THE INTERNATIONAL JOURNAL OF SCIENCE & TECHNOLEDGE

Advance Two-Tier Protocols Attempt for Data Upload in Computer (ATTPDC)

Somendra Kumar
M. Tech. Student, Madhav Proudyogiki Mahavidyalaya, Bhopal, India
Dr. Gireesh Dixit
Professor and HOD, Department of Computer Science, Madhav Proudyogiki Mahavidyalaya, Bhopal, India

Abstract:
The several clients-to-one upload problem arises when a set of clients needs to upload data to a single server. Uploads correspond to an important class of applications, many of which are limited by a deadline. This includes, e.g., electronic submissions of income tax forms, online shopping for limited-time bargain products, a n d gathering data from sensor networks. Such services create hot spots, which is a major hurdle in achieving scale ability in network-based applications. Two-tier protocols attempt to relieve hot spots in many-to-one applications. In this model, clients upload their data to intermediaries (also called supporting servers s1 and s2), to reduce the traffic to the destination around a deadline. The destination server then computes a time interval for pulling the data from bistros after the deadline. Bistros have limited storage capacity, and each bistro j may fail with some probability 0 < pj < 1.Previous works on reliable data upload report on experimental results.

Keywords: *HITS, PageRank, WPR, General Web Pages, Web Structure Mining, Link Analysis, ISM, AISM.*

1. Introduction

1.1. Data Upload Models
In a centralized model, each user is instructed by the main server to upload her data to a subset of the bistros. The bistros will be queried for the data to the main server after the last. Since some bistros may fail, the algorithm proceeds in iterations (re-scheduling the deadline), and the users whose data was lost need to re-upload their data to the bistros in the next iteration, until all the files reach the server. In the distributed model, each user searches for a group of bistros to which he can upload his data. In this model, the set of bistros in which the user's data is stored may be revealed to the main server only transmission of the data from the bistros to the main server (after each deadline)

1.2. Related Work
The bistro framework was proposed by Bhattacharji et al. [2]. Later works present simulation studies of a centralized bistro model. In particular, Cheung et al. proposed in [6] a FEC-based protocol for fault-tolerant data upload using the bistro model.
The problem of one-to-many/1-n communication, where a single server transmits data to multiple/n receivers, has been studied extensively (see, e.g.,[8, 20, 1, 19]). This problem arises, e.g., in download of data and in multicast. Attiya and Shachnai [1] presented a randomized distributed algorithm system.

1.3. Summary of Results
In analyzing the space and communication complexities of the algorithm we assume that all files are of the same size (and require one unit of storage). Let p = max jpj be the maximum failure probability of any bistro. Since in practice the jpj values are typically unknown, we only assume that the max failure probability is in (0, 1⁄4].[1] The value *T stop* is the number of the iteration at which the algorithm stops (see Section
The parameters s and d are the number of copies of each file and the total number of pieces of each file under the FEC scheme, respectively.
.

1.4. Advantage
Advantage! which makes it suitable for the particular applications. In particular, our replication-based algorithms (see Section 2) are simple to implement/use.
In this category, Algorithm "alb" has the advantage of balancing the load generated due to communication to the main server.

2. Advance Replication Algorithms for the Reliable Data Upload in Computer System

2.1. About Bounded Number of Iterations

In the following we present the algorithm which uses the Bistro model to upload a large amount of data to a main server. The algorithm guarantees that (*i*) each file is uploaded to the server. (*ii*) reliability is achieved while maintaining a small number of (re)transmission rounds, and (*iii*) the expected amount of data uploaded directly to the server does not exceed the capacity of a particular bistro. Each one bistro can store *C* client files. There are *n* clients, and for each one file the system keeps $s \geq 2$ copies. The bistros are partitioned into n/c gr~~o~~ups, each consists of s bistros holding copies of C distinct files.
The server informs the clients to which bistros they should upload their own data. After the end or deadline the server collects the data from the bistros. If some data is lost the server will initiate this process again for the client whose data was lost how these Page Rank calculations are happening.

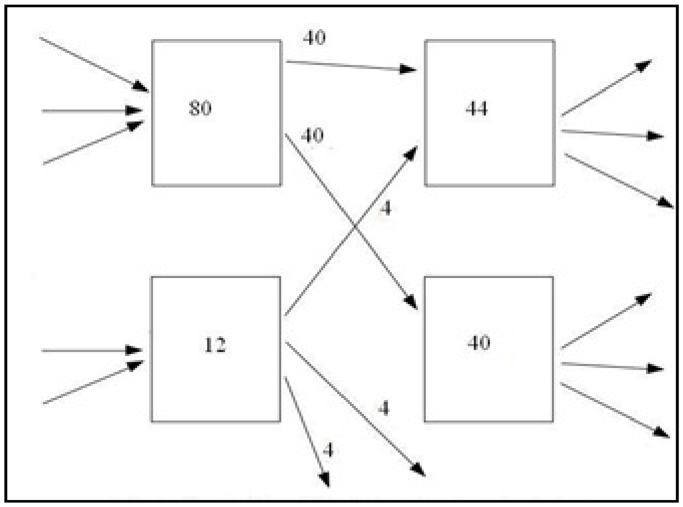

Figure 1: Simplified Page Rank Calculation

The link coming from an important page is given higher weightage. Similarly the backlinks coming from non-important pages will be given less weight age. We now analyze the algorithm. Clearly, the total number of iterations is at most *T-stop*. In the next result, we will bound the expected number of files uploaded directly to the server in this last iteration.
Proof. Denote by P m = P $m(ABI\,)$ the total number of messages sent by the algorithm, and by $P^t_m = P^t_m(ABI\,)$ the number of messages in iteration t, for $t \geq 1$. Let pt be the number of remaining clients in iteration t. We note that each client sends in each iteration a message to main server and to s bistros; the client then receives a message from server and from each of these bistros. At the end of iteration t the server collects the pt files from the bistros and sends acknowledge messages to each of the pt clients and $s\ p\ t/C$ bistros. Then the overall number of messages sent in iteration t is ${}_{P\,m}{}^{t} \cdot 2n_ts + 5n_t + {}^{2n}{}_{C}{}^{ts} \cdot 5p_t(s + 1);$
where s is the number of copies of particular file. We note that equation (2.1) holds for any $t = T\ stop > 1$. Hence, we get that the expected number of messages sent in iteration $t >= 1$ is $E[P^t{}_m] = 5p^{st}n(s + 1)$.
Let *I* want denote the number of iteration at which the algorithm terminates, then we have that Here t is the tth page in the result page-list Rl(p), **n** represents the first **n** pages chosen from the list El(p), and W_{ith} is the weight value of ith page.
Wt = (s1,s2,s3,s4,s5)
Here, s1, s2, s3,s4 and s5 are the values assigned to a page if the page is **VRl, Rl, WRl** and **IRl** respectively. It is also obvious that
s1>s2
S2>s3
s3>s4
Experiments conducted on these algorithms prove that we get higher relevancy values with WSR as compared to Page Rank.
This algorithm is also a total link analysis algorithm proposed by J. K. Berg. It classifies Computer pages into two categories called as hubs and authorities. Hubs are the pages that links to other important pages so they act Authorities are the pages containing The third inequality follows from Co_rollary 1.We now obtain a bound on message complexity in terms of the number of clients in the system. We need₃for that the next technical lemma.
$\underline{C}$
Lemma 4 *Let* $s = \log_p 1$ ¡ (1 ¡ ") $^{\underline{n}}$ *then* $s = O(\log n)$.
Proof. let $x = 1$ ¡ ", we show below that there exist n_0 ¸ 1 and a constant r, such that for all $n > n_0$
¡ ln(1 ¡ $x^{C=n}$) · r ln(n)
$_e$¡ ln(1¡$x^{C=n}$) . $_e r$ ln(n)
· $_n r$
$_{1\ ¡\ x}C=n$
1$\underline{C}$

$1 ¡ x^n$

$\_\_\_\_\_\_\_\_ \_\_{}_n r$

For C ¸ 2, let $k = \frac{1}{''}$ and choose $r = k^2$. We need to show that

$$1 ¡ _k \quad \cdot \quad {}^{\mu}1 ¡ _{nk2} \quad ¶ \quad \underline{n}$$

$$1 \qquad 1 \qquad 2$$

2.2. Using the Main Server as a Bistro
2.3. Algorithm *ABI* in Section 2.1 uses *T-stop* = *O*(log *n*) as an upper bound on the number of iterations. Thus, with some positive probability the number of files that
2.4. need to be uploaded directly from clients to the main server in the last iteration may be much larger than *C*. We now describe algorithm *ALB* which uses the bistro
2.5. system iteratively, as long as the number of remaining files is larger than *C*.
We show that (*i*) by using the main server as a bistro in each iteration, the number of iterations is bounded (and there is no need to define *T stop*), and (*ii*) the expected number of messages used by the algorithm is the same as calculated for *ABI* . As before, we use the assumption that the main server is reliable (i.e., its failure probability is equal to 0).
2. Informally, algorithm *ALB* proceeds in iterations. In each iteration, the users can upload their files to bistros. If a group of bistros has failed, the algorithm move to the next iteration. Since the main server is a bistro itself, at the end of each iteration at least *C* files were uploaded to the server; therefore, the number of iterations is bounded by $^{n/c}$. In the next result, we show that the expected number of iterations until all files are uploaded is a small constant, for typical values of *ε*. Since algorithm *ALB* does not use any bound on the number of iterations, this tightens the bound of *n/C* on a total number of iterations.
Therefore, we feel the requirement of a mechanism so that we get the relevant information according to the query posted by me. Informally, algorithm *ALB* proceeds in iterations. In each iteration, the users can upload their files to bistros. If a group of bistros has failed, the algorithm move to the next iteration. Since the main server is a bistro itself, at the end of each iteration at least *C* files were uploaded to the server; therefore, the number of iterations is bounded by $^{n/c}$. In the next result, we
7.1 ATTPDC ALGORITHM

- Step 1: Input Query For Upload Data
- Step 2: Generate interpretations of search query using get Interpretation() method and ATTPDC Database .
- Step 3: Post interpreted query to the search engine.
- Step 4: generate the search results.

7.2 ATTPDC Database

Original_Data	**Uploaded Data**
Mr. Narayan Murthy except Infosys Founder	Mr. Narayan Murthy –Infosys
CSS tutorial on website w3schools.com	CSS tutorial site:w3schools.com
sites similar to facebook.com	related: www.facebook.com
Only PDF tutorial on Database Management System	"Database Management System" tutorial file type: pdf
books on "C#".Net from 2002 to 2010	C#.Net books 2002..2010
useful links on DBMS	In anchor: DBMS
Url including the com	In url: company

Table 1: Interpret_Query

3. Advance FEC-Based Algorithm for Reliable Data Upload in Computer

3.1. Using Advance FEC vs. File Duplication

The Advance FEC (Forward Error Correction) algorithm creates *d* fragments from a file. To recover the file it success to obtain only $m < d$ fragments. In the duplication method, *m* is always equal to 1, therefore the number of copies stored of each file is *d*. In the Advance FEC scheme, the amount of storage units required for each file is *d/m*. Bistro failures are treated as follows. If at the most $d - m$ bistros fail in a given group, then the system can still retrieve the files uploaded to this group; however, if the number of failures in one group exceeds $d - m$, all of the client whose files were uploaded to this

3.2. Now the Algorithm

We give below algorithm "*AF*" that uses the Advance FEC scheme in the bistro model for reliable data upload. As algorithm *ABI* , algorithm *AF* proceeds in iterations, until it reaches iteration *T stop*, after which all the remaining files are uploaded directly from the clients to the main server. However, instead of duplicating files, "*AF*" uses Advance *F EC*, i.e., each file is partitioned to *d* pieces, out of which is least *m* pieces are needed for reconstructing the file.

The server's algorithm is given in Figure 5.4. The bistros and client's algorithm is given in Figure 5.5. We now analyze the algorithm.

TTDU		ATTPDC	
Original Query	No. of data upload in single Sided. (TTDU)	Interpreted query (Using ATTPDC)	No. of data uploaded in advance two tier (By ATTPDC)
Mr. Narayan Murthy except Infosys Founder	2	Mr. Narayan Murthy –Infosys	9
CSS tutorial on website w3schools.com	16	CSS tutorial site:w3schools.com	30
sites similar to facebook.com	12	related: www.facebook.com	2
Only PDF tutorial on Database Management System	19	"Database Management System" tutorial filetype:pdf	24
books on C#.Net from 2002 to 2010	15	C#.Net books 2002..2010	27
useful links on DBMS	16	inanchor:DBMS	18
Url including the word company	4	inurl: company	4
indian classical dance form except bharatnatyam	5	Indian classical dance form – bharatnatyam	21
Total	**89**		**135**

Table 2: Comparison between Two-Tier Data Upload (TTDU) and ATTPDC

4. Advance Distributed FEC-Based Algorithm for Computer System

To address this issue, we present below a distributed version of the "FEC"-based algorithm of Section 5. The Advance distributed algorithm for computer system starts with a preprocessing phase, in which the set of mid-level bistros is partitioned into groups of size *d*. The message complexity of this phase depends on the method of distributing the bistro code among the bistros, which is beyond the scope of our work. Thus, in analyzing the algorithm, we do not refer to the preprocessing step, or to the construction phase of the network. After the pre processing phase, each bistro has the set of "IDs" of the ($d - 1$) other bistros in its group. The files are transmitted to the main server in two/multi stages. In the first stage, the files are collected from the clients and uploaded to the bistro groups. Each client initially sends a request to allocate the storage space for the *d* pieces of its file. This requires finding the group of bistros which has available storage for the *d* file pieces stored at the client. Once the storage allocation is allowed by a group, the file pieces are uploaded to the bistros in this group. Each group sends a list of the participating bistros to the main server, either when all of its storage has been allocated or at the last end. Then the server schedules the file retrieval from the bistro groups. Once all of the files have received and processed by the main server, it sends acknowledgments to all of the bistros and clients.

If case of a failure of the main server, a new particular server can start the data collection process of the second phase without losing any data. (The re-placement of the cause server is done at system level; my algorithm accepts the id of the new server as an input). Since the particular server may fail at any time after the start of the second phase of the algorithm, the client files are kept at the bistros till the end of the algorithm. The pseudocode of the distributed algorithm, "*ADF* ", is given in Figures 5.6 and 5.8. We now analyze the message and space complexity of "*ADF"* Theorem 16 The expected message complexity of "ADF" *is Big* $O(n^2)$.

Proof. We note that since "*ADF"* is a distributed implementation of "*AF"*, the analysis is similar to the analysis of "*AF"*, except for a number of messages will display sent in each iteration. Let $P^t m = P^t m(ADF)$ be the number of messages sent in iteration *t* under *ADF* , and denote by *nt* the number of clients at the begin of iteration *t*, then at the begin of the iteration each of the clients needs to search for a group of bistros to store her file. Since the number of groups is ^{nt} , this search phase may require the *Cm* clients to send ___ *Big* $O(nt^2)$ messages. Sending the files to the bistros requires *ntd* messages. The bistros in each group then inform the server on the stored files; the server uploads *m* pieces of each file from each of the bistro groups and allows the bistros in each such group to release the space allocated for these files. Thus, we have, where *d* is the total number of pieces produced for each file. *I* Letdenote the number of iteration at which the algorithm terminates

5. The Summary and Open Problems for Computer System

Here we studied algorithms for reliable and scale m-1 data upload in Advance two-tier systems for Computer System such as Bistro. While previous work analyze algorithms through experiment study, we give theory performance guarantees for several (centralized and distributed) algorithms for reliable data upload. We also compare the performance of replication-based algorithms vs. algorithms that use Forward-Error-Correcting codes. We leave open several interesting avenues for future work: In our "FEC"-based model, the objective is to increase the probability that the algorithm terminates after a single iteration. Therefore, while *m* piece to retrieve a file at the main server, the clients, always upload $d > m$ pieces of their files. It would be interesting to explore the tradeo® between the (expected) run time of the algorithm and message complexity. For example, consider an algorithm which uploads at any iteration only some *missing pieces* of files from the clients to the bistros. To increase reliability, in the first iteration, the client uploads m >n pieces and leave some new pieces on a shelf to be used later for re transmissions. While the main server keeps the pieces it did manage for recover. We will focus mainly on Advance two-tier data upload. More generally, how would a two-tier upload model perform in terms of reliability? storage complexity? Consider the data upload problem from game-theoretic perspective. In particular, how should the system upload data to the main server when each client is an agent? What should be the performance measure in such type of system? Such example, particular agent can use different number of bistros; consequently, particular client may incur different costs.

- Consider a mixed system, where a client could also be a bistro, how will it perform? Will we reward clients who choose to be bistros? What is the tradeo® between performance and the cost of using bistros?

Consider the following problem, which arises in the centralized model.
Each user i, $1 \leq i \leq n$ initially contacts the server to obtain the addresses of the bistros to which i needs to send its data. To determine the destination bistros for each user, the following *data assignment* problem needs to be solved at the server. Each bistro j, $j \geq 1$, has limited storage capacity, cj ; each user i need to upload to a subset of bistros ai copies of its data for some $ai > 1$ (to assure a required level of reliability). We need to assign the data copies to the bistros subject to capacity constraints, example, that two or multiple copies of the same data are stored on *distinct* bistros, and the overall number of bistros used is minimized. This defines a special case of the *bin packing with the conflicts* problem [7]. It would be interesting to study also the online version of the problem, in which items arrive one by one, each having a size and a set of edges to be added to the corresponding vertex in the conflict graph; an arriving item needs to be packed immediately and irrevocably.

- Consider data upload problems arising in sensor networks, where the limitation on energy consumption and communication range should be taken into consideration.

```
Server2
C
set q = $d\log_p(1 ¡ (1 ¡ '')^n)e$
set T issingData = 1, $T_{now} = 1$
while $T_{now} < T_{stop}$ AND T issingData = 1 do
f Assign each bistro to one of $\frac{n}{C}$ groups,
a group consists of q bistros until the deadline do:

f receive client i request
send to the client i the list of bistros of group (i mod $\frac{n}{C}$) + 1 g
after the deadline set Missing Data = 0

for each group do:
f          set Got All Group = 0; ij= 1
while Got All Group = 0 AND $i \cdot sq$do f connect to the j-th bistro of the group
ask for the data of the clients for that group set j = j+ 1
if got all data for that group then send \all clear" to all the bistros of that group set GotAllGroup = 1

g if GotAllGroup = 0 then add the clients whose data is not received to a \NACK list".
set T issingData = 1g
if T issingData = 1 set new deadline
set n = size of \NACK list", $T_{now} = T_{now} + 1$ send NACK to all the clients on the "NACK list" g
if T is sing Data = 1 then do until the deadline OR until got all data

Receive clients request for a bistro list, and send a server as the only bistro Receive client's data and send
ACKNOWLEDGE
```

Figure 2: The Server Algorithm "ABI"

6. Conclusion

It is clear from the above experiment that ATTPDC produces better results in most of the cases and it fails only in few cases. This method can be implemented on the top of any existing experiment or any algorithm to produce more relevant results or advance result.

7. References

i. H. Attiya and H. Shachnai. Tight bounds for FEC-based reliable multicast. Inf. Comput. 190(2): 117{135, 2004.
ii. Bhattacharjee, W. C. Cheng, C.-F. Chou, L. Golubchik, and S.Khuller. Bistro: a framework for building scalable wide-area up load application. ACM SIGMETRICS Performance Evaluation Review, 28(2):29{35, 2000.
iii. Bistro – publications.
iv. W. C. Cheng, C.-F. Chou, L. Golubchik, and S. Khuller. A secure and scalable wide-area upload service. In Proc. 2nd International Conf. on Internet Computing, vol. 2, pp. 733{739, 2001
v. W. C. Cheng, C-F. Chou, L. Golubchik and S. Khuller. A performance study of Bistro, a scalable upload architecture. SIGMET-RICS Performance Evaluation Review 29(4): 31{39, 2002.

vi. L. Cheung, C-F. Chou, L. Golubchik, Y. Yang: A Fault Tolerance Protocol for Uploads: Design and Evaluation. ISPA 2004: 136145

vii. L. Epstein and A. Levin. On Bin Packing with Con°icts. In Proc. of WAOA, 160{173, 2006.

viii. T. Lestayo, M. Fernandez and C. Lopez. Adaptive approach for FEC reliable multicast. Electronic letters 37, 1333 (2001).

ix. W. C. Cheng, C-F. Chou, L. Golubchik, S. Khuller, Y-C. Wan. Large-scale Data Collection: a Coordinated Approach. IEEE IN-FOCOM, vol. 1, pp.218{228, 2003

x. M. Ma, Y. Yang. Data gathering in wireless sensor networks with mobile collectors. Proc. of IPDPS, 2008.

xi. F.J. Mac Williams and N. J. A. Sloane , the Theory of Error Correcting Codes, North-Holland, 1977.

xii. M. Mosko and J. J. Garcia-Luna-Aceves. An Analysis of Packet Loss Correlation in FEC-Enhanced Multicast Trees. In Proc. of ICNP, 2000.

xiii. R. Mus¸aloiu-Elefteri, R. Mus¸aloiu-Elefteri, A. Terzis: Gateway Design for Data Gathering Sensor Networks. SECON 2008: 296-304.

xiv. N. Nikaein, H. Labiod and C. Bonnet. MA/FEC: A QoS-Based Adaptive FEC for Multicast Communication in Wireless Net-works. Proc. of ICC, 2, pp. 954{958, 2000.

xv. U. Shani, A. Sela,A. Akilov, I. Skarbovski and D. Berk. A scalable heterogeneous solution for massive data collection and database loading. Proc. of BIRTE, 2006.

xvi. Y. Yang, L. Cheung, L. Golubchik. Data assignment in fault tolerant uploads for digital government applications: a genetic algorithms approach. DG.O, pp. 29{38, 2005.

xvii. H. Samet, L. Golubchik. Scalable data collection and retrieval infrastructure for digital government applications. DG.O, pp. 301{ 302, 2006.

xviii. Future Plans for the Deep Space Network.

xix. A. Popescu, D. Constantinescu, D. Erman and D. Ilie, A survey of Reliable multicast communication. Proc. of 3rd EURO-NGI Conf. on Next Generation Internet Networks, 2007.

xx. P. Paul and S. V. Raghavan, Survey of multicast routing algorithms and protocols. Proc. International Conf. on Computer Communication, Vol. 15(3) pp. 902-926, 2002.