Building a Performance Model of Streaming Media Applications in Utility Data Center Environment

Ludmila Cherkasova, Loren Staley
Internet Systems and Storage Laboratory
HP Laboratories Palo Alto

HPL-2003-15
January 23rd , 2003*

E-mail: cherkasova@hpl.hp.com, loren_staley@hp.com

Utility Data Centers, enterprise media servers, media system benchmarks, measurements, capacity metrics, media server capacity, performance models

Utility Data Center (UDC) provides a flexible, cost-effective infrastructure to support the hosting of applications for Internet services. In order to enable the design of a "utility-aware" streaming media service which automatically requests the necessary resources from UDC infrastructure, we introduce a set of benchmarks for measuring the basic capacities of streaming media systems. The benchmarks allow one to derive the scaling rules of server capacity for delivering media files which are: i) encoded at different bit rates, ii) streamed from memory vs disk. Using an experimental testbed, we show that these scaling rules are non-trivial. In this paper, we develop a workload-aware, media server performance model which is based on a cost function derived from the set of basic benchmark measurements. We validate this performance model by comparing the predicted and measured media server capacities for a set of synthetic workloads.

* Internal Accession Date Only Approved for External Publication
A shorter version of this paper will be published in IEEE/ACM International Symposium on Cluster and the Grid (CCGrid 2003) 12-15 May 2003, Tokyo, Japan

Building a Performance Model of Streaming Media Applications in Utility Data Center Environment

Ludmila Cherkasova
Hewlett-Packard Laboratories
Internet Systems and Storage Lab
1501 Page Mill Road
Palo Alto, CA 94303, USA
cherkasova@hpl.hp.com

Loren Staley
Hewlett-Packard Co
Always On Infrastructure Solutions
19111 Pruneridge Ave
Cupertino, CA 95014, USA
loren_staley@hp.com

Abstract *Utility Data Center (UDC) provides a flexible, cost-effective infrastructure to support the hosting of applications for Internet services. In order to enable the design of an "utility-aware" streaming media service which automatically requests the necessary resources from UDC infrastructure, we introduce a set of benchmarks for measuring the basic capacities of streaming media systems. The benchmarks allow one to derive the scaling rules of server capacity for delivering media files which are: i) encoded at different bit rates, ii) streamed from memory vs disk. Using an experimental testbed, we show that these scaling rules are non-trivial. In this paper, we develop a workload-aware, media server performance model which is based on a cost function derived from the set of basic benchmark measurements. We validate this performance model by comparing the predicted and measured media server capacities for a set of synthetic workloads.*

1 Introduction

A Utility Data Center (UDC) [23, 24] provides a flexible, cost-effective solution by using advanced management software which allows resources to be automatically reassigned in response to changing business and IT requirements. The UDC architecture is an ideal platform to support the efficient hosting of applications for Internet services. Most Internet applications are difficult to manage due to their highly variable service demand. The most typical practice used by current service providers is to significantly over provision the amount of resources needed to support such applications by tailoring the resources to maximum expected load. 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.

Streaming media has quickly become the most popular form of multimedia content on the Internet. Video from news, sports, and entertainment sites are more popular than ever. Media servers are being used for educational and training purposes by many universities. Use of the media servers in the enterprise environment is catching momentum too.

The area of multimedia services in a networked environment is one of the rapidly expanding fields in today's technological world. 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 evidence from media workload analysis [1, 2, 4] indicates that client demands are highly variable: some days (hours) exhibit 20-30 times higher load comparing to the typical daily (hourly) averages. Additionally, media applications are characterized by stringent real-time constraints, as each stream requires isochronous data playout. These features make media application a perfect candidate to benefit from flexible resource management and resource provisioning in the Utility Data Center infrastructure.

Our ultimate goal is to design and implement an "utility-aware" streaming media service which automatically requests the necessary resources from the UDC infrastructure in order to adapt to variable workload demand. Among the *prerequisites* to a design of such a utility service is the ability to measure the capacity of media servers (nominal and currently in use) in order to evaluate the amount of available system resources for admitting new client requests or to request/release resources as needed.

In this paper, we develop *a workload-aware performance model of streaming media application*s which lays down a foundation of our utility-aware service design. In order to build such a performance model several classic performance questions should be answered:

- how to measure the *basic* capacity of a streaming media server?
- what is the set of *basic* benchmarks exposing the performance limits and main bottlenecks of a media server?
- how to estimate the expected media server capacity for realistic workload if the measured capacities of streaming media server under the basic benchmarks are given?

Commercial media server solutions are distinguished by the number of concurrent streams a server can support [17] without loosing a quality of stream, i.e. until real-time constraint of each stream can be met. A standard commercial stress test measures a maximum number of concurrent streams delivered by the server when all the clients are accessing the same file encoded at a certain bit rate, e.g. 500 Kb/s.

However, a multimedia content is typically encoded at different bit rates depending on a type of content and a targeted population of clients and their connection bandwidth to the Internet. What are *the scaling rules for server capacity* when delivered media content encoded at different bit rates? For example, if a media server is capable of delivering N concurrent streams encoded at 500 Kb/s, will this server be capable of supporting $2 \times N$ concurrent streams encoded at 250 Kb/s? The other issue with a standard commercial stress test is that all the clients are accessing the same file. Thus another question to answer is: how a media server performance is impacted when different clients retrieve different (unique) files of a media content?

We introduce a simple set of *basic benchmarks* to analyze the basic performance limits and main bottlenecks of media server:

- *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. Through the paper, we use a six basic bit rates: 28 Kb/s, 56 Kb/s, 112 Kb/s, 256 Kb/s, 350 Kb/s, and 500 Kb/s. This list can be customized accordingly to the specifics of media content delivered by the service provider.

The *Single File Benchmark* and the *Unique Files Benchmark*, establish the scaling rules for server capacity when delivered streams are encoded at different bit rates. The measurement results reported in the paper show that these scaling rules are non-trivial. 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 2.5-3 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.

There are different system resources limiting a media server performance under the different basic benchmarks. Under the *Single File Benchmark*, the server performance is CPU bounded, while under the *Unique Files Benchmark*, the server performance is disk-bound. While there are general performance tools for monitoring the CPU utilization, the disk bottleneck is hard to measure with the usual performance tools. The maximum bandwidth delivered by a disk depends on the number of concurrent streams it can support with an acceptable level of *jitter*, i.e. without violating on-time delivery constraints.

A typical media server traffic presents a combination of some clients accessing the same files (during the same period of time) and some clients retrieving the "unique", different files. Under the "mixed" workload, neither a CPU nor a disk are the limiting performance system resources, and hence, they can not be used for monitoring of the available server capacity. In order to manage streaming media service efficiently, we need to be able to estimate:

- what is a currently available/used service capacity?
- what fraction of service capacity is required to support a particular stream without a degradation of its quality?

In this paper, we develop a *workload-aware performance model* of media server aiming to estimate the media server capacity for delivering a realistic workload. This model is based on a *cost* function derived from the set of basic benchmark measurements. The cost function 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). We validate this performance model by comparing the predicted and measured media server capacities for a set of synthetic workloads.

In summary, the main contribution of the paper is a new unified framework for

- measuring a media service capacity via a set of basic benchmarks;
- deriving the resource requirements (a cost) of a particular stream from the basic benchmarks measurements;
- estimating a currently available/used service capacity under the realistic workloads via a new workload-aware performance model based on a cost function.

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

2 Experimental Testbed

Figure 1 shows our experimental testbed used for measuring the media system performance:

- a server is rp2450 system with 1-way 550MHz PA 8600 processor, memory – 4 GB, IO Cards – 2 1000SX Ethernet, 4 Ultra2 SCSI port on 2 dual port cards, disk configuration – 4x15K rpm 18.2 GB disks 4-way striped using HP-UX LVM, HFS file system with block size configured to 64KB; operating system – HPUX 11.0,

- 14 clients machines are rp2450 systems with 2-way 550MHz PA 8600 processors, running HPUX 11.0 operating system,

- 14 clients machine are connected to a server by a Gbit switch 4108gl via 1Gb/s links,

- a media server software: RealServer 8.0 from RealNetworks[17].

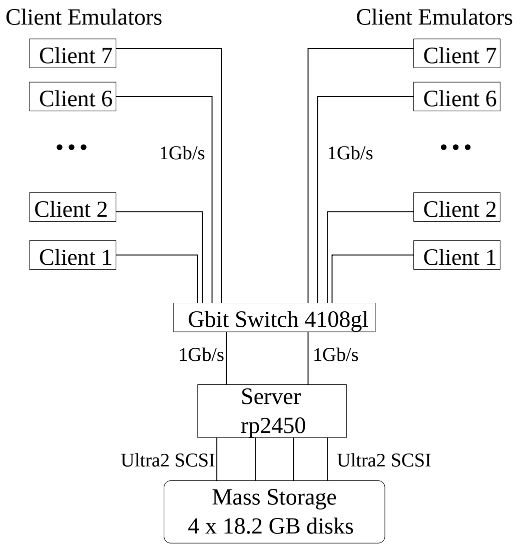


Figure 1: Experimental setup.

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. In Sections 3.2, 3.3, we will show that achievable bandwidth under the maximum server load is significantly lower than the physical bandwidth of communication links available in the system. Such system configuration exposes the performance limits specific to application itself, and allows us to derive more general conclusions about its performance. Our experimental setup uses the general purpose components available in Utility Data Center environment. [1]

For the load tests, the client emulators are instructed to start a predefined number of streams requesting a certain set of files. The emulator software has a flexibility allowing different clients to request different files. Additionally, different emulators may be configured with their own specific parameters as for the number of streams as for a set of requested files. These capabilities provide a convenient way to imitate a wide range of realistic workloads of interest.

[1]Typically, UDC uses a SAN-based storage solution. A technique and approach proposed in the paper is not limited to a particular configuration of the experimental testbed used in the study: it allows one to measure a performance of multimedia applications in a very general setting.

3 Set of Basic Benchmarks and Metrics

In this section, we introduce a set of *capacity metrics* and *basic benchmarks* aimed at determining the performance limits of media server and analyzing its main performance bottlenecks. In Section 3.1, we describe two capacity metrics used in our experiments. Section 3.2 introduces a *Single File Benchmark* and the related measurement results. Section 3.3 is devoted to a *Unique Files Benchmark* and its performance results.

Both of the benchmarks introduced below use specially created 20 min video clips encoded at different bit rates with RealProducer G2 [18] from RealNetworks. The following six bit rates are chosen to represent the typical needs of the Internet audience:

- 28 Kb/s for analog modem users,
- 56 Kb/s for analog modem and ISDN users,
- 112 Kb/s for dual-ISDN users,
- 256 Kb/s for cable modem users,
- 350 Kb/s for DSL/cable users,
- 500 Kb/s for high-bandwidth users.

The primary objective of the basic benchmarks is to define how many concurrent streams of the *same bit rate* can be supported by a media server without degrading the quality of any streams.

3.1 Capacity Metrics: Server Overload and Degraded Stream Quality

In our experimental setup, we use two main performance metrics to determine whether a server has reached its maximum capacity for the applied workload. One of these metrics is collected at the media server side and the other one - at the client side. These metrics are complementary. Typically, the server side metric reliably reflects the server state and can be used alone to determine whether the server capacity for applied workload is reached. [2] The client side metric is useful to double check that the achieved server capacity does not degrade the quality of delivered streams. Additionally, the client side metric helps to evaluate the *fairness of media service* for mixed workloads and answer more specific questions such as: once a media server gets to an overload state which streams are experiencing the server overload and as a result have a degrading quality of service?

Server Side Metric: *Server Overload Metric.* RealServer can be configured to report a set of useful statistics (*ServerStats*) about its current state. The default value for a reporting interval is 3 sec, we use a 10 sec value in our experimental setup. In particular, *ServerStats* provide the information on:

[2]In case, when the network bandwidth in the measurement testbed is a limiting performance bottleneck, the combined information from server-side and client-side metrics helps to reveal this bottleneck.

- the number of streams currently playing by the server,
- the aggregate bandwidth requirements of currently accepted streams,
- the average bandwidth delivered by the server during the reporting interval,
- the number of packets sent by the server during the reporting interval,
- the number of packets being sent late against the application target time but still in time: a warning information,
- the number of packets being sent with violation of the real-time constraints: an alarming information about a server being in *overload state*.

We list here only a few statistics from *ServerStats* which we observe during the stress tests for server monitoring purpose.

To determine whether the server capacity is reached, we use a *server overload metric* providing the information about the packets sent with violation of "on-time delivery". These packets are indicative of a server being overloaded and that its available capacity is exceeded.

Client Side Metric: *Degraded Stream Quality Metric.* On a client side, we are interested in observing whether a client has entered a *rebuffering state* which means:

- the current "play" buffer is empty,
- the client has stopped playing,
- the client waits until the "play" buffer is filled to acceptable level to continue play back.

We use a specially written software, called *ClientStats*, to monitor the client state and to observe:

- the number of *rebuffering events*: how many times the client entered a rebuffering state,
- the statistics on the average stream bandwidth received by the client.

In our experimental setup, where communication infrastructure between the server and the clients is not a limiting resource, the existence of rebuffering events on the client side reflects a degrading quality of the delivered streams as a results of the server being overloaded. Degraded stream quality metric serves as a complementary metric to determine whether the media server capacity has been reached.

3.2 Single File Benchmark

The *Single File Benchmark* measures a media server capacity when all the clients in the test are accessing the same file.

We designed a completely automatic benchmark which runs a sequence of tests with an increasing number of clients requesting the same file from the server for each of the six encoding bit rates. During each test, the following performance data and measurements are collected:

- *ServerStats*,
- *ClientStats*,
- general performance information of a server machine using *vmstat*,
- general performance information of the client machines using *vmstat*.

To make sure that none of the video clip data are present in the file buffer cache (memory), and a file is streamed from the disk, a file system is unmounted and mounted back before each test point.

The main goal of the basic benchmarks is to define how many concurrent streams of the *same bit rate* can be supported by the media server without degrading the quality of streams.

While the maximum achievable media server capacity depends on the encoding bit rate of the file, the performance data collected at the server and the clients are similar. Using two representatives, files encoded at 112 Kb/s and 350 Kb/s, we show more details about server performance, its limits and bottlenecks under the *Single File Benchmark*.

For a file encoded at 112 Kb/s, a media server capacity is reached at 1350 concurrent streams, and for a file encoded at 350 Kb/s – at 500 concurrent streams. It is determined by a server overload metric and a complementary, degraded stream quality metric at the client side.

First of all, we observe that the media server capacity scales differently than the encoding bit rates of the underlying files: the difference in encoding bit rates between 112 Kb/s file and 350 Kb/s file is 3.125, while the difference in server capacity for corresponding bit rates is only 2.7. Intuitively, media server incurs an additional overhead while handling a **higher number** of concurrent streams encoded at lower bit rate comparing with handling a **smaller number** of concurrent streams encoded at higher bit rate (even for a case, when the aggregate bandwidth requirements of the delivered streams are the same).

Figures 2 a), b) show the CPU utilization and the application related part of it for corresponding file bit rates. Both figures look similar. Under the *Single File Benchmark*, the media server is CPU bounded: CPU utilization reaches 100%. It is the main resource limiting server performance.

The shape of the CPU utilization curve is not linear for both encoding bit rates. During the first half of the server capacity load, the CPU utilization grows linearly and reaches nearly 85%. The second half of the server capacity is achieved by utilizing the remaining 15% of the CPU resources, and finally, when the CPU utilization reaches 100%, it becomes a limiting factor of the media server performance.

Often, the utilization of a bottlenecked resource is used to build a simple normalized performance model of a corresponding system or application. For example, a system capacity achievable at 100% utilization of a bottleneck resource represents a 100% of applied load. Then at 50% utilization of a limiting resource

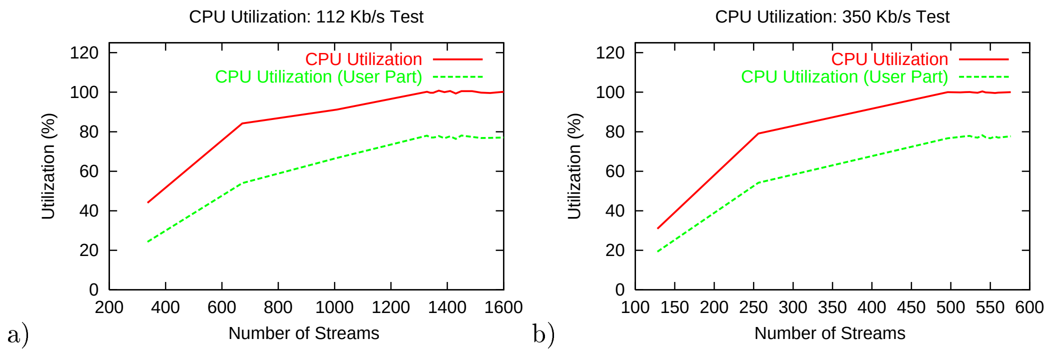


Figure 2: Server utilization under *Single File Benchmark*: a) 112 Kb/s streams b) 350 Kb/s streams.

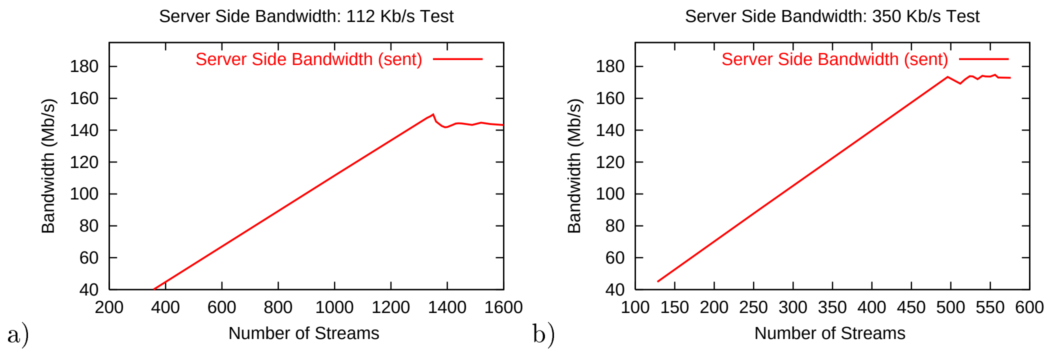


Figure 3: Server bandwidth under *Single File Benchmark*: a) 112 Kb/s streams b) 350 Kb/s streams.

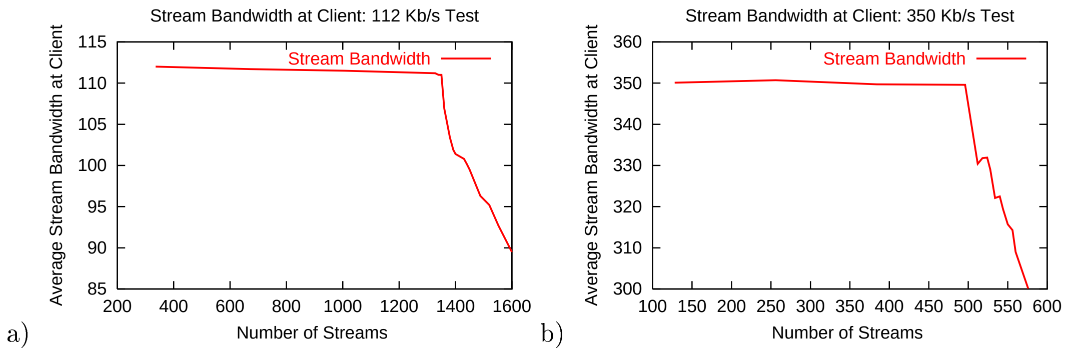


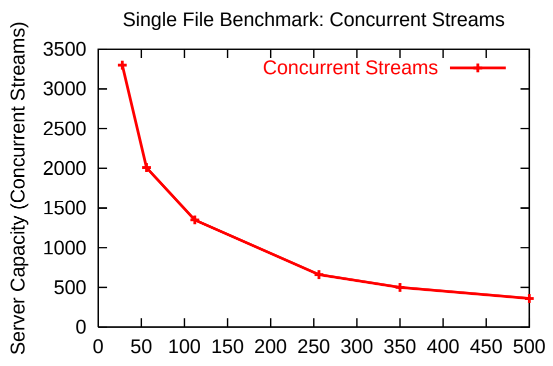
Figure 4: Stream bandwidth under *Single File Benchmark*: a) 112 Kb/s streams b) 350 Kb/s streams.

a system can process a 50% of an applied load (i.e. 50% utilization of a bottleneck resource corresponds to 50% of server capacity). The shape of the CPU utilization curve in Figure 2 overwrites an existence of such simple performance model for streaming media server (even for a case, when a workload is delivered from memory). We would like to stress this observation in light of a performance model developed in Section 4.

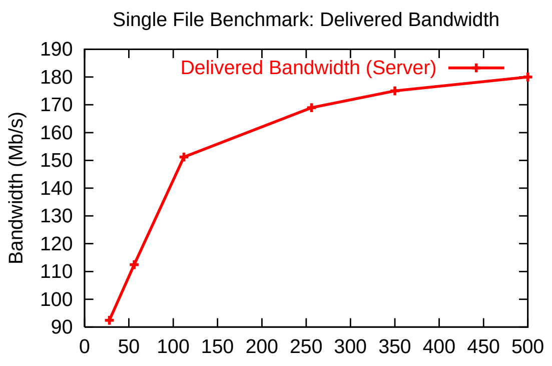
Figures 3 a), b) show the overall bandwidth in Mb/s delivered at the server side. It grows linearly, until the server capacity is reached, and after that bandwidth delivered by the server flattens. For a *Single File Benchmark* and a file encoded at 112 Kb/s, the bandwidth delivered by a server peaks at 151.2 Mb/s, and for a file encoded at 350 Kb/s, the maximum delivered bandwidth is 175 Mb/s. Thus, the bandwidth delivered by a server is far smaller than the capacity of the server network connections: server is connected to the clients via 2x1Gb/s links.

Figures 4 a), b) present a complementary information from the client side: once a server capacity is reached, the quality of the streams starts to degrade significantly: 10% of extra load causes around 9% of stream bandwidth degradation.

The set of graphs in Figures 3, 4 helps to predict the quality of service implications under server overload condition. Typical media server application does not have a "built-in" admission control mechanism. Since a maximum server capacity defines the maximum bandwidth a server can deliver at a particular bit rate, any accepted load in addition to a basic server capacity results in degradation of delivered stream quality: X% of additional load causes $\frac{100\times X}{(100+X)}$ % of degradation in delivered stream bandwidth. Thus, measurements of media server performance under a set of basic workloads provide useful insights into the specifics of application behavior under overload conditions which can be used in design of an efficient admission control strategy for



a)



b)

Figure 5: *Single File Benchmark*: a) Media Server Capacity b) Maximum Delivered Bandwidth.

media servers. Utility-aware services typically require a certain level of quality guarantees. Understanding the relationship between the server overload and QoS of delivered streams is essential for implementing such guarantees.

Figure 5 a) presents the maximum capacity in concurrent streams achievable by the streaming media server across six different encoding bit rates under the *Single File Benchmark*. Figure 5 b) shows the corresponding maximum bandwidth in Mb/s delivered by the media server for different encoding bit rates. The shape of the curve is interesting: for higher encoding bit rates the difference in achievable bandwidth is much less significant than for lower bit rates. Many admission control strategies designed in literature use the "fixed" maximum bandwidth a server is capable to deliver as a main "scheduling" resource for admission of a new stream. Evidently, the amount of bandwidth a server is capable to deliver is variable and depends on the encoding bit rates of current streams in progress.

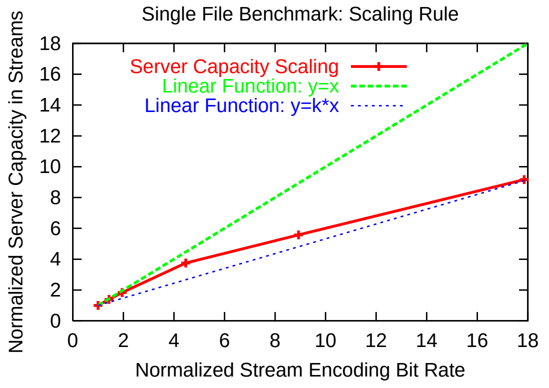


Figure 6: *Single File Benchmark*: Server Capacity Scaling Rules.

Figure 6 shows the normalized graph reflecting the scaling rules for the media server capacity under the *Single File Benchmark* and different encoding bit rates. In this figure, point (1,1) presents the maximum capacity (360 concurrent streams) achievable by a server when all the clients in the test are accessing the same file encoded at a 500 Kb/s bit rate. Each absolute value for the other encoding bit rate is normalized with respect to it. For example, the maximum capacity achievable by a server under *Single File Benchmark* and a 28 Kb/s encoding bit rate is 3300 concurrent streams and is represented by a point (17.9, 9.2).

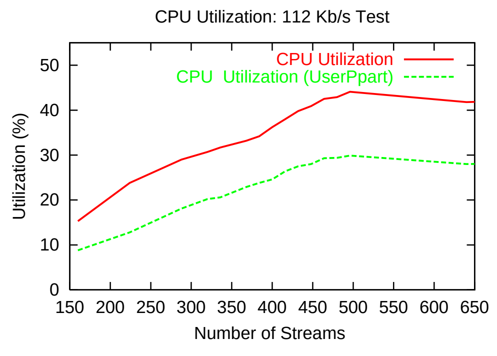
Figure 6 reflects that the server capacity scaling rules are non-trivial. For example, the difference between the highest and lowest bit rate of media streams used in our experiments is about 18 times. However, the difference in maximum number of concurrent streams a server is capable of supporting for corresponding bit rates is only around 9 times. Figure 6 shows that a media server capacity scales non-linearly comparing to the encoding bit rates of underlying files.

In summary, under the *Single File Benchmark* only one stream reads a file from the disk, while all the other streams read the corresponding bytes from the file buffer cache. Thus, practically, this benchmark measures a streaming server capacity when the media content is delivered from memory. However, it is not necessary that the streamed media file completely resides or fits in the memory. In essence, this benchmark exercises the shared access by multiple clients to the same file.

3.3 Unique Files Benchmark

The *Unique File Benchmark* measures the media server capacity when each client in the test is accessing a different (unique) file.

Similarly, for the *Unique File Benchmark*, we designed a completely automatic benchmark for all of the six encoding bit rates, which runs a sequence of tests with an increasing number of clients, where different clients request different files from the server. For a test point with N different clients, each file from the original basic set of files is replicated N times correspondingly, where different copies of the file are given different file names. We used the LVM features of HP-UX to create a striped volume across the 4 disks in our experimental setup. To make sure that none of the video clip data are present in the file buffer cache (memory), and all the files are streamed from the disk, a file system is


a)

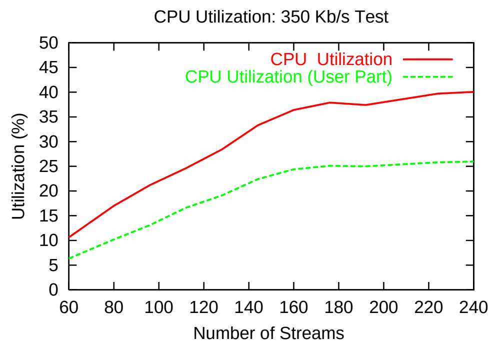
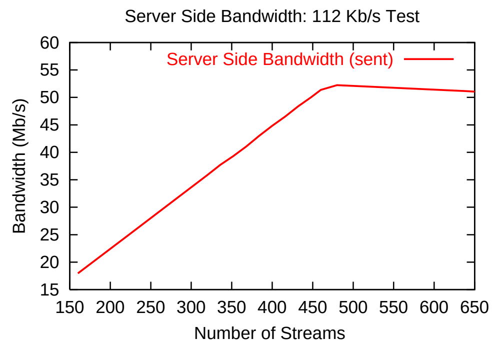

b)

Figure 7: Server utilization under *Unique File Benchmark*: a) 112 Kb/s streams b) 350 Kb/s streams.


a)

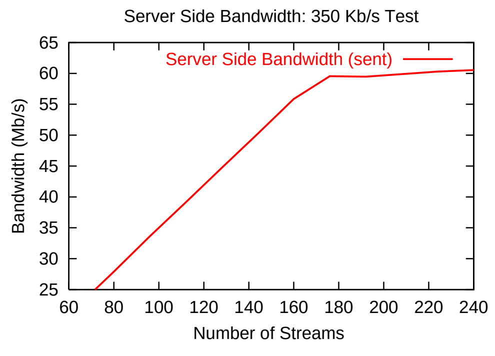
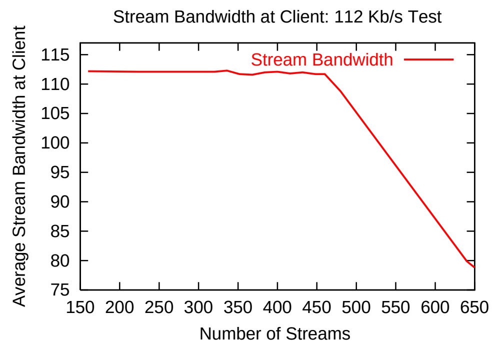

b)

Figure 8: Server bandwidth *Unique File Benchmark*: a) 112 Kb/s streams b) 350 Kb/s streams.


a)

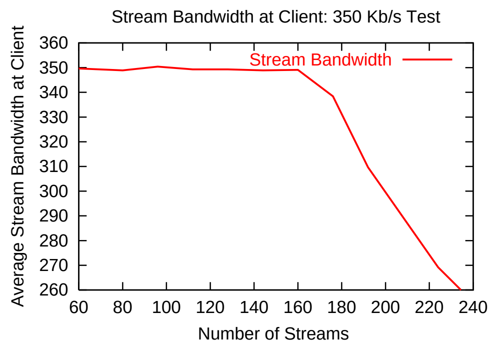

b)

Figure 9: Stream bandwidth under *Unique File Benchmark*: a) 112 Kb/s streams b) 350 Kb/s streams.

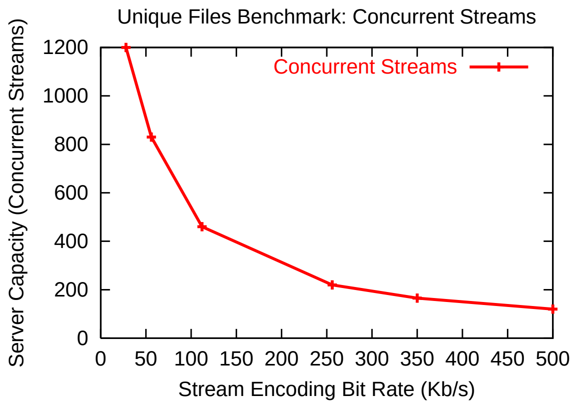
unmounted and mounted back before each test point.

While the maximum achievable media server capacity depends on the encoding bit rate of the files, the performance data collected at the server and the clients are again similar. Using results from the two sub-benchmarks with files encoded at 112 Kb/s and 350 Kb/s, we show more details about server performance, its limits and bottlenecks under the *Unique File Benchmark*.

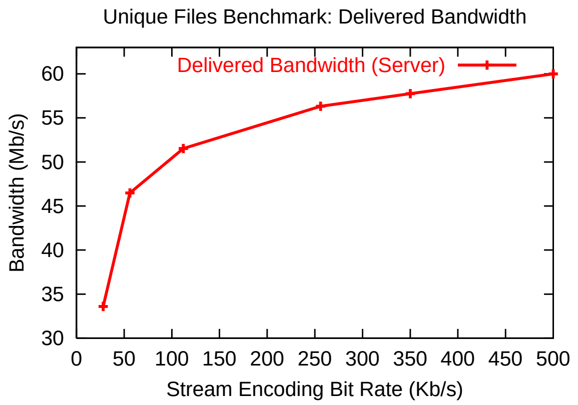
For files encoded at 112 Kb/s, a media server capacity is reached at 460 concurrent streams, and for a file encoded at 350 Kb/s – at 165 concurrent streams. Again, we observe that the media server capacity scales differently than the encoding bit rates of the underlying files: the difference in encoding bit rates between 112 Kb/s file and 350 Kb/s file is 3.125, while the difference in server capacity for corresponding bit rates is only 2.8.

Figures 7 a), b) show the CPU utilization and the application related part of it for corresponding file bit rates. For *Unique File Benchmark*, the CPU utilization of a server is much lower than for *Single File Benchmark*. For all the tests in this study, it is below 45% and it is not a resource which limits server performance. The server performance under *Unique Files Benchmark* is disk-bound: this particular bottleneck is hard to measure with the usual performance tools. The maximum bandwidth delivered by a disk depends on the number of concurrent streams it can support with an acceptable level of *jitter*, i.e. without violating on-time delivery constraints.

Figures 8 a), b) show the overall bandwidth in Mb/s delivered at the server side. It grows linearly, until server capacity is reached, and after that bandwidth delivered by a server flattens. For the *Unique Files Benchmark* and a file encoded at 112 Kb/s, the bandwidth delivered by a server peaks at 51.2 Mb/s (which is equivalent of 6.4 MBytes/s), and for a file en-



a)



b)

Figure 10: *Unique Files Benchmark*: a) Media Server Capacity b) Maximum Delivered Bandwidth.

coded at 350 Kb/s, the maximum delivered bandwidth is 57.8 Mb/s (which is equivalent of 7.2 MBytes/s). Thus, the bandwidth delivered by a server is far lower than the available IO bandwidth provided by four Ultra2 SCSI connections in our experimental setup: each of theUltra2 SCSI connection is officially rated at 80 MBytes/s and is capable of sustaining 60 MBytes/s.

Figures 9 a), b) present complementary information from the client side: once server capacity is reached, the quality of the stream starts to degrade significantly: 10% of extra load causes around 9% of stream bandwidth degradation. The set of graphs in Figures 8, 9 reveals a similar server behavior under overload for the *Unique Files Benchmark* as was observed for the *Single File Benchmark* in Section 3.2 in Figures 3, 4. This reflects that a media server has a typical behavior under overload conditions across different workload types.

Figure 10 a) presents the maximum capacity in concurrent streams achievable by the streaming media server across six different encoding bit rates under *Unique File Benchmark*. Figure 10 b) shows the corresponding maximum bandwidth in Mb/s delivered by the media server for different encoding bit rates. Again, we see much lower bandwidth a server is capable of delivering under the lower encoding bit rates.

Figure 11 shows the normalized graph reflecting the scaling rules for the media server capacity under the *Unique File Benchmark* and different encoding bit rates. In this figure, point (1,1) presents the maximum capacity (120 concurrent streams) achievable by a server when all the clients in the test are accessing the same file encoded at a 500 Kb/s bit rate. Each absolute value for the other encoding bit rate is normalized with respect to it. For example, the maximum capacity achievable by a server under *Unique File File Benchmark* and a 28 Kb/s encoding bit rate is 1200 concurrent streams and is represented by a point (17.9, 10). Figure 11 shows that a media server capacity scales non-linearly comparing to the encoding bit rates of underlying files. For example, the difference between the highest and lowest bit rate of media streams used in our experiments is about 18 times. However, the difference in maximum number of concurrent streams a server is capable of supporting for corresponding bit rates is only 10 times.

Figure 11 shows that the capacity scaling rules under *Unique File Benchmark* are non-trivial and are different from the capacity scaling rules under *Single File Benchmark*. The knowledge of server capacity scaling rules for different bit rates and different workload types is absolutely essential for designing an efficient utility-aware media service and accurate admission control strategies.

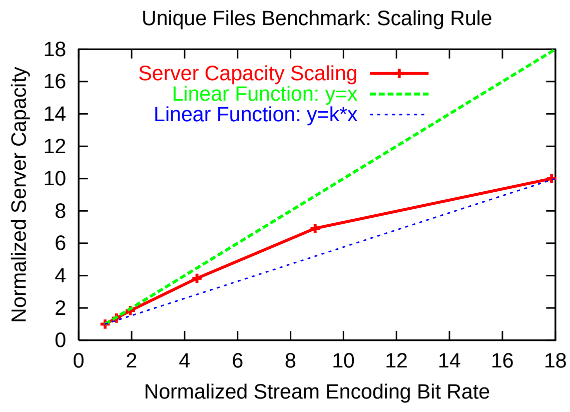


Figure 11: *Unique Files Benchmark*: Server Capacity Scaling Rules.

In summary, under the *Unique Files Benchmark* all the streams read their files from disk. Thus, practically, this benchmark measures a streaming media server capacity when a media content is delivered entirely from the disk. The performance results under such workload type are highly dependent on a choice of a file system and disk configuration. Service providers might use the proposed *Unique Files Benchmark* for performance evaluation of different components available in UDC infrastructure to make the right choices for the best performing system configurations.

Finally, Figure 12 compares achievable media server capacity for *Single* vs *Unique File Benchmark*. The media server performance is 2.5-3 times higher under the *Single File Benchmark* than under the *Unique*

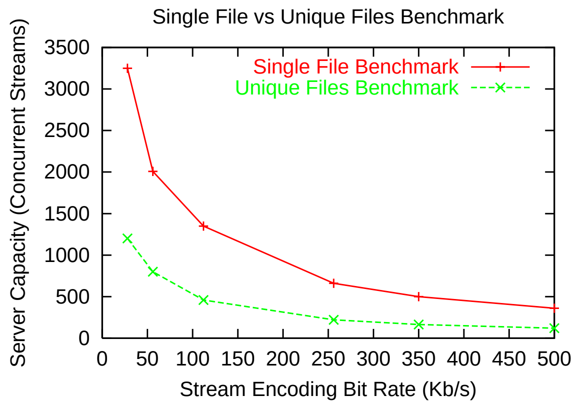


Figure 12: Server Capacity: Single File vs Unique Files Benchmark.

Files Benchmark. These results quantify the performance benefits for multimedia applications when media streams are delivered from memory as well as suggest that much higher performance can be achieved for workloads which exhibit a high degree of shared client accesses to the same media files. The results from media workload analysis [1, 2, 4] show that there is a high degree of shared client accesses to a small subset of media files. Moreover, 50-60% of accesses are due to the client browsing behavior [4], i.e. most of the clients are watching only the initial prefix of the file (less than 2 min long). An opportunity to combine a knowledge of server performance under different workload types with a workload specifics knowledge enables the service providers to correctly size and configure their systems for better support of multimedia applications.

4 Workload-Aware Performance Model of Streaming Media Server Capacity

In Section 3, we showed how to measure the basic capacity of a streaming media server using the set of basic benchmarks. The main goal of this section is to answer a question: how to compute the expected media server capacity for realistic workload if the measured capacities of streaming media server under the basic benchmarks are given.

In this section, we develop a workload-aware server capacity model which is based on a *cost* function derived from the set of basic benchmark measurements. Intuitively, the cost function defines a *fraction* of system resources needed to support a particular stream depending on the encoding bit rate of the corresponding file and access type to a corresponding file.

With each stream delivered by a server we associate an access type:

- *memory access* if a stream retrieves a corresponding file (or the corresponding bytes of the file) from the memory;
- *disk access* if a stream retrieves a corresponding file from the disk.

Additionally, we use the following notations:

- $X = X_1, ..., X_k$ - a set of encoding bit rates of the files used in basic benchmarks,
- $N^{single}_{X_i}$ - the maximum measured server capacity in concurrent streams under the *Single File Benchmark* for a file encoded at X_i Kb/s,
- $N^{unique}_{X_i}$ - the maximum measured server capacity in concurrent streams under *Unique Files Benchmark* for a file encoded at X_i Kb/s,
- $cost^{memory}_{X_i}$ - a value of cost function for a stream with memory access to a file encoded at X_i Kb/s,
- $cost^{disk}_{X_i}$ - a value of cost function for a stream with disk access to a file encoded at X_i Kb/s.

Under the *Unique Files Benchmark*, all the streams have a disk access type, because all the streams read the corresponding files from the disk. Hence each stream requires a fraction of system resources defined by the $cost^{disk}_{X_i}$ value.

Under the *Single File Benchmark*, one stream has a disk access type (it reads the corresponding file from disk, and hence requires a fraction of system resources defined by the $cost^{disk}_{X_i}$ value) while the rest of the streams retrieve a corresponding file from the memory and, therefore, have a memory access type and require a fraction of system resources defined by the $cost^{memory}_{X_i}$ value.

The following *capacity equations* describe the maximum server capacity measured under a set of basic benchmarks for each encoding bit rate $X_i \in X$:

$$N^{unique}_{X_i} \times cost^{disk}_{X_i} = 1$$

$$1 \times cost^{disk}_{X_i} + (N^{single}_{X_i} - 1) \times cost^{memory}_{X_i} = 1$$

By solving the equations above, we can derive the corresponding cost function values:

$$cost^{disk}_{X_i} = \frac{1}{N^{unique}_{X_i}}$$

$$cost^{memory}_{X_i} = \frac{N^{unique}_{X_i} - 1}{N^{unique}_{X_i} \times (N^{single}_{X_i} - 1)}$$

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

- $X_w = X_1, ...X_{k_w}$ - a set of encoding bit rates of the files used in W ($X_w \subseteq X$),
- $N^{memory}_{X_{w_i}}$ - a number of streams having a memory access type to 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 to a subset of files encoded at X_{w_i} Kb/s.

Then the applied *Load* to a media server under workload W can be computed by a formula:

$$Load = \sum_{i=1}^{k_w} N_{X_{w_i}}^{memory} \times cost_{X_{w_i}}^{memory} + \sum_{i=1}^{k_w} N_{X_{w_i}}^{disk} \times cost_{X_{w_i}}^{disk}$$

If $Load \geq 1$ then the media server is overloaded and its capacity is exceeded. The difference $Load - 1$ defines the amount of overload or server exceeded capacity. If $Load \leq 1$ then the media server operates within its capacity, and the difference $1 - Load$ defines the amount of available server capacity.

We validated this performance model of media server by comparing the predicted (computed) and measured media server capacities for a set of different synthetic workloads:

- *single-unique-one-bit-rate*,
- *single-six-bit-rates*,
- *unique-six-bit-rates*.

Under *single-unique-one-bit-rate* workload, all the clients are accessing the files encoded at the same bit rate. Let the encoding bit rate be X_i Kb/s. Let the number of clients accessing the unique files and the number of clients accessing the same (single) file be defined as following:

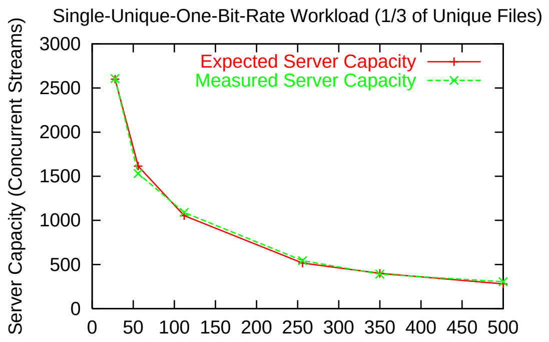
- $\alpha \times N_{X_i}^{unique}$ clients are accessing the unique files, where $\alpha \leq 1$ and $N_{X_i}^{unique}$ is a measured server capacity in concurrent streams under *Unique Files Benchmark* for files encoded at X_i Kb/s,
- $(1-\alpha) \times N_{X_i}^{single}$ clients are accessing the same file where $N_{X_i}^{single}$ is a measured server capacity in concurrent streams under *Single File Benchmark* for a file encoded at X_i Kb/s.

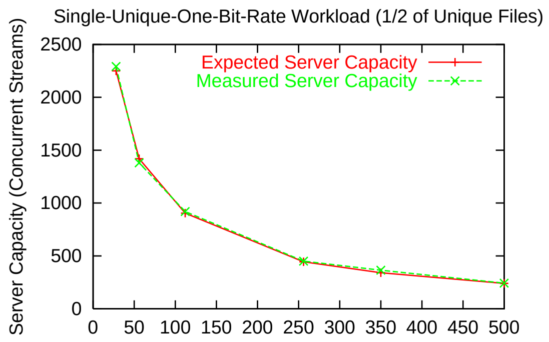
The number of clients (or concurrent streams) in this workload is designed in a special way: if our performance model of server capacity is correct, then under the *single-unique-one-bit-rate* workload the server maximum capacity should be reached. We call it *expected (computed) server capacity*.

Using our experimental testbed, we ran *single-unique-one-bit-rate mix* workload for $\alpha = 1/3, \alpha = 1/2$, and $\alpha = 2/3$ and six bit rates of interest. In the workloads under test, we fixed the number of clients accessing the unique files accordingly to a formula defined above and slightly varied the number of clients accessing the same file to determine experimentally when the server capacity under test is reached using the capacity metrics defined in Section 3.1.

Figure 13 shows the measured vs expected server capacities under *single-unique-one-bit-rate* workload. The measured server capacity closely matches the expected media server capacity across different encoding bit rates and different values of α (the error is within 1-8%).

Under *single-six-bit-rates* workload an equal number of clients are accessing the six single files encoded at six basic bit rates used in the study, i.e. N clients are accessing the same file encoded at 28 Kb/s, N clients are accessing the same file encoded at 56 Kb/s, etc.

a)


b)


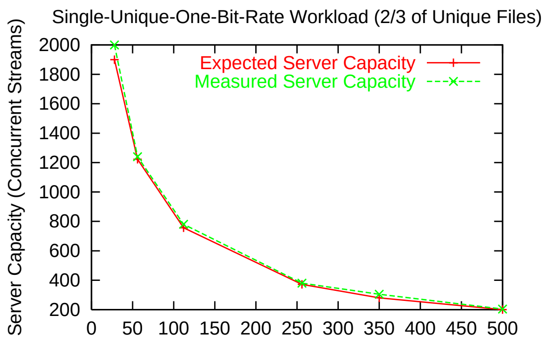
c)


Figure 13: Measured vs Expected Server Capacities for a *single-unique-one-bit-rate* workload: a) $\alpha = 1/3$, b) $\alpha = 1/2$, and c) $\alpha = 2/3$.

Under *unique-six-bit-rates* workload an equal number of clients are accessing the sets of different (unique) iles encoded at six basic bit rates, i.e. N clients are accessing N different files encoded at 28 Kb/s, N clients are accessing N different files encoded at 56 Kb/s, etc.

Figure 14 shows the measured vs expected server capacities under *single-six-bit-rates* and *unique-six-bit-rates* workloads. [3] The measured server capacity matches the expected server capacity very well for both

[3]We used these two workloads to observe: what are the rules for degradation of stream quality under server overload conditions for media files streamed at different encoding bit rates? The preliminary results show that high bit rate streams experience degradation of quality first. We are going to quantify these *QoS fairness rules* in our future work.

workloads (the error is less than 4%).

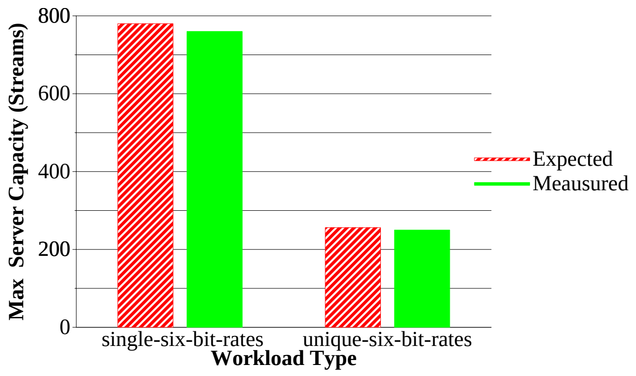


Figure 14: Server Capacity: Measured vs Expected a) *single-unique-one-bit-rate* workload b) *unique-six-bit-rates* workload.

Thus, the performance model of media server capacity based on a *cost* function derived from the set of basic benchmark measurements closely approximates the expected media server capacity for realistic workloads.

5 Related Work

Delivering multimedia content over Internet faces many challenges: an immature broadband Internet infrastructure, the real-time nature of multimedia content and its sensitivity to congestion conditions in the Internet, high backbone transmission cost, a multiplicity of competing standards for media encoding and delivery. 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 (e.g. [7, 12, 9, 20], building caching infrastructure support (e.g. [21, 8, 19]), and designing scalable streaming protocols (e.g. [3, 10, 6, 11, 15, 22].

In [13] a probe-based algorithm for QoS specification and adaptation is proposed to evaluate the resources available (required) on a client and a server side to support certain QoS guarantees for Video-On-Demand service. In later work [14], as a part of QoS-aware resource management for distributed multimedia applications, the resource broker performs admission control using client QoS profile in which the amount of required resources is specified. Such a profile either is prebuilt off-line or created in on-line manner using testing service proposed in [13]. Our work proceeds in a similar direction and proposes a unified framework for measuring a performance of commercially available media server applications. The approach and performance model designed in the paper allow one to estimate the necessary system resources required to support a particular stream without QoS degradation.

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 particular, how to provision server resources for co-hosted services in a way that automatically adapts to the offered load and how to size the amount of used resources to maximize their efficiency and utilization. In [5], authors present *Muse* - a design and implementation of an architecture for resource management in a hosting center with an emphasis on energy as a driving resource management issue for large server clusters. To demonstrate *Muse* functionality and its main principles authors use web server workloads. In paper [16], authors develop a framework for QoS driven dynamic resource allocation in Internet Data Centers. The authors goal is to adaptively migrate servers between the set of hosted applications accordingly to their workload demands.

Web server architecture has evolved to provide performance isolation, service differentiation, and QoS guarantees. Among the basic assumptions in web server cluster management are the assumptions on web server capacities, web server bottlenecks and the requirements on amount of system resources for a particular set of transactions (requests). Web server performance is well studied and understood. A set of commercial benchmarks are proposed for measuring a performance of commercial web servers. 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.

6 Conclusion and Future Work

The Utility Data Center provides a convenient, flexible infrastructure to dynamically support hosting services for Internet applications. Our ultimate goal is to design and implement the utility-aware streaming media service which automatically requests the needed resources from UDC infrastructure in order to adapt to variable workload demand. Among the *prerequisites* to a design of such a utility service is the ability to measure the capacity of media servers (nominal and currently in use) in order to evaluate the amount of available system resources for admitting new client requests or to request/release resources as needed.

A standard commercial stress test used by most of the media server vendors measures a maximum number of concurrent streams delivered by the server when all the clients are accessing the same file encoded at a certain fixed bit rate. We argue, that such test measurements are obviously not sufficient for media server benchmarking. We also argue, that monitoring of a particular system resource can not be used for evaluating of a currently available/used capacity of media service.

In this work, we introduce a set of basic benchmarks to measure the basic capacities of streaming media systems and to reveal the main system/application bottlenecks and properties. The set of basic benchmarks allows one to derive the scaling rules of server capacity for media files encoded at different bit rates and delivered from memory vs disk. Using experimental testbed, we show that these scaling rules are non-trivial. We design a performance model of media server which uses the basic benchmark measurements for estimating a resource requirements of a particular stream in order to compute the server capacity for delivering a realistic workload.

In earlier work [4], we developed *MediaMetrics* – a media workload analysis tool for service providers. In our future work, we are going to combine the proposed media server performance model with workload analysis output from *MediaMetrics* to design the admission control strategies (for a case of sparse resources in UDC) and the request distribution policies in a media server cluster aiming to optimize the overall media system performance.

Acknowledgements: Authors would like to thank Wenting Tang and Yun Fu for stimulating discussions, Nic Lyons, Brett Bausk, and Jeff Krenek for their help in media content creation, and John Sontag for his active support of this work.

References

[1] S. Acharya, B. Smith, P. Parnes. Characterizing User Access to Videos on the World Wide Web. In Proc. of ACM/SPIE Multimedia Computing and Networking. San Jose, CA, January 2000.

[2] Almeida, J. M., J. Krueger, D. L. Eager, and M. K. Vernon. Analysis of Educational Media Server Workloads, Proc. 11th Int'l. Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV 2001), June 2001.

[3] S. Carter, D. Long. Improving Video-on-demand Server Efficiency Through Stream Tapping. Int'l Conf. on Computer Communications and Networks, 1997.

[4] L. Cherkasova, M. Gupta. Characterizing Locality, Evolution, and Life Span of Accesses in Enterprise Media Server Workloads. 12th Int'l. Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV 2002), May 2002.

[5] J. Chase, D. Anderson, P. Thakar, A. Vahdat, and R. Doyle. Managing Energy and Server Resources in Hosting Centers. Jeffrey Proceedings of the Eighteenth ACM Symposium on Operating Systems Principles (SOSP), October 2001.

[6] L. Gao, J. Kurose, and D. Towsley. Catching and Selective Catching: Efficient Latency Reduction Techniques for Delivering Continuous Multimedia Streams. Proc. ACM Multimedia'99, Orlando, FL, Nov.1999.

[7] A. Dan, S. Dias, R. Mukherjee, D. Sitram, and R. Tewari. Buffering and caching in large scale video servers. Proc. IEEE Compcon, 1995.

[8] D. Eager, M. Ferris, and M. Vernon. Optimized Regional caching for on Demand Data Delivery. In Proc. Multimedia Computing and Networking, (MMCN'99), San Jose, CA, January, 1999.

[9] D. Eager, M. Vernon, and J. Zahorjan. Optimal and Efficient Merging Schedules for Video-on-Demand Servers. Proc. 7th ACM Multimedia Conf. (Multimedia '99), Orlando, Nov., 1999.

[10] D. Eager, M. Vernon, and J. Zahorjan. Bandwidth Skimming: A Technique for Cost-Effective Video-on-Demand. In Proc. Multimedia Computing and Networking 2000 (MMCN'00), San Jose, CA, Jan., 2000.

[11] K. Hua, Y. Cai, S. Sheu. Patching: A Multicast Technique for True Video-on-demand Services. Proc. ACM Multimedia'98, Bristol, U.K., Sep.1998.

[12] I. Kim, J. Kim, J. Lee, K. Chung. Video Data Scheduling using Window-Based Prefetching. Proc. IEEE Multimedia Computing and Systems, Florence, Italy, June, 1999.

[13] K. Nahrstedt, A. Hossain, S.-M. Kang. A Probe-based Algorithm for QoS Specification and Adaptation, in Proc. of 4th IFIP International Workshop on QoS (IWQoS'96), Paris, France, March, 1996.

[14] K. Nahrstedt, H. Chu, S. Narayan. QoS-aware Resource Management for Distributed Multimedia Applications, Journal on High-Speed Networking, Special Issue on Multimedia Networking, vol. 8, num. 3-4, IOS Press, 1998.

[15] J-F. Paris, S.W.Carter, and D.D. Long. A Hybrid Broadcasting Protocol for Video-on demand. In Proc. Multimedia Computing and Networking 1999, San Jose, CA, January, 1999.

[16] S. Ranjan, J. Rolia, H. Fu, E. Knightly. QoS-Driven Server Migration for Internet Data Centers. Proc. of ACM/IEEE Int'l Workshop on Quality of Service (IWQoS), Miami Beach, FL, May 2002.

[17] Realsystem Server Product Line Comparison. `http://www.realnetworks.com/products/servers/comparison.html`

[18] RealProducer G2 from RealNetworks. `http://www.realnetworks.com/products/producer/index.html`

[19] R. Rejaie, M. Handley, H. Yu, and D. Estrin. Proxy Caching Mechanism for Multimedia Playback Streams in the Internet. Proceedings of the 4th International Web Caching Workshop , San Diego, CA., March 1999.

[20] P. Shenoy, S. Hasan, P. Kulkarni, and K. Ramamritham. Middleware versus Native OS Support: Architectural Considerations for Supporting Multimedia Applications. Proc. IEEE Real-time Technology and Applications Symposium (RTAS'02), San Jose, CA, September 2002.

[21] S. Sen, J. Rexford, and D. Towsley. Proxy Prefix Caching for Multimedia Streams. In Proc. IEEE INFOCOM, April, 1999.

[22] H. Tan, H., D. L. Eager, M. K. Vernon, and H. Guo. Quality of Service Evaluations of Multicast Streaming Protocols. Proc. ACM SIGMETRICS 2002 Int'l. Conf. on Measurement and Modeling of Computer Systems (SIGMETRICS 2002), Marina del Rey, CA, June 2002.

[23] Utility Data Center: Solutions. `http://www.hp.com/solutions1/infrastructure/solutions/utilitydata/index.html`

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