VERITAS

Software-defined Storage Architecture for Analytics Computing – Rev2

Arati Joshi – Performance Engineering
Carlos Carrero – Product Management

May 2016

Reference Architecture

Contents

Introduction

IT is demanding more scalability, reliability and responsiveness as the demands to manage more data, users, and applications increase. With the increase in storage and computing needs, more cost effective solutions and technologies like Software Defined Storage (SDS) are emerging in the market to solve these customer challenges.

One area not typically associated with SDS solutions is big data analytics solutions. Typical grid solutions provide highly available and parallel analytic computing environments and require a shared repository of data where every node within the grid can access and manage the data. This need has been traditionally satisfied with expensive and complex SAN architectures or lower performing NAS solutions.

SDS options are emerging to allow data center architects to design solutions to meet the unique SLA requirements of each application while avoiding the cost and performance barriers of traditional layouts. SDS utilizes commodity servers and leverages any type of server-side storage devices, while providing shared storage capabilities such as mirroring and replication. With servers increasing the slots available for internal storage, CPUs being more dense and powerful, and the internals and network carrying more data, SDS eliminates the need for storage arrays and storage area networks. While most solutions make lofty promises, none can provide a hyper-converged architecture enabling visibility, management, and data protection for the complete application and storage stack.

This white paper presents a reference architecture that suits the demands of traditional grid applications using standard servers and commodity storage. Veritas Cluster File System technology, part of the Veritas InfoScale™ Storage product, is used as the software defined storage solution enabler to create a highly available and resilient configuration for both the data and the application.

Storage Demands

Grid and analytics environments traditionally require a sustained I/O bandwidth so the different jobs being executed will complete in the timeframe required by the users. These I/O requirements differ from traditional relational databases that typically generate a very high number of small I/O requests. Grid and analytics are optimized when storage can provide high throughput for very large I/O requests driven by sequential reads and writes.

While a file system will be using server memory to cache data, it is important to measure the performance of the underlying storage. RAM is a low capacity and high cost resource with respect to the amount of data that needs to be managed. To maximize throughput for the unique I/O characteristics of this application, Veritas Cluster File System will utilize striping across the drives, issuing several I/Os in parallel. Intelligent write and read ahead will also improve performance for the file system, providing in many cases better performance than raw devices.

Veritas Cluster File System allows the creation of file systems that spread across several servers and several internal devices, creating the maximum possible parallelism. Specific parameters can be adjusted to determine how much data is going to be read ahead and how much data to read and write from each spindle. All this can affect the performance when reading and writing large files - details are provided in this white paper.

Veritas Cluster File System uses the operating system page cache to automatically buffer file system data. This means that programs using buffered I/O will get the benefit of using that cache to accelerate reads and writes. While this is a common feature for any file system, Veritas offers a feature called SmartIO, which intelligently caches data in a Flash or SSD tier local to the node. This automatically increases the amount of space allocated for caching and dramatically increases performance for files that needs to be read several times.

Performance varies dramatically when using SAN or NAS storage which are expensive and typically shared across different environments. Applications that need steady I/O performance may get impacted when other systems start using the shared hardware. For example, a random-oriented application can impact the performance if it is sharing the same underlying physical storage. The converged infrastructure presented here allocates the storage that is going to be directly accessed by the application that is running in the same hosts where the storage is locally connected.

ETL (extract, transform, and load) workloads may or may not define in advance how much storage will need to be allocated to complete their jobs. If the space is not defined in advance, the file system needs to dynamically allocate more space to the file as it grows. Veritas Cluster File System helps to avoid difficulties with this approach by allowing control of the initial and subsequent allocation extent sizes. For example, if one application is writing in 8k bock sizes, the file system can be tuned to use 256k extents in a single reservation, minimizing the overhead and reducing fragmentation. Veritas Cluster File System also allows initially allocating all the space that is going to be needed in a single extent.

Performance

The performance requirements may vary among workloads. Leading analytics applications like SAS Grid recommend a minimum I/O throughput rate of 100MB/s per core, and file systems that may need 150MB/s of average I/O. Determining the right characteristics of the storage and the file system configuration to achieve such throughput are critical to a successful configuration.

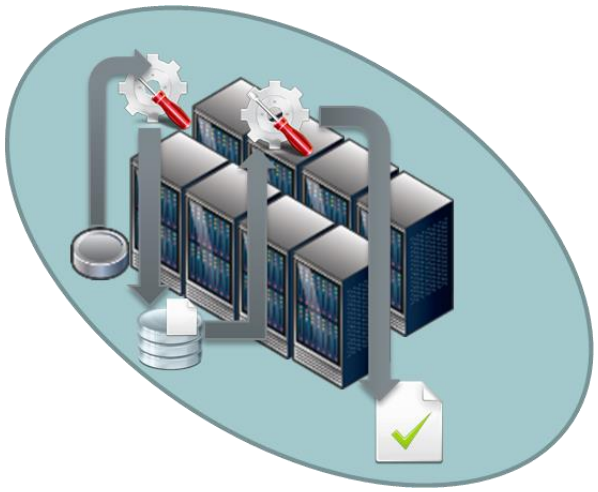
In a traditional SAN environment, determining the number of HBAs, memory cache, spindles, etc., can be a complex task. As we will see in the performance results the reference architecture presented here can easily scale compute and storage maintaining the I/O requirements for the most demanding analytics jobs.

ETL workflow

A typical scenario in an ETL work flow consists of a job that has to consume a large amount of data and run some kind of processing or ordering on it. Once that transformation is done, the data is stored so other jobs may consume it. In order to provide a scalable and efficient model, a grid infrastructure may be used, where any node may run the job. This means that any node needs to have access to the data that any other node may have produced. Having to copy data among nodes is a very expensive and lengthy process, so shared storage is used instead.

Using Software Defined Storage solution from Veritas these typical ETL jobs can be implemented using server-side storage capabilities. This new paradigm requires no changes in previous operations, but provides a significant reduction in CAPEX and OPEX.

The following graphic shows a classic ETL process where data is extracted, transformed and loaded using different nodes:



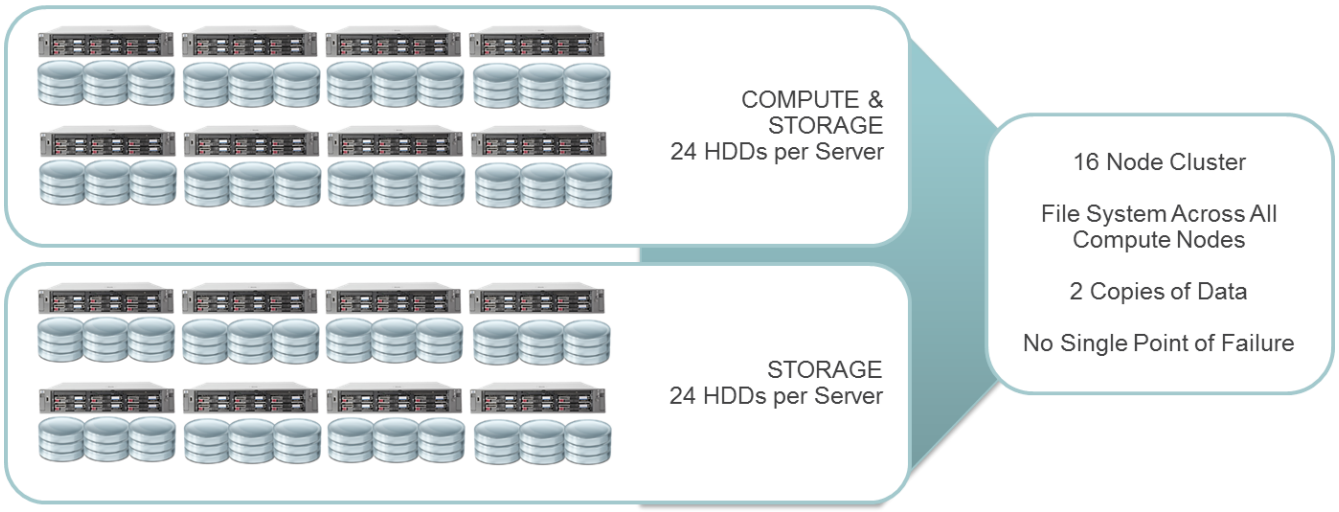
Architecture

One of the key aspects of a Software Defined Architecture is having the freedom to choose the hardware that best meets the specific business needs. The architecture should avoid any vendor lock-in and should allow future replacements or modifications when needs change.

In general the desired aspects are:

- Management
- Performance
- Scalability
- Resiliency

For our reference architecture we have used generic x86 servers with 16 and 4 cores. The 4-core servers will be used as storage nodes and keep a second copy of the data. The 16-core servers will be running jobs in a hyper-converged architecture:

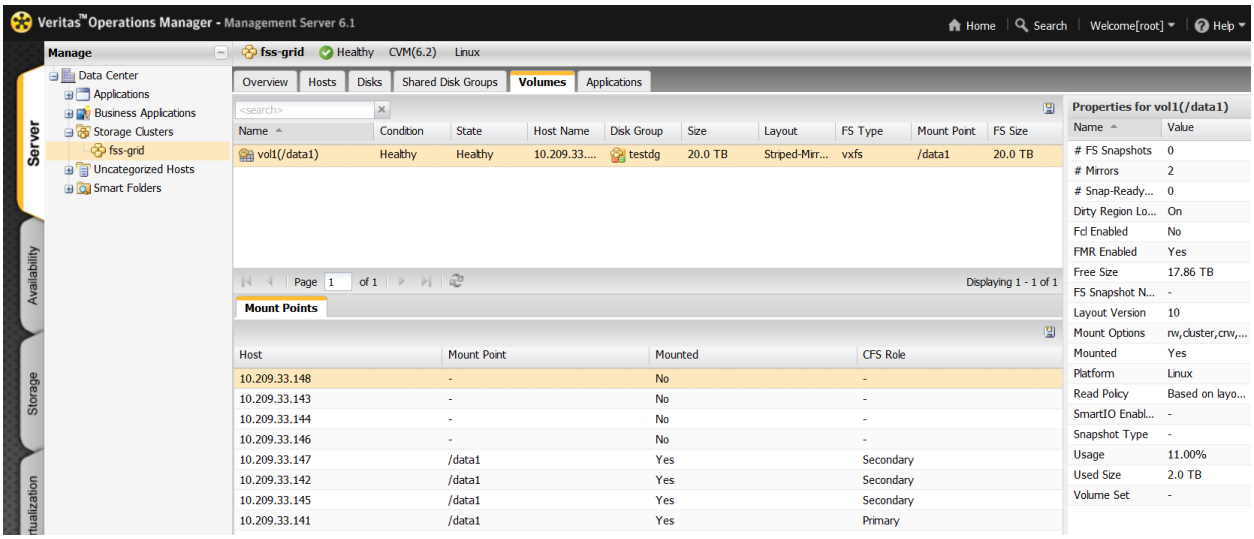


Although all the storage used is internal, Veritas' Flexible Storage Sharing technology, a part of the Veritas InfoScale Storage product, allows the use of SAN storage connected to any of the nodes if greater capacity is needed or to re-use some hardware already available. In the reference architecture discussed here there is no external storage. All the nodes are connected using one InfiniBand switch.

Management

The proposed software solution reduces the time needed to provision and consume traditional storage. Nodes simply need to be added to the cluster and automatically file systems that are accessible by all (or only the required) nodes can be automatically created. In order to simplify all the operation, a single pane of glass for operation and management can be used. Veritas InfoScale Operations Manager offers the complete visibility and management experience for Flexible Storage Sharing.

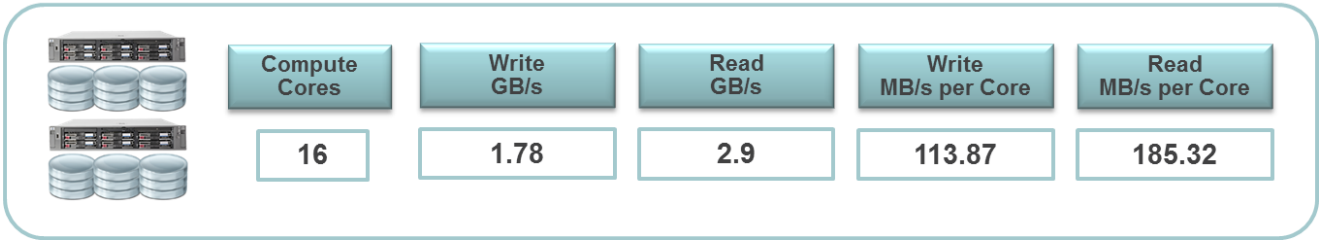
The following screenshot shows an eight node cluster where one 20TB file system is available across all the nodes.



Performance

With the technology evolution in terms of interconnects and DAS, the 100MB/s requirements from SAS Grid solutions are available now without having to use expensive hardware. The only need is the software running on the top that virtualizes all that storage, providing a highly available and performant configuration. SAS Institute provides some scripts to measure the performance of the underline hardware and verify that it will satisfy the most demanding jobs.

Using just two servers, which provide a highly available configuration, we can provide the performance needed for 16 cores.



We can see in the above table how starting with the smallest configuration (two nodes) we can achieve 1.78GB/s for reads, which translates into 113MB/s per core for writes, and 2.9GB/s for reads with 185MB/s per core.

Scalability

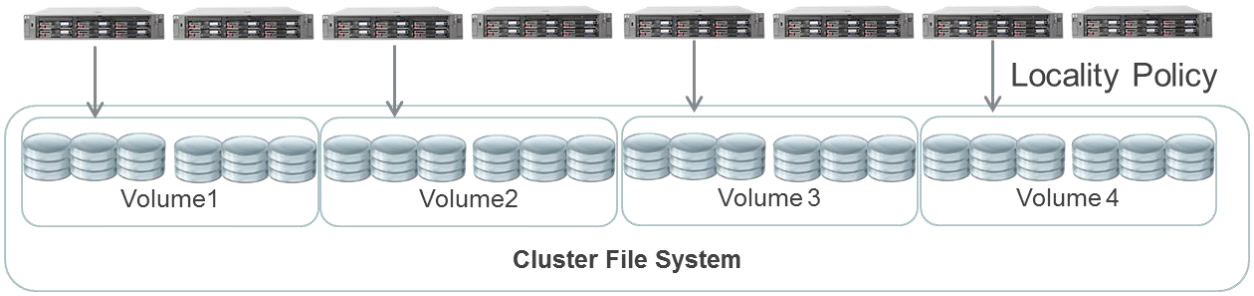
One of the advantages of using software to define a storage solution is that it allows scale up and scale out. We can add more storage in the form of DAS to any local node or we can simply add more nodes that bring either storage and/or compute. Veritas Cluster File System enables nodes to be added and the additional storage can be automatically consumed by any node within that cluster.

To demonstrate the scalability of this solution for analytics type workloads, we have increased the number of nodes from 2 to 16, increasing both capacity and computing.

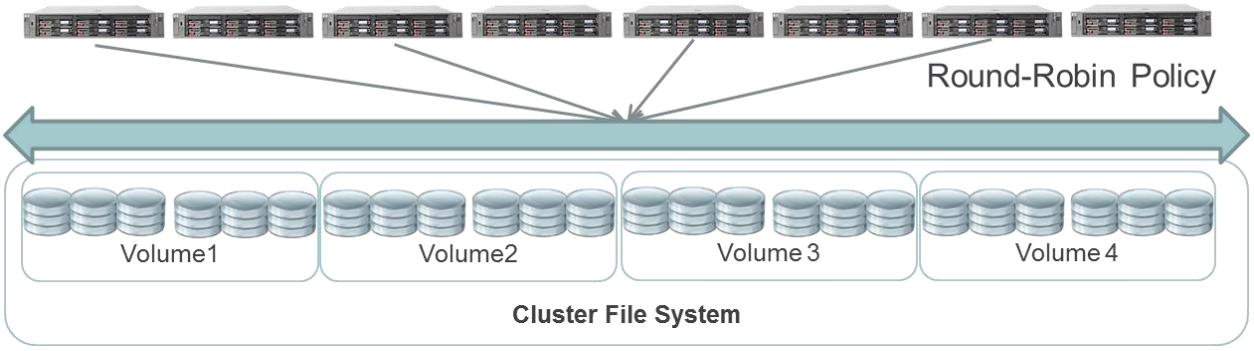
InfoScale 7.1 increases the node count up to 64 nodes and introduces better capabilities to manage local storage that needs to be shared across the cluster nodes. The first capability introduced is the Storage Access Layer that simplifies scaling either compute and/or storage.

The second capability is the utilization of Multi-Volume File Systems with local affinity. The architecture proposed in this paper, when using 16 nodes, will have to manage 384 disks. While striping data across all of them will provide the best performance, the usage of Multi-Volumes will simplify the operation and management of the platform. Multi-Volumes will be the building blocks of the platform and they will be built using two servers (ideally one compute and one storage). InfoScale 7.1 has added a new policy which allows the writes (and therefore subsequent reads) to be local to the volume that is attached to the server. But because a File System is used which is composed from all those volumes, any server can get access to that information.

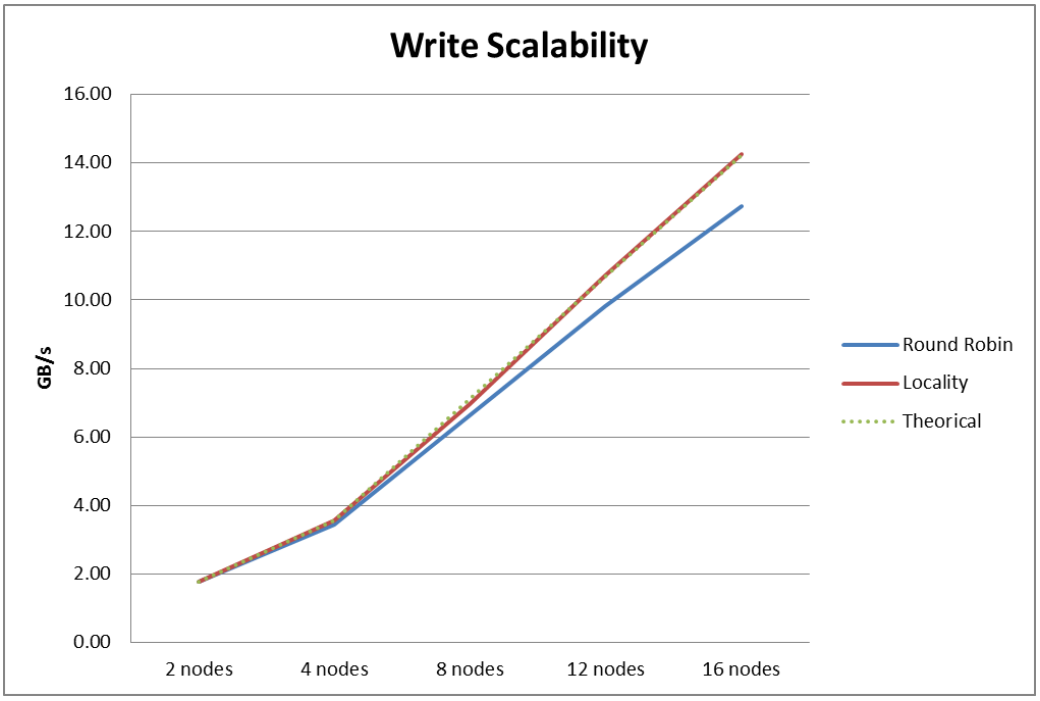
The following diagram represents one 8 node cluster using locality policy. By default, and when possible, the writes will be allocated in the closer volume to the host:



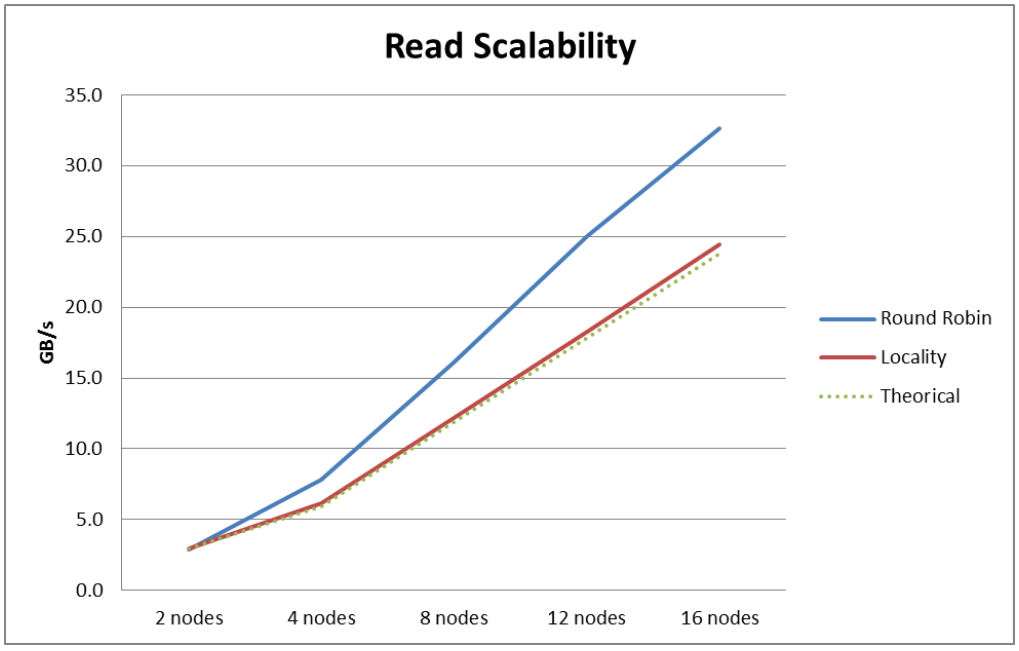
The round robin policy, will spread all the reads across all the volumes, no matter if they are local or not:



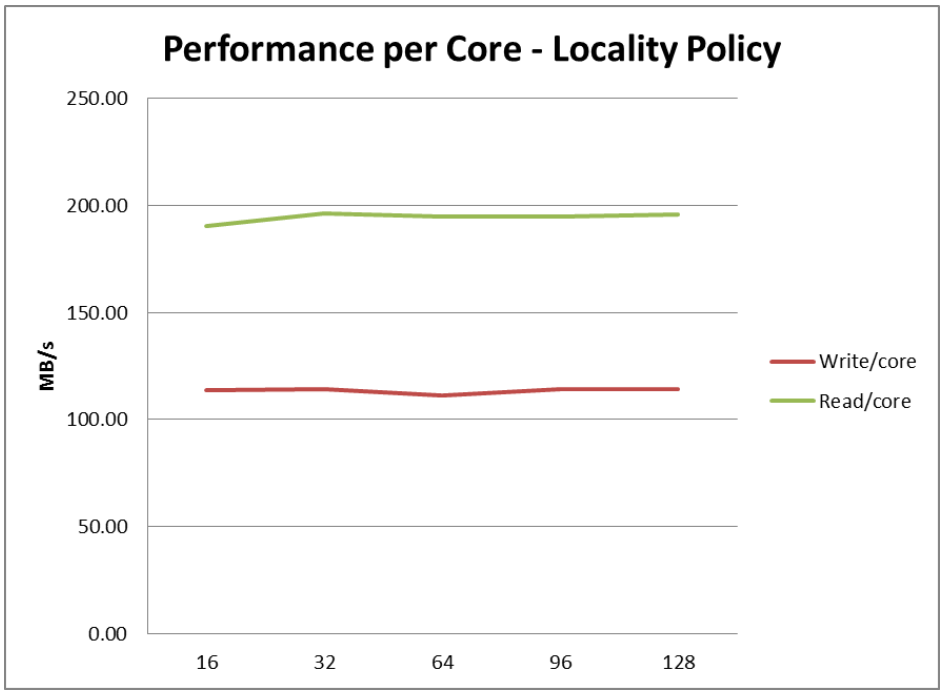
For write scalability we can see how we have a clear linear gain in scalability when using the locality policy, and almost linear when using round robin. When using 16 nodes, the architecture is able to obtain 14.27 GB/s for writes while maintaining two copies of the data for resiliency.



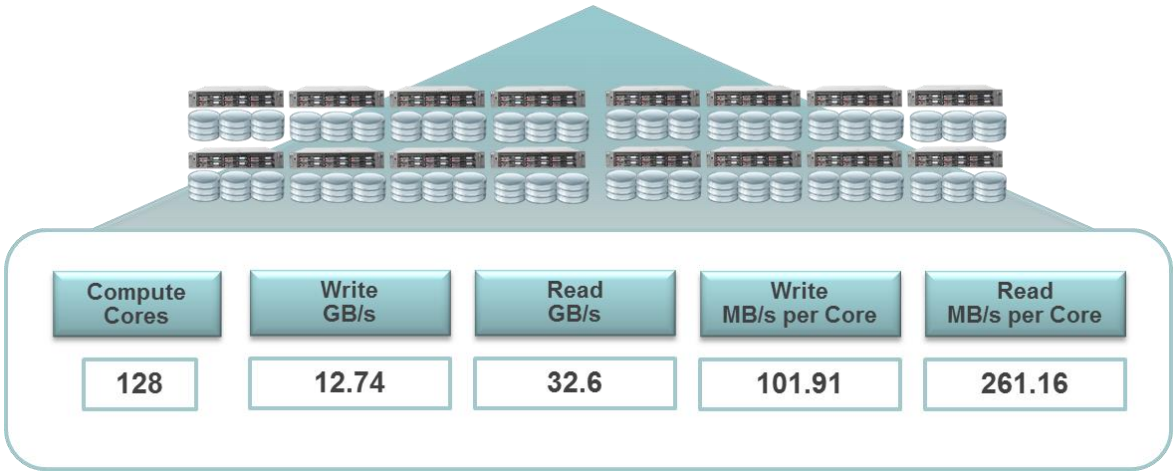
For reads, we can see that performance gains exceed linear expectations when more nodes are added. In the case of locality policy, the scalability is linear, while using round robin the theorical limit is exceeded. Because there are two copies of data, all devices are available for reads, Veritas Cluster File System allows our solution to grow reads faster than what the pure hardware provides. We can see how using 16 nodes the platform is able to read at 32.6GB/s.



While the overall performance is an important metric we have to guarantee a specific performance per core. Again we can see how the performance is maintained for both reads and writes when using locality policy for each of the cores.



The following graphic shows the numbers for a solution with 16 nodes and 128 compute cores, using a total of 384 HDDs.



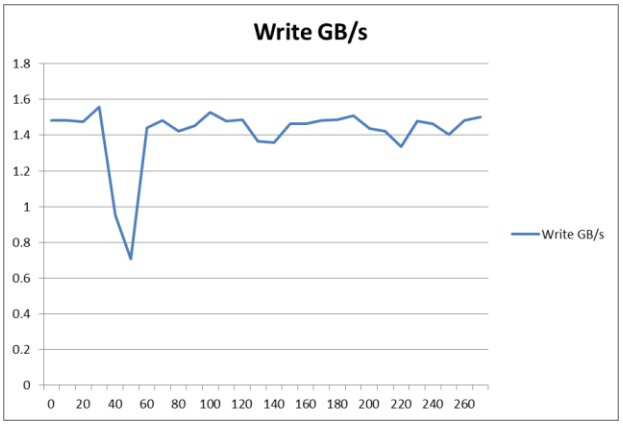
Resiliency

During file system creation, the number of copies for each piece of data can be specified. This number can be later reduced or increased, which is very useful when for example performing storage or node migrations without incurring any downtime.

The file system and volume configuration in this architecture avoid any single point of failure, so if a single disk or a complete server fails, the configuration is still up and running. Each pair of compute and storage maintain two copies of data and each one uses a multi-volume for easy of use.

One DCO (Data Change Object) region is used, whose goal is to track the changes that happen while a disk or server is offline. Only those changes are resynchronized back when the server or storage is online again.

The following graph shows the performance for a node when another node in the cluster crashes. Around second 40, the cluster needs to reconfigure itself which has a small effect in performance. Once the reconfiguration is done the performance goes backs to normal, making the server failure transparent for the application.



Configuration Details

The goal of this section is to describe some of the parameters that have been tuned for this particular configuration. LLT (Low Latency Transport) is in charge of moving the data among the different nodes. Some of their parameters have been modified in order to increase the buffer sizes and allow a higher bandwidth:

```
LLT_MAXADVBUFS=4000
LLT_ADVBUF_SIZE=65536
```

At the Volume Manager layer, the stripe unit defines how much data is sent to each disk. The layout chosen is stripe-mirror to provide higher resiliency:

```
layout=stripe-mirror
stripeunit=512k
```

From the File System point of view, we modified the default values for write_nstream. Those values are set according to the volume manager layout by default. Given the large number of columns used, that default value can be very high for this configuration.
Also, given that large files are expected in this workload, we are increasing the initial size for an extent:

```
write_nstream = 8
write_throttle = 1024
initial_extent_size = 32768
max_seqio_extent_size = 32768
```

In order to create a Multi-Volume File System, these are the steps to follow:

Initialize VSET with first volume.

```
vxvset -g testdg make vset vol1
```

Add other volumes to VSET

```
vxvset -g testdg addvol vset vol2
vxvset -g testdg addvol vset vol3
```

Make the added volume to store both data and metadata

```
fsvoladm clearflags dataonly /data1 vol2 vol3
```

In order to use the round robin policy for allocation.

```
fsapadm define -o round-robin /data1 mypolicy vol1 vol2 vol3
fsapadm assignfs /data1 mypolicy mypolicy
```

In order to use locality policy in addition of round robin:

```
fsapadm define -o round-robin -f locality /data1 mypolicy vol1 vol2 vol3
fsapadm assignfs /data1 mypolicy mypolicy
```

References

- Testing Throughput for your SAS® 9 File Systems: http://support.sas.com/kb/51/660.html
- Veritas Cluster File System: http://www.symantec.com/cluster-file-system/