Cray DVS: Data Virtualization Service

Stephen Sugiyama *and* **David Wallace**, *Cray Inc.*

ABSTRACT: *Cray DVS, the Cray Data Virtualization Service, is a new capability being added to the XT software environment with the Unicos/lc 2.1 release. DVS is a configurable service that provides compute-node access to a variety of file systems across the Cray high-speed network. The flexibility of DVS makes it a useful solution for many common situations at XT sites, such as providing I/O to compute nodes from NFS file systems. A limited set of use cases will be supported in the initial release but additional features will be added in the future.*

KEYWORDS: DVS, file systems, network

1. Introduction

The Cray Data Virtualization Service (Cray DVS) is a network service that provides compute nodes transparent access to file systems mounted on service nodes. DVS *projects* file systems mounted on service nodes to the compute nodes: the remote file system appears to be local to applications running on the compute nodes. This solves several common problems:

- Access to multiple file systems in the data center
- Resource overhead of file system clients on the compute nodes
- File system access scaling to many thousands of nodes

Lustre is the standard file system on the Cray XT platform, and is optimized for high-performance parallel I/O [Sun 2008]. Cray's Lustre implementation is intended to be used for temporary scratch storage. Most sites have one or more external file systems that contain more permanent data. Rather than copying data between file systems, DVS can be used to provide compute-node access to that data.

DVS was developed with high-performance I/O in mind. The DVS client is very lightweight in its use of system resources, minimizing the impact to the compute node memory footprint and contribution to OS jitter. The protocol used in the communication layer takes advantage of the high-performance network.

Most file systems are not capable of supporting hundreds or thousands or tens of thousands file system clients. With DVS, there may be thousands of clients that connect to a small number of DVS servers; the underlying file system sees only the aggregated requests from the DVS servers.

2. Architecture

The Cray XT series of supercomputers contain compute nodes and service nodes.

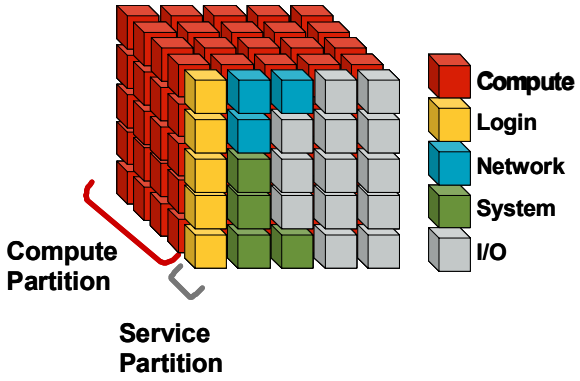

Figure 1. Node Specialization

Compute nodes do not have direct support for peripheral devices, such as disk storage, and run a limited number of operating system services. Service nodes have adapter slots for connections to external devices and run a full Linux distribution. All nodes in the system are connected through a high-speed network. This division of nodes and OS functionality helps minimize OS jitter [Wallace 2007] and maximize application performance.

DVS is composed of two main components, the DVS client, which runs on compute or service nodes, and the DVS server, which runs on I/O service nodes. The file systems configured on the DVS server are projected to the nodes with the DVS client.

The DVS client is lightweight. DVS is not a file system. DVS packages up client I/O requests and forwards them to the DVS server. When DVS is configured without data caching then DVS file data does not compete with the application for compute node memory.

The DVS server component provides a distribution layer on top of the local file system or file systems. The local file system can be located on direct-attached disk, such as the Linux ext3 or XFS file systems; the local file system can be remotely located when mounted on the service node via a service like NFS or GPFS.

DVS can be configured in several ways to suit various needs.

DVS Client	DVS Server with Simple File System (ext3, NFS)	DVS Server with Cluster File System (Lustre, GPFS)
Serial	Serial DVS	Serial DVS
Parallel	Parallel DVS	Clustered Parallel DVS

Figure 2. DVS Configurations

The table in Figure 2 describes the main possible configurations:

- Serial DVS means that a single DVS server projects a file system to all DVS clients using that file system.
- Clustered Parallel DVS means that multiple DVS servers project a common cluster file system to all DVS clients using that file system.
- Parallel DVS means that multiple DVS servers project a portion (stripe) of a file system to all DVS clients using that file system

These configurations are described in more detail below.

2.1 Serial DVS

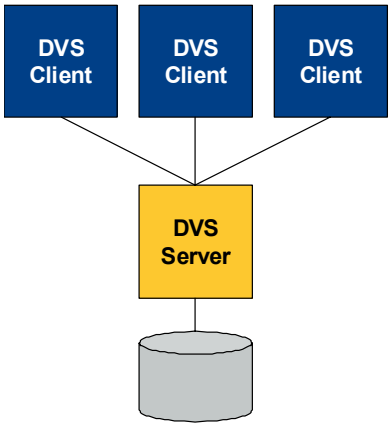

Figure 3. Serial DVS

A simple Serial DVS configuration is illustrated in Figure 3. There are multiple DVS clients, each communicating to a DVS server that projects a file system mounted on the server. The file system could be a simple (non-clustered) file system such as XFS; the file system could be a remote file system mounted via NFS.

Multiple file system can be projected to the clients through one or more DVS servers. For example, two separate NFS file systems could be made available to the DVS clients either by mounting both on a single DVS server or by using two DVS servers, each mounting one of the NFS file systems. Of course, the approach using multiple DVS servers will have better performance.

2.2 Clustered Parallel DVS

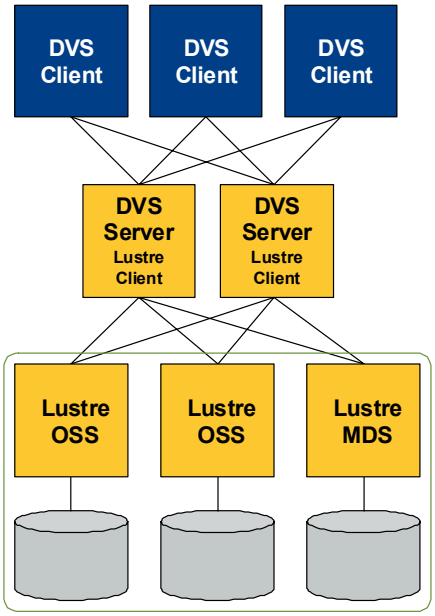

Figure 4. Clustered Parallel DVS

A Clustered Parallel DVS configuration is illustrated in Figure 4. In this arrangement multiple DVS servers project a single clustered file system to the DVS clients. The DVS clients can spread their I/O traffic between the DVS servers using a deterministic mapping. The DVS servers talk to the cluster file system. It is also possible to configure the DVS clients to use a subset of the available DVS servers. This may be useful to balance the client load on the servers when there are a large number of clients.

One benefit of Clustered Parallel DVS is that it reduces the number of clients that communicate with the backing file system. This is important for file systems such as GPFS, which only supports a limited number of clients.

Another benefit of using DVS on top of a cluster file system is that coherency and lock traffic is eliminated because any given file is always accessed from the same file system client (DVS server).

2.3 Parallel DVS

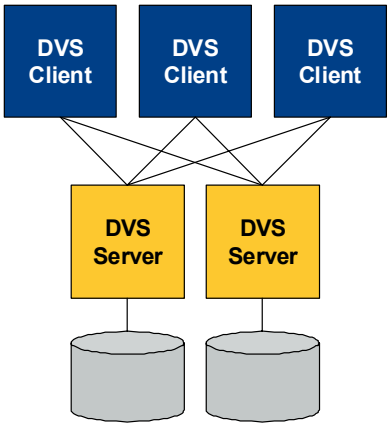

Figure 5. Parallel DVS

A Parallel DVS configuration is illustrated in Figure 5. In this configuration, multiple DVS servers are each backed by a simple file system. There are two ways to operate a Parallel DVS configuration: whole file and striped file.

In whole file mode, files are distributed among the DVS servers. Any given file is located on one of the servers. Whole file mode spreads the I/O load among the DVS servers.

In striped file mode, DVS divides the I/O into "stripes" that are transferred to the DVS servers and written to the backing file system; when reading data the DVS client reassembles the stripes transferred from each DVS server. Striped file mode spreads the I/O across multiple storage devices for parallel data movement. Parallel DVS with striped files can achieve very high performance when configured with many DVS servers.

3. Implementation

DVS has two main components: client and server. The DVS client fits under the Linux Virtual File System (VFS) on the application node; the DVS server fits over the VFS on the service node; the DVS client and DVS server communicate over the Cray SeaStar high-speed network.

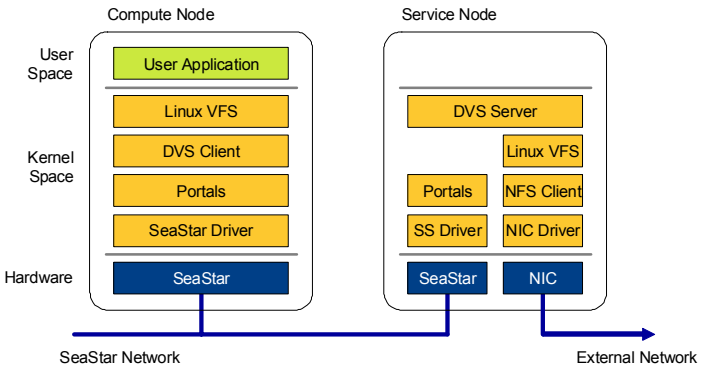

Figure 6. Software Stack

The main software components are shown in Figure 6. Applications perform I/O through the Linux VFS. The VFS allows Linux to support many file systems through a common interface. Applications use standard Linux I/O calls, such as open(), read(), and write(); the VFS vectors those I/O requests to the appropriate file system code.

Under the VFS is the DVS client. The DVS client is a Linux kernel module that appears as a file system to the Linux VFS. The DVS client packages up the I/O requests into messages that are sent across the network to the DVS server.

The DVS communication subsystem is layered on the transport mechanism of the host system. On Cray XT systems a RDMA interface to the Portals protocol is used to interface with the SeaStar network [Brightwell 2005]. Large requests, such as those greater than 1 MB, are efficiently handled.

The DVS server also operates in the Linux kernel space. It communicates with the DVS client and performs I/O through the Linux VFS. Any file system mounted via the Linux VFS can be used by DVS. Figure 6 shows NFS but it could have been ext3, XFS, or any other Linux file system. Metadata, such as file names, access times, etc., are managed by the underlying file system.

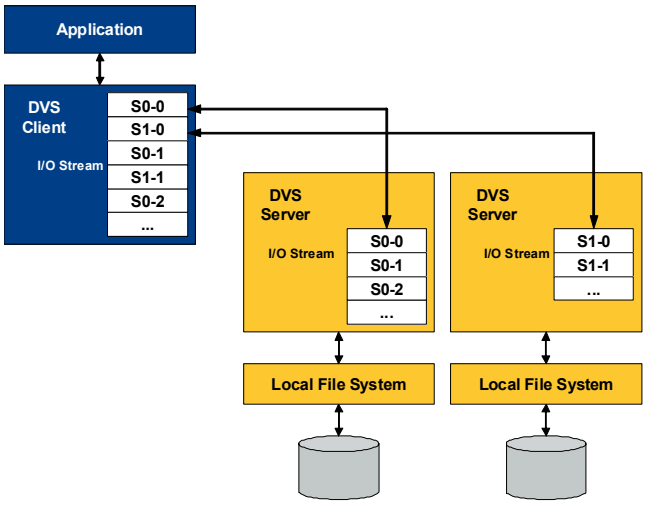

Figure 7. Parallel DVS I/O

In a Parallel DVS configuration with striped files DVS manages the striping of data across multiple DVS servers. This operation is illustrated in Figure 7. The application reads and writes a stream of data. The DVS client divides the stream into stripes and sends the stripes to the appropriate DVS server to be stored in the local file system. On reading data the DVS client requests the stripes from the servers and reassembles the stream. A few items to note:

- Striping is a fixed mapping computed by DVS without any metadata
- The DVS server with the starting stripe is selected by a hash to avoid hot-spots
- DVS server nodes do read-ahead and write aggregation in parallel
- Files on the local file system only contain partial data: a DVS client is needed to reassemble stripes into whole files

Although DVS has the ability to stripe files across multiple servers to perform parallel I/O, it is not a cluster file system. It does not maintain any metadata. It does not perform any locking. It does not do any client cache coherency. All disk allocation and disk I/O is performed by the underlying file system. This simplicity allows DVS to scale and to achieve high performance.

4. Current Results

DVS has been running in field trials at several sites over the past six months. It has demonstrated scaling to over 7000 client nodes projecting a NFS file system from a single DVS server. Obviously, the bandwidth supporting these clients is restricted by the single communication channel but this illustrates the capability of DVS to operate at a scale that is out of reach for NFS.

The following charts are recent performance tests of DVS. Tests were performed using the bringsel benchmark [Kaitschuck 2007]. The test system was a Cray XT4 running Unicos/lc 2.1.07 pre-release software. The storage hardware is a DDN S2A85000 RAID connected to the system via 2 Gb/s Fibre Channel links. The DVS servers on the XT4 service nodes were mounted with ext2 file systems. These directly-mounted file systems were used to eliminate the overhead of NFS from the measurements.

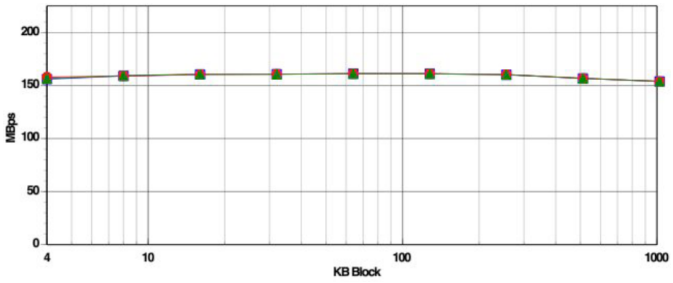

Figure 8. DVS Server Performance

The chart in Figure 8 shows the performance of the DVS server performing I/O to the mounted ext2 file system. This measures I/O without the DVS client or network. The server achieves approximately 160 MB/s for data sizes ranging from 4 KB to 1 MB. This transfer rate is close to the limit of the 2 Gb/s Fibre Channel connection.

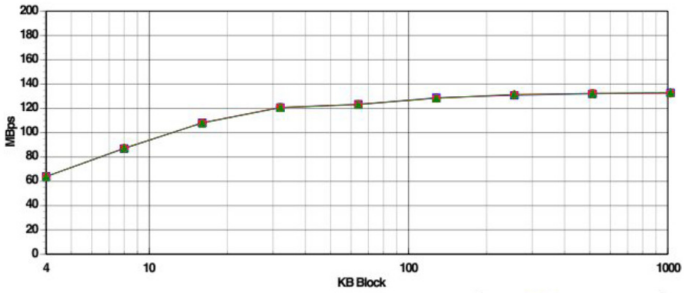

Figure 9. DVS Client-Server Performance

The chart in Figure 9 shows the performance of a single DVS client performing I/O that is forwarded to the DVS server and to the mounted ext2 file system. I/O performance runs from approximately 60 MB/s for small 4 KB blocks up to approximately 130 MB/s for block sizes of 128 KB and larger.

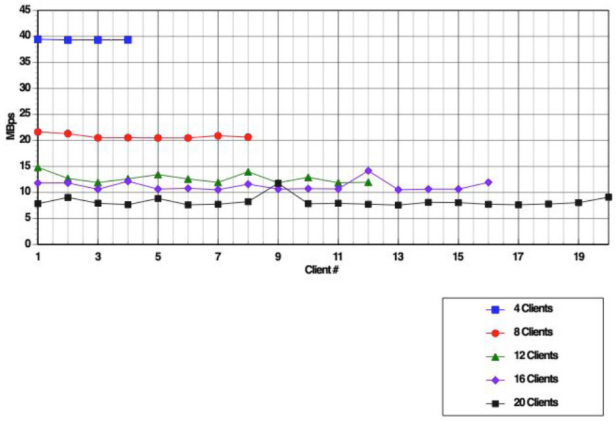

Figure 10. DVS Multi-Client Performance

The chart in Figure 10 shows the performance of multiple DVS clients forwarding to a single DVS server that has an ext2 file system mounted. Each point on the chart shows the transfer rate of that particular client, so the aggregate transfer rate is the sum of each set. For example, the “4 Clients” line shows that each of the four clients achieves approximately 40 MB/s. Note that the aggregate transfer rate is approximately 160 MB/s for all client counts from 4 to 20.

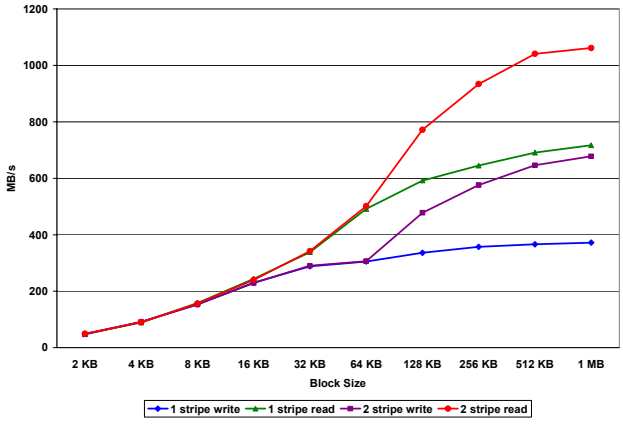

Figure 11. Striped Parallel DVS Performance

The chart in Figure 11 shows the performance of a single DVS client to one and two DVS servers set up in a striped Parallel DVS configuration. The application was

ioperf, a simple program that reads and writes blocks of data. The data block size was varied from 2 KB to 1 MB for both reads and writes. At the larger block sizes the configuration with data striping across two servers shows a significant performance increase over the single server configuration for both reads and writes. Some of the very high rates achieved are due to the read-ahead and caching done on the DVS server.

5. Future Directions

DVS is very flexible and has many possible uses. The upcoming Unicos/lc 2.1 software release with the Cray Linux Environment (CLE) will be the first release to support DVS. In this first release only the Serial DVS to NFS file systems configuration will be supported. But DVS has many capabilities that can be further developed in the future.

Expanded file system support – Anyone who has worked with a variety of file systems will realize that different file systems have widely different characteristics. It isn't possible to treat NFS just like GPFS, for example. DVS needs to be tuned to work well with each file system. Future releases may provide support for more file systems, such as GPFS, Lustre, QFS, PanFS, etc.

Parallel DVS and Clustered Parallel DVS – These configurations are available but not yet supported.

Server failover – Adding support for server failover will increase file system availability.

External DVS servers – Instead of porting a file system to the XT system the DVS server could be ported to an external system. This would require an intermediate network module to run on a service node and forward between the internal and external fabrics.

Stacked servers – DVS can support a large number of clients with a small number of servers. Looking farther out into the future, stacking DVS on top of DVS is a potential way to increase scalability for exascale systems.

6. Summary

DVS is a high-performance I/O forwarding service that provides transparent compute-node access to service-node mounted file systems. DVS can solve the problem of accessing a diverse set of file systems on XT compute nodes. The DVS implementation is lightweight and scales to many thousands of nodes. On Cray XT systems DVS takes advantage of the high-performance SeaStar network. Early performance measurements show good results. Although the first release of DVS supports a limited set of features DVS is flexible and can be configured or enhanced to address many file system issues.

Acknowledgments

The authors would like to thank Kitrick Sheets and Tim Cullen for their work on DVS and John Kaitschuck for his efforts in benchmarking DVS.

References

[Brightwell 2005] *Implementation and Performance of Portals 3.3 on the Cray XT3*, R. Brightwell, T. Hudson, K. Pedretti, R. Riesen, K.D. Underwood, Cluster Computing 2005.

[Kaitschuck 2007] *Bringsel: A Tool for Measuring Storage System Reliability, Uniformity, Performance and Scalability*, John Kaitschuck and Mathew O'Keefe, CUG 2007.

[Sun 2008] *Lustre Operations Manual*, Sun Microsystems, 2008.

[Wallace 2007] *Compute Node Linux: Overview, Progress to Date & Roadmap*, David Wallace, CUG 2007.

About the Authors

Stephen Sugiyama is an Engineering Manager with Cray Inc. and can be reached at sss@cray.com. David Wallace is a Technical Program Lead with Cray Inc. and can be reached at dbw@cray.com.