

What communication library can do with a little hint from programmers?

Takeshi Nanri (Kyushu Univ. and JST CREST, Japan)

Background

- Various tuning opportunities in communication libraries:

Protocol?

Eager
or Rendezvous
or ...

Transport service?

UD or RC
or Hybrid or...

Algorithm of Collectives?

Ring or Linear
or Binary or ...

- Information available for such tuning is limited:
 - Parameters of each operation
 - Basic information of the system
 - Results of some benchmarks
- As the HPC systems become larger and more complicated, "information about the program" will be required for the effective use of resources.

Motivation

- Provide "hints" (= information of the program) to enable more aggressive tuning in communication libraries.
- Who provides "hints"?
 - Programmers, Compilers of high level languages, Scientific libraries, etc.
- Target library?
 - MPI libraries (MVAPICH, MPICH, Open MPI, etc.)
 - Other communication libraries (ARMCI, GASNet, etc.)
 - There can be some hints that are commonly useful for various libraries.

Proposal: Hint API

- `int Hint(char *key, char *value);`
 - key: name of the hint
 - value: value applied to the hint
 - return value: 0=success, -1=fail(no such hint)
- Example:
 - `Hint("IN_KERNEL", "1");`
- Behavior of the function:
depends on the implementation of the library.

Sample usage of hint:

Hint about "kernel region" of the program

```

main()
{
  MPI_Init();

  Initialization } May include some Broadcasts. No need for high-speed.

  Main loop{
    Preparation } May include some collectives. No need for high-speed.

    Hint("IN_KERNEL", "1") ← start of a kernel region

    Kernel region } May include neighbor communications, some patterns of
    send/recv or collectives. Need to be very fast.

    Hint("IN_KERNEL", "0") ← end of a kernel region
  }

  Finalization } May include some collectives. No need for high-speed.

  MPI_Finalize();
}

```

What kind of tuning can be done with the hint of the "kernel region"?

- Example) Adaptive choice of transport services:
 - Transport services of InfiniBand:
 - UD (Unreliable Datagram)
 - Low memory usage
 - Low bandwidth
 - RC (Reliable Connection)
 - High memory usage
 - High bandwidth
 - Establish RC only for the communications in the kernel region.

Effect of the adaptive choice of transport service

- Number of peers in kernel region can be a small number, if the communications in the region are:
 - Neighbor communication:
 - 1 (shift), 2 (1D), 4~8(2D), 6~26 (3D)
 - Collectives with tree algorithm:
 - 2 (binary tree algorithm)
 - a pattern of sends/recvs with specific peers
- Hint about "kernel region" enables effective memory usage

Sample implementation of Hint function

```
int param1=0, param2=0, param3=0;

#define HINTS_LEN 3

typedef struct HINT_item{
    char *key;
    int *val;
} HINT_item_t;

static HINT_item_t HINTs[HINTS_LEN]={
    {"key1", &param1},
    {"key2", &param2},
    {"key3", &param3}
};

int Hint(char *key, char *val)
{
    int i;

    for (i = 0; i < HINTS_LEN; i++){
        if (strcmp(HINTs[i].key, key)==0){
            *HINTs[i].val = atoi(val);
            return 0;
        }
    }

    printf("WARN: key\n");
    return -1;
}
```

Structure of Hint:

Map a "key" to the corresponding parameter.

Table of Hints:

Associate a system parameter to each of the keys.

Search the "key" in the Hint table,
and apply the value to the corresponding
parameter.

No search for keys needed at the
reference of the hint.

Modification of MVAPICH2-1.9 to accept the hint of "kernel region"

- Add a file to define the Hint function with one Hint item:
{ "IN_KERNEL", &HINT_in_kernel }
- Modify Hybrid UD/RC send function:
 - Original code:
 - Establish an RC to the peer
 - if the number of messages to it has exceeded the threshold,
 - and the number of RCs less than the limit.

```
if ((total_messages > msg_limit) &&  
    ((rc_connections + pending_rc_conn) < max_rc_conn))  
    Establish_RC_Connection
```

- New code:
 - Establish an RC to the peer
 - if it is in the **"kernel region"**,
 - and the number of RCs is less than the limit.

```
if ((HINT_in_kernel) &&  
    ((rc_connections + pending_rc_conn) < max_rc_conn))  
    Establish_RC_Connection
```

Hint API vs. MPI_Info

- MPI_Info: a key-value system in MPI standard
 - Functions:
 - MPI_Info_create(&info) : create an info
 - MPI_Info_set(info, key, val) : set a key-value pair to the info
 - Usage of info: apply info to the function that accepts it
 - Ex) MPI_Comm_spawn(..., info, ...);
- MPI_Info is not suitable for providing "hint" of the program:
 - Need to search key in the info at each time the library wants to refer.
 - Supported functions are limited.
 - MPI_Comm_spawn, MPI_Comm_connect, MPI_Win_create, MPI_Alloc_mem, MPI_File_open, etc.

Preliminary experiment

- Examine the effect of Hint about "kernel region" in the worst case.
 - Test program:
 - Repeat two communications:
 - in Preparation: 100 times of small (64byte) alltoall communication
 - in Kernel Region: 10000 times of large (100byte ~ 1MB) shift communication

```

main()
{
    MPI_Init();

    for (r = 0; r < 100){
        for (i = 0; i < 100; i++){
            for (p = 1; p < procs; p++){
                MPI_Isend(sbuf, 64, MPI_CHAR, (myrank+p)%procs, 0, MPI_COMM_WORLD, &req);
                MPI_Recv(rbuf, 64, MPI_CHAR, (myrank-p+procs)%procs, &stat);
                MPI_Wait(&req, &stat);
            }
            Hint("IN_KERNEL", 1);
            for (i = 0; i < 10000; i++){
                MPI_Isend(sbuf, m, MPI_CHAR, (myrank+1)%procs, 0, MPI_COMM_WORLD, &req);
                MPI_Recv(rbuf, m, MPI_CHAR, (myrank-1+procs)%procs, &stat);
                MPI_Wait(&req, &stat);
            }
            Hint("IN_KERNEL", 0);
        }
    }
    MPI_Finalize();
}
  
```

Alltoalls in the preparation.
Can be slow.

Shifts in the kernel region.
Need to be very fast.

Experimental environment

- PC cluster connected with InfiniBand
 - 128 nodes of 1476nodes
 - 2 x 8core Intel Xeon E5-2680 2.7GHz
 - 128GB RAM
 - HCA: Mellanox MT4099 (FDR)
 - OS: RedHat 6.1 (kernel 2.6.32)
 - Compiler: gcc 4.4.6
 - MPI: MVAPICH2-1.9
 - One process per node.

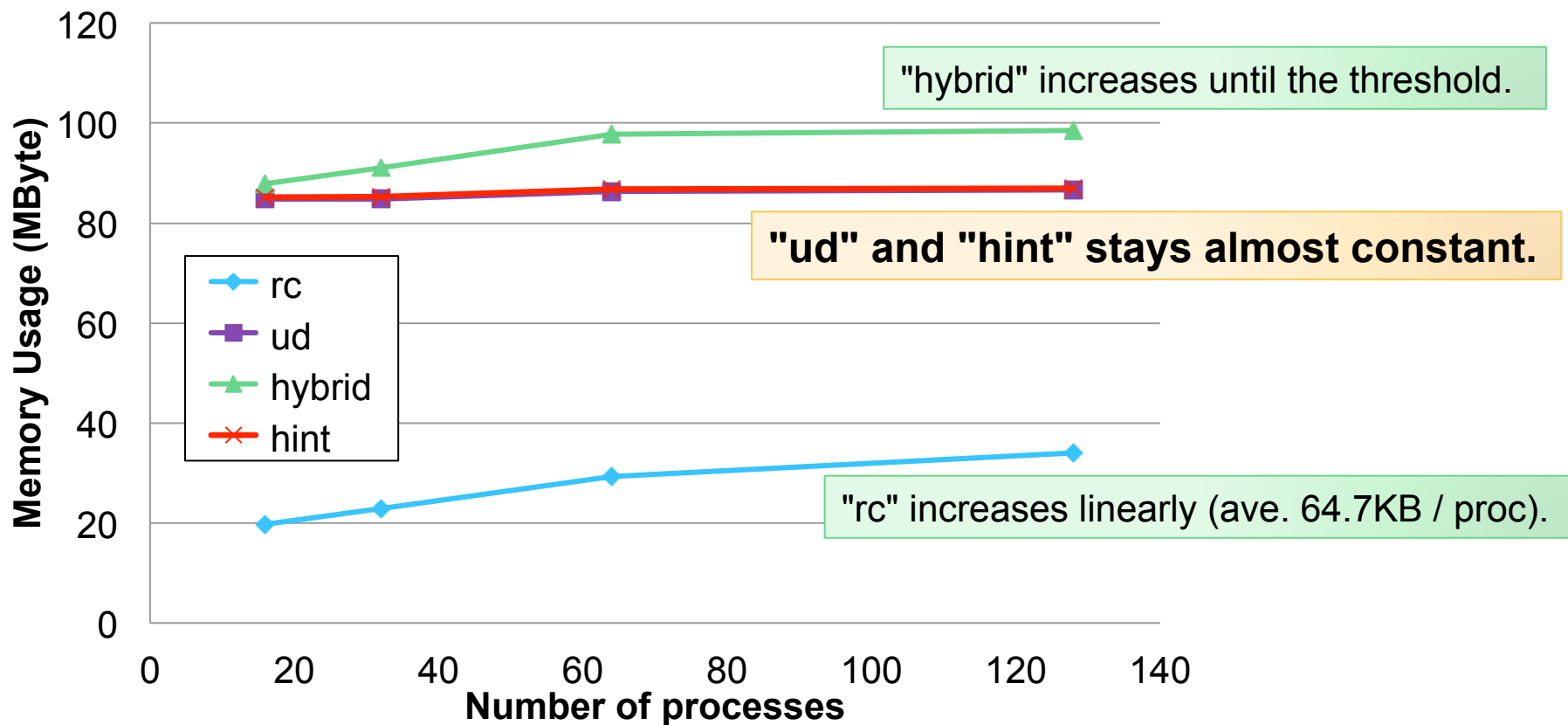
Settings of MVAPICH2

- rc:
 - default configuration
- ud
 - configure --enable-hybrid
 - export MV2_USE_ONLY_UD=1
- hybrid
 - configure --enable-hybrid
 - export MV2_USE_UD_HYBRID=1
 - export MV2_HYBRID_ENABLE_THRESHOLD=1
- hint
 - Codes are modified to accept Hint about "kernel region"
 - configure --enable-hybrid
 - export MV2_USE_UD_HYBRID=1
 - export MV2_HYBRID_ENABLE_THRESHOLD=1
 - export MV2_UD_NUM_MSG_LIMIT=1
 - export MV2_HYBRID_MAX_RC_CONN=128

Threshold of the number of comms for switching from UD to RC

Effect of hint on memory usage

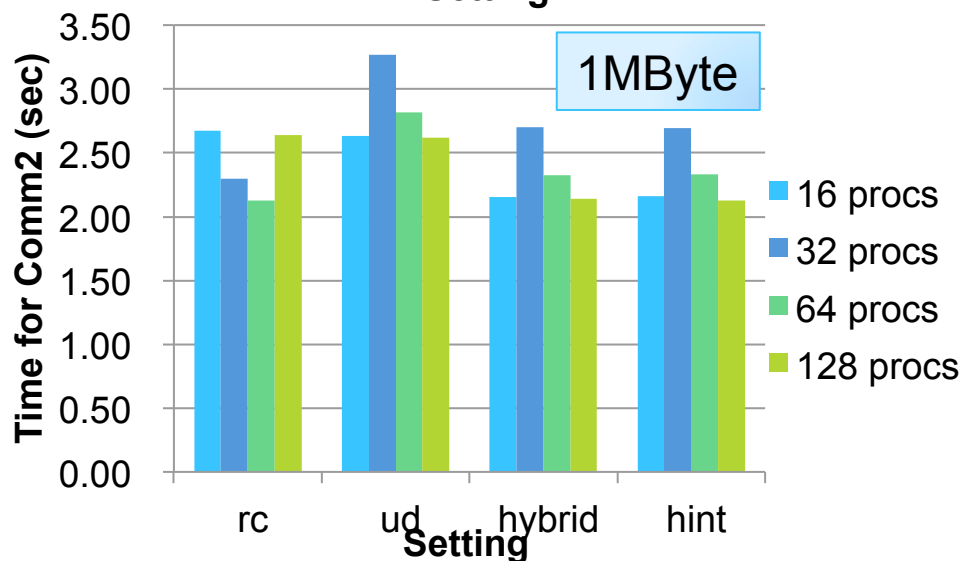
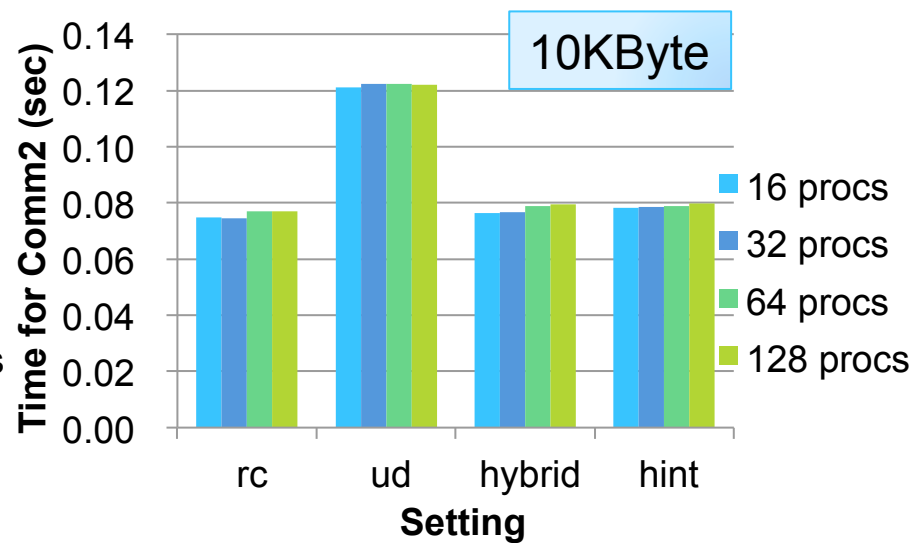
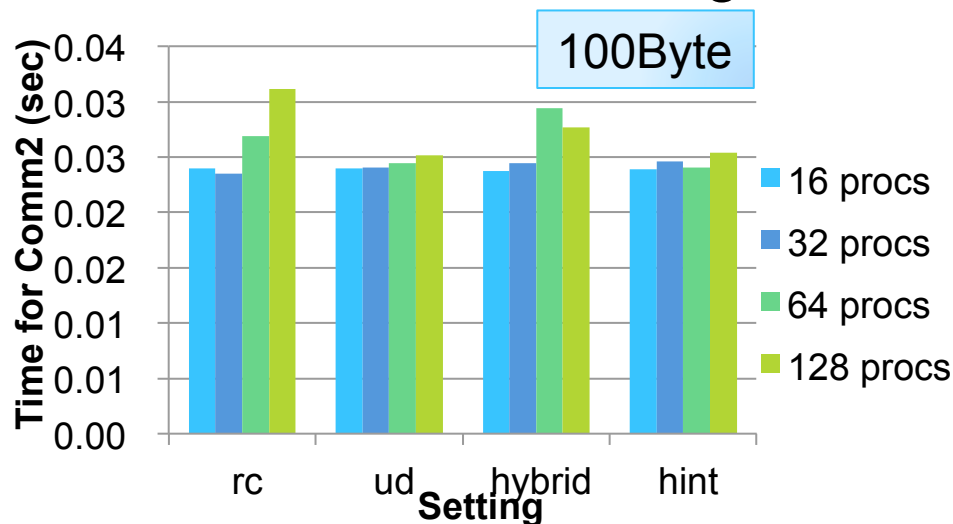
- Memory usage per process



RC will require more than 6GB on 100,000 nodes, while UD, HYBRID and HINT remains constant. (HYBRID depends on the threshold.)

Effect of hints on performance

- Time for Shift communications (10000times) in the kernel with different message sizes.



- "hint" shows the best performance in most of the cases.
- "hybrid" will slow down after it reaches to the limit of RC connections.

A little more realistic case

• Program:

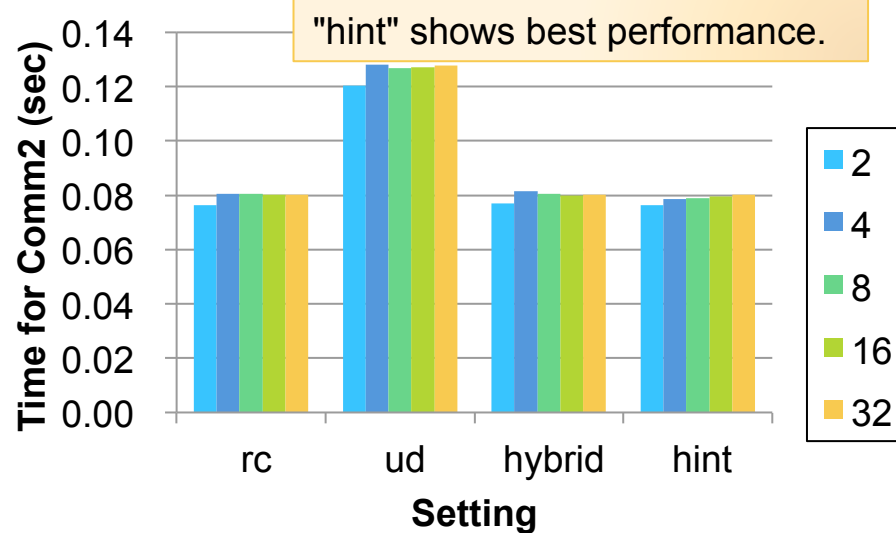
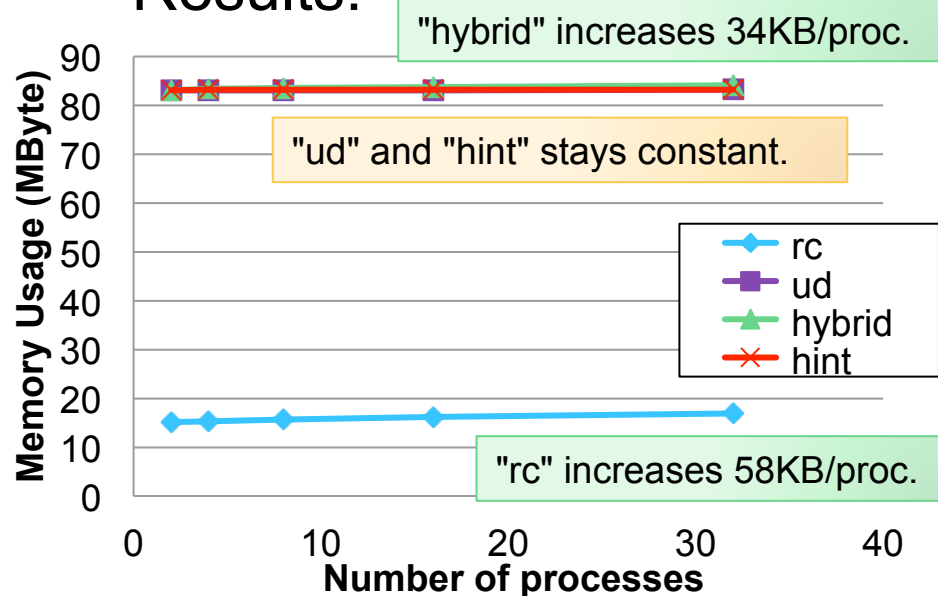
```

for (r = 0; r < 100){
  for (i = 0; i < 10; i++){
    MPI_Allreduce();
  }

  Hint("IN_KERNEL", 1);
  for (i = 0; i < 10000; i++){
    Shift communication from neighbor to neighbor.
  }
  Hint("IN_KERNEL", 0);
}
  
```

Change preparation communication
from 100 times of Alltoalls
to 10 times of Allreduces.
... Requires less connections on RC.


• Results:



Conclusions and Future Works

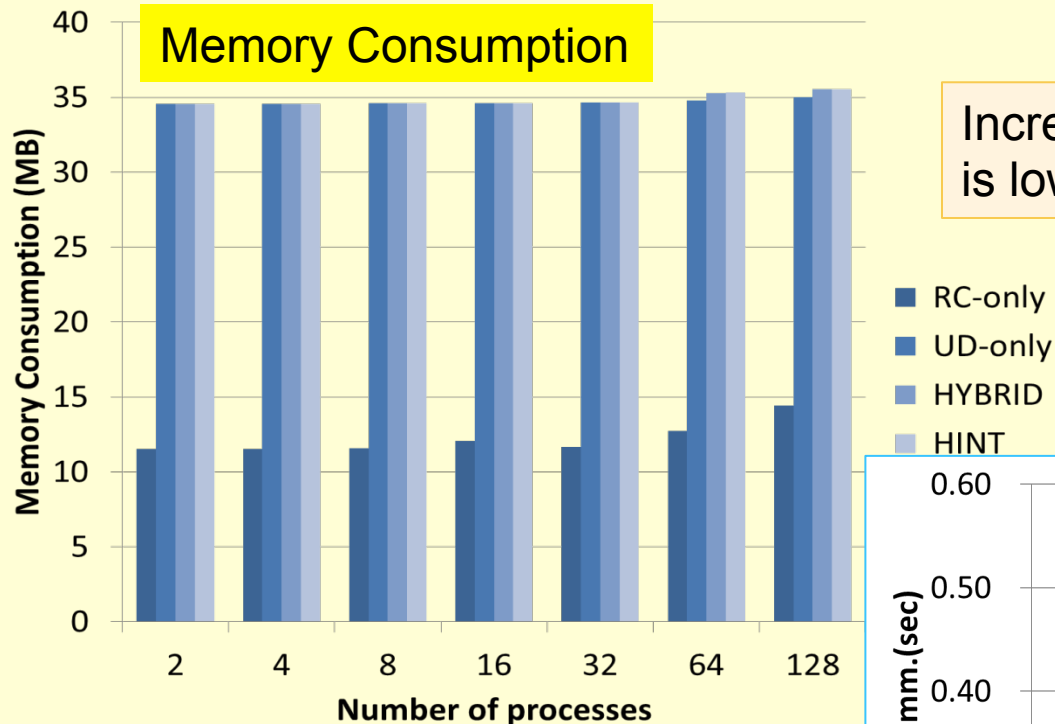
- Proposed an API for providing "Hint" of programs
 - Enable aggressive tuning in communication libraries.
- Examined the effect of hint about "kernel region"
 - Memory efficient choice of the transport service, UD or RC.
- More experiments with:
 - larger environment
 - other benchmarks and applications
- Examine other "Hints".
- (If this API is accepted to be useful)
Prepare a proposal of this API for MPI-3.x or MPI-4 standard.

Experiments on NAS Parallel Benchmark

- Examined programs: MG, LU and FT
 - Communications are invoked in both of the kernel region and the preparation (or summarization) part of the program.
- Measurement of memory consumption
 - Memory consumption by communication library must be determined.
- Use DMATP-MPI (by FUJITSU Ltd.)
 - Hook memory related functions (malloc, memalign and free) to keep logs of increase and decrease of memory consumption.
 - To appear in EuroMPI 2013 (poster)
 - Logs are summarized according to the library that invoked the function.
 - > Peak amount of memory consumed by MVAPICH
 - This tool is not thread-safe
 - ... "registration-cache" of MVAPICH is disabled.

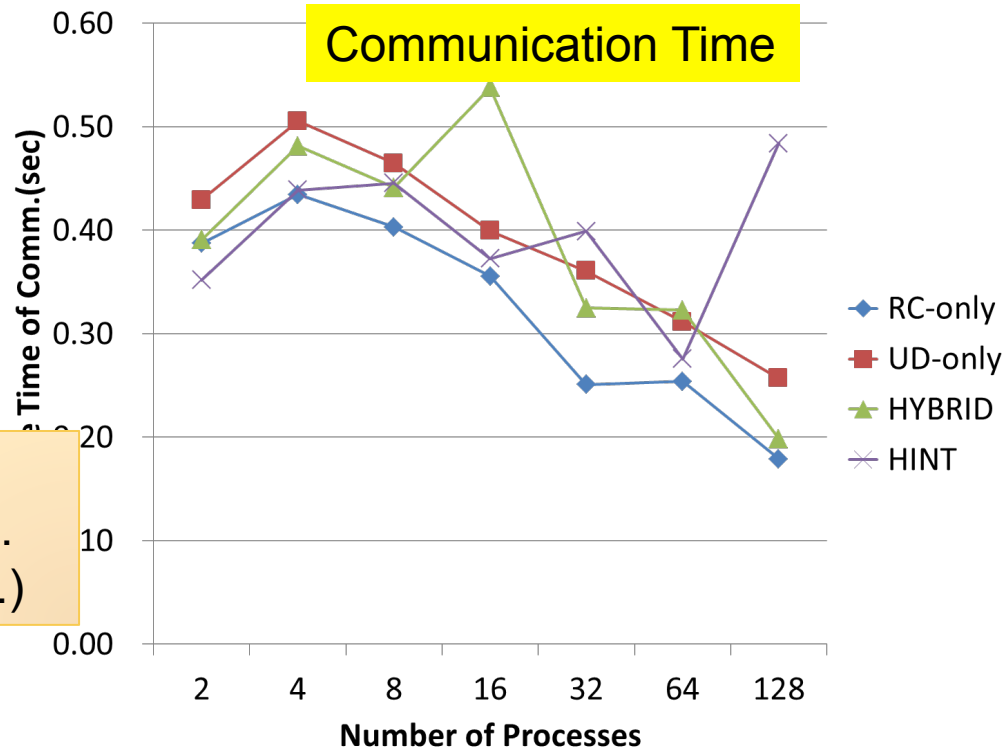
MG

Memory Consumption



Increase of memory consumption is low on UD, HYBRID and HINT.

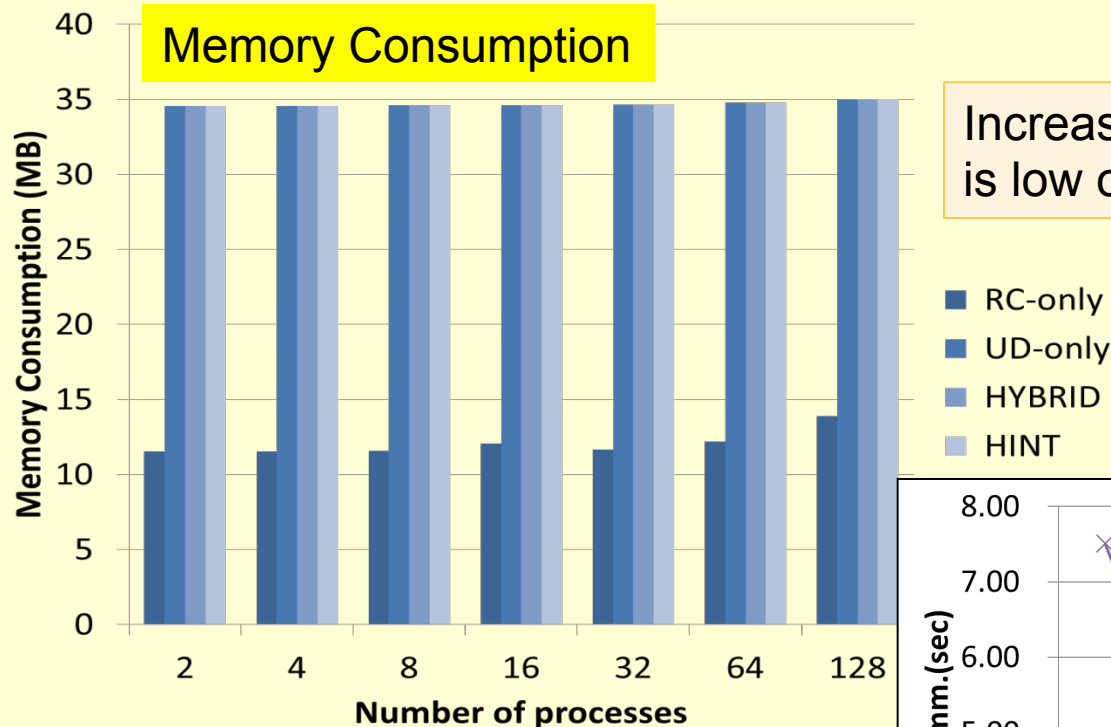
Communication Time



Speed of communication with HINT is close to the case with RC, in most cases. (There are some strange points though...)

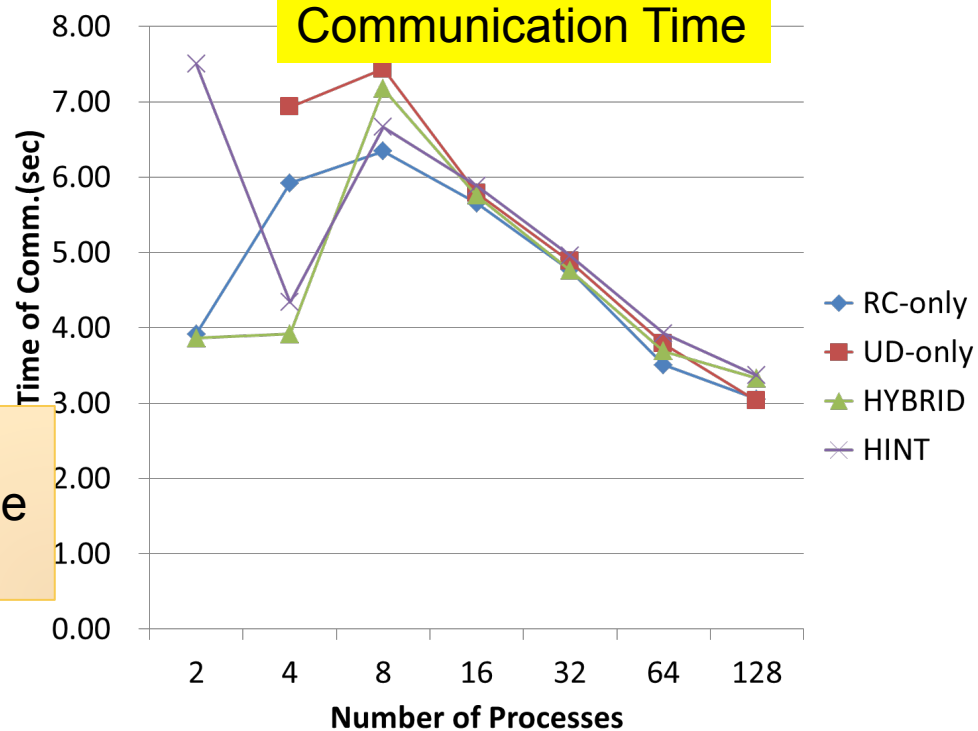
LU

Memory Consumption



Increase of memory consumption is low on UD, HYBRID and HINT.

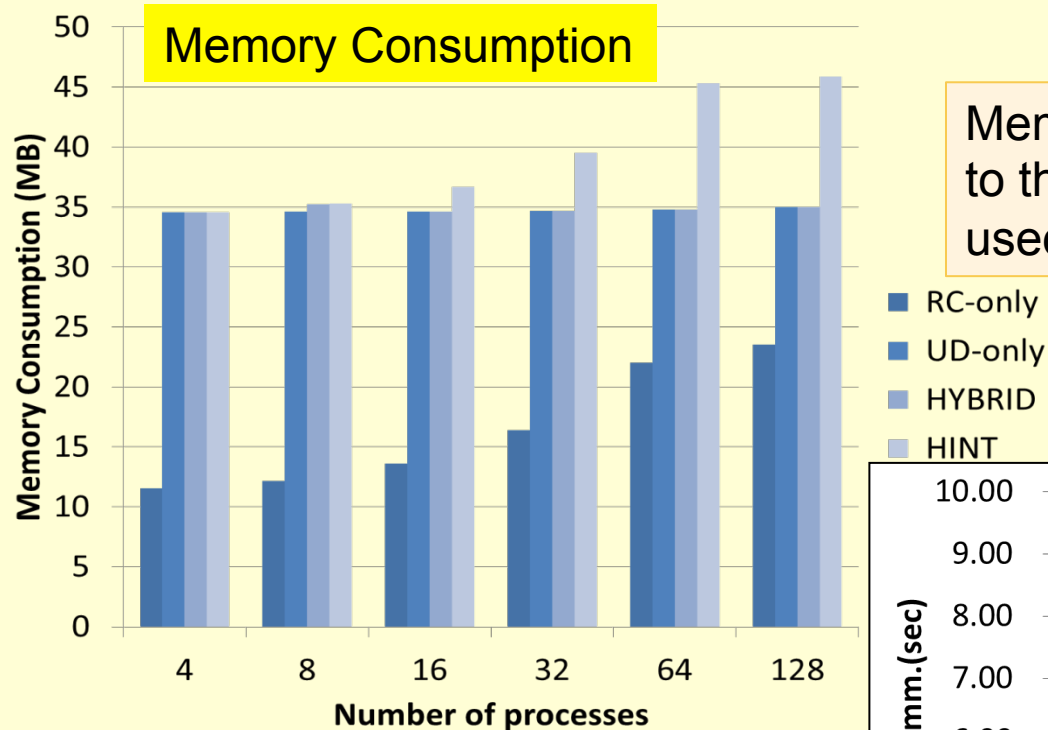
Communication Time



Difference of the communication time is not big among four settings, because message sizes are small.

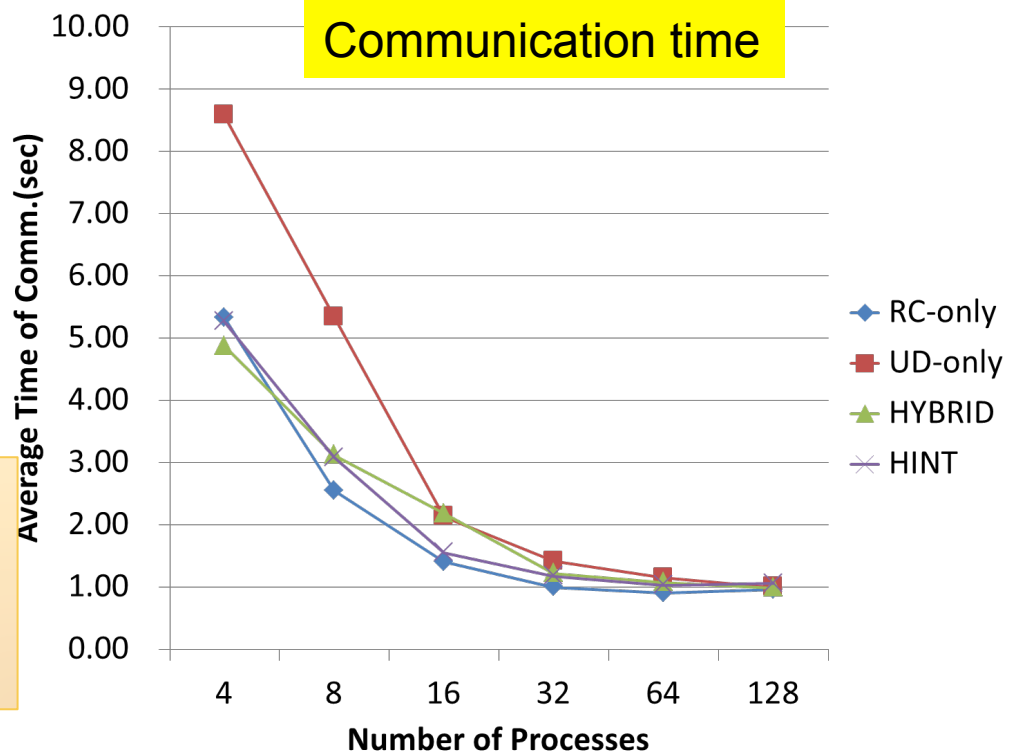
FT

Memory Consumption



Memory consumption with HINT is similar to the case with UD, because Alltoall is used in the kernel region.

Communication time



RC and HINT perform better than UD. Advantage decreases for larger number of processes, because the message size becomes smaller.

Acknowledgement

- This work is a part of ACE (Advanced Communication library for Exa) project (<http://ace-project.kyushu-u.ac.jp/main/en/index.html>) of JST CREST, Japan.
- Members of ACE project:
 - Kyushu Univ.
 - T. Nanri, T. Takami, K. Fukazawa, H. Honda, R. Susukita, Y. Morie and H. Sugiyama
 - Fujitsu Ltd.
 - S. Sumimoto, Y. Ajima, K. Miura, T. Okamoto, H. Akimoto, K. Saga, T. Adachi, T. Nose, H. Imade and R. Kanbayashi
 - Institute of Systems, Information Technologies and Nanotechnologies
 - H. Shibamura and T. Soga