Received: 13 October 2020 | Revised: 9 February 2021 | Accepted: 25 February 2021

DOI: 10.4218/etrij.2020-0393

ORIGINAL ARTICLE

ETRI Journal WILEY

Multi-communication layered HPL model and its application to GPU clusters

Young Woo Kim[1] | **Myeong-Hoon Oh**[1] | **Chan Yeol Park**[2]

[1]Artificial Intelligence Research Laboratory, Electronics and Telecommunications Research Institute, Daejeon, Rep. of Korea

[2]Center for Development of Supercomputing System, Korea Institute of Science and Technology Information, Daejeon, Rep. of Korea

Correspondence
Young Woo Kim, Artificial Intelligence Research Laboratory, Electronics and Telecommunications Research Institute, Daejeon, Rep. of Korea.
Email: bartmann@etri.re.kr

Funding information
This work was supported by the Creative Allied Project program of National Research Council of Science & Technology (NST), Rep. of Korea (CAP-17-KISTI, Development of heterogeneous many-core hardware systems for a next-generation high-performance computer).

High-performance Linpack (HPL) is among the most popular benchmarks for evaluating the capabilities of computing systems and has been used as a standard to compare the performance of computing systems since the early 1980s. In the initial system-design stage, it is critical to estimate the capabilities of a system quickly and accurately. However, the original HPL mathematical model based on a single core and single communication layer yields varying accuracy for modern processors and accelerators comprising large numbers of cores. To reduce the performance-estimation gap between the HPL model and an actual system, we propose a mathematical model for multi-communication layered HPL. The effectiveness of the proposed model is evaluated by applying it to a GPU cluster and well-known systems. The results reveal performance differences of 1.1% on a single GPU. The GPU cluster and well-known large system show 5.5% and 4.1% differences on average, respectively. Compared to the original HPL model, the proposed multi-communication layered HPL model provides performance estimates within a few seconds and a smaller error range from the processor/accelerator level to the large system level.

KEYWORDS
GPU cluster, GPU model, HPL, Linpack, mathematical model, multi-communication layered model

1 | INTRODUCTION

Supercomputing or high-performance computing (HPC) is a computing systems area that requires strong computation capabilities to solve scientific and/or engineering problems. These types of computing systems typically consist of hundreds or thousands of computing and storage nodes connected via complex network equipment to create a cluster system. In the initial system design stage, estimating the overall system performance in various areas is important because it affects several system characteristics, including the scale of the system, required performance of each node, speed, and topology of the interconnection network, storage capacity, and file system performance. Therefore, one of the most important factors in determining the system configuration of supercomputers and HPC systems is performance. To accurately evaluate system performance, system designers and engineers perform simulations and estimations based on various sets of performance benchmarks and mathematical models.

High-performance Linpack (HPL) is the most well-known benchmark for evaluating and estimating the capabilities of computing systems [1–4]. HPL has been used as the standard for comparing the performance of computing systems since the early 1980s, and the results of the HPL benchmark

are used as a common metric for measuring the most powerful computing systems in the world [5]. The HPL benchmark is a linear solver program that calculates the solution to an $N \times N$ dense matrix problem. The benchmark calculates and measures the number of floating-point operations performed and the total time required for calculation. Therefore, the HPL benchmark outputs metrics in the form of floating-point operations per second (FLOPS) for computing systems. Since the announcement of LINPACK 100, Linpack has continued to increase in scale and performance as the sizes of target computing systems have grown from LINPAK 100 to LINPACK 1000 and HPL [3].

Over the past few decades, significant breakthroughs and performance enhancements related to processor microarchitectures, memory, and network systems have been achieved based on the development of multicore processors, accelerators, high-bandwidth memory, 100 Gbps networks, and so on. Based on these developments, many researchers have attempted to enhance the HPL benchmark to evaluate state-of-the-art architectures and gain additional FLOPS. HPL algorithms focus on developing and enhancing the basic linear algebra subprogram libraries for Linpack [6], research on efficient block size determination [7], communication overhead obfuscation during lower-upper (LU) factorization [8], and fault tolerance [9]. In [10–13], efficient modifications and adaptations of HPL to general-purpose computing on graphics processing units (GPGPU) were investigated. The roofline model and analysis of the processor performance model [14,15] for various components of HPL have been proposed, and evaluations of commodity systems [16] have been presented.

The skeleton key algorithm and mathematical model of HPL are still maintained and useful, even though the details of their implementations have changed. Basic computation time is proportional to the problem size N in $O(N^3)$ and communication overhead in $O(N^2)$ [2]. The numerical model of original HPL for performance estimation is based and assumed the use of single processor per node and single layer of system network. Based on the simplicity of this model, system designers and engineers can easily estimate the performance of a system at a first glance.

However, this simplicity can lead to system designers and engineers over- or underestimating system performance as technology advances. Recently, it has become common to use multicore processors and/or accelerators, such as GPGPU with multiple layers of communication networks in supercomputers and HPC [17–20]. These communication layers include PCI Express (PCIe) and NVLink within nodes and system interconnection networks.

With the use of accelerators and added communication layers, the error between the measured and numerically estimated HPL performance becomes larger. The numerical model of the original HPL with a simple processor and single communication layer is not sufficient to estimate real-world system performance.

In this paper, to reflect recent developments in processor/accelerator architectures and the multi-layered nature of system interconnection networks, a numerical model for multi-communication layered (MCL) HPL is proposed and evaluated. By adopting multiple communication layers and considering the characteristics of interconnection networks, it is possible to derive more precise and robust results when estimating system performance.

The MCL HPL model contributes to a more accurate estimation of a system's performance, as follows:

- Multiple communication layered modeling method is proposed: The proposed MCL HPL model can reflect communication overhead more precisely for each layer; thus, it can produce less error in estimating the system performance compared to the original HPL model.
- The MCL HPL model can be expanded to any number of communication layers: The proposed methodology can model single or multiple layers and provides flexibility in modeling with enhanced precision.
- An accelerator model is proposed: For the first layer, an abstracted accelerator model is proposed for more precise estimation in multiple communication layers.
- A numerical model that requires multiple communication layers can be modeled by the method used in the MCL HPL model: The proposed modeling method provides clues to decide the problem sizes and boundaries of each communication layer for a given numerical model.

The remainder of this paper is organized as follows. Section 2 briefly discusses the background of the HPL model. The modeling of MCL HPL is described in Section 3. Sections 4 and 5 present some preliminary results and analysis for the proposed MCL HPL model and detailed application results for an experimental GPU cluster system, respectively. Finally, the conclusions and some additional considerations regarding the proposed MCL HPL model are discussed in Section 6.

2 | BACKGROUND OF HPL

The HPL benchmark counts the number of floating-point operations and measures the execution time to calculate the FLOPS of a target system. The number of operations to solve $Ax = b$ for an $N \times N$ dense matrix using Gaussian elimination is known to be $(2N^3/3 + 3N^2/2)$. If a system requires t seconds to solve a given problem, then the ideal performance of the system is $(2N^3/3 + 3N^2/2)/t$ FLOPS [2,3,7]. In a real computing system, the HPL benchmark is executed in parallel with a message-passing method. An entire matrix is divided into

small submatrices, which are distributed to every node in a large system and executed in parallel. During parallel execution, there are many factors affecting ideal performance, such as memory read and write speeds for calculation, copying data to the network device, and data movement among nodes via the network. Therefore, the HPL benchmark algorithm is based on a parallel execution environment, and parallel calculations reflect the system performance considering network data movement and characteristics (initial latency and bandwidth).

In the HPL benchmark, a randomly generated problem matrix A (size of $N \times N$) is processed in units called panels and solved using the LU factorization method. The size of the panels is defined as NB (block size), and the entire problem is divided into N/NB panels in each matrix dimension. To distribute and execute these panels in parallel on a $P \times Q$ processor (or node) grid, the problem matrix A is also divided by P and Q (2D block cyclic fashion) to distribute blocks on the processor grid evenly [7]. LU factorization performs two major operations: panel factorization and updating, which are performed on the problem matrix in the diagonal direction to derive a solution. LU factorization is a key part of the HPL algorithm. The HPL algorithm is designed to solve $Ax = b$, where the problem matrices A and b are given. The equation $Ax = b$ is transformed into $LUx = Pb$ to solve vector x using LU factorization. The equation $LUx = Pb$ is considered as $Ly = Pb$ and $Ux = y$, where P is a row-permutation matrix. The HPL algorithm first finds pivot P for each NB-size panel (pivoting and factorization), broadcasts and exchanges information in the column and row directions, and finally performs permutation to calculate $Ux = y$ (backward substitution). The task is distributed on $P \times Q$ processors and performed in parallel. The operations and performance are thoroughly analyzed and described in [1–3,7,17].

According to [3,7], the numerical model (equation) of the time required for panel factorization t_{pFact} and updating t_{Update} for the ith panel are expressed by the following equations:

$$\begin{aligned} t_{\mathrm{pFact}} &= t_{\mathrm{pFact\_CALC}} + t_{\mathrm{pFact\_COMM}} = \gamma\left(\frac{N - i\times NB}{P} - \frac{NB}{3}\right)\times NB^2 \\ &\quad + (NB(\log P))(\alpha + 2\beta NB) + \left(\alpha + \beta\left(\frac{(N - i\times NB)\,NB}{P}\right)\right), \end{aligned} \tag{1}$$

$$\begin{aligned} t_{\mathrm{Update}} &= t_{\mathrm{Update\_CALC}} + t_{\mathrm{Update\_COMM}} \\ &= \gamma\left(\frac{(N-(i-1)\times NB)\times NB^2}{Q} + \frac{2((N-(i-1))\times NB)^2\,NB}{PQ}\right) \\ &\quad + \left(\alpha(\log P + P - 1) + 3\beta\left(\frac{(N-(i-1)\times NB)\,NB}{Q}\right)\right), \end{aligned} \tag{2}$$

where γ is the floating-point operation rate of the matrix-matrix operation, α is the initial latency of the network, and β is the network speed [3,7]. After factorizing and updating the given problem matrix and considering only the main terms in α, β, and γ, the final numerical HPL model for execution time T_{HPL} is simply expressed as

$$\begin{aligned} T_{\mathrm{HPL}} &= T_{\mathrm{CALC}} + T_{\mathrm{COMM}} = \left(\gamma\frac{2N^3}{3PQ}\right) \\ &\quad + \left(\alpha\frac{N((NB+1)\log P + P)}{NB} + \beta\frac{N^2(3P+Q)}{2PQ}\right), \end{aligned} \tag{3}$$

$$T_{\mathrm{CALC}} = t_{\mathrm{pFact\_CALC}} + t_{\mathrm{Update\_CALC}}, \tag{4}$$

$$T_{\mathrm{COMM}} = t_{\mathrm{pFact\_COMM}} + t_{\mathrm{Update\_COMM}}, \tag{5}$$

where T_{CALC} is the time required for HPL calculation, and T_{COMM} is the total communication time [3].

According to the T_{HPL} equation above, the performance of a system is inversely proportional to the number of problems in $O(N^3)$ from an execution perspective, and the numbers of problems in $O(N^2)$ and $O(N)$ from a data movement perspective. Because data movement through a system network incurs a much higher cost than the calculations themselves, the final performance of a system relies heavily on the performance (latency and bandwidth) of the corresponding network. Additionally, if a system utilizes multiple layers of communication (multiple hierarchies of networks in a system) and the differences in bandwidth between communication layers are large, then the final performance estimation for that system differs from the result provided by (3).

It is not unusual to utilize multiple communication layers in state-of-the-art HPC systems. To fill the gaps in performance estimation between the early mathematical model of HPL and recent computing systems, it is necessary to develop and consider the MCL nature of such computing systems.

3 | MCL HPL MODEL

In this section, a mathematical model and algorithm for MCL HPL are described.

3.1 | Modeling of multi-communication layers for HPL

In this section, a numerical model for MCL HPL is presented and discussed. In modern computing system architectures, there are many communication layers for memory access, communication between processors through the processor bus, between the processor and accelerator via PCIe, among accelerators using proprietary intranetworks, and for system level interconnects such as Infiniband (IB) or Ethernet. These buses, intranetworks, and interconnect

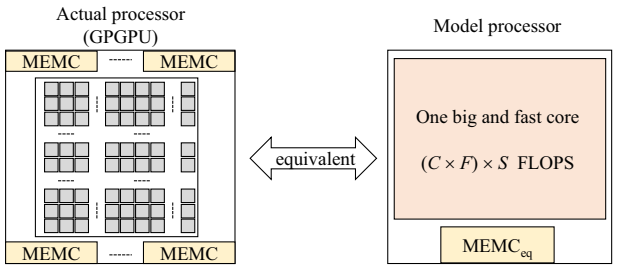
networks can be considered communication networks with different characteristics at different levels. A precise and accurate model must be able to identify and differentiate the characteristics of the communication layers at each level.

3.1.1 | The first layer: memory

Over the past decade, significant breakthroughs in memory technology have been achieved, resulting in high-bandwidth memory (HBM). HBM consists of 3D (or 2.5D) stacked DRAM technology that provides terabit-per-second bandwidths and large bit widths per package [17,18,20]. HBM can be integrated with conventional processors and/or accelerators, though it requires some interconnection silicon components, such as interposers. The use of HBM is growing rapidly, particularly in accelerators such as GPUs. To reflect the recent trends in HPC systems, the proposed model focuses on the use of accelerators (particularly GPUs) and the communication characteristics of HBM in accelerators.

The first communication layer model considers memory as a communication device that moves data from one location to another location, similar to a typical network. Communication in the first-layer comprises memory reads and writes between the accelerator cores and memory. Therefore, the boundary of the first communication layer is limited and assumed to be within an accelerator containing local memory.

Based on these assumptions, the memory access latency and access speed of HBM become the latency of the first layer α_{L1} and speed of the network β_{L1}, respectively. However, applying these parameters directly to an HPL model does not yield an accurate performance estimate because the GPGPU uses multiple HBM packages, and thousands of internal cores access each HBM package simultaneously. In the proposed model, GPGPU is considered to consist of one large, fast core that provides the same theoretical performance as many cores and one equivalent memory controller, as in Figure 1.



Parameters	Actual processor	Model processor
Number of cores	C cores	1 cores
FLOP/core	F FLOP / core	$C \times F$ FLOP / core
Operating frequency	S Hz	S Hz
Number of controller	M controllers	1 controller
Number of QW/memc.	W quad-words (64bits)	$M \times W$ quad-words (64bits)
Operating frequency	H Hz	H Hz

FIGURE 1 An equivalent GPGPU model for the first communication layer

The theoretical memory bandwidth per core in actual GPGPU can be modeled as dividing total memory bandwidth (total memory bits multiplied by operation speed) by the number of cores:

$$\mathrm{BW}_{\mathrm{perCore}} = \frac{(M \times W) \times H}{C}, \tag{6}$$

where $\mathrm{BW}_{\mathrm{perCore}}$ is the memory bandwidth per core, M is the number of memory controllers, W is the number of quad-words (QWs, 64 bits) per memory controller, H is the memory operation frequency, and C is the total number of actual cores. However, each core in the real hardware shares multiple memory controllers; it is not sufficient to apply (6) directly as the bandwidths of model processor. The bandwidth of the equivalent memory controller ($\mathrm{BW}_{\mathrm{Eq}}$) is modeled by multiplying (6) by $(M \times W)$, because one large model core can access $(M \times W)$ QWs simultaneously using one equivalent memory controller:

$$\mathrm{BW}_{\mathrm{Eq}} = \frac{(M \times W)^2 \times H}{C}, \tag{7}$$

where $\mathrm{BW}_{\mathrm{Eq}}$ is the equivalent bandwidth for the model core.

α_{L1} is the access latency, and β_{L1} is the speed of the first layer (memory access), and their final values are expressed using the equivalent bandwidth, as in (8) and (9), respectively.

$$\alpha_{L1} = \frac{\text{number of cyecles for accessing data}}{\mathrm{BW}_{\mathrm{Eq}}}, \tag{8}$$

$$\beta_{L1} = \frac{1}{\mathrm{BW}_{\mathrm{Eq}}}. \tag{9}$$

Generally, the theoretical maximum performance (R_{peak}) of the real GPGPU and the proposed model is calculated by (10).

$$R_{\mathrm{peak}} = C \times F \times S \ [\mathrm{FLOPS}], \tag{10}$$

where C is number of cores, F is number of floating operations per cycle, and S is the operating frequency [3]. By applying α_{L1} and β_{L1} to (1) and (2) and accumulating accordingly, the complete $T_{\mathrm{CALC},L1}$ and $T_{\mathrm{COMM},L1}$ estimates for the first communication layer are expressed as (11) and (12), respectively.

$$\begin{aligned} T_{\mathrm{CALC},L1} = {} & \gamma_{L1} \frac{2N_{L1}^3}{3PQ} + \gamma_{L1} \left(\frac{NB\,(3Q+3P+6)\,N_{L1}^2}{6PQ} \right) \\ & + \gamma_{L1} \left(\frac{NB^2\,((3P+2)-(2P+3)\,Q)\,N_{L1}^2}{6PQ} \right), \end{aligned} \tag{11}$$

$$T_{\mathrm{COMM},L1} = \alpha_{L1}\frac{N_{L1}\left((NB+1)\log P+P\right)}{NB} + \beta_{L1}\frac{N_{L1}^2\left(Q+3P\right)}{2PQ} + \beta_{L1}\frac{N_{L1}\left(NB\left((4\log P-1)Q+3P\right)+8PQ\log P\right)}{2PQ}, \tag{12}$$

where the subscript $L1$ indicates that each corresponding parameter belongs to the first communication layer. $T_{\mathrm{HPL},L1}$ is the sum of (11) and (12). If $\alpha = \alpha_{L1}$, $\beta = \beta_{L1}$, $\gamma = \gamma_{L1}$, and $N = N_{L1}$, the sum of (11) and (12) is simplified to (3).

3.1.2 | The second layer: intranetwork

As supercomputing and HPC systems continue to utilize accelerators such as GPUs and improve the performance of input/output (I/O) interconnection, data movement among accelerators becomes increasingly rapid. PCIe is the de facto standard for I/O subsystems, and the speed of PCIe has been enhanced from 2.5 Gbps per lane to 32 Gbps per lane and is still evolving [19,21–23]. By providing peer-to-peer data movement among I/O devices, modern PCIe provides much greater speeds compared to when it first appeared. If a node supports 16 lanes at a speed of 32 Gbps per lane, then the theoretical speed among PCIe devices is 512 Gbps and the performance exceeds the existing system interconnection (the speed of Infiniband HDR for 4x links is 400 Gbps). In addition to PCIe, NVIDIA provides a proprietary intra-interconnection network called NVLink with a speed of 50 GT/s per lane [23–25].

In the proposed model, these types of intranode interconnection networks are defined as the second communication layer. The characteristics of the second communication layer can be summarized as follow:

- Communication is typically limited to within a node boundary and up to tens of accelerators: The number of accelerators and size of the problem are limited to the boundary of node to obtain best communication performance and reduce hardware and software communication intervention. T_{CALC} and T_{COMM} are applied starting from the boundary of the first layer to the node boundary (the total memory of the accelerators in a node).
- The communication speed exceeds that of conventional interconnection networks: The network is based on I/O bus or intranetwork in a node. The second layer is implemented fully in hardware, including control, to minimize software intervention, and the physical distance between two devices is very short compared to that in a conventional network. Therefore, α and β of the second layer differ from those of a conventional network (more efficient and faster).
- Direct peer-to-peer communication is provided between accelerators: modern I/O bus and intranetwork provide hardware-based peer-to-peer communication to eliminate data copies between I/O devices and main memory and provide direct data movement between devices. The peer-to-peer communication contributes to enhance the communication bandwidth and throughput (so the α and β of the second layer are faster than the conventional network).

The access latency between peer devices is defined in the second layer as α_{L2} and the link speed between peer devices becomes the speed of network β_{L2} for the problem size of N_{L2}. HPL communication time in the second layer is generally expressed by (13) and (14),

$$\begin{aligned} T_{\mathrm{CALC},L2} &= \sum_{\mathrm{start\_of\_}L2}^{\mathrm{end\_of\_}L2}\left(t_{\mathrm{pFact\_CALC}} + t_{\mathrm{Update\_CALC}}\right) \\ &= \alpha_{L2}\frac{N_{L2}-N_{L1}}{NB} \\ &\quad + \beta_{L2}\frac{N_{L2}^2 - 2N_{L2}N_{L1} - \left(N_{L2}-N_{L1}\right)NB + N_{L1}^2}{2P}, \end{aligned} \tag{13}$$

$$\begin{aligned} T_{\mathrm{COMM},L2} &= \sum_{\mathrm{start\_of\_}L2}^{\mathrm{end\_of\_}L2}\left(t_{\mathrm{pFact\_COMM}} + t_{\mathrm{Update\_COMM}}\right) \\ &= \alpha_{L2}\left(\log P + P - 1\right)\frac{N_{L2}-N_{L1}}{NB} \\ &\quad + 3\beta_{L2}\frac{N_{L2}^2 - 2N_{L2}N_{L1} + \left(N_{L2}-N_{L1}\right)NB + N_{L1}^2}{2Q}, \end{aligned} \tag{14}$$

where N_{L1} is problem size of the first layer. By applying Equations (4) and (5) to (13) and (14), the total execution time T_{HPL} for the second communication layer ($T_{\mathrm{HPL},L2}$) can be obtained.

3.1.3 | The third layer and above: conventional network

The third and higher communication layers are conventional system-wide interconnection network layers. For these types of system interconnections in supercomputing and HPC systems, infiniband is the most practical commercial interconnect, owing to its high speed and support of remote direct memory access in hardware. Conventional Ethernet is also widely used in systems, for which communication among applications and nodes is not critical. For top-tier supercomputing systems, to achieve more FLOPS and enhanced system-wide performance, more efficient and high-speed proprietary system interconnects are adopted based on topology concerns, such as TOFUD and Aries Dragonfly [26–29].

The HPL communication time of the third and above layers follows the conventional model and is expressed by (15) and (16) in general form.

$$T_{\mathrm{CALC},Lx} = \sum_{\mathrm{start\_of\_}Lx}^{\mathrm{end\_of\_}Lx} \left(t_{\mathrm{pFact\_CALC}} + t_{\mathrm{Update\_CALC}}\right) = \alpha_{Lx}\frac{N_{Lx} - N_{L(x-1)}}{NB} + \beta_{Lx}\frac{N_{Lx}^2 - 2N_{Lx}N_{L((x-1)} - \left(N_{Lx} - N_{L(x-1)}\right)NB + N_{L(x-1)}^2}{2P}, \quad (15)$$

$$T_{\mathrm{COMM},Lx} = \sum_{\mathrm{start\_of\_}Lx}^{\mathrm{end\_of\_}Lx} \left(t_{\mathrm{pFact\_COMM}} + t_{\mathrm{Update\_COMM}}\right) = \alpha_{Lx}\left(\log P + P - 1\right)\frac{N_{Lx} - N_{L(x-1)}}{NB} + 3\beta_{Lx}\frac{N_{Lx}^2 - 2N_{Lx}N_{L(x-1)} + \left(N_{Lx} - N_{L(x-1)}\right)NB + N_{L(x-1)}^2}{2Q}, \quad (16)$$

where x indicates the corresponding communication layer number, and the access latency of the layer x as α_{Lx} and the speed of network β_{Lx}. The problem size N_{Lx} is the total memory size belonging to layer x, and $N_{L(x-1)}$ is the memory size of the lower layer. $T_{\mathrm{HPL},Lx}$ of arbitrary layer x is the sum of (15) and (16).

3.1.4 | Determination of layer boundaries and generalization

To work with MCL HPL, it is necessary to clearly identify and define the boundaries of each communication layer. Most supercomputing and/or HPC systems have one, two, or three communication layers. The LU factorization calculation and communication time are divided among each communication layer. The size of the first layer is clear because it is bounded by the amount of GPGPU memory. The boundary for the column direction of an $N \times N$ problem matrix is a function of P and Q for HPL. The size of the final communication layer is also obvious and is $N \times N$ in each direction. The intermediate layers depend on the scale of the accelerators and/or nodes in those layers, and their boundaries are somewhat ambiguous. In this paper, the following boundary decision mechanisms are proposed for the MCL HPL model.

3.1.4.1 | Boundary for the first layer

The HPL algorithm divides an $N \times N$ problem matrix into $P \times Q$ submatrices to process on $P \times Q$ processors (or accelerators) in divide and conquer fashion. If an arbitrary size N is not exactly divisible by NB, then some units of the submatrices (that form many $NB \times NB$ panels) are padded with zeros to achieve the correct matrix dimensions. To simplify this process and reduce the complexity of model construction, a new size N_{new} is defined as (17).

$$N_{\mathrm{new}} = NB \times \mathrm{ceil}\left(\frac{N}{NB}\right). \quad (17)$$

The $N_{\mathrm{new}} \times N_{\mathrm{new}}$ matrix is divided to form $P \times Q$ submatrices of size $m_{L1} \times n_{L1}$, as shown in Figure 2. N_{new} is also divided by P and Q in each matrix dimension. The sizes m_{L1} and n_{L1} for the units of the submatrices are defined by (18) and (19) and applied as N_{L1} for $T_{\mathrm{HPL},L1}$.

$$m_{L1} = NB \times \mathrm{ceil}\left(\frac{N_{\mathrm{new}}}{NB \times P}\right), \quad (18)$$

$$n_{L1} = NB \times \mathrm{ceil}\left(\frac{N_{\mathrm{new}}}{NB \times Q}\right). \quad (19)$$

3.1.4.2 | Boundary for the intermediate layers

The problem size and grid assignment in the second layer and above (excluding the final layer) vary according to the system configuration. The size is dependent on the capacity and grid assignment of each new layer, which represent how many accelerators can be equipped within a node and how they are divided into grids in the corresponding layer. The grid assignment for intermediate layers is strongly dependent on the rank assignment of the message-passing interface (MPI) and job distribution, which are not major considerations of the MCL HPL model. Therefore, the proposed MCL model follows the basic HPL P and Q guidelines for these intermediate layers [30], and m_{Lx} and n_{Lx} are applied as N_{Lx} to calculate $T_{\mathrm{HPL},Lx}$.

- If the total ranks in the intermediate layer L_x are equal to p_{Lx} and q_{Lx}, then the subgrid for layer L_x is made as square as possible ($p_{Lx} \le q_{Lx}$).
- If either p_{Lx} or q_{Lx} is a prime number, then we follow the original grid distribution of P and Q.

$$m_{Lx} = NB \times \mathrm{ceil}\left(\frac{N_{\mathrm{new}}}{NB \times p_{Lx}}\right), \quad (20)$$

$$n_{L1} = NB \times \mathrm{ceil}\left(\frac{N_{\mathrm{new}}}{NB \times q_{Lx}}\right). \quad (21)$$

3.1.4.3 | Boundary of the final layer

The last layer includes all the problems of HPL, so the problem size in this layer is equal to N (or N_{new}).

3.1.4.4 | Generalization of MCL HPL

The proposed MCL HPL model calculates the time required for factorization and communication according to the range of each communication layer. By applying the parameters in (8), (9), and (17)–(21) to (1) and (2), T_{HPL} can be derived as shown in (22) for all communication layers.

$$T_{\mathrm{HPL}} = T_{\mathrm{CALC}} + T_{\mathrm{COMM}} = \sum_{j=1}^{x}\left(T_{\mathrm{CALC},Lj} + T_{\mathrm{COMM},Lj}\right), \quad (22)$$

where j is the communication layer index. $T_{\mathrm{CALC},Lj}$ and $T_{\mathrm{COMM},Lj}$ are expressed in (23) and (24).

$$T_{\mathrm{CALC}} = \sum_{j=1}^{x} T_{\mathrm{CALC},Lj} = \sum_{i=1}^{\frac{N_{L1}}{NB}} T_{\mathrm{CALC},L1}\, i + \sum_{i=\frac{N_{L1}}{NB}+1}^{\frac{N_{L2}}{NB}} T_{\mathrm{CALC},L2}\, i + \sum_{i=\frac{N_{L2}}{NB}+1}^{\frac{N_{L3}}{NB}} T_{\mathrm{CALC},L3}\, i + \ldots + \sum_{i=\frac{N_{Lx-1}}{NB}+1}^{\frac{N_{\mathrm{new}}}{NB}} T_{\mathrm{CALC},Lx}\, i\,, \tag{23}$$

$$T_{\mathrm{COMM}} = \sum_{j=1}^{x} T_{\mathrm{COMM},Lj} = \sum_{i=1}^{\frac{N_{L1}}{NB}} T_{\mathrm{COMM},L1}\, i + \sum_{i=\frac{N_{L1}}{NB}+1}^{\frac{N_{L2}}{NB}} T_{\mathrm{COMM},L2}\, i + \sum_{i=\frac{N_{L2}}{NB}+1}^{\frac{N_{L3}}{NB}} T_{\mathrm{COMM},L3}\, i + \ldots + \sum_{i=\frac{N_{Lx-1}}{NB}+1}^{\frac{N_{\mathrm{new}}}{NB}} T_{\mathrm{COMM},Lx}\, i\,, \tag{24}$$

where i is a panel number that is included in the corresponding communication layer j. N_{new} is the total size of problem as in (17), and $N_{L(x-1)}$ is the size of memory of lower layer. In (23), the calculation time is invariant with respect to communication parameters α_{Lx} and β_{Lx}, and γ is the same for all processors or accelerators ($\alpha = \alpha_{Lx}$ and $\beta = \beta_{Lx}$). Therefore, T_{CALC} is as follows:

$$\begin{aligned} T_{\mathrm{CALC}} &= \sum_{j=1}^{x} \left(T_{\mathrm{CALC},Lj}\right) = \sum_{i=1}^{\frac{N_{\mathrm{new}}}{NB}} \left(t_{\mathrm{pFact\_CALC}}(i) + t_{\mathrm{Update\_CALC}}(i)\right) \\ &= \gamma \frac{2N_{\mathrm{new}}^3}{3PQ} + \gamma \left(\frac{NB\,(3Q+3P+6))\,N_{\mathrm{new}}^2}{6PQ} \right) \\ &\quad + \gamma \left(\frac{NB^2\,((3P+2)-(2P+3)\,Q)\,N_{N_{\mathrm{new}}}^2}{6PQ} \right). \end{aligned} \tag{25}$$

In (24), $T_{\mathrm{COMM},Lj}$ consists of three communication components, as in (26). These communication components for pivoting, broadcasting, and updating are defined in (27), (28), and (29), respectively. The communication parameters α_{Lj} and β_{Lj} must be considered in each communication layer j. The total time required for each communication component is generalized and calculated as demonstrated in (26)–(29).

$$T_{\mathrm{COMM},Lj} = T_{\mathrm{pivot\_COMM},Lj} + T_{\mathrm{broadcast\_COMM},Lj} + T_{\mathrm{update\_COMM},Lj}, \tag{26}$$

$$T_{\mathrm{pivot\_COMM},Lj} = NB\,(\log P)\left(\alpha_j + \beta_j\,(2NB+4)\right)\left(\frac{m_j - m_{j-1}}{NB}\right), \tag{27}$$

$$\begin{aligned} T_{\mathrm{boradcast\_COMM},Lj} &= \alpha_j \left(\frac{m_j - m_{j-1}}{NB}\right) \\ &\quad + \beta_j \left(\frac{m_j^2 - 2m_j m_{j-1} - (m_j - m_{j-1})\,NB + m_{j-1}^2}{2P} \right), \end{aligned} \tag{28}$$

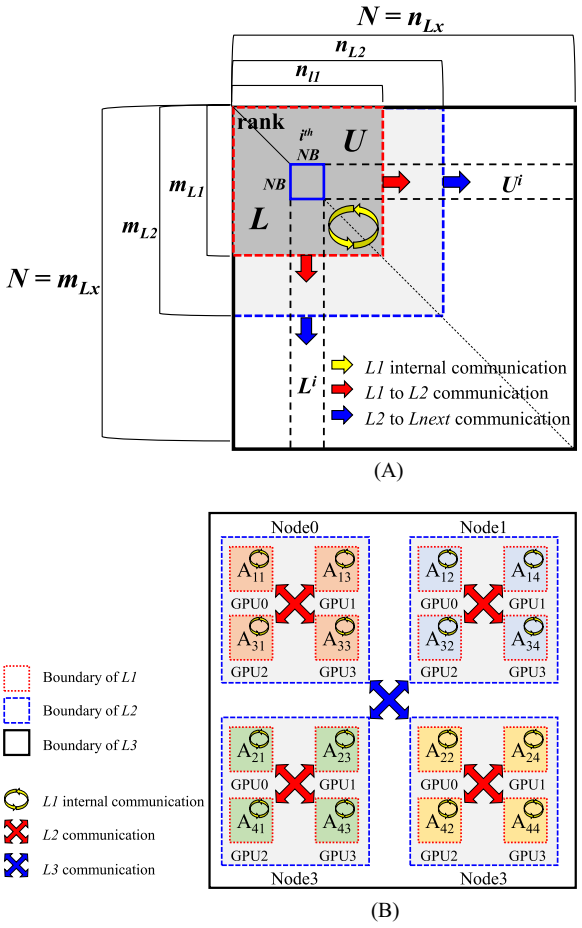


FIGURE 2 Problem matrix boundaries between layers. (A) Represents the concept of overall boundaries respect to total problem size N, and (B) is the representation example for 2×2 ($P \times Q$) block cyclic distribution

$$\begin{aligned} T_{\mathrm{Update\_COMM},Lj} &= \alpha_j\,(\log P + P - 1)\left(\frac{n_j - n_{j-1}}{NB}\right) \\ &\quad + 3\beta_j \left(\frac{n_j^2 - 2n_j n_{j-1} + (n_j - n_{j-1})\,NB + n_{j-1}^2}{2Q} \right), \end{aligned} \tag{29}$$

where m_{Lj} and n_{Lj} are the problem matrix dimensions in the jth layer (refer to Figure 2(A)). The final T_{HPL} value is obtained by summing all the calculation and communication times of all communication layers by applying (22), (25), and (26).

3.1.4.5 | *Verification of MCL HPL*

The proposed MCL HPL model expands communication layers from single layer (the original HPL model) to multi-layers. As calculation time is invariant to the communication parameters, the $O(N^3)$ term in T_{CALC} of the MCL HPL (25) is same for the original equation, (3). For T_{COMM}, if there exists only one communication layer, then (24) can be simplified as (30), which leads to the same results (α and β terms) as (3).

$$\sum_{j=1}^{x}\left(T_{\mathrm{COMM},Lj}\right)=\sum_{i=1}^{\frac{N_{L1}}{NB}}\left(T_{\mathrm{COMM},L1}(i)\right)+\ldots+\sum_{i=\frac{N_{L(x-1)}}{NB}+1}^{\frac{N_{\mathrm{new}}}{NB}}\left(T_{\mathrm{COMM},Lx}(i)\right)$$
$$=\sum_{i=1}^{\frac{N_{\mathrm{new}}}{NB}}\left(T_{\mathrm{COMM},L1}(i)\right). \quad (30)$$

The main difference between the original HPL model and the MCL HPL model comes from the communication parameter differences. The original HPL model takes the communication parameter (α and β) values from the last layer, but the MCL HPL model takes communication parameters from each detailed communication layer. Usually, α and β for the last communication layer are larger than that of lower layers ($\alpha_{\mathrm{last\_layer}} \geq \alpha_{\mathrm{lower\_layer}}$ and $\beta_{\mathrm{last\_layer}} \geq \beta_{\mathrm{lower\_layer}}$), the original HPL model over estimates the communication overhead than the MCL HPL.

4 | SIMULATION USING THE MCL HPL MODEL

The proposed MCL HPL model was implemented and evaluated in this study. For preliminary evaluation, a single-GPGPU system and some well-known large cluster systems were simulated using the proposed MCL HPL model.

4.1 | Single-GPGPU system

The first-layer model in the proposed MCL HPL model is a special model for a processor and/or accelerator and is well suited to GPGPU-like accelerators. To evaluate the first-layer model, a P100 GPGPU system from NVIDIA was modeled, evaluated using MCL HPL model, and measured and compared with real performance measurement according to various problem size.

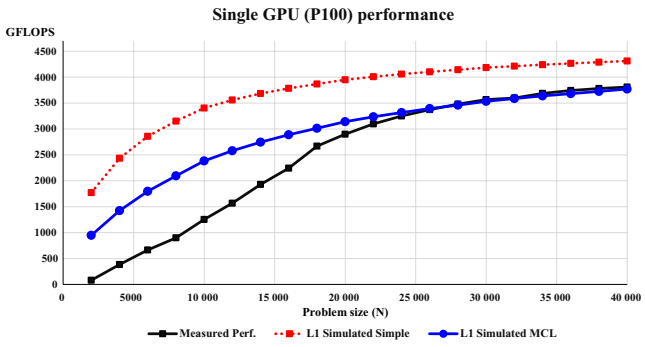


FIGURE 3 Comparisons of the estimated performance of an initial layer-one model (L1 Simulated Simple), the proposed first-layer MCL model (L1 Simulated MCL), and the measured performance of a P100 GPGPU system

Because the first-layer model is based on the equivalent bandwidth of HBM in an accelerator, two types of model evaluation were performed and the results were compared to the measured results. One is the bandwidth parameter calculated as a simple numerical sum ("L1 Simulated Simple"), and the other is the bandwidth parameter calculated by the proposed method ("L1 Simulated MCL," equivalent bandwidth parameter).

The P100 GPGPU system contains four HBM2 packages with a total bandwidth of 732.2 GB/s with 204 MB/s per core [31]. The equivalent bandwidth per core based on the MCL HPL model is 13 GB/s per core. An initial latency of 1029 cycles was considered based on [18]. The total bandwidth was applied and simulated for the L1 Simulated Simple case, and the equivalent bandwidth was applied to the L1 Simulated MCL case. The results are presented in Figure 3. One can see that the proposed MCL HPL case (L1 Simulated MCL) has a much smaller error rate than the L1 Simulated MCL case relative to the measured results. For a problem size of 44 000 (corresponding to 93% of the total HBM memory), the proposed MCL HPL model exhibits a relatively small (−1.07% error, 3840 GFLOPS) compared to the L1 Simulated Simple case (11.99% error, 4347 GFLOPS) relative to the measured performance (3882 GFLOPS).

In Figure 3, there is a relatively large gap between the estimated and measured results, particularly for small problem sizes (up to 25 000). This gap stems from details of the microarchitecture that are not considered in the proposed MCL HPL model. The data and core allocation for problems, cache operation characteristics, memory access patterns, and other architectural details may affect this gap.

However, these errors are generally not problematic because a system typically utilizes the full memory of each GPGPU and the maximum problem size to achieve more FLOPS.

4.2 | Large cluster system

To validate the scalability of the proposed MCL HPL model for large systems, some of the most well-known systems [32–39] from the TOP500 site [5] were modeled and simulated using the MCL HPL and the original HPL model. Table 1 summarizes the reported and estimated performance measures using the original HPL model [3]. Table 2 summarizes the performance measures of the MCL HPL model for multiple- and single-layer cases. The original HPL model calculates the HPL runtime for the major components of the equations. Diff (%) in Table 1 is defined as (31) and ranges from 12.2% to 29.6%, with an average performance overestimation of 20.2%. In Tables 1 and 2, the HPL efficiency (Eff.), which represents the measured or estimated performance (R_{max}) over theoretical performance (R_{peak}), is also listed for each system for comparison.

TABLE 1 Summary of reported and estimated performance measures for the original HPL for well-known systems on TOP500.org [5]

System	# of Nodes	TOP500, June 2020				Original HPL [3]				Network	Diff (%)
		R_{max} (PFLOPS)	R_{peak} (PFLOPS)	N_{max}	Eff. (%)	R_{max} (PFLOPS)	R_{peak} (PFLOPS)	N_{max}	Eff. (%)		
Fugaku [32]	152 064	415.53	513.85	20 459 520	80.9	498.45	513.85	20 486 016	97.0	TOFUD	20.0
Summit [33]	4608	148.60	200.79	16 473 600	74.0	188.21	205.97	16 452 096	91.4	IB EDR	26.7
Sierra [34]	4320	94.64	125.71	11 902 464	75.3	110.08	128.73	11 990 016	85.5	IB EDR	16.3
HPC5 [35]	1820	35.45	51.72	5 750 784	68.5	45.93	51.44	5 746 176	89.3	IB HDR	29.6
Selene [36]	220	27.58	34.57	3 363 840	79.8	32.83	34.31	3 384 192	95.7	IB HDR	19.0
Marconi100 [37]	980	21.64	29.35	2 722 137	73.7	24.27	29.50	2 728 320	82.3	IB EDR Dragonfly	12.2
Pix-Daint [38]	5704	21.23	27.15	3 743 232	78.2	26.42	27.15	3 756 288	97.3	Aries Dragonfly	24.4
DGX SuperPod [39]	96	9.44	11.21	2 519 232	84.3	10.68	11.19	2 478 336	95.5	IB EDR	13.1

TABLE 2 Summary of estimated performance measures for single layers and multiple layers using the proposed MCL HPL model

System	MCL HPL - single communication layer					MCL HPL - multi-communication layers							
										Communication layer			
	R_{max} (PFLOPS)	R_{peak} (PFLOPS)	N_{max}	Eff. (%)	Diff (%)	R_{max} (PFLOPS)	R_{peak} (PFLOPS)	N_{max}	Eff. (%)	1st	2nd	3rd	Diff (%)
Fugaku [32]	487.42	513.85	20 486 016	94.9	17.3	486.88	513.85	20 486 016	94.7	HBM2	TOHUD	-	17.2
Summit [33]	187.94	205.97	16 470 144	91.2	26.5	157.36	205.97	16 470 144	76.4	HBM2	NVLink2	IB EDR	5.9
Sierra [34]	111.56	128.73	11 756 160	86.7	17.9	93.10	128.73	11 756 160	72.3	HBM2	NVLink2	IB EDR	1.6
HPC5 [35]	44.53	51.44	5 722 368	86.6	25.6	37.19	51.44	5 722 368	72.3	HBM2	PCIe G3	IB HDR	4.9
Selene [36]	32.18	34.31	3 368 064	93.8	16.7	27.92	34.31	3 382 272	81.4	HBM2e	NVLink2	IB HDR	1.2
Marconi100 [37]	24.22	29.50	2 728 320	82.1	11.9	17.72	29.50	2 728 320	60.1	HBM2	NVLink2	IB EDR	18.1
Pix-Daint [38]	25.81	27.15	3 756 288	95.1	21.6	21.50	27.15	3 756 288	79.2	HBM2	Aries	Aries	1.3
DGX SuperPod [39]	10.52	11.19	2 478 336	94.0	11.4	8.52	11.19	2 478 336	76.1	HBM2	NVLink2	IB EDR	9.8

$$\text{Diff} = \text{absolute}\left(\frac{R_{\max}\text{ of Original HPL}}{R_{\max}\text{ of TOP500}} - 1\right) \times 100\,[\%]. \quad (31)$$

Table 2 also presents two additional simulation results for the proposed MCL HPL model for each system: one for a single communication layer, and one for up to three communication layers. The MCL HPL model utilizes and calculates the times for each communication layer based on every component of an equation. The Diff (%) column in Table 2 is similarly defined as (32), and for each MCL HPL cases, it decreases compared to the estimation results of the original HPL model in Table 1.

$$\text{Diff} = \text{absolute}\left(\frac{R_{\max}\text{ of MCL HPL}}{R_{\max}\text{ of TOP500}} - 1\right) \times 100\,[\%]. \quad (32)$$

The MCL HPL model with a single communication layer calculates and estimates HPL performance based on the parameters of the main system interconnection network, and the MCL HPL model with multiple communication layers utilizes the communication parameters of each layer (such as HBM2/2e for the first layer and PCIe/NVLink and/or IB EDR/HDR for the second and third layers).

The absolute differences when using the MCL HPL model with a single-layer range from 11.4% to 26.5% with an average performance overestimation of 18.6%. In contrast, the MCL HPL model with multiple communication layers significantly decreases the differences between the real values and model outputs. The absolute differences range from 1.2% to 18.1% with an average value of 7.5%.

Most systems modeled by the MCL HPL model exhibit small differences ranging from 1.2% to 9.8%, excluding the systems from [32,37]. It is assumed that the microarchitectures and/or network topology characteristics differ between systems. For the Fugaku system [32], the processor does not utilize GPGPU and the network applies a 6D torus topology. Therefore, there may be some discrepancies in model performance at the first layer (eg, the equivalent memory bandwidth per core) and second layer (eg, blocking ratio based on the torus topology).

FIGURE 4 HD-PEX, a proprietary high-density multi-GPU hardware subsystem

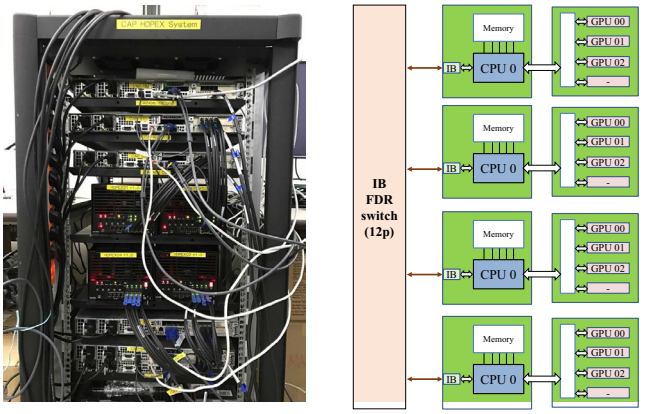


FIGURE 5 HD-PEX-based four-node GPGPU cluster test platform

The Marconi100 system [37] utilizes a GPGPU (V100) and IB EDR DragonFly+. This topology groups several nodes to form a rank, but in the MCL HPL model, this structure is not reflected owing to a lack of information. Excluding these two special cases, the other GPGPU-based systems exhibit small and consistent errors, with an average difference of 4.1% for the MCL HPL (average difference of all listed system is 7.5%).

The network parameters for each layer in our simulations were determined based on the corresponding specifications and experimental results in [14,15,19–28,40–43].

5 | EXPERIMENTS USING MCL HPL ON A GPU CLUSTER SYSTEM

In this section, the experimental results for a GPGPU cluster system are presented and compared to the simulation results of the proposed MCL HPL model.

TABLE 3 Specifications of the HD-PEX multi-GPGPU system

Specifications	Descriptions
Host server Interface	Dual PCIe Gen3 × 16
Expansion HW board	4 × PCIe Gen3 × 16 slots per SLED
Cooling	Air cooled, dual 134 CFM fans per SLED
Power	Dual 1600 W, 1 + 1 redundancy per SLED
Chassis	OCP Rack V1, 2 compliant, 3 OU (Open rack Unit) 537 mm × 800 mm × 141 mm (W × L × H)
SLED enclosure	OCP Rack V1, 2 compliant, 3 OU (Open rack Unit) 175 mm × 800 mm × 138 mm (W × L × H)

TABLE 4 Specifications of the test platform

Specifications	Descriptions
Node	CPU: Xeon E5-2650 v4, 12c @2.2 GHz Memory: 256 GB, DDR4 1867 MHz HD-PEX: 1 × SLED per node Network: Single port IB FDR
Expansion HW	One SLED per node, 3 × P100 per SLED
Network Switch	IB FDR 12 ports
OS	CentOS 7.6
OFED	Mellanox OFED 4.6-1.0.1.1
MPI	OpenMPI 1.10.2
CUDA	Cuda 9.2
HPL	Cuda based HPL

5.1 | Experimental environment setup

To evaluate the proposed MCL HPL model, a GPGPU cluster system based on a proprietary multi-GPU and high-density PCIe expansion system (HD-PEX) was developed and used as a test platform (Figures 4 and 5).

The HD-PEX hardware was developed to utilize multiple GPGPUs for a one- (1 U) or two-rack-unit (2 U) general server by expanding the PCIe bus of the server through high-speed cable assemblies. By using the HD-PEX hardware, a server that lacks GPGPU capability can easily expand and utilize multiple GPGPU systems with various configurations. The HD-PEX hardware has two PCIe connections and can be configured as one of four GPGPU systems per PCIe connection or two GPGPU systems per PCIe connection (up to four GPGPU systems per SLED). In this experiment, three GPGPU systems per PCIe connection were considered.

Tables 3 and 4 list the specifications of the HD-PEX hardware and test platform, respectively. The test platform consists of four x86 nodes, where each node contains three NVIDA P100 GPGPU systems. Therefore, a total of 12 GPGPU systems are used in the test platform.

Two test cases were set up and executed on the test platform.

- Test case A: Multi-GPGPU test on a single node with two communication layers: memory and PCIe
- Test case B: Multi-GPGPU test on multiple nodes with three communication layers: memory, PCIe, and Infiniband FDR.

5.2 | Experimental results for the GPGPU cluster system

Based on the experimental environment and test case setup, measurement and simulation are performed for experimental GPGPU cluster. The experimental results for test case A

TABLE 5 Performance comparisons between GPU clusters and the MCL HPL model

Configurations	Problem size (N)	Measured performance (GFLOPS)	Estimated performance of MCL HPL (GFLOPS)	Difference (%)
1-node				
1 GPGPU (1N1G)	44 000	3882	3840	−1.07
2 GPGPU (1N2G)	62 000	7605	7389	−2.84
3 GPGPU (1N3G)	76 000	10 480	10 715	2.25
4 GPGPU (1N4G)	88 000	13 570	15 464	13.96
2-node				
2 GPGPU (2N2G)	62 000	5878	6229	5.98
4 GPGPU (2N4G)	90 000	14 000	14 847	6.05
6 GPGPU (2N6G)	110 000	21 230	21 306	0.36
8 GPGPU (2N8G)	120 000	25 330	28 681	13.23
3-node				
3 GPGPU (3N3G)	78 000	8403	7556	−10.08
6 GPGPU (3N6G)	110 000	21 460	21 480	0.09
9 GPGPU (3N9G)	130 000	30 980	31 710	2.36
12 GPGPU (3N12G)	152 000	39 960	45 381	13.57
4-node				
4 GPGPU (4N4G)	88 000	14 420	14 677	1.79
8 GPGPU (4N8G)	124 000	26 320	27 621	4.94
12 GPGPU (4N12G)	152 000	40 050	41 077	2.57

represent the absolute differences between estimates using the MCL HPL model with two communication layers and the measured performance values. The absolute differences (absolute value of ((MCL HPL)/Measured) − 1) range from 1.07% to 13.96% with an average value of 5.03% (the average of absolute difference from 1N1G to 1N4G). For test case B (multiple nodes, two to four nodes), the absolute differences between the model and real system range from 0.09% to 13.57% with an average of 5.55% (the average of absolute difference from 2N2G to 4N12G). The performance values and differences are listed in Table 5.

In Table 5, the absolute differences of four GPUs per node configuration (1N4G, 2n8G, and 3N12G in Table 5) show large differences exceeding 13%. The main causes of the absolute differences are suggested below:

- First, in the real HPL benchmark, the MPI is controlled by main CPU and requires some portion of bandwidth of PCIe for each GPU. In GPGPU cluster system test, each GPUs shares single PCIe channel. Because of the additional bandwidth usage for MPI control and communication between the CPU and each GPU, the actual PCIe bandwidth for communication is limited to less than the theoretical PCIe bandwidth. However, the MCL HPL model assumes that the bandwidth of PCIe is fully dedicated to MPI communication only, and there is no interaction between CPU and GPUs. Due to these reasons, the MCL HPL model may over estimates the performance than real cases for four GPUs per node configurations.
- Second, the CPU controls parallel execution and synchronization on each GPU and takes some time to control MPI tasks. This may reduce the performance in real cases, but not in the MCL HPL model.

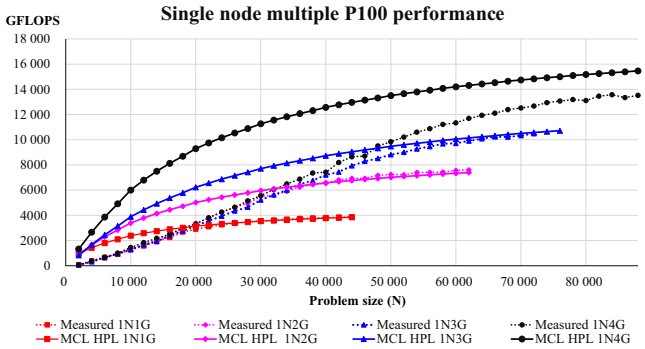


FIGURE 6 Measured and simulated performance for test case A: one-node multi-GPGPU (1, 2, 3, and 4)

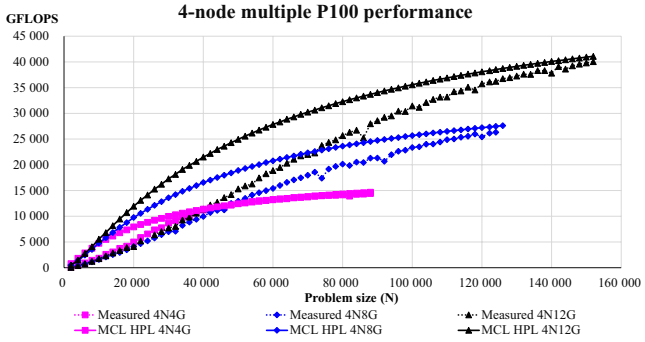


FIGURE 7 Measured and simulated performance for test case B: four-node multi-GPGPU (4, 8, and 12)

In Figures 6 and 7, the test results exhibit similar behavior to those shown in Figure 3 for small problems. Based on a lack of microarchitectural details for the MCL HPL model, the MCL HPL model yields gaps between estimations and measurements, but for the maximum problem size, the difference converges to within a few percentage points.

6 | CONCLUSIONS

In this work, a mathematical model for MCL HPL was proposed, analyzed, modeled, and evaluated. Modern computing systems require more precise and accurate models for the prediction of system performance because the microarchitectures, system architectures, and network architectures of such systems have changed rapidly over the past few decades.

Evaluations and experimental results demonstrated that the proposed MCL HPL model produces reasonable estimates of performance for real systems ranging from individual accelerators to large cluster systems. Based on its ability to model multiple communication layers ranging from memory to hierarchical communication networks, the performance differences between the model results and measured results for the maximum problem size were significantly reduced from 20.2% for the original HPL model to 4.1% for the proposed MCL HPL model for a large cluster system, on average.

Like any numerical model, the proposed MCL HPL model has some advantages and disadvantages. The errors between the proposed MCL HPL model and real systems are significantly reduced as a result of accurately modeling the communication layers. The proposed MCL HPL model can be applied to systems ranging from individual processors/accelerators to large-scale clusters and can handle multiple communication layers simultaneously. However, the proposed MCL HPL model does not support microarchitectures in detail and can introduce estimation gaps for small problems, compared to the maximum possible problem size. To develop a more accurate model, concerns regarding the microarchitectural details of processors and accelerators, as well as network topologies, should be considered in the future.

The experimental results for the proposed MCL HPL model revealed the performance difference of 1.1% for

single-GPGPU case. Moreover, average absolute performance differences of 5.5%, and 4.1% for a wide range of models and problem sizes (GPGPU cluster, and well-known large cluster systems, respectively). The proposed MCL HPL model can simulate large cluster systems with thousands or tens of thousands of nodes within a few seconds with a small error range compared to the original HPL model. It can also adopt future architectural changes in communication layers easily, such as the microarchitectures of next-generation processors, accelerators, and future network systems.

ACKNOWLEDGMENTS

We give special thanks to Hyungon Ryu and Simon See at NVAITC and Wan Seo at NVIDIA for technical supports.

ORCID

Young Woo Kim https://orcid.org/0000-0002-3435-737X

REFERENCES

1. J. J. Dongarra and W. G. Stewart, *LINPACK working note no 15: LINPACK-a package for solving linear systems*, no. ANL-82-30, W-31-109-Eng-38, Springfield, VA, USA, 1982.
2. J. Dongarra, *The LINPACK benchmark: An explanation* in Supercomputing, vol. 297, Springer, Berlin, Heidelberg, 1988, pp. 456-474.
3. J. J. Dongarra, L. Piotr, and P. Antoine, *The LINPACK benchmark: Past, present and future*, University of Tennessee, Technical report, 2001, mimeo.
4. J. J. Dongarra and L. Julien, *The problem with the linpack benchmark 1.0 matrix generator*, Int. J. High Perfom. Comput. Appl. **23** (2009), 5–13.
5. TOP500 The List. Sept. 25, 2020, available at https://www.top500.org/.
6. G. Quintana-Ortí, S. Xiaobai, and H. C. Bischof, *A BLAS-3 version of the QR factorization with column pivoting*, SIAM J. Sci. Comput. **19** (1998), 1486–1494.
7. Z. Wenli, J. Fan, and M. Chen, *Efficient determination of block size NB for parallel LINPACK test*, in Proc. IASTED Int. Conf. Parallel Distrib. Comput. Syst. (Las Vegas, NV, USA), Nov. 2004.
8. T. Nguyen and S. B. Baden, *Lu factorization: Towards hiding communication overheads with a lookahead-free algorithm*, in Proc. IEEE Int. Conf. Cluster Comput. (Chicago, IL, USA), Sept. 2015, pp. 394–397.
9. T. Davies et al., *High performance linpack benchmark: A fault tolerant implementation without checkpointing*, in Proc. Int. Conf. Supercomput. (Tucson, AZ, USA), May 2011, pp. 162–171.
10. M. Fatica, *Accelerating linpack with CUDA on heterogenous clusters*, in Proc. Workshop GPGPU (Washington, DC, USA), Mar. 2009, pp. 46–51.
11. E. Phillips and F. Massimiliano, *A CUDA implementation of the high performance conjugate gradient benchmark*, in International Workshop on PMBS, vol. 8966, Springer, Cham, Switzerland, 2014, pp. 68–84.
12. D. Rohr, J. Cuveland, and V. Lindenstruth, *A model for weak scaling to many GPUs at the basis of the linpack benchmark*, in Proc. IEEE Int. Conf. Cluster Comput. (Taipei, Taiwan), Sept. 2016, pp. 192–202.
13. R. David, K. Matthias Kretz, and B. Matthias, *CALDGEMM and HPL*, Tech. Rep. Dec. 2010, Available at: http://code.compeng.uni-frankfurt.de/attachments/10/techreport.pdf (2010). [last accessed September 2020].
14. R. Milan et al. *D2.2: Report on the ExaNoDe architecture design guidelines*, ExaNode. Tech. Rep. 2016, Available at: https://exanode.eu/wp-content/uploads/2017/04/D2.2.pdf. [last accessed September 2020].
15. A. Kazi, *D2.5: Report on the HPC application bottlenecks of the state-of-the-art HPC platforms*, ExaNode, Tech. Rep. 2016, Available at: https://exanode.eu/wp-content/uploads/2017/04/%20D2.5.pdf. [last accessed September 2020].
16. C. Tom et al., *Emulating high performance linpack on a commodity server at the scale of a supercomputer*, hal-01654804, 2017.
17. D. Zivanovic et al., *Main memory in HPC: do we need more or could we live with less?*, ACM Trans. Architect. Code Optim. **14** (2017), 1–26.
18. Z. Jia et al., *Dissecting the NVIDIA Volta GPU architecture via microbenchmarking*, arXiv preprint, CoRR, 2018, arXiv:1804.06826 (2018).
19. A. Goldhammer and A. Ayer Jr., *Understanding performance of PCI express systems*, Xilinx WP350(v1.2), Sept. 4, 2014.
20. J. Razzaq et al., *Performance characterization of multiprocessors and accelerators using micro-benchmarks*, Int. J. Adv. Syst. Measure. **9** (2016), no. 1–2, 77–90.
21. H. Nakamura et al., *Thorough analysis of PCIe Gen3 communication*, in Proc. Int. Conf. ReConFigurable Comput. FPGAs (Cancun, Mexico), Dec. 2017, pp. 1–6.
22. R. Neugebauer et al., *Understanding PCIe performance for end host networking*, in Proc. 2018 Conf. ACM Special Interest Group Data Commun. (Budapest, Hungary), Aug. 2018, pp. 327–341.
23. A. Li et al., *Evaluating modern GPU interconnect: PCIe, NVLink, NV-SLI, NVSwitch and GPUDirect*, IEEE Trans. Parallel Dist. Syst. **31** (2019), no. 1, 94–110.
24. A. Li et al., *Tartan: Evaluating modern GPU interconnect via a multi-GPU benchmark suite*, in Proc. 2018 IEEE Int. Symp. Workload Charact. (Raleigh, NC, USA), Sept. 2018, pp. 191–202.
25. C. Pearson et al., *Evaluating characteristics of CUDA communication primitives on high-bandwidth interconnects*, in Proc. 2019 ACM/SPEC Int. Conf. Perform. (Mumbai, India), Eng. Apr. 2019, pp. 209–218.
26. Y. Ajima et al., *The tofu interconnect D*, in Proc. IEEE Int. Conf. Cluster Comput. (Belfast, UK), Sept. 2018, pp. 646–654.
27. S. Thomas, Network fabrics: Cray aries, Zuse Institute, Berlin, 2017, Sept. 25, 2020, available at https://support.hlrn.de/twiki/pub/NewsCenter/ParProgWorkshopFall2017/03_Networks_Cray_Aries.pdf
28. D. De Sensi, S. Di Girolamo, and T. Hoefler, *Mitigating network noise on dragonfly networks through application-aware routing*, in Proc. Int. Conf. High Perform. Comput., Netw., Stor. Analysis (Denver, CO, USA), Nov. 2019, pp. 1–32.
29. J. Louis, *Networks for high-performance computing*, Available from: https://louisjenkinscs.github.io/survey/Networks_for_High-Performance_Computing.pdf [last accessed September 2020].
30. M. Sindi, *HowTo–High Performance Linpack (HPL)*, Tech. Rep. Center for Research Computing, University of Notre Dame, Jan. 2009.
31. TechPowerUp, NVIDIA Tesla P100 PCIe 16 GB, Sept. 25, 2020, available at https://www.techpowerup.com/gpu-specs/tesla-p100-pcie-16-gb.c2888.

32. FUGAKU system, RIKEN, Japan, Sept. 25, 2020, available at https://www.r-ccs.riken.jp/en/fugaku/project.
33. SUMMIT system, Oak Ridge National Laboratory, Oak Ridge, Sept. 25, 2020, available at https://www.olcf.ornl.gov/olcf-resources/compute-systems/summit/.
34. Sierra system, Lawrence Livermore National Laboratory, Livermore, Sept. 25, 2020, available at https://computing.llnl.gov/computers/sierra.
35. HPC5 system, Eni, Sept. 25, 2020, available at https://www.eni.com/en-IT/operations/green-data-center-hpc5.html.
36. SELENE system, NVIDIA, Santa Clara, CA, Sept. 25, 2020, available at https://blogs.nvidia.com/blog/2020/06/22/top500-isc-supercomputing/.
37. MARCONI100 system, CINECA, Bologna, Sept. 25, 2020, available at https://www.hpc.cineca.it/hardware/marconi100.
38. PIZ DAINT system, CSCS, Sept. 25, 2020, available at https://www.cscs.ch/computers/piz-daint/.
39. DGX SuperPOD, NVIDIA, Santa Clara, CA, Sept. 25, 2020, available at https://developer.nvidia.com/blog/dgx-superpod-world-record-supercomputing-enterprise/.
40. W. Feng, *Analyzing MPI performance over 10-Gigabit Ethernet*, J. Parallel Distr. Comput. **65** (2005), 1253–1260.
41. S. N. Kandadio and H. Xinghong, Performance of HPC Applications over Infiniband, 10 Gb and 1 Gb Ethernet, IBM, Armonk, NY, USA, 2007.
42. HPC Advisory Council, *Interconnect analysis: 10GigE and infiniband in high performance computing*, HPC Advisory Council, Tech. Rep. 2009.
43. J. Vienne et al., *Performance analysis and evaluation of infiniband FDR and 40GigE RoCE on hpc and cloud computing systems*, in Proc. IEEE Symp. High-Perform. Interconnects (Santa Clara, CA, USA), Aug. 2012, pp. 48–55.

AUTHOR BIOGRAPHIES

Young Woo Kim received his BS, MS, and PhD in electronics engineering from Korea University, Seoul, Rep. of Korea, in 1994, 1996, and 2001, respectively. He was an associate professor at University of Science and Technology, Daejeon, Rep. of Korea, from 2009 to 2019. In 2001, he joined ETRI, Daejeon, Rep. of Korea. He has been researching on high-performance computer system development. His current research interests include high-speed networking and supercomputing system architectures.

Myeong-Hoon Oh received his PhD in information and communication engineering from Gwangju Institute for Science and Technology, Gwangju, Rep. of Korea. In 2005, he joined ETRI, Daejeon, Rep. of Korea. He is a faculty member in Honam University, Gwangju, Rep. of Korea. His current research interest focuses on high-performance computing system and high-speed fabric interconnection design. He has also been an editor for cloud computing in ITU-T SG13 since 2012.

Chan Yeol Park received his BS in Mathematics and MS and PhD in Computer Science from Korea University, Seoul, Rep. of Korea, in 1993, 1995, and 2000, respectively. He joined Supercomputing Center, Korea Institute of Science and Technology Information as a principal researcher since 2002. His research focuses on high-performance computing architecture and parallel/distributed computing algorithm.