LETTER

On a Cup-Stacking Concept in Repetitive Collective Communication

Takashi YOKOTA[†a)], **Kanemitsu OOTSU**[†], ***and*** **Shun KOJIMA**[†], ***Members***

SUMMARY Parallel computing essentially consists of computation and communication and, in many cases, communication performance is vital. Many parallel applications use collective communications, which often dominate the performance of the parallel execution. This paper focuses on collective communication performance to speed-up the parallel execution. This paper firstly offers our experimental result that splitting a session of collective communication to small portions (slices) possibly enables efficient communication. Then, based on the results, this paper proposes a new concept *cup-stacking* with a genetic algorithm based methodology. The preliminary evaluation results reveal the effectiveness of the proposed method.

key words: *interconnection networks, parallel computers, collective communication*

1. Introduction

Interconnection network is one of the vital components in parallel supercomputers [1]. A parallel program runs as an aggregate of concurrent processes that interchange their data occasionally or frequently. Thus, inter-process communication is essential in parallel computation and parallel performance is described as a some kind of mixture of computation and communication.

Communications are carried out according to the instructions in the parallel program. Many of them are based on collective communication that forms some kind of logical structure. This paper focuses discussions on reducing duration time of collective communication.

This paper assumes packet exchange networks that have regular network topology, 2-dimensional torus, and each router in the network operates the deterministic routing algorithm for simplification of discussions. Furthermore, this paper assumes static approaches that arrange communication strategy beforehand.

The unique point in this paper is to split a large packet into a series of small ones that are injected with an adequate interval. In general collective communication, each node injects a packet in unison. If the packet is small, the communication will be completed rapidly. However, when large packets are injected, the network suffers from severe congestion for a relatively long time. What is the most essential problem here is few degrees of freedom in controlling the vast amount of packets. Splitting large packets into small portions is worth considering.

This paper further arranges the injection timing of small packets, as well as the inter-packet interval, so that interferences between packets are minimized. We propose a new concept, *cup-stacking*, that runs well in many of collective communication patterns.

The rest of this paper is organized as follows. After Sect. 2 summarizes related work, Sect. 3 shows that collective communication may be accelerated by splitting a large packet. Based on the results, Sect. 4 proposes a new method *cup-stacking*, and Sect. 5 evaluates the effectiveness. Finally, Sect. 6 concludes this paper.

2. Related Work

Discussing congestion control methods is essential for maximizing the performance of interconnection network. Once a local congestion occurs, a blocked packet obstructs other ones so that the congested situation spreads, which is described as tree saturation [2].

One of the promising approaches is to smooth the packet flow. The major reason of severe congestion is too dense packet flow that cannot resolve local congestion. Throttling is one of the typical idea, where packet injection is appropriately controlled according to the network situation [3]. Pacing is another idea for smooth packet flow [4].

As a static approach, a packet scheduling method is proposed [5], where packet injection timing is optimized. The method operates effectively, however, it has not reached at clear and systematic solutions. Furthermore, the idea of packet splitting is not discussed so far.

3. Splitting Packet with Constant Interval

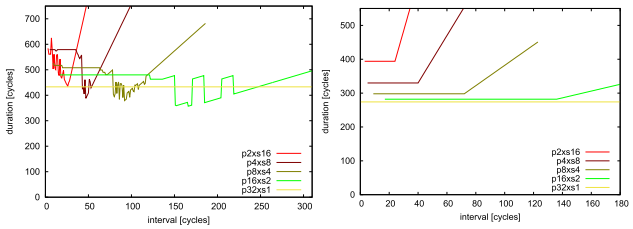
In general, in packet exchange networks, a packet consists of a header and payload. The former includes necessary information for packet delivery and the latter contains objective communication data. A packet forms with a series of *flits*, and we assume that a packet consists of a header flit followed by succeeding p payload flits, i.e., the packet size in length is expressed as $l_c = p + 1$ [flits].

In general, communication performance of the network is discussed in terms of throughput and latency. As the throughput means the number of received payloads in every time unit, the header flit is treated as an overhead. Thus, small packets pay relatively large overheads and large pack-

Manuscript received October 25, 2021.
Manuscript revised March 16, 2022.
Manuscript publicized April 15, 2022.

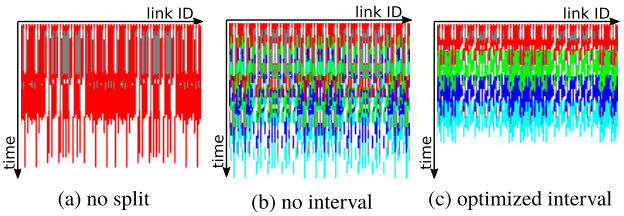
[†]The authors are with Utsunomiya University, Utsunomiya-shi, 321–8585 Japan.
a) E-mail: yokota@is.utsunomiya-u.ac.jp

DOI: 10.1587/transinf.2021EDL8098



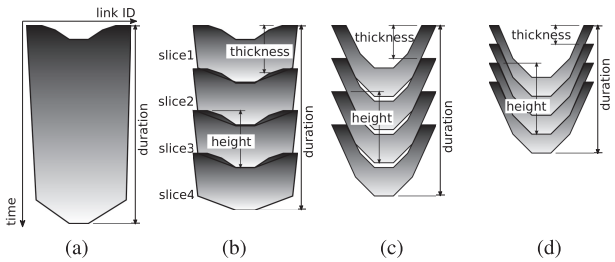
(a) Bit-reverse traffic (brev) (b) Transpose traffic (trns)

Fig. 1 Duration times for inter-slice intervals (16×16, $p = 32$).



(a) no split (b) no interval (c) optimized interval

Fig. 2 Visualized communication behavior.



(a) (b) (c) (d)

Fig. 3 Abstract model of cup-stacking concept.

ets are welcomed in terms of throughput.

In a congested situation, once a packet conflict occurs, the situation continues according to the packet size. We firstly focus on propagation behaviors of the congested situation. A locally congested situation will easily induce further congestion when the situation sustains. However, from the opposite point of view, we can expect that the congested situation is expired by short packets.

The important point here is not to propagate congestion. It is necessary that the network traffic is smooth when a local congestion is resolved [6]. This consideration conducts us to the new idea that a large packet is split to multiple portions that are injected with an interval.

We call a split packet *slice* to clearly distinct from the original packet. When a p-flit payload packet is split to s slices, each slice consists of $\lfloor p/s \rfloor + 1$ flits whereas the original one $l_c = p + 1$.

We have tried a preliminary experiment for feasibility study of the *slice* idea by using the environment described in Sect. 5.1. We split a 32-flit payload packet to 2, 4, 8, and 16 slices and measured duration times by varying the inter-slice interval. Figure 1 shows the results. Horizontal axis shows the inter-slice interval and the vertical axis shows the total duration time. Only the exception is the horizontal solid line that shows the duration time in the non-splitting case. In this figure, the notation ppxss means that an original packet is split into s slices with p-flit payload.

Figures 1 (a) and (b) show two typical results. In many traffic patterns, except tornado and transpose traffic, duration time marks the minimum value that supersedes the original duration.

4. Cup-Stacking

4.1 Observation of Communication Situation

As shown in Fig. 1, in some traffic patterns, duration time curves show complicated behaviors for inter-slice intervals. It seems difficult to precisely model the behavior analytically (and mathematically), however, we take an intuitive approach. Figure 2 visualizes the communication behavior of bcmp traffic in 8×8 2D-torus network. In this figure, each horizontal position shows the corresponding (physical) link of the router. For example, the leftmost position corresponds to the north link of router $(0, 0)$. Vertical axis shows the time. This figure draws a colored dot when a flit is transferred via the link. Different colors show different slices: red, green, blue, cyan colors show the first, second, third, and fourth slice, respectively. A gray dot means that the packet transfer is suspended because of busy buffer and a white dot shows the link is not used at the time.

Figure 2 (a) shows the non-splitting case and Fig. 2 (b) shows the consecutive four-slice case where the duration time becomes worse as described in the previous section. Figure 2 (c) also shows the four-slice case but it has an appropriate inter-slice interval. We can visually recognize that an appropriate interval leads a 'tidy' appearance of communication situation that results in performance improvement.

4.2 Cup-Stacking Concept

Based on the observation results, we abstract the shapes of slice (in Fig. 2) to model a new concept. Figure 3 illustrates the abstracted concept. As shown in Fig. 2 (a), ordinary (i.e., non-split) communication starts and ends as Fig. 3 (a) illustrates. Figure 3 (b) models the sliced communication with an appropriate interval (that corresponds to Fig. 2 (c)).

Here, we focus on the shape of the slice. As Fig. 2 (b) shows, when the inter-slice interval is small, packets are disordered in terms of slice. The disordered transfer means that packets from different slices interfere with each other, whereas the tidy situation (as Fig. 2 (c)) expects smooth flow of packets.

Our idea is *effective thickness* of the shape in slice communication. If the slice communication is re-shaped as Fig. 3 (c), we can shorten the inter-slice interval so that the overall duration time is reduced. Figure 3 (d) shows the optimized situation where the no rooms exist between consecutive slices.

We call the concept *cup-stacking*. Cups are stacked and stored in the kitchen cabinet smoothly, even if each cup has a certain height. In this paper, the shape of slice communica-

tion corresponds to the *cup*. Each slice has certain *height* (in duration time), however, if the effective thickness of slice is small, slices can be started successively with a short interval that is given by the thickness.

4.3 Cup-Stacking Method

The previous section offers a conceptual model, cup-stacking. Then, we discuss a practical method, cup-stacking method, as the second step. The problem is simple, that is, how we can re-shape the slice communication, and our first answer is to manage packet injection timing, not unison injection.

Inefficiencies in network communication mainly come from interferences between packets transferred in-flight. Interference is caused by resource conflict in which multiple packets require the same resource in a router simultaneously. A simple way to resolve the conflict is to manage the packets' arrival times to the router where the conflict occurs. This paper simply manages injection timing of packets.

Solving an optimal injection timing is a class of NP-hard problems especially in large-scale networks. A packet is relayed via a number of routers until it reaches the destination. Thus, even when a packet injection is delayed to solve a local conflict, the packet may cause a new conflict at another router. To handle the hard problem, we introduce the genetic algorithm (GA).

In our GA method, a gene simply consists of packet injection times at every node, presented as I_i for i-th node. Maximum value of I_i is conducted by $I_{max} = t_{d0} - l_c$, t_{d0} is the duration time of the slice where $I_i = 0$ for all nodes and l_c is the packet length (in flit unit).

We prepare two evaluation functions for the GA operation: *height* and *effective thickness* in the previous section. The former is determined by the duration time of slice t_d. To define the latter parameter, we introduce the idea of *occupation time*. As shown in Fig. 2, every router link differently behaves according to the transferring packets. The occupation time is given by the time length between the first and last flits transferred via the link. Given the occupation time of link k as O_k, the effective thickness is defined as $T_e = max(O_k)$.

Mutation operation of the GA randomly selects one or more members (I_i) in a gene (i.e., chromosomes) and changes their values within the range of $[0 : I_{max}]$. Every gene is mutated in every generation except when the crossover operation takes place. If the mutated gene performs worse, the original one survives.

Crossover operation of GA firstly sorts the genes by the evaluation value order. Then, it systematically selects two genes from the sorted order as $(1, 2)$, $(1, 3)$, $(2, 3)$, $(1, 4)$, $(2, 4)$, $(3, 4)$, and so on, where (a, b) means a combination of two genes at the a- and b-th orders. The child gene overrides the weaker parent gene and the stronger one survives. The crossover function substitutes randomly-selected members in the a-th gene (I^a) to the corresponding members in the b-th gene (I^b) as $I^a_i = I^b_i$.

Furthermore, to prevent local-minima situations, we introduce *lifetime* and *watchdog timer* in our GA method. The former restricts the maximum number of generations of a gene, and the latter re-sets a timer at every update of the best score of the evaluation value. The timer re-initializes all the genes at its expiration time.

5. Evaluation

The objective issues in this evaluation section are (1) how we can re-shape the slice communication to fit for the cup-stacking, and (2) how the re-shaped slice works effectively in the cup-stacking method where the slice is repeatedly applied with an adequate interval. Thus, we firstly apply the GA method that is described in the previous section, we then apply the resolved (re-shaped) slice to fit the cup-stacking by adjusting the inter-slice interval.

5.1 Evaluation Environment and Condition

We built an evaluation environment by extending our cellular automata (CA) based simulator ([7]) for rapid and precise experiments. The simulator models a simple non-pipelined router and it assumes two-dimensional torus topology. Throughout the evaluations, the number of virtual channels (VC) is three and each VC has a four-flit buffer at the every input port of router. Routing algorithm is dimension order where each packet firstly traverses along the x-axis then it goes in y-direction until it reaches the destination. We use six traffic patterns as Table 1 summarizes. In this table, X and Y represent x- and y- addresses, respectively. Small letters show bit representation: x_i means the i-th bit in X. W is concatenation of Y and X: $W = w_{2n-1} \cdots w_0 = y_{n-1} \cdots y_0 x_{n-1} \cdots x_0$. In this evaluation section, unlike the previous sections, we uniformly use 8-flit packets that has a 7-flit payload and a header, so that we can discuss the effects of the re-shaped slices for various division numbers.

GA parameters are as follows. The number of genes is 50. Crossover operation is carried periodically in the specified interval $i_c \in \{100, 200\}$ [generations]. Every member in the gene is selected to crossover at a given possibility $r_c \in \{0.1, 0.2, 0.4, 0.6, 0.8, 0.9\}$. When the crossover operation is not carried out, mutation operation is applied

Table 1 Traffic patterns used.

abbreviation	description
bcmp	bit-complement. $w_{2n-1}w_{2n-2}\cdots w_0 \longrightarrow \overline{w_{2n-1}}\,\overline{w_{2n-2}}\cdots\overline{w_0}$
brev	bit-reverse. $w_{2n-1}w_{2n-2}\cdots w_0 \longrightarrow w_0\cdots w_{2n-2}w_{2n-1}$
brot	bit-rotation. $w_{2n-1}\cdots w_1 w_0 \longrightarrow w_0 w_{2n-1}\cdots w_1$
shfl	perfect shuffle. $w_{2n-1}w_{2n-2}\cdots w_0 \longrightarrow w_{2n-2}\cdots w_0 w_{2n-1}$
torn	tornado. $W \longrightarrow \mathrm{mod}(W + N/2, N^2)$
trns	transpose. $(X, Y) \longrightarrow (Y, X)$

Table 2 16×16 network results [cycles].

traffic	unit slice ($p = 7$)				$p = 14$			$p = 28$		
	unison (us)		GA		$s =$	$s = 2$		$s =$	$s = 4$	
pattern	t_{d0}	T_e	t_d	T_e	1	us	GA	1	us	GA
bcmp	65	62	53	50	109	104	91	193	181	164
brev	108	105	79	76	199	179	151	381	326	298
brot	91	86	78	75	178	147	143	363	259	271
shfl	95	90	71	67	243	152	133	427	269	256
torn	67	64	64	64	123	131	131	235	259	259
trns	74	71	71	71	130	138	138	242	266	266

where every chromosome is selected at another probability $r_m \in \{0.1, 0.05, 0.02\}$. Lifetime is given as $lt \in \{0, 100, 200\}$ [generations] and the watchdog timer is set as $wd \in \{0, 100, 200\}$ [generations], where $lt = 0$ and $wd = 0$ mean that the corresponding functions are inhibited. We ran possible combinations of parameter values where each combination has five simulation runs. We ran two independent GA sessions for t_d and T_e as evaluation values for each parameter set.

5.2 Results

Table 2 shows the evaluation results of 16×16 2D-torus network. In this table, *unison* (*us* for short) means that all the packets are injected in unison, i.e., $I_i = 0$ for all nodes, *GA* means the our proposed method.

Firstly, let us compare the results in the *unit slice* section in the table. This section has four columns that show the heights (t_{d0} and t_d) and effective thicknesses (two T_e's), respectively. By comparing the corresponding values in unison and GA, we can confirm the effectiveness of the proposed method. For example, the GA method reduces 27.6 percent of the effective thickness in brev traffic: T_e of brev is reduced from 105 to 76.

Then, we compare the collective communication performance in the two payload cases (i.e., $p = 14$ and $p = 28$). Each case consists of three columns: the leftmost column (that is marked $s = 1$) shows the duration time of the non-split case, where all the packets are injected in unison. The center column (marked *us*) shows the duration time when each slice starts in unison with an optimized interval. The right column (GA) shows the minimum duration time that the optimized slice by the proposed method marks. The proposed method marks at most 45 percent improvement in the $p = 14$ case in shfl traffic (i.e., duration time is reduced from 243 to 133 [cycles]).

5.3 Discussions and Future Work

The proposed method reduces both height (duration time) and effective thickness (maximum occupation time) of a slice in bcmp, brev, brot and shfl traffic patterns. These results suggest that network behavior in a collective communication involves much room to optimize (like *bubbles*) and that the proposed method can *press* the behavior to eliminate the room so that the communication performance is improved.

This paper tries two objective evaluation metrics, height and thickness. As a simple intuition from the core idea of the proposed method, the shortest thickness marks the best performance (i.e., the shortest duration). However, the evaluation results shows that our forecast is not always true. Although the effective thickness of slice, height, and the total duration time have strong correlation, small thickness does not necessarily guarantee the short duration time.

Then, we will take a look at the resulting cup-stacking situations in terms of fragility (or robustness). As described in Sect. 4, the duration time by the cup-stacking method is $T_d \leq T_e(s-1) + t_d$, where t_d and T_e are height and effective thickness of a slice, respectively, and s is the number of slices split from the original form.

In the unison injection case, where all the packets are injected at the same timing, $T_e(s-1) + t_d = 417$ and $T_d = 326$, their difference is 91 [cycles]. On the other hand, in the cup-stacking method, $T_e(s-1) + t_d = 307$ and $T_d = 298$, and the difference is 9 [cycles]†. This means that the slice in the former case involves many *bubbles* and it can tolerate shorter intervals than the effective thickness. On the other hand in the latter case, the slice is rather rigid since the bubbles do not sufficiently absorb the inter-slice interferences.

We recognize that the observation results offer us a hint for further improvement of the proposed method, which will be the our future work.

6. Conclusions

To improve the performance of collective communication, this paper firstly discussed a new idea of packet splitting and showed that appropriate interval may improve performance. Based on the preliminary results, this paper presented a new concept of *cup-stacking* and a GA-based method, where packet injection timing is optimized.

Evaluation results reveal that the proposed method can improve collective communication performance at most 45 percent in 16×16 torus network. This paper further discussed the robustness of the method for further improvements.

Acknowledgments

This work was partly supported by JSPS KAKENHI Grant Number 20K11726.

References

[1] W.J. Dally and B.P. Towles, Principles and Practices of Interconnection Networks, Morgan Kaufmann Pub., 2004.
[2] G.F. Pfister and V.A. Norton, "'Hot spot' contention and combining in multistage interconnection networks," IEEE Trans. Comput., vol.C-34, no.10, pp.943–948, 1985. DOI: 10.1109/TC.1985.6312198.
[3] E. Baydal, P. López, and J. Duato, "A family of mechanisms for congestion control in wormhole networks," IEEE Trans. Parallel Distrib. Syst., vol.16, no.9, pp.772–784, 2005. DOI: 10.1109/TPDS.2005.102.

† 16×16 network with brev traffic pattern ($s = 4$).

[4] H. Shibamura, "Active packet pacing as a congestion avoidance technique in interconnection network," Parallel Computing: On the Road to Exascale, vol.27, pp.257–264, 2016. DOI: 10.3233/978-1-61499-621-7-257.
[5] T. Yokota, K. Ootsu, and T. Ohkawa, "A static packet scheduling approach for fast collective communication by using PSO," IEICE Trans. Inf. & Syst., vol.E100-D, no.12, pp.2781–2795, Dec. 2017. DOI: 10.1587/transinf.2017PAP0015.
[6] T. Yokota, K. Ootsu, F. Furukawa, and T. Baba, "Phase transition phenomena in interconnection networks of massively parallel computers," Journal of the Physical Society of Japan, vol.75, no.7, 078401 (7 pages), 2006. DOI: 10.1143/JPSJ.75.074801.
[7] T. Yokota, K. Ootsu, and T. Ohkawa, "Accelerating large-scale interconnection network simulation by cellular automata concept," IEICE Trans. Inf. & Syst., vol.E102-D, no.1, pp.52–74, Jan. 2019. DOI: 10.1587/transinf.2018EDP7131.