RSI: A High-Efficient and Fault-Tolerant Interconnection for Resources-Pooling in Rack Scale

Mingche Lai, Xiangxi Zou, Shi Xu, Jie Jian, Xingyun Qi, Jiaqing Xu
College of Computer
National University of Defense Technology
Changsha, China
zouxiangxi15@nudt.edu.cn

***Abstract*—Data centers are originally composed of servers that deal with different services, and have been developed into rack-mounted computers and blade server systems. At the same time, the traffic between different servers is increasing, and the interconnection network performance between servers has significant influence on the overall performance of the system. Based on the resource pools background in the development of data center recently, this paper proposes a rack-scale interconnection network structure named RSI. According to the features of the interconnection structure, the routing tables are designed with two levels, which can reduce the size of tables and be convenient to realize adaptive routing. Then a low cost adaptive routing called LAR with a threshold to decide the choice of routing is proposed. We come up with a deadlock prevention mechanism and the deadlock can be prevented in LAR with only 2 VCs. In addition, a fault tolerance algorithm is used to deal with potential failures. Compared to current interconnection topologies in a rack, RSI hierarchical topology can support larger scale of nodes with comparable performance. Finally, the evaluation results show that in extreme traffics, LAR achieves about 6 times throughput than minimal routing which performs well in uniform random traffic.**

Keywords—Rack scale; resource pooling; interconnection; adaptive routing; fault-tolerant

I. INTRODUCTION

With the development of cloud computing and large data technology, the scale of the data center is growing, but the cost of building a large-scale data center cannot be ignored. With the expansion of the data center, the cost will grow geometrically if the servers, storages and other equipment are simply stacked. This type of expansion is not only costly, but also the utilization of resources is relatively low. This is because the equipment is isolated, forming a seat "island". In order to solve this problem, the concept of "resource pool" has been proposed, such as Intel's RSD (Rack-Scale Design) [9]. In "resource pool" design, computing, storage and network devices are abstracted into logical resources or services with APIs which can be invoked by the upper platform or application and the resources can be allocated flexibly.

Nowadays some new storage technologies such as 3D XPoint [2] and NVMe [3] are used to build the next generation of data center. And the interconnection network among the new type of storages should be better than current ones. The storage medium itself has a lower data operation delay, so the network needs to have a lower communication delay to match the performance of the new storage medium. In addition, the communication of storages is different from other flows such as video traffic, which allows the data to a certain error. The communication between storage devices requires high reliability support, so the storage network should guarantee the quality of communication. Most of the current data center storage networks are based on Ethernet or FC protocol. The versatility and low cost of Ethernet is the main reason of its wildly usage, but also led to excessive protocols, bringing a higher communication delay.

In this paper, a storage structure for resource pool design is proposed, and an interconnection structure is designed to meet the rack-scale interconnection request. The main innovations of this paper are as follows: (1) New storage architecture in order to build a storage pool. With new storage technologies such as Non-Volatile Memory (NVM), the performance of storage system grows rapidly. So we propose a new way to interconnect storage devices with SerDes technology in an integrated way other than current I/O subsystems with bridge chips to match the performance of emergency NVMs. (2) A rack scale interconnection architecture with routing and deadlock prevention mechanisms. Rack scale design (RSD) was proposed by Intel [9], and develops well in recent years. We combine the new storage architecture with RSD to build a storage pool in a rack. So we design a hierarchical inter-connection topology to connect nodes in a rack. And we design the routing tables with two layers to match hierarchical topology. Then we propose a low cost adaptive routing algorithm (LAR) to balance the load in extreme traffic in which the evaluation result shows that LAR can realize about 6 times throughput than minimal routing algorithm. Moreover, the deadlock can be prevented with only 2 VCs in LAR. (3) A fault-tolerance mechanism to update the routing tables when failures occur or recover. The mechanism will keep listening relative packets to judge whether some links or nodes are failed or recovered from failure. Then the routing tables will be updated according to the algorithms of it.

Some related work is discussed in Section II. A new storage structure and the rack scale interconnection design are proposed in Section III. Section IV contains a low cost adaptive routing algorithm called LAR with deadlock prevention. Then a fault-tolerance mechanism has been proposed in Section V. The evaluation results are illustrated in Section VI.

II. Related Work

A. Storage Network Technologies

Storage devices are able to communicate with the processor through the I/O expansion port provided by the bridge chip on the motherboard, which is called direct-attached storage (DAS) [12]. This approach has been used in personal computers, but DAS is difficult to expend to a large scale such as in a data center. So storage area network (SAN) [13] came up. SAN systems often use Fiber Channel (FC) and specific FC switches to connect storage arrays with servers. But there are only a small number of vendors who can provide FC equipment, which makes the cost of SAN high. Another way to support large scale storage capacity is Network Attach Storage (NAS) [14] and NAS can be implemented over TCP/IP networks. Most of the existing storage networks are based on Ethernet and require a wide range of protocols and standard conversions, resulting in higher latency and lower efficiency. One possible solution is a new and efficient in-rack communication protocols that have lower overhead to improve link bandwidth utilization, enabling high bandwidth and low latency.

In addition, InfiniBand technology [15] is wildly used in Data Center interconnection now. InfiniBand contains an interconnection protocol with a flat exchange architecture but centralized management. It is originally designed for low latency and high bandwidth communication. The I/O performance in Intel architecture is restricted by the speed of PCI or PCI-X bus which is 500MBps and 1GBps, respectively. The communication ability between servers and storage devices is restricted much. Infiniband will be integrated directly into the system board and interact directly with the CPU and the memory/storage subsystem. But it is still supported by the PCI adapter and is restricted by to PCI bus by now. And expensive specific Infiniband switches are necessary.

B. Noval Interconnection Technologies to Storage Systems

With the development of storage technology like some Non-Volatile Memory (NVM) [10], [11], the performance of storage system grows rapidly. So the traditional interconnect technology may not be able to be well matched about the performance of storage media performance. It has been mentioned that the storage and network will often become a bottleneck in some descriptions of RSD by Intel. We need to explore some new interconnect structure.

Hewlett-Packard officially announced a new computer prototype “The Machine” with NVM called memristors in London on November 27, 2016. And all memories of nodes are connected so every processor can fetch memories of any other nodes by load/store instructions. With the interconnection in memory level, the low cost communication can be achieved. In the meanwhile, the interface to the storage medium also needs to be changed to match the high performance of storage devices. The PCIe or DDR3 interface called Dual In-line Memory Module (DIMM) can solve this problem. But they need more pins, and the board wiring is a great challenge. The latest 25G SerDes technology may become a new solution to it. Table 1 illustrates the comparison of the above three high-speed interfaces. The high-speed SerDes doubles the performance of DDR3, and the number of pins is about 1/12. Compared to the latest PCIe technology, SerDes consumes 1/4 pins of the former.

Due to high degree of integration, the rack scale inter-connection network is limited by the number of pins. In addition, the storage network is different from the general transmission network, requiring a certain quality assurance to avoid the high cost of retransmission. And current storage networks like bus or star networks are difficultly expend to rack-scale. We propose a hierarchical network which can easily support 512 to 1024 nodes with low latency to support rack scale interconnection.

TABLE I. Comparison of Three Types of High Speed Interfaces

Items	4*25G[17]	DDR3	PCIe3.0 16x
Maximum transfer rate	12.5GB/s	6.4GB/s	12.5GB/s
64-bit transmission delay	0.64ns	1.25ns	0.5ns
Number of pins	16	200	64

In high performance computing (HPC) [18], Blue Gene [16] adopts 5D Torus to connect nodes which integrate routing unit to avoid using additional switches. But the network is larger in diameter due to the limited number of I/O pins. The tree like structure such as Fat Tree (FT) [4] is also commonly used now. But it does not suit for a rack because the limited pins or degree of nodes, k. For example, the maximum number of nodes that can be interconnected at two levels FT is 128 with k=8. If more nodes should be connected, additional switches may be used to form an indirect network [1].

C. Routing and Deadlock Prevention

When the network traffic is evenly distributed, the shortest routing can maximize the use of network bandwidth. But local link congestion may occur in some extreme cases. For example, when many nodes send packets to the same one, intermediate nodes will be needed to alleviate congestion. We proposed a low cost adaptive routing mechanism where the egress port of each hop is dynamically selected based on the local state.

Deadlocks arise because the number of resources is finite [1]. When a packet is holding a channel, and then it requests another channel, so there is a dependency between those channels. Cyclic dependencies may lead to deadlocks. Using Virtual Channel (VC) [1] can prevent deadlocks, and the number of VC is related to the diameter of the network [7]. In Dragonfly with Valiant routing mechanism [5], a packet will traverse no more than 5 hops (l-g-l-g-l, l: intra-layer routing, g: inter-layer routing) to reach the destination. So it need 5 VCs (VC1 to VC5) to pretend deadlocks, and the order of the VC request must be in an ascending way. For example, the first hop will apply for VC1 while the 5th hop will apply for VC5. Furthermore, combined with the feature of Dragonfly and Valiant routing, it can be seen that 1, 3, 5 hops only occur within a layer and the 2, 4 hops only occur between two layers. So the number of VCs can be set differently according to the type of port to save resources. Moreover, Marina [8] proposed a “symbol + parity” routing mechanism where there are 4 types of channels and the request rules of them are restricted in order

to prevent deadlocks without external VC but more complicated logics.

III. Rack Scale Interconnection Architecture

In traditional architecture (Fig. 1 right), the processor fetch the local storage devices through bridge chips. If the processor wants to access some storage devices in other nodes (such as SAN storage array), PCIe and network is necessary, leading to much delay. The NIC is integrated to the storage controller in RSI (Fig. 1 left). With some specific protocols, high-speed access to remote storage devices and a pooling storage will come true. The high-speed interfaces in Table 1 can be used in RSI, but the SerDes technology is more about to be used in the future.

To interconnect nodes in a rack, a hierarchical topology is proposed in Fig. 2 named RSI topology. Nodes (represented by cycles) will be interconnected in layer 1 to form a "super node" (represented by a matrix). The number of ports in a super node is called "virtual radix" which depends on a) the radix (the number of ports) of each node and b) the interconnection way in layer 1. In layer 2, the number of links between two super nodes ranges from 1 to more which depends on the scale of whole network.

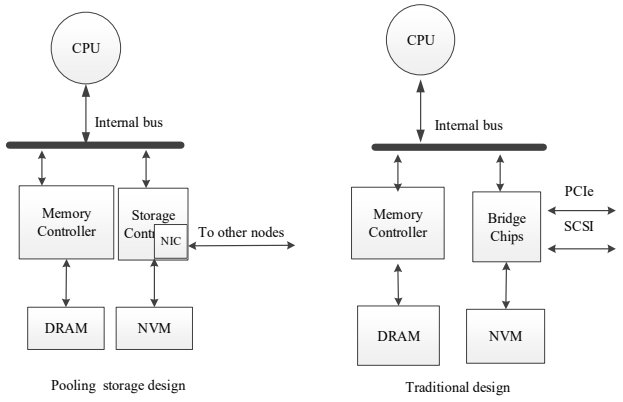

Fig. 1. RSI architecture(left) vs. traditional architecture(right).

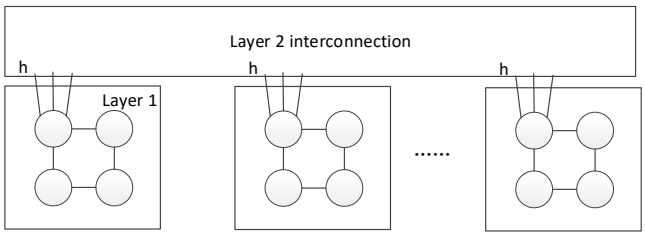

Fig. 2. Hierarchical interconnection design.

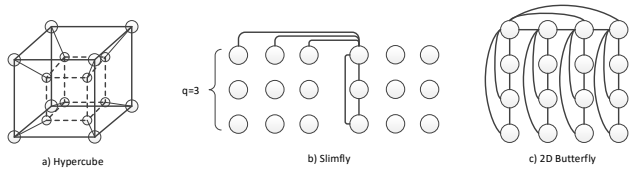

Fig. 3. Layer 1 interconnection topology.

TABLE II. Scale of Whole Network with Different Layer 1 Topology

Layer 1 topologies (number of nodes)	Virtual radix of super nodes	The maximum number of nodes in network
Full (5)	20	105
Hypercube (16)	64	1040
Slim Fly (16)	54	990

We will next compare RSI topology with Dragonfly [5] which is also a hierarchical topology. In Dragonfly, nodes are fully interconnected in each group and groups are fully interconnected too. When the radix of each node equal to 8 (k=8), the maximum number of nodes can be interconnected is 105 which is much less than the number of nodes in a rack. There are three types of layer 1 interconnection topology illustrated in Fig. 3. They are Hypercube, Slim Fly [6] and 2D Butterfly. We choose them to interconnect nodes in layer in because of their scalability compared to fully interconnection topology and comparable local performance. For example, the communication distance in 2D Butterfly is no more than 2 hops, and the resulting super node owns 32 virtual ports which can connect to other 32 super nodes at most. The largest number of nodes can be interconnected in different way are shown in Table 2, when k=8. Full means nodes are fully interconnected, so there are 5 nodes in each super node with virtual radix equal to 20 in Dragonfly.

Three layer 1 topologies in Fig. 3 are able to form a network which is able to interconnect all nodes in a rack. As a consequence, there may be only one link between every two super nodes. And it is more likely to become a bottle neck. There should be an appropriate routing mechanism to balance the load and we will show that in Sector IV. Moreover, the diameter of Hypercube is larger than another two topologies which makes the performance degradation faster in hot spot traffic.

IV. Low Cost Adaptive Routing

A. Routing Table Design

Since the topology is hierarchical in RSI with two layers, we design two routing tables: 1) a routing table for communication within each layer; 2) a routing table of super nodes information. The hierarchical routing tables can a) reduce the size of whole routing table, and b) bring convenience to design adaptive routing. For example, in a RSI network with 3 super nodes and 16 nodes in every super node, there need to be 47 routing entries in each node before but only 2+15 entries totally with two routing tables. Meanwhile, adaptive routing inside layer 1 can be achieved by looking for Table 1. According to the above, we design the format of head and routing table in Fig. 4. Note that there may be more than one minimal path for some nodes.

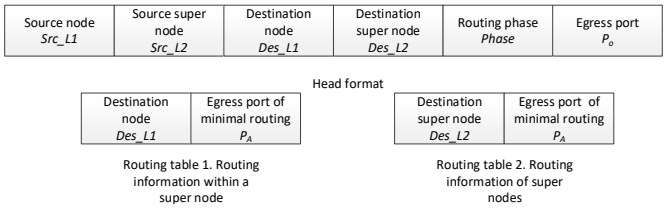

Fig. 4. Head format and two routing tables in RSI.

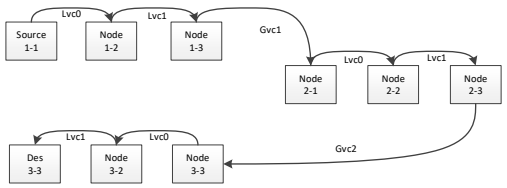

Fig. 5. VC allocation in BR (L: links inside a super node, G: links between two super nodes).

B. Low Cost Adaptive Routing with Deadlock Prevention

We define two different priority routing methods before we discuss the adaptive routing algorithm: 1) the minimal routing with high priority; and 2) non-minimal routing with low priority. In non-minimal routing, packets will not be sent to the destination super node directly but an intermediate super node. Once a packet arrives at the intermediate super node, it will be sent to the destination super node in minimal routing to avoid live-lock [1]. We use routing phase tag to identify the routing phase of a packet in Fig. 4.

A counter is set in each egress port to record the request number of it. When a packet arrives at a node, the egress port P_o will be selected by minimal routing computing (or looking for tables) initially. If the counter number of P_o is less than a threshold, the packet will be sent to P_o and the counter will self-increase. Otherwise, the egress port will be recalculated by non-minimal routing. The threshold is vital in adaptive routing and should be carefully selected. The low cost adaptive routing algorithm named LAR is described in detail below in Table 3.

TABLE III. LOW COST ADAPTIVE ROUTING ALGORITHM

Algorithm 1. Low cost adaptive routing algorithm(LAR) of RSI

1. Input：Head of packet; Address of current node: Cur_L1, Cur_L2; Routing tables generated by minimal routing algorithm: T1, T2; The counter value of each egress port, Qi, i=0..n-1; The threshold value, Th.
2. Ouput：The No of egress port P_o
3.
4. *Receiving FD(Region)*
5. if *Cur_L2 == Des_L2* then
6. if *Cur_L1 == Des_L1* then
7. *The packet arrive at destination*
8. else
9. *Get P_A from T1 by Des_L1*
10. end if
11. else
12. *Get P_A from T2 by Des_L2*
13. end if
14. if *Qa < Th or Phase == 1* then
//Phase == 1 means that the adaptive routing has ever been done, so //the minimal routing must be adopted next to avoid //live-lock
15. *$P_o = P_a$*
16. else
17. *if Cur_L2 == Des_L2 then*
18. //congestion occurs in the super node
19. *Get a set of egress port no, P(exclude P_A) where Qi < Th and Pi connect to nodes in the same super node*
20. else //congestion occurs between super nodes
21. *Get a set of egress port no, P(exclude P_A) where Qi < Th and Pi connect to nodes in other super nodes*
22. end if
23. *P_o is selected from P randomly*
24. *set Phase = 1*
25. end if

Then we will show how to prevent deadlock with additional VCs in above adaptive routing. Packets will travel an external super node in non-minimal routing, leading to 2 hops of inter-layer routing. So we set 2 VCs in egress ports which connect to nodes in different super nodes. For those egress ports that belong to intra-layer links, 2 VCs are enough because there are 2 hops at most within a super node. But the request rule of those VCs is restricted by the hop number within the super node, which is illustrated at Fig. 5. It will need more VCs if they are set as [7] where VCs are added according to the total hops that a packet will travel. There are 4 local hops and 2 global hops at most in RSI with 2D Butterfly (BR), and it will need 4 VCs in local egress ports if we do as [7]. So we use less VCs to prevent dead-locks but need more logics to record the hops information of routing.

V. FAULT-TOLERANCE MECHANISM

In order to cope with the possible problems such as node or link fault and man-made loading and unloading operations, we propose a fault- tolerance mechanism to update the routing information. Based on link state, the algorithm will generate fault messages or handle messages received from other nodes to update the routing table, and then the fault messages will be sent out to other nodes. When the failure link recovering or receiving recover messages from other nodes, the routing table will be recovered and updated by the algorithm. In this way, we can achieve the purpose of fault tolerance with little reduce of network performance. What we assume is that each port has a copy of two routing tables in Section IV to speed up the process of looking for them. Note that the router component should track and identify the availability of links, which is the premise of the algorithm. And the information and state of links can be obtained from some related registers of router. We define two types of packets used to error detection and recovery: FD (Fault discovery) and FR (Fault recovery). The failure detection and recovery algorithms are shown in Tables 4 and 5 below.

TABLE IV. FAILURE DETECT ALGORITHM

Algorithm 2. Failure detect algorithm

1. Inpu: FD packet because of the failure of port A; or FD packet from port A.
2. Output: Forward FD(Region) to other nodes; Updated routing table.
3.
4. *Receiving FD packet*
5. if *FD is generated by local* then
6. *Compute the region influeced by the failure*
7. end if
8. for *each port B* do
9. for *each entry C in routing table* do
10. if *C contains the Region* then
11. if *port A is not in C* then
12. *Pass*
13. else if *port A is in C* then
14. *Set A invalid in C*
15. end if
16. if *C contains Port A only* then
17. *Send FD(Region) out throug port B*
18. end if
19. end if
20. end for
21. end for

TABLE V. FAILURE REVOCER ALGORITHM

Algorithm 3. Failure recovery algorithm

```
1.  Input: Local port A recovers and FR is generated; or FD packet
    from port A.
2.  Output: Forward FD to other nodes;
    Updated routing table.
3.
4.  Status = 0
5.  Receiving FD packet
6.  for each port B do
7.    for each entry C in routing table do
8.      if port A is not in C then
9.        Pass
10.     else
11.       Set A valid in C
12.     end if
13.     if C contains Port A only then
14.       Set Status = 1
15.     end if
16.   end for
17.   for each routing entry D do
18.     if there is invalid port in D then
19.       Set Status = 0
20.       Break
21.     end if
22.   end for
23.   if Status == 1 then//only if all entries are //recovered
24.     Forward FR out through port B
25.   end if
26. end for
```

As illustrated in Fig. 6, a node in BR network fails and the links among the adjacent nodes and it are signed as dash lines. Then the adjacent nodes will detect the failure and generate a FD packet. After that, Algorithm 2 in Table 4 will be executed first in these adjacent nodes to update their routing tables. In line 9, the layer 2 and layer 1 routing tables illustrated in Fig. 4 will be detected respectively, which is same below in Algorithm 3. By examining each entry C in the routing tables of B, the algorithm checks whether the parameter Region belongs to address space of entry C or not. Once the fault region overlaps with the address area of entry C, the route options of entry C will be set invalid. And if C is the only one choice from B to A, the FD (Region) packet will be propagated to neighbors via port B. With many times of Algorithm 3, FD packets will then be forwarded to more and more nodes in the network gradually. When the FD packets are spread to other super nodes, the global links among the super nodes will be signed dashed meaning that there is some failure occurred in the area. Only if all nodes in a super node are failed, the corresponding entry in routing Table 2 of nodes in adjacent super nodes will be set invalid. Otherwise, there will be still many paths to reach the super node with some failure in it such as the case illustrated in Fig. 6.

In turn, if the failure node is recovered, FR packets will be generated adjacent nodes will receive FR to update the state of tables by Algorithm 3 in Table 5. In Algorithm 3, the variable Status is used to distinguish whether all the routing entries of current port are valid. If none route option is valid, the Status will be set 0 in line 19. And the recovery procedure about the relative fault Region will be terminated at current node. Only if all nodes are recovered in a super node, the corresponding routing tables will be updated as valid if the entry was set invalid before. Note that only when all entries are recovered and become valid, the FR will be forwarded to other nodes.

VI. EVALUATION

We use OMNETPP [20] to model and simulate the networks in which each node contains a router module and a processor module used to generate and consume packets. There are input queues, a routing component, an arbiter with maximum match logic and a crossbar in router module. We set the length of input queues large enough in order to simplify design. And the threshold in adaptive routing are implemented as a percentage of the length of queues. We model Dragonfly and three types of RSI with different layer 1 topologies in Fig. 3 layer 1 interconnection topology. First of all, we set the number of groups or super nodes to the same which is 8 and there is only one link between every two groups or super nodes. So there are 40, 128, 144 and 128 nodes in Dragonfly (DF), RSI with Hypercube (HR), RSI with Slim Fly (SR) and RSI with 2D Butterfly (BR), respectively.

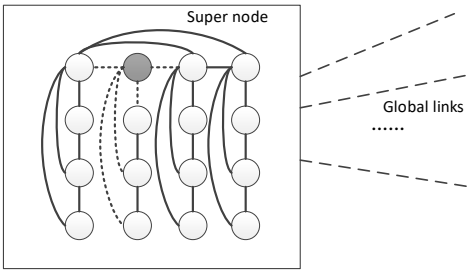

Fig. 6. Failure occurs and will be resolved by fault-tolerance mechanism.

The latency and throughput performance of four networks in uniform random traffic are shown in Fig. 7. We evaluate four networks with different injection rate (the horizontal axis) and minimal routing algorithm. It can be seen in Fig. 7 left that four networks have good delay performance at low level of load (injection rate below 0.3) but HR performs worse than the other three. Moreover, the saturation throughput of DF has reached 0.8 which is better than RSI networks because the average distance of nodes is lower than RSIs. We count the average number of hops from all the packets in the four networks from the source node to the destination node, and they are 2.5, 4.6, 4.0 and 3.7 in DF, HR, SR and BR respectively. And packets will travel 9 hops at most in HR which can explain the worse performance of HR.

What we mentioned before is that there could be more than one links between two super nodes when the network is small. Then we construct two types of BR networks with 3 super nodes: 1) only one link between every two super nodes; 2) there are 16 links between every two super nodes. And the comparison of two BR networks is illustrated in Fig. 8 from which we see that BF-2 outperforms BF-1 too much.

In addition, the network work load can be optimized by scheduling more traffic to be in the same super node to reduce the communication between super nodes. Fig. 9 illustrates the performance of four networks when traffics are distributed in the same super node called local performance. We see that DF performs best because nodes are fully interconnected in super nodes. And HR is going to be saturated first because there will be 4 hops at most between two nodes in Hypercube.

In fact, some congestion may occur in network. For example, when a virtual machine is going to be migrated from one node to another, data will continuously be transmitted in the link between two nodes. So it is necessary to balance the load between two nodes by using more links to share the traffic. In other words, the adaptive routing mechanism is vital in this pattern of traffic. And we evaluate the low cost adaptive routing algorithm in BR with hot spot traffic where nodes in a super node send packets to other nodes in another super node. There is only one link between these two super nodes so the network soon reaches saturation with minimal routing in Fig. 10. We find that the network outperforms much more in LAR with threshold at 0.6 than in minimal routing. The adaptive routing mechanism will improve the performance of the network in extreme traffics.

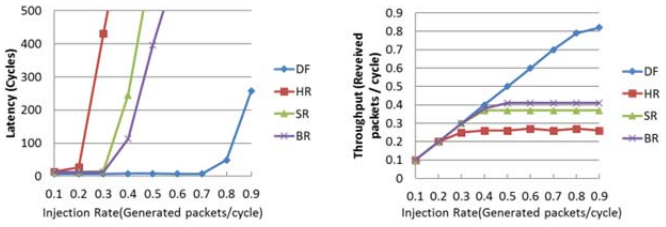

Fig. 7. Latency (left) and throughput (right) of four types of networks in Uniform Random traffic.

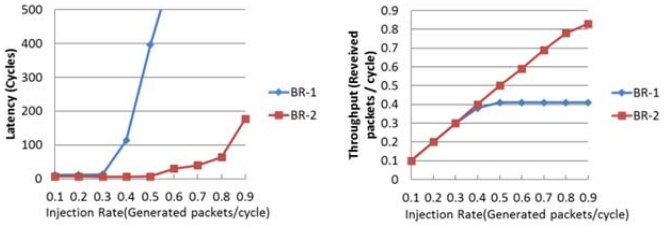

Fig. 8. Latency (left) and throughput (right) with two types of global interconnection ways in BR.

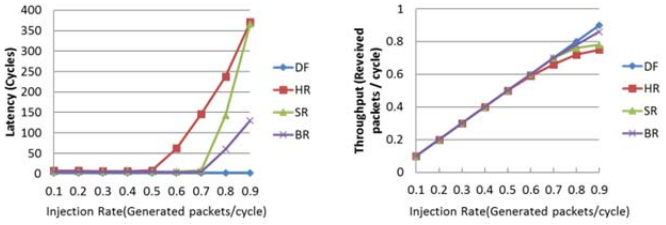

Fig. 9. Latency (left) and throughput (right) when traffics are distributed in the same super node.

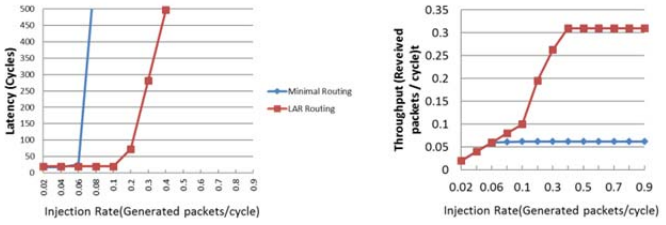

Fig. 10. Latency(left) and throughput (right) with minimal routing and LAR.

VII. Conclusion

We propose a new architecture for rack-scale design and a new type of interconnection network with limited number of pins. Current solutions such as bus and stars structures are difficult to interconnect all the nodes in a rack with high performance. And Dragonfly is hard to extend to larger scale with limited number of pins, although it performs well. RSI is hierarchical topology which can easily connect 512 to 1024 nodes in a rack. And we discuss different types of layer 1 topologies to form a super node. Then we propose a low cost adaptive routing algorithm with dead-lock free mechanism to solve extreme traffics. The fault-tolerance mechanism is proposed to deal with some practical situation but it need to be evaluated in the future. At last, we evaluate the networks in OMNETPP simulator to verify the performance of our propositions.

Acknowledgment

The authors would like to thank the anonymous reviewers for the feedback and revision suggestions. Then, we would thank National key research and development projects (2016YFB0200203) and NSFC (61572509) for providing the assistance to make this research possible.

References

[1] Dally, W., & Towles, B., "Principles and Practices of Interconnection Networks," Morgan Kaufmann Publishers Inc, 2003.

[2] Sokolov, D. A., "IDF 2015: intel zeigt virtuelle touchdisplays und flottere ssds mit 3d-xpoint-technik", online.

[3] Kim, H. J., Lee, Y. S., & Kim, J. S, "NVMeDirect: a user-space I/O framework for application-specific optimization on NVMe SSDs," In 8th USENIX Workshop on Hot Topics in Storage and File Systems (HotStorage 16), June. 2016.

[4] Hring, S. R., Ibel, M., Das, S. K., & Kumar, M. J., "On generalized fat trees," IEEE Computer Society, Vol.15, pp.37, 1995.

[5] Kim, J., Dally, W. J., Scott, S., & Abts, D, "Technology-driven, highly-scalable dragonfly topology," IEEE. Computer Society, Vol. 36, No. 3, pp. 77-88, June. 2008.

[6] Besta, M., & Hoefler, T, "Slim Fly: A cost effective low-diameter network topology," In High Performance Computing, Networking, Storage and Analysis, SC14: International Conference, IEEE, pp. 348-359, November. 2014.

[7] Gunther, K., "Prevention of deadlocks in packet-switched data transport systems," IEEE Transactions on Communications, vol. 4, pp. 512-524, 1981.

[8] García, M., Vallejo, E., Beivide, R., Odriozola, M., & Valero, M., "Efficient routing mechanisms for dragonfly networks," IEEE. In Parallel Processing (ICPP), 2013 42nd International Conference, pp. 582-592, October. 2013.

[9] "Intel Rack Scale Design," in http://www.intel.com/content/www/us/en/architecture-and-technology/rack-scale-design-overview.html, online.

[10] Lam, C. (2008, April), "Cell design considerations for phase change memory as a universal memory," IEEE. In VLSI Technology, Systems and Applications, pp. 132-133, April. 2008.

[11] Sun, G., Dong, X., Xie, Y., Li, J., & Chen, Y., "A novel architecture of the 3D stacked MRAM L2 cache for CMPs," IEEE. In High Performance Computer Architecture, pp. 239-249, February. 2009.

[12] Morris, R. J., & Truskowski, B. J., "The evolution of storage systems," IBM systems Journal, vol. 2, pp. 205-217, 2003.

[13] Glider, J. S., Fuente, C. F., & Scales, W. J., "The software architecture of a san storage control system," IBM Systems Journal, vol. 2, pp. 232-249, 2003.

[14] Gibson, G. A., & Van Meter, R., "Network attached storage architecture," Communications of the ACM, vol. 11, pp. 37-45, 2000.

[15] InfiniBand Trade Association, "InfiniBand Architecture Specification: Release 1.0," 2000.

[16] Adiga, N. R., Blumrich, M. A., Chen, D., Coteus, P., Gara, A., Giampapa, M. E., ... & Tsao, M., "Blue Gene/L torus interconnection network,". IBM Journal of Research and Development, vol. 3, pp. 265-276, 2005.

[17] Lee G, "Cloud Networking: Understanding Cloud-based Data Center Networks," Morgan Kaufmann Publishers Inc, 2014, pp. 191–203.

[18] Zhou, Wenhao, et al. "Detailed and clock-driven simulation for HPC interconnection network." Frontiers of Computer Science 10.5(2016):797-811.

[19] Zhengbin, et al. "The TH Express high performance interconnect networks." Frontiers of Computer Science 8.3(2014):357-366.

[20] "Omnetpp", refer to https://omnetpp.org/, online.