InvisiSpec: Making Speculative Execution Invisible in the Cache Hierarchy (Corrigendum)

Mengjia Yan
Univ. of Illinois Urbana-Champaign
myan8@illinois.edu

Jiho Choi
Univ. of Illinois Urbana-Champaign
jchoi42@illinois.edu

Dimitrios Skarlatos
Univ. of Illinois Urbana-Champaign
skarlat2@illinois.edu

Adam Morrison
Tel Aviv University
mad@cs.tau.ac.il

Christopher W. Fletcher
Univ. of Illinois Urbana-Champaign
cwfletch@illinois.edu

Josep Torrellas
Univ. of Illinois Urbana-Champaign
torrella@illinois.edu

ACM Reference Format:
Mengjia Yan, Jiho Choi, Dimitrios Skarlatos, Adam Morrison, Christopher W. Fletcher, and Josep Torrellas. 2019. InvisiSpec: Making Speculative Execution Invisible in the Cache Hierarchy (Corrigendum). In *The 52nd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO-52), October 12–16, 2019, Columbus, OH, USA.* ACM, New York, NY, USA, 1 page. https://doi.org/10.1145/3352460.3361129

InvisiSpec Performance

We found a bug in our simulator that causes the simulation to incorrectly represent the InvisiSpec design described in the MICRO 2018 paper [1]. In InvisiSpec, an unsafe speculative load (USL) issues an invisible load request (Spec-GetS) to obtain data. As soon as the data is returned, the data is used by the requesting instruction and propagated to all of the dependent instructions. This is one of the keys to InvisiSpec's performance. However, in our simulator, due to a bug, the operation of data propagation to dependent instructions was performed only after the USL reached its visibility point—not as soon as the data requested was received by the core. This late propagation of speculatively accessed data hurt InvisiSpec's performance. We have fixed the bug and updated the code repository at https://github.com/mjyan0720/InvisiSpec-1.0.

After fixing the bug, InvisiSpec's overhead drops by a large amount. As an example, for the SPEC applications under TSO, in the Spectre attack model, the average overhead decreases from 22% (before the bug fix) to 7.6% (after the bug fix); in the Futuristic attack model, the average overhead decreases from 80% (before) to 18.2% (after). The new execution bars are shown in Figure 1. In the figure, *Base* is an unsafe, conventional processor, *IS-Sp* is InvisiSpec in Spectre, and *IS-Fu* is InvisiSpec in Futuristic.

Similar InvisiSpec's improvements are obtained for the PARSEC applications. For PARSEC applications, to obtain repeatable results, we have also pinned threads to cores. Specifically, we have annotated the PARSEC applications to configure the affinity attribute of thread creation so that each thread is pinned to a core.

Transforming a Validation into an Exposure

When issuing a USL, we refined the condition that allows the hardware to mark it as needing an Exposure rather than a Validation. Specifically, we added the second condition below:

"At the time of issuing a USL_1 load, all of the loads in the ROB earlier than USL_1 need to satisfy two conditions: 1) they have already obtained the data they requested—in particular, if they are USLs, the data they requested has already arrived at the SB and been passed to a register, and 2) if they needed to perform validations, they have completed them."

This change has no noticeable effect on InvisiSpec's performance.

Revised Version of the MICRO 2018 Paper

We have created a revised version of the MICRO 2018 paper in [2]. The paper revises the evaluation (Section IX), the condition of how to transform a validation into an exposure (Section V-C1), and the proof that InvisiSpec supports TSO (Appendix). It also clarifies how to use the load queue State bits (Section VI-A1).

MICRO-52, October 12–16, 2019, Columbus, OH, USA

ACM ISBN 978-1-4503-6938-1/19/10.
https://doi.org/10.1145/3352460.3361129

REFERENCES

[1] M. Yan, J. Choi, D. Skarlatos, A. Morrison, C. Fletcher, and J. Torrellas. 2018. InvisiSpec: Making Speculative Execution Invisible in the Cache Hierarchy. In *MICRO'18.*

[2] M. Yan, J. Choi, D. Skarlatos, A. Morrison, C. Fletcher, and J. Torrellas. 2019. Correction: InvisiSpec: Making Speculative Execution Invisible in the Cache Hierarchy. Technical Report, University of Illinois at Urbana-Champaign, http://iacoma.cs.uiuc.edu/iacoma-papers/corrected_micro18.pdf.

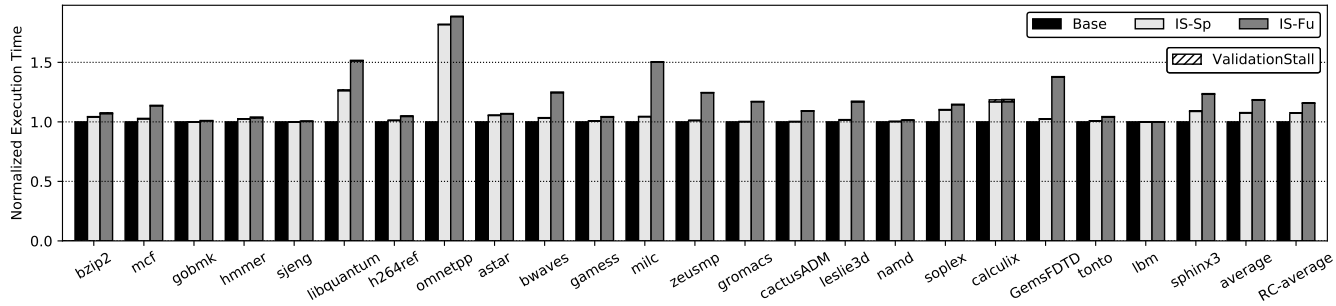

Figure 1: SPEC execution time with TSO and RC-average.