An Approach for Matching Communication Patterns in Parallel Applications

Chao Ma, Yong Meng Teo, Verdi March, Naixue Xiong,
Ioana Romelia Pop, Yan Xiang He, Simon See

Asia Pacific Science and Technology Center
Sun Microsystems Inc.

December 2008
Singapore

Last Update: 1-Jul-2009

An Approach for Matching Communication Patterns in Parallel Applications

Chao Ma, Yong Meng Teo, Verdi March, Naixue Xiong,
Ioana Romelia Pop, Yan Xiang He, Simon See

TECHNICAL REPORT
APSTC-TR-2008-05

Abstract

Interprocessor communication is an important factor in determining the performance scalability of parallel systems. The communication requirements of a parallel application can be quantified to understand its communication pattern and communication pattern similarities among applications can be determined. This is essential for the efficient mapping of applications on parallel systems and leads to better interprocessor communication implementation among others. This paper proposes a methodology to compare the communication pattern of distributed-memory programs. *Communication correlation coefficient* quantifies the degree of similarity between two applications based on the communication metrics selected to characterize the applications. To capture the network topology requirements, we extract the communication graph of each application and quantifies this similarity. We apply this methodology to four applications in the NAS parallel benchmark suite and evaluate the communication patterns by studying the effects of varying problem size and the number of logical processes.

Keywords: communication pattern similarity, communication correlation coefficient, communication graph similarity

Email:
teoym@comp.nus.edu.sg

Asia Pacific Science and Technology Center
50 Nanyang Avenue, N3-01-C10
Singapore 639798

An Approach for Matching Communication Patterns in Parallel Applications*

Chao Ma[1,2], Yong Meng Teo[1,4], Verdi March[1,4], Naixue Xiong[2],
Ioana Romelia Pop[1,3], Yan Xiang He[2], Simon See[4]

[1]Department of Computer Science, National University of Singapore
[2]College of Computer Science & Technology, Wuhan University
[3]Faculty of Automatic Control and Computer, Politechnica University of Bucharest
[4]Asia-Pacific Science and Technology Center, Sun Microsystems, Inc.

teoym@comp.nus.edu.sg

Last Update: 1-Jul-2009

Abstract

Interprocessor communication is an important factor in determining the performance scalability of parallel systems. The communication requirements of a parallel application can be quantified to understand its communication pattern and communication pattern similarities among applications can be determined. This is essential for the efficient mapping of applications on parallel systems and leads to better interprocessor communication implementation among others. This paper proposes a methodology to compare the communication pattern of distributed-memory programs. Communication correlation coefficient *quantifies the degree of similarity between two applications based on the communication metrics selected to characterize the applications. To capture the network topology requirements, we extract the communication graph of each application and quantifies this similarity. We apply this methodology to four applications in the NAS parallel benchmark suite and evaluate the communication patterns by studying the effects of varying problem size and the number of logical processes.*

Keywords: *communication pattern similarity, communication correlation coefficient, communication graph similarity*

1. Introduction

As scientific parallel computing matures, the demand for large-scale computing is expected to grow rapidly [10, 16, 18]. With larger computing systems, the overhead of communication becomes an increasing important factor that limits application performance. When interconnection networks lack behind the

*A version of this report is accepted for publication at the proceedings of the 23^{rd} IEEE International Parallel and Distributed Processing Symposium (IPDPS), IEEE Computer Society Press, Rome, Italy, May 25–29, 2009.

capacity required by applications, application performance will degrade. However, over provision of communication requirements increases the overall system cost. Both performance and cost are important to users and system vendors. To achieve good cost-performance trade-off, it is important to understand the underlying communication behavior and patterns of parallel applications [2, 8, 21]. Researchers and practitioners in parallel computing also recognized that understanding application communication patterns play an important role in application performance [9, 11, 17, 20, 23].

The process of studying communication patterns, application tuning, and configuring new parallel systems, is both time-consuming and costly. Thus, it is desirable to reuse communication performance characterization to reduce the time and cost in tuning new applications and systems. To address this issue, we propose an approach where by design the communication properties are independent of system characteristics. This allows our characterization of communication patterns for an application to be reused to estimate its performance on different systems.

The main contribution of this paper is an approach for matching communication patterns of scientific parallel applications. In our PACPM (Parallel Application Communication Pattern Matching) approach, a communication pattern comprises of four main properties, namely, *temporal*, *spatial*, *volume*, and *communication graphs*. All these properties are by design system independent. Given two applications, we first measure their four properties using a profiling tool. Next, we apply *rank transformation* on the measured temporal, spatial, and volume properties to normalize the impact of problem size [7, 14]. Communication similarity between applications is quantified using two metrics, (θ and ε). The first metric, θ, where $-1 \leq \theta \leq 1$, quantifies similarity based on the temporal, volume, and spatial properties. This metric measures, for each communication property, the *correlation coefficient* between the two applications. The second metric, ε, where $0 \leq \varepsilon \leq 1$, quantifies the similarity between two communication graphs (or maximum common subgraphs).

We tested our approach using four applications in the NPB (NAS Parallel Benchmark) 3.0 benchmark suite [4], namely BT (Block Tri-diagonal), SP (Pentadiagonal Solver), MG (Multi Grid), and LU (LU Decomposition). We observe that communication patterns are similar for each application run with a different problem size. When comparing among different applications, we find that BT and SP exhibit the highest level of similarity for temporal, volume, and spatial properties, compared to other application combinations. In addition, we show that MG has a distinct communication graph, and BT, SP, and LU have a remarkably common subgraph.

The remainder of this paper is organized as follows. In section 2, we discuss the related work. Section 3 presents our proposed approach. An evaluation of our approach using the NPB benchmarks is discussed

in section 4. Section 5 contains our conclusions.

2. Related Work

To the best of our knowledge, this is the first paper on matching the communication patterns of parallel applications. Existing related work focuses either on the characterization of collective and point-to-point communication of parallel applications [11, 20, 23], or amelioration of the efficiency of the communication of parallel applications [2, 3, 17].

A number of research has been done in describing the significant role that the communication pattern plays in current scientific parallel applications. Researchers employed various strategies to characterize the communication patterns of current parallel applications. For example, Zamani et. al. characterize the temporal, spatial and volume properties of six parallel applications [23]. Visualization of communication patterns is discussed in [9]. A different method of characterizing communication patterns by extracting the causal relationships from the group of logical processes (LPs) is proposed in [6]. Kim et. al. introduce the term "locality" to describe the possibility of certain kind of events occur during the execution of the parallel application [11]. In [3], Afsahi et. al. ameliorate the communication efficiency of parallel applications by using message prediction based on the message communication locality which exists at the receiver sides of message-passing parallel applications. Shalf et. al. propose a reconfigurable hybrid interconnect based on an application's communication characteristic to achieve a better performance [17].

However, these studies discussed attempts to characterize the communication of parallel applications from different aspects or ameliorate the efficiency of current applications but without detecting the similarity between the communication patterns which is also important for users and developers. On the other hand, our work aims to detect and quantify the similarity of communication pattern between different parallel applications.

3. Design

3.1. Architecture Overview

The architecture of the PACPM (Parallel Application Communication Pattern Matching) consists of five main components: data extraction, rank transformation, compute communication graph similarity, compute communication correlation coefficient and determine pattern similarity (Figure 1). Integrated Performance Monitoring (IPM) [1], a low overhead profiler is used to obtain the four properties: temporal, volume, spatial, and communication graphs. Next, we employ *rank transformation* to the temporal, volume, and spatial

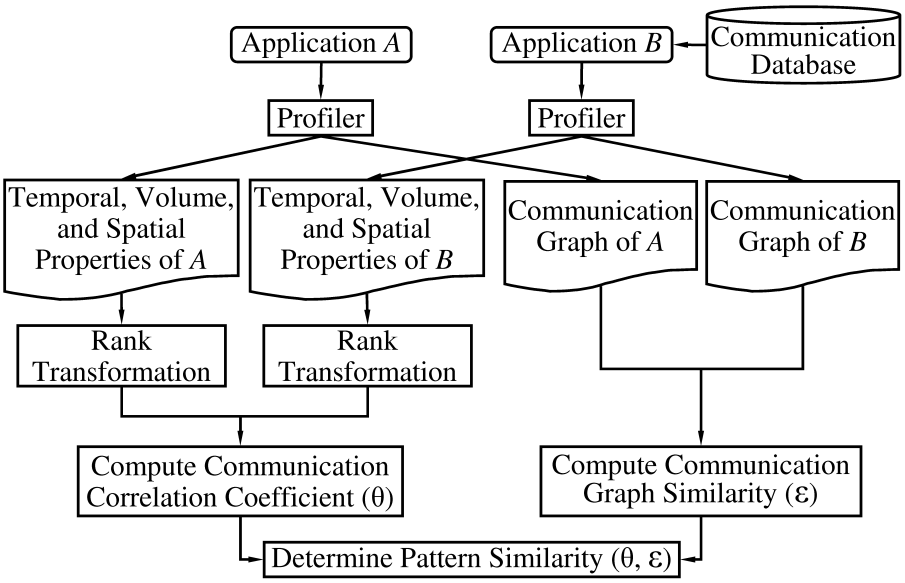


Figure 1. Overview of PACPM

properties. For each *ranked* property i, we derive a *metric correlation coefficient* θ_i. Then, the *communication correlation coefficient* is defined as the mean of $\theta_{temporal}$, θ_{volume}, and $\theta_{spatial}$ which are the metric correlation coefficients of temporal, volume, and spatial properties, respectively. For simplicity, we used the arithmetic mean but for example weighted mean can be used to factor in the importance of different properties. Communication graph similarity (ε) quantifies the similarity between two communication graphs. Both θ and ε constitute the degree of communication pattern similarity between two applications.

3.2. Detailed Design

Consider a distributed-memory application consisting of n communicating *logical processes* (LPs). In MPI applications, each LP corresponds to an MPI process (rank). In this paper, we considers only two main MPI operations: `MPI_Send` and `MPI_Recv`. However, our approach can be generalized to other types of MPI operations as well such as collective operations.

There are two main issues in communication pattern matching:

1. *Number of LPs*

   An application may exhibit different communication patterns when run with different number of logical processes. To address this issue, we assume that the number of logical processes is the same in both communication patterns to be compared.

2. *Problem Size*

   Communication patterns generated by the same applications may vary because of the different appli-

cation problem size [19]. To address this issue, we introduce *rank transformation* (see Section 3.2.1). Thus, a communication pattern matching model should identify similar patterns instead of merely the exact patterns.

Our approach includes a novel graph isomorphism approach and a brute-force maximum common subgraph algorithm to communication pattern matching. Maximum common subgraph (MCS) method has been used to detect pattern similarities in areas such as bioinformatics [22], chemistry [13, 15] and pattern recognition [5, 12].

3.2.1. Communication Correlation Coefficient (θ)

To determine the matching degree between two communication patterns, we derive the *correlation coefficient*. Generally speaking, the correlation coefficient measures the linear association between data sets. The benefit is that this measurement is relatively simple and well understood, and it works with performance data containing noisy sample data which is hard to avoid. Quantitatively, a correlation coefficient is a value ranging from -1 to +1 that measures the strength of the linear association between data sets. A correlation of -1 and +1 denotes a perfect negative and positive linear relationship between data sets, respectively. A correlation of 0 denotes no linear relationship between data sets.

The similarity between communication patterns is quantified using *communication correlation coefficient* θ. Figure 2 illustrates the steps to calculate θ. Firstly, *observed* properties are measured using the IPM profiler. Then, *rank transformation* is applied to these observed properties, which results in *ranked* properties. Then, *metric correlation coefficients* θ_i are calculated. Finally, θ is calculated as the arithmetic mean of θ_i.

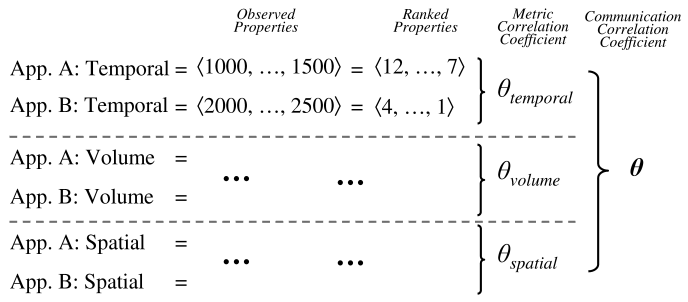


Figure 2. Calculating Communication Correlation Coefficient θ

As illustrated in Figure 2, three properties are used to derive θ, namely *temporal*, *volume*, and *spatial*. Definition 1–3 formalizes these three properties. Basically, in a communication pattern consisting of n LPs, each property is a vector of n elements. That is, every logical process produces one value in each of the properties.

Definition 1 (Temporal Property) *Given an application consisting of n communicating logical processes, the temporal property is defined as a vector of n elements:*

$$temporal = \langle temporal_1, ..., temporal_j, ..., temporal_n \rangle \tag{1}$$

where j denotes LP_j, and $temporal_j$ denotes the rate of message generated by LP_j. Furthermore, $temporal_j$ is defined as:

$$temporal_j = |sends_j| / work_j \tag{2}$$

where $|sends_j|$ is the total number of messages sent by LP_j and $work_j$ denotes the amount of computation performed by LP_j.

An example of $work_j$ is the number of floating-point operations (i.e. FLOP count) in numeric application.

Definition 2 (Volume Property) *Given an application consisting of n communicating logical processes, the volume property is defined as a vector of n elements:*

$$volume = \langle volume_1, ..., volume_j, ..., volume_n \rangle \tag{3}$$

where j denotes LP_j and $volume_j$ denotes the total volume of messages sent by LP_j. Furthermore, $volume_j$ is defined as:

$$volume_j = \sum_{m=1}^{|sends_j|} msg\_size_{j,m} \tag{4}$$

where m denotes a message, $|sends_j|$ is the total number of messages sent by LP_j, and $msg\_size_{j,m}$ denotes the size of message m sent by LP_j.

Definition 3 (Spatial Property) *For an application consisting of n communicating logical processes, the spatial property is defined as a vector of n elements:*

$$spatial = \langle spatial_1, ..., spatial_j, ..., spatial_n \rangle \tag{5}$$

where j denotes LP_j and $spatial_j$ denotes the total number of unique message sources and destinations of LP_j. Furthermore, $spatial_j$ is defined as:

$$spatial_j = |neighbors_j| \tag{6}$$

where neighbors$_j$ is a set of LPs who communicates (both send and receive) with j.

Rank transformation is applied on the observed properties to obtain ranked properties. This is done for two reasons [7, 14]. Firstly, the ranked metrics remain relatively stable while the real values can fluctuate widely with changing problem sizes. Secondly, this statistical method is non-parametric so it does not rely on assumptions that the raw data are drawn from a given probability distribution. Since each observed property is a vector of n elements where n is the number of LPs, each ranked property is also a vector of n elements. As an example of rank transformation, consider a vector of $\langle 12,3,5,28\rangle$. The rank of each element, as shown in Figure 3(a), is derived from a descending order of the vector's. Thus, the ranked vector becomes $\langle 2,4,3,1\rangle$. Another vector of $\langle 100,60,70,95\rangle$ will also yield a similar rank vector of $\langle 2,4,3,1\rangle$ (Figure 3(b)).

Values (Descending Order)	**Rank**
28	1
12	2
5	3
3	4

(a) Vector $\langle 12,3,5,28\rangle$

Values (Descending Order)	**Rank**
100	1
95	2
70	3
60	4

(b) Vector $\langle 100,60,70,95\rangle$

Figure 3. Example of Rank Transformation

A *metric correlation coefficient* θ_i, where $-1 \leq \theta_i \leq 1$, represents the linear correlation of property i between applications. Given a ranked property i which is a vector of n elements, θ_i is calculated as shown in Definition 4.

Definition 4 (Metric Correlation Coefficient θ_i) *The metric correlation coefficient θ_i $(-1 \leq \theta_i \leq 1)$ is determined by the following equation:*

$$\theta_i = \frac{n(\sum_{j=1}^{n} r_j s_j) - (\sum_{j=1}^{n} r_j)(\sum_{j=1}^{n} s_j)}{\sqrt{n(\sum_{j=1}^{n} r_j^2) - (\sum_{j=1}^{n} r_j)^2}\sqrt{n(\sum_{j=1}^{n} s_j^2) - (\sum_{j=1}^{n} s_j)^2}} \tag{7}$$

where r_j is the ranked value in metric m_i for application A, s_j is the ranked value in metric m_i for application B, n is the number of values in data sets r and s. The value of j ranges from 1 to n (i.e. the number of LPs), and i ranges from 1 to the cardinality of M which is the set of all selected metrics in this work.

For simplicity, the communication correlation coefficient θ is defined as the arithmetic mean of $\theta_{spatial}$, θ_{volume}, and $\theta_{spatial}$ as shown in Definition 5.

Definition 5 (Communication Correlation Coefficient θ) *Communication correlation coefficient θ is defined as:*

$$\theta = \frac{1}{|i|}\sum_{i}\theta_i \quad where\ i = \{temporal, volume, spatial\} \tag{8}$$

where $-1 \leq \theta \leq 1$.

3.2.2. Communication Graph Similarity (ε)

The topology of a communication pattern is represented as an undirected graph $G = (V_G, E_G)$, where V_G is the set vertices of G and E_G is the set of edges. Each node or vertex in G represents a logical process of an application, and each undirected edge represents communications between logical processes. A communication graph complements the spatial metric. While the spatial metric measures the total number of communication partners of each logical process, it does not fully reflects the network topology. For instance, as shown in Figure 4, G and H represent communication graphs of two different parallel applications.

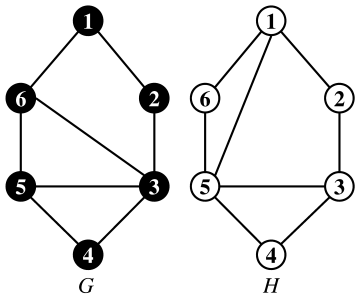


Figure 4. Example of Communication Graphs

From Figure 4, we can see that in the perspective of the spatial metric, G and H are the same because they all have a 4-degree node, two 3-degree nodes and three 2-degree nodes. This means that in each application, there is one logical process communicating with 4 partners, two logical processes communicating with 3 partners, and three logical processes communicating with 2 partners. However, the communication graphs reveal the differences in communication topologies, i.e., there is no one-to-one node mapping between G and H.

The similarity between undirected graphs is formulated as a maximum common subgraph problem, i.e., to find the largest subgraphs such that there exists a bijection between them (also known as *isomorphism*). Two communication patterns exhibit the same topology if their communication graphs are isomorphic. Otherwise, the graphs are partially similar when the subgraphs are isomorphic. The metric ε quantifies the degree of similarity between communication graphs based on graph isomorphism.

Definition 6 (Isomorphic Degree ε) *Let $G = (V_G, E_G)$, $H = (V_H, E_H)$ denote the undirected communication graphs for two applications and $S = (V_S, E_S)$ be the maximum common subgraph of G and H. The communication graph similarity ε ($0 \leq \varepsilon \leq 1$) is determined by the following equation:*

$$\varepsilon = \frac{2 \mid E_S \mid}{\mid E_G \mid + \mid E_H \mid} \tag{9}$$

where $\mid E_S \mid$, $\mid E_G \mid$, and $\mid E_H \mid$ are the sum of edges in sets E_S, E_G and E_H, respectively.

Figure 5–9 shows the algorithm to find the maximum common subgraph based on graph isomorphism. Let $G = (V_G, E_G)$ and $H = (V_H, E_H)$ denote two communication graphs. As shown in Figure 5, the main function $StructureMatching(G,H)$ firstly splits G and H according to the $Split(G)$ function (Figure 6). Then, it invokes $GraphIsomorphism(G,H)$ to check for isomorphism. Otherwise, it invokes $SubgraphIsomorphism(G,H)$, a costlier function to find the maximum subgraphs that are isomorphic.

```
1.  Main function : StructureMatching (Graph G, Graph H)
2.  Input : two undirected communication graphs G and H
3.  Output: a valid bijection between G and H or their maximal common subgraphs
4.  {
5.      //partition V_G into subsets (V_G1, V_G2, ..., V_Gr), (d_1, d_2, ..., d_r)
6.      call Split (G);
7.      //partition V_H into subsets (V_H1, V_H2, ..., V_Ht), (c_1, c_2, ..., c_t)
8.      call Split (H);
9.      if (r=t) and (|V_Gi| = |V_Hi| for each i ∈ [1,r]) and (d_i = c_i) then
10.         //Optimized subfunction to detect isomorphism of G and H
11.         call GraphIsomorphism(G, H);
12.     else
13.         //Find the maximum common subgraphs of G and H
14.         call SubgraphIsomorphism(G, H);
15.     end if
16. }
```

Figure 5. Detection of Structural Similarity

The $Split(G)$ function, Figure 6, partitions V_G into subsets $(V_{G1}, V_{G2}, ..., V_{Gr})$ such that two vertices belonging to the same subset V_{Gi} if they have the same degree. Let d_i denotes the set of the degree of vertices in V_{Gi}. We reorder the partitions so that $d_1 < d_2 < ... < d_r$, and then output the partition $(V_{G1}, V_{G2}, ..., V_{Gr})$ and its degree $(d_1, d_2, ..., d_r)$. Similarly for graph H, we output the partition $(V_{H1}, V_{H2}, ..., V_{Ht})$ and its degree $(c_1, c_2, ..., c_t)$.

$GraphIsomorphism(G,H)$ is an optimized algorithm to detect the isomorphism between G and H. As shown in Figure 7, the function repeatedly *refine* the initial partition until there is no change between the

```
1.  Function: Split (Graph G)
2.  Input: an undirected communication graph G(V_G, E_G);
3.  Output: a set of subsets of all nodes in G in ascending order of their degree;
4.  {
5.      partition V into subsets (V_G1, V_G2, ..., V_Gr) so that two vertices belong to the same
6.          subset V_Gi if they have the same degree;
7.      let d_i be the degree of the vertices in V_Gi;
8.      reorder the partition so that d_1 < d_2 < ... < d_r;
9.      for each node v_Gi ∈ V_G
10.         for each V_Gj ∈ (V_G1, V_G2, ..., V_Gr)
11.             if (the degree of v_Gi equals to d_j) then
12.                 V_Gj = V_Gj + {v_Gi};
13.             end if
14.         end loop
15.     end loop
16.     return (V_G1, V_G2, ..., V_Gr);
17. }
```

Figure 6. Procedure Split

current partition ρ and the last partition, afterwhich the matching (isomorphic) part is found. The partition refining is implemented as function *Refine* (Figure 8) which includes finding the lexicographic relationship between sets of nodes (Definition 7).

Definition 7 (Lexicographical Relationship) *Given two strings* $\vec{X} = (x_1, x_2, ..., x_k)$ *and* $\vec{Y} = (y_1, y_2, ..., y_k)$, *whose entries are real numbers,* $\vec{X}$ *is lexicographically smaller than* $\vec{Y}$, *written as* $\vec{X} \prec \vec{Y}$, *if there exists* $i \in [1,k]$ *such that* $x_i < y_i$ *and* $x_j = y_j$ *for each* $j \in [1,i]$.

The notation and pseudocode of $SubgraphIsomorphism(G,H)$ is shown in Table 1 and Figure 9. It is called by the main function *StructureMatching* (Figure 5, line 14) if the main function's heuristic determine that G and H cannot be perfectly isomorphic (Figure 5, line 14). As $SubgraphIsomorphism(G,H)$ traverses a subgraph search space, in the worst case its execution time is exponential. To keep the execution time within a practical limit, we can set an upper bound for the number of nodes to be matched between graphs. However, the consequence is that the resulted common subgraphs may be suboptimal.

3.2.3. Pattern Similarity

Communication pattern similarity can be classified by considering (θ, ε) as a coordinate in a 2-dimensional Cartesian space. Specifically, two patterns are similar if θ and ε is equal to or greater than a threshold. Figure 10 shows an example using a threshold of 0.6. If (θ, ε) falls in the shaded top-right region, the communication patterns are classified as similar.

```
1.  Function: GraphIsomorphism(Graph G, Graph H)
2.  Input: two undirected communication graphs G and H;
3.  Output: a valid bijection between G and H;
4.  {
5.      //refine G whose partition is π_G = (V_G1, V_G2, ..., V_Gk) of V_G
6.      set ρ = π_G;
7.      repeat
8.          ρ = Refine(ρ);
9.      until (Refine(G, ρ) = ρ)
10.     //refine H whose partition is π_H = (V_H1, V_H2, ..., V_Hk) of V_H
11.     set σ = π_H;
12.     repeat
13.         σ = Refine(H, σ);
14.     until (Refine(H, σ) = σ)
15.     //find all valid bijections between ρ and σ;
16.     if (each subset in ρ equals to the corresponding subset in σ) then
17.         return this bijection ;
18.     end if
19. }
```

Figure 7. Detection of Graph Isomorphism

Notation	Description
$N(v_{Gi})$	Set of nodes in G which are adjacent to node v_{Gi}
$N(v_{Hi})$	Set of nodes in H which are adjacent to node v_{Hi}
Largest	Bijection of maximum common subgraphs of G and H
Current	Bijection of current common subgraphs of G and H
S	Maximum common subgraph between G and H
S_G	Set of nodes which have been matched in G
S_H	Set of nodes which have been matched in H
Φ	Null set

Table 1. Notations used in the function SubgraphIsomorphism

```
1.  Function: Refine (ρ)
2.  Input: graph G(V_G, E_G), π = (V_G1, V_G2, ..., V_Gr) of V_G;
3.  Output: a new partition in V_G;
4.  {
5.      set ρ=π;
6.      //initialize the vector of each vertex in graph G
7.      for each vertex x ∈ V_G then
8.          let (d_1, d_2, ..., d_r) denotes a vector where d_i(x) is the number of vertices in V_Gi
9.          that are adjacent to node x;
10.     end loop
11.     //refine the subsets based on vectors of vertices
12.     for each i ∈ [1, r] then
13.         partition V_Gi into subsets (V_Gi,1, V_Gi,2, ..., V_Gi,p) so that two vertices x and y
14.         belong to V_Gi, iff d(x) = d(y);
15.     end loop
16.     let d_i,t be the vector of the vertices in V_Gi,t where i ∈ [1, r] and t ∈ [1, p];
17.     //reorder subsets lexicographically
18.     for each i ∈ [1, r] then
19.         lexicographically reorder {V_Gi,t} into {d_i,t};
20.         let L_i be the resulting list ;
21.     end loop
22.     concatenate L_i where i = {1, 2, ..., k};
23.     return the total partition (V_G1,1, ..., V_G1,p, ..., V_Gr,1, ..., V_Gr,p)
24.       and the set of vectors {d_i,t};
25. }
```

Figure 8. Vertice Set Refinement

4. Evaluation

To validate our approach, we apply PACPM to quantify the similarity of communication patterns among four applications in the NPB 3.0 benchmark suite [4]. As defined in Section 3.2.1, a communication pattern is described using four properties: *temporal*, *volume*, *spatial*, and *communication graph*. By definition (Definition 1–3), temporal, volume, and spatial properties are *LP-based* metrics. The communication graph is an application-level property. These four metrics are designed to be independent of the architecture and configuration of the underlying parallel systems. We verified the correctness of this design criteria on three different systems[1]: (i) Intel x64 with 100 MB/s Ethernet, Linux, gcc, and OpenMPI, (ii) Opteron x64 with InfiniBand SDR, Linux, gcc, and Scali MPI, and (iii) SunFire 6800 with US-IV 1.2GHz, SunOS 5.10, Sun Studio 12, and Sun HPC ClusterTool 8.0. In addition, these properties are time-independent, and as such, we have verified that our results are not influenced by the mapping of LPs to physical processors.

The four selected applications are BT (Block Tridiagonal), SP (Scalar Pentadiagonal), MG (Multigrid),

[1]Though the number of floating-point operations (i.e. FLOP count) is theoretically affected by the compilers and CPU architecture, we find that the difference measured on the three systems is negligible ($< 1\%$).

```
1.  Function: SubgraphIsomorphism(Graph G , Graph H)
2.  Input: two undirected communication graphs G and H;
3.  Output: the maximum common subgraph between G and H;
4.  {
5.      k = 0, Largest = Current = S = S_G = S_H = Φ;
6.      for each v_{G_i} ∈ V_G then
7.        for each v_{H_i} ∈ V_H then
8.          select (v_{G_i}, v_{H_i}) as the root entry point of the search tree ;
9.          set Current_{k,G} = v_{G_i}, Current_{k,H} = v_{H_i};
10.         set S_G = S_G + {v_{G_i}}, S_H = S_H + {v_{H_i}};
11.         k++;
12.         repeat
13.           select (v_{G_j}, v_{H_j}) as the child point where v_{G_j} ∈ (N(v_{G_i}) − S_G) and v_{H_j} ∈ (N(v_{H_i}) − S_H);
14.           set Current_{k,G} = v_{G_j}, Current_{k,H} = v_{H_j};
15.           set S_G = S_G + {v_{G_j}}, S_H = S_H + {v_{H_j}};
16.           k++;
17.         until ({N(v_{G_j}) − S_G} = Φ) or ({N(v_{H_j}) − S_H} = Φ)
18.         if (|Current| > |Largest|) then
19.           set Largest = Current;
20.           clear Current;
21.         end if
22.         if all possible branches have been searched then
23.           return Largest;
24.         else
25.           go to the upper level of the search tree which has unsearched branches;
26.         end if
27.       end loop
28.     end loop
29. }
```

Figure 9. Detection of Graph Isomorphism

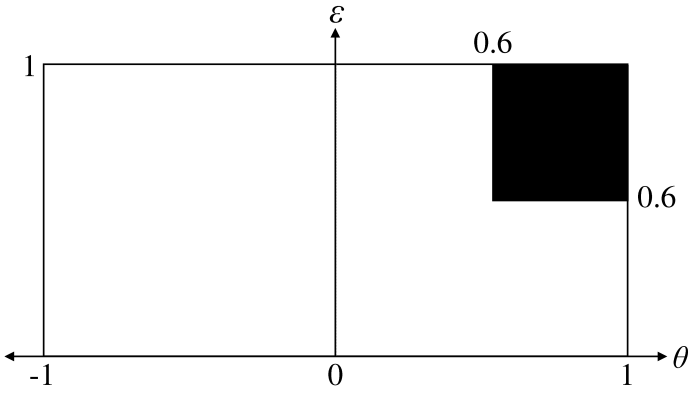


Figure 10. Interpretation of Pattern Similarity

and LU (Lower-Upper Diagonal). For each application, we consider three problem sizes: A, B, and C, which correspond to *small*, *medium*, and *large* problem size, respectively. Both BT and SP require that the number of processor cores to be a square, whereas MG and LU require that the number of processor cores

to be a power of two. In this experiment, the amount of work in each LP $work_j$ (Definition 1) is represented by the number of floating-point operations (in GFLOP).

The testbed infrastructure is a cluster of servers. Each node has two quad-core Intel E5320 CPUs (i.e. 1.86 GHz, 64 KB per-core L1 cache, and 4096 KB shared L2 cache), 4 GB memory, and is running Linux 2.6.25 (which is patched with PerfCtr 2.6 to enable hardware-counter profiling). Nodes are interconnected using a 100 MB/s Ethernet network, and the MPI stack is OpenMPI 1.2.

In this paper, all experiments are performed on 16 and 64 processor cores. While these system sizes are relatively small compared to the present state-of-the-art parallel systems, it is sufficient for demonstrating the feasibility of PACPM because the chosen communication properties are independent of the architecture and configuration of the execution platform.

4.1. Methodology

To quantify how communication patterns are influenced by increased workload or degree of parallelism, we design two sets of experiments:

1. Vary workload, but fix the number of logical processes (LP) to 16.

2. Fix workload to problem size B, but vary the number of logical processes from 16 to 64.

Communication metrics for an MPI application such as the total number of messages sent and received, the volume of messages, and the source-destination of messages are collected by IPM [1], a low-overhead application profiler. IPM also supports hardware-counter profiling through PAPI. In our experiments, we use PAPI 3.6 to enable IPM to measure the number of floating point operations executed by each logical process.

4.2. Results and Analysis

4.2.1. Effect of Varying Problem Sizes

We first present a general observation on the communication patterns of BT, LU, SP, and MC under different workload. Next, we evaluate the pattern similarity among these applications. Lastly, we discuss the potential implications of our findings.

To simplify our analysis, we base our general observation on summarized properties. Specifically, we look at the average send rate of all LPs within an application[2], the total volume of all LPs, and the average

[2]The average send rate is defined as $\frac{\sum_{j=1}^{n} |send_j|}{n \sum_{j=1}^{n} GFLOP_j}$ where $GFLOP_j$ denotes the GFLOP count incurred by LP j.

in-degree or out-degree of all LPs within an application. Table 2 shows these summarized properties for BT, LU, SP, and MC, assuming they are parallelized into 16 LPs.

Property	BT	SP	MG	LU
Temporal (sends/GFLOP)	A: 31.6 B: 7.6 C: 1.8	A: 127.2 B: 30.6 C: 7.4	A: 129.6 B: 142.4 C: 20.4	A: 455.4 B: 174.8 C: 68.2
Volume (bytes)	A: [4K, 256K) B: [16K, 256K) C: [64K, 1M)	A: [16K, 64K) B: [16K, 256K) C: [16K, 1M)	A: [4, 256K) B: [4, 256K) C: [4, 1M)	A: 98% in [256, 1K) 2% in [64K, 256K) B: 99% in [256, 4K) 1% in [64K, 256K) C: 99% in [1K, 4K) 1% in [256K, 1M)
Spatial	Out: 6 In: 6	Out: 6 In: 6	Out: 4.5 In: 4.5	Out: 3 In: 3

Table 2. Communication Profile of Four NPB Applications (16 LPs)

Based on Table 2, the following are observed:

1. The average send rate (i.e. the per-application temporal property) for BT, SP, and LU decreases as the problem size is increased. For MG, the send rate increases from problem A to B, but then decreases from problem B to problem C. The explanation is as follow.

    - In BT and SP, the send rate decreases because the number of sends is the same for the all problem sizes, but their GFLOP count is increased 4x and 17x for problem A-to-B and problem A-to-C, respectively.
    - In LU, when the problem size is increases, the number of sends also increases, but is still outpaced by the growth of GFLOP by 2.5x and 6.5x for problem A-to-B and problem A-to-C, respectively. Hence, its average send rate decreases with larger problems.
    - Lastly, in MG, when increasing the problem size from A to B, the growth of sends is 1.1 times of the growth of GFLOP; this explains the increased send rate. However, from A to C, the growth of GFLOP becomes more than 6x of the send's growth. Hence the send rate in problem size C decreases into less than 6x of A's send rate.

2. The total message size, i.e. the per-application volume property, increases as problem size is increased. This is clearly shown by the upper-limit of message sizes in each problem size. We also determine that both BT and SP have medium-to-large messages (i.e. 4 KB to 1 MB); MG has small-to-large message sizes (i.e. 4 bytes to 1 MB), while LU has predominantly small-to-medium messages (i.e. 256 bytes to 4 KB). Specifically:

- In BT-A, messages are distributed evenly across sizes, but more than 60% messages in BT-B and BT-C are closer to their respective upper limit. The same characteristic is exhibited by SP.
- Messages in MG are nearly evenly spread within the ranges shown in Table 2.
- LU-A has 98% small messages (up to 1 KB); LU-B uses 74% small messages (up to 1 KB), 25% medium messages (up to 4 KB), and 1% large messages (up to 256 KB); whereas LU-C has 99% medium messages and 1% large messages (up to 1MB).

3. The average connectivity degree (i.e. the per-application spatial property) is independent of the problem size, because we observe that the communication graphs show the same connectivity regardless of the problem size (Figure 11).

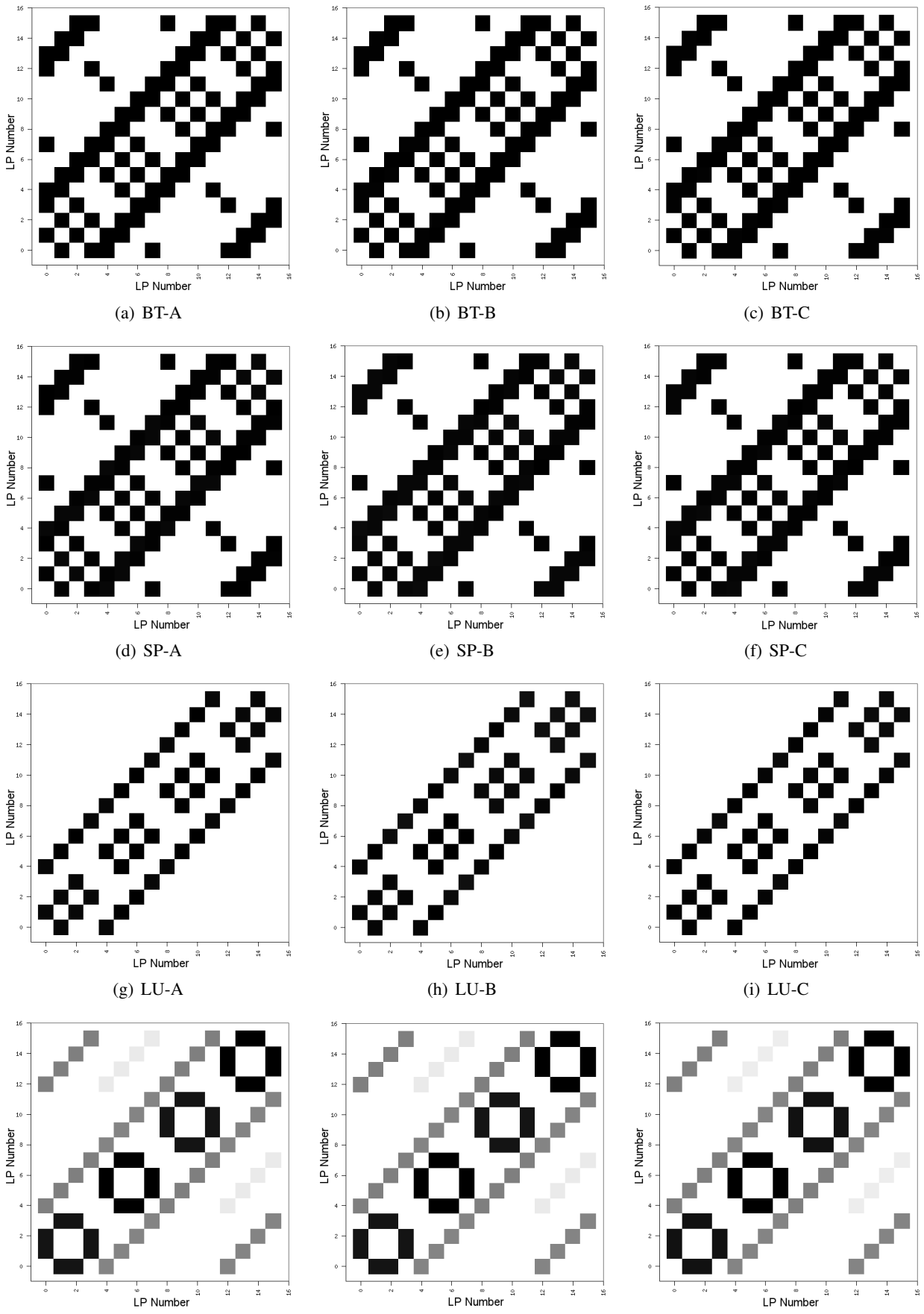
To quantify how the communication pattern of each application changes with different problem size, we first determine the correlation coefficient for temporal property ($\theta_{temporal}$), volume property (θ_{volume}), and spatial property ($\theta_{spatial}$). Based on these correlation coefficients (Table 3), we derive the pattern similarity (θ, ε).

Application	Problem	$\theta_{temporal}$	θ_{volume}	$\theta_{spatial}$	(θ, ε)
BT	A-B	1.000	1.000	1.000	(1.000, 1.000)
	B-C	1.000	1.000	1.000	(1.000, 1.000)
	C-A	1.000	1.000	1.000	(1.000, 1.000)
LU	A-B	0.992	0.998	1.000	(0.997, 1.000)
	B-C	0.998	1.000	1.000	(0.999, 1.000)
	C-A	0.997	0.999	1.000	(0.997, 1.000)
SP	A-B	1.000	0.822	1.000	(0.941, 1.000)
	B-C	1.000	0.682	1.000	(0.894, 1.000)
	C-A	1.000	0.655	1.000	(0.885, 1.000)
MG	A-B	1.000	1.000	1.000	(1.000, 1.000)
	B-C	1.000	1.000	1.000	(1.000, 1.000)
	C-A	1.000	1.000	1.000	(1.000, 1.000)

Table 3. Pattern Similarity with Different Problem Sizes (16 LPs)

In terms of temporal and spatial, our results show that in all applications, both properties have a positive correlation (i.e. close to 1) for all combinations of problem sizes. This is because the temporal and spatial properties have a similar fluctuation among LPs, even as the problem size is increased. The example in Figure 12 clearly illustrates this phenomenon: the ranked temporal properties from both problems have a similar fluctuation, even though the actual send rates differ by 1.5x.

In terms of volume, our results indicate that only in SP where the problem size affects the communication patterns. This is evidence from the relatively low θ_{volume} of SP when problem size C is involved, i.e., $\theta_{volume} = 0.682$ and $\theta_{volume} = 0.655$ for B–C and C–A, respectively. The reason for θ_{volume} of SP to be lower, compared to other applications, is the different fluctuation between SP-C and the other two problem sizes.



(a) BT-A (b) BT-B (c) BT-C

(d) SP-A (e) SP-B (f) SP-C

(g) LU-A (h) LU-B (i) LU-C

(j) MG-A (k) MG-B (l) MG-C

Figure 11. Communication Graphs of Four Selected NPB Applications (16 LPs)

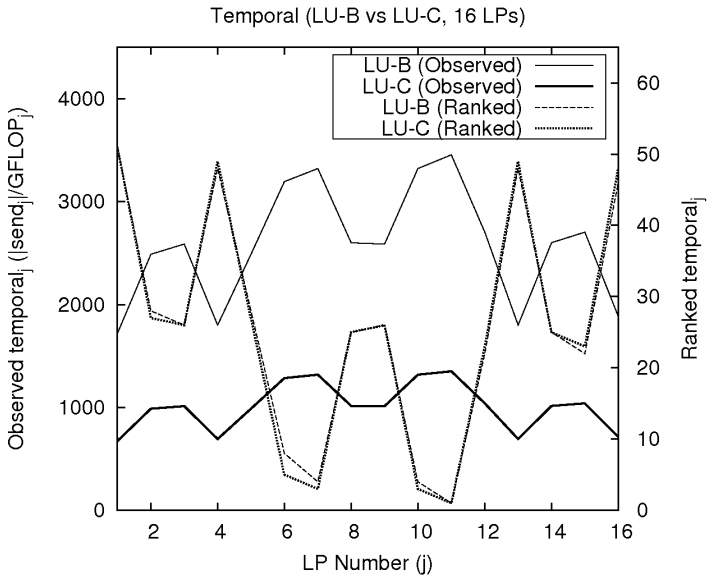


Figure 12. Observed and Ranked Temporal Properties for LU-B and LU-C (16 LPs)

Figure 13 illustrates an example of the fluctuations observed between SP-B and SP-C, where the ranked volumes of SP-B and SP-C show a different trend on the following LPs: 2-3, 4-5, 6-8, and 11-13.

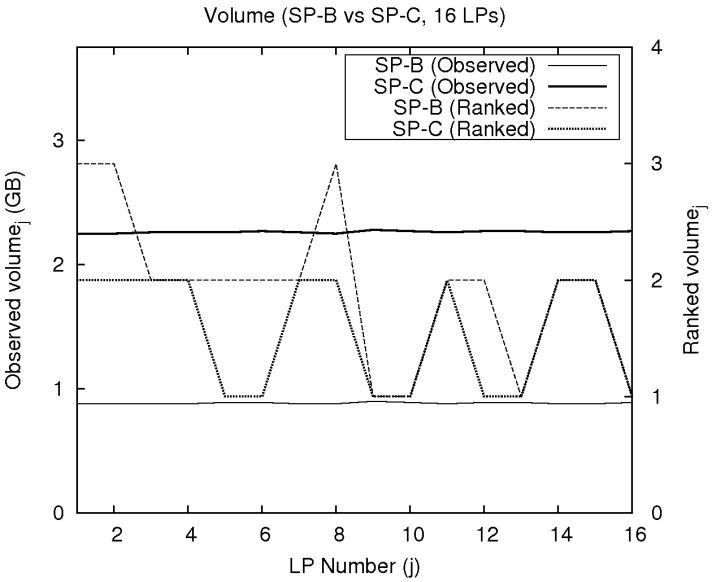


Figure 13. Observed and Ranked Volume Properties of SP-B and SP-C (16 LPs)

The pattern similarity (θ, ε) is shown by the right-most columns in Table 3. In terms of θ, BT, LU, and MG shows a similar fluctuation across LPs, regardless of the problem size. The similarity in SP is relative lower due to the non-correlation of θ_{volume}. However, the value of θ being at least 0.885 suggests that volume properties of different problem size are still fairly similar. We also observe that the values of ε are all 1.000, because the communication graphs are not affected by the problem size. This is similar with the observed $\theta_{spatial}$.

From the general observation and the similarity results, several implications can be drawn. Firstly, code tuning to improve load-balance among LPs would give a similar outcome in BT, LU, and MG regardless of problem sizes, since their temporal and volume show a positive correlation. And secondly, optimizing the mapping of LPs to processor cores is likely to benefit all problem sizes due to the positive correlation of $\theta_{spatial}$ and $\varepsilon = 1$.

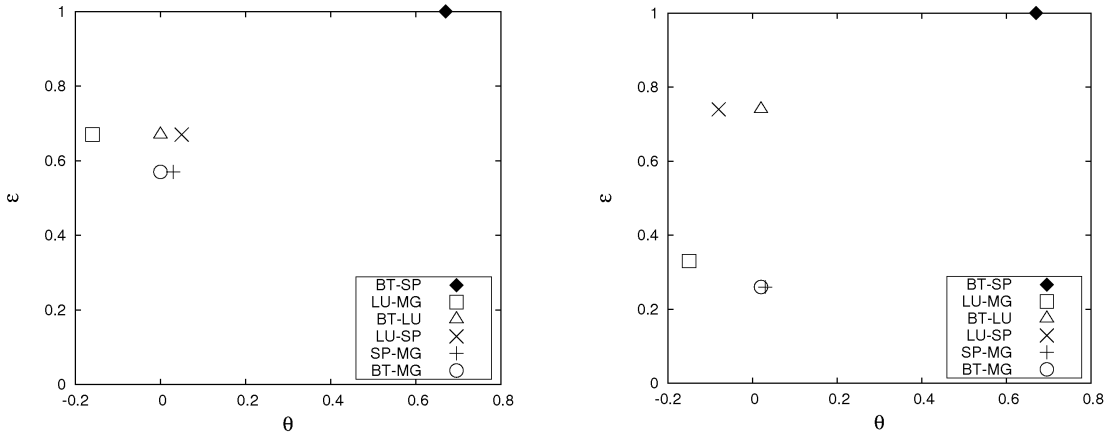
4.2.2. Varying Number of Logical Processes

To evaluate the impact of the number of LPs to the similarity degree, we experiment with problem B on 16 LPs and 64 LPs. Table 4 shows the similarity degree of BT-SP, LU-MG, BT-LU, LU-SP, SP-MG and BT-MG.

Application	LP	$\theta_{temporal}$	θ_{volume}	$\theta_{spatial}$	(θ, ε)
BT-SP	16	1.000	0.000	1.000	(0.667, 1.000)
	64	1.000	0.000	1.000	(0.667, 1.000)
LU-MG	16	0.000	-0.475	0.000	(-0.158, 0.667)
	64	0.022	-0.330	-0.154	(-0.154, 0.329)
BT-LU	16	0.000	0.000	0.000	(0.000, 0.667)
	64	0.072	0.000	0.000	(0.024, 0.737)
LU-SP	16	0.000	0.163	0.000	(0.054, 0.667)
	64	0.072	-0.308	0.000	(-0.079, 0.737)
SP-MG	16	0.000	-0.102	0.000	(0.034, 0.571)
	64	0.071	-0.172	0.000	(0.034, 0.263)
BT-MG	16	0.000	0.000	0.000	(0.000, 0.571)
	64	0.071	0.000	0.000	(0.024, 0.263)

Table 4. Pattern Similarity under Different LPs (Problem Size B)

To further facilitate the interpretation of these results, we plot the results of pattern similarity in a 2-D Cartesian space (Figure 14).



(a) 16 LPs (b) 64 LPs

Figure 14. Pattern Similarity among BT, LU, SP and MG for Problem Size B

Based on these plots, we report four main observations:

1. Our method classifies the communication patterns of BT and SP as similar to each other. This is signified by both θ and ε being larger than 0.6 (i.e. the threshold introduced in Section 3.2.3).

2. In terms of θ, except for BT-SP, there is no positive or negative correlation between applications as shown by their θ being close to zero. In addition, the number of LPs has a negligible impact on θ, with the absolute difference is at most 0.13. This indicates that θ among the four NPB applications can be detected reliably with as low as 16 LPs.

3. For BT-SP, LU-SP and BT-LU, the communication graphs (ε) is not significantly affected by the number of LPs. However, the similarity of communication graphs for LU-MG, BT-MG, and SP-MG decreases as the number of LPs is increased from 16 to 64. As shown in Figure 15, this is because only on 64 LPs that the communication graph of MG shows a remarkable difference in repeatable sub-patterns with LU, BT, and SP.

4. As can be seen from Figure 15, BT and SP exhibits the same communication graph, and hence, their ε is 1. LU shares the same inner and outer diagonal with BT and SP, and hence, the ε is from 0.667 to 0.737. However, none of the inner and outer diagonal of BT and LU are apparent in MG. This explains the low ε especially with 64 LPs. Overall, this finding suggests that for NPB, communication graphs can be compared reliably with a relatively large number of LPs.

5. Conclusion and Future Work

We discussed an approach to determine the communication pattern similarities of distributed-memory programs. Communication similarity between two programs is quantified using two metrics that form a coordinate on a 2-dimensional Cartesian space. Firstly, the communication correlation coefficient measures the degree of similarity between two applications using metrics selected to characterize its communication behavior. Secondly, communication topology among logical processes is abstracted as undirected communication graph and similarity between graphs is determined through graph (or maximum common subgraph) isomorphism. We applied our approach on four NPB benchmarks using three LP-level communication properties, namely, temporal, volume and spatial. These LP-level properties are independent of the underlying machine architecture and configuration. Another advantage of our approach is that the impact of problem size on the similarity detection is minimized by applying rank transformation on the raw data.

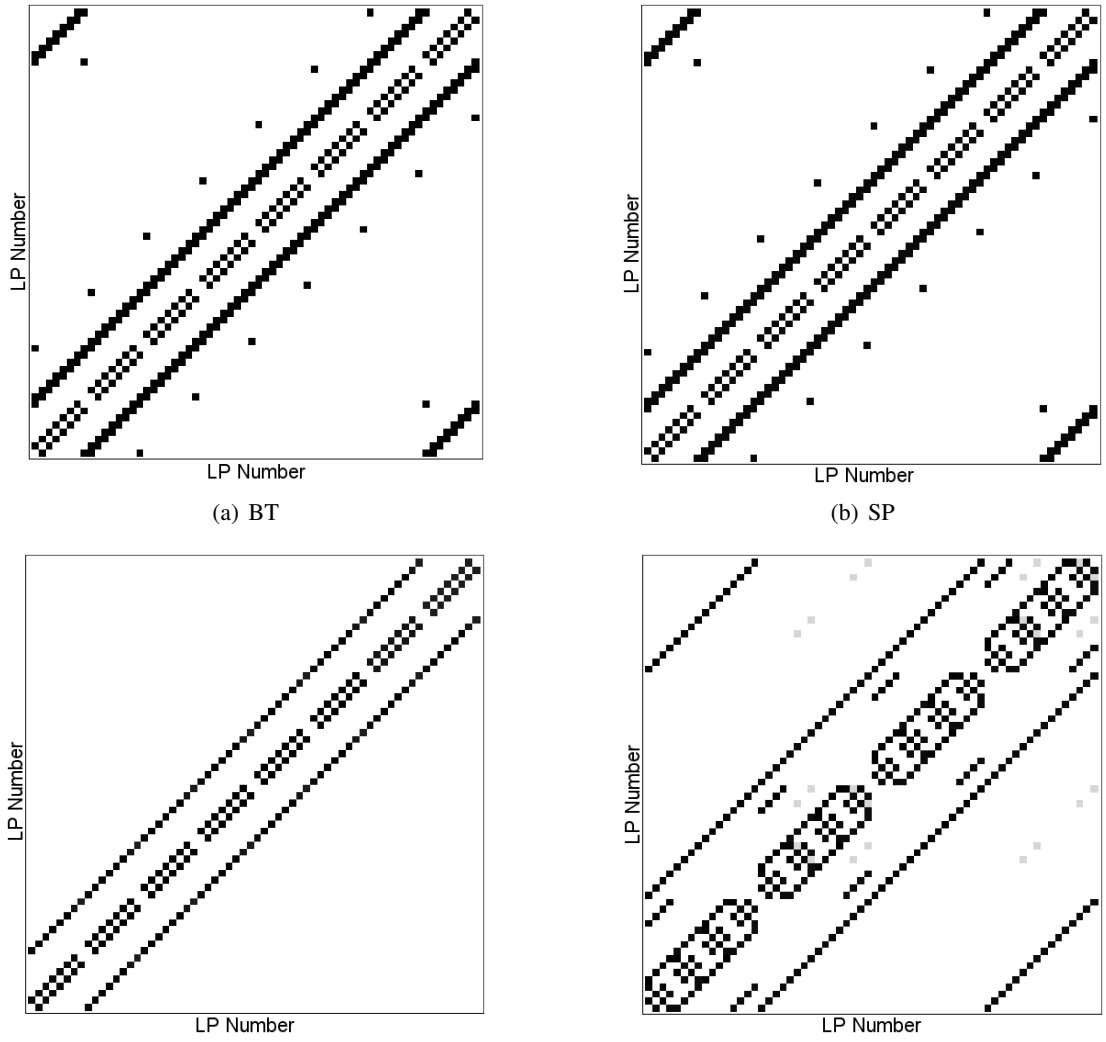


Figure 15. Communication Graphs of Four Selected NPB Applications (Problem Size B, 64 LPs)

However, when the number of logical processes is large, the overhead of identifying graph isomorphism is time-consuming.

As further work, we are adding temporal, spatial and volume locality [3, 11] to the pool of communication properties. This enables us to investigate, for a wider range of communication properties, the effect of weights in the communication correlation coefficient on effectiveness of similarity detection.

Acknowledgments

This work is supported in part by Sun Microsystems, Inc. and the National University of Singapore under grant number R252-000-298-720. The Opteron x64 system and Sun 6800 system are provided by the Hillsboro SSC (Sun Solution Center).

References

[1] Integrated performance monitoring (IPM). `http://ipm-hpc.sourceforge.net`.

[2] A low-level communication library for Java HPC. `http://www.hpjava.org`.

[3] A. Afsahi and N. J. Dimopoulos. Efficient communication using message prediction for clusters of multiprocessors. *Concurrency and Computation: Practice and Experience*, 14(10):859–883, Aug. 2002.

[4] D. Bailey, E. Barszcz, J. Barton, D. Browning, R. Carter, L. Dagum, R. Fatoohi, S. Fineberg, P. Frederickson, T. Lasinski, R. Schreiber, H. Simon, V. Venkatakrishnan, and S. Weeratunga. The NAS parallel benchmarks. Technical Report RNR-94-007, NASA Advanced Supercomputing (NAS), Mar. 1994.

[5] H. Bunke, P. Foggia, C. Guidobaldi, C. Sansone, and M. Vento. A comparison of algorithms for maximum common subgraph on randomly connected graphs. *Proc. of the ACM SIGPLAN Symp. on Principles and Practice of Parallel Programming*, pages 123–132, Windsor, Ontario, Canada, Aug. 2002. Springer Verlag.

[6] W. Jiunn Tsaur and S. Jinn Horng. Auditing causal relationships of group multicast communications in group-oriented distributed systems. *The Journal of Supercomputing*, 18(1):25–45, Jan. 2001.

[7] R. A. Johnson and D. W. Wichern, editors. *Applied Multivariate Statistical Analysis*. Prentice Hall, Dec. 2001.

[8] D. J. Johnston, M. Fleury, M. Lincoln, and A. C. Downton. Performance of parallel communication and spawning primitives on a Linux cluster. *Cluster Computing*, 9(4):375–384, Oct. 2006.

[9] S. Kamil, J. Shalf, L. Oliker, and D. Skinner. Understanding ultra-scale application communication requirements. *Proc. of the IEEE Intl. Symp. on Workload Characterization*, pages 178–187, Austin, TX, USA, Oct. 2005. IEEE Computer Society Press.

[10] D. E. Keyes, editor. *A Science-Based Case for Large-Scale Simulation*. DOE, June 2003.

[11] J. Kim and D. J. Lilja. Characterization of communication patterns in message-passing parallel scientific application programs. *Proc. of the 2nd Intl. Works. on Network-Based Parallel Computing: Communication, Architecture, and Applications*, pages 202–216, Las Vegas, Nevada, USA, Jan. 1998. Springer-Verlag.

[12] A. Massaro and M. Pelillo. Matching graphs by pivoting. *Pattern Recognition Letters*, 24(8):1099–1106, May 2003.

[13] J. J. McGregor and P. Willett. Use of a maximum common subgraph algorithm in the automatic identification of ostensible bond changes occurring in chemical reactions. *Journal of Chemical Information and Computer Sciences*, 21(3):137–140, Aug. 1981.

[14] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, editors. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, Oct. 1992.

[15] J. W. Raymond and P. Willett. Maximum common subgraph isomorphism algorithms for the matching of chemical structures. *Journal of Computer-Aided Molecular Design*, 16(7):521–533, July 2002.

[16] D. A. Reed, editor. *Workshop on the Road Map for the Revitalization of High-End Computing*. Computing Research Association, June 2003.

[17] J. Shalf, S. Kamil, L. Oliker, and D. Skinner. Analyzing ultra-scale application communication requirements for a reconfigurable hybrid interconnect. *Proc. of the 2005 ACM/IEEE Conf. on Supercomputing*, page 17, Seattle, WA, USA, Nov. 2005. IEEE Computer Society Press.

[18] H. Simon, W. Kramer, W. Saphir, J. Shalf, D. Bailey, L. Oliker, M. Banda, C. W. McCurdy, J. Hules, A. Canning, M. Day, P. Colella, D. Serafini, M. Wehner, and P. Nugent. Science-driven system architecture: A new process for leadership class computing. *Journal of the Earth Simulator*, 2:2–10, Mar. 2005.

[19] J. S. Vetter and M. O. McCracken. Statistical scalability analysis of communication operations in distributed applications. *Proc. of the ACM SIGPLAN Symp. on Principles and Practice of Parallel Programming*, pages 123–132, Snowbird, Utah, USA, June 2001. ACM Press.

[20] J. S. Vetter and F. Mueller. Communication characteristics of large-scale scientific applications for contemporary cluster architectures. *Journal of Parallel and Distributed Computing*, 63(10):853–865, Oct. 2003.

[21] J. S. Vetter and A. B. Yoo. An empirical performance evaluation of scalable scientific applications. *Proc. of ACM/IEEE Conf. on Supercomputing*, pages 1–18, Baltimore, Maryland, USA, Nov. 2002. ACM Press.

[22] A. Yamaguchi, K. F. Aoki, and H. Mamitsuka. Finding the maximum common subgraph of a partial k-tree and a graph with a polynomially bounded number of spanning trees. *Information Processing Letter*, 92(2):57–63, Oct. 2004.

[23] R. Zamani and A. Afsahi. Communication characteristics of message-passing scientific and engineering applications. *Proc. of Intl. Conf. on Parallel and Distributed Computing Systems*, pages 644–649, Phoenix, AZ, USA, Nov. 2005. IASTED/ACTA Press.