Table of Contents

About This Document .........................................................................................................................9
  Intended Audience .........................................................................................................................9
  Document Organization ...............................................................................................................9
  Typographic Conventions ...........................................................................................................9
  Related Information .....................................................................................................................10
  Publishing History .......................................................................................................................10
  HP Encourages Your Comments .................................................................................................10

1 Introduction
  Where SVA Fits in the High Performance Computing Environment .............................................11
  SVA Clusters ...............................................................................................................................12
  Displays ......................................................................................................................................13
  SVA Functional Attributes .........................................................................................................13
    Scalability .................................................................................................................................13
    Flexibility .................................................................................................................................13
  Application Support ....................................................................................................................14
    OpenGL Applications ..............................................................................................................14
    Scenegraph Applications ........................................................................................................15

2 SVA Architecture
  SVA as a Cluster ..........................................................................................................................17
    Background on Linux Clusters ...............................................................................................17
  Architectural Design ..................................................................................................................17
    Components of the HP Cluster Platform .............................................................................18
    Main Visualization Cluster Tasks .........................................................................................18
    Components of an SVA ...........................................................................................................19
    Configuration Flexibility ........................................................................................................19
  SVA Operation ............................................................................................................................19
    Cluster Data Flow ....................................................................................................................19
    File Access ...............................................................................................................................20

3 SVA Hardware and Software
  Hardware Component Summary .................................................................................................21
  Network Configurations .............................................................................................................22
    System Interconnect (SI) .........................................................................................................22
    Administrative Network Connections ....................................................................................22
  Display Devices .........................................................................................................................22
  SVA Software Summary .............................................................................................................23
    Linux Operating System ..........................................................................................................23
    HP XC Clustering Software ....................................................................................................24
    Additional System Software ....................................................................................................24

4 Setting Up and Running a Visualization Session
  Configuration Data Files ............................................................................................................27
  Running an Application Using Scripts ......................................................................................27
    Selecting a Template ................................................................................................................28
    Modifying a Script Template ....................................................................................................28
    Using a Script to Launch an Application ................................................................................29
  Running an Interactive Session .................................................................................................29
List of Figures

1-1 System View of a Computing Environment with Integrated SVA.................................................................11
1-2 Standalone SVA Data Flow..........................................................................................................................12
1-3 Software Support for Application Development and Use.............................................................................14
2-1 SVA Data Flow Overview..........................................................................................................................20
3-1 Sample SVA Bounded Configuration........................................................................................................22
3-2 Software Hierarchy in the SVA..................................................................................................................23
5-1 Using a Single SVA Node from Local Desktop..........................................................................................32
5-2 ParaView Flow of Control on the SVA.......................................................................................................37
5-3 Processes Running with Chromium-DMX Script......................................................................................41
List of Tables

3-1 Operating System and Driver Components .............................................................. 24
3-2 HP XC System Components Relevant to SVA Operation ........................................ 24
3-3 HP SVA System Software ...................................................................................... 25
3-4 Third Party System Software ................................................................................ 25
3-5 Application Development Tools .......................................................................... 25
5-1 Comparison Summary of Application Scenarios .................................................. 31
About This Document

The *SVA User's Guide* introduces the components of the HP Scalable Visualization Array (SVA). The SVA product has hardware and software components that together make up the HP high performance visualization cluster. This document provides a high level understanding of SVA components.

The main purpose of the SVA is to give HP customers a platform on which to develop and run graphics applications that require high performance combined with large data throughput on single or multi-tile displays.

Intended Audience

The *SVA User's Guide* is intended for all users of the SVA. This includes visualization application developers, visualization application users, system managers, and technical managers who need a high level understanding of SVA.

Document Organization

This manual is organized into the following sections:

- **Chapter 1** Overview of SVA and where it fits in the HP Cluster Platform environment. It also describes attributes of the SVA.
- **Chapter 2** Overview of SVA architecture, hardware, and software that make up the system.
- **Chapter 3** Additional detail on the hardware and software that make up the SVA.
- **Chapter 4** Description of how to run a visualization application on the SVA.
- **Chapter 5** Description of common application examples as well as how to set them up on the SVA.

Typographic Conventions

This document uses the following typographical conventions:

<table>
<thead>
<tr>
<th>%, $, or #</th>
<th>A percent sign represents the C shell system prompt. A dollar sign represents the system prompt for the Bourne, Korn, and POSIX shells. A number sign represents the superuser prompt.</th>
</tr>
</thead>
<tbody>
<tr>
<td>audit(5)</td>
<td>A manpage. The manpage name is <em>audit</em>, and it is located in Section 5.</td>
</tr>
<tr>
<td>\ (backslash)</td>
<td>Indicates the continuation of a command, where the line is too long for the current page width.</td>
</tr>
<tr>
<td>Command</td>
<td>A command name or qualified command phrase.</td>
</tr>
<tr>
<td>Computer output</td>
<td>Text displayed by the computer.</td>
</tr>
<tr>
<td>Ctrl+x</td>
<td>A key sequence. A sequence such as <em>Ctrl+x</em> indicates that you must hold down the key labeled <em>Ctrl</em> while you press another key or mouse button.</td>
</tr>
<tr>
<td>ENVIRONMENT VARIABLE</td>
<td>The name of an environment variable, for example, <em>PATH</em>.</td>
</tr>
<tr>
<td>ERROR NAME</td>
<td>The name of an error, usually returned in the <em>errno</em> variable.</td>
</tr>
<tr>
<td>Key</td>
<td>The name of a keyboard key. <em>Return</em> and <em>Enter</em> both refer to the same key.</td>
</tr>
<tr>
<td>Term</td>
<td>The defined use of an important word or phrase.</td>
</tr>
<tr>
<td>User input</td>
<td>The name of a placeholder in a command, function, or other syntax display that you replace with an actual value.</td>
</tr>
<tr>
<td>Variable</td>
<td>The contents are optional in syntax. If the contents are a list separated by a pipe (</td>
</tr>
<tr>
<td></td>
<td>The contents are required in syntax. If the contents are a list separated by a pipe (</td>
</tr>
<tr>
<td>...</td>
<td>The preceding element can be repeated an arbitrary number of times.</td>
</tr>
<tr>
<td></td>
<td>Indicates the continuation of a code example.</td>
</tr>
<tr>
<td></td>
<td>Separates items in a list of choices.</td>
</tr>
</tbody>
</table>
WARNING  A warning calls attention to important information that if not understood or followed will result in personal injury or nonrecoverable system problems.

CAUTION  A caution calls attention to important information that if not understood or followed will result in data loss, data corruption, or damage to hardware or software.

IMPORTANT  This alert provides essential information to explain a concept or to complete a task

NOTE  A note contains additional information to emphasize or supplement important points of the main text.

Related Information

Related documentation is available via links from the home page for the SVA Documentation Library. It also includes links to third party documentation available on the Web that is relevant to users of SVA.

Publishing History

The document printing date and part number indicate the document’s current edition. The printing date will change when a new edition is printed. Minor changes may be made at reprint without changing the printing date. The document part number will change when extensive changes are made. Document updates may be issued between editions to correct errors or document product changes. To ensure that you receive the updated or new editions, subscribe to the appropriate product support service. See your HP sales representative for details. You can find the latest version of this document on line at:


<table>
<thead>
<tr>
<th>Manufacturing Part Number</th>
<th>Supported Operating Systems</th>
<th>Supported Versions</th>
<th>Edition Number</th>
<th>Publication Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>A-SVAUG-2A</td>
<td>HP XC System Software</td>
<td>Version 1.1</td>
<td>1</td>
<td>January, 2006</td>
</tr>
<tr>
<td></td>
<td>Version 3.0</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

HP Encourages Your Comments

HP encourages your comments concerning this document. We are committed to providing documentation that meets your needs. Send any errors found, suggestions for improvement, or compliments to:

feedback@fc.hp.com

Include the document title, manufacturing part number, and any comment, error found, or suggestion for improvement you have concerning this document.
1 Introduction

This chapter gives an overview of the HP Scalable Visualization Array (SVA). It describes how the SVA works within the context of overall HP cluster solutions. It also discusses attributes of the SVA that make it a powerful tool for running data intensive graphics applications.

The SVA is a scalable visualization solution that brings the power of parallel computing to bear on many demanding visualization challenges.

The SVA leverages the advances made across the industry in workstation class systems, graphics technology, processors, and networks by integrating the latest generations of these components into its clustering architecture. This base of scalable hardware underlies powerful Linux clustering software from HP. It is further enhanced by a set of utilities and support software developed by HP and its partners to facilitate the use of the system by new and existing user applications.

Where SVA Fits in the High Performance Computing Environment

The SVA is an HP Cluster Platform system. It can be a specialized, standalone system consisting entirely of visualization nodes, or it can be integrated into a larger HP Cluster Platform system and share a single System Interconnect with the compute nodes and a storage system. Either way, the SVA can integrate seamlessly into the complete computational, storage, and display environment of customers as shown in Figure 1-1.

Figure 1-1 System View of a Computing Environment with Integrated SVA

High-speed networks make feasible the transfer of large amounts of data among the following:

- Individual users at their desktops, or logged into a cluster.
- The compute cluster, the visualization cluster, and local and remote display devices.
- Servers that are part of data storage farms.

A typical usage model for the type of system shown in Figure 1-1 has the following characteristics:

- A compute intensive application, for example, an automobile crash test simulation, runs on the supercomputing compute nodes of the cluster.
- The large dataset generated on the compute nodes can be stored in the storage servers for later retrieval, or directed in realtime for rendering on the SVA portion of the overall system.
- One or more users can log into the SVA concurrently, which allocates resources efficiently to meet the rendering and display requirements of each user application.
- Users’ visualization applications use parallel programming techniques and visualization middleware software to distribute their graphical rendering across the SVA nodes, each of which in turn renders a portion of the output for the final image. Image data can be apportioned by a master application to a set of visualization nodes for rendering.
- Each portion of the final image rendered by a visualization node is sent to a tile of a single or multi-tile display. The complete image is available for display locally. The complete image is also available for display remotely, but limited to single or two-tile output from a single graphics card.
The SVA serves as a key unit in an integrated computing environment that displays the results of generated data in locations where scientists and engineers can most effectively carry out analyses individually or collaboratively.

### SVA Clusters

This section gives a high-level description of a standalone SVA, that is, an HP Cluster Platform system built using visualization nodes. The SVA can also provide a visualization solution that is fully integrated into an existing HP Cluster Platform system with compute and storage components, as shown in Figure 1-1.

The SVA image-based approach works with a variety of visualization techniques, including isosurface extraction and volume visualization. Such a graphics architecture combines the high performance of clusters of rendering machines with the interactivity made possible by the speed, scalability, and low latency of the cluster network.

HP SVA offers a graphics visualization solution that can be used by a variety of applications that run on distributed computing systems; in this case, a cluster of Linux workstations. Figure 1-2 illustrates the makeup of a standalone SVA.

**Figure 1-2  Standalone SVA Data Flow**

Key points of Figure 1-2 are the following:

- Industry standard workstations with standard OpenGL 3D graphics cards serve as visualization nodes (render and display), and run clustering software and Linux. Use of industry standard graphics cards lets the system take advantage of new generations of cards as they become available.
- Depending on the design of the application, an application “master” can run the application and the user interface for the application on a specified node.
- Display nodes transfer their rendered output to the display devices and can synchronize multi-tile displays. A range of displays are supported at locations local and remote to the SVA. A series of render nodes can also contribute composited images to the display nodes, depending on the visualization application.
- The System Interconnect (SI) supports data transfer among visualization nodes. High-speed, low-latency networks such as InfiniBand and Myrinet can be used for the SI to speed the transfer of image data and drawing commands to the visualization nodes.

Each portion of an image is rendered on its visualization node as determined by the application and the visualization middleware being used. For example, you can use Chromium or a scenegraph application in conjunction with Distributed MultiHead X (DMX). The final images are transmitted by the graphics cards in the display nodes to the display devices.

Final images can also be transmitted to a remote workstation display over a network external to the cluster. This lets users interact with applications running on the cluster from their offices. Optionally, you can use HP Remote Graphics Software (RGS) to accomplish this more easily.

Figure 1-2 also shows a master application node communicating with the other visualization nodes over the SI. The SI carries file I/O and application communications; for example, MPI traffic. The user interface for a visualization application can run on a master application node and communicate with the visualization nodes over the SI, sending control information such as changes in point of view, data, or OpenGL commands.
Displays

The SVA supports a wide range of displays and configurations, including single displays, tiled displays in walls, and immersive CAVE environments. The SVA relies on the display capabilities of the graphics cards in the display nodes. This means that the SVA lets you use whatever display devices are supported by the graphics card.

Depending on the demands of the display devices, you can use digital or analog output. The aggregate resolution of these displays can range from 10s to 100s of megapixels.

The SVA supports up to eight display nodes1 in a Display Surface. The display nodes in your cluster can drive one or two display devices in the case of xw8200 nodes, and one to four display devices in the case of xw9300 nodes. This means that you can drive a maximum of 32 display devices using eight xw9300s. See the SVA System Administration Guide for more information on setting up and cabling display nodes, display devices, and Display Surfaces.

SVA Functional Attributes

The key to SVA scalability and flexibility is its combination of cluster technology with high-speed graphics cards and networks to transfer data. The SVA enables scaling up the number of nodes working on a problem in parallel to handle larger dataset sizes, to increase frame rates, and to display at higher image resolutions.

Scalability

There are a number of ways that applications can be designed and implemented to take advantage of an SVA for effective scaling:

- Performance scaling: Render image data on separate nodes in the SVA. In effect, the work is divided up among nodes working in parallel. Larger datasets can be accommodated by more render nodes. The system design can scale from four to forty visualization nodes. This count does not include the required head node.

  The parallel attributes of the rendering pipeline removes a key performance bottleneck of a conventional hardware accelerated graphics architecture, which feeds data sequentially to a centralized pipeline. In addition, the choice of a network that transmits data among the visualization nodes with adequately low latency and high speed maintains interactive frame rates for delivery to the display devices.

- Resolution scaling: Parallel rendering, combined with the parallel display of multiple tiles makes such scaling possible. You can display high-resolution data and use large display surfaces, including immersive displays and display walls.

  In general, adding nodes to a dataset of fixed size provides good scaling up of the frame rate, although speed-up is not linear because of the inevitable overhead due to portions of an application’s code that cannot be made parallel. However, a strength of SVA as a cluster visualization platform is that scalability is nearly linear when the dataset size and node count are both increased. For example, doubling the node count from four to eight makes it possible to double the distributed dataset size with virtually no loss of frame rate. To achieve such gains in frame rate, an application must be a true parallel application to efficiently distribute data and to load balance across cluster nodes.

Flexibility

One of the most powerful attributes of the SVA is its flexibility, which makes it possible to apply the SVA effectively to a wide range of technical problems. This flexibility derives from the architectural characteristics of the SVA.

When the architectural characteristics of the SVA are integrated with an HP high performance compute cluster (see Figure 1-1), you can select an optimal number of application or compute nodes and match them with an appropriate number of render and display nodes. Visual applications with high computation requirements can be distributed over the compute nodes and the visualization nodes; thus the render nodes can double as compute nodes.

This flexibility is critical because visualization applications often need to perform intensive computations to compute isosurfaces, streamlines, or particle traces. You can select application nodes based on factors such as computational load.
as model size, and match them to the visualization nodes your application needs to yield the desired performance and resolution.

Application Support

This section introduces software support for application developers. Chapter 3 contains more information on the software tools available for application developers.

HP recognizes that a key capability of the SVA is to make it possible for serial applications to run without extensive recoding. To that end, HP works with both commercial ISVs and the open source community to ensure solutions are available for the SVA.

Figure 1-3 illustrates the layers of software support and their hierarchical interrelationships that are part of the SVA. These include:

- Cluster management software (HP XC) and visualization resource management software (SVA Software Utilities).
- Visualization toolkits and libraries.
- User and third-party visualization applications.

Figure 1-3 also shows the tasks carried out by the SVA Software Utilities (part of the Visualization System Software (VSS)). These tasks — allocate, launch, initialize, cleanup — are aligned alongside the software layers they impact.

Visualization and graphics toolkits are provided by third party vendors and the open source community. ISV applications and applications written by end users can run on the SVA, taking full advantage of the various toolkits and libraries. The SVA uses standards such as OpenGL, Linux, InfiniBand, and Gigabit Ethernet for portability and interoperability.

To achieve maximum performance scaling when running on the SVA, an application must be parallel and distributed. There are two main pathways to this state: applications made parallel by design and serial applications made parallel automatically through middleware libraries or toolkits; for example, Chromium or other middleware.

OpenGL Applications

If your application is already parallel and distributed, you can use OpenGL directly.

Most visualization applications support OpenGL directly or through graphics toolkits. Autoparallel toolkits such as Chromium, enable standard OpenGL applications to run on an SVA with increased resolution, although without the performance advantages of a true parallel application.
Scenegraph Applications

The SVA lets you take advantage of scenegraph applications available through scenegraph middleware libraries and toolkits. The result is that the application is available on the SVA and can take advantage of its parallel scalability features.
2 SVA Architecture

This chapter gives a detailed look at the architecture of the HP Scalable Visualization Array (SVA). It compares the SVA to other clusters and describes the flow of data within the cluster.

SVA as a Cluster

It is important to understand the cluster characteristics of the SVA. These characteristics have implications for how SVA functions. They also affect how applications take advantage of cluster features to achieve graphical performance and display goals.

Background on Linux Clusters

In the taxonomy of parallel computers, the SVA is most similar to a Beowulf class Linux cluster. Beowulf clusters have many servers of the same type that communicate on high speed connections such as channel bonded Ethernet. In this way, the cluster provides high performance for applications capable of using parallel processing. This type of cluster can provide exceptional computational performance.

A Beowulf cluster falls somewhere between the class of systems known as Massively Parallel Processors (MPP) and a network of workstations (NOW). Examples of MPP systems include the nCube, CM5, Convex SPP, Cray T3D, and Cray T3E. Beowulf clusters benefit from developments in both these classes of architecture.

MPPs are typically larger and have a lower latency interconnect than a Beowulf cluster. However, programmers on MPPs must take into account locality, load balancing, granularity, and communication overheads to obtain the best performance. Even on shared memory machines, many programmers develop programs that use message passing. Programs that do not require fine-grain computation and communication can usually be ported and run effectively on a Linux cluster.

Programming a NOW is usually an attempt to harvest unused cycles on an already-installed base of workstations in a lab or on a campus. Programming in this environment requires algorithms that are extremely tolerant of load balancing problems and large communication latency. Any program that runs on a NOW runs at least as well on a cluster.

A Beowulf cluster is distinguished from a NOW by several subtle but significant characteristics. These characteristics are shared by the SVA.

- Nodes in the cluster are dedicated to the cluster. This helps ease load balancing problems because the performance of individual nodes is not subject to external factors.
- Because the System Interconnect (SI) is isolated from the external network, the network load is determined only by the applications being run on the cluster. This eases problems associated with unpredictable latency in NOWs.
- All nodes in the cluster are within the administrative jurisdiction of the cluster. For example, the SI for the cluster is less visible to the outside world. Often, the only authentication needed between processors is for system integrity. On a NOW, network security is an issue.

Architectural Design

The SVA derives its most powerful attributes from its architectural design, which consists of a cluster of visualization nodes, high-speed interconnects, and advanced graphics cards.

SVA runs parallel visualization applications efficiently. The SVA also is an integral part of the HP Cluster Platform and storage (HP Scalable File Share) solutions. To accomplish this, the SVA architecture extends the HP Cluster Platform architecture with the addition of visualization nodes, which you can use as specialized compute nodes. Further, an SVA can be made up entirely of visualization nodes, or it can share an interconnect with compute nodes and a storage system. Thus, the SVA provides the HP Cluster Platform with a visualization component for those applications that require visualization in addition to computation.

The following sections describe the components that make up an HP Cluster Platform, followed by those tasks and components that are unique to an SVA.
Components of the HP Cluster Platform

Because the SVA is an extension of the HP Cluster Platform, you can begin by understanding its base components without any visualization nodes. The following are the key architectural components of an HP Cluster Platform system without visualization nodes:

**Compute Nodes** and **Administrative/Service Nodes**
The compute cluster consists of compute nodes and administrative or service nodes. Parallel applications are allocated exclusive use of the compute nodes on which they run. The other nodes provide administration, software installation, remote login, file I/O, external network access, and so on. These nodes are shared by multiple jobs, and are not allocated to individual jobs.

**System Interconnect (SI)**
A high-bandwidth, low-latency network which connects all nodes. This supports communication among the compute nodes (for example, MPI and sockets) and file I/O between compute nodes and a shared file system.

**Administrative Network**
An Administrative Network connects all nodes in the cluster. In an HP XC compute cluster, this consists of two branches, the Administrative Network and the Console Network. This private local Ethernet network runs TCP/IP. The Administrative Network is Gigabit Ethernet (GigE); the Console Network is 10/100 BaseT. (Because visualization nodes do not support console functions, visualization nodes are not connected to a console branch.)

**Linux**
The nodes of the cluster run a derivative of 64-bit Red Hat® Enterprise Linux Advanced Server.

---

**Note**
All nodes must attach to two networks using different ports, one for the SI and one for the Administrative Network.

---

**Main Visualization Cluster Tasks**
The SVA has a number of tasks that are unique to a visualization cluster. It accomplishes these tasks using a set of unique node types that differ in their hardware configurations, and so are capable of different functional tasks. The main tasks are as follows:

**Render images.**
A node must have a graphics card to render images. A visualization job uses multiple nodes to render image data in parallel. A render node typically communicates over the SI with other render and display nodes to composite and display images.

**Display images.**
The final output of a visualization application is a complete displayed image that is the result of the parallel rendering that takes place during an application job. To make this possible, a display node must contain a graphics card connected to a display device. The display can show images integrated with the application user interface, or full screen images. The output can be a complete display or one tile of an aggregate display.

**Remote images.**
The SVA also supports the transmission of a complete image to a system external to the cluster over an external network for remote viewing; for example, to an office workstation outside the lab. A node with a port connected to the external network is recommended. Alternatively, you can connect to the external network by routing through another cluster node with such a port.

**Integrate an application user interface.**
An application user interface (UI) usually runs on a cluster node. The UI typically controls the parts of the distributed application running on other nodes. A node that provides users with access to the UI can have an attached keyboard, mouse, and monitor for user interaction. Alternatively, the node can export the application UI to an external node using the X protocol or using the HP Remote Graphics Software.
Components of an SVA

The main tasks described in “Main Visualization Cluster Tasks” (pg. 18) are supported by two types of visualization nodes, which differ in their configuration and in the tasks they carry out. The two nodes types can carry out multiple tasks. These node types are unique to the SVA configuration and extend HP compute clusters to support visualization functions. See Chapter 3 for detailed information on the hardware configurations of these node types.

Display Nodes

Display nodes carry out the display task. Typically, a display node contains one or two graphics cards, each connected to its display device. The output of each graphics card on a display node is sent to a display device. Final output can be a single tile or a partial image in the form of a single tile, which is part of an aggregate multi-tile display.

The SVA supports up to eight display nodes in a Display Surface. The display nodes in your cluster can drive one or two display devices in the case of xw8200 nodes, and one to four display devices in the case of xw9300 nodes. See the SVA System Administration Guide for more information on setting up display nodes, displays, and Display Surfaces.

Render Nodes

Render nodes render images, as do display nodes. However, render nodes are not connected directly to display devices. Typically, render nodes are used by visualization applications that composite images. Render nodes render a part of the final image. These sub-images are combined with sub-images from other nodes. The composed image data is transferred to another render node, or to a display node to be routed to a display device.

Render nodes are industry standard workstations with standard OpenGL 3D graphics cards.

Both types of nodes can perform UI and remote graphics functions. When nodes are allocated to a job, the job typically requires specific display nodes that correspond to the display devices intended for use. Typically, there is no requirement for specific render nodes.

Configuration Flexibility

The SVA supports several different configurations and uses. These include:

- Multiple displays with different resolutions.
- Use of a variable number of display and render nodes to solve the computational and rendering requirements of an application.
- Bounded configuration designed for a single user.
- Larger, modular, expandable systems designed for one or more concurrent users.

See Chapter 3 for more information on the physical configurations of the SVA.

SVA Operation

This section describes a common way data flows through an SVA.

Cluster Data Flow

Figure 2-1 shows a high-level view of the basic components of an SVA.
A common usage scenario includes a master application node that runs the controlling logic of an application, processes the 3D data, and updates the virtual display or scene in the case of scenegraph applications. The master node typically does no rendering. Because it transmits data changes to other visualization nodes, it must be able to communicate with these nodes using the cluster SI. The SI is the fastest network available to the SVA and is the best choice for internode communication when performance is important.

A different scenario does not use a master application node. Instead, an application relies on Distributed Multi-Head X (DMX) to distribute the display output to multiple nodes and displays. It does this by controlling the back-end X Servers running on each of the display nodes. Partial images routed to individual display devices are assembled and displayed as a single virtual image.

Other scenarios arise depending on the application and the capabilities of visualization middleware running on the SVA. For example, several render nodes can carry out rendering and compositing tasks. The render nodes can rely on middleware software to handle the compositing of any partial images. The image data then flows to a display node before being sent to a display device or remote node, for example; a desktop display outside the SVA.

See Chapter 5 for other usage scenarios.

File Access

Visualization applications typically read data from files in response to user input. For example, after starting an application, you specify a data file to open and load. Without exiting the application, you can select additional data files to open and load, replacing or adding to the data already loaded. Much visualized data is static rather than time-varying. When visualizing time-varying data, the application must read and cache multiple time steps. The application may not be able to visualize the data as it is being read. Each time step may need to be analyzed and features extracted based on application settings. The application then caches the results of the analysis or rendering to display an animation of the time steps.

Although parallel visualization is a relatively new approach, some file access patterns that applications use include the following:

- Master portion of the application reads data from files and distributes data to visualization nodes using the SI.
- Visualization nodes all read data from the same files.
- Visualization nodes all read data from different files.
- Master writes data; for example, to save an animation sequence.

Dataset sizes can range from less than 1GB to more than 100GB. Some examples include seismic datasets that are 1GB to 128GB, and medical datasets that are 1GB to 50GB.

Applications access files using HP Scalable File Share (SFS) or NFS. When visualization nodes are integrated into a cluster with HP SFS, they access this file system using the SI. When HP SFS is in a separate cluster and not accessible by the SI, access is with GigE.

See the SVA System Administration Guide for more information.
3 SVA Hardware and Software

This chapter provides information on the hardware and software that make up the SVA. It is a useful reference for anyone involved in managing the SVA. It is also useful for anyone who wants to understand the hardware that makes up the SVA and the software that is installed on it.

The SVA combines commodity hardware components with software that include the following:

- A cluster of Intel EM64T or AMD Opteron HP workstations as visualization nodes.
- NVIDIA® Quadro® FX 3450 or NVIDIA Quadro FX 4500 graphics cards with optional G-sync or hardware SLI.
- InfiniBand, Gigabit Ethernet (GigE), or Myrinet system interconnects.
- Third-party software tools and libraries.
- Custom and enhanced software tools.

Hardware Component Summary

You can use the SVA with a variety of applications that run on distributed computing systems; in this case, a cluster of Linux workstations. The SVA is a specialized version of the HP Cluster Platform systems; in this case, based on HP ProLiant DL380 G4 and 385 G4 servers and xw8200 or xw9300 visualization nodes.

There are two SVA physical configurations:

**Bounded configuration**
Contains only visualization nodes and is limited in size to one to three racks. The bounded configuration serves as a standalone visualization cluster. It can be connected to a larger HP XC cluster via external GigE connections. This level of inter-cluster integration supports communication with a compute cluster and data retrieval from a file share such as an HP Scalable File Share (SFS).

**Modular, expandable system**
This configuration has two or more racks as needed to contain the maximum number of supported nodes. It is based on HP Cluster Platform building blocks. It can be exclusively visualization nodes or be combined with compute nodes as part of an integrated HP Cluster Platform system. When integrated into a larger Cluster Platform system, the visualization nodes can use a high speed system interconnect to load data from an HP SFS.

The two SVA physical configurations are built using one or more of three types of building blocks. Each building block uses a single rack.

- **Utility Visualization Block (UVB)** Base utility unit of a bounded physical configuration.
- **Utility Building Block (UBB)** Base utility unit of a modular expandable system.
- **Visualization Building Block (VBB)** Rack of visualization nodes that can be added to either base units. The VBB contains a maximum of eight nodes.

A bounded physical configuration has the following components as summarized in Chapter 2:

- Display nodes.
- Render nodes.
- Head node (found in HP Cluster Platform systems, and thus not unique to the SVA).
- System Interconnect and Administrative Network (found in HP Cluster Platform systems, and thus not unique to the SVA).

The head node is a typical node type found in HP Cluster Platform systems. SVA bounded configurations support either a workstation (xw8200 or xw9300) or server (DL 380 or DL 385) as the head node. Modular configurations support only a DL 380 or a DL 385 as the head node.

Figure 3-1 illustrates a sample bounded configuration. The UVB contains the network switches, PDU, five visualization nodes, and the head node. The visualization nodes support a 2x2 multi-tile display. Additional VBBs can be added to this configuration, with up to eight workstations in each rack.
Network Configurations

This section describes the different networks used in the SVA.

System Interconnect (SI)

The SI for visualization nodes can be GigE, InfiniBand, or Myrinet. When the visualization nodes are integrated with compute nodes, the choice of SI is usually determined by the requirements of the compute nodes.

Administrative Network Connections

A GigE interconnect serves as the Administrative Network to control the operation of the cluster (for example, boot, shutdown, restart) and to control the running parts of a distributed application (for example, launching and stopping processes).

The SVA adds visualization capability to an HP XC cluster; therefore, the Administrative Network is implemented with the logical configuration defined by the HP XC and Cluster Platform architectures. However, the physical implementation differs.

The management switches are collected together in one rack. SVA nodes connect to branch switches in the Administrative Network. SVA nodes do not connect to the console branch.

Nodes connect to the switches according to the Cluster Platform Administrative Network connections for HP XC. Display and render node types are typically grouped together.

Display Devices

This section describes the supported devices you can add to an SVA. Display devices are not necessarily provided as part of the SVA. For example, your site can use projector display systems or immersive displays provided by third party suppliers.

Displays fall into a number of categories, including immersive CAVE displays, single monitors, multiheaded monitors, large wall displays, multiheaded desktops, flat panels, and projector displays used in theaters. SVA hardware and software deliver images to digital or analog standard interfaces. The SVA depends on the graphics cards to drive the image output. This means whatever display devices the graphics cards support are available for use.

Theoretically, SVA technology can scale to arbitrarily large displays. Realistically, the bandwidth of the network delivering subparts of the image to various nodes, and the resolution of display devices are limiting factors. You can create large displays by arranging a grid of smaller displays.
See the SVA *System Administration Guide* for more information on setting up display nodes and devices.

**SVA Software Summary**

The SVA combines third party software tools and libraries with custom and enhanced software tools and libraries. SVA software must be installed and run on each visualization node as well as the head node of a valid cluster configuration, such as an HP Cluster Platform 3000 or HP Cluster Platform 4000, properly configured for HP XC System Software with the SVA option. This section describes the following software categories:

- Linux Operating System.
- HP XC clustering software.
- Additional system software.

Figure 3-2 illustrates the various software categories and their hierarchical interrelationships, as well as the tasks carried out by the utilities provided by HP Visualization System Software (VSS). These tasks — allocate, launch, initialize, cleanup — are aligned alongside the software categories they impact.

**Figure 3-2  Software Hierarchy in the SVA**

<table>
<thead>
<tr>
<th>Applications</th>
<th>Launch</th>
</tr>
</thead>
<tbody>
<tr>
<td>Visualization Libraries (optional)</td>
<td>Initialize</td>
</tr>
<tr>
<td>OpenGL</td>
<td>Cleanup</td>
</tr>
<tr>
<td>X Servers</td>
<td>Allocate</td>
</tr>
<tr>
<td>HP XC Linux</td>
<td></td>
</tr>
<tr>
<td>Cluster Nodes and Displays</td>
<td></td>
</tr>
</tbody>
</table>

**Linux Operating System**

The SVA software is layered on top of HP XC System Software Version 3.0, a clustering Linux distribution compatible with Red Hat Enterprise Linux Advanced Server V4.0 Update 2. The kernel version is V2.6.9–x. See “HP XC Clustering Software” (pg. 24).

The windowing system used by Red Hat is the X.Org windowing system.

Table 3-1 summarizes the main operating system components as well as the low-level drivers.
Table 3-1 Operating System and Driver Components

| Component                                           | Notes                                                                 |
|-----------------------------------------------------|                                                                      |
| **Base Operating System**                           |                                                                      |
| Red Hat Enterprise Linux Advanced Server V4.0 Update 2.   | HP XC Linux is compatible with this version of Red Hat Enterprise Linux; however, it is built by HP and does not contain all the components distributed by Red Hat. www.redhat.com |
| XC Web Site http://www.hp.com/techservers/clusters/xc_clusters.html |                                                                      |
| X.Org Windowing System. X.Org Foundation: www.x.org  | Official Windowing System of the X.Org Foundation that is also included as part of the standard Red Hat distribution. |
| **Low-Level Drivers**                               |                                                                      |
| Linux driver for the supported NVIDIA Quadro FX graphics cards. | Device driver for the graphics cards provided by NVIDIA Corporation. Qualified by HP WGBU group. |
| Driver for selected high-speed interconnect.         | InfiniBand, GigE, and Myrinet are supported.                        |

HP XC Clustering Software

The SVA runs HP XC System Software Version 3.0 clustering software. The SVA uses HP XC for the following key system management tasks:

- Installing and reinstalling a uniform set of software on visualization nodes.
- Booting and shutting down the cluster.
- Managing user accounts and user directories across the cluster.
- Naming each of the nodes in the cluster and determining which nodes are up and running.
- Serializing application use of the cluster.

For more information on HP XC, consult the HP XC documentation set at the following Web site:

Table 3-2 summarizes the software components that are provided by the HP XC operating system that relate to the SVA.

| Component                                           | Notes                                                                 |
|-----------------------------------------------------|                                                                      |
| Simple Linux Utility for Resource Management (SLURM) | A resource manager for Linux clusters. Used to set up visualization sessions and launch visualization jobs. Preferred allocation utility of HP XC. |
| Platform Load Sharing Facility for High Performance Computing (LSF HPC) | Layered on top of SLURM to provide high-level scheduling services for the HP XC system software user. LSF can be used in parallel with SVA job launching techniques that rely on SLURM. |
| SystemImager                                        | System file replication used to install cluster nodes.              |
| pdsh                                                | A parallel distributed shell that replaces rsh and ssh.             |
| HP-MPI                                              | Message Passing Interface.                                          |
| HP Scalable File Share (HP SFS)                     | High performance file system (optional).                            |
| FlexLM®                                             | License management.                                                |

Additional System Software

The SVA Software Kit provides software focused on making the job of developing and running visualization applications easier. This section summarizes additional system software of interest related to the SVA, as well as where to get information on packages or applications. The main categories of additional system software include:
• Main software components provided by HP (Table 3-3).
• Main software components provided by third parties (Table 3-4).
• Application development tools available on the SVA (Table 3-5).

### Table 3-3 HP SVA System Software

<table>
<thead>
<tr>
<th>Software</th>
<th>Description</th>
</tr>
</thead>
</table>
| Visualization System Software (VSS) | Collection of various categories, including:  
- Data Access Functions that permit an application to access and use the Configuration Data Files for job allocation and launch.  
- Command syntax for job launch scripts commands. |
| SVA Visualization System Software Reference Guide | |
| HP Remote Graphics Software | Transmits 2D and 3D images across standard computer networks to remote users. (optional) |

### Table 3-4 Third Party System Software

<table>
<thead>
<tr>
<th>Software</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OpenGL</td>
<td>Primary interface programmers use to create images.</td>
</tr>
<tr>
<td><a href="http://www.opengl.org/developers/about/overview.html">www.opengl.org/developers/about/overview.html</a></td>
<td></td>
</tr>
<tr>
<td>OpenGL Utility library (GLU)</td>
<td>Contains routines that build on the lower level OpenGL library to perform such tasks as setting up matrices for specific viewing orientations and projections, performing polygon tessellation, and rendering surfaces.</td>
</tr>
<tr>
<td>freeglut</td>
<td>This library masks a number of the operating system specific calls for creating windows and managing input devices.</td>
</tr>
<tr>
<td>OpenMotif</td>
<td>Royalty-free version of Motif®, the industry standard graphical user interface. It provides users with the industry’s most widely used environment for standardizing application presentation on a wide range of platforms.</td>
</tr>
<tr>
<td><a href="http://www.openmotif.org">http://www.openmotif.org</a></td>
<td></td>
</tr>
<tr>
<td>Distributed MultiHead X (DMX)</td>
<td>An open source application that provides a proxy X Server that distributes its display over multiple X Servers.</td>
</tr>
<tr>
<td>X Server (X.Org)</td>
<td>X Server available as part of the Red Hat distribution.</td>
</tr>
<tr>
<td><a href="http://x.org">http://x.org</a></td>
<td></td>
</tr>
<tr>
<td>Chromium</td>
<td>A system for interactive rendering on clusters of graphics workstations.</td>
</tr>
</tbody>
</table>

### Table 3-5 Application Development Tools

<table>
<thead>
<tr>
<th>Software</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default GNU C and C++ compilers and run-time</td>
<td>Included as part of the Red Hat distribution.</td>
</tr>
<tr>
<td>gdb</td>
<td>Included as part of the Red Hat distribution.</td>
</tr>
<tr>
<td>Perl, Tcl/tk, Python</td>
<td>Scripting tools available as part of the Red Hat distribution.</td>
</tr>
</tbody>
</table>

64 bit Linux applications are supported on SVA systems. 32 bit applications may run if their providers have validated them to run on 64 bit Linux systems. You may find that you need to install additional libraries. You may also find that certain libraries, for example, MPI, don’t work for particular hardware configurations.
4 Setting Up and Running a Visualization Session

This chapter explains how to run visualization applications on the SVA. A visualization session relies primarily on HP XC utilities to do the underlying work; however, you can avoid manually using the underlying utilities by means of job launch scripts and associated templates provided by the SVA kit.

For details on HP XC utilities, see the HP XC system documentation (link available from the SVA Documentation Library).

Configuration Data Files

This section provides an overview of the Configuration Data Files applicable to all users of the SVA.

Configuration Data Files provide specific information about the system configuration of an SVA. File details are mainly of interest to the system administrator who manages and configures the cluster. All visualization sessions that you initiate to run your application depend on input from the Configuration Data Files. For details on the content and syntax of the Configuration Data Files and how to modify them, see the SVA System Administration Guide.

The Configuration Data Files contain important information on a visualization job session. For example, they can specify the default applications available (DMX or HP RGS), host names, node roles (render, display, compute), and Display Surface names and associated nodes. A Display Surface is a named assemblage of display nodes and their associated display devices, including the physical orientation of the display devices relative to one another.

There are three Configuration Data Files, which can interact to provide input when you start a visualization session:

- **Site Configuration File.**
  This file contains the default system settings and Display Surface definitions. It is generated initially by HP (and a site administrator if necessary) using the svaconfigure utility when the cluster software is installed. Only root users can change this file. This file is named /opt/sva/etc/sva.conf.

- **User Configuration File.**
  Using this file, any user can override some of the job preferences specified in the Site Configuration File. This file is named ...~/sva_<cluster_name>.conf.

- **Job Settings File.**
  Job data is defined at job allocation time from options specified to the job launch scripts and from data access calls embedded in the script. The Job Settings File is named /hptc_cluster/sva/job/<id>.conf. This file has a life span equal to that of the job.

Because the three files overlap in several (but not all) of their settings, a hierarchy exists for the use of values specified from the command line and in more than one file. The prioritized order of use of tag values during a job is as follows:

1. Command line options.
2. Job Settings File.
4. Site Configuration File.

Running an Application Using Scripts

Typically, you encapsulate the various commands needed to run applications using a script file. This speeds the process of running the application, given the likelihood that this is a task you repeat.

The installation of the SVA Software Kit provides several general purpose script templates. These templates are the starting points for creating scripts to launch your own application. They carry out the tasks of allocating cluster resources, launching necessary ancillary applications (for example, X servers and DMX), running the application on the right nodes, and terminating the application at the end of the session. Script templates typically need site-specific editing.
The kit installation also provides fully functional job launch scripts that you can use as is or customize for your own site. These are typically located in the `/opt/sva/bin` directory and are configured to be on your PATH.

Follow these steps to use a script template:

1. Select a template.
2. Modify a copy of the script template to suit the specific needs of the visualization application.
3. Execute the modified script from the head node to launch your application as part of a visualization session.

Manpages for each template and script are installed on all SVA nodes. This information is also available in the *SVA Visualization System Software Reference Guide*.

### Selecting a Template

Select one of the available templates or fully functional scripts that most closely suits the needs of your application and environment. The following templates and scripts are available:

- **sva_job_template.sh**
  A generic template you can use as the basis for scripts to launch visualization applications. The `sva_job_template.sh` template is located in the `/opt/sva/samples` directory.

- **sva_paraview.sh**
  This script template is specifically optimized to launch the ParaView application. It can serve as a template for your use of ParaView. However, it needs site-specific editing before use. The `sva_paraview.sh` template is located in the `/opt/sva/samples` directory.

- **sva_chromium_dmx.sh**
  This script assumes that your application works on a single workstation. The script automates getting the application to run on the SVA using a multi-tile display. It also uses DMX to distribute an image across multiple nodes. It uses Chromium in the case of standard OpenGL applications. This is a fully functional script on your PATH. The `sva_chromium_dmx.sh` script is located in the `/opt/sva/bin` directory.

  **Tip**
  A useful feature of the `sva_chromium_dmx.sh` script is its interactive mode for running all sorts of applications, including regular X Server applications. For example, you can display high resolution images with a variety of applications, or you can run standard OpenGL applications with Chromium. The script provides an easy way to take advantage of a multi-tile display. See “Running an Interactive Session” (pg. 29) for more information on running an interactive session.

- **sva_remote.sh**
  This script sets up the SVA to let you access a visualization node in the cluster from a remote desktop using HP RGS. This is a fully functional script on your PATH. The `sva_remote.sh` script is located in the `/opt/sva/bin` directory.

### Modifying a Script Template

After selecting a script template, you must edit it to suit your specific application and system environment. It’s a good idea to copy the selected template to your working directory on the head node before making changes. The templates are commented to describe what each template does and what areas of a template you must edit. These comments are the place to begin when you are ready to select and edit a script template.

Templates have several major functional areas that execute successively using Data Access Functions and associated environment variables. These functions and variables are defined in `/opt/sva/bin/sva_init.sh` and draw their default values from the Configuration Data Files. The *SVA Visualization System Software Reference Guide* details the functions and their associated environment variables.

After the script initializes the variables, the main functional areas of the script execute as follows:

1. The ParaView application is not shipped as part of the SVA Kit.
1. **Allocate**: Allocates cluster resources for the visualization job. The allocation phase launches an HP XC SLURM job using the `srun` command. A SLURM Job ID is assigned to the job, which starts a session with the appropriate cluster resources; for example, the Display Surface and the requested number of display and render nodes. The number of resources can be specified using command line options in the script.

2. **Launch**: Starts the visualization session and necessary services on the render and display nodes; for example, X Servers and DMX.

3. **Run**: Starts the master application component on the display node and the worker application components (if any) on the render nodes and display nodes. The run phase also starts any Chromium components, if appropriate to the job. Where the master runs is based on the Execution Host value in the Configuration Data Files. This varies based on the Display Surface you choose for the job. See the SVA System Administration Guide for detailed information on the Configuration Data Files and Display Surfaces.

4. **Terminate**: Stops the visualization session cleanly. Also stops the services that were started by the script; for example, X Servers and DMX.

---

### Using a Script to Launch an Application

After modifying the template (if necessary) to create a launch script that matches the application requirements, start the script from the head node. Each template has default options that you can respecify. These options are documented in the SVA Visualization System Software Reference Guide, and in manpages for each template or fully functional script.

The following command runs the `atlantis` application on the `FULL_DISPLAY` Display Surface using the Chromium/DMX launch script. The application-specific command-line parameter `-count 20` is used.

```
% sva_chromium_dmx.sh -d FULL_DISPLAY \
"/usr/X11R6/lib/xscreensaver/atlantis -count 20"
```

This script command uses the `FULL_DISPLAY` Display Surface (a site-specific multi-tile Display Surface), allocates the resources, and starts the X Servers. You do need to enter the name of an existing Display Surface on your cluster as a command option.

A Display Surface is a named assemblage of one or more display devices with a particular orientation and their associated display nodes. Initial configuration of the SVA creates a named single tile Display Surface for each display node and its associated display device. Your site administrator may have created additional Display Surfaces using the Display Surface Configuration Tool. The site administrator also can use the Display Node Configuration Tool to specify which (if any) display nodes output more than one tile. This is the way the `FULL_DISPLAY` Display Surface in this example was created. You can use the Display Surface Configuration Tool to list all the named Display Surfaces on your cluster. Both tools are documented in the SVA System Administration Guide.

---

### Running an Interactive Session

Use the Chromium/DMX launch script to interactively control the launching of your applications.

You must set the `DISPLAY` environment variable correctly before you launch the script, or it fails. You set the `DISPLAY` environment variable to point to the node’s X Display where the DMX Console Window appears. (Refer to the DMX documentation for details of its user interface, including the DMX Console Window.) For example:

```
% export DISPLAY node:0.0
```

Use the following command to launch an interactive session:

```
% sva_chromium_dmx.sh -I -d FULL_DISPLAY
```

This script command uses the `FULL_DISPLAY` Display Surface (a site-specific multi-tile Display Surface), allocates the resources, and starts the X Servers. It also starts a desktop environment (for example, KDE or Gnome) from which you can launch applications repeatedly while retaining the same job resources. To launch an application, open a terminal window and then run the application as usual.

In the specific case of OpenGL applications, you use the Chromium/DMX script again from the terminal window to run it. For example:

```
% sva_chromium_dmx.sh "/usr/X11R6/lib/xscreensaver/atlantis -count 20"
```
You can start, stop, and restart an application to make it easier to test and debug. You must be able to view the SVA Display Surface because the DMX Console provides limited visual feedback.

Creating an interactive session in this way lets you take advantage of your multi-tile display for other applications. Your desktop environment is available to start any application and display it on the multi-tile display; for example, to display high-resolution images or to launch an application like ParaView.

**Tip**

For convenience, you can create desktop icons as shortcuts to application launch commands.
5 Application Examples

This chapter describes the steps to start several representative applications that vary in their structure and requirements:

- A workstation application that is launched remotely to use only a single node in the SVA. See “Running an Existing Application on a Single SVA Workstation” (pg. 31).

- An application that uses render and display capabilities of the SVA (for example, ParaView). See “Running Render and Display Applications Using ParaView” (pg. 35).

- A workstation application that uses Chromium software and DMX to display on multiple tiles using the SVA. See “Running a Workstation Application Using a Multi-Tile Display” (pg. 38).

Table 5-1 summarizes the differences among three application scenarios, detailed in the following sections.

<table>
<thead>
<tr>
<th>Table 5-1 Comparison Summary of Application Scenarios</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Scenario</strong></td>
</tr>
<tr>
<td>----------------------------------------------------</td>
</tr>
<tr>
<td>Remote access using HP RGS</td>
</tr>
<tr>
<td>Resolution scaling/multi-tile</td>
</tr>
<tr>
<td>Multi-display, cluster application</td>
</tr>
<tr>
<td>Data scaling and compositing</td>
</tr>
</tbody>
</table>

Running an Existing Application on a Single SVA Workstation

This section describes the main steps and considerations to get an application that already runs on a single workstation to run on a single node within an SVA. Control takes place using a workstation remote to the cluster.

Assumptions and Goal

This example assumes you have a visualization application that currently runs on a single workstation. It also assumes that you have not specifically modified it to take advantage of the parallel features of a cluster.

The goal of this example is to make the application run on the SVA while maintaining control remotely from a desktop that is outside the cluster. This desktop is remote relative to the SVA although you may consider it your local workstation. In this chapter, your local workstation is meant to designate a machine that is remote to the SVA.

Working in this way lets you take advantage of the more powerful features of the cluster. These include more powerful graphics cards, or specific software libraries such as OpenGL extensions. It is also helpful and convenient for testing and debugging your application. It also facilitates collaborative work.

In addition to having your cluster set up with the HP XC and SVA software, you also need to have HP RGS installed and configured on those nodes within the cluster that you intend to access remotely. You also must have the RGS client software (the RGS Receiver) installed and configured on your local desktop where you intend to route the output from your application.
The SVA Software Installation Guide has specific RGS installation instructions that you must use to supplement the HP RGS installation instructions.

**HP Remote Graphics Software and Use**

HP RGS is an advanced utility that makes it possible to remotely access and share 3D graphics workstation desktops. This can be done across Windows and Linux platforms. With RGS, you can:

- Remotely access 3D graphics workstations.
- Access applications running on SVA from a Linux or Windows desktop.
- Perform multiuser remote collaborations.

A link to the HP RGS documentation is available from the SVA Documentation Library.

**Location for Application Execution and Control**

This example requires that you configure the SVA so that it can run your application while you control it from your local desktop. Additionally, display output is routed to your local desktop using HP RGS.

See “Launch Script” (pg. 33) for examples of using the installed RGS script. A summary of the overall process follows.

1. An SVA Kit installed RGS launch script allocates resources on the SVA based on the Display Surface you specify. In the case of the RGS script, you must use a Display Surface with a single display node that has the RGS Sender installed. Your application runs on the same display node denoted by the Display Surface.

2. The RGS Receiver starts on your local desktop (Linux or Windows). Configure it by manually entering the external name for the display node defined by the Display Surface you chose.

3. The RGS Receiver and Sender connect.

4. A desktop environment (for example, KDE or Gnome) appears on your local desktop in the same way it would appear if you were directly logged into an individual SVA node.

5. You control your application, that is, provide input to the application while it is running, using the local desktop keyboard and mouse. Display output from the application appears on your local desktop. Display output simultaneously appears on the display device in the SVA as determined by the Display Surface you chose when you started the launch script.

Figure 5-1 shows the relationships among the various processes that run when you launch visualization jobs.

There are four processes that must run when a remote visualization session begins.

- The X Server.
- RGS Sender on the SVA display node.
- RGS Receiver on your local desktop.
- Your visualization application.

**Figure 5-1** Using a Single SVA Node from Local Desktop
Data Access

If you use a single SVA display node, place the data files in a convenient location given your site configuration. One location that provides reasonably fast access to the data is on a local disk of the display node, which is the node running your application. Given that the application in this scenario runs on a single node, there is little to be gained by distributing the data.

If you choose to store data locally, you can copy the data file to the display node after the application starts. This ensures that you access a node allocated to your job. You can use the `/tmp` directory to store the data on the local disk.

Tip

Consider running the launch script interactively if you plan to use local disk access to the data. When run in interactive mode, the script allocates cluster resources first. You can then copy the data file to the allocated display node before actually launching your visualization application.

Alternatively, NFS and HP Scalable File Share (SFS) can provide access to the data. Because HP SFS provides high bandwidth access to data over the SI of SVA, it is the best choice if performance is a high priority.

See the SVA System Administration Guide for guidelines and alternatives for accessing data files when running visualization applications on the SVA.

Use of Display Surfaces

The SVA provides the infrastructure and utilities to simplify the task of allocating display devices. The primary mechanism that you use to set up displays is the Display Surface. A Display Surface is composed of one or more display nodes and their associated display devices; for example, a simple Display Surface is a specific display node and an attached flat panel display device. Initial configuration of the SVA sets up a series of default named Display Surfaces, one for each display node and its directly cabled display device. Any of these default Display Surfaces should work for this example, assuming the display node has the RGS Sender software installed, an external NIC, and uses a single graphics card to output one or two tiles.

Because the RGS Sender routes the display output to your local desktop, its display device is the one you typically use. Display output can appear simultaneously on the display device of the SVA if you specify a Display Surface when you start the launch script. Alternatively, if you choose not to specify a Display Surface and accept a default node that is RGS-enabled, your display output may only appear remotely. This takes place when the assigned node is a render node rather than a display node. Refer to the SVA Visualization System Software Reference Guide for detailed syntax of script options.

Tip

You can also share your RGS Receiver on your desktop. In this way, other users can see the output of your application simultaneously at their desktops. See the RGS documentation for more information on how to do this.

See the SVA System Administration Guide for information on setting up displays, display nodes, Display Surfaces, and how to create new ones.

Launch Script

The SVA Software Kit installs a fully functional script that you can use to launch your visualization application using HP RGS. The script derives some of its key input parameters from the Configuration Data Files. You can override some of these default values by creating a User Configuration File or by direct input on the command line.

You typically specify three pieces of information when using the script:

- A Display Surface with a single display node configured as follows:
  - With a single graphics card supporting one or two tiles. This results in a large window on the local display.
  - With two graphics cards, only one of which can be used for the remote session.

This display node must have the RGS Sender software installed and have a NIC to access the external network. You can specify the Display Surface as an option when you use the RGS launch script.
Alternatively, you can omit the Display Surface option (–d) and accept a render or display node allocated automatically by the script. The allocated node will be one that supports RGS functions. The Site Configuration File (/opt/sva/etc/sva.conf) specifies all the available Display Surfaces. You can also use the Display Surface Configuration Tool to list the Display Surfaces. See the SVA System Administration Guide for more information.

- The application name with or without application parameters.
- The external Ethernet name of the display node specified by the same Display Surface. You must specify the external name when you start the RGS Receiver on your local desktop. The RGS launch script provides the Ethernet name immediately after you start it.

Non-Interactive Example

The following steps launch the RGS script in non-interactive mode (a batch job) to run the atlantis application:

1. Log in to the SVA from your local desktop using a terminal window.
2. Enter the following command from the terminal window:
   
   ```bash
   % sva_remote.sh -d SVA_DS_1_2 "/usr/X11R6/lib/xscreensaver/atlantis -count 20"
   ```

   This command specifies that the SVA_DS_1_2 Display Surface be used. If you omit the –d option, the script automatically allocates a visualization node capable of using RGS as the remote node. The script draws the node from the pool of render nodes. If there are no render nodes available, then the script chooses a node from the pool of available display nodes.

   The window immediately displays the external name of the display node running the atlantis application. You need this name for the next step.

3. Start the RGS Receiver on your local desktop. In the RGS window that appears, enter the external name of the display node in the Connect to Sender field. Click Go.

   The RGS login window appears.

4. Enter in the RGS login window your Linux user name and password assigned for the SVA cluster.
   The desktop environment login window for the cluster appears on your local desktop.

5. Log in to the desktop environment window using your Linux user name and password.
   The desktop environment appears on your local desktop in the RGS Receiver window. The atlantis application display begins running.

6. Exit the application to terminate the visualization job.

   Provide input to the application while it is running, using the local desktop keyboard and mouse. Display output from the application appears on your local desktop and on the display device in the SVA. The SVA_DS_1_2 Display Surface in this example has a single display device, and its display node has a NIC that connects to the external network.

   For details on the syntax of the RGS script and its options, refer to the SVA Visualization System Software Reference Guide or the sva_remote.sh manpage.

Interactive Mode Example

The following steps launch the atlantis application in interactive mode:

1. Log in to the SVA from your local desktop using a terminal window.
2. Enter the following command from the terminal window:
   
   ```bash
   % sva_remote.sh -I
   ```

   The script allocates a visualization node (render or display) as the remote node. The script draws the node from the pool of render nodes. If there are no render nodes available, then the script chooses a node from the pool of available display nodes. The window immediately displays the external name of the display node that runs the application. You need this name for the next step.

3. Start the RGS Receiver on your local desktop. In the RGS window that appears, enter the external name of the display node in the Connect to Sender field. Click Go.

   The RGS login window appears.
4. Enter your Linux user name and password for the cluster in the RGS login window.  
The desktop environment login window for the cluster appears on your local desktop.

5. Log in to the desktop environment window using your Linux user name and password.  
The desktop environment appears on your local desktop in the RGS Receiver window.

6. Open a terminal window in the desktop environment and enter the following command:  
   % sva_remote.sh "/usr/X11R6/lib/xscreensaver/atlantis -count 20"  
   The atlantis application display begins.

7. Exit the application to stop the application only. You can then restart the application using the same  
   application command or another command, including a command with a different application. Cluster  
   resources remain allocated.  
   To deallocate the cluster resources and stop the RGS process on the cluster, exit the desktop environment  
   completely.

Provide input to the application while it is running using the local desktop keyboard and mouse. Display  
output from the application appears on your local desktop and simultaneously appears on the display device  
in the SVA.

Running Render and Display Applications Using ParaView
This section describes how to run a parallel visualization application on the SVA using both render and  
display nodes, using ParaView as a representative example.

Assumptions and Goal
This example assumes you have a rendering application such as ParaView to analyze, display, and enhance  
an existing data file for analysis. An application such as ParaView can run on a single workstation; however,  
it can also take advantage of the more powerful parallel features of the SVA to display data on a multi-tile  
display, and improve performance by distributing the rendering and compositing among the cluster nodes.

This example also assumes that you want to run the rendering application on the SVA while maintaining  
control remotely from a desktop that is outside the cluster.

You must have the cluster set up with the HP XC and the SVA software. You must also have your rendering  
application (in this example, ParaView) installed and properly configured on those nodes within the cluster  
that you will use for rendering and display. For ParaView, you must build the MPI version of ParaView to  
take advantage of the parallel features of the SVA before you install it on the SVA nodes. Note that ParaView  
is not provided as part of the SVA Kit.

You also must have the X Server on your local desktop configured to accept ParaView display output.

ParaView Overview
ParaView is an open source, multiplatform, extensible application designed for visualizing large datasets.  
This scalable application runs on single-processor workstations as well as on large parallel supercomputers.  
ParaView features include:

- Runs parallel on distributed and shared memory systems using MPI. These include workstation clusters,  
  visualization systems, large servers, supercomputers, and so on.
- The user interface can run either on the root MPI node or on a separate workstation using client/server  
  mode.
- ParaView uses the data parallel model, in which the data is broken into pieces to be processed by  
  different processes. Most of the visualization algorithms function without any change when running in  
  parallel.
- Supports both distributed rendering (where the results are rendered on each node and composited later  
  using the depth buffer), local rendering (where the resulting polygons are collected on one node and  
  rendered locally) and a combination of both (for example, level-of-detail models can be rendered locally  
  whereas the full model is rendered in a distributed manner). This provides scalable rendering for large  
  datasets without sacrificing performance when working with smaller datasets.
ParaView supports tiled displays through a built-in display manager.

- Handles structured (uniform rectilinear, non-uniform rectilinear, and curvilinear grids), unstructured, polygonal, and image data.
- All processing operations (filters) produce datasets. This enables you to either further process or save as a data file the result of every operation.
- Contours and isosurfaces can be extracted from all data types using scalars or vector components. The results can be colored by any other variable, or processed further.
- Vector fields can be inspected by applying glyphs (arrows, cones, lines, spheres, and various 2D glyphs) to the points in a dataset.
- Streamlines can be generated using constant step or adaptive integrators.
- Supports a variety of file formats including VTK, EnSight 6 and EnSight Gold, Plot3D, polygonal file formats including STL and BYU, and many other file formats.

As noted in the previous list, ParaView supports a variety of configurations and work models. However, the example scenario described in this section uses the MPI version of ParaView with a Render Client and a group of Render Servers.

A link to the ParaView documentation is available from the SVA Documentation Library.

**Location for Application Execution and Control**

This example requires that you configure the SVA so that it can run your application while you control it from your local desktop. Additionally, display output is routed to your desktop.

When run in parallel on a cluster, ParaView has two distinct functional areas:

- **ParaView Render Servers.**
  The Render Servers handle the rendering, compositing, and display functions. On the SVA, these Render Servers run on the render and display nodes. The pieces of data being rendered by the Render Servers change dynamically, which is handled by ParaView. The SVA render and display nodes carry out the same functions for ParaView, except that the display nodes are capable of sending the display output to a display device or a local desktop.

- **ParaView Client.**
  The ParaView Client handles the command and control functions for your display. It has a window interface with menus and toolbars and a simplified version of the model that appears on the display device. In the case of SVA, the ParaView Client typically runs on the Execution Host while its display is pushed back to your local desktop. HP recommends that you use a Display Surface that uses a display node as its Execution Host.

The Execution Host is defined for each Display Surface. You specify the Display Surface by name when launching a visualization job. The Execution Host for a Display Surface is the default location for running an application. You can locate the default Execution Host by reading the value for the `SVA_EXECUTION_HOST` tag in the Site Configuration File, `/opt/sva/etc/sva.conf`. Each instance of a named Display Surface in the Site Configuration File has an associated default Execution Host. You can override the default by setting the `SVA_EXECUTION_HOST` tag in your User Configuration File to indicate which host to use to run the application for a given Display Surface. See Chapter 4 and the SVA System Administration Guide for details on changing Configuration Data Files and their tag content.

**Figure 5-2** shows the flow of control for the ParaView application when run on the SVA.
Follow these steps to run ParaView on the SVA. You can use a script to carry out these steps.

1. Allocate the render and display nodes on the SVA that act as the ParaView Render Servers. Specify a named Display Surface to allocate the display device you intend to use.

   **Tip**
   You can use a SLURM `srun` command to do this.

2. Launch X Servers on all the allocated nodes.

3. Launch the ParaView Client on the Execution Host. When launching the Client, set the `DISPLAY` environmental variable to your local desktop in order to push the Client display to that machine. You also need to set the ParaView launch command option to `listen` mode. See the ParaView documentation for details on syntax.

4. Use a command to launch the ParaView Render Servers on the allocated nodes. This launch command must also specify the node location of the ParaView Client (the Execution Host). Specify the name of the Execution Host using its `ic-name`, which forces communication between the ParaView Render
Servers and Client to use the SI. This improves performance. (The ic-name is the HP XC convention used to denote that the SI communication mode is to be used.)

5. To terminate ParaView, select the File: Exit menu item from the ParaView Console window on your desktop. Kill the various X Servers on the allocated cluster nodes. You can use the SLURM scancel command.

Once you complete these steps, ParaView runs on the cluster while you maintain control of the application from your local desktop. You have a simple version of the image on the ParaView Console window that you can use to manipulate the image. These changes are displayed simultaneously on the Display Surface you specified when you allocated the cluster resources.

Data Access

In the specific case of using ParaView, you are likely to want to place the data files where it is convenient given your site configuration. Because ParaView controls the distribution of the data among the Render Servers, you typically want to make sure that the data is available on all the nodes allocated as Render Server nodes to allow data to load in parallel. One good location for the data is on the local disks of the Render Server nodes. If you choose to store your data locally, you can copy the data files to the /tmp directories of all the Render Server nodes.

If you choose to store data locally, you can copy the data file to the display node after the application starts. This ensures that you access a node allocated to your job. You can also run the launch script interactively if you plan to use local disk access to the data. When run in interactive mode, the script allocates cluster resources first. You can then copy the data file to the allocated display node before launching your visualization application.

Alternatively, NFS and the HP Scalable File Share (SFS) can provide access to the data. Because HP SFS can provide high-bandwidth access to data over the SI of the SVA, it is recommended if performance is a high priority.

See the SVA System Administration Guide for general guidelines and alternatives for accessing data files when running visualization applications on the SVA.

Use of Display Surfaces

The SVA provides the infrastructure and utilities to simplify allocating display devices. The primary mechanism that you use to set up displays is the Display Surface. A Display Surface is composed of one or more display nodes and their associated display devices. For example, a simple Display Surface is a specific display node and an attached flat panel display device. Initial configuration of the SVA sets up a series of default named Display Surfaces, one for each display node and its directly cabled display device. Any of these default Display Surfaces work for this example.

Your site administrator must define multi-tile Display Surfaces using the Display Surface Configuration Tool. The Display Surface Configuration Tool also can list all the named Display Surfaces for the cluster. Entering specific Display Surfaces to the script to access the display resources of the cluster.

Because this example routes the display output to your local desktop, its display device is the one you use to manipulate any image. Display output simultaneously appears on the display device in the SVA as determined by the Display Surface you chose when you started the launch script.

See the SVA System Administration Guide for details on setting up Display Surfaces, display nodes, and display devices.

Launch Script Template

The SVA Software Kit installs a script template that you can use as a guide to create your own site-specific script to run ParaView. It is called /opt/sva/samples/sva_paraview.sh. Follow the procedure described in “Location for Application Execution and Control” (pg. 36). Chapter 4 and the SVA Visualization System Software Reference Guide describe how to use launch templates to run applications, including the underlying functions and commands contained in the script.

Running a Workstation Application Using a Multi-Tile Display

This section describes how to run a serial workstation application on the SVA using Chromium and DMX.
Assumptions and Goal

This example assumes you have a visualization application that currently runs on a single workstation. It also assumes that you have not specifically modified it to take advantage of the parallel features of a cluster. This example also assumes that your goal is to run the application on the SVA and to take advantage of the multi-tile capabilities of the cluster.

Chromium Overview and Usage Notes

Chromium creates a way for many programs using the OpenGL standard to take advantage of cluster technology by automatically distributing OpenGL. Chromium provides a common parallel graphics programming interface to support clusters such as the SVA. In addition, it enables many existing applications to display on multiple tiles without modification.

Chromium provides the following features:

- A method for synchronizing parallel graphics commands.
- Streaming graphics pipeline based on the industry standard OpenGL API.
- Support for multiple physical display devices clustered together, such as powerwall displays.
- Support for aggregation of the output of multiple graphics cards to drive a single display at higher levels of performance and capability.

Chromium is automatically installed and configured on the SVA in several ways of interest to application developers:

- Autostart is not used.
- CR-Servers and CR Mothership are launched by the SVA launch script. See “Launch Script” (pg. 42).
- Tile information is taken from the SVA Configuration Data Files, which eliminates the need to hard code this information in the Chromium configuration files.
- Chromium uses tilesort and TCP/IP over the SI for DMX and Chromium connections.
- There is a ten second delay between the time that the Mothership and Clients launch. This adds a brief delay to the startup time.

Although Chromium has several configuration files that you typically need to edit, the SVA launch script eliminates this need by using configuration data from the SVA Configuration Data Files.

A link to the Chromium documentation is available from the SVA Documentation Library.

Distributed Multi-Head X (DMX)

Xdmx is a proxy X Server that provides multi-head support for multiple displays attached to different machines (each of which is running a typical X Server). A simple application of Xdmx provides multi-head support using two desktop machines, each of which has a single display device attached to it. A complex application of Xdmx unifies a four by four grid of 1280x1024 displays, each attached to one of 16 computers, into a unified 5120x4096 display.

The front end proxy X Server removes the limit on the number of physical devices that can coexist in a single machine (for example, due to the number of PCI-Express slots available for graphics cards). Thus, large tiled displays are possible.

A link to the DMX documentation is available from the SVA Documentation Library.

Location for Application Execution and Control

Although an application can run on any node in the SVA, HP recommends that you run it on one of the display nodes. The SVA is configured to use the default Execution Host for the Display Surface you choose when launching the visualization job. The Execution Host for a Display Surface is the default location for running an application. You can locate the default Execution Host by reading the value for the SVA_EXECUTION_HOST tag in the Site Configuration File, /opt/sva/etc/sva.conf.

Each instance of a named Display Surface in the Site Configuration File has an associated default Execution Host. You can override the default by setting the SVA_EXECUTION_HOST tag in your User Configuration...
File to indicate which host to use to run the application for a given Display Surface. See Chapter 4 and the SVA System Administration Guide for details on changing Configuration Data Files and their tag content.

The Chromium Mothership and DMX also run on the Execution Host node. See the Chromium documentation for details on the Mothership.

You must also provide input to the application as it runs. This means you must be able to provide keyboard and mouse input to the application. One common way to do this is to use the keyboard and mouse on a node external to the visualization job, such as the head node or a desktop external to the cluster, to control the multi-tile display directly connected to the cluster. You can use DMX to push your input from any external machine that has access to the cluster to the node acting as the Execution Host. For example, you can sit at a remote workstation running the DMX Console window. Set the DISPLAY environment variable before launching the script to point to where the DMX Console window will appear; for example, to the X Server in your office. As with all X Server remote use, your desktop must be configured to accept remote displays.

Another simple way to do this is to use the same display node that is the Execution Host as your console by using that display node's mouse and keyboard. If you don’t have a keyboard and mouse directly connected to the display node, you can use a KVM to provide input to the display node from the node you are using as your console.

Figure 5-3 shows the relationships among the processes that run when you launch a visualization job.

There are four processes that must run when a visualization session begins:

- The X Servers.
- Xdmx.
- Chromium.
- The visualization application.

Xdmx is a process that begins when you submit a visualization session. It must be launched before the application. Xdmx is the single frontend proxy X Server that acts as a proxy to a set of backend X servers; thus there is a single instance of Xdmx running on one of the display nodes.
Data Access

For a serial application that uses Chromium, place the data files in a convenient location for your site configuration. One location that provides fast access to data is on a local disk on the Execution Host (the node running your application). Given that the application in this example runs on a single node, there is little to be gained by distributing the data. You can choose the /tmp directory to store data on the local disk.

If you choose to store data locally, you can copy the data file to the display node after the application starts. This ensures that you access a node allocated to your job.

Tip
Consider running the launch script interactively if you plan to use local disk access to the data. When run in interactive mode, the script allocates cluster resources first. You can then copy the data file to the allocated display node before launching the visualization application. See "Location for Application Execution and Control" (pg. 39) to determine the Execution Host node for a given Display Surface.

Alternatively, NFS and HP SFS can provide access to the data. Because HP SFS provides high bandwidth access to data over the SI of SVA, use it if performance is a high priority.

See the SVA System Administration Guide for guidelines and alternatives for accessing data files when running visualization applications on the SVA.

Using Display Surfaces

The SVA can display the application output on a multi-tile display. It provides the infrastructure and utilities to simplify this task.
The primary mechanism that you use to set up displays is the Display Surface. A Display Surface is composed of one or more display nodes and their associated display devices; for example, a simple Display Surface is a specific display node and an attached flat panel display device. Initial configuration of the SVA sets up a series of default named Display Surfaces, one for each display node and its directly cabled display device. Your site administrator needs to define multi-tile Display Surfaces using the Display Surface Configuration Tool. This tool can also list all the named Display Surfaces for the cluster. A named Display Surface is a key input to the launch script. Entering specific Display Surfaces to the script is the way you access the display resources of the cluster.

Refer to the SVA System Administration Guide for details on setting up Display Surfaces, display nodes, and display devices.

Launch Script

The SVA Software Kit installs a script that you can use to launch standard X and OpenGL visualization applications. The script derives key input parameters from the Configuration Data Files. You can override some of these default values by creating a User Configuration File or by direct input on the command line. The key pieces of data you need to provide when you start the launch script are the following:

- The node’s X Display where the DMX Console Window appears. It is taken from the operating system DISPLAY environment variable. You must set this correctly before you launch the script or it fails. For example:
  \%
  \texttt{export DISPLAY node:0.0}

  You can set the display to a local desktop that has access to the SVA. Once the application is running, you can provide input using the DMX Console window on the local desktop. Using the DMX Console window typically requires that you can view the main SVA Display Surface. See the DMX documentation for a description of the DMX Console window. A link is available in the SVA Documentation Library.

- The invocation command for your application.

- The name of the Display Surface on which to display the application output.

You begin by logging in to the SVA using a terminal window. The following command runs the \texttt{atlantis} application on the \texttt{FULL_DISPLAY} Display Surface using the DMX-Chromium launch script. Note the use of the application-specific command line parameter: \texttt{-count 20}. Launch scripts must start from the head node.

\%
\texttt{sva_chromium_dmx.sh -d FULL_DISPLAY \\
"/usr/X11R6/lib/xscreensaver/atlantis -count 20"}

For details on the syntax of the Chromium script, see the SVA Visualization System Software Reference Guide or the \texttt{sva_chromium_dmx.sh} manpage.

You can also run applications interactively using this script. See “Running an Interactive Session” (pg. 29).
### Glossary

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Administrative Network</td>
<td>Connects all nodes in the cluster. In an HP XC compute cluster, this consists of two branches: the Administrative Network and the Console Network. This private local Ethernet network runs TCP/IP. The Administrative Network is Gigabit Ethernet (GigE); the Console Network is 10/100 BaseT. Because the visualization nodes do not support console functions, visualization nodes are not connected to a console branch.</td>
</tr>
<tr>
<td>bounded configuration</td>
<td>An SVA configuration that contains only visualization nodes and is limited in size to one to three racks. The bounded configuration serves as a standalone visualization cluster. It can be connected to a larger HP XC cluster via external GigE connections.</td>
</tr>
<tr>
<td>Chromium</td>
<td>Chromium is an open source system for interactive rendering on clusters of graphics workstations. Various parallel rendering techniques such as sort-first and sort-last may be implemented with Chromium. Furthermore, Chromium allows filtering and manipulation of OpenGL command streams for non-invasive rendering algorithms. Chromium is a flexible framework for scalable real-time rendering on clusters of workstations, derived from the Stanford WireGL project code base.</td>
</tr>
<tr>
<td>compute node</td>
<td>Standard node in an HP XC cluster to be used in parallel by applications.</td>
</tr>
<tr>
<td>Configuration Data Files</td>
<td>Configuration Data Files provide specific information about the system configuration of an SVA. File details are mainly of interest to the system administrator who manages and configures the cluster. All visualization sessions that you initiate to run your application depend on input from the Configuration Data Files. There are three such files: Site Configuration File, User Configuration File, and Job Settings File.</td>
</tr>
<tr>
<td>display node</td>
<td>Display nodes are standard Linux workstations containing graphics cards. They transfer image output to the display devices and can synchronize multi-tile displays. The final output of a visualization application is to display a complete image that is the result of the parallel rendering that takes place during an application job. To make this possible, a display node must contain a graphics card connected to a display device. The display can show images integrated with the application user interface, or full screen images. The output can be a complete display or one tile of an aggregate display.</td>
</tr>
<tr>
<td>display block</td>
<td>The tile output from a single display node, including the relative orientation of the tiles in the case of multi-tile output generated by two ports on a single graphics card or two cards.</td>
</tr>
<tr>
<td>Display Surface</td>
<td>A Display Surface is a named assemblage of one or more display nodes, their associated display devices, including the physical orientation of the display devices relative to one another. A Display Surface is made up of the output of display nodes, that is, display blocks.</td>
</tr>
<tr>
<td>Display Surface Configuration Tool</td>
<td>Defines the arrangement of display blocks that make up a Display Surface, including the relative spatial...</td>
</tr>
</tbody>
</table>
arrangement of the display blocks. Invoked using the svadisplaysurface command. Requires root privileges.

DMX Distributed Multi-Head X is a proxy X Server that provides multi-head support for multiple displays attached to different machines (each of which is running a typical X Server).

interactive session A visualization session typically launched using a VSS script provided by HP. Such a script allocates the cluster resources, and starts the X Servers. It also starts a desktop environment (for example, KDE or Gnome) from which you can launch your applications repeatedly while retaining the same job resources. To launch your application, open a terminal window and then run your application as usual.

Job Settings File A Configuration Data File that determines the way in which a visualization job runs. The visualization job data is defined at job allocation time from options specified to the job launch scripts, from data access calls embedded in the script, and the other configuration data files. The Job Settings File is named /hptc_cluster/sva/job/<id>.conf. This file has a life span equal to that of the job.

LSF Platform Load Sharing Facility for High Performance Computing. Layered on top of SLURM to provide high-level scheduling services for the HP XC system software user. LSF can be used in parallel with SVA job launching techniques that rely on SLURM.

Node Configuration Tool Defines the display block output from a single display node, including the relative spatial arrangement of the tiles. Invoked using the svaconfigurenode command. Requires root privileges.

ParaView An open-source, multi-platform, extensible application designed for visualizing large datasets. This scalable application runs on single-processor workstations as well as on large parallel supercomputers.

Remote Graphics Software (HP) HP RGS is an HP product that facilitates access to cluster workstations from offices over a standard ethernet network. Optional purchase. Script support is provided in the SVA Kit for this product.

render node A type of visualization node used to render images. A visualization job uses multiple nodes to render image data in parallel. A render node typically communicates over the System Interconnect with other render and display nodes to composite and display images. Requires a NIC if used with HP RGS.

Site Configuration File A Data Configuration File. This file contains the default system settings and Display Surface definitions. It is generated initially by HP (and a site administrator if necessary) using the svaconfigure Utility when the cluster software is installed. Only root users can change this file. This file is named /opt/sva/etc/sva.conf.

SLURM Simple Linux Utility for Resource Management. A resource manager for Linux clusters. Used to set up visualization sessions and launch visualization jobs. Preferred allocation utility of HP XC.

svaconfigure Utility Generates the Site Configuration Data file.

System Interconnect The System Interconnect (SI) supports data transfer among HP XC cluster nodes, including visualization nodes. High-speed, low-latency networks such as InfiniBand and
Myrinet can be used for the System Interconnect to speed the transfer of image data and drawing commands to the visualization nodes.

tile
The image output from the a single port of a graphics card in a display node. Typically, a tile is also considered the image displayed on a single display device such as a flat panel or projector.

UBB
Utility Building Block (UBB). Base utility unit of a modular expandable SVA system.

UVB
Utility Visualization Block (UVB): Base utility unit of a bounded physical SVA configuration.

VBB
Utility Visualization Block (UVB): Base utility unit of a bounded physical SVA configuration.
<table>
<thead>
<tr>
<th>Page</th>
<th>Uppercase</th>
<th>Lowercase</th>
</tr>
</thead>
<tbody>
<tr>
<td>47</td>
<td>Index</td>
<td>Index</td>
</tr>
</tbody>
</table>

| A   | Admin/service node, 18 | Administrative network, 18, 21–22 | Architecture of SVA, 17 |
| B   | Beowulf cluster, 17 | Bounded configuration, 21 |
| C   | Chromium, 25 | Compilers on kit, 25 | Compute cluster components of, 18 | Compute node, 18 | Configuration data files hierarchy of, 27 | job settings, 27 | overview, 27 | site, 27 | user, 27 |
| D   | Data flow within SVA, 19 | Debugger on kit, 25 | Development tools, 25 | Diagnostics, 25 | Display flow of control for, 40 | Display devices, 22 | setting up, 13 | support for, 13 | Display node, 19 in SVA, 19 | Display surface used by Chromium-DMX script, 41 | used by ParaView script, 38 | used by RGS script, 33 | DMX, 25 launching of process, 40 |
| F   | File access, 20 | freeglut, 25 |
| H   | HP Mlib, 25 |
| I   | Interactive session how to run, 29 |
| J   | Job settings file, 27 |
| K   | Kit software installed by, 25 |
| L   | Linux clusters background on, 17 | Beowulf type, 17 | types of, 17 |
| M   | Master application running of, 29 | Mathematical Software Library, 25 | Modular configuration, 21 |
| N   | Network in SVA, 19 | Nodes types of in compute cluster, 18 | types of in SVA, 19, 21 |
| P   | ParaView example use, 35 flow of control for, 36 launching of process, 36 |
| R   | Remote Graphics Software (HP), 25 | Render node, 19 in SVA, 19 | RGS example use, 32 launched via script, 34 launching of process, 32 | RGS Display flow of control for, 32 Run application, 27 |
| S   | Sample program types of, 31 Scripts used for launching applications, 29 | Serial application example use, 38 | Site Configuration file, 27 |
SLURM
    use in job launch, 29
SVA
    architecture for, 17
    cluster components, 19
    data flow in, 19
    file access with, 20
    main tasks of, 18
    overview of, 12
    scalability of, 13
    software installed, 25
    usage model for, 11
sva_remote.sh script, 34
System interconnect, 18, 21–22

T
Templates
    modifying, 28
    parts of, 28
    selecting, 28
    steps to use, 28
    types of, 28
    used to launch application, 29
    used to run application, 27

U
UBB, 21
User configuration file, 27
UVB, 21

V
VBB, 21
VSS, 25

X
X Server, 25
XC
    components of, 24
    use with SVA, 23–24
Xfree, 25