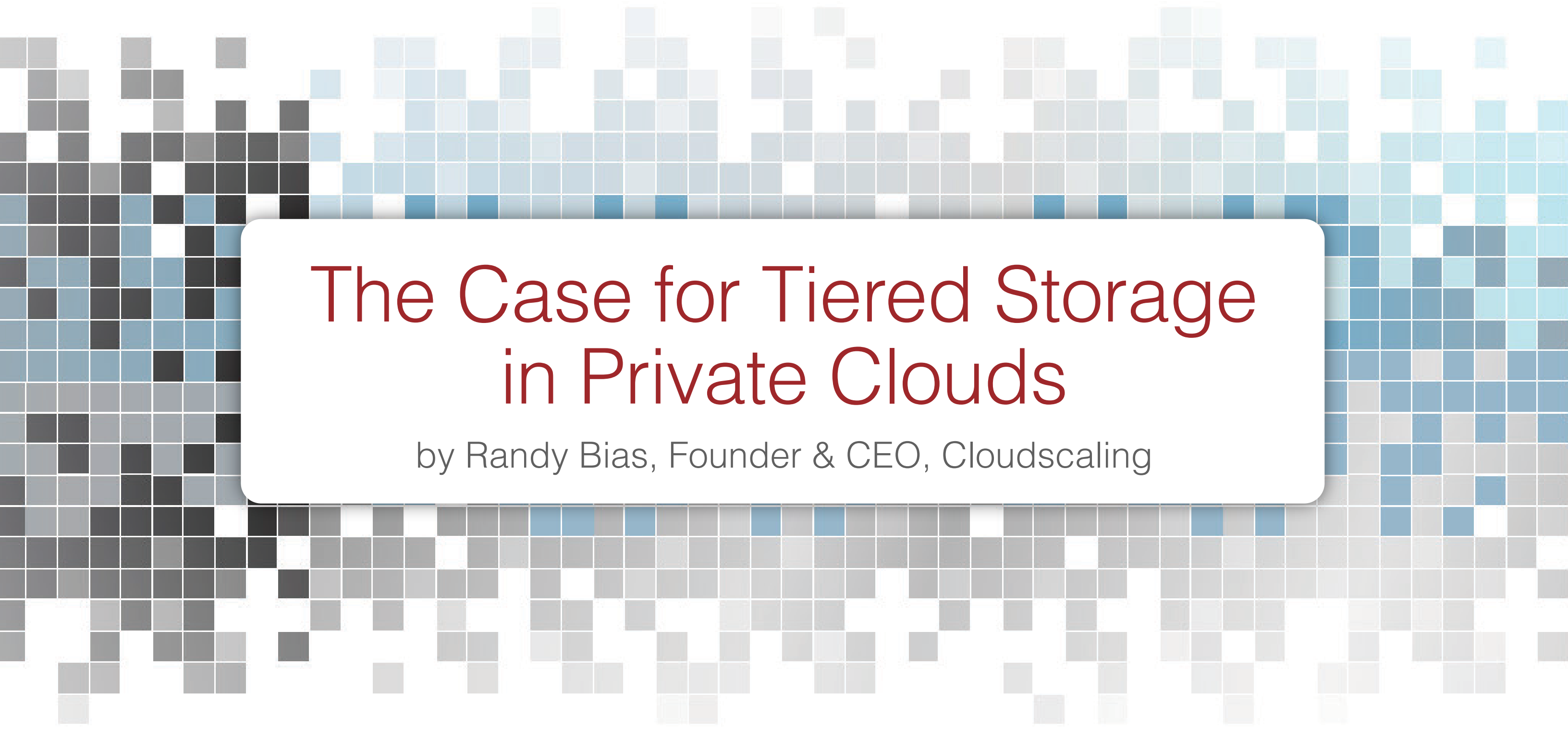
cloudscaling

WHITE PAPER



The Case for Tiered Storage in Private Clouds

by Randy Bias, Founder & CEO, Cloudscaling

Table of Contents

Introduction

One of the things I find particularly frustrating about the current open cloud marketplace is the surfeit of folks who are relatively new to infrastructure. With both buyers and sellers, you see a certain amount of reinvention of the wheel combined with wishful thinking. This is particularly egregious in the area of distributed storage systems, where instead of looking for practical solutions to the challenges, we see a number of marketing departments promulgating the idea that there are simple answers to complex storage problems.

I have news for you, there aren't. You can pick the red pill, the blue pill, or even both depending on what you are trying to do. What you cannot do is pick a single pill to cure all ills.

Unified, Distributed Storage Systems are Not a Panacea

Wishful thinking typically comes in the form of distributed storage systems. I don't mean that these technologies are bad; they are not. What I mean is that how the sellers market them and how the buyers often want them to behave is disconnected from reality.

Distributed storage systems are promising and valuable, but like anything, you have to understand the problems you are solving before you deploy them. As each storage system is designed to solve specific problems, matching the storage system architecture to the problem being solved is critical to success.

Enterprises use Tiered Storage

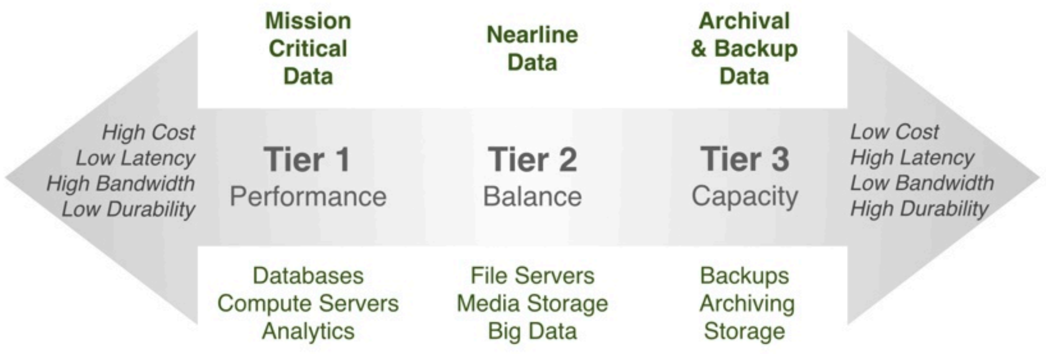
Long ago, businesses determined that there was a need for a variety of storage systems for a variety of applications. These applications generally fell into several key camps:

- Mission critical data (Performance): Fast and accessible regardless of cost
- Nearline (Balance): Easy to access, but cost effective data
- Archival and backup (Capacity): Slow to access, but as cheap and reliable as possible

Storage systems that solved for these particular use cases were typically referred to as Tier 1, Tier 2 and Tier 3. If we were to look at the detailed requirements for these use cases it would look like this:

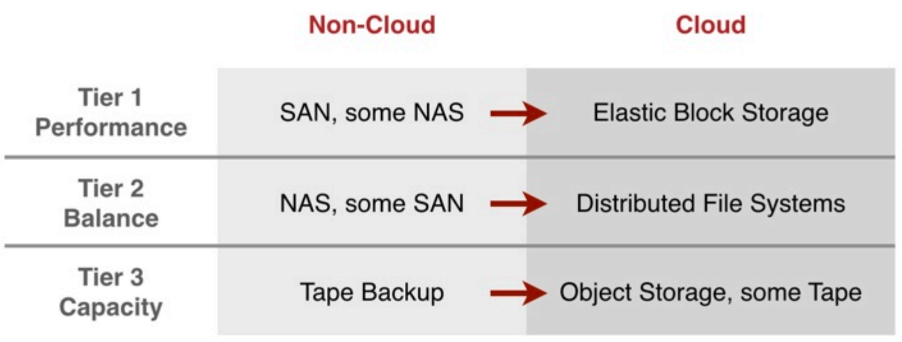
	Use Case	Cost	Latency	Throughput	Availability	Durability	MTBF	Consistency
Tier 1	Performance	$$$$$	Low	High	High	Low	High	Strong
Tier 2	Balance	$$$	Medium	Medium	Medium	Medium	Medium	Strong or Weak
Tier 3	Capacity	$	High	Lower	Low	Extreme	Low	Eventual

For most purposes you can think of tiered storage as on a continuum upon which your use case would fall, dictating which tier makes sense:



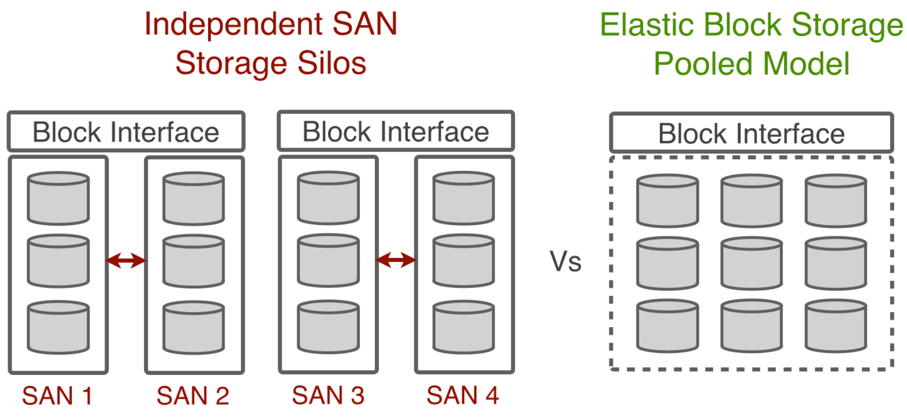
Now, this isn't exact as the variety of systems available for tiered storage have a tendency to blur the lines; however, this is how most enterprise storage engineers and enterprise storage vendors think of the world.

In today's cloud era, the process of tiering storage looks much the same, but the actual solutions have changed significantly. It looks more like this:



	Non-Cloud		Cloud
Tier 1 Performance	SAN, some NAS	→	Elastic Block Storage
Tier 2 Balance	NAS, some SAN	→	Distributed File Systems
Tier 3 Capacity	Tape Backup	→	Object Storage, some Tape

Why the change? There is a new imperative amongst mid-tier and larger enterprises to address the ongoing storage sprawl problem that is now becoming quite serious. These businesses have many independent storage silos that are difficult to manage, scale, and in aggregate are excessively expensive. Elastic Block Storage (EBS) is simply an approach to abstracting away this sprawl of SAN/NAS storage into a more manageable "pooled" model. Some will refer to this notion as storage virtualization or software-defined storage (SDS).



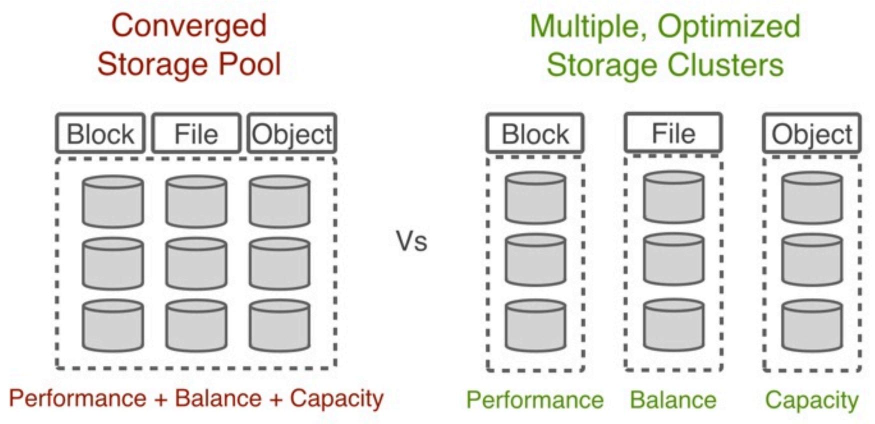
What's important is that the contract is still similar to the existing Tier 1 block storage solutions EBS is replacing. Low latency, high throughput, and high cost for mission critical applications. This replaces the SAN/NAS model by mostly being a better, more abstracted, and hopefully more "scale out" solution than the prior approach.

The next time you get a chance, ask your typical enterprise storage vendor how they are dealing with all of the pairs of NetApp and EMC boxes. Mostly they will tell you they aren't. It's manual, it's painful, and it's very expensive. Enterprises want this resolved.

Everyone is tired of managing hundreds of discrete and expensive storage systems, so the promise of a single system is appealing, but, much like a mirage in the distance, it vanishes as you approach.

A Single, Unified Distributed Storage System is a False Promise

The marketing departments of certain distributed storage system companies promise not only this pooled model for storage, but also the collapse and convergence of all three tiers of storage in a single, unified system. Now, you may, theoretically, buy one storage solution and use it for all parts of your cloud infrastructure storage: Tier 1 through Tier 3.



This is the false promise of deploying a single, unified storage system. At the very least, you will deploy the same storage technology multiple times to reduce the size of failure domains, but more likely you will deploy several different storage systems (and storage hardware configurations) that are architected to solve the problems specific to each tier.

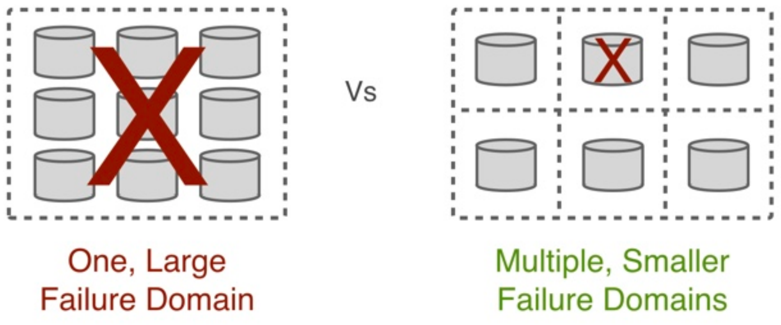
You can reduce costs moving to a cloud storage model, it's just that it won't be done by having one massive, infinitely scalable storage pool. Such a pipe dream can never exist.

Why?

With a distributed storage system deployed, you will still need to run at least two storage clusters for 1) risk mitigation by fault isolation and 2) to optimize for cost / performance. The dream of running one big distributed, unified storage cluster across all your systems is a mirage.

Risk Mitigation

Large failure domains are truly bad, with a capital B. Distributed storage systems solve the scale-out problem, but they don't solve the failure domain problem. Instead, they can make the failure domain much larger. You have to manage the size of your failure domain.



Running a single, homogeneous storage system versus multiple heterogeneous storage systems greatly increases the size of your failure domain.

With a Single System, (Implied) Promises are Broken

Modern cloud applications rely on promises between the end user and the cloud infrastructure. These implied contracts exist throughout a storage system and come from assumptions that end-users make about how the system works. For example, storage systems architected for block storage (usually Tier 1) use cases imply they are strongly consistent. If I write a block and then read the block, it will always be the block I just wrote. This implied contract does not exist in an eventually consistent system because it is optimized and architected for a different use case (typically Tier 3).

Frequently these assumptions are set over time by best practices. For example, on Amazon Web Services (AWS), all of your snapshots for the Elastic Block Storage (EBS) service are stored in the Simple Storage Service (S3). What that means is that you have an implicit guarantee that in the case of an EBS node or volume failure, you can always restore from a snapshot using the snapshot as a de facto backup. In effect, one service (S3) acts as a hedge against failure of the other service (EBS).

Great, but if you converge your storage tiers, then this implicit contract with the end user is violated. Data damage to your single, unified storage system could cause the loss of ALL data with no backup. This is why enterprises think in terms of data retention and backup policies and use multiple, independent, storage technologies and systems for each tier of storage.

A Single System Exposes You to Loss from Operator Error and Software Bugs

Modern IT systems fail for one of three reasons:

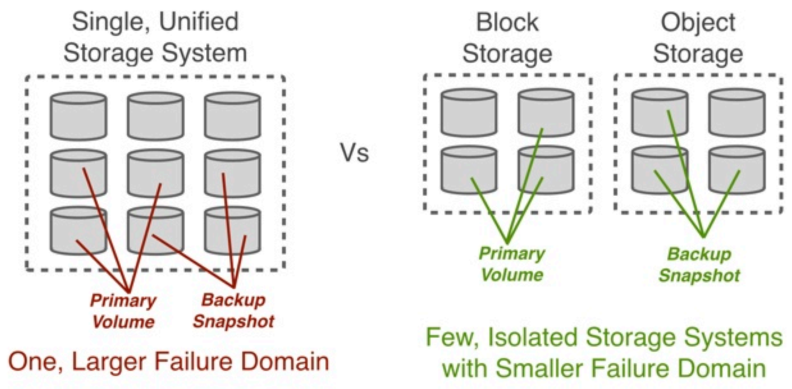
- Hardware failure
- Operator error
- Software bugs

Operator error and software bugs are far more common causes of system outages than hardware failure. Most hardware failure modes have been solved for a long time with traditional approaches such as high availability (HA) pairs and more cloudy approaches such as load balancing arrays and sharding.

With a single storage system, you are still exposed to operator error and software bug risks.

A failure in a single homogeneous system due to operator error or a software bug can be catastrophic. Though many public service outages occur each year such as with Flexiscale, Google, Microsoft and Twitter, these types of data loss and service outage events are far more common in private data centers where they are never publicly visible.

The point is that if you want to survive a single system outage, you need a second system for redundancy. Dreamhost runs two separate clusters of Ceph for their storage service for this reason. Google still backs up email data to tape.


Single, Unified Storage System
Primary Volume
Backup Snapshot
One, Larger Failure Domain
Vs
Block Storage
Object Storage
Primary Volume
Backup Snapshot
Few, Isolated Storage Systems with Smaller Failure Domain

Fault isolation between the major components of an integrated system is critical. There is inherently increased risk in a single homogeneous system without second system redundancy. As with the earlier example, by storing both the primary block volume and the volume's backup snapshots in the same system, the single system fails to meet the implied contracts of being able to reliably restore the volume in the event of a block storage system outage or data corruption. Both the original and the backup are at risk of being inaccessible or lost at the same time. Without the second storage system as a failsafe, all your eggs are in one basket.

You can always use a second deployment of the same distributed storage system for failure isolation[1]. This means that the problem is mostly a marketing expectations issue, not a technology issue with distributed

1 Although I would not recommend it. A better strategy is to have a dedicated storage tier for long term durable archival and storage on the cheap with a storage system designed for this purpose (aka "Tier 3 storage").

storage systems themselves. If there is one place I can criticize this strategy, it's that there is still an increased risk when running the same software across all tiers. A software bug could exist simultaneously on both systems. Ultimately, you can attain a much higher level of resiliency and reliability over time with multiple, fully isolated clusters running different storage systems (distributed or otherwise).

Google Tech Talk Series: How Google Backs up the Internet

In an October 2013 tech talk summarized here, Google Engineer Raymond Blum explains the strategy Google employs to ensure zero customer data loss while managing exabytes of data.

One of the key principles raised is that **data redundancy is not the same as data recoverability**. While making multiple data copies is effective for certain kinds of outages, it does not help meet the no loss guarantee.

Redundancy is neither a backup, nor a guarantee of integrity or recoverability. Redundancy can protect you from a meteorite strike on a data center location, but not a software bug, user error or corrupt buffer write that damages copies in all the redundant locations. And there aren't as many meteorite strikes as there are bugs in code, user errors and corrupted buffer writes. See the 2011 Gmail outage as an example. With a bug in the storage software stack, the same bug can crop up everywhere at once.

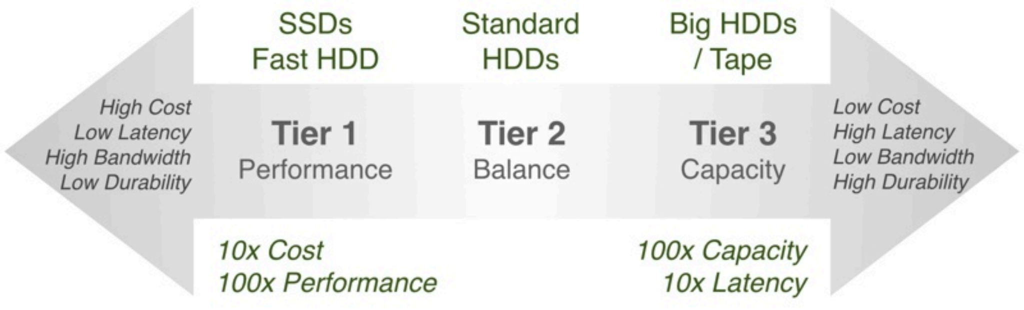
For Google, the primary purpose of redundancy of location is to support scalable processing, where you want all data references as close as possible to where the data is being used. Running massively parallel systems actually increases the opportunity for data loss. According to Raymond, "MapReduce running on 30K machines is great, until you have a bug. You have the same bug waiting to run everywhere at once which magnifies the effect."

To protect from data loss, you need redundancy and diversity in everything:

- **Location Isolation** - To protect against site problems, put the data in multiple sites.
- **Media Failure Isolation** - To protect against media failure, store multiple copies of the data.
- **User Error Isolation** - To protect against user error, have levels of isolation from user interaction.
- **Software Bug Isolation** - To protect against software bugs, store the data using diverse storage technologies (different software and storage media).

Price/Performance Storage Tiering

Collapsing tiers 1 to 3 in a single system is a bad idea. You need to tune storage tiers for specific applications and use storage systems architected and optimized for the tier's purpose. If you need to optimize the storage system for high performance block storage using SSDs for high IO applications, it is no longer cost effective for large capacity file storage. Conversely, if you need to optimize for long term, cost effective storage, it's no longer performant enough for high IO applications.



Distributed storage systems are beginning to add tiering features to support deployment of multiple storage pools, each with different underlying hardware configuration and performance characteristics, but these feature sets are new, and they effectively create non-isolated storage subpools within the larger homogeneous storage system.

Many distributed storage system architectures are based on an object storage model providing additional API interfaces for block and file storage. This means they are already optimized for capacity versus performance. Block storage performance using these systems is often unsuitable for applications requiring low latency, high bandwidth storage volumes.

There is significant effort and risk to follow the single, unified storage system path. For example, you're configuring multiple hardware configurations to support multiple use cases. The alternative is to deploy a handful of heterogeneous storage systems that are optimized for specific tasks. This solves the sprawl problem as you're only dealing with a small number of storage systems, while simultaneously avoiding a single homogenous storage system that creates a single point of failure.

The Case for Tiered Storage in Private Clouds

At Cloudscaling, we recommend following the principle that *simplicity scales*. For cloud storage, this means deploying a few, optimized storage clusters, one for each storage tier rather than a single, unified storage cluster that covers all three. The advantages are summarized as follows:

Single, Unified, Homogeneous Storage Cluster	Few, Heterogeneous Storage Clusters	
X	✓	Upgrade one cluster of software at a time
X	✓	Reduce risk of operator errors taking down entire system
X	✓	Reduce risk of network partitions taking down entire system
X	✓	Reduce risk of software bugs destroying ALL data
X	✓	Reduce risk of security breach "owning" entire storage cluster

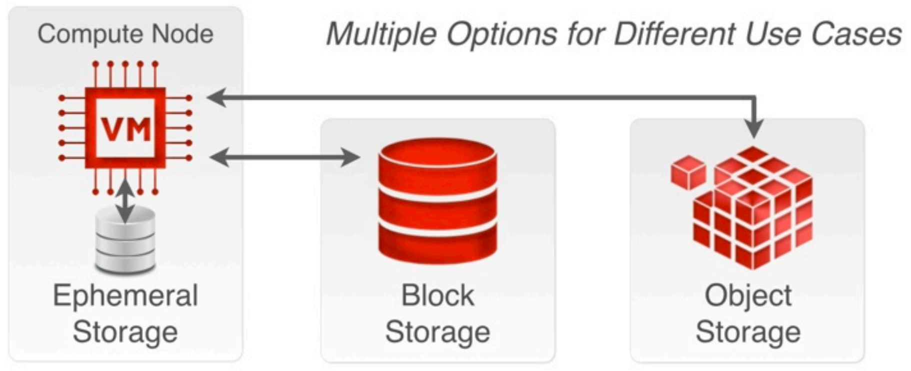
Building Elastic Cloud Storage Systems Using Tier Requirements

While there are several options in each tier to flexibly tailor the solution to the deployment requirements, we strive to apply the right tool for the job by deploying purpose built storage for each of the cloud storage systems (object, block and ephemeral). These system architectures are modeled on the storage systems employed by the largest of Internet cloud service providers (e.g. Amazon and Google) for cost-effectively managing data, and they are well understood technologies that have worked reliably and predictably at massive scale for years.

	Use Case	Cost	Latency	Throughput	Availability	Durability	MTBF	Consistency
Tier 1	Performance	$$$$$	Low	High	High	Low	High	Strong
Tier 2	Balance	$$$	Medium	Medium	Medium	Medium	Medium	Strong or Weak
Tier 3	Capacity	$	High	Lower	Low	Extreme	Low	Eventual

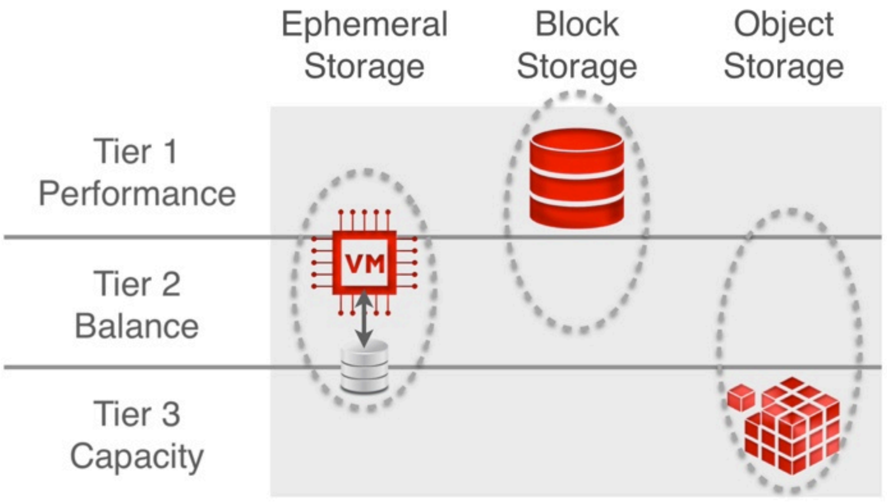
Cost / Performance Optimized

In modern elastic clouds, there are generally three types of storage deployed: ephemeral, block and object storage.



Block Level, Ephemeral Storage	*Block Level, Persistent Storage*	*Object Level, Persistent Storage*
Dedicated to the local instances	*Attached to instance*	*Referenced from instances*
Stateless instances, temp files, and high IOPS SSD only	*Boot volumes, legacy DBs and dynamic content*	*Used for static content storage, archiving, backups*

These different storage types are typically used for different tiers depending on the use case.



Ephemeral Storage delivers cost-effective local block storage that's co-located within the compute nodes, providing instances with direct access to storage for boot and data volumes. Ephemeral storage exists only for the life of an instance. It persists across reboots of the guest operating system, but is deleted when the instance is deleted. The advantage of ephemeral storage is that it is typically delivered with direct-attached storage (DAS). DAS based storage is almost always significantly faster than SAN/NAS since the PCI bus in a modern x86 server is inherently faster than Ethernet. In the past, this was less true because the bottleneck was typically the storage spindles themselves, but now with high speed SSDs and similar technologies, the network is once again becoming the bottleneck. The downside of ephemeral storage delivered via DAS is that it presumes that data replication and persistence is handled at the application layer. Of course, with modern cloud applications using a shared nothing architecture, this is always true. Depending on your workloads, you can view ephemeral storage as designed for Tier 1 high performance applications (particularly if using all SSD drives) or Tier 2 balanced performance applications (where you are using lower speed disk drives to boot VMs from instead of a more expensive SAN/NAS).

Block Storage provides persistent data volumes that are dynamically attached to virtual machine instances over the network to deliver high performance, scalable and cost-effective virtual disk drives. Persistent block storage volumes supplement the ephemeral boot volumes provisioned from local storage on the compute node. Block Storage usually supports booting instances from block storage volumes as well as snapshots to object storage for backup[2]. Similar to ephemeral storage, the hardware provisioned with block storage can be tuned for Tier 1 very high performance use cases (with all SSDs) or tuned for capacity and performance balanced use cases (with SSD accelerated HDDs). As an advantage over ephemeral storage, multiple volumes can be striped across storage nodes for increased capacity, performance and throughput. Block storage generally costs more per TB than ephemeral storage, but as it doesn't require expensive and proprietary hardware SANs, it costs significantly less than equivalent storage in enterprise virtualization environments.

Object Storage is a separate system optimized for capacity and designed for long term durability and archival. Object storage systems use data replication (usually having two or three copies) and are usually designed for the long term durable storage requirements of Tier 3 use cases. They are ideal for storing media files, logs, virtual machine images and backups. Object storage is highly resilient against server and disk drive failures as the data is replicated and distributed, and it can be scaled nearly infinitely. It uses a placement algorithm to maintain multiple copies of each file across disk drives on the cluster.

A well designed cloud should provide all three storage options to cost-effectively address durability and performance requirements. It should be architected for flexibility to allow customers to build OpenStack-based cloud infrastructure using one or more of the above storage types in a variety of combinations to address a spectrum of applications, use cases and cost.

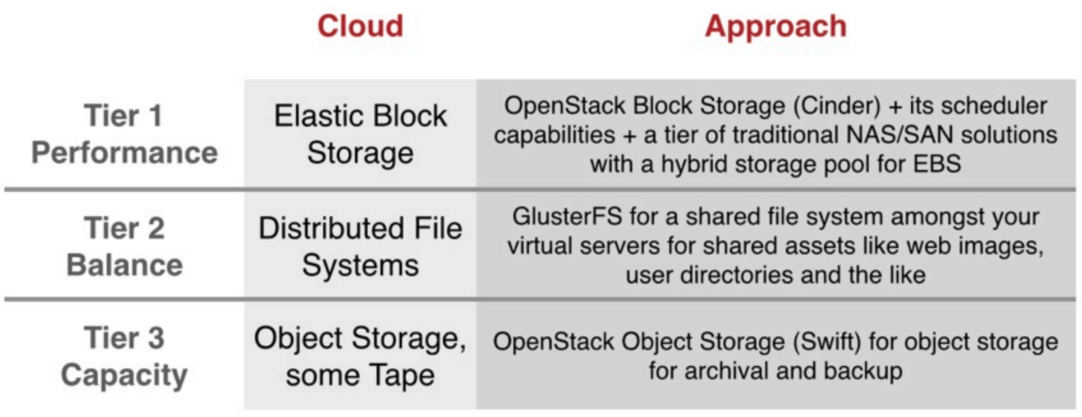
2 Again, this is a method for reducing risk by leveraging the different advantages of your storage systems together.

How Distributed Storage Should be Deployed

Distributed storage solutions are certainly worthy of being deployed in an elastic cloud. While some of the technology is still, frankly, experimental in my book, I think there are very clear and compelling use cases for it. As long as you don't try to converge your storage tiers, you can deploy them successfully.

Instead, as you are evaluating storage options for your cloud infrastructure you should be thinking in terms of the three tiers. Pick one solution for elastic block storage that is strongly consistent and network partition tolerant. Pick another solution for shared files via a distributed file system approach for your tenant's VMs. And pick yet another solution for your long term archival and storage. What might that look like?

Well, a conservative approach might be:



	Cloud	Approach
Tier 1 Performance	Elastic Block Storage	OpenStack Block Storage (Cinder) + its scheduler capabilities + a tier of traditional NAS/SAN solutions with a hybrid storage pool for EBS
Tier 2 Balance	Distributed File Systems	GlusterFS for a shared file system amongst your virtual servers for shared assets like web images, user directories and the like
Tier 3 Capacity	Object Storage, some Tape	OpenStack Object Storage (Swift) for object storage for archival and backup

Of course, you can reconfigure this a number of different ways. Ceph or Sheepdog for Tier 1, parallel NFS (pNFS) on multiple VMs for Tier 2, and tape for Tier 3. Do it whichever way you think makes sense. Just don't be under the illusion that the tiers are going away.

Summary

Enterprises are wary of managing dozens of discrete storage systems over time (as has been the norm). Naturally, the promise of managing a single, unified storage system is attractive. Unfortunately, as we have seen, a homogeneous system has its own challenges. There is, however, a sensible middle ground that involves running a handful of discrete and proven systems, each optimized for its purpose. You avoid the cost and risk of managing dozens of different systems, while avoiding the single system risk of catastrophic failure via operator errors and software bugs.

Distributed storage systems are new, fast evolving and promising technologies, but as marketed, they introduce unnecessary and avoidable risks. When we look at how cloud storage issues should be solved we want to pay attention to the lessons of history. The middle ground is to find a way to reduce operational overhead while retaining protection from catastrophic single system failures.

This is not to say that distributed storage systems don't have a place. They do, but as part of a thoughtful, tiered strategy rather than a single unified system.

Situation normal. Cloud causes a rethink.

cloudscaling

Cloudscaling
45 Belden Place
San Francisco, CA, 94104
Main: +1-877-636-8589
International: +1-415-508-3270