Platform LSF
Version 9 Release 1.3

Using Platform Dynamic Cluster, Version 9.1.3

SC27-5320-02

Platform LSF
Version 9 Release 1.3

Using Platform Dynamic Cluster, Version 9.1.3

SC27-5320-02

Note
Before using this information and the product it supports, read the information in "Notices" on page 67.

First edition

This edition applies to version 9, release 1, modification 3 of IBM Platform Dynamic Cluster (product number 5725G82) and to all subsequent releases and modifications until otherwise indicated in new editions.

Significant changes or additions to the text and illustrations are indicated by a vertical line (|) to the left of the change.

If you find an error in any Platform Computing documentation, or you have a suggestion for improving it, please let us know.

In the IBM Knowledge Center, add your comments and feedback to any topic.

You can also send your suggestions, comments and questions to the following email address:

pccdoc@ca.ibm.com

Be sure include the publication title and order number, and, if applicable, the specific location of the information about which you have comments (for example, a page number or a browser URL). When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

Contents

Chapter 1. Overview

Introduction

This chapter describes the IBM Platform Dynamic Cluster (Dynamic Cluster) system architecture and basic concepts. It also explains the benefits of using Dynamic Cluster, and explains some of the concepts required to effectively administer the IBM Platform LSF (LSF) cluster with Dynamic Cluster enabled.

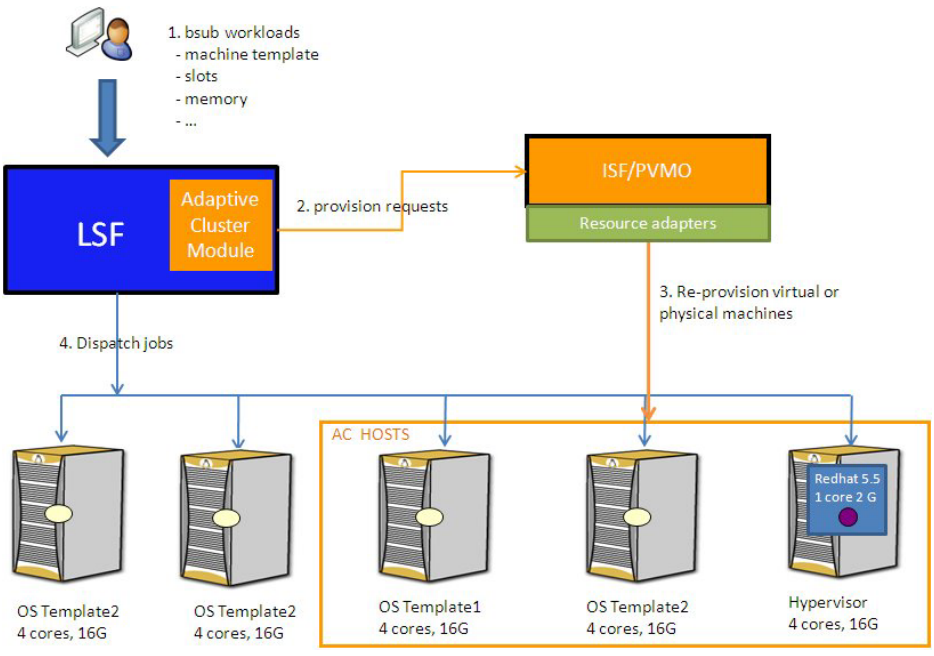
This guide assumes that you have a good knowledge of standard LSF features, as well as a working familiarity with common virtual infrastructure concepts such as hypervisors and virtual machines.

High-level architecture

Broadly speaking, Dynamic Cluster can be thought of as an add-on to LSF that provides the following benefits:

- Dynamically create new virtual machines to satisfy job demands
- Allow jobs to be saved and migrated to another host to release the resources of a priority host
- Restrict job memory usage on a host by running it in a virtual machine, avoiding the possibility of one job hoarding all of a host's memory and interfering with other running jobs

The following figure shows the high-level architecture of Dynamic Cluster.



The Dynamic Cluster system consists of two main components:

- The Dynamic Cluster module built into LSF that makes the scheduler provisioning-aware

- IBM Platform Cluster Manager Advanced Edition (Platform Cluster Manager), which is a product for managing infrastructure resources.

  PVMO (Physical and Virtual Machine Orchestrator) is the component of Platform Cluster Manager that interacts with the underlying virtual provisioning systems. The Platform Cluster Manager master host cannot be used to run virtual machines.

Note:

This means that an installation of Dynamic Cluster requires an installation of both LSF and Platform Cluster Manager. You can configure which hosts in your cluster can participate in dynamic provisioning. This allows you to isolate Dynamic Cluster functionality to as small or as large a subset of a standard LSF cluster as you wish.

Dynamic provisioning

Dynamic Cluster supports virtual machine provisioning. The LSF scheduler enabled with Dynamic Cluster module is provisioning aware, so there are no race conditions or issues with "two-brain" scheduling.

The general flow of workload driven job scheduling in Dynamic Cluster is as follows:

- A user submits a job and requests machine resource requirement, including the desired OS/application stack and machine type, number of CPUs, and memory.
- According to these resource requirements and configured LSF policies, Dynamic Cluster works with the LSF scheduler to select suitable hosts to run the job.
- If the selected machines match the job-level resources requested, LSF dispatches the job right away. Otherwise, a machine provisioning request is generated by LSF and communicated to Platform Cluster Manager, which connects to an external virtual provisioning systems to provision machines. In the middle of provisioning, LSF will reserve the selected Dynamic Cluster host's memory/CPU resources for the job.
- Once the provisioned machine is up and connects to the LSF master host, LSF dispatches jobs to the selected machine.
- When the job completes, the machine remains provisioned, and is able serve new workload.

Without Dynamic Cluster, LSF finds suitable resources and schedules jobs, but the resource attributes are fixed, and some jobs may be pending while resources that do not match the job's requirements are idle. With Dynamic Cluster, idle resources that do not match job requirements are dynamically repurposed, so that LSF can schedule the pending jobs.

Dynamic Cluster can provision the machine type that is most appropriate for the workload:

- Jobs can demand to be scheduled to run on physical machines at submission time to assure greater performance.
- Other jobs can be contained in virtual machines (VMs) for greater flexibility.

  The VM memory and CPU allocations can be modified when powering them on.

Dynamic Cluster hosts in the cluster are flexible resources. If workload requirements are constantly changing, and different types of workload require

different execution environments, Dynamic Cluster can dynamically provision infrastructure according to workload needs (OS, memory, CPUs).

Maximize resource utilization

With Dynamic Cluster you can keep hardware and license utilization high, without affecting the service level for high priority workload. Instead of reserving important resources for critical workload, you can use the resources to run low priority workload, and then preempt those jobs when a high priority job arrives.

Migration is driven by workload priority.

Without Dynamic Cluster, if the LSF job runs on a physical machine, the job is not mobile. The low-priority job must be terminated and rescheduled if it is preempted. If preemption occurs frequently, the job may be started and restarted several times, using up valuable resources without ever completing.

With Dynamic Cluster, the low-priority job can run on a VM, and if the job is preempted, the VM and job can be saved. When the VM is restored, the job continues and eventually finishes without wasting any resources.

Running workload is packed onto the hypervisor to use the smallest possible number of hypervisors. This maximizes availability for new jobs, and minimizes the need for migration.

Restrict resource usage with VMs

Users of HPC applications cannot always predict the memory or CPU usage of a job. Without Dynamic Cluster, a job might unexpectedly use more resources than it asked for and interfere with other workload running on the execution host.

When Dynamic Cluster jobs run on a VM, one physical host can run many jobs, and the job is isolated in its environment. For example, a job that runs out of memory and fails will not interfere with other jobs running on the same host.

Dynamic Cluster concepts

- Dynamic Cluster hosts:
  - Any number of physical hosts in the LSF cluster can become Dynamic Cluster hosts. Dynamic Cluster hosts are physical machines that are managed by Platform Cluster Manager and can run LSF jobs. Dynamic Cluster manages the hypervisors on which to run virtual machines.
  - Dynamic Cluster hosts are identified in LSF by tagging them with the resource `dchost` in the LSF cluster file.
  - The remaining physical hosts in the cluster are ordinary LSF hosts that cannot be repurposed based on workload demand.
- Job VMs:
  - The virtual machines that are treated specially by LSF in Dynamic Cluster. They are a special kind of LSF host that serves only as an execution container for the LSF job, and not as a scheduling unit.
  - These machines are identified in LSF by the resource `jobvm`. The suggested installation described in this guide configures all dynamically-created virtual machines to provide the `jobvm` resource.
- Dynamic Cluster machine templates:

- A template can be viewed as a machine image containing an operating system and an application stack that can be used to instantiate a VM.
- Platform Cluster Manager requires that each provisioning request it receives references a template that it is aware of. This template is used to load an OS installation onto the target machine.
- Dynamic Cluster extends the notion of templates in Platform Cluster Manager. When a user from Dynamic Cluster submits a job, that user must at least specify one of the Dynamic Cluster templates, which consist of:
  - The Platform Cluster Manager template. The host that this job will run on must be an instance of this template, whether it already exists or must be provisioned.
  - An optional post provisioning script. A user defined script that is executed on the virtual machine after the virtual machine is powered on, allowing for further customization.

    The post provisioning script only executes the first time the virtual machine is powered on. The post provisioning script no longer executes on subsequent boot ups.
- Use the `any` template if you do not care which template the job runs in (Windows or Linux).

Compatibility notes

- Platform Cluster Manager must be installed to manage the virtual machines.
- To use Dynamic Cluster 9.1.3, you must complete a fresh installation of LSF 9.1.3.
- Dynamic Cluster can be enabled for some or all of the hosts in an existing cluster.
- Supported hypervisor operating systems:
  - RHEL version 6.3 KVM with the following patches:
    - `kernel-2.6.32-279.14.1.el6`
    - `libvirt-0.9.10-21.el6_3.5`
    - `qemu-kvm-0.12.1.2-2.295.el6_3.2`
  - VMware 5.x
- Supported virtual machine guest operating systems:
  - RHEL version 4.x, version 5.x, version 6.x (64-bit)
  - Windows 7 (32-bit)
  - Windows 2008 (64-bit)

Setup

Set up a starter installation of Dynamic Cluster to enable dynamic virtual machine provisioning.

Complete these steps to install LSF and the Dynamic Cluster enabled version of IBM Platform Cluster Manager Advanced Edition (Platform Cluster Manager) on your chosen hosts.

Important:

- For a VM save to disk to work properly, ensure that the swap space on your KVM hypervisors are at least double the physical memory.
- The host name resolution in the cluster must be configured correctly:

  - Each node in the cluster must have an official name and all nodes in the cluster must have a consistent view of other host's official names.
  - The DNS A and DNS PTR records on all nodes must be configured correctly.

Prepare for installation by making sure the following are available:

- A dedicated host to use as an NFS server, when using KVM hypervisors.
- A dedicated host to act as both Platform Cluster Manager management server and LSF master host
- Hosts representing your hypervisors and physical server hosts
- Platform Cluster Manager management server and agent installer binary files:
  - `pcmae_3.2.0.3_mgr_linux2.6-x86_64.bin`
  - `pcmae_3.2.0.3_agt_linux2.6-x86_64.bin`

  Note: The agent can only run on KVM hosts. There is no dedicated package for VMware hypervisors.
- Platform Cluster Manager entitlement file:
  - `pcmae_entitlement.dat`
- LSF distribution file:
  - `lsf9.1.3_linux2.6-glibc2.3-x86_64.tar.Z`
- LSF installer (**`lsfinstall`**) file:
  - `lsf9.1.3_lsfinstall.tar.Z`
- LSF entitlement file, which is one of the following files:
  - LSF Express Edition: `platform_lsf_exp_entitlement.dat`
  - LSF Standard Edition: `platform_lsf_std_entitlement.dat`
  - LSF Advanced Edition: `platform_lsf_adv_entitlement.dat`
- Dynamic Cluster add-on distribution package:
  - `lsf9.1.3_dc_lnx26-lib23-x64.tar.Z`
- Oracle database:

  Either use an existing Oracle database installation or download the following Oracle database packages:
  - Oracle Instant Client RPM packages for Oracle version 11.2.0.2.0:
    - `oracle-instantclient11.2-basic-11.2.0.2.0.x86_64.rpm`
    - `oracle-instantclient11.2-sqlplus-11.2.0.2.0.x86_64.rpm`

    After you accept the license agreement at the top of the page, download the Oracle Instant Client RPM packages from http://www.oracle.com/technetwork/topics/linuxx86-64soft-092277.html
  - Oracle Database XE package:
    - `oracle-xe-11.2.0-1.0.x86_64.rpm`

    After you accept the license agreement at the top of the page, download the Oracle Database XE package from http://www.oracle.com/technetwork/products/express-edition/downloads/index.html.

  Copy the Oracle database packages to the directory where the Platform Cluster Manager management server package is located. The Platform Cluster Manager management server installer installs the Oracle packages automatically.

Restriction:

All hypervisor hosts must be running the same operating system type and version.

Installing the Platform Cluster Manager management server

To install the Platform Cluster Manager master services in the default installation folder `/opt/platform`, complete the following installation steps on the intended management server:

1. Log in to the management server as root.
2. Navigate to the directory where the management server package is located.
3. Set the installation environment variables.

   The following environment variables assume that the management server is named `HostM` and the license file is in `/pcc/software/license/pcmae_entitlement.dat`:

   ```
   export MASTERHOST=HostM
   export LICENSEFILE=/pcc/software/license/pcmae_entitlement.dat
   export CLUSTERNAME=PCMAE_DC
   export CLUSTERADMIN=admin
   export BASEPORT=15937
   ```

   Note: `CLUSTERNAME` (the name of the Platform Cluster Manager cluster) must be different from the name of the LSF cluster.

   If you want to use management server failover, set the **`SHAREDDIR`** environment variable with the file path to the shared directory:

   ```
   export SHAREDDIR=/usr/share/platform
   ```

   To configure manager server failover for Platform Cluster Manager, refer to *Configuring master management server failover* in the *Administering IBM Platform Cluster Manager Advanced Edition* guide.
4. Run the installer binary.

   ```
   ./pcmae_3.2.0.3_mgr_linux2.6-x86_64.bin
   ```

   If the Oracle database packages are in the same directory as the management server package, the Platform Cluster Manager installer installs Oracle XE on the host.

   - If the installer installs Oracle XE and the host already has an OS account that is named `oracle`, you must enable interactive login to the `oracle` account while you install Oracle XE.
   - If you reinstall Platform Cluster Manager and want to transfer the accumulated Oracle data to your new installation, you are prompted for the credentials to access the database.

   When the Platform Cluster Manager installer prompts you to install the provisioning engine on the following prompt, select `no`:

   ```
   Do you want to install the provisioning engine on the same host as your management server?(yes/no)
   ```

   By default, the Platform Cluster Manager installer uses the following parameter values:

   - Username: isf
   - Password: isf
   - Port: 1521
   - Service name: XE
   - The installer creates `/etc/init.d/ego`.
5. Source the Platform Cluster Manager environment.
   - csh or tcsh: `source /opt/platform/cshrc.platform`
   - sh, ksh, or bash: `. /opt/platform/profile.platform`
6. Start the manager services.

```
egosh ego start
```

Installing Platform Cluster Manager agents on KVM hypervisors (KVM only)

To install the Platform Cluster Manager agent (PVMO agent) on the KVM hypervisors, complete the following installation steps on each intended hypervisor:

1. Log in to the hypervisor host as root.
2. Prepare the hypervisor host for virtualization.
   a. Use **yum** to install the required virtualization files.
      ```
      yum groupinstall Virtualization*
      ```
   b. Ensure that VT-x or AMD-V is enabled in the BIOS.
   c. Ensure that the **libvirtd** service is enabled and running.
   d. Stop and disable the **libvirt-guests** service.
      ```
      chkconfig libvirt-guests off
      service libvirt-guests stop
      ```
   e. Configure a network bridge to give the virtual machines direct access to the network.

      For more details, see *Bridged network with libvirt* in the RHEL product documentation: https://access.redhat.com/site/documentation/en-US/Red_Hat_Enterprise_Linux/6/html/Virtualization_Host_Configuration_and_Guest_Installation_Guide/sect-Virtualization_Host_Configuration_and_Guest_Installation_Guide-Network_Configuration-Network_Configuration-Bridged_networking_with_libvirt.html.

      Note: The RHEL KVM supports two methods of connecting virtual machines to the physical network: software bridge and MacVTap. Use the software bridge because MacVTap has performance issues with Windows guest operating systems.
3. Navigate to the directory where the agent package is installed.
4. Set the installation environment variables.

   The following environment variables assume that the management server is named `HostM`:

   ```
   export MASTERHOST=HostM
   export CLUSTERNAME=PCMAE_DC
   export CLUSTERADMIN=admin
   export BASEPORT=15937
   ```

   Important:

   - The values of these environment variables must match the environment variables that you used when you installed the Platform Cluster Manager management server.
   - **CLUSTERADMIN** must be a valid OS user account on the host with the same user ID and group ID as the corresponding user account on the management server.
   - **CLUSTERNAME** (the name of the Platform Cluster Manager cluster) must be different from the name of the LSF cluster.
5. Run the installer binary.
   ```
   ./pcmae_3.2.0.3_agt_linux2.6-x86_64.bin
   ```
6. Source the Platform Cluster Manager environment.

- csh or tcsh: `source /opt/platform/cshrc.platform`
- sh, ksh, or bash: `. /opt/platform/profile.platform`

7. Start the agent services.

   ```
   egosh ego start
   ```

Adding your vCenter Server host to Platform Cluster Manager (VMware only)

- Place the hypervisor hosts that will join the LSF cluster into their own VMware Data Center or VMware Cluster.
- If you are using VMware Cluster, you must disable Distributed Resource Scheduler (DRS) and High Availability (HA) in the VMware Cluster.

1. Log in to the Platform Cluster Manager web user interface (Portal) as an administrator.
2. From the Resources tab, select **Inventory** > **VMware**.
3. Click the **vCenter Servers** tab
4. Click the **Add** button to add a vCenter Server host.
5. Specify the host name, user name, and password for your vCenter Server.

It can take several minutes for Platform Cluster Manager to connect and load the vCenter inventory details.

Adding IP addresses to the Platform Cluster Manager web user interface (Portal)

To complete the Platform Cluster Manager installation, add IP address to the IP pool.

When Dynamic Cluster powers on a VM, it must be assigned an IP address by the management server from its IP pool. VM IP addresses and host names must be DNS resolvable.

1. Log in to the Platform Cluster Manager management server.
2. Source the Platform Cluster Manager environment:
   - C shell: `source /opt/platform/cshrc.platform`
   - Bourne shell: `. /opt/platform/profile.platform`
3. Authenticate with Platform Cluster Manager.

   ```
   egosh user logon -u Admin -x Admin
   ```

   The default password (for **-x**) is `Admin`.
4. Prepare a file with the IP address information. The file contains four space-delimited columns. The columns contain (in this order) the IP address, the host name, the subnet mask, the default gateway.

   For example,

   ```
   $ cat SAMPLE_INPUT_FILE
   172.27.101.184 myip101184.lsf.example.com 255.255.0.0 172.27.232.2
   172.27.101.185 myip101185.lsf.example.com 255.255.0.0 172.27.232.2
   ```

   Note:
   - Make sure the host name/IP pairs are added to the DNS server on your network.
   - All the addresses added to the IP pool must be on the same subnet.

- If a Windows guest operating system joins an Active Directory domain, make sure that the domain does not change the guest's fully qualified domain name. In addition, the guest's fully qualified domain name must exactly match the name in the IP pool.

5. Load the IP addresses into Platform Cluster Manager with the following command:

   ```
   vsh ips add -f SAMPLE_INPUT_FILE
   ```

6. View the list of available IP addresses.

   You can view the list of available IP addresses in the Portal for Platform Cluster Manager by logging in to the Portal as an administrator and navigating to **IP Pool** > **Dynamic Cluster IP Pool** in the **Resources** tab.

   You can also view the list of available IP addresses from the command line by using the following command:

   ```
   vsh ips list
   ```

Making a VM template in Platform Cluster Manager

Platform Cluster Manager creates new virtual machines that are based on a template that contains the guest operating system and the application stack. To create a VM template, you must first manually create a VM on one of your hypervisors, install the basic software components that are required for the LSF compute host, and convert it to a template by using the Portal for Platform Cluster Manager.

1. Create a VM by using the hypervisor tools.
   - For KVM hypervisors, refer to the *RedHat Virtualization Host Configuration and Guest Installation Guide*:

     https://access.redhat.com/knowledge/docs/en-US/Red_Hat_Enterprise_Linux/6/html/Virtualization_Host_Configuration_and_Guest_Installation_Guide/

     Note:
     - When you select storage for the VM, select the **Select managed or other existing storage** option and create a new volume in the default storage pool. The volume must use the qcow2 format.

       You need to create the VM on a local qcow2 disk despite the *RedHat Virtualization Host Configuration and Guest Installation Guide* stating that you must install VMs on shared network storage to support live and offline migrations. Dynamic Cluster copies the VM image onto shared storage to manage when the VM is converted to a template.
     - If you intend to migrate virtual machines with running jobs from one hypervisor host to another (live migration of VMs), select the **Cache mode** as **none** (in **Advanced options**) when you create a VM.
   - For VMware hypervisors, use the VMware vSphere Client.
2. Start the VM with your intended guest operating system.

   Note: Certain combinations of RHEL guest operating systems and hardware architectures may cause timekeeping errors to appear in the VM console. These can usually be resolved by modifying the kernel parameters in the guest OS. The following documentation gives more information on configuring guest timing management if these errors arise:
   - KVM: https://access.redhat.com/knowledge/docs/en-US/Red_Hat_Enterprise_Linux/6/html/Virtualization_Host_Configuration_and_Guest_Installation_Guide/chap-

Virtualization_Host_Configuration_and_Guest_Installation_Guide-KVM_guest_timing_management.html

- VMware: http://kb.vmware.com/selfservice/microsites/search.do?language=en_US&cmd=displayKC&externalId=1006427

3. Configure the DNS servers in your guest operating system.
4. For VMware hypervisors, install VMware Tools into your guest operating system.
5. Copy the VMTools installer to your VM.

   The installer is on your Platform Cluster Manager management server in the folder: `/opt/platform/virtualization/4.1`.
6. Install VMTools from within your VM.
   - If you are making a template for a Linux guest operating system, extract and run the VMTools installer package that you copied to your VM.
   - If you are making a template for a Windows guest operating system, install VMTools as follows:
     a. Log in to the VM as the Windows OS Administrator account.
     b. For VMware hypervisors, add the `C:\Program Files\VMware\VMware Tools` directory to your system **PATH** environment variable.

        Click **Start**, right-click **Computer**, then select **Advanced System Settings** > **Environment Variables**. Select **Path** and click **Edit**.
     c. Extract the VMTools installer package that you copied to your VM.
     d. Open a command-line window and run the `cscript install.vbs` script file to install VMTools.
7. Install the LSF server host into your VM.
   -

     If you are making a template for a Linux guest operating system, install LSF as follows:
     a. Follow the steps in the LSF installation guide for Linux.

        Make sure to specify the following parameter in the `slave.config` configuration file:

        ```
        LSF_LOCAL_RESOURCES="[resource jobvm]"
        ```
     b. Run the **hostsetup** and **chkconfig** commands to configure the host to not start LSF when the VM starts:

        ```
        # hostsetup --boot="n"
        # chkconfig lsf off
        ```
   -

     If you are making a template for a Windows guest operating system, install LSF as follows:
     a. Follow the steps in the LSF installation guide for Windows.
     b. Edit the `lsf.conf` file and specify the **LSF_USER_DOMAIN** parameter in the `lsf.conf` file.
        – If all Windows accounts are local accounts, use "." as the domain:

          `LSF_USER_DOMAIN="."`
        – If the VM joins a Windows domain, specify the domain name:

          `LSF_USER_DOMAIN="`*domain_name*`"`
     c. Edit the `lsf.conf` file and specify the **LSF_LOCAL_RESOURCES** parameter:

        `LSF_LOCAL_RESOURCES="[resource jobvm]"`

d. Create a Windows OS account with the same name as the LSF cluster administrator (for example, `lsfadmin`).

   Note: The same Windows user account in all templates must use the same password.

8. For KVM hypervisors, if your VM has a mounted CD-ROM device and bound ISO file, use **`virt-manager`** to unmount the ISO file for the CD-ROM and remove the CD-ROM device from the virtual machine.
9. Power off your VM.
10. Log in to the Portal for Platform Cluster Manager as an administrator.
11. Add storage repositories to your VM.
    - For KVM hypervisors, Platform Cluster Manager manages storage repositories. Use the Platform Cluster Manager Portal to add a storage repository:

      Click the **Resources** tab, then navigate to **Inventory** > **KVM** > **Storage List** and click **Add Storage**.
    - For VMware hypervisors, VMware vCenter manages storage repositories. In a Dynamic Cluster-enabled LSF cluster, all VMs might be started on any hypervisor host, so the VM data store must make all VMs disks available to all hypervisors.

    For more information on storage repositories, see *Managing storage repositories* in the Platform Cluster Manager Administration Guide.
12. Convert the VM into a template.

    For VMware, Platform Cluster Manager supports two types of templates: standard templates and snapshot templates. With a standard template, the full VM disk must be copied before a new VM can be powered on. Depending on the size of the disk and the performance of your storage infrastructure, a full copy of a VM disk can take minutes or hours to complete. A snapshot template uses copy on write to instantly clone the VM disk from the template. Local disk intensive applications have better performance with a standard template.
    - Create a KVM standard template:
      a. Log in to the Platform Cluster Manager Portal as an administrator.
      b. Click the **Resources** tab, then navigate to **Inventory** > **KVM** in the navigation tree.
      c. Click the **Machines** tab in the main window.
      d. Select your VM from the list.
      e. Click **Manage** and select **Convert to Template**.
    - Create a VMware standard template:
      a. Log in to the Platform Cluster Manager Portal as an administrator.
      b. Click the **Resources** tab, then navigate to **Inventory** > **VMware** > ***vCenter_host_name*** in the navigation tree.
      c. Click the **Machines** tab in the main window.
      d. Select your VM from the list.
      e. Click **Manage** and select **Convert to Template**.
    - Create a VMware snapshot template:
      a. In the VMware vSphere Client, create a snapshot of your VM.
      b. Log in to the Platform Cluster Manager Portal as an administrator.

c. Click the **Resources** tab, then navigate to **Inventory** > **VMware** in the navigation tree.
d. Select your vCenter Server in the navigation tree.
e. Click the **VM Snapshots** tab in the main window.
f. Select your VM snapshot from the list.
g. Click **Set as template**.

13. Add a post-provisioning script to the template.

    For more information, see "Create a post-provisioning script" on page 15.

Installing the LSF master host

1. Follow the steps in *Installing LSF* to install the LSF master host.

   The following `install.config` parameter is required for Dynamic Cluster to work:

   - **`ENABLE_DYNAMIC_HOSTS="Y"`**

2. Source the LSF environment
   - csh or tcsh: `source /opt/lsf/conf/cshrc.platform`
   - sh, ksh, or bash: `./opt/lsf/conf/profile.platform`
3. Extract the Dynamic Cluster add-on distribution package and run the setup script.
4. If you are using Windows guest operating systems, use the **`lspasswd`** command to register the password for the account you created.

   Use the same domain name as the one you specified in the `LSF_USER_DOMAIN` parameter.

   For example, use `".\lsfadmin"` for local accounts or `"`*domain_name*`\lsfadmin"` for a Windows domain.

The rest of this guide assumes that you have used the directory `/opt/lsf` as your top-level LSF installation directory.

Enable Dynamic Cluster in LSF

Complete the following steps to make your LSF installation aware of the Dynamic Cluster functionality and mark hosts in your cluster as valid for dynamic provisioning.

1. Install the Dynamic Cluster add-on package (`lsf9.1.3_dc_lnx26-lib23-x64.tar.Z`).

   For installation instructions, extract the package and refer to the README file in the package.

2. Create the `dc_conf.`*`lsf_cluster_name`*`.xml` file in `$LSF_ENVDIR`.

   A template for `dc_conf.`*`lsf_cluster_name`*`.xml` is provided for easy setup.

   a. Copy the template file `TMPL.dc_conf.CLUSTER_NAME.xml` from `/opt/lsf/9.1/misc/conf_tmpl/` into `$LSF_ENVDIR` .
   b. Change the `<Templates>` section in the file to match the templates that you created and make other required changes for your configuration.
   c. If you set the **`SHAREDDIR`** environment variable when installing the Platform Cluster Manager management server (to enable management server failover), change the file path in the **`DC_CONNECT_STRING`** parameter from the default `/opt/platform` to the value that you set for the **`SHAREDDIR`** environment variable.

For example, if you set **SHAREDDIR** to /usr/share/platform by running the following command:

```
export SHAREDDIR=/usr/share/platform
```

Navigate to the **DC_CONNECT_STRING** parameter and change the file path as follows:

```
<Parameter name="DC_CONNECT_STRING">
    <Value>Admin::/usr/share/platform</Value>
</Parameter>
```

d. Rename the file to dc_conf.*lsf_cluster_name*.xml.

For example, for KVM hosts:

```
# cat /opt/lsf/conf/dc_conf.lsf_cluster_name.xml
<?xml version="1.0" encoding="UTF-8"?>
<dc_conf xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <ParametersConf>
        <Parameter name="DC_VM_MEMSIZE_DEFINED">
            <memsize>8192</memsize>
            <memsize>4096</memsize>
            <memsize>2048</memsize>
            <memsize>1024</memsize>
        </Parameter>
        <Parameter name="DC_VM_RESOURCE_GROUPS">
            <Value>KVMRedHat_Hosts</Value>
        </Parameter>
        <Parameter name="DC_CONNECT_STRING">
            <Value>Admin::/opt/platform</Value>
        </Parameter>
    </ParametersConf>
    <Templates>
        <Template>
            <Name>RH_VM_TMPL</Name>
            <PCMAE_TemplateName>isf_rhel56vm_tmpl</PCMAE_TemplateName>
            <PostProvisioningScriptName>start-lsf.sh</PostProvisioningScriptName>
            <PostProvisioningScriptArguments/>
            <Description>RHEL 5.6 Virtual Machine</Description>
            <RES_ATTRIBUTES>
                <bigmem />
                <hostType>linux</hostType>
                <OSTYPE>linux2.6-glibc2.4</OSTYPE>
            </RES_ATTRIBUTES>
        </Template>
        <Template>
            <Name>RH_KVM</Name>
        </Template>
    </Templates>
    <ResourceGroupConf>
        <HypervisorResGrps>
            <ResourceGroup>
                <Name>KVMRedHat_Hosts</Name>
                <Template>RH_KVM</Template>
                <MembersAreAlsoPhysicalHosts>Yes</MembersAreAlsoPhysicalHosts>
            </ResourceGroup>
        </HypervisorResGrps>
    </ResourceGroupConf>
</dc_conf>
```

For VMware hosts, make the following changes to the previous example file:

a. Change the **ParametersConf** node:

Change the **DC_VM_RESOURCE_GROUPS** parameter to match the name of your VMware resource group.

To determine the name of the VMware resource group, use the Platform Cluster Manager Portal:

1) Log in to the Platform Cluster Manager Portal as an administrator.

2) Click the **Resources** tab, then navigate to **Inventory** > **VMware** in the navigation tree.
3) Click the **Resource Group** tab in the main window.
4) Locate the name of your resource group in the **Resource Group Name** column of the table.

b. Change the **ResourceGroupConf** node:

1) Change the name of the hypervisor resource group to the same resource group name used in the **ParametersConf** node.

   The hypervisor resource group is found in the following XML path: `HypervisorResGrps\ResourceGroup\Name`.
2) Delete the `HypervisorResGrps\ResourceGroup\Template` node.
3) Change the value of the **MembersAreAlsoPhysicalHosts** parameter to `No`.

If you set the SHAREDDIR environment variable when installing the Platform Cluster Manager managment server, change the file path in the **DC_CONNECT_STRING** parameter from the default `/opt/platform` to the value that you set for the **SHAREDDIR** environment variable.

The **DC_VM_MEMSIZE_DEFINED** parameter specifies the possible virtual machine memory sizes that will be created. Jobs that specify resource requirements that fall between these values will run in VMs whose memory allocations are rounded up to the next highest value in this list.

The **<Templates>** section defines all Dynamic Cluster templates. Each **<Template>** section has the following parameters:

- **<Name>**: REQUIRED. Provide a unique name for the template.
- **<PCMAE_TemplateName>**: Optional. The Platform Cluster Manager template name. The specified template is used for provisioning through Dynamic Cluster. The same Platform Cluster Manager template can be used by different Dynamic Cluster templates at the same time.
- **<PostProvisioningScriptName>**: OPTIONAL. The script to be run once the provision is finished. You can upload post-provisioning script files through the Portal for Platform Cluster Manager.
- **<PostProvisioningScriptArguments>**: OPTIONAL. Arguments to be passed to the post-provisioning script defined in **<PostProvisioningScriptName>** when the script is run.
- **<Description>**: OPTIONAL. The description of the template.
- **<RES_ATTRIBUTES>**: OPTIONAL. Defines resource names and initial values that are available as resource requirements in the template. Each resource is defined as an element with the same name as the resource name. Numeric or string resource values are defined within the element (between the opening and closing tags) while Boolean resources are specified without a defined value within the element. The resource names must be defined in `lsf.shared`, otherwise the resource is ignored in this element.

The **<ResourceGroupConf>** section defines the resource groups.

Hypervisor resource groups are defined in a **<HypervisorResGrps>** section containing one or more **<ResourceGroup>** sections. Each **<ResourceGroup>** section has the following parameters:

- **<ResourceGroup>**: Contains the detailed configuration of a resource group.
- **<Name>**: REQUIRED. The unique name of the resource group.
- **<Template>**: REQUIRED. The Dynamic Cluster template name associated with the hypervisor hosts in the resource group.

- **<MembersAreAlsoPhysicalHosts>**: OPTIONAL. The parameter has the following values:
  - Yes - The member hosts belonging to this resource group can accept physical machine jobs. When retrieving Dynamic Cluster host status, type PM_HV is displayed. This is the default value for KVM hosts.
  - No - The member hosts belonging to this resource group cannot run physical machine jobs, and the type will be displayed as HV. This is the default value for VMware hosts because VMware hypervisor hosts do not support running jobs on the physical machine.

  The default the value is No.

See the Reference section of this guide for details about other dc_conf.*lsf_cluster_name*.xml parameters.

3. Add desired hypervisors and physical hosts to the cluster file and tag them dchost.

```
# cat /opt/lsf/conf/lsf.cluster.lsf_cluster_name
Begin Host
HOSTNAME  model    type        server r1m  mem  swp  RESOURCES    #Keywords
#apple    Sparc5S  SUNSOL       1     3.5  1    2   (sparc bsd)
#Example
...
<Master>  !        !            1     3.5  ()   ()  (mg)
<Host1>   !        !            1     3.5  ()   ()  (dchost)
<Host2>   !        !            1     3.5  ()   ()  (dchost)
...
<HostN>   !        !            1     3.5  ()   ()  (dchost)
End Host
```

4. Edit lsb.params and increase the value of PREEMPTION_WAIT_TIME.

   The default value in LSF is 5 minutes, but this is too short and can cause performance problems with Dynamic Cluster.

```
# cat /opt/lsf/conf/lsbatch/lsf_cluster_name/configdir/lsb.params
...
Begin Parameters
DEFAULT_QUEUE  = normal   #default job queue name
MBD_SLEEP_TIME = 20       #mbatchd scheduling interval (60 secs is default)
SBD_SLEEP_TIME = 15       #sbatchd scheduling interval (30 secs is default)
JOB_ACCEPT_INTERVAL = 1   #interval for any host to accept a job
                          # (default is 1 (one-fold of MBD_SLEEP_TIME))
ENABLE_EVENT_STREAM = n   #disable streaming of lsbatch system events
PREEMPTION_WAIT_TIME=1800 #at least 30 minutes
End Parameters
```

5. Edit lsf.cluster and define LSF_HOST_ADDR_RANGE=*.*.*.* to enable LSF to detect dynamic job VMs.
6. If you want to use hyperthreading, edit lsf.conf and define LSF_DEFINE_NCPUS=threads.

   If you do not want to use hyperthreading, disable hyperthreading in the BIOS of your hypervisor hosts and reboot the hosts.

Create a post-provisioning script

You can define a *post-provisioning script* in Dynamic Cluster to run when a virtual machine of a given template is first started. For example, this functionality allows you to further customize the VM instance configurations by adding users, installing applications, or running any other commands you need before running a VM job.

The name and arguments of the post-provisioning script for each given template is specified in the template definition of the Dynamic Cluster configuration file.

Enable your script by copying it to the post provisioning scripts directory on your Platform Cluster Manager management server. The post provisioning scripts directory is the `virtualization/conf/postProvisionScript` subdirectory of either the top level Platform Cluster Manager installation directory, or the shared directory if you are using Platform Cluster Manager management server failover.

Note: If Platform Cluster Manager is already started, you must restart the VMOManager service after copying the script to the `postProvisionScript` directory:

```
egosh user logon -u Admin -x Admin
egosh service stop VMOManager
egosh service start VMOManager
```

Verify the installation

After completing the installation steps, start the LSF services on your master host and run the **bdc host** and **bdc tmpl** commands to verify that your cluster is up and running:

```
# bdc host
NAME         STATUS       TYPE    TEMPLATE  CPUS  MAXMEM    RESGROUP  NPMJOBS
dc-kvm2      on           HV_PM   RH_KVM    4     7998 MB   KVMRedHa  0
dc-kvm1      on           HV_PM   RH_KVM    4     7998 MB   KVMRedHa  0
```

```
# bdc tmpl
NAME                MACHINE_TYPE        RESGROUP
RH_KVM              VM                  KVMRedHat_Hosts
```

Update the Platform Cluster Manager entitlement file

If you update the Platform Cluster Manager entitlement file, restart the **ego** service by running **egosh ego restart**.

To verify that the entitlement file is working, run **egosh resource list** and verify that there are no error messages about entitlement files.

Using Dynamic Cluster

This section contains information for all users of Dynamic Cluster. It describes how to submit and monitor jobs and demonstrates some basic commands to query job status and history.

Submitting Dynamic Cluster jobs

Configure Dynamic Cluster templates

The Dynamic Cluster template contains virtual machine template information necessary for Dynamic Cluster jobs.

Defining Dynamic Cluster jobs guarantees that your jobs will run on Dynamic Cluster hosts (that is, hosts that are marked dchost in the cluster file) if your jobs are VM jobs. This means that submitting Dynamic Cluster jobs using the **DC_MACHINE_TYPE=vm** application profile parameter or the `dc_mtype=vm` guarantees that jobs will run in Dynamic Cluster VMs.

There are two ways to submit Dynamic Cluster jobs:

- Define Dynamic Cluster templates and other Dynamic Cluster parameters in the LSF application profile and submit the job with the application name.
- Define Dynamic Cluster templates and other Dynamic Cluster parameters and submit the job with the **bsub** command.

You cannot combine Dynamic Cluster parameters on the **bsub** command line with Dynamic Cluster parameters defined in the LSF application profile. If you define Dynamic Cluster templates on the command line, Dynamic Cluster parameters in the LSF application profile are ignored.

Submit jobs using application profiles

The following parameters are supported in `lsb.applications`:

DC_MACHINE_TEMPLATES=*template_name***...**

Specify all Dynamic Cluster machine templates that this job can use. The Dynamic Cluster and LSF scheduler may run the job on any suitable host.

DC_MACHINE_TYPE=vm

Specify this parameter if you require a VM for the job. By default, the system provisions any machine.

DC_JOBVM_PREEMPTION_ACTION=savevm

Specify this parameter to save the VM when jobs from this application profile are preempted. By default, low priority jobs on the VM will not be considered as preemptable and will keep running until it completes.

Submit jobs using bsub

If the LSF application profile does not define the Dynamic Cluster machine template, the following options are supported with **bsub**:

-dc_tmpl template_name...

Specify the name of one or more Dynamic Cluster templates that the job can use. Using this option makes the job use Dynamic Cluster provisioning.

For example, to submit Dynamic Cluster jobs that can run on machines provisioned using the Dynamic Cluster template named "DC3" or "DC4":

-dc_tmpl "DC3 DC4"

When you define Dynamic Cluster templates on the command line, DC_MACHINE_TEMPLATES in `lsb.applications` is ignored.

-dc_mtype vm

If you used **bsub -dc_tmpl**, and you want the Dynamic Cluster job to be a VM job, you must use the bsub option **-dc_mtype vm**.

If no value is specified for **-dc_mtype**, Dynamic Cluster jobs run on any machine.

When you define Dynamic Cluster templates on the command line, DC_MACHINE_TYPE in `lsb.applications` is ignored.

-dc_vmaction *action*

If you used **bsub -dc_tmpl** and **bsub -dc_mtype**, and you want to specify an action on the VM if this job is preempted, you must use the bsub option **-dc_vmaction** *action*.

The following are a list of preemption actions that you can specify with this option:

- **-dc_vmaction savevm**: Save the VM.

Saving the VM allows this job to continue later on. This option defines the action that the lower priority (preempted) job should take upon preemption, not the one the higher priority (preempting) job should initiate.

- **-dc_vmaction livemigvm**: Live migrate the VM (and the jobs running on them) from one hypervisor host to another.

  The system releases all resources normally used by the job from the hypervisor host, then migrates the job to the destination host without any detectable delay. During this time, the job remains in a RUN state.

- **-dc_vmaction requeuejob**: Kill the VM job and resubmit it to the queue.

  The system kills the VM job and submits a new VM job request to the queue.

Note:

By default, a low priority VM job will not be preempted if this parameter is not configured. It will run to completion even if a higher priority job needs the VM resources.

When you define the preemption action on the command line, DC_VMJOB_PREEMPTION_ACTION in `lsb.applications` is ignored.

Find available templates and application profiles

To see information about available Dynamic Cluster templates, run the **bdc** command on your LSF master host:

```
# bdc tmpl
NAME                MACHINE_TYPE        RESGROUP
RH_VM_TMPL          VM                  KVMRedHat_Hosts
RH_KVM              -                   -
```

To find application profiles that will submit Dynamic Cluster jobs, and see which templates they use, run the **bapp** command:

```
# bapp -l
APPLICATION NAME: AP_PM
 -- Dynamic Cluster PM template
STATISTICS:
   NJOBS     PEND      RUN    SSUSP    USUSP      RSV
       2        0        2        0        0        0
PARAMETERS:
DC_MACHINE_TYPE: PM
DC_MACHINE_TEMPLATES: DC_PM_TMPL
------------------------------------------------------------------------------
APPLICATION NAME: AP_VM
 -- Dynamic Cluster 1G VM template
STATISTICS:
   NJOBS     PEND      RUN    SSUSP    USUSP      RSV
       0        0        0        0        0        0
PARAMETERS:
DC_MACHINE_TYPE: VM
DC_VMJOB_PREEMPTION_ACTION: savevm
DC_MACHINE_TEMPLATES: DC_VM_TMPL
```

Resizable or chunk jobs

You cannot submit Dynamic Cluster jobs as resizable or chunk jobs. LSF rejects any Dynamic Cluster job submissions with resizable or chunk job options.

Define virtual resources

When submitting a single Dynamic Cluster job to a virtual machine, LSF will by default request a single CPU virtual machine with at least 512MB of memory. Users who require more virtual CPUs or more memory for their jobs can request these resources using the following **bsub** command line options:

-n *num_slots*

Requests a virtual machine instance with *num_slots* CPUs. When Dynamic Cluster powers on the virtual machine allocated for this job, it sets its vCPUs attribute to match the number of slots requested using this parameter. The default value is 1.

Note:

In the current release, a VM job can only run on a single VM on a single host, therefore at least one host in your cluster should have *num_slots* physical processors.

-R "rusage[mem=*integer***]"**

Specifies the memory requirement for the job, to make sure that it runs in a virtual machine with at least *integer* MB of memory allocated to it. This value determines the actual memory size of the virtual machine for the job, as defined by the DC_VM_MEMSIZE_DEFINED parameter in `dc_conf.`*`LSF_cluster_name`*`.xml`, and the DC_VM_MEMSIZE_STEP parameter in `lsb.params` .

Monitor Dynamic Cluster jobs

Use the LSF **bjobs** and **bhist** commands to check the status of Dynamic Cluster jobs.

Check the status of Dynamic Cluster jobs (bjobs)

bjobs -l indicates which virtual machine the Dynamic Cluster job is running on:

```
# bjobs -l 1936
Job <1936>, User <root>, Project <default>, Application <AP_vm>, Status <RUN>,
                     Queue <normal>, Command <myjob>
Thu Jun  9 00:28:08: Submitted from host <vmodev04.corp.com>, CWD
                      </scratch/user1/testenv/lsf_dc/work/
                     cluster_dc_/dc>, Re-runnable;
Thu Jun  9 00:28:14: Started on <host003>, Execution Home </root>, Execution CWD
                      </scratch/user1/testenv/lsf_dc/work/
cluster_dc/dc>, Execution rusage <[mem=1024.00]>;
Thu Jun  9 00:28:14: Running on virtual machine <vm0>;
Thu Jun  9 00:29:01: Resource usage collected.
                     MEM: 3 Mbytes;  SWAP: 137 Mbytes;  NTHREAD: 4
                     PGID: 11710;  PIDs: 11710 11711 11713

 SCHEDULING PARAMETERS:
           r15s   r1m  r15m   ut      pg    io   ls    it    tmp    swp    mem
 loadSched   -     -     -     -       -     -    -     -     -      -      -
 loadStop    -     -     -     -       -     -    -     -     -      -      -
```

Display machine provisioning information (bhist)

Use **bhist -l** to display machine provisioning information such as the history of provisioning requests initiated for the job, as well as their results:

```
# bhist -l 1936
Job <1936>, User <root>, Project <default>, Application <AP_vm>, Command <myjob>
Thu Jun  9 00:28:08: Submitted from host <vmodev04.corp.com>, to
                     Queue <normal>, CWD </scratch/user1/testenv/lsf_dc/work/
                     cluster_dc/dc>, Re-runnable;
Thu Jun  9 00:28:14: Provision <1> requested on 1 Hosts/Processors <host003>;
Thu Jun  9 00:28:14: Provision <1> completed; Waiting 1 Hosts/Processors <vm0>
                     ready;
Thu Jun  9 00:28:14: Dispatched to <vm0>;
Thu Jun  9 00:28:14: Starting (Pid 11710);
Thu Jun  9 00:28:14: Running with execution home </root>,
                     Execution CWD </scratch/user1/testenv/lsf_dc/work/
                     cluster_dc/dc>, Execution Pid <11710>,Execution rusage <[mem=1024.00]>;
Summary of time in seconds spent in various states by  Thu Jun  9 00:30:53
  PEND     PSUSP    RUN      USUSP    SSUSP    UNKWN    TOTAL
  6        0        159      0        0        0        165
```

Display recent provisioning action information (bdc action)

Use **bdc action** to display information about recent provisioning actions. This command shows information from memory.

```
# bdc action
REQ_ID  JOB_ID       STATUS  BEGIN                END                  NACT
10075   10449        done    Thu Apr  4 15:45:00  Thu Apr  4 15:45:26  1
10076   10461        done    Thu Apr  4 15:45:06  Thu Apr  4 15:46:06  2
10077   -            done    Thu Apr  4 15:45:26  Thu Apr  4 15:45:56  1
10078   -            done    Thu Apr  4 15:45:56  Thu Apr  4 15:46:26  1
10079   10461        done    Thu Apr  4 15:48:56  Thu Apr  4 15:49:46  1
10080   10453        done    Thu Apr  4 15:55:46  Thu Apr  4 15:56:16  1
10081   10456        done    Thu Apr  4 15:55:46  Thu Apr  4 15:56:16  1
10082   10454        done    Thu Apr  4 15:55:46  Thu Apr  4 15:56:16  1
```

Use **bdc action -p** *prov_id* to display information about a specific provisioning action by specifying its provisioning ID and **bdc action -l -p** *prov_id* to display details on the provisioning actions associated with the specific provisioning ID:

```
# bdc action -p 376
REQ_ID  JOB_ID       STATUS  BEGIN                END                  NACT
376     490          done    Mon Jun 30 10:10:24  Mon Jun 30 10:11:31  2
```

```
# bdc action -l -p 376
REQ_ID<376>
JOB_ID       STATUS  BEGIN                      END                        NACT
490          done    Mon Jun 30 10:10:24 2014  Mon Jun 30 10:11:31 2014  2

HOSTS           ac-kvm1

<Action details>
ACTIONID       1.1.1
ACTION         INSTALL_VM
STATUS         done
TARGET         36a7486f-a468-4feb-85eb-e3ecd7bc877b
HOSTNAME       kvmvm7
DC_TEMPLATE    kvmrh62
HYPERVISOR     ac-kvm1

ACTIONID       1.2.1
ACTION         NEW_VM
STATUS         done
TARGET         36a7486f-a468-4feb-85eb-e3ecd7bc877b
HOSTNAME       kvmvm7
DC_TEMPLATE    kvmrh62
HYPERVISOR     ac-kvm1
```

Use **bdc action -j** *job_id* to display the provisioning actions associated with a specific job by specifying its job ID and **bdc action -l -j** *job_id* to display details on the provisioning actions associated with the specific job.

```
# bdc action -j 490
REQ_ID  JOB_ID       STATUS  BEGIN                END                  NACT
376     490          done    Mon Jun 30 10:10:24  Mon Jun 30 10:11:31  2
```

```
# bdc action -l -j 490
REQ_ID<376>
JOB_ID       STATUS  BEGIN                      END                        NACT
490          done    Mon Jun 30 10:10:24 2014  Mon Jun 30 10:11:31 2014  2

HOSTS           ac-kvm1

<Action details>
ACTIONID       1.1.1
ACTION         INSTALL_VM
STATUS         done
TARGET         36a7486f-a468-4feb-85eb-e3ecd7bc877b
HOSTNAME       kvmvm7
DC_TEMPLATE    kvmrh62
HYPERVISOR     ac-kvm1

ACTIONID       1.2.1
ACTION         NEW_VM
STATUS         done
TARGET         36a7486f-a468-4feb-85eb-e3ecd7bc877b
HOSTNAME       kvmvm7
DC_TEMPLATE    kvmrh62
HYPERVISOR     ac-kvm1
```

Display machine provisioning request history (bdc hist)

Use **bdc hist** to display historic information about machine provisioning requests. This command shows information from the event log files. The options for this command are similar to **bdc action**, including the use of **-p** to display information on a specific provisioning action and **-j** to display information on a specific job. However, if a provisioning action fails, **bdc hist** also shows the error message from Platform Cluster Manager. For example, the last line in the following output is the error message from Platform Cluster Manager:

```
# bdc hist -l -p 4029
Provision request <4029> for Job <7653>
Wed Mar  6 12:59:02: Requested on 1 Hosts <hb05b15.mc.platformlab.ibm.com>; Power on 1 Machine with
                     Template <WIN2K8> Processors <1> Memory <1024 MB>
Wed Mar  6 12:59:03: Requested Power on 1 Machine <1ac50039-7851-4f30-acd3-c5b4701afd48>
Wed Mar  6 12:59:19: Failed Power on 1 Machine <1ac50039-7851-4f30-acd3-c5b4701afd48>
Wed Mar  6 12:59:19: Request failed: com.platform.rfi.manager.exceptions.RFIMachineNotFoundException
                     : Machine ID 1ac50039-7851-4f30-acd3-c5b4701afd48 is not found.
```

Commands with job counters combine PROV and RUN job states

Certain LSF commands report job details, while others only report counters (for example, 10 RUN jobs, 15 PEND jobs). The commands that only report counters, which includes **bqueues** and **bapp**, treat PROV jobs as identical to RUN jobs, so the counter for RUN jobs also includes PROV jobs. This is because PROV is a special type of RUN job: it is basically a job in a RUN state with an active provision action.

For example, if there are 10 RUN jobs and 10 PROV jobs, commands that report job details (such as **bjobs** and **bhist**) report 10 RUN jobs and 10 PROV jobs, while commands that report job counters (such as **bqueues** and **bapp**) report 20 RUN jobs.

Advanced features and administration

This section describes features of interest to administrators.

Managing VM memory

Configure Dynamic Cluster to manage the amount of memory for VMs.

VM job memory requirement

The scheduler will match the job to a VM that has sufficient memory. The VMs created by LSF may have standardized memory sizes, so it is normal for a job to run on a VM that has more memory than the job requires.

- If the VM job does not have any other memory requirement defined, the memory requirement depends on DC_VMJOB_MEM_REQ in `lsb.params`.

  The default value of this parameter is 1024 MB, so by default the memory requirement for a VM job is 1024 MB.
- If the VM job is submitted using an application profile, the memory requirement depends on RES_REQ in `lsb.applications`.
- To explicitly define the memory requirement for a VM job, overriding the application profile, define the memory requirement with the **bsub -R "rusage[mem=mem_req]"** option.

  For example, to request 2048 MB of memory:

  ```
  bsub -R "rusage[mem=2048]" myjob
  ```

VM memory size

The minimum memory size for a new VM cannot be less than 512 MB. The minimum memory size can be greater than 512 MB depending on configuration.

The VMs should be created with standardized memory size, to prevent the hypervisor hosts from being fragmented with multiple VMs of different size. These VMs can easily be reused for jobs with similar memory requirements.

There are two ways to control the memory sizes of new VMs:

- Define fixed values.
- Allow any value that is a multiple of the step size.

If memory sizes are defined as fixed values, the step size configuration is ignored.

Fixed VM memory sizes

When you define absolute values for memory size, the memory size of any new VM is the smallest of all the choices that satisfy the job's resource requirement.

For example, if the `dc_conf.`*`lsf_cluster_name`*`.xml` parameter DC_VM_MEMSIZE_DEFINED is set with the following values:

```
512
1024
2048
4096
```

Then a job that requires 800 MB memory will run on a VM created with 1024 MB memory. If a job requires 1025 MB memory, a VM is created with 2048 MB memory.

Note: When defining the maximum VM size allowable on your cluster, use the memory values reported by the **bdc host** command rather than those reported by **lshosts**.

VM memory size based on steps

If you do not define absolute values for memory size, define the minimum step between sizes. By default, a step is 512 MB. The memory size of any new VM is a multiple of this number. The memory size of any new VM is rounded up from the value required by the job.

For example, in lsb.params, set:

DC_VM_MEMSIZE_STEP = 512

In this example, if a job requires 500 MB memory, the VM is created with 512 MB memory. If a job requires 600 MB memory, a VM is created with 1024 MB memory. A job requiring 1025 MB memory will run in a VM created with 1536 MB memory.

Enabling CPU binding

Dynamic Cluster supports CPU binding of virtual machines in VMware and KVM environments.

By default on a multi-core hypervisor system, a virtual machine is eligible to run on any core of the hypervisor. From time to time, the hypervisor system may switch a VM to a different core based on the load conditions of the core. This behaviour may lead to performance degradation for memory intensive VMs because memory access latency increases if the VM accesses its memory at different memory node. If memory is always allocated from the local node where the VM is running, this latency will be smaller. Ideally, when powering on a virtual machine, a VM should be bound to run on a particular core for its entire lifetime, and therefore use its local memory node based on the default hypervisor policy.

Dynamic Cluster supports this CPU binding feature of virtual machines in VMware and KVM environments. Our tests show a significant performance improvement for jobs running in VMs with CPU binding compared to those running in unbound VMs.

Note:

- Dynamic Cluster will not bind a single VM to different memory nodes.
- You cannot enable CPU binding if you are using live migration in the VMware environment.
- You can enable CPU binding if you are using live migration in the KVM environment, but the VM loses the CPU binding after the live migration is complete.

Complete the following steps to enable CPU binding:

1. Edit the following file on your LSF master host:

   ```
   /opt/platform/eservice/esc/conf/services/vmoagent_service.xml
   ```

2. In the file, find the following line:

   ```
   <ego:EnvironmentVariable name="VMO_AGENT_VM_CPU_BINDING">FALSE</
   ego:EnvironmentVariable>
   ```

3. Change the value to `TRUE`:

```
<ego:EnvironmentVariable name="VMO_AGENT_VM_CPU_BINDING">TRUE</
ego:EnvironmentVariable>
```

4. Save the changes to the file.
5. Log in as the Platform Cluster Manager administrator. `# egosh user logon -u Admin -x Admin`
6. Restart Platform Cluster Manager:

```
# egosh service stop VMOAgent
# egosh service start VMOAgent
```

Hypervisors as physical machines

I/O operations are not efficient when executed on VMs, it is preferable to run I/O intensive jobs directly on physical machines. To maximize resource usage, Dynamic Cluster supports physical host repurposing between physical machines and hypervisors, based on job demand.

Some supported hypervisor platforms (KVM) allow physical machine jobs to run directly on the idle physical resources of a hypervisor. The physical machine jobs whose templates are satisfied by the hypervisor can run on the hypervisor without the need for any reprovisioning. The hypervisor behaves as both a physical machine and a hypervisor, depending on job demand.

VM job preemption

Preemptive scheduling in LSF allows a pending high-priority job to preempt a running job of lower priority. LSF suspends the lower-priority job and resumes it as soon as possible.

For more information on LSF preemption, see *Preemptive Scheduling* in *Administering IBM Platform LSF*.

When preempting VM jobs, you can live migrate the VM, save the VM to disk, or requeue the job (that is, kill the job and resubmit it to the queue). If a high-priority job is pending, Dynamic Cluster can preempt a lower-priority VM job, then start the high-priority job after the preemption action is finished on the hypervisor. For this to happen, the lower priority job must have a specified preemption action, and either high-priority job must be pending in a preemptive queue (a queue that can preempt other queues), or the low-priority job must belong to a preemptable queue (a queue that can be preempted by other queues).

If the preemption action is to live migrate the VM, you can also specify a second preemption action if the live migration does not start before the specified wait trigger time.

To specify a preemption action for the VM job specify the `–dc_vmaction` *action_name* option at submission time, or specify `DC_JOBVM_PREEMPTION_ACTION="`*`action_name`*`"` in `lsb.applications` (for jobs that are submitted to the specified application profile), where *action_name* is one of the following actions:

- **`-dc_vmaction savevm`**: Save the VM.

  Saving the VM allows this job to continue later on. This option defines the action that the lower priority (preempted) job takes upon preemption, not the one the higher priority (preempting) job initiates.
- **`-dc_vmaction requeuejob`**: Kill the VM job and resubmit it to the queue.

  The system kills the VM job and submits a new VM job request to the queue.

- **-dc_vmaction livemigvm**: Live migrate the VM (and the job that is running on it) from one hypervisor host to another.

  The system migrates the job to the destination host, then releases all resources that are normally used by the job from the hypervisor host. During this time, the job remains in a RUN state.

  RHEL KVM hypervisors do not support live migration with VM job checkpointing. Do not use `-dc_chkpntvm` with `-dc_vmaction livemigvm`.

  You can also specify a second preemption action to trigger if the live migration action fails. Use a space to separate the two actions and quotation marks to enclose the two actions.

  In addition, **livemigvm** has parameters to specify as arguments. Use a colon (`:`) to separate the different parameters and square brackets (`[]`) to enclose the parameters:

  wait_trigger_time=*seconds*
  If you also specified a second preemption action to trigger if the live migration fails, specifies the amount of time to wait for the live migration to start before the live migration takes the second action, in seconds. The default value is infinite (that is, Dynamic Cluster waits indefinitely for the live migration to trigger).

  livemig_max_downtime=*seconds*
  The maximum amount of time that a VM can be down during a live migration. This is the amount of time from when the VM is stopped on the source hypervisor and started on the target hypervisor. If the live migration cannot guarantee this downtime, the system continues to try the live migration again until it can guarantee this maximum downtime (or the **livemig_max_exectime** value is reached). Specify the value in seconds, or specify 0 to use the hypervisor default for the downtime. The default value is 0.

  livemig_max_exectime=*seconds*
  For KVM hypervisors only. The maximum amount of time that the system can attempt a live migration. If the live migration cannot guarantee the downtime (as specified by the **livemig_max_downtime** parameter) within this amount of time, the live migration fails. Specify the value in seconds 1 - 2147483646. The default value is 2147483646.

  For example,

```
bsub -dc_tmpl rhel58_vm -dc_mtype vm -dc_vmaction \
"livemigvm[wait_trigger_time=100:livemig_max_downtime=0:livemig_max_exectime=1000] requeuejob" \
myjob
```

  If the live migration fails (because the trigger time or run time is exceeded), the **requeuejob** action triggers.

Note: RHEL KVM hypervisors cannot live migrate a VM that has a snapshot. When you submit jobs to RHEL KVM hypervisors, do not use VM job checkpointing (**bsub -dc_chkpntvm** option) with live migration.

Live migration script

Dynamic Cluster supports the live migration of virtual machines with running jobs. The VMO agents running on both the source and target hypervisors will call out a script (`live_migrate_script` located in the `/opt/platform/virtualization/4.1/scripts` directory on the hypervisors) to perform the live migration.

The VMO agent is suspended while the script is running, but if the script is still running after a timeout period, the live migration fails and the VMO agent resumes activity. This timeout is 300 seconds by default, but you can modify this timeout period as follows:

1. Edit the following file on your LSF master host:
   ```
   /opt/platform/eservice/esc/conf/services/vmoagent_service.xml
   ```
2. In the file, find the following line:
   ```
   <ego:EnvironmentVariable
   name="VMO_AGENT_LIVE_MIGRATION_PRE_POST_SCRIPT_TIMEOUT">timeout_period</
   ego:EnvironmentVariable>
   ```
3. Change the timeout period to a new timeout period value, in seconds.
4. Save the changes to the file.
5. Log in as the Platform Cluster Manager administrator.
   ```
   # egosh user logon -u Admin -x Admin
   ```
6. After saving the changes to the file, restart Platform Cluster Manager:
   ```
   # egosh service stop VMOAgent
   # egosh ego restart
   ```

Manual live migration of VM jobs

LSF administrators can manually request a live migration of VM jobs from one hypervisor (the source hypervisor) to another hypervisor (the target hypervisor) by using the **bmig** command. The VM jobs remain running after the live migration. This is useful for removing running jobs from a hypervisor, for example, to prepare the hypervisor for maintenance.

To prevent other jobs from being dispatched onto the hypervisor being prepared for maintenance, keep the hypervisor closed by running the following command:

```
badmin hclose source_hypervisor_name
```

Manual live migration only applies to VM jobs that are running on the source hypervisor at the time of the live migration request. If the source hypervisor is a KVM hypervisor that runs both PM and VM jobs, only VM jobs are live-migrated. Manual live migration occurs after the scheduler finds space on a target hypervisor to place the VM jobs, and only the VM jobs that were running at the time of the request are live-migrated. In addition, a manual live migration only succeeds if there are available hypervisors to accept VM jobs.

To manually live migrate VM, run **bmig** with the `-dc_vmaction livemigvm` option:

```
bmig                    -dc_vmaction livemigvm [-M source_hypervisor_name]
```

Note: You cannot specify a target hypervisor using the `-m` option. The scheduler automatically finds any target hypervisor capable of hosting the VM jobs from the source hypervisor.

A manual live migration is not guaranteed, since there might not be available hypervisors to accept VM jobs, and there is no way to cancel a manual live migration once you requested it. You can set limits on the live migration request by specifying arguments in the `-dc_vmaction livemigvm` option:

```
-dc_vmaction "livemigvm[arguments]"
```

Specify any of the following optional arguments:

wait_trigger_time=*seconds*
The amount of time to wait for a target hypervisor to be available for live migration. After this time, the live migration request times out and the running VM job remains on the source hypervisor. Specify the value in seconds.

livemig_max_downtime=*seconds*
The maximum amount of time that a VM can be down during a live migration. This is the amount of time from when the VM is stopped on the source hypervisor and started on the target hypervisor. If the live migration cannot guarantee this downtime, the system continues to try the live migration again until it can guarantee this maximum downtime (or the value of `livemig_max_exectime` or the **DC_LIVEMIG_MAX_EXEC_TIME** parameter in `dc_conf.`*`cluster_name`*`.xml` is reached). Specify the value in seconds, or specify 0 to use the hypervisor default for the downtime. This argument overwrites the value of the **DC_LIVEMIG_MAX_DOWN_TIME** parameter in `dc_conf.`*`cluster_name`*`.xml`.

livemig_max_exectime=*seconds*
The maximum amount of time that the system can attempt a live migration. Specify the value in seconds 1 - 2147483646. This argument overwrites the value of the **DC_LIVEMIG_MAX_EXEC_TIME** parameter in `dc_conf.`*`cluster_name`*`.xml`.

When you use any of the optional arguments, separate multiple arguments with a colon, enclose the arguments in square brackets ([]), then enclose the entire **livemigvm** subcommand string in quotation marks.

For example,

```
bmig
"livemigvm[wait_trigger_time=600:livemig_max_downtime=0:livemig_max_exectime=600]"
```

Host memory defragmentation

Enable and configure host memory defragmentation to allow large memory jobs to run on large memory hosts.

A large memory host tends to experience memory fragmentation when small memory jobs are running on the host. Since jobs run with different memory requirements, it is unlikely for a large memory host to have large blocks memory available for large memory jobs, especially in busy clusters. Large memory jobs must wait for small memory jobs to complete before running on the large memory hosts.

When this occurs, the host memory may become underutilized when a large memory host is running a few small memory jobs, which means that the remaining memory in the host is not being utilized. Another aspect of this case is that relatively large memory jobs remain pending while waiting for these small memory jobs to finish. Large memory jobs can only run on large memory hosts, while other hosts may run small memory jobs.

In order to schedule large memory jobs, Dynamic Cluster can use live migration to move small memory jobs from large memory hosts. In this way, large memory hosts are reserved for large memory jobs while small memory jobs are live-migrated. This is host memory defragmentation. Host memory defragmentation works when jobs are using guaranteed resources and does not

rely on queue priority. Large memory jobs in lower priority queues can trigger a live migration to smaller memory jobs in a higher or equal priority queue.

How host memory defragmentation works

Host memory defragmentation is configured at the queue level. Users can submit large memory jobs (with memory resource requirements) to this queue. If the large memory job pends because there are no hosts with enough available memory to run the job, Dynamic Cluster finds and reserves a host that can run the large memory job. After reserving the host, Dynamic Cluster live-migrates the smaller VM jobs to other hosts.

The reservation for the large memory job lasts until one of the following occurs:

- The job is dispatched to the reserved host.
- The reservation times out.
- Another large memory host becomes available for the job.
- The job is preempted by a higher priority job.

In general, a reservation made by host memory defragmentation occurs as long as a large memory job is entitled to use a guarantee pool that includes a source hypervisor. Small memory jobs can find other (target) hypervisors as long as its guaranteed SLA is satisfied.

A large memory job can be a VM job or a PM job, but it must be a Dynamic Cluster job (that is, the job must be submitted with a Dynamic Cluster template). For PM jobs, the hypervisor host must be KVM and configured with a Dynamic Cluster template.

Configuring host memory defragmentation

Enable and configure host memory defragmentation at the queue level. When you enable host memory defragmentation for a queue, users can submit large memory jobs to this queue to ensure that Dynamic Cluster can make large memory hosts available to run these large memory jobs.

1. Log in as the LSF administrator to any host in the cluster.
2. Edit the `lsb.queues` file.
3. Define **DC_HOST_DEFRAG_TIMEOUT** to specify how long a host memory defragmentation reservation lasts, in minutes, until the reservation times out.

   DC_HOST_DEFRAG_TIMEOUT = *time_in_minutes*

   Specifying this parameter enables host memory defragmentation for the queue.
4. Optional: Define **DC_HOST_DEFRAG_MIN_PENDING_TIME** to specify how long a job is pending, in minutes, before it triggers a host memory defragmentation.

   DC_HOST_DEFRAG_MIN_PENDING_TIME = *time_in_minutes*

   If a job cannot find an available host with enough memory, the job pends. This parameter specifies how long the job waits for a host to become available before triggering a host memory defragmentation to make a host become available.

   The default is 0 (the job triggers a host memory defragmentation immediately if it cannot find an available host).
5. Optional: Define **DC_HOST_DEFRAG_MIN_MEMSIZE** to specify that a job triggers a host memory defragmentation if its rusage mem requirement (that is, its memory resource requirement) is larger than or equal to this value.

   DC_HOST_DEFRAG_MIN_MEMSIZE = *size_in_GB*

This parameter specifies the minimum memory that is requested before a job is considered a "large memory job" that can trigger a host memory defragmentation.

The default is 0 (any jobs with a memory resource requirement can trigger a host memory defragmentation).

6. Optional: To throttle the number of concurrent live migrations due to host memory defragmentation, edit the `lsb.params` file and specify **DC_HOST_DEFRAG_MAX_CONCURRENT_NUM_JOBS**.

   DC_HOST_DEFRAG_MAX_CONCURRENT_NUM_JOBS = *integer*

   Specify the maximum concurrent number of Dynamic Cluster jobs that can trigger a host memory defragmentation.

   Using this parameter allows you to manage any performance impact on regular scheduler, and controls the load on network and storage infrastructure (that is, to prevent "I/O storms").

By default, VM jobs are subject to live migration when their hypervisor hosts are selected for host memory defragmentation. To indicate that a job cannot be live migrated, specify `-dc_livemigvm n` at the job submission time. At the application profile level, specify `DC_LIVEMIGVM = N` and a job can be submitted to the application file. In esub scripts, specify `LSB_SUB4_DC_NOT_MIGRATABLE=Y`.

Submit a job with a large memory job requirement (that is, a job with `-R 'rusage[mem=`*memory_requirement*`]'`) to a queue configured with **DC_HOST_DEFRAG_TIMEOUT**. If there is no immediate resource available for the job (within the period as specified by the **DC_HOST_DEFRAG_MIN_PENDING_TIME** parameter), Dynamic Cluster attempts a host memory defragmentation under the following conditions:

- The job is pending because there are no hosts with enough available memory to run the job.
- There are smaller VM jobs that are running on a source hypervisor and these jobs can be live-migrated to other (target) hypervisors.

For example, if you specify **DC_HOST_DEFRAG_TIMEOUT** in `lsb.queues` for the `hostdefrag` queue, any job with a memory requirement that you submit to the `hostdefrag` queue (that is, by submitting a job with `-q hostdefrag -R 'rusage[mem=`*memory_requirement*`]'`) can trigger a host memory defragmentation if there are no available hosts to run the job (subject to the other host memory defragmentation parameters).

To verify that a queue has host memory defragmentation enabled, run `bqueues -l` *queue_name* and verify that **DC_HOST_DEFRAG_TIMEOUT** displays a value. If there are other host memory defragmentation parameters, these parameters also display their configured values.

Manually saving VMs to disk

VMs (and jobs running on them) may be saved to disk using the **bmig** command (normally used to migrate VMs).

To save a VM to disk, run **bmig** with the `-dc_vmaction savevm` option without selecting a source or target hypervisor name:

bmig `[-f ] [-J` *VM_job_name*`] [-dc_vmaction savevm]`

Distinguishing provisioning time from running time

Dynamic Cluster jobs will distinguish between provisioning time (when the jobs are in the PROV job state) and running time (when the jobs are in the RUN job state) to show a more accurate job status and be able to determine how much time jobs are spent actually running.

To enable this feature, specify `LSB_DC_DISP_PROVISION_TIME=Y` in `lsf.conf`. When enabled, the following commands distinguish provisioning time from running time:

- **bacct** shows the total provisioning time of each Dynamic Cluster job.
- **bhist -l** shows the amount of time the Dynamic Cluster jobs spent in the PROV state in addition to the RUN state.
- **bjobs** and **bjobs -sum** show Dynamic Cluster jobs in provisioning as PROV instead of RUN until the provisioning is complete and the job is started.

The **bacct**, **bhist**, and **bjobs** commands show the PROV job state when Dynamic Cluster jobs are provisioning. In addition, the job query API also uses the PROV job state, allowing other applications to distinguish between provisioning time and running time.

Certain LSF commands report job details, while others only report counters (for example, 10 RUN jobs, 15 PEND jobs). The commands that only report counters, which includes **bqueues** and **bapp**, treat PROV jobs as identical to RUN jobs, so the counter for RUN jobs also includes PROV jobs. This is because PROV is a special type of RUN job: it is basically a job in a RUN state with an active provision action.

For example, if there are 10 RUN jobs and 10 PROV jobs, commands that report job details (such as **bjobs**, **bhist**, and **bacct**) report 10 RUN jobs and 10 PROV jobs, while commands that report job counters (such as **bqueues** and **bapp**) report 20 RUN jobs.

Enabling Dynamic Cluster hosts to run any LSF job

Enable Dynamic Cluster hosts to run any non-Dynamic Cluster jobs.

By default, Dynamic Cluster hosts only run Dynamic Cluster jobs. To enable Dynamic Cluster hosts to run any non-Dynamic Cluster jobs (that is, to run jobs that do not specify any Dynamic Cluster templates), set up a default external submission (**esub**) script that modifies all non-Dynamic Cluster to run on either physical or virtual Dynamic Cluster hosts.

1. Create an external submission (**esub**) script file named `esub.myesub` in the directory specified by the **LSF_SERVERDIR** paramater in `lsf.conf`.

   Create the script file based on whether you want non-Dynamic Cluster jobs to run on physical or virtual machines.

   - To modify all non-Dynamic Cluster jobs to run on physical Dynamic Cluster hosts:

     ```
     #!/bin/sh
     #
     if [ -z "$LSB_SUB_PARM_FILE" ]; then
         # if not set do nothing
         exit 0
     fi

     exec 1>&2
     . $LSB_SUB_PARM_FILE
     ```

```
if [ -z "$LSB_SUB4_DC_TEMPLATE" ]; then
    if [ "$LSB_SUB_QUEUE" == "priority" ]; then
        echo "LSB_SUB4_DC_TEMPLATE=\"VM_T1\"" >> $LSB_SUB_MODIFY_FILE
    else
        echo "LSB_SUB4_DC_TEMPLATE=\"any\"" >> $LSB_SUB_MODIFY_FILE
    fi
fi

if [ -z "$LSB_SUB4_DC_MTYPE_VM" ] && [ -z "$LSB_SUB4_DC_MTYPE_PM" ]; then
    echo "LSB_SUB4_DC_MTYPE_ANY=Y" >> $LSB_SUB_MODIFY_FILE
fi
```

- To modify all non-Dynamic Cluster jobs to run on virtual Dynamic Cluster hosts:

```
if [ -z "$LSB_SUB4_DC_TEMPLATE" ]; then
    # Select dc template based on queue
    if [ "$LSB_SUB_QUEUE" == "QUEUE_1" ]; then
        echo "LSB_SUB4_DC_TEMPLATE=\"vm1\"" >> $LSB_SUB_MODIFY_FILE
    else
        echo "LSB_SUB4_DC_TEMPLATE=\"any\"" >> $LSB_SUB_MODIFY_FILE
    fi
fi

#Check if -dc_vmaction save is specified, if not, then select action based on queue
if [ -z "$LSB_SUB4_DC_PREEMPTACTION_VM" ]; then
    #Select dc_vmaction based on queue
    echo "LSB_SUB4_DC_PREEMPTACTION_VM=\"savevm\"" >> $LSB_SUB_MODIFY_FILE
fi

#Check if -dc_mtype vm is specified, if not, then let job running on DC VM host
if [ -z "$LSB_SUB4_DC_MTYPE_PM" ] && [ -z "$LSB_SUB4_DC_MTYPE_ANY" ]; then
    echo "LSB_SUB4_DC_MTYPE_VM=Y" >> $LSB_SUB_MODIFY_FILE
fi
```

2. Configure LSF to specify a mandatory **esub** method to apply to all job submissions.

   Edit `lsf.conf` and specify the LSB_ESUB_METHOD parameter:

   ```
   LSB_ESUB_METHOD="myesub"
   ```

3. Restart **mbatchd** for your changes to take effect.

   ```
   badmin mbdrestart
   ```

VM job checkpoint and restart

Checkpointing enables Dynamic Cluster users to pause and save the current state of memory and local disk of a VM running a job. The checkpoint files allow users to restart the VM job on the same physical server or a different physical server so that it continues processing from the point at which the checkpoint files were written.

When a user submits a VM job with a checkpoint, Dynamic Cluster saves the current state of the VM ("checkpoints") at the initial specified time, and repeats the process again when the job reaches the specified time interval. Dynamic Cluster only checkpoints the VM job when the job is running, so if the job state changes during the checkpointing period (for example, the job is finished, killed, or suspended), Dynamic Cluster does not checkpoint the VM job. Dynamic Cluster only keeps one checkpoint file for each job. If Dynamic Cluster checkpoints a VM job multiple times, the newest checkpoint file always overwrites the last checkpoint file.

Dynamic Cluster automatically restarts a checkpointed VM job only when the job status becomes `UNKNOWN`. When restarting the VM job, Dynamic Cluster restores the

VM from the last checkpoint. If there is no checkpoint for the VM job yet, LSF kills the job and requeues it, where it is rerun from the beginning with the same job ID. To use checkpointing, LSF must be able to rerun the job, either by submitting it to a rerunnable queue or by using the **bsub -r** option.

It is possible for the VM to be down but the physical execution host to be available. Therefore, it is possible for the VM to reschedule on the same execution host. To reduce the chance of repeating the failure, Dynamic Cluster places the original execution host at the end of the candidate host list, so that Dynamic Cluster attempts to reschedule the job on other execution hosts first.

If Dynamic Cluster fails to create a checkpoint, the VM job continues to run and Dynamic Cluster attempts to create another checkpoint at the next scheduled checkpoint time. The last successful checkpoint is always kept regardless of subsequent checkpoint failures.

If the VM could not be restored from the last checkpoint, the VM job cannot be restarted, so the job status remains UNKNOWN. If this occurs, LSF will continue attempting to restore the VM.

Preemption of checkpointed VM jobs

A VM job cannot be preempted if a checkpoint is in progress. The preemption occurs after the checkpoint is complete.

If a checkpointed VM job is preempted, the checkpoint for the VM job may be kept or deleted for the newly-preempted job, depending on the specified preemption action:

savevm
: The checkpoint is kept. After the VM job resumes, if the job status becomes UNKNOWN, its VM is restored from the last checkpoint.

livemigvm
: The checkpoint is kept. After the live migration finishes, if the job status becomes UNKNOWN, its VM is restored from the last checkpoint.

  RHEL KVM hypervisors do not support live migration with VM job checkpointing. Do not use `-dc_chkpntvm` with the `livemigvm` preemption action.

requeuejob
: The checkpoint is deleted, and no further checkpoint action is taken.

Submitting a VM job with checkpointing enabled

To submit a VM job with checkpointing enabled, use the **bsub -dc_chkpntvm** option or the **DC_CHKPNTVM** parameter in `lsb.applications` to specify the initial and recurring time to create a checkpoint of the VM job.

Note:

- Do not use LSF checkpointing (the **bsub -k** option) with Dynamic Cluster VM job checkpointing. If you use LSF checkpointing with VM job checkpointing, Dynamic Cluster rejects the job submission.
- RHEL KVM hypervisors do not live migrate a VM that has a snapshot. Do not use VM job checkpointing (`-dc_chkpntvm`) with live migration (`-dc_vmaction livemigvm`) for the same job.
- To specify a custom checkpoint interval for an individual job, use the **bsub -dc_chkpntvm** option:

`-dc_chkpntvm "init=`*initial_minutes interval_minutes*`"`

where *initial_minutes* is the time for the initial VM job checkpoint in minutes after the job was dispatched and *interval_minutes* is the amount of time after the previous checkpoint to create subsequent checkpoints.

For example,

```
bsub ... -r -dc_chkpntvm "init=60 15"
```

The first checkpoint is created 60 minutes after the job is dispatched, and a new checkpoint is created every 15 minutes after the previous checkpoint.

- To specify a custom checkpoint interval for all jobs, use the **DC_CHKPNTVM** parameter in `lsb.applications`:

  `DC_CHKPNTVM=`*initial_minutes interval_minutes*

  where *initial_minutes* is the time for the initial VM job checkpoint in minutes after the job was dispatched and *interval_minutes* is the amount of time after the previous checkpoint to create subsequent checkpoints.

  For example,

  ```
  DC_CHKPNTVM=120 30
  ```

  The first checkpoint is created 120 minutes after the job is dispatched, and a new checkpoint is created every 30 minutes after the previous checkpoint.

Chapter 2. Reference

Commands

The following commands are new or modified.

bdc

Synopsis

bdc *subcommand*

bdc [**-h** | **-V**]

Description

bdc provides a set of subcommands to monitor Dynamic Cluster. If there are no specified subcommands, **bdc** prompts for a subcommand from standard input.

Information about each subcommand is available through the help command.

Subcommand synopsis

The following subcommands are supported:

action

hist

param

tmpl

vm

host

help [*subcommand ...*] | **?** [*subcommand ...*]

quit

Options

subcommand

Executes the specified subcommand. See Usage section.

-h

Prints command usage to `stderr` and exits.

-V

Prints LSF release version to `stderr` and exits.

Usage

action [-l] [-w] [-p req_id...|-j job_id...]

Show information about provision actions. This subcommand shows information from memory. For information about older jobs, use **bdc hist**.

By default, only shows the following information:

REQ_ID - provision request ID. There is one provision request per job. (numeric)

JOB_ID - LSF job ID (numeric)

STATUS - status of the job's provisioning request:

- active = one or more actions in the request are in progress
- done = all actions in the request are done
- failed = the request failed to complete

BEGIN - date and time the request was made (Weekday Month dd hh:mm:ss)

END - date and time the request finished (Weekday Month dd hh:mm:ss)

NACTS - number of provision actions in the request (numeric)

-l

Long format. Shows the following additional information:

HOSTS - host names

ACTIONID - provision action ID (req_ID.subreq_ID.step_ID)

ACTION - provision action

TARGET - VM or PM machine ID

HOSTNAME - VM or PM machine name

HYPERVISOR - host name. Shows a dash "-" if the job is a PM job.

DC_TEMPLATE - Dynamic Cluster machine template used. Shows a dash "-" if not required.

STATUS - status of the provisioning action:

- active = the action is in progress
- done = the action is complete
- wait = the action is not started, it may be dependent on another action in the same provision request
- failed = the action failed to complete

-w

Wide format. Displays information without truncating it.

-p req_ID

Provision request ID.

Cannot be used with -j.

-j job_ID

Job ID.

Cannot be used with -p.

Example output

```
# bdc action
REQ_ID  JOB_ID       STATUS  BEGIN                     END                       NACT
235     1[24]        done    Tue Nov 20 11:43:46 2012  Tue Nov 20 11:43:46 2012  1

# bdc action -l
REQ_ID<235>
JOB_ID       STATUS  BEGIN                     END                       NACT
1[24]        done    Tue Nov 20 11:43:46 2012  Tue Nov 20 11:43:46 2012  1

HOSTS           ac-kvm1

<Action details>
ACTIONID       1.1.1
ACTION         IDLE_VM
STATUS         done
TARGET         4c136d83-2639-4ca0-b1d4-fe3a85aa66fe
HOSTNAME       kvmvm6
DC_TEMPLATE    rh48
HYPERVISOR     dc-kvm1
```

hist [-a] [-l | -w] [-f event_file | -n num_event_files][-t begin_time[,end_time]] [-p req_ID... | -j job_ID | -j jobID[index]...]

Show historic information about provisioning requests. This subcommand shows information from the event log files. For more information about recent jobs, use **bdc -action**.

By default, show information about unfinished requests only (exclude failed and done status).

By default, show information from the current event log file only.

By default, show the following information:

REQ_ID - provision request ID (numeric)

STATUS - status of the job's provision request (wait, active, failed, done ...)

JOB_ID - LSF job ID (numeric)

BEGIN - date and time the request was made (Weekday Month dd hh:mm:ss)

END - date and time the request finished (Weekday Month dd hh:mm:ss)

-a

Show information about all provision requests (both finished and unfinished).

-l

Long format.

-w

Wide format. Displays information without truncating it.

-f event_file

Show information from the specified event log file. Specify the event log file name. Specify either an absolute or a relative path.

The specified file path can contain up to 4094 characters for UNIX.

-n num_event_files | -n 0

Show information from the specified number of event log files. Specify an integer up to 100.

Searches the specified number of event logs, starting with the current event log and working through the most recent consecutively numbered logs. Specify 0 to specify all the event log files, up to a maximum of 100.

If you delete a file, you break the consecutive numbering, and older files are inaccessible to **bdc hist**. For example, if you specify 3, LSF searches:

`dc.events`

`dc.events.1`

`dc.events.2`

If you specify 4, LSF searches:

`dc.events`

`dc.events.1`

`dc.events.2`

`dc.events.3`

However, if `dc.events.2` is missing, both searches include only `dc.events` and `dc.events.1`.

[-t begin_time[,end_time]]

Show information about events in the specified time frame. If you do not specify the end time, the end time is the current time.

Specify time in the format yyyy/mm/dd/HH:MM. Do not include spaces in the time interval string.

For more information about the syntax, see "Time interval format" at the end of the LSF **bhist** command reference.

[-j job_ID | -j jobID[index]...]

Show information about the specified jobs. Specify the job ID or job array ID.

To search multiple files, use **-j** with **-f** or **-n**.

-l

Long format. For each request, shows detailed information about the actions in the request and their progress.

-p req_ID...

Show information about the specified provision requests only.

To search multiple files, use **-p** with **-f** or **-n**.

Example output

```
# bdc hist -l -p 1
Provision request <1> for Job <1936>
Thu Jun  9 00:28:14: Requested on 1 Hosts <host003>;
Thu Jun  9 00:28:14: Requested 1 idle Machine <d22c1e89-2fa5-4b24-9f25-f278469d915b>
Thu Jun  9 00:28:14: Request completed
```

host [-l | -w] [host_name...]

Show information about physical machines or hypervisors.

By default, only shows the following information:

NAME - host name

STATUS - host status (for example, on, off)

TYPE - machine type (for example, HV_PM)

TEMPLATE - Dynamic Cluster machine template name

CPUS - number of CPUs on the host (numeric)

MAXMEM - maximum memory on the host, in MB (numeric)

RESGROUP - resource group that the host belongs to

NPMJOBS - number of physical machine jobs on the host (numeric) directly

-l

Long format. Shows the following additional information:

PCMAE_TMPL_NAME - Platform Cluster Manager Advanced Edition template name

PPS_NAME - post-provisioning script

PPS_ARGUMENTS - arguments to the post-provisioning script

JOBIDS - job ID of jobs running on the host

MIN_TTL - minimum time to live, in seconds (numeric)

CREATION_TIME - creation time

POWER_ON_TIME - time the host powered on

POWER_OFF_TIME - time the host powered off

-w

Wide format. Displays information without truncating it.

Example output

```
# bdc host
NAME         STATUS       TYPE    TEMPLATE  CPUS  MAXMEM    RESGROUP  NPMJOBS
host000      on           PM      DC_PM_T   4     7985 MB   PCM_172_  2
host003      on           HV_PM   KVM_HV    4     7729 MB   KVMRedHa  0
```

```
# bdc host -l
NAME            host000
STATUS          on
TYPE            PM
MAXMEM          7985 MB
CPUS            4
RESGROUP        PCM_172_17_1_153-2c918127-3069788a-0130-69827ade-0027
TEMPLATE        DC1_PM
PCMAE_TMPL_NAME DC1
PPS_NAME        setup_lsf.sh
PPS_ARGUMENTS   -
JOBIDS          -
MIN_TTL         -
CREATION_TIME   Tue Jun  7 07:11:26 2011
POWER_ON_TIME   Tue Jun  7 07:11:26 2011
POWER_OFF_TIME  Tue Jun  7 07:11:26 2011

NAME            host003
STATUS          on
TYPE            HV_PM
MAXMEM          7729 MB
CPUS            4
RESGROUP        KVMRedHat_Hosts
TEMPLATE        KVM_HV
PCMAE_TMPL_NAME -
PPS_NAME        -
```

```
PPS_ARGUMENTS   -
JOBIDS          -
MIN_TTL         -
CREATION_TIME   -
POWER_ON_TIME   -
POWER_OFF_TIME  -
```

param [-l]

Show information about Dynamic Cluster configuration parameters.

By default, shows parameter name and value.

-l

Long format. Shows a description of each parameter displayed.

tmpl [-l | -w] [-i template_ID] [-n template_name][-g resource_group]

Show information about Dynamic Cluster machine templates.

By default, only shows the following information:

NAME = template name

MACHINE_TYPE = PM or VM

RESGROUP = resource group name (if type is shared).

-l

Long format. Shows the following additional information:

HV_TYPE = technical type of template hypervisor

PCMAE_TMPL_NAME = name of Platform Cluster Manager Advanced Edition template associated to this Dynamic Cluster template

PCMAE_TMPL_ID = unique template ID

PPS_NAME = name of the post-provisioning script associated to this template

PPS_ARGUMENTS = arguments of the post-provisioning script associated to this template

DESCRIPTION = description string

Cannot be used with -w.

-w

Wide format. Displays information without truncating it.

Cannot be used with -l.

-g resource_group

Show information for templates in the specified resource group.

-n template_name

Show information for the specified templates. Specify the template name.

-i template_ID

Show information for the specified templates. Specify the template ID.

Example output

```
# bdc tmpl
NAME                 MACHINE_TYPE       RESGROUP
vm1                  VM                 KVMRedHat_Hosts
DC1_PM               PM                 PCM_172_17_1_153-2c918
KVM_HV               PM                 PCM_172_17_1_153-2c918
# bdc tmpl -l
NAME               vm1
HV_TYPE            KVM
MACHINE_TYPE       VM
RESGROUP           KVMRedHat_Hosts
PCMAE_TMPL_NAME    my_tmpl1
PCMAE_TMPL_ID      de2d35bc-dba7-ddf2-5e34-3d53323d2d48
PPS_NAME           setup.sh
PPS_ARGUMENTS      -
DESCRIPTION           Dynamic Cluster template for VM
```

vm [-l | -w] [-a] [*vm_host_name* **...]**

Show information about VMs.

By default, only shows information for VMs that have a job associated, or are in the following states:

on

saving

saved

starting

shuttingdown

installing

uninstalling

bdc vm -a shows the following extra states:

unknown

off

By default, only shows the following information:

HOST_NAME

STATUS

HYPERVISOR

TEMPLATE

MAXMEM

VCPUS

JOB_ID

-a

Show information about VMs in all states.

-l

Long format. Shows the following additional information:

UUID

VM_NAME

PCMAE_TMPL_NAME

PPS_NAME

PPS_ARGUMENTS

RESGROUP

MIN_TTL

MAX_TTL

CREATION_TIME

POWER_ON_TIME

POWER_OFF_TIME

-w

Wide format. Displays information without truncating it.

vm_host_name **...**

Show information about the specified VMs only. Specify the VM host name. Use space to separate names in a list.

Example output

```
# bdc vm
HOSTNAME     STATUS       HYPERVISOR TEMPLATE  MAXMEM    VCPUS JOB_ID
vm4          on           host003    DC_VM_T   1024 MB   2     1733
vm0          on           host003    DC_VM_T   1024 MB   1     -

# bdc vm -l
HOSTNAME        vm4
UUID            4e9f9549-59b7-4e99-ba83-788808dfb2eb
STATUS          on
TEMPLATE        DC_VM_TMPL
PCMAE_TMPL_NAME isftmpl1
PPS_NAME        setup.sh
PPS_ARGUMENTS   -
HYPERVISOR      host003
MAXMEM          1024 MB
VM_NAME         _dyn_dc_2
JOB_ID          1733
VCPUS           2
RESGROUP        KVMRedHat_Hosts
MIN_TTL         0
MAX_TTL         -
CREATION_TIME   Wed Jun  8 06:05:18 2011
POWER_ON_TIME   Wed Jun  8 06:18:24 2011
POWER_OFF_TIME  Wed Jun  8 06:09:41 2011

HOSTNAME        vm0
UUID            d22c1e89-2fa5-4b24-9f25-f278469d915b
STATUS          on
TEMPLATE        DC_VM_TMPL
PCMAE_TMPL_NAME isftmpl1
PPS_NAME        setup.sh
PPS_ARGUMENTS   -
HYPERVISOR      host003
MAXMEM          1024 MB
VM_NAME         _dyn_dc_1
JOB_ID          -
VCPUS           1
RESGROUP        KVMRedHat_Hosts
MIN_TTL         0
MAX_TTL         -
CREATION_TIME   Wed Jun  8 23:17:41 2011
POWER_ON_TIME   Wed Jun  8 23:18:07 2011
POWER_OFF_TIME  Wed Jun  8 23:17:41 2011
```

bacct

-l

New **bacct -l** output displays the provisioning time of Dynamic Cluster jobs.

bapp

-l Shows Dynamic Cluster parameters:

- DC_MACHINE_TEMPLATES
- DC_MACHINE_TYPE
- DC_VMJOB_PREEMPTION_ACTION

bhist

-l

New **bhist -l** output displays provisioning requests and the status related to jobs.

If the display of provisioning time is enabled (LSB_DC_DISP_PROVISION_TIME=Y in lsf.conf), new **bhist -l** output displays the provisioning time of Dynamic Cluster jobs (in the PROV state), which is distinguished from the RUN state.

bhosts

-a

Dynamic Cluster only. Show information about all hosts, including Dynamic Cluster virtual machine hosts configured with the jobvm resource.

Default output includes only standard LSF hosts and Dynamic Cluster hosts configured with the dchost resource.

bjobs

-l

Displays Dynamic Cluster-specific information.

-sum

New **bjobs -sum** output displays the number of Dynamic Cluster jobs that are currently provisioning.

Example output

```
# bjobs -l 1936
Job <1936>, User <root>, Project <default>, Application <AP_vm>, Status <RUN>,
                     Queue <normal>, Command <myjob>
Thu Jun  9 00:28:08: Submitted from host <vmodev04.corp.com>, CWD
</scratch/qa/user1/testenv/lsf_dc/work/
                     cluster_dc/dc>, Re-runnable;
Thu Jun  9 00:28:14: Started on <host003>, Execution Home </root>, Execution CWD
 </scratch/qa/user1/testenv/lsf_dc/work
                     /cluster_dc/dc>, Execution rusage
<[mem=1024.00]>;
Thu Jun  9 00:28:14: Running on virtual machine <vm0>;
Thu Jun  9 00:29:01: Resource usage collected.
                     MEM: 3 Mbytes;  SWAP: 137 Mbytes;  NTHREAD: 4
                     PGID: 11710;  PIDs: 11710 11711 11713
 SCHEDULING PARAMETERS:
```

```
          r15s   r1m  r15m   ut      pg    io   ls    it    tmp    swp    mem
loadSched   -     -     -     -       -     -    -     -     -      -      -
loadStop    -     -     -     -       -     -    -     -     -      -      -
```

bmig

bmig [**-dc_vmaction savevm**] [**-M "***host_name ...***"**] [**-m "***host_name ...***"**] [**-u** *user_name* | **-u all**] [**-J job_name**] [*job_ID* | "*job_ID*[*index_list*]" ...]

bmig [**-dc_vmaction livemigvm**] [**-M "***host_name ...***"**] [**-u** *user_name* | **-u all**] [**-J job_name**] [*job_ID* | "*job_ID*[*index_list*]" ...]

Saves a VM job to disk.

-dc_vmaction savevm

Saves the specified VM job to disk.

If specified with `-m` *target_hypervisor_name* (and, optionally, `-M` *source_hypervisor_name*), migrates the VM job to the target hypervisor.

-dc_vmaction livemigvm
-dc_vmaction "livemigvm[*arguments***]"**

LSF administrators only. Manually live-migrates a VM job from the specified source hypervisor to another hypervisor, if available. The VM job remains running after the live migration. This is useful for removing running jobs from a hypervisor, for example, to prepare the hypervisor for maintenance.

The following arguments are optional.

wait_trigger_time=*seconds*
The amount of time to wait for a target hypervisor to be available for live migration. After this time, the live migration request times out and the running VM job remains on the source hypervisor. Specify the value in seconds.

livemig_max_downtime=*seconds*
The maximum amount of time that a VM can be down during a live migration. This is the amount of time from when the VM is stopped on the source hypervisor and started on the target hypervisor. If the live migration cannot guarantee this down time, the system will continue to retry the live migration until it can guarantee this maximum down time (or the value of `livemig_max_exectime` or the **DC_LIVEMIG_MAX_EXEC_TIME** parameter in `dc_conf.`*`cluster_name`*`.xml` is reached). Specify the value in seconds, or specify 0 to use the hypervisor default for the down time. This argument overwrites the value of the **DC_LIVEMIG_MAX_DOWN_TIME** parameter in `dc_conf.`*`cluster_name`*`.xml`.

livemig_max_exectime=*seconds*
The maximum amount of time that the system can attempt a live migration. If the live migration cannot guarantee the down time (as specified by `livemig_max_exectime` or the **DC_LIVEMIG_MAX_DOWN_TIME** parameter in `dc_conf.`*`cluster_name`*`.xml`.) within this amount of time, the live migration fails. Specify the value in seconds from 1 to 2147483646. This argument overwrites the value of the **DC_LIVEMIG_MAX_EXEC_TIME** parameter in `dc_conf.`*`cluster_name`*`.xml`.

When using any of the optional arguments, separate multiple arguments with a colon, enclose the arguments in square brackets ([]), then enclose the entire **livemigvm** subcommand string in quotation marks.

For example,

```
bmig
"livemigvm[wait_trigger_time=600:livemig_max_downtime=0:livemig_max_exectime=600]"
```

bmod

bmod [-dc_tmpl none | -dc_tmpln | -dc_vmactionn | -dc_livemigvmn]

bmod [-dc_tmpl any | "*DC_template_name ...***"] [-dc_mtype pm | -dc_mtype vm [[[-dc_vmaction savevm | livemigvm | requeuejob] | -dc_vmactionn] [-dc_livemigvm y | -dc_livemigvm n | -dc_livemigvmn]] | -dc_mtype any [[[-dc_vmaction savevm | livemigvm | requeuejob] | -dc_vmactionn] [-dc_livemigvm y | -dc_livemigvm n | -dc_livemigvmn]] [-dc_chkpntvm "[init=***initial_chkpnt_period chkpnt_period***]"]**

-dc_tmpl none | any | "template_name..."

Dynamic Cluster only.

Modifies jobs with the specified Dynamic Cluster templates that the job can use. Specify the name of one or more Dynamic Cluster templates that the job can use, or use the any or none keywords. Separate names in the list with space. Using this option makes the job use Dynamic Cluster provisioning.

Specify `-dc_tmpl none` to use none of the templates, and the job also ignores the `-dc_mtype` and `-dc_vmaction` options.

When you define Dynamic Cluster templates on the command line, DC_MACHINE_TEMPLATES in `lsb.applications` is ignored.

-dc_tmpln

Dynamic Cluster only.

Cancels previous `-dc_tmpl` settings.

-dc_chkpntvm "[init=*initial_chkpnt_period chkpnt_period***]"**

Dynamic Cluster only.

Modify the VM job checkpointing parameters by modifying the initial checkpoint time and recurring checkpoint interval.

-dc_livemigvm y | n

Dynamic Cluster only.

Modifies a job on whether it can be live migrated when its hypervisor host is selected for host memory defragmentation.

Specify **-dc_livemigvm y** and your job is live migrated if its hypervisor host is selected for host memory defragmentation.

Specify **-dc_livemigvm n** to indicate that your job cannot be live migrated.

-dc_livemigvmn

Dynamic Cluster only.

Cancels previous `-dc_livemgvm` settings.

-dc_mtype vm | pm | any

Dynamic Cluster only.

Modifies the Dynamic Cluster job to run on a particular type of machine. This value is ignored if **-dc_tmpl** is not also used or if you used **-dc_tmpl none**.

Specify **-dc_mtype vm** if you want the Dynamic Cluster job to be a VM job.

Specify **-dc_mtype pm** if you want the job to run on physical machines.

By default, Dynamic Cluster jobs run on any machines. The syntax **-dc_mtype any** is supported to specify this explicitly (physical machines are selected before virtual machines).

When you define Dynamic Cluster templates on the command line, DC_MACHINE_TYPE in `lsb.applications` is ignored.

The default value for **-dc_mtype** is **-dc_mytype any** if you modify a job that was previously not a Dynamic Cluster and specify **-dc_tmpl** without specifying a value for **-dc_mtype**.

-dc_vmaction savevm | livemigvm | requeuejob

Dynamic Cluster only.

Modifies the preemption action for the Dynamic Cluster job. This value is ignored if `-dc_tmpl` and `-dc_mtype vm` are not also used or if you used **-dc_tmpl none**. Specify **-dc_vmaction** *action* to specify an action on the VM if this job is preempted.

The following are a list of preemption actions that you can specify with this option:

- **-dc_vmaction savevm**: Save the VM.

  Saving the VM allows this job to continue later on. This option defines the action that the lower priority (preempted) job should take upon preemption, not the one the higher priority (preempting) job should initiate.
- **-dc_vmaction livemigvm**: Live migrate the VM (and the jobs running on them) from one hypervisor host to another.

  The system releases all resources normally used by the job from the hypervisor host, then migrates the job to the destination host without any detectable delay. During this time, the job remains in a RUN state.

  RHEL KVM hypervisors do not support live migration with VM job checkpointing. Do not use `-dc_chkpntvm` with `-dc_vmaction livemigvm`.
- **-dc_vmaction requeuejob**: Kill the VM job and resubmit it to the queue.

  The system kills the VM job and submits a new VM job request to the queue.

The default value for **-dc_vmaction** is undefined (no action taken) if you modify a job that was previously not a Dynamic Cluster and specify **-dc_tmpl** without specifying a value for **-dc_vmaction**.

The action defines the behavior only when this job is the lower priority (preempted) job, not the higher priority (preempting) job.

When you define the preemption action on the command line, DC_VM_PREEMPTION_ACTION in `lsb.applications` is ignored.

When using `livemigvm` to specify a live migration preemption action, you can also specify a second preemption action to trigger if the live migration action fails. Use a space to separate the two actions and quotation marks to enclose the two actions:

`-dc_vmaction "livemigvm[`*details*`] `*second_action*`"`

For example,

```
bmod -dc_vmaction "livemigvm[wait_trigger_time=100:livemig_max_downtime=0:livemig_max_exectime=1000] requeuejob" \
myjob
```

If the live migration fails (because the trigger time or execute time is exceeded), the **requeuejob** action triggers.

bsub

Submits a job to LSF by running the specified command and its arguments.

Synopsis

bsub [*options*] *command* [*arguments*]

bsub -h[**elp**] [**all**] [**description**] [*category_name ...*] [**-***option_name ...*]

Categories and options

Use the keyword `all` to display all options and the keyword `description` to display a detailed description of the **bsub** command. For more details on specific categories and options, specify **bsub -h** with the name of the categories and options.

Categories

Category: dc:

Dynamic Cluster only. Submit Dynamic Cluster jobs: `-dc_chkpntvm`, `-dc_mtype`, `-dc_tmpl`, `-dc_vmaction`.

Options

-dc_chkpntvm:

Dynamic Cluster only. Enable VM job checkpointing by specifying an initial checkpoint time and recurring checkpoint interval.

Categories

dc

Synopsis

bsub -dc_chkpntvm "init=*initial_minutes interval_minutes***"**

Conflicting options

Do not use with the `-k` option.

Description

Dynamic Cluster creates the first VM job checkpoint in the specified number of minutes (*initial_minutes*) after the job dispatched. Dynamic Cluster creates subsequent checkpoints in the specified number of minutes (*interval_minutes*) after the previous checkpoint.

The job must be rerunnable, either by using the **bsub -r** option, or by submitting the job to a rerunnable queue.

-dc_livemigvm:

Dynamic Cluster only. Specifies whether the job can be live migrated when its hypervisor host is selected for host memory defragmentation.

Categories

dc

Synopsis

bsub -dc_livemigvm y | n

Conflicting options

This option is ignored if `-dc_tmpl` and `-dc_mtype vm` are not also used or if you used `-dc_tmpl none`.

Description

Specify **`-dc_livemigvm y`** and your job is live migrated if its hypervisor host is selected for host memory defragmentation. This option is the default behavior.

Specify **`-dc_livemigvm n`** to indicate that your job cannot be live migrated.

-dc_mtype:

Dynamic Cluster only. Specifies the machine type for the job.

Categories

dc

Synopsis

bsub -dc_mtype vm | pm | any

Conflicting options

This option is ignored if `-dc_tmpl` is not also used or if you used `-dc_tmpl none`.

Description

Specify **`-dc_mtype vm`** if you want the Dynamic Cluster job to be a VM job.

Specify **`-dc_mtype pm`** if you want the job to run on physical machines.

By default, Dynamic Cluster jobs run on any machines. The syntax **`-dc_mtype any`** is supported to specify this explicitly (physical machines are selected before virtual machines).

When you define Dynamic Cluster templates on the command line, DC_MACHINE_TYPE in `lsb.applications` is ignored.

-dc_tmpl:

Dynamic Cluster only. Specifies the Dynamic Cluster templates that the job can use.

Categories

dc

Synopsis

bsub -dc_tmpl none | **any** | "*DC_template_name* ..."

Description

Specify the name of one or more Dynamic Cluster templates that the job can use, or use the any or none keywords. Separate names in the list with space. Using this option makes the job use Dynamic Cluster provisioning.

Specify `-dc_tmpl none` to use none of the templates and the job also ignores the `-dc_mtype` and `-dc_vmaction` options. This cancels the default Dynamic Cluster template configured by the default application profile (DC_MACHINE_TEMPLATES in `lsb.applications`) or **esub** to allow the end user to the choice of using only non-Dynamic Cluster hosts. (with **-dc_tmpl**)

When you define Dynamic Cluster templates on the command line, DC_MACHINE_TEMPLATES in `lsb.applications` is ignored.

-dc_vmaction:

Dynamic Cluster only. Specifies the VM behavior if this job is preempted.

Categories

dc

Synopsis

bsub -dc_vmaction savevm | **requeuejob** | **livemigvm**

Conflicting options

This option is ignored if `-dc_tmpl` and `-dc_mtype vm` are not also used or if you used `-dc_tmpl none`.

Description

Specify **-dc_vmaction** *action* to specify an action on the VM if this job is preempted.

The following are a list of preemption actions that you can specify with this option:

- **-dc_vmaction savevm**: Save the VM.

  Saving the VM allows this job to continue later on. This option defines the action that the lower priority (preempted) job should take upon preemption, not the one the higher priority (preempting) job should initiate.

- **-dc_vmaction requeuejob**: Kill the VM job and resubmit it to the queue.

  The system kills the VM job and submits a new VM job request to the queue.
- **-dc_vmaction livemigvm**: Live migrate the VM (and the job running on it) from one hypervisor host to another.

  The system migrates the job to the destination host, then releases all resources normally used by the job from the hypervisor host. During this time, the job remains in a RUN state.

  RHEL KVM hypervisors do not support live migration with VM job checkpointing. Do not use `-dc_chkpntvm` with `-dc_vmaction livemigvm`.

The action defines the behavior only when this job is the lower priority (preempted) job, not the higher priority (preempting) job.

When you define the preemption action on the command line, DC_VM_PREEMPTION_ACTION in `lsb.applications` is ignored.

When using `livemigvm` to specify a live migration preemption action, you can also specify a second preemption action to trigger if the live migration action fails. Use a space to separate the two actions and quotation marks to enclose the two actions:

`-dc_vmaction "livemigvm[`*details*`] `*second_action*`"`

Examples

```
bsub -dc_tmpl rhel58_vm -dc_mtype vm -dc_vmaction \
"livemigvm[wait_trigger_time=100:livemig_max_downtime=0:livemig_max_exectime=1000] requeuejob" \
myjob
```

If the live migration fails (because the trigger time or execute time is exceeded), the **requeuejob** action triggers.

lsadmin

The **limstartup**, **limshutdown**, **limrestart**, **resstartup**, **resshutdown**, and **resrestart** subcommands all skip Dynamic Cluster hosts with the `dchost` and `jobvm` resource. To run these subcommands on these hosts, you must specify these hosts manually.

lshosts

Shows Dynamic Cluster information.

-a

Dynamic Cluster only. Shows information about all hosts, including Dynamic Cluster virtual machine hosts configured with the jobvm resource.

Default output includes only standard LSF hosts and Dynamic Cluster hosts configured with the dchost resource.

lsload

Shows Dynamic Cluster information.

-a

Dynamic Cluster only. Show information about all hosts, including Dynamic Cluster virtual machine hosts configured with the jobvm resource.

Default output includes only standard LSF hosts and Dynamic Cluster hosts configured with the dchost resource.

Parameters

Some parameters are new or have added functionality in Dynamic Cluster.

dc_conf.*cluster_name*.xml parameters

This is a new file for Dynamic Cluster configuration parameters.

ResourceGroupConf section

Example:

```
<ResourceGroupConf>
   <HypervisorResGrps>
      <ResourceGroup>
         <Name>KVMRedHat_Hosts</Name>
          <Template>rhel55hv</Template>
          <MembersAreAlsoPhysicalHosts>Yes</MembersAreAlsoPhysicalHosts>
       </ResourceGroup>
    </HypervisorResGrps>
</ResourceGroupConf>
```

The following parameters are configured in the `<ResourceGroupConf>` section of the file.

MembersAreAlsoPhysicalHosts:
Syntax

`Yes`

Description

Defines a host as a hypervisor which can also run physical machine workload on idle resources.

Default

Not defined.

Name:
Syntax

KVMRedHat_Hosts

Description

The name of the resource group to which this hypervisors of the given template belong. The only valid value is KVMRedHat_Hosts.

Default

Not defined.

Template:
Syntax

template_name

Description

The name of the template associated with the hypervisor resource group. This must match a template name defined in the Templates section.

Default

Not defined.

Parameters section

Most parameters are configured as shown:

```
<ParametersConf>
   <Parameter name="PARAMETER_1">
      <Value>value_1</Value>
      </Parameter>
   <Parameter name="PARAMETER_2">
      <Value>value_2</Value>
      </Parameter>
</ParametersConf>
```

The following parameters are configured in the <ParametersConf> section of the file.

DC_CLEAN_PERIOD:
Syntax

seconds

Description

Time to keep provision requests in memory for query purposes after they are completed. Before this period is completed, the requests appear in the output of the **bdc action** command. After this period expires, they only appear in the output of **bdc hist**. Specify the value in seconds.

Default

1800

DC_CONNECT_STRING:
Syntax

admin_user_name::*directory_path*

Description

A connection string to specify the Platform Cluster Manager Advanced Edition administrator account name and installation directory

Default

`Admin::/opt/platform`

The Platform Cluster Manager Advanced Edition administrator is `Admin` and the installation directory is `/opt/platform`.

DC_EVENT_FILES_MAX_NUM:

Syntax

integer

Description

Maximum number of Dynamic Cluster event log files (`dc.events`) to keep.

Dependency

DC_EVENT_FILES_MIN_SWITCH_PERIOD

Default

10

DC_EVENT_FILES_MIN_SWITCH_PERIOD:
Syntax

seconds

Description

The minimum elapsed time period before archiving the Dynamic Cluster event log. Define the value in seconds.

Works together with DC_PROVISION_REQUESTS_MAX_FINISHED to control how frequently Dynamic Cluster archives the file `dc.events`. The number of finished requests in the file is evaluated regularly, at the interval defined by this parameter. The file is archived if the number of requests has reached the threshold defined by DC_PROVISION_REQUESTS_MAX_FINISHED.

The event log file names are switched when a new file is archived. The new file is named `dc.events`, the archived file is named `dc.events.1`, and the previous `dc.events.1` is renamed `dc.events.2`, and so on.

Dependency

DC_PROVISION_REQUESTS_MAX_FINISHED

Default

1800

DC_LIVEMIG_MAX_DOWN_TIME:
Syntax

seconds

Description

For KVM hypervisors only. The maximum amount of time that a VM can be down during a live migration. This is the amount of time from when the VM is stopped on the source hypervisor and started on the target hypervisor. If the live migration cannot guarantee this down time, the system will continue to retry the live migration until it can guarantee this maximum down time (or the

DC_LIVEMIG_MAX_EXEC_TIME parameter value is reached). Specify the value in seconds, or specify 0 to use the hypervisor default for the down time.

Default

0

DC_LIVEMIG_MAX_EXEC_TIME:
Syntax

seconds

Description

For KVM hypervisors only. The maximum amount of time that the system can attempt a live migration. If the live migration cannot guarantee the down time (as specified by the **DC_LIVEMIG_MAX_DOWN_TIME** parameter) within this amount of time, the live migration fails. Specify the value in seconds from 1 to 2147483646.

Default

2147483646

DC_MACHINE_MAX_LIFETIME:
Syntax

minutes

Description

Limits the lifetime of a dynamically created virtual machine. After the specified time period has elapsed since a VM's creation, if the VM becomes idle, the system automatically powers off and destroy the VM.

This parameter is useful when propagating updates to a shared template. If a shared template is updated, all VM instances which were generated from this template will still run with the old template even if they were powered off.

Setting this parameter to a finite value will cause VMs to be uninstalled (deleted from disk) after the specified period and completing running workload. Any further requests for the shared template must install a new VM, which will then be based on the new version of the template. Therefore administrators can be sure that the template update has been propagated throughout the system after the specified time period.

Default

Not defined. Infinite time.

DC_MACHINE_MIN_TTL:
Syntax

minutes

Description

Minimum time to live of the Dynamic Cluster host or virtual machine before it can be reprovisioned. This parameter is used to prevent system resources from being reprovisioned too often and generating unnecessary load on the infrastructure. For example, if the value is set to 60, any freshly provisioned machine will not be reprovisioned in less than 60 minutes.

Default

0

DC_PROVISION_REQUESTS_MAX_FINISHED:
Syntax

integer

Description

Number of finished provisioning requests in the Dynamic Cluster event log before it is archived. Works together with DC_EVENT_FILES_MIN_SWITCH_PERIOD to control how frequently Dynamic Cluster archives the file `dc.events`. The number of jobs in the file is evaluated regularly, at the interval defined by DC_EVENT_FILES_MIN_SWITCH_PERIOD. The file is archived if the number of jobs has reached or exceeded the threshold defined by DC_PROVISION_REQUESTS_MAX_FINISHED.

The event log file names are switched when a new file is archived. The new file is named `dc.events`, the archived file is named `dc.events.1`, the previous `dc.events.1` is renamed `dc.events.2`, and so on.

Dependency

DC_EVENT_FILES_MIN_SWITCH_PERIOD

Default

5000

DC_REPROVISION_GRACE_PERIOD:
Syntax

seconds

Description

After each job finishes, allow a grace period before the machine can accept another provision request. Specify the value in seconds.

By default, when a job completes, the machine it was running on becomes eligible for reprovisioning. However, some jobs have post-execution processes that may be interrupted if the host is reprovisioned too quickly. This parameter configures a grace period after job termination during which the host cannot be reprovisioned, which gives these processes a chance to complete.

To ensure that the machine is not reprovisioned until post-execution processing is done, specify `JOB_INCLUDE_POSTPROC=Y` in `lsb.params`.

Default

0

DC_VM_UNINSTALL_PERIOD:
Syntax

minutes

Description

Time period to uninstall (delete from storage) a dynamically created VM in the off state. Specify the value in minutes.

A virtual machine can be created to meet peak workload demands. However, after peak loads pass, those virtual machines will be powered off and stored in storage. Those dynamic virtual machines can be configured to be deleted if they remain off for a long time.

Note: VMs in the OFF status still hold an IP address reservation. To release this IP address, the VM must be uninstalled (deleted from disk). To have VMs release their IP reservation immediately when powered down, specify 0 as the value of this parameter to uninstall them immediately."

Default

1440

DC_VM_MEMSIZE_DEFINED:
Syntax

integer

Multiple values allowed.

This parameter wraps each value in <memsize> instead of <Value>. For example:

```
<memsize> integer </memsize>
```

Description

Specify one or more choices for VM memory size. Specify the value in MB. Separate values in a list with space.

The memory size of any new VM is the smallest of all the choices that satisfy the job's resource requirement.

For example, if a job requires 500 MB memory, and this parameter is set to "1024 1536", the VM is created with 1024 MB memory. If a job requires 1025 MB memory, a VM is created with 1536 MB memory.

Using this feature helps prevent the hypervisor hosts from being fragmented with multiple VMs of different size. When the VMs have standardized memory size,

they can easily be reused for jobs with similar memory requirements.

Dependency

Dynamic Cluster only.

If this parameter is used, DC_VM_MEMSIZE_STEP in `lsb.params` is ignored.

Valid Values

512 minimum

Default

Not defined

DC_VM_PREFIX:
Syntax

string

Description

Prefix for naming a new VM that is created by Dynamic Cluster. Specify a text string.

You can specify more than one name using multiple <Value/> entries, but only the first value is used for dynamically creating new VMs. However, all VMs named with any of these prefixes will be treated as dynamically created VMs even if they were manually created. They will be subject to DC_VM_UNINSTALL_PERIOD, DC_MACHINE_MAX_LIFETIME.

Default

vm_lsf_dyn

DC_VM_RESOURCE_GROUPS:
Syntax

resource_group_name

Description

Specify names of Dynamic Cluster resource groups which are allowed to create new VMs. For KVM hypervisors, the only valid value is KVMRedHat_Host.

DC_WORKDIR:
Syntax

directory_path

Description

Dynamic Cluster working directory. This is the location where the `dc.events` file will be stored.

Default

`/opt/lsf/work/cluster_name/dc`

Templates section

For more information, see the setup instructions.

lsb.applications parameters

Note:

If you enable resizable jobs for an application profile (by specifying RESIZABLE_JOBS=Y or RESIZABLE_JOBS=auto), Dynamic Cluster parameters are ignored in that application profile. That is, any jobs submitted to an application profile configured for resizable jobs cannot use Dynamic Cluster options.

DC_CHKPNTVM

Syntax

`DC_CHKPNTVM=`*initial_minutes interval_minutes*

Description

Specify a custom VM job checkpoint time. Dynamic Cluster creates the first VM job checkpoint in the specified number of minutes (*initial_minutes*) after the job dispatched. Dynamic Cluster creates subsequent checkpoints in the specified number of minutes (*interval_minutes*) after the previous checkpoint.

Defining the Dynamic Cluster template in the application profile or command line makes the LSF job use Dynamic Cluster provisioning.

Dependency

Dynamic Cluster only.

This parameter is equivalent to the **bsub** option **-dc_chkpntvm**.

The job must be rerunnable, either by using the **bsub -r** option, or by submitting the job to a rerunnable queue.

VM job checkpointing cannot be used together with LSF checkpointing (**bsub -k**).

RHEL KVM hypervisors do not support live migration with VM job checkpointing. Do not use **DC_CHKPNTVM** with the `-dc_vmaction livemigvm` preemption action.

If VM job checkpoint times are defined on the command line, this parameter is ignored.

Default

Not defined.

DC_MACHINE_TEMPLATES

Syntax

`DC_MACHINE_TEMPLATES=`*template_name...*

Description

Specify the names of Dynamic Cluster templates. Separate names in the list with space.

Defining the Dynamic Cluster template in the application profile or command line makes the LSF job use Dynamic Cluster provisioning.

Dependency

Dynamic Cluster only.

If Dynamic Cluster templates are defined on the command line, all Dynamic Cluster parameters in `lsb.applications` are ignored.

Default

Not defined.

DC_MACHINE_TYPE
Syntax

```
DC_MACHINE_TYPE= VM | PM | any
```

Description

Specify VM to run the Dynamic Cluster job on a virtual machine. Specify PM for a physical machine.

Dependency

Dynamic Cluster only.

This parameter is ignored if DC_MACHINE_TEMPLATES is not also defined in `lsb.applications`.

If Dynamic Cluster templates are defined on the command line, all Dynamic Cluster parameters in `lsb.applications` are ignored.

Default

`any`

DC_JOBVM_PREEMPTION_ACTION
Syntax

```
DC_JOBVM_PREEMPTION_ACTION=savevm
```

Description

Specify savevm to save the VM when jobs from this application profile are preempted. This allows the job to continue later on.

The action defines the behavior only when jobs from this queue are the lower priority (preempted) job, not the higher priority (preempting) job.

By default, low priority job running in the VM will not be considered as preemptable and will keep running until it completes.

Dependency

This parameter is equivalent to the bsub option **-dc_vmaction**.

If Dynamic Cluster machine templates are defined on the command line, this parameter is ignored.

Default

Not defined.

lsb.modules parameters

schmod_dc.so plugin

lsb.params parameters

DC_HOST_DEFRAG_MAX_CONCURRENT_NUM_JOBS

Syntax

```
DC_HOST_DEFRAG_MAX_CONCURRENT_NUMB_JOBS=integer
```

Description

Specify the maximum concurrent number of Dynamic Cluster jobs that can trigger a host memory defragmentation.

This parameter allows you to manage any performance impact on regular scheduler, and controls the load on network and storage infrastructure (that is, to prevent "I/O storms").

Default

Undefined. There is no limit to the number of jobs that can trigger a host memory defragmentation.

DC_JOB_DISPATCH_RETRY_NUM

Syntax

```
DC_JOB_DISPATCH_RETRY_NUM=integer
```

Description

Number of retries for job dispatch to the Dynamic Cluster machine. If job dispatch fails during a scheduling cycle, LSF can retry the specified number of times, then the job returns to the pending state to wait for the next LSF scheduling cycle.

For example, if the number of retries is 5, LSF may make 6 attempts to dispatch the job.

Job dispatch can fail if the Dynamic Cluster host is newly provisioned but **sbatchd** has not started yet.

Default

5

DC_TIMEOUT_WAITING_SBD_START

Syntax

```
DC_TIMEOUT_WAITING_SBD_START=time_in_seconds
```

Description

Specify how long **mbatchd** waits for **sbatchd** on the VM to start and connect. After this time, Dynamic Cluster considers the guest OS to be corrupted. Dynamic Cluster forcefully powers off the VM and uninstalls the corrupted guest OS. The corresponding VM job becomes pending until a VM is available.

Default

1800 seconds (30 minutes).

DC_VM_MEMSIZE_STEP

Syntax

```
DC_VM_MEMSIZE_STEP=megabytes
```

Description

Each VM is created with memory size that is a multiple of this number. Specify the value in MB. The memory size of the VM is rounded up from the value required by the job.

For example, if a job requires 500 MB memory, and this parameter is set to 512, the VM is created with 512 MB memory. If a job requires 600 MB memory, a VM is created with 1024 MB memory.

Using this feature helps prevent the hypervisor hosts from being fragmented with multiple VMs of different size. When the VMs have standardized memory size, they can easily be reused for jobs with similar memory requirements.

The minimum memory size for a new VM is DC_VMJOB_MEM_REQ. If DC_VMJOB_MEM_REQ is not set, the minimum memory size for a new VM is hard-coded to 1024MB, even if the step size is smaller.

Dependency

Dynamic Cluster only.

This parameter is ignored if DC_VM_MEMSIZE_DEFINED is defined in `dc_conf.cluster_name.xml`.

Default

1024

DC_VMJOB_MEM_REQ

Syntax

DC_VMJOB_MEM_REQ=*megabytes*

Description

Default memory requirement for a VM job. Specify the value in MB. This default is used for VM jobs that do not have any memory requirement defined at the job, queue, or application level.

A VM job must have a memory requirement because it affects the minimum memory size of the VM.

Dependency

Dynamic Cluster only.

This parameter works with DC_VM_MEMSIZE_DEFINED in dc_conf.*cluster_name*.xml or DC_VM_MEMSIZE_STEP in lsb.params. For example, if this parameter defines the memory requirement as 2000 MB, and the nearest allowed VM memory size is 2048, then the VM created for the job has 2048 MB memory.

Valid Values

512 minimum.

Default

1024

lsb.queues parameters

DC_HOST_DEFRAG_MIN_MEMSIZE

Syntax

DC_HOST_DEFRAG_MIN_MEMSIZE=*size_in_GB*

Description

Specify that a job triggers a host memory defragmentation if its rusage mem requirement (that is, its memory resource requirement) is larger than or equal to this value.

This parameter specifies the minimum memory requested before a job is considered a "large memory job" that can trigger a host memory defragmentation.

This parameter is only valid in queues that specify **DC_HOST_DEFRAG_TIMEOUT**.

Default

0. Any jobs with a memory resource requirement can trigger a host memory defragmentation.

DC_HOST_DEFRAG_MIN_PENDING_TIME

Syntax

```
DC_HOST_DEFRAG_MIN_PENDING_TIME=time_in_minutes
```

Description

Specify how long a job is pending, in minutes, before it triggers a host memory defragmentation.

If a job cannot find an available host with enough memory, the job pends. This parameter specifies how long the job waits for a host to become available before triggering a host memory defragmentation to make a host become available.

This parameter is only valid in queues that specify **DC_HOST_DEFRAG_TIMEOUT**.

Default

0. The job triggers a host memory defragmentation immediately if it cannot find an available host.

DC_HOST_DEFRAG_TIMEOUT

Syntax

```
DC_HOST_DEFRAG_TIMEOUT=time_in_minutes
```

Description

Specify how long a host memory defragmentation reservation lasts, in minutes, until the reservation times out.

This parameter enables host memory defragementation for the queue.

Default

Undefined. Host memory defragmentation is disabled.

DC_IGNORE_MACHINE_TTL

Syntax

```
DC_IGNORE_MACHINE_TTL= Y | N
```

Description

Specify whether the DC_MACHINE_MIN_TTL parameter should be ignored for jobs submitted to this queue. By default, if the DC_MACHINE_MIN_TTL has not elapsed, an unmatched pending job cannot trigger the reprovisioning of the given machine. With this parameter turned on, the job in the queue will ignore this limitation and trigger reprovisioning.

This parameter is especially useful for workloads in high priority queue which require immediate resources.

Default

N

JOB_ORDER_BY_MEM
Syntax

```
JOB_ORDER_BY_MEM=Y
```

Description

Enable a job scheduling policy that gives preference to high-memory jobs.

If set, jobs in a queue are ordered for dispatch according to the following:

- User priority (defined by user-level fairshare)
- Job priority (defined by **-sp**)
- Job memory (priority to jobs needing the most memory)
- Job submission order (priority to jobs submitted earliest)

By default, the job memory is not considered at all.

Dependency

This parameter is *NOT* restricted to Dynamic Cluster.

Default

Not enabled.

lsf.cluster file

Hosts section

RESOURCES:
Two reserved resources for Dynamic Cluster:

- dchost - identifies the LSF server host managed by Dynamic Cluster. These hosts can be provisioned as physical machines or virtual machines, depending on the demand. These hosts require LSF and are part of LSF job scheduling.
- jobvm - identifies a job container created by Dynamic Cluster. These are virtual machines that run on the LSF server hosts managed by Dynamic Cluster.

External job submission (esub)

The temporary file (pointed to by **LSB_SUB_PARM_FILE**) that LSF uses to store **bsub** options entered in the command line stores the following information specific to Dynamic Cluster job submission:

Option	bsub or bmod option	data type	Description
LSB_SUB4_DC_TEMPLATE	-dc_tmpl	string	Dynamic Cluster only. String that contains the name of the Dynamic Cluster templates that the job can use. The any keyword allows the use of any Dynamic Cluster template, while the none keyword uses none of the Dynamic Cluster templates (while ignoring any other Dynamic Cluster options).
LSB_SUB4_DC_NOT_MIGRATABLE	-dc_livemigvm n	boolean	Dynamic Cluster only. "Y" specifies that Dynamic Cluster jobs cannot be live migrated when its hypervisor is selected for host memory defragmentation.
LSB_SUB4_DC_MIGRATABLE	-dc_livemigvm y	boolean	Dynamic Cluster only. "Y" specifies that Dynamic Cluster jobs can be live migrated when its hypervisor is selected for host memory defragmentation.
LSB_SUB4_DC_MTYPE_ANY	-dc_mtype any	boolean	Dynamic Cluster only. "Y" specifies that Dynamic Cluster jobs run on any machines.
LSB_SUB4_DC_MTYPE_PM	-dc_mtype pm	boolean	Dynamic Cluster only. "Y" specifies that Dynamic Cluster jobs run on physical machines.
LSB_SUB4_DC_MTYPE_VM	-dc_mtype vm	boolean	Dynamic Cluster only. "Y" specifies that Dynamic Cluster jobs run on virtual machines.

Option	bsub or bmod option	data type	Description
LSB_SUB4_DC_PREEMPTACTION_VM	`-dc_vmaction savevm | livemigvm | requeuejob`	string	Dynamic Cluster only. Specifies the action taken on the lower priority job if the Dynamic Cluster job is preempted when running on a virtual machine: • "savevm" specifies that the lower priority job is saved so that it can continue later on. • "livemigvm" specifies that the lower priority job is live migrated to another hypervisor. • "requeuejob" specifies that the lower priority job is killed and resubmitted to the queue.

Notices

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

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

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte character set (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

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

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web

sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

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

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
Intellectual Property Law
Mail Station P300
2455 South Road,
Poughkeepsie, NY 12601-5400
USA

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurement may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

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.

All 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 the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrates 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.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

© (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp. _enter the year or years_.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Trademarks

IBM, the IBM logo, and ibm.com® are trademarks of International Business Machines Corp., 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.

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

Java™ and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

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

LSF®, Platform, and Platform Computing are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide.

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

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

Privacy policy considerations

IBM Software products, including software as a service solutions, (“Software Offerings”) may use cookies or other technologies to collect product usage information, to help improve the end user experience, to tailor interactions with the end user or for other purposes. In many cases no personally identifiable information is collected by the Software Offerings. Some of our Software Offerings can help enable you to collect personally identifiable information. If this Software

Offering uses cookies to collect personally identifiable information, specific information about this offering's use of cookies is set forth below.

This Software Offering does not use cookies or other technologies to collect personally identifiable information.

If the configurations deployed for this Software Offering provide you as customer the ability to collect personally identifiable information from end users via cookies and other technologies, you should seek your own legal advice about any laws applicable to such data collection, including any requirements for notice and consent.

For more information about the use of various technologies, including cookies, for these purposes, See IBM's Privacy Policy at http://www.ibm.com/privacy and IBM's Online Privacy Statement at http://www.ibm.com/privacy/details the section entitled "Cookies, Web Beacons and Other Technologies" and the "IBM Software Products and Software-as-a-Service Privacy Statement" at http://www.ibm.com/software/info/product-privacy.

Printed in USA

SC27-5320-02