

Comparison:

Perforce and IBM Rational ClearCase

Perforce 2008.1 and ClearCase 7.002 with New Appendix: 2011 Update - What's Changed in Perforce and ClearCase

This document compares Perforce (version 2008.1) with IBM Rational ClearCase (version 7.002) with qualitative updates between Perforce 2011 and ClearCase 8.0. Read this comparison to:

- See what has been updated in 2011
- Understand Perforce and ClearCase's major feature differences
- See head-to-head metrics for operations like branching, merging, check-ins, and checkouts
- Get a general comparison of the effects of scaling on both systems

Table of Contents

Executive Summary	1
Qualitative Differences	2
Installation	2
User Setup	2
Distributed Development	3
Defect Tracking	3
Scalability and Performance	3
Cost of Ownership	3
System Administration	4
Atomic Transactions	4
Build Avoidance	4
Performance Analysis	5
Benchmark Setup	5
Test Data	5
Test Procedures	5
Test Issues	6
Benchmark Results	6
Test Analysis	10
Conclusions	10
Perforce Is Faster Than ClearCase	10
Perforce Costs Less Than ClearCase	10
Appendix A: 2011 Update – What’s Changed in Perforce and ClearCase	11
Executive Summary	11
Deployment and Administration	11
Distributed and Offline Support	11
Scalability	11
Performance	11
Administration	11
Binary File Management	12
Extensibility	12
Branching and Usage	12
Branching and Release Management	12
Ease of Use	12
Atomic Transactions	12
Build Acceleration	12
Learn More	13
Evaluating Perforce	13
Scheduling a Demo of Perforce	13
Migrating to Perforce	13

Executive Summary

This document compares Perforce (version 2008.1) with IBM Rational ClearCase (version 7.002) and contrasts their significant differences. Included are qualitative aspects such as distributed development

and administration, as well as quantitative metrics of the time it takes to perform common software version management operations. See Appendix A: 2011 Update to see what has changed since this comparison was first written in 2008.

Overview

Attribute	ClearCase	Perforce
Installation	ClearCase requires significant preinstall planning and an investment in hardware and infrastructure.	Perforce installs in minutes on existing hardware.
User Setup	ClearCase relies on the operating system for authentication. Heterogeneous environments require special considerations.	Perforce's built-in server-based security works uniformly across all supported platforms.
Distributed Development	Multisite replication requires the purchase of a separate product, ClearCase MultiSite, which requires further administration and additional user processes. Real-time replication might not be practical.	The Perforce Proxy is a caching solution for remote users with minimal administrative overhead at no extra cost.
Defect Tracking	Requires the purchase of a separate product, IBM Rational ClearQuest. Also integrates with popular defect-tracking systems.	Perforce includes a built-in defect tracking system, and also integrates with popular defect tracking systems. Perforce provides extensive support for building defect tracking integrations via the Perforce Defect Tracking Gateway (P4DTG).
Scalability and Performance	ClearCase's underlying architecture and MVFS can slow response times even for non-version-management-related activity.	Perforce is an order of magnitude faster for all operations.
Cost of Ownership	ClearCase can be expensive due to licensing, infrastructure, training requirements, and administration complexity.	Perforce has significantly lower pricing, with simple deployment and administration.
System Administration	The complexity of the system requires one full-time administrator for every 25 users.	One part-time administrator can support hundreds of users.
Atomic Transactions	Atomic transactions are not natively supported. UCM, a layered product available for purchase separately, runs on top of ClearCase and provides partial support.	Atomic transactions are natively supported via changelists.
Build Avoidance	The winkin feature of dynamic views saves compile time but requires using the MVFS file system, which has associated performance penalties. Requires "clearmake," the ClearCase build tool.	Perforce has no special requirements for build tools or file systems.

Qualitative Differences

Installation

Perforce's straightforward installation procedure completes in minutes. Setting up a Perforce server merely requires specifying a host name, assigning a port number, and selecting a storage directory where versioned files will be kept. The server imposes light demands on hardware; in fact, the CPU load is usually in single-digit percentages. The major consideration is allocating sufficient disk space with a fast I/O subsystem to house current and future sets of versioned files.

In addition, a single instance of a Perforce server can support thousands of concurrent users, regardless of the users' location on the network. If network bandwidth becomes saturated or is limited, deploying the Perforce Proxy, a local caching server, at remote sites is an option. The Perforce Proxy, like all Perforce product components, is included in the license fee.

Perforce uses the native OS file system for storing versioned files on both servers and clients, with no dependency on proprietary technologies. The Perforce Server uses TCP/IP for all its communication with Perforce clients across all supported platforms so users can work in mixed-platform environments. Perforce also is supported on a larger number of platforms than ClearCase.

By contrast, the process of installing and setting up ClearCase is involved and requires a planning phase to configure server storage, security, and the network. A typical ClearCase installation consists of the following server processes:

- VOB Server
- View Server
- License Server
- Registry Server
- Release Server

The VOB Server and View Server processes are CPU-intensive. Most ClearCase installations therefore require a minimum of two server machines: one machine dedicated to the VOB Server process and another for the View Server. The remaining License, Registry, and Release Servers are assigned to whichever one of these machines is less loaded. (IBM typically recommends that these remaining processes be installed on the machine hosting the VOB Server.) License Servers, though not particularly CPU-intensive, need to be readily accessible to users. Consequently, the License Server often ends up being assigned to a dedicated machine. As the number of

projects and concurrent users grows, performance issues often demand multiple VOB Servers and, along with them, multiple machines to host them.

ClearCase uses a proprietary file system, called a multiversion file system (MVFS), to support some of its features, such as dynamic views. Running MVFS results in network and processor overhead to operate effectively, which potentially impacts simple client actions, such as determining file attributes or listing the contents of a directory. If MVFS is unable to communicate with an appropriate License Server, VOB Server, or View Server, the system is rendered inoperable.

To be deployed in heterogeneous environments (for example, an environment with a Unix server and Windows clients), ClearCase depends on additional products such as Samba.

User Setup

The Perforce Server implements its own security, which is consistent across all supported platforms. True to its client/server architecture, user permissions are defined and stored on the server and don't rely on the underlying operating system. The administrative effort involved for user setup and maintenance is minimal.

By contrast, ClearCase doesn't manage its own authentication scheme and instead uses the native operating system's authentication system. This approach causes problems in heterogeneous environments. For example, in a Unix environment, all user and group relations must be identical for all Unix clients and servers.

Furthermore, in a mixed environment of Unix and Windows, there are the following special considerations:

- Users must have both Unix and Windows accounts.
- When using Samba, users must use the same passwords.
- A special run-as-server domain account is used on Windows for credential mapping on Unix.
- User and group names must match exactly between Unix and Windows.

Removing a user also poses special problems with ClearCase. If a Unix user is removed from the system, any ClearCase history for that user shows a numeric account ID rather than the account name.

Distributed Development

Perforce's distributed architecture bypasses the intricacies of replication by using a central repository that's supported by the Perforce Proxy at each remote location.

The Perforce Proxy caches and serves files to users at remote locations, thereby reducing traffic across slower WAN links. All users, local or remote, connect to the same central depot and access the same files. This approach removes the necessity of replicating branched codelines for remote users and the additional overhead of merging codelines back together.

The Perforce Proxy requires little administration.

By contrast, to implement distributed development in ClearCase, ClearCase MultiSite must be purchased at an additional cost. MultiSite is a replicated solution, where each remote site has a local repository for local access, known as a replica. A replica can be synchronized on a regular basis through automated processes. IBM recommends mirroring the entire repository at each remote site, setting up separate branches for each remote team, and subsequently merging the changes back into the parent branch.

Furthermore, real-time data replication requires careful implementation and administration. Any disruption during updates between nodes can introduce inconsistencies. Updates must be frequent to ensure a consistent view for all users. The requirement that all sites have high availability often results in higher costs for hardware and system administration.

Instead of attempting real-time replication, ClearCase MultiSite is usually implemented with batch replication. As a consequence, ClearCase MultiSite cannot provide the most current project status to all team members.

Defect Tracking

Perforce provides a defect-tracking system called "jobs." A job typically represents an enhancement request or a bug to be fixed. The set of data fields associated with a job is customizable, and typically includes fields such as "description," "priority," "severity," and "status."

Perforce also integrates with leading defect tracking systems, such as Atlassian JIRA, Mercury Quality Center, TechExcel DevTrack, and Bugzilla.

In addition to existing integrations, the Perforce Defect Tracking Gateway (P4DTG), available on both Windows and Linux, provides a GUI interface to map data fields from defect trackers to the Perforce jobs system. P4DTG provides a relatively easy way to

develop simple one-way, or sophisticated two-way data replication/mirroring integrations between Perforce and defect tracking systems.

ClearCase does not provide defect tracking functionality out of the box. You can add this functionality to ClearCase by purchasing and incorporating a defect-tracking system such as Rational ClearQuest for an additional cost. Other defect tracking systems, such as TechExcel DevTrack, also offer ClearCase integrations.

Integrating ClearCase with a third-party defect tracking system usually involves either trigger development or, more commonly, a rudimentary check-in comment-scanning system. Check-in comments are scanned for defect identifiers, and the check-in information is pulled into the defect-tracking system.

With no way to represent defect tracking fields within ClearCase, and no integration support tools, custom integrations must be developed for systems that are not already integrated with ClearCase. The most common types of integrations are simple one-way integrations that scan checkin comments. Such integrations are not particularly robust, because they rely on humans to use standardized check-in comments. Also, defect tracking integrations with ClearCase tend to suffer performance problems due to slow interaction with ClearCase.

Scalability and Performance

Perforce is an order of magnitude faster than ClearCase for most operations while ClearCase's underlying architecture and MVFS can slow response times even for non-version-management-related activity. See the benchmark results in "Performance Analysis."

Cost of Ownership

Perforce is cost-effective; ClearCase is far more expensive than Perforce out of the box. The following factors contribute to ClearCase's higher cost of ownership:

Licensing: Perforce offers cumulative discounts through a tiered pricing model. Pricing starts at \$900 per named user and includes all Perforce product components on all platforms. Telephone and email support, as well as software upgrades, are also included for the first year.

By contrast, the price for ClearCase starts at \$4,380 for a floating license, which includes 12 months of product maintenance. In addition, if you want the functionality that's included in Perforce out of the box, you'll have to pay extra for components such as MultiSite, ClearQuest, and UCM.

Administration: With Perforce, a part-time administrator is usually capable of supporting several hundred users. IBM recommends at least one full-time administrator for every 25 ClearCase users.

Hardware: ClearCase requires more expensive server hardware than Perforce does for the same performance, whether using dynamic or snapshot views. When using dynamic views, network traffic between the developer machines and the servers is greatly increased, and developers' machines must be upgraded to offset the CPU load created by View Servers.

System Administration

Because Perforce uses standard file systems, network features, and a single server, the only administration required is backup and other maintenance activities.

When deploying ClearCase, the proprietary file system, multiple servers, and the dependence on network operation adds to the training, maintenance, and diagnostic responsibilities of system administrators. This overhead is in addition to backups and other maintenance activities required to support ClearCase.

Both Perforce and ClearCase provide comprehensive, well-documented backup and recovery solutions. Regardless, the typical multiple server deployments associated with ClearCase make disaster-recovery planning and execution more complex.

IBM recommends at least one full-time administrator for every 25 users. With Perforce, a part-time administrator is usually capable of supporting several hundred users.

Atomic Transactions

Perforce organizes the changes made to multiple files into units of work called changelists. Typically, changelists represent features or bug fixes that are implemented by modifying multiple files. Because changelists are atomic, they ensure the integrity of each check-in and avoid the corruption introduced by partial file submissions. If any file in a changelist cannot be checked in for some reason (such as an unresolved conflict), the entire submission is rolled back.

Therefore, Perforce guarantees that whenever a changelist is submitted, the state of the entire system before and after the changelist varies only by the set of changes (adds, deletes, and edits) in the changelist.

When users read the history of a file, all the other files that changed (along with any changes to the file being examined) can be seen because each change is connected to a changelist. This approach enables changes to be

understood in a larger context. When viewing the historical list of changelists, all the changes on the entire server, or one project, can be seen without searching every directory.

By contrast, ClearCase lacks native support for atomic check-ins: one file at a time is checked in. To achieve atomicity of submissions, ClearCase UCM, a layered product that runs on top of ClearCase, must be purchased for an additional fee. UCM simulates atomic transactions while delivering code to parent branches, which leaves users with the choice of risking potential inconsistencies during concurrent check-ins to shared branches or creating branches for even the smallest tasks.

Build Avoidance

Perforce has no special requirements for build tools or file systems.

One of the best-known features in ClearCase is build avoidance, otherwise known as winkins. However, winkins are supported only for dynamic views, because they require MVFS. When ClearCase's "clearmake" or "clearaudit" (its versions of the "make" utility and the command shell) are run, they keep track of which versions of files were read and which were written. ClearCase stores the written files as derived objects, keeping track of what command was run and which files were read.

Later, possibly in another user's view, when the same command is run, "clearmake" or "clearaudit" scans its list of derived objects, looking for one built with the same command and file versions. If it finds one, it will wink in the file, placing it there without actually running the compile command. This approach enables developers to open a new view (with only source files and no object files) and get a full set of compiled objects without doing the compilation themselves.

When ClearCase was first released, the winkin capability represented a tremendous time savings. However, this time savings has been largely eroded because of Moore's law. Hard drive data transfer rates simply have not grown at the same rate as CPU speeds and RAM access rates. Build avoidance is still faster than actual builds; you can compare wink in times to compilation times in the benchmark results. However, the state of the art is nearing the point where performing a database query to wink in the right object file is more expensive than recompiling the sources. This is commonly the case for modern object-oriented software products that tend to be modular rather than monolithic.

When looking at the overall performance impact, consider how much MVFS is slowing down the rest of the development process, even non-version-management-related activities such as file edits and incremental compilation.

Performance Analysis

Benchmark Setup

The following benchmark tests were performed on two Linux computers named `ccserver` and `ccclient`. They were the only two computers on the switch end of a router.

The `ccserver` machine was a rack-mounted server with two Xeon 5420 processors each running at 2.5 GHz, 4GB of RAM, and a 5-disk SAS 146GB RAID 5 array.

The `ccclient` machine was a Dell 260 laptop with an Intel Core2 Duo running at 1.66 GHz, 1GB of RAM, with a 120GB hard drive.

Both machines were running Red Hat Enterprise Linux 5, update 1. The kernel was 2.6.18-53.el5. They were running Perforce 2008.1/158777 and ClearCase 7.002-IFIX02.

The Perforce Server was also on `ccserver`, and the Perforce clients were on `ccclient`. The ClearCase VOB storage and Perforce Depot storage were on the same file system on `ccserver`. View storage, the snapshot views themselves, and the Perforce client roots were likewise on the same file system and spindle on `ccclient`.

ClearCase had a VOB Server and a ClearCase registry server on `ccserver`, and this test setup had its own region. The `ccclient` machine could see only this region, so it could access only the single VOB on `ccserver`. All five clients (`dynamic/mainline`, `dynamic/branch`, `dynamic/third`, `snapshot/mainline`, `snapshot/branch`) were on `ccclient`, which means there were two views served on `ccclient`.

Test Data

Three projects were tested:

Large: The large project contains the video files (no audio) to the open source movie Elephants Dream, rendered frame-by-frame as individual `.png` files. This source is compiled into a single, silent `.avi` movie using the free `ffmpeg` program. The sources are about 3.3GB. Many of today's large products, especially in the media and computer gaming industries, use their source control systems to handle large binary files such as graphics,

sound, and animations. This test data simulates a large multimedia project.

Medium: The medium project contains eight copies of the Apache HTTP server. The makefile simply cycles through eight subdirectories and runs "make" on them. The source was configured for this platform with the `./configure` command before being submitted to the software version management system, so the build times listed do not include the configuration time.

For each copy, when handling the ClearCase dynamic views, the file `./srclib/pcre/config.h` was renamed through ClearCase and then copied back to its original name as a view-private file. The build process for Apache deletes and rebuilds this file, which the MVFS does not allow for its VOB files. Neither Perforce nor ClearCase snapshot views had this issue, so `config.h` was left alone in those two circumstances.

Small: The small project is one copy of the Apache HTTP server. The copy was prepared like the eight copies in the medium project, including the `config.h` workaround.

Test Procedures

For each software version management system and every data size, the tests were run in the following order:

Initial import: Copying the data from the file system into the client or view, and submitting those changes to the Perforce Depot or ClearCase VOB. In Perforce, a number of small `p4` add commands are used (one per file), followed by one `p4` submit command. For ClearCase, a single `clearimport` command is used.

First build: Building the project for the first time with GNU Make (Perforce or ClearCase snapshot view) or Clearmake in GNU emulation mode (ClearCase dynamic view). Because the `clearmake` command's main advantage is its build avoidance (`winkins`), which are available only for dynamic views, snapshot views use GNU make.

Incremental (null) rebuild: Running the `make` command used in the first build immediately after the first build without changing anything. This action measures the time required for GNU make or `clearmake` to determine that all files are up-to-date.

Winkin from View Server (ClearCase dynamic views only): Running a build in a second view that is identical to the first. This task makes ClearCase wink in files from the first view rather than recompile them. Typically, this happens when a user tries to recompile a system from scratch (perhaps with a new view). The usefulness of the

winkin, or build avoidance feature, is its speed of builds. These numbers should be compared with the first build times for both ClearCase snapshot views and Perforce clients.

Winkin from VOB (ClearCase dynamic views only):

Running a third build in a third view identical to the first. The “winkin from view” puts the derived objects into the VOB, so the original view does not need to be consulted. Otherwise, this task is the same as winkin from the View. Compare this task against the first build times for both ClearCase snapshot views and Perforce clients.

Populating the workspace: After clearing all files from the Perforce client space or ClearCase snapshot view, telling the software version management system to copy all files within the depot into the client space. This task is not applicable (effectively zero) for ClearCase dynamic views, which present the files as a file system. In Perforce, this task is a single `p4 sync -force` command. In ClearCase, this is a single `cleartool update -force` command.

Incremental (null) update: Telling the software version management system to copy all files within the depot into the client space right after populating the workspace. This measures the time it takes for the software version management system to calculate that everything is up-to-date. This task does not apply to ClearCase dynamic views.

Branching for edit: This task omits creating the branch but includes making it visible to the user. We assume that ClearCase views can be created instantly. For Perforce, this figure will be the time it takes to run a `p4 integrate` command using the branch specification (Perforce can show multiple branches within the same workspace). For ClearCase snapshot views, this figure will be the time it takes to populate the new view.

Adding files: Adding and checking in 100 identical small files to the top-level directory in the newly created branch.

Editing files: Checking out a specified number of files (50 for the small data, 100 for the medium, and 200 for the large) in the new branch, having the contents of a system file (`/etc/group`) copied there, and checking them back in. This changed code will fail to compile, which is why compilation tests were run earlier.

Merging back to mainline: After doing the add and edit benchmarks on the branch, this figure is the time taken to merge these changes back into the mainline, autoreresolve any conflicts, and check the merge in. In Perforce, this task will be a `p4 integrate` command

using the branch spec in reverse, followed by an automated `p4 resolve` command and a `p4 submit` command.

In ClearCase, this task is done with `cleartool merge` followed by a recursive `cleartool checkin` command.

Reading all files: Concatenating every file in the mainline to `/dev/null` to force the OS to read every byte of every file, to measure reading speed. For Perforce and ClearCase snapshot views, the results should be similar, because they depend on file system access speeds and do not use software version management directives at all. This tests the reading performance of ClearCase MVFS (from ClearCase dynamic views) versus the local file system.

Deleting files: With a `cleartool rmname` or a `p4 delete` command, removing all files (but not directories) from the latest versions. For both Perforce and ClearCase, the system keeps copies of the file for historical purposes. Removing the file from the software version management system entirely (with `cleartool rmelem` or `p4 obliterate`) erases data and is not routinely done by developers (if at all). Thus, this report does not benchmark how fast each system can erase its own histories.

Test Issues

In this test, selection was limited to open source software to allow for the independent verification of the results. As described in the “Test Data” section, ClearCase dynamic views need some tweaking when dealing with open source software. Open source software using the `./configure` or `autoconf` functionality regularly edits its own sources or makefiles during the compile process, which MVFS prohibits.

To remedy this issue, When preparing commercial software, on-the-fly dependency generation can be avoided. Similar configuration issues with Perforce and ClearCase snapshot views were dealt with by making certain files writable with the Linux `chmod` command.

Benchmark Results

In the following tables (see Tables 1, 2, and 3), the values are in “wall clock” seconds, specifically, the total time returned by the Linux `time` command. Times are rounded to the nearest whole second. All tests were run at least three times and averaged to produce the results.

Table 1: Large Project (all time in seconds)

Procedure	ClearCase Dynamic View	ClearCase Snapshot View	Perforce
Initial import	42,524	60,103	1,281
First build	740	758	890
Incremental (null) rebuild	19	<1	<1
Wink in from view	30	N/A	N/A
Wink in from VOB	19	N/A	N/A
Populating the workspace	N/A	602	411
Branching for edit	N/A	553	323
Adding files	142	149	17
Editing 200 files	260	252	2
Merging	338	308	7
Reading all files	482	128	405
Deleting files	5,752	9,619	167

Table 2: Medium Project (all time in seconds)

Procedure	ClearCase Dynamic View	ClearCase Snapshot View	Perforce
Initial import	20,794	61,364	1,542
First build	3,011	1,050	1,036
Incremental (null) rebuild	33	3	3
Wink in from View	2962	N/A	N/A
Wink in from VOB	364	N/A	N/A
Populating the workspace	N/A	185	26
Incremental (null) update	N/A	36	<1
Branching for edit	N/A	170	34
Adding files	129	146	23
Editing 100 files	85	107	7
Merging	219	244	1
Reading all files	98	1	1
Deleting files	5,162	14,764	14

Table 3: Small Project (all time in seconds)

Procedure	ClearCase Dynamic View	ClearCase Snapshot View	Perforce
Initial import	2,376	7,157	182
First build	373	98	129
Incremental (null) rebuild	5	<1	<1
Wink in from View	350	N/A	N/A
Wink in from VOB	40	N/A	N/A
Populating the workspace	N/A	21	3
Incremental (null) update	N/A	5	<1
Branching for edit	N/A	26	6
Adding files	110	123	8
Editing 50 files	40	46	4
Merging back into mainline	150	169	1
Reading all files	17	<1	<1
Deleting files	602	1,541	3

Test Analysis

One obvious question about the previous test data is, “Why should speed matter?” After all, purchasing decisions for a software version management system are based more on features and reliability than benchmark speeds. What difference can an extra 10% make to the bottom line?

The important consideration here is that many of the differences are not fractional: one software version management system may take several times as much time to perform an operation than another. In some cases, the difference is more than an order of magnitude for common software version management operations. This level of delay can be significant enough to annoy developers, causing a costly distraction.

The performance situation gets worse when using dynamic views with ClearCase. Because dynamic views turn every file access into a software version management operation, check-ins, compilers, and editors are slowed down. This level of slowdown tempts developers to work outside the software version management system sandbox to get an adequate response time, negating most of the benefits of having a software version management system in the first place.

No benchmark involving ClearCase with MVFS is complete without discussing wintins. The data shows that a wintin from the VOB Server (which occurs the third time the same object is compiled) is much faster than a Perforce compilation from scratch. This speed gives you an advantage after opening a new view and “recompiling the world.” However, this advantage applies only at that time. Subsequent compilations are normally performed by dependency tracking systems such as the “make” program, so only the code affected by the developer’s changes is recompiled in either case.

Moreover, the wintin advantage can be obtained in Perforce by breaking up a project into subassemblies such as libraries and checking the compiled subassemblies back into the software version management system. Using this approach, a developer need check out only the source code for the subassembly being worked on and thus avoid “recompiling the world.”

The compilation results for the large project are a remarkable departure from the other two projects. It must be noted that the large project is a multimedia project rather than a software project. The compilation is one step, reading more than 10,000 files to generate 1

file. No intermediate results (such as object files) are ever written.

Here, ClearCase wintins enjoy a tremendous advantage over the other two systems because there is only one file to wintin in. But this savings is only on the first recompilation for a user and can be duplicated in Perforce by checking large compiled objects back into the source control.

Conclusions

Perforce Is Faster Than ClearCase

Perforce is faster than ClearCase for most common software version management operations. ClearCase with snapshot views is several times slower when running software version management commands. ClearCase with dynamic views (MVFS) is also slower than Perforce in running software version management commands, and becomes much slower than either Perforce or snapshot views when performing simple file operations.

The wintin ability of MVFS accelerates certain compilations, but not nearly enough to counter the drag it induces on the rest of the development process. Especially with MVFS, the performance difference is enough to impact the performance of the development team well beyond the time spent running software version management commands.

Perforce Costs Less Than ClearCase

Perforce costs less than ClearCase for licensing, hardware, and training. Perforce also requires fewer administrative resources and saves development time by being simpler and less obtrusive to the development process than ClearCase. Perforce lets developers spend more time creating software and less time wrestling with the system.

If a ClearCase installation is administered by developers (often the case with smaller development efforts), switching to Perforce will free up some of those developers to get back to developing product. Indeed, in a small operation, Perforce administration is a part-time position and that admin can double as a sysadmin or developer.

Appendix A: 2011 Update – What’s Changed in Perforce and ClearCase

This section details changes to Perforce and IBM Rational ClearCase since this comparison paper was first published. This update will consider the qualitative differences between the latest versions of both products: Perforce version 2011.1 and ClearCase version 8.0.

Executive Summary

Since 2008, Perforce has introduced several innovative new features, and expanded its arsenal of tools for managing distributed development and managing complex projects. In the same time span, ClearCase appears to have been superseded largely by the new Rational Team Concert platform. ClearCase introduced a new set of clients backed by a WebSphere application server and a Java API, and has deepened the integration with other IBM products. However, little in the core ClearCase platform has changed. As a result, the major conclusions of the 2008 paper, which highlighted Perforce’s advantages in speed, flexibility, and cost, are still valid.

Deployment and Administration

Distributed and Offline Support

Perforce’s support for distributed and offline work has improved tremendously since the 2008.1 release. Perforce’s deployment architecture now includes fully replicated servers, improved remote depots, and P4Sandbox. P4Sandbox is a significant innovation that bridges the gap between distributed and centralized software version management. It allows for fully independent private or local work while still maintaining a strong relationship to the central server.

ClearCase has improved support for distributed work by introducing the ClearCase Remote Client (CCRC). CCRC, commonly used inside an IDE like Eclipse, uses a special form of snapshot view called a web view, which precludes the use of clearmake and other features that depend on dynamic views. The ClearTeam Explorer client provides a unified GUI for working with dynamic views or web views.

Using CCRC requires the use of an intermediate WebSphere application server that includes a CM Server or CCRC WAN server. CCRC communicates with the

application server over HTTP. The CM or CCRC WAN servers then communicate with a traditional ClearCase client and server via RPC.

ClearCase still offers only limited support for offline work and no support for private branching outside the context of the central environment.

To sum up, Perforce has developed an innovative and advanced new deployment architecture that is considerably easier to deploy and maintain than ClearCase.

Scalability

Perforce’s deployment architecture is scaling out in two dimensions. First, the fully replicated servers continue to evolve to serve build farms and other automated processes. Second, the replicated servers and P4Sandbox offer improved local and offline performance for users at any location. These tools continue to evolve to shift more work away from the central server.

ClearCase, in version 7.1, introduced a new architectural component, WebSphere application servers with CM or CCRC WAN applications. These tools provide improved performance for non-LAN scenarios. However, MultiSite remains the primary means of scaling a ClearCase installation.

Performance

Perforce has seen significant performance improvements since the 2008.1 release, including a faster integration engine and improved concurrency support.

ClearCase has focused on performance improvements for UCM and operations over a WAN using a web view client, built on the new CM Server architecture. There is no indication that performance of the core platform has improved.

Administration

Perforce has continued to evolve its administration tools, with an updated administration GUI, support for coordinated authentication and changelists between several servers, dynamic configuration, and other improvements. The addition of replicated servers helps Perforce scale to support larger user bases, automation, and out-of-the-box disaster recovery with minor administrative overhead. Other Perforce components, such as P4Sandbox, have no extra administrative overhead. Streams actually reduces administrative support requirements in many cases, by incorporating client view management and workflow management best practices into the core product.

Administering ClearCase has grown more complex, as it now uses WebSphere, the CCRC WAN or CM server, and the IBM HTTP server. ClearCase has added a Tivoli-based monitoring solution for MultiSite and a unified installation manager.

Binary File Management

Many industries, including EDA and gaming, must consider a wide variety of digital assets beyond software source code. Graphics, animation clips, story boards, wire frames, and chip designs all comprise important intellectual property and must be versioned in the same manner as text-based assets.

Perforce has improved its support for handling binary files. Large digital assets can be stored in external storage units, and Perforce can be configured on a file-by-file basis to retain only a specified number of revisions of these assets, purging older versions. Obsolete assets can also be archived to offline storage, rather than purged.

ClearCase has added no support for managing large digital assets.

Extensibility

Perforce has fully supported APIs for C/C++, Perl, Ruby, Python, PHP, and Java. The Perforce Broker provides a framework for additional guidance and management of user activity, while the JavaScript API allows users to add visual extensions to Perforce clients. P4toDB allows report engines and other data mining tools to access Perforce metadata through a standard relational database like MySQL or SQL Server.

ClearCase introduced a Java API and the WebSphere architecture, but relies primarily on the UCM and Rational Team Concert platforms for providing workflow support.

Branching and Usage

Branching and Release Management

Perforce's revamped integration engine is substantially faster and now handles the most complex branching and merging scenarios. Non-content changes, such as renames, refactorings, and file type changes, receive special handling.

Perforce Streams provide a lightweight branching and release management framework. Streams help to model a project's branching structure, including how branches relate to one another, the intended merge pathways, and what components are included in a stream. This information is used to simplify and automate common operations such as creating workspaces and merging changes between streams. Visual tools like the Stream Graph provide a rich set of information to all users.

ClearCase has introduced a way to prevent and detect "evil twins", an obscure oddity of the ClearCase modeling of software development. In other ways the ClearCase branching tools are largely unchanged. ClearCase UCM and Rational Team Concert provide process templates and other release management features.

Ease of Use

Perforce introduced shelving, or the ability to save work-in-progress on the server without officially committing it. Shelving is a powerful feature with a variety of uses, such as task switching, collaborating with other users for task hand-off, cross-platform pre-submit testing, and code reviews. ClearCase has no shelving facility.

Atomic Transactions

Perforce has always supported atomic transactions across the repository.

ClearCase has added optional support for atomic transactions on a per-VOB basis. That is, if a checkin contains elements from three VOBs, there will be three atomic transactions, one per VOB. Atomic transactions only work with base ClearCase, not UCM.

Build Acceleration

clearmake, the ClearCase build tool, uses the ClearCase MVFS file system and ClearCase dynamic views to detect build dependencies and linkin objects that were built elsewhere. clearmake could save a significant amount of build time, particularly for the first build in a workspace.

Similar build acceleration technology is now available from vendors including Electric Cloud, with superior performance and no dependencies on the slow MVFS file system.

Perforce has always practiced a best-of-breed approach to take advantage of complementary solutions.

Learn More

Evaluating Perforce

More than 400,000 users at 5,500 companies rely on Perforce for enterprise version management. Perforce encourages prospective customers to judge for themselves during a typical 45-day trial evaluation. Free technical support is included with your evaluation. Get started: perforce.com/trial

Scheduling a Demo of Perforce

To learn more about Perforce, schedule an interactive demo tailored to your requirements:

perforce.com/product/demos

Migrating to Perforce

Perforce Consulting Services has experience assisting customers with migrations from various software version management systems. For more information, visit:

perforce.com/consulting

perforce.com



North America
Perforce Software Inc.
2320 Blanding Ave
Alameda, CA 94501
USA
Phone: +1 510.864.7400
info@perforce.com

Europe
Perforce Software UK Ltd.
West Forest Gate
Wellington Road
Wokingham
Berkshire RG40 2AT
UK
Phone: +44 (0) 845 345 0116
uk@perforce.com

Australia
Perforce Software Pty. Ltd.
Suite 3, Level 10
221 Miller Street
North Sydney
NSW 2060
AUSTRALIA
Phone: +61 (0)2 8912-4600
au@perforce.com