

第236回 お試しアカウント付き 並列プログラミング講習会

「異種システム間連成アプリケーション開発を学ぶ:WaitIO/MP
- シミュレーションとアプリケーション間連携編-」

東京大学情報基盤センター
住元 真司、荒川 隆

講習会概略

- **開催日：**

2024年10月30日（水） 13：00 - 17：30

- **形態：**Zoomを用いたオンライン講習会

- **講習会プログラム：** 講師：住元、荒川

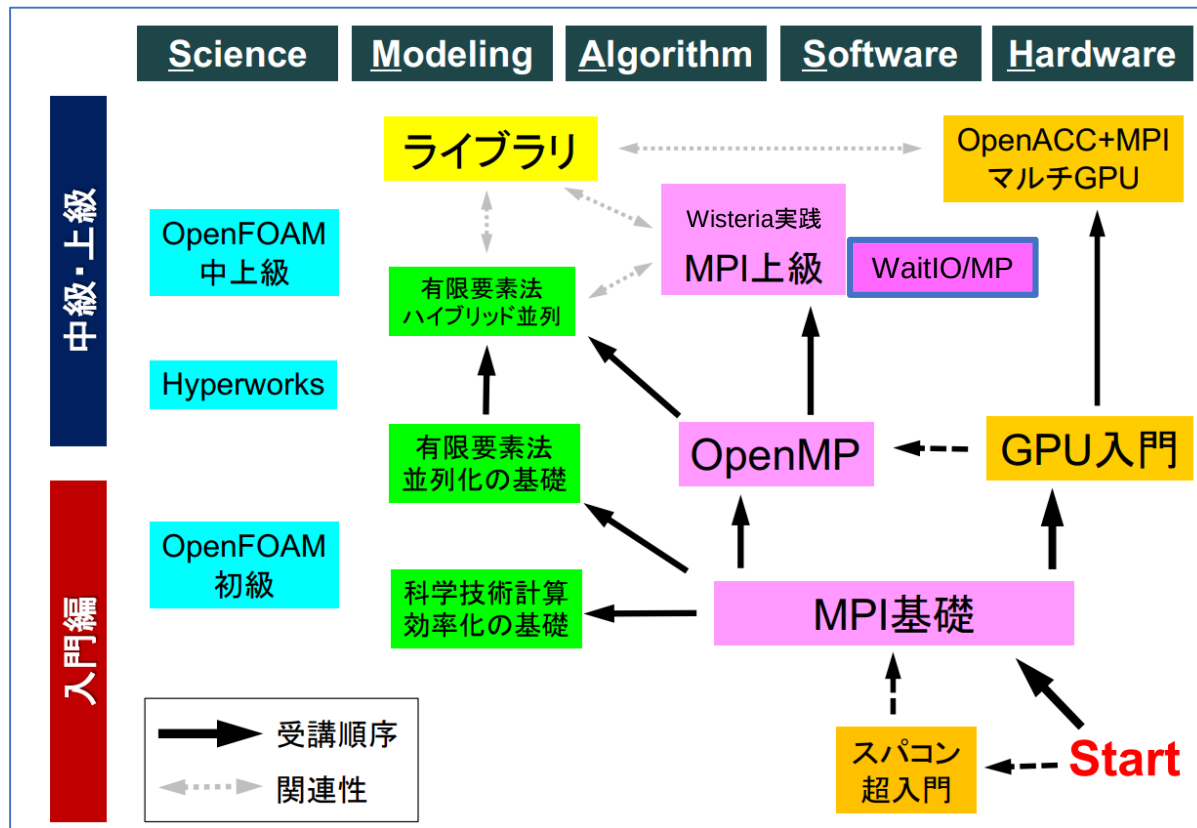
- 13:00 – 13:10 受講イントロダクション：10分
 - 全体の位置づけ、Zoom, 注意事項
- 13:10 – 13:30 Wisteria/BDEC-01の概要：20分
- 13:30 – 14:10 システム利用制度案内, h3-Open-BDECソフトウェア紹介
 - <10 分休憩>
- 14:20 – 15:40 h3-Open-SYS/WaitIO, h3-Open-UTIL/MPの概要とプログラミング
 - <10 分休憩>
- 15:50 – 17:30 h3-Open-SYS/WaitIOとh3-Open-UTIL/MPを使ってみよう

h3-Open-BDECソフトウェア: WaitIO/MP 講習会シリーズ

過去トピックもご興味あればアクセス下さい：講習会資料・ビデオ

- 第217回 2023年10月19日 « **オンライン講習会** »
「異種システム間連成アプリケーション開発を学ぶ:WaitIO/MP講習会 - シミュレーションと機械学習融合編 -」
<https://www.cc.u-tokyo.ac.jp/events/lectures/217/>
- 第191回 2022年10月14日 « **オンライン講習会** »
「異種システム間連成アプリケーション開発を学ぶ:WaitIO/MP講習会」
<https://www.cc.u-tokyo.ac.jp/events/lectures/191/>

当基盤センター主催・講習会概要: ご活用ください



受講イントロダクション

お願い・Zoom利用について

お願い等

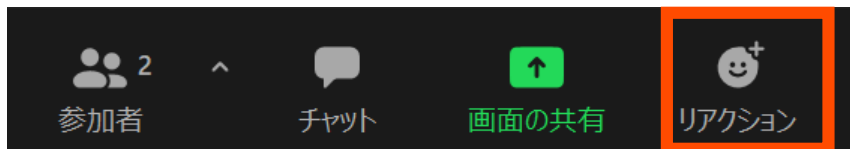
- ハンズオンのためのPC, Zoom及びスパコンへ接続するためのネットワーク環境は各受講者でご準備ください。
- PCは Windows/Microsoft Update, Apple Security Updateなどで最新のセキュリティアップデートを行ってください。
- 必ずウィルス対策ソフトウェアをインストールし, ウィルス検索を実行して問題がないことを事前に確認してから受講してください。
 - セキュリティ対策未実施の場合はオンライン講習会受講を認めません。
- OSは、Windows、Macどちらでも構いませんが、SSHを用いてセンターのスーパーコンピューターへ接続ができることが必要です（後述）。
- 演習の実施に当たり, 受講生にセンターのスーパーコンピューターを1月間利用できる無料アカウント（お試しアカウント）を発行します。

Zoom関連

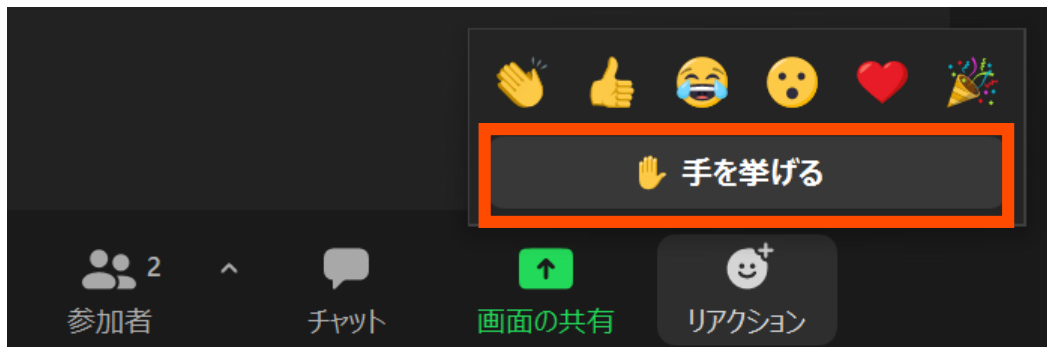
- 「手をあげる」機能
 - 質問がある際、全体の状況を確認するため使用
- ブレークアウトセッション
 - 画面を共有しながらエラー対応する際に使用
 - （なるべく口頭でのやりとりで対応する予定）
- https://utelecon.adm.u-tokyo.ac.jp/zoom/how_to_use

「手を挙げる」方法

1. Zoomメニュー中の「リアクション」をクリック

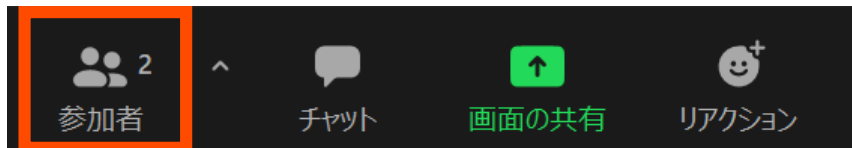


2. ポップアップで表示された「手を挙げる」をクリック

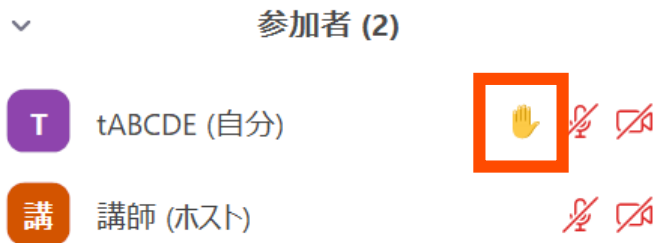


手が挙がっていることの確認方法

1. Zoomメニュー中の「参加者」をクリックして、参加者一覧を表示

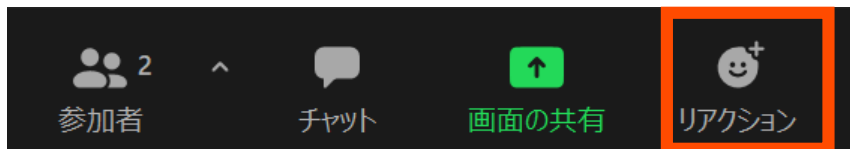


2. 表示された参加者一覧の、自分のところを見ると手が挙がっている

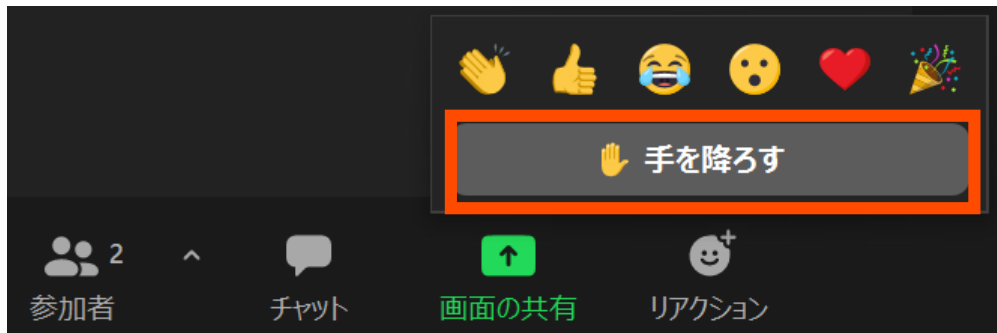


「手を降ろす」方法

1. Zoomメニュー中の「リアクション」をクリック

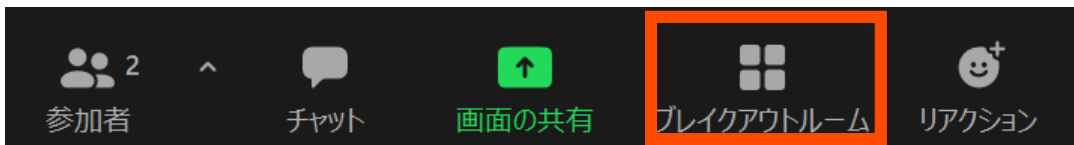


2. ポップアップで表示された「手を降ろす」をクリック



ブレイクアウトルーム (1/6)

- 演習時に使用するかもしれません
- 演習中に「ヘルプを求める」ことができます
 - ホストを招待した後に「画面を共有」することで、皆さんの記述したプログラムを一緒に見ながら問題解決にあたります
- Zoomメニュー中の「ブレイクアウトルーム」をクリック



ブレイクアウトルーム (2/6)

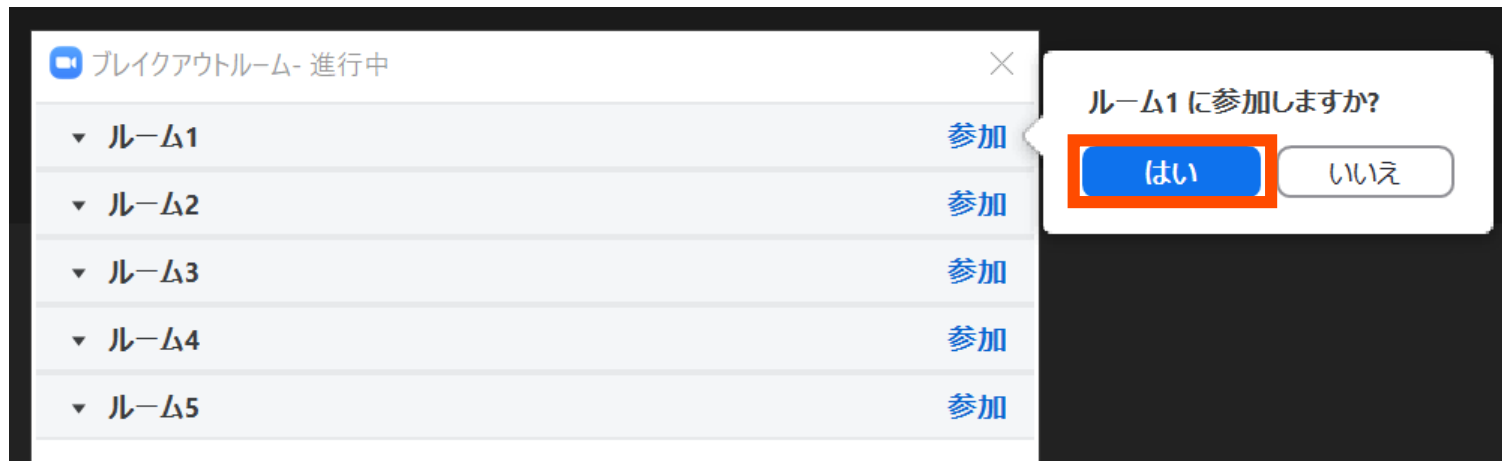
- 進行中のブレイクアウトルームのリストが表示されるので、空いている部屋に「参加」してください
 - 左の例では5部屋がすべて空室、右の例ではルーム1のみ参加者がいる

ブレイクアウトルーム- 進行中		×
▼ ルーム1	参加	
▼ ルーム2	参加	
▼ ルーム3	参加	
▼ ルーム4	参加	
▼ ルーム5	参加	

ブレイクアウトルーム- 進行中		×
▼ ルーム1	参加	
● tABCDE		
▼ ルーム2	参加	
▼ ルーム3	参加	
▼ ルーム4	参加	
▼ ルーム5	参加	

ブレイクアウトルーム (3/6)

- 「参加」をクリックすると確認画面が出てくるので、「はい」を選択すると入室できます



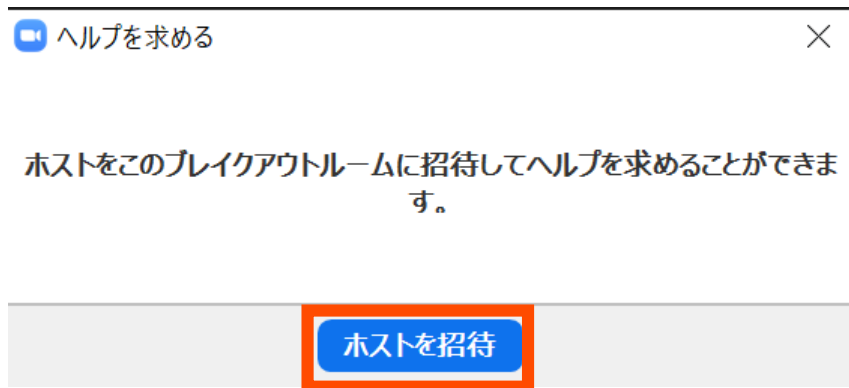
ブレイクアウトルーム (4/6)

- 再度メニュー中の「ブレイクアウトルーム」をクリックすると、「ヘルプを求める」が増えているのでクリックしてください



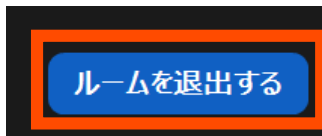
ブレイクアウトルーム（5/6）

- ポップアップで出てきた「ホストを招待」をクリック
- ホスト（講師）の参加待ちに移行します（画面はそのまま）
 - 他の受講者のヘルプ中など、直ちに対応できない場合もあります

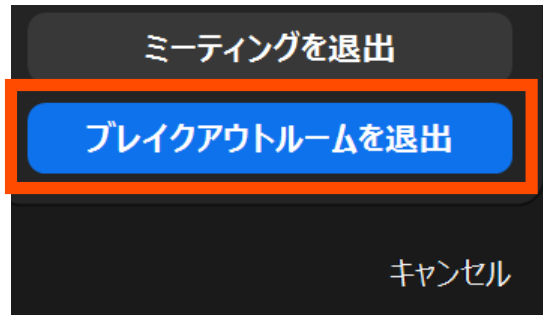


ブレイクアウトルーム（6/6）

- 問題解決後は、Zoomメニュー中の「ルームを退出する」



- 「ブレイクアウトルームを退出」が表示されるのでクリックして、元の講習会会場にお戻りください
 - 間違えて「ミーティングを退出」すると講習会から退出します



情報基盤センターシステム・ システム利用制度の紹介

- 13:10 – 13:20 情報基盤センターシステムWisteria/BDEC-01の概要
- 13:50 – 14:00 システム利用制度案内：10分
- 14:00 – 14:10 h3-Open-BDECソフトウェア概要

別資料参照

休憩 10分

Next

h3-Open-BDECソフトウェア: WaitIO/MP

14:20 –

講習会の概要: 4.5時間コース

- 13:00 – 13:10 受講イントロダクション
 - 全体の位置づけ、Zoom, 注意事項
- 13:10 – 13:50 Wisteria/BDEC-01の概要
- 13:50 – 14:00 システム利用制度案内：10分
- 14:00 – 14:10 h3-Open-BDECソフトウェア概要
- 14:20 – 15:40 h3-Open-BDECソフトウェア: WaitIO/MP
 - h3-Open-UTIL/MPの概要とプログラミング: 50分
 - h3-Open-SYS/WaitIOの概要とプログラミング: 30分
- 15:50 – 17:30 h3-Open-SYS/WaitIOとh3-Open-UTIL/MPを使ってみよう
 - Wisteria/BDEC-01システム利用イントロダクション：20分
 - サンプルプログラムの実行WaitIO: 30分
 - サンプルプログラムの実行MP: 50分
 - 自由に動かしてみよう・Q&A: 10分

h3-Open-BDECソフトウェア: WaitIO/MP

h3-Open-UTIL/MPの概要とプログラミング

講習の内容

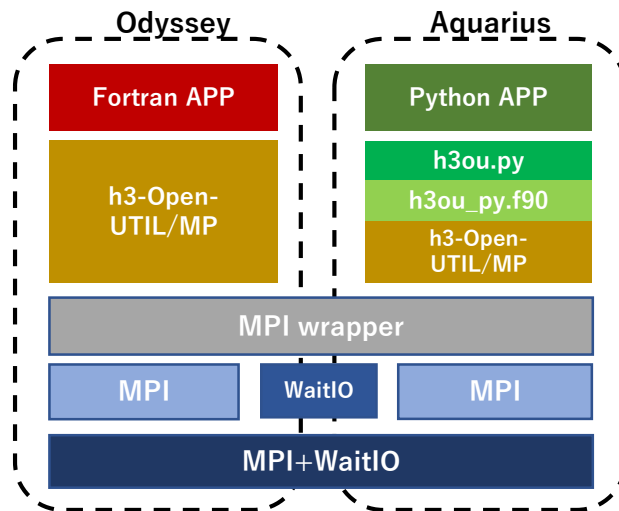
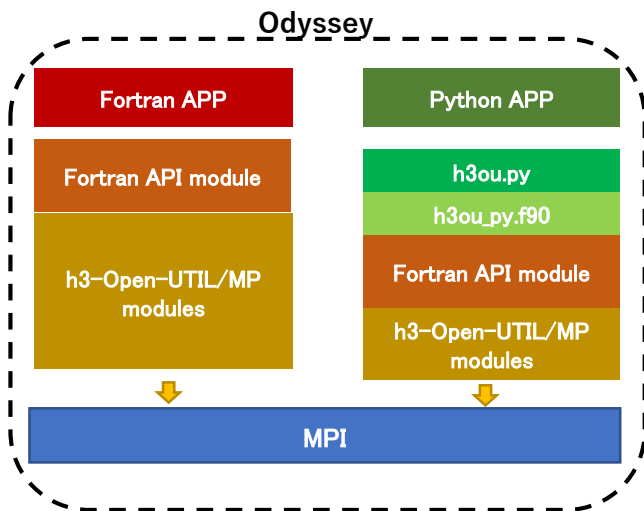
- h3-Open-UTIL/MPの概要
- UTIL/MPの利用方法
 - 簡易通信
 - 連成計算
- 具体的な利用手順
 - 設定ファイルの準備
 - APIの組み込み
- 異機種間連成
 - コンパイルと実行
 - Python使用時の注意

h3-Open-UTIL/MPの概要

- 連成計算のための汎用ライブラリ
 - 弱連成計算を対象
 - 異なる時空間スケールを持つ複数のシミュレーションモデル間で情報（データ）を交換しながら計算を実行
- 特徴
 - 格子形状に依存しない→任意の格子のモデルに適用可能
 - 空間補間方法を利用者が定義できる
 - アンサンブル連成可能
 - h3-Open-SYS/WaitIOと連携し異機種間で連成計算できる
 - Pythonインターフェースを持つ→機械学習ライブラリとの「連携」
 - **単純にデータ交換だけを行う簡易通信機能を装備**

UTIL/MPの構造

- モデル間通信はカプラが実行
 - 利用者はモデル間通信について気にする必要がない
 - モデル内のMPI通信はカプラから取得したコミュニケータを使う
 - 同機種内連成と異機種間連成
 - リンクするライブラリが異なる
 - ジョブの実行方法が異なる
- それ以外は同じ→同一モデルのコードを同機種内・異機種間で連成できる



UTIL/MPの利用方法

- 連成計算

- 異なる時空間スケールを持つ複数のシミュレーションモデル間で変数（格子データ）を交換しながら計算を実行
- 異なる格子系間での補間計算
- データ交換時間間隔のコントロール
- 時間平均計算機能

- 簡易通信

- 配列データの通信のみに対応
- bcastとsend, recv
- 整数、単精度実数、倍精度実数
- 大域通信、モデル間通信、（ローカル通信）

講習の内容

- h3-Open-UTIL/MPの概要
- UTIL/MPの利用方法
 - 簡易通信
 - 連成計算
- 具体的な利用手順
 - 設定ファイルの準備
 - APIの組み込み
- 異機種間連成
 - コンパイルと実行
 - Python使用時の注意

設定ファイルの準備

```
! log_level : "SILENT" or "WHISPER" or "LOUD"  
! debug_mode : .true. or .false.  
! stop_step: 0, 1 to 10  
&h3ou_coupling  
  log_level = "LOUD"  
  debug_mode = .false.  
  stop_step = 0  
&end
```

log_level : “LOUD”, “WHISPER”, “SILENT”のいずれか

debug_mode : .true.から.false.のいずれか

.true.にするとJcupのログが出力される

stop_step : 0もしくは1から10までの整数

カプラーチンの所定の場所でプログラムを停止する
0ならば停止しない

APIの組み込み：初期化プロセス

```
h3ou_init(comp_name, config_file_name)
```

カプラの初期化

comp_name: モデルコンポーネント名

config_file_name: 設定ファイルのファイル名

```
h3ou_get_mpi_parameter(comp_name, my_comm, my_group, my_size, my_rank)
```

カプラからローカルなMPI情報を取得

comp_name: モデルコンポーネント名

my_comm, my_group, my_size, my_rank: ローカルなMPI情報

APIの組み込み：グローバル通信

```
h3ou_bcast_global(comp_name, data)
```

全モデルのbcast

comp_name: 送信元モデルコンポーネント名

data: 整数、単精度実数、倍精度実数の1次元配列

APIの組み込み：モデル間通信

```
h3ou_bcast_model(source_name, target_name, data)
```

モデル間のbcast

source_name: 送信元モデルコンポーネント名

target_name: 送信先モデルコンポーネント名

data: 整数、単精度実数、倍精度実数の1次元配列

APIの組み込み：モデル間通信

```
h3ou_send_model(target_name, target_rank, data)
```

モデル間の送信

target_name: 送信元モデルコンポーネント名

target_rank: 送信先ランク番号

data: 整数、単精度実数、倍精度実数の1次元配列

```
h3ou_send_model(target_name, target_rank, data)
```

モデル間の送信

target_name: 送信元モデルコンポーネント名

target_rank: 送信先ランク番号

data: 整数、単精度実数、倍精度実数の1次元配列

APIの組み込み：終了処理

h3ou_end()

カプラの終了

コードの説明 1/3

```
program compA
```

```
  use h3ou_api
```

```
  use mpi
```

```
  implicit none
```

```
  character(len=5) :: my_name = "compA"
```

```
  integer          :: my_comm, my_group, my_size,
```

```
  my_rank
```

```
  integer,parameter :: array_size = 10
```

```
  integer          :: my_array_int(array_size)
```

```
  real(kind=4)      :: my_array_real(array_size)
```

```
  real(kind=8)      :: my_array_double(array_size)
```

```
  integer          :: target_array_int(array_size)
```

```
  real(kind=4)      :: target_array_real(array_size)
```

```
  real(kind=8)      :: target_array_double(array_size)
```

```
  integer          :: status(MPI_STATUS_SIZE)
```

```
  integer          :: ierror
```

```
  integer          :: i
```

```
! initialize
```

```
call h3ou_init(my_name, "coupling.conf")
```

```
! get mpi parameters
```

```
call h3ou_get_mpi_parameter(my_name, my_comm,  
my_group, my_size, my_rank)
```

```
write(0, *) my_name//" my_size = ", my_size, "  
my_rank = ", my_rank
```


コードの説明 2/3

```
! set my array
do i = 1, array_size
  my_array_int(i) = 10000 + my_rank * 100 + i
  my_array_real(i) = my_array_int(i)
  my_array_double(i) = my_array_int(i)
end do
```

! bcast global

```
if (my_rank == 0) then
  call h3ou_bcast_global("compA", my_array_int)
else
  call h3ou_bcast_global("compA", target_array_int)
end if
call h3ou_bcast_global("compB", target_array_real)
call h3ou_bcast_global("compC",
target_array_double)
```

! bcast model

```
if (my_rank == 0) then
  call h3ou_bcast_model("compA", "compB",
my_array_int)
else
  target_array_int = 0
  call h3ou_bcast_model("compA", "compB",
target_array_int)
  write(0, *) "compA bcast model, ", target_array_int
end if
```

コードの説明 3/3

! send/recv model

```
if (my_rank == 0) then
  call h3ou_send_model("compB", 0,
my_array_double)
  call h3ou_recv_model("compB", 0,
target_array_real)
  write(0, *) "compA send/recv model, ",
target_array_real
end if

! bcast local

if (my_rank == 0) then
  call mpi_bcast(my_array_int, array_size,
MPI_INTEGER, 0, my_comm, ierror)
else
  call mpi_bcast(target_array_int, array_size,
MPI_INTEGER, 0, my_comm, ierror)
```

write(0, *) "compA bcast local, ", target_array_int

! send/recv local

```
if (my_size > 1) then
  if (my_rank == 0) then
    call mpi_send(my_array_real, array_size,
MPI_REAL, 1, 0, my_comm, ierror)
  end if
  if (my_rank == 1) then
    call mpi_recv(target_array_real, array_size,
MPI_REAL, 0, 0, my_comm, status, ierror)
    write(0, *) "compA send/recv local, ",
target_array_real
  end if
end if

call h3ou_coupling_end()
end program compA
```

コンパイル方法

- 具体的な手順は実践編で
 - OdysseyとAquariusで必要なモジュールが異なる

```
$ export SYSTEM=odyssey
```

```
$ module load odyssey
```

```
$ module load waitio
```

```
$ mpifrtpx -L/work/share/h3-Open-UTIL-MP/h3ou2024/lib/odyssey -lh3ou -  
ljcup4 -L*** -lwatio-a64fx
```

```
$ export SYSTEM=aquarius
```

```
$ module load intel impi
```

```
$ module load waitio
```

```
$ module unload intel impi
```

```
$ module load gcc omp
```

```
$ mpifort -L/work/share/h3-Open-UTIL-MP/h3ou2024/lib/aquarius -lh3ou -  
ljcup4
```

2024/10/30

WaitIO/MP

講習会

実行結果

- 設定ファイルでlog_level = “LOUD”の場合
 - h3ou.モデル名.pe?????というログファイルが出力される

```
=====
=====
===== h3-Open-UTIL/MP coupling log
=====

=====
=====
>>>>>>>>>>>>>>>> h3ou_init IN   } ここまで出ていたらOK
<<<<<<<<<<<<<<<<< h3ou_init OUT
```

中略

```
>>>>>>>>>>>>>>>> h3ou_coupling_end IN
<<<<<<<<<<<<<<<<< h3ou_coupling_end OUT
```

Pythonアプリケーションの実行

- ジョブスクリプトの書き方
 - スクリプトの中で各自のPython環境をactivateする
 - mpiexecの引数にpythonを与える

```
module unload intel  
module unload impi  
module load gcc omp  
source /ユーザ毎のPythonディレクトリ/bin/activate
```

```
export UCX_IB_FORK_INIT=yes  
export IBV_FORK_SAFE=1
```

```
mpiexec -n 1 python3 compD.py
```

APIリファレンス

API リファレンス(簡易版): Fortran

h3ou_init	カプラの初期化		
	character(len=*), intent(IN)	comp_name	モデルコンポーネント名
	character(len=*), intent(IN)	config_file_name	設定ファイル名
h3ou_end	カプラの終了(簡易版)		

API リファレンス(簡易版): Fortran

h3ou_bcast_global	全体Bcast		
	character(len=*), intent(IN)	comp_name	コンポーネント名
	integer, real(4), real(8), intent(INOUT)	data(:)	送受信データ配列
h3ou_bcast_model	モデル間Bcast		
	character(len=*), intent(IN)	source_name	送信コンポーネント名
	character(len=*), intent(IN)	target_name	受信コンポーネント名
	integer, real(4), real(8), intent(INOUT)	data(:)	送受信データ配列
h3ou_send_model	モデル間送信		
	character(len=*), intent(IN)	target_name	送信先コンポーネント名
	integer, intent(IN)	target_rank	送信先ランク番号
	integer, real(4), real(8), intent(IN)	data(:)	送信データ配列
h3ou_rcv_model	モデル間受信		
	character(len=*), intent(IN)	source_name	送信元コンポーネント名
	integer, intent(IN)	source_rank	送信元ランク番号
	integer, real(4), real(8), intent(INOUT)	data(:)	受信データ配列

API リファレンス(簡易版): C

h3ouc_init	カプラの初期化		
	char *	comp_name	モデルコンポーネント名
	char *	config_file_name	設定ファイル名
h3ouc_end	カプラの終了(簡易版)		

API リファレンス(簡易版): C

h3ouc_bcast_global_int h3ouc_bcast_global_float h3ouc_bcast_global_double	全体Bcast		
	char *	comp_name	コンポーネント名
	int *, float *, double *	data	送受信データ配列
	int	array_size	配列サイズ

API リファレンス(簡易版): C

h3ouc_bcast_model_int h3ouc_bast_model_float h3ouc_bcast_model_double	モデル間Bcast		
	char *	source_name	送信コンポーネント名
	char *	target_name	受信コンポーネント名
	int *, float *, double *	data	送受信データ配列
	int	array_size	配列サイズ
h3ouc_send_model_int h3ouc_send_model_float h3ouc_send_model_double	モデル間送信		
	char *	target_name	送信先コンポーネント名
	int	target_rank	送信先ランク番号
	int *, float *, double *	data	送信データ配列
	int	array_size	配列サイズ
h3ouc_recv_model_int h3ouc_recv_model_float h3ouc_reacv_model_double	モデル間受信		
	char *	source_name	送信元コンポーネント名
	int	source_rank	送信元ランク番号
	int *, float *, double *	data	受信データ配列
	int	array_size	配列サイズ

2024/10/30

WaitIO/MP

講習会

API リファレンス(簡易版): Python

h3ou_init	カプラの初期化		
	string	comp_name	モデルコンポーネント名
	string	config_file_name	設定ファイル名
h3ou_end	カプラの終了(簡易版)		

API リファレンス(簡易版): Python

np_array h3ou_bcast_global_int np_array h3ou_bcast_global_float np_array h3ou_bcast_global_double	全体Bcast(返値は引数と同じ型のnp.array)		
	char *	comp_name	コンポーネント名
	np.array((array_size), np.int32 or np.float32 or np.float64)	data	送受信データ配列

API リファレンス(簡易版): Python

np_array h3ou_bcast_model_int np_array h3ou_bast_model_float np_array h3ou_bcast_model_double	モデル間Bcast(返値は引数と同じ型のnp.array)		
	string	source_name	送信コンポーネント名
	string	target_name	受信コンポーネント名
	np.array((array_size), np.int or np.float32 or np.float64)	data	送受信データ配列
h3ouc_send_model_int h3ouc_send_model_float h3ouc_send_model_double	モデル間送信		
	string	target_name	送信先コンポーネント名
	int	target_rank	送信先ランク番号
	np.array((array_size), np.int32 or np.float32 or np.float64)	data	送信データ配列
np_array h3ouc_recv_model_int np_array h3ouc_recv_model_float np_array h3ouc_reacv_model_double	モデル間受信(返値は引数と同じ型のnp.array)		
	string	source_name	送信元コンポーネント名
	int	source_rank	送信元ランク番号
	np.array((array_size), np.int 32 or np.float32 or np.float64)	data	受信データ配列

API リファレンス(簡易版): Python

np_array h3ou_bcast_local_int np_array h3ou_bast_local_float np_array h3ou_bcast_local_double	モデル内Bcast(返値は引数と同じ型のnp.array)		
	int	source_pe	送信ランク番号
	np.array((array_size), np.int or np.float32 or np.float64)	data	送受信データ配列
h3ouc_send_local_int h3ouc_send_local_float h3ouc_send_local_double	モデル内送信		
	int	target_rank	送信先ランク番号
	np.array((array_size), np.int32 or np.float32 or np.float64)	data	送信データ配列
np_array h3ouc_recv_local_int np_array h3ouc_recv_local_float np_array h3ouc_reacv_local_double	モデル内受信(返値は引数と同じ型のnp.array)		
	int	source_rank	送信元ランク番号
	np.array((array_size), np.int 32 or np.float32 or np.float64)	data	受信データ配列

h3-Open-BDECソフトウェア: WaitIO/MP

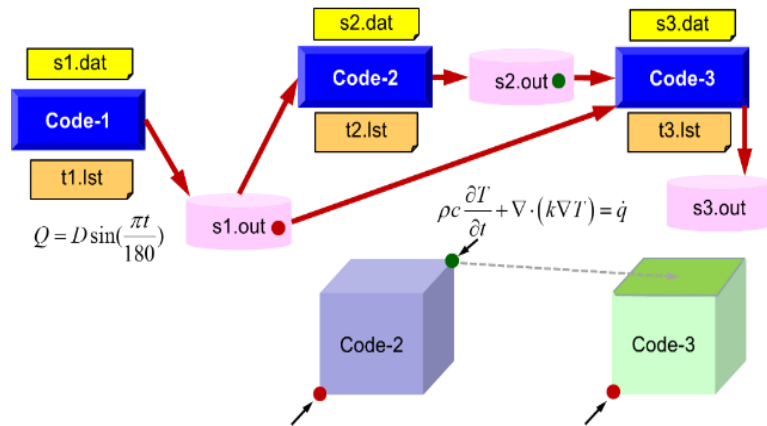
h3-Open-SYS/WaitIOの概要とプログラミング

h3-Open-SYS/WaitIO(WaitIO)

- 異種システム間でのデータ通信を可能にする通信ライブラリ
 - 既存アプリケーションコードに後付けで追加・修正可能
- WaitIOの活用方法：
 1. 異種アプリケーション間のデータ通信手法:
 - WaitIO API or WaitIO-MPI Conversion Library
 - データ共有をWaitIO-Socketで実現
 2. 複数の異種システム上で既存アプリケーション協調実行:
 - WaitIO-MPI Conversion Library
 - より密な協調計算の実現
 3. 上位のカップラーライブラリ利用:
 - h3-Open-UTIL/MPを用いた高度な連成計算機能の利用(既説明済)

WaitIO:異種アプリケーション間のデータ通信手法

- 既存の連成アプリケーションではファイルによるデータ連携が多い
 - ファイル共有によるデータ連携をWaitIO通信によるデータ連携に変換
- 有限要素法による三次元非定常熱伝導解析Toyプログラムによる実例
 - 3つのプログラムから構成される連成アプリケーション



	Code-1	Code-2	Code-3
概要	発熱量(点源)計算	FEMによる非定常三次元熱伝導方程式, MPI並列化	FEMによる非定常三次元熱伝導方程式, MPI並列化
入力	s1.dat	s2.dat s1.out	s3.dat s1.out, s2.out
出力	s1.out, t1.lst	s2.out, t2.lst	s3.out, t3.lst

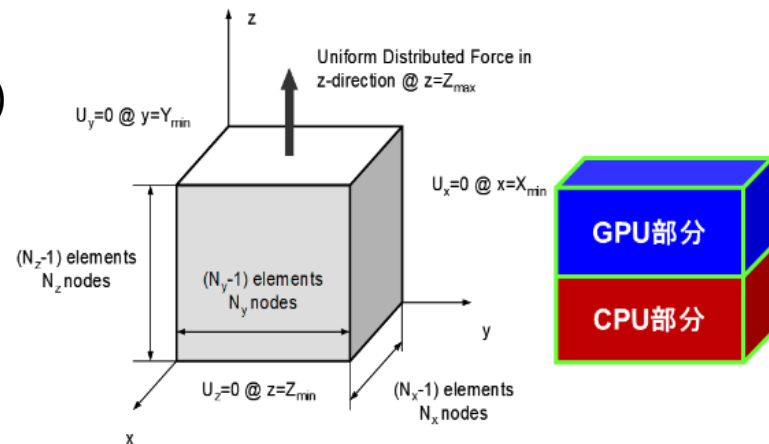
Code-3: Code-2の出力を読み、 $Z=Z_{\max}$ の面に強制温度固定条件

WaitIO:既存アプリケーションの協調実行

- 適したシステムで適した処理を協調実行
 - 複数システムを融合、より大きな問題を解く
 - システム毎の欠点を補完したアプリケーション実行
 - 特性の異なる場合(例：線形問題と非線形問題)が混在するコードを適したシステムで分散実行

• 応用例

- GeoFEM/CubeのToyプログラム
 - 領域分割して、GPUとCPUで協調処理
- メモリ量の違い、CPUコア数の違いで負荷分散



h3-Open-SYS/WaitIO(WaitIO)のAPI概要

- WaitIOは複数MPIアプリケーション間を融合する通信ライブラリ

- WaitIO Instance

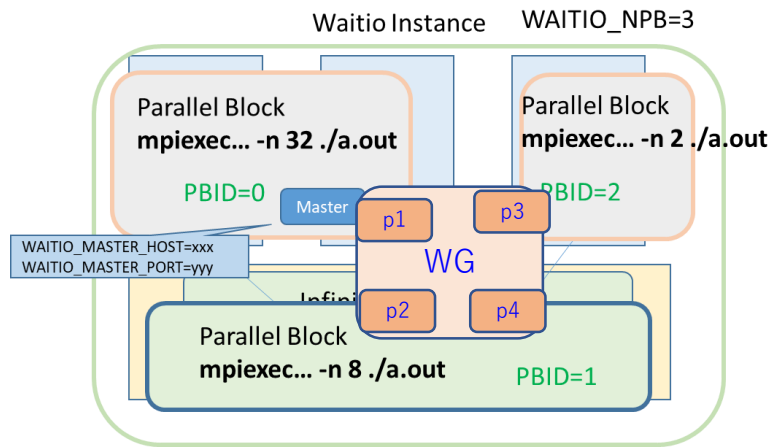
- WaitIOシステム全体を示す

- Parallel Block(PB)

- 各アプリケーションが実行されるプロセスグループ
 - PBID識別子：数値
 - 環境変数WAITIO_PBIDで指定
 - PB数は環境変数WAITIO_NPBで指定
 - PB MASTER：PBID=0
 - 全体のデータ同期を実行
 - 環境変数WAITIO_MASTER_HOSTとWAITIO_MASTER_PORTで指定

- WaitIO Group(WG)

- WaitIO Instanceを構成するPB内の任意プロセスから構成される相互に通信可能グループ



WaitIO API

- APIは通信実現の最小限に限定
 - グループ作成、isend(), irecv(), wait()とmisc関数を提供
 - TAGサポート(64bit), ANY_SOURCE, ANY_TAGサポートなし

WaitIO API	概要
waitio_isend	Non-Blocking送信
waitio_irecv	Non-Blocking受信
waitio_wait	送受信完了待ち合わせ
waitio_init	WaitIO初期化
waitio_finalize	WaitIO終了
waitio_get_nprocs	PB毎の参加プロセス数獲得
waitio_create_group waitio_create_group_wranks	PB間通信グループ生成（メンバ配列指定、関数指定）
waitio_group_rank	グループ内Rankの獲得
waitio_group_size	グループサイズの獲得
waitio_pb_size	全PBのサイズ獲得
waitio_pb_rank	全PB内Rankの獲得

WaitIO API: 常用API概要

- 初期化API: `int waitio_init(int timeout);`
 - 初期化を実施する。MPIにおけるMPI_Init()と同様に全プロセスが必ず最初に呼ぶ必要がある。本関数の実行によりマスタープロセスが立ち上がり、全員が参加するまでtimeout秒待つ。
- 各PBを構成するプロセス数獲得API: `int waitio_get_nprocs(int ary[]);`
 - 返り値は WAITIO Instanceに含まれる並列プロセス数(環境変数WAITIO_NPBで設定した数)
 - aryに各PBに含まれるプロセス数が返る
- 終了API : `int waitio_finalize();`
 - 終了処理を実施する。実行完了後、再びwaitio_init()を呼び出すことはできない。
- グループ作成API :
`waitio_group_t waitio_create_group(int gid, waitio_filter_func_t[], int order[]);`
 - 本関数は構成される各PBに対してPB内プロセスを選別する関数(waitio_filter_func_t)を与え選別した結果をorderの順に並べてWGを作成する
- グループ作成API :
`waitio_group_t waitio_create_group_wranks(int gid, int *rankap[], int order[]);`
 - 本関数は構成される各PBに対してPB内のMPIプロセス毎に選別するMPI rank番号の配列 (終端は-1) を与えた結果をorderの順に並べてWGを作成する

WaitIO API: 常用API概要2

- 通信API : `int waitio_isend(waitio_group_t group, int dst, char *buf, size_t len, unsigned long tag, waitio_req_t *req);`
 - group内のランクdstに送信する。
- 通信API : `int waitio_irecv(waitio_group_t group, int src, char *buf, size_t len, unsigned long tag, waitio_req_t *req);`
 - group内のランクsrcから受信する。ただしMPIのMPI_ANY_SOURCE, MPI_ANY_TAGの概念はない。
- 通信API : `int waitio_wait(waitio_req_t *req);`
 - reqで指定したisend, irecv処理の完了を待つ

MPIによる簡単なPingPongプログラム

```
/* -*- 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();
}
```

- 偶数rankのプロセスから奇数rankプロセスにPing-Pongするプログラム

WaitIOによる簡単なPingPongプログラム

```
/* -*- 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();
}
```

- 初期化以外は
Non-Blocking
MPIプログラム
と同等

WaitIO初期化/終了

```
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, &wrnk);
```

```
waitio_finalize();
```

- `waitio_init(10)`
 - 10 secのタイムアウト指定でWaitIO初期化
- `waitio_create_group(0, func, array);`
 - それぞれのPBIDを持つMPIプロセスに対してfunc関数の実行結果がtrue==1となるプロセスの集合のGroupを生成する
 - truef関数はすべてのMPIプロセスをWaitIO Groupとして定義
 - array[]はPB毎の順序を数値の小さな順に指定。WAITIO NPB分準備必要、要素はPBIDがindex、数値が同じ場合はPBID順
- `waitio_group_rank(grp1, &wrnk);`
 - WaitIOのPB GROUP内のランク番号
- `waitio_finalize();`
 - WaitIOの終了

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受信関数
 - 引数: Group, 相手先, バッファへのポインタ, バイト数, tag, Request構造体へのポインタ
- **waitio_wait**(&req)
 - 指定Requestの完了待ち
 - WaitIOの送受信処理はこのwaitio_wait関数内で開始・実行される。
- **waitio_isend**(grp1, wrank-1, (char *)buf, 4, 0, &req);
 - Non Blocking送信関数
 - 引数: Group, 相手先, バッファへのポインタ, バイト数, tag, Request構造体へのポインタ

WaitIO-MPI Conversionライブラリ

- WaitIOはMPIのDatatypeの概念を持たない
 - MPIプログラムの移植が煩雑
- 機械的書き換えが可能なWaitIO-MPI Conversionライブラリ準備

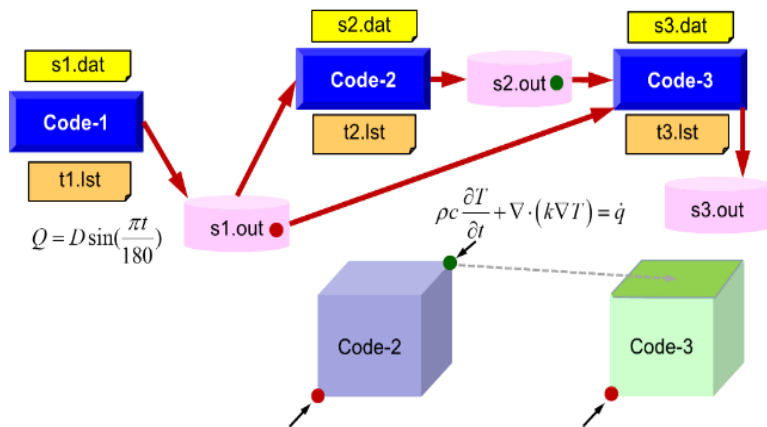
WaitIO-MPI API (2022/9/1現在)	概要
waitio mpi isend	WaitIO版 MPI_Isend
waitio mpi irecv	WaitIO版 MPI_Irecv
waitio mpi reduce	WaitIO版 MPI_Reduce
waitio mpi bcast	WaitIO版 MPI_Bcast
waitio mpi allreduce	WaitIO版 MPI_Allreduce
waitio mpi waitall	WaitIO版 MPI_Waitall
waitio create universe	WaitIO 初期化関数(全ランク)
waitio create universe pbhead	WaitIO 初期化関数(各PBランク0)
waitio mpi gather	WaitIO版 MPI_Gather
waitio mpi algather	WaitIO版 MPI_Algather
waitio mpi scatter	WaitIO版 MPI_Scatter
waitio mpi scatterv	WaitIO版 MPI_Scatterv
waitio mpi gatherv	WaitIO版 MPI_Gatherv
waitio mpi barrier	WaitIO版 MPI_Barrier
waitio mpi type size	WaitIO版 MPI_Type_size

WaitIO-MPI Conversion API: 常用API概要

- 初期化API: `int waitio_create_universe (WAITIO_MPI_Comm *commp) ;`
 - 初期化を実施する。MPIにおけるMPI_Init()と同様に全プロセスが必ず最初に呼ぶ必要がある。
 - すべてのプロセスが参加するWGが生成される。 `WAITIO_MPI_Comm` は `waitio_group_t` と同定義
- 初期化API: `int waitio_create_universe_pbhead (WAITIO_MPI_Comm *commp) ;`
 - 初期化を実施する。MPIにおけるMPI_Init()と同様に全プロセスが必ず最初に呼ぶ必要がある。
 - 各PBのランク0番プロセスが参加するWGが生成される
- 通信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` は `waitio_req_t` と同定義
- 通信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` は `waitio_req_t` と同定義
 - `MPI_Irecv`と異なりMPI_Status引数無し
- 通信API : `int waitio_wait(waitio_req_t *req);`
 - `req`で指定した処理の完了を待つ。 `WAITIO_MPI_Comm` は `waitio_group_t` と同定義
- 他、集団通信関数はMPI関数の定義に準ずる。

WaitIO:異種アプリケーション間のデータ通信手法

- 既存の連成アプリケーションではファイルによるデータ連携が多い
 - ファイル共有によるデータ連携をWaitIO通信によるデータ連携に変換
- 有限要素法による三次元非定常熱伝導解析Toyプログラムによる実例
 - 3つのプログラムから構成される連成アプリケーション



	Code-1	Code-2	Code-3
概要	発熱量(点源)計算	FEMによる非定常三次元熱伝導方程式, MPI並列化	FEMによる非定常三次元熱伝導方程式, MPI並列化
入力	s1.dat	s2.dat s1.out	s3.dat s1.out, s2.out
出力	s1.out, t1.lst	s2.out, t2.lst	s3.out, t3.lst

Code-3: Code-2の出力を読み、 $Z=Z_{\max}$ の面に強制温度固定条件

WaitIO: Toyプログラムの書き換え

- コード例：Code-1のWaitIO修正

```
001  implicit REAL*8(A-H,O-Z)
002  real(kind=8) :: Interval
003  include 'mpif.h'

004  call MPI_Init(ierr)
005  open (11,file='s1.dat',status='unknown')
006  read (11,*) ITERmax, Period, Interval, Val
007  write (*,'(a,i8)') '##ITERmax ', ITERmax
008  write (*,'(a,1pe16.6)') '##Period ', Period
009  write (*,'(a,1pe16.6)') '##Interval', Interval
010  write (*,'(a,1pe16.6//)') '##Val ', Val
011  close (11)

012  open (12,file='s1.out',status='unknown')

013  pi = 4.d0*datan(1.d0)
014  coef= pi/180.d0
015  time= 0.d0
016  S0time= mpi_wtime ()
```

```
017  do
018    S1time= mpi_wtime ()

019    do
020      do iter= 1, ITERmax
021        a= 1.d0
022      enddo
023      E0time= mpi_wtime (ierr)
024      if ((E0time-S1time).gt.Interval) exit
025    enddo

026    ttt= E0time - S0time
027    Source= Val * dsin(ttt*coef)
028 !    rewind (12)
029    write (12,'(2(1pe16.6))') ttt, Source
030  enddo

031  call MPI_Finalize()
032  stop
033  end
```

Code-1: オリジナル

2024/10/30

WaitIO/MP

講習会

WaitIO: Toyプログラムの書き換え:2

- Open/Read-Writeを isend-irecv/waitに書き換え
 - MPI Non-blocking送受信関数と同様

Code-1: WaitIO修正

```
001 implicit REAL*8(A-H,O-Z)
002 real(kind=8) :: Interval
003 integer :: dest
004 integer(kind=4), dimension(:,:), save, allocatable :: sta1
005 integer(kind=4), dimension(:,:), save, allocatable :: req1
```

```
006 include 'mpif.h'
007 include 'waitio_mpf.h'
```

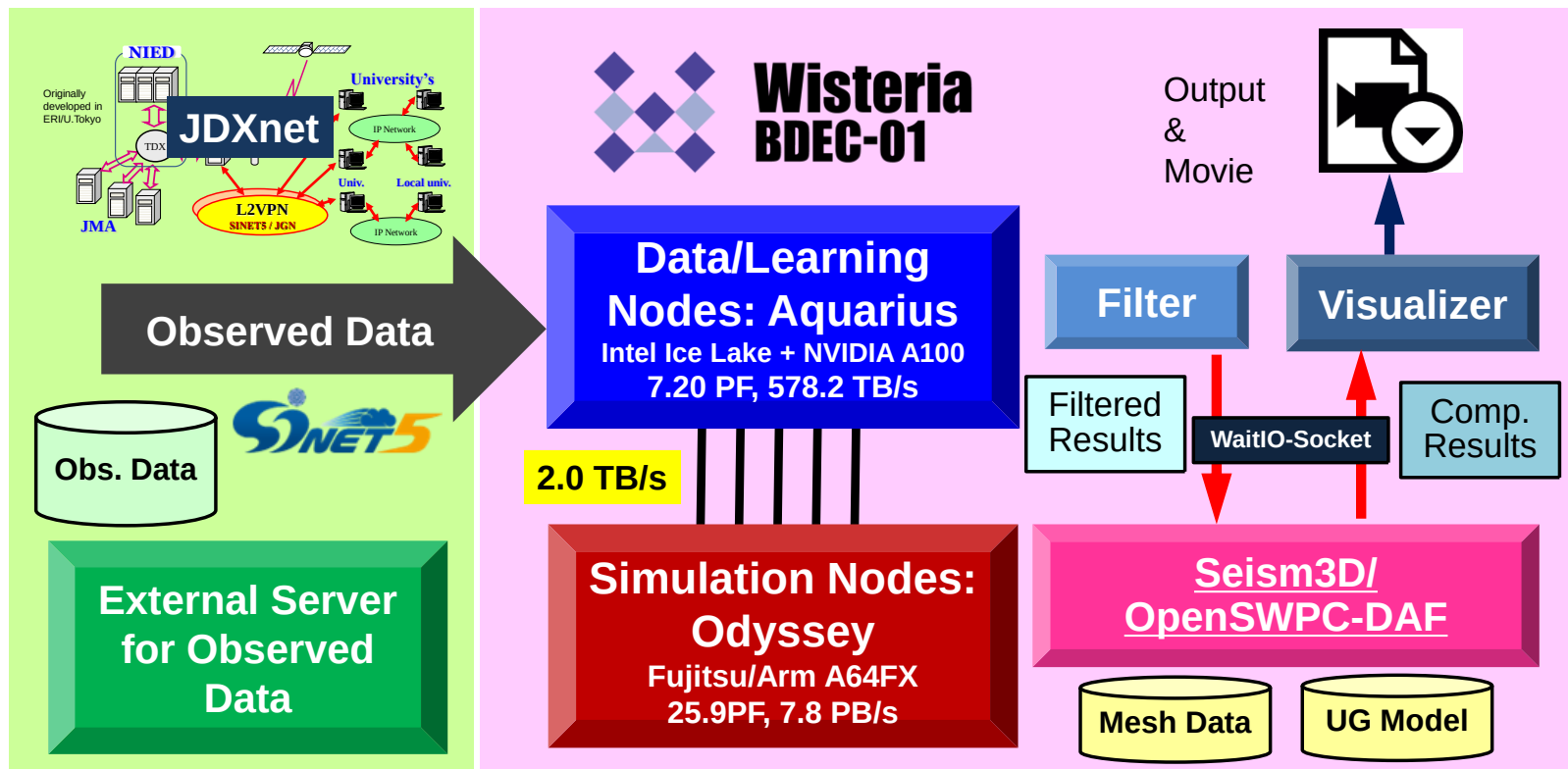
```
008 call MPI_Init(ierr)
009 open (11,file='s1.dat',status='unknown')
010 read (11,*) ITERmax, Period, Interval, Val
011 write (*,'(a,i8)') '##ITERmax ', ITERmax
012 write (*,'(a,1pe16.6)') '##Period ', Period
013 write (*,'(a,1pe16.6)') '##Interval', Interval
014 write (*,'(a,1pe16.6//)') '##Val ', Val
015 close (11)
```

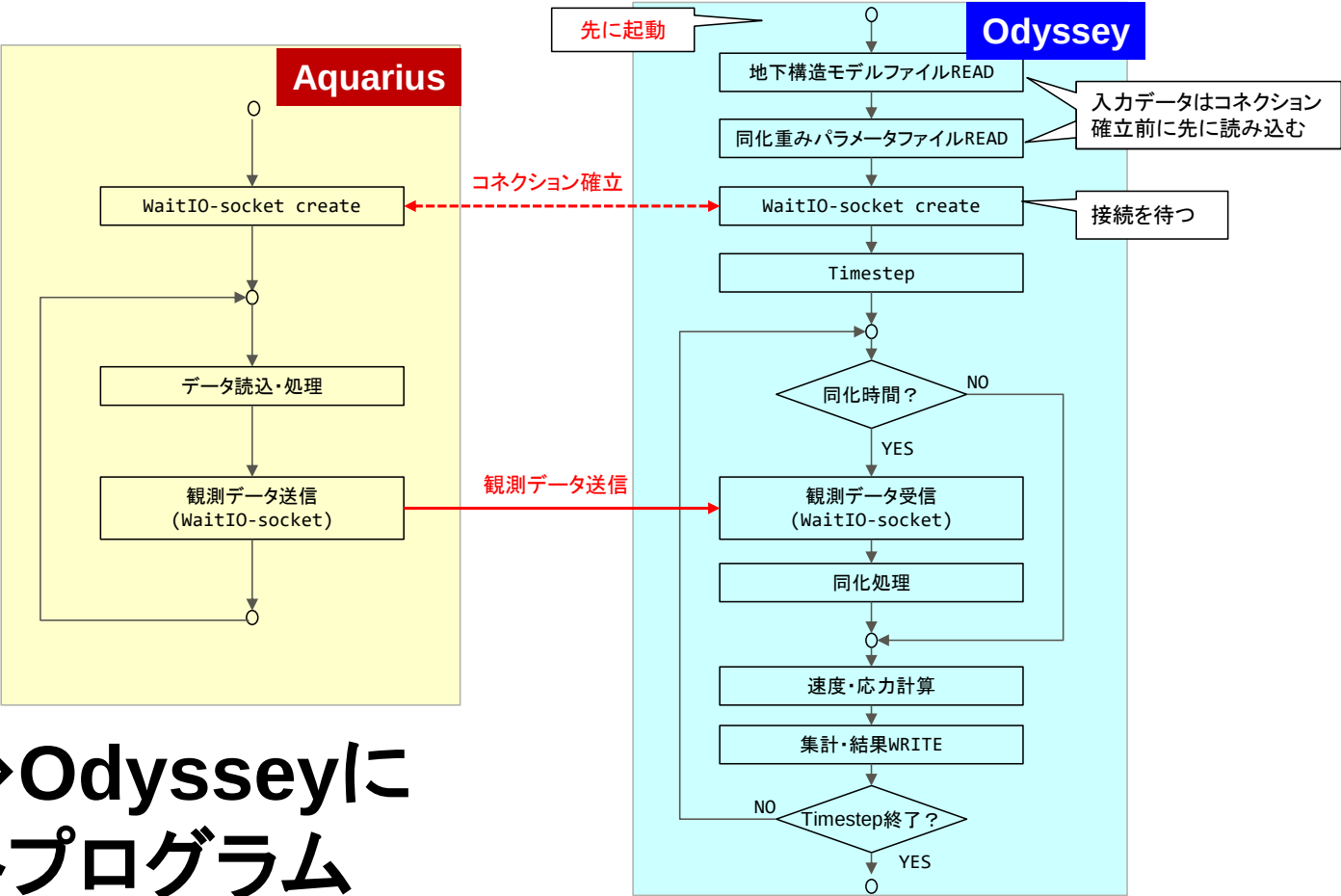
```
016 allocate (sta1(WAITIO_STATUS_SIZE,2*2))
017 allocate (req1(WAITIO_REQUEST_SIZE,2*2))
018 call WAITIO_CREATE_UNIVERSE_PBHEAD (WAITIO_SOLVER_COMM, ierr)
```

```
019 open (12,file='s1.out',status='unknown')
```

```
037 write (*,'(2(1pe16.6))') ttt, Source
038 do dest=1, 2
039   call WAITIO_MPI_Isend (ttt, 1,
040    & WAITIO_MPI_DOUBLE_PRECISION,
041    & dest, 0, WAITIO_SOLVER_COMM, req1(1,2*(dest-1)+1), ierr)
042   if(ierr.ne.0) goto 100
043   call WAITIO_MPI_Isend (Source, 1,
044    & WAITIO_MPI_DOUBLE_PRECISION,
045    & dest, 0, WAITIO_SOLVER_COMM, req1(1,2*(dest-1)+2), ierr)
046   if(ierr.ne.0) goto 100
047 enddo
048 call WAITIO_MPI_Waitall (4, req1, sta1, ierr)
049 if(ierr.ne.0) goto 100
050 enddo
051 100 continue
```


観測データ同化による長周期地震動リアルタイム予測 2021年度成果: Wisteria/BDEC-01への実装





Aquarius⇒Odysseyに おけるテストプログラム WaitIO-Socket通信

WaitIO-Socketによる通信 (Aquarius側)

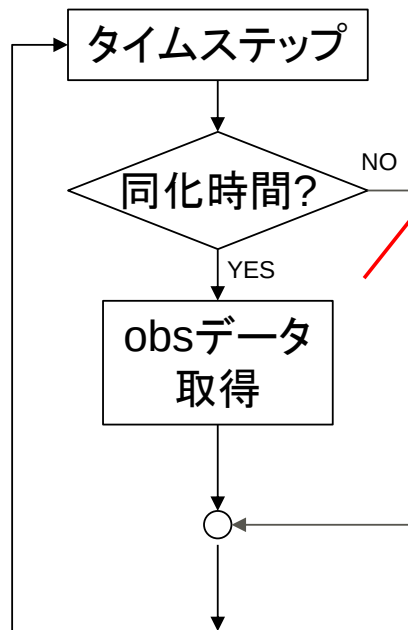
```

program dmy_filter
<省略: 型宣言等>
call mpi_init (ierr)
call mpi_comm_size (MPI_COMM_WORLD, nprocs, ierr)
call mpi_comm_rank (MPI_COMM_WORLD, myrank, ierr)
call WAITIO_CREATE_UNIVERSE (WAITIO_COMM_UNIVERSE, ierr)

if (myrank==0) then
  open(100,file='./obsfile_list.txt', form='formatted', status='old', iostat=ierr)
  do i=1,300
    <省略: obsデータ読み込み処理>
    print *, "Send obs data ..... "
    call WAITIO_MPI_ISEND (NTMAX1_o, 1, WAITIO_MPI_INTEGER, 2,1, WAITIO_COMM_UNIVERSE, req(1,1), ierr)
    call WAITIO_MPI_ISEND (DT_o, 1, WAITIO_MPI_FLOAT, 2,2, WAITIO_COMM_UNIVERSE, req(1,2), ierr)
    call WAITIO_MPI_ISEND (NST_o, 1, WAITIO_MPI_INTEGER, 2,3, WAITIO_COMM_UNIVERSE, req(1,3), ierr)
    call WAITIO_MPI_ISEND (AT_o, 1, WAITIO_MPI_FLOAT, 2,4, WAITIO_COMM_UNIVERSE, req(1,4), ierr)
    call WAITIO_MPI_ISEND (T0_o, 1, WAITIO_MPI_FLOAT, 2,5, WAITIO_COMM_UNIVERSE, req(1,5), ierr)
    call WAITIO_MPI_ISEND (ISO_X_o, NSMAX, WAITIO_MPI_INTEGER, 2,6, WAITIO_COMM_UNIVERSE, req(1,6), ierr)
    call WAITIO_MPI_ISEND (ISO_Y_o, NSMAX, WAITIO_MPI_INTEGER, 2,7, WAITIO_COMM_UNIVERSE, req(1,7), ierr)
    call WAITIO_MPI_ISEND (ISO_Z_o, NSMAX, WAITIO_MPI_INTEGER, 2,8, WAITIO_COMM_UNIVERSE, req(1,8), ierr)
    call WAITIO_MPI_ISEND (ISTX_o, NST, WAITIO_MPI_INTEGER, 2,9, WAITIO_COMM_UNIVERSE, req(1,9), ierr)
    call WAITIO_MPI_ISEND (ISTY_o, NST, WAITIO_MPI_INTEGER, 2,10, WAITIO_COMM_UNIVERSE, req(1,10), ierr)
    call WAITIO_MPI_ISEND (ISTZ_o, NST, WAITIO_MPI_INTEGER, 2,11, WAITIO_COMM_UNIVERSE, req(1,11), ierr)
    call WAITIO_MPI_ISEND (STC_o, 6*NST, WAITIO_MPI_CHAR, 2,12, WAITIO_COMM_UNIVERSE, req(1,12), ierr)
    call WAITIO_MPI_ISEND (VxAll_obs, NST*NOBS_LEN, WAITIO_MPI_FLOAT, 2,13, WAITIO_COMM_UNIVERSE, req(1,13), ierr)
    call WAITIO_MPI_ISEND (VyAll_obs, NST*NOBS_LEN, WAITIO_MPI_FLOAT, 2,14, WAITIO_COMM_UNIVERSE, req(1,14), ierr)
    call WAITIO_MPI_ISEND (VzAll_obs, NST*NOBS_LEN, WAITIO_MPI_FLOAT, 2,15, WAITIO_COMM_UNIVERSE, req(1,15), ierr)
    call WAITIO_MPI_WAITALL (15, req, status, ierr)
    call sleep(1)
  enddo
  close (100)
endif
call WAITIO_FINALIZE (ierr)
call mpi_finalize (ierr)
end

```

WaitIO-Socketによる通信 (Odyssey側)



```
OPEN(40, FILE=trim(IN_OBS),FORM='UNFORMATTED',
      STATUS='OLD', IOSTAT=IER)
```

<略>

```
READ (40) NTMAX1_o, DT_o, NST_o
READ (40) AT_o, T0_o
READ (40) ISO_X_o,ISO_Y_o,ISO_Z_o,
        ISTX_o,ISTY_o,ISTZ_o,STC_o
READ (40) VxAll_obs
READ (40) VyAll_obs
READ (40) VzAll_obs
```

WaitIO-Socketに変更

```
call WAITIO_MPI_Irecv (NTMAX1_o, 1, WAITIO_MPI_INTEGER, 0,1, WAITIO_COMM_UNIVERSE,...)
call WAITIO_MPI_Irecv (DT_o, 1, WAITIO_MPI_FLOAT, 0,2, WAITIO_COMM_UNIVERSE,...)
call WAITIO_MPI_Irecv (NST_o, 1, WAITIO_MPI_INTEGER, 0,3, WAITIO_COMM_UNIVERSE,...)
call WAITIO_MPI_Irecv (AT_o, 1, WAITIO_MPI_FLOAT, 0,4, WAITIO_COMM_UNIVERSE,...)
call WAITIO_MPI_Irecv (T0_o, 1, WAITIO_MPI_FLOAT, 0,5, WAITIO_COMM_UNIVERSE,...)
call WAITIO_MPI_Irecv (ISO_X_o, NSMAX, WAITIO_MPI_INTEGER, 0,6, WAITIO_COMM_UNIVERSE,...)
call WAITIO_MPI_Irecv (ISO_Y_o, NSMAX, WAITIO_MPI_INTEGER, 0,7, WAITIO_COMM_UNIVERSE,...)
call WAITIO_MPI_Irecv (ISO_Z_o, NSMAX, WAITIO_MPI_INTEGER, 0,8, WAITIO_COMM_UNIVERSE,...)
call WAITIO_MPI_Irecv (ISTX_o, NST, WAITIO_MPI_INTEGER, 0,9, WAITIO_COMM_UNIVERSE,...)
call WAITIO_MPI_Irecv (ISTY_o, NST, WAITIO_MPI_INTEGER, 0,10, WAITIO_COMM_UNIVERSE,...)
call WAITIO_MPI_Irecv (ISTZ_o, NST, WAITIO_MPI_INTEGER, 0,11, WAITIO_COMM_UNIVERSE,...)
call WAITIO_MPI_Irecv (STC_o, 6*NST, WAITIO_MPI_CHAR, 0,12, WAITIO_COMM_UNIVERSE,...)
call WAITIO_MPI_Irecv (VxAll_obs, NST*NOBS_LEN, WAITIO_MPI_FLOAT, 0,13, WAITIO_COMM_UNIVERSE,...)
call WAITIO_MPI_Irecv (VyAll_obs, NST*NOBS_LEN, WAITIO_MPI_FLOAT, 0,14, WAITIO_COMM_UNIVERSE,...)
call WAITIO_MPI_Irecv (VzAll_obs, NST*NOBS_LEN, WAITIO_MPI_FLOAT, 0,15, WAITIO_COMM_UNIVERSE,...)
```

pHEAT-3Dアプリケーションへの適用例

- pHEAT-3Dアプリケーション(Fortran+MPI)をWaitIO-MPI Conversionライブラリを用いて移植(solver部分のみ変換)
 - WaitIO-MPI Conversionコード変換は機械的に変換可能: 数分で変換

```
include 'mpif.h'
include 'waitio_mpf.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)
```

2024/10/30

WaitIO/MPI

講習会

休憩 10分

Next

h3-Open-SYS/WaitIOと
h3-Open-UTIL/MPを使ってみよう

15:50 – 17:30

h3-Open-SYS/WaitIOと h3-Open-UTIL/MPを使ってみよう

15:50 – 17:30

Wisteria/BDEC-01システム利用イントロダクション:20分
サンプルプログラムの実行WaitIO:20分
サンプルプログラムの実行MP: 50分
自由に動かしてみよう・Q&A: 10分

Wisteria/BDEC-01システム利用 イントロダクション

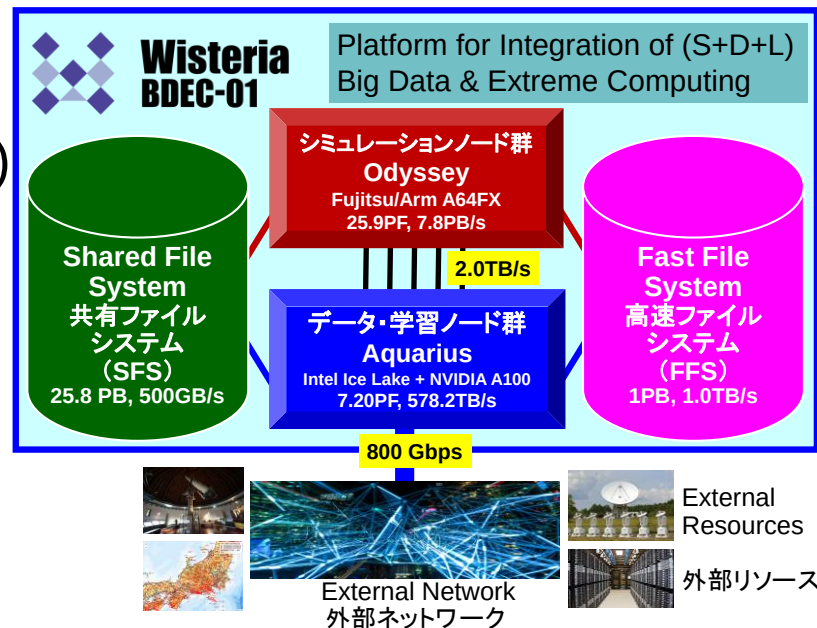
Wisteria/BDEC-01システム利用 イントロダクション

- システム構成再び
- バッチ用途インタラクティブ利用
- バッチ処理システム利用の基本
 - 基本コマンド
- 連成計算用異種連携システム＜今回利用＞

Wisteria/BDEC-01

- 2021年5月14日運用開始
 - 東京大学柏Ⅱキャンパス
- 33.1 PF, 8.38 PB/sec., **富士通製**
 - ~4.5 MVA(空調込み), ~360m²
- Hierarchical, Hybrid, Heterogeneous (h3)
- 2種類のノード群**
 - シミュレーションノード群(S, SIM) : Odyssey**
 - 従来のスパコン
 - Fujitsu PRIMEHPC FX1000 (A64FX), 25.9 PF
 - 7,680ノード(368,640 コア), 20ラック, Tofu-D
 - データ・学習ノード群(D/L, DL) : Aquarius**
 - データ解析, 機械学習
 - Intel Xeon Ice Lake + NVIDIA A100, 7.2 PF
 - 45ノード(Ice Lake:90基, A100:360基), IB-HDR
 - 一部は外部リソース(ストレージ, サーバー, センサーネットワーク他)に直接接続
- ファイルシステム: 共有(大容量) + 高速

BDEC:「計算・データ・学習(S+D+L)」
融合のためのプラットフォーム
(Big Data & Extreme Computing)



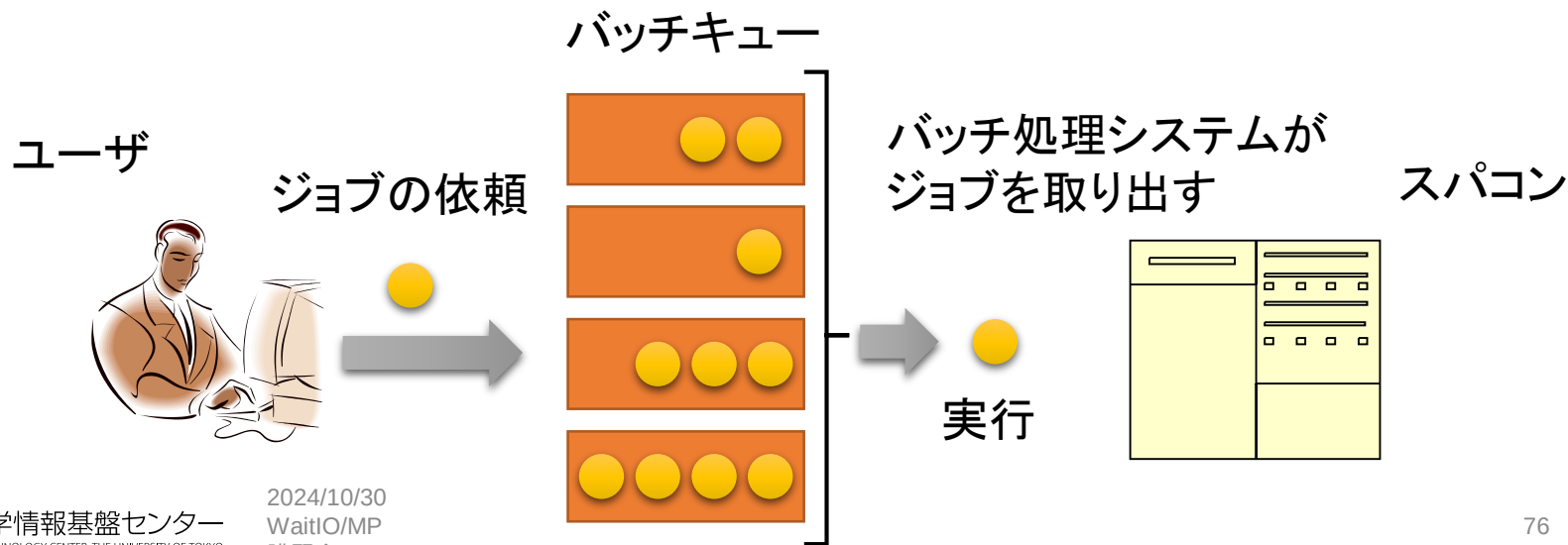
Wisteria/BDEC-01スーパーコンピュータシステムでのジョブ実行形態

- 以下の2通りがあります
- インタラクティブジョブ実行
 - PCでの実行のように、コマンドを入力して実行する方法
 - スパコン環境では、あまり一般的でない
 - デバック用、大規模実行はできない
- バッチジョブ実行
 - バッチジョブシステムに処理を依頼して実行する方法
 - 実行させたい処理をファイル（ジョブスクリプト）で指示する
 - スパコン環境で一般的
 - 大規模実行用

※講義アカウントでは
バッチジョブ実行のみ、
最大12ノード15分まで

バッチ処理とは

- スパコン環境では、通常は、インタラクティブ実行（コマンドラインで実行すること）はできません。
- ジョブはバッチ処理で実行します。
 - キュー: 待ち行列



バッチ処理：リソースグループと利用グループ

- リソースグループ：使用する計算機リソースの指定
 - 用途・規模指定: debug, short, regular, coupler, share, [lecture](#), [tutorial](#)
 - Odyssey指定: 用途・規模指定に-oを付与: [debug-o](#)
 - Aquarius指定: 用途・規模指定に-aを付与: [debug-a](#)
- 利用グループ：参加プロジェクトを指定、
 - トークン、総利用リソース資源量(時間)、ノード数制限

バッチキューの設定のしかた

- Wisteriaでのバッチ処理は、富士通のバッチシステムで管理されています。
- 以下、主要コマンドを説明します。
 - ジョブの投入：`pjsub <ジョブスクリプトファイル名>`
 - 自分が投入したジョブの状況確認：`pjstat`
 - 投入ジョブの削除：`pjdel <ジョブID>`
 - バッチキューの状態を見る：`pjstat --rsc`
 - バッチキューの詳細構成を見る：`pjstat --rsc -x`
 - 投げられているジョブ数を見る：`pjstat -b`
 - 過去の投入履歴を見る：`pjstat -H`
 - 同時に投入できる数／実行できる数を見る：`pjstat --limit`

pjstat --rsc の実行画面例

```
$ pjstat --rsc
```

```
SYSTEM: Odyssey
```

```
RSCGRP
```

```
lecture-o
```

```
lecture8-o
```

```
STATUS  
[ENABLE,START]  
[ENABLE,START]
```

```
NODE  
96  
2x12x16
```

ノードの
利用可能数

```
SYSTEM: Aquarius
```

```
RSCGRP
```

```
lecture-a
```

```
lecture8-a
```

```
STATUS  
[ENABLE,START]  
[DISABLE,STOP]
```

```
NODE GPU  
7 56  
2 16
```

GPUの
利用可能数
(Aquarius)

現在使えるか

使える
キュー名
(リソース
グループ)

pjstat --rsc -x の実行画面例

```
$ pjstat --rsc -x
```

```
SYSTEM: Odyssey
```

RSCGRP	STATUS	MIN_NODE	MAX_NODE	MAX_ELAPSE	REMAIN_ELAPSE	MEM(GiB)	PROJECT
lecture-o	[ENABLE,START]	1	12	00:15:00	00:15:00	28	gt68
lecture8-o	[DISABLE,STOP]	1	12	00:15:00	--:--:--	28	gt68

```
SYSTEM: Aquarius
```

RSCGRP	STATUS	MIN_NODE	MAX_NODE	AVAIL_GPU	MAX_ELAPSE	REMAIN_ELAPSE	MEM(GiB)	PROJECT
lecture-a	[ENABLE,START]	1	1	1,2,4	00:15:00	00:15:00	448	gt68
lecture8-a	[DISABLE,STOP]	1	1	1,2,4	00:15:00	--:--:--	448	gt68

↑
使える
キュー名
(リソース
グループ)

↑
現在
使えるか

↑
ノードの
実行情報

↑
課金情報(財布)
実習では1つのみ

pjstat --rsc -b の実行画面例

```
$ pjstat --rsc -b
```

SYSTEM: Odyssey

RSCGRP	STATUS	TOTAL	RUNNING	QUEUED	HOLD	OTHER	NODE
lecture-o	[ENABLE,START]	0	0	0	0	0	96
lecture8-o	[DISABLE,STOP]	0	0	0	0	0	2x12x16

SYSTEM: Aquarius

RSCGRP	STATUS	TOTAL	RUNNING	QUEUED	HOLD	OTHER	NODE
lecture-a	[ENABLE,START]	0	0	0	0	0	7
lecture8-a	[DISABLE,STOP]	0	0	0	0	0	2

使えるキュー名
(リソースグループ)

現在
使えるか

ジョブの
総数

実行している
ジョブの数

待たされている
ジョブの数

ノードの
利用可能数

JOBスクリプトサンプルの説明（ピュアMPI）（hello-pure.bash, C言語、Fortran言語共通）

```
#!/bin/bash
#PJM -L rscgrp=lecture8-o
#PJM -L node=12
#PJM --mpi proc=576
#PJM -L elapse=0:01:00
#PJM -g gt68
```

```
module load fj fjmpi
mpiexec ./hello
```

リソースグループ名
:lecture8-o

利用ノード数、
MPIプロセス数

実行時間制限
:1分

利用グループ名
:gt68

MPIジョブを $48 \times 12 = 576$ プロセス
で実行する。

教育利用のジョブクラス: バッチジョブ (Odyssey)

キュー名	ノード数	制限時間 (Elapsed)	メモリ容量 (GB)
lecture-o (教育利用全体で共有)	1 ~ 12	15 分	28
tutorial-o (この講習会の時間中のみ使用可能)	1 ~ 12	15 分	28

ノード当たり 48コア => 最大576コアまで使用可能！

教育利用のジョブクラス: バッチジョブ (Aquarius)

キュー名	GPU数	制限時間 (Elapsed)	メモリ容量 (GB)
lecture-a (教育利用全体で共有)	1,2,4	15 分	56/GPU
tutorial-a (この講習会の時間中のみ使用可能)	1,2,4	15 分	56/GPU

share-aと同様の使い方

Wisteria/BDEC-01における 連成ジョブ実行環境

h3-Open-SYS/WaitIO(WaitIO)の概要

- WaitIOは複数MPIアプリケーション間を融合する通信ライブラリ

- WaitIO Instance

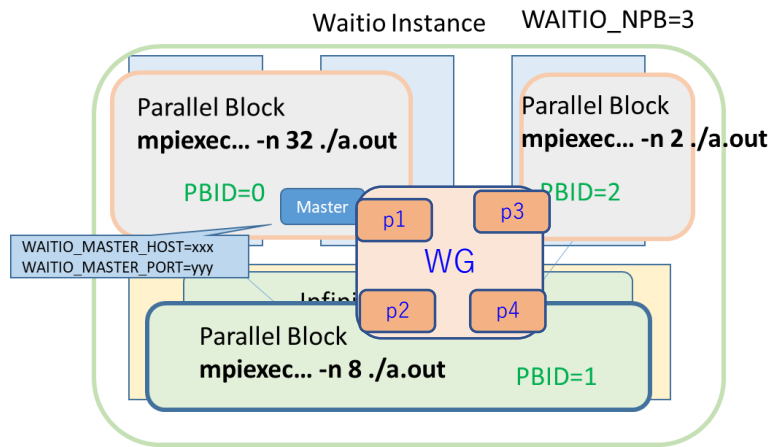
- WaitIOシステム全体を示す

- Parallel Block(PB)

- 各アプリケーションが実行されるプロセスグループ
 - PBID識別子：数値
 - 環境変数**WAITIO_PBID**で指定
 - PB数は環境変数**WAITIO_NPB**で指定
 - PB MASTER：**WAITIO_PBID=0**
 - 全体のデータ同期を実行
 - 環境変数**WAITIO_MASTER_HOST**と**WAITIO_MASTER_PORT**で指定

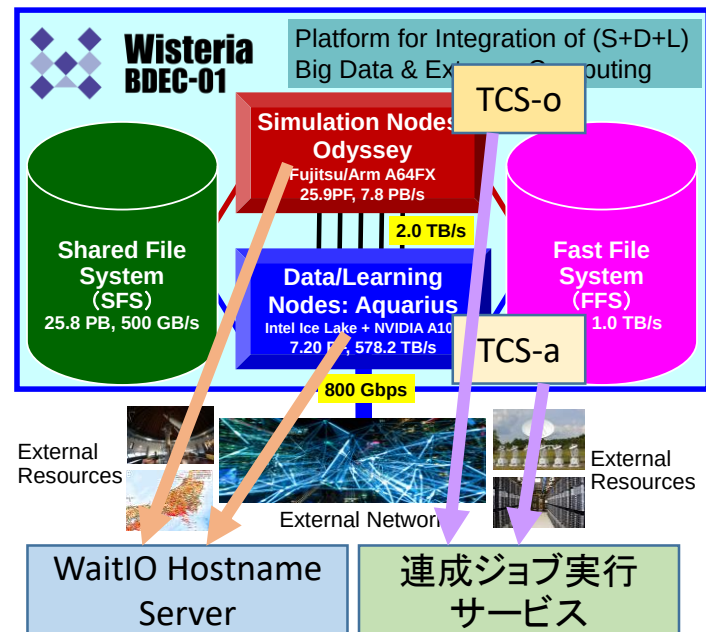
- WaitIO Group(WG)

- WaitIO Instanceを構成するPB内の任意プロセスから構成される相互に通信可能グループ



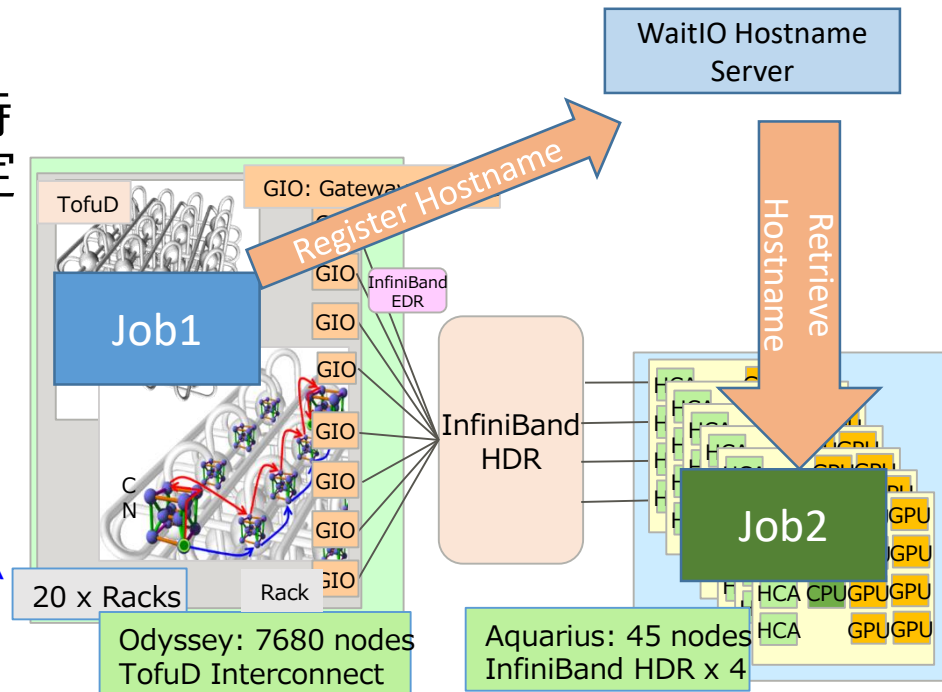
異種バッチ処理システム間ジョブ実行連携機構

- OdysseyとAquarius上での連成ジョブ実行仕組み
 - 通常実行ジョブと、連成実行ジョブの区別
 - 複数システムのリソース空き時に同時実行
 - WaitIO-Socket実行環境設定
- 連成ジョブ実行サービスとWaitIO Hostname Serverで実現
 - waitio モジュールで環境設定
 - `module load waitio@バッチスクリプト`
 - WaitIO利用に必要な設定を実施



WaitIO Hostname Server

- バッチ実行は実行時にホスト名決定
 - 複数ジョブ間で実行ホスト情報交換要
- waitio-serv: 固定ホスト上で実行、特定のSocket Portに接続して各種設定
 - waitio moduleで環境設定
- waitio-servコマンド
 - Odyssey用: waitio-serv-a64fx
 - Aquarius用: waitio-serv(-intel)
- waitio-servオプション
 - 設定: `waitio-serv -m `hostname``
 - 獲得: `waitio-serv -c`



教育利用のジョブクラス: バッチジョブ (Odyssey+Aquarius)

キュー名	ノード数	制限時間 (Elapsed)	メモリ容量 (GB)
lecture-o (教育利用全体で共有)	1 ~ 12	15 分	28
coupler-lec-o (この講習会関連のみ使用可能)	1 ~ 12	15 分	28
lecture-a (教育利用全体で共有)	1	15 分	508
coupler-lec-a (この講習会関連のみ使用可能)	1	15 分	508

連成プログラムの実行

- バッチスクリプトでの設定

- 同一ユーザ名での実行： 違うユーザの実行は区別されます

```
#PJM -N "SYStest-waitio" # Job名を一致  
#PJM -L rscgrp=coupler-lec-[ao] #リソースグループは coupler-xxx  
#PJM -g jh21xxxxo # group名も一致(最後のoaは無視される)
```

- WaitIO ModuleとPB(Parallel Block)の設定

PBID=0

```
module load waitio  
export WAITIO_MASTER_HOST=`hostname`  
export WAITIO_MASTER_PORT=7100 # Socket Port  
export WAITIO_PBID=0  
export WAITIO_NPB=2  
waitio-serv-<arch> -m $WAITIO_MASTER_HOST
```

`hostname`: IPアドレス取得に利用、
Aquariusでは `hostname`-ib0 がOdysseyと通信可能なホスト名

PBID> 0

```
module load waitio  
export WAITIO_MASTER_HOST=`waitio-serv-<arch> -c`  
export WAITIO_MASTER_PORT=7100 # Socket Port  
export WAITIO_PBID=1  
export WAITIO_NPB=2
```

<arch>: Odyssey向けスクリプト= a64fx
Aquarius向けスクリプト= intel

サンプルプログラムの実行 WaitIO

WaitIOサンプルプログラム実行概要

- サンプルプログラムの実行前準備
 1. ポータルサイトへのssh公開鍵登録（事前送付済）
 2. Wisteriaへのログイン（事前送付済）
 3. Linux操作の基本（ディレクトリ変更、エディター、適宜）
- サンプルプログラム実行
 1. サンプルプログラムの/workディレクトリへの展開
 2. コンパイル
 3. バッチスクリプト準備・修正
 4. バッチ処理実行

事前準備：問題なくできましたでしょうか？

-問題ありましたらBreakout roomにて個別に対応いたします-

- SSH環境の準備
- Wisteria/BDEC-01システムへのログイン
 - 公開鍵の生成
 - スパコンポータルサイトにログイン・公開鍵登録
 - sshコマンドでWisteria/BDEC-01システムへのログイン
- ログインしたら
 - Homeと/workディレクトリの確認（ジョブ実行は/work利用）
 - 手元のPCとの間のファイル転送
 - マニュアル情報の参照
- Zoom利用

WaitIOサンプルプログラム実行:PingPong

- サンプルプログラムの/workディレクトリへの展開
 - `tar -zxf /work/share/waitio/src/examples/PingPong.tgz`
- コンパイル
 - `module load intel impi`
 - `make clean all`
- 実行
 - バッチスクリプトの修正
 - バッチ処理実行
 - `pjsub xxx.sh`
 - 単体: odyssey, aquarius, 複数: odyssey+aquarius(odyssey+odyssey)

WaitIOを用いたPingPongプログラム例

```
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);
049         }
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 }
```

WaitIO-MPI Conversion APIを用いた PingPongプログラム例

```
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 }
```


プログラムのコンパイル

- サンプルプログラムのworkディレクトリへのコピーと展開

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

- コンパイル

```
[z30xxx@wisteria01 sample]$ cd PingPong/  
[z30xxx@wisteria01 PingPong/]$ make clean  
[z30xxx@wisteria01 PingPong/]$ ls  
Makefile  test-a64fx-1.sh test.c          test-mpi.c  
mpi-ltest.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 impi-intel test-a64fx-1.sh test-intel test-mpi.c  
impi-intel mpi-ltest.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/]$
```

プログラムの実行準備: PB数=1

- バッチスクリプトの修正: 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 gt00
#PJM -j
#----- Program execution -----#

module purge
module load waitio
module load fj
module load fjmp
export WAITIO_MASTER_HOST=`hostname`
export WAITIO_MASTER_PORT=7100
export WAITIO_PPID=0
export WAITIO_NPB=1

mpiexec ./test-a64fx
exit
```

プログラムの実行: PB数=1

- バッチスクリプトの起動

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

- 出力: pjstatコマンドでジョブ完了確認後

```
[z30xxx@wisteria01 PingPong/]$ cat test-a64fx.sh.636562.out  
Unloading odyssey  
  WARNING: Did not unuse /work/opt/local/modules/modulefiles/VO/odyssey/core  
  WARNING: Did not unuse /work/opt/local/modules/modulefiles/VO/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/]$
```

プログラムの実行準備: PB数=2(a64fx+a64fx)

- バッチスクリプトの修正: 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 gt00
#PJM -j
#----- Program execution -----#
```

```
hostname
module purge
module load fj
module load fjmp
module load waitio
```

```
export WAITIO_MASTER_HOST=`hostname`
export WAITIO_MASTER_PORT=7100
export WAITIO_PPID=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 gt00
#PJM -j
#----- Program execution -----#
```

```
hostname
module purge
module load fj
module load fjmp
module load waitio
```

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

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

プログラムの実行: PB数=2(a64fx+a64fx)

• バッチスクリプトの起動

```
[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]$
```

• 出力

```
[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: fmpi/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
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 30
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 Connector 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: fmpi/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 Connector 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
```

便利なプログラムの実行状態確認: PB数=2

- 出力: watch コマンドは一定間隔でコマンドを実行: 終了はctrl-C
- % watch pjstat コマンドで一定間隔でpjstatコマンド実行

```
[z30xxx@wisteria01 PingPong/J]$ watch pjstat
```

Watch command: watch pstat

```
z30455@wisteria01:~/workj/examples/PingPong
File Edit View Search Terminal Help
Every 2.0s: pstat wisteria01: Sun Oct 9 14:28:11 2022

Wisteria/BDEC-01 scheduled stop time: 2022/10/27(Thu) 13:00:00 (Remain: 17days 22:31:49)

JOB_ID    JOB_NAME  STATUS  PROJECT  RSCGROUP  START_DATE  ELAPSE  TOKEN  NODE  GPU
659384    test1     HOLD    gt00     coupler-lec-o  --/-- --:--:--  00:00:00  -     1     -
659385    test1     HOLD    gt00     coupler-lec-o  --/-- --:--:--  00:00:00  -     1     -
```

しばらくHOLDが続きます

```
z30455@wisteria01:~/workj/examples/PingPong
File Edit View Search Terminal Help
Every 2.0s: pstat wisteria01: Sun Oct 9 14:30:06 2022

Wisteria/BDEC-01 scheduled stop time: 2022/10/27(Thu) 13:00:00 (Remain: 17days 22:29:53)

JOB_ID    JOB_NAME  STATUS  PROJECT  RSCGROUP  START_DATE  ELAPSE  TOKEN  NODE  GPU
659384    test1     RUNNING gt00     coupler-lec-o  (10/09 14:30)<  00:00:00  -     1     -
659385    test1     RUNNING gt00     coupler-lec-o  (10/09 14:30)<  00:00:00  -     1     -
```

同時に実行開始されます

プログラムの実行準備: PB数=2(a64fx+intel)

- バッチスクリプトの修正: test-a64fx-1.sh, test-intel-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 gt00
#PJM -j
#----- Program execution -----#

hostname
module purge
module load fj
module load fjmp
module load waitio

export WAITIO_MASTER_HOST=`hostname`
export WAITIO_MASTER_PORT=7100
export WAITIO_PPID=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-a
#PJM -L node=1
#PJM --mpi proc=2
#PJM -L elapse=00:05:00
#PJM -g gt00
#PJM -j
#----- Program execution -----#

module purge
module load intel
module load impi
module load waitio

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

mpiexec -n 2 ./test-intel

exit
```


プログラムの実行: PB数=2(a64fx+intel)

• バッチスクリプトの起動

```
[z30xxx@wisteria01 sample]$ pjsub test-a64fx-1.sh  
[INFO] PJM 0000 pjsub Job 636569 submitted.  
[z30xxx@wisteria01 PingPong]$ pjsub test-intel-2.sh  
[INFO] PJM 0000 pjsub Job 636570 submitted.  
[z30xxx@wisteria01 sample]$
```

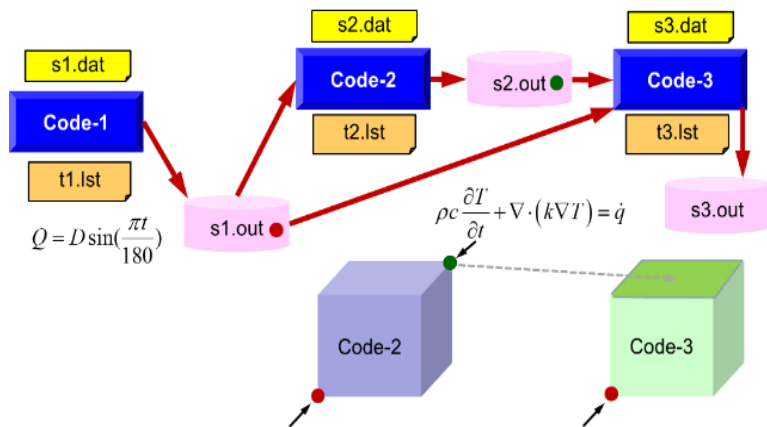
• 出力

```
[z30xxx@wisteria01 PingPong/]$ cat test1.636569.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: fjmp/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(111):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#=10.0.6.21:8000 nprocs=2, nGIO=0  
wo5575:0/2(111): Multiple WaitIO Connector now ready NPB=2!  
wo5575:0/2(111):PBserver: waitio_fetch_PBdata accepted from 10.0.6.21 port=13966 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 test1.636570.out  
waitio-server host wa21:6500 started  
wa21:1:0/2(94):PBclient:trying connect host 10.11.135.7,port 48155  
wa21:1/2(94):PBclient: connected to host 10.11.135.7,port 48155  
wa21:1/2(94):PBclient: upstream to WaitIO master Nproc=2 done!  
wa21:1:0 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#=10.0.6.21:8000 nprocs=2, nGIO=0  
wa21:1/2(94): Multiple WaitIO Connector now ready NPB=2!  
rank2: waitio_isend to rank3  
rank2: waitio_irecv from rank3  
rank3: waitio_irecv from rank2  
rank3: waitio_isend to rank2
```

WaitIO:異種アプリケーション間のデータ通信手法

- 既存の連成アプリケーションではファイルによるデータ連携が多い
 - ファイル共有によるデータ連携をWaitIO通信によるデータ連携に変換
- 有限要素法による三次元非定常熱伝導解析Toyプログラムによる実例
 - 3つのプログラムから構成される連成アプリケーション



	Code-1	Code-2	Code-3
概要	発熱量(点源)計算	FEMによる非定常三次元熱伝導方程式, MPI並列化	FEMによる非定常三次元熱伝導方程式, MPI並列化
入力	s1.dat	s2.dat s1.out	s3.dat s1.out, s2.out
出力	s1.out, t1.lst	s2.out, t2.lst	s3.out, t3.lst

Code-3: Code-2の出力を読み、 $Z=Z_{\max}$ の面に強制温度固定条件

WaitIO: Toyプログラムの書き換え

- コード例： Code-1のWaitIO修正: [src1/test1.f](#)

```
001  implicit REAL*8(A-H,O-Z)
002  real(kind=8) :: Interval
003  include 'mpif.h'

004  call MPI_Init(ierr)
005  open (11,file='s1.dat',status='unknown')
006  read (11,*) ITERmax, Period, Interval, Val
007  write (*,'(a,i8)') '##ITERmax ', ITERmax
008  write (*,'(a,1pe16.6)') '##Period ', Period
009  write (*,'(a,1pe16.6)') '##Interval', Interval
010  write (*,'(a,1pe16.6//)') '##Val ', Val
011  close (11)

012  open (12,file='s1.out',status='unknown')

013  pi = 4.d0*datan(1.d0)
014  coef= pi/180.d0
015  time= 0.d0
016  S0time= mpi_wtime ()
```

```
017  do
018    S1time= mpi_wtime ()

019    do
020      do iter= 1, ITERmax
021        a= 1.d0
022      enddo
023      E0time= mpi_wtime (ierr)
024      if ((E0time-S1time).gt.Interval) exit
025    enddo

026    ttt= E0time - S0time
027    Source= Val * dsin(ttt*coef)
028 !    rewind (12)
029    write (12,'(2(1pe16.6))') ttt, Source
030  enddo

031  call MPI_Finalize()
032  stop
033  end
```

Code-1: オリジナル

2024/10/30

WaitIO/MP

講習会

WaitIO: Toyプログラムの書き換え:2

- Open/Read-Writeを isend-irecv/waitに書き換え:[src1/test1_waitio.f](#)
 - MPI Non-blocking送受信関数と同様

Code-1: WaitIO修正

```
001 implicit REAL*8(A-H,O-Z)
002 real(kind=8) :: Interval
003 integer :: dest
004 integer(kind=4), dimension(:,), save, allocatable :: sta1
005 integer(kind=4), dimension(:,), save, allocatable :: req1

006 include 'mpif.h'
007 include 'waitio_mpf.h'

008 call MPI_Init(ierr)
009 open (11,file='s1.dat',status='unknown')
010 read (11,*) ITERmax, Period, Interval, Val
011 write (*,'(a,i8)') '##ITERmax ', ITERmax
012 write (*,'(a,1pe16.6)') '##Period ', Period
013 write (*,'(a,1pe16.6)') '##Interval', Interval
014 write (*,'(a,1pe16.6//)') '##Val ', Val
015 close (11)

016 allocate (sta1(WAITIO_STATUS_SIZE,2*2))
017 allocate (req1(WAITIO_REQUEST_SIZE,2*2))
018 call WAITIO_CREATE_UNIVERSE_PBHEAD (WAITIO_SOLVER_COMM, ierr)

019 open (12,file='s1.out',status='unknown')
```

```
037 write (*,'(2(1pe16.6))') ttt, Source
038 do dest=1, 2
039   call WAITIO_MPI_Isend (ttt, 1,
040    & WAITIO_MPI_DOUBLE_PRECISION,
041    & dest, 0, WAITIO_SOLVER_COMM, req1(1,2*(dest-1)+1), ierr)
042   if(ierr.ne.0) goto 100
043   call WAITIO_MPI_Isend (Source, 1,
044    & WAITIO_MPI_DOUBLE_PRECISION,
045    & dest, 0, WAITIO_SOLVER_COMM, req1(1,2*(dest-1)+2), ierr)
046   if(ierr.ne.0) goto 100
047 enddo
048 call WAITIO_MPI_Waitall (4, req1, sta1, ierr)
049 if(ierr.ne.0) goto 100
050 enddo
051 100 continue
```

WaitIO: Toyプログラムの書き換え:3

src2/test1_waitio.f

Code-2: WaitIO修正

```
007 integer :: dest
008 integer(kind=kint), dimension(:,:), save, allocatable :: sta1
009 integer(kind=kint), dimension(:,:), save, allocatable :: req1
```

```
010 allocate (sta1(WAITIO_STATUS_SIZE,1*6))
011 allocate (req1(WAITIO_REQUEST_SIZE,1*6))
```

!=== SNIP SNIP ===

```
053!c      read (12,*,end=100) ttt, Source
054      dest=0
055      call WAITIO_MPI_Irecv (ttt, 1,
056 &      WAITIO_MPI_DOUBLE_PRECISION,
057 &      dest, 0, WAITIO_SOLVER_COMM, req1(1,1), ierr)
058      call WAITIO_MPI_Irecv (Source, 1,
059 &      WAITIO_MPI_DOUBLE_PRECISION,
060 &      dest, 0, WAITIO_SOLVER_COMM, req1(1,2), ierr)

061      call WAITIO_MPI_Waitall (2, req1, sta1, ierr)
```

```
100      write (22,'(6(1pe16.6))') TIME, ttt, Source, X(ii1),
101 &      X(ii2), VAL
102      dest= 2
103      call WAITIO_MPI_Isend (TIME, 1,
104 &      WAITIO_MPI_DOUBLE_PRECISION,
105 &      dest, 0, WAITIO_SOLVER_COMM, req1(1,1), ierr)
106      call WAITIO_MPI_Isend (ttt, 1,
107 &      WAITIO_MPI_DOUBLE_PRECISION,
108 &      dest, 0, WAITIO_SOLVER_COMM, req1(1,2), ierr)
109      call WAITIO_MPI_Isend (Source, 1,
110 &      WAITIO_MPI_DOUBLE_PRECISION,
111 &      dest, 0, WAITIO_SOLVER_COMM, req1(1,3), ierr)
112      call WAITIO_MPI_Isend (X(ii1), 1,
113 &      WAITIO_MPI_DOUBLE_PRECISION,
114 &      dest, 0, WAITIO_SOLVER_COMM, req1(1,4), ierr)
115      call WAITIO_MPI_Isend (X(ii2), 1,
116 &      WAITIO_MPI_DOUBLE_PRECISION,
117 &      dest, 0, WAITIO_SOLVER_COMM, req1(1,5), ierr)
118      call WAITIO_MPI_Isend (VAL, 1,
119 &      WAITIO_MPI_DOUBLE_PRECISION,
120 &      dest, 0, WAITIO_SOLVER_COMM, req1(1,6), ierr)

121      call WAITIO_MPI_Waitall (6, req1, sta1, ierr)
```

WaitIO: Toyプログラムの書き換え:4

src3/test1_waitio.f

Code-3: WaitIO修正

```
007 integer :: dest
008 integer(kind=kint), dimension(:,:), save,allocatable :: sta1
009 integer(kind=kint), dimension(:,:), save,allocatable :: req1
```

```
010 allocate (sta1(WAITIO_STATUS_SIZE,1*6))
011 allocate (req1(WAITIO_REQUEST_SIZE,1*6))
```

!=== SNIP SNIP ===

054!C read (12,*,end=200) ttx, Source

```
055 dest=0
056 call WAITIO_MPI_Irecv (ttx, 1,
057 & WAITIO_MPI_DOUBLE_PRECISION,
058 & dest, 0, WAITIO_SOLVER_COMM, req1(1,1), ierr)
059 call WAITIO_MPI_Irecv (Source, 1,
060 & WAITIO_MPI_DOUBLE_PRECISION,
061 & dest, 0, WAITIO_SOLVER_COMM, req1(1,2), ierr)
```

```
062 call WAITIO_MPI_Waitall (2, req1, sta1, ierr)
063 if (ttx.ge.TIME) goto 200
```

071!C read (13,*,end=100) ttt, tt1, s0, s1, s2, TBOU

```
072 dest= 1
073 call WAITIO_MPI_Irecv (ttt, 1,
074 & WAITIO_MPI_DOUBLE_PRECISION,
075 & dest, 0, WAITIO_SOLVER_COMM, req1(1,1), ierr)
076 call WAITIO_MPI_Irecv (tt1, 1,
077 & WAITIO_MPI_DOUBLE_PRECISION,
078 & dest, 0, WAITIO_SOLVER_COMM, req1(1,2), ierr)
079 call WAITIO_MPI_Irecv (s0, 1,
080 & WAITIO_MPI_DOUBLE_PRECISION,
081 & dest, 0, WAITIO_SOLVER_COMM, req1(1,3), ierr)
082 call WAITIO_MPI_Irecv (s1, 1,
083 & WAITIO_MPI_DOUBLE_PRECISION,
084 & dest, 0, WAITIO_SOLVER_COMM, req1(1,4), ierr)
085 call WAITIO_MPI_Irecv (s2, 1,
086 & WAITIO_MPI_DOUBLE_PRECISION,
087 & dest, 0, WAITIO_SOLVER_COMM, req1(1,5), ierr)
088 call WAITIO_MPI_Irecv (TBOU, 1,
089 & WAITIO_MPI_DOUBLE_PRECISION,
090 & dest, 0, WAITIO_SOLVER_COMM, req1(1,6), ierr)
```

```
091 call WAITIO_MPI_Waitall (6, req1, sta1, ierr)
092 if (ttt.ge.TIME) goto 100
```

WaitIOサンプルプログラム実行: SYStest-waitio-socket

- サンプルプログラムの/work/xxx ディレクトリへの展開
 - `tar -zxf /work/share/waitio/src/examples/SYStest-waitio-socket.tgz`
- コンパイル
 - `module load intel impi`
 - `cd SYStest-waitio-socket; make`
- 実行
 - バッチスクリプトの修正 `cd run; edit s[123]-xxx.sh`
 - バッチ処理実行
 - `pjsub s1-xxx.sh; pjsub s2-yyy.sh; pjsub s3-zzz.sh`
 - 複数: `odyssey+odyssey+odyssey, odyssey+odyssey+aquarius`

プログラムのコンパイル

- サンプルプログラムのworkディレクトリへのコピーと展開

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

- コンパイル

```
[z30xxx@wisteria01 sample]$ cd SYSTest-waitio-socket/  
[z30xxx@wisteria01 SYSTest-waitio-socket]$ module purge  
[z30xxx@wisteria01 SYSTest-waitio-socket]$ module load intel impi  
[z30xxx@wisteria01 SYSTest-waitio-socket]$ make  
(cd src1 ; make )  
....  
make[1]: Leaving directory '/work/01/jh210xxxx/z30xxx/sample/SYSTest-waitio-socket/src3'  
[z30xxx@wisteria01 SYSTest-waitio-socket]$  
[z30xxx@wisteria01 SYSTest-waitio-socket]$ ls -t run/  
sol3-a64fx sol2 sol1 s3-intel.sh s1-a64fx.sh s2.dat OrgResults  
sol3-intel sol2-intel s3.out s2-intel.sh s3-a64fx.sh s3.sh  
sol3 sol1-a64fx s2.out s2-a64fx.sh s1.dat s2.sh  
sol2-a64fx sol1-intel s1.out s1-intel.sh s3.dat s1.sh  
[z30xxx@wisteria01 SYSTest-waitio-socket]$
```


連成プログラムの実行条件

- バッチスクリプトでの設定
 - 同一ユーザ名での実行： 違うユーザの実行は区別されます

```
#PJM -N "SYStest-waitio" # Job名を一致  
#PJM -L rscgrp=coupler-lec-o #リソースグループは coupler-xxx  
#PJM -g gt00          # group名も一致(最後のoaは無視される)
```

WaitIO: Toyプログラム向けバッチスクリプト

```
#!/bin/sh
# = s1-a64fx.sh===
#PJM -N "SYStest-waitio"
#PJM -L rscgrp=coupler-lec-o
#PJM -L node=1
#PJM --mpi proc=1
#PJM -L elapse=00:30:00
#PJM -g gt00
#PJM -j
#PJM -e err
```

```
module purge
module load fj
module load fmpi
module load waitio
```

```
hostname
export WAITIO_MASTER_HOST=`hostname`
export WAITIO_MASTER_PORT=7100
export WAITIO_PPID=0
export WAITIO_NPB=3
waitio-serv-a64fx -d -m $WAITIO_MASTER_HOST
```

```
mpiexec -n ${PJM_MPI_PROC} ./sol1-a64fx
```

```
#!/bin/sh
# = s2-a64fx.sh===
#PJM -N "SYStest-waitio"
#PJM -L rscgrp=coupler-lec-o
#PJM -L node=1
#PJM --mpi proc=32
#PJM -L elapse=00:30:00
#PJM -g gt00
#PJM -j
#PJM -e err
```

```
module purge
module load fj
module load fmpi
module load waitio
```

```
hostname
export WAITIO_MASTER_HOST=`waitio-serv-a64fx -c`
export WAITIO_MASTER_PORT=7100
export WAITIO_PPID=1
export WAITIO_NPB=3
```

```
mpiexec -n ${PJM_MPI_PROC} ./sol2-a64fx
```

```
#!/bin/sh
# = s3-a64fx.sh===
#PJM -N "SYStest-waitio"
#PJM -L rscgrp=coupler-lec-o
#PJM -L node=1
#PJM --mpi proc=32
#PJM -L elapse=00:30:00
#PJM -g gt00
#PJM -j
#PJM -e err
```

```
module purge
module load fj
module load fmpi
module load waitio
```

```
hostname
export WAITIO_MASTER_HOST=`waitio-serv-a64fx -c`
export WAITIO_MASTER_PORT=7100
export WAITIO_PPID=2
export WAITIO_NPB=3
```

```
mpiexec -n ${PJM_MPI_PROC} ./sol3-a64fx
rm wk.*
```

Code-1 for a64fx

Code-2 for a64fx

Code-3 for a64fx

WaitIO: Toyプログラム向けバッチスクリプト 2

```
#!/bin/sh
# = s1-intel.sh===
#PJM -N "SYStest-waitio"
#PJM -L rscgrp=coupler-lec-a
#PJM -L node=1
#PJM --mpi proc=1
#PJM -L elapse=00:30:00
#PJM -g gt00
#PJM -j
#PJM -e err
```

```
module purge
module load intel
module load impi
module load waitio
```

```
export WAITIO_MASTER_HOST=`hostname`-ib0
export WAITIO_MASTER_PORT=7100
export WAITIO_PPID=0
export WAITIO_NPB=3
```

```
hostname
waitio-serv -d -m ${WAITIO_MASTER_HOST}
mpiexec -n ${PJM_MPI_PROC} ./sol1-intel
```

Code-1 for intel

```
#!/bin/sh
# = s2-intel.sh===
#PJM -N "SYStest-waitio"
#PJM -L rscgrp=coupler-lec-a
#PJM -L node=1
#PJM --mpi proc=16
#PJM -L elapse=00:30:00
#PJM -g gt00
#PJM -j
#PJM -e err
```

```
module purge
module load intel
module load impi
module load waitio
```

```
export WAITIO_MASTER_HOST=`waitio-serv -c`
export WAITIO_MASTER_PORT=7100
export WAITIO_PPID=1
export WAITIO_NPB=3
```

```
hostname
mpiexec -n ${PJM_MPI_PROC} ./sol2-intel
```

Code-2 for intel

```
#!/bin/sh
# = s3-intel.sh===
#PJM -N "SYStest-waitio"
#PJM -L rscgrp=coupler-lec-a
#PJM -L node=1
#PJM --mpi proc=16
#PJM -L elapse=00:30:00
#PJM -g gt00
#PJM -j
#PJM -e err
```

```
module purge
module load intel
module load impi
module load waitio
```

```
export WAITIO_MASTER_HOST=`waitio-serv -c`
export WAITIO_MASTER_PORT=7100
export WAITIO_PPID=2
export WAITIO_NPB=3
```

```
hostname
mpiexec -n ${PJM_MPI_PROC} ./sol3-intel
rm wk.*
```

Code-3 for intel

WaitIOサンプルプログラム実行: SYStest-waitio-socket

- 実行

- バッチスクリプトの修正
- バッチ処理実行: 3つのスクリプトを選択実行
 - `pjsub s1-xxx.sh; pjsub s2-yyy.sh; pjsub s3-zzz.sh`
 - 複数: odyssey+odyssey+odyssey, odyssey+odyssey+aquarius
- Aquariusはノード数が少ないので複数ジョブ実行が制限
- Odysseyの方がノード数に余裕があります

```
• cd run
• pjsub s1-a64fx.sh; pjsub s2-a64fx.sh; pjsub s3-a64fx.sh
or
• pjsub s1-a64fx.sh; pjsub s2-a64fx.sh; pjsub s3-intel.sh
or
• pjsub s1-a64fx.sh; pjsub s2-intel.sh; pjsub s3-a64fx.sh
```

Watch pjstatの結果

```
z30455@wisteria01:~/workj/examples/PingPong
File Edit View Search Terminal Help
Every 2.0s: pjstat wisteria01: Sun Oct 9 15:38:55 2022

Wisteria/BDEC-01 scheduled stop time: 2022/10/27(Thu) 13:00:00 (Remain: 17days 21:21:05)

JOB_ID      JOB_NAME    STATUS  PROJECT  RSCGROUP    START_DATE    ELAPSE      TOKEN      NODE  GPU
659543      SYStest-wa  RUNNING gt00      coupler-lec-o 10/09 15:36:09< 00:02:47    -        1    -
659544      SYStest-wa  RUNNING gt00      coupler-lec-o 10/09 15:36:09< 00:02:47    -        1    -
659545      SYStest-wa  RUNNING gt00      coupler-lec-o 10/09 15:36:09< 00:02:47    -        1    -
```

s1-a64fx.sh+s2-a64fx.sh+s3-a64fx.sh

```
z30455@wisteria01:~/workj/examples/PingPong
File Edit View Search Terminal Help
Every 2.0s: pjstat wisteria01: Sun Oct 9 15:40:00 2022

Wisteria/BDEC-01 scheduled stop time: 2022/10/27(Thu) 13:00:00 (Remain: 17days 21:20:00)

JOB_ID      JOB_NAME    STATUS  PROJECT  RSCGROUP    START_DATE    ELAPSE      TOKEN      NODE  GPU
659549      SYStest-wa  HOLD    gt00      coupler-lec-o --/-- --:--:-- 00:00:00    -        1    -
659550      SYStest-wa  HOLD    gt00      coupler-lec-o --/-- --:--:-- 00:00:00    -        1    -
659551      SYStest-wa  HOLD    gt00      coupler-lec-a --/-- --:--:-- 00:00:00    -        1    8
```

s1-a64fx.sh+s2-a64fx.sh+s3-intel.sh

s3.outの出力

#	時間	s2.out時間	●発熱量	●結果	●の隣結果	●結果	●結果
	1.000000E-01	1.000000E-01	1.745347E-03	5.602511E-02	3.225460E-03	0.000000E+00	0.000000E+00
	2.000000E-01	2.000000E-01	1.663485E-02	5.372070E-01	3.184216E-02	0.000000E+00	0.000000E+00
	3.000000E-01	3.000000E-01	1.838060E-02	6.213567E-01	4.488534E-02	1.749074E-64	1.749074E-64
	4.000000E-01	4.000000E-01	2.012594E-02	6.853573E-01	5.309708E-02	1.931968E-35	1.931968E-35
	5.000000E-01	5.000000E-01	2.187122E-02	7.472359E-01	5.987997E-02	7.930161E-32	7.930161E-32
<Snip><Snip><Snip>							
	2.920000E+01	2.920000E+01	4.993690E-01	1.739279E+01	1.646121E+00	9.098841E-05	9.098841E-05
	2.930000E+01	2.930000E+01	5.008806E-01	1.744550E+01	1.651154E+00	9.284215E-05	9.284215E-05
	2.940000E+01	2.940000E+01	5.023908E-01	1.749815E+01	1.656181E+00	9.472301E-05	9.472301E-05
	2.950000E+01	2.950000E+01	5.038994E-01	1.755075E+01	1.661204E+00	9.663123E-05	9.663123E-05
	2.960000E+01	2.960000E+01	5.054064E-01	1.760330E+01	1.666221E+00	9.856701E-05	9.856701E-05
	2.970000E+01	2.970000E+01	5.069119E-01	1.765580E+01	1.671233E+00	1.005306E-04	1.005306E-04
	2.980000E+01	2.980000E+01	5.084159E-01	1.770824E+01	1.676241E+00	1.025222E-04	1.025222E-04
	2.990000E+01	2.990000E+01	5.099183E-01	1.776062E+01	1.681243E+00	1.045420E-04	1.045420E-04
	3.000000E+01	3.000000E+01	5.114191E-01	1.781295E+01	1.686240E+00	1.065902E-04	1.065902E-04

h3-Open-SYS/WaitIOと h3-Open-UTIL/MPを使ってみよう

15:50 – 17:30

Wisteria/BDEC-01システム利用イントロダクション:20分
サンプルプログラムの実行WaitIO:20分
サンプルプログラムの実行MP: 50分
自由に動かしてみよう・Q&A: 10分

サンプルプログラムの実行 UTIL/MP

WaitIOサンプルプログラム実行概要

- サンプルプログラムの実行前準備
 1. ポータルサイトへのssh公開鍵登録（事前送付済）
 2. Wisteriaへのログイン（事前送付済）
 3. Linux操作の基本（ディレクトリ変更、エディター、適宜）
- サンプルプログラム実行
 1. サンプルプログラムの実行ディレクトリへの展開
 2. コンパイル
 3. バッチスクリプト準備・修正
 4. バッチ処理実行

UTIL/MPサンプルプログラム実行:Odyssey - Aquarius間通信:Fortran-C, Fortran-Python

• サンプルプログラムの概要

- Fortranのサンプルプログラム2コンポーネントとC言語のサンプルプログラムをOdysseyとAquariusで実行する
- Fortranのサンプルプログラム 2 コンポーネントとPythonのサンプルスクリプトをOdysseyとAquariusで実行する

UTIL/MPサンプルプログラム実行:Odyssey -Aquarius連成:Fortran-Fortran

- サンプルプログラムのworkディレクトリへのコピー

```
[z30xxx@wisteria01 sample]$ cp -r /work/share/h3-Open-UTIL-MP/sample_simple ./  
[z30xxx@wisteria01 sample]$
```

- コンパイル

```
[z30xxx@wisteria01 sample]$ cd sample_simple  
[z30xxx@wisteria01 src]$ export SYSTEM=odyssey  
[z30xxx@wisteria01 src]$ module load odyssey  
[z30xxx@wisteria01 src]$ module load waitio  
[z30xxx@wisteria01 src]$ make clean  
[z30xxx@wisteria01 src]$ make odyssey  
[z30xxx@wisteria01 src]$ export SYSTEM=aquarius  
[z30xxx@wisteria01 src]$ module purge  
[z30xxx@wisteria01 src]$ module load intel impi  
[z30xxx@wisteria01 src]$ module load waitio  
[z30xxx@wisteria01 src]$ module unload intel impi  
[z30xxx@wisteria01 src]$ module load gcc omp  
[z30xxx@wisteria01 src]$ make clean  
[z30xxx@wisteria01 src]$ make aquarius
```

プログラムの実行

・バッチスクリプトの修正: goA.sh, goB.sh, goC.sh

```
#!/bin/bash
#PJM -L rscgrp=coupler-lec-o
#PJM -L node=1x1x1:torus
#PJM --mpi proc=1
#PJM -L elapse=00:20:00
#PJM -g jh210022o
#PJM -N waitio_test
```

```
module load fj
module load fjmp
module load waitio
```

```
export COUPLER_JOB_ID=1
export WAITIO_HYBRID_THRESHOLD=524288
export WAITIO_SCRATCH_DIR="."
```

```
rm -rf ${WAITIO_SCRATCH_DIR}/.waitio-file-${COUPLER_JOB_ID}/
###export WAITIO_COMM_TYPE=hybrid
export WAITIO_COMM_TYPE=file
###export WAITIO_COMM_TYPE=socket
export WAITIO_MAXFILE_SIZE_PER_PROC=8048576
```

```
export WAITIO_MASTER_HOST=`hostname`
export WAITIO_MASTER_PORT=7100
export WAITIO_PPID=0
export WAITIO_NPB=3
```

```
/work/share/waitio/bin/waitio-serv-a64fx -d -m $WAITIO_MASTER_HOST
```

```
mpiexec -np 1 ./compA.odyssey
```

```
#!/bin/bash
#PJM -L rscgrp=coupler-lec-a
#PJM -L node=1
#PJM --mpi proc=1
#PJM -L elapse=00:20:00
#PJM -g jh210022a
#PJM -N waitio_test
```

```
module purge
module load intel impi
module load waitio
```

```
export COUPLER_JOB_ID=1
export WAITIO_HYBRID_THRESHOLD=524288
export WAITIO_SCRATCH_DIR="."
```

```
###export WAITIO_COMM_TYPE=hybrid
export WAITIO_COMM_TYPE=file
###export WAITIO_COMM_TYPE=socket
export WAITIO_MAXFILE_SIZE_PER_PROC=8048576
```

```
export WAITIO_MASTER_HOST=`/work/share/waitio/bin/waitio-serv -c`
```

```
export WAITIO_MASTER_PORT=7100
export WAITIO_PPID=2
export WAITIO_NPB=3
```

```
module unload intel impi
module load gcc omp
```

```
mpiexec -np 1 ./compC.aquarius
```

プログラムの実行

- ジョブ投入

```
[z30xxx@wisteria01 run]$ pjsub goA.sh  
[INFO] PJM 0000 pjsub Job 665211 submitted.  
[z30xxx@wisteria01 run]$ pjsub goB.sh  
[INFO] PJM 0000 pjsub Job 665214 submitted.  
[[z30xxx@wisteria01 run]$ pjsub goC.sh  
[INFO] PJM 0000 pjsub Job 665215 submitted.
```

プログラムの実行完了

- pjstatコマンドでジョブ完了確認後
- 出力ファイル

```
[z****@wisteria02 sample_simple]$ ls
compA.f90  compC.aquarius  goC.sh          TM.compA          waitio_test.5108105.out  waitio_test.5108707.err
compA.lst  compC.c          goD.sh          TM.compA.pe00000  waitio_test.5108106.err  waitio_test.5108707.out
compA.odyssey  compC.o          go.sh           TM.compB          waitio_test.5108106.out  waitio_test.5108708.err
compB.f90  compD.py         h3ou.compA.log.PE00000  TM.compB.pe00000  waitio_test.5108107.err  waitio_test.5108708.out
compB.lst  coupling.conf    h3ou.compB.log.PE00000  TM.compC          waitio_test.5108107.out
compB.odyssey  goA.sh          h3ou.compC.log.PE00000  TM.compC.pe00000  waitio_test.5108706.err
compC      goB.sh          Makefile         waitio_test.5108105.err  waitio_test.5108706.out
```

h3-Open-SYS/WaitIOと h3-Open-UTIL/MPを使ってみよう

15:50 – 17:30

Wisteria/BDEC-01システム利用イントロダクション:20分
サンプルプログラムの実行WaitIO:30分
サンプルプログラムの実行MP: 50分
自由に動かしてみよう・Q&A: 10分

自由に動かしてみよう・Q&A: 10分

- 質問などあれば口頭・zoom chat, slack他でお知らせ下さい
- アカウントは1ヶ月有効です。
- 資料のPDF版はWEBページに掲載します。
 - <https://www.cc.u-tokyo.ac.jp/events/lectures/236/>
 - アンケートへの協力をお願いします。

ご清聴ありがとうございました。

第236回 お試しアカウント付き
並列プログラミング講習会
「異種システム間連成アプリケーション
開発を学ぶ:WaitIO/MP」