# An SLA-Oriented Capacity Planning Tool for Streaming Media Services

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

**Abstract** *The main problem we address in this paper is how to map the requirements of a known media service workload into the corresponding system resource requirements and to accurately size the required system. In this paper, we propose a new capacity planning framework for evaluating the resources needed for processing a given streaming media workload with specified performance requirements. In our tool, the service provider specifies the desirable system performance by stating two types of requirements in a Service Level Agreement (SLA): i) basic capacity requirements that define the percentage of time the configuration is capable of processing the workload without performance degradation while satisfying the specified system utilization; and ii) performability requirements that define the acceptable degradation of service performance during the remaining, non-compliant time and in case of node failures. Using a set of specially benchmarked media server configurations, the capacity planning tool matches the overall capacity requirements of the media service workload profile with the specified SLAs to identify the number of nodes necessary to support the required service performance.*

## 1 Introduction

The delivery of continuous media from a central server complex to a large number of (geographically distributed) clients is a challenging and resource intensive task. The trend toward media content hosting is seeing a significant growth as more rich media is used in the enterprise environment. In this paper, we consider a scenario where a service provider, supporting a busy media site, needs to migrate the site to a new, more efficient infrastructure.

Traditionally, network bandwidth or disk system throughput has been the target of optimization and sizing for streaming media services. In our paper, we assume that site' media content is encoded at constant bit rates (CBR). Thus, for a given workload, it is easy to determine what network bandwidth is required. 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.

Earlier analysis [3] shows that emerging streaming workloads (e.g., for enterprise settings, news servers, sports events, and music clips) exhibit a high degree of temporal and spatial reference locality: i.e. a high percentage of requests access a small subset of media files. Moreover, many clients do not finish the playback of a full video/audio clip and 50%-60% of the accesses last less than 2 minutes [3, 1]. Therefore, most of the accesses to popular media objects can be served from memory, even when a media server relies on traditional file system and memory support and does not have additional application level caching. Thus, the locality available in a particular workload has a significant impact on the behavior of the system, and the capacity planning tool should account for the impact of the server's main memory file buffer cache when evaluating and sizing the required configuration.

The ability to plan and operate at the most cost effective capacity is a critical competitive advantage. 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. In our tool, the service provider specifies the desirable system performance by stating two types of requirements in an *Service Level Agreement (SLA)*: *i) basic capacity requirements* that define the percentage of time the configuration is capable of processing the applied load without performance degradation while satisfying the specified system utilization; and *ii) performability requirements* that define the acceptable degradation of service performance during the remaining, non-compliant time and in case of node failures during the service time.

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 map the requirements of a known media service workload into the corresponding system resource requirements and to accurately size the required system.*

The core of our capacity planning tool is a media service workload profiler, called *MediaProf*, which extracts a set of quantitative and qualitative parameters that characterize the service demand. In particular, *MediaProf* evaluates the number of simultaneous (concurrent) connections over time and classifies them into the encoding bit rate bins. Using a high-level memory model, *MediaProf* classifies the simultaneous connections by the file access type: whether a particular request is likely to be served from memory or disk, based on the history of previous accesses and the behavior of the server's main memory file buffer cache and its size.

Additionally, to reflect the specific access patterns of a given workload, *MediaProf* builds the *interval workload profile*: the characterization of the service demands over a set of predefined time intervals. This way, *MediaProf* can characterize the "amount" of possible load during the continuous time intervals. This characterization is useful to evaluate the performability of the particular system configuration for the amount of possible overload (or performance degradation) during a node failure lasting a continuous time. Using a set of synthetic media workloads, we demonstrate the importance of such a workload characterization. While different workloads may result in a similar aggregate demand profile, they may have a very different *interval workload profiles.*

Design of a capacity planning tool requires measurement and comparison of the capacities of different media servers. Currently, there is no a standard benchmark for measuring a media server capacity. In our recent work [4], we proposed a set of benchmarks for measuring the basic capacities of streaming media systems. Using the set of specially benchmarked media server configurations, the capacity planning tool matches the overall capacity requirements of the aggregate and interval media service workload profile with the specified SLAs to identify the number of nodes necessary in the media solution to support the required service performance.

In summary, our capacity planning tool provides a new unified framework with means for:

- measuring a media server capacity via a set of basic benchmarks and deriving a *cost* function for resource requirements of a particular media stream;
- deriving a special media site workload profile that can be directly mapped to the corresponding resource demand profile;
- combining the workload capacity requirements and the specified SLAss to produce the configuration with required performance characteristics.

The remainder of the paper presents our results in more detail.

## 2 Media Server Capacity Equations

Commercial media servers are characterized by the number of concurrent streams a server can support without loosing a stream quality, i.e. while the real-time constraint of each stream can be met. In paper [4], 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*: measures the media server capacity when all the clients in the test access the same file, and
- *Unique Files Benchmark*: measures the media server capacity when each client in the test accesses a different file.

Each of these benchmarks consists of a set of sub-benchmarks with media content encoded at a different bit rate. Using an experimental testbed, we measured capacity and scaling rules of a media server running RealServer 8.0 from RealNetworks.

Our 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.* This quantifies the performance benefits for multimedia applications when media streams are delivered from memory.

Using our basic benchmark measurements, we derived 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):

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

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 $W$,
- $N^{memory}_{X_{w_i}}$- a number of streams having a memory access type for a subset of files encoded at $X_{w_i}$ Kb/s,
- $N^{disk}_{X_{w_i}}$ - 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 capacity equation:

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

If $Demand \leq 1$ then the media server operates within its capacity. If, for example, the computed service demand is $Demand = 4.5$ it indicates that the workload requires *5* nodes (of the corresponding media server configuration) to avoid overload. In [5], 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).

## 3 Workload Profiler *MediaProf*

Media access logs are a critical component in decision making about future infrastructure. The access logs record information about the requests processed by the media server. The access log' entry provides a description of the user request for a particular media file. The typical fields contain information about the time of the request, the filename of the requested video, the advertised video duration (in seconds), the size of the requested file (in bytes), the elapsed time of the requested media file when the play ended, the average bandwidth (Kb/s) available to the user while the file was playing, etc. *MediaProf* creates a traffic profile for capacity planning by extracting the following characteristics from the access logs:

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

A media server capacity is characterized by the number of concurrent streams (connections) a server can support without loosing a stream quality. Thus first, *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.

• *Classification of the simultaneous connections into the encoding bit rate bins.*

Since the amount of system resources ("cost") and the server bandwidth needed to support a particular client request depend on the file encoding bit rate, *MediaProf* classifies the simultaneous connections into the encoding bit rate bins.

• *Classification of 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 in time to one another, then the first access may read a file from disk, while the subsequent requests may be accessing the corresponding file prefix from memory.

For this classification, we developed a *segment-based memory model* that reflects 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 exploits the real-time nature of streaming media applications and the sequential access to file content. 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.

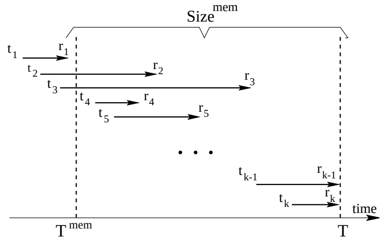


Figure 1: Memory state computation example.

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 1. This way, the files' segments streamed by the media server between times $T^{mem}$ and $T$ will be in memory. Thus, we can identify whether request $r$ will stream file $f$ (or some portion of it) from memory.

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

| Time Stamp | Concur. 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 represent minutes from the beginning of the trace (i.e. $t_i = t_{i-1} + 1$). 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.

In summary, *MediaProf* processes 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.

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

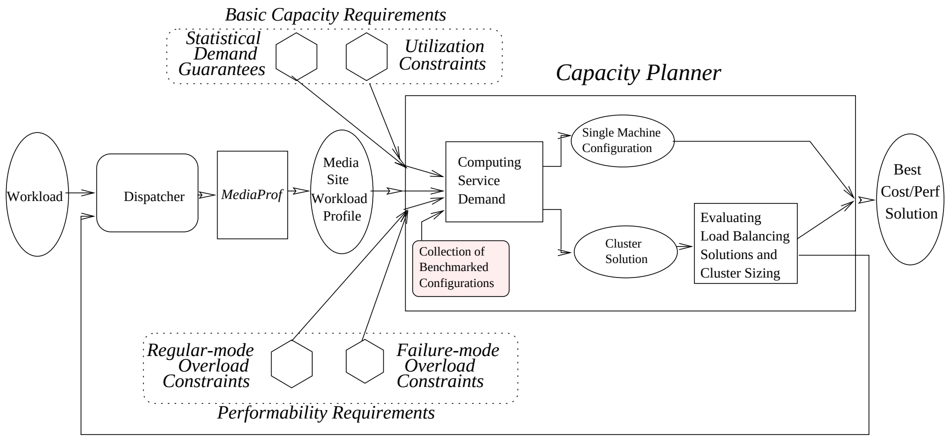

Basic Capacity Requirements
Statistical Demand Guarantees
Utilization Constraints
Capacity Planner
Workload
Dispatcher
MediaProf
Media Site Workload Profile
Computing Service Demand
Single Machine Configuration
Cluster Solution
Evaluating Load Balancing Solutions and Cluster Sizing
Collection of Benchmarked Configurations
Best Cost/Perf Solution
Regular-mode Overload Constraints
Failure-mode Overload Constraints
Performability Requirements

Figure 2: Capacity planning process.

## 4 Overall Capacity Planning Process

The overall capacity planning process is shown in Figure 2. There are the three phases in the capacity planning procedure:

- **Basic capacity planning** that derives the desirable configuration by taking into account two basic capacity requirements:
  - *Statistical Demand Guarantees*;
  - *Utilization Constraints*.
- **Performability capacity planning** that evaluates and refines the desirable configuration by taking into account two additional performability requirements:
  - *Regular*-mode *Overload Constraints*;
  - *Node-Failure*-mode *Overload Constraints*.
- **Cluster size validation**. If the configuration recommended by the basic capacity planning process is a single machine configuration then the capacity planning procedure is completed. Otherwise the *Capacity Planner* performs an additional refinement of the cluster sizing by evaluating the impact of the load balancing solution.

### 4.1 Basic Capacity Planning

Suppose the service provider would like to size the cluster solution for a given workload using a media server $S$ with a memory size $M_S$. There are several logical steps in the basic capacity planning procedure:

- *Compute the media site workload profile.*

Using the media site access logs, *MediaProf* computes a workload profile for a given memory size $M_S$ of interest. During the initial analysis, the *Dispatcher* component assumes that the media cluster contains a single node.

- *Compute the service demand profile.*

The next module, called the *Capacity Planner*, 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 using Equation (1) from Section 2 with the the cost functions corresponding to the media server $S$. Thus, the resulting (intermediate) profile is the list of pairs $(t_i, d_i)$ reflecting that in time $t_i$ the service demand is $d_i$. Then the *Capacity Planner* computes a *cumulative density function (CDF)* of aggregate service demand that is normalized over time. [2]

- *Combine the service demand profile and the basic capacity requirements of a desirable configuration.*

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, the service provider may specify:

- *Statistical Demand Guarantees*: "Based on the past workload history, find an appropriate performance solution that is capable of processing the applied load 95% of the time". Using the CDF of computed service demand profile, the *Capacity Planner* finds the 95th percentile of the site's service demands over time. Let us denote this demand as $D_{95\%}$.

- *Utilization Constraints*: "Based on the past workload history, find an appropriate performance solution that is utilized under 70% of its capacity 90% of the time". This way, a service provider may specify a configuration with some reasonable "spare" capacity for future growth and changing access patterns. The *Capacity Planner* finds the 90th percentile of the site's service demands. i.e. $D_{90\%}$. Then the requirement for a configuration that is utilized under 70% of its capacity is $(D_{90\%}/0.7)$. Let us denote this demand as $D_{Util}$.

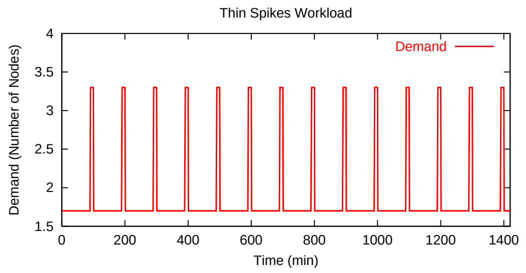
Thus, the basic capacity requirement for a desirable configuration is: $D_{basic} = max(D_{95\%}, D_{Util})$ rounded up to the closest integer.

[2]Since we assume that media files are encoded at a constant bit rate it is a straightforward task to compute the CDF of network bandwidth requirements and incorporate them in the capacity planning process. In this paper, we concentrate on the number of nodes in the cluster needed to process a given workload with specified performance requirements.

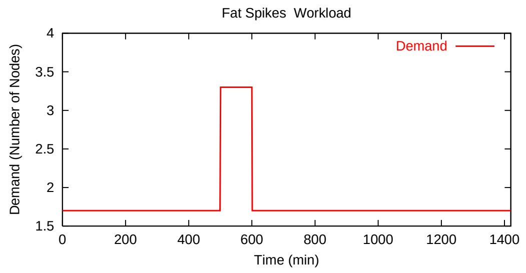
## 4.2 Evaluating Performability Requirements

The basic capacity planning process, described in Section 4.1, derives the desirable configuration by sizing the system according to the main performance requirements for the compliant time, e.g. identifies the system that is capable of processing the applied load with no performance degradation for 95% of the time. However, it does not provide any guarantees or limits on how "bad" the system performance could be in the remaining 5% of non-compliant time. The performability capacity planning evaluates the workload performance on the configuration recommended by the basic capacity planning process and refines it in order to limit the amount of possible overload per node during the regular processing time and/or to avoid the excessive performance degradation during periods when a node may fail.

Let us first consider a simple example. Figure 3 shows the service demands of two workloads over time (more exactly, a day-long sample of the workloads).



a)



b)

Figure 3: a) "Thin Spikes" workload; b) "Fat Spikes" workload.

The access patterns of these two workloads are very different. Workload shown in Figure 3 a) has a number of peak-load spikes each lasting for 10 min with more than 1 hour-time gaps between the spikes. Let us call this workload as a "Thin Spikes" workload. Workload shown in Figure 3 b) has a different access pattern: there is a single peak-load spike lasting for a duration of 100 min. Let us call this workload as a "Fat Spikes" workload.

These two workloads have the same CDF of service demand: 90% of the time, the service demand is 1.7 nodes, while for 10% of the time it reaches a peak load demand of 3.3 nodes.

Let a service provider specify the desirable configuration as one that:

- 90% of the time satisfies the workload demand;
- 90% of the time is utilized under 70%.

Then the basic capacity planning will recommend a 3-node cluster as an appropriate solution:

$$D_{basic} = max(D_{90\%}, D_{Util}) = max(1.7, 2.4) = 2.4$$

Since the peak service demand is 3.3 nodes, it means that in the 3-node cluster, the applied load reaches 110% per node, i.e. the maximum overload per node reaches 10%, and it is observed for 10% of the time for both workloads.

While the "aggregate" amount of overload per node is the same for both workloads, there is a significant qualitative difference in the amount of "continuous" overload exhibited in the two considered workloads. Intuitively, while the "Thin Spikes" workload looks more bursty, the amount of overload per any continuous hour is limited: no more than 10 min of 10% overload. For the "Fat Spikes" workload, any 1 hour interval between the time stamps 500 and 600 (as shown in Figure 3 b) experiences the continuous 10% overload.

From the QoS point of view, the short spikes of the performance degradations are less devastating than the longer periods of degraded performance. We believe that it is important to analyze workloads for the amount of continuous overload and take it into account during the capacity planning process [3].

In order to achieve this goal, the *Capacity Planner* builds an *interval overload profile* as follows. Let the $N$-node cluster be a configuration recommended for a given workload, and let $I$ be a duration of time interval of interest (in min). To compute the $I$-interval overload profile, we use the service demand profile described in Section 3. We use a "moving window" technique. A window is set to be $I$ min duration, and it is advanced in 1 min increments. For each such $I$-interval, any service demand above $N$ nodes is aggregated and then averaged over $N \times I$. This way, we can evaluate the average overload per node in any $I$-interval over the entire workload duration. Thus, the $I$-interval overload profile is the list of pairs $(t_i, d_i)$ reflecting that for the $I$-interval starting in time $t_i$, the average overload is $d_i$. For performability analysis, the *Capacity Planner* computes a *cumulative density function (CDF)* of aggregate $I$-interval overload which is normalized over the number of intervals.

Let us clarify this process with an example. For the two workloads considered above ("Thin Spikes" and "Fat Spikes" workloads) and the 3-node cluster configuration, let us consider the $I$-interval overload profiles for $I = 10\ min, 30\ min$, and $60\ min$.

Figures 4 a) and b) show the CDF of $I$-interval overload for $I = 10\ min, 30\ min$, and $60\ min$ for both workloads respectively. For the "Thin Spikes" workload, the CDF of the three interval overload profiles are very different. For $I$ of longer duration, the overall percentage

[3]In this work, we do not introduce the models of how the application may deal with the overload. Typically, it is application specific. There is a set of known strategies used in the commercial media servers to cope with overload, which result in delivering a lower quality stream and in this sense, leading to a degraded service performance. It is an interesting future direction to design a set of models representing the most popular strategies of delivering content under overload, and provide more specific metrics of degraded service performance.

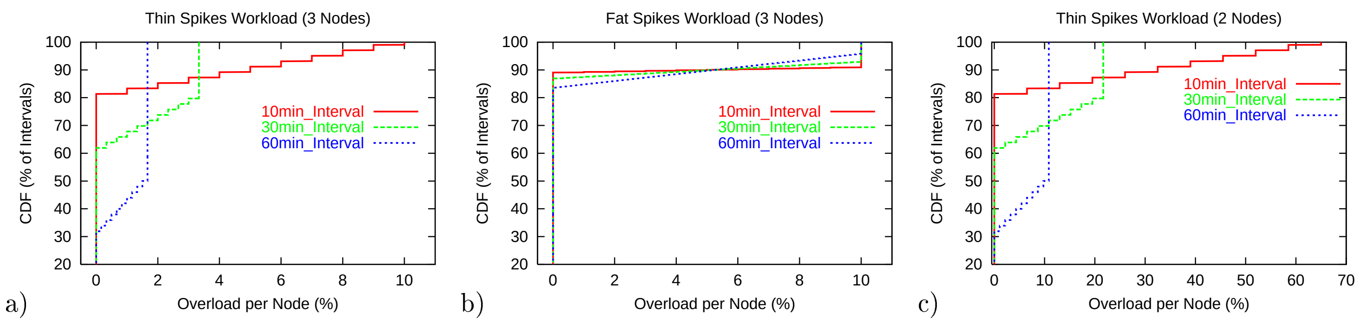


Figure 4: CDF of I-interval overload per node, where $I = 10\ min, 30\ min$, and $60\ min$: a) "Thin Spikes" workload, 3-node cluster; b) "Fat Spikes" workload, 3-node cluster; c) "Thin Spikes" workload, 2-node cluster.

of intervals with overload is higher than for $I$ of shorter duration. However, the amount of average overload in longer intervals is correspondingly lower. This is consistent with the nature of access patterns in this workload: while the longer intervals more likely have the overloaded time periods within them, these overload periods are short. It leads to a lower average overload per interval. On the other hand, for the "Fat Spikes" workload, the percentage of overloaded intervals and the amount of overload per interval are similar for all the three profiles reflecting the longer periods of consistent overload in the given workload.

Suppose the service provider specifies the following performability requirement for a solution of interest: "Based on the past workload history, find an appropriate performance solution such that the amount of average overload is limited by 2% in any 60 min interval".

Let us compare the CDF of 60 min-interval overload profiles for "Thin Spikes" and "Fat Spikes" workloads. For the "Thin Spikes" workload, the average overload is limited by 2% in any 60 min-interval, while for the "Fat Spikes" workload, 10% of the 60 min-intervals have overload higher than 2%. Thus, for the "Thin Spikes" workload, the 3-node cluster solution meets the performability requirement above. However, for the "Fat Spikes" workload, the 3-node cluster solution does not satisfy the desirable overload constraints, and the *Capacity Planner* will propose 4-node cluster as the minimal solution satisfying given performability requirements.

Let us denote the outcome of the performability capacity planning for acceptable overload during the regular processing as $D^{Reg}_{Overload}$.

For performability requirements, the service provider should choose the interval and degree of overload that reflect the service tolerance to QoS degradation. Specifying a short overload interval (i.e. setting the tolerance to continuous overload being very low) might diminish the usefulness of interval overload analysis because the CDF of interval overload profile will be clos to the CDF of the original service demand, and it might lead to overprovisioning for rare demand spikes.

The $I$-interval overload profile provides a useful insight into possible performance degradation in case of a node failure in the cluster. When a node fails in the $N$-node cluster, it will provide the service with the remaining $N-1$ nodes, but possibly, at a price of degraded performance. Since a node failure lasts a continuous period of time, the $I$-interval overload analysis for the $N-1$-node cluster provides both quantitative and qualitative characterization of possible amount of overload and its nature for the remaining cluster. (Similar analysis can be performed for 2-node failures, etc.)

Figure 4 c) shows the CDF of $I$-interval overload for $I = 10\ min, 30\ min$, and $60\ min$ in the 2-node cluster that is processing the "Thin Spikes" workload. While there are 10 min intervals with high continuous overload reaching 65%, these intervals are very rear, and 80% of 10 min intervals do not have any overload.

Suppose the service provider specifies the following performability requirement: "Based on the past workload history, find an appropriate cluster solution such that in case of 1-node failure the average overload per node in the remaining system is less than 20% in any 60 min interval". To satisfy this performability requirement, 3-node cluster will be required for the "Thin Spikes" workload and 4-node cluster for the "Fat Spikes" workload. Let us denote the outcome of the performability capacity planning for acceptable overload during 1-node failures $D^{N-1}_{Overload}$.

In summary, the desirable configuration that satisfies the specified performability requirement is determined by: $D_{overall} = max(D_{basic}, D^{Reg}_{Overload}, D^{N-1}_{Overload})$ rounded up to the closest integer.

## 4.3 Cluster Size Validation

If the configuration recommended by the basic capacity planning process is a single machine configuration then the capacity planning procedure is completed. Otherwise the *Capacity Planner* performs an additional refinement of the cluster sizing by evaluating the impact of the load balancing solution as well as the implications of the increased overall cluster memory.

A cluster of $N$ nodes represents $N$ times greater processing power, and at the same time, it has $N$ times larger combined memory. During the first iteration of capacity planning process, the classification of client requests into memory/disk accesses is done using a "single node" memory model. We need to re-evaluate workload performance on the cluster of recommended size by taking into account the load-balancing solution and the impact of increased memory in a cluster (due to multiple nodes).

Currently, in our capacity planning tool, we assume the traditional *Round-Robin (RR)* load balancing solution, that distributes the requests uniformly to all the machines in the cluster. We assume that each media server in a cluster has access to all the media content. Therefore, any server can satisfy any client request.

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

- Partition the original media site workload $W$ into $k$ sub-workloads $W_1, W_2, .., W_k$ using the *Dispatcher* employing a load balancing strategy (in our case, the *Round-Robin* strategy).
- Compute the media workload profile for each of sub-workloads $W_1, W_2, .., W_k$ using *MediaProf*.
- Merge the computed sub-workload profiles in the overall media site workload profile by using the time stamps of individual sub-workload profiles.
- Compute the overall service demand profile.
- Compute the refined basic service demand requirements $D_{basic}$.
- Compute the refined performability service demand requirements $D^{Reg}_{Overload}$ and $D^{N-1}_{Overload}$.
  - If the outcome of this step is still the capacity requirements of $k$ or less 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 ($l > k$) then the capacity planning process is repeated for the cluster configuration of $l$ nodes.

Typically, for the RR load balancing strategy, the sizing process converges at the second iteration. 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 in the memory of each machine. The results of our simulation experiments (presented in Section 5) show that under the *RR* strategy, the increased memory (due to combined memory of multiple nodes in the cluster) practically does not provide any additional performance benefits.

## 5 Capacity Planning: a Case Study

In this section, we present a capacity planning example based on realistic media workloads. The main goal of the example is to demonstrate the workload profiling, models and techniques introduced in the paper for accurate SLA-based capacity planning.

For workload generation, we use the publicly available, synthetic media workload generator *MediSyn* [13]. In our example, we explore two synthetic media workloads $W1$ and $W2$ that both closely imitate parameters of real enterprise media server workloads [3].

Both synthetic workloads have the same media file duration distribution, which can be 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. This distribution represent a media file duration mix that is typical for enterprise media workloads [3], where along with the short and medium videos (demos, news, and promotional materials) there is a representative set of long videos (training materials, lectures, and business events).

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: a new request arrives each second on average.

The file popularity for both workloads is defined by a generalized Zipf distribution [13] with $\alpha = 1.5$ and $k = 7$ in $k$-transformation. In summary, $W1$ and $W2$ have a fileset with 4000 files (with overall storage requirements of 207 GB), where 90% of the requests target 8% of the files. Correspondingly, these 8% of the most popular files have an overall combined size of 16.7 GB.

The major difference in generation of two workloads is in diurnal access pattern which defines how the number of accesses to a site varies during a given period of time, e.g., a day. In MediSyn, a user can specify a global diurnal pattern, which contains a set of bins. Each bin specifies a time period and the ratio of accesses in this bin. Workload $W1$ is defined by the 1-hour-long bins. Workload $W2$ has a diurnal access pattern defined by the 15-min-long bins. Hence, these two workloads slightly resemble "Fat Spikes" and "Thin Spikes" workloads considered in Section 4.2.

Let the capacity planning task be to find the appropriate media system configurations satisfying the following performance requirements for workloads $W1$ and $W2$:

- *Statistical Demand Guarantees*: for 95% of the time, the system configuration is capable of processing the given workload without overload;
- *Utilization Constraints*: for 90% of the time, the system configuration is utilized under 70% of its capacity;
- *Regular*-mode *Overload Constraints*: during any 60 min-interval, the average overload per node is less than 5%;
- *Node-Failure*-mode *Overload Constraints*: in case of 1-node failure, with 95% probability the amount of average overload per node in the remaining system is less than 10% during any 60 min interval.

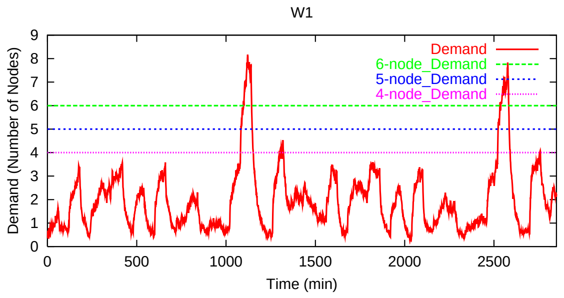
Let the benchmarked capacity of the media server of interest be defined as shown in Table 2.

| 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 |

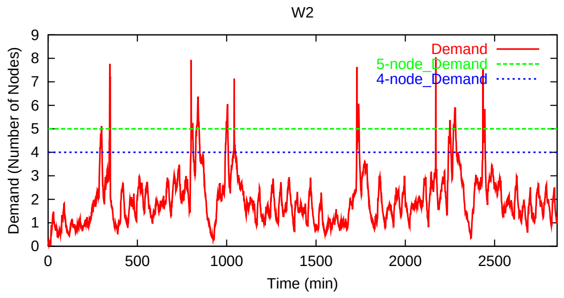
Table 2: Benchmarked media server capacity.

The server capacity scaling rules for different encoding bit rates (shown in Table 2 are similar to those measured using the experimental testbed described in Section 2. 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.

Finally, let the memory size of interest be 0.5 GB. [4] Let us denote this media server type as $\hat{S}$.



a)



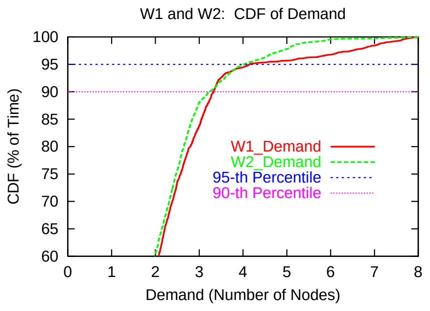
b)

Figure 5: Service Demand Profile for a) W1; b) W2.

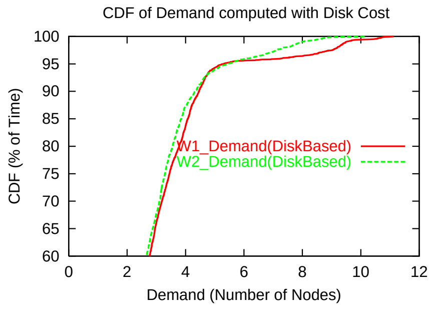
Figures 5 a), b) show the service demands of two workloads over time (more exactly 2 days sample of the workloads). Most of the time, the service demand of both workloads is below 4 nodes (we show 4-node demand through a horizontal line in order to easier see the demand below and above this mark). The peak load demand reaches 8 nodes for both workloads. However, the access patterns are clearly different: workload $W2$ is somewhat more bursty than $W1$; it has a larger number of high-demand spikes than $W1$, but these spikes are of a shorter duration compared to the high-demand spikes in $W1$.

Figure 6 a) shows the computed CDF of capacity requirements for processing workloads $W1$ and $W2$ on the media server $\hat{S}$. We present the upper half of the curve for better visibility.

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



a)



b)

Figure 6: W1 and W2 a) CDF of service demand profile; a) CDF of service demand profile computed with the disk cost.

One of the steps involved in computing the service demand profile is classifying whether a particular media request is likely to be served from memory vs disk. This computation is based on the high-level memory model used in *MediaProf*. The locality available in a particular workload has a performance impact on the behavior of the system because serving content from memory incurs much lower overhead than serving the same content from disk. In order to quantify this impact for correctly sizing the system configuration, we compute the resource demand profile, where all the requests are assigned a cost of disk (for brevity, we call this demand as disk-based), i.e. we assume that all the requests are served from disk. Figure 6 b) shows the computed CDF of disk-based capacity requirements for processing workloads $W1$ and $W2$. The maximum of disk-based service demand reaches 11 nodes compared to 8 nodes computed with our model-based approach. Thus, accounting for whether a particular media request might be served from memory or disk has significant implications for accurate capacity planning.

Applying the basic capacity planning to satisfy the *Statistical Demand Guarantees* and *Utilization Constraints*, we receive the 5-node cluster solution for both workloads:

$$W1: \quad D_{95\%} = 4.1 \quad D_{Util} = 3.3/0.7 = 4.7 \quad \Rightarrow D_{basic} = 5.$$

$$W2: \quad D_{95\%} = 4 \quad D_{Util} = 3.2/0.7 = 4.6 \quad \Rightarrow D_{basic} = 5.$$

Now, let us analyze whether the performability requirements (both during the regular processing and during 1-node failure scenario) are satisfied for the 5-node cluster configuration.

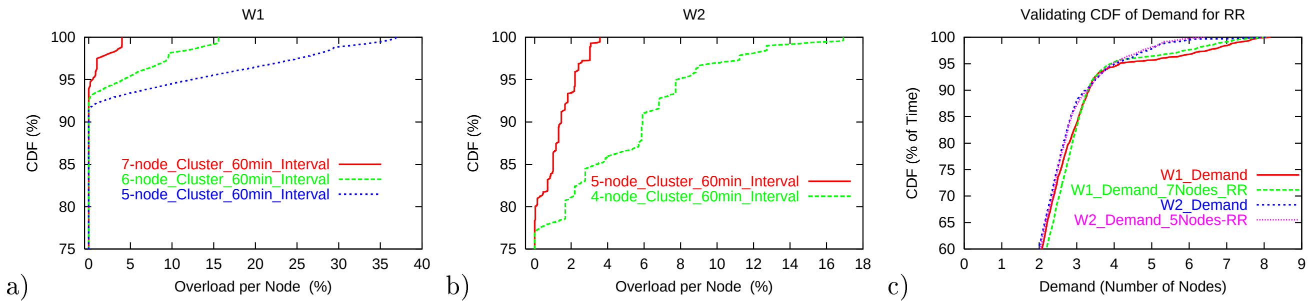


Figure 7: a) W1: CDF of overload per node measured in 60 min intervals in 5-node, 6-node, and 7-node media cluster; b) W2: CDF of overload per node measured in 60 min intervals in 4-node and 5-node media cluster; c) capacity demands under RR-strategy for $W1$ processed by 7-node cluster and $W2$ processed by 5-node media cluster.

Figure 7 a) shows the CDF of overload per node measured in 60 min intervals in 5-node media cluster for workload $W1$: the peak overload reaches 36%, and hence 5-node configuration does not meet performability requirements. In this situation, the capacity planner verifies if the 6-node configuration meets the performability requirements. As it is shown in Figure 7 a), the 6-node configuration does not meet the required conditions either: the peak overload reaches 15.5%. Only the 7-node configuration (see again Figure 7 a) finally meets the performability requirements on the amount of overload during the regular processing mode: the peak overload is only 4%. In order to validate whether the 7-node configuration meets the 1-node failure overload requirements, the capacity planner checks the CDF of overload per node measured in 60 min intervals in 6-node cluster: the 95th percentile of overload per node is 4.2% as shown in Figure 7 a).

Therefore the outcome of the performability capacity planning for workload $W1$ is a recommendation of 7-node cluster.

Let us analyze whether performability requirements on overload are satisfied by the 5-node cluster configuration for workload $W2$. Figure 7 b) shows the CDF of overload per node measured in 60 min intervals in 5-node media cluster for workload $W2$: the peak overload reaches 4%, and hence 5-node configuration does meet the performability requirements under the regular processing mode. Figure 7 b) also shows the CDF of overload per node measured in 60 min intervals in 4-node cluster (a configuration when 1 node fails): the 95th percentile of overload per node is 8.1% as shown in Figure 7 b).

Therefore the outcome of the performability capacity planning for workload $W2$ is a recommendation of 5-node cluster.

The final phase of capacity planning is the cluster size validation for evaluating the impact of the Round-Robin load balancing solution and the implications of the increased overall cluster memory. Figure 7 c) shows the computed CDF of capacity requirements for processing workload $W1$ on the 7-node cluster and $W2$ on the 5-node cluster using RR-strategy in comparison to the CDF of service demands computed for workloads $W1$ and $W2$ at the first iteration of capacity planning process. Validation shows that under the $RR$ load balancing strategy, the increased memory (due to combined memory of multiple nodes in the cluster) does not provide the additional performance benefits, and the recommended solutions for both workloads are correct. This completes the capacity planning process.

The difference in the outcome – 7-node cluster for $W1$, and 5-node cluster for $W2$ – is intuitively expected: workload $W1$ has a longer durations of continuous overload compared to short spikes of high-load in $W2$, and the designed performability framework incorporated in the capacity planning tool is capable of capturing this difference.

## 6 Related Work

It is commonly recognized that multimedia applications can consume significant amounts of server and network resources. Traditionally, network bandwidth or disk system throughput has been the target of optimizations and sizing for streaming media services [14, 6, 2, 7]. Most of the designed models deal with the complexity of real-time delivery of variable bit rate content. In our paper, we assume a constant bit rate encoding for media content (that is typical for commercial systems). Thus, for a given workload, it is easy to determine what network bandwidth is required. The difficult task is to determine the system resources (amount of CPU, memory, and disk resources) needed to process a given workload with specified performance requirements. This is the main focus of our work.

The concept of performability [8, 11, 12] captures the combined performance and dependability characteristics of the system, i.e. how well it performs in the presence of failures over some time interval. Direct measurements of performability are only possible when a full-scale working prototype exists. Before then, system designers and service providers must rely on models of the candidate configurations. In our work, we design appropriate models of the media workload and the supporting media system that allow the capacity planning process to evaluate the possible service degradation (overload conditions) in case of node failures during the service time. In this work, we do not consider the space of all possible failure scenarios. Instead, we rather concentrate on providing the analysis of the potential performance degradation, when one (or more) of the nodes in the cluster fails to support the service.

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. For measuring a performance of commercial web servers, there is a well-defined suite of commercial benchmarks. There are also well-developed methods and techniques for capacity planning of commercial web and e-commerce sites [9, 10]. However, similar benchmarks and the corresponding performance studies are not currently available for commercial (academia/research) media servers. Work performed in this paper presents a step in this direction.

## 7 Conclusion and Future Work

In this paper, we outlined a new model-based capacity planning framework for evaluating the capacity requirements of a given media workload. Our capacity planning tool is comprised of several novel inter-related components:

- the capacity measurements of different h/w and s/w solutions using a specially designed set of media benchmarks and the derived *cost* function that provides a single value to reflect the combined resource requirements (e.g., CPU, disk, memory) necessary to support a particular media stream;
- the workload profiler *MediaProf*, which derives a special media site workload profile that characterizes a number of concurrent connections over time and their encoding bit rates. Then using a high-level memory model, it evaluates whether a request will be likely streaming data from memory or will be accessing data from disk. Constructed media workload profile can be directly mapped to the capacity requirement for a particular media server of interest using the cost function from the benchmarked configuration of this server;
- the *Capacity Planner* that allows a service provider to specify the desirable system performance via the requirements of the *Service Level Agreement (SLA)*. A special feature of the *Capacity Planner* is that it provides a performability analysis of the configuration suggested by the basic capacity planning process for the amount of continuous overload per node both during the regular processing and during the node failure periods.

We envision an interesting application of our capacity planning tool when a streaming media service is hosted in Utility Data Center (UDC) [15]. 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 the future, we intend to use our capacity planning tool as a core of an adaptive management system in the streaming media utility for determining when to deploy additional server resources in order to accommodate growing user demand or changing access characteristics.

**Acknowledgments:** We would like to thank our HPLabs colleagues Kim Keeton, Mustafa Uysal, Renato Santos, and Xiaoyun Zhu for stimulating discussions which helped in improving the presentation of the paper.

## References

[1] J. Almeida, J. Krueger, D. Eager, and M. Vernon. Analysis of Educational Media Server Workloads. Proc. of ACM NOSSDAV, 2001.

[2] 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.

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

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

[5] 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.

[6] 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.

[7] 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.

[8] J. Meyer. On Evaluating the Performability of Degradable Computing Systems. IEEE Transactions on Computers, C-29(8), August, 1980.

[9] D. Menasce, V. Almeida. Capacity Planning for Web Performance: Metrics, Models, and Methods. Prentice Hall, 1998.

[10] D. Menasce, V. Almeida. Scaling for E-Business: Technologies, Models, Performance, and Capacity Planning. Prentice Hall, 2000.

[11] . W. Sanders and J. Meyer. A unified approach for specifying measures of performance, dependability, and performability. In Dependable Computing for Critical Applications, vol. 4 of Dependable Computing and Fault-Tolerant Systems, Springer-Verlag, 1991.

[12] T. Triverdi, G. Ciardo, M. Malhutra, and R. Sahner. Dependability and performability analysis. In Performance Evaluation of Computer and Communication Systems. Springer-Verlag, 1993.

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

[14] H. Vin, P. Goyal, A. Goyal, A. Goyal. A Statistical Admission Control Algorithm for Multimedia Servers. In Proc. of ACM Multimedia, San Francisco, 1994.

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