Developing and Maintaining a Strategic Perforce Plan at Google

Geoff Mendal, Senior Tools Strategist

mendal@google.com

Google, Inc.

Synopsis

This paper describes the tenets of the Google strategic plan for Perforce in a manner that should be usable by any Perforce customer, regardless of size. Not every aspect of the Google plan will be appropriate for other Perforce customers, but that is to be expected. The point of the paper is to get the reader thinking about what aspects he or she does require and approaches one can consider when customizing them for your company and customers.

Five years ago Google made the jump from a volunteer engineering development team supporting Perforce company-wide to a full-time P4 Admin operations team. The sustained growth of the company and thus its Perforce use increased at a rate not often seen in technology companies (at least, not by this author). Amongst the challenges facing the newly formed P4 Admin team was how to manage the explosive growth of Perforce use within the company. Pain points were everywhere, and it was all the team could do to whack-a-mole the issues as they arose. The team was in a pure reactionary mode and battled to tread water.

It was clear to everyone that a different approach would be needed if Perforce was to succeed at Google for the long-term. Developing and maintaining a strategic plan for Perforce use at Google was one of the new approaches undertaken. On its face, it would seem obvious that every company – especially one the size of Google – should have such a plan in place, and that the tenets of the plan should be fairly obvious. But the reality is that is often not the case. A strategic plan alone just sets a stake in the ground: its existence alone will likely not cause or promote positive change in a company. But every journey begins with a first step, and a strategic plan can help assure that the first step and subsequent ones are likely headed in the right direction.

The Strategic Perforce Plan at Google has many benefits including:

- shaping periodic discussions with upper management over needed resources and anticipated spending,
- serving as a vision of Perforce use in the future,
- providing a roadmap for the P4 Admin and supporting teams to follow, and
- setting both medium and long-term engineering priorities, e.g., helping answer questions of the type, "Why are we doing *this*?"

Overall tenets of the Google Plan

You cannot have a plan if you don't write it down, vet it with owners who share responsibility for the P4 service, and commit to keeping the plan up to date. **The only thing worse than having no plan at all is having a plan that does not reflect reality.** At Google the plan is reviewed at a minimum once a quarter, usually dovetailing with the setting of the company and team's quarterly goals and objectives.

Multiple groups in the company have overlapping responsibilities for the P4 service. Segregation boundaries exist for the service itself, the platforms on which it runs, and the underlying hardware and machine rooms from which it runs. Cooperation amongst the various groups is essential for optimal operation of the P4 service. Owners and point-persons in each group remain in close contact when setting quarterly and yearly goals and objectives. For example, upgrading the service to a newer release is a more involved operation than just shutting down the service, invoking `p4d -xu`, and restarting the server with the newer binary. At Google a p4d binary goes through an extensive burn-in test cycle, the sequence in which various P4 servers are upgraded is highly orchestrated, upgrade timeframes are announced well in advance and mutually agreed upon by all groups, and the implications on downstream infrastructure are carefully vetted (see the section *P4 Release Upgrades* below for details). This is not to say that Google gets it perfect every time, quite the contrary. But Google strives to not make the same mistake twice and if/when our procedures and processes fail, we document those failures, modify the processes, and move forward with a more robust plan.

Assuring scalability of the P4 service is one of the most important tenets of the plan. Google takes a long-term approach to managing scalability of its key infrastructure services. The P4 strategic plan therefore projects out multiple years of world-wide use in order to better understand what new hardware, platforms, and dependent infrastructure may be required to keep the service running efficiently. Source code growth rates are monitored and trended in a number of ways in order to better predict future needs.

Google has engineering offices located world-wide. Approximately half of our P4 use emanates from company headquarters in Mountain View, CA where our primary production P4 servers are presently located. Understanding use patterns and projecting growth in each region of the world is an important tenant of the plan. For example, when Google recently relocated infrastructure services from a machine room from one area of the country to another, the implications on the P4 service were material, requiring advance planning in order to have confidence that the shift in load could be handled without issue. Monitoring and trending both local and remote P4 performance are critical to understanding where architectural issues may arise, for example, for remote users network bandwidth and P4 proxy performance are often more significant factors than P4 server performance.

The company prides itself in its “go anywhere” approach to both product development and engineering resources. With very few exceptions, Google engineers are expected and encouraged to share code with every other engineer in the company. Therefore having a single, centralized depot will not scale for all users world-wide. The P4 service at Google has to account for several types of replication in order to deliver acceptable performance to both local and remote users. In other words, deploying a P4 proxy in a remote office is often not good enough. The Google P4 strategic plan includes internally developed replication technologies as well as P4 proxy deployment.

At the scale and size of Google, management expects and often encourages core infrastructure teams to purchase needed equipment to keep operations humming years into the future. The problem at Google is not having the budget for or obtaining authorization to purchase equipment, but rather making certain that what we do purchase will do the job and scale for the duration of the equipment’s useful life if not well beyond. The cost in lost engineering productivity of a single unanticipated outage or extended period of poor P4 performance is known to usurp the cost of the hardware on which the P4 service runs (and we run on extremely expensive hardware). So at Google, we make our purchasing decisions not on what will minimally get us by today, but what will keep us sailing smoothly multiple years into the future. This is not only for the production P4 servers themselves, but also for our P4 test servers, proxies, replicas, and network attached storage. When purchasing top-of-the-line hardware, management really appreciates having a document that justifies the expenditure. Our strategic plan is one of those documents.

Google’s product development methodology includes a plethora of build and test systems that require P4 resources. So it is usually not sufficient to simply count live engineers and budget our needs using that count. Automated systems that query the P4 service put as much if not more load on P4 than engineers themselves. Our planning processes therefore need to take into account the growth needs of automated systems in addition to those of the users themselves.

As mentioned earlier, Google’s greatest challenge in maintaining the P4 service is assuring that it will scale *infinitely*. Out of the box, it is our belief that the P4 product scales better than any other commercial SCM product. Nevertheless, that is not good enough to scale to Google Engineering’s needs. So we complement Perforce Software provided solutions – which are often bundled with the semi-annual product releases – with our own internal technology. Planning for scalability is therefore more complex than just reading through the release notes of the upcoming P4 product release. Because we augment Perforce to such a great degree with our own internally developed technology (example: replication), each Perforce Software release has its own unique challenges to overcome. The P4 strategic plan therefore needs to account for changes to downstream infrastructure dependent on P4.

As mentioned above, an outage or slowdown of any significant period of time will incur a tremendous cost in terms of lost engineering productivity. To reduce the chance of such an outage as well as to better secure the life blood data (our source code) on which the company operates, Google has chosen to spend whatever is necessary on disaster recovery and sustained operational support of the P4 service. For our purpose, that means we have at the ready multiple, identical sets of hardware and infrastructure needed to run the P4 service in production mode in both local and remote locations for an extended period of time. These additional servers are not the only consideration of the strategic plan however. Having a team that can run them is also a primary concern. Should a disaster knock out the ability of the existing P4 Admin team or any of its support teams to operate the P4 service, we have in place remote P4 Admins and support teams that are able to take over operational duties. Google performs an annual company world-wide week-long disaster recovery exercise comprising most of its infrastructure. Each year we raise the bar, challenging ourselves with ever more difficult disaster recovery (DR) scenarios (which are not known ahead of time to any of the participants). How well each team performs is documented and each team takes that feedback to improve its operational readiness going forward. The P4 Admin team has participated every year, and integrates the lessons learned into its operational readiness going forward. The important takeaway here is that it is not enough to have a DR plan, one must also exercise it to have confidence that it will work if/when needed.

As recent security issues have been brought to light in the press, it is imperative that critical data be secured, monitored, and audited. The P4 Admin team works closely with the internal security operations team to assure that the P4 service is properly configured. In addition, P4 logs are regularly scanned and audited for irregularities. Company policies dictating access to the P4 service are checked by automated tools which look for nonconformance. SOX and other external auditing processes occur on a regular basis. No issue is deemed too small to be ignored. For example, the company has a policy that P4 submissions be reviewed by a second person and approved before being submitted. In addition to P4 triggers which catch inadvertent attempts to circumvent this policy, auditors conduct their own analysis and follow-up as needed when violations are detected. One of the benefits of having a security policy and adhering to it is that all of the work done to analyze the logs is available for normal monitoring of the P4 service and long-term trend analysis. Mining the P4 logs for usage greatly helps when predicting future growth needs of the service. Google wrote its own log analysis tools years ago but today P4 customers can make use of a comparable package supplied by Perforce Software so it is not necessary to start from scratch.

First Things First, Getting Your Feet Wet

At first, it may seem that committing to developing a P4 strategic plan is a daunting task with little if any immediate ROI. In the wake of more pressing tactical issues, prioritizing the writing and adoption of such a plan is not an easy decision. This was certainly the case at Google. But every journey begins with a first step and as we have learned, having a plan is better than none at all, as long as the commitment to maintain and improve it going forward exists. Here are some suggestions for where to begin when developing a P4 strategic plan:

- Document the current P4 architecture.

  At Google, we maintain a blueprint of our P4 servers, proxies, clients, depot file servers, replication servers, hot standbys, DR sites, networks, and dedicated switches. In addition, the configuration of every server running a P4 service is checked into P4 itself and kept up to date by a Site Reliability Engineering team that supports the P4 service.

- Benchmark existing performance and scalability.

  At Google we have customized the benchmarking tools developed by Perforce Software itself as well as written many of our own internal tools to do the same. Having comparable data helps to justify decisions on what type of hardware to purchase as well as how to optimally configure the P4 service. For those who are just getting started, Perforce Software has available several configurable settings that help to get the most out of the hardware and platform you currently have. In addition, Perforce has several consulting partners who have expertise in this area and can either make suggestions for how to proceed or can offer ready-made solutions that may quickly improve P4 performance at your company.

  The issue that Google struggles with often is what exactly we should benchmark. Identifying the "typical" user experience is a challenging task, since there is no typical use pattern in the company. For some users, performance may be measured by a simple `p4 edit` of a single file while for other users it may involve the speed by which a complex `p4 integrate` operation takes place. Still for others, it may simply be how long it takes to sync a client to the head revision in the depot. Performance can and often does diverge based on how loaded the service is, how saturated the network may be at the time the operation is invoked, and several other external factors. In our experience over the past 5 years, the most reliable and accepted server health indicator is the wall clock time it takes for a single file `p4 edit` operation to complete.

  Another problem Google struggles with is how to prevent falling off the edge of the cliff from a scalability point of view. Google's main Perforce server database is extremely large; we believe one of the largest in terms of database records in the world. It is not uncommon for users to run into imposed system limits such as `MaxLockTime` and `MaxScanRows` while performing seemingly innocent P4 operations, for example, `p4 opened -a` or `p4 changes -u` *`user`*. Customizing the available performance and scalability settings is an ongoing challenge as Google's P4 use continues to grow.

- Capture and report on performance/scalability medium and long-term trends.

  At Google we employ many tools that monitor various aspects of server performance. Some of these are obvious such as current CPU load, RAM usage, and I/O throughput. `p4 monitor show` offers a good starting point for any P4 customer. At Google, we continuously up our game to include more complex and deeper diagnostics, both in terms of real-time server statistics, P4 service statistics, and long-term trends. Having a plethora of data points for both the instantaneous server health as well as the sustained operational load of the P4 service over time allows us to pinpoint potential issues quickly and develop solutions for managing scalability going forward.

Managing P4 (and Google) Releases

As mentioned above, at Google the P4 release upgrade process is quite rigorous. As upgrades are usually a one-way street, we need to make very certain that nothing will break as a result of revving the product. We start by accepting pre-release versions of the P4 release for testing. We have isolated test environments (exact duplicates of the p4 production servers) where catastrophic errors are expected and will not poison the production code base. Release notes and advance documentation from Perforce Software are reviewed, and depending on the nature of the release, we will devote the proper amount of resources to testing core and/or the most risky features of the new release. For example, if a release includes a new feature that we cannot live without or is potentially disruptive to the point that we may need to rule it out (example: `p4 move` in the 2009.1 release), we will test accordingly.

Once Perforce Software releases the GA version of a product, we install it on a test server that trails the main production server. It is not uncommon for us to run a GA release for many months before taking the next step. And again we will pour through the release notes examining anything that may have been inserted into the release at a later stage of the cycle. Tracking new features (including `p4 undoc` ones) and bug fixes are always of interest to us.

Once we and a select group of our power users are satisfied with the quality and stability of a release, we schedule a staged production rollout starting with our smallest and least risky P4 servers. The staged rollout is usually completed within a couple of weeks. The goal is that by the time we upgrade the largest and most risky P4 server (our main server) we have 100% confidence in the release.

One of the consequences of our rollout approach is that we have to manage and tolerate different versions of the product on production servers at the same time. Though we strive for all servers being equally revved, the reality is that there are times when we are not in equilibrium. Usually this is not an issue since Perforce Software strives for backward compatibility, but there can be releases where this is problematic. For this reason, we attempt to dive deep into every release to understand at its core what changes to the database (db.* files) it entails. For example, the new `p4 move` command in the 2009.1 release causes a different sequence of journal records to be written and slightly different output for queries such as `p4 opened` and `p4 filelog -l`. So any tools or downstream infrastructure which assumes that the output of those commands will conform to a known set will have to be modified. Where this is usually most important for us is with our internally developed journal replication system and downstream infrastructure which relies on that replication.

Our usual upgrade process has us revving the P4 servers and replicas first, then proxies, and at a later time of convenience, the client binaries and applications. In cases where a release requires the revving of two of more of these at the same time, we make adjustments to our plan.

In addition to testing releases of Perforce Software products before deploying them company-wide, a large number of internal tools with various levels of dependencies into P4 must also be kept up to date. This is generally not an issue with P4 server releases as Perforce Software assures that older versions of client tools and APIs will continue to function. Still, in order to benefit from the latest features and bug fixes we strive to keep our own internal tools up to date. By the same token, in order to promote stability Google has deployed an API and client wrapper that most internal tools invoke instead of the Perforce Software client products directly. Having our own API and client wrapper provides us flexibility and some measure of control over invocation of new features and changes to the product. Of course, this all comes with a cost: engineering resources are required to develop, test, and maintain our own API and client wrapper.

Short and Medium Term Actions

At the end of the day, the success of P4 at Google boils down to whether the product can continue to scale to fill our constantly expanding needs. Scalability is therefore usually #1 on the P4 Admins hit parade. Over the past five years we have dealt with a large number of issues that affect P4 scalability including:

- Geometric growth of P4 metadata

  The team has dealt with this problem in a number of ways including archival and deletion of old P4 clients to reduce the db.have table size, archival and deletion of old labels to reduce the db.label table size (plus migration to automatic labels), offline obliteration of deleted and unneeded portions of the depot, and migration to a sparse branching strategy to curb the growth rate of db.rev* and db.integed tables. Most of our scalability issues are a result of the rate of growth in our largest db.* tables, and so we focus our actions on those tables.

- Supporting selected `p4 undoc` features

  Google has multiple production P4 servers and in order to manage P4 users more easily, we make use of the undoc'd feature `P4AUTH`. Automatic labels were originally introduced as a `p4 undoc` feature and we began migrating many of our branches over to this feature at that time (the feature has since been officially supported). Our client wrapper utilizes the `p4 -zmax*=n` options for queries where the default values are known to be problematic. We use the `p4p -w` option for some of our proxies. Our consistency checking tools make use of `p4d -xv`, `-xx`, and `-xf 925` options. Our replication tools include the `-f` option for `p4d -jr` (journal replay). With the support and recommendations of Perforce Software, we have begun to set `P4DEBUG=tunables` server parameters to improve overall performance. `P4CHANGE` is likely our next `p4 undoc` feature to address as we move to a model of supporting unique change numbers across all of our production servers.

- Evaluating, planning for, and deploying new hardware

  P4 servers the size of Google require careful consideration of proper hardware. At Google, P4 db.* table lock time winds up being a critical issue, and so deploying hardware that can reduce the time required to lock the tables, perform the needed actions, and unlock the tables will increase overall engineering productivity by a significant factor. Also, due to the critical nature of Perforce within the Google corporate infrastructure, the reliability of the hardware on which the service runs is a paramount concern. For these reasons, Google regularly evaluates new hardware and plans for hardware deployments usually one year in advance.

- Changing development practices (zero in on the "top 10" p4 abuses)

  Mining P4 logs and having established sophisticated monitoring and alerts of the P4 service allows us to identify users who are inadvertently abusing the service. Depending on the nature of their actions, our scripts may automatically kill their activity, we may engage immediately to have them cease their actions, or we may follow up at a later point in time to enquire what they are doing and why. Often times we can suggest alternate queries or point them at replica servers or other Google internal tools to get their job done without placing an undo burden on the P4 service. At times this can feel like a whack-a-mole response, but over time we have been successful in curbing the worst abuses and staying ahead of the curve of ever present newer ones.

Longer Term Actions

In addition to the actions taken above, there are several more complex, invasive, and therefore longer term solutions that can be undertaken to improve scalability and performance of the P4 service at Google. The reason that some of what is listed below has not yet been completed is due to the complexity of implementing it, and the amount of pain each would inflict on our user base versus the amount of pain we currently feel from each of the issues cited. If and when the pain point of an issue reaches a tipping point, we will act (and in some cases as noted below, we have already).

- Protect table

  Google has a multi-thousand line P4 protect table which reflects the nature of how vast the P4 service is used within the company and the fine-grained permissions allotted to various users, groups, and role accounts throughout the world. Having such a complex permissions structure comes with a cost of extra time required to process many queries. Therefore, the team is always on the lookout for ways to trim the table without leaking depot content to those who should not have access to various paths.

- Branching

  As mentioned earlier, sparse branches is the new model for most P4 branches. Though P4 supports this model, at Google the issue is more than just what `p4 integrate` command is invoked to create or maintain a branch. The nature of engineering in the company necessitates the creation and maintenance of many thousands of branches, each of which is managed by a set of downstream infrastructure (build and release tools, etc.). Significant scalability improvements are to be had by converting our existing full branch integration model over to the sparse branch approach.

- Depot path length

  With the help of Perforce Software, we have come to understand that our rather long depot paths are contributing to scalability and performance issues with the P4 server. By shortening the overall path by a factor of two we can obtain vastly improved performance. The problem of course is that the cat is already out of the bag and switching over to a more modest depot path length will incur massive pain in downstream infrastructure.

- Replication and Caching

  Google has a multi-faceted approach to replicating both content and metadata from its various P4 production servers. We replicate the journal to both hot standby (failover) servers and DR servers. Depot content is replicated across network attached storage hardware, cached in P4 proxies world-wide, as well as being cached in internally developed infrastructure. Finally, we cache and replicate P4 pending and submitted change metadata for an internally developed code review system (Mondrian) and source code analysis tools.

- Managing acquisitions and large imports into P4

  On a regular basis, the P4 Admin team finds itself supporting the import of outside source code bases into its P4 servers. Some of these come from acquisitions, others from partnering relationships, and also from third-party sources (example: open source projects). Depending on the size and nature of the source code, the P4 Admin team will suggest an import strategy. Other factors that drive a solution include whether existing history has to be maintained and the to-be-imported SCM system currently in use. Due to the size and scope of the P4 Admin team, we mostly act in an advice and consent role, requiring the development team importing code to drive the process.

- Managing federated (multiple) SCM systems

  Google is a large enough software enterprise that any single SCM system will not satisfy every need. For example, Google is well known for championing open source and hosts several open source repositories – some internal and some external – which are not implemented using Perforce. Additionally, some projects find that Subversion (svn) suits their focused needs quite nicely. And there is a growing internal user community who swear by git. The P4 Admin team is indifferent to all of these uses. In many circumstances, users have gone to painstaking trouble to integrate other SCM systems with Perforce. Perforce is the SCM system of record at Google, but that does not preclude using other tools on the edges, as long as the source code of record is checked into P4 everyone is happy.

- Partner with Perforce Software by being a proactive and engaged customer

  One of the most important and beneficial decisions the P4 Admin team at Google has made was to commit to being a proactive and engaged Perforce customer. This was not the case five years ago: other than routine support calls and an occasional encounter with Perforce Software, we were very much a passive customer. Our interactions with Perforce Software were purely reactionary to issues that emerged in the normal course of using the product. As was stated above, Google came to the conclusion that the success of the product in the company would require us to have more direct interaction with Perforce Software. Our decision was to move from being a passive and reactionary customer to a proactive and engaged one, in effect partnering with Perforce Software for the long-term success of the product. From our vantage point, the positive impact it has had on our company as well as many other enterprise sized Perforce Software customers is evidence that our commitment has been worth the effort expended many times over.

Deploying and Maintaining the Strategic Plan

As is typical and encouraged throughout Google Engineering, nearly all of our P4 documentation is shared online using the company's internal network. This includes the P4 strategic plan. The Plan references our quarterly goals and objectives within a multi-year time horizon. As noted earlier, we review and update the plan alongside our setting of the quarterly team objectives. Having the Plan available for review by senior engineers and engineering management allows us to have deep discussions whenever they deem a project review is warranted (usually on an annual basis).

Obtaining buy-in from customers (developers, testers, release engineers) and our support team (system administrators, machine room and corporate hardware operations, field techs) is strongly desired. As we all share responsibility for the success of the P4 service, having our support teams engaged and aware of our strategy for maintaining and improving the service helps them to help us.

Lastly, as mentioned above, our decision to migrate from a passive customer to one who is partnering with Perforce Software means that we also share the P4 strategic plan with them (under an NDA). Perforce Software reviews the plan annually and this helps to shape discussions we have about our upcoming and most pressing needs. Several critical fixes and enhancements in recent P4 releases were a direct result of our documenting the issues in our Plan and meeting with Perforce Software to discuss. So not only has Google benefited from this approach, but all Perforce Software customers reap the same rewards.