YGM

YGM Developers

CONTENTS:

1	Getting Started	1		
	1.1 What is YGM?	1		
	1.2 General YGM Operations	1		
	1.3 Requirements	2		
	1.4 Using YGM with CMake	2		
	1.5 License	3		
	1.6 Release	3		
2	ygm::comm class reference.	5		
3	Communicator Overview			
4	Communicator Hello World			
5	ygm::container module reference.			
6	Implemented Storage Containers	15		
7	Typical Container Operations	17		
	7.1 for_all Operations	17		
	7.2 async_ Operations	17		
8	YGM Container Example	39		
9	Container Transformation Objects	41		
10	Documents for YGM developers	43		
	10.1 Developing YGM	43		
11	Indices and tables	45		
Ind	dex	47		

ONE

GETTING STARTED

1.1 What is YGM?

YGM is an asynchronous communication library written in C++ and designed for high-performance computing (HPC) use cases featuring irregular communication patterns. YGM includes a collection of distributed-memory storage containers designed to express common algorithmic and data-munging tasks. These containers automatically partition data, allowing insertions and, with most containers, processing of individual elements to be initiated from any running YGM process.

Underlying YGM's containers is a communicator abstraction. This communicator asynchronously sends messages spawned by senders with receivers needing no knowledge of incoming messages prior to their arrival. YGM communications take the form of *active messages*; each message contains a function object to execute (often in the form of C++ lambdas), data and/or pointers to data for this function to execute on, and a destination process for the message to be executed at.

YGM also includes a set of I/O primitives for parsing collections of input documents in parallel as independent lines of text and streaming output lines to large numbers of destination files. Current parsing functionality supports reading input as CSV, ndjson, and unstructured lines of data.

1.2 General YGM Operations

YGM is built on its ability to communicate active messages asynchronously between running processes. This does not capture every operation that can be useful, for instance collective operations are still widely needed. YGM uses prefixes on function names to distinguish their behaviors in terms of the processes involved. These prefixes are:

- async_: Asynchronous operation initiated on a single process. The execution of the underlying function may occur on a remote process.
- local_: Function performs only local operations on data of the current process. In uses within YGM containers with partitioning schemes that determine item ownership, care must be taken to ensure the process a local_ operation is called from aligns with the item's owner. For instance, calling ygm::container::map::local_insert will store an item on the process where the call is made, but the ygm::container::map may not be able to look up this location if it is on the wrong process.
- No Prefix: Collective operation that must be called from all processes.

The primary workhorse functions in YGM fall into the two categories of async_ and for_all operations. In an async_ operation, a lambda is asynchronously sent to a (potentially) remote process for execution. In many cases with YGM containers, the lambda being executed is not provided by the user and is instead part of the function itself, e.g. async_insert calls on most containers. A for_all operation is a collective operation in which a lambda is executed locally on every process while iterating over all locally held items of some YGM object. The items iterated over can be

items in a YGM container, items coming from a map, filter, or flatten applied to a container, or all lines in a collection of files in a YGM I/O parser.

1.2.1 Lambda Capture Rules

Certain async_ and for_all operations require users to provide lambdas as part of their executions. The lambdas that can be accepted by these two classes of functions follow different rules pertaining to the capturing of variables:

- async_ calls cannot capture (most) variables in lambdas. Variables necessary for lambda execution must be provided as arguments to the async_ call. In the event that the data for the lambda resides on the remote process the lambda will execute on, a ygm::ygm_ptr should be passed as an argument to the async_.
- for_all calls assume lambdas take only the arguments inherently provided by the YGM object being iterated over. All other necessary variables *must* be captured. The types of arguments provided to the lambda can be identified by the for_all_args type within the YGM object.

These differences in behavior arise from the distinction that async_lambdas may execute on a remote process, while for_all lambdas are guaranteed to execute locally to a process. In the case of async_operations, the lambda and all arguments must be serialized for communication, but C++ does not provide a method for inspection of variables captured in the closure of a lambda. In the case of for_all operations, the execution is equivalent to calling std::for_each on entire collection of items held locally.

1.3 Requirements

- C++20 GCC versions 11 and 12 are tested. Your mileage may vary with other compilers.
- Cereal C++ serialization library
- MPI
- Optionally, Boost 1.77 to enable Boost.JSON support.

1.4 Using YGM with CMake

YGM is a header-only library that is easy to incorporate into a project through CMake. Adding the following to CMakeLists.txt will install YGM and its dependencies as part of your project:

(continues on next page)

(continued from previous page)

```
set(YGM_INSTALL ON)
    FetchContent_Populate(ygm)
    add_subdirectory(${ygm_SOURCE_DIR} ${ygm_BINARY_DIR})
    message(STATUS "Cloned ygm dependency " ${ygm_SOURCE_DIR})
    endif ()
else ()
    message(STATUS "Found installed ygm dependency " ${ygm_DIR})
endif ()
```

1.5 License

YGM is distributed under the MIT license.

All new contributions must be made under the MIT license.

See LICENSE-MIT, NOTICE, and COPYRIGHT for details.

SPDX-License-Identifier: MIT

1.6 Release

LLNL-CODE-789122

1.5. License 3

СНАРТ	ΓER
TV	VO

YGM::COMM CLASS REFERENCE.

THREE

COMMUNICATOR OVERVIEW

The communicator ygm::comm is the central object in YGM. The communicator controls an interface to an MPI communicator, and its functionality can be modified by additional optional parameters.

Communicator Features:

- Message Buffering Increases application throughput at the expense of increased message latency.
- Message Routing Extends benefits of message buffering to extremely large HPC allocations.
- **Fire-and-Forget RPC Semantics** A sender provides the function and function arguments for execution on a specified destination rank through an *async* call. This function will complete on the destination rank at an unspecified time in the future, but YGM does not explicitly make the sender aware of this completion.

FOUR

COMMUNICATOR HELLO WORLD

Here we will walk through a basic "hello world" YGM program. The [examples directory](/examples/) contains several other examples, including many using YGM's storage containers.

To begin, headers for a YGM communicator are needed:

```
#include <ygm/comm.hpp>
```

At the beginning of the program, a YGM communicator must be constructed. It will be given argc and argv like MPI_Init.

```
ygm::comm world(&argc, &argv);
```

Next, we need a lambda to send through YGM. We'll do a simple hello_world type of lambda.

```
auto hello_world_lambda = [](const std::string &name) {
        std::cout << "Hello " << name << std::endl;
};</pre>
```

Finally, we use this lambda inside of our *async* calls. In this case, we will have rank 0 send a message to rank 1, telling it to greet the world

```
if (world.rank0()) {
      world.async(1, hello_world_lambda, std::string("world"));
}
```

The full, compilable version of this example is found here. Running it prints a single "Hello world".

class comm

Public Functions

```
inline comm(int *argc, char ***argv)
inline comm(MPI_Comm comm)
inline ~comm()
inline void welcome(std::ostream &os = std::cout)
    Prints a welcome message with configuration details.
inline void stats_reset()
```

```
inline void stats_print(const std::string &name = "", std::ostream &os = std::cout)
template<typename AsyncFunction, typename ...SendArgs>
inline void async(int dest, AsyncFunction fn, const SendArgs&... args)
template<typename AsyncFunction, typename ...SendArgs>
inline void async(int dest, AsyncFunction fn, const SendArgs&... args) const
template<typename AsyncFunction, typename ...SendArgs>
inline void async_bcast(AsyncFunction fn, const SendArgs&... args)
template<typename AsyncFunction, typename ...SendArgs>
inline void async_mcast(const std::vector<int> &dests, AsyncFunction fn, const SendArgs&... args)
inline void cf_barrier() const
     Control Flow Barrier Only blocks the control flow until all processes in the communicator have called it.
     See: MPI_Barrier()
inline void barrier()
     Full communicator barrier.
inline void barrier() const
inline void local_progress()
     Checks for incoming unless called from receive queue and flushes one buffer.
inline bool local_process_incoming()
template<typename Function>
inline void local_wait_until(Function fn)
     Waits until provided condition function returns true.
         Template Parameters
             Function -
         Parameters
             fn – Wait condition function, must match []() -> bool
template<typename T>
inline ygm_ptr < T > make_ygm_ptr(T &t)
inline void register_pre_barrier_callback(const std::function<void()> &fn)
     Registers a callback that will be executed prior to the barrier completion.
         Parameters
             fn – callback function
template<typename T>
inline T all_reduce_sum(const T &t) const
template<typename T>
inline T all_reduce_min(const T &t) const
template<typename T>
inline T all_reduce_max(const T &t) const
template<typename T, typename MergeFunction>
```

```
Tree based reduction, could be optimized significantly.
         Template Parameters
             • T -
             • MergeFunction -
         Parameters
             • in -
             • merge -
         Returns
             Т
inline int size() const
inline int rank() const
inline MPI_Comm get_mpi_comm() const
inline const detail::layout &layout() const
inline const detail::comm router &router() const
inline bool rank0() const
template<typename T>
inline void mpi_send(const T &data, int dest, int tag, MPI_Comm comm) const
template<typename T>
inline void mpi_send(const T &data, int dest, int tag) const
template<typename T>
inline T mpi_recv(int source, int tag, MPI_Comm comm) const
template<typename T>
inline T mpi_recv(int source, int tag) const
template<typename T>
inline T mpi_bcast(const T &to_bcast, int root, MPI_Comm comm) const
template<typename T>
inline T mpi_bcast(const T &to bcast, int root) const
inline std::ostream &cout0() const
inline std::ostream &cerr0() const
inline std::ostream &cout() const
inline std::ostream &cerr() const
template<typename ... Args>
inline void cout (Args&&... args) const
template<typename ...Args>
```

inline T all_reduce(const T &t, MergeFunction merge) const

```
inline void cerr(Args&&... args) const
template<typename ... Args>
inline void \mathbf{cout0}(Args\&\&... args) const
template<typename ... Args>
inline void cerr0 (Args&&... args) const
Friends
friend class detail::interrupt_mask
friend class detail::comm_stats
struct header_t
     Public Members
     uint32_t message_size
     int32_t dest
struct mpi_irecv_request
     Public Members
     std::shared_ptr<ygm::detail::byte_vector> buffer
     MPI_Request request
struct mpi_isend_request
     Public Members
     std::shared_ptr<ygm::detail::byte_vector> buffer
     MPI_Request request
```

FIVE

YGM:: CONTAINER MODULE REFERENCE.

ygm::container is a collection of distributed containers designed specifically to perform well within YGM's asynchronous runtime. Inspired by C++'s Standard Template Library (STL), the containers provide improved programmability by allowing developers to consider an algorithm as the operations that need to be performed on the data stored in a container while abstracting the locality and access details of said data. While insiration is taken from STL, the top priority is to provide expressive and performant tools within the YGM framework.

IMPLEMENTED STORAGE CONTAINERS

The currently implemented containers include a mix of distributed versions of familiar containers and distributed-specific containers:

- ygm::container::bag An unordered collection of objects partitioned across processes. Ideally suited for iteration over all items with no capability for identifying or searching for an individual item within the bag.
- ygm::container::set Analogous to std::set. An unordered collection of unique objects with the ability to iterate and search for individual items. Insertion and iteration are slower than a ygm::container::bag.
- ygm::container::multiset Analogous to std::multiset. A set where multiple instances of the same object may appear.
- ygm::container::map Analogous to std::map. A collection of keys with assigned values. Keys and values can be inserted and looked up individually or iterated over collectively.
- ygm::container::multimap Analogous to std::multimap. A map where keys may appear with multiple values.
- ygm::container::array A collection of items indexed by an integer type. Items can be inserted and looked up by their index values independently or iterated over collectively. Differs from a std::array in that sizes do not need to known at compile-time, and a ygm::container::array can be dynamically resized through a (potentially expensive) function at runtime.
- ygm::container::counting_set A container for counting occurrences of items. Can be thought of as a ygm::container::map that maps items to integer counts but optimized for the case of frequent duplication of keys.
- ygm::container::disjoint_set A distributed disjoint set data structure. Implements asynchronous union operation for maintaining membership of items within mathematical disjoint sets. Eschews the find operation of most disjoint set data structures and instead allows for execution of user-provided lambdas upon successful completion of set merges.

SEVEN

TYPICAL CONTAINER OPERATIONS

Most interaction with containers occurs in one of two classes of operations: for_all and async_.

7.1 for_all Operations

for_all-class operations are barrier-inducing collectives that direct ranks to iteratively apply a user-provided function to all locally-held data. Functions passed to the for_all interface do not support additional variadic parameters. However, these functions are stored and executed locally on each rank, and so can capture objects in rank-local scope.

7.2 async_ Operations

Operations prefixed with async_ perform operations on containers that can be spawned from any process and execute on the correct process using YGM's asynchronous runtime. The most common *async* operations are:

- async_insert Inserts an item or a key and value, depending on the container being used. The process responsible for storing the inserted object is determined using the container's partitioner. Depending on the container, this partitioner may determine this location using a hash of the item or by heuristics that attempt to evenly spread data across processes (in the case of ygm::container::bag).
- async_visit Items within YGM containers will be distributed across the universe of running processes.
 Instead of providing operations to look up this data directly, which would involve a round-trip communication with the process storing the item of interest, most YGM containers provide async_visit. A call to async_visit takes a function to execute and arguments to pass to the function and asynchronously executes the provided function with arguments that are the item stored in the container and the additional arguments passed to async_visit.

Specific containers may have additional async_ operations (or may be missing some of the above) based on the capabilities of the container. Consult the documentation of individual containers for more details.

7.2.1 array

template<typename **Value**, typename **Index** = size_t>

class **array**: public ygm::container::detail::base_async_insert_key_value<array<Value, size_t>, std::tuple<size_t, Value>>, public ygm::container::detail::base_misc<array<Value, size_t>, std::tuple<size_t, Value>>, public ygm::container::detail::base_async_visit<array<Value, size_t>, std::tuple<size_t, Value>>, public ygm::container::detail::base_iteration_key_value<array<Value, size_t>, std::tuple<size_t, Value>>, public ygm::container::detail::base_iteration_key_value<array<Value, size_t>, std::tuple<size_t, Value>>, public ygm::container::detail::base_async_reduce<array<Value, size_t>, std::tuple<size_t, Value>>

```
using self_type = array<Value, Index>
using mapped_type = Value
using key_type = Index
using size_type = Index
using for_all_args = std::tuple<Index, Value>
using container_type = ygm::container::array_tag
using ptr_type = typename ygm::ygm_ptr<self_type>
Public Functions
array() = delete
inline array(ygm::comm &comm, const size_type size)
inline array(ygm::comm &comm, const size_type size, const mapped_type &default_value)
inline array(ygm::comm &comm, std::initializer_list<mapped_type> 1)
inline array(ygm::comm &comm, std::initializer_list<std::tuple<key_type, mapped_type>> 1)
inline array (const self_type &rhs)
template<typename T>
inline array(ygm::comm &comm, const T &t)
inline ~array()
inline void local_insert(const key_type &key, const mapped_type &value)
template<typename Function, typename ...VisitorArgs>
inline void local_visit(const key_type index, Function &fn, const VisitorArgs&... args)
```

```
inline void async_set(const key_type index, const mapped_type &value)
template<typename BinaryOp>
inline void async_binary_op_update_value(const key_type index, const mapped_type &value, const
                                             BinaryOp &b)
inline void async_bit_and(const key_type index, const mapped_type &value)
inline void async_bit_or(const key_type index, const mapped_type &value)
inline void async_bit_xor(const key_type index, const mapped_type &value)
inline void async_logical_and(const key_type index, const mapped_type &value)
inline void async_logical_or(const key_type index, const mapped_type &value)
inline void async_multiplies(const key_type index, const mapped_type &value)
inline void async_divides(const key_type index, const mapped_type &value)
inline void async_plus(const key_type index, const mapped_type &value)
inline void async_minus (const key type index, const mapped type &value)
template<typename UnaryOp>
inline void async_unary_op_update_value(const key_type index, const UnaryOp &u)
inline void async_increment(const key type index)
inline void async_decrement(const key_type index)
const mapped_type &default_value() const
inline void resize(const size_type size, const mapped_type &fill_value)
inline void resize(const size_type size)
inline size_t local_size()
inline size_t size() const
inline void local_clear()
inline void local_swap(self_type &other)
template<typename Function>
inline void local_for_all(Function fn)
template<typename ReductionOp>
inline void local_reduce(const key_type index, const mapped_type &value, ReductionOp reducer)
inline void sort()
inline void async_insert (const typename std::tuple_element<0, for_all_args>::type &key, const typename
                          std::tuple_element<1, for_all_args>::type &value)
inline void async_insert(const std::pair<const typename std::tuple_element<0, for_all_args>::type,
                          typename std::tuple_element<1, for_all_args>::type> &kvp)
inline void clear()
```

```
inline void swap(derived_type &other)
inline ygm::comm &comm()
inline const ygm::comm &comm() const
inline ygm::ygm_ptr<derived_type> get_ygm_ptr()
inline const ygm::ygm_ptr<derived_type> get_ygm_ptr() const
template<typename Visitor, typename ...VisitorArgs>
inline void async_visit(const std::tuple_element<0, for_all_args>::type &key, Visitor visitor, const
                         VisitorArgs&... args)
template<typename Visitor, typename ... VisitorArgs>
inline void async_visit_if_contains(const std::tuple_element<0, for_all_args>::type &key, Visitor
                                       visitor, const VisitorArgs&... args)
template<typename Visitor, typename ... VisitorArgs>
inline void async_visit_if_contains(const std::tuple element<0, for all args>::type &key, Visitor
                                       visitor, const VisitorArgs&... args) const
template<typename Function>
inline void for_all(Function fn)
template<typename Function>
inline void for_all(Function fn) const
template<typename STLContainer>
inline void gather(STLContainer &gto, int rank) const
template<typename Compare = std::greater<std::pair<key type, mapped type>>>
inline std::vector<std::pair<key_type, mapped_type>> gather_topk(size_t k, Compare comp = Compare())
                                                                const
template<typename YGMContainer>
inline void collect(YGMContainer &c) const
template<typename MapType, typename ReductionOp>
inline void reduce_by_key(MapType &map, ReductionOp reducer) const
template<typename TransformFunction>
transform_proxy_key_value<derived_type, TransformFunction> transform(TransformFunction ffn)
inline auto keys()
inline auto values()
inline flatten_proxy_key_value<derived_type> flatten()
template<typename FilterFunction>
filter_proxy_key_value<derived_type, FilterFunction> filter(FilterFunction ffn)
template<typename ReductionOp>
inline void async_reduce(const typename std::tuple_element<0, for_all_args>::type &key, const typename
                          std::tuple_element<1, for_all_args>::type &value, ReductionOp reducer)
```

Public Members

```
detail::block_partitioner< key_type> partitioner
```

Friends

friend class detail::base misc< array< Value, Index >, std::tuple< Index, Value > >

7.2.2 bag

template<typename Item>

```
class bag: public ygm::container::detail::base_async_insert_value<bag<Item>, std::tuple<Item>>, public ygm::container::detail::base_count<bag<Item>, std::tuple<Item>>, public ygm::container::detail::base_misc<bag<Item>, std::tuple<Item>>, public ygm::container::detail::base_iteration_value<bag<Item>, std::tuple<Item>>
```

Public Types

```
using self_type = bag<ltem>

using value_type = Item

using size_type = size_t

using for_all_args = std::tuple<Item>

using container_type = ygm::container::bag_tag

Public Functions

inline bag(ygm::comm &comm)

inline bag(ygm::comm &comm, std::initializer_list<Item>1)

template<typename STLContainer>
inline bag(ygm::comm &comm, const STLContainer &cont)

template<typename YGMContainer>
inline bag(ygm::comm &comm, const YGMContainer &yc)

inline bag(ygm::comm &comm, const YGMContainer &yc)

inline bag(const self_type &other)
```

inline **bag**(*self_type* &&other) noexcept

```
inline bag & operator=(const self_type & other)
inline bag & operator=(self_type & & other) noexcept
inline void async_insert(const Item &value, int dest)
inline void async_insert(const std::vector< Item> & values, int dest)
inline void local_insert(const Item &val)
inline void local_clear()
inline size_t local_size() const
inline size_t local_count(const value_type &val) const
template<typename Function>
inline void local_for_all(Function fn)
template<typename Function>
inline void local_for_all(Function fn) const
inline void serialize(const std::string &fname)
inline void deserialize(const std::string &fname)
inline void rebalance()
template<typename RandomFunc>
inline void local_shuffle(RandomFunc &r)
inline void local_shuffle()
template<typename RandomFunc>
inline void global_shuffle(RandomFunc &r)
inline void global_shuffle()
inline void async_insert(const typename std::tuple_element<0, std::tuple<Item>>::type &value)
inline size_t count (const std::tuple_element<0, std::tuple<ltem>>::type &value) const
inline size_t size() const
inline void clear()
inline void swap(bag<Item> &other)
inline ygm::comm &comm()
inline const ygm::comm &comm() const
inline ygm::ygm_ptr<bag<ltem>> get_ygm_ptr()
inline const ygm::ygm_ptr<br/>
bag<ltem>> get_ygm_ptr() const
inline void for_all(Function fn)
inline void for_all(Function fn) const
```

```
inline void gather(STLContainer &gto, int rank) const
inline std::vector<value_type> gather_topk(size_t k, Compare comp = std::greater<value_type>()) const
inline value_type reduce(MergeFunction merge) const
inline void collect(YGMContainer &c) const
inline void reduce_by_key(MapType &map, ReductionOp reducer) const
transform_proxy_value<bag<Item>, TransformFunction> transform(TransformFunction ffn)
inline flatten_proxy_value<bag<Item>> flatten()
filter_proxy_value<bag<Item>, FilterFunction> filter(FilterFunction ffn)

Public Members
```

detail::round_robin_partitioner partitioner

Friends

friend class detail::base misc< bag< Item >, std::tuple< Item > >

7.2.3 counting set

template<typename Key>

class **counting_set**: public ygm::container::detail::base_count<*counting_set*<*Key*>, std::tuple<*Key*, size_t>>, public ygm::container::detail::base_misc<*counting_set*<*Key*>, std::tuple<*Key*, size_t>>, public ygm::container::detail::base_iteration_key_value<*counting_set*<*Key*>, std::tuple<*Key*, size_t>>

```
using self_type = counting_set<Key>
using mapped_type = size_t

using key_type = Key

using size_type = size_t

using for_all_args = std::tuple<Key, size_t>

using container_type = ygm::container::counting_set_tag
```

Public Functions

```
inline counting_set(ygm::comm &comm)
counting_set() = delete
inline counting_set(ygm::comm &comm, std::initializer_list<Key> l)
template<typename STLContainer>
inline counting_set(ygm::comm &comm, const STLContainer &cont)
template<typename YGMContainer>
inline counting_set(ygm::comm &comm, const YGMContainer &yc)
inline void async_insert(const key_type &key)
template<typename Function>
inline void local_for_all(Function fn)
template<typename Function>
inline void local_for_all(Function fn) const
inline void local_clear()
inline void clear()
inline size t local_size() const
inline mapped_type local_count(const key_type &key) const
inline mapped_type count_all()
template<typename CompareFunction>
inline std::vector<std::pair<key_type, mapped_type>> topk(size_t k, CompareFunction cfn)
inline std::map<key_type, mapped_type> gather_keys (const std::vector<key_type> &keys)
inline ygm::ygm\_ptr < self\_type > get\_ygm\_ptr() const
inline void serialize(const std::string &fname)
inline void deserialize(const std::string &fname)
inline size_t count (const std::tuple_element<0, std::tuple<Key, size_t>>::type &value) const
inline size_t size() const
inline void swap(counting_set<Key> &other)
inline ygm::comm &comm()
inline const ygm::comm &comm() const
inline ygm::ygm_ptr<counting_set<Key>> get_ygm_ptr()
inline void for_all(Function fn)
inline void for_all(Function fn) const
inline void gather (STLContainer &gto, int rank) const
```

```
inline std::vector<std::pair<key_type, mapped_type>> gather_topk(size_t k, Compare comp = Compare())
                                                                        const
     inline void collect(YGMContainer &c) const
     inline void reduce_by_key (MapType &map, ReductionOp reducer) const
     transform\_proxy\_key\_value < \textit{counting\_set} < \textit{Key} >, TransformFunction > \textbf{transform} (TransformFunction ffn)
     inline auto keys()
     inline auto values()
     inline flatten_proxy_key_value<counting_set<Key>> flatten()
     filter_proxy_key_value<counting_set<Key>, FilterFunction> filter(FilterFunction ffn)
     Public Members
     const size_type count_cache_size = 1024 * 1024
     detail::hash_partitioner<std::hash<key_type>> partitioner
     Friends
     friend class detail::base_misc< counting_set< Key >, std::tuple< Key, size_t > >
7.2.4 disjoint set
template<typename Item, typename Partitioner = detail::old_hash_partitioner<Item>>
class disjoint_set
     Public Types
     using self_type = disjoint_set<Item, Partitioner>
     using value_type = Item
     using size_type = size_t
     using ygm_for_all_types = std::tuple<Item, Item>
     using container_type = ygm::container::disjoint_set_tag
     using impl_type = detail::disjoint set impl<Item, Partitioner>
```

Public Functions

```
disjoint_set() = delete
inline disjoint_set(ygm::comm &comm, const size_t cache_size = 8192)
template<typename Visitor, typename ...VisitorArgs>
inline void async_visit(const value_type &item, Visitor visitor, const VisitorArgs&... args)
inline void async_union(const value_type &a, const value_type &b)
template<typename Function, typename ...FunctionArgs>
inline void async_union_and_execute(const value type &a, const value type &b, Function fn, const
                                       FunctionArgs&... args)
inline void all_compress()
template<typename Function>
inline void for_all(Function fn)
inline std::map<value type, value type> all_find(const std::vector<value type> &items)
inline void clear()
inline size type size()
inline size_type num_sets()
inline ygm::ygm_ptr<impl_type> get_ygm_ptr() const
inline ygm::comm &comm()
```

7.2.5 map

template<typename Key, typename Value>

```
class map: public ygm::container::detail::base_async_insert_key_value<map<Key, Value>, std::tuple<Key, Value>>, public ygm::container::detail::base_async_insert_or_assign<map<Key, Value>, std::tuple<Key, Value>>, public ygm::container::detail::base_misc<map<Key, Value>, std::tuple<Key, Value>>, public ygm::container::detail::base_count<map<Key, Value>, std::tuple<Key, Value>>, public ygm::container::detail::base_async_reduce<map<Key, Value>, std::tuple<Key, Value>>, public ygm::container::detail::base_async_erase_key<map<Key, Value>, std::tuple<Key, Value>>, public ygm::container::detail::base_async_erase_key_value<map<Key, Value>, std::tuple<Key, Value>>, public ygm::container::detail::base_async_erase_key_value<map<Key, Value>, std::tuple<Key, Value>>, public ygm::container::detail::base_batch_erase_key_value<map<Key, Value>, std::tuple<Key, Value>>, public ygm::container::detail::base_async_visit<map<Key, Value>, std::tuple<Key, Value>>, public ygm::container::detail::base_iteration_key_value<map<Key, Value>, std::tuple<Key, Value>>, public ygm::container::detail::base_iteration_key_value<map<Key, Value>, std::tuple<Key, Value>>
```

```
using self_type = map<Key, Value>
using mapped_type = Value
using ptr_type = typename ygm::ygm_ptr<self_type>
using key_type = Key
using size_type = size_t
using for_all_args = std::tuple<Key, Value>
using container_type = ygm::container::map_tag
Public Functions
map() = delete
inline map(ygm::comm &comm)
inline map(ygm::comm &comm, const mapped_type &default_value)
inline map(ygm::comm &comm, std::initializer_list<std::pair<Key, Value>> 1)
template<typename STLContainer>
inline map (ygm::comm &comm, const STLContainer &cont)
template<typename YGMContainer>
inline map(ygm::comm &comm, const YGMContainer &yc)
inline ~map()
inline void local_insert(const key_type &key)
inline void local_erase(const key_type &key)
inline void local_erase(const key_type &key, const key_type &value)
inline void local_insert(const key_type &key, const mapped_type &value)
inline void local_insert_or_assign(const key_type &key, const mapped_type &value)
inline void local_clear()
template<typename ReductionOp>
inline void local_reduce(const key_type &key, const mapped_type &value, ReductionOp reducer)
inline size_t local_size() const
inline mapped_type &local_at(const key_type &key)
```

```
inline const mapped_type &local_at(const key_type &key) const
template<typename Function, typename ...VisitorArgs>
inline void local_visit(const key type &key, Function &fn, const VisitorArgs &... args)
template<typename Function, typename ... VisitorArgs>
inline void local_visit_if_contains(const key type &key, Function &fn, const VisitorArgs &... args)
template<typename Function, typename ...VisitorArgs>
inline void local_visit_if_contains(const key_type &key, Function &fn, const VisitorArgs &... args)
                                       const
template<typename STLKeyContainer>
inline std::map<key_type, mapped_type> gather_keys(const STLKeyContainer &keys)
inline std::vector<mapped_type> local_get(const key_type &key) const
template<typename Function>
inline void local_for_all(Function fn)
template<typename Function>
inline void local_for_all(Function fn) const
inline size_t local_count (const key_type &key) const
inline void async_insert (const typename std::tuple_element<0, std::tuple< Key, Value>>::type &key, const
                          typename std::tuple_element<1, std::tuple<Key, Value>>::type &value)
inline void async_insert(const std::pair<const typename std::tuple_element<0, std::tuple<Key,
                          Value>>::type, typename std::tuple_element<1, std::tuple<Key, Value>>::type>
                          &kvp)
inline void async_insert_or_assign(const std::tuple_element<0, std::tuple<Key, Value>>::type &key,
                                      const std::tuple_element<1, std::tuple<Key, Value>>::type &value)
inline void async_insert_or_assign(const std::pair<typename std::tuple_element<0, std::tuple<Key,
                                      Value>>::type, typename std::tuple_element<1, std::tuple<Key,
                                      Value>>::type> &kvp)
inline size_t size() const
inline void clear()
inline void swap(map<Key, Value> &other)
inline ygm::comm &comm()
inline const ygm::comm &comm() const
inline ygm::ygm_ptr<map<Key, Value>> get_ygm_ptr()
inline const ygm::ygm_ptr<map<Key, Value>> get_ygm_ptr() const
inline size t count (const std::tuple element<0, std::tuple<Key, Value>>::type &value) const
inline void async_reduce(const typename std::tuple element<0, std::tuple</p>
                          typename std::tuple_element<1, std::tuple<Key, Value>>::type &value,
                          ReductionOp reducer)
```

```
inline void async_erase(const typename std::tuple_element<0, std::tuple<Key, Value>>::type &key)
inline void async_erase(const typename std::tuple_element<0, std::tuple< Key, Value>>::type &key, const
                         typename std::tuple_element<1, std::tuple<Key, Value>>::type &value)
inline void erase(const Container &cont)
inline void erase (const Container &cont)
inline void erase(const Container &cont)
inline void erase (const Container &cont)
inline void erase(const Container &cont)
inline void async_visit(const std::tuple_element<0, std::tuple<Key, Value>>::type &key, Visitor visitor,
                         const VisitorArgs&... args)
inline void async_visit_if_contains(const std::tuple_element<0, std::tuple<Key, Value>>::type &key,
                                        Visitor visitor, const VisitorArgs&... args)
inline void async_visit_if_contains(const std::tuple_element<0, std::tuple<Key, Value>>::type &key,
                                        Visitor visitor, const VisitorArgs&... args) const
inline void for_all(Function fn)
inline void for_all(Function fn) const
inline void gather (STLContainer &gto, int rank) const
inline std::vector<std::pair<*key_type, mapped_type>> gather_topk(size_t k, Compare comp = Compare())
                                                                  const
inline void collect(YGMContainer &c) const
inline void reduce_by_key (MapType &map, ReductionOp reducer) const
transform_proxy_key_value<map<Key, Value>, TransformFunction> transform(TransformFunction ffn)
inline auto keys()
inline auto values()
inline flatten_proxy_key_value<map<Key, Value>> flatten()
filter_proxy_key_value<map<Key, Value>, FilterFunction> filter(FilterFunction ffn)
Public Members
detail::hash_partitioner<std::hash<key_type>> partitioner
```

Friends

friend class detail::base_misc< map< Key, Value >, std::tuple< Key, Value > >

7.2.6 multimap

template<typename Key, typename Value>

class **multimap**: public ygm::container::detail::base_async_insert_key_value<*multimap*<*Key*, *Value*>, std::tuple<*Key*, *Value*>>, public ygm::container::detail::base_misc<*multimap*<*Key*, *Value*>>, public ygm::container::detail::base_count<*multimap*<*Key*, *Value*>>, std::tuple<*Key*, *Value*>>, public ygm::container::detail::base_async_erase_key<*multimap*<*Key*, *Value*>, std::tuple<*Key*, *Value*>>, public ygm::container::detail::base_async_erase_key_value<*multimap*<*Key*, *Value*>, std::tuple<*Key*, *Value*>>, public ygm::container::detail::base_batch_erase_key_value<*multimap*<*Key*, *Value*>, std::tuple<*Key*, *Value*>>, public ygm::container::detail::base_async_visit<*multimap*<*Key*, *Value*>, std::tuple<*Key*, *Value*>>, public ygm::container::detail::base_iteration_key_value<*multimap*<*Key*, *Value*>, std::tuple<*Key*, *Value*>>, public

```
using self_type = multimap<Key, Value>
using mapped_type = Value
using ptr_type = typename ygm::ygm_ptr<self_type>
using key_type = Key
using size_type = size_t
using for_all_args = std::tuple<Key, Value>
using container_type = ygm::container::multimap_tag
Public Functions
multimap() = delete
inline multimap(ygm::comm &comm)
inline multimap(ygm::comm &comm, const mapped_type &default_value)
inline multimap(ygm::comm &comm, std::initializer_list<std::pair<Key, Value>> 1)
template<typename STLContainer>
inline multimap(ygm::comm &comm, const STLContainer &cont)
template<typename YGMContainer>
inline multimap(ygm::comm &comm, const YGMContainer &yc)
```

```
inline ~multimap()
inline void local_insert(const key_type &key)
inline void local_erase(const key_type &key)
inline void local_erase(const key_type &key, const key_type &value)
inline void local_insert(const key_type &key, const mapped_type &value)
inline void local_clear()
inline size_t local_size() const
template<typename Function, typename ... VisitorArgs>
inline void local_visit(const key_type &key, Function &fn, const VisitorArgs &... args)
template<typename Function, typename ...VisitorArgs>
inline void local_visit_if_contains(const key_type &key, Function &fn, const VisitorArgs&... args)
template<typename Function, typename ... VisitorArgs>
inline void local_visit_if_contains(const key_type &key, Function &fn, const VisitorArgs &... args)
                                      const
inline std::vector<mapped_type> local_get(const key_type &key)
template<typename Function>
inline void local_for_all(Function fn)
template<typename Function>
inline void local_for_all(Function fn) const
inline size_t local_count(const key_type &key) const
inline void async_insert (const typename std::tuple_element<0, std::tuple</p>
                         typename std::tuple_element<1, std::tuple<Key, Value>>::type &value)
inline void async_insert (const std::pair<const typename std::tuple_element<0, std::tuple<Key,
                         Value>>::type, typename std::tuple element<1, std::tuple<Key, Value>>::type>
                         &kvp)
inline size t size() const
inline void clear()
inline void swap(multimap<Key, Value> &other)
inline ygm::comm &comm()
inline const ygm::comm &comm() const
inline ygm::ygm_ptr<multimap<Key, Value>> get_ygm_ptr()
inline const ygm::ygm_ptr<multimap<Key, Value>> get_ygm_ptr() const
inline size_t count (const std::tuple_element<0, std::tuple</ri>
inline void async_erase(const typename std::tuple_element<0, std::tuple<Key, Value>>::type &key)
```

```
inline void async_erase(const typename std::tuple_element<0, std::tuple<Key, Value>>::type &key, const
                         typename std::tuple_element<1, std::tuple<Key, Value>>::type &value)
inline void erase(const Container &cont)
inline void erase(const Container &cont)
inline void erase (const Container &cont)
inline void erase(const Container &cont)
inline void erase(const Container &cont)
inline void async_visit(const std::tuple_element<0, std::tuple<Key, Value>>::type &key, Visitor visitor,
                         const VisitorArgs&... args)
inline void async_visit_if_contains(const std::tuple_element<0, std::tuple<Key, Value>>::type &key,
                                        Visitor visitor, const VisitorArgs&... args)
inline void async_visit_if_contains(const std::tuple_element<0, std::tuple<Key, Value>>::type &key,
                                        Visitor visitor, const VisitorArgs&... args) const
inline void for_all(Function fn)
inline void for_all(Function fn) const
inline void gather (STLContainer &gto, int rank) const
inline std::vector<std::pair<key_type, mapped_type>> gather_topk(size_t k, Compare comp = Compare())
                                                                  const
inline void collect(YGMContainer &c) const
inline void reduce_by_key (MapType &map, ReductionOp reducer) const
transform_proxy_key_value<multimap<Key, Value>, TransformFunction> transform(TransformFunction
                                                                                  ffn)
inline auto keys()
inline auto values()
inline flatten_proxy_key_value<multimap<Key, Value>> flatten()
filter_proxy_key_value<multimap<Key, Value>, FilterFunction> filter(FilterFunction ffn)
Public Members
detail::hash_partitioner<std::hash<key_type>> partitioner
```

Friends

friend class detail::base_misc< multimap< Key, Value >, std::tuple< Key, Value > >

7.2.7 multiset

```
template<typename Value>
```

```
class multiset: public ygm::container::detail::base_async_insert_value<multiset<Value>>, std::tuple<Value>>, public ygm::container::detail::base_async_erase_key<multiset<Value>>, std::tuple<Value>>, public ygm::container::detail::base_batch_erase_key<multiset<Value>>, std::tuple<Value>>, public ygm::container::detail::base_async_contains<multiset<Value>>, std::tuple<Value>>, public ygm::container::detail::base_async_insert_contains<multiset<Value>>, std::tuple<Value>>, public ygm::container::detail::base_count<multiset<Value>>, std::tuple<Value>>, public ygm::container::detail::base_misc<multiset<Value>>, std::tuple<Value>>, public ygm::container::detail::base_iteration_value<multiset<Value>>, std::tuple<Value>>
```

Public Types

```
using self_type = multiset < Value>
using value_type = Value
using size_type = size_t
using for_all_args = std::tuple<Value>
using container_type = ygm::container::set_tag
using key_type = std::tuple_element_t<0, std::tuple<Value>>
Public Functions
inline multiset(ygm::comm &comm)
inline multiset(const self_type &other)
inline multiset(self_type &&other) noexcept
inline multiset(ygm::comm &comm, std::initializer_list<Value> 1)
template<typename STLContainer>
inline multiset(ygm::comm &comm, const STLContainer &cont)
template<typename YGMContainer>
inline multiset(ygm::comm &comm, const YGMContainer &yc)
inline ~multiset()
```

```
multiset() = delete
inline multiset & operator=(const self_type & other)
inline multiset & operator=(self_type & & other) noexcept
inline void local_insert(const value_type &val)
inline void local_erase(const value_type &val)
inline void local_clear()
inline size_t local_count(const value_type &val) const
inline size_t local_size() const
template<typename Function>
inline void local_for_all(Function fn)
template<typename Function>
inline void local_for_all(Function fn) const
inline void serialize(const std::string &fname)
inline void deserialize(const std::string &fname)
inline void async_insert(const typename std::tuple element<0, std::tuple<Value>>::type &value)
inline void async_erase(const typename std::tuple_element<0, std::tuple<Value>>::type &key)
inline void erase(const Container &cont)
inline void erase(const Container &cont)
inline void async_contains(const std::tuple_element<0, std::tuple<Value>>::type &value, Function fn, const
                             FuncArgs&... args)
inline void async_insert_contains(const std::tuple_element<0, std::tuple<Value>>::type &value,
                                     Function fn, const FuncArgs&... args)
inline size_t count (const std::tuple_element<0, std::tuple<Value>>::type &value) const
inline size_t size() const
inline void clear()
inline void swap (multiset < Value > & other)
inline ygm::comm &comm()
inline const ygm::comm &comm() const
inline ygm::ygm_ptr<multiset<Value>> get_ygm_ptr()
inline const ygm::ygm_ptr<multiset<Value>> get_ygm_ptr() const
inline void for_all(Function fn)
inline void for_all(Function fn) const
```

```
inline void gather(STLContainer &gto, int rank) const
inline std::vector<value_type> gather_topk(size_t k, Compare comp = std::greater<value_type>()) const
inline value_type reduce(MergeFunction merge) const
inline void collect(YGMContainer &c) const
inline void reduce_by_key (MapType &map, ReductionOp reducer) const
transform_proxy_value<multiset<Value>, TransformFunction> transform(TransformFunction ffn)
inline flatten_proxy_value<multiset<Value>> flatten()
filter_proxy_value<multiset<Value>, FilterFunction> filter(FilterFunction ffn)
Public Members
detail::hash_partitioner<std::hash<value_type>> partitioner
Friends
friend class detail::base_misc< multiset< Value >, std::tuple< Value > >
```

7.2.8 set

```
template<typename Value>
```

```
class set: public ygm::container::detail::base_async_insert_value<set<Value>>, std::tuple<Value>>, public
ygm::container::detail::base_async_erase_key<set<Value>, std::tuple<Value>>, public
ygm::container::detail::base_batch_erase_key<set<Value>>, std::tuple<Value>>, public
ygm::container::detail::base_async_contains<set<Value>, std::tuple<Value>>, public
ygm::container::detail::base_async_insert_contains<set<Value>, std::tuple<Value>>, public
ygm::container::detail::base_count<set<Value>>, std::tuple<Value>>, public
ygm::container::detail::base_misc<set<Value>, std::tuple<Value>>, public
ygm::container::detail::base_iteration_value<set<Value>, std::tuple<Value>>
```

Public Types

```
using self_type = set<Value>
using value_type = Value
using size_type = size_t
using for_all_args = std::tuple<Value>
using container_type = ygm::container::set_tag
```

```
using key_type = std::tuple_element_t<0, std::tuple<Value>>
Public Functions
inline set(ygm::comm &comm)
inline set(const self_type &other)
inline set(self_type &&other) noexcept
inline set(ygm::comm &comm, std::initializer_list<Value> 1)
template<typename STLContainer>
inline set(ygm::comm &comm, const STLContainer &cont)
template<typename YGMContainer>
inline set(ygm::comm &comm, const YGMContainer &yc)
inline ~set()
set() = delete
inline set & operator=(const self_type & other)
inline set &operator=(self_type &&other) noexcept
inline void local_insert(const value_type &val)
inline void local_erase(const value_type &val)
inline void local_clear()
inline size_t local_count(const value_type &val) const
inline size t local_size() const
template<typename Function>
inline void local_for_all(Function fn)
template<typename Function>
inline void local_for_all(Function fn) const
inline void serialize(const std::string &fname)
inline void deserialize (const std::string &fname)
inline void async_insert(const typename std::tuple_element<0, std::tuple<Value>>::type &value)
inline void async_erase(const typename std::tuple_element<0, std::tuple<Value>>::type &key)
inline void erase(const Container &cont)
inline void erase(const Container &cont)
inline void async_contains(const std::tuple_element<0, std::tuple<Value>>::type &value, Function fn, const
                            FuncArgs&... args)
```

```
inline void async_insert_contains(const std::tuple_element<0, std::tuple<Value>>::type &value,
                                     Function fn, const FuncArgs&... args)
inline size_t count (const std::tuple_element<0, std::tuple<Value>>::type &value) const
inline size_t size() const
inline void clear()
inline void swap(set<Value> &other)
inline ygm::comm &comm()
inline const ygm::comm &comm() const
inline ygm::ygm_ptr<set<Value>> get_ygm_ptr()
inline const ygm::ygm_ptr<set<Value>> get_ygm_ptr() const
inline void for_all(Function fn)
inline void for_all(Function fn) const
inline void gather(STLContainer &gto, int rank) const
inline std::vector<value_type> gather_topk(size_t k, Compare comp = std::greater<value_type>()) const
inline value_type reduce(MergeFunction merge) const
inline void collect(YGMContainer &c) const
inline void reduce_by_key (MapType &map, ReductionOp reducer) const
transform_proxy_value<set<Value>, TransformFunction> transform(TransformFunction ffn)
inline flatten_proxy_value<set<Value>> flatten()
filter_proxy_value<set<Value>, FilterFunction> filter(FilterFunction ffn)
Public Members
detail::hash_partitioner<std::hash<value_type>> partitioner
Friends
friend class detail::base misc< set< Value >, std::tuple< Value > >
```

YGM CONTAINER EXAMPLE

```
// Copyright 2019-2021 Lawrence Livermore National Security, LLC and other YGM
// Project Developers. See the top-level COPYRIGHT file for details.
// SPDX-License-Identifier: MIT
#include <ygm/comm.hpp>
#include <ygm/container/map.hpp>
int main(int argc, char **argv) {
 ygm::comm world(&argc, &argv);
 ygm::container::map<std::string, std::string> my_map(world);
 if (world.rank0()) {
   my_map.async_insert("dog", "bark");
   my_map.async_insert("cat", "meow");
 world.barrier();
  auto favorites_lambda = [](auto key, auto &value, const int favorite_num) {
   std::cout << "My favorite animal is a " << key << ". It says '" << value
              << "!' My favorite number is " << favorite_num << std::endl;</pre>
 };
 // Send visitors to map
 if (world.rank() % 2) {
   my_map.async_visit("dog", favorites_lambda, world.rank());
 } else {
   my_map.async_visit("cat", favorites_lambda, world.rank() + 1000);
  }
 return 0;
```

CHAPTER

NINE

CONTAINER TRANSFORMATION OBJECTS

ygm::container provides a number of transformation objects that can be applied to containers to alter the appearance of items passed to for_all operations without modifying the items within the container itself. The currently supported transformation objects are:

- filter Filters items in a container to only execute on the portion of the container satisfying a provided boolean function.
- flatten Extract the elements from tuple-like objects before passing to the user's for_all function.
- map Apply a generic function to the container's items before passing to the user's for_all function.

DOCUMENTS FOR YGM DEVELOPERS

10.1 Developing YGM

This page contains information for YGM developers.

10.1.1 Build Read the Docs (RTD)

Here is how to build RTD document using Sphinx on your machine.

Listing 1: How to build RTD docs locally

```
# Install required software
brew install doxygen graphviz sphinx-doc
pip install breathe sphinx_rtd_theme
# Set PATH and PYTHONPATH, if needed
# For example:
# export PATH="/opt/homebrew/opt/sphinx-doc/bin:${PATH}"
# export PYTHONPATH="/path/to/python/site-packages:${PYTHONPATH}"
git clone https://github.com/LLNL/ygm.git
cd ygm
mkdir build && cd build
# Run CMake
cmake ../ -DYGM_RTD_ONLY=ON
# Generate Read the Docs documents using Sphinx
# This command runs Doxygen to generate XML files
# before Sphinx automatically
make sphinx
# Open the following file using a web browser
open docs/rtd/sphinx/index.html
# For running doxygen only
make doxygen
# open the following file using a web browser
open docs/html/index.html
```

Rerunning Build Command

Depending on what files are modified, one may need to rerun the CMake command and/or make sphinx. For instance:

- Require running the CMake command and make sphinx:
 - Adding new RTD-related files, including configuration and .rst files
 - Modifying CMake files
- Require running only make sphinx
 - Existing files (except CMake) are modified

CHAPTER

ELEVEN

INDICES AND TABLES

- genindex
- modindex
- search

INDEX

Υ	ygm::comm::rank0 (C++ function), 11
ygm::comm(C++class), 9	ygm::comm::register_pre_barrier_callback
ygm::comm:: \sim comm ($C++$ function), 9	(C++function), 10
ygm::comm::all_reduce $(C++function)$, 10	ygm::comm::router(C++ function), 11
ygm::comm::all_reduce_max(C++ function), 10	ygm::comm::size(C++ function), 11
ygm::comm::all_reduce_min(C++ function), 10	ygm::comm::stats_print(C++ function), 9
ygm::comm::all_reduce_sum(C++ function), 10	ygm::comm::stats_reset(C++ function), 9
ygm::comm::async $(C++function)$, 10	ygm::comm::welcome(C++ function), 9
ygm::comm::async_($C++$ function), 10 ygm::comm::async_bcast ($C++$ function), 10	ygm::container::array(C++ class), 17
ygm::comm::async_bcast(C++ function), 10 ygm::comm::async_mcast(C++ function), 10	ygm::container::array::~array(C++function), 18
ygm::comm::barrier(C++ function), 10	ygm::container::array::array(C++ function), 18
ygm::comm::cerr (C++ function), 10	<pre>ygm::container::array::async_binary_op_update_value</pre>
ygm::comm::cerr($C++$ function), 11 ygm::comm::cerr0($C++$ function), 11, 12	(C++ function), 19
ygm::comm::cf_barrier(C++ function), 11, 12 ygm::comm::cf_barrier(C++ function), 10	<pre>ygm::container::array::async_bit_and (C++</pre>
ygm::comm::ci_barrier ($C++$ function), 10 ygm::comm::comm ($C++$ function), 9	function), 19
ygm::comm::cont $(C++function)$, 9 ygm::comm::cout $(C++function)$, 11	ygm::container::array::async_bit_or (C++
ygm::comm::cout $(C++function)$, 11 ygm::comm::cout0 $(C++function)$, 11, 12	function), 19
ygm::comm::get_mpi_comm(C++ function), 11	<pre>ygm::container::array::async_bit_xor (C++</pre>
ygm::comm::header_t ($C++$ struct), 12	function), 19
ygm::comm::header_t:(C++ struct), 12 ygm::comm::header_t::dest(C++ member), 12	<pre>ygm::container::array::async_decrement (C++</pre>
ygm::comm::header_t::message_size (C++ mem-	function), 19
ber), 12	<pre>ygm::container::array::async_divides (C++</pre>
ygm::comm::layout $(C++function)$, 11	function), 19
ygm::comm::local_process_incoming ($C++$ func-	<pre>ygm::container::array::async_increment (C++</pre>
tion), 10	function), 19
ygm::comm::local_progress(C++ function), 10	<pre>ygm::container::array::async_insert (C++</pre>
ygm::comm::local_plogress(C++ function), 10 ygm::comm::local_wait_until(C++ function), 10	function), 19
ygm::comm::make_ygm_ptr $(C++function)$, 10	<pre>ygm::container::array::async_logical_and</pre>
ygm::comm::make_ygm_ptf (C++ function), 10 ygm::comm::mpi_bcast (C++ function), 11	(C++ function), 19
ygm::comm::mpi_bcast(C++ junction), 11 ygm::comm::mpi_irecv_request(C++ struct), 12	ygm::container::array::async_logical_or
ygm::comm::mpi_irecv_request::buffer (C++	(C++ function), 19
member), 12	ygm::container::array::async_minus (C++ func-
ygm::comm::mpi_irecv_request::request (C++	tion), 19
member), 12	ygm::container::array::async_multiplies
ygm::comm::mpi_isend_request (C++ struct), 12	(C++ function), 19
ygm::comm::mpi_isend_request::buffer (C++	ygm::container::array::async_plus (C++ func-
member), 12	tion), 19
ygm::comm::mpi_isend_request::request (C++	<pre>ygm::container::array::async_reduce (C++</pre>
member), 12	function), 20
ygm::comm::mpi_recv(C++ function), 11	ygm::container::array::async_set (C++ func-
ygm::comm::mpi_recv(C++ function), 11 ygm::comm::mpi_send(C++ function), 11	tion), 18
ygm::comm::mpi_send(C++ function), 11 ygm::comm::rank(C++ function), 11	ygm::container::array::async_unary_op_update_value
ygmcommrank (C++ junction), 11	, , , , , , , , , , , , , , , , , , ,

(C++ function), 19	<pre>ygm::container::array::transform (C++ func-</pre>
<pre>ygm::container::array::async_visit (C++ func-</pre>	<i>tion</i>), 20
<i>tion</i>), 20	<pre>ygm::container::array::values(C++function), 20</pre>
<pre>ygm::container::array::async_visit_if_contain</pre>	nsygm::container::bag(C++ class),21
(C++ function), 20	<pre>ygm::container::bag::~bag(C++ function), 21</pre>
<pre>ygm::container::array::clear (C++ function), 19</pre>	<pre>ygm::container::bag::async_insert (C++ func-</pre>
<pre>ygm::container::array::collect (C++ function),</pre>	tion), 22
20	ygm::container::bag::bag(C++ function), 21
<pre>ygm::container::array::comm (C++ function), 20</pre>	ygm::container::bag::clear(C++ function), 22
ygm::container::array::container_type (C++	ygm::container::bag::collect(C++ function), 23
type), 18	ygm::container::bag::comm (C++ function), 22
<pre>ygm::container::array::default_value (C++)</pre>	ygm::container::bag::container_type (C++
function), 19	type), 21
ygm::container::array::filter(C++function), 20	ygm::container::bag::count(C++ function), 22
ygm::container::array::flatten ($C++$ function),	ygm::container::bag::deserialize (C++ func-
20	tion), 22
<pre>ygm::container::array::for_all (C++ function),</pre>	ygm::container::bag::filter(C++ function), 23
20	ygm::container::bag::flatten(C++ function), 23
$ygm::container::array::for_all_args$ (C++	ygm::container::bag::for_all(C++ function), 22
<i>type</i>), 18	<pre>ygm::container::bag::for_all_args (C++ type),</pre>
<pre>ygm::container::array::gather(C++function), 20</pre>	21
<pre>ygm::container::array::gather_topk (C++ func-</pre>	<pre>ygm::container::bag::gather(C++function), 22</pre>
tion), 20	<pre>ygm::container::bag::gather_topk (C++ func-</pre>
<pre>ygm::container::array::get_ygm_ptr (C++ func-</pre>	tion), 23
tion), 20	<pre>ygm::container::bag::get_ygm_ptr (C++ func-</pre>
<pre>ygm::container::array::key_type (C++ type), 18</pre>	tion), 22
ygm::container::array::keys(C++function), 20	<pre>ygm::container::bag::global_shuffle (C++</pre>
<pre>ygm::container::array::local_clear (C++ func-</pre>	function), 22
tion), 19	<pre>ygm::container::bag::local_clear (C++ func-</pre>
<pre>ygm::container::array::local_for_all (C++</pre>	tion), 22
function), 19	<pre>ygm::container::bag::local_count (C++ func-</pre>
<pre>ygm::container::array::local_insert (C++</pre>	tion), 22
function), 18	<pre>ygm::container::bag::local_for_all (C++ func-</pre>
ygm::container::array::local_reduce (C++	tion), 22
function), 19	ygm::container::bag::local_insert (C++ func-
ygm::container::array::local_size (C++ func-	tion), 22
tion), 19	ygm::container::bag::local_shuffle (C++ func-
ygm::container::array::local_swap (C++ func-	tion), 22
tion), 19	ygm::container::bag::local_size(C++ function),
ygm::container::array::local_visit (C++ func-	22
	ygm::container::bag::operator= (C++ function),
tion), 18	· · · · · · · · · · · · · · · · ·
<pre>ygm::container::array::mapped_type (C++ type),</pre>	21, 22
18	ygm::container::bag::partitioner (C++ mem-
<pre>ygm::container::array::partitioner (C++ mem-</pre>	ber), 23
ber), 21	<pre>ygm::container::bag::rebalance (C++ function),</pre>
ygm::container::array::ptr_type(C++ type), 18	22
<pre>ygm::container::array::reduce_by_key (C++</pre>	ygm::container::bag::reduce(C++ function), 23
function), 20	<pre>ygm::container::bag::reduce_by_key (C++ func-</pre>
ygm::container::array::resize(C++function), 19	tion), 23
$ygm::container::array::self_type(C++ type), 18$	<pre>ygm::container::bag::self_type(C++ type), 21</pre>
ygm::container::array::size(C++function), 19	<pre>ygm::container::bag::serialize (C++ function),</pre>
$ygm::container::array::size_type(C++ type), 18$	22
<pre>ygm::container::array::sort(C++ function), 19</pre>	<pre>ygm::container::bag::size (C++ function), 22</pre>
<pre>ygm::container::array::swap (C++ function), 19</pre>	<pre>ygm::container::bag::size_type (C++ type), 21</pre>

```
ygm::container::bag::swap(C++ function), 22
                                                        (C++ type), 23
ygm::container::bag::transform (C++ function),
                                                ygm::container::counting_set::partitioner
                                                        (C++ member), 25
                                                ygm::container::counting_set::reduce_by_key
ygm::container::bag::value_type (C++ type), 21
ygm::container::counting_set (C++ class), 23
                                                         (C++ function), 25
ygm::container::counting_set::async_insert
                                                ygm::container::counting_set::self_type
        (C++ function), 24
                                                        (C++ type), 23
                                                ygm::container::counting_set::serialize
ygm::container::counting_set::clear
       function), 24
                                                         (C++function), 24
ygm::container::counting\_set::collect (C++
                                                ygm::container::counting_set::size (C++ func-
       function), 25
                                                        tion), 24
ygm::container::counting_set::comm (C++ func-
                                                ygm::container::counting_set::size_type
       tion), 24
                                                        (C++ type), 23
ygm::container::counting_set::container_type ygm::container::counting_set::swap (C++ func-
        (C++ type), 23
                                                         tion), 24
ygm::container::counting_set::count
                                          (C++ ygm::container::counting_set::topk (C++ func-
       function), 24
                                                        tion), 24
ygm::container::counting_set::count_all
                                                ygm::container::counting_set::transform
                                                        (C++function), 25
        (C++ function), 24
ygm::container::counting_set::count_cache_sizeygm::container::counting_set::values
                                                                                          (C++
        (C++ member), 25
                                                        function), 25
ygm::container::counting_set::counting_set
                                                ygm::container::disjoint_set (C++ class), 25
        (C++ function), 24
                                                ygm::container::disjoint_set::all_compress
ygm::container::counting_set::deserialize
                                                         (C++ function), 26
                                                ygm::container::disjoint_set::all_find (C++
        (C++ function), 24
ygm::container::counting_set::filter
                                         (C++
                                                        function), 26
       function), 25
                                                ygm::container::disjoint_set::async_union
ygm::container::counting_set::flatten
                                         (C++
                                                         (C++ function), 26
       function), 25
                                                ygm::container::disjoint_set::async_union_and_execute
ygm::container::counting_set::for_all
                                         (C++
                                                        (C++ function), 26
       function), 24
                                                ygm::container::disjoint_set::async_visit
ygm::container::counting_set::for_all_args
                                                         (C++ function), 26
                                                ygm::container::disjoint_set::clear
        (C++ type), 23
                                                                                          (C++
ygm::container::counting_set::gather
                                         (C++
                                                        function), 26
       function), 24
                                                ygm::container::disjoint_set::comm (C++ func-
ygm::container::counting_set::gather_keys
                                                        tion), 26
        (C++ function), 24
                                                ygm::container::disjoint_set::container_type
ygm::container::counting_set::gather_topk
                                                         (C++ type), 25
        (C++ function), 24
                                                ygm::container::disjoint_set::disjoint_set
ygm::container::counting_set::get_ygm_ptr
                                                        (C++function), 26
        (C++ function), 24
                                                ygm::container::disjoint_set::for_all (C++
ygm::container::counting_set::key_type (C++
                                                        function), 26
        type), 23
                                                ygm::container::disjoint_set::get_ygm_ptr
ygm::container::counting_set::keys (C++ func-
                                                        (C++function), 26
                                                ygm::container::disjoint_set::impl_type
        tion), 25
ygm::container::counting_set::local_clear
                                                         (C++ type), 25
        (C++ function), 24
                                                ygm::container::disjoint_set::num_sets (C++
ygm::container::counting_set::local_count
                                                        function), 26
        (C++ function), 24
                                                ygm::container::disjoint_set::self_type
ygm::container::counting_set::local_for_all
                                                        (C++ type), 25
        (C++function), 24
                                                ygm::container::disjoint_set::size (C++ func-
ygm::container::counting_set::local_size
                                                        tion), 26
        (C++ function), 24
                                                ygm::container::disjoint_set::size_type
ygm::container::counting_set::mapped_type
                                                        (C++ type), 25
```

```
ygm::container::disjoint_set::value_type
                                                ygm::container::map::local_reduce (C++ func-
        (C++ type), 25
                                                        tion), 27
ygm::container::disjoint_set::ygm_for_all_typeysgm::container::map::local_size(C++ function),
        (C++ type), 25
ygm::container::map(C++ class), 26
                                                ygm::container::map::local_visit (C++ func-
ygm::container::map::\simmap (C++ function), 27
                                                        tion), 28
ygm::container::map::async_erase (C++ func-
                                                ygm::container::map::local_visit_if_contains
                                                        (C++ function), 28
        tion), 28, 29
ygm::container::map::async_insert (C++ func-
                                                ygm::container::map::map (C++ function), 27
                                                ygm::container::map::mapped_type(C++ type), 27
        tion), 28
ygm::container::map::async_insert_or_assign
                                                ygm::container::map::partitioner (C++ mem-
        (C++ function), 28
                                                        ber), 29
ygm::container::map::async_reduce (C++ func-
                                                ygm::container::map::ptr_type (C++ type), 27
                                                ygm::container::map::reduce_by_key (C++ func-
        tion), 28
ygm::container::map::async_visit (C++ func-
                                                        tion), 29
        tion), 29
                                                ygm::container::map::self_type (C++ type), 27
ygm::container::map::async_visit_if_contains
                                                ygm::container::map::size(C++ function), 28
        (C++ function), 29
                                                ygm::container::map::size\_type(C++ type), 27
ygm::container::map::clear (C++ function), 28
                                                ygm::container::map::swap(C++ function), 28
                                                ygm::container::map::transform (C++ function),
ygm::container::map::collect (C++ function), 29
ygm::container::map::comm (C++ function), 28
ygm::container::map::container_type
                                                ygm::container::map::values(C++ function), 29
                                                ygm::container::multimap (C++ class), 30
        type), 27
ygm::container::map::count(C++ function), 28
                                                ygm::container::multimap::~multimap
                                                                                          (C++
ygm::container::map::erase(C++ function), 29
                                                        function), 30
ygm::container::map::filter(C++ function), 29
                                                ygm::container::multimap::async_erase (C++
ygm::container::map::flatten(C++ function), 29
                                                        function), 31
ygm::container::map::for_all (C++ function), 29
                                                ygm::container::multimap::async_insert (C++
ygm::container::map::for_all_args (C++ type),
                                                        function), 31
                                                ygm::container::multimap::async_visit (C++
ygm::container::map::gather(C++ function), 29
                                                        function), 32
ygm::container::map::gather_keys (C++ func-
                                                ygm::container::multimap::async_visit_if_contains
                                                        (C++function), 32
ygm::container::map::gather_topk (C++ func-
                                                ygm::container::multimap::clear (C++ function),
        tion), 29
ygm::container::map::get_ygm_ptr (C++ func-
                                                ygm::container::multimap::collect (C++ func-
                                                        tion), 32
ygm::container::map::key_type(C++ type), 27
                                                ygm::container::multimap::comm (C++ function),
ygm::container::map::keys (C++ function), 29
ygm::container::map::local_at(C++function), 27
                                                ygm::container::multimap::container_type
ygm::container::map::local_clear (C++ func-
                                                        (C++ type), 30
        tion), 27
                                                ygm::container::multimap::count (C++ function),
ygm::container::map::local_count (C++ func-
                                                ygm::container::multimap::erase(C++ function),
        tion), 28
ygm::container::map::local_erase (C++ func-
                                                ygm::container::multimap::filter (C++ func-
        tion), 27
ygm::container::map::local_for_all (C++ func-
                                                        tion), 32
                                                ygm::container::multimap::flatten (C++ func-
ygm::container::map::local_get (C++ function),
                                                        tion), 32
                                                ygm::container::multimap::for_all (C++ func-
ygm::container::map::local_insert (C++ func-
                                                        tion), 32
                                                ygm::container::multimap::for_all_args (C++
ygm::container::map::local_insert_or_assign
                                                        type), 30
        (C++ function), 27
                                                ygm::container::multimap::gather (C++ func-
```

tion), 32	<pre>ygm::container::multiset::async_erase (C++</pre>
$\verb"ygm"::container::multimap::gather_topk" (C++$	function), 34
function), 32	<pre>ygm::container::multiset::async_insert (C++</pre>
$\verb"ygm"::container::multimap::get_ygm_ptr" (C++$	function), 34
function), 31	<pre>ygm::container::multiset::async_insert_contains</pre>
$ygm::container::multimap::key_type (C++ type),$	(C++function), 34
30	ygm::container::multiset::clear(C++ function),
<pre>ygm::container::multimap::keys (C++ function),</pre>	34
32	<pre>ygm::container::multiset::collect (C++ func-</pre>
<pre>ygm::container::multimap::local_clear (C++</pre>	tion), 35
function), 31	<pre>ygm::container::multiset::comm (C++ function),</pre>
<pre>ygm::container::multimap::local_count (C++</pre>	34
function), 31	ygm::container::multiset::container_type
<pre>ygm::container::multimap::local_erase (C++</pre>	(C++ type), 33
function), 31	<pre>ygm::container::multiset::count (C++ function),</pre>
ygm::container::multimap::local_for_all	34
(C++ function), 31	ygm::container::multiset::deserialize (C++
ygm::container::multimap::local_get (C++	function), 34
function), 31	<pre>ygm::container::multiset::erase (C++ function), 34</pre>
<pre>ygm::container::multimap::local_insert (C++ function), 31</pre>	ygm::container::multiset::filter (C++ func-
ygm::container::multimap::local_size (C++	tion), 35
function), 31	ygm::container::multiset::flatten (C++ func-
ygm::container::multimap::local_visit (C++	tion), 35
function), 31	ygm::container::multiset::for_all (C++ func-
ygm::container::multimap::local_visit_if_con	· · · · · · · · · · · · · · · · · ·
(C++ function), 31	ygm::container::multiset::for_all_args (C++
<pre>ygm::container::multimap::mapped_type (C++</pre>	type), 33
type), 30	ygm::container::multiset::gather (C++ func-
<pre>ygm::container::multimap::multimap (C++ func-</pre>	tion), 34
tion), 30	<pre>ygm::container::multiset::gather_topk (C++</pre>
ygm::container::multimap::partitioner (C++	function), 35
member), 32	<pre>ygm::container::multiset::get_ygm_ptr (C++</pre>
$ygm::container::multimap::ptr_type (C++ type),$	function), 34
30	<pre>ygm::container::multiset::key_type (C++ type),</pre>
ygm::container::multimap::reduce_by_key	33
(C++function), 32	$ygm::container::multiset::local_clear$ (C++
$\verb"ygm"::container::multimap::self_type" (C++$	function), 34
<i>type</i>), 30	<pre>ygm::container::multiset::local_count (C++</pre>
<pre>ygm::container::multimap::size (C++ function),</pre>	function), 34
31	<pre>ygm::container::multiset::local_erase (C++</pre>
<pre>ygm::container::multimap::size_type (C++</pre>	function), 34
type), 30	ygm::container::multiset::local_for_all
ygm::container::multimap::swap (C++ function),	(C++ function), 34
31	ygm::container::multiset::local_insert (C++
<pre>ygm::container::multimap::transform (C++ function), 32</pre>	<pre>function), 34 ygm::container::multiset::local_size (C++</pre>
ygm::container::multimap::values (C++ func-	function), 34
tion), 32	ygm::container::multiset::multiset (C++ func-
ygm::container::multiset (C++ class), 33	tion), 33
ygm::container::multiset(C++ tass), 35 ygm::container::multiset(C++ tass), 35	ygm::container::multiset::operator= (C++
function), 33	function), 34
ygm::container::multiset::async_contains	ygm::container::multiset::partitioner (C++
(<i>C</i> ++ <i>function</i>), 34	member), 35

```
ygm::container::multiset::reduce (C++ func- ygm::container::set::local_for_all (C++ func-
        tion), 35
                                                         tion), 36
ygm::container::multiset::reduce_by_key
                                                ygm::container::set::local_insert (C++ func-
        (C++ function), 35
                                                         tion), 36
ygm::container::multiset::self_type
                                         (C++ ygm::container::set::local_size(C++ function),
        type), 33
ygm::container::multiset::serialize
                                         (C++ ygm::container::set::operator= (C++ function),
       function), 34
ygm::container::multiset::size (C++ function), ygm::container::set::partitioner (C++ mem-
                                                         ber), 37
ygm::container::multiset::size_type
                                         (C++
                                                ygm::container::set::reduce (C++ function), 37
                                                ygm::container::set::reduce_by_key (C++ func-
        type), 33
ygm::container::multiset::swap (C++ function),
                                                         tion), 37
                                                ygm::container::set::self_type (C++ type), 35
ygm::container::multiset::transform
                                          (C++
                                                ygm::container::set::serialize (C++ function),
       function), 35
ygm::container::multiset::value_type
                                                ygm::container::set::set (C++ function), 36
                                         (C++
       type), 33
                                                ygm::container::set::size(C++ function), 37
ygm::container::set(C++ class), 35
                                                ygm::container::set::size\_type(C++ type), 35
ygm::container::set::\simset (C++ function), 36
                                                ygm::container::set::swap (C++ function), 37
ygm::container::set::async_contains
                                         (C++
                                                ygm::container::set::transform (C++ function),
       function), 36
ygm::container::set::async_erase (C++ func- ygm::container::set::value_type (C++ type), 35
        tion), 36
ygm::container::set::async_insert (C++ func-
       tion), 36
ygm::container::set::async_insert_contains
        (C++ function), 36
ygm::container::set::clear (C++ function), 37
ygm::container::set::collect (C++ function), 37
ygm::container::set::comm(C++ function), 37
ygm::container::set::container_type
                                         (C++
        type), 35
ygm::container::set::count (C++ function), 37
ygm::container::set::deserialize (C++ func-
        tion), 36
ygm::container::set::erase(C++ function), 36
ygm::container::set::filter(C++ function), 37
ygm::container::set::flatten (C++ function), 37
ygm::container::set::for_all (C++ function), 37
ygm::container::set::for_all_args (C++ type),
ygm::container::set::gather (C++ function), 37
ygm::container::set::gather_topk (C++ func-
        tion), 37
ygm::container::set::get_ygm_ptr (C++ func-
        tion), 37
ygm::container::set::key_type (C++type), 35
ygm::container::set::local_clear (C++ func-
        tion), 36
ygm::container::set::local_count (C++ func-
        tion), 36
ygm::container::set::local_erase (C++ func-
        tion), 36
```