International Journal of Research in Engineering, Science and Management
Volume-2, Issue-5, May-2019
www.ijresm.com | ISSN (Online): 2581-5792

An Optimized Scheduler and Compiler for Image Processing

N. Shashank[1], M. Kunal[2], U. Puneet[3], A. Parkavi[4]
[1,2,3]*Student, Department of CSE, M. S. Ramaiah Institute of Technology, Bengaluru, India*
[4]*Professor, Department of CSE, M. S. Ramaiah Institute of Technology, Bengaluru, India*

***Abstract*: This paper presents an idea that will help in the design of a compiler for domain specific approaches that will help in the conversion of the image processing codes that have been written in a dynamic language into a highly optimized and specific code. We will be borrowing some of the concepts from the halide language and the sa-c language to help in the construction of a highly efficient and optimized compiler to help in the image processing approaches. One of the approaches that we will be taking is that of automatic scheduling of the pipelines involved in the image processing approach and combining with the concepts from building an optimized compiler for dynamic languages. By using the concepts and understanding the image processing pipelines and approaches we will be able to develop a new and improved compiler based on the techniques known and by using some predicting techniques.**

***Keywords*: Compiler, Image Processing, Image processing pipelines, Scheduler, FPGA, DDL, Halide**

1. Introduction

Compiler form a very important part of every major system today and cannot function without one. Compilers will help in the conversion of the high level languages into assembly code used by the machines. We use compilers all the time and without one the system will not be able to process our language into language that the system will understand. There are different layers to the compilers lexical analyzer, syntactic, semantic, code generator and code optimizer. We will try to exploit the code optimizer to help us achieve our goals. We will now try to shed some light on the image processing approaches that is very widely used and important. Image processing by digital means is the ability to process images using some form of computerized approaches. The following steps can be used:

1. Importing images by using some tools
2. Analysis and modification of the images
3. Output of the modified image.

Now we are going to combine the concepts of image processing and compiler design and build a better and more powerful specialized compiler for image processing. We will be shedding some light on the new and improved languages and the specialized compilers based on the new language that can help speed up the image processing speed and improve accuracy by a large scale. We will be going in depth about the various languages that were designed for better improving the image processing architecture and we will be talking about the optimization of the compilers for these languages. The optimized compilers will help to speed up the process. We will be talking about the various languages like the SA-C and HALIDE and also we will be talking about FPGA and External and Shallow/Deep Embedding's in the image processing setup. First we will be talking about image processing pipelines and the efficient scheduling of it for the improved processing and about a new and improved halide compiler that can help optimize the compiler design. We will also be talking about External and Shallow/Deep Embedding's in the image processing where we are using domain specific languages and we will be comparing them. The engineering cost of shallow embedding is cheap, and that of deep embedding's is expensive. We here in this paper are trying to build a compiler which is optimized so that it can work more efficiently. We will be building a system that will run a scheduling system along with an optimized compiler approach.

2. Literature survey

The first paper talks about the introduction of a new and improved compiler for domain specific languages. The specialized compiler written in python and is more efficient then the modern compilers the are present in the market now to do image processing. The compiler exploits the fundamental approaches and ideas involved in the image processing approaches and help in building the compiler based on that.[1]. The second paper talks compiling of image processing in reconfigurable hardware and is developed in the sa c language for image processing and talks about the language and the optimization approaches to the compiler design. It talks about the data flow diagram and the abstract approaches to the design. [2]. The third paper talks about optimizing and compiling of the image processing approaches in FPGA. It talks about FPGA and the methods in which the compiler can exploit the usage of this for image processing. It also talks about the sac languages and the benefits of the language and the optimized compiler for that language [3]. The fourth paper talks about the automatic scheduling of the image processing pipelines in the halide languages and tells us that by using this automatic approaches the system can perform better than when an expert writes the codes. It talks about the various approaches that the system will

take to help in the optimization of the compiler so as to get a faster execution [4]. The fifth paper talks about the different languages and the embedding whether it is a shallow embedding or if it is a deep embedding and it gives a comparison of the two [5].

3. Methodology

We will be talking about the method in which we will be able to develop the most optimized and efficient compiler by combing the concepts of automatic scheduling approaches. However efficient the codes may be the scheduling of the pipelines in the image processing approaches is very necessary then only will the compiler be performing up to the maximum and deliver the desired results. This compiler will be exploiting the three conditions

1. A lot of the data structures that are involved in the image processing are arrays of small sizes so if we have the size information about the arrays and the different data structures the we will be able to help in a lot of the optimization processes, such as stack allocation and unfolding.
2. A lot of the processes in the image processing approaches can be parallelized and a lot of time have no data dependency to each other therefore can be used for multiple thread approaches which will help in the optimization and will reduce the execution time by a major factor. Therefore, all processes that do not have any form of data dependency can be run at the same time.
3. Whenever humans execute the coding in image processing a few errors are not really recognized by humans due to human nature and the capacity of our eyes, like the error in color of an image of about 0.001% is not noticed. So we do not really require such high precision as our eyes will not be able to make out the difference so we can make a tradeoff between precision and efficiency in speed.

For all these compiler designs we will be having two parts to it the optimization phase in the code such as removing or unrolling the loops or the design using languages which will allow single assignment so that there is no confusion for the compiler and can run smoothly and also the another main component of the imag3 processing the scheduling of the different pipelines of the approach. By combing the methodology of having a highly optimized compiler to optimize the code as well as have a proper scheduling methodology to help support it will greatly increase the efficiency of the compiler for image processing. In our system we will be having the three phases of the image processing compilers they are the

1. Analysis phase that will analyze the requirements
2. Program transformations
3. Automatic scheduling

A. Analysis phase

We will be talking about the analysis phase and the methods in which we can help in the optimization of the compiler, we will be doing three main things in the analysis part they are First we will analyze the type of input and the relative sizes of the different data types involved in it. By getting to know about the different type of data we will be able to combine them and compile portions of the program to an efficient code. By also knowing the size of the different data types we should be able allocate the space required more efficiently.

Second step is the analysis to find out the parallelism. We should be able to determine the extent of dependence between the different parts of the code so as to efficiently parallelize the code to run on different threads.

Third step is the allocation of certain arrays such as the temporary and output buffers. We also develop a method in which we can reduce the precision and increase efficiency of the code as the human eye will not be able to differentiate between the highly precise images. In the next phase we will be doing some form of transformations such as

Rewriting the api calls so that it will be calling a more efficient and customized functions that will work better. The system also does some loop changes such as unrolling the loops so that it is easier for the compiler to access the information. It also can use a single assignment language or the prevention of the use of the control stack that will help in the smooth running of the compilers. The compiler does not need to store much information if it is in such a way that will help in the optimizing of the compilers.

More optimization is also achieved by the use of some ai principles. There are certain parameters on which optimization is done and a good guess of the initial value to these parameters is chosen called the heuristic, then by using hill climb racing approaches the system will be able to get to a better estimate and optimize the code from there.

All of these will help in the optimizing of the compiler and the smooth and efficient running of the code, to help improve the efficiency of the system we will be trying to also involve the scheduling approaches to the image processing pipelines to help in the optimization of the code.it is the job of the automatic scheduler to help in choosing the different tasks to be completed in the and to perform the most critical tasks. All the scheduling should be done on the basis of producer consumer dependencies and should strictly follow the rules of data dependency. With the help of the automatic scheduler the memory usage is also reduced as compared to one where there is no proper scheduling. Consider the blur function in image processing in which without the scheduling the intermediate buffer is big and the memory has to store in the main memory and the access time will be large, so by doing some proper scheduling a small portion of the image can be processed at a time and therefore the output and data can be stored in the cache therefore the access time will increase. In this way the automatic scheduling will help in the reduction of memory as well as in the increase of execution time. The method by which the automatic scheduling will take place is by estimating the cost, for all the given inputs and outputs. Deciding which stages to interleave

for better data locality. Picking the data in such a way so as to improve locality, and mainly to improve stability to execute the code in a parallel fashion.

B. Future works

There is a possibility to improve the working of the proposed compiler in this paper. By the better understanding of the image processing approaches and by using better pipelining and the use of artificial intelligence approaches to the image processing approaches the task of compiling the code can be improved.

4. Discussions and conclusion

We have in this paper we have tried to create a new and improved compiler based on the various image processing approaches that are available. We were able to combine the automatic scheduling concepts and the code optimization techniques that are designed for image processing. Image processing is a very large and in demand field in the present age, with companies spending a considerable amount of resources into image processing approaches. With the help of this new and improved compiler designed the code for the image processing will be greatly reduced and it will become more optimized and efficiency will improve The automatic scheduling approach will be beneficial to the image processing methods as it will provide easier and more efficient to the image processing methodology.

References

[1] Yuting Yang, Sam Prestwood, Connelly Barnes, “VizGen: Accelerating Visual Computing Prototypes in Dynamic Languages.”

[2] Ravi Teja Mullapudi, Andrew Adams, Dillon Sharlet, Jonathan Ragan Kelley, and Kayvon Fatahalian, “Automatically Scheduling Halide Image Processing Pipelines,” in ACM Transactions on Graphics, 35(4), July 2016.

[3] B. Draper *et al*., "Compiling and optimizing image processing algorithms for FPGAs," *Proceedings Fifth IEEE International Workshop on Computer Architectures for Machine Perception*, Padova, Italy, 2000, pp. 222-231.

[4] Robert Stewart and Heriot-Watt. An Image Processing Language: External and Shallow/Deep Embedding, RWDSL '16 Proceedings of the 1st International Workshop on Real World Domain Specific Languages, 2016.

[5] R. Rinker, J. Hammes, W. A. Najjar, W. Bohm and B. Draper, "Compiling image processing applications to reconfigurable hardware," *Proceedings IEEE International Conference on Application-Specific Systems, Architectures, and Processors*, Boston, MA, USA, 2000, pp. 56-65.

[6] R. Membarth, O. Reiche, F. Hannig, J. Teich, M. Körner and W. Eckert, "HIPAcc: A Domain-Specific Language and Compiler for Image Processing," in *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 1, pp. 210-224, 1 Jan. 2016.

[7] Jonathan Ragan-Kelly, Connelly Barnes and Andrew Adams, Halide: A Language and Compiler for Optimizing Parallelism, Locality and Recomputation in Image Processing Pipelines,” PLDI '13 Proceedings of the 34th ACM SIGPLAN Conference on Programming Language Design and Implementation, pp. 519-530, 2013.

[8] N. Chugh, V. Vasista, S. Purini and U. Bondhugula, "A DSL compiler for accelerating image processing pipelines on FPGAs," *2016 International Conference on Parallel Architecture and Compilation Techniques (PACT)*, Haifa, 2016, pp. 327-338.

[9] James Hegarty, John Brunhaver, Zachary DeVito, Jonathan Ragan-Kelley, Noy Cohen Steven Bell, Artem Vasilye, Mark Horowitz, Pat Hanrahan, “Darkroom: CompilingHigh-Level Image Processing Code into Hardware Pipelines,” Proceedings of SIGGRAPH 2014.

[10] Pierre Guillou, Benoît Pin, Fabien Coelho and François Irigoin, “A Dynamic to Static DSL Compiler for Image Processing Applications,” 2016.

[11] Hammes J.P., Draper B.A., Willem Böhm A.P. (1999) Sassy: A Language and Optimizing Compiler for Image Processing on Reconfigurable Computing Systems. In: Computer Vision Systems. ICVS 1999. Lecture Notes in Computer Science, vol. 1542. Springer, Berlin, Heidelberg.