Joint Load-Balancing and Energy-Aware Virtual Machine Placement for Network-on-Chip Systems

Xuanzhang Liu
Department of Computer
and Information Sciences
University of Delaware
Newark, DE, 19716
xzliu@udel.edu

Lena Mashayekhy
Department of Computer
and Information Sciences
University of Delaware
Newark, DE, 19716
mlena@udel.edu

***Abstract*—Virtualization is one of the key enabler technologies of cloud computing in providing on-demand sharing of computing resources. Virtualization requires mechanisms and algorithms for virtual resource allocation, virtual machine deployment, migration, and servers consolidation. Most of the existing studies have only focused on how to solve the problem of virtual resource allocation among servers. However, as cloud servers with multi-core architectures become popular, the virtual machine resource allocation in a single server becomes a critical challenge. In this paper, we propose a multi-objective virtual machine placement algorithm by jointly considering energy efficiency and load balancing criteria in a multi-core server with the Network-on-Chip architecture. Our proposed algorithm is based on Markov approximation optimization theory. We perform extensive experiments to evaluate our proposed algorithm. The results show that our proposed algorithm achieves higher energy efficiency, load balancing, and calculation speed compared with the state-of-the-art algorithms.**

***Index Terms*—VM placement; Network-on-Chip; Markov Approximation; Server consolidation**

I. Introduction

Cloud computing has become a promising computing paradigm, which provides a simple pay-as-you-go business model for users. A cloud computing environment may have multiple data centers, and these data centers are interconnected by tens of thousands of high-performance computers and servers. One of the important issues in such a large-scale computing environment is how to effectively allocate resources to users by using virtualization technology. In recent years, managing energy consumption has become a key challenge in data centers, and research shows that it is necessary to take energy saving into account in the process of allocating resources [1], [2]. Server consolidation [3] is becoming an increasingly popular technique to make more efficient use of hardware and computing resources. In server consolidation, virtual machines (VMs) running different applications are deployed on minimum number of physical servers according to an elaborate design of VM placement algorithms. Using this method, unused servers can run on lower power states (e.g., idle or shutdown) to reduce unnecessary energy consumption of the data center. Most existing studies in this domain have focused on how to place VMs or migrate them between servers. However, there is limited information on intra-server VM placement methods and their impacts on the internal energy consumption and communication quality of the server.

In addition to the application-level methods used for server consolidation, the hardware-level supports such as Chip Multiprocessors (CMPs) are also critical for server consolidation [4]. Moreover, emerging technologies such as self-driving cars and industrial Internet of Things require to move computing to the edge in order to obtain low-latency and fast-response-time computing services. This leads to new advancements in designing powerful chips for edge computing such as Intel Xeon D-2100 processor, by integrating several identical processor/cache tiles on a single chip. With increasing number of integrated Intellectual Property (IP) cores and processors, the overall performance of the server has also been greatly improved accordingly, while the amount of communication traffic inside the server is significantly increased. Network-on-Chip (NoC) is a technology of on-chip interconnection network that provides efficient communication schemes for these multicore servers [5]. Compared with the traditional bus structure used for small-scale multicores, NoC uses interconnection network and packet switching technologies to obtain low latency, high performance, and low power consumption, which avoid the competition in the bus architecture and make it more suitable for server consolidation. Therefore, NoC architectures are widely used as the most promising design for the communication platform for many-core systems including cloud computing infrastructures, primarily due to their scalability.

Using virtualization on a NoC, each VM can be held by a single processor core to achieve its functionality. However, the communication between VMs increases the energy consumption of the processors [6]. In addition, with the enhanced capability of the processor cores and a higher degree of virtualization, a single processor core can carry multiple VMs at the same time [7]. For example, a 3GHz CPU could be shared by three small instances in the Amazon EC2 platform. To shield application performance from infrastructure management and performance interference due to VM co-location, resource management in virtualized data centers requires a careful VM placement both on a single chip and a data center to avoid (or minimize) resource contention on diverse physical resources.

Currently, the inter-core communication power consumption has already taken an important part of the total power budget due to the long distance on-chip communication and the ultra-high bandwidth requirement (e.g., tens to hundreds of terabits per second). The intuitive idea to decrease the energy consumption is to place VMs of each application in a relatively close distance to each other. However, placing many VMs on a processor core forms a chip "hotspot". The generation of "hotspot" not only increases the network delay, but also surges the leakage current, which seriously damages system performance. Optimizing the temperature of the system is relatively hard since a key parameter, the resistance matrix, is varied for different topologies [8]. In this study, we demonstrate that the temperature of a processor core and the CPU usage of VMs are related. Therefore to avoid hotspots, we optimize the CPU usage of processor cores by proposing an on-chip VM placement algorithm, which considers both energy efficiency and load balancing criteria.

Our Contribution. We propose an energy-aware and load balancing on-chip VM placement algorithm for cloud computing. We formulate the problem considering these objectives and investigate their combinatorial structures. An approximation algorithm based on a Markov chain model is proposed to solve this multi-objective problem. We run extensive experiments to evaluate the energy consumption and workload of each processor core and analyze the performance of our proposed VM placement algorithm compared with the state-of-the-art algorithms. The results show that our proposed algorithm achieves significantly better performance compared with the existing algorithms.

Organization. The rest of the paper is organized as follows. In Section II, we discuss the state-of-the-art research in this domain. In Section III, we introduce the problem of VM placement in NoC, and we optimally formulate the problem. In Section IV, we present our proposed approximation method and VM placement algorithm. In Section V, we evaluate the proposed VM placement algorithm by extensive experiments. In Section VI, we summarize our results.

II. Related Work

In multi-core systems, there is a body of literature on task mapping to utilize the system efficiently [9]–[12]. Yoosefi et al. [9] proposed a communication-aware mapping of task graphs to the processing reconfigurable cores. Ruggiero et al. [10] combined Integer programming and Constraint programming, and proposed an iterative procedure for task mapping in Multi-Processor System-on-Chip (MPSoC). In [11], an Integer Linear Programming (ILP) model is proposed to optimize the energy yield of MPSoC. All these frameworks are based on a traditional or an improved bus architecture, which are not scalable. Moreover, a bus interconnect architecture allows only one device to obtain the bus usage right at one time. When there are more devices participating in the competition at the same time, the usage right is allocated to one device according to a pre-designed allocation mechanism to prevent bus contention, and other devices can only wait for the current device to release the bus, which causes heavy bottlenecks.

Task or application mapping on a NoC system has attracted considerable attention in recent years. In [13], an ant colony-based heuristic approach is proposed for a task to core mapping in order to reduce the network traffic and communication energy consumption. He et al. [14] proposed an ILP and a heuristic algorithm for a unified task scheduling and core mapping for NoC architectures. However, these studies are not in the context of virtualization and VM placement; moreover, sharing of a processor core by multiple processes is not considered.

The problem of VM placement mainly focuses on how to reasonably allocate cloud resources according to cloud users and cloud service providers by abstracting physical resources (e.g., computing resources, storage resources, and network resources) into uniform virtual resources in order to make them accessible over the Internet. We proposed offline and online mechanisms for VM placement in clouds, federated clouds, and edge computing [15]–[19]. Early placement schemes [8], [20]–[22] only took the CPU capacity of a physical machine and the computational resources of the VM into account, which simplifies the placement problem into only one dimension. That means a set of VMs can be placed on a server if and only if their total CPU load is not greater than the CPU capacity of the server. In reality, both servers and VMs may have multiple cores, and they require other resources such as memory. When a VM is placed on a server, each of the VM's cores must also be placed on one of the server's cores. Careless placement of VMs on cores may lead to performance degradation due to the competition of the hardware resources, especially when the cores in the servers adopt a unique architecture such as NoC.

For the emerging field of virtualization in NoCs, Triviño et al. [23] proposed a partitioning mechanism that enables management and allocation of resources to applications to improve the performance of the applications running simultaneously on a CMP (chip-level multiprocessor). Grot et al. [24] proposed the Kilo-NoC architecture, which guarantees the service requirements of data flows by placing the VMs on a shared CMP. Wang et al. [25] proposed a VM placement algorithm for a heterogeneous multi-core system that exploits different properties of each core to optimize the overall system performance and energy efficiency. Hu et al. [26] presented a VM scheduling model to solve the I/O performance bottleneck based on the multi-core dynamic partitioning. Beechu et al. [27] proposed a system-level mapping of spare cores of tiles of the NoC to enhance the performance. For multicore platforms such as Cyber Physical Systems, Kanduri et al. [28] explored the impacts of application mapping on network contention due to their data flows. All the current studies of on-chip VM placement for multi-core systems mainly target some specific architectures and applications. However, none of the existing studies have focused on general mathematical formulations and optimal solutions for the on-chip VM placement problem considering multiple objectives, which is critical in designing high performance and scalable multi-core systems.

III. VM Placement Model

In this section, we describe our model for the VM placement problem on the NoC architecture. We formulate an optimization problem by jointly considering energy efficiency and load management criteria. A 4×4 2D Mesh topology is shown in Fig. 1.

A. Problem Statement

We consider a NoC architecture composed of N cores. The set of processor cores is denoted by $C = \{C_1, C_2, \ldots, C_N\}$. Each core provides d resources (e.g., CPU, memory), and the capacity of these resources is presented by a d-length vector $\vec{H}^n$ for core C_n.

We consider K applications from users in the form of VMs to be placed on the NoC. Each application k has v_k VMs. As a result, the total number of VMs to be placed is $M = \sum_{k \in K} v_k$, and their set is represented by $V = \{V_1, V_2, \ldots, V_M\}$.

An application k is characterized by $(\mathbf{W}^k, \{\vec{D}_{k,i}\}_{i=1,\ldots,v_k})$, where $\mathbf{W}^k$ is a $v_k \times v_k$ matrix such that its element W_{ij}^k represents the communication traffic between VM V_i and VM V_j belonging to application k. Each element of the vector $\vec{D}_{k,i}$ is a d-tuple representing the d resources required by VM V_i.

For each application k, we need to find c_k feasible host cores to support the physical resource requirements of the application. We define a binary decision variable X_{in} for placing VM V_i on the processor core C_n as follows:

$$X_{in} = \begin{cases} 1 & \text{if } V_i \text{ is assigned to } C_n, \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

Each VM can be only placed on one processor core. Therefore, the following constraint condition must be satisfied:

$$\sum_{n=1}^{N} X_{in} = 1, \quad \forall V_i \in V, C_n \in C. \tag{2}$$

In addition, constraints (3) ensure that the placement of VMs on a core does not violate its capacity:

$$\sum_{k=1}^{K} \sum_{i=1}^{v_k} X_{in} \vec{D}_{k,i} \preceq \vec{H}^n, \quad \forall C_n \in C. \tag{3}$$

B. Optimization Model

In this section, we describe our VM placement model optimizing the CPU loads and minimizing energy consumption.

1) Minimizing Energy Consumption: Energy consumption consists of two parts: static and dynamic. The static energy consumption of a core is influenced by the NoC technology, temperature, and supply voltage. For different placement schemes on a NoC chip, supply voltage and temperature tend to vary within a small range, while the NoC technology remains the same. In addition, the computation energy, as a part of the dynamic energy, is consumed by processor cores for computing the tasks of the applications. This energy remains the same for different placement schemes as the overall tasks are the same and the cores are homogeneous. Unlike these two types of energy consumption (static and computation), communication energy consumption varies dramatically when the placement scheme is changed. Moreover, the communication energy is about 28% of the total energy consumption of a router [29], and for multimedia applications, this can reach to 40% [30]. Therefore, we only focus on the communication energy consumption.

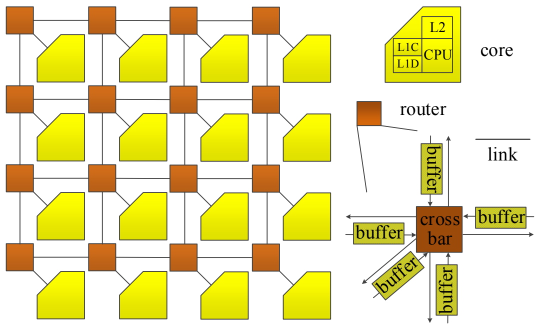

Fig. 1. The architecture of a 4×4 2D Mesh topology

The communication energy consumption is defined as the energy consumption of transmitting data in the system caused by communication requirements of VMs belonging to an application, which includes i) the energy consumption of transferring data through a router and ii) the energy consumption consumed by the transmission links between routers. According to [31], the energy consumption of transmitting 1-bit data is calculated as follows:

$$E_{bit} = E_{bit}^R + E_{bit}^L, \tag{4}$$

where E_{bit}^R and E_{bit}^L are the energy consumption of transferring data through a router and a one-hop inter-router link, respectively. As a result, the energy consumed by transmitting 1-bit data between VM V_i and V_j, where they are placed on core C_n and C_l is:

$$E_{bit}^{n,l} = (d(C_n, C_l) + 1) \times E_{bit}^R + d(C_n, C_l) \times E_{bit}^L, \tag{5}$$

where $d(C_n, C_l)$ is the number of hops between two cores C_n and C_l. Since each core is connected to a router, thus the data is transferred through $d(C_n, C_l) + 1$ routers.

Finally, our objective to minimize the energy consumption is formulated as follows:

$$\min \ E = \sum_{i=1,j=1}^{i=M,j=M} \sum_{n=1,l=1}^{n=N,l=N} E_{bit}^{n,l} \times W_{ij}^k \times X_{in} \times X_{jl} \tag{6}$$

2) Balancing Load: Transistors generate heat during the execution of applications. When a large number of transistors are integrated on a chip, the chip temperature rises rapidly. High chip temperature not only results in leakage current, but also increases the network delay, leading to the system performance degradation. As a result, reducing the chip temperature is an important issue in VM placement. Hung [32] proposed a temperature calculation method, called HotSpot, where the temperature of each core depends on the power consumption and the position of the core on the chip. Let R_{nm} denote the thermal resistance between cores C_n and $C_m \in C$ (for simplicity, we use n and m). The increase of ΔP_n in power consumption of core n leads to temperature rise of ΔT_m at core m. This relationship can be expressed as the following equation:

$$\Delta T_m = R_{nm} \times \Delta P_n. \tag{7}$$

When the system is in a steady state, the temperature of each core can be determined by Eq. (8):

$$\begin{pmatrix} T_1 \\ T_2 \\ \vdots \\ \vdots \\ T_N \end{pmatrix} = \begin{pmatrix} R_{11} & R_{12} & \cdots & \cdots & R_{1N} \\ R_{21} & R_{22} & \cdots & \cdots & R_{2N} \\ \vdots & \vdots & \cdots & \cdots & \vdots \\ \vdots & \vdots & \cdots & \cdots & \vdots \\ R_{N1} & R_{N2} & \cdots & \cdots & R_{NN} \end{pmatrix} \begin{pmatrix} P_1 \\ P_2 \\ \vdots \\ \vdots \\ P_N \end{pmatrix}, \tag{8}$$

where P_n is the power consumed by core n, and T_n is the temperature at core n. The thermal resistance matrix **R** is a constant matrix related to the positions of the cores. This matrix can be obtained from HotSpot [32]. The peak temperature of the cores on the chip is calculated as follows:

$$T_{max} = \max \left\{ T_n \left| n = 1, 2, \cdots, N, \forall C_n \in C \right. \right\}. \tag{9}$$

According to the recent study [8], the power consumption of a core has an approximated linear relationship with the computational resource utilization of the core. Therefore, the power consumption of core C_n is:

$$P_n = \alpha \cdot \bar{L}_n + \beta, \tag{10}$$

where $\bar{L}_n = \sum_k \sum_{i=1}^{v_k} X_{in} \vec{D}_{k,i}(CPU)$ is the sum of CPU consumption by the VMs placed on core C_n. Combined with Eq. (7), the temperature equation can be transferred to:

$$\mathbf{T} = \alpha \mathbf{R} \cdot \bar{\mathbf{L}} + \beta \cdot \mathbf{R}. \tag{11}$$

Since the thermal resistance matrix **R** is deterministic, the peak temperature of the core can be rewritten as:

$$T_{max} = \alpha R \cdot \bar{L}_{max} + \beta \cdot R, \tag{12}$$

where $\bar{L}_{max}$ is the maximum CPU consumption by the VMs on the chip. As a result, in order to minimize the core peak temperature, we focus on minimizing the maximum load of cores on the chip. According to the definition of minimum - maximum load balancing rule by [33], our optimization goal can be expressed as follows:

$$\min \max\nolimits_{1 \le n \le N} \bar{L}_n. \tag{13}$$

Moreover, when VMs of an application are placed on the same core at the same time, the performance is severely degraded due to a competing relationship between resources [34]. As a result, we set the thresholds of CPU utilization to $\bar{L}_n^k \le 90\%$ (following [35]), where $\bar{L}_n^k = \sum_{i=1}^{v_k} X_{in} \vec{D}_{k,i}(CPU)$ represents the sum of CPU consumption by the VMs of application k on core n.

3) Joint Load-Balancing and Energy-Aware VM Placement Model: Based on the two objectives described in this section, we formulate the VM placement problem considering both energy consumption and load balancing as the following optimization problem (called, ELVMP):

$$\begin{aligned} \min \quad & E = \sum_{i=1,j=1}^{i=M,j=M} \sum_{n=1,l=1}^{n=N,l=N} E_{bit}^{n,l} \times W_{ij}^k \times X_{in} \times X_{jl} \\ \min \quad & \max\nolimits_{1 \le n \le N} \bar{L}_n \\ \text{Subject to:} \quad & \sum_{k=1}^{K} \sum_{i=1}^{v_k} X_{in} \vec{D}_{k,i} \preceq \vec{H}^n, \quad \forall C_n \in C \\ & \sum_{n=1}^{N} X_{in} = 1, \qquad \forall V_i \in V, C_n \in C \end{aligned} \tag{14}$$

ELVMP belongs to the quadratic assignment problem, which has already been demonstrated as an NP-hard problem [36]. As a result, we propose an approximation algorithm for the VM placement problem in the next Section.

IV. Approximate VM Placement

In this section, we describe our approximation VM placement model optimizing the CPU loads and minimizing energy consumption, and we present our proposed Markov-based VM placement algorithm.

A. Approximation Method

The ELVMP problem depicted in Eq. (14) is NP-hard [36], meaning that no computationally-efficient algorithm exists to obtain optimal solutions for all cases. Our goal is to find an approximate solution to this problem in a distributed manner. Systems adopting distributed algorithms are more robust than those running centralized algorithms. Moreover, for the Noc architecture, it is quite difficult to set a dedicated core to have the global information of the chip. Inspired by the idea of Markov chain approximation to obtain an approximated solution, we reconstruct the problem of on-chip VM placement to specific log-sum-exp structures.

We define $f = \{X\}$ as a solution (called configuration) of the ELVMP problem, and $\mathscr{F}$ as a set of feasible configurations. Since the log-sum-exp approximation cannot be deployed directly on multiobjective problems according to [37], we reconstruct each objective of the ELVMP in this form.

For the energy-efficiency objective, let p_f^e be the probability associated with configuration $f \in \mathscr{F}$. Then, we define x_f^e as the value of the energy consumption under configuration f.

Therefore, the energy-aware objective (i.e., Eq. (6)) is reconstructed as follows:

$$\begin{aligned} \min_{P^e \geq 0} \quad & \sum_{f\in\mathscr{F}} p^e_f x^e_f + \tfrac{1}{\xi_e} \sum_{f\in\mathscr{F}} p^e_f \log p^e_f \\ s.t. \quad & \sum_{f\in\mathscr{F}} p^e_f = 1, \end{aligned} \tag{15}$$

where ξ_e is a large positive constant.

Different from the standard form, which belongs to L1 norm given in [37], the form of the load-balance objective (i.e., Eq (13)) is an infinite norm. Since each type of norms is a convex function in the linear space, it is feasible to use log-sum-exp approximation for this objective. Therefore, similar to the energy-aware objective Eq (6), the objective Eq (13) is also approximated to the log-sum-exp structure as follows:

$$\begin{aligned} \min_{P_l \geq 0} \quad & \sum_{f\in\mathscr{F}} p^l_f x^l_f + \tfrac{1}{\xi_l} \sum_{f\in\mathscr{F}} p^l_f \log p^l_f \\ s.t. \quad & \sum_{f\in\mathscr{F}} p^l_f = 1 \end{aligned} \tag{16}$$

Therefore, there are 2 independent approximation optimization problems. By combining them into one problem, we define ELVMP-Approx, while we plan to find the Pareto optimal solutions. Therefore, the formulation of ELVMP-Approx is as follows:

$$\begin{aligned} \min_{p\geq 0} \quad & \sum_{f\in\mathscr{F}} \mathbf{p}_f \cdot \mathbf{x}_f + \tfrac{1}{\xi_e} \sum_{f\in\mathscr{F}} p^e_f \log p^e_f + \tfrac{1}{\xi_l} \sum_{f\in\mathscr{F}} p^l_f \log p^l_f \\ s.t. \quad & \sum_{f\in\mathscr{F}} p^e_f = 1, \ \sum_{f\in\mathscr{F}} p^l_f = 1. \end{aligned} \tag{17}$$

For convenience, let $\mathbf{p}_f = (p^e_f, p^l_f)$ and $\mathbf{x}_f = (x^e_f, x^l_f)$ be the vector variables. According to Cauchy-Schwarz inequality, we obtain a relaxation problem, called ELVMP-Re, as follow:

$$\begin{aligned} \min_{P\geq 0} \quad & \sum_{f\in\mathscr{F}} p_f x_f + \frac{1}{\xi} \sum_{f\in\mathscr{F}} p_f \log p_f \\ s.t. & \sum_{f\in\mathscr{F}} p_f = \sqrt{2}, \end{aligned} \tag{18}$$

where $p_f = |\mathbf{p}_f|$, $x_f = |\mathbf{x}_f|$, and $\xi = \sqrt{\frac{1}{(\xi_e)^2} + \frac{1}{(\xi_l)^2}}$.

By introducing the Karush-Kuhn-Tucker (KKT) conditions of ELVMP-Re problem, the optimal solution to ELVMP-Re is given by:

$$p^*_f(x) = \frac{\exp(-\xi x_f)}{\sum_{f'\in\mathscr{F}} \exp(-\xi x_{f'})}, \forall f \in \mathscr{F} \tag{19}$$

B. Markov-based VM Placement Algorithm

To solve ELVMP-Re, we design a reversible Markov chain, which allows a distributed implementation. In order to design a distributed solution, the proposed Markov chain should satisfy the following equilibrium equation, which ensures its convergence to a stationary distribution:

$$p^*_f(\mathbf{x})\, q_{f,f'} = p^*_{f'}(\mathbf{x})\, q_{f',f}, \quad \forall f, f' \in \mathscr{F}, \tag{20}$$

where $q_{f,f'}$ is the transition probability of the current configuration f to the next configuration f'. This equation shows that the transmission probability is only determined by the performance of the current configuration. The transition between the two configurations means changes to the existing virtual machine location (placement). In particular, the reassignment of a virtual machine is a transition to a next configuration, and we set the $q_{f\to f'} \propto \exp^{-1}(\xi x_{f'})$ according to [37]. By doing so, the Markov chain for the static problem is constructed.

Algorithm 1 MVMPA: VM placement algorithm given a fixed number of VMs

```
Input:
  Set of VMs $V$, set of processor cores $C$, and traffic matrix $W$
Output:
  A VM placement solution
 1: /*Initialization*/
 2: Rank all the cores based on their average distance to each other
 3: Calculate the communication traffic for each application
 4: /*Placing the application with the highest communication traffic
    such that its VMs are placed on different cores*/
 5: for $k = 1$ to $K$ do /*$K$ is the number of applications*/
 6:     Calculate the communication traffic for each VM in application $k$
 7:     $placed = \emptyset$, $not\_placed = \{V_1, \ldots, V_{v_k}\}$ /*$placed$ includes
        the VMs that have been placed and $not\_placed$ includes the VMs
        that have not been placed*/
 8:     while $not\_placed \neq \emptyset$ do
 9:         Place VM $V_i \in not\_placed$ with the largest communication
            traffic on the current highest priority core $C_n$
10:         Place the VMs connected to $V_i$ on the cores with the
            shortest distance to $C_n$
11:     end while
12: end for
13: Consider this initial placement of all applications as $f_0$
14: $f_{best} \leftarrow f_0$    /*$f_0$ is the initial feasible configuration*/
15: $x_{best} \leftarrow x_{f_0}$    /*$x_{f_0}$ is the value of the objective for Eq. (18).
    */
16: /*Improving the placement*/
17: for $t = 0$ to $T$ do /*$T$ is the number of iterations*/
18:     Calculate the load of each core
19:     Pick a VM $V_i$ from the highest-load core
20:     $g_i \leftarrow V_i'$s current placement
21:     $f_t \leftarrow f_t \backslash g_i$    /*Remove $V_i$ from the system*/
22:     $\mathscr{G}_i \leftarrow$ FindConfigurations$(f_t, V_i)$
23:     $\mathscr{F}_t \leftarrow \{f_t\} \times \mathscr{G}_i$
24:     Choose a configuration $f \in \mathscr{F}_t$ according to Eq (20)
25:     Reassign VM $V_i$ to a new configuration $f$
26:     if $x_f < x_{best}$ then
27:         $f_{best} \leftarrow f$, $x_{best} \leftarrow x_f$, $f_{t+1} \leftarrow f$
28:     end if
29: end for
30: Return $f_{best}, x_{best}$
```

Algorithm 1 is proposed to find a near-optimal configuration using the equilibrium equation (20). This algorithm keeps track of the observed configurations and finds the best configuration among them as the final solution. Algorithm 1 first selects a feasible configuration as an initial solution. Then, it calculates the load of each core and selects a VM (e.g., V_i) on the highest-load core. The algorithm calls **FindConfigurations** function (given in Algorithm 2) for V_i, which returns $\mathscr{G}$ a set of Δ feasible potential future positions for this VM.

In previous algorithms using the Markov approximation method, an initial solution is usually set by a random pick.

Algorithm 2 FindConfigurations

```
Input:
  Current State $f_t$, VM $V_i$
Output:
  A set $\mathscr{G}$ of $\Delta$ feasible configurations for VM $V_i$ given $f_t$
1: Divide the topology of NoC into four sections
2: Choose section $S$ with the maximum number of VMs communicating with $V_i$
3: $\mathscr{G} \leftarrow \emptyset$
4: while $|\mathscr{G}| < \Delta$ do /*$\Delta$ is the number of feasible configurations*/
5:     Calculate the resource usage of core $C_n$ in section $S$
6:     if $\sum_{k=1}^{K} \sum_{j=1}^{v_k} X_{jn} \vec{D}_{k,i} + \vec{D}_{k,i} \preceq \vec{H}^n$ then
7:         $\mathscr{G} \leftarrow \mathscr{G} \cup C_n$
8:     end if
9: end while
10: Return $\mathscr{G}$
```

However, such a random method will cause a longer time for these algorithms to converge. In order to overcome this limitation, we design our algorithm by taking advantage of the topology used in a NoC system (e.g., mesh). As a result, our proposed algorithm starts with an intelligent initial solution so that the time to find the final result is cut down. For the multi-objective problem, the general method is to find an optimal (or near-optimal) solution for one of the objectives, and then evaluate the current solution iteratively until there is a non-dominated solution (leading to a Pareto-optimal result for the original multi-objective problem). Our proposed algorithm, MVMPA (Markov-based VM Placement Algorithm), optimizes the energy consumption and load-balance for each application first, and then it iteratively decreases the maximum load on the chips to balance the load for the system.

MVMPA receives the set of VMs V, set of processor cores C, and traffic matrix W as inputs. It prioritizes the cores based on their average distance (to all other cores on the NoC) such that the core in the center of the NoC has the highest priority, C_n. For each application, MVMPA algorithm finds the sum of the total communication traffic among its VMs. The application with the highest communication traffic has the highest priority among all the applications, and it will be placed first. For each VM, we calculate the total communication traffic that it has with other VMs belonging to the same application. We place the VM with the highest communication traffic, V_i, on the current highest priority core. Then, all VMs communicating with V_i are placed on the next neighbor cores.

In the initialization step, MVMPA ranks the cores based on their average distance to each other, and calculates the communication traffic among VMs of each application as a priority. MVMPA places the applications bases on their priorities, where an application with the highest communication traffic is placed on the NoC first. VMs belonging to the same application are ranked based on their communication traffic as well. VM V_i with highest communication traffic is picked first, and it is placed on the current highest priority core, C_n. Then, MVMPA places the VMs connected to V_i on the cores with the shortest distance to C_n until all VMs from the application are placed. Thus, each VM of the application is placed on one core in order to make the load of the application balanced. By the end of this step, we have an initial VM placement configuration, f_0.

MVMPA iteratively improves the VM placement. In doing so, it evaluates the load of each core, and selects VM V_i randomly from a core with the maximum load. For this placement of V_i, the energy consumption is calculated as follow: $E_k^i = w_1 d_1 + \ldots + w_{i-1} d_{i-1} + w_{i+1} d_{i+1} + \ldots + w_{v_k} d_{v_k}$, where w_{v_k} is the communication traffic between V_i and V_{v_k} and d_{v_k} represents the Manhattan distance between V_i and V_{v_k}. The optimal placement minimizes E_k^i for each VM. To find a near-optimal solution, MVMPA calls **FindConfigurations** to generate a feasible solution set that can be used for the Markov approximation to evolve. Since w_{v_k} is a constant value, we need to minimize the distance between V_i and any other VM belonging to application k. Recall the NoC topology, the characteristics of a regular topology imply that some of the cores are symmetric. Therefore, **FindConfigurations** divides the topology into four sections, and finds the number of VMs which have communication with V_i in different sections. It then selects a section with the most number of VMs communicating with V_i, where only the cores in that section are included in the feasible solution set. **FindConfigurations** returns a feasible solution set $\mathscr{G}$ containing Δ configurations for reassignment of V_i. Among these potential placements for V_i, MVMPA chooses one of them with probabilities according to Eq. (20). When the state transmissions are done, the current best solution is selected as the final solution.

V. Performance Evaluation

In this section, we evaluate the performance of our proposed algorithm, MVMPA, under different scenarios. Two existing multi-objective VM placement algorithms, VMPACS [8] and MGGA [38], are augmented based on the objectives of the ELVMP problem and are used as benchmarks. VMPACS is an ant-colony based algorithm, and MGGA is derived from a genetic algorithm. Both algorithms are categorized as intelligent algorithms requiring some predefined parameters and a long search time. All algorithms are implemented in C and the experiments are conducted on an iMac with 2.3 GHz Intel Core i5 and 16 GB 2133 MHz DDR4.

The VM instances for each application are generated using similar configurations in [8], where the CPU and memory utilization of each VM are correlated. Based on GT-ITM Rule [39], each VM is communicating with other VMs of the application with a probability of 0.5. The communication traffic demands between VMs are interdependent and subject to normal distributions [22].

Since the mesh topology has plenty of advantages such as high regularity, modularity, and simple routing, most studies on NoC [25], [40], [41] are based on this topology. Therefore, we also employ mesh topology in the experiment. All algorithms are evaluated over the mesh NoC topologies with 64 to 256 cores. The processor cores in the system are homogeneous. The E_{bit}^R and E_{bit}^L are set to be 4.171 nj/bit and

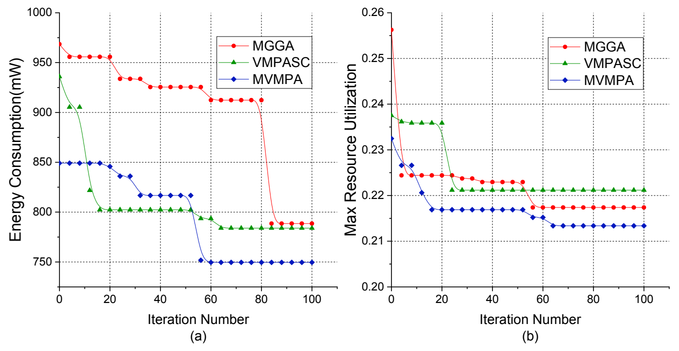

Fig. 2. Comparison of the obtained solutions, where $\overline{D_{CPU}} = \overline{D_{MEM}}$ = 20%

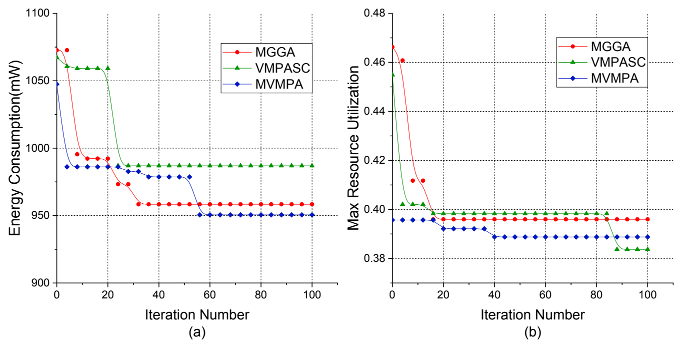

Fig. 3. Comparison of the obtained solutions, where $\overline{D_{CPU}} = \overline{D_{MEM}}$ = 40%

0.449 nj/bit, respectively, according to [42]. Every experiment for each VM placement instance is conducted 20 times, and the average results over these 20 independent runs are reported.

A. Evaluation of MVMPA with different resource demand

We compare the performance of MVMPA with the two benchmark algorithms considering different VM resource requirements. Each VM uses CPU and memory resources, and it is assumed that its CPU and memory demands are linearly correlated [43]. All VMs' utilizations are randomly generated using the method presented in [35] such that the distributions of the generated CPU and memory demands are [0, 40%) with $\overline{D_{CPU}}$ and $\overline{D_{MEM}}$ being 20%, and [0, 80%) with $\overline{D_{CPU}}$ and $\overline{D_{MEM}}$ being 40%. $\overline{D_{CPU}}$ and $\overline{D_{MEM}}$ are the mean value of CPU demand and memory demand respectively indicating the resource intensity of VM requests. All algorithms are evaluated over an 8×8 mesh NoC topology considering communication traffic of VMs follows $N(0.4, 0.1)$ distribution. To make a fair comparison of the algorithms, we fix the total number of VMs of all the applications to a constant 64.

Figs. 2 and 3 show the convergence of the algorithms after 100 iterations. Each point in Figs. 2 and 3 represents the value of energy consumption and resource utilization of the obtained solution, respectively. The results show that all algorithms improve their solutions as they converge, and MVMPA outperforms VMPACS and MGGA algorithms. The initial solution of MVMPA is better than those of MGGA and VMPACS since MVMPA takes into account the characteristic of NoC topology at its initial steps. Moreover, MVMPA achieves faster convergence. The maximum resource utilization is almost doubled in Fig. 3 compared to that of Fig. 2 since the VMs are more resource intensive. This is also the case for the energy consumption. As VMs need to satisfy the application load-balance constraint, they have to be placed on different cores leading to more distance among VMs and a higher energy consumption.

To evaluate the computational efficiency of our proposed algorithm, we also present the execution time of the algorithms. The execution time reflects the time used on searching for the approximate solution after 100 iterations. The results are presented in Table I. The execution time of MVMPA is much smaller since our algorithm uses a heuristic method to decrease the searching space. The results show that our proposed algorithm is computationally efficient.

B. Scalability of MVMPA with respect to topology

This section mainly considers the performance of the algorithms when VMs are placed on different scales of a mesh topology from a 8×8 to a 16×16 NoC while considering communication traffic of VMs follows $N(0.4, 0.1)$ distribution. We also increase the number of VMs from 64 to 256, according. We fix $\overline{D_{CPU}} = \overline{D_{MEM}}$ = 40%. The results are obtained after 100 iterations.

In Fig. 4(a), MVMPA provides about a combined average of 11.65% improvements in both energy efficiency and balancing load compared with MGGA and 5.65% improvements compared with VMPASC. Since MVMPA takes into account the characteristic of the NoC topology and the communication traffic of each application, it searches the solution space more efficiently. In Fig. 4(b), the maximum utilization of the cores increases with respect to the increasing number of VMs. This is due to the fact that the energy consumption will augment sharply when the scale of the topology becomes larger. This makes MVMPA give priority to energy consumption, and thus MVMPA places more VMs on the same core to reduce the total communication traffic on the NoC. The results show that MVMPA is suitable for large-scale NoC topologies.

C. Evaluation of MVMPA with different communication traffic

In this section, we compare the performance of the algorithms when the communication traffic increases among the VMs. The communication

TABLE I
EXECUTION TIME

Reference value	Algorithm	Execution time (s)
$\overline{D_{CPU}} = \overline{D_{MEM}}$ = 20%	MGGA	12.059
	VMPASC	10.523
	MVMPA	2.249
$\overline{D_{CPU}} = \overline{D_{MEM}}$ = 40%	MGGA	18.168
	VMPASC	11.682
	MVMPA	2.520

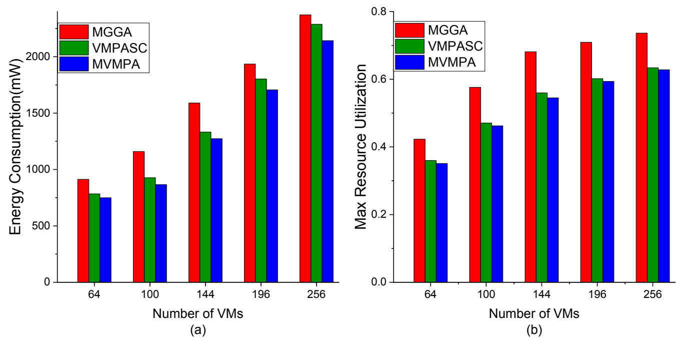

Fig. 4. Scalability of the algorithms with increasing demand

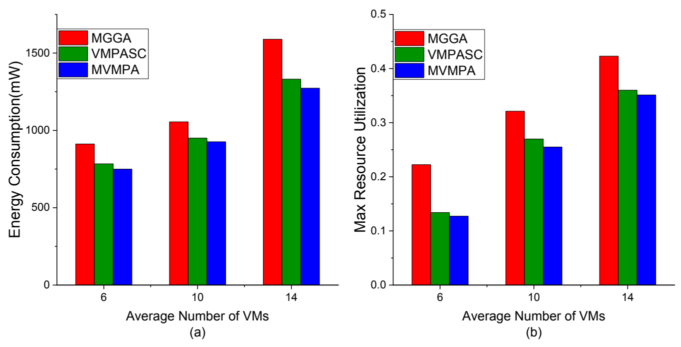

Fig. 6. Performance comparison considering different application sizes

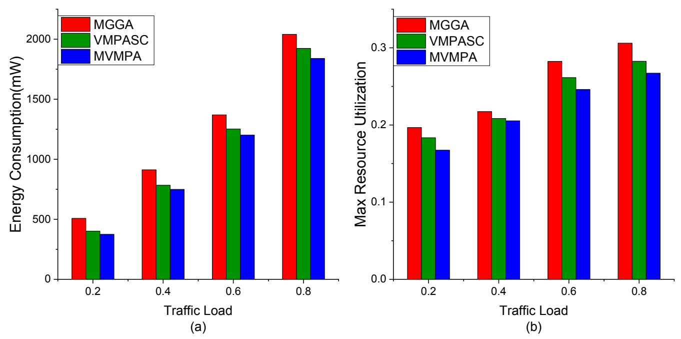

Fig. 5. Performance comparison considering different communication traffic

traffic of VMs is drawn from the following normal distributions: $N(0.2, 0.1)$, $N(0.4, 0.1)$, $N(0.6, 0.1)$, and $N(0.8, 0.1)$ [44]. The increasing mean value of the normal distributions implies that the communication traffic among VMs are getting intensive.

The performance of the algorithms considering different communication traffic is depicted in Fig. 5, where x-axes represent the mean traffic of each VM. The results show a similar trend for the obtained solutions by the algorithms. MVMPA can also be used for traffic intensive scenarios.

D. Evaluation of MVMPA with different application scale

Finally, we study whether the proposed MVMPA algorithm is scalable by considering different application sizes. In these experiments, the number of VMs for each application is varied from 4 to 16, which is generated by using a uniform distribution. These applications are then organized in three scenarios. The other settings remain the same as in subsection B.

Fig. 6 shows the results, where x-axes represent the mean number of VMs in the applications. The MVMPA has a better performance than MGGA and VMPACS for different range of applications. Although the difference between the results of MVMPA and VMPACS is not distinguish, MVMPA converges quicker towards a solution leading to a better performance compared with the existing algorithms.

VI. Conclusion

In this paper, we proposed a load-balancing and energy-aware VM placement for NoCs. We formulated the on-chip VM placement problem as an optimization model and derived the detailed energy model and load model. We designed a Markov-based approximation algorithm, considering the characteristic of the NoC architecture and the communication traffic among the VMs of applications. Compared with the classical multi-objective placement algorithms, the results show that our algorithm obtains much better results in different scenarios including different resource demands of the VMs, NoC topologies, communication traffic between VMs, and scales of the applications. For the future work, we plan to consider the dynamic virtual machine placement on chip.

Acknowledgment

This research was supported in part by NSF grant CNS-1755913.

References

[1] M. P. Mills, "The cloud begins with coal: Big data, big networks, big infrastructure, and big power," *Digital Power Group*, 2013.

[2] L. Mashayekhy, M. Nejad, D. Grosu, Q. Zhang, and W. Shi, "Energy-aware scheduling of mapreduce jobs for big data applications," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 10, pp. 2720–2733, 2015.

[3] D. A. Alboaneen, B. Pranggono, and H. Tianfield, "Energy-aware virtual machine consolidation for cloud data centers," in *Proceedings of the 2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing*, 2014, pp. 1010–1015.

[4] M. R. Marty and M. D. Hill, "Virtual hierarchies to support server consolidation," in *ACM SIGARCH Computer Architecture News*, 2007, pp. 46–56.

[5] M. M. Ahmed, M. S. Shamim, N. Mansoor, S. A. Mamun, and A. Ganguly, "Increasing interposer utilization: A scalable, energy efficient and high bandwidth multicore-multichip integration solution," in *Proc. of the IEEE 8th International Green and Sustainable Computing Conference*, 2017, pp. 1–6.

[6] C. Batten, A. Joshi, V. Stojanovć, and K. Asanović, "Designing chip-level nanophotonic interconnection networks," *Integrated Optical Interconnect Architectures for Embedded Systems*, pp. 81–135, 2013.

[7] C. Xu, S. Gamage, H. Lu, R. R. Kompella, and D. Xu, "vturbo: Accelerating virtual machine i/o processing using designated turbo-sliced core," in *USENIX Annual Technical Conf.*, 2013, pp. 243–254.

[8] Y. Gao, H. Guan, Z. Qi, Y. Hou, and L. Liu, "A multi-objective ant colony system algorithm for virtual machine placement in cloud computing," *Journal of Computer and System Sciences*, vol. 79, no. 8, pp. 1230–1242, 2013.

[9] A. Yoosefi and H. R. Naji, "A clustering algorithm for communication-aware scheduling of task graphs on multi-core reconfigurable systems," *IEEE Transactions on Parallel & Distributed Systems*, no. 10, pp. 2718–2732, 2017.

[10] M. Ruggiero, A. Guerri, D. Bertozzi, F. Poletti, and M. Milano, "Communication-aware allocation and scheduling framework for stream-oriented multi-processor systems-on-chip," in *Proc. of the IEEE Conference on Design, Automation and Test in Europe*, vol. 1, 2006, pp. 6–16.

[11] M. Ghorbani, "A variation and energy aware ILP formulation for task scheduling in MPSoC," in *Proc. of the 13th International Symposium on Quality Electronic Design*, 2012, pp. 772–777.

[12] L. Ghalami and D. Grosu, "Scheduling parallel identical machines to minimize makespan: A parallel approximation algorithm," *Journal of Parallel and Distributed Computing*, pp. –, 2018.

[13] M. Farias, E. Barros, A. Araujo, A. Silva, J. Melo *et al.*, "An ant colony metaheuristic for energy aware application mapping on nocs," in *Proc. of the IEEE 20th International Conference on Electronics, Circuits, and Systems*, 2013, pp. 365–368.

[14] O. He, S. Dong, W. Jang, J. Bian, and D. Z. Pan, "Unism: Unified scheduling and mapping for general networks on chip," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 20, no. 8, pp. 1496–1509, 2012.

[15] L. Mashayekhy, M. Nejad, D. Grosu, and A. Vasilakos, "An online mechanism for resource allocation and pricing in clouds," *IEEE Transactions on Computers*, vol. 65, no. 4, pp. 1172–1184, 2016.

[16] L. Mashayekhy, M. Nejad, and D. Grosu, "Physical machine resource management in clouds: A mechanism design approach," *IEEE Transactions on Cloud Computing*, vol. 3, no. 3, pp. 247–260, 2015.

[17] ——, "A PTAS mechanism for provisioning and allocation of heterogeneous cloud resources," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 9, pp. 2386–2399, 2015.

[18] ——, "A two-sided market mechanism for trading big data computing commodities," in *Proc. of the IEEE International Conference on Big Data*, 2014, pp. 153–158.

[19] N. Sharghivand, F. Derakhshan, and L. Mashayekhy, "QoS-aware matching of edge computing services to internet of things," in *Proceedings of the 37th IEEE International Performance Computing and Communications Conference*, 2018, pp. 1–8.

[20] A. Beloglazov and R. Buyya, "Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers," *Concurrency and Computation: Practice and Experience*, vol. 24, no. 13, pp. 1397–1420, 2012.

[21] D. Ihara, F. L. Pirez, and B. Baran, "Many-objective virtual machine placement for dynamic environments," in *Proceedings of the 2015 IEEE/ACM 8th International Conference on Utility and Cloud Computing*, 2015, pp. 75–79.

[22] T. Yapicioglu and S. Oktug, "A traffic-aware virtual machine placement method for cloud data centers," in *Proceedings of the 2013 IEEE/ACM 6th international conference on Utility and Cloud Computing*, 2013, pp. 299–301.

[23] F. Triviño, J. L. Sánchez, F. J. Alfaro, and J. Flich, "Virtualizing network-on-chip resources in chip-multiprocessors," *Microprocessors and Microsystems*, vol. 35, no. 2, pp. 230–245, 2011.

[24] B. Grot, J. Hestness, S. W. Keckler, and O. Mutlu, "Kilo-noc: a heterogeneous network-on-chip architecture for scalability and service guarantees," in *ACM SIGARCH Computer Architecture News*, 2011, pp. 401–412.

[25] Y. Wang, X. Wang, and Y. Chen, "Energy-efficient virtual machine scheduling in performance-asymmetric multi-core architectures," in *Proceedings of the 8th International Conference on Network and Service Management*, 2012, pp. 288–294.

[26] Y. Hu, X. Long, J. Zhang, J. He, and L. Xia, "I/o scheduling model of virtual machine based on multi-core dynamic partitioning," in *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing*, 2010, pp. 142–154.

[29] A. B. Kahng, B. Li, L.-S. Peh, and K. Samadi, "Orion 2.0: A fast and accurate noc power and area model for early-stage design space exploration," in *Proceedings of the Conference on Design, Automation and Test in Europe*, 2009, pp. 423–428.

[27] N. K. R. Beechu, V. M. Harishchandra, and N. K. Y. Balachandra, "High-performance and energy-efficient fault-tolerance core mapping in NoC," *Sustainable Comp.: Informatics and Systems*, vol. 16, pp. 1–10, 2017.

[28] A. Kanduri, A.-M. Rahmani, P. Liljeberg, and H. Tenhunen, "Predictable application mapping for manycore real-time and cyber-physical systems," in *Proceedings of the 9th IEEE International Symposium on Embedded Multicore/Many-core Systems-on-Chip*, 2015, pp. 135–142.

[30] A. Das, A. Kumar, and B. Veeravalli, "Energy-aware communication and remapping of tasks for reliable multimedia multiprocessor systems," in *Proceedings of the 18th IEEE International Conference on Parallel and Distributed Systems*, 2012, pp. 564–571.

[31] J. Hu and R. Marculescu, "Exploiting the routing flexibility for energy/performance aware mapping of regular NoC architectures," in *Design, Automation and Test in Europe Conference and Exhibition*, 2003, pp. 688–693.

[32] W. Hung, C. Addo-Quaye, T. Theocharides, Y. Xie, N. Vijakrishnan, and M. J. Irwin, "Thermal-aware ip virtualization and placement for networks-on-chip architecture," in *Proceedings of IEEE International Conference on Computer Design: VLSI in Computers and Processors*, 2004, pp. 430–437.

[33] L. Lu, H. Zhang, E. Smirni, G. Jiang, and K. Yoshihira, "Predictive vm consolidation on multiple resources: Beyond load balancing," in *Proceedings of the IEEE/ACM 21st International Symposium on Quality of Service*, 2013, pp. 1–10.

[34] S. Govindan, J. Liu, A. Kansal, and A. Sivasubramaniam, "Cuanta: quantifying effects of shared on-chip resource interference for consolidated virtual machines," in *Proceedings of the 2nd ACM Symposium on Cloud Computing*, 2011, p. 22.

[35] Q. Zheng, R. Li, X. Li, N. Shah, J. Zhang, F. Tian, K.-M. Chao, and J. Li, "Virtual machine consolidated placement based on multi-objective biogeography-based optimization," *Future Generation Computer Systems*, vol. 54, pp. 95–122, 2016.

[36] S. Sahni and T. Gonzalez, "P-complete approximation problems," *Journal of the ACM*, vol. 23, no. 3, pp. 555–565, 1976.

[37] M. Chen, S. C. Liew, Z. Shao, and C. Kai, "Markov approximation for combinatorial network optimization," *IEEE Transactions on Information Theory*, vol. 59, no. 10, pp. 6301–6327, 2013.

[38] J. Xu and J. A. Fortes, "Multi-objective virtual machine placement in virtualized data center environments," in *Proc. of the IEEE/ACM Int'l Conf. on Green Computing and Communications*, 2010, pp. 179–188.

[39] E. W. Zegura, K. L. Calvert, and S. Bhattacharjee, "How to model an internetwork," in *Proc. of IEEE INFOCOM*, vol. 2, 1996, pp. 594–602.

[40] R. Marculescu, U. Y. Ogras, L.-S. Peh, N. E. Jerger, and Y. Hoskote, "Outstanding research problems in noc design: system, microarchitecture, and circuit perspectives," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 28, no. 1, pp. 3–21, 2009.

[41] K. Bhardwaj, W. Jiang, and S. M. Nowick, "Achieving lightweight multicast in asynchronous nocs using a continuous-time multi-way read buffer," in *Proc. of the 11th Eleventh IEEE/ACM International Symposium on Networks-on-Chip*, 2017, pp. 1–8.

[42] C. Wu, C. Deng, L. Liu, J. Han, J. Chen, S. Yin, and S. Wei, "An efficient application mapping approach for the co-optimization of reliability, energy, and performance in reconfigurable NoC architectures," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, no. 8, pp. 1264–1277, 2015.

[43] Y. Ajiro and A. Tanaka, "Improving packing algorithms for server consolidation," in *Int. CMG Conference*, vol. 253, 2007, pp. 399–406.

[44] X. Meng, V. Pappas, and L. Zhang, "Improving the scalability of data center networks with traffic-aware virtual machine placement," in *Proc. of IEEE INFOCOM*, 2010, pp. 1–9.