Design of a Convolution Engine optimised for Reverb

Fons Adriaensen

4th Linux Audio Conference
ZKM Karlsruhe 27. . . 30 April 2006

Overview

- Using convolution for reverb.
- Anatomy of natural reverb.
- Aella — DSP structure
- Aella — Requirements.
- Using non-uniform partition sizes.
- Using multiple priorities.
- Demo

Using convolution for reverb – (1)

Partitioned convolution with uniform partition size K (as used in BruteFir, jack_convolve and JACE) requires three basic operations, each of size $2K$:

- forward FFT (FWD)
- multiply and accumulate (MAC)
- inverse FFT (INV)

Except for very small sizes, the complexity of each step is approximately proportional to K.

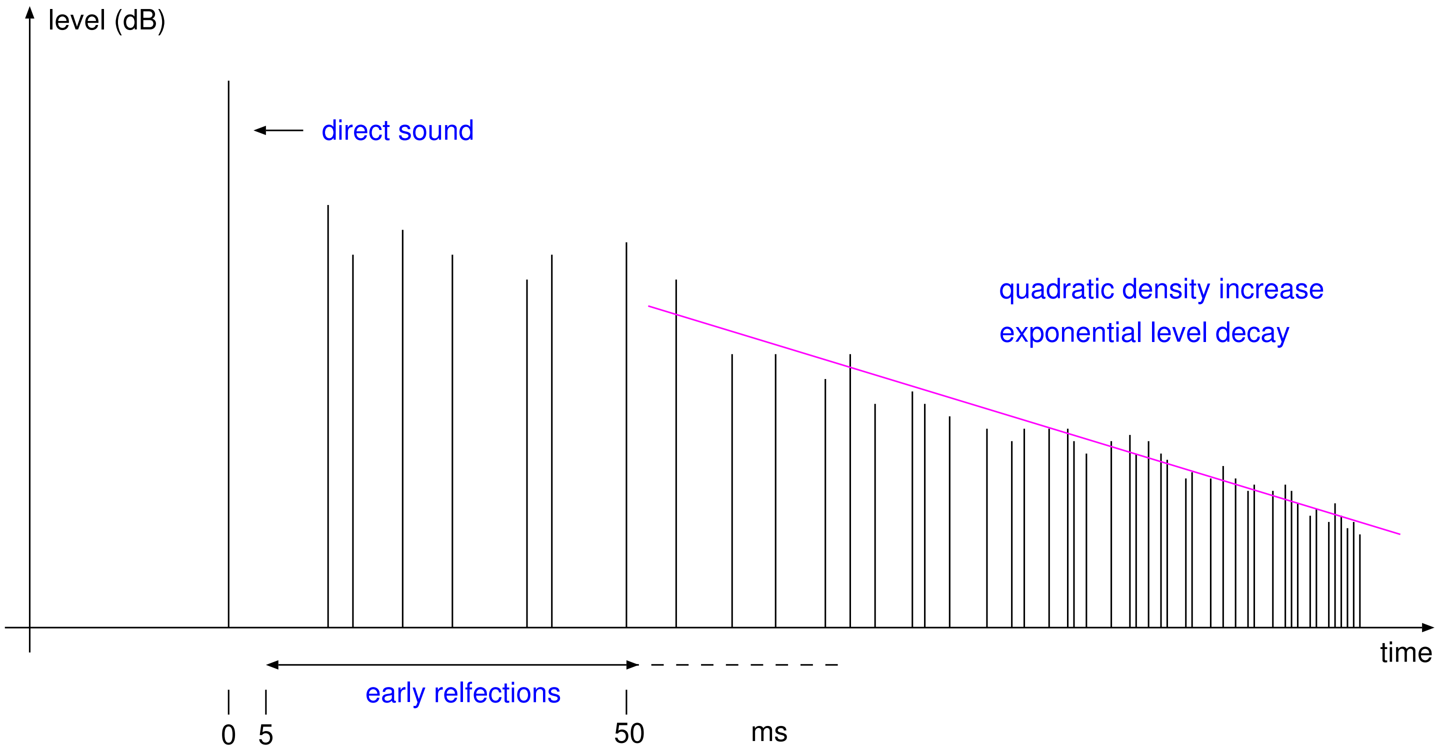
For N_i inputs and N_o outputs, the work to be done per second is proportional to $F\times$ ($N_i\times$ FWD + $N_o\times$ INV + $\sum M_{ij}\times$ MAC) where F is the number of partitions per second, and M_{ij} is the number of partitions in the the convolution from input i to output j.

If the MAC steps dominate the complexity, then the CPU load will be inversely proportional to partition size.

Using convolution for reverb – (2)

- Processing delay is zero if the partition size K is equal to the period size P.
- For $P < K$ it will be $K - P$.
- There is a tradeoff to be made between CPU load and delay.
- Zero delay with reasonable CPU load *is* possible, but requires the use of *non-uniform* partition sizes.
- Apparently no Linux application using non-uniform partition size convolution does exist.
- There is a requirement for *negative processing delay* to compensate for a loop in JACK's processing graph.

Anatomy of natural reverb – (1)



Anatomy of natural reverb – (2)

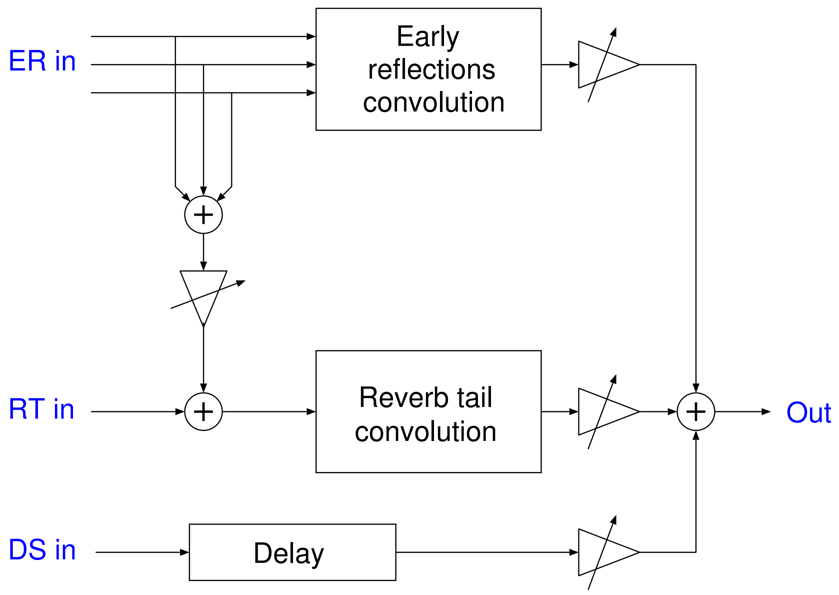
Early reflections

- ER occur in the first 5...80 ms after the direct sound.
- They are not heard as a separate sound but merge with the DS.
- The pattern and directions of ER provides information about the size and shape of the acoustic environent.
- ER patterns depend on source location, and must be correct in order not to contradict the direct sound.

The reverb tail

- The RT is heard as a distinct separate sound.
- Typical decay is exponential, with exceptions.
- The RT depends again on source location, but we can't hear the difference in most cases.
- One RT concolution can often be shared for all sound sources that appear from the same general direction.

Aella – DSP structure



- Output up to 4-ch Ambisonics B-format (5.1 decoded from B-format).
- Multiple inputs for ER, dependent on sound source placement.
- One input for RT, shared by all sources.
- Direct sound path used only to compensate for processing delay when necessary.
- Existing DAWs do not support required structure very well.

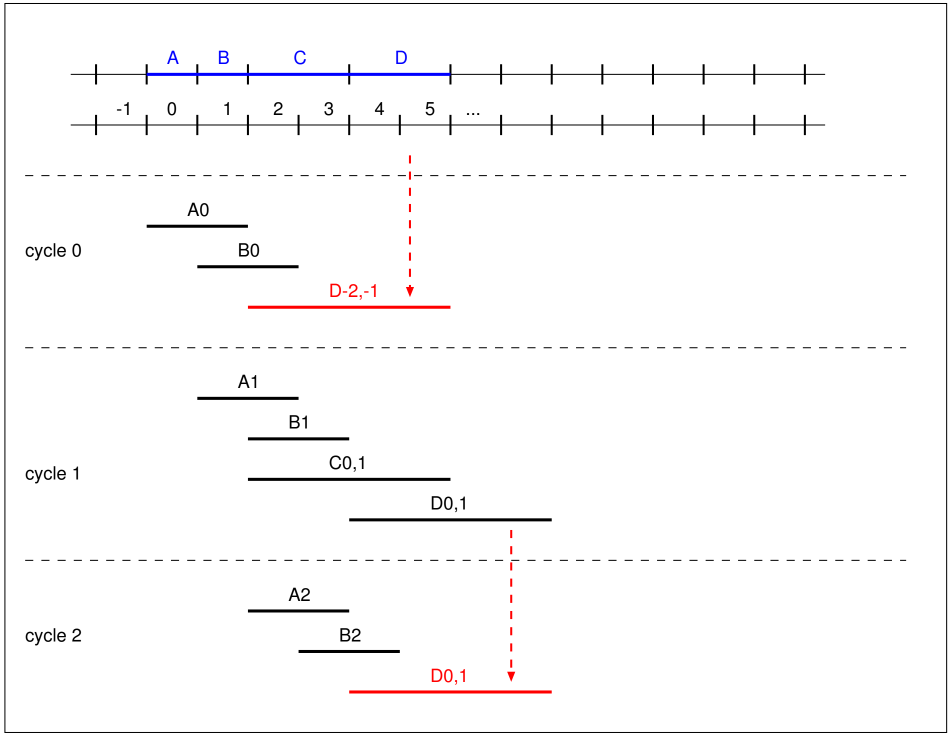
Aella – Requirements

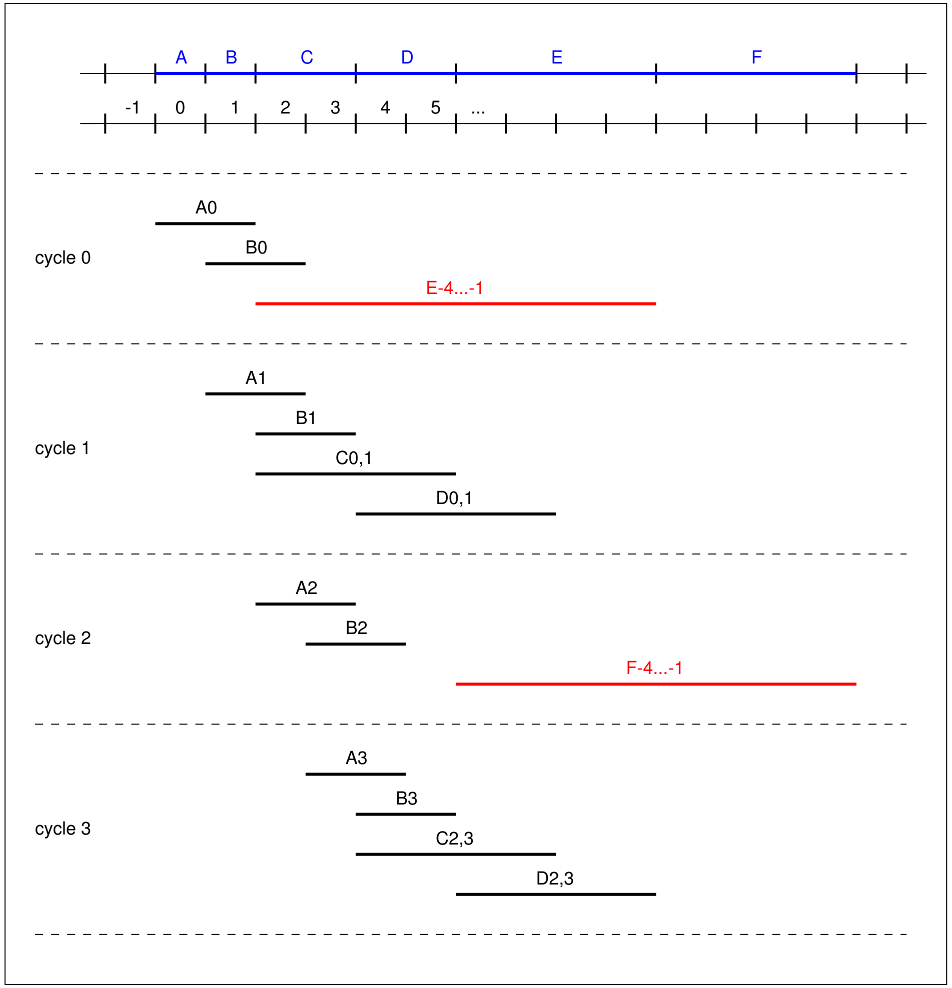
- Ease of use. Select reverb and configuration, click & go (TM).
- Support multichannel surround formats.
- Permit trade-off between CPU power and processing delay.
- Work with JACK period sizes down to 64 without any processing delay.
- Support easy (automatic) compensation for loop delay.
- Be usable for both 'natural' and 'effect' reverb programs.
- Allow real-time modification of the reverb envelope.
- ...

Non-uniform partition sizes – (1)

- Applications should generate the same CPU load in the JACK callback for each period.
- Using partition sizes larger than the JACK period size (JACE) already requires careful planning.
- Using multiple sizes complicates things no end – a *hard* problem.
  - Many parameters: period size, maximum allowed delay, number of inputs and outputs, number and length of convolutions,
  - Multiple FFT operations of different sizes on the same data,
  - Complex buffering schemes.
- An interesting scheme was proposed by Bill Gardner in 1995 (patented).

Non-uniform partition sizes – (2)



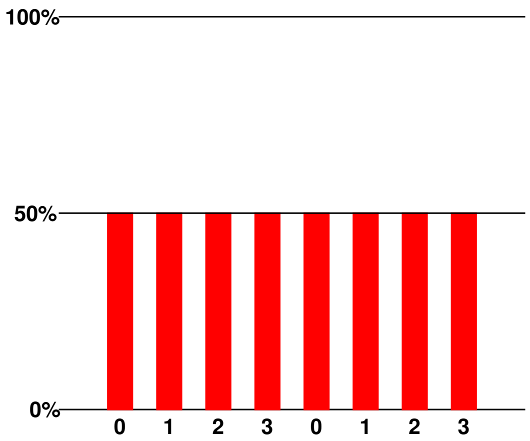


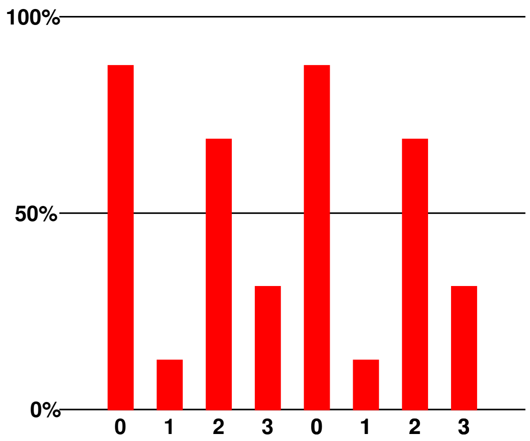
Non-uniform partition sizes – (3)

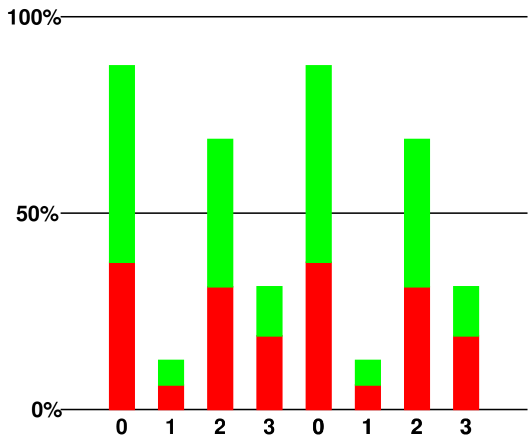
- Gardner-like schemes are valid only in simple cases, e.g. one input and one output.
- Typical applications including reverb require a complete matrix of $N_i \times N_o$ convolutions, sometimes of different lengths.
- There is no simple nor even a complicated algorithm that finds the optimal way to organise the work while respecting real-time constraints.
- The key to the solution is to observe that not all work that can be started within a callback need to be finished in the same period.

Multiple priorities – (1)

- Processing that need not be terminated in the current cycle can be moved to one or more threads running at lower priority.
- The work remaining in the process callback may be irregular, but should be less than the average.

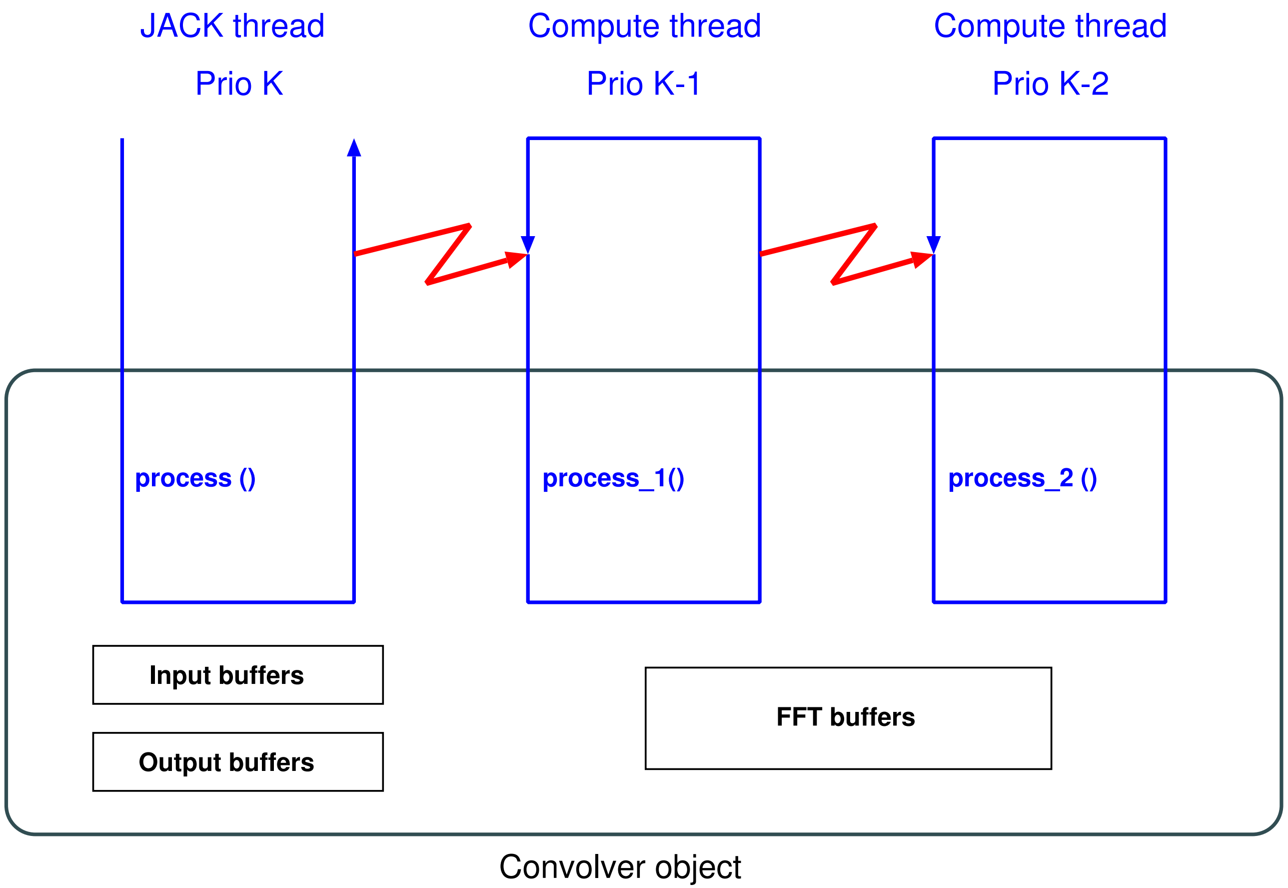






Multiple priorities – (2)

Practical solution: multiple threads using a single convolution object.



Multiple priorities – (3)

$A\epsilon\lambda\lambda\alpha$ uses a mix of techniques:

- Small convolutions for the ER are performed in the process callback using a Gardner-like scheme.
- Larger partitions for the ER go to auxiliary threads at lower priority.
- All work for the reverb tail is done at low priority but still real-time.

There are some further complications:

- Work on very large partitions in the auxiliary threads must be split up in order to preserve system responsiveness.
- Real-time 'live' modification (by cross-fade) of the IR complicates matters.
- The process callback must never wait for the lower-pri threads - they fail softly.

Demo

Current state:

- DSP code works in 'test-bench' mode.
- Application framework & GUI (separate process) nearly finished.
- The two don't yet know each other ...
- Future depends on the Free Impulse project.

Demo using JACE with Ardour:

- Playing piano in St. Lucia church.

Design of a Convolution Engine optimised for Reverb

Questions and Answers . . .