On-chip shared caches, data-parallel programming, and the relevance of data layout optimizations

J. (Ram) Ramanujam
Louisiana State University

Joint work with

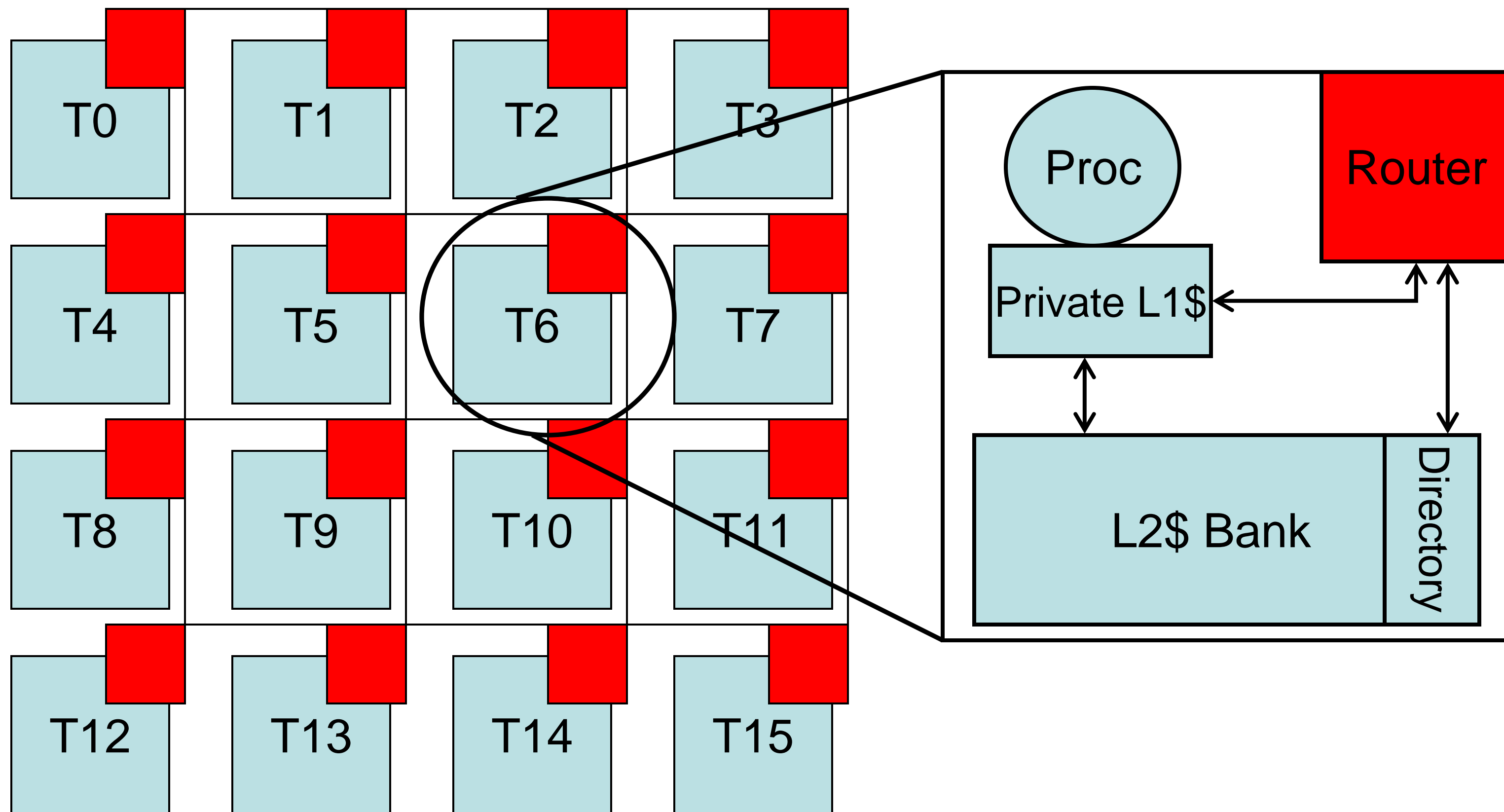
Q. Lu, U. Bondhugula, S.Krishnamoorthy, P. Sadayappan
The Ohio State University

Y. Chen, H. Lin, T. Ngai
Intel

Introduction

- Chip Multiprocessors
  - Nun-Uniform Cache Architecture (NUCA) for shared on-chip caches for tiled CMPs
  - Cache banks connected via an on-chip network
  - Address space is block-cyclically mapped across the on-chip cache banks with a certain block size
  - We have studied cache-line interleaving
- Data-parallel programming
- Mismatch between the cache architecture and the canonical row-major/column-major layout of arrays
- One Solution: Find non-canonical array layouts that match iteration mapping

A tiled CMP architecture



Example: Summation

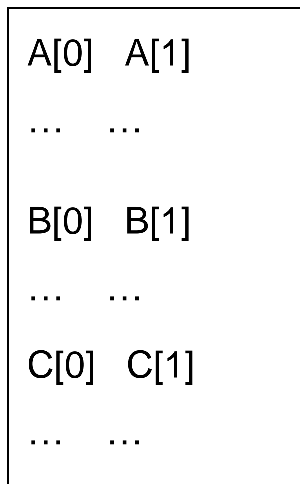
```
Array A(N+1),B(N+2),C(N);

C = shift(A(0:N),{1}) +
     shift(B(0:N),{2});
```

```
for (i = 0; i < N; i++)
    C[i] = A[i+1] + B[i+2];
```

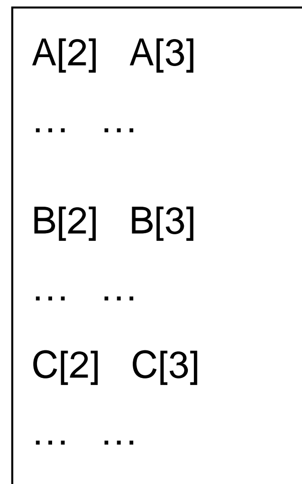
Example: Summation

Thread 0



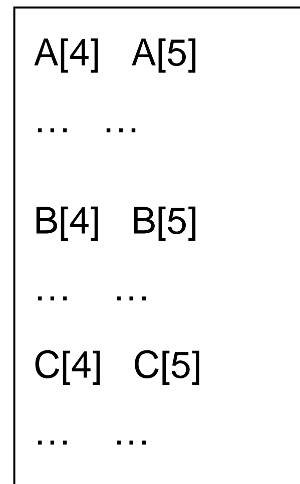
Bank 0

Thread 1



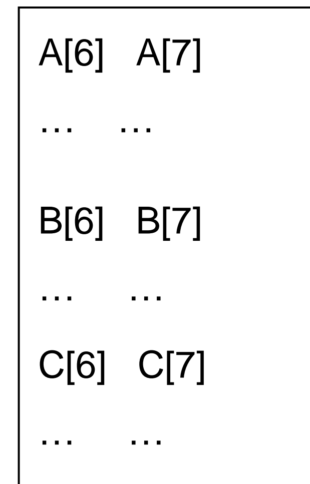
Bank 1

Thread 2



Bank 2

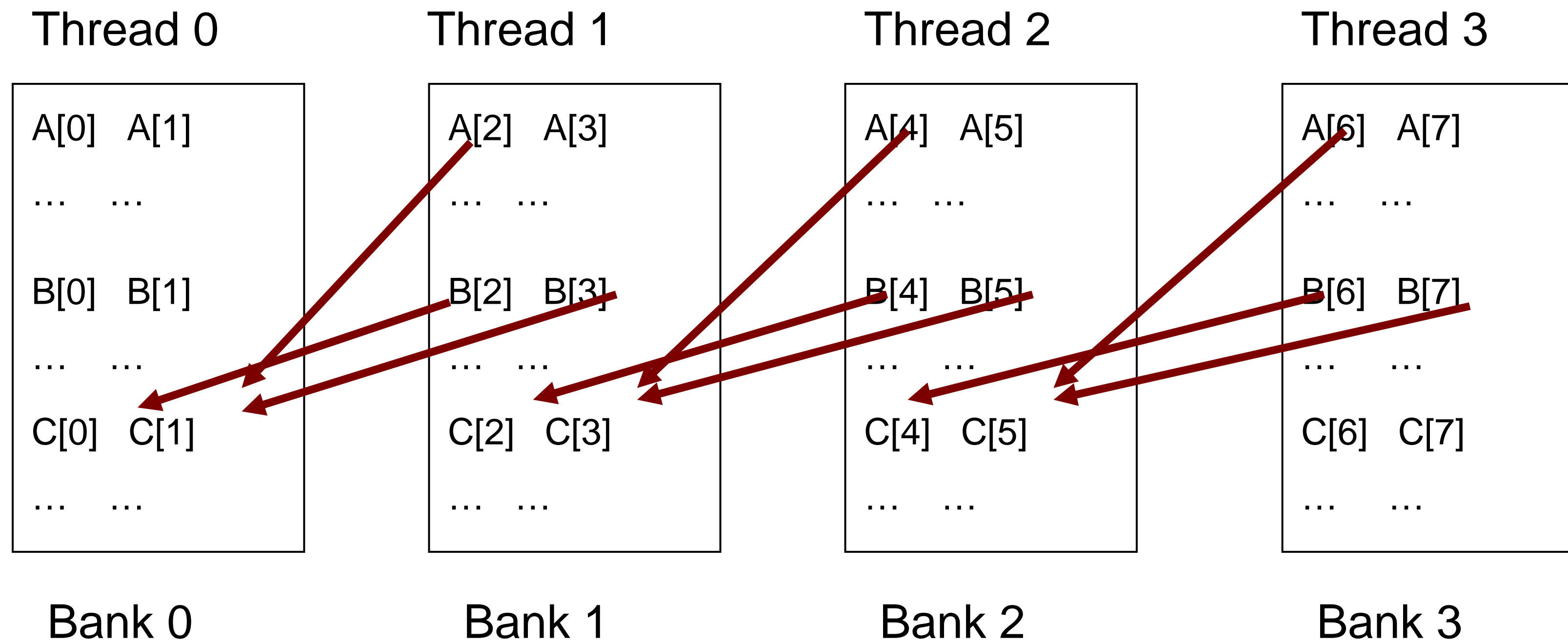
Thread 3



Bank 3

We use line size = 2 elements for all examples; also use line interleaving

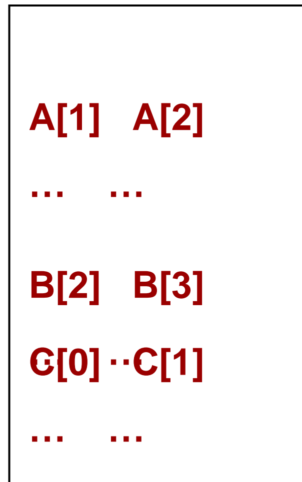
Example: Summation



Non-local bank accesses

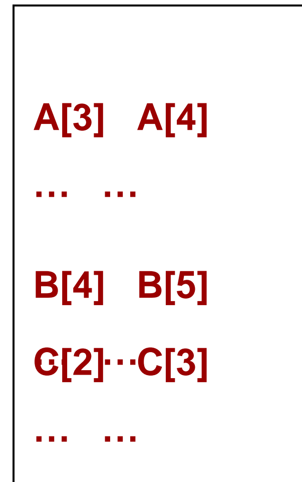
Example: Summation

Thread 0



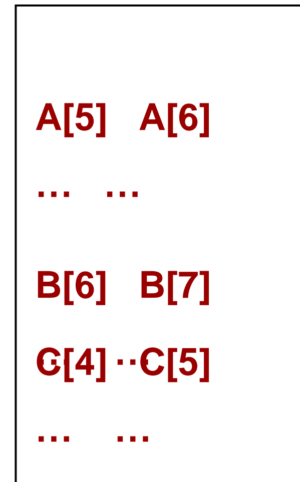
Bank 0

Thread 1



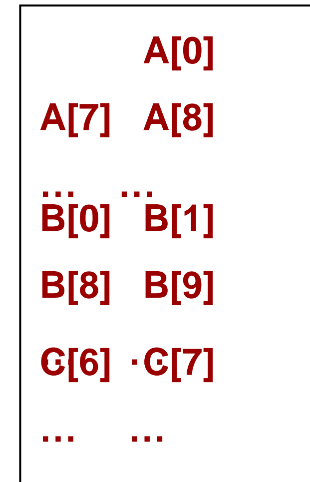
Bank 1

Thread 2



Bank 2

Thread 3



Bank 3

Only local bank accesses after data alignment

Example: 1-D Jacobi

```
Array A(N), B(N);
while (condition) {
   B(1:N-1)=A(1:N-1)+ shift(A(2:N),{-1})+
                      shift(A(0:N-2),{1});
   A(1:N-1)=B(1:N-1);
}
```

```
while (condition) {
   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];
}
```

Example: 1-D Jacobi

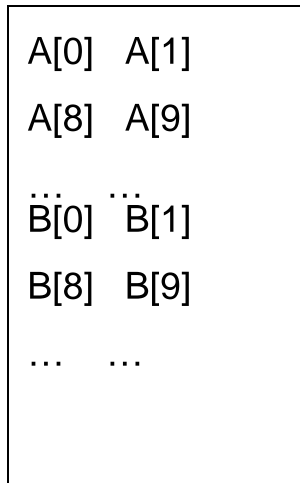
- Completely localized bank access is impossible for any balanced mapping of iterations to processors

- For every outer time loop iteration, the total inter-bank transfer volume is of the same order as the total number of inner loop iterations

1-D Jacobi

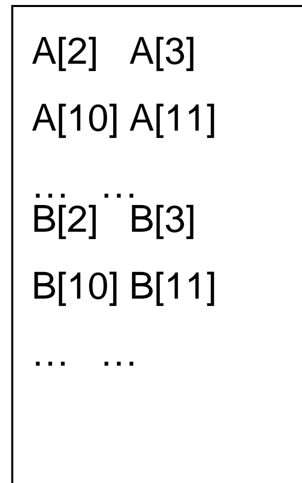
OpenMP static scheduling (chunk size=2)

Thread 0



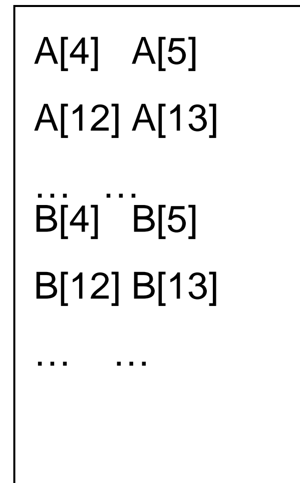
Bank 0

Thread 1



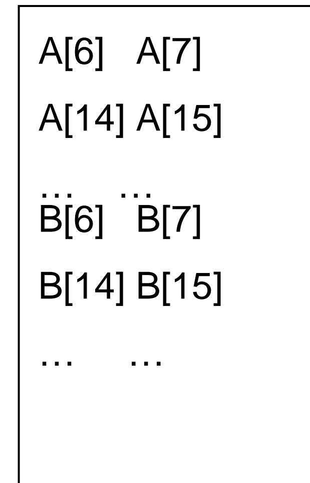
Bank 1

Thread 2



Bank 2

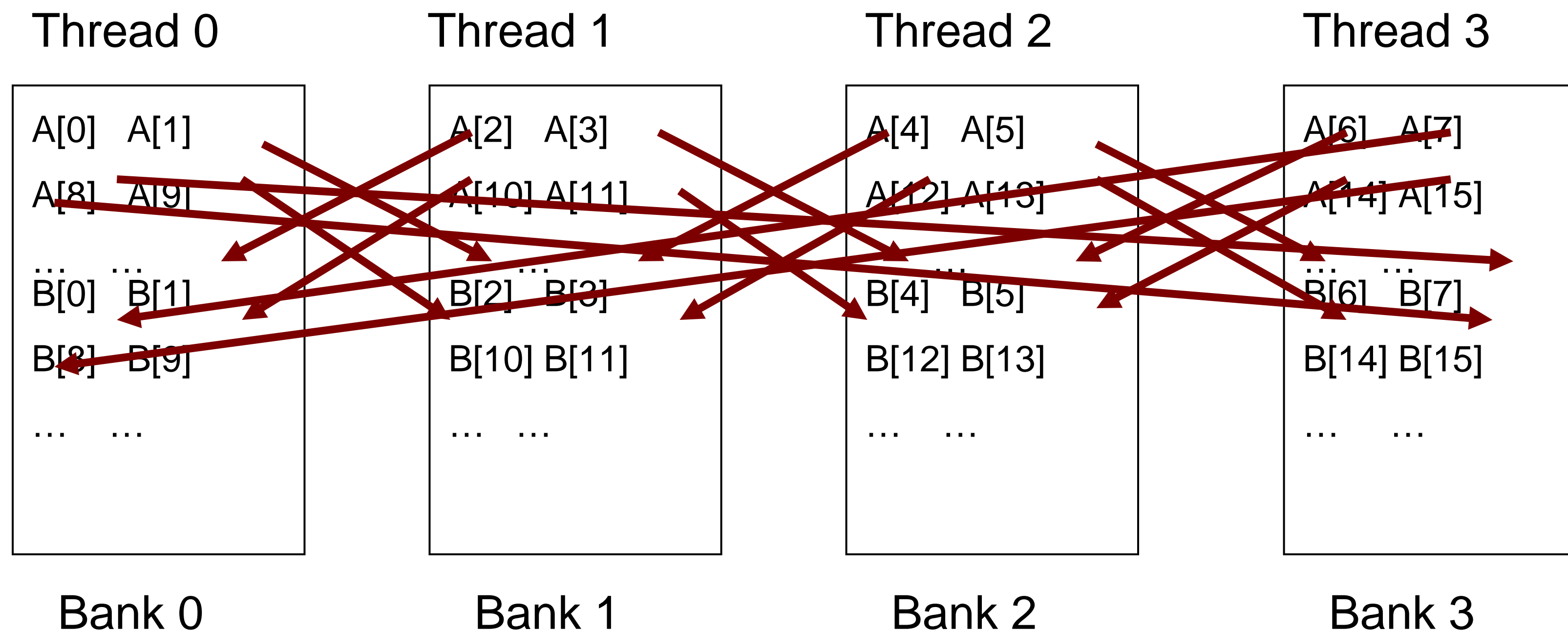
Thread 3



Bank 3

1-D Jacobi

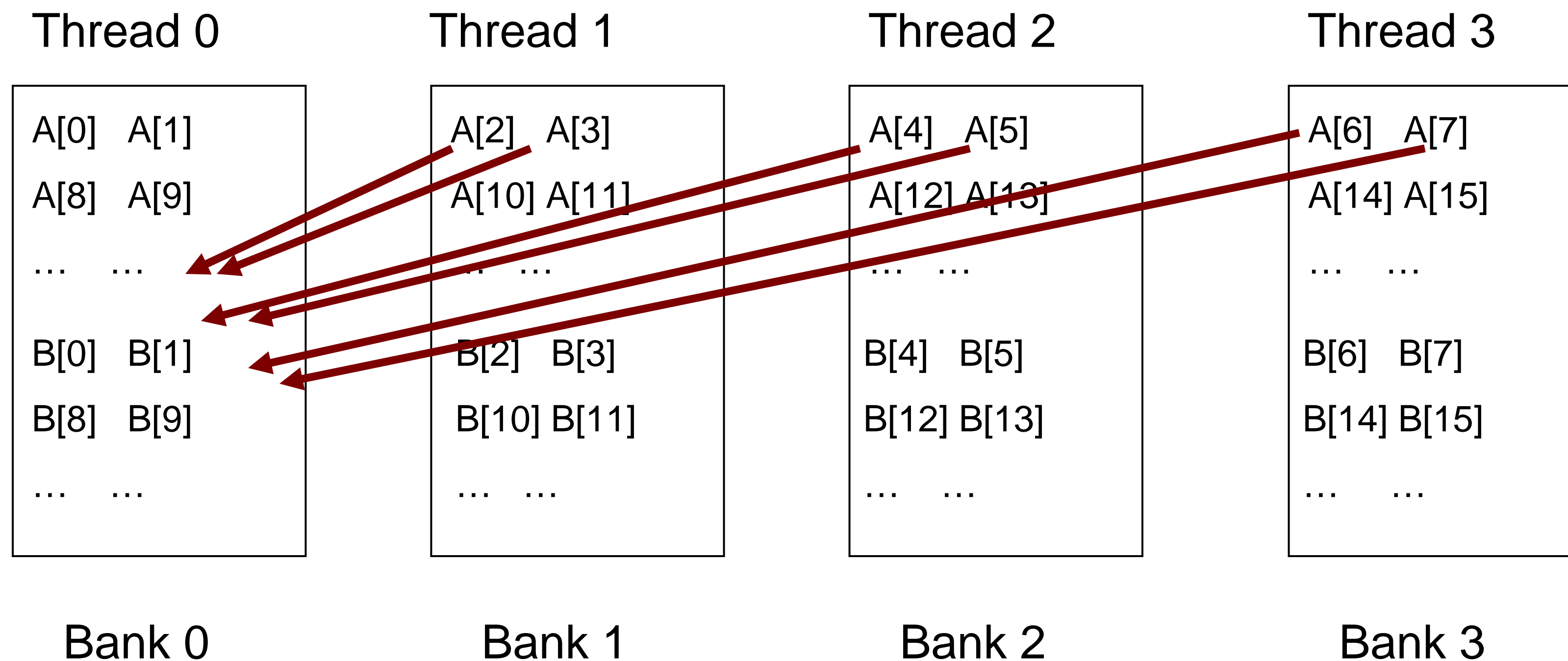
OpenMP static scheduling (chunk size=2)



Lots of non-local bank accesses

1-D Jacobi: block scheduling

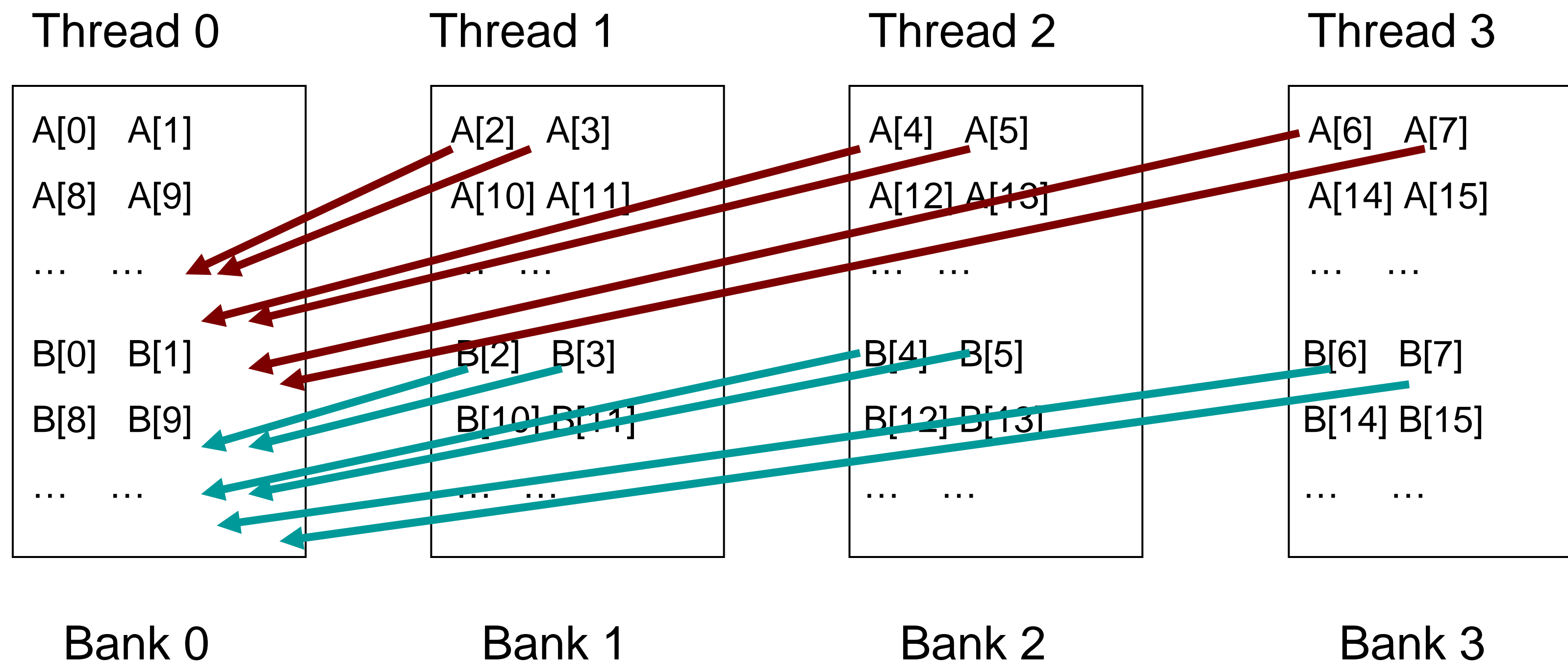
OpenMP default static scheduling (block)



Non-local accesses due to A

1-D Jacobi: block scheduling

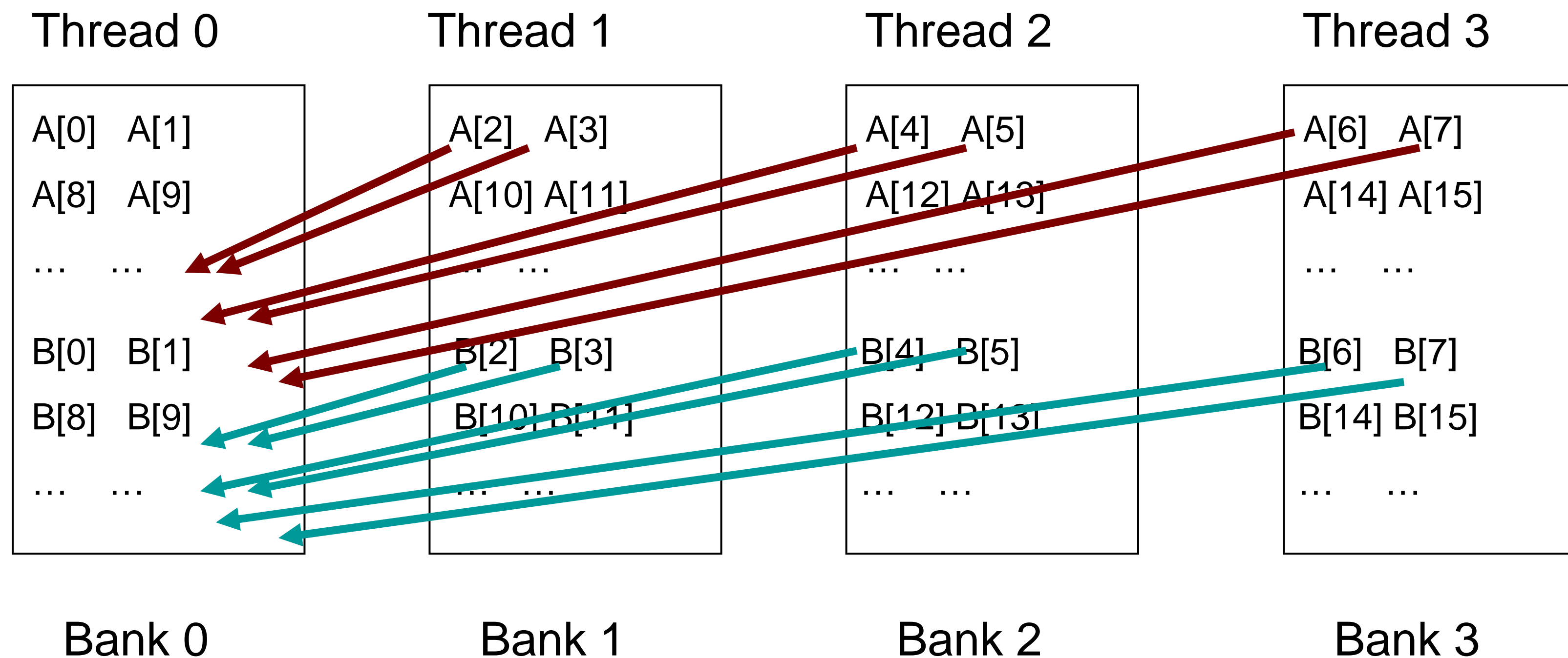
OpenMP default static scheduling (block)



Non-local accesses due to A **Non-local accesses due to B**

1-D Jacobi: block scheduling

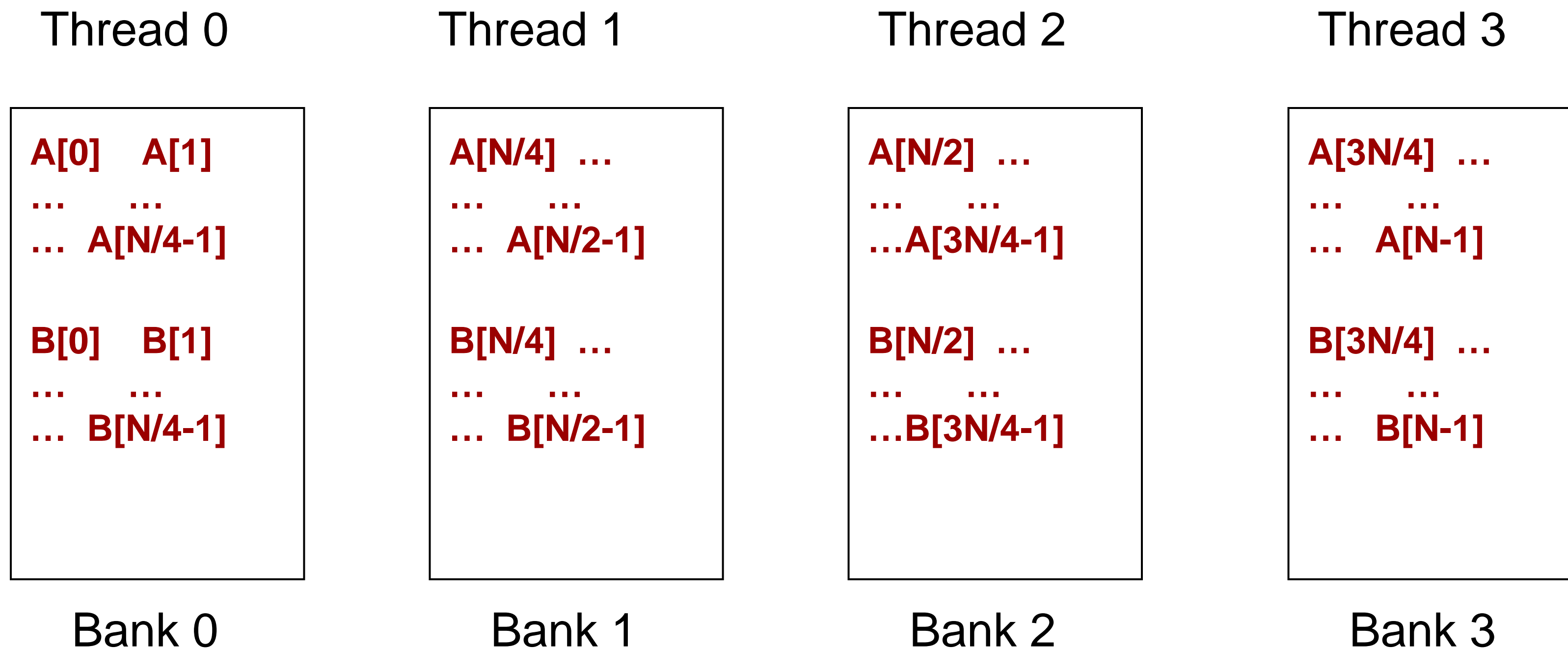
OpenMP default static scheduling (block)



View this as a data layout transformation (line-level transpose) done once

1-D Jacobi: Optimized

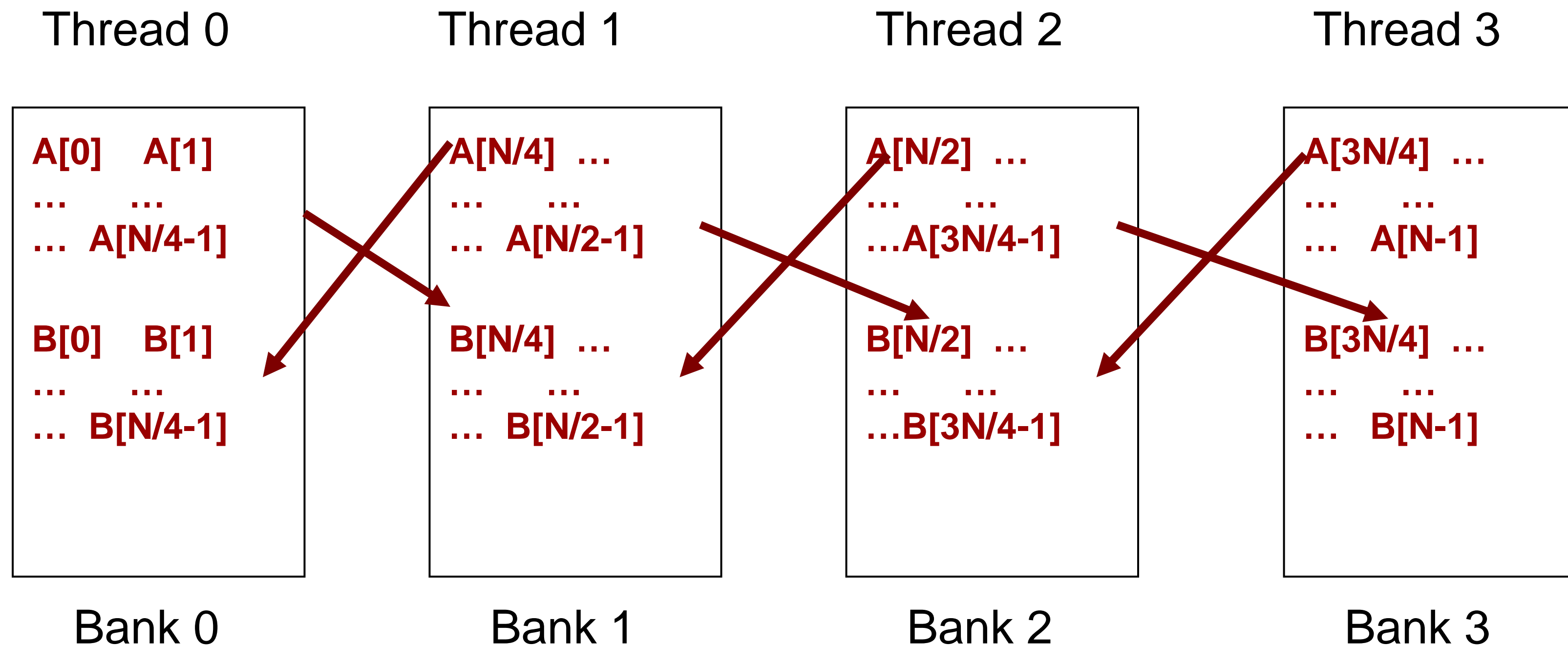
OpenMP default static scheduling (block)
+ data layout transformation



1-D Jacobi: Optimized

OpenMP default static scheduling (block)

+ data layout transformation



Very few non-local bank accesses

Compiler optimizations

- Integrated computation mapping and data layout transformations are needed
  - Data localization into banks in order to match distribution of iterations among processors
  - When is this useful? We can analyze this
  - Can we automate this? Yes
- Preliminary results from simulators (thanks to Mark Hill and his group) on several benchmarks show that the use of suitable non-canonical data layout improves perfomance.

Implications

- Productivity
- Performance
- Multiple personalities:
  - Elvis -> Mort -> Elvis -> Mort ...
- It is not just programmer personas
- These may be “tool personas,” “language personas,” ... as well ...

Mortenstein … (from Eric White)

"When talking about programmer types, there are three personas: Mort, Elvis, and Einstein.

Through deep research and extensive analysis, I have determined that there is a fourth type: Mortenstein. This programmer is an Einstein who is opportunistic. He or she will grab any tool available to do a job quickly - it doesn't matter if the tool is complicated or not - so long as it gets the job done quickly.

My boss suggested that there are a lot of Mortensteins in the Perl community!"

Thank You!