Towards Capturing Nondeterminism Motifs in HPC Applications

Dylan Chapp
University of Delaware
dchapp@udel.edu

Kento Sato (Advisor)
Lawrence Livermore National Laboratory
sato5@llnl.gov

Dong H. Ahn (Advisor)
Lawrence Livermore National Laboratory
ahn1@llnl.gov

Michela Taufer (Advisor)
University of Delaware
taufer@udel.edu

ABSTRACT

HPC applications employ nondeterministic asynchronous communication to achieve greater performance. However, this nondeterminism can significantly hamper numerical reproducibility and debugging. Various software tools have been developed to control nondeterminism in HPC applications, but a high-level application-agnostic taxonomy for this nondeterminism is absent and limits these tools' effectiveness in practice. We address this need by proposing a workflow to extract common nondeterministic communication motifs from representative applications. We present a first step towards capturing nondeterminism motifs by way of our workflow for detecting and summarizing sender-nondeterminism in message passing applications.

ACM Reference format:
Dylan Chapp, Kento Sato (Advisor), Dong H. Ahn (Advisor), and Michela Taufer (Advisor). 2017. Towards Capturing Nondeterminism Motifs in HPC Applications. In *Proceedings of , , ,* 2 pages.
DOI: 10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

With the increasingly large degree of concurrency and asynchrony expected in future scientific applications on exascale systems, there will be an associated decrease in the degree of determinism associated with the applications' execution. With this decrease in determinism, undesirable side-effects associated to numerical irreproducibility, hampered debugging, and incompatibility with tools that assume certain forms of determinism are expected to emerge and will required proper consideration by the exascale community. As nondeterminism becomes more and more prevalent at the application level through increased use of, for example, nonblocking communication, coping with and reasoning about nondeterminism in an ad-hoc manner as is common in the present will become infeasible. Even now it is apparent that nondeterminism can drastically increase the cost of debugging both in terms of developer time and computational resources. The case study presented in [9] is indicative of this growing trend.

In this work, we propose to address the problems of nondeterminism in next-generation exascale applications by extracting common communication patterns or *motifs* in applications that exhibit nondeterminism due to the use of message-passing programming models, thereby freeing developers and users from having to reason about the nondeterminism in their applications in an ad-hoc manner, and enabling them to target their application's specific nondeterminism with appropriate tools. In our poster, we present our workflow for identifying and analyzing representative nondeterminism snippets (i.e., small regions of source code). We move the first steps toward building a classification system of nondeterminism patterns or motifs associated to these snippets to enable systematic side-effects' control strategies in exascale applications.

2 MOTIVATION

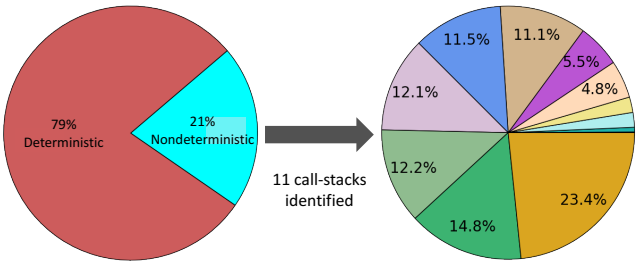
There are uncountable examples of undesirable side-effects associated to nondeterminism. We focus on HPC applications that employ the message passing programming model in particular because of the large community using codes based on this model, the mechanisms by which nondeterminism manifests from its use, and the fact that it is expected this model will be one of the most popular on exascale platforms.

In particular, we focus on the sender-nondeterminism because it is twisted together with the usability of existing tools for controlling receiver-nondeterminism [8] [4]. Sender-nondeterminism also has grave implications for a particular class of fault-tolerance protocols [6] [7] that explicitly assume sender-determinism, as defined by Cappello and Snir in [5]. While it is true that at the time of these tools' development many HPC applications exhibited strict sender-determinism, that assumption will become less valid for future codes on exascale platforms. Therefore, detection of sender-nondeterminism and characterization of its causes will be vital to avoiding misapplication of software tools to inappropriate use cases.

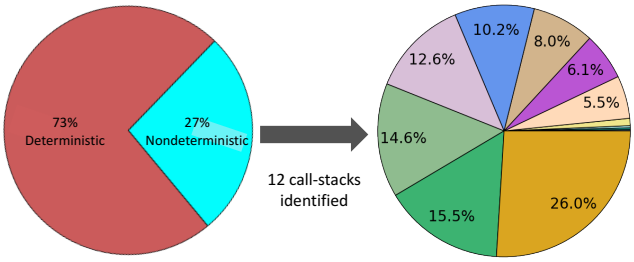
3 METHODOLOGY

The goal of our study is to provide the ability to HPC developers to refine their understanding of application-level sender-nondeterminism by identifying the specific sequences of function calls that terminate in calls to message passing send-family functions (e.g., MPI_Isend) that manifest some form of nondeterminism. For our purposes, we consider run-to-run variation in any of (1) the ordering of a set of sends, (2) their contents, and (3) their destinations to be a manifestation of nondeterminism.

Our workflow consist of two phases. In the first phase, an application is linked with our tracing library and executed multiple times. The trace contains a record of every message passing send-family function call as well as the call-stack that terminates in the send. In the second phase, the sequences of sends for each message passing rank for each run are extracted and compared, facilitating detection of nondeterministic sends. Once the nondeterministic sends



(a) ParaDiS with load-balancing off



(b) ParaDiS with load-balancing on

Figure 2: ParaDiS's percentages of deterministic sends vs. nondeterministic sends (left) and percentage breakdown of nondeterministic sends by call-stack (right). Note that when load-balancing is turned on (Figure 2b, the number of call-stacks responsible for nondeterministic sends increases–i.e., load-balancing introduces an additional form of nondeterminism.

are identified, the call-stacks associated with the nondeterministic sends are aggregated, thus presenting a summary of the application-level source of sender-nondeterminism. Moreover, if finer-grained understanding of the nondeterminism is desired, reduction of the search space from the entire code base down to just the fraction of call stacks actually associated with the nondeterminism represents a significant convenience.

We implement our sender-nondeterminism identification workflow via the MPI Profiling interface [3] and a set of scripts that perform the nondeterminism detection phase.

4 RESULTS

We demonstrate how our approach provides insight into the sender-nondeterminism of applications via case studies on two HPC applications: the Monte Carlo Benchmark (MCB) proxy application [1], and ParaDiS, a dislocation dynamics code [2]. For each application, we collected traces of 100 8-node, 64-process executions on the Catalyst Linux cluster at Lawrence Livermore National Laboratory.

In our analysis of the sender-nondeterminism in MCB, shown in Figure 1, we observe that only a small fraction of the total sends over all executions exhibited nondeterminism (i.e., 11%). Moreover, out of the nondeterministic sends most are accounted for by a single call-stack. In this sense, the sender-nondeterminism in MCB is compactly summarized by our approach.

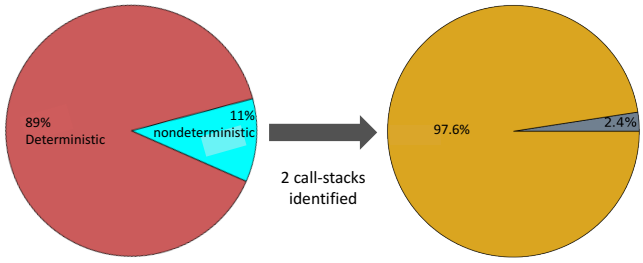


Figure 1: MCB's percentages of deterministic sends vs. nondeterministic sends (left) and percentage breakdown of nondeterministic sends by call-stacks (right).

In contrast, our analysis of ParaDiS, shown in Figure 2, reveals much more prevalent and diverse sender-nondeterminism. We collected traces of ParaDiS both with its load-balancing feature off, shown in Figure 2a and on, shown in Figure 2b. We observed that nondeterminism sends can range between 21%5 and 27% and that when load-balancing is turned on, an additional call-stack associated with nondeterministic sends is detected (i.e., from 11 to 12 different call-stacks). This is evidence that our approach can detect differences in nondeterminism sources caused by seemingly innocuous changes in configuration options.

5 CONCLUSION AND FUTURE WORK

We have presented an approach to capturing and summarizing sender-nondeterminism in message passing applications, and demonstrated the effectiveness of the approach on two such applications thus revealing their distinct nondeterminism characteristics. This work is a critical first step towards the overarching goal of capturing *nondeterminism motifs and building a taxonomy of nondeterminism in HPC applications. Future work will leverage our approach to enable finer-grained classification of nondeterminism. Moreover, our approach crucially narrows the scope of investigation into the causes of nondeterminism by paring down complex code bases into managable subsets known to contribute to nondeterminism.*

REFERENCES

[1] *Monte Carlo Benchmark. https://codesign.llnl.gov/mcb.php.*
[2] *ParaDiS. http://paradis.stanford.edu/.*
[3] *MPI: A Message-Passing Interface Standard, Version 3.0. Technical report, 2012.*
[4] *D. H. Ahn, G. L. Lee, G. Gopalakrishnan, Z. Rakamarić, M. Schulz, and I. Laguna. Overcoming Extreme-scale Reproducibility Challenges Through a Unified, Targeted, and Multilevel Toolset. In* Proc. of the 1st International Workshop on Software Engineering for High Performance Computing in Computational Science and Engineering (SE-HPCCSE), *2013.*
[5] *F. Cappello, A. Guermouche, and M. Snir. On communication determinism in parallel hpc applications. In* 2010 Proceedings of 19th International Conference on Computer Communications and Networks, *pages 1–8, 2010.*
[6] *A. Guermouche, T. Ropars, M. Snir, and F. Cappello. HydEE: Failure Containment without Event Logging for Large Scale Send-Deterministic MPI Applications. In* Proc. of the 2012 IEEE 26th International Parallel and Distributed Processing Symposium, *pages 1216–1227, May 2012.*
[7] *A. Lefray, T. Ropars, and A. Schiper. Replication for Send-deterministic MPI HPC Applications. In* Proc. of the 3rd Workshop on Fault-tolerance for HPC at Extreme Scale (FTXS), *2013.*
[8] *K. Sato, D. H. Ahn, I. Laguna, G. L. Lee, and M. Schulz. Clock Delta Compression for Scalable Order-replay of Non-deterministic Parallel Applications. In* Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC), *2015.*
[9] *K. Sato, D. H. Ahn, I. Laguna, G. L. Lee, M. Schulz, and C. M. Chambreau. Noise Injection Techniques to Expose Subtle and Unintended Message Races. In* Proc. of the 22Nd ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming (PPoPP), *2017.*