IBM

IBM Optimized Analytic Infrastructure Solution

Installation Guide V1.0

- Financial solution that is scalable and reliable
- IBM hardware, software, and strategic software products
- Installing and configuring the components

Ira Chavis
Vivek Salve
Amarjit Bahl
Krista Morris

Redbooks

ibm.com/redbooks

International Technical Support Organization

IBM Optimized Analytic Infrastructure Solution: Installation Guide V1.0

December 2006

SG24-7381-00

Note: Before using this information and the product it supports, read the information in “Notices” on page vii.

First Edition (December 2006)

This edition applies to Version 1, Release 1 of the IBM Optimized Analytic Infrastructure solution for financial markets.

Contents

Notices

This information was developed for products and services offered in the U.S.A.

IBM® may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:
IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785 U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

Trademarks

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

1350™
AIX
AIX® 5L™
BladeCenter®
eServer™
IBM
ibm.com®
LoadLeveler®
pSeries®
Redbooks
Redbooks™ (logo)™ 
TotalStorage®
WebSphere®
xSeries®

The following terms are trademarks of other companies:

ITIL®, is a registered trademark, and a registered community trademark of the Office of Government Commerce, and is registered in the U.S. Patent and Trademark Office.

Java™, JDBC™, JDK™, JRE™, Sun™, Sun Microsystems™, and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Microsoft®, Windows® NT, Windows, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

i386™, Celeron®, Intel® Inside, Intel SpeedStep®, Intel, Itanium®, Pentium®, Xeon®, Intel logo, Intel Inside® logo, and Intel Centrino logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States, other countries, or both.

UNIX® is a registered trademark of The Open Group in the United States and other countries.

Linux® is a trademark of Linus Torvalds in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.

Preface

IBM Optimized Analytic Infrastructure (OAI) is a solution suitable for institutions seeking either to move their financial system to a scalable industrial-strength solution or to build an integrated, high-performance infrastructure from the ground up.

The aim of this guide is to help you install and configure the components of Optimized Analytic Infrastructure. It consists of IBM hardware and software, along with strategic software products from other selected vendors.

The team that wrote this redbook

This document was prepared by the Systems and Technology Group (STG) Development Center for Solution Integration, Poughkeepsie, NY.

Ira Chavis is a Certified Consulting IT Specialist in the Industry Solutions and Proof of Concept Centers in IBM Systems and Technology Group. Working in the Center for Solution Integration, he currently specializes in infrastructure architecture and solutions involving IBM server and storage technologies. He has more than 26 years of diversified software engineering and IT experience. Prior to working at IBM, Ira worked at Digital Equipment Corporation in varying assignments. Ira holds certifications as an IBM eServer Certified Expert in xSeries, IBM Grid Technical Sales, Microsoft Certified System Engineer (NT4), and Red Hat Certified Technician.

Vivek Salve is an IT Architect in the Center for the Solution Integration team in the IBM Systems and Technology Group. His areas of expertise are architecture and design of infrastructure solutions for complex business applications.

Amarjit Bahl is an IBM Certified IT Architect and a Certified Project Management Professional (PMP). He has filed three patents and published various technical papers about grid and distributed technologies. His areas of expertise are application architecture, application design, and implementation.

Krista Morris is a Certified Project Management Professional (PMP) and is also Certified in ITIL IT Service Management, currently working in the Industry Solutions and Proof of Concept Centers in IBM Systems and Technology Group. She is currently working as a Project Executive for the IBM Deep Computing Capacity in Demand offering. Krista has more than 23 years experience in information technology (IT) and has held various technical, project leadership, and executive assignments in the service and solutions development areas in IBM.

Thanks to the following people for their contributions to this project:

Mike Ebbers
International Technical Support Organization, Poughkeepsie Center

David Champagne
IBM STG Center for Solution Integration, Poughkeepsie NY

Become a published author

Join us for a two- to six-week residency program! Help write an IBM Redbook dealing with specific products or solutions, while getting hands-on experience with leading-edge technologies. You'll have the opportunity to team with IBM technical professionals, Business Partners, and Clients.

Your efforts will help increase product acceptance and customer satisfaction. As a bonus, you'll develop a network of contacts in IBM development labs, and increase your productivity and marketability.

Find out more about the residency program, browse the residency index, and apply online at:

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

We want our Redbooks to be as helpful as possible. Send us your comments about this or other Redbooks in one of the following ways:

- Use the online **Contact us** review redbook form found at:

  ibm.com/redbooks

- Send your comments in an e-mail to:

  redbooks@us.ibm.com

- Mail your comments to:

  IBM Corporation, International Technical Support Organization
  Dept. HYTD Mail Station P099
  2455 South Road
  Poughkeepsie, NY 12601-5400

1

Introduction

This guide was written to help you install and configure the components of Optimized Analytic Infrastructure (OAI), which is an infrastructure solution for financial markets. It consists of IBM hardware and software and strategic software products from other selected vendors.

Optimized Analytic Infrastructure is suitable for institutions seeking either to move their financial system to a scalable industrial-strength solution or to build an integrated, high-performance infrastructure from the ground up.

This chapter provides a brief overview of Optimized Analytic Infrastructure, describes the product portfolio, and illustrates the proof of concept environment.

1.1 Optimized Analytic Infrastructure overview

Many firms on Wall Street have been using innovative computing technology to improve business performance for many years. This technology has been used for tasks such as options pricing and valuation/risk calculations for derivatives, where greater precision and faster response times have justified the significant investments that have often been made in symmetric multiprocessing (SMP) technology. The advent of distributed computing technology and the availability of inexpensive high-performance blade factor servers have given financial firms an opportunity to explore new areas of computing power. Deployment of new technology is dramatically changing the opportunity for firms to make money in many areas of the market.

In the midst of this technology change, the basic tenets of trading in financial markets have not changed. Trading means responding to information and transferring risk. The firm that can most accurately analyze information, respond the quickest, and transfer risk most efficiently will establish market leadership. This leadership ultimately translates into the opportunity for significant market share and profits. In seeking to gain this leadership, firms continue to optimize the trading and risk management process, continually looking to save 10 or 20 milliseconds from the process. Although this might not seem significant, it can be the difference between transacting and not transacting. This is especially true in areas such as algorithmic trading, where latency that exceeds a few milliseconds more than the time it takes light to travel a physical distance is considered problematic.

Mindful of these technology trends and the constant need for innovation, IBM has developed an infrastructure solution for financial markets, called *Optimized Analytic Infrastructure*.

Optimized Analytic Infrastructure is suitable for institutions that are seeking either to make the most of their existing infrastructure or that are implementing a new foundation for high-performance infrastructure that significantly surpasses current offerings in this space today.

To build this solution, IBM used its client experience and deep expertise in high-performance computing (HPC), grid computing, Linux, service-oriented architectures (SOAs), and mathematical research. The result is a high-performance computing solution that is specifically tailored to meet the requirements of financial markets. Simply put, the Optimized Analytic Infrastructure was designed to support a broad spectrum of numerically-intensive business processes and applications.

There are four attributes that make the Optimized Analytic Infrastructure different from other approaches to infrastructure:

- It is inherently a *shared environment* that is able to support different applications and a spectrum of different work loads.
- It was built by *leveraging open standards* such as Linux and POSIX and has the flexibility to support third-party hardware and software configurations.
- The Optimized Analytic Infrastructure is one of the most *scalable infrastructure* solutions that are available for the financial services market segment today. It can scale in multiple sites, clusters, and grids.
- IBM has designed this environment to be *service-oriented*, which allows computing-intensive applications to take advantage of HPC services more easily.

The Optimized Analytic Infrastructure was designed to support a broad spectrum of numerically-intensive business processes and applications. It combines clustering, virtualization, and grid technologies into an infrastructure solution for hosting complex analytic applications, whether that infrastructure is a single departmental cluster or an enterprise-wide

utility. It combines high performance and high availability clustering with greater resource utilization and efficient data access through virtualization and grid technologies. In addition, it uses an SOA to make HPC applications accessible in on demand, real-time environments.

The key features of the Optimized Analytic Infrastructure that address the business problems described at the beginning of this section are:

- Numerical applications that can be accessible as a service
- Policy-based resource sharing
- Multisite, multicluster support to enable both computational and data virtualization
- Support for highly scalable and potentially highly distributed systems
- Extremely lightweight system software stacks
- Support for native code applications (such as C, C++, and Perl) and for a range of computational complexity
- Development and runtime tooling that hide the complexities of parallel programming and interfacing from grid middleware
- The ability to run applications in dedicated or shared mode as required, for example, sharing or giving one user exclusive access to processes that are running on servers
- An infrastructure that favors scale-out capacity (many commodity computers) versus scale-up (with large SMPs) while being able to support both
- Support for a broad spectrum of workloads, including real time and batch
- Support for performance-driven and scalable I/O in a distributed environment
- Support for pre-emption of jobs, backfill scheduling, advanced reservation of resources, and recurrent scheduling
- Resource virtualization with an innovative cluster and HPC environment that includes (but is not limited to) shared compute nodes across applications, remote job scheduling, federated data in cluster file systems, network switch provisioning switches, and high availability
- Customer support and service for the entire system from a single point of contact

The Optimized Analytic Infrastructure is a fully tested system solution. It brings together an optimized union of operating system, hardware, and (most importantly) the management of systems, workload, data, and applications.

The solution components include:

- Hardware resources for computations (server, storage, and network).
- Linux as the base operating system for each computing resource in the system, providing a distributed system environment stack consisting of system management, scheduling, resource management, data management, communications, and application development. It supports two-tier (client-job/service) and three+ tier (such as client-job/service-workers and so on) models.
- Middleware (provisioning systems, databases, and message queuing systems).
- An environment that provides interfaces and services that the applications (calculations, spreadsheets) and application servers such as IBM WebSphere can use productively.

Figure 1-1 illustrates the hardware, operating system, and management framework that makes up the Optimized Analytic Infrastructure.

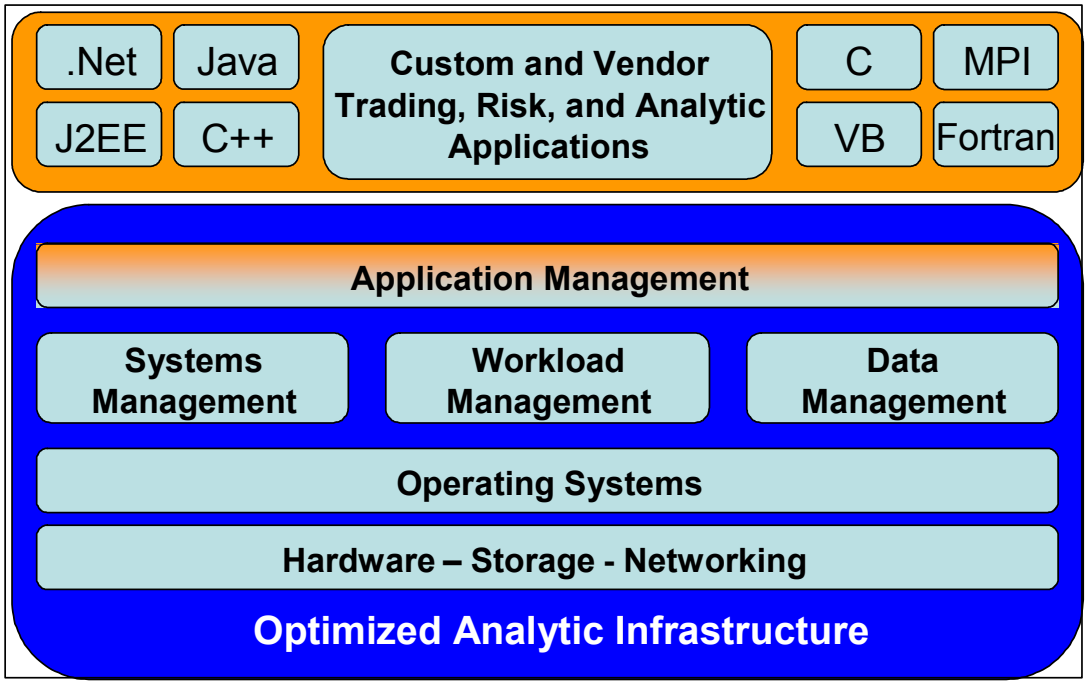

Figure 1-1 Optimized Analytic Infrastructure overview

The systems management component performs:

- Remote hardware control
- Hardware installations
- Inventory and monitoring (network, storage, and server)

In the context of the Optimized Analytic Infrastructure, the systems management component provides scalable and centralized configuration, monitoring, and events, such as remote deployment and monitoring of clusters. The systems management component incurs very low system processing resources for all system management functions and the support of a multicluster, multisite model. It also provides external monitoring of application performance (that is, client-side throughput), because source code instrumentation of independent software vendor (ISV) applications is almost always impossible.

The workload management component handles:

- Job queuing
- Scheduling
- Utilization monitoring

Workload management in the Optimized Analytic Infrastructure provides job scheduling to a set of resources. Remote and distributed job scheduling products, such as IBM LoadLeveler, Condor, HP Platform LSF, and PBS Professional, have long existed in the HPC cluster space. These were developed to keep servers and idle workstations busy by running numerically intensive computing simulations. For the Optimized Analytic Infrastructure, IBM built on a base of lightweight, highly scalable scheduling technology, including resource matching, backfill scheduling for resource utilization, optimization for long running parallel batch jobs, preemption, gang scheduling, and advanced reservation. A far more interactive environment is available for users, making it possible for financial services firms to take advantage of the high-performance scheduling technology without the previously

encountered inflexibilities. This increased interactivity includes functions such as policy-based resource sharing in scheduling decisions where the policies have business value associated with the application and resulting computation.

The data management component manages file and database sharing and remote data access. It is a critical component because all financial markets applications generally input and output huge volumes of data. At the very high-end, a single job in a job stream might process hundreds of gigabytes of data. The performance of such an application is tightly coupled to the I/O throughput rate at which this data can be moved to and from disks, because such a large amount significantly exceeds system memory capacity. The Optimized Analytic Infrastructure uses a high-performance parallel file system and data caching technology to manage these large files to maintain high levels of bandwidth between system memory and disks even for multiple sites.

The application management component is designed to:

- Build, debug, and deploy modules
- Handle input/output
- Provide exception handling/signaling
- Manage transactions

The Optimized Analytic Infrastructure is special because of the distributed processing development, low latency runtime tooling, and the application environment flexibility. The application environment supports native applications that are serial, easily parallellized, or of significant communications complexity. An example is latency-driven collective communications, or global synchronization of numerical processes. This is not only a runtime consideration, but it is also a requirement for the tooling and development environment, including compilers, debuggers, performance tracing tools, and code generation tools. For the Optimized Analytic Infrastructure, open standards are used so that any tool-generating code can be readily available for many application development platforms.

Figure 1-2 illustrates the hardware, operating system, systems management, workload management, data management, and application management products that were used for the Optimized Analytic Infrastructure test environment.

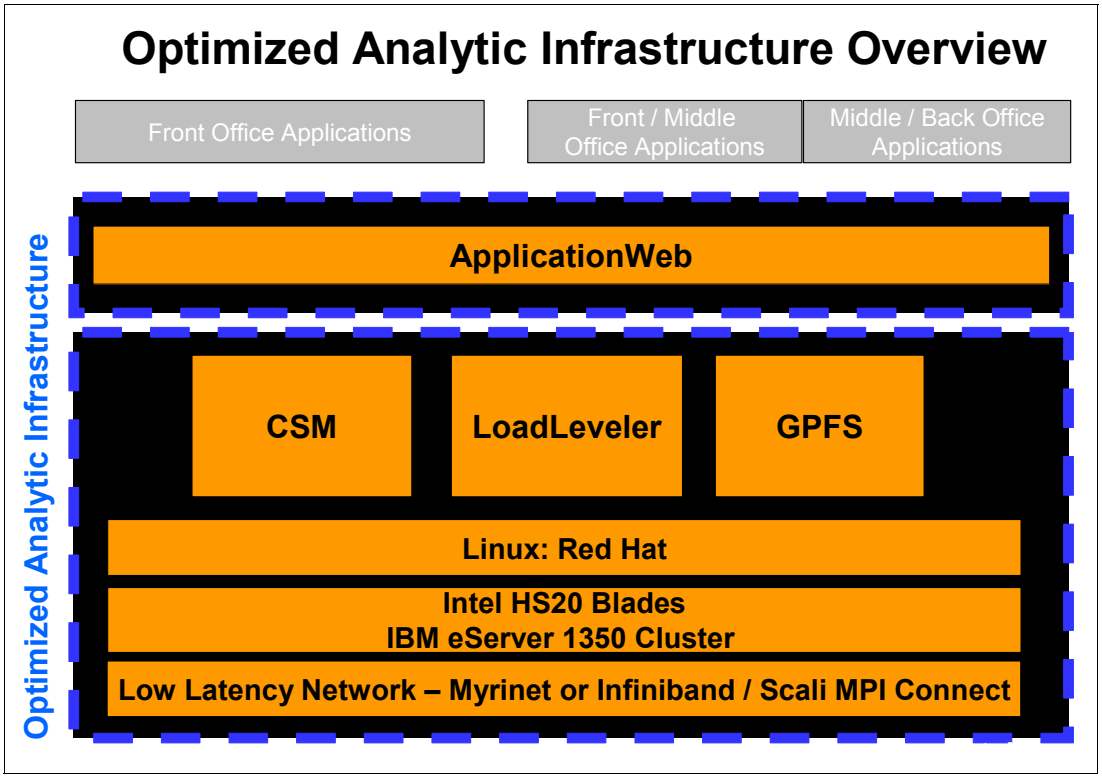

Figure 1-2 Optimized Analytic Infrastructure implementation overview

1.2 Product portfolio

The Optimized Analytic Infrastructure solution has the following software components.

1.2.1 IBM products

The IBM products are:

- IBM Cluster Systems Management

  IBM Cluster Systems Management (CSM) for AIX 5L and Linux is designed for simple, low-cost management of distributed and clustered IBM eServer pSeries and xSeries servers in technical and commercial computing environments. CSM, which is included with the IBM eServer Cluster 1600 and Cluster 1350, dramatically simplifies administration of a cluster by providing management from a single point-of-control. CSM is available for managing homogeneous clusters of xSeries servers that are running Linux or pSeries servers that are running AIX 5L, or heterogeneous clusters that include both.

- IBM ApplicationWeb

  Application management in the Optimized Analytic Infrastructure is provided by IBM ApplicationWeb, which is a differentiating technology from IBM Research. This technology combines application tooling and a runtime environment to offer:

  - Easy porting of applications from serial to distributed environments with higher level message passing abstractions such as each, any, and all (unicast, anycast, and broadcast).
  - Easy application building with tooling that allows heterogeneous clients (Excel/VB, Java, MATLAB, .NET) to access networked, shared objects and libraries for a variety of platforms (such as Linux/C/C++, MATLAB, S+, and R+).
  - Innovative external RPC and marshalling/demarshalling technology that enables you to port applications with non-intrusive code modifications. ApplicationWeb tooling can move from header files and Eclipse-based tooling to create applications from shared objects without modifying source code in the shared object.
  - Easy application migration by abstracting the developer from the underlying runtime infrastructure to distribute the workload (for example, MPI, Sockets, GemStone®, DS) and the underlying client connection protocol (JMS, HTTP, SOAP) so that various infrastructures and protocols can be easily substituted.
  - Additional performance benefit because of in-memory internal and low-level external messaging.
  - High-level distributed computing semantics (each, any, and all) to facilitate application porting from serial to distributed environments quickly.
  - Dynamic binding of native libraries and subroutines from local or remote heterogeneous clients (for example, calling C++/Linux numerical libraries from an Excel/VB client).
  - Tooling to support the development of applications.
  - An abstraction layer between the application developer and the underlying infrastructure so that the application writer need only use the MathShell tooling to provide language conversion bindings and so that no changes are required by the application writer to move from one run time (for example, DataSynapse) to another run time (for example, LL/PE).

  ApplicationWeb accomplishes all of these significant capabilities in an extremely lightweight package with no material impact to latency or system resources, compared with hand coding low-level communications directly in the applications.

- IBM General Parallel File System

  IBM General Parallel File System (GPFS) is a high-performance, shared-disk file system that can provide fast data access from all nodes in a homogenous or heterogeneous cluster of IBM UNIX servers running either the IBM AIX 5L or a Linux operating system.

  GPFS gives parallel applications simultaneous access to the same files, or different files, from any node that has the GPFS file system mounted on it while managing a high level of control over all file system operations.

  GPFS provides high availability through logging and replication and can be configured for failover from both disk and server malfunctions. It is designed to provide high-performance I/O by “striping” data.

- IBM LoadLeveler

  IBM LoadLeveler provides a facility for building, submitting, and processing jobs—batch, interactive, serial, and parallel—in a dynamic environment. It is designed to match application processing needs with available resources, which increases job throughput in the cluster and minimizes individual job turnaround times. Job requirements can include a combination of memory, disk space, processor types, operating systems, and application programs. LoadLeveler collects resource information and dispatches the job when it locates suitable servers.

1.2.2 ISV products

The ISV products for Optimized Analytic Infrastructure are:

- DataSynapse GridServer

  DataSynapse GridServer is application infrastructure software that virtualizes and distributes application services and executes them in a guaranteed, scalable manner over existing computing resources. By creating a virtualized environment that matches demand (user/application service requests) and supply (available system and data resources), GridServer provides an on demand environment that dramatically improves application performance, resiliency, and uptime. With its patent-pending technology, GridServer creates a highly scalable and flexible architecture that addresses a wide range of application models and resource types.

- GemStone GemFire Enterprise™

  GemFire Enterprise is a high-performance data services software solution that makes data available at the time that it is needed to applications regardless of the underlying data sources or formats. It is built on one of the industry's fastest and most reliable data distribution and caching system that is proven in mission-critical programs used by financial services firms, telecommunications companies, the energy industry, logistics businesses, and the federal government.

  GemFire Enterprise maintains data in multiple formats, whether that means objects, database tables, or flat data structures, and allows applications and users to interface with the fabric in a seamless, non-intrusive manner. It also ensures high availability of data through mechanisms such as mirroring and data replication and provides the basis for a reliable data infrastructure. GemFire Enterprise is based on most well-known technology standards (for example, JDBC, JMS, JCache, SOAP, HTTP, and Web services) and seamlessly connects to databases, application stores, analytical tools, messaging systems, and mainframes, enabling the deployment of high-performance architectures at significantly lower costs. GemFire-based architectures are characterized by reduced data latency, enhanced scalability, and increased performance for new application server, portal, and integration deployments.

- Scali MPI Connect

  Scali MPI Connect is a fully integrated message passing interface (MPI) solution. Companies can use Scali MPI Connect to take advantage of leading interconnect technologies to build high-performance clusters. The Scali MPI delivers high-bandwidth, low-latency performance to clusters with communication libraries that enable high performance execution of applications that are designed for parallelism and portability.

- Altair PBS Professional

  The Portable Batch System, PBS, is a leading workload management solution for HPC systems and Linux clusters. PBS was originally designed for NASA, because at that time, existing resource management systems were inadequate for modern parallel/distributed computers and clusters. From the initial design forward, PBS has included innovative new approaches to resource management and job scheduling, such as the extraction of scheduling policy into a single, separable, completely customizable module.

  The professional edition, PBS Professional, operates in networked multiplatform UNIX, Linux, and Microsoft Windows environments and supports heterogeneous clusters of workstations, supercomputers, and massively parallel systems. Sites using PBS Professional to manage their computing resources can expect many tangible benefits, including:

  - Increased utilization of costly resources (both hardware and software)
  - Unified interface to all computing resources
  - Reduced burden on system administrators, freeing them to focus on other activities
  - Enhanced understanding of computational requirements and user needs by very accurate accounting data and reports
  - Expandability

  PBS Professional supports dynamic distribution of production workloads across wide-area networks, and the logical organization of physically separate computing systems.

- Platform LSF

  The Platform LSF family of products is the industry's most powerful and comprehensive family of grid-enabled solutions. This product family helps manage and optimize expensive and complex IT environments, delivering higher IT efficiency, faster time to business results, dramatically reduced cost of computing, and guaranteed service execution.

  The Platform LSF family of products is a superior grid-enabled solution that is optimized for solving technical computing problems for such segments as the electronics industry, including semiconductor design, government and research for aerospace and defense contractors, automotive industrial manufacturers, and life sciences organizations such as biotechnology firms.

  Platform LSF fully uses all IT resources regardless of operating system, including desktops, servers, and mainframes, to ensure policy-driven, prioritized service levels for “always-on” access to resources.

- Platform Symphony

  Platform Symphony uses your existing heterogeneous IT resources to manage and accelerate single or multiple applications and their workloads for individual or multiple lines-of-business in a shared, scalable, and fault-tolerant infrastructure.

  With Platform Symphony, you can:

  - Boost revenue and profit by executing more and greater complexity computations
  - Increase the fidelity of computing-intensive simulations
  - Reduce IT operating, maintenance, support and application development costs
  - Tie more processes into the grid for faster ROI and increased application productivity and IT reliability

1.3 The proof of concept environment

Figure 1-3 shows the testing environment that we set up.

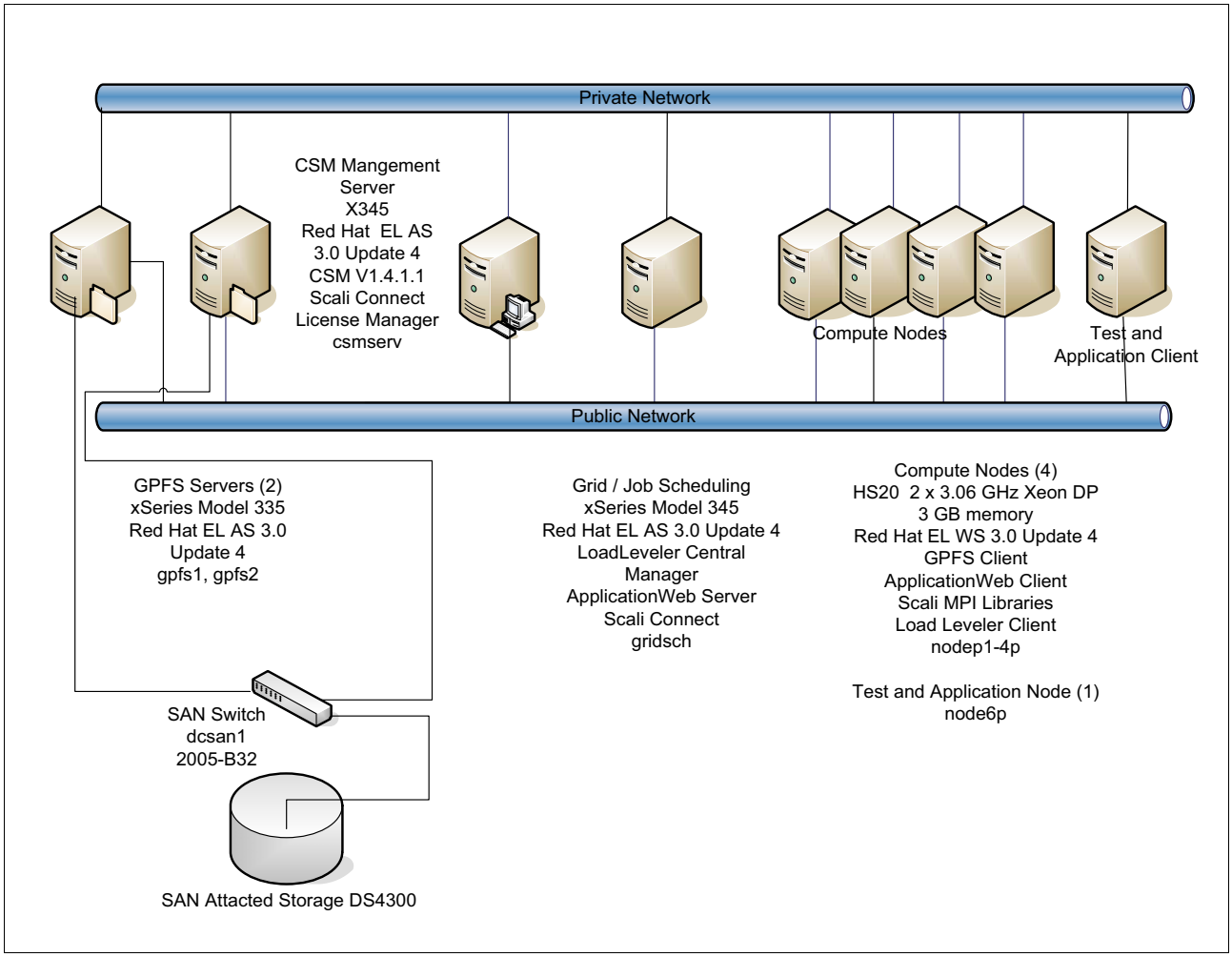

Figure 1-3 Optimized Analytic Infrastructure network infrastructure

The environment consisted of a CSM management server, two GPFS servers, a grid/job scheduling server, and four compute nodes in a 1350 cluster.

The servers and compute nodes were built with the following operating systems and software:

- CSM management server
  - Red Hat Enterprise Linux V3.0 Advanced Server Update 4 - 32 bit
  - Cluster Systems Management V1.4.1
  - Scali MPI Connect License Manager V4.4.0
- GPFS primary Network Shared Disk (NSD) server
  - Red Hat Enterprise Linux V3.0 Advanced Server Update 4 - 32 bit
  - GPFS Server V2.2.0-3
- GPFS backup NSD server
  - Red Hat Enterprise Linux V3.0 Advanced Server Update 4 - 32 bit
  - GPFS Server V2.2.0-3

- Grid/job scheduler server
  - Red Hat Enterprise Linux V3.0 Advanced Server Update 4
  - LoadLeveler Central Manager V3.2.1
  - IBM ApplicationWeb server V1.0.5
  - Optional ISV software: Scali MPI Connect 4.4
- Four managed compute nodes

  The four managed compute nodes were for testing purposes only. A production infrastructure configuration was deployed in an IBM eServer Cluster 1350 consisting of significantly more managed compute nodes:

  - Red Hat Enterprise Linux V3.0 Workstation Update 4
  - GPFS Client V2.2.0-3
  - IBM ApplicationWeb Client V1.0.5
  - LoadLeveler Client V3.2.1
  - Optional ISV software: Scali MPI Connect Libraries V4.4

Note: Using other processor architecture, operating system versions, and product versions might produce unpredictable results, and therefore it is not recommended.

The rest of this guide describes how to install these components as part of the Optimized Analytic Infrastructure environment.

2

Installing the environment

This chapter describes how to install the environment for IBM Optimized Analytic Infrastructure. This chapter addresses the following topics:

- Network infrastructure
- Installing and configuring IBM BladeCenter
- Installing and configuring the CSM management server
- Installing Scali MPI Connect
- Installing the grid/job scheduling server
- Installing the LoadLeveler Central Scheduler
- Installing the ApplicationWeb server
- Installing the GPFS servers
- Creating the GPFS cluster
- Configuring the CSM management server files for GPFS clusters

2.1 Network infrastructure

The network infrastructure for this solution requires that all servers and compute nodes be multihomed. This means that they are connected to a management network and a private network. The BladeCenter management module is the only network component that cannot be connected to multiple networks because it only has a single network interface. The requirements for networks are as follows:

- Management network: Used for management and administration network access between servers and compute nodes, the management network is also used for NFS access between servers in the environment. All servers have public IP addresses and can be accessed from anywhere in the network.
- Private network: Used for compute, GPFS, MPI, and grid/scheduling communications, this is a non-routable network segment and thus has no router/gateway.

Table 2-1 and Table 2-2 on page 15 list the management and private network information for the Optimized Analytic Infrastructure.

Table 2-1 Management network

Server	Public network host name	Public network IP address	Public network IP gateway	Public network netmask	Network interface
CSM management server	csmserv	192.168.10.1	192.168.10.254	255.255.255.0	eth0
BladeCenter management module	bcmm	192.168.10.253	192.168.10.254	255.255.255.0	N/A
GPFS primary server	gpfs1	192.168.10.2	192.168.10.254	255.255.255.0	eth0
GPFS secondary server	gpfs2	192.168.10.3	192.168.10.254	255.255.255.0	eth0
Grid/job control scheduler	gridsch	192.168.10.4	192.168.10.254	255.255.255.0	eth0
Compute node 1	node1	192.168.10.5	192.168.10.254	255.255.255.0	eth1
Compute node 2	node2	192.168.10.6	192.168.10.254	255.255.255.0	eth1
Compute node 3	node3	192.168.10.7	192.168.10.254	255.255.255.0	eth1
Compute node 4	node4	192.168.10.8	192.168.10.254	255.255.255.0	eth1

Table 2-2 Private network

Server	Private network host name	Private IP address	Private network IP gateway	Private network netmask	Network interface
CSM management server	csmservp	192.168.20.1	192.168.20.254	255.255.255.0	eth1
BladeCenter management module	N/A	N/A	192.168.20.254	N/A	N/A
GPFS primary server	gpfs1p	192.168.20.2	192.168.20.254	255.255.255.0	eth1
GPFS secondary server	gpfs2p	192.168.20.3	192.168.20.254	255.255.255.0	eth1
Grid/job control scheduler	gridschp	192.168.20.4	192.168.20.254	255.255.255.0	eth1
Compute node 1	node1p	192.168.20.5	192.168.20.254	255.255.255.0	eth0
Compute node 2	node2p	192.168.20.6	192.168.20.254	255.255.255.0	eth0
Compute node 3	node3p	192.168.20.7	192.168.20.254	255.255.255.0	eth0
Compute node 4	node4p	192.168.20.8	192.168.20.254	255.255.255.0	eth0

2.2 Installing and configuring BladeCenter

The BladeCenter management module must be installed for each chassis. See *Installing the IBM eServer BladeCenter management module* or *IBM eServer xSeries and BladeCenter Server Management*, SG24-6495, for more details.

The following settings or configuration changes must be made to the BladeCenter to work in the Optimized Analytic Infrastructure environment:

- Configure the IP address of the BladeCenter management module. You must use a cross-over cable to access the factory default address for the BladeCenter management modules.
- Set the BladeCenter Management Module name. In our example, we named it *bcmm*.
- Change the BladeCenter default password from PASSW0RD to something else.
- Set the names for each Blade server. In our example, we used the names node1, node2, and so forth.
- Check the network switch module settings (you might need to change these, depending on your environment if there are problems with DHCP, and so forth):
  - Turn off **Port Fast enable** on BladeCenter Server network ports.
  - Turn off **spanning tree** on BladeCenter Server network ports.

2.3 Installing and configuring the CSM management server

The CSM management server is the core component of the Optimized Analytic Infrastructure architecture. The server is responsible for providing a single point of control for managing compute nodes in this environment and supplies the methods for provisioning and configuring these servers. This minimizes the installation and configuration time for provisioning and orchestration.

2.3.1 Installing the base Linux operating system on the CSM management server

To install the base Linux operating system on the CSM management server, follow these instructions:

1. Install Red Hat Enterprise Linux v3.0 Advanced Server Update 4 on the CSM management server.
2. Enable NFS:

   ```
   chkconfig -add nfs
   chkconfig nfs on
   ```

3. Create a shared directory to be used by all servers and compute nodes to share packages and scripts. This directory is called /dist:

   ```
   mkdir -p /dist
   ```

4. Create an entry in the /etc/exportfs configuration to define the /dist file system to other nodes:

   ```
   vi /etc/exportfs
   ```

   Add this to the file:

   ```
   /dist   *.ihost.com(rw,no_root_squash,insecure,sync)
   ```

   Change the *.ihost.com to your current DNS domain or machine list (see NFS documentation for more examples).

5. After saving the /etc/exportfs file, you must restart NFS to export the /dist file system that you just added:

   ```
   service nfsd restart
   ```

6. To verify the NFS export is in effect, issue:

   ```
   exportfs
   ```

   The **exportfs** command results in a list of all file systems that have been exported by this server. An example of the **exportfs** command output is:

   ```
   /dist *.pbm.ihost.com
   ```

2.3.2 Installing the CSM management server

To install the CSM management server, follow these instructions:

1. You must download several packages before CSM can be installed. Because you are installing CSM on an x86 and Red Hat Enterprise Linux environment, you must download the following RPMs, which are copied to the /tmp/csm/csm_reqs directory:
   - syslinux 2.10 (because of bugs with later versions)
   - autoupdate RPMs
     - autoupdate-5.4-1.noarch.rpm
     - autoupdate-cfg-autoupdate-5.4-1.noarch.rpm
     - autoupdate-cfg-redhat-5.4-1.noarch.rpm

   See the *CSM Installation Guide* for the specific packages that need to be downloaded onto your CSM management server. This depends on your server architecture and operating system vendor.

2. Create the /csminstall directory:

   ```
   mkdir -p /csminstall
   ```

3. Copy the CSM distribution from your media or file server to the /tmp directory. In our example, the CSM installation package was located in the /dist/csm directory:

   ```
   cp /dist/csm/csm-linux-1.4.1.1.i386.tar.gz  /tmp
   ```

4. Create a temporary directory for the CSM packages:

   ```
   mkdir /tmp/csm
   ```

5. Extract the CSM install packages to the /tmp/csm directory:

   ```
   tar -zxvf /tmp/csm-linux-1.4.1.1.i386.tar.gz -C /tmp/csm
   ```

6. Copy the CSM license file to /tmp. This should have been received with your CSM software. Our CSM license file is called csmlum.full and it was located on the /dist/csm file share:

   ```
   cp /dist/csmlum.full /tmp
   ```

7. Change the directory to the CSM installation directory:

   ```
   cd  /tmp/csm
   ```

8. Install the CSM core packages:

   ```
   rpm -ivh csm.core-*
   ```

9. Install the CSM management server. If you have Red Hat EL 3 AS Update 4 CDs, issue:

   ```
   /opt/csm/bin/installms -p /tmp:/tmp/csm
   ```

   Have your CDs ready; you must insert them one at a time.

   ```
   /opt/csm/bin/installms -p tmp:/tmp/csm/:
   ```

   Note: While inserting and copying CDs, you might receive a message that states that CSM does not recognize the correct distribution media. Make sure that you have inserted the correct distribution media. If you are sure that you have the correct media, press Control+F to force the mounting of the distribution CD.

2.3.3 Installing the CSM management server license

You must install the full use license for CSM. In step 6 you copied the CSM license file to the /tmp directory. To install the server license, follow these instructions:

1. Issue:

   ```
   csmconfig -L /tmp/csmlicense.txt
   ```

2. Select your installation language. We chose English.
3. Accept the terms of the license.

2.3.4 Creating /etc/hosts

Although you can use DNS to define servers and compute nodes for your cluster, it is advisable to use a /etc/hosts file on each individual system. Having a local /etc/host file also increases name lookup performance.

Create a /etc/hosts file that contains all of the CSM, LoadLeveler Central Scheduler, GPFS primary, GPFS backup, and compute nodes. Add public and private networks to the host file.

Your /etc/hosts file should resemble Example 2-1.

Example 2-1 Sample /etc/hosts file

```
# Do not remove the following line, or various programs
# that require network functionality will fail.

/etc/hosts
127.0.0.1       localhost.localdomain localhost
192.168.10.253  bcmm.pbm.ihost.com bcmm# bladecenter management module
192.168.20.1    csmservp.pbm.ihost.com csmservp    # csm management
192.168.20.4    gridschp.pbm.ihost.com gridschp    # ll central scheduler
192.168.20.2   gpfs1p.pbm.ihost.com gpfs1p    # gpfs primary
192.168.20.3   gpfs2p.pbm.ihost.com gpfs2p    # gpfs backup
192.168.20.5   node1p.pbm.ihost.com node1p    # compute node 1
192.168.20.6   node2p.pbm.ihost.com node1p    # compute node 2
192.168.20.7   node3p.pbm.ihost.com node1p    # compute node 3
192.168.20.8   node4p.pbm.ihost.com node1p    # compute node 4
192.168.10.1    csmserv.pbm.ihost.com csmserv  # csm management public
192.168.10.4   gridsch.pbm.ihost.com gridsch   # loadleveler central scheduler public
192.168.10.2    gpfs1.pbm.ihost.com gpfs1       # gpfs primary public
192.168.10.3    gpfs2.pbm.ihost.com gpfs2       # gpfs backup public
192.168.10.5    node1.pbm.ihost.com node1     # compute node 1 public
192.168.10.6    node2.pbm.ihost.com node2     # compute node 2 public
192.168.10.7    node3.pbm.ihost.com node3     # compute node 3 public
192.168.10.8     node4.pbm.ihost.com node4     # compute node 4 public
```

Rename the soft link from the /etc/hosts file to the /cfmroot/etc/hosts file so that any changes to the management server copy will be updated on each compute node:

```
mv /cfmroot/etc/hosts._EmptyGroup /cfmroot/etc/hosts.
```

2.3.5 Configuring DHCP

Some network configurations require reconfiguring which subnet DHCP listens to for requests by changing the /etc/sysconfig/dhcpd file on the CSM management server. In our configuration, eth0 is our public network interface and eth1 is the private compute network. Because DHCP listens from both interfaces by default, we configured DHCP to only listen on our private network, eth1. To do this, follow these instructions:

1. Edit the /etc/sysconfig/dhcpd file:

   ```
   vi /etc/sysconfig/dhcpd
   ```

2. Change the line with DHCPDARGS as follows:

   ```
   # Command line options here
   DHCPDARGS="eth1"
   ```

3. Restart the DHCP server:

   ```
   service DHCP restart
   ```

2.3.6 Copying CSM files to the CSM management server

If the operating system of the CSM management server and the CSM managed nodes are different, the source files must be copied from the installation media. (In our case, our CSM management server ran Red Hat Enterprise Linux V3.0 Update 4 Advanced Server and our CSM managed nodes ran Red Hat Enterprise Linux V3.0 Update 4 Workstation.) Do this using the **copycsmpkgs** command:

```
copycsmpkgs -c -p /tmp/csm InstallDistributionName=RedHatEL-WS
InstallServiceLevel=QU4
```

You will be asked to insert each of the four Red Hat 3.0 Update 4 WS CDs.

2.3.7 Storing the BladeCenter management module password

You must store the user ID and password required for internal programs to access remote hardware such as RSA cards and BladeCenter management modules. In our example, our compute cluster was in a BladeCenter chassis. We stored the BladeCenter management module user ID and password into the CSM configuration using the **systemid** command:

```
systemid <management_module_name>  <user_id>
```

Where *management_module_name* is the IP address or host name of the BladeCenter management module and *user_id* is the user ID that is used for hardware control of the target host.

After entering the command, you are prompted for a password. This must match the user ID and password for the BladeCenter management modules.

For example:

```
systemid bcmm.pbm.ihost.com USERID
New Password:
Verifying, please re-enter password:
systemid: following entries were created
192.168.10.253
```

2.3.8 Fixing syslog to accept remote messages

You must change the syslog daemon to accept remote message requests so that you can log messages from your CSM managed compute nodes as follows:

1. Issue:

   ```
   vi /etc/sysconfig/syslog
   ```

2. In the file, locate the line that starts with SYSLOGD_OPTIONS.
3. Change the line to look like this:

   ```
   SYSLOGD_OPTIONS="-m 0 -r"
   ```

4. Restart the syslog daemon so that the changes take effect:

   ```
   service syslog restart
   Shutting down kernel logger:                               [  OK  ]
   Shutting down system logger:                               [  OK  ]
   Starting system logger:                                    [  OK  ]
   Starting kernel logger:                                    [  OK  ]
   ```

2.3.9 Defining the compute nodes for the CSM management server

You must now define the compute nodes to be used for your configuration with the CSM **definenode** command. The **definenode** command runs in the CSM management server to define all the nodes in a CSM cluster. The command creates node definitions in the CSM database. To simplify configuration of the CSM managed nodes, create a text file that contains the CSM managed node attributes:

1. Create a /tmp/csmnodes.def file.
2. The file must contain entries for each node in your cluster. In our test configuration, there are only four nodes: node1, node2, node3, and node4.

   Note: Make sure that the node names that you specify in the HWControlNodeId attribute match the names you specified in the BladeCenter management configuration.

3. Issue:

   ```
   vi /tmp/csmnodes.def
   ```

   Example 2-2 shows a sample configuration.

Example 2-2 Sample results

```
default:
ConsoleMethod=blade
PowerMethod=blade
        HWControlPoint=bcmm
        InstallAdapterName=eth0
        ManagementServer=csmservp
        InstallCSMVersion=1.4.0
        InstallOSName=Linux
        InstallDistributionName=RedHatEL-WS
        InstallDistributionVersion=3
        InstallPkgArchitecture=i386
        InstallServiceLevel=QU4
        InstallDiskType=ide

node1p :
        HWControlNodeId=node1
node2p :
        HWControlNodeId=node2
node3p :
        HWControlNodeId=node3
node4p :
        HWControlNodeId=node4
```

4. Save the CSM node attribute file.
5. Define the nodes to the CSM management cluster:

   ```
   definenode -f /tmp/csmnodes.def
   ```

   You should see the following output, which indicates that all nodes have been defined:

   ```
   Defining CSM Nodes:
   4 nodes have been defined successfully.
   ```

2.3.10 Downloading the latest CSM drivers

To make sure that you have the most recent CSM drivers, take the following steps:

1. Visit this Web site:

   http://techsupport.services.ibm.com/server/cluster/fixes/csmdriverdownload.html

2. Check for any updated drivers for CSM.
3. If there are updated drivers, follow the instructions for downloading them.
4. After you download the appropriate driver TAR files, you need to extract them into the correct locations on your management server. Before running the **csmsetupks** command, run the following commands to create the directory and extract the files:

```
mkdir -p /csminstall/csm/drivers
cd /csminstall/csm/drivers
tar -xzvf <driver_tarfile>.tar.gz
```

2.3.11 Configuring the default shell for CSM

CSM can use either Remote Shell (RSH) or Secure Shell (SSH) for remote shell functionality. In our configuration, we used RSH. For RSH, issue:

```
csmconfig RemoteShell=/usr/bin/rsh SetupRemoteShell=1
```

If you want to use SSH, use the following command instead:

```
csmconfig RemoteShell=/usr/bin/ssh SetupRemoteShell=1
```

If you do not want CSM to set up the shell automatically, set `SetupRemoteShell=0`.

2.3.12 Verifying the CSM installation

To verify the CSM installation, issue:

```
probemgr -p ibm.csm.ms -l 0
```

The result is several pages of installation verification. If you see error messages or codes, consult the CSM technical documentation for diagnostic procedures.

2.3.13 Testing the CSM installation

At this point in the configuration of the Optimized Analytic Infrastructure, it is a good idea to test some of the functions of the CSM management server to see if the installation has been successful.

The **rpower** command remotely turns the CSM managed server on or off. The **rpower** command communicates with the management module or service processor of the target server. Try turning the nodes on and off as follows:

```
rpower -n node1p off
Node1p.pbm.ihost.com off complete rc=0
rpower -n node1p on
Node1p.pbm.ihost.com on complete rc=0
```

Try turning all nodes on as follows:

```
rpower -a on
Node1p.pbm.ihost.com on complete rc=0
Node2p.pbm.ihost.com on complete rc=0
Node3p.pbm.ihost.com on complete rc=0
Node4p.pbm.ihost.com on complete rc=0
```

For procedures for testing the CSM management server, refer to the CSM documentation.

2.3.14 Time service in a CSM cluster: Configuring NTP

Although running network time synchronization in a CSM cluster is not required for CSM, there are important advantages to doing so. If you are not running a time service, we encourage you to install one.

See /opt/csm/samples/docs/ntp.README on your CSM management server for information about how to install and configure NTP on your CSM management server.

2.3.15 Creating the LoadLeveler user account

LoadLeveler uses a common user account and shared home directory between all the nodes.

Create the account in the CSM management server. When a node is provisioned by the **installnode** command, the account is created in the compute nodes. This is done when CSM synchronizes the cluster password and group files in all of the CSM managed nodes:

```
groupadd -g 1001 loadl
useradd -c "LoadLeveler User" -d /home/loadl -s /bin/bash -u 1001 -g 1001 -m loadl
```

2.3.16 Setting PATH for LoadLeveler in the compute nodes

To set up the LoadLeveler environment on each compute node, create a profile to be distributed to the compute nodes that adds the LoadLeveler binary directories and man pages to your current login profile. CSM distributes this profile during the **installnode** operation, which builds the initial compute node environment:

1. If the directory does not exist, issue:

   ```
   mkdir -p /cfmroot/etc/profile.d
   ```

2. Create /cfmroot/etc/profile.d/loadl.sh so that it has the script shown in Example 2-3.

Example 2-3 /cfmroot/etc/profile.d/loadl.sh script

```
LOADL_HOME=/opt/ibmll/LoadL/full
export LOADL_HOME
PATH=$PATH:$LOADL_HOME/bin
export PATH
MANPATH=$MANPATH:$LOADL_HOME/man
export MANPATH
```

3. Issue:

   ```
   chmod 755 /cfmroot/etc/profile.d/loadl.sh
   ```

4. No other action is required, because Configuration File Manager (CFM) is run automatically during the normal full installation of CSM.

2.3.17 Creating a shared .rhosts file for all nodes

Because RSH and dancer's/digital shell (DSH) is done from your GPFS, CSM, and LoadLeveler Central Scheduler nodes to each compute node, you use CFM to distribute a /root/.rhosts file to each compute node as follows:

1. Create a root directory under /cfmroot:

   ```
   mkdir -p /cfmroot/root
   ```

   You must create a /cfmroot/root/.rhosts file that contains a list of authorized host names that allow remote hosts to access this machine with RSH. These hosts should include:

   - CSM management server
   - GPFS server
   - LoadLeveler Central Scheduler
   - Other management servers that are part of your cluster

2. Create an .rhosts file:

   ```
   vi /cfmroot/root/.rhosts
   ```

3. Put your CSM, LoadLeveler Central Scheduler, and GPFS primary and secondary servers in the file.

Example 2-4 shows how your .rhosts file should look.

Example 2-4 .rhosts file

```
csmserv
gpfs1
gpfs2
gridsch
csmservp
gpfs1p
gpfs2p
gridschp
```

4. Save the file.
5. Change the file protection for the file:

   ```
   chmod 600 /cfmroot/root/.rhosts
   ```

2.3.18 Creating a shared /etc/passwd file for all nodes

Create central /etc/passwd and /etc/shadow files so that all user IDs and passwords are common throughout the cluster by renaming the soft link to /etc/passwd._EmptyGroup and /etc/shadow._EmptyGroup in the /cfmroot/etc directory. Any updates to the CSM management server /etc/passwd file can then be propagated to the compute nodes:

```
mv /cfmroot/etc/passwd._EmptyGroup /cfmroot/etc/passwd
mv /cfmroot/etc/shadow._EmptyGroup /cfmroot/etc/shadow
```

2.3.19 Creating a shared /etc/groups file for all nodes

Create a central /etc/group file so that all groups are common throughout the cluster by renaming the soft link to /etc/groups._EmptyGroup in the /cfmroot/etc directory. Any updates to the CSM management server /etc/groups file can then be propagated to the compute nodes:

```
mv /cfmroot/etc/groups._EmptyGroup /cfmroot/etc/group
```

2.3.20 Creating a cluster-wide shared file system for shared files

Certain components of this environment are installed on a cluster-wide, NFS-mounted file system. When a node is built with an installation node, the file system and mount point that are created are installed automatically when CSM reboots the node after installation.

Create a file in the /csminstall/csm/scripts/installpostreboot directory called mkshare as follows:

1. Issue:

   ```
   vi /csminstall/csm/scripts/installpostreboot/mkshare
   ```

2. Add the lines in Example 2-5.

Example 2-5 Lines to add to mkshare

```
#!/bin/bash
# add mount point for shared cluster directory
if [ ! -d "/dist" ]; then
mkdir /dist
fi
if ! grep "csmservp:/dist" /etc/fstab 1> /dev/null; then
   echo "csmservp:/dist /dist        nfs     defaults        0 0" >> /etc/fstab
fi
```

3. Make the script executable by changing its protection:

   ```
   chmod 755 mkshare
   ```

2.3.21 Creating the LoadLeveler account and shared file system for CSM managed nodes

To create the LoadLeveler account and shared file system for CSM managed nodes, follow these steps:

1. Create a file in the /csminstall/csm/scripts/installpostreboot directory called mkloadl:

   ```
   vi /csminstall/csm/scripts/installpostreboot/mkloadl
   ```

2. Insert the script in Example 2-6 in your script file.

Example 2-6 Script for mkloadl

```
#!/bin/bash
# create loadl account if it does not exist
groupadd -g 1001 loadl
useradd -c "LoadLeveler User" -d /home/loadl -s /bin/bash -u 1001 -g 1001 -m loadl
# add /home/loadl if it does not exist
if [ ! -d "/home/loadl" ]; then
    mkdir /var/loadl
        mkdir /home/loadl
        chown loadl /home/loadl
    chmod 777 /var/loadl
fi
# add mount point for shared cluster directory (change gridschp to your own)
# loadleveler central manager
if ! grep "gridschp:/home/loadl" /etc/fstab 1> /dev/null; then
   echo "gridschp:/home/loadl /home/loadl        nfs     defaults        0 0" >>
/etc/fstab
fi
mount -a
```

3. Save the file.
4. Make the script executable by changing its protection:

```
chmod 755 mkloadl
```

2.3.22 Creating the script for the ApplicationWeb shared file system for CSM managed nodes

If you are installing ApplicationWeb, follow these steps:

1. Create a file in the /csminstall/csm/scripts/installpostreboot directory called mkawshare:

```
vi /csminstall/csm/scripts/installpostreboot/mkawshare
```

2. Insert the script in Example 2-7 in your script file.

Example 2-7 Script for mkawshare

```
#!/bin/bash
# add /opt/applicationwebruntime if it does not exist
if [ ! -d "/opt/applicationwebruntime" ]; then
        mkdir /opt/applicationwebruntime
    chmod 777 /opt/applicationwebruntime
fi
# add mount point for shared applicationweb directory (change gridschp to your
own)
# applicationweb central manager
if ! grep "gridschp:/opt/applicationwebruntime" /etc/fstab 1> /dev/null; then
  echo "gridschp:/opt/applicationwebruntime /opt/applicationwebruntime      nfs
defaults        0 0" >> /etc/fstab
fi
mount -a
```

3. Save the file.
4. Make the script executable by changing its protection:

```
chmod 755 mkawshare
```

2.3.23 CSM updates

You use CSM to install all the RPMs that are needed for the product portfolio that is part of the Optimized Analytic Infrastructure. Because you are installing Red Hat Enterprise Linux 3.0 WS Update 4, you must place certain RPMs in the installation and update directories. The installation directories contain any base RPMs and the updates directory contains update RPMs (such as those used to upgrade GPFS). To update CSM, follow these instructions:

1. Copy the following GPFS RPMs to /csminstall/Linux/RedHatEL-WS/3/i386/install:
   - gpfs.base-2.3.0-3.i386.rpm
   - gpfs.gpl-2.3.0-3.noarch.rpm
   - gpfs.docs-2.3.0-3.noarch.rpm
   - gpfs.msg.en_US-2.3.0-3.noarch.rpm
2. Copy the following j2sdk RPM to /csminstall/Linux/RedHatEL-WS/3/i386/install:
   - j2sdk-1_4_2_08-linux-i586.rpm
3. Copy the following LoadLeveler RPM to /csminstall/Linux/RedHatEL-WS/3/i386/install:
   - LoadL-full-license-RH3-X86-3.2.1.0-0.i386.rpm

2.3.24 Changing the kickstart file

You must make modifications to the Red Hat kickstart file to change characteristics of the installed operating system, which is installed with CSM. To change the kickstart file, follow these instructions:

1. Make a copy of the kickstart file:

   ```
   cp /opt/csm/install/kscfg.tmpl.RedHatEL-WS3
   /opt/csm/install/kscfg.tmpl.RedHatEL-WS3.orig
   ```

2. We used a flat file system versus a partitioned system. We modified the kickstart file as shown in Example 2-8.

Example 2-8 Modifications to the kickstart file

```
#
# Disk partition table. Customize it to fit your needs.
# /boot is strongly recommended
#
autopart
```

3. Find the package section so that you can add the following packages to support GPFS and building of the portability layer (add these after the last entry on the line that starts with "@"):

   ```
   @ editors
   @ base-x
   @ development-tools
   @ x-software-development
   @ compat-arch-development
   @ kernel-development
   ```

2.3.25 Running the csmsetupks command

At this point, all nodes should have been defined in the CSM database by the **definenode** command. The nodes are also in PreManaged status (issue a **monitorinstall** command to check this). The next step in the process of building the Optimized Analytic Infrastructure is to run the **csmsetupks** command.

The **csmsetupks** command copies the operating system RPMs to an installation directory, collects MAC addresses from the compute nodes, and configures the DHCP environment on the CSM management server. It also configures the provisioning infrastructure so that the **installnode** command operates properly. To run the **csmsetupks** command, have your four Red Hat Enterprise Linux CDs ready.

Start by issuing:

```
csmsetupks -P
```

This runs the setupks process for all nodes. You can also issue:

```
csmsetupks -P -n node
```

Note: *node* is one of the CSM managed nodes to be installed.

Instead of using the four installation CDs, you can copy the ISO formatted files to the /dist shared file system as follows:

1. Create mount points for the four ISO images on your CSM management server:

   ```
   mkdir -p /tmp/disc1
   mkdir -p /tmp/disc2
   mkdir -p /tmp/disc3
   mkdir -p /tmp/disc4
   ```

2. Copy all four ISO formatted distribution files to the /tmp directory on your CSM management server.

3. Mount each ISO file as a loop device so that each can be accessed as a file system:

   ```
   mount -o loop /tmp/RHEL3-U4-re1216.0-i386-WS-disc1-ftp.iso  /tmp/disc1
   mount -o loop /tmp/RHEL3-U4-re1216.0-i386-WS-disc2-ftp.iso  /tmp/disc2
   mount -o loop /tmp/RHEL3-U4-re1216.0-i386-WS-disc3-ftp.iso  /tmp/disc3
   mount -o loop /tmp/RHEL3-U4-re1216.0-i386-WS-disc4-ftp.iso  /tmp/disc4
   ```

4. Invoke the **csmsetupks** command with the **-p** switch as shown to use the ISO mounted disc images instead of having to use the physical CD media:

   ```
   csmsetupks -a -p
   /dist/tmp/disc1:/dist/tmp/disc2:/dist/tmp/disc3:/dist/tmp/disc4
   ```

   Your output should look like the sample shown in Example 2-9.

Example 2-9 Output of csmsetupks

```
Copying Red Hat Enterprise Linux WS 3 QU4 disk 1.
100% complete
Copying Red Hat Enterprise Linux WS 3 QU4 disk 2.
100% complete
Copying Red Hat Enterprise Linux WS 3 QU4 disk 3.
100% complete
Copying Red Hat Enterprise Linux WS 3 QU4 disk 4.
100% complete
Installing LINUX OS PRE REQUISITE RPMs.
Installing OPEN SOURCE PRE REQUISITE RPMs.
Securing file permissions in the /tftpboot directory and all subdirectories...

Running getmacs for nodes node1p.pbm.ihost.com
Acquiring adapter information using dsh.
Attempting to use dsh to collect MAC addresses.

# Name::Adapter Type::Adapter Name::MAC Address::Location Code::Adapter
Speed::Adapter Duplex::Install Server::Adapter Gateway::Ping Status::Machine Type

<You will see all your MAC address info here for each system>

Setting up Kickstart.
Adding nodes to /etc/dhcpd.conf file for Kickstart install:
node1p.pbm.ihost.com,node2p.pbm.ihost.com,node3p.pbm.ihost.com,node4p.pbm.ihost.co
m,node5p.pbm.ihost.com,node6p.pbm.ihost.com,node7p.pbm.ihost.com
```

Note: While inserting and copying CDs, you might receive a message stating that CSM does not recognize the correct distribution media. Make sure that you have inserted the correct distribution media. If you are sure that you have the correct media, you can press Control+F to force the mounting of the distribution CD.

5. If you add nodes to your CSM cluster, you do not have to copy the Red Hat distributions again. You can run the **csmsetupks** command with the **-x** switch. This adds the node without copying the distribution files.

   For all nodes, issue:

   ```
   csmsetupks -P -x
   ```

   For a single node, issue:

   ```
   csmsetupks -n node -x
   ```

   Note: *node* is one of the CSM managed nodes to be installed.

2.3.26 Adding an additional Ethernet device for a managed host

If your CSM management server has multiple Ethernet interfaces, the kickstart installation only installs a single interface (eth0 or eth1). To create the additional network interface, the script must first be copied from the /csminstall/csm/scripts directory to the /csminstall/csm/scripts/installprereboot directory so that it runs during the node installation:

```
cp /csminstall/csm/scripts/adapter_config_Linux
/csminstall/csm/scripts/installprereboot/adapter_config_Linux
```

Customization scripts that are put in this directory are run before the first reboot of the system. The customization script can be renamed so that it is targeted to a single node or group of nodes (for example, adapter_config_Linux._LinuxNodes). See the *CSM Administration Guide* for more information about how to use the CSM support for customization scripts.

The information that the script needs to configure the adapters must be provided in the form of a stanza file. This stanza file must be copied to the /csminstall/csm/scripts/data directory so that the script has access to it during the installation. By default, the customization script looks for the file named Linux_adapter_stanza_file. If another file name is used, edit the script to look for the correct name.

Because all our devices were the same, we used the default clause as a set of global definitions that apply to all devices to reduce the number of definitions that we had to make.

For more information, see the file /csminstall/csm/scripts/adapter_config_Linux.README on your CSM management server.

After you have created the stanza file, copy it to:

/csminstall/csm/scripts/data/Linux_adapter_stanza_file

The adapter_config_Linux customization script is run automatically during the installation of the node. Any error messages that are produced are saved in the /var/opt/csm/install.log file on the node (Example 2-10).

Example 2-10 Node customization

```
node1p :
# Linux RedHat adapter information
machine_type=secondary
network_type=eth
gateway=192.168.10.254
interface_name=eth1
BOOTPROTO=static
device=eth1
```

```
StartMode=onboot
IPADDR=192.168.10.5
NETMASK=255.255.255.0
ONBOOT=yes
TYPE=Ethernet
```

Repeat the process described in this section for every CSM managed node in your cluster.

2.4 Installing Scali MPI Connect

Scali MPI Connect is an MPI implementation that is fully compliant with the MPI 1.2 specification for message passing.

The Optimized Analytic Infrastructure implements Scali MPI Connect as follows:

- A central licensing model is deployed. The central License Manager is installed on the CSM management server.
- For clients, MPI is implemented with TCP/IP in a private network, so you might have to change the configuration to support other network types such as Myrinet or Ifiniband.

The Scali MPI Connect version is specific to Red Hat 3.0 Enterprise Linux Update 4. After obtaining the package from Scali, you must extract the archive into the /tmp directory. Assuming that you have mounted the CD with the Scali MPI Connect archive on /mnt/cdrom, use these commands:

```
cp /mnt/cdrom/Scali_MPI_Connect_4_4_0_rhel3_i386.tar.gz /tmp
cd tmp
gzip -d Scali_MPI_Connect_4_4_0_rhel3_i386.tar.gz
tar -xvf Scali_MPI_Connect_4_4_0_rhel3_i386.tar
```

This extracts the Scali MPI connect files to a directory called /tmp/Scali_MPI_Connect_4_4_0.

2.4.1 Installing the Scali MPI Connect License Manager

You install the Scali MPI Connect License Manager on your CSM management server. Before proceeding, make sure that you have extracted the files and that you have a proper license file that has been obtained from Scali. Then, follow these steps:

1. Log on to the CSM management server as root if you have not already done so.
2. Copy your license file to the /tmp/ Scali_MPI_Connect_4_4_0 directory.
3. Issue:

   ```
   cd /tmp/Scali_MPI_Connect_4_4_0
   ```

4. Issue:

   ```
   ./smcinstall -u /tmp/Scali_MPI_Connect_4_4_0/license.dat
   ```

5. Accept the licensing agreement to continue the installation.

2.4.2 Copying the scali.sh profile to the CFM directory

CFM is used to distribute the scali.sh shell script to all compute nodes. This shell script is used to add the PATH and man pages to your login profile:

```
cd /tmp/Scali_MPI_Connect_4_4_0
mkdir -p /cfmroot/etc/profile.d
cp scali.sh  /cfmroot/etc/profile.d
```

This copies the scali.sh file, which is installed with the Scali MPI Connect package to the CFM distribution share:

```
chmod 755 /cfmroot/etc/profile.d/scali.sh
```

2.4.3 Using SMS to copy the Scali MPI Connect distribution RPMs

Software Maintenance System (SMS) is used to copy the Scali MPI Connect client RPMs to the compute nodes. The following RPMs must be copied to the /csminstall/Linux/RedHatEL-WS/3/i386/install directory:

- scalm
- scampi
- scaenv
- scampitst

To copy Scali MPI Connect RPMs, issue the following commands:

```
cp /tmp/Scali_MPI_Connect_4_4_0/repository/scali/rhel3/i386/
scalm-9.1.0-3.rhel3.i386.rpm /csminstall/Linux/RedHatEL-WS/3/i386/install
cp /tmp/Scali_MPI_Connect_4_4_0/repository/scali/generic/noarch/
scaenv-1.5.6-1.generic.noarch.rpm /csminstall/Linux/RedHatEL-WS/3/i386/install
cp /tmp/Scali_MPI_Connect_4_4_0/repository/scali/rhel3/i386/
scampi-3.5.0-3.rhel3.i386.rpm /csminstall/Linux/RedHatEL-WS/3/i386/install
cp /tmp/Scali_MPI_Connect_4_4_0/repository/scali/rhel3/i386/
scampitst-1.11.2-0.rhel3.i386.rpm /csminstall/Linux/RedHatEL-WS/3/i386/install
```

2.4.4 Creating a Scali MPI Connect License Manager file

A shared License Manager model for Scali MPI Connect is used; therefore, you create a configuration file that points the compute nodes to the License Manager as follows:

1. Put this in the CFM directory structure so that it is distributed to the compute nodes:

   ```
   mkdir -p /cfmroot/opt/scali/etc/
   ```

2. Create the scalm.conf file:

   ```
   vi /cfmroot/opt/scali/etc/scalm.conf
   ```

3. Enter the following line, where “csmserv” is your Scali MPI Connect License Manager:

   ```
   scalm_net_server="csmserv"
   ```

4. Finally, push the Scali MPI Connect configuration file to the nodes by running the **cfmupdatenode** command:

   ```
   cfmupdatenode -a
   ```

Otherwise, no other action is required, because CFM is run automatically during normal full installation of the CSM compute nodes.

2.5 Installing the grid/job scheduling server

The grid/job scheduling server is used for load balancing and scheduling work for the Optimized Analytic Infrastructure compute nodes. This server is used to host middleware components, such as LoadLeveler.

2.5.1 Installing base Linux operating system on the grid/job scheduling server

Install Red Hat Enterprise Linux (RHEL) V3.0 Advanced Server Update 4 on the grid/job scheduling server.

2.5.2 Mounting the shared file system /dist on CSM management server

You must create an entry in the server /etc/fstab file to point to the NFS file share located on csmserv, which contains the common file system for all servers in the Optimized Analytic Infrastructure environment (this share was created when the CSM management server was installed):

1. Edit the file system table:

   ```
   vi /etc/fstab
   ```

2. Insert the following line at the end of the file and then save the file:

   ```
   csmserv:/dist /dist        nfs     defaults       ; 0 0
   ```

3. Mount the file system by running the **mount** command:

   ```
   mount -a
   ```

2.5.3 Setting up RSH

To execute commands on the grid/job scheduling server from the CSM management server, set up RSH on this server. To install and configure RSH:

1. Issue:

   ```
   chkconfig --add rsh
   chkconfig  rsh on
   ```

2. Create a /root/.rhosts file containing management and private network host names for the servers in the Optimized Analytic Infrastructure configuration. We copied the .rhosts file that was created in the CSM management server installation section and placed in the /cfmroot file system:

   ```
   scp root@csmserv:/cfmroom/root/.rhosts  /root/.rhosts
   ```

3. Change file owner and protection for the file:

   ```
   chmod 600 /root/.rhosts
   ```

2.5.4 Configuring time synchronization

Because all nodes have work scheduled by the grid/job scheduling server, it is important that all nodes have the same system time. See /opt/csm/samples/docs/ntp.README on your CSM management server for details about how to install and configure NTP on your CSM management server.

2.5.5 Renaming Java and Javac shell scripts

Red Hat Enterprise Linux includes two script files as placeholders for Java and Javac. You must rename these placeholder scripts so that the installed Java Development Kit (JDK) can be referenced without manipulating the PATH statement as follows.

Log in to the grid/job scheduling server as root and issue:

```
mv /usr/bin/java /usr/bin/java.old
mv /usr/bin/javac /usr/bin/javac.old
```

2.5.6 Installing the JDK for the grid/job scheduling server

To install the JDK for the grid/job scheduling server:

1. Log in to the grid/job scheduling server as root, if you have not already done so.
2. Copy the JDK RPM to the /tmp directory:

   ```
   rpm -ivh /tmp/ j2sdk-1_4_2-08-linux-i586.rpm
   ```

2.5.7 Setting PATH for JDK for the grid/job scheduling server

To create a profile that sets the path for the JDK on the server:

1. Log in to the grid/job scheduling server as root.
2. Create /etc/profile.d/jdk.sh:

   ```
   JAVA_HOME=/usr/java/j2sdk1.4.2_08/
   export PATH
   PATH=$PATH:$JAVA_HOME/bin
   export PATH
   ```

   In this case, we used the JDK V1.4.2_08. If you use a different JDK, change the JAVA_HOME path in your script as follows:

   ```
   chmod 755 /etc/profile.d/jdk.sh
   ```

3. Execute the JDK profile script by running:

   ```
   source /etc/profile.d/jdk.sh
   ```

2.6 Installing the LoadLeveler Central Scheduler

The LoadLeveler Central Scheduler is responsible for scheduling and dispatching work to the compute nodes.

For the Optimized Analytic Infrastructure, LoadLeveler is installed in the following servers:

- Central Scheduler: gridschp
- Compute nodes: node1p-4p
- Test client: node6p, which is installed as a submit-only node

2.6.1 Installing the LoadLeveler RPMs

To copy LoadLeveler RPMs and Java Runtime Environment (JRE) 1.4.2, which is shipped with LoadLeveler to the /dist/LoadL directory, follow these steps:

1. Create the LoadLeveler distribution directory on the /dist file system:

   ```
   mkdir -p /dist/LoadL
   ```

2. Copy the LoadLeveler distribution CD to the /dist/LoadL directory.
3. Install the LoadLeveler License RPM by running:

   ```
   rpm -ivh /dist/LoadL/LoadL-full-license-RH3-X86-3.2.1.0-0.i386.rpm
   ```

4. Run the install_ll script by issuing the following commands:

   ```
   cd /opt/ibmll/LoadL/sbin/
   ./install_ll -y -d /dist/LoadL/
   ```

5. Your output should look like the output shown in Example 2-11.

Example 2-11 Install_ll script output

```
./install_ll -y -d /dist/LoadL/
Installing /dist/LoadL//IBMJava2-JRE-1.4.2-0.0.i386.rpm at /opt/ibmll
Preparing...                ########################################### [100%]
   1:IBMJava2-JRE           ########################################### [100%]
Running the IBM License Acceptance Program...
Installing the RPM: /tmp/LoadL-full-RH3-X86-3.2.1.0-0.i386.rpm
This step could take a few minutes, please be patient
Preparing...                ########################################### [100%]
        package LoadL-full-RH3-X86-3.2.1.0-0 is already installed
```

2.6.2 Creating the LoadLeveler user account

LoadLeveler uses a common user account and shared home directory between all the nodes. Create the account in the grid/job scheduling server.

Add the loadl group:

```
groupadd -g 1001 loadl
```

Add the loadl user account:

```
useradd -c "LoadLeveler User" -d /home/loadl -s /bin/bash -u 1001 -g 1001 -m loadl
```

2.6.3 Creating local directories for Central Scheduler

To create the local directories that the LoadLeveler Central Scheduler needs and to change the owner so that loadl can write to the /var file system, follow these steps:

1. Create the local directories:

   ```
   mkdir /var/loadl
   ```

2. Change the owner so that loadl can write to the /var file system:

   ```
   chown loadl /var/loadl
   ```

3. If you are installing ApplicationWeb, create the following required directory:

   ```
   mkdir /opt/applicationwebruntime
   ```

2.6.4 Exporting directories for compute nodes

To export directories for compute nodes, follow these instructions:

1. Enable NFS adding and enable the service:

   ```
   chkconfig --add nfs
   chkconfig nfs on
   ```

2. Create an entry in the /etc/exportfs configuration to define the /dist file system to other nodes:

   ```
   vi /etc/exportfs
   ```

3. Add the following line to the file:

   ```
   /home/loadl   *.ihost.com(rw,no_root_squash,insecure,sync)
   ```

4. If you are installing ApplicationWeb, also add the following line:

   ```
   /opt/applicationwebruntime   *.ihost.com(rw,no_root_squash,insecure,sync)
   ```

5. Change *.ihost.com to your current DNS domain or machine list (see the NFS documents for more examples). After saving the /etc/exportfs file, you must restart NFS to export the /home/loadl file system that you just added:

   ```
   service nfsd restart
   ```

6. To verify that the NFS export is in effect, issue:

   ```
   exportfs
   ```

   The **exportfs** command displays all file systems that have been exported by this server. The following output is an example of the **exportfs** command output:

   ```
   /opt/applicationwebruntime
                  *.pbm.ihost.com
   /home/loadl    *.pbm.ihost.com
   ```

2.6.5 Updating the LoadLeveler PATH variable and man pages

The loadl account profiles must be updated so that the LoadLeveler binaries and man pages can be accessed easily as follows:

```
su - loadl
vi $HOME/.bash_profile
```

Edit your PATH statement in your $HOME/.bash_profile so that it includes the LoadLeveler directories and then save the file. Your file should look similar to this:

```
PATH=$PATH:$HOME/bin:/opt/ibmll/full/LoadL/bin:/home/loadl/bin:
```

After the PATH statement, you must add the following line to add the MANPATH:

```
MANPATH=$MANPATH:/opt/ibmll/full/LoadL/man
```

2.6.6 Initializing LoadLeveler

To configure the LoadLeveler Central Scheduler and initialize it, follow these instructions:

1. Make sure that you are running under the user loadl to obtain access permissions:

   ```
   su - loadl
   ```

2. Run the LoadLeveler initialization script:

```
cd /opt/ibmll/LoadL/full/bin
llinit -local /var/loadl -release /opt/ibmll/LoadL/full -cm gridschp
```

   In this example, we used the gridschp (grid scheduler private network) host name as a Central Scheduler definition.

3. Create a /home/loadl/.rhosts file with management and private network host names for the servers in the Optimized Analytic Infrastructure configuration. You can copy the .rhosts file that was created in the CSM management server installation section and placed in the /cfrmroot file system:

```
scp root@csmservp:/cfmroom/root/.rhosts /home/loadl/.rhosts
```

4. Change file owner and protection for the file:

```
chmod 600 /home/loadl/.rhosts
chown loadl /home/loadl/.rhosts
```

5. Edit the /home/loadl/loadl_Admin configuration file:

```
vi /home/loadl/loadl_Admin
```

6. Find the Machine Stanzas section in the file and edit the stanza files as shown in Example 2-12.

Example 2-12 Stanza file edits

```
########################################################################
# MACHINE STANZAS:
# These are the machine stanzas; the first machine is defined as
# the central manager.  mach1:, mach2:, etc. are machine name labels -
# revise these placeholder labels with the names of the machines in the
# pool, and specify any schedd_host and submit_only keywords and values
# (true or false), if required.
########################################################################

gridschp: type = machine          central_manager = true
                                  schedd_host = true
                                  alias = gridsch
node1p: type = machine
node2p: type = machine
node3p: type = machine
node4p: type = machine
node6p: type = machine            schedd_host = false
                                  submit_only = true
```

   In our example, we used gridsch as the Central Scheduler name and gridschp as the alias name for the Central Scheduler. This is important if you have a management and private network, which you do in the Optimized Analytic Infrastructure configuration.

   Add all of the private network host names after the Central Scheduler definitions. In our example, we had only four hosts, but your configuration might have more. Save the file.

7. Create all the local configuration files to support your central manager and compute nodes. These are stored in the /home/loadl/local directory. Create this directory as follows:

```
mkdir /home/loadl/local
```

8. To tell LoadLeveler where the configuration files are stored and what their name format is, edit the /home/loadl/LoadL_config file:

```
vi /home/loadl/LoadL_config
```

Search for the line that contains the LOCAL_CONFIG directive. It should look like this:

```
LOCAL_CONFIG   = $(tilde)/LoadL_config.local
```

Change it as follows and then save the file.

```
LOCAL_CONFIG    = $(tilde)/local/$(HOST).LoadL_config.local
```

9. Make two copies of the /home/loadl/LoadL_config.local file to use as the configuration files for the Central Scheduler machine and the compute nodes:

```
cp /home/loadl/LoadL_config.local  /home/loadl/local/gridsch.LoadL_config.local
cp /home/loadl/LoadL_config.local
/home/loadl/local/computenode.LoadL_config.local
```

10. Edit the local configuration file for your Central Scheduler machine. The important changes are shown in Example 2-13. Then save the file.

Example 2-13 Important edits for the local configuration file

```
MAX_STARTERS    = 2             # Set to the # of CPU's
STARTD_RUNS_HERE = false  # cannot run work on this node
SCHEDD_RUNS_HERE = true  # queues run on this machine
SCHEDD_HOST = true        # can submit jobs from here
```

Edit the file:

```
vi  /home/loadl/local/gridsch.LoadL_config.local
```

Your file should then resemble Example 2-14.

Example 2-14 Edited local configuration file

```
#
# file: gridsch _config.local
# Statements included here override on a PER MACHINE BASIS the statements in
# LoadL_config and may be used to tailor each machine individually.
# See samples in your RELEASEDIR/samples directory
#
START_DAEMONS   = TRUE
MAX_STARTERS    = 2             # Set to the # of CPU's
STARTD_RUNS_HERE = false  # cannot run work on this node
SCHEDD_RUNS_HERE = true  # queues run on this machine
SCHEDD_HOST = true        # can submit jobs from here

# Alternative ways of specifying the CLASS keyword
# the following is the old-style specification
#
#CLASS = { "small" "small" "small" "small" "small" "small" "small" "small" "medi
cum" "medium" "medium" "medium" "medium" "large" "large" }
# while the following is a newer, more concise specification
CLASS = small(8) medium(5) large(2)
```

11. Edit the local configuration file for your compute/worker nodes. The important changes are:

```
MAX_STARTERS    = 2             # Set to the # of CPU's
vi  /home/loadl/local/ computenode.LoadL_config.local
```

Your file should resemble Example 2-15.

Example 2-15 Edited configuration file for compute/worker nodes

```
#
# file: computenode.LoadL_config.local
# Statements included here override on a PER MACHINE BASIS the statements in
# LoadL_config and may be used to tailor each machine individually.
# See samples in your RELEASEDIR/samples directory
#
START_DAEMONS   = TRUE
MAX_STARTERS    = 2             # Set to the # of CPU's

# Alternative ways of specifying the CLASS keyword
# the following is the old-style specification
#
#CLASS = { "small" "small" "small" "small" "small" "small" "small" "small" "medi
cum" "medium" "medium" "medium" "medium" "large" "large" }
# while the following is a newer, more concise specification
CLASS = small(8) medium(5) large(2)
```

12. To minimize the number of configuration files for the compute nodes, create a soft link for each compute node to the /home/load/local/computenode.LoadL_config.local file:

    ```
    cd /home/loadl/local
    ln computenode.LoadL_config.local node1p.LoadL_config.local
    ln computenode.LoadL_config.local node2p.LoadL_config.local
    ln computenode.LoadL_config.local node3p.LoadL_config.local
    ln computenode.LoadL_config.local node4p.LoadL_config.local
    ```

    In this example, we have only four nodes; your configuration might have more.

2.6.7 Operating the LoadLeveler Central Scheduler

This section describes how to operate the LoadLeveler Central Scheduler.

Starting the LoadLeveler Central Scheduler

To start the LoadLeveler Central Scheduler, issue:

```
llctl -h gridschp start
```

Where *gridschp* is the private network host name of the server.

Stopping the LoadLeveler Central Scheduler

To stop the LoadLeveler Central Scheduler only, issue:

```
llctl -h gridschp stop
```

2.6.8 The LoadLeveler GUI

The LoadLeveler GUI, xload, provides an alternative to the command line interface for LoadLeveler. To use xload, the following steps are necessary for customizing the xload interface:

1. Issue:

   ```
   cp /opt/ibmll/LoadL/full/lib/Xloadl /usr/lib/X11/app-defaults
   ```

2. Issue:

```
chmod 644 /usr/lib/X11/app-defaults/Xloadl
```

2.7 Installing the ApplicationWeb server

The ApplicationWeb server is installed in the grid/job scheduling server as follows:

1. Log in to the grid/job scheduling server as root if you have not already done so.
2. Copy the ApplicationWeb RPMs from the source media to the /tmp directory.
3. Install the ApplicationWeb RPM by running the **rpm** command as shown in Example 2-16.

Example 2-16 RPM command output

```
rpm -ivh /dist/applicationwebruntime/applicationwebruntime-1.0-5.i386.rpm
Preparing...                ########################################### [100%]
   1:applicationwebruntime  ########################################### [100%]
1) Run the /opt/applicationwebruntime/bin/installmsh.sh script to
   set up your environment.  This will create the directory
   $HOME/mshhome.
2) Cd to your $HOME/mshhome directory and run the testXX.sh scripts.
3) Edit the testXX.sh scripts to see how the individual tests are run.
```

4. ApplicationWeb is configured to use a non-root account such as the loadl account created for LoadLeveler. For the Optimized Analytic Infrastructure, use the loadl account:

   ```
   su - loadl
   ```

5. Run the installmsh.sh script, which is installed as part of ApplicationWeb:

   ```
   cd /opt/applicationwebruntime/bin
   ./installmsh.sh
   ```

   Your output should resemble Example 2-17.

Example 2-17 Output of the installmsch.sh script

```
./installmsh.sh
To run ApplicationWeb, you must have
        1) the directory /opt/applicationwebruntime/bin in your PATH
        2) the directory /opt/applicationwebruntime/bin in your LD_LIBRARY_PATH
        3) The MSHHOME variable set to a directory containing the .mshusr file.
A sample shell script can be run using
"/opt/applicationwebruntime/bin/mshmain.sh", which will start the ApplicationWeb
interpreter.
The  "/opt/applicationwebruntime/bin/mshmain.sh" shell script will set the
environment variables listed above before invoking the interpreter.
See the ApplicationWeb README file for further instructions.
```

2.7.1 Creating the ApplicationWeb nodelist file

To support ApplicationWeb MPI testing, you must create a text file with all of the compute nodes in the cluster in the $HOME/mshhome/test directory of the loadl account as follows:

1. Log on to the grid/job scheduling server as user loadl.
2. In the loadl user home directory, create a file named nodelist that contains all of the compute nodes in the cluster:

   ```
   vi $HOME/mshhome/test/nodelist
   ```

3. Insert all of the node names into the file and then save it.

   Your output should resemble Example 2-18.

Example 2-18 Output of file with node names

```
node1p
node2p
node3p
node4p
```

If you are not planning to test LoadLeveler with ApplicationWeb and MPI, you do not need to perform this step.

2.7.2 Installing Scali MPI Connect in the ApplicationWeb server

ApplicationWeb supports low-latency interconnection between its components. To support MPI for IBM ApplicationWeb, you install Scali MPI Connect on the server as follows:

1. Log on to the grid/job scheduling server as root if you have not already done so.
2. Issue:

   ```
   cp /mnt/cdrom/Scali_MPI_Connect_4_4_0_rhel3_i386.tar.gz /tmp
   ```

3. Issue:

   ```
   cd tmp
   ```

4. Issue:

   ```
   gzip -d Scali_MPI_Connect_4_4_0_rhel3_i386.tar.gz
   ```

5. Extract the Scali MPI Connect files to a directory called /tmp/Scali_MPI_Connect_4_4_0:

   ```
   tar -xvf Scali_MPI_Connect_4_4_0_rhel3_i386.tar
   ```

6. Install the four required Scali MPI Connect RPMs from the /tmp/Scali_MPI_Connect_4_4_0/repository/scali/rhel3/i386 directory as shown in Example 2-19.

Example 2-19 Installing the required Scali MPI Connect RPMs

```
rpm -ivh /tmp/Scali_MPI_Connect_4_4_0/repository/scali/rhel3/i386/
scalm-9.1.0-3.rhel3.i386.rpm
rpm -ivh /tmp/Scali_MPI_Connect_4_4_0/repository/scali/generic/noarch/
scaenv-1.5.6-1.generic.noarch.rpm
rpm -ivh /tmp/Scali_MPI_Connect_4_4_0/repository/scali/rhel3/i386/
scampi-3.5.0-3.rhel3.i386.rpm
rpm -ivh /tmp/Scali_MPI_Connect_4_4_0/repository/scali/rhel3/i386/
scampitst-1.11.2-0.rhel3.i386.rpm
```

See 2.4, “Installing Scali MPI Connect” on page 29 for more details about installing Scali MPI Connect in your environment.

2.7.3 Rebuilding the ApplicationWeb binaries for other MPI implementations

Depending on your implementation of MPI, you might need to rebuild the ApplicationWeb binaries to comply with your MPI implementation. To rebuild your libraries, follow these instructions:

1. Log on to the grid/job scheduling server as root if you have not already done so.
2. Change directory to the /opt/applicationwebruntime/build directory:

   ```
   cd /opt/applicationwebruntime/build directory
   ```

3. Clean the make environment by running the **`make clean`** command:

   ```
   make clean
   ```

   Your output should have no errors.
4. Create the build dependencies by running the **`make depends`** command:

   ```
   make depends
   ```

   Your output should have no errors.
5. Finally, rebuild the executables by running the **`make all`** command:

   ```
   make all
   ```

   Your output should contain no errors.

2.7.4 Testing the ApplicationWeb installation

After you install the ApplicationWeb server, test it by running the test programs that are provided with the installation package. These are the ApplicationWeb test server program, the fast test program, the MPI test program (without LoadLeveler), and the MPI test program (with LoadLeveler). To use any of these programs, log on to the grid/job scheduling server as loadl.

ApplicationWeb test server program

To run this program, follow these steps:

1. Change the directory to the ApplicationWeb test program directory:

   ```
   cd /home/loadl/mshhome/test
   ```

2. Run the ApplicationWeb test server program, testServer.sh:

   ```
   ./testServer.sh
   ```

   Your output should resemble Example 2-20.

Example 2-20 Output of testServer.sh

```
./testServer.sh
[testServer.sh] Using APPWEB_RT = /opt/applicationwebruntime
[testServer.sh] Compiling testServer.java
[testServer.sh] Executing testServer.java
[testServer.java] Connected to node1p.pbm.ihost.com:1192
[testServer.java] Press enter to continue

type:Double Precision
ndimensions:1
```

```
dimensions:  100
        0.0000000000        1.0000000000        1.4142135623        1.7320508075
        2.0000000000        2.2360679774        2.4494897427        2.6457513110
        2.8284271247        3.0000000000        3.1622776601        3.3166247903
        3.4641016151        3.6055512754        3.7416573867        3.8729833462
        4.0000000000        4.1231056256        4.2426406871        4.3588989435
        4.4721359549        4.5825756949        4.6904157598        4.7958315233
        4.8989794855        5.0000000000        5.0990195135        5.1961524227
        5.2915026221        5.3851648071        5.4772255750        5.5677643628
        5.6568542494        5.7445626465        5.8309518948        5.9160797830
        6.0000000000        6.0827625302        6.1644140029        6.2449979983
        6.3245553203        6.4031242374        6.4807406984        6.5574385243
        6.6332495807        6.7082039324        6.7823299831        6.8556546004
        6.9282032302        7.0000000000        7.0710678118        7.1414284285
        7.2111025509        7.2801098892        7.3484692283        7.4161984870
        7.4833147735        7.5498344352        7.6157731058        7.6811457478
        7.7459666924        7.8102496759        7.8740078740        7.9372539331
        8.0000000000        8.0622577482        8.1240384046        8.1853527718
        8.2462112512        8.3066238629        8.3666002653        8.4261497731
        8.4852813742        8.5440037453        8.6023252670        8.6602540378
        8.7177978870        8.7749643873        8.8317608663        8.8881944173
        8.9442719099        9.0000000000        9.0553851381        9.1104335791
        9.1651513899        9.2195444572        9.2736184954        9.3273790530
        9.3808315196        9.4339811320        9.4868329805        9.5393920141
        9.5916630466        9.6436507609        9.6953597148        9.7467943448
        9.7979589711        9.8488578017        9.8994949366        9.9498743710
[testServer.java] Press enter to continue

type:Double Precision
ndimensions:1
dimensions:  100
        0.0000000000        1.0000000000        1.4142135623        1.7320508075
        2.0000000000        2.2360679774        2.4494897427        2.6457513110
        2.8284271247        3.0000000000        3.1622776601        3.3166247903
        3.4641016151        3.6055512754        3.7416573867        3.8729833462
        4.0000000000        4.1231056256        4.2426406871        4.3588989435
        4.4721359549        4.5825756949        4.6904157598        4.7958315233
        4.8989794855        5.0000000000        5.0990195135        5.1961524227
        5.2915026221        5.3851648071        5.4772255750        5.5677643628
        5.6568542494        5.7445626465        5.8309518948        5.9160797830
        6.0000000000        6.0827625302        6.1644140029        6.2449979983
        6.3245553203        6.4031242374        6.4807406984        6.5574385243
        6.6332495807        6.7082039324        6.7823299831        6.8556546004
        6.9282032302        7.0000000000        7.0710678118        7.1414284285
        7.2111025509        7.2801098892        7.3484692283        7.4161984870
        7.4833147735        7.5498344352        7.6157731058        7.6811457478
        7.7459666924        7.8102496759        7.8740078740        7.9372539331
        8.0000000000        8.0622577482        8.1240384046        8.1853527718
        8.2462112512        8.3066238629        8.3666002653        8.4261497731
        8.4852813742        8.5440037453        8.6023252670        8.6602540378
        8.7177978870        8.7749643873        8.8317608663        8.8881944173
        8.9442719099        9.0000000000        9.0553851381        9.1104335791
        9.1651513899        9.2195444572        9.2736184954        9.3273790530
        9.3808315196        9.4339811320        9.4868329805        9.5393920141
        9.5916630466        9.6436507609        9.6953597148        9.7467943448
```

```
        9.7979589711        9.8488578017        9.8994949366        9.9498743710
[testServer.java] Press enter to continue
[testServer.java] Success!
```

Fast test program

To run the Fast test program, follow these instructions:

1. Change to the ApplicationWeb Fast test program directory:

   ```
   cd /home/loadl/mshhome/fast/test
   ```

2. Run the ApplicationWeb fast test program, runMain.sh:

   ```
   ./runMain.sh
   ```

   Your output should resemble Example 2-21.

Example 2-21 Output for runMain.sh

```
./runMain.sh
[runMain.sh] Running main program.
..................
Number of Runs:         10
Number of Total Paths:  320
Number of Factors:      7
Number of Instruments:  9

Instrument      vanilla         5.90074
Instrument      asian           0.00044842
Instrument      min             0.0155507
Instrument      discBond60              0.980199
Instrument      s1              0.00570081
Instrument      barrier         4.04626
Instrument      varianceCall            0.00897944
Instrument      asian1          0.000425143
Instrument      barrierRebate           4.91314
[runMain.sh] Done.
```

MPI test program (without LoadLeveler)

To run the MPI test program (without LoadLeveler), follow these instructions:

1. Change to the ApplicationWeb test program directory:

   ```
   cd /home/loadl/mshhome/test
   ```

2. Run the ApplicationWeb MPI test program, testMPI.sh:

   ```
   ./testMPI.sh
   ```

   Your output should resemble Example 2-22.

Example 2-22 Output for tstMPI.sh

```
./testMPI.sh
[testMPI.sh] Executing MshMPIClient.
[testMPI.sh] Executing MshMPIServer
Initializing connection from node1p.pbm.ihost.com...
name: result
  nd: 1
   n: 1
type: 2 (Double Precision) MSH_IMMEDIATE rw-rw-rw-
```

```
result[0:0]=    1.414213562373E+00
name: result
  nd: 1
   n: 1
type: 2 (Double Precision) MSH_IMMEDIATE rw-rw-rw-
result[0:0]=    1.414213562373E+00
name: result
  nd: 1
   n: 1
type: 2 (Double Precision) MSH_IMMEDIATE rw-rw-rw-
result[0:0]=    1.414213562373E+00
name: result
  nd: 1
   n: 1
type: 2 (Double Precision) MSH_IMMEDIATE rw-rw-rw-
result[0:0]=    1.414213562373E+00
[MshMPIClient] Results returned successfully from MPIALL; number of processes: 4
[MshMPIClient] Results returned successfully from MPIANY; number of requests: 100
[MshMPIClient] Result returned from MPIQUERY numProcs: numProcs = 4
[MshMPIClient] Server gethostname() returns node1p
[MshMPIClient] Server gethostname() returns node2p
[MshMPIClient] Server gethostname() returns node3p
[MshMPIClient] Server gethostname() returns node4p
[MshMPIClient] Success!
```

MPI test program (with LoadLeveler)

To run the MPI test program (with LoadLeveler), follow these instructions:

1. Change to the ApplicationWeb test program directory:

   ```
   cd /home/loadl/mshhome/test
   ```

2. Run the ApplicationWeb MPI test program, testLL.sh:

   ```
   ./testLL.sh
   ```

   Your output should resemble Example 2-23.

Example 2-23 Output of testLL.sh

```
[loadl test]$ ./testLL.sh
[testMPI.sh] Executing MshMPIClient.
[testMPI.sh] Submitting mpiServer.cmd
llsubmit: The job "gridschp.pbm.ihost.com.114" has been submitted.
[test]$ name: result
  nd: 1
   n: 1
type: 2 (Double Precision) MSH_IMMEDIATE rw-rw-rw-
result[0:0]=    1.414213562373E+00
name: result
  nd: 1
   n: 1
type: 2 (Double Precision) MSH_IMMEDIATE rw-rw-rw-
result[0:0]=    1.414213562373E+00
name: result
  nd: 1
   n: 1
type: 2 (Double Precision) MSH_IMMEDIATE rw-rw-rw-
```

```
result[0:0]=    1.414213562373E+00
name: result
  nd: 1
   n: 1
type: 2 (Double Precision) MSH_IMMEDIATE rw-rw-rw-
result[0:0]=    1.414213562373E+00
[MshMPIClient] Results returned successfully from MPIALL; number of processes: 4
[MshMPIClient] Results returned successfully from MPIANY; number of requests: 100
[MshMPIClient] Result returned from MPIQUERY numProcs: numProcs = 4
[MshMPIClient] Server gethostname() returns node1p.pbm.ihost.com
[MshMPIClient] Server gethostname() returns node2p.pbm.ihost.com
[MshMPIClient] Server gethostname() returns node3p.pbm.ihost.com
[MshMPIClient] Server gethostname() returns node4p.pbm.ihost.com
[MshMPIClient] Success!
```

3. Verify the LoadLeveler output from the testLL.sh shell script.
4. When you ran the testLL shell script, LoadLeveler created a queue entry to run the job. The output of the **`llsubmit`** command indicates a queue entry number. In this example, the job name is “gridschp.pbm.ihost.com.114”. An output file by the name of LLmshMPIServer.dcx08.114.0.log was created. View the output as follows:

   ```
   more LLmshMPIServer.dcx08.114.0.log
   ```

   Your output should resemble Example 2-24.

Example 2-24 Output forLLmshMPIServer.dcx08

```
# more LLmshMPIServer.dcx08.114.0.log
[mshMPIServer.cmd] Processor list: node1p.pbm.ihost.com node2p.pbm.ihost.com
node3p.pbm.ihost.com node4p.pbm.ihost.com
[mshMPIServer.cmd] Listener Host: gridschp.pbm.ihost.com
[mshMPIServer.cmd] Listener Port: 1195
[testMPI.sh] Executing MshMPIServer
[testMPI.sh] ... mpimon -working_directory /opt/applicationwebruntime/bin ./MshM
PIServer  --notify gridschp.pbm.ihost.com:1195 - node1p.pbm.ihost.com
node2p.pbm.ihost.com node3p.pbm.ihost.com node4p.pbm.ihost.com
```

2.8 Installing the GPFS servers

The Optimized Analytic Infrastructure configuration uses GPFS configured with NSD support. There are two GPFS servers, using a primary and secondary configuration. A GPFS nodeset configured with NSD network-attached servers provides access to disks and replication can be through one or two storage attached nodes.

In our configuration, we used two GPFS storage nodes, which provide access for all GPFS clients (compute nodes).

2.8.1 Installing the base Linux operating system in the GPFS server

Install Red Hat Enterprise Linux V3.0 Advanced Server Update 4 on the GPFS server.

Be sure to install the following packages during the Linux installation:

- Kernel Development
- X-Windows Development

2.8.2 Mounting the shared file system /dist file system located in the CSM management server

You must create an entry in the server /etc/fstab file to point to the NFS file share located on csmserv, which contains the common file system for all servers in the Optimized Analytic Infrastructure environment. This share was created when the CSM management server was installed. To create the entry, follow these instructions:

1. Edit the file system table:

   ```
   vi /etc/fstab
   ```

2. Insert the following line at the end of the file and then save it:

   ```
   csmserv:/dist /dist        nfs     defaults        0 0
   ```

3. Mount the file system by running the **mount** command:

   ```
   mount -a
   ```

2.8.3 Setting up RSH

To execute commands on the GPFS servers from the CSM management server, set up RSH on this server as follows:

1. Install and configure RSH:

   ```
   chkconfig --add rsh
   chkconfig  rsh on
   ```

2. Create a /root/.rhosts file with management and private network host names for the servers in the Optimized Analytic Infrastructure configuration. Copy the .rhosts file that was created during the CSM management server installation and that was placed in the /cfmroot file system:

   ```
   scp root@csmservp:/cfmroot/root/.rhosts  /root/.rhosts
   ```

3. Change the file owner and protection for the file:

   ```
   chmod 600 /root/.rhosts
   ```

2.8.4 Configuring time synchronization

Because all nodes have work scheduled by the GPFS server, it is important that all nodes have the same system time. See the /opt/csm/samples/docs/ntp.README file on your CSM management server for details about how to install and configure NTP in your server.

2.8.5 Installing the GPFS RPMs in the GPFS server

To install the GPFS RPMs in the GPFS:

1. Log on to the GPFS server.
2. Copy the GPFS RPMs from your distribution media to the /tmp directory.
3. Install the base GPFS RPMs:

   ```
   rpm -ivh /tmp/gpfs.base-2.3.0-3.i386.rpm
   rpm -ivh /tmp/gpfs.gpl-2.3.0-3.noarch.rpm
   rpm -ivh /tmp/gpfs.docs-2.3.0-3.noarch.rpm
   rpm -ivh /tmp/gpfs.msg.en_US-2.3.0-3.noarch.rpm
   ```

2.8.6 Setting the PATH for the GPFS command in the GPFS server

To set the PATH for the GPFS command:

1. Log on to the GPFS server as root.
2. Add the following lines by editing the $HOME/.bash_profile script:

   ```
   vi $HOME/.bash_profile
   ```

   Add the lines in Example 2-25 to the bash_profile script.

Example 2-25 Lines to add to the bash_profile script

```
GPFS_HOME=/usr/lpp/mmfs/
export GPFS_HOME
PATH=$PATH:$GPFS_HOME/bin
export PATH
MANPATH=$MANPATH:$GPFS_HOME/man
export MANPATH
```

3. Save the file and execute the shell script:

   ```
   source  $HOME/.bash_profile
   ```

2.8.7 Building the GPFS open source portability layer in GPFS NSD servers

For the CSM managed compute nodes to be able to mount GPFS file systems as part of the cluster, the GPFS open source portability layer binaries must be built. When the binaries are built, they then must be copied to the CSM management server.

See the /usr/lpp/mmfs/src/README file in your GPFS server for more information about building the open source portability layer.

Note: The portability layer binaries must be built with the same server architecture and operating system version as the compute nodes. If they are not the same, you must build them on a compatible server and copy the binaries to the CSM management server as described in this section.

When the GPFS open source portability layer binaries have been built:

1. Log on to your CSM management server, csmserv, as root.
2. Copy the GPFS open source portability layer binaries to the /cfmroot/usr/lpp/mmfs/bin directory by issuing the following commands:

   ```
   mkdir -p /cfmroot/usr/lpp/mmfs/bin
   scp gpfs1:/cfmroot/usr/lpp/mmfs/bin/mmfslinux /usr/lpp/mmfs/bin
   scp gpfs1:/cfmroot/usr/lpp/mmfs/bin/tracedev /usr/lpp/mmfs/bin
   scp gpfs1:/cfmroot/usr/lpp/mmfs/bin/dumpconv /usr/lpp/mmfs/bin
   ```

   If you are copying the binaries from a server other than your GPFS server, use the correct node name in the **scp** command.

   When the compute nodes are provisioned with the **installnode** command, the GPFS open source binaries are automatically copied during the installation process.

2.8.8 Configuring external storage for GPFS

This section describes the general process for configuring external storage for use with GPFS. Because there are many variations of external storage arrays, it is difficult to describe the process for preparing storage for different fibre channel arrays, but we do describe the process for configuring the GPFS server for an external fibre channel array.

Our configuration used an IBM TotalStorage DS4300 Turbo fibre channel array. We configured a 200 GB RAID-5 array to support the GPFS file system requirements. Our GPFS primary and secondary servers use a QLogic QLA2300 host bus adapter to connect to the array.

After you configure the fibre-attached storage array, you must configure your GPFS servers to use the fibre-attached storage. GPFS needs the sg.o module for failover between the primary and secondary NSD servers. This driver is not loaded automatically when the machine reboots.

To configure your GPFS servers to use the fibre-attached storage:

1. Log on to your CSM management server, csmserv, using the root account.
2. Create a script called /cfmroot/etc/rc.modules for automatic loading and then distribute it to all nodes in the cluster:

   ```
   vi /cfmroot/etc/rc.modules
   ```

   Insert the following lines in the script and save the file:

   ```
   #!/bin/sh
   /sbin/modprobe sg
   ```

3. Execute the edited file:

   ```
   chmod 755 /cfmroot/etc/rc.modules
   cfmupdatenode -a
   ```

4. Add a line to the /etc/modules.conf file for your primary and secondary GPFS servers, which allows MULTI_LUN support for the kernel.
5. Log on to your GPFS primary NSD server, gpfs1, using the root account.
6. Edit the /etc/modules.conf file and add the SCSI options line:

   ```
   vi /etc/modules.conf
   ```

   Insert the following line and save the file:

   ```
   options scsi_mod max_scsi_luns=255
   ```

7. After configuring the storage, you must unload and reload the FC HBA driver so that it recognizes the new storage. Unload the current driver by issuing the **rmmod** command:

   ```
   rmmod qla2300
   ```

   Install the new driver by issuing:

   ```
   insmod qla2300
   ```

8. Check to see if the disks have been found by issuing the **dmsg** command.

   You should see several lines that show the new devices being recognized:

   ```
   Attached scsi disk sda at scsi0, channel 0, id 0, lun 1
   Attached scsi disk sdb at scsi0, channel 0, id 0, lun 31
   SCSI device sda: 426973855 512-byte hdwr sectors (218611 MB)
   ```

   In our example, our new partitions were displayed as /dev/sda.

9. Format the disks so that you can create four primaries in the /dev/sda LUN to support the GPFS file system:

   ```
   fdisk /dev/sda
   ```

   The **fdisk** command prompts you for the partition number (1-4) and the size, which is 50 GB. You need to create four partitions that are 50 GB each. When you are finished, you must write out the partition table to the disk buy, using the “w” option for disk.

10. Repeat steps 5-8 for your secondary GPFS Server, gpfs2. Do not repeat step 9 because you only have to **fdisk** the SCSI device once.

2.8.9 Creating the GPFS backup server

To create the GPFS backup server, repeat the steps described in 2.8.1, “Installing the base Linux operating system in the GPFS server” on page 44 to 2.8.9, “Creating the GPFS backup server” on page 48. Be sure to add gpfs1, gpfs1p, gpfs2, and gpfs2p to /root/.rhosts.

2.9 Creating the GPFS cluster

The first thing you must do in the process of creating a GPFS cluster is define the characteristics of the cluster with the **mmcrcluster** command. This section describes this process.

2.9.1 Creating the node descriptor file

When creating the GPFS cluster, you must provide a list of node descriptors, one per line for each node, to be included in the cluster including the storage nodes (gpfs1, gpfs2).

The descriptor is specified in the form:

```
PrimaryNodeName::SecondaryNodeName
```

You create a file in the primary GPFS servers /tmp directory called gpfs.nodes as follows:

1. Log on to the primary GPFS server, gpfs1, as root.
2. Create a file called /tmp/gpfs.nodes that contains the primary and secondary GPFS servers and all compute nodes. Also include the public (admin) network for each machine in the same definition.
3. Create the node descriptor file:

   ```
   vi /tmp/gpfs.nodes
   ```

4. Insert the contents of Example 2-26 in the file.

Example 2-26 Contents for the node descriptor file

```
gpfs1p::gpfs1
gpfs2p::gpfs2
node1p::node1
node2p::node2
node3p::node3
node4p::node4
```

Save the file. You can also add additional nodes to the file to match your configuration.

2.9.2 Defining the GPFS cluster

You must now define the GPFS cluster with the **mmcrcluster** command, using RSH as a remote shell as follows (you can also use SSH if your security requirements dictate it):

1. Log on to the primary GPFS server, gpfs1, as root.
2. Change the directory to the /tmp directory, which contains the /tmp/gpfs.nodes file that you created. Create the GPFS cluster by running the **mmcrcreate** command. Use gpfs1p as your primary GPFS server and gpfs2p as your secondary GPFS server:

   ```
   mmcrcluster -t lc -p gpfs1p -s gpfs2p -n /tmp/gpfs.nodes
   ```

If you are successful, the GPFS output shows the cluster configuration. You can view the cluster configuration anytime by issuing **mmlscluster** from the command prompt.

2.9.3 Starting GPFS

You start GPFS with the **mmstartup** command. Log on to the primary GPFS server, gpfs1, as root and issue:

```
mmstartup -a
```

Your results should resemble the following output:

```
Tue Jun  7 17:53:09 EDT 2005: mmstartup: Starting GPFS ...
```

2.9.4 Creating NSDs

You create a descriptor file before creating your NSDs. The file should contain information about each disk in the NSD. In 2.8.8, “Configuring external storage for GPFS” on page 47, you created several file systems with SAN-based storage. You also created four partitions that were attached to your SAN. The device names for these partitions can be determined by using **fdisk -l** on your primary GPFS server.

The process for creating NSDs, including creating the descriptor file, is as follows:

1. Log on to your primary GPFS server, gpfs1, as root.
2. Issue:

   ```
   fdisk -l
   ```

   Your output should resemble Example 2-27.

Example 2-27 Output of fdisk -1

```
Disk /dev/sda: 218.6 GB, 218610613760 bytes
255 heads, 63 sectors/track, 26577 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

   Device Boot    Start       End    Blocks   Id  System
/dev/sda1             1      6500  52211218+  83  Linux
/dev/sda2          6501     13001  52219282+  83  Linux
/dev/sda3         13002     19502  52219282+  83  Linux
/dev/sda4         19503     26577  56829937+  83  Linux

Disk /dev/sdb: 20 MB, 20971520 bytes
64 heads, 32 sectors/track, 20 cylinders
Units = cylinders of 2048 * 512 = 1048576 bytes
```

```
Disk /dev/sdb doesn't contain a valid partition table

Disk /dev/hda: 80.0 GB, 80032038912 bytes
255 heads, 63 sectors/track, 9730 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes

   Device Boot    Start       End    Blocks   Id  System
/dev/hda1   *         1        13    104391   83  Linux
/dev/hda2            14      9476  76011547+  83  Linux
/dev/hda3          9477      9730   2040255   82  Linux swap
```

On our IBM HS20 BladeCenter, the internal IDE drive was designated as /dev/hda, so the external SAN attached LUN has four 50 GB partitions designated as /dev/sda (/dev/sda1,2,3,4).

3. Create a descriptor file that contains the NSD information:

   ```
   vi /tmp/descfile
   ```

   ```
   /dev/sda1:gpfs1p:gpfs2p::dataAndMetadata:1
   /dev/sda2:gpfs1p:gpfs2p::dataAndMetadata:1
   /dev/sda3:gpfs1p:gpfs2p::dataAndMetadata:1
   /dev/sda4:gpfs1p:gpfs2p::dataAndMetadata:1
   ```

4. Create the NSDs with the **mmcrnsd** command:

   ```
   mmcrnsd -F /tmp/descfile
   ```

   The output should look as follows:

   ```
   mmcrnsd: Propagating changes to all effected nodes
   ```

   This is an asynchronous process.

   After successfully creating the NSD for the GPFS cluster, **mmcrnsd** will reformat the file descriptor to comment out the original disk devices and add the GPFS-assigned global name.

5. To view these changes, issue:

   ```
   cat /tmp/descfile
   ```

   Your output should look resemble Example 2-28.

Example 2-28 Output of cst /tmp/descfile

```
# /dev/sda1:gpfs1p:gpfs2p:dataAndMetadata:1
gpfs1nsd:::dataAndMetadata:1
# /dev/sda2:gpfs1p:gpfs2p:dataAndMetadata:1
gpfs2nsd:::dataAndMetadata:1
# /dev/sda3:gpfs1p:gpfs2p:dataAndMetadata:1
gpfs3nsd:::dataAndMetadata:1
# /dev/sda4:gpfs1p:gpfs2p:dataAndMetadata:1
gpfs4nsd:::dataAndMetadata:1
```

6. To view the new device names, use the **mmlsnsd** command:

   ```
   mmlsnsd
   ```

Your output should look resemble Example 2-29.

Example 2-29 Output of the mmlsnsd command

```
File system   Disk name    Primary node         Backup node
---------------------------------------------------------------------------
 gpfs0         gpfs1nsd    gpfs1p               gpfs2p
 gpfs0         gpfs2nsd    gpfs1p               gpfs2p
 gpfs0         gpfs3nsd    gpfs1p               gpfs2p
 gpfs0         gpfs4nsd    gpfs1p               gpfs2p
```

2.9.5 Creating the GPFS file system

After you have created your NSDs, you can now create the GPFS file system. You must collect the following information before proceeding:

- Mount point (use /gpfs)
- The name of the device for the file system (use gpfs0)
- The descriptor file (/tmp/descfile)

The **mmcrnsd** command formats the NSDs, prepares them for mounting the system, and adds an entry in the systems /dev/fstab file for the new file system that you are creating.

To create the GPFS file system, use the **mmcrfs** command, pointing to the /tmp/descfile that you created in the last step. You also add the **-A** option to automatically mount the file systems on the nodes as shown in Example 2-30.

Example 2-30 Creating the GPFS file system

```
mmcrfs /gpfs gpfs0 -F /tmp/descfile/ -A yes
The following disks of gpfs0 will be formatted on node gpfs1p:
    gpfs1nsd: size 52219282 KB
    gpfs2nsd: size 56829937 KB
    gpfs3nsd: size 52219282 KB
    gpfs4nsd: size 56829937 KB
Formatting file system ...
Creating Inode File
Creating Allocation Maps
Clearing Inode Allocation Map
Clearing Block Allocation Map
Flushing Allocation Maps
Disks up to size 241 GB can be added to this file system.
Completed creation of file system /dev/gpfs0.
mmcrfs: Propagating the changes to all affected nodes.
This is an asynchronous process.
```

After creating the file system, you can run the **mmlsfs** command to display your file system attributes:

```
mmlsfs gpfs0
```

Your output should resemble Example 2-31.

Example 2-31 Output of mmlsfs gpfs0

```
flag value          description
---- -------------- -----------------------------------------------------
 -s  roundRobin     Stripe method
 -f  8192           Minimum fragment size in bytes
 -i  512            Inode size in bytes
 -I  16384          Indirect block size in bytes
 -m  1              Default number of metadata replicas
 -M  1              Maximum number of metadata replicas
 -r  1              Default number of data replicas
 -R  1              Maximum number of data replicas
 -j  cluster        Block allocation type
 -D  posix          File locking semantics in effect
 -k  posix          ACL semantics in effect
 -a  1048576        Estimated average file size
 -n  32             Estimated number of nodes that will mount file system
 -B  262144         Block size
 -Q  none           Quotas enforced
     none           Default quotas enabled
 -F  102400         Maximum number of inodes
 -V  8.01           File system version. Highest supported version: 8.01
 -u  yes            Support for large LUNs?
 -z  no             Is DMAPI enabled?
 -E                 Exact mtime default mount option
 -S                 Suppress atime default mount option
 -d  gpfs10nsd;gpfs6nsd  Disks in file system
 -A  yes            Automatic mount option
 -o  none           Additional mount options
```

Also, after creating the file system, GPFS adds a new file system to the /etc/fstab as shown:

```
/dev/gpfs0          /gpfs               gpfs       rw,mtime,atime,dev=gpfs0,au
tostart 0 0
```

2.9.6 Mounting the GPFS file system

The newly created GPFS file system is not automatically mounted when you install the GPFS cluster. To mount the file system:

1. Log on to the CSM management server as root.
2. Mount the GPFS file system on the primary GPFS server, gpfs1p, by issuing:

   ```
   rsh -n gpfs1p mount -a
   ```

3. Mount the GPFS file system on the secondary GPFS server, gpfs2p, by issuing:

   ```
   rsh -n gpfs2p mount -a
   ```

4. Mount the GPFS file system on the remaining GPFS clients on the compute nodes by issuing:

   ```
   dsh -a mount -a
   ```

2.9.7 Shutting down GPFS

GPFS is stopped with the **mmshutdown** command. Log on to the primary GPFS server, gpfs, as root and issue:

```
mmshutdown -a
```

Your output should resemble Example 2-32.

Example 2-32 Output of mmshutdown -a

```
Tue Jun  7 17:51:34 EDT 2005: mmshutdown: Starting force unmount of GPFS file
systems
Tue Jun  7 17:51:39 EDT 2005: mmshutdown: Shutting down GPFS daemons
node6p:  Shutting down!
node3p:  Shutting down!
node2p:  Shutting down!
node4p:  Shutting down!
gpfs2p:  Shutting down!
node1p:  Shutting down!
gpfs1p:  Shutting down!
node6p:  Unloading modules from /usr/lpp/mmfs/bin
node6p:  Unloading module mmfs
node3p:  'shutdown' command about to kill process 1720
node3p:  Unloading modules from /usr/lpp/mmfs/bin
node2p:  'shutdown' command about to kill process 1729
node2p:  Unloading modules from /usr/lpp/mmfs/bin
node4p:  'shutdown' command about to kill process 1662
node4p:  Unloading modules from /usr/lpp/mmfs/bin
node3p:  Unloading module mmfs
node2p:  Unloading module mmfs
node4p:  Unloading module mmfs
node4p:  Unloading module mmfslinux
node4p:  Unloading module tracedev
node2p:  Unloading module mmfslinux
node2p:  Unloading module tracedev
gpfs2p:  'shutdown' command about to kill process 27589
gpfs2p:  Unloading modules from /usr/lpp/mmfs/bin
gpfs1p:  'shutdown' command about to kill process 19264
gpfs1p:  Unloading modules from /usr/lpp/mmfs/bin
gpfs2p:  Unloading module mmfs
node1p:  'shutdown' command about to kill process 1618
node1p:  Unloading modules from /usr/lpp/mmfs/bin
gpfs1p:  Unloading module mmfs
node1p:  Unloading module mmfs
node3p:  Unloading module mmfslinux
node3p:  Unloading module tracedev
node6p:  Unloading module mmfslinux
node6p:  Unloading module tracedev
gpfs2p:   Unloading module mmfslinux
gpfs2p:  Unloading module tracedev
gpfs1p:  Unloading module mmfslinux
gpfs1p:  Unloading module tracedev
node1p:  Unloading module mmfslinux
node1p:  Unloading module tracedev
Tue Jun  7 17:51:47 EDT 2005: mmshutdown: Finished
```

2.10 Configuring CSM management server files for GPFS clients

In this section, we describe the steps necessary for configuring the CSM management server files for GPFS clients.

To set up the GPFS environment on each compute node, create a profile that is distributed to the compute nodes, which adds the GPFS binary directory, as follows:

1. Log on to the CSM management server as root if you have not already done so.
2. If the directory does not exist, issue:

   ```
   mkdir -p /cfmroot/etc/profile.d
   ```

3. Create the /cfmroot/etc/profile.d/mmfs.sh file as shown in Example 2-33.

Example 2-33 Creating the /cfmroot/etc/profile.d/mmfs.sh file

```
GPFS_HOME=/usr/lpp/mmfs/
export GPFS_HOME
PATH=$PATH:$GPFS_HOME/bin
export PATH
MANPATH=$MANPATH:$GPFS_HOME/man
export MANPATH
```

4. Issue:

   ```
   chmod 755 /cfmroot/etc/profile.d/mmfs.sh
   ```

Your environment is now ready to use GPFS clients.

3

Installing and configuring CSM managed compute nodes

In this chapter we describe how to install and configure CSM managed compute nodes. This chapter addresses the following topics:

- Installing CSM managed nodes
- Adding GPFS managed nodes
- Configuring LoadLeveler engines
- Testing the Scali MPI Connect installation

3.1 Installing CSM managed nodes

After the installing the CSM management server and completing the required configuration steps, you are ready to install Linux on the CSM managed nodes. This is done with the **installnode** command.

To install Linux on a single node, issue the following command:

```
installnode node1p
```

The results will resemble this output:

```
Rebooting Node for full install: node1p.pbm.ihost.com,
```

To install Linux on all nodes, issue:

```
installnode -P
```

The results will resemble this output:

```
Rebooting Node for full install: node1p.pbm.ihost.com,
Rebooting Node for full install: node2p.pbm.ihost.com,
Rebooting Node for full install: node3p.pbm.ihost.com,
Rebooting Node for full install: node4p.pbm.ihost.com,
```

To check the status of the installation, use the **monitorinstall** command:

```
monitorinstall
```

When an **installnode** command is first issued, **monitorinstall** returns the Rebooting and Installing Node status for all nodes as shown in Example 3-1.

Example 3-1 Rebooting and Installing Node status

```
monitorinstall
Node                    Mode                Status
-------------------------------------------------------------------------
node1p.pbm.ihost.com    Managed             Rebooting and Installing Node
node2p.pbm.ihost.com    Managed             Rebooting and Installing Node
node3p.pbm.ihost.com    Managed             Rebooting and Installing Node
node4p.pbm.ihost.com    Managed             Rebooting and Installing Node
```

After the base operating system has been installed, CSM runs **makenode** on the managed node to install CSM files and enter the compute node into the CSM management server database. During this phase, **monitorinstall** returns the Starting makenode to install CSM RPMs status as shown in Example 3-2.

Example 3-2 Starting makenode to install CSM RPMs status

```
Node                    Mode                Status
-------------------------------------------------------------------------
node1p.pbm.ihost.com    Managed             Starting makenode to install CSM RPMs
node2p.pbm.ihost.com    Managed             Starting makenode to install CSM RPMs
node3p.pbm.ihost.com    Managed             Starting makenode to install CSM RPMs
node4p.pbm.ihost.com    Managed             Starting makenode to install CSM RPMs
```

Infrastructure dictates the amount of time that the installation takes, and as a result, this time varies. When the installation is completed, **monitorinstall** returns the Installed status for all nodes as shown in Example 3-3.

Example 3-3 Installed status

```
monitorinstall

Node                    Mode                Status
-------------------------------------------------------------------------
node1p.pbm.ihost.com    Managed             Installed
node2p.pbm.ihost.com    Managed             Installed
node3p.pbm.ihost.com    Managed             Installed
node4p.pbm.ihost.com    Managed             Installed
```

3.2 Adding GPFS nodes

To add GPFS nodes to your cluster, first log on to your primary GPFS server as root.

To add nodes to a GPFS cluster:

1. Create a file in /tmp called gpfs.allnodes that contains the host name of all GPFS nodes:

   ```
   vi /tmp/gpfs.allnodes
   ```

   Insert the following lines into the file and save it:

   ```
   node1p
   node2p
   node3p
   node4p
   ```

2. Issue the following command to add nodes:

   ```
   mmaddnode -n /tmp/gpfs.allnodes
   ```

To add a single node to a GPFS cluster:

1. Issue:

   ```
   mmaddnode node
   ```

   Where *node* is the private network host name of the CSM managed node. Repeat the **mmaddnode** command for all nodes in the GPFS cluster.

2. After the nodes have been added to the GPFS cluster, you must start GPFS and mount all the /gpfs partitions on the node that you have added:

   ```
   mmstartup -w node
   dsh -n node mount -a
   ```

3. Repeat the **mmaddnode** command for all nodes in the GPFS cluster.

3.3 Configuring LoadLeveler engines

This section describes how to configure LoadLeveler engines.

3.3.1 Creating the LoadLeveler configuration script

To configure your LoadLeveler engines, you must create a script in the shared file system that configures each server in the LoadLeveler cluster as follows:

1. Log on to the LoadLeveler Central Scheduler machine as root.
2. Create an installation script file:

   ```
   vi /dist/Loadl/insloadl.sh
   ```

   Insert the following lines into the file:

   ```
   #!/bin/bash
   cd /opt/ibmll/LoadL/sbin/
   ./install_ll -y -d /dist/LoadL
   ```

   Save the file.
3. Make the script executable so that it can run under dsh:

   ```
   chmod 755 vi /dist/Loadl/insloadl.sh
   ```
4. Create a configuration script file. You should substitute the machine name gridschp with the host name of your LoadLeveler Central Scheduler server:

   ```
   vi /dist/Loadl/cfgloadl.sh
   ```

   Insert the following lines into the file:

   ```
   #!/bin/bash
   source  /etc/profile.d/loadl.sh
   llinit -local /var/loadl -release /opt/ibmll/LoadL/full -cm gridschp
   ```

   Save the file.
5. Make the script executable so that it can run under dsh:

   ```
   chmod 755 vi /dist/Loadl/cfgloadl.sh
   ```

3.3.2 Adding LoadLeveler engine definitions to the cluster

To add LoadLeveler engine definitions to the cluster:

1. Log on to the LoadLeveler central manager machine as loadl if you have not already done so.
2. Edit the /home/loadl/loadl_Admin configuration file:

   ```
   vi /home/loadl/loadl_Admin
   ```
3. Find the Machine Stanzas section in the file and edit them as shown in Example 3-4.

Example 3-4 Editing the Machine Stanzas section

```
Edit the stanzas files as shown below
################################################################################
# MACHINE STANZAS:
# These are the machine stanzas; the first machine is defined as
# the central manager.  mach1:, mach2:, etc. are machine name labels -
# revise these placeholder labels with the names of the machines in the
# pool, and specify any schedd_host and submit_only keywords and values
```

```
# (true or false), if required.
#############################################################################
gridschp: type = machine        central_manager = true
                                schedd_host = true
                                alias = gridsch
node1p: type = machine
node2p: type = machine
node3p: type = machine
node4p: type = machine
node6p: type = machine          schedd_host = false
                                submit_only = true
```

4. In our example, we used gridsch as the Central Scheduler name and gridschp as the alias name for the Central Scheduler. This is important if you have a management and a private network, as we did in our Optimized Analytic Infrastructure configuration.
5. Add all of the private network host names after the Central Scheduler definitions. In our example, we have only four hosts, but your configuration might have more.
6. Save the file.
7. To minimize the number of configuration files for the compute nodes, create a link for each compute node to the /home/load/local/ computenode.LoadL_config.local file:

```
cd /home/loadl/local
ln computenode.LoadL_config.local node1p. LoadL_config.local
ln computenode.LoadL_config.local node2p. LoadL_config.local
ln computenode.LoadL_config.local node3p. LoadL_config.local
ln computenode.LoadL_config.local node4p. LoadL_config.local
```

In this example, we had only four nodes; your configuration might have more.

3.3.3 Running the LoadLeveler configuration script to set up engines

The final step for configuring the LoadLeveler engines is to run the installation and configuration scripts for all nodes:

1. Log on to the CSM management node as root. You can configure a single or multiple nodes from the CSM management server by using dsh.
2. To install and configure all nodes, run the following scripts:

```
dsh -a -l loadl  /dist/Loadl/insloadl.sh
dsh -a -l loadl  /dist/Loadl/cfgloadl.sh
```

3. To install and configure a single or a set of nodes, run:

```
dsh -n node01 -l loadl  /dist/Loadl/insloadl.sh
dsh -n node01 -l loadl  /dist/Loadl/cfgloadl.sh
```

3.3.4 Operating the LoadLeveler engines

This section describes how to start, stop, and check the LoadLeveler engines.

To start a LoadLeveler worker engine:

1. Log on to the grid/job scheduling node as user loadl.

2. Issue:

   ```
   rsh -n node1 llctl -h node start
   ```

   Where *node* is the private network host name of the worker engine server.

3. To start all LoadLeveler servers, issue:

   ```
   llctl -g start
   ```

To stop a LoadLeveler worker engine:

1. Log on to the grid/job scheduling node as user loadl.
2. Run:

   ```
   rsh -n node llctl -h node stop
   ```

   Where *node* is the private network host name of the worker engine server.

3. To stop the entire LoadLeveler cluster, run:

   ```
   llctl -g start
   ```

To view the status of the LoadLeveler cluster:

1. Log on to thE Grid/job Scheduling node as user loadl.
2. Run:

   ```
   llstatus
   ```

   Your output should resemble Example 3-5.

Example 3-5 Output of llstatus

```
Name                 Schedd  InQ Act Startd Run LdAvg Idle Arch      OpSys
node1p.pbm.ihost.com Avail     0   0 Idle     0 0.00  9999 i386      Linux2
node2p.pbm.ihost.com Avail     0   0 Idle     0 0.00  9999 i386      Linux2
node3p.pbm.ihost.com Avail     0   0 Idle     0 0.00  9999 i386      Linux2
node4p.pbm.ihost.com Avail     0   0 Idle     0 0.00  9999 i386      Linux2
node6p.pbm.ihost.com Avail     0   0 Idle     0 12.00 9999 i386      Linux2
node7p.pbm.ihost.com Down      0   0 Down     0 0.00     0 i386      Linux2
gridschp.pbm.ihost.com Avail   0   0 Idle     0 0.00     0 i386      Linux2

i386/Linux2              7 machines      0  jobs      0  running
Total Machines           7 machines      0  jobs      0  running

The Central Manager is defined on gridschp
The BACKFILL scheduler is in use
All machines on the machine_list are present.
The following machine is present, but not on the machine_list.
node7p
```

3.4 Testing the Scali MPI Connect installation

To test the Scali MPI Connect installation:

1. Log on to the CSM management server as root.
2. Run the mpimon test program for all the CSM managed nodes in your cluster:

   ```
   dsh -a /opt/scali/bin/mpimon /opt/scali/examples/bin/hello -- node1p node2p
   node3p node4p
   ```

Your output should resemble Example 3-6.

Example 3-6 Output of mpimon test program

```
node1p.pbm.ihost.com: Starting on rank 0 node1p.pbm.ihost.com
node1p.pbm.ihost.com: Starting on rank 1 node2p.pbm.ihost.com
node1p.pbm.ihost.com: Starting on rank 2 node3p.pbm.ihost.com
node1p.pbm.ihost.com: Starting on rank 3 node4p.pbm.ihost.com
node2p.pbm.ihost.com: Starting on rank 0 node1p.pbm.ihost.com
node2p.pbm.ihost.com: Starting on rank 2 node3p.pbm.ihost.com
node2p.pbm.ihost.com: Starting on rank 3 node4p.pbm.ihost.com
node2p.pbm.ihost.com: Starting on rank 1 node2p.pbm.ihost.com
node3p.pbm.ihost.com: Starting on rank 2 node3p.pbm.ihost.com
node3p.pbm.ihost.com: Starting on rank 0 node1p.pbm.ihost.com
node3p.pbm.ihost.com: Starting on rank 1 node2p.pbm.ihost.com
node3p.pbm.ihost.com: Starting on rank 3 node4p.pbm.ihost.com
node4p.pbm.ihost.com: Starting on rank 0 node1p.pbm.ihost.com
node4p.pbm.ihost.com: Starting on rank 1 node2p.pbm.ihost.com
node4p.pbm.ihost.com: Starting on rank 2 node3p.pbm.ihost.com
node4p.pbm.ihost.com: Starting on rank 3 node4p.pbm.ihost.com
```

3.5 Summary

If you followed the instructions in Chapter 2 and this chapter, using the versions of the components we described in Chapter 1, your installation of the components for the Optimized Analytic Infrastructure is complete and it is ready to use.

If you used other processor architectures, operating system versions, and product versions, your results may vary.

A

ApplicationWeb user's guide

This appendix is a user guide for ApplicationWeb.

Introduction

ApplicationWeb supports the deployment of native binaries as services in an SOA.

Service orientation

SOAs are implemented by serialized, encapsulated interactions between managed processes. There are three basic types of interactions:

- Initializations
- Repeatable transactions
- Terminations

Serialization is required for transferring data between processes. Encapsulation is required for executing the methods remotely, as services.

Step 1: Encapsulate

All remote methods are encapsulated in C modules. It does not matter if the code is C++ internally. The first step is to move all the actions that can be executed remotely into separate modules. The argument list can include objects or classes at this point. The goal is to have a main program that approximates the following example:

```
main()
{
Foo foo;
Bar bar;
Error e;
e=remoteInitialize(foo);
...
e=remoteRun(foo, bar);
...
e=remoteTerminate(foo,bar);
}
```

Techniques for encapsulation

The first step is to separate the code block into the three basic types, and encapsulate the code into separate modules. When you do this, you might have to fix the following compile time errors:

- Try blocks might have no catch blocks and vice versa.

  In these cases, copy the relevant catch blocks into the module and add a throw.

- Variables might now be undeclared in some blocks:
  - For simple variables: Copy declarations above the module calls and pass their references as arguments to the module interface.
  - For objects: Change your code to use new/delete operators. Then, declare pointers above the module calls and pass pointer references as arguments. This prevents trouble with copy constructors and garbage collection.
  - For arrays: Be sure to include the length of the array in the argument lists for the various methods. Compile, build, and test the application to be sure that the encapsulation is working.

Wrapper

As a final step, write a single wrapper function, the argument list of which is the union of the argument lists in the remote methods that you have just written plus one argument to indicate the action to be performed. Here is a simple prototype:

```
Error remoteWrapper( Action action, Foo foo, Bar bar)
{
if (action == Initialize)
return remoteInitialize(foo);
if (action == Run)
return remoteRun(foo, bar);
if (action == Terminate)
return remoteTerminate(foo, bar);
return BadAction;
}
```

Put the function prototype into a separate header file, such as remoteAW.h. Put the function implementation into a separate source file, for example, remoteAW.cpp. The goal is to have the file with the main program and the remoteAW.h file completely free of application-specific header files, and all types and constructs that are not basic C (such as enum or bool). Put all necessary header files into the SOURCE file. For example, if Remote.h is needed by remoteAW.cpp, put the include file directly into remoteAW.cpp.

Step 2: Serialize

Data passed between processes must be serialized. ApplicationWeb automatically performs serialization/deserialization as well as marshaling and demarshaling between operating systems. Currently, the data structures to be serialized must be compatible with formats supported by basic C types and structures.

C: Complete

If the interface that you developed uses only C source code, you are done. The ApplicationWeb tooling develops the appropriate interfaces for remote execution of these methods.

C++: Additional tasks

If your code is C++, there are additional tasks. Each class passed through the function interface that you selected must be serialized at the client, deserialized at the server, and then possibly reserialized back to the client. ApplicationWeb does not process C++ class header files at this time.

To implement a serialization pattern, follow this basic outline:

1. For each Foo class passed through the interface for the encapsulated methods, declare a struct like this:

   ```
   typedef struct {} Foo\_S;
   ```

2. If you actually use a Bar::Foo derived class, declare both structs, with the child struct having a pointer to the parent struct:

   ```
   typedef struct {} Foo\_S;
   typedef struct { Foo\_S *foo\_S; } Bar\_S;
   ```

3. Place the struct declarations in the projectionAW.h HEADER file.
4. Declare deserialization constructors and reserialization methods in the class definitions:

```
#include "projectionAW.h"
...
class Foo
{
public:
...
Foo( Foo\_S *foo\_S);
void to\_S( Foo\_S **);
}
```

5. In the remote method interfaces, replace all Foo class variables by their serialized versions Foo S.
6. Classes passed as input arguments are initialized by the deserialization constructor at the beginning of the method, and outputs are serialized at the end:

```
#include "projectionAW.h"
...
void remote(Foo\_S *foo\_S, Bar\_S **bar\_S)
{
Foo foo(foo\_S);
...
foo.to\_S(bar\_S);
}
```

7. In the client method, replace all instances of the class argument with its serialized counterpart. The serialized class might be changed during the remote method, so it is a good idea to pass the address of the struct so that those changes can be reflected in the client:

```
#include "projectionAW.h"
...
Foo *foo;
Foo\_S *foo\_S;
...
foo->to\_S(&foo\_S);
delete foo;
remote(&foo\_S);
foo = new Foo(foo\_S);
...
```

At the end of this process, you should be able to compile the project. It will not be built until you provide the implementations of the serialization methods, however.

Implementing the serialization methods

It is helpful to construct (and test) a copy constructor for each class to be passed through the remote interface and set up your unit tests and other tasks before you implement the serialization methods. Check for simple traps, such as classes whose state depends on a file pointer or a socket connection. The remote methods cannot serialize these types of simple traps.

The deserialization constructor is basically a copy constructor, except that you do not use methods from the incoming object. The following example is a simple prototype:

```
#include "projectionAW.h"
```

```
...
Foo::Foo(Foo\_S *pf\_S) {
// duplicate copy constructor logic
...
// duplicate constructor initializations
...
}
...
```

The serialization method has two parts: one part to allocate memory regions and another part to copy over the regions. Again, it is basically a copy constructor, so you can hack the code that you wrote for the deserialization constructor. You might also have to create fields in the struct to replicate the initialization processes in the constructor as shown in the following prototype:

```
#include "projectionAW.h"
...
void Foo::to\_S(Foo\_S **ppf\_S)
{
Foo\_S *pf\_S = *ppf\_S;
// memory allocation
if (!pf\_S)
{
pf\_S = (Foo\_S *)malloc(sizeof(Foo\_S));
//modify the copy constructor logic to allocate regions
...
}
// modify copy constructor logic to copy over regions
...
// pass back pointer
*ppf\_S = pf\_S;
}
...
```

Finally, implement a method that frees the serialized data. Follow the pattern in the deserialization constructor to be sure that all memory regions are freed. Here is a prototype:

```
#include "projectionAW.h"
...
void Foo\_S_free(Foo\_S **ppf\_S)
{
Foo\_S *pf\_S = *ppf\_S;
// modify the copy constructor logic to free regions
...
// free pointer

free(pf\_S);
*ppf\_S = NULL;
}
```

Put all this together and build and test the program.

Step 3: Library

Build a library, such as remoteLib, that exports the remoteWrapper wrapper symbol. Then rebuild your main program to access that symbol from the library. Perform all your unit tests because this part must work properly.

Step 4: ApplicationWeb interface generation

The ApplicationWeb tooling is an Eclipse plug-in that generates an XML file that describes the interfaces. From this XML file, various style sheets can generate implementations. The tooling is basically self-explanatory with helpful editors.

Interpreting pointers can be difficult: Are they references for output? Are they arrays? If so, how long is the array? The tooling makes educated guesses, but basically you have to make these decisions.

At the end of the tool processes, you have generated a pair of Java files, remoteBase.java and remoteDriver.java, and an XML file, remoteLib.Augmented.xml. The remoteDriver class encodes a simple use case for a single call to the remoteWrapper method; you must modify this code to fill the data regions and to reproduce the logic of the program.

The remoteBase class implements the ApplicationWeb interface to the method remoteWrapper; you probably do not have to look at this class unless you are curious about the technology.

The XML file, remoteLib.Augmented.xml, describes the interface. At the top of the file, there is a string that identifies the location of the remoteLib library. It is important that this string describe where the library is located on the remote installation. Otherwise, you receive mysterious error messages such as “remoteLib.so: file not found,” even though, when you look for the file, you see that it is exactly where it should be.

Summary

You have now created a remote service method that can be called from a number of client environments. The remote instantiation, deployment, and life cycle can be managed by application servers interfacing with the ApplicationWeb server.

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

IBM Redbooks

For information about ordering these publications, see “How to get IBM Redbooks” on page 70. Note that some of the documents referenced here may be available in softcopy only.

- *Linux Clustering with CSM and GPFS*, SG24-6601
- *Workload Management with LoadLeveler*, SG24-6038
- *IBM eServer xSeries and BladeCenter Server Management*, SG24-6495

Other publications

These publications are also relevant as further information sources: They will assist you with installing, configuring, testing and debugging the individual components of the Optimized Analytic Infrastructure solution.

- CSM
  - *CSM for AIX 5L and Linux V1.4.1 Administration Guide*, SA23-1343-00
  - *CSM for AIX 5L and Linux V1.4.1 Planning and Installation Guide*, SA23-1344-00
  - *CSM for AIX 5L and Linux V1.4.1 Command and Technical Reference*, SA23-1345-00
- LoadLeveler
  - *LoadLeveler for AIX 5L and Linux V3.3 Using and Administering*, SA22-7881-03
  - *LoadLeveler for AIX 5L and Linux V3.3 Installation Guide*, GI10-0763-00
  - *LoadLeveler for AIX 5L and Linux V3.3 Diagnosis and Messages*, GA22-7882-03
- GPFS
  - *GPFS V2.3 Administration and Programming Reference*, SA22-7967-02
  - *GPFS V2.3 Concepts, Planning, and Installation Guide*, GA22-7968-02
  - *GPFS V2.3 Problem Determination Guide*, GA22-7969-02

Online resources

These Web sites are also relevant as further information sources:

- Scali MPI Connect

  http://www.scali.com

  - Scali MPI Connect Users Guide, V4.4:

    http://www.scali.com/index.php?option=com_content&task=view&id=7&Itemid=109/

- IBM eServer Cluster support, Downloading and Installing Drivers for CSM

  http://techsupport.services.ibm.com/server/cluster/fixes/csmdriverdownload.html

How to get IBM Redbooks

You can search for, view, or download Redbooks, Redpapers, Hints and Tips, draft publications and Additional materials, as well as order hardcopy Redbooks or CD-ROMs, at this Web site:

ibm.com/redbooks

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

Index

O

P

R

S

T

V

W

(0.1”spine)
0.1”<->0.169”
53<->89 pages

IBM Optimized Analytic Infrastructure Solution

Installation Guide V1.0

Redbooks

Financial solution that is scalable and reliable

IBM hardware, software, and strategic software products

Installing and configuring the components

IBM Optimized Analytic Infrastructure (OAI) is a solution suitable for institutions seeking either to move their financial system to a scalable industrial-strength solution or to build an integrated, high-performance infrastructure from the ground up.

The aim of this guide is to help you install and configure the components of Optimized Analytic Infrastructure. It consists of IBM hardware and software, along with strategic software products from other selected vendors.

INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION

BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

For more information:
ibm.com/redbooks

SG24-7381-00 ISBN 0738496723