Efficient Low-Latency Real-Time Convolution for Multi Core
Eric Battenberg, David Wessel
ericb@eecs.berkeley.edu

The Application
- First real-time app in the Par Lab.
- Partitioned Convolution – an efficient way to do low-latency audio filtering with a long impulse response.
- Used in convolution reverb for environment simulation, creative effect processing, and electronic instrument creation.

Partitioned Convolution
- Direct convolution is expensive.
- Block-FFT convolution has higher latency.
- Partitioning a filter into smaller Block-FFT filters allows us to reduce latency while preserving efficiency.

Multiple Partition Sizes
- To increase efficiency, we can increase the partition size as we progress through the filter.
- We can reuse FFT’s amongst same-size partitions in a Frequency Delay Line (FDL)
- So, what is the most efficient combination of FDL’s for a particular filter length and latency?

Auto-Tuning for Real-Time
- Each FDL gets its own thread which is preemptive with fixed-priority.
- Longer partition sizes are allowed a longer compute time to preserve uniform processor loading.
- We tune for Worst-Case Processor Load (WCPL) using an algorithm based on dynamic programming.

Optimizations
- SSE3 instructions for complex mult-adds.
- Synchronization between FDL’s is done using Condition Variables (CV) and Atomic ops.
- To reduce system calls, synchronization is organized so that only a single CV is used in each callback to signal all worker threads.

Conclusions/Future Work
- Tuning and optimizations allow us to process 100+ channels of audio on current multi-core machines.
- The multi-rate structure of a non-uniform partitioning provides an interesting scheduling problem when interfacing with our cooperative audio graph host.
- An explicitly cooperative implementation could provide more reliable performance at the expense of programmer effort.

Code Examples:
- FFT
- IFFT
- Block FFT Convolver
- Impulse Response Partitioning

Processor Loads for 661500 sample (15sec) filter with latency of 64 (1.5ms):

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Partitioning (number of blocks)</th>
<th>Load</th>
</tr>
</thead>
<tbody>
<tr>
<td>Uniform</td>
<td>64x12036</td>
<td>316%</td>
</tr>
<tr>
<td>Gardner (fastest FDL growth)</td>
<td>64x2, 128x3, 256x4, 512x6, 1024x8, 2048x12, 4096x18</td>
<td>3.54%</td>
</tr>
<tr>
<td>Mix WCPL</td>
<td>64x2, 128x4, 256x6, 512x12, 1024x20, 2048x32, 4096x49</td>
<td>2.55%</td>
</tr>
</tbody>
</table>

Example scheduling of each FDL work unit

Diagram:
- Frequency Delay Line
- Impulse Response Partitioning