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

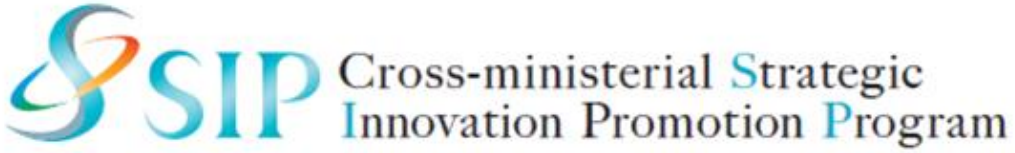

Generic TCP/IP and File-based Communication Library for Heterogeneous Parallel Computer

Hiroya Matsuba
Information Technology Center
The University of Tokyo

International Workshop on the Integration of (Simulation + Data + Learning) : Towards Society h3-Open-BDEC, December 3, 2021 (Online)

Acknowledgements

- JSPS Grant-in-Aid for Scientific Research (S) (19H05662)
- New Energy & Industrial Technology Development Organization (NEDO): Cross-ministerial Strategic Innovation Promotion Program (SIP): Big-Data and AI-Enabled Cyberspace Technologies
- Joint Usage/Research Center for Interdisciplinary Large-scale Information Infrastructures (JHPCN)
  - jh210022-MDH

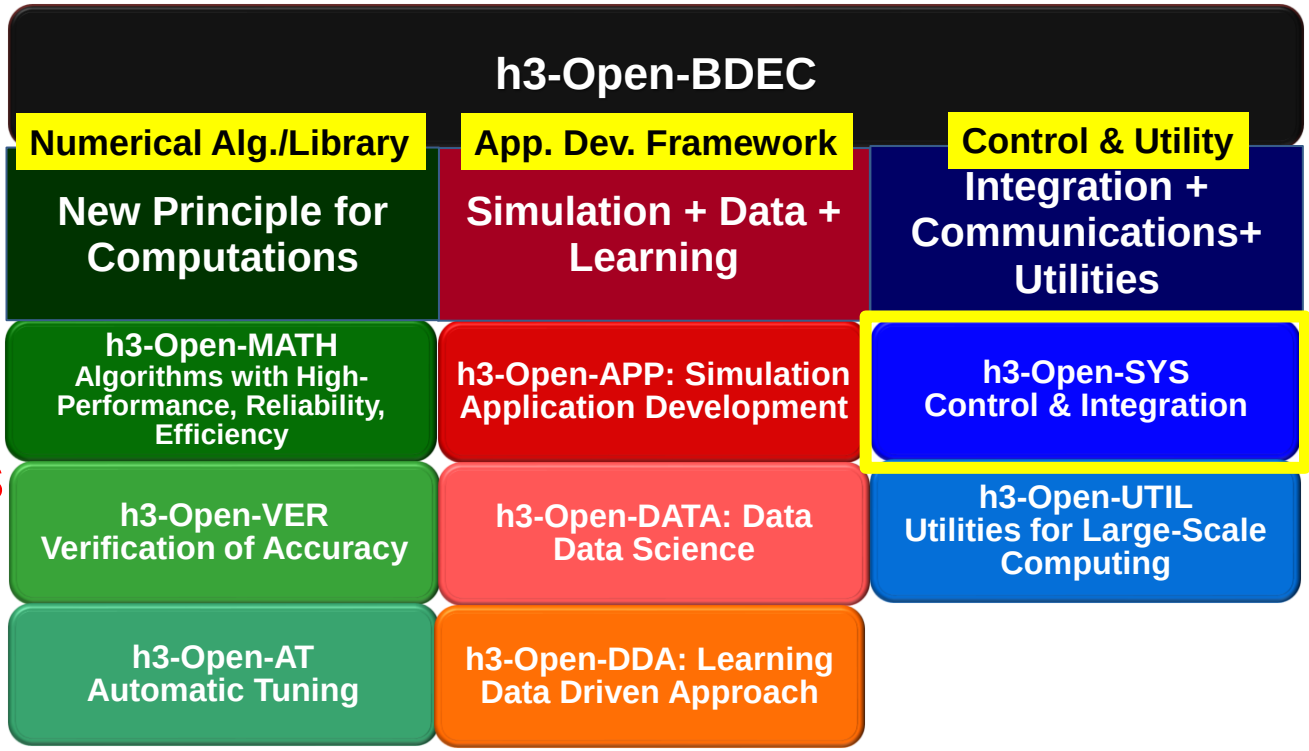


Contributors

- **Shinji Sumimoto (Fujitsu)**
  - Most of implementation and evaluation work in this talk was done by Shinji Sumimoto
- Takashi Arakawa (RIST)
- Kengo Nakajima (University of Tokyo)
- Yoshio Sakaguchi (Fujitsu)
- Hisashi Yashiro (NIES)
- Toshihiro Hanawa (University of Tokyo)
- Hiroya Matsuba (University of Tokyo)

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

- Overview
  - 5-year project supported by Japanese Government (JSPS) since 2019
  - Leading-PI: Kengo Nakajima (The University of Tokyo)
  - Total Budget: 1.41M USD
- Two Innovations
  - New Principles for Numerical Analysis by Adaptive Precision, Automatic Tuning & Accuracy Verification
  - Hierarchical Data Driven Approach (*h*DDA) based on Machine Learning

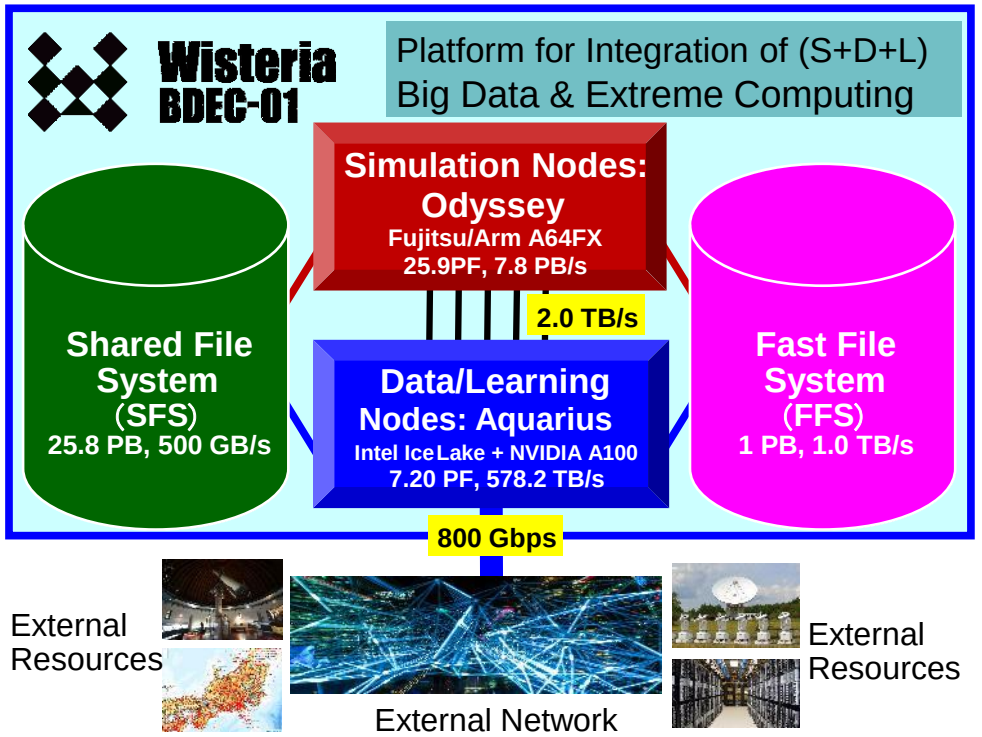


h3-Open-SYS/WaitIO-Socket (WaitIO)

WaitIO is a communication library for communication between two different systems

Wisteria/BDEC-01（Fujitsu）
- Simulation Nodes (Oddysey): A64FX (#17)
- Data/Learning Nodes (Aquarius) (#106)
- 33.1 PF, Operation started on May 14, 2021
- Platform for Integration of “Simulation+Data+Learning (S+D+L)”
- Innovative Software Platform “h3-Open-BDEC” supported by Janapese Government (JSPS Grant-in-Aid for Scientific Res. (S) FY.2019-2023)

Simulation+Data+Learning software development
- 2.0TB/s InfiniBand EDR is physically available
- No MPI communication between Odyssey and Aquarius



Simulation Nodes（Odyssey）

Data/Learning Nodes（Aquarius）

Simulation Nodes
Odyssey
25.9 PF, 7.8 PB/s

Fast File System (FFS)
1.0 PB, 1.0 TB/s

Shared File System (SFS)
25.8 PB, 0.50 TB/s

Data/Learning Nodes
Aquarius
7.20 PF, 578.2 TB/s

Simulation Codes

Optimized Models & Parameters

Simulation Nodes Odyssey

Wisteria/BDEC-01

Results

Observation Data

Machine Learning, DDA

Data/Learning Nodes, Aquarius

Data Assimilation Data Analysis

Server, Storage, DB, Sensors, etc.

External Network

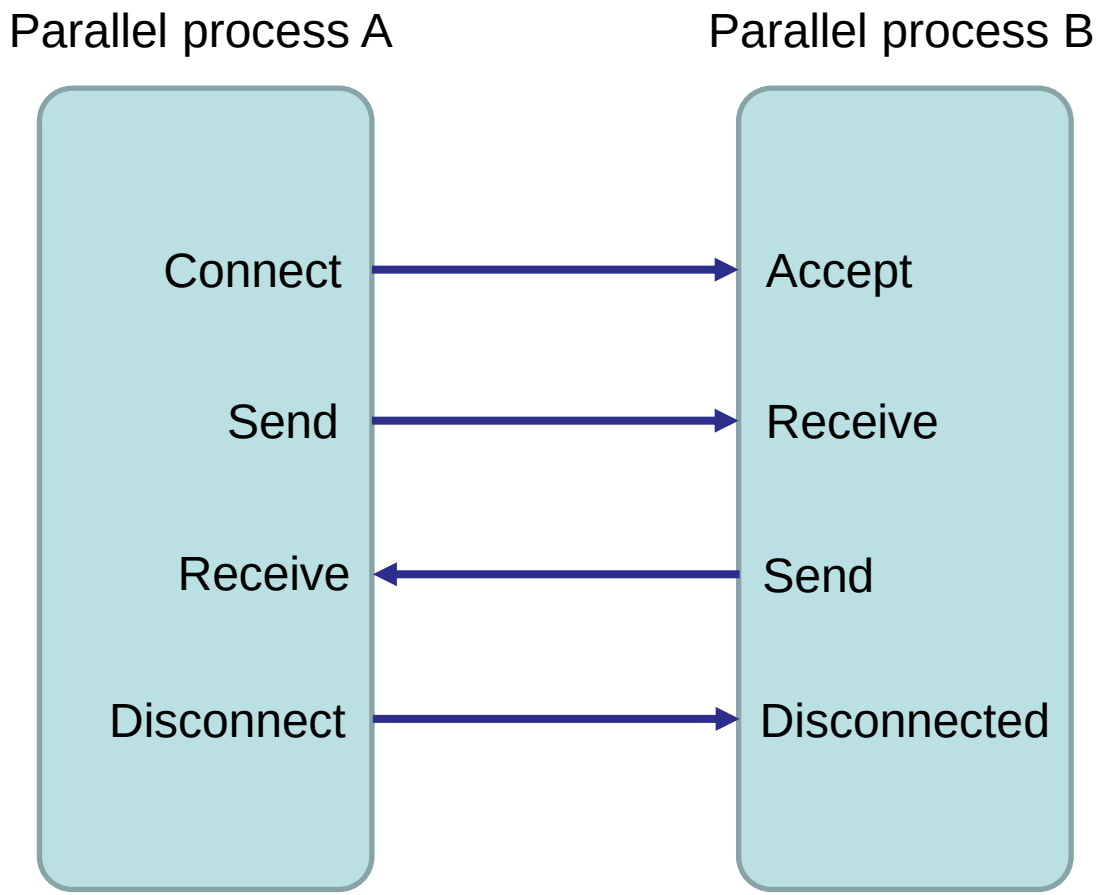
External Resources

Concept of WaitIO

WaitIO is a virtual communication channel between two parallel processes.

Conceptual View

WaitIO enables a parallel process to establish a communication channel to another parallel process.

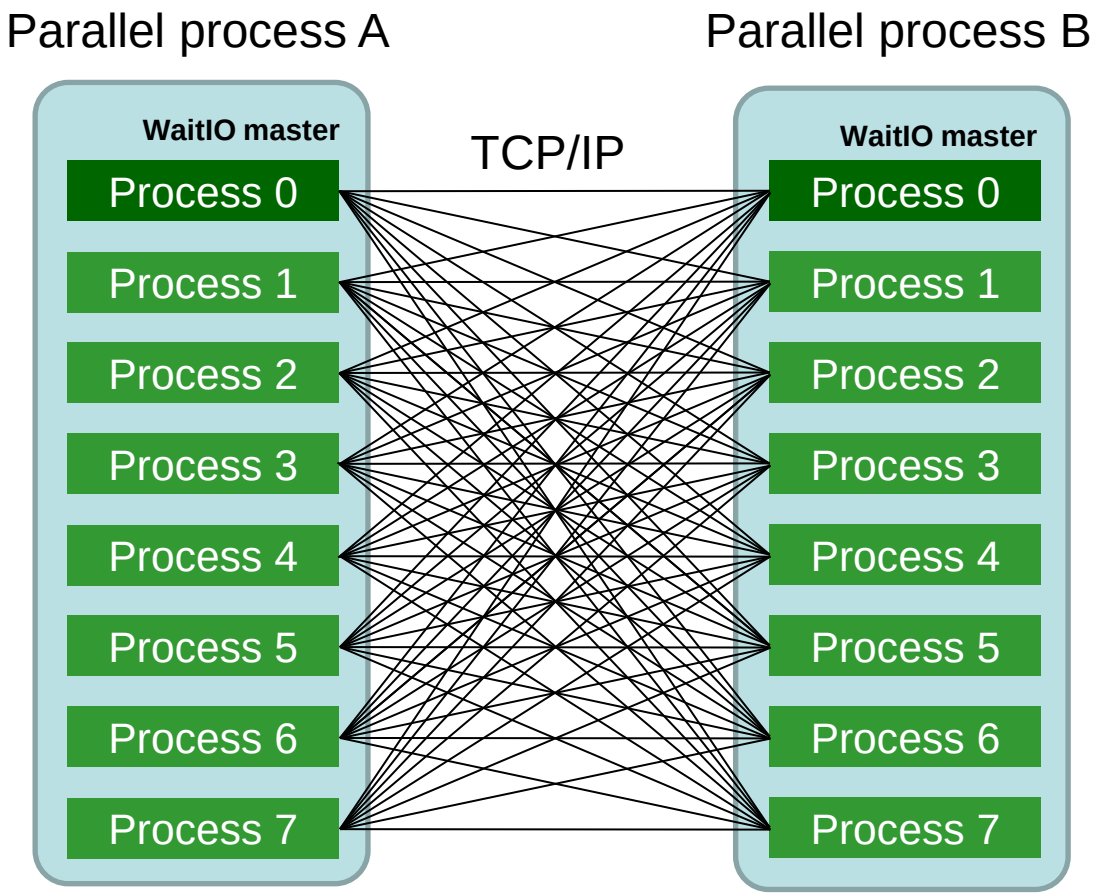


Concept of WaitIO

WaitIO is a virtual communication channel between two parallel processes.

Implementation

WaitIO is a communication library that handles massive number of TCP/IP connection between each pair of processes.



Features of WaitIO

Communication between parallel processes are possible without WaitIO, but WaitIO has the following advantages over the existing method.

Little dependency on hardware/software environment

WaitIO can work anywhere as long as communicating processes are IP reachable.

- MPI has features to connect two parallel processes (MPI_Comm_spawn, MPI_Comm_connect, MPI_Comm_accept) but they can be used only by MPI parallel programs.
- MPI_Comm_connect usually requires two parallel processes using the same MPI implementations.
- A WaitIO variant that does not require IP reachability is also planned (using shared file system as a communication channel)

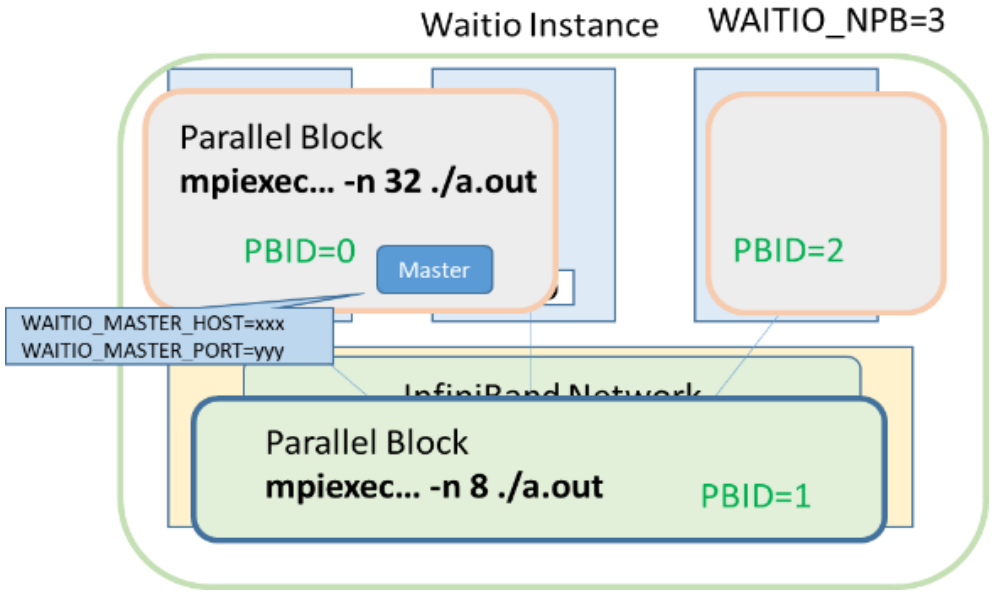
Easy to use APIs

WaitIO provides APIs that is similar to MPI.

- Users who are familiar with MPI can easily use WaitIO

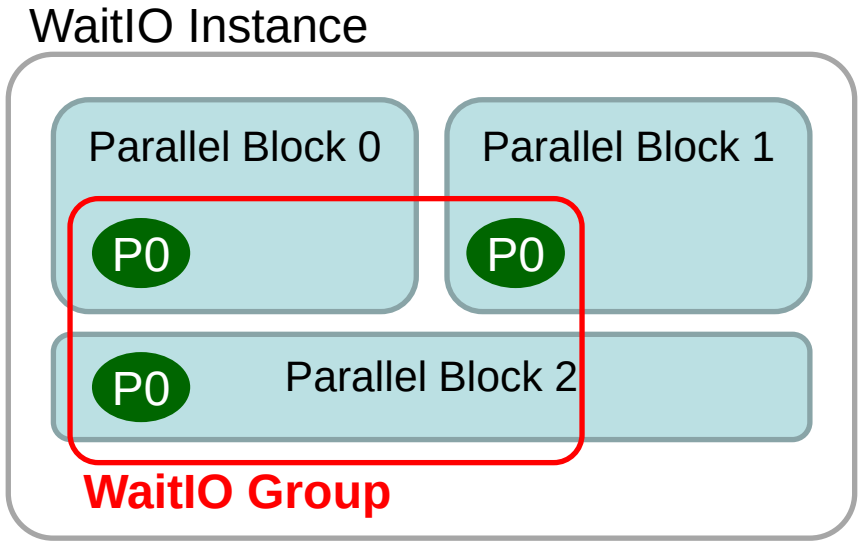
How WaitIO works

- WaitIO Instance
  - Entire WaitIO world
- Parallel Block
  - Parallel block means a group of processes that are members of a single parallel process
    - Usually created with mpiexec command
  - PBID: Integer
    - Identifier of parallel block in a WaitIO instance
    - This should be set with an environment variable WAITIO_PBID
    - Total number of PB in the WaitIO instance must be set by the environment variable WAITIO_NPB.
  - PB MASTER：PBID=0
    - Contact point from a new parallel process
    - All the process must have the same WAITIO_MASTER_HOST and WAITIO_MASTER_PORT environment variable



How WaitIO works

- WaitIO Group
  - Set of processes to communicate each other
  - Selected from each Parallel Block
  - Similar concept with MPI communicator



WaitIO API

- WaitIO provides a similar API with MPI
  - Limited to minimum set for simplicity
  - Group ceration, isend(), irecv(), wait() functions are provided

WatiIO API	Description
waitio_isend	Non-Blocking send
waitio_irecv	Non-Blocking receive
waitio_wait	Wait for send/receive complete
waitio_init	Initialize WaitIO
waitio_get_nprocs	Get number of process in PB
waitio_create_group waitio_create_group_wranks	Create WaitIO group
waitio_group_rank	Get rank number in WaitIO group
waitio_group_size	Get the size of WaitIO group
waitio_pb_size	Get the number of parallel blocks
waitio_pb_rank	Get the rank number in parallel block

Performance evaluation

	Wisteria/Odyssey	Wisteria/Aquarius
Peak performance	25.9 PFLOPS	7.2 PFLOPS
Number of nodes	7,680	45
Interconnection	Tofu-D (6D mesh/torus)	InfiniBand HDR (200Gbps) x 4HCA (Full Fat Tree)
Processor/Node	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/Node	32 GB	512GiB
Memory BW/Node	1024GB/s	409.6GB/s
GPU/Node	-	NVIDIA A100 x8
Compiler、MPI	Fujitsu Compiler, Fujitsu MPI	Intel Compiler, Intel MPI, (Open MPI)
OS	Red Hat Enterprise Linux 8	Red Hat Enterprise Linux 8

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

PingPong 1/2 RTT

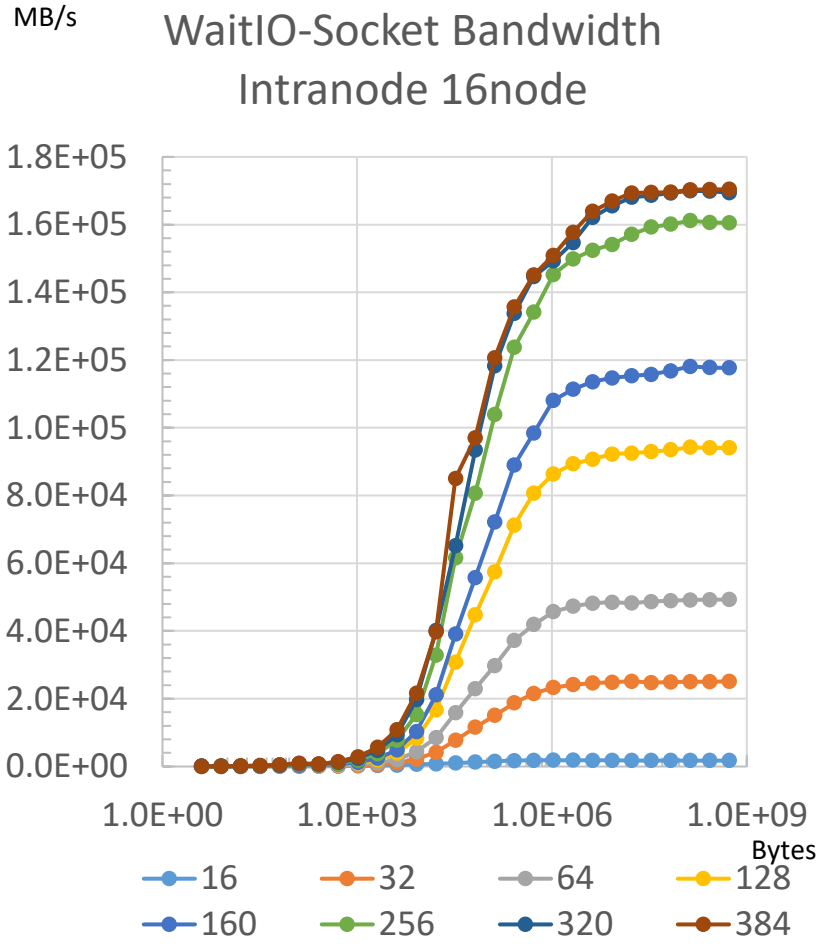
usec(8B)	1/2 RTT(Intra-node)	1/2 RTT(Inter-node)
Odyssey	26.8	37.1 (Tofu-D)
Aquarius	6.57	21.1 (IB)
OBCX	6.85	14.5 (OPA)
OFP	49.7	83.7 (OPA)

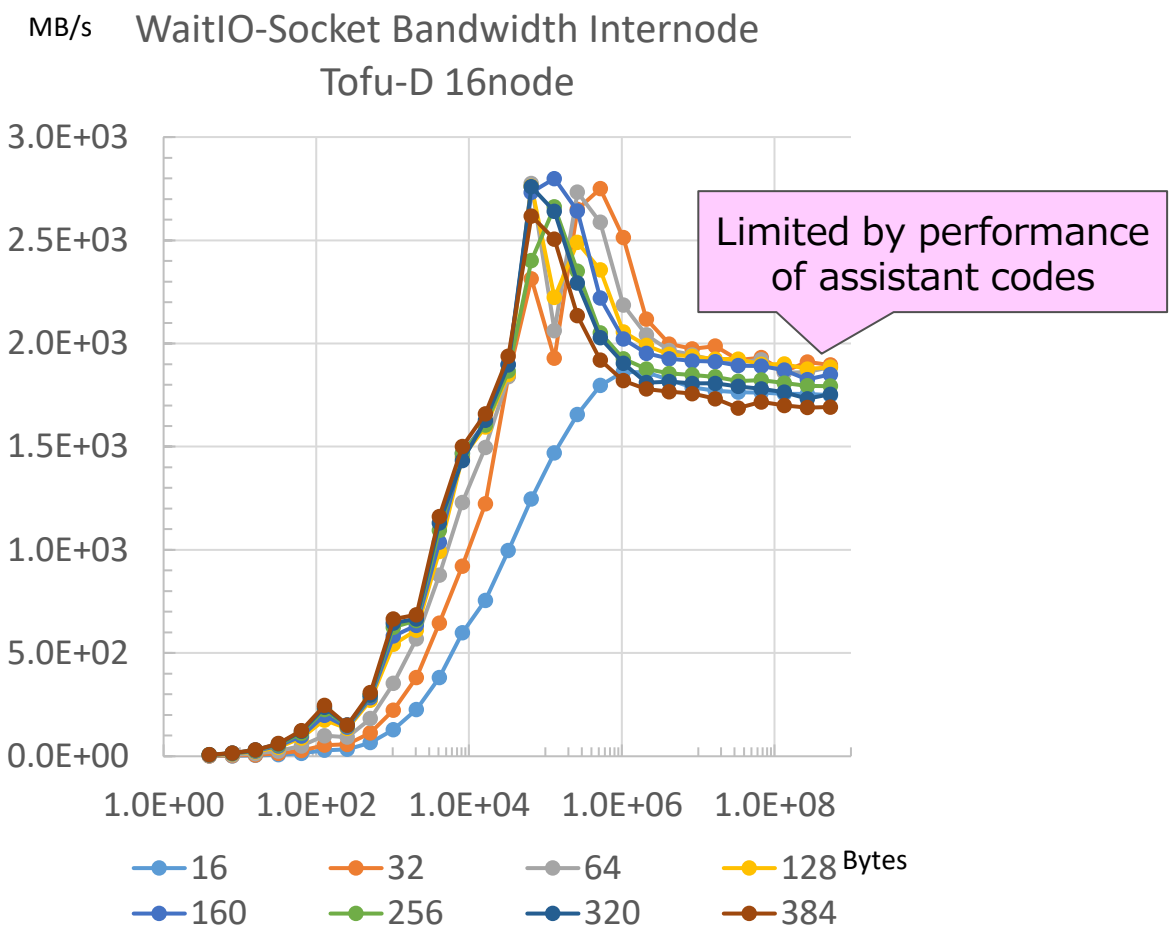
good

bad

- CPU performance is a limiting factor
  - Xeon nodes (Aquarius, OBCX) achieves 6 usec
  - A64FX, Xeon Phi nodes are slower

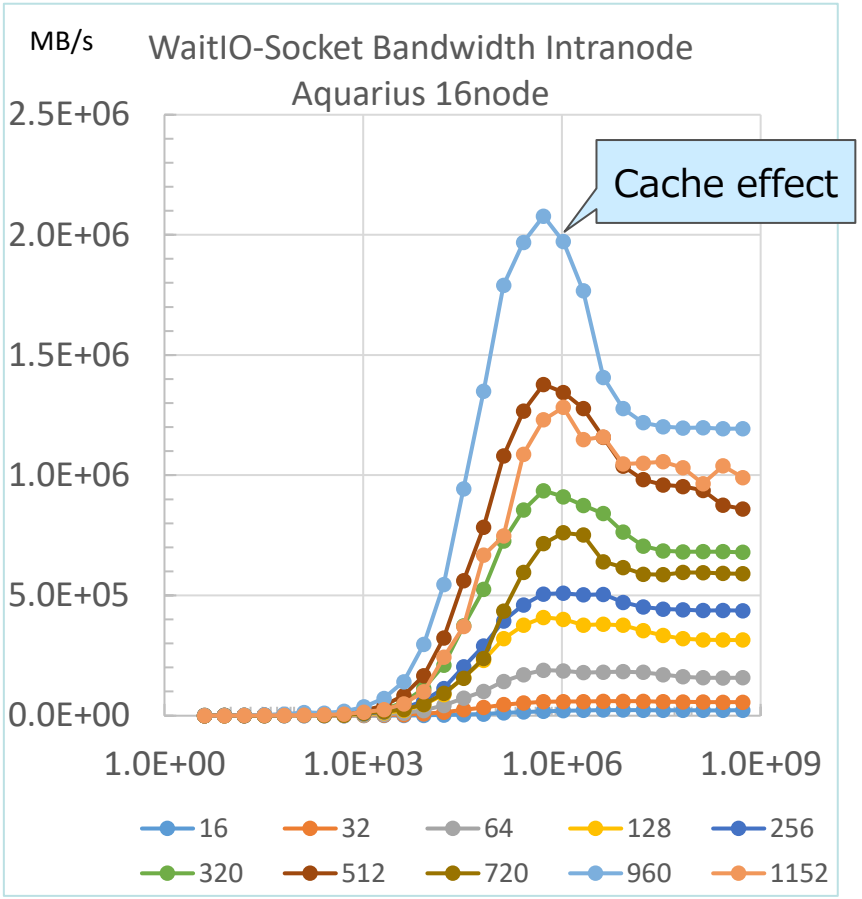
Bandwidth (Wisteria/Odyssey)

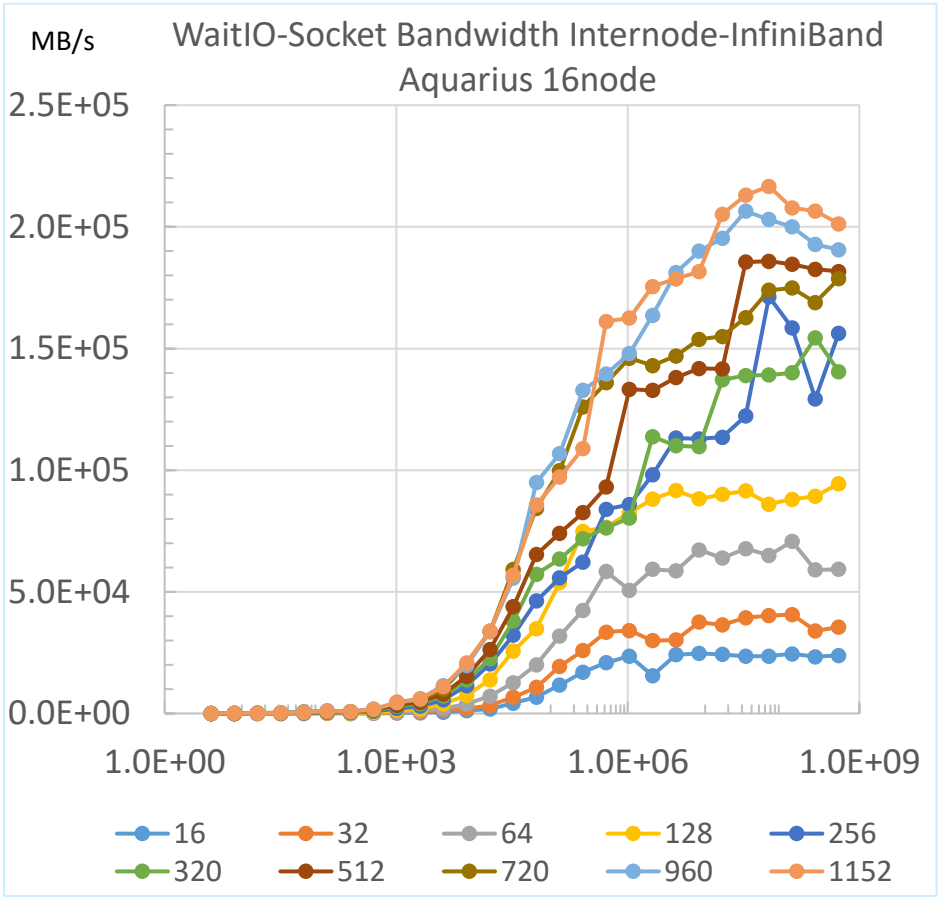




- Intra-node: 170.4 GB/s (21.3 GB/s /node)
- Inter-node: 2.80 GB/s (0.35 GB/s /node)

Bandwidth (Wisteria/Aquarius)

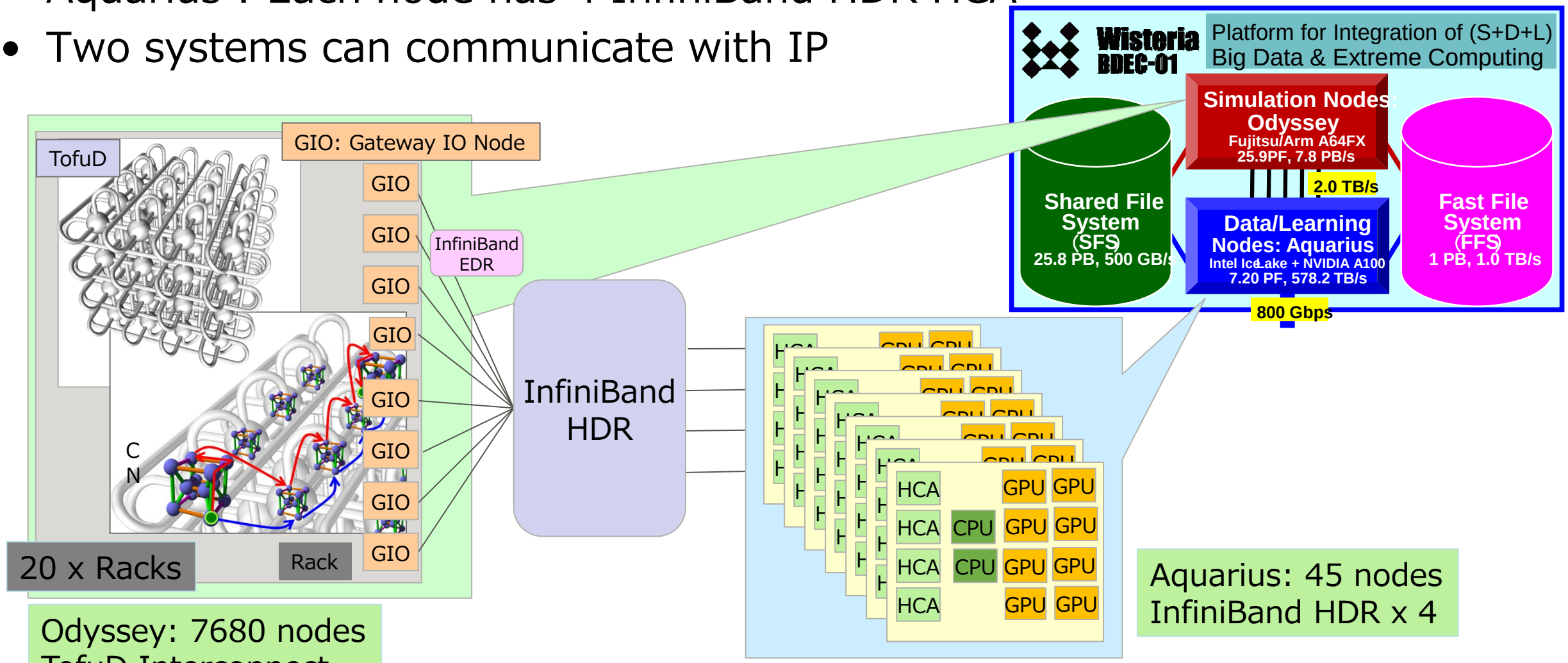




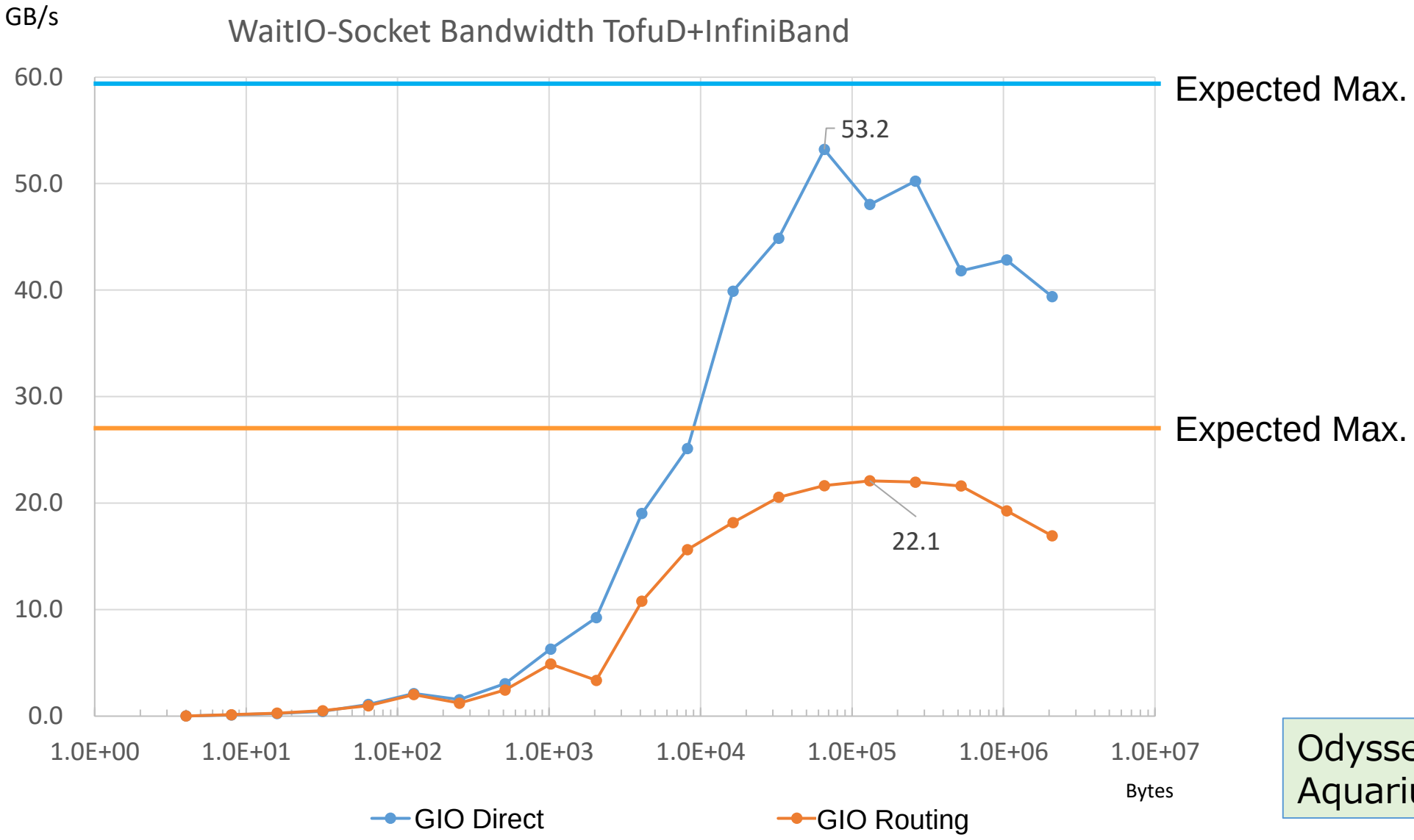
- Intra-node: 2,077.6 GB/s (259.7 GB/s /node)
- Inter-node: 216.6 GB/s (27.1 GB/s /node)

Odyssey - Aquarius Communication

- Odyssey: Each rack (384 nodes) has 8 GIO with InfiniBand EDR
- Aquarius : Each node has 4 InfiniBand HDR HCA
- Two systems can communicate with IP



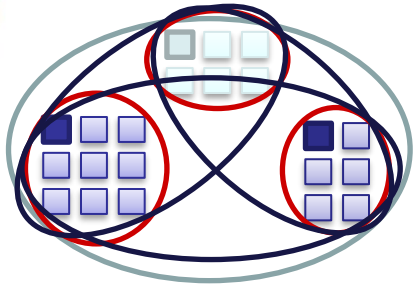
Odyssey - Aquarius Communication

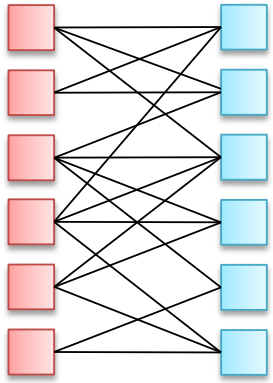


Odyssey:24proc/Node
Aquarius:72proc/Node

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

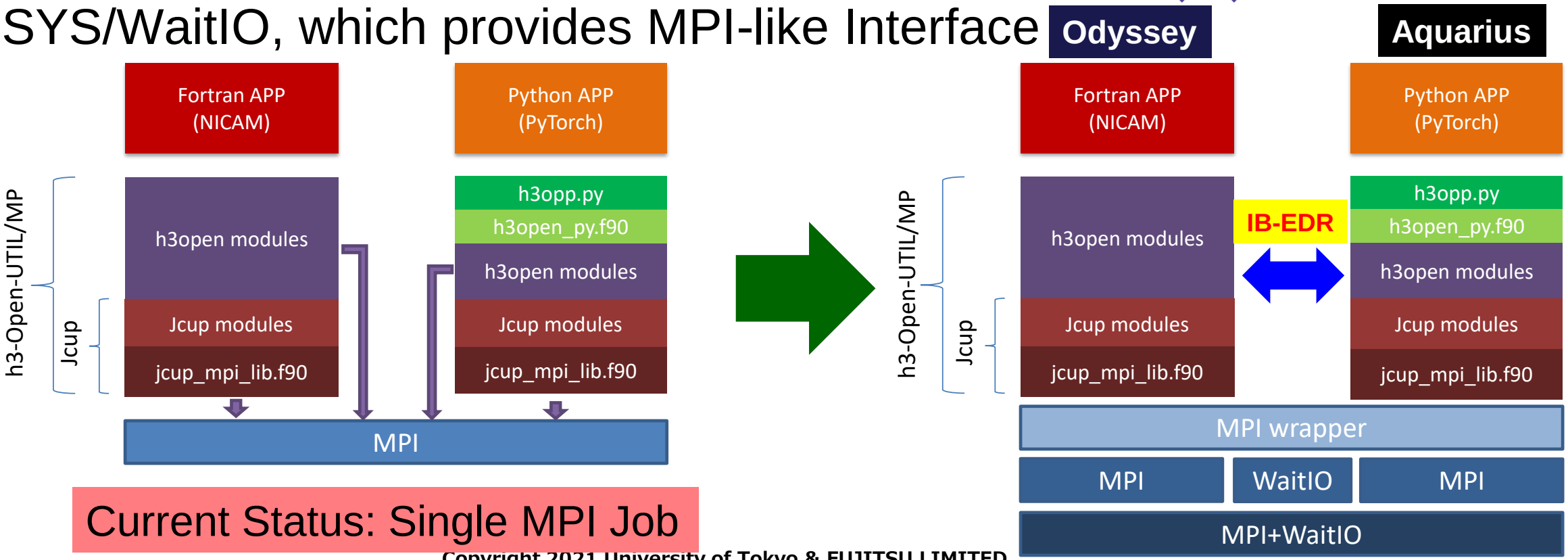
Hierarchical, Hybrid, Heterogeneous
h3-Open-BDEC
Big Data & Extreme Computing





- Current Statue: Single MPI Job
- Direct Communication between Odyssey-Aquarius through IB-EDR by h3-Open-SYS/WaitIO, which provides MPI-like Interface

Wisteria
BDEC-01



Current Status: Single MPI Job

Conclusion

- WaitIO is a communication library that provides a virtual connection from a parallel program to another parallel process.
- TCP/IP connection is established between actual processes. No special hardware is required for using WaitIO.
- WaitIO provides a convenient way for realizing communication between simulation and data analytics software.

backup

WaitIO-Socketの実現要件

- 異種システム上で動作可能:
  - 複数の異なるノード群上の複数アプリケーションでデータ交換を実現
  - ハードウェア、ソフトウェアが異なるノード群上で動作
- 複数のアプリケーション群を手間少なく結合可能:
  - 既存ソフトウェア改変は最小限
  - 結合部分のみのコード追加で実現可能
  - 他のシステムソフトウェアと共存可能
  - プログラミングが容易
- システム最大規模でアプリケーション実行可能:
  - 数万プロセス規模で動作可能

WaitIO-Socketの設計方針

- Socketインタフェースの採用:
  - 異種プロセッサ、インターコネクト、MPI利用のプログラム間を結合
    - OdysseyのTofu-DインターコネクトもTCP/IP通信をサポート
      - 計算ノードとシステム外のシステムとの通信はGatewayノードを経由して実現
  - 直接Socketインタフェース上での通信ライブラリを実装することにより、MPI他のシステムソフトウェアとの共存が可能
- MPI仕様に準ずるAPIの提供:
  - 提供されるAPIや通信のセマンティックスは可能な限り通常利用の通信ライブラリと同じであることが望ましい。
- 最小限の計算機リソースで動作:
  - 全体同期の排除
  - 計算機リソース使用の抑制

実装概要

- 初期化：waitio_init()呼び出し時
  - すべてのPB上のプロセスのIPアドレスとPort番号のセットすべてをPB上のプロセス間で共有
- グループ作成： waitio_create_group() で指定された情報でグループ作成
  - すべてプロセスローカルで処理が完了、同期不要
- 通信プロトコル： 動的Socket通信確立
  - 128 byteまで：制御・データ一体型のEager、128byte超はRendezvousプロトコル
  - 64KB超の場合：64KB単位に分割送信
- 通信処理のプログレス処理：
  - 通信確立済みのEagerプロトコル以外、waitio_wait()関数呼び出し時実行
- 全体通信処理実装概要:
  - 受信処理： Linuxのepoll(event polling)関数群を用いて実装
  - 送信処理： 基本writeシステムコール呼び出し、
    write処理のエラー発生時にepoll処理に移行、write処理が完了後epoll処理を解除
- 通信の信頼性確保：制御パケットにマジック番号とシーケンス番号導入
- 利用ネットワークデバイス： システムごとに指定可能
- プロセッサEndian type： 現状Littleのみ実装。
- 実装コード：C でライブラリ5.7Kstep, テストプログラム1Kstep
  - スケジュール： 開発開始2021/05, First実装2021/06, 拡張・評価2021/07(予稿)

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
- アプリケーション開発者が必要に応じて作成することを想定
  - サポートDatatype, Operation他も必要に応じて追加
- 現状簡易実装: アルゴリズムはシーケンシャル実装とBinTree 実装（gatherv, scatterv除く）

WaitIO-MPI API (2021.08/07当時)	概要
waitio_mpi_isend	WaitIO実装版MPI_Isend
waitio_mpi_irecv	WaitIO実装版MPI_Irecv
waitio_mpi_reduce	WaitIO実装版MPI_Reduce
waitio_mpi_bcast	WaitIO実装版MPI_Bcast
waitio_mpi_allreduce	WaitIO実装版MPI_Allreduce
waitio_mpi_waitall	WaitIO実装版MPI_Waitall
waitio_create_universe	Waitio初期化サポート関数

WaitIO-MPI API (2021.9/10追加)	概要
waitio_mpi_gather	WaitIO実装版MPI_Gather
waitio_mpi_algather	WaitIO実装版MPI_Algather
waitio_mpi_scatter	WaitIO実装版MPI_Scatter
waitio_mpi_scatterv	WaitIO実装版MPI_Scatterv
waitio_mpi_gatherv	WaitIO実装版MPI_Gatherv
waitio_mpi_barrier	WaitIO実装版MPI_Barrier
waitio_mpi_type_size	WaitIO実装版MPI_Tyep_size

試作評価向けPingPong通信プログラム

- 評価目的：Wisteriaシステム間のデータ通信性能評価
- 評価プログラム:　複数ストリームでのPingPong通信
  - 全PB内プロセスで2プロセスペアを作りPingPong通信を実行
  - 全ペアのPingPong通信終了をMPI_Barrier()で同期
  - Intra-Nodeの場合：偶数rank+1のプロセスをペア
    - ノード内での複数プロセスコピー影響を評価
  - Inter-Nodeの場合：PB間 or rank+1/2*(PBサイズ)のプロセスをペア
    - 複数システム間のデータ転送性能評価
- 実行評価プログラムの実行環境
  - ベンダ製コンパイラ＋ベンダ製MPIで評価
    - Xeonプロセッサ搭載システムではOpen MPI+GCCでも動作確認済み

PingPong バンド幅評価

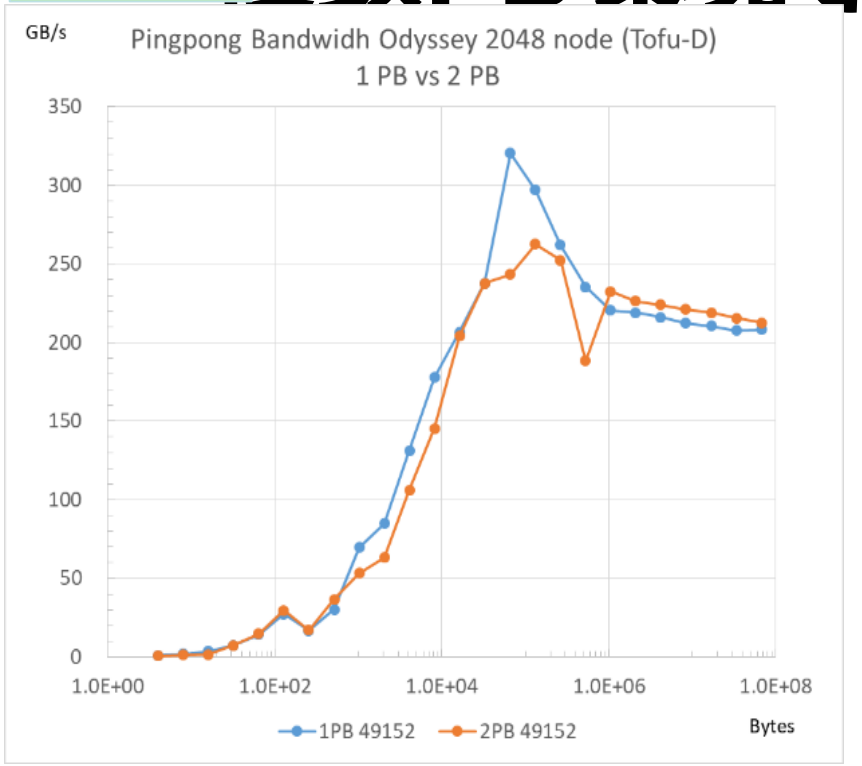
可変 proc- 16 Node固定

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)

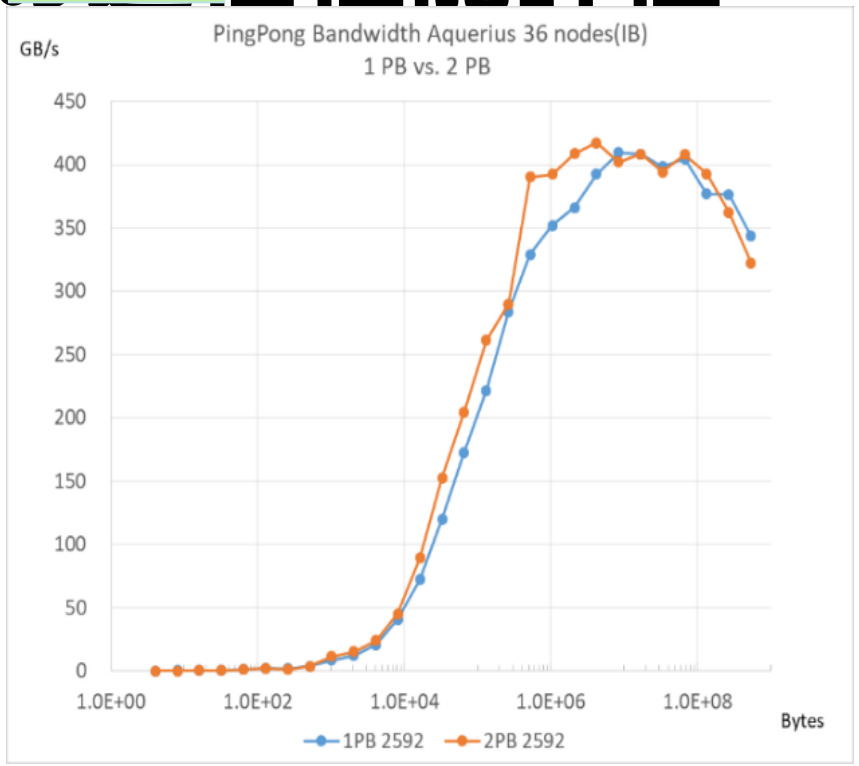
- CPU性能とインタコネクト性能のバランス
  - Odyssey: ノード内に比べTofu-D性能が大幅に劣化
  - Aquarius: ノード内、InfiniBand性能ともに他システムに比べ高速
  - OFP: CPU性能が通信性能律速の主因

複数PB環境での通信性能評価

Odyssey



Aquarius



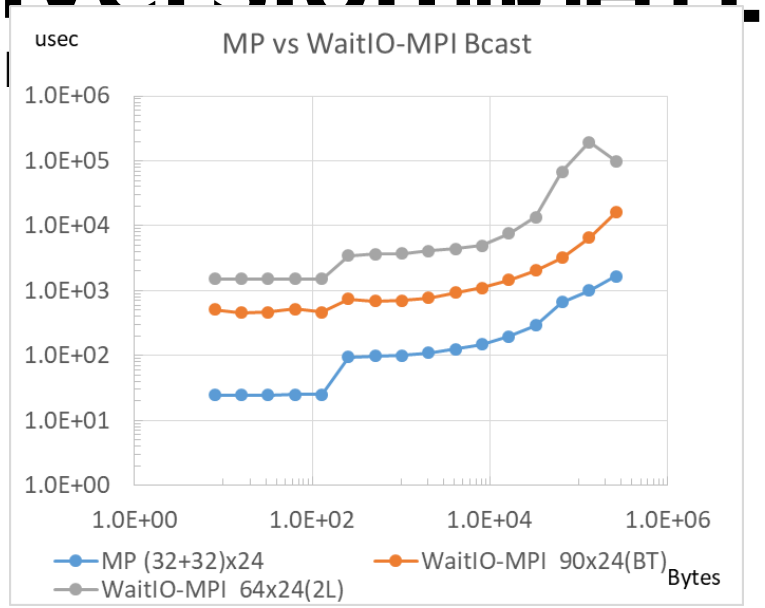
- Odyssey, Aquarius内での2PB間での通信性能比較
  - 実装上は通信性能差なし
  - 一部の性能差はアプリケーションのノードマッピングの影響と想定

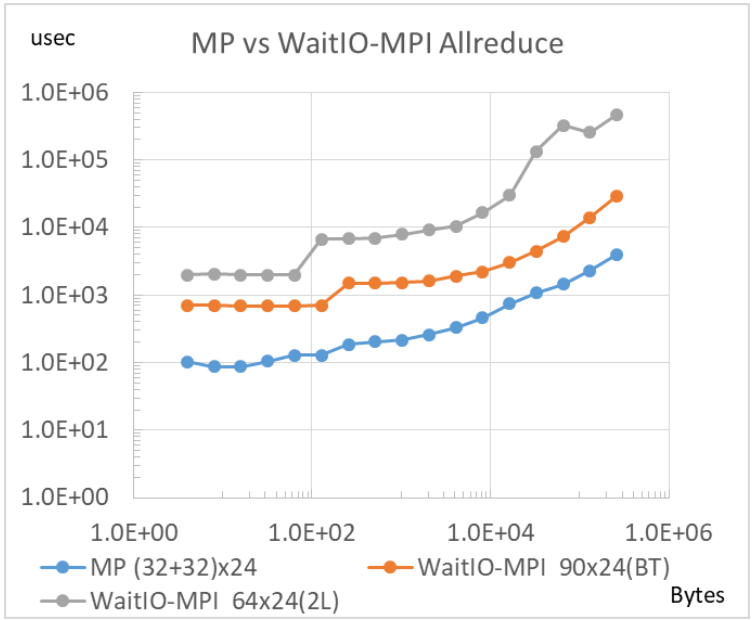
MP vs WaitIO-MPI Conversion通信性能

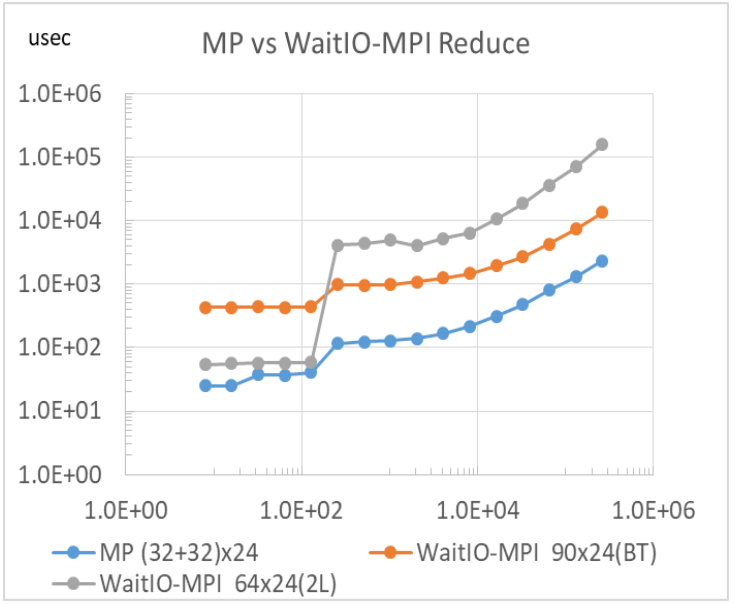
- Odyssey上で評価P(h3-Open-UTL/I
  - ハイブリッド実装に性能優位性あり

24proc/Node

	MP usec	WaitIO-MPI Lib(BT) usec	WaitIO-MPI Lib(BT)/iMPI
Bcast 8 bytes	24.7	509.3	20.6
Bcast 256 bytes	94.4	744.2	7.9
Reduce 8 bytes	24.9	433.6	17.4
Reduce 256 bytes	116.2	985.0	8.5
Allreduce 8 bytes	102.0	698.5	6.8
Allreduce 256 bytes	186.7	1500.0	8.0







pHEAT-3Dアプリケーションでの性能評価

- 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
  - W

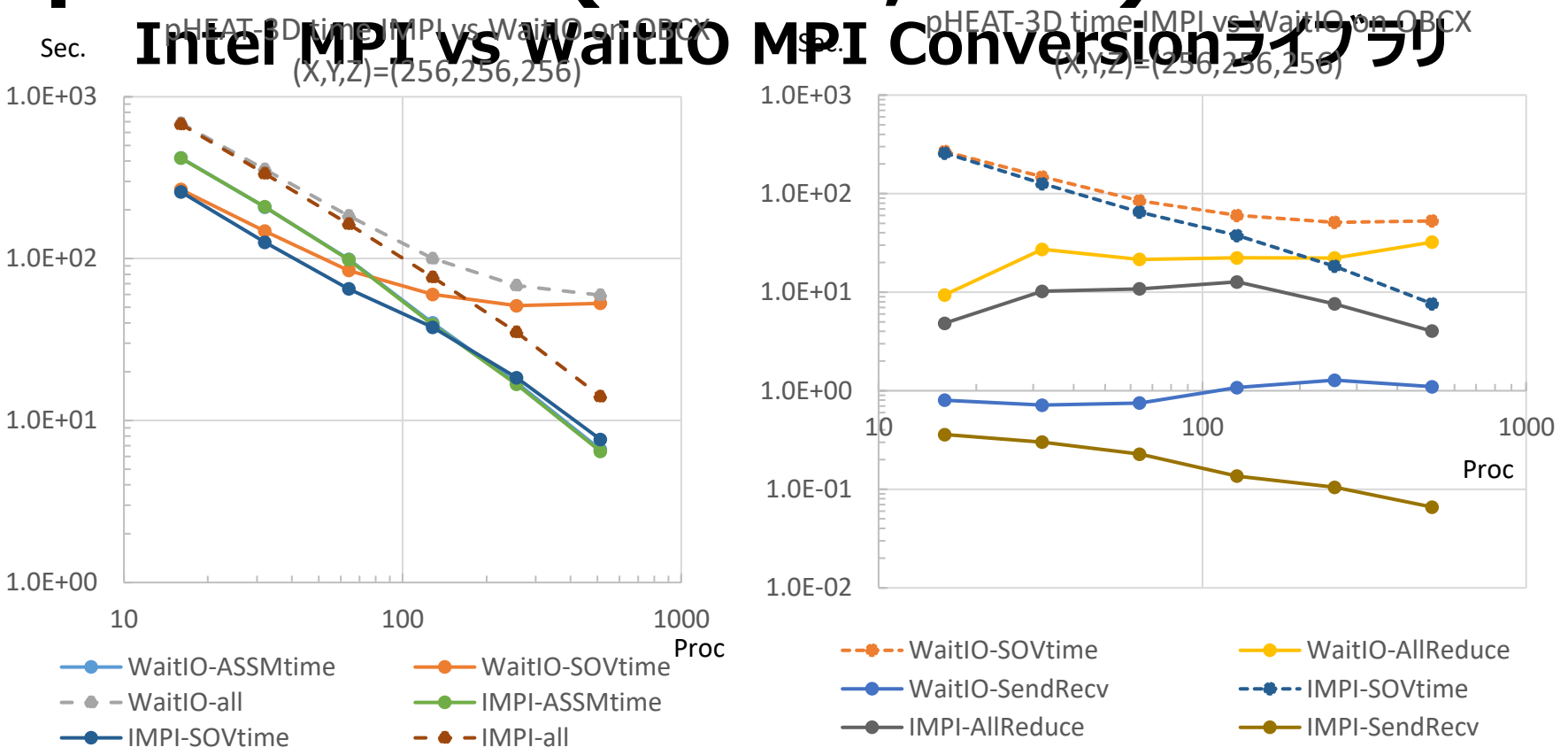
```
      include  'mpif.h'
      include 'waitio_mpif.h'
      integer(kind=kint ), dimension(:,:), save,allocatable :: req1
      integer, save :: NFLAG
      data NFLAG/0/

!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)
```

pHEAT-3D(OBCX,1PB)実行時間

Intel MPI vs WaitIO MPI Conversionライブラリ

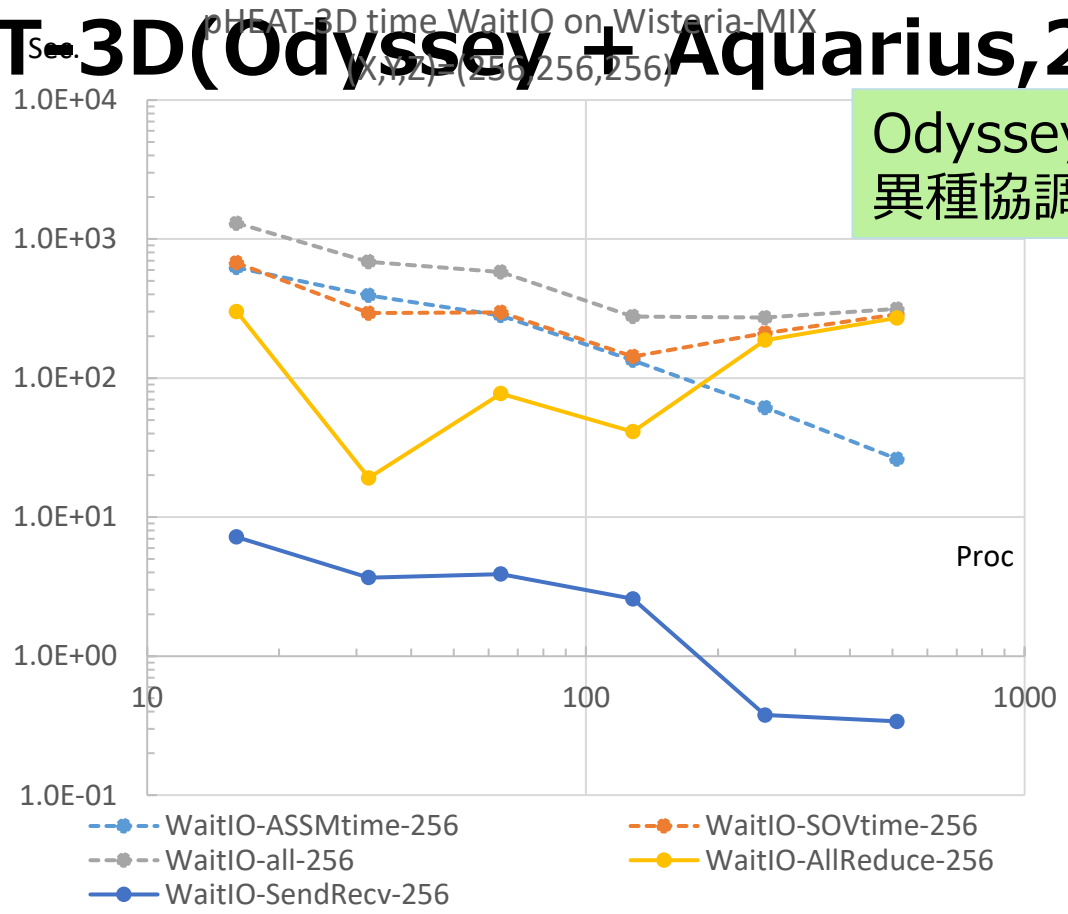


- Intel MPI: スケーラブル
- WaitIO: 256プロセス以上頭打ち、
  - 更なる高速化にはハイブリッド実装が必要
  - 通信ではAllreduceのコストが支配的

8 proc/Node

予稿提出後の進捗:

pHEAT-3D(Odyssey + Aquarius,2PB)測定結果



Odyssey + Aquarius で
異種協調計算を実行確認

- Aquarius(2node,2T/Proc)+Odyssey(4-112node,12T/Proc)で協調動作
  - 全処理WAITIO-MPI Conversionで変換, metis処理はAquariusで処理
- 128プロセス以上頭打ち
  - Allreduce時間が悪化：Odysseyの通信遅延の影響

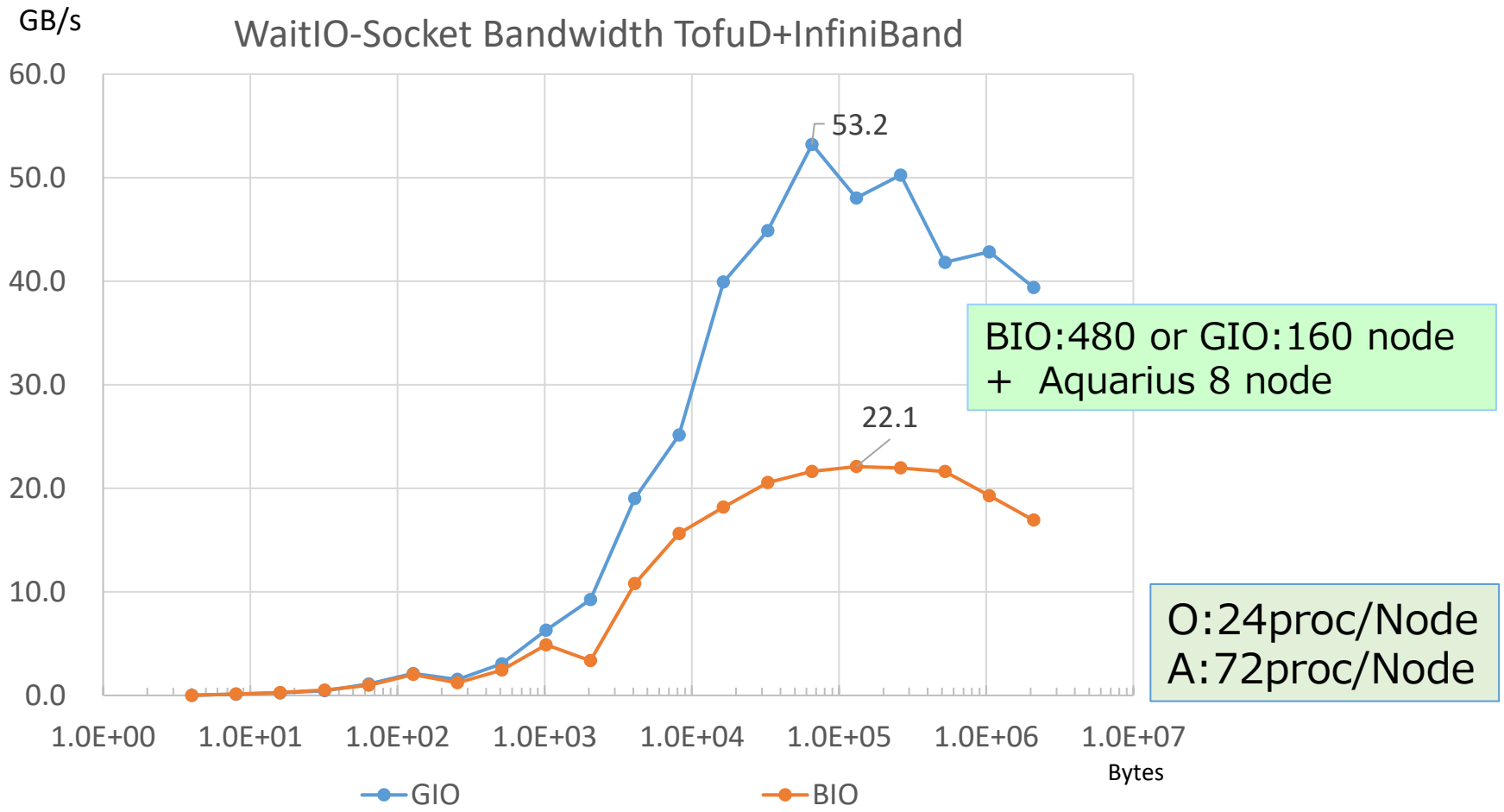
関連研究

- MPI規格を用いる手法：Open MPI, MPICH
  - MPI_Comm_spawn()とMPI_Comm_connect() / MPI_Comm_accept()
    - Open MPI: 異種ネットワーク、異種プロセッサ上でアプリケーション実行可能
    - MPI_Comm_spawn()： ジョブスケジューラとMPIライブラリの対応が必要
      - MPIライブラリの統一化: 異種システム環境は容易でない：富士通MPI, Intel MPI
      - ジョブスケジューラ統合： 様々なシステム環境での実行を考慮すると非現実的
    - MPI_Comm_connect()/MPI_Comm_accept()利用: Socket生成MPI通信で利用
      - 期待通りに動作しない場合あり、MPI version他
      - Socket通信のシステム内管理は別途必要
  - MPI Session(提案中): 規格のため実装依存、異種環境実行は保証せず
- 複数プラットフォーム対応の低レベル通信を用いる手法: UCX, OFED
  - UCX, OFED：TCP/IP, InfiniBandなど異種ネットワーク、異種プロセッサ対応
  - すべての環境で利用可能であるかが不明
    - Odyssey： UCX, OFEDをユーザアプリケーションでサポートなし
- WaitIO-Socket: Socket通信さえできれば実行可能
  - Odyssey(Arm+SVE, TofuD, 富士通MPI) と Aquarius(Intel Xeon, InfiniBand, Intel MPI)上で相互通信できることを確認

Odyssey + Aquarius 全系評価概要

- 評価パターン
  - Odyssey 計算ノード- Aquarius計算ノード間(GIOルーティング有)
  - Odyssey GIOノード- Aquarius計算ノード間(GIOルーティングなし)
- 全系システム間性能 (推定):
  - Odyssey 計算ノード- Aquarius計算ノード間
    - 1ノードあたり転送性能: 350MB/s
    - GIO Routing性能1/2: 175MB/s
    - 1ラックあたり 175 x 8: 1.4GB/s
    - 全系通信性能20ラック : 28 GB/s
  - Odyssey GIOノード- Aquarius計算ノード間
    - GIO vs Aquarius計算ノード間通信性能: 370MB/s (iperf3)
    - 1ラックあたり 370 x 8: 2.96GB/s
    - 全系通信性能20ラック : 59.2 GB/s

Odyssey - Aquarius Communication



- 通信性能：Odyssey全系＋Aquarius 8 nodeで評価
  - BIO通信(w/ routing)性能は 22.1 GB/s （推定: 28 GB/s）
  - GIO通信(w/o routing)性能は 53.2 GB/s （推定: 59.2 GB/s）
- 実行プロセス数：全系(Max 187,200 procs)まで動作確認