






Performance Comparison of Database Server based on SoC FPGA and ARM Processor

Víctor Asanza , Rebeca Estrada , Jocelyn Miranda , Leiber Rivas , Danny Torres 
Escuela Superior Politécnica del Litoral, Espol (FIEC & CTI), Guayaquil, Ecuador
{vasanza, restrada, daaltorr, jocammir, lvrivas}@espol.du.ec

Abstract—New emerging storage technologies have a great application for IoT systems. Running database servers on development boards, such as Raspberry or FPGA, has a great impact on effective performance when using large amounts of data while serving requests from many clients at the same time. In this paper, we describe the design and implementation of a MySQL database server that runs on Linux in a standard FPGA DE10 using HPS resources, which is oriented to an IoT system in charge of monitoring and controlling the temperature inside greenhouses. For comparison purposes, we carried out a performance analysis of the service running on the FPGA and in a Raspberry Pi 4 B to determine the efficiency of the data server in both development cards. The performance metrics analyzed were response time, memory and CPU usage taking into account scenarios with one or more requests from clients simultaneously.

Index Terms—IoT, FPGA, HPS, Database server

I. INTRODUCTION

Embedded systems allows us to design custom architectures using specific hardware and software resources, which implies the reduction of power consumption and the increase of flexibility.

In this paper, we designed and implemented an embedded system to monitor a database server using the Terasic DE10 Standard board that integrates Field Programmable Gate Array (FPGA) and Hard Processor System (HPS) technology based on ARM. It contains peripherals, memory interfaces and a Dual Core ARM Cortex-A9 MPCore processor [1].

The MySQL database server is implemented in the Linux operating system embedded in the Hard Processor System (HPS) and managed by remote access in a local network [2]. This handles temperature, light, air humidity and percentage humidity values. This information was collected by the respective sensors housed in a distributed system focused on the control and monitoring of a greenhouse.

The FPGA has the role of client of the server, it makes inquiries to the database when the user indicates it through its input peripherals. When one of them is activated, a file that returns the result of the MySQL request is executed, these values are stored in the shared memory existing between FPGA and HPS to later be presented on the screen by means of the VGA protocol [3].

The rest of the document is organized as follows. Works related to the project are presented in Section II. Section III analyzes the performance of the database server based on the complexity and concurrency of the requests. In Section IV explains the general operation of the embedded system. In section V, the performance of the embedded system implemented in the Terasic DE10 Standard board is discussed and compared to performing the same implementation in another micro-controller such as Raspberry Pi 4 B. Finally, Section VI details important points of the entire document.

II. RELATED WORK

FPGAs are capable of having an embedded Linux version, having several IoT applications, such as a database server, web server, DNS server, traffic analyzer, among others. As for work done on a database server running on Linux on an FPGA, the performance of this system is always important, as it must be able to serve many customers with optimal resource consumption. Benchmark tests are performed to measure performance, Lee et Al. [4] performed benchmark tests with SQLite, evaluating the system with the use of DRAM/PRAM hybrid memories. In IoT applications it is important to use some physical peripheral for the drive of some other system, work related to this is shown in the job [5], which shows the configuration and interaction that can occur between the HPS part, FPGA with peripherals such as LEDs or own switches a DE1-SoC FPGA with an ARM Cortex-A9 processor. Wielgosz et Al. [6] having an optimal latency level in a database system is indispensable in its operation, Costas et Al. [7] seek to improve the latency and throughput of a database system, accelerating the processing of queries for handling large amounts of data, Ito et Al. [8] which describes a system that optimizes the performance of a database system, demonstrating performance and energy consumption improvements when using an FPGA. Lee et Al. [4] proposed to design and implement a database system in MySQL, whose service runs on Ubuntu 16.04 in a 10 Standard FPGA. This system interacts with the hardware of this card, allowing the presentation of the result of a specific query by VGA when triggering switches [9, 10]. A comparative analysis of the performance of this system is also shown running in the FPGA against the same system running on an

Raspberry Pi 4 B, to determine which one is most efficient in processing requests from multiple customers to have a real application, which can be on an IoT system by including hardware interaction and sensor data processing.

TABLE I
DESCRIPTION OF FPGA HARDWARE TERCASIC DE10 - STANDARD AND RASPBERRY PI 4 B USED IN COMPARATIVE ANALYSIS

Features	HPS-FPGA	Raspberry Pi 4 B
Total memory	750 MB	8 GB
Active memory with no requests to the system	155 MB	285 MB
Architecture	armv7l	Cortex-A72 (ARM v8)
Number of CPU(s)	2	4
Socket(s)	1	1
Core(s) by socket	2	4
CPU model	925 MHz Dual core ARM Cortex A9 MPCore processor	1.4GHz 64-bit quad-core processor

III. DATASET

The dataset to analyze the performance of the MySQL database system in the FPGA and compare it with that of a Raspberry Pi 4 B, was obtained by making different requests to the system running on Ubuntu 16.04 and getting the response time, CPU and RAM consumption for different client numbers. For time capture the Mysqslap benchmarking tool was used and for CPU and RAM consumption, a bash script called rendimiento.sh that makes multiple requests to the database and captures CPU and RAM consumption when running that new process. The data obtained are based on the hardware specifications, both for the Raspberry and the FPGA, this is shown in Table I.

The tasks executed by the rendimiento.sh script are as follows:

- MySQL service startup.
- Write the current CPU and RAM consumption.
- Start n request processes to the database system, simulating clients by querying. The result of each query is written to a text file.
- Write the new CPU and RAM consumption.
- Stop request processes to the database system.
- Stopping the MySQL service.

The performance of the database system was evaluated with different queries, changing its complexity, which are shown in Table II.

With the Mysqslap tool, benchmark tests were run with 100 iterations each, varying the concurrent number of clients between 10, 20, 30, 40 and 50. From this tool and with the script performance.sh, the data shown in Table III, where the response time, percentage of CPU and memory usage for the different queries and number of queries shown are presented.

TABLE II
QUERIES USED TO EVALUATE DATABASE SYSTEM PERFORMANCE ON THE FPGA AND RASPBERRY PI 4 B

Identifier	Query
Query1	select soil,node_id from measures limit 10;
Query2	select temperature, humidity, light, node_id from measures where temperature >'30';
Query3	select temperature from measures where node_id=1 order by dtm desc limit 10;
Query4	select telecontrol.id, node_id, crop_id from telecontrol inner join nodes on nodes.id = telecontrol.node_id where nodes.id=1;
Query5	select temperature, soil from history where temperature > '25' and soil = 100 limit 8;

IV. METHODOLOGY

This section presents a system overview, data acquisition, the architecture used, implementation of both hardware and software.

A. System overview

The development of the project consists of four stages, the first is the process of acquiring data through specialized devices present in the Greenhouse project; the next stage, focuses on deploying a database server on an embedded system; Subsequently the configuration of the architecture for that system is carried out, this will allow to have as a client of the server to the FPGA; and finally in the fourth stage, the configuration of the same database server is developed in a general purpose system. A comparative analysis of MySQL server performance was performed between a custom architecture system and a system with an already predefined architecture; where, a DE10 Standard card and a Raspberry PI B+ were used respectively. The first device mentioned was used for the development of the embedded system, because it is one of the most robust equipment within Prototyping platforms [9]. Next, proceed to upload the Linux operating system image to a Micro SD card; the configuration of the MySQL database server embedded in the HPS was performed and used in conjunction with the FPGA. The database containing the collected greenhouse values must be exported and then imported into the Linux embedded system, implemented in the HPS as mentioned previously.

The proposed architecture incorporates components to build a basic embedded system such as: System ID, Arria V/Cyclone V Hard Processor System, NIOS II Processor, Interval Timer, Avalon, On-Chip Memory y PIO (Switches); to which the necessary blocks for VGA communication are added, which will be described in detail in the subsection IV-C . Once the specialized architecture is implemented, access to memory and different peripherals such as buttons, LEDs, switches is obtained. The first three switches are used to query the database server and the last one for displaying the data on the screen. When one of the three is activated, a file that returns the result of the MySQL request is executed, these values are stored in the shared memory existing between FPGA and HPS and then presented on screen via the VGA protocol.

TABLE III
DATABASE SYSTEM PERFORMANCE ON THE FPGA AND RASPBERRY PI 4
B WITH 10, 20, 30, 40 AND 50 REQUESTS AT THE SAME TIME

Using Query1						
Number of requests	10	20	30	40	50	
Time response Raspberry Pi (ms)	0.006	0.013	0.019	0.025	0.034	
Response time HPS (ms)	0.011	0.023	0.038	0.047	0.058	
% Memory Raspberry Pi	1.10	1.10	1.20	1.20	1.24	
% Memory HPS	1.40	6.60	6.70	6.80	6.90	
% CPU Raspberry Pi	4.24	5.30	5.82	5.86	6.34	
% CPU HPS	6.64	8.68	9.62	8.44	11.66	
Using Query2						
Number of requests	10	20	30	40	50	
Time response Raspberry Pi (ms)	0.025	0.049	0.068	0.090	0.111	
Response time HPS (ms)	0.047	0.095	0.143	0.190	0.238	
% Memory Raspberry Pi	1.00	1.20	5.50	5.60	5.70	
% Memory HPS	1.40	6.60	6.70	6.80	6.90	
% CPU Raspberry Pi	12.58	19.00	25.36	23.98	28.28	
% CPU HPS	11.98	9.70	9.92	6.30	16.50	
Using Query3						
Number of requests	10	20	30	40	50	
Time response Raspberry Pi (ms)	0.033	0.062	0.092	0.122	0.153	
Response time HPS (ms)	0.072	0.143	0.216	0.288	0.36	
% Memory Raspberry Pi	1.06	1.26	3.86	5.60	5.72	
% Memory HPS	2.24	6.68	6.88	7.00	7.16	
% CPU Raspberry Pi	27.54	30.38	31.50	35.88	35.26	
% CPU HPS	10.10	10.46	6.94	11.60	15.94	
Using Query4						
Number of requests	10	20	30	40	50	
Time response Raspberry Pi (ms)	0.007	0.016	0.024	0.031	0.039	
Response time HPS (ms)	0.013	0.028	0.042	0.056	0.07	
% Memory Raspberry Pi	1.06	1.10	1.20	1.26	1.30	
% Memory HPS	4.56	6.70	6.90	7.00	7.20	
% CPU Raspberry Pi	6.04	7.78	8.12	7.66	9.20	
% CPU HPS	9.86	7.54	8.30	11.54	11.42	
Using Query5						
Number of requests	10	20	30	40	50	
Time response Raspberry Pi (ms)	0.019	0.034	0.049	0.066	0.082	
Response time HPS (ms)	0.035	0.070	0.105	0.140	0.175	
% Memory Raspberry Pi	1.00	1.12	1.20	1.26	1.28	
% Memory HPS	1.40	6.60	6.74	6.90	7.00	
% CPU Raspberry Pi	18.22	19.10	19.54	19.78	21.18	
% CPU HPS	10.64	13.04	9.50	11.28	16.34	

B. Data Acquisition

For data collection by the isolated system, DHT22, YL-69 and light sensor sensors were used. Same system that has SLAVES nodes capable of collecting the values perceived by the sensors, and then sending them to a GATEWAY node, which is configured on a Raspberry. The Raspberry PI B+ micro-controller forwards this data in JSON format to the Back-End designed specifically for the control and monitoring system of a greenhouse, http requests are communicated with the database server configured on a computer on a local network. The MySQL server performs the function of storing temperature, light, soil moisture and percentage humidity data. Subsequently, this data is consulted by the graphical interface (Front-End), designed in the Vuejs framework, based on the

needs of the greenhouse, allowing the user to provide a feasible and effective interaction when viewing the data in real time.

The database (telemetry_greenhouse) used in this system is described in the Entity-Relationship diagram, Figure 1

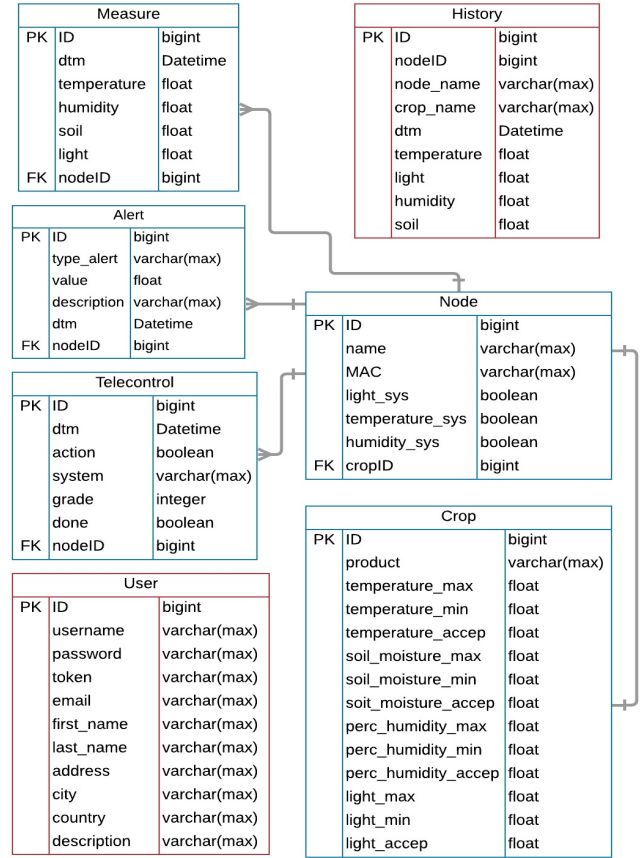


Fig. 1. Entity-Relation diagram of database telemetry_greenhouse

C. Architecture

The implementation of client-database server system on the DE10 Standard development card requires the use of the following components:

- System ID
- Interval timer
- On Chip Memory
- PIO Controller
- NIOS II Processor
- USB Blaster
- JTAG UART
- Clocks
- Avalon Switch Bridge
- Counter
- Frequency divider
- Character Buffer
- Dual Clock FIFO
- VGA Controller
- SDRAM Controller

- AXI Bridge
- SD Card Controller
- Dual Core ARM Cortex A9 MPCore Processor
- Triple Speed Ethernet
- 1Gigabit Ethernet PHY
- SDRAM 1GB DDR3

The Qsys platform was used to make the connection, which exclusively harnesses the power of the Intel processor, real-time, has a robust and reliable Linux Kernel design for specific purposes [10]. It has been proposed to use the basic blocks for an architecture in conjunction with VGA controllers to present relevant information on the screen. The FPGA has a controller block for the input signals of the switches, called the PIO Controller. In addition, the necessary blocks for the processing of information are added, SYSTEM ID, INTERNAL TIMER, NIOS II PROCESSOR y ON CHIP MEMORY. The SYSTEM ID block verifies that the executable program is compiled with the current Hardware image configured in the FPGA; On the other hand, the INTERNAL TIMER block controls the run-time intervals in the system. ON CHIP MEMORY works in conjunction with NIOS II PROCESSOR, this memory allows to store the data and provide flow control of the SOPC Builder system; the second block mentioned, has the function of preparing the output frames by means of VGA protocols. The VGA Controller block synchronizes the output of frames for the correct presentation of the data by means of a VGA-connected display on the respective port. To interrelate the above blocks, the AVALON WITCH FABRIC block is used, which serves to provide communication and interconnection between blocks.

On the HPS side it uses the SD CARD SOCKET block, which represents a Linux server port used for information exchange. This will be controlled by the SD CARD CONTROLLER block, which behaves like a signal receiver by the analog/digital converter, signals that are processed by the Core (The Core is responsible for generating the screen resolution and storing the sensor data). The DUAL CORE ARM CORTEX A9 MPCORE PROCESSOR block will be directly taken care of the database server and will work in conjunction with the SD CARD CONTROLLER. On the other hand, the Controller and TRIPPLE-SPEED ETHERNET SDRAM blocks are available; the first stores the data for early processing of the same and in turn interconnects with the DDR3 GB SDRAM block, which was considered to improve the data transfer flow and execute more quickly the number of instructions given by the server. The TRIPPLE-SPEED ETHERNET block is characterized by having multiple PHY speeds, this is interconnected with the block 1 GB ETHERNET PHY, which represents an Ethernet interface and is where the system will connect to a router to access the network, this block connects at once to RJ-45 PORT, which represents the space where the physical connection to the network will be made. All the blocks mentioned are interrelated by the AXI BRIDGE block, this block allows communication between them in the HPS.

The USB BLASTER block, is the USB port of the FPGA

card communicates with the JTAG UART block, which enables the creation of the embedded system in the development card and processor programming, to properly operate the FPGA together with the HPS. The Figure 2 shows a diagram describing the interconnections of each aforementioned block.

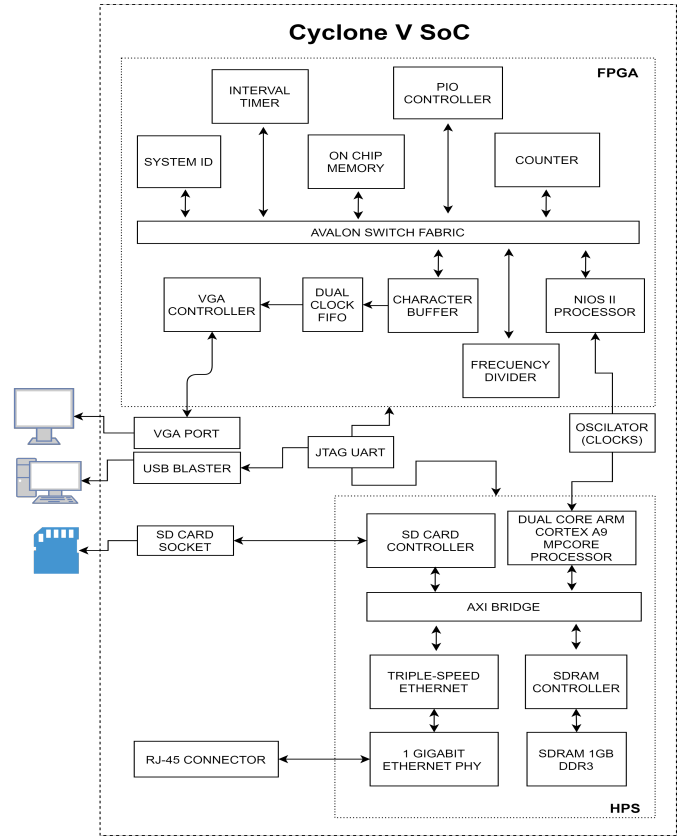


Fig. 2. Architecture System

Counter and Frequency divider are own blocks developed in VHDL in order to keep track of the time that will be displayed on the screen the result of a query by means of a 1 Hz CLOCK.

After you build and build the project architecture, two files are created, one of them is a compiled binary format (dtb) generated by Device Tree in order to avoid competition for resources as it allows these and loading modules. A Device Tree Blob (dtb) is a data structure that describes the hardware components of a particular device so that the the operating system kernel can use it and manage its components. The other file with RBF extension, allows you to have a linear sequence of operations to be executed by the Operating System.

These files replace those that come by default on the micro SD card that has the Linux image loaded so that, when you boot the system, our architecture design is implemented by default and thus have access to all the added components.

D. Configuring the server in the HPS

The database (telemetry_greenhouse) used in the control and monitoring system of a greenhouse, Figure 3, it is also used on our database server once all the package needed to

successfully run the MySQL service is installed. To do this, once you boot the operating system, the server is configured so that it has access to the Internet as it needs to configure the dependencies of the mentioned service. This process is described in Figure 3 and are summarized in the following steps:

- Configure an IP Address, Subnet Mask, and Gateway in the Network file
- Set up a default gateway
- Configure DNS in the resolv.cnf file
- Test connectivity with a ping
- Install MySQL

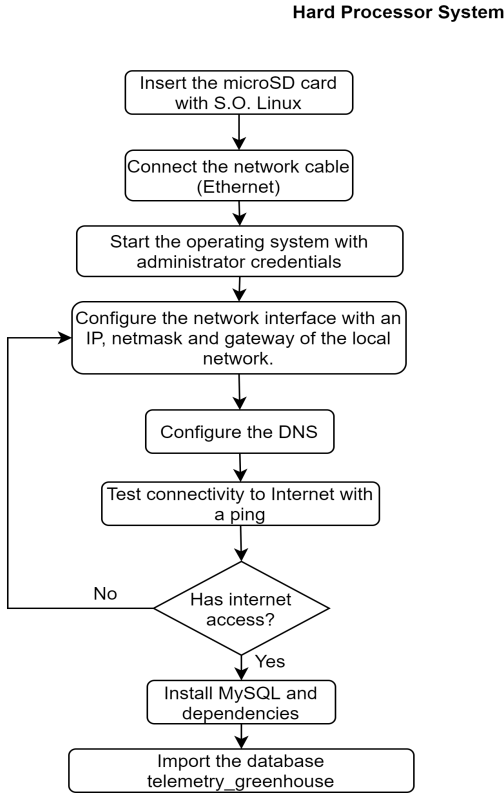


Fig. 3. Configuring a database server in the Hard Processor System

E. General operation of the system

On the Linux operating system runs the HPS_FPGA file generated from a C script, this allows constant reading of memory to know if any of the switches are in the active state or not. If one of the first switches is active, a file is running in bash, which will consequently run another file, which has been named MAIN. This file contains the basic functionalities to make a request to the server and once the server is finished running, it will return the result as integer values written to a generated text file. If the last switch (SW9) is activated accordingly, the read of the generated text file will run and the query result is saved in memory. Thanks to this action and the programming of the NIOS II processor, it is possible to read in memory the previously stored result, finally present it

by screen to the user. This process is described in Figure 4 and repeats for the remaining switches.

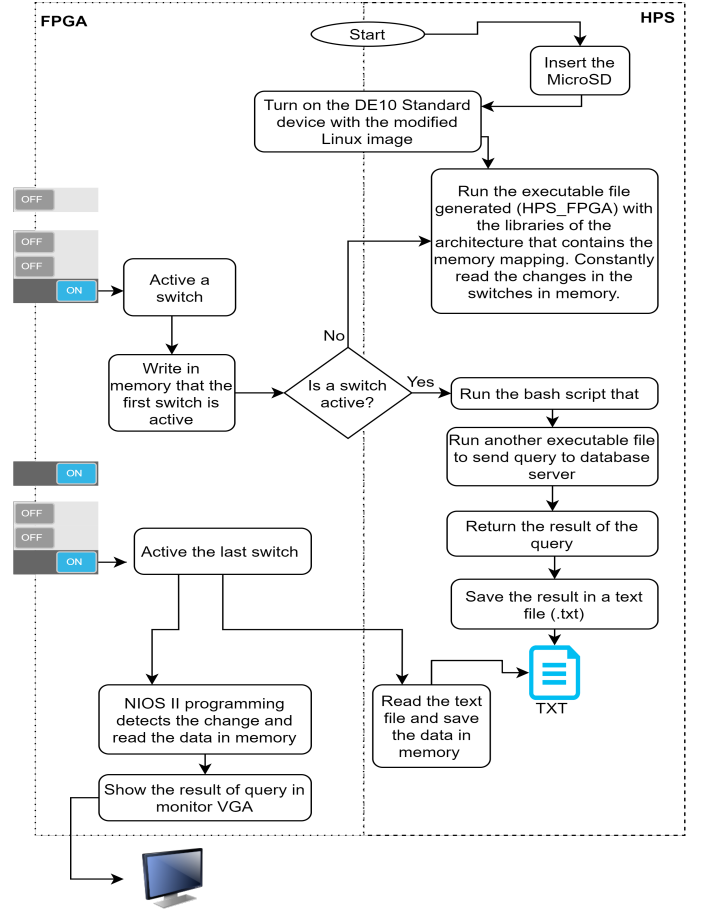


Fig. 4. General Operation of the System Diagram

V. RESULTS

With each query, different results were obtained, either in quantity records or number of fields obtained, shown in Table IV.

When considering the response time for each query to determine its complexity, ordering the data obtained according to this and making a comparison of the response time, percentage of CPU and memory usage, this is shown in Table V.

According to the data shown in Table IV, it can be determined that the response time with the different queries used is better in the FPGA than in the Raspberry Pi 4 B considering a single client making requests to the database system. But analyzing the percentage consumption of memory and CPU, the Raspberry has a better performance. The same result can be observed in the data of Table III - Query 1 shown, the Raspberry being better used as a simple query such as Query1, which has a low response time for both development cards. But in Table III - Query 2, 3 and 5 the results change, since Raspberry has a better memory usage and time response, but has a higher percentage of CPU usage than FPGA. This is a more complex request such as Query2,

TABLE IV
RESULTS OBTAINED WITH EACH QUERY PERFORMED

Query	Result	Number of registers
Query1	<pre> +-----+-----+ soil node_id +-----+-----+ 17.00 1 +-----+-----+ </pre>	10
Query4	<pre> +-----+-----+-----+ id node_id crop_id +-----+-----+-----+ 1 1 2 +-----+-----+-----+ </pre>	1
Query5	<pre> +-----+-----+ temperature soil +-----+-----+ 27.60 100.00 +-----+-----+ </pre>	8
Query2	<pre> +-----+-----+-----+-----+ temperature humidity light node_id +-----+-----+-----+-----+ 30.60 72.00 35.00 1 +-----+-----+-----+-----+ </pre>	40
Query3	<pre> +-----+ temperature +-----+ 27.20 +-----+ </pre>	10

Query3 and Query5, since they require an ordering of the data consulted in descending order, only obtain the first 10 resulting records and use logical operators such as the AND operation to search for records. Although the Raspberry has 4 CPUs, the FPGA uses its processing resources more efficiently to give a better response time in the client's requests, having a lower CPU use in complex processing although it uses more memory. This data can also be verified in Table IV, where it can be observed that the memory consumption does not increase with the complexity of the client's request processing, only the CPU usage and the response time vary; but the increase in the number of clients increases the amount of memory used, according to Table III - Query 1, 2, 3, 4 and 5.

TABLE V
COMPARISON IN THE PERFORMANCE OF THE DATABASE SERVER IN THE FPGA AND RASPBERRY PI 4 B WITH ONE CLIENT

Query	HPS - FPGA			Raspberry Pi 4 B		
	% CPU	% Memory	Time response (ms)	% CPU	% Memory	Time response (ms)
1	4.00	1.30	1.0776	0.20	1.10	1.2402
4	5.62	1.30	1.4269	0.38	1.10	1.8489
5	6.26	1.30	6.0800	2.00	1.10	9.2277
2	2.98	1.30	8.7325	1.90	1.10	14.1183
3	5.40	1.30	14.0245	2.22	1.10	20.5611

VI. DISCUSSION AND CONCLUSIONS

The FPGA is a development card capable of running a Linux version and functioning as a server for different services, like a database server in MySQL. For this reason, this device can be

used in real-world applications that involve the use of sensors and where the measured values need to be stored.

Using an FPGA as a database server is efficient if you will have multiple clients simultaneously making requests to the system, because of its hardware capacity, it provides good response times without excessive CPU consumption, unlike other development cards such as a Raspberry, and regardless of the complexity of the application processing.

Benchmarking tools are indispensable for a comparative analysis of the operation of a service on 2 different development cards. In this way, it is possible to determine which card is the most optimal to use according to the application required.

REFERENCES

- [1] V. A. Armijos, N. S. Chan, R. Saquicela and L. M. Lopez, "Monitoring of system memory usage embedded in FPGA," 2020 International Conference on Applied Electronics (AE), 2020, pp. 1-4, doi: 10.23919/AE49394.2020.9232863.
- [2] Haron, A. S., Talip, M. S. A., Khairuddin, A. S. M., Izam, T. F. T. M. N. (2017, December). Internet of things platform on ARM/FPGA using embedded linux. In 2017 International Conference on Advanced Computing and Applications (ACOMP) (pp. 99-104). IEEE.
- [3] Papaphilippou, P., Luk, W. (2018, August). Accelerating database systems using fpgas: A survey. In 2018 28th International Conference on Field Programmable Logic and Applications (FPL) (pp. 125-1255). IEEE.
- [4] Lee, J. H., Zhang, H., Lagrange, V., Krishnamoorthy, P., Zhao, X., Ki, Y. S. (2020). SmartSSD: FPGA Accelerated Near-Storage Data Analytics on SSD. IEEE Computer Architecture Letters.
- [5] Preethi, D., KG, D. (2020). FPGA Based Hardware Accelerator for Data Analytics: An Overview. K, Dr. Thivakaran, FPGA Based Hardware Accelerator for Data Analytics: An Overview (July 31, 2020).
- [6] Wielgosz, M., Karwatowski, M. (2019). Mapping neural networks to fpga-based iot devices for ultra-low latency processing. Sensors, 19(13), 2981.
- [7] Costas, L., Fernández-Molanes, R., Rodríguez-Andina, J. J., Fariña, J. (2017, March). Characterization of FPGA-master ARM communication delays in zynq devices. In 2017 IEEE International Conference on Industrial Technology (ICIT) (pp. 942-947). IEEE.
- [8] Ito, S. A., Carro, L. (2000, September). A comparison of microcontrollers targeted to FPGA-based embedded applications. In Proceedings 13th Symposium on Integrated Circuits and Systems Design (Cat. No. PR00843) (pp. 397-402). IEEE.
- [9] Cedeño, C., Cordova-Garcia, J., Asanza, V., Ponguillo, R., Muñoz, L. (2019, August). k-NN-Based EMG Recognition for Gestures Communication with Limited Hardware Resources. In 2019 IEEE SmartWorld, Ubiquitous Intelligence Computing, Advanced Trusted Computing, Scalable Computing Communications, Cloud Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCOM/IOP/SCI) (pp. 812-817). IEEE.
- [10] V. Asanza, A. Constantine, S. Valarezo and E. Peláez, "Implementation of a Classification System of EEG Signals Based on FPGA," 2020 Seventh International Conference on eDemocracy eGovernment (ICEDEG), 2020, pp. 87-92, doi: 10.1109/ICEDEG48599.2020.9096752.