UNIVERSITY of DELAWARE

Locality Aware Concurrent Start for Stencil Applications

Sunil Shrestha[1], Joseph Manzano[2], Andres Marquez[2], John Feo[2] and Guang R. Gao[1]

[1]University of Delaware

[2]Pacific Northwest National Laboratory

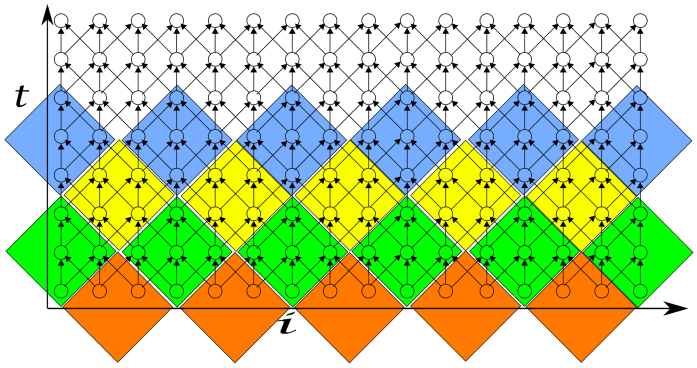
Pacific Northwest
NATIONAL LABORATORY

Introduction

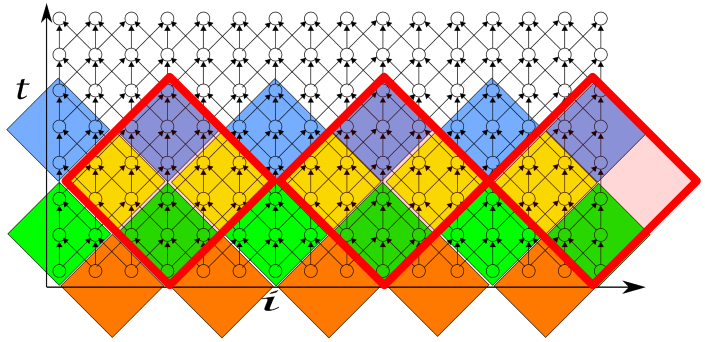
- Old/New Story:
  - Abundant processing/Limited shared resources
  - Memory access latency a bottleneck
  - Tiling: Coarse grain Vs Fine Grain

- Stencils
  - Compute intensive,
  - Symmetric dependence / concurrent start
  - Diamond tiling an efficient solution
  - Exploiting inner parallelism difficult when hierarchically tiled.

```
for (t=0; t<T;t++)
  for (i=1; i<N;i++)
    A[t+1][i]=α*(A[t][i+1]-
        β*A[t][i] + A[t][i-1])
```



One-Level Tiling

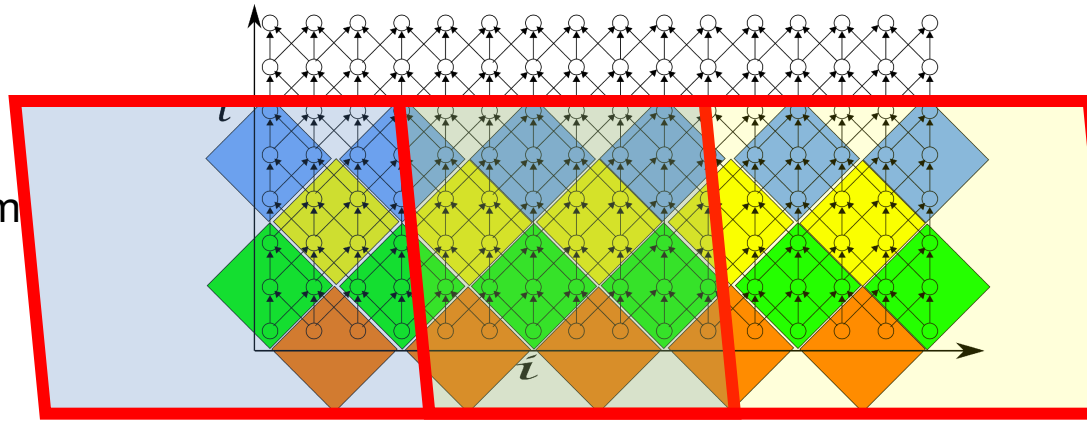


Hierarchical-Level

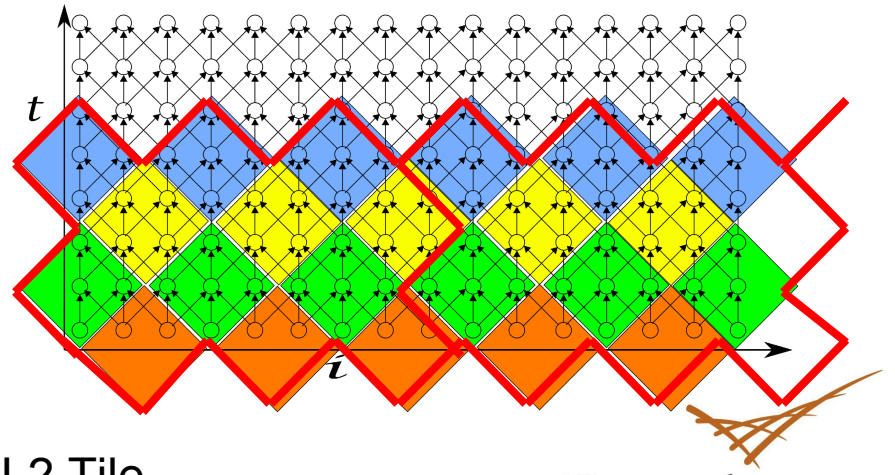
Heat-1d

Introduction

- Motivation
  - Can we do better than Diamond tiling?
  - Can we improve inner parallelism without compromising locality
  - Can we improve threads collaboration?

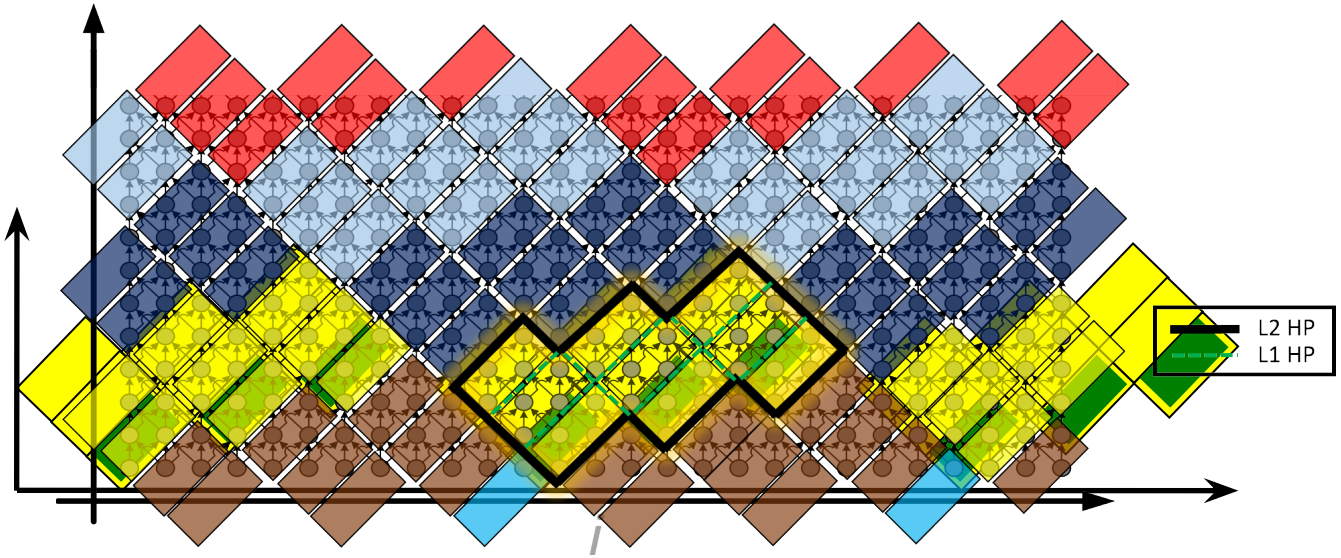



```
for (t=0; t<T;t++)
  for (i=1; i<N;i++)
    A[t+1][i]=α*(A[t][i+1]-
        β*A[t][i] + A[t][i-1])
```



L2 Tile

Contributions

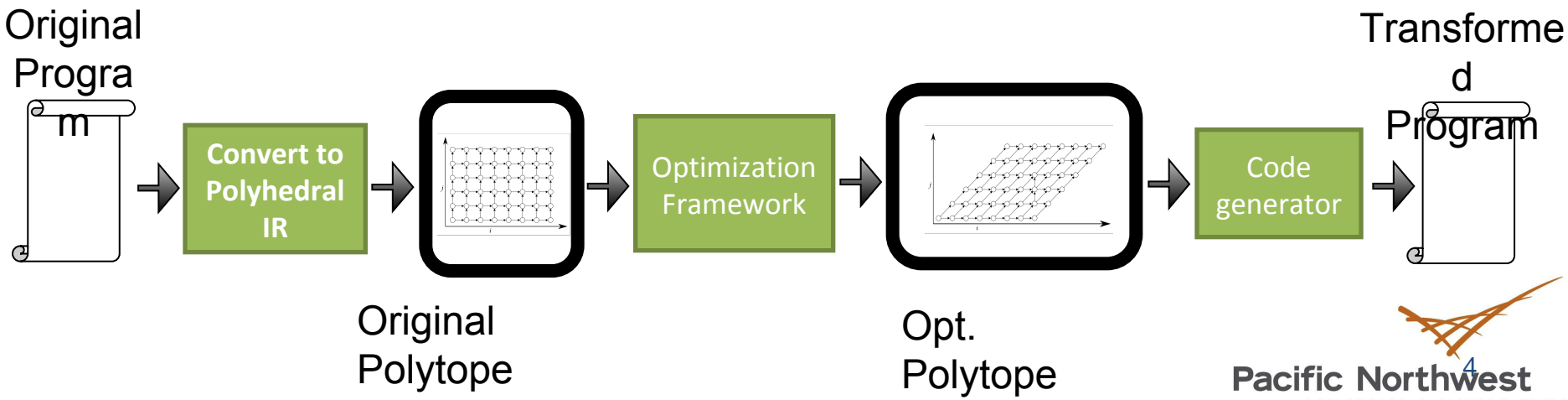


Exploiting locality of outer tiles without compromising inner parallelism:

1. Highly parallel tiling technique that exploits concurrent start at multiple levels.
2. Detailed analysis of such technique at different levels of the memory hierarchy.

Background

- The Polyhedral Model
  - Optimization framework for loop nests
  - Each loop iteration ➔ lattice points inside polytopes
- Allows:
  - Arbitrary composition of transformations
  - Dependency analysis on a class of programs
- Disadvantages
  - Very complex and expensive algorithms

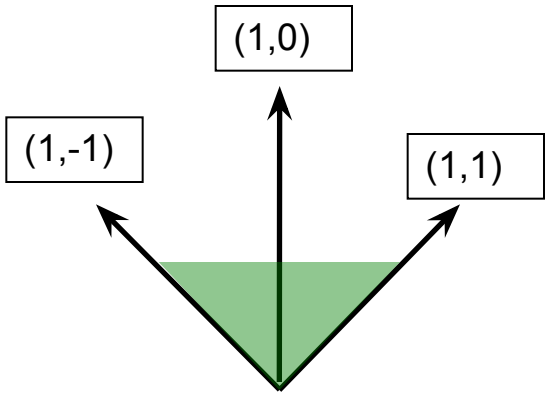


Background: Concurrent Start

- Conic Combination:
  - Given the set of vectors $x_1, x_2, x_3 \ldots x_n$,
  - A *conical combination* is a vector of the form

$$\lambda_1 x_1 + \lambda_2 x_2 \lambda_3 x_3 + \ldots + \lambda_k x_k$$

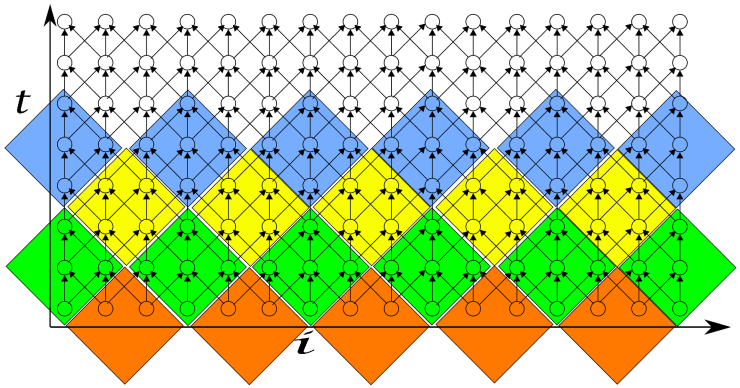
where $1 <= i <= k$ *and* $\lambda_i >= 0$



- Condition for Concurrent start:
  - Using hyperplanes representing a face and conical combination

$$\Phi = \lambda_1 \Phi^1 + \lambda_2 \Phi^2 + \ldots + \lambda_2 \Phi^k$$

where $\lambda_i >= 0$



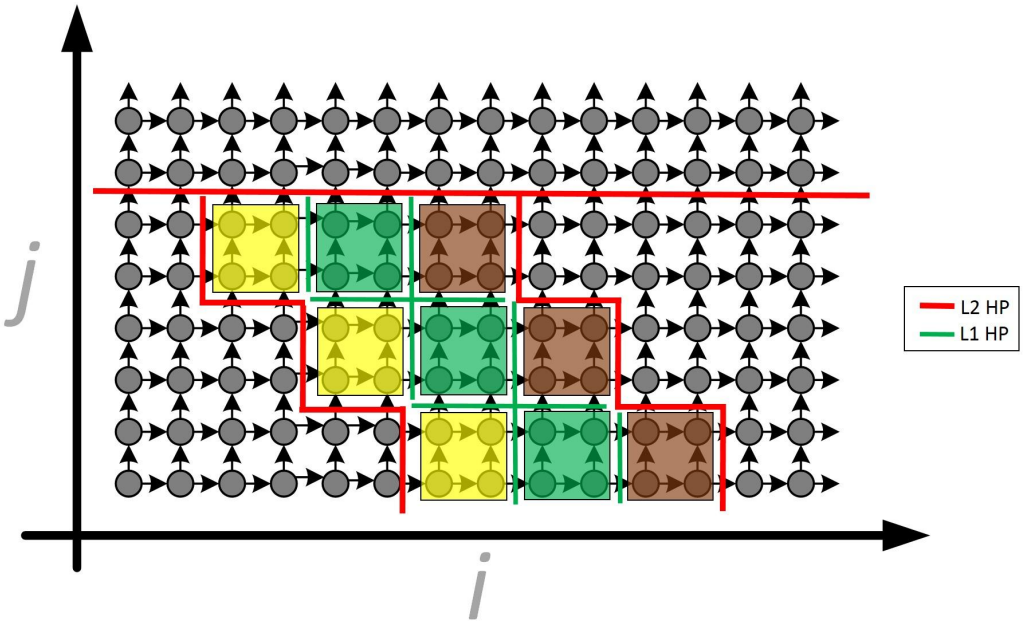
- Partial Concurrent Start:

$$\Phi = \lambda_1 \Phi^1 + \lambda_2 \Phi^2$$

Background: Jagged Tiling for Pipeline Start

```
for (i=1; i<n;i++) {
   for (j=1; j<n;j++){
      A[i][j]=A[i-1][j]+A[i][j-1];
   }
}
```

HP: Hyperplane
(n-1) dimensional affine subspace in n dimensional space



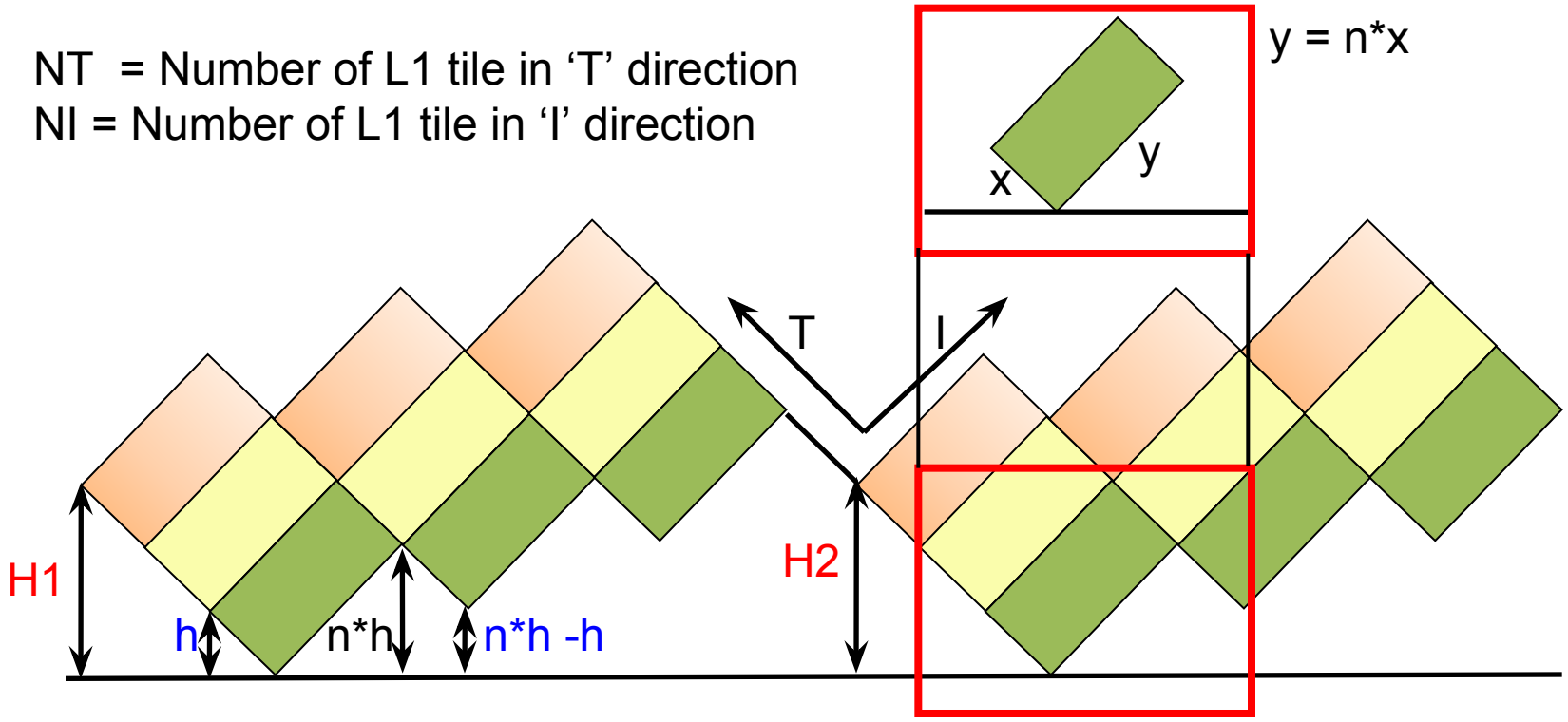
Two level hierarchical Tiling with
L1 HP : (1,0) and (0,1)
L2 HP : (1,1) and (0,1)

Background: Tools

- PLUTO
  - Polyhedral analysis tool
  - Takes C code as input
  - Uses CLOOG as for code generation
  - Produces communication minimal OpenMP code
- CLOOG
  - Code generation tools
  - Takes domain and scattering function (scheduling) as input
  - Oblivious to dependencies
- Our Framework uses:
  - Uses L1 hyperplanes generated by PLUTO
  - Finds hyperplanes needed for jagged tiling
  - Modifies CLOOG file and generates code

Jagged Polygon Tiling

NT = Number of L1 tile in 'T' direction
Nl = Number of L1 tile in 'l' direction



H1 = NT * h
H2 = Nl * (n*h –h)

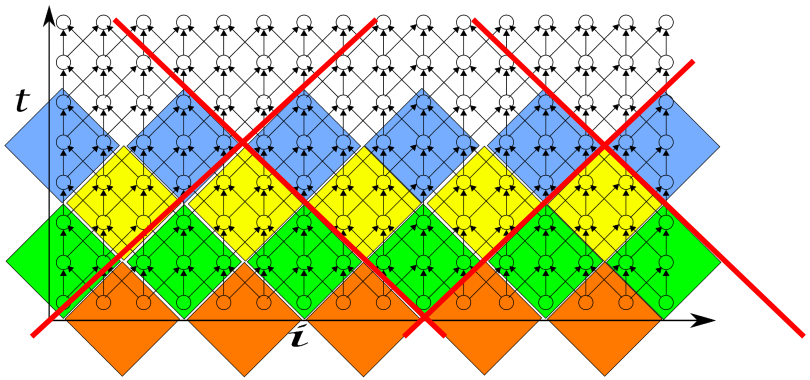
When,
H1 = H2
NT * h = Nl *h (n-1)

Example:

When n = 2,
NT = Nl
Number of L1 tiles required to form L2 are equal (3x3, 4x4 …)

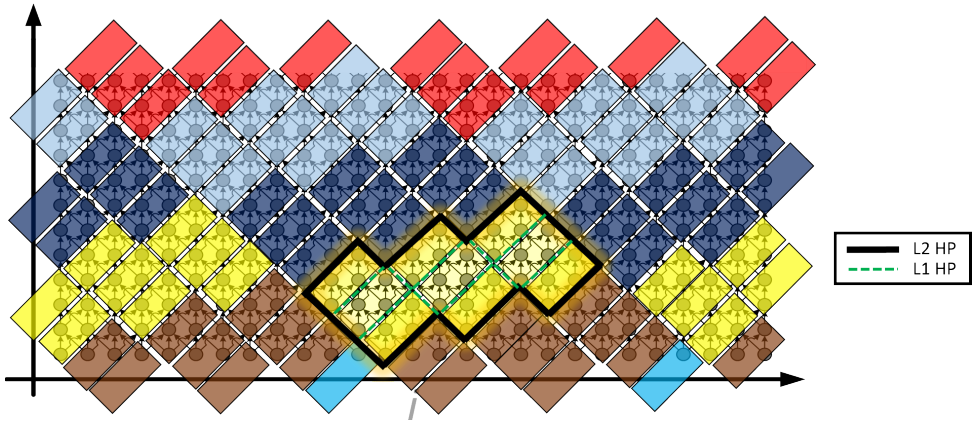
Jagged Polygon Tiling

Traditional Hierarchical Tiling



Two level hierarchical Tiling with
L1 HP : (1,1) and (1,-1)
L2 HP : (1,0) and (0,1)

Jagged Hierarchical Tiling



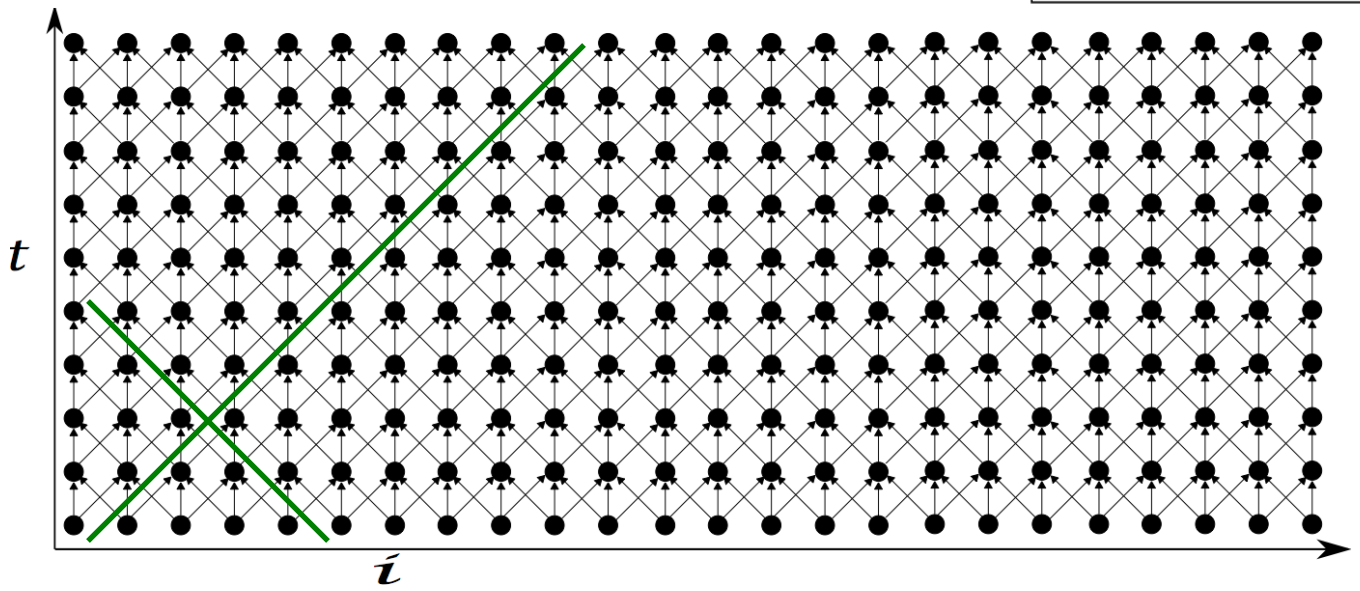
Two level hierarchical Tiling with
L1 HP : (1,1) and (1,-1)
L2 HP : (1,1) and (0,1)

Algorithm

1. Given:
   - Tiling hyperplanes: $\Phi^1, \Phi^2 \ldots \Phi^m$
   - L1 tile sizes : $tL1^1, tL1^2 \ldots tL1^m$
   - L2 tile sizes : $tL2^1, tL2^2 \ldots tL2^m$
   - Original Domain : D_s

Example:

L1 Tile: (1,1) and (1,-1)
L2 Tile: (1,1) and (0,1)
L1 Tile Size: 2x4
L2 Tile Size: 3*2x3*4
Original Domain: $0 <= t <= T-1$
$1 <= i <= N-1$



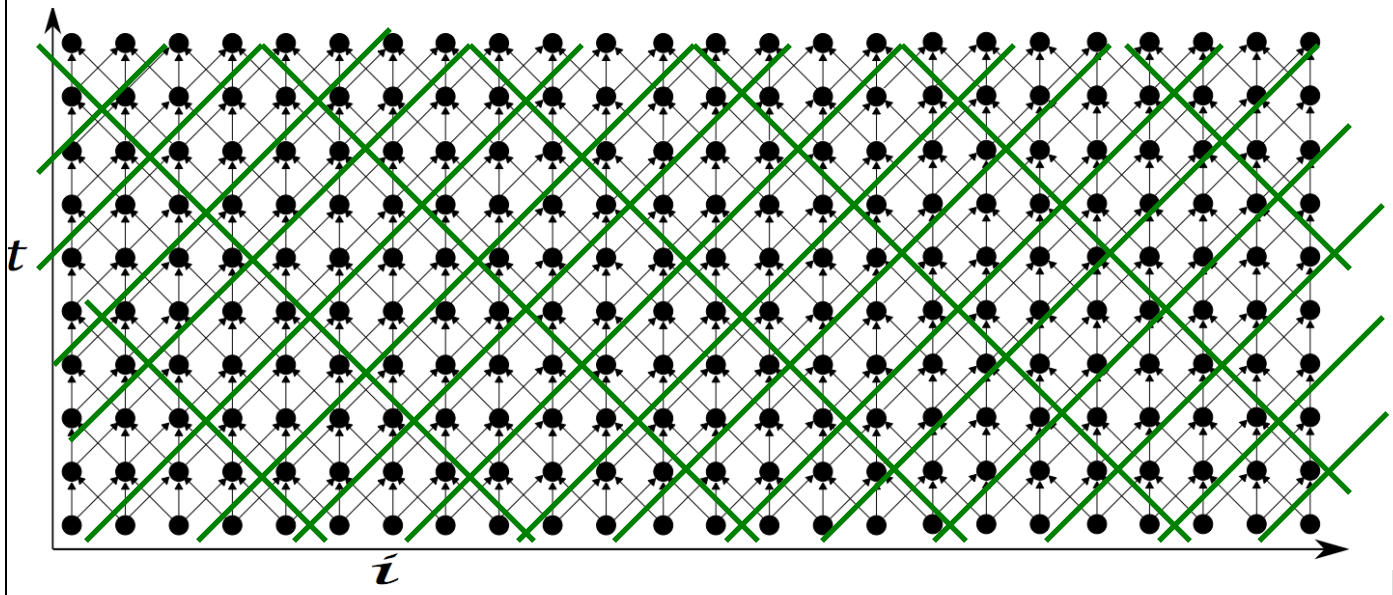
Algorithm

2. Tile for L1 using PLUTO:

Example:
Tiled Domain: $2T_{L1} <= t <= 2T_{L1} +1$

$4I_{L1} <= i <= 4I_{L1} +3$

3. Update Domain Constraint go get $\varphi^1_{L1s} \rightarrow \varphi^1_{L1s} + \varphi^2_{L1s}$ such than $\varphi^1_{L1s(t)} - \varphi^1_{L1s(s)} \geq 1$



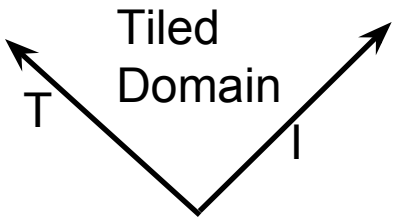
Algorithm

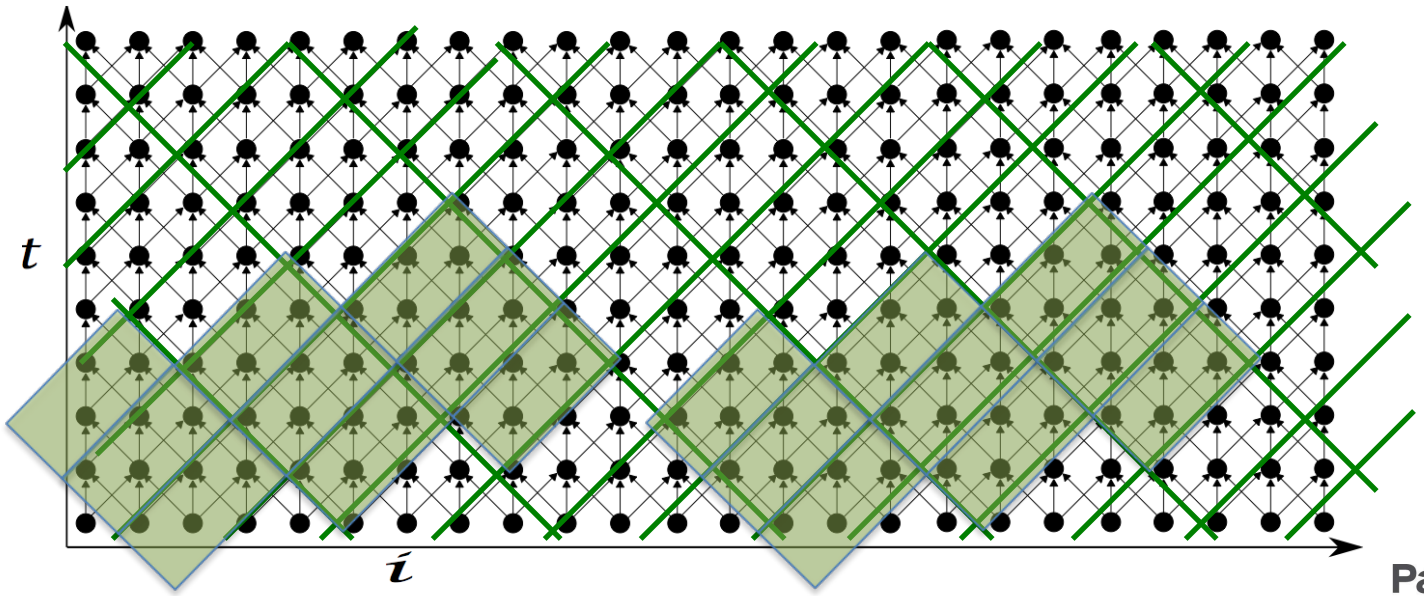
4. Tile for L2 using PLUTO:

Example:

Tiled Domain: $3T_{L2} <= T_{L1}+I_{L1} <= 3T_{L2}+2$

$3I_{L2} <= I_{L1} <= 3I_{L2}+2$

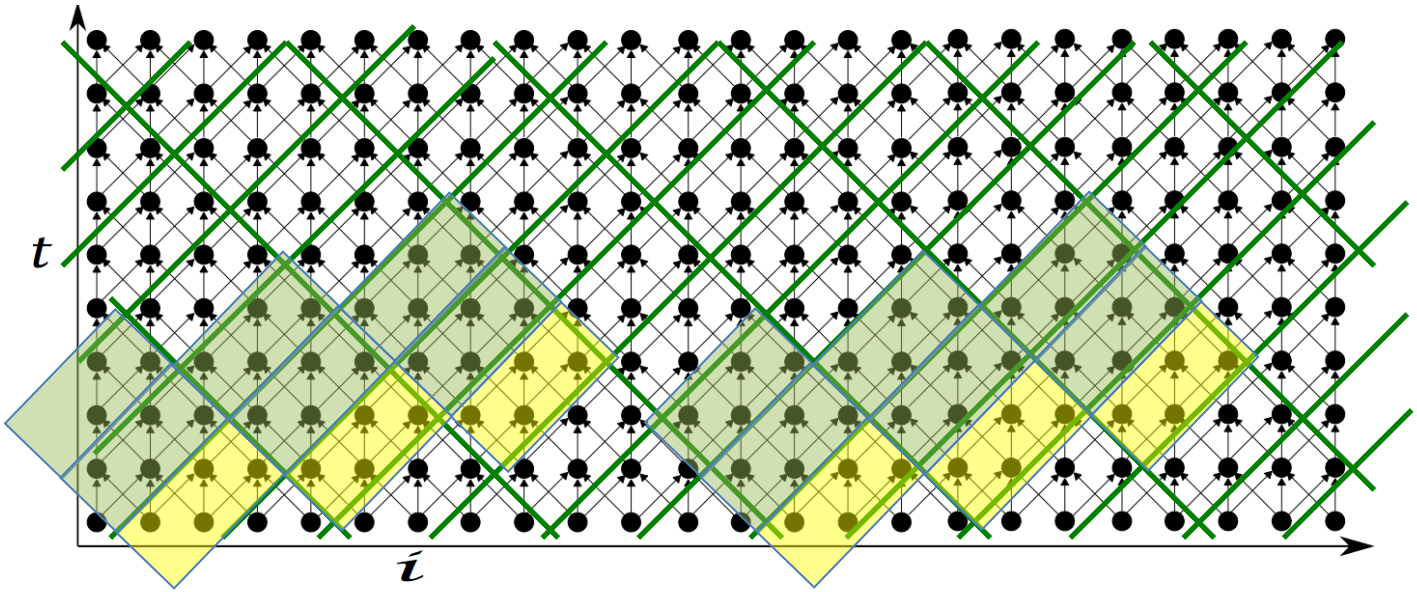




Algorithm

5. Perform Unimodular transformation on L1 supernode.

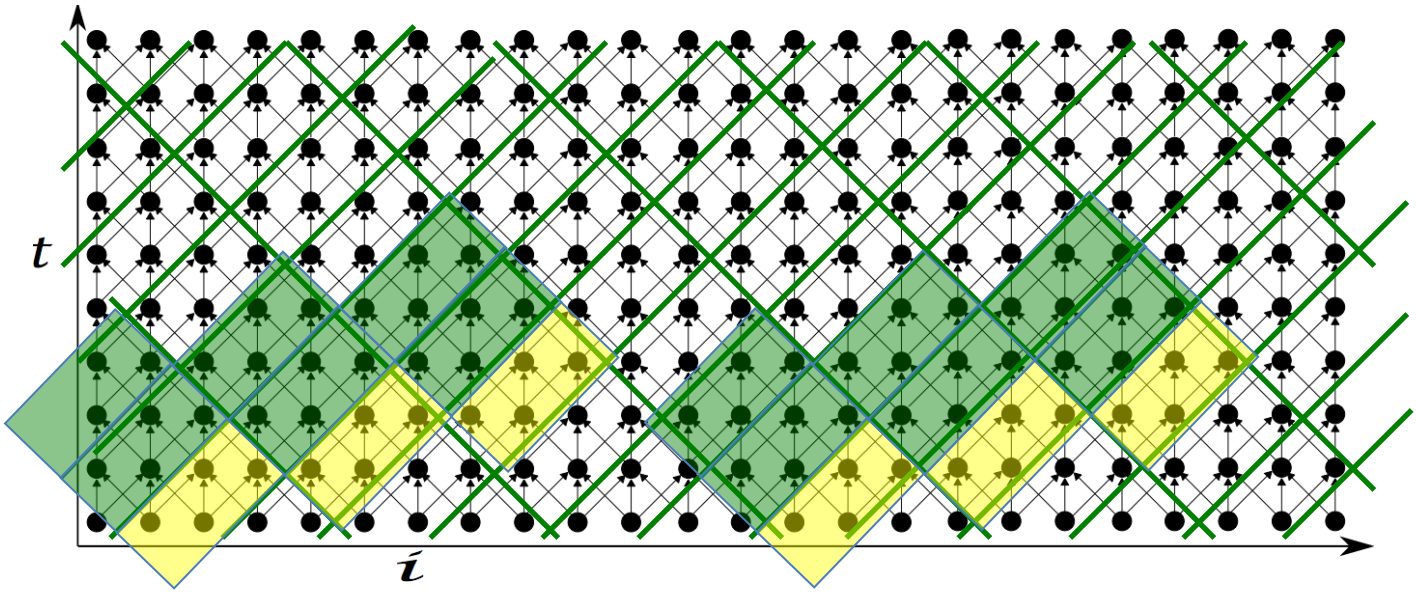
$$\varphi T^1_{L1s} \rightarrow \varphi T^1_{L1s} + \varphi T^2_{L1s}$$



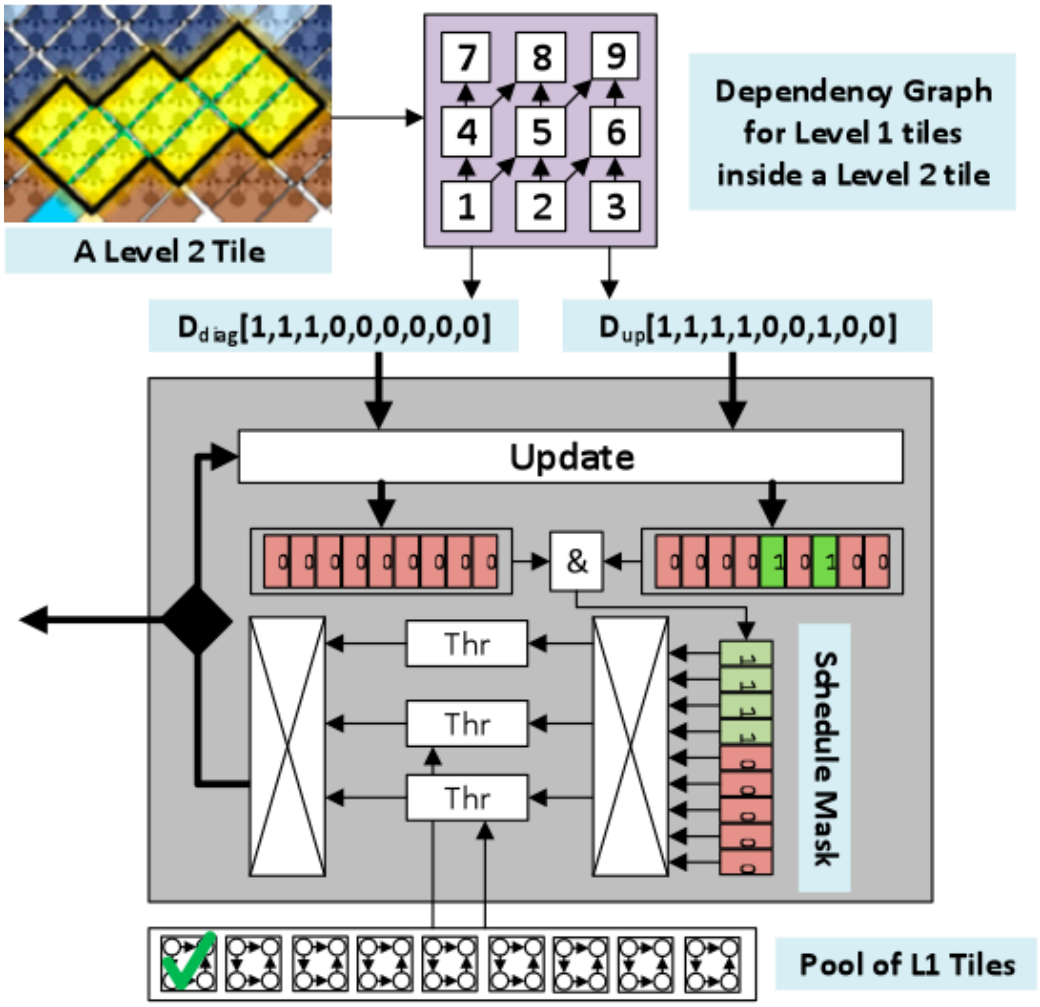
Algorithm

6. Perform Unimodular transformation on L2 supernode:

$$\Phi T^1_{L2s} \rightarrow \Phi T^1_{L2s} + \Phi T^2_{L2s}$$

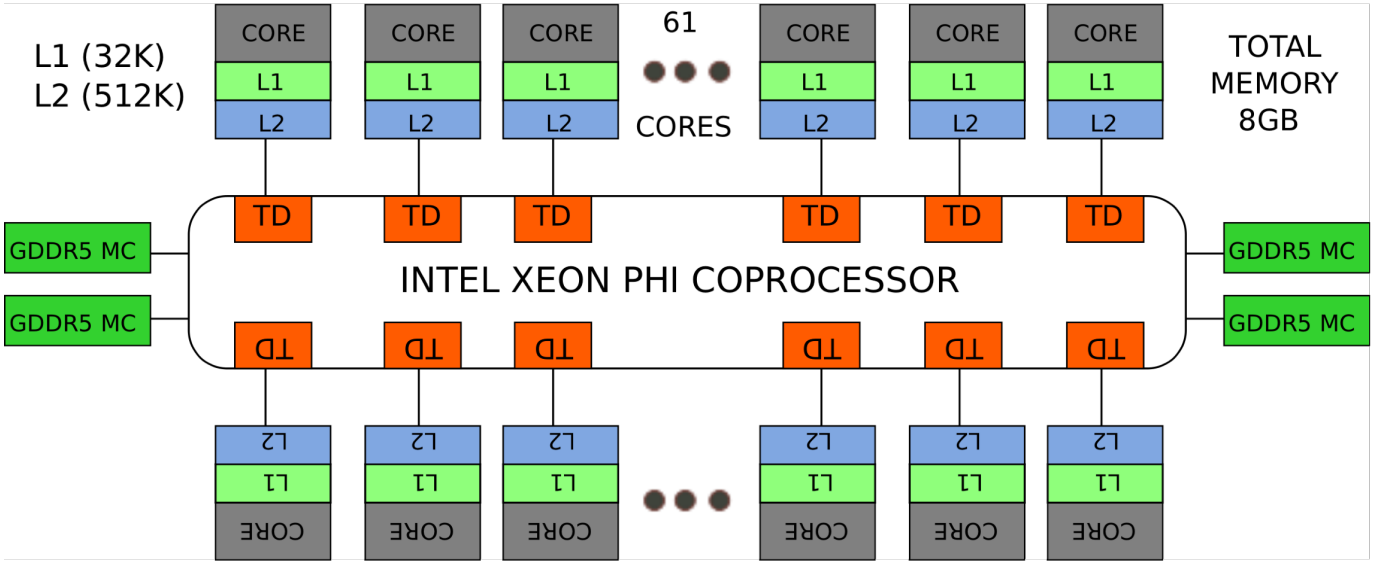


Fine-Grain Execution



- Thread Grouping
- Take advantage of outer tile(L2) locality.
- Uses low overhead atomic operation
- Hierarchical

Experimental Platform



TD: Tag Directory

- 61 cores, 1.1GHz with 4 hyper-threads
- L1 Cache: 1 cycle
- Unified L2 Cache: ~12 cycles
- DRAM:~230 cycles)
- Shared ring L2 caches: 180-250 cycles

- **Intel Xeon Phi 7110P coprocessor**

Applications and Sizes

Application	Size (N)	Size (T)
Heat-1d	10000000	10K
Heat-2d	11504x11504	2K
Heat-3d	480x480x480	100
Jacobi-2d	11504x11504	2K
7point-3d	480x480x480	100

Performance

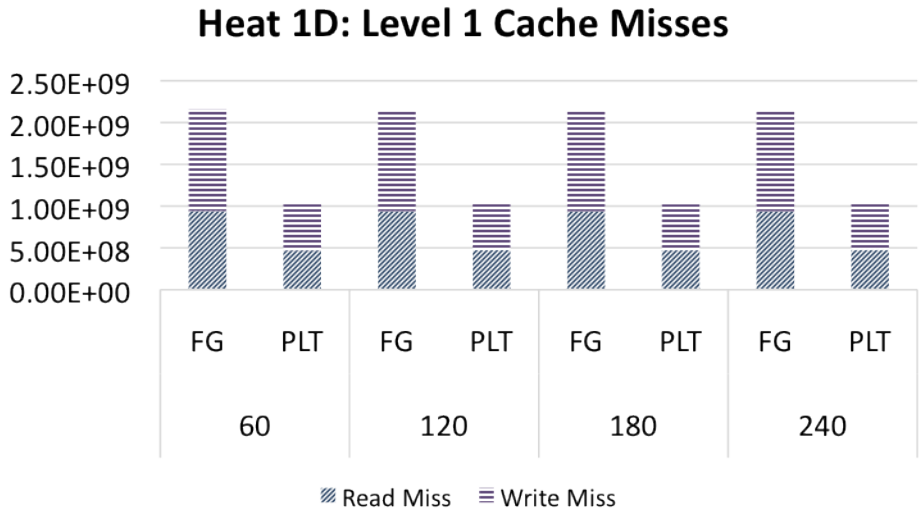
PLUTO OpenMP

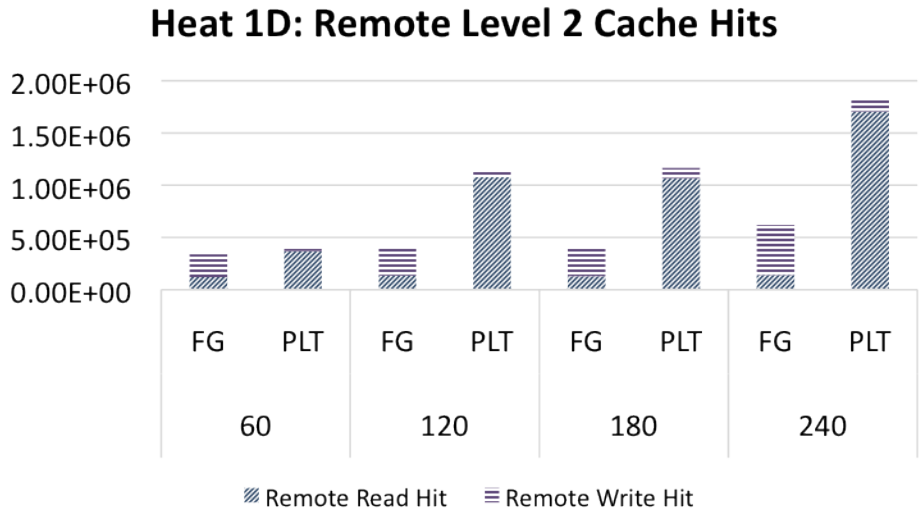
Application	Balanced	Compact	Tile Size	Hierarchical
Heat-1d	106.59	107.05	4Kx4K	99.65
Heat-2d	122.42	122.78	16x16x256	109.78
Heat-3d	59.375	57.22	3x3x2x480	**27.91**
Jacobi-2d	68.26	68.40	16x16x256	-
7point-3d	31.48	31.82	2x2x4x480	-

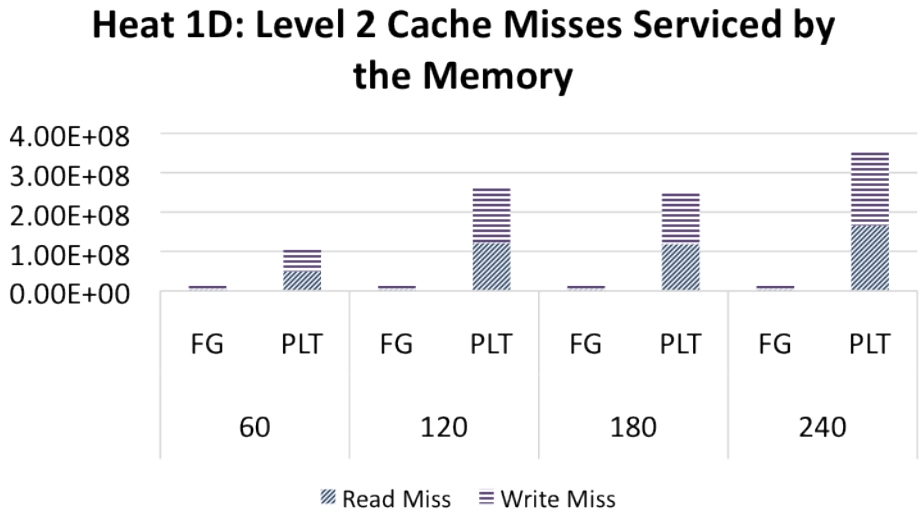
FineGrain Pthread

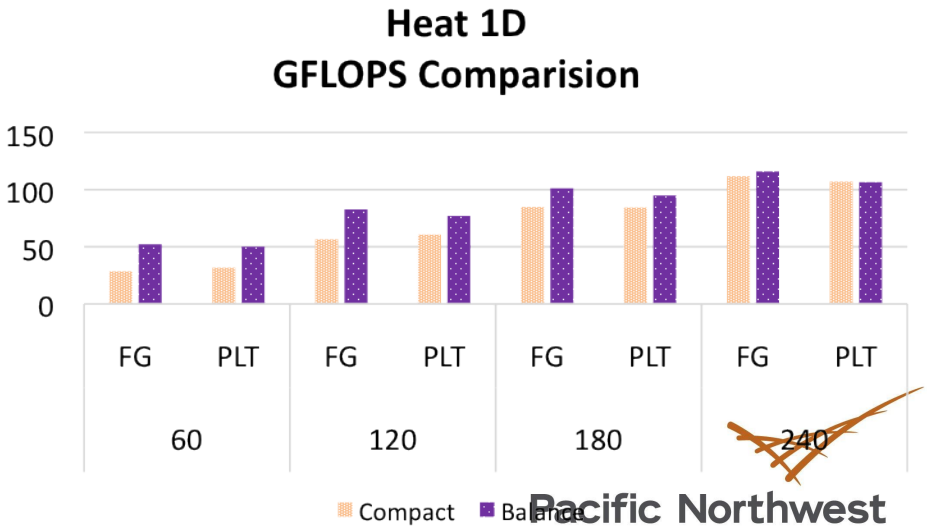
Application	Balanced	Compact	Tile Size	Gain
Heat-1d	115.95	111.77	1Kx2K / 4x4	8.31%
Heat-2d	131.47	134.95	16x32x256 / 4x4x2	9.91%
Heat-3d	61.73	74.168	1x2x4x480/4x4x2x1	**24.91%**
Jacobi-2d	67.94	72.22	16x32x256 / 4x4x2	5.58%
7point-3d	33.46	41.74	1x2x4x480/4x4x2x1	**31.17%**

Heat 1-D

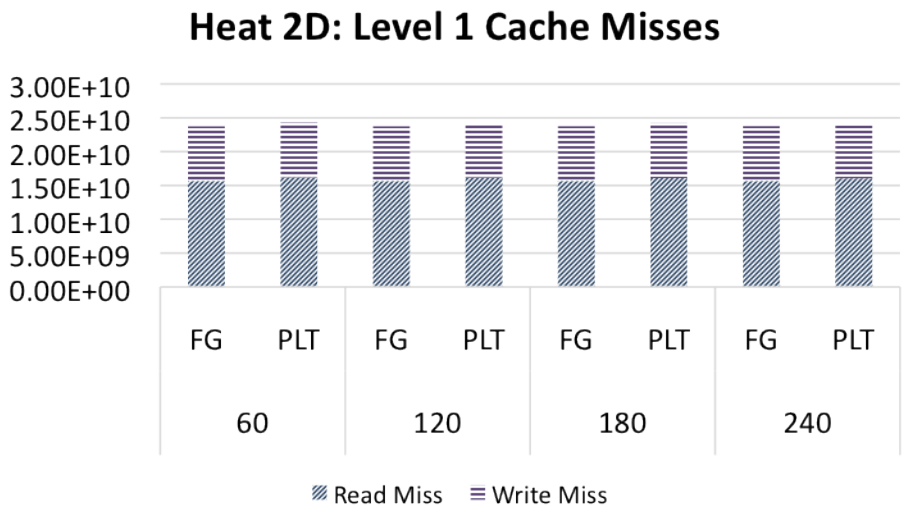

Heat 1D: Level 1 Cache Misses
2.50E+09
2.00E+09
1.50E+09
1.00E+09
5.00E+08
0.00E+00
FG PLT FG PLT FG PLT FG PLT
60 120 180 240
Read Miss
Write Miss

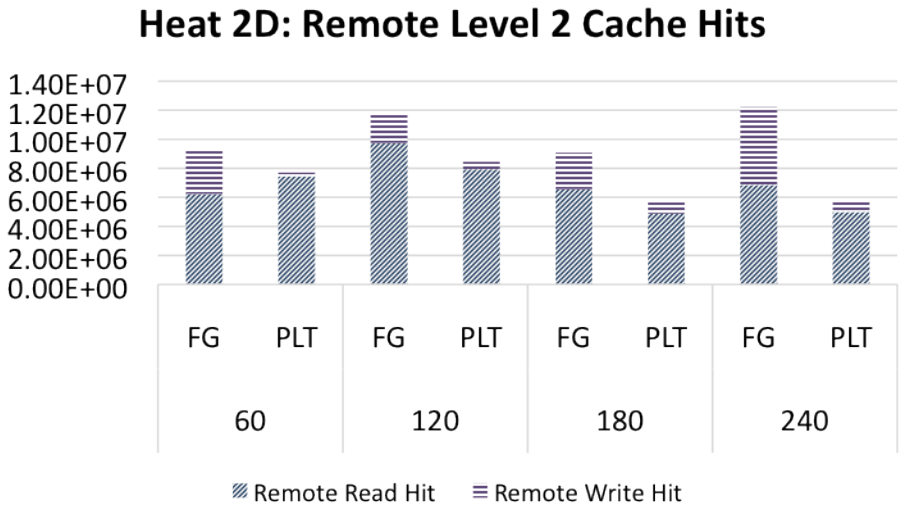

Heat 1D: Remote Level 2 Cache Hits
2.00E+06
1.50E+06
1.00E+06
5.00E+05
0.00E+00
FG PLT FG PLT FG PLT FG PLT
60 120 180 240
Remote Read Hit
Remote Write Hit

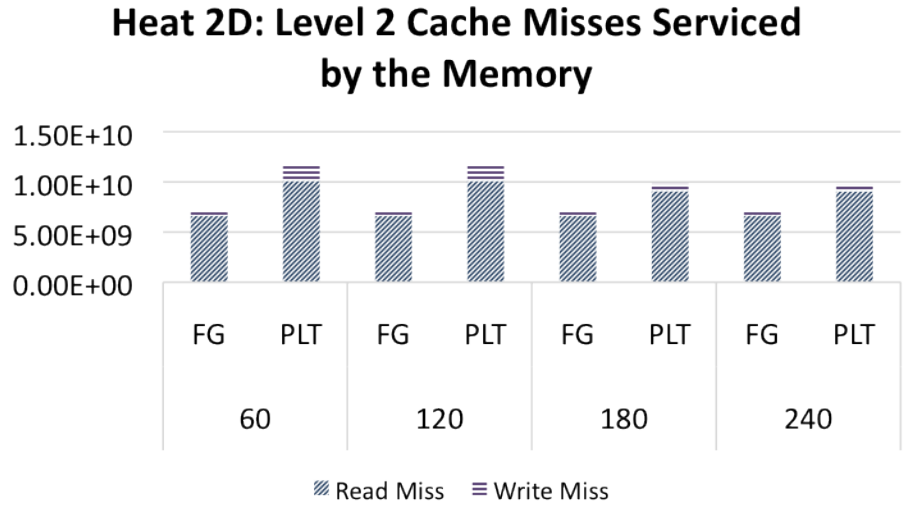

Heat 1D: Level 2 Cache Misses Serviced by the Memory
4.00E+08
3.00E+08
2.00E+08
1.00E+08
0.00E+00
FG PLT FG PLT FG PLT FG PLT
60 120 180 240
Read Miss
Write Miss

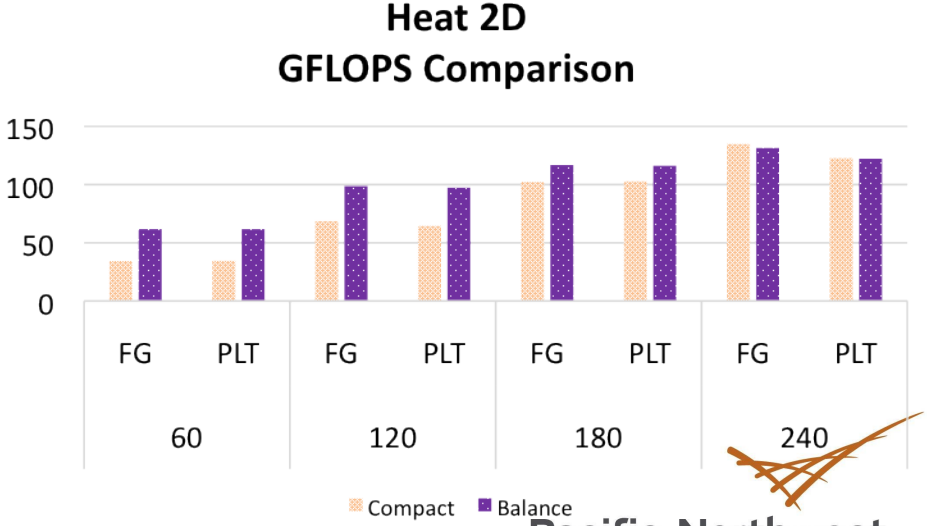

Heat 1D
GFLOPS Comparision
150
100
50
0
FG PLT FG PLT FG PLT FG PLT
60 120 180 240
Compact
Balance

Heat 2-D



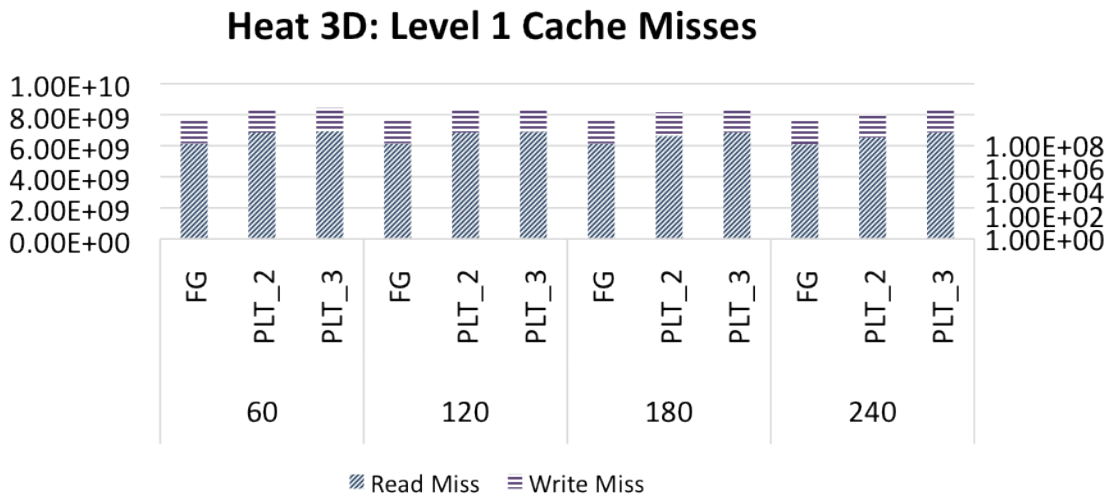


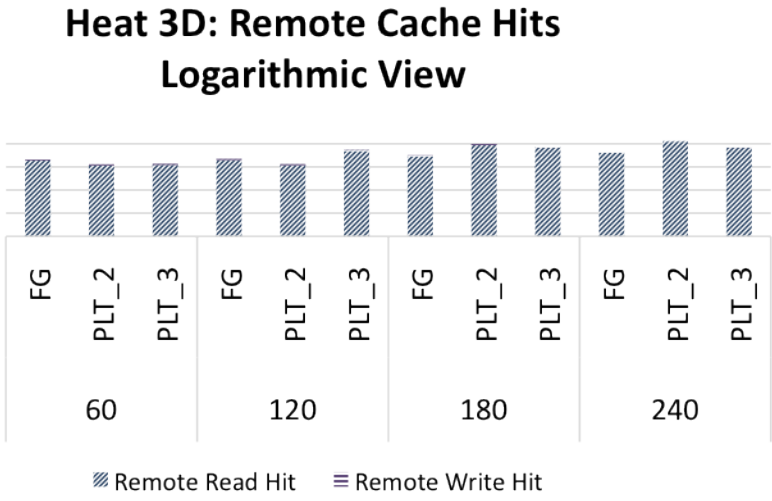


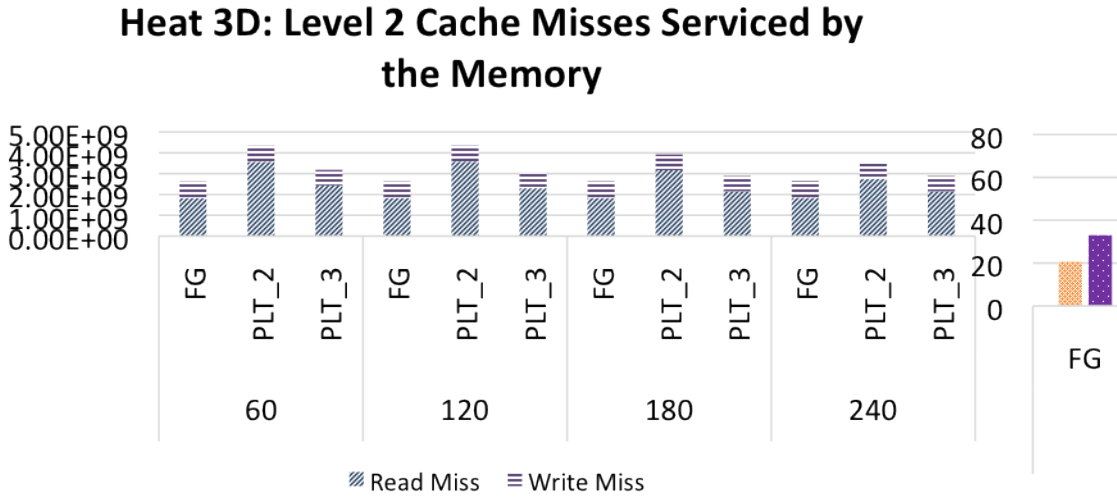


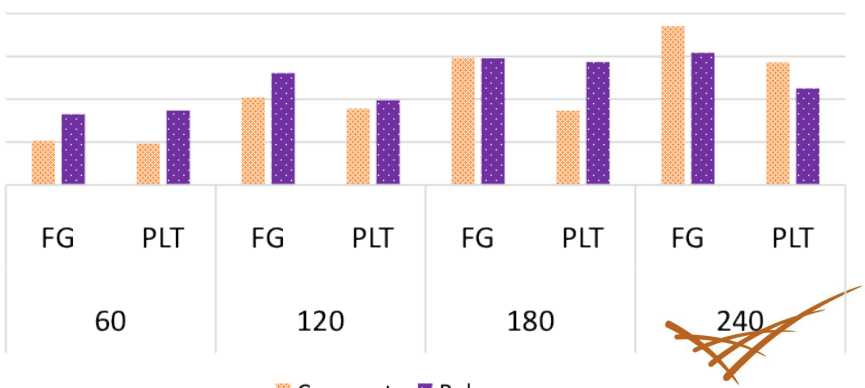
Pacific Northwest
NATIONAL LABORATORY

Heat 3-D









Pacific Northwest NATIONAL LABORATORY

Conclusion and Future Work

- Hierarchical tiling technique that improves locality and reuse.
- Collaborative view with grouping of threads.
- Orchestration of data movement among threads working in close proximity in time and space to improve reuse.
- Explore other multicore architectures.

Back up slides

Iteration Space Domain
An Example

```
for (i=1; i<n;i++) {
   for (j=1; j<n;j++){
      A[i][j]=A[i-1][j]+A[i][j-1];
   }
}
```

Stencil Example

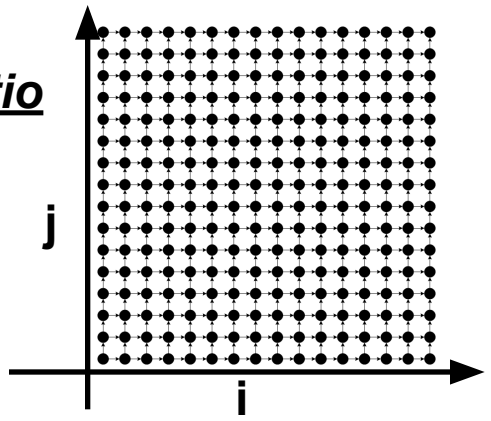
$$\begin{pmatrix} 1 & 0 & 0 & -1 \\ -1 & 0 & 1 & 0 \\ 0 & 1 & 0 & -1 \\ 0 & -1 & 1 & 0 \end{pmatrix} \begin{pmatrix} i \\ j \\ n \\ 1 \end{pmatrix} \geq 0$$

Iteration Space as Matrices

$$\begin{aligned} i &\geq 1 \\ j &\geq 1 \\ i \leq n \text{ or } -i+n &\geq 0 \\ j \leq n \text{ or } -j+n &\geq 0 \end{aligned}$$

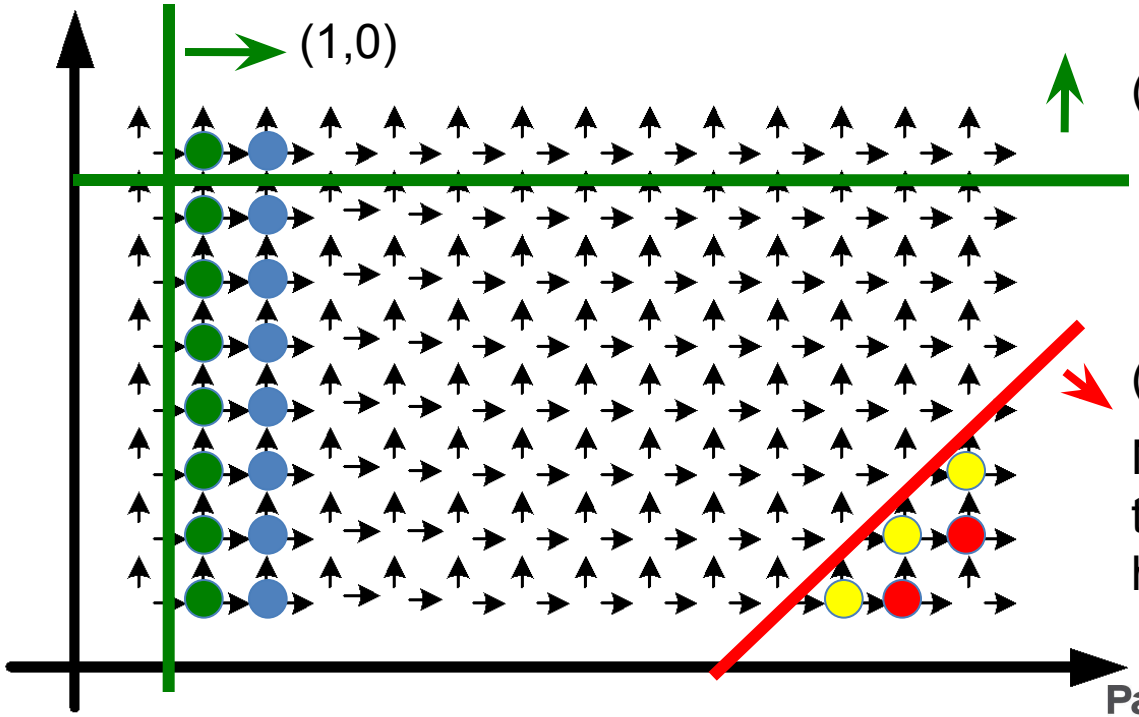
Iteration space as a System of Equations

Graphical Representation of the Iteration Space



Background

- Hyperplane
  - (n-1) dimensional affine subspace in n dimensional space
- Tiling Hyperplane
  - For source(s) and target(t): $\phi_{S_i}(\boldsymbol{t}) - \phi_{S_j}(\boldsymbol{s}) \geq 0$
  - Dependencies are satisfied or can be satisfied within.
- Example



Scheduling Hierarchical Tiled Domain

```
for (t=0; t<T;t++) {
  for (j=1; j<N;j++){
    A[t+i][j]=α*(A[t][i+1]-
              β*A[t][i] + A[t][i-1]);
  }
}
```

Tiling Hyperplanes

L1 Tile: (1,1) and (1,-1)
L2 Tile: (1,0) and (0,1)

Schedule:

$$\begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$$

A Stencil Example