Scalable Computing: Practice and Experience ISSN 1895-1767
Volume 10, Number 1, pp. 75–86. http://www.scpe.org

MINIMIZATION OF DOWNLOAD TIME VARIANCE IN A DISTRIBUTED VOD SYSTEM*

ANNE-ELISABETH BAERT†, VINCENT BOUDET†, ALAIN JEAN-MARIE†, AND XAVIER ROCHE†

Abstract. In this paper, we examine the problem of minimizing the variance of the download time in a particular Video on Demand System. This VOD system is based on a Grid Delivery Network which is a hybrid architecture based on P2P and Grid Computing concepts. In this system, videos are divided into blocks and replicated on hosts to decrease the average response time. The purpose of the paper is to study the impact of the block allocation scheme on the variance of the download time. We formulate this as an optimization problem, and show that this problem can be reduced to finding a Steiner System. We analyze different heuristics to solve it in practice, and validate through simulation that a random allocation is quasi-optimal.

Key words: performance evaluation, video on demand, approximation algorithms, constraint optimization problem, simulation.

1. Introduction and Problems. This paper is about the replication of data in a particular Grid Delivery Network (GDN). A GDN is a distributed data delivery system that is able to provide video services, among which Video On Demand (VOD). The idea of VOD is to allow users to request video documents at any time, without a preestablished time schedule. One of the main challenge of GDN system resides therefore in ensuring that users can download video at any time with guaranteed, pre-established download time.

The recent literature has pointed out the interest of using distributed systems for massive video distribution [9, 12, 13]. Documents replication and repartition can also solve some performance problems for video distribution, and is an effective way to improve the performance of video download time. Those problems have been studied extensively in the literature [2, 13, 10]. Ibaraki *et al.* employed a replication optimization technique borrowed from the theory of resource allocation problems [5]. Chou *et al.* gave a comparison of scalability and reliability characteristics of servers by the use of stripping and replication [2].

Venkatasubramanian *et. al.* proposed a family of heuristic algorithms for dynamic load balancing in a distributed VOD server [10]. Zhou *et al.* formulate the problem of video replication and placement on a distributed storage cluster as a combinatorial optimization problem [13].

For a popular document, increasing its replication may decrease its download time. However, as the total storage of the system is usually fixed, increasing its replication implies to decrease the replication of other documents. Video replication and placement algorithms based on popularity were therefore developed in [4, 9, 12]. In these algorithms, the most popular videos are all duplicated on the servers, the problem is then to find how much the less popular documents should be replicated. The response waiting time problem is then restricted to certain videos. In [8], the authors analyzed the impact of the division of the documents into blocks on availability and average download time in a general, BitTorrent-like P2P file systems.

As this brief review of the literature shows, there already exist many replication and placement algorithms aimed at optimizing the average download time for videos. However: on the one hand, none of these models applies directly to the GDN architecture, and on the other hand, no attention has been given yet to optimizing the *variance* (or the standard deviation) of download times. Yet, the system can foresee statistically and *guarantee* the download time only if this variance is small. This is the question we address in the present paper. We have shown in a previous work [1] that with an appropriate replication factor, the average download response time is independent of the allocation methods. On the other hand, we showed that the allocation methods have a direct impact on the variance of response waiting time.

The remainder of this paper is structured as follows. In Section 2, we briefly describe the architecture of the system, we also summarize some of the most important differences between GDN and traditional P2P approaches for VOD. In Section 3, we formulate the problem of minimizing the variance of the waiting time as a combinatorial optimization problem called MINVAR. We study this objective function and show the difficulty of this problem. We compute a lower bound on the optimal value for this objective function. In Section 4 we analyze some heuristics to solve the MINVAR problem. We evaluate them in Section 5 our algorithms using extensive simulations representing different situations. Section 6 concludes the paper.

*This research is supported the French National Agency for the Research, Audiovisual and Multimedia program, project VOODDO (contract 2007 AM 012-02).

†LIRMM, CNRS, Université de Montpellier 2, 161 rue ADA, 34392 MONTPELLIER Cedex 5, FRANCE. A. Jean-Marie is with INRIA Méditerannée. ({baert, boudet, ajm, roche}@lirmm.fr)

2. Grid Delivery Network description. The system under study is a hybrid architecture based on ideas from P2P Networking and "Grid Computing" called the *Grid Delivery Network* (GDN). This concept uses the principle of Grid Computing to keep the control of this centralized network. However, to optimize the data transmission and to propose a document delivery network within guaranteed time, this network uses the transport capacity of Peer to Peer networks, through its capacity of multi-source data delivery.

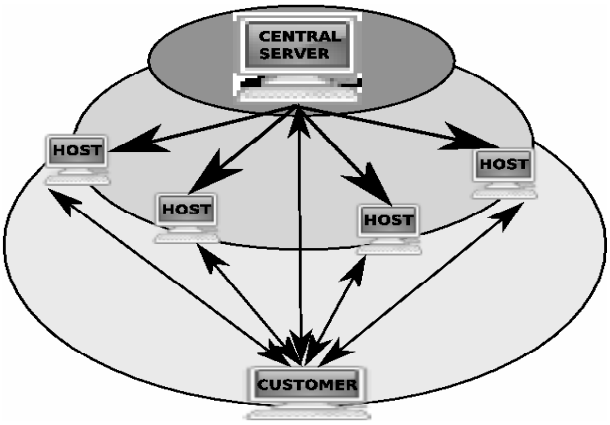

FIG. 2.1. *Grid Delivery network's architecture*

The particular GDN system analyzed in this paper is an implementation described in [11] (see also [1]) and represented in Figure 2.1. It is composed of a central server which manages the whole network and hosts which are "partners" of the GDN and deliver the documents to customers. In this GDN, each host is assumed to be a personal computer of the Grid Computing network, which is bound by contract to provide some resources to the system. Nevertheless, hosts are not available all the time to the network; the proportion of time each host is online is denoted with δ.

Documents are divided into pieces called "blocks", which are distributed *and* replicated over the partners, in order to improve reliability and decrease response time see Figure 2.2.

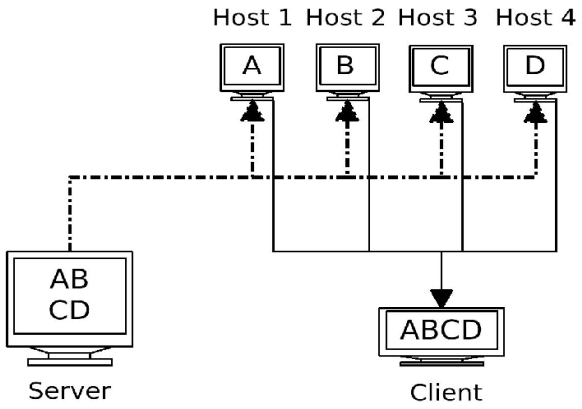

FIG. 2.2. *Document Delivery in the GDN*

The operation of the system is as follows. When some client wants to download a document on the GDN, it first contacts the central server. The server sends a *download plan* which specifies, for each block of the document, the identity of some host on which the block is stored. The client then executes this plan, by downloading the blocks in parallel from the partners which are online. Then, all missing blocks are downloaded directly from the central server (or some backup data storage).

In [1], we have proposed a probabilistic model of this GDN, and we have shown that, under appropriate assumptions, the download time for a whole document is given by

$$R \;=\; a \;-\; bN \;, \tag{2.1}$$

where N is the number of blocks of the document that happen to be available on the network, and a, b are positive constants.

By replicating a block on several hosts, the GDN increases the probability that some available host owns this block, and therefore that the download time will be smaller. However, as the total hosts capacity of the GDN is limited it is not possible to replicate every document on every host. We have studied in [1] the problem of minimizing the *average* response time, by picking the replication factor of each document appropriately, in function of their popularity, and under memory constraints. We have proved that the *average* download time is independent of the particular way the replicated blocks are assigned to distinct partners. On the other hand, it turns out that this assignment has a very strong impact of the *variance* of the download time. Since the Quality of Service objective of the GDN is to guarantee the download time to customers, it is important to find assignments which minimize this variance.

3. Response waiting time optimization.

3.1. Variance of the response waiting time. We begin with computing the variance of the download time of a particular document. According to Eq. (2.1), the variance of R is proportional to the variance of N, on which we concentrate in the remainder.

Let us consider a particular document D, segmented into T blocks. Let $B_1, \cdots, B_T$ be these blocks. Assume that the optimal replication factor L has been computed so as to optimize the average download time [1]. Define $U_j = 1$ if the block B_j is available on some online partner, 0 otherwise. Assuming that the hosts' on-time is uniformly distributed over time, and that hosts are independent, we have:

$$\begin{aligned}\mathbb{P}(U_j = 1) &= 1 - \prod_{i=1}^{L} (1-\delta) \\ &= 1 - (1-\delta)^L \ .\end{aligned}$$

Let N be the number of available blocks of the document D. We have $N = \sum_{j=1}^{T} U_j$. The variance of N is given by:

$$\mathrm{Var}(N) = \sum_{j=1}^{T} \mathrm{Var}(U_j) + \sum_{k \neq j} \mathrm{Cov}(U_k, U_j)\,. \tag{3.1}$$

We denote by $L(j)$ the list of hosts which own the block B_j. It is known that, using the notation $\bar{\delta} = 1 - \delta$,

$$\begin{aligned}\mathrm{Cov}(U_k, U_j) &= \mathbb{E}(U_k U_j) - \mathbb{E}(U_k)\mathbb{E}(U_j) \\ &= \bar{\delta}^{|L(k)\cup L(j)|} - \bar{\delta}^{|L(k)|+|L(j)|}\,.\end{aligned} \tag{3.2}$$

The variance of U_j is given by :

$$\begin{aligned}\mathrm{Var}(U_j) &= \mathbb{E}({U_j}^2) - \mathbb{E}(U_j)^2 \\ &= \bar{\delta}^{|L(j)|} - \bar{\delta}^{2|L(j)|}\,.\end{aligned} \tag{3.3}$$

We have :

$$|L(k) \cup L(j)| = |L(j)| + |L(k)| - |L(k) \cap L(j)|\,.$$

We use the assumption that the replication factor is the same for each block i. e. $|L(j)| = L\,, \forall j$, we obtain for Eq. (3.1):

$$\mathrm{Var}(N) = T\gamma^{-L}(1-\gamma^{-L}) + \gamma^{-2L} \sum_{k \neq j} \left(\gamma^{|L(k)\cap L(j)|} - 1\right)$$

where $\gamma = (1-\delta)^{-1}$.

Accordingly, minimizing Eq. (3.1) is equivalent to minimizing the following objective function

$$\sum_{k \neq j} \gamma^{|L(k)\cap L(j)|}\,.$$

3.2. The MINVAR Problem. In this part, we define the MINVAR problem in the language of combinatorial optimization.

DEFINITION 3.1. *Given a bipartite graph $\mathcal{G} = (\mathcal{T}, \mathcal{P}, \mathcal{E})$, where $\mathcal{T}$, $\mathcal{P}$ are a set of vertices and $\mathcal{E}$ the set of edges, we define the interference function as*

$$J(\mathcal{G}, \gamma) = \sum_{b \neq b' \in \mathcal{T}} \gamma^{|L(b) \cap L(b')|}, \tag{3.4}$$

where $L(b)$ denotes the neighborhood of b. Obviously, the problem does not depend on the nature of the sets $\mathcal{T}$ and $\mathcal{P}$ but only on their size.

Our problem is the following:

DEFINITION 3.2. *The $MINVAR(T, P, L, \gamma)$ optimization problem consists in finding one bipartite graph $\mathcal{G} = (\mathcal{T}, \mathcal{P}, \mathcal{E})$ with $|\mathcal{T}| = T$ and $|\mathcal{P}| = P$, which minimizes the interference function $J(\mathcal{G}, \gamma)$ under constraints*

$$\forall b \in \mathcal{T}\ , |L(b)| = L\ . \tag{3.5}$$

3.3. Complexity of MINVAR. Now, we turn on the complexity of the MINVAR problem.

DEFINITION 3.3 ([3, 7]). *A Steiner system $S(B, L, P)$ is a collection of L-subsets (also called blocks) of a P-set such that each B-tuple of elements of this P-set is contained in a unique block.*

We have the following:

THEOREM 3.4. *There is a Steiner system $S(B, L, P)$ iff there is a solution $\mathcal{G}$ of the $MINVAR(T, P, L, \gamma)$ problem with*

$$T = \frac{P!(L-B)!}{L!(P-B)!} \text{ and } \gamma = T(T-1) + 1 \tag{3.6}$$

that satisfies $J(\mathcal{G}, \gamma) < \gamma^B$.

Proof.

If there is a collection C solution of the Steiner system $S(B, L, P)$, then we can assign to each vertex $b \in T$ the neighborhood that matches one of the subsets of C. There is exactly

$$\frac{P!(L-B)!}{L!(P-B)!}$$

subsets in C. So each pair $(b, b') \in T^2$, $b \neq b'$ shares at most $(B-1)$ neighbors. We have created a solution $\mathcal{G}$ such as

$$\begin{aligned} J(\mathcal{G}, \gamma) &\leq T(T-1)\gamma^{B-1} \\ &< \gamma^B\ . \end{aligned}$$

On the contrary, if there is a solution $\mathcal{G}$ of the MINVAR problem, i. e.,

$$T = \frac{P!(L-B)!}{L!(P-B)!}, \text{ and } \gamma = T(T-1) + 1 \tag{3.7}$$

with $J(\mathcal{G}) < \gamma^b$ then we define a solution of $S(B, L, P)$ such as the collection $C = \{L(b) | b \in T\}$. As $J(\mathcal{G}) < \gamma^B$, each subset of P of size L appears at most once in C. Furthermore, we have

$$\frac{P!(L-B)!}{L!(P-B)!}\binom{L}{B} = \binom{P}{B}$$

then each subset of P of size L appears; we have a Steiner system $S(B, L, P)$. □

REMARK 1. *In fact, finding the optimal solution of $MINVAR(T, P, L, \gamma)$ with*

$$T = \frac{P!(L-B)!}{L!(P-B)!} \text{ and } \gamma > T(T-1)$$

is equivalent to know if there exists a Steiner system $S(B, L, P)$.

It is well-known that Steiner systems are very difficult to find. Therefore Problem $MINVAR(T,P,L,\gamma)$ is a complex problem in the general case with no obvious efficient algorithmic solution.

We now provide a lower bound for the interference function $J(\mathcal{G},\gamma)$, which will be useful later to validate solution heuristics.

THEOREM 3.5. *For every bipartite graph $\mathcal{G}=(\mathcal{T},\mathcal{P},\mathcal{E})$ with $|\mathcal{T}|=T$ and $|\mathcal{P}|=P$ and such that $|L(b)|=L$ for all $b\in\mathcal{T}$, we have:*

$$J(\mathcal{G},\gamma) \;\geq\; T(T-1)\;\gamma^{\frac{T^2L^2/P-TL+q(P-q)/P}{T(T-1)}}\;, \tag{3.8}$$

where $q=(TL) \mod P$.

Proof. We have, using the convexity of the power function $f(x)=\gamma^x$:

$$\begin{aligned}\frac{J(\mathcal{G},\gamma)}{T(T-1)} &= \sum_{b\neq b'}\frac{1}{T(T-1)}\gamma^{(|L(b)\cap L(b')|)}\\ &\geq \gamma^{\left(\sum_{b\neq b'}\frac{1}{T(T-1)}|L(b)\cap L(b')|\right)}\;.\end{aligned}$$

Using the notation $\mathbf{1}_{b,p}=1$ if $p\in L(b)$ and 0 otherwise,

$$\begin{aligned}\sum_{b\neq b'\in\mathcal{T}}|L(b)\cap L(b')| &= \sum_{b\neq b'\in\mathcal{T}}\sum_{p\in\mathcal{P}}\mathbf{1}_{b,p}\mathbf{1}_{b',p}\\ &= \sum_{p\in\mathcal{P}}\sum_{b\neq b'\in\mathcal{T}}\mathbf{1}_{b,p}\mathbf{1}_{b',p}\\ &= \sum_{p\in\mathcal{P}}\delta(p)\times(\delta(p)-1)\\ &= \sum_{p\in\mathcal{P}}\delta(p)^2 - T\times L\;.\end{aligned}$$

The square function is convex and the minimum of

$$\sum_{p\in\mathcal{P}}\delta(p)^2\,,\ \text{under constraint}\ \sum_{p\in\mathcal{P}}\delta(p)=T\times L\,,$$

is obtained for values of $\delta(p)=\lfloor TL/P\rfloor$ or $\delta(p)=\lceil TL/P\rceil$.

We decompose $TL=Pr+q$ with $1\leq q<P$, and any vector $(\delta(p))_p$ which components are q times the value $r+1$ and $P-q$ times the value r minimizes the function while obeying the constraint. Evaluating the function for such vectors gives the lower bound. □

4. Heuristics. In this part, we analyse heuristics to solve $MINVAR(T,P,L,\gamma)$ problem. We propose differents algorithms: a greedy algorithm, two algorithms based on the round-robin paradigm, and two based on random choices.

Algorithm 1: Greedy

```
Data: T, P, L and γ
Result: A bipartite graph G
begin
    G = ∅
    for k ∈ {1 ··· L} do
        for i ∈ {0 ··· T − 1} do
            j = arg min_j {J(G ∪ (i, j), γ) + J((G ∪ (i, j))^c), γ)}
            G = G ∪ (i, j)
end
```

4.1. Greedy Algorithm. This algorithm begins with an empty graph and, at each iteration, finds the edge which minimizes $J(\mathcal{G},\gamma)+J(\mathcal{G}^c,\gamma)$, where $\mathcal{G}^c$ is the complementary bipartite graph of $\mathcal{G}$. It turns out that this algorithm gives better results than a simple greedy algorithm that would minimize $J(\mathcal{G},\gamma)$ at each step. This is illustrated by the example (where $L=2$ and $\gamma=3$) given by Figure 4.1 .

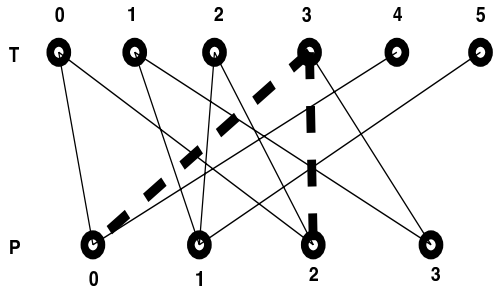

FIG. 4.1. *A greedy example for* $L=2$ *and* $\gamma=3$

Starting from the graph represented with continuous edges, choosing any of the dotted edges gives the same result for $J(\mathcal{G},\gamma)$, but only the edge $(3,2)$ minimizes $J(\mathcal{G},\gamma)+J(\mathcal{G}^c,\gamma)$. It turns out that this is the only choice towards the optimal solution. Minimizing $J(\mathcal{G},\gamma)+J(\mathcal{G}^c,\gamma)$ therefore guides the choice of edges towards better solutions.

4.2. Round-Robin Algorithm. Round-robin algorithms are generally based on a cyclical scan of vertices or edges. We propose two variants.

Algorithm 2: Blocks Round-Robin (BRR)

Data: T, P and L
Result: A bipartite graph $\mathcal{G}$
begin
  $\mathcal{G}=\emptyset$; $j=0$
  for $i\in\{0\cdots T-1\}$ **do**
    for $k\in\{1\cdots L\}$ **do**
      $\mathcal{G}=\mathcal{G}\cup(i,j)$;$j=j+1 \bmod P$
end

In this Blocks Round-Robin (BRR) algorithm, we give as neighborhood to each vertex of $\mathcal{T}$, the L "next" vertices of $\mathcal{P}$.

Algorithm 3: Mixed Round-Robin (MRR)

Data: T, P and L
Result: A bipartite graph $\mathcal{G}$
begin
  $\mathcal{G}=\emptyset$
  $j=0$
  for $k\in\{1\cdots L\}$ **do**
    for $i\in\{0\cdots T-1\}$ **do**
      while $(i,j)\in\mathcal{G}$ **do**
        $j=j+1 \bmod P$
      $\mathcal{G}=\mathcal{G}\cup(i,j)$
      $j=j+1 \bmod P$
end

In the Mixed Round-Robin (MRR) algorithm, we scan simultaneously the vertices of $\mathcal{T}$ and $\mathcal{P}$, and select the corresponding edge unless it is already in the graph; in that case the corresponding partner is skipped. After L turns, each vertex of $\mathcal{T}$ has a complete neighborhood.

4.3. Random Algorithms. The RandomSubset algorithm computes for a vertex b its neighborhood of size L among the P possible vertices. It draws uniformly at random one neighborhood among the $\binom{P}{L}$ possibilities.

Algorithm 4: RandomSubset

```
Data: b, P and L
Result: L vertices from a vertex b
begin
    S = ∅
    while |S| < L do
        j = random(1 ··· P)
        S = S ∪ (b, j)
end
```

The Random Algorithm simply performs such a choice, independently, for each $b \in \mathcal{T}$.

Algorithm 5: Random

```
Data: T, P and L
Result: A bipartite graph G
begin
    G = ∅
    for b ∈ {0 ··· T − 1} do
        S=RandomSubset(b, P, L)
        G = G ∪ S
end
```

We show now that it is possible to compute the average value of the objective function for such randomly generated solutions. The key result for this is the distribution of the number of common partners in two neighborhoods.

THEOREM 4.1. *Let $X_{bb'} = |L(b) \cap L(b')|$ be the number partners common to both neighborhoods. For every $b \neq b'$, the distribution of $X_{bb'}$ is given by:*

$$\mathbb{P}(X_{bb'} = k) = \frac{\binom{L}{k}\binom{P-L}{L-k}}{\binom{P}{L}} \quad \textit{and}$$
$$\mathbb{E}(X_{bb'}) = \frac{L^2}{P}\,.$$

Proof. If k elements of $\mathcal{P}$ are common between $L(b)$ and $L(b')$, the set $L(b')$ is formed by adding $L-k$ elements chosen among the $L-P$ elements that are not in $L(b')$ (see the interpretation of the Chu-Vandermonde convolution in e.g. [6, p. 56]). Hence the formula for $\mathbb{P}(X_{bb'} = k)$. As a consequence,

$$\begin{aligned}\mathbb{E}(X_{bb'}) &= \sum_{k=0}^{L} k\mathbb{P}(X_{bb'} = k) = \sum_{k=0}^{L}(L-k)\frac{\binom{L}{k}\binom{P-L}{k}}{\binom{P}{L}} \\ &= L\sum_{k=0}^{L}\frac{\binom{L}{k}\binom{P-L}{k}}{\binom{P}{L}} - \sum_{k=0}^{L} k\frac{\binom{L}{k}\binom{P-L}{k}}{\binom{P}{L}} \\ &= L - (P-L)\,\frac{\binom{P-1}{L-1}}{\binom{P}{L}} = \frac{L^2}{P}\,. \qquad \square\end{aligned}$$

REMARK 2. *In practical cases, Theorem 4.1 gives us the average number of the blocks in common for a given number of partners in the GDN. For example, if the number of partners is* 100 *and if* $L < 10$*, we can conclude that this number of blocks in common is less than 1 in the GDN.*

Corollary 4.2 gives the average value of the objective function for this algorithm.

COROLLARY 4.2. *If* $\mathcal{G}$ *is the graph generated by Algorithm 5, then the average interference function is*

$$\mathbb{E}(J(\mathcal{G},\gamma)) = \frac{T(T-1)}{\binom{P}{L}} \sum_{k=0}^{L} \gamma^k \binom{L}{k} \binom{P-L}{L-k} .$$

If we are convinced that it is possible to find a solution where $|L(b) \cap L(b')| \leq K$ for each $b, b' \in \mathcal{T}$, then we may improve the random algorithm : we refuse to accept a neighborhood that has more than K vertices in common with the previous neighborhoods. After Nb_Max unsuccessful tries, we increase K to accept neighborhoods more easily.

For example, if we consider 100 partners and the replication $L = 30$, we obtain by theorem 4.1 an average intersection of 9. We may want to consider the following constraint: forbid an intersection number greater than $K = 5$. We come back to the effectiveness of this idea in Section 5.

Thus we obtain the variant fully described in the following Algorithm named Random2.

Algorithm 6: Random2

```
Data: T, P, L, Nb_Max and K
Result: A bipartite graph G
begin
    G = ∅
    for b ∈ {0 ··· T − 1} do
        OK=false
        nb=0
        while !OK do
            S=RandomSubset(b, P, L)
            OK=true
            for b' ∈ {0 ··· b − 1} do
                if |L(b') ∩ S| > K then
                    OK=false
                    nb=nb+1
                if nb > Nb_max then
                    nb=0
                    K=K+1
        G = G ∪ S
end
```

5. Experimental Evaluation. We proceed now to determine how well these algorithms perform. Since the optimal solution to problem MINVAR is not known, we compare the outcome of the algorithms with an ideal situation given by the theoretical lower bound given in Theorem 3.5. In this section, we describe the experimental settings and then discuss the results.

5.1. General settings. We have built a specific simulator for the heuristics of Section 4. The program implements the three deterministic algorithms, and for the random algorithms, it allows to generate any specified number of independently drawn samples. We have carried out extensive experiments to evaluate the respective performance of the algorithms. In all simulations, we assume $\gamma = 2$, which corresponds to an availability of $\delta = 50\%$ for hosts. The size of documents ranges from $T = 100$ to 1000 blocks. The number P of hosts is 10 or 100, see the figure's captions.

5.2. Comparison. Figures 5.1, 5.2 and 5.3 display the ratio of the objective function to the lower bound of Proposition 3.5, for various sets of parameters. We observe on those figures that the two round robin algorithms have a relatively bad performance, which becomes very bad for large values of L.

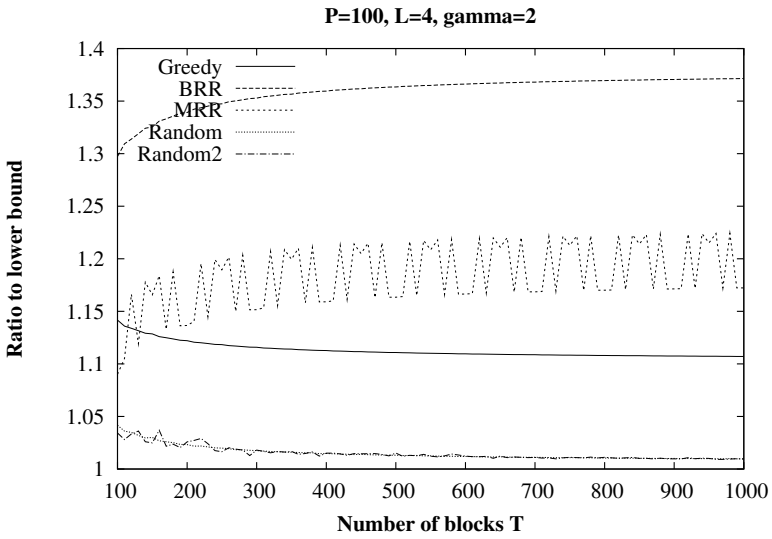

FIG. 5.1. *Algorithms Performance for* P=100, L=4

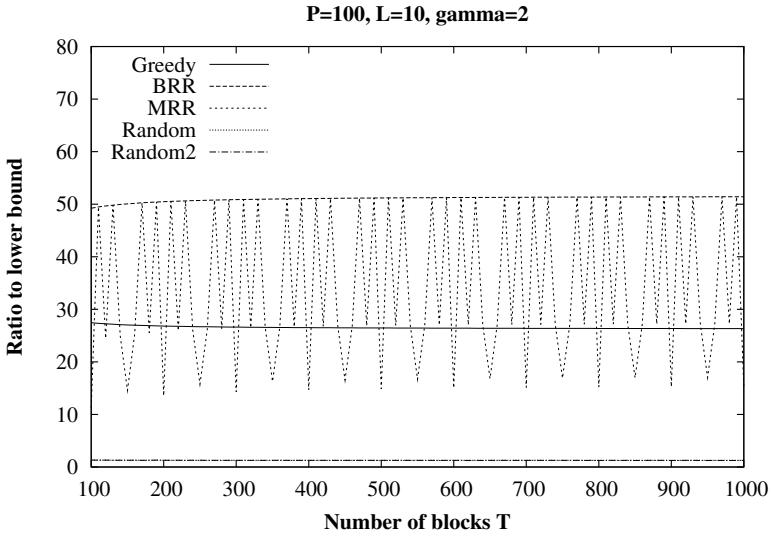

FIG. 5.2. *Algorithms Performance for* P=100, L=10

In real data distribution schemes, these two algorithms are often used because of their simplicity. It is interesting to see that for the present optimization problem, their performance is very far from the lower bound. In addition, the result of MRR appears to suffer sometimes from large fluctuations of arithmetic nature.

Another information is that the performance of the greedy algorithm is very dependent on the data. For some values of T, its results are better but for some other worst, than the random algorithms. Generally, the algorithm seems to perform well for small L and P parameters. When P gets large, the algorithm performs relatively well if L is small (ex. $L = 4$), but degrades if L is larger (ex. $L = 10$).

The figures also show that the greedy algorithm and the two random algorithms are very close to the lower bound when L is not too large.

We investigate this issue more closely using Figure 5.4, which is a zoom on Figure 5.3. This figure, as well as Figure 5.1, top, suggest that all three algorithms have an asymptotically robust performance when T is large, in the sense that their performance ratio remains bounded. However, the greedy algorithm turns out to be numerically unstable. Evaluating the J function is very hard for large values of T, L and P. Another

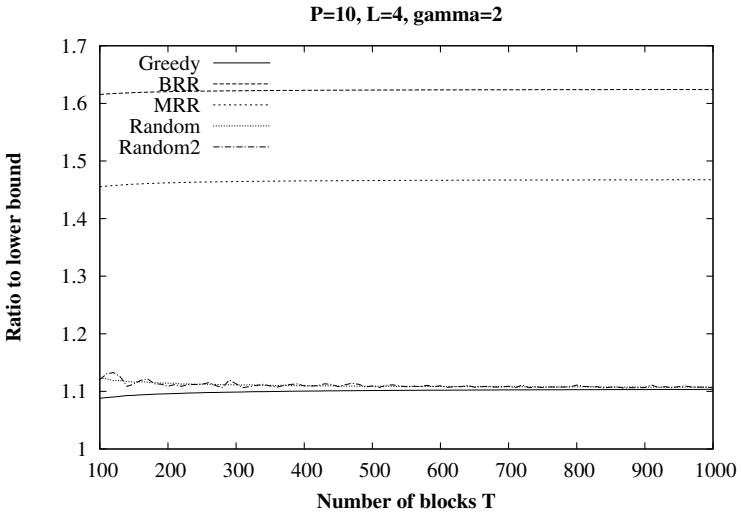

FIG. 5.3. *Algorithms Performance for* P=10, L=4

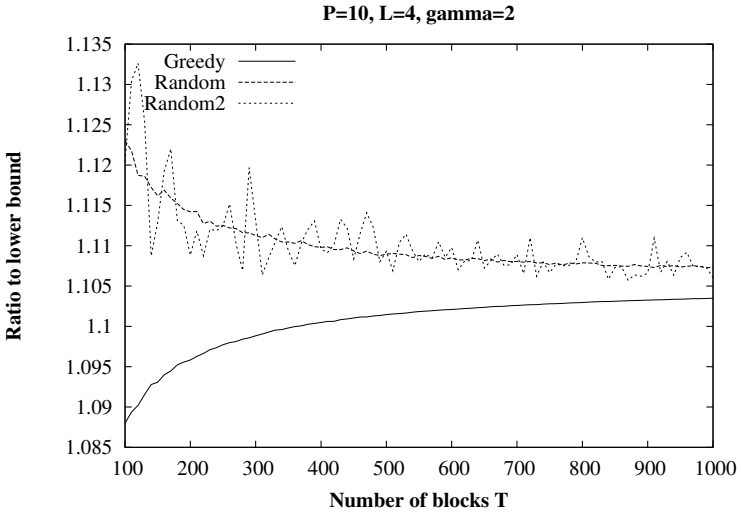

FIG. 5.4. *Best algorithms comparison*

disadvantage for the greedy algorithm is its algorithmic complexity: it is $O(PLT^3)$ whereas the complexity of the random algorithms is $O(PLT)$.

It appears therefore that the random algorithms are the best choice to determine a good block allocation.

5.3. Analysis of the random algorithm. Focusing now on the random algorithms, we compare in Figure 5.5 the average of the 1000 experiments with the *best* allocation found among them. For the Random2 algorithm, we take $Nb\_Max = 1000$.

We observe that the ratio is very close to 1; the standard deviation is actually very small. We conclude that in practice, the first allocation obtained by a random algorithm is probably already close to the optimal solution.

Finally, in Figure 5.6, we compare the Random and Random2 algorithms, through the ratio of their average and best performances.

In general, these two algorithms give similar results, but in some particular cases, there is a real improvement for Random2. In this figure, we have $P = 100$, $L = 30$ and $\gamma = 3$: the average interference number for each couple of blocks is 9 (see corollary 4.2). Still, rejecting neighborhoods with more than 5 vertices in common with

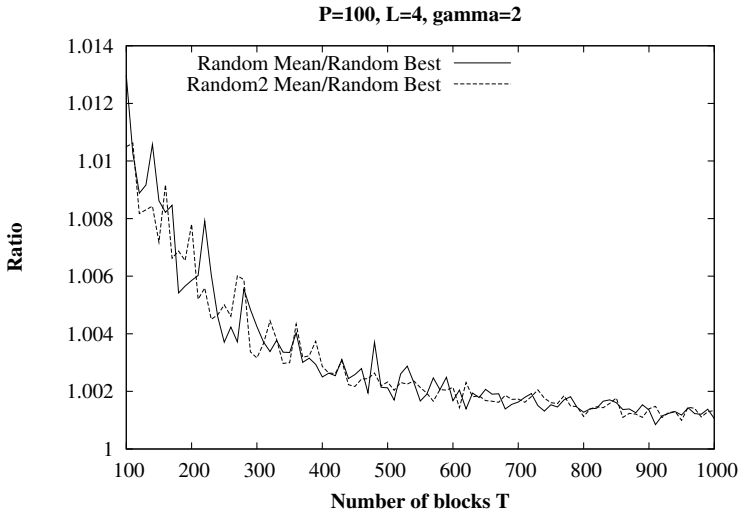

FIG. 5.5. *Ratio average to best*

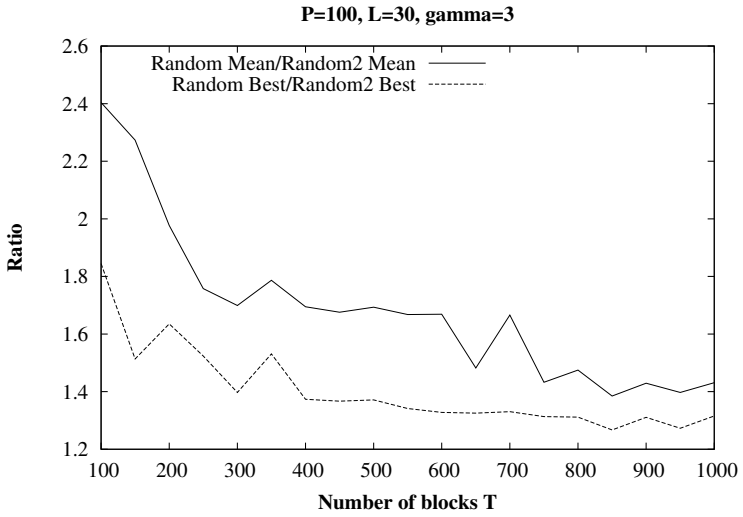

FIG. 5.6. *Random vs Random2 algorithm*

the previous neighborhoods turns out to be possible and this improves the block allocation found by $20-40\%$ for a large range of document sizes. For a smaller range of documents size (between 100 and 400), we can say that we obtain an average improvement of 80%.

6. Conclusion and future work. In this paper, we have proposed, and analyzed by simulation, various algorithms for optimizing the variance of download times in a particular distributed architecture, by placing appropriately replications on distributed data servers. In particular, we have shown that conventional Round Robin algorithms do not perform well in general, and that random algorithms are the best practical choice. Our methods have been applied to a real GDN deployment, and we are currently investigating their effectiveness in practice.

REFERENCES

[1] A.-E. BAERT, V. BOUDET, AND A. JEAN-MARIE, *Performance analysis of data replication in grid delivery networks*, in Int. Conf. on Complex, Intelligent and Software Intensive Systems, 2008, pp. 369–374.

[2] C.-F. Chou, L. Golubchik, and J. C. S. Lui, *Striping doesn't scale: How to achieve scalability for continuous media servers with replication*, in Int. Conf. on Distributed Computing Systems, 2000, pp. 64–71.
[3] C. Colbourn and J. Dinitz, *The CRC handbook of combinatorial designs, 2nd edition*, 2006.
[4] S. Gonzalez, S. Navarro, A. Lopez, and E. L. J. Zapata, *A case study of load sharing based on popularity in distributed VoD systems*, in IEEE transactions on Multimedia, vol. 8, 2006, pp. 1299–1304.
[5] T. Ibaraki and N. Katoh, *Resource allocation problems: algorithmic approaches*, MIT Press, Cambridge, MA, USA, 1988.
[6] D. Knuth, *The art of computer programming Vol. 1*, Addison-Wesley, 1973.
[7] F. MacWilliams and N. Sloane, *The theory of error-correcting codes*, Horth Holland, 1977.
[8] R. Susitaival and S. Aalto, *Analyzing the file availability and download time in a P2P file sharing system*, in Int. Conf on Next Generation Internet Networks, 2007, pp. 88–95.
[9] K. Tanaka, H. Sakamoto, H. Suzuki, and K. Nishimura, *Performance improvements of large-scale video servers by video segment allocation*, Syst. Comput. Japan, 35 (2004), pp. 27–35.
[10] N. Venkatasubramanian and S. Ramanathan, *Load management in distributed video servers*, in Int. Conf. on Distributed Computing Systems, 1997.
[11] VodDNET Company, `http://www.voddnet.com`.
[12] J. Z. Wang and R. K. Guha, *Data allocation algorithms for distributed video servers*, in ACM int. conf. on Multimedia, New York, NY, USA, 2000, ACM Press, pp. 456–458.
[13] X. Zhou and C.-Z. Xu, *Efficient algorithms of video replication and placement on a cluster of streaming servers*, J. Netw. Comput. Appl., 30 (2007), pp. 515–540.

Edited by: Fatos Xhafa, Leonard Barolli
Received: September 30, 2008
Accepted: December 15, 2008