Performance and Database Locking at Large Perforce Sites

Dan Bloch

Google™

Sep. 19, 2006

Google

- Google's mission: Organize the world's information and make it universally accessible and useful.
- The world's premier web search
- Headquartered in Mountain View, California
- ~7000 employees worldwide

Perforce at Google

- More than 3000 users and 100GB of metadata on one primary server
- Also: read-only replicas, proxies, and a few smaller servers
- Hardware is an HP DL585 4-way Opteron with 128GB memory
- Depot is on a NetApp filer
- Metadata and journal on RAID-10 local disk
- Perforce 2005.1, Linux 2.4

Outline

- Overview of Perforce performance issues
- In depth discussion of one specific issue: server hanging due to database lock contention
- Details on some Perforce internals: process model, error log, database files, locking model

Performance in General

- CPU
- Memory
- Network
- Disk access
- Usage patterns

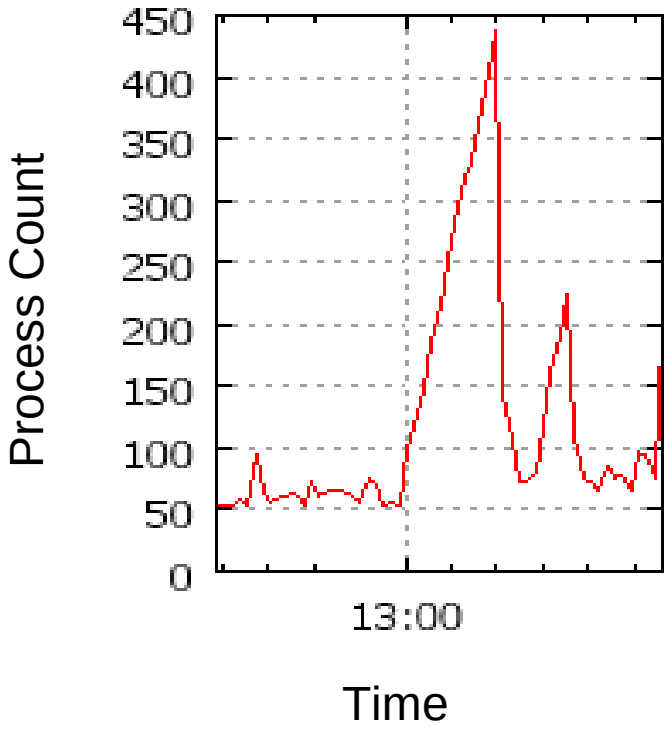
Performance Solutions in General

- Set **`MaxScanRows`**, **`MaxResults`**
- Run big read-only operations on read-only replicas (e.g., **`p4jrep`**)
- Turn off auto-refresh in GUIs
- Monitor what your users are doing (hard). Scripts tend to be biggest users.
- **`strace`** is a valuable diagnostic tool
  - (On Linux. Solaris equivalent is **`truss.`**)
- There is no silver bullet.

The Problem

- Several times a day our Perforce server stops responding to commands.
- To the user, it appears the server is hanging.
- To the administrator, it's visible as a spike in the number of p4d processes.
- Spikes can last for one minute, five minutes, ten minutes, or longer.
- Users and administrators don't know the cause.

A Server Spike



The Explanation

- The problem is always caused by a single Perforce command, not by overall load.
- This command holds one or more database locks, and blocks other commands from running until it completes.
- You can either kill the command (if it's read-only) or wait for it to finish.
- Knowing the command in real time is key.

Background

- Perforce process model and ways of counting processes
- Perforce error log
- Perforce databases and locking model

Perforce Process Model

- Every command, from any client (e.g., **p4v**), appears to the server as if it came from the **p4** command line.
- Each command starts a process on the server.

Ways of Counting Processes: **`pgrep p4d`**

- Unix command. Equivalent to **`ps | grep p4d`**
- Includes some p4d processes with no corresponding p4 commands, because the p4 API allows you to run commands and hold their sockets open for future commands. This feature is used by **`p4v`**.

Ways of Counting Processes: `p4 monitor show`

- **Not suitable**. Doesn't reflect true process count when server is hanging, because commands are blocked before records are entered in process table. (Fixed in 2005.2?)
- Hangs when server is hanging (but there is a workaround).
- Quirk: When the Perforce server is restarted, commands from before the restart still appear.

Output: **`p4 monitor show`**

```
% p4 monitor show
 3114 R build      00:00:09 changes
 4394 R naveen     00:00:01 have
 4419 R dbloch     00:00:00 monitor

% p4 monitor show -l
 3114 R build      00:00:09 changes -m 1 -s submitted ...
 4394 R naveen     00:00:01 have
     /home/naveen/google/search/trace.h
 4419 R dbloch     00:00:00 monitor show -l
```

Perforce Error Log

- Specified with **`p4d -L <log>`.**
- Referred to as an “error log”, but with debug mode set to 2 or 3 (**`-v server=3`**), it includes process start and completion records.
- In Perforce 2005.2, additional entries are added containing information on locks held during commands.
- Mainly useful for analysis after the fact.

Sample Error Log Output

```
2006/01/20 00:10:04 pid 23520
  nora@nora-maps 172.30.0.103 [p4]
  'user -add -c 2002444
  //depot/google/search/maps.cc'

2006/01/20 00:10:16 pid 23520
  completed 12.512s 20+100us 0+0io
  0+0net 0k 23pf
```

Error Log Quirks

- Some commands, such as sync, have additional “compute end” log entries. (Note: Only CPU time in the first “compute end” line is valid.)
- p4 submit commands have additional **`dm-CommitSubmit`** and **`dm-SubmitChange`** entries.
- If a process is killed, there will be a “Process 10927 exited on a signal 15!” line instead of a “completed” line.
- If a command is interrupted on the client side, it may have two “completed” entries.
- p4 info commands have “completed” entries only, and don't have entries for when they start.
- Some of these quirks are fixed in 2005.2

Perforce Database

- There are about thirty-five database files, with names like “db.have”, “db.labels”, “db.counters”, etc.
- Details about the schema can be found at **`http://www.perforce.com/perforce/doc.051/schema/index.html`**

Database Locking

- p4d processes lock database files with either read or write locks using OS-level **`flock`** calls.
- All processes acquire locks in the same order, so there is never an actual deadlock.
- Locks are not always held for the full duration of a Perforce command.
- Locks are not held while accessing depot files or running triggers.

Putting the Pieces Together: **`locks.pl`**

- To diagnose server hangs, you care only about p4 commands holding locks.
- The (Unix) **`lsof`** command will list all locks held by all processes.
- Comparing the results of this to **`p4 monitor show -l`** output lets you correlate these PIDs with p4 commands.
- One more trick is needed. Since **`p4 monitor show`** may hang if the server is hanging, run another Perforce server on a different port, with its db.monitor a link to the main server's, and run **`p4 monitor show`** against it.

Sample **`locks.pl`** Output

```
Sun Jan 22 22:54:45 PST 2006

24324: WRITE locks: db.change, db.counters,
  db.have, db.integed, db.resolve, db.locks,
  db.archive, db.rev, db.revcx, db.revhx,
  db.revpx, db.working, db.changex
25913: read lock: db.domain

24324: smith       00:01:48 dm-CommitSubmit
25913: johnson     00:01:34 clients
```

`locks.pl` in Action

- Set to run every 30 seconds, when there are more than 200 **`p4d`** processes
- Originally, just sent mail so admins could take appropriate action
- Enhanced to kill known bad commands

Commands Reported by **`locks.pl`**

- Legitimate commands:
  `p4 integrate`, **`p4 submit`**
- Commands that the script kills automatically:
  `p4 changes -i`, **`p4 integrated`**
  We may add others as they come up.
- Commands we've killed manually:
  - **`p4 changes //depot/foo/bar/.../baz/...`**
  - **`p4 filelog ...`**
  - etc.

Commands Reported by **`locks.pl`** (continued)

Commands which we were able to eliminate by changing people's scripts or work habits:

- **`p4 opened -a`**: Changed scripts which were using this. Changed the default behavior of p4win and the Eclipse p4 plugin not to use it.
- **`p4 labels //depot/branches/foo/...`**: Changed script which was doing this.
- **`p4 changes @2005/11/01,@2005/12/01`**: Moved to replica server.

`locks.pl` Availability

- In Perforce Public Depot in **`//guest/dan_bloch`**

Notes on Killing Processes

- **`p4 monitor terminate`** is too slow to be useful. Must use **`kill`**, on server machine.
- **Killing processes can corrupt your database.**
- For safety, kill read-only commands only (e.g., **`p4 changes`**, **`p4 integrated`**).
- Must run **`p4 monitor clear <pid>`** after killing processes or they continue to appear in **`p4 monitor show`** output.

Future Work

- **`p4 integrate`** is supposed to be faster in 2006.1
- Move our database to SAN or RAM-SAN
- Write a script to break up big submits
- Split depot, maybe

Conclusion

- Server spikes are caused by long-running commands holding database locks.
- The **`locks.pl`** script allows offending commands to be identified in real time.
- Some commands can be killed, others must be allowed to finish.
- Knowledge is power.