[GZLO99] *Carsten Griwodz, Michael Zink, Michael Liepert, Giwon On;* **Analytical Model of Patching VoD Cache Hierarchies;** Technical Report 03, KOM TU Darmstadt, September 1999.

DARMSTADT UNIVERSITY OF

**Industrial Process and System Communications (KOM)**

**Department of Electrical Engineering & Information Technology**
Merckstraße 25
D-64283 Darmstadt • Germany

| | |
|---|---|
| Phone: | +49 6151 166150 |
| Fax: | +49 6151 166152 |
| Email: | info@KOM.tu-darmstadt.de |
| URL: | http://www.kom.e-technik.tu-darmstadt.de/ |

Carsten Griwodz, Michael Zink, Michael Liepert, Giwon On

# Analytical Model For Patching VoD Cache Hierarchies

Technical Report TR-KOM-1999-03

17. April 2000

# Analytical Model For Patching VoD Cache Hierarchies

Carsten Griwodz[1], Michael Zink[1], Michael Liepert[1], Giwon On[1], Ralf Steinmetz[1,2]
[1]KOM, Darmstadt University of Technology, Merckstrasse 25, 64283 Darmstadt, Germany
[2]GMD IPSI, Dolivostr. 15, 64293 Darmstadt, Germany
medianode@kom.tu-darmstadt.de

## 1 MODEL OVERVIEW

We calculate cost functions for various approaches of serving movies to users in hierarchical distribution systems with the topology of binary trees. Figure 1 is a sketch of the base model topology

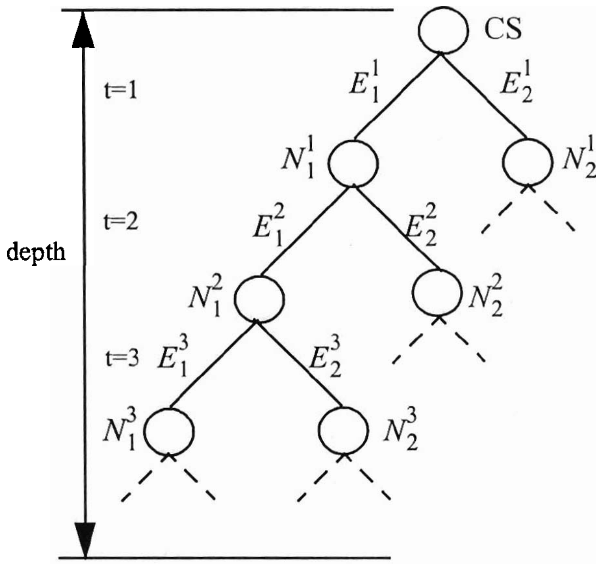

*Figure 1:* binary tree of analytical distribution system model

central server *CS*, optional cache servers $N_i^t$ with an index *i* at depth *t* in the binary tree, and network links $E_i^t$. Table 1 lists the symbols that are used in the formulas, and Table 2 presents the formulas for calculating the cost of the distribution systems.In Appendix B we provide the details on terms, assumptions and calculations that are presented in this section. The most important limitations of the model are summarized below, but still, this analysis motivates us to realize caching with *patching*. We get a strong hint to combine caching with *patching* in the example below, for a VoD system with rather realistic characteristics, following the assumptions of the analysis.

The effort to set up the system is modeled as an abstract "cost" for basic server installations (including central server and cache servers), cost of server support for concurrent stream deliveries, the cost of concurrent streams support by each network link, and cost for the storage of movies in cache servers. As we assume all movie files to be optimally located in the caching hierarchy, there is no cost for transporting the movies to store and cache and for unnecessary copies. There are several noteworthy aspects to this assumption:

- assuming a perfect distribution of movies to cache servers according to their long-term relevance would also render movements due to relocation minimal

This work is supported in part by a grant of: Volkswagen-Stiftung, D-30519 Hannover, Germany.

- for a downstream movement, caches that work according to our approach do not generate additional network load because they work in write-through mode - upstream movement is certainly missing
- if caching strategies are not sufficiently elaborate (or centrally controlled), they will react to short-term or at least to day-time variations in the request patterns, these calculations will be extremely optimistic

The numerical optimization assumes a distribution of movie hit probabilities according to the Zipf distribution. Although various papers state that the Zipf distribution describes the distribution of hit probabilities at any given time very well, a caching architecture is unable to achieve a distribution according to Zipf.

- The relevance of movies is changing with respect to other movies, which implies that their index value in the Zipf distribution is changing,
- Hit rates do not typically conform perfectly to the Zipf distribution because of user behavior. The divergence is greater for small user populations, which means that distribution systems without an exchange of hit rate information will estimate a movies popularity less exact than a centrally coordinated system.
  Movies must be relocated between cache servers according to their estimated relevance. This may be done predictively (which reduced accurateness of the estimation), so the optimal location for each movie is achieved timely, but such relocations do still incur additional network and server load.
- Homogenous distribution systems are unrealistic.
- Not all movies have equal length and data rate.

Note, that a non-hierarchical approach will probably result in additional savings but for hierarchies, any algorithm should be unable to reach the optimum that can be computed numerically from the formulas in Table 2.

To verify the effects of these computations, we present an example that demonstrates the vast options for savings. This example is simplified from the reality that we envision with the combination of *patching* and caching. For example, we assume that *patching* is implemented in the clients, which is not realistic in a widely distributed network of heterogeneous clients.

Table 1: Elements used in folrmulas

| **Symbol** | **Meaning** | **Symbol** | **Meaning** |
|---|---|---|---|
| $S_0$ | Basic cost of a server/cache server installation. | $S_1$ | Cost for one supported stream of a server. |
| $C_t^E$ | Cost for one supported stream on a network link at level *t*. | $C_t^N$ | Cost for the storage needed to store one movie in a cache server. |
| $M$ | Number of available movies. | $P(m)$ | Hit probability of movie *m*. |
| t(m) | Optimal tree level for caching movie *m*. | r(m) | Optimal patching window for movie *m*. |

Table 2: Analysis of cost effects of patching on caching hierarchies, cf. Section 2

| **Distribution Method** | **Calculated Cost Formula** |
|---|---|
| unicast directly from central server | $S_0 + 2^d \cdot S_1 + 2^d \sum_{t=1}^{d} C_t^E$ |
| unicast with caches | $\left[\sum_{t=1}^{d-1}\left(2^t \cdot \bigcup_{m \in M} \delta_{t(m)}(t)\right)\right] \cdot S_0 + 2^d \cdot S_1 + \sum_{m \in M}\left[P(m)2^d \sum_{t=t(m)+1}^{d} C_t^E + 2^{t(m)} C_{t(m)}^N\right]$ |
| greedy patching from central server | $S_0 + \left[2^{d-1} + \sum_{m \in M}\left(1-(1-\eta_m)^{2^d}\right)\right] \cdot S_1 + \sum_{m \in M}\sum_{t=1}^{d}\left[\left(2^t - 2^t \cdot (1-\eta_m)^{2^{d-t}} + 2^{d-1} \cdot \eta_m\right) \cdot C_t^E\right]$ ,where $\eta_m = P(m)$ |
| patching with limited buffer from central server | $S_0 + \left[\sum_{m \in M}\left(2^{d-1} \cdot \eta_m + r(m)\left(1-(1-\eta_m)^{2^d}\right)\right)\right] \cdot S_1$ $+ \sum_{m \in M}\sum_{t=1}^{d}\left[\eta_m \cdot 2^{d-1} + \left(2^t \cdot r(m) \cdot \left(1-(1-\eta_m)^{2^{d-t}}\right)\right) \cdot C_t^E\right]$ ,where $\eta_m = \frac{1}{r(m)} \cdot P(m)$ |
| patching with caches | $\left[\sum_{t=0}^{d-1}\left(2^t \cdot \delta\left(\bigcup_{m \in M}(t = t(m))\right)\right)\right] \cdot S_0 + \sum_{m \in M}(2^{t(m)} C_{t(m)}^N)$ $+ \left[\sum_{m \in M}\left(2^{d-1} \cdot \eta_m + 2^{t(m)} \cdot r(m)\left(1-(1-\eta_m)^{2^{d-t(m)}}\right)\right)\right] \cdot S_1$ $+ \sum_{m \in M}\sum_{k=1}^{d-t(m)}\left[\left(\eta_m \cdot 2^{d-1} + 2^{t(m)} \cdot 2^k \cdot r(m) \cdot \left(1-(1-\eta_m)^{2^{d-t(m)-k}}\right)\right) \cdot C_k^E\right]$ ,where $\eta_m = \frac{1}{r(m)} P(m_m)$ |

In our example, the movie probabilities are distributed according to the Zipf distribution:

$$P(drawm_m) = z(m) = \frac{C}{m}, C = \sum_{m \in M} \frac{1}{index(m)}$$

Besides the predefinitions from the analytical model, we define

- 500 different movies
- $2^{20}$ active users (i.e. a binary distribution depth of 10, where most nodes do not contain a server)
- a cost of 25000 $ for a basic server installation
- a cost of 100 $ for each concurrent high quality movie stream supported by a server
  a cost of 350 $ for each concurrent high quality movie stream supported on a network link
- a cost of 1000 $ for storage to hold one high quality movie

The location of the caches in the distribution hierarchy for examples 2 and 5 was not optimized. Rather, the caches were moved heuristically upstream until no immediate gain was perceived any more. For the example 2, "unicast with caches", the approach "installed" caches at levels $t$=12, 10, 8, 6 and 4 in the order to decreasing movie popularity. For the example 5, "patching with caching", the approach "installed" caches at levels $t$=9, 7, 3, 5 and 1. The heuristic prohibited to choose the level 0 for the least popular movies which would have been roughly three quarters of all movies

Table 3: Example for theoretical effect of the various methods

| **Modeled Distribution Method** | **Calculated System Cost** |
|---|---|
| 1. unicast from central server | 7,445 Mio $ |
| 2. unicast with caches | 4,664 Mio $ |
| 3. greedy patching from central server | 3,722 Mio $ |
| 4. patching with limited buffer from central server | 375 Mio $ |
| 5. patching with caching | 276 Mio $ |

These numbers indicate, that there are scenarios with a large potential for savings in the joint use of the *patching* and caching techniques. When (costly) caches are introduced in a *patching* distribution system, savings are made with much less expensive necessary system links and storage space (cf. the last two rows in Table 3).

Although this model and these numbers are quite illusionary, and we can not expect clients that to implement *patching* buffers and *patching*-capable protocols, this potential for savings demonstrates that:

- the use of cache servers generates savings that make up for their installation cost
- *patching* with optimized window sizes is the major advancement in savings
- The most important issue for our architecture is:
  The installation of caches in conjunction with *patching* does not eliminate the effect of *patching*. With an appropriately dimensioned cache server, it will even increase the savings by keeping the most popular titles in the cache. Thus, we can proceed to build a wide-area caching architecture that relies on *patching* for wide-area distribution of the videos to cache servers that act of proxies for clients without these specific features.

## 2 MODEL CALCULATIONS

For simplification, our example calculations assume binary distribution trees as shown in Figure 1. With appropriate weight and cost settings we can model a limited class of balanced, hierarchical distribution topologies compliant with these assumptions. We are currently working on a more complex and realistic simulation for video caching integrating these techniques in order to receive more detailed results.

We think of a binary tree distribution architecture of depth $d$. We denote a link in the tree by its level $t$ and its index $n$ at this level: $E_n^t$. If we select an arbitrary link, it is called $E^t$. Similarly, cache servers are labeled $N_n^t$ and $N^t$, respectively. For convenience, we consider $N^d$ a client rather than a cache server. We assume the cost per concurrent video stream to be the same $C_t^E$ for each link $E_n^t$ on one level *t*. Also, we assume the hard disk cost for one video to be $C_t^N$ at each cache server $N_n^t$ on one level *t*. The numbers of links and caches at one level *t*are $2^t$

We assume a set of movies *M*. All of these movies $m \in M$ have the same length, measured in time, $L_l$ and the same data rate, but possibly different draw probabilities *P(m)*. In caching scenarios, we assume that each cached movie $m_i$ is stored in all caches of one optimally chosen level *t(m)*.

The necessity to have sufficiently large central servers that are able to handle the number of streams that are concurrently requested imposes a cost $S_0$ for the basic installation of each server, and a cost $S_1$ for each concurrent stream that is supported by a server. Each end-user in the system is watching exactly one video at any time, ie. $\sum_{m \in M} P(m) = 1$. The number of clients is very big compared to the number of different movies and active (cache) servers, the popularity of movies is constant for all clients. This gives us draw probablities being independent of time and hierarchy location, but also gives the problematic postulation of a majority of inactive, thus zeroed cache servers. We enforce this by defining the base server setup cost $S_0$ sufficiently high.

## 2.1 Unicast: No Patching, No Caching

The simplest approach to deliver video is the distribution from a central server via unicast. This allows all kinds of video-on-demand features, but is intense in terms of network as well as server load. We calculate costs for such an approach first.

Since there are no movies stored in the caches there will be no storage costs: $C_t^N = 0, \forall t$

Network costs for each currently running movie: $\sum_{t=1}^{d} C_t^E$, which are the cost of a complete link from the central server to the end-user. As every client is watching exactly one movie at any point of time, the overall network cost for streaming is $\sum_{m \in M}\left(P(m)2^d \sum_{t=1}^{d} C_t^E\right) = 2^d \sum_{t=1}^{d} C_t^E$ The interarrival time is irrelevant in this case, because no streams are shared. With this and a number of clients of $2^d$, the central server approach has an overall cost of $S_0 + 2^d\ S_1 + 2^d \sum_{t=1}^{d} C_t^E$

## 2.2 Unicast: No Patching, Caching

### 2.2.1 Network Cost

This implies that networking costs are generated only for the delivery of the movie from the cache server to the clients that are located downstream from this cache server or, in terms of the binary tree, in each subtree with a root node at level *t(i)*. The networking costs for this movie and for this subtree of depth *d-t(m)* can be calculated as in section 2.1:

$$P(m) \cdot 2^{d - t(m)} \sum_{t = t(m) + 1}^{d} C_t^E$$

Although the formula concerning the distribution probability of the movies does still apply in this case (the sum of probabilities equals 1), this should not be integrated into this formula, because the optimal level *t(m)* is different for each movie, depending on its probability.

### 2.2.2 Server Cost

Since there are $2^{t(m)}$ cache servers at level *t(m)*, the above networking cost occurs $2^{t(m)}$ times. The cost generated by the movie *m* that is stored at level *t(m)* is then $2^{t(m)} \cdot P(m) \cdot 2^{d - t(m)} \sum_{t = t(m) + 1}^{d} C_t^E = P(m) \cdot 2^d \sum_{t = t(m) + 1}^{d} C_t^E$

The resulting storage cost for a movie *m* on all cache servers at level *t(m)* is $2^{t(m)} C_{t(m)}^N$

The cost of the capacity needed by this cache server depends on the average number of concurrent streams it has to serve for each movie *m*. This is calculated from the hit probability of the movie and the number of clients that the cache server serves. The setup cost for a needed cache server on level *t* is

$$S_0 + S_1 \cdot 2^{d-t} \cdot \mathop{}_{m \in M} P(m) \cdot \delta(t(m) = t) \qquad \text{where } \delta(p) = \begin{cases} 1, p \text{ is true} \\ 0, p \text{ is false} \end{cases}$$

A cache server has to be set up if its level is the optimal cache level for any movie, thus installation cost for serving clients is as . $\mathop{}_{t=0}^{d-1} \left[ 2^t \delta\left( \bigcup_{m \in M} (t(m) = t) \right) \cdot \left( S_0 + S_1 2^{d-t} \sum_{m \in M} P(m) \cdot \delta(t(m) = t) \right) \right]$

As we assume a constant system state, there are no cost to store or stream movies on the root server cached elsewhere.

Simplified and increased by the network cost, this gives the following formula for the overall cost for our model with caching: $\left[ \sum_{t=0}^{d-1} \left( 2^t \cdot \bigcup_{m \in M} \delta_{t(m)}(t) \right) \right] \cdot S_0 + 2^d \cdot S_1 + \mathop{}_{m \in M} \left[ P(m) 2^d \mathop{}_{t = t(m) + 1}^{d} C_t^E + 2^{t(m)} C_{t(m)}^N \right]$

## 2.3 Greedy Patching with central server

The simplest form of Patching is Greedy Patching without buffering limits at the clients. Besides the fact that clients will be overly expensive when they are built to buffer complete movies, we have shown in [1] that the optimal restart time in terms of server load depends on *P(m)* and thus, the largest required buffer does not need to hold a complete movie.

However, we assume this kind of Patching to find an approximation for the cost of a distribution system. Assume a binary distribution tree of depth *d*, caching is not applied in this tree.

For each movie *m*, we define $\eta_m = P(m)$ for ease of reuse of the formulas.

### 2.3.1 Server effort

Since this approach is using a central server, $S_0$ is needed only once. The number of streams that need to be served concurrently is also reduced in comparison to the unicast case with a central server. The formula is derived as in section 2.3.2, and yields the setup cost, the basic server cost for multicast streams of *m* and the total cost of unicast patch streams: $S_0 + \left(1-(1-\eta_m)^{2^d}\right) \cdot S_1 + 2^{d-1} S_1$

### 2.3.2 Multicast portion

First, we try to calculate the network load that is generated at each level of the binary tree due to the probability of a joint stream for multiple clients; ie. we want to find a formula for savings of network bandwidth in the upper levels of the binary tree. We assume a random distribution of the clients that share a stream of movie *m* in the overall set of clients. The probability of a network link to be involved in a multicast playout of a specific movie is the probability, that any client below demands that specific movie. This probability is

$$2^{d-t}$$

which means that at each level *t*, an average of $\left(1-(1-\eta_m)^{2^{d-t}}\right) \cdot 2^t$ links are involved in the same multicast of movie *m*, and a cost that is generated at level t by the multicast streams is $\left(1-(1-\eta_m)^{2^{d-t}}\right) \cdot 2^t \cdot C_t^E$

### 2.3.3 Unicast portion

At the same time, the unicast patches need to be distributed to the clients. These unicast patches require a direct transmission from the central server to the end-user, and this unicast transmission behaves mainly like a regular video transmission according to section 2.1. The major difference is that the length of a unicast patch is less than a full length video transmission rather the length of the unicast patch is on average 1/2 of the patching window, which is in this case the full movie length [1]. Thus, the load of unicast streams at level *t* is in this case: $\frac{1}{2}(\eta_m 2^d C_t^E) = \eta_m \cdot 2^{d-1} \cdot C_t^E$

### 2.3.4 Overall cost

When the unicast and multicast formulas are combined, the overall cost at level *t* is $\left[2^t \left(1-(1-\eta_m)^{2^{d-t}}\right) + 2^{d-1} \cdot \eta_m\right] \cdot$

and the overall cost of distribution of all movies, through the whole tree is the summation

$$S_0 + \left[2^{d-1} + \sum_{m \in M} \left(1-(1-\eta_m)^{2^d}\right)\right] \cdot S_1 + \sum_{m \in M} \sum_{t=1}^{d} \left(2^t - 2^t \cdot (1-\eta_m)^{2^{d-t}} + 2^{d-1} \cdot \eta_m\right) \cdot C_t^E$$

### 2.3.5 Savings Compared To Unicast With Central Server

The average Greedy Patching case is not costlier than the unicast with central server. With the inequality

$$\forall t \le d, \forall m \in M \text{ with } P(m) < 1:$$

$$1-(1-P(m))^{2^{d-t}} \le 1-(1-P(m)) = P(m) \le P(m) 2^{d-t-1}$$

the comparison of the server efforts to section 2.1 gives a possible saving: $\left[2^{d-1} + \sum_{m\in M}\left(1-(1-\eta_m)^{2^d}\right)\right]\cdot S_0 \le 2^d\cdot S_1$

Together with the comparison of network load below this is a first hint to integrate Patching in the delivery system.

$$\sum_{m\in M}\sum_{t=1}^{d}\left[2^t - 2^t\cdot(1-P(m))^{2^{d-t}} + 2^{d-1}\cdot P(m)\right]\cdot C_t^E \le \sum_{m\in M}\sum_{t=1}^{d} C_t^E P(m)2^d = 2^d\sum_{t=1}^{d} C_t^E$$

## 2.4 Patching with limited buffer and central server

When the restart rate *r(m)* for the multicast stream of a specific movie *m* is increased, i.e., the window size to covered by patch streams is reduced, then the probability that clients receive the same multicast is reduced, but the use of a limited patching window size realistically limits the needed buffer size at the client. As in [1], we assume for simplicity that the multicast transmissions are repeated regularly, and that the length of such a cycle is called the restart time. The restart time here is expressed as a portion of the movie length: $\frac{1}{r(m)}L_1$ The probability for a client to join a specific multicast playout of a specific movie *m* follows as $\eta_m = \frac{1}{r(m)}\cdot P(m)$

### 2.4.1 Server effort

With a patching window of $\frac{1}{r(m)}L_1$, we calculate the average number of concurrent unicast patches to be served according section 2.3.1. Ie. that the number of concurrent unicast streams for *m* is $\frac{1}{2}\cdot 2^d\cdot\frac{P(m)}{r(m)} = 2^{d-1}\eta_m$

This yields the number of concurrent unicast streams that need to be supported by the central server at each time. Unlike for Greedy Patching, *r(m)* is assumed to be optimal but different for different *m*. The server cost for unicast streams is $2^{d-1}S_1\sum_{m\in M}\eta_m$ which is inverse proportional to the restart rate! The server cost per movie *m* for multicast streams is increasing with the restart rate: $r(m)\left(1-(1-\eta_m)^{2^d}\right)\cdot S_1$

### 2.4.2 Multicast portion

The multicast cost of the distribution system is calculated as in the section 2.3.2, with the redefined $\eta_m$. As *r(m)* copies of the stream can be active at any time, the average load at level *t* is $r(m)\cdot\left(1-(1-\eta_m)^{2^{d-t}}\right)\cdot 2^t\cdot C_t^E$

### 2.4.3 Unicast portion

The computation of the unicast load of the distribution system is the same as in the last section, but with the reduced average length of the unicast patch streams, the values differ. With the redefined value $\eta_m$, however, the formula remains the same as in the previous section, the cost a level *t* is $\eta_m\cdot 2^{d-1}\cdot C_t^E$

### 2.4.4 Overall cost

The combined costs of elements yield the average cost for a distribution system that uses Patching with a central server and movie-dependent window sizes for the delivery of unicast patch streams.

$$S_0 + \left[\sum_{m \in M} \left(2^{d-1} \cdot \eta_m + r(m)\left(1 - (1-\eta_m)^{2^d}\right)\right)\right] \cdot S_1 \\ + \sum_{m \in M}\left[\sum_{t=1}^{d}\left(C_t^E\left(\eta_m \cdot 2^{d-1} + r(m) \cdot \left(1-(1-\eta_m)^{2^{d-t}}\right) \cdot 2^t\right)\right)\right]$$

## 2.5 Patching with Caching

We assume that for large hierarchies, savings can be increased by combining patching with caching. To verify this, we start with the inner part of the formula from section 2.4.4. We assume that for each movie *m* there is exactly one level *t(m)*, where this movie is cached in all servers. We calculate the server cost for one cache server for *m*. The depth of the distribution sub-tree is *d-t(m)*. Analogous to section 2.4, for this movie, the effort to support streams on the cache server (without basic setup $S_0$ and the movie storage cost $C_t^N$, which can be calculated as in section 2.2.2) and on the network links below this server is given by

$$S_1\left(2^{d-t(m)-1} + r(m)\left(1-(1-\eta_m)^{2^{d-t(m)}}\right)\right) + \sum_{k=1}^{d-t(m)}\left(C_k^E\left(\eta_m \cdot 2^{d-t(m)-1} + r(m) \cdot \left(1-(1-\eta_m)^{2^{d-t(m)-k}}\right) \cdot 2^k\right)\right)$$

This cost occurs once for each server at this level, and that cost, in turn, needs to be calculated once for each movie *m*. This results in an overall cost for Patching with caching of

$$\sum_{m \in M}\left[2^{t(m)} \cdot \left(S_1\left(2^{d-t(m)-1} + r(m)\left(1-(1-\eta_m)^{2^{d-t(m)}}\right)\right) + \sum_{k=1}^{d-t(m)}\left(C_k^E\left(\eta_m \cdot 2^{d-t(m)-1} + r(m) \cdot \left(1-(1-\eta_m)^{2^{d-t(m)-k}}\right) \cdot 2^k\right)\right)\right)\right] \\ + S_0 \cdot \sum_{t=0}^{d-1}\left(2^t \cdot \delta\left(\bigcup_{m \in M}(t = t(m))\right)\right) + \sum_{m \in M}\left(2^{t(m)} C_{t(m)}^N\right)$$

**REFERENCES**

C. Griwodz, M. Liepert, M. Zink, R. Steinmetz. Tune to Lamda Patching. WISP 1999, Atlanta, GA, USA, May 1999

DARMSTADT
UNIVERSITY
OF

**Industrial Process and System Communications (KOM)**

**Department of Electrical Engineering & Information Technology**
Merckstraße 25
D-64283 Darmstadt • Germany
Phone: +49 6151 166150
Fax: +49 6151 166152
Email: info@KOM.tu-darmstadt.de
URL: http://www.kom.e-technik.tu-darmstadt.de/

Carsten Griwodz, Michael Zink, Michael Liepert, Giwon On

# Analytical Model For Patching VoD Cache Hierarchies

Technical Report TR-KOM-1999-03

05. July 2000

# Analytical Model For Patching VoD Cache Hierarchies

Carsten Griwodz[1], Michael Zink[1], Michael Liepert[1], Giwon On[1], Ralf Steinmetz[1,2]
[1]KOM, Darmstadt University of Technology, Merckstrasse 25, 64283 Darmstadt, Germany
[2]GMD IPSI, Dolivostr. 15, 64293 Darmstadt, Germany
medianode@kom.tu-darmstadt.de

## 1 MODEL OVERVIEW

We calculate cost functions for various approaches of serving movies to users in hierarchical distribution systems with the topology of binary trees. Figure 1 is a sketch of the base model topology

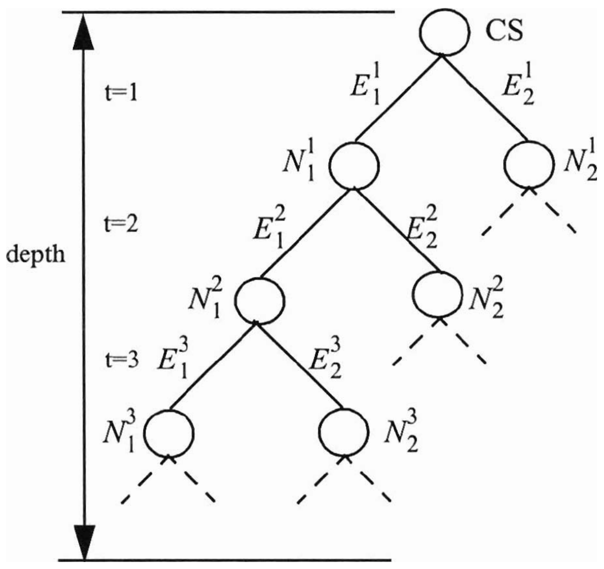

*Figure 1:* binary tree of analytical distribution system model

central server *CS*, optional cache servers $N_i^t$ with an index *i* at depth *t* in the binary tree, and network links $E_i^t$. Table 1 lists the symbols that are used in the formulas, and Table 2 presents the formulas for calculating the cost of the distribution systems.In Appendix B we provide the details on terms, assumptions and calculations that are presented in this section. The most important limitations of the model are summarized below, but still, this analysis motivates us to realize caching with *patching*. We get a strong hint to combine caching with *patching* in the example below, for a VoD system with rather realistic characteristics, following the assumptions of the analysis.

The effort to set up the system is modeled as an abstract "cost" for basic server installations (including central server and cache servers), cost of server support for concurrent stream deliveries, the cost of concurrent streams support by each network link, and cost for the storage of movies in cache servers. As we assume all movie files to be optimally located in the caching hierarchy, there is no cost for transporting the movies to store and cache and for unnecessary copies. There are several noteworthy aspects to this assumption:

- assuming a perfect distribution of movies to cache servers according to their long-term relevance would also render movements due to relocation minimal

This work is supported in part by a grant of: Volkswagen-Stiftung, D-30519 Hannover, Germany.

- for a downstream movement, caches that work according to our approach do not generate additional network load because they work in write-through mode - upstream movement is certainly missing
- if caching strategies are not sufficiently elaborate (or centrally controlled), they will react to short-term or at least to day-time variations in the request patterns, these calculations will be extremely optimistic

The numerical optimization assumes a distribution of movie hit probabilities according to the Zipf distribution. Although various papers state that the Zipf distribution describes the distribution of hit probabilities at any given time very well, a caching architecture is unable to achieve a distribution according to Zipf.

- The relevance of movies is changing with respect to other movies, which implies that their index value in the Zipf distribution is changing,
- Hit rates do not typically conform perfectly to the Zipf distribution because of user behavior. The divergence is greater for small user populations, which means that distribution systems without an exchange of hit rate information will estimate a movies popularity less exact than a centrally coordinated system.
- Movies must be relocated between cache servers according to their estimated relevance. This may be done predictively (which reduced accurateness of the estimation), so the optimal location for each movie is achieved timely, but such relocations do still incur additional network and server load.
- Homogenous distribution systems are unrealistic.
- Not all movies have equal length and data rate.

Note, that a non-hierarchical approach will probably result in additional savings but for hierarchies, any algorithm should be unable to reach the optimum that can be computed numerically from the formulas in Table 2.

To verify the effects of these computations, we present an example that demonstrates the vast options for savings. This example is simplified from the reality that we envision with the combination of *patching* and caching. For example, we assume that *patching* is implemented in the clients, which is not realistic in a widely distributed network of heterogeneous clients.

Table 1: Elements used in folrmulas

| **Symbol** | **Meaning** | **Symbol** | **Meaning** |
|---|---|---|---|
| $S_0$ | Basic cost of a server/cache server installation. | $S_1$ | Cost for one supported stream of a server. |
| $C_t^E$ | Cost for one supported stream on a network link at level *t*. | $C_t^N$ | Cost for the storage needed to store one movie in a cache server. |
| $M$ | Number of available movies. | $P(m)$ | Hit probability of movie *m*. |
| t(m) | Optimal tree level for caching movie *m*. | r(m) | Optimal patching window for movie *m*. |

Table 2: Analysis of cost effects of patching on caching hierarchies, cf. Section 2

| **Distribution Method** | **Calculated Cost Formula** |
|---|---|
| unicast directly from central server | $S_0 + 2^d \cdot S_1 + 2^d \sum_{t=1}^{d} C_t^E$ |
| unicast with | $\left[\sum_{t=1}^{d-1}\left(2^t \cdot \bigcup_{m \in M} \delta_{t(m)}(t)\right)\right] \cdot S_0 + 2^d \cdot S_1 + \sum_{m \in M}\left[P(m)2^d \sum_{t=t(m)+1}^{d} C_t^E + 2^{t(m)} C_{t(m)}^N\right]$ |
| greedy patching from central server | $S_0 + \left[2^{d-1} + \sum_{m \in M}\left(1-(1-\eta_m)^{2^d}\right)\right] \cdot S_1 + \sum_{m \in M}\sum_{t=1}^{d}\left[\left(2^t - 2^t \cdot (1-\eta_m)^{2^{d-t}} + 2^{d-1} \cdot \eta_m\right) \cdot C_t^E\right]$ ,where $\eta_m = P(m)$ |
| patching with limited buffer from central server | $S_0 + \left[\sum_{m \in M}\left(2^{d-1} \cdot \eta_m + r(m)\left(1-(1-\eta_m)^{2^d}\right)\right)\right] \cdot S_1 + \sum_{m \in M}\sum_{t=1}^{d}\left[\eta_m \cdot 2^{d-1} + \left(2^t \cdot r(m) \cdot \left(1-(1-\eta_m)^{2^{d-t}}\right)\right) \cdot C_t^E\right]$ ,where $\eta_m = \frac{1}{r(m)} \cdot P(m)$ |
| patching with caches | $\left[\sum_{t=0}^{d-1}\left(2^t \cdot \delta\left(\bigcup_{m \in M}(t = t(m))\right)\right)\right] \cdot S_0 + \sum_{m \in M}\left(2^{t(m)} C_{t(m)}^N\right) + \left[\sum_{m \in M}\left(2^{d-1} \cdot \eta_m + 2^{t(m)} \cdot r(m)\left(1-(1-\eta_m)^{2^{d-t(m)}}\right)\right)\right] \cdot S_1 + \sum_{m \in M}\sum_{k=1}^{d-t(m)}\left[\left(\eta_m \cdot 2^{d-1} + 2^{t(m)} \cdot 2^k \cdot r(m) \cdot \left(1-(1-\eta_m)^{2^{d-t(m)-k}}\right)\right) \cdot C_k^E\right]$ ,where $\eta_m = \frac{1}{r(m)} P(m_m)$ |

In our example, the movie probabilities are distributed according to the Zipf distribution:

$$P(drawm_m) = z(m) = \frac{C}{m}, C = \sum_{m \in M} \frac{1}{index(m)}$$

Besides the predefinitions from the analytical model, we define

500 different movies
$2^{20}$ active users (i.e. a binary distribution depth of 10, where most nodes do not contain a server)
a cost of 25000 \$ for a basic server installation
a cost of 100 \$ for each concurrent high quality movie stream supported by a server
a cost of 350 \$ for each concurrent high quality movie stream supported on a network link
a cost of 1000 \$ for storage to hold one high quality movie

The location of the caches in the distribution hierarchy for examples 2 and 5 was not optimized. Rather, the caches were moved heuristically upstream until no immediate gain was perceived any more. For the example 2, "unicast with caches", the approach "installed" caches at levels $t$=12, 10, 8, 6 and 4 in the order to decreasing movie popularity. For the example 5, "patching with caching", the approach "installed" caches at levels $t$=9, 7, 3, 5 and 1. The heuristic prohibited to choose the level 0 for the least popular movies which would have been roughly three quarters of all movies

Table 3: Example for theoretical effect of the various methods

| **Modeled Distribution Method** | **Calculated System Cost** |
|---|---|
| 1. unicast from central server | 7,445 Mio $ |
| 2. unicast with caches | 4,664 Mio $ |
| 3. greedy patching from central server | 3,722 Mio $ |
| 4. patching with limited buffer from central server | 375 Mio $ |
| 5. patching with caching | 276 Mio $ |

These numbers indicate, that there are scenarios with a large potential for savings in the joint use of the *patching* and caching techniques. When (costly) caches are introduced in a *patching* distribution system, savings are made with much less expensive necessary system links and storage space (cf. the last two rows in Table 3).

Although this model and these numbers are quite illusionary, and we can not expect clients that to implement *patching* buffers and *patching*-capable protocols, this potential for savings demonstrates that:

- the use of cache servers generates savings that make up for their installation cost
- *patching* with optimized window sizes is the major advancement in savings
- The most important issue for our architecture is:
  The installation of caches in conjunction with *patching* does not eliminate the effect of *patching*. With an appropriately dimensioned cache server, it will even increase the savings by keeping the most popular titles in the cache. Thus, we can proceed to build a wide-area caching architecture that relies on *patching* for wide-area distribution of the videos to cache servers that act of proxies for clients without these specific features.

## 2 MODEL CALCULATIONS

For simplification, our example calculations assume binary distribution trees as shown in Figure 1. With appropriate weight and cost settings we can model a limited class of balanced, hierarchical distribution topologies compliant with these assumptions. We are currently working on a more complex and realistic simulation for video caching integrating these techniques in order to receive more detailed results.

We think of a binary tree distribution architecture of depth $d$. We denote a link in the tree by its level $t$ and its index $n$ at this level: $E_n^t$. If we select an arbitrary link, it is called $E^t$. Similarly, cache servers are labeled $N_n^t$ and $N^t$, respectively. For convenience, we consider $N^d$ a client rather than a cache server. We assume the cost per concurrent video stream to be the same $C_t^E$ for each link $E_n^t$ on one level $t$. Also, we assume the hard disk cost for one video to be $C_t^N$ at each cache server $N_n^t$ on one level $t$. The numbers of links and caches at one level $t$are $2^t$

We assume a set of movies $M$. All of these movies $m \in M$ have the same length, measured in time, $L_l$ and the same data rate, but possibly different draw probabilities $P(m)$. In caching scenarios, we assume that each cached movie $m_i$ is stored in all caches of one optimally chosen level $t(m)$.

The necessity to have sufficiently large central servers that are able to handle the number of streams that are concurrently requested imposes a cost $S_0$ for the basic installation of each server, and a cost $S_1$ for each concurrent stream that is supported by a server. Each end-user in the system is watching exactly one video at any time, ie. $\sum_{m \in M} P(m) = 1$. The number of clients is very big compared to the number of different movies and active (cache) servers, the popularity of movies is constant for all clients. This gives us draw probablities being independent of time and hierarchy location, but also gives the problematic postulation of a majority of inactive, thus zeroed cache servers. We enforce this by defining the base server setup cost $S_0$ sufficiently high.

## 2.1 Unicast: No Patching, No Caching

The simplest approach to deliver video is the distribution from a central server via unicast. This allows all kinds of video-on-demand features, but is intense in terms of network as well as server load. We calculate costs for such an approach first.

Since there are no movies stored in the caches there will be no storage costs: $C_t^N = 0, \forall t$

Network costs for each currently running movie: $\sum_{t=1}^{d} C_t^E$, which are the cost of a complete link from the central server to the end-user. As every client is watching exactly one movie at any point of time, the overall network cost for streaming is $\sum_{m \in M}\left(P(m)2^d \sum_{t=1}^{d} C_t^E\right) = 2^d \sum_{t=1}^{d} C_t^E$ The interarrival time is irrelevant in this case, because no streams are shared. With this and a number of clients of $2^d$, the central server approach has an overall cost of $S_0 + 2^d \cdot S_1 + 2^d \sum_{t=1}^{d} C_t^E$

## 2.2 Unicast: No Patching, Caching

### 2.2.1 Network Cost

This implies that networking costs are generated only for the delivery of the movie from the cache server to the clients that are located downstream from this cache server or, in terms of the binary tree, in each subtree with a root node at level $t(i)$. The networking costs for this movie and for this subtree of depth $d$-$t(m)$ can be calculated as in section 2.1:

$$P(m) \cdot 2^{d-t(m)} \sum_{t=t(m)+1}^{d} C_t^E$$

Although the formula concerning the distribution probability of the movies does still apply in this case (the sum of probabilities equals 1), this should not be integrated into this formula, because the optimal level *t(m)* is different for each movie, depending on its probability.

### 2.2.2 Server Cost

Since there are $2^{t(m)}$ cache servers at level *t(m)*, the above networking cost occurs $2^{t(m)}$ times. The cost generated by the movie *m* that is stored at level *t(m)* is then $2^{t(m)} \cdot P(m) \cdot 2^{d-t(m)} \sum_{t=t(m)+1}^{d} C_t^E = P(m) \cdot 2^d \sum_{t=t(m)+1}^{d} C_t^E$

The resulting storage cost for a movie *m* on all cache servers at level *t(m)* is $2^{t(m)} C_{t(m)}^N$

The cost of the capacity needed by this cache server depends on the average number of concurrent streams it has to serve for each movie *m*. This is calculated from the hit probability of the movie and the number of clients that the cache server serves. The setup cost for a needed cache server on level *t* is

$$S_0 + S_1 \cdot 2^{d-t} \cdot \sum_{m \in M} P(m) \cdot \delta(t(m) = t) \qquad \text{where } \delta(p) = \begin{cases} 1, p \text{ is true} \\ 0, p \text{ is false} \end{cases}$$

A cache server has to be set up if its level is the optimal cache level for any movie, thus installation cost for serving clients is as . $\sum_{t=0}^{d-1} \left[ 2^t \delta( \bigcup_{m \in M} (t(m) = t)) \cdot \left( S_0 + S_1 2^{d-t} \sum_{m \in M} P(m) \cdot \delta(t(m) = t) \right) \right]$

As we assume a constant system state, there are no cost to store or stream movies on the root server cached elsewhere.

Simplified and increased by the network cost, this gives the following formula for the overall cost for our model with caching: $\left[ \sum_{t=0}^{d-1} \left( 2^t \cdot \bigcup_{m \in M} \delta_{t(m)}(t) \right) \right] \cdot S_0 + 2^d \cdot S_1 + \sum_{m \in M} \left[ P(m) 2^d \sum_{t=t(m)+1}^{d} C_t^E + 2^{t(m)} C_{t(m)}^N \right]$

## 2.3 Greedy Patching with central server

The simplest form of Patching is Greedy Patching without buffering limits at the clients. Besides the fact that clients will be overly expensive when they are built to buffer complete movies, we have shown in [1] that the optimal restart time in terms of server load depends on *P(m)* and thus, the largest required buffer does not need to hold a complete movie.

However, we assume this kind of Patching to find an approximation for the cost of a distribution system. Assume a binary distribution tree of depth *d*, caching is not applied in this tree.

For each movie *m*, we define $\eta_m = P(m)$ for ease of reuse of the formulas.

### 2.3.1 Server effort

Since this approach is using a central server, $S_0$ is needed only once. The number of streams that need to be served concurrently is also reduced in comparison to the unicast case with a central server. The formula is derived as in section 2.3.2, and yields the setup cost, the basic server cost for multicast streams of *m* and the total cost of unicast patch streams: $S_0 + \left(1-(1-\eta_m)^{2^d}\right) \cdot S_1 + 2^{d-1} S_1$

### 2.3.2 Multicast portion

First, we try to calculate the network load that is generated at each level of the binary tree due to the probability of a joint stream for multiple clients; ie. we want to find a formula for savings of network bandwidth in the upper levels of the binary tree. We assume a random distribution of the clients that share a stream of movie *m* in the overall set of clients. The probability of a network link to be involved in a multicast playout of a specific movie is the probability, that any client below demands that specific movie. This probability is

$2^{d-t}$

which means that at each level *t*, an average of $\left(1-(1-\eta_m)^{2^{d-t}}\right) \cdot 2^t$ links are involved in the same multicast of movie *m*, and a cost that is generated at level t by the multicast streams is $\left(1-(1-\eta_m)^{2^{d-t}}\right) \cdot 2^t \cdot C_t^E$

### 2.3.3 Unicast portion

At the same time, the unicast patches need to be distributed to the clients. These unicast patches require a direct transmission from the central server to the end-user, and this unicast transmission behaves mainly like a regular video transmission according to section 2.1. The major difference is that the length of a unicast patch is less than a full length video transmission rather the length of the unicast patch is on average 1/2 of the patching window, which is in this case the full movie length [1]. Thus, the load of unicast streams at level *t* is in this case: $\frac{1}{2}(\eta_m 2^d C_t^E) = \eta_m \cdot 2^{d-1} \cdot C_t^E$

### 2.3.4 Overall cost

When the unicast and multicast formulas are combined, the overall cost at level *t* is $\left[2^t \cdot \left(1-(1-\eta_m)^{2^{d-t}}\right) + 2^{d-1} \cdot \eta_m\right] \cdot C_t^E$

and the overall cost of distribution of all movies, through the whole tree is the summation

$$S_0 + \left[2^{d-1} + \sum_{m \in M} \left(1-(1-\eta_m)^{2^d}\right)\right] \cdot S_1 + \sum_{m \in M} \sum_{t=1}^{d} \left(2^t - 2^t \cdot (1-\eta_m)^{2^{d-t}} + 2^{d-1} \cdot \eta_m\right) \cdot C_t^E$$

### 2.3.5 Savings Compared To Unicast With Central Server

The average Greedy Patching case is not costlier than the unicast with central server. With the inequality

$$\forall t \le d, \forall m \in M \text{ with } P(m) < 1:$$
$$1-(1-P(m))^{2^{d-t}} \le 1-(1-P(m)) = P(m) \le P(m) 2^{d-t-1}$$

the comparison of the server efforts to section 2.1 gives a possible saving: $\left[2^{d-1} + \sum_{m \in M}\left(1-(1-\eta_m)^{2^d}\right)\right] \cdot S_0 \le 2^d \cdot S_1$

Together with the comparison of network load below this is a first hint to integrate Patching in the delivery system.

$$\sum_{m \in M}\sum_{t=1}^{d}\left[2^t - 2^t \cdot (1-P(m))^{2^{d-t}} + 2^{d-1} \cdot P(m)\right] \cdot C_t^E \le \sum_{m \in M}\sum_{t=1}^{d} C_t^E P(m) 2^d = 2^d \sum_{t=1}^{d} C_t^E$$

## 2.4 Patching with limited buffer and central server

When the restart rate *r(m)* for the multicast stream of a specific movie *m* is increased, i.e., the window size to covered by patch streams is reduced, then the probability that clients receive the same multicast is reduced, but the use of a limited patching window size realistically limits the needed buffer size at the client. As in [1], we assume for simplicity that the multicast transmissions are repeated regularly, and that the length of such a cycle is called the restart time. The restart time here is expressed as a portion of the movie length: $\frac{1}{r(m)}L_1$ The probability for a client to join a specific multicast playout of a specific movie *m* follows as $\eta_m = \frac{1}{r(m)} \cdot P(m)$

### 2.4.1 Server effort

With a patching window of $\frac{1}{r(m)}L_1$, we calculate the average number of concurrent unicast patches to be served according section 2.3.1. Ie. that the number of concurrent unicast streams for *m* is $\frac{1}{2} \cdot 2^d \cdot \frac{P(m)}{r(m)} = 2^{d-1}\eta_m$

This yields the number of concurrent unicast streams that need to be supported by the central server at each time. Unlike for Greedy Patching, *r(m)* is assumed to be optimal but different for different *m*. The server cost for unicast streams is $2^{d-1}S_1 \sum_{m \in M} \eta_m$ which is inverse proportional to the restart rate! The server cost per movie *m* for multicast streams is increasing with the restart rate: $r(m)\left(1-(1-\eta_m)^{2^d}\right) \cdot S_1$

### 2.4.2 Multicast portion

The multicast cost of the distribution system is calculated as in the section 2.3.2, with the redefined $\eta_m$. As *r(m)* copies of the stream can be active at any time, the average load at level *t* is $r(m) \cdot \left(1-(1-\eta_m)^{2^{d-t}}\right) \cdot 2^t \cdot C_t^E$

### 2.4.3 Unicast portion

The computation of the unicast load of the distribution system is the same as in the last section, but with the reduced average length of the unicast patch streams, the values differ. With the redefined value $\eta_m$, however, the formula remains the same as in the previous section, the cost a level *t* is $\eta_m \cdot 2^{d-1} \cdot C_t^E$

### 2.4.4 Overall cost

The combined costs of elements yield the average cost for a distribution system that uses Patching with a central server and movie-dependent window sizes for the delivery of unicast patch streams.

$$S_0 + \left[\sum_{m \in M} \left(2^{d-1} \cdot \eta_m + r(m)\left(1-(1-\eta_m)^{2^d}\right)\right)\right] \cdot S_1$$
$$+ \sum_{m \in M} \left[\sum_{t=1}^{d} \left(C_t^E\left(\eta_m \cdot 2^{d-1} + r(m) \cdot \left(1-(1-\eta_m)^{2^{d-t}}\right) \cdot 2^t\right)\right)\right]$$

## 2.5 Patching with Caching

We assume that for large hierarchies, savings can be increased by combining patching with caching. To verify this, we start with the inner part of the formula from section 2.4.4. We assume that for each movie *m* there is exactly one level *t(m)*, where this movie is cached in all servers. We calculate the server cost for one cache server for *m*. The depth of the distribution sub-tree is *d-t(m)*. Analogous to section 2.4, for this movie, the effort to support streams on the cache server (without basic setup $S_0$ and the movie storage cost $C_t^N$, which can be calculated as in section 2.2.2) and on the network links below this server is given by

$$S_1\left(2^{d-t(m)-1} + r(m)\left(1-(1-\eta_m)^{2^{d-t(m)}}\right)\right) + \sum_{k=1}^{d-t(m)} \left(C_k^E\left(\eta_m \cdot 2^{d-t(m)-1} + r(m) \cdot \left(1-(1-\eta_m)^{2^{d-t(m)-k}}\right) \cdot 2^k\right)\right)$$

This cost occurs once for each server at this level, and that cost, in turn, needs to be calculated once for each movie *m*. This results in an overall cost for Patching with caching of

$$\sum_{m \in M} \left[2^{t(m)} \cdot \left(S_1\left(2^{d-t(m)-1} + r(m)\left(1-(1-\eta_m)^{2^{d-t(m)}}\right)\right) + \sum_{k=1}^{d-t(m)} \left(C_k^E\left(\eta_m \cdot 2^{d-t(m)-1} + r(m) \cdot \left(1-(1-\eta_m)^{2^{d-t(m)-k}}\right) \cdot 2^k\right)\right)\right)\right]$$
$$+ S_0 \cdot \sum_{t=0}^{d-1} \left(2^t \cdot \delta\left(\bigcup_{m \in M} (t = t(m))\right)\right) + \sum_{m \in M} \left(2^{t(m)} C_{t(m)}^N\right)$$

**REFERENCES**

C. Griwodz, M. Liepert, M. Zink, R. Steinmetz. Tune to Lamda Patching. WISP 1999, Atlanta, GA, USA, May 1999