IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received December 27, 2020, accepted January 11, 2021, date of publication January 19, 2021, date of current version February 1, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3052976

Towards Optimal Matrix Partitioning for Data Parallel Computing on a Hybrid Heterogeneous Server

TANIA MALIK AND ALEXEY LASTOVETSKY
School of Computer Science, University College Dublin, Dublin 4, D04 V1W8 Ireland
Corresponding author: Tania Malik (tania.malik@ucd.ie)

This work was supported by the Science Foundation Ireland (SFI) under Grant 14/IA/2474.

ABSTRACT Optimal partitioning of a square computational domain over several heterogeneous processors, balancing the load of the processors and minimizing the inter-processor communication cost, is crucial for data parallel dense linear algebra and other applications having similar communication pattern on modern hybrid servers. Although a solution has been found for two processors, the cases of three and more processors are still open. The state-of-the-art solution for three processors uses an approximation communication cost function which fails to accurately account for the total amount of data moved between processors, leaving thus the question of its global optimality unanswered. In this work, we formulate and solve a mathematical problem of optimal partitioning a real-valued square over three heterogeneous processors using a new cost function, which accurately accounts for the total amount of data communicated between processors. We also develop an original method for accurate experimental evaluation of the communication time of data movement between memories of the compute devices in the hybrid platform during the execution of data parallel applications. We successfully use this method in the experimental validation of our mathematical results. Finally, we propose a communication energy model predicting the dynamic energy consumption of data movement between processors and experimentally validate its accuracy. This model predicts, and the experiments confirm, that the performance-optimal partition is not necessarily energy optimal.

INDEX TERMS Data partitioning, communication optimization, non-rectangular partitioning, matrix multiplication, heterogeneous computing, performance model, data parallelism, energy model, energy of communication.

I. INTRODUCTION

The problem of matrix partitioning over heterogeneous processors originates in dense linear algebra on heterogeneous platforms. Its solution is also applicable in other application domains dealing with rectilinear computational regions, e.g., in stencil computations. The growing popularity of hybrid high-performance computing platforms, integrating CPUs and various accelerators, motivated the increased attention to this problem in the last decade.

The problem of optimal partitioning a matrix into *rectangular* submatrices is originally introduced by Kalinov and Lastovetsky [1], [2], with the objective to minimize the computation time through balancing the load of the heterogeneous processors. The communication cost is not considered in this original optimization problem and is first included in the problem by Beaumont *et al.* [3], aiming to minimize not only the computation time but also the communication cost. The latter is formalized as the sum of half-perimeters of rectangular submatrices [3], which is motivated by the communication cost of 2D parallel matrix-matrix multiplications algorithms [4]. The introduction of the communication cost made the problem NP-complete [3]. Many approximate algorithms solving the communication-aware matrix partitioning problem are proposed.

Although the communication-aware problem [3] is NP-complete when the number of processors, p, is arbitrary, it has simple exact solutions for $p = 2$ and $p = 3$. Optimal solutions of the communication-aware matrix partitioning problem for small numbers of heterogeneous processors are practically very important as they help minimize the execution time of parallel applications on hybrid servers typically

The associate editor coordinating the review of this manuscript and approving it for publication was Muhammad Imran Tariq.

integrating few compute devices. It is discovered however that these exact solutions are not globally optimal [5], [6].

Matrix partitioning for parallel matrix-matrix multiplication on two heterogeneous processors is studied by Becker and Lastovetsky [5]. The authors construct a *non-rectangular* partition with a smaller communication cost than the *rectangular* partition, when the ratio of speeds of the processors is less than 1 : 3. They also study the case of three heterogeneous processors and propose a *non-rectangular* partition, whose communication cost will be less than the communication cost of any rectangular partition for some particular ratios of the processors' speeds [6]. These results prove that the globally optimal matrix partition does not have to be rectangular, and all possible partitions should be considered as potentially optimal. This inspired research in communication-optimal load-balanced matrix partitioning over several heterogeneous processors assuming that the optimal partition can be of any shape.

The unrestricted problem of communication-optimal load-balanced matrix partitioning between two heterogeneous processors has been comprehensively studied by DeFlumere *et al.* [7], where the authors prove that for any given ratio of the speeds of the processors either the rectangular partition or the non-rectangular partition [5] will be superior to any other, arbitrary, partition. The extension of this result to the case of three processors proved hard. Application of the mathematical technique [7], developed to solve the 2-processor problem, to the case of 3 processors helped identify six candidates for optimal shapes but failed to prove their optimality [8].

The unrestricted 3-processor problem is recently revisited by Beaumont et al [9]. The authors manage to mathematically prove that three potentially optimal partition shapes from the list proposed by DeFlumere and Lastovetsky [8] are superior to any other arbitrary shape. However, the definition of the communication cost used in the mathematical formulation of the problem does not represent the total number of matrix elements moved between the processors during the execution of the motivating parallel applications. Instead, a cost function, which approximates the total amount of moved data, is introduced. This approximate function is calculated as the sum of half-perimeters of minimal rectangular submatrices, each containing all matrix elements allocated to the same processor. The use of this approximate communication cost function results in exclusion from consideration of many partitions during the proof of the optimality of the three identified shapes. Therefore, while the result of Beaumont *et al.* [9] represents an important step towards finding the globally optimal partitions over three heterogeneous processors, it still leaves the problem open. In this article, we propose a solution of this problem using a communication cost function, which accurately represents the total amount of data moved between the processors, and therefore proves the global optimality of the identified optimal partitions.

This article is an extended version of our conference paper [10]. In addition to the performance-related results, it presents a theoretical and experimental study of the communication energy cost of the performance-optimal partitions. While there is a good understanding of performance of communication and how to design sufficiently accurate performance communication models [11], very little is done in energy of communication, and very little is known about the impact of the energy cost of communication on performance/energy optimal configurations of applications. In this work, we propose an analytical model of the energy of communication for data-parallel matrix computations on three heterogeneous processors interconnected via three heterogeneous communication links. We also propose an experimental methodology for accurate measurement of the energy of data movement between the main memories of heterogeneous devices during the execution of parallel applications. We use this methodology to experimentally validate the accuracy of the proposed analytical energy model and to experimentally study the correlation between the performance optimality and energy optimality of the partitions.

The main contributions of the presented work include:

1) We propose an integer-valued cost function of an arbitrary partition of a square matrix over three heterogeneous processors, which returns the exact number of matrix elements moved between the processors during the execution of the parallel matrix-matrix multiplication algorithm. We then construct a continuous extension of this function for estimation of the communication cost of an arbitrary partition of a real-valued square over three heterogeneous processors.
2) We use the proposed accurate cost function to introduce a mathematical problem of globally optimal partitioning of a real-valued square. We then solve this problem and prove the correctness of the solution.
3) We develop an original method for accurate experimental evaluation of the time and energy of data movement between memories of the compute devices of the hybrid platform during the execution of data parallel applications. We successfully use this method in the experimental validation of our mathematical results.
4) We propose an energy model of communication, predicting the dynamic energy consumption for a partition, and design and perform experiments to validate the model. We find out that this model predicts, and the experiments confirm this prediction, that the performance-optimal partition does not have to be energy optimal.

The rest of the paper is structured as follows. Section II presents related work. Section III analyses the state of the art in optimal matrix partitioning over three processors. In Section IV, we mathematically define the exact cost communication function. In Section V, the partitioning optimization problem is formulated and solved. Section VI experimental validates the mathematical solution. Section VII proposes the energy model of communication and studies the energy cost of performance optimal partitions. In section VIII, we discuss the performance and energy

experiments results. Finally, Section IX concludes the paper and outlines some future research.

II. RELATED WORK

For data parallel applications, optimal partitioning is perhaps the utmost important challenge for their efficient execution on hybrid heterogeneous platforms. Application performance is mainly determined by how its computational domain is partitioned among heterogeneous resources of the platform. An optimal data partition would significantly reduce the application execution time.

Matrices and other rectangular computational domains are omnipresent in computational science. The problem of optimal partitioning of a matrix into rectangular submatrices is originally introduced by Kalinov and Lastovetsky [1], [2]. In this problem, heterogeneous resources were modeled according to their constant relative speed aiming to minimize the computation time through balancing the load of the heterogeneous resources. This problem has exact solutions and efficient algorithms due to the simplicity of the performance model of the heterogeneous platform.

Lastovetsky and Reddy [12] introduced the smooth functional performance models of the heterogeneous platform for optimal partitioning. A smooth functional performance model represents the speed of each processor by a smooth function of the problem size. That was a more realistic variation of the original partitioning problem, which had been extensively studied since its introduction [12]–[17]. The crucial artifact of all these optimization problems is that any optimal solution will be load balanced.

With the emergence of multi-core processors, smooth functional performance models of the heterogeneous platform became less realistic [18], [19]. Therefore, Lastovetsky and Manumachu [19] introduced a variant of the same optimization problem and used arbitrary discrete speed functions of the processors, which was further explored by Khaleghzadeh *et al.* [20]. Contrary to previous optimization problems, optimal solutions of this problem do not have to balance the load of the processors.

The execution time of a data parallel application includes both the communication time and computation time. Beaumont *et al.* [3] appeared to be the first to introduce the communication cost of the application as a deciding factor and expended the initial problem [1], which did not take into account the communication cost. They formally proposed that the optimal matrix partitioning would not only be load balanced but also minimize the overall communication volume of the application. They defined the communication cost as the sum of half-perimeters of rectangular submatrices [3], which was motivated by the communication cost of two-dimensional parallel matrix-matrix multiplication algorithms [4].

The introduction of the communication cost made the optimization problem NP-complete [3]. The first communication-aware approximation algorithm was proposed by Beaumont *et al.* [3] and had a bound ratio of 1.75. Many other approximate algorithms solving the communication-aware matrix partitioning problem and its variants have been proposed [21]–[26]. All these solutions are based on different heuristics and use different performance models (smooth or arbitrary discrete speed functions). However, they mainly focus on finding the solution that creates partitions with a rectangular shape where each processor assigned a rectangular region for computation [1], [2], [21]–[23], [27].

It was however discovered by Becker and Lastovetsky [5] that non-rectangular partitions can outperform rectangular ones. The authors relaxed the constraint of rectangular partitions for parallel multiplication of square matrices on two heterogeneous processors and proposed one non-rectangular optimal shape, which resulted in a lower number of matrix elements moved between the processors for speed ratios less than $1:3$ in comparison with the traditional rectangular partitioning. This non-rectangular partitioning allocates a square area in the top left corner of the matrix to the slower processor and the balance is allocated to the faster one. The authors also studied the case of three heterogeneous processors [6], where they proposed an optimal non-rectangular shape, called the square corner, which lowered the total volume of communication and appeared superior to any *rectangular* partitioning but again subject to certain processor speed ratios and topology requirements. In the square corner shape, for two slower processors, squares are allocated in the opposite corners of the matrix and the balance is allocated to the faster processor.

These results inspired researchers to work on the problem of communication-aware optimal matrix partitioning over heterogeneous processors with no assumptions about the optimal partitioning shapes. A novel mathematical method, referred to as the Push technique, was developed for the case of two processors [7]. By using the Push technique, it was proved [7] that either the rectangular or the non-rectangular partition [5] would always outperform any other arbitrary partition and remain optimal for any speed ratio between the processors.

The Push technique was further extended for the case of three heterogeneous processors and used to identify six potentially optimal shapes [8], [28]. Out of these six partitioning shapes, three were non-rectangular. However, the optimality of these shapes was not mathematically proven. For matrix partitioning problem, Lambert *et al.* [29] also proposed a recursive approximation algorithm for an arbitrary number of processors by relaxing the rectangular partitioning restriction. This algorithm reduced the approximation ratio to $2/\sqrt{3}$, which is the best known approximation.

The unrestricted 3-processor matrix optimization problem is recently revisited by Beaumont et al [9]. The authors manage to mathematically prove that three potentially optimal partition shapes from the list proposed by Deflumere and Lastovetsky [8] are superior to any other arbitrary shape. However, the definition of the communication cost used in the mathematical formulation of the problem does not represent the total number of matrix elements moved between

the processors during the execution of the motivating parallel applications. Instead, a cost function, which approximates the total amount of moved data, is introduced. This approximate function is calculated as the sum of half-perimeters of minimal rectangular submatrices, each containing all matrix elements allocated to the same processor. The use of this approximate communication cost function results in exclusion from consideration of many partitions during the proof of the optimality of the three identified shapes. Therefore, while the result of Beaumont *et al.* [9] represents an important step towards finding the globally optimal partitions over three heterogeneous processors, it still leaves the problem open.

For accurate measurement of the communication cost, a cost function is required which accurately accounts for all matrix elements moved between the processors. DeFlumere *et al.* [7] proposed such a cost function, counting the total number of matrix elements moved between the processors. However, using this discrete metric to mathematically prove the optimality of the identified partitioning shapes is proved to be hard. In this work, we extend this discrete metric into the continuous space to overcome this difficulty. We propose a cost function of an arbitrary partition of a real-valued square which accurately represents the total amount of data moved between the processors. Our cost function is also motivated by 2D data-parallel matrix multiplication applications [3], [4], [24], similar to the cost function of Beaumont *et al.* [9]. The matrix multiplication applications calculate the product $C = A \times B$ of two matrices A and B. The elements of A, B and C are partitioned among the processors in proportion to their relative speeds. Element c_{ij} is calculated as the dot product of i-th row of matrix A, A_i, and j-th column of matrix B, B_j. To calculate c_{ij}, all elements of A_i and B_j, which do not belong to the processor that owns c_{ij}, must be sent to this processor. Based on this observation, our cost function will precisely and accurately reflect the overall amount of data moving between processors.

III. OPTIMAL PARTITIONING A SQUARE BETWEEN THREE HETEROGENEOUS PROCESSORS: STATE OF THE ART

This section revisits the optimal partitioning problem of a square computational domain for three heterogeneous processors, where the domain is partitioned among processors in proportion to their speed to get both load balanced and communication optimal a partitioning shape. Three partitioning shapes of Fig. 1 proved to be sufficient for optimally partitioning a square in the state-of-the-art solution of this problem [9].

However, the definition of the communication cost used in the solution [9] does not represent the total number of matrix elements moved between the processors. Instead, a cost function, which approximates the total amount of moved data, is introduced. This approximate function is calculated as the sum of half-perimeters (SHP) of minimal rectangular submatrices, each containing all data elements allocated to the same processor. The partition which minimizes this cost function is considered optimal. Motivated by the communication cost

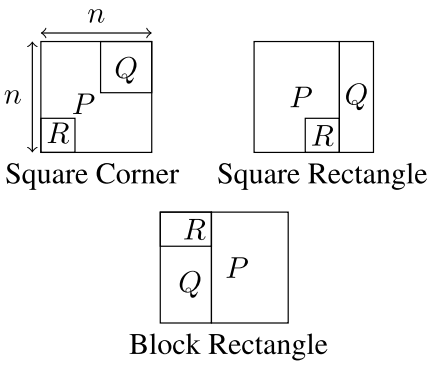


FIGURE 1. Optimal partitioning shapes of three heterogeneous processors for square computational domain. Processors Q and R have square regions in Square Corner partition whereas processor R has a square region in square rectangle partition.

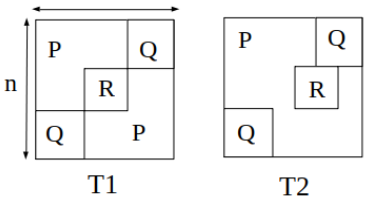
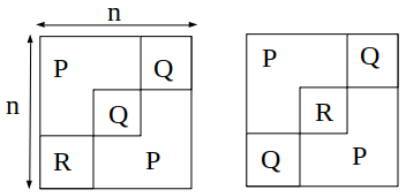


FIGURE 2. The figure shows two partitions, $T1$ and $T2$, of a square $n \times n$ matrix between three heterogeneous processors P, Q and R, with the same SHP cost but different exact communication costs $Cost(T1)$ and $Cost(T2)$. Since Q and R regions are assumed to be squares of the same size $\frac{n}{3} \times \frac{n}{3}$, we can derive that $SHP(T1) = SHP(T2) = 4n + 2\sqrt{S_R}$. However, as shown in Section V-A (Lemma 4), $Cost(T1) = 2n^2$, $Cost(T2) = 2n^2 - \frac{1}{2}\sqrt{S_R} \times n + \sqrt{S_Q} \times n$, and hence Cost(T1) < Cost(T2).

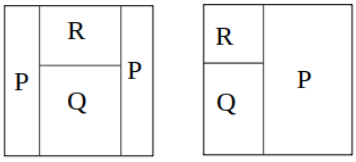
of 2D parallel matrix-matrix multiplication algorithms [4], the SHP function accurately represents the communication cost of rectangular matrix partitions but fails to distinguish the cost of many non-rectangular partitions, for which the total number of matrix elements, moved between processors, will be different. This is illustrated in Fig. 2 where two partitions with different amounts of communicated data have the same SHP cost. At the same time, many non-rectangular partitions, which are equivalent in terms of the total amount of communicated data, will have different SHP costs as illustrated in Fig. 3.

In Figures 2 and 3, the exact communication cost, $Cost$, of a given partition is calculated as the total number of matrix elements moved between processors P, Q, and R during parallel matrix-matrix multiplication, $C = A \times B$, given the matrices are identically partitioned between the processors. S_X designates the total number of elements in a region marked by X. The use of the approximate communication cost function, SHP, results in exclusion from consideration of many partitions during the proof of optimality of the three identified shapes. Therefore, while the result [9] represents an important step towards finding the globally optimal partitions over three heterogeneous processors, it still leaves the problem open.

In this work, we propose a solution of this problem using a communication cost function, which accurately represents



$SHP(T3) = 2n + 4\sqrt{S_Q} + 2\sqrt{S_R} < SHP(T4) = 4n + 2\sqrt{S_R}.$
However, as shown in Section V-A (Lemma 5),
$Cost(T3) = Cost(T4) = 2n^2.$



$SHP(T5) = 4n + (n - 2\frac{S_P}{n}) > SHP(T6) = 4n - \frac{S_P}{n}.$
However, as shown in Section V-A (Lemma 5),
$Cost(T5) = Cost(T6) = 2n^2 - S_P.$

FIGURE 3. The figure shows two pairs of partitions, (*T3*, *T4*) and (*T5*, *T6*), which are discriminated by the *SHP* cost but have the same exact communication cost, *Cost*.

the total amount of data moved between the processors, and therefore proves the global optimality of the identified optimal partitions.

IV. COST FUNCTION

This section mathematically defines the exact communication cost function of an arbitrary partition of a square $n \times n$ computational domain. We assume that between different pairs of processors, for one data unit, the communication cost is the same. The exact communication cost function is first derived in a discrete form for matrices and then, in a continuous form, for real-valued squares. All notation with bar relates to the discrete case (e.g. $\overline{f}_c$) and without a bar – to the continuous case (e.g.f_c).

A. DEFINITION OF DISCRETE COST FUNCTION

In this subsection, we derive a discrete cost function from the communication cost of parallel matrix-matrix multiplication. $C = A \times B$, of two square matrices A and B, assuming that the elements of matrices A, B and C are identically partitioned between processors in proportion with relative speeds of the processors. Element c_{ij} is calculated as the dot product of i-th row of matrix A, A_i, and j-th column of matrix B, B_j. To calculate c_{ij}, all elements of A_i and B_j, which do not belong to the processor that owns c_{ij}, must be sent to this processor. Derived from this observation, we define the communication cost of a partition of a square matrix to be equal to the total number of elements of matrices A and B moved between the processors.

Mathematically, each partition of an $n \times n$ matrix between three processors P, Q, and R, is represented by a mapping $[0, 1, .., n] \times [0, 1, .., n] \mapsto \{P, Q, R\}$, of the set of indices of the matrix into the set of processors. The set of all possible partitions is denoted as

$$\mathbb{M} = \Big\{ [1, .., n] \times [1, .., n] \mapsto \{P, Q, R\} \Big\},$$

and for each partition $M \in \mathbb{M}$ our cost function $\overline{f}_c : \mathbb{M} \mapsto \mathbb{Z}_{\geq 0}$ returns the defined communication cost, $\overline{f}_c(M)$.

This mathematical definition makes no assumption about partition shapes, including *all* possible partitions in consideration and guarantees that for any partition each matrix element is allocated to exactly one processor.

TABLE 1. Parameters of an arbitrary partition $M \in \mathbb{M}$ of a square $n \times n$ matrix used in the discrete cost function $\overline{f}_c(M)$.

Rows		Columns	
$\overline{a}_{\mathrm{P}}$:	Number of rows allocated to P	$\overline{b}_{\mathrm{P}}$:	Number of columns allocated to P
$\overline{a}_{\mathrm{Q}}$:	Number of rows allocated to Q	$\overline{b}_{\mathrm{Q}}$:	Number of columns allocated to Q
$\overline{a}_{\mathrm{R}}$:	Number of rows allocated to R	$\overline{b}_{\mathrm{R}}$:	Number of columns allocated to R
$\overline{a}_{\mathrm{PQ}}$:	Number of rows partitioned between P and Q	$\overline{b}_{\mathrm{PQ}}$:	Number of columns partitioned between P and Q
$\overline{a}_{\mathrm{PR}}$:	Number of rows partitioned between P and R	$\overline{b}_{\mathrm{PR}}$:	Number of columns partitioned between P and R
$\overline{a}_{\mathrm{QR}}$:	Number of rows partitioned between Q and R	$\overline{b}_{\mathrm{QR}}$:	Number of columns partitioned between Q and R
$\overline{a}_{\mathrm{PQR}}$:	Number of rows partitioned between P, Q and R	$\overline{b}_{\mathrm{PQR}}$:	Number of columns partitioned between P, Q and R
$\Sigma\overline{a} = n$		$\Sigma\overline{b} = n$	

B. ANALYTICAL FORMULAS FOR DISCRETE COST FUNCTION

In this subsection, we derive two analytical formulas for the cost $\overline{f}_c(M)$ of an arbitrary partition $M \in \mathbb{M}$, characterized by the parameters summarized in Table 1. For any $X \in \{P, Q, R\}$, parameter $\overline{a}_X$ is the total number of rows, all elements of which are allocated to processor X. Similarly, $\overline{b}_X$ is the total number of columns allocated to X as a whole. For any $X, Y \in \{P, Q, R\}$, parameter $\overline{a}_{XY}$ is the total number of rows, elements of which are allocated to processors X and Y only (but not to any of them as a whole). Similarly, $\overline{b}_{XY}$ is the total number of columns, partitioned between X and Y only. Parameter $\overline{a}_{PQR}$ is the number of the remaining rows, that is, those partitioned between all three processors, and $\overline{b}_{PQR}$ is the number of columns partitioned between the three processors.

Given $X \in \{P, Q, R\}$, let $\overline{A}_{PQR,X}$ be the total number of elements allocated to processor X in the $\overline{a}_{\mathrm{PQR}}$ rows, elements of which are distributed between all three processors P, Q, and R. Similarly, let $\overline{B}_{PQR,X}$ be the total number of elements allocated to processor X in the $\overline{b}_{\mathrm{PQR}}$ columns, elements of which are distributed between all three processors P, Q, and R. Then, the total number of elements moved between any pair of processors $X, Y \in \{P, Q, R\}$ can be calculated as

follows:

$$C_{XY} = (\overline{a}_{XY} \times n) + (n \times \overline{b}_{XY}) + (\overline{A}_{PQR,X} + \overline{A}_{PQR,Y}) + (\overline{B}_{PQR,X} + \overline{B}_{PQR,Y})$$

Note that

$$(\overline{A}_{PQR,P} + \overline{A}_{PQR,Q} + \overline{A}_{PQR,R}) = \overline{a}_{PQR} \times n,$$

and

$$(\overline{B}_{PQR,P} + \overline{B}_{PQR,Q} + \overline{B}_{PQR,R}) = \overline{b}_{PQR} \times n.$$

Therefore,

$$\begin{aligned}\overline{f}_c(M) &= C_{PQ} + C_{PR} + C_{QR} \\ &= (\overline{a}_{PQ} + \overline{b}_{PQ} + \overline{a}_{PR} + \overline{b}_{PR} + \overline{a}_{QR} + \overline{b}_{QR}) \times n \\ &\quad + \overline{A}_{PQR,P} + \overline{A}_{PQR,Q} + \overline{B}_{PQR,P} + \overline{B}_{PQR,Q} \\ &\quad + \overline{A}_{PQR,P} + \overline{A}_{PQR,R} + \overline{B}_{PQR,P} + \overline{B}_{PQR,R} \\ &\quad + \overline{A}_{PQR,Q} + \overline{A}_{PQR,R} + \overline{B}_{PQR,Q} + \overline{B}_{PQR,R} \\ &= (\overline{a}_{PQ} + \overline{b}_{PQ} + \overline{a}_{PR} + \overline{b}_{PR} + \overline{a}_{QR} + \overline{b}_{QR}) \times n \\ &\quad + 2 \times (\overline{A}_{PQR,P} + \overline{A}_{PQR,Q} + \overline{A}_{PQR,R}) \\ &\quad + 2 \times (\overline{B}_{PQR,P} + \overline{B}_{PQR,Q} + \overline{B}_{PQR,R}) \\ &= (\overline{a}_{PQ} + \overline{b}_{PQ} + \overline{a}_{PR} + \overline{b}_{PR} + \overline{a}_{QR} + \overline{b}_{QR} \\ &\quad + 2 \times \overline{a}_{PQR} + 2 \times \overline{b}_{PQR}) \times n \end{aligned} \quad (1)$$

As

$$(\overline{a}_P + \overline{a}_Q + \overline{a}_R + \overline{a}_{PQ} + \overline{a}_{PR} + \overline{a}_{QR} + \overline{a}_{PQR}) \times n = n \times n = n^2,$$
$$(\overline{b}_P + \overline{b}_Q + \overline{b}_R + \overline{b}_{PQ} + \overline{b}_{PR} + \overline{b}_{QR} + \overline{b}_{PQR}) \times n = n \times n = n^2,$$

the alternative formula will be

$$\overline{f}_c(M) = 2 \times n^2 - (\overline{a}_P + \overline{a}_Q + \overline{a}_R + \overline{b}_P + \overline{b}_Q + \overline{b}_R) \times n + (\overline{a}_{PQR} + \overline{b}_{PQR}) \times n \quad (2)$$

C. CONTINUOUS EXTENSION OF DISCRETE COST FUNCTION

While motivated by the problem of optimal partitioning of a square $n \times n$ matrix between three heterogeneous processors, in this work we aim to solve a more general problem, namely, the problem of optimal partitioning of a real-valued $[0, n] \times [0, n]$ square. Each partition T of the $[0, n] \times [0, n]$ square between three processors P, Q, and R, is defined as a mapping $T : [0, n] \times [0, n] \mapsto \{P, Q, R\}$ such that the inverse images $T^{-1}(P)$,$T^{-1}(Q)$, and $T^{-1}(R)$, are all Lebesgue-Borel measurable sets; the measure of the Lebesgue-Borel measurable set L is here denoted by $\mu(L)$. The set of all possible partitions is denoted as

$$\mathbb{T} = \left\{ [0, n] \times [0, n] \mapsto \{P, Q, R\} \right\},$$

and each partition $T \in \mathbb{T}$ is characterized by the parameters summarized in Table 2. For any $X \in \{P, Q, R\}$, parameter a_X is defined as follows. Let A_X be the set of all horizontal lines mapped to X as a whole. Then, $a_X = \mu(A_X)$. Similarly, $b_X = \mu(B_X)$, where B_X is the set of all vertical lines mapped to X as a whole. For any $X, Y \in \{P, Q, R\}$, parameter $a_{XY} = \mu(A_{XY})$, where A_{XY} is the set of all horizontal lines partitioned between X and Y only. Similarly, $b_{XY} = \mu(B_{XY})$, where B_{XY} is the set of all vertical lines partitioned between X and Y only. Finally, $a_{PQR} = \mu(A_{PQR})$ and $b_{PQR} = \mu(B_{PQR})$, where A_{PQR} and B_{PQR} are the sets of all horizontal and vertical lines correspondingly, partitioned between all three processors.

TABLE 2. Parameters of an arbitrary partition $T \in \mathbb{T}$ of a real-valued $n \times n$ square used in the continuous cost function $f_c(T)$.

Horizontal lines	Vertical lines
$a_{\mathbf{P}}$: $\mu(A_P)$	$b_{\mathbf{P}}$: $\mu(B_P)$
$a_{\mathbf{Q}}$: $\mu(A_Q)$	$b_{\mathbf{Q}}$: $\mu(B_Q)$
$a_{\mathbf{R}}$: $\mu(A_R)$	$b_{\mathbf{R}}$: $\mu(B_R)$
$a_{\mathbf{PQ}}$: $\mu(A_{PQ})$	$b_{\mathbf{PQ}}$: $\mu(B_{PQ})$
$a_{\mathbf{PR}}$: $\mu(A_{PR})$	$b_{\mathbf{PR}}$: $\mu(B_{PR})$
$a_{\mathbf{QR}}$: $\mu(A_{QR})$	$b_{\mathbf{QR}}$: $\mu(B_{QR})$
$a_{\mathbf{PQR}}$: $\mu(A_{PQR})$	$b_{\mathbf{PQR}}$: $\mu(B_{PQR})$
$\Sigma a = n$	$\Sigma b = n$

Note that if we consider the $[0, n] \times [0, n]$ square as a $n \times n$ set of unit squares, that is, squares of size 1×1, then any partition $T \in \mathbb{T}$, which is mapping each unit square to a single processor, will represent a matrix partition, $M \in \mathbb{M}$.

Now for each partition $T \in \mathbb{T}$, we define the cost function $f_c(T)$ as follows

$$f_c(T) = (a_{PQ} + b_{PQ} + a_{PR} + b_{PR} + a_{QR} + b_{QR} + 2 \times a_{PQR} + 2 \times b_{PQR}) \times n \quad (3)$$

This definition guarantees that if $T \in \mathbb{T}$ represents the matrix partition $M \in \mathbb{M}$, then $f_c(T) = \overline{f}_c(M)$.

Also, as
$(a_P + a_Q + a_R + a_{PQ} + a_{PR} + a_{QR} + a_{PQR}) \times n = n \times n = n^2$,
and
$(b_P + b_Q + b_R + b_{PQ} + b_{PR} + b_{QR} + b_{PQR}) \times n = n \times n = n^2$,
the alternative formula will be

$$f_c(T) = 2 \times n^2 - (a_P + a_Q + a_R + b_P + b_Q + b_R) \times n + (a_{PQR} + b_{PQR}) \times n \quad (4)$$

V. OPTIMAL PARTITIONS OF SQUARE

Now we formulate the problem of optimal partitioning of a square with the cost function $f_c(T)$ defined in the previous section and give its solution.

Problem: Given a real-valued square $[0, n] \times [0, n]$ and three positive real numbers $\{S_P, S_Q, S_R\}$ such that $S_P + S_Q + S_R = n^2$, find a partition $T \in \mathbb{T}$ of this square that minimizes the cost function $f_c(T)$ and $S_P = \mu(T^{-1}(P))$, $S_Q = \mu(T^{-1}(Q))$, $S_R = \mu(T^{-1}(R))$.

A. OPTIMAL PARTITION SHAPES

Our primary result determines that the solution to this problem will always be of one of the three partitioning shapes shown in Fig. 1. In the first step, we derive the cost of these three partitions to mathematically formulate and prove this result. We denote the Square Corner partition as T_{SC}, Square Rectangle partition as T_{SR} and Block Rectangle partition as T_{BR}. Also, note that for these partitions, the measure of each of the regions of the square mapped to a single processor P, Q, or R, will be equivalent to the usual area of this region and will be equal to S_P, S_Q, S_R respectively. The formulas for the cost of these partitions are given by the following three lemmas.

Lemma 1: $f_c(T_{SC}) = 2 \times n \times (\sqrt{S_R} + \sqrt{S_Q})$.

Proof: Here, $a_{PQR} = b_{PQR} = a_{QR} = b_{QR} = 0$. On the other hand, $a_{PQ} = \sqrt{S_Q}$, $a_{PR} = \sqrt{S_R}$ and $b_{PQ} = \sqrt{S_Q}$ and $b_{PR} = \sqrt{S_R}$. Therefore, according to formula (3):

$$f_c(T_{SC}) = 2 \times n \times (\sqrt{S_R} + \sqrt{S_Q}) \tag{5}$$

□

Lemma 2: $f_c(T_{SR}) = n^2 + 2 \times (\sqrt{S_R} \times n)$.

Proof: Here, $a_{PQR} = \sqrt{S_R}$, $a_{QR} = a_{PR} = 0$, $a_{PQ} = (n - \sqrt{S_R})$, $b_{PQR} = b_{PQ} = b_{QR} = 0$, and $b_{PR} = \sqrt{S_R}$. Thus, according to formula (3):

$$f_c(T_{SR}) = n^2 + 2 \times (\sqrt{S_R} \times n) \tag{6}$$

□

Lemma 3: $f_c(T_{BR}) = 2 \times n^2 - S_P$.

Proof: Here, $a_{PQR} = a_P = a_Q = a_R = 0$, $b_{PQR} = b_Q = b_R = 0$ and $b_P = S_P$. Therefore, according to formula (4):

$$f_c(T_{BR}) = 2 \times n^2 - b_P \times n = 2 \times n^2 - S_P \tag{7}$$

□

The following two lemmas derive the communication cost of partitions $T1 - T6$ shown in Figures 2 and 3 in Section III.

Lemma 4: $f_c(T1) = 2n^2$. $f_c(T2) = 2n^2 - \frac{1}{2}\sqrt{S_R} \times n + \sqrt{S_Q} \times n$.

Proof: For partition $T1$, we have $a_{PQ} + a_{PR} = n$, $b_{PQ} + b_{PR} = n$ and $a_{QR} = b_{QR} = 0$ whereas, $a_P = a_Q = a_R = b_P = b_Q = b_R = 0$ and $a_{PQR} = b_{PQR} = 0$. Therefore, according to formula (4), $f_c(T1) = 2 \times n^2$.

For partition $T2$, $b_P = \frac{1}{2}\sqrt{S_R}$ whereas, $a_P = a_Q = a_R = b_Q = b_R = 0$, $a_{PQR} = 0$, $b_{PQR} = \sqrt{S_Q}$. Therefore, according to formula (4), $f_c(T2) = 2n^2 - \frac{1}{2}\sqrt{S_R} \times n + \sqrt{S_Q} \times n$. □

Lemma 5: $f_c(T3) = f_c(T4) = 2 \times n^2$ *and* $f_c(T5) = f_c(T6) = 2 \times n^2 - S_P$

Proof: For partition $T3$, we have $a_{PQ} + a_{PR} = n$, $b_{PR} + b_{PQ} = n$ and $a_{QR} = b_{QR} = 0$ whereas, $a_P = a_Q = a_R = b_P = b_Q = b_R = 0$ and $a_{PQR} = b_{PQR} = 0$. Therefore, according to formula (4), $f_c(T3) = 2 \times n^2$.

For partition $T4$, we have $a_{PQ} + a_{PR} = n$, $b_{PQ} + b_{PR} = n$ and $a_{QR} = b_{QR} = 0$ whereas, $a_P = a_Q = a_R = b_P = b_Q = b_R = 0$ and $a_{PQR} = b_{PQR} = 0$. Therefore, according to formula (4), $f_c(T4) = 2 \times n^2$.

For partition $T5$, we have $b_P = \frac{S_P}{n}$ and $a_P = a_Q = a_R = b_Q = b_R = 0$ and $a_{PQR} = b_{PQR} = 0$. Therefore, according to formula (4), $f_c(T5) = 2 \times n^2 - S_P$.

For partition $T6$, we have $b_P = \frac{S_P}{n}$ and $a_P = a_Q = a_R = b_Q = b_R = 0$ and $a_{PQR} = b_{PQR} = 0$. Therefore, according to formula (4), $f_c(T6) = 2 \times n^2 - S_P$. □

The following theorem formulates our main result for the formulated problem of optimal partitioning of a square.

Theorem 1: $\forall T \in \mathbb{T} : (f_c(T) \geq f_c(T_{SC})) \vee (f_c(T) \geq f_c(T_{SR})) \vee (f_c(T) \geq f_c(T_{BR}))$.

Proof: The proof is split into lemmas, depending on the measure of the set of horizontal and vertical lines where all points of a line are mapped to a single processor P, Q and R for any partition T.

Lemma 1.1: If $(a_P = 0) \wedge (a_Q = 0) \wedge (a_R = 0) \wedge (b_P = 0) \wedge (b_Q = 0) \wedge (b_R = 0)$, *then* $f_c(T) \geq f_c(T_{BR})$.

Proof: According to formula (4):

$$\begin{aligned} f_c(T) &= 2 \times n^2 + (a_{PQR} \times n) + (n \times b_{PQR}) \\ &\geq 2 \times n^2 > 2 \times n^2 - S_P = f_c(T_{BR}) \end{aligned}$$

□

Lemma 1.2: If $(a_P > 0) \oplus (a_Q > 0) \oplus (a_R > 0) \oplus (b_P > 0) \oplus (b_Q > 0) \oplus (b_R > 0)$, *then* $f_c(T) \geq f_c(T_{BR})$.

Proof: Let there exist exactly one $X \in \{P, Q, R\}$ such that $a_X > 0$ and $b_P = b_Q = b_R = 0$. Then, according to formula (4), $f_c(T) = 2 \times n^2 - (a_X \times n) + (a_{PQR} \times n) + (n \times b_{PQR})$, where $a_X \times n \leq S_X$, so that $f_c(T) \geq 2 \times n^2 - S_X$.

As $S_P \geq S_Q \geq S_R$, $S_X \leq \max\{S_P, S_Q, S_R\} = S_P$. Therefore, $f_c(T) \geq 2 \times n^2 - S_X \geq 2 \times n^2 - S_P = f_c(T_{BR})$.

Similarly, $f_c(T) \geq f_c(T_{BR})$ when there exists exactly one $X \in \{P, Q, R\}$ such that $b_X > 0$ and $a_P = a_Q = a_R = 0$. □

Lemma 1.3: If $[(a_P > 0) \wedge (b_P > 0)] \oplus [(a_Q > 0) \wedge (b_Q > 0)] \oplus [(a_R > 0) \wedge (b_R > 0)]$, *then* $f_c(T) \geq f_c(T_{SC})$.

Proof: In this case, we assume that there is exactly one processor $X \in \{P, Q, R\}$ such that $a_X > 0$ and $b_X > 0$ and for remaining processors Y and Z, $a_Y = a_Z = b_Y = b_Z = 0$. Also, as any horizontal line and any vertical line contain a point mapped to X, therefore, $a_{YZ} = b_{YZ} = 0$.

Then, according to formula (3), $f_c(T) = (a_{XY} + b_{XY} + a_{XZ} + b_{XZ} + 2 \times a_{XYZ} + 2 \times b_{XYZ}) \times n$.

The measure of the set of all horizontal and vertical lines containing points mapped to Y will be equal to $(a_{XY} + a_{XYZ} + b_{XY} + b_{XYZ})$ and cannot be less than the half-perimeter of a square with the area of S_Y. Therefore, $a_{XY} + a_{XYZ} + b_{XY} + b_{XYZ} \geq 2 \times \sqrt{S_Y}$. Similarly, $a_{XZ} + a_{XYZ} + b_{XZ} + b_{XYZ} \geq 2 \times \sqrt{S_Z}$.

Thus, $f_c(T) = (a_{XY} + b_{XY} + a_{XZ} + b_{XZ} + 2 \times a_{XYZ} + 2 \times b_{XYZ}) \times n = ((a_{XY} + a_{XYZ} + b_{XY} + b_{XYZ}) + (a_{XZ} + a_{XYZ} + b_{XZ} + b_{XYZ})) \times n \geq 2 \times (\sqrt{S_Y} + \sqrt{S_Z}) \times n \geq 2 \times (\sqrt{S_Q} + \sqrt{S_R}) \times n = f_c(T_{SC})$. □

Lemma 1.4: If $[(a_P > 0) \wedge (a_Q > 0)] \oplus [(a_P > 0) \wedge (a_R > 0)] \oplus [(a_Q > 0) \wedge (a_R > 0)] \oplus [(b_P > 0) \wedge (b_Q > 0)] \oplus [(b_P > 0) \wedge (b_R > 0)] \oplus [(b_Q > 0) \wedge (b_R > 0)]$, *then* $f_c(T) \geq f_c(T_{SR})$

Proof: Let $X, Y \in \{P, Q, R\}$, $a_X > 0$ and $a_Y > 0$ while $b_X = b_Y = 0$. Note that for the remaining processor

$Z \in \{P, Q, R\}$, $a_Z = b_Z = 0$, Then, according to formula (3), $f_c(T) = (a_{XY} + a_{XZ} + a_{YZ} + b_{XY} + b_{XZ} + b_{YZ} + 2(a_{XYZ} + b_{XYZ})) \times n$.

The measure of the set of all horizontal and vertical lines containing points mapped to Z will be equal to $(a_{XZ} + a_{YZ} + a_{XYZ} + b_{XZ} + b_{YZ} + b_{XYZ})$ and cannot be less than the half-perimeter of a square with the area of S_Z. Therefore, $a_{XZ} + a_{YZ} + a_{XYZ} + b_{XZ} + b_{YZ} + b_{XYZ} \geq 2 \times \sqrt{S_Z}$. Also, all vertical lines contain points from X and Y, therefore, $b_{XY} + b_{XYZ} = n$. Thus,

$$\begin{aligned} f_c(T) &= (a_{XY} + a_{XZ} + a_{YZ} + b_{XY} + b_{XZ} + b_{YZ} \\ &\quad + 2(a_{XYZ} + b_{XYZ})) \times n \\ &\geq (a_{XY} + a_{XYZ} + b_{XY} + b_{XYZ}) \times n + 2(\sqrt{S_Z} \times n) \\ &\geq n \times (b_{XY} + b_{XYZ}) + 2(\sqrt{S_Z} \times n) \\ &\geq n^2 + 2(\sqrt{S_Z} \times n) \\ &\geq n^2 + 2(\sqrt{S_R} \times n) = f_c(T_{SR}). \end{aligned}$$

Similarly, $f_c(T) \geq f_c(T_{SR})$ when there exist $X, Y \in \{P, Q, R\}$ such that $b_X > 0$ and $b_Y > 0$. □

Lemma 1.5: If $[(a_P > 0) \wedge (a_Q > 0) \wedge (a_R > 0)] \oplus [(b_P > 0) \wedge (b_Q > 0) \wedge (b_R > 0)]$, *then* $f_c(T) \geq f_c(T_{BR})$

Proof: Let $(a_P > 0) \wedge (a_Q > 0) \wedge (a_R > 0)$. Then, $b_P = b_Q = b_R = 0$ and $b_{PQR} = n$. Therefore, according to formula (4),

$$\begin{aligned} f_c(T) &\geq 2 \times n^2 - ((a_P + a_Q + a_R) \times n) + (a_{PQR} \times n) + n^2 \\ &\geq 2 \times n^2 \\ &> 2 \times n^2 - S_P = f_c(T_{BR}). \end{aligned}$$

Similarly, $f_c(T) \geq f_c(T_{BR})$ when $(b_P > 0) \wedge (b_Q > 0) \wedge (b_R > 0)$. □

Hence, it is proved that for any feasible combination of the partition parameters from Table 2, one of the shape of Fig. 1 will always be an optimal solution to Problem 1. The following last lemma of this section proves that lemmas 1.1 – 1.5 covered all the possible combinations of the partition parameters.

Lemma 1.6: For any partition $T \in \mathbb{T}$, *its parameters from Table 2 will satisfy at least one of the cases of lemmas 1.1 – 1.5.*

Proof: Let us denote $\mathfrak{A}_P \equiv (a_P > 0), \mathfrak{A}_Q \equiv (a_Q > 0), \mathfrak{A}_R \equiv (a_R > 0), \mathfrak{B}_P \equiv (b_P > 0), \mathfrak{B}_Q \equiv (b_Q > 0), \mathfrak{B}_R \equiv (b_R > 0)$.

Then, the case of lemma 1.1 can be expressed as follows,
$(\neg\mathfrak{A}_P \wedge \neg\mathfrak{A}_Q \wedge \neg\mathfrak{A}_R \wedge \neg\mathfrak{B}_P \wedge \neg\mathfrak{B}_Q \wedge \neg\mathfrak{B}_R)$

The case of lemma 1.2:

$$\begin{aligned} &(\neg\mathfrak{A}_P \wedge \neg\mathfrak{A}_Q \wedge \neg\mathfrak{A}_R \wedge \neg\mathfrak{B}_P \wedge \neg\mathfrak{B}_Q \wedge \mathfrak{B}_R) \\ &\quad \oplus (\neg\mathfrak{A}_P \wedge \neg\mathfrak{A}_Q \wedge \neg\mathfrak{A}_R \wedge \neg\mathfrak{B}_P \wedge \neg\mathfrak{B}_R \wedge \mathfrak{B}_Q) \\ &\quad \oplus (\neg\mathfrak{A}_P \wedge \neg\mathfrak{A}_Q \wedge \neg\mathfrak{A}_R \wedge \neg\mathfrak{B}_Q \wedge \neg\mathfrak{B}_R \wedge \mathfrak{B}_P) \\ &\quad \oplus (\neg\mathfrak{B}_P \wedge \neg\mathfrak{B}_Q \wedge \neg\mathfrak{B}_R \wedge \neg\mathfrak{A}_P \wedge \neg\mathfrak{A}_Q \wedge \mathfrak{A}_R) \\ &\quad \oplus (\neg\mathfrak{B}_P \wedge \neg\mathfrak{B}_Q \wedge \neg\mathfrak{B}_Z \wedge \neg\mathfrak{A}_P \wedge \neg\mathfrak{A}_R \wedge \mathfrak{A}_Q) \\ &\quad \oplus (\neg\mathfrak{B}_P \wedge \neg\mathfrak{B}_Q \wedge \neg\mathfrak{B}_R \wedge \neg\mathfrak{A}_Q \wedge \neg\mathfrak{A}_R \wedge \mathfrak{A}_P) \end{aligned}$$

The case of lemma 1.3:

$$\begin{aligned} &(\neg\mathfrak{A}_P \wedge \neg\mathfrak{A}_Q \wedge \neg\mathfrak{B}_P \wedge \neg\mathfrak{B}_Q \wedge \mathfrak{A}_R \wedge \mathfrak{B}_R) \\ &\quad \oplus (\neg\mathfrak{A}_P \wedge \neg\mathfrak{A}_R \wedge \neg\mathfrak{B}_P \wedge \neg\mathfrak{B}_R \wedge \mathfrak{A}_Q \wedge \mathfrak{B}_Q) \\ &\quad \oplus (\neg\mathfrak{A}_Q \wedge \neg\mathfrak{A}_R \wedge \neg\mathfrak{B}_Q \wedge \neg\mathfrak{B}_R \wedge \mathfrak{A}_P \wedge \mathfrak{B}_P) \end{aligned}$$

The case of lemma 1.4:

$$\begin{aligned} &(\neg\mathfrak{A}_P \wedge \neg\mathfrak{A}_Q \wedge \neg\mathfrak{A}_R \wedge \neg\mathfrak{B}_P \wedge \mathfrak{B}_Q \wedge \mathfrak{B}_R) \\ &\quad \oplus (\neg\mathfrak{A}_P \wedge \neg\mathfrak{A}_Q \wedge \neg\mathfrak{A}_R \wedge \neg\mathfrak{B}_Q \wedge \mathfrak{B}_P \wedge \mathfrak{B}_R) \\ &\quad \oplus (\neg\mathfrak{A}_P \wedge \neg\mathfrak{A}_Q \wedge \neg\mathfrak{A}_R \wedge \neg\mathfrak{B}_R \wedge \mathfrak{B}_P \wedge \mathfrak{B}_Q) \\ &\quad \oplus (\neg\mathfrak{A}_P \wedge \neg\mathfrak{B}_P \wedge \neg\mathfrak{B}_Q \wedge \neg\mathfrak{B}_R \wedge \mathfrak{A}_Q \wedge \mathfrak{A}_R) \\ &\quad \oplus (\neg\mathfrak{A}_Q \wedge \neg\mathfrak{B}_P \wedge \neg\mathfrak{B}_Q \wedge \neg\mathfrak{B}_R \wedge \mathfrak{A}_P \wedge \mathfrak{A}_R) \\ &\quad \oplus (\neg\mathfrak{A}_R \wedge \neg\mathfrak{B}_P \wedge \neg\mathfrak{B}_Q \wedge \neg\mathfrak{B}_R \wedge \mathfrak{A}_P \wedge \mathfrak{A}_Q) \end{aligned}$$

The case of lemma 1.5:

$$\begin{aligned} &(\neg\mathfrak{A}_P \wedge \neg\mathfrak{A}_Q \wedge \neg\mathfrak{A}_R \wedge \mathfrak{B}_P \wedge \mathfrak{B}_Q \wedge \mathfrak{B}_R) \\ &\quad \oplus (\mathfrak{A}_P \wedge \mathfrak{A}_Q \wedge \mathfrak{A}_R \wedge \neg\mathfrak{B}_P \wedge \neg\mathfrak{B}_Q \wedge \neg\mathfrak{B}_R) \\ &[(\neg\mathfrak{A}_P \wedge \neg\mathfrak{A}_Q \wedge \neg\mathfrak{A}_R \wedge \neg\mathfrak{B}_P \wedge \neg\mathfrak{B}_Q \wedge \neg\mathfrak{B}_R)] \\ &\quad \vee[(\neg\mathfrak{A}_P \wedge \neg\mathfrak{A}_Q \wedge \neg\mathfrak{A}_R \wedge \neg\mathfrak{B}_P \wedge \neg\mathfrak{B}_Q \wedge \mathfrak{B}_R) \\ &\quad \oplus (\neg\mathfrak{A}_P \wedge \neg\mathfrak{A}_Q \wedge \neg\mathfrak{A}_R \wedge \neg\mathfrak{B}_P \wedge \neg\mathfrak{B}_R \wedge \mathfrak{B}_Q) \\ &\quad \oplus (\neg\mathfrak{A}_P \wedge \neg\mathfrak{A}_Q \wedge \neg\mathfrak{A}_R \wedge \neg\mathfrak{B}_Q \wedge \neg\mathfrak{B}_R \wedge \mathfrak{B}_P) \\ &\quad \oplus (\neg\mathfrak{B}_P \wedge \neg\mathfrak{B}_Q \wedge \neg\mathfrak{B}_R \wedge \neg\mathfrak{A}_P \wedge \neg\mathfrak{A}_Q \wedge \mathfrak{A}_R) \\ &\quad \oplus (\neg\mathfrak{B}_P \wedge \neg\mathfrak{B}_Q \wedge \neg\mathfrak{B}_R \wedge \neg\mathfrak{A}_P \wedge \neg\mathfrak{A}_R \wedge \mathfrak{A}_Q) \\ &\quad \oplus (\neg\mathfrak{B}_P \wedge \neg\mathfrak{B}_Q \wedge \neg\mathfrak{B}_R \wedge \neg\mathfrak{A}_Q \wedge \neg\mathfrak{A}_R \wedge \mathfrak{A}_P)] \\ &\quad \vee[(\neg\mathfrak{A}_P \wedge \neg\mathfrak{A}_R \wedge \neg\mathfrak{B}_P \neg\mathfrak{B}_Q \wedge \mathfrak{A}_R \wedge \mathfrak{B}_R) \\ &\quad \oplus (\neg\mathfrak{A}_P \wedge \neg\mathfrak{A}_R \wedge \neg\mathfrak{B}_P \wedge \neg\mathfrak{B}_R \wedge \mathfrak{A}_Q \wedge \mathfrak{B}_Q) \\ &\quad \oplus (\neg\mathfrak{A}_Q \wedge \neg\mathfrak{A}_R \wedge \neg\mathfrak{B}_Q \wedge \neg\mathfrak{B}_R \wedge \mathfrak{A}_P \wedge \mathfrak{B}_P)] \\ &\quad \vee[(\neg\mathfrak{A}_P \wedge \neg\mathfrak{A}_Q \wedge \neg\mathfrak{A}_R \wedge \neg\mathfrak{B}_P \wedge \mathfrak{B}_Q \wedge \mathfrak{B}_R) \\ &\quad \oplus (\neg\mathfrak{A}_P \wedge \neg\mathfrak{A}_Q \wedge \neg\mathfrak{A}_R \wedge \neg\mathfrak{B}_Q \wedge \mathfrak{B}_P \wedge \mathfrak{B}_R) \\ &\quad \oplus (\neg\mathfrak{A}_P \wedge \neg\mathfrak{A}_Q \wedge \neg\mathfrak{A}_R \wedge \neg\mathfrak{B}_R \wedge \mathfrak{B}_P \wedge \mathfrak{B}_Q) \\ &\quad \oplus (\neg\mathfrak{A}_P \wedge \neg\mathfrak{B}_P \wedge \neg\mathfrak{B}_Q \wedge \neg\mathfrak{B}_R \wedge \mathfrak{A}_Q \wedge \mathfrak{A}_R) \\ &\quad \oplus (\neg\mathfrak{A}_Q \wedge \neg\mathfrak{B}_P \wedge \neg\mathfrak{B}_Q \wedge \neg\mathfrak{B}_R \wedge \mathfrak{A}_P \wedge \mathfrak{A}_R) \\ &\quad \oplus (\neg\mathfrak{A}_R \wedge \neg\mathfrak{B}_P \wedge \neg\mathfrak{B}_Q \wedge \neg\mathfrak{B}_R \wedge \mathfrak{A}_P \wedge \mathfrak{A}_Q)] \\ &\quad \vee[(\neg\mathfrak{A}_P \wedge \neg\mathfrak{A}_Q \wedge \neg\mathfrak{A}_R \wedge \mathfrak{B}_P \wedge \mathfrak{B}_Q \wedge \mathfrak{B}_R) \\ &\quad \oplus (\mathfrak{A}_P \wedge \mathfrak{A}_Q \wedge \mathfrak{A}_R \wedge \neg\mathfrak{B}_P \wedge \neg\mathfrak{B}_Q \wedge \neg\mathfrak{B}_R)], \end{aligned}$$

and its disjunctive normal form will be

$$\begin{aligned} &(\neg\mathfrak{A}_P \wedge \neg\mathfrak{A}_Q \wedge \neg\mathfrak{A}_R \wedge \neg\mathfrak{B}_P \wedge \neg\mathfrak{B}_Q \wedge \neg\mathfrak{B}_R) \\ &\quad \vee(\neg\mathfrak{A}_P \wedge \neg\mathfrak{A}_Q \wedge \neg\mathfrak{A}_R \wedge \neg\mathfrak{B}_P \wedge \neg\mathfrak{B}_Q \wedge \mathfrak{B}_R) \\ &\quad \vee(\neg\mathfrak{A}_P \wedge \neg\mathfrak{A}_Q \wedge \neg\mathfrak{A}_R \wedge \neg\mathfrak{B}_P \wedge \mathfrak{B}_Q \wedge \neg\mathfrak{B}_R) \\ &\quad \vee(\neg\mathfrak{A}_P \wedge \neg\mathfrak{A}_Q \wedge \neg\mathfrak{A}_R \wedge \neg\mathfrak{B}_P \wedge \mathfrak{B}_Q \wedge \mathfrak{B}_R) \\ &\quad \vee(\neg\mathfrak{A}_P \wedge \neg\mathfrak{A}_Q \wedge \neg\mathfrak{A}_R \wedge \mathfrak{B}_P \wedge \neg\mathfrak{B}_Q \wedge \neg\mathfrak{B}_R) \\ &\quad \vee(\neg\mathfrak{A}_P \wedge \neg\mathfrak{A}_Q \wedge \neg\mathfrak{A}_R \wedge \mathfrak{B}_P \wedge \neg\mathfrak{B}_Q \wedge \mathfrak{B}_R) \\ &\quad \vee(\neg\mathfrak{A}_P \wedge \neg\mathfrak{A}_Q \wedge \neg\mathfrak{A}_R \wedge \mathfrak{B}_P \wedge \mathfrak{B}_Q \wedge \neg\mathfrak{B}_R) \\ &\quad \vee(\neg\mathfrak{A}_P \wedge \neg\mathfrak{A}_Q \wedge \neg\mathfrak{A}_R \wedge \mathfrak{B}_P \wedge \mathfrak{B}_Q \wedge \mathfrak{B}_R) \\ &\quad \vee(\neg\mathfrak{A}_P \wedge \neg\mathfrak{A}_Q \wedge \mathfrak{A}_R \wedge \neg\mathfrak{B}_P \wedge \neg\mathfrak{B}_Q \wedge \neg\mathfrak{B}_R) \\ &\quad \vee(\neg\mathfrak{A}_P \wedge \neg\mathfrak{A}_Q \wedge \mathfrak{A}_R \wedge \neg\mathfrak{B}_P \wedge \neg\mathfrak{B}_Q \wedge \mathfrak{B}_R) \end{aligned}$$

$$\vee(\neg\mathfrak{A}_P \wedge \mathfrak{A}_Q \wedge \neg\mathfrak{A}_R \wedge \neg\mathfrak{B}_P \wedge \neg\mathfrak{B}_Q \wedge \neg\mathfrak{B}_R)$$
$$\vee(\neg\mathfrak{A}_P \wedge \mathfrak{A}_Q \wedge \neg\mathfrak{A}_R \wedge \neg\mathfrak{B}_P \wedge \mathfrak{B}_Q \wedge \neg\mathfrak{B}_R)$$
$$\vee(\neg\mathfrak{A}_P \wedge \mathfrak{A}_Q \wedge \mathfrak{A}_R \wedge \neg\mathfrak{B}_P \wedge \neg\mathfrak{B}_Q \wedge \neg\mathfrak{B}_R)$$
$$\vee(\mathfrak{A}_P \wedge \neg\mathfrak{A}_Q \wedge \neg\mathfrak{A}_R \wedge \neg\mathfrak{B}_P \wedge \neg\mathfrak{B}_Q \wedge \neg\mathfrak{B}_R)$$
$$\vee(\mathfrak{A}_P \wedge \neg\mathfrak{A}_Q \wedge \neg\mathfrak{A}_R \wedge \mathfrak{B}_P \wedge \neg\mathfrak{B}_Q \wedge \neg\mathfrak{B}_R)$$
$$\vee(\mathfrak{A}_P \wedge \neg\mathfrak{A}_Q \wedge \mathfrak{A}_R \wedge \neg\mathfrak{B}_P \wedge \neg\mathfrak{B}_Q \wedge \neg\mathfrak{B}_R)$$
$$\vee(\mathfrak{A}_P \wedge \mathfrak{A}_Q \wedge \neg\mathfrak{A}_R \wedge \neg\mathfrak{B}_P \wedge \neg\mathfrak{B}_Q \wedge \neg\mathfrak{B}_R)$$
$$\vee(\mathfrak{A}_P \wedge \mathfrak{A}_Q \wedge \mathfrak{A}_R \wedge \neg\mathfrak{B}_P \wedge \neg\mathfrak{B}_Q \wedge \neg\mathfrak{B}_R) = \text{TRUE}$$

□
□

Now, we are ready to propose the algorithm, solving the problem of optimal partitioning of a square with the cost function $f_c(T)$, formulated in the beginning of this section.

B. SELECTION OF OPTIMAL PARTITION

In the previous section, we proved that to find the optimal partition of a square of size $n \times n$ into three regions of areas S_P, S_Q and S_R, which minimizes the cost function $f_c(T)$, we only have to check the partitions of the three shapes shown in Fig. 1. As there are three regions in a shape, the number of different partitions of the shape in the given proportion $S_P : S_Q : S_R$ will be equal to the total number of different mappings of three processors P, Q and R to the three regions of the shape, which is calculated as $3! = 6$. Therefore, the total number of candidate partitions to solve the problem will be $6 \times 3 = 18$.

We do not need however to calculate the cost of all these 18 partitions to find the optimal one. Indeed, according to formula (5), in any feasible partition of the Square Corner shape, the P region of the shape (as shown in Fig. 1) must be the largest. Formula (6) indicates that out of 6 partitions of the Square Rectangle shape, the minimal cost will be achieved by any partition, which minimizes the area of the R region. Finally, as formula (7) suggests, any partition maximizing the area of the P region will have the minimal cost among partitions of the Block Rectangle shape. Thus, in order to find the optimal partition, we only need to compare the cost of 3 partitions, one for each shape, but not 18.

VI. EXPERIMENTAL VALIDATION

This section presents experiments for validation of our theoretical results. The challenging part was to design these experiments in such a way which accurately measured the contribution of inter-memory data movements between the tightly coupled compute devices of our hybrid heterogeneous server into the total execution time. To the best of our knowledge, the methodology proposed in this paper for accurate experimental evaluation is the first of its kind which successfully solves this challenging issue for hybrid data parallel applications. The solutions we come across in the literature have to a great extent underestimated the time of these data movements and often have instability in results.

The purpose of these experiments is to validate the predictive accuracy of our theoretical models and also to demonstrate its usefulness in making practical decisions on partitioning square computational domains between tightly integrated compute devices in hybrid heterogeneous servers.

A. EXPERIMENTAL METHODOLOGY

We perform our experiments on HCLServer01. HCLServer01 is a hybrid heterogeneous server which integrates multi-core CPU (Intel Haswell $E5-2670\ V3$) and two accelerators-GPU (Nvidia $K40c$) and Xeon Phi (Intel $3120P$). The CPU has 24 physical cores and 64 GB main memory. Table.3 shows the detailed specifications of HCLServer01.

TABLE 3. HCLServer1 specifications.

Intel Haswell E5-2670 V3	
Cores per socket	12
No. of Sockets	2
CPU MHz	1200.40
L1d, L1i cache	3 KB
L2 cache	256 KB
LL3 cache	30720 KB
Main memory DDR4	64 GB
Memory Bandwidth	68 GB/sec
NVIDIA K40c	
Processor cores	2880
L2 cache	1536 KB
Board memory GDDR5	12 GB
Memory Bandwidth	288 GB/sec
Intel Xeon Phi 3120P	
Processor cores	57
Main memory GDDR5	6 GB
Memory Bandwidth	240 GB/sec

A data-parallel application running on HCLServer01 would naturally consist of three kernels – one running on the multi-core CPU, the other running on the GPU, and the third running on the Xeon Phi. Each accelerator kernel is hosted by one CPU core, which is not involved in the execution of the CPU kernel. Thus, the resources used by the CPU kernel include 22 CPU cores and DRAM. The resources used by the GPU kernel include one CPU core and the GPU together with its memory. The resources used by the Xeon Phi kernel include one CPU core and the Intel Xeon Phi accelerator together with its memory. In our model, such a hybrid parallel application is modeled by three abstract processors – one representing the CPU kernel (CPU abstract processor), the second representing the GPU kernel (GPU abstract processor), and the third representing the Xeon Phi kernel (PHI abstract processor). In our platform, the GPU kernel is the fastest, the CPU kernel is medium, and the Xeon Phi kernel is slowest. In order to connect this notation with the notation in the theoretical section, we will also use P, Q and R to denote abstract processors GPU, CPU and PHI respectively and S_P, S_Q and S_R to denote their respective speeds. We perform the experiments for different computational speed ratios of P, Q and R where the relative speed of a processor is represented by a number between 1 to 0.01, with 1 representing the fastest processor.

The ratio shows the portion of square domain area assigned to respective processors as per their speed.

In our experimental study, we do not consider the computation time of the application but only its communication time. The main reason why we exclude the computation time is because we want to experiment with different speed ratios of the processors. However, in our platform, we only have one ratio, and for experiments with other ratios, we assume the balanced workload and distribute the data and perform experiments accordingly. Theoretically, if the workload is balanced then the computation time of all three processors will be the same and its addition will not change the relative optimality of the studied partitions.

To achieve this, we partition the computation domain between GPU, CPU and PHI in proportion with their assumed relative speeds, which can be different in different experimental sets. However, during the execution of the application we do not execute its computation code but only the code implementing data movements. This way we obtain a load-balanced application for different assumed relative speeds of the abstract processors GPU, CPU and PHI, which representative of the original application and the execution time of which will be equal to the time of data movements in the original application.

Our experiments are carefully designed to make sure that what we measure is the time of data transfer between the main memories of the CPU, the GPU and the Intel Xeon Phi, not the time of data transfer between virtual memories of the processes.

We experiment with a parallel matrix multiplication application. The matrix multiplication applications calculate the product $C = A \times B$ of two dense square matrices A and B. We exclude the computation time from its execution time by commenting out the computation code of the application. For each experimental set, matrices are partitioned in proportion to their assumed relative speeds between CPU, GPU and PHI in three optimal partitioning shapes, Square Corner, Square Rectangle and Block Rectangle, as well as the regular 1D partitioning shape called Straight Line. For each partitioning shape, we then measure the execution time of the application and also calculate the predicted communication time by using the accurate communication cost function and the average communication bandwidth between the compute devices. The communication time of real experiments is then compared with the predicted communication time to confirm the accuracy of our communication cost function.

There are three communicating links in our experimental platform - between GPU and CPU, between GPU and PHI and between CPU and PHI. Our communication model considers these links as homogeneous. However, actually these communicating links are heterogeneous in nature so we measure the model-predicted communication time using the average bandwidth.

We make sure that the server is fully reserved and dedicated to our experiments only. We also monitor its load and check for any abnormal event that results in drastic fluctuation. To obtain a data point, we repeatedly execute the application until the sample mean lies in the 95% confidence interval and 0.025 (2.5%) precision has been achieved. To determine the sample mean, we used Student's t-test which assumes that the individual observations are independent and their population follows the normal distribution, and we check that this is the case for each population. We also allow sufficient time to elapse between application successive runs to make sure that cache effects and pipelining do not happen. We make sure that the problem size of our experiments does not exceed the main memory of the compute devices and no paging occurs.

B. PERFORMANCE EXPERIMENTS

Our communication cost function assumes synchronous communications. However, to demonstrate its predictive capability in the situation of asynchronous data transfer, we additionally perform experiments with asynchronous communications. We perform these experiments for three different ratios of $S_P : S_Q : S_R$ while the problem size $N \times N$ is set to 22528×22528 for all the cases. The $CPU - GPU$ link has measured bandwidth of 9.7 GB/s whereas, the $CPU - PHI$ link has 6.3 GB/s, and the $GPU - PHI$ link has 3.6 GB/s. The average bandwidth is calculated as 6.6 GB/s.

The first set of experiments assumes a speed ratio of 1.0 : 0.5 : 0.25 between GPU, CPU and PHI respectively which is the actual configuration of our platform. Our partitioning solutions aim to balance the workload of the processors. Therefore, the computation time is supposed to be same for all devices independent of the configuration of the application, and only the difference in communication time will make their overall execution time different. This is confirmed by our experiments with the actual speed ratios of the heterogeneous processors in our platform shown in Fig. 4. Therefore, we can safely exclude the computation time and only consider the communication time when experimentally comparing the relative optimality of the partitions.

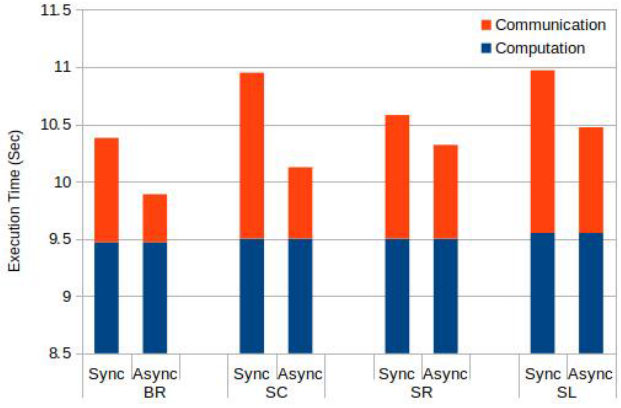


FIGURE 4. Total execution time (computation + communication) of 4 partition shapes for synchronous and asynchronous communication experiments for the problem size of 22528 × 22528 and speed ratio of $S_P : S_Q : S_R = 1.0 : 0.5 : 0.25$.

Our communication model predicts that optimal partition is the Block Rectangle (*BR*) for this configuration.

Subsequently, experiments that incorporate the synchronous and asynchronous communications also confirm that the optimal partitioning with the least communication time is Block Rectangle as shown in Fig. 5.

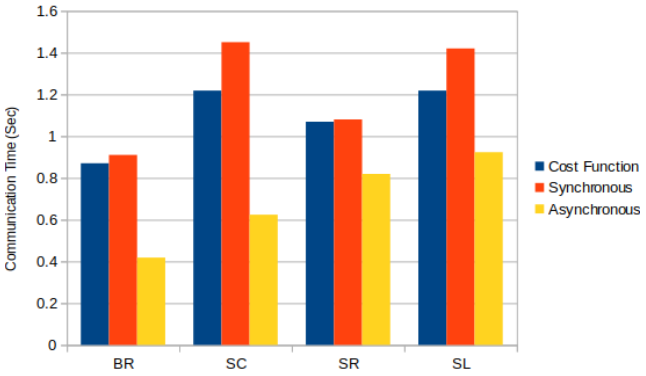


FIGURE 5. Communication time comparison of model-predicted and measured synchronous and asynchronous communication experiments for the problem size of 22528 × 22528 and speed ratio of $S_P : S_Q : S_R = 1.0 : 0.5 : 0.25$.

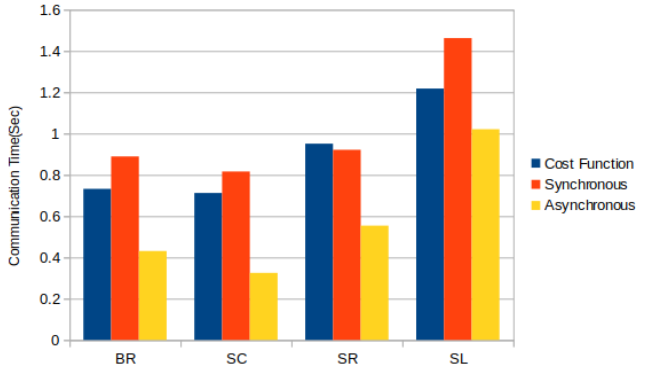


FIGURE 6. Communication time comparison of model-predicted and measured synchronous and asynchronous communication experiments for the problem size of 22528 × 22528 and speed ratio of $S_P : S_Q : S_R = 1.0 : 0.15 : 0.10$.

The second set of experiments assumes a speed ratio of $1.0 : 0.15 : 0.10$ for $S_P : S_Q : S_R$. The Square Corner (SC) is predicted to be optimal by the communication model, which is validated by the real measurements in both synchronous and asynchronous communication experiments Fig. 6. Our last set of experiments assumes a speed ratio of $1 : 0.7 : 0.10$ for $S_P : S_Q : S_R$. For this case, Square Rectangle (SR) is identified as the optimal shape as shown in Fig. 7, both by the communication model and the experiments. Additionally, it is evident from the experiments that predictions of the communication model are adequately close to the actual measurements of synchronous communication. Hence, they can be used for accurate pairwise comparison of different partitions.

While the model also predicts that there is no case where the Straight Line (SL) would outperform any optimal partitioning shapes, its theoretical cost will be equal to the Square

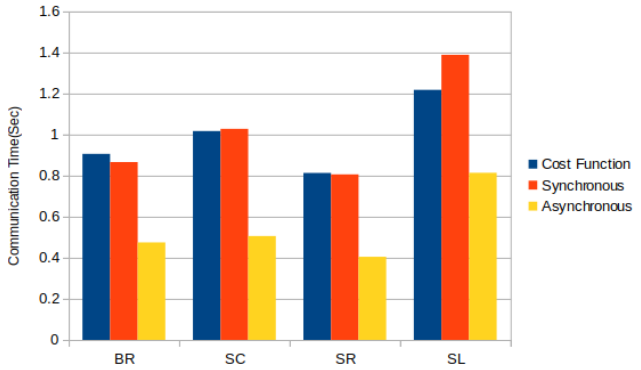


FIGURE 7. Communication time comparison of model-predicted and measured synchronous and asynchronous communication experiments for the problem size of 22528 × 22528 and speed ratio of $S_P : S_Q : S_R = 1.0 : 0.7 : 0.10$.

Corner (SC) cost when $\sqrt{S_R}+\sqrt{S_Q} = n$, as it is in Scenario 1. We do see in our experiments that the measured time and the model predicted communication time of these two shapes are very close in the case of synchronous communication.

Altogether, the experimental results exhibit the accuracy of the proposed communication cost function in the case of synchronous communications. The slight difference between the actual measurement and predicted communication time is due to the use of the average bandwidth in the theoretical calculations. We believe that if the communication links had the same bandwidths, our model would very accurately predict the communication time. Unfortunately, we are unable to verify this experimentally because of different bandwidths of the communication links in our experimental platform. Our dedicated platform is hybrid heterogeneous server comprising CPU, GPU and Xeon Phi, and the execution of hybrid applications on such platform will involve the CPU host-core, DRAM and PCIe's to transfer the data between GPU, CPU and Intel Xeon Phi. For example, data transfer will be slower between the GPU and PHI because it passes through the CPU DRAM and PCIe as compared to data transfer between GPU and CPU.

VII. ENERGY OF COMMUNICATION OF PERFORMANCE OPTIMAL SHAPES

In Section V, we solved the problem of performance optimal partitioning and proved the optimality of the three partition shapes, which minimize the total amount of data moved between three heterogeneous processors. In this section, we discuss the energy cost of communication which has not been studied before. There are some recent results for energy optimization through workload partition [30], [31]. However, they only address the energy cost of computation and do not address the energy cost of communication. In this paper, we try to fill this gap and propose an energy model of communication attributing the energy cost to any partition, and derive energy formulas for the performance optimal partitions. We then use these formulas to predict the communication energy consumption for these partitions

and discover that the performance optimal partition is not necessarily the energy optimal. We perform experiments accurately measuring the energy of data movement between the main memories of three heterogeneous devices on our hybrid server. These experiments validate the accuracy of the proposed energy model and confirm the theoretical finding that the performance optimality does not imply the energy optimality.

A. ENERGY MODEL OF COMMUNICATION

In Section V, when defining the communication cost of a partition of a square matrix, we assumed that the cost (or time) of sending a matrix element between any pair of processors would be the same. However, for energy consumption on a heterogeneous platform, a similar assumption would be highly un-realistic. Therefore, we assume a direct communication link $X \leftrightarrow Y$ between each pair of processors X, $Y \in \{P, Q, R\}$, and introduce a new, real-valued, parameter, ε_{XY}, representing the energy consumption cost of moving one matrix element through this link.

In this energy model of communication, for any partition $M \in \mathbb{M}$, we assume that all matrix elements, which must be moved between processors $X, Y \in \{P, Q, R\}$, will always be sent through the direct link $X \leftrightarrow Y$. The energy consumption of moving the elements between processors X and Y can be calculated as:

$$\begin{aligned} E_{XY} = \varepsilon_{XY} \times [&(\overline{a}_{XY} \times n) + (n \times \overline{b}_{XY}) \\ &+ (\overline{A}_{PQR,X} + \overline{A}_{PQR,Y}) \\ &+ (\overline{B}_{PQR,X} + \overline{B}_{PQR,Y})] \end{aligned}$$

and the total energy consumption cost, $E_{PQ} + E_{PR} + E_{QR}$, as

$$\begin{aligned} \overline{f}_{Ec}(M) = &\varepsilon_{PQ} \times [(\overline{a}_{PQ} + \overline{b}_{PQ}) \times n] \\ &+ \varepsilon_{PR} \times [(\overline{a}_{PR} + \overline{b}_{PR}) \times n] \\ &+ \varepsilon_{QR} \times [(\overline{a}_{QR} + \overline{b}_{QR}) \times n] \\ &+ (\varepsilon_{PQ} + \varepsilon_{PR}) \times [(\overline{A}_{PQR,P}) + \overline{B}_{PQR,P}] \\ &+ (\varepsilon_{PQ} + \varepsilon_{QR}) \times [(\overline{A}_{PQR,Q} + \overline{B}_{PQR,Q})] \\ &+ (\varepsilon_{PR} + \varepsilon_{QR}) \times [(\overline{A}_{PQR,R} + \overline{B}_{PQR,R})], \end{aligned} \quad (8)$$

The continuous extension of this discrete energy model of communication cost is defined as follows. Let $\varepsilon_{XY} > 0$ be the real-valued energy cost of moving a set of points of measure 1 between processors X and Y ($X, Y \in \{P, Q, R\}$), and for any $X, Y, Z \in \{P, Q, R\}$, $\varepsilon_{XY} \leq \varepsilon_{XZ} + \varepsilon_{YZ}$. Then, for each partition $T \in \mathbb{T}$ of the $[0, n] \times [0, n]$ square, its energy communication cost $f_{Ec}(T)$ is defined as follows

$$\begin{aligned} f_{Ec}(T) = &\varepsilon_{PQ} \times [(a_{PQ} + b_{PQ}) \times n] \\ &+ \varepsilon_{PR} \times [(a_{PR} + b_{PR}) \times n] \\ &+ \varepsilon_{QR} \times [(a_{QR} + b_{QR}) \times n] \\ &+ (\varepsilon_{PQ} + \varepsilon_{PR}) \times [(A_{PQR,P}) + B_{PQR,P}] \\ &+ (\varepsilon_{PQ} + \varepsilon_{QR}) \times [(A_{PQR,Q} + B_{PQR,Q})] \\ &+ (\varepsilon_{PR} + \varepsilon_{QR}) \times [(A_{PQR,R} + B_{PQR,R})], \end{aligned} \quad (9)$$

This definition guarantees that if $T \in \mathbb{T}$ represents the matrix partition $M \in \mathbb{M}$, then $f_{Ec}(T) = \overline{f}_{Ec}(M)$.

B. ENERGY COST OF OPTIMAL SHAPES

Now we use the general formula (9) to derive formulas calculating the energy cost of communication for the Square Corner, Square Rectangle, Block Rectangle and Straight Line partitioning shapes.

1) Square Corner:

$$\begin{aligned} f_{Ec}(T_{SC}) = &\varepsilon_{PQ} \times [2 \times (\sqrt{S_Q} \times n)] \\ &+ \varepsilon_{PR} \times [2 \times (\sqrt{S_R} \times n)] \end{aligned} \quad (10)$$

2) Block Rectangle:

$$\begin{aligned} f_{Ec}(T_{BR}) = &\varepsilon_{PQ} \times (a_{PQ} \times n) \\ &+ \varepsilon_{PR} \times (a_{PR} \times n) \\ &+ \varepsilon_{QR} \times (b_{QR} \times n) \end{aligned} \quad (11)$$

3) Square Rectangle:

$$\begin{aligned} f_{Ec}(T_{SR}) = &\varepsilon_{PQ} \times [(n - \sqrt{S_R}) \times n] \\ &+ \varepsilon_{PR} \times (\sqrt{S_R} \times n) \\ &+ (\varepsilon_{PQ} + \varepsilon_{PR}) \times A_{PQR,P} \\ &+ (\varepsilon_{PQ} + \varepsilon_{QR}) \times A_{PQR,Q} \\ &+ (\varepsilon_{PR} + \varepsilon_{QR}) \times A_{PQR,R} \end{aligned} \quad (12)$$

4) Straight Line:

$$\begin{aligned} f_{Ec}(T_{SL}) = &(\varepsilon_{PQ} + \varepsilon_{PR}) \times A_{PQR,P} \\ &+ (\varepsilon_{PQ} + \varepsilon_{QR}) \times A_{PQR,Q} \\ &+ (\varepsilon_{PR} + \varepsilon_{QR}) \times A_{PQR,R} \end{aligned} \quad (13)$$

The predictive accuracy of these formulas is experimentally validated in the following sections. One interesting implication of this theoretical communication energy model is that performance optimal partitions are not necessarily energy optimal. This finding is also experimentally validated in the following sections.

C. EXPERIMENTAL METHODOLOGY OF ENERGY MEASUREMENT

We follow a detailed energy measurement methodology [32]–[34] to ensure the reliability of experimental results. A Watts Up Pro power meter is used to measure the energy. The Watts Up Pro power meter is connected with HCLServer01 through a data cable to one of the USB ports of the server. The data is collected from power meter using the USB interface by a script written in Perl. The script consumes insignificant power and it execution is non-intrusive. The Watts Up Pro power meter is periodically calibrated using the ANSI $C12.20$ revenue-grade power meter, Yokogawa $WT310$. The maximum sampling speed of Watts Up Pro power meters is one sample every second. The accuracy specified in the data-sheets is $\pm 3\%$. The minimum measurable power is 0.5 watts. The accuracy at 0.5 watts is ± 0.3 watts. The power meter provides the total power consumption

of the server. This total energy is the sum of static and dynamic energy consumption. The static energy consumption is calculated by multiplying the idle power of the platform (without application execution) by the execution time of the application. Whereas, dynamic energy consumption is calculated by subtracting this static energy consumption from the total energy consumption of the platform during the given application execution. For our experiments, we focus purely on dynamic energy consumption because static energy consumption is a constant property of a platform and does not depend on the application configuration whereas dynamic energy consumption is the dominating energy dissipator in the used application and platform.

We use an automated tool HCLWattsUp API [35] to gather the readings from the power meter to determine the dynamic energy consumption during the execution of application. It follows a detailed statistical methodology to ensure the reliability of experimental results. HCLWattsUp API has no extra overhead and therefore does not influence the energy consumption of the application execution.

We carefully design our experiments to accurately measure the energy consumed during the main-memory to main-memory data transfer between the heterogeneous processors, and consider this as communication energy cost. There are three communication links in our experimental platform. One is between GPU and CPU, the second is between CPU and PHI, and the third is between GPU and PHI. In the first step, we calculate the communication energy cost (Joules per GB) of these three links and use this communication energy cost of each link to calculate the model-predicted energy consumption of the communication.

For energy experiments, we use the same application as in the performance experiments and the same three scenarios with different ratios of $S_P : S_Q : S_R$ for each case. In all experiments, matrices are partitioned in proportion to their assumed relative speed between GPU, CPU and PHI into Square Corner, Square Rectangle, Block Rectangle and Straight Line partitioning shapes, and the energy consumed by the application for each partitioning shape is measured. This measured energy consumption is then compared with the predicted communication energy consumption, calculated using the energy model of communication.

We make sure that the server is fully reserved and dedicated to our experiments only. We also monitor its load and check for any abnormal event that results in drastic fluctuation. We followed a strict experimental methodology to eliminate the potential contribution by other components such as SSD (Solid State Drives), fans, etc. when measuring the communication energy consumption. We also set the fans at full speed before executing the application to rule out the contribution of fans in dynamic energy consumption. To obtain a data point, we repeatedly execute the application until the sample mean lies in the 95% confidence interval and 0.025 (2.5%) precision has been achieved. To determine the sample mean, we used Student's t-test which assumes that the individual observations are independent and their population follows the normal distribution, and we check that this is the case for each population. We also allow sufficient time to elapse between application successive runs to make sure that cache effects and pipelining do not happen.

D. ENERGY EXPERIMENTS

Same as in the performance experiments, although our energy model of communication only assumes synchronous communications, we perform energy experiments with both synchronous and asynchronous communications. The problem size $N \times N$ is also set to same 22528×22528. On HCLServer01, the actual measured energy cost of the CPU-GPU link is 4.30 Joule/GB, the CPU-PHI link is 12.69 Joule/GB, and the GPU-PHI link is 62.60 Joule/GB. We use this energy cost of each communicating link to predict the communication energy cost by our energy model of communication for the three optimal and 1D partitioning shapes. We then compare the predicted communication energy cost with real experimental results for these partitioning shapes.

The first set of experiments assumes a speed ratio of 1.0 : 0.5 : 0.25 between GPU, CPU and PHI respectively, which is the actual configuration of our platform. Our partitioning solutions aim to balance the workload distribution between the processors during the execution of the application. Therefore, the energy consumption by the computation, which is the product of the aggregate power, consumed by the three processors, and the computation time, is supposed to be same for different partitions. Thus, the difference in the total energy consumption for different partitions will only be due to the difference in the communication energy consumption. This is confirmed by our experiments with the actual speed ratios of the heterogeneous processors in our platform shown in Fig. 8. Therefore, we can safely exclude the computation energy and only consider the communication energy when experimentally comparing of the energy optimality of the partitions.

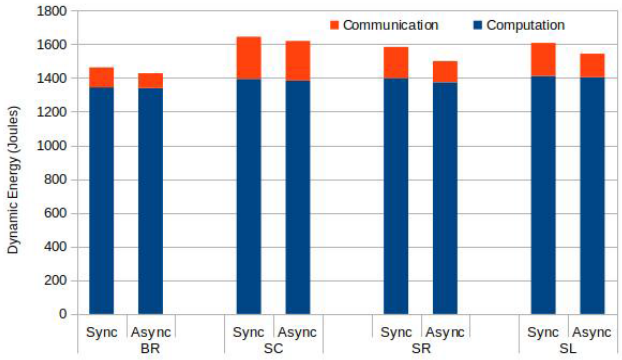


FIGURE 8. Total energy cost (computation + communication) of 4 partition shapes for synchronous and asynchronous communication experiments for the problem size of 22528×22528 and speed ratio of $S_P : S_Q : S_R = 1.0 : 0.5 : 0.25$.

Our energy model predicts that the energy optimal partition is the Block Rectangle (*BR*) for this speed ratio. Subsequently, the experiments that incorporate the synchronous and asynchronous communications also confirm that the optimal

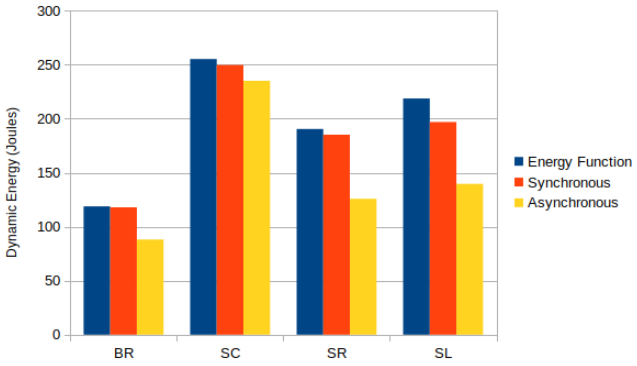


FIGURE 9. Model-predicted and measured energy cost of synchronous and asynchronous communication experiments for the problem size of 22528 × 22528 and speed ratio of $S_P : S_Q : S_R = 1.0 : 0.5 : 0.25$.

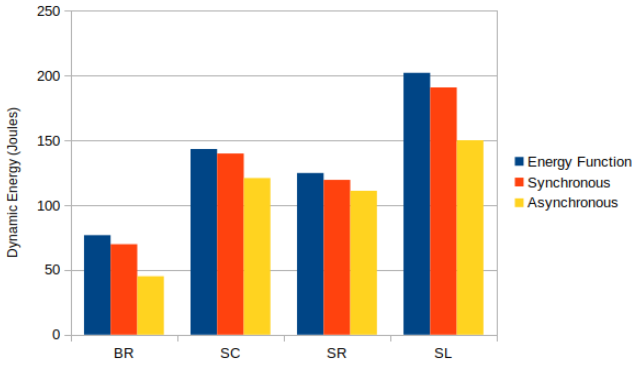


FIGURE 11. Model-predicted and measured energy cost of synchronous and asynchronous communication experiments for the problem size of 22528 × 22528 and speed ratio of $S_P : S_Q : S_R = 1.0 : 0.7 : 0.10$.

partitioning with the least energy consumption is Block Rectangle as shown in Fig. 9. Note that for this scenario, BR has been proved performance optimal as well.

The second set of experiments assumes a speed ratio of $1.0 : 0.15 : 0.10$ for $S_P : S_Q : S_R$. The Square Corner (SC) is predicted to be optimal by the energy model, which is validated by the real measurements in both synchronous and asynchronous communication experiments Fig. 10. SC has also been proved performance optimal for this scenario.

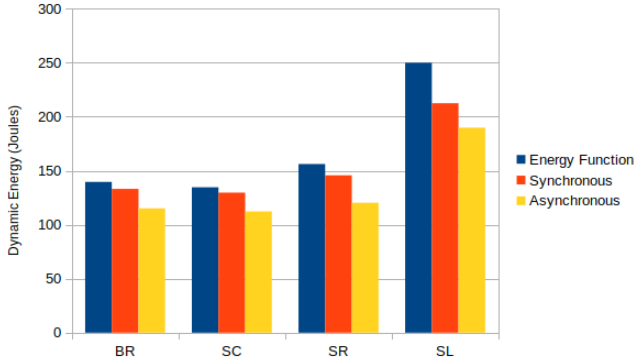


FIGURE 10. Model-predicted and measured energy cost of synchronous and asynchronous communication experiments for the problem size of 22528 × 22528 and speed ratio of $S_P : S_Q : S_R = 1.0 : 0.15 : 0.10$.

Our last set of experiments assumes a speed ratio of $1 : 0.7 : 0.10$ for $S_P : S_Q : S_R$. For this case, Blocked Rectangle (BR) is identified as the energy optimal shape as shown in Fig. 11, both by the energy model and the experiments. However, the performance model and performance experiments have proved Square Rectangle (SR) to be performance optimal, not BR.

Thus, the experiments demonstrate that our energy model of communication is able to correctly predict the relative energy efficiency of partitions. In terms of Joules, the predictive accuracy is good for synchronous communications, which are assumed in the model, and satisfactory for asynchronous. In scenario 3 of the performance and energy experiments, Square Rectangle (SR) proves performance optimal but Block Rectangle (BR) – energy optimal. Hence, we can conclude that performance optimal partitions are not necessarily energy optimal too.

VIII. DISCUSSION

In this work, we solved the problem of optimal partitioning of a square computational domain in a given proportion between three processors, minimizing the total amount of communicated data. We proved that depending on the proportion, the optimal partition will be one of the three described in the paper. While motivated by optimization of data parallel applications on modern heterogeneous hybrid servers, our solution, however, does not take into account the intrinsic heterogeneity of data links between devices in such platforms. Therefore, we conducted experiments on a modern hybrid server, integrating three heterogeneous devices, to study the applicability of our theoretical solutions in practice. The experiments showed that in the case of moderately heterogeneous data links (9.7, 6.3 and 3.6 GB/s), the model-predicted performance, calculated using the average bandwidth of the data links of 6.6 GB/s, appeared accurate enough to correctly select the partition minimizing the communication time. Moreover, our preliminary experiments show that incorporation of the heterogeneity of data links in the communication performance model will significantly improve its predictive accuracy. This motivates us to study the extended partitioning problem where the cost of moving data between processors can be different for different pairs of processors.

It should be noted that our mathematical solution will return the optimal partition, minimizing the communication cost, for any given proportion, not necessarily load balanced. This is important as the state-of-the-art methods of minimization of the computation time of data parallel applications on modern multicore platforms often return load-imbalanced workload distributions [18]–[20], but our algorithm will be able to minimize their communication cost as well.

Two main results of our study of the energy cost of communication are as follows. First, our relatively straightforward theoretical energy model of communication appeared accurate enough to correctly select the energy optimal partition.

Second, we found that the performance optimal partition is not necessarily energy optimal. Indeed, for the ratio $S_P : S_Q : S_R = 1 : 0.7 : 0.1$, the Blocked Rectangle (BR) partition was the energy optimal but the Square Rectangle (SR) one – performance optimal. This is explained by a high level of heterogeneity in energy efficiency of data links on our experimental platform. While their performance heterogeneity of 9.7 : 6.3 : 3.6 is rather moderate, the energy heterogeneity of 4.30 : 12.69 : 62.9 is much higher. Therefore, although the SR partition minimizes both the amount of data moved between the processors and the communication time, the amount of data moved through the most energy expensive link will almost triple in comparison with the BR partition, resulting in a surge of energy consumption. To the best of our knowledge, this is the first work studying the impact of data partitioning on the energy communication cost of the application. While there have been some works on energy optimization through workload partitioning [30], [31], they only address the energy cost of *computation* and do not consider the energy cost of *communication*.

IX. ONGOING WORK AND FUTURE DIRECTION

In this paper, we solved the problem of communication optimal partitioning of a square computation domain over three heterogeneous processors, We also proposed an energy model of communication predicting the dynamic energy consumption for a partition, and designed and performed experiments to validate the model. We found out that this energy model predicts, and the experiments confirm, that the performance-optimal partition does not necessarily have to be energy optimal.

The next step would be to solve the problem of communication optimal partitioning of a square computational domain over three heterogeneous processors when taking into account the heterogeneity of the communicating links. We are currently working on this extended problem of heterogeneous communication links and in the process of formulating a solution by taking into account the bandwidth of communication links between the heterogeneous processors.

REFERENCES

[1] A. Kalinov and A. Lastovetsky, "Heterogeneous distribution of computations while solving linear algebra problems on networks of heterogeneous computers," in *High-Performance Computing and Networking* (Lecture Notes in Computer Science), vol. 1593. Apr. 1999, pp. 189–200.

[2] A. Kalinov and A. Lastovetsky, "Heterogeneous distribution of computations solving linear algebra problems on networks of heterogeneous computers," *J. Parallel Distrib. Comput.*, vol. 61, no. 4, pp. 520–535, Apr. 2001.

[3] 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.

[4] R. A. Van De Geijn and J. Watts, "SUMMA: Scalable universal matrix multiplication algorithm," *Concurrency, Pract. Exper.*, vol. 9, no. 4, pp. 255–274, Apr. 1997.

[5] B. Becker and A. Lastovetsky, "Matrix multiplication on two interconnected processors," in *Proc. IEEE Int. Conf. Cluster Comput.*, Sep. 2006, pp. 1–9.

[6] B. A. Becker and A. Lastovetsky, "Towards data partitioning for parallel computing on three interconnected clusters," in *Proc. 6th Int. Symp. Parallel Distrib. Comput. (ISPDC)*, Jul. 2007, p. 39.

[7] A. DeFlumere, A. Lastovetsky, and B. A. Becker, "Partitioning for parallel matrix-matrix multiplication with heterogeneous processors: The optimal solution," in *Proc. IEEE 26th Int. Parallel Distrib. Process. Symp. Workshops*, May 2012, pp. 125–139.

[8] A. DeFlumere and A. Lastovetsky, "Searching for the optimal data partitioning shape for parallel matrix matrix multiplication on 3 heterogeneous processors," in *Proc. IEEE Int. Parallel Distrib. Process. Symp. Workshops*, May 2014, pp. 17–28.

[9] O. Beaumont, B. A. Becker, A. DeFlumere, L. Eyraud-Dubois, T. Lambert, and A. Lastovetsky, "Recent advances in matrix partitioning for parallel computing on heterogeneous platforms," *IEEE Trans. Parallel Distrib. Syst.*, vol. 30, no. 1, pp. 218–229, Jan. 2019.

[10] T. Malik and A. Lastovetsky, "Optimal matrix partitioning for data parallel computing on hybrid heterogeneous platforms," in *Proc. 19th Int. Symp. Parallel Distrib. Comput. (ISPDC)*, Jul. 2020, pp. 5–8.

[11] J. A. Rico-Gallego, J. C. Díaz-Martín, R. R. Manumachu, and A. L. Lastovetsky, "A survey of communication performance models for high-performance computing," *ACM Comput. Surveys*, vol. 51, no. 6, pp. 1–36, Feb. 2019.

[12] A. Lastovetsky and R. Reddy, "Data partitioning with a realistic performance model of networks of heterogeneous computers," in *Proc. 18th Int. Parallel Distrib. Process. Symp.*, 2004, pp. 26–30.

[13] A. Lastovetsky and R. Reddy, "Data partitioning with a functional performance model of heterogeneous processors," *Int. J. High Perform. Comput. Appl.*, vol. 21, no. 1, pp. 76–90, Feb. 2007.

[14] A. Lastovetsky and R. Reddy, "Data partitioning for multiprocessors with memory heterogeneity and memory constraints," *Sci. Program.*, vol. 13, no. 2, pp. 93–112, 2005.

[15] A. Lastovetsky and R. Reddy, "Two-dimensional matrix partitioning for parallel computing on heterogeneous processors based on their functional performance models," in *Proc. Eur. Conf. Parallel Process.* Springer, 2009, pp. 112–121.

[16] A. Lastovetsky and R. Reddy, "Data distribution for dense factorization on computers with memory heterogeneity," *Parallel Comput.*, vol. 33, no. 12, pp. 757–779, Dec. 2007.

[17] A. Lastovetsky and R. Reddy, "Distributed data partitioning for heterogeneous processors based on partial estimation of their functional performance models," in *Euro-Par* (Lecture Notes in Computer Science), vol. 6043. Springer, 2009, pp. 91–101.

[18] A. Lastovetsky, L. Szustak, and R. Wyrzykowski, "Model-based optimization of EULAG kernel on Intel Xeon Phi through load imbalancing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 3, pp. 787–797, Mar. 2017.

[19] A. Lastovetsky and R. Reddy Manumachu, "New model-based methods and algorithms for performance and energy optimization of data parallel applications on homogeneous multicore clusters," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 4, pp. 1119–1133, Apr. 2017.

[20] H. Khaleghzadeh, R. R. Manumachu, and A. Lastovetsky, "A novel data-partitioning algorithm for performance optimization of data-parallel applications on heterogeneous HPC platforms," *IEEE Trans. Parallel Distrib. Syst.*, vol. 29, no. 10, pp. 2176–2190, Oct. 2018.

[21] E. Dovolnov, A. Kalinov, and S. Klimov, "Natural block data decomposition for heterogeneous clusters," in *Proc. Int. Parallel Distrib. Process. Symp.*, Apr. 2003, p. 10.

[22] A. Lastovetsky, "On grid-based matrix partitioning for heterogeneous processors," in *Proc. 6th Int. Symp. Parallel Distrib. Comput. (ISPDC)*, Jul. 2007, p. 51.

[23] A. Fágenschuh, K. Junosza-Szaniawski, and Z. Lonc, "Exact and approximation algorithms for a soft rectangle packing problem," *Optimization*, vol. 63, no. 11, pp. 1637–1663, Nov. 2014.

[24] D. Clarke, A. Lastovetsky, and V. Rychkov, "Column-based matrix partitioning for parallel matrix multiplication on heterogeneous processors based on functional performance models," in *Proc. Eur. Conf. Parallel Process.* Springer, 2011, pp. 450–459.

[25] D. Clarke, A. Ilic, A. Lastovetsky, and L. Sousa, "Hierarchical partitioning algorithm for scientific computing on highly heterogeneous CPU+GPU clusters," in *Eur. Conf. Parallel Process.* Springer, 2012, pp. 489–501.

[26] Z. Zhong, V. Rychkov, and A. Lastovetsky, "Data partitioning on multicore and multi-GPU platforms using functional performance models," *IEEE Trans. Comput.*, vol. 64, no. 9, pp. 2506–2518, Sep. 2015.

[27] O. Beaumont, V. Boudet, A. Legrand, F. Rastello, and Y. Robert, "Heterogeneous matrix-matrix multiplication or partitioning a square into rectangles: NP-completeness and approximation algorithms," in *Proc. 9th Euromicro Workshop Parallel Distrib. Process.*, Feb. 2001, pp. 298–305.

[28] A. DeFlumere and A. Lastovetsky, "Optimal data partitioning shape for matrix multiplication on three fully connected heterogeneous processors," in *Proc. Eur. Conf. Parallel Process.* Springer, 2014, pp. 201–214.
[29] O. Beaumont, L. Eyraud-Dubois, and T. Lambert, "A new approximation algorithm for matrix partitioning in presence of strongly heterogeneous processors," in *Proc. IEEE Int. Parallel Distrib. Process. Symp. (IPDPS)*, May 2016, pp. 474–483.
[30] H. Khaleghzadeh, M. Fahad, R. R. Manumachu, and A. Lastovetsky, "A novel data partitioning algorithm for dynamic energy optimization on heterogeneous high-performance computing platforms," *Concurrency Comput., Pract. Exper.*, vol. 32, no. 21, Nov. 2020.
[31] H. Khaleghzadeh, M. Fahad, R. Reddy Manumachu, and A. Lastovetsky, "Optimization of data-parallel applications on heterogeneous HPC platforms for dynamic energy through workload distribution," in *Euro-Par* (Lecture Notes in Computer Science), vol. 11997. Gottingen, Germany: Springer, Aug. 2019, pp. 320–332.
[32] M. Fahad, A. Shahid, R. Reddy, and A. Lastovetsky, "A novel statistical learning-based methodology for measuring the goodness of energy profiles of applications executing on multicore computing platforms," *Energies*, vol. 13, p. 22, Aug. 2020.
[33] M. Fahad, A. Shahid, R. Manumachu, and A. Lastovetsky, "A comparative study of methods for measurement of energy of computing," *Sci. Found. Ireland*, vol. 12, p. 2204, Jan. 2019.
[34] M. Fahad, A. Shahid, R. R. Manumachu, and A. Lastovetsky, "Accurate energy modelling of hybrid parallel applications on modern heterogeneous computing platforms using system-level measurements," *IEEE Access*, vol. 8, pp. 93793–93829, 2020.
[35] HCL. (2016). *HCL WattsUp: API for Power and Energy Measurements Using Wattsup Pro Meter*. [Online]. Available: http://git.ucd.ie/hcl/hclwattsup

TANIA MALIK received the M.S. degree from the Virtual University of Pakistan, and the Ph.D. degree from the School of Computer Science, University College Dublin, Ireland, in 2017. She is currently a Research Fellow with the Heterogeneous Computing Laboratory (HCL), University College Dublin. Her research interests include high performance heterogeneous computing, parallel/distributed computing, performance optimization, and energy-efficient computing.

ALEXEY LASTOVETSKY received the Ph.D. degree from the Moscow Aviation Institute, in 1986, and the D.Sc. degree from the Russian Academy of Sciences, in 1997. He has published over 150 articles in refereed journals, edited books, and international conferences. He has authored two monographs–*Parallel Computing on Heterogeneous Networks* (Wiley, 2003) and *High Performance Heterogeneous Computing* (with J. Dongarra, Wiley, 2009). His research interests include high performance heterogeneous computing and energy efficient computing.