Speculative Parallelization Needs Rigor

—Probabilistic Analysis for Optimal Speculation of Finite-State Machine Applications

Zhijia Zhao, Bo Wu and Xipeng Shen
Computer Science Department
College of William and Mary, VA, USA
{zzhao,bwu,xshen}@cs.wm.edu

ABSTRACT
Software speculative parallelization has shown effectiveness in parallelizing certain applications. Prior techniques have mainly relied on simple exploitation of heuristics for speculation. In this work, we introduce probabilistic analysis into the design of speculation schemes. In particular, by tackling applications that are based on Finite State Machine (FSM) which have the most prevalent dependences among all programs, we show that the obstacles for effective speculation can be much better handled with rigor. We develop a probabilistic model to formulate the relations between speculative executions and the properties of the target computation and inputs. Based on the formulation, we propose two model-based speculation schemes that automatically customize themselves with the best configurations for a given FSM and its inputs. The new technique produces substantial speedup over the state of the art.

1. INTRODUCTION
Parallelization is key to computing efficiency and scalability. Many programs, however, are hard to parallelize for their data dependences. A typical solution is to circumvent the dependences through speculation. Prior speculative parallelizations have mainly relied on simple use of heuristics. In this work, we explore rigorous analysis for tapping into the full potential.

Particularly, we concentrate on applications that are based on Finite-State Machine (FSM)—when it comes to the prevalence of dependences, FSM is truly unbeatable. An example is the FSM in Figure 1 for matching regular expression ([01][00][01][11][01]*])10. Dependences exist between every two steps of an FSM computation. They effectively chain through the computation, making parallelization difficult.

Enabling FSM parallelization has some immediate value, given that FSM forms the core of many important applications, including lexing in web browsers, intrusion detection in networking security, Huffman decoding in multimedia processing and string pattern matching in document processing. Further, it will shed insights on circumvention of dependences in general, benefiting other parallelizations.

State of the Art.
The key difficulty for parallelizing FSM applications is to determine the starting state for each thread. Researchers have attempted to circumvent the difficulty by speculation—letting a thread Ti guess the correct starting state to process segment Si. A random guess is subject to large errors. Researchers have found it helpful to do a lookback [3, 1]—that is, thread Ti runs the FSM on a number of ending symbols (called a suffix) of the preceding segment Si−1, and uses the ending state as the speculated starting state for processing its own segment Si. The lookback helps speculation by offering some context. For instance, for the FSM in Figure 1, if the suffix of Si−1 is "11", a lookback will always finish at an even-numbered state. Hence, the lookback successfully prevents the speculation from taking those impossible states.

However, performance benefits from lookback is inconsistent as the "heuristic" bars in Figure 2 shows. The main reason is the lack of rigor in the designs of speculation, reflected in multiple dimensions. The first is in the length of the suffix to examine by a lookback. A longer suffix exposes more context, but at the same time incurs more overhead. The second dimension is in the selection of the state for starting a lookback. The third dimension is in the usage of lookback results. All prior studies used the initial state as the lookback starting state, the ending state of a lookback as the speculated state for processing the next segment, and empirically chose the lookback length. Are these always the best choices? Without rigorous ways to treat these factors and dimensions in the design, existing parallelizations are vulnerable to FSM complexities, yielding the inconsistent speedups.

2. OPTIMAL FSM SPECULATION
We present the first principled understanding of FSM parallelization so that the parallelization can best match with the properties of an FSM and its inputs. We start with understanding lookback, the key operation for speculation accuracy.

2.1 Essence of Lookback
Lookback is essentially a process that tries to use the context (i.e., a suffix) to improve the knowledge about the chance for a state to be the correct state at a speculation point. For convenience, we introduce the term feasibility as the probability for state s to be the correct state at a speculation point.

Without consideration of contexts, statistically, the feasibility of a state s at every speculation point is the same,
approximately equaling the frequency for the FSM to visit that state in normal executions. We call these probabilities context-free feasibilities. Accordingly, the feasibilities after an l-long input string are called conditional feasibilities.

A straightforward estimation of the conditional feasibility is infeasible due to the exponential space of value. To circumvent this difficulty, we observe that: State transitions essentially lead to an incremental propagation of conditional feasibilities, with contexts enriched gradually. According to this observation, we design a probabilistic model to derive each state’s feasibility during lookback. Refer [4] for details.

2.2 Model-Based Speculative Parallelization

Based on conditional feasibility, we create a “Makespan Formula”, which computes the statistical expectation of the performance of a lookback-based speculative parallel execution of an FSM. It provides the foundation for choosing appropriate parameters in designing speculative parallelization schemes.

With the Makespan formula, we develop two novel, model-based speculation schemes: All-State Lookback and Single-State Lookback. In all-state lookback, the lookback uses the complete state set as the starting state set—that is, during the lookback, each thread other than the first processes a suffix for |S| times (S for state set), each time starting with a different state. The single-state lookback starts lookback from a carefully chosen, single state.

Facilitated by the Makespan Formula, our design automatically finds the parameter values so that the two models can best meet the probabilistic properties of the FSM and inputs, hence find the optimal design parameters. These parameters include lookback length, the state to start a lookback, and the best way to use the lookback results for speculative parallelization. We prove the optimality through a series of rigorous inference [4].

3. EVALUATION

The implementations of the two speculation schemes both consist of a profiler and a controller. The controller runs online. By feeding information collected by the profilers to the analytic models described in the previous section, it configures the speculation schemes on the fly to suite the properties of the FSM and inputs. The profiler can run either online or offline. The online profiler can switch back to heuristic-based parallelization when the overhead is larger than a threshold.

To make the model-based speculative schemes easy to use, we develop a library named OptSpec which integrates the all-state and single-state speculative schemes and the online and offline profiling procedures together.

We evaluate the proposed techniques on seven FSM applications, with different complexities, including XML processing (lexing and xval), mathematics (div), Huffman decoding (huff), and string pattern matching (str1, pval, and str2). Figure 2 shows that our techniques (single-state online/offline, all-state online/offline) boost the performance by more than a factor of 4 over the state of the art (heuristic 1 and 2 are two versions of current techniques using simple lookback for speculation [3]).

4. CONCLUSION

This paper introduces formal analysis into the design of speculative parallelization. It reveals the essential connections between the design of speculation schemes and the characteristics of FSM and their inputs. It deepens the understanding to speculative execution of FSM computations with a series of theoretical findings. It provides two model-based speculation schemes, with suitable configurations automatically determined. Experiments show that the new techniques outperform the state of the art substantially. The findings may apply to applications beyond FSM, prompting more studies in shifting speculative parallelization to a rigorous paradigm.

5. REFERENCES