Universidade Federal do Rio Grande do Sul
Instituto de Informática
Grupo de Processamento Paralelo e Distribuído

Proceedings

XVII Workshop de Processamento Paralelo e Distribuído

WSPPD 2019

13 de Setembro de 2019
Auditorio Centro de Eventos do Instituto de Informática
Porto Alegre, RS
ISSN: 2175-6848

List of Sessions

I/O Workload Characterization on Supercomputers using Unsupervised Learning

Pablo J. Pavan[1], Jean Luca Bez[1], Matheus S. Serpa[1], Francieli Zanon Boito[2], Philippe O. A. Navaux[1]
[1]Institute of Informatics, Federal University of Rio Grande do Sul (UFRGS) — Porto Alegre, Brazil
[2]Univ. Grenoble Alpes, Inria, CNRS, Grenoble INP, LIG, 38000 Grenoble, France
{pjpavan, jean.bez, msserpa, navaux}@inf.ufrgs.br, francieli.zanon-boito@inria.fr

***Abstract*—I/O operations are the bottleneck for an increasing number of applications due to the difference between the processing and data access speeds. The application access patterns reflect directly on their performance. Therefore, it is essential to characterize the I/O workload of large systems in order to identify problems and propose solutions. With this premise, we propose an I/O characterization approach that uses unsupervised learning to cluster jobs with similar I/O behavior using information from high-level aggregated traces. We apply our proposal to four months of activity — a total of 28.938 jobs — from the Intrepid machine at Argonne Laboratory. We show that our approach helps identify the most representative behaviors among the applications.**

***Index Terms*—Application I/O Behavior, Parallel I/O, I/O workload characterization, unsupervised learning, clustering algorithms**

I/O Access Pattern Detection of HPC Workloads at Runtime

Jean Luca Bez[1], Gessica F. M. Azevedo[1], Francieli Zanon Boito[2], Ramon Nou[3], Alberto Miranda[3], Toni Cortes[3,4], and Philippe O. A. Navaux[1]

[1]Institute of Informatics, Federal University of Rio Grande do Sul (UFRGS) — Porto Alegre, Brazil
[2]Univ. Grenoble Alpes, Inria, CNRS, Grenoble INP, LIG, 38000 Grenoble, France
[3]Barcelona Supercomputing Center (BSC) — Barcelona, Spain
[4]Universitat Politècnica de Catalunya — Barcelona, Spain

{jean.bez,gessica.azevedo,navaux}@inf.ufrgs.br,
francieli.zanon-boito@inria.fr,{ramon.nou,alberto.miranda}@bsc.es, toni@ac.upc.edu

Abstract

In this paper, we seek to guide optimization and adaptation techniques according to the applications' I/O access pattern classified at runtime. We evaluate three machine learning techniques to detect the I/O access pattern of HPC applications: decision trees, random forests, and neural networks. We seek to detection the access pattern by using file-level metrics as seen by the I/O nodes. We applied these three detection strategies in a case study in which the correct detection of the current access pattern is paramount to adjust a parameter of an I/O scheduling algorithm. We show that such approaches accurately classify the access pattern, regarding file layout and spatiality of accesses with up to 99% *precision. The access patterns selected as classes are the most common ones used by the HPC I/O community and by I/O benchmarking tools. Finally, after applying these approaches to the TWINS case study, we observe the tuning the mechanism achieve* 99% *performance for an Oracle solution.*

Aggregating Network Interfaces in Lightweight Virtualization Clouds for HPC Applications

Anderson Mattheus Maliszewski*†, Adriano Vogel‡, Dalvan Griebler†‡, Eduardo Roloff*,
Luiz Gustavo Fernandes‡, Philippe O. A. Navaux*
*Informatics Institute, Federal University of Rio Grande do Sul (UFRGS), Porto Alegre, Brazil
†Laboratory of Advanced Research on Cloud Computing (LARCC), Três de Maio Faculty (SETREM), Três de Maio, Brazil
‡School of Technology, Pontifical Catholic University of Rio Grande do Sul (PUCRS), Porto Alegre, Brazil
Email: {andersonm.maliszewski,eroloff,navaux}@inf.ufrgs.br,
{dalvan.griebler,adriano.vogel}@acad.pucrs.br, luiz.fernandes@pucrs.br

***Abstract*—In the recent past, we have witnessed continuous efforts to execute HPC applications in the cloud due to potential advantages, including higher scalability and lower costs using the pay-per-use billing. Although the industry and academy have adopted this model, there are still significant challenges to be addressed concerning network interconnection bottlenecks. One of them is that network-intensive applications may not scale in the cloud due to the sharing of the network infrastructure. Therefore, our proposal to overcome this loss of performance is the aggregation of Network Interface Cards (NICs) in a ready-to-use integration with the OpenNebula cloud orchestrator using lightweight virtualization. We conducted a set of experiments with a network microbenchmark to obtain throughput and latency network performance metrics and the NPB kernels considering execution time to analyze the performance impact on HPC applications. The results highlighted that the NIC aggregation approach improves network performance in terms of throughput and latency. Moreover, the HPC applications presented different performance patterns based on the behavior of each kernel, depending on the communication and the amount of data transmission. While network-intensive applications increased the performance up to 38%, other ones maintained the same or slightly decreased the performance.**

OPEN DATA PLATFORMS: ANALYSIS OF THE BRAZILIAN OPEN DATA PORTAL

Shirlei L. O. Carmo, Claudio F. R. Geyer, Julio C. S. dos Anjos,
Institute of Informatics - PPGC/UFGRS
Porto Alegre, Brazil
shirlei,geyer,jcsanjos@inf.ufrgs.br

Abstract

Open data is a concept attributed to the practice of sharing data with anyone, besides being accessed, this data can be manipulated and redistributed. This is a global trend that encourages the transparency of governments and entities in their transactions, as well as providing society with knowledge about relevant data in areas such as infrastructure, health, public spending and the environment. In this work, the authors presented an analysis of the Brazilian Government Open Data platform, presenting metrics about how the organization and quality of the data. The study will show data from the Brazilian platform based on Global Open Data Index (GODI) metrics.
Keywords*: OpenData. Sharing. Data.*

1. Introduction

Transparency, innovation, social and business value addition, participation, impact and engagement of society [1] are justifications for the need for openness, described in Open Knowledge Brazil, a civil organization that promotes free knowledge in various sectors of society.

According to the Open Knowledge Foundation's Open Definition project, the term open in the context of open data and open content means that data can be freely accessed, used, modified and shared by anyone, for any purpose - subject, at the most, to requirements that preserve their origin and openness [2].

Open Knowledge Brasil also defines the main conditions for the data to be considered open. In short, they are: availability and access, that is, they must be available in full and at the cost only of copying, also in a convenient and changeable format; reuse and redistribution, ie in addition to being reusable, it must be possible to combine it with another data set; and universal participation, ie.: everyone should be able to use without any discrimination against persons, groups or fields of action (such as for non-profit or educational purposes only) [3].

Thus, this paper aims to make a sample analysis of the Brazilian Government Open Data platform. Metrics such as amount of resources per dataset will be exposed, if these resources are available for download, the most used data types, if they have exposed licensing in the dataset and the type, also the update periodicity will be checked. These metrics build on the GODI methodology (which is a global reference for open data publishing) and guides institutions that want to open their data.

In the next sections, a contextualization of the use of open data will be presented, briefly exposing the difficulties in the data availability in Brazil, the types of data relevant to society, will be demonstrated how the data were analyzed and the result of the study. At the end, the work will be completed and the possibilities for future work will be verified.

2. Development

In the next subsections will be demonstrated the use and importance of having open data, and exposed the methodology and results of the study.

2.1. Importance of opening data

In addition to the benefits already mentioned, such as transparency of public spending and social engagement due to data exposure, there is also their use in developing solutions, such as *Para onde foi meu dinheiro*, which used data from OpenData-BR, the Brazilian open data portal. The site allows us to see the distribution of government investments in thematic areas such as education, health, social assistance, work, among others [4]. In the open data portal, by the Brazilian data portal application page [1] , you can ac-

1 http://dados.gov.br/aplicativos

cess solutions created for society with open data.

2.2. Difficulties encountered in providing data

According to a report produced by FGV / DAPP in partnership with Open Knowledge Brazil in 2016, the main problems in Brazilian datasets are: incomplete and outdated dataset; unavailability of open format; difficulty working the data; access restriction; difficulty locating data; full base download unavailable; and non-transparent license.

It is possible to check the full analysis of Brazilian datasets through opendatabarometer site [2] , a channel developed by the WordWideWeb Foundation that takes a global measure of how governments are publishing and using open data for accountability, innovation and social impact.

2.3. Data types

Many types of data have potential use and application, such as in the areas of culture, science, finance, statistics, climate, environment, and transport [5]. In VisPublica, Brazil's public data visualization portal, which aims to investigate and apply Information Visualization (InfoVis) techniques to facilitate transparency of public data and decision making [6]. It is possible to check the employment data registered from 2002 to 2010, as shown in Figure 1:

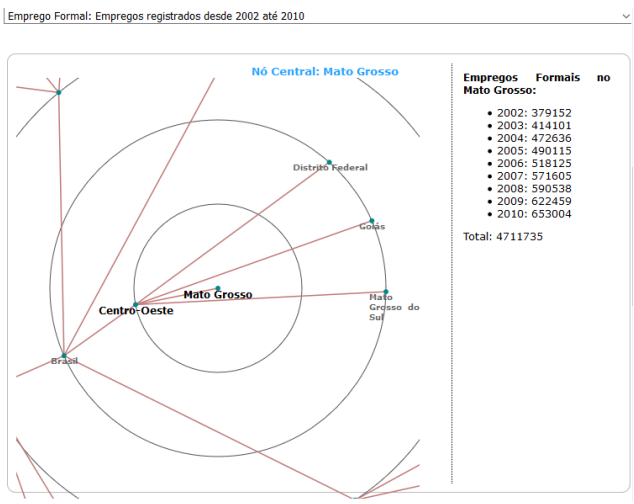


Figure 1. Formal jobs between 2002 and 2010 in the state of Mato Grosso

2 opendatabarometer.org

2.4. Methodology

For data analysis, we used the CKAN API, the largest open source data portal platform in the world. CKAN is a data management system that provides tools for publication, sharing, localization and use of data and which, in this work, was used for data exposure [7].

The technical structure used the GET-able API function and the data was extracted through CKAN's Action API with the following features:

JSON formatted lists of datasets
/api/3/action/package_list , returns available dataset set by platform;
/api/3/action/group_list, returns the groups that are contained in the datasets;
/api/3/action/tag_list, return subdivision of tags by dataset;
Full JSON representation of a dataset, resource, or object
/api/3/action/package_show?id=adur_district_spending, returns the representation of the dataset specified in param 'id'.
Search for packages or resources matching a query
/api/3/action/package_search?facet.field=[%22fieldType%22]&rows=0.

Figure 2. CKAN API features

A processing prototype with the Python programming language was developed through the Jupyter development platform. Data analysis metrics were based on the GODI methodology. This methodology helps to assess data openness in institutions around the globe, and makes an annual progress report on this openness.

According to the methodology mentioned, each data category must contain at least three characteristics. And the characteristics must contain: the required dataset content; data aggregation level; and whether the dataset is up to date.

Only these conditions being met the dataset is analyzed. In this methodology, there is also a questionnaire with 11 questions, 6 with punctuation, which together generate a dataset evaluation score.

Following are the questioner's questions: *Is the data collected by government (or a third-party related or linked to government)?* ; *Is the data available online without the need to register or request access to the data?* ***[15 points]***; *Is the data available online at all?*; *Is the data available free of charge?* ***[15 points]***; *Where did you find the data?*; *How much do you agree with the following statement: It was easy for me to find the data.*; *Is the data downloadable at once?* ***[15 points]***; *Data should be updated every [Time Interval]: Is the data up-to-date?* ***[15 points]*** ; *Is the data openly licensed/in public domain?* ***[20 points]***; *Is the data in open and machine-readable file formats?* ***[20 points]***; *How much human effort is required to use the data. (1 = little to no effort is required, 3 = extensive effort is required).*

2.5. Analysis and Results Obtained

In this work we analyzed the datasets provided by the Banco Central do Brasil (BCB) because, among the organizations contained in the portal, it provides the largest data set (up to the study date 3115 datasets) and also due to the importance of their data to society.

The results to be verified will be: amount of resources per datasets; of these how much are actually downloadable; the formats of these resources; if the dataset has an open data license; if so,what type of license is it.

One thousand BCB datasets were analyzed, sorted by relevance and update data. A JSON formatted list of the dataset has been created, available from CKAN [3] for extraction, then the data was submitted to the code in python and the returned json manipulated. The next paragraphs will display the results of each metric.

Resource is the data itself, made available in a dataset. For example, in BCB's 'Balance Sheets (IFs and Conglomerates)' dataset there are 5 resources which are, among others, the Individualized Balance Sheet in CSV format and the Individualized Financial Consolidated in HTML format.

Due to the amount of datasets and resources analyzed an average of the quantity was made. The result showed that BCB datasets only average 3 resources per dataset.

According to Opendata Commons, a site that centralizes information about licensing and providing open data, open data licensing means freedom of use and fundamental reuse of the data set.

The license type used for the 1,000 data analyzed was Open Data Commons Open Database License (ODbL): applicable to database schemas, information architecture, and data organization. Requires authoring and sharing under the same license. This means that 100% of the verified datasets use the ODbL license.

Each resource in a dataset, must have an option to download the data. The download option, is usually made by a URL, and this URL should make the download available at one time, without any login or without having to pay to download the data.

In the result obtained were analyzed 3999 resources, of which 2974 was possible to download. That is, 74% of the resources were downloadable. This means that more than 50% of resources can be downloaded. Which can result in manageable resources.

According to GODI, the formats provided for each feature must be machine readable, and readable through non-paid software. Thus formats like Json, CSV and HTML are considered acceptable as they meet both requirements.

In the datasets analyzed, the most used formats were HTML and CSV with an average of 25% of resources in this format, followed by JSON and WSDL with 20% utilization. PDF had one occurrence and XML none.

This means that CSV JSON, which is generally manageable, results in 45% of available resources. That is, on average 45% of resources could be easily manipulated.

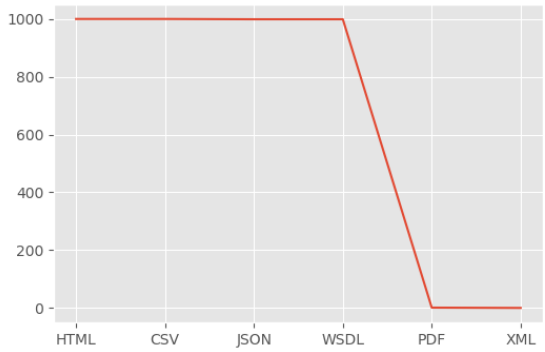


Figure 3. Most commonly used formats in datasets.

3. Conclusions

This work has shown the importance of having data open to society - when it allows transparency of public spending, for example, through resources -, and the use in developing applications - such as vispublica, mentioned in this paper -, or the like, that can generate value and engage citizens.

We evaluated a sample of datasets from a federal government organization, the Banco Central do Brasil, using metrics developed by leading open data evaluators on the globe. And it was found that all data sets analyzed have an ODbL license, over 50% of the features are downloadable and these resources formats,are mainly CSV and HTML.

In the future the data evaluation prototype may be evolved and also the evaluation metrics may be expanded to more control agencies.

References

[1] Open Definition(OD).Definicao de Aberto[online]. Available:http://opendefinition.org/od/2.1/en/. Accessed: 2019-04-20.

[2] Open Knowledge Brasil. Dados Abertos[online]. Available:https://br.okfn.org/dados-abertos/. Accessed: 2019-07-15.

[3] Open Data Barometer. Global Report[online]. Available:https://opendatabarometer.org/4thedition/report/. Accessed: 2019-07-15.

3 api/3/action/packagesearch?q=datasetid

[4] Portal Brasileiro de Dados Abertos. Aplicativos e servicos que utilizam dados abertos.[online]. Available:https://dados.gov.br/pagina/aplicativos. Accessed: 2019-03-15.

[5] OKFN. What is open?.[online]. Available:https://okfn.org/opendata/. Accessed: 2019-07-05.

[6] VISPUBLICA. Sobre o VisPublica. Braslia[online]. Available:https://vispublica.gov.br/vispublica/publico/contato.jsp. Accessed: 2019-05-08.

[7] CKAN. Sobre[online]. Available:https://ckan.org/about/. Accessed: 2019-07-10.

IoT Cloud Framework

Desiree dos Santos
Institute of Informatics - PPGC/UFGRS
Porto Alegre, Brazil
desiree.santos@inf.ufrgs.br

Abstract

Internet of things is more than intelligent objects. It is new area which converts analog into digital world for better decisions makers and improving human being life. For this dream become reality, infrastructure behind is needed to support high demand of connected sensor network, to process and analyse data and them give back an answer with accuracy and precision for user in an acceptable time. Cloud computing address in a good shape Internet of Things needs. It reflects enthusiastic scenario of opportunities for new brands and revolutionary the industry.

The main proposal of this research is to evaluate the state of the art of Internet of Things cloud framework in order to support a better understanding which characteristics are important to build better IoT solutions on cloud. This work focused on three most popular IoT cloud providers(AWS, Azure and Google) as well re-evaluate the cloud services characteristics presented by Mushed [13] highlighting the new features and present benchmark results about performance for Google Core Internet of Things.

Keywords: IoT, cloud, internet of things, cloud computing

1. Introduction

Internet of things is one of the driving forces which generates data all the time from household appliances, cars, and other physical devices without requiring human-to-human or human-to-computer interactions.

The computer scientist Mark David Weiser created in 1988 ubiquitous computing term and vision, but in 1999 Market Ashton introduced Internet of Things(IoT) paradigm, which converted the ubiquitous computing concept in applicable by the embedded short range of sensors in the network-enabled objects or devices with Radio Frequency Identification(RFID) to integrate communication between people and devices [4]. IoT technology transforms all raw material in a pipeline of capturing, processing and transform to make a decision with high precision to facilitate human in life simpler.

Based on Help net security portal [3] 41.6 billion IoT devices will be generating 79.4 zettabytes of data in 2025 from devices on the ground. The existence of marketing and assistance of IoT is inevitable. This scenario demands a particular infrastructure to play this critical role of making efficient and reliable supporting high demand to process data, in closer or real-time efficient communication to fit user and system feedbacks.

In the same direction, cloud technology provides parallel and distributed computing infrastructure environment which can be offered in different levels of services like platform as service(PaaS), Software as a Service(Saas), infrastructure as a service(Iaas) [15] enabling the flexible and scalable environment with low financial cost to supply for different channels like web and mobile applications.

The development of IoT demands high efficient storage, high-speed and computing power, which cloud computing is an advantage and combines IoT needs [9], for collects, organizes data and information by using wireless sensor identification, then transmits it to the application layer if the cloud computing platform [16].

According to Mukesh among the current available Cloud IoT solutions in the market, there are three Cloud with the most notable services for IoT Computing [7, 13], based on market share at 2018 (Figure1).

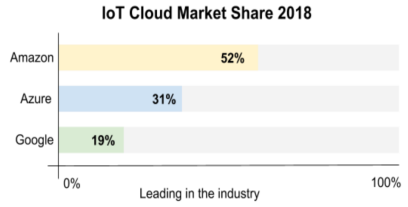


Figure 1. Cloud IoT market share in 2018.

This market share list address the following cloud providers: Amazon Web Services IoT, Azure IoT Hub e Google Cloud Core IoT.

Guth et al. [5] present open-source platforms and cloud for IoT like FIWARE, OpenMTC, SiteWhere, Webinos and proprietary solutions IBM's Watson IoT Platform, Samsung's SmartThings and Salesforce.

The structure of this article contemples 4 sections. The next presents methods and procedures to presents the process to collect the data. Section 3 represents the related word represents the research finding and section 4 conclusion.

2. Methods and Procedures

The methodology apply a protocol based on systematic mapping of Peterson et al.(2015). It consist three steps: planning, conducting and present the results.

In order to support the planning step becomes important to defined research question(RQ) to support better conducting and final results, the following primary research questions (RQ1 and RQ2) and the secondary ones (a, b, c) for each RQ were defined:

RQ1) Questions to answer concerning IoT cloud characteristics between the three most popular cloud IoT :

a) Which cloud services?

b) Supported protocols?

c) Process, analytic?

RQ2) Questions to answer about benchmark concerning specific for Google Cloud Core IoT:

a) What are the important items to test in the benchmark for IoT?

b) Performance

To discover the state of cloud IoT framework were elaborated a search string (Figure 2) to apply against a database with filters on IEEE explorer[1] and google scholar[2] database.

IEEE	
Articles	1.270
Search String	(cloud OR cloud computing) AND (internet of things OR iot) AND framework
Link	https://bit.ly/2luXzdn
Google Scholar	
Articles	651
Search String	”cloud” AND ”internet of things” AND ”framework”
Link	https://bit.ly/2MOJQdw

1 http://IEEexplore.ieee.org
2 https://scholar.google.com.br

2.1. Data Gathering and criteria

1. Research on search engine IEEE and Google scholar
2. Filter IEEE explore articles manually applying exclude rules provided by own engines, as cited below:
   (a) Discarded articles before 2017
   (b) Discarded articles out of scope for cloud IoT topic
3. Filter Google scholar articles manually applying exclude rule
   (a) Discarded articles before 2017
   (b) Discarded articles out of scope for cloud IoT
4. Import BibTeX from step 3 into StArt tool program1
5. Filter articles manually multiple from step 4 and 5
6. Result in 30 relevant articles for this research

Inclusion and Exclusion criteria: The Inclusion and exclusion criteria is to ensure only relevant works aligned objective of this article will be selected and in parallel to facilitate the filtering process. The follow inclusion criteria(IC) enabled the section of appropriated works:

- IC1: The result must be in the defined languages;
- IC2: The result must be available in full and be available free of charge;
- IC3: The result must contain in the title, keywords, or in the abstract some relation with the theme of this work; The result will be from the area of engineering or computer science;
- IC4: Works that address cloud issues for IoT in a general way.

The follow exclusion criteria(EC) enabled the section of appropriated works:

- EC1: The result is not related to the work theme
- EC2: Articles with languages other than English and Portuguese In the case of similar or duplicate studies, only the most recent was considered.
- EC3: Works that address cloud issues for IoT in a general way.

3. Related Work

The major cloud provides specialized services for data collection from IoT devices, support popular protocols like Message Queue Telemetry Transport(MQTT), Advanced Message Queuing Protocol (AMQP), CUPUS pub/sub, Extensible Messaging and Presence Protocol (XMPP), Representational State Transfer (REST) over Hypertext Transfer Protocol (HTTP). However applications for industrial area

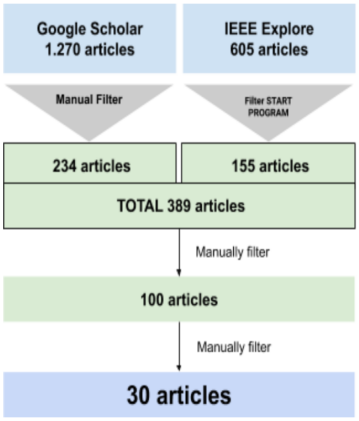


Figure 2. Step by step to filter articles

works around close to real-time, cases like industrial machines are controlled by applications running in the cloud or use the cloud as a central instance for data exchange. By another hand, Daniel Happ et al. [8] describes the limitations of publishing and subscribe architecture for constrained devices, high latency, low-bandwidth, unreliable links used in protocols MQTT, AMQP. The first drawback is a side-effect of the decoupling properties and space. In pub/sub systems, the subscriber does not know if there are any publishers publishing on a given topic, how many of them there might be or who the publisher of a given message might be. Likewise, the publisher does not know if there are any subscribers actually interested in messages on a certain topic.

The platforms come in various shapes and sizes and there are not standard for IoT [17], although the effort to become homogeneous exists [17]. At some level, most of all platform landscape allows: to connecting different devices, accessing and processing their data and using the knowledge gained through this ecosystem to create automated control [5]. The particularity of the platforms are competitive because they are differential between them, however, it places a heavy burden on the user who will choose which tool to start their application, demanding as a result of well-defined and clear documentation to get the answer immediately. The most platforms and architecture found about cloud IoT does not provide an easy way to compare in high granularity, but Gut el at. [10] extends this work by a refinement of the description, the extension of the analysis to eight IoT platforms, and a more extensive survey on related work.

Quality of Service (QoS) in the Internet of Things is an important topic because there are multiple IoT applications type and behaviors and in the same range there are different expectations of quality of services, describing them in a more comprehensive manner. Manish et al. [6] mention in three pillars: QoS param for Things, communication network in IoT ecosystem responsible for transporting in real-time data and computing. The first pillar things there is item mentioned security, which Google Cloud IoT core is addressed. The strong advantage of secure credentials, like proper device registration, identity authorizations. According to Amir Google Core IoT provides reliable and secure ingestion, connection and management of data from an incredible diversity of resource [1, 2].

Metrics provides parameters to track and measuring fundamental results. In this way, IBM, Microsoft, Oracle examining the performance of their products for storing, repairing and analyzing [11, 12, 14]. Benchmarking is another related domain with industry-standard created by groups like SPEC [3] and TPC [4]. We could find a benchmark for social graph, cloud services, but there is not a consolidated guide for benchmarking applications for the Internet of things. On the other hand, Martin et al [10] introduced IoTAbench to address this gap of standard for evaluating a big data analytic platforms for IoT.

4. Conclusions

internet of things is gaining increasing attention, turning from vision into reality and showing booming trend by transforming analog data into digital. In the same direction IoT Cloud platform offer scalable, flexible and low cost infrastructure to simplify the ”Things” pipeline of capture raw data from sensor, send to cloud to process and analysis for better decision-maker. In order to understand the current scenario of cloud IoT framework the some initial question was made characteristics and benchmark

RQ1) Questions to answer concerning IoT cloud characteristics between the three most popular cloud IoT:

a) Which cloud services?

b) Supported protocols?

c) Process, analytic?

The result of this research presented clearly 3 AWS, Azure and Google as the most popular proprietary cloud based on market share, confirmed Mushed [13] research published in 2018. The leader in market keeps Amazon Service Web and Forbes predicts by 2020 this scenario is still the same. The table [5] which compare cloud services, protocols, pricing, analytics Google Core IoT enhanced (SDK, support protocols, services and security/Authentication).

RQ2) Questions to answer about benchmark concerning specific for Google Cloud Core IoT:

3 https://www.spec.org
4 http://www.tpc.org
5 https://bit.ly/2lOqVTY

a) What are the important items to test in the benchmark for IoT?

b) Performance

Although many studies on cloud benchmarking in general exist, we could find a research which address clearly standard of metrics for IoT Cloud Framework, but Manush et al. [16] studies in Quality of Service(QoS) area divided the metrics for IoT in three categories:

In this research Google Core IoT is referenced as stronger in IoT Cloud Platform for metrics security in category Things, because of the full-stack security for providing reliable and secure ingestion, connection and management of data from in a heterogeneous range of devices.

In the same direction, IoTABench is toolset fix the gap in terms of computing and communication categories in QoS, but it's not fully operation on production. On the other hand, there are several benchmark framework not specialized for Iot, but valid to test Cloud Platform in general, like PerfKitBenchmarker, PerfKitExplorer, Geekbench, UnixBench, Sysbench(cpu), Stream, SPEC CPU2006 Integer, Stream, MAIH, AWSI.

Benchmark: Test performance was executed using the framework PerfKitBenchmarker on Google Core IoT the result is present in figure . Cluster created in region europe zone west1. Simulated calls on GCP Engine from PerfKitBenchmarker and the second benchmark was device sending data connected in service Core pub/sub.

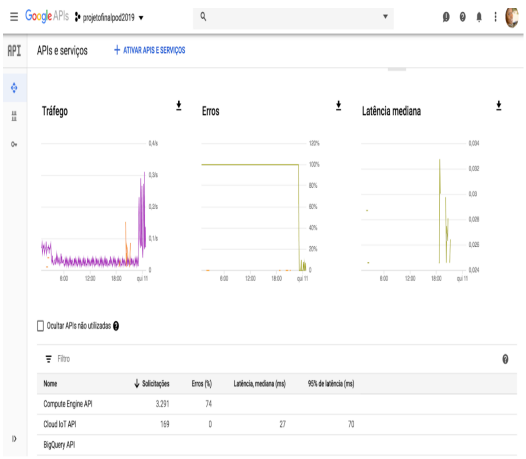


Figure 3. Google Cloud Core IoT

References

[1] Cloud IoT Core solutions[online]. Available:https://cloud.google.com/solutions/iot/. Accessed: 2019-08-20.

[2] Cloud IoT Core[online]. Available:https://cloud.google.com/iot-core. Accessed: 2019-08-20.

[3] Connected-iot-devices-forecast[online]. https://www.helpnetsecurity.com/2019/06/21/connected-iot-devices-forecast/. Accessed: 2019-08-20.

[4] J. Townsend, The Function of Quality Assurance (QA) with the Internet of Things[online]. https://www.ctl.io/blog/post/qa-with-theiot. Accessed: 2019-08-20.

[5] J. G. authorUwe BreitenbcherMichael FalkenthalPaul FremantleOliver KoppFrank LeymannLukas Reinfurt. A detailed analysis of iot platform architectures: Concepts, similarities, and differences.

[6] J. G. authorUwe BreitenbcherMichael FalkenthalPaul FremantleOliver KoppFrank LeymannLukas Reinfurt. Limitations of the pub/sub pattern for cloud based iot and their implications.

[7] P. P. Ayyappadas, Kavitha. Design and implementation of weather monitoring system using wireless communication. *International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering*, 5(5):1– 7, January 2017.

[8] Borgia. The internet of things vision: Key features, applications and open issues. *Computer Communications*, page 131.

[9] P. P. D. T. De Leusse P. Self managed security cell, a security model for the internet of things and services. *IEEE*, pages 47–52.

[10] F. L. R. Guth, Breitenbucher. Comparison of iot platform architectures: A field study based on a reference architecture. in proceedings of the international conference on cloudification of the internet of things.

[11] F. L. R. Guth, Breitenbucher. Comparison of iot platform architectures: A field study based on a reference architecture. in proceedings of the international conference on cloudification of the internet of things.

[12] Itron and Microsoft. Benchmark testing results: Unparalleled scalability of itron enterprise edition on sql server. Technical report, Microsoft, 2018.

[13] G. Shewale. An iot based real-time weather monitoring system using raspberry pi. *International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering*, 6(6):4242–4249, January 2017.

[14] M. to-cash performance using Oracle Utilities applications on Oracle Exadata and O. Exalogic. Benchmark testing results: Unparalleled scalability of itron enterprise edition on sql server. Technical report, Oracle, 2018.

[15] O. T. Yorozu, Hirano. Electron spectroscopy studies on magneto-optical media and plastic substrate interface. *IEEE Transl. J. Magn. Japan*, 2(5):740–741, January.

[16] Y. Q. Zhu, Yang. Investigation of technical thought and application strategy for the internet of things. *Journal of China Institute of Communications*, pages 2–9, 2010.

[17] L. B. Zorzi, Gluhak. From todays intranet of things to a future internet of things: A wireless- and mobility-related view. page 4451.

Efficiency of LBM Applications on Low-Power Heterogeneous Architectures

Gabriel Freytag*, Matheus Serpa*, João V. F. Lima†, Paolo Rech*, Philippe O. A. Navaux*
* Federal University of Rio Grande do Sul (UFRGS), Porto Alegre, Brazil
† Federal University of Santa Maria (UFSM), Santa Maria, Brazil

***Abstract*—As the computing power of High-Performance Computing systems increases, and as a result, its energy consumption increases as well, new architectures arise with the purpose of minimizing energy consumption. An example of this is the hybrid architectures that integrate devices from different architectures into a single chip (System on Chip - SoC). In this work we evaluate the performance and power consumption of the Lattice Boltzmann method (LBM) in two hybrid architectures: one composed of CPU and GPU processing units integrated into a single chip (AMD Kaveri SoC), and another by CPU units and an FPGA integrated in a single chip (Intel Arria 10 SoC). The performance and power consumption of the method were evaluated using each device individually and also in a collaborative way through the decomposition of its data domain. We also evaluated the execution time of each of its five subroutines in both the architectures and their devices. The experimental results show that the collaborative execution of the method reduces its execution time. However, the execution times of each subroutine show that non-uniform partitioning for the subroutines could further optimize method execution time. Finally, by evaluating the energy consumption, it is possible to observe that, due to the high execution time of the non-optimized method in Arria 10 SoC in comparison to Kaveri SoC, the consumption is equivalent or even higher in Arria 10 SoC.**

Prefetcher's Impact Over Parallel Architecture Simulation Accuracy

Valéria Soldera Girelli[1], Francis B. Moreira[1], Matheus S. Serpa[1] and Philippe O. A. Navaux[1]
[1]Informactics Institute – Federal University of Rio Grande do Sul
Postal Code: 15.064 – 91.501-970, Porto Alegre – RS – Brasil
{vsgirelli, fbmoreira, msserpa, navaux}@inf.ufrgs.br

Abstract

In computer architecture research, the use of simulators is predominant. There are diverse approaches and implementations of modern simulators. However, validation studies lack a detailed analysis of parallel behavior in high-performance architecture simulators. We could observe that, due to the lack of prefetching simulation, the memory hierarchy statistics are inaccurate. Thereby, this study analyzes the impact of the prefetcher in the parallel simulation methodology used by ZSim.

1. Introduction

In computer architecture research, physical implementation and analysis are infeasible due to the complexity and high cost for manufacture. Consequently, architecture simulators are considered the primary mechanism to implement and evaluate a new idea in this research field [1]. In High-Performance Computing (HPC) systems, there are many other problems besides those inherent in the architecture. To develop and analyze new ideas that attenuate problems arising from parallelism, we need multicore architecture simulators that support parallel workloads. For example, systems with dozens of cores point out problems as the interference among the different threads and the communication costs among them. Thread interactions occur mainly through shared memory, with several threads accessing the same memory addresses, making necessary to keep data coherence in the several cache levels. Thereby, parallel simulators must provide an accurate implementation of cache coherence protocols and ensure data consistency through the memory hierarchy.

Another example is the prefetcher, a technique that identifies access patterns from each core and creates speculative memory requests. Thereby, the behavior of the processor's memory hierarchy receives a new level of complexity since simulations must consider information previously discarded, like cache line content [2], register values [3] or physical addresses [4].

In this paper, we analyze ZSim, a parallel architecture simulator that implements a parallel simulation methodology [5]. ZSim uses the Pin [6] tool to instrument the application, using dynamic binary translation to simulate the application's behavior. Likewise, the simulation of the parallel architecture and parallel workloads also is performed in parallel, decreasing the simulation time. By analyzing the implementation approach that ZSim applies for the parallel simulation, we observed that the method precludes the correct simulation of prefetchers in the coherence protocol.

The remainder of this paper is as follows. In Section 2, we detail the motivations of our study and the implementation approach of the simulator. Section 3 presents the configuration of the experiments and simulations. In Section 4, we present and discuss the obtained results, and in Section 5, we present our conclusions and future work.

2. Motivation and Related Work

Hardware companies use intellectual property law and hide information to avoid competition. Thus, it is difficult to obtain an ideal simulator which correctly implements all the processor's components and its architecture. In [7], the authors presented the difficulty of obtaining accurate information and its relevance when validating the SimpleScalar simulator [8]. By obtaining more information about the Alpha 21264 processor model, the authors were able to reduce the average experimental error with the SPEC-CPU 2000 workload from 36.7% to 18.2%. The authors demonstrated how the found features generate bottlenecks in different parts of the system than previously modeled in various studies. With this result, they could invalidate ideas from other articles that demonstrated performance gains where there was no bottleneck.

Among the several available simulators, we selected ZSim for an in-depth study due to its speed and accuracy, shown in its validation [9] and in Akram's work [10]. In Akram study, the simulators gem5 [11], Multi2Sim [12], MARSSx86 [13], PTLsim [14], Sniper [15], and ZSim [5] are evaluated. After thorough characterization of the simulators, the authors analyzed single and multicore workloads on each simulator. The target architecture was the In-

tel Haswell architecture [16]. The authors then highlighted the simulators' error sources, their sensitivity to different architectural parameters, and their relative error. Thus, they concluded that the lack of validation of simulators leads to poor accuracy and may result in invalid experiments due to erroneous conclusions. Akram's work does not use multi-thread workloads to verify the existence of cumulative relative error when increasing the number of threads.

2.1. Prefetcher

The memory available inside a processor has different levels. Private levels closer to the processor have less storage capacity, but accessing and transferring data from these levels is much more efficient. If a copy of the requested data is not in the L1 data cache, the first level of the processor's internal memory hierarchy, it forwards the request to the next level of cache memory, which repeats the same procedure. The last level cache (LLC), in turn, forwards the request to main memory. Thus, the cache levels closest to the processor should have data as relevant as possible at any given time in the execution of a program.

Prefetching is a technique to reduce data access latency. Based on the pattern of accesses generated by the processor, the prefetcher speculates on which is the next address to be requested and requests the data block in advance. Thereby, when the data block is indeed requested, it will already be in caches closer to the processor. Some of the patterns the prefetcher identifies are the strided [4] and the stream [17].

Figure 1 shows an example of the prefetcher identifying a stridden access pattern. The L2 cache level forwards requests to the LLC (indicated in Figure 1 as event **1**). The prefetcher, in turn, intercepts these requests (**2**) and identifies their access pattern. Based on the identified pattern, speculative accesses to the LLC are performed (**3**). These accesses are typical requests made by L2, so LLC forwards these data directly to L2. Thereby, when data previously requested by prefetch is needed, it is already at cache levels closer to the processor. Prefetchers are prevalent in today's architectures as one way to mitigate the main memory access bottleneck [18].

2.2. ZSim

ZSim implements the simulation in a two-phase method called Bound and Weave. In the Bound phase, a few thousand cycles are simulated, ignoring contention and using minimum latencies for all memory accesses. The simulator records the trace of memory accesses, including their passage through caches, invalidations, and evictions, among other details. In the Weave phase, the actual access latencies are determined based on the events of the previous phase. As the simulator identified the interactions between memory accesses in the first phase, it can efficiently simulate access timing while maintaining high accuracy.

The authors note that within a few thousand cycles, most concurrent access between different cores happens to separate cache lines. Thereby, regardless of the order of treatment of accesses, nothing changes in cache system consistency, and no request for ownership, invalidation, or update requests are required. However, when accesses are related, i.e., accessing the same cache line for communication, there is the need to maintain the coherence of different copies of data across different cores. For being considered rare, the **order** of these interactions is not modeled by ZSim, which may change the path of this data through the cache hierarchy, possibly impact the number of cycles, misses, coherence messages and even the path of prefetch requests. Therefore, ZSim does not simulate prefetchers in this model.

In its validation, ZSim uses the PARSEC benchmark [19] and only demonstrates that the speed-up is close to the obtained in real executions when varying the number of threads. In this study, we aim to evaluate the accuracy of the Bound and Weave simulation approach applied in the parallel simulation of several resources and threads.

3. Experimental Environment

In this section are specified the configurations of the environments used in the experiments.

3.1. Processor's Configuration

The *perf* [20] tool was used to get hardware counter values for all real executions. We evaluated performance and statistics of the real execution using 10 executions of each statistic, for each benchmark, on identical *draco* machines. These machines belong to the Parque Computacional de Alto Desempenho (PCAD)[1], which belongs to the GPPD - Parallel and Distributed Processing Group. For the simulations, we used configurations that approach the real machine, an Intel(R) Xeon(R) CPU E5-2640 v2, Ivy Bridge architecture [21], respecting the limitations of the simulator. Each simulation executed only once, and all statistics come from this same simulation.

In this article, we display a selection of statistics, which illustrates problems in ZSim memory hierarchy modeling. We chose statistics for execution cycles, L1 data cache accesses, and LLC accesses. Cycle statistics illustrate the highest number of cycles taken by the parallel threads, representing the time it took the application to execute. L1 cache access statistics represent the average among the caches of all threads, where we can see the data structures being split between the threads as we increase the application parallelism. The statistics of the LLC count accesses to all banks, since any application will use all banks. The standard deviation of the ten real executions is not shown in the images because it is smaller than 1%.

1 `http://gppd-hpc.inf.ufrgs.br/`

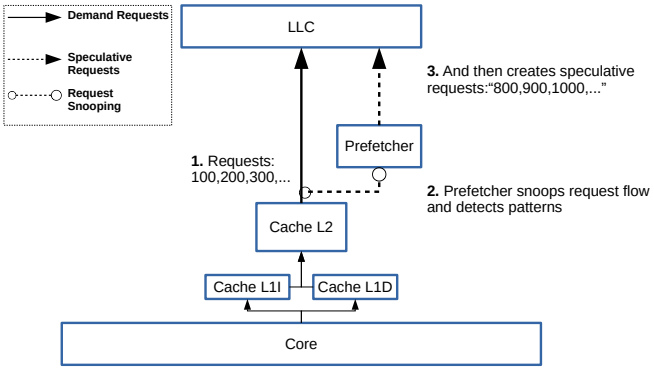


Figure 1: A simple prefetcher abstraction.

3.2. Benchmark's Configuration

We used the Numerical Aerodynamic Simulation Parallel (NPB) [22] benchmark to evaluate the scalability of the error regarding the number of threads, was used. The NPB benchmark is made up of ten parallel applications, of which we used nine. The applications used were BT, CG, EP, FT, IS, LU, MG, SP, and UA. The **DC** ("Data Cube") application was not used due to the large use of I / O, which is not modeled by the simulators.

4. Memory Hierarchy Behavior Analysis

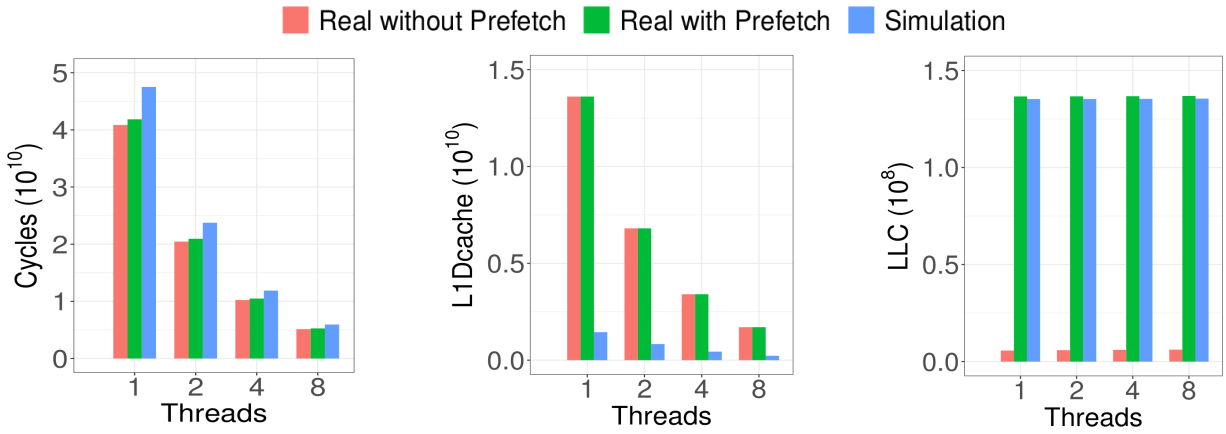
Since ZSim does not simulate the prefetching system, there could be changes in the number of cycles, misses, and LLC accesses. We analyzed the number of cycles and accesses to the data cache observed in the simulation and real execution with and without the prefetcher interference. We found that, for all benchmarks, the number of simulated cycles is always higher than in the real executions. Furthermore, all the simulations follow the speed-up trends observed in the real executions. Concerning L1 accesses, the number observed in the simulation is always smaller and also follows the trends of the real executions. LLC access statistics, on the other hand, show erratic behavior as a result of inaccuracy in cache hierarchy modeling.

In Figure 2, we can see from the EP application the lack of prefetch in ZSim. The simulator follows the real execution trends faithfully, but the percentage of accesses to the LLC avoided by the machine prefetcher is explicit in this benchmark, as shown in Figure 2c. This value represents a difference higher than 65000% in the number of accesses that reach the LLC.

The IS, LU, UA and MG applications present similar behaviors. Although the absolute number of accesses to the LLC is smaller in this applications if compared to the simulated in the EP application, all executions tend to increase in this metric as the number of threads increases. In the MG application, requests to the LLC of the real execution with prefetch increase so much that they reach the level of execution without prefetch. That may be due to inaccurate prefetchers or increase in communication. We could observe such an increase in almost all applications. As the number of threads increases, it becomes clear the decrease of the difference between the execution times of the simulation and the real execution without prefetch when compared to the execution with prefetch. Therefore, parallelism makes communication a bottleneck by increasing the number of requests to the LLC, which makes prefetchers unusable and limits the behavior in all cases analyzed. The only application where the observed behavior does not occur is EP, where memory and communication requirements are so small that its memory footprint fits in the LLC.

Some erratic behaviors were observed in some applications. The BT application presents similar behavior to that noted so far for first-level cache accesses and cycles. However, the ZSim simulation contains erratic behavior for one and two threads, where the number of hits to the LLC is much higher than the real execution without prefetching. In the CG application, the single-threaded real execution without prefetcher has a very high number of hits to the LLC. The observed standard deviation is less than 1%, which does not indicate a possible variance in the experiments as an explanation for such behavior.

Table 1 illustrates a summary of the differences in accesses to the last level cache between simulation and real execution with prefetch. We can observe the impact of prefetcher on memory access in the selected benchmarks. Simulations of cache memory have their effects altered by the lack of prefetcher, which could invalidate results from several articles. That would reduce the effect of cache replacement policy papers because prefetch already mitigates latency of main memory accesses [23]. On the other hand, the communication effect and the prefetch action itself increase contention in the LLC banks. Therefore, as the number of threads increases, the difference between the real execution time with prefetch decreases when compared to the real execution without prefetch and the simulation.



(a) Number of Cycles (b) Number of L1 Data Cache Accesses (c) Number of LLC Accesses

Figure 2: Comparison of real execution with prefetch, real execution without prefetch, and simulation of the EP application.

Threads:	**1**	**2**	**4**	**8**
CG	7%	8%	1%	3%
EP	2.600%	2.600%	2.150%	2.150%
FT	132%	112%	97%	79%
IS	130%	98%	64%	44%
MG	114%	99%	57%	00%
UA	86%	83%	79%	72%
BT	387%	212%	49%	34%
LU	471%	333%	305%	310%
SP	184%	164%	118%	83%

Table 1: Error resulting of the absence of prefetcher in ZSim.

5. Conclusions and Future Work

It is possible to observe some inconsistencies resulting from the approach taken by ZSim to model the accesses implemented by ZSim, as in BT and CG applications. The results show that the lack of prefetchers in the ZSim simulation can lead to significant differences in application behavior. ZSim's lack of prefetcher is a direct consequence of the Bound and Weave model, which makes it challenging to insert speculative accesses into the cache hierarchy. Future work will be investigating a possible implementation of prefetch in ZSim, given its popularity and fast simulation, which result in great practicality for conducting experiments.

Besides, the number of data cache accesses observed in the simulation is always lower than the real execution. Thus, it is also necessary to investigate the relationship of the simulator with memory requests. Specifically, it is necessary to investigate simulator instruction decoding, which requires updates to support new instructions, e.g., AVX (Advanced Vector Extensions) [24].

References

[1] K. Skadron, M. Martonosi, D. I. August, M. D. Hill, D. J. Lilja, and V. S. Pai, "Challenges in computer architecture evaluation," *Computer*, vol. 36, no. 8, pp. 30–36, 2003.

[2] R. Cooksey, S. Jourdan, and D. Grunwald, "A stateless, content-directed data prefetching mechanism," in *ACM SIGPLAN Notices*, vol. 37, pp. 279–290, ACM, 2002.

[3] K. J. Nesbit and J. E. Smith, "Data cache prefetching using a global history buffer," in *10th International Symposium on High Performance Computer Architecture (HPCA'04)*, pp. 96–96, IEEE, 2004.

[4] T.-F. Chen and J.-L. Baer, "Effective hardware-based data prefetching for high-performance processors," *IEEE transactions on computers*, vol. 44, no. 5, pp. 609–623, 1995.

[5] D. Sanchez and C. Kozyrakis, "Zsim: Fast and accurate microarchitectural simulation of thousand-core systems," in *ACM SIGARCH Computer architecture news*, vol. 41, pp. 475–486, ACM, 2013.

[6] C.-K. Luk, R. Cohn, R. Muth, H. Patil, A. Klauser, G. Lowney, S. Wallace, V. J. Reddi, and K. Hazelwood, "Pin: Building customized program analysis tools with dynamic instrumentation," in *Proceedings of the 2005 ACM SIGPLAN Conference on Programming Language Design and Implementation*, PLDI '05, (New York, NY, USA), pp. 190–200, ACM, 2005.

[7] R. Desikan, D. Burger, and S. W. Keckler, "Measuring experimental error in microprocessor simulation," in *Proceedings of the 28th annual international symposium on Computer architecture*, pp. 266–277, ACM, 2001.

[8] T. Austin, E. Larson, and D. Ernst, "Simplescalar: An infrastructure for computer system modeling," *Computer*, no. 2, pp. 59–67, 2002.

[9] D. Sanchez, "Zsim tutorial validation." http://zsim.csail.mit.edu/tutorial/slides/validation.pdf, 2016.

GPPD–PCAD
HPC Resources Management Infraestruture Description and 10-month Statistics

Lucas Leandro Nesi, Matheus S. Serpa, Lucas Mello Schnorr, Philippe Olivier Alexandre Navaux
Institute of Informatics, Federal University of Rio Grande do Sul - UFRGS, Porto Alegre, Brazil

***Abstract*—Managing HPC resources for sharing its computational power among different users is a complex task and requires different software to assist its control. This paper describes the GPPD–PCAD cluster located at the Informatics Institute of the Federal University of Rio Grande do Sul. We present hardware infrastructure of the computational resources, including the interconnection among them. Moreover, we describe most of the software used to manage, control, and report the utilization of the resources, including the specific configurations used, and the versions used. We also present utilization statistics of the cluster using data provided by the resource manager. This information includes the resources utilization over time, most daytime use of the resources and the distribution of jobs sizes. We conclude our paper with some ideas to surpass the limitations we have identified for the current software infrastructure.**

I. Introduction

High-performance computing (HPC) has been responsible for a scientific revolution. The evolution of computer architectures improved the computational power, increasing the number of problems and quality of solutions solved in the required time, e.g., weather forecast. Furthermore, the industry has shifted its focus to parallel and heterogeneous architectures, which nowadays are part of all supercomputers.

The ability to share resources of a supercomputer or a cluster among many different users is a complex task that requires to establish some goals. Questions that may appear on the configuration of such environment are: should the users be able to choose the operating system of the machines? Should the users be able to install and control the installed software? How to effectively and fairly share the resources among the involved people? How to provide an efficient and global authentication system? There are different software with alternative goals to control such systems. Some influential examples are OAR [1], of the French Grid5000 [2], and Slurm [3], present in some Top500 supercomputers [4].

We present GPPD–PCAD[1], which stands for *Parque Computacional de Alto Desempenho* of the Grupo de Processamento Paralelo e Distribuído (GPPD) [2] of INF-UFRGS to share the knowledge gathered voluntarily during its configuration and to familiarize future users about the cluster structure. The main contributions of this paper are as follows. (a) We detail the resources infrastructure, including each node description and the intercommunication. (b) We present the software stack used in the system to manage, control, and supervise all the resources and users. (c) We also present some utilization statistics since the adoption of the current software solution and accountability.

Section II presents the hardware and software infrastructure. Section III details the 10-month utilization statistics metrics. Finally, Section IV concludes the paper with general perspectives about resource management with some future ideas for the software infrastructure of the cluster.

[1]The GPPD–PCAD website: http://gppd-hpc.inf.ufrgs.br/
[2]The GPPD website: http://www.inf.ufrgs.br/gppd/site/

II. Infrastructure

A. Hardware

The resources available on GPPD–PCAD are presented on Table I, with the description of its CPU, Total memory RAM, the storage device, and the accelerator. The machines were gathered across different hardware generations and via a different number of projects and partnerships. To summarize the resources, the machines have Intel CPUs, and an NVIDIA GPU as an accelerator if present.

Multiple switches are used for two purposes. 1 – Internet connection. 2 – Internal services and experiments. One that connects all the resources to the building internet back-end, proving the DHCP service and connection across the campus machines and external users. Another switch, with 1Gb bandwidth, is used to isolate the communication of the machines. All the essential services use this secondary channel, including Slurm, NFS, and LDAP. When executing jobs, users can select one of the network interfaces to be used, and it is strongly suggested to use the dedicated switch to transfer and communicate computational data among the nodes.

B. Software

This Section describes the software stack used though the environment. It is essential to notice that some software choices and configurations are arbitrary and exploratory, without an extensive and exhaustive examination over it.

The operating system used across the cluster is Ubuntu [5], more specifically, Ubuntu server 18.04.02 LTS. Ubuntu is an open-source Operating System developed by Canonical Ltd that used the Linux kernel and is based on the Debian system. The choice regarding its utilization comes from its widespread adoption, where many users are familiar with the system.

Slurm [3] (Simple Linux Utility for Resource Management) is an Open-source job scheduler for Linux-based systems

TABLE I
COMPUTATIONAL RESOURCES AVAILABLE ON GPPD–PCAD

Machine	CPU	RAM	Storage	Accelerator
gppd-hpc	2 x Intel Xeon E5-2630	16 GB DDR3	11 TB	
knl[1-4]	Intel Xeon Phi 7250	96 GB DDR4	440 GB	
blaise	2 x Intel Xeon E5-2699 v4	256 GB DDR4	1.6 TB	4 x NVIDIA Tesla P100
tupi1	Intel Xeon E5-2620 v4	64 GB DDR4	440 GB	
tupi2	Intel Xeon E5-2620 v4	80 GB DDR4	3.6 TB	2 x NVIDIA GeForce GTX 1080Ti
hype[1-3]	2 x Intel Xeon E5-2650 v3	128 GB DDR4	550 GB	
hype[4-5]	2 x Intel Xeon E5-2650 v3	128 GB DDR4	550 GB	2 x NVIDIA Tesla K80
orion1	2 x Intel Xeon E5-2640 v2	48 GB DDR3	916 GB	1 x NVIDIA Tesla K20m
orion2	2 x Intel Xeon E5-2640 v2	32 GB DDR3	916 GB	2 x NVIDIA Tesla K20m
draco[1-6]	2 x Intel Xeon E5-2630	64 GB DDR3	1.8 TB	1 x NVIDIA Tesla K20m
draco7	2 x Intel Xeon E5-2630	128 GB DDR3	1.8 TB	2 x NVIDIA Tesla K20m
bali[1-2]	2 x Intel Xeon E5-2650	32 GB DDR3	916 GB	
beagle	2 x Intel Xeon E5-2650	32 GB DDR3	916 GB	
turing	4 x Intel Xeon X7550	128 GB DDR3	3.6 TB	

created by LLNL (Lawrence Livermore National Laboratory). Currently, Slurm is deployed at many supercomputers around the world including the Brazilian super computer SDumont[3]. Slurm permits users to allocate and deploy batch jobs to a series of resources guaranteeing that specific resources, like CPUs, memory, or entire nodes, are correctly shared among users and jobs without erroneous interaction. Slurm is adopted on PCAD with version 18.08.5-2. Primarily, each set of equal resources (Draco machines, for example) are aggregated on a partition of resources that have an individual queue for submissions of jobs. Users can send jobs and allocate entire nodes of different partitions. PCAD also have a `shared` queue for allocation of CPUs instead of entire nodes. Multiple jobs can then be executed at the same time on the same node.

LDAP [6] (Lightweight Directory Access Protocol) is an Open-source application protocol for accessing and maintaining distributed directory information services over an IP network. A common use of LDAP is to provide a central place to store usernames and passwords. This allows many different applications and services to connect to the LDAP server to validate users. LDAP is used on PCAD as the central authentication system. The users are not individually registered on each resource, but, when authentication is required, the node will communicate to the LDAP server on the PCAD front-end node to validate the authentications credentials.

PAM [7] (Pluggable authentication modules) is a mechanism to integrate multiple low-level authentication schemes into a high-level application programming interface (API). It allows programs that rely on authentication to be written independently of the underlying authentication scheme. In PCAD, PAM enables Slurm to guarantee the exclusive access on the resources for only users that have jobs executing on it. It is also used to validate the authentication with private keys instead of passwords (The default approach of LDAP).

Ganglia [8] is a distributed monitoring tool for high-performance computing systems, clusters, and networks. The software is used to view either live or recorded statistics covering metrics such as CPU and network utilization for many nodes. Every PCAD resource has the ganglia client. The overall monitor is publicly available on PCAD's website [4].

NFS [9] (Network File System) is a distributed file system protocol originally developed by Sun Microsystems, allowing a user on a client computer to access files over a computer network much like local storage is accessed. PCAD uses NFS to provide the users a directory on all nodes. The NFS server is on the PCAD front-end and has a total of 12TB for all users.

Another critical aspect of managing resources is security. PCAD uses two software to deal with this aspect. The Snoopy library [10] capable to log all commands and arguments on a system. And the Fail2ban [11], that is an intrusion prevention software framework that protects computer servers from brute-force attacks. Written in the Python programming language, Fail2ban can run on POSIX systems that have an interface to a packet-control system or iptables.

The time of the different machines is synchronized in PCAD using NTP [12] (Network Time Protocol). It is a networking protocol for clock synchronization between computer systems over packet-switched, variable-latency data networks. In operation since before 1985, NTP is one of the oldest Internet protocols in current use.

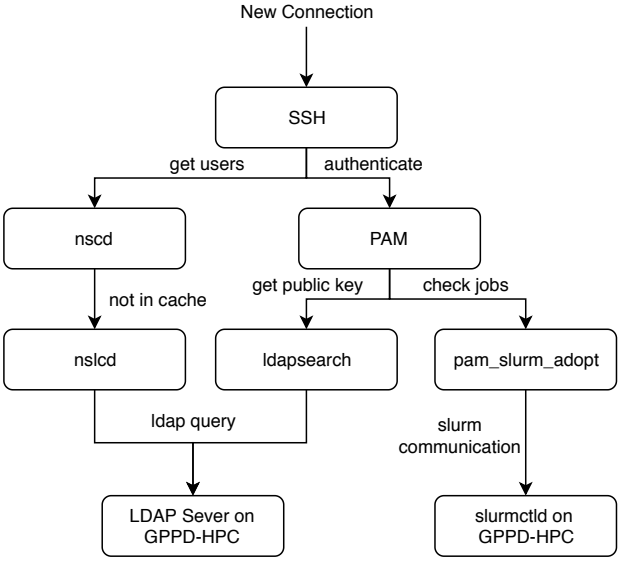
User Interaction and Resource Reservation: The job submission is carried out accessing the PCAD front-end using `SSH` and either using `salloc` or `sbatch` Slurm commands. The commands require to specify a machine's partition and the total expected time (maximum execution time) for the job. The majority of the partitions have a time limit of three days. The global view of some of the used software's interaction can be viewed when inspecting the authentication process. The authentication flow is presented in Figure 1. Basically, when a new connection is made to a resource node or the front end via (and only via) `ssh` a series of software are consulted to validate the authentication. First, SSH needs to check if the user exists and get common Linux information, including user's groups and location folder. This is made thought name service requests cache process `nscd`. In case the requested user is not present in the cache, `nscd` will consult `nslcd`

[3]The SDumont website: https://sdumont.lncc.br/

[4]The PCAD glanglia link: http://gppd-hpc.inf.ufrgs.br/ganglia/

that will query the LDAP server on GPPD–PCAD. If the user is found, the SSH now needs to confirm the credentials using the user public key. To get this information, SSH uses PAM to execute a bash script and query the LDAP server to return the key. In case the key is valid, the user is authenticated, and now PAM needs to confirm if the user can access the machine, i.e., if there is a Slurm job running on it (This step is only executed on the resource nodes and not in the front-end). Slurm provided a PAM module, pam_slurm_adopt, that will consult if the user has jobs running on the machine. In case the user has a job, PAM will validate the user and grant the access. If in any part of this process an invalid operation is done the connection is closed.

Fig. 1. Authentication process and the respective software used



III. Utilization Statistics

This Section presents resource utilization since the implementation of the current managing approach. The data of jobs executed was collected using Slurm, starting at 2018-10-23 and ending on 2019-08-26. There were a total number of jobs of 292836 by 55 different users. Different aspects can be used to present the resources utilization. Figure 2 presents the density of the size of the jobs in hours, where on the X-axis there is the size of the job in hours and the Y-axis the total density computed by `stat_density` R function. It is possible to check the majority of jobs are smaller than three hours, with a slight pick on 12 hours. Also, the number of jobs with size greater than 24 hours is small.

Figure 3 presents the daytime utilization of the partitions. Where the X-axis is the hour of daytime (24 hours) and in the Y-axis there is the total number of jobs that were executed in that time instant. The data show that there is a pick of utilization during work hours (8:00-17:00) when the count of executing jobs is increasing. Moreover, during the night, the resources utilization is gradually decreasing, probably justified because the jobs begin to end.

Fig. 2. Density of jobs size in hours (of jobs with duration greater than 2 minutes) in different partitions

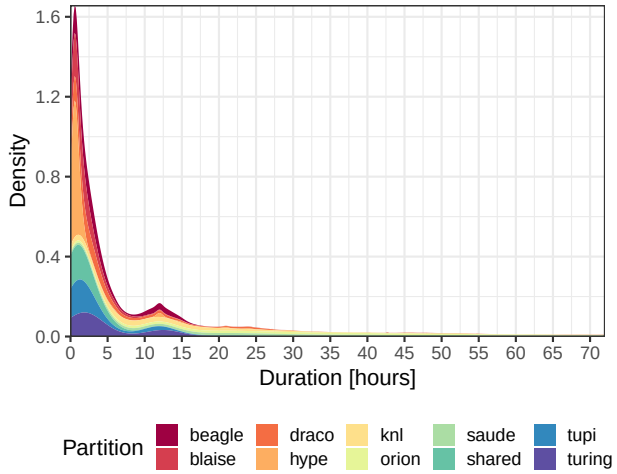


Fig. 3. Daytime utilization of the resources partition (of jobs with duration greater than 2 minutes)

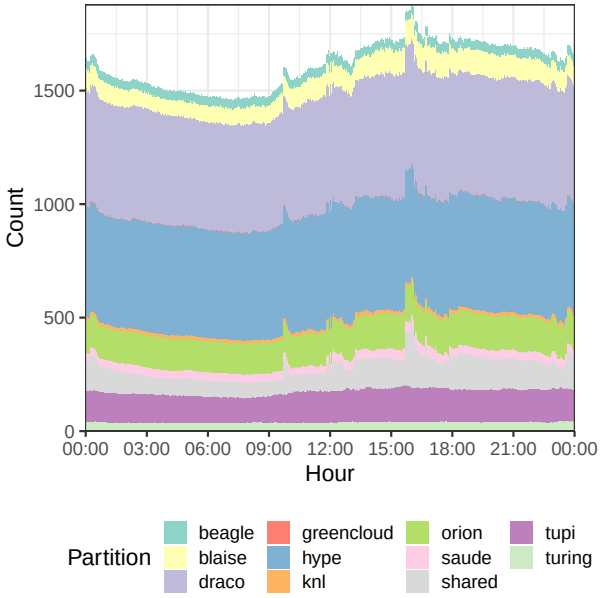
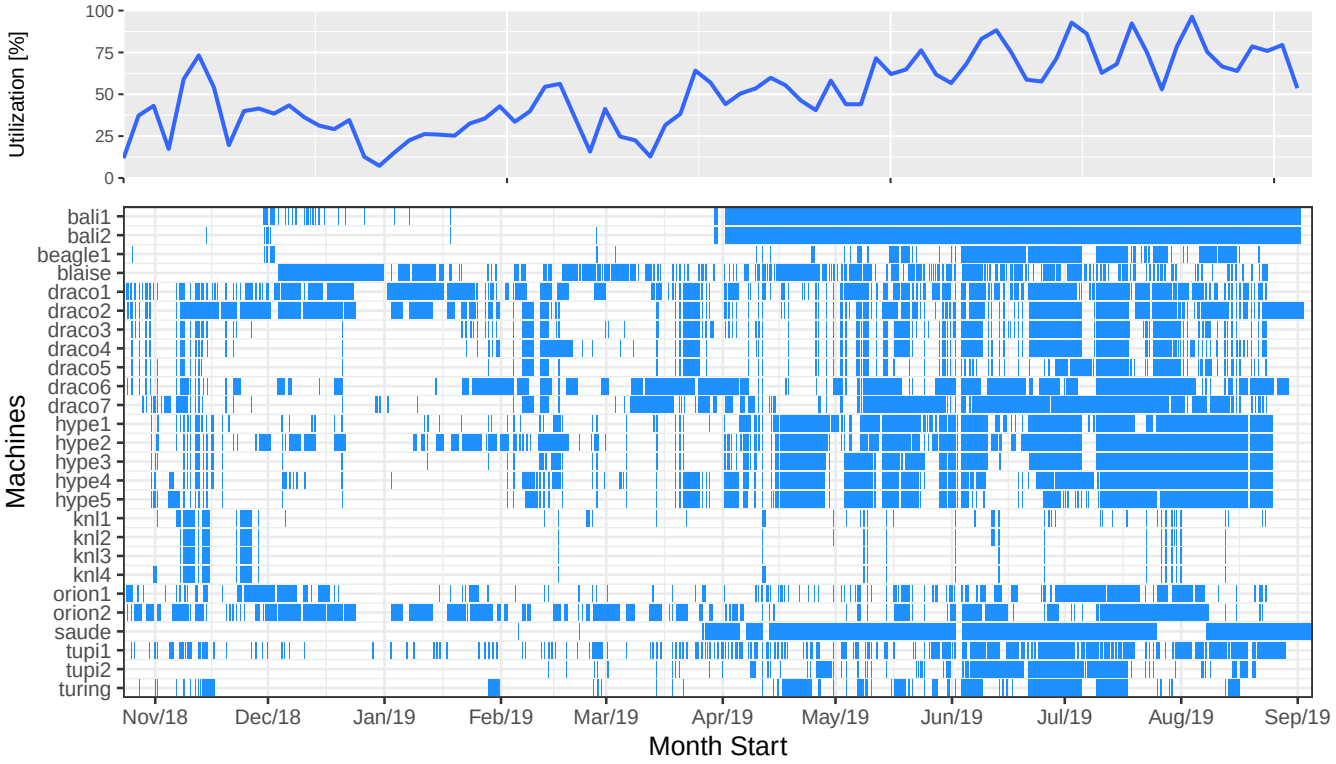


Figure 4 presents the occupation of the resources by jobs over time since the installation of the system management in October 2018 until the end of August 2019. The top panel presents the total utilization of the resources per day. The bottom panel presents the utilization per resource. The vertical red line represents the day that the data was taken. With the aid of this visualization, it is possible to check the periods of more utilization, in the same way, it gives a notion of the most utilized resources. For example, the `knls` machines are the lesser utilized resources, while the `hypes` are been used extensively during the last months. From the data, it is also possible to notice that the resources are being more utilized in 2019 when compared to 2018.

Fig. 4. Machines Utilization



IV. Conclusion

This paper presented a study case of managing HPC resources thought new and modern software solutions that permit hardware sharing with different users. The hardware infrastructures are presented together with the software stack used. Also, we present some real utilization data of the resources. First, we present the hours that the machines were most utilized, that is the work hours. Second, we present the distribution of jobs size submitted; the majority is smaller than two hours. Third, the Gantt chart diagram of individual resource utilization, showing the increased use of the resources in the last months. Future work on PCAD includes the study of different batch schedulers like OAR and enabling users to deploy their custom operating system images. Also, other solutions to report utilization can be used to show PCAD's user behavior.

Acknowledgements

This study was financed by the "Coordenação de Aperfeiçoamento de Pessoal de Nível Superior" (CAPES) - Finance Code 001, the National Council for Scientific and Technological Development (CNPq), and the projects: FAPERGS MultiGPU (16/354-8), FAPERGS GreenCloud (16/488-9), FAPERGS Internacionalização (19/711-6), the CNPq project 447311/2014-0, the CAPES/Brafitec EcoSud 182/15, the CAPES/Cofecub 899/18, Petrobras (2016/00133-9 and 2018/00263-5), and donations by HPE and Intel under the *Lei da Informática* grants.

References

[1] N. Capit, G. Da Costa, Y. Georgiou, G. Huard, C. Martin, G. Mounié, P. Neyron, and O. Richard, "A batch scheduler with high level components," in *CCGrid 2005. IEEE International Symposium on Cluster Computing and the Grid, 2005.*, vol. 2. IEEE, 2005, pp. 776–783.

[2] D. Balouek, A. Carpen Amarie, G. Charrier, F. Desprez, E. Jeannot, E. Jeanvoine, A. Lèbre, D. Margery, N. Niclausse, L. Nussbaum, O. Richard, C. Pérez, F. Quesnel, C. Rohr, and L. Sarzyniec, "Adding virtualization capabilities to the Grid'5000 testbed," in *Cloud Computing and Services Science*, ser. Communications in Computer and Information Science, I. I. Ivanov, M. van Sinderen, F. Leymann, and T. Shan, Eds. Springer International Publishing, 2013, vol. 367, pp. 3–20.

[3] A. B. Yoo, M. A. Jette, and M. Grondona, "Slurm: Simple linux utility for resource management," in *Workshop on Job Scheduling Strategies for Parallel Processing*. Springer, 2003, pp. 44–60.

[4] TOP500, "TOP500 statistics list," https://www.top500.org/statistics/list/, 2018, accessed: 2019-05-24.

[5] Canonical, "Ubuntu," https://ubuntu.com/, 2019.

[6] K. Zeilenga, "Lightweight directory access protocol (ldap): Technical specification road map," 2006.

[7] P. A. M. PAM, "Unified login with pluggable authentication modules (pam)," 1995.

[8] M. L. Massie, B. N. Chun, and D. E. Culler, "The ganglia distributed monitoring system: design, implementation, and experience," *Parallel Computing*, vol. 30, no. 7, pp. 817–840, 2004.

[9] B. Nowicki, "NFS: Network file system protocol specification," 1989.

[10] M. Eriksen and M. Baker, "Log every executed command to syslog (a.k.a. snoopy logger)," 2010. [Online]. Available: https://github.com/a2o/snoopy

[11] C. Jacquier, "Fail2ban," 2010. [Online]. Available: https://www.fail2ban.org/wiki/index.php/Main_Page

[12] D. Mills, "Network time protocol (version 3) specification, implementation and analysis," 1992.

FPGA Code Generator for High-Level Neural Network Descriptions

Matheus Woeffel, Gabriel Freytag, Philippe O. A. Navaux
Federal University of Rio Grande do Sul (UFRGS)– Porto Alegre, RS – Brazil
matheuswoeffel@gmail.com, {gfreytag,navaux}@inf.ufrgs.br

Abstract

FPGAs are known for their inherent architecture parallelism and their low power consumption. Despite advantages for specific application ranges, this architecture has a more laborious workflow that can inhibit the implementation and enjoyment of all architectural features. High-level synthesis alternatives target a large number of applications and therefore, while representing easier and faster implementation alternatives, often do not retrieve the full benefits of the architecture. Hence, this work presents the proposal and current state of the implementation of a tool that automatically converts high-level descriptions of neural networks to synthesizable VHDL for FPGAs. With a narrower range of applications, it is possible to combine customization, ease of implementation, and full use of the concepts inherent to the problem as well as the characteristics of the architecture itself.

1. Introduction

Research in Neural Networks (NN) in recent years has advanced in such a way that today, it is possible to use neural networks in a large number of problems [9] [1]. Some of the most common applications of NN are natural language processing, image recognition, and generic approximation algorithms. However, to compute these problems, a large amount of computational power and consequently, energy is required.

Neural Networks are usually run in High-Performance Computing systems with a large number of CPU and GPU cores, reducing the amount of time required by large applications [10]. Despite the reduced running time, the amount of energy consumed by theses architectures is quite high. As an alternative to reduce energy consumption, Field Programmable Gate Array (FPGA) architectures are gaining prominence due to its intrinsic parallelism and low energy consumption. [7]

Although being highly parallel and power-efficient, a significant drawback of FPGA architectures is the programming complexity. As in these architectures, the hardware needs to be configured to do the desired computations with extremely low-level hardware descriptions. Some of the most common programming languages for FPGAs are Verilog, VHDL, and RTL. In this way, developing applications for FPGAs involves specific knowledge of architecture itself as well as a significant amount of time.

Thus, the objective of this work is to present the proposal and the current state of the implementation of an automated Neural Network code generator for FPGAs based on a high-level description. With this tool, it will be possible to convert textual descriptions of generic neural networks into synthesizable VHDL code that can be then executed by FPGAs. This tool will be open-source to turn it easier to use by the community and also easy to modify in the future.

2. Background

Neural networks are a computational model inspired by the way processing is performed in the central nervous system of animals. Its basic unit is the neuron, which is a module responsible for making the scalar product of an input vector by a corresponding weight vector and then summing it with an offset and applying it to an activation function. The activation function can be any mathematical function, but usually derivable and soft functions such as *sigmoid*, *tanh* and *ReLU* are chosen. The neurons are then replicated in parallel to form a layer, and their concatenation is performed to form the neural network itself. This concatenation is performed by adopting the output of each neuron from an anterior layer as the input of the current layer.

Recently, this computational model has gained emphasis in the literature, due to the possibility of automatically training the network by *machine learning* algorithms. The use of such model usually involves two distinct phases: *classification/inference* and *training*. In the *classification/inference* the neural network is used to process the input, while in the *training* phase *machine learning* algorithms are used to find the weights and bias of the neurons to optimize the functionality implemented by the neural network.

In FPGAs, with rare exceptions, neural networks are implemented considering only the inference/classification phase, being the training phase usually performed in GPUs or heterogeneous architectures [14]. The present tool currently only considers the classification phase, but as future work it would be possible to evaluate alternatives for implementing the training phase also targeting the FPGAs architectures.

3. Related Work

In [7] the authors implemented a fixed point deep neural network for recognizing handwritten digits in a Xilinx FPGA. After comparing the results with GPU and CPU implementations, they found that the FGPA implementation consumed significantly less energy compared to the CPU and GPU implementations.

Given the difficulty of deploying varied applications using only hardware description languages, several manufacturers provide high-level synthesis tools to enable software application deployment and automatic conversion to synthesizable code for FPGAs. Due to the generality of the targeted applications of such tools, it is expected that the full benefit of the architecture will not be fully extracted. There are several tools in the literature that aims at enabling high level conversion using different interfaces whose target applications are specifically neural networks (like fpgaConvNet [12], DeepBurning [14], Angel-Eye [4], ALAMO [6], Haddoc2, DnnWeaver [8], and Caffeine [15]) and each of them adopt a different architectural model to implement Neural Networks in FPGAs using different interfaces and targeting different boards [13].

In [3], beside reporting the difficulty in its implementations, which are based on hardware description languages, the authors proposed an automated framework for mapping neural networks on FPGA using RTL-HLS hybrid templates. Compared to the present work, we aim to use RTL exclusively, without using other High-Level Synthesis tools.

Our work aims to target deep neural networks in general, and as many boards as possible, not being limited by manufacturers libraries or synthesis tools. Moreover, our tool will be open source and customizable to turn it easier to incorporate the user needs and ideas. Due to the direct mapping of the problem domain modules and the implementation modules, it is easy to switch between different solutions.

4. Implementation

The language selected for development was Python, due to the large number of functions that support string manipulation, which was very useful when generating the VHDL files describing the neural network. The implementation was divided in a modular way, so that the development was more easily tested, and as later justified, customizable. The following modules were considered:

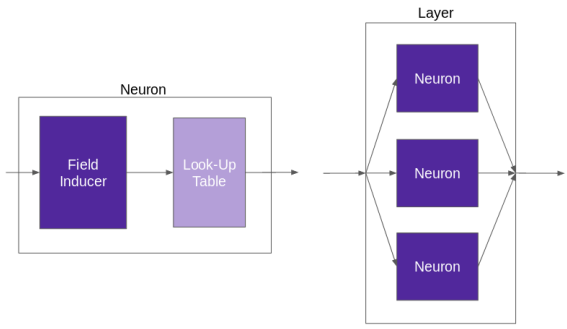


Figure 1. Neuron and Layer implementation.

- Field Inducer - Module responsible for calculating the scalar product of the input vector multiplied by the weight vector, added with the bias.
- Activation Function - Module that implements an Activation Function, which is applied to the field inducer output.
- Neuron - Module that joins the field applicator to the activation function. applicator, responsible for the synchronization and control logic of the two modules above.
- Layer - Module that gathers the neurons that define it and assumes the responsibility of mapping the inputs and weights to the respective neurons that it encompasses.
- Network - Module that brings together the layers and takes responsibility for synchronization and control between them.

In Figure 1 we show the implementation of the neurons and the layers using the five modules previously described. With the objective of facilitating posterior changes and generality of the neural networks implemented, all the model concepts defined above were directly mapped to computational modules chosen to solve the problem. Any changes in the strategy of implementation concerned with some module it is then easily mapped and modified.

From the definition of the modules to be implemented, the following design alternatives were determined for each one: considering the linear computation of the neurons, the use of parallel multipliers together with an accumulator and a state machine or the use of multipliers in parallel with tree adders were considered as alternatives. Considering the application of the activation function were found as alternatives the use of Look-up tables, approximation by *sigmoid-allipi* [11] and other linear piece-wise approximations [2]. Some alternatives for the architecture of the layers are the

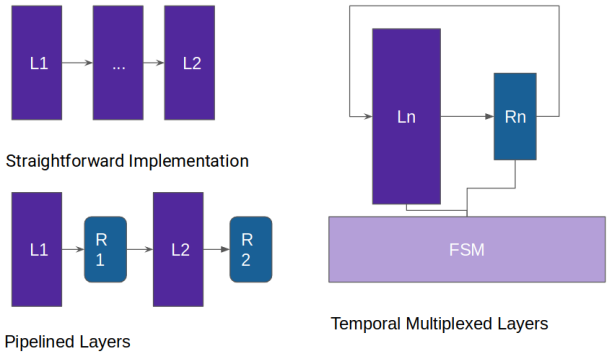


Figure 2. Layer architecture alternatives.

integration of cascaded layers with and without pipeline, as well as temporal multiplexing of a single layer together with a FSM [5]. The difference of the three alternatives can be observed in Figure 2.

Each module was represented as a class of the developed software, responsible for implementing the necessary functions of each module and the respective generation of the VHDL code that implements it. In order to assemble common responsibility for VHDL syntax pertaining to all entities used in the project, a class called VHDL entity was created, which brings together the default structure of all files as well as the instantiating syntax functions common to different components.

One of the reasons to choose modular design was the possibility of fast customization. Given the relative independence of modules, it is easy to change their implementation independently and evaluate different strategies, also supporting the goal of facilitating future code modifications for various purposes and the process of evaluating different strategies.

To measure the consumed area of each implementation alternative, the numbers of FFs, DSPs, RAMs, BRAMs, ALUTs will be collected from reports generated by the Quartus Pro tool after the synthesis process. To assess the consequences in terms of performance and energy, the neural network classification phase will be performed using the Arria X FPGA to extract execution time, instant energy consumption and power-delay product. This will allow a futher refinement of the tool and a future comparison with other deployment alternatives, such as the high-level Intel HLS compiler[1].

1 Intel HLS - https://www.intel.com/content/www/us/en/software/programmable/quartus-prime/hls-compiler.html

Area Usage	
Resource Name	Resource Usage
ALM	404
LABs	50
Logic Registers	171
DSPs Blocks	6

Table 1. Arria 10 FPGA resource usage.

5. Initial Results

Considering the different design alternatives specified before, the present work implements the neurons using parallel multipliers in conjunction with a finite state machine and adder, layer architecture using pipeline and activation function implemented from look-up tables.

A problem encountered during the implementation was the need to keep the tool compatible with different models from the same manufacturer or even from different manufacturers, which made it difficult to use specific libraries normally provided in conjunction with each company's synthesis software. To work around this problem, only IEEE-standardized VHDL libraries were used to avoid potential compatibility issues and keeping the tool as generic as possible.

The neural network tested so far consisted of 2 layers with 2 neurons in the first layer and 1 neuron in the last layer. The network implements a simple XOR function of 2 inputs and was trained in software using the genann[2] library, which was also used for testing and debug during the implementation phase of the tool. The results for area usage are summarized in Table 1.

Due to the usage of look-up tables for the activation function, the data width used was of only 8 bits, 4 for each fractional and integer portion. To test the tool with more realistic networks will be necessary to implement approximation of the *sigmoid* or other activation functions, because of the exponential nature characteristic to look up tables.

6. Conclusions and Future Work

In this paper, we present the proposal and current state of the implementation of an automated FPGA code generator for high-level neural network descriptions. We show the various obstacles and alternatives encountered during the process of design and implementation of the tool to automatically convert high-level descriptions of neural networks to synthesizable VHDL code.

One of the major obstacles was the absence of standard libraries between different vendors, which made us implement the tool from a very low level of abstraction. Another

2 Genann - https://github.com/codeplea/genann

difficulty was the process of debugging and testing,that was laborious and error-prone due to the RTL exclusive implementation.

Given the current temporary results it is difficult to drawn symbolic conclusions and infer considerations about the project alternatives chosen. To evaluate the results and drawn more deep conclusions will be necessary to test more realistic networks and compare the results with other options presented in the literature. The aspects that will be evaluated include area usage, execution time and mainly energy consumption, using the same metrics mentioned in 4.

Regarding interface issues, future work would include the development of a caffe[3]-like scripts parser to provide an interface well known for the implementation of neural networks applications. Using an already established interface will facilitate the use and future researches regarding the tool here presented.

References

[1] U. R. Acharya, H. Fujita, S. L. Oh, Y. Hagiwara, J. H. Tan, and M. Adam. Application of deep convolutional neural network for automated detection of myocardial infarction using ecg signals. *Information Sciences*, 415:190–198, 2017.

[2] K. Basterretxea, J. Tarela, and I. Del Campo. Digital design of sigmoid approximator for artificial neural networks. *Electronics Letters*, 38(1):35–37, 2002.

[3] Y. Guan, H. Liang, N. Xu, W. Wang, S. Shi, X. Chen, G. Sun, W. Zhang, and J. Cong. Fp-dnn: An automated framework for mapping deep neural networks onto fpgas with rtl-hls hybrid templates. In *2017 IEEE 25th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, pages 152–159. IEEE, 2017.

[4] K. Guo, L. Sui, J. Qiu, J. Yu, J. Wang, S. Yao, S. Han, Y. Wang, and H. Yang. Angel-eye: A complete design flow for mapping cnn onto embedded fpga. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 37(1):35–47, 2017.

[5] S. Himavathi, D. Anitha, and A. Muthuramalingam. Feedforward neural network implementation in fpga using layer multiplexing for effective resource utilization. *IEEE Transactions on Neural Networks*, 18(3):880–888, 2007.

[6] Y. Ma, N. Suda, Y. Cao, S. Vrudhula, and J.-s. Seo. Alamo: Fpga acceleration of deep learning algorithms with a modularized rtl compiler. *Integration*, 62:14–23, 2018.

[7] J. Park and W. Sung. Fpga based implementation of deep neural networks using on-chip memory only. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1011–1015. IEEE, 2016.

[8] H. Sharma, J. Park, D. Mahajan, E. Amaro, J. K. Kim, C. Shao, A. Mishra, and H. Esmaeilzadeh. From high-level deep neural models to fpgas. In *The 49th Annual IEEE/ACM International Symposium on Microarchitecture*, page 17. IEEE Press, 2016.

[9] B. Shi, X. Bai, and C. Yao. An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. *IEEE transactions on pattern analysis and machine intelligence*, 39(11):2298–2304, 2016.

[10] D. Strigl, K. Kofler, and S. Podlipnig. Performance and scalability of gpu-based convolutional neural networks. In *2010 18th Euromicro Conference on Parallel, Distributed and Network-based Processing*, pages 317–324. IEEE, 2010.

[11] M. Tommiska. Efficient digital implementation of the sigmoid function for reprogrammable logic. *IEE Proceedings-Computers and Digital Techniques*, 150(6):403–411, 2003.

[12] S. I. Venieris and C.-S. Bouganis. fpgaconvnet: A framework for mapping convolutional neural networks on fpgas. In *2016 IEEE 24th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, pages 40–47. IEEE, 2016.

[13] S. I. Venieris, A. Kouris, and C.-S. Bouganis. Toolflows for mapping convolutional neural networks on fpgas: A survey and future directions. *ACM Comput. Surv.*, 51(3):56:1–56:39, June 2018.

[14] Y. Wang, J. Xu, Y. Han, H. Li, and X. Li. Deepburning: automatic generation of fpga-based learning accelerators for the neural network family. In *Proceedings of the 53rd Annual Design Automation Conference*, page 110. ACM, 2016.

[15] C. Zhang, G. Sun, Z. Fang, P. Zhou, P. Pan, and J. Cong. Caffeine: Towards uniformed representation and acceleration for deep convolutional neural networks. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2018.

3 Caffe - https://caffe.berkeleyvision.org/

Study Towards Enhanced Performance Analysis In QR MUMPS Task-Based Sparse Factorization

Marcelo Cogo Miletto, Lucas Mello Schnorr
Institute of Informatics, Federal University of Rio Grande do Sul - UFRGS, Porto Alegre, Brazil

***Abstract*—Linear algebra solvers are commonly present in scientific computing. They help in the research process through a cheaper way of experimentation that numerical applications provide. As the problems in this context are continually growing in size and complexity, the time spent to compute these experiments increase as well. For keeping the use of these applications viable, parallel high-performance solutions are an essential tool nowadays. In this work, we study a sparse direct solver called qr_mumps. It uses the StarPU library to explore the task-based parallelism. We focus on finding information that can be used to enable better performance analysis of the application.**

I. Introduction

Many research areas depend on linear algebra fundamentals. Numerical applications used in areas such as economy, mechanics, and geophysics, continuously end up with the problem of solving systems of linear equations [1]. These applications play an essential role in the research process, they allow simulating the reality using a computational environment, thus, providing a cheaper way of experimentation from which we can validate hypothesis or make predictions.

This kind of application is characterized by intensive computational operations, demanding a lot of computing power to its executions. As these applications play an essential role in scientific computing. The linear algebra methods developed a long time ago to solve such problems, needed to be updated to face the size of the ever-increasing problems that arise in this context. Thus, to reach good performance, many solutions rely on parallel computing, exploring multiple cores and accelerator devices (GPUs) to speed up calculations providing results faster. Ondes3D [2] for example, uses a cluster to predict earthquake ground motion, and RAFEM [3] uses GPU to accelerate the simulation of a medical procedure to treat hepatic cancer.

Besides the computational intensity, most of the real problems lead to large and sparse matrices because of the form they are structured. This adds a new level of complexity, implying in the use of efficient data structure to represent the matrix and strategies to take advantage of its sparsity, saving computational effort and memory. Depending on how the solver performs operations and on the matrix pattern, all the sparsity can be lost due to the introduced fill-ins. This way, parallel implementations of both direct and iterative classes of solvers depend on the permutation or reordering of rows and columns [4] to optimize what they can get from sparsity. Parallel algorithms for solving sparse systems must consider this irregularity characteristic in the workload when partitioning the data. Otherwise, poor resource utilization can bound application performance. A good way to fight this problem is by adopting a task-based approach, breaking the problem into a set of smaller tasks and letting a runtime system responsible for their scheduling. For this, libraries like StarPU [5] can be used to handle data partitioning, providing a dynamic runtime system and different scheduling policies.

Some of these runtime scheduling policies are based on user given scheduling hints, such as the performance model of application tasks. These hints help to eliminate the source of load imbalance by using a measure to quantify the computation done by each processing unit according to task costs. Thus, it helps to distribute the workload evenly among them. This strategy is commonly used in high-performance libraries [5]. However, besides this scheduling utility, the performance model of an application can be used both to faithfully simulate the application execution to help developers on its optimization, and to compare a real execution of the application with a prediction based on the model.

The qr_mumps [6] is a task-based solver that uses the StarPU library. It already has a performance model that was implemented in [7]. As StarPU enables tracing the application behavior, what we propose in this work is to enrich the generated traces with this performance model, attaching the quantified computational cost of the application tasks in the traces generated by a real execution. This way, we can detect anomalous tasks by comparing the time one task took to execute in the real platform with the expected time given by the model, enabling a more in-depth analysis of the application characteristics.

The rest of the paper is organized as follows: Section II presents libraries and sparse solvers, the task-based programming model, details of qr_mumps and the performance model. Section III presents our motivation and work proposal describing our methodology. Finally, Section IV concludes the paper and presents future directions for this work.

II. Background

This section presents information about parallel sparse solvers, the StarPU library, going trough its programming model characteristics and listing its capabilities in terms of scheduling task workloads. Also, we provide a description of the qr_mumps concepts and implementation details, plus an overview of its performance model.

A. Parallel Sparse Solvers

With the advent of multicore and the accelerators devices, also considering them in a distributed memory system. New challenges arose in the developing of linear algebra solvers for such complex heterogeneous systems. As the developed solution should adapt a complex workload to the different speeds and capacities of the computing resources and different costs for communications, this problem becomes even harder. Load balancing, communication, memory consumption, and portability are some of the essential aspects that should be considered when developing such algorithms.

Numerous libraries and packages implement high-performance sparse solvers. They all adopt a standard set of basic routines defined by BLAS [8], which have implementations with hand-tunned code for specific devices like for example Intel MKL, AMD BLIS and openBLAS for CPUs, CUBLAS and MAGMA BLAS for GPUs. Libraries like PARDISO [9] and MAGMA [10] provides several solvers focusing both on shared and distributed memory multiprocessors and heterogeneous platforms containing multiple cores and GPUs. MAGMA uses a hybrid methodology that breaks the problem into tasks which a dynamic runtime system is responsible for scheduling. Besides these packages that offer numerous sparse solver methods, we can also look into individually developed solvers like MUMPS [11]. It was one of the first multifrontal QR solvers, and it was first developed to work on shared-memory multiprocessors, exploring the parallelism using the multifrontal method [12], based on the concept of an elimination tree [13]. This solver uses a hand-coded task queuing system to orchestrate tasks in this tree structure along with a multithreaded BLAS implementation to explore parallelism further.

Employing a task-based approach to sparse multifrontal solvers enable to handle the irregular workloads that come with the different task granularities and characteristics. The qr_mumps is a further step when compared to MUMPS. This newer solver explores more fine-grained parallelism using StarPU to handle the task-based parallelism. Trough it, various architectures can be targetted as there are the different BLAS implementations that can be used in the application, allied with the StarPU dynamic runtime system which is capable of scheduling tasks over heterogeneous platforms, using a wide range of scheduling policies.

B. Task-Based Programming Model in StarPU

The task-based programming paradigm has a simple yet expressive way to describe problems, offering a portable way to deliver performance of complex workloads over many cores [14]. The application can be described using the concept of a Directed Acyclic Graph (DAG), dividing the whole computational workload into smaller tasks. The DAG nodes represent computational tasks, and the edges between them are their dependencies. A task can only execute if all of its dependencies are satisfied. Runtime systems are responsible for managing and distributing the DAG tasks among the available processing units during execution time.

StarPU is a C/C++ task programming library for heterogeneous platforms that has a dynamic scheduling runtime system, supporting multicore CPU/GPU in both shared and distributed memory systems. Its programming model is based on the concept of a codelet, which is a piece of code that defines what instructions a given task executes. Task dependencies are described in terms of the access pattern of a task to a specified data handle, which is the structure used to partition the problem data. Additional task information like priority and performance model can be associated with tasks. This extra information can be used by some of StarPU scheduling algorithms to achieve better performance, for example, deciding which tasks are going to be executed by the CPU and the GPU. The available scheduling policies algorithms can be into two categories:

- **Non performance modelling policies**: The eager scheduler uses a central task queue from which all the workers (computational resources) get tasks to work on. The random scheduler have a task queue per worker and distribute them randomly according to workers overall performance. The work stealing have a task queue per worker. When a worker becomes idle, it steals from the most loaded queue. The local work stealing policy consider neighbor workers for stealing and also takes into account task priorities. The prio scheduler sort the tasks by priority in a central queue. Heteroprio can define different priorities for different processing units.
- **Performance model-based task scheduling policies**: The DM (deque model) policy tries to minimize tasks finishing time, this is done as soon as the tasks become available. DMDA (DM data aware) also consider the data transfers cost. The DMDAR (DMDA ready) gives privilege to tasks whose data is already available on the target device. The DMDAS (DMDA sorted) consider task priority besides the data transfer cost. The DMDASD (DMDAS decision) is similar to the DMDAS but considers priority to find the minimum completion time.

C. Multifrontal QR Factorization

Direct methods like LU, Cholesky, and QR, are preferable because of their robustness when compared to iterative methods, whose efficiency depend on the numerical properties of the input matrix [1]. Among the direct methods, the QR factorization and its variations are very popular because of the capacity to achieve high performance and its numerical robustness, especially the dense Householder QR factorization and the multifrontal QR [15]. It can be used to solve sparse systems of linear equations and the least-squares problem.

Given a sparse matrix, we can take advantage of its sparsity in the following ways: (1) economize memory not storing all of its zero values explicitly, (2) perform cheaper operations because the zero values can be skipped, and (3), parallelize operations that modify distinct nonzero subsets of the matrix. The last item is the one responsible for building the elimination

tree concept, which is the main idea behind the multifrontal technique. The elimination tree nodes hold smaller dense matrices called frontal matrices or fronts, and the tree is then traversed bottom-up, enabling the processing of different branches in parallel. This source of parallelism in the multifrontal method is called tree parallelism [12]. Other factors that influence the sparse solvers performance is the ordering applied to the sparse matrix structure. It can harshly impact the tree structure, leading to unbalanced trees in the case of the multifrontal method and can control the generated fill-in level. To treat this, applications use fill-reducing matrix ordering techniques like the Cuthill-McKee, Average Minimum Degree, and the Nested Dissection orderings [16].

The elimination tree can be seen as a DAG of tasks, where the nodes are factorization tasks, and they depend on the previous child nodes factorizations. In the classical approach, the multifrontal method is divided into two steps: assembly and factorization. The assembly task groups a set of rows that have nonzero elements in the pivot column (the one that is being eliminated) along with the other elements that are affected by the factorization, originating a front. Then, in the factorization step, this front matrix is factorized through the Householder method, producing one row of the R factor, a part of the Q factor, and the contribution block that is incorporated to its parent front.

D. QR MUMPS: Fine-Grained Task-Based Multifrontal QR

The previously detailed approach has two limitations. As we go upward the tree, the tree parallelism shrinks, and depending only on this parallelism source does not produce good speedups [17]. Also, threads cannot start processing a parent node until all of its children were factorized, limiting, even more, the parallelism and hampering reaching a good load balance. This way, qr_mumps goes further by exploring fine-grained tasks in the multifrontal method on top of StarPU, exploring efficient ways to perform the nodes factorization using 1D and 2D block factorization algorithms, allowing starting the computation of a father node concurrently with its children.

The elimination tree structure in qr_mumps is diluted in a finer-grained set of tasks. The task creation and the matrix partitioning are fully dynamic, depending on the front matrix size, the corresponding tree node is partitioned into 1D block columns or in 2D tiles. Also, it can be pruned, which means that the node or a pruned subtree will be computed sequentially by a single thread. Only nodes that represent a small portion of the factorization cost are pruned. The LAPACK-based factorization routines in qr_mumps were slightly modified by its authors to allow the user to pass a parameter defining a value for the internal block size. Helping thus to reduce the generated fill-in by the blocked operations. There are nine types of tasks that the application threads can execute:

- **init/clean front**: initialize and assemble a frontal matrix or clean it, deallocating from memory.
- **init/clean block**: initialize and clean blocks of a specific frontal matrix.
- **geqrt**: computes the QR factorization in a front matrix block, generating a set of Householder reflectors and part of the R factor.
- **gemqrt**: update the matrices block at the right of the diagonal with the previously calculated reflectors by geqrt.
- **tpqrt**: eliminate the blocks below the diagonal factorized by geqrt.
- **tpmqrt**: updates the remaining blocks in the front matrix.
- **do subtree**: define the factorization of an entire subtree of the elimination tree to be computed sequentially using the previous described routines.

To generate this fine-grained version of the elimination tree, the application performs a sequence of steps: given a sparse input matrix A, first, a fill-reducing ordering is applied, for this, it uses libraries such as SCOTCH, METIS and COLAMD. Then, a symbolic factorization step computes the generated fill-in and the structures of all the fronts to be incorporated in the tree. Pruning rules define which tree nodes will be pruned. With the structure of the tree and the fronts defined, the front factorization is partitioned into a set of tasks defined by the block sizes. Those tasks are submited following the bottom-up traversal of the tree, and just the tasks that are part of initialized frontal matrices are visible to the runtime system.

E. The Application Performance Model

Performance modeling of individual computational tasks is of great value to overcome some problems [7]. It enables to obtain trustful performance predictions using low-cost simulations, helps in application tunning, and supports complex scheduling algorithms. In the StarPU context, it is possible to emulate/simulate StarPU applications with SimGrid [18]. Also, the StarPU library supports performance model-based scheduling algorithms, which can provide better scheduling decisions by knowing the targeted architecture capabilities, communication costs, and by estimating the computational weight of each task.

Although we know that simulations are useful in many ways and because it is hard for a model to capture every factor that can affect performance, there can be differences in the predicted and measured values. This way, performance models values can be used to compare expected results with the real obtained results, highlighting anomalous tasks where some interference in a real experiment perturbed the task duration. For qr_mumps a reliable performance model was already made in [7]. For each factorization task, there is a function that is called in the moment of its creation. The function reads the data structure which the task will work on and calculate the floating point operations. This value is then associated to the task and write in the application trace.

III. Motivation and Methodology

Given the application and its performance model, we want to incorporate this estimated cost per task from the model to the StarPU generated traces for real executions. This way, we have the real execution information collected by StarPU

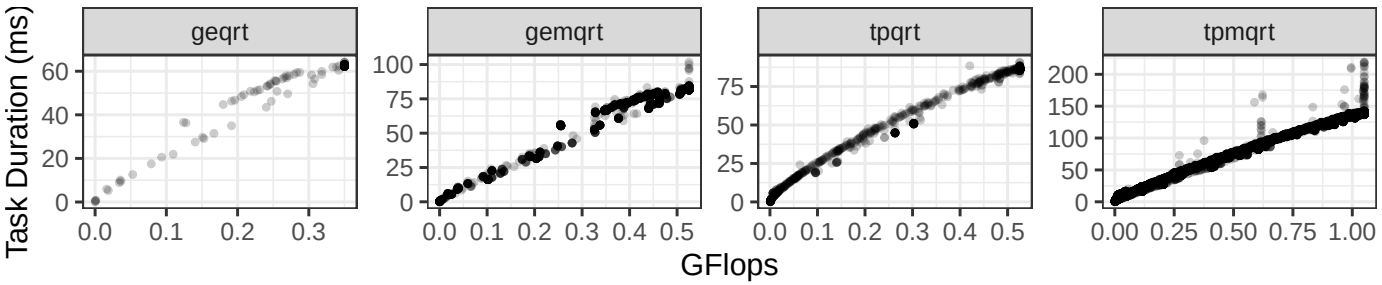


Fig. 1. Duration of tasks according to their calculated GFlops.

plus the estimated cost in floating-point operations (flops) from the model. Since the expected results are trustable, we can estimate the time that a given task should take to execute in a specific computing unit and compare it with the real execution time obtained in the trace. This extra information allows us to detect anomalous tasks in a real execution. By doing so, we can enhance existing application behavioral visualizations like in StarVZ [19].

To achieve our goal of detecting anomalous tasks, we need to combine the information of the expected flops and the task execution time from an execution trace. A model of analysis of variance (ANOVA) can be used to assess how far a real task execution is from what the model predicted. Having this data, we can use visualization techniques to highlight tasks that have a duration that was not expected by the performance model. The Figure 1 show the relation between task duration and its computational weight. We can see that there is a linearly increasing pattern, but some points lie outside of that line. That means that for a similar amount of flops that a task have, there are different duration times.

IV. Conclusion

This work presented an initial description of the qr_mumps application concepts and its performance model. Our study aims to use the model information included into the execution traces generated by the StarPU library. With this new information on the traces, we can enhance application visualization tools to detect anomalous tasks which can lead to further investigations.

Future work needs to explore the floating-point operations values integrated into the StarPU tracing system and work on developing and using visualization techniques to highlight identified anomalous tasks. For the classification of these tasks, an ANOVA model can be employed. With the identification of anomalous tasks, a more in-depth analysis can be made to assess the reason that it presented this abnormal behavior.

References

[1] F. Lopez, "Task-based multifrontal qr solver for heterogeneous architectures," Ph.D. dissertation, Université de Toulouse, Université Toulouse III-Paul Sabatier, 2015.

[2] R. Keller Tesser, L. Mello Schnorr, A. Legrand, F. C. Heinrich, F. Dupros, and P. O. Navaux, "Performance modeling of a geophysics application to accelerate over-decomposition parameter tuning through simulation," *Concurrency and Computation: Practice and Experience*, vol. 31, no. 11, p. e5012, 2019.

[3] M. C. Miletto, "Acelerando uma aplicação de simulação computacional para o processo de ablação por radiofrequência usando gpu," 2018.

[4] Y. Saad, *Iterative methods for sparse linear systems*. siam, 2003, vol. 82.

[5] C. Augonnet, S. Thibault, R. Namyst, and P.-A. Wacrenier, "Starpu: a unified platform for task scheduling on heterogeneous multicore architectures," *Concurrency and Computation: Practice and Experience*, vol. 23, no. 2, pp. 187–198, 2011.

[6] E. Agullo, A. Buttari, A. Guermouche, and F. Lopez, "Multifrontal qr factorization for multicore architectures over runtime systems," in *European Conference on Parallel Processing*. Springer, 2013, pp. 521–532.

[7] L. Stanisic, E. Agullo, A. Buttari, A. Guermouche, A. Legrand, F. Lopez, and B. Videau, "Fast and accurate simulation of multithreaded sparse linear algebra solvers," in *2015 IEEE 21st International Conference on Parallel and Distributed Systems (ICPADS)*. IEEE, 2015, pp. 481–490.

[8] J. J. Dongarra, J. Du Croz, S. Hammarling, and R. J. Hanson, "An extended set of fortran basic linear algebra subprograms," *ACM Transactions on Mathematical Software (TOMS)*, vol. 14, no. 1, pp. 1–17, 1988.

[9] O. Schenk and K. Gärtner, "Solving unsymmetric sparse systems of linear equations with pardiso," *Future Generation Computer Systems*, vol. 20, no. 3, pp. 475–487, 2004.

[10] J. Dongarra, M. Gates, A. Haidar, J. Kurzak, P. Luszczek, S. Tomov, and I. Yamazaki, "Accelerating numerical dense linear algebra calculations with gpus," *Numerical Computations with GPUs*, pp. 1–26, 2014.

[11] P. R. Amestoy, I. S. Duff, and J.-Y. L'excellent, "Multifrontal parallel distributed symmetric and unsymmetric solvers," *Computer methods in applied mechanics and engineering*, vol. 184, no. 2-4, pp. 501–520, 2000.

[12] I. S. Duff and J. K. Reid, "The multifrontal solution of unsymmetric sets of linear equations," *SIAM Journal on Scientific and Statistical Computing*, vol. 5, no. 3, pp. 633–641, 1984.

[13] J. W. Liu, "The role of elimination trees in sparse factorization," *SIAM journal on matrix analysis and applications*, vol. 11, no. 1, pp. 134–172, 1990.

[14] J. Dongarra, S. Tomov, P. Luszczek, J. Kurzak, M. Gates, I. Yamazaki, H. Anzt, A. Haidar, and A. Abdelfattah, "With extreme computing, the rules have changed," *Computing in Science & Engineering*, vol. 19, no. 3, p. 52, 2017.

[15] A. Buttari, "Fine-grained multithreading for the multifrontal qr factorization of sparse matrices," *SIAM Journal on Scientific Computing*, vol. 35, no. 4, pp. C323–C345, 2013.

[16] G. H. Golub and C. Loan, "Matrix computations, forth edition," 2013.

[17] P. R. Amestoy, I. S. Duff, J.-Y. L'Excellent, and J. Koster, "A fully asynchronous multifrontal solver using distributed dynamic scheduling," *SIAM Journal on Matrix Analysis and Applications*, vol. 23, no. 1, pp. 15–41, 2001.

[18] H. Casanova, A. Giersch, A. Legrand, M. Quinson, and F. Suter, "Versatile, scalable, and accurate simulation of distributed applications and platforms," *Journal of Parallel and Distributed Computing*, vol. 74, no. 10, pp. 2899–2917, 2014.

[19] V. Garcia Pinto, L. Mello Schnorr, L. Stanisic, A. Legrand, S. Thibault, and V. Danjean, "A visual performance analysis framework for task-based parallel applications running on hybrid clusters," *CCPE*, vol. 30, no. 18, p. e4472, 2018.

Resolution Impact on Deep Learning Approaches for Diabetic Retinopathy Detection

Francis B. Moreira and Philippe O. A. Navaux
Informatics Institute
Universidade Federal do Rio Grande do Sul
Porto Alegre, Brazil
{fbmoreira, navaux}@inf.ufrgs.br

Beatriz D. Schaan, Josiane Schneiders and Mateus A. dos Reis
Universidade Federal do Rio Grande do Sul
Hospital de Clínicas de Porto Alegre
Porto Alegre, Brazil
bschaan@hcpa.edu.br

***Abstract*—Screening for diabetic retinopathy in diabetic patients is the most effective way to reduce the risk of sight loss. The demand for screening is increasing, and the currently overloaded medical doctors can not efficiently attend it. Therefore, there is a need for a methodology to automate and increase the efficiency of screening. In this study, we discuss the development of deep neural networks to detect diabetic retinopathy. We have achieved 0.93 area under the receiver operating characteristic curve.**

I. Introduction

Diabetes is a chronic disease associated with hyperglycemia due to failure in the insulin release or insulin resistance. It has an increasing prevalence, currently affecting 12% of the Brazilian population [16]. High blood glucose levels (Hyperglycemia) cause damage to the blood vessels of the retina (diabetic retinopathy), potentially leading to blindness. As a prevention method, retinographies are taken and analyzed by ophthalmologist doctors to evaluate whether a person suffers from this symptom [14]. Effective screening for diabetic retinopathy that is evaluated by ophthalmologists has been proven to reduce the risk of sight loss [1]. However, the assessment for retinopathy in our country is low (13.2% at primary health care, 11.5% at Family Health teams, 14.9% at Basic Health Care Units, and 35.9% at the tertiary health care unit) [13]. There is a significant shortfall of ophthalmologists in developing countries, and resources are limited [11].

In this context, machine learning offers a prime opportunity to perform detection on a large scale within a limited budget. A neural network model can make an initial analysis of retinographies taken by a technician. This process serves as a filter, thus reducing the number of ophthalmologists required [2], [3], [4], [5].

We perform this work in collaboration with Endocrinology Division of Hospital de Clínicas de Porto Alegre, as a part of the project "Diabetic retinopathy screening in patients with diabetes mellitus: validation of innovative method (machine learning). In this study, we show the results obtained with deep, convolutional neural networks using inception v3. We demonstrate that higher resolution increases the average neural network accuracy. We reach over 0.93 area under the curve (AUC) in the receiver operating characteristic (ROC) of our best model.

In Section II, we discuss the related work covering the used techniques and their application to diabetic retinopathy. In Section III, we show the results we obtain with our models while detailing the methodology and the effects observed with changes applied in images. In Section IV, we conclude this work and briefly discuss our future directions.

II. Background

Machine learning has gone through a revolution in the last decade. The increase in computational power has enabled artificial intelligence researchers to create much more complex models. Although Hornik et al. [6] formally proved that neural networks with a single hidden layer could be universal approximators, specialization of multiple layers has shown that these "approximators" can become accurate enough to perform human tasks.

One particularly well-known case is the deep convolutional neural network [9]. It emulates human eye behavior by filtering sub-areas of the eye through convolutions. That is, instead of connecting all layers of neurons as in a regular neural network, it constructs a layer by combining sectors of neurons from the previous layer, as seen in Figure 1. Each neuron observes only a portion of the image, much like in the human eye. Convolutions between different regions create the identification of shapes, defining features that will identify classes. Thus, deep CNNs have layers that automatically perform feature engineering. By combining multiple layers of convolution, max-pooling, rectified linear units, and fully connected layers, a convolutional neural network can accurately identify objects in an image [8].

Several works have improved the field of computer vision, with many focusing on the Imagenet dataset in the "Imagenet Large Scale Visual Recognition Challenge" (ILSVRC) [12]. The most popular and currently used architecture for a deep neural network is the Inception architecture [15]. The idea behind inception is to use features from multiple resolution levels in an efficient way, using a very deep neural network, and exploiting general-purpose graphic processing units (GPGPU) to train the neural network efficiently.

These ideas have recently been used to detect diabetic retinopathy in retinographies. In Gulshan et al.'s work [5],

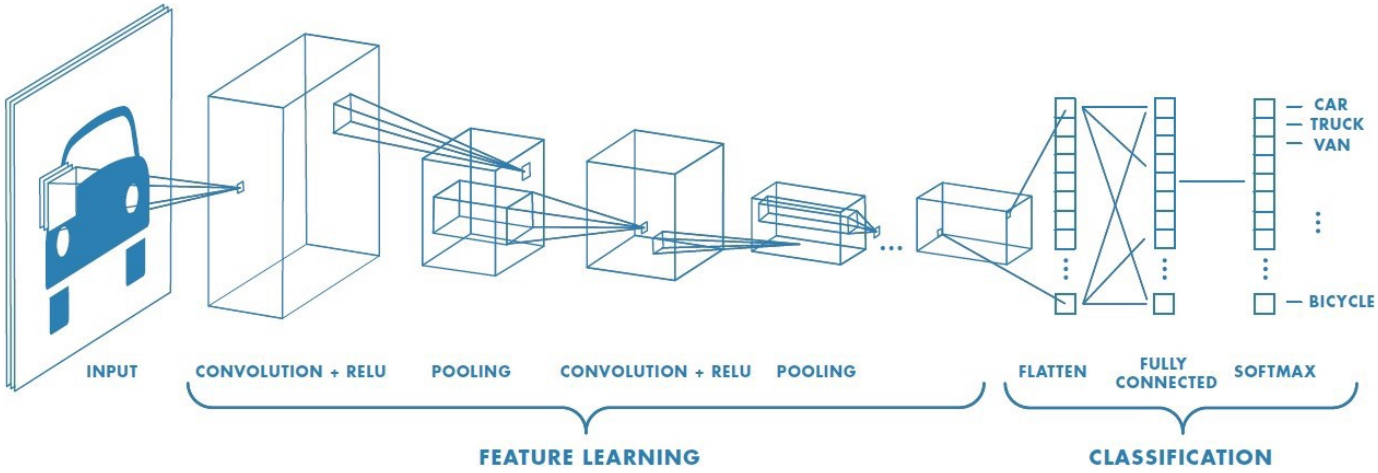


Fig. 1. Convolutional Neural Network abstraction. Source: https://www.mathworks.com/

a convolutional neural network-based in the Inception architecture version 3 is used. They feed the neural network with the data from three Indian hospitals (Aravind Eye Hospital, Sankara Nethralaya, and Narayana Nethralay) and the EyePACS dataset [2]. For validation of the network, they used additional EyePACS images and the Messidor-2 dataset [3]. An essential step in their procedure was to reevaluate all images with several graders. Their algorithm achieved an AUC of the ROC of 0.991 for EyePACS and 0.990 for Messidor-2. Thus by selecting a high sensitivity point, the algorithm achieves 97.5% sensitivity and a 93.4% specificity for the EyePACS dataset; and it achieves 96.1% sensitivity and 93.9% specificity for the Messidor-2 dataset.

These results are far superior to other works on the same task [10], [17], [19], [4]. A recent study by Voets et al. [18] attempted to reproduce these results with the EyePACS dataset available in the Kaggle competition for diabetic retinopathy [1], but it was not able to do so. After a significant change to the activation function of the final layer, the reproduction algorithm was able to achieve 0.94 AUC for EyePACS and 0.80 AUC for Messidor. The authors of the reproduction point four possible reasons for this difference; first, the reproduction had a single grade per image, whereas the original study had multiple grades as mentioned above. Second, the published list of hyper-parameters from the original study does not detail the normalization method and validation procedure used, so the reproduction algorithm is not able to achieve the same level of tuning like the original. Third, there might be errors in the original study or methodology. Fourth, the reproduction study has errors in its methodology. The study demonstrates the difficulty of reproducing a deep learning algorithm without an available model or code. The models are complex, and any lack of information will make the reproduction of a model incomplete.

Krause et al. [7] extended the original work. The focus of this study is to use adjudication to quantify errors in diabetic retinopathy grading based on individual graders and majority decision. They use the improved data to train a better neural network, using the Inception v4 architecture and increasing the resolution of the images for training. The work has found that out of the discrepancies between adjudication by retinal specialists and the majority decision of ophthalmologists, the most common were missing microaneurysms (36%), artifacts (20%), and misclassified hemorrhages (16%). They then compare the performance of the ophthalmologist majority decision (OMD) and the deep learning algorithm (DLA), using the adjudicated consensus of retinal specialists as the reference standard. For moderate or worse diabetic retinopathy, the OMD obtains 83.8% sensitivity and 98.1% specificity, whereas DLA obtains 97.1% sensitivity, 92.3% specificity, and AUC of 0.986. Thus they found that by using a small number of adjudicated consensus grades from retinal specialists as a tuning dataset and higher-resolution images as input, the algorithm improved in AUC from 0.934 to 0.986 for moderate or worse diabetic retinopathy detection.

In this study, we base our code on Voets' reproduction [18], but adapting ideas from the more recent work from Krause et al.' [7], such as higher resolution for input images.

III. Experiments

A. Algorithm Development

For an initial assessment of the viability of using machine learning as a screening tool, we trained ten neural networks based on the Inception v3 architecture. We used the same parameters available in Gulshan et al. 's work [5] and its reproduction [18]. Our initial codebase is the reproduction's codebase, available in https://github.com/mikevoets/jama16-retina-replication. All networks were initialized with imagenet weights for all layers except the fully connected layer on top, which received random weights. In Figure 2, the 42 layers and operation of the Inception v3 architecture are detailed.

As input, we used Kaggle's EyePACS dataset, redimensioning every picture to $299x299$ pixels. The dataset contains

[1] https://www.kaggle.com/c/diabetic-retinopathy-detection/data

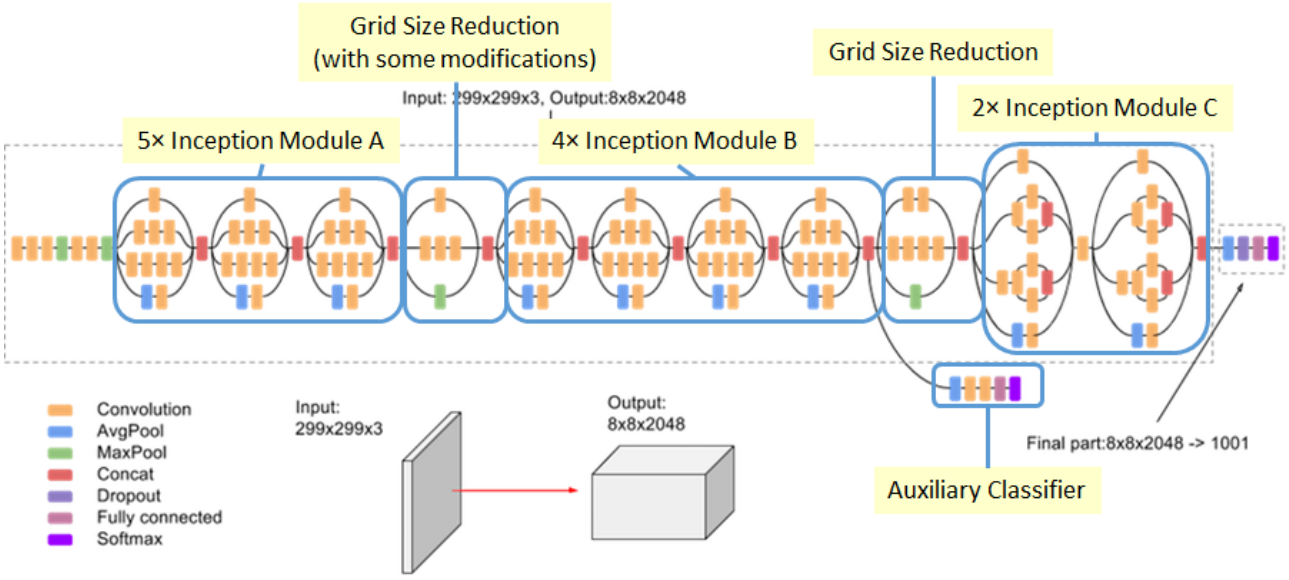


Fig. 2. Inception V3 architecture. Source: https://cloud.google.com/tpu/docs/images/inceptionv3onc–oview.png

$88,612$ images, where $65,343$ have no signs of disease; $6,205$ present mild retinopathy; $13,153$ present moderate retinopathy; $1,997$ present grave retinopathy; and $1,914$ present proliferative retinopathy. We split these images into binary training and testing datasets. Thus, we perform a binary classification: either the patient has mild or no retinopathy (class 0), or he presents moderate or worse retinopathy (class 1). We used the same data augmentation techniques to increase the set of conditions in the training dataset artificially: left to right flip, random saturation, random hue, random brightness, and random contrast. Currently, we only transform the images and do not add any new image.

The training dataset consists of $40,688$ images in class 0 and $16,458$ images in class 1. We use 80% of the training dataset to optimize the weights of the neural network and 20% of the dataset to tune hyperparameters, such as cross-entropy and AUC.

B. Receiver Operating Characteristic Curves of the Ensembles

To illustrate the performance of our models, we used receiver operating characteristic curves. These are graphical plots to illustrate the performance of a binary classification system given different discrimination thresholds. One axis represents the true positive rate, also known as sensitivity, while the other represents the false positive rate, also known as specificity. Sensitivity is a ratio between the number of images in class 1 the model correctly labeled as class 1 and all images in class 1. Specificity is a ratio between the number of images in class 0 the model correctly labeled as class 0 and all images in class 0. Each curve in our figures has 200 threshold points.

Figure 3 presents the receiver operating characteristic curve of the ensemble model composed of the ten neural networks.

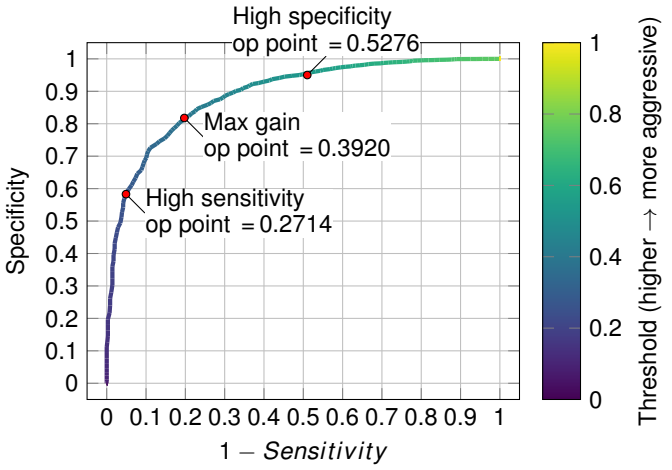


Fig. 3. Receiver Operating Characteristics of the ensemble model composed of ten neural networks with $299x299$ pixels input image resolution.

The three chosen points indicate a high specificity point, a maximum gain point, and a high sensitivity point. The high specificity point achieves 95.02% specificity, 51.05% sensitivity. The maximum gain point achieves 81.78% specificity, 80.26% sensitivity. The high sensitivity point achieves 58.30% specificity, 95.10% sensitivity. The AUC of this model's ROC is 0.89.

To improve upon this model, we have discussed with the endocrinologists from the collaborating hospital regarding the disease. We have found that the lesions and details of the disease can be represented in tiny segments of the image, which led us to increase the resolution of the input image, much like Krause et al.'s work.

Figure 4 presents the ROC curve of the ensemble model

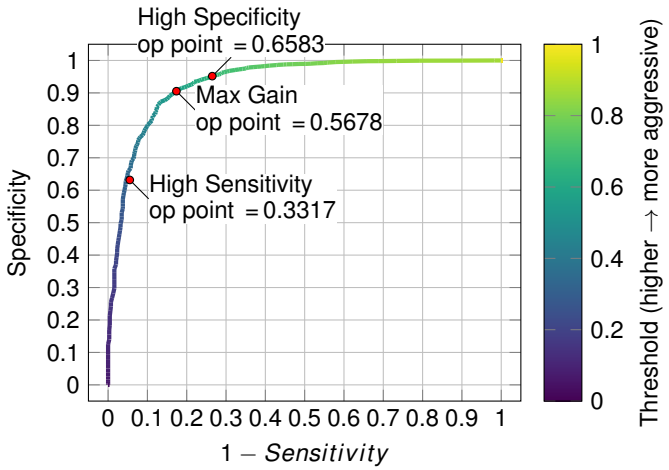


Fig. 4. Receiver Operating Characteristics of the ensemble model composed of ten neural networks with $500x500$ pixels input image resolution.

composed of the new ten neural networks. We can observe a drastic improvement, especially on the trade-off for a high sensitivity point, which now achieves over 60% accuracy. The high specificity point achieves 95.17% specificity, 73.49% sensitivity. The maximum gain point achieves 90.53% specificity, 82.85% sensitivity. The high sensitivity point achieves 64.46% specificity, 95.1% sensitivity. The AUC of this model's ROC is 0.93.

IV. Conclusions

In the healthcare context, sensitivity to screen a disease is the most critical metric of a detection mechanism. Detection of inexistent illnesses is not a serious problem when compared to an undetected illness that may require urgent treatment. However, if a predictive model aims to be efficient in reducing the load on ophthalmologists in public health, then the accuracy of the model represents how much it can save in a percentage. In this study, we have shown how a simple change to a neural network can impact its performance. Given the same sensitivity, higher resolution images provided an improvement of 6% in the accuracy of the detections. For the receiver operating characteristic curve, the area under the curve changed from 0.89 to 0.93.

We are currently working on improving the model. Our current idea is to use Inception v4, and obtain a specialized dataset to tune the model, as Krause et al. have done. Research on hyperparameters should also be considered, given the much higher area under the curve obtained in their research. Finally, the largest source of difference is the dataset available to us. Unfortunately, large datasets are not readily available. In the future, we will fuse smaller datasets to create a more extensive training dataset with varied camera configurations, which should help the network to generalize artifacts from different cameras better.

References

[1] A. D. Association et al. Standards of medical care in diabetes—2010. *Diabetes care*, 33(Supplement 1):S11–S61, 2010.

[2] J. Cuadros and G. Bresnick. Eyepacs: an adaptable telemedicine system for diabetic retinopathy screening. *Journal of diabetes science and technology*, 3(3):509–516, 2009.

[3] E. Decencière, X. Zhang, G. Cazuguel, B. Lay, B. Cochener, C. Trone, P. Gain, R. Ordonez, P. Massin, A. Erginay, et al. Feedback on a publicly distributed image database: the messidor database. *Image Analysis & Stereology*, 33(3):231–234, 2014.

[4] R. Gargeya and T. Leng. Automated identification of diabetic retinopathy using deep learning. *Ophthalmology*, 124(7):962–969, 2017.

[5] V. Gulshan, L. Peng, M. Coram, M. C. Stumpe, D. Wu, A. Narayanaswamy, S. Venugopalan, K. Widner, T. Madams, J. Cuadros, et al. Development and validation of a deep learning algorithm for detection of diabetic retinopathy in retinal fundus photographs. *Jama*, 316(22):2402–2410, 2016.

[6] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

[7] J. Krause, V. Gulshan, E. Rahimy, P. Karth, K. Widner, G. S. Corrado, L. Peng, and D. R. Webster. Grader variability and the importance of reference standards for evaluating machine learning models for diabetic retinopathy. *Ophthalmology*, 125(8):1264–1272, 2018.

[8] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[9] Y. LeCun, Y. Bengio, et al. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995, 1995.

[10] H. Pratt, F. Coenen, D. M. Broadbent, S. P. Harding, and Y. Zheng. Convolutional neural networks for diabetic retinopathy. *Procedia Computer Science*, 90:200–205, 2016.

[11] S. Resnikoff, W. Felch, T.-M. Gauthier, and B. Spivey. The number of ophthalmologists in practice and training worldwide: a growing gap despite more than 200 000 practitioners. *British Journal of Ophthalmology*, 96(6):783–787, 2012.

[12] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[13] J. Schneiders, G. H. Telo, L. G. Bottino, B. Pasinato, J. L. Neyeloff, and B. D. Schaan. Quality indicators in type 2 diabetes patient care: analysis per care-complexity level. *Diabetology & metabolic syndrome*, 11(1):34, 2019.

[14] I. M. Stratton, A. I. Adler, H. A. W. Neil, D. R. Matthews, S. E. Manley, C. A. Cull, D. Hadden, R. C. Turner, and R. R. Holman. Association of glycaemia with macrovascular and microvascular complications of type 2 diabetes (ukpds 35): prospective observational study. *Bmj*, 321(7258):405–412, 2000.

[15] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[16] G. H. Telo, F. V. Cureau, M. S. de Souza, T. S. Andrade, F. Copês, and B. D. Schaan. Prevalence of diabetes in brazil over time: a systematic review with meta-analysis. *Diabetology & metabolic syndrome*, 8(1):65, 2016.

[17] A. Tufail, C. Rudisill, C. Egan, V. V. Kapetanakis, S. Salas-Vega, C. G. Owen, A. Lee, V. Louw, J. Anderson, G. Liew, et al. Automated diabetic retinopathy image assessment software: diagnostic accuracy and cost-effectiveness compared with human graders. *Ophthalmology*, 124(3):343–351, 2017.

[18] M. Voets, K. Møllersen, and L. A. Bongo. Replication study: Development and validation of deep learning algorithm for detection of diabetic retinopathy in retinal fundus photographs. *arXiv preprint arXiv:1803.04337*, 2018.

[19] T. Y. Wong and N. M. Bressler. Artificial intelligence with deep learning technology looks into diabetic retinopathy screening. *Jama*, 316(22):2366–2367, 2016.

List of Authors