Redbooks
ibm.com/redbooks

IBM Spectrum LSF Suite Installation Best Practices Guide

Dino Quintero

Mark Black

Ahmad Hussein

Bill McMillan

Gabor Samu

John Welch

Cloud

Infrastructure Solutions

IBM

Redpaper

IBM Redbooks

IBM Spectrum LSF Suite: Installation Best Practices Guide

March 2020

REDP-5572-00

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

First Edition (March 2020)

This edition applies to:

IBM Spectrum LSF Suite Editions: Workgroup, HPC and Enterprise 10.2 Fix Pack 9

This document was created or updated on April 21, 2020.

Contents

Notices

This information was developed for products and services offered in the US. This material might be available from IBM in other languages. However, you may be required to own a copy of the product or product version in that language in order to access it.

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 grant you any license to these patents. You can send license inquiries, in writing, to:
IBM Director of Licensing, IBM Corporation, North Castle Drive, MD-NC119, Armonk, NY 10504-1785, US

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 jurisdictions 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 websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

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

The performance data and client examples cited are presented for illustrative purposes only. Actual performance results may vary depending on specific configurations and operating conditions.

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.

Statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

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 actual people or business enterprises 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. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the web at “Copyright and trademark information” at http://www.ibm.com/legal/copytrade.shtml

The following terms are trademarks or registered trademarks of International Business Machines Corporation, and might also be trademarks or registered trademarks in other countries.

Redbooks (logo) ®	IBM Cloud™	POWER®
AIX®	IBM Spectrum®	POWER9™
IBM®	LSF®	Redbooks®

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

Preface

This IBM® Redpaper publication describes IBM Spectrum® LSF® Suite best practices installation topics, application checks for workload management, and high availability configurations by using theoretical knowledge and hands-on exercises. These findings are documented by way of sample scenarios.

This publication addresses topics for sellers, IT architects, IT specialists, and anyone who wants to implement and manage a high-performing workload management solution with LSF. Moreover, this guide provides documentation to transfer how-to-skills to the technical teams, and solution guidance to the sales team. This publication compliments documentation that is available at IBM Knowledge Center, and aligns with educational materials that are provided by IBM Systems.

Authors

This paper was produced by a team of specialists from around the world working at IBM Redbooks, Poughkeepsie Center.

Dino Quintero is an IT Management Consultant and an IBM Level 3 Senior Certified IT Specialist with IBM Redbooks® in Poughkeepsie, New York. He has 24 years of experience with IBM Power Systems technologies and solutions. Dino shares his technical computing passion and expertise by leading teams that develop technical content in the areas of enterprise continuous availability, enterprise systems management, high-performance computing, cloud computing, artificial intelligence (including machine and deep learning), and cognitive solutions. He also is a Certified Open Group Distinguished IT Specialist. Dino holds a Master of Computing Information Systems degree and a Bachelor of Science degree in Computer Science from Marist College.

Mark Black is a Principal Architect in IBM Canada. He has 28 years of experience in IT and development. Mark holds a M.E.Sc. from the University of Western Ontario. His areas of expertise include system administration, HPC cluster, and Cloud deployment. Mark has VLOGed extensively about IBM Spectrum LSF Suite and hybrid LSF clusters.

Ahmad Hussein is an IBM Systems Lab Services Lead Consultant covering the Middle East and Africa regions. He is IBM Certified Technical Specialist, Expert Level, focusing on designing, leading, and implementing solutions for IBM POWER®, HPC, AIX®, and Linux. Holding a M.Sc in Industrial Engineering, and a B.Sc in Electrical and Computing Engineering, Ahmad has more than 15 years of experience and has been working for IBM since 2011.

Bill McMillan is the Global Offering Leader for IBM Spectrum LSF. With over 25 years experience in High Performance Computing in a wide range of industries, Bill is responsible for the offering management, strategic direction, and OEM relationships for the IBM Spectrum LSF family of products. His interests include the use of GPUs, containers, and cloud for high-performance and high throughput computing. He joined IBM in 2012 as part of IBM acquisition of Platform Computing. At Platform Computing, he undertook various roles including Product Architect, Solution Architect, Consultant, Technical Marketing, Systems Engineer, and Product Management. Before Platform Computing, he worked in various roles in Aerospace Engineering and Marine Engineering. Bill holds a BEng (Hons) in Naval Architecture and Offshore Engineering from the University of Strathclyde, and a Diploma in Management from The Open University.

Gabor Samu is a Skills Leader in IBM Systems specialized on IBM Spectrum Computing products. With over 20 years of experience in High Performance Computing, Gabor is responsible for sales enablement for the IBM Spectrum Computing product family. Gabor has held numerous roles ranging from technical support and product marketing through to sales enablement and has significant hands on experience with IBM Spectrum LSF. He contributes to technical forums on regular basis articles focused on Spectrum LSF and High-Performance Computing and creates technical labs on Spectrum Computing products for enablement purposes. Prior to IBM, Gabor worked in various roles providing technical support on NFS and X-Windows. Gabor holds a BSc in Computer Science and a minor in Mathematics from the University of Toronto.

John Welch is a LSF Technical Specialist. He has over 30 years of experience in IT and software development and has worked on numerous LSF implementation over the last 20 years in a pre-sales and post-sales role. John has built LSF technical labs on Tensorflow and OpenFOAM for IBM Technical University and IBM Think and published articles and demonstrations on LSF related to containers, machine learning, cloud, and MPI. John graduated from Virginia Tech with a B.S. degree in Computer Science.

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

Ludovic Enault
IBM France

Lisa Waddell
IBM US

Yi Sun, Bohai Zhang, Feng Li, Jianfeng Sun, Gabor Samu
IBM Canada

Now you can become a published author, too!

Here’s an opportunity to spotlight your skills, grow your career, and become a published author—all at the same time! Join an IBM Redbooks residency project and help write a book in your area of expertise, while honing your experience using leading-edge technologies. Your efforts will help to increase product acceptance and customer satisfaction, as you expand your network of technical contacts and relationships. Residencies run from two to six weeks in length, and you can participate either in person or as a remote resident working from your home base.

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 papers to be as helpful as possible. Send us your comments about this paper or other IBM Redbooks publications in one of the following ways:

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

  ibm.com/redbooks

- Send your comments in an email to:

  redbooks@us.ibm.com

- Mail your comments to:

  IBM Corporation, IBM Redbooks
  Dept. HYTD Mail Station P099
  2455 South Road
  Poughkeepsie, NY 12601-5400

Stay connected to IBM Redbooks

- Find us on Facebook:

  http://www.facebook.com/IBMRedbooks

- Follow us on Twitter:

  http://twitter.com/ibmredbooks

- Look for us on LinkedIn:

  http://www.linkedin.com/groups?home=&gid=2130806

- Explore new Redbooks publications, residencies, and workshops with the IBM Redbooks weekly newsletter:

  https://www.redbooks.ibm.com/Redbooks.nsf/subscribe?OpenForm

- Stay current on recent Redbooks publications with RSS Feeds:

  http://www.redbooks.ibm.com/rss.html

1

Introduction to IBM Spectrum LSF Suite

This chapter provides an introduction to IBM Spectrum LSF. This chapter also describes the latest features of IBM Spectrum LSF Suite 10.1.0.9 (Service Pack 9).

This chapter includes the following topics:

1.1 Overview

IBM Spectrum LSF Suites redefine cluster virtualization and workload management by providing a tightly integrated solution for demanding, mission-critical HPC and AI environments that can increase user productivity and hardware usage while decreasing system management costs. The heterogeneous, highly scalable and available architecture provides support for traditional high-performance computing and high throughput workloads for big data, cognitive workloads, GPU machine learning, and containerized workloads.

The capabilities of the suites are focused on the principal outcomes:

- Accelerated workloads: Advanced workload management that features policy driven scheduling that maximizes the use of computing environments for HPC and AI workloads.
- Increased productivity: Better user outcomes by way of enhanced user interfaces that are combined with policy and automation, which eliminates the need for users to become cluster experts so they can stay focused on outcomes.
- Simplified management: Easy to download and install by using Ansible, you get a fully functional cluster (often in less than an hour). Simple, central management of large, distributed systems.

IBM Spectrum LSF Suite is available in three configurations: Workgroup, HPC, and Enterprise, as shown in Figure 1-1. This document focuses on the IBM Spectrum LSF Suite for HPC.

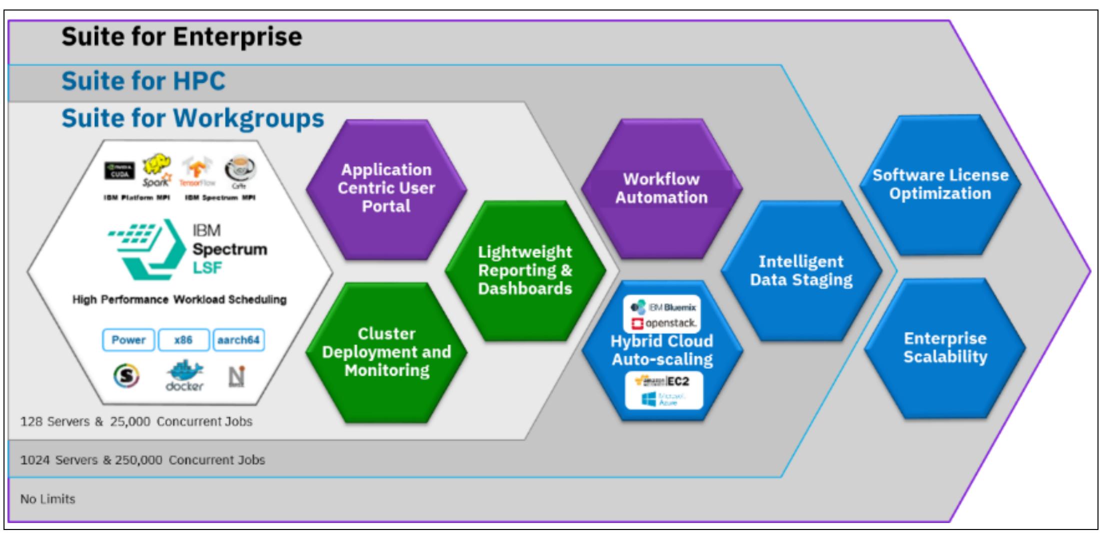


Figure 1-1 IBM Spectrum LSF Suites

A no-cost Community Edition of the suite also is available, which supports up to 10 servers.

With over 25 years of client-driven enhancements, IBM Spectrum LSF Suite is an accomplished scheduler with many flexible and extendable workload management policies. The suite offers the following key capabilities:

- GPU Scheduling: LSF provides unmatched capabilities for workload management of GPUs. From auto detection and configuration to power managements and integration with NVIDIA MPS and DCGM, which enables HPC and AI workloads.

- Container Support: LSF natively supports Docker, Singularity, Shifter, and other container technologies, which enables containerized HPC and AI workloads to be run in the same way as other workloads. Its tight integration with Docker, LSF handles all privileged operations and eliminates the need for the user to be in the Docker user group.
- Kubernetes: With Service Pack 9, LSF can now act as a central common scheduler between LSF and Kubernetes environments. This feature allows LSF a plethora of workload management policies to be applied to Kubernetes pod placement and enables Kubernetes workloads to coexist in the LSF environment. For more information, see Appendix A, “IBM Spectrum LSF and Kubernetes integration” on page 63.
- Hybrid Cloud: When your on-premises resources are insufficient, LSF supports bursting out to the cloud (AWS, Azure, GCE, IBM Cloud™, and OpenStack) by way of the Resource Connector capability. This feature enables workload-driven flexing of the cluster, which ensures that the cluster is flexed for the correct workloads at the correct time.
- Web portal: The web portal provides an application-focused experience to users. By guiding them through wizards and forms, users can be productive with little training and with significantly fewer submission errors, they are more productive. Users can spend less time monitoring their workload by using automated watchdog capabilities, and from push notifications to their browser, desktop, or mobile device by way of REST APIs.
- Delegation and compartmentalization: The suite allows administration to be delegated to project managers who can control priorities, fairshare, and membership of their projects. Information compartmentalization ensures that users are allowed only to see the appropriate information about other users and workloads in the cluster.

1.2 Selected enhancements in IBM Spectrum LSF Suite 10.2.0.9

Service Pack 9 was released in November 2019 and it includes a rollup of all fixes and enhancements in previous service packs. Full more information about the content of Service Pack 9 and its highlights, see IBM Knowledge Center.

1.2.1 Docker image affinity

When containerized workloads are run, the container images are downloaded to the execution hosts. Unlike web services, HPC containers can be gigabytes, which use significant local disk space and can take a significant time to download, which increases the overall run time of the job. Consider the following points:

- Service Pack 8 introduced the capability for the cluster administrator to view which containers were installed on which hosts, including how much space they were using.
- Service Pack 9 introduces the concept of image affinity. When enabled, LSF tries and reuses a container image, which eliminates the download overhead.

1.2.2 New example templates

Jobs submission templates make it easier for users to run complex jobs. They provide a GUI interface to simplify the process of submitting and running a workload. Service Pack 9 now includes the templates for the following products:

- Tensorflow
- Jupyter Notebooks
- H20.ai

1.2.3 IBM Spectrum LSF Explorer and Kibana

The native charting capabilities in IBM Spectrum LSF Explorer are intended for generating simple reports and visualizations. For more complex visualizations, or those visualizations that are processing a significant amount of data, we recommend Kibana. Kibana can now be installed optionally as part of the IBM Spectrum LSF Suite installation.

Service Pack 9 also upgrades the ELK stack version to 7.2.1. An example Kibana dashboard is shown in Figure 1-2.

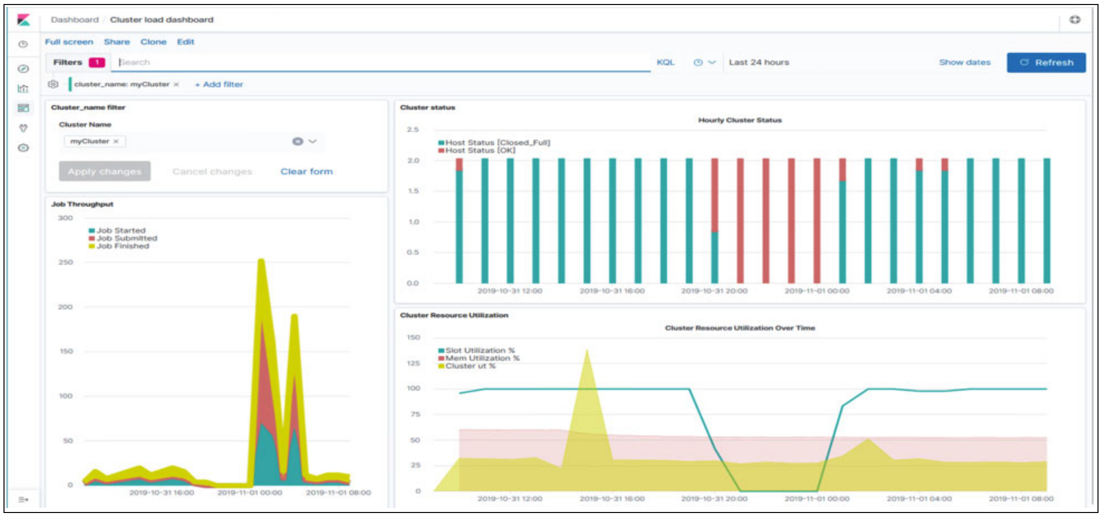


Figure 1-2 Cluster Load Dashboard

1.2.4 Azure CycleCloud Resource Connector

Service Pack 9 includes a new resource connector for Azure CycleCloud that enables LSF to burst to Azure Cyclecloud. The existing connector direct to Azure is still available.

LSF can request Cloud resources through the Resource Connector (not available is IBM Spectrum LSF Suite Workgroup Edition). The Resource Connector can request and release cloud machines based on the load in LSF. When a high demand exists, LSF can burst on to the cloud. After the workload drops, the cloud machines can be stopped.

Azure CycleCloud is a more scalable interface for requesting Azure hosts for LSF. Previous editions of LSF supported Azure CycleCloud by way of a resource connector that is posted on GitHub. For more information, see this web page.

Service Pack 9 adds an LSF-supported version of the resource connector for Azure CycleCloud. The configuration documentation is available at IBM Knowledge Center.

2

IBM Spectrum LSF Suite installation

This chapter describes the IBM Spectrum LSF Suite installation and provides best practices information to set up the environment with specific features that later cannot be changed without reinstalling the environment.

This chapter includes the following topics:

2.1 Installing IBM Spectrum LSF Suite using a high availability database

IBM Spectrum LSF Suite features several database installation options. By default, it automatically sets up a single MariaDB instance. For installations that require high availability (HA), the LSF Suite can be configured to use MariaDB with Galera.

IBM Spectrum LSF Suite can also use an external database. This section explains how to install an HA database for IBM Spectrum LSF Suite to use.

This section also explains how to install an HA MariaDB database for use with IBM Spectrum LSF Suite. This section provides the steps that are needed to get a functional configuration that uses MariaDB 10.4 with Galera. This section does not cover tuning the database.

If you use an HA database, it must be configured before installing the LSF Suite. The following prerequisites must be met:

- Three machines to host the database
- An internet connection to retrieve the packages

> **Note:** Before proceeding, it is important to remove all traces of other databases, including deleting the contents of `/var/lib/mysql`.

The installation requires several tasks:

- Prepare the database hosts for installation
- Install and configure MariaDB and Galera
- Create the database for LSF Suite
- Install LSF Suite using an external database
- Reconfigure the LSF Suite database connection

2.1.1 Preparing the database hosts for installation

Before installing the database, you must clean up any databases that were installed on these hosts. Previous installations of Mysql or MariaDB must be removed. This process can be done by running the following command on the three database hosts:

```
# yum remove mysql mysql-server mariadb mariadb-devel mariadb-libs MariaDB-server
MariaDB-client MariaDB-common galera-4
```

Older installations can also create a mysql user and group. Different databases use different UID/GIDs, which must be synchronized between the database machines. Log in to each database machine and compare the `/etc/passwd` and `/etc/group` entries for mysql; for example:

```
# grep mysql /etc/passwd
mysql:x:27:27:MariaDB Server:/var/lib/mysql:/sbin/nologin

# grep mysql /etc/group
mysql:x:27:
```

Repeat the command on the three database hosts. If needed, correct any differences in the UID and GID.

Removing the RPM packages still leaves files in `/var/lib/mysql`. Remove the following directory:

```
# rm -rf /var/lib/mysql
```

The database machines are now ready for the database installation.

2.1.2 Installing and configuring MariaDB and Galera

Complete the following steps to install MariaDB 10.4 and Galera on the database hosts and configure them in an active-active configuration:

1. Obtain the MariaDB packages.

   The easiest way is to set up a repository. Follow the instructions that are available at this web page.

2. Select version 10.4 and follow the instructions that are provided.
3. Install the database and Galera with by using the following command:

   ```
   # yum install -y MariaDB-server MariaDB-client galera-4 rsync
   ```

4. Enable the database startup at start. However, do not start it yet:

   ```
   # systemctl enable mariadb
   ```

5. You must secure the installation by running the following command:

   ```
   # mysql_secure_installation
   ```

6. Determine the host name and IP address of each of the database hosts with:

   ```
   # hostname
   # ip addr
   ```

   For the rest of the instructions, the following database hosts are used:

   – dbhost01 10.10.10.20
   – dbhost02 10.10.10.21
   – dbhost03 10.10.10.22

You substitute the host names and IP addresses with your values.

7. On each database host, edit `/etc/my.cnf.d/server.cnf`. Locate the `[galera]` section and change it as directed:

   ```
   [galera]
   # Mandatory settings
   wsrep_on=ON
   wsrep_provider=/usr/lib64/galera-4/libgalera_smm.so
   wsrep_cluster_address="gcomm://10.10.10.20,10.10.10.21,10.10.10.22"
   binlog_format=row
   default_storage_engine=InnoDB
   innodb_autoinc_lock_mode=2

   wsrep_cluster_name="galeracluster1"
   wsrep_sst_method=rsync
   #
   #Allow server to accept connections on all interfaces.
   #
   bind-address=0.0.0.0
   #
   # Optional setting
   ```

```
#wsrep_slave_threads=1
innodb_flush_log_at_trx_commit=0

# Set these lines to the name and address of the database host
# this file is on.
wsrep_node_address="10.10.10.20"
wsrep_node_name="dbhost01"
```

8. Save the file.
9. Repeat steps 1 - 8 on the three database hosts.
10. On the first database host, initialize Galera by running the following command:

```
# galera_new_cluster
```

11. Galera uses port 4567. Verify that the process is running by using the following command:

```
# lsof -i:4567
COMMAND   PID  USER   FD   TYPE    DEVICE SIZE/OFF NODE NAME
mysqld  37055 mysql   11u  IPv4 211103109      0t0  TCP *:tram (LISTEN)
```

12. If a firewall is used, you must open ports:

```
3306 TCP
4567 TCP and UDP
```

13. The Galera server is running now. Check that it is responding by using the following command:

```
# mysql -uroot -p
```

At the prompt, enter:

```
SHOW STATUS LIKE 'wsrep_cluster_size';
```

You see something that is similar to the following example:

```
MariaDB [(none)]> SHOW STATUS LIKE 'wsrep_cluster_size';
   +--------------------+--------------+
   | Variable_name      |   Value      |
   +--------------------+--------------+
   | wsrep_cluster_size |    1         |
   +--------------------+--------------+
```

14. On the other database hosts, start the mariadb service by using the following command:

```
# systemctl start mariadb
```

Note: This command can time out, but it can still be running. Check port 3306 to see whether the database is running.

15. Check that the other database hosts connected by running the following command:

```
# mysql -uroot -p
```

At the prompt, enter:

```
SHOW STATUS LIKE 'wsrep_cluster_size';
```

You see something that is similar to the following example:

```
MariaDB [(none)]> SHOW STATUS LIKE 'wsrep_cluster_size';
   +--------------------+-------+
   | Variable_name      | Value |
   +--------------------+-------+
   | wsrep_cluster_size | 3     |
   +--------------------+-------+
```

MariaDB is now configured for HA.

2.1.3 Creating the database for IBM Spectrum LSF Suite

The deployer machine is the machine that uses Ansible to install the IBM Spectrum LSF Suite cluster. It hosts the YUM repository that contains the IBM Spectrum LSF Suite rpms. It also includes the SQL scripts to start the database.

Complete the following steps to prepare the database:

1. Log in to the deployer machine and browse to `/opt/ibm/lsf_installer/DBschema/MySQL`.
2. Copy the SQL scripts in this directory to the first database host, for example:

   ```
   $ scp *.sql dbhost01:/root/
   ```

3. Log in to the first database host and browse to the directory to where the sql files were copied:

   ```
   ssh dbhost01
   cd /root
   ```

4. Log in to the database:

   ```
   # mysql -u root -p
   ```

5. Create the pac database:

   ```
   mysql> create database pac default character set utf8 default collate utf8_bin;
   ```

6. Create the `pacuser` with password `pacpass` and grant this user all privileges on the pac database:

   ```
   mysql> GRANT ALL PRIVILEGES ON pac.* to pacuser@'%' IDENTIFIED BY 'pacpass';
   ```

7. Allow the GUI hosts and deployer machines to connect to this database:

   ```
   mysql> GRANT ALL PRIVILEGES ON pac.* to pacuser@'gui_host1' IDENTIFIED BY
   'pacpass';
   mysql> GRANT ALL PRIVILEGES ON pac.* to pacuser@'gui_host2' IDENTIFIED BY
   'pacpass';
   mysql> GRANT ALL PRIVILEGES ON pac.* to pacuser@'deployer host' IDENTIFIED BY
   'pacpass';
   ```

   Note: Use your GUI host name and deployer host name.

8. Run the sql scripts to start the database:

   ```
   mysql>use pac;
   mysql>source MySQL/egodata.sql;
   mysql>source MySQL/lsfdata.sql;
   mysql>source MySQL/lsf_sql.sql;
   mysql>source MySQL/create_schema.sql;
   mysql>source MySQL/create_pac_schema.sql;
   mysql>source MySQL/init.sql;
   ```

The database is now initialized and ready for the IBM Spectrum LSF Suite installation.

2.1.4 Installing IBM Spectrum LSF Suite by using an external database

Complete the following steps to deploy IBM Spectrum LSF Suite by using the configured HA database:

1. On the deployer node, browse to `/opt/ibm/lsf_installer/playbook` and edit the `lsf-inventory` file. Set the names and roles of all the machines in the cluster.
2. Edit the `lsf-conf.yml` file. Set all the needed parameters, including the JDBC_string. Set the value to: `JDBC_string: jdbc:mariadb://{name of first DB host}:{DB port, usually 3306}/{database name}`, for example:

   ```
   JDBC_string: jdbc:mariadb://dbhost01:3306/pac
   ```

3. Set the JDBC_USER, and JDBC_PASSWORD environment variables:

   ```
   export JDBC_USER=pacuser
   export JDBC_PASSWORD=pacpass
   ```

   Note: Use the user name and password that you used when the database was created.

4. Deploy the cluster by running:

   ```
   # ansible-playbook -i lsf-inventory lsf-deploy.yml
   ```

5. Change the database connection to use the HA database connection. Log in to the first GUI host and browse to `/opt/ibm/lsfsuite/ext/perf/conf`.
6. Back up the `datasource.xml` file:

   ```
   # cp datasource.xml datasource.xml.BACKUP
   ```

7. Edit the `datasource.xml` and change the ReportDB DataSource connection string:

   ```
   <ds:DataSource Name="ReportDB"
      Driver="org.mariadb.jdbc.Driver"
      Connection="jdbc:mariadb:sequential://dbhost01,dbhost02,dbhost03/pac"
      Default="true"
   ```

8. Test the database connection string by running:

   ```
   /opt/ibm/lsfsuite/ext/perf/1.2/bin/dbconfig.sh -edit ReportDB -console

   Configure your database connection.
   Data source name: ReportDB
   ```

   When prompted, provide the pac database user ID and password:

   ```
   User ID: [pacuser]
   Password: [*******]
   ```

   You can have more than one driver. Select the jdbc driver and accept the JDBC URL and maximum connections:

   ```
   JDBC driver class name:
      0 - org.mariadb.jdbc.Driver*
      1 - org.gjt.mm.mysql.Driver
   To select an item, enter its number. Or press 2 to type a driver not in the
   list: [0]

   URL of the JDBC connection:
   [jdbc:mariadb:sequential://ma1conv03a,ma1conv03b,host87b1/pac]

   Maximum connections : [100]
   ```

a. Press “1” to test the connection.

b. Press “2” to save the information and exit.

9. Restart the associated services on all the GUI hosts:

```
# pmcadmin list
# pmcadmin stop WEBGUI
# pmcadmin stop EXPLORER
# pmcadmin stop PNC
# perfadmin stop all
# perfadmin start all
# perfadmin list
# pmcadmin start WEBGUI
# pmcadmin start EXPLORER
# pmcadmin start PNC
# pmcadmin list
```

10. Repeat steps 1 - 9 on the other GUI hosts.

11. Log in to the Application Center and submit some workload.

2.2 Multiple network installations

Production IBM Spectrum LSF Suite installations often include multiple networks, as shown in Figure 2-1.

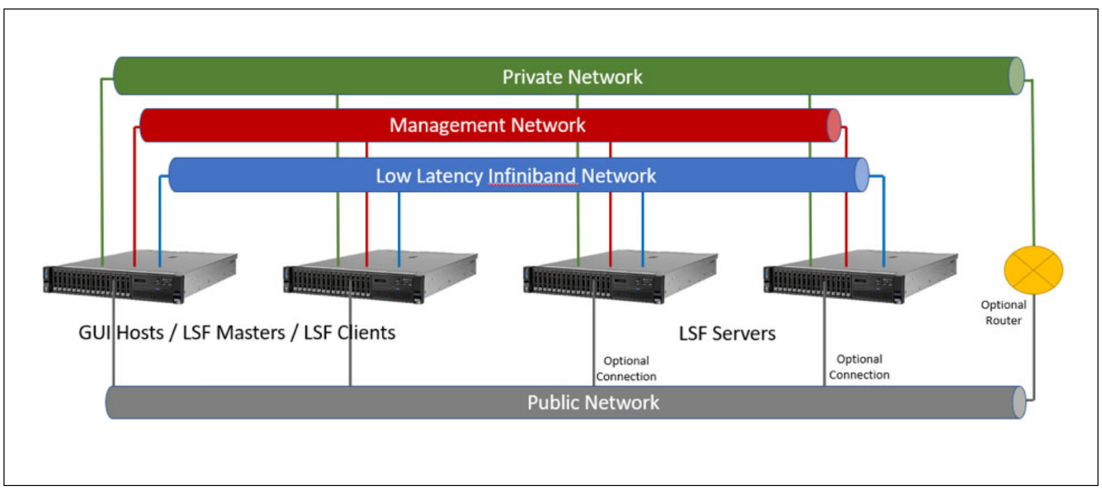


Figure 2-1 IBM Spectrum LSF Suite networking examples

Each of the networks handles the following functions:

- Management network

  An Ethernet network that handles all the hardware management functions. The Baseboard Management Consoles (BMC) use this network. The system administrators access the IPMI devices on this network.

- Low Latency InfiniBand network

  Handles the MPI traffic, or other traffic that requires high speed and low latency.

- Private network

  An Ethernet network that enables the machines to communicate with each other. Typically, this network is the network that you want LSF to use.

- Public Network

  An Ethernet network that allows users to access the LSF cluster. Other data center systems can be on this network, such as user and host name resolution services. The LSF servers optionally can be connected to this network. This network is primarily driven by the type of workloads that are run in LSF. Interactive jobs, such as Jupyter Notebooks, often to require the LSF servers be connected to this network.

Users typically interact with the Application Console on the GUI hosts, or the command-line interface (CLI) that is hosted on all the LSF machines. At a minimum, the GUI hosts and LSF clients must be routable to the Public Network. However, you do not want the LSF traffic to be routed over the Public Network, you want to limit it to the Private Network. You can control this routing at installation time by using the `lsf-inventory` and `lsf-config.yaml` files that are on the deployer node.

In the `lsf-inventory` file, you want to list the fully qualified machines names that are on the Public Network for the GUI_Hosts and LSF_Client roles. If the LSF_Masters are also the GUI_Hosts, use the fully qualified machines names that are on the Public Network. The LSF_Servers must use the machine names that are assigned to the machine network interface cards (NICs) on the Private Network, as shown in Example 2-1.

Example 2-1 Machine names on the Private Network

```
[LSF_Masters]
master1.public.network.name
master2.public.network.name

[LSF_Servers]
server1.private.network.name
server2.private.network.name

[LSF_Clients]
client1.public.network.name

[GUI_Hosts]
master1.public.network.name
master2.public.network.name
```

In the `lsf-config.yaml` file, you want to set the `Private_IPv4_Range` parameter to the CIDR range for the Private Network; for example:

```
192.168.1.0/24
172.16.0.0/16
```

After these parameters are set, IBM Spectrum LSF Suite can be installed. After IBM Spectrum LSF Suite is installed, you can check its network preference by reviewing the `/opt/ibm/lsfsuite/lsf/conf/hosts` file. This file lists all of the hosts along with their IP addresses.

2.3 Using a different UID or GID for the lsfadmin user

At installation time, IBM Spectrum LSF Suite creates a default user and group. The user and group are both called `lsfadmin`. The User ID (UID) is 495, and the Group ID (GID) is 491. (The UID and or GID might be in use by some other system user or group.)

The IBM Spectrum LSF Suite installation can use an alternative UID and GIDs if they are prepared before installing IBM Spectrum LSF Suite. The installation looks in */etc/group* for the `lsfadmin` group and looks up the `lsfadmin` user using whatever authentication is configured on the machines. If they are pre-created, that UID and GID are used.

The following procedure installs IBM Spectrum LSF Suite by using a different UID or GID. This procedure uses the IBM Spectrum LSF Suite deployer node. It has Ansible and can run commands rapidly on all nodes. This procedure can be done only before IBM Spectrum LSF Suite is installed. Attempting to change it after installation is not recommended.

Complete the following steps to change the GID:

1. Log in to the IBM Spectrum LSF Suite deployer node and change to the `/opt/ibm/lsf_installer/playbook` directory.
2. Prepare the lsf-inventory file with the list of all nodes and their roles.
3. Comment out the `[local]` role name and `localhost` lines.
4. Use Ansible to add the `lsfadmin` group by running:

   ```
   #ansible all -ilsf-inventory -m group -a “name=lsfadmin local=yes system=yes
   gid={Your GID}”
   ```

5. Verify that the `/etc/group` file includes the updated group and GID with:

   ```
   #ansible all -ilsf-inventory -m command -a “grep lsfadmin /etc/group
   ```

6. Restore the `[local]` role name and `localhost` lines in the `lsf-inventory` file.

Change the UID by creating the `lsfadmin` user in your authentication system or creating the user locally on each system.

Complete the following the steps to create locally:

1. Log in to the IBM Spectrum LSF Suite deployer node and change to the `/opt/ibm/lsf_installer/playbook` directory.
2. Prepare the `lsf-inventory` file with the list of all nodes and their roles.
3. Comment out the `[local]` role name and `localhost` lines.
4. Use Ansible to add the lsfadmin user by running:

   ```
   #ansible all -i lsf-inventory -m user -a “name=lsfadmin create_home=no
   group={Your GID} local=yes system=yes uid={Your UID}”
   ```

5. Verify that the user exists by running:

   ```
   #ansible all -i lsf-inventory -m command -a “grep lsfadmin /etc/passwd”
   ```

6.Restore the `[local]` role name and `localhost` lines in the `lsf-inventory` file.

After the user and group are set up, run the IBM Spectrum LSF Suite installation.

Note: If you deployed IBM Spectrum LSF Suite and discovered the UID or GID issue, you must use the `lsf-uninstall.yml` playbook to remove any files or directories that include the wrong UID or GID.

2.4 Debugging installation issues

This section describes installation challenges and how to solve them.

2.4.1 Managing port conflicts

Before deploying LSF Suite, it is recommended to check the environment. This check can be done by running the `lsf-predeploy-test.yml` Ansible playbook. This playbook performs various tests on the environment, including checking the following issues:

- Port conflicts
- Name resolution
- Network connectivity between LSF masters and the other roles
- Operating system repository access
- Free space

If a problem is detected, it is shown as a failed task in the summary, as shown in Figure 2-2.

```
PLAY RECAP ***************************************************************************************************
host87f4                    : ok=18    changed=0    unreachable=0    failed=0    skipped=49   rescued=0    ignored=0
host87f5                    : ok=18    changed=0    unreachable=0    failed=0    skipped=49   rescued=0    ignored=0
localhost                   : ok=2     changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
ma1hyper6                   : ok=53    changed=5    unreachable=0    failed=0    skipped=32   rescued=0    ignored=0
ma1hyper8                   : ok=15    changed=0    unreachable=0    failed=1    skipped=7    rescued=0    ignored=0
```

Figure 2-2 Failed task summary

Figure 2-2 shows that machine ma1hyper8 failed. You must determine why it failed before you can proceed with the installation. This process is done by scrolling through the output until you find the failed task. In this case, you find that a needed port is in use, as shown in Figure 2-3.

```
TASK [Set ports_in_use] **************************************************************************************
ok: [ma1hyper6]
ok: [ma1hyper8]
ok: [host87f4]
ok: [host87f5]

TASK [fail] **************************************************************************************************
fatal: [ma1hyper8]: FAILED! => {"changed": false, "msg": "On every host, a set of connection ports must be free for us
e by LSF Suite.\n"}
```

Figure 2-3 Port in use

The output did not display which port is in use. To get this information, run the command again with an extra argument, as shown in the following example:

```
# ansible-playbook -i lsf-inventory lsf-predeploy-test.yml -e show_debug=Y
```

The result of the command shows which port is in use, as shown in Figure 2-4.

```
TASK [Debug ports_in_use] *******************************************************************************************
ok: [ma1hyper6] => {
    "msg": [
        "PORTS_ARE_IN_USE=N"
    ]
}
ok: [ma1hyper8] => {
    "msg": [
        "PORTS_ARE_IN_USE=Y",
        "PORTS_IN_USE=8443"
    ]
}
ok: [host87d1] => {
    "msg": [
        "/root/.ansible/tmp/ansible-tmp-1572967496.17-108934383061012/check_ports.sh: line 11: lsof: command not found
",
        "/root/.ansible/tmp/ansible-tmp-1572967496.17-108934383061012/check_ports.sh: line 11: lsof: command not found
",
        "/root/.ansible/tmp/ansible-tmp-1572967496.17-108934383061012/check_ports.sh: line 11: lsof: command not found
",
        "PORTS_ARE_IN_USE=N"
    ]
}
```

Figure 2-4 Displays port in use

Figure 2-4 shows port 8443 is in use on ma1hyper8. You also see a secondary issue in that machine host87d1 does not include the **lsof** command; therefore, the playbook cannot determine whether a port conflict exists.

After the port is freed on the affected machine, the IBM Spectrum LSF Suite installation continues without issue.

3

IBM Spectrum LSF Suite administration: Health checks

This chapter describes IBM Spectrum LSF Suite administration scenarios to show recommended best practices.

3.1 IBM Spectrum LSF Suite Health checks

LSF provides several commands that make checking the health of the cluster easy. This section describes those commands along with test jobs that can be run to verify LSF. These tests provide a basic diagnostic flow.

3.1.1 System start

IBM Spectrum LSF Suite daemons are started by `systemd`. The role of the machine determines which services are present. Table 3-1 lists the roles and services that they run.

Table 3-1 IBM Spectrum LSF Suite daemons roles and services

Service name	LSF_Masters	LSF_Servers	LSF_Clients	GUI_Hosts
acd	No	No	No	Yes
elasticsearch-for-lsf	No	No	No	Yes
explorer	No	No	No	Yes
filebeat-for-lsf	Yes**	Yes**	No	Yes**
kibana-for-lsf	No	No	No	Yes (first host)
logstash-for-lsf	No	No	No	Yes
lsfd	Yes	Yes	No	Yes
metricbeat-for-lsf	Yes**	Yes**	No	Yes**
gpfsio-collector-for-lsf*	No*	No*	No*	No*
lsf-beat	No*	No*	No*	No*

* The lsf-beat and gpfsio-collector-for-lsf are optional.
** File monitoring is disabled; the beat services are not running.

Machines in multiple roles can include a superset of the services. The state of the service can be checked by running the **`systemctl status {`*`Service Name`*`}`** command, as shown in Example 3-1.

Example 3-1 Checking the state of the service

```
# systemctl status lsfd
lsfd.service -IBM Spectrum LSF
Loaded: loaded (/etc/systemd/system/lsfd.service; enabled; vendor preset: disabled)
Active: active (running)since Thu 2019-11-14 13:46:54 EST; 4 days ago
  Tasks: 162
Memory: 334.8M
CGroup: /system.slice/lsfd.service
            ?? 3074 /opt/ibm/lsfsuite/ext/ppm/10.2/linux2.6-glibc2.3-x86_64/etc/eauth -s[1]
            ?? 9230 /opt/ibm/lsfsuite/lsf/10.1/linux2.6-glibc2.3-x86_64/etc/dmd
            ?? 9237 /opt/ibm/lsfsuite/lsf/10.1/linux2.6-glibc2.3-x86_64/etc/eauth -s
            ??12465 /opt/ibm/lsfsuite/lsf/10.1/linux2.6-glibc2.3-x86_64/etc/mbatchd -d
/opt/ibm/...
            ??13448 /opt/ibm/lsfsuite/lsf/10.1/linux2.6-glibc2.3-x86_64/etc/eauth -s
            ??26587 /opt/ibm/lsfsuite/lsf/10.1/linux2.6-glibc2.3-x86_64/etc/lim
```

[1] The package can be used with any kernel v2.6 or later.

```
        ??26589 /opt/ibm/lsfsuite/lsf/10.1/linux2.6-glibc2.3-x86_64/etc/res
        ??26591 /opt/ibm/lsfsuite/lsf/10.1/linux2.6-glibc2.3-x86_64/etc/sbatchd
        ??27080 /opt/ibm/lsfsuite/lsf/10.1/linux2.6-glibc2.3-x86_64/etc/pim
        ??27081 /opt/ibm/lsfsuite/lsf/10.1/linux2.6-glibc2.3-x86_64/etc/pem
        ??27082 /opt/ibm/lsfsuite/lsf/10.1/linux2.6-glibc2.3-x86_64/etc/vemkd
        ??27153 /opt/ibm/lsfsuite/lsf/10.1/linux2.6-glibc2.3-x86_64/etc/melim
        ??27157 /opt/ibm/lsfsuite/lsf/10.1/linux2.6-glibc2.3-x86_64/etc/lsfbeat -c
/opt/ibm/...
        ??27171 /opt/ibm/lsfsuite/lsf/10.1/linux2.6-glibc2.3-x86_64/etc/egosc
        ??27217 /opt/ibm/lsfsuite/lsf/10.1/linux2.6-glibc2.3-x86_64/etc/pem
        ??27218 /opt/ibm/lsfsuite/ext/ppm/10.2/linux2.6-glibc2.3-x86_64/etc/jfd -2
        ??27380 /opt/ibm/lsfsuite/lsf/10.1/linux2.6-glibc2.3-x86_64/etc/mbatchd -d
/opt/ibm/...
        ??27397 /opt/ibm/lsfsuite/lsf/10.1/linux2.6-glibc2.3-x86_64/etc/mbschd
        ??27502 /opt/ibm/lsfsuite/ext/ppm/10.2/linux2.6-glibc2.3-
x86_64/etc/eem.local

Nov 14 13:46:54 ma1conv03a systemd[1]: Started IBM Spectrum LSF.
```

Example 3-1 on page 18 is from an LSF master host. You can see that all of the processes that it started, and that it is active and running.

3.1.2 Checking LSF daemons

Debugging must be started from the primary LSF master. This master is the first host that is listed in the LSF_Masters roles in the deployer's `lsf-inventory` file. Start the health checks from this machine by running the **`lsid`** command, as shown in Example 3-2.

Example 3-2 Checking the LSF master is running

```
# lsid

IBM Spectrum LSF 10.1.0.9, Nov 03 2019
Suite Edition: IBM Spectrum LSF Suite for Enterprise 10.2.0.9
Copyright International Business Machines Corp. 1992, 2016.
US Government Users Restricted Rights -Use, duplication or disclosure
restricted by GSA ADP Schedule Contract with IBM Corp.

My cluster name is myCluster
My master name is master01
```

If the command times out, the LSF master is not ready or not running. Check the service state and logs for issues. If no issues are found, the cluster name is displayed along with the primary LSF master. If the primary master is different than expected, log in to the expected primary master and check the services state and logs.

Next, check the state of the other machines. Start with the lim state by running the **`lshosts`** command, as shown in Example 3-3.

Example 3-3 Checking the state of other machines

```
# lshosts

HOST_NAME      type    model  cpuf ncpus maxmem maxswp server RESOURCES
master01     X86_64 Opteron8  60.0     4   7.6G   3.9G    Yes (mg)
```

```
master02     X86_64 Opteron8  60.0     4  15.5G   3.9G    Yes (mg)
compute01    X86_64 Intel_EM  60.0    16 255.8G      -    Yes ()
compute01    X86_64 Intel_EM  60.0    16 255.9G      -    Yes ()
gui01        X86_64 Opteron8  60.0     8   7.6G   3.9G    Yes ()
```

The HOST_NAME lists all of the machines that are listed in the `lsf.cluster.${Cluster Name}` file. The RESOURCES field shows to which resource groups the machines belong. The "`mg`" indicates that it is in the management group, which is used by the LSF masters. Next, review the lim load metrics by running the **`lsload`** command, as shown in Example 3-4.

Example 3-4 Checking the lim load metrics

```
#lsload

HOST_NAME       status r15s  r1m   r15m  ut   pg   ls   it   tmp  swp   mem
  gui01           ok   0.0   0.0   0.1   1%   0.0   0   35   29G  3.9G  14.7G
  compute01       ok   0.0   0.4   0.3   0%   0.0   1    0  344G    0M  248G
  master01        ok   0.3   1.0   1.9   47%  0.0   1    1   14G  3.9G  2.3G
  master02        ok   0.3   0.5   1.0   47%  0.0   0   52   13G  3.4G  1.2G
  compute02    unavail
```

The important field is the status. If it is `unavail`, it typically means that the lim daemon on that host is not running or is blocked by a firewall.

If you have GPUs, you can see that LSF is aware of them by running the following command:

```
#lsload -gpu
```

After you see that the lim state is OK, you can move on to reviewing the batch state by running the **`bhosts`** command, as shown in Example 3-5.

Example 3-5 Checking the batch state

```
#bhosts

HOST_NAME         STATUS     JL/U    MAX  NJOBS    RUN  SSUSP  USUSP    RSV
Master01          ok            -      1      0      0      0      0      0
master02          ok            -      1      0      0      0      0      0
gui01             ok            -      1      0      0      0      0      0
compute02         unreach       -      1      0      0      0      0      0
compute01         ok            -     16      0      0      0      0      0
```

Example 3-5 shows that compute02 is `unreach`, which means that the `sbatchd` daemon is not functioning properly in this host. Check the logs on that host. If you see that the `lsload` state is OK, check for the following potential issues:

- A firewall rule that is blocking tcp on port 6882.
- A difference between the host name the machine reports and the name that is used on the deployer.
- Multiple NICs on the machine, where the machine can connect to the LSF master over more than one network. If this issue is occurring, use the `Private_IPv4_Range` in the `lsf-config.yaml` file on the deployer node.

3.1.3 Checking basic LSF jobs

LSF is a batch scheduler. To test it, you need to submit work to the cluster.

Complete the following steps:

1. Start first from the command line by running:

   ```
   # bsub sleep 60
   User permission denied. Job not submitted.
   ```

   This command failed because you try to run a job as root. This attempt is blocked by default. You must use a regular user account to submit jobs.

2. Try using a normal user. You get:

   ```
   $ bsub sleep 60
   Job <202> is submitted to default queue <normal>.
   ```

   The command succeeds and returns the job ID. Here, the Job ID is 202 and the queue that is used is called `normal`.

3. You can try the same from the GUI by logging in to:

   `http://{GUI Host name}:8080`

   Note: Log in with the same user as before.

4. Select **Workload** from the tabs at the top (1 in Figure 3-1). Click **New Workload** (2 in Figure 3-1). New installations feature an application that is called `generic`. Choose that application (3 in Figure 3-1).

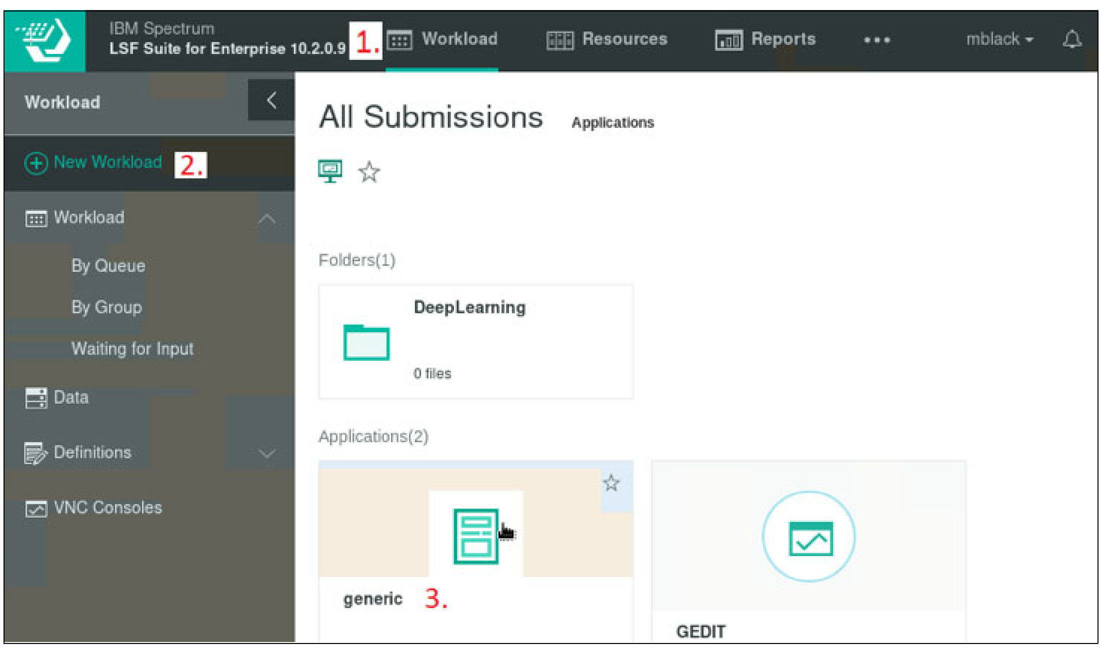


Figure 3-1 IBM Spectrum LSF job submission pane

5. In the Submission form enter the following information:
   - Command to run: **`sleep 60`**
   - Job Name: My_Test_Job

6. Click **Submit**. Then, close the window. View the workload in the GUI, as shown in Figure 3-2.

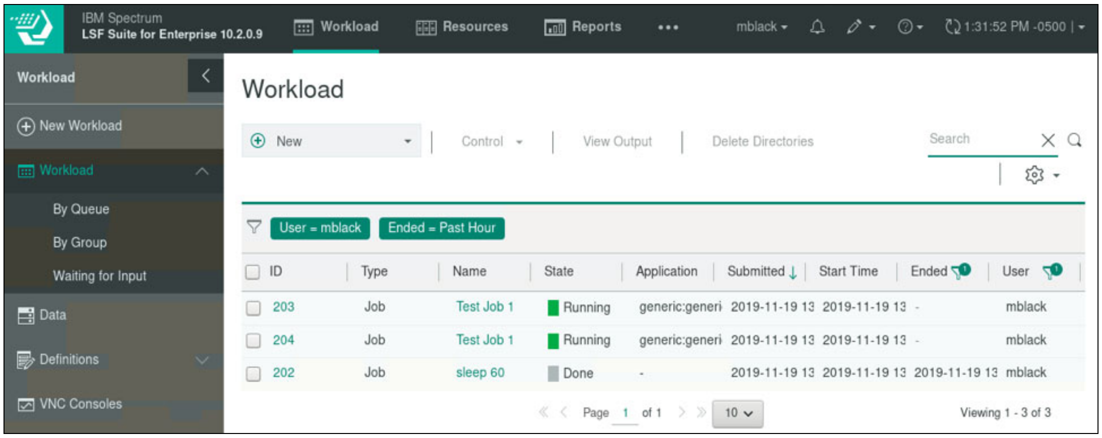


Figure 3-2 IBM Spectrum LSF Suite workload submission pane

Figure 3-2 shows the job submitted through the command line and GUI.

For more information about running various benchmarks, see Chapter 4, “Performance benchmarks” on page 25.

If you use LSF in a Kubernetes environment, more checks can be performed. For more information, see Appendix A, “IBM Spectrum LSF and Kubernetes integration” on page 63.

3.1.4 GPU checks

This section shows how to run a simple GPU job in your environment before trying the examples that are described in 5.1, “Enabling Tensorflow sample submission template for IBM Spectrum LSF Suite” on page 46.

The following prerequisites must be met before continuing:

- CUDA is installed on machines in the LSF cluster with NVIDIA GPUs.
- LSF 10.1 Fix Pack 6 or higher is used.
- Set LSF_GPU_AUTOCONFIG=Y in lsf.conf, and restart LSF.

Verify that LSF recognizes hosts with GPUs:

```
$ lshosts -gpu
HOST_NAME   gpu_id       gpu_model   gpu_driver   gpu_factor      numa_id
ac922b           0 TeslaV100_SXM2_       418.39          7.0            0
                 1 TeslaV100_SXM2_       418.39          7.0            0
                 2 TeslaV100_SXM2_       418.39          7.0            8
                 3 TeslaV100_SXM2_       418.39          7.0            8
ac922c           0 TeslaV100_SXM2_       418.39          7.0            0
                 1 TeslaV100_SXM2_       418.39          7.0            0
                 2 TeslaV100_SXM2_       418.39          7.0            8
                 3 TeslaV100_SXM2_       418.39          7.0            8
$
```

A job in which one GPU is requested and shows the nvidia-smi output is shown in the following example:

```
$ bsub -gpu "num=1" -I nvidia-smi
Job <410> is submitted to default queue <interactive>.
<<Waiting for dispatch ...>>
<<Starting on ac922b>>
Thu Jan 16 13:03:20 2020
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 418.39       Driver Version: 418.39       CUDA Version: 10.1     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  Tesla V100-SXM2...  On   | 00000004:04:00.0 Off |                    0 |
| N/A   36C    P0    37W / 300W |      0MiB / 16130MiB |      0%   E. Process |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                       GPU Memory |
|  GPU       PID   Type   Process name                             Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
$
```

A job in which two GPUs are requested and shows the nvidia-smi output is shown in the following example:

```
$ bsub -gpu "num=2" -I nvidia-smi
Job <411> is submitted to default queue <interactive>.
<<Waiting for dispatch ...>>
<<Starting on ac922b>>
Thu Jan 16 13:03:53 2020
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 418.39       Driver Version: 418.39       CUDA Version: 10.1     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|===============================+======================+======================|
|   0  Tesla V100-SXM2...  On   | 00000004:04:00.0 Off |                    0 |
| N/A   36C    P0    37W / 300W |      0MiB / 16130MiB |      0%   E. Process |
+-------------------------------+----------------------+----------------------+
|   1  Tesla V100-SXM2...  On   | 00000004:05:00.0 Off |                    0 |
| N/A   39C    P0    37W / 300W |      0MiB / 16130MiB |      0%   E. Process |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                       GPU Memory |
|  GPU       PID   Type   Process name                             Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
$
```

3.1.5 Docker checks

This section shows how to run a simple Docker job in your environment before trying the examples that are described in 5.1, "Enabling Tensorflow sample submission template for IBM Spectrum LSF Suite" on page 46.

The following prerequisites must be met before continuing:

- Docker is installed on some of the machines in the LSF cluster.
- The lsfadmin must be able to run Docker commands on each machine with Docker. For more information, see the Docker documentation about Managing Docker as a non-root user at this web page.

1. Use the instructions that are available at IBM Knowledge Center to prepare the LSF cluster to run Docker jobs:
2. Configure a Docker application profile in the `lsb.applications` file:

```
Begin Application
NAME         = ubuntu
DESCRIPTION  = Example Docker Ubuntu application
CONTAINER = docker[image(ubuntu:18.04)  \
            options(--rm --net=host --ipc=host)  \
            starter(root) ]
EXEC_DRIVER = context[user(lsfadmin)] \
              starter[LSF_SERVERDIR/docker-starter.py] \
           controller[LSF_SERVERDIR/docker-control.py] \
              monitor[LSF_SERVERDIR/etc/docker-monitor.py]
End Application
```

> **Note:** Change LSF_SERVERDIR to your specific $LSF_SERVERDIR directory location.

For more information, see IBM Knowledge Center.

3. Submitting an interactive Docker job:

```
$ bsub -I -app ubuntu cat /etc/os-release | grep 18
<<Waiting for dispatch ...>>
Job <510> is submitted to default queue <interactive>.
<<Starting on compute1>>
VERSION="18.04.3 LTS (Bionic Beaver)"
PRETTY_NAME="Ubuntu 18.04.3 LTS"
VERSION_ID="18.04"
$
```

3.1.6 Application health check with the application watchdog functionality

Although LSF provides many features for detecting and running jobs that are not running correctly, users sometimes find themselves repeatedly checking the job output to see whether the job is progressing correctly. From the outside, it appears that nothing is wrong with the job; however, looking inside can reveal that the solution is not converging, or it skipped some key optimization because a license is unavailable.

LSF supports the concept of an *Application Watchdog* where periodic application-specific automated checks can be conducted on the job to see whether it is performing as expected, which removes the need for the user to watch their jobs.

4

Performance benchmarks

High-performance computing (HPC) benchmarks are widely used to evaluate and rank system performance.

This chapter introduces some of the most commonly used benchmarks, and describes a procedure about how to conduct them by using IBM Spectrum LSF on IBM POWER9™ AC922 servers. A similar methodology can be used on x86 (including NVIDIA DGX) and arm based servers.

This chapter includes the following topics:

4.1 Highly Parallel LINPACK

Highly Parallel LINPACK (HPL) is a portable implementation of the Highly Parallel LINPACK (HPLinpack) benchmark that was written in C. It also can be regarded as a software package that generates, solves, checks, and times the solution process of a random dense linear system of equations on distributed-memory computers.

The code solves a uniformly random system of linear equations and reports time and floating-point execution rate by using a standard formula for operation count.

The benchmark today is used by scientists worldwide to evaluate computer performance, particularly for innovative advanced architecture machines.

This section documents a sample procedure for running HPL benchmarks for POWER9 AC922 systems and the considerations to be taken to ensure a successful run by way of IBM Spectrum LSF.

4.1.1 HPL (with GPU) procedure on IBM POWER9 AC922

Complete the following steps to run the HPL (with GPU) on POWER9 AC922:

1. Get NVIDIA HPL binary.

   Contact your IBM representative for more information about getting NVIDIA HPL binary.

2. Check that the following prerequisites are met:
   - Use `ldd` {hpl binary} to determine the requirements.
   - Generally ESSL, CUDA, gcc, and smpi are required.
   - In our case, we sourced smpi.sh within the LSF job.

     Other cases might include loading modules by way of `ml load` {essl, cuda, gcc or xl, smpi modules}.

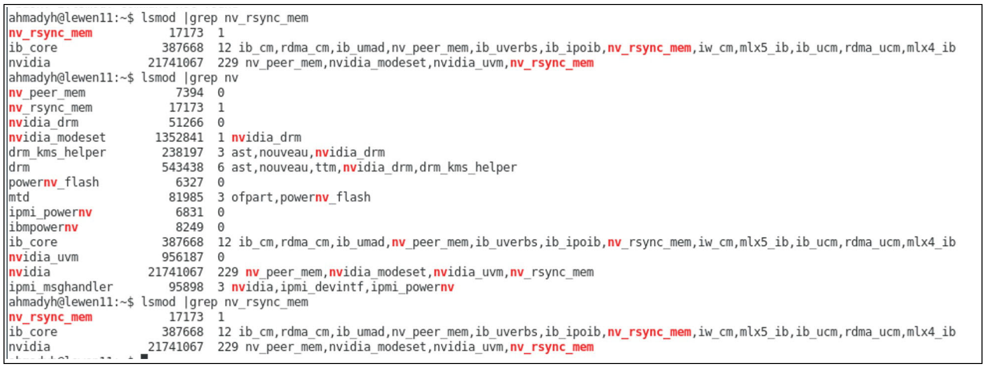
3. Check that the following prerequisites are met for POWER9 AC922:
   - CUDA toolkit 10.1 or later is installed and configured (`nvidia-persistence` is loaded).
   - Datacenter GPU manager package is installed and configured (dcgm service enabled and loaded).
   - IBM Spectrum MPI (smpi) version 10.3 is installed, which is installed automatically when you install IBM Spectrum LSF Suite 10.2.0.8.
   - If InfiniBand is used, build from source files and install `nvidia_peer_mem` and `nvidia_rsync` kernel modules (required for GPU jobs), which are provided by the `ibm_smpi_gpusupport` component of smpi. These modules enable direct data transfer to and from GPU memory to the InfiniBand network adapter.

     For more information about building and installing `nvidia_peer_memory` and the `nv_rsync_mem` RPMs, see IBM Knowledge Center.

     Note: To load `nv_rsync_mem`, you must add the following line at the end of the [Unit] block of the `/usr/lib/systemd/system/nv_rsync_mem.service`:

     ```
     Before=nvidia-persistenced.service dcgm.service
     ```

– Run the **lsmod |grep nv** command to verify that `nvidia_peer_mem` and `nvidia_rsync` kernel modules are loaded correctly, as shown in Figure 4-1.



```
ahmadyh@lewen11:~$ lsmod |grep nv_rsync_mem
nv_rsync_mem           17173  1
ib_core               387668  12 ib_cm,rdma_cm,ib_umad,nv_peer_mem,ib_uverbs,ib_ipoib,nv_rsync_mem,iw_cm,mlx5_ib,ib_ucm,rdma_ucm,mlx4_ib
nvidia              21741067  229 nv_peer_mem,nvidia_modeset,nvidia_uvm,nv_rsync_mem
ahmadyh@lewen11:~$ lsmod |grep nv
nv_peer_mem             7394  0
nv_rsync_mem           17173  1
nvidia_drm             51266  0
nvidia_modeset       1352841  1 nvidia_drm
drm_kms_helper        238197  3 ast,nouveau,nvidia_drm
drm                   543438  6 ast,nouveau,ttm,nvidia_drm,drm_kms_helper
powernv_flash           6327  0
mtd                    81985  3 ofpart,powernv_flash
ipmi_powernv            6831  0
ibmpowernv              8249  0
ib_core               387668  12 ib_cm,rdma_cm,ib_umad,nv_peer_mem,ib_uverbs,ib_ipoib,nv_rsync_mem,iw_cm,mlx5_ib,ib_ucm,rdma_ucm,mlx4_ib
nvidia_uvm            956187  0
nvidia              21741067  229 nv_peer_mem,nvidia_modeset,nvidia_uvm,nv_rsync_mem
ipmi_msghandler        95898  3 nvidia,ipmi_devintf,ipmi_powernv
ahmadyh@lewen11:~$ lsmod |grep nv_rsync_mem
nv_rsync_mem           17173  1
ib_core               387668  12 ib_cm,rdma_cm,ib_umad,nv_peer_mem,ib_uverbs,ib_ipoib,nv_rsync_mem,iw_cm,mlx5_ib,ib_ucm,rdma_ucm,mlx4_ib
nvidia              21741067  229 nv_peer_mem,nvidia_modeset,nvidia_uvm,nv_rsync_mem
```

Figure 4-1 Verification loading of ibm_smpi_gpusupport components

4. The default Parallel Active Messaging Interface (PAMI) settings for multi-host support on POWER9 systems might not be optimal for bandwidth-sensitive applications (multi-hosting is a POWER9 feature to improve off-node bandwidth).

   The settings that are shown in Example 4-1configure tasks on socket 0 to use HCA port 1 and tasks on socket 1 to use HCA port 2 (use **lstopo-no-graphics** and **ibstatus** commands to better identify wanted adapters and ports to be used).

   Example 4-1 Specifying InfiniBand devices to be used per socket

```
export PAMI_IBV_DEVICE_NAME=mlx5_0:1
export PAMI_IBV_DEVICE_NAME_1=mlx5_3:1
```

5. Build the HPL input file (HPL.dat), which requires to input values of N, P and Q.

   Where:

   – N: The order of the coefficient matrix A.
   – P: The number of process rows.
   – Q: The number of process columns.

   Choose a problem size as close as possible of the physical memory, which does not include GPU memory (85%-90% is a good starting point).

   You can decide the value of N based on the following formula:

   Memory that is used (in Bytes) = 8*N*N, N must be an integer number

Note: Consider the following points:

- The value of 8 represents double precision floating point (default usage) and a value of 4 in case of single precision.
- Memory that is used (in bytes) is dividable on NVIDIA, if the partitioning blocking factor (NB), which is in our case is 768 and must result in an integer number. N divided by NB must always produce integer number.

  For example, in our 1 TB of physical memory servers, we selected 703.125 GB (754974720000 bytes) of memory to be used for our test:

  – For 1 node test and keeping 8*N*N formula and division on NB rule in mind, N=307200

  – For 2 nodes test, N= 430080

  – For 4 nodes test, N= 614400

  – For 16 nodes test, N= 1228800

Consider the following points when you are working with Q and P:

– Q>=P

– Q must be a multiple of RANKS_PER_SOCKET.

Note: RANKS_PER_SOCKET is interpreted as the number of GPUs per socket. In current IBM POWER9 AC922 models, it can be 2 or 3, depending on the model.

– QxP equals to the number of GPUs that are used.

Note: Examples for determining values of P and Q, considering previously mentioned rules:

- 1 node with 4 GPUs, P=2 and Q=2
- 2 nodes each with 4 GPUs (8 total) P=2 and Q=4
- 4 nodes each with 4 GPUs (16 total) P=4 Q=4
- 8 nodes each with 4 GPUs (32 total) P=4 Q=8
- 16 nodes each with 4 GPUs (64 total) P=8 and Q=8

c. The content of input file (HPL.dat) resembles the example that is shown in Example 4-2 (Modify the **bold** parts per your testing needs).

Example 4-2 HPL.dat file

```
HPLinpack benchmark input file
Innovative Computing Laboratory, University of Tennessee
HPL.out      output file name (if any)
6            device out (6=stdout,7=stderr,file)
1            # of problems sizes (N)
430080          Ns # here you define the value of N
1            # of NBs
768          NBs
0            PMAP process mapping (0=Row-,1=Column-major)
1            # of process grids (P x Q)
2          Ps # here you define the value of P
4          Qs # here you define the value of Q
16.0         threshold
```

```
1            # of panel fact
2            PFACTs (0=left, 1=Crout, 2=Right)
1            # of recursive stopping criterium
4            NBMINs (>= 1)
1            # of panels in recursion
2            NDIVs
1            # of recursive panel fact.
0            RFACTs (0=left, 1=Crout, 2=Right)
1            # of broadcast
1            BCASTs (0=1rg,1=1rM,2=2rg,3=2rM,4=Lng,5=LnM)
1            # of lookahead depth
0            DEPTHs (>=0)
1            SWAP (0=bin-exch,1=long,2=mix)
192          swapping threshold
1            L1 in (0=transposed,1=no-transposed) form
0            U  in (0=transposed,1=no-transposed) form
0            Equilibration (0=no,1=yes)
8            memory alignment in double (> 0)
```

6. We chose to conduct the HPL test by way of LSF and make it responsible for resources binding, so we built a submission file (named `job-submission.sh`) that also exports all variables that are needed for HPL and includes the `xlxs` file. Example 4-3 is for a 4 GPU POWER9 AC922 with 40 cores configuration (**bold text** must be customized per your environment).

Example 4-3 IBM Spectrum LSF job submission script

```
#!/bin/bash
#BSUB -cwd /u/ahmadyh/HPL_benchmark
#BSUB -gpu "num=4:mode=exclusive_process:mps=no"
#BSUB -J HPL-8tasks
#BSUB -n 8
#BSUB -o hpl.%J.log
#BSUB -q lewen
#BSUB -R "affinity[core(10)] span[ptile=4]"
#BSUB -W "02:00"
#BSUB -x
#BSUB -m "node1 node2"
#export TEST_LOOPS=10
export USE_IRECV=1
export USE_MANAGED=0
export USE_ZERO_COPY=0
export USE_COPY_KERNEL=1
export USE_LP_GEMM=100
export PAD_LDA_ODD=0
export USE_SSEND=0
export MAX_H2D_MS=100
export MAX_D2H_MS=100
export RANKS_PER_SOCKET=2
export RANKS_PER_NODE=4
export NUM_WORK_BUF=6
#export SCHUNK_SIZE=768
#export GRID_STRIPE=6
export CHUNK_SIZE=768
export ICHUNK_SIZE=768
export SCHUNK_SIZE=768
```

```
export FACT_GEMM=1
export FACT_GEMM_MIN=64
export SORT_RANKS=0
export PRINT_SCALE=1.0
export NUM_PI_BUF=8
export NUM_L2_BUF=6
export NUM_L1_BUF=8
export NUM_WORK_BUF=6
export TEST_SYSTEM_PARAMS=0
rm -rf /dev/shm/sh_*
#export END_PROG=5.0
#export LIBC_FATAL_STDERR_=1
#export PAMI_ENABLE_STRIPING=0
export CUDA_CACHE_PATH=/tmp
# commented LSF job: export OMP_NUM_THREADS=$CPU_CORES_PER_RANK
export CUDA_DEVICE_MAX_CONNECTIONS=12
export CUDA_COPY_SPLIT_THRESHOLD_MB=1
export GPU_DGEMM_SPLIT=1.0
export TRSM_CUTOFF=9000000
export MONITOR_GPU=1
export GPU_TEMP_WARNING=70
export GPU_CLOCK_WARNING=1310
export GPU_POWER_WARNING=350
export GPU_PCIE_GEN_WARNING=3
export GPU_PCIE_WIDTH_WARNING=2
# IBM Spectrum MPI tuning and run command
ulimit -c 1
source /opt/ibm/spectrum_mpi/smpi.sh # you need to make sure all required
libraries for HPL binary are included (xl smpi cuda essl)
export OMP_WAIT_POLICY=active
export PAMI_ENABLE_STRIPING=0
export PAMI_IBV_DEBUG_PRINT_DEVICES=1
export PAMI_IBV_DEVICE_NAME="mlx5_0:1"   # IB device used by process spawned on
socket 0
export PAMI_IBV_DEVICE_NAME_1="mlx5_3:1" # IB devide used by process spawned on
socket 1
mpirun --report-bindings -prot ./xhpl_spectrum_10.1 # put the full phath of HPL
binary
# remove core files if any to avoid filing up the filesystem
rm -f core*
```

All these files (HPL input file (HPL.dat), HPL job submission (job-submission.sh), and HPL binary (xhpl_spectrum_10.1)) are in one place.

7. Create the LSF job submission output file by running run the **`bsub < job-submission.sh`** command.

   The resulting LSF output file includes the HPL output, which looks something similar to the following examples:

   – Cores binding, as shown in Figure 4-2.

```
[node1:97743] MCW rank 2 bound to socket 1[core 20[hwt 0-3]], socket 1[core 21[hwt 0-3]], socket 1[core 22[hwt 0-3]], socket 1[core 23[hwt 0-3]], socket 1[core 24[hwt 0-3]], socket 1[core 25[hwt 0-3]], socket 1[core 26[hwt 0-3]], socket 1[core 27[hwt 0-3]], socket 1[core 28[hwt 0-3]], socket 1[core 29[hwt 0-3]]:
[..../..../..../..../..../..../..../..../..../..../..../..../..../..../..../..../..../..../..../....][BBBB/BBBB/BBBB/BBBB/BBBB/BBBB/BBBB/BBBB/BBBB/BBBB/..../..../..../..../..../..../..../..../..../....]
[node1:97743] MCW rank 3 bound to socket 1[core 30[hwt 0-3]], socket 1[core 31[hwt 0-3]], socket 1[core 32[hwt 0-3]], socket 1[core 33[hwt 0-3]], socket 1[core 34[hwt 0-3]], socket 1[core 35[hwt 0-3]], socket 1[core 36[hwt 0-3]], socket 1[core 37[hwt 0-3]], socket 1[core 38[hwt 0-3]], socket 1[core 39[hwt 0-3]]:
[..../..../..../..../..../..../..../..../..../..../..../..../..../..../..../..../..../..../..../....][..../..../..../..../..../..../..../..../..../..../BBBB/BBBB/BBBB/BBBB/BBBB/BBBB/BBBB/BBBB/BBBB/BBBB]
[node1:97743] MCW rank 0 bound to socket 0[core 0[hwt 0-3]], socket 0[core 1[hwt 0-3]], socket 0[core 2[hwt 0-3]], socket 0[core 3[hwt 0-3]], socket 0[core 4[hwt 0-3]], socket 0[core 5[hwt 0-3]], socket 0[core 6[hwt 0-3]], socket 0[core 7[hwt 0-3]], socket 0[core 8[hwt 0-3]], socket 0[core 9[hwt 0-3]]:
[BBBB/BBBB/BBBB/BBBB/BBBB/BBBB/BBBB/BBBB/BBBB/BBBB/..../..../..../..../..../..../..../..../..../....][..../..../..../..../..../..../..../..../..../..../..../..../..../..../..../..../..../..../..../....]
[node1:97743] MCW rank 1 bound to socket 0[core 10[hwt 0-3]], socket 0[core 11[hwt 0-3]], socket 0[core 12[hwt 0-3]], socket 0[core 13[hwt 0-3]], socket 0[core 14[hwt 0-3]], socket 0[core 15[hwt 0-3]], socket 0[core 16[hwt 0-3]], socket 0[core 17[hwt 0-3]], socket 0[core 18[hwt 0-3]], socket 0[core 19[hwt 0-3]]:
[..../..../..../..../..../..../..../..../..../..../BBBB/BBBB/BBBB/BBBB/BBBB/BBBB/BBBB/BBBB/BBBB/BBBB][..../..../..../..../..../..../..../..../..../..../..../..../..../..../..../..../..../..../..../....]
[node2:69204] MCW rank 7 bound to socket 1[core 30[hwt 0-3]], socket 1[core 31[hwt 0-3]], socket 1[core 32[hwt 0-3]], socket 1[core 33[hwt 0-3]], socket 1[core 34[hwt 0-3]], socket 1[core 35[hwt 0-3]], socket 1[core 36[hwt 0-3]], socket 1[core 37[hwt 0-3]], socket 1[core 38[hwt 0-3]], socket 1[core 39[hwt 0-3]]:
[..../..../..../..../..../..../..../..../..../..../..../..../..../..../..../..../..../..../..../....][..../..../..../..../..../..../..../..../..../..../BBBB/BBBB/BBBB/BBBB/BBBB/BBBB/BBBB/BBBB/BBBB/BBBB]
[node2:69204] MCW rank 4 bound to socket 0[core 0[hwt 0-3]], socket 0[core 1[hwt 0-3]], socket 0[core 2[hwt 0-3]], socket 0[core 3[hwt 0-3]], socket 0[core 4[hwt 0-3]], socket 0[core 5[hwt 0-3]], socket 0[core 6[hwt 0-3]], socket 0[core 7[hwt 0-3]], socket 0[core 8[hwt 0-3]], socket 0[core 9[hwt 0-3]]:
[BBBB/BBBB/BBBB/BBBB/BBBB/BBBB/BBBB/BBBB/BBBB/BBBB/..../..../..../..../..../..../..../..../..../....][..../..../..../..../..../..../..../..../..../..../..../..../..../..../..../..../..../..../..../....]
[node2:69204] MCW rank 5 bound to socket 0[core 10[hwt 0-3]], socket 0[core 11[hwt 0-3]], socket 0[core 12[hwt 0-3]], socket 0[core 13[hwt 0-3]], socket 0[core 14[hwt 0-3]], socket 0[core 15[hwt 0-3]], socket 0[core 16[hwt 0-3]], socket 0[core 17[hwt 0-3]], socket 0[core 18[hwt 0-3]], socket 0[core 19[hwt 0-3]]:
[..../..../..../..../..../..../..../..../..../..../BBBB/BBBB/BBBB/BBBB/BBBB/BBBB/BBBB/BBBB/BBBB/BBBB][..../..../..../..../..../..../..../..../..../..../..../..../..../..../..../..../..../..../..../....]
[node2:69204] MCW rank 6 bound to socket 1[core 20[hwt 0-3]], socket 1[core 21[hwt 0-3]], socket 1[core 22[hwt 0-3]], socket 1[core 23[hwt 0-3]], socket 1[core 24[hwt 0-3]], socket 1[core 25[hwt 0-3]], socket 1[core 26[hwt 0-3]], socket 1[core 27[hwt 0-3]], socket 1[core 28[hwt 0-3]], socket 1[core 29[hwt 0-3]]:
[..../..../..../..../..../..../..../..../..../..../..../..../..../..../..../..../..../..../..../....][BBBB/BBBB/BBBB/BBBB/BBBB/BBBB/BBBB/BBBB/BBBB/BBBB/..../..../..../..../..../..../..../..../..../....]
```

Figure 4-2   Resources bindings for HPL run

– Used communication devices and protocols, as shown in Figure 4-3.

```
Choosing IB Device Port from mlx5_3:1
2: choosing device ID 0, name mlx5_3
Choosing IB Device Port from mlx5_0:1
1: choosing device ID 0, name mlx5_0
Choosing IB Device Port from mlx5_3:1
3: choosing device ID 0, name mlx5_3
Choosing IB Device Port from mlx5_0:1
0: choosing device ID 0, name mlx5_0
Choosing IB Device Port from mlx5_0:1
Choosing IB Device Port from mlx5_3:1
6: choosing device ID 0, name mlx5_3
5: choosing device ID 0, name mlx5_0
1: for device ID 0, name mlx5_0 lid: 0x28
2: for device ID 0, name mlx5_3 lid: 0x29
3: for device ID 0, name mlx5_3 lid: 0x29
Choosing IB Device Port from mlx5_3:1
Choosing IB Device Port from mlx5_0:1
7: choosing device ID 0, name mlx5_3
4: choosing device ID 0, name mlx5_0
0: for device ID 0, name mlx5_0 lid: 0x28
1: for device ID 0, name mlx5_0 lid: 0x28
2: for device ID 0, name mlx5_3 lid: 0x29
3: for device ID 0, name mlx5_3 lid: 0x29
6: for device ID 0, name mlx5_3 lid: 0x2d
0: for device ID 0, name mlx5_0 lid: 0x28
5: for device ID 0, name mlx5_0 lid: 0x2c
4: for device ID 0, name mlx5_0 lid: 0x2c
7: for device ID 0, name mlx5_3 lid: 0x2d
4: for device ID 0, name mlx5_0 lid: 0x2c
5: for device ID 0, name mlx5_0 lid: 0x2c
6: for device ID 0, name mlx5_3 lid: 0x2d
7: for device ID 0, name mlx5_3 lid: 0x2d
Host 0 [node1] ranks 0 - 3
Host 1 [node2] ranks 4 - 7

 host | 0    1
======|===========
    0 : pami pami
    1 : pami pami

Connection summary:
  on-host:  all connections are pami
  off-host: all connections are pami
```

Figure 4-3 InfiniBand devices and protocols that are used for HPL run

– HPL iterations and results, as shown in Figure 4-4. The results achieved 4.691 TFlops for two nodes, which is approximately 79.1% efficiency for our specific cluster and configuration. Better numbers can be achieved by way of further configuration tuning.

```
..........

540        ** FACT GEMM: 0.000381199 s 366.707 GF        (without delay:      0.000351296 s        397.922 GF      7584      96       96)
540        ** FACT GEMM: 0.000753022 s 733.146 GF        (without delay:      0.000724992 s        761.492 GF      7488     192      192)
540        ** FACT GEMM: 0.000345687 s 394.141 GF        (without delay:      0.000316416 s        430.602 GF      7392      96       96)
540        ** FACT GEMM: 0.00155054 s 1387.69 GF         (without delay:      0.00151552 s         1419.76 GF      7296     384      384)
540        ** FACT GEMM: 0.000335861 s 395.135 GF        (without delay:      0.000303072 s        437.884 GF      7200      96       96)
540        ** FACT GEMM: 0.000689847 s 759.246 GF        (without delay:      0.000659456 s        794.236 GF      7104     192      192)
540        ** FACT GEMM: 0.000322802 s 400.157 GF        (without delay:      0.00029184 s         442.611 GF      7008      96       96)
!!!! WARNING: Rank: 0 : node1 : GPU 0004:04:00.0 [0;31mClock: 1290 MHz [0m      Temp: 49 C  Power: 58 W          PCIe  gen 3          x2
!!!! WARNING: Rank: 4 : node2 : GPU 0004:04:00.0 [0;31mClock: 1290 MHz [0m      Temp: 55 C  Power: 65 W          PCIe  gen 3          x2
!!!! WARNING: Rank: 1 : node1 : GPU 0004:05:00.0 [0;31mClock: 1290 MHz [0m      Temp: 55 C  Power: 109 W         PCIe  gen 3          x2

556        ** FACT GEMM: 0.000201741 s 78.9391 GF        (without delay:      0.000180224 s        88.3636 GF       864      96       96)
[0;31m Prog= 100.00%    N_left= 768 Time= 1128.20        Time_left= 0.00      iGF= 10308.95        GF= 47008.01          iGF_per= 1288.62
           GF_per= 5876.00 [0m

================================================================================
T/V                N    NB     P     Q               Time                 Gflops
--------------------------------------------------------------------------------
WR01L2R4      430080   768     2     4            1130.44              4.691e+04
--------------------------------------------------------------------------------
||Ax-b||_oo/(eps*(||A||_oo*||x||_oo+||b||_oo)*N)=     1.033969e-06 ...... PASSED
================================================================================

Finished      1 tests with the following results:
              1 tests completed and passed residual checks,
              0 tests completed and failed residual checks,
              0 tests skipped because of illegal input values.
--------------------------------------------------------------------------------

End of Tests.
================================================================================
```

Figure 4-4 HPL results sample output

Important: Consider the following points:

- Conduct HPL runs and calculate efficiency in scalable format; for example, start runs on single node, then 2, 4, 8, and so on. The best case is when you achieve consistent efficiency as you scale up.
- Always repeat HPL runs on the same scenario (for example, same nodes) to ensure that performance and efficiency is consistent across runs.

4.2 OSU micro-benchmarks

The OSU micro-benchmarks (OMB) feature a series of MPI benchmarks that measure the performances of various MPI operations, including the following examples:

- Point-to-Point MPI Benchmarks
- Collective MPI Benchmarks
- One-sided MPI Benchmarks
- Startup Benchmarks

Note: For more information about all OMB latest benchmarks, see this web page.

The objective of this section is to document a procedure for compiling and running some OSU micro-benchmarks on the IBM Power9 AC922 set-up that used SMPI 10.3 and IBM Spectrum LSF Suite for HPC 10.2.0.8.

We focus on the following point-to-point MPI benchmarks:

- `osu_latency`: Latency Test

  The latency tests are conducted in a ping-pong fashion. The sender sends a message with a specific data size to the receiver and waits for a reply from the receiver. The receiver receives the message from the sender and sends back a reply with the same data size.

Many iterations of this ping-pong test are conducted and average one-way latency numbers are obtained. Blocking version of MPI functions (MPI_Send and MPI_Recv) are used in the tests.

- `osu_bibw`: Bidirectional Bandwidth Test

  The bidirectional bandwidth test is similar to the bandwidth test, except that both of the nodes that are involved send out a fixed number of back-to-back messages and wait for the reply. This test measures the maximum sustainable aggregate bandwidth by two nodes.

These tests and many others were extended to evaluate performance of MPI communication from and to buffers on NVIDIA GPU devices.

4.2.1 OMB procedure on IBM POWER9 AC922

Complete the following steps to run the OMBs on POWER9 AC922:

1. Get the latest OMB `.tar` file from this web page.
2. Check the compilation prerequisites that are available (exported and sourced):
   - GNU Compiler Collection (gcc, g++ and gfortran). We used version 7.4 in our exercise.
   - IBM Spectrum MPI
   - IBM ESSL
   - CUDA
3. Extract the downloaded `.tar` file. We used osu-micro-benchmarks version 5.6.2.
4. Within the extracted folder (namely osu-micro-benchmarks-5.6.2), complete the following steps:
   a. Ensure that the correct environment is exported and sourced, as shown in Example 4-4.

Example 4-4 Environment settings before OMB compilation

```
ml load cuda xl gcc essl smpi # our lab specific to make sure all required
libraries are loaded and declared, others might only require to source
/opt/ibm/spectrum_mpi/smpi.sh
export OMPI_CC=gcc
export OMPI_FC=gfortran
export OMPI_CXX=g++
export OMB_ROOT=[full path to osu-micro-benchmarks-5.6.2 folder}
export OMB_MPI_ROOT=$OMB_ROOT/mpi
export PATH=${PATH}:$OMB_MPI_ROOT/collective
export PATH=${PATH}:$OMB_MPI_ROOT/one-sided
export PATH=${PATH}:$OMB_MPI_ROOT/pt2pt
export PATH=${PATH}:$OMB_MPI_ROOT/startup
export PATH=${PATH}:{cuda bin path}:{cuda include path}:{cuda lib64 path}
export OMP_WAIT_POLICY=active
export PAMI_ENABLE_STRIPING=0
export PAMI_IBV_ADAPTER_AFFINITY=1
export PAMI_IBV_DEBUG_PRINT_DEVICES=1
export PAMI_IBV_DEVICE_NAME="mlx5_0:1"   # IB device used by process spawned
on socket 0
export PAMI_IBV_DEVICE_NAME_1="mlx5_3:1" # IB devide used by process spawned
on socket 1
```

b. Because GPUs are used, we configured the microbenchmarks by using the following commands:

- `./configure CC=/products/spectrum_mpi/10.03.00.01rtm3/bin/mpicc CXX=/products/spectrum_mpi/10.03.00.01rtm3/bin/mpicxx --enable-cuda --with-cuda-include=/products/cuda/10.1.168/targets/ppc64le-linux/include --with-cuda-libpath=/products/cuda/10.1.168/targets/ppc64le-linux/lib`

> **Note:** The following syntax is used:
>
> `./configure CC={path to mpicc} CXX={path to mpicxx} --enable-cuda --with-cuda-include={path to cuda include} --with-cuda-libpath={path to cuda lib}`

- `make` (or `make -j` for parallel make)
- `make install`

At this point, all benchmarks binary must be ready within $OMB_MPI_ROOT.

c. Check that all required libraries are available by using the **`ldd`** command on one of the compiled tests. Figure 4-5 shows the availability of all libraries that are required for `osu_bw` benchmark test.

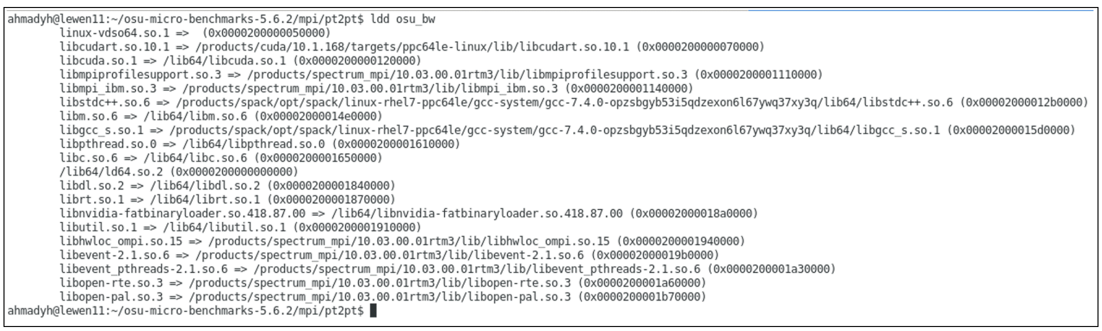


Figure 4-5 ldd osu_bw command output

d. We use LSF to submit all of our osu benchmarks immediately to make IBM Spectrum LSF handle resources management. Therefore, we created a submission script and named it `osu_lsf_job_submission.sh`. This script resembles the example that is shown in Example 4-5 (**bold text** must be customized per your setup).

Example 4-5 osu sample submission file by way of LSF

```
#!/bin/bash
#BSUB -cwd /u/ahmadyh/osu-micro-benchmarks-5.6.2/lsf_integration
#BSUB -gpu "num=1:mode=exclusive_process:mps=no"
#BSUB -J OMB
#BSUB -n 2
#BSUB -o job.%J.log
#BSUB -q lewen
#BSUB -R "affinity[core(1):distribute=balance] span[ptile=1]"
#BSUB -x
ml purge
ml load cuda xl gcc essl smpi # our lab specific to make sure all required
libraries are loaded and declared
ml list
export OMPI_CC=gcc
export OMPI_FC=gfortran
```

```
export OMPI_CXX=g++
export OMB_ROOT=/u/ahmadyh/osu-micro-benchmarks-5.6.2
export OMB_MPI_ROOT=$OMB_ROOT/mpi
export PATH=${PATH}:$OMB_MPI_ROOT/collective
export PATH=${PATH}:$OMB_MPI_ROOT/one-sided
export PATH=${PATH}:$OMB_MPI_ROOT/pt2pt
export PATH=${PATH}:$OMB_MPI_ROOT/startup
export OMP_WAIT_POLICY=active
export PAMI_ENABLE_STRIPING=1 #
export PAMI_IBV_ADAPTER_AFFINITY=0 #
export PAMI_IBV_DEBUG_PRINT_DEVICES=1
export PAMI_IBV_DEVICE_NAME="mlx5_0:1,mlx5_3:1"   # IB device used by
process spawned on socket 0
export PAMI_IBV_DEVICE_NAME_1="mlx5_3:1,mlx5_0:1" # IB devide used by
process spawned on socket 1
#bandwidth
mpirun --report-bindings -prot osu_bibw #host to host OSU MPI bi-directional
bandwidth test
mpirun -gpu --report-bindings -prot osu_bibw D D #OSU MPI-CUDA
Bi-Directional Bandwidth Test
# Latency
mpirun --report-bindings -prot osu_latency #OSU MPI Latency Test
mpirun -gpu --report-bindings -prot osu_latency D D #OSU MPI_CUDA Latency
Test
```

e. Create an OSU MPI bidirectional bandwidth test output by running the **bsub < osu_lsf_job_submission.sh** command.

The resulting output job file included the following results:

– OSU MPI bidirectional bandwidth test of 49213.55 MBps. Figure 4-6 shows the performance results of **`mpirun --report-bindings -prot osu_bibw`**, which was submitted in the job file.

```
 host | 0    1
======|===========
    0 : pami pami
    1 : pami pami

Connection summary:
  on-host:  all connections are pami
  off-host: all connections are pami

# OSU MPI Bi-Directional Bandwidth Test v5.6.2
# Size      Bandwidth (MB/s)
1                       4.47
2                       9.60
4                      15.65
8                      32.56
16                     62.52
32                    127.08
64                    261.44
128                   466.67
256                   690.86
512                  1228.98
1024                 2093.00
2048                 4069.95
4096                 6585.63
8192                 9042.61
16384               10224.32
32768               12087.40
65536               12601.05
131072              44433.04
262144              46915.74
524288              47904.36
1048576             48656.73
2097152             48994.61
4194304             49213.55
```

Figure 4-6 Results of OSU MPI bidirectional bandwidth test

– OSU MPI-CUDA bidirectional Bandwidth Test of 35942.10 MBps. Figure 4-7 shows the performance results of **mpirun -gpu --report-bindings -prot osu_bibw D D**, which was submitted in the job file.

```
 host | 0    1
======|===========
    0 : pami pami
    1 : pami pami

Connection summary:
  on-host:  all connections are pami
  off-host: all connections are pami

# OSU MPI-CUDA Bi-Directional Bandwidth Test v5.6.2
# Send Buffer on DEVICE (D) and Receive Buffer on DEVICE (D)
# Size      Bandwidth (MB/s)
1                       1.45
2                       3.00
4                       5.98
8                      11.95
16                     24.03
32                     46.73
64                     95.95
128                   189.82
256                   376.06
512                   756.96
1024                 1493.24
2048                 2541.54
4096                 5083.09
8192                 9677.62
16384               16802.79
32768               21657.04
65536               24096.18
131072              25812.82
262144              26868.16
524288              20897.66
1048576             21431.91
2097152             35908.06
4194304             35942.10
```

Figure 4-7 Results of OSU MPI-CUDA bidirectional Bandwidth Test

– OSU MPI Latency Test of 1.27 µs. Figure 4-8 shows the performance results of **mpirun --report-bindings -prot osu_latency**, which was submitted in the job file.

```
 host | 0    1
======|===========
    0 : pami pami
    1 : pami pami

Connection summary:
  on-host:  all connections are pami
  off-host: all connections are pami

# OSU MPI Latency Test v5.6.2
# Size          Latency (us)
0                       1.27
1                       1.24
2                       1.22
4                       1.21
8                       1.22
16                      1.23
32                      1.30
64                      1.34
128                     1.41
256                     1.76
512                     1.88
1024                    2.45
2048                    2.96
4096                    3.58
8192                    4.97
16384                   6.92
32768                   9.26
65536                  12.11
131072                 14.13
262144                 19.57
524288                 30.39
1048576                56.04
2097152                94.22
4194304               178.90
```

Figure 4-8 Results of OSU MPI Latency Test

– OSU MPI_CUDA Latency Test of 6.39 µs. Figure 4-9 shows the performance results of **`mpirun -gpu --report-bindings -prot osu_latency D D`**, which was submitted in the job file.

> **Note:** We selected the second reading because the first reading is from CPUs, not GPUs.

```
 host | 0    1
======|===========
    0 : pami pami
    1 : pami pami

Connection summary:
  on-host:  all connections are pami
  off-host: all connections are pami

# OSU MPI-CUDA Latency Test v5.6.2
# Send Buffer on DEVICE (D) and Receive Buffer on DEVICE (D)
# Size          Latency (us)
0                       1.25
1                       6.39
2                       6.41
4                       6.40
8                       6.36
16                      7.84
32                      7.04
64                      6.75
128                     6.72
256                     6.67
512                     6.68
1024                    6.87
2048                    8.18
4096                    8.00
8192                    8.36
16384                   9.62
32768                  11.37
65536                  14.34
131072                 18.32
262144                 28.15
524288                107.94
1048576               172.68
2097152               214.39
4194304               302.28
```

Figure 4-9 Results of OSU MPI_CUDA latency test

> **Important:** Ensure that your OMB runs cover all any node-to-any node scenarios in your HPC cluster.

4.3 STREAM benchmark

The STREAM benchmark is a simple synthetic benchmark program that measures sustainable memory bandwidth (in MBps) and the corresponding computation rate for simple vector kernels.

It is designed to work with datasets that are much larger than the available cache on any system. Because of this feature, the results are more indicative of the performance of large, vector style applications.

The next section documents a procedure for compiling and running the STREAM benchmark on IBM POWER9 AC922.

4.3.1 STREAM benchmark procedure on IBM POWER9 AC922

Complete the following steps:

1. Get `stream.c` from this website.
2. Determine the array size that you want to use to compile your `stream.c` against. In our case, we selected an array size of 900000000.

   > **Note:** Consider the following points:
   >
   > - Selected Array_size determines memory size that is used in the test:
   >
   >   `Memory per Array = 8 (bytes) * Array_size`
   >
   > - Memory size must be at least 10 times the total cache size for all cores. The **`lstopo-no-graphics`** command can be used to help understand your system’s cache size.

3. We used GCC 7.4 and compiled by using the following syntax:

   ```
   gcc -m64 -O3 -mcpu=power9 -mtune=power9 -mcmodel=large -fopenmp
   -DSTREAM_ARRAY_SIZE= 900000000 -DNTIMES=10 stream.c -o stream.gcc74
   ```

   As indicated, the resulting binary file was named *`stream.gcc74`*.
4. Validate the availability of the required library files by using the **`ldd stream.gcc74`** command, as shown in Figure 4-10.

```
ahmadyh@lewen11:~/stream_benchmark$ ldd stream.gcc74
        linux-vdso64.so.1 =>  (0x0000200000050000)
        libgomp.so.1 => /products/spack/opt/spack/linux-rhel7-ppc64le/gcc-system/gcc-7.4.0-opzsbgyb53i5qdzexon6l67ywq37xy3q/lib64/libgomp.so.1 (0x0000200000070000)
        libpthread.so.0 => /lib64/libpthread.so.0 (0x00002000000e0000)
        libc.so.6 => /lib64/libc.so.6 (0x0000200000120000)
        libdl.so.2 => /lib64/libdl.so.2 (0x0000200000310000)
        /lib64/ld64.so.2 (0x0000200000000000)
```

Figure 4-10 ldd command output for the compiled stream binary file

5. If you want to run the benchmark directly, use export **`OMP_NUM_THREADS=`** (specifies number of threads that are used for the test, usually equals to number of cores).

   We ran the binary **`./stream.gcc74`**, and the output is shown in Figure 4-11 on page 42, which resulted “Triad” of 251994.9 MBps.

```
-------------------------------------------------------------
STREAM version $Revision: 5.10 $
-------------------------------------------------------------
This system uses 8 bytes per array element.
-------------------------------------------------------------
Array size = 900000000 (elements), Offset = 0 (elements)
Memory per array = 6866.5 MiB (= 6.7 GiB).
Total memory required = 20599.4 MiB (= 20.1 GiB).
Each kernel will be executed 10 times.
 The *best* time for each kernel (excluding the first iteration)
 will be used to compute the reported bandwidth.
-------------------------------------------------------------
Number of Threads requested = 144
Number of Threads counted = 144
-------------------------------------------------------------
Your clock granularity/precision appears to be 1 microseconds.
Each test below will take on the order of 56292 microseconds.
   (= 56292 clock ticks)
Increase the size of the arrays if this shows that
you are not getting at least 20 clock ticks per test.
-------------------------------------------------------------
WARNING -- The above is only a rough guideline.
For best results, please be sure you know the
precision of your system timer.
-------------------------------------------------------------
Function    Best Rate MB/s  Avg time     Min time     Max time
Copy:          218383.0     0.069149     0.065939     0.071854
Scale:         225846.8     0.071109     0.063760     0.113808
Add:           251420.1     0.087497     0.085912     0.090779
Triad:         251994.9     0.087169     0.085716     0.092365
-------------------------------------------------------------
Solution Validates: avg error less than 1.000000e-13 on all three arrays
-------------------------------------------------------------
```

Figure 4-11 Results of STREAM benchmark in our lab

Note: In STREAM benchmark results:

- Copy function measures transfer rates in the absence of arithmetic.
- Scale function adds a simple arithmetic operation.
- Sum function adds a third operand to allow multiple load and store ports on vector machines to be tested.
- Triad function allows chained, overlapped, fused, multiply, and add operations.

6. Prepare a STREAM job submission file (named `stream.lsf.sh`) by using IBM Spectrum LSF, as shown in Example 4-6 (**bold text** must be customized per your environment).

Example 4-6 Sample STREAM job submission file by way of LSF

```
#!/bin/bash
#BSUB -cwd /u/ahmadyh/stream_benchmark
#BSUB -a opemp
#BSUB -env all
#BSUB -J Stream
#BSUB -n 1
#BSUB -m lewen11 #define the machine you want to conduct STREAM benchmark
against
#BSUB -o job.%J.log
#BSUB -q lewen
```

```
#BSUB -R "affinity[core(36):distribute=balance] span[ptile=1]" # our machine
has 36 cores
ml load gcc #this is to load all stream compiled file (stream.gcc74) required
libraries, this is determined via ldd command
export OMP_DISPLAY_ENV=True
/u/ahmadyh/stream_benchmark/stream.gcc74
```

You might need to **unset OMP_NUM_THREADS**. The number of threads is determined by using the lsf job resource requirements, which is 36, as shown in Example 4-6 on page 42.

7. Submit the stream lsf job by using the **bsub < stream.lsf.sh** command.

Important: Make sure that you run the STREAM benchmark on each of the nodes that are in your HPC cluster and the results are per expected optimal performance.

5

IBM Spectrum LSF application configuration scenarios

This chapter provides configuration case scenarios and includes the following topics:

5.1 Enabling Tensorflow sample submission template for IBM Spectrum LSF Suite

This section describes how to set up Tensorflow job submission templates in IBM Spectrum LSF Suite. This section also explains how to enable the following templates:

- MNIST_Training
- Classify_Image
- Classify_Directory_Of_Images
- Retrain_Model
- Tensorboard

The original source code and general instruction are available at this web page.

This template uses pre-packaged containers of Tensorflow and Tensorboard and LSFs Docker integration to start the jobs. The following prerequisites must be met before continuing:

- Docker is installed on some of the machines in the cluster.
- A shared directory to host the data is available on all Docker machines.
- The Docker images are on an internal registry or local cache.
- The lsfadmin user can access Docker.
- Internet access is available for the Docker machines that run the MNIST_Training to download the training set.

Complete the following steps to complete the configuration:

1. Use the instructions at this web page to prepare the LSF cluster to run Docker jobs.

   > **Note:** Red Hat Docker packages allow root access to `/var/run/docker.sock` only. You must use a group or SELinux to grant lsfadmin access.

2. A shared directory is needed between the machines. This directory is used for sample data and scripts that are needed. This directory must be writable by the lsfadmin user. Modify the directory permissions for lsfadmin to read and write to this directory, which is the MLDL_TOP directory for the rest of this configuration.
3. Pull the Docker image from the dockerhub; for example:

   ```
   # docker pull tensorflow/tensorflow:1.10.0
   # docker pull ibmcom/tensorflow-ppc64le:1.13.1   (for POWER)
   ```

4. If an internal registry does not exist or the Docker machines cannot access the internet, load the image on the Docker machines; for example:

   ```
   $ docker save {Image name} -o tensorflow.img
   ```

   a. Copy image to next Docker machine, and on the next Docker machine run:

      ```
      $ docker load -I tensorflow.img
      ```

   b. Repeat until all Docker machines have the image.

5. Log in to the Application Center as lsfadmin. The lsfadmin can edit the templates and LSF configuration files. Do not use another user.

   > **Note:** The default lsfadmin password is lsfadmin. This password must be changed.

6. Browse to the DeepLearning workload templates directory. Click **Workload** → **Definitions** → **Templates** → **DeepLearning**, as shown in Figure 5-1.

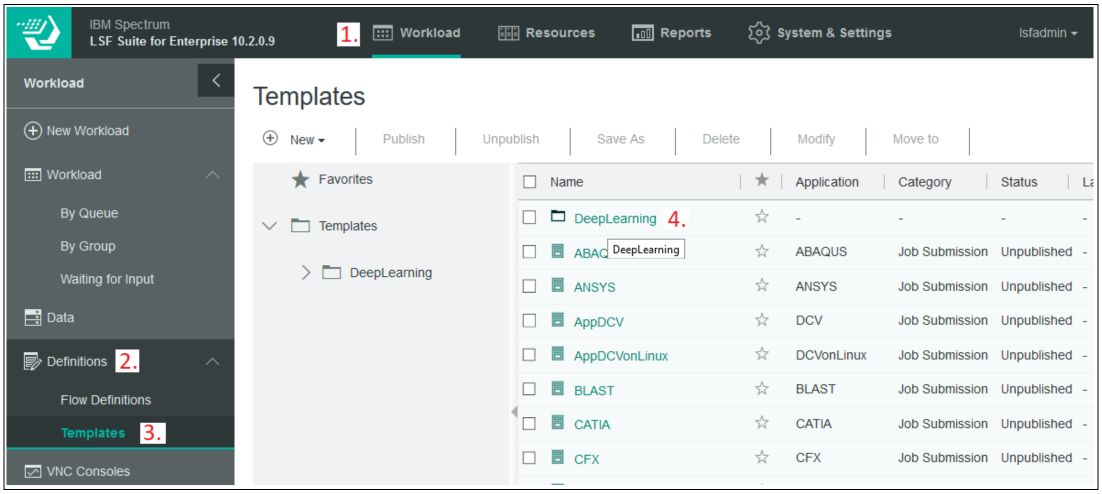


Figure 5-1 DeepLearning workload templates

7. In the DeepLearning directory, select **Tensorboard** and click **Publish**. You are prompted to select a directory into which the tutorial is downloaded (the MLDL_TOP directory).

8. After a few minutes, check the contents of the directory. You see the following directories:
   – imagenet
   – images
   – retain
   – scripts

9. If the directories are not present, check the logs in `/opt/ibm/lsfsuite/ext/gui/logs/{Name of GUI Host}/Tensorboard.publish.log`. Correct any errors. Then, complete the following steps to retrigger the publish script:

   a. Unpublish the application.

   b. Go to the GUI host and browse to:

   ```
   /opt/ibm/lsfsuite/ext/gui/conf/application/draft/Tensorboard
   ```

   c. Reset Tensorboard.cmd.org back to Tensorboard.cmd:

   ```
   mv Tensorboard.cmd.org Tensorboard.cmd
   ```

   d. Publish the Tensorboard application and check the log file to make sure that the error is corrected.

10. Check that the Docker Tensorflow application was created:

```
$ bapp -l
APPLICATION NAME: docker_tensorflow
 -- Example Docker Tensorflow application
```

11. The integration script assumes that lsfadmin can write to the job submission users home directory, which might not be true for your installation. This issue can be fixed by modifying the `dockerPasswd.sh script` that is in the MLDL_TOP `/scripts` directory. Modify it as follows:

```
#JOBTMPDIR=$LS_EXECCWD # Assume the job's current working directory is shared
for parallel jobs
#if [ "x$JOBTMPDIR" = "x" ] ; then
#echo "Are you testing outside of an LSF job?"
```

```
#JOBTMPDIR=/tmp/$USER
#mkdir $JOBTMPDIR
#fi
MLDL_TOP=/nfs/mldl
mkdir ${MLDL_TOP}/tmp 2>&1
if [ ! -d ${MLDL_TOP}/tmp/${USER} ]; then
    mkdir ${MLDL_TOP}/tmp/${USER}
fi
JOBTMPDIR=${MLDL_TOP}/tmp/${USER}
```

12. Log out of the Application Center and log in as a regular user.
13. In the GUI, click **Workload** → **New Workload**. In the folders, select the **DeepLearning** folder and click **Tensorboard**.
14. Enter a job name and the MLDL_TOP value. Then, click **Submit**.
15. When the job is running, click the job link icon (see Figure 5-2) that is next to the job ID to start Tensorboard GUI.

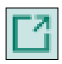


Figure 5-2 Job link icon

Other Tensorflow job submission templates are included in LSF Suite 10.2.0.9 as follows:

- MNIST_Training
- Classify_Image
- Classify_Directory_Of_Images
- Retrain_Model

They use the same scripts and MLDL_TOP directory structure. As the lsfadmin user, publish them the same way as described in step 7.

These examples show how to start Tensorflow jobs. They also provide samples that can be taken and modified to suit your needs.

5.2 OpenFOAM sample submission template for IBM Spectrum LSF Suite

This section shows how to set up the OpenFOAM submission template in IBM Spectrum LSF Suite, and how to set up ParaView for visualization of the OpenFOAM results. This section also explains how to enable the following templates:

- OpenFOAM
- ParaView

The following prerequisites must be met before continuing:

- LSF Suite 10.2 Fix Pack 9 or higher.
- A shared directory to host the data is available on all Docker machines.
- Follow 3.1.5, “Docker checks” on page 24 to verify that Docker is working with LSF.

The source code and general instructions are available at the following links:

- GitHub web page for OpenFOAM
- GitHub web page for ParaView

After reading the README.md that is found in the first URL (GitHub web page for OpenFOAM), complete the following steps to finalize the OpenFOAM configuration:

1. A shared directory is required between the machines. This directory is used for sample data and scripts. This directory must be writable by the lsfadmin user. Modify the directory permissions for lsfadmin to read and write to this directory. This is the JOB_REPOSITORY_TOP directory for the rest of the paper and be sure to replace JOB_REPOSITORY_TOP with the actual JOB_REPOSITORY_TOP directory in your environment.
2. Pull the Docker image from dockerhub or build a custom OpenFOAM container image:
   a. For x86_64 only:
   ```
   # docker pull openfoam/openfoam6-paraview54
   ```
   b. For x86_64 or IBM POWER, use the steps in this blog to build a custom OpenFOAM container image.
3. Submit an OpenFOAM job by using LSF Application Center with OpenFOAM-v1912, as shown in Figure 5-3.

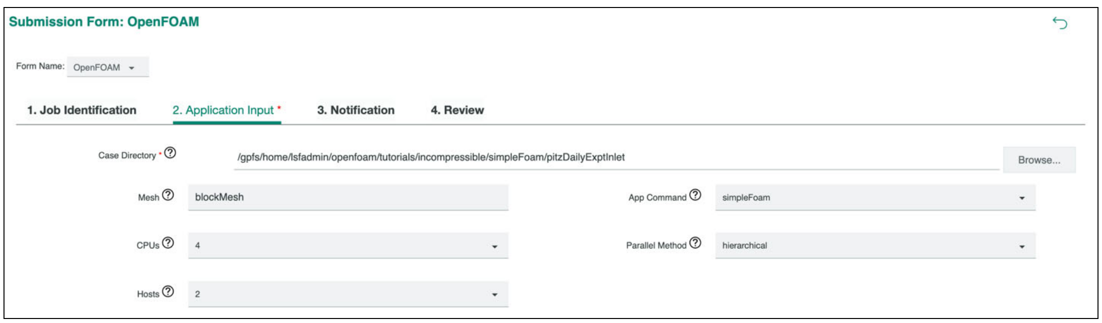


Figure 5-3 Submission Form: OpenFOAM

The following steps are the command line equivalent of the submission to LSF when using OpenFOAM-v1912 is used:

1. Copy the pitzDailyExptInlet example:
   ```
   $ cd
   $ cp -Rp
   JOB_REPOSITORY_TOP/tutorials/tutorials/incompressible/simpleFoam/pitzDailyExptInlet .
   $ cd pitzDailyExptInlet
   ```
2. Create a `bsub.script` file and set MPI_INTERFACE to a high-speed network interface in your environment, as shown in Example 5-1.

Example 5-1 Creating bsub.script file

```
#!/bin/bash
function on_error_exit {
RT=$?;if [ $RT -ne 0 ]; then exit $RT;fi
}

MPI_INTERFACE=enP48p1s0f0

rm -rf processor*

source /opt/OpenFOAM-v1912/etc/bashrc
on_error_exit
```

```
blockMesh
on_error_exit

touch apitzDailyExptInlet.foam
on_error_exit

decomposePar
on_error_exit

mpirun -mca btl_tcp_if_include $MPI_INTERFACE -mca btl ^openib -mca pml ob1 -mca
plm ^rsh /bin/bash -c 'source /opt/OpenFOAM-v1912/etc/bashrc && simpleFoam
-parallel'
on_error_exit
```

3. Submit a job to run on a single host:

```
$ bsub -N -R "span[hosts=1]" -app openfoam -n 4 -o output.team01.txt -e
error.team01.txt ./bsub.script
```

4. Submit a job to run on two hosts:

```
$ bsub -N -R "span[ptile=2]" -app openfoam -n 4 -o output.team01.txt -e
error.team01.txt ./bsub.script
```

5. Examine the output and error files:

```
$ tail output.team01.txt
SIMPLE solution converged in 791 iterations

streamLine streamLines write:
    seeded 10 particles
    Tracks:10
    Total samples:10860
    Writing data to
"/gpfs/tmp/LSF/home/team01/pitzDailyExptInlet_1581699373341oOvwb/pitzDailyExptI
nlet/postProcessing/sets/streamLines/791"
End

Finalising parallel run
$
```

5.3 ParaView sample submission template for IBM Spectrum LSF Suite

With the working installation of the OpenFOAM container that was created in 5.2, “OpenFOAM sample submission template for IBM Spectrum LSF Suite” on page 48, you are ready to move on to setting up ParaView to visualize the results from OpenFOAM.

The OpenFOAM container that was created or downloaded also contains the open source visualization tool ParaView. ParaView is a generic visualization tool that can visualize data from a wide range of supported scientific applications, including OpenFOAM.

For more information about deployment steps for the ParaView template, see this web page. Users are required to first complete the steps that are outlined in the Deployment Steps section there.

Complete the following steps:

1. When enabling the ParaView template according to the procedure that is defined at this web page, the administrator is prompted to specify the JOB_REPOSITORY_TOP directory, as shown in Figure 5-4. This directory must be set to the same value as during the configuration of the OpenFOAM template as described in 5.2, “OpenFOAM sample submission template for IBM Spectrum LSF Suite” on page 48.

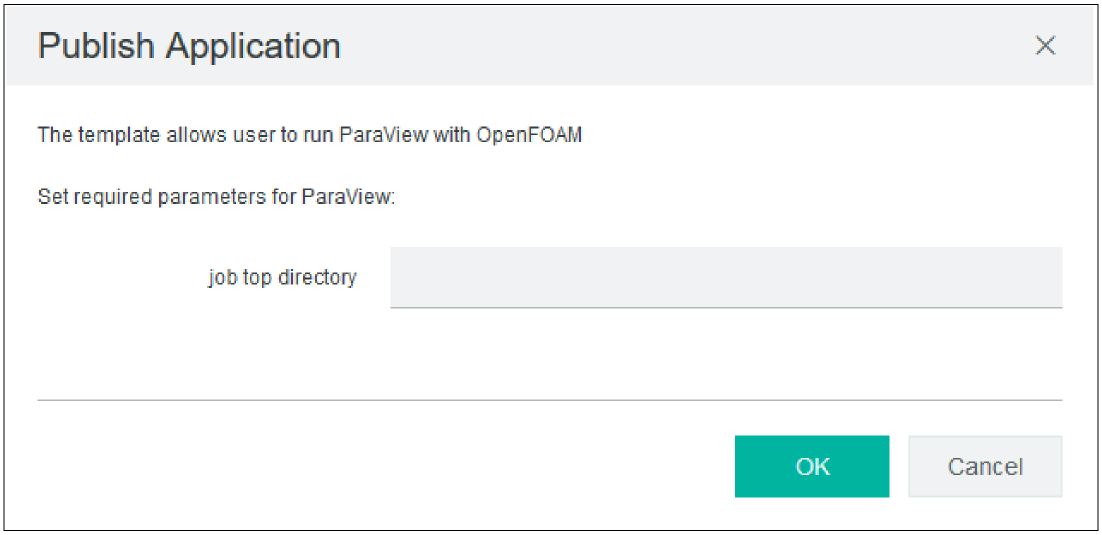


Figure 5-4 Publish Application pane: job top directory entry

2. After it is published, test that ParaView starts and displays successfully through a VNC session on the desktop. A corresponding LSF job is started for each ParaView instance that is started.

   As a normal LSF user, login to the Spectrum LSF Suite web interface and click **Workload** → **New Workload**. Here, you see that tiles are available for ParaView and OpenFOAM (as created in 5.2, “OpenFOAM sample submission template for IBM Spectrum LSF Suite” on page 48).

Complete the following steps:

a. Click **ParaView** to start the application, as shown in Figure 5-5.

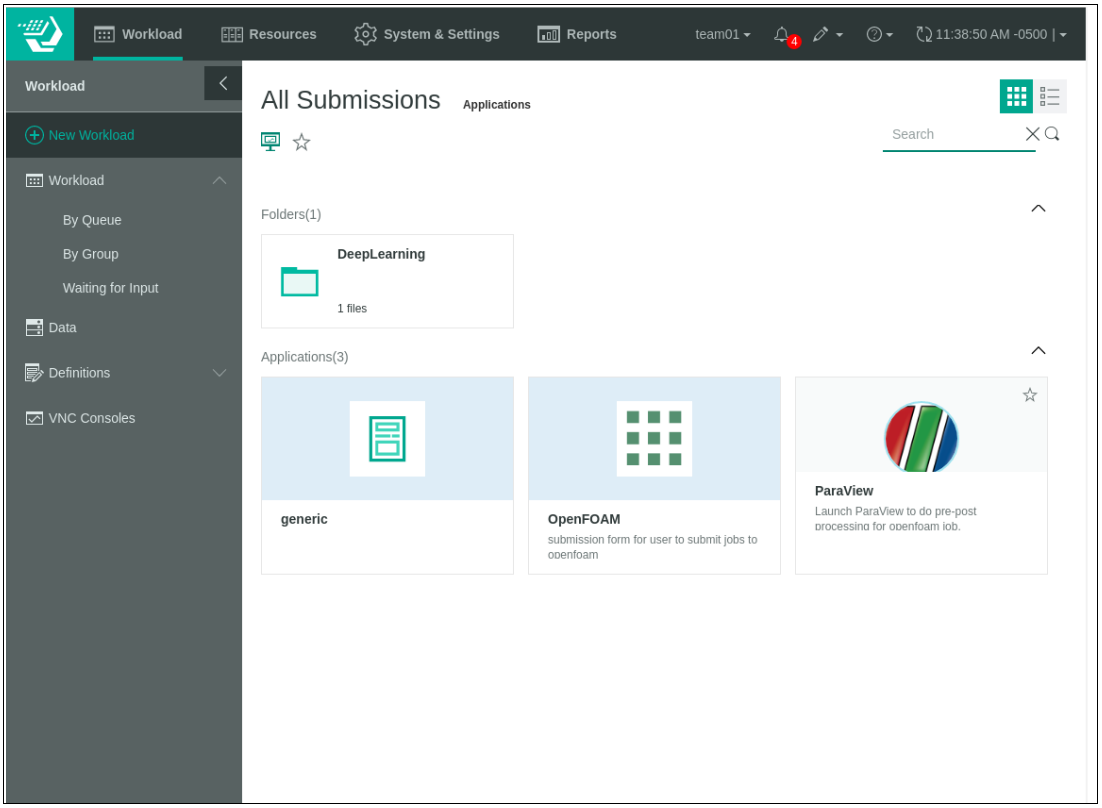


Figure 5-5 All Submissions pane: Applications

b. After a moment, ParaView is started and displayed in a web browser VNC pop-up window, as shown in Figure 5-6 on page 53.

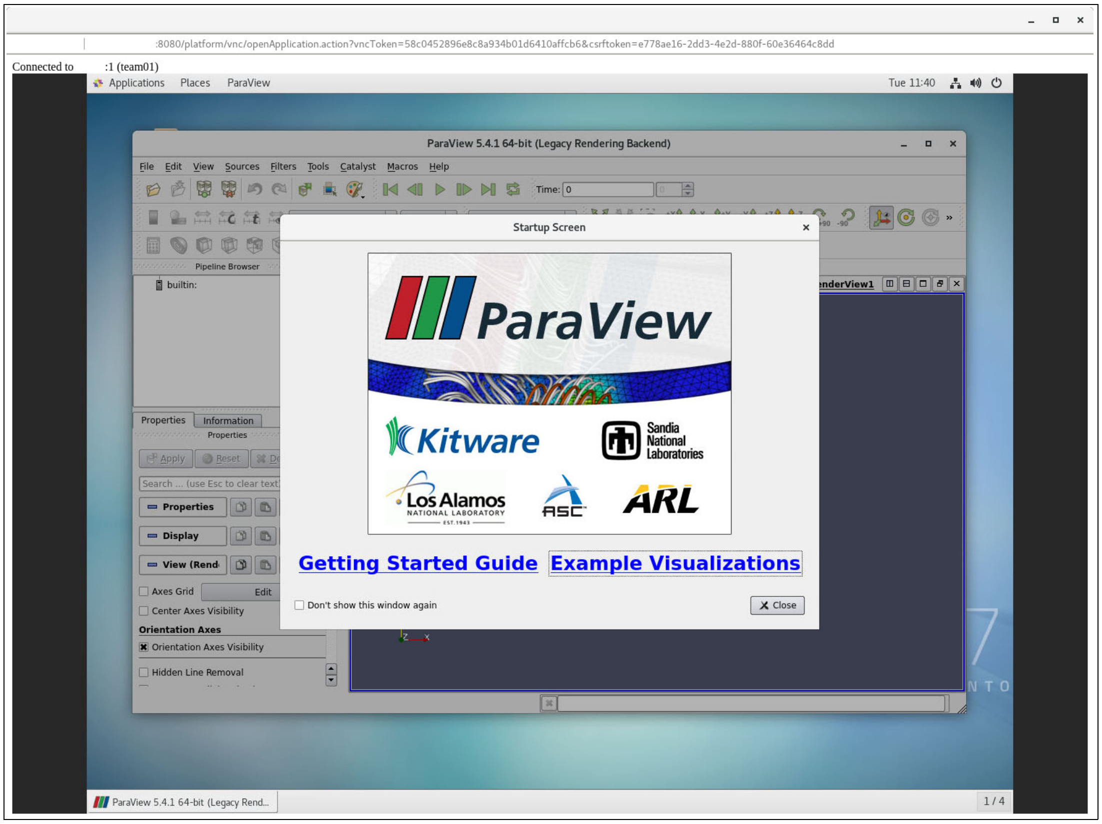


Figure 5-6 ParaView Start window

c. You note that for each ParaView instance that is started, a corresponding LSF job also is listed, as shown in Figure 5-7. You see the running job instance in the Spectrum LSF Suite web interface by clicking **Workload** → **Workload**.

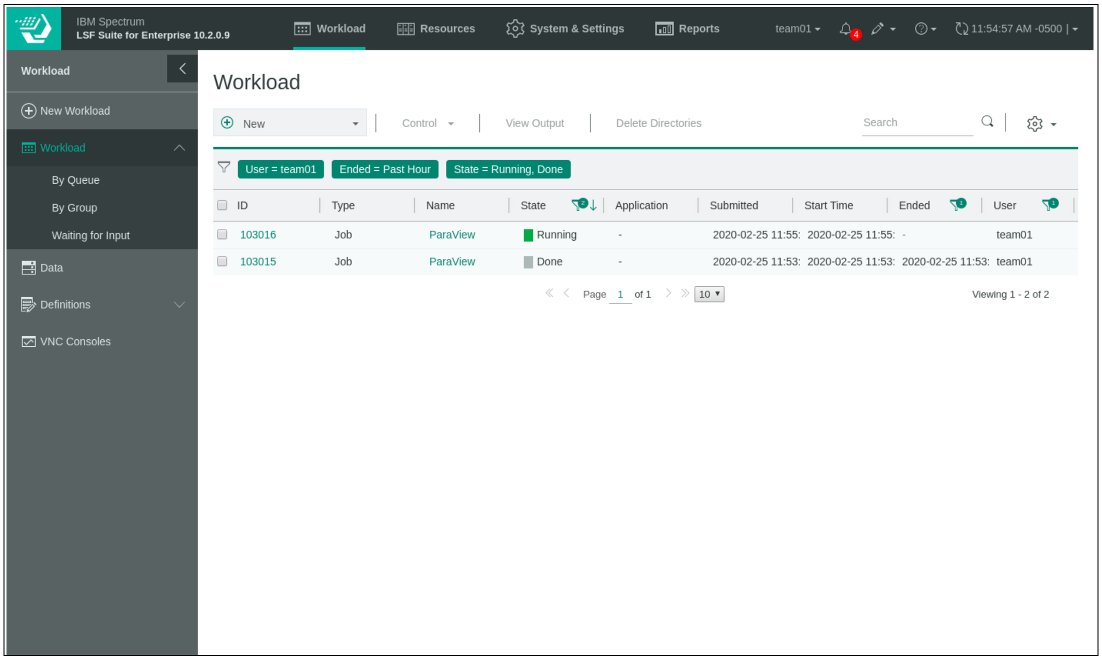


Figure 5-7 IBM Spectrum LSF Suite web interface pane

d. Now that you validated that ParaView starts (see Figure 5-8), close the ParaView application by selecting **File** → **Exit** or the Exit window widget. The corresponding LSF job's status is changed to DONE.

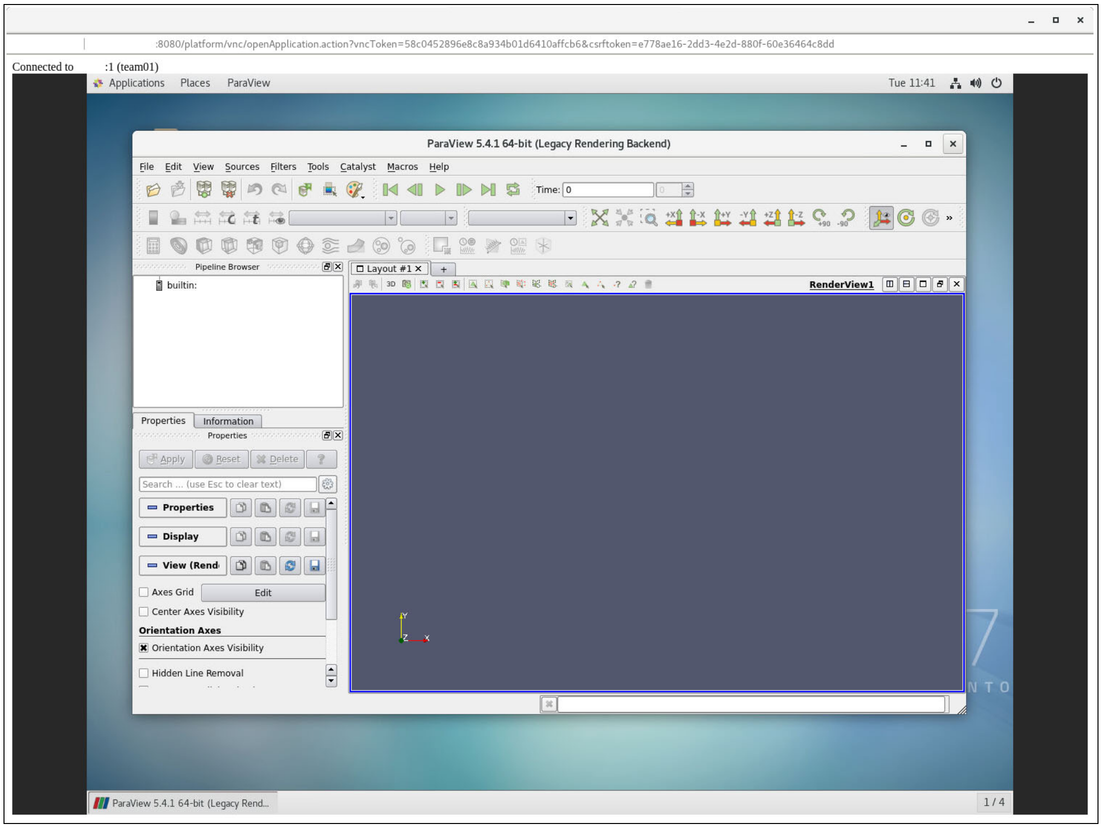


Figure 5-8 ParaView rendering pane

5.4 Running an OpenFOAM job and visualizing data with ParaView by way of the Spectrum LSF Suite web interface

How to configure templates in the Spectrum LSF web interface for the applications OpenFOAM and ParaView was described in 5.2, "OpenFOAM sample submission template for IBM Spectrum LSF Suite" on page 48 and 5.3, "ParaView sample submission template for IBM Spectrum LSF Suite" on page 50.

This section shows an example of submitting an OpenFOAM job and visualizing the data with ParaView when the job is complete.

Complete the following steps:

1. Log in as a standard LSF user to the Spectrum LSF Suite web interface and click **Workload** → **New Workload**, as shown in Figure 5-9 on page 56. Select **OpenFOAM**. The OpenFOAM submission form is displayed, which walks through the process to submit an OpenFOAM job. Here, you can specify which tutorial example to run, and other parameters for the job.

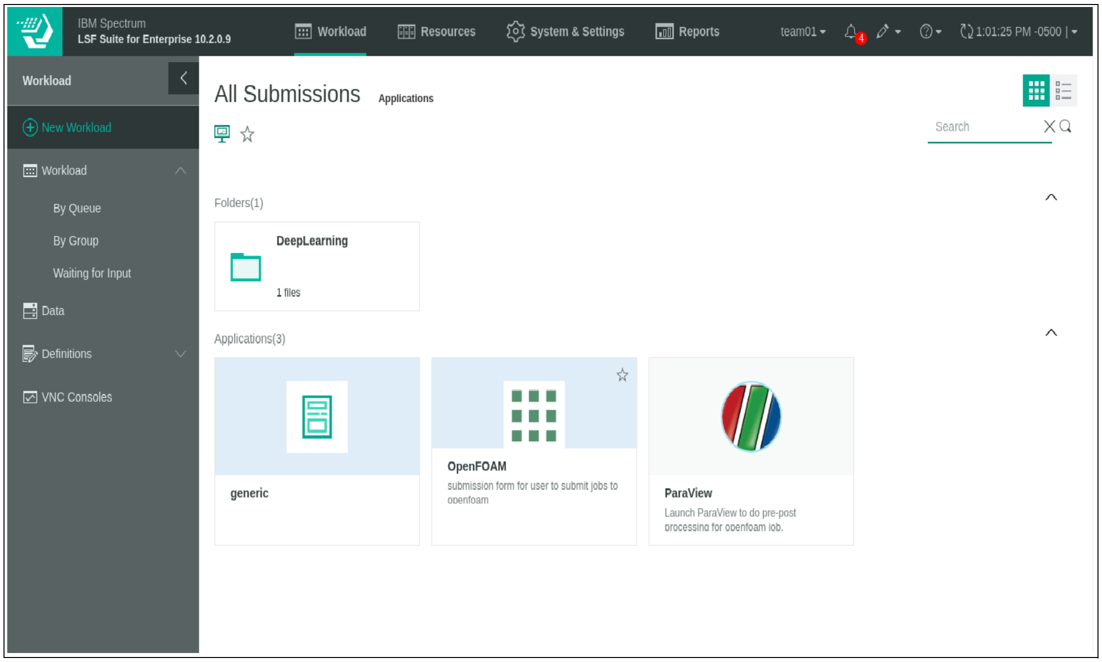


Figure 5-9 IBM Spectrum LSF Suite web interface: OpenFOAM

Complete the following steps:

a. In the Job Identification tab, a job name and description for the job can be specified, as shown in Figure 5-10. Click **Next** to move to the next tab.

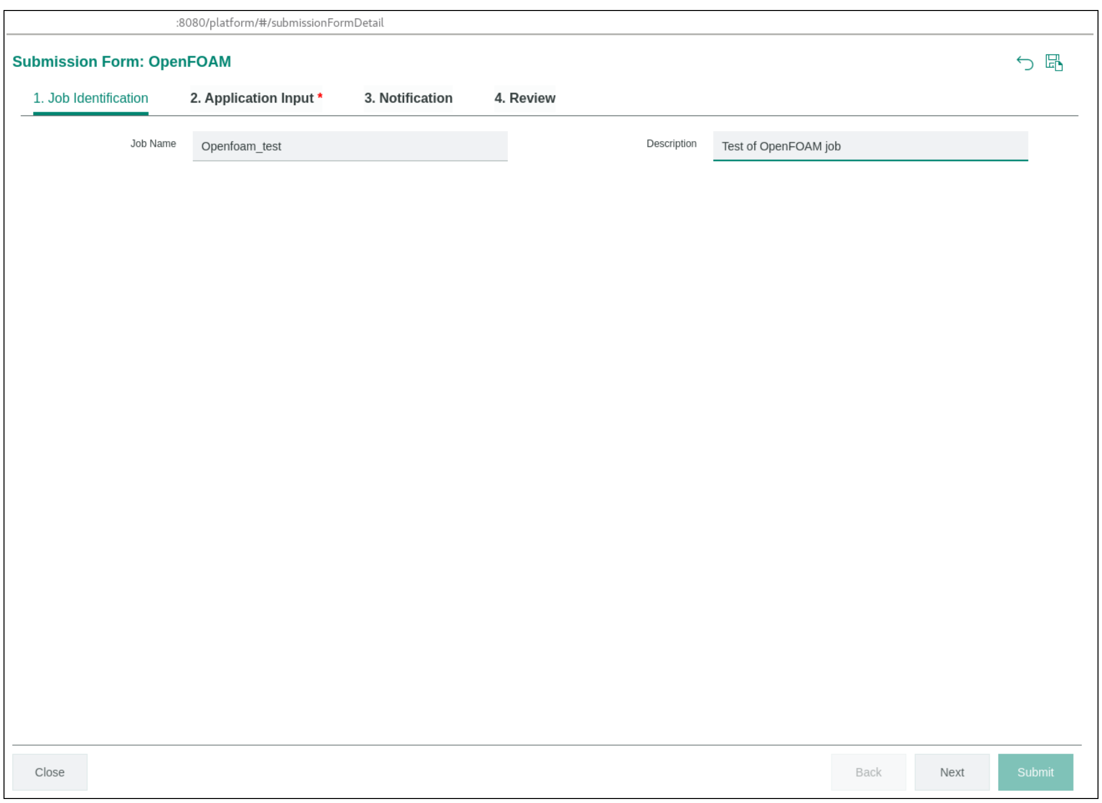


Figure 5-10 Submission Form: OpenFOAM Job Identification pane

b. Click **Browse** for the Case Directory on the Application Input window. OpenFOAM provides several examples (tutorials). Select the **motorBike** example, which can be found in: `JOB_REPOSITORY_TOP/tutorials/incompressible/simpleFoam`, as shown in Figure 5-11. The OpenFOAM template automatically completes some parameters based on the motorBike data files. Figure 5-11 and Figure 5-12 on page 58 show the selection of the motorBike example and the parameters that were set.

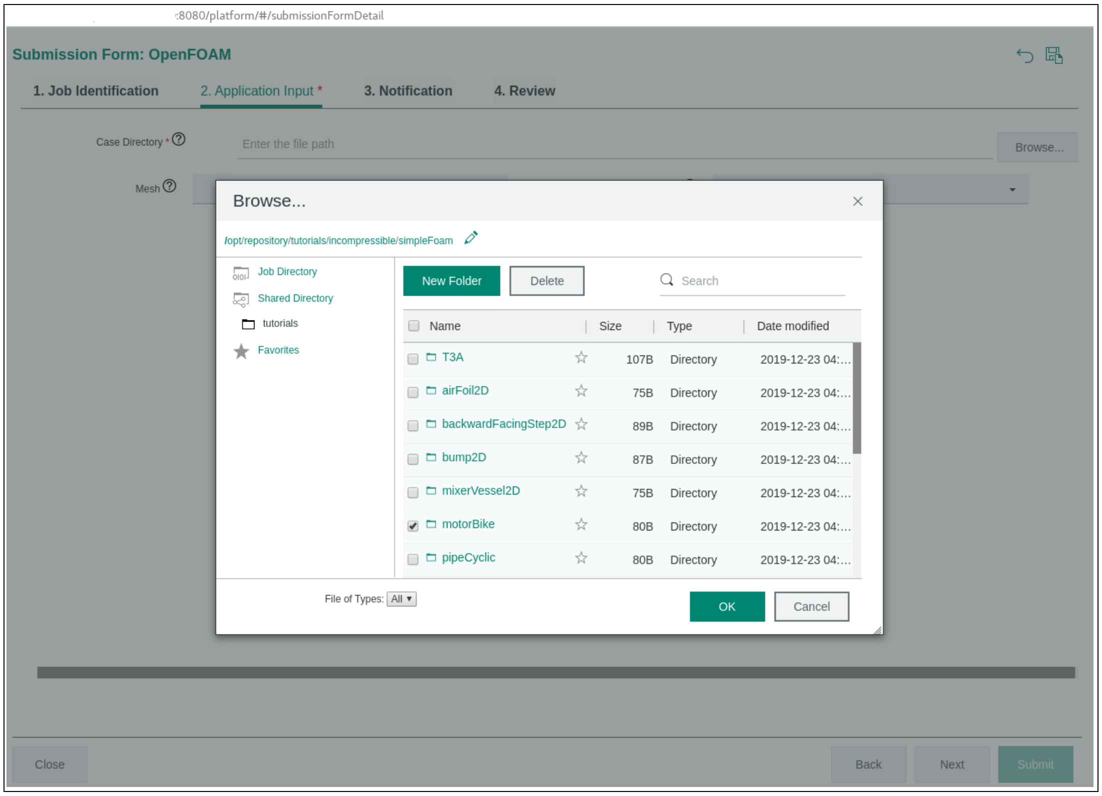


Figure 5-11 Submission Form: OpenFOAM Application Input pane

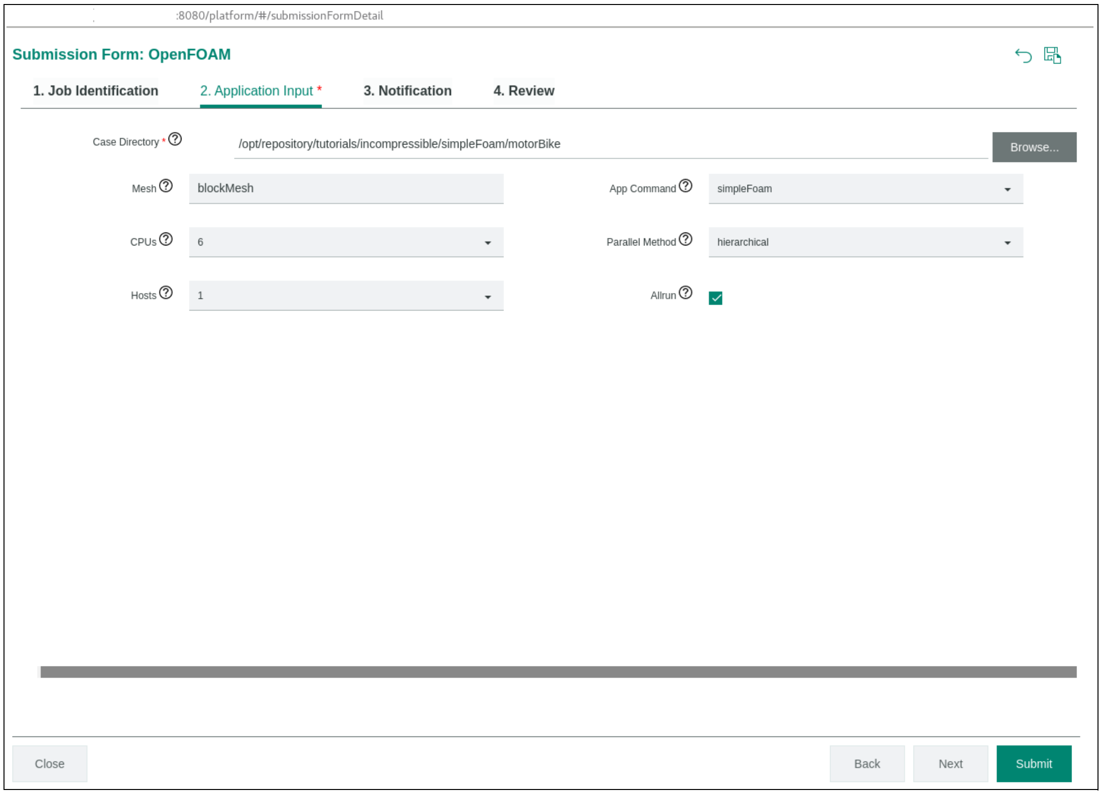


Figure 5-12 Submission Form: OpenFOAM Application Input pane continues

The Notification window allows users to enable notifications for the job that is being submitted, as shown in Figure 5-13 on page 59. Notifications can be delivered to the Spectrum LSF Suite web interface, email, and to the Spectrum LSF mobile and desktop clients.

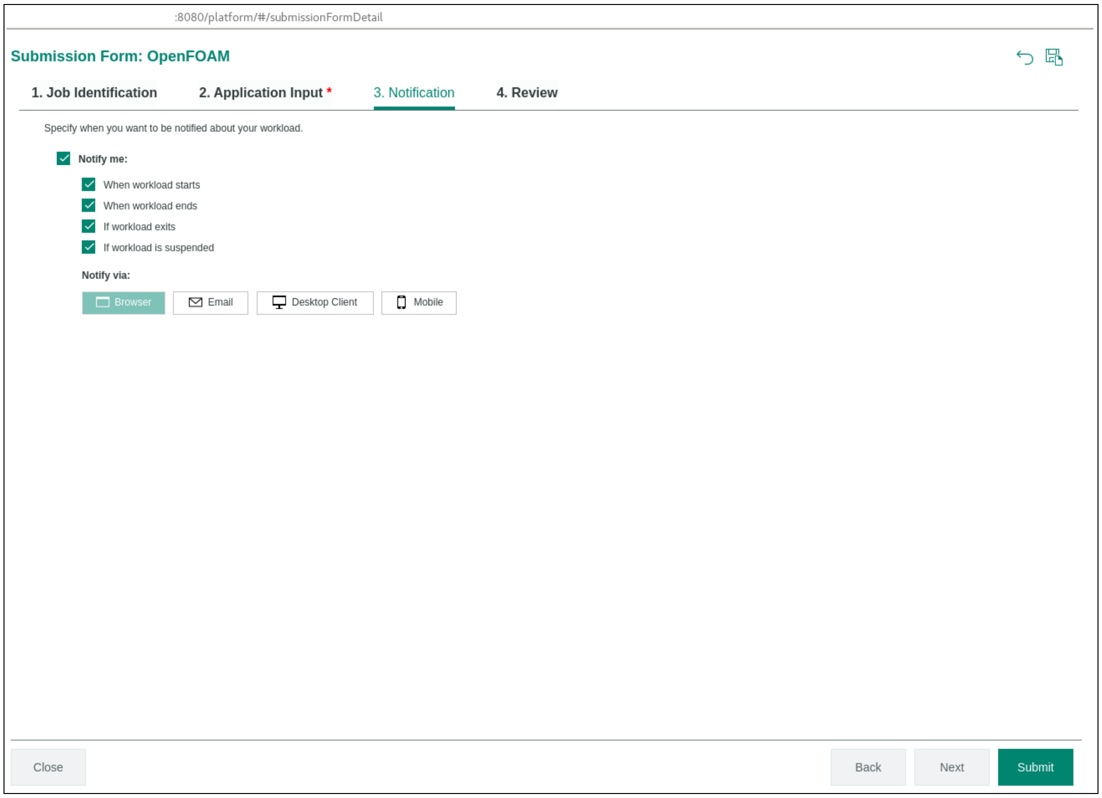


Figure 5-13 Submission Form: OpenFOAM Notifications pane

The Review pane shows all the settings that were specified so far for the OpenFOAM job, as shown in Figure 5-14 on page 60. This gives users the ability to complete a final check before clicking Submit. After clicking Submit, the unique jobID is displayed in the web interface. The job enters RUN state after the requested resources are available in the cluster.

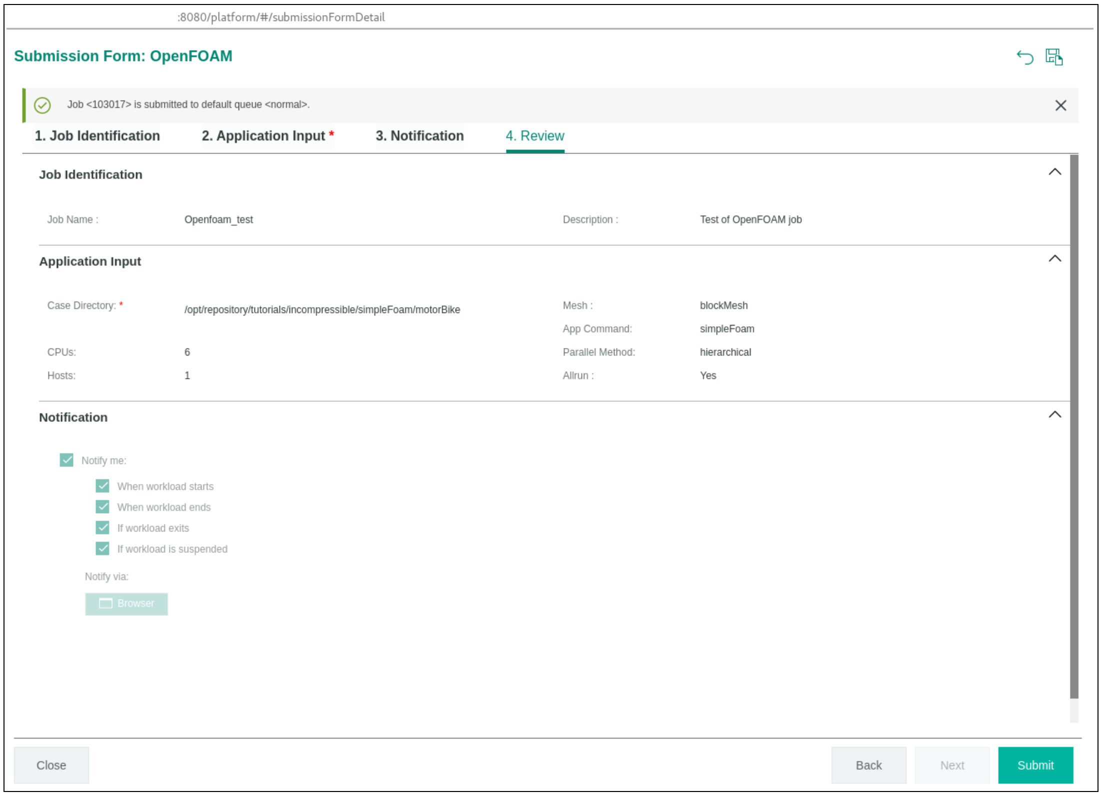


Figure 5-14 Submission Form: OpenFOAM Review pane

2. With the OpenFOAM example job submitted, click **Workload** → **Workload** in the Spectrum LSF web interface (as shown in Figure 5-15) to check the status of the job. The job enters RUN state and details regarding the running job can be found by selecting the jobID link. You can view the data (file listing) for the job.

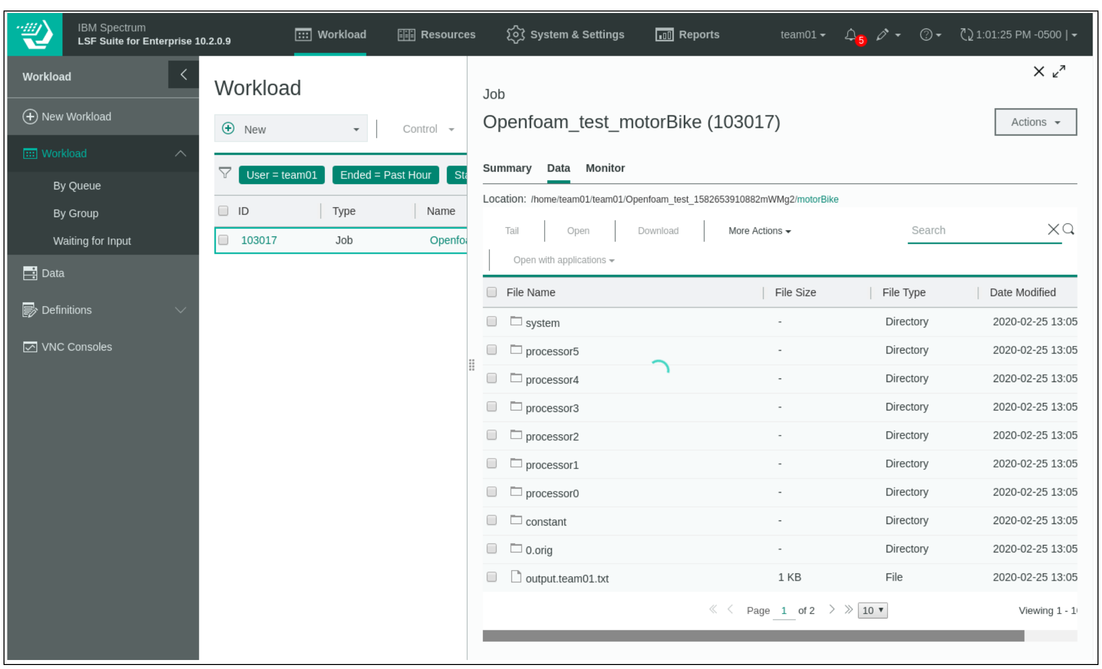


Figure 5-15 IBM Spectrum LSF Suite: Workload pane

3. After the job runs to completion, browse to the Data view of the job and search for the file `amotorBike.foam`, as shown in Figure 5-16. This trigger file is automatically created by the OpenFOAM application template. This file is used to instruct ParaView to display data from the OpenFOAM application. Select the **amotorBike.foam** file and click **Open with applications** → **ParaView**. ParaView starts and automatically opens the data files from the motorBike OpenFOAM job run.

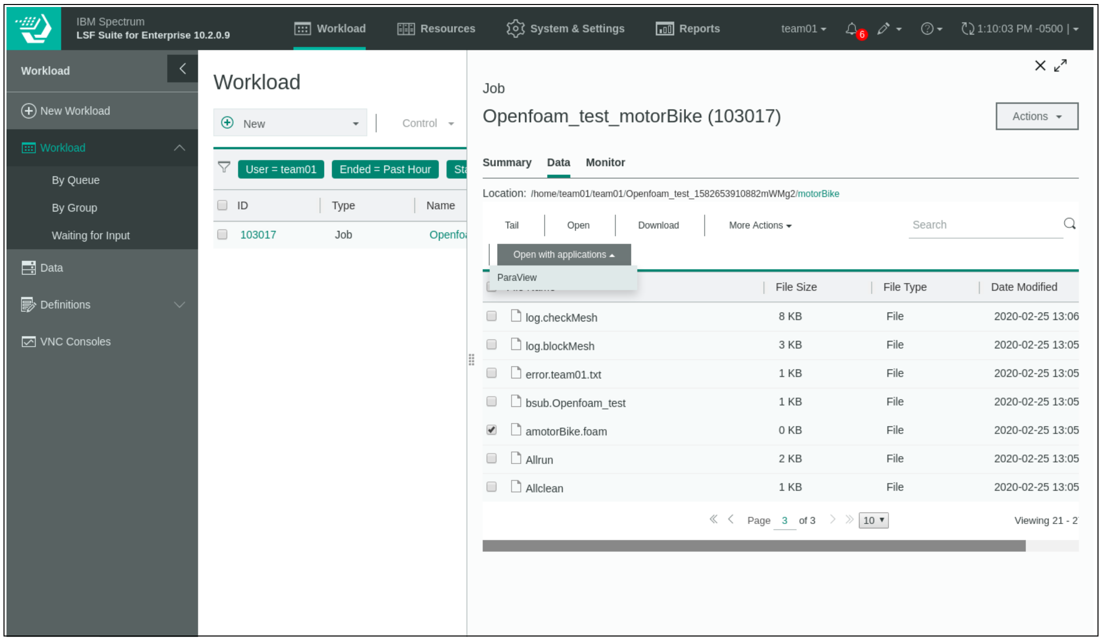


Figure 5-16 IBM Spectrum LSF Suite: Workload Job pane

4. The ParaView application starts in a web browser VNC window, as shown in Figure 5-17 on page 62. It automatically loads the data from the OpenFOAM motorBike job that was run.

   Under Properties on the lower left side of the ParaView window, select the **Mesh Regions** option. Clear the following items:

   – internalMesh
   – frontAndBack
   – inlet
   – outlet
   – lowerWall
   – upperWall

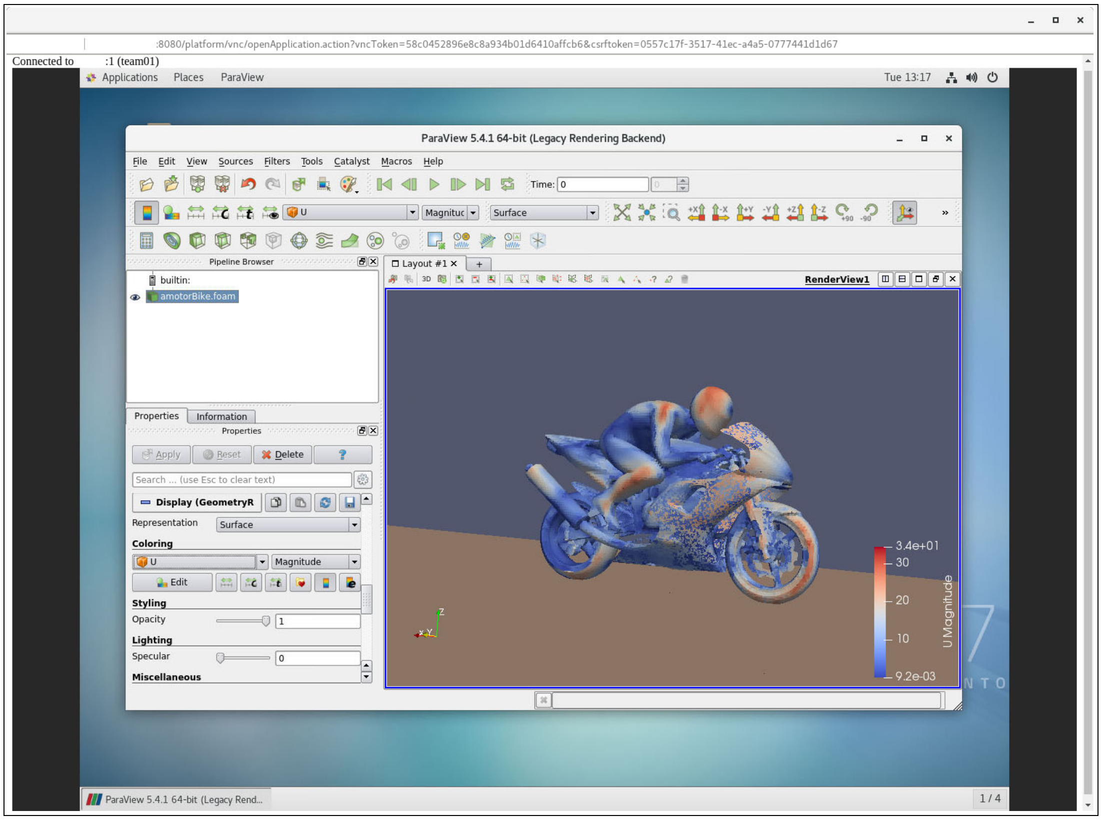


Figure 5-17 ParaView: Legacy Rendering Backend pane

5. Click **Apply**. To visualize, click the eye icon in the Pipeline Browser that is next to the entry `amotorBike.foam`. Now, you can manipulate the rendering in the Render View window.

A

IBM Spectrum LSF and Kubernetes integration

This appendix provides more information about LSF and Kubernetes integration.

This appendix includes the following topics:

LSF and Kubernetes

Kubernetes provides a way for administrators to deploy long running services. It also provides an abstraction that eases the deployment of prebuilt services. More recently, LSF administrators and users recognized containers as a convenient way to package applications and their dependencies into a single image. This feature allows a packaged application to be run on different underlying environments without needing extensive time to port to that environment. For example, a container with OpenFoam can easily be run on premises or on the cloud.

IBM Spectrum LSF Connector for Kubernetes provides a way to run Kubernetes pods in an LSF environment. It creates a single environment where pods and batch jobs can coexist on the same infrastructure. It removes the need to create siloed LSF and Kubernetes environments. Administrators can maintain a single environment and users can run new types of workloads.

IBM Spectrum LSF Connector for Kubernetes allows LSF and Kubernetes to share a physical infrastructure (see Figure A-1).

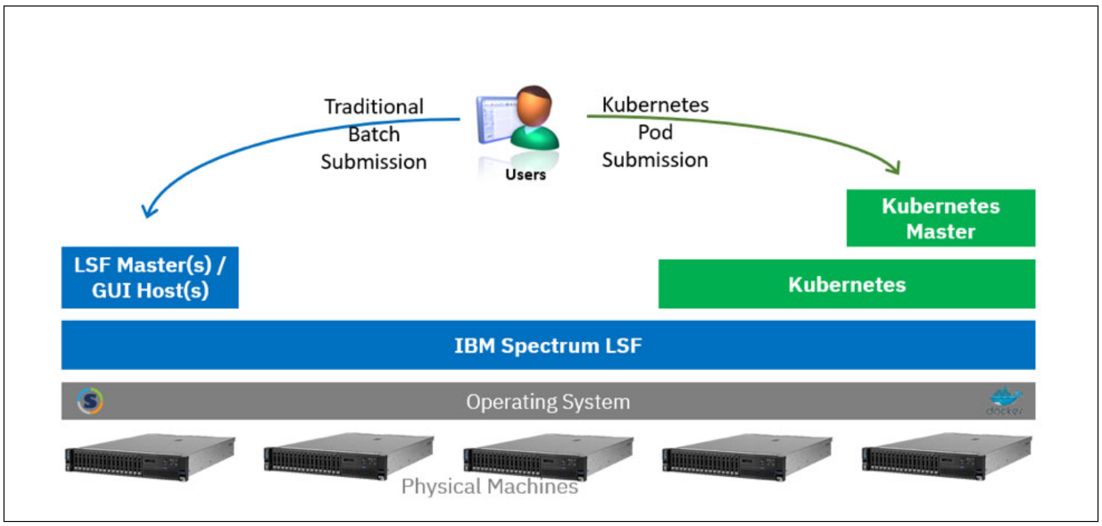


Figure A-1 Logical diagram of LSF and Kubernetes integration

The LSF workload likely is the major workload, so most of the machines are dedicated to LSF. However, a subset of the machines can be shared between LSF and Kubernetes.

LSF is installed on all the machines in the cluster, and Kubernetes is installed on a subset of the LSF cluster. The LSF Master and Kubernetes Masters must be kept on separate machines. With this configuration, users can run batch and Kubernetes workloads on the same infrastructure.

For more information about how to configure the integration, see IBM Knowledge Center.

Checking the Kubernetes Connector

LSF now includes a connector for interfacing with Kubernetes. When enabled, this connector allows LSF to function as a scheduler for Kubernetes clusters. It also allows LSF to schedule pods as jobs. It is not enabled by default.

IBM Spectrum LSF Suite HPC and Enterprise Editions include the IBM Spectrum LSF Connector for Kubernetes. This connector allows LSF to provide the scheduling for Kubernetes. Testing the connector depends on how users submit jobs to the cluster. For traditional LSF users job submission, follow the instructions that are available at IBM Knowledge Center.

For users that want to use the Kubernetes CLI for job submission, use the following examples to test the integration. Save the test job sample (as shown in Example A-1) to `job-1.yaml`.

Example A-1 Testing integration with the Kubernetes CLI example

```
apiVersion: batch/v1
kind: Job
metadata:
  name: job-1
spec:
  template:
    metadata:
      name: job-1
      annotations:
        lsf.ibm.com/queue: "normal"
    spec:
      schedulerName: lsf
      containers:
      - name: myjob
        # Change the image to something you have locally
        image: ubuntu:latest
        imagePullPolicy: IfNotPresent
        command: ["sleep", "120"]
        resources:
          requests:
            cpu: 1
            memory: 128M
          limits:
            cpu: 1
            memory: 256M
      restartPolicy: Never
```

The sample job submits the job to the normal queue and uses LSF as the scheduler. In a mixed LSF Kubernetes environment, LSF must be used to submit all jobs. LSF still contacts the Kubernetes scheduler when scheduling on Kubernetes nodes to ensure that no limits are exceeded. To submit the test job, run the following command:

```
$ kubectl create -f job-1.yaml
```

The job is created in the users namespace. To check the state of the Kubernetes job, run the following command:

```
$ kubectl get jobs
NAME      DESIRED   SUCCESSFUL   AGE
job-1     1         0            7s
```

The job is initially in a pending state, but starts after LSF schedules it. You can see the pod state by running, the following command:

```
# kubectl get pods
NAME                        READY     STATUS    RESTARTS   AGE
job-1-76qfx                 1/1       Running   0          40s
```

LSF adds labels to the pod. These labels include the Job ID and any Pending reasons. You can see them by running the **oc describe pod job-1-76qfx** command, as shown in Example A-2.

Example A-2 LSF job report for a pod

```
# kubectl describe pod job-1-76qfx
Name:               job-1-76qfx
Namespace:          default
Priority:           0
PriorityClassName:  <none>
Node:               compute01/10.10.10.21
Start Time:         Tue, 19 Nov 2019 15:39:51 -0500
Labels:             controller-uid=1664dfdf-0b0d-11ea-b2b6-0cc47a85dfa6
                    job-name=job-1
                    lsf.ibm.com/jobId=305
Annotations:        lsf.ibm.com/queue=normal
                    openshift.io/scc=anyuid
Status:             Running
IP:                 10.129.0.2
```

You can create a test job that remains pending forever. Copy the `job-1.yaml` to `job-2.yaml` and set the name to job-2. Change the CPU request to 100 cores as shown in the following example:

```
requests:
  cpu: 100
  memory: 128M
limits:
  cpu: 100
  memory: 256M
```

Now, submit this job and check its state, as shown in Example A-3.

Example A-3 Submitting and checking the status of job2

```
# vi job-2.yaml

# oc create -f j2.yaml
job.batch/job-2 created

# oc get pods
NAME                        READY     STATUS      RESTARTS   AGE
job-1-76qfx                 0/1       Completed   0          9m
job-2-wvwjj                 0/1       Pending     0          4s
```

You can get more information from the pod, and you can see why it is pending, as shown in Example A-4.

Example A-4 Query more information from pod job-2

```
# oc describe pod job-2-wvwjj

Name:              job-2-wvwjj
Namespace:         default
Priority:          0
PriorityClassName: <none>
Node:              <none>
Labels:            controller-uid=5e6dcd20-0b0e-11ea-b2b6-0cc47a85dfa6
                   job-name=job-2
                   lsf.ibm.com/jobId=306
Annotations:       lsf.ibm.com/pendingReason="Blocked by Kubernetes policies
(Insufficient cpu): 2 hosts;"
                   lsf.ibm.com/queue=normal
                   openshift.io/scc=anyuid
Status:            Pending
```

Testing parallel jobs in Kubernetes

Kubernetes does not natively support parallel jobs. The LSF Connector for Kubernetes provides a new parallel job kind. By using this job kind, users can create parallel jobs and elastic jobs that run within Kubernetes. To check that the capability exists, run the following command:

```
# kubectl get crds

NAME                                    CREATED AT
<removed for clarity>
paralleljobs.ibm.com                    2019-11-19T20:02:44Z
```

Note: Non-OpenShift users can use **`kubectl`** instead of **`oc`**.

The `paralleljobs.ibm.com` Custom Resource Definition (CRD) extends the Kubernetes API to support the parallel job type. For more information about the sample parallel job yaml file, see Appendix A, “IBM Spectrum LSF and Kubernetes integration” on page 63.

Save the file as `pjob-3.yaml`.

This sample parallel job features the structure that is shown in Figure A-2.

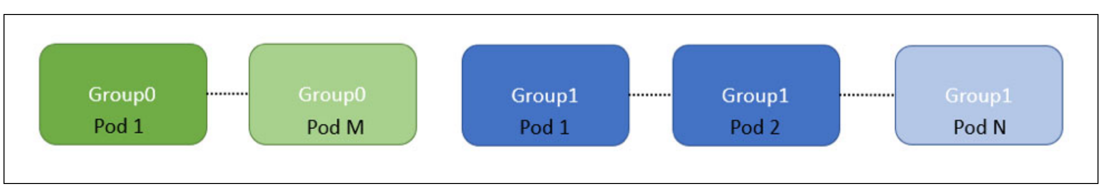


Figure A-2 Parallel job structure

It can contain several groups of tasks. Each group is performing a specific function. For AI workloads, one group might be used for running a parameter server. Within a group, you can have one or more pods. These pods are running the same image in parallel.

In the sample that is provided, two groups are available. The first group run one pod and the second group runs two pods. You can adjust the size by changing the number of replicas in each group.

Run the parallel job by running the following command:

```
# kubectl create -f pjob-3.yaml
paralleljob.ibm.com/pjob-3 created
```

You can get more information about the parallel job by running the **`kubectl describe pj pjob-3`** command, as shown in Example A-5.

Example A-5 Query information about parallel job

```
# kubectl describe pj pjob-3
Name:         pjob-3
Namespace:    default
Labels:       test=experiment10
Annotations:  deletePodOnFlexDown=y
              lsf.ibm.com/queue=priority
API Version:  ibm.com/v1alpha1
Kind:         ParallelJob
  :              :
```

You can also see the pods in this job by running the following command:

```
$ kubectl get pods

NAME                        READY     STATUS     RESTARTS   AGE
pjob-3-group0-4pb4h         1/1       Running    0          13s
pjob-3-group1-9gkvx         1/1       Running    0          13s
pjob-3-group1-hl2kp         1/1       Running    0          13s
```

Note: The sample uses an Ubuntu image. If the image is not available, a container creation error is shown. Alter the sample to use an image that you can access.

To remove the parallel job, run the following command:

```
# kubectl delete -f pjob-3.yaml
paralleljob.ibm.com "pjob-3" deleted
```

Successfully running pods demonstrate that the LSF Kubernetes connector is functioning properly and the Custom Resource Definition for parallel and elastic jobs is working.

Parallel job yaml file sample

The following sample parallel job features two types of tasks with one of those tasks being a parallel task:

```
apiVersion: ibm.com/v1alpha1
kind: ParallelJob
metadata:
```

```
  name: pjob-3
  namespace: default
  annotations:
    lsf.ibm.com/queue: "priority"
    deletePodOnFlexDown: "y"
  labels:
    test: experiment10
spec:
  name: pjob-3 #same with metadata/name
  description: This is a parallel job with 2 tasks.
  headerTask: group0
  priority: 100
  resizable: false
  schedulerName: lsf
  taskGroups:
  - metadata:
      name: group0
    spec:
      replica: 1
      template:
        spec:
          nodeSelector:
            beta.kubernetes.io/arch: amd64
            beta.kubernetes.io/os: linux
          containers:
          - args:
            command: ["sleep", "90"]
            image: ubuntu
            name: task1
            resources:
              limits:
                cpu: 1
                memory: 400Mi
              requests:
                cpu: 1
                memory: 400Mi
          restartPolicy: Never
  - metadata:
      name: group1
    spec:
      replica: 2
      template:
        spec:
          nodeSelector:
            beta.kubernetes.io/arch: amd64
            beta.kubernetes.io/os: linux
          containers:
          - args:
            command: ["sleep", "90"]
            image: ubuntu
            name: task1
            # For GPU Jobs add the following
            # env:
            # - name: NVIDIA_DRIVER_CAPABILITIES
            #   value: "compute,utility"
```

```
  # securityContext:
  #   allowPrivilegeEscalation: false
  #   capabilities:
  #     drop: ["ALL"]
  #   seLinuxOptions:
  #     type: nvidia_container_t
  resources:
    limits:
      # nvidia.com/gpu: 1
      cpu: 1
      memory: 200Mi
    requests:
      # nvidia.com/gpu: 1
      cpu: 1
      memory: 200Mi
restartPolicy: Never
```

B

Managing Elasticsearch data

This appendix describes how to manage Elasticsearch data.

Overview

Over time, the amount of data that is stored in Elasticsearch grows. It is important to make sure that the file system that contain the data never runs out of space. If it does run out of space, the indexes are corrupted.

The following script is provided to trim the data in Elasticsearch:

```
/opt/ibm/lsfsuite/ext/explorer/server/config/ drop_es_indices.cron
```

This script must be run daily, as with a cron job, to remove old data. The configuration file `/opt/ibm/lsfsuite/ext/explorer/server/config/indexpurge.conf` controls how many days of data to keep. Edit the configuration file and set the number of days of data that you want to keep.

> **Note:** Consider the following points:
>
> - Monitor the file system to ensure that enough space exists.
> - We strongly recommend that you view management and performance turning tutorials and videos that are available on the Elasticsearch website.

Moving the Elasticsearch data

Elasticsearch stores its data in `/opt/ibm/elastic/elasticsearch/data`. If the file system where this data is stored becomes full, the following options are available to fix the problem:

- Use the procedure that is described in “Overview” on page 72 to reduce the number of days of data that is retained.
- Relocate the data to a different directory.
- Mount a large disk to `/opt/ibm/elastic/elasticsearch/data`.

The location that Elasticsearch uses to store data is provided in the `systemd` startup script `/etc/systemd/system/elasticsearch-for-lsf.service`. Inside this file, you see:

```
[Service]
Type=forking
Environment=ES_HOME=/opt/ibm/elastic/elasticsearch
Environment=CONF_DIR=/opt/ibm/elastic/elasticsearch/config
Environment=DATA_DIR=/opt/ibm/elastic/elasticsearch/data
Environment=LOG_DIR=/opt/ibm/elastic/elasticsearch/log
```

The DATA_DIR defines the directory to use to store data. If this installation is new, complete the following steps:

1. Edit `/etc/systemd/system/elasticsearch-for-lsf.service` and change this location to another location. Do not use the same shared location for the other GUI hosts.
2. Run the **`systemctl daemon-reload`** command.
3. Restart Elasticsearch by using the **`systemctl restart elasticsearch-for-lsf.service`** command.
4. Repeat these steps on all GUI hosts.

If a large disk is available, complete the following steps:

Note: Elasticsearch is sensitive to disk performance. When possible, use a fast disk or SSD.

1. Mount the new disk in a temporary location:

   ```
   # mkdir /root/tmpmnt
   # mount {name of device} /root/tmpmnt
   ```

2. Stop elasticsearch:

   ```
   # systemctl stop elasticsearch-for-lsf.service
   ```

   Note: Other services require the Elasticsearch service; therefore, you can stop the logstash-for-lsf.service.

3. Copy the data over to the temporary mounted disk:

   ```
   # cp /opt/ibm/elastic/elasticsearch/data/* /root/tmpmnt
   ```

4. Rename the old data:

   ```
   # mkdir /opt/ibm/elastic/elasticsearch/data.OLD
   # mv /opt/ibm/elastic/elasticsearch/data/nodes
   /opt/ibm/elastic/elasticsearch/data.OLD
   ```

5. Unmount the new disk:

   ```
   # umount /root/tmpmnt
   ```

6. Update the `/etc/fstab`. Add an entry for the disk and mount it to `/opt/ibm/elastic/elasticsearch/data`.

7. Mount the disk:

   ```
   # mount /opt/ibm/elastic/elasticsearch/data
   You see: /opt/ibm/elastic/elasticsearch/data/nodes
   ```

8. Restart elasticsearch:

   ```
   # systemctl stop elasticsearch-for-lsf.service
   ```

9. Repeat these steps as quickly as possible on the other GUI hosts.

C

IBM Spectrum LSF Suite common questions and issues

This appendix provides resources for more information about some of the most common questions and issues regarding IBM Spectrum LSF Suite. Most of the questions are related to LSF Suite deployment.

LSF common questions and issues

This section provides a list of LSF most common questions and issues and includes links for answers to these questions.

LSF User Community:

https://ibm.biz/LSFCommunity

- Use lsf-list-packages.yml to check the installed packages in LSF Suite:

  https://ibm.co/37QRsTq
- Start up failed elasticsearch-for-lsf service in LSF Suite:

  https://ibm.co/2NdfG2o
- Setting up high-availability ElasticSearch for LSF Explorer:

  https://ibm.co/2NePuV8
- Show Explorer data with Grafana:

  https://ibm.co/2FOHg21
- systemctl starts ElasticSearch shows time-out:

  https://ibm.co/2FDk5rc
- Is there an approach to migrate from LSF10.1 Standard to LSF Suite 10.2?

  https://ibm.co/2QDMJ1A
- Can I install bundled product’s native patch/fix for LSF Suite for HPC?

  https://ibm.co/2sb2Yd5
- How many ways to deploy IBM Spectrum LSF Suite for HPC?

  https://ibm.co/308Srvz
- What is the supported operation system on X86 platform in IBM Spectrum LSF Suite for HPC cluster?

  https://ibm.co/2t8dt1c
- How to avoid dependency error during installation of LSF Suite.

  https://ibm.co/36ICOxh
- How to find bundled product's document in LSF Suite For HPC?

  https://ibm.co/2tO7Cy4
- How to add nodes in LSF Suite for HPC cluster?

  https://ibm.co/30ah8be
- Is it necessary to set up yum repository on all nodes before LSF Suite for HPC installation?

  https://ibm.co/2ux8PKx
- How to uninstall LSF Suite for HPC?

  https://ibm.co/2FF4hV0
- How to plan roles in LSF Suite cluster?

  https://ibm.co/2NcYKco

- Why LSF Suite failed to install with rsync errors?

  https://ibm.co/36Jl30X
- LSF Suite 10.2.0.6 rpms on different types of nodes in an NFS installation case

  https://ibm.co/2NfzAKd
- Unable to deploy lsf cluster by way of the GUI in LSF Suite for Workgroup 10.1.1

  https://ibm.co/2t4hLa1
- How to move ElasticSearch data directory?

  https://ibm.co/36IEJSB

For more information, see the IBM Support web page.

Related publications

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

IBM Redbooks

The IBM Redbooks publication *IBM Spectrum Computing Solutions*, SG24-8373, provides more information about the topic in this document. Note that this publication might be available in softcopy only.

You can search for, view, download, or order this document and other Redbooks, Redpapers, Web Docs, draft, and more materials at the following website:

ibm.com/redbooks

Online resources

The following websites are also relevant as further information sources:

- IBM Spectrum MPI Knowledge Center (GPU support):

  https://ibm.co/37SvIXf
- IBM Support website:

  https://www.ibm.com/support/home/

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

REDP-5572-00

ISBN 0738458570

Printed in U.S.A.