GPU-Based Parallel Finite State Machines

Kameron W. Kincade Ryan P. Langewisch
Colorado School of Mines
kkincade@mines.edu — rlangewi@mines.edu

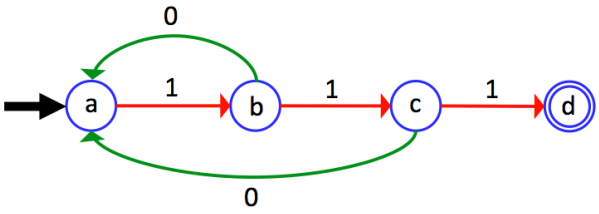
Abstract

Abstract goes here.

General Terms finite state machines, GPU, parallelization

1. Introduction

A finite state machine (FSM) is used to represent an object that can exist in a limited number of states and that can change state based on a number of well-defined rules (Figure 1). Each FSM is responsible for processing various inputs (each usually represented as a string of characters) and determining if these inputs satisfy the machine's acceptance criteria, such as ending in a correct final state. To operate on an input string, the FSM begins with the start state (A in Figure 1) and processes each character one after another. The state machine's rules, along with the input character, dictate the next location to move to. An acceptance state (D) is commonly indicated by a double circle and represents a valid input solution that satisfies the FSM's criteria.



FSMs are embedded in many different types of applications, including web application computations, such as lexing, parsing, translations, and media encoding/decoding. With the recent emphasis on mobile, handheld devices, the need for a computationally efficient implementation of FSMs is becoming more important. Parallelization is one of the obvious ways to speed up the execution of a program. However, the simple structure and procedures of a FSM also make them difficult to parallelize. The most evident way to parallelize a FSM is to divide the input string into a number of different FSM input strings, known as substrings, each of which can be processed in parallel. The problem with this approach is determining the start state for each new substring. The start state of one substring should be the end state of the previous substring, but due to the nature of FSMs this can't be calculated in parallel. One approach is to make an educated guess for the start state. This paper describes a proposed method to make this educated guess, along with a parallelization technique to utilize the processing power provided by GPUs.

CONF '14, September 29, 2014, 2014, Golden, CO, USA.

http://dx.doi.org/10.1145/nnnnnnn.nnnnnnn

2. Methods

In efforts to correctly guess the start state of a substring, a technique known as lookback is utilized. Lookback involves analyzing a certain number of input characters prior to the start of the particular substring. The advantage of lookback lies in the context provided by analyzing the previous input characters. This context might eliminate a number of states from being possible start states due to the nature and structure of the FSM. As a result, this increases the chances of choosing the correct start state for a given substring. A couple subjects of research include how many substrings to divide the original input string into, how many lookback characters to analyze, and how the number of available processors influences the execution time of a given parallelized FSM.

Beyond the concept of lookback, additional steps can be taken to try and decrease the execution time of a parallelized FSM. To fully utilize the number of cores on a GPU, a parallel FSM implementation can use an enumerative approach when it is unsure of the start state for a particular substring. The enumerative approach begins calculating all possible paths for the FSM. Once a correct start state is found, the correct path from the multiple, enumerated paths is used as part of the solution for the state machine. To make the enumerative approach more efficient, when a start state is correctly identified, all unfinished subprocesses using incorrect start states are terminated to make room for more subprocesses to begin.

Lots of text.
More text.
Lots of text.
More text.
Lots of text.
More text.
Lots of text.
More text.
Lots of text.
More text.
Lots of text.
More text.
Lots of text.
More text.
Lots of text.
More text.
Lots of text.
More text.
Lots of text.
More text.
Lots of text.
More text.
Lots of text.
More text.

Lots of text.
More text.
Lots of text.
More text.
Lots of text.
More text.
Lots of text.
More text.
Lots of text.
More text.
Lots of text.
More text.
Lots of text.
More text.
Lots of text.
More text.
Lots of text.
More text.
Lots of text.
More text.
Lots of text.
More text.
Lots of text.
More text.
Lots of text.
More text.
Lots of text.
More text.
Lots of text.
More text.
Lots of text.
More text.
Lots of text.
More text.
Lots of text.
Lots of text.
More text.
Lots of text.
More text.
Lots of text.
More text.
Lots of text.
More text.
Lots of text.
More text.
Lots of text.
More text.
Lots of text.
More text.
Lots of text.
More text.
Lots of text.
More text.
Lots of text.
More text.
Lots of text.
More text.
Lots of text.
More text.
Lots of text.

Acknowledgments

Acknowledgments, if needed.

References

[1] P. Q. Smith, and X. Y. Jones. ...reference text...

A. Appendix Title

This is the text of the appendix, if you need one.