Loop transformations and parallelization

Claude Tadonki

LAL/CNRS/IN2P3 University of Paris-Sud
claude.tadonki@u-psud.fr

December 2010

Most of the time, the most time consuming part of a program is on loops. Thus, loops optimization is critical in *high performance computing*. Depending on the target architecture, the goal of loops transformations are:

- improve data reuse and data locality
- efficient use of memory hierarchy
- reducing overheads associated with executing loops
- instructions pipeline
- maximize parallelism

Loop transformations can be performed at different levels by the programmer, the compiler, or specialized tools. At high level, some well known transformations are commonly considered:

- loop interchange
- loop reversal
- loop skewing
- loop blocking
- loop (node) splitting
- loop fusion
- loop fission
- loop unrolling
- loop unswitching
- loop inversion
- loop vectorization
- loop parallelization

Extract and analyze the dependencies of a computation from its polyhedral model is a fundamental step toward loop optimization or scheduling.

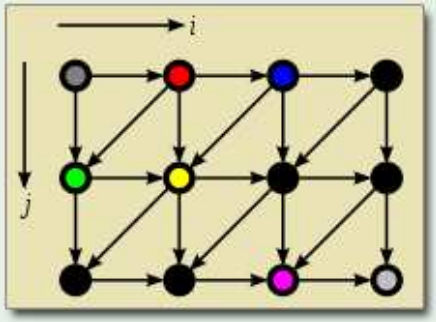
Definition

For a given variable V and given indexes I_1, I_2, if the computation of $X(I_1)$ requires the value of $X(I_2)$, then $I_1 - I_2$ is called a dependence vector for variable V. Drawing all the dependence vectors within the computation polytope yields the so-called dependencies diagram.

Example

```
for (j=1; j<= n; j++)
 for (i=1; i<= n; i++)
   V[i][j]=f(V[i-1][j],
        V[i][j-1],V[i+1][j-1]);
```

The dependence vectors are $(1,0),(0,1),(-1,1)$.



Definition

The computation on the entire domain of a given loop can be performed following any valid schedule. A timing function t_V for variable V yields a valid schedule if and only if

$$t(x) > t(x-d), \forall d \in D_V, \tag{1}$$

where D_V is the set of all dependence vectors for variable V.
For regular loops, affine schedules (i.e. $t(x) = u^T x + v$) are required. In that case, we should have

$$u^T d > 0, \forall d \in D_V. \tag{2}$$

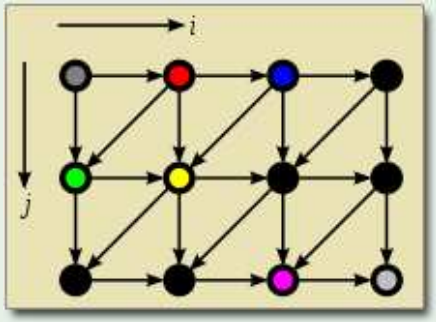
Loop scheduling is important for

- expressing the loop (if not yet done)
- rewriting the loop (for a specific purpose)
- automatic loop transformations

Example

```
for (j=1; j<= n; j++)
 for (i=1; i<= n; i++)
   V[i][j]=f(V[i-1][j],
        V[i][j-1],V[i+1][j-1]);
```

The dependence vectors are $d_1 = (1, 0)$, $d_2 = (0, 1)$, $d_3 = (-1, 1)$.



A valid schedule is given by $t(i, j) = i + n(j - 1)$ (I.e $u = (1, n)^T$ and $v = -n$). We have $u^T d_1 = 1, u^T d_2 = n, u^T d_3 = -1 + n > 0$, for $n > 1$.

Exercise 1. Show that $t(i, j) = n(i - 1) + j$ is not valid.
Exercise 2. Exhibit another valid schedule (check anti-diagonal one).

Definition

The loop interchange transformation switches the order of loops in order to improve data locality or increase parallelism.

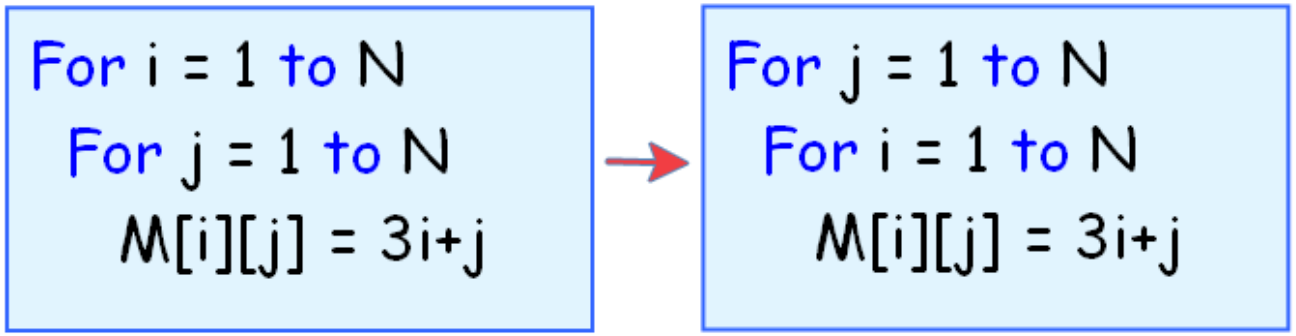


Figure: Loop interchange

Exercise 1. Is it always valid to apply the *loop interchange* ?. Explain
Exercise 2. Give one example where *loop interchange* can help to improve data locality.
Exercise 3. Write and interchange a nested loops that calculates the sum of the values of a $N \times M$ upper-triangular matrix G.

Definition

The loop reversal transformation reverses the order in which the index variable moves. This can can help eliminate dependencies and thus enable other optimizations.

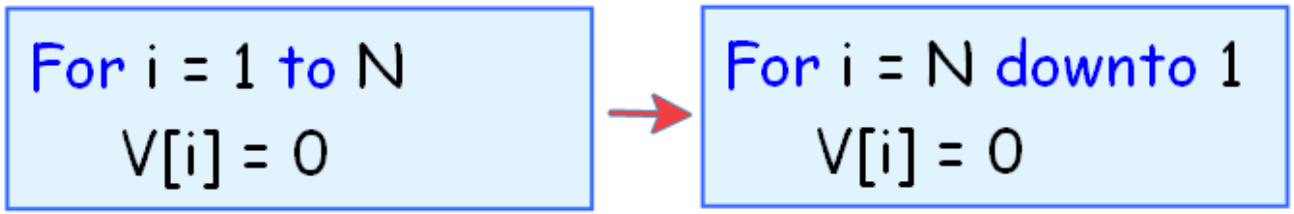


Figure: Loop reversal

In order to apply this transformation, on should care about

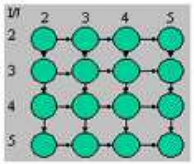
- the direction of the dependence vector
- the commutativity of the computation

Exercise 1. Is it always valid to apply the *loop reversal* ?. Explain
Exercise 2. Write and reverse a loop that calculates the sum of the values of a N-array V.

Definition

The loop skewing transformation changes the shape of the iteration space without changing the dependencies. It looks like a geometrical transformation, which can help to expose a canonical parallelism.

```
For i = 2 to N
  For j = 2 to N
    A[i][j]=A[i-1][j]+A[i][j-1]
```

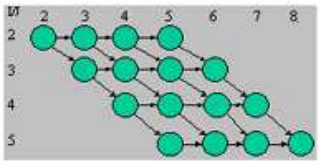


A[i][j] = B[i][i+j-2]

```
For i = 2 to N
  For j = 2 to N
    B[i][i+j-2]=B[i-1][i+j-3]+B[i][i+j-3]
```

k = i+j-2

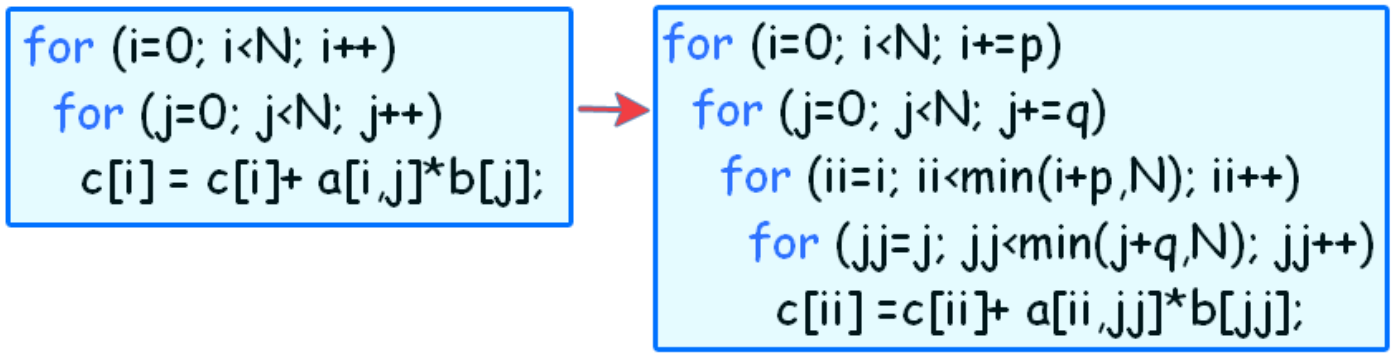
```
For i = 2 to N
  For k = i to i+N-2
    B[i][k]=B[i-1][k-1]+B[i][k-1]
```



Our loop can now be parallelized along the k (i.e. j) axis.

Definition

Loop blocking is a common loop transformation which consists in breaking the entire loop into chunks. This is mainly done on the iteration space and can be seen as a task partitioning.



Loop blocking can be considered for

- improve cache performance (matrix product is a good example)
- derive a coarse grained parallelism from a fine grained model
- handle memory constraints

Definition

Loop splitting breaks a loop into multiple loops which have the same bodies but iterate over different contiguous portions of the index range.

```
For i = 1 to N
  U[i] = 2*W[i] +1;
```

↓

```
For i = 1 to p
  U[i] = 2*W[i] +1;
For i = p+1 to q
  U[i] = 2*W[i] +1;
For i = q+1 to N
  U[i] = 2*W[i] +1;
```

Figure: Loop splitting

```
For i = 1 to N
    U[i] = V[i] + I;
    W[i] = U[i-1] + U[i+1]
```

↓

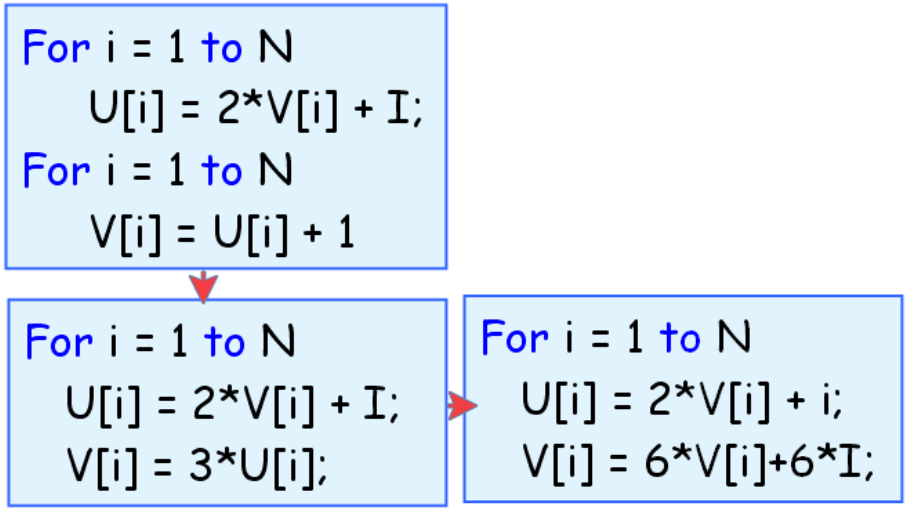
```
For i = 1 to N
    U[i] = V[i] + I;
For i = 1 to N
    W[i] = U[i-1] + U[i+1]
```

Figure: Node splitting

A special case of loop splitting is the so-called loop speeling, where first (or last) few iterations are isolated from the main loop and performed outside.

Definition

Loop fusion (also called loop jamming) combines two adjacent isomorphic loops.

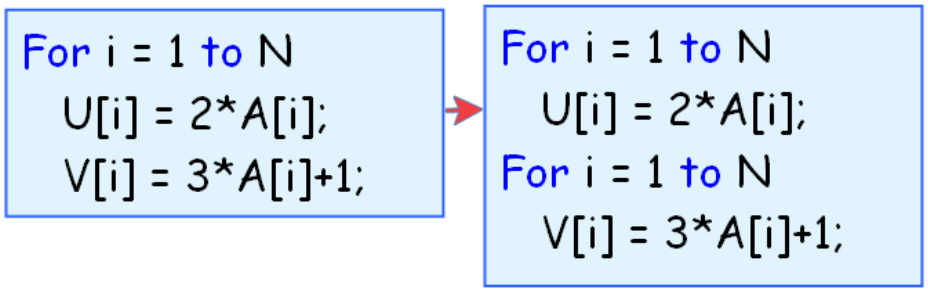


Its purpose is to

- reduce loop overheads
- improve (immediat) data reuse
- reduce data transfers

Definition

Loop fission (also called loop distribution) breaks a loop into multiple loops over the same index range but each taking only a part of the loop's body.



Its purpose is to

- achieve better utilization of locality of reference
- isolate parallelizable loops
- create independent loops, hence creating separate tasks

Definition

Loop unrolling (also called loop unwinding) aggregates consecutive steps of the loop and write them explicitely (without loop constrols).

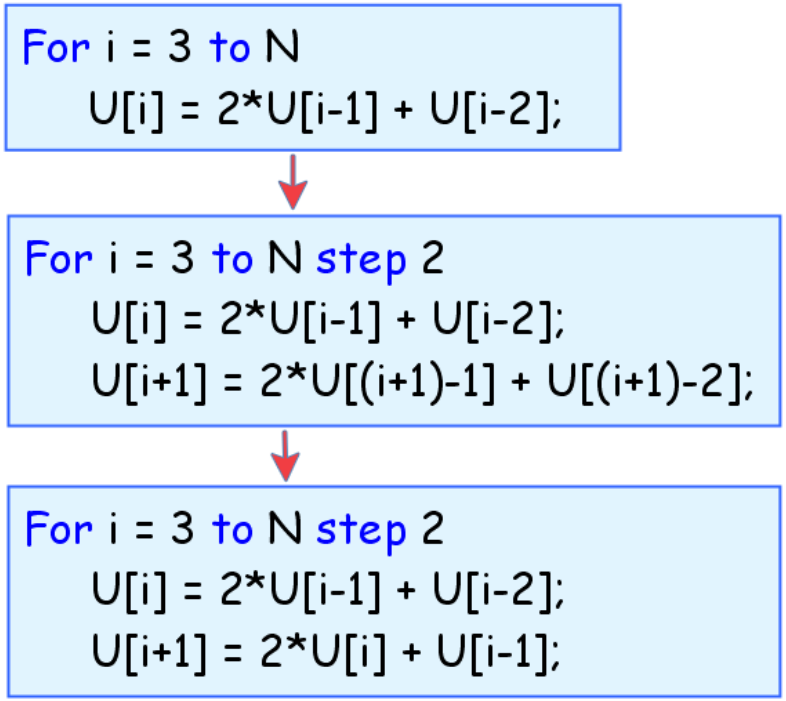
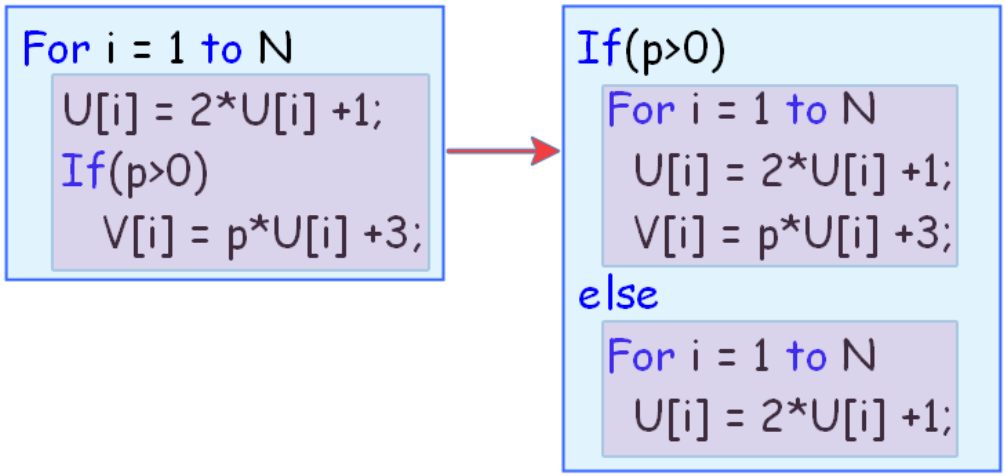


Figure: Loop unrolling with factor 2

Definition

Loop unswitching moves a conditional inside a loop outside of it by duplicating the loop's body accordingly.

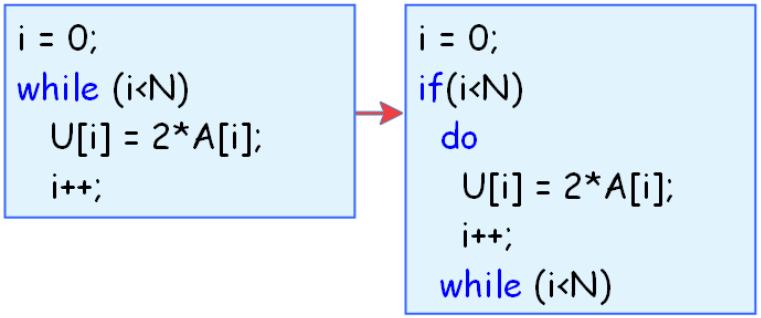


Its purpose is to

- remove intensive (conditional) tests
- simplify the body of the loop

Definition

Loop inversion replaces a while loop by an if block containing a do..while loop.



Definition

Loop vectorization attempts to rewrite the loop in order to execute its body using vector instructions. Such instructions are commonly referred as SIMD (Single Instruction Multiple Data), where multiple identical operations are performed simultaneously by the hardware.

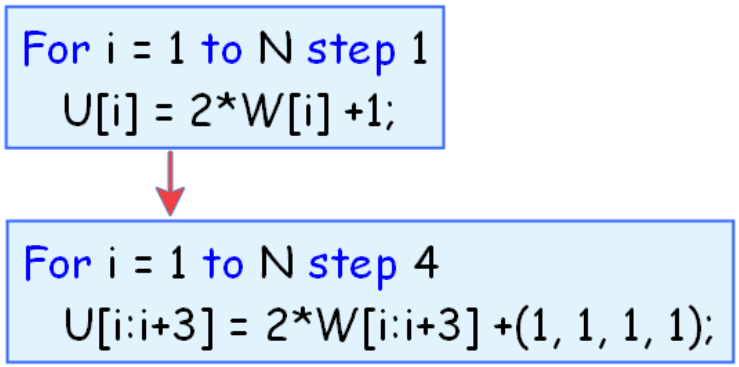
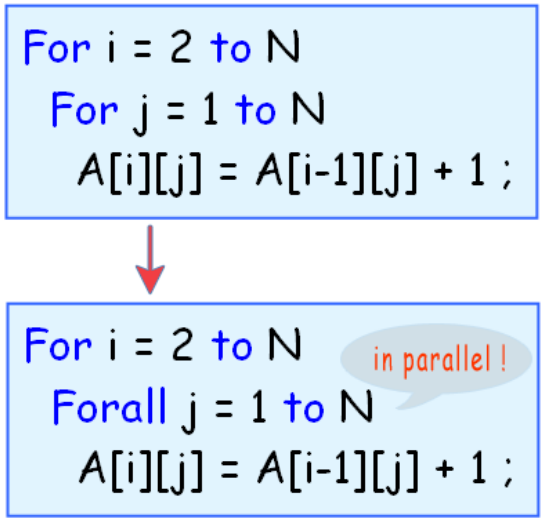


Figure: Loop vectorization scheme (vectors of length 4)

Definition

Loop parallelization restructures the loop to run efficiently on multiprocessor systems. This is a major topic in automatic/systematic parallelization.



Different kinds of loop transformations can be applied in order to expose the parallelism before moving into explicit parallelization.

- R. Allen and K. Kennedy. *Optimizing Compilers for Modern Architectures*. Morgan and Kaufman, 2002.
- F. Irigoin and R. Triolet. *Supernode partitioning*. POPL '88 Proceedings of the 15th ACM SIGPLAN-SIGACT symposium on Principles of programming languages
  `http://www.cri.ensmp.fr/classement/doc/A-179.pdf`
- Utpal Banerjee. *Dependence Analysis (Loop Transformation for Restructuring Compilers)*. Springer; 1 edition (October 31, 1996).
- Utpal Banerjee. *Loop parallelization*. Kluwer Academic Publisher, 1994.