Multimedia Object Placement in Transcoding-enabled Wide-area Storage Systems

Wenyu Qu, Kazuo Goda, and Masaru Kitsuregawa
Institute of Industrial Science
The University of Tokyo
4-6-1 Komaba, Meguro-ku, Tokyo, Japan

With the rapid growth of audio and video applications on the Internet and the popularization of various mobile appliances such as portable notebooks, personal digital assistants, etc, which are divergent of sizes, weight, input/output capabilities, network connectivity, and computing power, how to meet the diverse needs to the same object efficiently and effectively has become an important problem, that raise the research on optimally distributing multiple versions of the same multimedia object in a wide-area storage systems.

To meet the diverse content presentation preferences from different users, the technology of transcoding is used for transforming the multimedia object to proper versions. A full object version has numerous transcoded versions such that different clients' capabilities can be accommodated. Clients' requests for multimedia objects are directed to a storage device in the storage system, usually the nearest one. The requests consist of the name of the multimedia objects and the capability of the client device. When a user's request arrives at a storage device, the storage searches itself for the appropriate multimedia object version.

We model the network as a graph $G=(V,E)$, where $V=(v_1,v_2,\cdots,v_n)$ is the set of nodes, and E is the set of network links. We use $A=(A_j, j=1,2,\cdots,M)$ to denote the set of all versions of a multimedia object and use b_{A_j} to denote the size of A_j. The original version, which can be transcoded to a less detailed object called the transcoded version, is denoted as A_1, whereas the least detailed version, which cannot be transcoded any more, is denoted as A_M. Let $f_{v_i}(A_j)$ denote the access frequency for A_j through node v_i per unit time. The transmission cost for transferring A_j between nodes v_i and v_k is denoted by $c_{v_i,v_k}(A_j)$. If a request goes through multiple network links, the cost is the summation of the cost on all these links. We use B_{v_i} to denote the version stored or to be stored at node v_i. Obviously, we have $B_{v_i}\in A$.

We begin with computing the cost saving (the cost delay saved by placing a version of a multimedia object at a storage device) and the cost loss (the access cost increased by removing a version of a multimedia object at a storage device) of placing a version of a multimedia object at a single storage device. We assume that each storage device has a limited size such that one or more objects may need to be removed from the storage when a version of a multimedia object is stored in. Let $m_{v_i}(A_j)$ be the miss penalty for version A_j with respect to node v_i, which is defined as the additional cost of accessing A_j if B_{v_i} is removed from node v_i; thus, we have $m_{v_j}(A_j)=c_{v_i,v_i^+(A_j)}(A_j)+w(B_{v_i^+(A_j)},A_j)-w(B_{v_i},A_j)$, where $v_i^+(A_j)$ is the nearest higher level node of v_i that stores a more detailed version than A_j (including A_j), $c_{v_i,v_i^+(A_j)}(A_j)$ is the additional transmission cost, $w(B_{v_i^+(A_j)},A_j)$ is the additional transcoding cost, and $w(B_{v_i},A_j)$ is the original transcoding cost. Therefore, the cost saving of storing A_j at v_i, denoted by $s_{v_i}(A_j)$, can be defined as $s_{v_i}(A_j)=\sum_{A_x \in D(A_j)} f_{v_i}(A_x)\cdot m_{v_i}(A_x)$ since removing A_j from v_i will affect those versions that can be transcoded from A_j.

Next, we consider the cost loss of storing a version of a multimedia object at a node。. Let $l_{v_i}(A_j)$ denote the cost loss of storing A_j at v_i. We apply the following greedy heuristic to decide replacement candidates. Note that the normalized cost loss (NCL, i.e., the cost loss introduced by creating one unit of free space) of ejecting A_j is $s_{v_i}(A_j)/b_{A_j}$. The objects in the storage are ordered by their NCLs and are selected sequentially, starting from the object with the smallest NCL, until enough space is created. The cost loss storing a version of a multimedia object at a node is calculated by all the selected candidates.

Let v_1 be the final storage device satisfying the object request, v_n be the client issuing the request, and $v_1,v_2,\cdots,v_{n-1}$ are the intermediate storage devices on the path from v_1 to v_n. The cost saving of storing B_{v_i} at v_i, denoted by $S_{v_i}(B_{v_i})$, is given by $\sum_{A_x \in D(B_{v_i})}\left(f_{v_i}(A_j)-f_{v_i^-(A_x)}(A_x)\right)\cdot m_{v_i}(A_x)$, where $v_i^-(A_x)$ is the nearest lower level node of v_i that stores a less detailed version than A_x. The cost loss of storing B_i at v_i, denoted by $L_{v_i}(B_i)$ $L_{v_i}(B_{i})$, is given by $\sum_{A_x \in D(B_{v_i})} l_{v_i}(A_x)$. For simplicity, we use $1,2,\cdots,n$ to denote in the following analysis, respectively. Let $v_1,v_2,\cdots,v_k$ be a set of k nodes such that $1 \le v_1 \le v_2 \le \cdots \le v_k \le n$. $F(n:v_1,v_2,\cdots,v_k)$, which is the aggregate profit of storing multiple versions of a multimedia object at $v_1,v_2,\cdots,v_k$, is defined as $\sum_1^k\left(S_{v_k}(B_{v_k})-L_{v_k}(B_{v_k})\right)$. If $k=0$, then we define $F(n:\phi)=0$. So the objective is to find k^* and $v_1,v_2,\cdots,v_{k^*}$ that maximizes $F(n:v_1,v_2,\cdots,v_k)$, which is referred to as an n-optimization problem in this paper.

The following theorem shows that an optimal solution for the whole problem must contain optimal solutions to some subproblems.

Theorem 1 Suppose that $v_1,v_2,\cdots,v_l$ is an optimal solution to the n-optimization problem and $u_1,u_2,\cdots,u_l$ is an optimal solution to the v_l-1-optimization problem, then $u_1,u_2,\cdots,u_l,v_l$ is also an optimal solution to the n-optimization problem.

Define F_n^* to be the maximum aggregate profit of $F(n:v_1,v_2,\cdots,v_k)$ obtained by solving the n-optimization problem and I_n the maximum index in the optimal solution. If the optimal solution is an empty set, define $I_n=-1$.

Obviously, we have $I_0=-1$ and $F_0^*=0$. From Theorem 1, we know that if $I_r>0$, $F_{I_r}=F_{I_r-1}+(S_{v_{I_r}}(B_{v_{I_r}})-L_{v_{I_r}}(B_{v_{I_r}}))$. Therefore, we can check all possible locations of $I_r(0\le r\le n)$ $I_{r} and select the one that maximizes $F(r:v_1,v_2,\cdots,v_k)$. So we have

$$\begin{cases} F_0^*=0 \\ F_r^*=\max\limits_{1\le v_i\le r}\{0,F^*_{v_i-1}+(S_{v_{I_r}}(B_{v_{I_r}})-L_{v_{I_r}}(B_{v_{I_r}}))\} \end{cases}$$

and

$$\begin{cases} I_0=-1 \\ I_r=\begin{cases} -1 & if \quad F_r^*=0 \\ v & if \quad F_r^*=F^*_{v_i-1}+(S_{v_{I_r}}(B_{v_{I_r}})-L_{v_{I_r}}(B_{v_{I_r}})) \end{cases} \end{cases}$$

¥begin{eqnarray*}

The original problem can be solved using a dynamic programming-based algorithm with the recurrences above. Theorem ¥ref{theo} ensures the correctness.