AutoRID module extensions to control ID generation

Saurabh Gawande, Abhidnya Joshi, Mathias Dietz
IBM

Introduction

- Saurabh Gawande, Abhidnya Joshi

  Working for *IBM India Systems and Technology Lab (ISTL)* in Pune on the IBM SONAS and Storwize V7000 Unified product.

- Mathias Dietz

  IBM Research and Development in Mainz/Germany. Architect in the IBM SONAS and Storwize V7000 Unified product team. Experience with Samba as part of the IBM SoFS offering since 2006 and the IBM OESV offering since 2003

IBM SONAS - Scale Out Network Attached Storage

Modular high performance storage with massive scalability and high availability.
Supports multiple petabytes of storage for organizations that need billions of files in a single file system.

http://www.ibm.com/systems/storage/network/sonas/

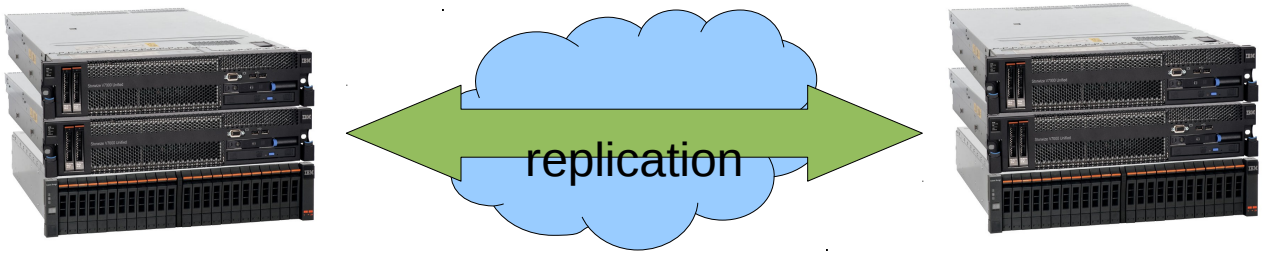
IBM Storwize V7000 Unified

A virtualized storage system designed to consolidate block and file workloads into a single storage system for simplicity of management, reduced cost, highly scalable capacity and high availability.

http://www.ibm.com/systems/storage/disk/storwize_v7000/

The Need!

- Unix based systems use UID/GID for authorization/access to files, whereas Windows based systems use SID.
- Unix based storage systems use Identity management modules from Samba to map Windows SID to Unix UID/GID.
- Complexity of customer environments is increasing, a typical environment has grown over years and consists of multiple (trusted) domains and sites.
- In today's world, **replication of storage systems is a common use case**. Hence to support replication, its mandatory to have same ID maps (xid ↔ sid) on all storage systems.



ID mapping modules in Samba

- Samba already provides several idmap modules
  - idmap_tdb2, idmap_rid, idmap_autorid, idmap_ad, ...
- Some of them (i.e. rid,ad) support the replication use case already. However, these modules have some disadvantages which makes them not the best choice for a storage system.
  - idmap_rid cannot be used by storage admins without prior knowledge of domain information
  - idmap_ad requires Active Directory schema extension and maintenance of dedicated UNIX UID/GID fields.
  - Both do not deal with BUILTIN and Well-known SIDs
- 'Autorid' ID mapping module eliminates the need of this knowledge but .....

“Autorid” Module – Challenge 1

Range allocation is NOT deterministic

Range of an AD domain is allocated on first come first served basis.

i.e an AD domain which access the storage system first gets the 1st range, whereas the same AD domain might get 2nd range on another storage system.
This may lead to different id mappings on different storage systems.

Thus 'Autorid' module cannot be used for replication.

“Autorid” Module – Challenge 1

Approach

In a multi-system environment where replication is used, configure one system as a ID mapping master and the others as slave.

Prevent creation of new domain ranges on slave systems by enabling Autorid “read only” option.

Provide a mechanism to export the ID mapping configuration from one storage system and import on another system.

Note: Since Domain additions are relatively seldom, the synchronization between multiple systems can be done on a scheduled base (daily, weekly)

Implementation details for challenge 1

To export/import the “Autorid” configuration from the master to the slave systems the following commands are proposed

EXPORT

net idmap autorid getrange

This command will dump the domain SID and domain number from autorid.tdb.

net idmap autorid getconfig

This command will dump the CONFIG section from autorid.tdb

IMPORT

net idmap autorid setrange

This command will create the ID map for the specified domain SID

net idmap autorid setconfig

This command will set the CONFIG section in autorid.tdb

Note: net idmap commands will work directly on the autorid tdb and therefore ignore read-only option of autorid.

Challenges - “Autorid” module

Challenge 2

Large RIDs may not fit into the range

In the current “Autorid” algorithm only one range is allocated to every domain. The current algorithm does not take into account the increase in RID of the domain. Hence, if a legitimate registered domain's RID crosses the range size, then that RID is not mapped, thus leading to no access.
Thus 'autorid' module cannot be used in large, complex customer environments.

S-1-5-21-1596698841-3664360031-4006597652-**2345678**

Prefix | Domain ID | **RID**

“Autorid” Module – Challenge 2

Approach

Avoid overflow of large RIDs by implementing a new ID mapping algorithm which extend the capability of “Autorid”.

This new algorithm allows to associate multiple ranges for a single domain in order to handle the unplanned growth of RID in an Active Directory domain.

Challenge 2 - “AutoRID” configuration

Goal: New implementation must be compatible with existing configuration and setups.

smb.conf:

```
idmap config * : backend = autorid
idmap config * : range = 10000000-299999999
idmap config * : rangesize = 1000000
idmap config * : read only = no
```

autorid.tdb

```
key(7) = "CONFIG\00"
data(49) = "minvalue:10000000 rangesize:1000000 maxranges:290"

key(41) = "S-1-5-21-606640139-1729040616-4042376577\00"
data(4) = "\01\00\00\00"

key(2) = "1\00"
data(41) = "S-1-5-21-606640139-1729040616-4042376577\00"

key(11) = "NEXT RANGE\00"
data(4) = "\02\00\00\00"
```

Implementation details for challenge 2

The new “Autorid” algorithm will allocate multiple ranges for a single domain if needed.

1. The module will calculate a MULTIPLIER (= floor(RID/RANGESIZE))

2. Depending upon the MULTIPLIER value, a search key is created.
   If MULTIPLIER is zero, then KEY is the SID itself (remain backward compatibility)
   If the MULTIPLIER is non-zero, then the KEY is SID#MULTIPLIER

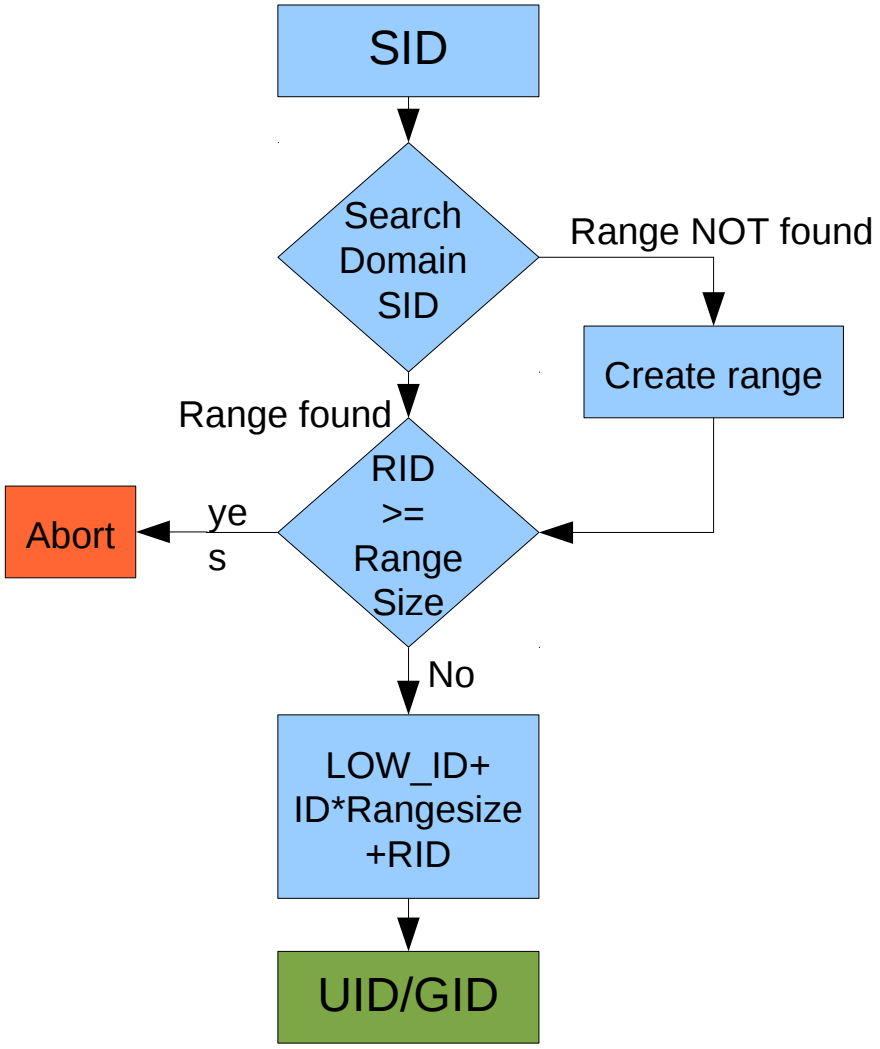
3. The module will try to find the search KEY in the autorid.tdb.
   If the KEY is present, then the domain number is returned
   If the KEY is not present, then an entry (KEY, DOMAIN NUMBER) is added into the autorid.tdb and high watermark is incremented

4. Apply the formula to calculate the UID/GID*
   *UID/GID = Lower ID + (DomainNr*RangeSize) + RID –(Multiplier*RangeSize)*

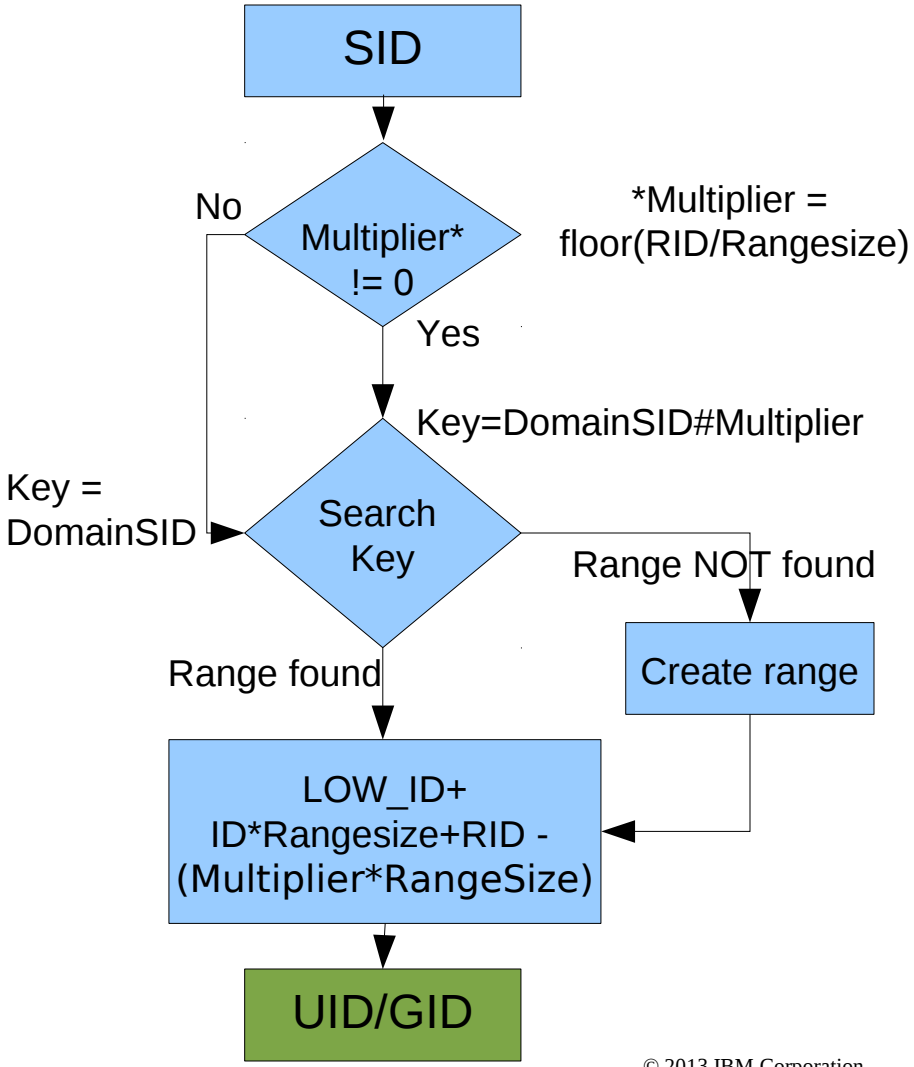
Note: Multiplier greater than 0 (say 2) does not mean that 2 domain ranges are allocated. For eg: if domain number 0 is already allocated and new SID comes up with multiplier as 10, then new SID will get domain number 1 (not 1 to 10). This implies that larger RID (falling in different domain number) might get lower ID depending upon the order of access.

*the exact formula has be optimized by Michael Adam but the main principle remains

Old Implementation



New Implementation



Implementation details for challenge 2 – Sample Flow (1)

1. 'testuser1' from domain XYZ tries to access the system.
   SID of testuser1 is S-1-5-21-3314179063-1446859507-3792912289-1025
   idmap config * : backend = autorid
   idmap config * : range = 10000000-299999999
   idmap config * : rangesize = 1000000

2. autorid module is calculating the multiplier.
   MULTIPLIER is floor(1025 / 1000000) = 0

3. autorid module searches KEY S-1-5-21-3314179063-1446859507-3792912289 in autorid.tdb. As it is not present, an entry is added into autorid.tdb
   (S-1-5-21-3314179063-1446859507-3792912289 , 0)
   Note:0 (domain number) is taken assuming it is the first slot to be allocated

4. UID of testuser1 is calculated
   *UID/GID = Lower ID + (DomainNr*RangeSize) + RID –(Multiplier*RangeSize)*
   UID/GID = 10000000 + 0 * 1000000 + 1025 – 0*1000000
   ***UID/GID* = 10001025**

Implementation details for challenge 2 – Sample Flow (2)

1. 'testuser1' from domain XYZ tries to access the system.
   SID of testuser1 is S-1-5-21-3314179063-1446859507-3792912289-2025000
   idmap config * : backend = autorid
   idmap config * : range = 10000000-299999999
   idmap config * : rangesize = 1000000

2. autorid module is calculating the multiplier.
   MULTIPLIER is floor(2025000 / 1000000) = 2

3. autorid module searches KEY S-1-5-21-3314179063-1446859507-3792912289#1 in autorid.tdb. As it is not present, an entry is added into autorid.tdb
   (S-1-5-21-3314179063-1446859507-3792912289#1, 1)
   Note:domain number 1 is taken assuming that 0th slot is already taken by sample flow #1

4. UID of testuser1 is calculated
   *UID/GID = Lower ID + (DomainNr*RangeSize) + RID –(Multiplier*RangeSize)*
   UID/GID = 10000000 + 1 * 1000000 + 2025000 – 2*1000000
   ***UID/GID* = 11025000**

“Autorid” Module – Challenge 3

Well-known IDs are not mapped at all

Well-known SIDs are special well defined SIDs for dedicated authorities. They don't belong to a domain .
e.g.

S-1-1-0 Everyone
S-1-3-0 Creator Owner
....

The autorid module did not support Well-known SIDs.

Approach

Preallocate Well-known SIDs in the ALLOC_POOL in a predefined order (deterministic).e.g. “Everyone” will always get the first ID in the first range (i.e.1000001)

“Autorid” Module – Challenge 4

BUILT-IN SIDs are not deterministic

Built-in SIDs are part of a special container called 'BUILTIN'. This includes the local groups that exist on a default NT installation
e.g
Administrators (SID: S-1-5-32-544),
Print Operators (SID: S-1-5-32-550),
....
The “Autorid” module does assign the next free ID from the alloc range. It does not honor the RID because the call to winbind asking for a free gid does not include the SID for which the gid is asked for (see wbcAllocateGid)

Approach
Still needs to be resolved ....
e.g introduce a new interface in winbind component which will identify the BUILTIN SID, allocate a separate range for BUILTINS and use the RID component of SID to generate the ID.

Questions ?

Thank you !