Parallel software pipeline

Toshihiro KONDA

Aim

- **Parallel Software Pipeline**
  - **Consider optimization in massively parallel environment.**

Background

- **In general, Software Pipeline is sequential optimization.**

- **Why?**
  - **In many cases, it becomes a stride access at thread parallel computation.**

  - **Even if you do Software Pipeline as sequential optimization, there is no effect of data prefetch.**
    - **And, since data of another thread is referred to, it is necessary to guarantee that writing is completed.**

    - **Therefore, dependence exists between threads**
      - **The problems are piled up.**

Solution 1. (Introduction of description method)

- **First of all, let's make such a writing possible as a premise**

```
#pragma omp parallel for
  for (i=0; i<max; i++) {
    d[i] = a[i] * b[i] + c[i];
  }
```

What kind of writing is not essence, it is important to be able to instruct.

- **above method of writing**

```
for (i=tid; i<max; i+=maxthread) {
  d[i] = a[i] * b[i] + c[i];
}
```

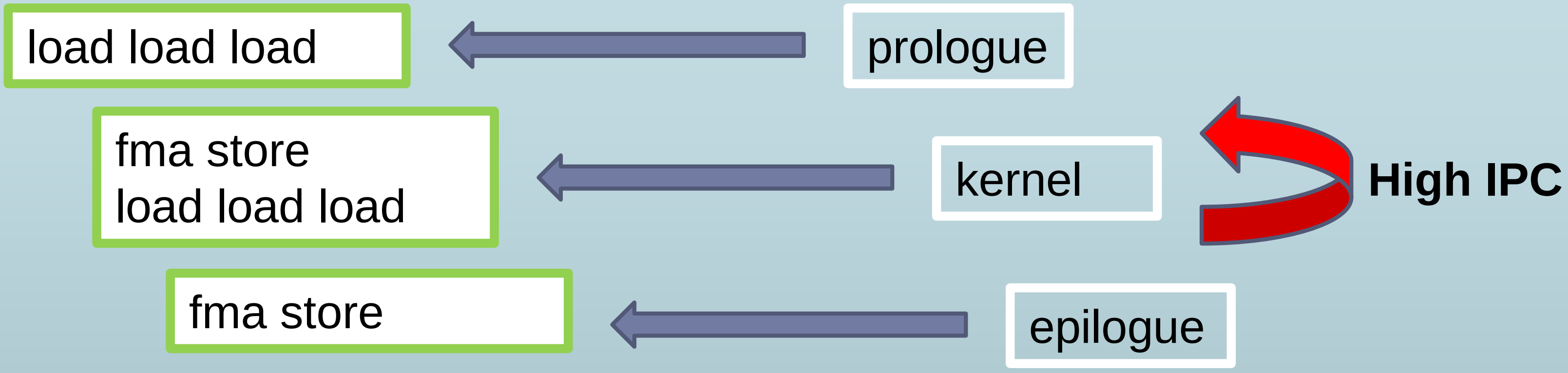
- **is like this in massively parallel environment.**

Solution 2. (Software Pipeline itself)

- **Optimize "as is" without being conscious of pragma on the previous slide.**

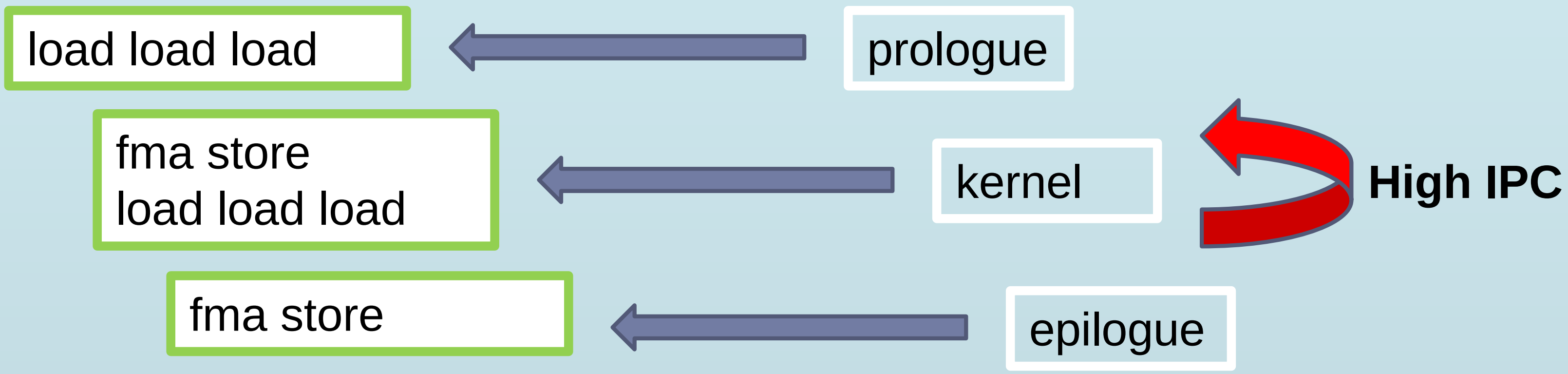
```
#pragma omp parallel for
  for (i=0; i<max; i++) {
    d[i] = a[i] * b[i] + c[i];
  }
```

- **Very simply the image looks like the following**



Solution 3. (Effect of original Software Pipeline)

- **From previous slide.**

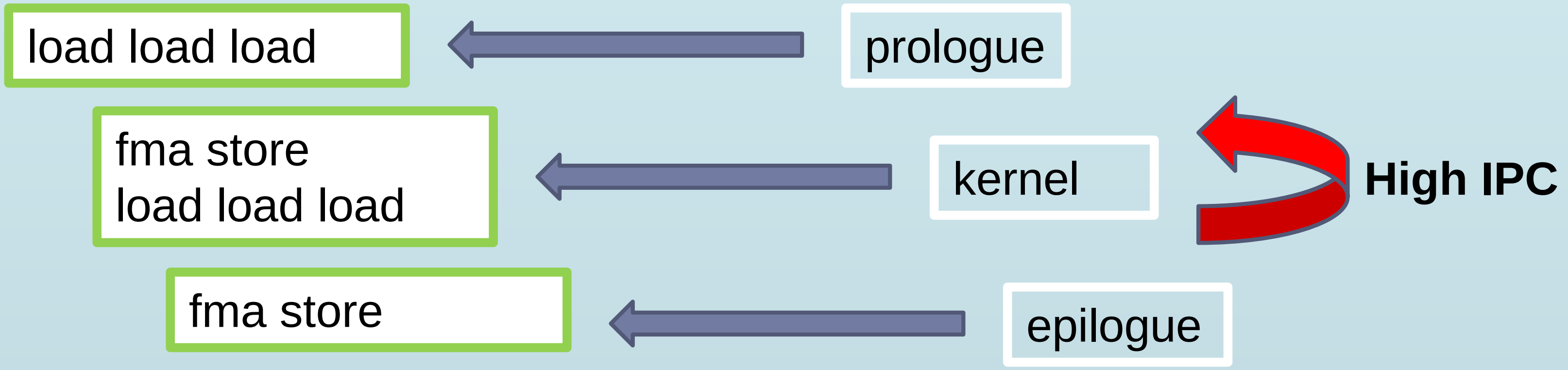


- **What is important here.**

kernel prefetches kernel for some later iterations.
(Depends on MVE<Modulo Variable Expansion>)

Solution 4.
(Problem in “sequential” Software Pipeline)

- **From previous slide.**

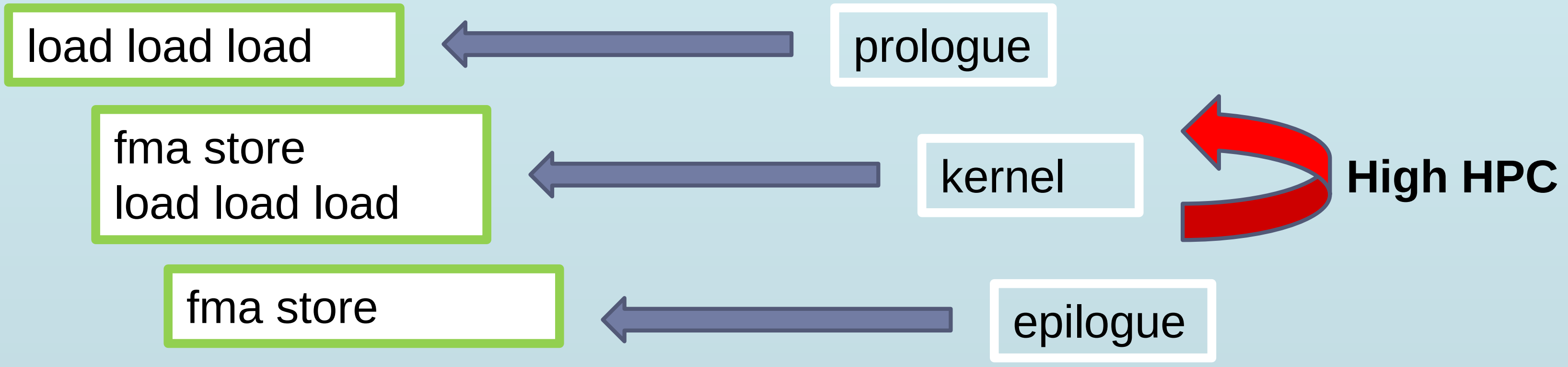


- **kernel prefetches kernel for some later iterations.**
  - **But, It does not become data-prefetch in each thread in this optimized image.**
  - **So, we can not expect the effect.**

  - **Moreover, since it can depend on each other's threads, the opposite effect.**

Solution 5. (Induction variable conversion)

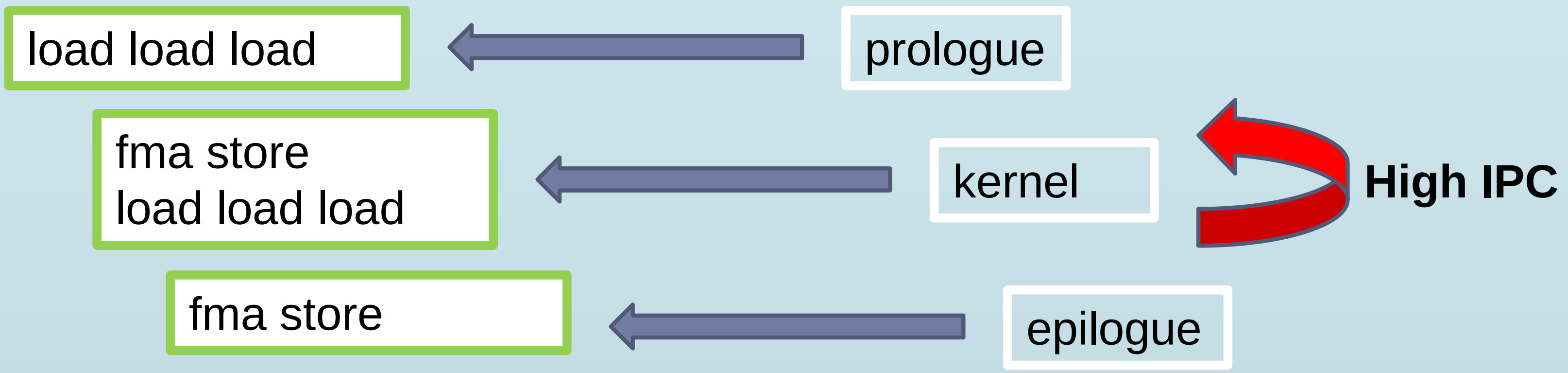
- **From previous slide.**



- **In the sequential image, (for example)"Thread 0" does not have any effect by merely prefetch on "Thread 3"**
- **To parallelize with the number of X, we want to prefetch the data of the element ahead of X.**
  - **So we do Induction variable conversion.**

Solution 6. (Induction variable conversion)

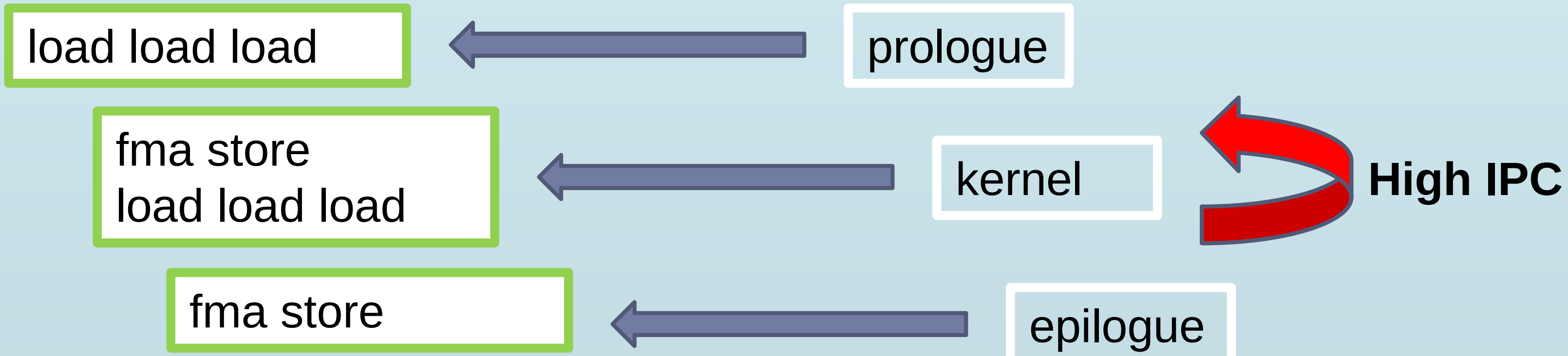
- **From previous slide.**



- **Suppose that kernel contains iterations of three times, perform the following conversion.**
- **0 → 0 3 → 1**
- **1 → 8192 4 → 8193**
- **2 → 16384 5 → 16385**

Solution 7. (What happens?)

- **From previous slide.**



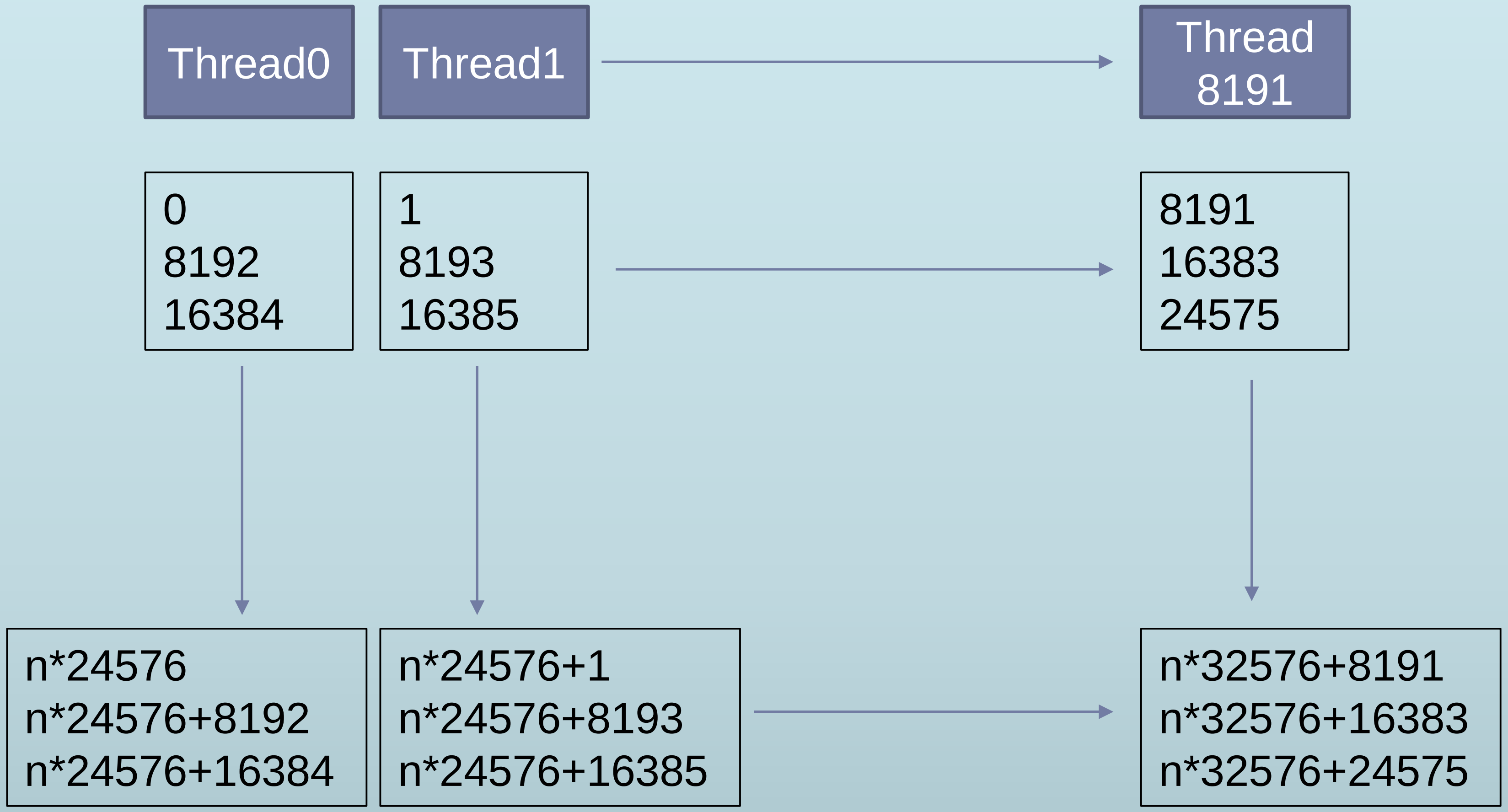
In case of 8192 threads parallelization and three unroll, it is possible to prefetch each 8192 iteration destination in Thread 0 · 1 · 2 (for example)

Latency hiding effect is obtained by prefetching data

※From then on too
(after thread 3)

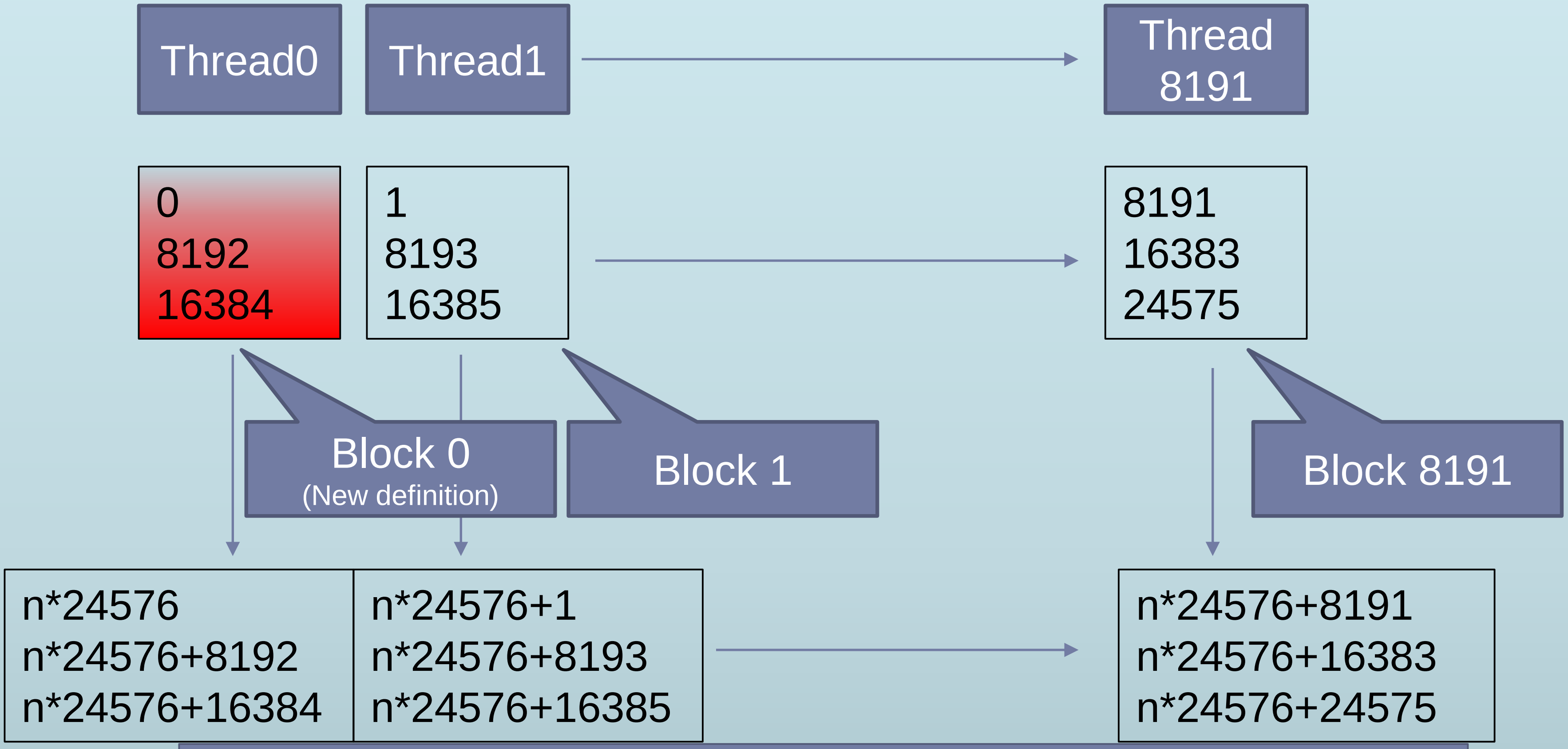
Solution 8.(Execution image of each thread)

- **Example**



Solution 9.(Precondition)

- **Example**



Thread0
Thread1
Thread 8191
0 8192 16384
1 8193 16385
8191 16383 24575
Block 0 (New definition)
Block 1
Block 8191
n*24576 n*24576+8192 n*24576+16384
n*24576+1 n*24576+8193 n*24576+16385
n*24576+8191 n*24576+16383 n*24576+24575

There is no dependence between each block
(It is self-evident since pragma and the loop is software pipelined)。

Solution 10.

From previous slide.

There is no dependence between each block

Each block is unrolled by Software pipelining (In other words, MVE).
And, We can prefetch data by 8192 iteration destination.

Generally, software pipeline which is sequential (horizontal) optimization can be done as vertical optimization.

Since the loop is software pipelined, the data of one iteration is used for subsequent iteration.
Also, latency can be hidden within the range where register files are sufficient.

What is innovative?

- **With explicit thread parallelization description**
  - **We can say that “Array Access” and “incremental value of induction variable” are “stride”.**
  - **So, in general it was difficult to improve performance with software pipeline, .**

- **I introduce a mechanism to generate parallel program with a description like sequential**
  - **In essence, a parallel prerequisite program**
  - **→ Serialization (assuming use of “pragma” etc.)**
  - **→ Serial optimization (classiacal software pipeline)**
  - **→ Inverse conversion to parallel program (induction variable conversion)**

- **Optimization making full use of branches possible Manycore (not GPGPU !)**

Conclusion

- We want to do parallel program easily.
  - introduce a mechanism to generate parallel program with a description like sequential
    - (Conventional way of thinking so far)

- In this time, by describing parallel program as a sequential description and plus alpha, I made “parallel software pipeline” possible.

- As a result, easy description (sequential description) and performance merit (data prefetch effect) are obtained.

Thanks.