XCAT 2 Rolling Update Support

10/13/10, PM 03:07:55

1.0 Overview

This document provides an overview of the xCAT Rolling Update support available in xCAT 2.6 and later releases. This support is available in xCAT 2.5 as beta.

The xCAT rolling update support is currently restricted to AIX diskless nodes. Linux support and diskfull support has not been implemented or verified yet.

The xCAT rolling update support allows you to update the OS image on a subset of cluster nodes at a time, such that the remaining nodes in the cluster can still be running jobs. This process uses the Tivoli Workload Scheduler LoadLeveler to determine node availability and control the update process.
Based on input from the xCAT administrator, LoadLeveler flexible reservations are created for specific sets of nodes, and xCAT will update a set of nodes when its reservation becomes active.

xCAT rollupdate support provides the framework to update sets of nodes in your cluster. Nodes to be updated can be specified in one of two ways:

1. As a list of all nodes to be updated, and the number of nodes to be updated together in one set. This approach can be used for simple updates across your cluster, where there are no special interdependencies between node updates and nodes can be updated in any order as they become available.

2. As specific update groups, in which the nodes to be updated together are explicitly listed. This approach is for more complex updates in which you wish to have more control over which nodes are updated together, the order of node updates within a group, and which groups of nodes cannot be updated at the same time in order to maintain critical system resources.

Both approaches allow you to limit the number of nodes to be updated at any one time, to provide a list of commands that can be run before the nodes in an update group are shut down (prescripts) and a list of commands that can be run after the nodes have been powered off, but before they are rebooted (outofbandcmds).

Prescripts can be useful to notify applications that nodes will be going down, and to move critical services to backup nodes before they are shut down.

Out-of-band commands can be used to perform operations that require the nodes to be powered down, such as firmware updates.

2.0 Prerequisites for Rolling Update Support

The following prerequisites must be set up before running the xCAT rolling update support:

- LoadLeveler must be installed and running on all of the cluster nodes that will be updated. See Setting up the IBM HPC Stack in an xCAT Cluster for information on installing and configuring LoadLeveler in an xCAT cluster. It is possible to run updates for some nodes that are not active in LoadLeveler; however to use the full rolling update support, LL should be active. See the stanza input entries for the standard update method with **update_if_down=yes**.
- LoadLeveler must be running with the database configuration option, using xCAT's database.
- The LoadLeveler central manager must either be the xCAT management node or be set up as an xCAT client machine. To set up a remote xCAT client, see the xCAT wiki how-to Granting Users xCAT privileges & Setting up a Remote Client.
- The LoadLeveler userid specified in the xCAT rollupdate input **scheduser** stanza must be authorized to run the xCAT "runrollupdate" command. See the xCAT wiki how-to Granting Users xCAT privileges & Setting up a Remote Client for instructions on setting this up.
- LoadLeveler settings need to be able handle flexibile reservations. See the chapter on LoadLeveler Settings for a listing of additional requirements.
- The root userid on the xCAT management node must have LOADL_ADMIN privileges.
- You have created the new OS image that will be loaded onto the nodes and run all of the necessary xCAT commands such that when the node is rebooted, it will load your updated image.

3.0 rollupdate Input Stanza File

The input to the xCAT rollupdate command is a stanza file piped through STDIN. xCAT provides sample stanza files in:

/opt/xcat/share/xcat/rollupdate

The rolling update support provides two different methods of specifying nodes to be update: The **updateall** method to update all specified nodes in any order as they become available, and the **standard** method requiring explicit update groups identifying nodes to be updated together. Different stanza keywords apply to these different methods and are noted below.

Stanza input is specified as:

keyword = value

with one keyword per line. Unless otherwise noted in the descriptions below, if multiple stanza lines are specified for the same keyword, only the FIRST entry will be used and all others will be ignored.

Valid keywords are:

scheduler = *scheduler*

where *scheduler* is the job scheduler used to submit the rolling update jobs. Currently only "loadleveler" is supported.

scheduser = *scheduser*

where *scheduser* is the userid with authority to submit scheduler jobs. Note that LoadLeveler does not allow reservation jobs to be submitted by the root userid.

oldfeature = *feature value*

(optional) where *feature value* is an existing LoadLeveler feature value that is set for the nodes being updated. xCAT will remove this value from the LoadLeveler machine definition after the update has completed.

newfeature = *feature value*

(optional) where *feature value* is a new LoadLeveler feature value to be set for the nodes being updated. xCAT will add this value to the LoadLeveler machine definition after the update has completed. This can be useful for users that wish to schedule jobs that can only be run on nodes that have been updated.

updateall = yes | no

Specifies whether this rolling update request is for the updateall method of specifying nodes. Default is **no**.

This method should be used for simple compute node updates that have no special dependencies on other nodes and the update order is not important. Only those nodes that are currently active in the scheduler will be updated.

If **updateall=yes**, the following stanza entries MUST also be specified:

updateall_nodes

updateall_numperupdate

job_template
job_dir

updateall_nodes = *xCAT nodereange*

Used with **updateall=yes**. The *xCAT nodereange* specifies the list of nodes that are to be included in this rolling update request (see the xCAT **noderange** man page).. All nodes must be active in the job scheduler in order to be updated.

updateall_numberupdate = *numeric value*

Used with updateall=yes. The *numeric value* specifies the number of nodes that will be reserved at one time in the scheduler and updated together. The smaller the number, the more scheduler reservation jobs that will be submitted.
NOTE: LoadLeveler performance decreases with large numbers of reservations. Do not set this value so low that you will exceed the maximum number of reservations allowed for your cluster or that you will degrade LL performance for your production jobs. You must also ensure that the LL MAX_RESERVATIONS setting is large enough to handle all the reservations that will be created.

updategroup = *name(noderange)*

For standard updates, at least one **updategroup** or **mapgroups** stanza must be specified. The *name* specifies the name to be assigned to this update group. The *noderange* is an xCAT noderange that specifies the list of nodes that will be included in this update group (see the xCAT **noderange** man page).
Multiple **updategroup** stanzas may be supplied, one for each group of nodes to be updated.

mapgroups = *nodegroup range*

For standard updates, at least one **updategroup** or **mapgroups** stanza must be specified. . The *nodegroup range* specifies a list or range of xCAT nodegroup names. This field is processed in the same way the xCAT noderange processing works for node names, except that it will generate a list of xCAT nodegroup names. Each nodegroup in the list will become its own update group for this rolling update request, with the update group name set to the nodegroup name.
Multiple **mapgroups** stanzas may be supplied.
For example, the following will create 10 updategroups from the 10 nodegroups named block01 to block10:

```
mapgroups=block[01-10]
```

mutex = *updategroup,updategroup,...*

(optional) For standard updates. The comma-delimited list of *updategroup* names specify which update groups are mutually

exclusive and must not be updated at the same time in order to maintain active resources within the cluster. Only one updategroup listed in the entry will be updated at a time.
You may list multiple **mutex** stanzas to identify different sets of mutual exclusion.
For example, the following states that the update processes for ns1 and for ns2 will not be allowed to run at the same time:
mutex=ns1,ns2

mutex = *updategroup range,updategroup range,...*
(optional) For standard updates. The comma-delimited list of *updategroup ranges* will be processed in the same way the xCAT noderange processing works for node names, except that it will generate a list of rolling update update group names. The first name in each range is paired together to make a mutual exclusion list, the second name in each range is paired together, etc.
For example, the following single entry:
mutex=block[1-3]a,block[1-3]b,block[1-3]c
would be equivalent to specifying these three entries:
mutex=block1a,block1b,block1c
mutex=block2a,block2b,block2c
mutex=block3a,block3b,block3c

maxupdates = *numeric value* | **all**
where *numeric value* is the maximum number of update groups that can be updated at one time (i.e. the maximum number of LoadLeveler rolling update reservations that can be active). This allows you to ensure you will always have enough computing resources in your cluster and that not all nodes will attempt to be updated at once.
A value of **all** specifies that there is no restriction.

jobtemplate = *filename*
where *filename* is a filename with full directory path that identifies the scheduler job command file template to be used to submit reservations. See sample LoadLeveler templates in:
/opt/xcat/share/xcat/rollupdate/*.tmpl
The following substitution values will be replaced by the xCAT rollupdate command to generate a unique job command file for each update group:

[[NODESET]] - the update group name for this reservation
[[JOBDIR]] - the directory specified in the rollupdate input **jobdir** stanza
[[LLHOSTFILE]] – (standard method) the file generated by the xCAT rollupdate command that contains the list of LL machines in

this update group that were available at the time the command was run.
[[MUTEXRESOURCES]] - The list of LL resources created by xCAT to handle mutual exclusion and maxupdates
[[LLCOUNT]] - (required for updateall method) used by xCAT to set the number of machines to reserve
[[UPDATEALLFEATURE]] – (required by updateall method) used by xCAT to control the rolling update

jobdir = *directory*
where *directory* is the directory to write the generated LoadLeveler job command files and other xCAT rolling update data files to. For LL, this directory needs to be on a filesystem available to all nodes.

reservationcallback = /opt/xcat/bin/rollupdate
INTERNAL KEYWORD used for development only. This is the reservation notify or callback command. For Loadleveler, this script must reside on the LoadLeveler central manager and will be called when the reservation for an updategroup becomes active.

reservationduration = *time*
where *time* is the maximum time to hold a LoadLeveler reservation for the update process. This value in minutes should be longer than the expected time to shutdown, update, and reboot all the nodes in an update group. xCAT will release the nodes from the reservation as they come back up, and will cancel the reservation when the last node has completed.

update_if_down = yes | no | cancel
Specifies whether nodes that are not active in the job scheduler should be updated.
For the rollupdate updateall method, only nodes with active startd daemons can be updated.
- If update_if_down is set to **cancel**, and specified nodes are not active in LoadLeveler, this update will be cancelled.
- If update_if_down is set to **no**, any specified nodes that are not active will be skipped (the assumption is that those nodes will load the new image the next time they are brought back online).
- A value of **yes** is not supported for the updateall method.

For the standard rollupdate method, only reservations for active LL nodes can be created.
- If update_if_down is set to **cancel**, and specified nodes are not active in LoadLeveler, this update group will not be processed.

- If update_if_down is set to **no**, any specified nodes in the update_group that are not active will be skipped. The assumption is that those nodes will load the new image the next time they are brought back online.
- If update_if_down is set to **yes**, a LoadLeveler reservation will only be created for the active nodes in the group, and when that reservation becomes active, all nodes in the group (both LL and non-LL) will be processed. This is useful for nodes that are not used for running jobs.

prescript = *command string*
prescriptnodes = *noderange*

(optional) where *command* is the name of a command to be run on the xCAT management node before issuing the shutdown command for the nodes in the updategroup.
prescriptnodes is only supported with the standard rollupdate method. If it is also specified, the command will only be run for the nodes being updated from the updategroup that are also included in that xCAT *noderange*. If **prescriptnodes** is not specified (and for the updateall method), the command will be run for all the nodes in the updategroup.
For **prescript**, you may specify the string $NODELIST in the *command string* if you would like the comma-delimited list of xCAT nodenames passed into your command.
Prescripts can be used to run operations such as shutting down the global filesystem on all the nodes, or moving critical services to a backup server for specific nodes.
Multiple **prescript** entries or **prescript**/**prescriptnodes** pairs of entries may be specified. Each command will be run in order.

shutdowntimeout = *time*

(optional) where *time* is the number of minutes xCAT should wait for an OS shutdown to complete before giving up and issuing a hard power off command and continuing with the rolling update process.
Default: **shutdowntimeout=5**

outofbandcmd = *command string*
outofbandnodes = *noderange*

(optional) where *command* is the name of a command to be run on the xCAT management node after the node has been shutdown but before it is rebooted.
outofbandnodes is only supported with the standard rollupdate method. If it is also specified, the command will only be run for the nodes being updated from the updategroup that are also included in that xCAT *noderange*. If **outofbandnodes** is not specified (and for

the updateall method), the command will be run for all the nodes in the updategroup.
For **outofbandcmd**, you may specify the string $NODELIST in the *command string* if you would like the comma-delimited list of xCAT nodenames passed into your command.
Out-of-band commands can be used to run operations when nodes must be powered down such as firmware updates.
Multiple **outofbandcmd** entries or **outofbandcmd**/**outofbandnodes** pairs of entries may be specified. Each command will be run in order.

bringuporder = *noderange*
(optional for standard update method only) where the nodes being updated from the updategroup that are also included in that xCAT *noderange* will be brought up first.
If more than one node in the updategroup matches a **bringuporder** entry, they will be brought up at the same time.
Multiple **bringuporder** entries may be specified, and they will be processed in order, completing bringup of all nodes in the previous entry before starting to power on the nodes in this entry.
Any nodes in the update group that are not listed in a **bringuporder** entry will be brought up at the end.

bringupstatus = *status value*
OR
bringupappstatus = *appstatus value*
(optional) The xCAT database node *status* or node *appstatus* value that xCAT will check and will wait for to determine that the node is up. Once this status is reached, xCAT will continue bringing up more nodes (if **bringuporder** is set) and will release this node from the scheduler reservation. If both attributes are set, only **bringupappstatus** will be used.
Default: **bringupstatus=booted**

bringuptimeout = *time*
(optional) The maximum *time* in minutes xCAT should wait after issuing the rpower on command for the nodes to reach **bringupstatus** or **bringupappstatus** before giving up. If using **bringuporder** and this timeout is reached for one set of nodes, no additional nodes will be attempted to be brought up. The scheduler reservation will be cancelled.
Default: **bringuptimeout=10**

4.0 LoadLeveler Settings

LoadLeveler must be running with the database configuration option in order to use the xCAT rolling update support.

The root userid on the xCAT management node must have LOADL_ADMIN privileges.

LoadLeveler provides many settings to support and control reservations in your job scheduler. Review the LoadLeveler documentation for a full list of these controls to ensure they are set correctly to support the flexible reservation jobs that will be submitted for xCAT rolling updates.

A few key settings are listed here:

SCHEDULER_TYPE=BACKFILL
MAX_RESERVATIONS

For each machine definition (or in the default machine definition):

reservation_permitted = true

For the *scheduser* LL userid specified in your xCAT rollupdate input stanzas (or the default LL user):

max_node
max_reservation_duration
max_reservation_expiration
max_reservations
max_total_tasks
maxidle
maxjobs
maxqueued
priority
total_tasks

Settings that may be used or changed by xCAT Rolling Updates:

FLOATING_RESOURCES
SCHEDULE_BY_RESOURCES
CENTRAL_MANAGER_LIST
RESOURCE_MGR_LIST
(for machines) FEATURE

5.0 Rolling Update Process Flow

The general process flow for the xCAT Rolling Update support is:

1. Create your new OS image(s) that will be used to update the nodes. Follow the appropriate xCAT documentation for creating or updating an image. Note that you still have all of your cluster nodes actively using the current image, so you will most likely need to create a new image. For AIX diskless nodes,

you will also need to create new NIM machine definitions so that you can assign NIM resources and run the xCAT mkdsklsnode -n command without impacting the current active nodes.

2. If you are using Linux service nodes that do not have the /install directory mounted, be sure to copy the new image down to your service nodes.
3. Run all necessary xCAT commands to prepare to boot the nodes with the new OS image (nodeset, mkdsklsnode).
4. If you wish to have your LoadLeveler users be able to run jobs on either just old nodes or just updated nodes during the rolling update process, you should set a LL FEATURE value for all of the machine definitions in your cluster. You can then specify this as an **oldfeature** stanza value in your rollupdate command input. Also, be sure to specify a **newfeature** stanza value to identify updated nodes. Your users can use these feature values in their job "requirements" field.
5. Create your rollupdate command input stanza file. See the chapter on 3 rollupdate Input Stanza Filefor details.
6. (Optional) If you wish to view the progress of the rollupdate process, in a separate window, open and monitor the rollupdate log:
   ```
   tail -f /var/log/xcat/rollupdate.log
   ```
   xCAT appends to the log file with each rollupdate run, so you may wish to move or remove the log file before a new rollupdate process.
7. Initiate the rollupdate process:
   ```
   cat <your stanza file> | rollupdate --verbose
   ```
   If you want to test your input first to make sure it is what you want, and to view the LL reservation job command files and other data files generated by xCAT, run the command with the test option:
   ```
   cat <your stanza file> | rollupdate –test --verbose
   ```
   The output files will be placed in the directory you specified in the **jobdir** stanza. The verbose keyword is optional, but allows you to view detailed progress.
8. The rollupdate command will do the following:
   - Process the stanza file and create update groups based on the input provided.
   - Create LoadLeveler FLOATING_RESOURCES and set SCHEDULE_BY_RESOURCES for all mutual exclusion sets and for controlling the maximum number of updates allowed.
   - Run llstatus to query LoadLeveler for a list of nodes with active startd daemons.
   - For each update group, the following will be done:
     - Process the update_if_down stanza entry and determine if this update group can be run:
       - For the rollupdate updateall method, only nodes with active startd daemons can be updated.
         - If update_if_down is set to "cancel", and specified nodes are not active in LoadLeveler, this update will be cancelled.
         - If update_if_down is set to "no", any specified nodes that are not active will be skipped (the assumption is that those nodes

will load the new image the next time they are brought back online).
    - For the standard rollupdate method, only reservations for active LL nodes can be created.
        - If update_if_down is set to "cancel", and specified nodes are not active in LoadLeveler, this update group will not be processed.
        - If update_if_down is set to "no", any specified nodes in the update_group that are not active will be skipped. The assumption is that those nodes will load the new image the next time they are brought back online.
        - If update_if_down is set to "yes", a LoadLeveler reservation will only be created for the active nodes in the group, and when that reservation becomes active, all nodes in the group (both LL and non-LL) will be processed. This is useful for nodes that are not used for running jobs.
    - Load your LoadLeveler job command file template, and fill in all substituion values to create a unique job command file for this update group.
    - Create the xCAT rollupdate data file for this update group to pass important data to the xCAT runrollupdate process that is run when the reservation becomes active. The data file contains the full list of xCAT nodes to be processed, all prescripts and out-of-band commands, old and new LoadLeveler feature values to be changed for the node, and other information to control the update process for this update group.
    - Create a notify script for this update group for LoadLeveler to run when the reservation becomes active.
    - For the updateall method, a unique LoadLeveler feature value for this update will be set in all specified nodes to control the update process.
    - Set the xCAT appstatus "RollingUpdate=update_job_submitted" for all the nodes in the update group.
    - Run the llmkres command to submit the flexible reservation for this update group, specifying the notify script created above.
- If any update groups do NOT have any nodes that are active in LoadLeveler, and update_if_down=yes, LoadLeveler cannot be used to process those nodes. Those update groups will be handled directly by the rollupdate command following the same process below (skipping any steps related to LoadLeveler).

9. You can view the flexible reservations that xCAT submitted:
   ```
   llqres
   ```
   or
   ```
   llqres -l -R <reservation id>
   ```
   for a specific reservation.
   If any reservation is stuck in a "Waiting" status that you feel should be active, you can check the job associated with the reservation (find the job id from the llqres -l output above):

```
llq -s <job id>
```
and debug as you would for any LL reservation or job.

10. When a reservation becomes active, LoadLeveler will invoke the notify script created above. This will invoke the internal xCAT runrollupdate command for the update group.
11. The xCAT runrollupdate will proceed as follows for an update group:
    - If this is for the updateall method, query the LoadLeveler reservation to get the list of nodes that have been reserved for this update group.
    - Query the xCAT appstatus for each node to ensure it is set correctly
    - Run the prescripts for the update group
    - Shut down and power off all the nodes in the update group
    - Run the out-of-band commands for the update group
    - Reboot the nodes. If bringuporder stanzas were set in the rollupdate input, the nodes will be booted in the order specified, waiting until the bringupstatus or bringupappstatus values have been set.
    - As soon as a node is up, the LoadLeveler feature values for the node will be changed, removing the oldfeature value and the xCAT updateall feature value (if applicable), and setting the newfeature value if these were specified in the rollupdate input.
    - The reservation is changed to remove the node that is up. This node is now available for running jobs again. If this is the last node, the reservation will be cancelled.

6.0 Updating xCAT

Updating xCAT software as part of the rolling update process is not supported. The xCAT software on your management node can be updated without impacting your running cluster, and should be done manually. See the xCAT Top Doc for instructions on updating your xCAT management node.

In hierarchical clusters, the xCAT rolling update process should not be used to update the xCAT software on service nodes. See the xCAT hierarchical documents Setting Up a Hierarchical Cluster or Setting Up an AIX Hierarchical Cluster.

7.0 Rolling Updates in a Hierarchical Cluster

If you are performing a rolling update in an xCAT hierarchical cluster, there are a few special considerations that you will need to address if your service nodes will be rebooted as part of the update process. Normally, you should try to update your service nodes manually outside of the xCAT rolling update process so that you have more control over the update. However, if you are performing CEC firmware updates, and will need to power down the CEC that contains your service node, you

will need to think about the services your service node is providing to your compute nodes and how to plan your updates.

If at all possible, you should create your update groups such that a service node and all of the compute nodes it serves can be updated together at one time. When you use this approach, make sure to use the **bringuporder** stanza in the rollupdate command input to bring up your service node first so that it is running when your compute nodes start to come up.

If it is not possible to update an entire block of service node with compute nodes because you will lose critical cluster services, you will need to plan more carefully. First, you can only bring down a service node if you have some type of backup for the services it is providing your compute nodes, or if you can tolerate the temporary loss of those services. Some things to consider:

- If you have Linux statelite nodes or AIX diskless nodes and your service node is your NFS server, you can only bring this down if you have an HA NFS solution in place. Otherwise, bringing down the service node will crash your compute nodes.
- If you are doing CEC firmware updates, at a minimum your update groups need to map to all the nodes within a CEC.
- You should plan two separate rolling update runs:
  1. The first run with just your service nodes CECs. Again, you should use the **bringuporder** stanza in the rollupdate command input to bring up your service node first within an update group so that it is running when your compute nodes in that group start to come up.
  2. The second run with all the remaining CECs in your cluster. Your service nodes should all be up and running, so these compute node updates can be done in any order.
- If you have backup service nodes in your cluster, use the **mutex** stanza to set mutual exclusion to ensure your service node and backup service node do not get updated at the same time.
- If you are bringing down a node that is running a critical service, use the **prescript** stanza to run a task to move the service to a backup server that is not part of this update group. Also, use the **mutex** stanza to ensure that the node and its backup are not updated at the same time.

8.0 Updating IBM HPC Infrastructure Nodes

If your cluster is running IBM HPC software such as GPFS or LoadLeveler, you have additional cluster services that you will need determine how to update and how to keep active during an xCAT rolling update process. Whenever possible, all cluster infrastructure nodes should be manually updated before running the xCAT rolling update process for compute nodes.

The term “infrastructure nodes” will be used to refer to any nodes that are not compute nodes. These include:

- xCAT management node and backup management node, and xCAT service nodes
- GPFS configuration server and backup configuration server, IO servers. monitoring collector node, session node (e.g. for HPSS)
- LoadLeveler central manager and backup central manager, region managers, resource managers, schedulers, login nodes

It is assumed that most updates to infrastructure nodes can be applied without impacting user jobs running on compute nodes. If xCAT service nodes are stateful (full-disk install) and are also used to run other infrastructure services (such as LL region managers), updates to infrastructure software that runs on these nodes can be applied using the xCAT updatenode command without rebooting the node. For updates to other servers that are running diskless or statelite images (e.g. GPFS IO servers), these nodes can be rebooted individually to load a new OS image without impacting the rest of the cluster.

8.1 GPFS

GPFS software can be migrated to a new level one node at a time without impacting the running GPFS cluster. In order to upgrade GPFS, the GPFS daemons must be stopped and all GPFS filesystems must be unmounted on the node. For GPFS infrastructure nodes, it is important to manage the updates such that all GPFS cluster services remain operational.

When updating GPFS, the following will need to be considered:

- When updating the GPFS configuration server, the service must first be moved to the backup configuration server.
- Quorum must be maintained at all times. No more than half-minus-one quorum nodes can be down at any time.
- Updates of the GPFS I/O servers in an NSD group must be staged so that filesystems are always available.

Please consult the GPFS documentation for updating your GPFS software.

8.2 LoadLeveler

LoadLeveler infrastructure nodes (nodes running the central manager, resource manager, and region manager daemons) are required to all be running the same version of LL. For maintaining critical cluster services, these daemons should all have backup servers. Locating both primary and backup servers on the xCAT management node or service nodes, and using full-disk install service nodes (i.e. not diskless) will ensure the best cluster stability during a rolling update process. In order to upgrade LL simultaneously on these nodes while still allowing jobs to run on the cluster, the LL software should be updated before running the rolling update

process for your cluster. You should do this manually following the documented LoadLeveler procedures:

- If any primary servers are also running startd daemons, drain the startd or flush the nodes.
- If any primary servers are running schedd daemons, drain the schedd or use llmovespool to move the job queue to another schedd server, and then drain the schedd.
- Stop the primary daemons
- Make sure failover to backup daemons has occurred
- Upgrade LoadLeveler on the servers
- Restart LoadLeveler on the servers
- Make sure primary daemons take back control
- Upgrade LoadLeveler on backup servers

See the LoadLeveler documentation for more information.

8.3 CEC Updates

Updating infrastructure nodes that require a node or CEC reboot is more complicated because the nodes and cluster-wide services that depend on them must be considered in the update algorithm.

Update groups must be defined to encompass a complete CEC. When an update group is being updated, the cluster-wide services that need to be maintained during the update are:

- GPFS configuration server
- GPFS quorum
- GPFS filesystem access
- LoadLeveler services

Therefore, you should use separate **mutex** stanzas in your xCAT rollupdate command input to define the following mutual exclusion sets:

- service nodes in a building block: Only one service node CEC may be updated at a time
- GPFS I/O servers: If you have defined multiple GPFS I/O servers for each filesystem to provide continuous availability, only one GPFS I/O server CEC for each filesystem may be updated at a time
- GPFS quorum nodes: Only less than half the quorum nodes can be updated at a time.
- GPFS configuration server and backup: only one can be updated at a time.
- LoadLeveler servers and their backups: do not allow updating a server and its backup at the same time

All CECS with xCAT service nodes should be updated before updating the other CECs. This will ensure that the service node will be available when a compute node reboots and needs to load a new OS image. Therefore, two separate xCAT

rollupdates should be performed – the first one for the service node CECs only, and the second one for all the other CECs in the cluster.

Use the **prescript** and **prescriptnodes** stanzas to define tasks that should be run to move GPFS and LL services to their backup servers before shutting down those nodes during an update.

9.0 References

- xCAT documentation: http://xcat.svn.sourceforge.net/viewvc/xcat/xcat-core/trunk/xCAT-client/share/doc/index.html
- xCAT man pages: http://xcat.sf.net/man1/xcat.1.html
- xCAT DB table descriptions: http://xcat.sf.net/man5/xcatdb.5.html
- xCAT mailing list: http://xcat.org/mailman/listinfo/xcat-user
- xCAT bugs: https://sourceforge.net/tracker/?group_id=208749&atid=1006945
- xCAT feature requests: https://sourceforge.net/tracker/?group_id=208749&atid=1006948
- xCAT wiki: http://xcat.wiki.sourceforge.net/
- LoadLeveler documentation: http://publib.boulder.ibm.com/infocenter/clresctr/vxrx/index.jsp?topic=/com.ibm.cluster.loadl.doc/llbooks.html
- GPFS documentation: http://publib.boulder.ibm.com/infocenter/clresctr/vxrx/topic/com.ibm.cluster.gpfs.doc/gpfsbooks.html