



Multiprocessing in SONAR 3.1

Cakewalk Technology White Paper

March 2004

Ron Kuper, VP/Engineering, Cakewalk

Key Points

Cakewalk audio software has been multithreaded since 1995. The benefit to users has been smoother performance on a single-CPU system. A multithreaded design also means faster performance on a multiple-CPU system, because threads can run simultaneously on different CPUs.

SONAR 3.1 has been redesigned to maximize multithreading benefits. Work is allocated more evenly among different threads and among multiple CPUs. This provides users for SONAR 3.1 with even smoother and faster performance.

Multiprocessing is moving into the mainstream. Intel's Hyper-Threading Technology (HT Technology) lets a single-CPU system act more like a multi-CPU system. As a result, the performance benefits of multithreading will continue to become available to many more customers, not just the lucky few with expensive multi-CPU systems.

Introduction

Cakewalk's music workstation products have utilized multithreading since Cakewalk Pro Audio 4.0. This design allows the system to balance its workload smoothly between the graphical user interface, audio mixing/DSP, MIDI and disk streaming. Multiprocessor systems would generally see smoother and more responsive performance thanks to this multithreaded architecture.

Within the last couple of years there has been a dramatic shift in how people are using music workstations. Low latency audio drivers are commonplace, with most customers routinely recording and mixing at less than 5 milliseconds of latency. This lower latency has in turn opened the door to the increased usage of software synthesizers and live audio effects. In other words, lower audio latency has driven demand for more real time audio DSP.

At the same time multiprocessing systems have become mainstream. At one time only the most die-hard power user would use a workstation with multiple CPUs. But today (thanks in part to Intel's Hyper-Threading Technology) most new desktop PCs incorporate some kind of multiprocessing. Both Intel's and AMD's strategic roadmaps indicate that multiprocessing is here to stay.

SONAR 3.1 has evolved to address the needs of the modern multiprocessing music workstation, where literally dozens of effects and software synthesizers need to be online and responsive. In SONAR 3.1 mixing and DSP are now subdivided across individual processors in a multiprocessor system. Rather than one CPU carrying the workload for all DSP, multiple CPUs can share the load.

Multithreading Concepts

In simple terms, a computer program is a series of instructions that are executed sequentially by a central processing unit (CPU).

A program that employs threads, or is multithreaded, takes this idea and replicates it. Rather than having one sequence of instructions for all tasks that need to be done by the application a multithreaded program uses many sequences of instructions, one per thread. It's just like having many different little sub-programs, all running simultaneously, all running in parallel.

Obviously on a uniprocessor computer these threads don't really run simultaneously. The computer's CPU brain can only perform one instruction at a time. But the operating system is able to provide the illusion that multiple threads are executing in parallel. It does this through a procedure known as scheduling.

The Windows scheduler allows multiple threads to run in orderly fashion, one at a time, by carefully selecting which one should run next and then limiting how much time it gets before moving on to the next one. This scheduling mechanism is very rich and complex, allowing applications to set thread priorities, to make sure that threads which involve user interactions remain responsive, and to allow threads to relinquish their "time slice" if they have nothing left to do.

Multithreading with Multiple CPUs: Multiprocessing

Multithreading on Windows starts to get interesting on multiprocessor systems. The Windows scheduler detects the presence of multiple CPUs, and will allow more than one thread to run – truly in parallel – by assigning one thread to one CPU and another thread to the other. Note that a Windows application must have a multithreaded architecture in order to benefit from multiple CPUs.

Even on a multiprocessor system the weakest link in an application will be the thread that carries the highest CPU burden. For example, prior to version 3.1, SONAR performed all audio mixing and DSP on the same single thread. This meant that one could create a project with a single solitary audio track, and then load this track with audio effects to the point where the project won't play – not on a single processor system, or a quad processor system! This is because one CPU would have needed to do all the mixing and DSP, ironically while other CPUs would be sitting idle.

In other words, the division of labor between threads must be carefully chosen in order to get the maximum benefit from multiple CPUs. Ideally the most CPU-intensive sections of code would be multithreaded. In the case of SONAR, it meant that the code which does mixing, runs plug-ins and synthesizers, needed to be designed a way to be run on multiple threads.

Mixing and DSP in SONAR 3.1

SONAR 3.1 exploits the new SONAR 3 busing architecture to enable multithreaded DSP. In the SONAR 3.1 audio engine there is a basic "audio processing unit" upon which user-visible constructs such as tracks and buses are built. So the logical progression in SONAR 3.1 was to allow the processing of each of these audio processing units to be performed on different CPUs.

An audio processing unit in SONAR 3.1 can process either a project track, a project bus, or a software synthesizer in the synth rack. Therefore a project with 8 tracks, 4 buses and 4 synths has 16 audio processing units for the purposes of this discussion.

Along with the existing set of threads already created for UI, disk and MIDI, SONAR 3.1 will create one mixing/DSP thread for each CPU on the system. During audio streaming, the SONAR audio engine executive constantly builds a first-in, first-out task list, with each task corresponding one audio processing unit. At the same time, in parallel, each mixing/DSP thread removes items from this task list until all tasks are complete and the list is empty.

The design is analogous to a construction site where the job manager has a checklist of jobs to be done. Some jobs must be done before others, and the manager knows how to order them. Workers come to the manager one at a time and receive a task to do. When a worker finishes a task he comes back to the manager to get another. When the task list is empty everyone can go home!

The benefit of this design is that it is self-adjusting to the work load of the project. Most real-world music projects have effects spread out among all tracks and buses, and have several synth instances. This means that the mixing/DSP work load naturally spreads out across all available processors.

By contrast, one could imagine a design where even-numbered tracks were assigned to one processor and odd-numbered tracks to the other. Such a design would not suitably divide the work done by buses and software synthesizers.

Intel Hyper-Threading Technology

Intel HT Technology is a technology that allows a low-cost CPU to present itself as dual CPUs to the operating system. According to Intel, for most applications the current generation of HT processors yields about a 10% performance gain.

Digital audio workstations such as SONAR 3.1 benefit somewhat from HT, but not in the ways one might think. To understand why, one must first understand some basics of the HT architecture.

In simplified terms, a modern CPU is comprised of several sub-units, including a math processing unit, a memory transfer unit, cache memory and registers. (A "register" is a storage location for the intermediate results of a computation. For all intents and purposes, the entire state of a program or a computation can be encapsulated by its registers.)

On a truly dual CPU system, every single one of these functional sub-units is replicated, since each CPU is physically distinct and has its own copy of each unit.

On an HT processor only the registers are duplicated (again, in simplified terms). Since the registers represent an entire computation, having a duplicate set of registers means the HT processor can "pretend" to be 2 processors. All of the other functional sub-units are shared between these 2 virtual processors, with logic to arbitrate the sharing so that only one processor uses each sub-unit at a time.

So turning back to a DAW, consider that an HT processor does not have a replicated math processing unit. In other words, even though it looks like dual processors to the operating system, when the DAW needs it to do math operations, it really is only a single processor.

On the other hand, if the work load contains a mixture of math operations and memory operations, then the HT design helps, because one virtual processor can be accessing the math sub-unit, while another accesses the memory sub-unit.

Given this background on how HT works, one would expect a modest performance gain when using SONAR on an HT system. This is indeed demonstrated by our benchmarks (see below).

One final note. Intel's processor roadmap points to more replication in future HT designs. Future HT processors will truly have multiple math units, which means that on these platforms SONAR will enjoy much bigger performance gains.

Benchmarks

The following table lists the results of a SONAR 3.1 benchmark organized by Scott Reams of LiquidDAW. SONAR users voluntarily downloaded a benchmarking file, <http://www.RoadRec.com/Sonar3Test.zip>, and then played it back at various audio buffer sizes (latencies). The results were tabulated and performance gain computed below.

| System | Mode | Latency | | | |
|--------------------------------|--------------------|---------|--------|------------|------------|
| | | 23.2ms | 11.6ms | 5.8ms | 2.9ms |
| Opteron 242, 1.6GHz | Single | 37 | 42 | 48 | 60 |
| | Multiprocessing | 22 | 24 | 32 | 44 |
| | <i>Improvement</i> | 1.7 | 1.8 | 1.5 | 1.4 |
| Opteron 240, 1.4GHz | Single | 41 | 45 | | |
| | Multiprocessing | 26 | 27 | | |
| | <i>Improvement</i> | 1.6 | 1.7 | | |
| AthlonMP 2800+, 2.08GHz | Single | 41 | 48 | 65 | |
| | Multiprocessing | 23 | 32 | 40 | 58 |
| | <i>Improvement</i> | 1.8 | 1.5 | 1.6 | "infinite" |
| AthlonMP 2000+, 1.67GHz | Single | 47 | 56 | 72 | |
| | Multiprocessing | 27 | 35 | 46 | 70 |
| | <i>Improvement</i> | 1.7 | 1.6 | 1.6 | |
| Pentium 3 1 GHz | Single | 80 | 93 | | |
| | Multiprocessing | 54 | 63 | 82 | |
| | <i>Improvement</i> | 1.5 | 1.5 | "infinite" | |
| Pentium 3 1.4GHz | Single | 70 | 80 | | |
| | Multiprocessing | 51 | 57 | 70 | |
| | <i>Improvement</i> | 1.4 | 1.4 | "infinite" | |
| Pentium 4 3Hz (HT) | Single | 31 | 38 | 51 | 95 |
| | Multiprocessing | 30 | 36 | 48 | 90 |
| | <i>Improvement</i> | 1.0 | 1.1 | 1.1 | 1.1 |
| Pentium 4 3.2GHz (HT) | Single | 28 | 33 | 42 | 60 |
| | Multiprocessing | 26 | 31 | 39 | 56 |
| | <i>Improvement</i> | 1.1 | 1.1 | 1.1 | 1.1 |
| Xeon 2.4 GHz | Single | 36 | 41 | 53 | 76 |
| | Multiprocessing | 25 | 27 | 34 | 52 |
| | <i>Improvement</i> | 1.4 | 1.5 | 1.6 | 1.5 |

Note: "Improvement" is measured as the single processor CPU load divided by the multiprocessor CPU load. Ideally a 2 CPU system has an improvement of 2. An "infinite" improvement means the test project would not even play in a single processor configuration, but would play in dual configuration.