Characterizing and Enhancing the Performance of Sound Synthesis Applications on Pentium III

Course Project Proposal
CMU CS 740: Computer Architecture, Fall 2002
Prof. Seth Goldstein
Ning Hu (ninghu@cs.cmu.edu) and Vahe Poladian (poladian@cs.cmu.edui)

Project web page: http://www.cs.cmu.edu/~poladian/arch/project

Motivation

Sound synthesis applications combine waveform tables and interpolation instead of actual computation involving sums of sinusoids. A large waveform table reduces the need for interpolation. On the other hand, a smaller table provides better cache hit rate, and combined with quality interpolation, can provide an acceptable noise rate. Linear interpolation and quadratic interpolation provide exceedingly lower noise rates at the exceedingly higher computational cost. It is therefore desirable to understand the performance characteristics of sound synthesis applications and explore possible improvements to the performance of such applications by possibly using multimedia extensions to modern architectures.

The Problem

A sound synthesis application processes a stream of data in a loop similar to this:

```
10      while ( stream )
20      {
30              phase += pinc;      //  pinc comes in the stream
40              phase = phase MOD N //  implemented in a loop, expensive
50              i = (long) phase;   //  get the integer approx of phase
60              f = p - i;          //   the decimal part
70              s1 = table[i];      //  do lookups
80              s2 = table[i+1];
90              val = ( s1 * (1-f) + s2 * f) * amp;    //  interpolate the value
100             amp += ainc;        //  increment amplitude
110     }
```

The mod operation at line 40, floating point to integer conversion at line 50, and interpolation at line 90 are computationally expensive (line 90 shows linear interpolation, and quadratic is more expensive). Further, floating point multiplies in the interpolation must wait for the outcome of the corresponding table lookups, and table lookups must wait for the conversion. We believe that this chain of dependencies may leave the CPU underutilized. Another bottleneck may be high rate of cache misses.

Characterizing and Optimizing the Performance

In our initial study, we would like to understand better the bottlenecks in the above-described program. We would like to experiment with various table sizes, and

various level of interpolation quality (e.g., none, linear, quadratic), understand the impact of such parameters on the performance.

If our initial hypotheses are true, then CPU is not the bottleneck, and we will attempt to make performance improvements by using simple techniques such as instruction re-ordering and loop unrolling. We are hoping that these simple techniques will eliminate the effects of the critical path, and make the computation CPU bound.

At that stage, we will explore the possibility of using SSE extensions to the Pentium instruction set to further explore any possible performance enhancements. A great deal of research has been done to understand the impact of using MMX / SSE extensions in applications such as image and sound decoding [cite the Usability of MMX / SSE instruction sets to JPEG decoding paper]. We think that sound synthesis applications potentially benefit from such enhancements.

Schedule

There are 6 weeks available till the deadline (excluding Thanksgiving, and allowing for some schedule slippage). We will distribute the available time as follows:

Weeks 1 and 2:

1.1. Install, learn to use V-Tune, and Visual Studio Profiler,
1.2. Learn how to interpret V-Tune data,
1.3. Obtain and modify music synthesis code,
1.4. Run experiments, take measurements, interpret data,

Weeks 3 and 4:

2.1. Interpret the results of the initial experiments,
2.2. Optimize (non-SSE),
2.3. Run experiments again,
Iterate the above, if needed.

In parallel, pursue the following thread:

2.4. Learn SSE,
2.5. Install, learn Quexal,
2.6. Start writing SSE-enhanced code for the algorithm,

In a separate section, we will report what we plan to have achieved by milestone.

Week 5:

5.1. Continue optimizing the SSE-enhanced version,
5.2. Run experiments,
5.3. Interpret,
Possibly iterate over the above 3.
5.4. Begin analysis of the findings,

Week 6:

6.1. Finalize analysis
6.2. Summarize experiment results,
6.3. Write-up report,
6.4. Prepare poster,

Milestone Deliverable

We plan to report findings from the initial experiments. This will help us identify bottlenecks in un-enhanced code. Report results from the initial optimizations (quick-hits). Report preliminary progress on SSE-enhancement of the code.

Resources

Pentium III workstations – available,
Source code to be analyzed – available,
Tools:

Performance analysis: V-Tune, Visual Studio Profiler, available,
Development: Quexal – need to be procured,

We may need guidance on learning how to use the performance tool and analyze results. We need to speak with faculty in signal processing to better interpret data as we get it. We hope we can get help from Architecture faculty and students, if needed. Also, we hope to leverage on the experience of the past projects, who have done similar analysis.

Work Done so Far

1. Spoken to Prof. Roger Dannenberg to understand problem better,
2. Gotten sign-off from Prof. Goldstein,
3. Read significant number of past projects on similar topics. This helped us identify tools and get an idea of plan of attack,
4. Read literature on using waveform tables for a music synthesizer,
5. Searched for tools and documentation on SSE and performance analysis,

Related Literature

Moore, F. R. 1977. Table lookup noise for sinusoidal digital oscillators. Computer Music Journal 1(2), pp26-29. Reprinted in C. Roads and J. Strawn, eds. 1985. Foundations of Computer Music. MIT Press. pp. 326-334.

Dannenberg, ``Interpolation Error in Waveform Table Lookup," in Proceedings of the 1998 International Computer Music Conference, (1998), pp 240-243.

Barbic, Rosencrantz, Potetz: Investigating The Utility of MMX/SSE Instruction Sets Now And In The Future. A Course Project for CMU CS-740, Fall 2001.