Measuring the performance of real-time Digital Signal Processing using Pure Data and GPU

André J. Bianchi and Marcelo Queiroz

Computer Science Department - Institute of Mathematics and Statistics - University of São Paulo

{ajb,mqz}@ime.usp.br

1 Introduction

We study the **computational performance** of **Digital Signal Processing** using **Pure Data** (http://puredata.info) to outsource parallel computation to commodity **GPU cards**. Data *roundtrip time* is analyzed by measuring memory transfer times to/from GPU and comparing a pure FFT kernel with a full Phase Vocoder analysis/synthesis kernel for different DSP block sizes. This makes it possible to establish the maximum DSP block sizes for which each task is feasible in real-time for different GPU card models.

In order to establish the feasibility of using a GPU-aided environment with Pd on realtime performances we perform the following measurements:

- **Memory transfer time.** Since a GPU only processes data that reside on its own memory, memory transfer can represent a bottleneck for parallel applications that use GPU. Generally, the fewer the amount of data transfers the better.
- **Kernel execution time.** This is the total time used by all instructions performed on the GPU, *after* memory is transferred from the host and *before* memory is transferred back to it.
- **Full roundtrip time.** This is the total time taken to transfer data from host to device, operate on that data, and then transfer it back to the host. This is the single most important value to compare with the DSP cycle period, in order to establish the feasibility of using the GPU in real-time environments.

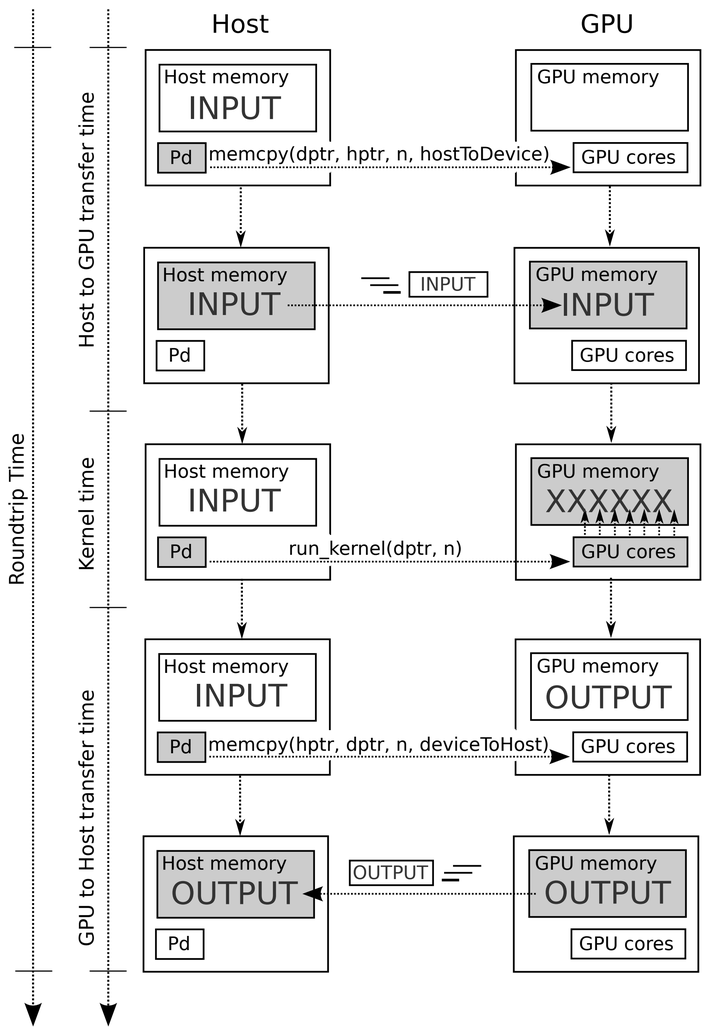
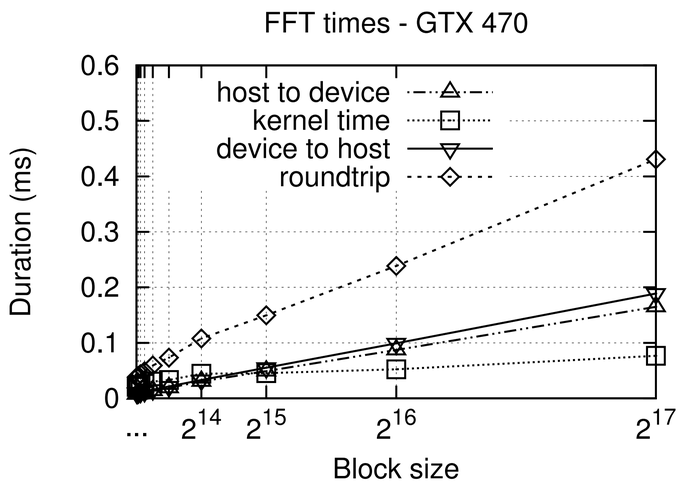


Figure 1: *Pd use of the GPU during one DSP cycle. Gray blocks indicate the active parts on each step.*

2 Results

2.1 FFT: data transfer and roundtrip

The figure below shows the time taken to transfer memory from host to device and back for different block sizes, as well as the time taken to perform one-way FFTs.

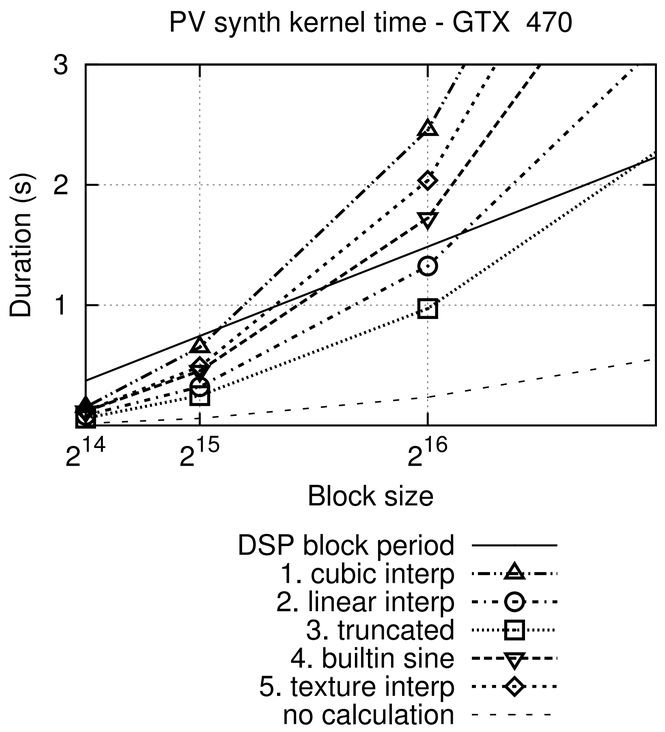


2.2 Phase Vocoder parallel synthesis

The Phase Vocoder represents a signal as a set of oscillators whose amplitudes and frequencies vary with time. After the analysis and parameter estimation, it is possible to reconstruct the signal by summing the oscillators' outputs to achieve effects such as independent modifications of pitch and speed.

Five different implementations of the oscillator calculation were compared, using different combinations of $1024-$point sine wave lookup table and GPU's built-in functions:

1. Table lookup with 4-point **cubic interpolation**.
2. Table lookup with 2-point **linear interpolation**.
3. Table lookup with **truncated** index.
4. Direct use of the **built-in sine** wave function.
5. Table lookup with (linearly interpolated) **texture fetching**.



Implementations (1), (2) and (3) grow proportionally, according to the number of operations involved: truncated table lookup is faster than linear interpolation, which is in turn faster than cubic interpolation. The built-in sine wave implementation (4) achieves an intermediate result between implementations (1) and (2). Texture fetching with linear interpolation, also known as implementation (5) is somewhat faster than cubic interpolation.

3 Conclusions

We could conclude that:

- Small implementation differences can have significant results regarding kernel time consumption on the GPU. Conscious choices have to be made in order to use the GPU's full potential for larger block sizes.
- If we restrict the roundtrip to few memory transfers in each direction, then there is no need to bother with memory transfer time as its magnitude is of the order of tenths of milliseconds while DSP block periods are of the order of several milliseconds even for the smaller block sizes considered.

The **maximum block sizes** for the realtime use of each oscillator implementation for each card we benchmarked are summarized below:

model \ implementation	1	2	3	4	5
GTX 275	2^{14}	2^{15}	2^{15}	2^{15}	2^{14}
GTX 470	2^{15}	2^{16}	2^{16}	2^{15}	2^{15}

4 Get the code

The source code is available as a `git` repository. You can clone the repo either by decoding the QR code on the left or by directly cloning the URL below:

http://www.ime.usp.br/∼ajb/repo/rtpdgpu.git

References

[1] Charles Henry. GPU audio signals processing in Pure Data, and PdCUDA an implementation with the CUDA runtime API. In *Pure Data Convention*, 2011.

[2] Lauri Savioja, Vesa Välimäki, and Julius O. Smith. Audio signal processing using graphics processing units. *J. Audio Eng. Soc*, 59(1/2):3–19, 2011.

[3] Nicolas Tsingos, Wenyu Jiang, and Ian Williams. Using programmable graphics hardware for acoustics and audio rendering. *J. Audio Eng. Soc*, 59(9):628–646, 2011.