SV1 Cluster
User Data Base Feature

Richard Lagerstrom
Resource Management
SGI
rnl@sgi.com

ABSTRACT: *Administration of an SV1 cluster requires that the user information contained within the User Data Base (UDB) is to some degree consistent among all the nodes. This feature offers central management of the UDB files local to each node in the cluster.*

Introduction

Each node of an SV1 cluster will have a local User Data Base (UDB) just as if it were a stand alone system. The administration of an SV1 cluster requires that the user information contained within the data base is consistent among all nodes in the cluster. This paper describes a feature to enforce that requirement to the level specified by the administrator within the range of flexibility provided by this feature.

Design Outline

There are two distinct parts comprising this feature; the node level UDB maintenance tools and the cluster level UDB administration tools. The cluster UDB tools use the node level tools to effect its requests.

Each node will have the UDB files common to all Unicos systems. Additionally, each node must have access to the cluster UDB configuration file, the distribution files and the cluster UDB source file.

Node Level UDB Maintenance Tools

The existing Unicos `udbgen` and `udbsee` tools are the basis for this feature. One of the design rules is to maintain upward compatibility with the existing UDB. Making sweeping changes to the UDB (although tempting) not only causes a great deal of administrative work but could also pose authorization risks and denial of service failures.

Cluster UDB administration is done using an enhanced UDB source file. This method has been selected to provide the greatest amount of flexibility to the administrator and a minimum dependence on the local UDB files. This choice also allows an administrator to develop special source manipulation aids using available tools such as `awk` or `perl`.

Changes to the UDB Source Language

To support many nodes with common UDB source and at the same time allow each node to have tailored user fields makes it necessary to include target node names with the source. Upward compatibility is also essential for easy conversion from a stand alone to a cluster environment. This will be done by adding a new `field-name:field-value:` pair to the source language[1] as follows:

- `node:node-name:` The new field name “node” will be recognized anywhere a field name may appear. `Node-name` can be a single node name, a comma separated list of node names or “*”. A null name (`node::`) is ignored. `Udbgen` will accept any subsequent name-value pairs in the input file as input directives only if a node-name given on the node directive matches the host name of the machine on which `udbgen` is running or node-name is “*”. `Udbgen` sets the initial node-name as though a node:host-name: directive had been read. If the current node-name does not match the machine’s host name, `udbgen` will skip directives as though they were comments after examining each field name for the keyword “node.”
- The rule that an introductory keyword appears as the first non-comment token on a line is relaxed to allow a preceding node directive on the same line.

[1] See the udbgen(8) man page for all the details of the UDB source language

Copyright 1999. SGI. All rights reserved.

- A new introductory keyword, `change`, has been added so records may be created or updated from the source. This is necessary since the UDB may not have the same record population on all nodes when a cluster UDB source file is distributed.

Since `udbsee` is expected to be the generator of the UDB source files, its output has been modified to include the `node:node-name:` pair in every output record if the new node (`-n`) option appears. If the source is intended for input to the cluster UDB, specific options (`-agnv -t change`) must be used to get the needed source information.

Changes were made to `udbgen` to handle the `node` directive. No changes to the existing `udbgen` options or interface were needed.

New Cluster Level UDB Maintenance Tools

Cluster UDB maintenance tools support or assist in these administrative functions:

- Add or delete records from all local UDBs with one action
- Add or delete user's system information using locally written scripts
- Update `/etc/passwd` to match the local UDB
- Alter specified fields in the UDBs on all or selected nodes
- Enforce cluster wide uniformity of user identity fields such as name, password, `uid`, `gid` and `acid` lists, and home directory. If a password or default shell is changed by the user or administrator, the corresponding field of the user's record on each node should be changed. Password and shell changes should optionally trigger a site specified process to perform site dependent work
- Optionally enforce cluster wide uniformity of the share tree structure, share mode (`uid` or `acid`), share entitlement and share group membership
- Allow node by node variations on limits, permissions, quotas, and other fields

Resiliency is also important in order to keep the nodes of the cluster synchronized as intended. Specifically,

- Information should not be lost if one or more nodes fail
- Logs of changes should be maintained
- When a node joins the cluster a check that all changes have been applied is made. There are two cases to consider: a node newly added to the cluster and a node joining the cluster after being temporarily disconnected. Pending update files are used to handle the rejoining case. Distribution of the Cluster UDB populates the UDB of the new node as desired.

Implementation

The core of cluster UDB administration is to categorize what the cluster environment should be and, from a user's view, how similar the nodes should appear. To accomplish this the Cluster UDB configuration file[2] allows uniformity rules to be established in a flexible way. There are three categories into which records and fields may be placed:

- Non-configurable and node invariant record identity information. The fields in this category are `name` and `uid`. Discounting any configured exceptions, all records must comply.
- Node invariant fields. The administrator names these fields in the configuration file with the `Uniform` keyword. The released configuration file specifies most of the UDB fields to be in this category
- Node specific information having no cluster wide meaning. Such fields as `shextime` and `shusage` belong to this category. The merge process will skip fields in this category. The administrator names these fields in the configuration file with the `Drop` keyword

The fields belonging to each category except the first will be configurable. Exceptions to the rules will also be configurable. To make it possible to define special UDB records on a node by node basis, a way to exempt records by UID value ranges or an enumeration of exceptions by UID or user name is provided. No pattern matching or other advanced capabilities for selection appear to be needed and are not available. The `Exempt` keyword is used to specify exception records.

The Cluster UDB configuration file is used by all of the cluster UDB tools. The file must be able to be read from all nodes. In addition to specifying the uniformity rules, the configuration file is needed to specify the directories where the cluster information files are located and a template for new record creation. All path information must be specified by the site administrator when installing this feature since path names to nfs or other shared files have no common naming plan.

Cluster UDB tools may be run from any node of the cluster. There is interlocking provided to reduce the possibility of confusion if these tools are being run simultaneously somewhere in the cluster. We recommend to coordinate[3] cluster UDB administration to avoid possible race conditions among the tools. Such races, when the same record is involved, could result in violation of node invariant rules.

Each of the four cluster UDB tools, the server and the site module will be described below. Some man-page style information appears but only to demonstrate the feel of the commands. The options and features are not completely described here. The commands are ordered by the typical sequence in which they would be used.

UDB Source Collection

The information from the UDB on each node must be made available for merging and rule evaluation as the first step in

[2] A recommended configuration file is supplied with the release materials.

[3] Only one administrator should be maintaining the cluster UDB at a time.

establishing cluster UDB administration. To do this the udb_collect tool causes udbsee to run on each node to write the source to a specified file in the cluster UDB directories.

Which nodes to include in the cluster is specified in the configuration file.

UDB Source Merging and Rule Verification

The UDB source from each node made available by the udb_collect command will be read by udb_merge and converted into a single cluster UDB source file. In each record a field having identical values in each node's UDB will be merged into a single field description. Non-uniform fields will be identified with the appropriate node names. Node specific fields will be dropped from the merged data. Required node invariants will be enforced and exceptions will be handled as well.

In order to deliver the flexibility necessary for specific needs the fields belonging to the node invariant and node specific categories will be configurable. Exception records may be specified by UID or user name. These records will not be required to follow the node invariant rules. Fields configured to be dropped will be eliminated from all records including exception records.

Dropped fields are intended to prevent the merged source file from containing any node specific information that should not be restored to the node's UDB when UDB synchronization is done. Fields such as share exit time and share usage fall into this category.

The command will exit with a status of 1 if it detects any errors. A completion message showing the number of errors will appear on stdout. The log file should be examined if errors are indicated. Field uniformity errors will also be marked in the cluster UDB source file using comment lines. The log file will record the date, the command line used for the run and the configuration as interpreted by the tool.

UDB Source Editing

The udb_edit command is used to generate a template record for creation of a new user record or to edit an existing cluster UDB record. For a new record the template is extracted from the cluster UDB configuration file. To edit an existing record, specify the user name to select from the cluster UDB source file. Only one record may be created or edited per udb_edit invocation.

A file is created in the directory named on the EditDirectory configuration statement. The name of the file is *username.yyyymmddhhmmss*. The date suffix makes the filename unique and orders the files on time of creation. This ordering is needed for proper update sequencing.

An editor is then started to allow the administrator to manipulate the record being changed or created. The EDIT environment variable chooses the editor. If no EDIT variable exists, udb_edit terminates with a message naming the file holding a copy of the unedited record. If an editor was executed and it terminates with a zero error code, the administrator is asked if the edited record should be saved. If the answer is no or the editor indicates an error exit, the file is deleted and no changes are retained. If the answer is yes and this is an update to an existing record, udb_edit simulates how the changes found in the edited file would alter the original record. A udbgen directive file is written over the edited file containing only the directives needed to alter the UDB record as intended. If this is a new user creation request, the edited record is used as is from the editor.

Each edited record results in an update file containing the udbgen directives needed to make the changes specific to that edit operation.

UDB Cluster Updating

When one or more changes have been made using udb_edit, udb_update is invoked to distribute the changes to the cluster. Actually, udb_update has a number of functions to support the idea of distributing change information to the nodes. The significant options are:

- **-u file1 file2.. filen**. Update the cluster from the list of update files. All of the named files will be concatenated in the order they appear in this list into a single file for distribution to all the nodes in the cluster. The file names are relative to EditDirectory unless they begin with "/".
- **-s source_file**. Synchronize the node UDBs. The cluster UDB source file named will be distributed to either the configured or the explicitly named nodes in the cluster. The file name is relative to DistributionDirectory unless it begins with "/".
- **-e user_name field**. The UDB record for the named user will be read from the UDB local to the node on which the command is running and the value contained in the named *field* will be placed in a udbgen directive file as though the administrator had used udb_edit. This results in an immediate update of all other nodes in the cluster. For example, if the password for user *uuu* has changed on this node, udb_update would be started with the command udb_update -e uuu passwd.
- **-j**: Join the cluster and apply pending changes. Udb_update will examine the files in DistributionDirectory for any that match the pattern Dist*.*nodename*. (The Dist prefix is set by the DistributionFile configuration directive.) Only the node name matching the host name of the node on which udb_update is executing will match. Changes will be applied in order of decreasing age.

The distribution file created by udb_update -u and -e options will be in the DistributionDirectory with the name prefix specified by DistributionFile. The released prefix is Dist. The file name will also encode the creation time for resiliency and ordering. The file name using the released configuration has the form Dist*yyyymmddhhmmss*.

The list of nodes to contact is provided through the NodeList configuration directive. Udb_update will create hard links to the distribution file for each node using the node specific name Dist*yyyymmddhhmmss.node*. If the distri-

bution file is generated with the `-u` or `-e` options, the file to which the hard links point will be unlinked before node distribution is started. When all of the node specific links are deleted, the distribution file will automatically disappear so no administrative cleanup is necessary. If the `-s` option is used, the links will point to the source file but that file is not unlinked and so remains after node distribution is finished.

As each node completes the update the link for that node will be removed. Any links remaining after the `udb_update` completes means those nodes did not perform the update.

Change Feedback to Cluster UDB Source

Updates requested with the `-u` or `-e` options need to be reflected in the cluster UDB source file so subsequent use of `udb_edit` displays correct information. This is done by appending the distribution file to the existing cluster UDB source file.

Change Logging

When a distribution file is created, a copy is appended to the change log to record the action. If the cluster source file is distributed (`-s` option), the change log contains the action but not the content of the source file. The change log is in `LogDirectory` and named by the `ChangeLog` configuration directive.

Performing Node Actions

A server named `udb_server` has been written to manage the commands run by the nodes. This is a simple server initiated by `inetd` on behalf of the `udb_collect` or `udb_update` tools. All response messages are written to `stdout` or `stderr`. For security reasons the configuration file must exist and be named `/etc/config/udb/clusterUDB.conf`.

All `udb_server` options must appear on the command line. The directive is read from `stdin` and consists of options and arguments for `udb_helper` followed by the prototype of the command to execute on the node, thus making it possible to run any authorized command without changing network configuration information.

Only commands residing in the directory named by `CommandDirectory` in the configuration file are authorized. To reduce the possibility of security breaches path names may not be included in the `udb_helper` options or on the command prototype. To enforce this the "/" character may not appear in the text read from `stdin`.

Site Support

In order to provide services that may be tailored by the site the shell script `udb_helper` will be executed directly by the server. The input from `stdin` (after the place holders are replaced by the proper strings and the absolute command path is specified with the `-b` option) will be passed to this script as its parameters. At some point `udb_server` must execute the requested command.

User Initiated UDB Updates

Certain node invariant UDB data can be changed by the user.

- When the default shell is changed by the user running on a specific node, that change should appear on all other nodes within a reasonably short period of time.
- The user can change password at any time. Usually this does not happen very often but sites using password aging may have a fairly high frequency of password change. In addition to propagating the new password to all nodes, some environments also need to make this change visible to an authentication server. `Udb_helper` is intended to perform this specialized role.

The commands `chsh`, to change the login shell, and `passwd`, to change the password, will be modified so cluster distribution of the information can occur quickly. User initiated actions that change the cluster UDB have the possibility of occurring on more than one node at nearly the same time. The cluster UDB administration tools do not prevent multiple changes to the same field within a short period of time. In this case it is possible for node UDBs to end up with different final field values. Periodic source collection and merging will be needed to detect and correct this situation.

Summary

Four commands, a server and a site helper have been added to Unicos to support centralized administration of the User Data Base (UDB) on SV1 clusters. This feature distributes changes to the UDB throughout the cluster as required by site policy. It is assumed that the administrator will never use the `udbgen` command to update the UDB once this feature is installed.

Every node in the cluster must be able to access a common set of cluster UDB files in order to use this feature. Security is provided through normal file access policy. In addition, all of the cluster UDB commands are restricted to privileged users and the server is able to execute only a specific set of commands. All changes are logged for auditing or problem analysis.