Performance Test Report for GZIPFilter

Bo Li, Maohai Huang

2005-06

Ref: SPIRE-NAO-NOT-002467

TOC:

1. Introduction

This report describes the performance test for GZIPFilter exercised on June 15, 2005. For more details of GZIPFilter see the GZIPFilter design document. The aim of the test is to evaluate the effectiveness of GZIPFilter in delivering objects of different size and network speed.

2. Assumptions

The bandwidth of the network is stable and constant during the test. The small r.m.s. variation of extracting time during each test case (see Appendix F), particularly the small difference between the first run and the rest runs tends to support the assumption, because unstable network speed and network caching will cause the extracting time to vary.

3. Test plan

3.1 Test setup

Server:

scott1.bnsc.rl.ac.uk (locate in RAL, Didcot, UK), tomcat 4.03, jdk1.4.1_02

Client:

libo.bao.ac.cn (locate in NOAC, Beijing, China) PentiumM 1.6GHz CPU, 512MB RAM, jdk1.4.2_07

Network:

512Kbits/s ADSL

Test program:

spire142/hcss620

GZIPFilter and other test code

Access.java
Modified for the test.

3.2 Test procedure for different sized objects delivery:

Main test procedure is as following. Detailed steps are listed in Appendix E:

- Deliver *TestExecution* objects on the server through *MockDataServer*, using SPIRE QLA *TestExecutionBrowser* locally as the receiver;
- Record response stream size;
- Record extracting time;
- Repeat above steps for 5 times using and not using GZIPFilter, respectively, with a time interval of 2 minutes;
- Calculate average (avg), Root Mean Squared (rms) deviations, and rms/avg ratio, response stream size decrease ratio (RSSDR), extracting time decrease ratio (ETDR)

Test parameter variation for each test case:

- A. Delivering 372 arbitrary sized *TestExecution* objects
- B. Delivering 100 1.06 k byte sized *TestExecution* objects
- C. Delivering 53 2.01 k byte sized *TestExecution* objects
- D. Delivering 23 4.69 k byte sized *TestExecution* objects
- E. Delivering 6 16.7 k byte sized *TestExecution* objects
- F. Delivering 1 1.06 k byte sized *TestExecution* objects
- G. Delivering 10 1.06 k byte sized *TestExecution* objects
- H. Delivering 100 *TmPackets* objects (size are from 0.29 to 1 k byte).
- I. Delivering 100 *TmPackets* objects in different bandwidths

The reasons for the selection of test cases above are:

- Case A is to see the general efficiency of the GZIPFilter in delivering *TestExecution* objects;
- Cases B, C, D, E is to see if there are relations between the efficiency of the GZIPFilter and object size.
- Cases A, B, C, D, E, F, G is to see if there are relations between the efficiency of the GZIPFilter and the total size of the delivered objects.
- Case H is to see the general efficiency of the GZIPFilter in delivering *TmPackets* objects.
- Case I is to see the general efficiency of the GZIPFilter in delivering *TmPackets* objects in different bandwidths.

Objects bigger than 17 k byte and objects less than 1 k byte are not tested since the main range of the size of *TestExecution* objects is from 1 to 17 k byte.

4. Test results and conclusions

4.1 Test results

The original test data are listed in appendix F. The average and derived ratios are list in the following table:

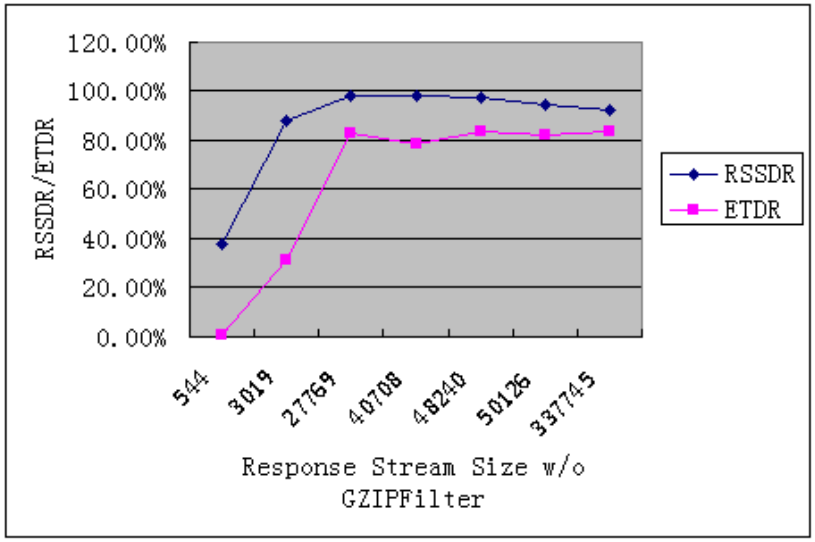
Table 1: Average of Test Data and their Derived RSSDR and ETDR

Test	Response stream size w/o GZIPFilter (byte)	Response stream size w/ GZIPFilter (byte)	RSSDR average	RSSDR rms/avg	Extracting time w/o GZIPFilter (ms, with var.)	extracting time w/ GZIPFilter (ms, with var.)	ETDR average	ETDR rms/average
A	337745	24676	93%	0	56704 (9%)	9143 (7%)	84%	2%
B	27769	513	98%	0	10779 (1%)	1854(0.5%)	83%	0.2%
C	40708	813	98%	0	6596 (2%)	1420 (2%)	78%	1%
D	48240	1315	97%	0	7371 (3%)	1218 (4%)	83%	1%

E	50126	2589	94%	0	9232 (1%)	1645 (1%)	82%	0.4%
F	544	337	38%	0	537 (4%)	535 (3%)	0.4%	592%
G	3019	367	88%	0	1530 (2%)	1049 (2%)	31%	4%
H	19874	3260	84%	0	6660 (4%)	4136 (2%)	38%	9%

4.2 Effectiveness of GZIPFilter for different data size

Figure 1. RSSDRs and ETDRs as Fuctions of Uncompressed Response Stream Size (byte)



The GZIPFilter decreases the stream size significantly, with RSSDR being more than 80%, when the uncompressed response stream size is greater than 3 k byte.

The extracting time is reduced significantly when the uncompressed response stream size higher than 3 k byte, with the ETDR staying almost constant at 80% when the uncompressed response stream size vary from 27 to 337 k byte as shown by test cases A, B, C, D, and E.

The five points on the right in the above figure for ETDR (shown in pink) represent test cases A, B, C, D, and E, respectively. The object size for these cases are in the range of 1 to 17 k byte. The above figure shows that object size has little influence on the efficiency of the GZIPFilter in the size range of 1 to 17 k byte.

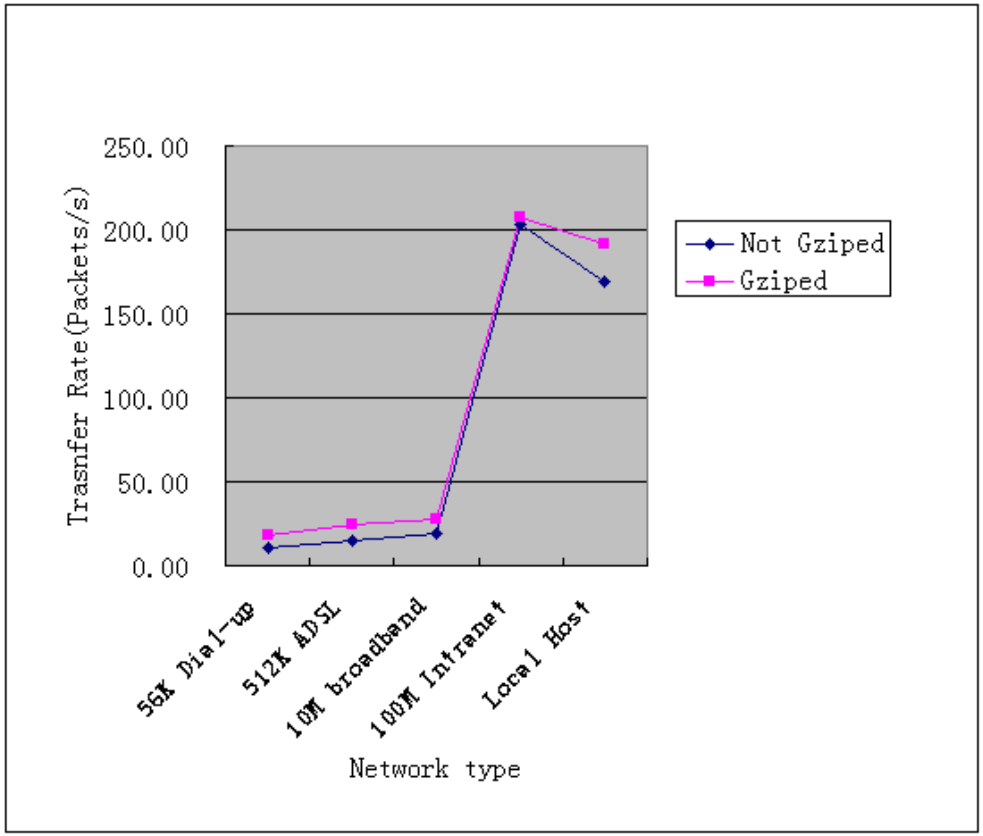
4.3 Effect of network bandwidth

The following are data and derived ratios using different type of network connections in test case I:

Table 2. Delivering 100 *TmPackets* objects using network of different bandwidths

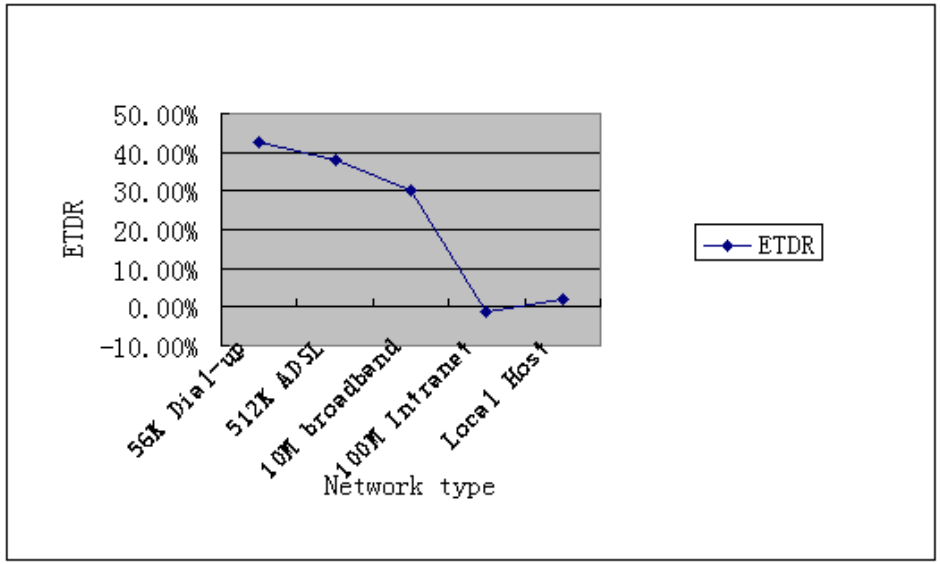
Test Runs	56K Dial-up -Extracting time w/o GZIPFilter (ms)	56K Dial-up - Extracting time w/ GZIPFilter (ms)	56K Dial-up -ETDR	512 K ASDL -Extracting time w/o GZIPFilter (ms)	512 K ASDL -Extracting time w/ GZIPFilter (ms)	512 K ASDL -ETDR	10M broadband - Extracting time w/o GZIPFilter (ms)	10M broadband - Extracting time w/ GZIPFilter (ms)	10M broadband - ETDR	100M Intranet - Extracting time w/o GZIPFilter (ms)	100M Intranet - Extracting time w/ GZIPFilter (ms)	100M Intranet - ETDR	Local host - Extracting time w/o GZIPFilter (ms)	Lo - Ex tin GZ (m
1	9534	5357	44%	6810	4096	40%	5107	3675	28%	471	491	-4%	591	58
2	9774	5368	45%	6178	4216	32%	5157	3575	31%	471	470	0.2%	520	53
3	9434	5356	43%	6710	4176	38%	5107	3615	29%	461	481	-4%	560	55
4	9373	5768	38%	6710	4156	38%	5148	3475	33%	470	470	0%	651	67
5	9536	5528	42%	6890	4036	41%	5157	3603	30%	491	481	2%	591	52
Transfer rate	10	18	N/A	15	25	N/A	19	28	N/A	204	208	N/A	169	19
Avg.	9530	5475	43%	6660	4136	38%	5135	3589	30%	473	479	-1%	583	57
rms.	137	160	0.02	250	63	0.03	23	66	0.01	10	8	0.03	43	54
rms./avg.	1%	3%	5%	4%	2%	9%	0.5%	2%	5%	2%	2%	-209%	7%	9%

Figure 2: Transfer Rate as a Function of network Bandwidth



The data and figure show that ransfer rates are highly dependent on the bandwidth. The decreased the transfer rate in "local host" is due to high load of the testing computer (running server-side and client-side programs simultaneously).

Figure 3: ETDR as a Function of network Bandwidth



The above figure shows that the effectiveness of the GZIPFilter expectedly decreases with the increasing bandwidth. When testing on a 100Mbps intranet it actually takes longer to use the GZIPFilter than not using it, showing that the processing speed has taken over network transfer speed as the main transfer bottleneck. The up-turn for "Local Host" is due to high load of the testing computer (running server-side and client-side programs simultaneously).

This test shows that in the specific environment of this test GZIPFilter increases transfer speed significantly on a network not faster than a 10M broadband. In a few years when Herschel is in routine operation phase the bandwidths for most astronomical community end-users and almost all users at home or using a mobile link are bound to increase compared with what they have today, is still unlikely be greater than 10Mbps. Considering that the processing speed of most computers will also increase in the same period, helping to shift the curve in the figure above to the right. It is usually easier to increase the speed of a computer than to increase the speed of large area networks. Therefore GZIPFilter will likely to stay relevant for a prolonged period of time.

Appendices

A. Response stream size definition

Uncompressed response stream size (size-before-gzip) means the total size (byte) of response stream before using GZIPFilter, with code below:

```
URL noCompress = new URL(url);
HttpURLConnection huc =
```

```
(HttpURLConnection)noCompress.openConnection();
huc.setRequestProperty("user-agent","Mozilla(MSIE)");
huc.connect();
ByteArrayOutputStream baos = new ByteArrayOutputStream();
InputStream is = huc.getInputStream();
while(is.read() != -1) {
   baos.write((byte)is.read());
}
byte[] b1 = baos.toByteArray();
int size1 = b1.length;
```

Compressed response stream size (size-after-gzip) means the total size (byte) of response stream after using GZIPFilter, with code below:

```
URL compress = new URL(url);
HttpURLConnection hucCompress =
(HttpURLConnection)noCompress.openConnection();
hucCompress.setRequestProperty("accept-encoding","gzip");
hucCompress.setRequestProperty("user-agent","Mozilla(MSIE)");
hucCompress.connect();
ByteArrayOutputStream baosCompress =
new ByteArrayOutputStream();
InputStream isCompress = hucCompress.getInputStream();
while(isCompress.read() != -1) {
   baosCompress.write((byte)isCompress.read());
}
byte[] b2 = baosCompress.toByteArray();
int size2 = b2.length;
```

B. Extracting time definition

Extracting time means time from `HcssConnection.getConnection()` to all `TestExecution` objects are obtained from the server, with code below:

```
long start = System.currentTimeMillwas();
ProductReader reader = HcssConnection.getConnection (_access);
ProductStream stream = reader.openStream();
int counter = 0;
```

```
while (stream.hasNext()) {
    counter ++;
    test = (TestExecution) stream.next();
    if (test.hasCompleted())
        _testExecutionObjects.add( test );
    else Access.log.info (test.getTestProcedure().getName()+ " was not complete - not
dwasplaying");
}
long stop1 = System.currentTimeMillwas();
```

C. Response stream size decrease ratio (RSSDR) definition

Response stream size decrease ratio = (1- (size-after-gzip / size-before-gzip))*100%

D. Extracting time decrease ratio (ETDR) definition

Extracting time decrease ratio = (1- (time-after-gzip / time-before-gzip))*100%

time-after-gzip: the extracting time after using GZIPFilter

time-before-gzip: the extracting time before using GZIPFilter

E. Data used in the test

Data used in the test can be found in `/export/home/hcss/servlets/WEB-INF/classes/TE_temp_dir` on scott1.bnsc.rl.ac.uk.

./dball/: 372 arbitrary sized `TestExecution` objects

./10_1k/: 10 1.06 k byte sized `TestExecution` objects

./10k/: 6 16.7 k byte sized `TestExecution` objects

./1_1k/: 1 1.06 k byte sized `TestExecution` object

./1k/: 100 1.06 k byte sized `TestExecution` objects

./2k/: 53 2.01 k byte sized `TestExecution` objects

./5k/: 23 4.69 k byte sized *TestExecution* objects

./TmPackets/: *TmPackets* objects

F. Detailed operation steps

Step 1: Run `> java demo.GzipPerformanceTester` to run all the test cases. Pipe the console results to a file named *log*.

Step 2: Use the generated *log* file in former step and run *>get_result_gawk.bat* to get formatted results.

Step 3: Copy the formatted results to the corresponding cells in *TestResult.xls*.

Step 4: Copy the data tables and chart from *TestResult.xls* and make this document.

G. Data for test cases A to I

A. Delivering 372 arbitrary sized *TestExecution* objects

Test Runs	Response stream size w/o GZIPFilter (byte)	Response stream size w/ GZIPFilter (byte)	RSSDR	Extracting time w/o GZIPFilter (ms)	Extracting time w/ GZIPFilter (ms)	ETDR
1	337745	24676	93%	55520	8022	86%
2	337745	24676	93%	55169	9344	83%
3	337745	24676	93%	51314	9654	81%
4	337745	24676	93%	54779	8883	84%
5	337745	24676	93%	66736	9814	85%
Avg.	337745	24676	93%	56704	9143	84%
rms.	0	0	N/A	5238	644	0.02
rms./avg.	0	0	N/A	9%	7%	2%

B. Delivering 100 1.06 k byte sized *TestExecution* objects

Test Runs	Response stream size w/o GZIPFilter (byte)	Response stream size w/ GZIPFilter (byte)	RSSDR	Extracting time w/o GZIPFilter (ms)	Extracting time w/ GZIPFilter (ms)	ETDR
1	27769	513	98%	10565	1843	83%
2	27769	513	98%	10936	1863	83%
3	27769	513	98%	10795	1843	83%
4	27769	513	98%	10812	1866	83%
5	27769	513	98%	10787	1857	83%
Avg.	27769	513	98%	10779	1854	83%
rms.	0	0	N/A	120	10	0.001
rms./avg.	0	0	N/A	1%	1%	0.2%

C. Delivering 53 2.01 k byte sized `TestExecution` objects

Test Runs	Response stream size w/o GZIPFilter (byte)	Response stream size w/ GZIPFilter (byte)	RSSDR	Extracting time w/o GZIPFilter (ms)	Extracting time w/ GZIPFilter (ms)	ETDR
1	40708	813	98%	6787	1405	79%
2	40708	813	98%	6593	1402	79%
3	40708	813	98%	6490	1412	78%
4	40708	813	98%	6691	1483	78%
5	40708	813	98%	6419	1402	78%
Avg.	40708	813	98%	6596	1420	78%
rms.	0	0	N/A	133	31	0.01
rms./avg.	0	0	N/A	2%	2%	1%

D. Delivering 23 4.69 k byte sized `TestExecution` objects

Test Runs	Response stream size w/o GZIPFilter (byte)	Response stream size w/ GZIPFilter (byte)	RSSDR	Extracting time w/o GZIPFilter (ms)	Extracting time w/ GZIPFilter (ms)	ETDR

1	48240	1315	97%	7321	1154	84%
2	48240	1315	97%	7123	1282	82%
3	48240	1315	97%	7371	1272	83%
4	48240	1315	97%	7771	1172	85%
5	48240	1315	97%	7270	1211	83%
Avg.	48240	1315	97%	7371	1218	83%
rms.	0	0	N/A	216	52	0.01
rms./avg.	0	0	N/A	3%	4%	1%

E. Delivering 6 16.7 k byte sized *`TestExecution`* objects

Test Runs	Response stream size w/o GZIPFilter (byte)	Response stream size w/ GZIPFilter (byte)	RSSDR	Extracting time w/o GZIPFilter (ms)	Extracting time w/ GZIPFilter (ms)	ETDR
1	50126	2589	95%	9063	1673	82%
2	50126	2589	95%	9353	1653	82%
3	50126	2589	95%	9233	1653	82%
4	50126	2589	95%	9377	1633	83%
5	50126	2589	95%	9134	1612	82%
Avg.	50126	2589	95%	9232	1645	82%
rms.	0	0	N/A	122	21	0.004
rms./avg.	0	0	N/A	1%	1%	0.4%

F. Delivering 1 1.06 k byte sized *`TestExecution`* object

Test Runs	Response stream size w/o GZIPFilter (byte)	Response stream size w/ GZIPFilter (byte)	RSSDR	Extracting time w/o GZIPFilter (ms)	Extracting time w/ GZIPFilter (ms)	ETDR
1	544	337	38%	551	551	0.0%
2	544	337	38%	521	531	-1.9%

3	544	337	38%	531	511	1.9%
4	544	337	38%	571	551	3.5%
5	544	337	38%	510	530	3.9%
Avg.	544	337	38%	537	535	0.4%
rms.	0	0	N/A	22	15	0.02
rms./avg.	0	0	N/A	4%	3%	592%

G. Delivering 10 1.06 k byte sized *TestExecution* objects

Test Runs	Response stream size w/o GZIPFilter (byte)	Response stream size w/ GZIPFilter (byte)	RSSDR	Extracting time w/o GZIPFilter (ms)	Extracting time w/ GZIPFilter (ms)	ETDR
1	3019	367	88%	1542	1031	33%
2	3019	367	88%	156	1082	31%
3	3019	367	88%	1492	1051	30%
4	3019	367	88%	1531	1045	32%
5	3019	367	88%	1521	1037	32%
Avg.	3019	367	88%	1530	1049	31%
rms.	0	0	N/A	23	18	0.01
rms./avg.	0	0	N/A	2%	2%	4%

H. Delivering 100 *TmPackets* objects

Test Runs	Response stream size w/o GZIPFilter (byte)	Response stream size w/ GZIPFilter (byte)	RSSDR	Extracting time w/o GZIPFilter (ms)	Extracting time w/ GZIPFilter (ms)	ETDR
1	19874	3260	84%	6810	4096	40%
2	19874	3260	84%	6178	4216	32%
3	19874	3260	84%	6710	4176	38%
4	19874	3260	84%	6710	4156	38%

5	19874	3260	84%	6890	4036	41%
Avg.	19874	3260	84%	6660	4136	38%
rms.	0	0	N/A	250	63	0.03
rms./avg.	0	0	N/A	4%	2%	9%