User Management with Perforce

Tommy Fad

National Instruments

May 8, 2003

Introduction

- Current system includes
  - Separate processes
  - Adding new users and groups manually
  - Modifying protections manually
  - Documenting your changes?
  - New licensing purchases

Introduction

- Future system provides
  - Automated user management system
  - Central location for information
  - Traceability
  - Friendly user interface for non Perforce users
  - Security

What should I manage?

- Perforce related data
  - User data
  - Group data
  - Protections data
- Other useful information
  - Cost center
  - Phone numbers
  - Specific comments or notes

User data

- Specific Perforce user data
  - Name
  - Full Name
  - E-mail
- Other data
  - Phone number
  - Cost center

User data

UserName	jdoe
Name	John Doe
Email	John.Doe@mydomain.com
PerforceGroups	Group1, Group2, Group3
Servers	Perforce1.mydomain.com, Perforce2.mydomain.com
CostCenter	123
Extension	1234
Department	Software
Comments	Forgets that Depot //Foo is on Perforce2.mydomain.com
ActiveUser	Y

Group data

- Specific Perforce group data
  - Group
  - MaxResults
  - MaxScanRows
  - Subgroups
  - Users

Group data

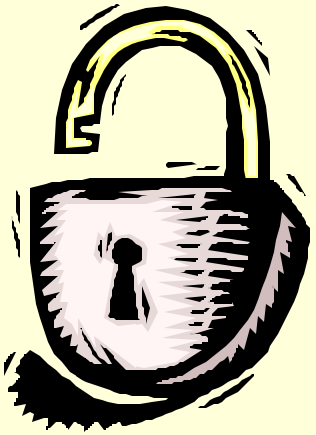
Group	Group1
MaxResults	Unlimited
MaxScanRows	100000
Users	tommyf sevel johnd
DepotAccess	//FooDepot/...
Server	Perforce1.mydomain.com
ActiveGroup	Y

Protections data

Protection	write group WebGroup 123.456.789.123 //WebGroupDepot/...
Active	Y
Comments	This grants write permission for the WebGroup to their depot
BelowLine	Read group AllUsers 123.456.789.123 //WebGroupDepot/...
Server	Perforce1.mydomain.com

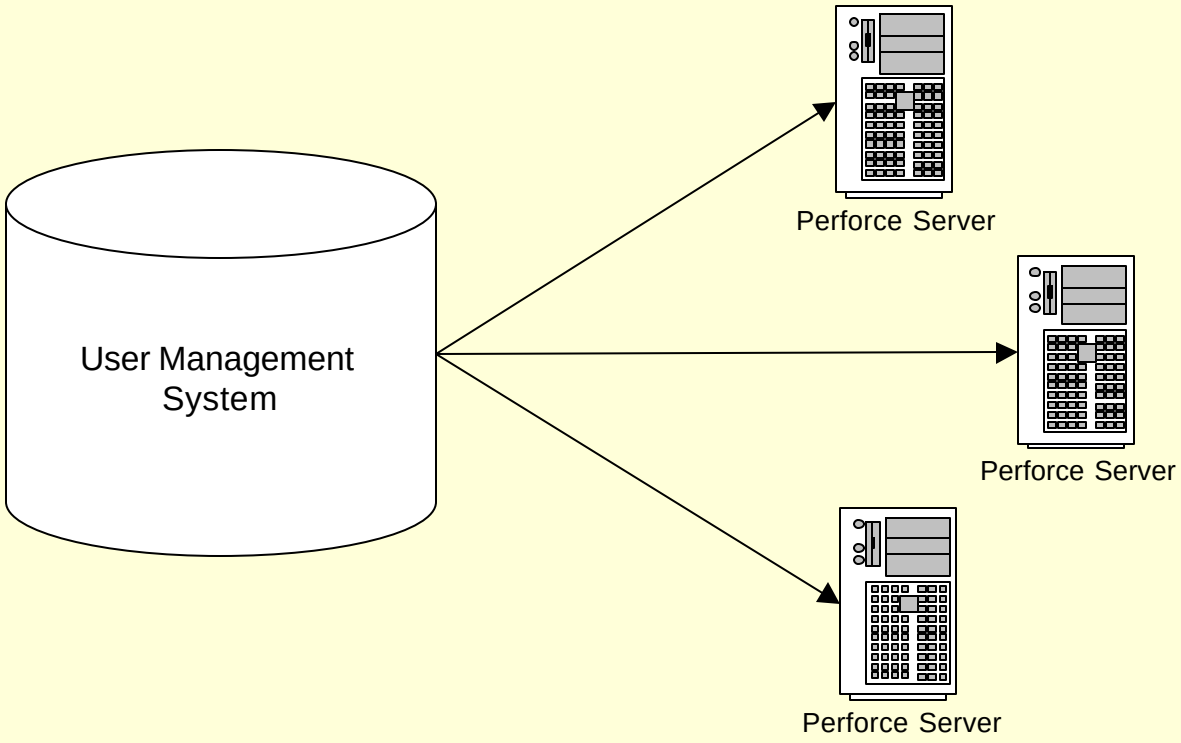
Data considerations

- Keys
  - Useful in a relational database
  - Faster searching
- Security
  - Protections data should be secure
  - Encrypted if necessary
  - User level access to database
- Supplemental information
  - People other than Perforce Admin may use this
  - Comments for users, groups, protect file



Where does the data live?

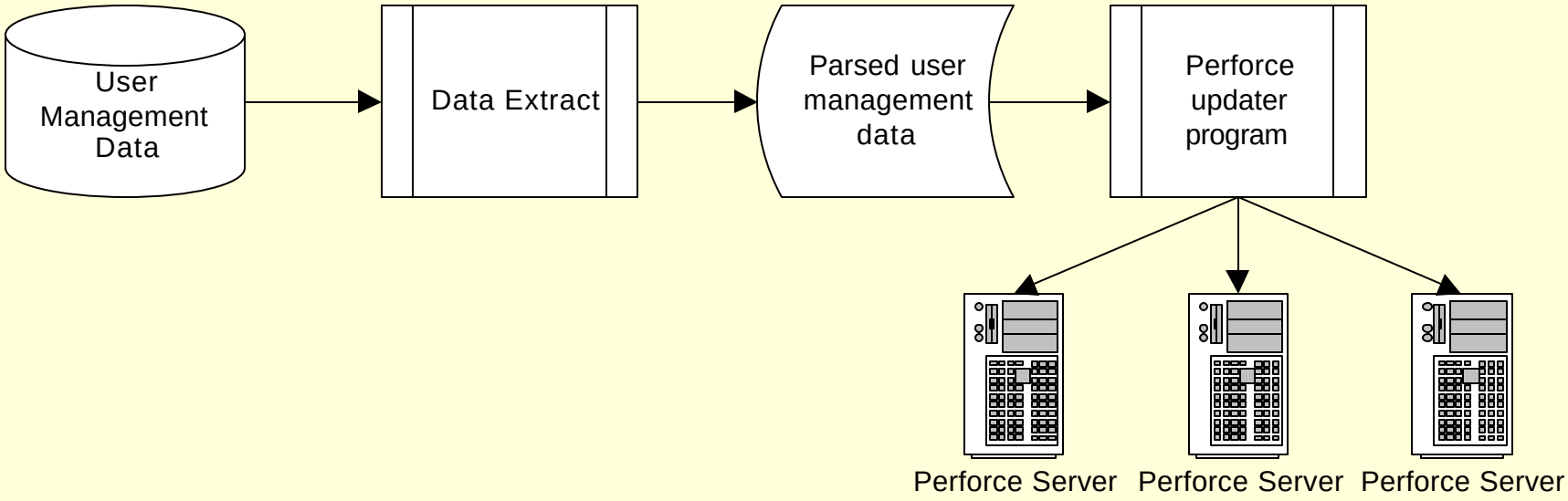
- Central server
  - Database and interface to data
  - Replicates data to Perforce servers



Central server

- User Interface
  - Entry of employee information
  - Reports of employee information
- Interface
  - Used by program/script to update user information to Perforce servers
  - Web page
  - JDBC, Perl Modules, Other APIs

Migrating data to Perforce



Migrating data to Perforce

- Output file
  - Error checking to verify format
  - Store in a secure location
  - Frequency (how often)
- Output file to Perforce
  - Program/Script to push data to Perforce
    - Java, Perl, Python, Ruby, etc.
  - Use checksum to save processing

Migrating data to Perforce

- Checksum (Perl example)

```
my $sha = new SHA;
$sha->reset();

open CHECKSUMLOG, "<$pathtochecksumFile";
my $oldDigest = <CHECKSUMLOG>;
close CHECKSUMLOG;

# Next you will want to read in the data file and compare it's checksum
# to the previous checksum:
open DATAFILE, "<$pathtodataFile" or dieOnErr "Can't open data file $!\n";
while (<DATAFILE>)
{
  $sha->add($_);
}
close DATAFILE;
my $digest = $sha->hexdigest();
if ($digest eq $oldDigest)
{
  exit;
}
```

Migrating data to Perforce

- User data
  - Appropriate data structures
    - Hash, array, user defined object
    - Perl hash (this example)
  - User hash should contain
    - All user information from user management system
    - List of the clients they own
    - List of files that the users have opened
    - Boolean to specify if those files are on a client owned by them or not

Migrating data to Perforce

- User data (contd.)
  - Rule 1
    - Don’t delete userA who owns clientA
    - If userB has files opened using clientA
  - Rule 2
    - Don’t delete userA who is using clientB which is not owned by userA
    - If any other user has files opened using clientB
  - Send alert
    - List the problem why you can’t delete the person

Migrating data to Perforce

- User data (contd.)
  - Useful commands when deleting users
    - `p4 users`
    - `p4 groups`
    - `p4 clients`
    - `p4 opened`
    - `p4 user –fd <user>`
    - `p4 client –fd <client>`
  - Backup clients
    - Keep a copy of delete clients on your server in text format
    - `p4 client –o <client> > client.txt`

Migrating data to Perforce

- User data (contd.)
  - Updated user information
    - Hash within a hash
    - Regular expressions
  - Alerts
    - Any users on Perforce server not mentioned in the user management system output file

Migrating data to Perforce

- Group data
  - Appropriate data structure
    - Hash of groups
  - Updating group data logic
    - Add/remove existing members to existing groups
    - Remove deleted users from groups
    - Add new group and users of the group
  - Alert
    - Groups which are not specified in the output file from the user management system

Migrating data to Perforce

- Protections data
  - Gather all protections data into a hash
  - Updating protections data logic
    - Update based on server
    - Remove inactive protections
    - Add any new protections (utilize ‘below line’ rule)
    - Remove/alert if lines which are not in output file are found in protect file

Migrating data to Perforce

- Protections data (contd.)
  - Print protect to standard output
    - `p4 –o protect`
  - Read protect from standard input
    - `p4 –i protect`

Migrating data carefully

- Error checking
  - Test numerous use cases
- Efficiency
  - Data structures
  - Search routines / sort algorithms
- Alerts
  - Determine when/what to alert on
  - Necessary action to take

What did I gain?

- Central location of data
  - Updates for multiple servers occur a central location
  - Queries of user/group/protections data occurs in one place
- Security
  - Super user commands can occur from a central IP address
  - Encryption of data in database
- Maintenance
  - Updates/changes to user data can be delegated

What did I gain?

- Information
  - Other departments can benefit from user management system information
  - Allocate budget information

Demonstration