Distributed Performance-Portable Stencil Computation with Bricks

https://bricks.run/

Tuowen Zhao[1](ztuowen@cs.utah.edu), Protonu Basu[2], Samuel Williams[2], Mary Hall[1], Hans Johansen[2]
[1]University of Utah [2]Lawrence Berkeley National Lab

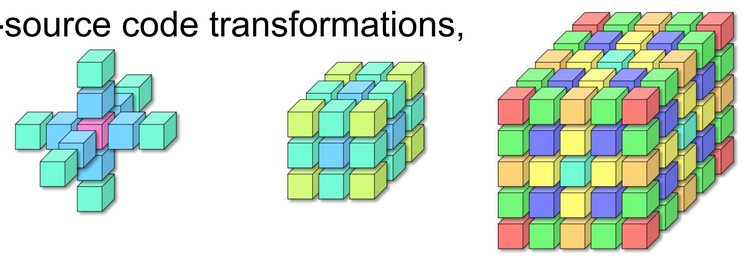
Introduction

Stencil computations are widely used in scientific applications to solve partial differential equations.

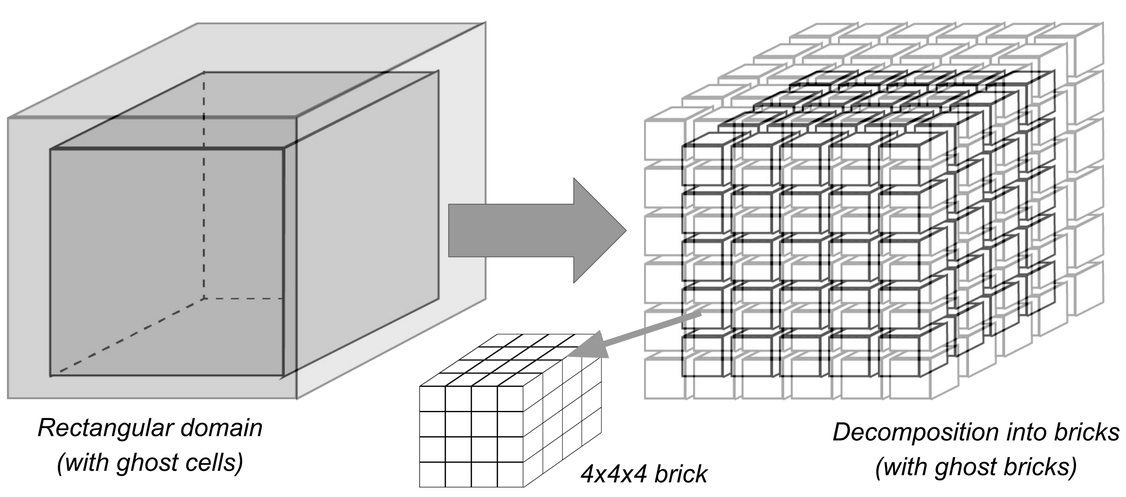
Bricks is a data layout and code generation framework, enable performance-portable stencil across Intel CPUs, Intel Knights Landing, Intel GPUs, NVIDIA GPUs. Vectorization abstraction enables easy support for other platforms, such as AMD GPUs (work-in-progress).

Performance portability enables application developers to target any supported platform using source-to-source code transformations, while achieving best-in-class performance on all platforms simultaneously.

Bricks uses a flexible data layout, which allows flexible domain shapes and enables fast “ghost cell” data communication performance through MPI.

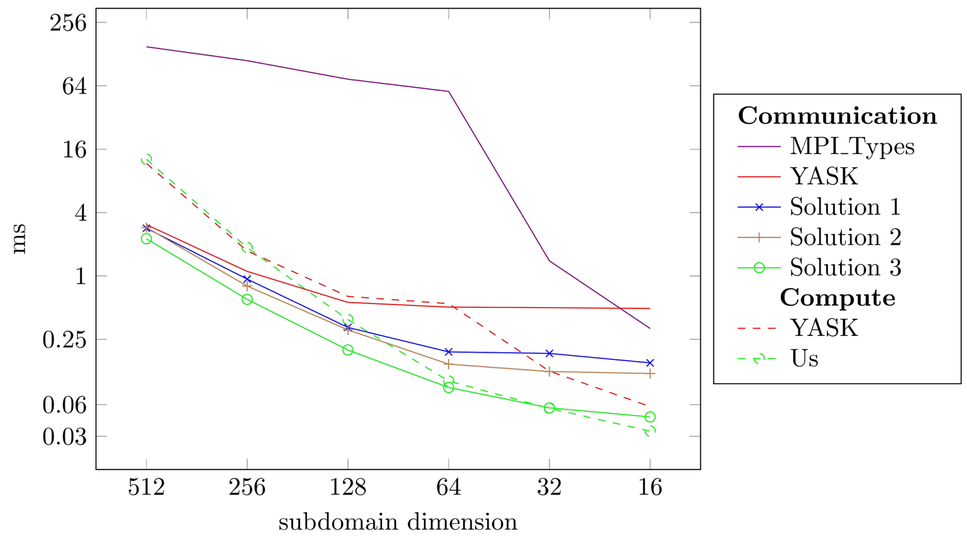


Brick Decompositions



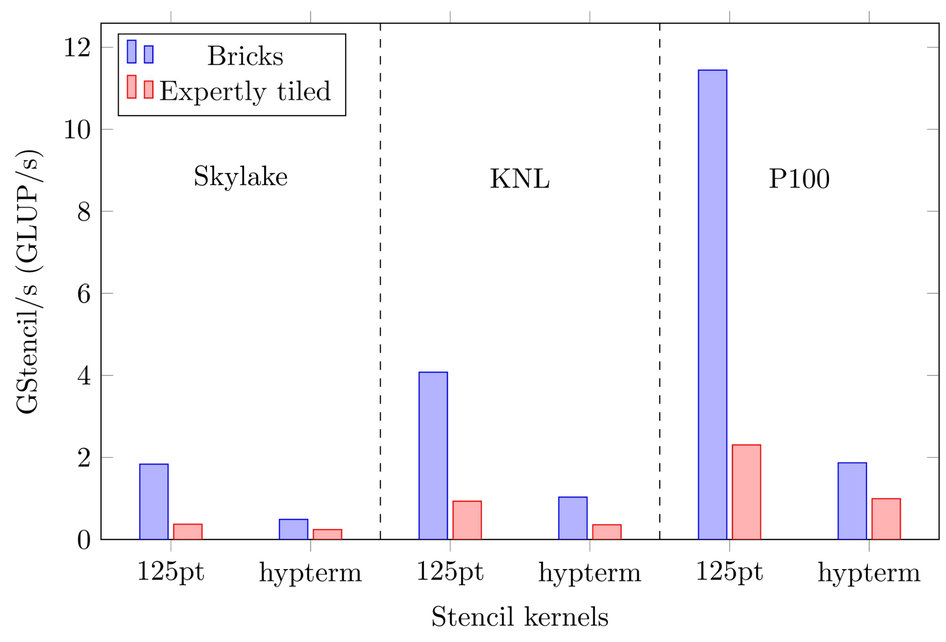
- Bricks are a tiled data layout
  - Decompose a grid domain into small, fixed-size subdomains
  - Improve data locality, nearby grid points are near in memory
  - Improve memory hierarchy and TLB usage by reducing the stride in memory accesses
- Stencil operations defined on bricks, each brick represents both
  - A unit of data, with a predefined local data layout
  - A unit of aggregated parallel-work
- Code generation optimizes with each brick for
  - Vectorization, reuse
  - Layout and streaming efficiency

Pack-free Communication with MPI



- Efficient ghost zone exchange achieves up to 10x faster than state-of-the-art YASK and up to 600x compared to cray-mpich/7.7.10 MPI_Types.
- Steady scaling of communication down to 16^3 subdomain per node.
- Eliminate communication inefficiencies in 2-level decomposition.
- Offer multiple solution suited to different compute environment for both CPUs and NVIDIA GPUs.

Performance Portability



- Well-suited to higher-order (bigger/wider) and complex stencils
- Achieve consistent 1.9x-4.9x speedup across different architectures including Skylake, Intel Knights Landing, and NVidia P100 GPUs
- Majority of kernel code is shared across all platforms

Using Brick Library and Codegen

Allocating and freeing of bricks are handled by the brick library. Functions available to create rectangular grids.

Stencil kernels are specified in Python and can be reused across all platforms.

For example, the script on the right shows a 5-point stencil. To use them, the following call in C++ will generate the 5-point stencil with CUDA for 16x16 bricks using 2x16 vectors for brick b and replace the call with the generated code.

```
brick("5pt.py", "CUDA", (16,16), (2,16), b);
```

Or for KNL using AVX512

```
brick("5pt.py", "AVX512", (16,16), (8), b);
```

```
from st.expr import Index, ConstRef
from st.grid import Grid

i = Index(0)
j = Index(1)

input = Grid("bIn", 2)
output = Grid("bOut", 2)

A = ConstRef("A")
B = ConstRef("B")

output(i, j, k).assign(
        A * input(i, j) +
        B * (input(i + 1, j) + input(i - 1, j) +
             input(i, j + 1, k) + input(i, j - 1, k))
    )
STENCIL = [output]
```

Future Work

Stencil Applications Welcome!

- Kernel decomposition for complex stencil expression
- Model-guided auto-tuning
- Sparse, chem, small solves
- Support next generation architectures

This research was supported by the Exascale Computing Project (17-SC-20-SC), a joint project of the U.S. Department of Energy's Office of Science and National Nuclear Security Administration. This research used resources of the Oak Ridge Leadership Facility at the Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725. This research used resources of the Argonne Leadership Computing Facility at Argonne National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under contract DE-AC02-06CH11357. This research used resources in Lawrence Berkeley National Laboratory and the National Energy Research Scientific Computing Center, which are supported by the U.S. Department of Energy Office of Science’s Advanced Scientific Computing Research program under contract number DE-AC02-05CH11231.