# Sizing the Streaming Media Cluster Solution for a Given Workload

Ludmila Cherkasova and Wenting Tang
Hewlett-Packard Laboratories
1501 Page Mill Road, Palo Alto, CA 94303, USA
{lucy.cherkasova, wenting.tang}@hp.com

**Abstract** *The goal of the proposed benchmarking and capacity planning framework is to evaluate the amount of resources needed for processing a given workload while meeting the specified performance requirements. There are two essential components in our capacity planning framework: i) the capacity measurements of different h/w and s/w solutions using a specially designed set of media benchmarks and ii) a media service workload profiler, called MediaProf, which extracts a set of quantitative and qualitative parameters characterizing the service demand. The capacity planning tool matches the requirements of the media service workload profile, SLAs and configuration constraints to produce the best available cost/performance solution. In case of a multi-node configuration, the Capacity Planner performs a cluster sizing evaluation by taking into account the choice of load balancing solution.*

## 1 Introduction

A Utility Data Center (UDC) [11] provides a flexible, cost-effective infrastructure to support the hosting of Internet services. The UDC infrastructure provides a set of new management capabilities for requesting/releasing the system resources to dynamically provision the application demands and their requirements.

In this paper, we consider a scenario where a service provider, supporting a busy media site, faces a necessity to migrate the site to a new, more efficient infrastructure, e.g. to a Utility Data Center infrastructure with its dynamic resource management system. We assume that a service provider collects the media server access logs, reflecting processed client requests and client activities at the site. Thus the problem is to find the appropriate cost/performance configuration for a support of a known media service workload.

Traditionally, network bandwidth or disk system throughput has been the target of optimizations and sizing for streaming media services [7, 1, 8, 9]. Most of designed models deal with the complexity of real-time delivery of variable bit rate content. In our paper, we assume a constant bit rate (CBR) stream model of multimedia delivery. Thus, we do not consider the network bandwidth explicitly in this work: for a given workload, it is easy to determine whether sufficient network bandwidth is available. However, it is more difficult to determine what amount of CPU, memory, and disk resources are needed to process a given workload with specified performance requirements.

Among the *pre-conditions* to a design of a capacity planning tool is the ability to measure and to compare the capacities of different media servers. Currently, there is no a standard benchmark for measuring a media server capacity. In our recent work [3], we proposed a set of benchmarks for measuring the basic capacities of streaming media systems. These benchmarks allow one to derive the scaling rules of server capacity for delivering media files which are: *i)* encoded at different bit rates and *ii)* streamed from memory versus disk.

The capacity planning framework is shown in Figure 1.

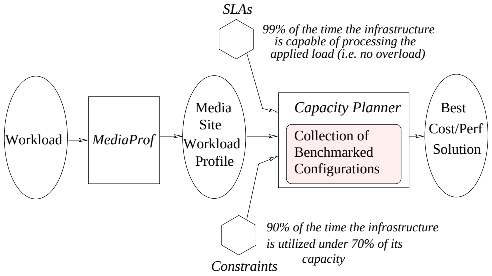



Figure 1: Capacity planning framework.

In order to derive the media site workload profile, which can be used by our *Capacity Planner*, we designed and implemented a new tool, called *MediaProf*, which characterizes the site access profile and its system resource usage in both a quantitative and qualitative way. The *MediaProf*'s analysis is entirely based on media server access logs. Using the media site workload profile, the *Capacity Planner* can evaluate the solution needed for workload processing from the existing set of benchmarked configurations. Since workload measurements of existing media services indicate that client demands are highly variable (the "peak-to-mean" ratio may be an order of magnitude), it might not be economical to overprovision the future system using the past "peak" demand. Thus one can specify the service level requirements (denoted as *SLAs* in Figure 1), which would mean: "Based on the past workload history, find the best cost/performance solution that 99% of the time is capable of processing the applied (past) load". Additionally, a service provider may wish to get a configuration with planned "spare" capacity for future growth (denoted as *Constraints* in Figure 1), which would mean: "Based on the past workload history, find the best cost/performance solution that 90% of the time is utilized under 70% of its available capacity." When a derived recommendation is a multi-node configuration, the *Capacity Planner* performs an additional cluster sizing evaluation by taking into account the choice of load balancing strategy.

Our capacity planning tool provides and promotes a new unified framework for:

- measuring a media server capacity via a set of specially designed basic benchmarks, which enable to derive the resource requirements of a particular media stream via a single value *cost* function from the benchmark measurements;
- deriving a media site workload profile that can be directly mapped to the resource demand profile. The proposed workload profile combines evaluating the number of concurrent connections over time; partitioning them into a set of bit rate groups; and classifying them further by the file access type: memory vs disk.
- combining the capacity requirements of a given media workload, a particular media server configuration and SLA requirements to produce the configuration with required performance characteristics.

Proposed framework provides a flexible and convenient mapping of a service demand (client requests) into the corresponding system resource requirements necessary for accurate capacity planning. The remainder of the paper presents our results in more detail.

## 2 Related Work

It is commonly recognized that multimedia applications can consume significant amount of server and network resources. Much of the research community efforts have been concentrating on improving media server technology, building caching infrastructure support, and designing scalable streaming protocols. Traditionally, network bandwidth or disk system throughput has been the target of optimizations and sizing for streaming media services [7, 1, 8, 9]. Most of designed models deal with the complexity of real-time delivery of variable bit rate content. In media servers, requests from different clients arrive independently. Commercial systems may contain hundreds to thousands of clients. Providing an individual stream for each client may require very high disk bandwidth and may result in significant overprovisioning the disk subsystem in the system. There have been several studies on the more efficient memory usage through the buffer sharing techniques [8, 13] and interval caching schemes [5]. Our work proceeds further in the same direction. We design a high-level model of a traditional memory system with LRU replacement strategy as used in today's commodity systems to reflect and quantify the impact of system level caching in delivering media applications for typical streaming media workloads.

The current trend of outsourcing network services to third parties has brought a set of new challenging problems to the architecture and design of automatic resource management in Internet Data Centers. In [15, 6], the authors promote the necessity of application profiling and adequate system/workload/application models, facilitating a utility service design. Our work follows a similar motivation and proposes the corresponding models for evaluating performance and sizing the streaming media cluster solutions.

## 3 Media Server Capacity Equations

Commercial media servers are characterized by the number of concurrent streams a server can support without loosing a stream quality. In paper [3], two basic benchmarks were introduced that can establish the scaling rules for server capacity when multiple media streams are encoded at different bit rates:

- *Single File Benchmark*: measuring a media server capacity when all the clients in the test are accessing the same file, and
- *Unique Files Benchmark*: measuring a media server capacity when each client in the test is accessing a different file.

Each of these benchmarks consists of a set of sub-benchmarks with media content encoded at a different bit rate (in our study, we used six bit rates representing the typical Internet audience: 28 Kb/s, 56 Kb/s, 112 Kb/s, 256 Kb/s, 350 Kb/s, and 500 Kb/s. Clearly, the set of benchmarked encoding bit rates can be customized according to targeted workload profile). Using an experimental testbed and a proposed set of basic benchmarks, we measured capacity and scaling rules of a media server running RealServer 8.0 from RealNetworks. The configuration and the system parameters of our experimental setup are specially chosen to avoid some trivial bottlenecks when delivering multimedia applications such as limiting I/O bandwidth between the server and the storage system, or limiting network bandwidth between the server and the clients. The measurement results show that the scaling rules for server capacity when multiple media streams are encoded at different bit rates are non-linear. For example, the difference between the highest and lowest bit rate of media streams used in our experiments is 18 times. However, the difference in maximum number of concurrent streams a server is capable of supporting for corresponding bit rates is only around 9 times for a *Single File Benchmark*, and 10 times for a *Unique Files Benchmark*. The media server performance is 3 times higher (for some disk/file subsystem up to 7 times higher) under the *Single File Benchmark* than under the *Unique Files Benchmark*.

Using a set of basic benchmark measurements, we derive a cost function which defines a *fraction* of system resources needed to support a particular media stream depending on the stream bit rate and type of access (memory file access or disk file access):

- $cost^{disk}_{X_i}$ - a value of cost function for a stream with disk access to a file encoded at $X_i$ Kb/s. If we define the media server capacity being equal to 1, the cost function is computed as $cost^{disk}_{X_i} = 1/N^{Unique}_{X_i}$, where $N^{unique}_{X_i}$ - the maximum measured server capacity in concurrent streams under the *Unique File Benchmark* for $X_i$ Kb/s encoding,
- $cost^{memory}_{X_i}$ - a value of cost function for a stream with memory access to a file encoded at $X_i$ Kb/s. Let $N^{single}_{X_i}$ - the maximum measured server capacity in concurrent streams under the *Single File Benchmark* for a file encoded at $X_i$ Kb/s. Then

the cost function is computed as $cost_{X_i}^{memory} = (N_{X_i}^{unique} - 1)/(N_{X_i}^{unique} \times (N_{X_i}^{single} - 1))$.

Let $W$ be the current workload processed by a media server, where

- $X_w = X_1, ...X_{k_w}$ - a set of distinct encoding bit rates of the files appearing in $W$,
- $N_{X_{w_i}}^{memory}$- a number of streams having a memory access type for a subset of files encoded at $X_{w_i}$ Kb/s,
- $N_{X_{w_i}}^{disk}$ - a number of streams having a disk access type for a subset of files encoded at $X_{w_i}$ Kb/s.

Then the service demand to a media server under workload $W$ can be computed by the following equation

$$Demand = \sum_{i=1}^{k_w} N_{X_{w_i}}^{memory} \times cost_{X_{w_i}}^{memory} + \sum_{i=1}^{k_w} N_{X_{w_i}}^{disk} \times cost_{X_{w_i}}^{disk} \quad (1)$$

If $Demand \leq 1$ then the media server operates within its capacity, and the difference $1 - Demand$ defines the amount of available server capacity. We validated this performance model by comparing the predicted (computed) and measured media server capacities for a set of different synthetic workloads (with statically defined request mix). The measured server capacity matches the expected server capacity very well for studied workloads (with the error 1%-8%).

If $Demand \geq 1$ then the media server is overloaded and its capacity is exceeded. For example, when computed service demand is $Demand = 4.5$ it indicates that considered media traffic requires *5* nodes (of the corresponding media server configuration) for the workload support. We use the *capacity equation* (1) in evaluating the capacity requirements of considered media workload on a particular media server configuration. Using a media site traffic profile (described in Section 4) and the cost functions of different media server configurations, we compute $Demand$ using the *capacity equation* (1), and then compare computed results to choose the best configuration.

## 4 Workload Profiler *MediaProf*

Media site profile, based on the past workload history, is a critical component in decision making about the future supporting infrastructure. We designed and implemented a media workload analysis tool, called *MediaProf*, which reflects the access traffic profile for capacity planning goal by extracting the following characteristics:

• *Evaluating the number of simultaneous (concurrent) connections over time.*

first of all, *MediaProf* extracts the number of *concurrent* connections over time and the corresponding *bandwidth requirements*. In our capacity planning tool, the number of concurrent connections is averaged and reported at 1 min granularity.

As an example, we use the access log from one of the media servers supporting the HP Corporate Media Site. The HPC log covers almost one year duration. Figures 3 a),b) show the number of concurrent client sessions and the maximum bandwidth requirements of the site over the considered workload duration. (Note that Y axes use a **logscale** for both figures).

• *Classifying the simultaneous connections into the encoding bit rate bins.*

The information on the simultaneous connections over time is still not sufficient for accurate capacity planning, because the amount of system resources ("cost") and the server bandwidth needed to support a particular client request depend on the file encoding bit rate. Thus, *MediaProf* further classifies the simultaneous connections into the encoding bit rate bins.

• *Classifying the simultaneous connections by the file access type: memory vs disk.*

The request processing cost within the same encoding bit rate group additionally depends on file access type: memory file access or disk file access. In order to assign a *cost* to a media request from the access log, we need to evaluate whether a request will be streaming data from memory or will be accessing data from disk. Note, that memory access does not assume or require that the whole file resides in memory: if there is a sequence of accesses to the same file, issued closely to each other on a time scale, then the first access may read a file from disk, while the subsequent requests may be accessing the corresponding file prefix from memory.

Taking into account the real-time nature of streaming media applications and the sequential access to file content, we developed a *segment-based memory model* reflecting data stored in memory as a result of media file accesses. This model closely approximates the media server behavior when the media server operates over a native OS file buffer cache with LRU replacement policy.

The basic idea of computing the request access type is as follows. Let $Size^{mem}$ be the size of memory in bytes [1]. For each request $r$ in the media server access log, we have the information about the media file requested by $r$, the duration of $r$ in seconds, the encoding bit rate of the media file requested by $r$, the time $t$ when a stream corresponding to request $r$ is started (we use $r(t)$ to reflect it), and the time when a stream initiated by request $r$ is terminated.

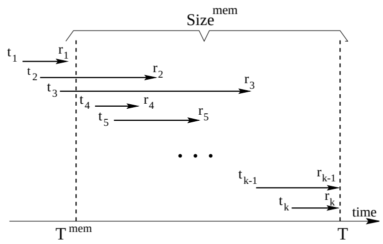



Figure 2: Memory state computation example.

Let $r_1(t_1), r_2(t_2), ..., r_k(t_k)$ be a recorded sequence of requests to a media server. Given the current time $T$ and request $r(T)$ to media file $f$, we compute some past time $T^{mem}$ such that the sum of the bytes stored in memory between $T^{mem}$ and $T$ is equal to $Size^{mem}$ as shown in Figure 2. This way, the files' segments streamed by the

[1] Here, a memory size means an estimate of what the system may use for a file buffer cache.

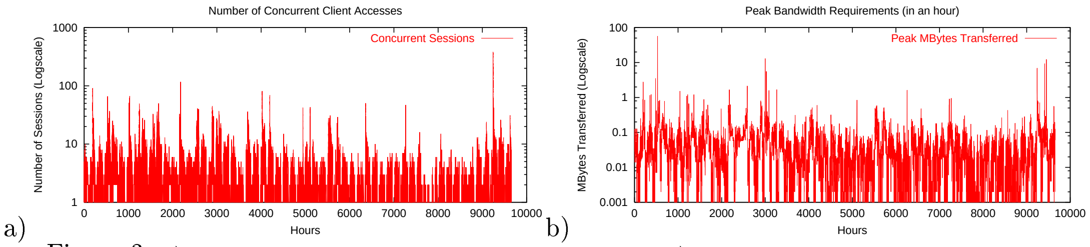



Figure 3: a) Number of concurrent client sessions over time, b) peak MBytes transferred over time.

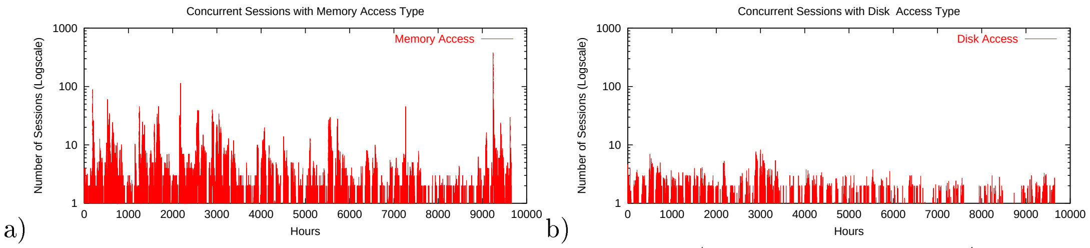



Figure 4: HPC log: concurrent client sessions from memory vs disk over time (bit rate group 112-256 Kb/s, $Size^{mem} = 1\ GB$.)

media server between times $T^{mem}$ and $T$ will be in memory. In such a way, we can identify whether request $r$ will stream file $f$ (or some portion of it) from memory.

Figure 4 shows the classification of client requests (HPC log) into memory accesses and disk accesses for encoding bit rate group of 112-256 Kb/s and a memory size of 1 GB. The results show that very large fraction of requests in this bit rate group can be served from memory. In particular, practically all the traffic bursts (spikes) can be served from memory as Figure 4 a) shows. Since a media server capacity is 3-7 times higher when media streams are delivered from memory versus from disk, such a qualitative media traffic classification and analysis directly translates in significant configuration savings.

In summary, a workload profiler *MediaProf* is processing the media server access logs by

- evaluating the number of concurrent connections at each moment of time;
- partitioning the concurrent connections into a predefined set of bit rate groups;
- classifying the concurrent connections by the file access type: memory vs disk.

Table 1 shows the snapshot of the *media workload profile* produced by *MediaProf*.

| Time Stamp | Concurrent Sessions | $< 56\ Kb/s$ | | $56 - 112\ Kb/s$ | | $> 112\ Kb/s$ | |
|---|---|---|---|---|---|---|---|
| | | Disk | Memory | Disk | Memory | Disk | Memory |
| ... | ... | ... | ... | ... | ... | ... | ... |
| $t_{i-1}$ | 100 | 2 | 0 | 5 | 2 | 85 | 6 |
| $t_i$ | 104 | 2 | 0 | 5 | 2 | 89 | 6 |
| $t_{i+1}$ | 103 | 1 | 0 | 5 | 2 | 89 | 6 |
| ... | ... | ... | ... | ... | ... | ... | ... |

Table 1: Output of *MediaProf*: media site workload profile.

First two columns reflect the concurrent connections over time. The time stamps in the first column show when the media server state changes, e.g. *i)* the media server accepts a new client request (or multiple new requests) or *ii)* some active media sessions are terminated by the clients. The other columns show how these concurrent connections are classified into encoding bit rate groups with further classification by the type of access: disk or memory file access.

*MediaProf* produces the media workload profiles $MP_1$, $MP_2$, ..., $MP_k$ for specified memory sizes $M_1$, $M_2$, ..., $M_k$ as shown in Figure 6.

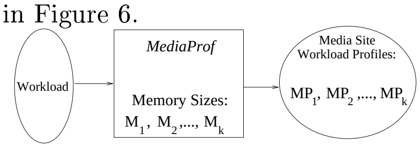



Figure 6: Media site workload profiling.

This way, *MediaProf* enables the evaluation of performance benefits of systems with different memory sizes when processing a particular workload.

## 5 Overall Capacity Planning Process

The overall capacity planning process is shown in Figure 5. There are several logical steps in the capacity planning procedure:

- *Computing the media site workload profile.*

Using the available media site workload, *MediaProf* computes a set of media site workload profiles for different memory sizes of interest. During the initial analysis, the *Dispatcher* component that imitates the load balancing strategy for the cluster of $N$ nodes, has $N = 1$.

- *Computing the service demand profile.*

The *Capacity Planner* module has a collection of benchmarked configurations. The *Capacity Planner* takes the media site workload profile (as it is shown in Table 1) and computes the corresponding service demands according to formula (1) from Section 3, where the cost functions represent a particular media server configuration from the set of benchmarked configurations.

Computed *service demand profile* is the list of pairs. The first element of a pair represents a time duration, e.g. 300 sec. The second element reflects the service demand over this duration, e.g. 4.5 means that the considered workload requires 4.5 servers during 300 sec.

The service demand profile is ordered (in descending order) by the second element. Pairs with the same second element are aggregated. Thus, the top element in the service demand profile represents the peak demand for considered workload and the time duration of this

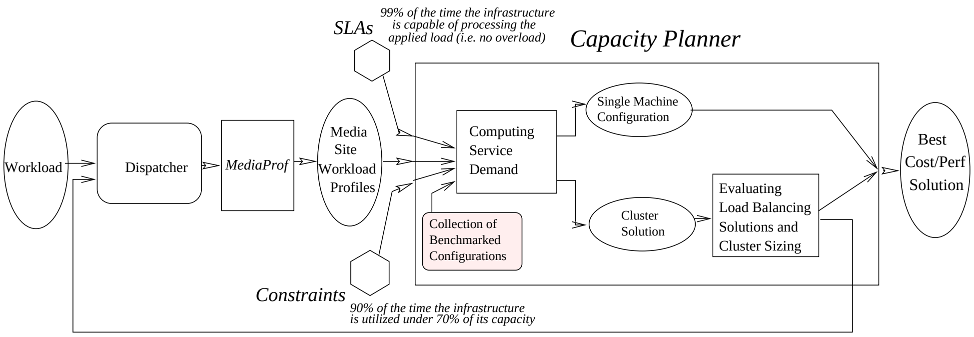



Figure 5: Capacity planning process.

peak. Equivalently, we use a *cumulative density function (CDF)* for plotting a service demand that is normalized over time.

• *Combining the service demand requirements, the SLAs, and the configuration constraints.*
Since workload measurements of existing media services indicate that client demands are highly variable (the "peak-to-mean" ratio may be an order of magnitude), it may not be cost-effective to overprovision the system for the peak load demand. In this case, one can specify the service level requirements (denoted as *SLAs* in Figure 5), which would mean: "Based on the past workload history, find the best cost/performance solution that 99% of the time is capable of processing the applied (past) load". Using the computed service demand profile, the *Capacity Planner* finds the maximum demand corresponding to the 99-th percentile of the site's service demands over time. Let us denote this demand as $Demand_{SLA}$.

Additionally, a service provider may wish to define a configuration with planned "spare" capacity for future growth (denoted as *Constraints* in Figure 5), which would mean: "Based on the past workload history, find the best cost/performance solution that 90% of the time is utilized under 70% of its available capacity." Thus using the computed service demand profile, the *Capacity Planner* finds the maximum demand corresponding to the 90-th percentile of the site's service demands over time. For example, if the service demand corresponding to the 90-th percentile is 3.5 then the requirement for a configuration utilized under 70% of its available capacity is 3.5/0.7=5. Let us denote this demand as $Demand_{Constraints}$.

Thus, the requirement for a desirable configuration is:

$$Demand_{overall} = max(Demand_{SLA}, Demand_{Constraints})$$

rounded up to the closest integer.

When a derived configuration is a single machine configuration, it is directly included in the final set of configurations which satisfy the specified performance requirements. There could be multiple configurations satisfying the specified requirements. Taking into consideration the price information of the corresponding configurations, the best cost/performance solution can be chosen.

• *Cluster sizing with an appropriate load balancing solution.* When a derived recommendation is a multi-node configuration, the *Capacity Planner* performs an additional cluster sizing evaluation by taking into account the choice of load balancing solution as described in the next section.

# 6 Evaluating Load Balancing Solutions and Cluster Sizing

Media server clusters are used to create scalable and highly available solutions. We assume that each media server in a cluster has access to all the media content. Therefore, any server can satisfy any client request. A cluster of $N$ nodes represents $N$ times greater processing power, and at the same time, it has $N$ times larger combined memory. Thus, for an accurate sizing of a cluster solution, we need to take into account both: its increased processing power and its increased memory size.

Traditional load balancing solution for a media server cluster, such as *Round-Robin (RR)*, tries to distribute the requests uniformly to all the machines. However, this may adversely affect an efficient memory usage because the frequently accessed content is replicated across the memories of all the machines. With this approach, a cluster having $N$ times bigger memory (which is the combined memory of $N$ nodes) might effectively utilize almost the same memory as one node. This observation led researchers to a design of "locality-aware" balancing strategies like *LARD* [12] and *DALA* [10], which aim to avoid the unnecessary file replication to improve the overall system performance.

In our capacity planning tool, we provide a cluster sizing evaluation for two load balancing solutions:

- *Round-Robin* strategy;
- *LARD* strategy: initially, the first access to a file is assigned to a random node in a cluster, while the subsequent requests to the file are sent to the same, already assigned node.

Let the outcome of the first iteration of the *Capacity Planner* for the original media cite workload be the capacity requirement of $k$ nodes of a particular media server. Then the the capacity planning procedure goes through the following sequence of steps to re-evaluate the identified cluster solution:

- Partitioning the original media site workload $W$ into $k$ sub-workloads $W_1, W_2, .., W_k$ using the *Dispatcher* employing a corresponding load balancing strategy.

- Computing the media workload profile for each of sub-workloads $W_1, W_2, .., W_k$ using *MediaProf*.
- Merging the computed sub- workload profiles in the overall media site workload profile by using the time stamps of individual sub-workloads profiles.
- Computing the overall service demand profile. [2]
- Combining the service demand requirements, the *SLAs*, and the *Configuration Constraints*.
  - If the outcome of this step is still the capacity requirements of $k$ nodes then the cluster sizing is done correctly and the capacity planning process for a considered cluster configuration is completed.
  - If the computed capacity requirements are $l$ nodes ($l \neq k$) then the capacity planning process is repeated for the cluster configuration of $l$ nodes.

In order to demonstrate the necessity and importance of described iterative process for accurate cluster sizing, let us consider the following example.

**Example.**

For workload generation, we use the publicly available, synthetic media workload generator *MediSyn* [14]. In our example, we explore a synthetic media workload $W_{syn}$ that closely imitates the parameters of real enterprise media server workloads [2].

Video duration distribution in $W_{syn}$ can be briefly summarized via following six classes: 20% of the files represent short videos 0-2min, 10% of the videos are 2-5min, 13% of the videos are 5-10min, 23% are 10-30min, 21% are 30-60min, and 13% of the videos are longer than 60 min.

The file bit rates are defined by the following discrete distribution: 5% of the files are encoded at 56Kb/s, 20% - at 112Kb/s, 50% - at 256Kb/s, 25% at 500Kb/s.

Request arrivals are modeled by a Poisson process. In our workload, on average, there is a new request arrival in 1 *sec* intervals.

The file popularity in $W_{syn}$ is defined by a Zipf-like distribution with $\alpha = 1.34$. In summary, $W_{syn}$ has a fileset with 800 files (with overall storage requirements of 41 GB), where 90% of the requests target 10% of the files in $W_{syn}$. Correspondingly, 10% of the most popular files in $W_{syn}$ have an overall combined size of 3.8 GB.

Let the capacity planning task be to find the appropriate media system configuration satisfying the following performance requirements while processing $W_{syn}$:

- *SLAs*: for 99% of the time, a system configuration can process a given workload without overload.
- *Configuration Constraints*: for 90% of the time, a system configuration is utilized under 70% of its available capacity.

Let the benchmarked capacity of the media server of interest be defined as shown in Table 2. Let us denote this media server type as $\hat{S}$.

[2] In case of the LARD load balancing strategy, we additionally compute the service demand profile for each of sub-workloads $W_1, W_2, .., W_k$ in order to provide a warning when a locality-aware request distribution leads to a significantly skewed load for different nodes.

| Benchmark | Server Capacity in Concurrent Streams for Files Encoded at a Given Bit Rate | | | |
|---|---|---|---|---|
| | $56\ Kb/s$ | $112\ Kb/s$ | $256\ Kb/s$ | $500\ Kb/s$ |
| Single File Benchmark | 600 | 400 | 200 | 100 |
| Unique Files Benchmark | 125 | 80 | 40 | 20 |

Table 2: Benchmarked media server capacity.

The server capacity scaling rules for different encoding bit rates provided above are similar to those measured using the experimental testbed described in Section 3. We use $cost^{disk}_{X_i} / cost^{memory}_{X_i} = 5$, i.e. the cost of disk access for files encoded at bit rate $X_i$ is 5 times higher than the cost of the corresponding memory access.

Let the memory sizes of interest be 0.5 GB and 1 GB [3]. To distinguish the media server configuration with memory of $Size^{mem}$, we use a denotation $\hat{S}(Size^{mem})$.

Figure 7 shows the computed CDF of capacity requirements for processing the workload $W_{syn}$ on the media server $\hat{S}$ with two different memory sizes as a result of the first iteration of capacity planning process.

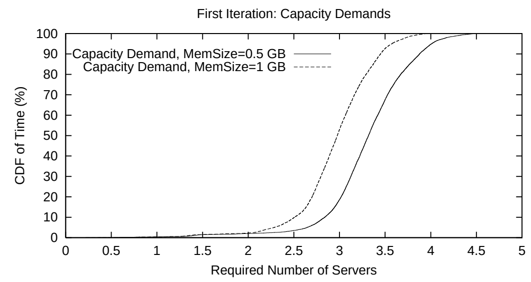



Figure 7: First iteration of capacity planning process: capacity demands for $W_{syn}$.

Table 3 summarizes the capacity demands for processing workload $W_{syn}$ by taking into account different performance requirements. Thus, the maximum capacity required for given workload processing on $\hat{S}(0.5\ GB)$ configuration is 4.6 servers. If we take into account the *SLA* requirements (i.e. capacity demands for 99-th percentile) then the required capacity is 4.3 servers. In order to satisfy the *Configuration Constraints* (i.e. solution should have a capacity that 90% of the time is utilized under 70%), the required capacity should be $3.9/0.7 = 5.6$. after combining the *SLAs* and *Configuration Constraints* requirements and rounding them up to the nearest integer, the capacity requirement for processing $W_{syn}$ is 6 nodes of the media server $\hat{S}(0.5\ GB)$.

| Media Server Type | Capacity Demands for Processing $W_{syn}$ | | | |
|---|---|---|---|---|
| | Max | *SLAs* | *Configuration Constraints* | Combined (rounded up) |
| $\hat{S}(0.5\ GB)$ | 4.6 | 4.3 | 3.9 / 0.7 = 5.6 | 6 |
| $\hat{S}(1\ GB)$ | 4 | 3.8 | 3.4 / 0.7 = 4.9 | 5 |

Table 3: Summary of capacity demands for $W_{syn}$ during the first iteration.

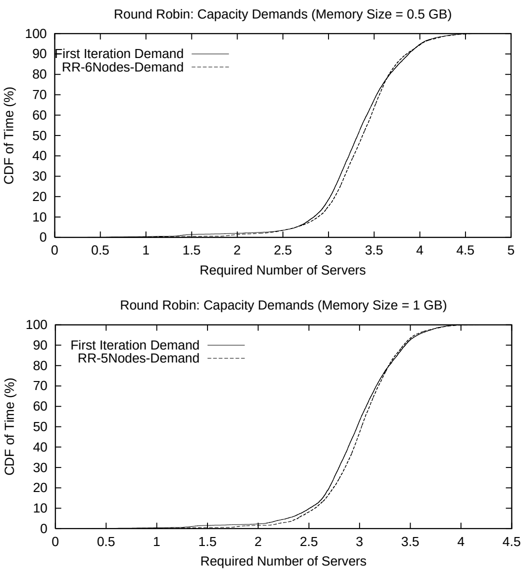
Following the similar computation process, the capacity requirement for processing $W_{syn}$ is 5 nodes of the media server $\hat{S}(1\ GB)$.

[3] Here, a memory size means an estimate of what the system may use for a file buffer cache.

Since the initial classification of client requests into memory/disk accesses as well as the subsequent computation of service demand profile is done on a base of a "single node" memory model, and since the computed capacity requirement for processing $W_{syn}$ on the media server $\hat{S}$ is a multi-node solution, we need to re-evaluate workload performance on the recommended solution by taking into account the specific load-balancing solution and the impact of increased memory in a cluster (due to multiple nodes).

Let us first analyze the outcome for *Round-Robin* load balancing strategy and $\hat{S}(0.5\ GB)$ configuration. During the second iteration, our capacity planning tool divides the original workload $W_{syn}$ into *6* sub-workloads $W^1_{syn}, W^2_{syn}, .., W^6_{syn}$ in a round-robin manner. For each sub-workload $W^i_{syn}$ $(1 \le i \le 6)$, *MediaProf* classifies the client requests into memory/disk accesses on a base of an individual "single node" memory model. Then the computed media sub-workload profiles are merged and the overall service demand profile is recomputed.

Figure 8 a) shows the CDF of capacity requirements for a given synthetic workload $W_{syn}$ and *6* nodes of media server $\hat{S}(0.5\ GB)$ using the *Round-Robin* load balancing strategy. Figure 8 b) shows the corresponding results for *5* nodes of media server $\hat{S}(1\ GB)$.



a) b)

Figure 8: Round-Robin load balancing strategy: capacity demands for a) Memory Size = 0.5 GB b) Memory Size = 1 GB.

The outcome of the second iteration, confirms the predicted workload performance on recommended solutions when the *Round-Robin* load balancing strategy is used. Since *RR* strategy distributes the requests uniformly to all the machines, this prohibits an efficient memory usage in a cluster because popular content is replicated across the memories (file buffer caches) of all the machines. As Figures 8 a), b) show, with *RR* strategy, the increased memory (due to combined memory of multiple nodes in the cluster) does not provide the additional performance benefits.

Now, let us analyze the outcome for the *LARD* load balancing strategy and $\hat{S}(0.5\ GB)$ configuration. During the *second iteration*, the dispatcher in our capacity planning tool divides the original workload $W_{syn}$ into *6* sub-workloads $W^1_{syn}, W^2_{syn}, .., W^6_{syn}$ according to the *LARD* strategy. For each sub-workload $W^i_{syn}$ $(1 \le i \le 6)$, *MediaProf* classifies the client requests into memory/disk accesses on a base of an individual "single node" memory model. Then the computed media sub-workload profiles are merged and the overall service demand profile is recomputed.

The first line in Table 4 summarizes the capacity demands for processing workload $W_{syn}$ using *6* nodes of media server $\hat{S}(0.5\ GB)$ and applying *LARD* as a load balancing strategy. As one can see, increased memory (due to *6* nodes in the cluster) helps to improve system performance, and the outcome recommendation of the second iteration is *4*-node configuration.

| Media Server Type | Nodes | Number of Nodes for Processing $W_{syn}$ | | |
|---|---|---|---|---|
| | | *SLAs* | *Configuration Constraints* | Combined (rounded up) |
| $\hat{S}(0.5\ GB)$ | 6 | 3 | 2.8 / 0.7 = 4 | 4 |
| $\hat{S}(0.5\ GB)$ | 4 | 3.4 | 3 / 0.7 = 4.3 | 5 |
| $\hat{S}(0.5\ GB)$ | 5 | 3.2 | 2.9 / 0.7 = 4.1 | 5 |
| $\hat{S}(1\ GB)$ | 5 | 2.8 | 2.5 / 0.7 = 3.6 | 4 |
| $\hat{S}(1\ GB)$ | 4 | 2.8 | 2.6 / 0.7 = 3.7 | 4 |

Table 4: Summary of capacity demands for $W_{syn}$ and different size cluster configurations using LARD as a load balancing strategy.

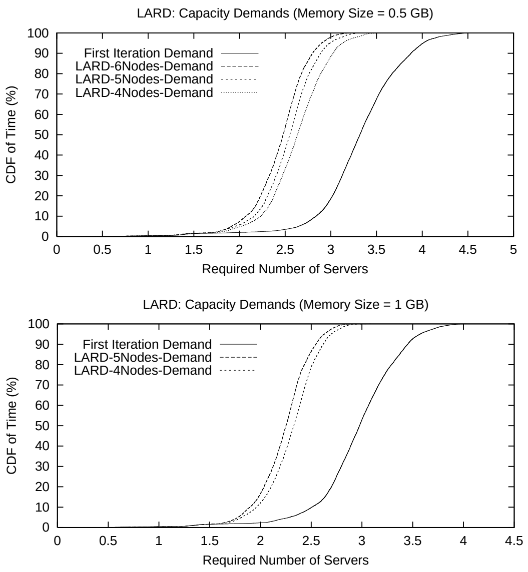
Then the capacity planning process performs the *third iteration* to re-evaluate workload performance on the recommended solution of *4* nodes. However, the capacity of combined memory and computing power of *4*-node configuration does not satisfy the specified performance requirements as shown by the outcome of the third iteration in Table 4: the computed capacity requirement is *5*-node solution. Finally, the *fourth iteration* (summarized by the third line in Table 4), re-evaluates workload performance on the *5*-node configuration and confirms that *5*-node solution with *LARD* strategy satisfies the specified performance requirements.

The capacity planning process for $\hat{S}(1\ GB)$ media server and *LARD* load balancing strategy evaluates *5*-node configuration (which is the outcome of the first iteration as shown in Table 3. It converges after the third iteration with recommended configuration of *4 nodes* as shown in Table 4.

Figure 9 a) shows the CDF of capacity requirements for a given synthetic workload $W_{syn}$ and *4-6* nodes of media server $\hat{S}(0.5\ GB)$ using *LARD* as a load balancing strategy. Figure 9 b) shows corresponding results for *5-6* nodes of media server $\hat{S}(1\ GB)$.

Since the requests for the same file are sent by *LARD* to the same node in the cluster, it helps to avoid the unnecessary file replication and to utilize the overall cluster memory more efficiently, and as a result to improve the overall system performance.

Table 5 summarizes configuration choices for processing workload $W_{syn}$ when using media server $\hat{S}$ with dif-




a)

b)

Figure 9: LARD load balancing strategy: capacity demands for a) Memory Size = 0.5 GB b) Memory Size = 1 GB.

ferent memory sizes (0.5 GB and 1 GB) and employing different request distribution strategies in the cluster.

| Media Server Type | Capacity Demands for Processing $W_{syn}$ with Different Load Balancing Strategies | |
|---|---|---|
| | *Round-Robin* | *LARD* |
| $\hat{S}(0.5\ GB)$ | 6 | 5 |
| $\hat{S}(1\ GB)$ | 5 | 4 |

Table 5: Summary of capacity demands for $W_{syn}$ and different load balancing strategies.

Overall, the cluster configuration with *LARD* as a balancing strategy outperforms the cluster configuration with *RR* strategy for processing workload $W_{syn}$. By taking into consideration the price information of the corresponding configurations, a service provider may choose whether to deploy *5* nodes of media server $\hat{S}$ with 0.5 *GB* of memory for file buffer cache or rather to deploy *4* nodes with upgraded memory of 1 *GB*.

The conclusion, derived in the example, is workload dependent. For the media workload with a small "footprint" of hot files that entirely fit in memory, the RR strategy may provide performance comparable with the LARD strategy. The proposed capacity planning tool performs sizing of the media server cluster and choice of the load balancing solution to satisfy the desirable performance requirements of a given workload.

## 7 Conclusion and Future Work

In this paper, we outlined a new capacity planning framework for evaluating the capacity requirements of considered media workload on a particular media server configuration. The proposed framework provides a flexible and convenient mapping of a service demand (client requests) into the corresponding system resource requirements necessary for accurate capacity planning. In case of a multi-node configuration, the *Capacity Planner* performs an additional cluster sizing evaluation by taking into account the load balancing strategy in the cluster.

In summary, our capacity planning tool is promoting a new unified framework for:

- measuring media server capacity via a set of basic benchmarks;
- deriving the resource requirements (a *cost*) of a media stream from the basic benchmark measurements;
- estimating the service capacity demands from the proposed media site workload profile and the corresponding cost functions of the benchmarked configurations.

In the future, we plan to incorporate in our tool a set of additional load balancing and content placement strategies for media server clusters.

## References


[1] E. Biersack, F. Thiesse. Statistical Admission Control in Video Servers with Constant Data Length Retrieval of VBR Streams. Proc. of the 3d Intl. Conference on Multimedia Modeling, France, 1996.

[2] L. Cherkasova, M. Gupta. Characterizing Locality, Evolution, and Life Span of Accesses in Enterprise Media Server Workloads. Proc. of ACM NOSSDAV, 2002.

[3] L. Cherkasova, L. Staley. Measuring the Capacity of a Streaming Media Server in a Utility Data Center Environment. Proc. of 10th ACM Multimedia, 2002.

[4] L. Cherkasova, L. Staley. Building a Performance Model of Streaming Media Applications in Utility Data Center Environment. Proc. of ACM/IEEE Conference on Cluster Computing and the Grid (CCGrid), May, 2003.

[5] A. Dan, D. Sitaram. A Generalized Interval Caching Policy for Mixed Interactive and Long Video Workloads . Proc. of IST/SPIE Multimedia Computing and Networking, 1996.

[6] R. Doyle, J. Chase, O. Asad, W. Jen, A. Vahdat. Model-Based Resource Provisioning in a Web Service Utility. Proc. of the USENIX Symposium on Internet Technologies and Systems (USITS), 2003.

[7] J. Dengler, C. Bernhardt, E. Biersack. Deterministic Admission Control Strategies in Video Servers with Variable Bit Rate. Proc. of European Workshop on Interactive Distributed Multimedia Systems and Services (IDMS), Germany, 1996.

[8] A. Dan, D. Dias R. Mukherjee, D. Sitaram, R. Tewari. Buffering and Caching in Large-Scale Video Servers. Proc. of COMPCON, 1995.

[9] Y. Fu, A. Vahdat. SLA-Based Distributed Resource Allocation for Streaming Hosting Systems. Proc. of the 7th Intl Workshop on Web Content Caching and Distribution (WCW-7), 2002.

[10] Z. Ge, P. Ji, P. Shenoy. A Demand Adaptive and Locality Aware (DALA) Streaming Media Cluster Architecture. Proc. of ACM NOSSDAV, 2002.

[11] Utility Data Center: HP's First Proof Point for Service-Centric Computing. IDC white paper. `http://www.idc.com`

[12] V.Pai, M.Aron, G.Banga, M.Svendsen, P.Drushel, W. Zwaenepoel, E.Nahum: Locality-Aware Request Distribution in Cluster-Based Network Servers. Proc. of the 8th Intl. Conf. on Architectural Support for Programming Languages and Operating Systems (ASPLOS VIII), ACM SIGPLAN, 1998.

[13] B. Ozden, R. Rastogi, A. Silberschatz. Buffer Replacement Algorithms for Multimedia Databases. In S. M. Chung, editor, Multimedia Information Storage and Management, chapter 7. Kluwer Academic Publishers, 1996.

[14] W. Tang, Y. Fu, L. Cherkasova, A. Vahdat. MediSyn: A Synthetic Streaming Media Service Workload Generator. Proc. of ACM NOSSDAV, 2003.

[15] B. Urgaonkar, P. Shenoy, T. Roscoe. Resource Overbooking and Application Profiling in Shared Hosting Platforms. Proc. of 5th Symposium on Operating Systems Design and Implementation (OSDI), 2002.