Enhancing Job Scheduling on Inter-Rackscale Datacenters with Free-Space Optical Links

Yao HU†a), Nonmember and Michihiro KOIBUCHI†, Member

SUMMARY Datacenter growth in traffic and scale is driving innovations in constructing tightly-coupled facilities with low-latency communication for different specific applications. A famous custom design is rackscale (RS) computing by gathering key server resource components into different resource pools. Such a resource-pooling implementation requires a new software stack to manage resource discovery, resource allocation and data communication. The reconfiguration of interconnection networks on their components is potentially needed to support the above demand in RS. In this context as an evolution of the original RS architecture the inter-rackscale (IRS) architecture, which disaggregates hardware components into different racks according to their own areas, has been proposed. The heart of IRS is to use a limited number of free-space optics (FSO) channels for wireless connections between different resource racks, via which selected pairs of racks can communicate directly and thus resource-pooling requirements are met without additional software management. In this study we evaluate the advantages of FSO links on IRS networks. Evaluation results show that FSO links reduce average communication hop count for user jobs, which is close to the best possible value of 2 hops and thus provides comparable benchmark performance to that of the counterpart RS architecture. In addition, if four FSO terminals per rack are allowed, the CPU/SSD (GPU) interconnection latency is reduced by 25.99% over Fat-tree and by 67.14% over 2-D Torus. We also present the advantage of an FSO-equipped IRS system in average turnaround time by 25.99% over Fat-tree and by 67.14% over 2-D Torus. We also present the advantage of an FSO-equipped IRS system in average turnaround time by 25.99% over Fat-tree and by 67.14% over 2-D Torus.

In this context, an evolved inter-rackscale (IRS) architecture [6] has been presented by further aggregating hardware resources with a more coarse-grained pooling strategy. The purpose of this study is to present an efficient job scheduling on IRS by the use of FSO links. For ease of understanding, in the following description we divide hardware resources into three areas, namely CPU (Central Processing Unit, with included memory), SSD (Solid-State Drive) and GPU (Graphics Processing Unit). An IRS system consists of CPU racks, SSD racks and GPU racks, and allows flexible allocation of these resources to user jobs.

In IRS, a main concern is communication latency among different resource racks. To reduce the impact of large inter-rack latency on running applications and make inter-rack communication latency comparable to that in RS, instead of cable communication we introduce a wireless technology called free-space optics (FSO) [7], [8] with high bandwidth over 10 Gbps for communication between racks in IRS. An important property of the wireless system is that link endpoints can be swapped on demand so as to lower communication latency and make job mapping and job scheduling more effective. Figure 1 shows an illustration of the RS architecture and our IRS architecture. In IRS, an FSO terminal can connect any other FSO terminals at different timings if they are both available.

This work is based on our previous research [6] which limits scalability in terms of network size and lacks job scheduling evaluation on IRS systems. The previous job scheduling evaluation on IRS systems. The previous job scheduling evaluation on IRS systems. The previous job scheduling evaluation on IRS systems.

1. Introduction

The rapid expansion of datacenters worldwide has been fueled by explosive growth in emerging fields, e.g., social media analysis with artificial intelligence (AI) processing. The same proliferation of rack servers has introduced new concerns because they are not optimally configured for different application purposes. This can result in resource waste and inefficiency; a different number of GPU/AI accelerators should work together when executing some specific applications. The computer systems require to be upgraded and customized as user applications evolve. Therefore, a novel redesign of datacenter rack infrastructure is direly needed to construct more efficient facilities in terms of scalability and optimization ability to each application.

Traditionally, in a machine room each rack consists of a number of homogeneous compute nodes which usually have the same amount of processors, memory, storages, and optionally accelerators. The rackscale (RS) computing architecture [1]–[4] proposed recently disaggregates compute, storage and network resources, and introduces ability to pool these resources for more efficient utilization. It simplifies resource management and dynamically composes hardware resources based on workload-specific demands. In RS, a different number of hardware resources runs as a differently configured server for a specific application appropriately. The pooled resources generate considerable amounts of provisioning and management data. Traditional IPMI interfaces and protocols, based on the IC bus, can be used for data communication, however their low data rate and inability to carry information render them incapable of meeting resource-pooling requirements [5]. For this reason, when a new user application appears for a specific interconnection network, the current techniques are still not mature to support flexible job mapping and job scheduling.

In this context, an evolved inter-rackscale (IRS) architecture [6] has been presented by further aggregating hardware resources with a more coarse-grained pooling strategy. The purpose of this study is to present an efficient job scheduling on IRS by the use of FSO links. For ease of understanding, in the following description we divide hardware resources into three areas, namely CPU (Central Processing Unit, with included memory), SSD (Solid-State Drive) and GPU (Graphics Processing Unit). An IRS system consists of CPU racks, SSD racks and GPU racks, and allows flexible allocation of these resources to user jobs.

In IRS, a main concern is communication latency among different resource racks. To reduce the impact of large inter-rack latency on running applications and make inter-rack communication latency comparable to that in RS, instead of cable communication we introduce a wireless technology called free-space optics (FSO) [7], [8] with high bandwidth over 10 Gbps for communication between racks in IRS. An important property of the wireless system is that link endpoints can be swapped on demand so as to lower communication latency and make job mapping and job scheduling more effective. Figure 1 shows an illustration of the RS architecture and our IRS architecture. In IRS, an FSO terminal can connect any other FSO terminals at different timings if they are both available.

This work is based on our previous research [6] which limits scalability in terms of network size and lacks job scheduling evaluation on IRS systems. The previous job...
scheduling algorithm can support up to 1,152 racks over 2-D Torus due to its high computation cost. To cover these issues, a new job scheduling algorithm with different mapping dilations is introduced, and detailed evaluation results in large-scale IRS systems are shown. In this study our job scheduling algorithm can support 3,072 racks over 3-D Torus by introducing the dilation in which multiple links are regarded as a single link on a job mapping.

Our main contributions in this work are as follows:

- We evaluated the impact of FSO links on IRS system performance. Evaluation results show that either hop count or communication latency can be largely reduced due to FSO link connections between IRS racks.
- We verified the advantage of the FSO-equipped IRS architecture in job scheduling performance such as average turnaround time of dispatched jobs for given sets of benchmark workloads.

The rest of this paper is organized as follows. Background information and related work are discussed in Sect. 2. Section 3 presents the IRS architecture and the floor-layout designs of interconnection networks for IRS racks. Section 4 shows simulation results for a large IRS system. Section 5 concludes with a summary of our findings in this paper.

2. Background and Related Work

2.1 Traditional Rack Server

A traditional rack server contains all resources required by various applications. However, the “one size fits all” server configuration does not meet the requirements of today’s datacenters for heavy computing applications since it suffers from low operation efficiency and low deployment density [9]. This leads to negative side-effects for resource-intensive applications. For instance, for compute-intensive applications, unused memory slots and HDD (hard disk drive) slots negatively affect computing density; for memory-intensive applications, unused HDD slots waste space that could be used for more memory; for storage-intensive applications, CPUs and memory might be exceedingly provided. Thus, it is difficult or perhaps impossible for a traditional server in a modern datacenter to support a wide variety of applications or workloads with a balanced configuration of compute, memory, storage and other resources.

In practice, most organizations customize rack servers with different hardware configurations to perform specific types of tasks according to different workload needs or purposes. This expansion of server types provides flexible and broad server diversity in a datacenter, however it introduces new challenges and complexities to server resource management, day-to-day maintenance and overall datacenter operations. For example, CPU performance improvement per year is different from that for storage and network. It is commonly observed for an all-in-one rack server that one area of its resources tends to become outdated more frequently than another area. This misalignment of technologies makes it difficult to upgrade to more efficient processors, memory and storages without unnecessarily discarding still useful resources. As a result, the majority of organizations would have old and less efficient servers, and seldom replace them as often as they like or as often as they should.

2.2 Custom Resource Sharing Technology

Virtualization [10] is a good technology in rack server optimization with a better user experience and improved resource utilization, especially for lightweight applications such as web hosting. Unlike a physical rack server environment that adding users means adding servers, a virtualized server environment allows multiple users to use the same resources, e.g., multicore processors, simultaneously. In this case, a single physical server can divide its physical resources into multiple virtual machines for different users or applications. This does reduce idle resources and improve utilization of hardware resources, but it does not entirely eliminate inefficiencies, as different workloads require specially configured servers that virtualization alone is not able to handle. Moreover, hardware resources in such a virtualized environment are still often underutilized due to the granularity of server resources. Another drawback of virtualization is the response time of each application due to the shared compute resources, and this may degrade user experiences.

The FireBox project [11] aims to develop a system architecture for custom resource sharing, which scales up to 10,000 compute nodes and up to an Exabyte of non-volatile memory connected via a low-latency, high-bandwidth optical switch. The FireBox project will produce distributed simulation tools for warehouse-scale machines, and systems softwares for FireBox-style disaggregated datacenters. However, it raises a number of novel questions in programming environments, operating systems, and hardware design.

2.3 Rackscale Architecture

The rackscale (RS) architecture [1–4] has been proposed recently, which addresses unavoidable mismatches in tech-
nology advancements by disaggregating hardware resources to multiple specific resource pools. A combination of traditional rack servers with virtualization cannot resolve these mismatches. The RS architecture aims to rearchitect a datacenter by gathering key server resource components into different resource pools, such as a compute pool, a memory pool, a storage pool and a network pool. These resources can be then flexibly assigned to meet demands of individual workloads. The pooled resources are reconstituted into an "instance" which can be seen as a traditional server. The underlying operating system and applications run the same way on these physical resources as if they were on the same physical server [12].

Under the RS architecture, different types of datacenter servers can be optimally provided according to different combinations of resource pools, such as a typical CPU sever configured with multiple high-performance CPUs and limited or no storage capabilities, and a typical storage server configured with several large HDDs and light processing resources. These differently configured servers can be treated appropriately [13]. However, the pooled resources generate considerable amounts of provisioning and management data. Traditional IPMI interfaces and protocols, based on the IC bus, can be used for data communication, but their data. Traditional IPMI interfaces and protocols, based on the IC bus, can be used for data communication, but their data.

Our IRS system introduces free-space optics (FSO) [7], [8] links with high bandwidth over 10 Gbps for inter-rack communications, so as to reduce the impact of large interrack latency on running applications and make inter-rack communication latency comparable to that in RS. With the use of FSO links, endpoints can be swapped on demand so that allocation of resources to user jobs can be more flexible.

2.4 Free Space Optics (FSO)

Free Space Optics (FSO) is utilized for high-bandwidth high-frequency wireless communication (e.g., several hundreds THz band for carrier signal) [7], [14]. FSO communication systems have been designed and deployed for home networks, indoor networks, inter-building links, links between two mobile vehicles, and long-distance data transfers such as high-bandwidth satellite communication.

Each wireless node can be equipped with a limited number of transmitters and receivers [15]. FSO communication has advantages of high bandwidth, low interference and fast set-up. More importantly, it has a similar BER (Bit Error Rate) to wired communication. Thus, the communication performance of an FSO link is reliable, and it has little impact on runtime of each job just as that of a wired link. In this work, the runtime of each job is assumed to be independent from whether a wired or a wireless link is used, and it is only determined by the timing when each job can start. In our previous works [7], [16], we have revealed the advantages of FSO links including the aspects of cable length, latency, cost, space, power, topology optimization for applications, and job mapping performance. We thus consider to deploy FSO terminals on top of racks under our IRS architecture, which has a fine-grained per-processor/storage/GPU job scheduling granularity.

FSO link reconfiguration mainly falls into two sequential steps: coarse pointing and fine tracking. The former can be performed with a commercial pan-tilt camera mechanical unit (e.g., PTU-D46, from FLIR), which can move terminal direction less than one second with the resolution of 0.013 degrees and the steering speed of 300 degrees/second [7]. The latter can be implemented by Arimoto’s device [17], which acquires target terminal direction in 0.092-degree field-of-view (FOV) within 10 milliseconds and maintains a stable link within 1.0-degree FOV. In this work, the default FSO reconfiguration latency is set to 1 second. We observe that in most cases the link reconfiguration overhead is negligible when compared to the time intervals between job submissions, thus it hardly affects the performance of job mapping and job scheduling.

Notice that, another option to reconfigure link endpoints is 60-GHz band radio communication, such as IEEE802.11ad [18], which has been considered in the context of datacenters. It was shown that offloading a portion of network traffic to radio network can improve the overall performance of datacenter applications [19]. However, the bandwidth of a 60-GHz radio link is up to several Gbps which is far below our target link bandwidth with 100 Gbps range [7]. In the following context, we interchangeably use the common term “wireless” to particularly refer to the FSO technology.

2.5 Inter-Rackscale Architecture

The inter-rackscale (IRS) architecture [6] can be deemed as an evolution of the RS architecture and it allows a datacenter to allocate hardware resources with greater efficiency by providing multiapplication-to-multirack (m-to-n) resource allocation. Compared to the generic virtualization of a single server into multiple nodes, the IRS architecture provides better system utilization and even approaches full resource utilization in a datacenter.

Under the IRS architecture, one rack consists of homogeneous hardware resources, such as CPU, SSD or GPU components. Multiple different racks cooperate as a whole computer system to run a specified user application. However, each rack in the system is not exclusively occupied by one application. In other words, resources in one rack can be flexibly allocated to multiple applications based on their respective hardware requirements. Therefore, the whole datacenter system under the IRS architecture can make full use of all hardware resources with a fine-grained granularity although it recomposes these resources in racks with a more coarse-grained granularity.

An essential part of IRS is to deploy FSO terminals on top of racks to directly connect two disjoint racks. Such wireless connections lower inter-rack communication latency, and enable full resource utilization of the system, regardless of host locations and interconnections.
that, the original RS architecture can be also extended with FSO communications, which however can not be fit for a per-processor/storage/GPU granularity during job allocation. As shown in Fig. 2, the more FSO terminals deployed for one RS rack, the less average turnaround time reached by 2-D Mesh and 3-D Mesh job mappings in a 3-D Torus (16×16×12) RS system (workload parameters are the same as those in Sect. 4.3.2). In this study, we focus on the use of FSO links for inter-rack communications in IRS systems.

3. IRS Considered in This Study

3.1 Behavior

Job mapping and job scheduling can be flexible in the IRS system due to its fine-grained granularity during resource allocation. When one job comes in, it is first mapped to CPU resources located at one or more racks that form a specified network topology such as Mesh or Torus. Then the CPU resources require to transfer necessary data with SSD and/or GPU resources based on the application’s property.

One of the biggest challenges in IRS is interconnectivity and latency because it offers a disaggregated server environment. If hardware resources in two racks communicate across many top-of-rack (ToR) switches or hops, it imposes a large communication delay on the system, and thus degrades application performance. To address the inter-rack communication problem, we use FSO terminals equipped on top of racks to enable their direct interconnections by wireless links. The FSO interconnections make inter-rack topologies more compact, and thus improve job mapping and application performance. Figure 3 shows job allocation in the original RS architecture and our IRS architecture. Usually, a switch delay (over 100 ns/hop) is far larger than a link delay (about 5 ns/m). The switch delay or path hop thus dominates the whole network delay for running applications. For simplicity, we disregard the link distance difference between an inter-rack communication and an intra-rack communication, since the performance penalty by distance is trivial. In our previous work [7], a simple prototype FSO terminal with the goal of achieving 360°-steerable communication for HPC use has been developed. It keeps a 1.6-mm positioning error at a 30-m distance, which is well within the 6-mm tolerance of an FSO link [14]. Furthermore, an experiment using Arimoto’s devices [17] has confirmed that the optical power loss over a 30-m indoor FSO link is well within SFP/XFP/QSFP standards. Therefore, optical transceivers of an FSO link emit a constant optical power regardless of the link loss, which does not affect the power consumption in the physical layer. Because an FSO link delay (3.3 ns/m) is even smaller than a wired link delay (5 ns/m), we consider that it is feasible to handle long, stable and efficient FSO communications as cables in datacenters. The evaluation results in Sect. 4.3.1 show that the use of FSO links can alleviate the impact of large inter-rack latency by providing a near 2-hop communication between IRS racks. We thus consider the performance can be comparable to different resource communication within one (RS) rack.

Note that one IRS rack’s hardware resources can be allocated to multiple applications on demand, however there is no guarantee that each application can utilize FSO links to facilitate its job mapping when the number of the on-top FSO terminals for one rack is limited. For one application, an FSO link can be used or reconfigured for its communication only if an FSO terminal is available or has been released by a previously occupied application.

3.2 Two Interconnection Networks

In this study, we mainly analyze two interconnection topologies for IRS racks: Fat-tree and Torus.

3.2.1 Fat-Tree

The Fat-tree structure [20] imposes large hops on inter-rack communication especially for the disaggregated resource environment under the IRS architecture. We decrease path hops between IRS racks by FSO links so as to reduce communication latency for a mapped job or application since it usually runs across multiple racks.

As shown in Figs. 4, 5 hops (yellow solid line) are required for cable communication between rack $r_1$ and rack...
3.2.2 Torus

Unlike in Fat-tree, the number of path hops between racks heavily depends on the geographical floor-layout of racks in Torus. Thus, varied relative positions of CPU, SSD and GPU racks lead to different diameters of inter-CPU networks and different average path hops for inter-CPU/SSD (GPU) communication.

In this study we analyze two prototypes, namely IRS-REPEAT and IRS-LOOP, according to geographical rack floor-layouts on y axis over 2-D torus (Fig. 5) or on z axis over 3-D torus (Fig. 6). In IRS-REPEAT CPU racks are in adjacent rows, thus we consider it is more suitable for CPU and memory intensive applications; in IRS-LOOP each CPU rack is adjacent to one SSD rack and one GPU rack, thus we consider it is more suitable for communication intensive applications. We use FSO links to enable inter-rack networks more compact and improve interaction performance. It is expected that the average inter-rack hop count and latency can be thus significantly reduced by FSO links.

Since a 2-D Torus can be obtained from a rectangle by identifying opposite sides, we first describe rack deployment that forms a rectangle, and then connect racks to make a 2-D Torus. For simplicity, we assume that the IRS system has an equal number of racks for CPU, SSD and GPU resources.

For comparison, the RS, IRS-REPEAT and IRS-LOOP architectures over 2-D Torus are described as follows. An interconnection network of 3-D Torus extends the concept of that of 2-D Torus, and all racks at different \((x, y)\) coordinates are homogeneous at the same \(z\) coordinate.

(1) RS

The RS racks are composed of all required resources (CPU, SSD and GPU components) and have the same capacity. Figure 5 (a) shows the RS racks over 2-D Torus.

(2) IRS-REPEAT

First, \(M\) racks of homogeneous resources (CPU, SSD or GPU resources) are deployed on consecutive \(N\) rows which form an \(M \times N\) rectangle. Then, the three rectangles of CPU, SSD and GPU racks are connected with each other so that they form a 2-D Torus network. Figure 5 (b) illustrates the IRS-REPEAT architecture over 2-D Torus, where homogeneous racks repeatedly appear on consecutive rows.

One advantage of IRS-REPEAT is that inter-CPU networks have high density since CPU racks are on consecutive rows, however the number of inter-CPU/SSD (GPU) path hops is large. We assume the inter-CPU/SSD (GPU) hop count to be the number of hops between one CPU rack and one SSD (GPU) rack located at the same \((x, y)\) coordinate within their own \(M \times N\) rectangles. As shown in Fig. 5 (b), the inter-CPU/SSD (GPU) hop count is equal to \(N + 1\).

For IRS-REPEAT, the main concern of running jobs or applications is not the density of inter-CPU networks but the inter-CPU/SSD (GPU) hop count. We thus use FSO links to connect disjoint different racks so as to reduce inter-CPU/SSD (GPU) hop counts to a minimum value 2 since any two different racks can be directly connected by an FSO link.
(3) IRS-LOOP

First, $M$ racks of homogeneous resources (CPU, SSD or GPU resources) are deployed on a row so that such $N$ rows each are ready for interconnections. Then, the rows of CPU, SSD and GPU racks are staggered to be connected with each other so that they form a 2-D Torus network. Figure 5(c) illustrates the IRS-LOOP architecture over 2-D Torus, where different resource racks loop in row.

One advantage of IRS-LOOP is that the number of inter-CPU/SSD (GPU) path hops is small since one CPU rack row is adjacent to one SSD (GPU) rack row, however inter-CPU networks are not compact. We assume the inter-CPU/SSD (GPU) hop count to be the number of hops between one CPU rack and its directly connected SSD (GPU) rack on y axis. As shown in Fig. 5(c), the inter-CPU/SSD (GPU) hop count can reach a minimum value of 2.

For IRS-LOOP, contrarily, the main concern of running jobs or applications is not the inter-CPU/SSD (GPU) hop count but the density of inter-CPU networks. We thus use FSO links to directly connect disjoint CPU racks so as to make inter-CPU networks more compact.

4. Simulation Evaluation

4.1 Methodology

To model job mapping and job scheduling on datacenters, we developed an event-driven HPC simulator in Python 2.7 and ran the simulation in a machine with Intel i7 CPU and 4GiB Memory. In this study, we use a common approach to model parallel “rigid” jobs [21], which refer to jobs that use a fixed number of resources during runtime. In other words, we allocate to one job or application a certain number of CPU, SSD and GPU resources for a certain time. We consider inter-CPU/SSD (GPU) communication latency for job execution time, whose values derive all originally from the data in Hara’s work [22].

As a job comes in, its requirements are evaluated and placed into the job queue, and then prioritized according to a queuing policy, e.g., first-come-first-serve (FCFS). For a job mapping and execution, it is first assigned to a set of unused CPU resources on a specified inter-CPU subtopology, and then requires to transfer data with SSD and/or GPU resources. The system utilization becomes low in many cases where an incoming job can not be mapped even though a large number of compute racks are available since they are disjoint. This situation can be improved by using FSO links to connect disjoint or distant compute racks. An efficient wireless job mapping algorithm on a 2-D Torus network topology has been described in our previous work [16]. For IRS, increased inter-rack path hops lead to greater communication latency between CPU and SSD (GPU) components since they are in different separate racks. In this case, FSO links can be used to reduce inter-CPU/SSD (GPU) path hops and to lower their communication latency accordingly. Thus, job execution time and turnaround time can be decreased.

In this work, we also investigate tradeoff between scheduling performance and a dilation-$n$ ($n > 1$) mapping. Here, the dilation in the topology embedding refers to the length (in number of hops) of the shortest path between two embedded vertices of an edge. Assume a mapping $F$ from the vertices of a graph $G$ to the vertices of a (larger) graph $H$. Given an edge in $G$ between two vertices $v$ and $w$, its dilation is the length of the shortest path between $F(v)$ and $F(w)$ in number of hops in $H$. The dilation for a mapping is the maximum value of all the edge dilations. For example, a mapping with dilation = 1 does not expand any edge of $G$ in $H$, and a mapping with dilation = 2 leads to the maximum 2 hops for any edge of $G$ in $H$. Thus, when the dilation increases, its job mapping becomes easier while imposing larger communication latency. A dilation-1 topology embedding is first considered for job mapping with or without the use of FSO links. Because the number of FSO terminals for one rack is limited, when they are all occupied we consider a dilation-$n$ topology embedding for job execution. If a dilation-2 topology embedding is still not found, we consider a dilation-$n$ ($n > 2$) topology embedding for job execution. Note that, a dilation-$n$ ($n \leq 2$) topology embedding is a contiguous mapping, while a dilation-$n$ ($n > 2$) topology embedding is a non-contiguous mapping [23] because corresponding racks are not directly connected. Algorithm 1 provides the whole process to make a job mapping and job scheduling according to a specified maximum dilation.

Algorithm 1: Job mapping and job scheduling.

| Line | Description |
|------|-------------|
| 1:   | procedure schedule(system, job_queue, max_dilation) |
| 2:   | while job_queue ≠ NULL do |
| 3:   | next_job ← getJobByFCFS(job_queue) |
| 4:   | dilation ← 1 |
| 5:   | if map(system, dilation) == True then |
| 6:   | dispatch(next_job) |
| 7:   | else |
| 8:   | while dilation ≤ max_dilation - 1 do |
| 9:   | dilation ← dilation + 1 |
| 10:  | if map(system, dilation) == True then |
| 11:  | dispatch(next_job) |
| 12:  | break |
| 13:  | end if |
| 14:  | end while |
| 15:  | end if |
| 16:  | end while |
| 17:  | end procedure |

4.2 Simulation Setup

The arrangement of racks in our evaluation is shown in Table 1.

4.2.1 Fat-Tree

We investigate an evolved instance of a $k$-ary Fat-tree topology [20]. In our case, it consists of 48 pods built from 48-port switches, each containing an edge layer and an aggre-
The whole network topology contains 384 CPU racks with 12 CPUs each, 384 SSD racks with 192 SSDs each, and 384 GPU racks with 192 GPUs each. The CPU racks, SSD racks and GPU racks belong to 16 pods respectively. Since the whole network is a three-tiered Fat-tree topology, it requires 5 hops to communicate between different resource racks. Due to large multihop communication latency, we use FSO links to reduce inter-rack hop counts.

### 4.2.2 Torus

The whole network topology supports 1,152 racks over 2-D Torus and 3,072 racks over 3-D Torus. For RS, all the racks are identical, containing 4 CPUs, 64 SSDs and 64 GPUs each. For IRS, the whole network topology contains 384 CPU racks with 12 CPUs each, 384 SSD racks with 192 SSDs each, and 384 GPU racks with 192 GPUs each. The CPU racks, SSD racks and GPU racks belong to 16 pods respectively. Since the whole network is a three-tiered Fat-tree topology, it requires 5 hops to communicate between different resource racks. Due to large multihop communication latency, we use FSO links to reduce inter-rack hop counts.

| Architecture           | # of Racks (Switches) | Arrangement of Racks | # of FSOs per Rack |
|------------------------|-----------------------|----------------------|-------------------|
| RS (Fat-tree)          | 1,152                 | z = 48               |                   |
| IRS (Fat-tree)         | 1,152                 | z = 48               | 0(1/2)/4          |
| RS (2-D Torus)         | 1,152                 | 24 × 48              |                   |
| IRS-REPEAT (2-D Torus) | 1,152                 | 24 × 48 (repeat on y axis) | 0(1/2)/4          |
| IRS-LOOP (2-D Torus)   | 1,152                 | 24 × 48 (loop on y axis) | 0(1/2)/4          |
| RS (3-D Torus)         | 3,072                 | 16 × 16 × 12         |                   |
| IRS-REPEAT (3-D Torus) | 3,072                 | 16 × 16 × 12 (repeat on z axis) | 0(1/2)/4          |
| IRS-LOOP (3-D Torus)   | 3,072                 | 16 × 16 × 12 (loop on z axis) | 0(1/2)/4          |

### 4.3 Simulation Results

#### 4.3.1 Impact of FSO on IRS Systems

We evaluate the impact of FSO links on IRS systems when assuming the host interconnection network to be Fat-tree and 2-D Torus respectively.

We generate \( n \in [1000, 10000] \) jobs as workloads with random arrival timings for a Poisson process with \( \lambda = \frac{n}{1000} \).

Each job specifies the required number of hardware resources, i.e., CPUs, SSDs and GPUs. We regard one CPU unit as 4 CPUs, one SSD unit as 64 SSDs, and one GPU unit as 64 GPUs. For one job, the resource requirements are randomly generated but they vary up to 25 units. Thus, the number of required CPUs, SSDs and GPUs can vary up to 100, 1600 and 1600, respectively. The CPU processing time of one job is designated as \( t \in [1, 10] \) seconds. Note that, for IRS the actual job execution time is usually larger than the specified CPU processing time, because we take into account communication latency overhead of inter-CPU/SSD (GPU) data transfers. In most cases, increased path hops impose greater communication latency between CPU racks and SSD (GPU) racks on job execution.

1. **Hop Count**

Because different resource racks are deployed in different pods over Fat-tree, the inter-CPU/SSD (GPU) hop counts are all equal to 5 if we simply utilize cable or wired communication between racks. However, different resource racks can be directly connected if we deploy FSO terminals on top of racks. Figure 7 illustrates the average inter-CPU/SSD (GPU) hop count of all dispatched jobs over Fat-tree when we deploy 0, 1, 2, 3 and 4 FSO terminals for one IRS rack, respectively. It is observed that the inter-CPU/SSD (GPU) hop counts can be significantly reduced by the use of FSO terminals for different workload sizes. Especially, if we deploy 4 FSO terminals per rack, the average inter-CPU/SSD (GPU) hop count can be decreased to 2.1, which approaches the minimum necessary path hops to directly connect two different resource racks.

A similar tendency can be seen from the case of 2-D Torus for IRS-REPEAT racks. As shown in Fig. 8, the inter-CPU/SSD (GPU) hop counts for all dispatched jobs are equal to 17 for cable or wired communication between racks. The greater utilization of FSO terminals helps to further reduce inter-CPU/SSD (GPU) hop counts for different workload sizes, and especially the average hop count can be decreased to 2.2 when we deploy 4 FSO terminals per rack. This largely alleviates the side effect of large communication hop count between different resource racks, es-

---

**Table 1** Arrangement of racks in RS/IRS systems.

| Architecture           | # of Racks (Switches) | Arrangement of Racks | # of FSOs per Rack |
|------------------------|-----------------------|----------------------|-------------------|
| RS (Fat-tree)          | 1,152                 | z = 48               |                   |
| IRS (Fat-tree)         | 1,152                 | z = 48               | 0(1/2)/4          |
| RS (2-D Torus)         | 1,152                 | 24 × 48              |                   |
| IRS-REPEAT (2-D Torus) | 1,152                 | 24 × 48 (repeat on y axis) | 0(1/2)/4          |
| IRS-LOOP (2-D Torus)   | 1,152                 | 24 × 48 (loop on y axis) | 0(1/2)/4          |
| RS (3-D Torus)         | 3,072                 | 16 × 16 × 12         |                   |
| IRS-REPEAT (3-D Torus) | 3,072                 | 16 × 16 × 12 (repeat on z axis) | 0(1/2)/4          |
| IRS-LOOP (3-D Torus)   | 3,072                 | 16 × 16 × 12 (loop on z axis) | 0(1/2)/4          |
especially for CPU and memory intensive applications where inter-CPU/SSD (GPU) communication time tends to be the performance bottleneck. For IRS-LOOP, since each CPU rack is adjacent to one SSD rack and one GPU rack, the inter-CPU/SSD (GPU) communication can reach the minimum 2 hops.

Due to the 2-hop inter-rack connection by FSO links, the communication latency between IRS racks can be comparable to the inter-CPU/SSD (GPU) communication latency within one RS rack. We thus demonstrate that the inter-CPU/SSD (GPU) communication ability of the FSO-equipped IRS architecture can be comparable to that of the counterpart RS architecture. This provides great latency optimization for user applications when they run on IRS systems.

(2) Communication Latency

We calculate relative execution time for data transfers between CPUs and SSDs (GPUs) based on Hara’s work [22], in which the execution time incrementally becomes large as the inter-CPU/SSD (GPU) hop count increases. According to these values, we evaluate CPU-SSD (GPU) interaction latency between IRS racks.

As shown in Fig. 9, the FSO utilization helps to reduce the CPU-SSD (GPU) interaction latency over Fat-tree. The theoretical minimum value is derived from the case that all the CPU-SSD (GPU) connections can be established by FSO links. In other words, the theoretical minimum value is calculated when the number of FSO terminals per rack is not limited for CPU-SSD (GPU) connections if required. It can be seen from the result that the FSO utilization improves the interaction performance by up to 25.99% when 4 FSO terminals per rack are set, which approaches the theoretical maximum value of 26.62%. Different workload sizes make a tiny difference for the improvement ratio regarding the CPU-SSD (GPU) interaction latency.

Figure 10 presents the case of 2-D Torus for IRS racks. Similarly, the greater FSO utilization brings a larger improvement ratio for the CPU-SSD (GPU) interaction latency. When 4 FSO terminals per rack are deployed, it reduces the CPU-SSD (GPU) interaction latency by up to 67.14%, which approaches 67.88% for IRS-LOOP where CPU racks and SSD (GPU) racks are adjacent. This largely alleviates the side effect of large communication time between different resource racks, especially for CPU and memory intensive applications where inter-CPU/SSD (GPU) communication time tends to be the performance bottleneck. Like over Fat-tree, different workload sizes make little difference for the improvement ratio over 2-D Torus.

4.3.2 Job Mapping and Job Scheduling

We evaluate the performance of a large-scale IRS system with a workload composed of various NPB applications [24] including FT (Fast Fourier), IS (Integer Sort), CG (Conjugate Gradient), BT (Block Tri), SP (Scalar Penta) and MG (Multi-Grid) take from NAS parallel benchmarks [25] and MM (Matrix Multiplication). The host topology is assumed to be a 3-D Torus ($16 \times 16 \times 12$) network. To acquire execution times (simulation cycles) for the NPB applications running on diverse guest topologies with different job mapping dilations, we first get a series of simulation results by the event driven simulator SimGrid [26]. The execution times are used as time options for an application with different mapping dilations. When a job is dispatched to the system, the exact execution time is determined by its mapping di-
**Table 2** Workload parameters in the evaluation.

| Benchmarks | # of Processors | Host Topology | Guest Topology | Dilation |
|------------|----------------|---------------|----------------|----------|
| FT         | [4, 16, 64, 256]| 3-D Torus     | 2D-Mesh        | 1/2/3/4  |
| BT         | (16 x 16 x 12)  | 4 x 2 x 2, 4, 4, 8 x 8 x 16 | 4 x 2 x 2, 4, 4, 8 x 8 x 4 | 1/2/3/4  |
| SP         | (4 x 2 x 2, 4, 4, 8 x 8 x 4) | 3D-Mesh       | 2D-Mesh        | 1/2/3/4  |
| MG         | Random          | 3D-Torus      | 3D-Mesh        | 1/2/3/4  |
| MM         | (degree = 6)    | Random        | Random         | 1/2/3/4  |

**Fig. 11** The average diameter of inter-CPU networks over the 3-D Torus interconnection network.

**Fig. 12** The average turnaround time of dispatched jobs over the 3-D Torus interconnection network.

...The workload parameters used in the evaluation are shown in Table 2. We generate \( n = 2000 \) jobs with random arrival timings for a Poisson process with \( \lambda = \frac{n}{T} \). Each job is prioritized to be mapped to the system with dilation-1. The dilation-2/3/4 mapping is also considered if current dilation-1 mapping cannot be found on the system. We also take into account communication latency overhead of inter-CPU/SSD (GPU) data transfers. In the evaluation, FSO links are used to connect non-adjacent CPU racks for IRS-LOOP where applications can always run with dilation-1 mappings. Because CPU racks are in adjacent rows for IRS-REPEAT, FSO links are not used for their connections when we make dilation-2/3/4 mappings for applications.

(1) Network Fragmentation

Figure 11 shows the average diameter of inter-CPU networks with different guest topologies. It can be seen that a job mapping with a larger dilation leads to a slightly larger average inter-CPU network diameter. Diverse architectures bring little difference over 2-D Mesh guest topologies because each embedding subtopology can be composed of adjacent 2-D \((x, y)\) processors at the same \(z\) coordinate. The IRS-LOOP racks without FSO links have a significantly larger average inter-CPU network diameter over 3-D Mesh/Torus guest topologies because each job (except a 4-processor job) cannot be mapped on consecutive processors on \(z\) axis. The difference becomes even larger when jobs are mapped on random degree-6 guest topologies. However, this gap can be incrementally filled by using more FSO links for each rack. If two of FSO links are used for inter-CPU network connections, the average embedded subtopology diameter becomes comparable to that in other architectures. This largely alleviates the side effect of large communication hop count between CPU racks, especially for communication intensive applications where inter-CPU communication time tends to be the performance bottleneck.

(2) Turnaround Time

We set larger execution times by combining multiple applications as a job. Specifically, a job consists of randomly selected 10 benchmark applications (FT, IS, CG, BT, SP, MG and MM). Notice that the order of execution time of the benchmark applications in SimGrid is five milliseconds to five seconds.

Figure 12 presents the average turnaround time of dispatched jobs. The turnaround time is divided into the queuing time (scheduling latency) and the execution time. The former refers to the time spent on the run queue and the latter refers to the time spent by the system executing that task. Overall, IRS-REPEAT has a slight advantage over RS because of its fine-grained granularity in resource allocation and high system utilization accordingly. For example, IRS-REPEAT provides 5.2% shorter turnaround time (mapping dilation \(\leq 4\) over 3-D Torus) when compared to the original RS architecture. Interestingly, a job mapping with a larger possible dilation can further decrease the average turnaround time, because it can shorten the waiting time for each queuing job while not increasing its execution time heavily. In other words, a job mapping with a larger possible dilation has a smaller slowdown for one job when compared to the strict dilation-1 job mapping. IRS-LOOP brings a significant advantage over IRS-REPEAT, because it has lower latency for inter-CPU/SSD (GPU) communication. FSO links used for inter-CPU network connections can further decrease the average turnaround time, and thus improve the job mapping and job scheduling performance. Compared to RS, IRS-LOOP brings even 17.6% shorter turnaround time (2 FSOs/rack over 3-D Torus).
5. Conclusion

The IRS architecture disaggregates various hardware resources into different racks according to their own areas. We used free-space optical (FSO) links between IRS racks to lower inter-rack communication latency, and thus made its job mapping and job scheduling more flexible. Compared to the original RS architecture, although IRS recompenses hardware resources in racks with a more coarse-grained granularity, it can make full use of these resources with a fine-grained granularity during job allocation with the use of FSO links.

In this study we illustrated the efficiency of the IRS architecture assuming that the inter-rack baseline network topology is Fat-tree and Torus respectively. For Torus, we analyzed two interconnection prototypes according to geographical rack floor-layouts for different application types (CPU and memory intensive applications and communication intensive applications), namely IRS-REPEAT and IRS-LOOP. We showed that either hop count or communication latency can be largely reduced due to FSO link connections between IRS racks. If four FSO terminals per rack are allowed, the CPU/SSD (GPU) interconnection latency can be reduced by 25.99% over Fat-tree and by 67.14% over 2-D Torus. We also verified the advantage of an FSO-equipped IRS system in average turnaround time of dispatched jobs for given sets of benchmark workloads. Compared to the original RS architecture, IRS-REPEAT provides 5.2% shorter turnaround time (mapping dilation ≤ 4 over 3-D Torus). IRS-LOOP brings even 17.6% shorter turnaround time (2 FSOs/rack over 3-D Torus) because it has lower latency for inter-CPU/SSD (GPU) communication.

Acknowledgments

This work was supported by JSPS KAKENHI Grant Number 17K12682.

References

[1] Intel® rack scale architecture: Faster service delivery and lower TCO. [Online]. Available: http://www.intel.com/content/www/us/en/architecture-and-technology/intel-rack-scale-architecture.html
[2] Dagsstuhl seminar 15421, rack-scale computing. [Online]. Available: http://www.dagstuhl.de/15421
[3] S. Legtchenko, N. Chen, D. Cletheroe, A. Rowstron, H. Williams, and X. Zhao, “Xiabric: A reconfigurable in-rack network for rack-scale computers,” NSDI ’16 Proceedings of the 13th Usenix Conference on Networked Systems Design and Implementation, pp.15–29, 2016.
[4] D. Zhang, Z. Cao, X. Liu, L. Wang, Z. Wang, and N. Sun, “PROP: Using PCIe-based RDMA to accelerate rack-scale communications in data centers,” IEEE 21st International Conference on Parallel and Distributed Systems (ICPADS), pp.465–472, Dec. 2015.
[5] Cisco visual networking index: Global mobile data traffic forecast update 2014–2019 whitepaper. [Online]. Available: http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white_paper_c11-520862.html, Feb. 2015.
[6] Y. Hu, H. Hará, I. Fujiwara, H. Matsutani, H. Amano, and M. Koibuchi, “Towards tightly-coupled datacenter with free-space optical links,” 2017 International Conference on Cloud and Big Data Computing (ICCBDC 2017), pp.33–39, Sept. 2017.
[7] I. Fujiwara, M. Koibuchi, T. Ozaki, H. Matsutani, and H. Casanova, “Augmenting low-latency HPC network with free-space optical links,” 21st International Conference on High-Performance Computer Architecture (HPCA), pp.390–401, Feb. 2015.
[8] Y. Katayama, T. Yamane, Y. Kohda, K. Takano, D. Nakanou, and N. Ohba, “MIMO link design strategy for wireless data center applications,” Wireless Communications and Networking Conference (WCNC), pp.3302–3306, 2012.
[9] [Online]. Available: http://www.tuangru.com/what-is-a-rack-server/
[10] E. Turban, D. King, J. Lee, and D. Viehland, “Electronic commerce a managerial perspective,” Building E-Commerce Applications and Infrastructure, p.27, 2008.
[11] [Online]. Available: http://bar.eecs.berkeley.edu/projects/2015-firebox.html
[12] T. Szepesi, B. Wong, T. Brecht, and S. Rizvi, “Towards reconfigurable rack-scale networking,” EuroSys, WRSC: Second International Workshop on Rack-scale Computing, April 2015.
[13] H. Wang, “Tencent explores datacenter resource-pooling using intel® rack scale architecture,” Tencent Holdings Limited, Tech. Rep., 2015.
[14] N. Hamedzimí, Z. Qazi, H. Gupta, V. Sekar, S.R. Das, J.P. Longtin, H. Shah, and A. Tanwer, “FireFly: A reconfigurable wireless data center fabric using free-space optics,” Proc. 2014 ACM Conference on SIGCOMM, pp.319–330, 2014.
[15] A. Kashyap, K. Lee, M. Kalantari, S. Khuller, and M. Sayyam, “Integrated topology control and routing in wireless optical mesh networks,” Computer Networks, vol.51, no.15, pp.4237–4251, Oct. 2007.
[16] Y. Hu, I. Fujiwara, and M. Koibuchi, “HPC job mapping over reconfigurable wireless links,” 16th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid 2016), pp.570–575, May 2016.
[17] Y. Arimoto, H. Yoshida, and K. Kısara, “Wide field-of-view single-mode-fiber coupled laser communication terminal,” Proc. SPIE, vol.8610, 861008, 2013.
[18] “Enhancements for Very High Throughput in the 60 GHz Band,” IEEE Standard for Information technology—Telecommunications and information exchange between systems—Local and metropolitan area networks—Specific requirements-Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, 2012. [Online]. Available: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6392842
[19] D. Halperin, S. Kandula, J. Padiye, P. Bahl, and D. Wetherall, “Augmenting data center networks with multi-gigabit wireless links,” Proc. ACM SIGCOMM, pp.38–49, 2011.
[20] M. Al-Fares, A. Loukissas, and A. Vahdat, “A scalable, commodity data center network architecture,” Proc. ACM SIGCOMM 2008 Conference on Data Communication, pp.63–74, 2008.
[21] D.G. Feitelson, Workload Modeling for Computer Systems Performance Evaluation, Cambridge University Press, March 2015.
[22] H. Hará, T. Ozaki, M. Koibuchi, H. Amano, and H. Matsutani, “Latency reduction on inter-component communication across racks using FSO,” IEICE Technical Report, vol.115, no.400, pp.161–166, Jan. 2016.
[23] O. Tuncer, V.J. Leung, and A.K. Coskun, “PaCMap: Topology mapping of unstructured communication patterns onto non-contiguous allocations,” Proc. 29th ACM on International Conference on Supercomputing, pp.37–46, 2015.
[24] The NAS Parallel Benchmarks, http://www.nas.nasa.gov/Software/NPB/
[25] D. Bailey, E. Barszcz, J. Barton, D. Browning, R. Carter, L. Dagum, R. Fatoohi, S. Fineberg, P. Frederickson, T. Lasinski, R. Schreiber, H. Simon, V. Venkataraman, and S. Weeratunga, “The NAS parallel benchmarks,” https://www.nas.nasa.gov/assets/pdf/techreports/
Yao Hu received the M.S. degree from Beijing University of Posts and Telecommunications, China, in 2009, and received the Ph.D. degree from the Department of Computer Science and Engineering, Waseda University, Tokyo, Japan, in 2015. He is currently working as a project researcher in the National Institute of Informatics, Tokyo, Japan. His main research interests include the area of high-performance computing.

Michihiro Koibuchi received the B.E., M.E., and Ph.D. degrees from Keio University, Yokohama, Japan, in 2000, 2002 and 2003, respectively. Currently, he is an associate professor in the Information Systems Architecture Research Division, National Institute of Informatics and the Graduate University of Advanced Studies, Tokyo, Japan. His research interests include the areas of high-performance computing and interconnection networks. He is a member of the IEEE and a senior member of the IEICE.