

"Large Perforce Installation Details - Perforce at Adobe"

Perforce Users Conference, April 2005 – Las Vegas

Stanton Stevens

Dave Foglesong

1. Introduction

1.1 About Adobe Systems, Inc.

An edited quote from Adobe Systems marketing information:

“Adobe helps people and businesses communicate better through its world-leading digital imaging, design, and document technology platforms for consumers, creative professionals, and enterprises. Adobe’s products include Photoshop, Acrobat, InDesign, Premiere, and other related software.

Founded in 1982, Adobe today is one of the world's largest software companies, generating revenue in 2004 of US\$1.7 billion. More than 3,700 employees across the world share Adobe's commitment to helping people communicate better. Headquartered in San Jose, California, Adobe is traded on the Nasdaq National Market under the symbol ADBE.”

1.2. The Authors

Stanton Stevens and Dave Foglesong comprise the current “Source Code Management Services” (SCMS) team.

Stanton Stevens

Stanton has been the main Perforce person at Adobe since January 2003, taking over Dave Foglesong’s role as Configuration Management Architect. His prior CM experience includes 2 years as Perforce admin with Dotcast, Inc, and 5 years as a ClearCase admin in various consulting and full-time positions. Before it became obvious to him that CM was the missing piece in most development efforts, Stanton was a C/C++ software engineer involved in medical and broadcast video animation for 10 years.

Stanton’s role at Adobe is to be the central focal point for Perforce issues, maintenance, planning, training, and conversions of acquired code bases to Perforce. He also serves as liaison with Perforce, Inc. for support questions and licensing. He handles Perforce tasks which require logging into the servers, and handles other tasks which the individual Perforce admins for each team escalate to him.

Dave Foglesong

Dave worked for Adobe Systems for almost 15 years doing mostly build and CM related work before leaving the company in 2002. He's now back at Adobe, this time as a

consultant. In his previous stretch at Adobe, Dave designed the build systems for several large applications and was the company's primary Perforce resource, taking the company through the transition to Perforce as the Adobe standard SCM tool.

Dave is currently working on a suite of web tools for company-wide Perforce security reports, and web utilities for Perforce admins to help manage their Perforce instances.

1.3. History of Perforce use at Adobe.

Adobe was founded by two engineers (both John Warnock and Chuck Geschke have PhDs in technical fields) and remains a very engineer-centric company. As a result of this, the various development teams have been allowed to arrange their own development processes. This is good for product development, but bad from an common infrastructure standpoint. As a result of this, for a long time each development team chose and ran their own source control tools.

The variety of tools used spanned the gamut: SourceSafe, Projector, PVCS, home grown tools built on RCS, even ClearCase. In the summer of 1998, our office in Minnesota made the switch from VSS to Perforce. This was a fairly small team (<15 at that time), but it was pretty painless for them. About the same time, Dave was evaluating alternatives to VSS for the InDesign team. This was a much bigger team (40+) and we switched in fall (Aug/Sep) 1998. Largely on the success of these two teams with Perforce, within 12 months nearly every team in the company had switched. Standardization was good, but due to the organizational structure that meant that we now had a hodge-podge of Unix and Windows servers, and each team was maintaining their own sets of license orders with Perforce, leading to many licenses being duplicated.

In February of 2001 Dave moved into the IS group as the company's first Configuration Management Architect. The primary goals were to get all systems onto a standard version and platform (Solaris). Dave left in December 2002, Stanton replaced him in January 2003 completed the move release 2002.2 and significant servers to Solaris that year.

Currently Perforce is used and accepted across the company, with relatively few complaints and general appreciation for the speed, features, and cost. The source code bases of all acquired companies are converted to Perforce right away. It has been a great benefit to the company to use only one kind of SCM tool.

2. Perforce Environment

2.1 Raw numbers

These statistics are as of 2/21/05:

- 11 servers, 9 Sun Solaris, 1 Linux, 1 Windows – 4 largest Sun Solaris servers are in San Jose, 1 each in Seattle, Minnesota, India, Hamburg, Norwich (England), Ottawa (soon 1 more in Dublin)
- 70+ systems/ports - ranging from 1 to 15 per server. Most were upgraded to 2004.2 from 2002.2 this year.
- 32 P4Proxy instances – most in India, others in Minnesota, Ottawa, Germany and several remote locations in California.
- 1900 + unique users, growing about 25/month. Mostly engineering and QE, also writers, marketing, IS.
- 3.0 Terabytes disk space in use for Perforce, 15% is db.* files and checkpoints. Nearly 80% of this is mirrored onto an equivalent amount of storage as “shadow images” used for nightly offline checkpoints, see 2.2.3.
- new changelists per month on average, 2740229 to date.

2.2. Infrastructure

2.2.1 Project Distribution

Most Perforce instances (ports on a server) correspond to a single project or team, though large ones like Acrobat have many different divisions. Due to the original non-centralized adoption of Perforce, there are a few cases of 2 or 3 instances where only one is needed, but for the most part the divisions are logical. The other possible arrangement, one or a few Perforce instances, would be very hard to switch to now, especially with the limitations of the perfmerge tool and associated downtime. Here are the pros and cons of our approach, and more information about how this infrastructure works for us:

Advantages of distributed Perforce instances:

- Each instance has its own Perforce admin(s), and a relatively simple protect table and user base to manage. These admins all have other duties, and work closely enough with the user base to know what is needed and be most helpful.
- The local instance admins know and can easily consult project managers to determine who should or should not have an account and what kind of access is appropriate. Security is tighter and simpler to manage.
- If an individual user locks up a Perforce database with a bad command, only that project is affected.
- It is more difficult to accidentally lock up a small Perforce instance with a command that accesses //... (for instance) than a larger one.
- Checkpoint time is relatively short per instance, downtime for similar maintenance operations is limited to the users for that instance.
- Remote teams can work more easily with a local Perforce instance.

Disadvantages of distributed Perforce instances:

- Heavy use is made of remote depots, which are relatively inefficient and often jam up both the local and remote Perforce instance when browsed.

- Some files are integrated into a number of Perforce instances via remote depots, causing duplicate storage.
- A user may have an account in a number of instances, we have needed to create tools to help manage this.
- Company-wide Perforce initiatives, such as upgrading, standardizing protect tables, depot naming, etc. are difficult.

With several hundred developers in India, as well as large teams in other locations far from Adobe's San Jose headquarters, Perforce instances local to those teams were necessary. Perforce proxies have proved very helpful, however. Since we began to use them in late 2003, they have allowed many development teams to switch from a local server to a San Jose based server and work more easily with San Jose counterparts.

One thing was needed before the proxies could be used widely; a way to automatically clean up the ever-growing cache directories. You can find a Perl script to do this in the Perforce public depot: `//guest/stanton_stevens/cache_clean.pl`. It works well on Unix and Mac proxy servers, and will work on Windows servers that have cygwin or something similar installed. It can be configured to limit a proxy cache to a set size by deleting the least recently accessed files. It will work a bit better when a bug concerning incorrect access dates in Solaris proxy cache files is fixed, there are notes about this with the script, and it may be fixed by the time of the conference in April.

2.2.2 User and Admin Support Structure

The support structure for users and admins has two tiers and an online presence:

A. Local Project Admins

Each of the 70+ Perforce instances has its own Perforce admin(s), none are full-time, they are also developers or otherwise involved in the project. Admins with large teams such as Photoshop are Perforce experts, others know just enough to add and remove users and their access. These admins are the first people that users contact for help with Perforce.

B. IS Source Code Management Services (SCMS)

A local Perforce admin's help resource is SCMS. Usually there is just one full-time Adobe employee in SCMS, currently Stanton Stevens, with the title of Configuration Management Architect. Dave Foglesong is currently working with him as a consultant as he also has in the past. SCMS provides:

- Help with issues the local admin can't handle. This includes advice on branching strategy, integration subtleties, and other specialized Perforce knowledge.
- A link with Perforce support, reproducing issues if possible, contacting support, and following up. This also includes license renewals. Only a few people at Adobe are allowed to contact Perforce support, both as a way to save money on support costs and to make sure that all open issue efforts are coordinated.
- Online documentation on how to install and get started with Perforce, current issues list, security standards, etc.

- Conversion of acquired code bases and training for new Perforce users and admins.
- Performs Perforce tasks that can only be done by logging onto the server, such as trigger installation, upgrades, space usage and backup management, offline obliterates, log file searches, etc.
- 24 hour monitoring and support for jammed or dead Perforce instances, help for teams when the local admin can't be reached.

An interesting aspect of Perforce use in a large corporation is the “red tape” associated with incidents and other downtime. To stop a Perforce instance, not to mention a server, even for a quick re-start, requires filing an RFC (Request For Change) at least a week in advance. This RFC requires approvals by two review groups and a manager before it can be executed, and has to contain a Risk Analysis Assessment, a test plan, a backout plan, text to be posted on the IS events website, links to appropriate team definitions, etc. If a Perforce instance goes down, for instance a checkpoint is started at the wrong time, it is a “P1” incident. These incidents have their own logging requirements and are reviewed by and have to be explained daily to a meeting of IS directors. SCMS is on call 24 hours a day for these incidents. Much of the procedure is driven by the new Sarbannes-Oxley (SOX) requirements of corporate accountability.

C. SCMS Website

We provide the information mentioned above via a website dedicated to Perforce use at Adobe. Though we are in the process of adding a number of tools for users and admins to this site, we currently provide at the site:

- An up-to-date list of all the Perforce servers, ports, instance names, and associated admins at Adobe.
- A tool to find all Perforce accounts across Adobe, and change his/her password on all of them at once.
- A similar tool for finding all Perforce clients, listing information about them, including the mapstate, which can be hard for a user to determine otherwise.
- A report for admins that lists all accounts for a given Perforce instance, with the following information about each account: password set y/n, super user y/n, LDAP (Adobe's main account tracking service) account type, LDAP active y/n, date of last access, number of clients.
- A report for admins listing which Perforce instance have remote depots that point at their Perforce instance, and listing obsolete depot references.
- A tool that reviews a protect table for problems such as allowing automatic account creation, insecure access, references to non-existent groups and users, etc.
- A tool for listing the log entries for currently running processes on a port. This tool will become obsolete when the upgrade of all instances to 2004.2 is complete, since the admins can then use p4 monitor.
- Tools for SCMS only that scan all systems for users, create license reports, etc.

2.2.3 Hardware: Platforms/OS, and Backups

Hardware – Sun Solaris

We have been very pleased with the reliability and performance of our Sun Solaris servers, as well as with how Perforce works on them. We have found them to be an improvement over our former Windows servers for several reasons:

- A single p4d child process can be killed without affecting other processes for the port. On a Windows server there was no way to kill a single thread without restarting the Perforce instance and sometimes rebooting the machine, terminating all ports and processes. Any Unix server has this advantage.
- Since we develop on Windows, Mac, and Unix platforms, the case-sensitivity of a Unix file server is important.
- No need to reboot our servers monthly for security updates. We seldom need to reboot for any reason, a machine can easily be up for a year or two.

Offline checkpoints and backups

Using a combination of Veritas SAN (Storage Area Network) and HDS (Hitachi Data Systems) storage, we are able to make “shadow images” of each Perforce server in less than a minute. About 80% of our Perforce data is shadow imaged. We use these shadow images for:

- Offline checkpoints. By locking the Perforce database files at the time of the shadow image, we can then mount the shadow image volumes on another server and run the checkpoints there. This got rid of our most significant (by far) source of Perforce downtime.
- Backups. These run against the shadow server, so they don't compete for disk access with processes on the live Perforce server.
- Experiments. It is easy to start Perforce instance on the shadow server and try out an upgrade, for instance. It is also a controlled environment to investigate troublesome commands, the speed of large operations such as syncs and integrates, etc. Offline operations that take less than 24 hours can also be done on the shadow server.

3. Looking Ahead

There are some features of Perforce that we are not yet taking full advantage of, and other features that we would like to see or are providing via our own tools. Sometimes we find ourselves developing tools that Perforce is already working on, so we try to coordinate our efforts as much as possible with Perforce support.

This might be a good place to mention that we are very happy with the level of support provided by Perforce, Inc. We usually have several issues open at a time, so Perforce support may not be aware how much we appreciate the help. The Perforce support team responds quickly, thoroughly, and professionally. We have also appreciated the opportunity to make our needs for future features and fixes known, and the changes we have seen in response to the dialog that resulted.

3.1 Managing the 70+ user databases

At Adobe we have tried to minimize the number of different logins and passwords that people need for daily tasks such as email access, scheduling, document access, HR forms access, etc. These tools, Outlook and CorporateTime, for instance, all authenticate users against an LDAP database. It would be great if Perforce could do likewise. The Central Authentication Server feature doesn't work with our structure since group management needs to remain in the hands of the admins of individual instances. And it is still an independent user database with its own authentication.

We have created a tool that lets users find their accounts on all systems and change the password in one operation. We are hoping that more tools are under development at Perforce to help admins manage multiple Perforce instances.

Fortunately, Perforce is good about ensuring that clients and servers of different revision levels can work together. With so many Perforce instances it is impossible to upgrade them all at the same time and it is a big task to coordinate it. Various deadline pressures prevent some teams from upgrading until after a release is done, so it has been vital that remote depot and proxy relationships, for instance, work between revision levels as different as 2004.2 and 2002.2, our most recent jump. We upgrade every 1.5 – 2 years.

3.2 Security

The security improvements in release 2004.2 have been very important to us and are much appreciated. They are vital to our continued use of Perforce. As we work with the admins for each Perforce instance to tighten up access one common problem is difficulty in visually parsing the protect table for medium sized or larger project to determine what access a user actually has. We would love to see a way to list access either by user or area, but don't see how to create this ourselves without writing a protect table logic simulator. Some have tried this and failed, and it is unlikely that we would get all the rules right.

A way to configure the command access for levels like "admin" would be helpful. We'd rather that "admin" level access couldn't start an obliterate, for instance.

Being able to limit access by user "remote" by more than IP number, (port number or account name, for instance) is another security improvement we'd like to see. An IP address can have a number of active Perforce ports.

The "spec" depot type is a great addition. Even better would be if it included the account name that made the change. Being able to know who did what is essential to security.

We are working on infrastructure to support expiration of accounts and passwords, and to support the use of utility accounts by multiple people. For utility accounts, it is very important for us to have an assigned owner who takes responsibility for what is done using the account and for regular password rotation.

Continually tighter security standards in all aspects of software development are expected, partly due to things like the Sarbannes-Oxley corporate accountability rules.

3.3 Distribution of Binaries - Obliterates

Several teams at Adobe found that Perforce was ideal for distribution of binaries created by regular builds, due to a standard interface from all platforms, easy identification of a set of revisions, obvious association with the original code, etc. Unfortunately, this wasted many GB of Perforce storage space keeping files that were only of interest to anyone for a few weeks at most. Because of the difficulty of managing and obliterating these files, the teams had change their processes to use the standard file server distribution method.

Stanton has been making the case with Perforce for more support for binary distribution, including improvements to p4 obliterate, more flexible ranges of revisions kept for +S (tempobj) file types, support for archiving and retrieving file trees, saving binary diffs instead of entire files, etc. You can find the complete write up in the Perforce public depot: [//guest/stanton_stevens/ObliterateLetter.pdf](http://guest/stanton_stevens/ObliterateLetter.pdf).

For even a medium sized Perforce instance, the considerable downtime associated with p4 obliterate is more than we can allow. With teams around the world and automated build processes accessing Perforce instances 24 hours a day, Perforce downtime is hard to schedule if not impossible. So we do offline obliterate; here are the steps:

- On the same server as the instance to be obliterated from, create a second root folder (root2) that contains symbolic links to the depot storage in the live root folder.
- Restore the most recent checkpoint for the instance into root2.
- Restore the live journal in root into root2
- Using the protect table in the live instance, mask off the areas to be obliterated.
- Start a Perforce instance against root2, using another port number. Be sure to enable journaling.
- Create a client spec on the root2 instance that can only see the files to be obliterated, use it to do a test obliterate with the command “p4 obliterate //oblit_spec/... > oblit_out”
- Review oblit_out for many lazy copy undo’s that could make the obliterate pointless. A Perl script to do this is available in the public Perforce depot: [//guest/stanton_stevens/oblit_check.pl](http://guest/stanton_stevens/oblit_check.pl). It will also tell you how much space the obliterate will reclaim/use.
- Edit the oblit client spec if necessary, if the obliterate is still reasonable, then run the obliterate command against the root2 instance with the -y option.
- Restore the journal file from root2 into the live root Perforce instance.
- Append the root2 journal file contents to the live root journal file.

Changes to obliterate functionality in upcoming Perforce releases will hopefully make this process unnecessary.

3.4 Integration With Other Tools

We have just begun to explore the very promising integration of Perforce with defect tracking tools, and have high hopes for it.

As already mentioned, a user authentication method in Perforce that reaches out to LDAP or Active Directory would be very helpful.

The way Perforce works with the Microsoft .NET IDE is also under review at the moment. Most of our developers have disabled the SCC connection to Perforce or tolerate and ignore the many pop-ups associated without understanding them. None have successfully used the 2004.2 client with .NET.

4. Conclusion

The SCMS team and Adobe in general is happy with Perforce, due to the following benefits:

- The money saved over other large corporation solutions such as ClearCase.
- Administration overhead is much less than with other systems.
- Good performance and reliability.
- Very good support, and the consideration shown for our needs.
- Users are able to come up to speed quickly, and solve most problems themselves.