TECHNICAL WHITE PAPER

# Accelerating Workspace Creation for Perforce Users

A Unique Methodology for Accelerating Workspace Creation for Perforce Users

# Contents

**Introduction** .......... **3**

**Challenges During Software Development** .......... **3**

Rapid File Tool (RFT) Kit .......... 4

Rapid File Tool Kit for Perforce Workspace Creation .......... 4

Test Results and Observations .......... 6

**Conclusion** .......... **8**

## Introduction

Software development processes are key focus to improving product and service delivery for faster time to market (TTM)—and the competitive advantage that comes with it. Software development has evolved over the years from a waterfall model to a continuous development (CD) process. Evolving to be able to continuously modify, test, build, and deliver software requires constant optimizations of the development tools and processes used in the software delivery pipeline to improve code quality and drive innovation.

The core components of any software development lifecycle are both source code management (SCM) tools and binary package management systems. Perforce, Subversion (SVN), and different flavors of Git are common SCM tools used in various industry verticals such as electronic device automation (EDA), finance, oil and gas, etc.

Perforce is a very popular SCM tool used in EDA, gaming, and organizations that deal with large codebases. A validated architecture of Perforce on Pure Storage FlashBlade® has many advantages such as high availability for edge servers with zero downtime, improved performance for Perforce databases, scalability for user workspaces, faster backup and recovery for Perforce databases, and data reduction for large repositories. Subversion is an SCM tool that is still used in some of the traditional design environments.

## Challenges During Software Development

While more and more customers are transforming and automating their development workflows to adapt a continuous integration (CI) process, there are still some kinks to work out. First, most of the EDA, semiconductor, and gaming companies have very large codebases with many development branches. The source code size can range from 10GB to multiple TBs in size. Developers have to wait to copy the source code files into their workspaces and perform a full build to complete during the checkout process out from the respective development branches.

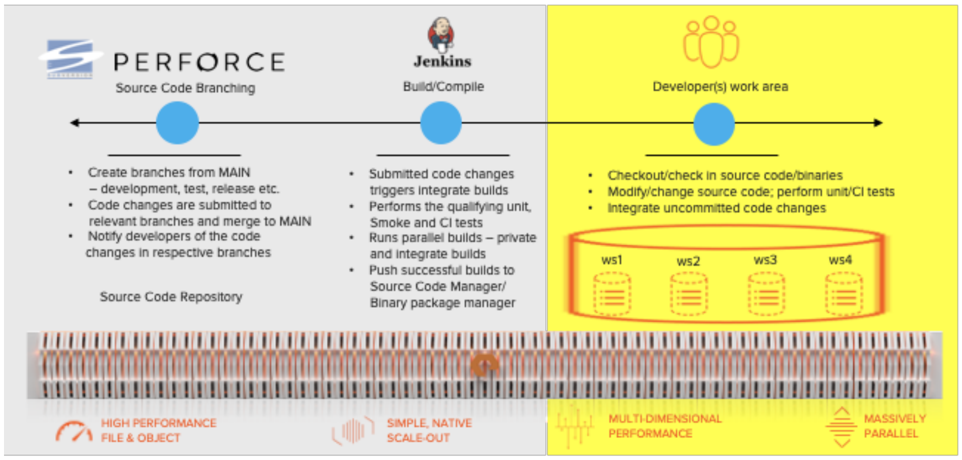

PERFORCE
Source Code Branching
Jenkins
Build/Compile
Developer(s) work area
• Create branches from MAIN – development, test, release etc.
• Code changes are submitted to relevant branches and merge to MAIN
• Notify developers of the code changes in respective branches
Source Code Repository
• Submitted code changes triggers integrate builds
• Performs the qualifying unit, Smoke and CI tests
• Runs parallel builds – private and integrate builds
• Push successful builds to Source Code Manager/ Binary package manager
• Checkout/check in source code/binaries
• Modify/change source code; perform unit/CI tests
• Integrate uncommitted code changes
ws1
ws2
ws3
ws4
HIGH PERFORMANCE FILE & OBJECT
SIMPLE, NATIVE SCALE-OUT
MULTI-DIMENSIONAL PERFORMANCE
MASSIVELY PARALLEL

**Figure 1:** CI workflow during the software development process.

As shown in Figure 1, depending on the size of the development branches, the users often have to checkout to their respective work areas and then run a full build of the source code before they can start working. A lot of time is lost in ramping up developers in a large-scale environment. In the above Figure, the user workspaces are configured as different directories that are stored in a single file system on FlashBlade.

The SCM or the Perforce servers are under a lot of stress as developers checkout and check in source code and its changes in parallel. This results in a high amount of load in the form of CPU utilization, IOPs, and bandwidth generated on the Perforce servers that can cause a major performance bottleneck. Adding more Perforce edge servers helps with the high availability and local caching for reads, but the writes or code changes will still be committed to the main Perforce servers.

This paper will focus on improving developer productivity by accelerating the creation of user workspaces on Pure Storage FlashBlade using Rapid File Tool (RFT) kit in Perforce environments, as well as address the challenges mentioned in this section.

## Rapid File Tool (RFT) Kit

Filesystem operations using standard UNIX commands like “ls,” “du,” “chmod,” “chown,” “find,” “rm,” and “pcp” take a very long time to complete—especially on large datasets stored on NFS shares with deep directory structures and varying file sizes. The standard UNIX commands are single-threaded serial RPC calls with a single TCP connection to the FlashBlade.

Pure Storage provides a host-based utility called Rapid File Tool Kit (RFT), a multi-threaded Linux utility that scales in performance with concurrent RPC calls and multiple TCP connections to FlashBlade. The RFT kit includes commands like “pls,” “pdu,” “pchmod,” “pchown,” “pfind,” and “pcp” that can manage, find, and copy millions of files stored on FlashBlade. These commands generate parallel threads that are 5x to 50x faster than traditional UNIX counterparts. The RFT kit can be used with various workflows like software development, EDA, AI/ML, genomics, and analytics.

## Rapid File Tool Kit for Perforce Workspace Creation

In Figure 2, most software development environments will have different development branches for the developers to work on. In a normal Perforce production environment, “p4 sync” is commonly used along with “chown” to copy all the source files, register the workspace in the Perforce database, and change the ownership of the files and directories. The "chown" operation normally takes longer for datasets that are larger than 100GB.

Pulling the code from every development branch and compiling them independently in different work areas can compound the time it takes to bootstrap the developer to start working. Instead, every development branch can have its own work area with the source files and a "full" build is performed at one time. The development branch work area now has the source files, binaries, and all of its dependencies. Each time the developer commits a change to the development branch it can be automated to trigger an incremental build on the changes only. This keeps the development branches up to date and notifies the developers who still have uncommitted code changes in their private work area. The development branch work areas can be configured in different directories sitting next to the MAIN source files in the same "p4 depot" or "svn-repository" file system on FlashBlade.

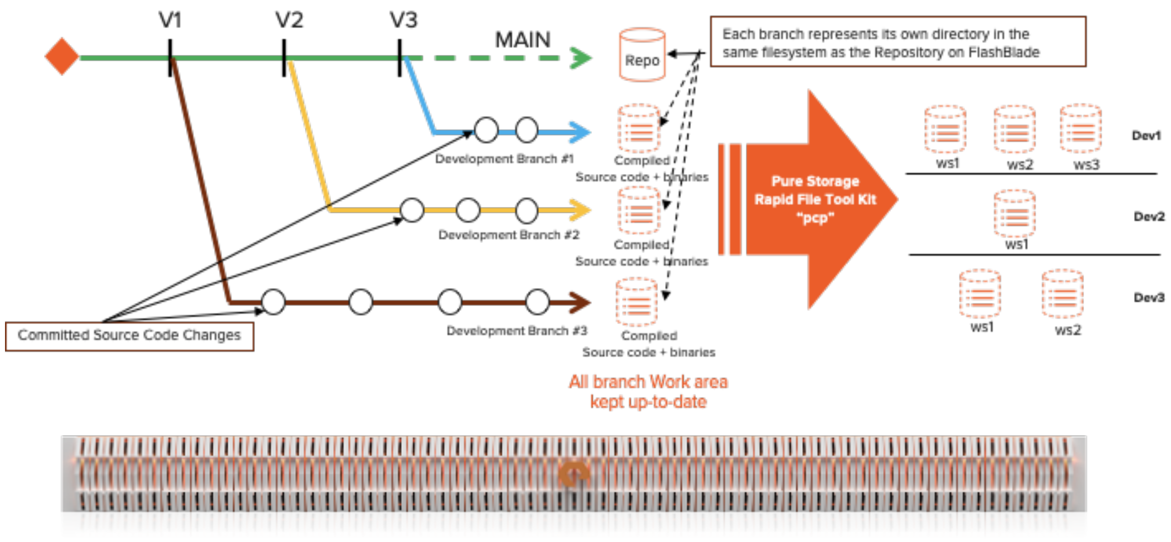

V1
V2
V3
MAIN
Repo
Each branch represents its own directory in the same filesystem as the Repository on FlashBlade
Development Branch #1
Compiled Source code + binaries
Development Branch #2
Compiled Source code + binaries
Development Branch #3
Compiled Source code + binaries
Committed Source Code Changes
Pure Storage Rapid File Tool Kit "pcp"
ws1
ws2
ws3
Dev1
ws1
Dev2
ws1
ws2
Dev3
All branch Work area kept up-to-date

**Figure 2:** Source Code Branches vs. Developer Workspaces.

The "pcp" command in the RFT kit is used to copy the latest source code and binaries from the development branch area to the developers' private workspaces. Once the files are copied, the "p4sync -k" command helps to register the developer workspace to the Perforce database. As the "p4 depot" and the user workspace are configured on FlashBlade over NFS, the following "pcp" command copies all the source files and the binaries from the development work areas in the "p4 depot" to the developers' private workspaces using source and target IP addresses.

```
pcp -r -p -u --source <ip_addr>:/p4depot --dest <ip_addr>:/p4workspaces /android-ws/. /u1ws3
```

In the above "pcp" command, the source and the destination IP addresses are on the FlashBlade. This could be the same or different IP address on the FlashBlade. However, it is recommended to have a different IP address for the source and the target. The source file system is the "p4 depot" or the svn-repository that has the MAIN source files and the compiled development branch work areas. The target is the "p4 workspaces" or the "svn-workingcopies" file system for all the developers private work areas.

## Test Results and Observations

A formal test was performed on different source datasets with Perforce and Subversion on FlashBlade.

- The P4 depot and svn-repository file systems were configured over NFS on FlashBlade.
- The datasets tests were 11GB, 52GB, 101GB and 153GB for the Perforce testing.
- The tests were performed by measuring the time it took to Perforce user workspaces using “p4 sync” and “chown” as the baseline.
- The Subversion testing was performed with 800MB dataset size.
- All the tests were repeated for the P4 and SVN datasets using Pure Storage RFT kit “pcp” command and the completion time was compared with the baseline.
- The workspace creation was scaled up to 20 Perforce clients.

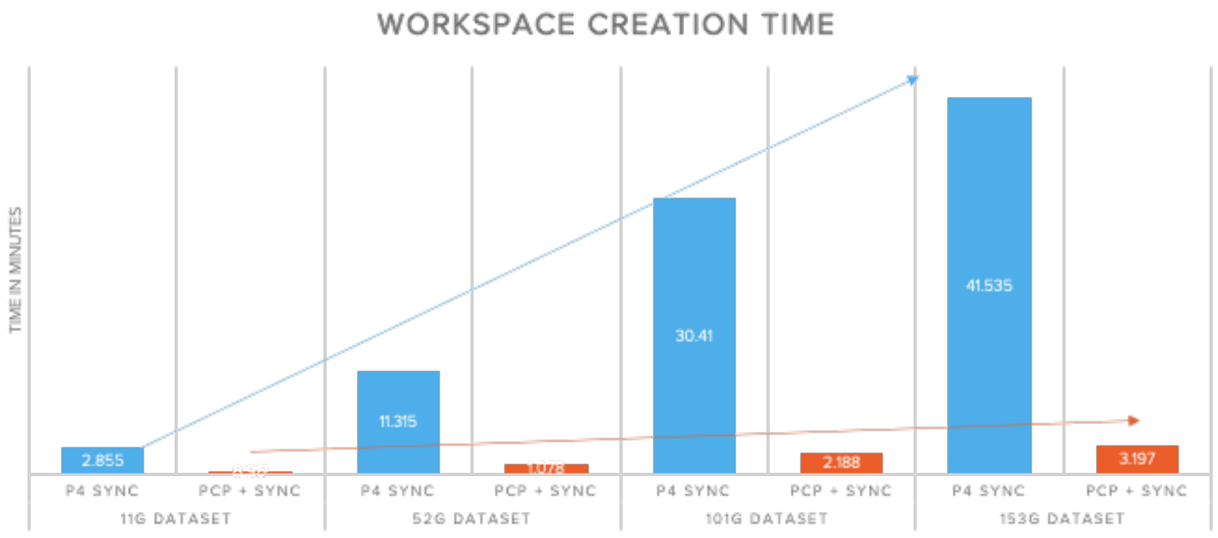
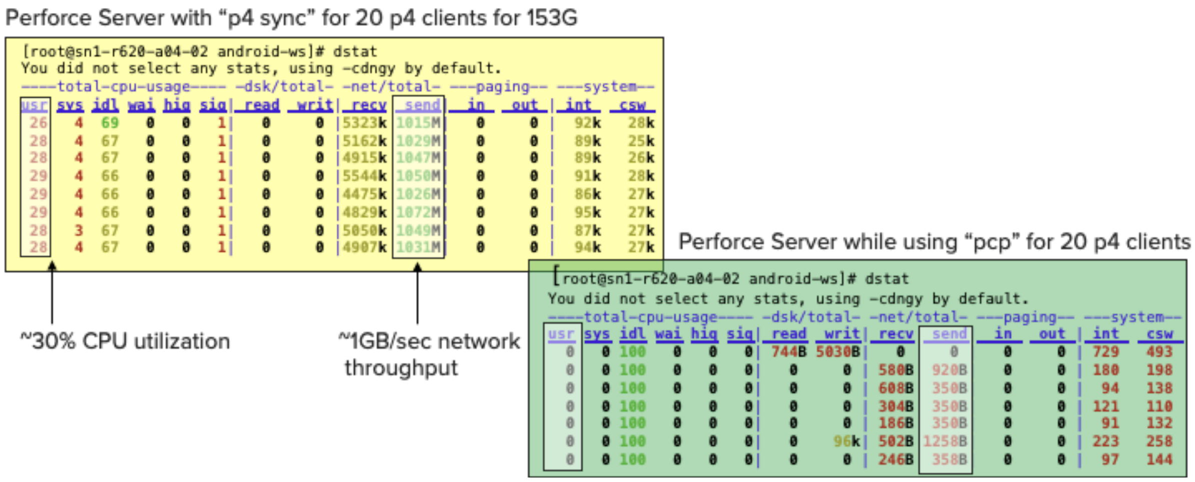


**Figure 3.** Perforce User Workspace Creation Tests.

Figure (3) shows that the “pcp” took 3:19 minutes to complete copying all the source files and the binaries to the Perforce developers’ private workspaces compared to the Perforce “p4sync,” which took 41.53 minutes. The completion time for all the dataset sizes shows a linear relationship between the native Perforce “p4sync” and Pure Storage “pcp + p4sync -k” with a 10x improvement using “pcp” in the developer workspace creation process.

Another critical observation was that the Perforce server was not performing any IOPs or throughput while the “pcp” operation was copying the files and the binaries to the developer workspaces. Figure (4) below illustrates the comparison of the Perforce server’s overhead when the native “p4sync” was used. The table on the left shows that CPU utilization for the Perforce server was ~30% and pushing 1GB/sec throughput. A 1GB/sec throughput saturates a 10Gbe ethernet connection. In most of the customer scenarios, the network becomes the most common bottleneck while scaling the number of developer workspaces.

Meanwhile, in Figure 4, the table on the right shows the resources Perforce was not using during the “pcp” operation to copy the source files and binaries to the developer’s private user workspaces. The CPU and the network utilization were offloaded to the FlashBlade during the “pcp” operation. Even though FlashBlade resources were heavily used during the “pcp” operation, the duration of the “pcp” operation was short. This allows the Perforce server to perform some core functionality to write and update committed code changes from local and remote users.


Perforce Server with “p4 sync” for 20 p4 clients for 153G
~30% CPU utilization
~1GB/sec network throughput
Perforce Server while using “pcp” for 20 p4 clients

```
[root@sn1-r620-a04-02 android-ws]# dstat
You did not select any stats, using -cdngy by default.
----total-cpu-usage---- -dsk/total- -net/total- ---paging-- ---system--
usr sys idl wai hiq siq| read  writ| recv  send|  in   out | int   csw 
 26   4  69   0   0   1|   0     0 |5323k 1015M|   0     0 |  92k   28k
 28   4  67   0   0   1|   0     0 |5162k 1029M|   0     0 |  89k   25k
 28   4  67   0   0   1|   0     0 |4915k 1047M|   0     0 |  89k   26k
 29   4  66   0   0   1|   0     0 |5544k 1050M|   0     0 |  91k   28k
 29   4  66   0   0   1|   0     0 |4475k 1026M|   0     0 |  86k   27k
 29   4  66   0   0   1|   0     0 |4829k 1072M|   0     0 |  95k   27k
 28   3  67   0   0   1|   0     0 |5050k 1049M|   0     0 |  87k   27k
 28   4  67   0   0   1|   0     0 |4907k 1031M|   0     0 |  94k   27k
```

```
[root@sn1-r620-a04-02 android-ws]# dstat
You did not select any stats, using -cdngy by default.
----total-cpu-usage---- -dsk/total- -net/total- ---paging-- ---system--
usr sys idl wai hiq siq| read  writ| recv  send|  in   out | int   csw 
  0   0 100   0   0   0| 744B 5030B|   0     0 |   0     0 | 729   493 
  0   0 100   0   0   0|   0     0 | 580B  920B|   0     0 | 180   198 
  0   0 100   0   0   0|   0     0 | 608B  350B|   0     0 |  94   138 
  0   0 100   0   0   0|   0     0 | 304B  350B|   0     0 | 121   110 
  0   0 100   0   0   0|   0     0 | 186B  350B|   0     0 |  91   132 
  0   0 100   0   0   0|   0    96k| 502B 1258B|   0     0 | 223   258 
  0   0 100   0   0   0|   0     0 | 246B  358B|   0     0 |  97   144 
```

**Figure 4.** Resource Utilization for the Perforce Server.

The third observation was the data reduction on the FlashBlade for the user's private workspaces. The test showed there was a data reduction of 2.3:1 for the dataset in the file system on FlashBlade used by all the twenty p4 clients.

The SVN testing was only performed on an 800MB dataset that took six minutes to checkout the code to the working copy. The “pcp” command was able to complete the copy operation within 3 seconds. There was a huge 60x improvement to copy the dataset to the SVN working copy directory. The expectation is that the results would scale for larger datasets in a similar fashion to the Perforce/pcp testing described above.

## Conclusion

This paper presents a unique methodology for accelerating workspace creation for Perforce users. It is easy to set up and use and requires minimal changes to processes and workflows. There are numerous solutions available in the market that can accelerate the creation of private developer workspaces. The RFT kit "pcp" command is unique in that it allows to copy millions of files in parallel in a very short time. This parallel copy operation allows the developer private workspaces to reside as directories in a single filesystem on FlashBlade. Offloading the Perforce server resource utilization overhead to the FlashBlade along with the data reduction makes a positive impact to the cost efficiency of the overall software development pipeline. This solution improves developer productivity by getting developers up  to speed up to 10X faster. Reducing build time after submitting the code changes to the development branch work areas and syncing up the source files and binaries in private work areas also provide significant productivity in the software development lifecycle.