MATEC Web of Conferences **76**, 04040 (2016) DOI: 10.1051/matecconf/20167604040
CSCC 2016

Rack-Scale Storage Fabric: A Practical Way to Build Best-Fit Infrastructure for High-Performance Data Processing

Ruiquan Ding[1], Xiao Hu[2,a], Guofeng Chen[1], Zhiwen Xiao[1], Jiajun Zhang[1], Chao Liu[1], Jian Wang[2], Huan Zhou[2] and Jun Zhang[2]

[1]Baidu. No. 10, Xibeiwang Road East, Beijing, PRC. +86-10-5679-5465
[2]Intel APAC R&D Co. Ltd. No. 880, Zixing Road, Shanghai, PRC. +86-21-6116-5000

Abstract. This paper is to address the resource utilization problem for high-performance data processing applications in a large IDC (Internet Data Center) environment. On one hand, each application calls for a best-fit infrastructure with a specific compute-storage ratio, to achieve the highest resource utilization while meeting its performance requirement. And such a ratio varies among applications. On the other hand, IDCs have always been trying to unify the infrastructures for lower TCO (Total Cost of Ownership). Therefore, it's getting harder and harder to adapt infrastructures to application needs. This issue results in significant waste of investment in large IDCs. Furthermore, the high-performance data processing applications always require the infrastructure to offer as high compute-storage performance as a DAS (Direct Attached Storage) server, which remains as a great challenge when addressing the resource utilization problem. This paper, as part of Baidu-Intel joint research program, first evaluates the state-of-the-art solutions, and then introduces a more practical infrastructure, the core of which is *rack-scale storage fabric*. This infrastructure disaggregates compute units and storage units by a SAS (Serial Attached SCSI) fabric, and allows to compose *logical servers* with arbitrary computer-storage ratios within a rack. And the experiments in Baidu's research environment show that the logical servers exhibit the similar throughput/IOPS as DAS servers, and also their compute-storage ratios can best-fit the needs of different Hadoop applications.

1 Introduction

Different applications often exhibit different resource demands. And even one application would have different resource demands at different phases of its lifecycle. The resource demand of an application is majorly represented by a ratio of its needs for compute resource (e.g. CPU and memory) to its needs for storage resource (e.g. disks), called *compute-storage ratio*. In another words, if the infrastructure that hosts a specific application matched the compute-storage ratio of this application, the application would be able to fully utilize the resources of this infrastructure. Otherwise, the infrastructure would be underutilized, which, in large-scale IDC environments, would result in significant waste of investment.

On the other hand, there is a solid trend that the IDC users keep simplifying their infrastructures, in order to reduce TCO (Total Cost of Ownership), because fewer types of infrastructures would lead to lower price and easier maintenance. Recently such standardizations, like OCP [5] and Scorpio [6], even intend to create unified and standardized infrastructures across the industry. As a result, the tension between the diversification of applications and the simplification of infrastructures keep increasing, which makes it impossible to find a best-fit infrastructure for each specific application, and so very often there are resources not fully utilized. In this context, the industry have been researching and developing many solutions to improve IDC resource utilizations, such as Mesos [4], OpenStack [8], flat datacenter storage [7], and Intel Rack Scale Architecture [10].

Moreover, a lot of IDC applications are distributed and high-performance data processing software, which require high throughput and low latency for the compute to access the data in the storage. For this sake, those applications have been often designed as being aware of the data locality, i.e. to process a piece of data right on the server where it is stored. In another words, the performance relies on a condition that the connection between each compute and each corresponding storage would be as fast as that of a normal DAS server, which has disks closely attached to CPUs via PCIe and/or SAS technologies. For instance, each SATA disk should offer about 150MB/s sequential read throughput to the corresponding CPU. This applies to many popular data processing frameworks, like Hadoop, Spark, MPP, and so on. Therefore, improving the resource utilization for those distributed high-performance data processing applications must always retain the above condition. Otherwise, it would significantly compromise the performance.

[a] Corresponding author: albert.hu@intel.com

This paper is organized as follows. First, Section 2 evaluates the state-of-the-art solutions. Next, Section 3 introduces a rack-scale storage fabric solution and how it is implemented at Baidu, from both hardware and software perspectives. Following that, Section 4 provides a cost comparison to other network based solutions as well as to normal DAS server. Section 5 first records a unit test for throughput/IOPS, proving storage fabric retains as high compute-storage performance as a normal DAS server, and, second, shows solution benchmarks, demonstrating this solution can build the best-fit infrastructures for different Hadoop applications. At last, Section 6 concludes the paper.

2 Related Work

Virtualization [8] or container [4] technologies are widely used to collocate different applications on each of the normal servers, hoping that the composite compute-storage ratio of multiple applications would match the server's compute-storage ratio. However, these solutions encounter the following problems in practice.

First, the introduction of virtualization or container increased the software complexity and cost. Second, for virtual machine based solutions, the hypervisors will cause certain performance overheads and increase maintenance cost as well. Third, more importantly, the number of virtual machines or containers on each server has to be limited in a production environment due to application's complexity and scheduling overheads. Thus, usually, one server can only host two or three virtual machines or containers of the corresponding distributed applications. Unfortunately, it'll be very hard for so few applications to compose a desired compute-storage ratio that matches the underlying infrastructure.

Second, using traditional networked storage service, e.g. NAS, SAN, etc., to disaggregate the storage from the compute is the most common way to address resource utilization problems. This method offers the maximum flexibility, because, in theory, one can compose any compute unit and any storage unit from anywhere in an IDC. However, it is not suitable for high-performance data processing applications. The top problem is performance limitation: (1) Traditional network structure can never offer as good compute-storage performance as that in a normal DAS server in a scalable way. (2) The networked storage services have to introduce quite a few software stacks in between compute and storage, which would certainly cause overheads and downgrade the compute-storage performance.

Third, in order to relief the performance limitation of traditional network based disaggregation, [7] used CLOS network to seek high compute-storage performance. Though CLOS network can theoretically offer a full bisectional bandwidth, this method still has a few road blocks in implementation. First of all, changing the IDC network will be much more complicated than changing the servers. Since most of the IDC networks are never designed as CLOS, changing traditional network to CLOS will be too expensive and too time-consuming to be practicable. Second, CLOS network actually has performance penalties, and hence it cannot achieve its ideal performance. For example, no perfect load-balance that can ensure full utilization of all links, the TCP in cast issue getting worse [7], etc. Moreover, as [7] does, in order to achieve such an IDC-scale disaggregation, there must be a powerful and complicated management system to provide the necessary runtime services. Such a complicated system must take time and efforts to develop and maintain, which will definitely increase the TCO.

3 Rack-scale storage fabric

This section will first define a generic *zone-scale storage fabric* architecture, and then describe the implementations in Baidu's specific environment.

3.1. A Generic Architecture

The overall system architecture is illustrated in Figure 1. All compute and storage resources are divided into many zones. In each zone, there are a pool of compute units and a pool of storage units, plus a storage fabric in between these two pools. Each upstream port (USP) of the storage fabric is attached to one compute unit, and each downstream port (DSP) of the storage fabric is attached to one storage unit.

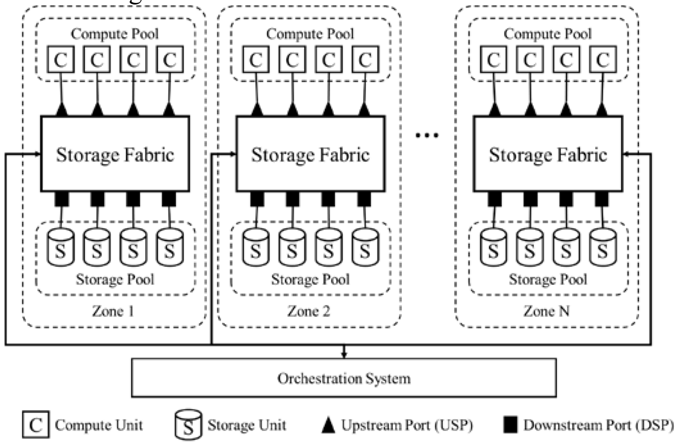

Figure 1. A Generic Zone-Scale Storage Fabric Architecture

First of all, the storage fabric allows users, via an orchestration system, to configure its internal links between USPs and DSPs. Once a USP and a DSP is configured to be linked up, the corresponding compute unit and storage unit are composed together, and then the software on the compute unit can access the data in the storage unit.

A compute unit can be composed with multiple storage units, and then makes up a new system, called *logical server*. In each zone, one can compose one or multiple logical servers to host a specific application, then all such logical servers across multiple zones make up the infrastructure for this application. Thus, if all the logical servers are composed with a compute-storage ratio that matches the application's compute-storage ratio, the whole infrastructure will best-fit this application.

Besides, each compute unit should be able to boot and execute a software environment independently from the storage units. Also, the storage protocols must support to hot-insert/remove a storage unit to/from a compute unit, without disrupting the software environment on that compute unit. In this way, adjusting the infrastructure will have minimum impact to the applications.

3.2 Implementation

This section will introduce our implementation for the above zone-scale storage fabric architecture, called *rack-scale storage fabric* solution, specifically fitting Baidu's business environment. This section will first explain the key requirements that drive our designs, and then describe the details of our prototype system that has been deployed and tested in Baidu research environment.

3.2.1 Requirements in Baidu's Environment

In Baidu's environment, a compute unit corresponds to a *server node*, which majorly consists of one or two CPUs, memory, network interfaces, as well as a local or remote disk installed with OS and applications. On the other hand, a storage unit corresponds to a *disk*, e.g. a SATA HDD (hard disk), or a SATA SSD (solid state disk).

The profiles of Baidu's high-performance data processing applications indicate the following requirements.

Requirement 1. A server node may need to access at max. 20 SATA HDDs, or, equivalently, 5 SATA SSDs. This translates to 30~40Gbps throughput and ~200,000 IOPS that a logical server would perform at maximum.

Requirement 2. Statistically, each server node may need to access in average 10 SATA HDDs, or, equivalently, 2 SATA SSDs. That's to say, the average compute-storage ratio of each zone should be equal to 1:10 (HDD) or 1:2 (SSD).

Requirement 3. Baidu's profiles tell that usually collocating 5 or more applications together would easily get any desired composite compute-storage ratio, e.g. 1:10 (HDD). This means there must be enough server nodes in each zone, so that at least 5 or more applications can be collocated there at the same time.

There're two more constraints when implementing the solution at Baidu.

Requirement 4. It's very much desirable to keep the IDC network infrastructure unchanged.

Requirement 5. The form of the rack has to be retained, because of Baidu's strategic investments in Scorpio [6] and the large amount of Scorpio-compliant racks been already deployed.

3.2.2 Hardware Configurations

Figure 2 is a photo of one rack of our prototype system that has been deployed and tested in Baidu's research environment. In this implementation, each rack contains 2 zones. And each zone contains 8 server nodes, so that the number of compute units in each zone is more than 5. In another words, **Requirement 3** has been met.

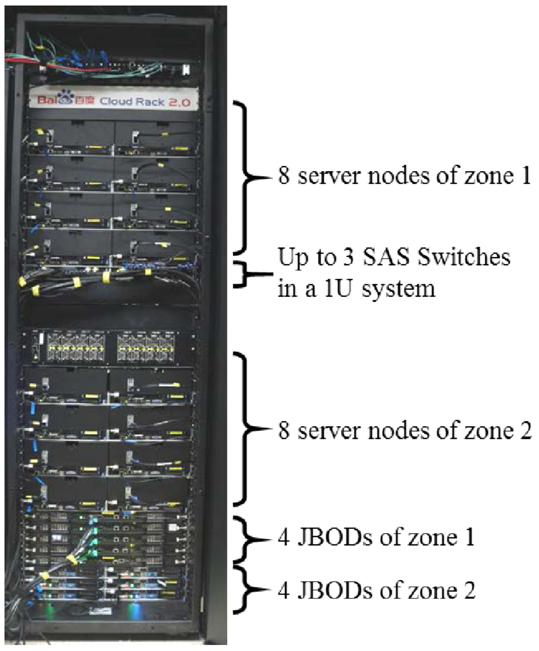

Figure 2. A Full-Rack Implementation at Baidu

Each zone also has 4 SAS JBODs. Each JBOD can be populated with max 20 SATA disks (both HDDs and SSDs can be used there). So, in HDD case, the compute-storage ratio of each zone is 8 server nodes vs. 80 HDDs, i.e. 1:10. In this way, **Requirement 2** has been met, too.

Each zone contains a SAS switch. Each server node has a SAS HBA (Host Bus Adapter), and each JBOD has a SAS expander. SAS cables are used to connect HBAs and expanders to SAS switch. In each zone, the SAS switch and the SAS expanders together make up a storage fabric. The USPs of this storage fabric are those SAS switch ports connected to server node's HBAs, and the DSPs are those SAS expander ports connected to disks.

In our prototype, server node is specially designed as 3U height, in order to leave enough space to test different vendors' HBAs. The current SAS HBA has two 48Gbps SFF8644 ports (x4 SAS 12G each), one of which is being used in our prototype to connect to SAS switch. So, the USP bandwidth (48Gbps) of each server node is designed to be enough to access 20 HDDs.

At the other end of the SAS switch, there are SAS JBODs. The embedded SAS expander in each JBOD gives each of the 20 disks a 12Gbps SAS port, which provides enough bandwidth for even a SSD. Connected to SAS switch are up to four 48Gbps SFF8644 ports (x4 SAS 12G each), two of which in this prototype is being used. So, the outstanding bandwidth (96Gbps) of each JBOD is more than enough to serve 20 enterprise SATA HDDs, and also enough to serve up to 10 SSDs. And, internally, the SAS expander provides a full-bandwidth fan-out from the four SFF8644 ports to the 20 disks.

Figure 3 is the prototype of our SAS switch system. It contains three separated SAS switches in a 1U height chassis, one for each zone, plus one network bridge board for remote management purpose. In this prototype, since each rack just contains two zones, only two SAS switches in that 1U system are being used.

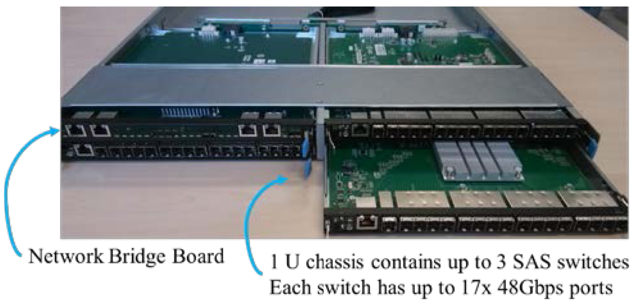
Network Bridge Board
1 U chassis contains up to 3 SAS switches
Each switch has up to 17x 48Gbps ports

Figure 3. SAS Switch System

The current SAS switch prototype has totally 17 48Gbps SFF8644 ports (x4 SAS 12G each). In this prototype, 8 ports are connected to the 8 HBA ports on the 8 server nodes, and another 8 ports are connected to the 8 ports on the 4 SAS JBODs. The SAS switch can provide a full-bandwidth switching.

With all above designs, there is no bottleneck for a server node to access the max 20 HDDs, because the outstanding bandwidth of each HBA and each JBOD are enough, and the SAS switch and SAS expander are all full-bandwidth switching. Therefore, **Requirement 1** has been met. Last, this prototype doesn't need to change either IDC's network (**Requirement 4**) or rack infrastructures (**Requirement 5**). So, all the aforementioned requirements have been met, and this implementation can fit into Baidu's environment.

3.2.3 Orchestration Mechanism

Another key component of our implementation is an orchestration system that offers IDC operators with a semantics to identify any specific compute unit (i.e. server node) and any specific storage unit (i.e. HDD, SSD), determines whether they can be composed together through a storage fabric, and then does the composition by telling the storage fabric to link up the corresponding USP and DSP. There are two problems that an orchestration system needs to be implemented to solve.

(1) One problem is how to dynamically configure the storage fabric, so that the links between its USPs and DSPs can be changed, and these changes won't disrupt the software environments on the compute units.

For this, on one hand, SAS protocol has been working very well on disk hot-plug/unplug, i.e. doing the logical server composition/de-composition wouldn't disrupt the software environments on logical servers. On the other hand, in order for orchestration system to manipulate the compute-storage composition, the SAS switch has been developed to expose a set of remote management APIs, following Intel Rack Scale Architecture definitions [10]. By calling these APIs, our orchestration system can easily manipulate the links between USPs and DSPs.

(2) The other problem is how to obtain the correspondences between each server node and its attached USP, and the correspondences between each disk and its attached DSPs, regardless the states (e.g. powered on or off) of server node and disks.

First, the orchestration system identifies each server node according to the descriptive information, e.g. serial numbers, etc., that is provided by the BMC (Baseboard Management Controller) on the server node. When a server node is first-time added to a rack, it is required to power-on once, during which its BIOS (Basic Input/Output System) can fetch the SAS addresses of its HBA ports and persistently store them to its BMC, and then the BMC sends this information to the orchestration system. In this way, the orchestration system can get a mapping between each server node and the SAS addresses of its HBA ports.

On the other side, the SAS switch can also record the SAS addresses of the HBA ports that are physically connected to its USPs. Then, through management APIs, the orchestration system can get another mapping between each USP and the SAS addresses of the HBA ports that are connected to the USP.

With the above two mapping information, the orchestration system can corresponds each server node to its attached USPs.

Second, the SAS switch itself can retrieve both the descriptive information (e.g. serial number, etc.) and the SAS addresses of each attached disk, then come up with a correspondence between each disk and its attached DSP. Then, the orchestration system can get this correspondences via SAS switch management APIs.

With these, when IDC operator wants to compose a specific server node with a specific disk, the orchestration system will be able to tell that it should be which USP and which DSP of SAS fabric that should be linked up.

4 Cost Analysis

4.1 BOM Cost Comparison

This section will first compare BOM (Bill of Material) costs among a CLOS network based disaggregation (see Section 2) and two rack-scale disaggregation solutions. One is based on intra-rack network, which can be implemented to achieve similar performance to our solution. The other is our solution. See Figure 4.

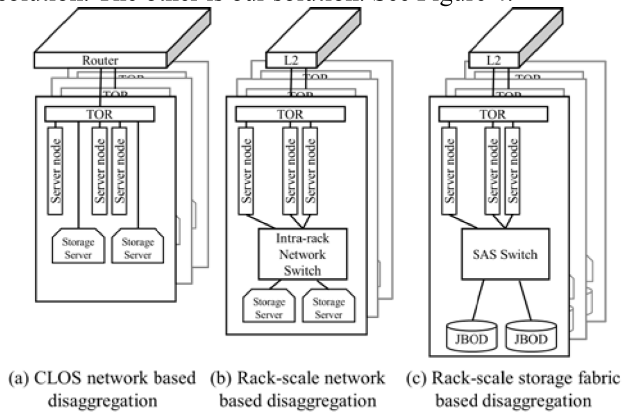
Router
TOR
Server node
Server node
Server node
Storage Server
Storage Server
L2
TOR
Intra-rack Network Switch
L2
TOR
SAS Switch
JBOD
JBOD
(a) CLOS network based disaggregation
(b) Rack-scale network based disaggregation
(c) Rack-scale storage fabric based disaggregation

Figure 4. Three Configuration for BOM Cost comparison

Config. 1. A simple CLOS network based disaggregation. As illustrated in Figure (a), the network consists of 2 layers of switches. One is TOR (Top of the Rack) switches, and the other is Router switches [7]. Both compute-storage and compute-compute traffics go through this single network.

Config. 2. A simple intra-rack network based disaggregation. As illustrated in Figure (b), there are two sets of networks. The first set includes a single 2-layer network for the normal compute-compute traffics across the racks, which consists of a layer of TOR switches and a layer of L2 switches. The second set includes many separated intra-rack networks, one for each rack, to support the compute-storage traffics within each rack.

Config. 3. Our rack-scale storage fabric based disaggregation. As shown in Figure (c), it is similar to Config. 2, except two major differences. First, storage server changes to SAS JBOD. Second, each intra-rack network for compute-storage traffics is replaced by a SAS switch.

For clarity, let's define the following denotations.

B*:* the total bandwidth needed by compute-storage traffics. In fact, since the traffics at compute side and the traffics at storage side are symmetric, the total outstanding bandwidth of all compute units and the total outstanding bandwidth of all storage units equal to *(B/2).*

k·B*:* the total bandwidth needed by the original compute-compute traffics, where ***k*** is a ratio of this bandwidth to the compute-storage one.

Px*: a* unit cost for ingredient x to offer a unit of bandwidth. E.g. P_{TOR} means unit cost of TOR switch.

h: the convergence rate of multi-layer switches, in terms of bandwidth. $0 < h \leq 1$. This only applies to the first set of networks, i.e. for compute-compute traffics, in Config. 2 and 3.

For simplicity, let's assume all types of network switches have the similar unit prices, i.e. $P_{router} = P_{L2} = P_{TOR} = P_{intra\_rack\_net\_switch}$. Also, without impacting the sufficiency, one can set $h = 1$ for Config. 2 and 3, i.e. there no convergence from TORs to L2 switches, which is the most expensive configuration for both Config. 2 and 3. Then, even in this case, we can first calculate the total cost difference between Config. 1 and 2 as below.

$$\mathrm{Diff}_{(1-2)} = B\cdot(P_{router} + P_{net\_cable} + 2\cdot P_{TOR} - P_{intra\_rack\_net\_switch})$$

Mathematically, this is always a positive value, because we assume the unit prices of all kinds of network switches are similar. This formula actually means Config. 1 has to spend more money building an IDC-scale CLOS network for high-performance compute-storage traffics, than Config. 2 that only needs to build many small-scale networks within each rack.

Then, we can calculate the difference between Config. 2 and 3, as below.

$$\begin{aligned}\mathrm{Diff}_{(2\text{-}3)} = (B/2)\cdot\{&[2\cdot(P_{net\_cable} - P_{SAS\_cable}) + \\ &2\cdot(P_{intra\_rack\_net\_switch} - P_{SAS\_switch}) + \\ &(P_{NIC} - P_{SAS\_HBA})] + \\ &(P_{storage\_server} + P_{NIC} - P_{JBOD})\}\end{aligned}$$

Mathematically, the first part of this formula includes 3 unit price differences, i.e. (1) between network cables (P_{net_cable}) and SAS cables (P_{SAS_cable}), (2) between the intra-rack network switch ($P_{intra\_rack\_net\_switch}$) and the SAS switch ($P_{SAS\_switch}$), and (3) between the NICs (P_{NIC}) and SAS HBAs (P_{SAS_HBA}) on the server nodes. By investigating the SAS 12G products and the 40Gbps network products in the marketplaces, it was concluded that all these three differences are actually trivial, and the SAS products even have cost advantages in some cases. On the other hand, this formula has a second part, which is the cost difference between storage servers and their NICs in Config. 2 and the SAS JBODs in Config. 3. This part is definitely a non-trivial positive value, because even the lowest-end storage server, which consists of CPU, memory, motherboards, etc., must be more costly than a JBOD, which is basically a SAS expander.

So, summing up the two parts, the difference is always a positive value, which means Config. 3 is cheaper than Config. 2. Actually, the root causes are, first, with the same performance (12G SAS vs. 40Gbps network), the network products have no cost advantage over SAS products, and, second, an intra-rack network based disaggregation always needs to use expensive storage servers, while rack-scale storage fabric only needs to use the cheaper SAS JBODs.

4.2 Cost Adder for Storage Fabric

For ROI (Return of Investment) analysis, it has been estimated how much extra investment would need be added, to enable our storage fabric based disaggregation, comparing to an original infrastructure that is just made up by normal DAS servers.

Without exposing Baidu's pricing data, Table transforms the absolute costs to percentages. You can see the total BOM increase would be less than 5%. Note that this only compares to normal servers. So, if counting in other ingredients in an original infrastructures, e.g. racks, which are basically the same between two options, this percentage will be even smaller.

Table 1. BOM Cost Adder (%) to Enable Storage Fabric

	SAS HBA	**SAS cables**	**SAS switch**	**JBOD**
Additional cost amortized to each server node (%)	2.57%	0.51%	1.03%	0.43%
Total (%)	4.543%			

On the other hand, based on Baidu's current resource utilization profile for high-performance data processing applications, it is very promising that the cost saving out of the higher resource utilization would be fairly enough to justify this small percentage of extra BOM investment.

5 Experimental Results

5.1 Unit Tests

The key advantage of storage fabric is to retain as high performance as a DAS server. Though the hardware configuration was designed to provide a sufficient bandwidth (see Section 3.2.2), quantitative performance testing are necessary to verify that our rack-scale storage fabric not cause any performance loss.

To verify the effect of SAS fabric on performance of a logical server, a server node (with a 48Gbps HBA port) is composed with 20 HDDs. Then we measured the throughput/IOPS of this logical server. As a fair comparison, we also tested a DAS server using the same type and number of server node and HDDs.

Table 2 shows the logical server performs very similarly to a 20-HDD DAS server. Also, note, per HDD's specification, which only discloses 160 MB/s sequential read peak bandwidth, our testing is valid.

Table 2. Performance comparison between logical server and DAS server

	Logical server Throughput /IOPS	**DAS server Throughput /IOPS**
Random read 4K	1,753 IOPS	1,700 IOPS
Random read 256K	2,535 IOPS	1,500 IOPS
Random write 4K	3,581 IOPS	3,760 IOPS
Random write 256K	3,730 IOPS	2,840 IOPS
Sequential read 4K	1,554 MB/s	1,297 MB/s
Sequential read 256K	3,210 MB/s	3,202 MB/s
Sequential write 4K	1,336 MB/s	1,170 MB/s
Sequential write 256K	2,563 MB/s	3,112 MB/s

5.2 Solution Benchmarks

In order to quantify the system-level effects, we deployed Hadoop benchmarks on our prototype system to simulate high-performance data processing applications.

We used BigBench [3] benchmark tool suite to simulate a real-life business scenario, where a retailer company wants to analyze a large amount of its sales data and figure out valuable information from them. This benchmark standardizes a series of queries, and implements them as a HIVE [9] application running on MapReduce [2] engine and HDFS [1] storage. It will measure how many queries an application can execute every hour, called queries per hour (QPH), to indicate if the application can meet a performance requirement.

On our prototype, one can compose different types of logical servers, in terms of compute-storage ratio, to best-fit different BigBench applications that have different performance requirements. We tested 7 BigBench applications. Their datasets are of the same size, but each application has a different performance requirement, ranging from 2 QPH to 5 QPH. Table 3 records the actual performances, as well as the different infrastructure configurations, of these 7 applications. From these test results, it can be concluded that each application can not only meet their performance requirement but also very well utilized the resources.

Table 3. BigBench Applications on Best-Fit Infrastructures

App No.	**Expected QPH**	**#of server nodes**	**# of hard disks**	**Compute -storage ratio**	**Actual QPH**
1	2	4	64	1 : 16	**1.92**
2	2.5	6	64	1 : 11	**2.67**
3	3	8	64	1 : 8	**3.36**
4	3.5	10	64	1 : 6.4	**3.72**
5	4	12	64	1 : 5.3	**4.23**
6	4.5	14	64	1 : 4.6	**4.96**
7	5	16	64	1 : 4	**5.55**

6 Summary

This paper introduced a *rack-scale storage fabric* solution to address resource utilization problem. The implementation uses SAS switch and SAS expanders to make up a storage fabric to disaggregate compute unit and storage unit, and an orchestration system was developed to configure underlying switch fabric. Our practical experience paves the way to build shareable data center infrastructure for high performance data processing applications with efficient resource utilization.

Our solution demonstrated many advantages. First, our *rack-scale storage fabric* accelerates new application deployment regardless of the limitation of underlying infrastructure, as logical server can be composed easily from compute resource pool and storage resource pool.

Second, compute resource and storage resource are separated from a normal server, each can independently upgrade depending on which resource in shortage. Plus, our orchestration system also provides a rack-wide view of resource discovery and management capability to help relieve labor of data center operators.

Third, all the components are final products, except that SAS switches are very close to production. Building such a system will not be limited by technique challenge or business ecosystem.

REFERENCES

[1] Borthakur D. The Hadoop Distributed File System. Hadoop Project Website. (2007).

[2] Dean J, Ghemawat S. MapReduce: Simplified Data Processing on Large Clusters. *Communications of the ACM*. **51**, pp: 107-113(2008).

[3] Ghazal A, Rabl T, Hu M, Raab F, Poess M, Crolotte A, Jacobsen HA. BigBench: Towards an Industry Standard Benchmark for Big Data Analytics. *In Proceedings of the 2013 ACM SIGMOD international conference on Management of Data.* pp: 1197-1208 (2013).

[4] Hindman B, Konwinski A, Zaharia M, Ghodsi A, Joseph AD, Katz RH, Shenker S, Stoica I. Mesos: A Platform for Fine-Grained Resource Sharing in the Data Center. In NSDI, **11 (**2011).

[5] http://www.opencompute.org/

[6] http://www.opendatacenter.cn/

[7] Nightingale EB, Elson J, Fan J, Hofmann O, Howell J, Suzue Y. Flat Datacenter Storage. *10th USENIX Symposium on Operating Systems Design and Implementation,* pp: 1-15 (2012).

[8] Sefraoui O, Aissaoui M, Eleuldj M. OpenStack: Toward an Open-Source Solution for Cloud Computing. *International Journal of Computer Applications*. **55** (2012).

[9] Thusoo A, Sarma JS, Jain N, Shao Z, Chakka P, Anthony S, Liu H, Wyckoff P, Murthy R. Hive: A Warehousing Solution Over A Map-Reduce Framework. *In Proceedings of the VLDB Endowment*. pp: 1626-1629 (2009).

[10] Kyathsandra, Jay, and Eric Dahlen. "Intel rack scale architecture overview ", *INTEROP,* (2013).