xCAT 2.0 Alpha Release Cookbook
10/27/2007

1.1 Release Description

xCAT 2.0 is a complete rewrite of xCAT 1.2/1.3 implementing a new architecture (see description at end of this document). All commands are client/server, authenticated, logged and policy driven. The clients can be run on any OS with Perl, including Windows. The code has been completely rewritten in Perl, and table data is now stored in a relational database. For the alpha, we are including SQLite with the xCAT OSS rpm.

The code is being released as RPMs and SRPMs . For the alpha release, there is support for x86_64 hardware (IPMI and Blades only) . The OS must be RedHat 5 or CentOS5.

The alpha code should not be used for production work When the beta is released, it may not be compatible with the alpha version.

1.1.1 Function supported:

- Tools to manipulate the database tables: tabdump,tabrestore,tabedit, chtab (must be run on the server), nodech, nodeadd,noderm.
- Cluster setup commands: makehosts, makedhcp,makeconservercf are available.

- Notification: infrastructure allowing users to register for xCAT database table changes.

- xdsh/xdcp - Parallel remote and remote copy commands. See xCAT 2.0 Commands(xdsh) below.

- Node discovery and diskfull deployment of CentOS5 and RHELS5 on the supported hardware (see Prerequisites).

- For a list of all 2.0 xCAT commands run ***rpm –ql xcat-client***.

1.1.2 Function not supported

- Diskless/Stateless clusters
- No imaging
- No flash
- You might be able to get it to work, but not supported in the alpha:
- SLES 10
- Diskless/stateless support using Perceus
- pSeries hardware control using HMC, IVM, FSP for Power5 and Power6 hardware
- Web GUI interface
- Data abstraction commands to make creating node and other database definitions easier
- xCAT monitoring plug-in infrastructure

1.1.3 Prerequisites:

- Hardware requirements:
  - x3455, x3550, x3650, x3455, LS21, HS21, LS41, x336, x346
  - no SOL for x386 or x486
  - Must be IPMI based, rack mounted unit.
  - Blades
  - Ethernet switch must be SNMP enabled for node discovery.
- Software supported
  - RedHat5 or CentOS5

1.1.4 Licensing

xCAT 2.0 is OSS with a EPL license. For license information visit

http://www.opensource.org/licenses/eclipse-1.0.php

1.2 Installing xCAT 2.0 Software

Install your xCAT management node with RedHat5 or CentOS5, making sure to install all packages available with the distribution to reduce the number of dependency RPMs you need to track down.

To install xCAT 2.0 using YUM

1. Down load the following two files to the management node and place in the /etc/yum.repos.d directory.
    1. http://xcat.org/xcat/xCAT-core.repo
    2. http://xcat.org/xcat/xCAT-oss.repo

   Make sure that the management node hostname resolves to the ip address set in /etc/hosts. The output of "hostname –d" should print the correct domain name. If /etc/hosts is set with the long and short hostname, this should work.

   For example: 7.113.47.250 rh5.clusters.com rh5 line in /etc/hosts results in

   [root@rh5 ~]# hostname -d

   clusters.com

2. If OpenIPMI-tools is installed on your system, remove it "rpm –e OpenIPMI-tools". The version shipped with Redhat 5/CentOS5 is back-level and has many bugs. xCAT will be installing a newer version from the xCAT-oss.repo.
3. Now run "yum install xCAT" to install the xCAT and dependent OSS rpms. Postscripts in the rpms will set up default xCAT configurations and start the xcatd daemons.
4. If you are reinstalling xCAT 2.0, run "yum update" to update the xCAT packages.
5. Check to make sure the xcatd daemons started:

```
[root@rh5 xCAT-core]# ps -ef | grep -i xcat
root      3471     1  0 14:07 ?        00:00:00 xcatd: SSL listener
root      3472  3471  0 14:07 ?        00:00:00 xcatd: UDP listener
root      3473  3471  0 14:07 ?        00:00:00 xcatd: install monitor
```

   If not, start it manually:

```
[root@rh5 xCAT-core]# service xcatd start
Starting xCATd
```

6. Edit /etc/xinetd.d/tftp and change "disable = yes" to "disable = no" and run "service xinetd restart" to enable the TFTP server.

1.3 xCAT 2.0 Commands

Note: use '<xCAT command> -h' for a usage message from each command. MAN pages are not available at this time.

XCAT COMMAND	DESCRIPTION
chtab (Note: will be renamed to **tabch** in beta)	To add or update rows in a table. Allows you to add nodes, create groups, add attributes to the xCAT tables. chtab node=devnode01 nodelist.group=all,compute will add a new node devnode01 to the nodelist table and assign to the all and compute groups. chtab key=rsh site.value=/usr/bin/ssh will assign the site table rsh attribute to /usr/bin/ssh chtab –d node=devnode01 will delete the previously create node from the nodelist table.
copycds	Copies Linux distributions and service levels to install directories.
makeconservercf	Make Conserver Configuration
makedhcp	Sets up the DHCP server.
makehosts	Creates entries in /etc/hosts for nodes. Node nodenames and ip addresses must be setup in the hosts table.
makenetworks	Builds the networks table
nodeadd	Add a node to the cluster For example: nodeadd <noderange> [table.column=value] [table.column=value]…. nodeadd blade1-blade7 nodelist.groups=all,compute nodeadd also supports some short cut tags: groups is equivalent to table.column = nodelist.groups • nodeadd blade1-blade8 groups=all,compute mgt is equivalent to table.column = nodehm.mgt • nodeadd blade7 mgt=blade switch is equivalent to table.colum= switch.switch • nodeadd blade8 switch=switch1
nodech	Change node information
nodels	Display information about a node or range of nodes or all nodes
noderm	Remove Node
nodeset	Installs, boots the nodes uses pxe.
psh	Runs a command across a list of nodes or nodegroups in parallel

rbeacon	Turns beacon on/off/blink or gives status of a node or a range of nodes.
rbootseq	For boot of Bladecenter node range. Change each node boot order.
regnotif	Register a Perl module or a command that will get called when changes occur in desired xCAT database tables. See Using xCAT Notification
reventlog	Retrieves or clears remote hardware event logs
rinv	Retrieves hardware configuration information for a single or range of nodes
rpower	Boots, resets, powers on and off and queries nodes
rsetboot	rsetboot (IPMI) is a way to specify the singular device to try to boot only for the next power cycle
rspreset	Used to reset service processors out-of-band
rvitals	Retrieves hardware vital information from the on-board Service Processor for a range of nodes
tabdump	Display Database table information for table requested. tabdump with no input will display a list of all valid table names.
tabedit	Edit a table . Must export EDITOR to define your editor.
tabrestore	Restore a table from the table.csv template or from a tabdump output file.
unregnotif	Unregistered a Perl module or a command that was watching for changes of desired xCAT database tables.
xdcp	Concurrently copies files to/from multiple nodes. Requires dsh rpms installed. See xdsh.
xdsh	Concurrently runs remote commands on multiple nodes. dsh rpms should be obtained from the following website. http://www14.software.ibm.com/webapp/set2/sas/f/csm/download/home.html. Download the "Cluster System Management for Linux Multiplatform" 1.6.0.13 tarball. You will only need to install csm.dsh1.6.0.13* rpm from the tarball.

1.4 xCAT Tables

Note: The Database Table Schema can be viewed in the /usr/lib/perl5/site_perl/5.8.3/xCAT/Schema.pm file or by running the tabdump command.

TABLE NAME	DESCRIPTION
chain	Lists action that occur during node install, node boot . Used by nodeset.

hosts	List of hosts, alias hostname, ip addresses. Used to update /etc/hosts with makehosts
hmc	List information about the hmc – hcp, username, password
ipmi	Lists information on the nodes IPMI interface – bmc, username, password
ivm	Lists information on the nodes IVM interface – hcp, username, password
mac	Lists mac address for each node.
mp	This is the management processor network. Whereas the mpa.tab lists the adapter, this table lists devices that are networked off that adapter via daisy chained networks, or in the case of Blade Center, an internal network..
mpa	Lists the MPA, username and password for the nodes.
networks	Defines masks, gateways and DNS servers. Build my makenetworks command.
nodehm	Defines the hardware management method for each node.
nodelist	Defines all nodes and groups.
noderes	Installation resources for the node.
nodepos	Node physical location
nodetype	Node install type (osversion, arch, type)
notification	Lists the Perl modules and commands that will get called for changes in certain xCAT database tables.
passwd	user names and passwords used by xCAT scripts
policy	Table controls the policy for the execution of the xcat commands.
ppc	Store Series p hardware components – HMC, IVM, BPA, FSP, LPAR
site	Main xCat configuration file. Holds global information for the cluster.
switch	Lists switch interface(s) for the node.
vpd	Vital product data table. Holds machine serial number and model type.

1.5 Adding and Installing Nodes

1) Check the default required site table attributes:

```
[root@rh5 xCAT-core]# tabdump site
#key,value,comments,disable
"xcatdport","3001",,
"master","9.114.47.251",,
"domain","ppd.pok.ibm.com",,
"installdir","/install",,
"timezone","America/New_York",,
"nameservers","176.60.50.209",,
```

   To change any of these values, use chtab or tabedit.
   chtab:
   a) chtab key=domain site.value=<your domain name>
      For example: chtab key=domain site.value=clusters.com
   b) chtab key=master site.value=<ip address on the cluster network of Master node>
      For example: chtab key=master site.value=8.777.43.5
   c) chtab key=dhcpinterfaces site.value=<comma delimited list of nics to run dhcp>
      For example: chtab key=dhcpinterfaces site.value=eth1

   tabedit:
   a) export EDITOR=vim (or your favorite editor)
   b) tabedit site
   c) make your changes, and use the editor command to save the file and quit. Your changes will automatically be imported into the xCAT database.
2) Check the 1350 default database template files in /usr/share/xcat/template/e1350 directory to see if they apply to your environment. These templates, or templates you create from them, can be used to load the database xCAT tables using the ***tabrestore <path to template>*** command. The README, in the directory, explains how to use these files.
3) The ***tabdump <tablename>*** will dump current contents of the database table. This can be used to dump the contents of a table and, if you redirect the output to a file, you can later reload the data using ***tabrestore***.
4) Use ***tabedit <tablename>*** to make any needed changes to the tables. Check the previous released xCAT tables for definitions. The 2.0 tables contain a header with the format of the fields in comments.
5) Define the nodes in your cluster by using the nodeadd command. Ensure that all nodes, bmcs or management modules, and switches have hosts definitions, or the dhcp configuration will not update, and the bmcsetup will not receive meaningful data. (see nodeadd command in the xCAT Tables).
6) If you want ***makehosts*** to update the /etc/hosts file for the defined nodes, bmcs/mms, and switches, use tabedit to update the ***hosts*** table with the hostnames and ip addresses to be added to /etc/hosts. Then run ***makehosts .***
7) ***makenetworks*** runs during the xCAT install and updates the networks table. You should ***tabdump networks*** to ensure the setting are correct. If any need changing, ***tabedit networks*** table. Ensure the networks to be managed have the “dynamicrange”

set to a hyphenated range of IP addresses to serve as staging for nodes being brought up**.** If any new networks are added, the ***makenetworks*** should be run again.

8) Run ***makedhcp –n***. Review the /etc/dhcp.conf file created to ensure all your network definitions are correct. Note that the node host definitions will no longer appear here, but rather will appear in the leases file (/var/lib/dhcpd/dhcpd.leases) after the initial DHCP request from the node. xCAT 2.0 sets up dhcp to use the OMAPI command shell to setup, query and change the dhcp configuration. See **man omshell**, and http://linux.die.net/man/3/omapi for more information.
9) Run "service dhcpd start" to load the initial omapi dhcp configuration.
10) For blades, make sure your bladecenter management module is configured for the SNMP protocol:
    a) Telnet into you management module. Once in, do the following (assumes "mm[1]" is the current active mm and "PASSW0RD" is your mm password).
    b) env -T mm[1]
    c) users -1 -ap sha -pp des -at set -ppw PASSW0RD
    d) Log off the management module and test the connection with a query command such as ***rpower <noderange> stat*** or ***rinv <noderange> all***.

    Note: This was only tested with the latest release level firmware BPET32D. Older firmware may not properly support SNMP.
11) Set up conserver. Note, **rcons** is not yet supported for xCAT 2.0. You will need to manually start the conserver daemon and open consoles.
    a) Update the nodehm table (***tabedit nodehm***) to set fields for **cons**, **termport**, and **termserver** for your nodes. Currently, supported values for **cons** are "blade" and "ipmi".
    b) Run makeconservercf to generate a conserver 8 configuration file. Review /etc/conserver.cf. Make sure you have valid "trusted" entries in the "access{}" stanza for any host starting a console (most likely your management node).
    c) Start the conserver daemon: service ***conserver start***
    d) Try opening a console: ***console -M <management node> <node>***
12) xCAT 2.0 will discover your hardware:
    a) Create the initrd:
        (1) rm /tftpboot/pxelinux.cfg/default
        (2) ***mknb x86_64*** (creates the netboot image and writes out the master parameter to the /tftpboot/pxelinux.cfg/default file).
    b) Make sure your boot sequence is set to boot from network before harddrive: ***rbootseq <noderange> list***
    If not, change it: ***rbootseq <noderange> f,c,n,h***
    c) Power up the system using ***rpower <noderange>.***
    d) Within a few seconds of booting to the network, any BMCs should be configured and be setup to allow ssh. All nodes will be network booted (you can watch /var/log/messages for DHCP and TFTP traffic).
    e) ***nodels <noderange> vpd.serial vpd.mtm mac.mac*** should show interesting data after discovery.
13) Run ***copycds*** with full path to the ISO images
14) Run ***nodech*** (or ***tabedit***) to change nodetype OS and setup node profile :

nodech <noderange> nodetype.os=<os> nodetype.profile=compute

(for now only, the **compute** template file has been provided. See /usr/share/xcat/install/). Current possible values for **os**: rhels5, rhelc5,centos5 If using 64 bit distro, the **nodetype.arch** should have been populated with "x86_64" at discovery time. If not, set this value, too. This is the only architecture supported for now.

15) Run ***nodech*** (or ***tabedit***) to set noderes nfsserver :
nodech <noderange> noderes.nfsserver=<server>
16) Run ***nodeset <noderange> install***, to install the OS.
17) Run ***rpower <noderange> boot***, to boot the systems.

- The kexec to installers doesn't have the client scripts written yet, necessitating the reboot, if wanting to try kexec for now, you have to manually transfer the kernel, initrd, and run kexec -f with the right arguments to the xCAT nbfs environment)

1.6 Using xCAT Notification

With xCAT 2.0, you can monitor xCAT database for changes such as nodes entering/leaving the cluster, hardware updates, node liveness (to be added later) etc. In fact anything stored in the xCAT database tables can be monitored through the xCAT notification infrastructure. To start getting notified for changes, simply register your Perl module or command as the following:

regnotif *filename tablename -o actions*
where
filename is the full path name of your Perl module or command.
tablenames is a comma separated list of table names that you are interested in.
actions is a comma separated list of data table actions. 'a' for row addition, 'd' for row deletion and 'u' for row update.

Example:
regnotif /usr/lib/xcat/monitoring/mycode.pm nodelist,nodehm -o a,d
regnotif /usr/bin/mycmd switch,noderes -o u
Use the following command to view all the modules and commands registered.
tabdump notification

To un-register, just do the following:
unregnotif *filename*
Example:
unregnotif /usr/lib/xcat/monitoring/mycode.pm
unregnotif /usr/bin/mycmd

If the *filename* specifies a Perl module, like /usr/lib/mypath/xxx.pm, the package name must be **xCAT_monitoring::xxx**. It must implement the following subroutine which will get called when database table change occurs:
processTableChanges*(tableop, table_name, old_data, new_data)*
where:
tableop Table operation. It can be 'a' for row addition, 'd' for row deletion and 'u' for row update.
tablename The name of the database table whose data has been changed.

old_data An array reference of the old row data that has been changed. The first element is an array reference that contains the column names. The rest of the elements are array references each contains attribute values of a row. It is set when the action is u or d.

new_data A hash reference of the new row data; only changed values are in the hash. It is keyed by column names. It is set when the action is u or a.

If the file name specifies a command (written by any programming languages or scripts), when the interested database table changes, the info will be fed to the command through the standard input. The format of the data in the STDIN is as following:

```
action(a, u or d)
tablename
[old value]
col1_name,col2_name...
col1_val,col2_val,...
col1_val,col2_val,....
...
[new value]
col1_name,col2_name,...
col1_value,col2_value,...
...
```

Sample code:

File name: **/usr/lib/xcat/monitoring/mycode.pm**.

Use command

regnotif /usr/lib/xcat/monitoring/mycode.pm nodelist -o a,d

to register for nodes adding or removing in the nodelist table.

```
package xCAT_monitoring::mycode;
1;
# This subroutine get called when new nodes are added into the cluster
# or nodes are removed from the cluster.
#
sub processTableChanges {
  my $action=shift;
  if ($action =~ /xCAT_monitoring::mycode/){
    $action=shift;
  }
  my $tablename=shift;
  my $old_data=shift;
  my $new_data=shift;

  my @nodenames=();
  if ($action eq "a") { #nodes added in the cluster
    if ($new_data) {
      push(@nodenames, $new_data->{node});
      $noderange=join(',', @nodenames);
      #log the node names that have entered the cluster
      open(FILE, ">>/var/log/mycode.log") or dir ("cannot open the file\n");
      print (FILE "new nodes in the cluster are: $noderange\n");
      close(FILE);
    }
  }
  elsif ($action eq "d") { #nodes removed from the cluster
    #find out the index of "node" column
    if ($old_data->[0]) {
      $colnames=$old_data->[0];
      my $i;
      for ($i=0; $i<@$colnames; ++$i) {
        if ($colnames->[$i] eq "node") {last;}
```

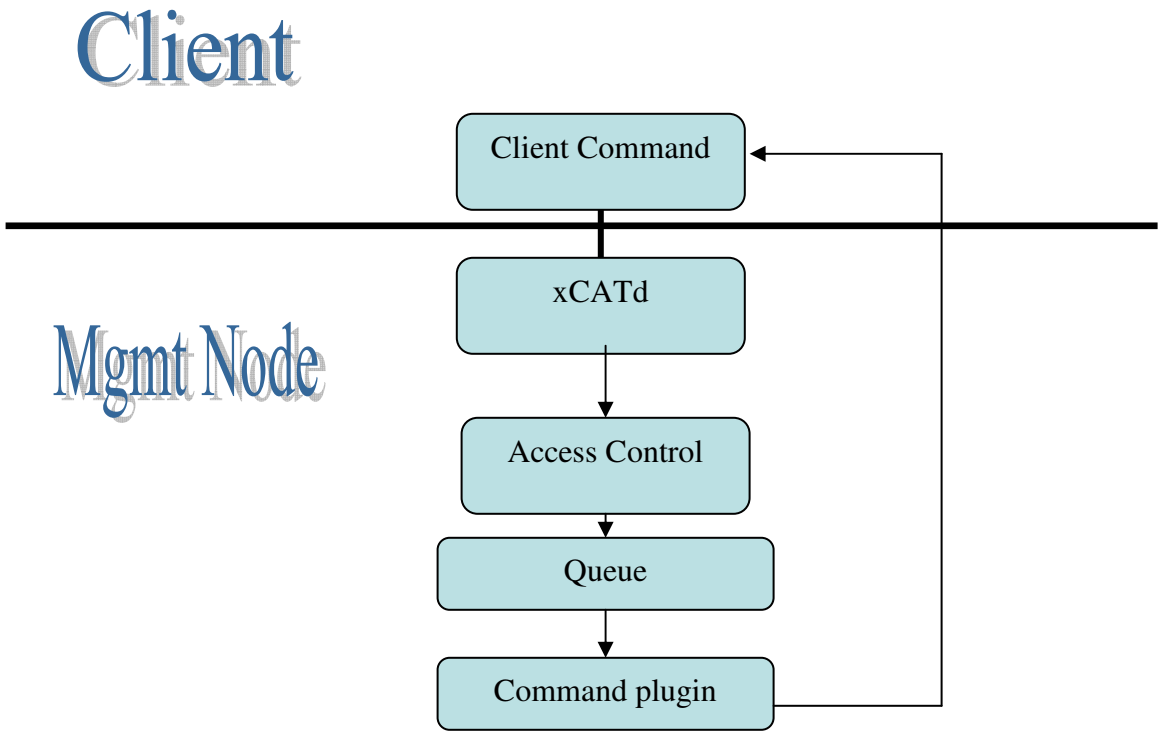
```
        }

        for (my $j=1; $j<@$old_data; ++$j) {
          push(@nodenames, $old_data->[$j]->[$i]);
        }

        if (@nodenames > 0) {
           $noderange=join(',', @nodenames);
           #log the node names that are leaving the cluster
           open(FILE, ">>/var/log/mycode.log") or dir ("cannot open the file\n");
           print (FILE "nodes leaving the cluster are: $noderange\n");
           close(FILE);
        }
      }
    }
    return 0;
}
```

1.7 xCAT Architecture

General/Overall Concepts

The heart of the xCAT 2.0 architecture is the xCAT daemon (xcatd) on the management node. This receives requests from the client, validates the requests, and then invokes the operation. The xcatd daemon also receives status and inventory information from the nodes

.



1.7.1 Client/Server

1.7.2 Flow

- User invokes an xcat cmd on the client
- The cmds can either be a sym link to xcatclient or a thin wrapper that calls xcatclient.
- Some cmds will implement their own xcatclient function, if they have more processing than the generic xcatclient function supports. (e.g. xdsh/xdcp).
- The xcatclient function packages the info into xml and passes it to xcatd
- xcatd receives the request and forks to process the request
- The ACL/Role Policy Engine determines whether this person is allowed to execute this request. It evaluates the following info:
  - The cmd name and args
  - Who executed the cmd on the client machine
  - The hostname/IP address of the client machine
  - The node range passed to the cmd
- If the ACL check is approved, the cmd is passed to the Queue:
  - The queue can run the action in either of 2 modes. The client cmd wrapper decides which mode to use (although it can give the user a flag to specify):

        - Keep the socket connection with the client open for the life of the action and continue to send back the output of the action as it is produced.
        - Initiate the action, pass the action ID back to the client, and close the connection. At any subsequent time, the client can use the action ID to request the status and output of the action. This is intended long running cmds.
    - The Queue logs every action performed, including date/time, cmd name, arguments, who, etc.
    - In phase 2, the Queue will support locking (semaphores) to serialize actions that should not be run simultaneously.
- To invoke the action, the xml is passed to the process_request() function of the appropriate plugin pm which contains the code for the function being run.
    - With the request examined per policy table, and noderange expanded to nodes, the request is passed in its entirety (including tags otherwise ignored) to a plugin's process_request function, which will receive two arguments, the first the aforementioned hash reference, the second a reference to a callback function to call per response message to send back.
    - The appropriate pm is chosen by loading all of the plugins from /usr/lib/xcat/plugins and invoking handled_commands to see which cmds each pm handles.
    - Data is returned from the command plugin back to the client command handle_response routine.