Improving hyrax performance for small data requests

Fedor Baart

January 12, 2009

Abstract

Testing the performance of the hyrax opendap server showed a latency for small requests. This report describes possibilities to improve the performance of hyrax for small requests.

1 Introduction

After using the hyrax server from existing code which was doing a lot of small (arrays of size 1×800) requests a lower than expected performance was noted.

2 Tests

2.1 Test system

A test system was setup in a virtual machine on the following hard & software

- host iMac 2.4Ghz Intel Core 2 Duo, 2GB, OSX 10.5.5
- virtual machine 1 processor, 512MB, Ubuntu 8.10, running under vmware fusion
- servlet container tomcat 6.0.18 (binary download, running under user folder)
- hyrax bes, dap_server, libdap, hdf5_handler, freeform_handler, netcdf_handler all at revision 20022 from the trunk
- hdf hdf5-1.8.2
- netcdf netcdf-4.0
- wireshark 1.0.3

2.2 Test script

To test the performance the script displayed in listing 1 was used.

Listing 1: Performance measurements

```
#!/bin/bash
url=http://localhost:8080/opendap/data/nc/coads_climatology.nc.ascii?
    COADSX
R=20
output=results.txt
format='%{time_namelookup};%{time_connect};%{time_pretransfer};%{
    time_starttransfer};%{time_total}\n'
echo "Lookup_time;Connect_time;Pretransfer_time;Starttransfer_time;Total_
    time" > $output
for i in `seq 1 $R`;
do
 curl -w "$format" -o /dev/null -s $url >> $output
 sleep 0.3 # add sleep big enough to make sure, no load effects are
     measured
done
```

Detailed analysis was done using the statistics from wireshark listening on the loopback interface.

3 Results

The conversation between curl and tomcat is listed in listing 2.

Listing 2: Conversation between

```
GET /opendap/data/nc/coads_climatology.nc.ascii?COADSX HTTP/1.1
User-Agent: curl/7.18.2 (i486-pc-linux-gnu) libcurl/7.18.2 OpenSSL/0.9.8g
    zlib/1.2.3.3 libidn/1.8
Host: localhost:8080
Accept: */*

HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Last-Modified: Thu, 08 Jan 2009 23:48:28 GMT
XDODS-Server: dods/3.1
XOPeNDAP-Server: bes/3.6.2 dap-server/ascii/3.8.5, freeform_handler
    /3.7.9, netcdf_handler/3.7.9, dap-server/usage/3.8.5, dap-server/www
    /3.8.5
XDAP: 3.1
Content-Description: dods_ascii
Content-Type: text/plain
```

```
Transfer−Encoding: chunked
Date: Sat, 10 Jan 2009 00:04:26 GMT

381
Dataset: coads_climatology.nc
COADSX, 21, 23, 25, 27, 29, 31, 33, 35, 37, 39, 41, 43, 45, 47, 49, 51, 53,
    55, 57, 59, 61, 63, 65, 67, 69, 71, 73, 75, 77, 79, 81, 83, 85, 87, 89,
    91, 93, 95, 97, 99, 101, 103, 105, 107, 109, 111, 113, 115, 117, 119,
    121, 123, 125, 127, 129, 131, 133, 135, 137, 139, 141, 143, 145, 147,
    149, 151, 153, 155, 157, 159, 161, 163, 165, 167, 169, 171, 173, 175,
    177, 179, 181, 183, 185, 187, 189, 191, 193, 195, 197, 199, 201, 203,
    205, 207, 209, 211, 213, 215, 217, 219, 221, 223, 225, 227, 229, 231,
    233, 235, 237, 239, 241, 243, 245, 247, 249, 251, 253, 255, 257, 259,
    261, 263, 265, 267, 269, 271, 273, 275, 277, 279, 281, 283, 285, 287,
    289, 291, 293, 295, 297, 299, 301, 303, 305, 307, 309, 311, 313, 315,
    317, 319, 321, 323, 325, 327, 329, 331, 333, 335, 337, 339, 341, 343,
    345, 347, 349, 351, 353, 355, 357, 359, 361, 363, 365, 367, 369, 371,
    373, 375, 377, 379

0
```

The total response time for this request is 0.40065 seconds with a standard deviation of 0.0086 seconds.

Next we look at the conversation between the BES and the olfs. The conversation is listed in listing 3.

Listing 3: Conversation between

```
00000f4d
<?xml version="1.0" encoding="UTF−8"?>
<request xmlns="http://xml.opendap.org/ns/bes/1.0#" reqID="[http
    −8080−1:16:bes_request]">
  <setContext name="errors">xml</setContext>
  <showInfo node="/data/nc/coads_climatology.nc" />
</request>
0000000d

000011ad
<?xml version="1.0" encoding="ISO−8859−1"?>
<showInfo><response><dataset isData="true" thredds_collection="false">
    <name>/data/nc/coads_climatology.nc</name><size>3114044</size>
    <lastmodified><date>2009−01−09</date><time>00:48:28</time></
    lastmodified></dataset></response></showInfo>
0000000d

00000f4d
<?xml version="1.0" encoding="UTF−8"?>
```

```
<request xmlns="http://xml.opendap.org/ns/bes/1.0#" reqID="[http
    -8080-1:16:bes_request]">
  <setContext name="errors">xml</setContext>
  <showInfo node="/data/nc/coads_climatology.nc" />
</request>
0000000d

000011ad
<?xml version="1.0" encoding="ISO-8859-1"?>
<showInfo><response><dataset isData="true" thredds_collection="false">
    <name>/data/nc/coads_climatology.nc</name><size>3114044</size>
    <lastmodified><date>2009-01-09</date><time>00:48:28</time></
    lastmodified></dataset></response></showInfo>
0000000d

00000f4d
<?xml version="1.0" encoding="UTF-8"?>
<request xmlns="http://xml.opendap.org/ns/bes/1.0#" reqID="[http
    -8080-1:16:bes_request]">
  <setContext name="errors">xml</setContext>
  <showInfo node="/data/nc/coads_climatology.nc" />
</request>
0000000d

000011ad<?xml version="1.0" encoding="ISO-8859-1"?>
<showInfo><response><dataset isData="true" thredds_collection="false">
    <name>/data/nc/coads_climatology.nc</name><size>3114044</size>
    <lastmodified><date>2009-01-09</date><time>00:48:28</time></
    lastmodified></dataset></response></showInfo>
0000000d

00000f4d
<?xml version="1.0" encoding="UTF-8"?>
<request xmlns="http://xml.opendap.org/ns/bes/1.0#" reqID="[http
    -8080-1:16:bes_request]">
  <setContext name="errors">xml</setContext>
  <showInfo node="/data/nc/coads_climatology.nc" />
</request>
0000000d

000011ad
<?xml version="1.0" encoding="ISO-8859-1"?>
<showInfo><response><dataset isData="true" thredds_collection="false">
    <name>/data/nc/coads_climatology.nc</name><size>3114044</size>
    <lastmodified><date>2009-01-09</date><time>00:48:28</time></
    lastmodified></dataset></response></showInfo>
```

```
0000000d

0000253d
<?xml version="1.0" encoding="UTF-8"?>
<request xmlns="http://xml.opendap.org/ns/bes/1.0#" reqID="[http
    -8080-1:16:bes_request]">
  <setContext name="xdap_accept">2.0</setContext>
  <setContext name="dap_explicit_containers">no</setContext>
  <setContext name="errors">xml</setContext>
  <setContainer name="catalogContainer" space="catalog">/data/nc/
      coads_climatology.nc</setContainer>
  <define name="d1" space="default">
    <container name="catalogContainer">
      <constraint>COADSX</constraint>
    </container>
  </define>
  <get type="ascii" definition="d1" />
</request>
0000000d

0000381d
Dataset: coads_climatology.nc
COADSX, 21, 23, 25, 27, 29, 31, 33, 35, 37, 39, 41, 43, 45, 47, 49, 51, 53,
    55, 57, 59, 61, 63, 65, 67, 69, 71, 73, 75, 77, 79, 81, 83, 85, 87, 89,
    91, 93, 95, 97, 99, 101, 103, 105, 107, 109, 111, 113, 115, 117, 119,
    121, 123, 125, 127, 129, 131, 133, 135, 137, 139, 141, 143, 145, 147,
    149, 151, 153, 155, 157, 159, 161, 163, 165, 167, 169, 171, 173, 175,
    177, 179, 181, 183, 185, 187, 189, 191, 193, 195, 197, 199, 201, 203,
    205, 207, 209, 211, 213, 215, 217, 219, 221, 223, 225, 227, 229, 231,
    233, 235, 237, 239, 241, 243, 245, 247, 249, 251, 253, 255, 257, 259,
    261, 263, 265, 267, 269, 271, 273, 275, 277, 279, 281, 283, 285, 287,
    289, 291, 293, 295, 297, 299, 301, 303, 305, 307, 309, 311, 313, 315,
    317, 319, 321, 323, 325, 327, 329, 331, 333, 335, 337, 339, 341, 343,
    345, 347, 349, 351, 353, 355, 357, 359, 361, 363, 365, 367, 369, 371,
    373, 375, 377, 379
0000000d
```

The time for each timing can be seen in the roundtrip time graph in figure 1 The long round trips are the time it takes to respond between the *000011ad* and the start of the xml response.

4 Analysis

A response time of 0.4 seconds for any request is higher than needed. If we look at where the 0.4 seconds is spent we can see that the communication between

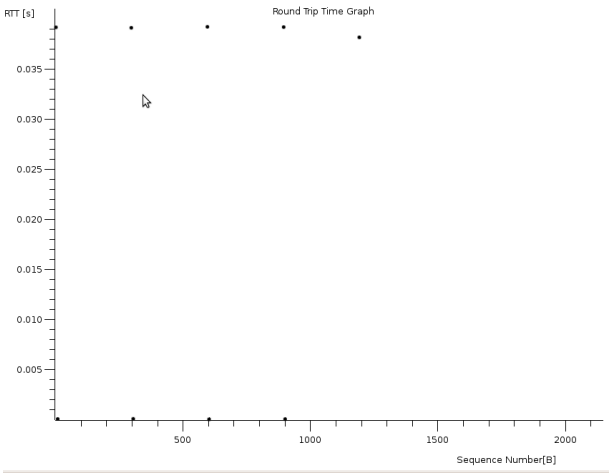

Figure 1: Roundtrip Time for bes-olfs communication

bes and olfs is repetitive. The request showinfo is repeated 4 times before the data request is sent to the bes. Approximately 0.04×4 seconds are spend on replying to the same showinfo request.

The 0.04 seconds can be reduced by profiling the bes and dap-server application. The number of requests currently 4 can be reduced by adapting the olfs application. We'll try to decrease the number of requests send from olfs to bes.

The showinfo request originates from the call in the file BesXmlApi as can be seen in listing 4

Listing 4: BesXmlApi.java

```
    public static boolean getInfo(String dataSource, Document response)
        throws
            PPTException,
3           BadConfigurationException,
            IOException,
            JDOMException {

        boolean ret;
8       Document request = showInfoRequest(dataSource);

        ret  = besTransaction(dataSource,
                  request,
                  response);
```

This is called by the constructor of BESDataSource

Listing 5: BESDataSource.java

```
        Document info = new Document();

5


        if(BesXmlAPI.getInfo(dataSourceName,info)){

            exists      = true;
10          accessible  = true;
```

The BESDataSource is constructed by the DapDispatchHandler

Listing 6: BESDataSource.java

```
    public DataSourceInfo getDataSourceInfo(String dataSourceName)
        throws Exception {
3       return new BESDataSource(dataSourceName);
    }
```

5 Suggested improvement

To limit the number of times the showinfo request is sent we can limit the number of times the BESDataSource is constructed. This can be done using memoization [1]. It can be implemented using a hash table as shown in listing 7.

Listing 7: Using a hash table to cache BESDataSource objects

```
Index: src/opendap/bes/DapDispatchHandler.java
===================================================================
--- src/opendap/bes/DapDispatchHandler.java (revision 20022)
+++ src/opendap/bes/DapDispatchHandler.java (working copy)
@@ -64,6 +64,8 @@
     }

     private HashMap<Pattern,Method> dispatchMethods;
+    // Cache data sources if  possible
+       private HashMap<String, BESDataSource> besDataSources;



@@ -80,8 +82,9 @@
          _servlet  = ds;
```

[1] http://en.wikipedia.org/wiki/Memoization

```
        dispatchMethods = new HashMap<Pattern,Method>();
+       besDataSources = new HashMap<String, BESDataSource>();
+


-
        registerDispatchMethod(".*.ddx",     "sendDDX");
        registerDispatchMethod(".*.dds",     "sendDDS");
        registerDispatchMethod(".*.das",     "sendDAS");
@@ -269,7 +272,16 @@


    public DataSourceInfo getDataSourceInfo(String dataSourceName)
        throws Exception {
-       return new BESDataSource(dataSourceName);
+       // Store datasources by name in a hashmap for quick retrieval.
    Constructing a besdatasource is an expensive operation.
+
+       if (!besDataSources.containsKey(dataSourceName))
+       {
+               besDataSources.put(dataSourceName, new BESDataSource(
    dataSourceName));
+       }
+       // we could create a local variable
+
+       return besDataSources.get(dataSourceName);
+
    }
```

There's probably a more java based way to do this, maybe using annotations or closures.

6 Testing

The performance test script as listed in listing 1 is again used to test the performance after the application of the changes listed in the previous section. The measurements of the response times are significantly lower with an average of 0.045 seconds and a standard deviation of 0.00339. The measurements of both pre and post patch response times are shown in figure 2.

7 Analysis

By caching the creation of BESDataSource onlybjects a speed increase of up to a factor 8.9 can be reached. This only applies to small requests. Also the first

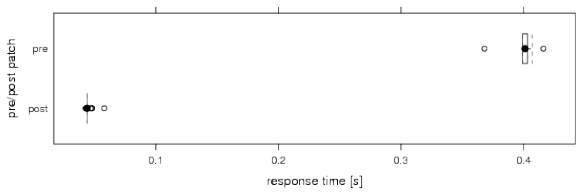

Figure 2: Response times for ascii web request (lower is better)

time a BESDataSource is accessed by a thread from the olfs the response will be twice as long because the data source is read and the BESDataSource object is created. Resulting in a speed increase of only 4.5.

A possible caveat is that if a BESDataSource object is cached and data is changed or removed on disk this might lead to unexpected behaviour using this approach.

Another thing to look at is the ammount the memory footprint increases by caching the BESDataSource objects.

The speed of small requests can further be improved by limiting the 0.04 seconds it takes for the bes to respond to a request. This can be done using by profiling the bes, libdap and dap-server applications.