Poster: Scaling Data Plane Verification with Throughput-Optimized Atomic Predicates

Dong Guo
Tongji University

Jian Luo, Kai Gao
Sichuan University

Y. Richard Yang
Yale University

ABSTRACT

Atomic predicate is a key enabler to the rapid development of data plane verification, a technique to monitor and verify correctness of forwarding rules. Binary Decision Diagram (BDD) is widely used as the representation of atomic predicates for its simplicity of use, memory efficiency, and good performance when verifying general forwarding behaviors. However, building the atomic predicates is still the bottleneck in real-time data plane verification for large-scale networks, as existing BDD libraries do not scale well. In this paper, we identify the root cause of the inefficiency: general-purpose BDD libraries are aimed at speeding up a single BDD operation using parallelism rather than a batch of operations. Further, we propose TOBDD, a throughput-optimized BDD library that enables scaling of real-time data plane verification. Evaluations of the data plane verification system based on TOBDD report 2-10x improvement over the state-of-the-art centralized data plane verifier.

CCS CONCEPTS

• **Networks** → **Network reliability**; • **Computing methodologies** → **Parallel algorithms**;

KEYWORDS

Network Verification; Parallel Processing

ACM Reference Format:
Dong Guo, Jian Luo, Kai Gao, and Y. Richard Yang. 2023. Poster: Scaling Data Plane Verification with Throughput-Optimized Atomic Predicates. In *ACM SIGCOMM 2023 Conference (ACM SIGCOMM '23), September 10, 2023, New York, NY, USA.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3603269.3610845

1 INTRODUCTION

Scaling data plane verification (DPV) for large networks poses a significant challenge in real-time data plane verification. As the size of the network continues to grow, the number of data plane rules and the complexity of the network increase exponentially. For example, as claimed in Flash [1], even small topology changes in a short time can result in a large number of data plane updates. Thus, implementing a scalable real-time data plane verification system for large-scale networks is challenging.

Many data plane verification tools [1–6] have been proposed to enhance verification performance. These tools utilize various data structures to efficiently represent the data plane model and improve verification efficiency. The Binary Decision Diagram (BDD) is widely recognized as a core data structure in mainstream data plane verification systems [1, 2, 4, 7–9] due to its simplicity, memory efficiency, and ability to express complex forwarding rules in a concise manner. However, existing BDD based data plane verification systems primarily focus on reducing the number of BDD operations rather than improving the scalability. For example, APKeep [2] reduces redundant BDD operations by employing predicate split and delay merge techniques, while Flash [1] eliminates duplicate BDD operations through batch processing. Both of these approaches utilize a single-thread BDD package [10], which restricts their scalability. For large-scale networks, we realize that **there is a strong demand for a scalable BDD infrastructure to achieve scalable data plane verification.**

Specifically, the scalability of data plane verification systems is restricted by the scalability of BDD operations for two reasons: 1) **BDD operations contribute to the majority of verification time**. Empirical studies, such as the one conducted in Flash [1], have shown that BDD operations can account for the majority of the verification time (*i.e.*, the number of BDD operations reaches the order of $O(10^9)$ and takes over 99% of the overall time). 2) **Many parallelizable processes are hindered due to the absence of parallelism support in the underlying BDD package.** In the case of Flash [1], despite its Map-Reduce design, the actual implementation executes the Map functions sequentially due to the lack of thread-safe operations in the BDD library. In addition, discussions with developers of the latest data plane verification systems (*e.g.*, Coral [9]) revealed that they face the same problem.

Existing general-purpose BDD libraries do not perform well to support the scalability of data plane verification. We investigate various publicly available BDD libraries as listed in Table 1. Single-threaded libraries such as BuDDy [11], CUDD [12], and JDD [10] lack support for thread-safe parallel operations. BeeDeeDee [13] utilizes mutex-based structures for multi-threaded BDD operations, which hinders effective utilization of multi-core systems. PJBDD [14] and Sylvan use thread-pools and lock-free data structures for parallel BDD operations, but they use *fork-join* (FJ) scheme for each operation (*i.e.*, use multiple threads to process a single recursive BDD operation), which leads to larger overhead compared with *run-to-completion* (RtC) scheme (See Section 3). Further, the thread-pool in Sylvan lacks reschedulability, resulting in deadlocks when handling large-scale data plane updates and exhausting the available threads. HermesBDD [15] adopts on-demand thread creation approach for high scalability, but it suffers from overhead of thread scheduling and the over-threading problem during high volumes of BDD operations. Overall, we realize that the root cause of unsuitability of using existing BDD packages is that **existing general-purpose BDD packages aim to speed up a single BDD**

ACM SIGCOMM '23, September 10, 2023, New York, NY, USA

ACM ISBN 979-8-4007-0236-5/23/09…$15.00
https://doi.org/10.1145/3603269.3610845

Table 1: Publicly available BDD libraries.

Name	*Lang*	*Thread-safe*	*Scalability*	*Synchronization*	*Execution*
BuDDy [11]	C/C++	N	Poor	User-lock	RtC
CUDD [12]	C++	N	Poor	User-lock	RtC
JDD [10]	Java	N	Poor	User-lock	RtC
BeeDeeDee [13]	C++	Y	Medium	Mutex-lock	RtC
PJBDD [14]	Java	Y	Medium	Lock-free	FJ
Sylvan [16]	C++	Y	Medium	Lock-free	FJ
HermesBDD [15]	C++	Y	High	Lock-free	FJ
TOBDD	C++	Y	High	Lock-free	RtC

operation but not the overall throughput to handle a large number of BDD operations.

It is worth noting that techniques such as header space partition [17] and topology partition [9, 18] are orthogonal to our specific objective. Thus, they can be utilized together to address the overall scalability challenge.

In this paper, we propose TOBDD, throughput-optimized BDD. TOBDD is motivated by the idea of lock-free structures proposed in existing BDD studies and the *run-to-completion* in parallel computing. We implement TOFlash based on TOBDD. The evaluation results show 2-10x improvement over the state-of-the-art centralized data plane verifier.

2 THROUGHPUT-OPTIMIZED BDD

TOBDD is designed to achieve high throughput parallel BDD operations by leveraging the multi-core capabilities of modern hardware. We give a compact overview of TOBDD.

Reschedulable Thread-pool. To eliminate the overhead of thread creation and destruction, TOBDD employs a thread-pool for BDD operations, and the number of threads is designed to equal the number of CPU cores. In addition, each thread is reschedulable which ensures that when a thread is blocked, such as waiting for another thread to complete, it can be automatically rescheduled to execute another job. This design effectively mitigates the issue of deadlocks in the system.

Lock-free Node Table and Cache. Similar to existing packages [15, 16], TOBDD leverages the atomic operation Compare-and-Swap (CAS) [19], which is provided by modern hardwares, to implement a lock-free BDD node table and cache. Different from other approaches, TOBDD uses dynamic linked lists to index BDD nodes, ensuring that the node table will never be full until the hardware memory is fully utilized. This eliminates unexpected garbage collections during the real-time verification.

Run-to-Completion. TOBDD applies the *run-to-completion* approach in parallel computing, where each BDD operation is executed without interruption until completion. This approach avoids the overhead of synchronization and is more efficient compared to the *fork-join* paradigm. The reason to adopt RtC rather than *fork-join* is that in the context of large-scale network verification, where a large number of parallel BDD operations already fully utilize the CPU, attempting to improve a single BDD operation through sub-task parallelism can actually degrade performance instead.

Commutative Hash Function. To optimize cache performance, TOBDD employs a hash function that maps commutative operations to the same hash value. For example, for the $\wedge$ operation of two predicates x and y, we use an ordered tuple hash function to guarantee $hash(x, y) = hash(y, x)$. Compared with the existing BDD packages, this design can further reduce cache misses and enhance the overall throughput.

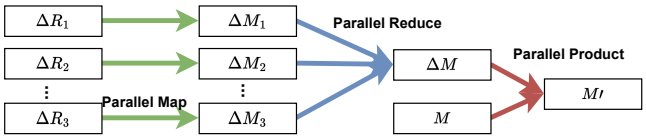

Figure 1: Workflow of throughput-optimized DPV.

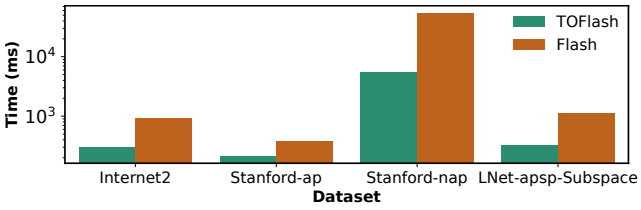

Figure 2: Evaluation result.

3 SCALING DPV WITH TOBDD

We use the inverse model transformation process in Flash [1] as a case study to show how data plane verification can benefit from TOBDD to achieve scalability. Specifically, we parallelize the Map-Reduce and Product processes in Flash as shown in Figure 1 (refer to Flash paper for notations).

Scaling Map. The Map functions for transforming native updates to atomic overwrites are executed in parallel, and within each Map function, the atomic overwrites are computed concurrently.

Scaling Reduce. The Reduce functions for aggregating predicates and actions are are parallelized using recursive splitting and merging. For example, when aggregating a set of predicates by an operator, the set is recursively divided into subsets, and the pairwise leaf subsets are recursive merged by the operator in parallel.

Scaling Product. The Product operation in the final model update process can be efficiently parallelized by concurrently processing each pair of elements in the product space.

4 EVALUATION

We implement TOBDD [1] and a throughput-optimized version of Flash called TOFlash in C++. We compare the data plane model construction time of TOFlash with our previously implemented Flash artifact on various datasets in Flash [1], setting the latter as the baseline. The evaluations are conducted on a server with Intel E5-2660 CPU (40core, 2.6GHz) and 256G memory. Figure 2 shows the evaluation results. We can see that TOFlash achieves 2-10x improvement compared with the baseline.

5 CONCLUSION

We present TOBDD, a throughput-optimized BDD library that enables the scaling of real-time data plane verification. Evaluations report 2-10x improvement over the state-of-the-art centralized data plane verification system.

[1] https://github.com/guodong/tobdd

REFERENCES

[1] Dong Guo, Shenshen Chen, Kai Gao, Qiao Xiang, Ying Zhang, and Y. Richard Yang. Flash: Fast, consistent data plane verification for large-scale network settings. In *Proceedings of the ACM SIGCOMM 2022 Conference*, pages 314–335, Amsterdam Netherlands, August 2022. ACM. ISBN 978-1-4503-9420-8. doi: 10.1145/3544216.3544246.

[2] Peng Zhang, Xu Liu, Hongkun Yang, Ning Kang, Zhengchang Gu, and Hao Li. APKeep: Realtime network verification for real networks. In *Proceedings of the 17th Usenix Conference on Networked Systems Design and Implementation*, NSDI'20, pages 241–256, USA, 2020. USENIX Association. ISBN 978-1-939133-13-7.

[3] Alex Horn, Ali Kheradmand, and Mukul R. Prasad. Delta-net: Real-time network verification using atoms. In *Proceedings of the 14th USENIX Conference on Networked Systems Design and Implementation*, NSDI'17, pages 735–749, USA, 2017. USENIX Association. ISBN 978-1-931971-37-9.

[4] Hongkun Yang and Simon S. Lam. Real-Time Verification of Network Properties Using Atomic Predicates. *IEEE/ACM Transactions on Networking*, 24(2):887–900, April 2016. ISSN 1063-6692, 1558-2566. doi: 10.1109/TNET.2015.2398197.

[5] Peyman Kazemian, George Varghese, and Nick McKeown. Header Space Analysis: Static Checking For Networks. In *Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation*, NSDI'12, USA, 2012.

[6] Peyman Kazemian, Michael Chang, Hongyi Zeng, George Varghese, Nick McKeown, and Scott Whyte. Real time network policy checking using header space analysis. In *10th USENIX Symposium on Networked Systems Design and Implementation*, NSDI'13, pages 99–111, Lombard, IL, April 2013. USENIX Association.

[7] Hongkun Yang and Simon S. Lam. Scalable verification of networks with packet transformers using atomic predicates. *IEEE/ACM Transactions on Networking*, 25 (5):2900–2915, 2017. doi: 10.1109/TNET.2017.2720172.

[8] Peng Zhang, Aaron Gember-Jacobson, Yueshang Zuo, Yuhao Huang, Xu Liu, and Hao Li. Differential Network Analysis. In *19th USENIX Symposium on Networked Systems Design and Implementation*, NSDI'22, pages 601–615, Renton, WA, April 2022. USENIX Association.

[9] Qiao Xiang, Ridi Wen, Chenyang Huang, Yuxin Wang, and Franck Le. Network can check itself: Scaling data plane checking via distributed, on-device verification. In *Proceedings of the 21st ACM Workshop on Hot Topics in Networks*, pages 85–92, Austin Texas, November 2022. ACM. ISBN 978-1-4503-9899-2. doi: 10.1145/3563766.3564095.

[10] Arash Vahidi. A BDD and Z-BDD Library written in Java, 2023. URL https://bitbucket.org/vahidi/jdd.

[11] Lind-Nielsen Jorn. Buddy: A bdd package, 1999. URL https://buddy.sourceforge.net/manual/main.html.

[12] Fabio Somenzi. Cudd: Cu decision diagram package, 1999. URL https://add-lib.scce.info/assets/documents/cudd-manual.pdf.

[13] Alberto Lovato, Damiano Macedonio, and Fausto Spoto. A Thread-Safe Library for Binary Decision Diagrams. In Dimitra Giannakopoulou and Gwen Salaün, editors, *Software Engineering and Formal Methods*, volume 8702, pages 35–49. Springer International Publishing, Cham, 2014. ISBN 978-3-319-10430-0 978-3-319-10431-7. doi: 10.1007/978-3-319-10431-7_4.

[14] Dirk Beyer, Karlheinz Friedberger, and Stephan Holzner. PJBDD: A BDD Library for Java and Multi-Threading. In Zhe Hou and Vijay Ganesh, editors, *Automated Technology for Verification and Analysis*, volume 12971, pages 144–149. Springer International Publishing, Cham, 2021. ISBN 978-3-030-88884-8 978-3-030-88885-5. doi: 10.1007/978-3-030-88885-5_10.

[15] Luigi Capogrosso, Luca Geretti, Marco Cristani, Franco Fummi, and Tiziano Villa. HermesBDD: A multi-core and multi-platform binary decision diagram package. *arXiv preprint arXiv:2305.00039*, 2023.

[16] Tom van Dijk and Jaco van de Pol. Sylvan: Multi-core framework for decision diagrams. *International Journal on Software Tools for Technology Transfer*, 19(6):675–696, November 2017. ISSN 1433-2779, 1433-2787. doi: 10.1007/s10009-016-0433-2.

[17] Hongyi Zeng, Shidong Zhang, Fei Ye, Vimalkumar Jeyakumar, Mickey Ju, Junda Liu, Nick McKeown, and Amin Vahdat. Libra: Divide and Conquer to Verify Forwarding Tables in Huge Networks. In *11th USENIX Symposium on Networked Systems Design and Implementation*, NSDI'14, pages 87–99, Seattle, WA, April 2014. USENIX Association.

[18] Karthick Jayaraman, Nikolaj Bjørner, Jitu Padhye, Amar Agrawal, Ashish Bhargava, Paul-Andre C Bissonnette, Shane Foster, Andrew Helwer, Mark Kasten, Ivan Lee, Anup Namdhari, Haseeb Niaz, Aniruddha Parkhi, Hanukumar Pinnamraju, Adrian Power, Neha Milind Raje, and Parag Sharma. Validating Datacenters at Scale. In *Proceedings of the ACM Special Interest Group on Data Communication*, SIGCOMM'19, pages 200–213, Beijing China, August 2019. ACM.

[19] Rachid Guerraoui, Alex Kogan, Virendra J. Marathe, and Igor Zablotchi. Efficient multi-word compare and swap, 2020.