­­­Introduction

* HPC system intro
* MPI intro
* Datatype intro
  + Datatype description
  + Ease coding on data movement
  + Support communication layer
  + Inconsistent optimization leads to inconsistent performance
    - Caused by user define same datatype using different methods
    - Introduce iovec implmentation
* datatype engine with different datatype description and data access pattern to improve performance (bandwidth)
  + introduced gather implementation, which improves performance by exploiting hardware prefetch and cache reuse
  + introduce pipelining in gather to minimize cache emit and maximize cache reuse

Background

* Datatype engine
  + Current datatype description in master
  + Support communication layer
    - Constant network latency
    - pipelining to hide network latency
    - Graph and equation to show how network latency is hidden
* Commit time optimization
  + Combine elements with no gap in between
  + Group elements using loops
  + Reduce datatype description size

Motives

* memory wall explanation, want to get as close as to the theoretical peak
  + theoretical peak = sparsity(data / extent) \* theoretical peak
* Eliminate inconsistent performance
  + Example
* With pipelining hiding network latency, improving pack/unpack performance is the only way to improve performance

Test platforms and datatypes and methods

* Fugaku?
* Alembert
* Datatypes
  + Doubles on different cache lines
  + Trash\_tlb
* Store rdtscp into memory in packing

Iovec datatype description

* Describe iovec datatype description
* Consistent datatype description and performance
  + Wont be affected by how user defines datatype
  + example
* Some datatype results in smaller datatype description
  + However, some datatype results in huge datatype description
  + Access huge description leads to poor performance
    - Example: vector( (huge #), 1, 8, double )
  + Thus eliminating description access will result in better performance
    - Introduce iovec gather

Iovec gather

* Gather implementation explanation
* Eliminate constant access for datatype description
* Reduce irregular memory accesses, more regular memory accesses
* Helped by hardware prefetch, since regular memory access
  + However, performance drops dramatically for large sizes
  + Caused by constantly emitting cache with data still waiting to be moved
    - Introduce iovec gather pipeline

Iovec gather pipeline

* Gather pipeline explanation
* To eliminate constant cache replacement
  + Reuse cache
* Hardware prefetch, regular access, maximum cache reuse, minimum description access

Performance

* Master vs iovec vs iovec\_gather
* Simple gather with rdtscp stored in packed data
  + Examine the access differences between current packing versus iovec and iovec\_gather
* Difference pipeline # in gather vs iovec vs current pack

Future work

* Find the optimal pipeline # for gather
* Find how much gap is needed to gather to perform better than master
* Unpack gather