Measuring the Capacity of a Streaming Media Server in a Utility Data Center Environment

Ludmila Cherkasova, Loren Staley
Internet Systems and Storage Laboratory
HP Laboratories Palo Alto
HPL-2003-6
January 9th , 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

In order to design a "utility-aware" streaming media service which automatically requests the necessary resources from Utility Data Center infrastructure, 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? In this paper, we propose a set of benchmarks for measuring the basic capacities of streaming media systems for different expected workloads, and demonstrate the results using an experimental testbed.

* Internal Accession Date Only Approved for External Publication
Published in the 10th Annual ACM International Conference on Multimedia, 1-6 December 2002, Juan Les Pines, France

Measuring the Capacity of a Streaming Media Server in a 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 *In order to design a "utility-aware" streaming media service which automatically requests the necessary resources from Utility Data Center infrastructure, 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? In this paper, we propose a set of benchmarks for measuring the basic capacities of streaming media systems for different expected workloads, and demonstrate the results using an experimental testbed.*

Categories and Subject Descriptors: *H.5.1 Multimedia Information Systems.*
General Terms: *Measurement, Management, Performance, Design, Human Factors.*
Keywords: *Utility Data Centers, enterprise media servers, media system benchmarks, measurements, media server capacity.*

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. 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.

The evidence from media workload analysis [1, 2, 3] indicates that client demands are highly variable: some days (hours) exhibit a magnitude 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 the media applications a perfect candidate to benefit from flexible resource management and resource provisioning in Utility Data Center infrastructure.

Utility Data Center (UDC) [6, 7] is programmable data center that creates and runs virtual IT environments as a highly automated service. UDC provides a flexible, cost-effective solution with advanced management software allowing resources to be automatically reassigned in response to changing business and IT requirements. UDC architecture is an ideal platform to support the efficient hosting services for Internet applications. 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.

In order to design and implement a "utility-aware" streaming media service which automatically requests from UDC infrastructure the resources, several classical 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 media server?

Commercial media server solutions are distinguished by the number of concurrent streams a server can support [4] 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 the 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 performance and main bottlenecks of media server:

- *Single File Benchmark* and
- *Unique Files Benchmark*

Using proposed set of basic benchmarks, the service providers could measure the capacities of their systems. These benchmarks provide the *basic performance envelope* for sizing streaming media systems under expected workload and lay down a performance foundation for "utility-aware" service design.

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 64 KB; and 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[4].

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 4, we will show that achievable

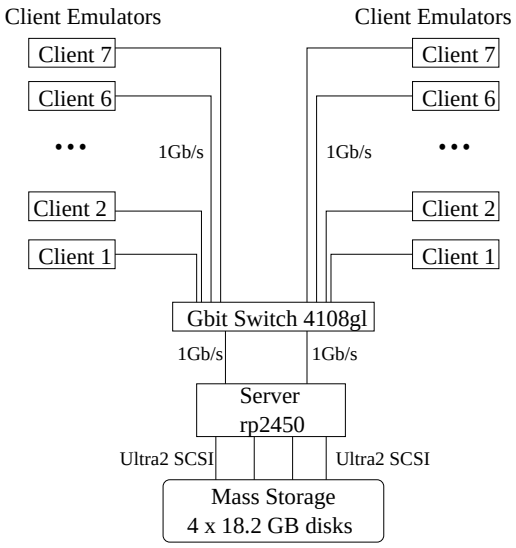


Figure 1: Experimental setup.

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.

3 Server Side and Client Side Capacity Metrics

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 ca-

[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.

pacity 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:

- 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.

[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.

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.

4 Single File and Unique Files Benchmarks

In this Section, we introduce a set of *basic benchmarks* aimed at determining the performance limits of media server and analyzing its main performance bottlenecks.

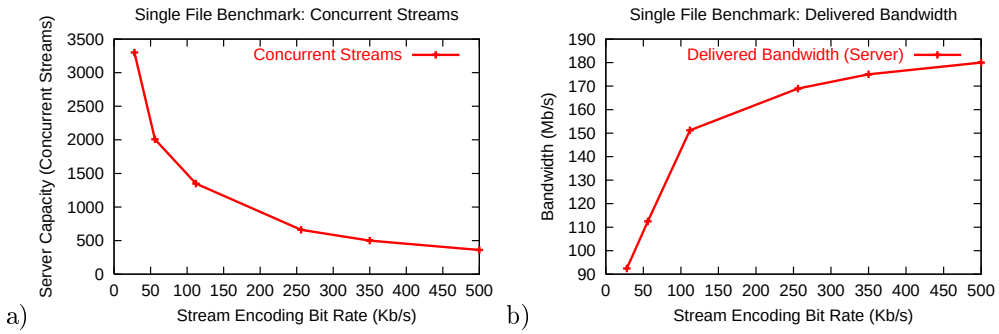
Both of the benchmarks introduced below use specially created 20 min video clips encoded at different bit rates with RealProducer G2 [5] from RealNetworks. The following six bit rates were used for typical target audience:

- 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 the media server without degrading a quality of streams.

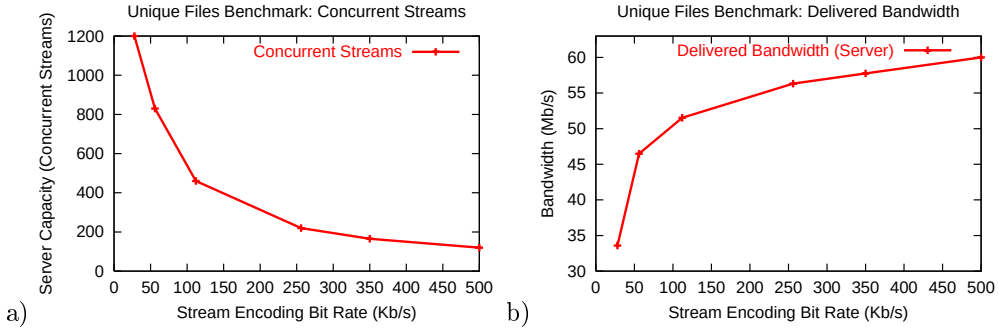
We introduce a set of *basic benchmarks* to measure the basic performance limits and main bottlenecks of media server:

- *Single File Benchmark* measuring a media server capacity when all the clients in the test are accessing the same file;
- *Unique Files Benchmark* measuring a media server capacity when each client in the test is accessing a different (unique) file.



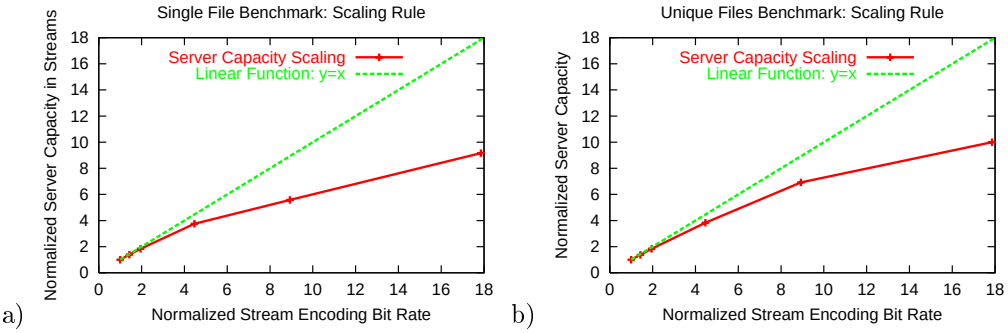
a) b)

Figure 2: *Single File Benchmark*: a) Media Server Capacity b) Maximum Delivered Bandwidth.



a) b)

Figure 3: *Unique Files Benchmark*: a) Media Server Capacity b) Maximum Delivered Bandwidth.



a) b)

Figure 4: Server Capacity Scaling Rules: a) *Single File Benchmark* b) *Unique Files Benchmark.*

We designed a completely automatic benchmark which runs a sequence of tests with an increasing number of clients for each of the six encoding bit rates.

During each test, the following performance data and measuremence are collected: *ServerStats*, *ClientStats*, and a general performance information of a server and client machines using *vmstat*.

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

Under the *Single File Benchmark*, the media server is CPU bounded: CPU reaches 100% of utilization, and it is the main resource which limits server performance. In essence, 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 RAM. However, this benchmark does not require that the streamed media file completely resides or fits in the RAM (the file can be larger than available RAM). In essence, this benchmark exercises the shared access by multiple clients to the same file. Figure 2 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 2 b) shows the

corresponding maximum bandwidth in Mb/s delivered by the media server for different encoding bit rates.

For *Unique File Benchmark*, the CPU utilization 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. Under the *Unique Files Benchmark*, the server performance 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. Figure 3 a) presents the maximum capacity in concurrent streams achievable by the streaming media server across six different encoding bit rates under the *Unique Files Benchmark*. Figure 3 b) shows the corresponding maximum bandwidth in Mb/s delivered by the media server for different encoding bit rates.

Comparing the results from Figure 2 and Figure 3, the media server performance is 2.5-3 times higher under the *Single File Benchmark* than under the *Unique 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.

Figure 4 a) shows the normalized graph reflecting the scaling rules for the media server under *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 4 b) shows similar normalized graph reflecting the scaling rules for the media server under *Unique Files 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 Files Benchmark* and a 28 Kb/s encoding bit rate is 1200 concurrent streams and is represented by a point (17.9, 10).

Figure 4 reflects that the 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 for *Single File Benchmark* and 10 times for *Unique Files Benchmark*.

Using proposed set of basic benchmarks, the service providers could measure the capacities of their systems for different expected workloads. These benchmarks provide the *basic performance envelope* for sizing and scaling streaming media systems under expected application workload. Set of proposed performance measurements allow to automatically evaluate the streaming media infrastructure in Utility Data Center environment and to provide measured system parameters for "utility-aware" service design.

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. Proc. of ACM/SPIE Multimedia Computing and Networking, 2000.

[2] Almeida, J. M., J. Krueger, D. L. Eager, 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, June 2001.

[3] 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 (ACM NOSSDAV 2002), May 2002.

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

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

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

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