

# High speed transfer of large files between HPCx and HECToR using bbFTP

K.J.D'Mellow, S.P.Booth

*EPCC, The University of Edinburgh, James Clerk Maxwell Building,  
Mayfield Road, Edinburgh, EH9 3JZ, UK*

October 20, 2009

## **Abstract**

It is increasingly common that extremely large datasets are produced on HPCx, and the subsequent transfer of these - either to one's research institution, or migration to HECToR - represents a significant challenge. We address the problem of migration of data to HECToR, in anticipation of this being commonly required in the closing months of the HPCx service. HPCx and HECToR are connected with a 1Gb/s end-to-end link over a 10Gb/s backbone. We attempt to achieve as much of this available bandwidth using the bbFTP package.

**This is a Technical Report from the HPCx Consortium**

**© HPCx UoE Ltd 2009**

Neither HPCx UoE Ltd nor its members separately accept any responsibility for loss or damage from the use of information contained in any of their reports or in any communication about their tests or investigations.

# 1 Overview

Commonly used (and recommended) methods of transferring files are FTP, and SCP. FTP is generally insecure, as it transmits authentication details over the internet in an unencrypted manner. For this reason FTP is prohibited on a number of systems including both HPCx and HECToR. A secure equivalent of FTP is the SCP utility (secure copy), which conducts file transfer within an encrypted SSL datastream. This has the advantage of high-security, but requires significant computational overhead at both ends of the connection, encrypting and decrypting the data. This computational overhead is the limiting factor in achieving high transmission rates, and typically, rates of no more than 10Mbytes/s are observed.

bbFTP [1] is an open-source secure multistreamed file transfer utility, released under the GNU General Public License. It was written by Gilles Farrache at IN2P3 Computing Center in Lyon, France. bbFTP has been optimised for the transfer of large files: it securely authenticates, but then transfers file data in multiple unencrypted streams, thereby avoiding the computational overhead of encryption and decryption. bbFTP also allows large windows as defined in RFC1323, and therefore is well suited to the problem. Previous studies of grid-ftp [2] [3] have shown that there is good scope for improvement over SCP performance. grid-ftp however, requires a non-trivial certificate based authentication mechanism that is often too large a barrier for user uptake. bbFTP allows password or public key based authentication in the same manner as scp, so is immediately accessible to users with minimal setup costs.

This report investigates optimising the bbFTP utility to make the most of the high speed dedicated link between HPCx and HECToR.

## 2 Tests Performed

As bbFTP makes use of multiple streams, a series of transfers were performed to determine the effect of stream number and file size on the transfer rate. All transfers have been performed between the HPCx and HECToR national facilities, and the time of each individual test has been determined over five transfers. For the majority of tests, the file sizes range between 46Mb and 1.4Gb, increasing in powers of two, and the number of concurrent streams range between 1 and 10.

To satisfactorily test both the client and server code on both architectures, we also test for any performance differences between issuing for example, a “put” from the source machine, versus a “get” from the destination machine. We do this in both directions.

bbFTP will support compression of the data-stream, but deliberately do not test this. As is noted by the bbFTP authors, compression and decompression of the data-stream requires a significant overhead and is not beneficial in the case where the link speed between sites is greater than 34Mb/s [1]. A couple of quick checks have verified that even with highly compressible data (ascii text), compression does indeed hinder the transfer rate between HPCx and HECToR. We therefore investigate this no further.

### 2.1 bbFTP mode of operation

There are a number of ways on initiating a bbftp transfer, but for user and administration simplicity, we use and recommend initiating both the client and server in the same

command, and using bbFTP's SSL based authentication (like scp). This allows the user to initiate a transfer in one command, and remain confident in the knowledge that passwords/authentication details are being handled by the trusted SSL protocol. As is common, if the user has established public key based authentication with ssh-agent or similar, then bbFTP will use this to login. Alternatively the user will be prompted for the password to the remote account as usual.

When run like this, the bbFTP client is run locally, and the server half of bbFTP needs to be executed on the remote site. Therefore it is necessary for the user to also supply the location to the remote server executable.

A typical test command is as follows, and is explained below:

```
/usr/local/packages/bbftp -s -E "/usr/local/packages/bbftpd -s -m 10"  
-m -V -p 10 -e "put filename.dat" login.hpcx.ac.uk
```

The above was issued from HECToR, in a "put" to HPCx. The client application, bbFTP, takes the `-s` flag to indicate the use of SSL, the `-E` flag to indicate the location of the server executable on the remote site (as can be seen above, this is a valid path on the HPCx filesystem), the `-p` flag to indicate the number of streams to be used, and the `-e` flag to indicate the command to be issued. In testing we also used the `-m` and `-V` flags for verbose output and statistics. The server specification also requires some flags, and we used `-s` for SSL, and `-m` to specify the number of streams the server can handle (note that this is nearly equivalent to the `-m` client flag).

When executed, this single command connects to the remote host using SSL, and executes the bbFTP server, which in turn publishes its port range and stream capabilities back to the client. The unencrypted transfer is then conducted across the specified number of streams using the server's port range.

### 3 Results

The put and get commands for a series of files were issued in either direction between HECToR and HPCx. Firstly, the maximum measured bandwidths from a series of five trials is given in Figure 1. Demonstrably stream count is important here, and these tests show that a near linear speedup is achievable by using multiple streams. As the HPCxHECToR transfers occasionally fail beyond 12 streams, we limited our benchmarking to 10. This indicates a maximum stream bandwidth of over 50Mbits/s.

The per stream bandwidths indicate that on smaller transfers, the transfer rates do not achieve full speed before the transfer is over. As observed in Figure 2, one can attain close to the maximum observed transfer rates when transferring files of around 200Mb and over.

A significant discrepancy was observed in transfer rates into HECToR (labelled Northward), compared with those into HPCx (Southward). Irrespective of which side the client is, and which command is issued, Northward transfers are a factor of 4 slower than Southward equivalents. A traceroute between the services yields differing paths in each direction, and we suspect that it is for this reason, that the transfer rates are different. This situation is currently beyond the control of the HPCx or HECToR support teams.

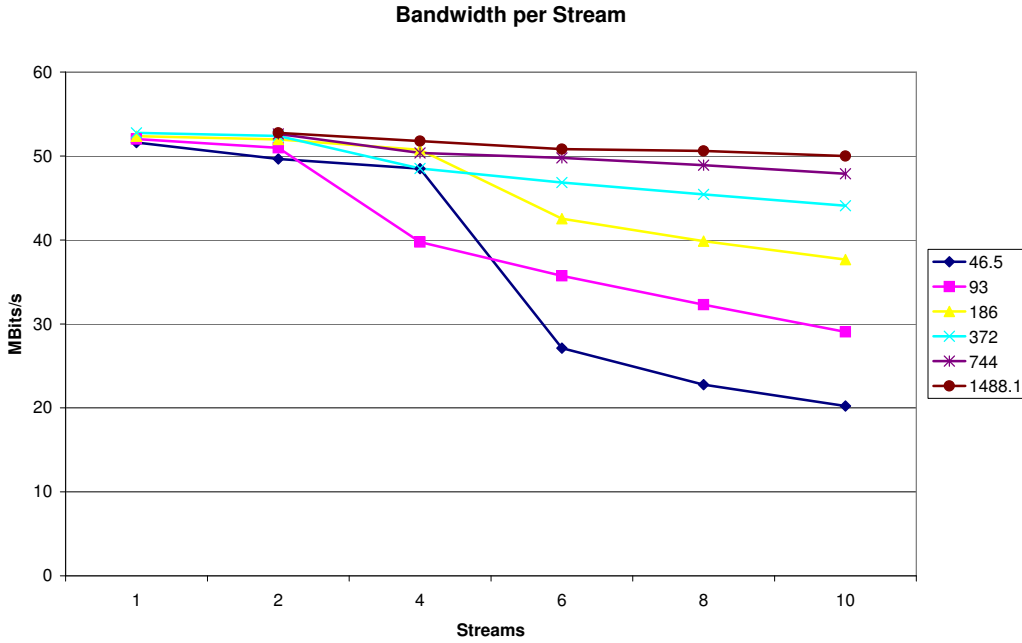


Figure 1: Maximum bandwidths observed for a variety of file sizes from 45Mb to 1.4Gb. The utilisation of multiple streams gives an near linear increase in bandwidth.

## 4 Issues

During the benchmarking process, a small number of issues were encountered. These are listed here. At present, none of the issues encountered impact significantly on the usability of bbFTP.

### 4.1 Stream limits

In principle, it should be possible for an arbitrary number of streams to be opened, and a process opened for each stream, allowing parallel execution of both the client and server. In practise, we found that a Maximum of 12 streams were possible when instigating on HPCx (clientside), while 16 streams were possible with HECToR clientside. Moreover, above 12 streams in any direction, bbFTP would occasionally fail, although this failure happens at the start, and not mid-transfer. We have not had time to investigate this further.

### 4.2 Ssh key failure on the HECToR system

Experienced clientside on HECToR only, bbFTP will occasionally corrupt the ssh key environment, in the process of execution. The corrupting instance will generally succeed, but subsequent ssh-key based authentication will not (for any application that uses this). Closing and reopening one's session (and hence resetting the environment) resolves this issue. We have experienced no such issues on HPCx.

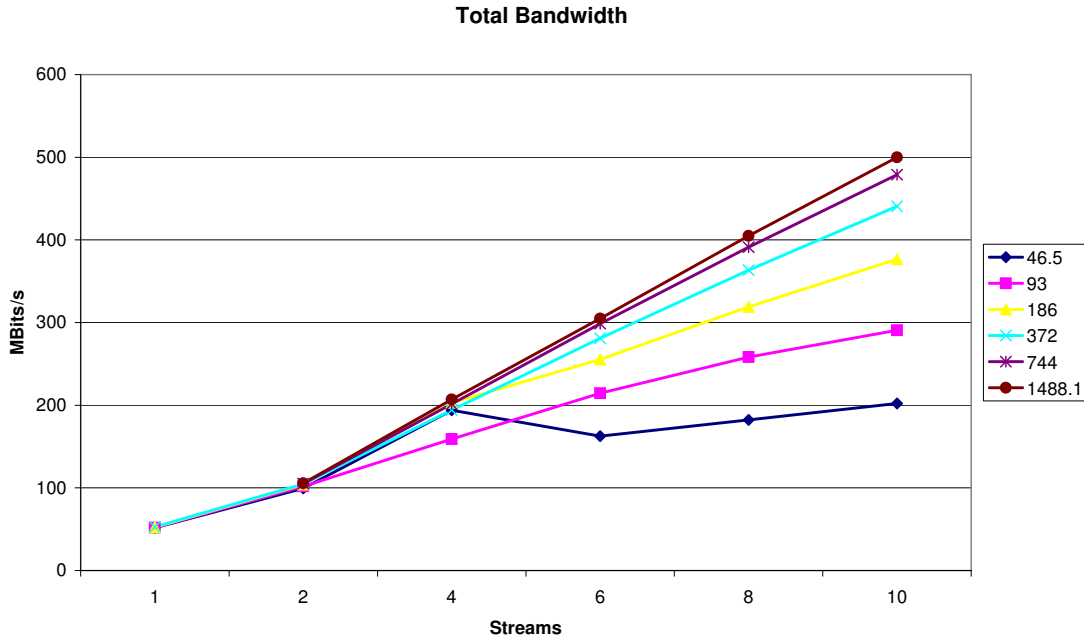


Figure 2: Total Bandwidth as a function of transfer size. Smaller transfer sizes do not attain the maximum transfer rate before completion, yielding lower transfer rates. File sizes of 200Mb+ are required to achieve the maximum bandwidth available.

### 4.3 Occasional errors

We have witnessed occasional errors when running bbFTP, but investigations indicate that these are generally not immediately repeatable or severe.

```
BBFTP-ERROR-00101 : Incompatible daemon and client (remote protocol version (1,1), local (2
Connection to login12.hector.ac.uk closed by remote host.
BBFTP-ERROR-00061 : Error waiting MSG_OK (on MSG_TRANS_START_V2) message
```

These errors appear infrequently, and do not cause corruption or damage on either filesystem. Moreover, if bbFTP issues BBFTP-ERROR-00061, it will still complete successfully, although the total time taken to execute will include one or several server timeouts.

## 5 Conclusions

bbFTP is a fast, multistreamed file transfer utility available on both HPCx and HECToR. It uses SSL for login and authentication, but transfers the payload data in clear-text, herefore avoiding the computation overhead of encryption and decryption. This means

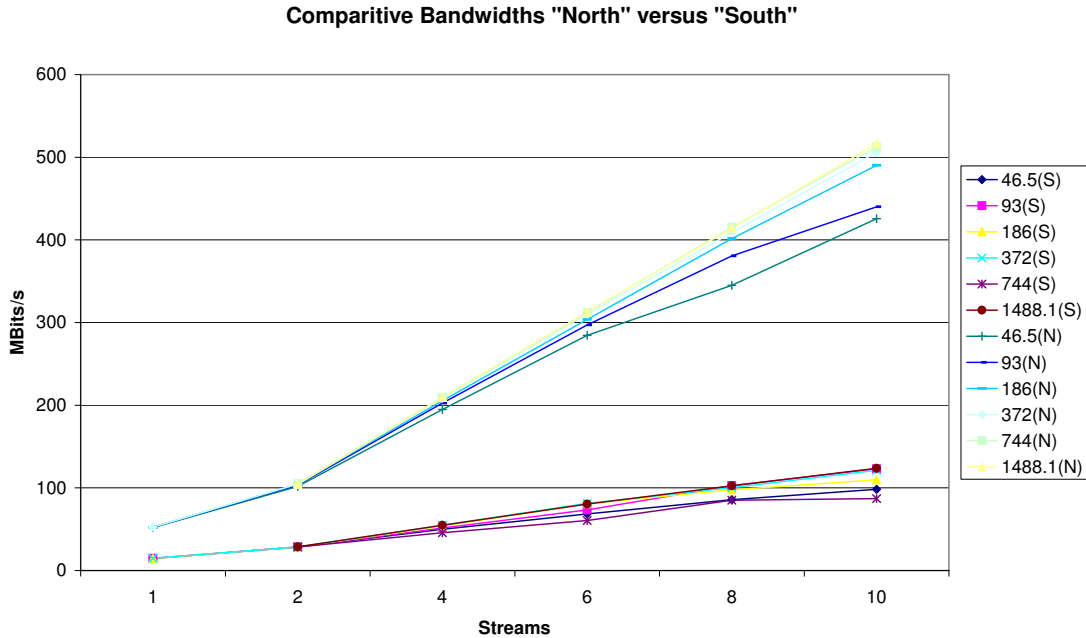


Figure 3: Northward versus Southward bound bandwidth. The discrepancy is nearly a factor of 4, but is due to differences in routing between traffic to and from HEC-ToR/HPCx. This is consistent across put and get commands, to and from either service.

it can achieve much faster transfer rates than SCP, and in a secure fashion. Its performance is similar to that of globus' grid-ftp, but unlike this, bbFTP uses a simple password or public-key authentication, so is as readily accessible to end-users as scp, and does not require the creation of certificates. The bbFTP client and server (bbftpd) are both executable from a single command on the client-side, and are available for use in /usr/local/packages/bbftp on both HPCx and HECToR services.

## References

- [1] IN2P3 Online documentation for bbFTP 3.2.0  
<http://doc.in2p3.fr/bbftp/doc.3.2.0.html>
- [2] Transferring large files to and from HPCx: a comparison of (parallel) 'gridftp' and 'scp'. C.Johnson, S.P.Booth, HPCx Technical report, 2006.  
[http://www.hpcx.ac.uk/research/hpc/technical\\_reports/HPCxTR0603.pdf](http://www.hpcx.ac.uk/research/hpc/technical_reports/HPCxTR0603.pdf)
- [3] Transferring large files to remote sites: an update. C.M.Maynard, S.P.Booth, HPCx Technical report, 2007.  
[http://www.hpcx.ac.uk/research/hpc/technical\\_reports/HPCxTR0702.pdf](http://www.hpcx.ac.uk/research/hpc/technical_reports/HPCxTR0702.pdf)

## Appendix A - Installation of bbFTP

We briefly outline the procedure for installation of bbFTP and the options we used. This may be of interest to users who wish to transfer large files to and from their host institutions. In all instances, refer to the documentation supplied at [1].

To transfer files to or from either HPCx or HECToR, users only need to install the bbFTP client, as the bbftp server is already available on these services. Users who wish to transfer files between two hosts not including HPCx and HECToR should also install the bbFTP server.

### Client installation

1. Download and unpack the bbFTP client. We used version 3.2.0, available from `ftp://ftp.in2p3.fr/pub/bbftp/bbftp-client-3.2.0.tar.gz`
2. In the bbftpc subdirectory, run `configure`, noting the use of the non-standard install directory.

```
$ ./configure --prefix=~/.bbftp/install/client-3.2.0
```

3. “Make” the server executable, which will appear in the chosen installation directory

```
$ make clean
```

```
$ make ; make install
```

Note that on HECToR, `configure` will pick up the back-end compiler, so prior to “make”, you should edit the Makefile and substitute the compiler with “`CC = gcc`”

### Server installation

Note that server installation is only required on the host machine. The bbFTP servers already exist on both HPCx and HECToR as described in section 2.1.

1. Download and unpack the bbFTP client. We used version 3.2.0, available from `ftp://ftp.in2p3.fr/pub/bbftp/bbftp-server-3.2.0.tar.gz`
2. In the bbftpd subdirectory, run `configure`, noting the use of the non-standard install directory. We chose to not use the RFIO interface as it is neither required, nor supported on HPCx. Installation of the server also requires specification of the ephemeral ports range - open ports that are used to instantiate data transfer. Here, XXXXX and YYYYY span a range of around 100 open ports.

```
$ ./configure --prefix=~/.bbftp/install/client-3.2.0 --without-rfio
  --with-ephemeral-ports-range=XXXXX:YYYYY
```

3. “Make” the server executable, which will appear in the chosen installation directory

```
$ make clean
```

```
$ make ; make install
```

Note that on HECToR, `configure` will pick up the back-end compiler, so prior to “make”, you should edit the Makefile and substitute the compiler with “`CC = gcc`”