A DHT Key-Value Storage System with Carrier Grade Performance

Guangyu Shi[1], Jian Chen[1], Hao Gong[1], Lingyuan Fan[1], Haiqiang Xue[2], Qingming Lu[1], and Liang Liang[1]

[1] Huawei Technologies Co., Ltd
Shenzhen, 518129, China
{shiguangyu,jchen,haogong}@huawei.com
[2] China Mobile Communications Co.
Beijing, 100053, China
xuehaiqiang@chinamobile.com

Abstract. The Peer-to-Peer (P2P) technology being widely adopted in today's both academic research and practical service providing, has many potential advantages and achieves a great success in Information Technology scope. Recently some researchers have proposed that P2P inspired architecture also might be one choice for the telecom network evolution. Most of such works adopted structured P2P (DHT) as the basic solutions, but they seldom discussed how to eliminate the huge gap between the telecom underlay performance requirement and the performance of existed DHT which mainly originated from the Internet applications. This paper presents the design and implementation of SandStone, a DHT based key-value storage system with carrier grade performance, such as good scalability, strong consistency and high reliability, which could be deployed as the cornerstone in such new P2P inspired networking architectures.

Keywords: Peer-to-Peer, DHT, Telecom Network, Key-Value Storage.

1 Introduction

Reliability and efficiency at massive scale is one of the important challenges in telecom applications, even the slightest outage or congestion has significant financial consequences and impacts customer experience. Although most of the current telecom networks were strictly central server based, many researchers have proposed a new networking architecture inspired by the P2P paradigm, in order to benefit from the advantages of decentralization such as high scalability and cost-effectiveness.

In the last few years, a number of systems and prototypes have been proposed to incorporate P2P technology into the field of telecommunications. For example, P2PSIP [1] proposed a DHT based conversational signaling system. Reference [2] and [3] presented two kinds of distributed IP Multimedia Subsystem (IMS) architectures, utilizing the DHT as the basic register/lookup functional entity, contrasted with central servers based on present IMS HSS/HLR. It's evident that methodology behind these works is to push P2P into the telecom underlay application, although P2P technology emerged as an Internet overlay technology.

H. Sips, D. Epema, and H.-X. Lin (Eds.): Euro-Par 2009, LNCS 5704, pp. 361–374, 2009.

On the other hand, P2P Distributed Storage Systems have been developed by leaps and bounds in recent years; typical examples include GFS [4], OceanStore [5], Pond [6], BitVault [7], Ceph [8], Dynamo [9], and WheelFS [10]. In the principles of these systems, the authors have a unified view that structured P2P or DHT should be used as the basic key-value storage system infrastructure. Nevertheless, in comparison with the Internet overlay applications, apparently telecom applications have a much stricter performance requirement for real-time response, reliability and consistency. Unfortunately, so far as we know, none of them has seriously considered the question that how to enhance DHT technology to face such performance requirements challenges.

In this paper, we present the design and implementation of SandStone, a highly decentralized, loosely coupled, service oriented storage system architecture consisting of thousands of PC peers. The name "SandStone" stands for transforming the enormous sand-like tiny and common PCs' capability into an invincible infrastructure cornerstone of distributed storage systems, to provide a "Carrier Grade Storage" experience. It means, the system is tested and engineered to meet or exceed "five nines" 99.999% high availability standards, and provide very fast fault recovery through redundancy.

The rest of this paper is organized as follows. Section 2 describes background requirements of SandStone. Section 3 elaborates the detail design and section 4 presents the implementation. Section 5 details the experiments and insights gained by running SandStone in production. And we summarize the conclusion in Section 6.

2 Background

2.1 Scenario

In telecom network, there are a large number of subscriber profile data in a centralized way to store, such as IMPI/IMPU (IP Multimedia Private Identity/Public Identity) in IMS and CDR (Call Detail Record) in BSS (Billing Support System). Originally, SandStone was designed to implement basic IMS HSS (Home Subscriber Server) functional entity in a decentralized and self-organized way. As works going on, we found that SandStone holds much more potential for being used as a key-value storage system in most of the P2P telecom application scenarios referred in section 1. For the sake of clarity, we'll choose the IMS HSS scenario as the application context.

According to 3GPP [11] specifications, HSS is the repository hosting the subscription related user data to support the other network entities such as CSCF (Call Session Control Function) to handle calls or sessions. Nowadays all the existing IMS HSS are centralized based. Typically it is hosted on expensive ATCA (Advanced Telecom Computing Architecture) or commercial super servers demanding a high robustness. The current HSS architecture works well on the deployment. Nevertheless, there are still some drawbacks other than the huge CAPEX (Capital Expenditure) and OPEX (Operating Expense). Firstly, central server is not a solution scaling well. Typically commercial HSSes are fully equipped nodes that can handle and store the data pertaining to a give maximum number of subscribers. Should the number of subscribers exceed the maximum limit, the operator is forced to deploy new HSS units that can handle the new subscriber data. Another question is the congested and overloaded servers can not be mitigated even there are spare resource somewhere else in the network. These drawbacks seriously hindered the development and deployment of IMS.

It is necessary to make some evolutions to adapt to the development trend in current network architectures. Although the distributed DHT based storage system seems quite capable to deal with these problems, the only doubt is whether it can achieve the same carrier grade performance goals as well as the central server based solutions.

2.2 Carrier Grade Objectives

According to current performance of HSS, SandStone for carrier grade of services has the following requirements:

Cost performance: Considering the saving of CAPEX and OPEX, SandStone is implemented on top of an infrastructure of tens of thousands of common computers located in many datacenters around the country. These common computers are very easy to maintain, add and replace than current ATCA servers.

Application Model: simple read and write operations to a data item that is uniquely identified by a key in SandStone. Every one million user corresponds to 1000 read and 100 write requests per second according to current application model of HSS. In other words, each peer in SandStone needs to support 10 read and 1 write requests per second at least in normal application model.

Scalability: the scalability in a self organized way, SandStone should be easily scaled up to handle 1 billion subscriber data. According to current HSS architecture, SandStone targets applications that need to store subscriber data that are relatively small. Each subscriber data should have a profile with 128K bytes, and tens of index mappings at the size of 128 bytes.

Reliability: Reliability is one of the most important requirements because even the slightest outage and congestion has significant financial consequences and impacts customers' experience. There are always a small but significant number of hosts that are failing at any given time. As such SandStone needs to be constructed in a manner that treats failure handling as the normal case without impacting availability or performance. The reliability objective is specified as "five nines" in telecom scopes.

Efficiency: For the best customers' quality of experience, any client operation should get the final response in 300 milliseconds, so as to limit the DHT lookup hops must be little with the presumption that SandStone will be deployed in a private network assuring 50 milliseconds is the maximum latency for any host-to-host pairs.

2.3 Failure Models

In a distributed system, the peer failure is a very common phenomenon. Peer's and network's failure models have an essential impact on the distributed storage systems, but this topic was rarely mentioned in above P2P inspired telecom research works.

We consider three failure models, including average failure (AF) and simultaneous failure (SBF1 and SBF2), respectively represents independent single computer fault, site/rack failure and inaccessible region due to IP backbone network break down.

Choosing exponential life distribution for a computer ($f(t)=\lambda e^{-\lambda t}$), we can get the failure rate as $\lambda(t)$ =1/MTBF (Mean time between failure). From China present

mandatory standard, the personal common computer must have a MTBF no less than 4000 hours. We choose 1000 hours for the safe margin, this means, each computer has the probability of 0.001 to failure in one hour. Taking N_{pc} as the number of total computers in the DHT overlay network, then AF model will result in N_{pc}/1000 computers failure in every hour.

The site/rack failure, SBF1, is usually caused by power system blackouts or other unknown reasons in a site. It happens quite rarely in a telecom scene. We still choose exponential distribution to model it, with 20000 as the MTBF. In the actual deployment, it's up to 50 hosts in a site/rack, so SBF1 exhibits the scenario that 50 computers will get out of service simultaneously in each period of 20000*50/N_{pc} hours.

Finally, it is hard to estimate the probability distribution for SBF2. For example, maybe the backbones between regions are being destroyed by unexpected tornados or earthquakes. We transformed it into a design requirement; SandStone has to sustain one of the largest regions inaccessible. The "largest" means it may contain all computers of a certain region. After overcome these failure models above, this distributed storage system truly could be regarded as a carrier grade system.

3 The Design of SandStone

In this paper, we presented SandStone, a novel DHT key-value storage system with carrier grade performance which could be applied to telecom network architecture such as IMS. The main contributions of SandStone are as follows: a multi-layer DHT architecture that decreases the traffic overload in backbone network; a enhanced one-hop routing table that achieves rapid positioning of data resources; a data partition mechanism that provides N:N resources backup and parallel recovery; an adaptive write/read strategy and synchronization mechanism that guarantee high data consistency, and a high-performance disaster recovery strategy that maintains most of the resources available and reliable even when a certain region was disconnected. In the following, we describe the major design considerations of SandStone.

3.1 Architecture Overview

Fig.1 depicts the architecture of a SandStone peer. The SandStone is composed of four main components: data storage module, key based routing module, communication module and configurations & statistics module. Intercommunication between peers is completed by the bottom layer communication module.

Middle layer key based routing, KBR, refers to find the best suitable host for a key as the input, also include peers' ID allocation, DHT routing protocol, peers failure detecting and etc. In this layer, we implemented a novel ID allocation mechanism for traffic localization in section 3.2, and a one-hop DHT enhancement in section 3.3.

The top layer Data storage module takes charge of storage and management of subscriber data, include data storage, data restoring, data consistency verification and etc. Here we came up with a unique and practical replica placement strategy in section 3.4, and the enforced strong consistency strategy in section 3.5.

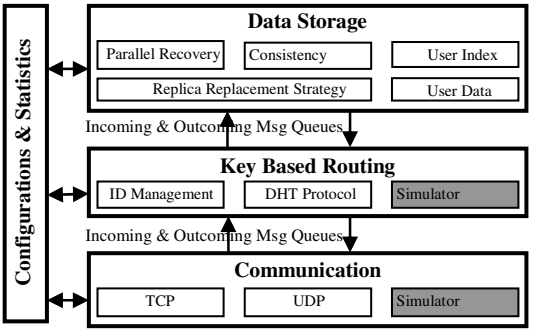

Fig. 1. The architecture of a SandStone peer

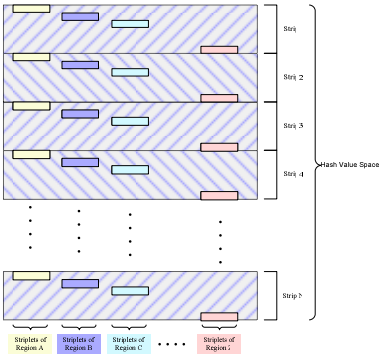

Fig. 2. The ID assignment

Furthermore, configurations & statistics module takes charge of managing and configuration the other layer modules. In addition, the simulation adaptor is introduced to let SandStone can be switched seamlessly between the simulation and realistic environment.

3.2 Traffic Localization

One of the key design requirements for SandStone is that it must make the traffic localization as far as possible. Recent studies [12, 13] have shown that a large volume of inter-domain redundant traffic by P2P mismatch problem already became the serious problem for ISPs. Due to the complexity of distributed networks, a novel traffic localization technique can not only consider the reduction of application response time but also decrease the backbone network overhead caused by inter-domain traffic.

Aiming at this goal, we have to answer two questions: (1) how to let peers carry region indication; (2) how to impel the data operations to exhibit a localized pattern.

First question, clearly, peer ID is an ideal place to embed regional identifiers such as province or city information. Most of solutions proposed to use a prefix for that, but such an ID assignment will result in severe load unbalance. SandStone use a Strip Segmentation solution as depicted in Fig.2. At first it divides the whole hash value space into N strips equally, and then it divides each strip into M striplets. M is the number of regions, and N can be user-defined, such as 1000. Notice that all the striplets at the same position of each strip constitutes the ID ranges for one region, peers belong to that region can randomly choose its ID within those intersected ranges. The Strip Segmentation can be regarded as a compromise to greatly mitigate the unbalance resulted by prefix solutions. This rule is known to every SandStone peer, thus it will be able to calculate the geographical region of any peer according to its ID.

For the second question, in the existing telecom network, subscriber data always carry the corresponding region information, such as subscriber attribution information, subscriber roaming information and etc. Utilized above characteristics, SandStone constructs a unique key from a subscriber data by using consistent hashing and with piggybacked its region information. Key k is assigned to the first peer whose identifier is equal to or follows k in the same region space. Every SandStone peer joins in the globe DHT and a logical region DHT at the same time, as shown in Fig.3.

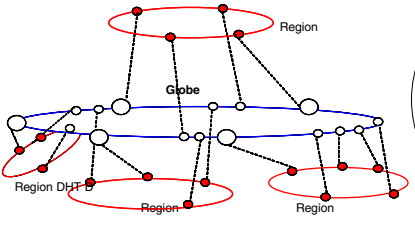

Fig. 3. The two layered DHT

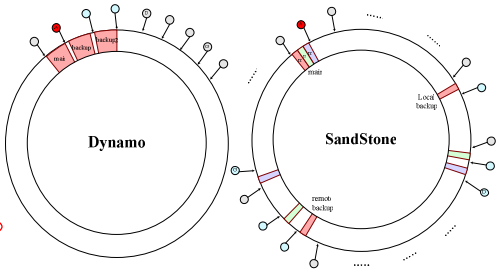

Fig. 4. The replica placement

Globe DHT and logical region DHTs are maintained by same KBR routing mechanisms. When a peer requests for a desired subscriber data, the default lookup strategy tries to search in the same region DHT as the requester's, then turn to the globe DHT if not found in region DHT. This simple design grasps some essence of the present user calling model, in which intra-region call attempts constitutes a major proportion, 70% for example. Because intra-region call/session handling related DHT operations will not bother the inter-region links anymore, it can be summarized that SandStone can achieve the same performance as traditional HSS in the aspect of traffic localization.

3.3 KBR Routing with One Hop Enhancement

In telecom networks, SandStone is built for latency sensitive applications that require at least 99.999% of read and write operations to be performed within 300 milliseconds. To meet these stringent latency requirements, it was imperative for us to avoid routing requests through multiple peers (which is the typical design adopted by several DHT systems such as Chord [14]). This is because multi-hop routing will bring on variability in response delay, increasing the latency at higher percentiles.

In order to meet the latency demand, SandStone can be characterized as a one-hop DHT, where each peer maintains enough routing information locally to route a request to the appropriate peer directly.

All SandStone peers belong to different regions form a globe ring topology just like Chord. In order to provide more efficient application response time, except the existing finger table router mechanism in Chord, SandStone makes a further step, to set up an extra One-hop Routing Table which include whole peers' router and state information for the one hop searching. One-hop Routing Table contains all the peers' router information in the DHT. Apparently the key question is that how to maintain the One-hop Routing Table freshness in every peer without too much resource consumption.

This is unrealistic that each peer maintains a One-hop Routing Table only by its self-organization mechanism on this large scale of network environment. This disorder organization and management method will cause a huge bandwidth overhead and routing table update latency. So there must be a number of special peers to play the role in unified managing and informing routing information. We will refer to these special peers which take charge of maintain routing information as SPM (Super Peer Maintenance).

There is at least one SPM node in certain region. All peers belong to this region must register to the local SPM when they joining into the SandStone system. The local SPM takes charge of peers in this region. SandStone failure detection mechanism uses a simple ping-style protocol that enables those peers in the system to learn about the arrival or departure of their neighbor peers. When joining or leaving the system, the peer's state change will be detected by its neighbor peer. And then the neighbor peer will inform local SPM this alternation information. A gossip-based protocol propagates peer alternation information and maintains a strongly consistent view of One-hop Routing Table between SPMs belongs to different regions. Finally, each SPM forward this notification in a broadcasting way to all peers which are located at its management region.

The advantage of SPM is that the routing information can be forwarded to all peers in entire system by a unified and effective fashion. It also decreases the traffic overhead in backbone network because one routing information message only transfers once in backbone links. It is worth mentioning that SPM nodes are only involved in routing table updating and managing in SandStone system. They don't deal with specific application requests so that they will not be the bottlenecks of the performance even with the increasing of application requests.

3.4 Replica Placement

To achieve high availability and durability, SandStone chooses the replica-based scheme that replicates the subscriber data on multiple peers. The replica placement policy serves two purposes: maximize data reliability and availability, and maximize network bandwidth utilization. For both, it is not enough to spread replicas across computers, which only guards against computer failures and fully utilizes each machine's network bandwidth. We must also spread data replicas across regions. This ensures that some replicas of a subscriber data will survive and remain available even if backbone network of a region is damaged in SBF2 scenario.

How to parallel recovery from the corresponding nodes is another important issue in replica mechanism of distributed storage system. BitVault and many related research works argued to spread the replicas randomly across the whole overlay, so as to make parallel recovery feasible. The drawback is that each peer must "remember" a long list of other peers hosting backups for its subscriber data, so make it not so scalability and reliability under churn. As shown in Fig.4, Dynamo still chose the original method to let the coordinator to replicate data at the N-1 clockwise successor nodes in the ring, N usually be 3. For a recovering operation, it is inefficient that node can only download data from the other corresponding N-1 nodes.

So in SandStone, we decided to choose three replicas as default R, represent the number of replicas for one subscriber data. Among the three replicas, the first would be stored in a peer of local region according to DHT key-value fundament, the second replica should be stored in a peer of different site but in same region, and the third would be maintained in the different region peer. This replicas' benefit is that not only improved traffic engineering based on specific application model (70% of applications are intra-region data operations), but also achieved good disaster recovery by out-regions.

At this time, new problems have emerged that how are replica locations selected? In SandStone, the backup data is stored in a certain segmentation rule so that it is no

need to remember the location of other replication data for any peers. To explain the replica placement, we need firstly define the ideal recovery factor L, as how many candidates should be used as the recovery source. Each peer calculates different offsets according to various data in its L striplet position. The keys of data are mapped to distinct strips after added different offsets, and then construct the backup keys of replica data. Finally, the replica data will be stored in different backup peers according to key-value principle and its backup key. Fig.4 depicts that a SandStone peer (red one) replicates its data partitions to 6 different peers, with L=3 and R=3. When red one recovering, it can pull the replication data from other 6 peers simultaneously. In this case, every peer only need to allocate one sixth space as the backup operation occupied for the failed peer, it's another benefit.

3.5 Consistency Strategy

To keep multiple replicas consistent is a difficult but inevitable task. Although data was partitioned, SandStone provides optimized eventual consistency, which allows for updates and amendments to be propagated to all replicas asynchronously.

Based on our specific application model, SandStone modifies the traditional "quorum" technique as its consistency strategy. For different peer failure modes, the read and write operation policies are defined as follows.

(1)When there is no peer fail, according to our business mode that read operation is more than write operation (read is 10 times of write), SandStone implements W=3, R=1 strategy. In this case, the data must be successfully updated in 3 peers simultaneously; otherwise write operation returns fail and peers has been updated roll back the data. With respect to read operation, SandStone try to fetch main replication firstly, if it is ok, then return the data to application client, otherwise it try to fetch local backup replication and remote backup replication in sequence. Only all three operations failed, this read operation return fails;

(2)When there are some peers fails, considering there is a trade-off between availability and consistency, SandStone implements an adaptive write policy for different data operation type. The "delete" operation still enforces of W=3 strategy; the "add" operation enforces of W>=2 strategy, and the "modify" operation enforces of W>=1 strategy. So each operation has a minimal acceptable number X (for delete/add/modify operation, the X is 3/2/1 respectively). Assuming, for one given data, Y is the number of current available peers which are responsible for storing three copies of the data. If Y>=X, then W=Y strategy is enforced; otherwise this data operation return fail. With respect to read operation, R=1 strategy is still enforced.

(3)When a fail peer comes back and under restoring process which synchronizes data from corresponding peers, the read operation will be blocked by this peer and return fail.

(4)If a peer joins, the data which the new peer is responsible for storing need to be moved from successor node. After that the incremental data restore process will be triggered and the corresponding data will be transmitted to the new arrived peer. On the other hand, the read operation on this peer will be blocked until the restoring process finished. If a peer leaves, the successor node will perform incremental data restore process for the new incremental key range space. Then the read operation within the incremental key range on this peer will be blocked until the restoring process finished.

In the aspect of data version, with the introduction of NTP servers, SandStone uses timestamp as version tag in order to capture causality between different versions of the same data. One can determine which version is the latest one by examine their timestamps. If the version of same data in corresponding peer conflict, then the older one will be overwritten and updated.

Base on consistency strategy described above, SandStone provides eventual consistency guarantee which has been proved in a wide range of experiments.

4 Implementation

We implemented SandStone in C++, atop the ACE [15] (Adaptive Communication Environment) toolkit which provides a rich set of reusable C++ wrapper facades and framework components that perform common communication software tasks across a range of OS platforms. In the choice of the storage engines for SandStone, we have two kinds of options. One is an open source database MySQL [16] which is easy to development and integration. On the other hand, we also implemented a simplified memory database MDB to improve the efficiency of data access.

We replaced the existing HSS module by our SandStone in IMS system. The architecture of IMS over SandStone is shown in Fig.5. A-CSCF is combined by P-CSCF, I-CSCF and S-CSCF for the flat structure design. Toward A-CSCF module, it doesn't need to know either HSS or SandStone is bottom subscriber database. The A-CSCF module access SandStone peers through existing IMS Cx interface and lookups the subscriber data in whole DHT network. When SandStone peers found the target data successfully, it returns the data to A-CSCF module.

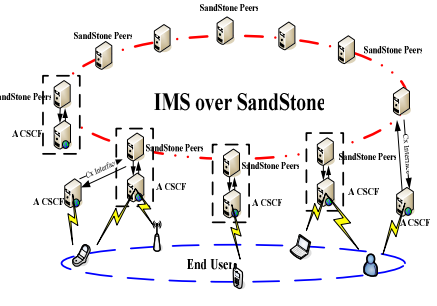

Fig. 5. The architecture of IMS over SandStone

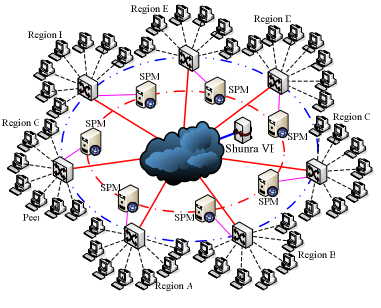

Fig. 6. The topology of SandStone

Furthermore, in order to simulate the various scenarios and the business modes for evaluating the performance of SandStone, we implemented two auxiliary testing tools TT (Testing Tool) and PTT (Performance Testing Tool) in addition. TT is a comprehensive test and management tool which can control the behavior of peers, trigger various peers failure model, and monitor the state of whole peers such as their routing table information. PTT is another test tool for performance evaluation. For simulating the CSCF module, it can send various application requirement packets by introducing different business scene model.

5 Experiments

In the process of building and deploying SandStone, we have experienced a variety of issues, some operational and technical. We built a SandStone prototype of 3000 peers located in three regions separately, each of which is a commodity PC with a 2.2GHz Core2 CPU and 2GB memory. The operating system on each peer is SUSE9 sp4, running the Linux 2.6.5-7.308 kernel. We use two experimental network environments to measure our system. The first is that these computers are connected with a series of Huawei Quidway S3500 1000Mb Ethernet switches. The second one introduces a Shunra Virtual Enterprise [17] to simulate a wide variety of network impairments, exactly as if SandStone were running in a real production environment. The topology is shown in Fig.6. Unless otherwise specified, the experiments lasted 24 hours.

5.1 Load Balance

In order to determine whether a distributed storage system stores data rationality and effectively, we first measure data load distribution in SandStone. We randomly generated 10 million subscriber data according to user registered statistics by China Mobile, and put them into the 3000 peers based on DHT key-value rule. Finally, we made a statistic for distribution of these data in 3000 peers.

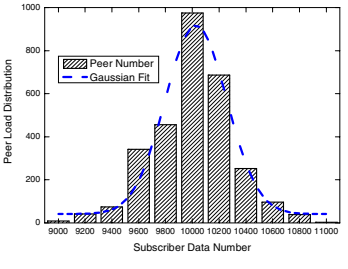

Fig. 7. Load balance

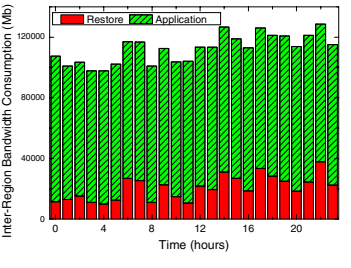

Fig. 8. The inter-region bandwidth consumption

From the Fig.7, through the optimization of strip segmentation arithmetic and ID assignment mechanism, the subscriber data obeys a Gaussian-like distribution. The number of data in 98% of peers maintained at around 10000. Data stored in SandStone have perfect load-sharing so that it also improves the traffic balance finally.

5.2 Reliability

To achieve high availability and durability, SandStone uses strict replica placement and consistency strategy. In this scenario, we evaluate the reliability of SandStone in the face of a variety of failure and application models.

As shown in table 1, under normal application model, each peer received 1 write and 10 read requests per second. 1% AF model will result in 1% of peers fail in every

hour, and in the next hour 80% of fail peers will be comeback. As is clear from table 1, the 1% and 3% AF model almost no impact on business success rate. The success rate in 12 hours is almost 100%. The SBF1 and SBF2 model lead to more number of peers' failure, but the SandStone still able to maintain more than 99.9% success rate. With the increased of business requests, the success rate has declined, but still more than 99%.

Table 1. The reliability of SandStone

Application Model	Failure Model	Success Rate
write:1/s per peer read:10/s per peer	AF (1%)	≈100%
	AF (3%)	≈100%
	SBF1	99.9992%
	SBF2	99.951%
write:30/s per peer read:50/s per peer	AF (1%)	99.99995%
	AF (3%)	99.9997%
	SBF1	99.9959%
	SBF2	99.68%
write:50/s per peer read:100/s per	AF (1%)	99.99992%
	AF (3%)	99.9973%
	SBF1	99.9934%

Table 2. The recovery time in SandStone

Database Mode	Peer Behavior	Recovery Time (seconds)
MySQL	Inactive (30 mins)	40.1
	Inactive (60 mins)	75.2
	Add	161.2
	Delete	82.6
Memory DB	Inactive (30 mins)	31.7
	Inactive (60 mins)	58.1
	Add	132.7
	Delete	74.1

5.3 Latency

While SandStone's principle design goal is to build a highly available data storage system, application response time is an equally important criterion. As mention above, to provide a best customer experience, SandStone must guarantees that the application can deliver its functionality in a bounded time. In this experiment, we measured the application response latency under various application models' frequency.

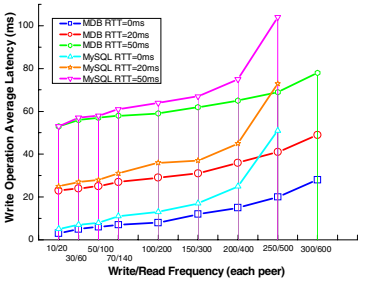

Fig. 9. Write operation average latency

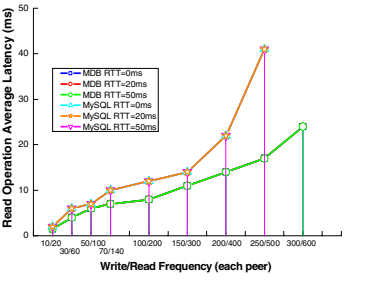

Fig. 10. Read operation average latency

It can be seen from the Fig.9 and Fig.10, with the expanding of business requests, the latency of write and read requests are increased. But even in the most unexpected application model situation (write/read frequency is 300/600 per second per peer); the latency is also much lower than the restriction of 300ms which is the business can tolerate. In the follow-up experiment, we turn on the Shunra Virtual Enterprise to

simulate the path propagation delay between different regions, from 0ms to 50ms. At this time we can see from the figure, with the increased propagation delay, the latency of write request operation has a greater degree of change obviously because of remote data replica placement strategy in section 3.4. However, because most of the data can be obtained from the local peers due to data localization, the latency of read request operations is basically no change, although frequency is increased. Furthermore, it is evident that the efficiency of Memory DB is higher than MySQL and the latency under Memory DB environment is lower than under MySQL's, especially in heavy business requests models. It is worth mentioning that the system can't afford the 300/600 frequency under MySQL environment because of the limitation of DB pools.

5.4 Recovery Time

In SandStone system, peer outages (due to failures and maintenance tasks) are often transient but may last for extended intervals. For systems prone to peer and network failures, availability can be increased by using optimistic replication techniques, where data are allowed to propagate to replicas in the background, and concurrent, disconnected work is tolerated. When peer come back again, it will recover incremental data from other replicas. So data recovery process time also determines the reliability of the system. In this experiment, we compared the data recovery time under series of node-state changes scenario. In experiment environment, each peer have stored approximately 10000 subscriber data, and the application model is normal, so each peer received 1 write and 10 read requests per second.

From the table 2, as a result of the parallel synchronization and incremental recovery technology, peer data recovery time maintained at a very small value. Even in the Add and Delete scenario that need to synchronize the entire data, a peer change process only consumed less than 200 second. This kind of recovery time in the Carrier Grade network is totally acceptable. Meanwhile, it is clear that the recovery time of Memory DB is shorter than MySQL's.

5.5 Bandwidth Consumption

Although SPM node is only responsible for the maintenance and synchronization of one-hop routing table, its bandwidth consumption is still one of our concerned issues. We recorded the bandwidth consumption of SPM under various failure models.

As can be seen in table 3, in the general failure model such as AF model, the bandwidth consumption of SPM is very tiny, almost can be ignored. Even in the case of SBF1 model (almost 50% of peers in same region failed simultaneously); the bandwidth consumption is just 820KB/s. For such a very low probability of unexpected events, this overhead of SPM is totally acceptable. It is worth mentioning that SPM uses a unified manner to inform the routing table change in SBF2 model, so the bandwidth consumption maintained at a very small value.

On the other hand, we calculate the inter-region bandwidth consumption in whole SandStone network. The application model is normal, and 3% AF failure model is introduced. From the Fig.8, we can see that despite the restore traffic in the ever-changing, but the overall inter-region bandwidth consumption is still maintained at a relatively stable value that decreases the impact to backbone network.

Table 3. The bandwidth consumption of SPM

Failure Model	Bandwidth Consumption (KB/S)
AF (1%)	1.9
AF (5%)	34.8
AF (10%)	75.6
SBF1 (50fails)	93
SBF1 (100fails)	210
SBF1 (200fails)	450
SBF1 (500fails)	820
SBF2	15.3

6 Conclusion

In this paper, we present the analysis and main design considerations of SandStone. The main contributions of SandStone are: a one-hop DHT enhancement, a Strip Segmentation ID assignment and a two layered DHT for traffic localization, a novel replica placement schemes and an enhanced protocol for data consistency strategy. Till now, by simulation and running in an experimental environment (thousands of nodes), SandStone achieved the Carrier Grade performance as listed in section 2.2. We'll seek more deployments to testify the SandStone performance, and adapt SandStone to more telecom application scenarios as listed in section 1.

References

1. P2PSIP (2009), `http://www.ietf.org/html.charters/p2psip-charter.html`
2. Matuszewskil, M., Garcia-Martin, M.A.: A Distributed IP Multimedia Subsystem (IMS). In: Proc. of WoWMoM (2007)
3. Van Leekwijck, W., Tsang, I.-J.: Distributed Multimedia Control System Using an In-network Peer-to-Peer Architecture. In: Proc. of ICIN (2006)
4. Ghemawat, S., Gobioff, H., Leung: The Google file system. In: Proc. of ACM SOSP (2003)
5. Oceanstore (2009), `http://oceanstore.cs.berkeley.edu`
6. Rhea, S., Eaton, P., Geels, D., Weatherspoon, H., Zhao, B., Kubiatowicz, J.: Pond: the oceanstore prototype. In: Proc. of USENIX FAST (2003)
7. Zhang, Z., Lian, Q., Lin, S., Chen, W., Chen, Y., Jin, C.: BitVault: a highly reliable distributed data retention platform. ACM SIGOPS Operating Systems Review archive 41(2), 27–36 (2007)
8. Weil, S.A., Brandt, S.A., Miller, E.L., Long, D.D.E., Maltzahn, C.: Ceph, A scalable, high-performance distributed file system. In: Proc. of OSDI (2006)
9. DeCandia, G., Hastorun, D., Jampani, M., Kakulapati, G.: Dynamo: Amazon's Highly Available Key-value Store. In: Proc. of ACM SOSP (2007)
10. Stribling, J., Sovran, Y., Zhang, I., Pretzer, X., Li, J., Kaashoek, F., Morris, R.: Simplifying Wide-Area Application Development with WheelFS. In: Proc. of USENIX NSDI (2009)
11. 3GPP (2009), `http://www.3gpp.org/`

12. Aggarwal, V., Feldmann, A., Scheideler, C.: Can ISPs and P2P Systems Cooperate for Improved Performance? ACM SIGCOMM Computer Communications Review 37(3), 29–40 (2007)
13. Shen, G., Wang, Y., Xiong, Y., Zhao, B., Zhang, Z.: HPTP: Relieving the Tension between ISPs and P2P. In: Proc. of USENIX IPTPS (2007)
14. Stoica, I., Morris, R., Karger, D., Kaashoek, M.F., Balakrishnan, H.: Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications. In: Proc of ACM SIGCOMM 2001, San Deigo, CA, August 2001, pp. 149–160 (2001)
15. ACE (2009), `http://www.cs.wustl.edu/~schmidt/ACE.html`
16. MySQL (2009), `http://www.mysql.com/`
17. Shunra, V.E.: (2009), `http://www.shunra.com/`