VMware: Advanced Perforce Administration Tools
(A Sloth’s Guide)

David Ferguson
Prakash Ranade

---

VMware Perforce (Primary Server) Growth

<table>
<thead>
<tr>
<th></th>
<th>2002</th>
<th>2003</th>
<th>2004</th>
<th>2005</th>
<th>2006</th>
<th>2007</th>
<th>2008</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clients</td>
<td>500</td>
<td>1900</td>
<td>2600</td>
<td>5700</td>
<td>8611</td>
<td>16849</td>
<td>12827</td>
</tr>
<tr>
<td>DB (GB)</td>
<td>8</td>
<td>14</td>
<td>22</td>
<td>67</td>
<td>128</td>
<td>244</td>
<td>201</td>
</tr>
<tr>
<td>Users</td>
<td>90</td>
<td>180</td>
<td>320</td>
<td>840</td>
<td>1089</td>
<td>2686</td>
<td>2663</td>
</tr>
</tbody>
</table>
Perforce Infrastructure Today:

17 Repositories
   Primary/ReadOnly/HotSpare/BuildDedicated (4)
   6 Other large repos (3 originally split off) (>4G)
   7 Smaller repos (VMs)

109,225 Clients
3861 Branches
2753 Users
1,521 GB of Metadata

We’re LAZY

Tools to help:
   BMPS (Branch Management Policy System)
   User Management
   Monitoring
   Performance Analysis
Policy Requests

Who was allowed to check in where?
Enforce that a real user approved the submission
Can I require a bug be added?
Non-trivial comment
Bill locked the tree, He’s on vacation, now what?
Require the user to provide his credit card during Final Freeze

Policy Requests

Initially:
> Provide template for change submission with hints.
> Write API to insert change description into bug
Policy Requests

Initially:
> Provide template for change submission with hints.
> Write API to insert change description into bug

*No biggie, triggers are easy to write*

Then:
> Custom-case a product to require filling in ‘Reviewed by’ field
> Allow ‘auto’ to satisfy Reviewed by when doing an automerge
> Starting Friday, bug number required on branch Foo
Policy Requests

Initially:
> Provide template for change submission with hints.
> Write API to insert change description into bug

*No biggie, triggers are easy to write*

Then:
> Custom-case a product to require filling in ‘Reviewed by’ field
> Allow ‘auto’ to satisfy Reviewed by when doing an automerge
> Starting Friday, bug number required on branch Foo

*Hmmm, better start having triggers do different things for different branches*

--

Policy Requests

Ultimately:
> Require bug numbers on this branch to be one of a provided list
> Only Bill, Jim, and Ted are approved reviewers
> All changes must be approved by the Grand Poobah, and have bug numbers, and be reviewed by Ted
> We’ll need the above for the next two weeks, then go back to the existing stuff
**Policy Requests**

**Ultimately:**
- Require bug numbers on this branch to be one of a provided list
- Only Bill, Jim, and Ted are approved reviewers
- All changes must be approved by the Grand Poobah, and have bug numbers, and be reviewed by Ted
- We’ll need the above for the next two weeks, then go back to the existing stuff

Sheesh, need to write data files so we can keep triggers unchanged and just change what they do externally on the fly

**Policy Changes**

**And of course:**
- I meant SALLY needed to approve. Why does it take hours for you guys to change Ted to Sally?
- I just approved bug 432895, why can’t your tool automatically know I can use it in the template now?
Policy Changes

I can’t drop everything else for you can I?

Policy: Locking

Lock my branch
Policy: Locking

Lock my branch
   Easy-peasy – go modify the protection table

And mine, me too, what about me?
Policy: Locking

Lock my branch
   Easy-peasy – go modify the protection table
And mine, me too, what about me?
   Okay, write the script to do it

Policy: Locking

Lock my branch
   Easy-peasy – go modify the protection table
And mine, me too, what about me?
   Okay, write the script to do it
Lock it for everybody but JimBob
   Special case, just muck the protection table
Policy: Locking

Lock my branch
   Easy-peasy – go modify the protection table
And mine, me too, what about me?
   Okay, write the script to do it
Lock it for everybody but JimBob
   Special case, just muck the protection table
Joe is my superman; Cindy is mine!!
   Oops, not so special. Create data files to make script more robust and change behavior dynamically

Policy: Locking

Can’t you let me lock my own branch?
   I don’t think so … I ain’t giving you unfettered access to the protection table. Guess it’s time to write a web interface
Policy: Locking

Can’t you let me lock my own branch?
   I don’t think so … I ain’t giving you unfettered access to the protection table. Guess it’s time to write a web interface

Why could Susie check in to my locked branch?
   Um, cause you forgot that you told her how to unlock the branch? Guess I add authentication/security to this webpage.

Policy: Locking

I’m on vacation next week. Let Susie lock/unlock my branch.
Branch Management Policy System (BMPS)

**Self-service**
- Branch managers set their own policy

**Ownership**
- Single owner of policy decisions

**Flexible**
- Different managers can choose different policies

**Extensible**
- Architected for expansion

---

**BMPS architecture**

Engineers
BMPS architecture
**BMPS**

**Triggers:**

- **Form-out:**
  - Add fields that need to be provided
  - Insert defaults
- **Form-in**
  - Are all requested fields present?
  - Don’t bother checking the values themselves as they may not yet be known.
- **Change-submit**
  - Query BMPS for needed values and validations
  - Validate accordingly

**BMPS Functionality**

- **Single defined branch owner**
  - Can delegate to multiple branch managers
  - Owner and managers can modify policy – owner responsible
  - Only owner can modify manager list

- **Required Audit Trail**
  - No policy change without provided comment
  - Full policy modification history available
BMPS Functionality

Granular locking
- User exception list
- As fine-grained as desired. We lock to the third tuple:
  - //<depot>/<project>/<branch>

Server-agnostic
- User access by branch/project group
- BMPS determines appropriate server hosting that project group
- Abstracts server (and depot) information away from user

BMPS Functionality

Form field flexibility
- Choose which fields go into submission template
- Determine whether fields can be deleted by user
- Validate field values if desired
  - Pre-defined validators (usernames, digits, non-null)
  - Branch-manager-defined validators (provided text strings)
  - Generated validators (output from arbitrary scripts)

NQWYSIWYG
- Ability to view ‘new’ default template before exiting BMPS
BMPS Functionality

Predefined fields:
- QA Notes
- Reviewed By
- Approved By
- Bug Number
- Doc Notes
- Merge to:
- Mail To
- Tests Run
BMPS

Implemented Jan, 2008

All new branches require it
90% of all maintenance branches using it.

Most people only use locking capabilities 😊
User Management

Problem:
Lost track of which users were on which repository with access to which files

Factors:
Executive fiat: Almost ALL engineers have access to almost ALL files.

Reality:
Our job to protect assets.

User Management

THANK YOU Perforce!!!!

- True-Up licensing
- Duplicate license files

Things to worry about:
Users utilizing multiple repos, reassigned
Contractors
SOX
public depot areas
User Management

No individual users in protection tables

ONE master repo with NOTHING but group and user files

Group/User specifications distributed to all active repos hourly
> Overwrites user modifications (EXCEPT reviews)

Positive authorization: Access only if explicitly provided

Performance Issues

General Grumbling

p4 edit takes forever
Everything is slow…
Convert to something else
Performance Cliff

1. Most users have no clue how slow/fast operations are.
2. They care only if commands are fast enough.
3. Expect syncs to take time.
4. Expect most interactive commands to be quick enough. They do not notice if it takes 1 second instead of .5 seconds.
5. Our job is to keep all operations under the visible threshold.
6. Perforce works very well except for every once in awhile, things fall apart completely:

Cascading operations:

Symptom:
A single ‘p4 opened’ command took 284 seconds to run (4 ½ minutes)
- Waited 284 seconds for a lock on db.locks file
  Db.locks held for 513 seconds (8 ½ min) by a ‘p4 change’ operation
    - Waited 513 seconds for lock on db.rev file
      db.rev held by over 200 invocations of ‘p4 changes’ accumulated in one minute.
P4 changes slow?

Long Waits:

50:--- db.rev
51---- locks wait+held read/write 0ms+36438ms/0ms+0ms
243:--- db.rev
244---- locks wait+held read/write 0ms+25103ms/0ms+0ms
2537:--- db.rev
2538---- locks wait+held read/write 0ms+425ms/0ms+0ms
2670:--- db.rev
2671---- locks wait+held read/write 0ms+3523ms/0ms+0ms

Ah hah! P4 changes is too slow …

Not really –

> Average for p4 changes is .4 seconds over past month.
> Conflicting commands swapped db.rev out
> If db.rev not in memory, need to load
> LONG waits while loading – 20 seconds or more
> How much needs to get in, how long does it take?
  - Block layout,
  - wrinkle factor,
  - query scope
Glacially Slow response

Observations:
Problem occurs occasionally when DB is 1.5 X RAM.
Unusable at 2X RAM
Solution:

*Keep DB smaller than physical RAM*
What we did:

Repository:
- Forked off QA on separate repository
- Forked off 30% of codebase into separate repository
- Duplicate remaining into primary, build-dedicated, and read-only servers
- Stopped adding new projects to big daddy in 2006

Primary Server:
- HP 585 with 256G of RAM
- 500GB of RAMSan Solid State Disk for DB
- DB rebuild every other month (10% shrinkage)
Performance

Other servers:
  Basically, just making sure memory is at least equivalent to DB size.

  NEVER received a single performance complaint on any other server…

Optimizing commands

P4 changes –m1 //depot/...#head  BAD
P4 changes –m1 //depot/...  GOOD

P4 changes –m1 //depot/coredev/v30patch/...#head  GOOD
P4 changes –m2 //depot/coredev/v30patch/...  GOOD
P4 changes –m1 //depot/coredev/v30patch/...  BAD
P4 changes –i –L –m50 //depot/...  SUPER-BAD
**Monitoring**

- Why log parsing? What are we looking for?
- Perforce log file and p4d’s option for log
- Scale of p4 ops and size of Perforce log
- Architecture of the continuous log parser system
- Perforce Dashboard

---

**What are we looking for?**

- To identify current performance problem
- To illustrate overall health of system
- To understand usage characteristics
- To make a policy decision for new system/p4d instance
- Upgrade P4d server and P4 client app version
- Proxy usage/investigation
- Runaway cron jobs/scripts
- To distribute read/write commands on primary/readonly replica
Perforce log file

Perforce log file:
-`v server=3` –`v track=3`
Command tracing flag in p4d startup command.

- The p4d server automatically produces diagnostic output
- Provides useful information such as user commands, IP address, resource usage(CPU, lapse time, database I/O, network I/O)
- `-v server=1` provides command start information only.
- `-v server=2` extends server tracing to include command start and stop.
- `-v server=3` adds a "compute end" message for sync/flush commands.

Scale of p4 ops and size of Perforce log

- More than dozen Perforce servers under watch
- On big daddy > 2 million operations
- 3 of our big servers > 1 million operations
- Primary server generates ~700 MB of Perforce log file per day
- Daily ~ 3 GB of Perforce log data scanned and analyzed
P4 log parser (feeder)

- Python script
- Continuously tail Perforce log file
- Maintains 4 lists (per minute based)
  - pid
  - p4d info such as, user@client ip p4-app and p4-cmd
  - db I/O, locks read/write, wait+held
  - done flag (start=0, complete=1 or 2 based on Db I/O info, script has a check on running pid)
- Notify p4 admin on slow processes which are above set threshold value
- Execute a single INSERT statement to add derived records on central MySQL database

MySQL database

- To optimize database lookup, every day _new_ mysql table is created per Perforce server
  - Perforce_1666_process_20090429
  - Perforce_1666_tableuse_20090429
- To keep MySQL db size under set threshold, we have a policy to drop/delete old table of Perforce log data
- Big single INSERT and small reads
- Use memcache to speed up lookup
Django Platform

- Heavy use of Django's ORM functionality and custom model manager to access daily log table.

```python
class Process(models.Model):
    processKey = models.AutoField(primary_key=True)
    time = models.DateTimeField()
    pid = models.IntegerField(null=True)
    user = models.CharField(max_length=255)
    client = models.CharField(max_length=255)
    ip = models.CharField(max_length=255)
    app = models.CharField(max_length=255)
    args = models.CharField(max_length=1024,null=True)
    lapse = models.DecimalField(max_digits=10, decimal_places=3, null=True)
    uCpu = models.IntegerField(null=True)
    sCpu = models.IntegerField(null=True)
    diskIn = models.IntegerField(null=True)
    diskOut = models.IntegerField(null=True)
    ipcIn = models.IntegerField(null=True)
    ipcOut = models.IntegerField(null=True)
    maxRss = models.IntegerField(null=True)
    pageFaults = models.IntegerField(null=True)
    rpcInGarb = models.IntegerField(null=True)
    rpcOutGarb = models.IntegerField(null=True)
    rpcSizeIn = models.IntegerField(null=True)
    rpcSizeOut = models.IntegerField(null=True)
    data_objects = ProcessManager()

class Tableuse(models.Model):
    processKey = models.ForeignKey(Process, primary_key=True, db_column='processKey')
    tableName = models.CharField(max_length=255, primary_key=True)
    pagesIn = models.IntegerField(null=True)
    pagesOut = models.IntegerField(null=True)
    pagesCached = models.IntegerField(null=True)
    readLocks = models.IntegerField(null=True)
    writeLocks = models.IntegerField(null=True)
    getRows = models.IntegerField(null=True)
    posRows = models.IntegerField(null=True)
    scanRows = models.IntegerField(null=True)
    putRows = models.IntegerField(null=True)
    readWait = models.IntegerField(null=True)
    writeWait = models.IntegerField(null=True)
    data_objects = TableuseManager()
```

Hourly summary

Per hour summary of p4 sync

```
<table>
<thead>
<tr>
<th>Hour</th>
<th>Total No. of p4 sync ends</th>
<th>Total No. of p4 sync ends above threshold</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>10</td>
<td>3</td>
</tr>
<tr>
<td>02</td>
<td>12</td>
<td>6</td>
</tr>
<tr>
<td>03</td>
<td>04</td>
<td>0</td>
</tr>
<tr>
<td>04</td>
<td>05</td>
<td>1</td>
</tr>
<tr>
<td>05</td>
<td>06</td>
<td>2</td>
</tr>
<tr>
<td>06</td>
<td>07</td>
<td>0</td>
</tr>
<tr>
<td>07</td>
<td>08</td>
<td>1</td>
</tr>
<tr>
<td>08</td>
<td>09</td>
<td>0</td>
</tr>
<tr>
<td>09</td>
<td>10</td>
<td>2</td>
</tr>
<tr>
<td>10</td>
<td>11</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>12</td>
<td>0</td>
</tr>
</tbody>
</table>
```
Dashboard widget

Dashboard for

Perforce dashboard

- Shows user summary as,
  - Hourly total # of Perforce command invocations.
  - Per day summary on user’s total Perforce command invocations.
  - Any command whose lapse time above the set threshold.
- Shows IP address summary as,
  - Per hour summary of number of IP/machines connecting.
  - Top ten machines in order of their command invocations.
  - Per day summary of total number of IP address/machine connections.
- Shows Perforce commands based summary as,
  - Per hour summary on type of Perforce command
  - Per day summary on total number of Perforce commands
  - Ratio of Read/Write Perforce commands
- DB/Depot disk size growth over time
- Growth of depot files under Perforce Version control over time
- Growth of Perforce branches, clients, users over time
Perforce Charts

Note – window of blatant abuse of Perforce operations. A single build monitor tool caused jump.
Perforce Charts

What happened when continuous builds moved to build-dedicated server

Perforce Chart

Workload before/after build-dedicated, persistent hunting
Perforce Charts

Teaching users how to improve script efficiency

Questions?