Optimal Download Time in a cloud-assisted Peer-to-Peer Video on Demand service

Pablo Rodríguez-Bocca and Claudia Rostagnol

Abstract In this paper we present a mathematical model to optimize the average download time in a Peer-to-peer *Video on Demand* system, where a set of resources are available in the cloud to assist the service. First, we propose a simple optimization model based on a Markov chain. Then, we provide some numerical results based on simulations and optimizations using a GRASP method on a real scenario.

1 Introduction

Nowadays, lots of applications used to share contents over the Internet are based on the BitTorrent protocol [3, 5]. One of such applications is the GoalBit Video Platform [1], currently used to share live content over a P2P network. We are working on adding *Video on Demand* (VoD) support to GoalBit, using the standard BitTorrent protocol. In GoalBit, as in BitTorrent, end-users are called *peers*. They are classified in two groups: if the peer is downloading a content it is a *downloader*; when a peer finishes downloading it becames a *seeder*. There is also an entity or node named *tracker*, which knows all the peers that are sharing a content (seeding or downloading). GoalBit introduces a new type of node to the P2P network named *super-peer*. This kind of node has better bandwidth than a normal peer, and usually is in the network for very long periods of time (very stable peers). The super-peers are intended to store and upload the contents to normal peers (with a very short life in the system). In the current GoalBit protocol specification, super-peers are nodes managed by the operator of the platform and they are hosted in the cloud. We are thinking about the possibility of promoting peers to super-peers in future specifications. For more details about GoalBit specification please refer to [1].

The work in [3] provides an analysis of the estimated average time to download a file (*content*) on a BitTorrent network, assuming that the behavior of peers can be

Instituto de Computación, Facultad de Ingeniería, Universidad de la República, Montevideo, Uruguay. e-mail: prbocca@fing.edu.uy · e-mail: crostag@fing.edu.uy

modeled by a Markov chain. This work is generalized in [5], extending the model for several concurrent contents, assuming that most BitTorrent users download several files at the same time. Our goal is to distribute video files over the GoalBit platform depending on the storage capacity and videos' popularity, minimizing the average download time for end-users. To achieve that, we extend those models, so when a video becomes very popular and lots of users want to watch it, we generate more copies for this video in the cloud (if we have enough resources), in order to satisfy the users demand and do not increase the average download time of the system. When the video becames less popular, we can remove some copies to free space and resources. All this process is made automatically and dynamically based on the model that we defined. In Section 2 we introduce our combinatorial optimization problem based on a fluid model that extends the previous work [5]. In Section 3 we present a GRASP [4] metaheuristic solution for that problem. Finally, in Section 4, we show the performance of the solution in a real scenario, and we present general conclusions of our work.

2 Video-on-Demand Fluid Model

To understand the behavior of peers in a P2P system like GoalBit we should analyze the evolution, scalability, and sharing efficiency of peers. We extend the stochastic fluid model presented in [5], providing insightful results for performance issues and the downloading average time for several contents downloaded simultaneously. Since each peer can download more than one content at time t, peers are grouped in classes: $\{C^1, C^2, \ldots, C^K\}$, such that a peer is in the class C^i if it is downloading i contents at the same time. The data and variables of the model are shown in Figure 1.

K	available videos
s_j	size (in `kbits`) of video j
$x_j^i(t)$	*downloaders* in class C^i downloading video j at time t
$y_j^i(t)$	*seeders* in class C^i seeding video j at time t
$z_j^i(t)$	*super-peers* in class C^i seeding video j at time t
λ_j^i	arrival rate for peers in class C^i requesting video j (where $\sum_i \lambda_j^i$ is the j-th video popularity)
γ	departure rate of *seeders*
c	total download bandwidth for each peer (in `kbps`)
μ	total upload bandwidth for each peer (in `kbps`)
ρ	total upload bandwidth for each *super-peer* (in `kbps`)
η	video sharing effectiveness between peers ($\eta \in [0,1]$)

Fig. 1 Data and variables of the fluid model

To simplify the model representation we assume the following (as in [5]):

1. Resources are used equitably among the contents that are downloaded or served simultaneously. If the peer belongs to class C^i, each video that it downloads will have the i-th part of the peer's bandwidth. Since videos have different sizes, we divide the bandwidth (in `kbps`) by the video size in order to know the download rate (in files per second) for video j. Therefore, if the peer is in class C^i, the file

portion downloaded per second for content j is $c_j^i = \frac{c}{is_j}$. The same is applied to $\mu_j^i = \frac{\mu}{is_j}$ and $\rho_j^i = \frac{\rho}{is_j}$ $\forall i, j \in \{1 \dots K\}$.

2. Peers in class C^i, that at time t are downloading video j, receive from all other ***downloaders*** an amount of content proportional to the upload bandwidth μ_j^i and their population $x_j^i(t)$: $\frac{\mu_j^i x_j^i(t)}{\sum_k \mu_j^k x_j^k(t)} \sum_k \eta \mu_j^k x_j^k(t) = \eta \mu_j^i x_j^i(t)$.
3. Peers in class C^i, that at time t are downloading video j, receive from all the ***seeders*** an amount of content proportional to the download bandwidth c_j^i and their population $x_j^i(t)$: $\frac{c_j^i x_j^i(t)}{\sum_k c_j^k x_j^k(t)} \sum_k \mu_j^k y_j^k(t)$.
4. Peers in class C^i, that at time t are downloading video j, receive from all the ***super-peers*** an amount of content proportional to the download bandwidth c_j^i and their population $x_j^i(t)$: $\frac{c_j^i x_j^i(t)}{\sum_k c_j^k x_j^k(t)} \sum_k \rho_j^k z_j^k(t)$.

Supposing that we can model the problem as a Markovian chain, we want to know the peers' behavior (how x_j^i and y_j^i vary as a function of time). Modeling the behavior as a simple fluid model we get the following equation $\forall i, j \in \{1 \dots K\}$:

$$\frac{dx_j^i}{dt} = \lambda_j^i - \eta \mu_j^i x_j^i(t) - \frac{c_j^i x_j^i(t)}{\sum_k c_j^k x_j^k(t)} \sum_k \mu_j^k y_j^k(t) - \frac{c_j^i x_j^i(t)}{\sum_k c_j^k x_j^k(t)} \sum_k \rho_j^k z_j^k(t) \tag{1}$$

$$\frac{dy_j^i}{dt} = \eta \mu_j^i x_j^i(t) + \frac{c_j^i x_j^i(t)}{\sum_k c_j^k x_j^k(t)} \sum_k \mu_j^k y_j^k(t) + \frac{c_j^i x_j^i(t)}{\sum_k c_j^k x_j^k(t)} \sum_k \rho_j^k z_j^k(t) - \gamma y_j^i(t) \tag{2}$$

Assuming that the system will reach its *steady state*, where the number of peers is stable ($\frac{dx_j^i}{dt} = \frac{dy_j^i}{dt} = 0 \; \forall i, j \in \{1 \dots K\}$), we can calculate the steady state value for $x_j^i(t)$ and $y_j^i(t)$: Equations (1) and (2) assume that the bandwidth constraint is in the

$$\overline{x_j^i} = \max\{\frac{\lambda_j^i i s_j}{c}, \frac{i\lambda_j^i}{\gamma\mu\eta} \frac{\gamma s_j \sum_k \lambda_j^k - \mu \sum_k \frac{\lambda_j^k}{k} - \gamma\rho \sum_k \frac{z_j^k}{k}}{\sum_k \lambda_j^k}\} \qquad \overline{y_j^i} = \frac{\lambda_j^i}{\gamma} \tag{3}$$

upload capacity of the system, Equation (3) generalizes this, considering also that the bandwidth constraint can be at the download capacity of peers.

Based on this model, we want to minimize the average download time of the system, making an efficient use of resources, trying to find an optimal distribution of files in super-peers nodes. The average download time for any *downloader* at steady state can be computed applying *Little's law*: $T_j^i = \frac{x_j^i}{\lambda_j^i}$. Then, our problem can now be written as a *Combinatorial Optimization Problem* (COP), where we have to add new data:

- E_j^p indicates if the *super-peer* p has a copy of video j (p can seed video j). E_j^p is either 0 or 1: 1 if p has video j, 0 otherwise.
- S^p is the storage capacity of *super-peer* p (in `kbits`).

The optimization problem is shown in Figure 2.

$$\min_{E_j^p} \sum_{j=1}^{K}\sum_{i=1}^{K} \lambda_j^i T_j^i \quad = \quad \min_{E_j^p} \sum_{j=1}^{K}\sum_{i=1}^{K} \overline{x}_j^i$$

$$s.t.$$

$$(1)\sum_j E_j^p s_j \leq S^p \ \forall p \qquad (2)\sum_p E_j^p \geq 2 \ \forall j \qquad (3)\sum_k \frac{z_j^k}{k} \leq (\frac{c}{\mu} - \eta)\sum_k \frac{x_j^k}{k} - \sum_k \frac{y_j^k}{k} \ \forall j \qquad (4)\ z_j^i = \sum_p z_j^{i,p} \ \forall i,j$$

$$(5)\ u^{i,p} = \left| \sum_l E_l^p - i \right| \ \forall i,p \qquad (6)\ z_j^{i,p} \geq E_j^p - u^{i,p} \ \forall i,j,p \qquad (7)\ E_j^p \in \{0,1\}, z_j^i \in \{0,P\}, z_j^{i,p} \in \{0,1\}, u^{i,p} \in \mathbb{R}^+ \ \forall i,j,p$$

Fig. 2 Combinatorial Optimization Problem

The problem constraint (1) indicates that no *super-peers* can store more videos than its storage capacity. Additionally, in constraint (2) we define that each video must have at least one replica (each video must be stored in at least 2 *super-peers*). Also, the number of video replicas is limited by the peers' download capacities and the seeders' upload capacities for this video, as described in constraint (3). Finally, z_j^i can be computed from E_j^p as the number of super-peers that hosts content j and other $i-1$ contents. Constraints (4)-(7) specify this relationship between z_j^i and E_j^p using some auxiliary variables ($z_j^{i,p}$ and $u^{i,p}$).

3 Model optimization based on GRASP

Considering that the number of feasible solutions of the problem increases a lot with the size of the problem' instance, we will use a metaheuristic approach in order to solve it (we did not a complexity analysis of the problem at this time). GRASP [4] is a wellknown metaheuristic that we have been successfully using to solve other similar hard COPs [2]. It is an iterative process which operates in two phases. In the *Construction Phase* an initial feasible solution is built, whose neighborhood is then explored in the *Local Search Phase*. In Figure 3 we present a GRASP customization to solve our problem. During *construction phase* we must distribute the video files in the super-peers taking into account the constraints of the problem. First, we sort the files depending on their sizes, starting by the largest one. Then, we select the 20% larger files (not copied yet) to create the *Restricted Candidate List* (RCL), and choose one of them randomly to be put in at least two super-peers (selecting super-peers with more storage capacity first). The pseudo-code for this construction phase is shown in Figure 3(a). To improve the solution constructed in the first phase, a local search is applied as second phase. The improvement can be done in 2 ways, applying only one per iteration, selected randomly (both without breaking the problem constraints): (a) inserting a new copy of video k in the super-peer sp; (b) swaping two videos k_1 and k_2 from two super-peers sp_A y sp_B. The pseudo-code for this local search phase is shown in Figure 3(b).

```
Procedure RandomGreedy
Input: data_0
x ← emptySolution(data_0)
RCL = biggerFiles(data_0, x, 20%)
i = randomSelect(RCL)
while i > 0 do
    sp_1 ← fstBestSP(data_0, x, i)
    x ← x ⋃ Copy(i, sp_1)
    if checkRestrictions()
        sp_2 ← sndBestSP(data_0, x, i)
        if checkRestrictions()
            x ← x ⋃ Copy(i, sp_2)
        else
            x ← x ⋃ Remove(i, sp_2)
        end if
    else
        x ← x ⋃ Remove(i, sp_1)
    end if
    RCL = biggerFiles(data_0, x, 20%)
    i = randomSelect(RCL)
end while
return x
```

(a) Construction phase

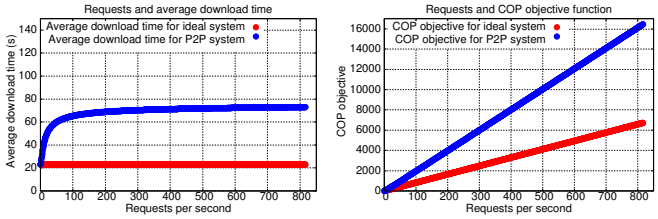
```
Procedure LocalSearch
Input: x
x* ← clone(x)
i ← 0
while i ≤ i_max do
    x_temp ← randomChange(x)
    if evaluate(x) < evaluate(x*)
        x* ← x_temp
        i = 0
    else
        i++
    end if
    x ← x_temp
end while
return x*
```

(b) Local Search phase

Fig. 3 Pseudo-cide for GRASP phases

4 Numerical Results and Discussion

We implement the COP in Matlab, and we calibrate the GRASP algorithm with generated instances. In this paper we present an application into a real scenario in order to show its potential. Using real information got from a local Internet Video-on-Demand Service Provider we construct a real scenario. From the log information of this service, we obtained the popularity and the size of the available video content. Specifically, our scenario has more than 700 videos (K), with an average size of 23 MB (s), 4 super-peers (P) with 100 GB of storage (S) and an upload rate of 80 Mbps (ρ), a peer download rate of 8 Mbps (c), and 4 Mbps of upload (μ), using an effectiveness of 50% ($\eta = 0.5$), having a seeders departure rate of 1 every 10 seconds ($\gamma = 0.1$).



(a) Average download time (b) COP objective function

Fig. 4 Comparing ideal system and P2P system

A lower boundary of the average download time can be computed as the time needed in a system with free upload capacity (i.e. when the download time is determined by the peers download rate). In this ideal scenario, the number of download-

ers is $\overline{x_{ideal}}^i_j = \frac{\lambda^i_j is_j}{c}$ and $T_{ideal}{}^i_j = \frac{\overline{x_{ideal}}^i_j}{\lambda^i_j}$. With data provided above, we computed an average ideal time of 23.1 seconds.

In order to determine the scalability of the service, we stress the system keeping the popularity proportional with the real data (i.e. multiplying the real λ by an incremental factor). These results are shown on Figure 4(a), where we can see that our P2P system is a bit far from the ideal system (3 times worse), but it is very scalable since the performance is stable regarding to the increment of requests. With 163 requests per second the average download time is 67 seconds, while with 816 requests the average time is 73 seconds. In Figure 4(b) we show the evolution of the COP objective function for the ideal system and for the P2P system. Notice that to reach the same level of service (the same average download time) in a client-server system we should increase the number of servers (or super-peers) proportionally with the end-user requests, while in the P2P system we have a natural scalability with the growing resources offered by the users (downloaders and seeders).

Therefore, we can conclude that our P2P solution is a good and very scalable approach. Although in this scenario we have a solution 3 times worse, it is better than a client-server option where we should increase the number of servers depending on the number of client requests. We also expect that the efficiency will be more evident in largest deployments.

Currently, we are working to include this model inside the GoalBit platform to test it in a real production scenario.

References

1. Bertinat, M.E., Vera, D.D., Padula, D., Robledo, F., Rodríguez-Bocca, P., Romero, P., Rubino, G.: Goalbit: The first free and open source peer-to-peer streaming network. In: Proceedings of the 5th international IFIP/ACM Latin American conference on Networking (LANC'09), pp. 83–93. ACM, New York, USA (2009)
2. Martínez, M., Morón, A., Robledo, F., Rodríguez-Bocca, P., Cancela, H., Rubino, G.: A GRASP algorithm using RNN for solving dynamics in a P2P live video streaming network. In: 8th International Conference on Hybrid Intelligent Systems (HIS'08). Barcelona, Spain (2008). URL http://his2008.lsi.upc.edu/
3. Qiu, D., Srikant, R.: Modeling and performance analysis of bittorrent-like peer-to-peer networks. In: Proceedings of SIGCOMM'04, pp. 367–378. ACM, ACM, New York, NY, USA (2004). DOI http://doi.acm.org/10.1145/1015467.1015508
4. Resende, M.G.C., Ribeiro, C.C.: Greedy Randomized Adaptive Search Procedures. In F. Glover and G. Kochenberger, editors, Handbook of Methaheuristics, Kluwer Academic Publishers (2003)
5. Tian, Y., Wu, D., Ng, K.W.: Analyzing multiple file downloading in bittorrent. In: Proceedings of ICPP'06, pp. 297–306. IEEE (2006). DOI 10.1109/ICPP.2006.23