Perforce Disaster Recovery at Google

Plans and Experiences

Rick Wright, Google, Inc.
rickw@google.com

Abstract

Source control systems such as Perforce are designed to protect a company's source code, documentation and intellectual property. However, it is also important that Perforce itself is protected against catastrophe. At Google, we have put into place hardware, infrastructure, plans and procedures to protect against disasters both small and large. While these plans and procedures are not perfect, they are continually being improved and tested in an effort to protect against all foreseeable threats to Perforce.

Introduction

Perforce is designed to protect a company's source code, documentation and intellectual property. Google has a very large, very active main Perforce server and several additional smaller servers. The Perforce team at Google is responsible for keeping this server running with high availability and reliability. Keeping Perforce available means putting plans, infrastructure and procedures in place to deal with unforeseen circumstances and disasters that can take the Perforce server out of service. Disasters can come in many shapes and sizes, and over the last few years, we have been continually improving our processes and infrastructure for dealing with those disasters. The initial submit of our failover procedure was over 3 years ago and the document is now at revision #43 as we have continued to improve and update our plans.

Keeping Perforce data protected and available means building redundancy into the system at nearly every level. Google has redundant Perforce hardware, redundant copies of the Perforce metadata and redundant copies of the Perforce versioned files. Google is even fortunate enough to have Perforce expertise outside of the main headquarters which should allow Perforce at Google to continue to operate even if offices in California are unavailable.

In addition to the redundancy built into the system, we regularly test and practice our procedures. While some of the things discussed in this paper will only apply specifically to Google, many of these concepts can be used at Perforce installations of any size. Protecting data and making disaster recovery a priority is something that should be in place at all levels.

Perforce at Google

Google's main Perforce server is very large, with approximately 600GB of metadata and more than 8000 active users. The main Perforce server recently passed changelist 10,000,000. In addition to the main server, we have 10 smaller servers, 6 read-only replicas of the main server, and 2 standbys for each of the Perforce instances. We have more than 60 proxies worldwide. The hardware for the main server has 4 Quad-core processors and 256GB of RAM. The metadata (db.* files) is stored on a RAM-based storage system (with hard drive backup) with about 900GB of total storage space. The versioned files are stored on a network attached storage system.

The Perforce team at Google is a group of 8 people who manage and support the Perforce server directly with

several additional people closely supporting the Perforce team including system administration and hardware roles. As of the time of this writing, Google is running the 2008.1 version of the Perforce server with plans to upgrade to 2008.2 during the second quarter of 2009. In general, upgrades are done 3-5 months after a new Perforce release is made generally available.

Disaster Planning Overview

In order to plan for a disaster, it it important to first ask the question: "What is a disaster?" Disasters can take various forms and can have both large and small impacts. Some examples of disasters include simple hardware failure and database corruption to larger-scale disasters including things like major network disruptions, localized power outages and fires, to natural disasters that can incapacitate an entire region. Disasters can affect a single machine, a building or an entire site. In our disaster plan, we have attempted to anticipate all of these different types of disasters.

A key component of this plan is redundancy. Google's Perforce infrastructure includes three sets of nearly identical hardware. One set of hardware runs the active Perforce server, a second set is a local standby machine in California and the third is a remote standby machine on the East coast of the United States. The metadata is replicated to both the local and remote standby machines with a 1-20 minute delay, and the versioned files are stored on a network attached storage system and mirrored to additional NAS devices, one local and one remote. We attempt to eliminate all single points of failure. For example, the local standby machine for each Perforce instance is in a different building from the active Perforce server, even though they are at the same general site.

To prepare for disasters, we have a plan that gets regularly used and updated. We test out our plan every quarter by failing over to our standby machines and additionally test our plan by doing an annual failover to our remote standby.

Data Protection

The first step of being able to successfully recover from a disaster is making sure that the data is protected. At Google, we protect our Perforce data in various ways. Perforce has two distinct sets of data. The first is the metadata, which is stored in the db.* files on the server. The second are the versioned files, which are stored as text and binary files and include the actual contents of the files stored on the Perforce server. Different approaches are taken to protect both types of data.

Protecting the metadata

The most obvious way to protect the metadata on the server is through the Perforce checkpoint. At Google, we do a checkpoint of the database every Saturday morning. This is done with an automated backup script that currently takes about 8.5 hours to run. To allow Perforce to keep running with minimal downtime, we create an LVM snapshot and checkpoint the snapshot. This allows us to do the checkpoint with less than 5 minutes of server downtime each week. The weekly checkpoints and journals are backed up to a network attached storage system which is mirrored to two other locations.

The metadata is also replicated to the standby servers. To do this, we use an internal tool called 'g4jrep'. This script is based on 'p4jrep' which is available in the Perforce public depot. Functionally, g4jrep and p4jrep are very similar. The primary difference between the two is that while p4jrep is a single process that runs on the standby machine, g4jrep has a client-server architecture with one process reading the journal on the main server while another process writes those changes to the standby.

Replication to the local standby server lags behind the main server by about a minute (or less), while replication to

the remote standby server lags behind the main server by 10-20 minutes. The lag to the remote standby is intentional, since the replication of the versioned files to the remote site lags by a similar amount (as described below). One of the additional features that we added to g4jrep is the ability to delay writes to the standby based on the replication delay of the versioned files. In addition, g4jrep also creates a local copy of the journal on the standby servers while it updates the database on the standby. This has the side effect of giving us a full backup of the current journal in two additional locations.

The journal itself is protected in a couple of ways. First, it is on a different volume than the db.* files on the Perforce server, so that in the event either filesystem is lost, there is redundancy between the journal and database. Secondly, as mentioned above, the journal is replicated to the standby machines along with the database. Before g4jrep had this ability, we had a cron job that gzipped the journal file hourly and copied it to a remote site.

As a very last resort, we also have regular tape backups of the metadata. We don't expect to ever need to use the tape backup to recover from a disaster.

Protecting the versioned files

The versioned files are also protected in several ways. The versioned files are stored on a network attached storage system with snapshots turned on. The snapshots have never been needed, but protect against accidental (or malicious) file deletion. The data on the NAS is also mirrored to two locations. It is mirrored synchronously to another local NAS, and it is also mirrored asynchronously to a remote NAS with a 10-20 minute delay. As with the metadata, the versioned files are backed up to tape as a last resort. Again, we never expect to need to use the tape backup, but we have tested our ability to restore files from tape.

Procedures

In addition to the infrastructure for disaster planning, Google has also developed a set of procedures that we follow before, during and after disasters. These procedures include a quarterly failover, an annual disaster recovery test, regular checkpoint restores, communication procedures and scripts to monitor the health of the system.

Quarterly failover

Once a quarter, we do a failover from our current server to our local standby. This process makes the standby server our new master server and makes the master server the standby. This task gets rotated among the Perforce administrators each quarter so that we all have recent experience executing the plan. During a failover, we follow a documented procedure that gets updated every time it is used. This document is checked into Perforce and also synced to several known locations so that it can be found if Perforce is down.

The failover document contains step-by-step instructions for the entire failover. It is designed so that when a pager goes off at 2:00AM, it can be easily followed by someone who has been woken up and wasn't expecting to have to fail over the server. Each step of the document includes instructions that can be copied and pasted directly into a terminal session. For example, instead of simply stating "Start Perforce" in the document, it says:

```
Start Perforce server:
  N=1
  /etc/init.d/p4d_${N} start
```

Note that the document is designed to be used with any of the Perforce servers and the commands can be customized just by setting "N" to a different value in your environment. All of the Perforce servers at Google are

identified by an instance number, and nearly every command or path specific to that instance is made unique by that number. (e.g. /p4_1/root vs. /p4_2/root for the main server and the "Perforce2" server respectively.)

The quarterly failover requires less than 30 minutes of downtime for the main server and less than an hour of downtime if doing a failover of multiple servers simultaneously. We usually do scheduled failovers on Saturday nights, following the checkpoint that runs early Saturday morning. We also use the quarterly failover as an opportunity to rebuild the database from checkpoint in order to free up disk space and rebalance the B-trees as described in the Perforce System Administrator's Guide.

Annual disaster recovery test

Once a year, we participate in a disaster recovery test. This differs from the quarterly failover in a number of ways. First, during the disaster recovery test, we fail over to our remote location and run Perforce from that site for a few days. Second, during the test, Perforce is supported and maintained completely by the Perforce administrators that are not located on the main Google campus in Mountain View, CA. Currently, this includes two administrators: One in Seattle, WA and one in Hamburg, Germany. Finally, during the test, the remote Perforce administrators are only given limited information about what will happen during those days, and may be required to fail over one or all of the servers with only a few minutes or a few hours of notice.

Communication procedures

For all major outages (and most minor ones), we communicate regularly with our users. We have an auto-generated mailing list that includes all Perforce users, and for scheduled outages, we generally try to give the users a 5 day warning, plus a reminder just before the outage and a follow-up email after the outage. In addition, on our internal website, we have a small "status" area that can be quickly updated with any outage information (both scheduled and unscheduled) and estimates about when the system will be back.

During any outage or maintenance, the Perforce administrators communicate in an internal IRC channel. This serves as a central meeting point for all of the administrators, provides step-by-step updates to the team about status, and serves as documentation about what happened during the outage. Every outage or maintenance procedure has a single administrator who is in charge (generally the person who is on-call that week) and this person can delegate tasks (such as end-user communication) to other administrators who are there assisting. In addition, other teams that help maintain the Perforce server also monitor and regularly join this IRC channel.

Not all disasters, outages and scheduled maintenance can be performed solely by the Perforce team. Often, we require assistance from other groups (such as the group that maintains the NAS), so we have documentation that lists how to contact those groups when necessary.

Checks and Monitoring

Each week, we restore that week's checkpoint and run "p4d -xv -vdb=2" and "p4d -xx" to watch for issues and changes. In addition, we also occasionally run "p4d -xu" if we are testing a new p4d version. We also have several scripts to monitor the health of the servers and notify us when there are issues. The monitoring scripts include fairly obvious ones like checking connectivity to the server, monitoring disk space and making sure that Perforce machines respond to 'ping'. We also have scripts that monitor whether the replication script is lagging, the number of p4d processes on the server, disk space growth and a check to make sure that the weekly backup will likely be successful.

Real disasters

As much as we try to simulate possible disaster scenarios, real disasters have some definite differences from simulated disasters. During all of our simulated disasters, we never allow for any data loss. In a real disaster, data loss is a real possibility, particularly in a site-wide or regional disaster. As mentioned earlier, our remote site lags behind the master Perforce server by up to 20 minutes and it is possible that we would lose up to 20 minutes of data in that event (200-300 submitted changelists at peak usage). We do not currently have a defined procedure to deal with this data loss, other than notifying users and having them resubmit changes from the local copies stored on their machines. We do, however, have a step in our failover document that instructs us to roll the change counter forward in this circumstance by 500-1000 changes to prevent conflicting changelist numbers between new and "lost" changes.

How to apply these principles

Every company is different and the plans and procedures that we have developed at Google may not apply to everyone. However, there are several things that all disaster recovery plans have in common. The first, and most important is "Have a plan." This seems obvious, but no matter how detailed or vague, it is important to have at least a basic plan for what to do when things go wrong. When determining what that plan should be, you should ask yourself some questions:

- What is my risk tolerance?
- How long can we afford to be down?
- How long does it take to restore from checkpoint?
- How long does it take to restore from a tape backup?
- How much data can we afford to lose? An hour? A day? A week?

The answers to these questions will form the basis for your plan. At Google, it takes about 3 hours to recover from a weekly checkpoint, and can take even longer to recover from the 60GB-70GB (occasionally more) of journal that can exist near the end of the week. This means that a restore from checkpoint & journal could take nearly 10 hours, and we have decided that that is too much of an outage for us and have set up standby machines that are kept continually up-to-date. At smaller sites, where a checkpoint & journal recovery can be done in less than an hour, it may not be as important.

Once the plan exists, the next most important thing is "Do it once." No plan is ready until it has been tried at least once. Even after doing failovers every quarter for the last 3 years or so, we still find minor corrections that need to be made to our failover document. Ideally, any plan should be practiced regularly and improved each time, but even without that, it needs to be done at least once.

Conclusion

It is important to develop a plan to protect Perforce from all disasters both large and small. In order to do this effectively, you must decide that it is important, have a plan and practice that plan. At Google, our plans, infrastructure and processes are continually improving, and while we hope to never need most of our plans, we are confident in our ability to survive nearly every foreseeable disaster.

References

Demystifying Perforce Backup and Near-Real-Time Replication, Richard Baum
http://www.perforce.com/perforce/conferences/eu/2008/presentations/perforce_richard_baum_whitepaper.pdf