PROPOSED APPROACH TO INCREASE EFFICIENCY OF POLYHEDRAL COMPILATION USING HASH MAPPING AND PARALLEL SYSTEMS

Hira Beenish and Muhammad Fahad
Pakistan Air Force, Karachi Institute of Economics and Technology, Karachi, Pakistan
E-mail: hira@pafkiet.edu.pk, mfahad@pafkiet.edu.pk

***ABSTRACT**: The polyhedral model is considered as a strong and powerful platform for efficient and optimizing programs. The polyhedral model is totally based on the algebraic representation of a program and consider as production and mature compilers. This compiler is a valid and good way to express the loop structure applied in run time compilation and it can be suitable for loop-based programs. To enhance the algebraic progression and resolving complexities, we are proposing two solutions that can enhance the speeding and memory efficiency during the compilation process which includes hash mapping and parallel systems. Moreover, we are comparing the different applications of the polyhedral compilers and detecting the lacking in performance which includes TIRAMISU and Alpha Z.*

Keywords: Polyhedral, Hash Function, Parallel Systems, GPU, CPU, Algebraic Expressions

1. INTRODUCTION

The polyhedral model is a vast and powerful platform or framework where it gives values to optimize and variation on parallel programs. Polyhedral compilation possesses the compilation process that depends on the representation of parallel and optimizing programs, which involves nested loops and arrays. In initial stages, it was composed as parallel compiler but it is now used for a wide range of applications that requires optimization, including automatic parallelization, data locality optimizations, memory managing situations, verification of programs, communication optimizations, SIMDization, code generation for hardware accelerators, high-level synthesis, etc. [3] It is an algebraic based representation of programs that allows to construct and search for complex sequences of optimizing programs. Furthermore, this model is now efficient and can reach production compilers [9].

The primary constraint of the polyhedral model is known to be its effective transformation of the loop traversing to statically, loop-based program parts. There has been involved in utilizing such procedures in compilers, in short time compilers, just as DSL compilers. The polyhedral compiler has a strong scope in maintaining the technicalities of handling the loop constraints while the compilation of each line of code happens, yet increasingly more industrial clients begin to adjust such advancements that have become so ease of usage [7].

A Polyhedral framework is designed to maintain and generate efficient performance code. This can be an advantage of using multicore programs and multiprogramming functions. This framework will be helpful for areas on linear algebraic, image processing, deep learning, and many more platforms. Compilers dependent on the Polyhedral model including ongoing applications like PoCC or CHiLL target code parts that precisely fit the relative constraints of the model [2].

The explanation for this restriction isn't that precise testing is required to make use of the polyhedral model, yet rather that there is no major plan to help dynamic control flow on compile-time in the compilation of changing factors and calculations. To battle a typical misconception, the intensity of the polyhedral model isn't to accomplish complicated information and testing but also analyses the array access functions [8].

According to the structure of the polyhedral model, the compiler creates two nested loops which bound to represent the dimension integral point. The second step is to schedule the affine changes producing efficient code for lite frameworks is ending up increasingly more difficult as these models are expanding in intricacy and decent variety. Getting the best execution requires complex code and information format changes, the board of complex memory chains of importance, and efficient information correspondence and synchronization. For instance, consider summed up framework increase (gemm), which figures C = αAB + βC and is a structure square of various calculations [4].

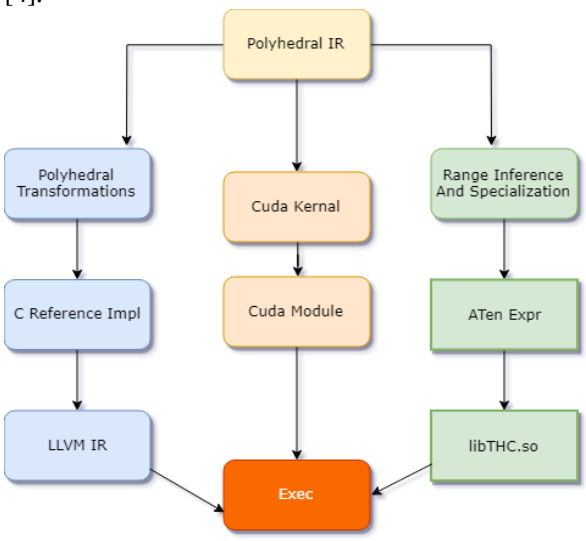

Fig. 1 Taxonomy of Polyhyderal Model.

2. Literature Review

There are multiple researchers which were revolving towards the polyhedral model, its advantages, its efficiency and the applications on which it works as a compiler. The overall sum up of the model is to generate and maintain an efficient code that carries the performance for highly efficient programs. The basic reviews on this model are listed below:

The polyhedral model is described as these compilers dependent on the variant of the language structure including ongoing examination apparatuses target code parts that precisely fit the relative imperatives of the model. It also fits the logarithmic and algebraic tasks [1]

The polyhedral structure is used to make efficient and correct calculations on the coding structures, where we use nested loops or loops to make such combinations to fulfill the user's output and optimizing the result in such a manner that it can generate efficiency. Where it allows constructing the sequence of a complex structure which defines the maturity of the model [2]

In this paper, this model's structure is based on the algebraic representation of moderate programs. The restrictions that are rotating towards this model are the static predictability of loop and nested loop performance, which makes the data allow to flow in control. Through this structure, it can be applicable in many variant situations to execute high performance [3]

The polyhedral model supports an application CUDA for generating parallel code. It affiliates with different sourcing compilers and extracting the parallel data using memory and polyhedral compilation. During the working on this application, the polyhedral model compiles with two different compiles like PoCC and CHiLL, which performs like loop transformations similar to this model. The output was to take out the dynamic data and its dependency [4]

In this paper, the polyhedral model is combined with deep learning techniques. During the optimizations of machine learning and deep learning, they need to run their training phase while instructing new architecture, there comes a challenge that how to compile the techniques of learning. To resolve the compilation problem, the polyhedral compiler played a role in the maintenance and it worked with deep learning kernels to speed up the loop transforming situations [5].

The polyhedral model has been effectively utilized in the making of compilers. Late proposition researched how runtime data could be utilized to apply the polyhedral model on applications that don't statically at the model. They proposed a dynamic investigation that manufactures a reduced polyhedral model from a program execution that can precisely identify conditions and fixed memory of programs [6, 7].

In this paper, the polyhedral model is implemented as an online auto-tuning. For a high-class auto tuner, it should have high dimensional speed, should be more efficient, can carry different variants of outputs and can possess a multitasking program process. To fulfill the demands of this auto tuner, the polyhedral model played a role in making a technique to solve the tuning problems, which can improve the speed, efficiency, and multitasking of the code [8].

In this paper, the author implemented an offline clustering statement using the source to source a polyhedral compiler. Modification of PPCG implemented in this work by performing offline clustering before affine scheduling and after analysis on dependency using python wrapper around isl. Different benchmarks are used to implement it namely image processing kernels, polytechnic-3AC, swim benchmarks, and Dist kernel [10].

Following the advantages of the polyhedral model, a new application is fused with this structure named TIRAMISU which is a scheduling language that defines the complexities of efficient and multitasking programs. This application working is based on linear algebra and in the area of image processing. This application is structured with 3 different layers with this model as data layout, loop optimization, and communication.

3. Proposed Methodology

In the previous workings of the polyhedral model, we have categorized different polyhedral compilers and compare their workings as comparing the different components and qualities of those compilers. These include the recent one compilers which are TIRAMISU and AlphaZ. They have the most qualities which fulfill the scope of the polyhedral model.

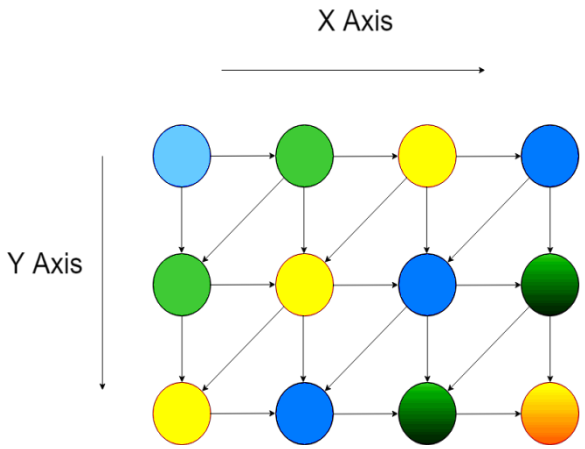

Fig. 2 Rotation of loop and working loop in compiler.

Before proposing our method, we are discussing some polyhedral compilers which have the applied implemented model and are in the working process. Those compilers can make some variations but in order to achieve the creditability of the efficient compiler, we need to compare the models according to their structure, their hype, and their computing data analysis.

3.1 Tiramisu

Tiramisu is a recent platform for a polyhedral compiler that is based on expressing and organizing fast, portable and efficient computations according to data layers. Moreover, Tiramisu offers C++ API integrations which allow reducing the consistency of repetition in the code compilation process especially in nested loops, where the iterations go in infinite series. Tiramisu can be used in different structured component areas such as linear and deep learning, image processing and machine learning. It is purely based on the polyhedral model which is distributed as a data layer for loop processing. It is designed to enable easy integration of code generators for new architectures.

3.2 AlphaZ

AlphaZ is a general structure for analysis, change, and code structure in the Polyhedral Model. The input structure of the polyhedral model comprises at least one numerical conditions that indicate exactly what should be figured. It tends to be seen as a determination. So as to create an

(ordinary/basic) program that needs to require this particular structure, one needs to indicate a time frame. Moreover, the distribution of processor based on memory consumption and memory designation. As a matter of fact, even this isn't carefully vital. We likewise have a "memorized request-driven" code generator that produces executable code without any time or memory/processor portion data.

According to our research in different research projects, [9] the pure essence of this model is to have onetime compilation of loops that can make the code efficient to use, Moreover, the running time of the code which increase the performance of CPU is depend on the size of memory accessible to the section, which can include the synchronization of the register memory and main memory. The below-mentioned table describes the comparison of the different polyhedral compilers:

TABLE 1: COMPARISION OF DIFFERENT POLYHEDRAL COMPILERS

Compiling Features	*Tiramisu*	*Alpha Z*	*Pencil*	*Pluto*	*Halide*
CPU Generation	*Yes*	*Yes*	*Yes*	*No*	*Yes*
GPU Generation	*Yes*	*Yes*	*No*	*Yes*	*No*
Affine Loop	*Yes*	*Yes*	*Yes*	*Yes*	*Yes*
Data Access	*Yes*	*No*	*Yes*	*Yes*	*Yes*
Optimizing Data	*Yes*	*Yes*	*Yes*	*Yes*	*Yes*
Memory Hierarchies	*No*	*Yes*	*No*	*No*	*No*
Data Flow Graphs	*Yes*	*No*	*Yes*	*Yes*	*Yes*
Iteration Spaces	*Yes*	*Yes*	*Yes*	*Yes*	*Yes*
Dependent Analysis	*Yes*	*Yes*	*No*	*Yes*	*No*
Compilation Time	*No*	*No*	*Yes*	*Yes*	*No*
Commands Comm.	*Yes*	*No*	*Yes*	*No*	*No*

The above mentioned table has two ideal polyhedral compilers, which are good at the mentioned qualities. What we found was the runtime and memory consumption problem which has found in lacking towards both compilers alternately i.e. TIRAMISU has compiled time error and AlphaZ has memory consumption error. The fact is that those two features can vary the compilation results of the code which can put up different input stats and getting nonfeasible outputs. Like if a loop is being run at some limit of time we can have the fix results, but for the infinite loop, the memory space and runtime of the compilers will hang.

3.3 Pencil

PENCIL is a characterized subset of GNU C99, and authorizes a lot of coding rules mainly identified with confining the way in which pointers can be controlled. The guidelines are intended to empower a compiler to perform better improvement and parallelization when making a structure of PENCIL to a lower-level transformation, for example, OpenCL. Moreover, PENCIL is likewise furnished with explicit language builds, including expecting precedence and calculations for capacities that empower correspondence of area explicit data to the PENCIL compiler, to be utilized for enhancement. Since it depends on C, the expectation to absorb information for PENCIL is quite complicated.

If we look at the architecture of a nested loop, the conditions which will be furnished to get outputs will be first starting with one iteration of x and then the required iterations of the y if the condition meets or not. Due to the possible and close iterations which should meet the conditions can be out of run time, which means the running time of the program will stick at one point and will consume a large amount of memory. In order to overcome the problem of high-efficiency redundancy, we can have two possible solutions which can be possible or maybe one of the efficient solutions to gain memory efficiency and speed.

The first proposed solution for maintaining efficiency is to make a parallel way to check the required memory and required runtime for getting maximum inputs. Due to maximum loop checks, it can be possible to make the parallel distance of memory and speed using the polyhedral model architecture. The required automating strategy is to make the check from first having the total sets of loop checks and then optimizing the parallel process of the code.

3.4 Process

- The model can be started by checking the cache.
- Excess amount of memory can be restored once the memory prediction completes according to the compiling code.
- The storage according to the flowchart, can be maintained by one file system which can create new dependencies and the external software distinguish the memory with or without the proper statement.

3.5 Advantages

- The main interest of this modification is to manipulate and optimize the speed of the compilation process.
- It will depend on the number of the items and elements which is present in the sets of nested loop. The items will be placed through x and y states.
- The process can be utilized in data localization, program validity and completion of a data structure according to the nested loop.
- Genetic and compact solutions through a nested loop can be advanced in polyhedral models as well.

Another way to have some possible solution is to work on cache memory using a hash map and functions. Hash functions and mapping can play a vital role in order to get the functioning results and minimizing the search according to the needs. A Cache is quick to support, that either attempts to gain access times to slower memory or to maintain a strategic distance from costly recalculations of possible outputs. Generally, the size of a cache is full when it gets the possible loop elements. Since our cache isn't restricted in size, our reserving calculation cache memory works out in a hash map. So as to do this, a hash capacity should be characterized. Besides the capacity of an equivalent is expected to check for the equity of two components.

A hash capacity maps estimations of info set onto fixed-size whole number qualities. An equivalent capacity accepts two

qualities as information and returns genuine if these qualities are viewed as equivalent and false generally. So as to look for a component in the hash map, it is adequate to scan for it in the impact rundown of its hash map. Moreover, in this way to restrict the hunt to all

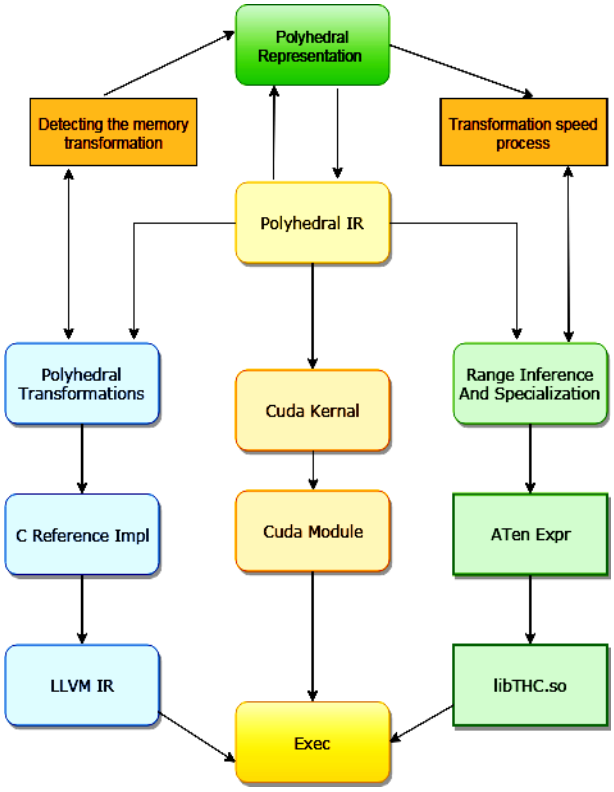

Fig.3. Modified proposed taxonomy for parallel process

stored components that have a similar hash function.

In order to make some general computations through the hashing functions, our caching algorithm is mapped through hash mapping. Furthermore, a structure of equations is derived to check the consistency of the elements that get solved through the hashing functions. A hashing map sets a value which considered as input towards the integer values and during computation, the compilation process gets an output. It makes a decision of what the fact is true or false due to the collision of hash values then to limit the searching of items if the cache elements have the same hash values.

These two possible solutions can make a major role in making the consistency and efficiency of the code. During the compilation process of the nested loop, it can make some possible outcomes while compiling the sets of processes. Using parallel ways, the process can be divided into two separate processes in which the memory assumption gaining on the process of the nested loop can full fill the architecture of the polyhedral model. Although we have the overall comparison of all polyhedral compilers which are flexible, although they have some flaws, they are running.

The polyhedral compilation is suitable and more interesting gaining when it uses in manipulation and optimization in algorithms because when the loop compiles, it depends on its iterating structure and complexions and then the output of the loop performance gets varied over the algorithm. Furthermore, generic and compact solutions through this process can be designed which will be based on polyhedral model techniques. The structure has some essence of acyclic flow that handles loop as a whole technicality of the loop elements. Also, compared to optimizations that can handle complex loops, the polyhedral model techniques work on the specification of looping scales, instead of having some iteration issues that might be taking time in resolving the factors.

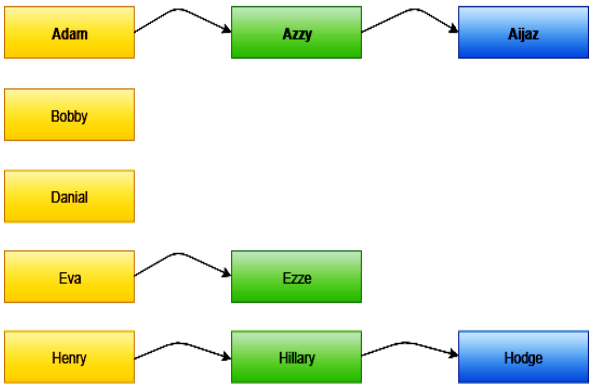

Fig. 4. Structure of Hash mapping

The applications that can make adjustments in the polyhedral structures gets somehow difficult to overcome all flaws which can affect the structure and efficiency of the compilers and its process. As an example, AlphaZ does not intend to take care of this issue, despite the fact that now and again we have utilized the framework to understand explicit cases of explicit advancement issues. Or maybe, AlphaZ gives a structure wherein such issues can be understood by various clients for various focuses with various target capacities to enhance. This is regularly the quick objectives of the individual structure at explicit occasions, however, the objective of the AlphaZ framework is to set the memory allocating structure for this exploration. It is consequently a framework wherein to investigate advancement techniques, regularly manual however in the long run programmed, to accomplish higher execution. Specifically, AlphaZ opens the capacity to adjust memory allotment and treats decreases.

4. CONCLUSION

This paper described the classic functions of the Polyhedral model and its applications, that how a polyhedral framework works with multiple functionalities and scheduling languages. Moreover, this paper describes the core features of the compilation of nested looping and indexing and getting the set inputs and outputs. The polyhedral model structure is described in which the looping and iterating process is discussed and works in an efficient process of compilation with TIRAMISU and Alpha Z.

Comparisons of the different polyhedral models following compilers were happening in which we found the major error of timing and speed due to the high compiling process and access to getting most outputs. The data layout is introduced in the shape of a parallel process in which the compilation steps go in parallel. Moreover, we also proposed hash mapping and hash functions that can support

the polyhedral model architecture with parallel systems. It checks the required memory with needed runtime for getting maximum input, all the steps are working parallel to each other to increase the efficiency in the polyhedral compiler.

REFERENCES

1. Benabderrahmane, M. W., Pouchet, L. N., Cohen, A., & Bastoul, C. The polyhedral model is more widely applicable than you think. In International Conference on Compiler Construction (pp. 283-303) (2010)
2. Verdoolaege, S., Carlos Juega, J., Cohen, A., Ignacio Gomez, J., Tenllado, C., & Catthoor, F. Polyhedral parallel code generation for CUDA. ACM Transactions on Architecture and Code Optimization (TACO), 9(4), 54 (2013).
3. Kobeissi, S., & Clauss, P. The Polyhedral Model Beyond Loops Recursion Optimization and Parallelization Through Polyhedral Modeling. (2019).
4. Vaidya, H., Badrinaaraayanan, A., Patwardhan, A. A., & Upadrasta, R. When Polyhedral Optimizations Meet Deep Learning Kernels (2019).
5. Sato, Y., Yuki, T., & Endo, T. An autotuning framework for scalable execution of tiled code via iterative polyhedral compilation. ACM Transactions on Architecture and Code Optimization (TACO), 15(4), 67 (2019).
6. Gruber, F., Selva, M., Sampaio, D., Guillon, C., Pouchet, L. N., & Rastello, F. Building of a Polyhedral Representation from an Instrumented Execution: Making Dynamic Analyses of non-Affine Programs Scalable (2019).
7. Mohammadi, M. S., Yuki, T., Cheshmi, K., Davis, E. C., Hall, M., Dehnavi, M. M., ... & Strout, M. M. Sparse computation data dependence simplification for efficient compiler-generated inspectors. In Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation (pp. 594-609) (2019).
8. Pfaffe, P., Grosser, T., & Tillmann, M. Efficient hierarchical online-autotuning: a case study on polyhedral accelerator mapping. In Proceedings of the ACM International Conference on Supercomputing (pp. 354-366) (2019).
9. Baghdadi, R., Ray, J., Romdhane, M. B., Del Sozzo, E., Akkas, A., Zhang, Y., ... & Amarasinghe, S. Tiramisu: A polyhedral compiler for expressing fast and portable code. In Proceedings of the 2019 IEEE/ACM International Symposium on Code Generation and Optimization (pp. 193-205) (2019).
10. Baghdadi, Riyadh, and Albert Cohen. "Scalable Polyhedral Compilation, Syntax vs. Semantics: 1–0 in the First Round." (2020).