PURESTORAGE®

PERFORCE

PERFORCE SOFTWARE DEVELOPMENT ON PURE

Best Practice Guide for FlashBlade and FlashArray

August 2017

TABLE OF CONTENTS

INTRODUCTION

Perforce Helix is a leader in the Software Configuration Management space (SCM). Helix is designed to manage large digital assets within a single development repo, and remains the versioning engine of choice for organizations looking to scale their development environment and accelerate the software development process. Helix can be deployed in engineering environments ranging from small teams to large enterprises and across multiple industries, such as gaming, engineering, automotive, and healthcare.

With Pure Storage Data Platform solutions, including FlashBlade™ and FlashArray, customers can harness the breakthrough performance of all-flash storage system architectures that are ideal for development projects requiring high concurrency. FlashBlade shatters current performance bottleneck limitations in a compact and easy to manage storage solution that allows developers to complete software builds significantly faster – thus accelerating their time to results.

With two organizations so focused on performance, Pure Storage and Perforce Helix provide an integrated solution, perfectly suited to Agile development methodologies at scale, that allows for collaboration between multiple development teams and simplifies data management, resulting in reduced development cycles and improved productivity. Customers can leverage these two platforms to accommodate their growing requirements well into the future.

SCOPE

This best practice report contains information that readers can use to aid both in their development approach and in implementing these integrated systems. We provide performance benchmarks of Helix on FlashBlade and FlashArray, as well as guidance and options to users for planning and deploying a Perforce Helix and Pure Storage solution in their environments.

INTENDED AUDIENCE

Our best practice report is aimed at any user that is interested in leveraging Perforce applications to add more performant and secure digital asset management within their Pure Storage Data Platform. This paper assumes the reader is familiar with Perforce Helix and Pure Storage FlashBlade and FlashArray all-flash storage solutions.

For more details on these two solutions, visit **perforce.com** or **purestorage.com**, or contact our support teams with questions.

PERFORCE HELIX ARCHITECTURE OVERVIEW

Perforce is designed as a distributed application and uses a client/server architecture. In such an architecture, the server and client components are typically installed on separate machines, accessible across a network. The client and server components communicate using standard network protocols, such as TCP/IP. The Perforce server host machine listens for commands from Perforce clients on a dedicated port. A Perforce client connects with the server using the server machine's host name and the dedicated port number.

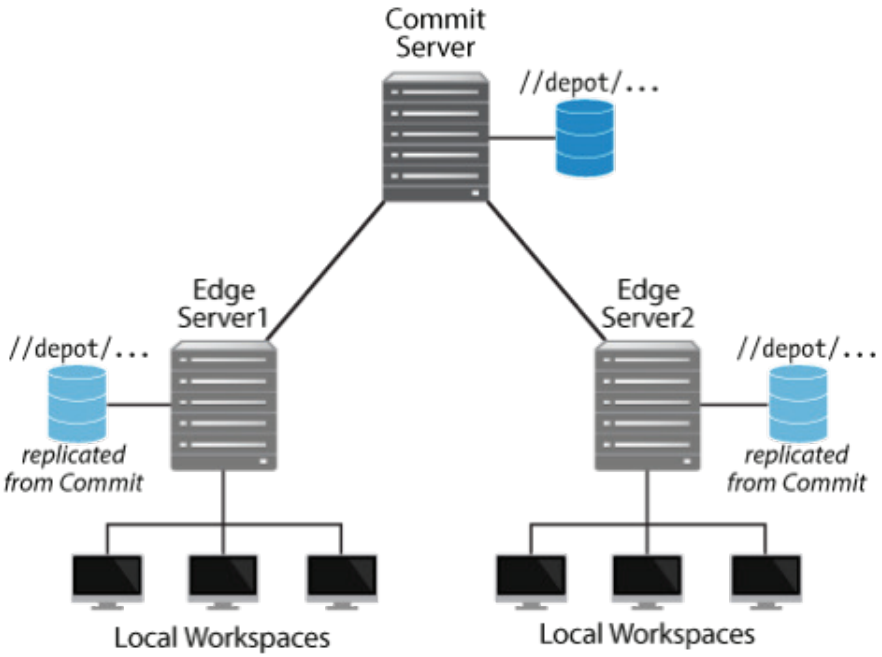


Figure 1. Perforce Helix Architecture

PERFORCE SERVER

In the Perforce client/server model, the server controls all access to the versioned files, while maintaining revision history and other system metadata.

PERFORCE CLIENT

Perforce provides a variety of client applications for end users, including standalone clients, plug-ins, and APIs. The clients include P4V, P4Admin, and the P4 command line client. Plug-ins include P4VS (Visual Studio), P4Eclipse, and P4GT. APIs, such as P4Java, P4Python, and P4Perl, help users create applications that interact with Perforce.

Clients connect to the Perforce server over TCP.

PERFORCE DATABASE

The Perforce metadata database (an embedded database, stored in the db.* files, and typically up to hundreds of GB) includes information about the versioned files, such as the revision history for each file. It also includes system information, such as user and group definitions, labels, and access permissions.

The server writes all metadata updates to a (transaction) journal, which is used for replication and to restore Perforce metadata in case of system failure. As a best practice, the journal should be located on a volume separate from the metadata. The journal and other server logs are sequentially written by the server process and are regularly rotated.

PERFORCE DEPOTS

The Perforce server stores files under its control in organizational units called depots. The depot files – often many TB – are stored directly in the filesystem. Text files can be stored in RCS format (reverse delta) or as a compressed copy of each revision. Binary files, typically stored as a compressed copy per revision, can also be stored uncompressed.

In Perforce, you can have multiple depots with different storage systems and purposes.

REPLICATION SERVICES

This is the duplication of server data from one Perforce server to another Perforce server. The benefits of using Replication Services are:

- Provide warm standby servers
- Reduce load and downtime on a primary server
- Provide support for build farms
- Forward write requests to a central server

CLIENT USAGE PATTERNS

Users do their work within workspaces on client machines. Actions for human users include:

- Syncing of the initial workspace (can be many GBs of files)
- Modifying files (adding/editing/deleting/renaming)
- Submitting changes back to the server
- Reporting
- Branching and merging

Build Farms are automated processes and mainly involve:

- Syncing of “sources” to workspace
- Builds of binary artefacts from sources (not requiring Perforce)
- (Optional) Submission of resulting binary artefacts on successful build

FACTORS WHICH AFFECT PERFORCE PERFORMANCE

These include:

- Size of workspaces (no of files, average size, split between text and binary) required for:
  - Users to work
  - Build farms and other automated processes
- Frequency of syncs
- Frequency of changes (submits to repository)
- Network connectivity/topology (latency/bandwidth/number of concurrent accesses)
- Geographical distribution, which influences the above

For best performance, the connectivity between Perforce server and its storage (database and depot files) should be independent of its connectivity to client machines.

VALUE OF THE PURE STORAGE DATA PLATFORM

FLASHBLADE

FlashBlade is a scale-out all-flash file- and object-based storage system that provides simplicity at scale. It is a new, innovative solution designed to support high concurrency workloads, such as software development, by providing best-of-breed performance in all dimensions of concurrency.

A 4U FlashBlade chassis with always-on data reduction can store as much as 1.6 petabytes of data and is connected to fabric modules delivering 320Gb/s to clients while providing over 18GB/s throughput and 1M IOPS. The entire system is redundant, with no single point of failure.

Critically, as the key component technologies – flash memory, PCIe buses and protocols, Ethernet switches and links – evolve, deployed FlashBlade systems can evolve along with them. Systems that implement the FlashBlade architecture can be expected to deliver uninterrupted service for a decade or more, during which time many, or even most, of their components may be upgraded while they are online.

FLASHARRAY

The FlashArray family delivers software-defined all-flash power and reliability for every need and every budget, from the entry-level FlashArray//M10 to the new FlashArray//X – the first mainstream, 100% NVMe, enterprise-class all-flash array. FlashArray features a modular, stateless architecture, designed to enable expandability and upgradability for generations. It leverages a chassis-based design with customizable modules, enabling

both capacity and performance to be independently improved over time with advances in compute and flash. Customers can thus meet business needs today – and tomorrow.

FlashArray delivers consistent <1ms average latency with inline de-duplication and compression that enables 5 – 10x space savings across a broad set of mixed I/O workloads. FlashArray provides mission-critical resiliency: proven >99.9999% availability, with built-in, fully-integrated, data reduction-optimized backup and disaster recovery. It offers game-changing management simplicity that makes storage installation, configuration, provisioning, and migration a snap.

ACCELERATE SOFTWARE DEVELOPMENT

One of the characteristics of an agile software development environment is its rapidly changing nature, with development scope increasing due to the addition of new features, or new developers added to accelerate schedule. The Pure Storage Data Platform is designed to provide the kind of unparalleled performance that accelerates development.

FlashBlade and FlashArray are both extremely easy to manage and provision, and highly adaptable to changing requirements. Additional storage resources can be quickly provisioned, non-disruptively, to support new requirements in the development process.

PURITY DIRECTFLASH PERFORMANCE

The Data Platform's unique all-flash architecture leverages direct access to NAND flash. It implements global flash management (allocation, I/O optimization, garbage collection, error correction) at the system level, driving 100% of connected raw flash within DirectFlash Modules and Blades, and eliminating the performance density limitations of large SSDs. DirectFlash sheds performance impacting disk-era protocols and I/O interfaces (SCSI, SAS, SATA) and disk emulation software (FTL) – avoiding the unpredictable latency of SSDs and enabling Purity to exploit the full potential of flash. The result is predictable, consistent, microsecond latency alongside higher throughput and reliability, better efficiency, and ultra-high density.

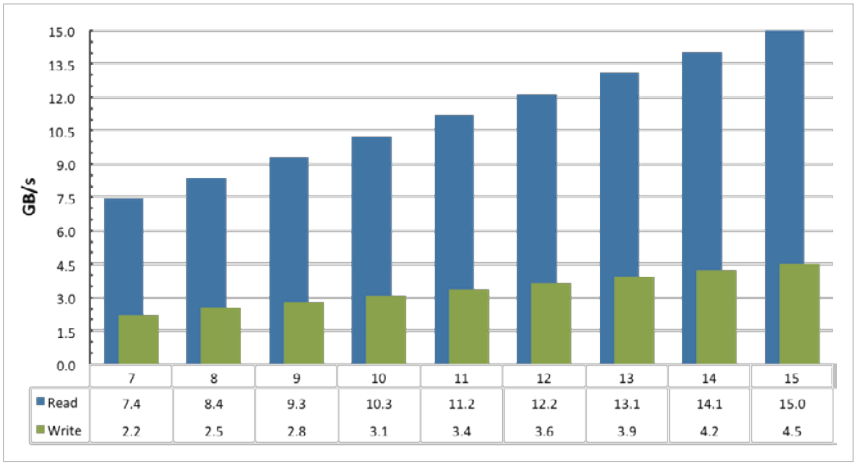


Figure 2. FlashBlade linear scale

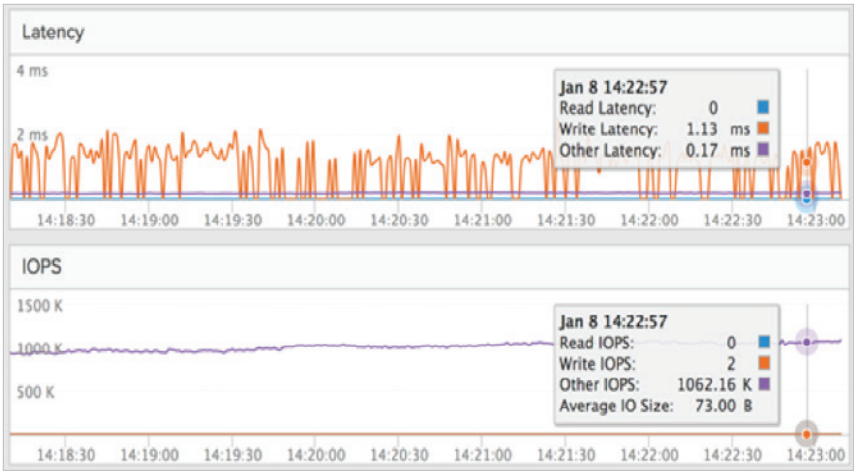


Figure 3. FlashBlade consistent latency and IOPs

MANAGEABILITY

Purity Operating Environment, the heart of every Pure Storage array, implements SW-defined storage services and APIs, advanced data services, and global flash management – all built-in and included with every array. Purity enables storage that is effortless, efficient, and evergreen.

EFFORTLESS

Pure Storage all-flash starts with unwavering reliability. That means your data is always-on, always-fast, and always-secure. Pure solutions are ultra-reliable and plug-n-play simple, with cloud-based management, machine learning predictive analytics, and unrivaled support and protection. The end result is storage that practically manages itself while fixing potential issues before they become problems.

EFFICIENT

Pure Storage all-flash supports in-line data reduction with average reduction that's 3:1 to 5:1 across the installed base and typically 2x better than the competition. Consolidate all your workloads safely with consistent mixed workload performance even through failures and upgrades, and get all your data services built-in and without performance penalty. Integrate and automate everything, seamlessly.

EVERGREEN

Deploy once and keep expanding and improving storage performance, capacity, density, and/or features for 10 years or more – without downtime, performance impact, or data migrations. Evergreen® Storage means no forklift upgrades or data migrations, and you won't need to rebuy TBs you already own.

PERFORCE BENCHMARKS

Perforce benchmarks were used on all testing across Pure storage data platforms. Perforce Consulting ran the benchmarks against Pure Storage FlashBlade and FlashArray configurations. The following Perforce Benchmarks were used:

- Branch submit
- Sync

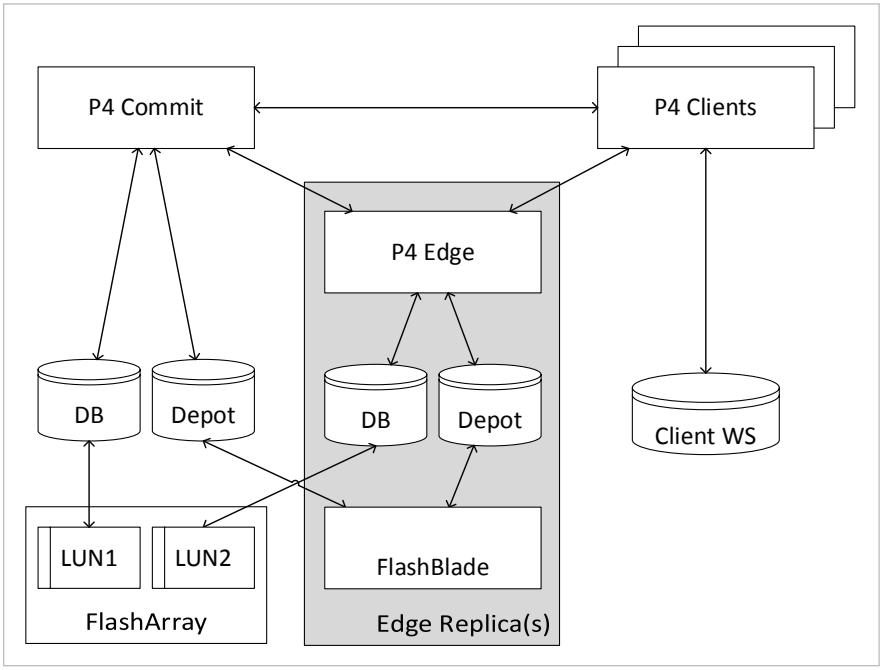


Figure 4. Perforce test setup

BRANCHSUBMIT

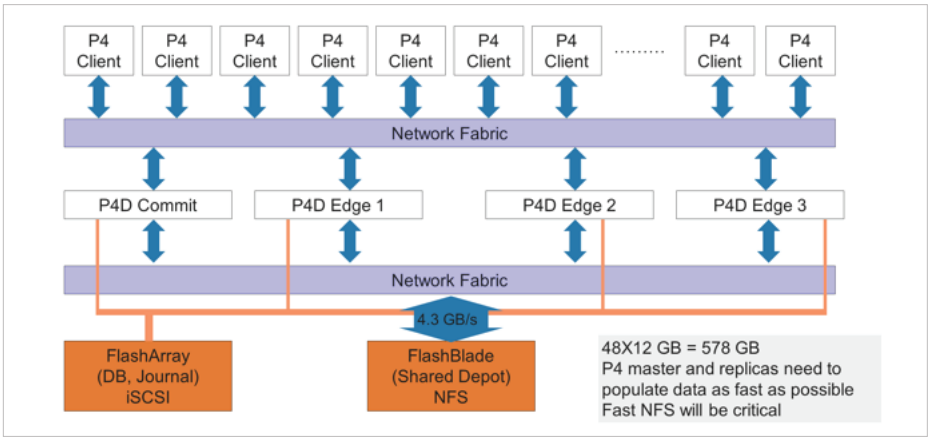
The BranchSubmit benchmark measures, among other things, the rate at which the Perforce Server can commit files to the Perforce metadata. This is an important operation to optimize. Enforcing changelist atomicity requires that the Perforce Server take exclusive locks on various tables while files are committed. Optimizing this operation will improve the overall responsiveness of the Perforce Server.

USING REPLICATION TO IMPROVE PERFORMANCE

It is fairly easy to saturate a 10Gbps link to a single Perforce server, and it is usually this network interface which is the bottleneck in overall performance.

By splitting client processes between a commit and one or more edge servers, we can improve overall performance because the work is distributed between multiple servers – and thus network contention is reduced.

SCENARIO: 48 CLIENTS SYNCING 12 GB EACH



Figures 5. Replication testing scenario

TEST ENVIRONMENT

SCENARIOS TESTED FOR PARALLEL SYNCS

We tested the following:

Scenario	Use Case	DB	Depot
1	Commit server only	FA	FA
2	Commit server only	FA	FB
3	Commit server + one edge server	FA	Shared FB
4	Commit Server + three edge servers	FA	Shared FB

Note that the Perforce clients were configured such that sync data was received over the network from the relevant commit/edge server but not actually written to local disk. This avoids any impact on the overall time in which the server is waiting for the client process to write the data to disk. (See below for details.)

PERFORCE HELIX SERVER VERSIONS TESTED

The Server version tested was: P4D/LINUX26X86_64/2016.2/1468155 (2016/11/28).

This was the latest current release of the server at the time of testing, and the results are broadly comparable to other releases back to 2014.1. Releases before 2013.3 do not have lockless reads, and thus concurrent database access will not perform as well.

Regarding performance, 2016.2 has improvements for replication between servers over the previous release.

PERFORCE DATA SET

The servers were all configured with 188GB of RAM, ensuring that more depot (versioned files) are synced to client workspaces, so that filesystem cache effects are not a factor (e.x., if a depot dataset of ~12GB is synced by multiple clients, all of the depot dataset is effectively cached by the first client). In addition, caches were dropped prior to each run.

The base dataset is 480GB of depot data files. The files are split into 40 buckets, each with roughly 55k files and 12GB in total. Client workspaces each sync a percentage of a single bucket (typically 50%), randomly selected from the 40 possible buckets.

These are a mixture of text and binary files, and they are stored uncompressed on the server to take advantage of FA/FB auto-compression.

PURE HW CONFIGURATION

The following storage technologies were benchmarked:

- **FlashArray (FA)**
  - Connected as iSCSI (block devices)
  - 1 x 10Gb/s link
  - Auto compression and data encryption
  - Auto de-duplication
- **FlashBlade (FB)**
  - Connected using NFS
  - ~15 GB/s (over multiple NICs)
  - Auto compression and data encryption

BENCHMARK RESULTS

The results of running the standard Perforce BranchSubmit benchmark are shown below. This tests performance of the DB filesystem only (it does not touch the depot filesystem).

BRANCHSUBMIT RESULTS

STANDARD PERFORCE BRANCH SUBMIT BENCHMARK

Both DB and depot files on same drives

	Branch Compute Time	Files Opened	Submit Elapsed Time	Submit Commit Time	Percent	Submit Commit Rate
SSD	2,973	70,000	8	1,462	100%	47,879
FlashArray	2,890	70,000	5	1,527	104%	45,841
FlashBlade	3,143	70,000	47	25,089	1716%	2,790

As we can see from last 2 columns, FA and SSD are similar in performance with FB less than 10% of the performance (this is to be expected as NFS protocol is not recommended by Perforce for DB files).

SYNC RESULTS

Results are for 48 sync commands, include parallel threads enabled, and with clients not writing to local storage.

Test Run ID	DB / Depot storage	Notes	Avg Time (secs)	Avg Sent (MB)	Rate (MB/s)	Rate Comp	FB Max Data Transfer GB/s	FB Data Comp
105	FlashArray/ FlashArray	Commit only	250	22,207	89	112%	N/A	N/A
96	FlashArray/ FlashBlade	Commit only	278	22,053	79	100%	1.1	100%
97	FlashArray/ FlashBlade	Commit + 1 edge	162	22,810	141	178%	2.1	191%
102	FlashArray/ FlashBlade	Commit + 3 edge	92	21,928	238	300%	4.1	373%

SUMMARY OF RESULTS

- FlashArray for Database files is only 2-5% slower than using SSD.
- FlashBlade has multiple network interfaces and can support the sharing of depot (versioned) files between multiple Perforce servers (commit and edge) to increase overall throughput. We saw fairly linear behavior when increasing the number of edge servers.
- Increasing the number of edge servers increased overall throughput. Going from 1 edge to 3 edge servers doubled data transfer rates.
- The bottleneck for concurrent syncs of fair-sized datasets (e.g. between 10 and 80 users syncing 10-20GB each) is the network bandwidth between the server and client machines.

BEST PRACTICE DEPLOYMENT OPTIONS

RECOMMENDED ARCHITECTURES

The results show two architectures/topologies which perform well:

- Everything (database, journals, logs and depot files) on a single FlashArray
- Database, journals and logs on FlashArray and depot files on FlashBlade, with multiple replicas sharing those depot files.

FLASHARRAY ONLY

This configuration performs well for small to medium size installations (at a single site):

- Everything (database, journals, logs and depot files) on a single FlashArray – a single instance is easy to manage
- The resulting sync performance is sufficient for the requirements (number of clients syncing in parallel and size of datasets being synced)

FLASHARRAY + FLASHBLADE

This configuration has greater flexibility to support a higher volume of parallel syncs due to the greater network bandwidth of the FlashBlade technology, and the relatively linear overall sync rates achievable by increasing the number of edge servers from 1 to 3 in our tests.

The database files for the commit and edge servers should be on FlashArray and the depot files should be on a single FlashBlade shared between the commit and edge servers.

CONCLUSION

There are two preferred options to consider when choosing a Pure Storage Data Platform with Perforce Helix, whether the customer operates within a small development environment or a large, multi-product continuous integration workflow.

For Helix installations with large environments where large amounts of data are synced in parallel, the FlashArray and FlashBlade technology offer excellent performance. The tested configuration showed a linear increase from a single Commit (master) server, to a Commit plus 3 Edge servers all sharing the same FlashBlade for storage of depot files.

For small to medium size organizations, a FlashArray only, single-site configuration is easy to manage and will offer suitable sync performance across workflows.

Pure Storage all-flash storage solutions can accelerate next generation product designs with their high-performance, easy to manage platform, and highly resilient availability, no matter the size or scope of the organization's product development.

If you have questions on how to best implement all-flash storage alongside secure versioning in your development environment, please contact the consulting teams at Perforce Helix or Pure Storage to help identify your ideal system configurations.

AUTHORS

Robert Cowham, Perforce Consulting

John Wiser, Pure Storage Vertical Solution Marketing

PERFORCE CONFIGURATION DETAILS

BASIC CONFIGURATION

The server instances were set up using SDP (Server Deployment Package) and run on ports such as 3666 and 4666.

During testing, 4 combinations of DB/depot file storage were tested, but in practice it became clear that only 2 were relevant: FA/FA and FA/FB, respectively.

PERFORCE CONFIGURABLES

From “p4 configure show” we get the values shown below.

These variables reflect tracing information which is logged to P4LOG and used for analysis:

```
rpc: 1 (configure)
server: 3 (configure)
track: 1 (configure)
rpl: 1 (configure)
```

These variables affect performance of sync and other commands:

```
db.reorg.disable: 1 (configure)
Usually set for SSD/FA type filesystems
filesys.bufsize: 1048576 (configure)
1M value. Size of buffer for read/write operations on client
lbr.bufsize: 1048576 (configure)
1M. Size of buffer for read/write operations on server
lbr.verify.out: 0 (configure)
Avoids MD5 checksum validation from server to client.
net.parallel.max: 10 (configure)
Max number of parallel sync threads (0 disables)
net.parallel.threads: 10 (configure)
Default parallel sync threads to use. Can be overridden by client.
net.tcpsize: 1048576 (configure)
1M. TCP send/receive buffer size.
net.backlog: 2048 (configure)
Max length of queue for pending connections.
```

To detect any replication errors we also have:

```
rpl.checksum.auto: 1 (configure)
```

The above are documented via “p4 help configurables” and in the **Command Reference Guide**.

An undocumented configurable was used in the P4CONFIG files on the client machines to avoid actually writing synced files to disk (which is often the cause of delays and was shown to reduce the performance of the benchmarks in this instance, as workspaces were written to a fileshare mounted via NFS).

```
filesys.client.nullsync=1
```

RUNNING THE BENCHMARKS

Full details of the scripts used to run the benchmarks are available in the Perforce Public Workshop.

The main components are:

- **Ansible** is used for:
  - Installation of python, p4python, p4 and p4d on various machines, as well as required Python packages (numpy, jupyter, etc)
  - Copying scripts and other files to both replica and client machines to run the benchmarks
  - Running scripts as part of the benchmark including monitoring of network usage (via nethogs package)
  - Running locust slaves which generate usage
  - Copying log files back to the master server for analysis
- **Locust** – a python based performance load test framework (originally written for web applications, but very customizable), for which customized scripts have been written to perform Perforce sync actions (p4_bench.py)
- **Log2sql.py** – a python based Perforce log analyser which parses P4LOG output from the commit and edge servers and creates a Sqlite database. Straight forward SQL statements can be used to produce analysis of the results, such as size of data synced, start and end times of sync commands, etc.
- **(Optional) IPython and Jupyter notebooks** (with Altair graphics package) installed for reporting from the Sqlite databases.

The commit/edge server instances are setup using the SDP, and configured using standard Perforce configuration techniques for replication.

By using the configurable “server.depot.root” and using soft links in file systems, it is easy to configure the 3 edge servers to share the same FlashBlade directory for depot files.

For ease of analysis, the server log files are stored on shared NFS storage which the analyse.sh script can access as part of the analysis of each benchmark run. The performance overhead of writing to the P4LOG files is not significant (keeping the P4JOURNAL files on NFS did incur a significant overhead).

SHARING DEPOT FILES BETWEEN MASTER (COMMIT) AND EDGE

This is required for the shared FlashBlade depot files configuration.

From Perforce docs:

https://www.perforce.com/perforce/doc.current/manuals/p4dist/chapter.distributed.html#distributed.managing.promoting_shelves.auto

GENERATING TEST DATA

A python script was written to generate a random selection of a configurable number of files at a configurable number of levels, and text or binary.

Simple Bash scripts could then check in the resulting files to the commit server at the start of test cycles.

Details are described in the Perforce Public Workshop.

ps_wp_perforce-best-practice-guide_01

PURESTORAGE®

Pure Storage, Inc.
Twitter: @purestorage
www.purestorage.com

650 Castro Street, Suite #260
Mountain View, CA 94041

T: 650-290-6088
F: 650-625-9667

Sales: sales@purestorage.com