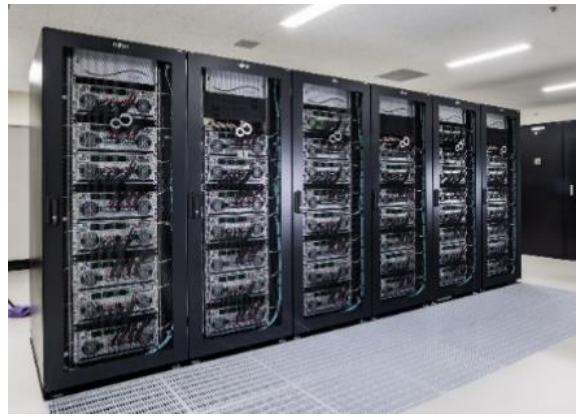
Simple WaitIO-Hands On Using Wisteria/BDEC-01

Shinji Sumimoto
The University of Tokyo



WaitIO API again

- WaitIO API is very simple
  - Group Creation、isend(), irecv(), wait()+misc funcsions
    - TAG Support(64bit), No ANY_SOURCE and ANY_TAG support

WatiIO API	Description
waitio_isend	Non-Blocking Send
waitio_irecv	Non-Blocking Receive
waitio_wait	Wait isend or irecv
waitio_init	WaitIO initialization
waitio_finalize	WaitIO finalization
waitio_get_nprocs	Get number of ranks of each PB
waitio_create_group waitio_create_group_wranks	PB Group Creation by member list or selecting function
waitio_group_rank	Get Rank number in a group (MPI_Comm_rank)
waitio_group_size	Get Group size
waitio_pb_size	Get PB(Paralell Block) size
waitio_pb_rank	Get Rank number in PBs

WaitIO API: API Description(1)

- Initialization API: int waitio_init(int timeout);
  - Same as MPI_Init() in MPI library. All process must call this function.
    After waitio_init() called, wait the “timeout” seconds until all PBs are ready.
- Get number of ranks of each PB API : int waitio_get_nprocs(int ary[]);
  - Returns the number of PBs in WAITIO Instance (same number of shell environment variable WAITIO_NPB)
  - The "Ary” includes each number of ranks(processes) in each PB
- Finalization API : int waitio_finalize();
  - Finalization function. After calling the function, waitio_init() can not be called again.
- WaitIO Group Creation API(1) :
  waitio_group_t waitio_create_group(ing gid, waitio_filter_func_t[], int order[]);
  - PB Group Creation by selecting function(waitio_filter_func_t), and ordering PBs in the “order[]” array.
- WaitIO Group Creation API(2) :
  waitio_group_t waitio_create_group_wranks(int gid, int *rankap[], int order[]);
  - PB Group Creation by member list array”rankap[]” and ordering PBs in the “order[]” array.

WaitIO API: API Description(2)

- Isend API : int waitio_isend(waitio_group_t group, int dst, char *buf, size_t len, unsigned long tag, waitio_req_t *req);
  - Non blocking send to rank number the “dst” in a group “group”.
- Irecv API : int waitio_irecv(waitio_group_t group, int src, char *buf, size_t len, unsigned long tag, waitio_req_t *req);
  - Non blocking receive from rank number the “src” in a group “group”.
  - No ANY_SOURCE, ANY_TAG support
- Wait API : int waitio_wait(waitio_req_t *req);
  - Wait until isend or irecv finished defined by req.

Simple MPI PingPong Program

```
/* -*- Mode: C; c-basic-offset:4 ; indent-tabs-mode:nil -*- */
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <mpi.h>

int main (int argc, char *argv[]) {
 int  data[2], *buf = data;
 MPI_Status status;
 int ret;
 int rank;

 if((ret = MPI_Init( &argc, &argv)) != MPI_SUCCESS) {
    fprintf(stderr, "MPI_Init failed code %d¥n", ret);
    exit(ret);
 }

 MPI_Comm_rank(MPI_COMM_WORLD, &rank);
 if((rank%2) == 1) {
    if((ret = MPI_Recv(buf, 4,MPI_CHAR, rank-1, 0, MPI_COMM_WORLD, &status)) != 0) {
       fprintf(stderr, "%d MPI_Recv error %d¥n", rank, ret);
    }
    else {
       fprintf(stderr, "rank%d: MPI_Recv from rank%d¥n", rank,  rank-1);
    }
    if((ret = MPI_Send(buf, 4, MPI_CHAR, rank-1, 0, MPI_COMM_WORLD)) != 0) {
       fprintf(stderr, "%d MPI_Send error %d¥n", rank, ret);
    }
    else {
       fprintf(stderr, "rank%d: MPI_Send to rank%d¥n", rank,  rank-1);
    }
 }
```

```
 else {
    if((ret = MPI_Send(buf, 4, MPI_CHAR, rank+1, 0, MPI_COMM_WORLD)) != 0) {
       fprintf(stderr, "%d MPI_Send error %d¥n", rank, ret);
    }
    else {
       fprintf(stderr, "rank%d: MPI_Send to rank%d¥n", rank,  rank+1);
    }
    if((ret = MPI_Recv(buf, 4, MPI_CHAR, rank+1, 0, MPI_COMM_WORLD, &status)) != 0) {
       fprintf(stderr, "%d MPI_Recv error %d¥n", rank, ret);
    }
    else {
       fprintf(stderr, "rank%d: MPI_Recv from rank%d¥n", rank,  rank+1);
    }
 }
 MPI_Finalize();
}
```

- Ping-pong from odd rank processes to even rank processes

Simple PingPong Program by WaitIO

```
/* -*- Mode: C; c-basic-offset:4 ; indent-tabs-mode:nil -*- */#include <mpi.h>
#include "waitio.h"

int truef(int pbid, int n) { return 1; }

int main (int argc, char *argv[]) {
  int ret, wrank;
  waitio_filter_func_t func[4]= {truef, truef, NULL, NULL};
  int  array[4] = {1, 2, 0, 0};
  int  data[2], *buf = data;
  waitio_req_t req;
  waitio_group_t grp1;

  if((ret = MPI_Init( &argc, &argv)) != MPI_SUCCESS) {
    fprintf(stderr, "MPI_Init failed code %d¥n", ret);
    exit(ret);
  }
  if((ret = waitio_init(10)) != 0) {
    fprintf(stderr, "waitio_init failed code %d¥n", ret);
    MPI_Abort(MPI_COMM_WORLD, ret);
    exit(ret);
  }

  grp1 = waitio_create_group(0, func, array);
  if(grp1 == NULL) {
    fprintf(stderr, "waitio_create_group failed code %d¥n", ret);
    MPI_Finalize();
    exit(ret);
  }
 waitio_group_rank(grp1, &wrank);
```

```
if((wrank%2) == 1) {
    waitio_irecv(grp1, wrank-1, (char *)buf, 4, 0, &req);
    if((ret = waitio_wait(&req)) != 0) {
      fprintf(stderr, "%d waitio_irecv error %d¥n", wrank, ret);
    }
    else {
      fprintf(stderr, "rank%d: waitio_irecv from rank%d¥n", wrank,  wrank-1);
    }
    waitio_isend(grp1, wrank-1, (char *)buf, 4, 0, &req);
    if((ret = waitio_wait(&req)) != 0) {
      fprintf(stderr, "%d waitio_isend error %d¥n", wrank, ret);
    }
    else {
      fprintf(stderr, "rank%d: waitio_isend to rank%d¥n", wrank,  wrank-1);
    }
  }
  else {
    waitio_isend(grp1, wrank+1, (char *)buf, 4, 0, &req);
    if((ret = waitio_wait(&req)) != 0) {
      fprintf(stderr, "%d waitio_isend error %d¥n", wrank, ret);
    }
    else {
      fprintf(stderr, "rank%d: waitio_isend to rank%d¥n", wrank,  wrank+1);
    }
    waitio_irecv(grp1, wrank+1, (char *)&buf, 4, 0, &req);
    if((ret = waitio_wait(&req)) != 0) {
      fprintf(stderr, "%d waitio_irecv error %d¥n", wrank, ret);
    }
    else {
      fprintf(stderr, "rank%d: waitio_irecv from rank%d¥n", wrank,  wrank+1);
    }
  }
  waitio_finalize();
  MPI_Finalize();
}
```

WaitIO Initialization/Finalization

```
if((ret = waitio_init(10)) != 0) {
      fprintf(stderr, "waitio_init failed code %d¥n", ret);
      MPI_Abort(MPI_COMM_WORLD, ret);
      exit(ret);
   }
```

```
int truef(int pbid, int n) { return 1; }
```

```
int  array[4] = {1, 2, 0, 0};
waitio_group_t grp1;

grp1 = waitio_create_group(0, func, array);
```

```
waitio_group_rank(grp1, &wrank);
```

```
waitio_finalize();
```

- waitio_init(10)
  - WaitIO Initialization with 10 seconds timeout
- waitio_create_group(0, func, array);
  - Generate a group of processes whose execution result of the “func” function is true==1 for MPI processes with each PBID
  - The “truef” function defines all MPI processes as a WaitIO Group
  - The “array[]” specifies the order of each PB in ascending numerical order and the number of array member needs to be prepared at least that of defined by WAITIO_NPB.
  - waitio_group_rank(grp1, &wrank);
  - Get Rank number in a group
- waitio_finalize();
  - WaitIO Finalization

WaitIOによるPingPongプログラム本体

```
if((wrank%2) == 1) {
    waitio_irecv(grp1, wrank-1, (char *)buf, 4, 0, &req);
    if((ret = waitio_wait(&req)) != 0) {
      fprintf(stderr, "%d waitio_irecv error %d¥n", wrank, ret);
    }
    else {
      fprintf(stderr, "rank%d: waitio_irecv from rank%d¥n", wrank,  wrank-1);
    }
    waitio_isend(grp1, wrank-1, (char *)buf, 4, 0, &req);
    if((ret = waitio_wait(&req)) != 0) {
      fprintf(stderr, "%d waitio_isend error %d¥n", wrank, ret);
    }
    else {
      fprintf(stderr, "rank%d: waitio_isend to rank%d¥n", wrank,  wrank-1);
    }
  }
```

```
else {
    waitio_isend(grp1, wrank+1, (char *)buf, 4, 0, &req);
    if((ret = waitio_wait(&req)) != 0) {
      fprintf(stderr, "%d waitio_isend error %d¥n", wrank, ret);
    }
    else {
      fprintf(stderr, "rank%d: waitio_isend to rank%d¥n", wrank,  wrank+1);
    }
    waitio_irecv(grp1, wrank+1, (char *)&buf, 4, 0, &req);
    if((ret = waitio_wait(&req)) != 0) {
      fprintf(stderr, "%d waitio_irecv error %d¥n", wrank, ret);
    }
    else {
      fprintf(stderr, "rank%d: waitio_irecv from rank%d¥n", wrank,  wrank+1);
    }
  }
```

- waitio_irecv(grp1, wrank-1, (char *)buf, 4, 0, &req);
  - Non Blocking receive function
  - Arguments: Group, source, buffer pointer, number of bytes, tag, Pointer to a request structure
- waitio_wait(&req)
  - Wait Operation of specified Request
  - WaitIO transmission/reception processing is started and executed within this waitio_wait function.
- waitio_isend(grp1, wrank-1, (char *)buf, 4, 0, &req);
  - Non Blocking send function
  - Arguments: Group, destination, buffer pointer, number of bytes, tag, Pointer to a request structure

WaitIO-MPI Conversion Library

- WaitIO does not have MPI Datatype
- WaitIO-MPI Conversion Library was developed to convert MPI program to WaitIO

WaitIO-MPI API (2022/9/1)	description
waitio_mpi_isend	WaitIO MPI_Isend
waitio_mpi_irecv	WaitIO MPI_Irecv
waitio_mpi_reduce	WaitIO MPI_Reduce
waitio_mpi_bcast	WaitIO MPI_Bcast
waitio_mpi_allreduce	WaitIO MPI_Allreduce
waitio_mpi_waitall	WaitIO MPI_Waitall
waitio_create_universe	WaitIO Initialization(all ranks)
waitio_create_universe_pbhead	WaitIO Initialization(rank0 of each PB)
waitio_mpi_gather	WaitIO MPI_Gather
waitio_mpi_algather	WaitIO MPI_Algather
waitio_mpi_scatter	WaitIO MPI_Scatter
waitio_mpi_scatterv	WaitIO MPI_Scatterv
waitio_mpi_gatherv	WaitIO MPI_Gatherv
waitio_mpi_barrier	WaitIO MPI_Barrier
waitio_mpi_type_size	WaitIO MPI_Tyep_size

WaitIO-MPI Conversion API

- Initialization API: int waitio_create_universe (WAITIO_MPI_Comm *commp) ;
  - Same as MPI_Init() in MPI library. All process must call this function.
  - A WaitIO Group is generated in which all processes participate. WAITIO_MPI_Comm is defined as same as waitio_group_t
- Initialization API: int waitio_create_universe_pbhead (WAITIO_MPI_Comm *commp) ;
  - Same as MPI_Init() in MPI library. All process must call this function.
  - A WaitIO Group is created in which the rank 0 process of each PB participates
- Comm.API : int waitio_mpi_isend (const void *buf, int count, WAITIO_MPI_Datatype datatype, int dest, int tag, WAITIO_MPI_Comm comm, WAITIO_MPI_Request *request);
  - #pragma weak WAITIO_MPI_Isend = waitio_mpi_isend, WAITIO_MPI_Request is defined as same as waitio_req_t
- Comm.API : int waitio_mpi_irecv (void *buf, int count, WAITIO_MPI_Datatype datatype, int source, int tag, WAITIO_MPI_Comm comm, WAITIO_MPI_Request *request);
  - #pragma weak WAITIO_MPI_Irecv = waitio_mpi_irecv, WAITIO_MPI_Request is defined as same as waitio_req_t
  - No Status argument
- Comm.API : int waitio_wait(waitio_req_t *req);
  - Wait for completion of the process specified by the “req”.
- In addition, collective communication functions follow the definition of MPI functions.

The pHEAT-3D Application Conversion Example

- Translates pHEAT-3D from Fortran+MPI to WaitIO-MPI Conversion Library
  - WaitIO-MPI Conversion code conversion mechanically convertible

```
      include  'mpif.h'
      include 'waitio_mpif.h'
      integer(kind=kint ), dimension(:,:), save,allocatable :: req1
      integer, save :: NFLAG
      data NFLAG/0/

!C
!C-- INIT.
      if (allocated(sta1)) deallocate (sta1)
      if (allocated(req1)) deallocate (req1)
      allocate (sta1(WAITIO_STATUS_SIZE,2*NEIBPETOT+4))
      allocate (req1(WAITIO_REQUEST_SIZE,2*(NEIBPETOT+4)))
!C
!C-- SEND
      do neib= 1, NEIBPETOT
        istart= STACK_EXPORT(neib-1)
        inum  = STACK_EXPORT(neib  ) - istart
!$omp parallel do private (k,ii)
        do k= istart+1, istart+inum
          ii= NOD_EXPORT(k)
          WS(k)= X(ii)
        enddo
        call WAITIO_MPI_Isend (WS(istart+1), inum,
     &       WAITIO_MPI_DOUBLE_PRECISION,
     &       NEIBPE(neib), 0, WAITIO_SOLVER_COMM, req1(1,neib), ierr)
      enddo
```

```
 !C-- RECV
      do neib= 1, NEIBPETOT
        istart= STACK_IMPORT(neib-1)
        inum  = STACK_IMPORT(neib  ) - istart
        call WAITIO_MPI_Irecv (X(istart+N0+1),inum,
     &                 WAITIO_MPI_DOUBLE_PRECISION,
     &                 NEIBPE(neib), 0, WAITIO_SOLVER_COMM,
     &                 req1(1,neib+NEIBPETOT), ierr)
      enddo
!C
      call WAITIO_MPI_Waitall (2*NEIBPETOT, req1, sta1, ierr)

      end subroutine SOLVER_SEND_RECV
      end module     solver_SR
```

```
!$omp parallel do private(i) reduction(+: RHO0)
      do i= 1, N
        RHO0= RHO0 + WW(i,R)*WW(i,Z)
      enddo

      call WAITIO_MPI_Allreduce (RHO0, RHO, 1,
     &                 WAITIO_MPI_DOUBLE_PRECISION,
     &                 WAITIO_MPI_SUM, WAITIO_SOLVER_COMM, ierr)
```

WaitIO Sample Program:PingPong on Wisteria/BDEC-01 system

- Sample Program extraction to /work directory
  - tar -zxf /work/share/waitio/src/examples/PingPong.tgz
- Compile
  - module install intel impi
  - make all
- Execution Steps
  - Modify batch script
  - Batch processing execution
    - pjsub xxx.sh
    - Single PB: odyssey, aquarius, Multiple PBs: odyssey+aquarius(odyssey+odyssey)

PingPong Program by WaitIO

```
001 /* waitio-test.c */
002 #include <mpi.h>
003 #include "waitio.h"

004 int truef(int pbid, int n) { return 1; }

006 int main (int argc, char *argv[]) {
007   int ret, wrank;
008   waitio_filter_func_t func[4]= {truef, truef, NULL, NULL};
009   int  array[4] = {1, 2, 0, 0};
010   int  data[2], *buf = data;
011   waitio_req_t req;
012   waitio_group_t grp1;

013   if((ret = MPI_Init( &argc, &argv)) != MPI_SUCCESS) {
014     fprintf(stderr, "MPI_Init failed code %d¥n", ret);
015     exit(ret);
016   }
017   if((ret = waitio_init(-1)) != 0) {
018     fprintf(stderr, "waitio_init failed code %d¥n", ret);
019      MPI_Abort(MPI_COMM_WORLD, ret);
020     exit(ret);
021   }

022   grp1 = waitio_create_group(0, func, array);
023   if(grp1 == NULL) {
024     fprintf(stderr, "waitio_create_group failed code %d¥n", ret);
025     MPI_Finalize();
026   exit(ret);
027    }
028   waitio_group_rank(grp1, &wrank);
```

```
029   if((wrank%2) == 1) {
030     waitio_irecv(grp1, wrank-1, (char *)buf, 4, 0, &req);
031     if((ret = waitio_wait(&req)) != 0) {
032       fprintf(stderr, "%d waitio_irecv error %d¥n", wrank, ret);
033     }
034     else {
035       fprintf(stderr, "rank%d: waitio_irecv from rank%d¥n", wrank,  wrank-1);
036     }
037     waitio_isend(grp1, wrank-1, (char *)buf, 4, 0, &req);
038     if((ret = waitio_wait(&req)) != 0) {
039       fprintf(stderr, "%d waitio_isend error %d¥n", wrank, ret);
040     }
041     else {
042       fprintf(stderr, "rank%d: waitio_isend to rank%d¥n", wrank,  wrank-1);
043     }
044   }
045   else {
046     waitio_isend(grp1, wrank+1, (char *)buf, 4, 0, &req);
047     if((ret = waitio_wait(&req)) != 0) {
048       fprintf(stderr, "%d waitio_isend error %d¥n", wrank, ret);
048     }
050     else {
051       fprintf(stderr, "rank%d: waitio_isend to rank%d¥n", wrank,  wrank+1);
052     }
053     waitio_irecv(grp1, wrank+1, (char *)&buf, 4, 0, &req);
054     if((ret = waitio_wait(&req)) != 0) {
055       fprintf(stderr, "%d waitio_irecv error %d¥n", wrank, ret);
056     }
057     else {
058       fprintf(stderr, "rank%d: waitio_irecv from rank%d¥n", wrank,  wrank+1);
059     }
060   }
061   waitio_finalize();
062   MPI_Finalize();
063 }
```

PingPong Program by WaitIO-MPI Conversion API


```
000 /* test-mpi.c*/
001 #include <mpi.h>
002 #include "waitio.h"
003 #include "waitio_mpi.h"

004 int main (int argc, char *argv[]) {
005   int  data[2], *buf = data;
006   waitio_group_t grp1;
007   int ret;
008   int wrank;

009   if((ret = MPI_Init( &argc, &argv)) != MPI_SUCCESS) {
010     fprintf(stderr, "MPI_Init failed code %d¥n", ret);
011     exit(ret);
012   }
013
014   waitio_create_universe (&grp1);
015   if(grp1 == NULL) {
016     fprintf(stderr, "waitio_create_universe failed code %d¥n", ret);
017     MPI_Finalize();
018     exit(ret);
019   }
```

waitio_mpi_recv はWAITIO_MPI_Recv
waitio_mpi_send はWAITIO_MPI_Send
としても定義されているため、MPIプログラムの機械的な置き換えも可能である。

```
020   waitio_group_rank(grp1, &wrank);
021   if((wrank%2) == 1) {
022     if((ret = waitio_mpi_recv(buf, 4, WAITIO_MPI_CHAR, wrank-1, 0, grp1)) != 0) {
023       fprintf(stderr, "%d waitio_mpi_recv error %d¥n", wrank, ret);
024     }
025     else {
026       fprintf(stderr, "rank%d: waitio_mpi_recv from rank%d¥n", wrank,  wrank-1);
027     }
028     if((ret = waitio_mpi_send(buf, 4, WAITIO_MPI_CHAR, wrank-1, 0, grp1)) != 0) {
029       fprintf(stderr, "%d waitio_send error %d¥n", wrank, ret);
030     }
031     else {
032       fprintf(stderr, "rank%d: waitio_mpi_send to rank%d¥n", wrank,  wrank-1);
033     }
034   }
035   else {
036     if((ret = waitio_mpi_send(buf, 4, WAITIO_MPI_CHAR, wrank+1, 0, grp1)) != 0) {
037       fprintf(stderr, "%d waitio_send error %d¥n", wrank, ret);
038     }
039     else {
040       fprintf(stderr, "rank%d: waitio_mpi_send to rank%d¥n", wrank,  wrank+1);
041     }
042     if((ret = waitio_mpi_recv(buf, 4, WAITIO_MPI_CHAR, wrank+1, 0, grp1)) != 0) {
043       fprintf(stderr, "%d waitio_recv error %d¥n", wrank, ret);
044     }
045     else {
046       fprintf(stderr, "rank%d: waitio_mpi_recv from rank%d¥n", wrank,  wrank+1);
047     }
048   }

049   waitio_finalize();
050   MPI_Finalize();
051 }
```

PingPong Program Compile

- Copy and expand the sample program to the work directory

```
[z30xxx@wisteria01 sample]$ realpath .
/work/jh21yyyya/z30xxx/sample
[z30xxx@wisteria01 sample]$ tar -zxf /work/share/waitio/src/examples/PingPong.tgz
[z30xxx@wisteria01 sample]$
```

- Complile

```
[z30xxx@wisteria01 sample]$ cd PingPong/
[z30xxx@wisteria01 PingPong/]$ls
Makefile     test-a64fx-1.sh  test.c           test-mpi.c
mpi-Itest.c  test-a64fx-2.sh  test-intel-2.sh     test-mpi-intel-2.sh
mpi-test.c   test-a64fx.sh    test-mpi-a64fx-1.sh
[z30xxx@wisteria01 PingPong/]$ module purge
[z30xxx@wisteria01 PingPong/]$ module load intel impi
[z30xxx@wisteria01 PingPong/]$ make all –k
[z30xxx@wisteria01 PingPong/]$ module purge
[z30xxx@wisteria01 PingPong/]$ module load fj fjmpi
[z30xxx@wisteria01 PingPong/]$ make all -k
..
[z30xxx@wisteria01 PingPong/]$ls
impi-a64fx  mpi-intel    test-a64fx-1.sh  test-intel         test-mpi.c
impi-intel  mpi-Itest.c  test-a64fx-2.sh  test-intel-2.sh     test-mpi-intel
Makefile    mpi-test.c   test-a64fx.sh    test-mpi-a64fx     test-mpi-intel-2.sh
mpi-a64fx   test-a64fx   test.c           test-mpi-a64fx-1.sh
[z30xxx@wisteria01 PingPong/]$
```

Preparation of PingPong Program Execution: PB number=1

- Modify batch script: test-a64fx.sh

```
#!/bin/sh
#------ pjsub option --------#
#PJM -L rscgrp=lecture-o
#PJM -L node=1
#PJM --mpi proc=2
#PJM -L elapse=00:02:00
#PJM -g gr07
#PJM -j
#------- Program execution -------#

module purge
module load waitio
module load fj
module load fjmpi
export WAITIO_MASTER_HOST=`hostname`
export WAITIO_MASTER_PORT=7100
export WAITIO_PBID=0
export WAITIO_NPB=1

mpiexec  ./test-a64fx
exit
```

PingPong Program Execution: PB number=1

- Submission of Batch Script

```
[z30xxx@wisteria01 sample]$ pjsub test-a64fx.sh
[INFO] PJM 0000 pjsub Job 636562 submitted.
[z30xxx@wisteria01 sample]$
```

- Job Output : After the Job finished by checking pjstat command

```
[z30xxx@wisteria01 PingPong/]$ cat test-a64fx.sh.636562.out
Unloading odyssey
  WARNING: Did not unuse /work/opt/local/modules/modulefiles/WO/odyssey/core
  WARNING: Did not unuse /work/opt/local/modules/modulefiles/WO/odyssey/util
Loading fj/1.2.35
  Loading requirement: fjmpi/1.2.35
wo0017:0:0:sock_create master_port 7100
wo0017:0:0: WAITIO_MASTER socket bind port 7100
rank0: waitio_isend to rank1
rank1: waitio_irecv from rank0
rank1: waitio_isend to rank0
rank0: waitio_irecv from rank1
[z30xxx@wisteria01 PingPong/]$
```

Preparation of PingPong Program Execution: PB number=2(a64fx+a64fx)

- Modify batch scripts：test-a64fx-1.sh, test-a64fx-2.sh

```
#!/bin/sh
#------ pjsub option --------#
#PJM -N "test1"
#PJM -L rscgrp=coupler-lec-o
#PJM -L node=1
#PJM --mpi proc=2
#PJM -L elapse=00:02:00
#PJM -g gr00
#PJM -j
#------- Program execution -------#

hostname
module purge
module load fj
module load fjmpi
module load waitio

export WAITIO_MASTER_HOST=`hostname`
export WAITIO_MASTER_PORT=7100
export WAITIO_PBID=0
export WAITIO_NPB=2
waitio-serv-a64fx -d -m $WAITIO_MASTER_HOST
echo  "serv host set ${WAITIO_MASTER_HOST} "

mpiexec  ./test-a64fx
exit
```

```
#!/bin/sh
#------ pjsub option --------#
#PJM -N "test1"
#PJM -L rscgrp=coupler-lec-o
#PJM -L node=1
#PJM --mpi proc=2
#PJM -L elapse=00:02:00
#PJM -g gr00
#PJM -j
#------- Program execution -------#

hostname
module purge
module load fj
module load fjmpi
module load waitio

export WAITIO_MASTER_PORT=7100
export WAITIO_PBID=1
export WAITIO_NPB=2
export WAITIO_MASTER_HOST=`waitio-serv-a64fx -c`

echo  "serv host is ${WAITIO_MASTER_HOST} "
mpiexec  ./test-a64fx
exit
```

PingPong Program Execution: PB number=2 (a64fx+a64fx)

- Submission of Batch Scripts

```
[z30xxx@wisteria01 sample]$ pjsub test-a64fx-1.sh
[INFO] PJM 0000 pjsub Job 636574 submitted.
[z30xxx@wisteria01 PingPong]$ pjsub test-a64fx-2.sh
[INFO] PJM 0000 pjsub Job 636575 submitted.
[z30xxx@wisteria01 sample]$
```

- Job Output : After the Job finished by checking pjstat command

```
[z30xxx@wisteria01 PingPong/]$ cat test1.636574.out
wo5575
Unloading odyssey
  WARNING: Did not unuse /work/opt/local/modules/modulefiles/WO/odyssey/core
  WARNING: Did not unuse /work/opt/local/modules/modulefiles/WO/odyssey/util
Loading fj/1.2.35
  Loading requirement: fjmpi/1.2.35
wo5575:-1::waitio_setargs user=z30455, jobn=test1!
waitio-server host wo5575:6500 started
wo5575:0/2(103):waitio_connect_serv_sock:trying connect host 10.1.0.1,port 25625
wo5575:0/2(103):waitio_connect_serv_sock: connected to host 10.1.0.1,port 25625
wo5575
wo5575
serv host set wo5575
wo5575:0:0:sock_create master_port 7100
wo5575:0:0: WAITIO_MASTER socket bind port 7100
 wo5575:0/2(112):PBserver: waitio_fetch_PBdata set timeout 10
wo5575:0:0 waitio_init Procs:
   PB No.0 rank=0 IP-addr:port#=10.11.135.7:8000 nprocs=2, nGIO=0
   PB No.1 rank=0 IP-addr:port#=10.11.135.15:8000 nprocs=2, nGIO=0
wo5575:0/2(112): Multiple WaitIO Connecter now ready NPB=2!
 wo5575:0/2(112):PBserver: waitio_fetch_PBdata accepted from 10.11.135.15 port=40648 fd=32
rank0: waitio_isend to rank1
rank1: waitio_irecv from rank0
rank1: waitio_isend to rank0
rank0: waitio_irecv from rank1
```

```
[z30xxx@wisteria01 PingPong/]$ cat test1.636575.out
wo5583
Unloading odyssey
  WARNING: Did not unuse /work/opt/local/modules/modulefiles/WO/odyssey/core
  WARNING: Did not unuse /work/opt/local/modules/modulefiles/WO/odyssey/util
Loading fj/1.2.35
  Loading requirement: fjmpi/1.2.35
waitio-server host wo5583:6500 started
serv host is wo5575
wo5583:1:0 waitio_init Procs:
   PB No.0 rank=0 IP-addr:port#=10.11.135.7:8000 nprocs=2, nGIO=0
   PB No.1 rank=0 IP-addr:port#=10.11.135.15:8000 nprocs=2, nGIO=0
wo5583:1/2(110): Multiple WaitIO Connecter now ready NPB=2!
wo5583:1:0/2(110):PBclient:trying connect host 10.11.135.7,port 48155
wo5583:1/2(110):PBclient: connected to host 10.11.135.7,port 48155
wo5583:1/2(110):PBclient: upstream to WaitIO master Nproc=2 done!
rank2: waitio_isend to rank3
rank3: waitio_irecv from rank2
rank3: waitio_isend to rank2
rank2: waitio_irecv from rank3
```

Questions?