System-Wide Coupling Communication for Heterogeneous Computing Systems

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

2023/04/14

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

Overview of This Talk

- Heterogeneous Coupling Computing
  - h3-Open-BDEC Project
- H3-Open-SYS/WaitIO
  - Design and Implementation
  - Evaluation

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

Hierarchical, Hybrid, Heterogeneous
h3-Open-BDEC
Big Data & Extreme Computing

JAPAN SOCIETY FOR THE PROMOTION OF SCIENCE · FOUNDED 1932 ·

- "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

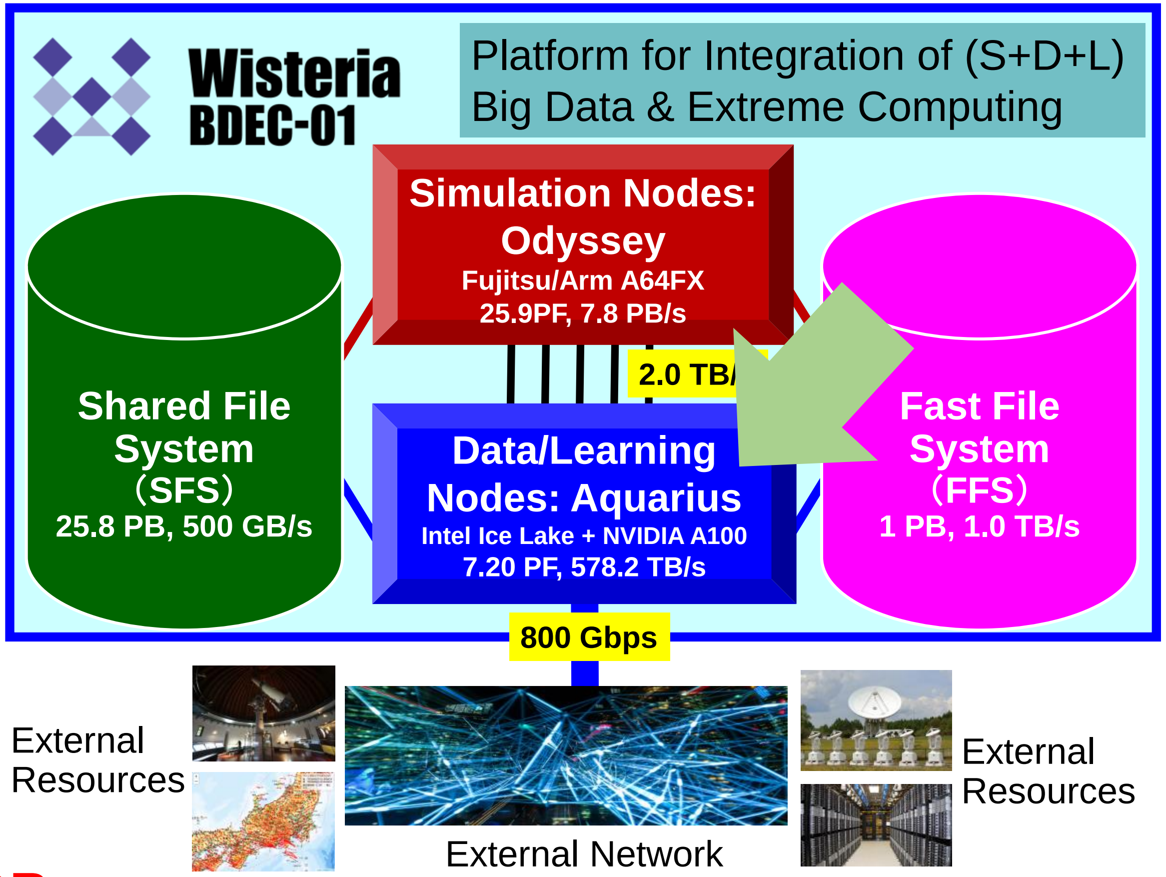
h3-Open-BDEC

Numerical Alg./Library	App. Dev. Framework	Control & Utility
New Principle for Computations	Simulation + Data + Learning	Integration + Communications+ Utilities
h3-Open-MATH Algorithms with High-Performance, Reliability, Efficiency	h3-Open-APP: Simulation Application Development	h3-Open-SYS Control & Integration
h3-Open-VER Verification of Accuracy	h3-Open-DATA: Data Data Science	h3-Open-UTIL Utilities for Large-Scale Computing
h3-Open-AT Automatic Tuning	h3-Open-DDA: Learning Data Driven Approach	

Wisteria/BDEC-01

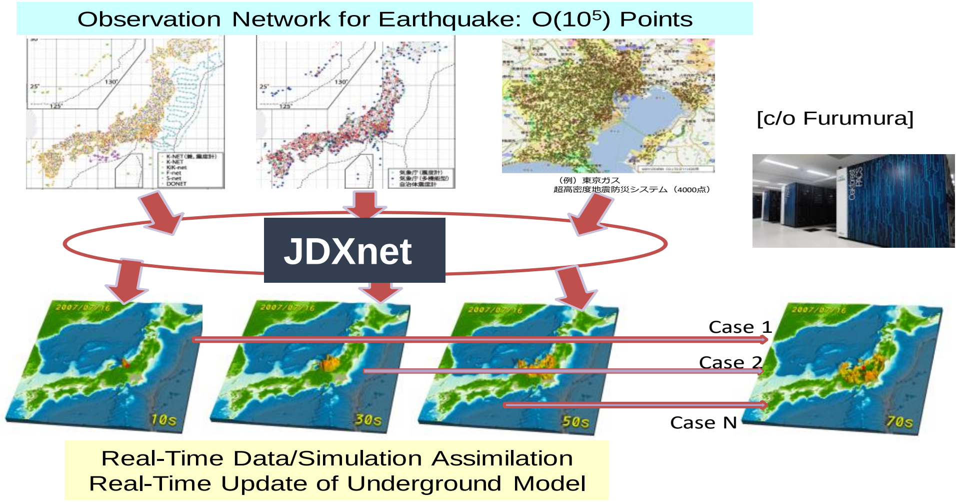
The 1st BDEC System (Big Data & Extreme Computing)
Platform for Integration of (S+D+L)

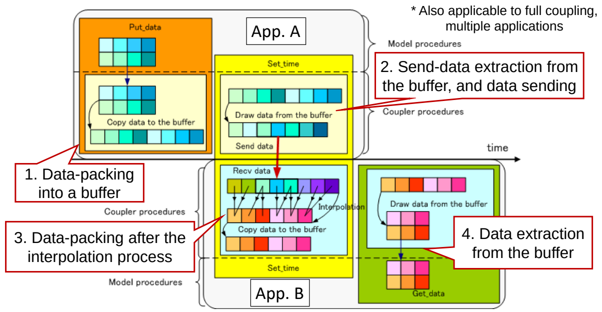
- Operation starts on May 14, 2021
- 33.1 PF, 8.38 PB/sec by **Fujitsu**
  - ~4.5 MVA with Cooling, ~360m^2
- **2 Types of Node Groups**
  - Hierarchical, Hybrid, Heterogeneous (h3)
  - **Simulation Nodes: Odyssey**
    - **Fujitsu PRIMEHPC FX1000 (A64FX), 25.9 PF**
      - **7,680 nodes (368,640 cores), 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)

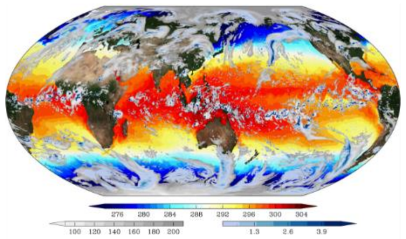


WaitIO: Heterogeneous Coupling Computing

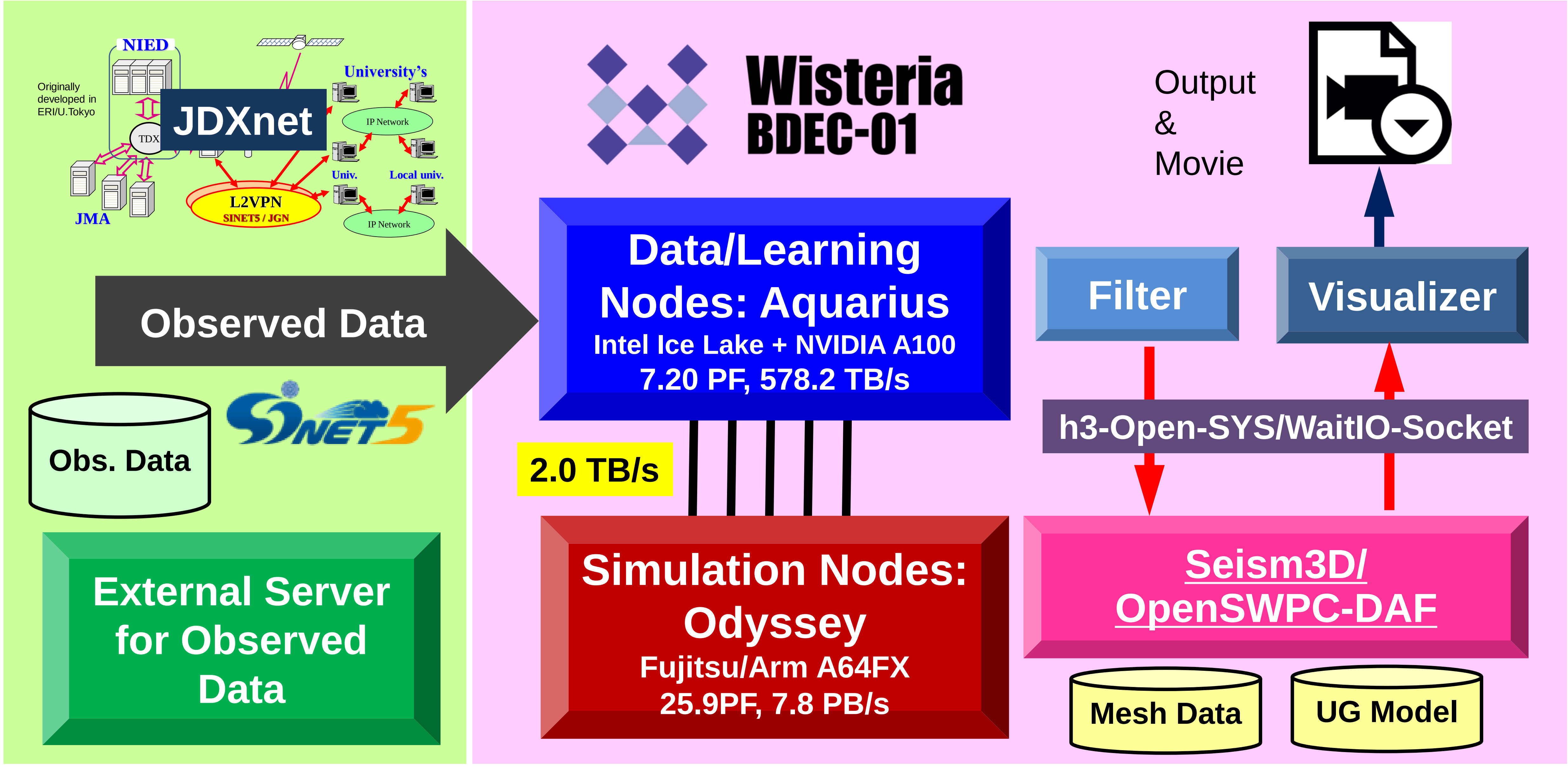
- With the spread of higher-scale computing systems, HPC applications have become able to solve more practical problems.
  - Example 1: Realizing real-time simulation with various kind of sensor data and estimate the future.
  - Example 2: Improving calculation accuracy by assimilating calculation results 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



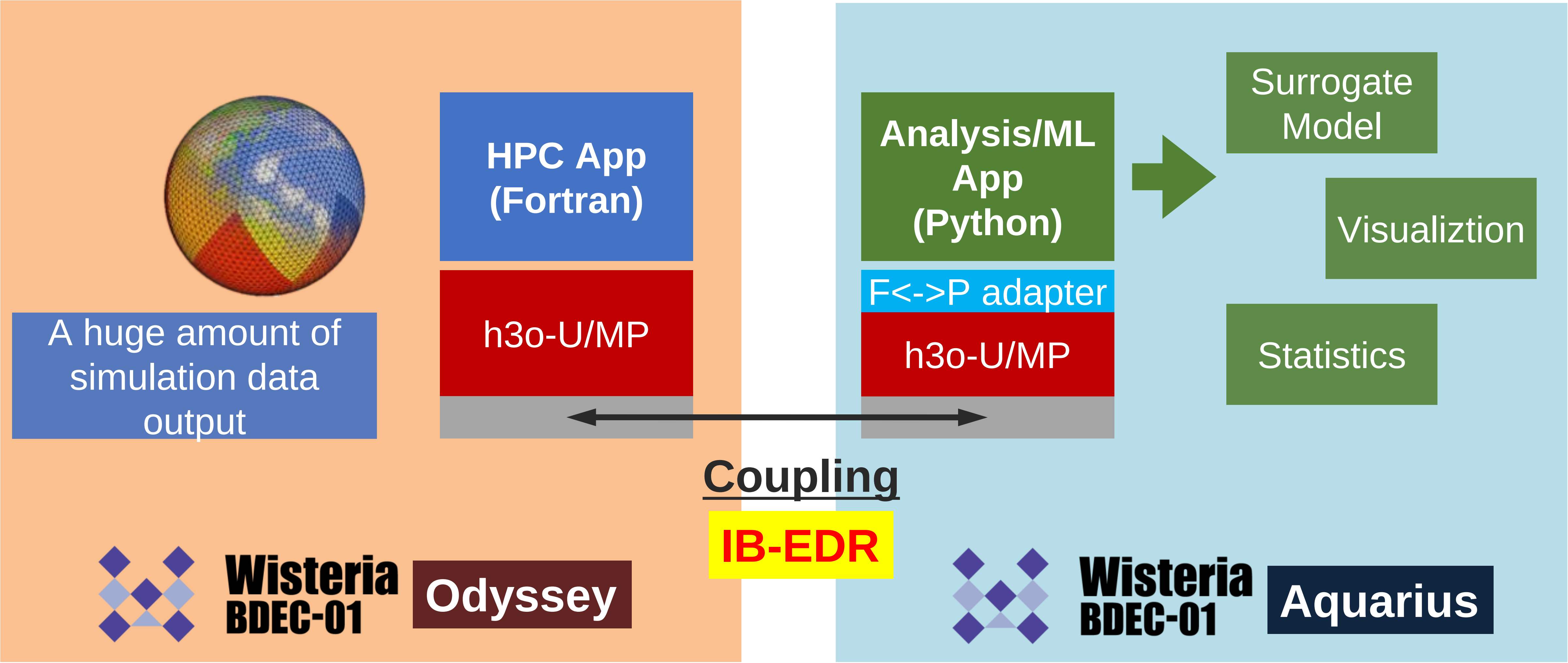




Real Time Heterogeneous Coupling Computing using WaitIO on Wisteria/BDEC-01

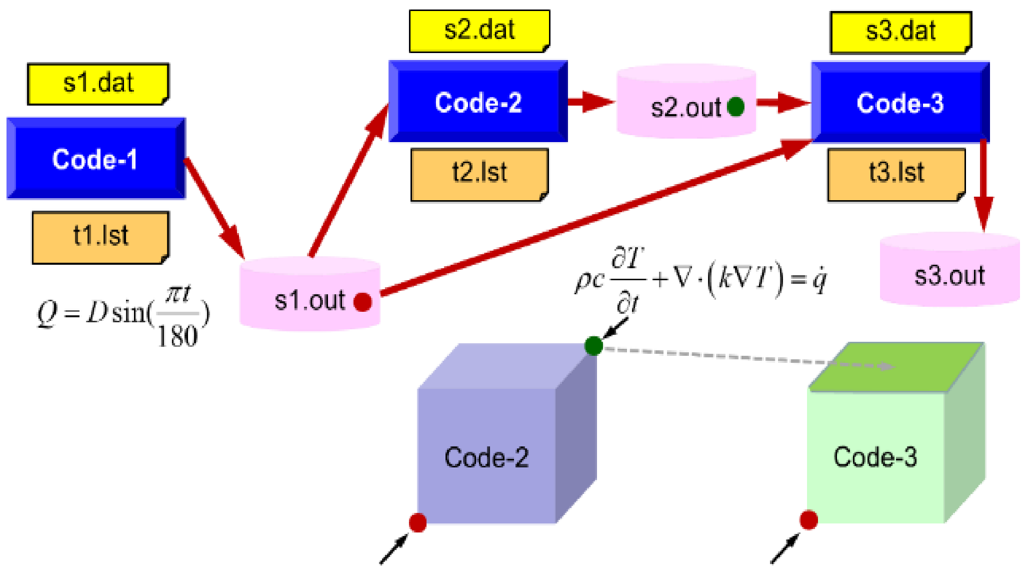


Simulation and Python Application Coupling
by h3-Open-UTIL/MP (h3o-U/MP) + h3-Open-SYS/WaitIO on Wisteria/BDEC-01



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

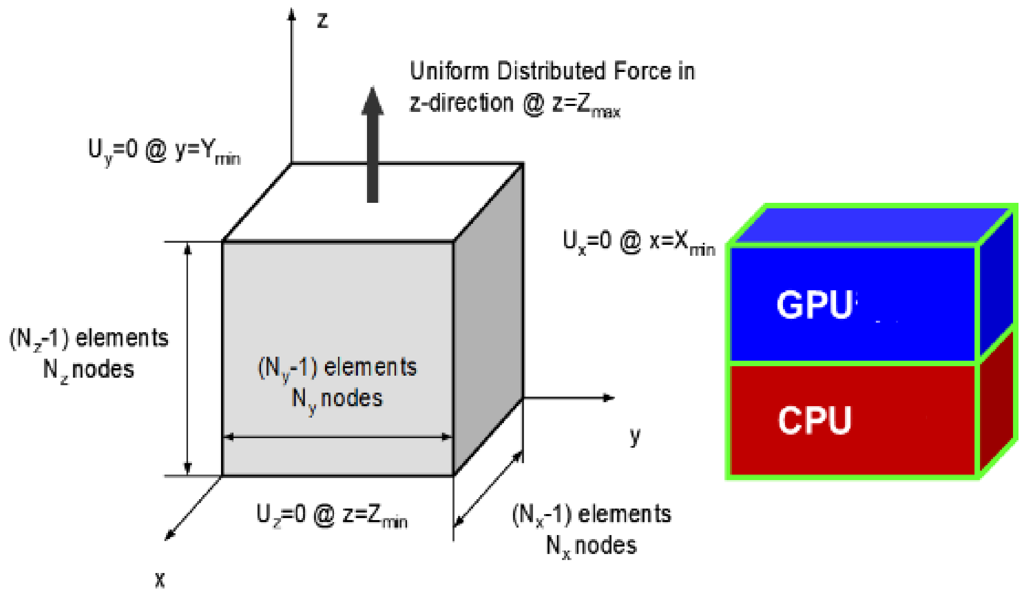


	Code-1	Code-2	Code-3
Computing	Point source calculation	Unsteady three-dimensional heat conduction equation by FEM (MPI)	Unsteady three-dimensional heat conduction equation by FEM (MPI)
Input	s1.dat	s2.dat, s1.out	s3.dat, s1.out, s2.out
Output	s1.out, t1.lst	s2.out, t2.lst	S3.out, t3.out

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

Heterogeneous Corporative Processing by WaitIO

- GeoFEM/Cube Toy Program
  - Co-processing with GPU and CPU by dividing into regions
- Co-processing with systems using different memory size and number of CPU cores
- Co-execute code with mixed cases (e.g. linear and non-linear problems) on each suitable system



GeoFEM/Cube

Heterogeneous Coupling Computing: Issue and Requirements

- Issue:
  - No standard communication protocols between systems
- Solution:
  - Developing Communication Library: h3-Open-SYS/WaitIO(WaitIO)
- WaitIO Communication Requirements:
  - Operating on Heterogeneous Hardware and Software:
    - Multiple applications on different kinds of nodes to be combined.
    - Applicable to Heterogeneous Hardware and Software Environments
  - Combining multiple application groups with less effort:
    - Software modification should be minimal
    - Close to the programming standard
    - Should be coexistent with other system software

WaitIO-Socket(File) Design

- Adoption of Socket(File) dedicated interface
  - Supports different types of processors, different types of interconnects, and different kinds of MPIs.
  - Possible to coexist with MPI and other system software.
- Providing APIs that conform to the MPI specifications:
  - Should be as much as possible the same as the communication libraries that are usually used.
- Operating with minimal computer resources: for > 10,000 procs.
  - Application user can select processes needed to communicate.
  - And all communication resources are allocated on demand.

WaitIO-File & WaitIO-Hybrid Design

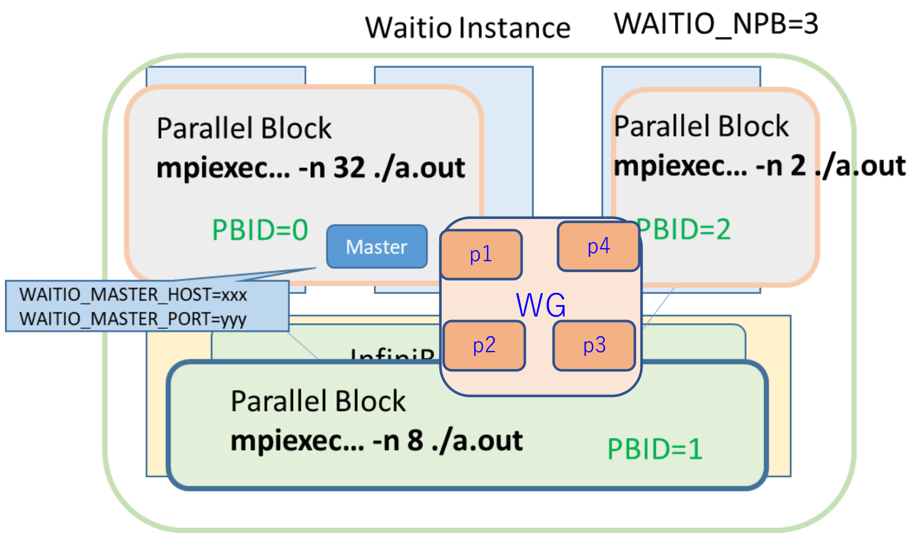
- WaitIO-File: Using Shared File System with POSIX semantics
  - FEFS, Lustre

- WaitIO-Hybrid:
  - For Short Messages: Using WaitIO-Socket
  - For Longer Messages: Using WaitIO-File

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



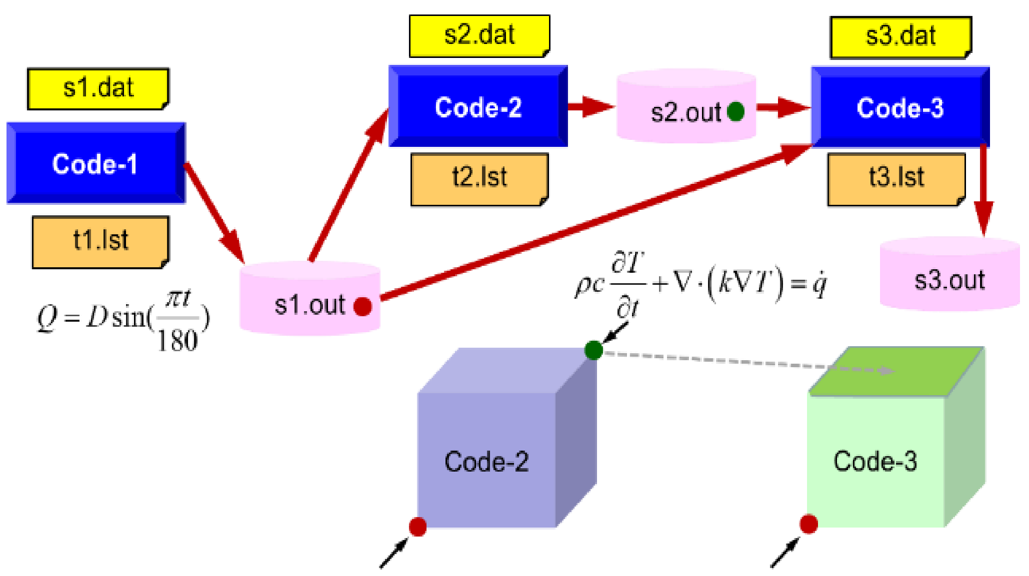
WaitIO-MPI Conversion Library

- 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

WaitIO: Program Conversion Example

- 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

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

call MPI_Init(ierr)
open (11,file='s1.dat',status='unknown')
read (11,*) ITERmax, Period, Interval, Val
write (*,'(a,i8)')      '##ITERmax ', ITERmax
write (*,'(a,1pe16.6)') '##Period  ', Period
write (*,'(a,1pe16.6)') '##Interval', Interval
write (*,'(a,1pe16.6//)') '##Val     ', Val
close (11)

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

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

```
do
  S1time= mpi_wtime ()

  do
    do iter= 1, ITERmax
      a= 1.d0
    enddo
    E0time= mpi_wtime (ierr)
    if ((E0time-S1time).gt.Interval) exit
  enddo

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

call MPI_Finalize()
stop
end
```

Code-1 Program Source: WaitIO Based

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

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

      include 'mpif.h'
      include 'waitio_mpif.h'

      call MPI_Init(ierr)
      open (11,file='s1.dat',status='unknown')
      read (11,*) ITERmax, Period, Interval, Val
      write (*,'(a,i8)')       '##ITERmax ', ITERmax
      write (*,'(a,1pe16.6)') '##Period  ',  Period
      write (*,'(a,1pe16.6)') '##Interval',  Interval
      write (*,'(a,1pe16.6//)') '##Val     ',  Val
      close (11)

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

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

```
        write (*,'(2(1pe16.6))') ttt, Source
        do dest=1, 2
          call WAITIO_MPI_Isend (ttt, 1,
     &          WAITIO_MPI_DOUBLE_PRECISION,
     &          dest, 0, WAITIO_SOLVER_COMM, req1(1,2*(dest-1)+1), ierr)
          if(ierr.ne.0) goto 100
          call WAITIO_MPI_Isend (Source, 1,
     &          WAITIO_MPI_DOUBLE_PRECISION,
     &          dest, 0, WAITIO_SOLVER_COMM, req1(1,2*(dest-1)+2), ierr)
          if(ierr.ne.0) goto 100
        enddo
        call WAITIO_MPI_Waitall (4, req1, sta1, ierr)
        if(ierr.ne.0) goto 100
      enddo
  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)
```

Implementation of WaitIO-Socket and File

- Communication protocol:
  - The Eager protocol (<= 128 bytes), the Rendezvous protocol (>128 bytes)
- Communication progress processing: When waitio_wait() is called
  - Except for the Eager protocol for which communication has been established.
- WaitIO-Socket Implementation
  - Uses POSIX-Socket and Linux epoll function.
  - WaitIO initialization: waitio_init()
    - A set of IP addresses and port numbers on all PBs is shared among all processes on PBs.
- WaitIO-File Implementation
  - Read/Write codes are the same as WaitIO-Socket
  - Initialization and Receive message are implemented using shared files.
- Receive message processing:
  - Socket: Implemented using the “Linux epoll” (event polling) functions.
  - File: Implemented using polling files.
- Send message processing: Implemented by the basic “write” system-call directly
  - Socket: Shifting to the processing by the “Linux epoll” when the write processing to the socket causes an error such as an EINTR factor.
  - File: Not use the “Linux epoll”, only write system call
- Ensuring communication reliability:
  - Magic numbers and sequence numbers are introduced in control packets to detect errors.

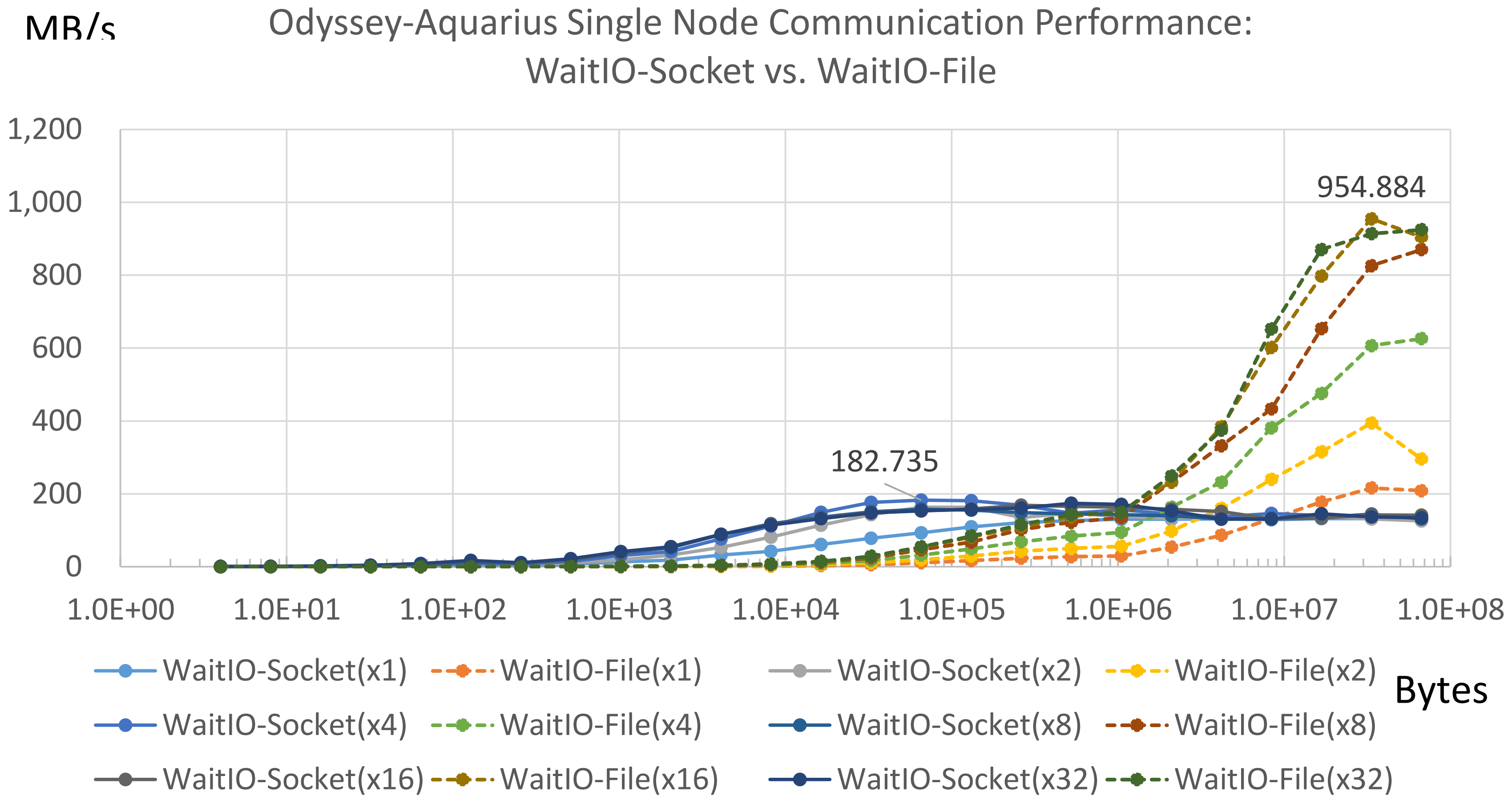
Evaluation of WaitIO-Socket

- PingPong Communication Bandwidth
  - Multi-stream PingPong Communication

- Application Performance with h3-OpenUTIL/MP on Wisteria/BDEC-01
  - For Heterogeneous Computing(Simulation + Machine Learning)

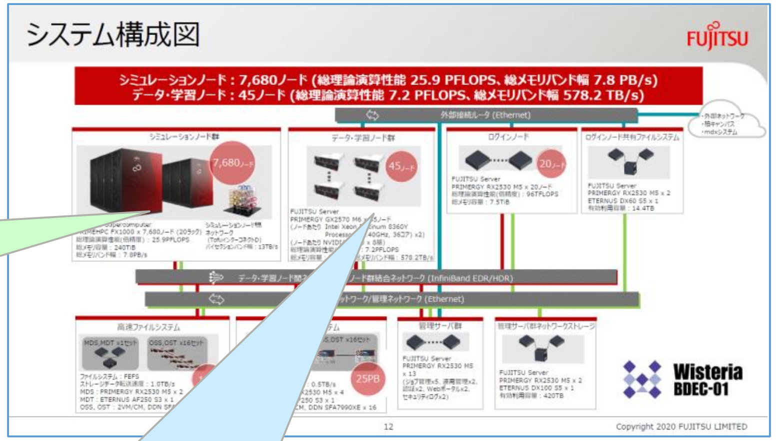
WaitIO-File vs. WaitIO-Socket PingPong Bandwidth Odyssey 1node – Aquarius 1 node (Heterogeneous)

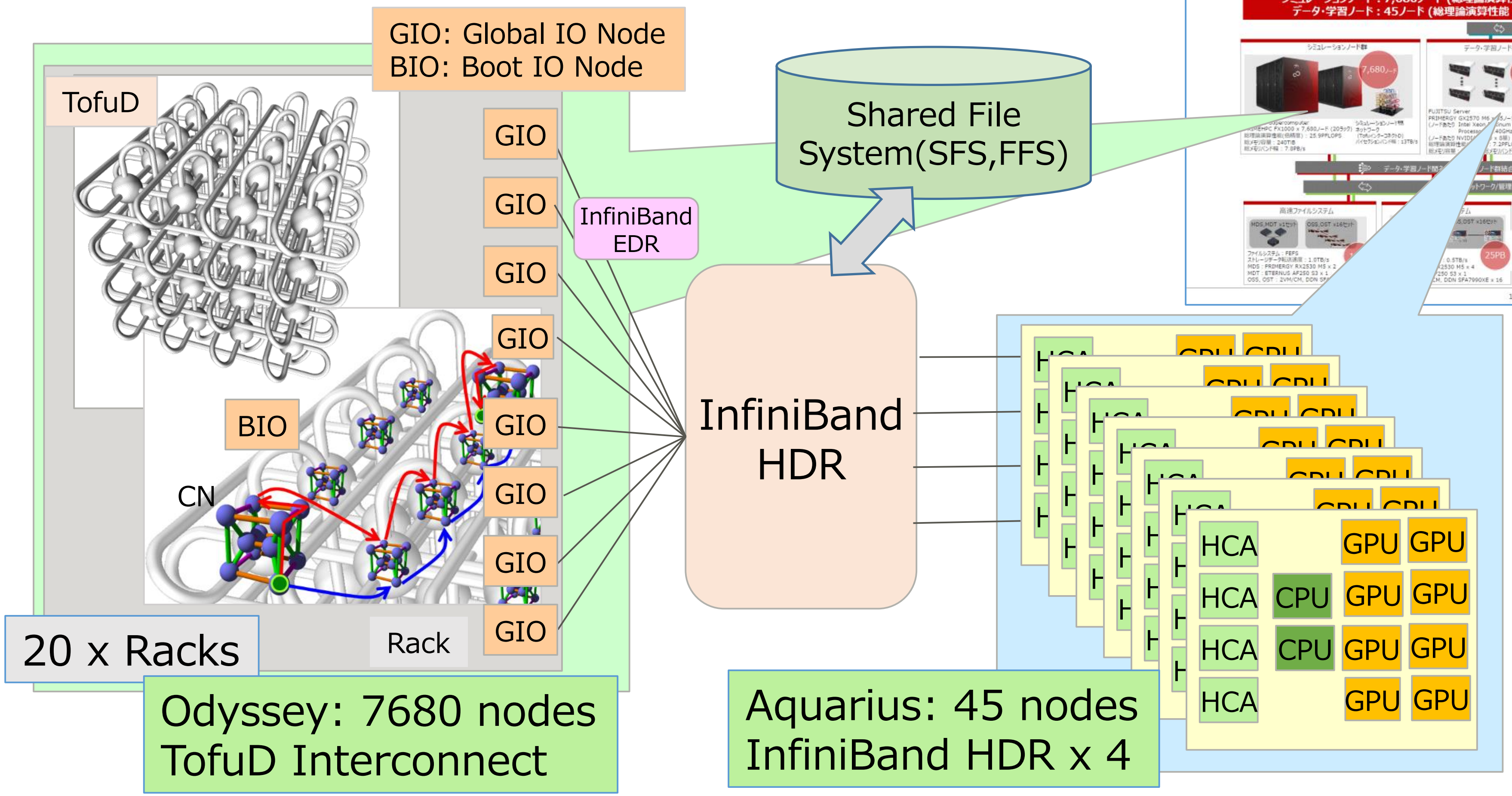
- WaitIO-File: Able to Avoid Odyssey GIO Performance Bottleneck



Wisteria Network Connection

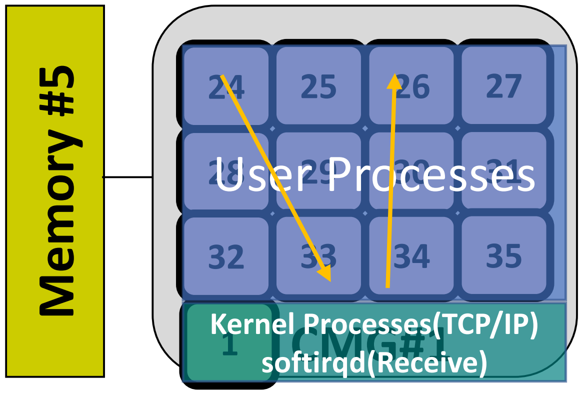
- Odyssey: 8x GIOs with InfiniBand EDR per Rack(384 nodes)
- Aquarius : 4x InfiniBand HDRs per Node



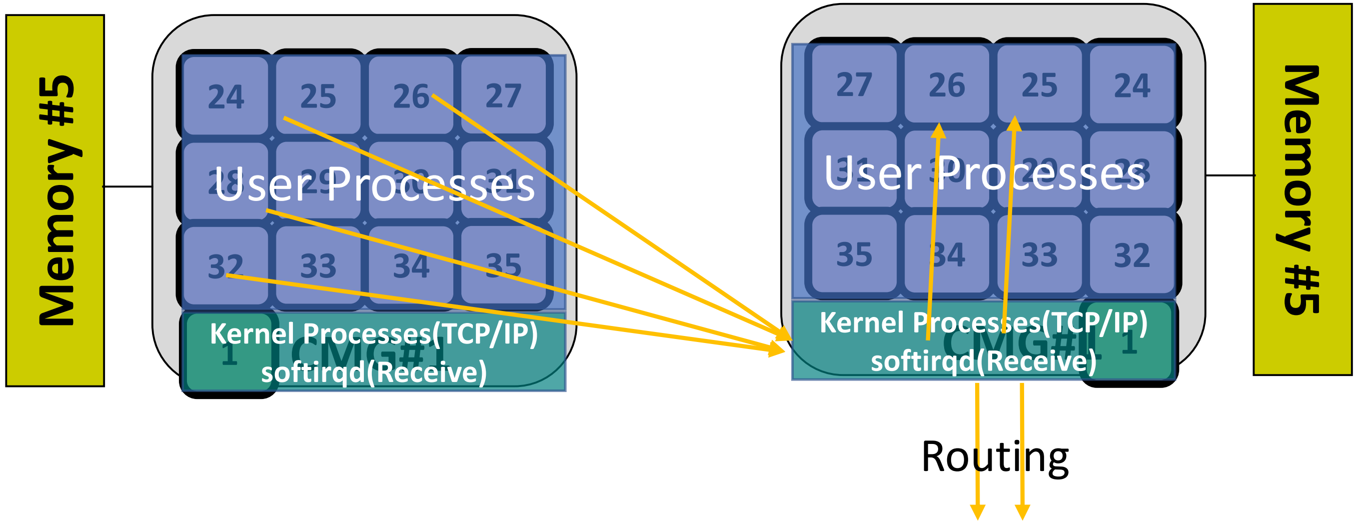


TCP/IP Processing on FX1000 TCS Software

- Intra-node Communication
  - Processed by Computing Core CPU Copy

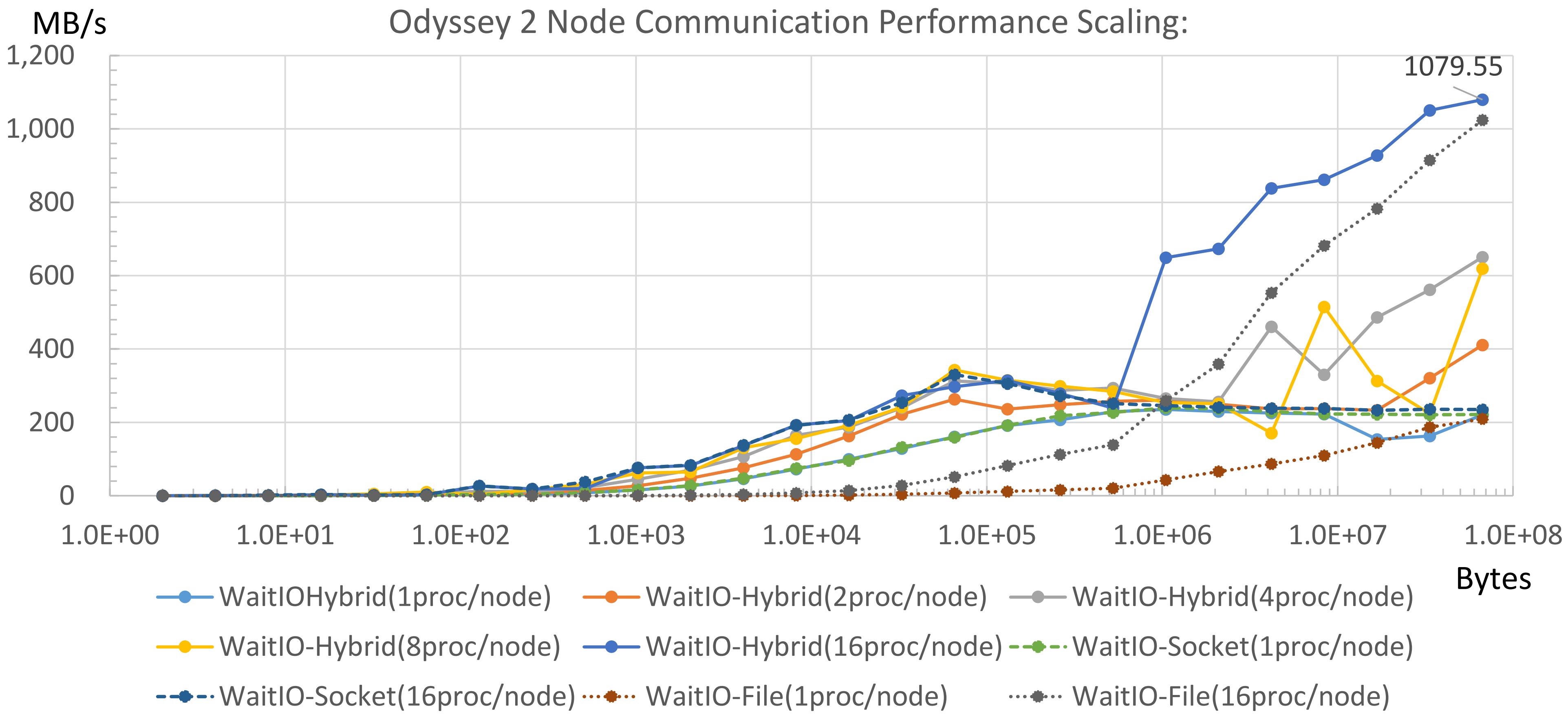


- Inter-node Communication
  - Sender: Processed by Computing Core
  - Receiver: Processed by Assistant Core
  - Routing: Processed by Assistant Core on GIO



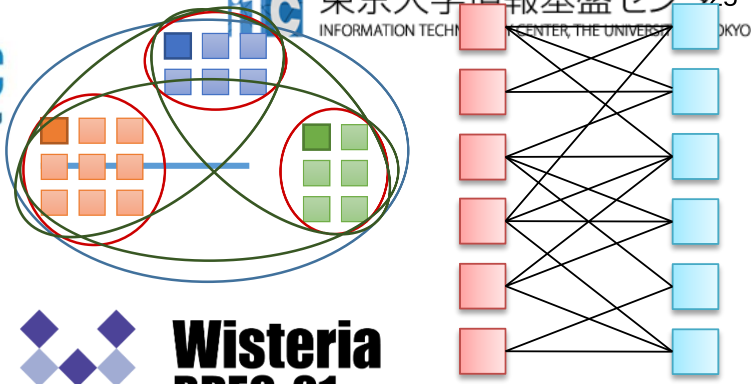
WaitIO-Socket, WaitIO-File vs. WaitIO-Hybrid PingPong Bandwidth Odyssey 2node (Homogeneous)

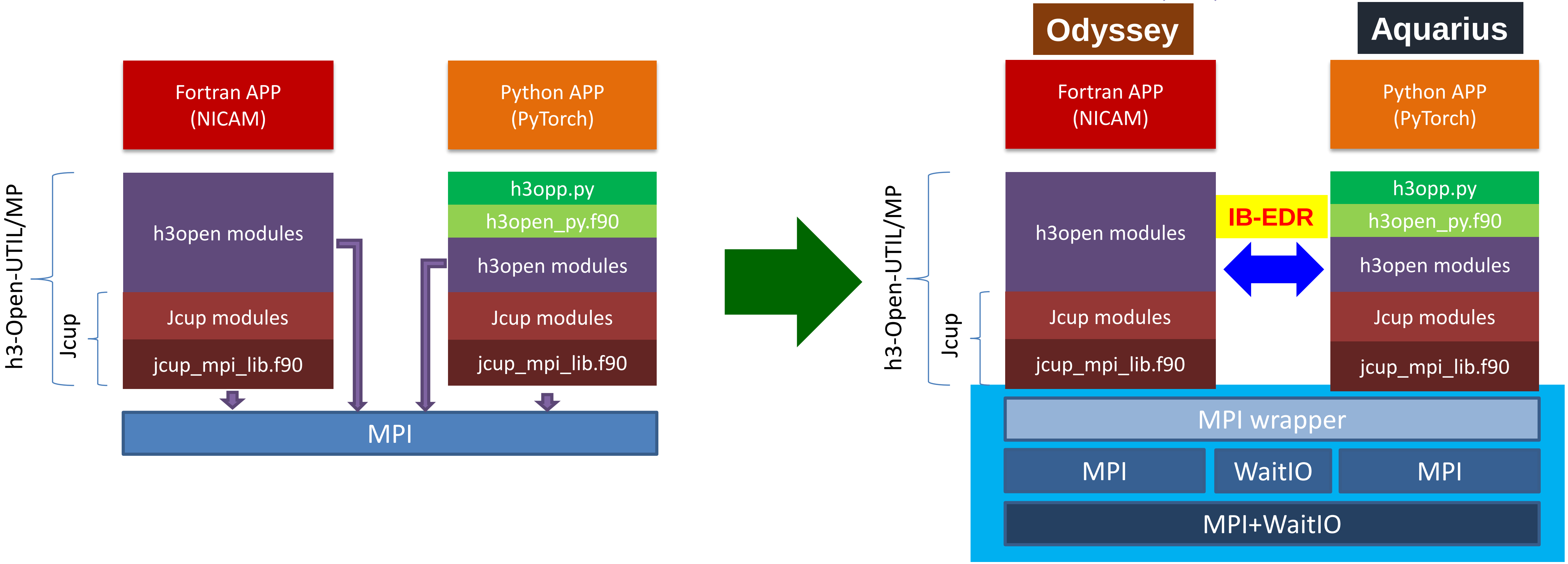
- WaitIO-Hybrid : Achieved 1.1GB/s
  - Rendezvous Protocol is accelerated by WaitIO-Socket



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

Operation Started in June 2022



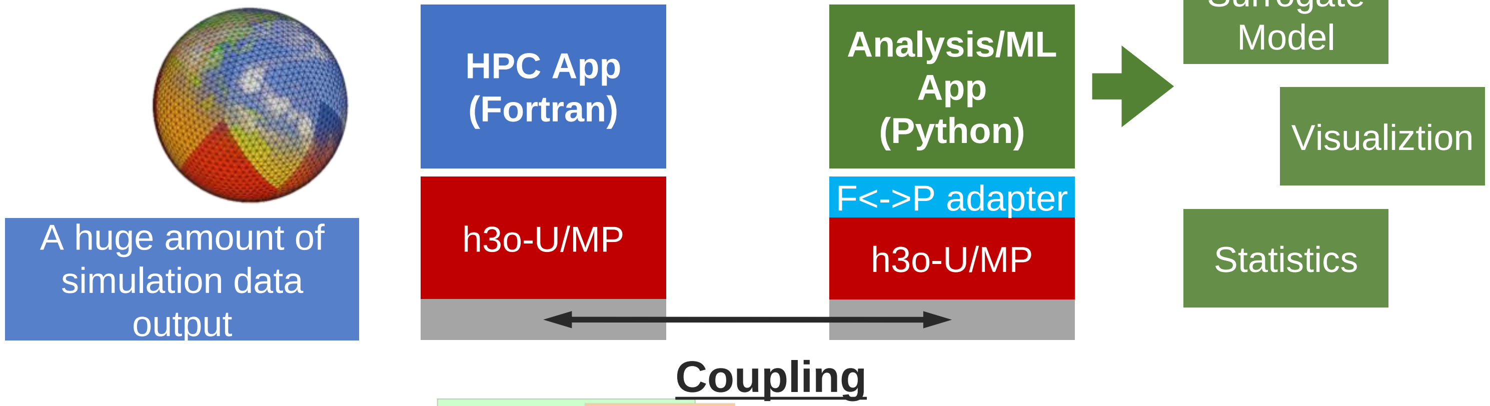


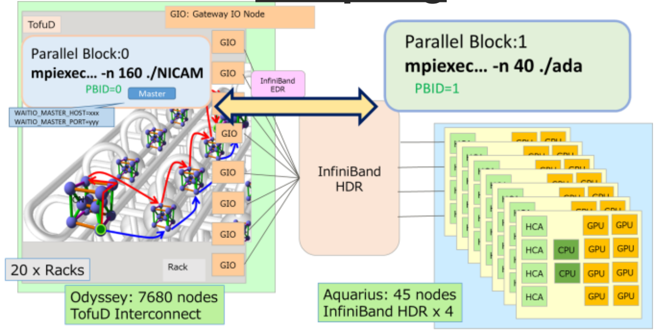
May 2021: Single MPI Environment

June 2022: Coupler＋WaitIO

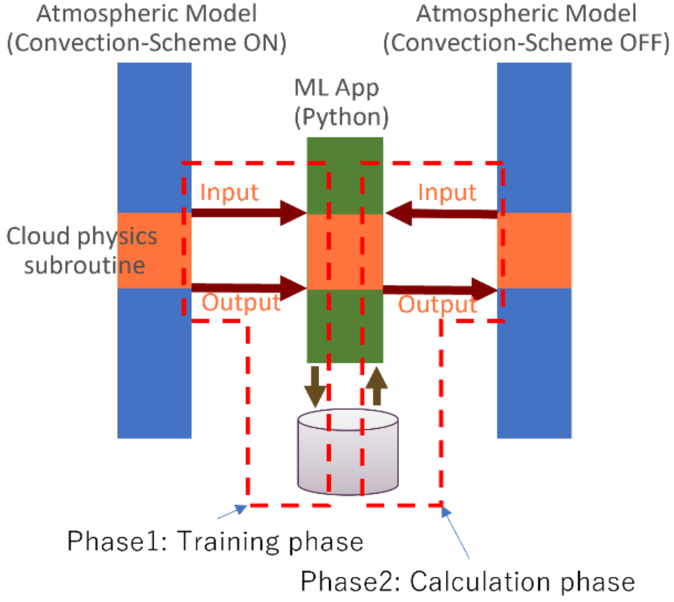
h3-Open-UTIL/MP+h3-Open-SYS/WaitIO Application Performance Evaluation

- 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

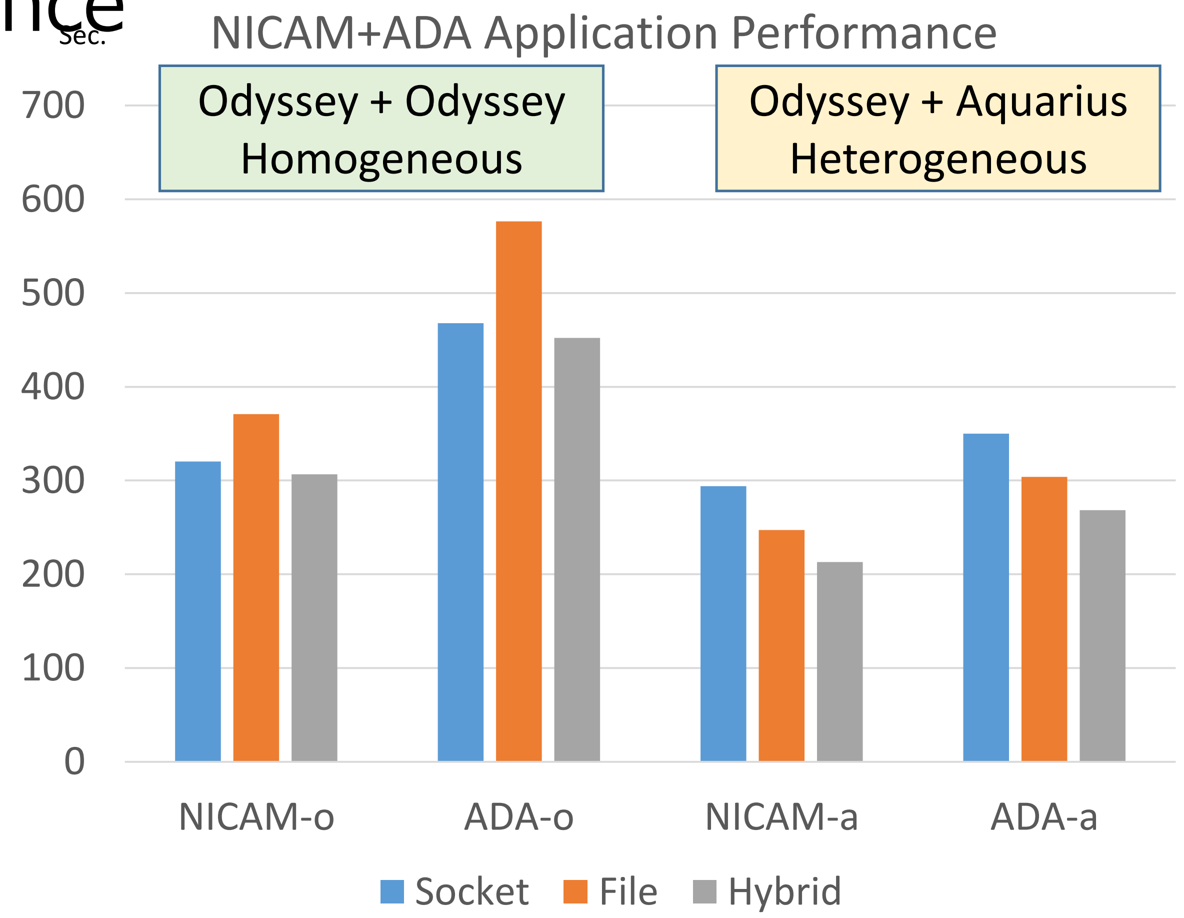


ML Processing Model for Climate Simulation

WaitIO Performance: NICAM+ADA

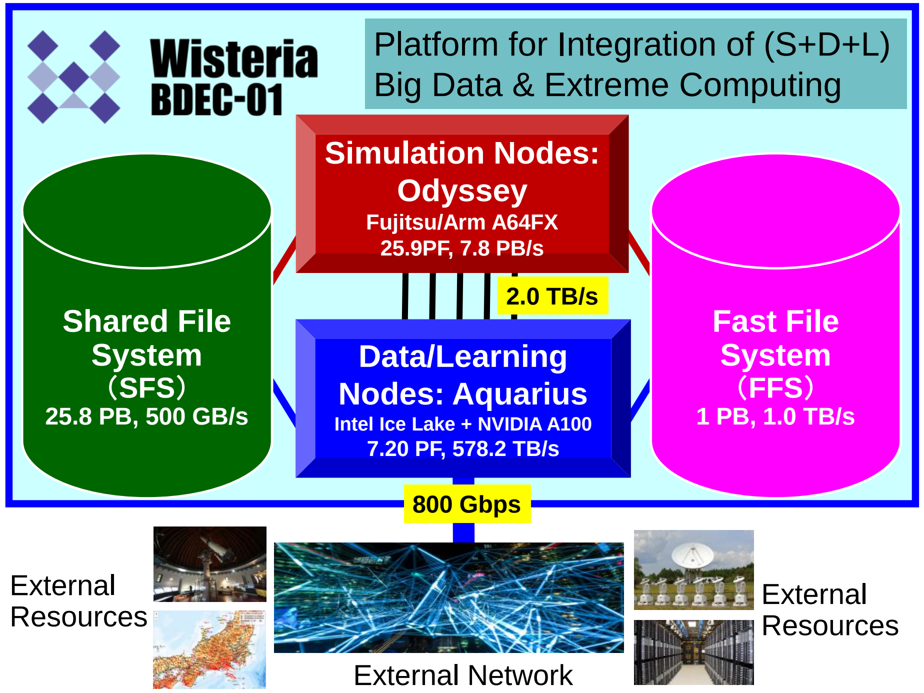
- Comparison of Socket, File, Hybrid
- WaitIO-Hybrid was best performance

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
  - 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?

Related Work

- IMPI, PACX-MPI, GridMPI: Grid computing using TCP/IP. Need to use same implementation
  - IMPI: communication library designed to integrate systems on the Grid.
    - Using Token-based protocol that connects systems called IMPI servers and determines the connection and communication specifications between systems.
    - IMPI also supports secure systems such as firewalls.
  - PACX-MPI: the Global view and Local view to integrate multiple application programs.
    - Transmission/reception processes relay between applications. The relay processes become a performance bottleneck.
  - GridMPI: Supports Grid and Cluster environment in single MPI library: http://aist-itri.github.io/gridmpi/
    - Supporting multiple MPI libraries with vendor MPI and others including IMPI protocol, needs to integrate in MPI level
- MPI standard: MPI_Comm_spawn() and MPI_Comm_connect() /MPI_Comm_accept():
  - Depends on its implementation, ex Open MPI and MPICH, and need to use same implementation and version
- Low-level communication compatible with multiple platforms : UCX, OFED:
  - Supporting Heterogeneous networks heterogeneous processors, and need to use same implementation
- WaitIO-Socket: Works with POSIX-Socket platform for heterogeneous coupling computing
  - Easy to use on existing systems: no needs to install the other software package
  - Scalable: Simple application-to-application direct communication with secure communication such as VPN.
  - Allows the application user to select the MPI process to participate

Evaluation Environments

	Wisteria/Odyssey	Wisteria/Aquarius
FLOPS	25.9 PFLOPS	7.2 PFLOPS
# of Nodes	7,680	45
Interconnect	Tofu-D (6-Dimm Mesh/Torus)	InfiniBand HDR (200Gbps) x 4HCA (Full Fat Tree)
Processor/Socket	A64FX(Armv8.1+SVE), 2.2GHz, 1 socket (48 Core+2 or 4 Assistant Core)	Intel Xeon Platinum 8360Y, 2.4GHz 2 sockets(36+36)
Memory	32 GiB	512GiB
Memory BW/Node	1024GB/s	409.6GB/s
GPU/Node	-	NVIDIA A100 x8
	Fujitsu Compiler, Fujitsu MPI	Intel Compiler, Intel MPI (Open MPI)
	Red Hat Enterprise Linux 8	Red Hat Enterprise Linux 8

	Oakbridge-CX (OBCX)	Oakforest-PACS (OFP)
FLOPS	6.61 PFLOPS	25 PFLOPS
# of Nodes	1368	8208
Interconnect	Omni-Path (100Gbps)	Omni-Path (100Gbps)
Processor/Socket	Intel Xeon Platinum 8280 2.7GHz, 2 socket (28+28)	Intel Xeon Phi 7250 1.4GHz 1 socket (68)
Memory	192 GB	96(DDR4)+16(MCDRAM) GB
Memory BW/Node	281.6GB/s	115(DDR4)+490(MCDRAM) GB/s
Compiler, MPI	Intel Compiler, Intel MPI (Open MPI	Intel Compiler, Intel MPI (Open MPI
Operating System	Red Hat Enterprise Linux 7, CentOS 7	Red Hat Enterprise Linux 7, CentOS 7

PingPong 1/2 RTT

Variable proc-16 Node

usec(8B)	1/2 RTT(Intra)	1/2 RTT(Inter)
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

- RTT depends on CPU performance: Intranode, Internode
  - Xeon CPU(Aquarius, OBCX): Around 6 usec
  - A64FX、Xeon Phi: 37 – 83.7 usec

PingPong Bandwidth Performance

Variable proc-16 Node

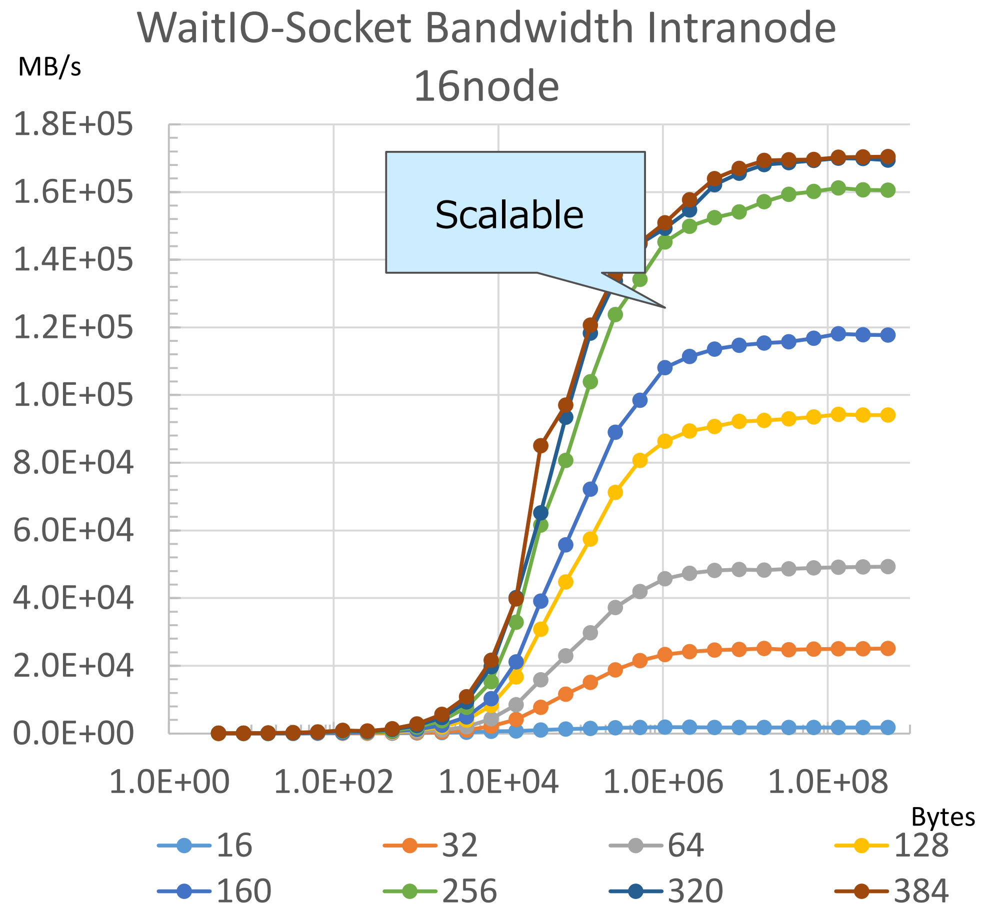
PingPong BW GB/s	Intra	Inter
Odyssey	21.3	0.35 (Tofu-D)
Aquarius	259.7	27.1 (IB)
OBCX	45.7	9.3 (OPA)
OFP	8.2	1.12 (OPA)

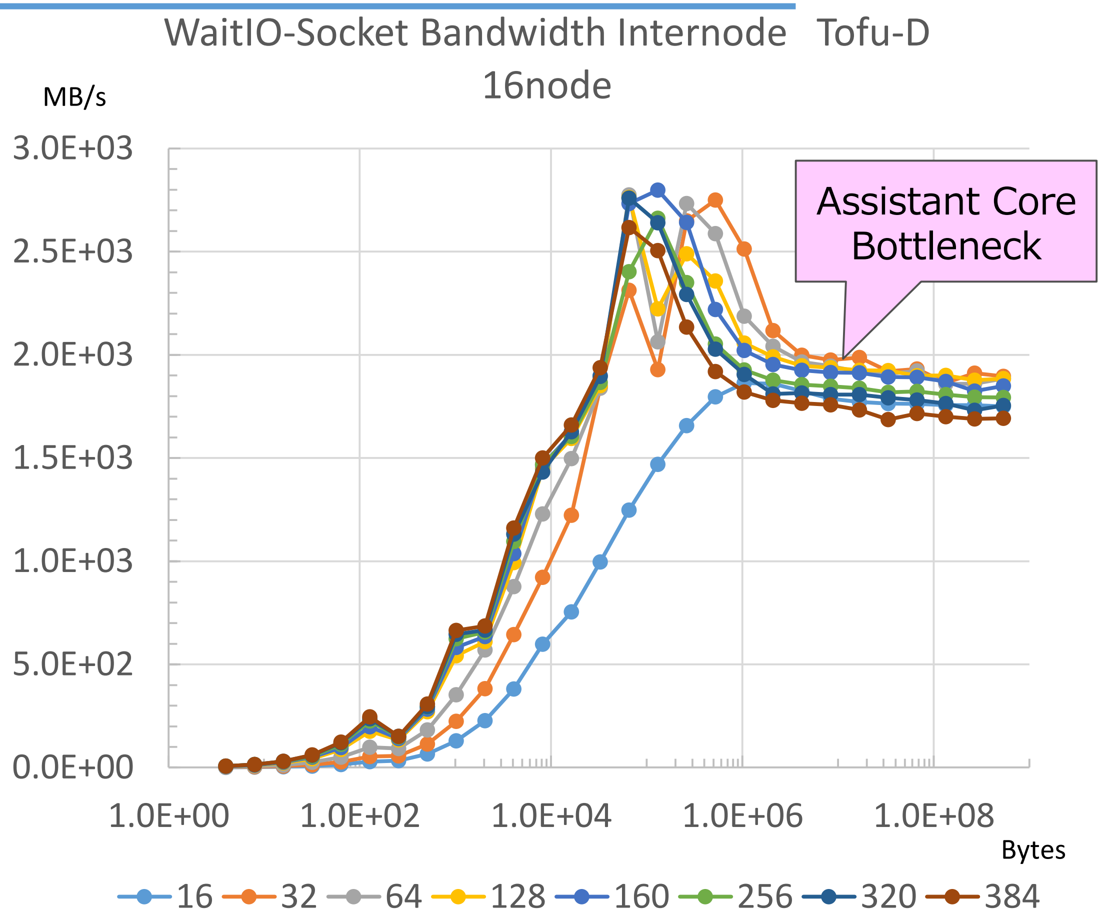
good

bad

- Depends on the balance of CPU and Interconnect performance
  - Odyssey, OFP: CPU performance is not enough
  - Aquarius: Well balanced

WaitIO-Socket: PingPong Bandwidth Performance on Odyssey 16node

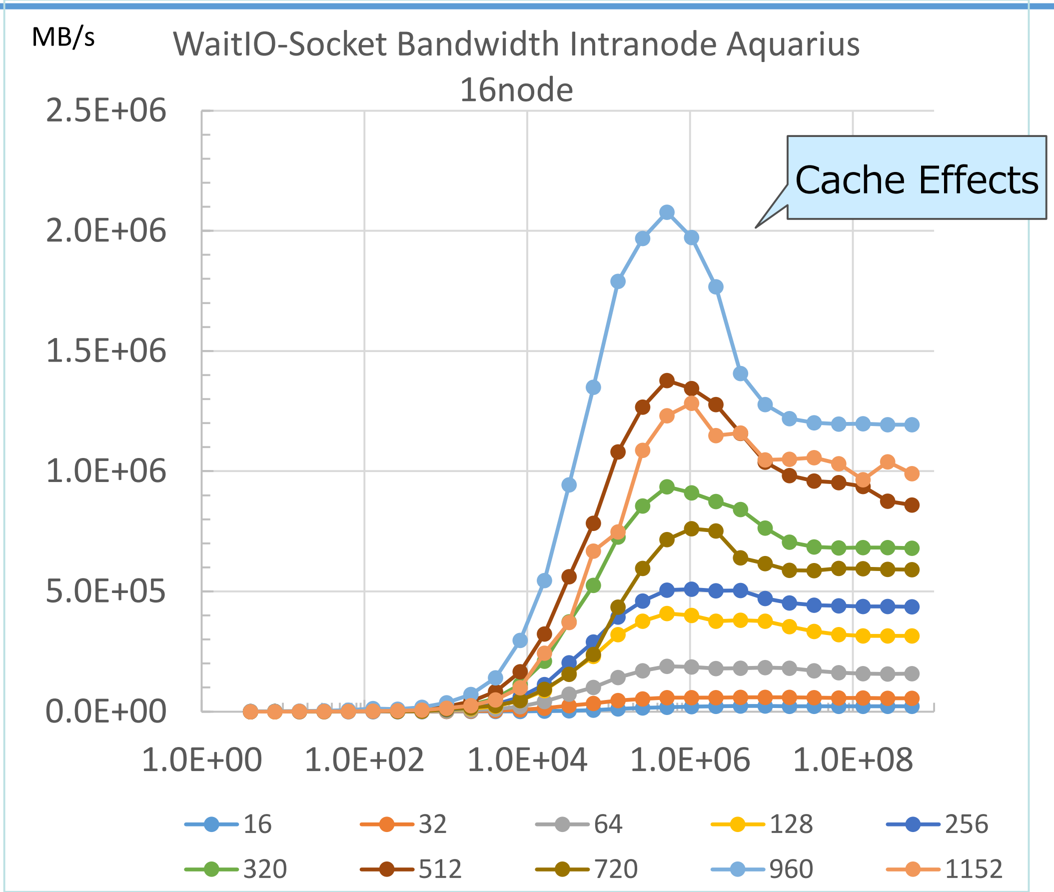


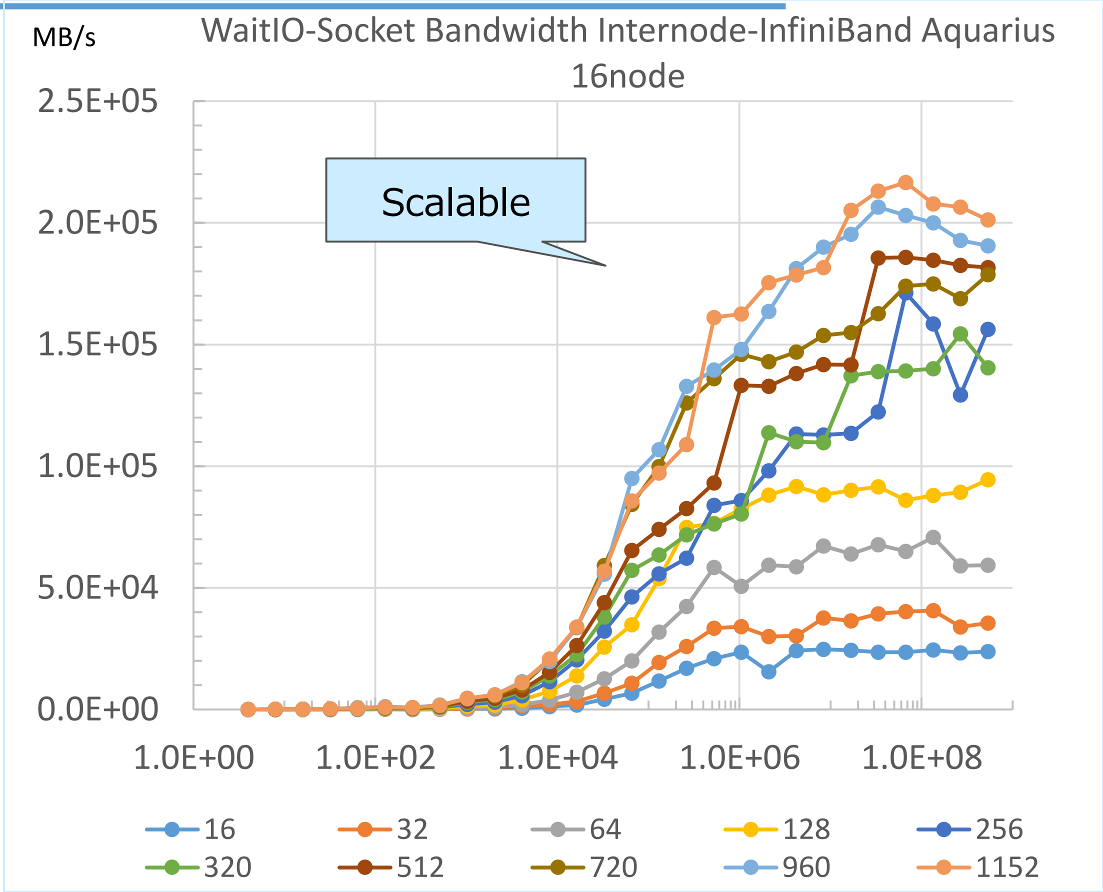


Variable proc-16 Node

- Intranode: 170.4 GB/s (21.3 GB/s /node)
- Internode: 2.80 GB/s (0.35 GB/s /node)

WaitIO-Socket: PingPong Bandwidth Performance on Aquarius 16node



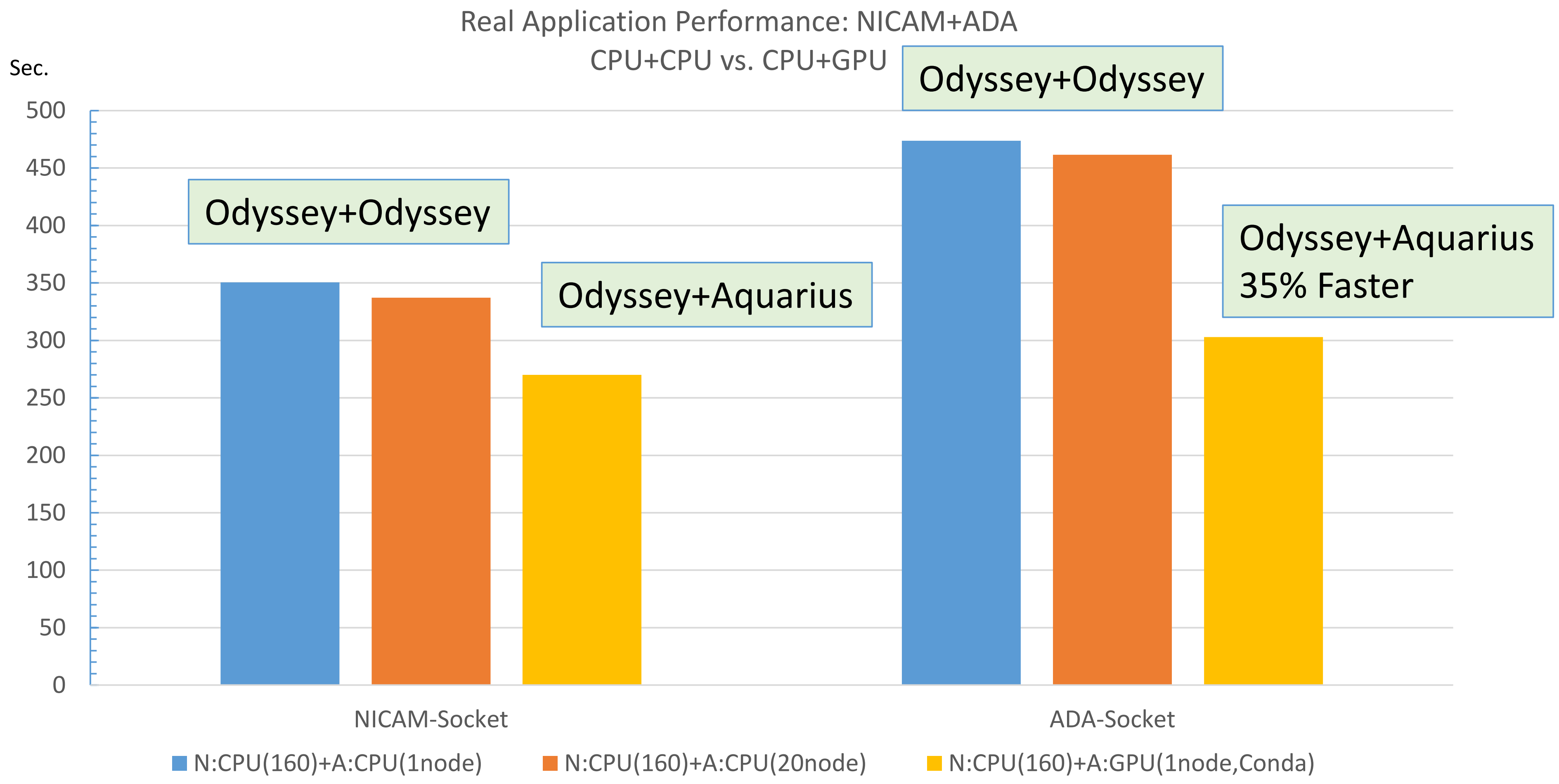


- Intranode: 2,077.6 GB/s (259.7 GB/s /node)
- Internode: 216.6 GB/s (27.1 GB/s /node)

Variable proc-16 Node

NICAM+ADA: Performance Results

- CPU+GPU is 35% faster than CPU+CPU



WaitIO-Hybrid Communication Performance: Odyssey 2node

- WaitIO-Hybrid : Achieved 1.1GB/s
  - Accelerating Rendezvous performance by WaitIO-Socket