h3-Open-SYS/WaitIO: A System-wide Heterogeneous Communication Library to Couple Multiple MPI programs

Shinji Sumimoto, Takashi Arakawa, Yoshio Sakaguchi, Hiroya Matsuba*, Hisashi Yashiro, Toshihiro Hanawa, Kengo Nakajima

2023/11/29

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

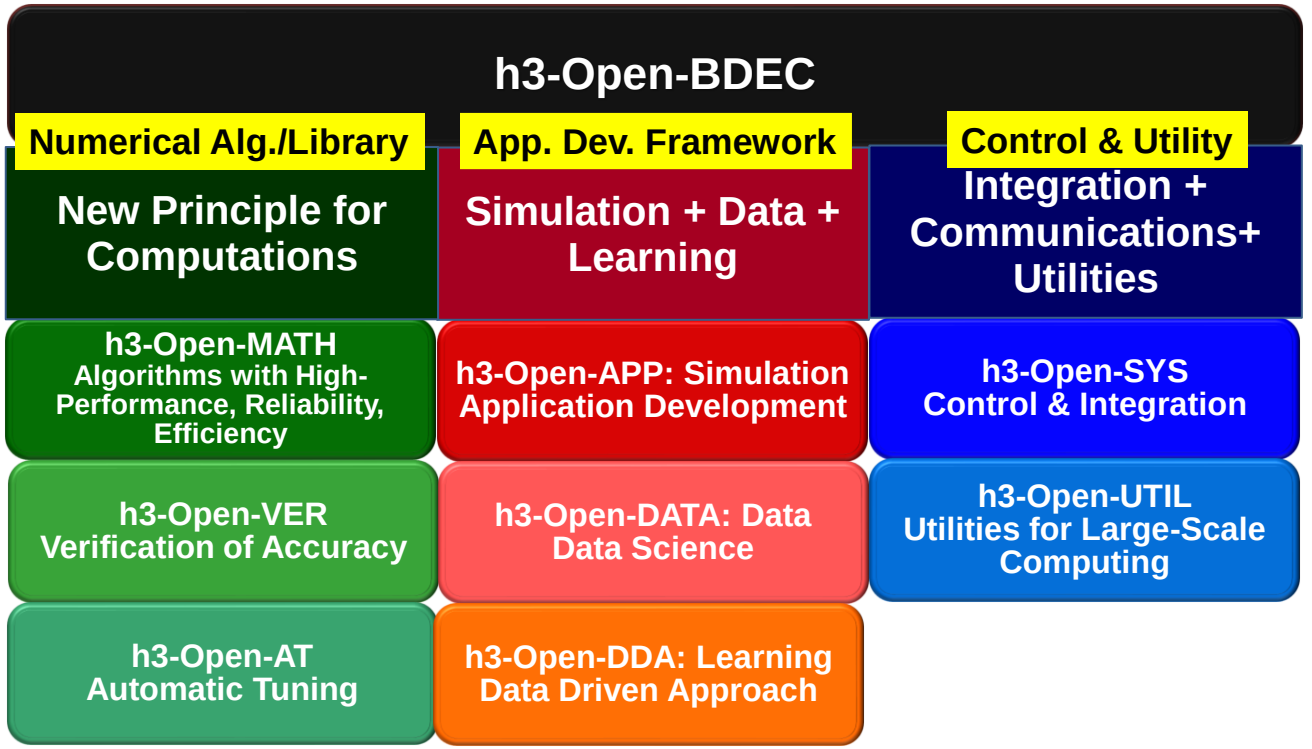
FUJITSU

R-CCS

*Achievements as a visiting professor at the University of Tokyo, 2021

h3-Open-BDEC Innovative Software Platform for Integration of (S+D+L) on the BDEC System, such as Wisteria/BDEC-01

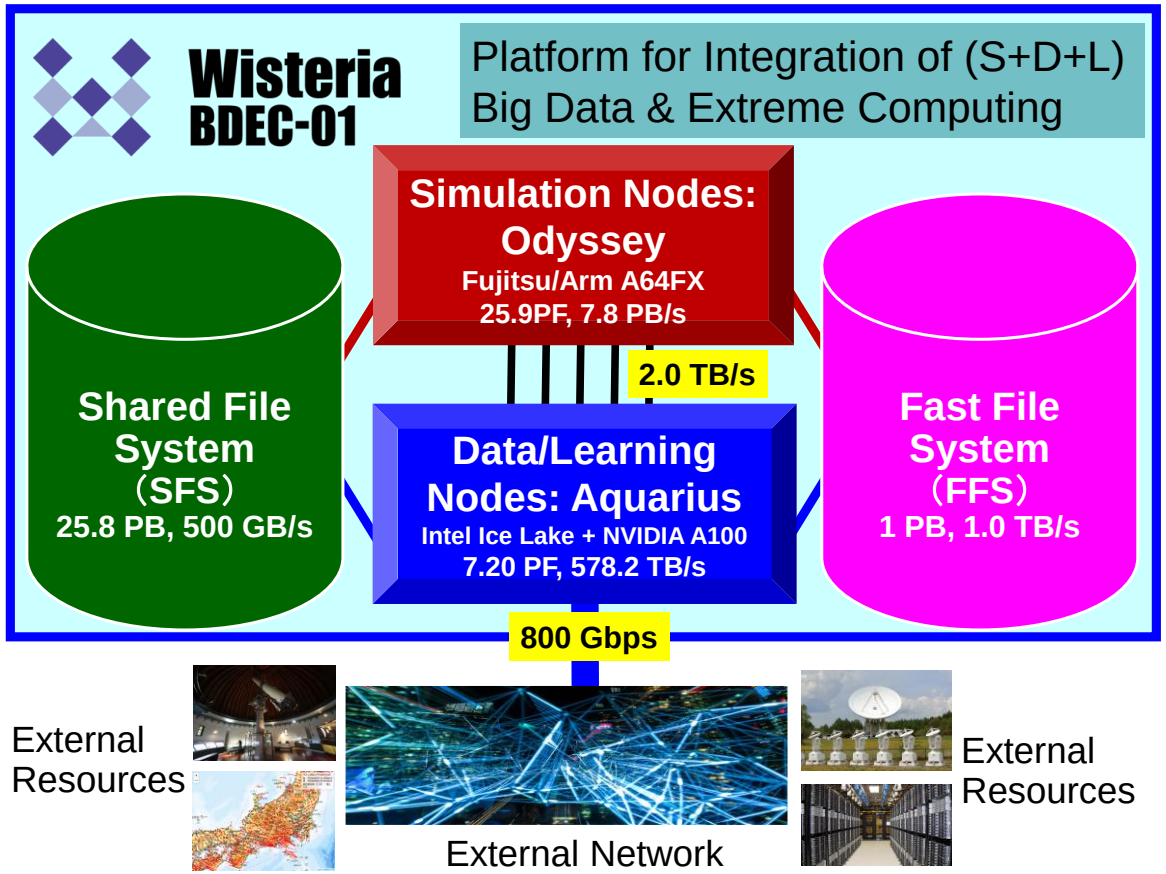
- “Three” Innovations
  - New Principles for Numerical Analysis by Adaptive Precision, Automatic Tuning & Accuracy Verification
  - Hierarchical Data Driven Approach (*h*DDA) based on Machine Learning
  - Software & Utilities for Heterogenous Environment, such as Wisteria/BDEC-01



Wisteria/BDEC-01

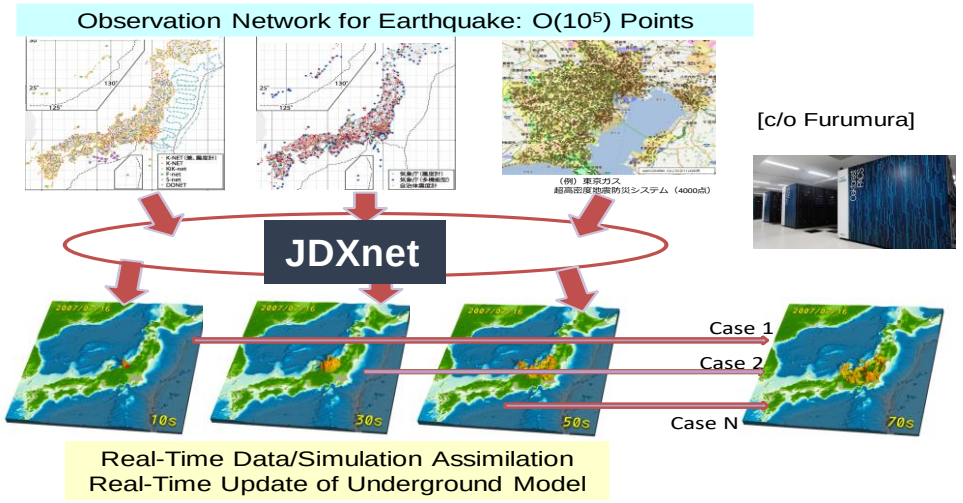
The 1st BDEC System (Big Data & Extreme Computing) Platform for Integration of (S+D+L)

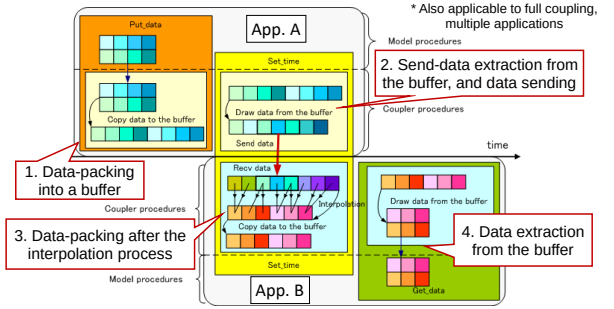
- Operation starts on May 14, 2021
- 33.1 PF, 8.38 PB/sec by **Fujitsu**
  - ~4.5 MVA with Cooling, ~360m^2
- **2 Types of Node Groups**
  - Hierarchical, Hybrid, Heterogeneous (h3)
  - **Simulation Nodes: Odyssey**
    - **Fujitsu PRIMEHPC FX1000 (A64FX), 25.9 PF**
      - **7,680 nodes (368,640 cores), Tofu-D**
      - **General Purpose CPU + HBM**
      - **Commercial Version of "Fugaku"**
  - **Data/Learning Nodes: Aquarius**
    - **Data Analytics & AI/Machine Learning**
    - **Intel Xeon Ice Lake + NVIDIA A100, 7.2PF**
      - **45 nodes (90x Ice Lake, 360x A100), IB-HDR**
    - **Some of the DL nodes are connected to external resources directly**
- File Systems: SFS (Shared/Large) + FFS (Fast/Small)

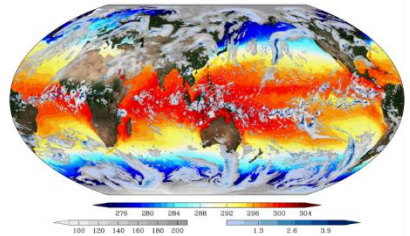


h3-OpenSYS/WaitIO: for Heterogeneous Coupling Computing

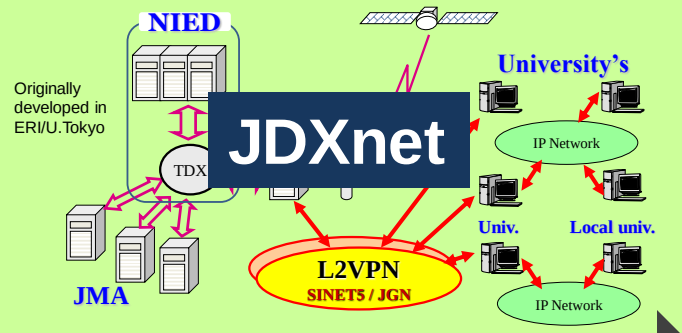
- With the spread of higher-scale computing systems, HPC applications have become able to solve more practical problems.
  - Example 1: Realizing real-time simulation with various kind of sensor data and estimate the future.
  - Example 2: Improving calculation accuracy by assimilating calculation results and real-time data.
  - Example 3: Using computer simulation results as machine learning data and doing simulation using inference computation.
- Fusion of multiple applications is the key with Simulation, Data processing and machine Learning
  - Heterogeneous Coupling Computing by WaitIO







Real Time Heterogeneous Coupling Computing using WaitIO on Wisteria/BDEC-01



Observed Data

Obs. Data

External Server for Observed Data

Wisteria BDEC-01

Data/Learning Nodes: Aquarius
Intel Ice Lake + NVIDIA A100
7.20 PF, 578.2 TB/s

2.0 TB/s

Simulation Nodes: Odyssey
Fujitsu/Arm A64FX
25.9PF, 7.8 PB/s

Output & Movie

Filter

Visualizer

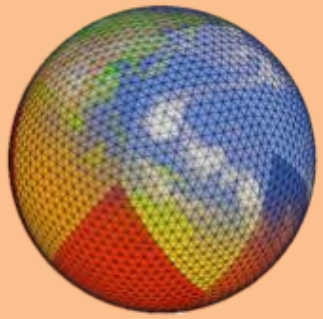
h3-Open-SYS/WaitIO-Socket

Seism3D/ OpenSWPC-DAF

Mesh Data

UG Model

Simulation and Python Application Coupling
by h3-Open-UTIL/MP (h3o-U/MP) + h3-Open-SYS/WaitIO on Wisteria/BDEC-01



A huge amount of simulation data output

HPC App (Fortran)

h3o-U/MP

Analysis/ML App (Python)

F<->P adapter

h3o-U/MP

Surrogate Model

Visualiztion

Statistics

Coupling

IB-EDR

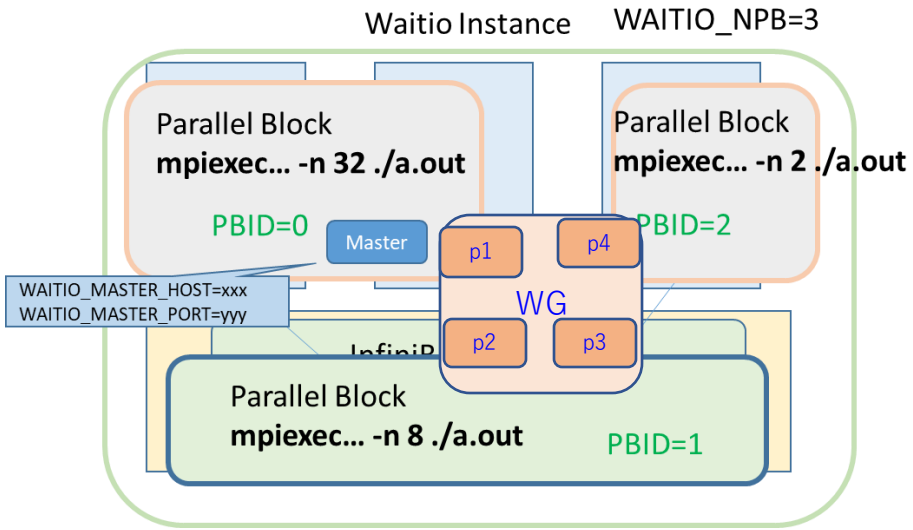
Wisteria BDEC-01 Odyssey

Wisteria BDEC-01 Aquarius

API of WaitIO: PB (Parallel Block) == Each Application

- Application is able to select communication processes among PBs

WaitIO API	Description
waitio_isend	Non-Blocking Send
waitio_irecv	Non-Blocking Receive
waitio_wait	Termination of waitio_isend/irecv
waitio_init	Initialization of WaitIO
waitio_get_nprocs	Process # for each PB (Parallel Block)
waitio_create_group waitio_create_group_wranks	Creating communication groups among PB's
waitio_group_rank	Rank ID in the Group
waitio_group_size	Size of Each Group
waitio_pb_size	Size of the Entire PB
waitio_pb_rank	Rank ID of the Entire PB



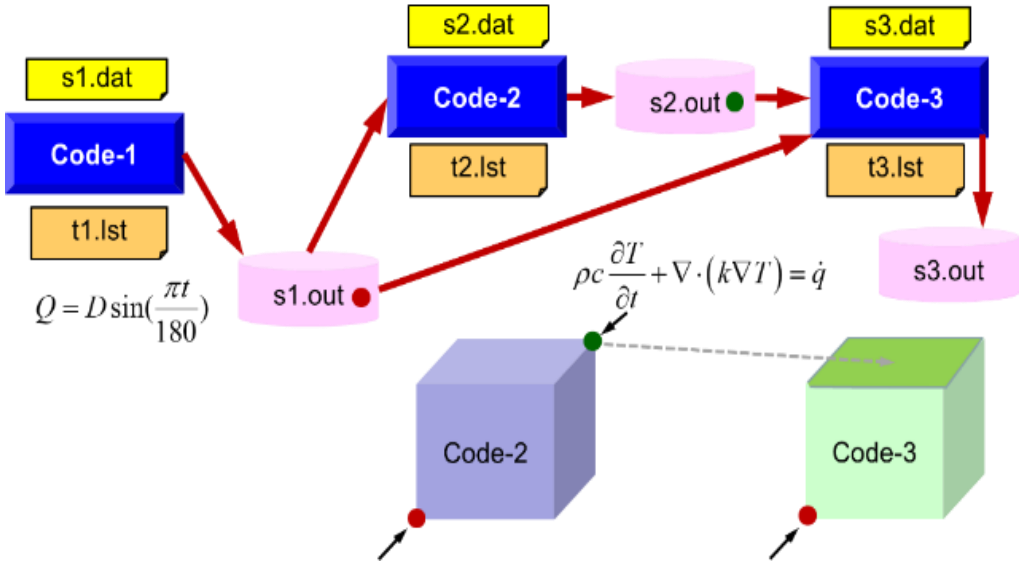
WaitIO-MPI Conversion Library

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

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

Existing File Coupling Application Program Conversion by WaitIO Easily

- Converting a Sample Coupling Program using File Coupling to WaitIO Coupling
  - Toy Programs of 3D unsteady-state heat transfer problems with the finite element method (FEM) using iterative linear solvers
  - Converting Three Toy Program Coupling from file based to WaitIO Based



	Code-1	Code-2	Code-3
Computing	Point source calculation	Unsteady three-dimensional heat conduction equation by FEM (MPI)	Unsteady three-dimensional heat conduction equation by FEM (MPI)
Input	s1.dat	s2.dat, s1.out	s3.dat, s1.out, s2.out
Output	s1.out, t1.lst	s2.out, t2.lst	S3.out, t3.out

Code-3: Reading Code-2 Output
Forced temperature fixed conditions in the plane of Z= Zmax

Code-1 Program Source: File Based

- Code-1: Original Code Using File Coupling

```
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 Program Source: WaitIO Based

- Converting Open/Read-Write to isend-irecv/wait
  - MPI Non-blocking Send/Recv Style Conversion

```
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')
```

```
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
```

The pHEAT-3D Application Conversion Example

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

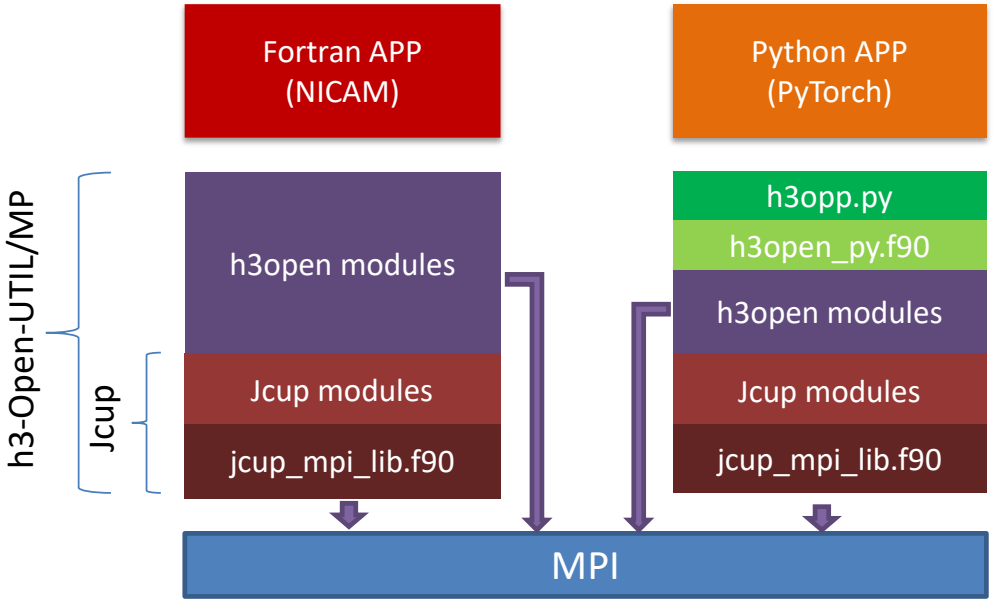
```
      include  'mpif.h'
      include 'waitio_mpif.h'
      integer(kind=kint ), dimension(:,:), save,allocatable :: req1
      integer, save :: NFLAG
      data NFLAG/0/

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

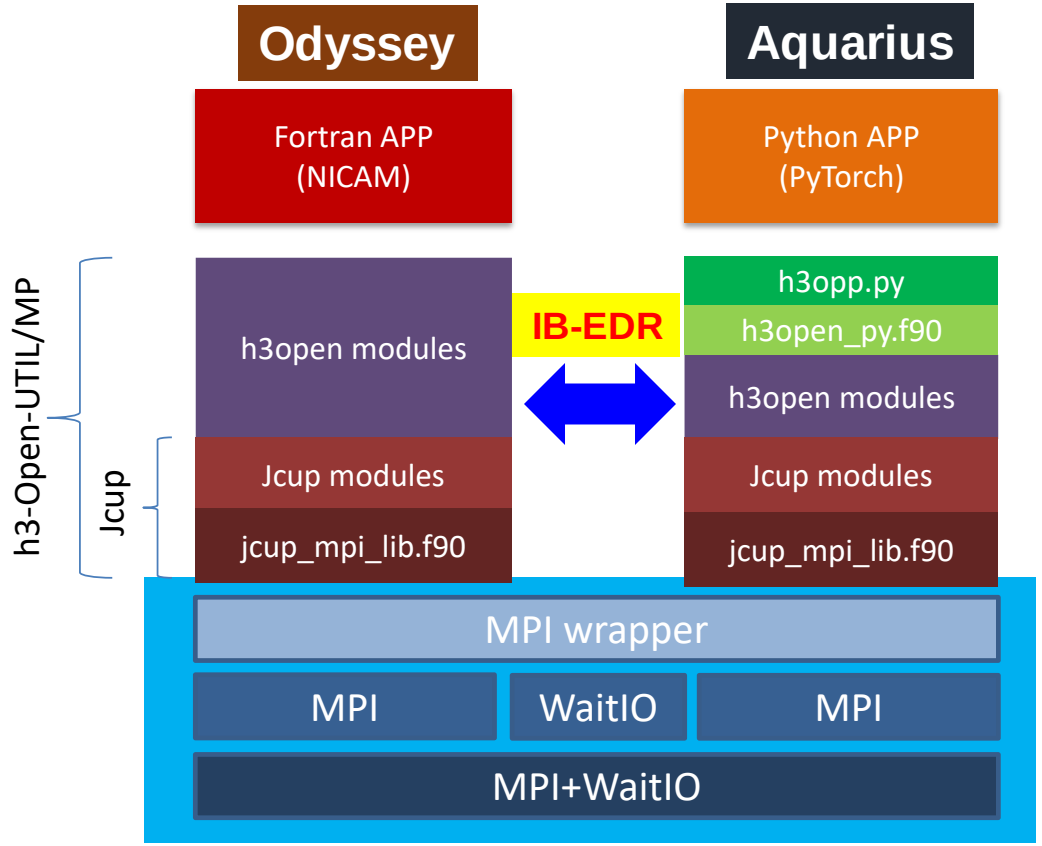
```
 !C-- RECV
      do neib= 1, NEIBPETOT
        istart= STACK_IMPORT(neib-1)
        inum  = STACK_IMPORT(neib  ) - istart
        call WAITIO_MPI_Irecv  (X(istart+N0+1),inum,
     &                WAITIO_MPI_DOUBLE_PRECISION,
     &                NEIBPE(neib), 0, WAITIO_SOLVER_COMM,
     &     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

      end subroutine  SOLVER_SEND_RECV
      end module      solver_SR
```

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

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

h3-Open-UTIL/MP + h3-Open-SYS/WaitIO-Socket

Operation Started in June 2022



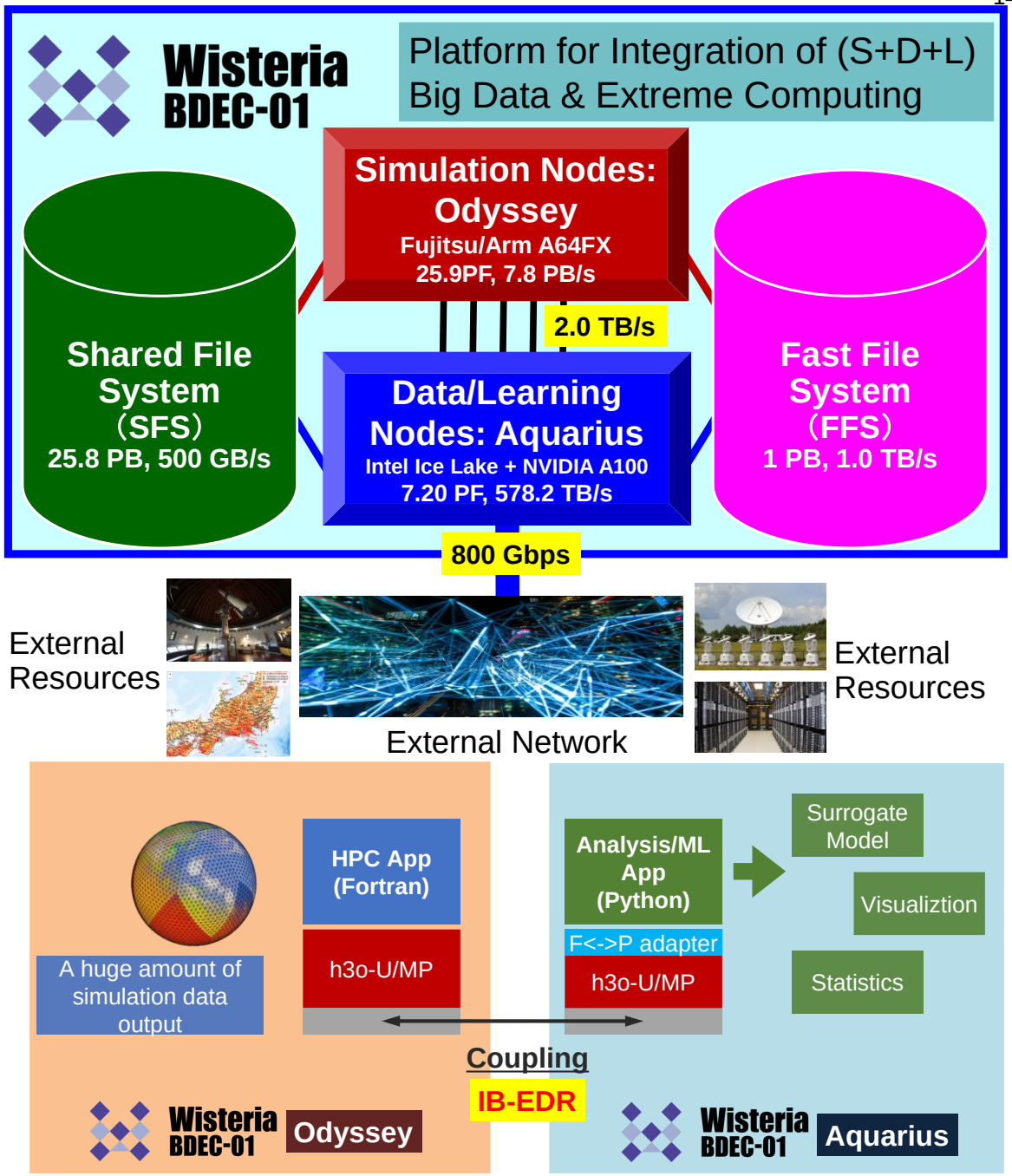


May 2021: Single MPI Environment

June 2022: Coupler＋WaitIO

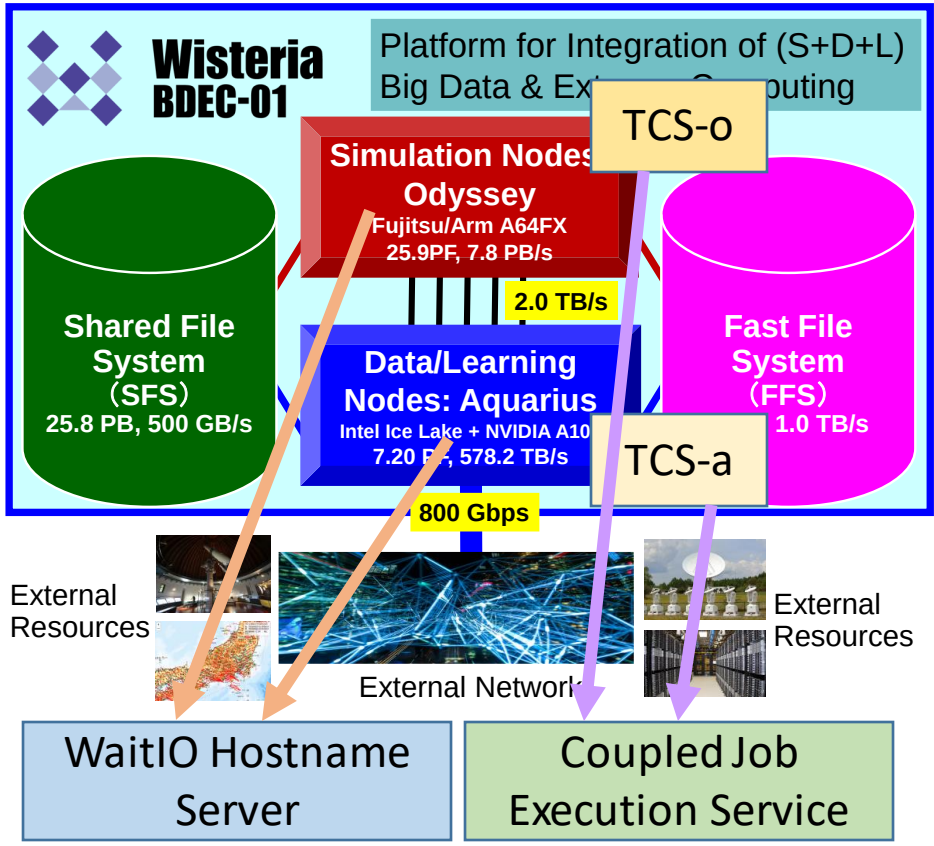
How to run the workloads

- Total Number of Nodes
  - Odyssey: 7,680 nodes: not so crowded
  - Aquarius: 45 nodes, 360 GPUs, very crowded
- One node of Aquarius is reserved for this type of workload on the integration of (S+D+L)
- 2 separate jobs (Odyssey, Aquarius) should be submitted
- If both jobs “grab” resources, execution starts.
- More flexible (& complicated) policy needed



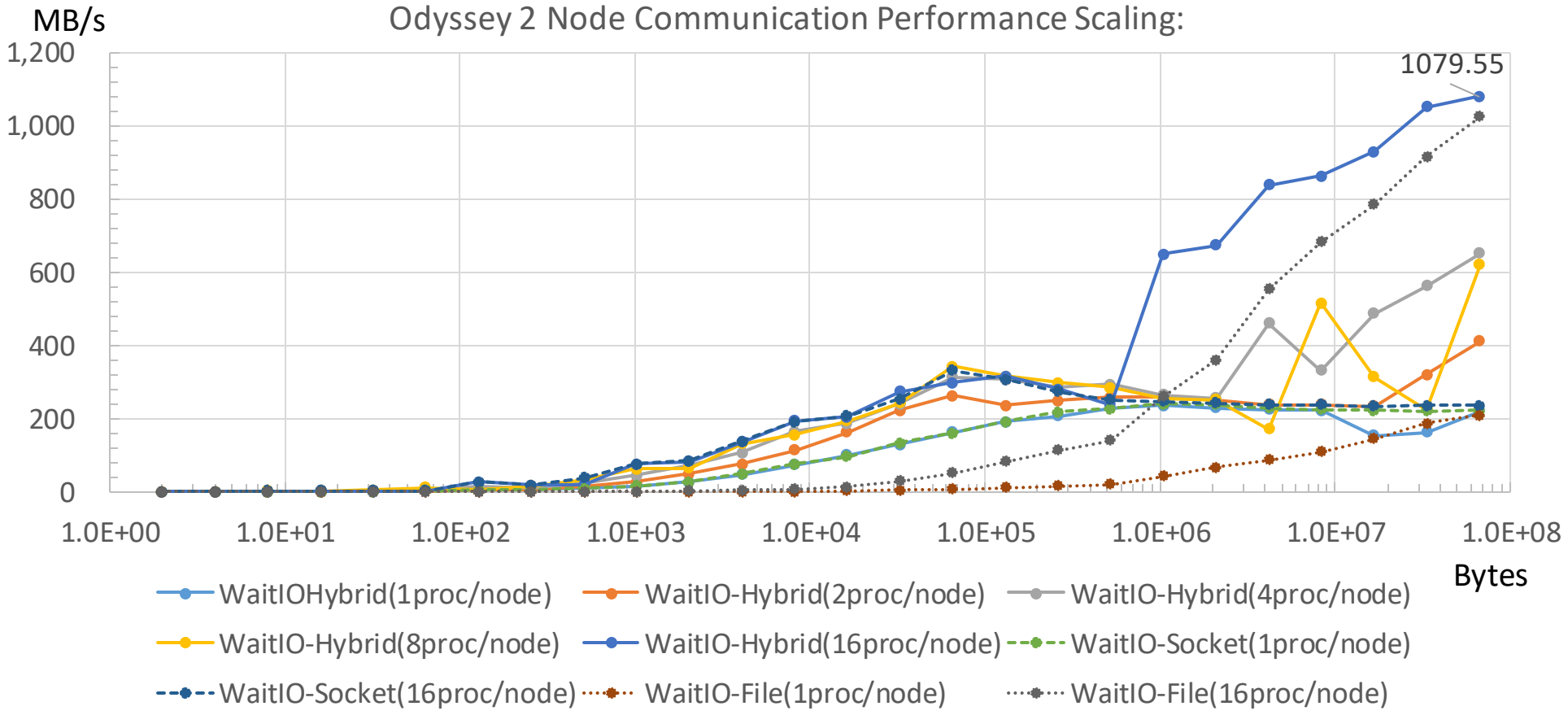
Job Co-execution between Different Batch Processing Systems

- We developed:
  - WaitIO Hostname Server:
    - Exchanging host information
  - Coupled Job Execution Service:
    - Watching job scheduler status and dispatching coupling jobs
- Coupled Job Execution Sequences:
  - Classification between Regular Execution Jobs and Coupled Execution Jobs
  - Do Simultaneous Execution: When resources of the multiple systems are available
  - WaitIO-Socket execution environment settings



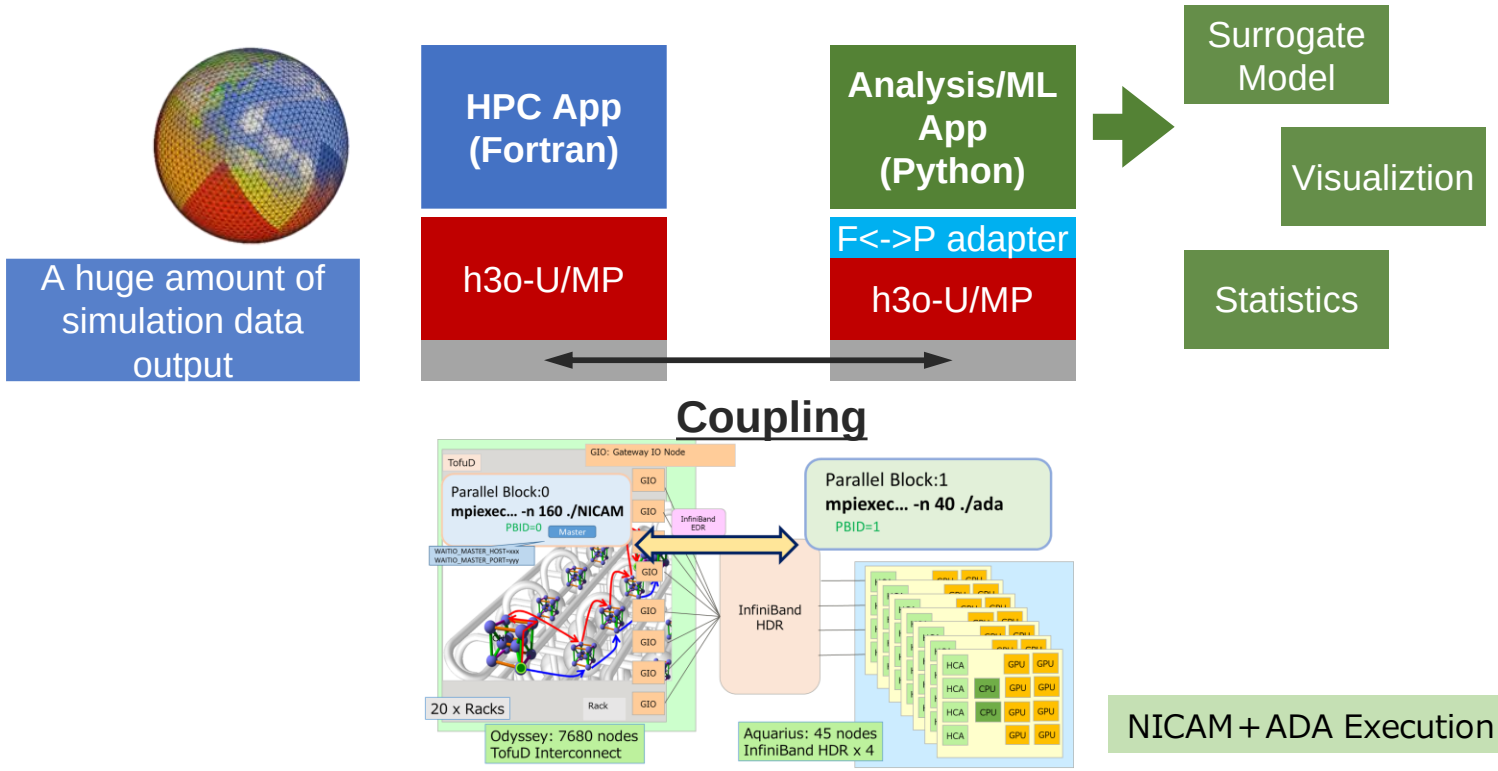
WaitIO-Socket, WaitIO-File vs. WaitIO-Hybrid PingPong Bandwidth Odyssey 2 node (Homogeneous)

- WaitIO-Hybrid : Achieved 1.1GB/s
  - Rendezvous Protocol is accelerated by WaitIO-Socket

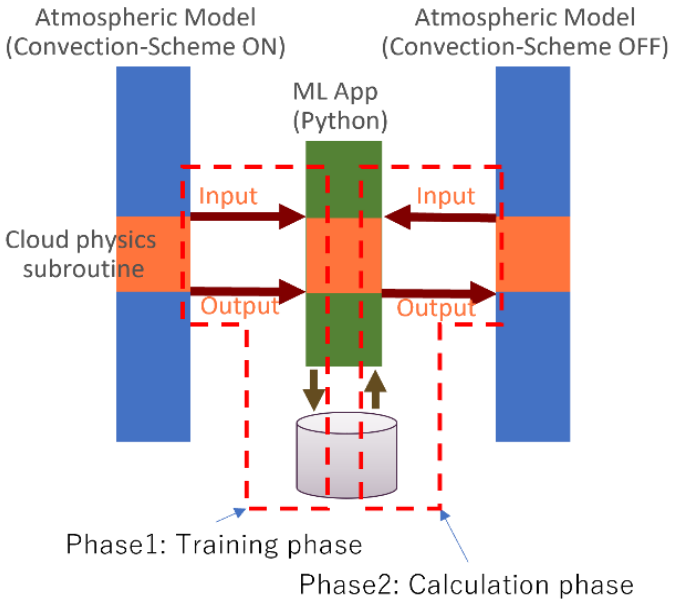


h3-Open-UTIL/MP+h3-Open-SYS/WaitIO Application Performance Evaluation

- NICAM-AI(ADA): Performance Evaluation of Coupling Computing
  - Total air density, Internal energy, Density of water vapor
  - Framework : PyTorch, Method : Three-Layer MLP
    - Resolution : horizontal : 10240, vertical : 78



NICAM+ADA Execution Image

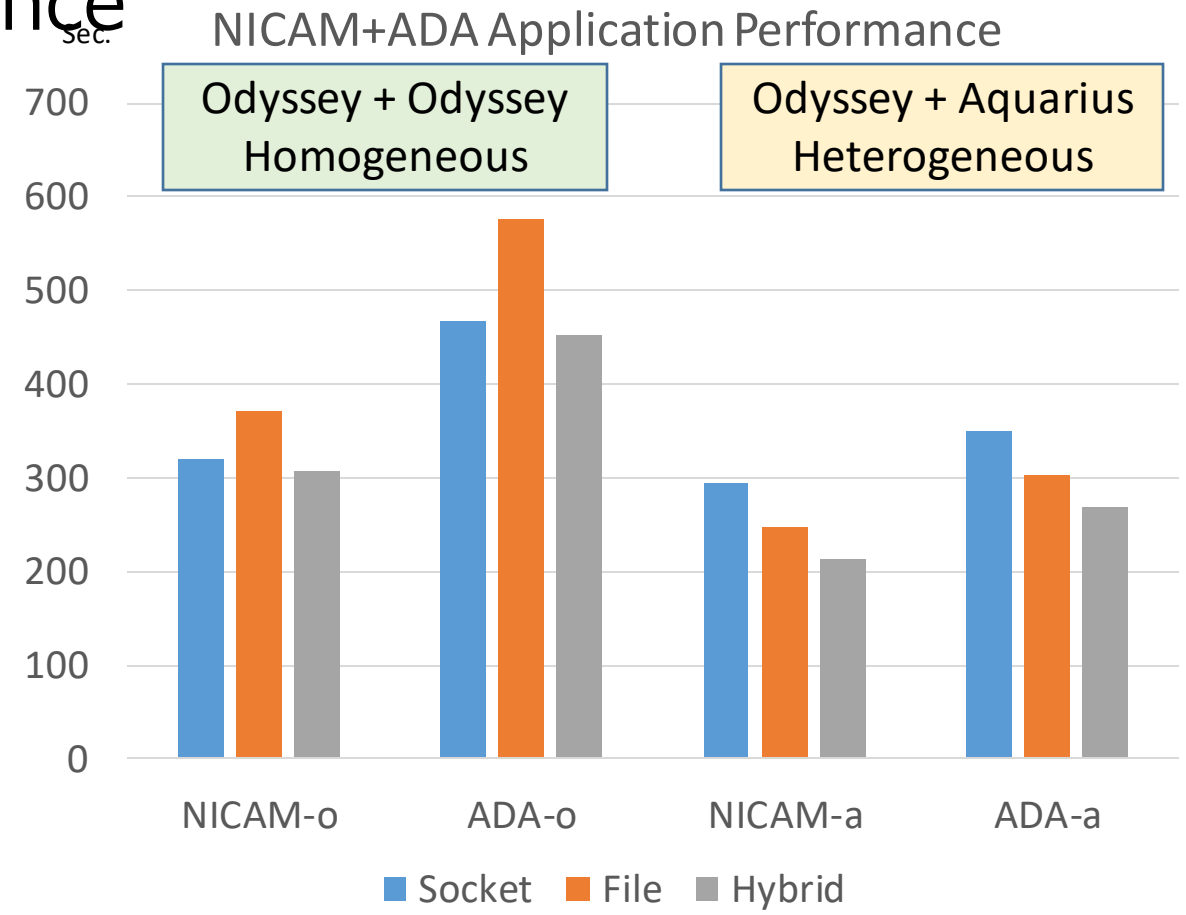


ML Processing Model for Climate Simulation

WaitIO Performance: NICAM+ADA

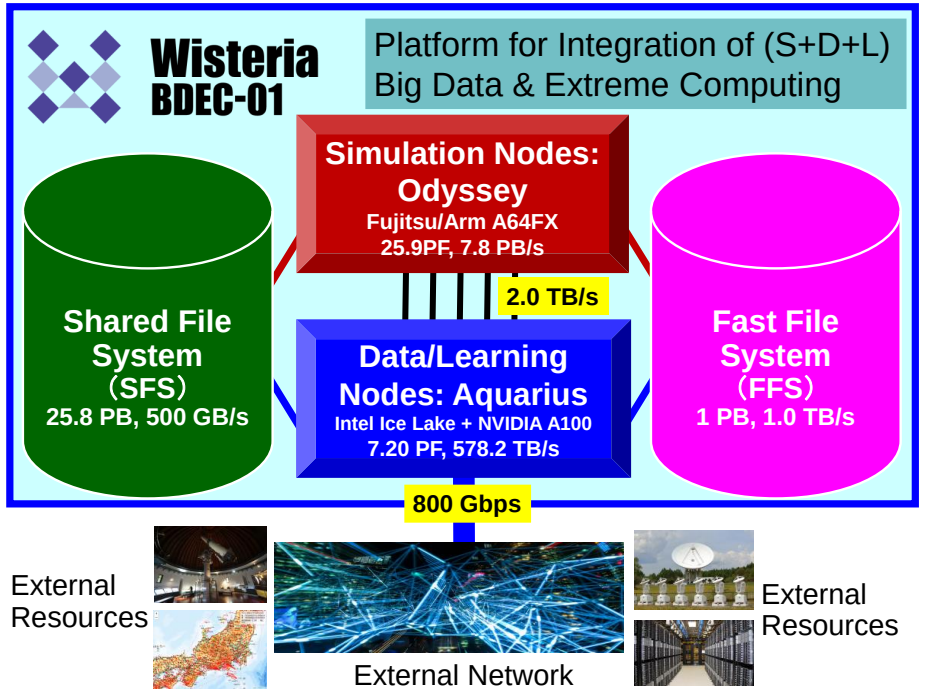
- Comparison of Socket, File, Hybrid
- WaitIO-Hybrid was best performance

Seconds	NICAM-o	ADA-o	NICAM-a	ADA-a
Socket	320.3	467.9	293.9	350.1
File	370.7	576.6	247.3	303.8
Hybrid	306.5	452.1	213.1	268.4



Summary of WaitIO(h3-OpenSYS/WaitIO)

- High performance System-wide communication for:
  - Coupling multiple MPI applications among multiple heterogeneous systems
- No extra software needed
  - Works among systems with POSIX Socket communication and Shared File
    - Existing related work needs same software stack among whole system nodes, so interoperability is limited
  - Existing MPI and compiler can be used
    - Fujitsu MPI, Intel MPI and Open MPI for GPU
- Future Work
  - WaitIO-Verbs for InfiniBand, RoCE, Slingshot **(Now Working)**
  - Real Application Porting: Several Projects are on-going



- Acknowledgement
  - This work is supported by “JSPS Grant-in-Aid for Scientific Research (S) (19H05662)”,and by "Joint Usage/Research Center for Interdisciplinary Large-scale Information Infrastructures (jh210022-MDH)".

Questions?