Evaluating Autotuning Heuristics for Loop Tiling

Tomoya Yuki[1], Yukinori Sato[1], Toshio Endo[1]
[1]Tokyo Institute of Technology

Background

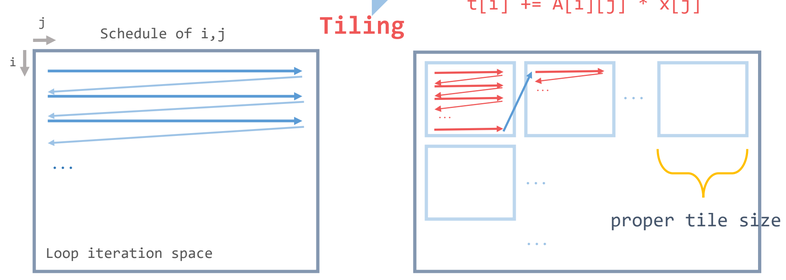
Loop Tiling is a technique that uses cache hierarchy efficiently

- It improves data locality of reference in nested loops.
- Tile sizes need to be properly adjusted in order to acquire better performance
- Polyhedral compilers enable to automate the loop transformation

```
for(i=0; i<IMAX; i++)
  for(j=0; j<JMAX; j++)
    t[i] += A[i][j] * x[j]
```

Tiling

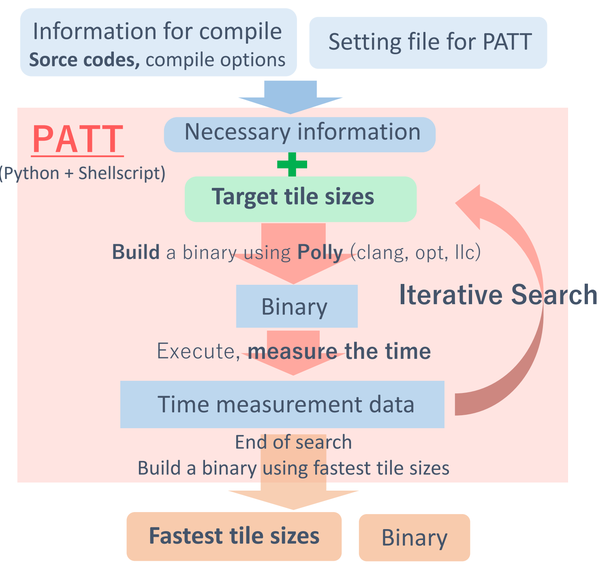
```
for(ii=0; ii<IMAX; ii+=TI)
  for(jj=0; jj<JMAX; jj+=TJ)
    for(i=ii; i<min(ii+TI,IMAX); i++)
      for(j=jj; j<min(jj+TJ,JMAX); j++)
        t[i] += A[i][j] * x[j]
```



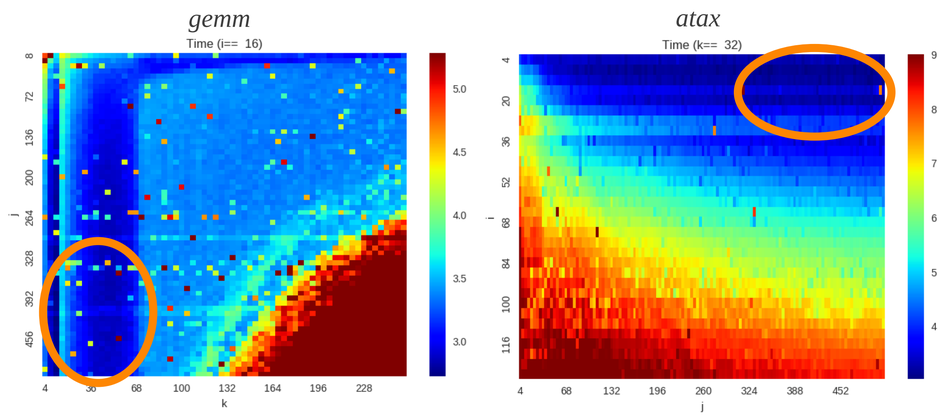
Improve data locality of reference

We developed **PATT** to search proper tile sizes
(Polyhedral compilation based AuTo Tile size optimizer)

- Using Polly [1] as a Polyhedral compiler
- Iterative building binary & measurement
- Implemented some heuristics



Heatmap that presents performance for each tile size (blue is fast)



Motivation

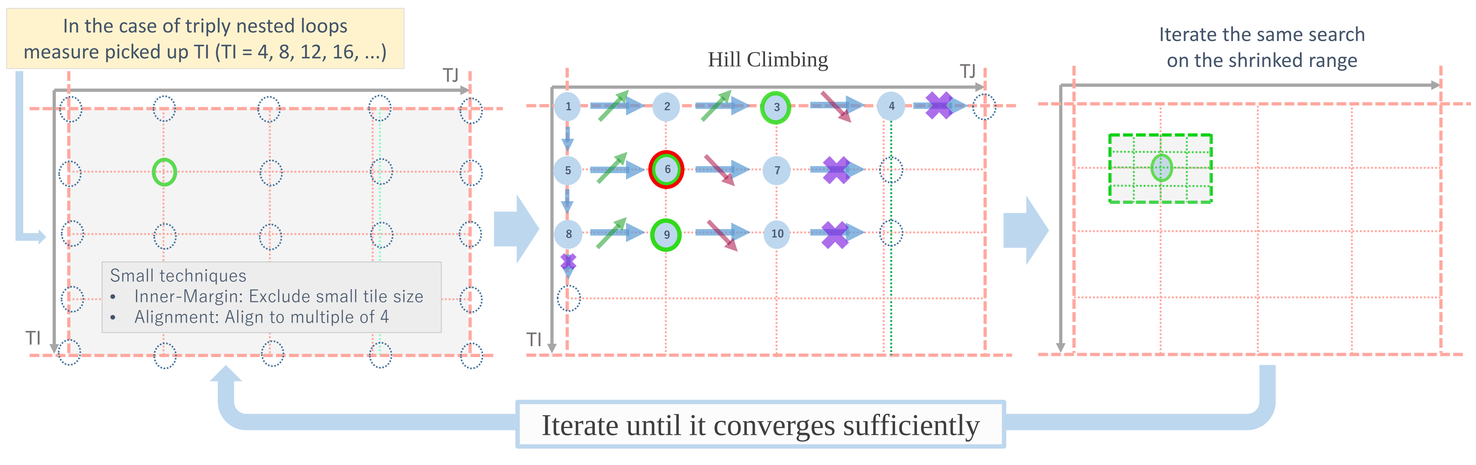
- Investigate the best tile size parameters for performance
  - Need to search from a huge parameter space
  - FAST convergence speed and HIGH accuracy
- Many core processors such as XeonPhi (KNL) are becoming popular

How do we achieve?
Which heuristic is the best?

Heuristics

We have developed **I-PATT** heuristic which is a specialized one for tile size adjustment

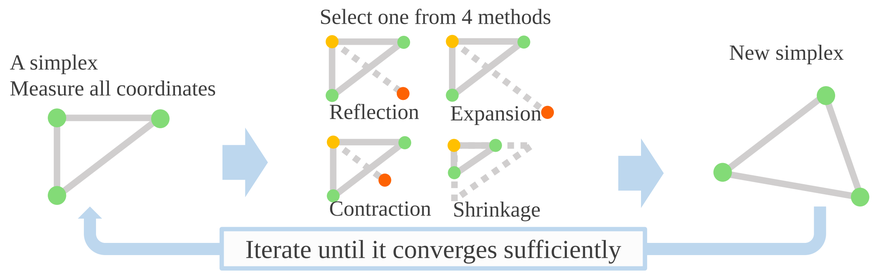
- Load balance is important in the case of many core processors, which is the dominant factor for performance
  - In the case of triply nested loops
    - First, fix TJ and TK (inner loop tile sizes) and find fastest TI (outer most loop tile size)
    - Then, search TJ and TK using the algorithm below
  - In the case of doubly nested loops
    - Search TI and TJ using the algorithm below
- 2-dimensional search algorithm: Starting from wide and coarse search space, it gradually focuses on a finer region → In order to be over flat heatmap region and fast convergence
- We adopted hill climbing technique to reduce search steps



Nelder-Mead Simplex (NM) is one of meta hauristics to find minimum

- Using the concept of "Simplex", which consists of problem dimension +1 coordinates (e.g. 2-dimension: 3 coordinates: means an triangle)
- Movivg a coordinate of a simplex, gradually reach the minimum

Ported from Orio [2]

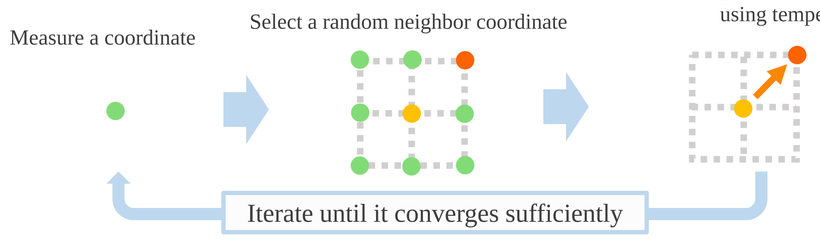


Simulated Annealing (SA) is also one of well-known meta hauristics

- Using the concept of "Temperature", which affects the probability below
- Selecting a random coordinate from neighbor
- Comparing the results of measurement, probabilistically moving the new coordinate

Ported from Orio [2]

Compare old one and new one
old < new : move to new one
old > new : probabilistically move using temperature

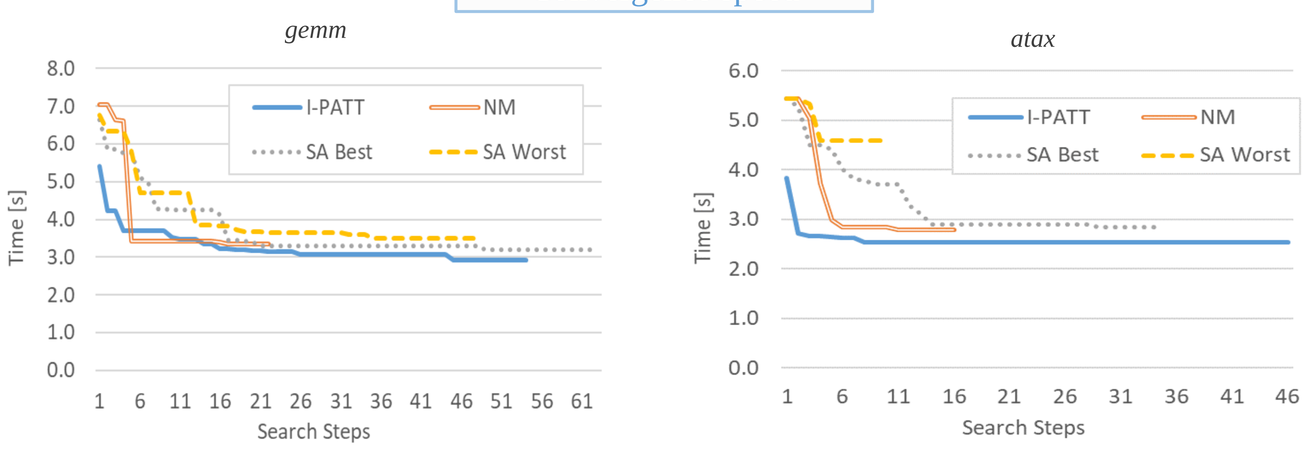


Evaluation

- We use **Polybench [3]** as a benchmark suite, which is familiar with Polly
- This time we pick up *gemm* as a triply nested loop kernel and *atax* as a doubly one
- A few user setting values of each heuristics are optimized based on the result of our preliminary experiments

Environment	
Processor	Xeon Phi Processor 7210
Num of threads	64
Problem size	LARGE (Polybench)
Polly Optimization	Tiling, Vectorization, Parallelization

Convergence speed



Accuracy

gemm

Heuristics	Tile Sizes	Time	%
Brute-Force	16, 1124, 12	2.89	100.0
I-PATT	16, 1192, 12	2.91	99.3
NM	16, 36, 48	3.36	86.0
SA (worst)	16, 28, 44	3.50	82.6
SA (best)	16, 24, 80	3.19	90.6

atax

Heuristics	Tile Sizes	Time	%
Brute-Force	8, 1940	2.50	100.0
I-PATT	8, 2096	2.53	98.8
NM	8, 80	2.78	89.9
SA (worst)	36, 36	4.60	54.3
SA (best)	8, 64	2.84	88.0

"Brute-Force" is an exhaustive experiment
We can see that the tile sizes of I-PATT is nearby the one of Brute-Force

- I-PATT achieves FASETR convergence speed and HIGH accuracy
- NM shows fast convergence, but low accuracy
- SA shows slow convergence and low accuracy. It is also sensitive to the random values that decide probabilistic behaviors (SA best vs. SA worst)
- I-PATT reaches almost 99% of the fastest performance of Brute-force while NM and SA remain at most 90% performance

Reasons for the low accuracy of NM and SA

- First, the outer-most tile size tends to be sensitive for performance than the other dimensions. However, NM and SA do not consider this within their search algorithms.
- Second, local search within neighboring inner loop tile size is less sensitive for performance. NM and SA lack wider and coarser search.

[1] Tobias Grosser, Armin Groesslinger, and Christian Lengauer. 2012. Polly - Performing polyhedral optimizations on a low-level intermediate representation. Parallel Processing Letters 22, 04 (2012), 1-28.
[2] A. Hartono, B. Norris, and P. Sadayappan. 2009. Annotation-based empirical performance tuning using Orio. In 2009 IEEE International Symposium on Parallel Distributed Processing. 1-11.
[3] Tomofumi Yuki and Louis-No el Pouchet. 2016. PolyBench. https://sourceforge.net/projects/polybench. (2016).

This work is supported by JST-CREST, "Software Technology that Deals with Deeper Memory Hierarchy in Post-petascale Era"

Tokyo Institute of Technology