

Global Communication Schemes on QCDOC¹

Y. Deng² and J. Glimm

Department of Applied Mathematics, Stony Brook University, Stony Brook, NY 11794 and
Center for Data Intensive Computing, Brookhaven National Laboratory, Upton, NY 11973

J. W. Davenport

Center for Data Intensive Computing, Brookhaven National Laboratory, Upton, NY 11973

Abstract

We present a global Allgather communication algorithm on the QCDOC, a switchless architecture with nearest neighbor communications on a high dimensional torus. This communication is needed for several applications, such as the molecular dynamics algorithm involving Ewald summation for long-ranged interactions, matrix multiplication, and the solution of linear systems. We show near optimal utilization of network bandwidth, for Allgather communication on this communication network. We reaffirm our previous conclusion that, coupled with our communication algorithms, this architecture is scalable for molecular dynamics algorithms even though the system was designed specifically for QCD simulations

Keywords: Global communication, Allgather, latency, bandwidth, torus, nearest neighbor, molecular dynamics

Submitted to IEEE Trans. Para. & Dist. Comp.

¹ Supported in part by DOE contracts DE-FG02-90ER25084 and DE-AC02-98CH10886, the National Science Foundation DMS-0102480 and the Army Research Office grant DAAG-01-0642.

² Corresponding author's email address: Yuefan.Deng@StonyBrook.edu.

1 Introduction

In a previous paper¹, we have shown that the high degree of hardware communication parallelism (24 channels per node) of a proposed novel supercomputer architecture allows exceptional scalability for simulation of molecular dynamics using the Ewald algorithm for long range (Coulomb) forces, for up to tens of thousands of nodes. The architectural design we refer to is a product of a collaboration among physicists and computer scientists at Columbia University, IBM, and RIKEN of Japan. It is called the QCDOC (Quantum Chromodynamics On Chip)^{2,3} in view of its special purpose conception of providing the optimal platform for QCD simulations.

It has been traditionally difficult to achieve scalability for fine grained parallelism on massively parallel processing systems with 1000s of nodes. For example, in the case of molecular dynamics (MD) simulations, we have projected¹ scalability for fine grain parallelism involving as few as 14 particles per node for 10,000 nodes on the QCDOC. The 6D nearest neighbor network of this architecture acts as a switch. The switch is the most important element in most conventional parallel computers, and the QCDOC network performs far better than the conventional switches do. A switch is the least scalable part of conventional supercomputer designs, with its nearly $O(N^2)$ cost and a practical limit on the number (some number of 100's) of ports which can support cross bar all-to-all message passing. To scale up the computer system, one must connect many distributed switches. This practice results in a topology that is locally centric but globally imbalanced for the uniform and global communications required by many applications. In other words, switches, which form a hierarchical topology, may provide adequate connectivity for nodes on one switch or a fast bus, but they typically offer too little connectivity for global communication for nodes on different or remote switches. This design feature causes serious barriers to programmers for the design of scalable algorithms. As the system size grows, this scalability problem becomes more difficult to overcome.

Our main point is to regard the network components of the QCDOC as a distributed programmable switch, in other words, as comprising an array of distributed programmable mini switches sandwiched in the array of distributed computing units. While this principle is very attractive from a hardware perspective for large systems, a weakness of this design is the burden on the programmers to manage messages and their buffers at a low level in such a distributed environment. The problem is amplified by lack of communication protocols, a common pitfall in dealing with hardware designs that represent a significant advance over previous architectures. As with most parallel computers, the first users must acquire an intimate knowledge of the hardware for developing applications software, particularly, in the present case, the message passing software.

As the first non-QCD applications developers, we address one aspect of the software issues in the present paper, *i.e.*, the design of global communication algorithms which optimize the distributed high performance network. We consider Allgather communication as perhaps the most demanding of the communication algorithms. This

communication problem is of interest in the Ewald algorithm for molecular dynamics, which plays an essential role in many physical and life science applications.

A number of authors have considered all-to-all communication in a variety of network topologies^{4,5,6}. It is common to consider the communication problem in which each node sends a different message to each other node. For n nodes, this problem generates n^2 distinct messages and requires $O(n^2)$ message transition steps. This problem is called personalized all-to-all communication. However, the Ewald algorithm for long range forces for MD is not of this type. The particle coordinates must be sent from each node to every other one. Thus each node has unique data, which is communicated to every other node. In other words, all nodes must gather the pieces of different data from each node to form a complete data set, as illustrated in Figure 1. For this restricted all-to-all communication, termed Allgather in the MPI classification^{7,8}, a simple lower bound on the number of communication steps is Ln , where L is the diameter of the network. The diameter is defined as the number of message hops required to communicate between the two most distant pairs of nodes.

2 The Allgather Algorithm Design Requirements

We proposed¹ an Allgather timing formula

$$(2.1) \quad t_{\text{all to all}} = \frac{sx}{n_{\text{channels}}} + n_{\text{hops}} \times L$$

where n_{channels} is the number of channels (24 in this architecture) per node, s is the inverse bandwidth per channel or the time required per byte when sending a large message, x is the total message size to be received by or sent from each node, n_{hops} is the number of messaging hops and L is the latency per single message. This formula is a theoretical lower bound on the timing, as it assumes perfect utilization of all hardware capabilities. Software contributions to the communication time are expected to follow the same formula, and so they can be included through modification of the parameter L and, if necessary, s . Hardware parameters give a low latency L (350 ns), and an estimate of a very minimal software layer, considering the function of store-and-forward⁹, raises this estimate¹⁰ to 660 ns. These figures indicate that the second term is smaller than the first for a system with tens of thousands of nodes. Here we discuss a detailed plan to realize a high fraction of the transmission estimate of the first term. The nonuniformity of the torus in different dimensions will lead to some deviation from this formula, as we shall see.

The QCDOC network is a six dimensional torus. For each of the six dimensions, a computational node has four channels (thus 24 in total) allowing independent communication (send and receive) in each direction along the simple ring network which is defined by one dimension of the torus. Three of the six torus dimensions are hard wired on the motherboard to be periodic, and are thus fixed in hardware to have size 2. Three dimensions have external ports on the motherboard, allowing a flexible 3D lattice topology. The most general topology we shall consider is thus a $2 \times 2 \times 2 \times L \times M \times N$ 6D torus. Our detailed estimates will be based on a $2 \times 2 \times 2 \times 10 \times 10 \times 10$ torus with 8,000 nodes. For this case, the maximum necessary number of hops for a message is

$1+1+1+5+5+5 = 18$ since $\lceil t/2 \rceil$ hops are required to reach all points of a ring of size t using two-way communication. Here $\lceil \cdot \rceil$ denotes the integer part of a real number. In other words, the diameter of the system we consider is 18 hops.

The algorithm design issue we address is the layout of a sequence of messages which will achieve global Allgather communication of data. The algorithm must achieve the following goals:

- (1) The data carried by each message must be received uniquely. Because the network functions as a distributed switch, each multi-step message must select a unique optimal pathway for transmission. This requirement is a programming burden normally handled by a switch.
- (2) The totality of messages during any single communication step should be of the same size and should saturate all network channels, i.e., each node should send 12 messages and receive 12 messages, each through a single channel (each torus dimension and each direction relative to the torus), each of identical size. Any deviation from this goal will lower the effective bandwidth and thus increase the first term contribution to the total communication time beyond that predicted by (2.1). Due to network nonuniformity, we find a degradation of this objective by approximately a factor of 2, i.e., we obtain approximately 50% communication efficiency.
- (3) Messages, once received and used in computations and forwarded as discussed earlier[†], should be discarded for buffer space reuse. Any deviation from this rule will increase buffer size and thus decrease the actual message size for a fixed allocation of space for both message and buffer. As a result, one must increase the latency due to the need of more messaging startups, giving rise to an increased second term to the communication time proposed in (2.1).
- (4) Subject to the above restrictions, the usual goals of program simplicity apply. Specifically, any computational tasks required for communication decisions will compete with the node computations. Since these computations are actually the rate limiting step for the entire computation, we wish to minimize this contention for computational resources. The computation and (pure) communication utilize separate hardware units, and thus proceed concurrently. The longer of the two processes marks the actual time to accomplish both.

3 The Allgather Algorithm

3.1 For the Irregular 6D QCDOC Torus

As stated above, we assume a 6D torus network, with three factors of size 2 and the other three dimensions of unrestricted size t , or $2 \times t \times 2 \times t \times 2 \times t$.

We propose to send messages first through a single $2 \times t$ factor (all communication pattern channel pairs will participate in this step, concurrently, due to independence of all channels), then through a second $2 \times t$ factor to reach nodes displaced by a second pair of torus coordinate dimensions, and finally we send messages through the final $2 \times t$ factor. We begin with an analysis of communication through a single $2 \times t$ $2D$ torus. For $t \gg 2$, a 50% loss of efficiency occurs. We select a simple algorithm which approximates this limit. First, we communicate in the first factor of size 2. As a result all nodes have complete information, *i.e.*, shared data regarding this factor. In effect, data relating to the first factor plays no role in the remainder of the communication steps. Next we send messages cyclically in both directions in the second factor. For t even, the final of the $[t/2]$ cyclic messages is nonunique, with the same information communicated in each of the two directions for this step. In this case, we split the message in half and send half through each of the oriented channels available in the hardware.

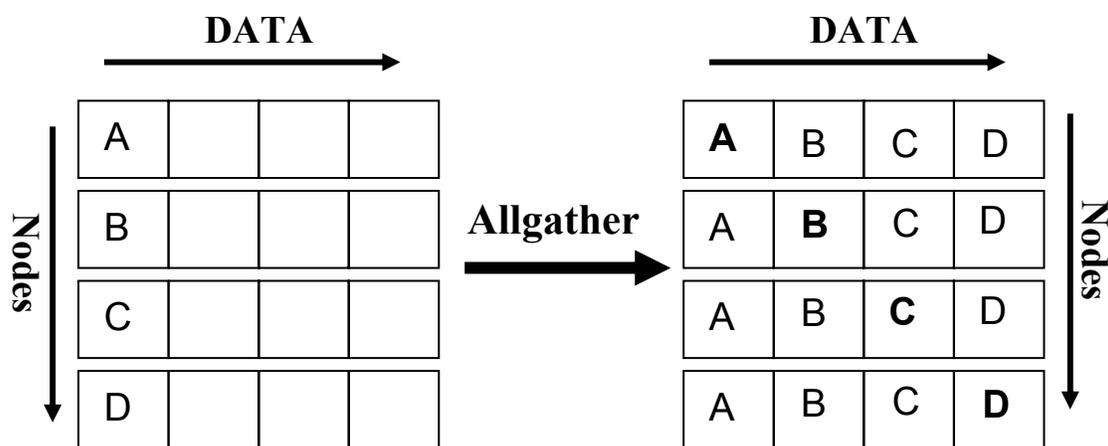


Figure 1 Illustration of Allgather message passing. Before Allgather, each one of the nodes 1, 2, 3, 4 possesses data A, B, C, and D respectively. After Allgather, all nodes have a complete data set ABCD gathered from individual nodes.

Each node of the $2D$ torus is labeled by a pair (n_1, n_2) of integers. Here $n_1 = 0, 1$ and $n_2 = 0, \pm 1, \dots, \pm[t/2]$ for t odd, with the final entry replaced by $t/2$ for t even. We use the same label to describe the data which is located on node (n_1, n_2) at the beginning of the communication algorithm. We introduce the notation $e_1 = (1, 0)$ and $e_2 = (0, 1)$ for the generators of the $2 \times t$ torus. Thus at the end of the first step, node (n_1, n_2) has data $(0, n_2)$ and $(1, n_2)$. The next $[t/2]$ steps communicate such pairs of data using the generators $\pm e_2$. After $l \pm e_2$ steps, node (n_1, n_2) has $2(l+1)$ data $(0, n_2), \dots, (0, n_2 \pm l), (1, n_2), \dots, (1, n_2 \pm l)$. For t even, the final move is two fold redundant. For this step, we split the data and send half in each direction. This algorithm uses half of the available communication channels at each step, and thus has a bandwidth efficiency of 50%.

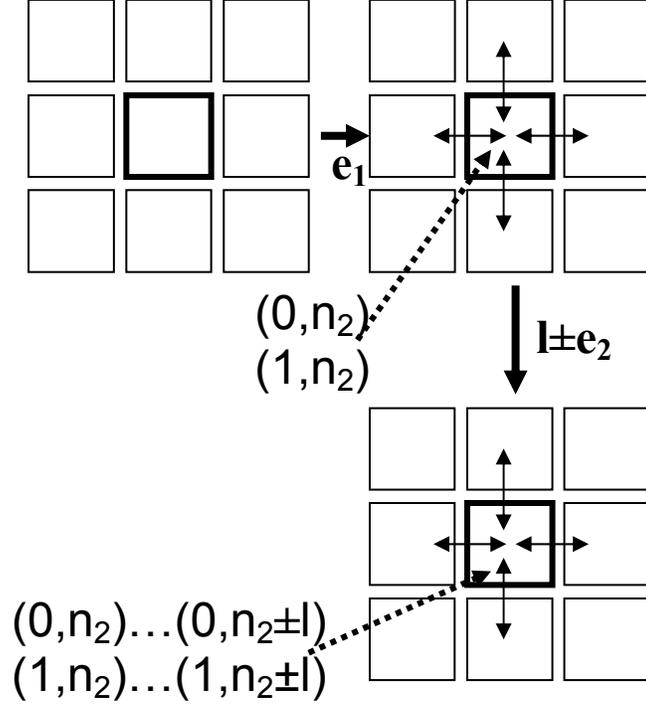


Figure 2 This graph illustrates the message passing process for a 2D torus for one node (n_1, n_2) with darker boarder in graph. After $l \pm e_2$ steps, it is filled with data $(0, n_2), \dots, (0, n_2 \pm l)$ and $(1, n_2), \dots, (1, n_2 \pm l)$.

Returning to the $6D$ torus, with three $2 \times t$ factors, we organize the messages into three groups of $1 + \lceil t/2 \rceil$ each. Let M_j denote message j . We have the following communication pattern:

- steps 1 through $1 + \lceil t/2 \rceil$: communicate in $1^{\text{st}} 2 \times t$ factor
- steps $2 + \lceil t/2 \rceil$ through $2 + 2\lceil t/2 \rceil$: communicate in a $2^{\text{nd}} 2 \times t$ factor
- steps $3 + 2\lceil t/2 \rceil$ through $3 + 3\lceil t/2 \rceil$: communicate in the $3^{\text{rd}} 2 \times t$ factor

We label nodes and data with a triple (k_1, k_2, k_3) , where each $k_i = (n_1^i, n_2^i)$ is the label for a single $2 \times l$ factor. The first block of steps proceeds exactly as for a single $2 \times l$ $2D$ torus, once the factor has been selected from the $\binom{3}{1} = 3$ possible choices. In the second block

of steps, the same algorithm would generate redundant messages, so that we must restrict, and send only some of these. This is accomplished by sending only data which has been transmitted further within its first factor than it has been within the second factor. Thus in the second step of the second block, step $3 + \lceil t/2 \rceil$ will not transmit data from the first step of the first block, *i.e.* step 1. Data at equal stages within the two blocks of steps will be called diagonally degenerate. The above rule does not provide guidance to break the redundancy for such data. The two $2 \times t$ factors have been selected from among

$\binom{3}{2} = 3$ choices. We introduce a cyclic order in the three factors. After cyclic reordering,

the three factors must be of the form $xx0$ where each x denotes a selected factor and 0 denotes the factor not selected. In this order, we pick the lower x factor as the channel for

communication for the diagonally degenerate terms. In the third block of steps, we follow the same rules, with data sent through the least advanced channel. Again we find doubly and triply degenerate cases. The doubly degenerate case is resolved as above. The triply degenerate case cannot be resolved by this method. We split the data three ways and send one third through each channel, or if this is too inefficient due to small message sizes, we incur a loss of bandwidth for these (rarely occurring) messages.

3.2 For a Regular D Dimensional Torus

We next consider a regular D dimensional torus with each factor of size t . The network has the structure of an Abelian group¹¹ G , with $|G| = t^D$ elements. More simply, the network is described as a group G which is a direct product of D factors, each being a cyclic group on t elements. Let $r = \lceil t/2 \rceil$ be the number of hops needed to traverse one dimension of G bi-directionally. Let $\mathbf{0}$ denote the origin of G . The messages are organized in groups of r messages each, with each group of messages filling out the communication within one dimension. Nodes are labeled by group elements, and data which reside at group element (or node) g before the allgather communication begins are also labeled by this group element g throughout the full communication. Let e_i denote the group element $(0, \dots, 0, +1, 0, \dots, 0)$. The elementary communication generators within a single lattice dimension are given by $\pm e_j$. For communication step l , $1 \leq l \leq r$, node g sends data $g \mp (l-1)e_i$ to node $g \pm e_i$ and receives data $g \pm le_i$ from node $g \pm e_i$. For the first block of r messages, this communication occurs concurrently in each dimension, and thus fills all available channels, with an exception for step r if t is even.

For later blocks of r messages, we follow a similar pattern, except that some (redundant) messages are not sent, to achieve uniqueness of message data received at each node. Consider the $j + 1$ block of r messages. For steps $jr + l$, $1 \leq l < r$, data passes through a channel distinct from the j channels already communicated in steps $k < jr$. Each of the $\binom{D}{j+1}$ distinct $j+1$ element subsets of the available D channels participate during these steps. Messages are sent only when the distance from sender to receiver for the data is smallest in the sending channel relative to any of the $j+1$ channels with a nonzero distance. Cases with equal distance allow a diagonally degenerate redundancy, to be resolved by further restriction of the messages, in a few cases with loss of bandwidth. Otherwise full bandwidth is preserved for all communications. The diagonally degenerate cases are grouped according to the number of channels with equal (smallest nonzero) distance. We introduce a preferred cyclic ordering of the dimensions, and relative to this order, we consider cyclic permutations of the dimensions. For each $j+1$ element subset of the D dimensions, we consider the effect of the cyclic permutation group on this subset. The cases which can be resolved without loss of bandwidth have an orbit of size D under the cyclic group and the group action at this point has a kernel (cyclic subgroup not modifying the subset in question) of size 1. In general the product of the orbit size and the kernel size remains to be a constant D . The kernel size determines the multiplicity of the message as optimally considered. If the message size allows efficient splitting, then

multiple fractions $1/[\text{kernel size}]$ are sent through each of the redundant channels concurrently, and are recombined at the receiving node to recover the original whole message. If the message is too small to split (due to increased latency), the message is sent through one of the available channels only while allowing others to idle, resulting in loss of bandwidth. Thus an arbitrary further restriction for this message increases the bandwidth for this message by a factor of the kernel size. Such communication patterns are rare relative to the total number of messages sent by this algorithm and they thus do not have a significant effect on the overall effective bandwidth.

3.3 Bandwidth and Message Sizes

For MD, we allow 3 doubles and an integer, *i.e.*, 28 Bytes, to describe the coordinates and charge of a single particle and consider them as the minimum necessary data to pass for one particle. The coordinates by themselves are sufficient although they occur in the potential as coordinate differences. The evaluation of various $\sin()$ and $\cos()$ functions appearing in the Ewald formula, seemingly requiring coordinate differences, can be carried out using only the values of the single coordinates by using trigonometric double angle formulas. For the QCDOC lattice, targeted at 8000 nodes, we see that the total data for one particle per node (total 8000 particles) is $8000 \times 28\text{B} = 224\text{KB}$, which fits well in the RAM of single node (total 4MB). Thus intermediate stages in the communication can be broken down as needed to fit into an available memory buffer. The exact formulas for messages can be worked out without difficulty, but are a bit messy, so we omit details.

Next we consider the case of a regular torus. The message sizes vary greatly with the steps. The middle steps have large messages, and need to be decomposed. Management of these message buffers is vital as the data cannot be altered until it has been used locally and have been sent.

To be explicit, we start with a $D = 6$ dimensional torus of size $t = 6$ and $r = \lfloor t/2 \rfloor = 3$, *i.e.*, $6^6 = 46,656$ nodes. The general case will be given below. There is an elementary message size estimate for the sum of three steps referring to a fixed number of coordinates. With r nonzero coordinates, we have a message size $5^r \binom{6}{r}$. The number 5 is the number of nonzero values $t + 1$ on each $1D$ ring of the torus, namely $\pm 1, \pm 2, 3$. Thus with M_j the message size at step j , we have

$$M_1 + M_2 + M_3 = 5 \times 6 = 30$$

$$M_4 + M_5 + M_6 = 25 \times 15 = 375$$

$$M_7 + M_8 + M_9 = 125 \times 20 = 2500$$

$$M_{10} + M_{11} + M_{12} = 625 \times 15 = 9375$$

$$M_{13} + M_{14} + M_{15} = 3125 \times 6 = 18750$$

$$M_{16} + M_{17} + M_{18} = 15760$$

If we add a trivial step 0 with $5^0 \binom{6}{0} = 1$ unit of data, then we see that the above is a binomial expansion for

$$(1+5)^6 = \sum 1^{6-r} 5^r \binom{6}{r} = 6^6$$

ensuring that all data has in fact been transmitted.

Next we want to examine the message sizes for the individual steps. We assert that

$$\text{step } j = 3r \text{ has } \binom{6}{r} \text{ terms}$$

$$\text{step } j = 3r - 1 \text{ has } [3^r - 1] \binom{6}{r} \text{ terms}$$

$$\text{step } j = 3r - 2 \text{ has } [5^r - 3^r] \binom{6}{r} \text{ terms}$$

so the above sum rules are necessarily enforced.

The case $j = 3r$ is the easiest to understand. The factor $\binom{6}{r}$ comes from selection of the r nonzero coordinates out of the 6 available. All data entries must have the value 3, so no further choice is possible.

Next we consider step $j = 3r$ and $j = 3r - 1$ combined. These terms are those with values ± 2 , or 3, unrestricted. Thus there are 3^r possibilities, and the stated value for step $3r - 1$ follows by subtraction. Finally the value for $j = 3r - 2$ follows from this by subtraction also.

We can summarize as follows:

$$\begin{aligned} M_0 &= 1 \\ M_1 &= 12, M_2 = 12, M_3 = 6 \\ M_4 &= 240, M_5 = 120, M_6 = 15 \\ M_7 &= 1960, M_8 = 520, M_9 = 20 \\ M_{10} &= 8160, M_{11} = 1200, M_{12} = 15 \\ M_{13} &= 17298, M_{14} = 3872, M_{15} = 6 \\ M_{16} &= 14896, M_{17} = 728, M_{18} = 1 \end{aligned}$$

The maximum buffer requirement occurs at step 14 where the size is $M_{12} + M_{13} + M_{14} = 21185$ units of data.

For a D dimensional torus, each dimension forms a circle of size t , the messages group into families of $r = \lceil t/2 \rceil$ each, as each of the dimensions is traversed. Extending the above reasoning, we see that the message size M_j can be recovered from the formula

$$M_{j_r} + \dots + M_{j_{r-l}} = \binom{D}{j} \times \begin{cases} (2l+2)^r & \text{if } r \text{ is odd} \\ (2l+1)^r & \text{if } r \text{ is even} \end{cases}, \quad l = 0, \dots, r-1$$

References

- ¹ Y. Deng, J. Glimm, J. W. Davenport, X. Cai, and E. Santos, *Performance models on QCDOC for molecular dynamics with coulomb potentials*. In Int. J. High Performance Computing Applications, Submitted (2003).
- ² P. A. Boyle, D. Chen, N. H. Christ, C. Cristian, Z. Dong, A. Gara, B. Joó, C. Kim, L. Levkova, X. Liao, G. Liu, R.D. Mawhinney, S. Ohta, T. Wettig, A. Yamaguchi, *Status of the QCDOC project*. In Nucl.Phys.Proc.Suppl. **106** (2002) 177-183.
- ³ IBM Blue Gene Team, *Blue Gene: A vision for protein science using a petaflop supercomputer*. In IBM systems, **40** 310-327 (2001).
- ⁴ Y. Saad and M. H. Schultz, *Topological properties of hypercubes*, IEEE Trans. on Computers, vol. 37, no. 7 (1988) 867--872.
- ⁵ S. Hinrichs, C. Kosak, D. R. O'Hallaron, T. M. Stricker, and R. Take, *An Architecture for Optimal all-to-all Personalized Communication*. In Proc. of SPAA '94, ACM, (1994) 310-319.
- ⁶ C.-C. Lam, C.-H. Huang, P. Sadayappan, *Optimal Algorithms for All-to-All Personalized Communication on Rings and Two Dimensional Tori*. In J. Parallel and Distributed Computing, **43** (1997) 3-13.
- ⁷ M. Barnett, L. Shuler, S. Gupta, D. G. Payne, R. van de Geijn, and J. Watts, *Building a high-performance collective communication library*. In Supercomputing, IEEE Computer Society Press, Los Alamitos, CA, (1994) 107--116.
- ⁸ M. Snir, S. Otto, S.Huss-Lederman, D. Walker, J. Dongarra, *MPI: The Complete Reference*, MIT Press (1995).
- ⁹ D. Chen, N. H. Christ, C. Cristian, Z. Dong, A. Gara, K. Garg, B. Joo, C. Kim, L. Levkova, X. Liao, R. D. Mawhinney, S. Ohta, T. Wettig, *QCDOC: A 10-teraflops scale computer for lattice QCD*, In Nucl. Phys. Proc. Suppl. **94** (2001) 825-832.
- ¹⁰ P. Boyle, Private communication. (Tests on QCDOC emulator yield a latency of 550 ns for gauge theory simulations. As expected, the latency depends on applications and quoting 660 ns as the latency is reasonable.)
- ¹¹ D. Arnold, *Abelian groups and representations of finite partially ordered sets*, Springer-Verlag, New York (2000).