Noname manuscript No.
(will be inserted by the editor)

On the Performance of a GPU-based SoC in a distributed Spatial Audio System

Jose A. Belloch · José M. Badía · Diego Larios · Enrique Personal · Miguel Ferrer · Laura Fuster · Mihaita Lupoiu · Alberto Gonzalez · Carlos León · Antonio M. Vidal · Enrique S. Quintana-Ortí

Received: date / Accepted: date

Abstract Many current System-on-Chip (SoC) devices are composed of low-power multicore processors combined with a small graphics accelerator (or GPU) offering a trade-off between computational capacity and low-power consumption. In this context, spatial audio methods such as Wave Field Synthesis (WFS) can benefit from a distributed system composed of several SoCs that collaborate to tackle the high computational cost of rendering virtual sound sources. This paper aims at evaluating important aspects dealing with a distributed WFS implementation that runs over a network of Jetson Nano boards composed of embedded GPU-based SoCs: computational performance, energy efficiency, and synchronization issues. Our results show that the maximum efficiency is obtained when the WFS system operates the GPU frequency at 691.2 MHz, achieving 11 sources-per-Watt. Synchronization experiments using the NTP protocol show that the maximum initial delay of 10 milliseconds between nodes does not prevent us from achieving high spatial sound quality.

Keywords wave field synthesis, spatial audio, real time, embedded systems, GPU, Jetson Nano, System-on-Chip (SoC).

Jose A. Belloch
Depto. de Tecnología Electrónica, Universidad Carlos III de Madrid, Spain
E-mail: jbelloc@ing.uc3m.es

José M. Badía
Depto. de Ingeniería y Ciencia de Computadores, Universitat Jaume I de Castellón, Spain.
E-mail: badia@uji.es

Diego Larios, Enrique Personal and Carlos León
Depto. de Tecnología Electrónica, Universidad de Sevilla, Spain
E-mail: {dlarios,epersonal,cleon}@us.es

Alberto Gonzalez, Miguel Ferrer, Antonio M. Vidal,
Enrique S. Quintana-Ortí, Laura Fuster and Mihaita Lupoiu
Universitat Politècnica de València, Spain
E-mail: {agonzal,mferrer}@dcom.upv.es, avidal@dsic.upv.es, quintana@disca.upv.es, lfuster@iteam.upv.es, myhay.360@gmail.com

1 Introduction

In the last decade, parallel systems are being employed in all segments of the industry in the form of multicore processors and many-core hardware accelerators. Digital signal processing is one of the fields which has largely benefited from these devices, and more specifically the synthesis of the spatial audio.

One of the spatial audio technologies available today is Wave Field Synthesis (WFS) in which a sound field is synthesized in a wide area by means of loudspeaker arrays. WFS is usually tackled via digital signal processing techniques to reproduce complex auditory scenes consisting of multiple acoustic objects. The WFS concept was introduced at the Delft University of Technology in the 1980s by Berkhout [1].

A large-scale WFS system requires performing costly computational operations in real time since it involves multiples input and output channels. In recent years, several efforts have aimed at implementing efficiently a WFS system [2]. Some of the authors of the present work carried out an implementation that enhanced the audio quality by offloading the additional computational cost to Graphics Processing Units (GPUs) [3]. To this end, in addition to rendering virtual sound sources, the aforementioned implementation took into account the computational cost that appears when a Room Compensation (RC) block is added to the system. Common RC blocks are based on multichannel inverse filter banks and their purpose is to correct the room effects or echoes at selected points within the listening area [4, 5]. The proposed system provided high performance in terms of computational and acoustic aspects [3]. However, this WFS implementation was designed for an specific room that required a centralized system with a dedicated power-hungry GPU. This fact limits the flexibility and scalability of the system, because any change in the number of loudspeakers and/or the hall distribution involves necessarily a reconfiguration of the system.

Nowadays, there exist System-on-Chips (SoCs) that deliver notable computational capacity while partially retaining the appealing low power consumption. This high performance capability is often provided by an embedded graphics accelerator (GPU) that is integrated in the SoC. One example of this type of system is the NVIDIA Jetson Nano board [6] which contains a high-performance 128-core Nvidia Maxwell GPU. One of its main features is its low power consumption combined with several levels of parallelism yielding a very high performance per Watt. Moreover, there exist also the possibility of adjusting its energy consumption by reducing the frequency of the CPU cores or the GPU.

This work introduces a distributed system containing multiple NVIDIA Jetson Nano boards, where each one of them is in charge of managing the output processing signal of a batch of loudspeakers. As the main result from this work, we have built an environment-adaptable and quickly-reconfigurable dynamic WFS system. From the computational point of view, the total cost is shared among the embedded GPUs of the Jetson boards. In this work we analyze the performance of the system in terms of audio processing (virtual

sound sources and sizes of RC filters), use of different GPU and CPU operating frequencies, and energy consumption. The aim of this work is to find an efficient implementation that exhibits a proper trade-off between performance and energy consumption taking into account real-time constraints.

From the acoustic point of view, each Jetson Nano acts as a node inside a network of acoustic nodes. All nodes aim to generate collaboratively sound fields with natural temporal and spatial properties within a volume or area bounded by secondary sources (arrays of loudspeakers). Therefore, in a distributed system, the secondary sources are distributed among the different acoustic nodes. As a consequence, besides considering the computational and energy aspects, a good time synchronization between the nodes is essential in order to reproduce in unison through the loudspeakers of the system and thus, to offer a correct spatial audio sensation. In order to support this outcome, we carried out a quantitative experiment using multi-channel audio cards that were devoted to acquire the audio outputs of all acoustic nodes (Jetson Nano audio outputs). The proposed experiment allowed us to measure real output delays among nodes by means of cross-correlations.

This paper is structured as follows. We briefly discuss some related work in the next subsection. Section 2 offers an overview of distributed algorithms in the field of audio as well as some background in WFS theory together with a RC filters block. Section 3 analyzes the node synchronization in the distributed WFS system. In Section 4, we explore the performance of the distributed system in terms of maximum number of sound sources that can be rendered in real time; provide a detailed analysis of the power dissipation; and analyze the energy efficiency of different hardware configurations. Finally, Section 5 closes the paper with a few concluding remarks.

1.1 Related Work

The use of System-on-Chips (SoC) within audio systems is becoming widespread. The emergence of SoC composed of parallel processors built either from multi-core CPUs or even a small manycore graphics accelerator (or GPU) contributes a notable increment of the computational capacity, while partially retaining the appealing low-power consumption of embedded systems. For example, in [7], the authors implement an acoustic sound localization system using an ESP32 SoC microcontroller [8]. A headphone-based spatial audio application which tackled a more powerful SoC was introduced in [9]. Multimedia mobile applications were evaluated in Heterogeneous Mobile Multi-Core Processors [10]. A GPU-based SoC is also used in [11] for binaural sound sound synthesis. Our work introduced in this paper differs in that none of the previous efforts has tackled a massive audio system dealing with a huge number of channels in a network of powerful SoCs.

2 Audio distributed systems

Distributed systems provide a cheap, flexible and efficient solution for environmental and habitat monitoring, as well as for monitoring and maintenance of industrial equipment [12]. These systems often involve acoustic applications [13]. Among them, we highlight the use of distributed systems for building huge 3D spatial audio systems, where the acoustic node is focused on the design of the output signals. In addition to producing spatial audio effect, the output signals control the sound field by reducing possible acoustic artifacts that are introduced by external agents. To this end, our designed audio system implements the WFS technique together with a RC filter block.

2.1 Wave Field Synthesis

WFS is a sound rendering method based on the Huygens' principle [14]. The propagation of a wave front can be described by adding the contribution of a number of secondary-point sources (arrays of loudspeakers) distributed along the wave front, as shown in Figure 1(a). Following a model-based rendering with point sources and plane waves [15], the field rendered by a sound source m at a point R, within the area surrounded by N loudspeakers, can be expressed as

$$P(r_{nR}, \mathbf{x}_R, \omega) = \sum_{n=0}^{N-1} Q_n(\mathbf{x}_m, \omega) \frac{e^{\frac{-j\omega r_{nR}}{c}}}{r_{nR}}, \tag{1}$$

where c is the speed of the sound, $\mathbf{x}_m$ is the position of the virtual sound source m, $\mathbf{x}_R$ is the position of the point R, and $r_{nR} = \mid \mathbf{x}_n - \mathbf{x}_R \mid$ is the distance between the n-th loudspeaker and the point R.

The signal to be reproduced by the n-th loudspeaker is represented by $Q_n(\mathbf{x}_m, \omega)$, which is given by

$$Q_n(\mathbf{x}_m, \omega) = S(\omega) \sqrt{\frac{j\omega}{2\pi c}} K \frac{1}{\sqrt{r_{mn}}} \cos(\theta_{mn}) e^{\frac{-j\omega r_{mn}}{c}}, \tag{2}$$

where K is a geometry-dependent constant, $r_{mn} = \mid \mathbf{x}_m - \mathbf{x}_n \mid$ and $\mathbf{x}_n$ is the position of the loudspeaker n. Figure 1(b) shows the geometry of the system, where θ_{mn} denotes the angle between the line that connects $\mathbf{x}_m$ and $\mathbf{x}_n$ and the normal vector $\mathbf{n}$ of the loudspeaker n. The guitar represents the sound source m.

The computed signal (2) consists of several elements that aim to add spatial audio properties to the original source signal whose frequency-domain characteristic is represented by the term $S(\omega)$ [15].

The synthesized sound field can be altered with new echoes introduced by the listening room, reducing the spatial effect. In [5,16], the authors designed and validated a multichannel inverse filter bank that corrects these room effects at selected points within the listening area. To this end, each WFS computed signal is filtered N times and accumulated by each loudspeaker as shown in

Figure 2.1, where the filter F_{jn} transmits the computed signal Q_j to the loudspeaker Y_n. Thus, the final signal to be reproduced by the n-th loudspeaker Y_n is a combination of all the filtered signals. Therefore, the reproduced signal at the n-th loudspeaker is:

$$Y_n(\omega) = \sum_{j=0}^{N-1} \left[F_{nj}(\omega) \odot \sum_{m=0}^{M-1} Q_n(\mathbf{x}_m, \omega) \right], \tag{3}$$

where $\odot$ represents the element-wise multiplication among the WFS signals and the room compensation filters in the frequency domain.

As a consequence, the computational demand increases considerably. It is important to highlight that the length of the filters can be customized before the execution of the WFS distributed system so that we can select higher lengths, which involves a more accurate synthesized signal. In this sense, large lengths avoid that the generated spatial sound is affected by external aspects such as room effects or loudspeakers setup, and provide high spatial sound quality [5]. Unfortunately, this also constrains the number of sound sources that can be rendered in real time.

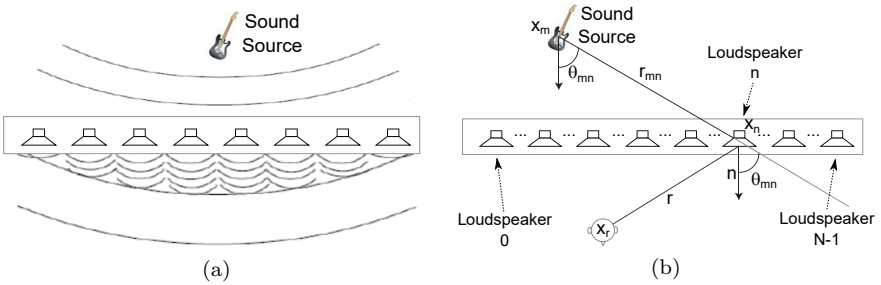
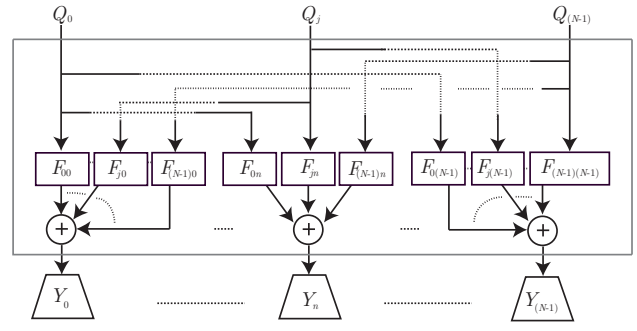


Fig. 1 (a) Theoretical basis for WFS that are based on Huygens' principle. (b) Geometry of a WFS system with the sound source m (guitar), N loudspeakers, and the distances among the sound source, the loudspeakers, and the listener (R).



Multichannel inverse filter bank, where each WFS computed signal filtered by N times. The signal that is reproduced by a loudspeaker is a combination of all of the filtered signals.

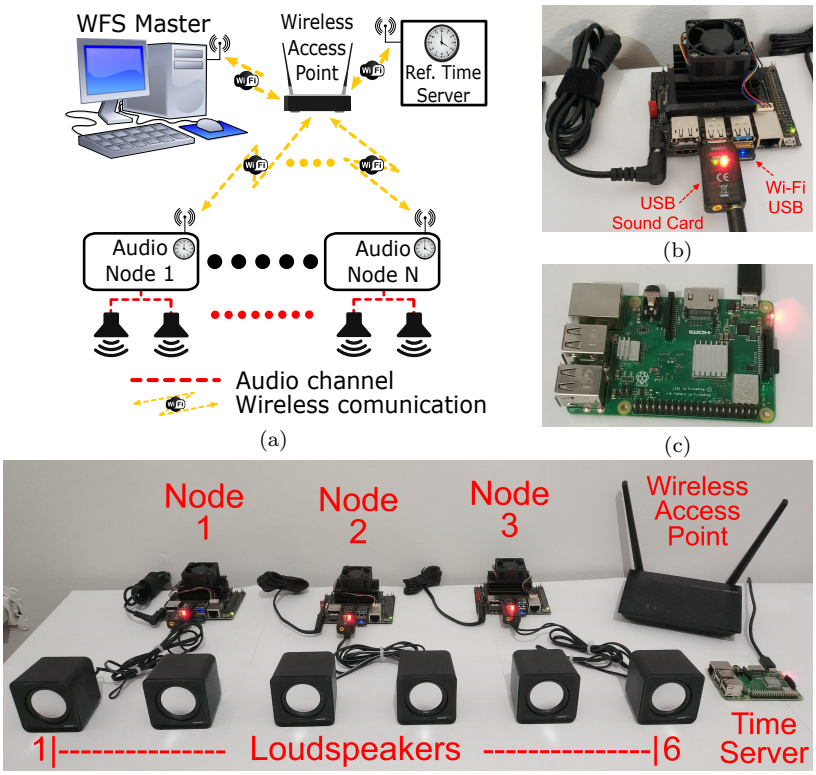


Fig. 2 Proposed distributed WFS architecture: (a) Interconnection diagram and elements. (b) Acoustic node based on NVIDIA® Jetson Nano™. (c) Reference time server based on Raspberry Pi 3B+. (d) Distributed WFS system prototype; three nodes and six loudspeakers.

3 Distributed system set-up and synchronization issues

As described on section 2.1, a signal delay among different speakers produces in wavefield synthesis an error in audio localization. Therefore, a reliable time synchronization mechanism is required in a distributed deployment. In order to tackle synchronization issues, we set a small distributed WFS system as a testbench (see Figure 2(a)) composed of the following components:

WFS Master: Its role is to send the audio samples to all WFS nodes.
Wireless Access Point: It is used for connecting the different network elements.
Reference Time Server: It broadcasts the reference clock to all WFS nodes.
Audio nodes: They are devoted to synthesize and reproduce the audio throughout their corresponding loudspeakers.

Figure 2(b) shows a detail of a WFS node based on a NVIDIA® Jetson Nano™ device with an external Wi-Fi and audio USB adapters. Additionally,

Figure 2(c) shows the Reference Time Server, implemented with Raspberry Pi 3B+. Finally, Figure 2(d) shows three WFS nodes.

The use of an external reference time server guarantees that all nodes are in the same tier level and are affected by the same delays. The reason for using an independent node is only to carry out the analysis at nodes without the possible effects of the time server service, and showing the ubiquitous character of the proposed solution. The time server could have been implemented in the WFS Master itself, or in another NVIDIA® Jetson Nano™. In this case, the election of a Raspberry Pi reduces the cost of the system, and also highlights the ubiquitous nature of the service.

A Client-Server application based on stream sockets was implemented to coordinate the different elements of the system. The server allows a dynamic interaction between the clients, indicating them when to start the reproduction of one or more sound sources on a given position; when to change the position of a source; or when to stop its reproduction. All clients read from a configuration file the number and location of the audio files; the number of loudspeakers and sound sources; as well as their positions in the system. Each node automatically computes its own outputs based on equations (2) and (3).

It is important to note that traditional centralized systems have a common clock source, so that synchronizing their DACs and audio data signals is straightforward. Conversely, the distributed strategy proposed in this work poses the problem that each node has its own clock. Hence, a synchronization mechanism between the nodes must be enforced. There are different synchronization procedures for networks in the literature, with the most used protocol for time synchronization being the Network Time Protocol (NTP) [17]. However, there are more accurate alternatives. The Precision Time Protocol (PTP) [18,19] is also a very widespread) option; and for applications with very high precision requirements, a synchronization through GPS signal is used [20]. Unfortunately, the last two options require additional hardware support. Also, even though PTP can be emulated via software, its performance gets significantly worse [21]. Due to this, we selected NTP as the synchronization protocol, using the standard package available in most Linux distributions (including Raspbian and NVIDIA L4T).

We initially carried out perceptual tests when the whole system was settled in order to evaluate the quality of the spatial sound. The people participating in these tests were able to locate easily the origin of the sound source. They highlighted the quality of the sound source movement which was really very smooth. Test results were similar to the tests of the centralized WFS system carried out in [3], so that the audio quality of distributed system resembles that of the centralized system.

3.1 Synchronization analysis

Several quantitative tests were carried out to test the usefulness of the proposed system as well as the effects of the synchronization. In these tests, the

final elements of the system (loudspeakers) were replaced by a multi-channel audio card, which allows us to acquire the different outputs from the WFS nodes; see Figure 3(a). Specifically, Figure 3(b) shows a small prototype with three nodes and an audio card used for the test.

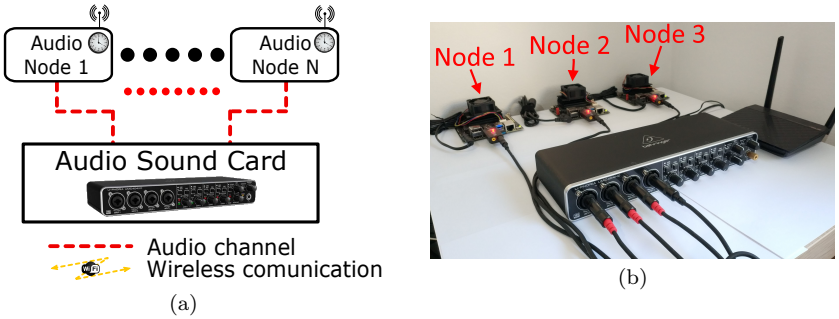


Fig. 3 Proposed synchronization test-beds: (a) Audio card connection to the WFS nodes under test. (b) Small prototype set-up with three WFS nodes.

To validate the NTP synchronization method, we configured all the nodes to generate the same audio pattern at their output. Then, we performed a correlation of each captured signal with the pattern signal in order to determine the delay between nodes. Figure 4 shows the delays between the three test-bed nodes for 3,000 test cycles. Each cycle plays a 10 second audio pattern (with a logarithmic sweep track), introducing a random pause of duration between 1 and 5 minutes after it.

As it is shown in Figure 4(a), the delay between nodes in all test was always under 10 milliseconds. Furthermore, by analyzing the dispersion of the results (see Figure 4(b)), we observes that 97.67%, 98.27% and 97.33% of the delays between nodes N1-N2, N1-N3 and N2-N3, respectively, were under 5 milliseconds.

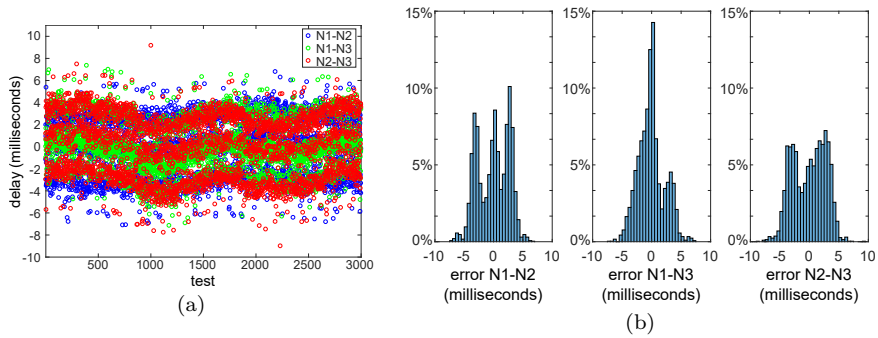


Fig. 4 Synchronization test results: (a) Recorded delays. (b) Delays distribution.

This quantitative analysis confirms the results obtained with the perceptual tests: The NTP protocol provides a precise synchronization method to implement a distributed WFS system.

4 Computational and energy performance

The NVIDIA Jetson Nano Developer Kit is a board that comprises a SoC based on a quad-core ARM Cortex-A57 CPU at 1.43 GHz with 4 Gbytes of LPDDR4 memory, and a 128-core Nvidia Maxwell GPU. One of the main features of this board is its low-power consumption combined with different levels of parallelism, which provides a very high performance-per-Watt. A second appealing feature of this platform is the possibility of adjusting its energy consumption by reducing the frequency of the CPU cores or the GPU. We briefly describe details regarding the GPU-based implementation as well as the energy-measurement procedure.

4.1 GPU-based implementation

According to the description carried out in Section 2, the implementation of the WFS requires to execute equations (2) and (3). Thus, given a real-time WFS system composed of M sound sources and N loudspeakers, the application begins with an input matrix of size (M x L), where L represents the audio samples that are captured by the audio card of the SoC (see Figure 5). The algorithm, applies N times a weight and a delay in the position of the samples to every row of the matrix, since we are dealing with N loudspeakers. To this end, a CUDA kernel (denoted as Kernel 1 in Figure 5) composed of $M \times N \times L$ threads is launched, given as a result a three-dimensional matrix of that size. This matrix corresponds to the operations given at the equation (2). Finally, we launch another CUDA kernel (denoted as Kernel 2 in Figure 5) composed of $N \times L$ threads that are in charge of generating the output samples that are sent to each of the N loudspeakers. More details of the implementation can be found in [3].

4.2 Energy-measurement procedure

The CPU power management strategy uses dynamic frequency scaling with dynamic voltage scaling. The CPU frequency is dynamically adjusted depending on how busy is the device. However, the GPU and CPU frequencies can be modified by using some features of the Linux kernel.

Dynamic Frequency Scaling is implemented by means of the Linux `cpufreq` subsystem. It is based on the use of governors that implement frequency scaling policies. Using the `cpufreq-set` command a specific frequency or a range of frequencies in a CPU can be chosen. For example,

```
cpufreq-set -c 0 -d 1224000 -u 1224000
```

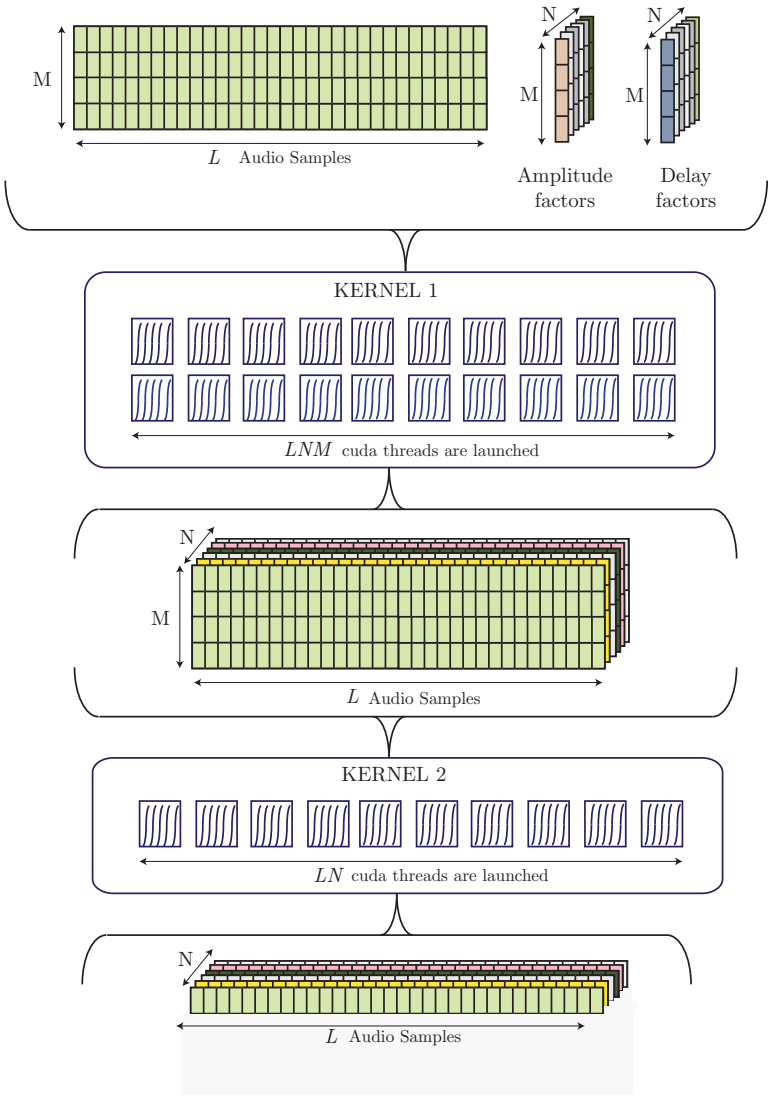


Fig. 5 Kernel 1 and Kernel 2 perform the parallel computation of the driving signals of a WFS system.

sets the frequency of core 0 to 1224 kHz. This change can be applied at runtime without having to restart the device. There is not an equivalent `gpufreq-set` command, but the root can change the frequency of the GPU by writing its minimum and maximum values in specific system files. For example:

`echo 3840>/sys/devices/57000000.gpu/devfreq/57000000.gpu/min_freq`
sets the minimum frequency to 384 MHz.

Besides, Jetson Nano supports two power modes, such as `5W` (5 watts) and `MaxN` (10 watts). Each mode determines the number of active cores (2 or 4 respectively), and the rank of frequencies allowed for the CPU and GPU, among other parameters. Users can also define their own custom mode. Power modes can be managed using the `nvpmodel` and `jetson_clocks` commands.

In our experiments, we measured the power dissipation of the Jetson Nano using the `pmlib` framework [22] to collect the instantaneous power readings from the internal energy-monitoring sensors [6]. The Jetson Nano board embeds real-time current sensors that can be sampled to obtain the power consumption of three separate power domains: main module, GPU and CPU.

Specifically, `pmlib` is a framework to trace and analyze the power and energy consumption made by applications. It uses a client-server paradigm and can interact both with internal and external powermeter devices to get the power dissipated by different components of the platform. First the application is instrumented to define the location of the server, set the power lines to measure, create counters or start and stop collecting data before and after some specific code sections, among other actions. When we run the application in the target platform, the tracing server steadily samples the power dissipation of the lines and saves the data into a file. In the case of the Jetson Nano platform, the three power lines are sampled every 0.125 seconds, and we store every individual and total power dissipation in a text file. Next we can analyze the data manually or using performance tracers or visualization tools.

4.3 Experimental evaluation

The following experiments evaluate the computational performance, energy consumption, and energy efficiency of the CUDA-based implementation of WFS and RC filter block proposed in [3] on one of the Jetson Nano devices shown in Figure 2(d). As a performance metric, we adopt the maximum number of sound sources that can be rendered in real time.

First, we analyze the evolution of the execution time of the CUDA algorithm with the number of sound sources and the effect of parameters such as the size of RC filters or the number of samples. Figure 6 shows that the execution time increases linearly with the number of sound sources and that we can process a large number of sound sources in real time. We use an audio buffer size of 256 samples, which implies a real-time threshold of 5.8 ms (sampling frequency of 44,1 kHz). The figure shows that the size of the applied RC filters only affects the execution time slightly and allows us to process around 100 sources in real time using filters with lengths that range from 256 to 1536 coefficients. It is important to highlight that there is an upper bound on the number of sound sources that can be processed due to the memory size of the GPU. As illustrated in the figure, the use of RC filters with a length of 1536 fills the GPU memory when the implementation handles more than 110 sound sources.

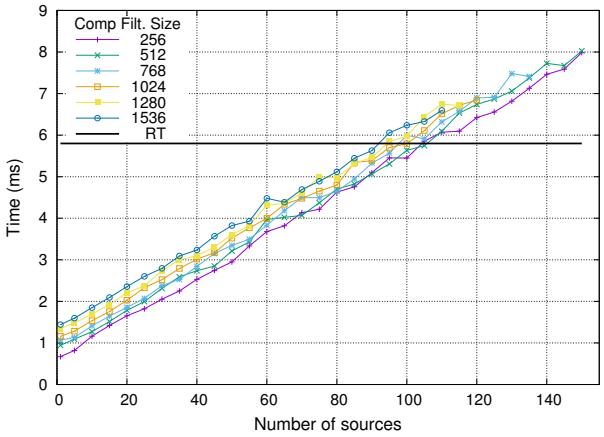


Fig. 6 Evolution of the execution time of the CUDA implementation with the number of sources using different sizes of the RC filters and an audio buffer of 256 samples at a sample frequency of 44.1 kHz. The horizontal line represents the time to process the sources in real time.

We may not be interested in processing a very large number of sound sources in real time, but instead in processing a smaller number of sound sources in real time while reducing the energy consumption. To this end we can diminish the frequency of the GPU as well as that of the CPU core that is used as host for the CUDA program. To obtain the results in the following figures, we used an audio buffer consisting of 256 samples, RC filters of size 1024 and 16 loudspeakers. For example, Figure 7 shows that we can process 10 sources in real time even when reducing substantially the GPU and CPU frequencies. Only if we set the lowest frequency of the GPU (76.8 MHz), we cannot process 10 sources in real time for any CPU frequency. However, we can reduce the frequency of the GPU to its second lowest value and process 10 sound sources in real time even with a very small CPU frequency: 403.2 kHz. In general terms, given a fixed GPU frequency, initially the execution time of the algorithm decreases quickly when we increase the CPU frequency from its lowest value. However, when we increase the CPU frequency above 403.2 kHz, the execution time decreases very slowly.

A deeper understanding of the effect of the GPU and CPU frequencies on the computational efficiency of the algorithm can be obtained by analyzing the maximum number of sound sources that can be processed in real time for every GPU-CPU frequencies combination. Figure 8 shows that the maximum number of sound sources increases both with the CPU-GPU frequencies. Once more, it increases very quickly for the lowest frequencies, and more slowly as we approach the highest frequencies.

We are not only interested in maximizing the number of sound sources that can be rendered in real time, but also in reducing the energy consumption of the system. Therefore, we next analyze the energy consumed by the different

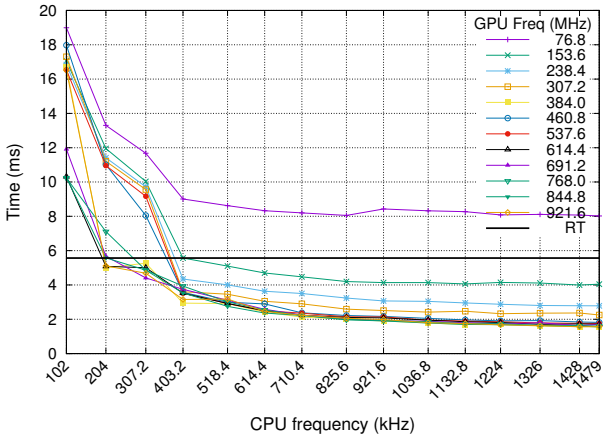


Fig. 7 Execution time of the CUDA implementation to process 10 sources with a buffer size of 256 samples and RC filters of size 1024. The horizontal line represents the time to process the sources in real time.

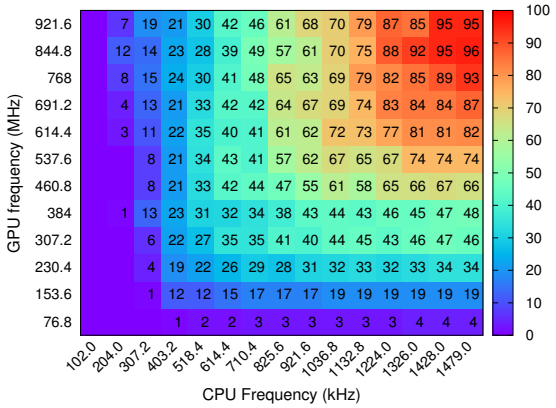


Fig. 8 Maximum number of sources processed in real time for the different combinations of GPU and CPU frequencies. Values are only shown when at least one source can be processed in real time.

configurations in order to maximize the sources-per-Watt that can be rendered in real time.

We obtain the energy consumed by the application by performing the product of the power dissipated by the whole platform and the execution time. In all cases, we report the "application energy consumption" obtained by subtracting the observed idle power at minimum CPU-GPU frequencies from the total energy consumption observed during the application execution. We prefer to analyze its energy efficiency by using the sources-per-Watt metric. Figure 9 shows that the maximum number of sources that we can process per Watt increases slowly with the CPU-GPU frequencies. This parameter is between 10 and 11 for an ample range of the highest frequencies of both components

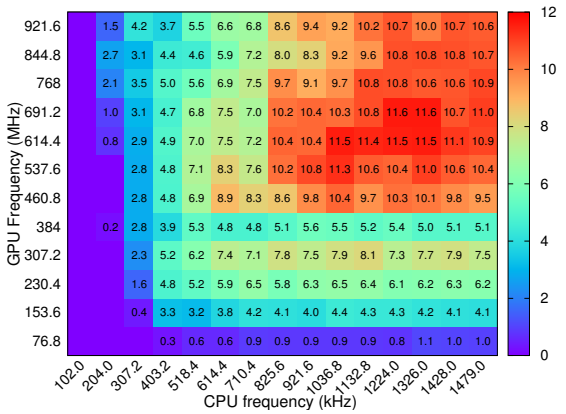


Fig. 9 Energy efficiency of the CUDA code computed as the maximum number of sources-per-Watt that can be processed in real time. Values are only shown when at least one source can be processed in real time.

of the platform, which shows that it is not necessary to use the highest CPU-GPU frequencies to maximize the energy efficiency of the WFS application in the Jetson Nano.

5 Conclusions

In this paper we have shown that it is possible to realize a flexible and efficient distributed spatial audio system based on WFS using low-power SoC devices as audio nodes. Specifically, we have leveraged the GPU of several Jetson Nano boards to distribute the audio processing of the sound sources.

We have shown that the application of the NTP synchronization protocol at the acoustic nodes allows us to render spatial sound with sufficient quality, as asserted by the subjective tests and the delay measurements among nodes.

A thorough analysis of the performance of the Jetson Nano device shows that it can process more than 100 sound sources in real time even using room compensation filters with a huge number of coefficients. We have leveraged the possibility of reducing the frequencies of the GPU and CPU components of the Jetson Nano to reduce the energy consumption of the application. Employing lower frequencies increases very slowly the processing time. The results show that we can reach the maximum number of sources-per-Watt even reducing substantially the frequencies of both components.

Acknowledgements

This work has been supported by the Spanish Government through TIN2017-82972-R, ESP2015-68245-C4-1-P, the Valencian Regional Government through PROMETEO/2019/109 and the Universitat Jaume I through UJI-B2019-36.

References

1. A. Berkhout, "A holographic approach to acoustic control," *J. Audio Engineering Society*, vol. 36, pp. 2764–2778, May 1988.
2. D. Theodoropoulos, G. Kuzmanov, and G. Gaydadjiev, "Multi-core platforms for beamforming and Wave Field Synthesis," *IEEE Transactions on Multimedia*, vol. 3, no. 2, pp. 235–245, April 2011.
3. J. A. Belloch, A. Gonzalez, E. S. Quintana-Ortí, M. Ferrer, and V. Välimäki, "GPU-Based Dynamic Wave Field Synthesis Using Fractional Delay Filters and Room Compensation," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 25, no. 2, pp. 435–447, Feb 2017.
4. S. Spors, H. Buchner, and R. Rabenstein, "Efficient active listening room compensation for Wave Field Synthesis," in *Proceedings of the 116th AES Convention*, Berlin, Germany, May 2004.
5. J. Lopez, A. Gonzalez, and L. Fuster, "Room compensation in wave field synthesis by means of multichannel inversion," in *IEEE workshop on Applications of Signal Processing to Audio and Acoustics, 2005.*, oct. 2005, pp. 146 – 149.
6. NVIDIA Corp., "NVIDIA Jetson Linux Developer Guide. PR-06076-R32," May 2020.
7. G. Fabregat, J. A. Belloch, J. M. Badía, and M. Cobos, "Design and implementation of acoustic source localization on a low-cost iot edge platform," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 67, no. 12, pp. 3547–3551, 2020.
8. Expressif Systems, *ESP32 Technical Reference Manual. Version 4.0*, Expressif Inc., 2018.
9. J. A. Belloch, J. M. Badía, F. D. Igual, A. Gonzalez, and E. S. Quintana-Ortí, "Optimized fundamental signal processing operations for energy minimization on heterogeneous mobile devices," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 65, no. 5, pp. 1614–1627, 2018.
10. T. Rhee, S. Thompson, D. Medeiros, R. dos Anjos, and A. Chalmers, "Augmented virtual teleportation for high-fidelity telecollaboration," *IEEE Transactions on Visualization and Computer Graphics*, vol. 26, no. 5, pp. 1923–1933, 2020.
11. J. A. Belloch, G. Ramos, J. M. Badia, and M. Cobos, "An efficient implementation of parallel parametric hrtf models for binaural sound synthesis in mobile multimedia," *IEEE Access*, vol. 8, pp. 49 562–49 573, 2020.
12. D. Puccinelli and M. Haenggi, "Wireless sensor networks: applications and challenges of ubiquitous sensing," *Circuits and Systems Magazine, IEEE*, vol. 5, no. 3, pp. 19–31, 2005.
13. W.-P. Chen, J. Hou, and L. Sha, "Dynamic clustering for acoustic target tracking in wireless sensor networks," *Mobile Computing, IEEE Transactions on*, vol. 3, no. 3, pp. 258–271, 2004.
14. A. Berkhout, D. de Vries, and P. Vogel, "Acoustic control by wave field synthesis," *J. Acoustic. Soc. Amer*, vol. 93, pp. 2764–2778, May 1993.
15. S. Spors, A. Kuntz, and R. Rabenstein, "An approach to listening room compensation with wave field synthesis," in *Proc. 24th AES Conference*, Banff, Canada, May 2003.
16. L. Fuster, J. J. Lopez, A. Gonzalez, and P. Faus, "Time and frequency domain room compensation applied to wave field synthesis," in *Proc. Conference on Digital Audio Effects (DAFx-05)*, Madrid, Spain, September 2005.
17. D. Mills, J. Martin, J. Burbank, and W. Kasch, "RFC 5905: Network time protocol version 4: Protocol and algorithms specification," *Internet Engineering Task Force*, 2010.
18. "IEEE standard for a precision clock synchronization protocol for networked measurement and control systems," *IEEE Std 1588-2019 (Revision of IEEE Std 1588-2008)*, pp. 1–499, 2020.
19. T. Neagoe, V. Cristea, and L. Banica, "NTP versus PTP in com puter networks clock synchronization," in *2006 IEEE International Symposium on Industrial Electronics*. IEEE, jul 2006.
20. M. A. Lombardi, L. M. Nelson, A. N. Novick, and V. S. Zhang, "Time and frequency measurements using the global positioning system," *Cal Lab: International Journal of Metrology*, vol. 8, no. 3, pp. 26–33, 2001.

21. P. Fubin, Y. Yubo, G. Lei, and S. Liangliang, "The accuracy of IEEE 1588 time synchronization protocol and its improvement," in *2015 12th IEEE International Conference on Electronic Measurement & Instruments (ICEMI)*. IEEE, jul 2015.
22. S. Barrachina, M. Barreda, S. Catalán, M. F. Dolz, G. Fabregat, R. Mayo, and E. Quintana-Ortí, "An integrated framework for power-performance analysis of parallel scientific workloads," *Energy*, pp. 114–119, 2013.