nexenta™

White Paper

Nexenta Replicast

By Caitlin Bestler, September 2013

Table of Contents

Overview

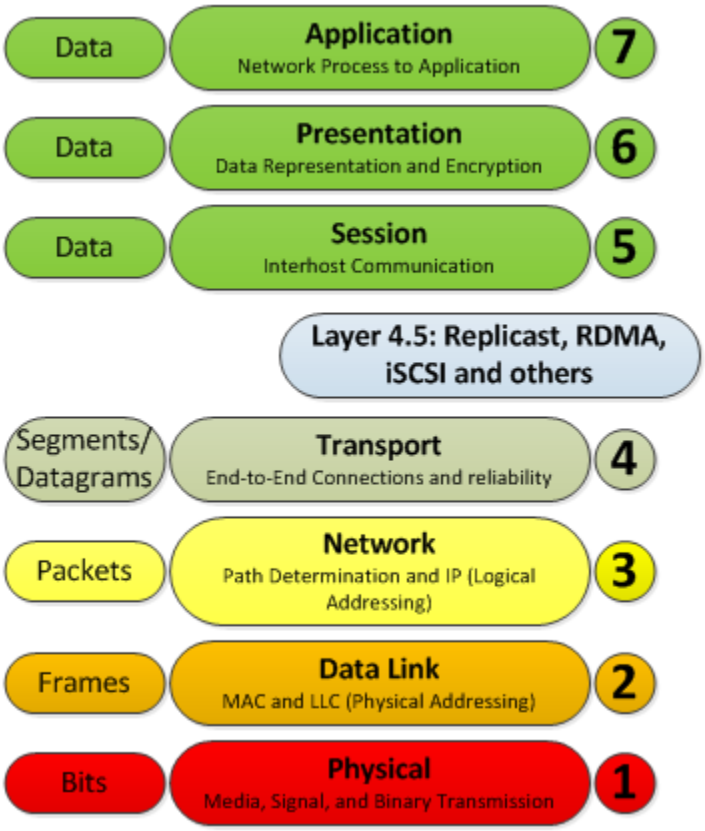
Nexenta Replicast is a storage and transport stack that enables efficient and reliable network replication of storage content. This storage software feature was developed by, and is the property of, Nexenta Systems, Inc. The purpose of this white paper is to educate the reader about the concepts of Nexenta Replicast and describes its functionality.

Nexenta Replicast Description

Nexenta Replicast is a storage ("Replicast Storage") and transport ("Replicast Transport") stack that enables efficient and reliable network replication of storage content.

Replicast Transport utilizes lower layer multicast protocols providing unreliable datagram service and is positioned directly on top of the Layer 4 in the conventional OSI/ISO 7-layer model.

Figure 1: OSI/ISO 7-layer model with Nexenta Replicast.



Nexenta Replicast Storage in turn uses its own underlying (Replicast) Transport layer in order to:

- Efficiently replicate content on a ”Send Once, Receive Many” (SORM) basis, which drastically reduces the required network bandwidth
- Use multicast communications to control which specific servers store replicas, thus enabling a new model for controlling distributed storage

Nexenta Replicast radically optimizes network bandwidth while simultaneously avoiding network congestion and dynamically load-balances replicated data between participating storage servers.

Send Once, Receive Many

Users demand uninterrupted access to their data at all times. Data availability implies a certain degree of redundancy of all system components providing the corresponding storage service: drives, servers, network. Protection against server and (often) drive failures means that multiple copies of the same data must be placed on multiple storage servers. To achieve that, data is being replicated over distance, typically over TCP/IP intranet, sometimes over WAN.

Data replication takes significant and constantly increasing portion of the overall network bandwidth.

As the cost of raw storage plummets, the cost of network replication determines how much protection against content loss is affordable. In 2011 the cost of 1TB 7200 RPM SATA drive was above $400, which compares favorably to today’s prices below $70. Same is happening to SAS and SSD drives. In short, same amount of dollars can now buy at least 3, often more times as many TBs than in 2011. But 3 times as many TBs of network traffic would be nearly 3x the cost.

Assume a given content must be replicated to protect against the simultaneous loss of 3 (three) replicas. Conventional distributed storage solutions would then require 4 network replicas – that is, 4 (four) transmissions of the same content over network interconnecting servers that store the content.

Complex or advanced solutions can reduce that below 4 replicas, but only very slightly. Nexenta Object Storage (NOST) 1.0 uses local RAID-1/RAID-10 mirroring to implement a "2*2" solution. NOST executes only two network replicas because each network replica is locally RAID-1 protected against the loss of a disk drive. However, these solutions have costs.

Erasure encoding based solutions implement a custom (typically configurable) N/M schema over multiple storage servers, where the storage system of N servers can withstand a loss of M (M < N) servers. Erasure encoding however slows access to content since majority of the data slices must be retrieved to restore the original data. Just as an army marches at the speed of its slowest soldier, a cluster using erasure encoding is as slow as its busiest server. The Nexenta

Object Storage "2*2" solution requires configuration of parallel networks to avoid risk of reduced availability from loss of two network links or forwarding elements.

Figure 2: Conventional Replication. Source sends whole chunk to each Replica.

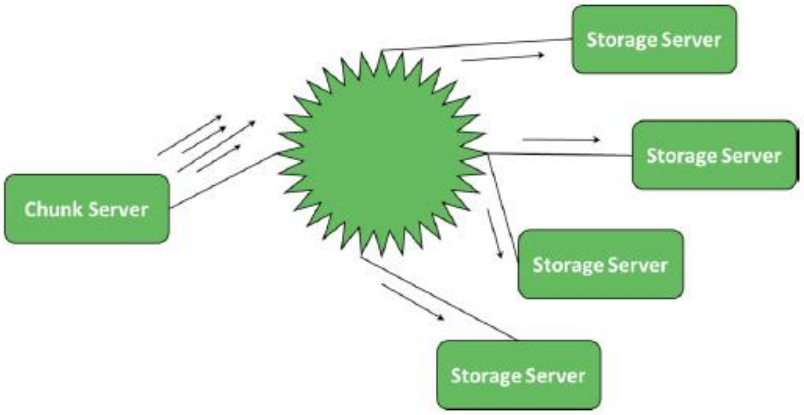


Figure 3: Erasure Encoding. About half of the payload is sent to each slice.

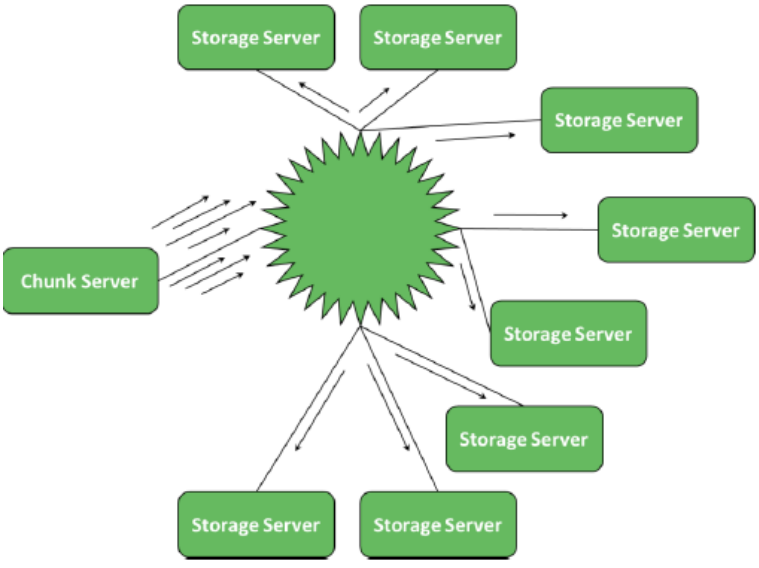
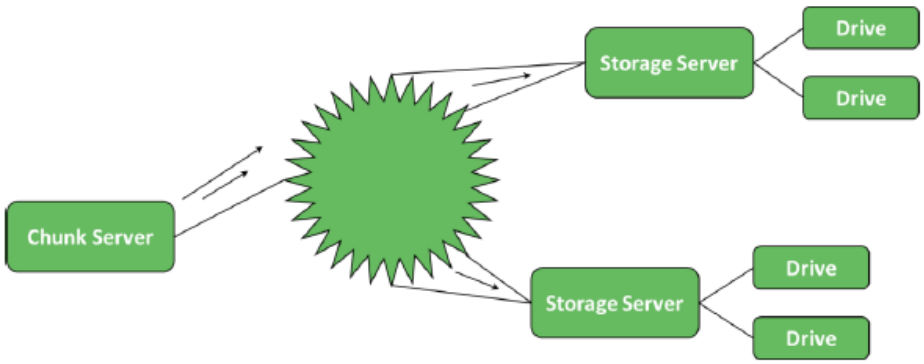


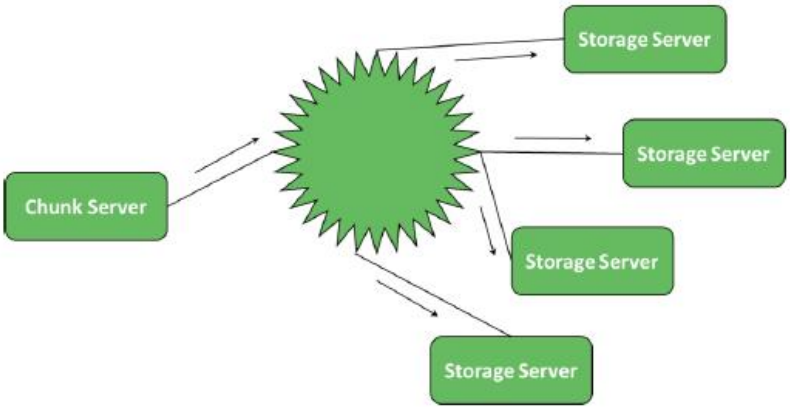
Figure 4: NOST 1.0 “2x2” Solution. Two network replicas that each protect against loss of one drive locally.



While working to make the 2*2 solution easier to deploy, we realized that a 2*2 solution still required transmitting the data twice. That’s a major improvement over transmitting 4 times. But - why not go for a solution where all four replicas could be created by transmitting just once?

Replicast enables exactly that – effectively reliable replication of content to any number of recipients with a single transmission.

Figure 5: Nexenta Replicast. Send once, receive many.



Each chunk is transmitted as a sequence of datagrams. Accurate reception is verified with a hash signature of the entire chunk. This radically reduces the number of packets required to

verify accurate reception. The cost is that when a retransmission is needed the entire chunk must be retransmitted.

But this is acceptable when virtually all errors are caused by congestion drops, rather than actual transmission errors.

When replicating storage chunks, a check against data corruption far better than a network checksum is required anyway. When you are storing petabytes, letting 1 in 64K bit errors go undetected simply will not work, 1 in 2**32 isn't much better.

Nexenta Object Storage (NOST) is based on Nexenta's Cloud Copy on Write (CCOW) technology that uses a cryptographic hash algorithm (such as SHA-256, SHA-512 or SHA-3) to protect against data corruption and forgery. Lesser hash algorithms may be used when only protection from data corruption is required. When specialized offload instructions and/or parallel processing solutions (such as GPUs) can perform the hashing there is minimal benefit to using a cheaper algorithm, however an algorithm such as Murmurhash is suitable when pre-image attacks are not a concern.

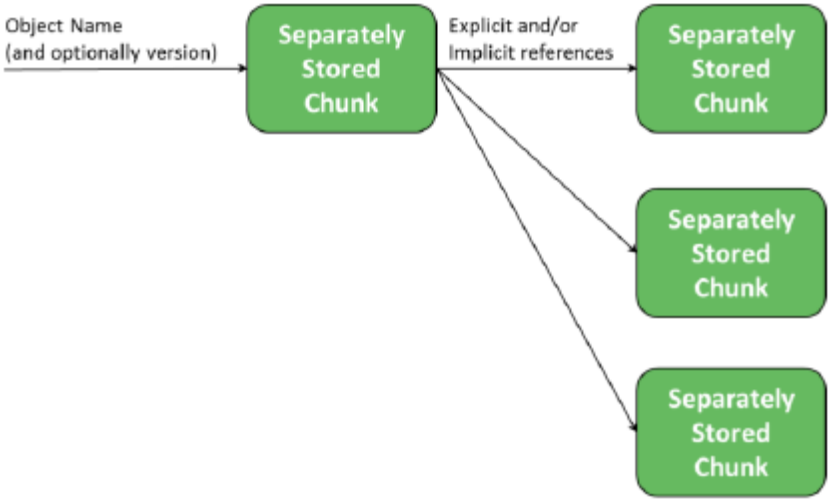
Nexenta Replicast relies on reservations issued by the receivers to avoid congestion on the destination links. Congestion on the inbound links is confined to the buffering within the originating host. Nexenta Replicast assumes that a maximum bandwidth can be stated for each port which can be carried through the core network to any destination in a non-blocking fashion. So if each Nexenta Replicast transmitter confines itself to the provisioned maximum bandwidth, then congestion is avoided because the destinations only issues reservations up to their provisioned input capacity.

Making 4 (four) replicas of a chunk with a single transmission is a major breakthrough, but it turns out that the techniques that enable making the reservations to support rendezvous transmissions have even greater benefits for distributed storage.

Distributed Storage Basics

A Distributed Storage system breaks each object down into a root (which can be version specific) and zero or more payload chunks. The root is identified by the object name (and version). The chunks have arbitrary identifiers that are referenced in the root. They can be derived from the payload or merely be generated as some form of global serial number. Nexenta Object Storage identifies chunks by the cryptographic hash of the compressed payload. OpenStack Swift does not chunk objects. The HDFS namenode generates arbitrary identifiers for its chunks ("blocks").

Figure 6: A distributed storage system.



Both the version roots ("Version Manifests" for Nexenta Object Storage) and the chunks must be distributed to multiple storage servers to protect against the loss of specific servers.

The fundamental issues for any distributed storage system are:

- How are the storage servers to hold a given Version Manifest or Chunk selected?
- How do you find them when you come back later looking for them?

Traditionally there have been two solutions:

1. Central metadata server makes the choices, and then remembers what it chose. This is how pNFS and HDFS work
2. A Consistent Hashing Algorithm picks the set of servers for any given Manifest/Chunk based on an agreed upon hash of its identifier
    - If two parties agree on the set of servers the resources should be distributed over, they will always assign resource X to the same set of servers
    - A keep-alive ring is responsible for reaching consensus on "the set of servers" that is connected

Nexenta Replicast enables a novel third solution:

3. A small group of servers uses multicast datagrams to dynamically select the storage servers. Instead of looking up N servers in a hash table, you look up a multicast group which contains M servers (probably 10 to 20). Those servers dynamically select the N that will store the chunk

When it is time to retrieve the chunk multicast datagrams are used to find the least busy server amongst the M (10 to 20) servers in the multicast group

Selecting the best N of M servers dramatically reduces the time it takes to process each put or get.

Making a dynamic selection from a small group (M) of servers provides virtually all the benefit of selecting from a larger group, at a fraction of the cost.

Picking just N on an arbitrary/random basis can never achieve a high utilization of IOP capacity.

Centralized control has its own costs of enforcing its hierarchy with sharp limits on scalability.

Consistent hashing, like all forms of hashing, has a heavy effectiveness fall-off once utilization of raw capacity goes over 50%.

Nexenta Replicast enables higher utilization, meaning ultimately that fewer servers can provide the same number of utilized IOPs as could be achieved with consistent hashing, and without the cliff effects inherent in centralized control.

Nexenta Replicast Advantages

Nexenta Replicast replaces the conventional Distributed Hash Table (DHT – see for instance http://en.wikipedia.org/wiki/Distributed_hash_table) with a collaboratively edited Hash Allocation Table. At a minimum, this allows the distribution of chunks across servers to be dynamically adjusted when a default hash algorithm results in a lop-sided, unbalanced distribution of chunks for a specific storage cluster.

Nexenta Replicast also allows storage servers to act as “volunteers”, to collaboratively optimize storage access latency, remaining free capacity, servere utilization and/or any other meaningful configurable management policy. While not being counted towards the required membership, volunteer servers can join specific “Negotiating Groups” to provide storage services to a specific subset of chunks. Volunteer servers using fast storage (RAM and/or SSD) will cache to optimize IOPs for the most frequently (and/or most recently) accessed content, thus improving the overall cluster efficiency. Nexenta Object Storage (NOST) enabled unlimited caching for Object Storage, but could only do so for in-line local caches. Nexenta Replicast enables free roaming caches.

Nexenta Replicast is fundamentally a method of enabling efficient collaboration within a storage cluster. All of its algorithms are intrinsically distributed. They are also heterogeneous - new heuristics can be deployed to achieve special objectives.