TPAoPI:A Thread Partitioning Approach Based on Procedure Importance in Speculative Multithreading

Yuxiang Li
School of Information Engineering,
Henan University of Science and Technology
Henan Joint International Research Laboratory
of Cyberspace Security Applications
Luoyang, China
Email: liyuxiang@haust.edu.cn

Danmei Niu
School of Information Engineering,
Henan University of Science and Technology
Henan Joint International Research Laboratory
of Cyberspace Security Applications
Luoyang, China
Email: niudanmei@163.com

Zhiyong Zhang
School of Information Engineering,
Henan University of Science and Technology
Henan Joint International Research Laboratory
of Cyberspace Security Applications
Luoyang, China
Email: xidianzzy@126.com

Lili Zhang
School of Information Engineering,
Henan University of Science and Technology
Henan Joint International Research Laboratory
of Cyberspace Security Applications
Luoyang, China
Email: lillyzh@126.com

***Abstract*—Thread partitioning is a core part of thread-level speculation (TLS) to achieve parallelization of irregular serial programs. The existing thread partitioning methods mostly adopt a unified partitioning scheme for all procedures in the same program, so that some procedures cannot obtain their best partition. This papers proposes a thread partitioning approach based on procedure importance (TPAoPI), which chooses to start with importance of procedures in irregular programs, adopting interdisciplinary research methods, creatively introducing the theory of importance in reliability theory, and calculates importance of procedures based on the characteristics of procedures. On the basis of the initial partition scheme, a manual intervention scheme suitable for the procedure is developed according to the importance of procedure, and a new performance evaluation model is used to theoretically evaluate obtained speedups, so that the best partitioning schemes of procedures are identified, exploring the intrinsic law that characteristics of procedures influence their best partitioning schemes. The paper does research from three aspects: calculating importance of procedures, generating the best partitioning schemes of procedures, and building a performance evaluation model, in order to fully exploit the potential parallelism for irregular programs. This paper provides a method for the research and applications of irregular programs parallelization and multi-core parallel computing.Experimental results show TPAoPI delivers an average 20.59% performance improvement than machine learning(ML)-based thread partition approach.**

***Keywords*-thread partition approach, thread-level speculation, adaptive, expert knowledge**

I. Introduction

The existing thread approaches are mainly divided into: thread partition approach based on heuristic rules [1], thread partition approach based on machine learning [2], graph-based thread partition approach [3] and so on. During the thread partition, the thread partition scheme is determined according to the heuristic rule, and the insertion positions of the thread partition statements are determined along the path of the program execution flow; Literature [4] used the minimum cut algorithm of the graph to partition program flow graphs, and uses heuristics to balance the cost of data dependence, performance cost, load imbalance and other factors. Literature [5] used heuristic rules to determine the granularity, priority, etc. of each thread after thread partitioning on the critical path of program execution.

In recent years, the development of machine learning has also shown strong predictive power in thread partitioning. Literature [6] used the clustering method to search for effective thread solution space to obtain a better thread partitioning. Literature [7] proposed a KNN-based thread partitioning method. The method mainly consisted of two parts: generation of the training sample set and extraction of knowledge contained in the sample set, and the k-most similar samples are selected by the similarity between each unknown program and the sample to determine the thread partition scheme of the program. Literature [8] used a machine learning method to partition stream programs on a mobile and automated compiler, learned prior knowledge offline and predicted the partitioning structure of unknown programs.

Literature [2] proposed a graph-based thread partitioning method, in which WCFG was used to formalize the expression of irregular programs, and machine learning methods

were used to learn thread partitioning knowledge and predict a thread partitioning scheme for unknown programs. In [9], the graphs are divided into sparse irregular data. A multi-thread graph divider mt-Metis was proposed, and 20 different graphs in multiple fields were used on 36 cores to verify the effectiveness of the method.

However, the above four types of thread partition methods regarded one program as the basic partitioning unit. When partition one program, these three types of thread partition methods adopted a unified partition scheme for all procedures of one program, leading to ignore the difference of the procedure, and some procedures cannot obtain their best partition. So, it is urgent to carry our the relevant research to obtain the optimal partition schemes for all procedures in programs.Based on the initial partitioning scheme of the machine-based thread partitioning method, this paper uses the interdisciplinary research method to creatively bring the importance theory (Importance Theory) in the reliability theory into the thread partition method. According to the characteristics of the procedures, its importance is calculated, and the manual intervention is used to explore the implicit partition based on the explicit partition method (manually intervening the partitioning flags), and the performance evaluation model is used to evaluate the acceleration ratio. To find out the optimal partitioning scheme of the procedure, it is verified that the effect of explicit parallelization is always better than the theory of implicit parallelism.

The remaining parts of this paper are organized as follows:in section II, we first briefly describe the SpMT execution model; the overall research framework is present in section III; in section IV, motivation of TPAoPI is presented; section V presents the implementation of TPAoPI; conclusion and future work are shown in section VI; finally section VII presents acknowledgement.

II. Description of Execution Model

TLS parallelizes serial programs and performs parallel execution on multi-core platforms to improve speedup performance. Fig.1 presents the TLS execution model [2], [10], in which the spawning point (SP) and the control quasi-independent point (CQIP) instructions map the serial programs into multithreaded programs. According to serial semantics, there is only one thread that allows data to be submitted to memory at each moment. This thread is called a definite thread, and the other threads are regarded as speculative threads. Every speculative thread consists of two parts, including precomputation-slice (p-slice) [11] and serial program code. P-slice is a small piece of code that is generated by the compiler based on slicing techniques to predict the live-ins used in speculative threads (a set of variables to be referenced before the value is defined).

Fig.1 shows four cases of SpMT execution. In Fig.1(a), it is assumed that a multithreaded program is equivalent to a serial execution program because it ignores SP-CQIP. Fig.1(b) shows the successful speculative execution: when the thread T1 encounters sp, if idle cores exist, the new speculative thread T2 is spawned; otherwise, the T2 is not spawned. When T1 encounters CQIP, it will validate the live-ins used by T2 in p-slice. If the validation is correct, T1 submits the execution results and releases the core resource. Then, the execution permission is transferred from T1 to the successor thread of T1; Fig.1(c) presents the state that validation of T2 fails, and speculative execution fails so to withdraw T2, and p-slice is not performed; Fig.1(d) illustrates the situation of restarting the thread in the current state when read-after-write (RAW) violation happens.

III. Overall Framework

This topic takes the irregular serial programs that traditional parallelization methods are hard to parallelize as the object. According to the importance of the program′s procedures, the importance calculation model is constructed, and its importance is calculated according to the procedure′s characteristics. Based on the initial partition of the procedures and artificial intervention, according to the importance of procedures, the partitioning flags (sp-cqip) are manually adjusted. In accordance with the performance evaluation results, the optimal partition of procedure is finally determined, thus forming a sequential chain:"procedure′s characteristics -> procedure importance -> manual intervention -> performance evaluation -> best partition". The sequential chain realizes the overall research goal that the optimal partitions are obtained for irregular programs, and the programs′ parallelism is fully exploited. The specific research ideas and research contents are respectively shown in Fig.2, Fig.3.

The research framework to be adopted is shown in Fig.3. The research framework takes the irregular serial programs as the input, and establishes the program complexity calculation model, the generation of candidate thread partition scheme, and the expert knowledge-based partition scheme selection as the main research points, and selects the most suitable thread partitioning scheme to execute the thread partition. The results are run on Prophet simulator to obtain speedups and programs′ results.

IV. Motivation of TPAoPI

Based on the importance calculation of program′s procedure and manual intervention, this paper explores the thread partition method for irregular programs , aiming at maximizing the parallelism of programs and providing necessary to make full use of the legacy serial programs on multi-core environment. The specific three research subgoals are:

- Design of Performance Evaluation Model
  According to the characteristics of the thread partition method of this subject, a performance evaluation model is designed to theoretically evaluate the speedup performance and make up for the defects that the existing

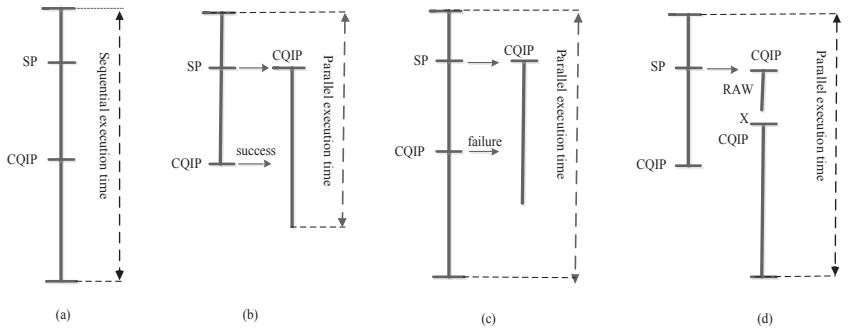

Figure 1. Model of Thread-Level Speculation: (a) Sequential Execution; (b) Successful Parallel Execution; (c) Failed Parallel Execution; (d) RAW.

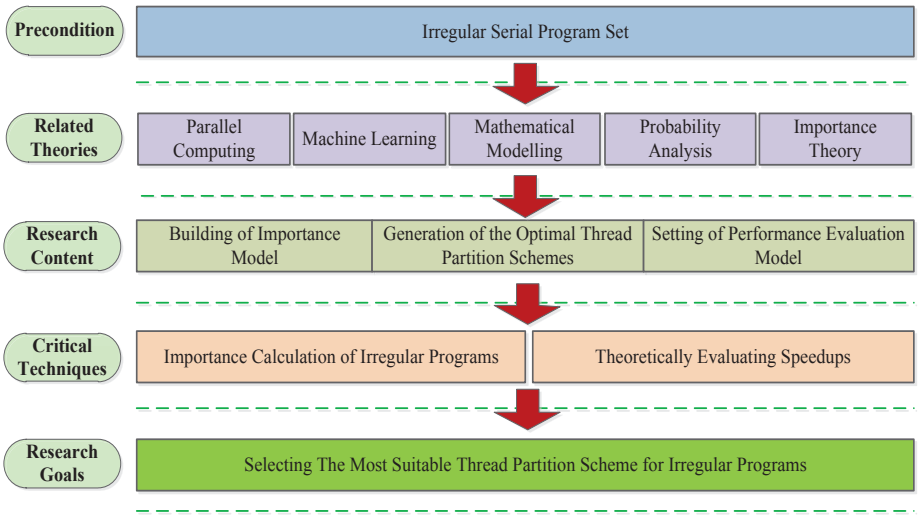

Figure 2. Design Flow of Research

TLS performance evaluation method can not be used in the thread division stage.

- Effective Construction of The Optimal Partition Scheme of Program Procedures
  Combined with the initial partition scheme generated by the machine-level thread-level estimation technology, the optimal partition scheme is constructed for the program procedure by calculating the importance of the program′s procedure and combining the performance evaluation results.Verifying the effect of explicit parallelization (ie, manual intervention partitioning) is always superior to the theory of implicit parallelization (ie, automatic parallelization).
- Exploring The Law that Program Characteristics Affecting the Speedup Performance
  By analyzing the factors affecting the parallelization of programs, the program complexity model, the candidate thread partitioning scheme set, the partitioning scheme selection mechanism are established, and the laws that program characteristics affect speedup performance are explored, which provide a method for parallelizing irregular programs on multi-core platforms.

V. Implementation of TPAoPI

The TPAoPI is the abbreviation of thread partition approach based on procedures′ importance. The implementation of it is as follows: firstly, on the foundation that a machine learning-based thread partition method automatically generated program′s partition schemes, combining with procedures′ importance and manual intervention, the optimal thread partition schemes are created; secondly, with performance evaluation model, the program and its procedures are evaluated, so to generate programs′ output and speedups.

A. Generation of Procedures′ Initial Partition Scheme and Calculation of Procedure′s Importance

This topic will be divided into the following four steps: i) according to the reference, irregular programs (Olden and SPEC benchmarks) are selected as the to-be-divided assembly; ii) compare and analyze the existing machine learning-based thread partition methods, the partitioning method that can best partition the Olden and SPEC programs so far is selected; iii) an automatic thread partitioning of programs

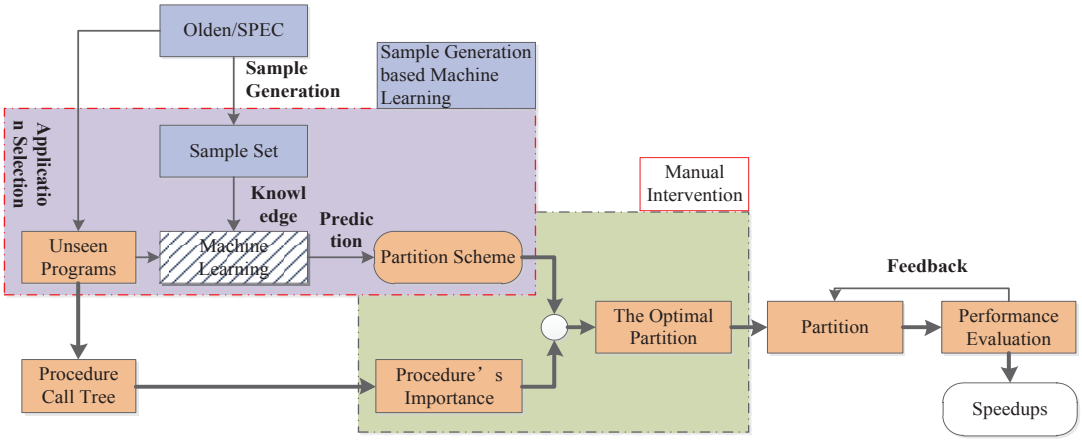

Figure 3. Overall Research Framework

is selected to generate a program partitioning scheme as the initial partitioning scheme of the procedures in the program; iv) perform statistics for procedures' characteristics, a procedure importance model is established to calculate the importance of the process.

B. Generation of The Optimal Partition Scheme

The generating process of the optimal partition scheme is a process of finding the optimal solution. This paper adopts the following technical methods: i) the solution is the set of partition scheme of the procedure, and the solution space of the partition scheme is constructed; ii) the importance of the procedure is designed to manually adjust the frequency and step size; iii) traverse the procedure call tree, and search the algorithm according to the partition scheme, so to find the possible solution of the procedure; iv) among all the results gotten by the performance evaluation model, the optimal solution of the procedure is selected.

In the flow of Fig.4, the first is to establish the call tree of the program's procedure; the second is to stratify the procedures to search for the best partitioning scheme; wherein, the process call tree adopts a bottom-up traversal sequence, and the same layer uses random order.

C. Establishment of Performance Evaluation Model

This paper comprehensively considers multiple speculative parallel overheads such as thread conflict and restart, inter-thread communication, thread distribution and submission, and load imbalance, and constructs a performance evaluation model based on probability graph. This section uses the following technical methods for this research: i) combining the profiling information of the input program in the run and the speculative control flow graph (SpCFG) of the program, constructing the WCFG of the program and the procedures' (the side in the figure represents branches of the program) probabilities); ii) calculation of serial execution time and speculative parallel time of programs and procedures based on analysis of SpCFG and WCFG; iii) calculation of serial execution time, parallel execution time of programs and procedures and speedups respectively, according to Amdahl's law.

VI. Experiment and Analysis

In order to show the effectiveness of TPAoPI, this paper makes a comparison between TPAoPI and the machine learning-based (ML-based) thread partition method, which makes use of programs as the partition unit. Olden benchmarks [12] which have complex data dependence and control dependence among basic blocks, are selected as the inputting programs. When we analyze the experimental results, we only compare the performance of the original ML-based thread partition approach and TPAoPI, and then we will analyze the experimental results, in which we only select several program analysis in the Olden benchmarks.

The main data structure of the program *health* is a two-way linked list, which contains both loop and nonloop structure. In *health*, the loop structure is the main source of parallelism, and compared with the HR-based partition approach, you can obtain the partition scheme of *health* suitable for its characteristics. During the partition of loop partition, although the loops occupy most of the program, but it has a large loop body and simple data dependence, so *health* gets 18.27% speedup improvement.

The main data structure of program *perimeter* is four fork tree, the program primarily contains loop structure, rather than nonloop structure. The parallelism of program mainly comes from the decomposition of function into multithreading. Because it is difficult to predict the return value of the function, the acceleration effect of these two approaches are not good. Compared with HR-based partition approach, TPAoPI selects the suitable partition scheme in line with its own characteristics, and the partition scheme is not affected by loops. The assessment models adopted by nonloops are used to find the better thread partition boundary

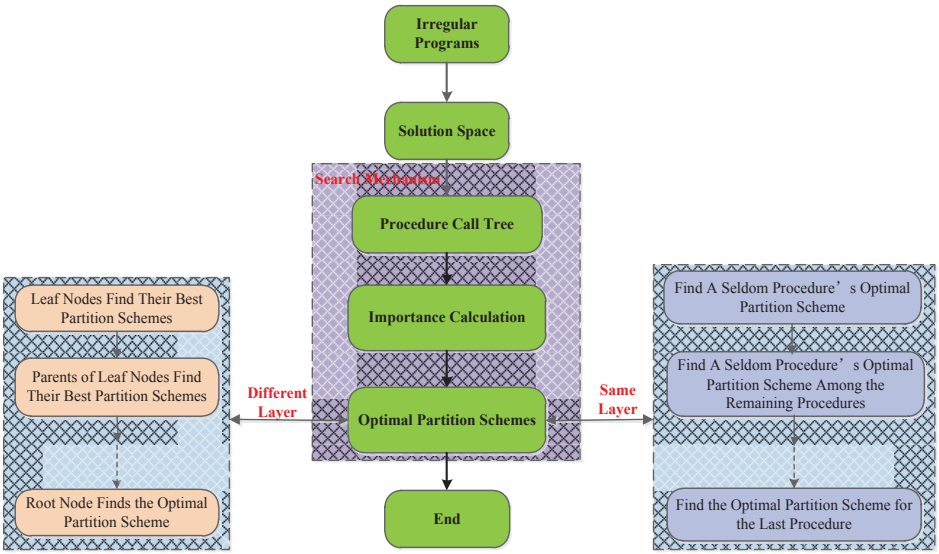

Figure 4. Flow of Procedures' Partition Schemes Generation

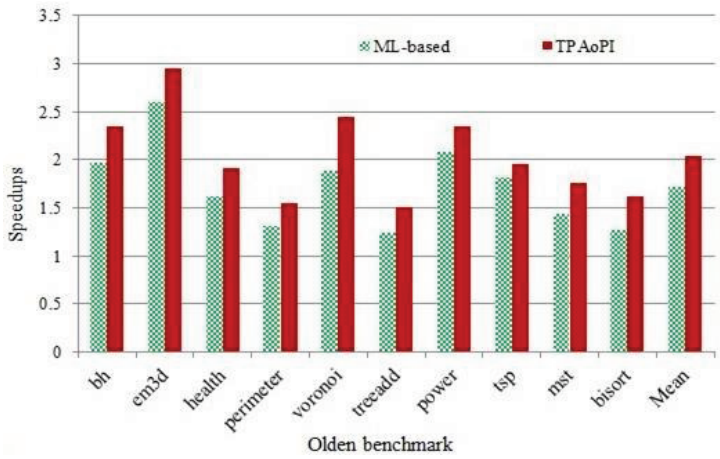

Figure 5. Comparison Diagram of Speedups for Olden Benchmarks

for the current program, so the final execution performance improves 18.23%.

The main data structure in program *bh* is a heterogeneous *octree*, which has very complex data dependence. Its parallelisms exist in and out of loop structures. For the heuristic rules, the same partition scheme is used to partition all the procedures in the *bh* program, and for the TPAoPI, the optimal partition scheme matching with the characteristic of every procedure in the program can be selected, and then the partition scheme is applied to the threads. However, due to the existence of more dependence, TPAoPI gains 19.54% performance improvement.

The main data structure of the program *em3d* is a single linked list, in which the loop structure occupies most of the total, and all the parallelism of program *em3d* comes mainly from the loop structure. Although TPAoPI can obtain the partition scheme suitable for its own characteristics, the characteristic extraction of the loop is not enough. Finally, compared with the HR-based partition approach, 13.17% performance improvement is achieved.

The main data structure of the program *bisort* is two fork tree. Through the analysis of the source code, we can see that there are only three loops in the program, and only two loops are executed, and the granularity of the loop is relatively small. Then the parallelism of program is mainly from the nonloops, although the program has a certain number of data dependence, but mining the potential parallelism from the application program can be performed based on the TPAoPI in every procedure. TPAoPI selects the suitable partition scheme for every procedure, finally obtains 27.47% performance improvement.

The main data structure of program *treeadd* is two fork

tree, which is a simple program structure. In this structure, only four procedures are included, and the program does not contain any loop structure, so the parallelism comes from the nonloops. TPAoPI can select the appropriate partition scheme for every procedure, but there are many recursive function calls and data dependence in *treeadd*, and finally the program achieves 21.19% performance improvement.

VII. Conclusion

Based on the Prophet system, this paper proposes a thread partitioning approach based on procedure importance (T-PAoPI), which starts with importance of procedures in irregular programs, adopting interdisciplinary research methods, creatively introducing the theory of importance in reliability theory, and calculates importance of procedures based on the characteristics of procedures. On the basis of the initial partition scheme, a manual intervention scheme suitable for the procedure is developed according to the importance of procedure, and a new performance evaluation model is used to theoretically evaluate obtained speedups, so that the best partitioning schemes of procedures are identified, exploring the intrinsic law that characteristics of procedures influence their best partitioning schemes. The paper does research from three aspects: calculating importance of procedures, generating the best partitioning schemes of procedures, and building of performance evaluation.

VIII. Acknowledgement

We thank all members of Henan Joint International Research Laboratory of Cyberspace Security Applications for their great support, and give our best hope to them for their collaboration. We also thank reviewers for their careful comments and suggestions. This work is supported by National Natural Science Foundation of China Grant No.61772174 and 61370220, and Plan For Scientific Innovation Talent of Henan Province Grant No.174200510011, as well as Program for Innovative Research Team (in Science and Technology) in University of Henan Province Grant No.15IRTSTHN010, and Open Foundation of State key Laboratory of Networking and Switching Technology (Beijing University of Posts and Telecommunications) SKLNST-2018-1-09.

References

[1] C. G. Quiones, C. Madriles, J. Sanchez, P. Marcuello, A. Gonzalez, and D. M. Tullsen, "Mitosis compiler: an infrastructure for speculative threading based on pre-computation slices," in *Acm Sigplan Conference on Programming Language Design & Implementation*, 2005.

[2] Y. Li, Y. Zhao, and Q. Wu, "A graph-based thread partition approach in speculative multithreading," in *IEEE International Conference on High Performance Computing & Communications; IEEE International Conference on Smart City; IEEE International Conference on Data Science & Systems*, 2017.

[3] ——, "Gba: A graphbased thread partition approach in speculative multithreading," *Concurrency & Computation Practice & Experience*, vol. 29, no. 21, p. e4294, 2017.

[4] T. A. Johnson, R. Eigenmann, and T. Vijaykumar, "Min-cut program decomposition for thread-level speculation," in *ACM Sigplan Notices*, vol. 39, no. 6. ACM, 2004, pp. 59–70.

[5] A. Bhowmik and M. Franklin, "A general compiler framework for speculative multithreading," in *Symposium on Parallelism in Algorithms & Architectures*, 2002.

[6] Y. Li, P. Yin, and Y. Zhao, "Thread partitioning algorithm of speculative multhreading based on fuzzy clusters," *Chinese Journal of Computers*, vol. 37, no. 3, pp. 580–592, 2014.

[7] B. Liu, Y. Zhao, X. Zhong, Z. Liang, and B. Feng, "A novel thread partitioning approach based on machine learning for speculative multithreading," in *High Performance Computing and Communications & 2013 IEEE International Conference on Embedded and Ubiquitous Computing (HPCC_EUC), 2013 IEEE 10th International Conference on*. IEEE, 2013, pp. 826–836.

[8] Z. Wang and M. F. O'Boyle, "Partitioning streaming parallelism for multi-cores: a machine learning based approach," in *Proceedings of the 19th international conference on Parallel architectures and compilation techniques*. ACM, 2010, pp. 307–318.

[9] D. Lasalle, M. Mostofa, A. Patwary, N. Satish, N. Sundaram, G. Karypis, and P. Dubey, "Improving graph partitioning for modern graphs and architectures," in *Workshop on Irregular Applications: Architectures & Algorithms*, 2015.

[10] Y. Li, Y. Zhao, and J. Shi, "A hybrid samples generation approach in speculative multithreading," in *High Performance Computing and Communications; IEEE 14th International Conference on Smart City; IEEE 2nd International Conference on Data Science and Systems (HPCC/SmartCity/DSS), 2016 IEEE 18th International Conference on*. IEEE, 2016, pp. 35–41.

[11] C. Madriles, C. Garcia-Quinones, J. Sanchez, P. Marcuello, A. Gonzalez, D. M. Tullsen, H. Wang, and J. P. Shen, "Mitosis:a speculative multithreaded processor based on precomputation slices," *IEEE Transactions on parallel and distributed systems*, vol. 19, no. 7, pp. 914–925, 2008.

[12] M. C. Carlisle, "Olden: parallelizing programs with dynamic data structures on distributed-memory machines," Ph.D. dissertation, Princeton University, 1996.