Performance Optimization of the HPCG Benchmark on the Sunway TaihuLight Supercomputer

Yulong Ao, Chao Yang, Lijuan Jiang, Fangfang Liu, Qiao Sun
Institute of Software, Chinese Academy of Sciences
High Performance Conjugate Gradients (HPCG)

- **Problem**: Poisson equation on a regular 3D grid discretized by a finite difference method.
- **Method**: conjugate gradient with multigrid preconditioner by using symmetric Gauss-Seidel smoother.
- **Feature**: related to unstructured problems, based on sparse matrix computation.

**Algorithm 1**: Preconditioned CG for \( Ax = b \)

**Input**: \( A, b, x_0, \text{it}_{max}, \varepsilon \)

\[
\begin{align*}
  r_0 & \leftarrow b - Ax_0 \\
  \text{for } i = 1, 2, \text{to } \text{it}_{max} \text{ do} & \\
  z_i & \leftarrow M^{-1} r_{i-1} \quad \triangleright \text{MG} \\
  s_i & \leftarrow (r_{i-1}, z_i) \quad \triangleright \text{DOT} \\
  \text{if } i = 0 \text{ then } p_i & \leftarrow z_i \\
  \text{else } p_i & \leftarrow z_i + (s_i / s_{i-1}) p_{i-1} \quad \triangleright \text{WAXPBY} \\
  \alpha_i & \leftarrow s_i / (p_i, A p_i) \quad \triangleright \text{SpMV, DOT} \\
  x_{i+1} & \leftarrow x_i + \alpha_i p_i \quad \triangleright \text{WAXPBY} \\
  r_i+1 & \leftarrow r_i - \alpha_i A p_i \quad \triangleright \text{WAXPBY} \\
  \text{if } ||r_{i+1}||_2/||r_0||_2 \leq \varepsilon \text{ then break} \quad \triangleright \text{DOT} \\
\end{align*}
\]

**Output**: \( x_{i+1} \)
Symmetric Gauss-Seidel: Block Multi-Coloring

1. Divide the matrix rows into blocks (64 rows for each) and label them by different colors (8 colors).

2. Compute the resulted blocks following the color order and distribute them evenly to the CPE cluster in the unit of parallel group.

3. Use the Athread library to spawn multithread (64 threads) for each parallel group (64 blocks).
Data layout Transformation

- Replace the default CSR format with the sliced ELL to allocate data in a bulk, aligned, continuous way.

- Reorder the matrix data (both value and index) as well as the vectors in consistency with the computation order of the multi-coloring.
Demand-based Data Access

1. Identify the data required by the current parallel group.
2. Map the current blocks and the required data onto the CPE cluster by DMA.
3. Get the data from other CPEs by the concurrent gather collective routine.

Basic idea: only map the required data block onto the CPE cluster and share them on chip by the register communication.
Concurrent Gather Collective: Request Stage

Step 1: Row broadcast

For example (col == 1):
1. CPEs in C1 column broadcast their packed request messages to their rows in parallel.
2. Other CPEs:
   a. Pack response messages for requests aim to them.
   b. Buffer request messages aim to other CPEs in their same columns.

Step 2: Column broadcast

For example (row == 1):
1. CPEs in R1 row broadcast their buffered request messages to their columns in parallel.
2. Other CPEs pack response messages base on request messages aim to them.

_request stage: each core can concurrently send the request messages to any CPE by two steps based on the register communication._

_request message format:_
<src core_id, src pos, dest core_id, dest pos>
Concurrent Gather Collective: Response Stage

Step 1: Row broadcast

For example (col == 1):
1. CPEs in C1 column broadcast the prepared response messages to their rows in parallel.
2. Other CPEs:
   a. Receive response messages aimed to them and store the values.
   b. Buffer response messages aimed to other CPEs in their same columns.

Step 2: Column broadcast

For example (row == 1):
1. CPEs in R1 row broadcast their buffered request messages to their columns in parallel.
2. Other CPEs pack response messages base on request messages aim to them.

Response message format:

<src core_id, src pos, dest core_id, val>
Classify the execution into different types of operations based on their properties.

Overlap the data access by the gather collective and computation within one parallel group and different parallel groups.
Many-core Acceleration

- **Performance:**
  - SpMV is better than SymGS; However, SymGS is closer to the final HPCG.
  - There is a strong correlation between the performance and the bandwidth.
  - Compared to MPE, the speedups for different optimizations are 1.9x (MEM), 9.6x (REG), 12.6 (+ASYNC), 14.4 (+MISC).
  - Performance: 3.45 Gflop/s; Fraction of the theoretical bandwidth: 77%.

- **Fraction of the bandwidth:**

---
**Scaling Results**

- **Problem size (each process):** $128 \times 128 \times 192$
- **Process number:** $64 \times 50 \times 50$ (160,000)
- **Performance:** 480.8 Tflop/s
- **Parallel efficiency:** 87.3%
Thank you