



Polarion Software®

WHITE PAPER



SUBVERSION 1.7: ZIPPY, BUT SHOULD YOU UPGRADE?

By Lutz Dornbusch
Senior Consultant, Polarion Software

Europe, Middle-East, Africa: Polarion Software GmbH
Lautlinger Weg 3 — 70567 Stuttgart, GERMANY
Tel +49 711 489 9969 - 0
Fax +49 711 489 9969 - 20
www.polarion.com - info@polarion.com

Americas & Asia-Pacific: Polarion Software, Inc.
406 Tideway Dr. Alameda, CA 94501, USA
Tel +1 877 572 4005 (Toll free)
Fax +1 510 814 9983
www.polarion.com - info@polarion.com

Table of Contents

- Measurement Procedure 3
 - The Candidate Commands 3
 - Test Machine Specs 4
 - The Test Repository 4
- TEST #1: svn checkout 5
 - Analysis 6
- TEST #2: show log all 8
 - Analysis 8
- TEST #3: svn status 8
 - Analysis 8
- Conclusion 9

Introduction

Subversion 1.7 is a release mostly devoid of new features, but with many performance improvements. From some users' highly appreciated WC-NG to mostly invisible HTTPv2, the question is what to expect from these improvements.

Because Polarion's application lifecycle management products are built on top of Subversion, making SVN a core issue for our customers, we felt the need to highlight and benchmark some of the most visible performance boosts.

I conducted tests with several key SVN operations and found some results impressive enough to suggest that upgrading to SVN is probably worthwhile for most users. But there are some important caveats as well, especially on the client side (at the time of this writing).

In this paper, I share the results of my performance evaluation of Subversion 1.7 and recommendations for upgrading.

Measurement Procedure

I measured mostly on a small 1.6 GHz dual-core laptop. The reason is that even if I have faster hardware for production, I want to see performance under low-tech conditions. But I also included some datasets on a high-end computer to compare improvements.

I also tested "Windows only", as this is the OS our customers are using most.

I usually performed all commands 10 times, threw out the best and worst performance times, and calculated the average of the rest. On some tests, like `svn log` when caching jumps in, I used only the first (much higher) value.

I used a locally-installed Apache to circumvent nearly all network latency. Since that is different everywhere and always, I decided to push it out of the equation. I also performed some checkouts on a SSD to rule out HD performance effects.

The Candidate Commands

I was only interested in three commands:

- `svn checkout`
- `svn log`
- `svn status`

These three commands are the commands SVN users most often blame for their low performance.

Note that `svn status` operates without a server connection so it is HD-bound and therefore caching is significantly affecting the measurement, which is why I used the first measured value.

Test Machine Specs

CPU	1,6,GHz Dual core centrino
RAM	2.5 GB
HD	2,5" 120 GB/7200rpm
apache httpd	2.2.21 (std- windows build) on localhost
SVN 1.6	Subversion 1.6.13
SVN 1.7	Subversion 1.7.0

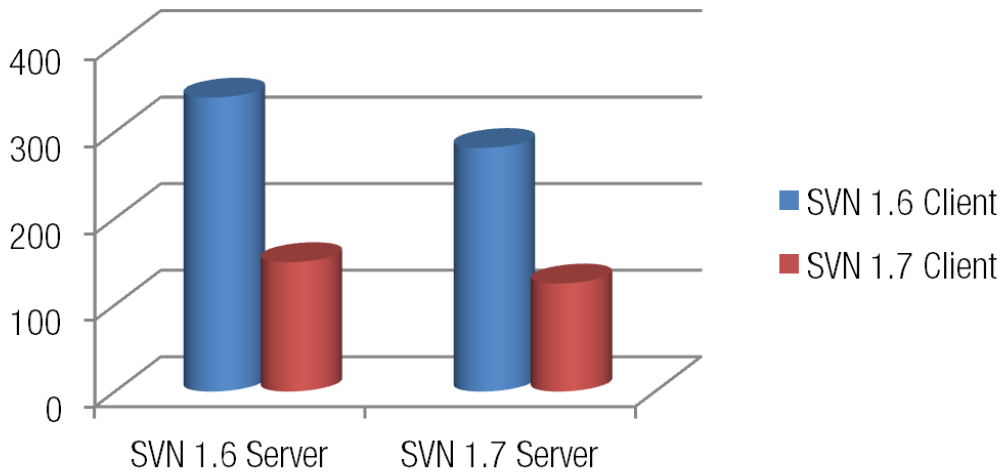
The Test Repository

I used a Polarion repository with a lot of projects and customer-specific configurations for my performance checks. This exemplifies Subversion's worst case scenario: a huge number of small files in another huge number of folders. As you see I have nearly as many folders as files and the average file size is about 3.2 KB.

Repository Size	32.454.371 Bytes (30,9 MB)
Revisions	2325 revisions
Number of Files in HEAD	5509 Files
Number of Folders in HEAD	4267 Folders
Overall File Size in HEAD	18,2MB

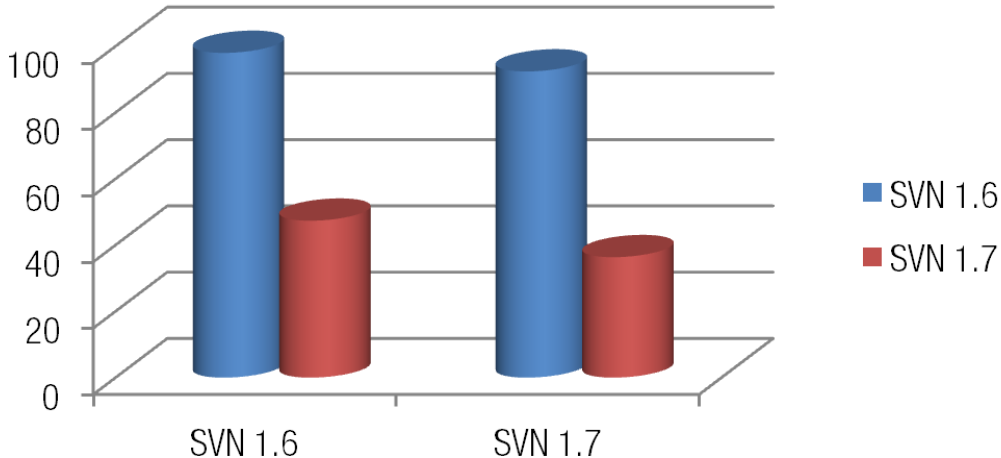
Test #1: Svn Checkout

Here are the results of a first test run in the environment I have already described:



Client / Server	SVN 1.6	SVN 1.7
SVN 1.6	339,79 sec	281,54 sec
SVN 1.7	149,63 sec	124,89 sec

We then made another checkout on a much faster machine with SSD (Intel core i7-2600, 8GB, Intel 320m 160Gb SSD). These are the results:



Client / Server	SVN 1.6	SVN 1.7
SVN 1.6	98,01 sec	92,41 sec
SVN 1.7	47,36 sec	36,35 sec

You can see the usage of SSD increases the performance dramatically. (Note that server and repo was sitting on a spinning 7200 rpm HD, only working copy was writing on SSD).

The ratio is somewhat equal to the slower case above: SVN1.7 is much faster, **even with a SVN 1.6 server.**

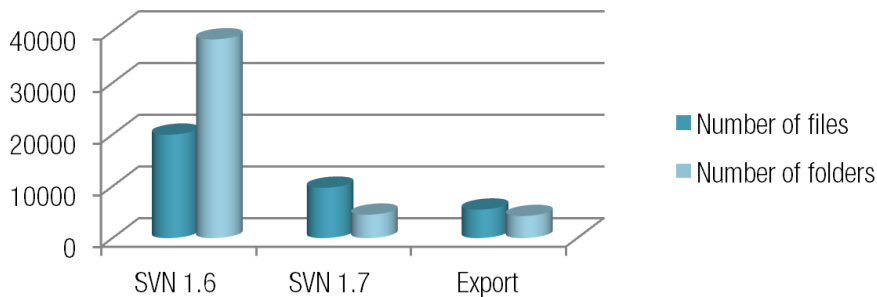
Analysis

These numbers are quite impressive! It seems that SVN 1.7 is more than twice as fast on an initial checkout even if you are using only the SVN 1.7 client. This makes an upgrade very recommendable. If you are using a SVN 1.7 Server, the improvements are even higher.

The reason for this performance impact is two-fold:

1. The new working copy format does not need to write so many files / folders

The new working copy format holds all SVN data on the topmost folder. No more .svn folders in each directory! As you see on the table, there is no improvement on working copy size though. However, you need many less folders (something around 260 folders max), while we had much, much more folders in old working copy format. Not so extreme, but still impressive, is the filecount: SVN1.7 will store the same files only once. In my Polarion repo are a lot files equal (mainly XML-files), so we end up with not twice of the file count. In Subversion 1.6 the file size roughly quadrupled:



	SVN 1.6	SVN 1.7	Export
Number of files	19992 files	9732 files	5509 Files
Number of folders	38411 folders	4526 folders	4267 Folders
Size	39,8MB	40,2MB	18,2MB

2. The new HTTPv2 Protocol uses far less handshakes and so reduces Apache overhead

Here is an Apache log of a checkout of a single file (client 1.6 / server 1.6):

```
127.0.0.1 - - "OPTIONS /repo/.dms/fields HTTP/1.1" 200 143
127.0.0.1 - - "PROPFIND /repo/.dms/fields HTTP/1.1" 207 353
127.0.0.1 - - "PROPFIND /repo!/svn/vcc/default HTTP/1.1" 207 244
127.0.0.1 - - "PROPFIND /repo!/svn/bln/2325 HTTP/1.1" 207 257
127.0.0.1 - - "PROPFIND /repo/.dms/fields HTTP/1.1" 207 353
127.0.0.1 - - "PROPFIND /repo!/svn/vcc/default HTTP/1.1" 207 244
127.0.0.1 - - "PROPFIND /repo!/svn/bln/2325 HTTP/1.1" 207 257
127.0.0.1 - - "PROPFIND /repo/.dms/fields HTTP/1.1" 207 353
127.0.0.1 - - "PROPFIND /repo!/svn/vcc/default HTTP/1.1" 207 257
127.0.0.1 - - "PROPFIND /repo!/svn/bc/2325/.dms/fields HTTP/1.1" 207 359
127.0.0.1 - - "OPTIONS /repo/.dms/fields HTTP/1.1" 200 143
127.0.0.1 - - "PROPFIND /repo/.dms/fields HTTP/1.1" 207 353
127.0.0.1 - - "PROPFIND /repo/.dms/fields HTTP/1.1" 207 353
127.0.0.1 - - "PROPFIND /repo!/svn/vcc/default HTTP/1.1" 207 244
127.0.0.1 - - "PROPFIND /repo!/svn/bln/2325 HTTP/1.1" 207 257
127.0.0.1 - - "PROPFIND /repo/.dms/fields HTTP/1.1" 207 353
127.0.0.1 - - "PROPFIND /repo!/svn/vcc/default HTTP/1.1" 207 244
127.0.0.1 - - "PROPFIND /repo!/svn/bln/2325 HTTP/1.1" 207 257
127.0.0.1 - - "REPORT /repo!/svn/vcc/default HTTP/1.1" 200 1067
```

Here is the same checkout on a SVN 1.7 client: (client 1.7 / server 1.6):

```
127.0.0.1 - - "OPTIONS /repo/.dms/fields HTTP/1.1" 200 143
127.0.0.1 - - "PROPFIND /repo/.dms/fields HTTP/1.1" 207 353
127.0.0.1 - - "PROPFIND /repo!/svn/vcc/default HTTP/1.1" 207 244
127.0.0.1 - - "PROPFIND /repo!/svn/bln/2325 HTTP/1.1" 207 257
127.0.0.1 - - "PROPFIND /repo/.dms/fields HTTP/1.1" 207 353
127.0.0.1 - - "PROPFIND /repo!/svn/vcc/default HTTP/1.1" 207 257
127.0.0.1 - - "PROPFIND /repo!/svn/bc/2325/.dms/fields HTTP/1.1" 207 359
127.0.0.1 - - "OPTIONS /repo/.dms/fields HTTP/1.1" 200 143
127.0.0.1 - - "PROPFIND /repo/.dms/fields HTTP/1.1" 207 353
127.0.0.1 - - "PROPFIND /repo/.dms/fields HTTP/1.1" 207 353
127.0.0.1 - - "PROPFIND /repo!/svn/vcc/default HTTP/1.1" 207 244
127.0.0.1 - - "PROPFIND /repo!/svn/bln/2325 HTTP/1.1" 207 257
127.0.0.1 - - "REPORT /repo!/svn/vcc/default HTTP/1.1" 200 1067
```

So even on a 1.6 Server there is a reduced number of http transactions (19 vs. 13), but if you upgrade to a 1,7 Server you got the full power of the new HTTPv2 Protocol(client 1.7 / server 1.7):

```
127.0.0.1 - - "OPTIONS /repo/.dms/fields HTTP/1.1" 200 143
127.0.0.1 - - "OPTIONS /repo/.dms/fields HTTP/1.1" 200 143
127.0.0.1 - - "PROPFIND /repo!/svn/rvr/2325/.dms/fields HTTP/1.1" 207 359
127.0.0.1 - - "OPTIONS /repo/.dms/fields HTTP/1.1" 200 143
127.0.0.1 - - "OPTIONS /repo/.dms/fields HTTP/1.1" 200 143
127.0.0.1 - - "REPORT /repo!/svn/me HTTP/1.1" 200 1099
```

Only 6 HTTP requests!

Test #2: Show Log All

Here are the results of a first test run in the environment I have already described:

Client / Server	SVN 1.6	SVN 1.7
SVN 1.6	20.63 sec	28.57 sec
SVN 1.7	20.58 sec	20.09 sec

Analysis

I did not get the reason why logging is slower between a SVN 1.6 client and a SVN 1.7 server. Although this would be something interesting to investigate, I felt it is probably not one of the more common usage scenarios and so didn't take the time to delve into it. So no big news here: performance is quite the same on my machine, but it seems that the HTTP round-trips have been reduced as well:

Log message for SVN 1.6 (client 1.6 / server 1.6):

```
127.0.0.1 - - "OPTIONS /repo HTTP/1.1" 200 143
127.0.0.1 - - "PROPFIND /repo HTTP/1.1" 207 337
127.0.0.1 - - "PROPFIND /repo!/svn/vcc/default HTTP/1.1" 207 244
127.0.0.1 - - "PROPFIND /repo!/svn/bln/2325 HTTP/1.1" 207 257
127.0.0.1 - - "PROPFIND /repo HTTP/1.1" 207 337
127.0.0.1 - - "PROPFIND /repo!/svn/vcc/default HTTP/1.1" 207 244
127.0.0.1 - - "PROPFIND /repo!/svn/bln/2325 HTTP/1.1" 207 257
127.0.0.1 - - "PROPFIND /repo HTTP/1.1" 207 337
127.0.0.1 - - "PROPFIND /repo!/svn/vcc/default HTTP/1.1" 207 244
127.0.0.1 - - "PROPFIND /repo!/svn/bln/2325 HTTP/1.1" 207 257
127.0.0.1 - - "PROPFIND /repo HTTP/1.1" 207 337
127.0.0.1 - - "PROPFIND /repo!/svn/vcc/default HTTP/1.1" 207 257
127.0.0.1 - - "REPORT /repo!/svn/bc/2325 HTTP/1.1" 200 36418
```

And here is the SVN 1.7 log (client 1.7 / server 1.7):

```
127.0.0.1 - - "OPTIONS /repo HTTP/1.1" 200 143
127.0.0.1 - - "OPTIONS /repo HTTP/1.1" 200 143
127.0.0.1 - - "OPTIONS /repo HTTP/1.1" 200 143
```

Test #3: Svn Status

Client	local operation only
SVN 1.6	7.42 sec
SVN 1.7	3.15 sec

Analysis

On the local SVN status command, the new working copy structure strikes again: the optimized structure in SVN 1.7 clocks in again more than twice as fast compared to SVN 1.6.

Note that I used the whole repository as working copy, and changed around 10 to 20 files deep in the nested folders.

Conclusion

Subversion 1.7 seems to dramatically improve the performance. The new working copy architecture is much faster, and the overall usage is much better: no `.svn` folders everywhere, so you can easily copy files from your working copy to other places.

We can definitely **recommend to upgrade** at least your clients to SVN 1.7 as soon as possible!

However, it's important to keep in mind that the new client's working copy format is incompatible with older (SVN1.6) clients. So make sure to update all clients you are using in your tool chain (e.g. TortoiseSVN, Subversive, smartSVN, etc.) If some of your clients don't yet support Subversion 1.7, then you will need to wait to upgrade until they do.

About Polarion Software



Polarion Software is a privately held company operating globally with offices in Europe and North America, and partners world-wide.

The company hit the market with their Software Application Lifecycle Management solution in Spring of 2005. Today, hundreds of Fortune 1000 customers and over 1,000,000 users in a wide variety of industries, ranging from automobile manufacturing, to aerospace, to medical device engineering rely daily on Polarion's solutions to help them deliver greater efficiency and productivity in requirements management, quality assurance and testing, team collaboration and communication, and full application lifecycle management.

Polarion Software also sponsors the development of [Subversive Team Provider](#), one of the most popular open source plug-ins for the Eclipse IDE and Subversion.

For more information, including a brief introductory video, visit <http://www.polarion.com/company/>.

About the Author



Lutz Dornbusch is a senior consultant with [Polarion Software's Professional Services](#) group, which delivers training and consulting services for Subversion as well as the company's own products.

He has more than 20 years' experience in configuration management and programming, and in his role as Subversion consultant he has successfully migrated various version control systems to Subversion.

Lutz also leads the [SubTrain Project](#) which delivers open source Subversion training materials.

Handy Links

- [Polarion Application Lifecycle Solutions](#)
- [Polarion solutions for Medical Device Manufacturing](#)
- [Polarion solutions for Automotive OEMs and suppliers](#)
- [Polarion Tools for Subversion](#)
- [Contact Polarion Software](#)

