HiPC SRS 2015, Bengaluru, India

SWIFT-A Fast Enhanced String Matching Algorithm for Heterogeneous Architectures

Sourabh S. Shenoy, Supriya Nayak U., Neelima B.
NMAM Institute of Technology, Nitte

ABSTRACT

This paper presents a modified version of quick search and brute force string searching algorithms, named as SWIFT. SWIFT algorithms are executed on dynamic parallelism enabled Kepler Graphics Processing Unit (GPU) from Nvidia. They perform significantly better when the pattern size is large.

ACKNOWLEDGEMENT

We would like to thank GPU Centre of Excellence (GCOE) at Indian Institute of Technology (IIT)-Bombay for giving GPU based server access to carry out our work. We also thank NVIDIA for providing us with the GPU hardware under the GPU Education Centre.

CONTACT

Mr. Sourabh S. Shenoy
NMAM Institute of Technology, Nitte
sourabhsshenoy@gmail.com

INTRODUCTION

String searching algorithms [1] play a key role in various applications designed to overcome real world problems like plagiarism checks, intrusion detection etc.

GPU, unlike the traditional CPU, has a hierarchy of memory and can manage over thousands of threads. These, if efficiently utilized can parallelize and speed up the task, thus reducing the overall time of execution.

DYNAMIC PARALLELISM

- One of the key features of Kepler GPUs from Nvidia wherein a GPU thread can launch a set of threads without any involvement of the CPU.
- Use of this feature always leads to enhanced performance provided the amount of work done by child kernel is not nullified by the overhead involved in calling the child kernel by parent kernel [2].

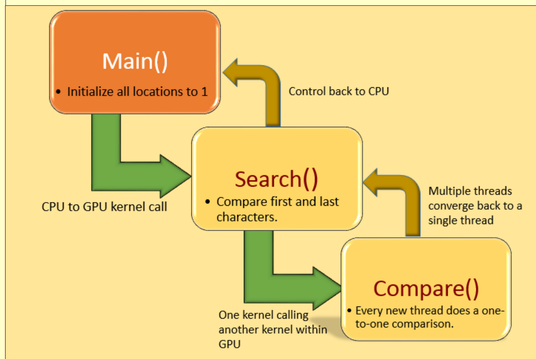

Fig 1. Dynamic Parallelism with respect to SWIFT

SWIFT

- Most of the existing string matching algorithms do not scale well for large pattern sizes.
- The authors propose a modified algorithm, called SWIFT based on dynamic parallelism.
- SWIFT algorithms perform poorly for small patterns, but improvement in their performance for large patterns is noteworthy.

STEPS OF WORKING

1. All the elements of array Arr[ ] of size 'n' initialized to 1.Each element corresponds to one text sub-string.
2. Letters on the extreme ends of both text sub-string and the given pattern compared.
3. On a match, a set of threads is spawned for one-to-one comparisons between the remaining letters in the sub-string and pattern.
4. On mismatch, corresponding array element set to 0.

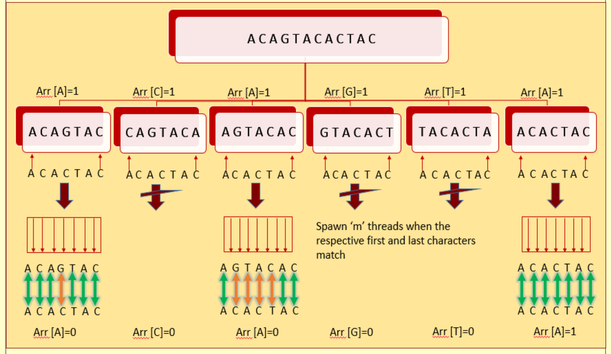

Fig 2. Working of SWIFT algorithms for text A C A G T A C A C T A C and pattern A C A C T A C

Algorithm for Quick Search with Dynamic Parallelism

```
//Host code
Main ():
Pre-process-pattern ()
Calculate-Shifts-Text ()
Call Search Kernel
Initialize all locations to 1. (Found)
//Kernel code
Search ():
Check if first and last characters match.
Yes: Call Compare Kernel with m (Pattern Length) threads
//Dynamically spawn threads.
No: Set corresponding location value to 0. (Not found)

Compare ():
Each thread checks for one locations. If found
Yes: Do nothing
No: Set corresponding location value to 0. (Not found)
```

Fig 3. Pseudo code for SWIFT Quick Search

PERFORMANCE

The experiments were performed using benchmark data such as "The Project Gutenberg Edition of THE WORLD FACTBOOK 1992, "Bible", "E.coli genome structure".
The brute force regular and shared memory codes and quick search code are inspired from Raymond Tay's implementation [3].

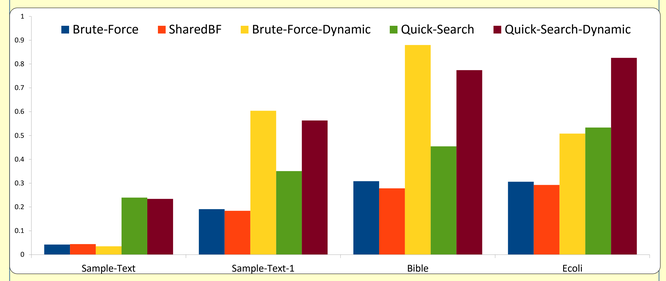

Fig 4.Performance for pattern length of 500

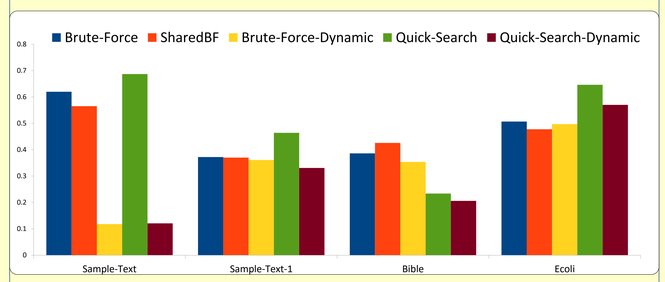

Fig 5.Performance for pattern length 1500

CHALLENGES

1. Not all string searching algorithms can adapt to Dynamic parallelism.
2. Incompatibility of Dynamic parallelism with shared memory of GPU.

CONCLUSION AND FUTURE WORK

The amount of work done by the child kernel outdoes the overhead involved in calling it. Speedup obtained is in the range of 1.2X to 5X for various patterns and datasets.
As a part of our future work, SWIFT will be expanded to various other important algorithms.

REFERENCES

[1] C. Charras, T. Lecroq, 'Exact String Matching Algorithms', www-igm.univ-mlv.fr

[2] J. Wang and Sudhakar Y., Characterization and analysis of dynamic parallelism in unstructured GPU applications, International Symposium on Workload Characterization, pp.51-60, (2014).

[3] Raymond T., A demonstration of Exact String Matching Algorithms in CUDA, code.google.com