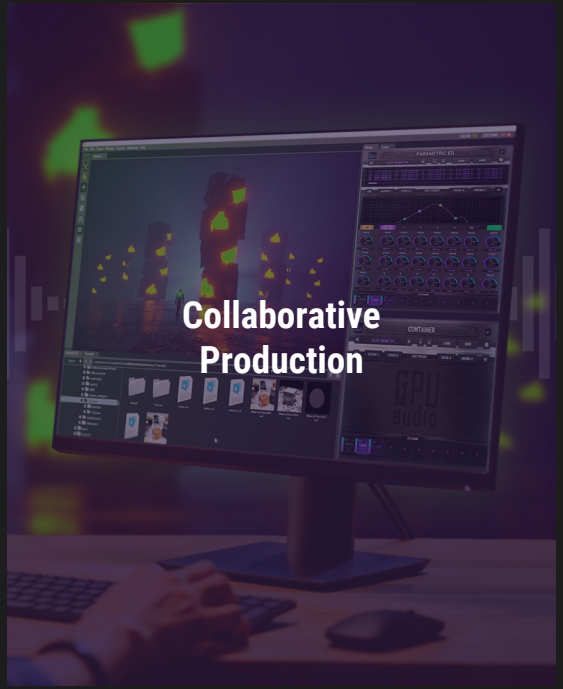
GPU audio | NVIDIA

Unlocking the GPU for Real-Time Audio


“It's kind of fun to do the impossible” – Walt Disney

Next Gen Audio needs a new standard



GPUs offer virtually unlimited power & scalability to digital audio compared to CPUs, but the problems are as inherent as the possibilities

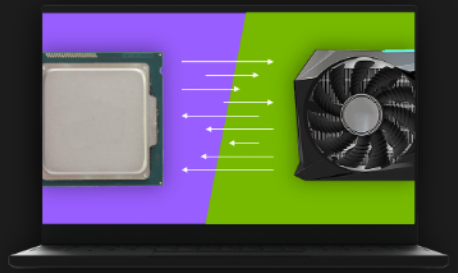
There are three major challenges for GPU Audio processing

GPU audio | NVIDIA

1 Parallelism and Heterogeneity

2 Multiple tracks and effects

3 Data transfer between CPU and GPU



Not only did we solve these issues,
but the journey led to the discovery of truly
next-gen features for audio production

Today we will

Tell the story about how we **solved the problems that kept GPUs back for years**

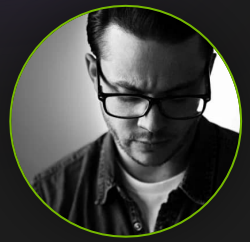
Demonstrate **GPU audio in action**

Share some of the possibilities and **next-gen audio features** of the tech stack

Announce a few **world's-first products**

Officially launch our **early access community and demos**

With the help of industry experts behind GPU Audio

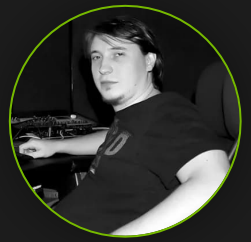


Jonathan Rowden

- Independent Sound Designer / Composer / Saxophonist
- Partnership Development

Alexander Talashov

- Technology Architect
- Managing Partner
- CS & HPC Scientist



Vasiliy Sumatokhin

- Managing Partner
- Sound Engineer for almost 21 years

Markus Steinberger

- Senior Researcher, Advisor
- MSc in Computer Engineering,
- PhD in Computer Science

Aleksandrs Prokopcuks

- Managing Partner
- CTO
- DSP Scientist

GPU Audio is helping developers to reimagine audio processing use cases on traditional IT platforms

Breaking away from the limits of typical audio software

GPU Audio: the world's first novel technology enabling real-time audio DSP on GPUs

- Real-Time high performance DSP
- 1ms latency, regardless of channel count or effect instance count
- Powerful parallel processing, without added latency
- Instant results
- Network ability and scalability
- Audio processing over Ethernet, wifi6 or 5-6g connections

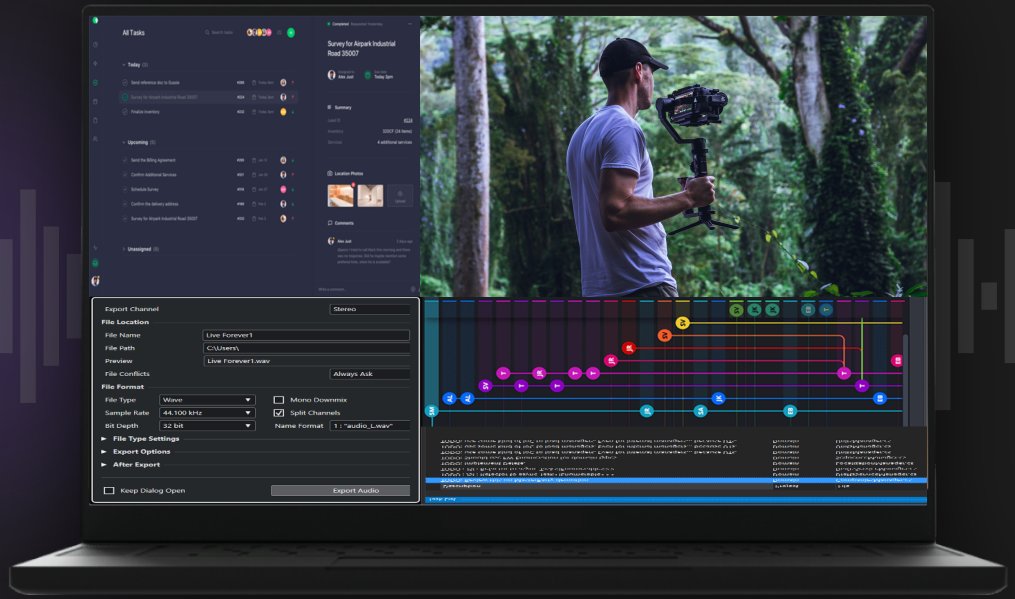
GPU Audio bridges the gap between Pro Audio and Next Gen Tech Standards

Features of our solution open up a world of high-performance audio and production to emerging technologies

- AI/ML
- Web3 experiences
- Interoperable collaboration platforms like NVIDIA's Omniverse
- Spatial audio computing and simulation
- Next Gen DAW and music production software

Bounce mode & rendering accelerated by GPU

Mixdowns for huge projects



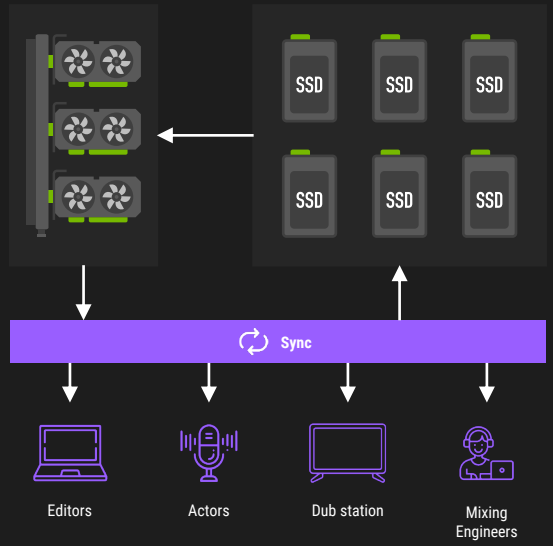
We can still rely on all our techniques for parallelization, especially as processors only work on a small number of input samples in parallel

We can now transfer even larger amounts of data at once and we do not have the 1ms delay requirement - so this is even the easier case

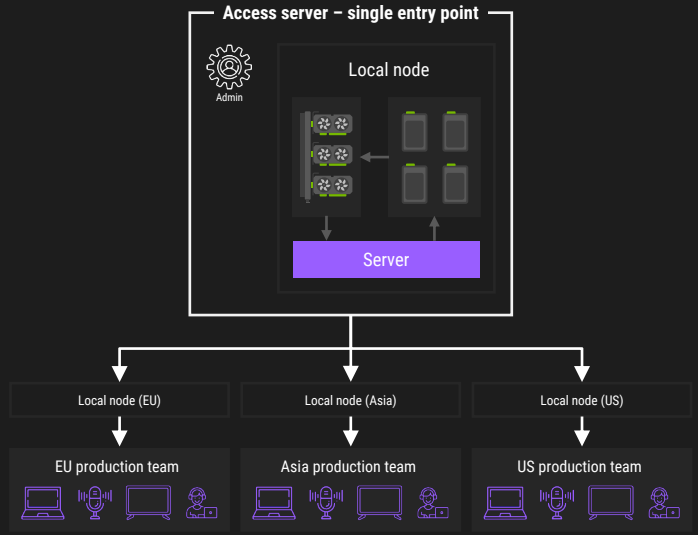
GPU-based solution can be easily scaled in all scenarios

On-Premise or cloud deployment solutions based on IT-first architecture

On-premise scenario

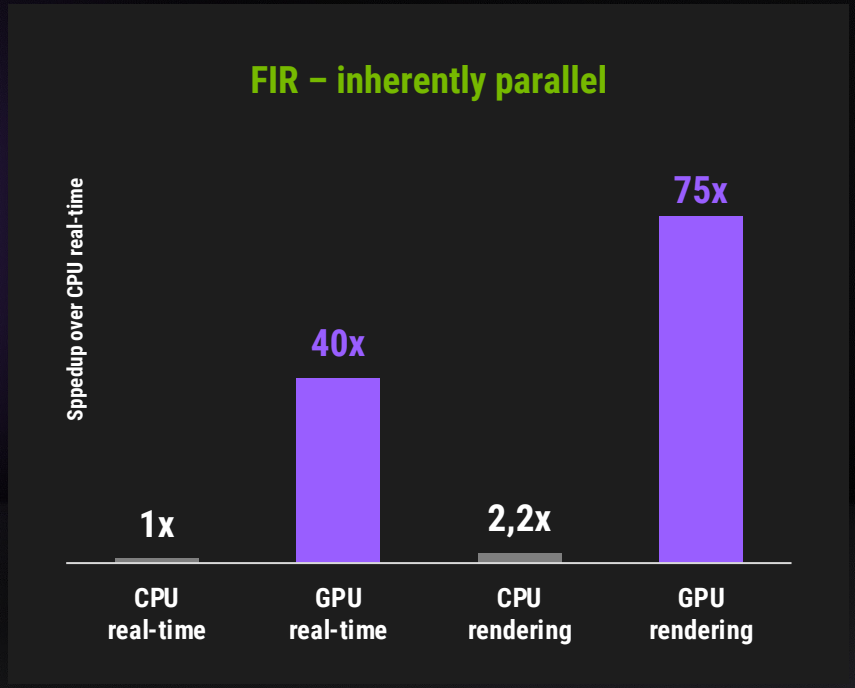


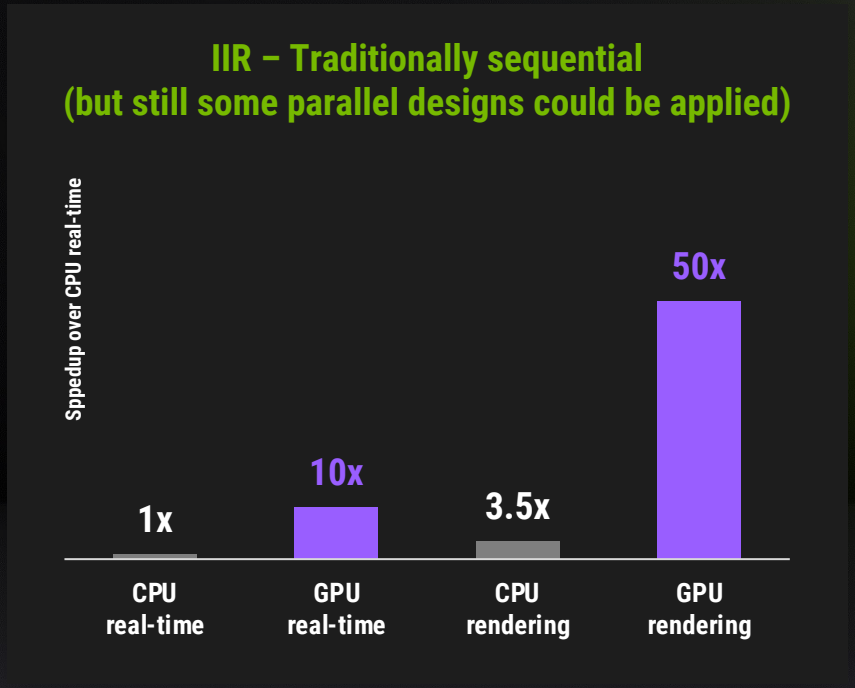
Cloud-based scenario



GPU Audio speedup over CPU implementations

Algorithms designed by GPU Audio achieve significant speedups over CPU processing in both real-time processing and rendering use cases, for both inherently parallel effects and traditionally sequential tasks



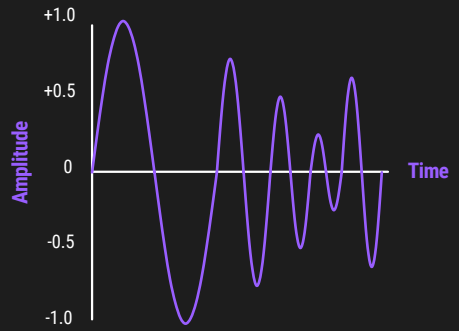


Audio processing on GPU

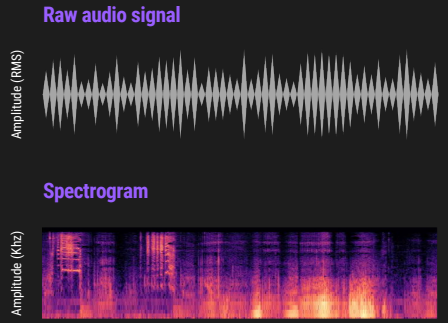
Fundamental problems prevent successful commercial solutions

GPU audio | NVIDIA

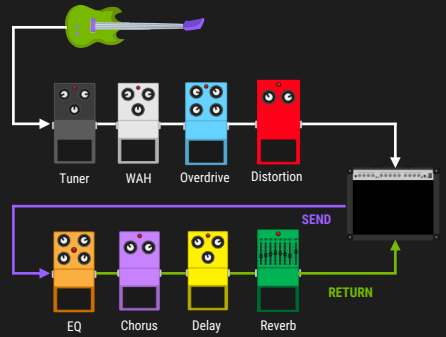
Single audio channel
is one-dimensional signal



Many classic approaches
to parallelize tasks on GPU
use multidimensional signals

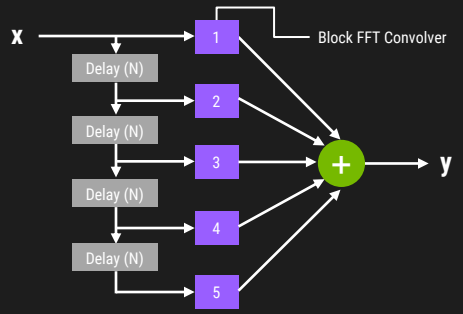


Classic audio processing
DSP algorithms have many
strict data dependencies

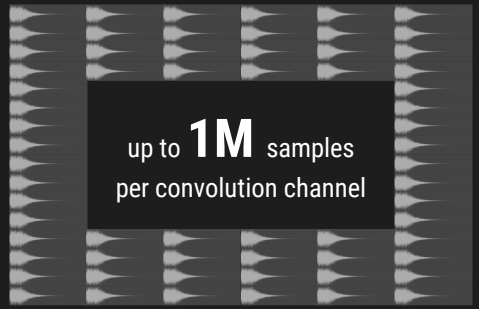


Example of easily parallelizable component: FIR

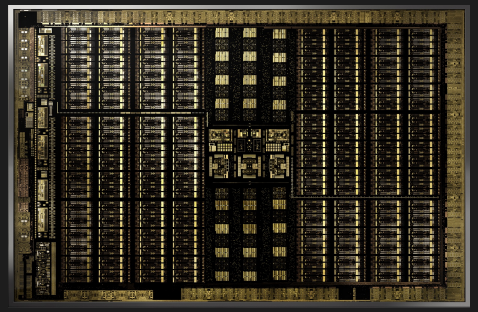
Definition $$y[n] = b_0x[n] + b_1x[n-1] + b_2x[n-2] + \ldots + b_Nx[n-N]$$



Finite impulse response filter or convolution due to the fact that **all samples can be computed independently** is easily parallelizable



However, the **number of computations it costs** even using non-trivial solutions **is massive**



Having so many cores in the modern GPU we can **greatly beat CPUs in terms of executing lots of long FIR filters at the same time**

Example of problematic component: IIR filter

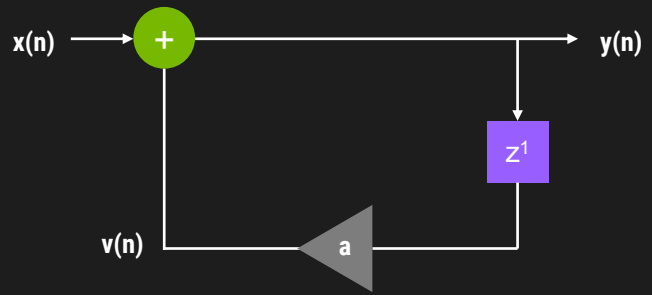
Definitions

Sequence: y[n] = b0 * x[n] + b1 * x[n - 1] + b2 * x[n - 2] - a1 * y[n - 1] - a2 * y[n - 2]

Transfer function: $H(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}}$

Fundamental problem:

Output sequence ***y*** can be evaluated only sample by sample due to the fact that each output samples depends on previous two output samples (feedback) even if part or all input signal ***x*** is known at the start of the processing



Solution for sequential components like IIR filter

Using classical audio DSP designs we cannot use GPU efficiently, as, for example, our equalizer will be able to use only one thread per audio channel

Find design which is producing the same result has some degree of output dependency freedom

Requirements

Retain original **filter numerical stability and SNR**

Doesn't vastly increase number of computations that are needed to **recalculate filter parameters** when user changes those parameters

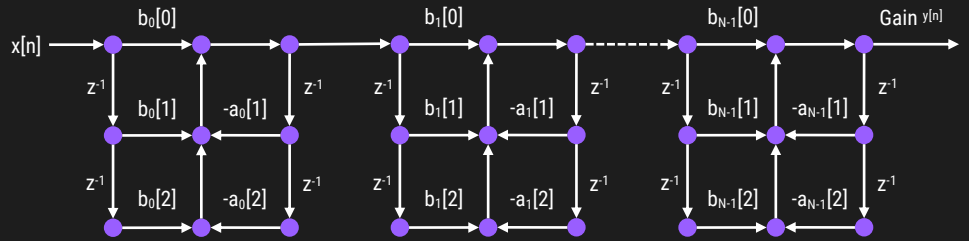
Doesn't vastly increase number of computations that are needed to perform filtering

Universally **works for any IIR filter type**

Further problem: cascade of IIR filters

Using classical audio DSP designs we cannot use GPU efficiently, as, for example, our equalizer will be able to use only one thread per audio channel

Many audio effects are formulated and implemented as a sequence of second order IIRs, for example equalizer is mostly the cascade of such filters, where the count of those filters depending on particular implementation can vary ~10-30 filters

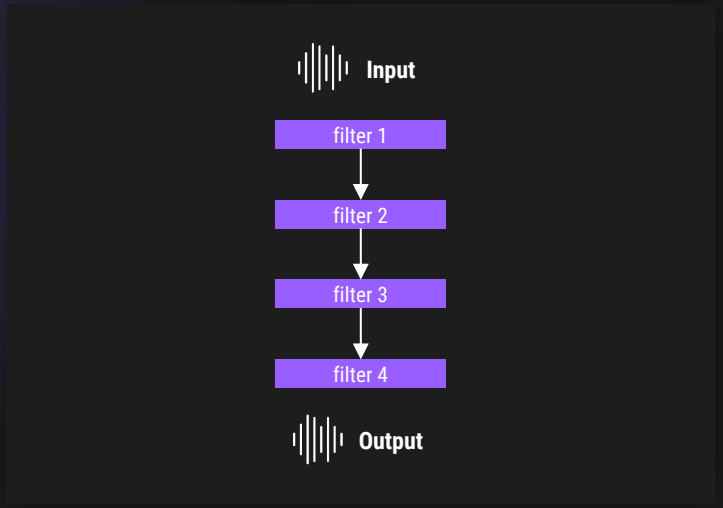


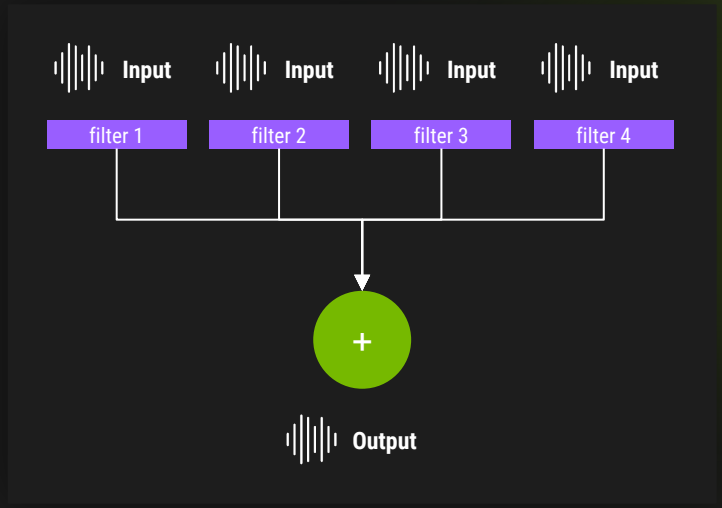
In the end we're getting **sequence of filters** and **each filter internally also can compute** output only sequentially

Solution for cascades

Develop topology of the audio processor to minimize sequential components, parallelize topology

This is not an easy task as we want to get the same audio output with completely different processor design





Engineering tricks for audio DSP parallelization

GPU audio | NVIDIA

1

In compressor detection line we can change attack/release state only once 32 samples without any significant difference in the output

2

Our IIR filter implementation can compute two output samples at one iteration using 16 parallel and independent multiplications

3

Usage of custom derived equations in the form that greatly improves numerical stability in some critical places

Engineering tricks for audio DSP parallelization

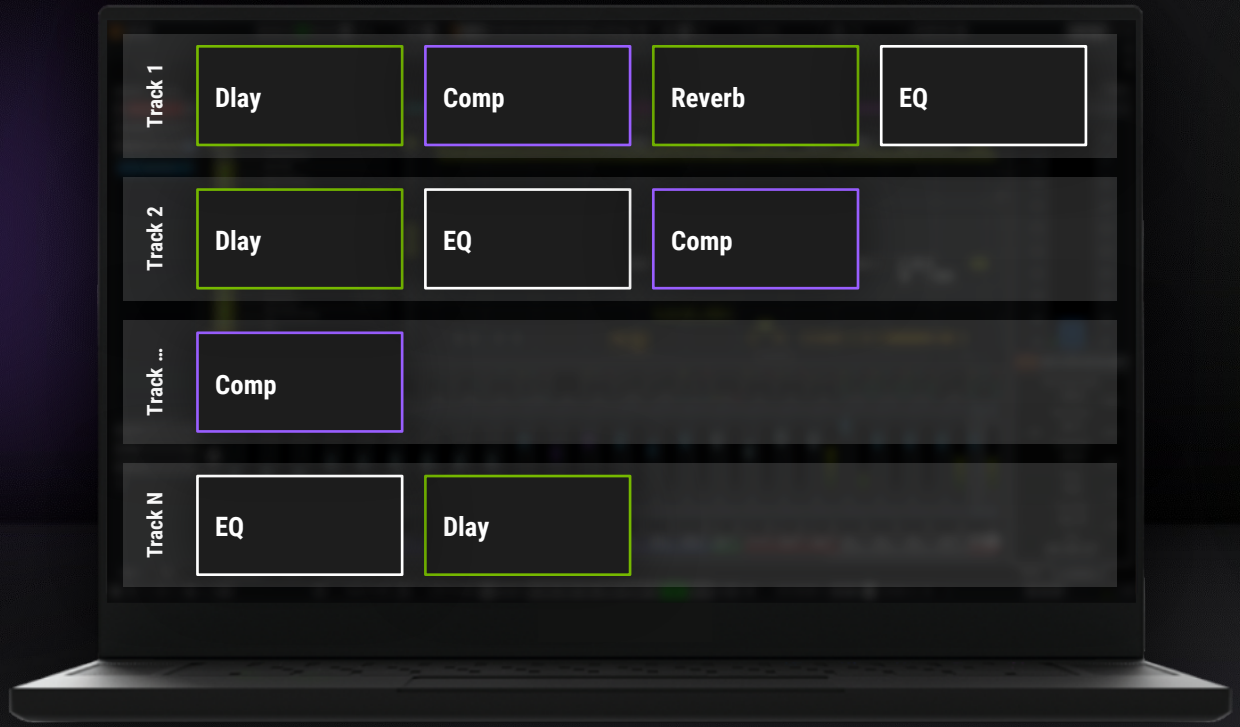
Our equalizer has all of its IIR filters in parallel composition

Our IIR filter implementation can compute two output samples at one iteration using 16 parallel and independent multiplications

Our delay lines and circular buffers are designed in the way that access to every sample in it will cost exactly the same amount of instructions


People spend thousands on external acceleration hardware
and still face bottlenecking limitations
GPU audio
NVIDIA

Pro Audio users want to have unlimited creative power in their DAWs



User's goals

Run loads of tracks with loads of effects

Get **1ms** latency

Challenge 2: Large number of tracks and effects with different parameters

The DAW sees each track individually - the GPU must see all of them at once

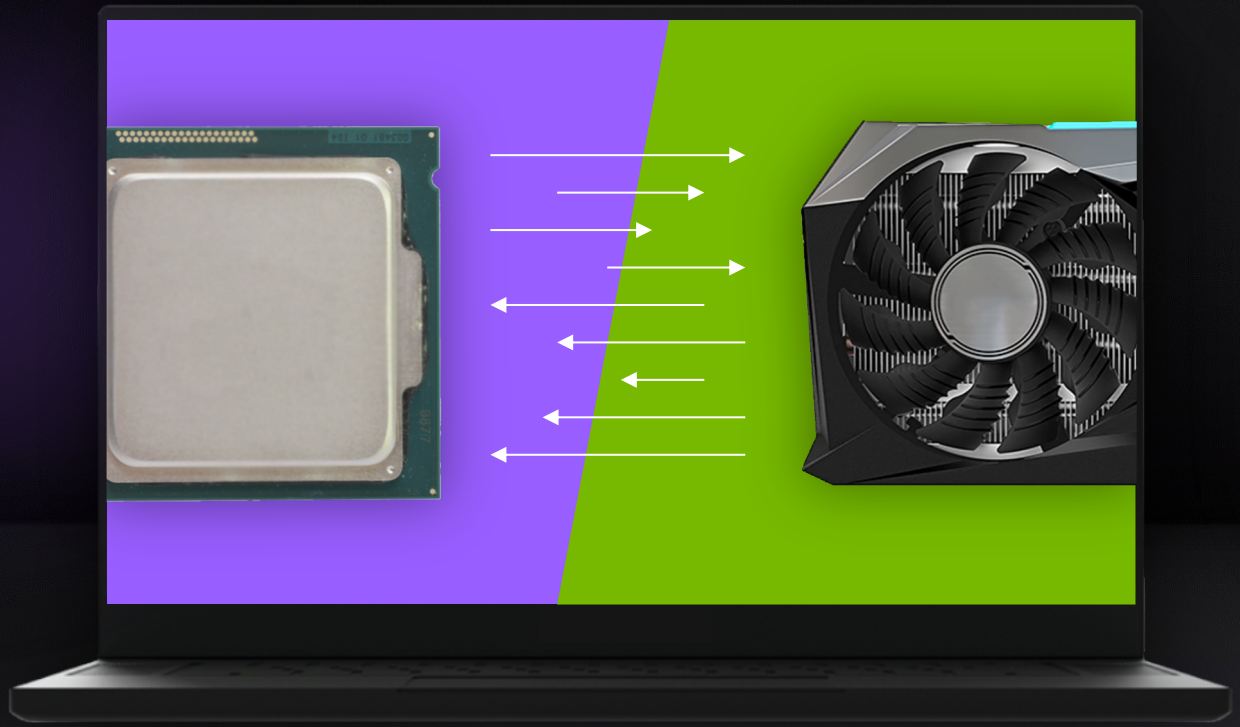
User's goals

Run loads of tracks with loads of effects

Get **1ms** latency

Challenge 3: Data transfer between CPU and GPU

1000 asynchronous data transfers in each direction of 96 samples each just do not work



User's goals ✕

Run loads of tracks with loads of effects

Get **1ms** latency

Meet the GPU Audio Rendering Engine powered by Scheduler

Our proprietary algorithm to solve these challenges

Scheduler Technology Overview

Host Scheduler

Multi-threaded lock-free
low overhead design

- Chain Blueprint generation
- Communication with the DAW
- Data collection and combination
- Triggering processing on the GPU
- Execution time prediction

Device Scheduler

Dynamic, dependency-aware, priority-based, parallel, distributed scheduler, running across kernel launches

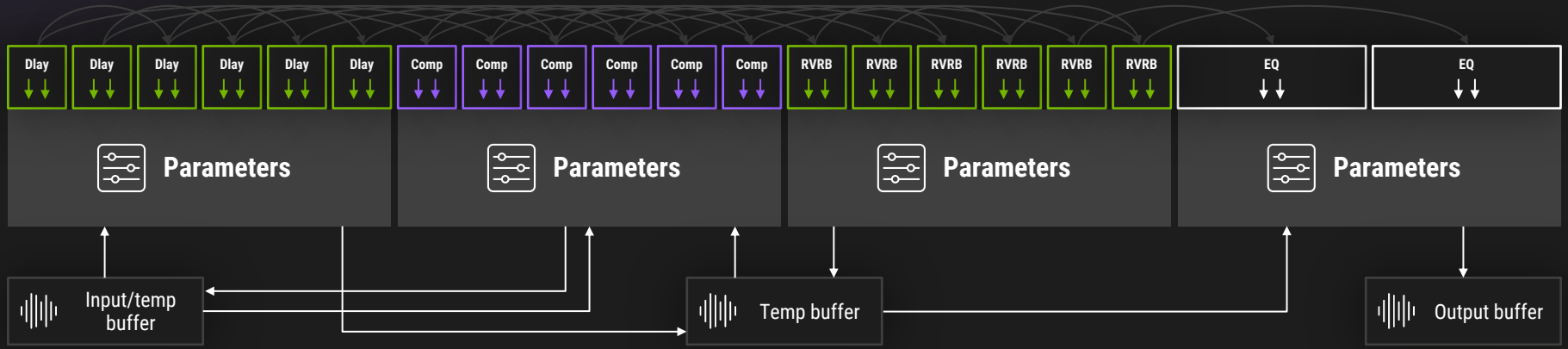
- Chain blueprint management
- Task collection
- Dynamic Resource management
- Dependency resolution
- Execution blocking

Scheduler Blueprints

Repeated processing for increased efficiency

Scheduler Blueprints

Final blueprint



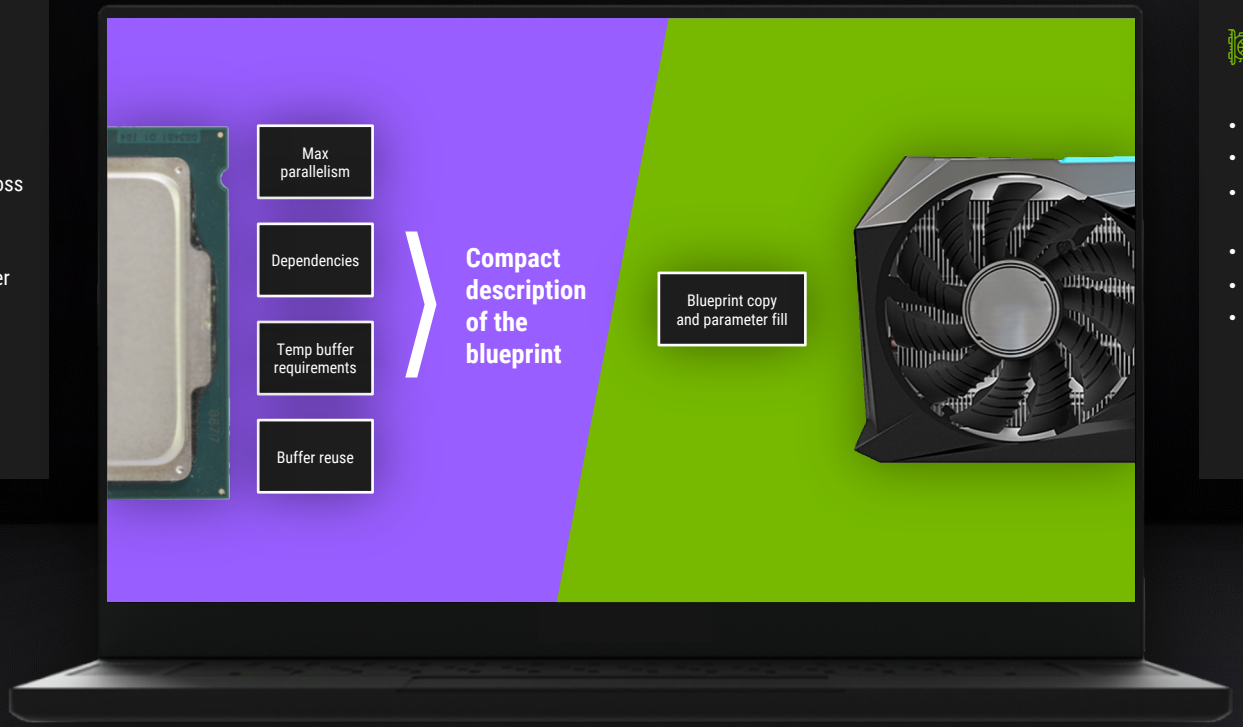
While the parameters and the buffer setup changes for every launch, the task descriptions remain the same every time. That is why we call it blueprints

Blueprint generation and instantiation

Blueprints are generated by the host scheduler and instantiated by the device scheduler

Host scheduler

- Derive parallelism from effects
- Determine parallelism across time and tasks
- Setup dependencies
- Determine temporary buffer requirements
- Find compact Blueprint description



Device scheduler

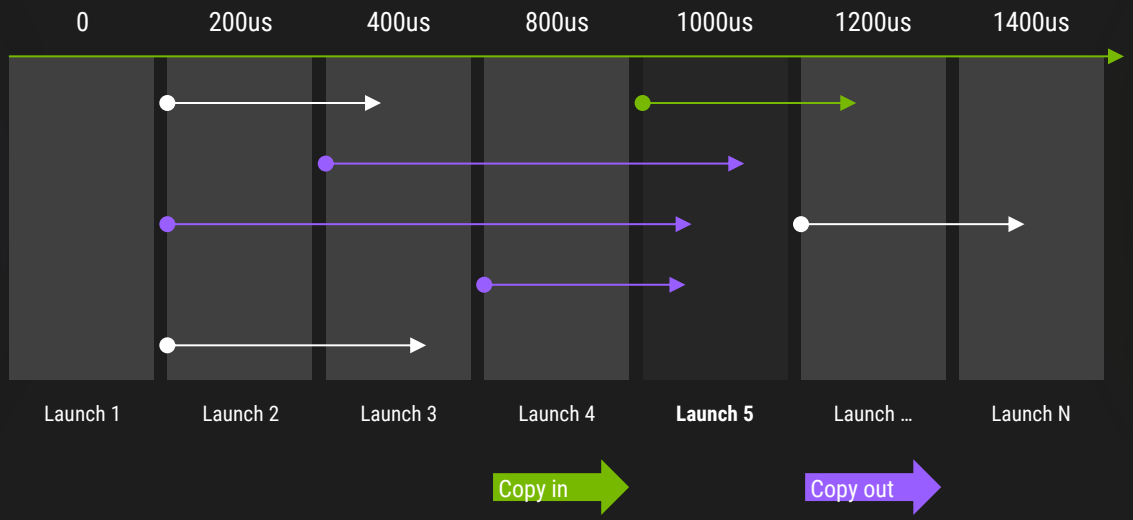
- Receive input data
- Receive parameters
- Copy and instantiate Blueprint
- Fill Blueprint with new data
- Kick off processing
- Run task execution

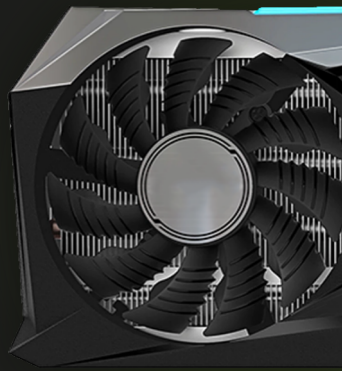
Challenge 1: Parallelism and Heterogeneity

Solved with Rendering Engine powered by Scheduler

Pulsing Scheduler Design

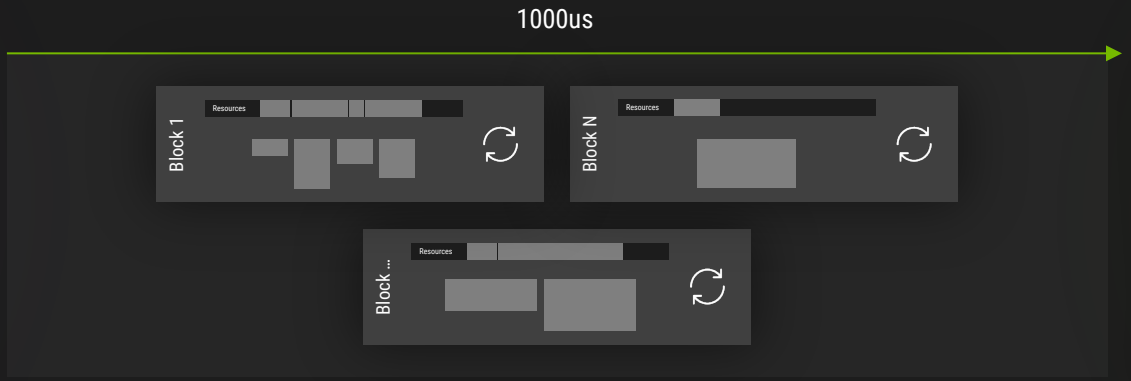
Memory transfers are merged for incoming and outgoing data

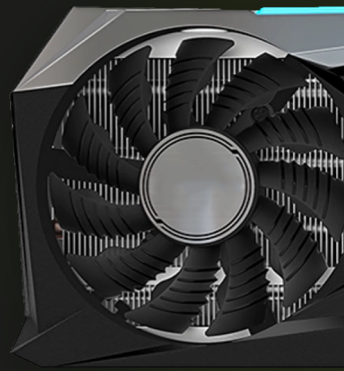




Pulsing Scheduler Design

Device scheduler in each window





Launch 5

GPU audio | NVIDIA

Challenges 2 and 3: Big number of tracks and effects with different parameters and data transfer

Solved with Rendering Engine powered by Scheduler

GPU Audio DSP SDK

Vast collection of **unique algorithms** which can be used to efficiently process sound on GPU

Well-known DSP **processing kernels** for easy implementation

SDK **for both enthusiasts and companies** to build their own products using those algorithms

Multi-layered – fits any use case

To summarize, the GPU Audio SDK is like an onion

This sets developers up with incredible features, across any platform that uses audio

Many layers that can be used as desired

Low latency

High performance DSP

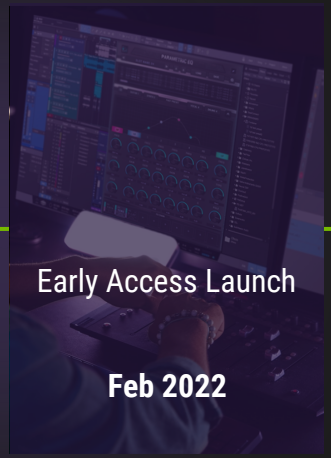
GPU Audio Early Access

Unlock your GPU for Lightning Fast Audio Production

- Zero latency performance
- Instantly rendered audio
- Freedom from DSP bottlenecks

What's next?

GPU Audio growing ecosystem



Early Access Launch

Feb 2022

Beta Suite Release

Mid 2022

SDK

XXX

New DAWs

XXX

GPU Audio Spatial Mixing platform

XXX

Join us!

Early Access Download

DEMO

Early Access – Convolution Networks on GPU

SIGN UP

Beta Sign-Up – Full Plugin Suite

Discord – Join the Community

SDK

SDK Signup – Developers