Computer Communications xxx (2015) xxx–xxx

Contents lists available at ScienceDirect

Computer Communications

journal homepage: www.elsevier.com/locate/comcom

FleCube: A flexibly-connected architecture of data center networks on multi-port servers

Deshun Li*, Yanming Shen, Keqiu Li

School of Computer Science and Technology, Dalian University of Technology, No. 2, Linggong Road, Dalian 116024, China

ARTICLE INFO

Article history:
Received 12 May 2015
Revised 6 September 2015
Accepted 24 September 2015
Available online xxx

Keywords:
Multi-port server
Data center networks
Multi-path routing

ABSTRACT

Underlying network provides infrastructures for cloud computing in data centers. The server-centric architectures integrate network and compute, which place routing intelligence on servers. However, the existing multi-port server based architectures suffer from determined scale and large path length. In this paper, we propose FleCube, a flexibly-connected architecture on multi-port servers without using any switches. FleCube is recursively constructed on division of multiple ports in a server by means of complete graph. FleCube benefits data center networks by flexible scale and low diameter, as well as large bisection width and small bottleneck degree. Furthermore, we develop multi-path routing (MPR) to take advantage of parallel paths between any two servers. MPR adopts random forwarding to distribute traffic load and relieve network congestion. Analysis and comparisons with existing architectures show the advantages of FleCube. Evaluations under different degrees of network traffic demonstrate the merits of FleCube and the proposed routings.

1. Introduction

Underlying architecture of data center networks (DCNs) provides infrastructures for cloud computing applications, such as web search, email and on-line gaming, as well as infrastructural services, such as GFS [10], HDFS [4], and BigTable [5]. The topologies of existing DCNs architectures fall into switch-centric and server-centric architectures [33]. Fat-tree [2], VL2 [11], Portland [26], Jellyfish [28], S2 [32], Scafida [17], Poincaré [8], and SWDC[27] belong to the former category, in which servers are attachments of switches fabric. DCell [14], BCube [13], CamCube [1] [7], FiConn [21], HCN&BCN [15], DPillar [24], SWCube&SWKautz [22], and FSquare [23] fall into server-centric category, in which servers undertake the task of processing and forwarding data. According to the usage of switch, server-centric architectures fall into two categories: with and without using switches. Without switches, the directly-connected architectures thoroughly place routing intelligence on servers. Recent research shows that, based on hardware forwarding, the performance of multiple ports in servers is close to that in commodity switches [25]. With the enhancement of forwarding function of servers, multi-port servers will be universally deployed in future DCNs [14] [13] [1]. This paper studies the architecture of DCNs without using any switches.

The existing directly-connected architecture, CamCube [1] [7] is constructed from a 3D torus topology by replacing nodes with 6-port servers. CamCube integrates the overlay and underlying network, and can provide coordinate-based API to send/receive packets with fault-tolerance. However, based on 6-port servers, CamCube suffers from coarse design space. CamCube can only be built at sizes of n^3, which is corresponding to n-ary 3-cube. Furthermore, CamCube suffers from a large path length in large-scale data centers, e.g., in a 20-ary 3-cube with 8,000 servers, the largest path length is 30 and the average path length is 15. To overcome the above deficiencies, we propose a novel directly-connected architecture on multi-port servers.

In this paper, we propose FleCube, a flexibly-connected architecture for interconnecting multi-port servers, without using any switches or routers. FleCube is recursively constructed on division of the multiple ports of servers, in which a high-level FleCube is built from low-level FleCubes by means of complete graph. In spite of the flexibly-connected structure, FleCube demonstrates various advantages in DCNs design. FleCube enjoys the flexible structure on the given number of ports in a server, and the number of servers in FleCube grows double-exponentially with the length of division. For example, given the port number of 12 in a server, FleCube on division {4, 4, 4} and {3, 3, 3, 3} accommodates 44205 and 0.2 billion servers, respectively. FleCube provides a large number of parallel paths and large bisection width, which can distribute traffic load and relieve congestion on network. Network diameter and bottleneck degree are small in FleCube, which can improve network efficiency and fault tolerance. For example, level-3 FleCube with tens of thousands servers has a diameter of 7, and level-4 FleCube with hundreds of millions of servers has a diameter of 15. To take advantage of parallel paths, we propose multi-path routing (MPR), which adopts a random

* Corresponding author. Tel.: +8615840831983.
E-mail address: lideshunlily@qq.com (D. Li).

http://dx.doi.org/10.1016/j.comcom.2015.09.023
0140-3664/© 2015 Published by Elsevier B.V.

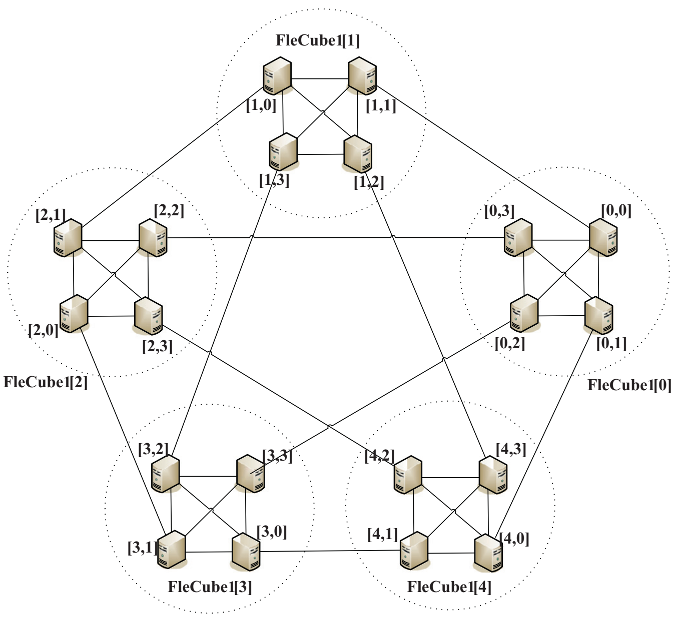

Fig. 1. $FleCube_2$ on division {3, 1} of 4-port server.

forwarding mechanism to distribute traffic load and relieve network congestion. In MPR, flows are confined in a small range of networks and have no effect on higher level. We conduct simulations on FleCubes under different degrees of traffic to evaluate the performance of FleCube and MPR. Simulations reveal the performance of FleCube in average length of routing path and the capability of MPR in spreading out network congestion.

The rest of paper is organized as follows. Section 2 describes physical structure and properties of FleCube. Section 3 presents multipath routing. Section 4 gives comparisons. Section 5 evaluates FleCube and routing, and Section 6 concludes our work.

2. FleCube

In this section, we present the physical structure and the properties of FleCube.

2.1. Physical structure

FleCube uses servers equipped with multiple ports to construct its architecture. In FleCube, a multi-port server is directly connected to other servers via bidirectional communication links, without using any switches or routers.

2.1.1. Construction

FleCube is a recursively defined architecture on division of the multiple ports of servers. If each lower-level FleCube is treated as a virtual node, a higher-level FleCube is constructed from the lower-level FleCubes by means of complete graph.

For clarity, let n denote the number of ports in a multi-port server. Let permutation $\{k_1, k_2, \ldots, k_r\}$ denote a division of n, with length of r and k_i ($k_i \geq 1$) ports in group i ($i \in [1, r]$). Let $FleCube_i$ denote a level-i FleCube, and f_i denote the number of $FleCube_{i-1}$s in a $FleCube_i$. Let $FleCube_{i-1}[j]$ ($j \in [0, f_i)$) denote the jth $FleCube_{i-1}$ in a $FleCube_i$. A FleCube constructed on $\{k_1, k_2, \ldots, k_r\}$ is also referred as FleCube$\{k_1, k_2, \ldots, k_r\}$. In Fig. 1, we take $n = 4$ with division {3, 1} as an example to illustrate the construction of FleCube.

A level-1 FleCube is constructed using $k_1 + 1$ n-port servers by means of complete graph via ports in group 1. As shown in Fig. 1, there are 4 servers connected to each other into a complete graph in each $FleCube_1$. Assuming severs in $FleCube_1$ are arranged in a logical cycle. Each server is assigned an identifier a_1 in a clockwise direction, taking a value from $[0, k_i + 1)$. The link between two servers in $FleCube_1$ is referred as the level-1 link. In $FleCube_1$, we define $f_1 = k_1 + 1$. A level-2 FleCube is constructed using $f_2 = k_2 \cdot s_1 + 1$ $FleCube_1$s via ports in group 2. In a $FleCube_2$, all $FleCube_1$s are connected by means of complete graph, if each of them is treated as a virtual node. In Fig. 1, there are 5 $FleCube_1$s interconnected into a complete graph in the $FleCube_2$. Assuming $FleCube_1$s are arranged in a logical cycle. Each $FleCube_1$ is assigned an identifier a_2 in a clockwise direction, taking a value from $[0, f_2)$. For a higher-level $FleCube_i$, it is constructed in the same way as above. The procedure of building a $FleCube_i$ from $FleCube_{i-1}$s is shown in Algorithm 1. The link between $FleCube_{i-1}$s in a $FleCube_i$ is referred as a level-i link.

Algorithm 1 Build $FleCube_i$.

```
/* f_i: number of FleCube_{i-1}s in a FleCube_i;*/
for (int j = 0; j < f_i; j++) do
    for (int k = j + 1; k < f_i; k++) do
        connect FleCube_{i-1}[j] to FleCube_{i-1}[k];
    end for
end for
```

Let s_i denote the number of servers in a $FleCube_i$. To construct a $FleCube_i$ on $FleCube_{i-1}$s with k_i ports of each server, the relationship between f_i and s_{i-1} satisfies $f_i = k_i \cdot s_{i-1} + 1$.

Notice that each server in $FleCube_1$ is assigned an identifier a_1 and each $FleCube_{i-1}$ in $FleCube_i$ is assigned an identifier a_i, $i \in [2, r]$. We assign each server in $FleCube_r$ a r-tuple in form of $[a_r, a_{r-1}, \ldots, a_2, a_1]$, $a_i \in [0, f_i)$ for $i \in [1, r]$. In this tuple, a_1 indicates the index of a server in $FleCube_1$ where it is located, and a_i for $i \in [2, r]$ indicates the index of $FleCube_{i-1}$ in $FleCube_i$ where the server is located.

2.1.2. Routing path

Note that there is a level-i link between any two $FleCube_{i-1}$s in a $FleCube_i$. This link is adopted as routing path between any two servers among this two $FleCube_{i-1}$s. As any two servers are connected with a level-1 link in a $FleCube_1$, this link services as routing path between this two servers.

Let *src* and *dst* denote the source and destination servers, respectively. Assuming that they are in the same $FleCube_i$ and different $FleCube_{i-1}$s. Let $FleCube^{src}_{i-1}$ and $FleCube^{dst}_{i-1}$ denote the two $FleCube_{i-1}$s where *src* and *dst* locate, respectively. Let $(s1, s2)$ denote the level-i link between $FleCube^{src}_{i-1}$ and $FleCube^{dst}_{i-1}$, where $s1$ and $s2$ locate in $FleCube^{src}_{i-1}$ and $FleCube^{dst}_{i-1}$ respectively. Algorithm 2

Algorithm 2 *Path*(*src*, *dst*).

```
/* src: source server;
   dst: destination server;
   src locates in FleCube^{src}_{i-1} of FleCube_i;
   dst locates in FleCube^{dst}_{i-1} of FleCube_i;
   (s1, s2): link between FleCube^{src}_{i-1} and FleCube^{dst}_{i-1}; */
1: if src and dst are adjacent then
2:     return (src, dst);
3: end if
4: obtain (s1, s2) between FleCube^{src}_{i-1} and FleCube^{dst}_{i-1};
5: path1 = Path(src, s1);
6: path2 = Path(s2, dst);
7: return path1+(s1, s2)+path2;
```

shows the procedure of path generation from *src* to *dst*. It first checks whether *src* and *dst* are adjacent. If so, it returns the link between them (lines 1–3). If not, it gets the level-i link $(s1, s2)$ between $FleCube^{src}_{i-1}$ and $FleCube^{dst}_{i-1}$ (line 4). Then the whole routing path is divided into three parts: *src* to $s1$, $(s1, s2)$, and $s2$ to *dst* (lines 5–6).

Please cite this article as: D. Li et al., FleCube: A flexibly-connected architecture of data center networks on multi-port servers, Computer Communications (2015), http://dx.doi.org/10.1016/j.comcom.2015.09.023

The complete path from *src* to *dst* is (*src*, *s*1)+(*s*1, *s*2)+(*s*2, *dst*) (line 7). Take Fig. 1 as an example, let [1, 0] and [3, 1] denote the source and destination servers, respectively. They are in the same $FleCube_2$ and different $FleCube_1$s. The algorithm first gets ([1, 3], [3, 2]) between $FleCube_1[1]$ and $FleCube_1[3]$. Then it gets ([1, 0], [1, 3]) and ([3, 2], [3, 1]) in $FleCube_1[1]$ and $FleCube_1[3]$ independently. The complete path between [1, 0] and [3, 1] is ([1, 0], [1, 3], [3, 2], [3, 1]).

The routing path generated by Algorithm 2 follows a divide-and-conquer principle, which takes advantage of the recursive characteristic of FleCube. We refer this routing as divide-and-conquer routing (DCR).

2.2. Properties of FleCube

FleCube is constructed on the flexible division of multiple ports in terms of complete graph, which provides it several nice properties for data center networks. In this section, we study the topological and routing properties, which serve as the foundation of performance for FleCube. We first present the scalability, flexibility, and parallel paths in FleCube. Then we study the diameter, bottleneck degree, and bisection width of FleCube. Finally, we study how many servers can be accommodated in a FleCube given the network diameter and the number of ports in a server.

Theorem 1. *The number of servers, s_r, scales double-exponentially with levels r in FleCube.*

Proof. Note that $s_i = f_i \cdot s_{i-1}$ and $f_i = k_i \cdot s_{i-1} + 1$, thus, we have $s_i = k_i \cdot s_{i-1}^2 + s_{i-1}$ for $i > 1$. For the division $\{k_1, k_2, \ldots, k_r\}$ with $k_i \geq 1$ ($i \in [1, r]$), we have $s_i \geq s_{i-1}^2 + s_{i-1} = (s_{i-1} + \frac{1}{2})^2 - \frac{1}{4} > (s_{i-1} + \frac{1}{2})^2 - \frac{1}{2}$. Thus, we have $s_i + \frac{1}{2} > (s_{i-1} + \frac{1}{2})^2 > (s_1 + \frac{1}{2})^{2^{i-1}}$ for $i > 1$. Note that $s_1 = k_1 + 1$, therefore, we have $s_i > (k_1 + \frac{3}{2})^{2^{i-1}} - \frac{1}{2}$ for $i > 1$. So we have $s_r > (k_1 + \frac{3}{2})^{2^{r-1}} - \frac{1}{2}$. Therefore, s_r scales double-exponentially with levels *r* in FleCube. □

Theorem 1 shows that FleCube has an excellent scalability for large-scale data center networks. For example, with $n = 12$, $FleCube_3$ defined on division {4, 4, 4} accommodates 44205 servers, and $FleCube_4$ on {3, 3, 3, 3} accommodates as many as 0.2 billion servers.

A concerned question is how many FleCubes can be created, given the number of ports *n* in a server. The quantity of FleCubes can be measured by the number of permutations of $\{k_1, k_2, \ldots, k_r\}$, with $\sum_{i=1}^{r} k_i = n$ and $k_i \geq 1$. Let g_n denote the number of permutations, then we have the following theorem on g_n,

Theorem 2. $g_n = 2^{n-1}$, *where n is the number of ports in a server.*

Proof. For an *n*-element set in a line, there are $n - 1$ interspaces among these elements. We insert $r - 1$ clapboards into these $n - 1$ interspaces, with no more than one clapboard in each interspace. Then, *n* elements are divided into *r* segments. The number of elements in segment *i* ($i \in [1, r]$) is treated as k_i in corresponding $FleCube_r$. There are $\binom{n-1}{r-1}$ kinds of inserting clapboards for a *r*-division. As *r* taking a value from 1 to *n*, the total number of permutations is $\sum_{r=1}^{n} \binom{n-1}{r-1}$. It is the sum of binomial coefficients in the polynomial expansion of $(1 + x)^{n-1}$. Let $x = 1$, we get $g_n = 2^{n-1}$. □

Theorem 2 shows there are 2^{n-1} FleCubes that can be created, if *n* is given. For example, with $n = 12$, there are 2048 FleCubes can be constructed, and with $n = 24$, this number is about 8 million. The flexibility of FleCube allows it to meet variable scale of DCNs.

Edge-disjoint parallel paths provide redundant paths for routing, which can be used to reduce load on a single link. Let P_r denote the number of edge-disjoint parallel paths between any two servers in $FleCube_r$. We have Theorem 3 on P_r:

Theorem 3. $P_r = n$, *where n is the number of ports in a server.*

Proof. Consider servers *a* and *b*. Each of them has a set of adjacent servers, denoted by Set_a and Set_b, respectively. For the order of both sets, we have $|Set_a| = |Set_b| = n$. The relationships of elements between Set_a and Set_b fall into 3 categories. (1) If *a* and *b* are adjacent, we have $a \in Set_b$ and $b \in Set_a$. In this case, there is a link between *a* and *b*. (2) If *a* and *b* share neighbor(s), the number of neighbors is $|Set_a \bigcap Set_b|$. In this case, there is a path between *a* and *b* via each of the common neighbors. (3) Apart from (1) and (2), each of the left servers in Set_a belongs to or connects to a $FleCube_{r-1}$, which is different with others. It is similar in Set_b. No matter these $FleCube_{r-1}$(s) are shared or not, there are paths between Set_a and Set_b via these $FleCube_{r-1}$(s). This implies there are paths between *a* and *b* via the left servers in Set_a and Set_b. There is no intersection among (1), (2), and (3), and the union of them covers Set_a and Set_b. Thus, the number of edge-disjoint parallel paths between any two servers is the order of the adjacent set, i.e., $P_r = n$. □

Theorem 3 shows that, each port of a source server provides an independent path with others to the destination server. For *n*-port server, FleCube provides *n* parallel paths between any two servers.

Diameter is the maximum path length between any two servers. Let D_r denote the diameter of a $FleCube_r$. We have Theorem 4 on D_r,

Theorem 4. $D_r <= 2^r - 1$, *where r is the number of levels in $FleCube_r$.*

Proof. Consider a routing path of DCR. Note that, level-*i* link (*s*1, *s*2) recursively divides a $FleCube_i$ into two independent parts. Let R_i denote the times of recursions. R_1 is 1 in a $FleCube_1$. In the worst case, $R_i = 2R_{i-1} + 1$ for $i \in [2,r]$, where the recursion occurs level by level. This leads the maximum path length between any two servers in a $FleCube_i$, with $R_i = 2^i - 1$. Thus, we have $D_r <= 2^r - 1$ in $FleCube_r$. □

Theorem 4 shows that the diameter of FleCube grows exponentially with the number of levels. Considering the double-exponential scalability of the total number of server, diameter is small in FleCube. For example, the diameter is 15 in $FleCube_4$ with hundreds of millions of servers. For DCNs with tens of thousands servers, level-3 FleCube with diameter 7 can meet requirements.

Bottleneck degree is the maximum number of flows over a single link under an all-to-all communication. In this model, there is a flow between any two possible servers simultaneously. A small bottleneck degree implies that the communication traffic is spread out over all links. Let p_i denote the number of flows over a level-*i* link under an all-to-all communication. We have Theorem 5 on p_i ($i \in [1, r]$),

Theorem 5. $p_i = 2 \cdot \frac{f_i - 1}{f_i} \cdot \frac{s_r - s_i}{k_i} + s_{i-1}^2$ *for* $i \in [1, r]$.

Proof. For a level-1 link, flows can be divided into two parts: intra- and inter-$FleCube_1$. For the intra part, flow derives from the adjacent servers of this link. The number of flows in this part is 1. For the inter part, we obtain it from the following analysis: A $FleCube_1$ has $s_1(s_r - s_1)$ flows with servers outside this $FleCube_1$. Among them, flows communicating with current server will not travel on level-1 link in this $FleCube_1$, the ratio of which is $\frac{1}{f_1}$; The left $\frac{f_1 - 1}{f_1}$ ratio of flows will evenly travel level-1 links once in this $FleCube_1$. The total number of level-1 links in a $FleCube_1$ is $\frac{k_1 \cdot s_1}{2}$. Thus, each level-1 link carries $\frac{f_1 - 1}{f_1} \cdot \frac{2 \cdot s_1 (s_r - s_1)}{k_1 \cdot s_1}$ flows in the inter part. Together, we have $p_1 = \frac{f_1 - 1}{f_1} \cdot \frac{2 \cdot (s_r - s_1)}{k_1} + 1$.

For a level-2 link, we follow the same analysis as above. The number of flows in intra part is ${s_1}^2$. The number of flows in inter part is $\frac{f_2 - 1}{f_2} \cdot \frac{2 \cdot s_2 (s_r - s_2)}{k_2 \cdot s_2}$. We have $p_2 = \frac{f_2 - 1}{f_2} \cdot \frac{2 (s_r - s_2)}{k_2} + {s_1}^2$. It is similar for a higher level link. Thus, $p_i = 2 \cdot \frac{f_i - 1}{f_i} \cdot \frac{s_r - s_i}{k_i} + {s_{i-1}}^2$ for $i \in [1, r]$. □

Let BoD_r denote the bottleneck degree of a $FleCube_r$. As defined, we have $BoD_r = max\{p_1, \cdot \cdot \cdot, p_r\}$. Characteristic of formula p_i implies that the bottleneck degree is a variable on parameters of permutation, which is conducive to different network requirements.

Please cite this article as: D. Li et al., FleCube: A flexibly-connected architecture of data center networks on multi-port servers, Computer Communications (2015), http://dx.doi.org/10.1016/j.comcom.2015.09.023

Bisection width denotes the minimal bandwidth to be removed to partition a network into two equal parts. A large bisection width implies that the structure provides a high routing performance and fault tolerance for data center networks. Let B_r denote the bisection width of a $FleCube_r$. We have Theorem 6 on the lower boundary of B_r,

Theorem 6. *$B_r > \frac{s_r \cdot k}{8}$, where $k = max\{k_1, k_2, \ldots, k_r\}$.*

Proof. Notice that, $s_i = k_i \cdot s_{i-1}^2 + s_{i-1}$ for $i > 1$. Considering p_i in Theorem 5, we have $p_i = 2 \cdot \frac{f_i - 1}{f_i} \cdot \frac{s_r - s_i}{k_i} + s_{i-1}{}^2 < 2 \cdot \frac{s_r - s_i}{k_i} + s_{i-1}{}^2 = \frac{2 \cdot s_r - 2 \cdot s_i + k_i s_{i-1}{}^2}{k_i} = \frac{2 s_r - s_i - s_{i-1}}{k_i} < \frac{2 s_r}{k_i}$, for $i \in [1, r]$. Let $k = max\{k_1, k_2, \ldots, k_r\}$, thus $p_i < \frac{2 s_r}{k}$. Based on [20], B_r is $\frac{k}{2 s_r}$ times of bisection width in its embedding complete graph. Thus, $B_r > \frac{k}{2 s_r} \cdot \frac{s_r^2}{4} = \frac{s_r \cdot k}{8}$, where $k = max\{k_1, k_2, \ldots, k_r\}$. □

Let N denote the total number of servers in a FleCube. Theorem 6 shows the bisection width of FleCube is $O(N)$. This implies that FleCube intrinsically provides fault tolerance and multi-path routing on top of it.

Another essential issue should be considered is how many servers can be accommodated in a FleCube, given network diameter d and the number of ports n in a server. We have Theorem 7 on this issue,

Theorem 7. *Given network diameter d and the number of port n in a server, the total number of servers N in FleCube satisfies $N > (k_1 + \frac{3}{2})^{\frac{d+1}{2}} - \frac{1}{2}$, where $k_1 < n$.*

Proof. Consider $D_r <= 2^r - 1$ in Theorem 4. In terms of d, we have $2^r - 1 = d$, i.e., $r = \log_2 (d + 1)$. Note that in proof of Theorem 1, $s_r > (k_1 + \frac{3}{2})^{2^{r-1}} - \frac{1}{2}$. Thus, we have $N > (k_1 + \frac{3}{2})^{\frac{d+1}{2}} - \frac{1}{2}$, where $k_1 < n$. □

Actually, Theorem 7 shows a rough lower boundary of N in terms of n and d. The capacity of servers in FleCube is far greater than it. For example, given $n = 10$ and $d = 15$, the maximum capacity is 1.7 hundred million servers with $r = 4$. Consider the double-exponential scalability, $FleCube_3$ is enough for most multi-port servers based architectures. For example, given $n = 10$ and $d = 7$, $FleCube_3$ constructed on division {5, 3, 2} composes of 26,106 servers.

3. Multipath routing

To take advantage of parallel paths, we propose multi-path routing (MPR), which benefits DCNs by relieving network congestion and improving bandwidth utilization.

3.1. Motivation and challenges

Network bandwidth is scarce resource in cloud computing, which results in fierce competitions among applications. Competitions on bandwidth give rise to optimized solutions, such as throughput-delay trade-off [30] [3] [29], bandwidth allocation [19] [31] [16]. However, optimized solutions cannot solve competitions fundamentally. Despite DCR takes advantage of the single path routing in FleCube, it does not consider the large number of parallel paths between any two servers. Multiple paths routing [18] [6] [9] [28] can effectively alleviate the competitions and congestions on a single path, thereby improving network efficiency and reducing latency. To overcome the shortage of DCR, we propose multi-path routing (MPR) in this section.

FleCube benefits DCNs from a large number of parallel paths between any two servers. Specifically, each port of the source server provides an edge-disjoint path with others to the destination server. However, can each path be adopted in the multi-path routing? For example, let [1, 1] and [1, 3] denote the source and destination server in Fig. 1. As we can see, paths ([1, 1], [1, 3]), ([1, 1], [1, 0], [1, 3]), and ([1, 1], [1, 2], [1, 3]) are alternative for multi-path routing. Path ([1, 1], [0, 0], [0, 2], [3, 3], [3, 2], [1, 3]) is not suitable to appear in the multi-path routing from [1,1] to [1,3].

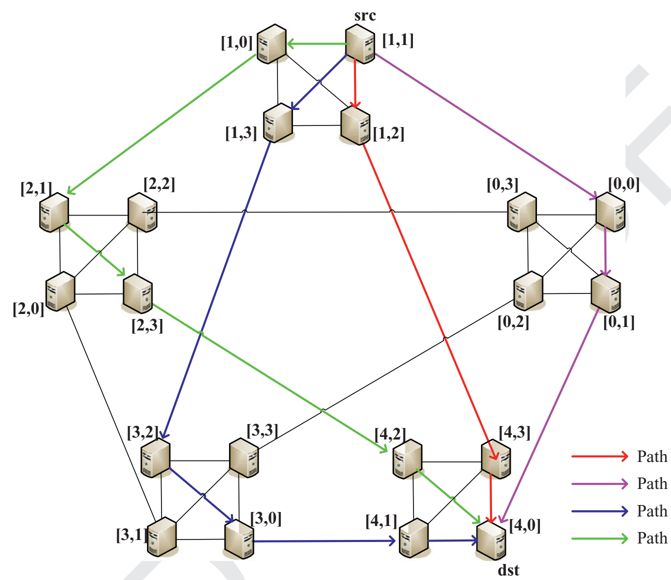

Fig. 2. Primary multi-path routing (PMPR). Paths 1, 2, 3, and 4 denote paths of PMPR from *src* [1,1] to *dst* [4,0].

The large number of parallel paths gives rise to another two challenges: loop routing and across path. In loop routing, a packet will be spread on a circular path until the end of time-to-live (TTL). This will lead to waste of resources in the whole life period of a packet. Across path is a result of duplicate selection of path in a distributed multi-path routing, which partly eliminates the benefits of multi-path routing. Flows will accumulate and cause congestion on the cross path, which increases latency and reduces network utilization.

3.2. Primary multi-path routing

Due to the large server population in data centers, we seek to compute multi-path routing in a distributed manner, relying on local information of the current server. One straightforward solution is that the source server sends flows to its neighbors and these neighbors forward flows to the destination, separately. We refer it as primary multi-path routing (PMPR).

We take examples of a $FleCube_2$ in Fig. 2 to display the multiple paths in PMPR. In example 1, let [1, 1] and [1, 3] denote the source and destination, respectively. There are 3 paths in PMPR from [1, 1] to [1, 3], composing of paths ([1, 1] ,[1, 3]), ([1, 1], [1, 0], [1, 3]), and ([1, 1], [1, 2], [1, 3]). In example 2, let [1, 1] and [4, 0] denote the source and destination, respectively. There are 4 paths in PMPR from [1, 1] to [4, 0], composing of paths ([1, 1], [1, 2], [4, 3], [4, 0]), ([1, 1], [0, 0], [0, 1], [4, 0]), ([1, 1], [1, 3], [3, 2], [3, 0], [4, 1], [4, 0]), and ([1, 1], [1, 0], [2, 1], [2, 3], [4, 2], [4, 0]).

Algorithm 3 shows the procedure of primary multi-path routing (PMRP). Let *src* and *dst* denote the source and destination, respectively. Assuming they are in the same $FleCube_i$ and different $FleCube_{i-1}$s: $FleCube_{i-1}^{src}$ and $FleCube_{i-1}^{dst}$. In part I, *src* randomly sends flows via ports in group 1 to *i* (lines 1–2). Part II shows the process of a current server *cur* forwarding flow. Upon receiving a flow, *cur* checks whether it is the destination. If so, *cur* delivers the flow to the upper layer and returns (lines 3–6). Otherwise, *cur* checks whether it is on *path*(*src*, *dst*) of DCR. If so, *cur* delivers the flow along *path*(*src*, *dst*) (lines 7–9). Else, *cur* checks whether *cur* and *src* are in the same $FleCube_{i-1}$. If so, *cur* randomly forwards the flow from a level-*i* link (lines 11–13). Otherwise, *cur* forwards the flows along *path*(*cur*, *dst*) (lines 15–16).

Please cite this article as: D. Li et al., FleCube: A flexibly-connected architecture of data center networks on multi-port servers, Computer Communications (2015), http://dx.doi.org/10.1016/j.comcom.2015.09.023

Algorithm 3 PMPR.

```
/* src: source server;
dst: destination server;
cur: current server;
src locates in FleCube^src_{i-1} of FleCube_i;
dst locates in FleCube^dst_{i-1} of FleCube_i;
path(src, dst): the path of DCR from src to dst; */
Part_I: /* for source server src */
src send flows via ports in groups [1, i];
return;
Part_II: /* for current server cur*/
if (cur == dst) then
    deliver the flow to upper layer;
    return;
end if
if (cur on path(src, dst)) then
    cur forward the flow along path(cur, dst);
    return;
else
    if (cur and src in the same FleCube_{i-1}) then
        cur randomly forward flow from a level-i link;
        return;
    else
        cur forward the flow along path(cur, dst);
        return;
    end if
end if
```

Algorithm 4 Path generation in MPR.

```
/* src: source server;
dst: destination server;
src locates in FleCube^src_{i-1} of FleCube_i;
dst locates in FleCube^dst_{i-1} of FleCube_i;
u_0: adjacent server of src on path(src, dst);
v_0: adjacent server of dst on path(src, dst);
PMPR(src, dst): a path of PMPR from src to dst; */
1: Set_s = GetAdjacent(src, i);
2: Set_d = GetAdjacent(dst, i);
3: add path(src, dst) into MPR;
4: set u_0 occupied in Set_s;
5: set v_0 occupied in Set_d;
6: while servers are available in Set_s and Set_d do
7:     randomly get u ∈ Set_s;
8:     randomly get v ∈ Set_d;
9:     if (PMPR(src, dst) via u, v is not in MPR) then
10:        add path PMPR(src,dst) via u, v into MPR;
11:        set u occupied in Set_s;
12:        set v occupied in Set_d;
13:    end if
14: end while
```

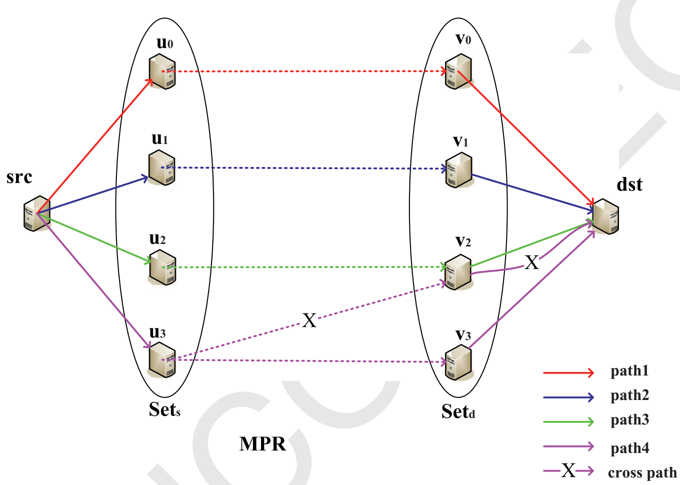

Fig. 3. Multi-path routing (MPR). Set_s and Set_d denote sets of neighbors of *src* and *dst*, respectively. Paths 1, 2, 3, and 4 denote the paths of MPR. The cross path is possible in PMPR.

PMPR relies on the neighbors of *src* to forward flows to *dst*. By passing the third FleCube_{i-1} and path of DCR, PMPR can confine flows in a FleCube_i and avoid loop routing. PMPR achieves its distributed multi-path routing relying on the information of current server.

3.3. Multi-path routing

By passing the third FleCube_{i-1}, PMPR can avoid cross path in the upstream. However, as shown in Fig. 3, PMRP cannot avoid cross paths. It is a result of the random selection of path on the current server. Notice that the proof of Theorem 3 provides a solution to edge-disjoint parallel paths between any two servers. We take these paths as multiple paths in multi-path routing (MPR).

Algorithm 4 shows the generation of parallel paths in MPR. *GetAdjacent*$(\cdot, i)$ returns the set of neighbors of a server from level 1 to level i (lines 1–2). The procedure first adds $path(src, dst)$ into the multiple paths of MPR, and sets the corresponding neighbors, u_0 and v_0, occupied in Set_s and Set_d (lines 3–5). Then the procedure adds the path of PMPR via available neighbors u and v, until all servers in Set_s and Set_d are occupied (lines 6–14).

MPR benefits multi-path routing by confining flows in FleCube_i. Furthermore, MPR has the following properties:

Theorem 8. *The number of paths used in MPR is* $\sum_{j=1}^{i} k_j$, *where i is the index of the lowest level FleCube shared by src and dst.*

The proof can be obtained from Theorem 3. Theorem 8 implies that the relative position of *src* and *dst* determines the number of parallel paths used in MPR. The higher level of FleCube_i, the larger number of parallel paths.

The following theorem shows the maximum length path used in MPR.

Theorem 9. *The maximum length path in MPR is less than* $2^{i-1} + 3$, *where i is the index of the lowest level FleCube shared by src and dst.*

Proof. Algorithms 3 and 4 show that some neighbors of *src* forward flow to the third FleCube_{i-1}s. Then these FleCube_{i-1}s forward flows to *dst* independently. This will lead to the maximum routing path from *src* to *dst*. It is the longest path in a FleCube_i plus two links, from *src* to the FleCube_{i-1} via a neighbor. Thus, we get $2^{i-1} + 3$. □

Contrast to path of DCR, the maximum path length in MPR increases only by 2 links. It is a preference for high-level FleCube_i shared by *src* and *dst*.

4. Comparisons

CamCube [1] [7] and FleCube belong to directly-connected architecture on multi-port servers. Fig. 4 illustrates a CamCube network with 27 6-port servers. For a CamCube with the number of servers N, the diameter is about $\frac{3\sqrt[3]{N}}{2}$, and the bisection is $2N^{\frac{2}{3}}$. Notice that CamCube is constructed on 6-port servers and FleCube can be built on any multi-port servers. In this section, we use FleCube built on 5, 6, 7, and 8-port servers to compare with CamCube built on 6-port servers in various aspects.

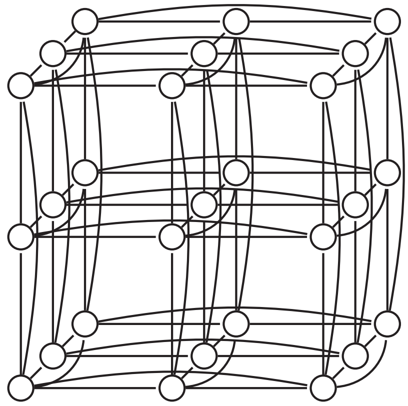

Fig. 4. CamCube architecture based on 3D torus topology.

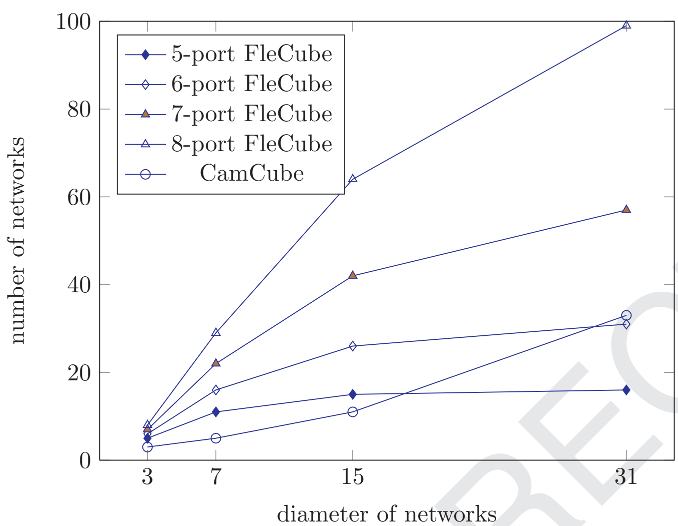

Fig. 5. Cumulative curves of number of networks versus the given diameter.

4.1. Flexibility

Given the number of ports in a server and network diameter, we compare the number of networks in CamCube and FleCube. We choose four typical diameter values for comparison, d =3, 7, 15, 31. Fig. 5 plots the cumulative curves of the number of networks versus the given diameter of networks. As we can see, with the increment of the given diameter, FleCubes with 7-port and 8-port servers accommodate more networks than CamCube. For 6-port servers, FleCube is not worse than CamCube within the scope of the given diameter. For 5-port servers, FleCube can accommodate more networks than CamCube with a given diameter less than 15.

4.2. Scalability

Due to flexibility of ports division, FleCube has a large span of scales. Notice that CamCube is constructed on 6-port servers and FleCube has a double-exponential scalability, we adopt FleCube built on 5,6,and 7-port servers in the comparison of scalability. We use the lower and upper boundary of the number of servers to denote the scalability of FleCube. Fig. 6 plots the boundary versus the given diameter of networks. As we can see, each upper boundary is far greater than that of CamCube. Each lower boundary is also greater than that of CamCube when diameter larger than 7. When the diameter is no more than 7, each lower boundary of FleCube is less than that of CamCube. This implies that FleCube has a large span of capacity to accommodate various demands of scale.

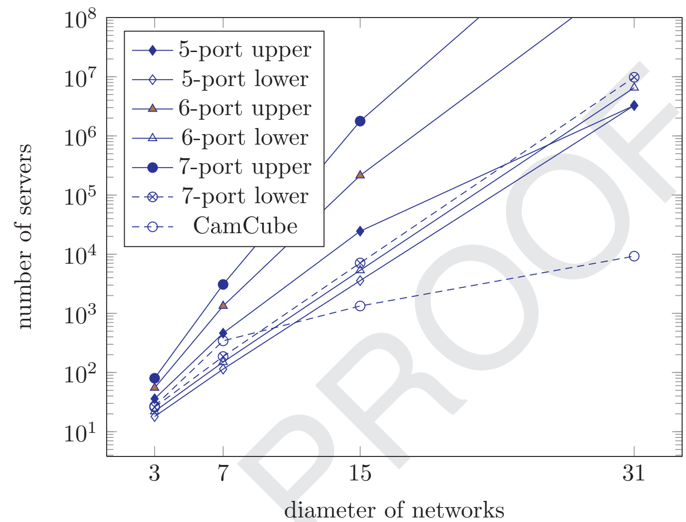

Fig. 6. The scale of networks versus the given diameter. "upper" and "lower" denote the upper boundary and lower boundary of FleCube with given number of ports..

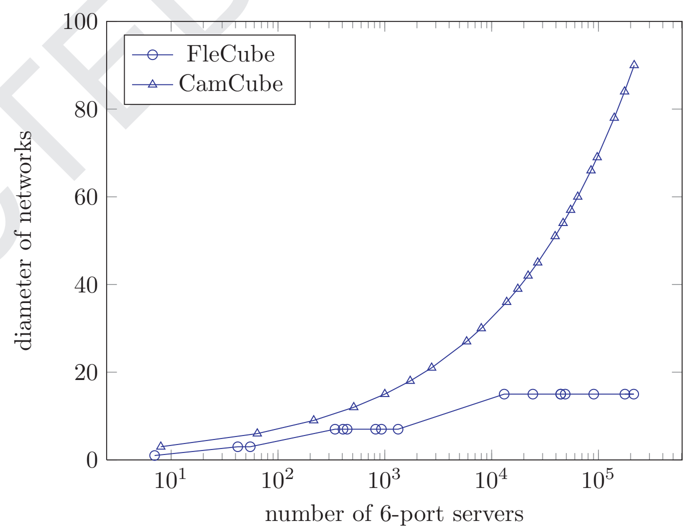

Fig. 7. Diameter of networks versus the number of servers.

4.3. Diameter

Given the total number of servers in DCNs, the diameter is an important measure of network performance. In fairness, we use 6-port servers only in FleCube. Fig. 7 plots the diameter of networks versus the number of servers accommodated by FleCube and CamCube. As we can see that diameter of FleCube grows in small increment with the exponential growth of server number, while the diameter in CamCube grows exponentially under the same condition.

4.4. Bisection width

FleCube is a flexible structure defined on the division of the multiple ports, therefore, different divisions will result in different bisection width with a great span. Due to the complexity of the bisection width and the diversity of divisions, we observe the lower boundary of the bisection width within certain network scales. We choose the lower boundary of bisection width of FleCube to compare with that in CamCube. Fig. 8 plots bisection width versus the number of servers in FleCube and CamCube. As we can see, when the number of servers is larger than 4096 (16-ary CamCube), each bisection width of FleCube is larger than that of CamCube. For the scale less than 4096,

Please cite this article as: D. Li et al., FleCube: A flexibly-connected architecture of data center networks on multi-port servers, Computer Communications (2015), http://dx.doi.org/10.1016/j.comcom.2015.09.023

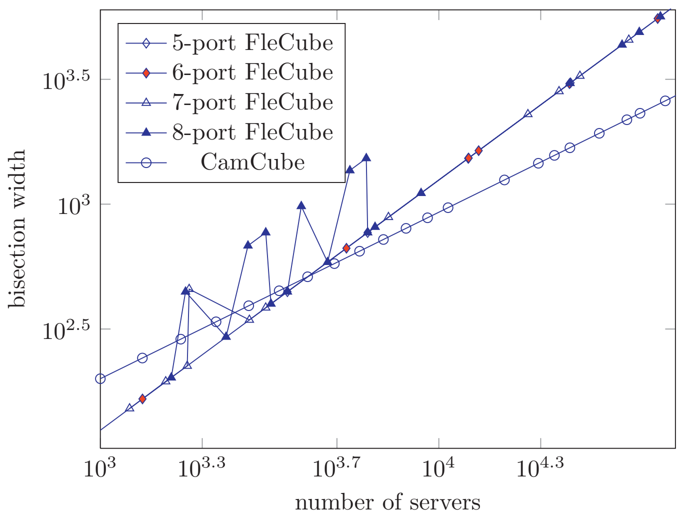

Fig. 8. Bisection width of networks versus the scale of networks.

Table 1
Comparisons with other server-centric architectures.

Structure	Degree	Diameter	BiW	BoD	Switches	Wires
DCell	n	$2^{l+1}-1$	$O(N/\log N)$	$N\log N$	N/m	$(n+1)N/2$
BCube	n	n	$O(N)$	$O(N)$	nN/m	nN
FiConn	2	$2^{l+1}-1$	$O(N/\log N)$	$N\log N$	N/m	$(3-\frac{1}{2^l})N/2$
FleCube	n	2^l-1	$O(N)$	$O(N)$	-	$nN/2$

bisection width of FleCube is less than that of CamCube, with several opposite cases. Due to the diverse division of multiple ports and the discrete samples, bisection width in FleCube shows fluctuation characteristics. However, the linear lower boundary of bisection width is a common lower boundary in FleCube, which is larger than that in CamCube.

4.5. With other architectures

The directly-connected architectures integrate networking in the multi-port servers, while architectures connected through switches rely on switches to forward data. According to [23] [12] [25], forwarding capacity on multi-port server in hardware is close to that of COTS switches per port, as mentioned in [22] with c=1 in normalized switch delay. Servers in architecture connected through switches are equipped with small number of ports, of which the typical value is 1, 2, 3, and 4; while servers in directly-connected architectures are equipped with more ports. Since servers can send data from each port, the more number of ports in a server benefits architecture by more multi-path routing paths from the source node.

In the multi-port server based architectures, DCell, FiConn, and FleCube have a double-exponential scalability of server population on the number of levels, which is more aggressive than BCube. The capacity of DCell, BCube, and FiConn is determined by the number of levels and switch ports, while it is the division of multiple ports of servers in FleCube, which provides a large flexibility. Given the number of ports n in a server and the total number of servers N, Table 1 shows comparisons of FleCube with other state-of-the-art server-centric architectures. “BiW” and “BoD” denote bisection width and bottleneck degree, respectively. Diameter is measured in terms of “server-to-server-direct” in FleCube, and it is “server-to-server-via-a-switch” in other architectures. l denotes the number of levels in DCell, FiConn and FleCube. As we can see, BCube and FleCube have the same bisection width and bottleneck degree, which are better than that of DCell and FiConn. Compared with DCell and FiConn, FleCube is a low-diameter network. For modular data centers, BCube has a short diameter of typically 4 in terms of “server-to-server-via-switches”, while it is 3 or 7 in FleCube in terms of “server-to-server”. With the same forwarding capability per port in server and switch, FleCube can provide a good network performance. For large scale data centers, the number of ports in a server in FleCube is typically double or triple of that in DCell or BCube, while there are large number of switches in DCell, BCube, and FiConn. Let m denote the number of ports in a switch. Both DCell and FiConn need N/m m-port switches, BCube needs nN/m m-port switches, and there is no deployment of switches in FleCube. Notice that for the different number of ports in a server, FleCube needs the same or a larger number of wires than other architectures. Take the multiple ports in a server, switches, and wires into consideration, FleCube does not introduce excessive cost in the construction of networks.

5. Evaluation

To evaluate the structure of FleCube and the performance of proposed routing algorithms, we conduct simulations on FleCube under different degrees of flows pressure. Notice that our simulations focus on the performance of network topology, instead of the routing algorithm itself.

5.1. DCR

We design time-step based routing simulations with congestion on servers. In simulations, flows randomly generated are imposed on servers at time slot 0. Specifically, we assume that each server can send out at most one flow in a time slot. Passing flow(s) will be sent to forwarding queue, and flows reaching destination will not be considered in the next time slot. If more than one flow in the forwarding queue, first-in-first-out (FIFO) scheme is adopted. For flows arriving forwarding server simultaneously, a randomly selected flow will be forwarded first. We assume that the flows are short enough, which can be forwarded completely within a time slot. Thus, in each time slot, only one flow in a server will be sent to its next hop, and others should be delayed. Under different traffic degree, we evaluate the average path length and the number of flows on an active server. “DWC” and “DWoC” represent DCR with congestion and without congestion, respectively. For each result, the statistical data is an average of 100 sets of generated flows.

5.1.1. DCR on FleCube$_2$

We conduct simulations of DCR on FleCube{8, 16} with 1305 servers , in which each server is equipped with 24 ports. We vary the number of flows from 100 to 1100, with a step size of 100.

Fig. 9 shows the average delay versus the number of flows with and without congestion. As we can see, when the number of flows is small, the average delay with congestion is almost the same as that without congestion. For the case of 100 flows, the average delay with congestion is only 3.37% greater than that of without congestion. As the number of flows increases from 100 to 1000, this proportion increases to 39.7%. It is a slightly linear growth when 1100 flows are initiated for 1305 servers at the same time.

Fig. 10 shows the average number of flows in an active server versus time slot. As we can see, the number of flows reaches the maximum at time slot 2. With congestion, each active server has only 1.42, 1.47, 1.52 flows at peak instant for 900, 1000, and 1100 flows, respectively. After time slot 3, the number of flows in an active server decreases drastically.

5.1.2. DCR on FleCube$_3$

We conduct simulations of DCR on FleCube{4, 4, 4} with 44205 servers, in which each server is equipped with 12 ports. We

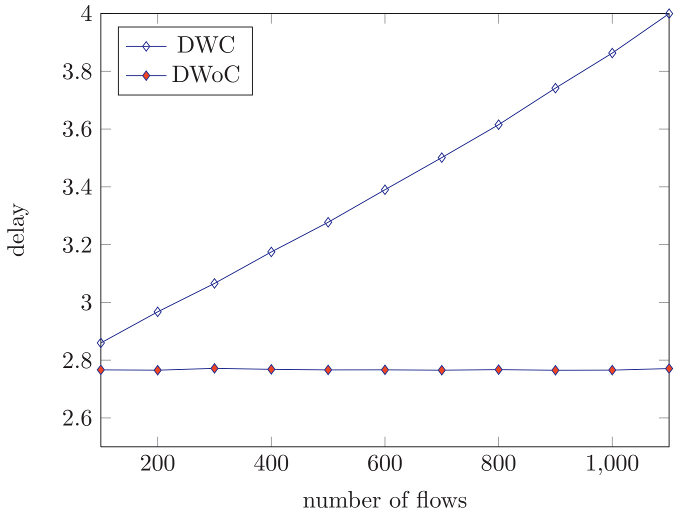

Fig. 9. Average delay versus the number of flows on FleCube{8, 16}.

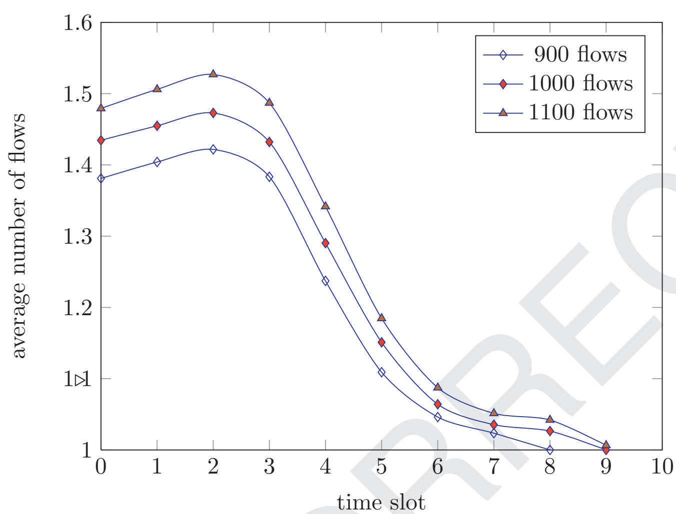

Fig. 10. Average number of flows in an active server versus time slot on FleCube{8, 16} with congestion.

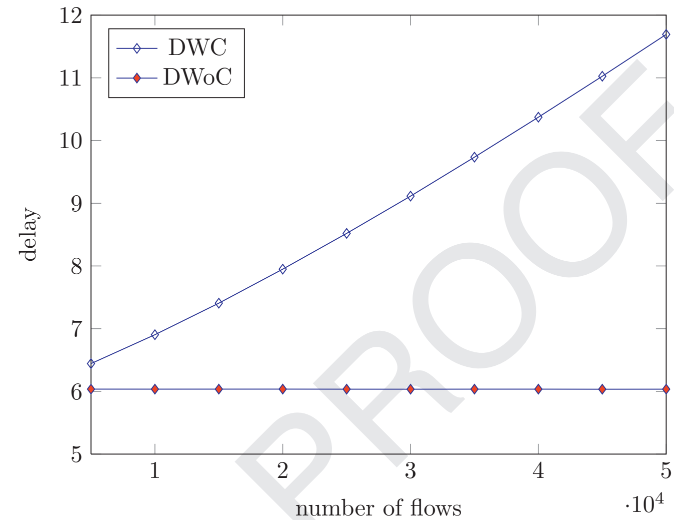

Fig. 11. Average delay versus the number of flows on FleCube{4, 4, 4}.

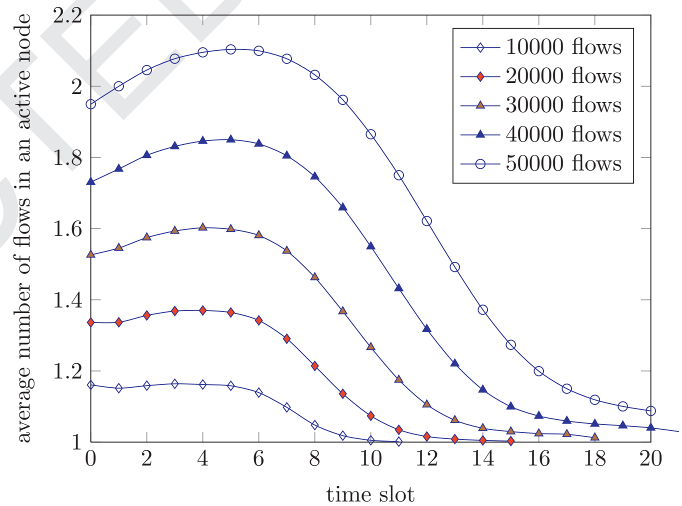

Fig. 12. Average number of flows in an active server versus time slot on FleCube{4, 4, 4} with congestion.

vary the number of flows from 5000 to 50,000, with a step size of 5000.

Fig. 11 shows the average delay versus the number of flows with and without congestion. When the number of flows is small, average delay with congestion is near that without congestion. For 5000 flows, the average delay with congestion is 6.74% greater than that of without congestion. As the number of flows increases from 5000 to 50,000, this proportion increases to 93.7%. It is about 13.9 times of that in 5000 flows. Notice that there are 50,000 flows and 44,205 servers, each server works as source and destination for 1.13 times simultaneously.

Fig. 12 shows the average number of flows in an active server versus time slot. As we can see, the average number of flows increases smoothly from time slot 0 to 5. They reach a maximum at time slot 4 or 5. With congestion, each server holds 1.60, 1.85, and 2.10 flows at peak instant for 30,000, 40,000, and 50,000 flows, respectively. After time slot 6, the number decreases drastically.

The result of simulations shows DCR has a good efficiency in data transmission with congestion. For randomly generated flows, DCR demonstrates low latency and high capability of spreading out load. This suggests the good performance of FleCube based networks.

5.2. PMPR and MPR

Different with former simulations, we focus on the performance of PMPR and MPR under burst network traffic. We adopt time-step based simulations with congestion on a link to evaluate PMPR and MPR. Specifically, each server can send out at most one flow from each port in each time slot. For each port, queuing and selection of flows are similar as former simulations. Under burst traffic, we evaluate average path length and the number of flows in an active server in PMPR and MPR.

We conduct simulation on FleCube{8, 16} with 1305 servers. In simulation, we vary the number of source-destination pairs from 100 to 1200, with a step size of 100. For each pair of source and destination, a random number of flows (≤ 10) is initialized in time slot 0. For each number of source-destination pair, the statistical data is an average of 100 sets.

Fig. 13 shows the average path length of flows versus the number of source-destination pairs without congestion. Compared with DCR, PMPR and MPR have a larger routing path. In average, path in PMPR is 0.82 longer than that of DCR, path in MPR is 0.97 longer than that of DCR.

Please cite this article as: D. Li et al., FleCube: A flexibly-connected architecture of data center networks on multi-port servers, Computer Communications (2015), http://dx.doi.org/10.1016/j.comcom.2015.09.023

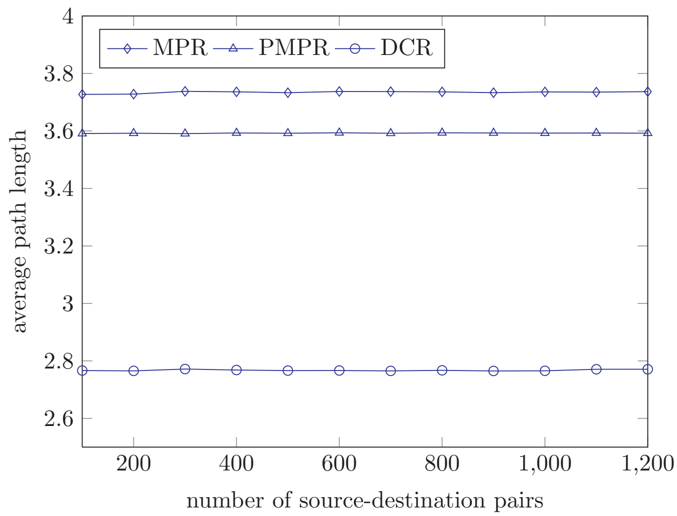

Fig. 13. Average path length versus the number of source-destination pairs without congestion.

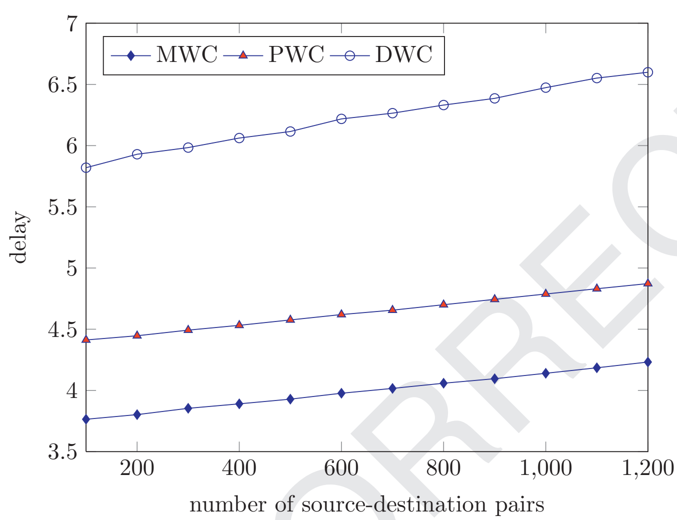

Fig. 14. Average delay of flows versus the number of source-destination pairs with congestion.

Fig. 14 shows the average delay of flows versus the number of source-destination pairs. "PWC" and "MWC" represent PMPR and MPR with congestion, respectively. Compared with DCR, PMPR and MPR achieve a small average delay with congestion under the same degrees of network traffic. For the case of 100 pairs of source-destination, average delay in MWC and PWC is about 64.7% and 75.8% of that in DWC. As the number of source-destination pairs increases, the proportions decrease a little. Comparing MWC with PWC, average delay in MWC is about 86.4% of that in PWC. And proportion is flat with the increment of the number of source-destination pairs.

Fig. 15 shows the average number of flows in an active server versus time slot for 1200 pairs of source-destination on FleCube{8, 16}. As we can see, the number of flows in a server reaches the maximum at time slot 0 for each routing. After time slot 3, the number decreases quickly. In this process, MPR declines the fastest and finishes first, DCR suffers from congestion. Performance of PMPR locates between MPR and DCR.

From above simulations, we can see that both PMPR and MPR can handle burst network traffic with congestion. The average path length in MPR is larger than that of PMPR, and MPR has less congestion

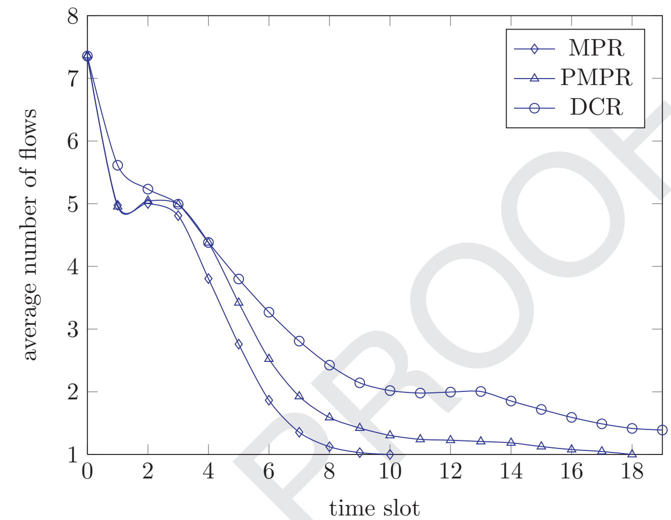

Fig. 15. Number of flows on a server versus the time slot for 1200 pairs of source-destination with congestion.

than PMPR. Compared with DCR, MPR and PMPR has better ability in spreading out burst flows on FleCube.

6. Conclusion

In this paper, we propose FleCube, a flexibly-connected architecture for interconnecting multi-port servers. FleCube is recursively constructed on the division of multiple ports of servers. It is highly flexible and scalable to accommodate hundreds of thousands of servers with low path length and large bisection bandwidth. The multi-path routing on FleCube takes advantage of parallel paths between any two servers in FleCube. Results of comparisons and simulations demonstrate the good performance of FleCube and our proposed routings under different degrees of network traffic.

Acknowledgment

This work was supported by the National Science Foundation for Distinguished Young Scholars of China (grant no. 61225010).

References

[1] H. Abu-Libdeh, P. Costa, A. Rowstron, G. O'Shea, A. Donnelly, Symbiotic routing in future data centers, in: Proceedings of ACM SIGCOMM'10, 2010.
[2] M. Al-Fares, A. Loukissas, A. Vahdat, A scalable, commodity data center network architecture, in: Proceedings of ACM SIGCOMM'08, 2008.
[3] M. Alizadeh, A. Greenberg, D.A. Maltz, J. Padhye, P. Patel, B. Prabhakar, S. Sengupta, M. Sridharan, Data center TCP (DCTCP), in: Proceedings of ACM SIGCOMM'11, 2011.
[4] D. Borthakur, The hadoop distributed file system: architecture and design, http://hadoop.apache.org/docs/r0.18.0/hdfs_design.pdf.
[5] F. Chang, J. Dean, S. Ghemawat, W.C. Hsieh, D.A. Wallach, M. Burrows, T. Chandra, A. Fikes, R.E. Gruber, Bigtable: a distributed storage system for structured data, ACM Trans. Comput. Syst. (TOCS) 26 (2) (2008) 4.
[6] M. Chiesa, G. Kindler, M. Schapira, Traffic engineering with equal-cost-multi-path: an algorithmic perspective, in: Proceedings of IEEE INFOCOM'14, 2014.
[7] P. Costa, A. Donnelly, G. Oĩ shea, A. Rowstron, CamCube: A Key-Based Data Center, Technical Report MSR TR-2010-74, Microsoft Research, 2010.
[8] M. Csernai, A. Gulyás, A. Kőrösi, B. Sonkoly, G. Biczók, Incrementally upgradable data center architecture using hyperbolic tessellations, Comput. Netw. 57 (6) (2013) 1373–1393.
[9] D. Eppstein, Finding the k shortest paths, SIAM J. Comput. 28 (2) (1998) 652–673.
[10] S. Ghemawat, H. Gobioff, S.-T. Leung, The google file system, in: Proceedings of ACM SIGOPS'03, 2003.
[11] A. Greenberg, J.R. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D.A. Maltz, P. Patel, S. Sengupta, Vl2: a scalable and flexible data center network, in: Proceedings of ACM SIGCOMM'09, 2009.
[12] A. Greenberg, D.A. Maltz, What goes into a data center? Microsoft Res. (2009).
[13] C. Guo, G. Lu, D. Li, H. Wu, X. Zhang, Y. Shi, C. Tian, Y. Zhang, S. Lu, Bcube: a high performance, server-centric network architecture for modular data centers, in: Proceedings of ACM SIGCOMM'09, 2009.

Please cite this article as: D. Li et al., FleCube: A flexibly-connected architecture of data center networks on multi-port servers, Computer Communications (2015), http://dx.doi.org/10.1016/j.comcom.2015.09.023

[14] C. Guo, H. Wu, K. Tan, L. Shi, Y. Zhang, S. Lu, Dcell: a scalable and fault-tolerant network structure for data centers, in: Proceedings of ACM SIGCOMM′08, 2008.
[15] D. Guo, T. Chen, D. Li, M. Li, Y. Liu, G. Chen, Expandable and cost-effective network structures for data centers using dual-port servers, IEEE Trans. Comput. (TC) 62 (7) (2013) 1303–1317.
[16] J. Guo, F. Liu, X. Huang, J.C. Lui, M. Hu, Q. Gao, H. Jin, On efficient bandwidth allocation for traffic variability in datacenters, in: Proceedings of IEEE INFOCOM′14, 2014.
[17] L. Gyarmati, T.A. Trinh, Scafida: A scale-free network inspired data center architecture, ACM SIGCOMM Comput. Commun. Rev. 40 (5) (2010) 4–12.
[18] C.E. Hopps, Analysis of an equal-cost multi-path algorithm, http://tools.ietf.org/html/rfc2992.
[19] J. Lee, Y. Turner, M. Lee, L. Popa, S. Banerjee, J.-M. Kang, P. Sharma, Application-driven bandwidth guarantees in datacenters, in: Proceedings of ACM SIGCOMM′11, 2014.
[20] F. Leighton, Introduction to Parallel Algorithms and Architectures: Arrays, Trees, Hypercubes., Morgan Kauffmann, San Mateo, CA, 1992.
[21] D. Li, C. Guo, H. Wu, K. Tan, Y. Zhang, S. Lu, Ficonn: using backup port for server interconnection in data centers, in: Proceedings of IEEE INFOCOM′09, 2009.
[22] D. Li, J. Wu, On the design and analysis of data center network architectures for interconnecting dual-port servers, in: Proceedings of IEEE INFOCOM′14, 2014.
[23] D. Li, J. Wu, Z. Liu, F. Zhang, Dual-centric data center network architectures, in: Proceedings of IEEE ICPP′15, 2015.
[24] Y. Liao, J. Yin, D. Yin, L. Gao, Dpillar: dual-port server interconnection network for large scale data centers, Comput. Netw. 56 (8) (2012) 2132–2147.
[25] G. Lu, C. Guo, Y. Li, Z. Zhou, T. Yuan, H. Wu, Y. Xiong, R. Gao, Y. Zhang, Serverswitch: a programmable and high performance platform for data center networks., in: Proceedings of USENIX NSDI′11, 2011.
[26] R. Niranjan Mysore, A. Pamboris, N. Farrington, N. Huang, P. Miri, S. Radhakrishnan, V. Subramanya, A. Vahdat, Portland: a scalable fault-tolerant layer 2 data center network fabric, in: Proceedings of ACM SIGCOMM′09, 2009.
[27] J.-Y. Shin, B. Wong, E.G. Sirer, Small-world datacenters, in: Proceedings of ACM SOCC′11, 2011.
[28] A. Singla, C.-Y. Hong, L. Popa, P.B. Godfrey, Jellyfish: networking data centers randomly, in: Proceedings of USENIX NSDI′12, 2012.
[29] L. Wang, F. Zhang, K. Zheng, A.V. Vasilakos, S. Ren, Z. Liu, Energy-efficient flow scheduling and routing with hard deadlines in data center networks, in: Proceedings of IEEE ICDCS′14, 2014.
[30] C. Wilson, H. Ballani, T. Karagiannis, A. Rowtron, Better never than late: meeting deadlines in datacenter networks, in: Proceedings of ACM SIGCOMM′11, 2011.
[31] L. Yu, H. Shen, Bandwidth guarantee under demand uncertainty in multi-tenant clouds, in: Proceedings of IEEE ICDCS′14, 2014.
[32] Y. Yu, C. Qian, Space shuffle: a scalable, flexible, and high-bandwidth data center network, in: Proceedings of IEEE ICNP′14, 2014.
[33] Y. Zhang, N. Ansari, On architecture design, congestion notification, TCP incast and power consumption in data centers, IEEE Commun. Surv. Tutor. 15 (1) (2013) 39–64.

Please cite this article as: D. Li et al., FleCube: A flexibly-connected architecture of data center networks on multi-port servers, Computer Communications (2015), http://dx.doi.org/10.1016/j.comcom.2015.09.023