Performance of MPI derived types

PRESENTED BY:
Victor Eijkhout
eijkhout@tacc.utexas.edu

Problem statement

- MPI has derived data types (vector, indexed, struct, subarray), which are convenient for sending non-contiguous buffers
- People always wonder: is there a performance penalty?
- This is an experimental study of 8 different mechanisms for sending non-contiguous data
- For simplicity:
  send every other element of a double precision array
  between two processes on two different nodes
- We time a ping-pong with zero byte "pong". Average of 10 trials.

Method 1: manual copying

- Copy element to a contiguous buffer, then ordinary send.
- Cost: read 2N elements from memory,
  write N elements
  read & send N elements (probably pipelined)
- We expect 3x slower than sending N elements contiguously.
- (We also try Bsend, to avoid internal buffering issues)

Method 2: derived datatypes

- This is regularly strided data, so:
  type Vector, and
  type Subarray
  are the candidates
- Since there is no explicit copy, this should be pipelineable
  (see Subramony/Lu/Panda 2015: with hardware support it pipelines

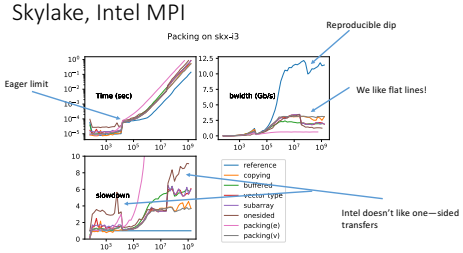
Method 3: one-sided transfer

- No "pong" needed so potentially faster
- Two schemes:
  one Put call per element; likely very slow
  Put call for derived datatype: this is the only we tested
- Use Win_fence for synchronization,
  with two processes this should be very efficient.
  Famoust last words.

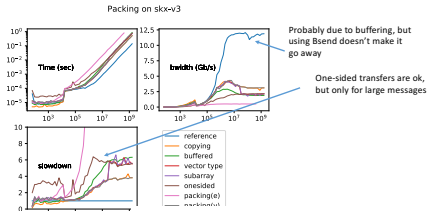
Method 4: packing

- MPI_Pack packs a buffer in user space
- Two schemes:
  pack individual elements: one call per element, likely slow
  pack derived data type: this is very much like manual packing, just more convenient

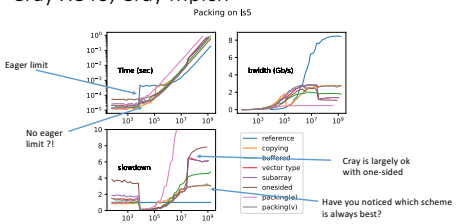
Skylake, Intel MPI



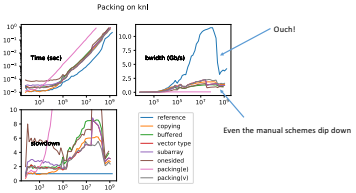
Skylake, mvapich2



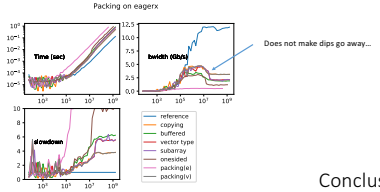
Cray XC40, Cray mpich



Knights landing, Intel MPI



Unlimited Eager



Experimental setup

- Stampede2, Skylake, both Intel MPI (update 19) and MVAPICH2 (2.3)
- Stampede2, Knights Landing
- Lonestar 5 (Cray XC40) with Cray Mpich (7.7)
- Size 1byte – 100Mbyte
  All arrays 64-byte aligned, arrays zeroed before use
- L2 cache is flushed before every ping-pong
- Report:
  - Time
  - Bandwidth
  - Slowdown over contiguous send (note: 3x expected)

Conclusions

- Derived datatypes have no intrinsic performance penalty
- Nothing beats copying-by-hand
  ..... but packing a derived type is as fast, and more convenient
- One-sided transfers have a limited range of applicability
- Packing per element is of course slooooooooow.....
- All schemes are affected by buffering phenomena at large scale
- Buffered send does not solve anything.