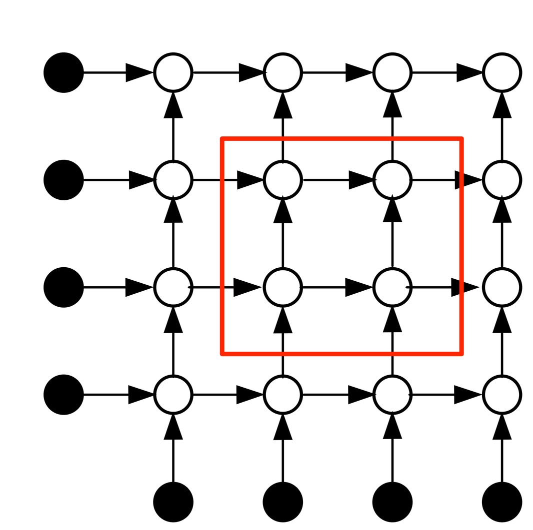
CDAG Lower Bounds: Hong/Kung S-Partitioning

P1 $\forall i \neq j,\ V_i \cap V_j = \emptyset$, and $\bigcup_{i=1}^{h} V_i = V \setminus I$
P2 there is no cyclic dependence between subsets
P3 $\forall i,\quad |\mathrm{In}(V_i)| \le S$
P4 $\forall i,\quad |\mathrm{Out}(V_i)| \le S$

S-Partition of CDAG satisfies 4 properties



- Any valid schedule using S registers is associated with a 2S-partition of CDAG
  - Divide trace into segments incurring exactly S load/stores
  - Ops executed in segment-i form a convex vertex set VS_i
  - $|In(VS_i)| \le 2S$ (up to S from prev. segment and up to S new loads)
  - Each segment (except last) has S loads/ stores => $S*NS \ge$ Total I/O $\ge S*(NS-1)$
  - Reasoning about minimum #vertex sets for any valid 2S-partition => Lower bound on # loads/stores



- Analyze CDAG structure
  - Establish Max $|VS_i| \le VS_{max}(S)$
  - => Min. # vertex sets = $NS_{min}(S) = N_{vertices}/VS_{max}(S)$
  - => $IO_{min}(S) \ge (NS_{min}-1)*S$
  - In example: S=2, $N_{vertices}$=16
  - $VS_{max}(S) = 4$ => NS_{min}= 16/4 = 4
  - $IO_{min} \ge 2*(4-1) = 6$

Lower Bounds: Research Directions

Theory & Models
1) Alternate lower bounds approach (graph min-cut based)
2) Composition of lower bounds
3) 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
[this poster; POPL '15]

Applications
1) Comparative analysis of algorithms via lower bounds
2) Assessment of compiler effectiveness
3) Algorithm/architecture co-design space exploration
[ACM TACO '14, Hipeac '15]