



# Comparing Perforce and Microsoft Team Foundation Server



## Overview

This document compares Perforce (version 2006.2) with Microsoft Visual Studio 2005 Team Foundation Server (TFS), and contrasts their significant differences. Included are qualitative aspects such as usability and administration, as well as quantitative metrics such as the time it takes to branch a codeline.

## Executive Summary

	Perforce	Microsoft TFS
<b>Performance</b>	Performance benchmarks highlight significant differences between Perforce and TFS. See page 3 for test results.	Performance benchmarks highlight significant differences between TFS and Perforce. See page 3 for test results.
<b>Usability</b>	Perforce supports sparse branching and indirect integration.	No support for advanced features such as sparse branching and indirect integration.
<b>Administration</b>	By using standard file systems, network features, and a single server, the only overhead added by Perforce is that associated with backup and other maintenance activities.	Reliance on SQL Server for file and metadata storage can lead to higher administrations costs.
<b>Platform support</b>	Available on a wide variety of platforms.	Primarily geared for Windows only.
<b>Third-party integration</b>	Integrates with a large variety of application lifecycle management tools.	Tightly integrated with Microsoft Visual Studio Team System.
<b>Support</b>	Available through Perforce.	Available through Microsoft.

2

## Performance

To test TFS performance against Perforce, comparative benchmarking tests were constructed to reflect real-life steps developers would take in carrying out their

day-to-day work. In some cases, individual commands were used. In other cases, scripts orchestrating multiple commands were employed. In all cases, functionally equivalent sets of commands were used and executed using the command line utilities.

## Test Environment

The test environment was selected to provide a common development environment contemporary to 2006. The configuration exceeded the minimum recommendations for both TFS and Perforce. The environment consisted of the following:

- Network: Both server and client on same segment of 100Mbps network
- Server OS: Windows Server 2003 SP1
- Client OS: Windows XP Professional SP2
- Server Hardware: Dell PowerEdge 1850 with a dual 3.2GHz Xeon processor, 4GB RAM, and 146GB hard disk
- Client Hardware: Dell Latitude D620 with a Pentium Core Duo 1.66GHz processor and 1GB RAM

## Test Procedure

Testing began with both source control systems in a newly installed state. Tests were run on each system alternately. All developer and source import tests were automated and timed using the DOS `time` command to a granularity of .01 second. Software installation was timed externally.

Where tests would invoke the file system cache or database cache, the tests were run three times, and average times for second and third iterations were recorded. This simulates an environment where many files are accessed repeatedly, as you would expect through builds and source code searches.

A common file structure was used. Each product created, deleted, and modified the same files during testing. File copies were used to simulate file creation, and automated edits were used to simulate file modifications.

When a sequence of commands was required to complete a single test for either system, a DOS batch file was used to execute the test for both systems. For TFS, the recommended<sup>1</sup> `@` switch was used to invoke the command file to reuse a single connection to the TFS server.

No performance-oriented techniques were used to optimize the use of either product, other than those used to remove interactive effects. For TFS, LAN connection settings in Internet Explorer were configured as recommended<sup>2</sup> on MSDN.

The tests were designed to compare the two products during normal developer usage. For all processing, automated merges were used, as the tests were constructed to avoid file conflicts. The following tasks were tested:

- Server installation
- Client installation
- Importing the source from a local file system
- Populating the workspace
- Updating the workspace (with no changes on the server)
- Labeling all files
- Checking out files
- Deleting files on the server
- Undeleting files on the server
- Branching files
- Merging back to the parent branch
- Checking out binary files
- Branching binary files

## Test Data

Test data was a combination of two open source software packages: Apache httpd 2.0.58 and Apache Tomcat 5.5.15. A "small" project was composed of one copy of each package: 5,548 files totaling 54.7MB. A "large" project was composed of five copies

of each package in separate trees: 27,740 files totaling 273MB. Tests for the large project performed the same operations in each tree. Test data for binary files consisted of 1GB of MPG files.

than installing Perforce, due to the number of applications, services, and service packs involved. Perforce outperformed TFS in all but two tests. The detailed test results are below. All times are in seconds except where otherwise noted.

## Test Results

Installing TFS was a significantly larger task

	Perforce	TFS
Install server	5 minutes	5 hours
Install client	3 minutes	55 minutes

	Perforce Small	TFS Small	Perforce Large	TFS Large
Import source	56.89	212.59	418.29	1406.97
Populate workspace	53.67	49.40	249.83	254.31
Update workspace (no updates)	0.43	2.79	1.43	5.27
Label/baseline all files	0.90	2.52	2.85	9.33
Check-out files	12.62	113.06	45.38	299.17
Delete files on server	16.95	72.11	139.36	364.30
Undelete files on server	126.47	69.47	813.15	359.39
Branch files	7.51	33.45	29.46	129.92
Merge back to parent branch	9.06	2.67	53.19	16.64
Binary: add files	-	-	629.30	1263.69
Binary: check-out files	-	-	0.81	3.22
Binary: branch files	-	-	1.81	309.57

## Usability

Every software configuration management (SCM) system has its own set of unique features. Its adoption and success is driven by how adaptable it is to the users' processes and practices. This section looks at several features, and compares their implementation in TFS and Perforce.

### Client Specification Model

A user needs to set up a client specification to work with files managed by the SCM system. A user's client specification tells the SCM system where in the local file system the user's workspace is rooted, through a series of folder mappings.

In TFS, each mapping takes files and folders in the repository and associates them to files and folders in the local workstation, creating one-to-one mapping.

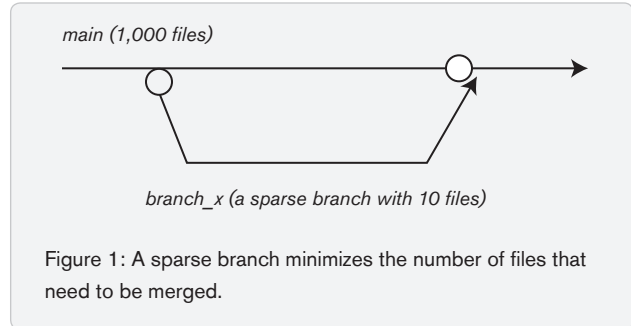
Perforce provides greater flexibility; the mapping does not have to be one to one. This can result in productivity gains, as illustrated in the following scenarios.

### Sparse Branching

Sparse branching is a popular branching practice wherein only subsets of files are branched, instead of branching all files from one branch to another (and subsequently merging them back). This approach can be beneficial when working with branches that consist of hundreds or thousands of files, but that require only a few files to be edited at a time (see Figure 1).

For example, a release codeline can be branched into a task branch to isolate a risky bug fix. Similarly, a release branch can be branched into a patch branch for an already

released version. All the other files (that have not been branched) are referenced from the original branch, so that the client workspace has a complete view of all project files.



TFS does not support sparse branching. A user can individually “cloak” or “uncloak” project folders. However, the same set of files from two different repository folders (for example, `main` and `branch_x`) cannot be mapped to the same local folder in a workspace.

Perforce provides a more flexible client specification model that allows the use of “overlays.” For example, a user “Joe” might have a client workspace `joe_dev` defined as follows:

```
//depot/main/...      //joe_dev/main/...  
+//depot/branch_x/... //joe_dev/main/...
```

By branching only those 10 files from `main` to `branch_x`, and by setting up the client workspace view as above, the workspace `joe_dev` has access to all 1,000 project files, with the caveat that the 10 files from `branch_x` overlay the corresponding files from `main`.

### Sharing Files and Folders

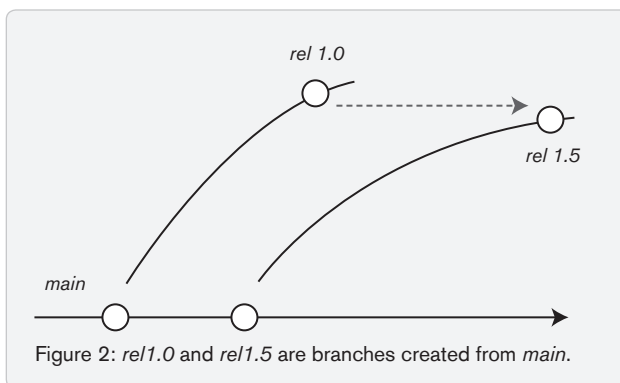
Users of Visual SourceSafe (VSS) might recall using shared folders. This provides the ability to put files that are common across multiple projects (DLLs, COM objects, class libraries, and so on) in one folder and share that folder across multiple projects.

TFS does not provide any such feature; simulating VSS shared folders would require doing repeated file gets between folders, consuming additional processing overhead.

Perforce's client view mechanism can be put to use to effectively control how components are shared across multiple projects. Perforce's "overlay" mechanism (optionally in conjunction with triggers) can be used for sharing components across multiple projects.

### Indirect Integration

Indirect integration comes into play when you want to merge changes between two branches that are not directly related. For example, while using the mainline branching model, `rel1.0` is a release branch created from `main`, and `rel1.5` is another (perhaps refactored) branch created from `main` (see Figure 2).



Although `rel1.0` and `rel1.5` have a common ancestor, they are not directly related. TFS will not allow merging between those two branches. Users need to merge from `rel1.0` back to `main`, and then from `main` to `rel1.5`. This extra step not only requires extra effort, but might induce unexpected side effects—for example, what if some of the files in `main` have already been deleted?

Perforce, on the other hand, allows merging between `rel1.0` and `rel1.5`. It is able to decipher the common heritage (`main`) between them and uses that as the basis of the merge operation, resulting in better, quicker merging of branches.

### Locking

To coordinate concurrent development, SCM systems offer different file locking mechanisms. For example, if Joe checks out `foo.c` from the branch `main`, the file can be locked in such a way that other users can concurrently check out the same file, but are not allowed to check it back in before Joe does. Alternatively, `foo.c` can be locked in such a way that no other user is allowed even to check it out while Joe has it checked out (exclusive check-out).

Based upon the selected locking behavior, SCM systems can automatically lock files, or users can do so manually. Automatic locking is useful for implementing uniform development policies across the entire team. The granularity at which automatic locking can be configured plays an important role in enhancing concurrency.

In TFS, automatic exclusive check-outs can be specified at the project level and for file types, but not for individual files; users need to do exclusive check-outs manually for each file.

Perforce provides greater flexibility, by specifying the behavior as part of typemap configuration. For example:

```
Typemap:  
+1 //depot/main/Images/...
```

The preceding settings in Perforce cause all new files in the `Images` folder within `main` to be configured for exclusive check-outs for all users. In TFS, locking can only be specified at the main level. Moreover, in Perforce, locking behavior can also be defined for individual files. For example:

```
p4 edit -t +1 //depot/files/image.jpg  
p4 submit
```

This configures the `image.jpg` file for automatic exclusive check-out for all users.

## Data Management

SCM systems typically retain old files and metadata for auditing and reporting purposes. When a user deletes a file, it is marked as being in a deleted state rather than actually being removed from the repository. However, at times it might be desirable to purge old repository data, such as a project consisting of gigabytes worth of binary files that is no longer needed.

TFS does not provide any utility to remove old data permanently from its database. This can present data management issues and potentially slow system response times as historic data grows over a period of time.

Perforce provides an `obliterate` command (restricted to Perforce administrators) for permanently removing files and their associated metadata from the depot.

## Administration

TFS architecture comprises of several product components:

- SQL Server 2005
- ASP.NET
- Windows SharePoint Services
- Internet Information Server

It takes several hours to install TFS. The presence of several product components distributed over multiple servers adds to the training, maintenance, and diagnostic responsibilities of system administrators. This overhead is in addition to that associated with backups and other maintenance activities required to support the operation of TFS.

For example, backing up TFS includes backing up 11 databases on SQL Server, as well as backing up the Report Server Encryption Key. For larger projects, the recommended multiple server deployments associated with TFS makes disaster-recovery planning and execution inherently more complex.

The Perforce Server architecture is much simpler. For core SCM functionality, the server software is self-contained, including an embedded database for tracking the metadata. By using standard file systems, network features, and a single server, the only overhead added by Perforce is that associated with backup and regular user maintenance activities. Typically, one part time Perforce administrator is adequate for supporting about a hundred users.

## Platform Support

Large enterprises usually consist of heterogeneous hardware and OS platforms, and the ability to manage projects using the same SCM tool becomes a compelling requirement.

TFS is a Windows-based product, with only a few third-party client interfaces available on non-Windows platforms.

The Perforce Server is available on a wide variety of platforms, including Windows, Linux, Mac OS X, Solaris, and UNIX. P4V, the Perforce Visual Client, is a cross-platform client that is available on Windows, Linux, Solaris, Mac OS X, and FreeBSD.

## Third-Party Integration

TFS is a core component of the Microsoft Visual Studio Team System (VSTS) product suite. While some third-party interfaces for TFS are available, it work best as part of the VSTS suite. Users of TFS are strongly

encouraged to buy into the entire VSTS suite, rather than using different products for different parts of the application development life cycle.

Perforce takes the opposite approach by providing a comprehensive SCM solution, integrating<sup>3</sup> well with a large number of third-party products (defect tracking, IDEs, build tools, and so on). This enables enterprises not only to preserve their investment in currently deployed tools, but also to take a of best-of-breed approach to selecting from the large variety of products available in the market that interoperate with Perforce.

## Support

Both Microsoft and Perforce Software provide technical support. Because the quality of a technical support organization is better experienced than described, Perforce encourages its prospective customers to judge for themselves during a typical 45-day trial evaluation.

- 
1. MSDN library, Microsoft, [http://msdn2.microsoft.com/en-us/library/1az5ay5c\(VS.80\).aspx](http://msdn2.microsoft.com/en-us/library/1az5ay5c(VS.80).aspx) (May, 2007).
  2. MSDN blog by Buck Hodges, Microsoft, <http://blogs.msdn.com/buckh/archive/2005/12/12/503015.aspx> (May, 2007).
  3. Third-party Software Integrations, Perforce Software, <http://www.perforce.com/perforce/products/integrations.html> (May, 2007).



**North America**  
Perforce Software Inc.  
2320 Blanding Avenue  
Alameda, CA 94501  
USA  
+1 510.864.7400  
[info@perforce.com](mailto:info@perforce.com)

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

**Australia**  
Perforce Software Pty. Ltd.  
Level 10, 100 Walker Street  
North Sydney  
NSW 2060  
AUSTRALIA  
+61 (0)2 8912-4600  
[au@perforce.com](mailto:au@perforce.com)

Additional SCM comparisons available at <http://www.perforce.com/comparisons>