

AWGR-Based Optical Topologies for Scalable and Efficient Global Communications in Large-Scale Multi-Processor Systems

Xiaohui Ye, S. J. B. Yoo, and Venkatesh Akella

Abstract—In large-scale multi-processor computing systems, global communications are typically supported by an auxiliary network (e.g., IBM Blue Gene) or with hardware support in the network (e.g., NEC Earth Simulator). We explore the potential for realizing efficient global communications that can scale beyond a million processors by harnessing the unique parallelism and wavelength routing properties of optical devices. Specifically, we use an arrayed waveguide grating router (AWGR) device as the basic building block in realizing scalable global communication. The AWGR is a passive switch fabric (wavelength router) that uses multiple wavelengths to interconnect outputs and inputs by following a specific cyclic wavelength routing (permutation) pattern. We analyze different network topologies using AWGR devices for barrier synchronization and propose techniques to pick parameters of the network for a given number of processors. We compare the performance and energy consumption for barrier synchronization with what is achievable with state-of-the-art electrical networks.

Index Terms—Barrier synchronization; Large-scale parallel computing; Optical interconnects; Scalable synchronization.

I. INTRODUCTION

This paper explores the potential of optical interconnects to address the global communication needs of next-generation high-performance computing systems. Operations in the control plane that require global communications, such as barrier synchronization, reduction, etc., are critical in multi-processor systems, since they are widely used to structure parallel algorithms and large-scale data analysis. These operations are characterized by one-to-all and all-to-one communications among all or a large subset of processors, which makes scalability and ensuring low latency challenging [1]. As a result, high-performance computers often have hardware support for global communications—either as a separate network or as dedicated hardware for implementing “collective” operations in the switches. For example, Blue Gene [2] has a separate network, while T3E [3], Earth Simulator [4], and Blue Waters have special functional units in the router to support collective operations. Compared to such electronic global communications, optical communications can exploit massive parallelism available in the wavelength domain.

In this paper, we explore the potential optical devices in the form of an arrayed waveguide grating router (AWGR) to realize a control plane global communication network that is arbitrarily scalable and offers lower latency and lower power consumption compared to networks based on electronic switches. An AWGR [5] is a passive optical device that is widely used in telecom networks to support multiplexing (MUX), demultiplexing (DEMUX), and wavelength routing. Over the past decade, significant advances have occurred in the building of a chip-scale AWGR integrated with other optical components to provide various functions [6–8].

We propose a network of AWGRs, the Generic AWGR-based optical Global Communication Network (G-AGCNet), to support global communication in a large-scale multi-processor system. The G-AGCNet can realize connectivity as a multi-ary butterfly network [9] with the wiring overhead only as a 2-ary butterfly network. This paper uses barrier synchronization as an example to illustrate the architecture of the G-AGCNet, but, as noted before, the proposed network supports efficient one-to-all and all-to-one communication, which could be useful in many ways. Specifically, we address the following problem: given a \( k \times k \) AWGR device, how can one realize an efficient network and algorithm to implement barrier synchronization in a system with \( N \) nodes, where \( N \) could be as high as a million? We develop analytical models for the relationship between network size (\( N \)) and \( k \).

This paper makes the following contributions. First, we develop a performance model and derive the optimal values for the different network parameters to minimize the number of sequential transmissions required to complete a barrier synchronization. Second, we show that a straightforward implementation of a G-AGCNet requires additional fixed wavelength converters, which adds to the cost and complexity of the network. We propose network topologies to eliminate the use of wavelength converters. Third, we show how the proposed network can perform data reduction operation efficiently to demonstrate the generality of the network. Finally, we compare the performance of the proposed network with state-of-the-art
electrical networks to perform global operations such as barrier synchronization.

The remainder of this paper is organized as follows. Section II presents the related work. Section III discusses the characteristics of the AWGR. Section IV presents the architecture of the G-AGCNet and the algorithms for realizing efficient barrier synchronization. Performing the reduce operation on the G-AGCNet is discussed in Section V. Section VI provides a comparison with other datacenter network topologies as well as latency and energy consumption analyses for the G-AGCNet. Section VII presents our conclusions.

II. RELATED WORK

Barrier synchronization for multi-processors has been studied for many years. Many efficient designs and algorithms have been proposed for on-chip multi-processor systems [10,11]. In the area of large-scale multi-processors various approaches have been taken to support global operations. For example, IBM Blue Gene adopts a dedicated network for barrier synchronization [2]; Cray T3D has the AND-tree barrier synchronization circuits [12]; another Cray multi-processor, T3E [3,13], embeds the barrier synchronization operation in the data communication network, while the routers are equipped with barrier synchronization units to map a virtual topology onto a physical topology; and NEC Earth Simulator [4] relies on a centralized control unit to perform barrier synchronization.

An optical bus-based all-to-all barrier network has been proposed in [14], but it is not very scalable, because it uses a shared bus. Binkert et al. [15] proposed an on-chip optical barrier network that is also bus based but uses micro-resonators as filters. Free-space optical interconnects are used to support barrier synchronization in [16], while both source and detector arrays contain all processors and barrier points. These designs are suitable for a system on a chip, but are not intended to scale to millions of nodes, as we discuss in this paper. To the best of our knowledge, there is no prior work involving the use of AWGR devices to realize global communications for large-scale multi-processors.

There has also been a lot of interest in performing the reduce operation efficiently especially with its widespread use in large-scale data analysis in the form of MapReduce operations. Many efficient platform-independent algorithms for this operation have been proposed [17,18]. Various architecture-specific all-reduce schemes have also been developed.

Recently, the topology of a data center network (DCN) has been extensively studied, and many new topologies have been proposed. Some representative topologies include Fat-tree [19], Torus [2], Flattened Butterfly [9], Dragonfly [20], DC cell [21], and FiConn [22]. In contrast to using sophisticated top-level switches in a traditional tree topology, the goal of the above-mentioned DCN topologies is to utilize small-size commodity switches to construct large-scale networks, so that the cost and power consumption are reduced. Note that most DCN topology designs focus on one-to-one data communications, while global communications often requires all-to-one, one-to-all, or all-to-all communications. Thus, a network tailored to support one-to-one data communications may not be good at handling global communications, and vice versa. Communication parallelism is strongly emphasized in G-AGCNet designs compared to other DCN approaches. For example, in DCell [21], if a DCell of nodes, a DCell consists of nodes, and each DCell has only one link to the other DCell under the same DCell; while for a AGCNet built based on AWGRs, a higher-level AGCNet contains lower-level AGCNet, but has wavelength channels between any two lower-level AGCNet, if each lower-level AGCNet contains nodes. Another fundamental difference between an AGCNet and the above-mentioned DCNs is that the AWGR switches used in the AGCNet do not introduce latencies as traditional electrical switches. In an AGCNet, the switching function is realized at the source node by sending out the signals on proper wavelengths and AWGRs perform wavelength routing. No switching latency is introduced by AWGRs, since packets travel through AWGRs at the speed of light without being stopped for label processing and routing table lookup. Furthermore, signals on multiple wavelengths can reach one AWGR output concurrently due to wavelength parallelism, so that no arbitration at the switch input and output is necessary and the corresponding arbitration and queuing delay can be avoided.

The AWGR device has been first used for optical MUX and DEMUX to exploit its cyclic wavelength routing property [23]. The AWGR has also been widely used in optical switch designs for many years [24–27]. Traditionally, the AWGR serves as a non-blocking switching fabric in the switch design for telecommunications networks [24–26], where each AWGR input or output can only serve one packet at a time, since wavelength converters that can only process one packet at a time are placed before AWGR inputs and after AWGR outputs. Recently, the AWGR-based switch design for datacenter networks allows multiple packets arriving at one AWGR output concurrently by adding an optical DEMUX after the AWGR output [27].

To support the need of efficient global communications in multi-processor systems, we place an optical MUX and optical DEMUX at the AWGR input and output, respectively, so that one-to-all and all-to-all communications can be realized. Compared to other AWGR-based switches, another unique feature of the G-AGCNet is that no control plane associates with each AWGR in the G-AGCNet: the wavelength is set by the source and detector arrays contain all processors and barrier points. Signals on multiple wavelengths can reach one AWGR output concurrently due to wavelength parallelism, so that no arbitration at the switch input and output is necessary and the corresponding arbitration and queuing delay can be avoided.

In addition to the AWGR, the optical switching fabric can also be constructed based on microelectromechanical systems (MEMS) [28] and a semiconductor optical amplifier (SOA) [29]. The MEMS-based switching fabric is cost effective in handling slowly changing unicast traffic but not suitable for global communication patterns. The SOA-based switching fabric internally utilizes the broadcast and select mechanism, but incurs unnecessary power consumption, since all ports may not require the broadcasted signals. In contrast to MEMS-based and SOA-based optical switching fabric, the AWGR-based switching fabric is efficient at handling one-to-all, all-to-one, and all-to-all communications in an energy-efficient manner, since the signal is delivered to only the desired ports via certain wavelengths.
III. The Arrayed Waveguide Grating Router

In this section, we will describe the operation of the AWGR device that forms the building block for the network topologies described in the paper. As Fig. 1 illustrates in an example, the AWGR allows the signal from one input to reach one output only on a particular wavelength determined by its cyclic routing property. If we have a set of wavelengths $\{\lambda_i, 0 \leq i < k\}$, we can use $\lambda_{\text{mod}}(1 - i - j, k)$ to deliver signals from input $i$ to output $j$. Figure 1 shows a wavelength routing map of a $5 \times 5$ AWGR with 5 wavelengths. At AWGR input 2, one can use $\lambda_0, \lambda_2, \lambda_3$, and $\lambda_4$ to reach outputs 0, 1, 2, 3, and 4, respectively; one can also use $\lambda_2, \lambda_0, \lambda_4$, and $\lambda_3$ to reach AWGR output 4 from inputs 0, 1, 2, 3, and 4, respectively. Note that wavelength routing in an AWGR exhibits a symmetric and cyclic routing characteristic, e.g., $\lambda_{\text{mod}}(1 - i - j, k)$ can also be used for routing signals from input $j$ to output $i$.

The AWGR is unique in that all-to-all communication can be realized if every node is equipped with multiple receivers and multiple transmitters working on different wavelengths so that signals on different wavelengths can be transmitted and received concurrently. Furthermore, the AWGR fabric by itself is passive and low loss at all data rates, which is a significant advantage in itself, as power consumption is a major obstacle to scalability in computing, especially at high speeds. The capability of building chip-scale AWGRs and integrating AWGRs with other components, such as transmitters and receivers [6–8], makes the use of an AWGR-based switching system in a multi-processor system feasible.

IV. AWGR-Based Optical Global Communication Network

Barrier synchronization [30] is one representative of global communications used in multi-processor systems. Multi-processor computing systems utilize barrier synchronization to execute parallel algorithms in stages. All processes (running on different processors) must announce stage completion and be subsequently informed that all other processes have also finished the stage: synchronization is achieved and the barrier may be passed allowing program execution to continue. Barrier synchronization normally consists of two stages, if the barrier is implemented based on a tree structure: an update stage, wherein processors update a root processor that a barrier has been reached, and a notification stage, wherein all processors are informed that the barrier may be passed. The two stages can be viewed as a global all-to-one communication followed by a global one-to-all communication. The two stages can be combined into one stage if low-latency all-to-all communications are feasible in each step.

This section uses barrier synchronization as an example of global communications to illustrate the AGCNet architecture. This section introduces two AWGR topologies, called AGCNet-I and AGCNet-II, and then proposes a generic architecture, G-AGCNet, which treats the AGCNet-I and the AGCNet-II as two special cases. In the proposed networks, each processor node connects to one port of the $k \times k$ AWGR using $k$ fixed wavelength transmitters and $k$ receivers, as shown in Fig. 2, so that a $k \times k$ AWGR and all processor nodes it connects can be treated as a compute node (CN) in the IBM PERCS [31] terminology. To simplify the discussion, we will use the term “processor” to refer to processor node for the rest of this paper. Furthermore, it is assumed that $k$ wavelengths are available for the wavelength routing in the AWGR. All-to-all connectivity can be realized within a CN, as each processor can talk to all other processors within the same CN on different wavelengths concurrently. The optical MUX aggregates $k$ wavelengths from $k$ transmitters, and the optical DEMUX distributes $k$ wavelengths for $k$ receivers. Only one bi-directional fiber is required to connect a CN with an input and an output of the AWGR. In order to reduce the cost and size, a processor node can replace $k$ fixed wavelength transmitters with a few tunable wavelength transmitters, so that most benefit provided by wavelength parallelism can still be gained, while the number of transmitters equipped for each processor can be significantly reduced.

A. AGCNet-I

As Fig. 3 illustrates, if the number of processors ($N$) is less than $k$, one AWGR can support an all-to-all barrier through connecting all $N$ processors. Figure 4 shows a simple one-step barrier algorithm (S1BS). The scheme is called one step, as all transmitting and all receiving are independent and can proceed in parallel because of wavelength parallelism. If $N$ is larger than $k$, multiple AWGRs are required for the construction of a hierarchical barrier synchronization network. We call processors “local processors” if they are located in the same CN as is the target processor; otherwise, we call them “remote processors.”

One way to interconnect multiple CNs involves the use of some ports of the AWGR for inter-CN connections. In general, for a $k \times k$ AWGR, $m (m < k)$ ports can be used to connect with $m$ CNs, one port for each CN. Then for an one-level hierarchy, the network has $m + 1$ CNs in total, thus supporting $(m + 1) \cdot (k - m)$
BEGIN
FOR each processor $p_i (0 \leq i < k)$
WAIT for the update sent from $p_j$ on $\lambda_{\text{mod}(i,j,k)} (0 \leq j < k, i,j)$
(S1) If $p_i$ reaches its barrier
SEND the update to $p_j$ on $\lambda_{\text{mod}(i,j,k)} (0 \leq j < k, i,j)$
(S1) ENDIF
IF $p_i$ receives updates from all local processors RETURN
ENDIF
END

Fig. 3. (Color online) An all-to-all connection is realized by connecting $k$ nodes to a $k \times k$ AWGR.

processors. The maximum value $(k/2 + 1) \cdot (k/2)$ is reached when $m = (k/2 - 1)$ or $m = k/2$ if $k$ is even and when $m = (k - 1)/2$ if $k$ is odd. Figure 5(a) shows an example with $k = 8$ and $m = 3$. Figure 6 shows the two-step barrier algorithm (S2BS-I) for the one-level hierarchical network, where M0 and M1 denote the set of ports connecting to the local processors and other AWGRs, respectively. It is called a two-step scheme because at least two steps are required, since some updates can be sent only after certain updates are received. The first step is the same as it is in S1BS, where local processors exchange updates in an all-to-all manner. In the second step, each processor exchanges updates with $m$ remote processors, one from each CN, still in an all-to-all manner. Consider the network shown in Fig. 5(a) as an example; in the first step, processors 0 to 4, 5 to 9, 10 to 14, and 15 to 19 exchange updates; in the second step, processors 0, 5, 10, and 15, processors 1, 6, 11, and 16, processors 2, 7, 12, and 17, processors 3, 8, 13, and 18, and processors 4, 9, 14, and 19 exchange updates. Note that an update delivered in the first step indicates only that one particular processor reached the barrier, while an update delivered in the second step indicates that all processors in one CN have reached the barrier.

For a given processor, the $m$ remote processors it (directly) connects to are determined solely by the AWGR wavelength routing pattern when the network is set up. When two AWGRs are connected by the two ports with the same index, the two processors connected with the ports having the same index are directly connected, e.g., processor 1 in CN(0) and processor 6 in CN(1) are directly connected in Fig. 5(a). To utilize this AWGR wavelength routing characteristic, ports with the same indices are chosen for inter-CN connections. If, in each AWGR, ports 0 to $m - 1$ are used for inter-CN connection, no wavelength converter modules (WCMs) are required if $m$ is an odd number, since the number of ports with the same index is even; while $(m - 1)/2$ WCMs must be used if $m$ is an even number, since the number of ports with the same index is odd. Figure 5(b) shows an example with $k = 8$ and $m = 4$. Figure 5(c) shows the wavelength conversion required at each WCM, assuming signals on $\lambda_{\text{mod}(1 - i, j, k)}$ can be delivered from input $i$ to output $j$. Figure 5(d) shows an example implementation of a WCM. In Fig. 5(b), if we keep the same connections but without using WCMs, the wavelength used to direct signals traveling from port 5 to 3 in CN(2) will finally reflect the signals back after traveling through 3 AWGRs (5 → 3 in CN(2), 0 → 0 in CN(3), 3 → 5 in CN(2)). The similar reflecting pattern is also applied for signals coming from port 7 to port 3 in CN(2) and from port 4 and 6 to port 0 in CN(3). Note that wavelength converters are all static and perform fixed wavelength conversion in order to realize direct connections between some particular processor pairs. In addition, the wavelengths involved in the conversion at each WCM are different, which complicates the design and deployment for a large-scale network. Furthermore, the power consumption of the entire system increases due to the use of wavelength converters.

In order to eliminate the use of WCMs while allowing each processor to directly reach four remote processors in four CNs, port 4 of CN(1), CN(2), CN(3), and CN(4) can be used for inter-CN connections, as shown in Fig. 5(e). Thus, processors connected with ports 5 to 7 in each CN are directly connected.
with processors connected with the ports having the same index in another CN. The processors connected with port 4 in CN(0), port 1 in CN(1), port 3 in CN(2), port 0 in CN(3), and port 2 in CN(4) are now directly connected with each other, and signals from any processor in this group must travel through five AWGRs to reach another processor, e.g., the wavelength routing path from port 4 in CN(0) to port 1 in CN(1) is 4 → 1 in CN(0), 1 → 4 in CN(2), 4 → 1 in CN(3), 1 → 4 in CN(4), 4 → 1 in CN(1).

In general, we propose the following inter-CN connection scheme to eliminate the use of WMGs:

\[
P^j_{mod(2 \cdot j - 1, m - 1)} \rightarrow P^m_{mod(2 \cdot j - 1, m - 1)} \quad (1.1)
\]

where \(0 \leq j < m - 1, 2 \cdot j \leq i < j + m - 2\), when \(m\) is odd;

\[
P^j_{mod(i, m)} \rightarrow P^{i + j}_{mod(i, m)} \quad (1.2)
\]

where \(0 \leq j < m - 1, 2 \cdot j \leq i < j + m - 2\), when \(m\) is even; the superscript is the index of a CN, while the subscript is the index of a port.

By applying the proposed scheme, when \(m\) is odd, ports 0 to \(m - 1\) in each CN are used for inter-CN connections; when \(m\) is even, ports 0 to \(m\) except for port mod(\(2 \cdot j - 1, m\)) in CN(j) (\(0 \leq j < m\)) are used for inter-CN connections. In addition, when \(m\) is even, the processors connected with the port with the index mod(\(2 \cdot j - 1, m\)) of CN(j) (\(0 \leq j < m\)) are all directly connected, thus becoming remote processors of each other, and signals leaving one of those processors must travel through \(m + 1\) AWGRs to reach another processor. In contrast, signals sent from a processor connected with port \(i\) (\(m < i \leq k\)) of a CN travel through only two AWGRs to reach its remote processors. Traveling through more AWGRs incurs higher power losses, especially when \(m\) is a large even number. In order to keep the uniform power loss on any path between two remote processors, port mod(\(2 \cdot j - 1, m\)) of CN(j) (\(0 \leq j < m\)) can be left unconnected when \(m\) is even, or \(m + 1\) ports in each CN can be used to form a network with \(m + 2\) CNs. In both cases, one CN connects with \(k - m - 1\) processors. Thus, \(m\) will be restricted to an odd number in later discussion.

To extend the network to connect more processors, one can organize the ports used for inter-CN connections as the inter-switch connections in a multi-ary multi-flat Flattened Butterfly (FB) network [9]. The ports used for inter-CN connections can be divided into groups. Each group corresponds to the inter-CN connections of one level of the hierarchy. Figure 7 shows an example topology with two levels of hierarchy. A CN(1) consists of CNs connected by the inter-CN connections, and a CN(1 + 1) consists of CNs connected by the ith level inter-CN connections (also denoted as inter-CN(1) connections). In general, the ports used for inter-CN connections can be divided into \(h\) groups, thus generating an \(h\)-level hierarchical network, which we call AGCNet-I. Each AWGR has \(m_i\) ports for inter-CN(1) (\(1 \leq i \leq h\)) connections and \(k - \sum_{i=1}^{h} m_i\) ports for connecting with local processors within a CN. The barrier algorithm Sh BS-I for an \(h\)-level AGCNet-I has \(h + 1\) steps. In the ith step of Sh BS-I, a processor exchanges updates with remote processors connected by the inter-CN(1) connections. The total number of processors that an \(h\)-level AGCNet-I can connect is

\[
N = \left( \frac{h}{1} \right) \cdot \left( k - \sum_{i=1}^{h} m_i \right). \quad (2)
\]

Equation (2) reaches its maximum value when all \(m_i\) are equal to \((k - 1)/(h + 1)\). Considering that \(m_j\) should be odd numbers, \(m_j\) can be either \(2 \cdot a - 1\) or \(2 \cdot a + 1\), where \(2 \cdot a - 1 < (k - 1)/h + 1\). If we assume that \(|m_j| = 2 \cdot a - 1\), \(|m_j| = 2 \cdot a + 1\), \(h = 10, N = 2 \cdot a \cdot h + h - 2 \cdot h + 1\), Eq. (2) becomes

\[
N = 2^h \cdot a \cdot h^{11} \cdot (a + 1)^{h11} \cdot (k - 2 \cdot a \cdot h + h - 2 \cdot h + 11), \quad (3)
\]

where \(0 \leq h_{11} \leq h\) and \(a\) is the only integer in the interval \([(k - h) - 2]/(2 \cdot (h + 1)), (k + h)/(2 \cdot (h + 1))\). For different \(k\) and \(h\), \(N\) reaches its maximum at different \(h_{11}\). If only \(k\) is fixed, \(N\) reaches its maximum \(2^h - 1\) when \(h = k - 2\) and all \(m_i\) are 1.

### B. AGCNet-II

Scaling the interconnection to one million processors in AGCNet-I would require AWGRs with at least 21 ports. Furthermore, the size of AWGRs used must be even larger if one wants to reduce the number of levels of the hierarchy for such a large-scale AGCNet-I. Although constructing the network based on large-size AWGRs can greatly simplify the architecture, each processor must have a large number of transmitters and receivers to utilize the wavelength parallelism, which in turn increases the complexity and cost of the processor. In this section, we investigate a more scalable topology, AGCNet-II, which allows constructing a large-scale network even based on relatively small-size AWGRs.

Instead of using some ports of the AWGR in a CN to connect directly with other CNs, an alternative way of connecting multiple CNs involves the use of a dedicated AWGR as a "switch" for inter-CN connection, as shown in Fig. 8. Here, an AWGR switch is purely an AWGR without any AWGR as a "switch" for inter-CN connection. The updated processor AWGR is shown to switch the AWGR AWGR connections). In general, we propose the following inter-CN connection scheme to eliminate the use of WMGs:
Fig. 8. (Color online) Using one $5 \times 5$ AWGR “switch” to connect five CNs.

BEGIN
FOR each processor $p_i$ with $0 \leq i \leq 4$

WAIT for the update sent from local processors and remote processors

(S1) If $p_i$ reaches its barrier

(S1) SEND the update to local processor $p_j$ on $\lambda_{s,w}(t_{i,j},h_0) \not\in \mathcal{E}_k$

(S2) ENDIF

(S2) If $p_i$ receives update from all local processors

(S2) SEND the update to the directly connected remote processor

(S2) ENDIF

(S3) If $p_i$ receives the update from the directly connected remote processor

(S3) RELAY the update to local processor $p_j$ on $\lambda_{s,w}(t_{i,j},h_0) \not\in \mathcal{E}_k$

(S3) ENDIF

IF $p_i$ receives updates from all local and remote processors

RETURN

ENDIF

END

Fig. 9. The three-step barrier scheme (S3BS-II).

one remote processor, and the corresponding remote processors of the processors within a CN are all in different CNs. As Fig. 9 illustrates, the barrier scheme for this architecture requires three steps to complete one barrier synchronization. We call this three-step scheme S3BS-II. Taking the network shown in Fig. 8 as an example, in the first step, processors 0 to 3, processors 4 to 7, processors 8 to 11, processors 12 to 15, and processors 16 to 19 exchange updates; in the second step, processors 0, 1, 2, 3, 5, 6, 7, 10, 11, and 15 exchange updates with processors 4, 8, 12, 16, 9, 13, 17, 14, 18, and 19, respectively; in the third step, again, processors 0 to 3, processors 4 to 7, processors 8 to 11, processors 12 to 15, and processors 16 to 19 exchange updates. In S3BS-II, while the first step is the same as that in S1BS, the second step can be viewed as an all-to-all barrier among CNs, as each CN sends $k - 1$ updates to the other $k - 1$ CNs connected by the AWGR switch. Since each processor knows the status of only the local processors and remote processors of one other CN after the second step, the third step is necessary for the exchange of updates received in the second step among local processors. For the network shown in Fig. 8, one can use multiple ports in a CN’s AWGR to connect to the AWGR switch, but the number of CNs that the AWGR switch can connect will be much smaller, since the AWGR switch is also a $k \times k$ AWGR. Therefore, each CN’s AWGR should have only one port connected to the AWGR switch.

Similar to extending a one-level AGCNet-I to a multi-level AGCNet-I, extending a one-level AGCNet-II to a multi-level AGCNet-II involves connecting multiple ports in one CN to multiple AWGR switches at different levels of the hierarchy. Figure 10 shows an example of a two-level AGCNet-II based on $5 \times 5$ AWGRs. Again, setting up direct connections between some remote processors requires the placement of WCMs on the corresponding paths. In general, for an $h$-level AGCNet-II, each CN’s AWGR has one port connected with the AWGR switch at each level of the hierarchy and $k - h$ ports connected with local processors within the CN. In contrast to an AGCNet-I, the addition of one more level of the hierarchy in the AGCNet-II requires the addition of two steps in the barrier algorithm. Therefore, the barrier scheme for an $h$-level AGCNet-II requires $2 \cdot h + 1$ steps to complete one barrier synchronization.

Considering the wavelength routing characteristics of the AWGR, in a multi-level AGCNet-II, no WCMs are required for all the connections with one particular AWGR switch if all CNs that connect with the AWGR switch use the ports with the same set of indices to connect with the switch and local processors; otherwise, WCMs are required to ensure the signal being delivered to the target output. For example, in Fig. 10, CN(1, 0) 0 \leq i \leq 3 use the ports with indices from the same set {0, 1, 2, 4} to connect with SW(1, 1) and local processors; thus, no WCM is required for connections with SW(1, 1); while for connections with SW(2, 2), the ports with the indices from the set {0, 1, 2, 4} are used to connect SW(2, 2) and local processors in CN(2, 2) and CN(3, 2), but the ports with the indices from another set {0, 1, 3, 4} are used in CN(0, 2) and CN(1, 2); thus, two WCMs are required for connections with SW(2, 2).

In order to eliminate the use of WCMs in a multi-level AGCNet-II, a CN can be replaced with a super-CN (SCN) which contains multiple CNs connected as a one-level AGCNet-I, as shown in Fig. 11. In an SCN, one CN uses only one port to connect with an AWGR switch, so that an SCN has a total of $h$ ports to connect with $h$ AWGR switches, one at each level. Since every CN in an SCN uses the ports with the same indices for intra-SCN connections, all CNs in an SCN can use a port with the same index to connect with all AWGR switches. If, for all SCNs, the ports used for intra-SCN connections have the same indices, the set of ports left for connecting with an AWGR switch and local processors will have the same indices for any CN. Therefore, no WCM is required in the SCN-based AGCNet-II.

Fig. 10. (Color online) An example of a two-level AGCNet-II with $h = 5$. 

Inter-CN connection

AWGR switch

WCM

Inter-CN connection

AWGR switch

WCM

Inter-CN connection

WCM
For an $h$-level AGCNet-II, the barrier synchronization within an SCN can be completed in two steps. The addition of one more level of hierarchy requires the addition of three steps in the barrier algorithm: one step for information exchanging among remote processors in different SCNs and two steps for information exchanging within an SCN. Therefore, the barrier scheme for an $h$-level AGCNet-II requires $3h + 2$ ($h > 1$) steps to complete one global barrier synchronization. The total number of processors that an $h$-level AGCNet-II can connect is

$$N = (k - h + 1)^h \cdot (k - h) \quad (h \geq 2).$$

Again, $h$ should be an even number so that the power loss on any path between two remote processors within an SCN can be kept uniform.

One can employ the multi-level AGCNet-I structure to increase the number of CNs contained in an SCN, so that the level of the hierarchy of the network can be increased, and the number of processors connected by an AGCNet-II can be increased. Assume that each SCN adopts an $h_{20}$-level AGCNet-I structure, and that in each level the number of ports (in a CN) used for inter-CN connection is $h_{2i}$ ($0 \leq i \leq h_{20}$, $h_{2i} \geq 1$, and $h_{2i}$ is odd); thus, the number of CNs that an SCN has is

$$h_2 = \prod_{i=1}^{h_{20}} (h_{2i} + 1),$$

and the number of processors in the AGCNet is

$$N = \left( k - \sum_{i=1}^{h_{20}} h_{2i} + 1 \right)^h \cdot h_2 \cdot \left( k - \sum_{i=1}^{h_{20}} h_{2i} \right).$$

The barrier synchronization among processors within an SCN can be completed in $1 + h_{20}$ steps. Therefore, the number of steps to complete one global barrier synchronization is

$$s = (h_{20} + 1) + (h_{20} + 2) \cdot \prod_{i=1}^{h_{20}} (h_{2i} + 1).$$

Adopting the multi-level AGCNet-I structure in an SCN can greatly increase the scale of the network, especially when $k$ is a relatively small number (e.g., for $k = 8$, $N$ can be $1.25 \times 10^7$ when $h_{20} = 3$ and $h_{21} = h_{22} = h_{23} = 1$), but the number of steps required to complete one global barrier synchronization also increases dramatically as the level of hierarchy in an SCN increases (e.g., if $h_{20} = 2$, $h_{21} = 4$ and $s \geq 19$; if $h_{20} = 3$, $h_2 \geq 8$ and $s \geq 44$). Note that the design goal of the AGCNet is to have a network that can efficiently conduct global communications for large-scale systems, which in turn requires minimizing $s$; thus, $h_{20}$ should be kept as small as possible.

From the connectivity point of view, the inter-SCN connection of the AGCNet-II is similar to a multi-ary multi-flat FB [9]. The difference is that a switch in a multi-ary multi-flat FB uses multiple ports to connect with other switches in the same level, whereas an SCN uses only one link to connect to an AWGR switch due to the wavelength parallelism and the use of wavelength routing. Therefore, the AGCNet-II can realize connectivity as a multi-ary FB with the wiring overhead only as a 2-ary FB. The AGCNet-II also has some similarity to the Dragonfly network [20]. The Dragonfly network constructs a large-radix group based on small-radix switches. In contrast, the AGCNet-II groups multiple CNs into an SCN not only because it can provide more inter-SCN connections when using the same size AWGRs, but also because it can help eliminate the use of fixed wavelength converters in inter-SCN connections. In addition, compared to the Dragonfly network, the layered inter-SCN connections increase the scalability of the AGCNet-II.

### C. G-AGCNet

The Generic AWGR-based optical Global Communication Network, G-AGCNet, is a general structure that treats both the AGCNet-I and the AGCNet-II as its special cases. An $h$-level G-AGCNet is composed of $h_1$ levels of the hierarchy following the AGCNet-I structure and $h_2$ levels of the hierarchy following the AGCNet-II structure, where $h = h_1 + h_2$. The number of steps that the corresponding barrier scheme has is

$$s = \begin{cases} 1 + h_1 & (h_2 = 0) \\ 3 + h_1 & (h_2 = 1) \\ h_1 + (h_{20} + 1) + (h_{20} + 2) \cdot h_2 & (h_2 > 1). \end{cases}$$

When $h_2 > 1$, the SCN structure is adopted to avoid the use of WCMs. The total number of processors that an $h$-level G-AGCNet can support is

$$N = (n + 1)^{h_2} \cdot \prod_{i=1}^{h_1} (m_i + 1) \cdot h_2 \cdot n$$

$$n = k - \sum_{j=1}^{h_2} h_{2j} \cdot \sum_{i=1}^{h_1} m_i, \quad h_2 = \prod_{j=1}^{h_{20}} (h_{2j} + 1).$$
Equation (9.1) can be maximized when fixing $h$ and $k$ or $k$ and $s$. $h$ represents the number of layers of the AGCNet, while $s$ represents the number of sequential transmissions (between processors) required to complete one global barrier synchronization. Table I shows the best values for $a$, $h_{10}$, $h_{11}$, and $h_{2}$ to maximize $N$, given $s$ and $k$. Furthermore, given $N$ and $k$, the best values for $a$, $h_{10}$, $h_{11}$, and $h_{2}$ can also be computed so that the number of steps required to complete one barrier synchronization can be minimized. Table I shows that, with four levels of hierarchy, the G-AGCNet can connect up to ten thousand processors when using $8 \times 8$ AWGRs as building blocks and more than 1.3 million processors when using $16 \times 16$ AWGRs. Table I also shows that, for a particular $s$, the G-AGCNet tends to have more layers following the structure of AGCNet-II when $k$ is small, and the G-AGCNet tends to have more layers following the structure of AGCNet-I when $k$ is large.

Table I also shows that, for a particular $s$, the G-AGCNet tends to have more layers following the structure of AGCNet-II when $k$ is small, and the G-AGCNet tends to have more layers following the structure of AGCNet-I when $k$ is large.

### V. Performing Reduction on the G-AGCNet

In Section IV, barrier synchronization is used as an example to illustrate the structure of the G-AGCNet. This section discusses the realization of a large-scale MapReduce [32] operation on the G-AGCNet. In general, we assume that a set of processors $\text{Proc}^M = \{p_1^M, p_2^M, \ldots, p_{N_1}^M\}$ is assigned to perform Map operations on $N_1$ non-overlapping work spaces, and each generates results for $N_2$ different keys $(\text{key}_1, \text{key}_2, \ldots, \text{key}_{N_2})$ on its own work space; another set of processors $\text{Proc}^R = \{p_1^R, p_2^R, \ldots, p_{N_2}^R\}$ outputs the results for Reduce operations, one key for each; for example, $p_{1}^R$ outputs the Reduce result for key_1. In order to aggregate all results for key_a $(1 \leq a \leq N_2)$
from Map operations, a tree structure that is rooted from \( P^R \) and contains all processors in \( \text{Proc}^M \) must be established. Since the G-AGCNet can be viewed as an optical multi-ary FB, for each node in the G-AGCNet, a multi-ary tree structure rooted from that node exists. Therefore, the Reduce operation can be easily implemented on the G-AGCNet.

Figure 12(a) shows an example based on a two-level AGCNet-I shown in Fig. 7, under the assumption that all processors perform Map operations. Note that some edges overlap for the two multi-ary trees, meaning that multiple key values are delivered on those edges. In addition, multiple Reduce operations can be performed in parallel on the G-AGCNet, while the number of steps to complete one reduction is exactly the same as that to complete one barrier synchronization. For one particular key, a tree construction process can be performed before executing the Reduce operation, so that each processor can know its parent and child processors. Figure 12(b) shows an example of the tree construction for the tree structures used in Fig. 12(a). The root processor first sends a tree construction message to all its local and remote processors. If a processor receives a tree construction message from a remote processor connected with the ith-level remote link, it will propagate the message to all other local processors and remote processors connected via lower-level remote links. If a processor receives a tree construction message through a local processor, which receives the message via the ith-level remote link, it will propagate the message only to the remote processors connected through lower-level remote links.

**VI. Latency and Power Analysis for the G-AGCNet**

In this section, barrier synchronization is used as an example to analyze the performance of the G-AGCNet. Note that the number of steps to complete a barrier synchronization represents the required minimum number of sequential transmissions between processors. Therefore, the number of steps to complete one global barrier synchronization can be used to compare the performance of the G-AGCNet with that of other datacenter networks. As most DCNs use electrical switches that adopt a store-and-forward mechanism for packet switching, we can assume that a barrier synchronization among \( n \) processors connected by one router via an all-to-all communication requires \( n \) sequential transmissions to complete: one transmission for a processor sending a broadcast message to the router, and \( n - 1 \) transmissions for the router sending \( n - 1 \) messages (from other processors) to each processor. Here, we use the Flattened Butterfly [9], the Dragonfly [20], and the DCell [21] as representative DCNs for comparison, and we assume that the barrier synchronizations on those networks are performed in a layer by layer manner, which is faster than a tree barrier, because the update and notification are performed concurrently. For a k-ary n-flat FB, a barrier synchronization among \( k \) processors connected by a router requires \( k \) steps. A barrier synchronization for one level of the hierarchy takes \( k + 1 \) steps: one step for \( k - 1 \) processors to send messages to the router, one step for the router to send those \( k - 1 \) messages to the other \( k - 1 \) routers in the same level, and \( k - 1 \) steps for the router to broadcast the received \( k - 1 \) messages (from the other \( k - 1 \) routers) to the processors. Therefore, it requires \( n \cdot (k + 1) - 1 \) steps to complete a barrier synchronization for a k-ary n-flat FB. For a Dragonfly network, we assume that one router has \( p \) ports to connect processors, \( a - 1 \) ports for intra-group connections, and \( h \) ports for inter-group connections, and the relation \( a = 2 \cdot p = 2 \cdot h \) holds [20]. A barrier synchronization among all processors connected by a router requires \( p \) steps. A barrier synchronization within one group containing a router requires \( 2 \cdot p + 4 \) steps: \( p \) steps for an all-to-all communication among processors connected by a router; one step for the aggregated broadcast message (that indicates all processors connected by a router arrive at their barriers) transmitted from processors to routers; one step for exchanging aggregated messages among routers; two steps for one router sending \( a - 1 \) messages to \( p \) processors (at most one processor receives two messages); and \( p \) steps for another round of all-to-all communication among processors connected by one router to exchange received aggregated messages. By applying a similar analysis, one can find that the total barrier synchronization can be finished in \( 4 \cdot p + 11 \) steps. For DCell, a barrier synchronization within a DCell requires \( n \) steps if one router connects with \( n \) processors; a barrier synchronization in a DCell requires \( 1 + 2 \cdot n \) steps; and a barrier synchronization within a DCell requires \( 2 \cdot k \cdot n + 2 \cdot k - 1 \) steps. Table II shows the number of steps required to complete one barrier synchronization for different sizes of G-AGCNet, FB, Dragonfly, and DCell. Clearly, the G-AGCNet can finish one global barrier synchronization much faster than other DCNs, since wavelength parallelism provides more parallel communication capacity in the G-AGCNet, which greatly benefits the global communications.

For the G-AGCNet, the total latency to complete one global barrier synchronization can be estimated as the sum of the time spent at each step. Since the AWGRs used in the G-AGCNet do not introduce switching latency as electrical switches, the latency of transmitting a packet from one
The latency of completing one global barrier synchronization (ns)

<table>
<thead>
<tr>
<th></th>
<th>100</th>
<th>10,000</th>
<th>100,000</th>
</tr>
</thead>
<tbody>
<tr>
<td>G-AGCNet</td>
<td>3</td>
<td>9</td>
<td>14</td>
</tr>
<tr>
<td>((k = 16))</td>
<td>((h_1 = 2, h_2 = 0))</td>
<td>((h_1 = 1, h_2 = 2))</td>
<td>((h_1 = 0, h_2 = 4))</td>
</tr>
<tr>
<td>Flattened Butterfly</td>
<td>17</td>
<td>34</td>
<td>49</td>
</tr>
<tr>
<td>((k\text{-ary n-fly}))</td>
<td>((k = 5, n = 3))</td>
<td>((k = 4, n = 7))</td>
<td>((k = 4, n = 10))</td>
</tr>
<tr>
<td>Dragonfly</td>
<td>23</td>
<td>43</td>
<td>103</td>
</tr>
<tr>
<td>((p = h = 3, a = 6))</td>
<td>((p = h = 8, a = 16))</td>
<td>((p = h = 23, a = 46))</td>
<td></td>
</tr>
<tr>
<td>DCell</td>
<td>15</td>
<td>31</td>
<td>55</td>
</tr>
<tr>
<td>((n = 3, k = 2))</td>
<td>((n = 3, k = 3))</td>
<td>((n = 6, k = 3))</td>
<td></td>
</tr>
</tbody>
</table>

Notes.

For G-AGCNet, \(k\): the size of AWGRs, \(h_1\): the number of levels of the hierarchy following the AGCNet-I structure, \(h_2\): the number of levels of the hierarchy following the AGCNet-II structure; for Dragonfly, \(p\): the number of terminals connected to each router, \(k\): the number of channels within each router used to connect to other groups, \(a\): the number of routers in each group; for DCell, \(n\): the number of servers in DCell, \(k\): the number of levels of the hierarchy.

The energy consumption of the G-AGCNet is the sum of the energy consumed by the transceivers. Regardless of the size of the network, there can be at most three cascaded stages of AWGRs; thus, the maximum light path attenuation is 10.3 dB, including power losses at the optical MUX and DEMUX (2 dB/each \([33,34]\), three AWGRs (2 dB/each, considering the AWGR and AWG to have comparable losses), two fiber connectors (0.1 dB/each), and fibers (0.2 dB/km, maximum

![Fig. 13. (Color online) The estimated latency to complete a global barrier synchronization for \(N\) processors. \(k\): the size of AWGRs, \(s\): the number of steps to complete a global barrier, \(h_1\): the number of levels of the hierarchy following the AGCNet-I structure, \(h_2\): the number of levels of the hierarchy following the AGCNet-II structure.](image-url)
500 m). For a receiver that has a sensitivity of −15 dBm (average 31.6 µW), the laser at the transmitter must emit the signal at 0.43 mW considering a power margin of 1 dB plus the path attenuation of 10.3 dB. A receiver (electrical part after O/E) working at 10 Gbps consumes 13.76 mW, if we conservatively assume that the power consumption of a receiver scales linearly with the data rate [35] and make the estimation based on a receiver shown in [36], which works at 6.25 Gbps and consumes 8.6 mW. The power consumption of the photodiode (O/E) is negligible. On the transmitter side, the power consumption of the modulator is 4 mW, if we estimate it based on the modulator with a power efficiency of 400 µW/Gbps shown in [37]. The power consumption of the laser is 2.15 mW if the wall plug power efficiency is 20%. The power consumption of the electrical part of the transmitter (without output driver) is 6.88 mW, if our estimation assumes that the power consumption scales linearly with the data rate and based on the transmitter shown in [36], which works at 6.25 Gbps and consumes 5.4 mW. The transceiver total power consumption is then 26.79 mW. The energy efficiency is then 2.68 pJ/bit. In contrast, the electrical network consumes 10 pJ/bit to send messages within a rack or 80 pJ/bit to send messages across a rack [38], where a rack refers to a group of processors, equivalent to an SCN in the G-AGCNet. Figure 14 shows the estimation of the energy consumed in completing one barrier synchronization by each processor for the G-AGCNet. It is interesting to see that the energy consumption per processor is primarily determined by the number of layers of the AGCNet-II structure adopted by the system; adding additional layers following the AGCNet-I structure may even slightly decrease the per-processor energy consumption. The reason is that, when adding additional layers following the AGCNet-I structure, fewer processors are contained in one CN, and one set of local all-to-all communications (that happen multiple times in a barrier synchronization if the network has multiple layers following the AGCNet-II structure) involves fewer processors, although more communications among remote processors are involved. Figure 14 also shows that the energy consumption per processor is proportional to the size of AWGRs used, since one-to-all and all-to-all communications based on larger-size AWGRs involve more message transmissions, although more communications can occur concurrently. Figures 13 and 14 show that the latency to complete a global barrier synchronization can be smaller at the expense that each processor consumes more energy.

VII. CONCLUSION

This paper proposes the G-AGCNet, which interconnects $k \times k$ AWGR devices to realize low-latency barrier synchronization that is scalable to a million processors or more. The G-AGCNet can support the efficient one-to-all and all-to-one communication that is not only needed in barrier synchronization, but is also required in other parallel computing primitives, such as the reduction as shown in the paper. Wavelength parallelism and wavelength routing allow the G-AGCNet to realize connectivity as a multi-ary multi-flat Flattened Butterfly network while the wiring overhead is similar to a 2-ary multi-flat Flattened Butterfly network. The paper proposes inter-CN connections that can eliminate the use of fixed wavelength converters and increase the scalability of the network. Analyses indicate that completing one barrier synchronization for one million processors takes only hundreds of nanoseconds and that the proposed optical interconnect is more power efficient than the electrical counterpart.

ACKNOWLEDGMENT

This work was supported in part by the US Department of Defense (DoD) under contract H88230-08-C-0202.

REFERENCES

100-channel × 10-GHz AWG and Michelson interferometers for 1-THz-bandwidth optical arbitrary waveform generation,” in Optical Fiber Communication Conf. (OFC), 2010, OThS1.


