Modeling Stencil Computations on Modern HPC Architectures

Raúl de la Cruz[1] and Mauricio Araya-Polo[2]

delacruz@bsc.es, mauricio.araya@shell.com

Barcelona Supercomputing Center (BSC-CNS), Barcelona (Spain)[1]
Shell Intl. E&P Inc., Houston, Texas (USA)[2]

Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems (SC14)
New Orleans, LA, USA, November 16th, 2014

Overview

Stencil in a Nutshell

Finite Difference Method

```
Domain decomposition of mesh
for time = 0 to time_end do
    Read Input
    Pre-processing
    Inject source
    Apply boundary conditions
    for all points in my domain do
        Stencil computation (X^t)
    end for
    Exchange overlapped points
    Post-processing
    Write Output
end for
```

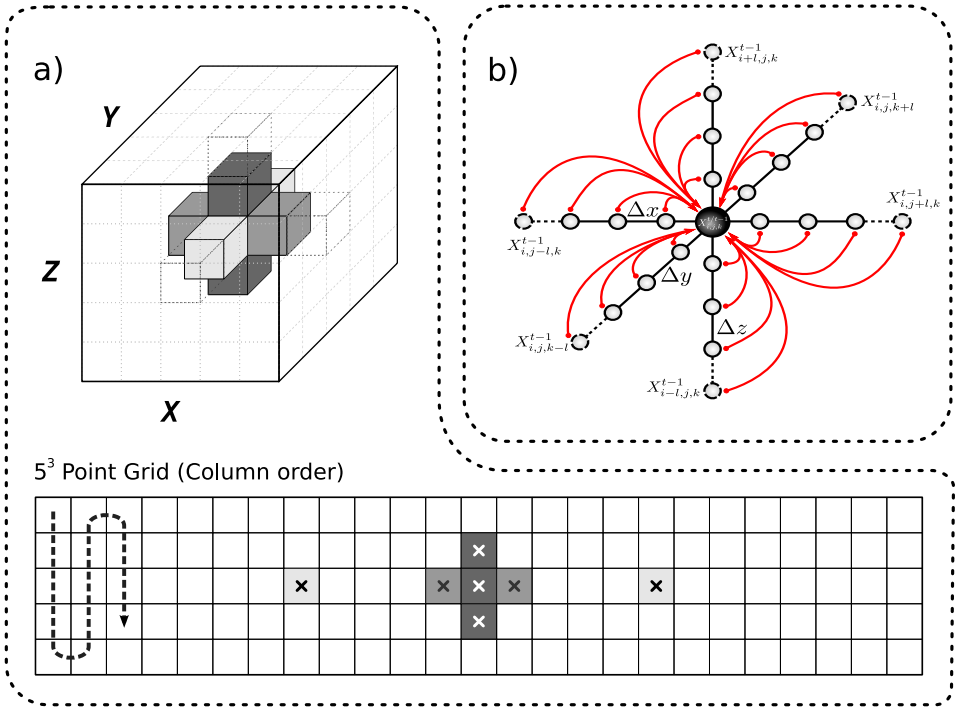
- Load balancing
- Kernel computation
- Intra/inter-node communication

Stencil Computation

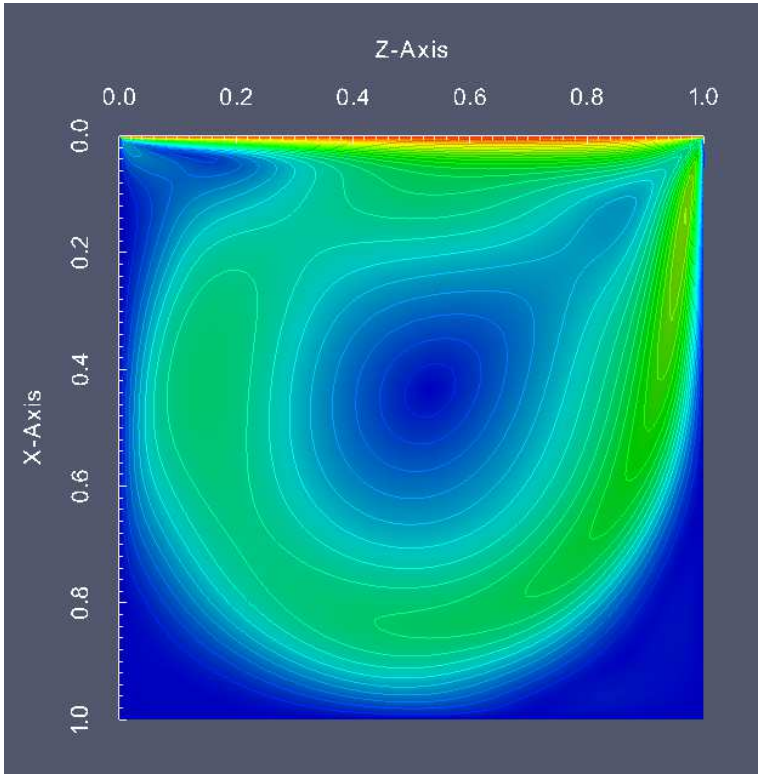
for $k = \ell$ to $Y - \ell$ **do**
  for $j = \ell$ to $X - \ell$ **do**
    for $i = \ell$ to $Z - \ell$ **do**

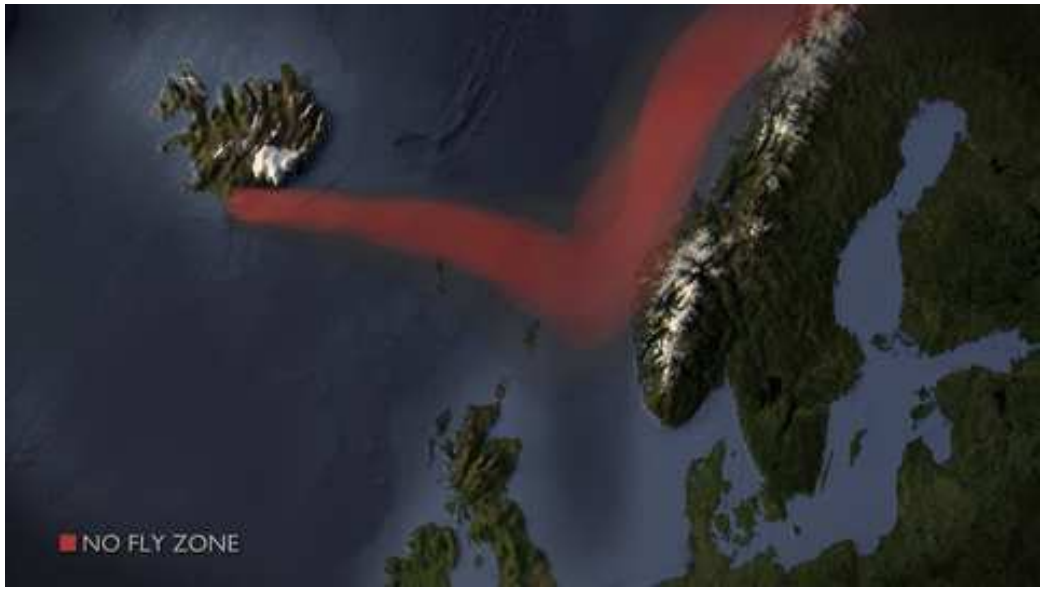
$$\begin{aligned}\mathcal{X}^{t}_{i,j,k} &= C_0 * \mathcal{X}^{t-1}_{i,j,k} \\ &+ C_{Z1} * (\mathcal{X}^{t-1}_{i-1,j,k} + \mathcal{X}^{t-1}_{i+1,j,k}) + .. + C_{Z\ell} * (\mathcal{X}^{t-1}_{i-\ell,j,k} + \mathcal{X}^{t-1}_{i+\ell,j,k} \\ &+ C_{X1} * (\mathcal{X}^{t-1}_{i,j-1,k} + X^{t-1}_{i,j+1,k}) + .. + C_{X\ell} * (\mathcal{X}^{t-1}_{i,j-\ell,k} + \mathcal{X}^{t-1}_{i,j+\ell,k} \\ &+ C_{Y1} * (\mathcal{X}^{t-1}_{i,j,k-1} + X^{t-1}_{i,j,k+1}) + .. + C_{Y\ell} * (\mathcal{X}^{t-1}_{i,j,k-\ell} + \mathcal{X}^{t-1}_{i,j,k+\ell}\end{aligned}$$

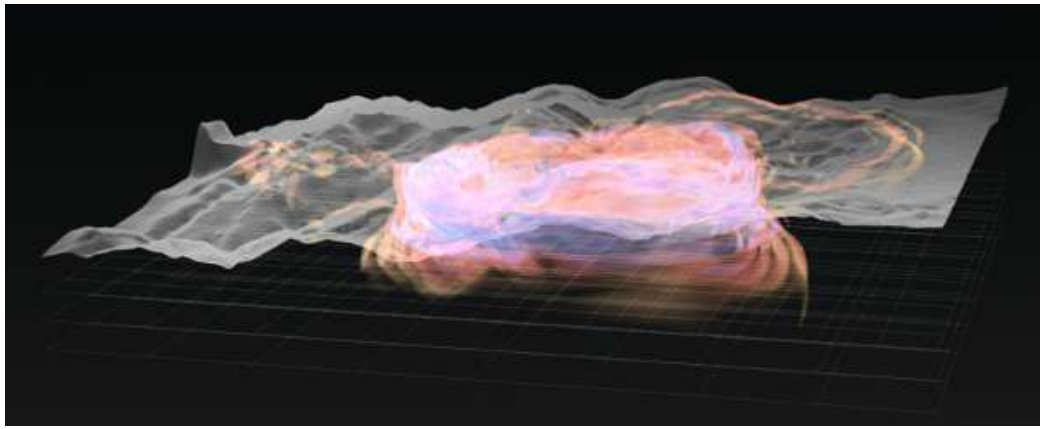
    end for
  end for
end for



Stencil in the Academia & Industry





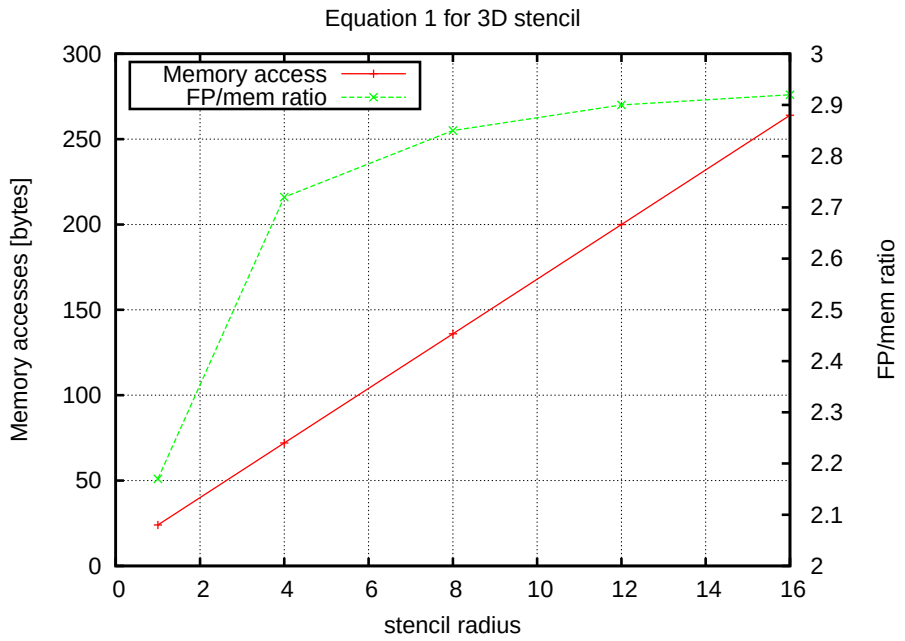


Stencil Performance Challenges

Two main challenges:

1. Low FLOPs/Memory ratio

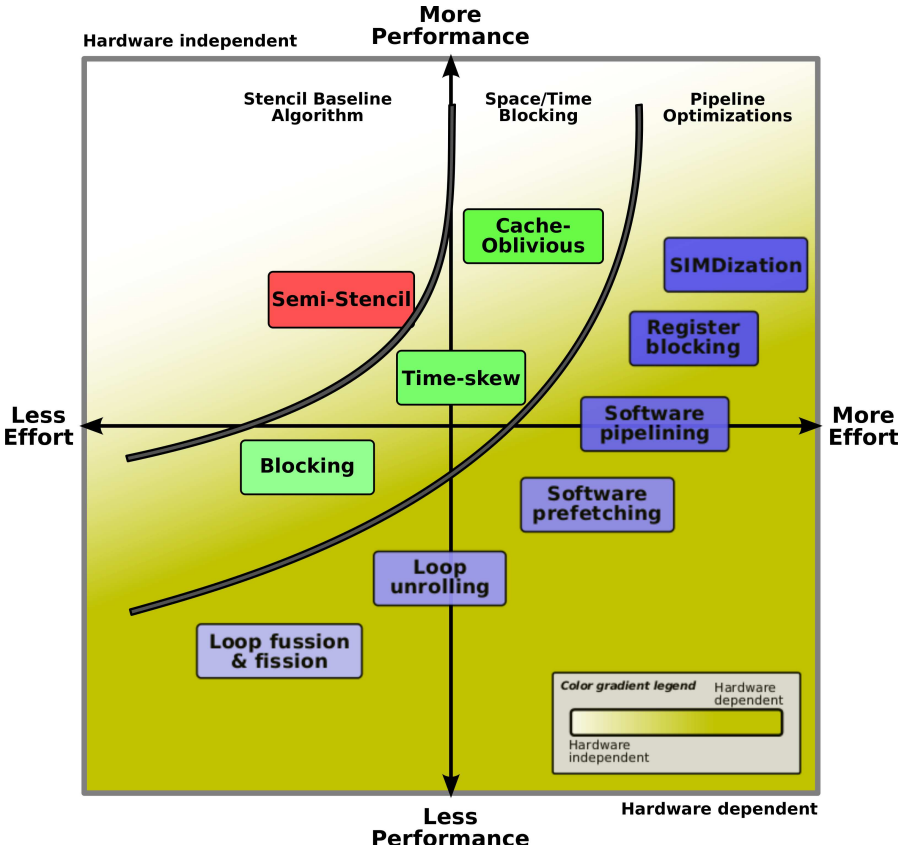
$$
\begin{aligned}
FP/Cache_{Classical} &= \frac{FloatingPoint\ Operations}{Data\ Cache\ Accesses} \\
&= \frac{2 * MultiplyAdd\ Instructions}{X^{t-1}\ Loads + X^{t}\ Stores} \\
&= \frac{2 * 2 * dim * \ell + 1}{\underbrace{(2 * (dim - 1) * \ell + 1)}_{X^{t-1}\ Loads} + \underbrace{(1)}_{X^{t}\ Stores}} \\
&= \frac{4 * dim * \ell + 1}{2 * dim * \ell - 2 * \ell + 2}
\end{aligned}
\tag{1}
$$

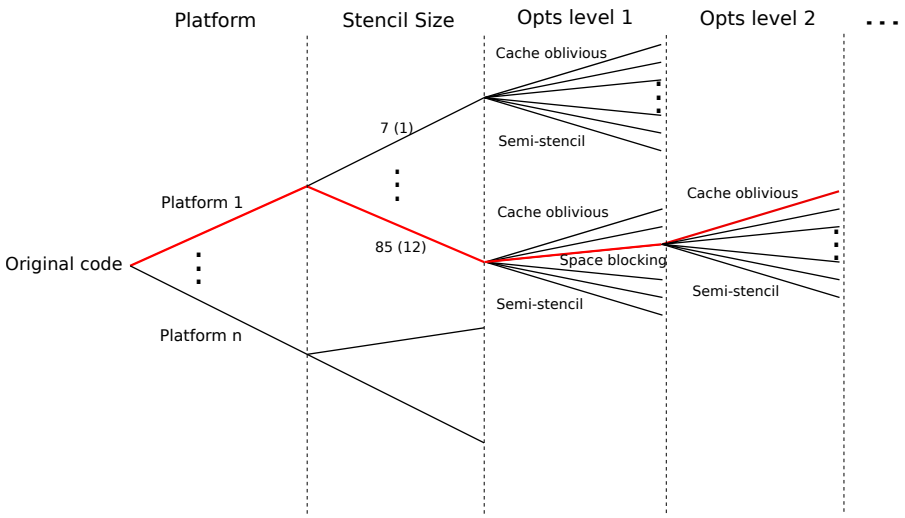


2. Low data reuse

Overview

Motivation: Modeling Stencil Performance

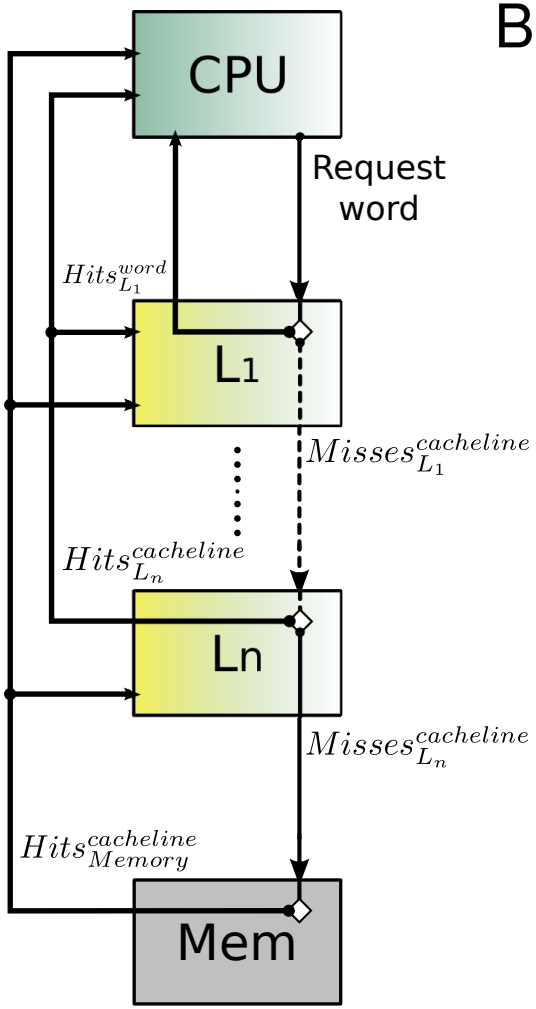




- Which one/combination of these should be implemented?
- How much performance improvement can be expected?
- Brute force (autotuning) is too expensive (combinatorial explosion, code management) then modeling seems to be a valid alternative

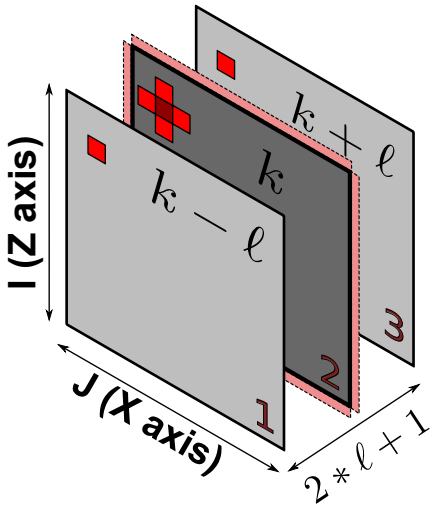
Overview

Stencil Performance Model: Base Model I



Basic considerations:

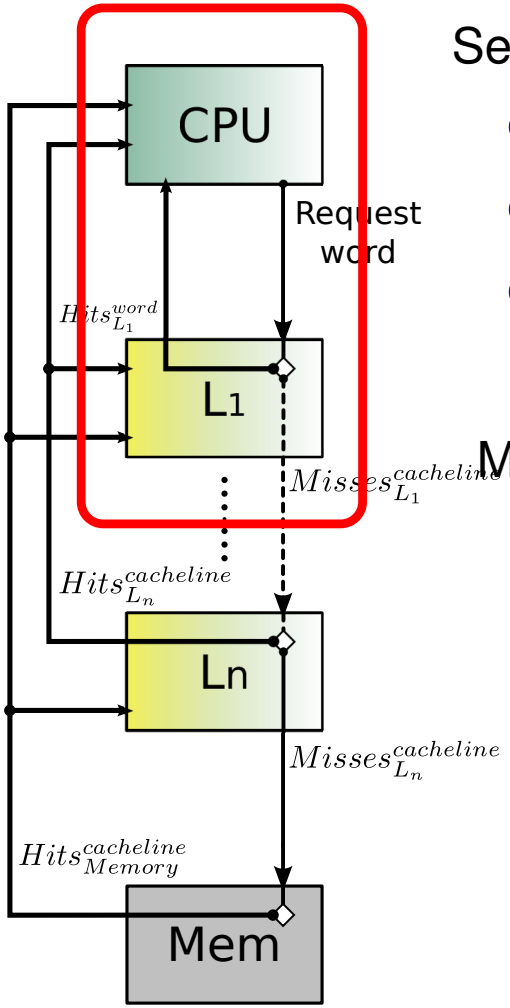
- Computation bottleneck is negligible (low FP/Byte) and reads dominate
- No cache interferences between instructions and data
- P_{read} $(2 * \ell + 1)$ Z-X planes from $\mathcal{X}^{t-1}$ to compute one $\mathcal{X}^{t}$ output plane (P_{write})
- $S_{total} = P_{read} \times S_{read} + P_{write} \times S_{write}$, being $S_{read} = II \times JJ$ and $S_{write} = I \times J$
- Three memory groups are established



$$T_{total} = \underbrace{T_{L1}}_{first} + \underbrace{\cdots + T_{Li} + \cdots + T_{Ln}}_{intermediate} + \underbrace{T_{Memory}}_{last}$$

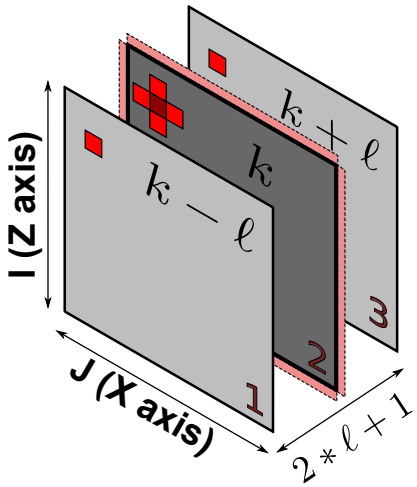
What are the $Hits_{Li}$ and $Misses_{Li}$ for each hierarchy level?

Stencil Performance Model: Base Model II



Several data are involved in a stencil comput.:

- Grid points (Input: $\mathcal{X}^{t-1}$, Output: $\mathcal{X}^{t}$)
- Contribution weights ($C_{Z,X,Y,0}$)
- Indices to access grid points (i, j, k)



Misses depend on II, JJ sizes and ℓ parameter:

$$Misses_{Li}^{cline} = \lceil II/W \rceil * JJ * KK * nplanes_{Li}$$

$$Hits_{Li}^{cline} = Misses_{Li-1}^{cline} - Misses_{Li}^{cline}$$

$$T_{Li}^{cline} = cacheline/Bw_{Li}^{read}$$

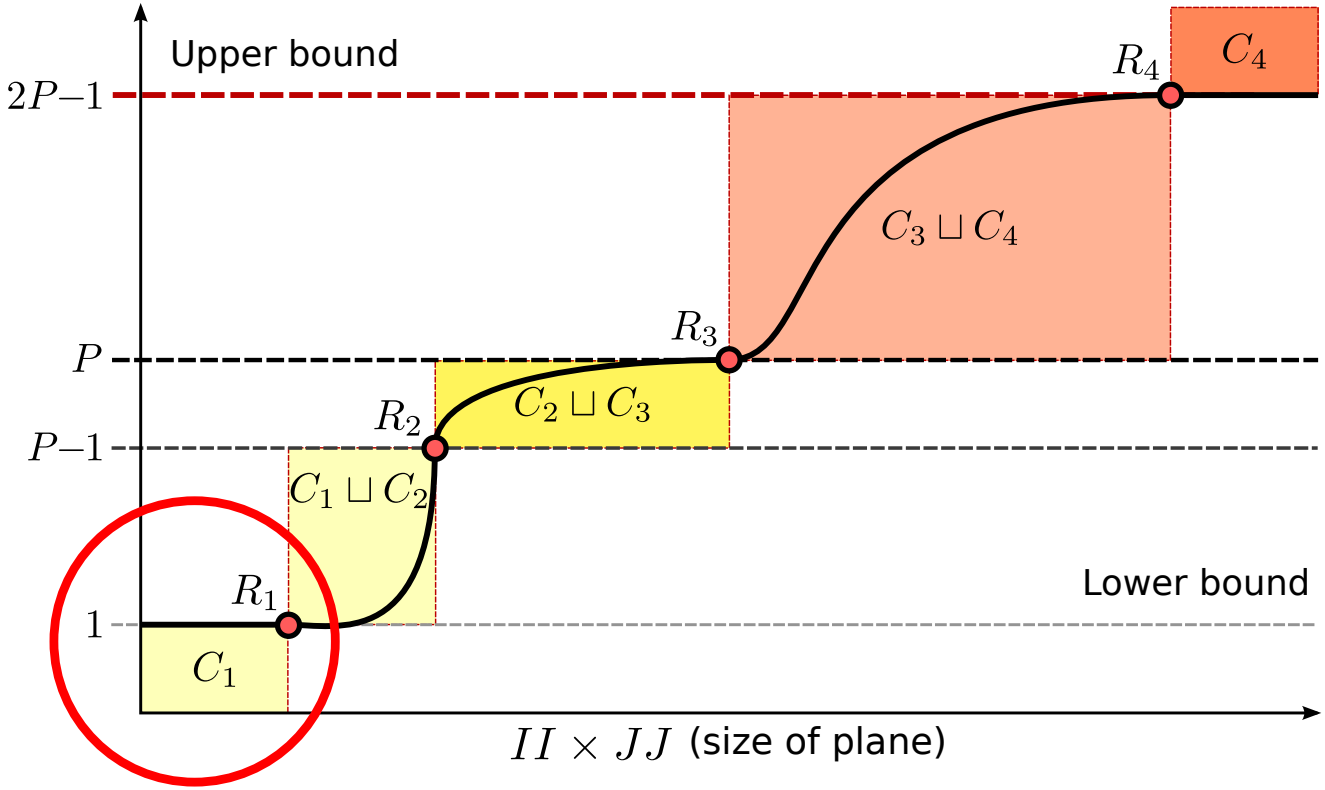
$$T_{Li} = Hits_{Li}^{cline} * T_{Li}^{cline}$$

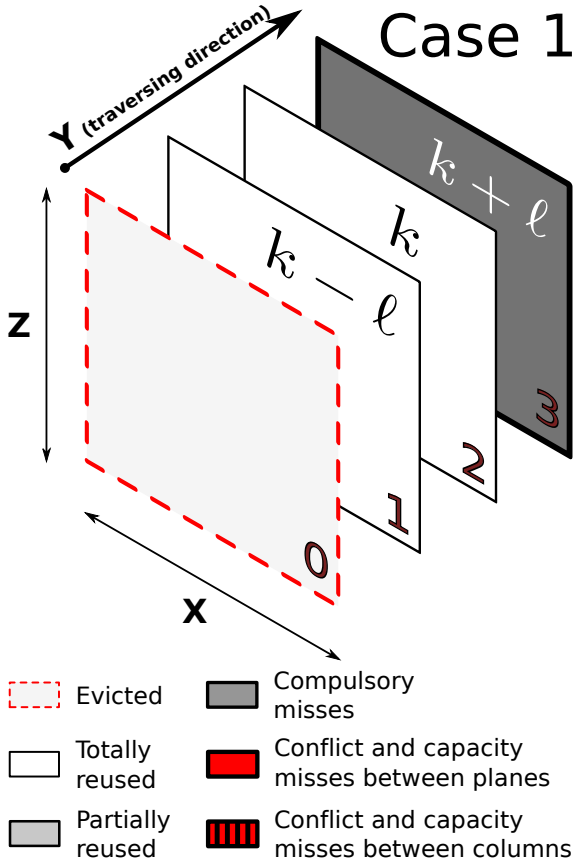
- $nplanes_{Li}$: $II \times JJ$ planes read from $Li+1$ for each k iteration

(Basic model description in ICCS-iWAPT2011 paper)

Stencil Model: Read Miss Cases & Rules

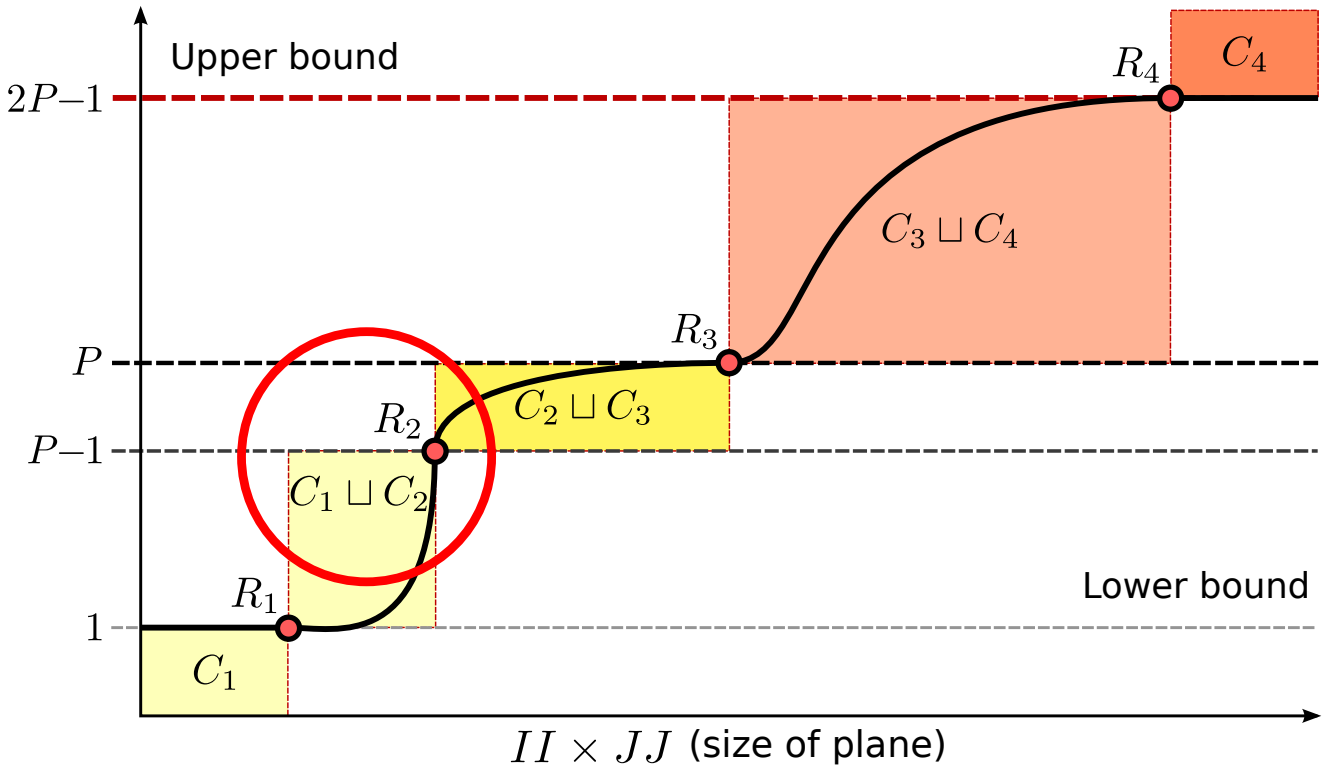
$$nplanes_{Li}(II, JJ) = \begin{cases} C_1 : 1, & \text{if } R_1 \\ C_1 \sqcup C_2 : (1, P_{read} - 1], & \text{if } \neg R_1 \wedge R_2 \\ C_2 \sqcup C_3 : (P_{read} - 1, P_{read}], & \text{if } \neg R_2 \wedge R_3 \\ C_3 \sqcup C_4 : (P_{read}, 2P_{read} - 1], & \text{if } \neg R_3 \wedge \neg R_4 \\ C_4 : 2P_{read} - 1, & \text{if } R_4 \ , \end{cases}$$

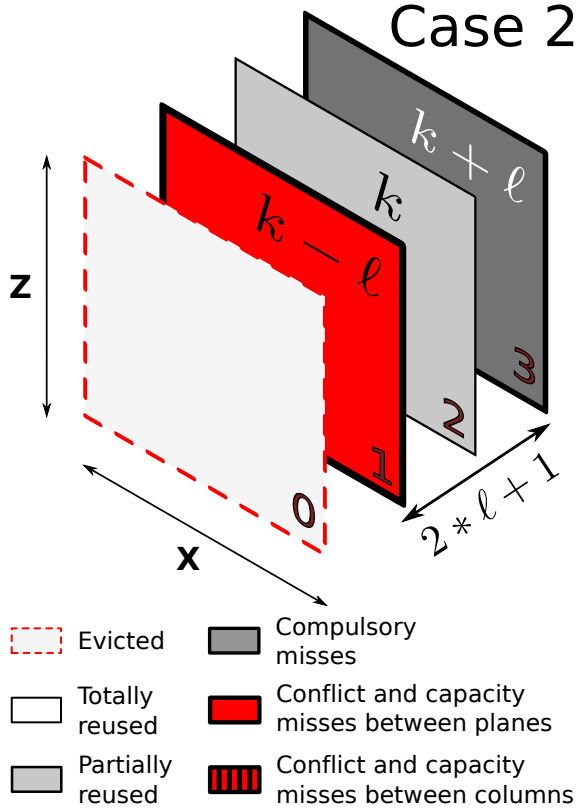




Stencil Model: Read Miss Cases & Rules

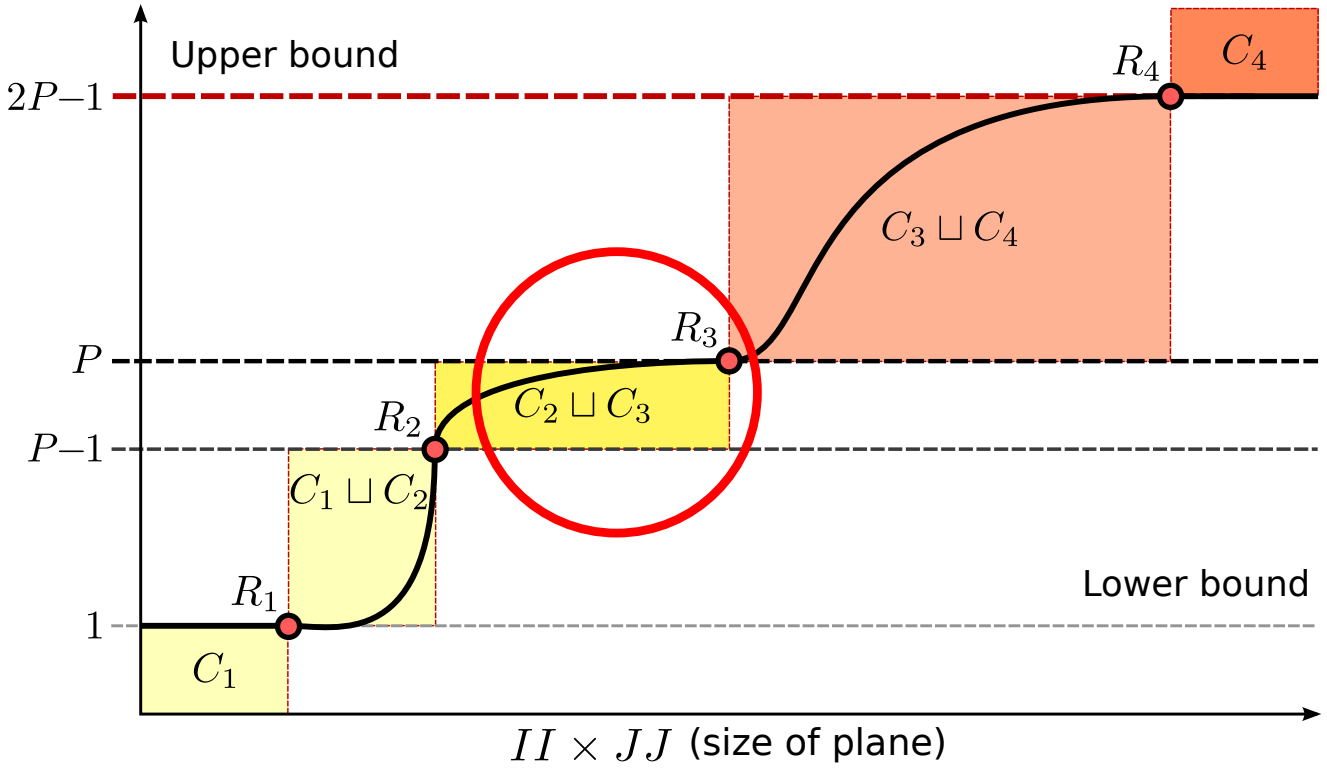
$$nplanes_{Li}(II, JJ) = \begin{cases} C_1 : 1, & \text{if } R_1 \\ C_1 \sqcup C_2 : (1, P_{read} - 1], & \text{if } \neg R_1 \wedge R_2 \\ C_2 \sqcup C_3 : (P_{read} - 1, P_{read}], & \text{if } \neg R_2 \wedge R_3 \\ C_3 \sqcup C_4 : (P_{read}, 2P_{read} - 1], & \text{if } \neg R_3 \wedge \neg R_4 \\ C_4 : 2P_{read} - 1, & \text{if } R_4 \ , \end{cases}$$

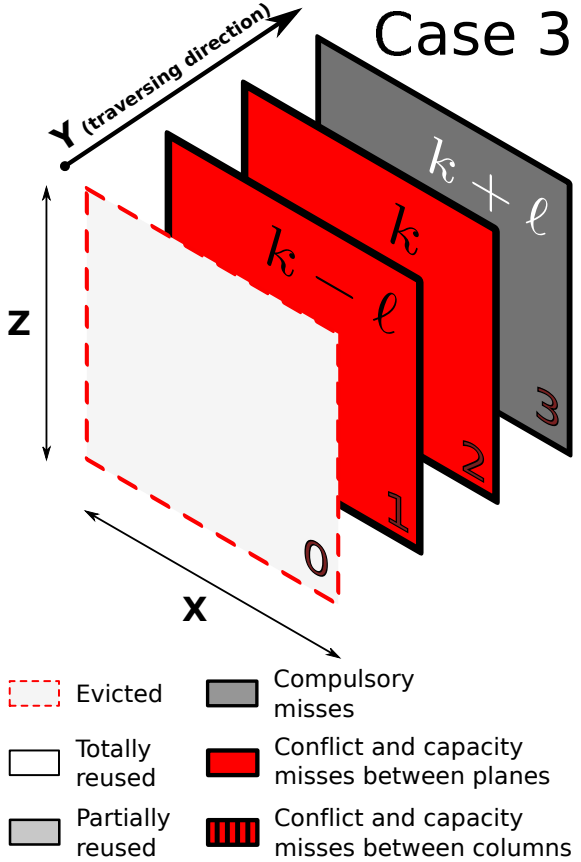




Stencil Model: Read Miss Cases & Rules

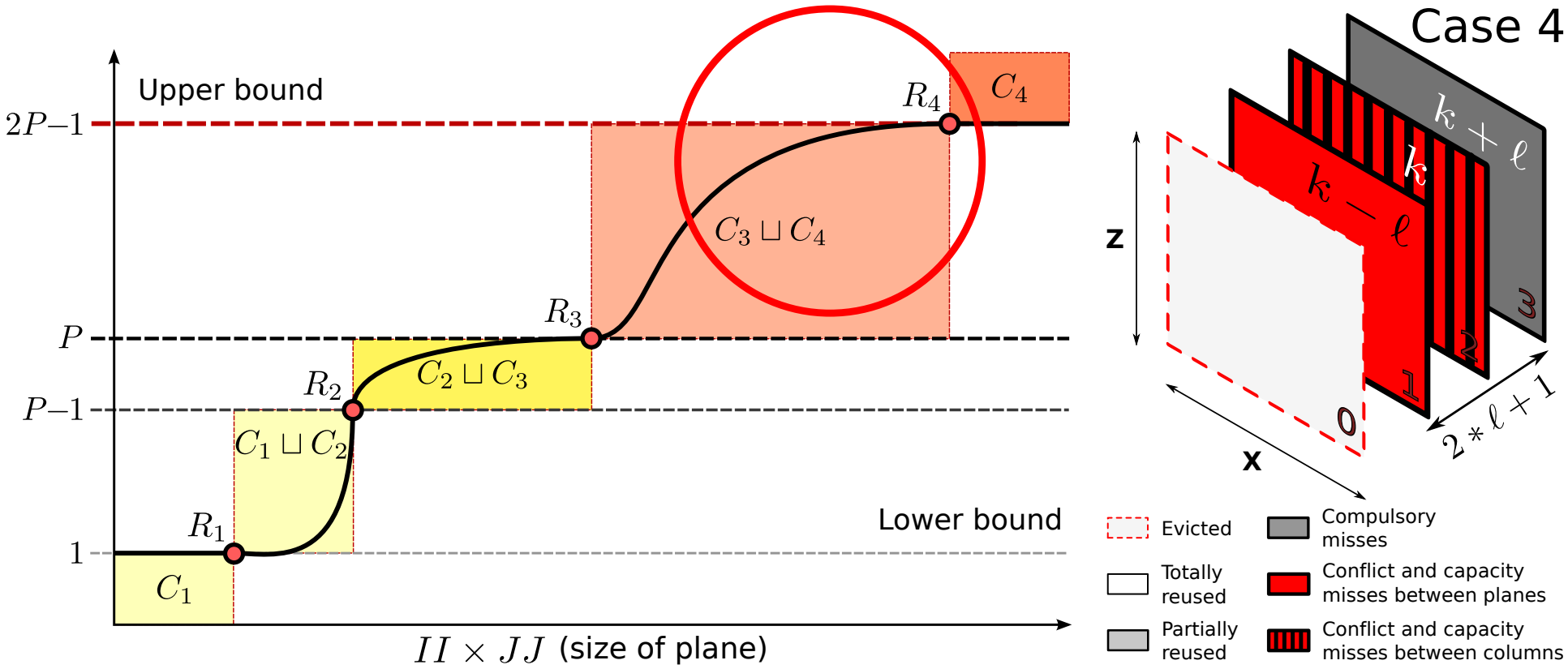
$$nplanes_{Li}(II, JJ) = \begin{cases} C_1 : 1, & \text{if } R_1 \\ C_1 \sqcup C_2 : (1, P_{read} - 1], & \text{if } \neg R_1 \wedge R_2 \\ C_2 \sqcup C_3 : (P_{read} - 1, P_{read}], & \text{if } \neg R_2 \wedge R_3 \\ C_3 \sqcup C_4 : (P_{read}, 2P_{read} - 1], & \text{if } \neg R_3 \wedge \neg R_4 \\ C_4 : 2P_{read} - 1, & \text{if } R_4 \ , \end{cases}$$





Stencil Model: Read Miss Cases & Rules

$$nplanes_{Li}(II, JJ) = \begin{cases} C_1 : 1, & \text{if } R_1 \\ C_1 \sqcup C_2 : (1, P_{read} - 1], & \text{if } \neg R_1 \wedge R_2 \\ C_2 \sqcup C_3 : (P_{read} - 1, P_{read}], & \text{if } \neg R_2 \wedge R_3 \\ C_3 \sqcup C_4 : (P_{read}, 2P_{read} - 1], & \text{if } \neg R_3 \wedge \neg R_4 \\ C_4 : 2P_{read} - 1, & \text{if } R_4 \ , \end{cases}$$

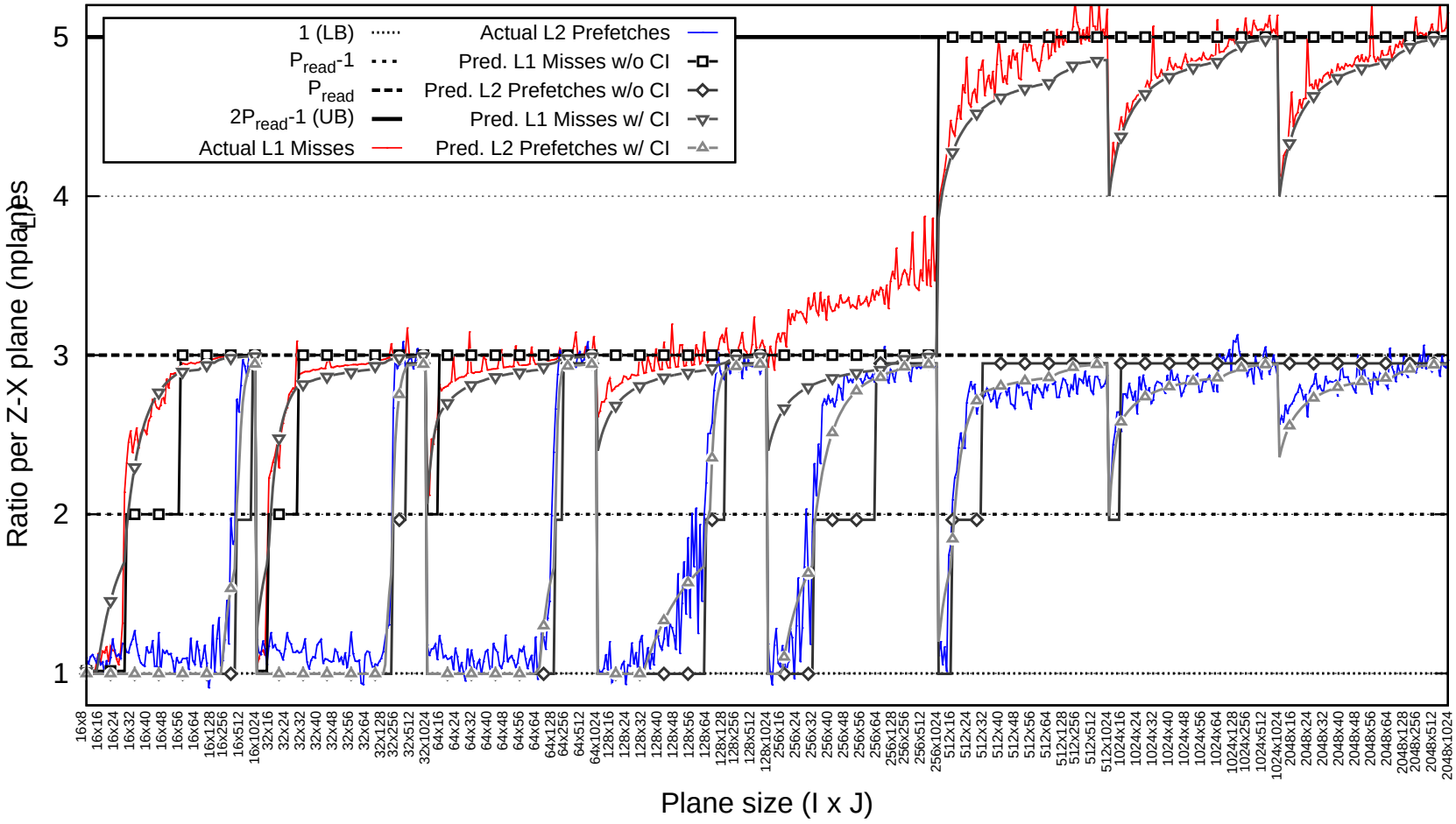


Overview

Stencil Model: Cache Interference Phenomena

- Add full 3C (compulsory, conflict and capacity) misses detection
- Convert the model from a discrete to a continuum space

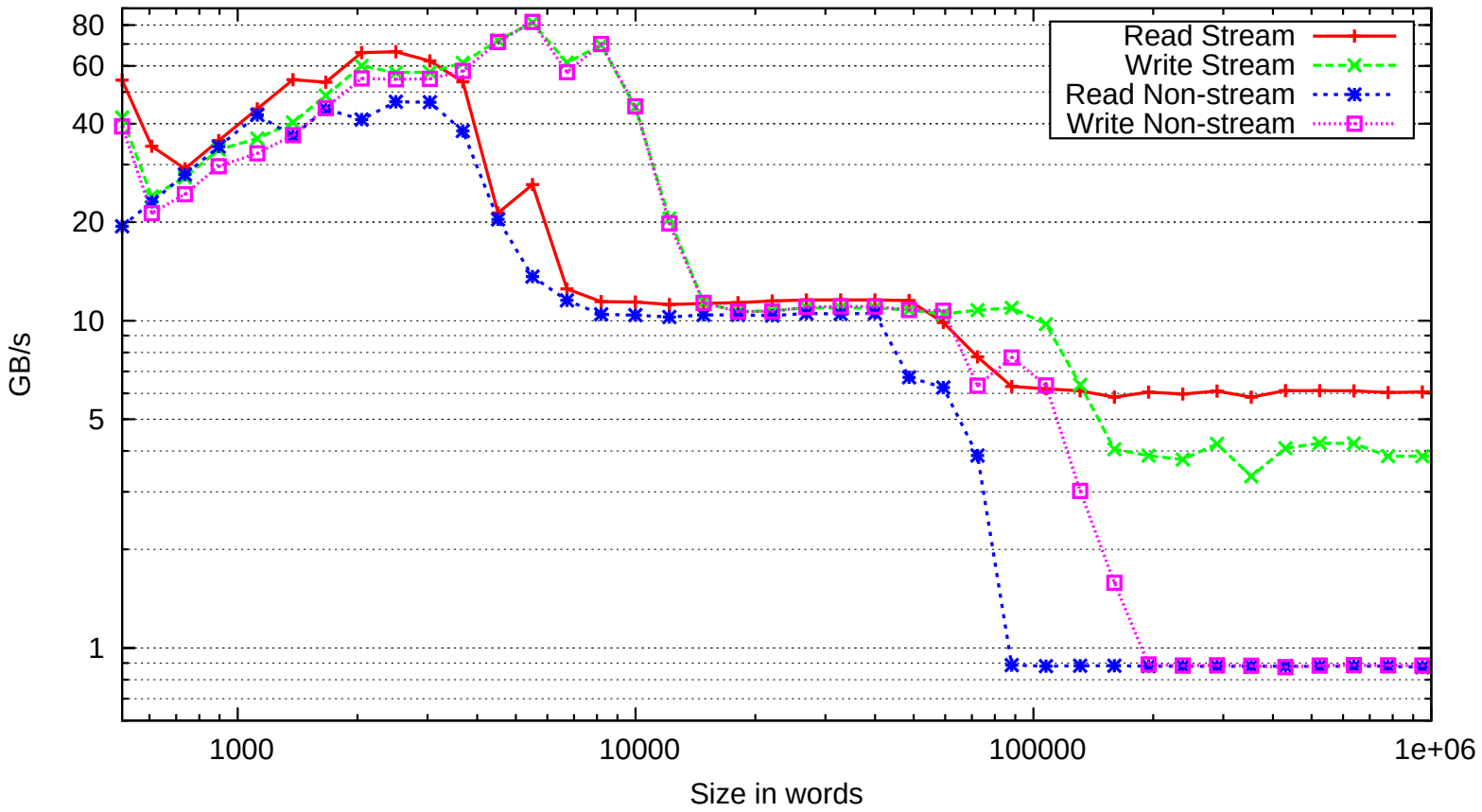
$$P(i) = \frac{\overbrace{II \times JJ}^{\text{whole plane}} - \overbrace{II \times (P_{read} - 1)}^{\text{columns reuse}}}{II \times JJ} \in [0,1] \rightarrow nplanes_{Li}\prime = nplanes_{Li} \times P(i)$$



Stencil Model: from Single-core to Multi-core

STREAM2 benchmark must mimic stencil environment:

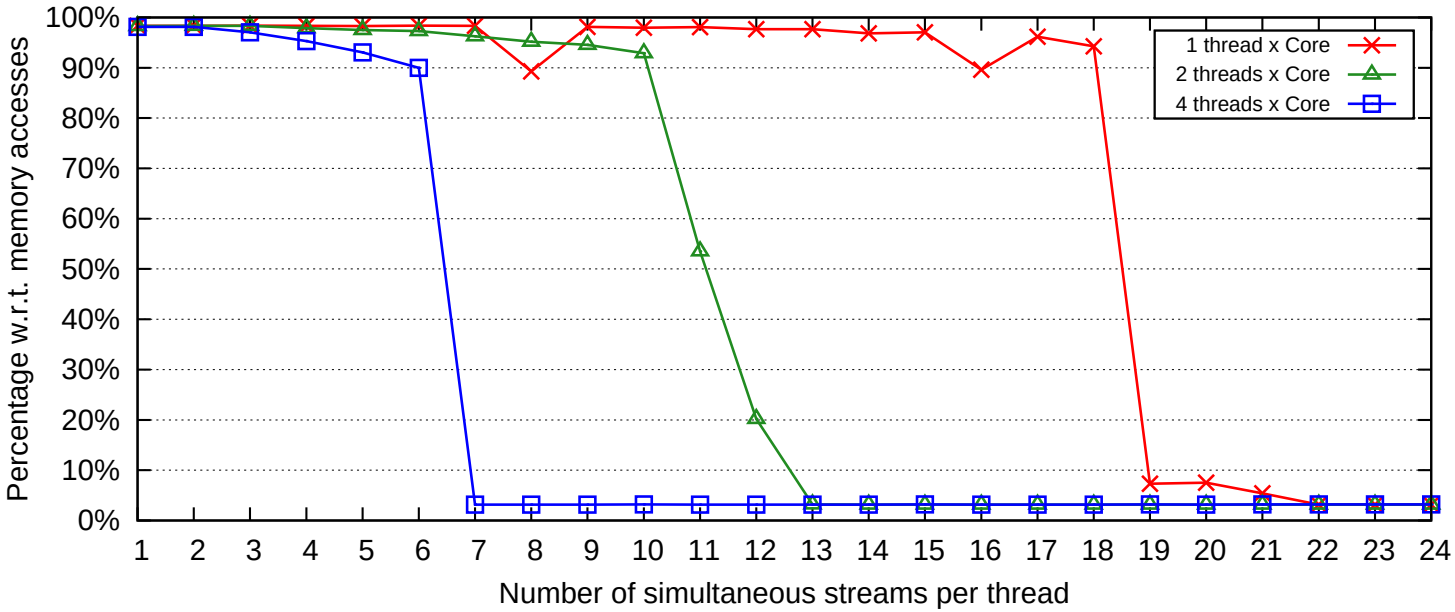
- Read Bw (DOT kernel), Write Bw (FILL kernel)
- Number of threads and their pinning to cores
- Memory alignment, temporal or non-temporal writes, SISD/SIMD ISA
- Shared resources $\rightarrow size_{Li} = N/nthreads_{core}$



Stencil Model: Prefetching Effect

Gabriel Marin, Collin McCurdy, and Jeffrey S. Vetter.
Diagnosis and optimization of application prefetching performance. (ICS '13)

$$DC_{effectiveness} = DC\_Req\_PF / DC\_Req\_All \in [0, 1]$$



$$nplanes_{Li}^{S} = nplanes_{Li} \times DC_{effectiveness}$$

$$nplanes_{Li}^{NS} = nplanes_{Li} \times (1 - DC_{effectiveness})$$

Stencil Model: Spatial Blocking Optimization

Traverse the domain in $TI \times TJ \times TK$ blocks:

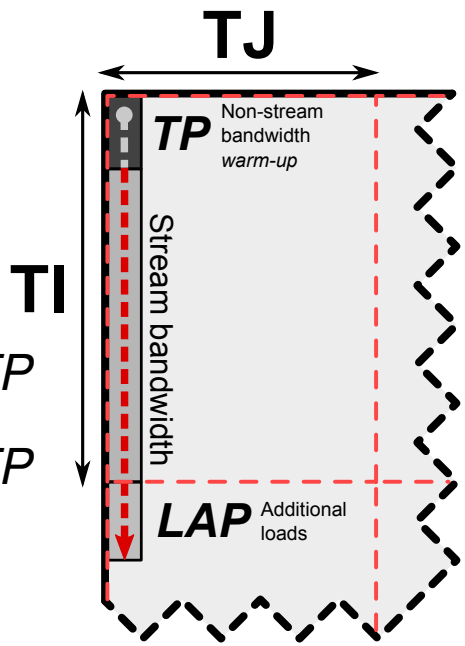
$$I = \lceil TI/W \rceil \times W, \qquad J = TJ, \qquad K = TK,$$
$$II = \lceil (TI + 2 \times \ell)/W \rceil \times W, \quad JJ = TJ + 2 \times \ell, \quad KK = TK + 2 \times \ell\,.$$

New II and JJ are used for $R_{1,2,3,4}$ to estimate $nplanes_{Li}$:

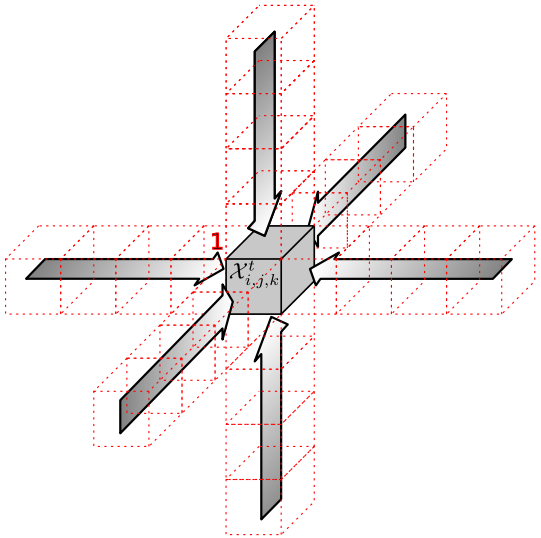
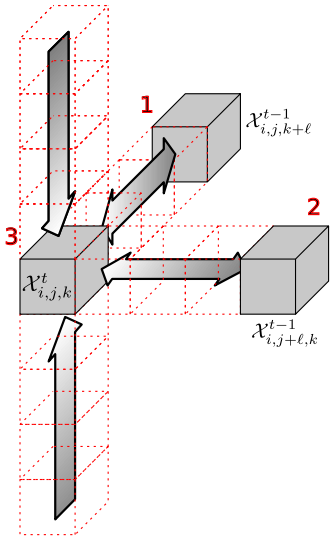
$$Misses_{Li}^{[S,NS]} = \lceil II/W \rceil \times JJ \times KK \times nplanes_{Li}^{[S,NS]} \times NB\,,$$

Consider penalties on caches with prefetching (may disrupt memory access and increase data transfers):

$$Misses_{Li}^{NS} \stackrel{+}{=} TP \times JJ \times KK \times nplanes_{Li}^{NS} \times NB, \quad \text{if } II/W \geq TP$$
$$Misses_{Li}^{S} \stackrel{+}{=} LAP \times JJ \times KK \times nplanes_{Li}^{S} \times NB, \quad \text{if } II/W \geq TP$$



Stencil Model: Semi-stencil Optimization

Dataset accesses	Classical Stencil	Partial *Semi-stencil* (X and Y axis)
Structure of the stencil		
$\mathcal{X}^{t-1}$ loads	$2 * (dim - 1) * \ell + 1$	$(dim - 1) * \ell + 1$
$\mathcal{X}^t$ loads	0	$dim - 1$
$\mathcal{X}^t$ stores	1	dim

Semi-stencil can be modeled by setting the Z-X planes read/written

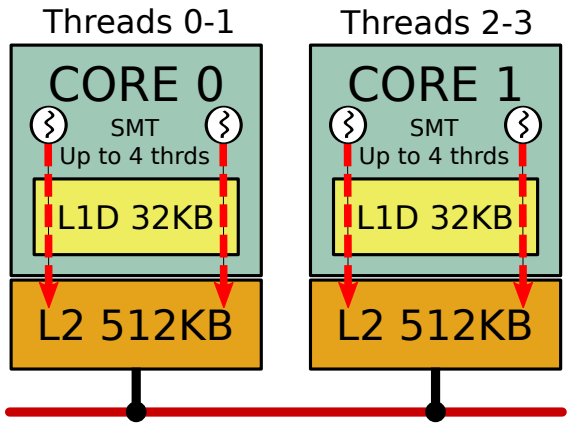
$$P_{read} = \overbrace{\ell + 1}^{\mathcal{X}^{t-1}} + \overbrace{1}^{\mathcal{X}^t}, \quad P_{write} = \overbrace{2}^{\mathcal{X}^t}, \quad \text{if } \neg R_4$$

$$P_{read} = \ell + 1 + 2, \quad P_{write} = 3, \quad \text{if } R_4$$

Overview

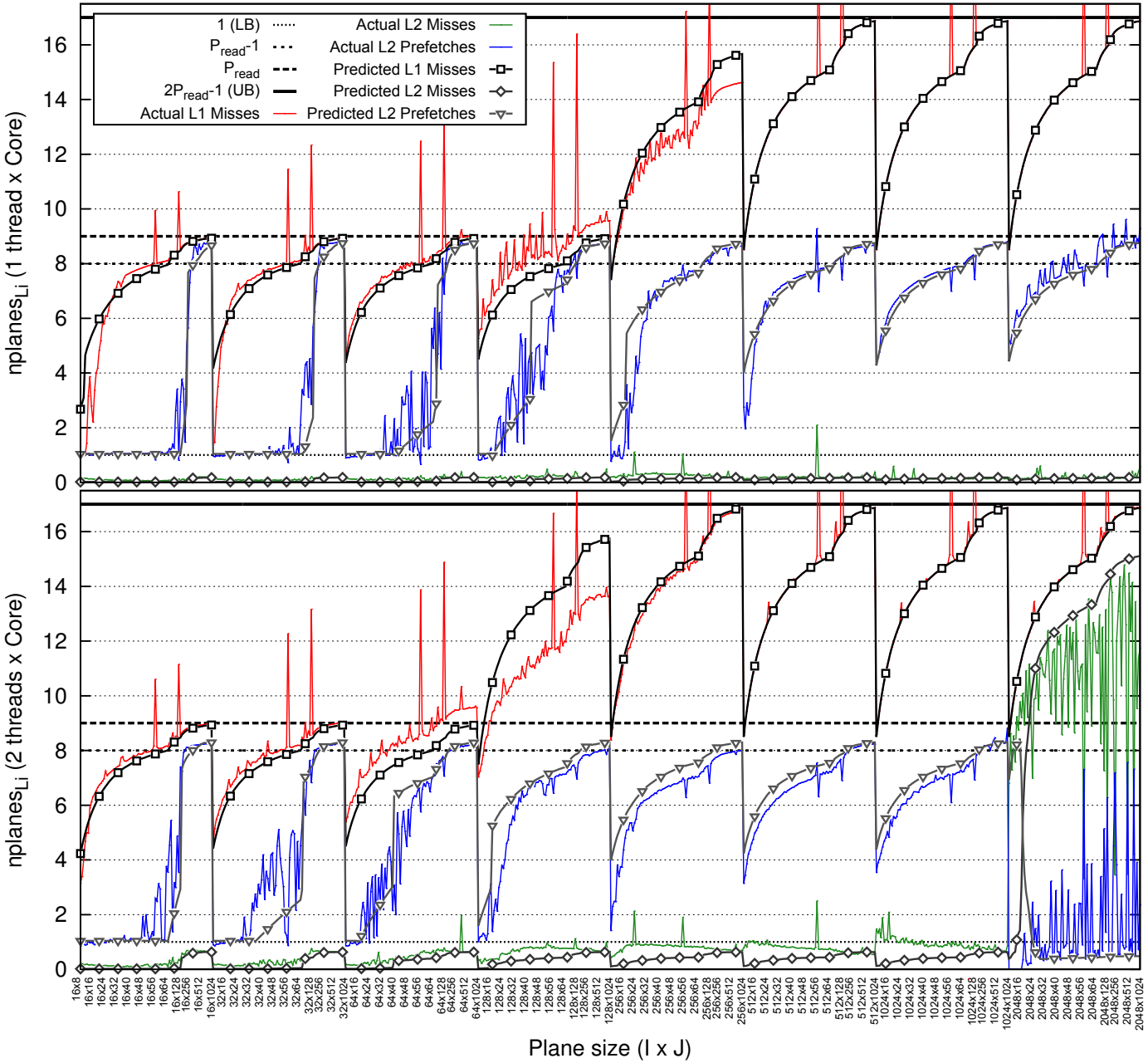
Model Validation: Experiments on Intel Xeon Phi

Parameters	Range of values
Naive sizes ($I \times J \times K$)	$8 \times 8 \times 128 \ldots 2048 \times 1024 \times 128$
Rivera sizes ($I \times J \times K$)	$512 \times 2048 \times 128$
Stencil sizes (ℓ)	1, 2, 4 and 7 (7, 13, 25 and 43-point)
Algorithms	{Naive, Rivera} $\times$ {Classical, *Semi-stencil*}
Block sizes (*TI* and *TJ*)	{8, 16, 24, 32, 64, 128, 256, 512, 1024, 1536, 2048}
SMT configuration	1, 2 and 4 threads x Core

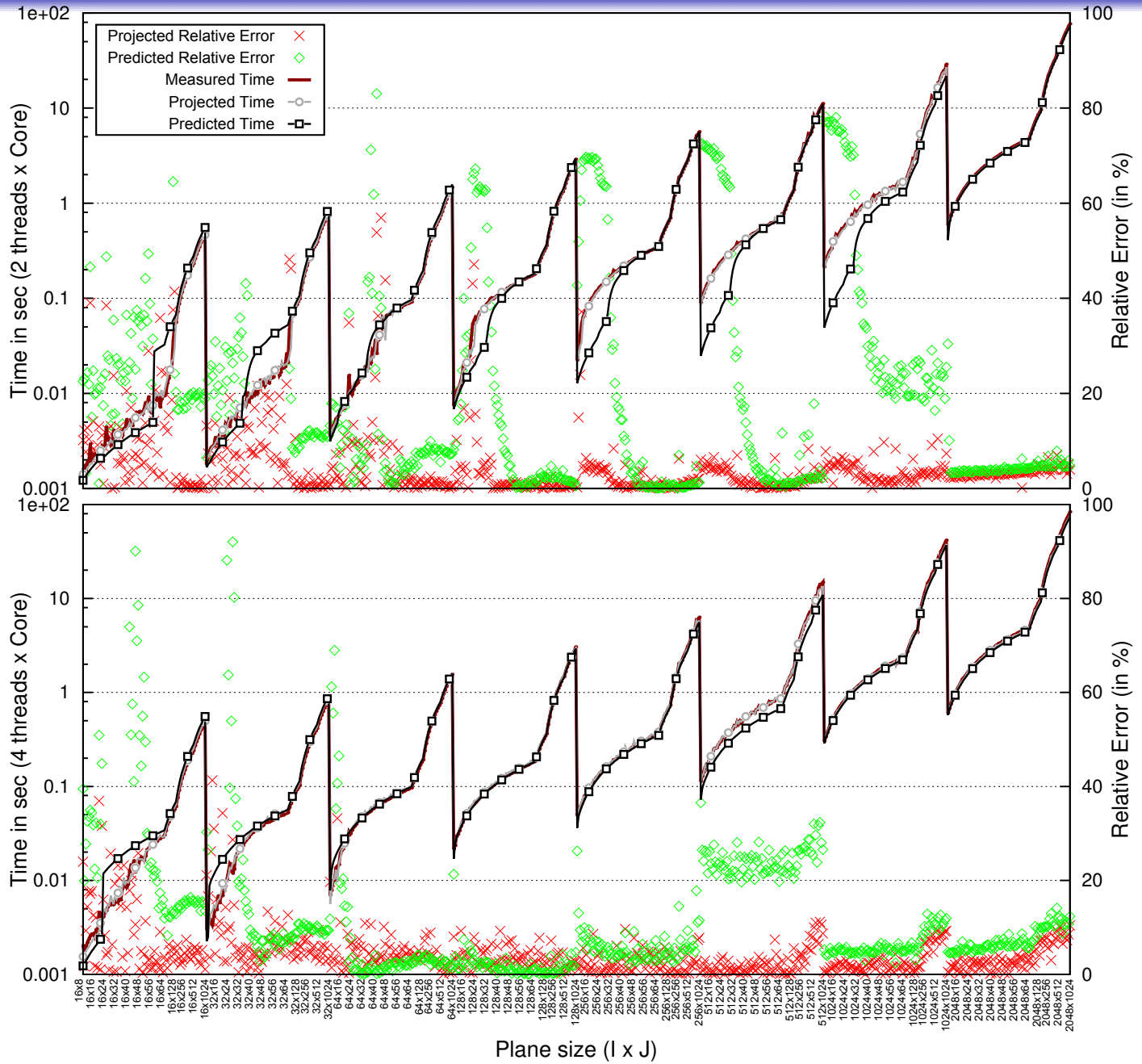


Description	Intel Xeon Phi Events	Time Cost Formulas
Cycles	CPU_CLK_UNHALTED	$T_{L1} =$ (L1 Reads - L1 Misses) $\times Bw_{L1}^{cline}$
L1 Reads	VPU_DATA_READ	$T_{L2} = Bw_{L2}^{cline} \times$ (L1 Misses - L2 Misses -
L1 Misses	VPU_DATA_READ_MISS	(L2 Prefetches - L2 Writes $\times$ Pref Eff))
L2 Misses	L2_DATA_READ_MISS_ MEM_FILL	$T_{Mem} =$ L2 Misses $\times Bw_{Mem}^{NS} + Bw_{Mem}^{S} \times$ (L2 Prefetches - L2 Writes $\times$ Pref Eff)
L2 Prefetches	HWP_L2MISS	$T_{Writes} =$ L2 Writes $\times$ Pref Eff $\times Bw_{Write}^{S}$
L2 Writes	L2_WRITE_HIT	+ L2 Writes $\times$ (1 - Pref Eff) $\times Bw_{Write}^{NS}$

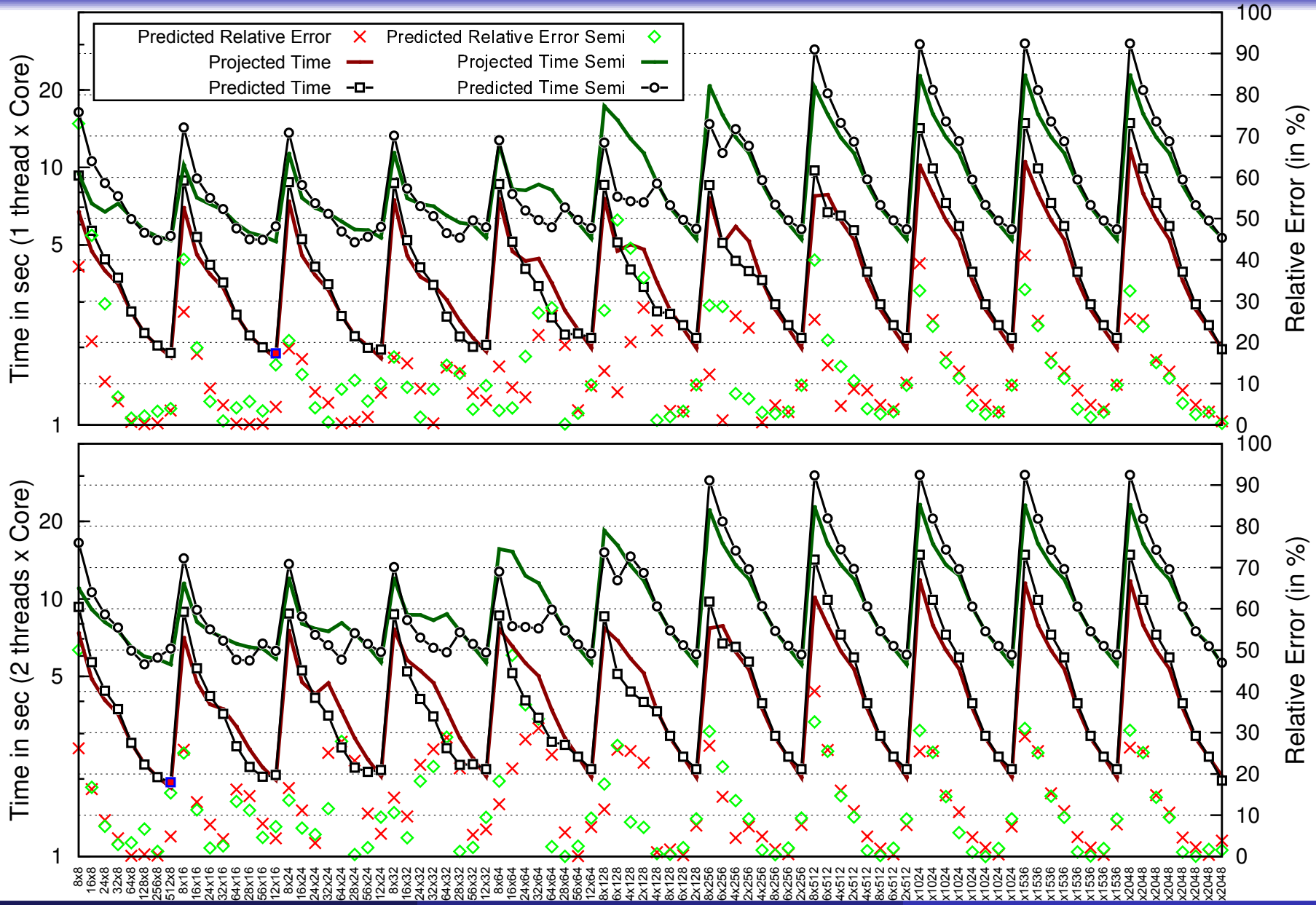
Model Validation: Cache Misses (Naive case, $\ell = 4$)



Model Validation: Time Prediction (Naive case, $\ell = 7$)

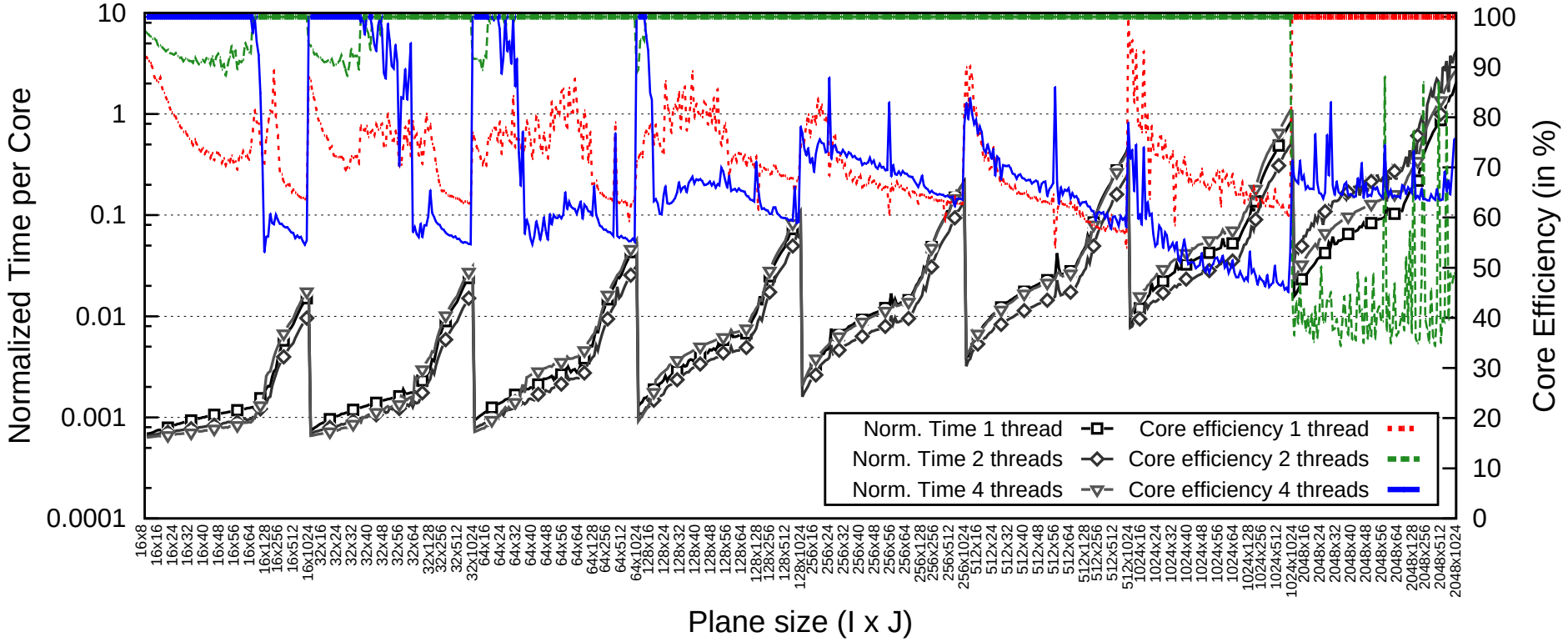


Model Validation: Spatial Blocking & Semi-stencil ($\ell = 1$, $\ell = 7$)



Model Validation: Core Efficiency in SMT Mode

$$Core^{SMT_i}_{efficiency} = \frac{\min(\tau^{SMT_1}, \ldots, \tau^{SMT_n})}{\tau^{SMT_i}} \in [0, 1]$$



- Decide best SMT configuration to be conducted
- Predict cache and prefetching contention (due to ℓ and $II \times JJ$)

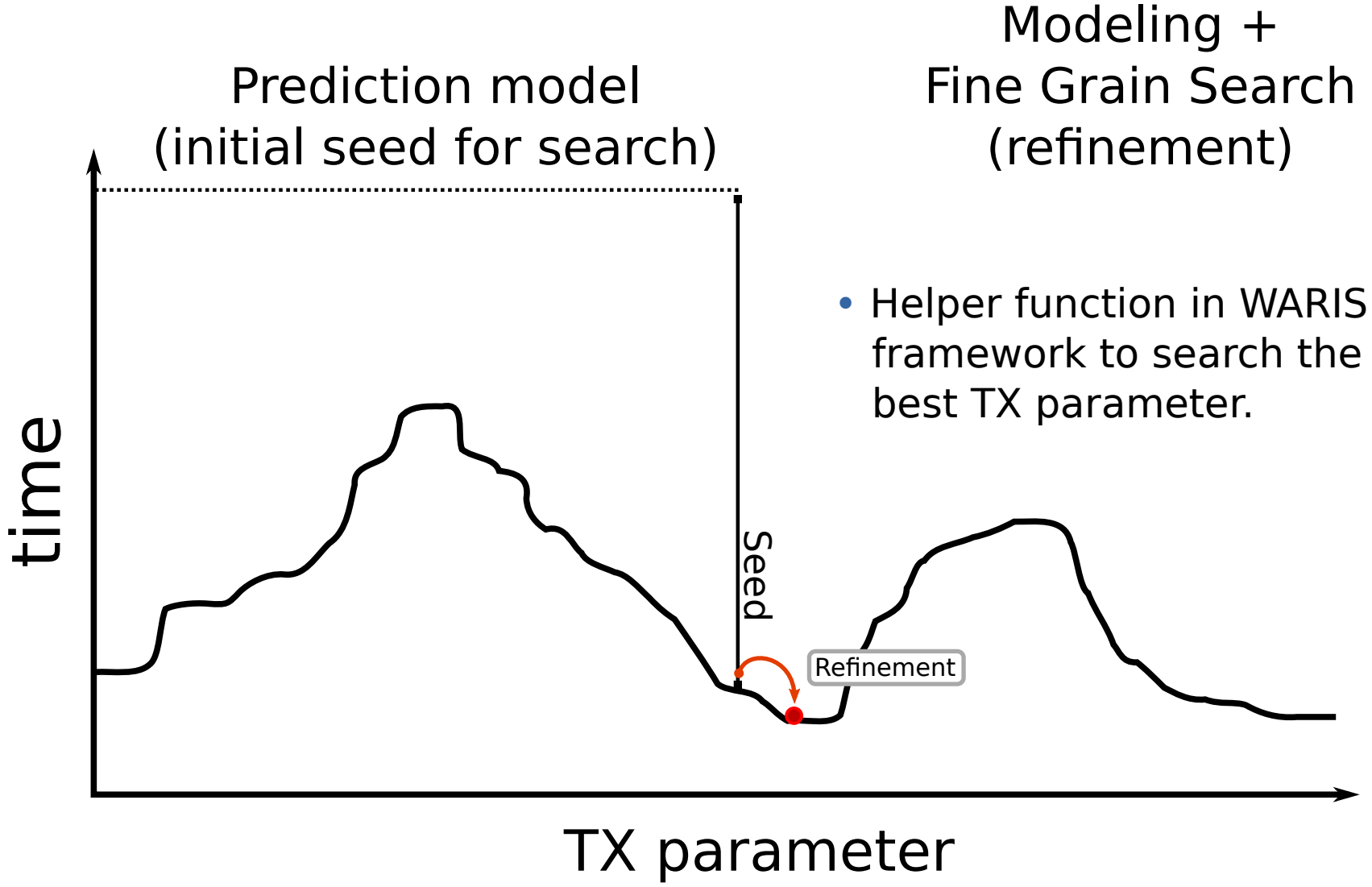
Conclusions & Future Work

- This model includes multi- and many-core support
- Robust hardware prefetching modeling (*prefetching effectiveness*)
- Considers cache interference phenomena (3C misses)
- Extended with stencil optimizations (blocking and Semi)
- Most results have a high accuracy (rel. error 5-15%)
- Core efficiency predictor (best SMT configuration)

Questions

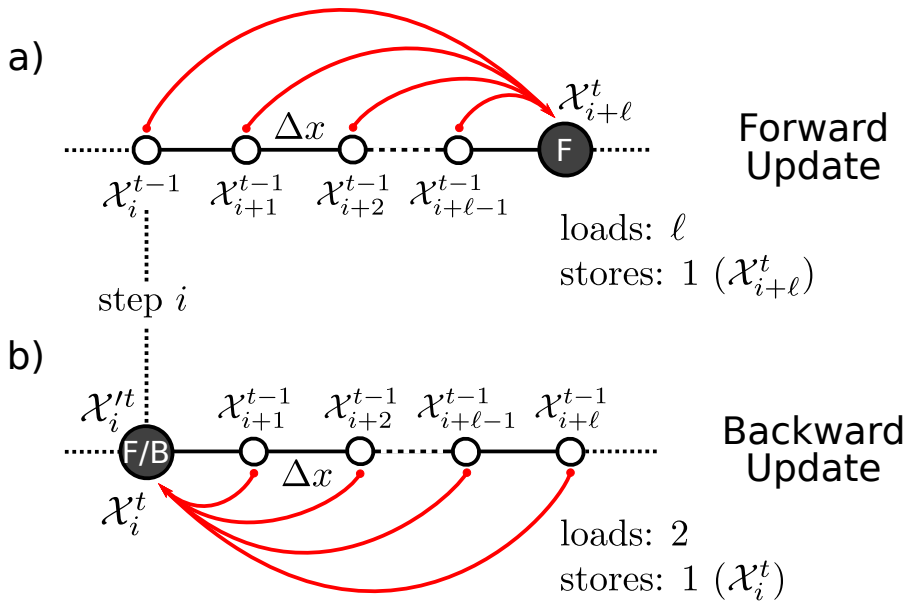

Thank you for your attention!
Copyright 2013. Barcelona Supercomputing Center - BSC

Auto-tuning blocking parameters

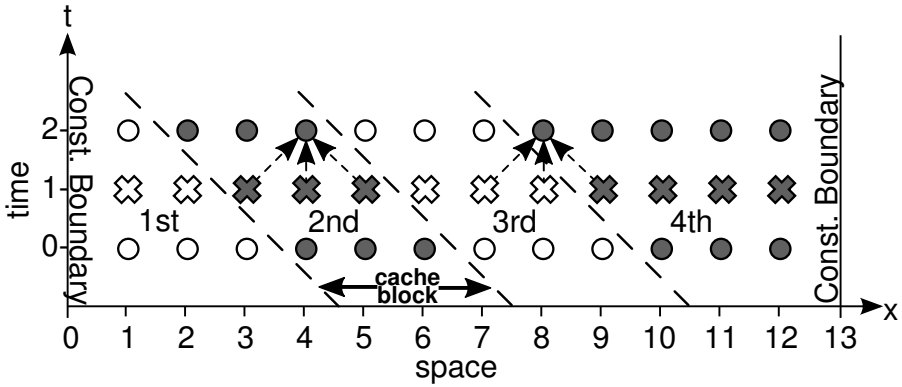


- Helper function in WARIS framework to search the best TX parameter.

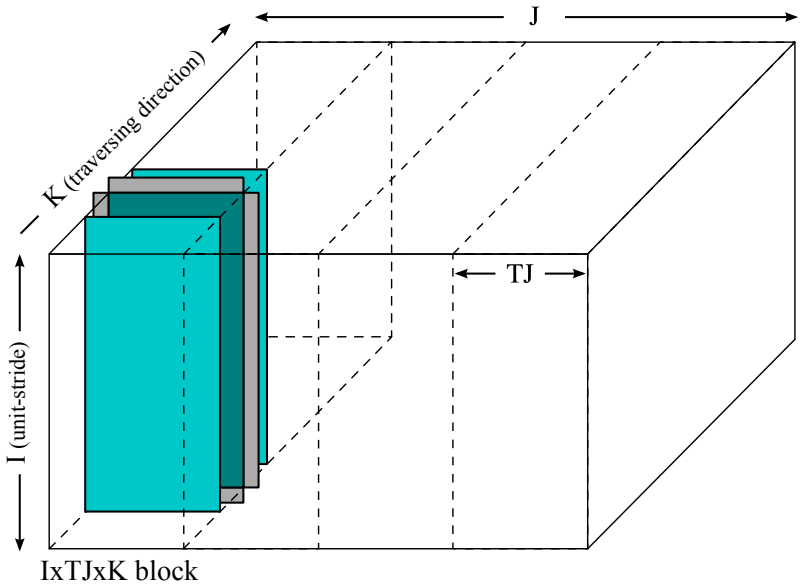
State of the Art - The most popular



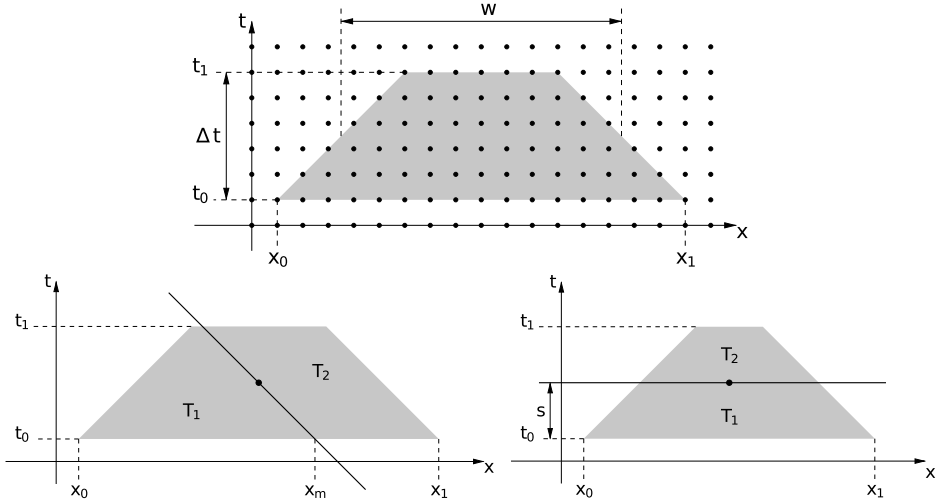
Semi-stencil



Time-skewing



Rivera



Cache-oblivious

Stencil Model: Read Miss Cases & Rules