Implementation of SHAPES Case Studies (Artifact)

Alexandros Tasos
Imperial College London, United Kingdom
at1917@ic.ac.uk

Juliana Franco
Microsoft Research, London, United Kingdom
juliana.franco@microsoft.com

Sophia Drossopoulou
Imperial College London, United Kingdom
Microsoft Research, London, United Kingdom
scd@doc.ic.ac.uk

Tobias Wrigstad
Uppsala University, Sweden
tobias.wrigstad@it.uu.se

Susan Eisenbach
Imperial College London, United Kingdom
sue@doc.ic.ac.uk

Abstract

Our main paper presents SHAPES, a language extension which offers developers fine-grained control over the placement of data in memory, whilst retaining both memory safety and object abstraction via pooling and clustering. As part of the development of SHAPES, we wanted to investigate the usefulness of the concepts SHAPES brings to the table. To that extent, we implemented five such case studies. This publication provides the corresponding code and instructions on how to run these case studies and derive the results we provide.

2012 ACM Subject Classification Software and its engineering → Classes and objects; Theory of computation → Formalisms; General and reference → Performance

Keywords and phrases Cache utilisation, Data representation, Memory safety

Digital Object Identifier 10.4230/DARTS.6.2.19

Funding *Alexandros Tasos*: The support of the EPSRC Centre for Doctoral Training in High Performance Embedded and Distributed Systems (HiPEDS, Grant Reference EP/L016796/1) is gratefully acknowledged.

Related Article Alexandros Tasos, Juliana Franco, Sophia Drossopoulou, Tobias Wrigstad, and Susan Eisenbach, "Reshape Your Layouts, Not Your Programs: A Safe Language Extension for Better Cache Locality", in 34th European Conference on Object-Oriented Programming (ECOOP 2020), LIPIcs, Vol. 166, pp. 31:1–31:3, 2020. https://doi.org/10.4230/LIPIcs.ECOOP.2020.31

Related Conference 34th European Conference on Object-Oriented Programming (ECOOP 2020), November 15–17, 2020, Berlin, Germany (Virtual Conference)

1 Scope

SHAPES is a language extension intended to allow developers to control how objects are laid out in memory in a more fine-grained way compared to conventional managed languages. This is achieved by introducing the concepts of object *pooling* (objects are placed into pools of a specific type, objects in the same pool are placed close to each other in memory) and *clustering* (each pool adheres to a layout, which dictates how the objects in that pool are laid out in memory). This is achieved in a type-based manner.

In order to show the usefulness of these concepts and justify the design of SHAPES, we have presented five case studies in our original paper (§3) which show that these concepts do have, indeed, merit. The case studies we have implemented are the following:

- *OP2* (§3.1, §G.2 in the original paper), where we compare against an existing open source C++ library (OP2 [1]). OP2 mainly attempts to tackle the issue of executing a kernel over a set of data in parallel in a declarative manner. It also provides pooling and clustering feautres, albeit more limited compared to those of SHAPES.
- *Skeletal Animation* (§3.2 in the original paper), which explores the use of different layouts to determine the fastest layout for animating a 3D model that uses the MD5Anim skeletal animation format [2].
- *Traffic* (§G.3 in the original paper), which explores the use of different layouts to speed up a traffic simulation based on the Nagel-Schreckenberg traffic model [3]. Our implementation is derived from a CUDA implementation presented in [4].
- *Doors* (§G.4 in the original paper), wherein we must determine whether a `Door` of a specific `Allegiance` must be open because a `Character` of the same `Allegiance` is close to it. The case study partitions objects into multiple pools to improve performance.
- *Currency* (§3.3 in the original paper), which reflects a query system with real-world data (daily exchange rates against the Euro) and made-up queries (exchange rate of a specific currency on a specific date) and attempts to improve performance by using multiple pools of the same class and having each pool use a different layout.

These case studies are run of *five different machines* and the results from the execution on each machine *are consolidated* in order to generate the final charts.

2 Content

This artifact contains a Zip file, consisting of the source code for our case studies and is split into the following directories:

- `op2orig/` and `op2reimpl/` provide the source code for the *original* OP2 commit (and our patch) we used and for *our own implementation*, respectively.
- `stickmen/` contains the source code for *Skeletal Animation*.
- `traffic/` contains the source code for *Traffic*.
- `doors/` contains the source code for *Doors*.
- `forex/` contains the source code for *Currency*.
- `lib/` contains the source code for the Google C++ Benchmarking library (which we use for measurements on all case studies but OP2).
- `charts/` contains the LaTeX source code we use in our paper to generate our charts. The LaTeX source code parses the CSV files located in `charts/csv_data` in order to generate the charts.

3 Getting the artifact

The artifact endorsed by the Artifact Evaluation Committee is available free of charge on the Dagstuhl Research Online Publication Server (DROPS). The artifact is also available at: `https://drive.google.com/file/d/17ioOVFJHwqPxztak-ouDeQhos9parQPh/view?usp=sharing`.

The GitHub repository of the artifact is also available at: `https://github.com/octurion/ecoop-artifact`.

4 Tested platforms

We use Docker to build our artifact; we expect our artifact to build and run on any Linux machine with Docker installed.

We require at least a dual core machine with hyperthreading (i.e. 4 logical cores) to run the use cases and obtain numbers similar to ours.

5 License

The OP2 project and our OP2-related patches are licensed under the 3-Clause BSD License (incorrectly stated as 2-Clause BSD). All remaining case studies are licensed under the MIT License.

6 MD5 sum of the artifact

0e70eb4c6eaaa071ceb3a0aaf2cb8dbc

7 Size of the artifact

28.94 MiB

References

1 M. B. Giles, G. R. Mudalige, Z. Sharif, G. Markall, and P. H.J. Kelly. Performance analysis of the op2 framework on many-core architectures. *SIGMETRICS Perform. Eval. Rev.*, 38(4):9–15, March 2011. doi:10.1145/1964218.1964221.

2 David Henry. Md5mesh and md5anim files formats. http://tfc.duke.free.fr/coding/md5-specs-en.html, Wayback Machine: https://web.archive.org/web/20180816101227/http://tfc.duke.free.fr/coding/md5-specs-en.html, 2005.

3 Kai Nagel and Michael Schreckenberg. A cellular automaton model for freeway traffic. *Journal de physique I*, 2(12):2221–2229, 1992.

4 Matthias Springer and Hidehiko Masuhara. Ikra-cpp: A c++/cuda dsl for object-oriented programming with structure-of-arrays layout. In *Proceedings of the 2018 4th Workshop on Programming Models for SIMD/Vector Processing*, page 6. ACM, 2018.