# Performance Impact of MPI Options on RMA

Keith D. Underwood, Brian Barrett

# Point of this Presentation

- There are many complaints about MPI one-sided operations
  - Semantics
    - Lack of progress
    - Synchronization
      - Overhead of current synchronization
      - Passive target's write completion semantics
  - Atomic operations
    - Compare and swap
    - Fetch and increment
  - ***Performance***
- ***This presentation only discusses the last one***
  - And only a narrow subset of it at that
  - *What is the overhead associated with MPI_Put?*

# What is the overhead for MPI_Put?

- We attempt to provide some measurement of the overhead associated with MPI_Put()
  - We focus on what is possible, ***not what is done***
  - We ignore semantic issues and the impact of those
- As a reminder:

```
int MPI_Put(void *origin_addr, int origin_count,
  MPI_Datatype  origin_datatype, int target_rank,
  MPI_Aint target_disp, int target_count,
  MPI_Datatype target_datatype, MPI_Win win)
```

- Several aspects that can impact performance
  - Long argument list
  - Dual datatype decoding
  - Windows (pointer indirection)
  - Target displacement (array lookup)

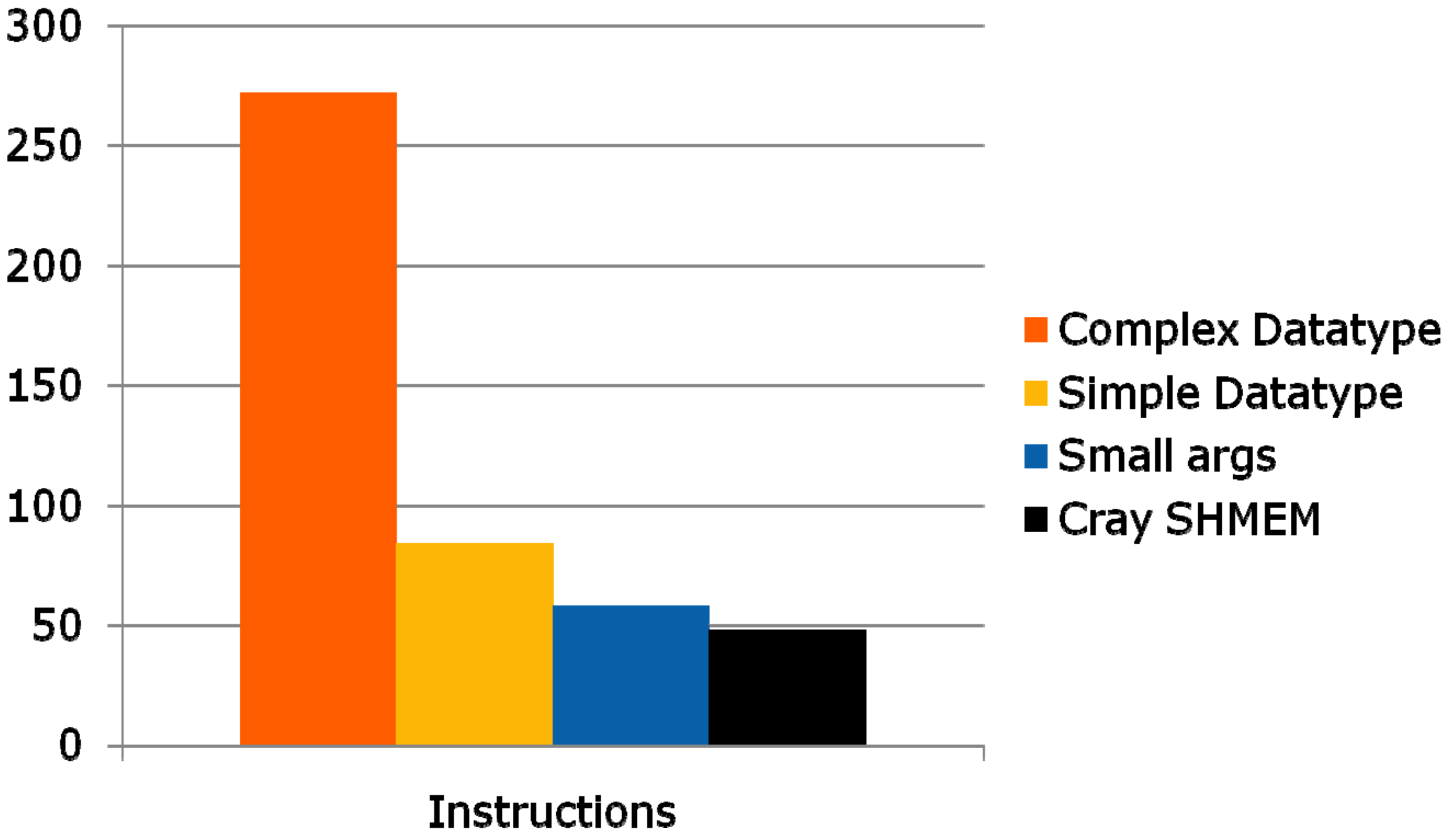
# Experiment

- Compare instruction count and message rate
- Memory-to-memory copy for data move (based on SMARTMAP)
- Comparisons:
  - Complex Datatype: Transfer based on a “complex” datatype which requires decoding
  - Simple Datatype: Transfer based on simple datatype, with information encoded in datatype
  - Small args: Assume origin and target datatypes are the same (and simple). Eliminates origin and target datatypes from the call
  - SHMEM: Cray SHMEM implementation over SMARTMAP

# Results

| Test | Instructions | Message Rate (million messages/sec) |
|---|---|---|
| Complex Datatype | 272 | 10.16 |
| Simple Datatype | 84 | 43.07 |
| Small args | 58 | 53.55 |
| Cray SHMEM | 48 | 59.38 |

# Instructions



# Message Rate