TACC
TEXAS ADVANCED COMPUTING CENTER
WWW.TACC.UTEXAS.EDU

TEXAS
The University of Texas at Austin

Performance of MPI Sends of Non-Contiguous Data

Victor Eijkhout eijkhout@tacc.utexas.edu

IPDPS / SNACS 2020

TACC

Question

MPI derived data types are very convenient;
do they carry a performance penalty?

What are MPI derived types, and why

MPI send buffers are contiguous and single datatype;
derived types describe non-contiguous or heterogeneous data

Convenience:
defining and using a derived type is much easier than
allocate buffer, copy data, send, deallocate

```
MPI_Datatype newtype;
MPI_Type_something( .... description ... &newtype  );
MPI_Type_commit( newtype );
MPI_Send( buffer, 1, newtype, dest,tag,comm );
MPI_Type_free(&newtype);
```

Simple case

Send strided data;

```
float memory[2*N],buffer[N];
for (int i=0; i<N; i++)
  buffer[i] = memory[2*i];
MPI_Send( buffer,N,MPI_FLOAT, dest,tag,comm );
```

Simple performance model

Sending N words means

- read from memory to buffer;
- send buffer, probably overlapped with read

$\Rightarrow$ time $O(N)$.

Strided data:

- Read $2N$ from memory;
- write back, probably overlapped;
- Read N from memory to buffer,
- send buffer, probably overlapped

$\Rightarrow$ time $O(3N)$.

Expectation: strided send $1/3$ performance of contiguous send.

Can we go faster than 1/3 peak?

Yes, but only with hardware support for direct streaming from non-contiguous locations.

> *M. Li, H. Subramoni, K. Hamidouche, X. Lu, and D. K. Panda. High performance mpi datatype support with user-mode memory registration: Challenges, designs, and benefits. In* 2015 IEEE International Conference on Cluster Computing*, pages 226–235, Sept 2015.*

Why would performance be lower than 1/3 peak?

- Construction of the datatype
- Index calculation overhead during copy

but in practice mostly:

- allocation of internal MPI buffers.

Big problem: an application can allocate a buffer and reuse,
but MPI needs to allocate, free, re-allocate, free, re-re-allocate,. . .

Simple use of derived types

Type vector and type subarray:
same performance, so probably no overhead in index calculations.
lower than $1/3$ performance: internal buffer allocation

Force MPI to maintain a buffer

- Buffered sends: $1/3$ performance as predicted
- Persistend sends: lower performance, reason unclear.

One-sided communication

Could have the same performance as other non-buffer schemes.

In practice:
much worse performance for small messages (overhead),
sometimes for large messages (reason unclear)

Packing

Use `MPI_Pack` to copy elements:
one function call per element, so very slow

Pack one derived data:
in practice attains 1/3 peak performance
because reused buffer in user space.

Findings

- Performance often as expected; lower than hardware peak by $1/3$
- Disappointments: persistend sends and one-sided performed worse than expected.
  Exception: one-sided on the new Intel UCX layer seems improved.
- The processor is important next to the network:
  Stampede2 has Knights Landing and Skylake processors on the same network
  The lower scalar performance of KNL is especially noticable on small messages.

Recommendations

- Use buffered sends if the calculation of buffer space is doable and not excessive.
  (Note: total space of *all* simultaneously outstanding `MPI_Bsend` calls.)
- Otherwise pack derived type.

Repository of code and results

https://github.com/TACC/mpipacking