SAND2018-7221C

Thread Parallel Message Packing for Sparse Matrix MPI Communication

Sandia National Laboratories
T.J. Fuller and M. Hoemmen
Sandia National Laboratories, New Mexico 87123

Problem

In the Petra object model, maps describe the distribution of data in linear algebra objects, such as vectors and matrices. Operations requiring data across MPI ranks, such as finite element assembly, communicate data from one rank to another according to their distribution in the source and target ranks. Data are packed in a contiguous message buffer in a row-major format, as depicted in Figure 1. On the receiving side, the data must be unpacked from the message buffer at the completion of the communication phase. The amount of data scales as $N \times n$, where N is the number of rows on the MPI rank and n is the number of nonzeros per row.

$$[a_{ij}] = \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ a_{21} & \cdots & a_{2n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{bmatrix}$$

$$a = \{a_{11} \cdots a_{1n}\ a_{21} \cdots a_{2n} \cdots a_{n1} \cdots a_{nn}\}$$

Figure 1: Row-major contiguous message buffer packing. The sparse matrix $[a_{ij}]$ is packed sequentially along rows in the message buffer a for MPI communication.

Approach

Thread parallel message packing/unpacking is achieved using Kokkos parallel data structures and algorithms. The use of Kokkos allows the implementation to run with any of the threading models supported by Kokkos, including OpenMP and Cuda. In the message packing phase, the two pass `Kokkos::parallel_scan` is used to determine offsets for each matrix row in to the message buffer. Matrix data is packed in to the message buffer at the appropriate offset in a `Kokkos::parallel_for` pattern. The process for packing rows of the matrix is depicted in Figure 1.

The thread parallel `Kokkos::parallel_for` is an abstraction similar to `#pragma omp parallel for.` Pseudo-code for the message packing phase is shown in Listing 1.

Listing 1: Pseudo code for message packing phase.

```
Kokkos::parallel_for("Pack Matrix Rows", number_of_rows,
  KOKKOS_LAMBDA(const int row_num) {
    int offset = offsets(row_num);
    ...
    pack_row(message_buffer, values, column_ids, offset);}
);
```

In the unpacking phase, the `Kokkos::parallel_for` policy is used to unpack data from the message buffer directly in to the target data structure (CRS matrix).

Results

Distributed matrices were created with N=100 to N=100000 rows per MPI rank and n=50 and $n = 100$ nonzeros per row. Data from all rows were packed and unpacked using each approach and the results timed. The threaded version used 1 Nvidia P100 GPU per MPI rank.

As observed in Figure 2, threaded message packing is observed to be slower than the sequential version, despite the extra throughput afforded by the GPU. The slow down is attributed to extra work done in computing offsets in to the message buffer. A new algorithm that avoids this extra step is being developed.

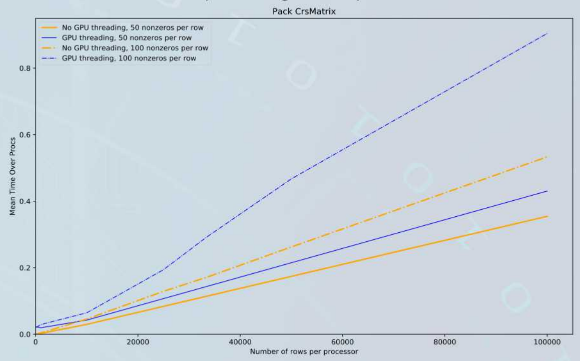

Figure 2: Message buffer packing time vs number of rows on the processor for 50 and 100 nonzeros per row.

Figure 3 demonstrates an up to 7x decrease in time to unpack for the threaded message buffer unpacking relative to the sequential. Note that message unpacking is a order of magnitude slower than packing, so that the improvement in message unpacking is more impactful on model problems.

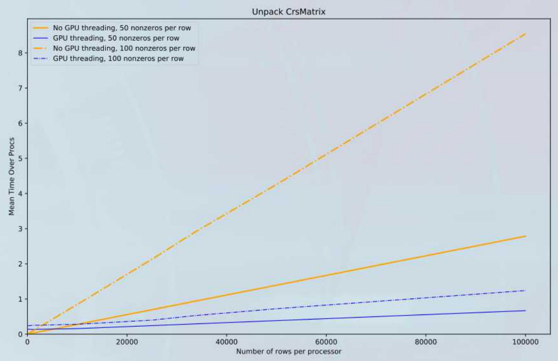

Figure 3: Message buffer unpacking time vs number of rows on the processor for 50 and 100 nonzeros per row.

Significance

MPI data communication plays an import role in finite element assembly. In an 8M element conduction problem, thread parallel MPI communication aided in obtaining a 1.5x decrease in assembly time on Intel Haswell archi-tecture with and without hyperthreads in Sandia's Aria flow simulator, as shown in Figure 4.

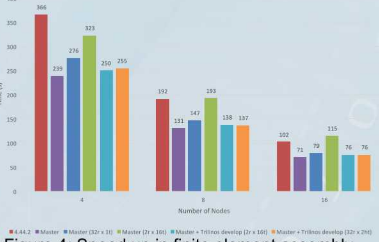

Figure 4: Speed up in finite element assembly time for 8M element finite element assembly

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC., a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under Contract DE-NA0003525.

SAND 2018-xxxP