EFFECTS OF LOCALITY, CONTENT AND JAVA RUNTIME ON VIDEO PERFORMANCE

VIKRAM CHHABRA, AKSHAY KOTHARE, MARK CLAYPOOL
Computer Science Department
Worcester Polytechnic Institute
100 Institute Road, Worcester, MA 01609, USA
{vikram, akshay, claypool}@cs.wpi.edu

ABSTRACT

Java has been growing tremendously as a language and as a platform. Multimedia applications written in Java are gaining popularity because of the potential for platform independence. However, to date there is a need for research to determine how multimedia performance is affected in Java, considering both characteristics of the media and Java runtime parameters. In this paper, we present experiments that measure the multimedia performance of an MPEG client in Java, varying video proximity, movie content and Just-In-Time compilation. We find that local media access has better frame rates and less jitter compared to access to media over a network. Different movie contents show significantly different amounts of jitter and frame rates. Although, overall Java lags behind C++ as far as multimedia performance is concerned, we believe that further improvement in performance can be achieved in Java through the use of runtime features such as Just in Time Compilation.
Keywords: Java, MPEG, Video, Performance.

1. INTRODUCTION

The increase in the bandwidth and connectivity of the Internet has enabled video to the desktop. *Frame Rate*, *Delay* and *Jitter* are critical measures of performance when it comes to delivering real-time video over the Internet. *Frame Rate* is the number of frames displayed per second. *Delay* is the time taken for a frame to be captured at the server until it is played at the client and *Jitter* is the variance in this delay. These parameters influence the quality of video output that the client receives. A substantial amount of delay, either from the network or from processing, between each frame can cause the video to play at a slower frame rate than its encoded frame rate and hence appear disturbing [1]. In the presence of jitter the client would see pictures either having a few frozen frames or see jumping of frames, in order to preserve the timing for its subsequent frames.

Java, being an interpreted language, lags in performance as compared to C++ [2]. However it manages to score over C++ in the areas of platform independence and security. The "write once - run anywhere" nature of Java bytecode makes it attractive for developing multimedia-streaming applications. Before a Java application can be executed, its source code needs to be compiled into bytecode. *Bytecode* is the object code that is processed by the Java Virtual Machine (JVM). The *virtual machine* acts as an interface between the compiled Java code and the underlying hardware platform. A *Just in Time* (JIT) compiler translates the bytecode into machine code just before it is to be used and caches the machine code in memory for reuse. Not many experiments have been performed to evaluate the performance of Java for streaming video. These included toggling JIT and garbage collection features of the JVM, playback of the MPEG file using different system setups such as different processors, comparison with C++ etc. These works point out that JIT, local access of the file and the processing power of the client mostly influence the performance [3].

Despite these previous results, there are a still many parameters that need to be tested to evaluate the performance of Java for streaming multimedia. In this work, we evaluate different network setups in which the distance between the client and server is varied. We find out how increasing the distance between the site from where the video is being served and from where it is being displayed affect the jitter and frame rate associated with multimedia streaming applications. We also evaluate different types of MPEG movie content, varying from low to high motion. Here we demonstrate that content being served also has an effect on the frame rate and jitter. Finally, we evaluate toggling JIT on and off and comparing Java with C++. We find JIT has an enormous performance difference. The rest of the paper is laid out as follows: Section 2 describes our experiments to measure Java performance; Section 3 presents the results of the experiments; Section 4 draws conclusions from these results and Section 5 outlines possible future work.

2. OUR APPROACH

To evaluate the performance of Java for streaming video, We designed a client-server using the TCP/IP protocol [3]. Although the client was written in Java [4], the server was written in C++ to avoid any performance degradation on the server side. The server separates an MPEG-1 layer file into frames and then sends these frames over the network using TCP/IP protocol. The server reads the MPEG file until it reaches a frame delimiter and then sends it to the client. The client receives this TCP stream and displays it on the local terminal. The client records the: start decompress time, stop decompress time, start display time and stop display time of the frame to be played out, in milliseconds. Server runs on both Digital UNIX and Linux. The client being written in Java, runs unchanged for Windows NT and Linux.

We tested three parameters that influence the performance of video in Java. First, we increased the distance between the client and the server, which would help us account for network traffic. Next, we varied the movie content from static to dynamic to examine how the content affects the frame rate and jitter. Finally, we compared Java with its JIT toggled ON and OFF with C. We believe that compilers perform their own optimizations to the source code, which in turn can lead to significant performance enhancements and thus in turn affect the frame rate and jitter, depending on whether the machine code is being cached or not or whether it is being interpreted which again has its own performance issues.

The following subsections describe these parameters in detail:

2.1 NETWORK

To perform the tests by varying the network setup, we used two dedicated Intel based machines and installed SUSE Linux 6.3 kernel version 2.2.13 on both of them, one to run the video server and the other the video client. The server was a Pentium MMX 233 MHz PC with 64MB of RAM and the client was a Pentium II 300 MHz PC with 128MB of RAM. The two machines had Etherlink Network Cards, which were connected to the network using a broadband Coaxial Cable at 10Mbps. Our performance measuring parameters were the *frame rate* and *jitter*. Since our goal is to find out the effect of increasing/decreasing the distance between the client and the server on performance, for each test the client was run on the same Linux machine and at every step the server was moved further away from the client:

" Local Playback " Local playback of the MPEG Video file from IDE hard disk using our client (written in Java) on our Linux box.
" Dedicated Server-Client Setup " Run the server and the client on our two Linux boxes connected to each other via a coaxial cable but disconnected from the Internet
" Normal Setup " Run the client and server on our two Linux boxes which are connected to the Internet
" Different Switch Setup " Run the client on our Linux box but run the server on a machine which cross a switch (Different departments of WPI)

2.2 MOVIE CONTENT

We next tested the effect of the content of the MPEG movie on the frame rate and jitter. Video clips could vary from static to dynamic. Static videos are like news clips or interviews, where there is less movement between successive frames. On the other hand, dynamic videos are fast changing video sequences, such as action or sports scenes. We used a WinNT machine running on Intel Pentium MMX 233 MHz PC with 64MB of RAM as our Java client. The server was a SuSE Linux 6.4 (i386) - Kernel version 2.2.14 machine running on an Intel PIII 500Mhz PC with 128 MB RAM. Both machines were connected to the network via 100 Mbps Ethernet. Table I describes the movies that were made or re-encoded for this test. All movies were 120 frames with a resolution of 200x200 pixels, the only difference between them was the visual content and hence the size and compression rate. Each movie was encoded in two frame formats, the first one containing only I frames and the second one using an IBBPBBPBBPBBPBB group of pictures.

Movie	***Size (Kb)***	***Classification***	***I frames***	***P frames***	***B frames***
blue-i	*66*	Static - Same image (blue screen) in all the frames.	*120*	*0*	*0*
blue-ibp4	*17*		*9*	*32*	*79*
cnn-i	*837*	News Clip - Only slight change associated with one object in adjacent frames.	*120*	*0*	*0*
cnn-ibp4	*161*		*9*	*32*	*79*
kick-i	*1024*	Motion Picture – Slightly more changes in adjacent frames compared to the news clip.	*120*	*0*	*0*
kick-ibp4	*320*		*9*	*32*	*79*
ski-i	*726*	Motion Sport - Quite a lot of difference between adjacent frames compared to the motion picture.	*120*	*0*	*0*
ski-ibp4	*447*		*9*	*32*	*79*
mixbag-i	*1229*	Extreme - Totally different adjacent frames made from random pictures.	*120*	*0*	*0*
mixbag-ibp4	*1434*		*9*	*32*	*79*

Table I : Description of various movies which were used to perform the movie content test

2.3 JUST IN TIME (JIT) COMPILATION

Our final test was to toggle the JIT ON and OFF. To perform this test, we used the same setup as we did for the network test with local playback. We used Shudo Kazuyuki's *ShuJIT* [5] for the Linux version of JDK 1.2.2. We noticed a substantial performance increase with the JIT turned on. We compared these results with a C player (mpeg_play[1]). We shall discuss this in detail in the next section.

3. RESULTS

Frame rate and jitter were our parameters for measuring the performance of Java. The frame rate was measured as the inverse of the average of display time of the frames. Jitter was measured as the standard deviation in the inter-frame display time. Frame rate gives a wide perspective of the performance whereas jitter helps to understand details such as effects of frame type or network congestion. Our results are presented in the following subsections:

3.1 NETWORK

Our first set of tests shows that the access to local media is faster than access to media over the network. Fig. I show these results. Increasing the distance between the client and server over the network does not seem to significantly affect frame rate. It did not show any significant difference or pattern. Like the frame rate, jitter follows the same pattern. It is less for local access and more for network access and does not vary much when tested for different network setups, as seen in Fig. II. Fig. III shows the jitter, as in the frame inter-arrival delay, when the MPEG file is accessed under different types of network settings.

3.2 MOVIE CONTENT

Our next results explore whether or not the content of MPEG file being served has any effects on the frame rate.

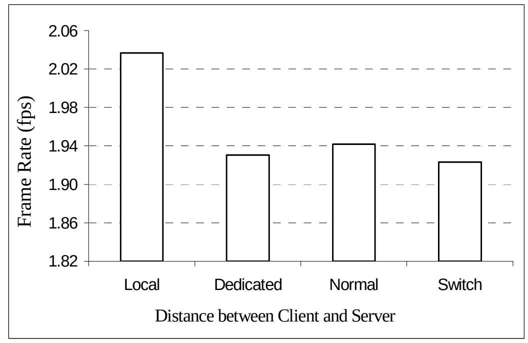

Figure I : Frame rate for varying the distance between the client and server.

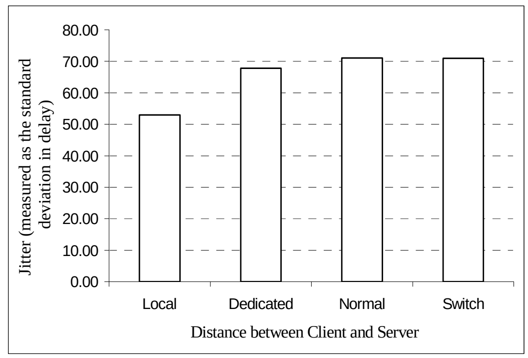

Figure II : Jitter for varying the distance between the client and server.

The tests were performed for two different frame formats. The IBBPBBPBBPBBPBB frame encoded movies show a linear decrease in frame rate from the static movie to the dynamic movie, as seen in Fig. IV. But the I-frame movies show a sudden drop in the frame rate from the Blue movie to the CNN movie, and remains mostly same for CNN, Kick and Ski. The frame rate again drops for the Mixbag movie. Jitter also shows a pattern.

The static movies have lesser jitter and the dynamic movies have a higher jitter. This jitter pattern is seen for both frame encoding types. Clearly, both tests indicate that video content does matter.

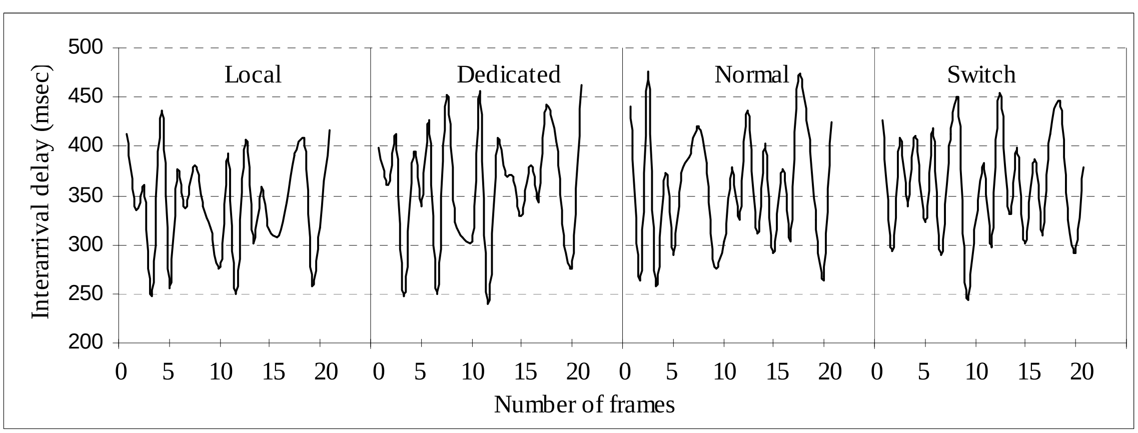

Figure III : Frame inter-arrival delay for the MPEG file under different network setups

[1] http://bmrc.berkeley.edu/frame/research/mpeg/index.html

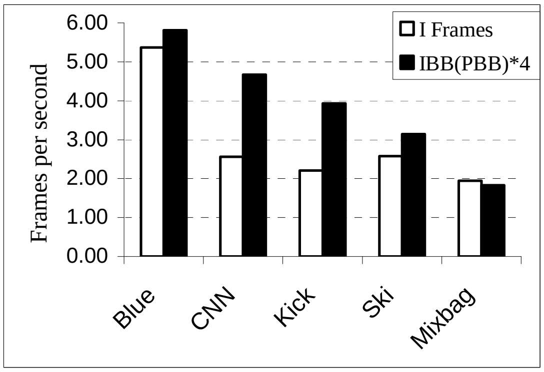

Figure IV : Frame rates for visually different movies, varying from static to dynamic

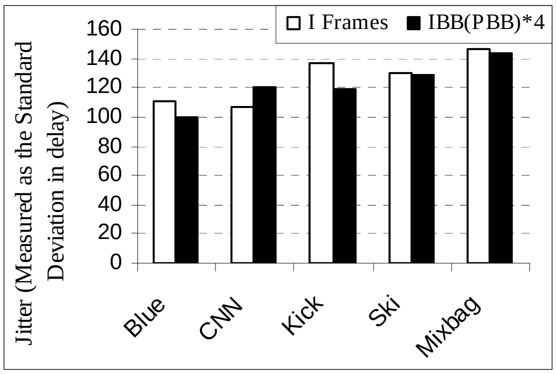

Figure V : Jitter for visually different movies, varying from static to dynamic

3.3 JIT COMPILATION

Our final parameter was to see the effect of JIT compilation and compare the results with C. Fig. VI shows the performance difference for the same MPEG file with JIT turned off and on, when played locally. The same file when played with mpeg_play gave a significantly higher frame rate. The jitter is lower with JIT turned on as seen in Fig. VII. Hence JIT not only gives a better frame rate, but lowers the jitter too, so a JIT definitely boosts the performance of video in Java.

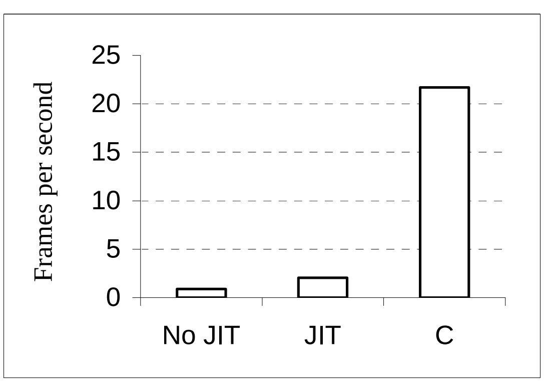

Figure VI : Frame rate comparison for different compilers, comparing Java (JIT on and off) with C

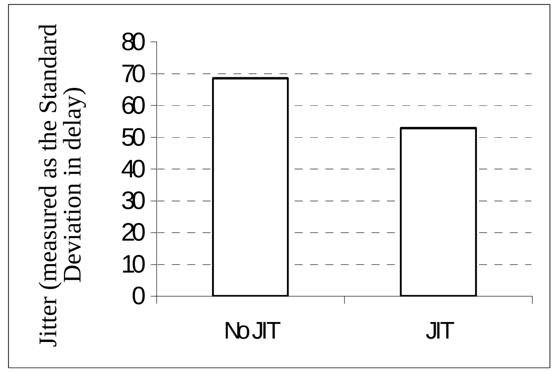

Figure VII : Jitter comparison with JIT on and off

4. CONCLUSIONS

Java has gained popularity because of its inherent features such as platform independence, garbage collection, etc. However, it still lags in performance as compared to C/C++. Not much research has been done to evaluate Java for streaming multimedia applications. In this paper, we have attempted to find the parameters that influence the performance of Java for streaming video over the network. We have analyzed the various network setups, movie content and Java's JIT compiler and observed the effect of these parameters on frame rate and jitter in the play out of multimedia. Based on our results, as described in the previous section, we deduce the following conclusions.

The local access of the media versus access over the network shows 5% improvement for the frame rate and 25% less jitter. But there is neither a noticeable difference between the various network setups nor could we observe any pattern from the number of tests we performed. It could be possible that we may need to perform these tests over a large number of times to observe some pattern, if at all such a pattern exists.

Our second set of tests show that movie content has significant implications on the performance. The frame rate of the Blue movie is almost three fold higher than that of the Mixbag movie. The only difference in these movies is the content, due to which the size and the compression rate is different. Jitter is also 40% higher for the Mixbag movie as compared to the Blue movie. The other three movies lie linearly between the Blue and the Mixbag. We hypothesize that this frame rate pattern is because the size of the movie affects the frame rate. To confirm this hypothesis, future work suggests, take the same content, encode it at different quality levels, via a mpeg_encode configure option, and test if the frame-rate is directly proportional.

Our final set of tests of toggling JIT ON and OFF shows a significant effect on performance. There is 100% improvement in the frame rate and 25% decrease in jitter with JIT on. The results were consistent for all different type of network tests and different MPEG files too. We can safely conclude that JIT significantly improves the performance of our Java MPEG player with a better frame rate and lower jitter. But Java even with its JIT compilation lags the power of C. The frame rate with the C player was 10 times faster than Java with JIT ON.

5. FUTURE WORK

We have tested some of the possible parameters to evaluate Java's performance. There are many more unexplored areas. Firstly there should be a code evaluation for possible optimization for the client and the server, which is necessary for certainty in our results. Then, we would like to try different network protocols such as using UDP, RMI or CORBA instead of TCP, as to determine if TCP is responsible for the jitter observed and if other protocols could help to reduce the jitter and improve performance. We would also like to try our experiments for different network buffer sizes for the client. Other research can discover any relationship between frame type and jitter. For example, it could be possible that jitter is more for a certain frame type. As mentioned in the previous section, we would like to see if the movie size is the only criterion for the pattern seen in our second set of test for movie content. Trying different JITs (such as JBuilder, TYA etc.) is another possible area for future work.

We believe that our current work, along with the previous tests and these proposed tests will help us to understand and evaluate the performance of Java for streaming multimedia and hence will unfold various ways to improve its performance.

6. ACKNOWLEDGEMENTS

We would like to thank Tom Coates, Shawn Hooley, Eric Shea and Chris Spellacy for letting us use their client/server code to perform these tests. We would also like to thank Badrinath Venkatachari and Harsh Vahalia for their help in managing the hardware setup.

REFERENCES

[1] Mark Claypool, Jonathan Tanner. The effects of jitter on the perpetual quality of video. *ACM Multimedia Conference*, Volume 2, October 30 - November 5, 1999.

[2] Tom R. Halfhill. *Byte Magazine - How to soup up Java Part I* (New York: The McGraw-Hill Companies Inc., May 1998).

[3] Mark Claypool, Tom Coates, Shawn Hooley, Eric Shea and Chris Spellacy. Video Performace in Java. *IRMA 2000 Multimedia Computing (MMC) Track*. May 21-24, 2000

[4] C. Hasan. *MPEG-1 Video Stream Decoder Applet* 1998 online at [http://www.dcc.uchile.cl/~chasan/MPEGPlayer.zip]

[5] JIT for JDK 1.2.2 for Linux, online at [http://www.shudo.net/jit]

[6] G. Ghineas and J. Thomas. QoS impact on user perception and understanding of multimedia video clips. *ACM Multimedia*. Sept 1998.

[7] LeGall, Didier, MPEG - A Video Compression Standard for Multimedia Applications, *Communications of ACM*, Vol.34, No.4, April 1991.

[8] JDK 1.2.2 for Windows and Linux, online at [http://java.sun.com]