Topology-aware Optimization of Communications for Parallel Matrix Multiplication on Hierarchical Heterogeneous HPC Platforms

Tania Malik, Vladimir Rychkov, Alexey Lastovetsky
School of Computer Science and Informatics
University College Dublin
Belfield, Dublin 4, Ireland
tania.malik@ucdconnect.ie, {vladimir.rychkov, alexey.lastovetsky}@ucd.ie

Jean-Noël Quintin
Extreme Computing R&D
Bull SAS
Paris, France
jean-noel.quintin@bull.net

***Abstract*—Communications on hierarchical heterogeneous HPC platforms can be optimized based on topology information. For MPI, as a major programming tool for such platforms, a number of topology-aware implementations of collective operations have been proposed for optimal scheduling of messages. This approach improves communication performance and does not require to modify application source code. However, it is applicable to collective operations only and does not affect the parts of the application that are based on point-to-point exchanges. In this paper, we address the problem of efficient execution of data-parallel applications on interconnected clusters and present a topology-aware optimization that improves data partition by taking into account the entire communication flow of the application. This approach is also non-intrusive to the source code but application-specific. For illustration, we use parallel matrix multiplication, where the matrices are partitioned into irregular 2D rectangles assigned to different processors and arranged in columns, and the processors communicate over this partitioning vertically and horizontally. By rearranging the rectangles, we can minimize communications between different levels of the network hierarchy. Finding the optimal arrangement is NP-complete, therefore, we propose a heuristic based on evaluation of the communication flow on the given topology. We demonstrate the correctness and efficiency of the proposed approach by experimental results on multicore nodes and interconnected heterogeneous clusters.**

***Index Terms*—heterogeneous clusters; topology-aware communications; data partitioning; matrix multiplication**

I. Introduction

Modern HPC platforms are comprised of highly heterogeneous computing devices connected by complex hierarchical network. To execute data-parallel scientific applications efficiently on such platforms, it is necessary to balance the load of processors and to minimize the cost of communications. The former can be achieved by partitioning data between the processors in proportion to their speed. The latter can be achieved by reducing the number and volume of communications, by optimal mapping of the data to the processors, and by optimal scheduling of communications. In this work, we optimize the communication performance, assuming that the data have been optimally partitioned between the processors so that the total volume of communicated data is minimized.

Communications between the processes of parallel applications executed on heterogeneous platforms involve multiple message hops, non-optimal routes and traffic congestion, which significantly affect performance. Communication operations can be optimized if topology information is available. Indeed, topology information has been used for optimal scheduling of messages in MPI collective communication operations on heterogeneous HPC platforms [1], [2], [3]. However, parallel applications based on point-to-point exchanges need another solution, which could take into account the application communication flow. If all point-to-point communications are performed over a virtual topology of processes, it can be optimally mapped onto physical topology of processors, and this will minimize communication cost of the application [4], [5], [6].

In this work, we target dedicated heterogeneous HPC platforms with two-level network hierarchy, such as interconnected clusters. The processors of such platforms are connected by a network that can be represented as a two-level rooted tree with faster communications within sub-trees and slower communications between. This topology information can be taken into account, when the application data is mapped to the processors, in order to minimize message hops and reduce network contention.

To optimize communications in scientific data-parallel MPI applications, we take into account both topology information and application communication flow. Performance of data-parallel applications, especially those designed for heterogeneous platforms, highly depends on balanced workload, which is achieved by partitioning the data in proportion to the speed of processors. In its turn, heterogeneous data partitioning affects the application communication flow and has to be taken into account in topology-aware optimization. Assuming that the workload is balanced among the processors, we propose to rearrange the given heterogeneous data partition in order to reduce the number of message hops and network contention. This rearrangement is based on network topology and communication flow of the application. This approach is also non-intrusive to the source code but application-specific.

As a case study, we consider a parallel matrix multiplication application for heterogeneous platforms that is based on the Scalable Universal Matrix Multiplication Algorithm (SUMMA) [7]. SUMMA is designed for homogeneous multiprocessors and implemented using MPI. It distributes the workload evenly between the processors, mapping dense matrices onto a 2D grid of processors. Each processor receives one rectangle of matrices and participates in two MPI communicators that combine all processors in the same row and column. The communication flow consists of multiple broadcasts of matrix elements over these communicators. If SUMMA is executed on a hierarchical network of processors, its performance may be lower than expected due to higher communication cost. When network topology is known, this problem can be solved by using topology-aware broadcasts.

In [8] and [9], SUMMA was adapted for heterogeneous platforms, with matrices being partitioned into irregular 2D rectangles in proportion to the speed of processors. The rectangles, and hence the processors, are arranged in columns. In columns, the processors communicate the same way as in the original algorithm. In horizontal direction, the partitioning, and hence the communication flow, is irregular. Usually, irregular communications between processors are implemented via point-to-point operations. Non-blocking point-to-point operations additionally allow for overlapping communications and computations, which can significantly improve the performance of heterogeneous algorithms. For parallel applications based on point-to-point exchanges, like heterogeneous SUMMA, there has been no solution proposed yet that could use topology information to minimize communication cost.

In this work, we propose to rearrange the rectangles of the matrix partition in order to minimize communications between different levels of the network hierarchy. Finding the optimal arrangement is an NP-complete combinatorial optimization problem, therefore, it can be solved by using heuristics. We propose a heuristic based on evaluation of the application communication flow on the given network topology. We demonstrate the accuracy and efficiency of the proposed solution on experiments with interconnected clusters.

The rest of paper is organized as follows. In Section II, we briefly describe parallel matrix multiplication algorithms for heterogeneous platforms, and then we overview existing approaches in topology-aware optimization of communications for MPI applications. In Section III, we formulate the problem of topology-aware optimization of communications in parallel matrix multiplication in terms of application communication flow and network topology. In Section IV, we present a heuristic that finds the communication-optimal matrix partition. In Section V, we evaluate the heuristic in experiments on computing clusters with two-level network hierarchy.

II. Related Work

In this section, we describe parallel matrix multiplication algorithms based on SUMMA, with the emphasis on their communication flow. We consider these algorithms because of their applicability to a wide range of HPC platforms. These algorithms can be executed on the platforms that do not form a 2D grid of processors. The workload in these algorithms can be balanced by irregular matrix partitioning, proportional to the speed of processors. The volume of communications can be minimized. Communication performance of parallel matrix multiplication on modern hierarchical HPC platforms can be improved further by taking into account information about network topology. However, to the best of the authors' knowledge, all existing modifications of SUMMA are topology unaware.

In this section, we also overview related work on topology-aware optimization of communications. In this area, the main directions are topology-aware MPI collectives and virtual MPI topologies. These approaches are quite generic and applicable to certain classes of parallel applications (which are based on collective operations) and HPC platforms (for example, BlueGene/L). However, they cannot be applied to heterogeneous matrix multiplication algorithms. In this work, we formulate the problem of optimization of communications in terms of application communication flow and network topology.

A. *Parallel Matrix Multiplication on Heterogeneous Platforms*

The Scalable Universal Matrix Multiplication Algorithm (SUMMA) [7] is designed for homogeneous platforms and implements parallel matrix multiplication $C = A \times B$. In this algorithm, dense matrices are partitioned over a 2D grid of processors. Each processor is a part of two MPI communicators that combine all processors in the same row and column. To take advantage of processor cache, a blocking factor, b, has been introduced, so that each matrix consists of $b \times b$ blocks. The algorithm iterates over the columns of blocks of matrix A and over the rows of blocks of matrix B. At each iteration, a column of blocks (the pivot column), $A_{(b)}$, is broadcast horizontally, and a row of blocks (the pivot row), $B_{(b)}$ is broadcast vertically. Then, matrix C is updated on all processors in parallel: $C_i += A_{(b)} \times B_{(b)}$. At the end of each iteration, the pivot column and row move horizontally and vertically respectively.

The update operation can be performed efficiently by invoking a highly optimized general matrix multiplication (GEMM) routine, available for most HPC platforms. This operation can be considered as a computation kernel of the application because it represents the computation performance of the entire application. Fig. 1 shows the communication flow of SUMMA, which consists of the broadcasts in the row and column communicators. The broadcasts pass the pivot column and row in rings, pipelining computations and communications.

Heterogeneous modifications of SUMMA are based on the approach to optimization of linear algebra computations on heterogeneous platforms proposed in [10]. In this approach, to balance the load of heterogeneous processors, the matrices are partitioned into uneven rectangles such that faster processors will process larger rectangles. Ideally, this way each processor should receive the workload proportional to its computing power. In the case of heterogeneous SUMMA, the amount of computations related to the *i*-th rectangle will be proportional

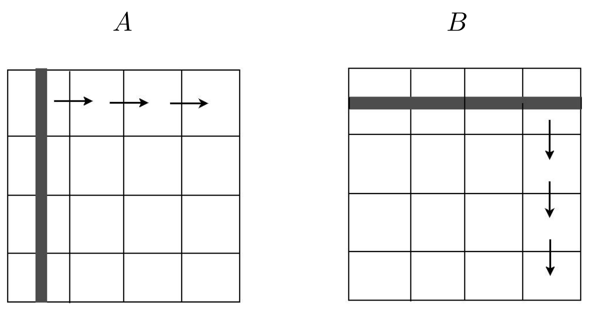

Fig. 1: Communication flow of SUMMA

to d_i, the number of blocks it contains. A number of efficient matrix partitioning algorithms have been proposed [10], [8], [11], [12] returning rectangular matrix partitions of different shapes. However, the most popular heterogeneous matrix multiplication algorithms implement column-based partitioning, when processors are arranged into columns, and all processors in a column are allocated rectangles of the same width. The widths of all the columns sum to the width of the matrix. The heights of rectangles in a column sum to the height of the matrix. More elaborated irregular matrix partitioning, such as [13], is out of scope of this paper. The overview of the heterogeneous column-based algorithms is presented in [9]. There are two main directions the column-based algorithms evolved: minimization of the volume of communications, and data partitioning based on accurate computation performance models of processors.

In [8], the algorithm minimizing the total volume of communication was presented. It arranges the processes into columns and sets the rectangles' dimensions (m_i, n_i), using the relative cycle times of processors as input. The total volume of communication is proportional to the sum of half-perimeters $\sum_{i=1}^{p}(m_i + n_i)$. The shape and ordering of rectangles are calculated to minimize this sum. The algorithm returns the optimal number of columns, the optimal number of rectangles in each column and the optimal dimensions of rectangles. The resulting rectangles are sorted in the order of increasing area, $d_i = m_i \times n_i$, with the shape as square as possible.

Fig. 2 describes the communication flow of the heterogeneous SUMMA presented in [8], which will be denoted by BR for the rest of the text. It consists of one-to-all non-blocking

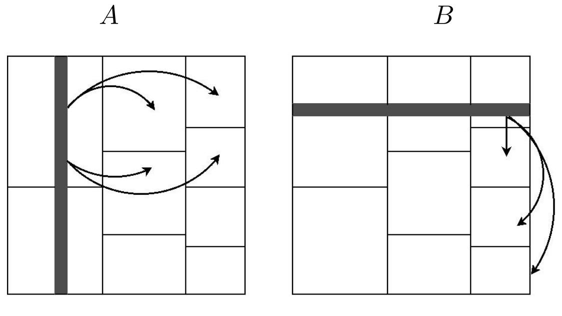

Fig. 2: Comm. flow of heterogeneous SUMMA: one-to-all

point-to-point communications in horizontal and vertical directions. In the horizontal direction, these communications are irregular: each processor holding a part of the pivot column sends multiple messages of different sizes to all processors in horizontal direction, whose rectangles are overlapped with the sender's rectangle. The size of each message is equal to the block size times the height of the overlap between the sender and receiver. In other words, the overlap is the maximum part of the pivot column required on the receiver to perform its local update operation. It should be noted that this communication pattern is not scalable if the number of communicating processors increases.

The main shortcoming of the BR algorithm is that it uses simplistic performance model of processors, where processor speed is represented by a single positive number. This approach may fail to balance the load, especially for highly heterogeneous platforms and self-adaptable applications. More reliable solution is data partitioning based on accurate performance models, such as functional performance model (FPM) [14]. It is built empirically and integrates many important features characterizing the performance of both the architecture and the application.

Under the functional performance model, the speed of each process is represented by a continuous function of problem size. The speed is defined as the number of computation units performed by the process per one time unit. The computation unit can be defined differently for different applications, but it is required not to vary during the execution of the application. For SUMMA-based matrix multiplication, it can be defined as one update of one $b \times b$ matrix block: $C_{b\times b}+ = A_{b\times b} \times B_{b\times b}$. In this case, the problem size assigned to a process is given by the number of $b \times b$ blocks. The amount of computations assigned to the process is proportional to the area of the rectangle formed by these blocks.

The processor speed is found experimentally by measuring the execution time over a range of problem sizes. This time can be found by benchmarking the full application. This benchmarking can be done more efficiently by using a serial code, the speed of execution of which is the same as that of the application but the execution time of which is significantly less. A benchmark made of one such core computation can be representative of the performance of the whole application and can be used as a kernel. The speed function of the application can be built more efficiently by timing this kernel. For SUMMA-based matrix multiplication, one update of a rectangle $C_i+ = A_{(b)} \times B_{(b)}$, implemented by highly optimized GEMM and performed many times for different pivot rows and columns, can be used as a kernel.

The problem of data partitioning using functional performance models was formulated and solved in [14]. In [9], FPM-based data partitioning was applied to the BR algorithm. We will refer to this modification of matrix multiplication as FPM-BR for the rest of the text. For FPM-BR algorithm, another communication scheme was implemented, which consists of non-blocking point-to-point communications in rings, in horizontal and vertical directions (see Fig. 3). In contrast to the

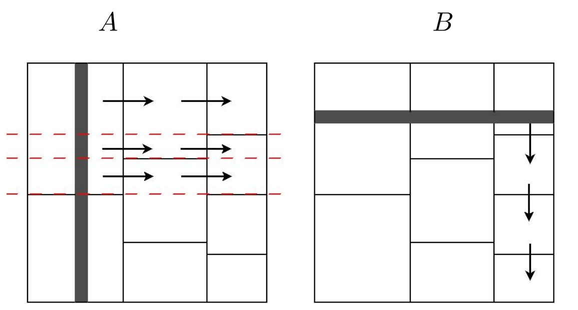

Fig. 3: Comm. flow of heterogeneous SUMMA: ring

one-to-all communication flow, each processor communicates only with the processors from its neighbouring columns and rows. In horizontal direction, the partitioning is irregular, and the processor holding the pivot row sends multiple messages to its right column. These messages can be addressed to the same processor. The size of each message is equal to the block size times the height of the overlap between all rectangles in the horizontal direction. Here the overlap is the maximum part of the pivot column that can be transmitted over the ring of processors.

Table I summarizes the above-mentioned matrix multiplication algorithms based on SUMMA. The FPM-BR algorithm better balances the workload and minimizes the total volume of communications. However, none of the algorithms takes into account the underlying networks topology, so that they are not communication-optimal. In this work, we propose to rearrange a given heterogeneous data partition in order to reduce the number of message hops and network contention.

TABLE I: Comparison of some SUMMA-based algorithms

Algorithm	Data partitioning	Comm. vol.	Comm. flow
SUMMA	homogeneous	–	broadcasts
BR	constant speeds	min	nb-p2p one-to-all
FPM-BR	speed functions	min	nb-p2p one-to-all/ring

B. *Topology-Aware Optimization of Communications*

A broad overview of optimization of communications on heterogeneous HPC platforms is given in [15], where all existing techniques were classified as performance or topology aware. The main goal of topology-aware optimization is to reduce communication traffic and contention by placing communicating tasks on physically nearby processors. Communication traffic is quantified by the number of links a message traverses. Contention is caused by multiple messages sharing a network link.

A number of topology-aware implementations of MPI collective communication operations have been proposed. In [1], two-level communication graphs were constructed for efficient execution of collective operations on interconnected clusters. Clusters communicated via selected nodes, coordinators, which formed the inter-cluster communicator. All nodes within a cluster communicated with the cluster coordinator, forming the intra-cluster communicator. These optimized implementations sent the minimal amount of data over the slow wide-area links.

In [2] and [16], collective operations were optimized for multilevel hierarchical heterogeneous networks and Grid. In [3], hierarchical approach was applied to optimize collectives for multi-core clusters: inter- and intra-node communications were overlapped, using offloading and pipelining techniques. Homogeneous supercomputers with complex network topologies, like BlueGene and Cray, can also benefit from topology-aware collectives [17].

MPI implementations try to exploit target architectures as efficiently as possible by using the most suitable communication channels and best algorithms for collective communication operations. Therefore, many existing MPI applications can be executed efficiently on hierarchical heterogeneous HPC platforms, without any modifications of the source code. However, the approach of topology-aware collectives does not address applications based on point-to-point exchanges. In this case, it is necessary to ensure that frequently communicating processes are placed close to each other. This closeness is application-specific.

The problem of topology-aware optimization of point-to-point communications can be solved by introducing a graph that represents the application communication flow and is mapped onto the network topology. In [4], this approach was applied to the mesh and graph virtual MPI topologies and SMP clusters. In [5], it was applied to the mesh topology on BlueGene/L. In [6], a tool for automatic profile-guided process placement was developed for interconnected clusters. In all these work, the heterogeneity of processors was not taken into account and therefore the processes were placed freely to processors in order to minimize the communication cost.

In our work, we focus on efficient execution of data-parallel applications on hierarchical heterogeneous HPC platforms. Their performance highly depends on balanced workload, which is achieved by partitioning the data in proportion to the speed of processors. In our case, the process placement is determined by data partitioning algorithms. Assuming that the workload is balanced among the processors, we propose to rearrange the given heterogeneous data partition in order to reduce the number of message hops and network contention. This rearrangement is based on network topology and communication pattern of the application. This approach is also non-intrusive to the source code but application-specific.

III. Communication-Optimal Matrix Partitioning

In this section, we formulate the problem of communication-optimal matrix partitioning for heterogeneous SUMMA on interconnected heterogeneous HPC clusters. To minimize communication cost, we use information about the network topology and the application communication flow.

In our target platform, interconnected heterogeneous HPC clusters, the network can be represented as a two-level rooted tree with faster communications within sub-trees (clusters) and slower communications between. Within each cluster, a single network switch provides no-contention point-to-point communications, appropriately forwarding packets between sources and destinations. Inter-cluster links may be shared by multiple processors from different clusters communicating with each other.

Our goal is to minimize communication cost of the parallel application that implements the FPM-BR matrix multiplication algorithm. In this application, each processor is assigned a matrix rectangle of the area and shape that balance the workload and minimize the communication volume. The communication flow of this application is based on non-blocking point-to-point communications in rings. Changing the position of a rectangle within the matrix does not affect the load balance and the communication volume, but the rectangles can be arranged so as to minimize the cost of communications between the processors. This forms the optimization problem we solve in this work.

Since column widths are different, we cannot move a rectangle to another column unless the whole columns are interchanged. In a column, there are no restrictions on interchanges of rectangles. All these limit the solution space of our optimization problem to a certain number of combinations. Let c be the number of columns and r_i be the number of rectangles in column i, $1 \leq i \leq c$. Then the number of combinations will be equal to the product $r_1! \times \ldots \times r_c!$. Which arrangement of rectangles is communication-optimal? This is an NP-complete problem.

We performed exhaustive search by running the application with all possible arrangements of rectangles on a small platform of three interconnected heterogeneous clusters. Each cluster consisted of several heterogeneous nodes, which were assigned rectangles proportional to their speed. From exhaustive search, we found several arrangements that reduced (Fig. 4) and increased (Fig. 5) the communication cost (different colors/fillings correspond to different clusters). We observed some regularity in the communication-optimal arrangements, which was related to the topology. In the optimal arrangements, the rectangles were grouped by clusters, whereas, in the worst cases, the rectangles assigned to the same cluster were dispersed vertically and horizontally. With the optimal arrangements, the application, which is based on non-blocking point-to-point communications in rings, performs less inter-cluster communications in horizontal and vertical directions.

The factorial design of the exhaustive search leads to a large number of trials, which becomes infeasible for large platforms. If topology information is available, we can avoid exhaustive search by applying some heuristic that efficiently finds a near-optimal arrangement. Heuristic search requires to estimate the communication cost incurred by each data partitioning. Communication cost can be estimated by taking into account the application communication flow and the network topology. Using the observations from the exhaustive search, we propose a cost function for the FPM-BR matrix multiplication with the ring communication flow and two-level network hierarchy. The main goal of this function is to characterize the number and volume of inter-cluster communications incurred by an arrangement of matrix rectangles. This function should monotonically increase with the increase of the number and the volume of inter-cluster communications.

In the FPM-BR-ring algorithm, the point-to-point communications in the vertical direction are related to matrix B (see Fig. 3). The volume of communications in each column is proportional to the column width. The number of communicating clusters in the vertical direction remains the same for any arrangement of matrix rectangles. The number of inter-cluster communications is proportional to the number of message hops between clusters. In the communication-optimal arrangements, the rectangles are grouped by clusters in each column. In this configuration, the number of message hops between clusters is minimal in each column. In the worst cases, the rectangles belong to the same group are dispersed.

To estimate the inter-cluster communication cost, we take the upper bound of the number of hops made to send the pivot row over the ring in the column. The rightmost column of the optimal arrangement in Fig. 6 illustrates the upper bound of the number of hops. Namely, when the pivot row is on the top of the matrix, there is only one communication between clusters: between the processors holding the second and third rectangles. The same happens when the pivot row is in the third rectangle: the part of pivot row is sent between the processors from different clusters that hold the fourth and first rectangles. In other cases, when the pivot row is in second and fourth rectangles, two inter-cluster communications are performed.

We define the cost function for the inter-cluster communications related to matrix B as follows: $cost_B = \sum_{i=1}^{c} h(i) \times v(i)$, where variable i iterates over the columns of matrix rectangles, functions h and v return the number of inter-cluster commu-

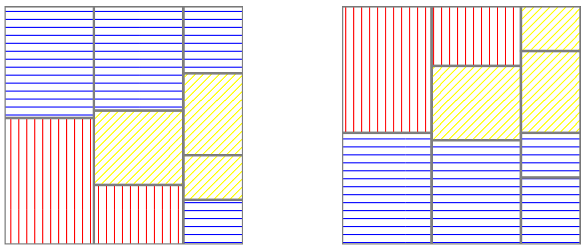
Fig. 4: Some of the communication-optimal arrangements

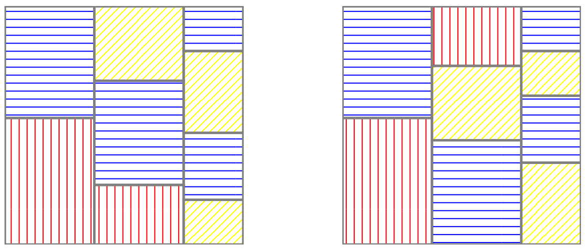
Fig. 5: Some of the worst case arrangements

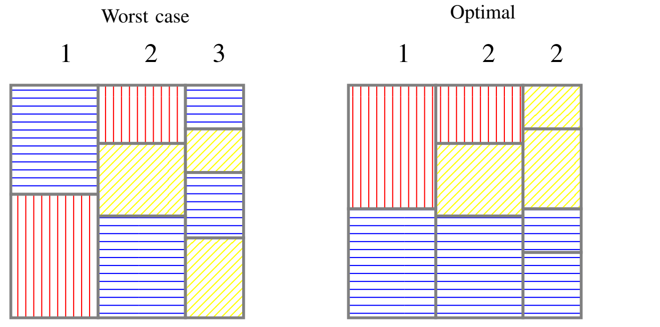

Fig. 6: Inter-cluster communications related to matrix B

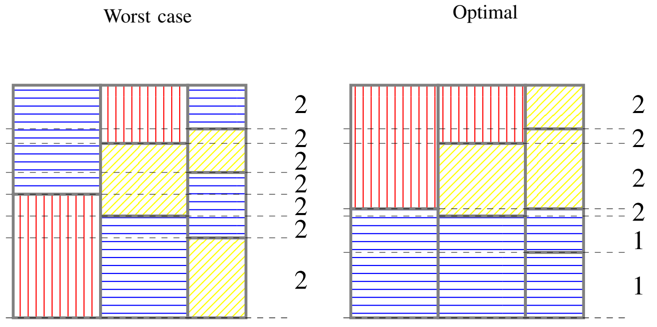

Fig. 7: Inter-cluster communications related to matrix A

nications in a column and the column width respectively. This estimate is equivalent to the integral of h over the space of columns of matrix $b \times b$ blocks: $cost_B = \int h(x)\,\mathrm{d}x$, where variable x iterates over columns of matrix blocks. The cost of the arrangements in Fig. 6 is calculated as follows:
Worst case: $(1 \times 12) + (2 \times 12) + (3 \times 9) = 63$
Optimal: $(1 \times 12) + (2 \times 12) + (2 \times 9) = 54$

The point-to-point communications in the horizontal direction are related to matrix A (see Fig. 3). The number of communicating clusters and the volume of inter-cluster communications depend on the arrangement. The number of inter-cluster communications along the pivot column varies. The volume of inter-cluster communications is proportional to the height of overlaps of matrix rectangles. The overlap is the maximum part of the pivot column that can be transmitted over the ring of processors in the horizontal direction. Fig. 7 illustrates both the numbers of inter-cluster communications in overlaps and the heights of overlaps. In the optimal arrangement, the rectangles assigned to the same cluster are grouped in rows as much as possible, while in the worst case, they are scattered over the matrix.

Similarly to the communications related to matrix B, we use the upper bound of the number of inter-cluster communications. For example, in the optimal arrangement in Fig. 7, the number of inter-cluster communications over the upper part of matrix A varies from one to two, depending on the location of the pivot column. If the pivot column is in the second column of matrix rectangles, there will be two inter-cluster communications in two top rings.

We define the cost function for the inter-cluster communications related to matrix A as follows: $cost_A = \sum_{i=1}^{o} h(i) \times v(i)$, where variable i iterates over the o overlaps of matrix rectangles, functions h and v return the number of inter-cluster communications in an overlap and the height of the overlap. This estimate is equivalent to integral of h over the space of rows of $b \times b$ matrix blocks: $cost_A = \int h(x)\,\mathrm{d}x$, where variable x iterates over rows of matrix blocks. The cost of the arrangements in Fig. 7 is calculated as follows:
Worst case: $2 \times (11 + 3 + 3 + 3 + 4 + 2 + 6) = 64$
Optimal: $1 \times (6 + 8) + 2 \times (1 + 9 + 2 + 6) = 50$

To conclude, the inter-cluster communication cost associated with arrangement M is represented by two values $(cost_A(M), cost_B(M))$, constituting a point in Euclidean space. The problem of finding the communication-optimal arrangement can be formulated as minimization of the Euclidean norm: $\|(cost_A(M), cost_B(M))\| \to \min$. This norm represents a combined cost and can be used to compare any two arrangements. The combined costs of the above arrangements are $\sqrt{64^2 + 63^2} = 89.80$ and $\sqrt{50^2 + 54^2} = 73.59$ respectively. Table II summarizes the execution time and the inter-cluster communication cost for the worst and optimal cases.

In the next section, we show how these intuitive and based on the observations cost functions can be used in a heuristic solution of the combinatorial problem of topology-aware optimization of communication cost in the heterogeneous matrix multiplication application.

TABLE II: Exhaustive search experimental results

	Cost		Exec time (sec)	
	Worst case	Optimal	Worst case	Optimal
Exhaustive search	89.80	73.59	6.00	2.78

IV. Heuristic Search

In this section, we use information about the network topology and the application communication flow in a heuristic that efficiently constructs a near-optimal arrangement. Our heuristic does not require to run the application to collect information about its communication performance. The main idea of our heuristic approach is to reduce the search space of rectangle arrangements and find the one that minimizes the inter-cluster communication cost of the application.

Our heuristic can be summarized as follows. First, we group the rectangles assigned to the same subnetwork in columns. This will minimize the inter-cluster communication cost related to matrix B. Then, we rearrange the groups of rectangles in columns to minimize the inter-cluster communications related to matrix A. Let us present the rationale for such a solution and describe the solution in detail.

A. *Rationale*

Finding the optimal arrangement is complicated by irregularity of communications over rows, which is related to matrix A. We propose to apply cost function $cost_A$ not to the whole matrix but to some of its columns. In such a way, $cost_A(A_1, \ldots, A_i)$ estimates the cost of communications between the first i columns of rectangles. Here A_i is the i-th column of matrix rectangles. We will construct the near-optimal arrangement by minimizing this cost function for successive submatrices that consist of two, three or more columns of rectangles: $(A_1, A_2), (A_1, A_2, A_3), \ldots$.

Let us assume that the rectangles in the first $i-1$ columns have been rearranged to minimize the cost: $cost_A(A_1, \ldots, A_{i-1}) = \min$. With these columns fixed, we can estimate the cost of i columns, with different permutations of rectangles in the i-th column. The permutation providing the minimal combined cost can be added to the solution. This approach reduces the number and volume of inter-cluster communications but does not guarantee finding a global minimum. It allows us to test a significantly smaller number of combinations of rectangles, which is equal to the sum (not the product) of permutations: $r_2! + \ldots + r_c!$, where r_i is the number of rectangles in column i.

We observed that in communication-optimal arrangements the matrix rectangles assigned to the same network subtree were grouped (Fig. 4). Indeed, this minimizes the number of hops between subnetworks, $g(i)$, in each column i, and therefore, reduces the communication cost related to matrix B, $cost_B$. If the rectangles in columns are grouped, we will have to estimate $cost_A(A_1, \ldots, A_i)$ for significantly less number of combinations. For g_i communicating clusters in column i, there will be $g_i!$ permutations of the grouped matrix rectangles. The number of combinations of groups $g_i!$ is significantly smaller than the number of combinations of individual rectangles $r_i!$.

In one of the optimal cases, namely, in the left picture of Fig. 4, one group of rectangles in the third column is split between the top and bottom of the column. In this case, the upper bound on the number of inter-cluster communications remains minimal, two, providing better communication performance of the parallel matrix multiplication application. Consideration of such cases significantly increases the search space and complicates the construction of the near-optimal arrangement. We exclude such cases from the search and only test permutations of the non-split groups of rectangles in each column. Nevertheless, our heuristic can find arrangements close to the one in the right picture of Fig. 4.

B. Formal Description

Our heuristic is summarized in Algorithm 1. First, in each column, we group the rectangles by subnetworks. We denote each permutation of the groups in a column i as A_i^k, $k = 1 \ldots g_i!$, and the permutation with the minimum submatrix cost as A_i^*, $cost_A(A_1, \ldots, A_i^*) = \min$. Let us show how the near-optimal arrangement is constructed by selecting the optimal permutations for each column. For the submatrix consisting of only one column (A_1), we have nothing to test because there are no communications in the horizontal direction. Therefore, we accept this column as the optimal permutation ($A_1^* := A_1$) and add it in the resulting arrangement.

Algorithm 1 Heuristic for the communication-optimal arrangement

```
for each column $i := 1$ to $c$ do
    group rectangles by clusters $\rightarrow g_i$ groups
end for
$A_1^* := A_1$
for each column $i := 2$ to $c$ do
    generate group permutations $\rightarrow A_i^1, \ldots, A_i^{g_i!}$
    for each permutation $k := 1$ to $g_i!$ do
        find $k$ such that $cost_A(A_1^*, \ldots, A_{i-1}^*, A_i^k) = \min$
    end for
    $A_i^* := A_i^k$
end for
```

Let us assume that we have found the optimal permutations in the first $i-1$ columns, and hence $cost_A(A_1^*, \ldots, A_{i-1}^*) = \min$. We add another column of rectangles and estimate the communication cost for the extended submatrix, trying different permutations A_i^k. The permutation with the minimal cost, A_i^*, such that $cost_A(A_1^*, \ldots, A_{i-1}^*, A_i^*) = \min$, is added to the resulting arrangement. We repeat this step for all columns of rectangles. The final arrangement will significantly minimize the communication cost of parallel matrix multiplication.

In total, this heuristic requires to test $g_2! + \ldots + g_c!$ arrangements of submatrices. This is significantly smaller than the solution space of the exhaustive search, which is equal to the product of the numbers of permutations of rectangles in each column $r_1! \times \ldots \times r_c!$. In addition, this heuristic does not require to run the application or any benchmarks to compare the communication cost of the application for different arrangements. Instead, it uses the information about network topology and application communication flow.

By minimizing the cost, this algorithm reduces the number and volume of inter-cluster communications. However, it does not guarantee finding the global minimum, and therefore, it provides only some near-optimal solution. By fixing the communication-optimal submatrices, we reduce the search space but may loose the optimal solution. A_i^*, the intermediate result of the search, may change if we rearrange groups in one of the first $i-1$ columns, and this configuration all together may be a better solution. Nevertheless, for the small platform used in Section III, the result of the heuristic search coincided with the communication-optimal arrangement found by the exhaustive search.

V. Experimental Results

In this section, we demonstrate that, with the near-optimal arrangement found by the heuristic, the communication performance of the heterogeneous matrix multiplication application can be significantly improved.

In our experiments, we used FuPerMod, a software tool for optimal data partitioning on dedicated heterogeneous HPC platforms [18]. In addition to the programming interface for

TABLE III: Hardware specifications

Cluster	Site	Processor	Cores	Memory
Edel	Grenoble	2.27 GHz Xeon	8	24GB
Sol	Sophia	2.6 GHz Opteron	4	4GB
Stremi	Reims	1.7 GHz Opteron	24	48GB
Paradent	Rennes	2.5 Ghz Xeon	8	32GB
Taurus	Lyon	2.3GHz Xeon	12	32GB
Chimint	Lille	2.4 GHz Xeon	8	16GB
Chinqchint	Lille	2.83 GHz Xeon	8	8GB

balancing the computational workload in data-parallel scientific applications, this tool provides the heterogeneous matrix multiplication algorithm that we referred to as FPM-BR in Section II-A. We improve the communication performance of this algorithm on two-level network hierarchy, using topology information. We compare the execution time of this algorithm with the topology-unaware data partitioning and with the communication-optimal data partitioning obtained from the heuristic search.

We performed experiments on Grid'5000 infrastructure in France, which consists of ten sites, geographically distributed and interconnected using Renater network. Within each site there are different clusters. Table III shows the specification of the experimental platform. Each experiment was carried out on a set of clusters from different sites. We had a priori information about the network topology. To increase heterogeneity, we used different numbers of threads on each node. The execution time was the mean of 30 runs.

A. Inter-Cluster Experiments

In these experiments, we use four clusters with different numbers of nodes. We spawn one MPI process per node, with different numbers of threads to increase the heterogeneity. The block size is 64 in all experiments. Table IV shows the inter-cluster communication cost and the total execution time for the following experiments:

- *16 nodes:* We start experiment with 16 nodes, 4 nodes on "Edel" cluster, 5 nodes on "Taurus" cluster, 3 nodes on "Sol" cluster and 4 nodes on "Chimint" cluster. The problem size is 16384, which is the area of a square matrix 128×128, given in the number of $b \times b$ blocks.
- *32 nodes:* In another set of experiment, we use 32 nodes in total, 9 nodes on "Chinqchint" cluster, 8 nodes on "Edel" cluster, 9 nodes on "Paradent" cluster and 6 nodes on "Taurus" cluster. The problem size is 32400. Fig. 8 shows the original topology-unaware data partitioning and the arrangement return by the heuristic.

TABLE IV: Inter-cluster experimental results

Nodes	Cost		Ratio	Exec time (sec)		Ratio
	Orig	Heuristic		Orig	Heuristic	
16	533	432	1.23	58.00	42.58	1.36
32	868	710	1.22	119.30	88.30	1.35
90	1719	1263	1.36	400.80	297.83	1.34

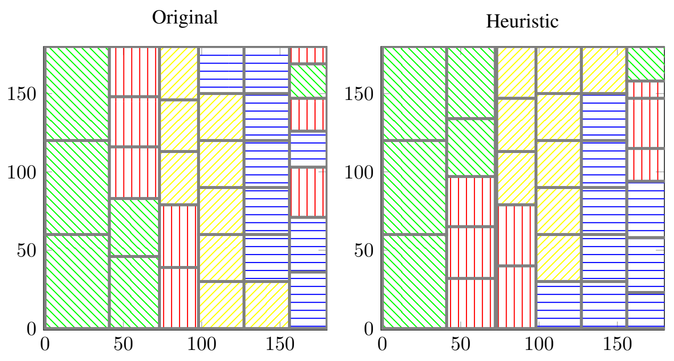

Fig. 8: Matrix partitioning for four interconnected clusters of heterogeneous processors: 32 nodes

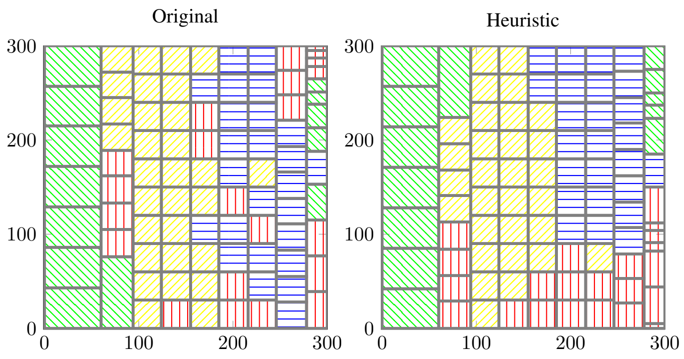

Fig. 9: Matrix partitioning for four interconnected clusters of heterogeneous processors: 90 nodes

- *90 nodes:* we extend the experiments with 90 nodes on four clusters, 25 nodes on "Chinqchint" cluster, 22 nodes on "Stremi" cluster, 29 nodes on "Paradent" cluster and 13 nodes on "Taurus" cluster. The problem size is 90,000. Fig. 9 shows the original topology-unaware data partitioning and the heuristic solution.

Experimental results show that the arrangement found by the proposed heuristic improves the total execution time by more than 30%. As expected, the rectangles assigned to the same cluster are grouped in columns and, as much as possible, in rows, which minimizes the inter-cluster communications. In horizontal direction, the heuristic minimizes not only the number of communications between clusters but also the number of communicating clusters. Indeed, the topology-unaware partitions in both pictures result in communications between all four clusters, which are performed in rings along all rows of matrix A. With the heuristic-based partitions, three clusters in average are involved in the ring communications.

B. Homogeneous Inter-Node Experiments

In parallel matrix multiplication algorithms for homogeneous platforms, matrices are usually partitioned into a Cartesian grid. If the number of processors is not equal to a square of some integer number, such partitioning will result in the non-square shape of matrix blocks. In this case, the volume

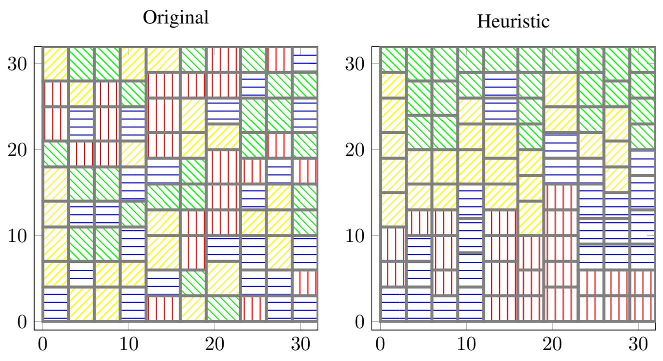

Fig. 10: Matrix partitioning for four homogeneous multi-core nodes, with 24 MPI processes per node

TABLE V: Homogeneous inter-node experimental results

Nodes	Cost		Ratio	Exec time (sec)		Ratio
	Orig	Heuristic		Orig	Heuristic	
4	336	199	1.68	3.85	3.17	1.21

of communications is not optimal. It is minimized in such SUMMA-based algorithms as BR.

We performed experiments on a homogeneous multi-core cluster, which is characterized by faster communications between the processes running on the same node and slower communications between the processes running on different nodes. Each node of the Stremi cluster has 24 cores. We spawned 24 MPI processes on four nodes. Problem size is 1024 and block size is 64. Table V shows the communication cost and the total execution time of the application with different arrangements of matrix rectangles. Fig. 10 shows how matrices are partitioned by the BR algorithm and our heuristic. These results show that the proposed topology-aware optimization technique can improve the communication performance on homogeneous platforms as well.

VI. Conclusion

In this paper, we presented the heuristic approach aimed to minimize communication cost of heterogeneous matrix multiplication using information about network topology and application communication flow. Our heuristic approach provides efficient near-optimal solution of the combinatorial problem of mapping the application communications onto the network topology. We validate this approach with experiments on Grid5000.

Acknowledgment

Experiments were carried out on Grid'5000 developed under the INRIA ALADDIN development action with support from CNRS, RENATER and several Universities as well as other funding bodies (see https://www.grid5000.fr). We are grateful to Arnaud Legrand who provided sample code for minimising the total volume of communication.

References

[1] T. Kielmann, R. F. Hofman, H. E. Bal, A. Plaat, and R. A. Bhoedjang, "MagPIe: MPI's collective communication operations for clustered wide area systems," in *ACM Sigplan Notices*, vol. 34, no. 8. ACM, 1999, pp. 131–140.

[2] N. Karonis, B. De Supinski, I. Foster, W. Gropp, E. Lusk, and J. Bresnahan, "Exploiting hierarchy in parallel computer networks to optimize collective operation performance," in *Parallel and Distributed Processing Symposium, 2000. IPDPS 2000. Proceedings. 14th International*, 2000, pp. 377–384.

[3] T. Ma, G. Bosilca, A. Bouteiller, and J. Dongarra, "HierKNEM: An adaptive framework for kernel-assisted and topology-aware collective communications on many-core clusters," in *Parallel Distributed Processing Symposium (IPDPS), 2012 IEEE 26th International*, 2012, pp. 970–982.

[4] J. Traff, "Implementing the MPI process topology mechanism," in *Supercomputing, ACM/IEEE 2002 Conference*, 2002, pp. 28–28.

[5] T. Agarwal, A. Sharma, A. Laxmikant, and L. Kale, "Topology-aware task mapping for reducing communication contention on large parallel machines," in *Parallel and Distributed Processing Symposium, 2006. IPDPS 2006. 20th International*, 2006, p. 10.

[6] H. Chen, W. Chen, J. Huang, B. Robert, and H. Kuhn, "MPIPP: An automatic profile-guided parallel process placement toolset for SMP clusters and multiclusters," in *Proceedings of the 20th Annual International Conference on Supercomputing*, ser. ICS '06. New York, NY, USA: ACM, 2006, pp. 353–360.

[7] R. A. Van De Geijn and J. Watts, "SUMMA: scalable universal matrix multiplication algorithm," *Concurrency-Practice and Experience*, vol. 9, no. 4, pp. 255–274, 1997.

[8] O. Beaumont, V. Boudet, F. Rastello, and Y. Robert, "Matrix multiplication on heterogeneous platforms," *IEEE Trans. Parallel Distrib. Syst.*, vol. 12, no. 10, pp. 1033–1051, 2001.

[9] D. Clarke, A. Lastovetsky, and V. Rychkov, "Column-based matrix partitioning for parallel matrix multiplication on heterogeneous processors based on functional performance models," in *Euro-Par 2011: Parallel Processing Workshops*, ser. LNCS. Springer Berlin Heidelberg, 2012, vol. 7155, pp. 450–459.

[10] A. Kalinov and A. Lastovetsky, "Heterogeneous distribution of computations while solving linear algebra problems on networks of heterogeneous computers," in *7th International Conference on High Performance Computing and Networking Europe (HPCN'99)*, 1999.

[11] E. Dovolnov, A. Kalinov, and S. Klimov, "Natural block data decomposition for heterogeneous clusters," in *IPDPS 2003*, April 2003.

[12] A. Lastovetsky, "On grid-based matrix partitioning for heterogeneous processors," in *6th International Symposium on Parallel and Distributed Computing (ISPDC 2007)*, IEEE Computer Society. Hagenberg, Austria: IEEE Computer Society, 5-8 July 2007, pp. 383–390.

[13] A. DeFlumere, A. Lastovetsky, and B. Becker, "Partitioning for parallel matrix-matrix multiplication with heterogeneous processors: The optimal solution," in *Parallel and Distributed Processing Symposium Workshops PhD Forum (IPDPSW), 2012 IEEE 26th International*, 2012, pp. 125–139.

[14] A. Lastovetsky and R. Reddy, "Data partitioning with a functional performance model of heterogeneous processors," *International Journal of High Performance Computing Applications*, vol. 21, no. 1, pp. 76–90, 2007.

[15] K. Dichev and A. Lastovetsky, "Optimization of collective communication for heterogeneous hpc platforms." Wiley-Interscience, 2013.

[16] C. Coti, T. Herault, and F. Cappello, "MPI applications on grids: a topology aware approach," in *Euro-Par 2009 Parallel Processing*. Springer, 2009, pp. 466–477.

[17] E. Solomonik, A. Bhatele, and J. Demmel, "Improving communication performance in dense linear algebra via topology aware collectives," in *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC '11. New York, NY, USA: ACM, 2011, pp. 77:1–77:11.

[18] D. Clarke, Z. Zhong, V. Rychkov, and A. Lastovetsky, "FuPerMod: A framework for optimal data partitioning for parallel scientific applications on dedicated heterogeneous HPC platforms," in *Parallel Computing Technologies*, ser. LNCS. Springer Berlin Heidelberg, 2013, vol. 7979, pp. 182–196.