FUJITSU

shaping tomorrow with you

Basic Layer and Data Library of ACP (Advanced Communication Primitives) Library

Takafumi Nose
(Fujitsu/JST-CREST)

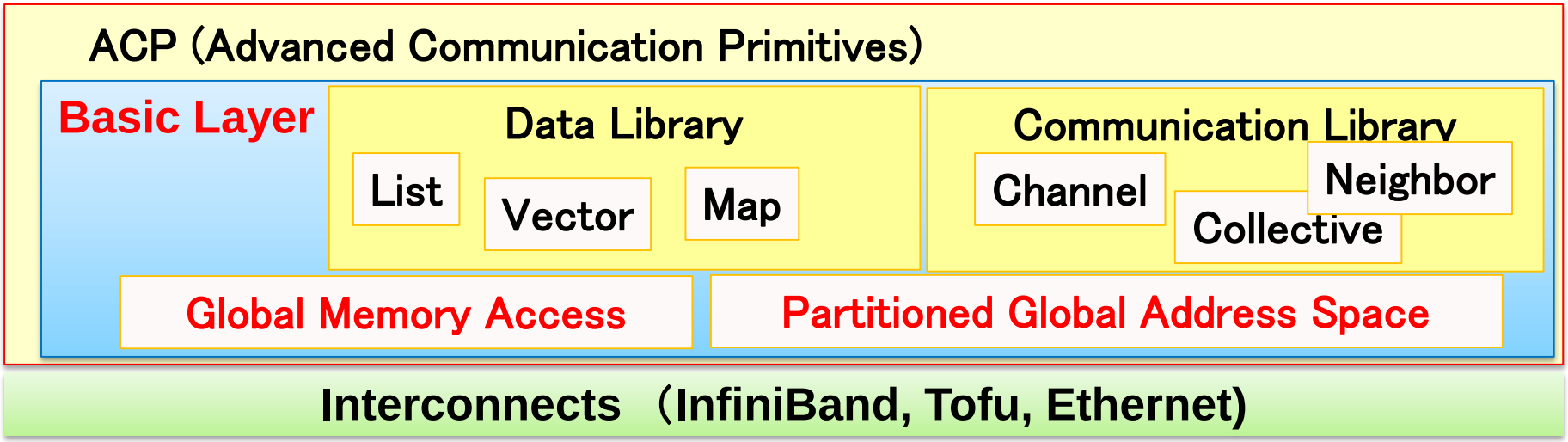
FUJITSU

Basic Layer

Design Policies of ACP Basic Layer

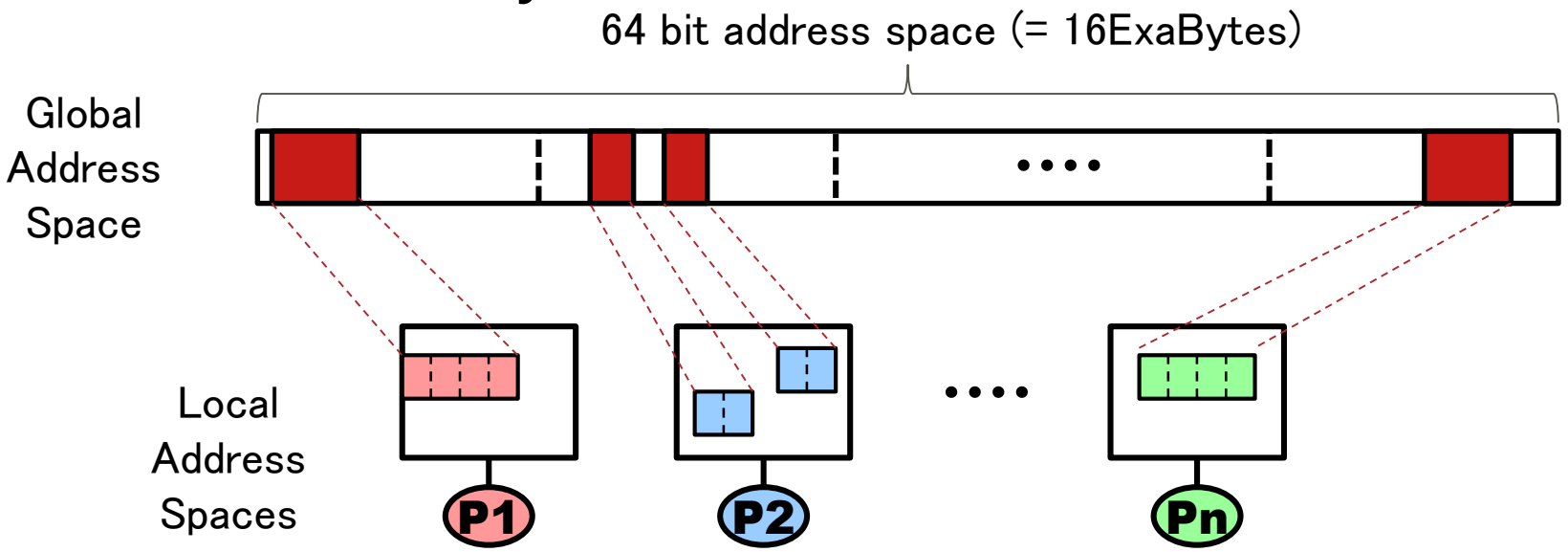
FUJITSU

- Design Policies:
  - For Memory Efficient Communication： Explicitly Controlled
    - Implicit Control is difficult to realize least memory usage
  - For Low Latency Data Exchange： RDMA Based
    - Low Latency and Data Exchange without CPU processing
- ACP Basic Layer Provides RDMA based Global Memory Access



Memory Model Design

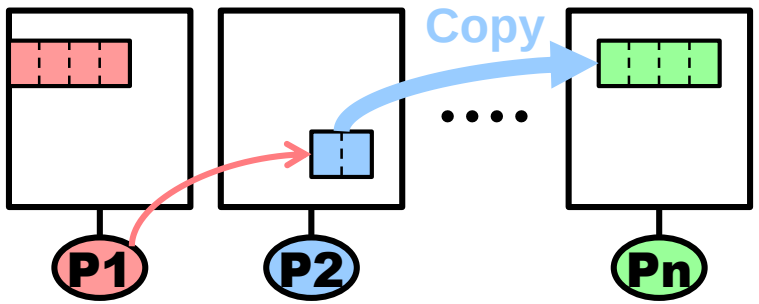
- 64 bit Global Memory Address so that remote atomic memory operation can be used.
  - Support to build distributed data structures
- Memory Registration APIs
  - Users allocate local memory and register it to Global Memory.



Data Transfer Method

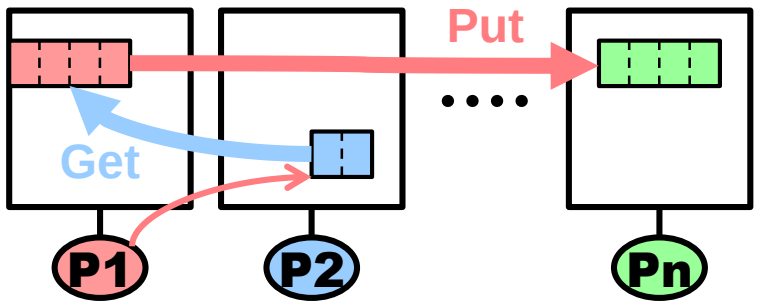
- Global Memory Access: GMA
  - Global Memory Copy between Distributed Processes including Non-Local Processes
  - Aimed to Optimize Asynchronous Data Transfer with Complicated Communication Pattern
- Existing Work: MPI Remote Memory Access(RMA)
  - Data Transfer between Distributed Memory and Local Memory Controlled by source or destination process.

GMA



Copy Request

Conventional Put/Get



Get Request

Data Transfer API: acp_copy

FUJITSU

```
acp_handle_t acp_copy(acp_ga_t dst, acp_ga_t src,
      acp_size_t size, acp_handle_t order);
```

Argument:

dst: Destination Global Address

src: Source Global Address

size: Copy Size

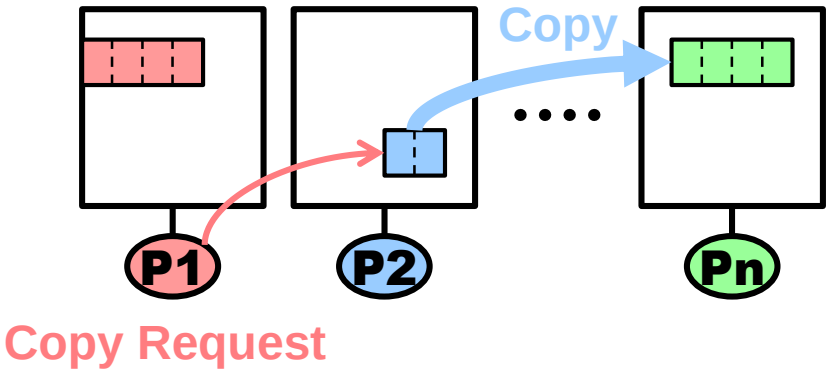
order: Specify acp_handle_t or ACP_HANDLE_ALL/NULL

Return Value:

!= ACP_HANDLE_NULL: GMA Handle

== ACP_HANDLE_NULL: Fail

```
acp_handle_t: ACP Handle
acp_ga_t:     ACP Global Address
acp_size_t:   ACP Data Size
```



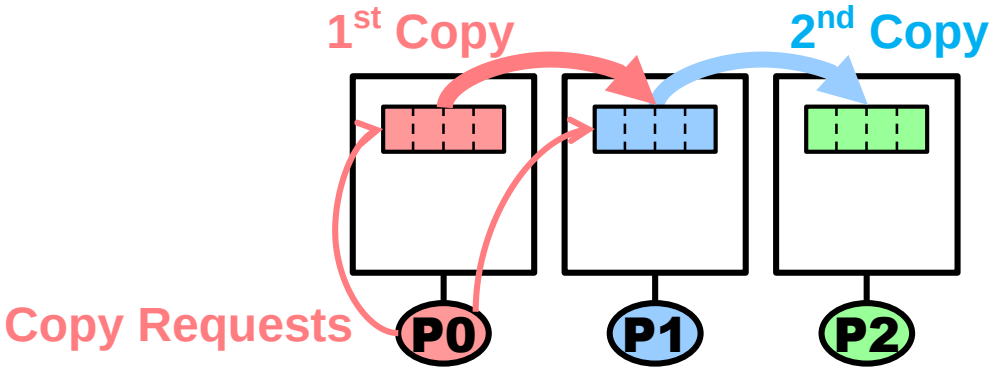
Sample Code of GMA

FUJITSU

```
acp_handle_t h0,h1;

/* Global addresses of 4-byte buffers.
   They belong to rank0, rank1, and rank2, respectively. */
acp_ga_t ga0, ga1, ga2;

/* this copy starts immediately */
h0 = acp_copy(ga1, ga0, 4, ACP_HANDLE_NULL);
```

Dependency

```
/* issued AFTER the completion of the first copy */
h1 = acp_copy(ga2, ga1, 4, h0);

acp_complete(h1); // wait completion of the second copy
```



ACP Basic Layer(ACPbl) Implementations

FUJITSU

- Initial implementations of the ACPbl:

  UDP (Ethernet), InfiniBand (IB) and Tofu Interconnect Versions

  - UDP Version: Developed as a Reference Implementation for validation
  - IB and Tofu Versions: Developed as Practical Execution Environments

Memory Usage Estimation@ 1M Procs.

FUJITSU

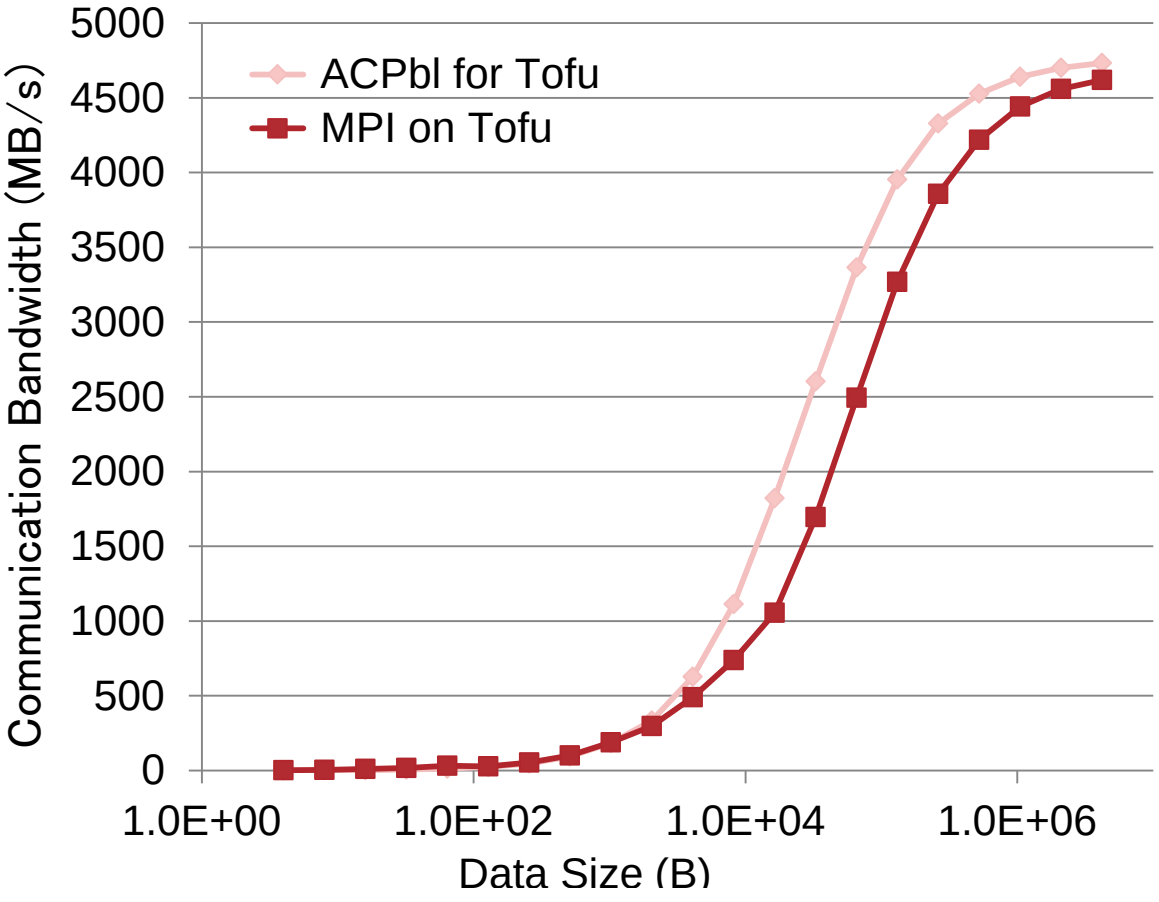
	Tofu	UDP
Per Number of Processes	69MB@1M Processes • Command Receive Buffer 64B • Tofu Address Table 4B • Tofu Routing Table 1B	18MB@1M Processes • IP Address 4B、Port Number 2B • Send Sequence Number 4B、Receive Sequence Number 4B • Process Number 4B
Per Memory Entry	9KB@128 Entries • Registration Table 40B • Address Lookup Table 16B/Proc. ＋ 8bytes×256（Fixed）	
Misc	262KB • Command Queue and Buffers 64B×4,096	647KB • Command Queue 80B×4,096 • Command Station 1,504B×64 • Delegate Station 3,480×64
Total@1M Processes	約70MB	約19MB

Only 70MB of memory for 1 Million Processes on Tofu

Preliminary ACP Basic Layer Performance

FUJITSU

- Reference Implementation has been finished and ACP Basic Layer performance is better than that of MPI.



ACP Basic Layer Performance on Tofu

Evaluation System:
System: Fujitsu PRIMEHPC FX10
CPU: SPARC64™ IXfx
Memory: 32GB/node

FUJITSU

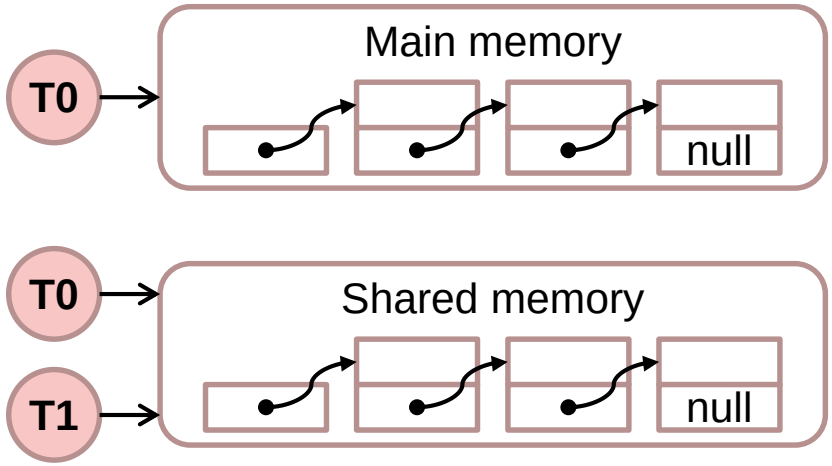
Data Library

Data Structure

FUJITSU

- Why data structure?
  - To handle irregular data
    - Ex. particle methods, sparse matrices, unstructured meshes
  - For sophisticated memory management
    - Reduce, Reuse, Recycle

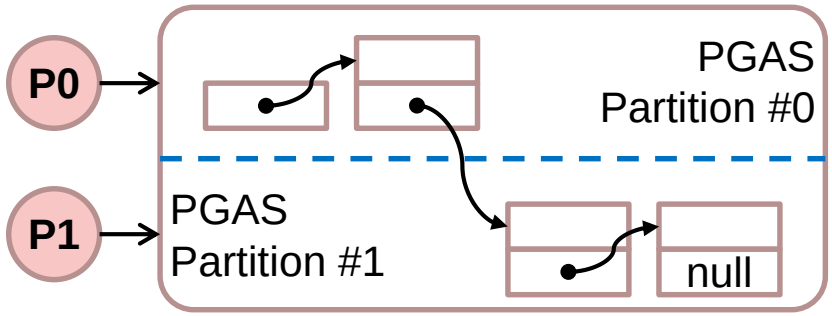
- Ordinary data structure
  - Main memory
  - Single thread
- Concurrent data structure
  - Shared memory
  - Multiple threads



Distributed Data Structure on PGAS

FUJITSU

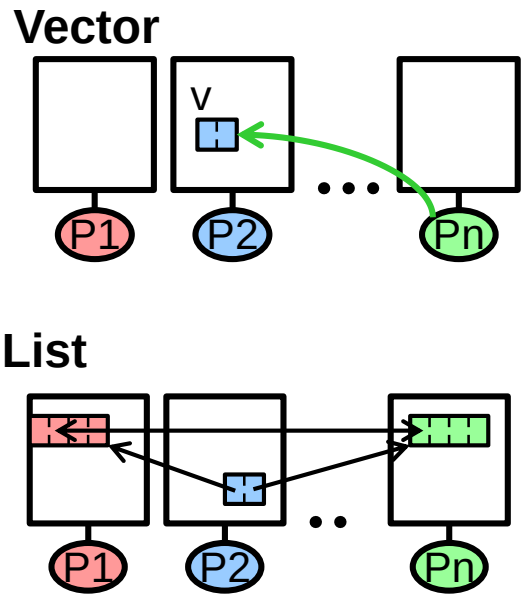
- Distributed data structure
  - Distributed memory
  - Multiple processes
  - Partitioned Global Address Space



- Why PGAS?
  - Locality-awareness for performance
  - Global address is easy to implement reference semantics
    - Allowing wrapper to be written for high-level language

Data Library of ACP

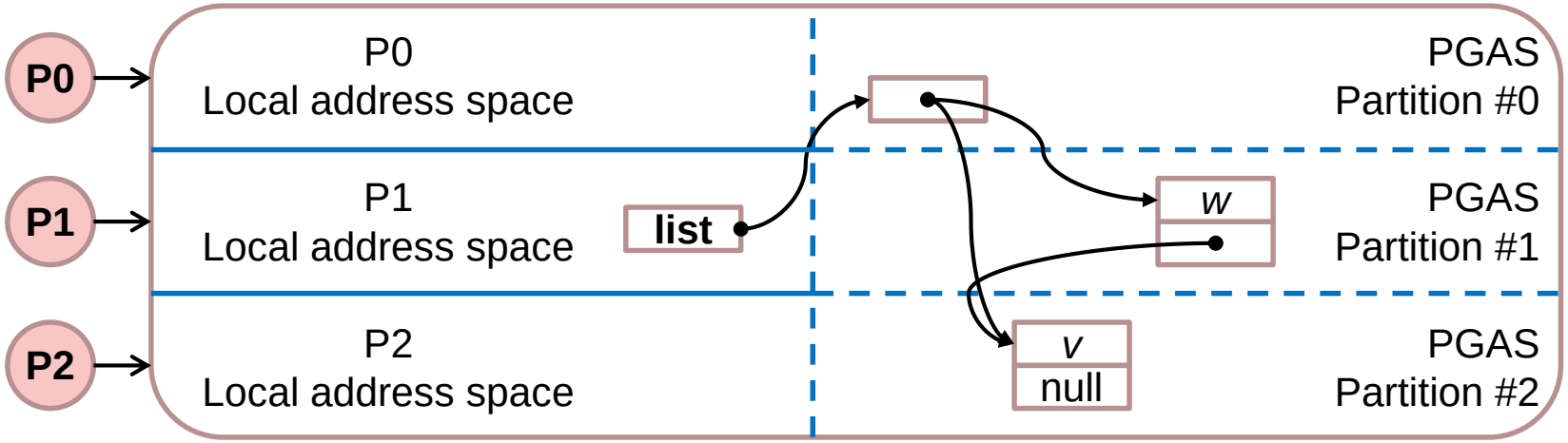
- Similar to C++ Standard Template Library
  - But data-type agnostic C library
- Five data structure types
  - acp_vector_t variable-length array
  - acp_list_t bi-directional linked-list
  - acp_deque_t bi-directional queue
  - acp_set_t unordered collection
  - acp_map_t unordered key-value store
- Iterators
  - Ex. acp_vector_it_t, acp_list_it_t, acp_map_it_t
- Functions
  - Ex. acp_create_vector(), acp_destroy_list(), acp_insert_map()



Data Placement Control

- Each object or element can be placed on different process ranks
- Example) executed at P1

```
list = acp_create_list(0); // on #0
acp_push_back_list(list, v, sizeof(int)); // on #2
acp_push_front_list(list, w, sizeof(int)); // on #1
/* v and w are global addresses belong to rank2 and rank1,
   respectively */
```



Iterator

FUJITSU

- An iterator is a reference to an element of an object
  - Iterator abstracts index and pointer to an element
  - Iterator also contains the object information
- Function **begin** provides the iterator of the first element
- Function **end** provides the iterator of the end sentinel
- An iterator can be **increment**ed, **decrement**ed or **dereferenc**ed

Ex. Iterator functions

acp_<type>_it_t **acp_begin_<type>**(acp_<type>_t *object*);

acp_<type>_it_t **acp_end_<type>**(acp_<type>_t *object*);

acp_<type>_it_t **acp_increment_<type>**(acp_<type>_it_t *iter*);

acp_ga_t **acp_dereference_<type>**(acp_<type>_it_t *iter*);

Iterator – Sample Code

FUJITSU

- To count number of keys in a set

```
acp_set_t set;
acp_set_it_t iter;
int count;
…

count = 0;
for (iter = acp_begin_set(set);
     iter != acp_end_set(set);
     iter = acp_increment_set(iter))
{
    count++;
}
```

Global Memory Allocator

FUJITSU

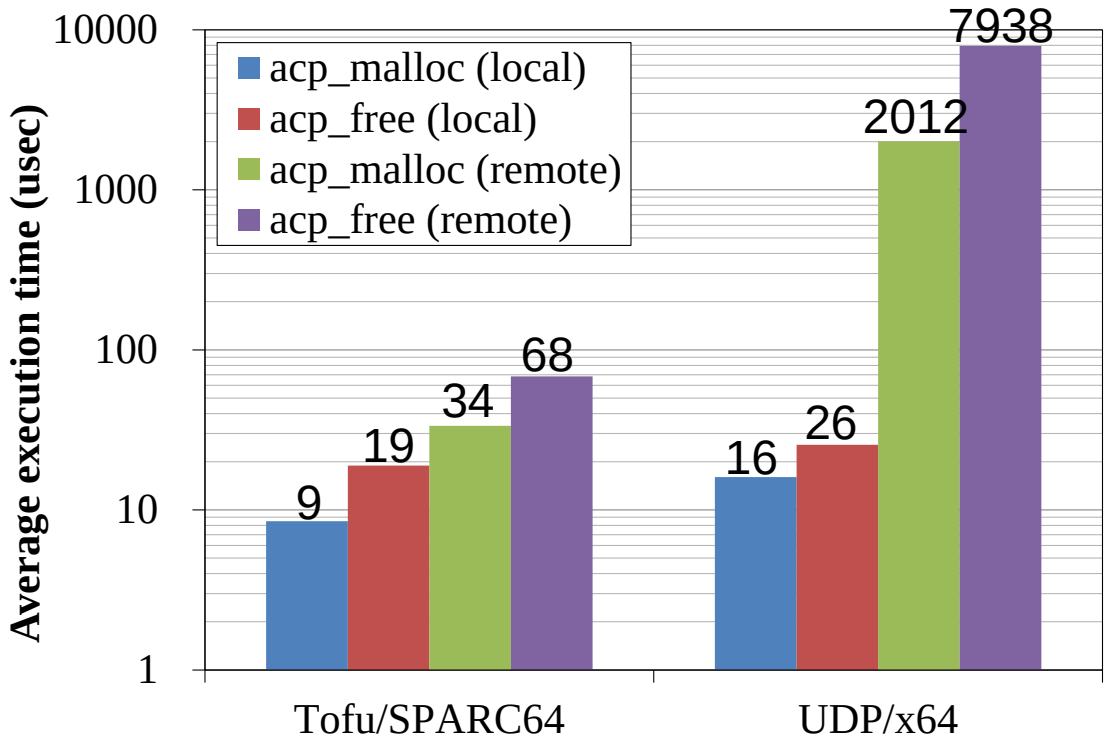
- Implemented for internal use
  - The Basic Layer only registers local memory
  - The Data Library requires dynamic allocator of remote memory
- acp_malloc()
  - Allocates specified size of remote memory on specified rank
  - Returns start global address of the allocated memory
- acp_free()
  - Deallocates an allocated memory specified by start global address

Global memory allocator functions

acp_ga_t **acp_malloc**(size_t *size*, int *rank*);

void **acp_free**(acp_ga_t *ga*);

Random Allocation/Deallocation Time



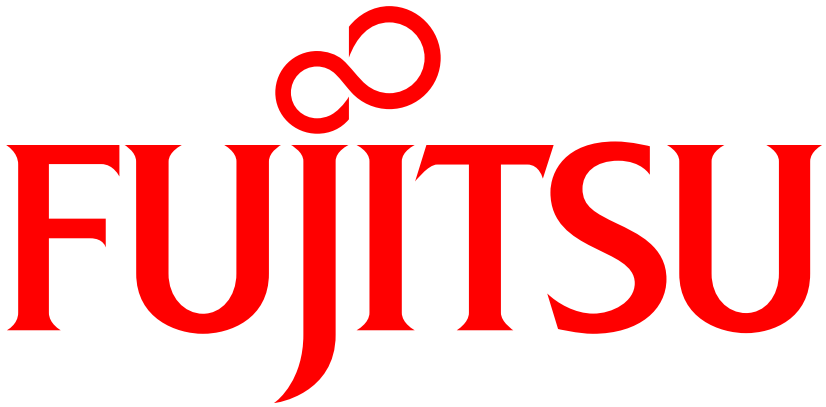
	Tofu/SPARC64	**UDP/x86**
CPU	Fujitsu SPARC64™ IXfx	Intel Xeon E5520
Network	Tofu interconnect (5.0 GB/s)	Gigabit Ethernet (125 MB/s)
ACP Basic Layer	Tofu version	UDP version

Summary

FUJITSU

- Basic Layer:
  - For Memory Efficient Communication： Explicitly Controlled
  - For Low Latency Data Exchange： RDMA Based
  - Global Memory Access
  - UDP (Ethernet), IB, and Tofu Implementations

- Data Library:
  - Data-type agnostic C library
  - Data structure types are similar to those of C++ STL
  - Based on Global Memory Allocator



shaping tomorrow with you