Achieving Ultralow-Latency Optical Interconnection for High Performance Computing (HPC) Systems by Joint Allocation of Computation and Communication Resources

Ruijie Luo[1,2], Yufang Yu[1,2], Nan Hua[1,2], Zhizhen Zhong[1,2], Jialong Li[1,2], Xiaoping Zheng[1,2], Bingkun Zhou[1,2]

1. *Beijing National Research Center for Information Science and Technology（BNRist）, Beijing, 100084, P. R. China*
2. *Department of Electronic Engineering, Tsinghua University, Beijing, 100084, P. R. China*

{huan, xpzheng}@mail.tsinghua.edu.cn

Abstract: We propose joint allocation of computation resource and optical transmission time slices to realize ultralow-latency optical interconnection in time-synchronized HPC systems. Results show that over 80% reduction in buffering time is achieved at high load. © 2019 The Author(s)

OCIS codes: (060.4250) Networks; (060.4256) Networks, network optimization;

1. Introduction

To fulfill the rapid increasing requirements of scientific computing such as climate modeling, earth subsurface modeling and phase-field simulation, high performance computing (HPC) systems are being built at the scale of hundreds of thousands of computing/storage nodes [1, 2]. It is a big challenge to build low-latency high-bandwidth interconnection network between such huge amounts of HPC nodes. Current designs for HPC interconnection networks are based on electronic packet switching (EPS), which suffers from the weakness in capacity, end-to-end latency and energy consumption due to the electronic processing and buffering at intermediate switching nodes. Thus, the EPS interconnection network becomes a bottleneck for improving the performances of HPC systems [2, 3].

Introducing optical circuit switching (OCS) technologies into HPC systems provides the potential benefits of breaking through the bottlenecks of EPS by enabling low-latency optical bypass. In literatures [2] and [3], researchers have proposed OCS-involved interconnection architectures in HPC systems to enhance network capacity and reduce end-to-end latency. However, these architectures are not able to provide fine-grained connections, which will lead to the shortage of optical channels and inefficient network utilization. Recently, researchers in works [4, 5] have proposed a fine-grained optical time slice switching (OTSS) method, which can provide abundant fine-grained low-latency optical channels based on network time synchronization [6] for HPC systems. However, the mismatch between traffic patterns and communication resources leads to extra buffering time and longer run-time for all jobs.

In this paper, we propose an OTSS-based joint allocation of computation and communication resources (JACC) method for ultralow-latency optical interconnection in HPC systems. To reduce the total run-time for HPC, we propose an integer programming (IP) model to minimize the mismatch between the job schedule and time slices transmission. The simulation and experiment results show that over 80% buffering time can be reduced at high load and 50-ns-level end-to-end latency is achieved.

2. JACC method for HPC system

Figure 1 presents the OTSS-based HPC architecture. HPC jobs are broken down into small tasks and each of the tasks are executed on a single computing group. There are dependency relationships [7] between tasks which require the tasks to run in sequence, and the backward tasks should wait until receiving the results of the forward tasks. The transmission tasks aim to transmit the result of the forward computation tasks. The job schedule on computing groups will form a traffic matrix which consists of transmission tasks for the HPC interconnection network. According to the OTSS method, periodic time slices will be allocated for the transmission tasks.

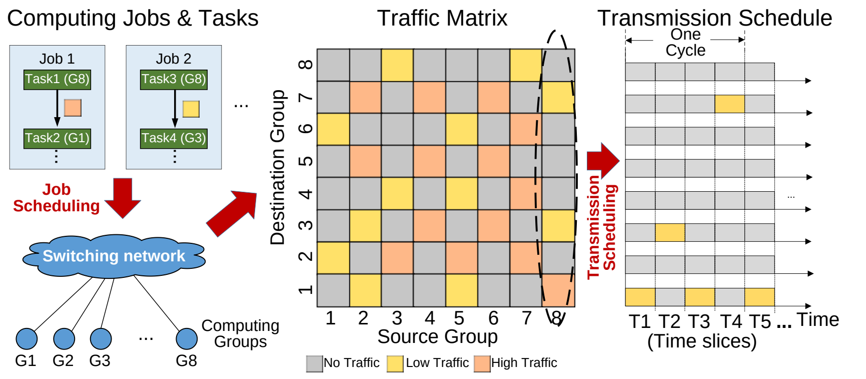

Fig. 1. OTSS-based HPC architecture

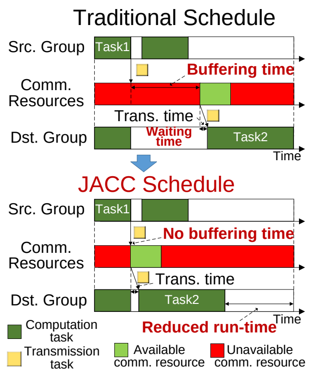

Fig. 2 JACC method

This work was supported in part by projects under NSFC grants No. 61871448 and 61621064.

In traditional scheduling method, the mismatch between job schedule and transmission schedule may lead to extra buffering time for transmission tasks as shown in Fig. 2. To reduce the end-to-end latency, the JACC method jointly allocates the computation and transmission tasks to minimize the buffering time, and further reduces the total run-time of the HPC jobs.

3. Integer programming model for JACC

To minimize the total run-time of the HPC jobs, we propose an IP model for JACC. Based on the fact that the transmission delays between computing groups (100-ns-level) are far less than the length of time slices (100-us-level), we ignore the transmission delay in our model for simplicity. Moreover, the computing times of the tasks are also organized into time slices.

- **Given:**

G(N, E) : the HPC switching network topology, where *N* and *E* denote the set of nodes and links, respectively.
t_m : denotes the m^th computation task.
T : set of computation tasks, $t_m \in T$.
e_{ij} : denotes the edge from node *i* to node $j\ (i, j \in N)$.
$D(t_m)$: set of time slices required for executing task t_m.
cg: denotes the serial number of a computing group.
CG : set of computing groups, $cg \in CG$.
$\varepsilon(x)$: the step function which takes the value of one if *x* is more than zero, otherwise takes the value of zero.
R : set of dependency relationships of the tasks.
TS(R) : set of time slices required in the transmission tasks between the related tasks in *R*.
(t_m, t_n) : denotes the transmission task between task t_m and task t_n, where t_m is the forward task and t_n is the backward task ($(t_m, t_n) \in R,\ t_m \in T,\ t_n \in T$).
TN : set of time slices in one cycle.
ts : denotes the serial number of a time slice in one cycle.
k : denotes the time slices required in transmission task (t_m, t_n) ($k = TS(t_m, t_n)$).

- **Variables:**

$\mathrm{C}_{tm,cg,ts}$: binary decision variable, which takes the value of one if computing group *cg* runs task t_m in time slice *ts*.
$\mathrm{L}_{eij}^{(tm,tn),k,ts}$: binary decision variable, which takes the value of one if the transmission task (t_m, t_n) occupies time slice *ts* on link e_{ij}.
Z : an integer variable, which represents the serial number of the time slice when the last task is finished.

- **Optimize:**

$$\text{Minimize } Z = \max_{\forall tm\in T, \forall cg\in CG} \Big(\sum_{\forall ts\in TN} (\varepsilon(\mathrm{C}_{tm,cg,ts} - \mathrm{C}_{tm,cg,ts+1}) \times ts)\Big) \tag{1}$$

- **Constraints:**

$$\sum_{\forall tm\in T} \mathrm{C}_{tm,cg,ts} \le 1, \forall ts \in TN, cg \in CG \tag{2}$$

$$\sum_{\forall cg\in CG} \mathrm{C}_{tm,cg,ts} \le 1, \forall ts \in TN, t_m \in T \tag{3}$$

$$\sum_{\forall cg\in CG} \sum_{\forall ts\in TN} \mathrm{C}_{tm,cg,ts} = D(t_m), \forall t_m \in T \tag{4}$$

$$\sum_{\forall cg\in CG} \sum_{\forall ts\in TN} (\left|\mathrm{C}_{tm,cg,ts} - \mathrm{C}_{tm,cg,ts+1}\right|) \le 2, \forall t_m \in T \tag{5}$$

$$\sum_{\forall (tm,tn)\in R} \mathrm{L}_{eij}^{(tm,tn),k,ts} \le 1, \forall e_{ij} \in E, ts \in TN \tag{6}$$

$$\sum_{\forall ts\in TN} (\left|\mathrm{L}_{eij}^{(tm,tn),k,ts} - \mathrm{L}_{eij}^{(tm,tn),k,ts+1}\right|) \le 2, \forall e_{ij} \in E, (t_m, t_n) \in R \tag{7}$$

$$\sum_{\forall j\in N} \mathrm{L}_{e_{ij}}^{(t_m,t_n),k,ts} - \sum_{\forall j\in N} \mathrm{L}_{e_{ji}}^{(t_m,t_n),k,ts} = \begin{cases} 1, & \mathrm{C}_{t_m,i=cg,ts} > 0 \\ -1, & \mathrm{C}_{t_n,i=cg,ts} > 0 \\ 0, & else \end{cases} \tag{8}$$
$$\forall (t_m, t_n) \in T, cg \in CG, ts \in TN$$

$$\sum_{\forall ts\in TN} \Big(\sum_{\forall j\in N} \mathrm{L}_{e_{ij}}^{(t_m,t_n),k,ts} - \sum_{\forall j\in N} \mathrm{L}_{e_{ji}}^{(t_m,t_n),k,ts}\Big) = \begin{cases} k, & \sum_{\forall ts\in TN} \mathrm{C}_{t_m,i=cg,ts} > 0 \\ -k, & \sum_{\forall ts\in TN} \mathrm{C}_{t_n,i=cg,ts} > 0 \\ 0, & else \end{cases} \tag{9}$$
$$\forall (t_m, t_n) \in R$$

$$\sum_{\forall cg\in CG} \sum_{\forall ts\in TN} (\varepsilon(\mathrm{C}_{tm,cg,ts} - \mathrm{C}_{tm,cg,ts+1}) \times ts) + \mathrm{k} \le \sum_{\forall cg\in CG} \sum_{\forall ts\in TN} (\varepsilon(\mathrm{C}_{t_n,cg,ts+1} - \mathrm{C}_{t_n,cg,ts}) \times ts), \forall (t_m, t_n) \in R \tag{10}$$

$$\sum_{\forall ts\in TN} (\varepsilon(\mathrm{C}_{tm,cg=i,ts} - \mathrm{C}_{tm,cg=i,ts+1}) \times ts) \le \sum_{\forall ts\in TN} \Big(\varepsilon\Big(\sum_{\forall j\in N} \mathrm{L}_{eij}^{(tm,tn),k,ts} - \sum_{\forall j\in N} \mathrm{L}_{eij}^{(tm,tn),k,ts-1}\Big) \times ts\Big), \forall (t_m, t_n) \in R, cg \in CG \tag{11}$$

$$\sum_{\forall ts\in TN} (\varepsilon(\mathrm{C}_{tn,cg=i,ts} - \mathrm{C}_{tn,cg=i,ts-1}) \times ts) \ge \sum_{\forall ts\in TN} \Big(\varepsilon\Big(\sum_{\forall j\in N} \mathrm{L}_{eij}^{(tm,tn),k,ts} - \sum_{\forall j\in N} \mathrm{L}_{eij}^{(tm,tn),k,ts+1}\Big) \times ts\Big), \forall (t_m, t_n) \in R, cg \in CG \tag{12}$$

The optimization objective function in (1) aims to minimize the total run-time for all jobs. Constraints (2) and (3) guarantee no conflict in computation resources and the atomicity of tasks. Constraint (4) ensures that all tasks can be finished. Constraint (5) guarantees the continuity in time for computing. Constraint (6) ensures no conflict in communication resources. Constraint (7) guarantees the continuity in time for communication. Constraints (8) and (9) allocate the routing paths and time slices for all the transmission tasks. Constraint (10) ensures the dependency between tasks, which infers that the backward tasks will not be executed until receiving the results of the forward tasks. Constraint (11) ensures that the transmission tasks start after the forward computation tasks are finished. Constraint (12) ensures that the transmission tasks are finished before the backward computation tasks start.

4. Simulations and experiments

To evaluate the latency performance of the proposed JACC method, we conduct a simulation under a 16-nodes fat-tree topology, which is simplified from Tianhe Express-2 HPC system [8]. Under the constraints of the JACC IP model, we calculate the average buffering time of the HPC system. The lengths of time slices for transmission and computation are both set to 100 us. As shown in Fig. 4, the proposed JACC method reduces over 80% average buffering time at high load compared with the traditional one.

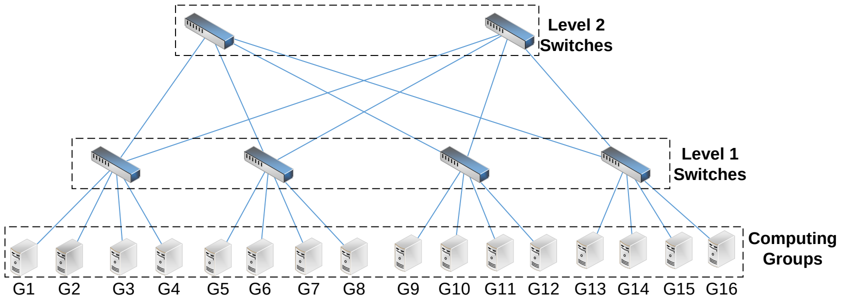

Fig. 3. Simplified Tianhe Express-2 topology for simulation

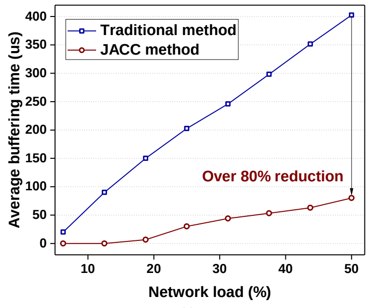

Fig. 4. Average buffering time vs. load

A prototype experiment is carried out to evaluate the performances of the JACC method as shown in Fig. 5. The experiment system consists of four computing groups, two level-1 switches (SW1 and SW3) and one level-2 switch (SW2). Four computing jobs which contain 8 tasks have to be scheduled into the computing groups. Traditional job scheduling method will cause link congestion between SW1 and SW2 leading to a 100-us buffering time. The JACC method optimizes task allocation and successfully avoids link congestions. The DSO screen shots in Fig. 5 present the transmission time slices and show that 57.52-ns end-to-end latency with zero buffering time is achieved through JACC.

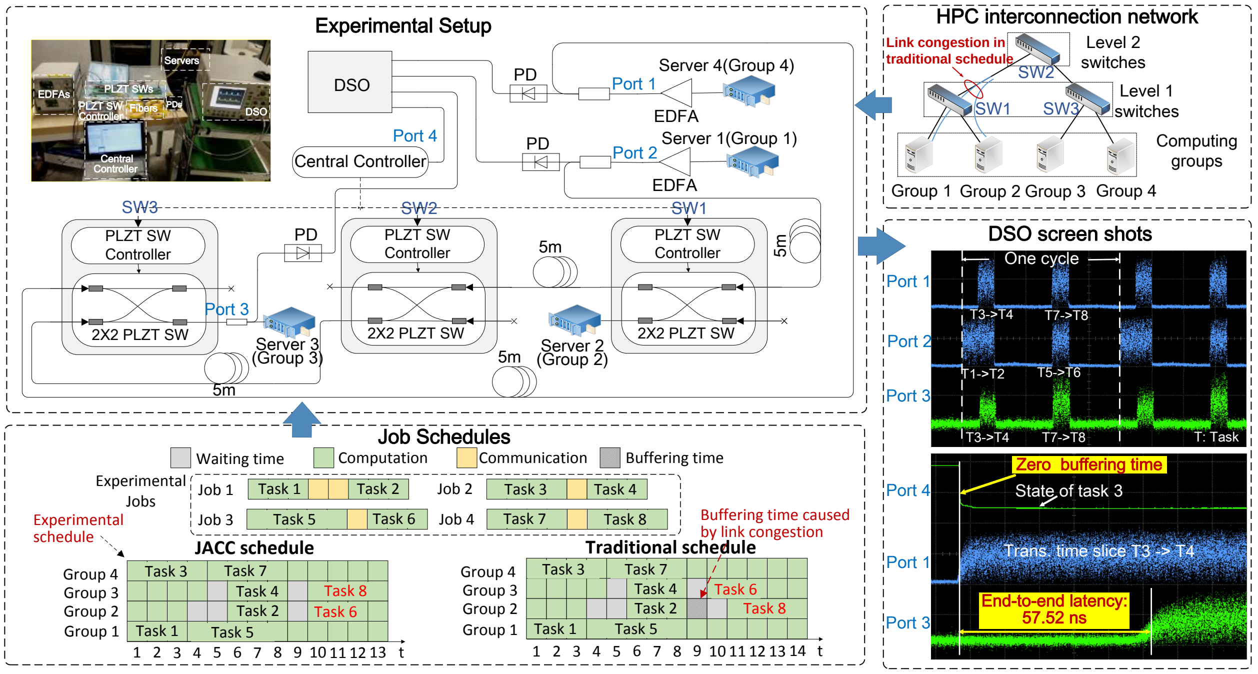

Fig. 5 Prototype experiment of JACC

5. Conclusions

We propose an OTSS-based JACC method for ultralow-latency optical interconnection in HPC systems. An IP model for JACC is proposed to minimize the mismatch between the job schedule and time slices transmission. In this way, the end-to-end latency brought by buffering time can be significantly reduced. The simulation and prototype experiment results show that over 80% of the buffering time can be reduced at high load, and the 50-ns-level end-to-end latency is achieved without buffering in the intermediate switching nodes.

6. References

[1] H. Fu, *et al.*, *SCIS*, 59(7): 072001, 2016.
[2] J. Barker, *et al.*, in *Proc. Conf. on Supercom.*, SC'05:16-38, 2005.
[3] M. Cyriel, *et al.*, in *Proc. High Perf. Interconnects*, 29-35, 2005.
[4] N. Hua and X. Zheng, US Patent 9,608,763, 2017.
[5] N. Hua, *et al.*, in *Proc. ONDM*, 1-4, 2017.
[6] R. Luo, *et al.*, in *Proc. OFC*, M2E.6, 2018.
[7] P. Brucker, "Scheduling Algorithms", *Springer*, 2007.
[8] X. Liao, *et al.*, *SCST* 30(2): 259-272, 2015.