Impact of Graph Reordering on 3D Stencil SpMV Performance: A Preliminary Study using Kokkos

Nama Dosen
Dept. of Informatics
Affiliation Name
Indonesia
email@example.com

***Abstract*—Sparse Matrix-Vector Multiplication (SpMV) is a critical kernel in many scientific applications, including climate modeling and physical simulations based on 3D stencils. The performance of SpMV is often bounded by memory bandwidth and irregular memory access patterns. This study investigates the impact of graph reordering techniques (specifically METIS Nested Dissection) on the performance of SpMV for shuffled 3D 7-point stencil matrices using the Kokkos C++ performance portability framework. Our preliminary results on a multi-core CPU show that while reordering improves cache locality for smaller grids, the benefits diminish for large-scale grids (3.3 million nodes) due to memory bandwidth saturation, yielding a speedup of approximately 0.99x. We discuss the implications of these findings and outline future work for GPU-based hierarchical parallelism.**

***Index Terms*—SpMV, Kokkos, METIS, Graph Reordering, High Performance Computing, 3D Stencil**

I. Introduction

Structured grid simulations, such as 3D stencils used in climate modeling and fluid dynamics, theoretically possess regular memory access patterns. However, in real-world workflows involving adaptive mesh refinement or parallel partitioning, the node ordering often becomes unstructured (shuffled), leading to severe cache misses during SpMV operations [**?**].

This work attempts to recover the locality of such matrices using graph reordering algorithms. We utilize the Kokkos framework [**?**] to implement a portable SpMV kernel and integrate METIS [**?**] for finding optimal permutations.

II. Methodology

A. Data Generation: Shuffled 3D Stencil

We generate a synthetic 7-point stencil graph representing a 3D grid ($N_x \times N_y \times N_z$). To simulate worst-case input scenarios, we apply a random permutation P to the node IDs:

$$A_{shuffled} = P \cdot A_{grid} \cdot P^T \tag{1}$$

This destroys the spatial locality inherent in the grid structure.

B. Kokkos Implementation

We implemented the SpMV kernel using Kokkos `TeamPolicy` to support hierarchical parallelism suitable for future GPU porting.

```
// Simplified Kokkos Kernel
Kokkos::parallel_for("SpMV", policy_t(N, AUTO),
  KOKKOS_LAMBDA(const member_t& team) {
    int row = team.league_rank();
    double sum = 0.0;
    Kokkos::parallel_reduce(
      Kokkos::TeamThreadRange(team, len),
      [&](int k, double& lsum) {
        lsum += val(k) * x(col(k));
      }, sum);
});
```

C. Reordering Strategy

We apply METIS `NodeND` (Nested Dissection) to compute a permutation vector that minimizes the fill-in and implicitly clusters the non-zero elements, aiming to improve the vector x cache reuse.

III. Preliminary Results

Experiments were conducted on a standard workstation CPU. We varied the grid size from 80^3 (512k nodes) to 150^3 (3.3M nodes).

TABLE I
SpMV Performance (GFLOPs)

Grid Size	Baseline	METIS	Speedup
80^3	2.53 GFLOPs	2.30 GFLOPs	0.91x
150^3	0.41 GFLOPs	0.40 GFLOPs	0.99x

The results (Table I) indicate no speedup. For the large grid (150^3), the performance drop significantly (from 2.5 to 0.4 GFLOPs), suggesting the workload became main-memory bandwidth bound, rendering cache optimizations ineffective.

IV. Conclusion and Future Work

Reordering typical 3D stencils provides limited benefit on bandwidth-bound CPU architectures. However, we hypothesize that on throughput-oriented architectures like GPUs, where memory coalescing is critical, reordering will show significant gains.

Future work includes:

- Porting the `TeamPolicy` kernel to NVIDIA GPUs using `Kokkos::CudaSpace`.

- Investigating `Kokkos::LayoutLeft` impacts on coalesced access.
- Comparing METIS against faster reordering tools like Rabbit Order.