Implementing Real-Time Partitioned Convolution Algorithms on Conventional Operating Systems

Eric Battenberg, Rimas Avizienis, David Wessel

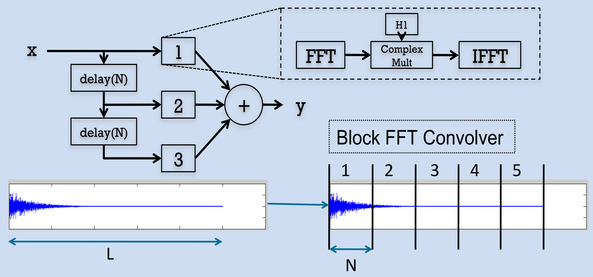
ericb, rimas@eecs.berkeley.edu

The Application

- Convolution is a DSP operation that is widely used in the audio realm to implement a variety of effects (filters, reverberation, etc.)
- When implementing convolution, there is a fundamental tradeoff between efficiency and latency. Partitioned convolution aims to provide low latency at a minimal computational cost.

Uniform Partitioned Convolution

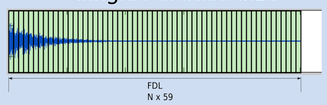
- Direct (time domain) convolution has zero inherent latency at a cost of $O(N^2)$ operations per output sample, where N is the impulse response (IR) length.
- FFT based block convolution has an inherent latency of N samples at a cost of O(N log N) arithmetic operations per output sample.
- Dividing the IR into a series of equally sized partitions, convolving each partition with a delayed version of the input signal, and summing the results provides a middle ground between these two extremes.



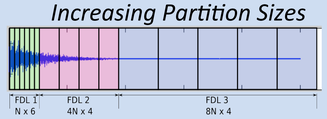
Non-Uniform Partitioned Convolution

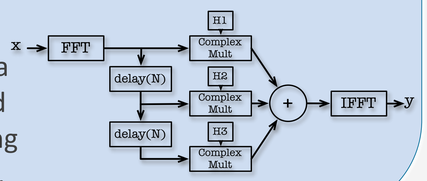
- To further improve efficiency, we can increase the partition size as we progress through the impulse response.
- We can reuse FFTs amongst same-size partitions in a Frequency Delay Line (FDL)
- The optimal partitioning for a given IR length is determined using a dynamic programming based auto-tuning algorithm.

Single Partition Size



Increasing Partition Sizes

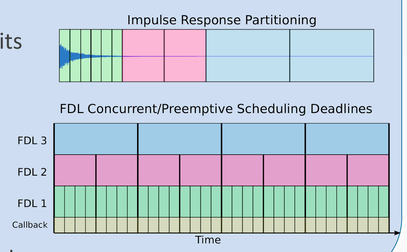




Frequency Delay Line

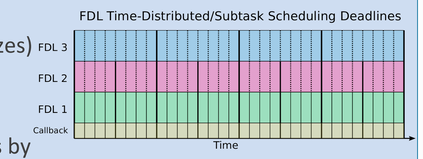
Preemptive Implementation

- A separate thread is created for each FDL and assigned a priority based on its length. Shorter FDLs are assigned higher priorities and will preempt longer FDLs as necessary.
- FFTW (a highly optimized, cross-platform library) is used to perform FFT/IFFT calculations.
- Any configuration of FDLs is supported.

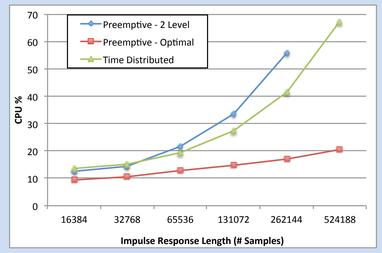


Cooperative Implementation

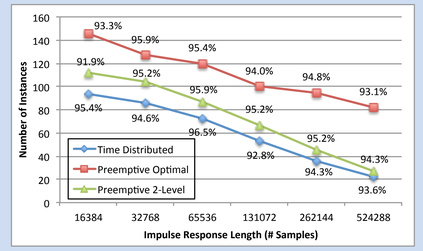
- All processing is done within the context of a single thread.
- Only 2 levels of partitioning (2 FDL sizes) are supported.
- The FFTs for the larger FDLs are subdivided into multiple smaller FFTs by applying decimation in frequency (DIF).
- Making these "time-distributed" FFTs comparable in efficiency to FFTW required hand-tuned assembly code and careful data layout.



Results



- CPU utilization vs. impulse response length for 16 channels of partitioned convolution.



- Maximum number of channels possible without missed deadlines vs. IR length

Conclusions

- Preemptive implementation using optimal partitioning outperforms cooperative implementation in all cases.
- Cooperative implementation required significantly more programmer effort.
- Reported CPU utilization for a fixed number of instances suggests that TD 2-level implementation would be able to sustain more simultaneous instances than PE 2-level implementation, but this is not the case.
- The overhead of context switching is smaller than the overhead of partitioning the work associated with multiple FDLs in a load-balanced manner.
- Audio host applications assume that plugins perform their processing in the context of a single high-priority thread.
- Running multiple plugins that spawn worker threads within such an application (i.e. Max/MSP) is likely to produce unpredictable behavior due to OS scheduling decisions.