h3-Open-SYS/WaitIO: 異種システム間通信を実現する通信ライブラリの概要

住元 真司(東大), 荒川 隆(CliMTech Inc.), 坂口 吉生(富士通),
松葉 浩也(日立), 八代 尚(環境研), 塙 敏博(東大), 中島 研吾(東大/理研CCS)

2023/03/30

FUJITSU

国立研究開発法人 国立環境研究所
National Institute for Environmental Studies

東京大学
THE UNIVERSITY OF TOKYO

東京大学情報基盤センター
INFORMATION TECHNOLOGY CENTER, THE UNIVERSITY OF TOKYO

RIKEN

R-CCS

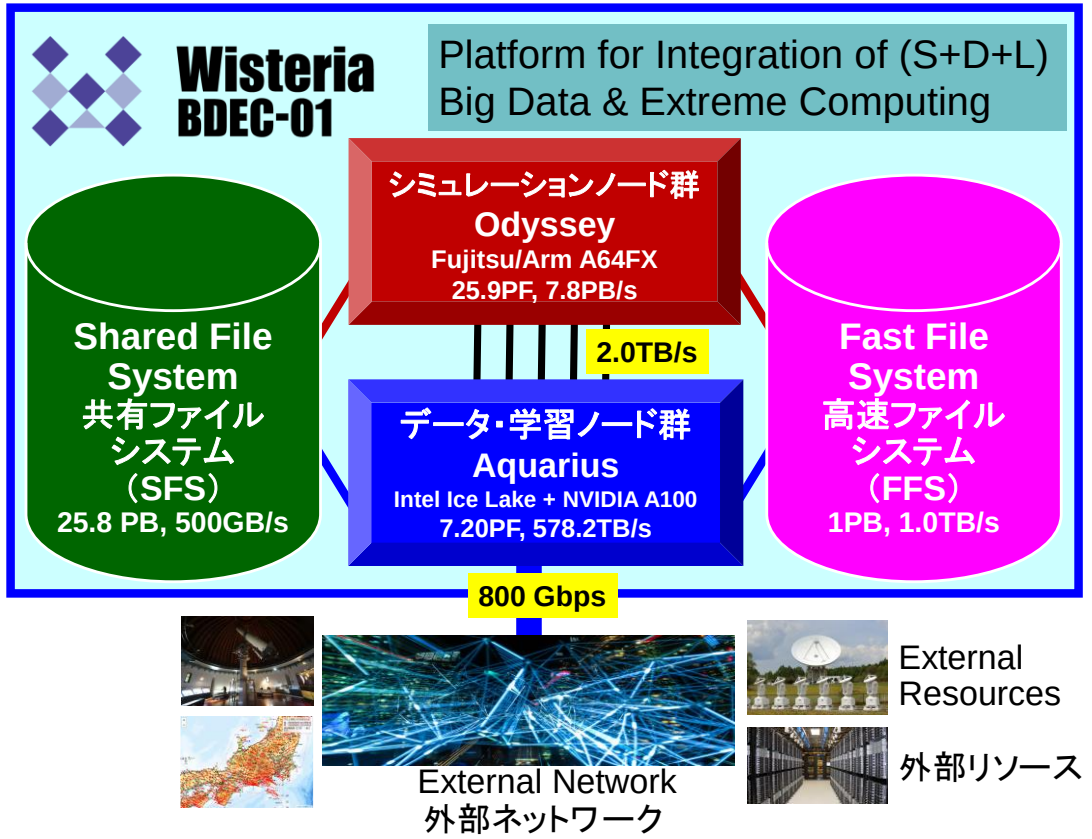
発表の概要

- Wisteria/BDEC-01
- 異種システム間連成計算
  - 意義
  - 構成要素: h3-Open-SYS/WaitIO, ジョブ実行環境
- Wisteria/BDEC-01システムでの異種システム間連成計算実行環境
  - h3-Open-SYS/WaitIOによる通信性能評価・連成実行
  - h3-Open-UTIL/MPによるアプリケーション評価
- おわりに

Wisteria/BDEC-01

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）

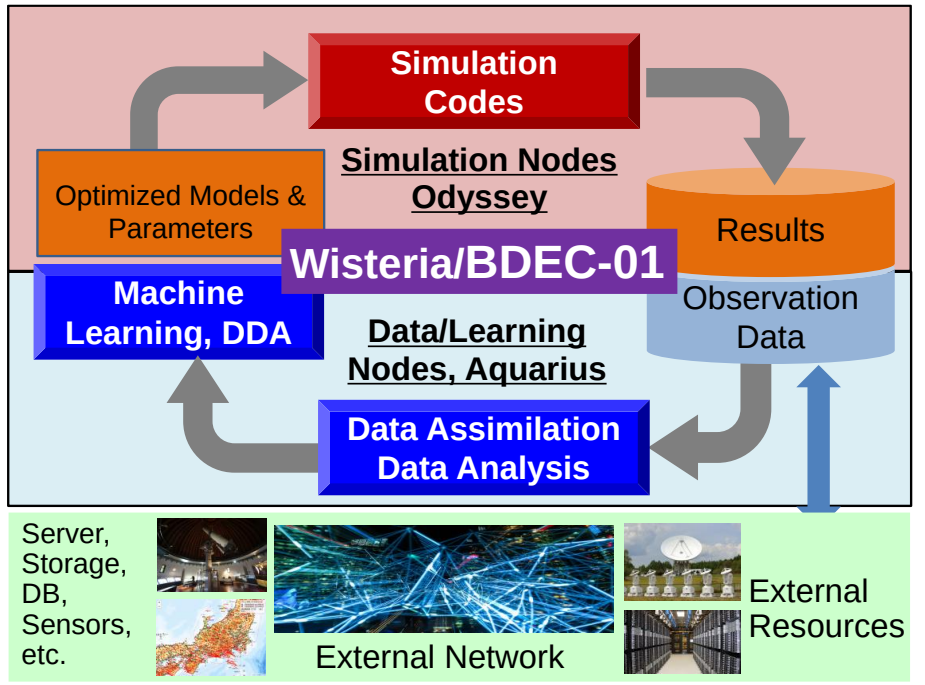
- 2021年5月14日運用開始
  - 東京大学柏Ⅱキャンパス
- 33.1 PF, 8.38 PB/sec.，富士通製
  - ~4.5 MVA（空調込み），~360m^2
- Hierarchical, Hybrid, Heterogeneous (h3)
- 2種類のノード群
  - **シミュレーションノード群 (S, SIM) : Odyssey**
    - 従来のスパコン
    - Fujitsu PRIMEHPC FX1000 (A64FX), 25.9 PF
      - 7,680ノード（368,640 コア），20ラック，Tofu-D
  - **データ・学習ノード群（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**



Wisteria BDEC-01

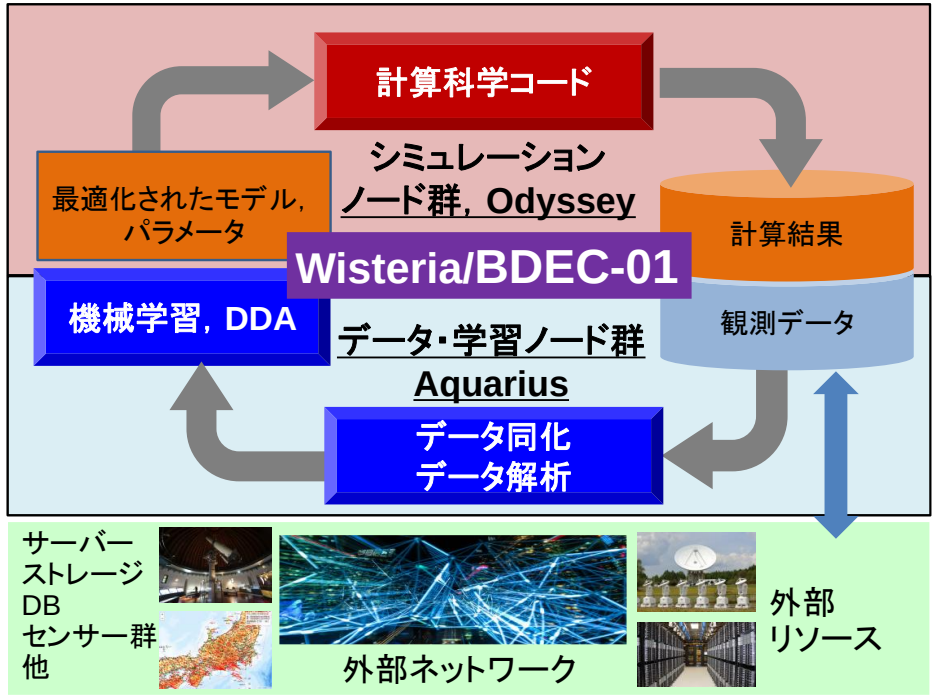
異種システム間連成計算の意義

- 並列計算機の性能向上：弱連成計算による解析広まる
  - CAEアプリケーションや気象分野のアプリケーション(例、NICAM)
  - 多くの場合単一システム内でファイルでのデータ連携で解析実施
  - アプリケーション高速化に限界：要求する計算機の特性が異なる
- 異種システム連携高速化：h3-Open-BDECの研究開発
  - アプリケーション毎に最適なシステムで解き、相互にデータ連携
- 異種システム間で連成計算を実現には以下の2つが必要
  - 異種システム上でのアプリケーション実行環境
  - バッチ処理システム間ジョブ実行連携機構

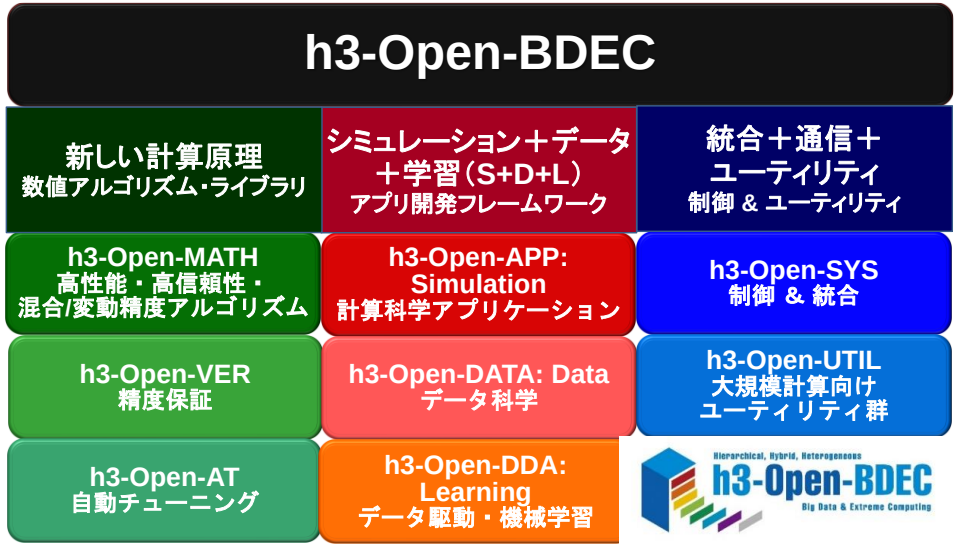


h3-Open-SYS/WaitIO

データ受け渡しライブラリ〔松葉，2020〕
〔住元他，HPC-181(2021), HPC-185,187(2022)〕

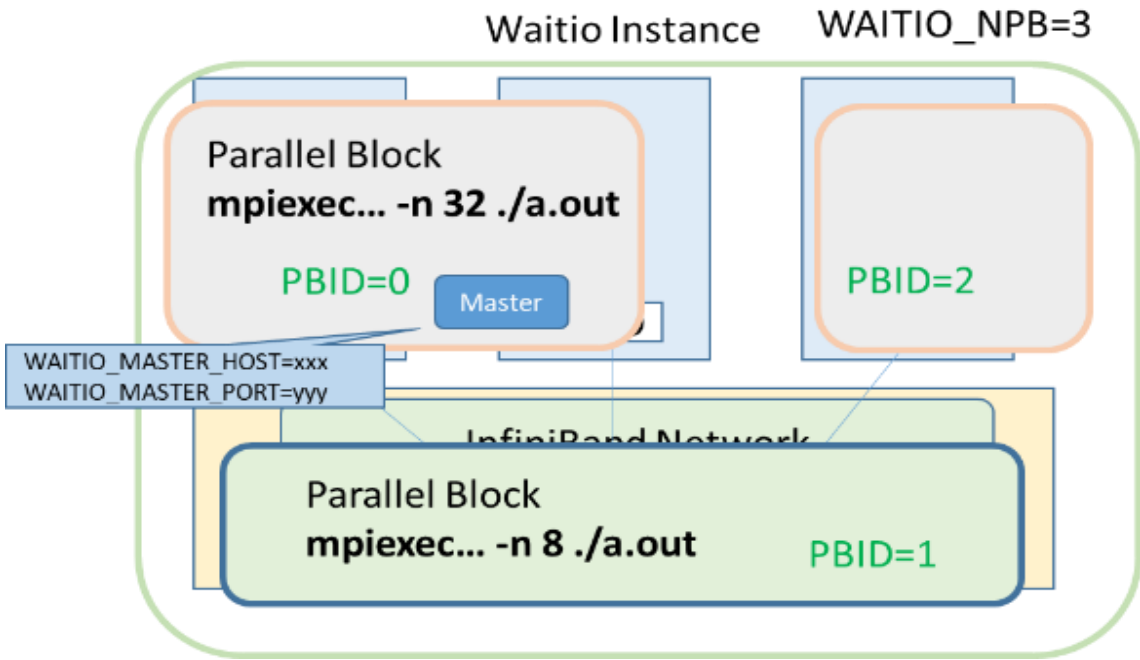


- ヘテロジニアス環境下での異なるコンポーネント間ファイル経由連携ライブラリとして考案
- 機能
  - ✓ Odyssey～Aquarius間連携
    - ☐ IB-EDR経由通信（WaitIO-Socket）
    - ☐ ファイル経由（WaitIO-File）
  - ✓ 外部からのデータ取得（観測データ等）
  - ✓ 読み込み・書き出しの同期
- API：C/C++，Fortranから呼び出し可能
  - ✓ MPIライクなインタフェースを提供



WaitIOの動作イメージ

- WaitIO Instance
  - WaitIOシステム全体を示す
- Parallel Block: MPIアプリケーション
  - アプリケーション毎のプロセスグループ
  - 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

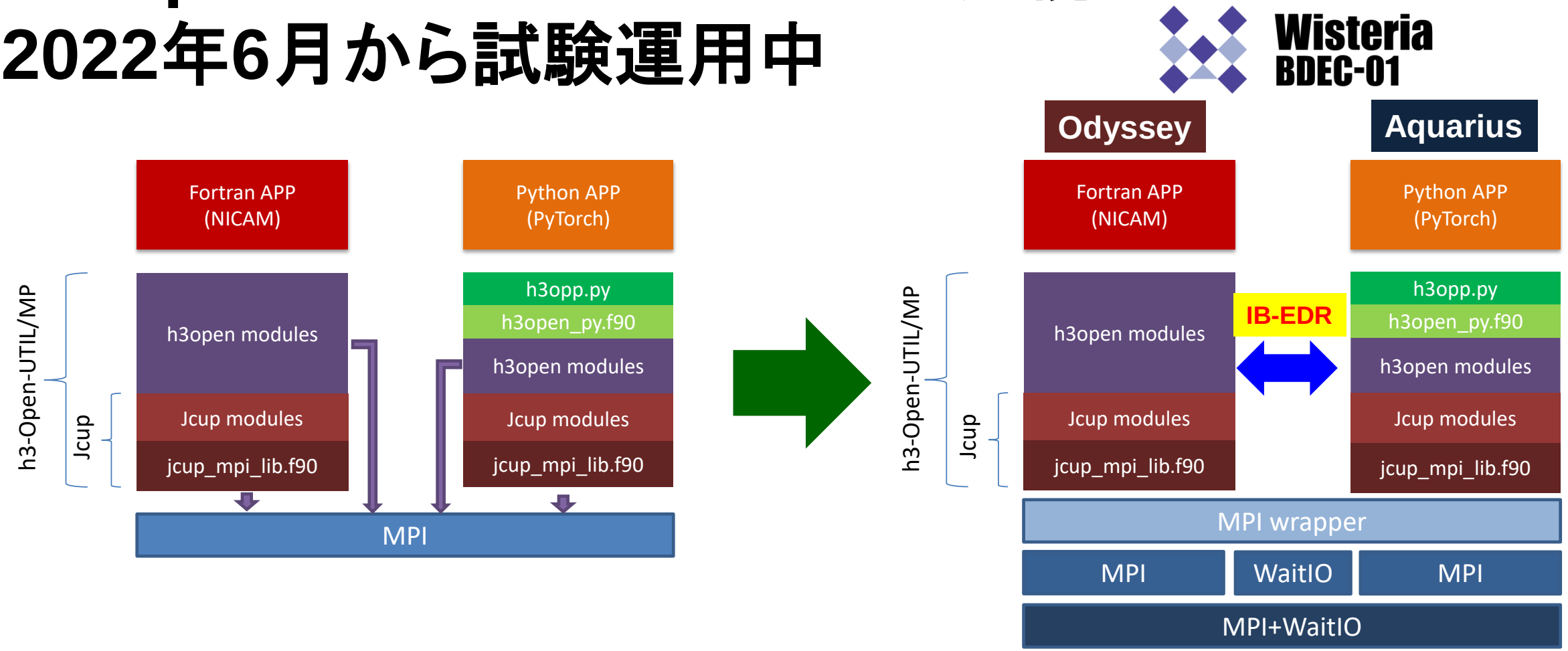


WaitIO API

- 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

WatiIO 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_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

h3-Open-UTIL/MP・h3-Open-SYS/WaitIO-Socket連携 2022年6月から試験運用中



2021年4月：MPI通信可能な環境を前提

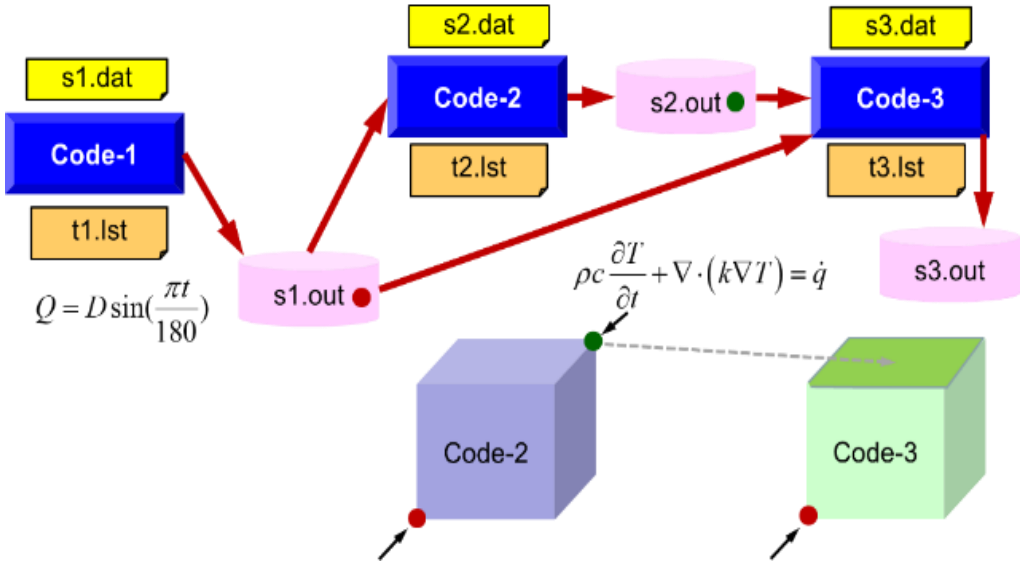
2022年6月：Coupler＋WaitIO

WaitIOの活用方法

- 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
- 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
- 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(後述)

WaitIO:異種アプリケーション間のデータ通信手法

- 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=Zmax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

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

```
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_mpif.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')
```

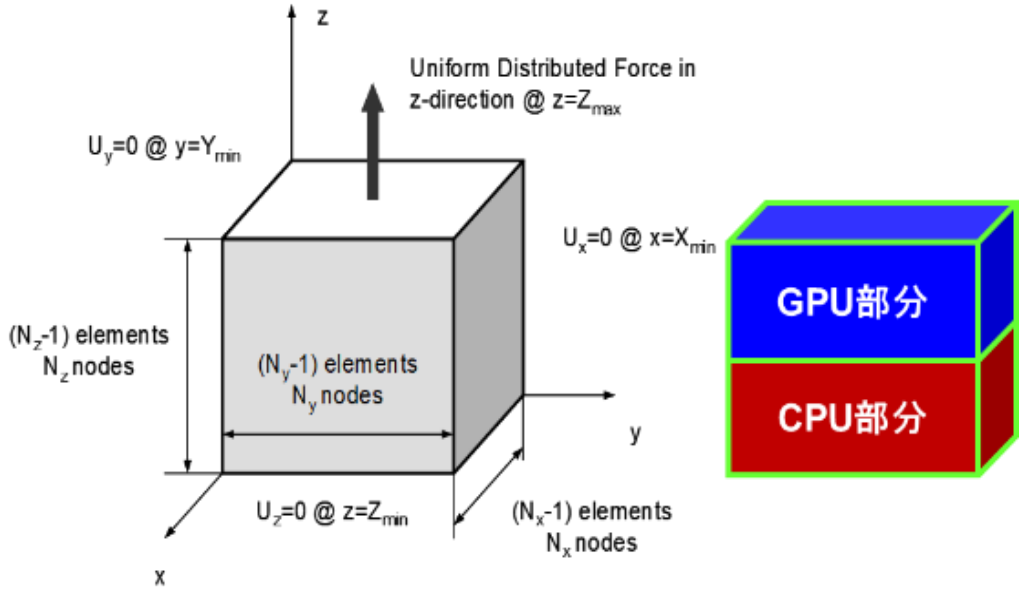
Code-1: WaitIO修正

```
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:既存アプリケーションの協調実行

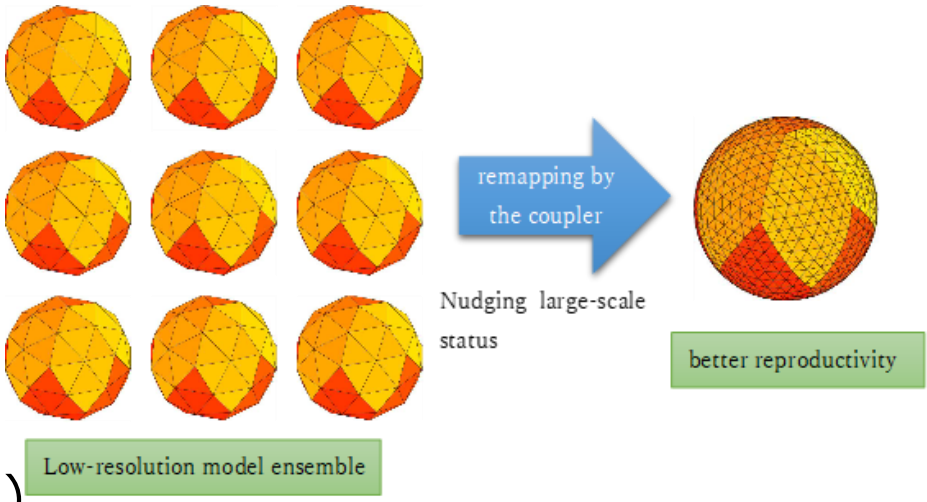
- 適したシステムで適した処理を協調実行
  - 複数システムを融合、より大きな問題を解く
  - システム毎の欠点を補完したアプリケーション実行
  - 特性の異なる場合(例：線形問題と非線形問題)が混在するコードを適したシステムで分散実行

- 応用例
  - GeoFEM〔18,19〕/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

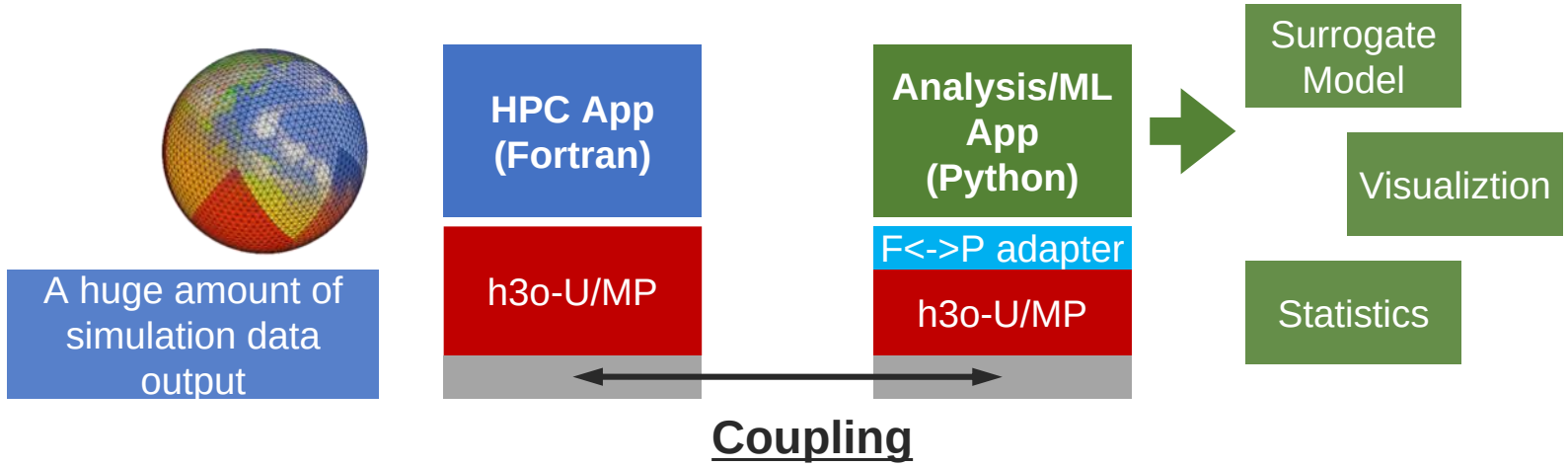


高機能カップラー:h3-Open-UTIL/MP

- 基本機能：連成計算機能提供
  - 時空間スケールが異なる複数モデル間で時間・空間差違の補整後データ交換
  - 複数の並列アプリケーション間でプロセス群間のデータ交換
- 拡張機能：
  - アンサンブル連成機能：複数連成計算を並列同時実行、かつ、連成計算同士の統計処理を行いながら実行
  - Python API 機能：Pythonアプリケーションを連成実行(PyTorch)
  - 異システム間での連成計算実行機能：h3-Open-SYS/WaitIO協調



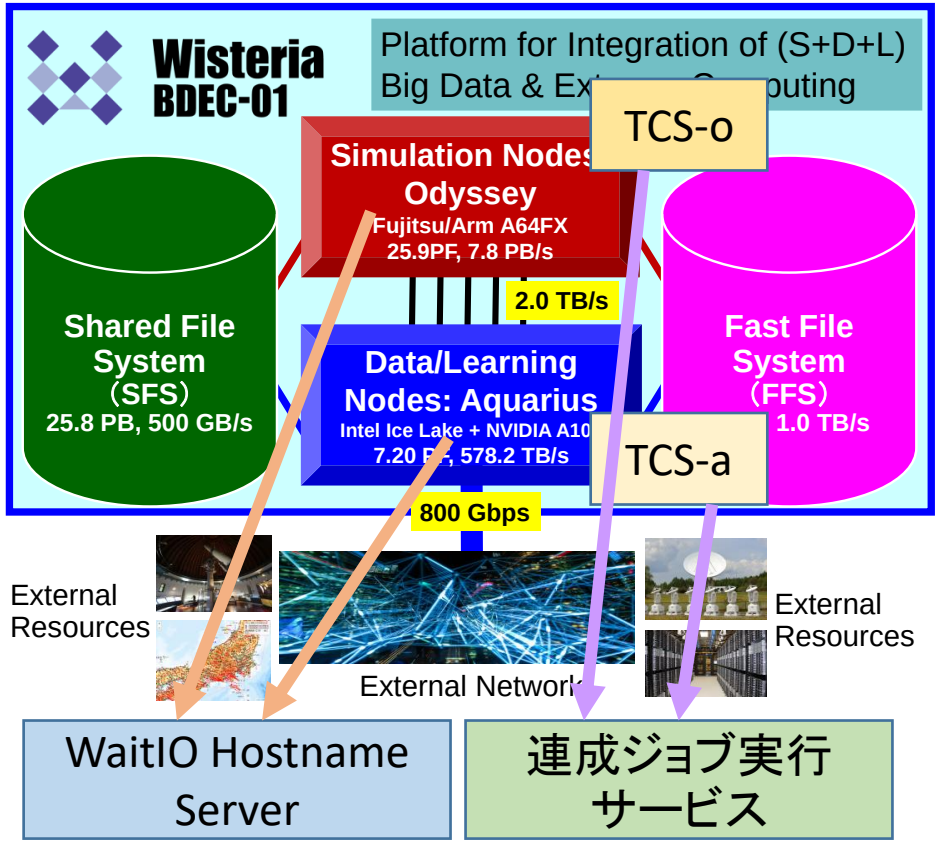
低解像度イメージを用いた多耐１アンサンブル連成



異種バッチ処理システム間ジョブ実行連携機構

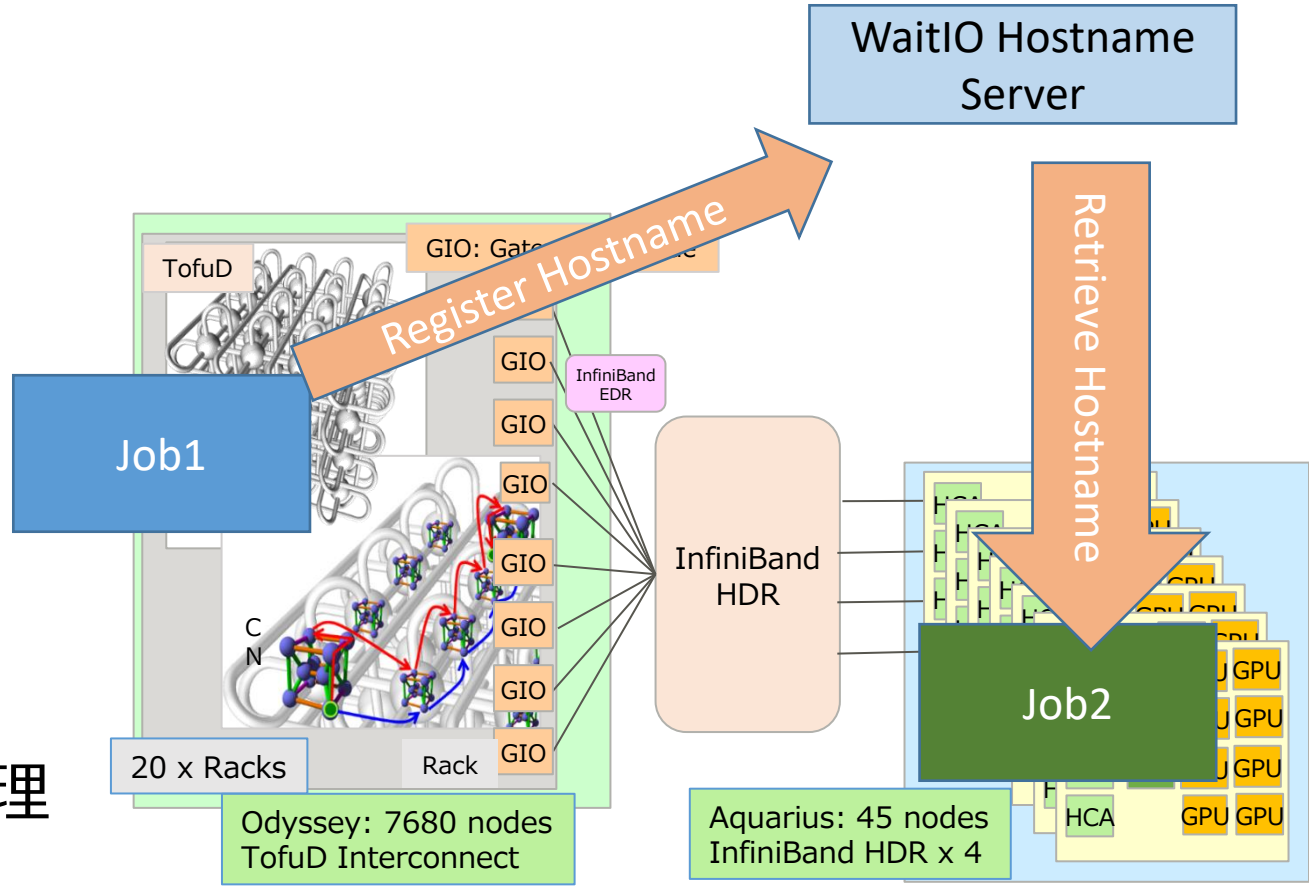
- OdysseyとAquarius上での連成ジョブ実行
  - 通常実行ジョブと連成実行ジョブの区別
  - 複数システムのリソース空き時に同時実行
  - WaitIO-Socket実行環境設定

- 現実行環境TCSでは要件未達：
  - WaitIO Hostname Server
  - 連成ジョブ実行サービス



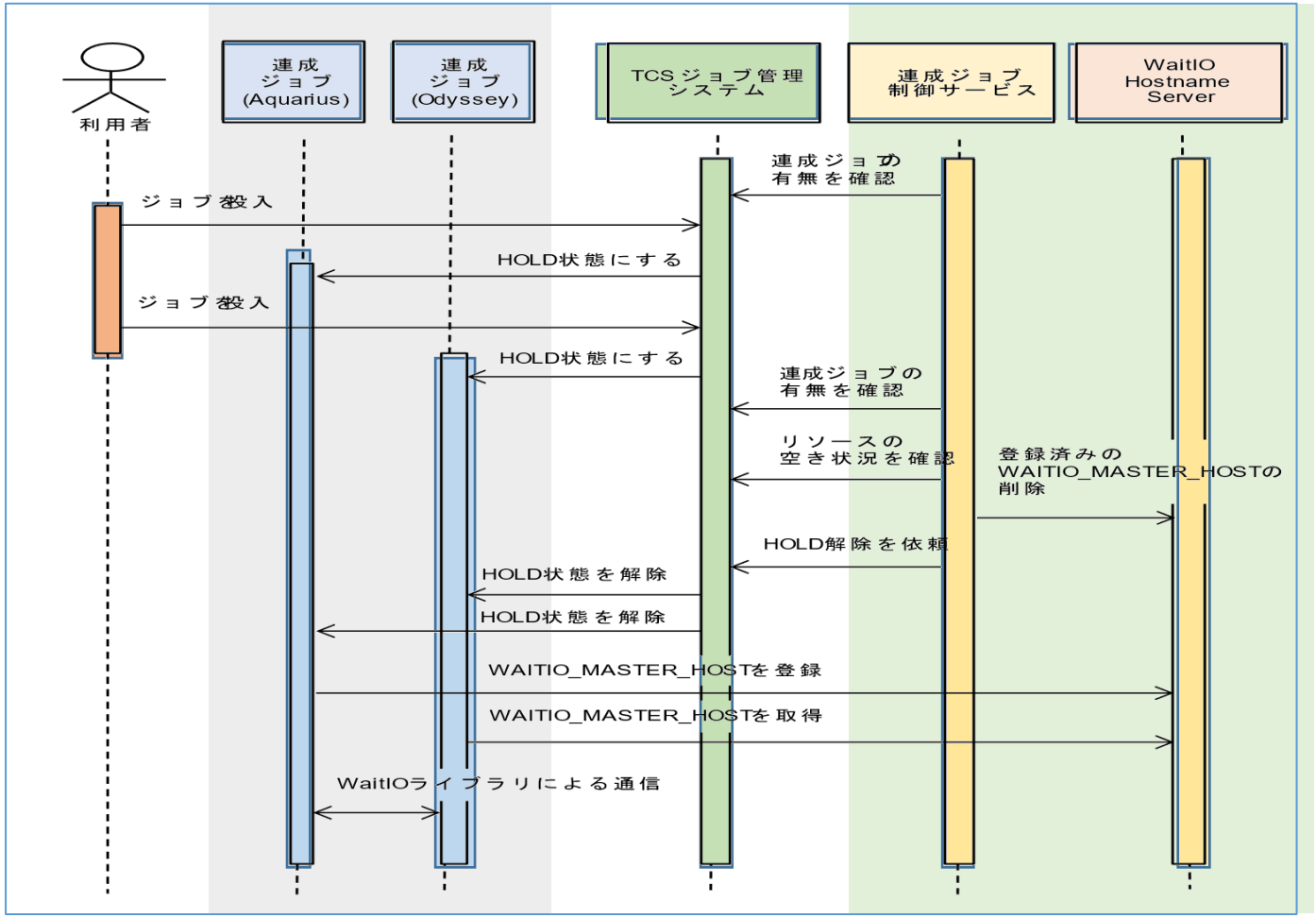
WaitIO Hostname Server

- 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
- 固定ホスト上で実行、特定のSocket Portに接続して各種設定
  - ジョブ実行ホスト名の設定
  - 設定されたホスト名の獲得
  - ホスト名リセットを実施
- フロントエンドサーバなど固定ホスト上実行を想定
  - USER名＋ジョブ名でカップラージョブ群管理 同時に複数ジョブ群を起動可能
  - カップラージョブ毎COUPER_JOB_ID割当



連成ジョブ実行サービスの動作シーケンス

- 連成ジョブ判断：
  - 同一ジョブ名、ユーザ名、グループ名、連成ジョブ用リソースグループ
- 同時実行方法：
  - ジョブ投入時にHOLD状態遷移
  - 同時リソース空き検出時にHOLD処理を解除
  - 現状、WaitIO Hostname Server制限のため3分に1度連成ジョブ起動



Wisteria/BDEC-01システムでのWaitIO実行

- 通信性能
- 異種システム上でのアプリケーション性能：
  - h3-Open-SYS/WaitIO+h3-Open-UTIL/MP
- WaitIO-Socketを用いたアプリケーション実行イメージ

評価環境

	Wisteria/Odyssey	Wisteria/Aquarius
総理論演算性能	25.9 PFLOPS	7.2 PFLOPS
総ノード数	7,680	45
インターコネクト	Tofu-D (6次元メッシュ/トーラス)	InfiniBand HDR (200Gbps) x 4HCA (Full Fat Tree)
プロセッサ/ノード	A64FX(Armv8.1+SVE), 2.2GHz, 1 socket (48 Core+2 or 4アシスタントコア)	Intel Xeon Platinum 8360Y, 2.4GHz 2 sockets(36+36)
メモリ量/ノード	32 GB	512GiB
メモリバンド幅/ノード	1024GB/s	409.6GB/s
GPU/ノード	-	NVIDIA A100 x8
コンパイラ、MPI	富士通コンパイラ、富士通MPI	Intelコンパイラ、インテルMPI、(Open MPI)
オペレーティングシステム	Red Hat Enterprise Linux 8	Red Hat Enterprise Linux 8

	Oakbridge-CX (OBCX)	Oakforest-PACS (OFP)
総理論演算性能	6.61 PFLOPS	25 PFLOPS
総ノード数	1368	8208
インターコネクト	Omni-Path (100Gbps)	Omni-Path (100Gbps)
プロセッサ/ノード	Intel Xeon Platinum 8280 2.7GHz, 2 socket (28+28)	Intel Xeon Phi 7250 1.4GHz 1 socket (68)
メモリ量/ノード	192 GB	96(DDR4)+16(MCDRAM) GB
メモリバンド幅/ノード	281.6GB/s	115(DDR4)+490(MCDRAM) GB/s
コンパイラ、MPI	Intelコンパイラ、インテルMPI、(Open MPI)	Intelコンパイラ、インテルMPI、(Open MPI)
オペレーティングシステム	Red Hat Enterprise Linux 7, CentOS 7	Red Hat Enterprise Linux 7, CentOS 7

PingPong 1/2 RTT評価

可変 proc- 16 Node固定

usec(8B)	1/2 RTT(ノード内)	1/2 RTT(ノード間)
Odyssey	26.8	37.1 (Tofu-D)
Aquarius	6.57	21.1 (IB)
OBCX	6.85	14.5 (OPA)
OFP	49.7	83.7 (OPA)

good

bad

- ノード内、ノード間ともに CPU性能が大きく性能に影響
  - Xeon搭載(Aquarius, OBCX) は6usec台
  - A64FX、Xeon Phi搭載システムはCPU性能に比例

PingPong バンド幅評価

可変 proc- 16 Node固定

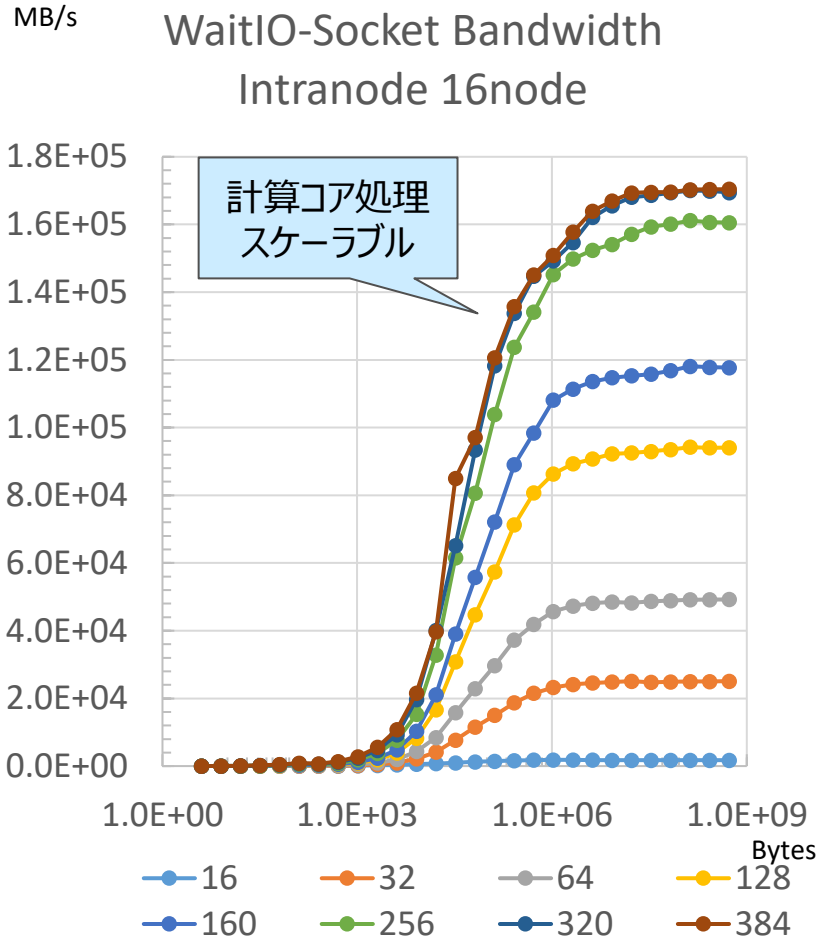
PingPong BW GB/s	BW ノード内	BW ノード間
Odyssey	21.3	0.35 (Tofu-D)
Aquarius	259.7	27.1 (IB)
OBCX	45.7	9.3 (OPA)
OFP	8.2	1.12 (OPA)

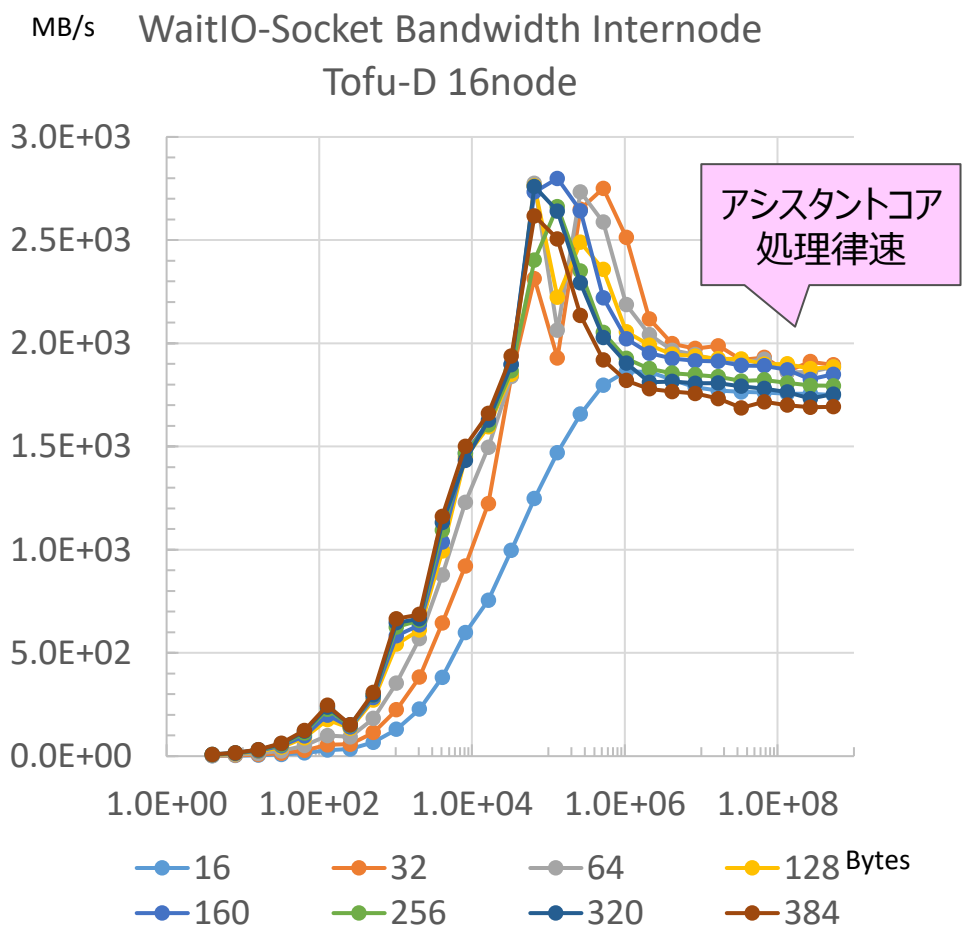
good

bad

- CPU性能とインタコネクト性能のバランス
  - Odyssey: ノード内に比べTofu-D性能が大幅に劣化
  - Aquarius: ノード内、InfiniBand性能ともに他システムに比べ高速
  - OFP: CPU性能が通信性能律速の主因

WaitIO-Socket性能: バンド幅(Odyssey) 16node

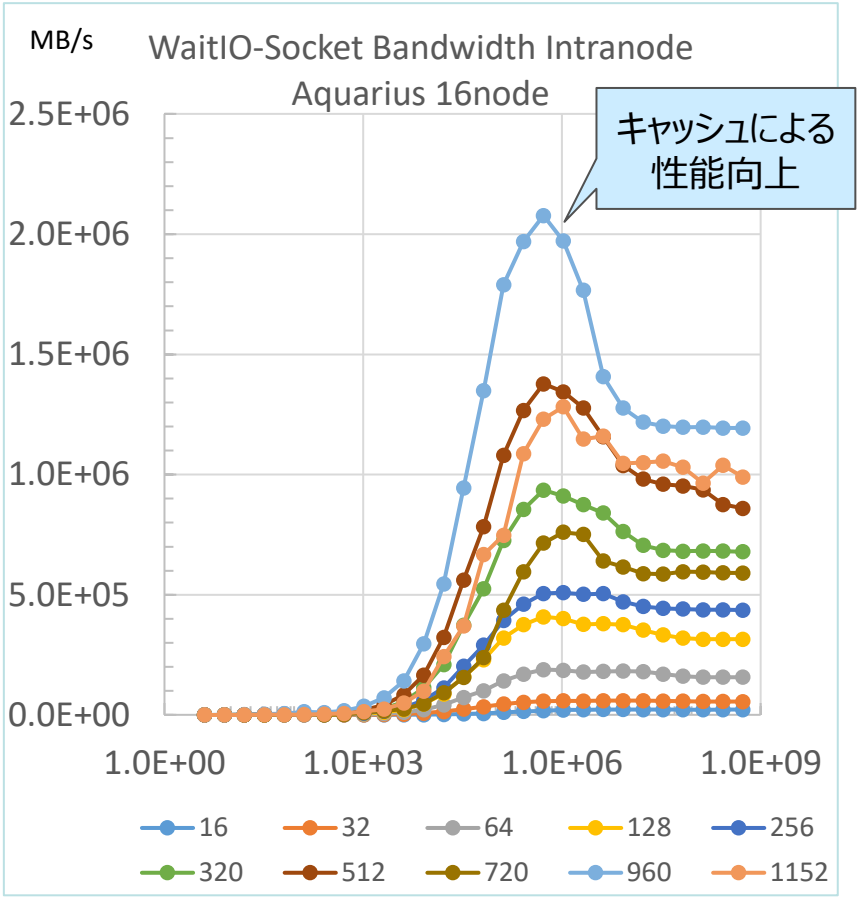


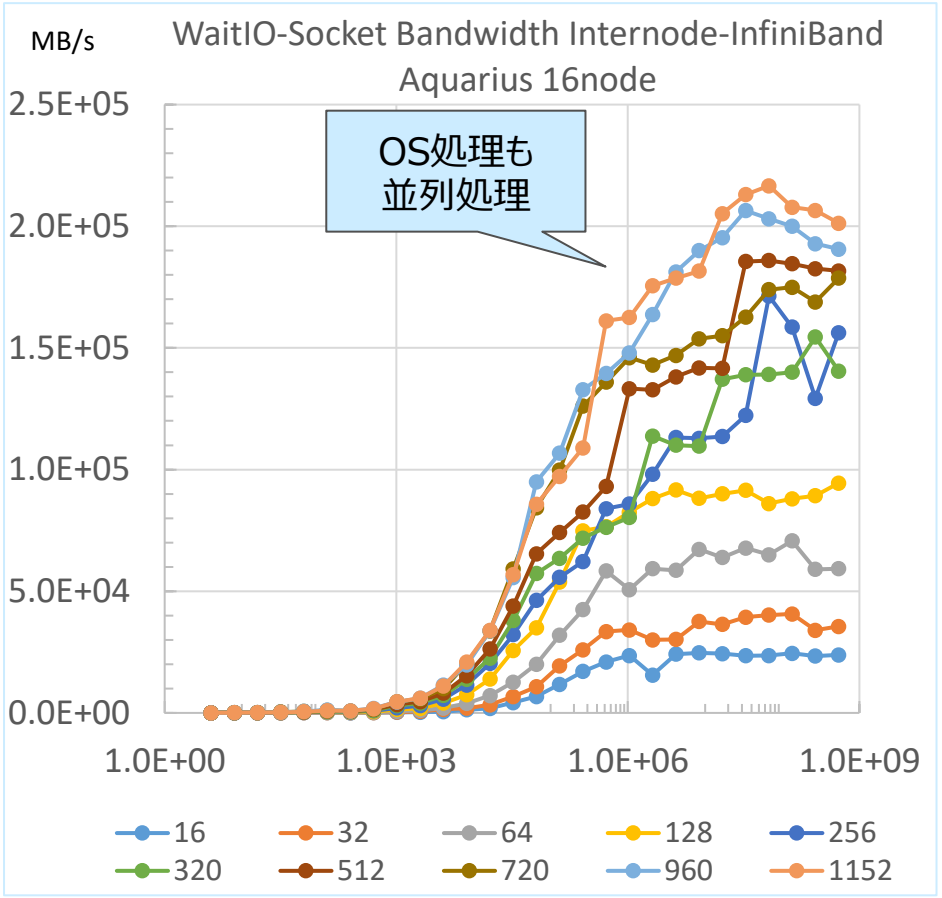


- ノード内: 170.4 GB/s (21.3 GB/s /node)
- ノード間: 2.80 GB/s (0.35 GB/s /node)

可変 proc-Node数固定

WaitIO-Socket性能: バンド幅(Aquarius) 16node



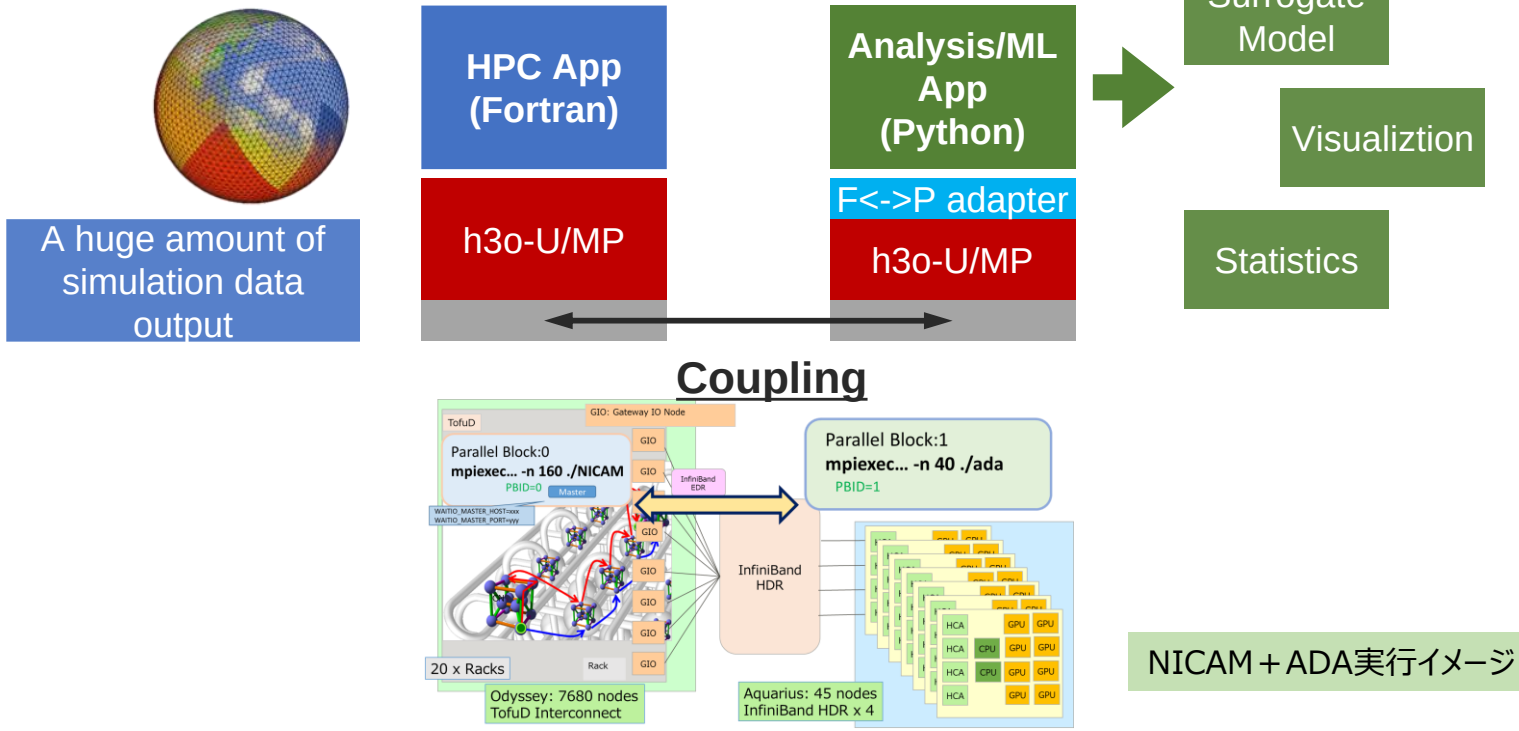


- ノード内: 2,077.6 GB/s (259.7 GB/s /node)
- ノード間: 216.6 GB/s (27.1 GB/s /node)

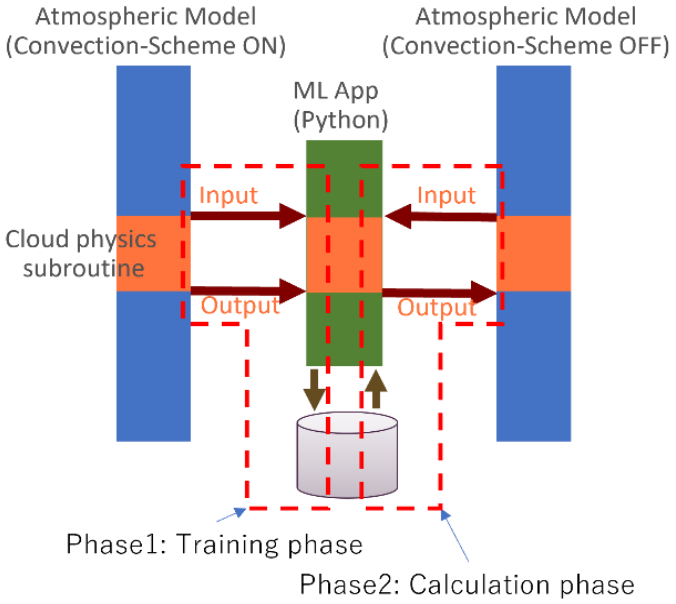
可変 proc-Node数固定

h3-Open-UTIL/MPを用いたアプリケーション実行評価

- 性能評価用のToyプログラムでの通信性能評価
- NICAM-AI 連成アプリケーション実行評価
  - 大気密度、内部エネルギー、水蒸気量の3変数で評価
  - DLライブラリ：PyTorch、学習アルゴリズム：3層MLP(多層パーセプトロン)



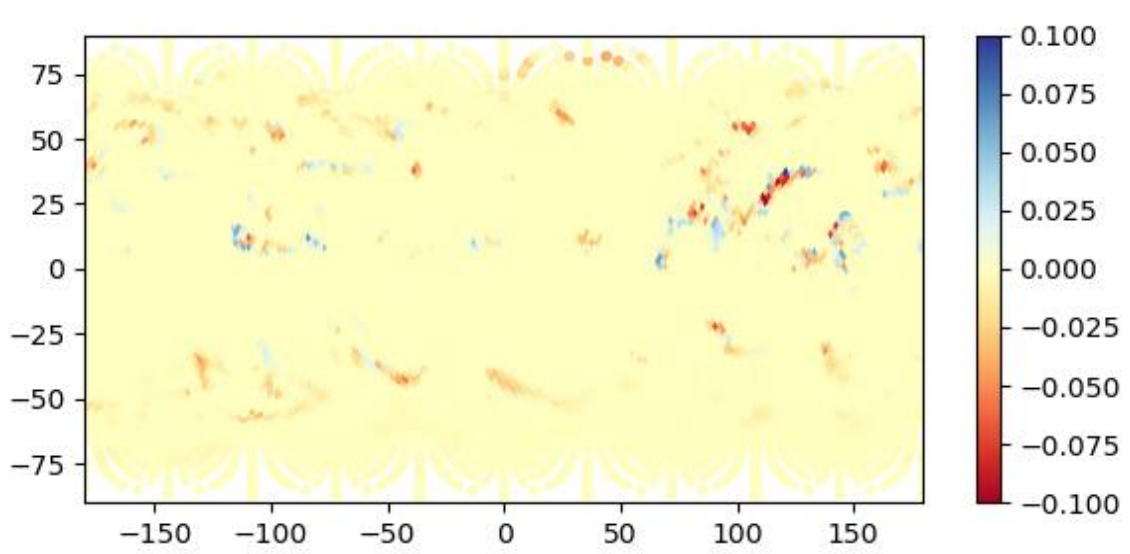
NICAM＋ADA実行イメージ



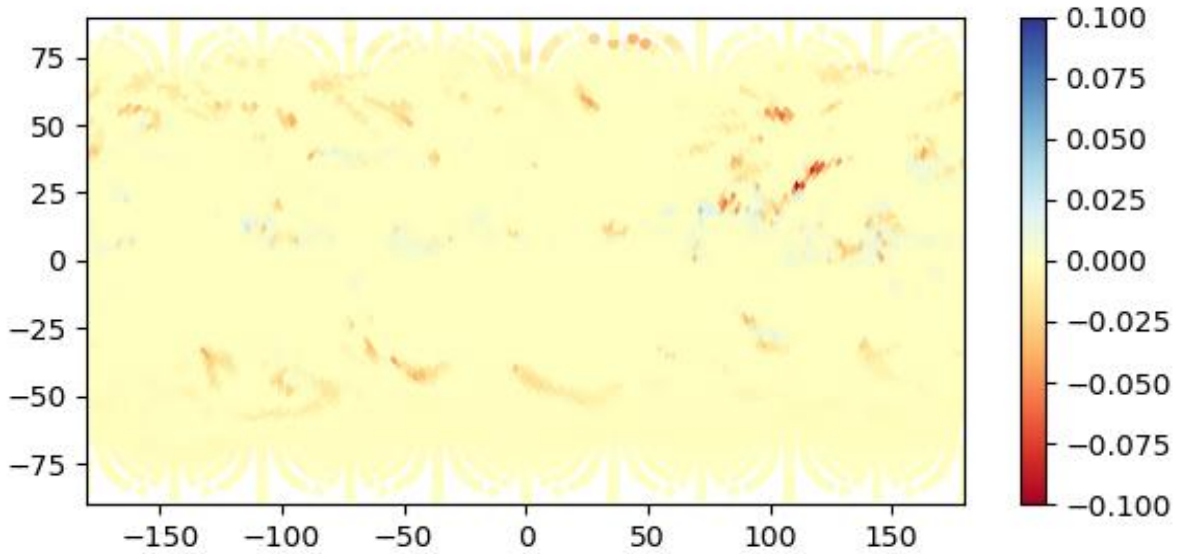
気候計算の学習・適用フェーズ時の処理モデル

NICAM-AI連成：大気密度の時間変化量比較

- 大気密度の時間変化量についてモデルによる計算値とAIによる再現値
  - 機械学習ライブラリ：PyTorch、学習アルゴリズム： 3層MLP
  - 単純アルゴリズムにも関わらず全体的な場の再現性は良好



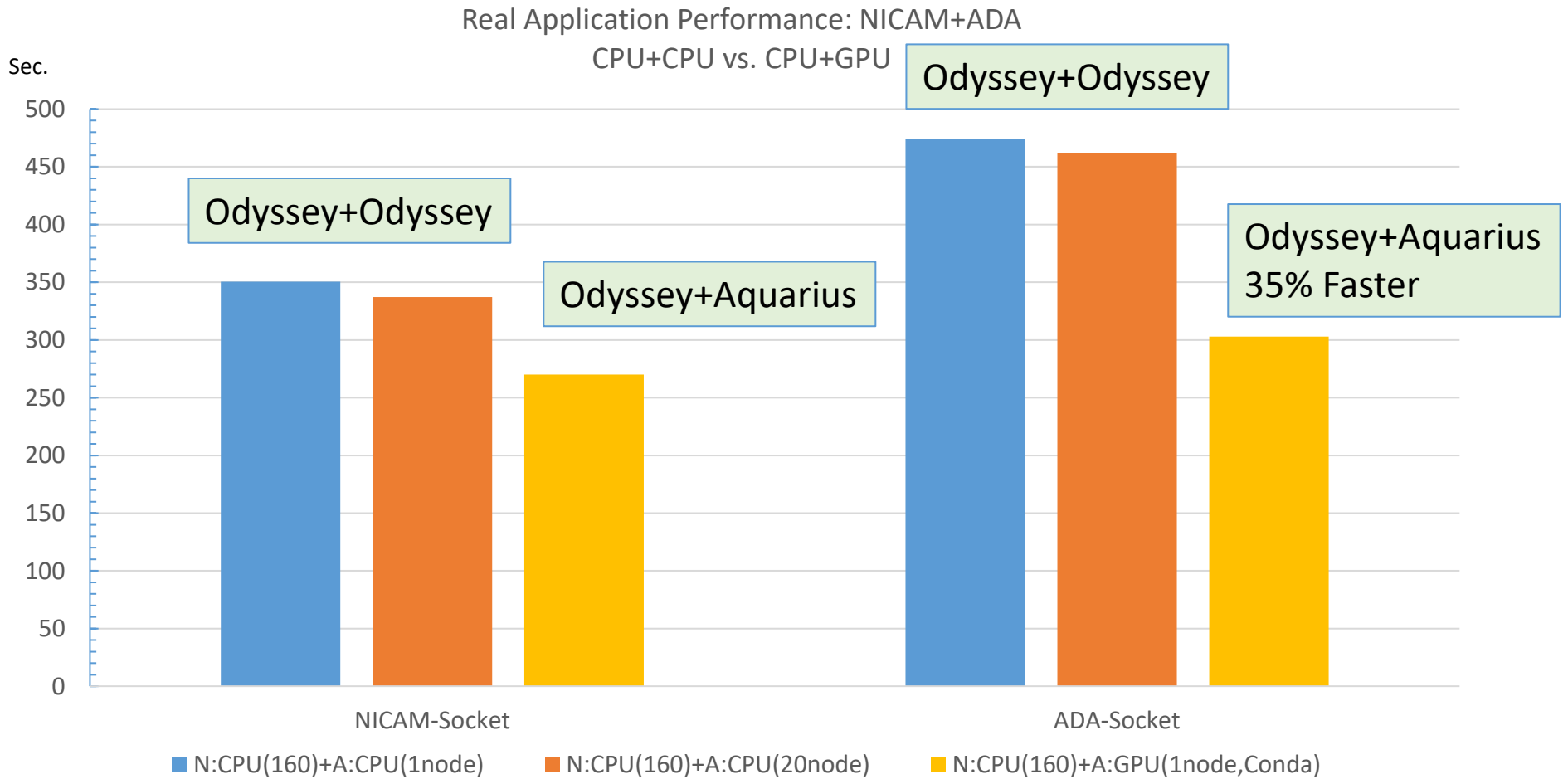
モデル計算による結果



AI学習による再現値

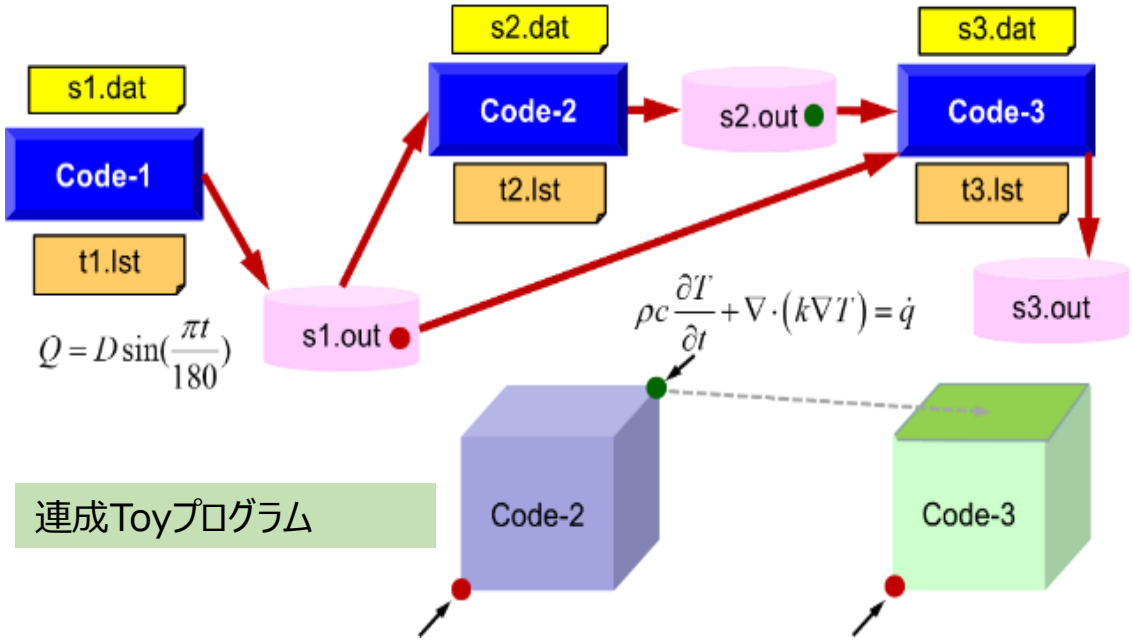
NICAM+ADA: 性能評価

- 異種分散(CPU+GPU)は同種分散(CPU+CPU)より 35%高速



WaitIO-Socketを用いたアプリケーション実行

- 連成ジョブ実行サービス利用せず
  - ジョブ１,2実行確認後、ジョブ3投入
  - 手動のため端末離席不能
  - ジョブHOLD発生でキャンセル必要：CPU時間節約のため



連成Toyプログラム

```
1) Submit Code-1 and Code-2 Jobs and wait until both jobs running
[z30xxx@wisteria01 run]$ pjsub s1-intel.sh
[INFO] PJM 0000 pjsub Job 335684 submitted.
[z30xxx@wisteria01 run]$ pjsub s2-a64fx.sh
[INFO] PJM 0000 pjsub Job 335685 submitted.
[z30xxx@wisteria01 run]$ pjstat
Wisteria/BDEC-01 scheduled stop time: 2022/04/22(Fri) 09:00:00 (Remain:  1days 18:55:30)

JOB_ID    JOB_NAME   STATUS  PROJECT   RSCGROUP       START_DATE      ELAPSE        TOKEN       NODE  GPU
335684    SYStest-wa RUNNING jhxxxxxa  debug-a        04/20 14:04:22<  00:00:09       0.1         1   8
335685    SYStest-wa RUNNING jhxxxxxo  debug-o        04/20 14:04:23<  00:00:08       0.0         1   -

2) Then Submit Code-3 Job and wait until Code-2 and Code-3 finish
[z30xxx@wisteria01 run]$ pjsub s3-a64fx.sh
[INFO] PJM 0000 pjsub Job 335686 submitted.
[z30xxx@wisteria01 run]$ pjstat
Wisteria/BDEC-01 scheduled stop time: 2022/04/22(Fri) 09:00:00 (Remain:  1days 18:55:15)

JOB_ID    JOB_NAME   STATUS  PROJECT   RSCGROUP       START_DATE      ELAPSE        TOKEN       NODE  GPU
335684    SYStest-wa RUNNING jhxxxxxa  debug-a        04/20 14:04:22<  00:00:24       0.2         1   8
335685    SYStest-wa RUNNING jhxxxxxo  debug-o        04/20 14:04:23<  00:00:23       0.0         1   -
335686    SYStest-wa RUNNING jhxxxxxo  debug-o        04/20 14:04:44<  00:00:02       0.0         1   -
[z30xxx@wisteria01 run]$

3) After Code-2 and Code-3 has finished, then kill Code-1 Job
[z30xxx@wisteria01 run]$ pjstat
Wisteria/BDEC-01 scheduled stop time: 2022/04/22(Fri) 09:00:00 (Remain:  1days 18:51:31)

JOB_ID    JOB_NAME   STATUS  PROJECT   RSCGROUP       START_DATE      ELAPSE        TOKEN       NODE  GPU
335684    SYStest-wa RUNNING jhxxxxxa  debug-a        04/20 14:04:22<  00:04:08       1.7         1   8
[z30xxx@wisteria01 run]$ pjdel 335684
[INFO] PJM 0100 pjdel Accepted job 335684.
[z30xxx@wisteria01 run]$
```

手動による連成ジョブ実行の様子

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 gr00
#PJM -j
#PJM -e err

module purge
module load fj
module load fjmpi
module load waitio

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

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

Code-1 for 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 gr00
#PJM -j
#PJM -e err

module purge
module load fj
module load fjmpi
module load waitio

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

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

Code-2 for 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 gr00
#PJM -j
#PJM -e err

module purge
module load fj
module load fjmpi
module load waitio

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

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

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 gr00
#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_PBID=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 gr00
#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_PBID=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 gr00
#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_PBID=2
export WAITIO_NPB=3

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

Code-3 for intel

WaitIO-Socketを用いたアプリケーション実行

- 投入後は連成ジョブ間で同期してアプリケーション実行開始

手動による連成ジョブ実行の様子

```
1) Submit Code-1 and Code-2 Jobs and wait until both jobs running
[z30xxx@wisteria01 run]$ pjsub s1-intel.sh
[INFO] PJM 0000 pjsub Job 335684 submitted.
[z30xxx@wisteria01 run]$ pjsub s2-a64fx.sh
[INFO] PJM 0000 pjsub Job 335685 submitted.
[z30xxx@wisteria01 run]$ pjstat
Wisteria/BDEC-01 scheduled stop time: 2022/04/22(Fri) 09:00:00 (Remain:  1days 18:55:30)

JOB_ID    JOB_NAME  STATUS  PROJECT   RSCGROUP       START_DATE      ELAPSE        TOKEN       NODE  GPU
335684    SYStest-wa RUNNING jhxxxxxa  debug-a        04/20 14:04:22<  00:00:09       0.1         1   8
335685    SYStest-wa RUNNING jhxxxxxo  debug-o        04/20 14:04:23<  00:00:08       0.0         1   -

2) Then Submit Code-3 Job and wait until Code-2 and Code-3 finish
[z30xxx@wisteria01 run]$ pjsub s3-a64fx.sh
[INFO] PJM 0000 pjsub Job 335686 submitted.
[z30xxx@wisteria01 run]$ pjstat
Wisteria/BDEC-01 scheduled stop time: 2022/04/22(Fri) 09:00:00 (Remain:  1days 18:55:15)

JOB_ID    JOB_NAME  STATUS  PROJECT   RSCGROUP       START_DATE      ELAPSE        TOKEN       NODE  GPU
335684    SYStest-wa RUNNING jhxxxxxa  debug-a        04/20 14:04:22<  00:00:24       0.2         1   8
335685    SYStest-wa RUNNING jhxxxxxo  debug-o        04/20 14:04:23<  00:00:23       0.0         1   -
335686    SYStest-wa RUNNING jhxxxxxo  debug-o        04/20 14:04:44<  00:00:02       0.0         1   -
[z30xxx@wisteria01 run]$

3) After Code-2 and Code-3 has finished, then kill Code-1 Job
[z30xxx@wisteria01 run]$ pjstat
Wisteria/BDEC-01 scheduled stop time: 2022/04/22(Fri) 09:00:00 (Remain:  1days 18:51:31)

JOB_ID    JOB_NAME  STATUS  PROJECT   RSCGROUP       START_DATE      ELAPSE        TOKEN       NODE  GPU
335684    SYStest-wa RUNNING jhxxxxxa  debug-a        04/20 14:04:22<  00:04:08       1.7         1   8
[z30xxx@wisteria01 run]$ pjdel 335684
[INFO] PJM 0100 pjdel Accepted job 335684.
[z30xxx@wisteria01 run]$
```

連成Toyプログラム：Code-3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

```
[z30xxx@wisteria01 waitio]$ ./waitio-serv -s -p 8801 -d
waitio-server host wisteria01:8801 started
wisteria01:-1::waitio_serv_loop:epoll called 4, 0x4 (sfd=4)!
 wisteria01:-1/-1(1722609):waitio-server: waitio_serv_loop accepted from 10.1.6.1 port=7858 fd=5
==== wisteria01:(1722609):waitio_serv_loop:master host has set to wa01-ib0 ====
wisteria01:-1::waitio_serv_loop:epoll called 4, 0x4 (sfd=4)!
 wisteria01:-1/-1(1722609):waitio-server: waitio_serv_loop accepted from 10.11.101.1 port=3210 fd=5
wisteria01:-1::waitio_serv_loop:epoll called 4, 0x4 (sfd=4)!
 wisteria01:-1/-1(1722609):waitio-server: waitio_serv_loop accepted from 10.11.101.9 port=30390 fd=5
wisteria01:-1::waitio_serv_loop:epoll called 4, 0x4 (sfd=4)!
 wisteria01:-1/-1(1722609):waitio-server: waitio_serv_loop accepted from 10.11.101.9 port=30902 fd=5
==== wisteria01:(1722609):waitio_serv_loop:master host  reset! ====
```

連成Toyプログラム：waitio-servの出力

終わりに

- Wisteria/BDEC-01システム上でh3-Open-SYS/WaitIO+h3-Open-UTIL/MPと異種システム間連成計算のためのジョブ実行環境
  - ジョブ実行環境の整備：手作業からジョブスケジューラ上で通常バッチ処理可能
  - Wisteria/BDEC-01上で6月より試験運用開始： アカウントがあれば利用可能

- ご興味を持たれた方へ
  - 詳しい利用方法を説明したチュートリアルを年1回ペースで開催予定
  - 昨年度開催資料・ビデオ：東京大学情報基盤センターの講習会ページ(2022/10/14)
    - 第191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講習会」
      - https://www.cc.u-tokyo.ac.jp/events/lectures/191/

Questions?