

Performance Portable Software Design Patterns – What they can do and why we should collect them.

Authors

Jakob Bludau bludauj@ornl.gov, Computing and Computational Sciences Directorate, Oak Ridge National Laboratory, 0000-0002-4128-4578

Keywords

Software Design Patterns, Performance portable programming, BSSw, Heterogeneous computing, Community Hub

Abstract

With the transition to heterogeneous supercomputing, the high-performance computing community was tasked with developing solutions to leverage various heterogeneous hardware. This led to the development of performance-portability libraries like Kokkos, Raja, Hemi, YAKL, etc. which provide performance portable abstractions for algorithmic elements (loops, reductions, etc.) and data storage. Nevertheless, leveraging these abstractions in scientific software in a sustainable way is left to the developers of that software.

Software design patterns are a common way to help developers create sustainable software by making software easier to understand and maintain. The design patterns achieve this by representing reusable solutions to common problems [1]. Therefore, they help to reduce complexity of large code bases and allow to design, teach, and learn software in steps. Furthermore, they are designed to be extensible and general, thus ensuring sustainable software design.

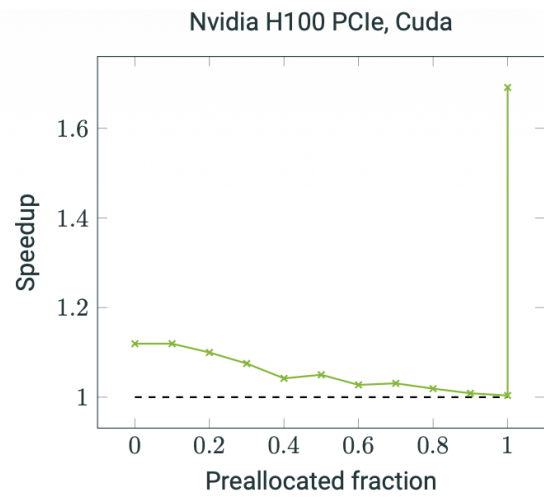
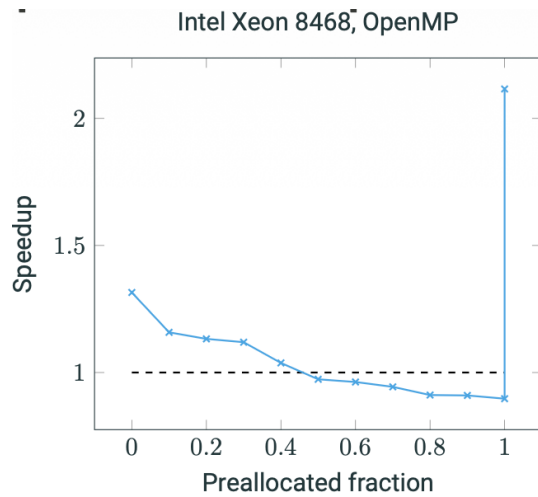
Most widely spread software design patterns are CPU focused [1]. But many of the techniques that the software patterns leverage are unavailable on contemporary computing hardware that is heterogeneous and massively parallel. For example, GPUs do not allow to allocate heap memory within kernels and dynamic polymorphism is restricted.

Here a crucial gap appears:

Performance portable design patterns that solve these abstract software design problems lack a central place where they are collected, discussed, and curated. I present a public Github Pages website that showcases performance portable patterns extracted from performance-portable open-source code. The collection helps developers to decide which design patterns apply to a problem via abstract descriptions (synopsis), example implementation, and links to open-source software where they are used. Furthermore, it lists the restrictions of heterogeneous hardware to motivate and explain the patterns.

This new resource becomes especially relevant in AI-assisted development. As the correctness check is crucial for scientific code, it needs to be reviewable, and the reviewer needs high confidence regarding correctness. Design patterns help to break the complexity of scientific codes into manageable pieces and allow scientists to focus more time on domain science.

A pattern with applicability in various areas like adaptive mesh refinement, is the “Generator-Processor-Scan” pattern, an alternative to the classical “Stream Compaction” pattern. Both process an unknown number of elements in order with comparable performance, but the “Generator-Processor-Scan” can reuse already allocated memory and provide a significant performance improvement. The following figure shows the performance comparison of the “Generator-Processor-Scan” over the “Stream Compaction” when creating an ordered vector from a generator that produces multiple elements per index.



The talk will present three exemplary patterns, discuss their potential applications, and performance. Furthermore the performanceportablepatterns.github.io website will be introduced including how patterns can be contributed, and how they get evaluated.

This work is done as part of my Better Scientific Software (BSSw) fellowship.

References

1. Design patterns, 1st Edition. Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. Addison Wesley. October 1994.
2. Performance Portable Design Patterns – A Gallery and Place to Discuss Them in the Community. Jakob Bludau. March 17th, HPSFCon 2026, Chicago, IL. Retrieved 2026-05-18. https://hosted-files.sched.co/hpsf2026/e1/HPSEF_2026_Design_Patterns-3.pdf

Connection to Mission, Goals, & Interests of US-RSE Community

This talk mainly contributes to the US-RSE mission to provide the ability to share knowledge, professional connections, and resources to the community. The project presented in this talk is a resource for the RSE community to discuss and share knowledge about software design that works on any hardware used in the high-performance computing space which is leveraged by many research projects.

The performance portable design patterns collected and presented in the talk and website help disseminate expert knowledge and help research software engineers at all knowledge levels to write long term sustainable code. Furthermore, the presented design patterns facilitate teaching and help anyone tasked with developing or refactoring research software. Additionally, the discussion on the website connects expert RSEs and not-yet expert RSEs from diverse scientific backgrounds.