Minimizing Synchronization in Parallel Nested Loops

Reza Rafeh, *Centre for Business, Information Technology and Enterprise, Waikato Institute of Technology, Hamilton, New Zealand, Reza.Rafeh@ wintec.ac.nz*

Mohammad Hossein Roosta, *Department of Computer Engineering, Islamic Azad University, Malayer Branch, Malayer, Iran*

Abstract

Although computing power increases continuously, the need for high computational speed in many scientific applications is growing too. As a result, implementation of parallel applications has gained more attention. Since nested loops are the most time-consuming parts of most programs, we propose a method for scheduling uniform nested loops to processors based on the equation of a straight line which includes the maximum possible number of dependence vectors. Experimental results show that the proposed method imposes a lower communication between processors compared with similar methods.

Background

Parallel processing is a form of computing in which many instructions are executed simultaneously. The process of parallelization in general consists of three steps as follows :

- Decomposing the application into tasks
- Analyzing the dependencies between the decomposed tasks
- Scheduling tasks into the target parallel or distributed system

Generally, there are two types of dependency:

- Data dependencies

  X = Y;

  Z = X;

- Control dependencies

  IF cond THEN s1 ELSE s2;

There are two types of nested loops based on dependencies:

- DOALL loops: Nested loops with no dependency.
- DOACROSS loops: Nested loops with dependencies which are divided into two categories:
  - Uniform: A loop in which the pattern of dependencies remains constant during its execution
  - Non-uniform. A loop in which the pattern of dependencies may have variations during execution.

Proposed Algorithm

Input: Array P which is initialized with the points of set DP= { J_1,. . . , J_m }

Output: The equation of the best straight line (BSL)

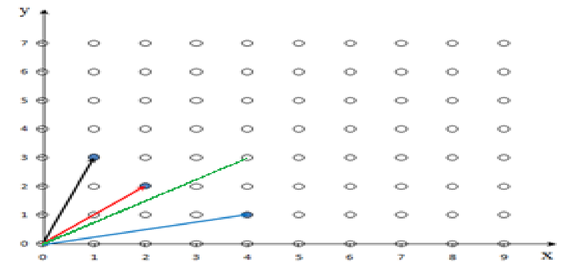
```
int n=1;
for (int i = 1; i <= m - 1; i++) {
  for (int k = i + 1; k <= m; k++) {
    if (there is no line between two points P[i] and P[k]) {
      equations[n].equation = compute the equation of the line between these two points
      equations[n].count = 2;
      for (int r = j + 1; r <= m; r++) {
        if (point P[r] lies on the line)
        equations[n].count++;
      }
      n++; } } }
```

BSL = find the equation with the largest counter

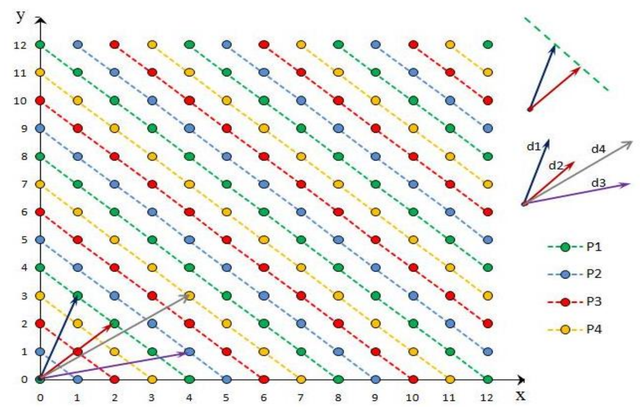
Methods

```
for(int i=k; i<N; i++)
          for(int j=l; j<M; j++){
             A[i][j] = 5 * B[i-1][j-3];
             A[i][j] = A[i][j] + B[i-2][j-2];
             A[i][j] = A[i][j] – 2*B[i-4][j-1];
             A[i][j] = A[i][j] + 4*B[i-4][j-3];
}
```

Finding the dependence vectors

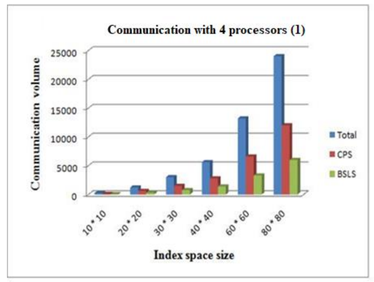


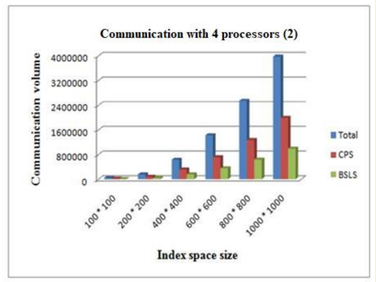
Assigning chains to processors

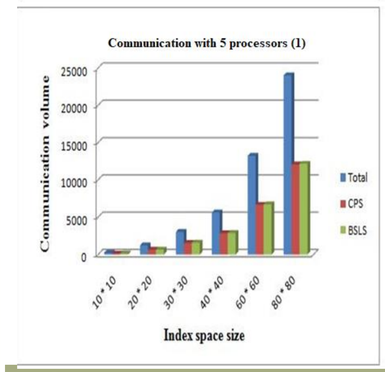


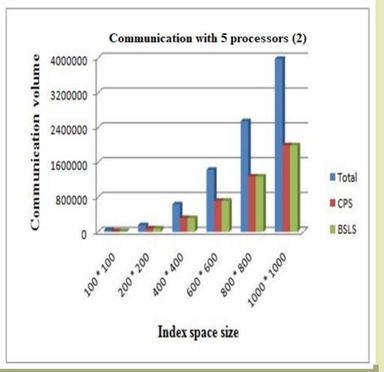
Results

We compared our algorithm (BSLS) with Chain Pattern Scheduling (CSP) method in terms of communication









Conclusions

We proposed a new algorithm, named BSLS, to reduce the communication cost of uniform nested loops which facilitates parallelizing such loops. To enhance data locality, BSLS finds the best straight line which encompasses the maximum number of dependence vectors. Chains are considered as lines parallel with the best straight line in the iteration space. Chains are assigned to processors periodically. Our experimental results show that BSLS imposes lower communication cost than Chain Pattern Scheduling (CSP).

References

- Beletska, A., Bielecki, W., Cohen, A., Palkowski, M., & Siedlecki, K. (2011). Coarse-grained loop parallelization: Iteration space slicing vs affine transformations. *Parallel Computing, 37*(8), 479-497.
- Bondhugula, U. (2013). *Compiling affine loop nests for distributed-memory parallel architectures.* Paper presented at the Proceedings of SC13: International Conference for High Performance Computing, Networking, Storage and Analysis.
- Boulet, P., Darte, A., Risset, T., & Robert, Y. (1994). (Pen)-ultimate tiling? *Integration, the VLSI Journal, 17*(1), 33-51.
- Calland, P. Y., Darte, A., Robert, Y., & Vivien, F. (1998). On the removal of anti-and output-dependences. *International Journal of Parallel Programming, 26*(3), 285-312.
- Ciorba, F. M., Andronikos, T., Drositis, I., Papakonstantinou, G., & Tsanakas, P. (2005). *Reducing the communication cost via chain pattern scheduling.* Paper presented at the Network Computing and Applications, Fourth IEEE International Symposium on.