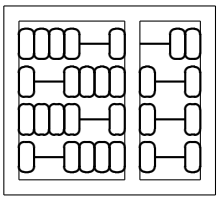
UNIVERSITY OF CAMPINAS

INSTITUTE OF COMPUTING



Dwsr: Predicting the memory usage of tensorial algorithms

Student: Daniel De Lucca Fonseca

Advisor: Prof. Edson Borin

Abstract: Lorem ipsum dolor

Contents

1 Introduction

Sit amet lorem

2 Background

Sit lorem dolor

3 Related Work

Lorem ipsum

4 Research proposal

In this research proposal, I aim to predict computer memory consumption from an algorithmic perspective. As seen in section 3, while there have been numerous studies on historical analysis of memory usage to predict resource requirements from the scheduler perspective, there is a significant gap in research when it comes to predicting memory usage of a single specific algorithm.

4.1 Problem statement

The Discovery [1] laboratory, located at UNICAMP [2], is working on a seismic analysis project with Petrobras [3]. This project relies heavily on machine learning and seismic operators for its computing graphs. However, the input of these graphs is a massive seismic dataset that can contain terabytes of data. Even supercomputers do not have enough memory to handle the computation on a single node. Therefore, usually the execution is distributed by using data parallelism.

To facilitate this process, our laboratory implemented a framework called DASF [1] that simplifies the development of data-parallel computing graphs using DASK [2] under the hood. DASF [1] has an input parameter called **block_size** that defines the size of each seismic block used during data parallelism. However, setting this parameter can be challenging because it required finding the optimal relationship between it and the network overhead caused by it.

[1] `https://discovery.ic.unicamp.br/`

[2] `https://ic.unicamp.br/`

[3] `https://petrobras.com.br/`

To illustrate this challenge, I present image **??**sizefig:block_sizeh contains three computing graphs receiving input data from a seismic dataset. The first graph shows the input data as the entire dataset, which requires a significant amount of memory to execute. The second graph divides the data into thousands of small parts, reducing the memory requirement but adding network overhead. The third graph divides the data into a smaller number of parts, minimizing both network and memory requirements.

While executing the graph, the developer must manually set the block_size parameter. Setting a large number may lead to memory issues and it causes a signifcant delay due the trial-and-error nature of the execution flow. Since Petrobras uses supercomputers to execute those graphs, this delay is even larger considering the time it takes to submit a job due to the queue waiting time.

On the other hand, setting a small number may increase the execution time due to network overhead. Since Petrobras have a large number of graphs to execute, and each graph usually takes a long time to execute, I need to find a way to optimize the block_size parameter.

DASK [2] provides an automatic chunking feature, but it relies on the chunk_size parameter, which is a static parameter define prior to execution. Since the developer do need to define this parameter prior to the execution, they can't rely on DASK [2] auto chunking feature to automatically split the data, but they can rely on the automatic chunking if they figure out the ideal chunk_size prior to the execution.

Figuring out the chunk_size for algorithms that doesn't require a large working memory is easy, since the developer can set that to a percentagem of the available memory. But, some of seismic operators used by Petrobras genereates a large working memory during the graph execution, which makes it difficult to determine the ideal chunk_size.

Based on this assumption, if someone predicts the memory usage of the graph that person can use the DASK [2] auto chunking feature to automatically split the data into the ideal number of chunks. Therefore, this research aims to develop a memory usage prediction model that can help us determine the optimal block_size parameter during execution. This will help Petrobras to optimize resource utilization and minimize waiting and execution time. The model will provide a comprehensive understanding of memory usage patterns for different block sizes and contribute to the development of more efficient resource allocation strategies in large-scale clusters.

4.2 Proposed solution

I see two possible solutions to optimize the block_size parameter: (i) I can create a model that would predict the ideal block_size of algorithm based on the input data or (ii) I can create a model that would predict the memory usage of an algorithm based on the input data, and use it to determine the optimal block_size.

Although appproach (i) is more straightforward, it is limited only to the block_size. Approach (ii), on the other hand, is broader and can be used not only to identify the block_size, but also as an heuristic for other tasks like scheduling, resource allocation, and cloud cost estimates.

Based on this, I have decided to pursue approach (ii) and create a machine learning model that would predict the memory usage of an algorithm based on the input data. While there are already alternatives available to predict memory consumption from the scheduler's perspective, no research has been conducted to predict memory consumption from an algorithmic perspective.

The proposed solution is divided into two strategies:

4.2.1 Strategy one: Algorithm-specific model

The first and primary goal of this research is creating a machine learning model that can predict the memory consumption of a given algorithm given the input data. For this strategy, I will train a model for each algorithm, considering the seismic data's shape and size as the primary features for the prediction. Since seismic dataset are usually normalized, I may also extract other features to improve the prediction, but such features are going to be defined during the exploration phase. I will start the exploration with polynomial model, since I belive it is suitable for the prediction, and the model's output would help me to determine the optimal block_size for each part of the graph.

This approach's primary limitation is that it requires a new model for each seismic operator, but it allows us to predict the output size and shape, enabling us to compose a graph prediction.

4.2.2 Strategy two: Generic algorithm model

The secondary goal is to develop a model that is algorithm-agnostic. If the results of the first approach are optimistic, I may extract features from the source code itself to predict memory consumption accurately. I believe that the algorithm could extract relevant information, such as how the code author deals with memory management. Although I do not have a clear picture of the features that I could extract, I believe that this is a possible experiment for this research.

However, this objective is secondary and will be pursued only if the results of the first approach are positive.

4.3 Potential risks and limitations

In this section, I will discuss the limitations of the proposed solution for predicting memory consumption of seismic algorithms using machine learning.

4.3.1 Sensibility towards too many control statements

A possible limitation for this solution is that it may not work if the algorithm has too many control statements. Depending on the structure of the algorithm, the amount of control statements may change the memory usage.

This limitation can be ignored if the memory allocation of the algorithm happens before the control statements. On this scenario, even with control statements that change drastically the execution of the algorithm the memory usage will not change.

Either way, all the seismic operators and machine learning models being used are tensorial algorithms, which means that they do not have too many control statements. Therefore, this possible limitation will not be a problem during the research.

4.3.2 Unpredictable bottlenecks

There are two possible memory bottlenecks in the proposed solution: CPU and GPU memory. Seismic operators are likely to be GPU-memory bounded, but I need to conduct experiments to verify this.

If the algorithms can be both GPU-memory and CPU-memory bounded, then the proposed solution must be able to handle both scenarios.

4.3.3 Language-agnosticit

y

The proposed solution is currently focused on using Python since it is the language used for the seismic operators and DASK [2] itself.

Although Python is a popular language in the scientific community, it may not be the language of choice for all researchers. This solution may not be language-agnostic at the moment. However, in the future, I can improve the solution to accept algorithms from any language.

In the case of strategy one, presented on section 4.2.1, I can improve the training structure to accept algorithms from any language. This can be done by allowing decoupling both the feature extraction, as well as the execution of the algorithm, from the training process.

In the case of strategy two, presented on section 4.2.2, it is possible improve the feature-extraction part to allow more languages as well. Since the only difference between the two strategies is allowing to exract features from the source code, a specialized feature extractor per-language should be enough.

4.4 Problems to be addressed

In this section, I will outline the potential problems that I may encounter during our research and how I plan to address them. Each problem is going to be presented as a subsection, with a brief description of the problem and how I plan to address it.

4.4.1 How to measure memory usage

One of the primary challenges in our research is accurately measuring memory usage. This is particularly true for GPU memory, which may differ from CPU memory. I aim to explore different options to measure memory usage accurately.

I plan to start by investigating if there is an API to measure GPU memory, and if not, I will explore common libraries that allow gathering memory consumption based on a specific process. I understand that the optimal approach would be to use a specific API for this, since this would allow more flexibility for our tool, but I will explore alternative options if necessary.

4.4.2 Historical data requirements

Many of the current approaches require a significant amount of historical data to train the models. This is not feasible for our research, as it would be time-consuming to generate such data for algorithm-specific models. I aim to overcome this limitation by creating a proper discovery approach by merging reinforcement learning with Bayesian analysis.

I plan to gather the minimal amount of data, even by executing the algorithm locally with a small amount of synthetic data, and then cover the exploration space for the input data for shapes and data with features the model have never seen before. With this approach, I think the model will be able to learn while it is being used, and it will be able to generalize to new data.

4.4.3 Python's garbage collection

Python's garbage collector can pose a problem for us. Python uses reference-counting as its garbage collection strategy, and it is lazy, so it usually waits for the memory to be needed to clean. If our operators are CPU-memory-bounded, I may need to figure out how to deal with Python's garbage collector to gather the real memory usage. However, I will only need to address this issue depending on the results of our experiments.

4.4.4 Graph execution

Another problem I may encounter is how to figure out the entire graph's memory requirements. While our proposed solution can help us find the amount of memory required for a specific algorithm, integrating multiple algorithms into a graph poses a challenge. I plan to address this issue by predicting not only the memory usage, but alos the output shape of the algorithm and its features. By having prior knowledge of the algorithms in the graph, I can compose a graph with the models using the output of the first model as the input data of the second one.

4.5 Research questions

In this section, I present the research questions I aim to answer in this study. The research questions are divided into different categories, including: (i) feasibility, (ii) accuracy, (iii) and applicability. Table 1 summarizes the research questions and their respective categories.

Table 1: Research questions

#	**Question**	**Category**
RQ1	What is the optimal way of gathering memory-usage data?	Feasibility
RQ2	Are seismic tensorial algorithms memory bounded by the CPU or GPU?	Feasibility
RQ3	Which features do we need to extract from the input data?	Feasibility
RQ4	What is the memory-usage behavior of our algorithms?	accuracy
RQ5	Under extreme circumstances, how is the memory usage of our algorithms?	accuracy
RQ6	How to integrate a reinforcement learning model with Bayesian analysis?	Applicability

4.6 Research methodology

In this section, I will describe the methodology I will follow to answer the proposed research. Each research questions is going be answered by a set of experiments. To organize the execution of the experiments, I will use the categories presented on section 4.5.

4.6.1 Feasibility

This experiment category aims to answer **RQ1**, **RQ2**, and **RQ3**. The goal is to understand the feasibility of our seismic operators and their main characteristics. I plan to start with a single experiment to figure out the proper way to gather memory usage metrics from both the CPU and GPU (RQ1). After this, I aim to have an experiment that will explore if our seismic operators are GPU-memory bounded or CPU-memory bounded (RQ2). Lastly, I will explore multiple executions of our seismic operators, trying to understand possible features we can extract from the inputs, not only the shape, but also other possible features that we can extract from the data (RQ3).

4.6.2 Accuracy

This experiment cateogry aims to answer **RQ4** and **RQ5**. The goal is to understand how reliable the results can be. I will first explore the behavior of the seismic operators, exploring how they use memory in a set of different synthetic executions (RQ4). Finally, I will try pushing our algorithms to the limit to check if, under extreme circumstances (like uncommon inputs), the algorithm can break or display an unpredictable memory-usage pattern (RQ5).

4.6.3 Applicability

The third, and last, experiment category is applicability, which aims to answer RQ6. The goal is to act as a pre-prototype experiment. I aim to explore reinforcement learning models and Bayesian analysis to understand how I can apply the results from the previous experiments to predict memory usage.

4.6.4 Prototype development

After we execute all the experiments, I will implement a prototype. The idea for that prototype is to act as a plugin for DASF [1] and use it as a contribution to the active Petrobras seismic project

in our laboratory. The plugin can act not only to tune the input block_size for every operator, but also as a decision heuristic for scheduling.

4.6.5 Research extension

Based on the results of all past experiments, I can focus on the second proposed solution. This part aims to explore the possibility of generalizing the developed machine learning model to be algorithm-agnostic. To implement this, I need to figure out how and which features to extract from the original source code. If this is possible, I can develop a generic model that can be used to predict memory usage for any seismic operator.

References

[1] J. Faracco, O. Napoli, and E. Borin, *Dasf is an accelerated and scalable framework*, `https://github.com/discovery-unicamp/dasf-core`, 2023.

[2] M. e. a. Rocklin, *Dask*, `https://github.com/dask/dask`, 2023.