FTS Transfer Parameter Optimisation

Ziwei Chen

CERN openlab
15-08-2012

oTN-2012-07 openlab Summer Student Report

FTS Transfer Parameter Optimisation

Author: Ziwei, Chen
Supervisor: Oliver, Keeble
20.07.2012
Version 2
Distribution: **Public**

Abstract

This report describes the development of an FTS (File Transfer Service) [1] third party transfer parameter optimization algorithm and the aim is to figure out the optimal configuration parameters which produce a better performance with a heuristic approach and historical transfer information.

1 Introduction

The gLite File Transfer Service (FTS) is a low-level data movement service for transferring files between Storage Elements. In addition, it provides features for administration and monitoring of these transfers. The FTS exposes an interface to submit asynchronous bulk requests and performs the transfers using either third-party GridFTP or SRM Copy. The FTS servers are typically deployed at (large) sites where there are large amounts of data to be transferred.

In the near future, the FTS2 currently using by CERN will be completely upgrade to FTS3. FTS3 support transfer without specify parameters, but as a constant default configuration might not be suitable for all transfer channels (source and destination pair), an auto-tuning mechanism is introduced.

The investigation of this report is focused on third party transfer through GridFTP protocol. GridFTP is an extension of the standard File Transfer Protocol (FTP) for use with Grid computing. It is defined as part of the Globus toolkit [2], under the organisation of the Global Grid Forum. GridFTP provides security with GSI (Grid Security Infrastructure), fault tolerance and restart, and support third party transfer, parallel and striped transfer, and partial file transfer.

The aim of this research is to develop an algorithm to figure out an optimal configuration which achieves a better transfer performance in different network environment. There are three highly related parameters of transfer configuration are proposed in the beginning of the research. They are number of streams per transfer, TCP buffer size and number of simultaneous transfers. The correlation of number of streams, TCP buffer size and goodput are widely investigated in many researches, which will be discussed in section 2, and a few strategies and solutions suit for FTS3 will be introduced in section 3, follows by the experiment and analysis in section 4, and the conclusion in the end.

2 Related Work

2.1 Optimal Number of Streams

Since GridFTP supports simultaneous transfers, the total number of TCP streams at a time is split to different transfers. In practice, it will produce a linear increment of goodput [3] (application level throughput) between 2 and 10 streams per transfer. Afterwards, additional streams have very little impact. But higher bandwidth, longer round trip times, and more congestion in the network will move this range higher.

From the equation 1 [4], we can easily obtain the optimal total number of streams by current bandwidth bottleneck, round trip time and TCP buffer size, the threshold of TCP buffer size equals to Bandwidth Delay Product (Bandwidth Bottleneck * RTT), I will explain the derivation of bandwidth and RTT in the following sections.

$$N = (3BR - 3W - \sqrt{3}\sqrt{9B^2R^2 - 16BRW + 7W^2}) \times \frac{BR}{9BR - 6W} \quad (1)$$

Apart from the highest goodput we can achieve, the transfer reliability also has to be taken into account. When an unexpected lost of connection happens, the more file transferring concurrently, we lost more file in that situation. Therefore assign a proper number of streams to each transfer is another main factor of this algorithm. As I observed, the streams used in past transfers are vary from 2 to 20, but the most frequently used number is 10, which covers more than half configurations. So I decided to assign 10 streams to each transfer, and the number of simultaneous transfer can be simply derived by number of total streams (n)

divided by number of streams per transfer (npt). Additionally, since FTS implementation makes transfers directly goes to disk server regardless of the head node of DPM, which refers that assign a different disk server to each of the simultaneous transfer makes great contribution on performance.
There is another constraint of streams, each storage server can only allocate a certain number of streams for both outbound and inbound transfers, if the derived total number of streams exceeds the upper bound number of streams can be allocated, we have to abandon the exceeded part as illustrated in Figure 1.

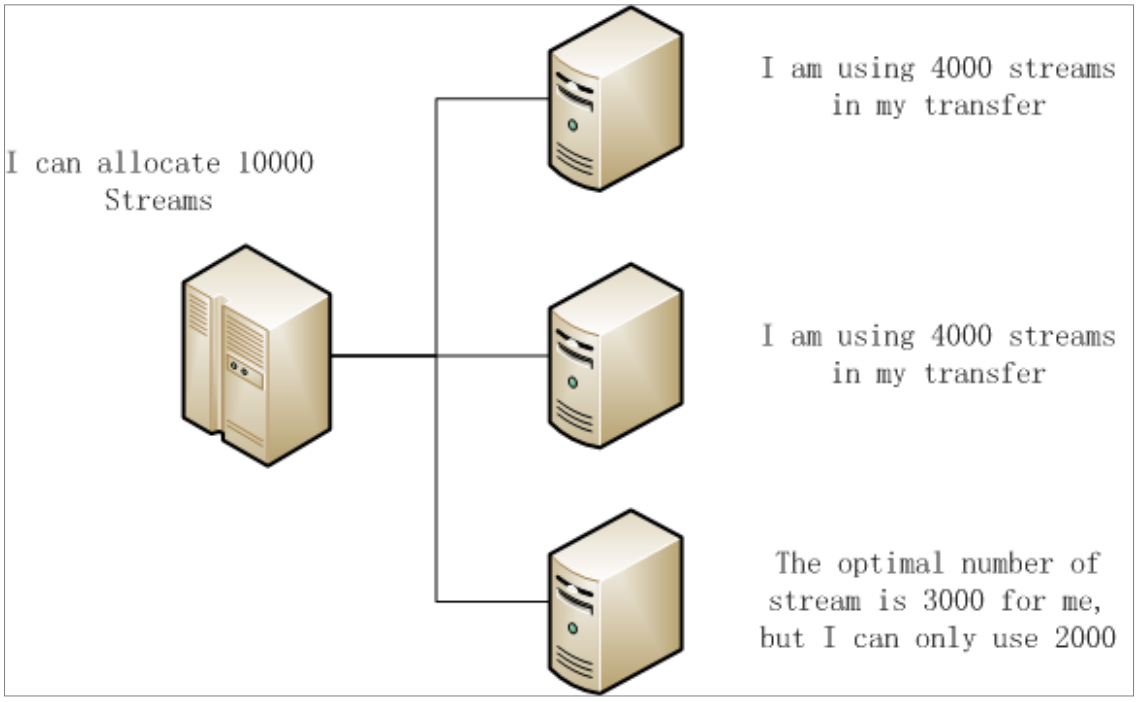


Figure 1: Stream allocation constraint

2.2 Estimation of RTT

As observed in experiment, the round trip time (RTT) is a very sensitive factor in equation 1, and the result of optimal number of streams highly relies on the accuracy of RTT estimation. perfSONAR[5], an infrastructure for network performance monitoring, will be ready to provide precise RTT and bandwidth for each transfer to FTS3 in the near future, so it is worthy to figure out a cheap and acceptable method to fill the gaps before perfSONAR gets ready. Here we have two candidate approaches.

2.2.1 Geographic Estimation

In this approach it is checked if the source and destination host are located in the same city, country or continent with GeoIP library [6]. But as I observed, the correlation of geographical distance and RTT is hard to predict if two hosts located on different continents. Within the same continent, we can approximate they have a linear correlation, but on different continents it is extremely unpredictable. For instance, from CERN to Moscow, Beijing and Taipei, the distances are similar. But as I tested the ping value, the round trip time between CERN and Moscow is just 180ms but to Beijing is around 600ms and Taipei 300ms. Fortunately, the impact of RTT becomes lower when it is above 200ms as I experiment, thus we can still obtain a reasonable result by this approach. Because of the independency and low cost, we temporarily integrated this estimation mechanism in the prototype for testing.

2.2.2 Estimation with King [7]

This tool provides highly accurate latency estimates without any additional infrastructure, and the only need is an empty domain used to play a trick among name servers.
As shown in figure 2 (retrieved from [7]), the basic idea is to fool name server A into believing that name server B is the authoritative name server for a domain owned by the client C. Suppose, for example, that client C owns *mydomain.com*. Let NS C denote the authoritative name server for this domain, and

assume the IP address of name server B is 10.0.0.0. First, C issues a recursive query R1 to A to resolve name servers for domain *10-0-0-0.mydomain.com*. This sub domain belongs to *mydomain.com*, and the query is forwarded to our name server NS C. Since we control NS C, we program it to reply that 10.0.0.0 (i.e., name server B) is the authoritative name server for the sub-domain *10-0-0-0.mydomain.com*. Name server A caches this reply, and forwards it back to client C.

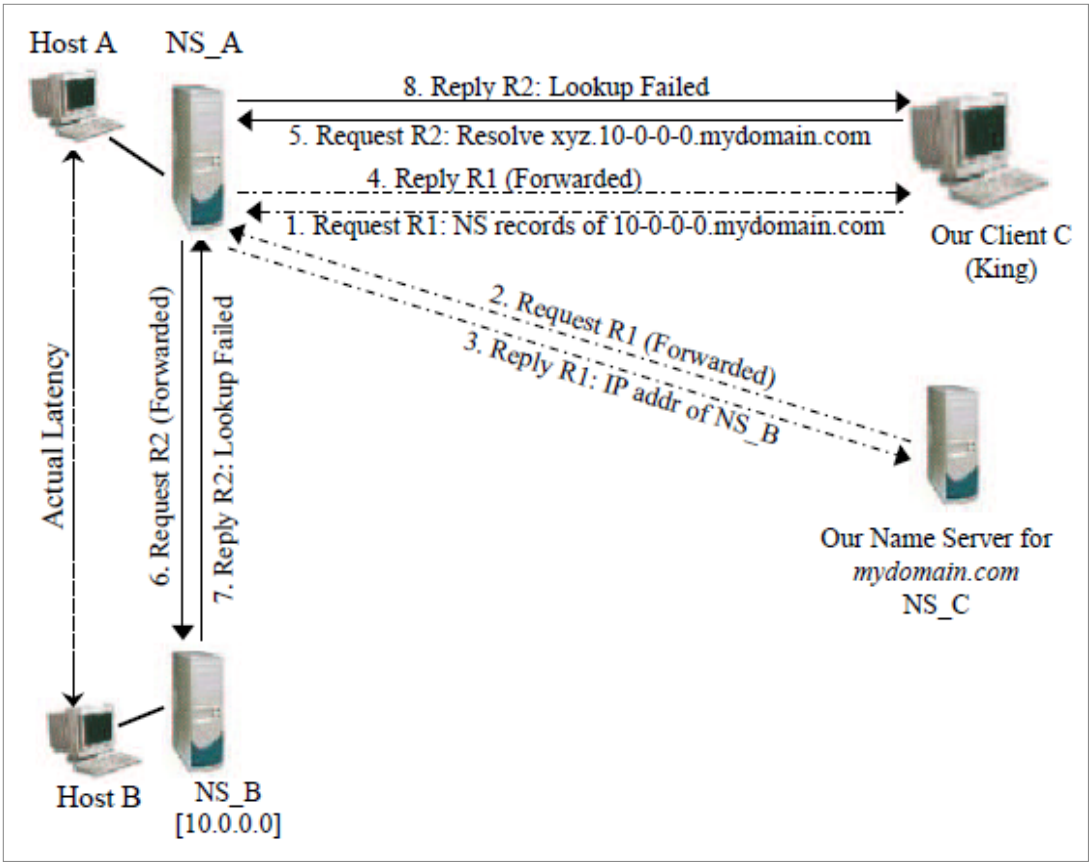


Figure 2: King tool working theory

2.3 Other Ideas

There are a many command passed between client, source host and destination host during a third party transfer, such as, USER, PASS, SITE, FEAT, TYPE, MODE, SIZE, OPTS, NOOP and PBSZ[8]. But there is just one command passed between source and destination which is RETR, unfortunately, there is no API exposed in Globus library to return the response time by this command.
Also latency estimation tools like GNP and IDMAP are both very useful, but GNP needs approval by both source and destination side and IDMPA needs some extra infrastructure, therefore both of them are not adopted by FTS3 development.

3 Solution

A LHC (Large Hadron Collider) Computing Grid system consists of three tiers, The CERN computer center is considered "Tier 0", the core of the system where most of data comes from, there are 11 tier 1 site located in Europe, Asia, and North America, via dedicated 10 Gbit/s links. This is called the LHC Optical Private Network. More than 150 Tier 2 institutions are connected to the Tier 1 institutions by general-purpose national research and education networks.
Since network environments between Tier 0 and Tier 1 are relatively stable, and latest data are transferred incessantly, a cheaper, low overhead and reliable strategy is enough. But between tier 1, tier 2 and among tier 1s, a more adaptable and flexible algorithm is required. Therefore our algorithm should be flexible for both cases.

3.1 Algorithm Design

In this algorithm, the current available bandwidth is estimated heuristically with the last goodput of transfer, the optimal number of simultaneous transfers come up with bandwidth, RTT and TCP buffer size used. The process is shown in Figure 3.

- Before the first transfer, FTS client will pass the source and destination host address into the algorithm and the algorithm will estimate the RTT (if there is no RTT specified), produce a default number of simultaneous transfers and TCP buffer size, if streams per file is not set, then use 10 as defaul. The other related variables will be initialized as 0. Then FTS client will start first transfer with the default parameters, and finally retrieve the average goodput of this transfer.
- If it is not the very first transfer of auto-tuning algorithm, which means the previous goodput, number of simultaneous transfers and TCP buffer size are initialized, the algorithm will first check if there are already 100 samples collected (we use 100 as an example here), a sample can be the average performance of 10 or 20 recently completed transfers.
  - If the number of samples has not reached 100 yet, the algorithm will compare the current goodput with the previous one.
    - If the previous goodput is higher, which means either the last configuration is better or there is a spike in the network traffic, and the last configuration will be restored.
    - If the current goodput is better, the current values (goodput, number of streams and TCP buffer size) will be assigned to those previous variables, and a new optimal number of simultaneous transfers and TCP buffer size will be calculated from the latest estimated bandwidth (current goodput multiplies a constant value depends on RTT).
  - If 100 samples are collected, the algorithm will not continue transfer but comes to find out which number of simultaneous transfers is most common used among these 100 samples. And compare the average goodput of most common used configuration with average goodput before auto-tuning is applied.
    - If the new goodput goes better, the algorithm will return the most common used number of simultaneous transfers and calculate TCP buffer size by RTT and new average goodput as an optimal result.
    - Otherwise if the goodput doesn't change obviously or even goes down, the result will be ignored and FTS keeps using the original configuration.

3.1.1 Alternative Termination Method

Apart from collecting 100 or a certain number of samples, there is another method to terminate the algorithm and find out the optimal result, which is a convergence approach.
In this method, we can put a counter to count if the result keeps stable in a small range for a certain times, we believe the result converged and reached the optimal. The advantage of this approach is that sometimes the algorithm terminates in a very short of time, and no extra work after the user start the program, which is more automatic than the current termination method of the implementation. But the problem is that if the network bandwidth is very unstable, the algorithm is unlikely to converge and the result is very unpredictable.

3.1.2 Algorithm Usage

In FTS3, the definition of channel will be deprecated, so that there would not be any default configuration inherit from FTS2. In the situation, this algorithm is very useful to be executed at least once to find out a proper default configuration. If the network is known very unstable or the routing path is changed sometime, especially transfer between tier 2 and tier 2, also between tier 1 and tier 2, this auto-tuning algorithm can be applied more frequently.

3.2 Implementation Details

3.2.1 RTT Estimation by GeoIP and Initialization

The RTT is estimated by GeoIP library. GeoIP library provides two free edition database, they are GeoLite City and GeoLite Country. There are also more accurate commercial editions which can

determine the name of organization, ISP and even Netspeed (dial-up, cellular, cable/DSL, and corporate). The implementation uses only first two free databases, and currently, if two hosts are in the same city, the RTT is set as 5ms; same country is 30ms; same continent is 50ms; and different continent is 300ms.
When the RTT is approximated, the default number of simultaneous transfers and buffer size are also decided, currently if RTT is less than 0.01, we start from 1 simultaneous transfers and 1M buffer size; if RTT is between 0.01 and 0.1, 2 simultaneous transfers are operated with 2M buffer size; otherwise we allocate 4 simultaneous transfers and 4M buffer size.
P.S. To use GeoIP library is very simple, it can be easily downloaded from the GeoIP official site in reference, and when compile and execute the binary, don't forget to add the related library file as linker object and add the library path to LD_LIBRARY_PATH.

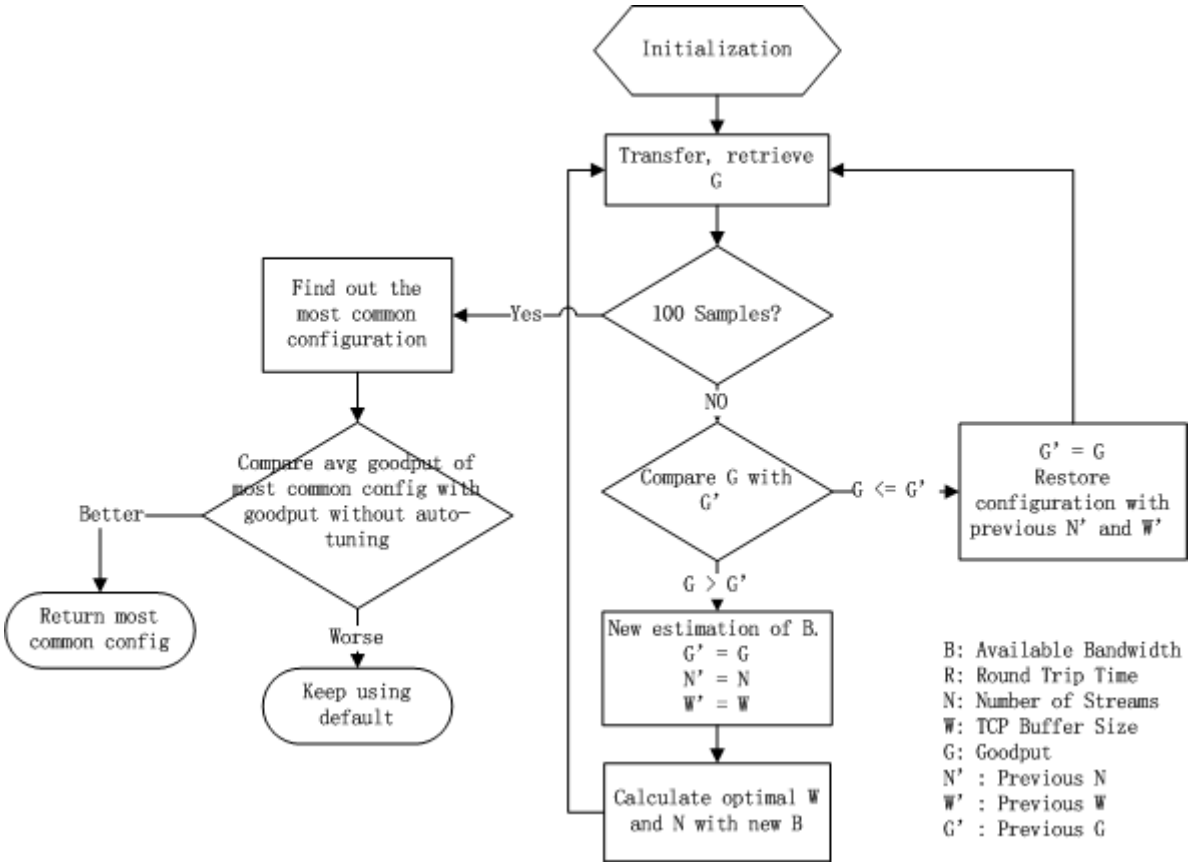


Figure 3: Heuristic Algorithm Process

3.2.2 Heuristic Algorithm

The core of this algorithm is to compare the current with previous goodput, then decide the parameters for next attempt. As mentioned before, if the goodput goes worse, the next configuration will be restored with the previous one; on the contrary, if the goodput goes better, a new available bandwidth will be estimated and the approximation of current available bandwidth heavily affect the efficiency of the heuristic process. In the algorithm, if RTT is less than 0.01, the bandwidth is estimated as latest goodput multiply 1.2; if RTT is between 0.01 and 0.1, the factor becomes 1.3; otherwise it increases to 1.4. Meanwhile there is a valid domain of TCP buffer size which is 1M to 16M. And a new optimal number of simultaneous transfers come up with equation 1.
Additionally, the unit of each parameter is shown in table 1.

Goodput	Bandwidth	RTT	Buffer Size
MByte/sec	Mbyte/sec	sec	KByte

Table 1: units of parameters

3.2.3 Determine the Result

When the 100th transfer finished, the database will pass a list of completed transfer information to the algorithm, the most common used number of simultaneous transfers will be discovered, and average goodput of it will be calculated to compare with the original goodput, which is either passed into the algorithm in the very beginning or set to 0 as default. The better configuration will be finally adopted.

The method to figure out the most common used number of simultaneous transfers is as following:

- Find out the highest number (h) of simultaneous transfers in the 100 samples.
- Allocate a 2 dimension array (h * 2), the subscription represents the number of simultaneous transfers, and first dimension is applied times, second dimension is average goodput of this number of concurrent transfers regardless TCP buffer size. As observed, the maximum number of h never exceeds 300; therefore we don't need to worry about memory problem.
- When the most common used value is found out, compare the goodput and return a better one.

4 Evaluation Result

Based on the auto-tuning algorithm, all experiments are taken between CERN and Taiwan, using globus-url-copy. The RTT between CERN and Taiwan are tested and set manually as 280ms. Since GridFTP third party transfers first go for a head node and then to an available disk server, but FTS always makes data directly transfer to a disk server. Therefore the algorithm is evaluated on both route, and both directions between CERN and Taiwan.

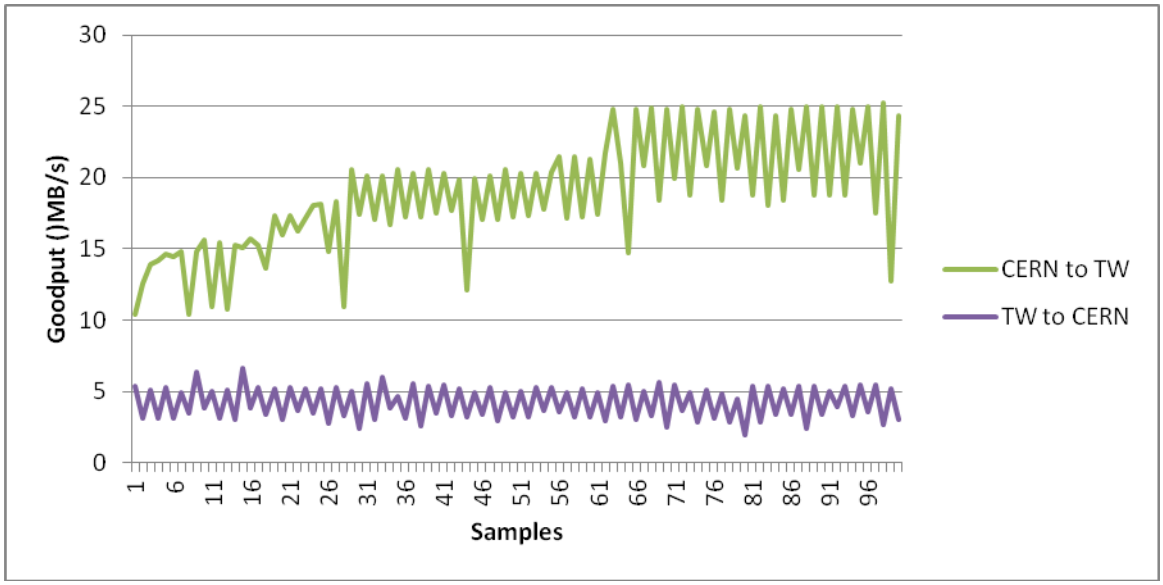
The available bandwidth changes fast from time to time between CERN and Taiwan, but RTT keeps stable as 280ms all the way. That is a big challenge to the adaptability and scalability of the algorithm, as discussed in the algorithm design, the result is unlikely to converge if the available bandwidth changes to fast in the network, what we can do is to adjust the configuration with the current network situation, instead of a roughly estimation at the beginning.

Another important criteria is the accuracy, the accuracy of the result highly depends on the accuracy of the current available bandwidth and RTT. With a heuristic approach we can eventually obtain an approximated bandwidth, in terms of RTT estimation, even if this problem will be solved by Perfsona in the near future. The current accuracy of GeoIP method is not enough to get the most optimal result every time.

4.1 Goodput

100 samples are collected for each experiment, 1 sample represents 1 complete transfer here. The green line categorized as "CERN to TW" means the experiment is on globus-url-copy third party transfers from a dpm (Disk Pool Manager) endpoint at CERN to a dpm disk server of Taiwan, conversely the purple line is from TW to CERN.

As we can see in Graph 1, the goodput of CERN to TW case starts from a low value then rise up and keeps staying in a higher range; in terms of TW to CERN experiment doesn't end up with a better goodput, the reason can be either a wrong estimation of RTT or the algorithm starts with a good default. If the average goodput suddenly goes down to a lower scope in another period of time, which means the network available bandwidth probably has changed.

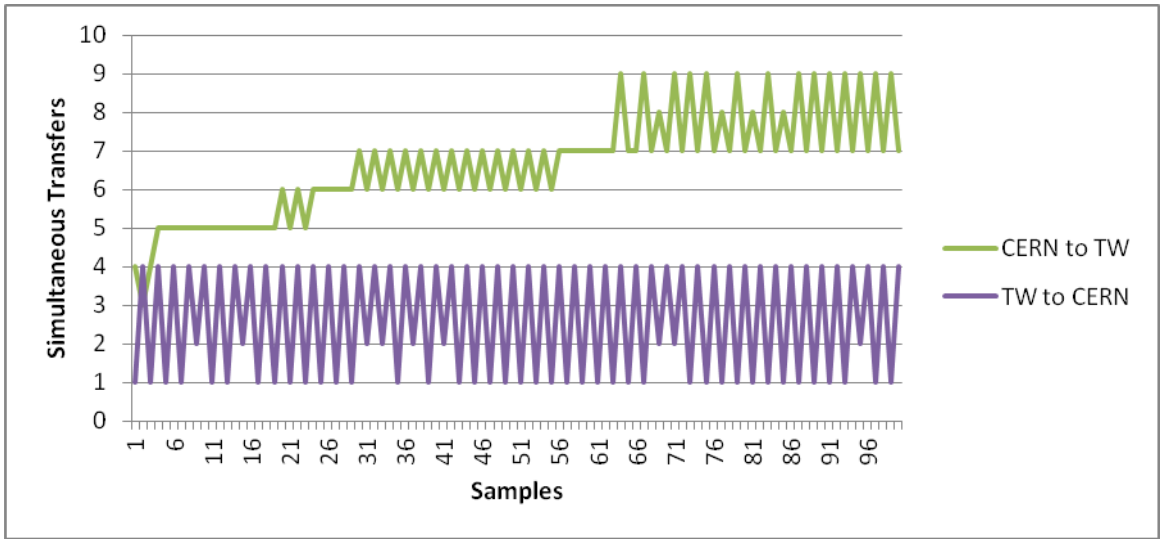


Graph 1: Goodput between Taiwan and CERN directly transfer to disk server

4.2 Simultaneous Transfers

Since the number of streams per transfer is fixed as 10, here we just take number of simultaneous transfers into account, so that the total number of streams can be only 10 * transfers, the rough input is good for the inaccurate estimation of RTT currently, and a new stream allocation algorithm can be developed when we have a better result of RTT.

As we can see Graph 2 is highly relevant to Graph 1, because the last goodput is the input of the algorithm to generate the optimal number of simultaneous transfers for next transfer. Similar with Graph 1, the green lines go well as expected; an optimal number can be easily decided afterwards. On the contrary, the purple lines floating in a scope, in this case we can either take a mean or median value of the result, both can certainly improve the performance of transfer, but we are not able to predict which one is better for the next moment.

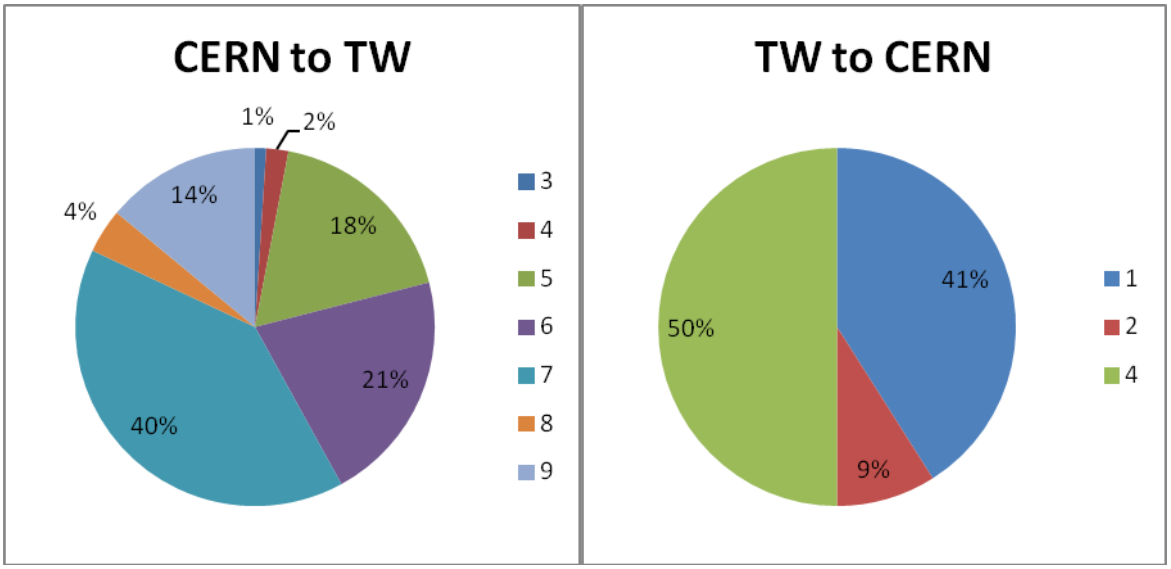


Graph 2: Number of simultaneous transfers between Taiwan and CERN

4.3 Result Distribution

The current approach to decide which optimal number of simultaneous transfers to adopt is to retrieve the most common used one. For instance, in the left of Graph 3, we can see in CERN to Taiwan experiment, the number of transfers 7 covers 40% of totally 100 samples, and the goodput produced with this configuration is fairly good, and then we can easily decide the optimal result. However in the Taiwan to CERN case, we

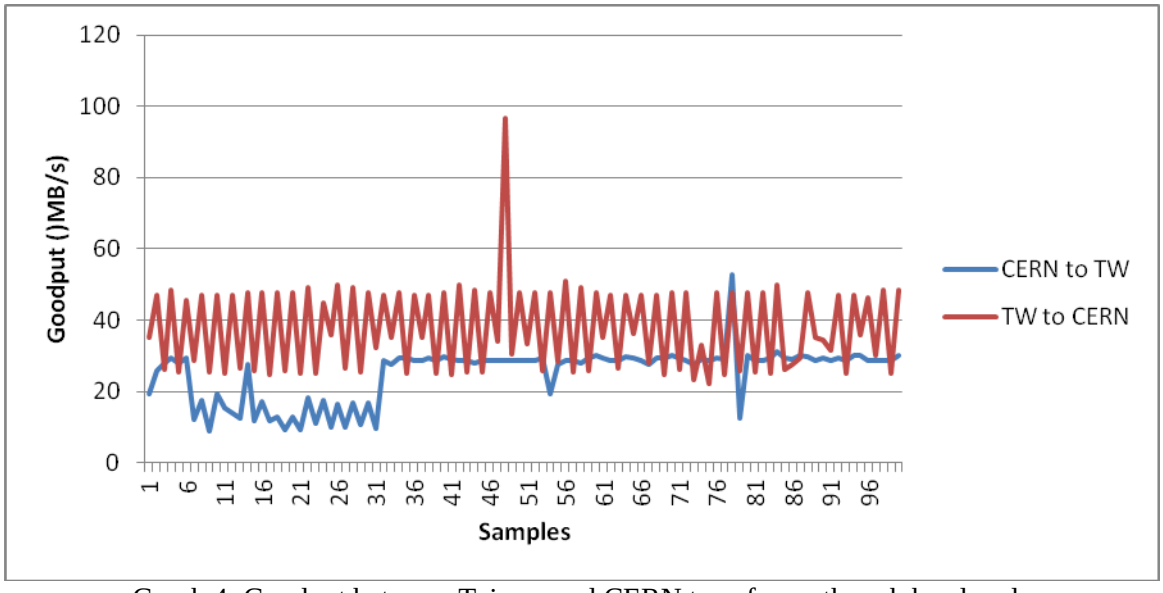
noticed that both 1 and 4 concurrent transfers are very common used, in that case we can simply take 4 as highest ratio, even if the goodput doesn't change to much, all according to the actual situation. When the optimal number of stream is decided, the algorithm will provide the average goodput as available bandwidth and easily obtain a proper TCP buffer size with RTT.



Graph 3: Distribution of number of simultaneous transfers

4.4 Further Experiments

Although FTS transfers always directly target to the disk server, it is also very interested to know how well the algorithm works if transfer go through a dpm head node. As we can see in Graph 4, the goodput increases significantly in CERN to TW direction when go through head node, we suspect that taking different routing path makes transfer go through head node generate even better goodput than directly to disk node sometimes. Also varying load of the target head node machine and disk server machine would become another factor.



Graph 4: Goodput between Taiwan and CERN transfer go though head node

5 Conclusion

In this report we have demonstrated an algorithm which is capable of optimising a set of default transfer parameters in order to maximise goodput. This will take FTS3 towards its goal of offering 'zero configuration' transfers. The optimization algorithm of FTS third party transfers is illustrated, a heuristic approach is applied in this algorithm with a roughly estimated RTT, and comes up with an optimal number of concurrent transfers and TCP buffer size, due to the number of streams per transfer is fixed at beginning. Size of file block is another parameter of globus-url-copy, but as we observed, it doesn't make too much difference in the experiment.
All experiments have been done relies on GridFTP protocol, to compare with the existing evaluation [9] from other papers with the similar source and destination pair, the result is valid and reasonable.
In addition, the algorithm is currently being integrated with FTS3, and will be tested in a real environment in the near future. More feedback and update would be followed at that time.

References

[1] https://svnweb.cern.ch/trac/fts3/wiki
[2] http://www.globus.org/toolkit/
[3] http://en.wikipedia.org/wiki/Goodput
[4] T. Ito, H. Ohsaki and M. Imase, *On Parameter Tuning of Data Transfer Protocol GridFTP for Wide-Area Networks*
[5] http://www.perfsonar.net/
[6] http://www.maxmind.com/app/ip-location
[7] K. P. Gummadi, S. Saroiu, S. D. Gribble, *King: Estimating Latency between Arbitrary Internet End Hosts*, Unv of Washington, Seattle, WA USA, 98195-2350
[8] T. Ito, H. Ohsaki and M. Imase, *On Automatic Parameter Configuration Mechanism for Data Transfer Protocol GridFTP*
[9] A. A. Ayllon, A. Beche, F. Furano, M. Hellmich, O. Keeble and R. B. Da Rocha, *Web enabled data management with DPM & LFC,* CERN, Geneva 1211, CH