Criso: An Incremental Scalable and Cost-Effective Data Center Interconnection by Using 2-port servers and low-end switches

Hao Feng
Jinan University
Guangzhou, P.R.China
76020634@qq.com

Yuhui Deng
Jinan University
Chinese Academy of Sciences
Guangzhou, P.R.China
tyhdeng@jnu.edu.cn (the corresponding author)

Yufan Zhao
Jinan University
Guangzhou, P.R.China
1540768718@qq.com

***Abstract*—With the data growing explosively, data center networks (DCN) have to possess the characteristics of incrementally scalable, cost-efficient, high network capacity and fault tolerance. However, the widely used DCNs can not meet the demands above. In this paper, we propose a new type of data center topology named *Criso* to settle the challenges. Different from the existed works, *Criso* has the advantages of both switch-centric topologies (servers do not participate in routing) and the server-centric topologies (the scalability is not limited by the ports of switches). It is constructed based on *pod*s, The internal structure of each pod is the same and there are only four external interfaces. By applying such structure, a *pod*-based and fault-tolerant routing algorithm is designed to handle multiple types of failures. *Criso* is hierarchically, recursively defined and high-network capacity which can scale up to millions of nodes. The analysis results demonstrate that the *Criso* model is significantly superior to four state-of-the-art data center structures in terms of the network capacity, scalability, cost, power consumption and other static characteristics. *Criso* achieves the target of low-cost, low-energy consumption and highly-scalability simultaneously.**

***Keywords*-Data center network; Scalability; Cost efficiency; Incremental scalability.**

I. INTRODUCTION

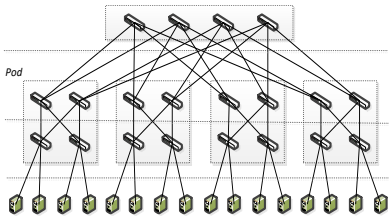
As a service-oriented platform, data centers form the core of cloud computing over the Internet. Data center networking designs both the network structure and associated protocols to interconnect thousands of or even hundreds of thousands of servers [1], [2], [3], [4], [5], [6]. Data centers are essential to offer numerous on-line applications, such as search, gaming, and Web mail. They also provide infrastructure services, such as GFS [7], Map-reduce [8], and Dryad [9]. Meanwhile, the cost and energy consumption have become the most important challenges of building data centers [10], [11], [12], [13], [14], [15], [16], [17], [18]. Therefore, data centers today normally use commodity computers and switches instead of high-end servers and interconnections to achieve cost-effectiveness. It is well understood that tree-vase solution in current practice can not meet the requirements [19], [20], [21], [22]. Therefore, many novel data center topologies are proposed and studied to meet different services.

The novel data center topologies can be divided into two categories: switch-centric and server-centric [3], [23]. A switch-centric network typically consists of multi-level trees of switches to connect the servers[24], [25]. However, data centers are growing large and the number of servers is increasing at an exponential rate. In order to cope with these challenges, the bandwidth of switches in the core level is increasing, and their cost is also getting higher and higher. Then the server-centric data center network models were proposed to reduce the cost and improve the network bandwidth [26]. In server-centric designs, interconnection intelligence is integrated into servers so that they can act as both computing nodes and routing nodes. These designs generate extra routing overhead on the servers. Many widely studied Researches of data center networks (DCN) fall into this category [20], [27], [28]. Hence, one data center topology with low-end switches and the servers which have a very low routing overhead is necessary.

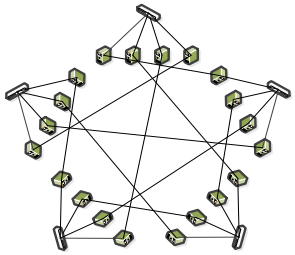
In this paper, we propose *Criso*, a scalable data center topology that works with servers with 2-port only and low-end commodity switches. *Criso* defines a recursive network structure in dimensions. A high-dimension *Criso* is constructed by two low-dimension *Criso*. In this way, the number of servers in *Criso*, denoted as n, grows more than two times with *Criso* dimensions.

The major contributions of this paper include:

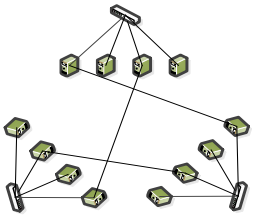
1) We propose a new type of data center network topology model, namely *Criso*. This model possesses lots of attractive characteristics. For example, the switches used in this model are low-end switches, the servers used require only two NICs, it is incrementally scalable and supports for massive servers. Besides, we study the fault-tolerant *Criso* routing algorithm (CRA) within this model. The CRA is self-adaption and it can handle multi-type faults.
2) We analyze the static characteristics including diameter, bisection width, the number of switches, ports, and wires against other five outstanding datacenter models including *Fat-Tree*, *BCube*, *DCell* and *Ficonn*. Furthermore, the cost and power consumption of the models



(a) Fat-Tree structure with n=4.



(b) $DCell_1$ structure with n=4.



(c) $FiConn_1$ with n=4.

(d) $Bcube_1$ with n=4.

Figure 1. Comparison of cost and energy consumption

are also investigated. The analysis results demonstrate that *Criso* is a superior candidate for building large-scale data centers.

The remainder of this paper is organized as follows. Section II introduces the related work. The definition and routing algorithm of *Criso* datacenter topology is introduced in Section III. Section IV compare the *Criso* topology with other four excellent structures. Section V presents the cost and energy consumption of *Criso* and other topologies. Section VI concludes this paper.

II. RELATED WORK

This section introduces four important data center network structures including *Fat-Tree* [19], *DCell* [20], *BCube* [27] and *FiConn* [28]. The *Fat-Tree* topology is a switch-centric structure. It has been widely used in practice by companies like Google, HP, SGI, Microsoft, IBM, SUN [7], [29]. Different from *Fat-Tree* topology, the other three structures belong to server-centric.

Fat-Tree: *Fat-Tree* [19], [30], [31] normally has three levels of switches. It consists of n pods, and each pod contains two levels (i.e., the edge level and the aggregation level) of n/2 switches. Each n-port switch at the edge level uses n/2 ports to connect n/2 servers, while using the remaining n/2 ports to connect the n/2 aggregation level switches in the pod. There are $n^2/4$ n-port switches at the core level. Each switch at the core level has one port connecting to one pod. Fig. 1(a) illustrates the topology of a *Fat-Tree* structure with n=4. However, The scalability of *Fat-tree* is limited by the ports of switches fundamentally.

DCell: *Dcell* [20], [32] is a level-based and recursively defined network structure. In $DCell_0$, n servers are connected to an n-port commodity switch. Given t servers in a $DCell_k$, t+1 $DCell_k s$ are used to build a $DCell_{k+1}$. The t servers in a $DCell_k$ connect to the other t $DCell_k s$, respectively. *DCell* has a high bisection width. Fig. 1(b) shows the topology of a $DCell_1$. The *DCell* structure uses dual-port NICs servers that increase the cost and deployment overhead.

FiConn: *Ficonn* [28], [33], [34] uses a recursive scheme that is similar to *DCell* to construct a data center network structure. $FiConn_0$ is composed of multiple servers and an n-port commodity switch connecting the servers. Every server in $FiConn_0$ has one port connected to the switch in $FiConn_0$. $FiConn_k$ is constructed by using b/2+1 $FiConn_{k-1}$, where b is the total number of available backup ports in $FiConn_{k-1}$. Fig. 1(c) depicts a $FiConn_1$ with n=4. However, its incremental scalability is not good.

BCube: *BCube* [27], [35] is also a server-centric interconnection topology. However, it is targeted for shipping-container-sized data centers containing 1-2*K* servers. It is also a level-based network structure. A $BCube_0$ is composed of n servers connecting to an n-port switch. A $BCube_1$ is constructed by employing n $BCube_0$ and n-port switches. By analogy, a $BCube_k$ is constructed by using n $BCube_{k-1}$ and n^k n-port switches. Each server in a $BCube_k$ has k+1 ports. Fig. 1(d) illustrates a $BCube_1$ with n=4. Like *DCell*, this structure uses dual-port NICs, leading a increase of cost.

III. CRISO DATA CENTER NETWORK

A. Definition of Criso

The *Criso* is defined recursively. It is composed of a series of low-end switches and 2-port commodity servers. In the $Criso_0$, we connect 8 switches as a cycle, and we connect k-3 servers in each switch while k is number of ports of switches adopted in such data center. This structure forms the 0 dimension *Criso*, denoted by $Criso_0$. We divide $Criso_0$ into 4 pods, each pod contains 2 switches and the correspondent servers. Each pod is labeled as (0,0),(0,1),(1,0),(1,1) in sequence according to its position (row and line) as shown in (Fig. 2). For example, the pod (0,1) means this pod is in the 0 row and 1 line. Both row and line are coded start from 0. Switches are labeled according to the pod they belong and the position of this pod as show in Fig. 3(a). For example, the switch (0,0,1) means it is in the (0,0) pod and the 1 position. Each position also represents for the way out of pod. Like switch in position 0 is responsible for the top exit of this pod, it links to the pod in above position, the switch in position 1 links to the pod in right position, and so on. It should be noted that if one pod has no neighbor pod in one position, the corresponding switch of this position is nonexistent. But when $Criso_n$

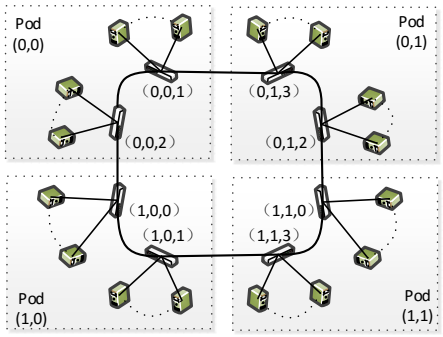
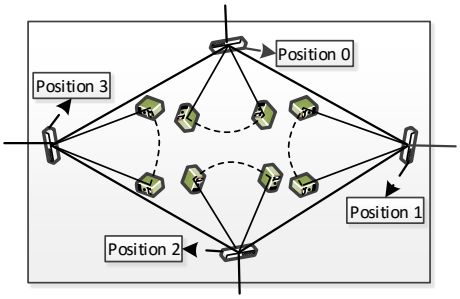
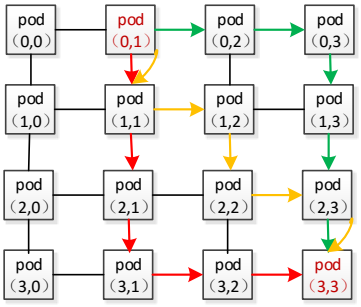


Figure 2. $Criso_0$.



(a) The topology of each pod



(b) $Criso_2$ with the example of routing.

Figure 3. The example of $Criso$

expands to $Criso_{n+1}$, the switch may be added. Let n denotes the dimension of *Criso*, for $n \geq 0$, $Criso_n$ contains the pods (a_0, a_1) from $Criso_{n-1}$ will maintain its value while the pods (a_0, a_1) from another $Criso_{n-1}$ changed according to the following rule:

1) If n is an odd, a_0 is maintained while $a_1 = a_1 + 2^{\frac{n+1}{2}}$.

2) If n is even, a_1 is maintained while $a_0 = a_0 + 2^{\frac{n}{2}}$.

Its worth noting that if n is an odd means this data center model is expanded crosswise (the row is maintained but line double), and the situation of n is even yield to the line is maintained but row double. For each pod, it can contain 2,3 or 4 switches according its position. The pods can on the corner, on the edge or inside *Criso*, and they contain 2,3 and 4 switches accordingly. For example, the four pods in $Criso_0$ are corner-pods (fig. 2). For each switch, there are always k-3 servers link with it. Hence, when one $Criso_n$

Algorithm 1 $Criso_0$ building algorithm

Input: *switches*, *servers*, k
Output: $Criso_0$
Define $pod(a_1, a_0)|a_i \in (0, 1)$
if $(a_1 = a_1'$ and $a_0 = !a_0')$ or $(a_0 = a_0'$ and $a_1 = !a_1')$ **then**
connect $pod(a_1, a_0)$ and $pod(a_1', a_0')$
end if
Define $switch(a_1, a_0, a_2)|a_i \in (0, 1)$ and $a_2 \in (0, ..., 3)$
connect $switch(a_1, a_0, a_2)$ to $switch(a_1, a_0, a_2 + 1 \ mod \ 4)$ and $switch(a_1, a_0, a_2 - 1 \ mod \ 4)$ for all switches
Define $server(a_1, a_0, a_2, b)|a_i \in (0, 1)$ and $a_2 \in (0, ..., 3)$ and $b \in (0, ..., k-3)$
connect each server(a_1, a_0, a_2, b) into switch (a_1, a_0, a_2)
return $Criso_0$

expanded to $Criso_{n+1}$, the amount of servers and switches is double. Servers are labeled according to the connected switch. We use a sequential number, $a_0, a_1, a_2.b$, to identify a server in *Criso*. Label (a_0, a_1, a_2) represents the connected switch of this server and b stand for the identity of this server (from 1 to k-3). From the definition of *Criso*, it can be obtained that each k-port switch connects k-3 servers and at most 3 switches. The servers used in *Criso* are 2-port servers, hence there is 1 back-up port available for each server in *Criso*.

Utilizing the definition above, it can be obtained that there are always 4 corner-pods in each $Criso_n$ (one in each corner), and two of they change into edge-pods when $Criso_n$ expand to $Criso_{n+1}$. In addition, when we use two $Criso_n$ to construct one $Criso_{n+1}$, the 1/4 of the edge-pods of $Criso_n$ will be chose to change into intra-pods according to the building rules.

At the structure of $Criso_n$, there are abundant switches between any two servers all the time. It means one server can access another server with multiple paths. For instance, in Fig .3, if servers in (0,0) pod wants access to (2,1) pod, under normal circumstances, we can choose the path $(0,0) \rightarrow (1,0) \rightarrow (2,0) \rightarrow (2,1)$ to route. Assume that the link between (1,0) and (2,0) fails, it means this link is unavailable. Under this circumstance, we can choose another path $(0,0) \rightarrow (1,0) \rightarrow (1,1) \rightarrow (2,1)$ to route. Therefore, it is still the communication between these servers even with faults appeared in some links. It also means any link will not be the bottleneck of the $Criso$. In addition, within applying low-end switches and 2-port servers, $Criso_n$ has high network capacity. Various paths make the routing protocol fault tolerant.

In our philosophy design, this structure is constructed and expanded according to pods. Thus, our $Criso_n$ is open and convenient to be expanded. $Criso_n$ also can scale exponentially and possess lots of advantages without high-end devices, we will discuss this below.

Algorithm 2 $Criso_n$ building algorithm

Input: $Criso_{n-1}$, k
Output: $Criso_n$
if n mod 2 =1 **then**
  Define $pod(a_1, a_2)$=$pod(a_1, a_0 + 2^{\frac{n+1}{2}})$
  for $i = 0, i \leq max(a_1), i++$ **do**
    Connect $pod(a_1, 2^{\frac{n+1}{2}})$,$pod(a_1, 2^{\frac{n+1}{2}} + 1)$
  end for
else n mod 2 =0
  Define $pod(a_2, a_0)$=$pod(a_1 + 2^{\frac{n}{2}}, a_0)$
  for $i = 0, i \leq max(a_0), i++$ **do**
    Connect $pod(2^{\frac{n}{2}}, a_0)$,$pod(2^{\frac{n}{2}} + 1, a_0)$
  end for
end if
return $Criso_n$

B. Criso Building Algorithm

Algorithm 1 shows the construction of a $Criso_0$ and Algorithm 2 describes the way to construct $Criso_n$ based on $Criso_{n-1}$. Because the position of servers is easy to determined once the closest switch is located, and the corresponding switch is effortless to find if the pod is located. Thus, this Algorithm 2 is mainly about the coding and sorting order of pods. Connect pod means connect the corresponding switches. $Criso$ is one neat data center and all of the pods are drawn up in orderly ranks. We use the two-dimensional array to identify the position of pods. Thus, if using the building algorithm to construct *Criso*, two-dimensional array of any one pod is given explicitly, and the clear position of this pod is obvious. It is also beneficial for the routing algorithm discussed below. We can recursively use Algorithm 2 to construct a large dimension $Criso_n$. For example, if we execute Algorithm 2 thrice based on $Criso_0$, one $Criso_3$ can be produced.

C. Criso Routing Algorithm

Criso routing algorithm (CRA) is efficient and simple by using switches represents the direction. According to the definition of *Criso* above, it is apt to get the pod of one server. Hence, the CRA mainly route among two pods. One server can be found easily if the switch it connected is sought out. Because the switches we use are low-end switches, and only at most 3 ports connected to switches while other ports connected to servers. We assign the position 0 switch connects the upside side pod, and position 1 links to the right side pod, position 2 switch connects to below side pod while position 4 switch connect pod in below side. Thus, if we want to route between pod *j* to pod *k*, we only need to know the position of *k* pod is by which side of *j*, and then we can choose the relevant port of each pod in the path to route. We use src to denote the source pod and dst denotes the destination pod. For example, assume that we want to work out the path from src *j*=(0,1) to dst *k*=(3,3), from the code of *j* and *k* can we know *j* is on the 0-row and 1-line while *k* on the 3-row and 3-line (as shown in fig. 3). Thus the routing between *j* and *k* needs to go downward 3-0=3 steps and go right 3-1=2 steps. It is noteworthy that it does not have certain direction about each step, but the sum of direction steps is decided. Hence, it can first go downward 3 steps and then go right, or first go right 2 steps and then go downward, or go cross-linked of downward and right (this three paths is shown in fig. 3). So it is not only convenient for routing but also facility for avoiding congestion. If any link is blocked of package or even breakdown, the routing algorithm can select another one path fast. The Algorithm 3 (src(a_1, a_0) to $dst(a_3, a_2)$) only showed the situation of a3¿a1 and a2¿a0 for convenient, other situations is similar.

Algorithm 3 Criso routing algorithm

Input: src(a_1, a_0), $dst(a_3, a_2)$
Output: *Path*
if src==dst **then**
  Return NULL
else Define $j = a_3 - a_1, k = a_2 - a_0$
  if $j \geq 0$ and $k > 0$ **then**
    Chose the switch in position 1 or position 3 randomly for routing
    if chose the switch in position 1 **then**
      Record this path and jump to next pod
      j=j-1
      Path=$CRA(src(a_1, a_0 + 1), dst(a_3, a_2))$
    else chose the switch in position 3
      Record the path and jump to next pod
      k=k-1
      Path=$CRA(src(a_1 + 1, a_0), dst(a_3, a_2))$
    end if
  end if
end if

IV. Comparison with other DC topology models

Table I compares *Criso* with other five significant data center networks including *Fat-Tree*, *DCell*, *BCube*, and *Ficonn* in terms of scalability, diameter, and the number of switches, where the parameter *N* stands for the number of servers in a *Criso* data center network, and *k* represents the number of switch ports. Because *DCell*, *BCube*, *Ficonn*, and *Criso* are all structures that recursively defined, the level of network structure is defined as *n*. All the characteristics have a significant effect the performance of one data center network structure.

A. Diameter

On the basis of the building scheme, we can calculate the diameter of the *Criso*. Let *Diam* denote the diameter of network. Because the *Criso* is building based on pods, we first calculate the diameter of any two pods.

Case 1. n is an odd

$$Diam(pod) = 6 \times 2^{\frac{n-1}{2}} - 2$$

Case 2. n is an even

Table I
THE CHARACTERISTICS OF DATA CENTER TOPOLOGIES

	Fat-Tree	**DCell**	**BCube**	**FiConn**	**Criso**
Diameter	$2log_2^N$	$\leq 2^{n+1}-1$	$n+1$	$\leq 2^{n+1}-1$	$\leq 3\times 2^{\frac{n+3}{2}}$
Bisection Width	$N/2$	$> N/(4\times log_k^N)$	$N/2$	$> \frac{N}{4*2^k}$	$> 2^{\lceil\frac{n+1}{2}\rceil}$
Scalability limited by server ports	No	Yes	Yes	No	No
Scalability limited by switch ports	Yes	No	No	No	No
Number of switches	$5N/k$	N/k	$(n+1)N/k$	N/k	$N/(k-3)$
Number of server ports	≤ 2	$n+1$	$n+1$	≤ 2	2
Number of switch ports	k	k	k	k	k

$$Diam(pod) = 4\times 2^{\frac{n}{2}} - 2$$

In each pod, the packet transmission goes through two routers at most. And the distance of one server to the associated switch is 1. Thus, the diameter of *Criso* can be denoted as:

Case 1. n is an odd

$$Diam(pod) \leq (6\times 2^{\frac{n-1}{2}} - 2)\times 2 + 4 = 3\times 2^{\frac{n+3}{2}}$$

Case 2. n is an even

$$Diam(pod) \leq (4\times 2^{\frac{n}{2}} - 2)\times 2 + 4 = 2^{\frac{n+6}{2}}$$

Theoretically, communication delay grows with the growth of the network diameter. As Table I shows, the diameter of *Criso* is ordinary.

B. Bisection width

Large bisection width means a high data center capacity and stronger ability against failure. Based on the definition of *Criso*, the bisection of it is large than $2^{\lceil\frac{n+1}{2}\rceil}$. As shown in Table I, the bisection widths of *Ficonn* and *Criso* are lower than *Fat-Tree*, *DCell* and *BCube*. This situation is mainly because these three data centers above use a large number of switches and links to build one data center. Unfortunately, this approach will increase energy consumption and cost. Hence, *FiConn* and *Criso* are more suitable for constructing one data center.

C. Scalability

A typical way to expand one data center is adding more servers and switches to such system rather than replacing old ones. Therefore, the scalability of one data center network is crucial to the overall system performance.

All the five data center networks shown in Table I can expand to an extremely large scale. However, the scalability of *DCell* and *BCube* is limited by the server ports, and the scalability of *Fat-Tree* is limited by the switch ports. This means, once a data center is constructed by certain servers and switches, the scalability of it is extremely restricted. If more switches and servers are needed, few additional ports have to be added to the original switches and servers. In contrast to *DCell*, *BCube* and *Fat-tree*, *Criso* and *FiConn* can be expanded without adding additional server ports or switch ports. Thus, this two data centers show better scalability than other three models.

V. COST AND ENERGY CONSUMPTION

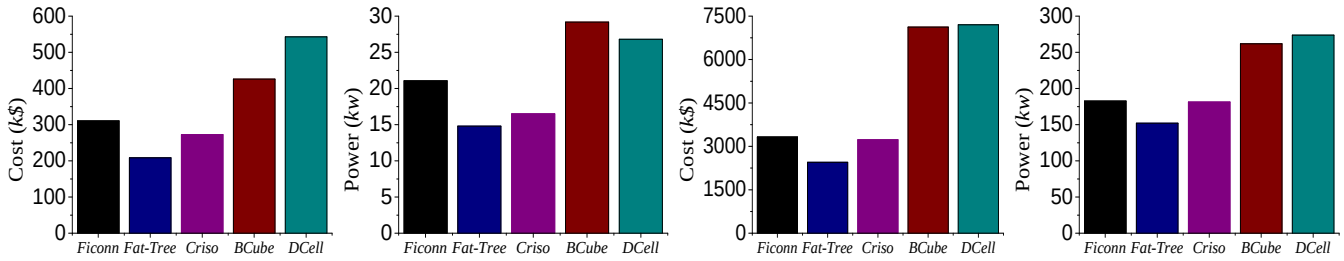
Both *FiConn* and *Criso* are kinds of recursively defined network structure models, it also means that the n-level (or dimension) structure is constructed by using more (n-1)-level structures. In *Criso*, one n+1 dimension *Criso* build based on two n dimension *Criso*s. But in *FiConn*, more k level structure will be used to construct one (k+1) level structure. This feature leads to a sharp increase in system scale.

The scalability of *Criso* is much better than *FiConn*. An incomplete *Criso* network can be constructed firstly. It means we do not add parallel-edges which contain servers for some circle-edges. And when the *Criso* is required to expand for more servers, we can dynamically add those parallel edges into the incomplete *Criso* network and servers are linking on it to increase the system scale. Additionally, the routing algorithm of the incomplete *Criso* network is the same as routing algorithm developed above.

Table II
PRICE AND POWER CONSUMPTION OF SWITCH AND NICS

	Product	Ports	Price($)	Power(W)
Switch	D-Link DES-1008A	8	12	4.3
	D-Link DES-1016A	16	39	6
	D-Link DES-1024R	24	76	6.4
	D-Link DES-1048	48	208	10
NIC	Intel EXPI9400PT	1	75	5
	Intel EXPI9402PT	2	130	7
	Intel EXPI9404PT	4	285	10

This section compares the energy consumption and cost of *Criso* against *Fat-Tree*, *DCell*, *FiConn* and *BCube*. In order to study the impact of different system scales on the cost and energy consumption, we simulate two different data centers that contain 2048 servers and 24640 servers, respectively. Since the number of servers in the five differ data center networks is identical (2048 servers or 24640



(a) The cost comparison of models with 2048 servers (b) The power comparison of models with 2048 servers (c) The cost comparison of models with 24640 servers (d) The cost comparison of models with 24640 servers

Figure 4. Comparison of cost and energy consumption

Table III
COST, POWER AND WIRING COMPARISON OF A DATA CENTER WITH 2048 SERVERS.

	Cost(k$)			Power(kw)			Wires
	Switch	NIC	Total	Switch	NIC	Total	
FC	3.3	307.2	310.5	0.56	20.5	21.06	2805
FT	54.7	153.6	208.3	4.6	10.2	14.8	8960
BC	6.1	419.8	425.9	2.2	24.6	26.8	8704
DC	9.9	532.8	542.7	0.48	28.7	29.18	3384
Cr	6.1	266.2	272.3	2.2	14.3	16.5	7424

Table IV
COST, POWER AND WIRING COMPARISON OF A DATA CENTER WITH 24640 SERVERS.

	Cost(k$)			Power(kw)			Wires
	Switch	NIC	Total	Switch	NIC	Total	
FC	121.3	3203.2	3324.5	10.2	172.5	182.7	48279
FT	600	1848	2448	28.8	123.2	152	79963
BC	178	7022.4	7200.4	27.3	246.4	273.7	85683
DC	99.9	7022.4	7122.3	15.4	246.4	261.8	62279
Cr	24.6	3203.2	3227.8	8.8	172.5	181.3	35840

servers), the summation of cost and energy consumption of servers are the same. Therefore, we only evaluate and analyze the cost and energy consumption occurred by the switches and NICs of the data centers. Table II shows the basic price and power consumption of switch and NICs may use in the data centers. Owing to the data center networks must be laid out with wires and cables, the number of links used in different networks is also presented. Table III shows the cost, power consumption and the number of links used in the five different network structures to construct one data center which containing 2048 servers. For convenience, FC denotes the Ficonn topology, FT denotes Fat-tree, BC denotes BCube, DC denotes DCell, and Cr denotes Criso topology. Table IV describes the same statistics when the data center is expanded to 24640 servers. Both tables demonstrate that the costs and power consumption of *DCell*, *BCube* and *Ficonn* are much higher than that of *Fat-Tree* and *Criso*. Additionally, the cost and the power consumption of *Fat-Tree* is the lowest among the above five data center structures. But, in fact, the scalability of *Fat-Tree* is limited by the number of switch ports. Therefore, it is not suitable for large-scale data centers constructed. Neither *BCube* nor *DCell* is suitable for constructing large-scale data centers with respect to cost and energy consumption according to the statistics in Table III and Table IV. Fig. 4 confirm this conclusion. In spite of the prices vary in the market, it will not affect the results summarized in Table III, Table IV, and Fig. 4. When different components are used to build data centers, once the number of servers is identified then both the number of switches and NICs are determined. Hence the prices of switches and NICs will affect the total cost and energy consumption of the data center network, but will not impact the comparison results. The analysis above indicated that both *FiConn* and *Criso* have the advantage in building reality data centers. But, in general, *Criso* is the most appropriate one to build one large-scale data centers in all respects.

Table III and Table IV also list the number of links (or wires) used in topologies. Table III indicates that the number of wires used in *FiConn* and *DCell* is smaller than other three data centers. But if the data center scales up from 2048 servers to 24640 servers, *Criso* takes the minimal number of links. Above all, *Criso* is a better candidate for building large-scale data centers.

VI. CONCLUSIONS

In this paper, we present *Criso*, a pod-based data center structure (or model) that utilizes the 2-port servers and low-end switches. The Criso is a low-cost and scalable structure with low diameter. The routing algorithm in *Criso* is also presented.

We compare *Criso* against other four typical new pattern data center network structures including *Fat-Tree*, *DCell*, *BCube* and *FiConn*. Criso structure shows great performance in all aspects. The cost, energy consumption and the number of wires are evaluated in the last. The results demonstrate that *Criso* is the best candidate for building large-scale data centers.

ACKNOWLEDGEMENTS

This work is supported by the NSFC (no.61572232), in part by the Science and Technology Planning Project of Guangzhou (no.201802010028,and no.201802010060),in part by the Science and Technology Planning Project of Nansha (no.2017CX006), and in part by the Open Research Fund of Key Laboratory of Computer System and Architecture, Institute of Computing Technology, Chinese Academy of Sciences under Grant CARCH201705. The corresponding author of this paper is Yuhui Deng.

REFERENCES

[1] J. Xie, Y. Deng, G. Min, and Y. Zhou, "An incrementally scalable and cost-efficient interconnection structure for datacenters," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 6, pp. 1578–1592, 2017.

[2] H. Rong, H. Zhang, S. Xiao, C. Li, and C. Hu, "Optimizing energy consumption for data centers," *Renewable and Sustainable Energy Reviews*, vol. 58, pp. 674–691, 2016.

[3] Y. Zhang and N. Ansari, "On architecture design, congestion notification, tcp incast and power consumption in data centers," *IEEE Communications Surveys and Tutorials*, vol. 15, no. 1, pp. 39–64, 2013.

[4] Y. Deng, "What is the future of disk drives, death or rebirth?" *Acm Computing Surveys*, vol. 43, no. 3, pp. 1–27, 2011.

[5] G. C. Sankaran and K. M. Sivalingam, "Optical traffic grooming-based data center networks: Node architecture and comparison," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 5, pp. 1618–1630, 2016.

[6] Y. Zhao, H. Jiang, K. Zhou, Z. Huang, and P. Huang, "Dream-(l)g: A distributed grouping-based algorithm for resource assignment for bandwidth-intensive applications in the cloud," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 12, pp. 3469–3484, 2016.

[7] S. Ghemawat, H. Gobioff, and S. T. Leung, "The google file system," *Acm Sigops Operating Systems Review*, vol. 37, no. 5, pp. 29–43, 2003.

[8] J. Dean and S. Ghemawat, "Mapreduce: simplified data processing on large clusters," in *Conference on Symposium on Opearting Systems Design and Implementation*, 2004, pp. 10–10.

[9] M. Isard, M. Budiu, Y. Yu, A. Birrell, and D. Fetterly, "Dryad:distributed data-parallel programs from sequential building blocks," *ACM SIGOPS Operating Systems Review*, vol. 41, no. 3, pp. 59–72, 2007.

[10] R. Lin and Y. Deng, "Allocating workload to minimize the power consumption of data centers," *Frontiers of Computer Science*, vol. 11, no. 1, pp. 105–118, 2017.

[11] H. Xu, C. Feng, and B. Li, "Temperature aware workload managementin geo-distributed data centers," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 6, pp. 1743–1753, 2015.

[12] L. Yu, T. Jiang, and Y. Cao, "Energy cost minimization for distributed internet data centers in smart microgrids considering power outages," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 1, pp. 120–130, 2015.

[13] A. M. Alqawasmeh, S. Pasricha, A. A. Maciejewski, and H. J. Siegel, "Power and thermal-aware workload allocation in heterogeneous data centers," *IEEE Transactions on Computers*, vol. 64, no. 2, pp. 477–491, 2015.

[14] Z. Zhang, Y. Deng, G. Min, J. Xie, and S. Huang, "Exccc-dcn: A highly scalable, cost-effective and energy-efficient data center structure," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 4, pp. 1046–1060, 2017.

[15] J. Wei, H. Jiang, K. Zhou, and D. Feng, "Efficiently representing membershipfor variable large data sets," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 4, pp. 960–970, 2014.

[16] L. Zhang, Y. Deng, W. Zhu, J. Zhou, and F. Wang, "Skewly replicating hot data to construct a power-efficient storage cluster," *Journal of Network and Computer Applications*, vol. 50, pp. 168–179, 2015.

[17] Y. Deng, Y. Hu, X. Meng, Y. Zhu, Z. Zhang, and J. Han, "Predictively booting nodes to minimize performance degradation of a power-aware web cluster," *Cluster Computing*, vol. 17, no. 4, pp. 1309–1322, 2014.

[18] Y. Deng and F. Wang, "Exploring the performance impact of stripe size on network attached storage systems," *Journal of Systems Architecture*, vol. 54, no. 8, pp. 787–796, 2008.

[19] M. Alfares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," *Acm Sigcomm Computer Communication Review*, vol. 38, no. 4, pp. 63–74, 2008.

[20] C. Guo, H. Wu, K. Tan, L. Shi, Y. Zhang, and S. Lu, "Dcell:a scalable and fault-tolerant network structure for data centers," 2008, pp. 75–86.

[21] K. Chen, C. Hu, X. Zhang, K. Zheng, Y. Chen, and A. V. Vasilakos, "Survey on routing in data centers: insights and future directions," *IEEE Network*, vol. 25, no. 4, pp. 6–10, 2011.

[22] Y. Deng, X. Huang, L. Song, Y. Zhou, and F. Wang, "Memory deduplication: An effective approach to improve the memory system," *Journal of Information Science and Engineering*, vol. 33, pp. 1103–1120, 2017.

[23] G. Qu, Z. Fang, J. Zhang, and S. Q. Zheng, "Switch-centric data center network structures based on hypergraphs and combinatorial block designs," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 4, pp. 1154–1164, 2015.

[24] O. Popoola and B. Pranggono, "On energy consumption of switch-centric data center networks," *Journal of Supercomputing*, vol. 74, no. 1, pp. 334–369, 2018.

[25] S. Azizi, N. Hashemi, and A. Khonsari, "A flexible and high-performance data center network topology," *Journal of Supercomputing*, vol. 73, no. 4, pp. 1–20, 2016.

[26] I. A. Stewart and A. Erickson, "The influence of datacenter usage on symmetry in datacenter network design," *Journal of Supercomputing*, no. 5, pp. 1–38, 2017.

[27] C. Guo, G. Lu, D. Li, H. Wu, X. Zhang, Y. Shi, C. Tian, Y. Zhang, and S. Lu, "Bcube:a high performance, server-centric network architecture for modular data centers," in *Acm Sigcomm Conference on Data Communication*, 2009, pp. 63–74.

[28] D. Li, C. Guo, H. Wu, and K. Tan, "Ficonn: Using backup port for server interconnection in data centers," in *INFOCOM*, 2009, pp. 2276–2285.

[29] A. Greenberg, J. Hamilton, D. A. Maltz, and P. Patel, "The cost of a cloud:research problems in data center networks," *Acm Sigcomm Computer Communication Review*, vol. 39, no. 1, pp. 68–73, 2008.

[30] Z. Guo, J. Duan, and Y. Yang, "On-line multicast scheduling with bounded congestion in fat-tree data center networks," *IEEE Journal on Selected Areas in Communications*, vol. 32, no. 1, pp. 102–115, 2013.

[31] Y. H. Lo, Y. Zhang, Y. Chen, H. L. Fu, and W. S. Wong, "The global packing number of a fat-tree network," *IEEE Transactions on Information Theory*, vol. 63, no. 8, pp. 5327–5335, 2017.

[32] X. Wang, J. Fan, J. Zhou, and C. K. Lin, "The restricted h h mathcontainer loading mathjax -connectivity of the data center network dcell," *Discrete Applied Mathematics*, vol. 203, pp. 144–157, 2016.

[33] Y. Zhang, A. J. Su, and G. Jiang, "Evaluating the impact of data center network architectures on application performance in virtualized environments," in *International Workshop on Quality of Service*, 2010, pp. 1–5.

[34] K. U. Bhanu and K. C. Shekar, "Shortest path routing algorithm for ficonn in load balanced data center networks," *International Journal of Scientific and Engineering Research*, pp. 120–123, 2013.

[35] V. Asghari, R. F. Moghaddam, and M. Cheriet, "Performance analysis of modified bcube topologies for virtualized data center networks," *Computer Communications*, vol. 96, pp. 52–61, 2016.