MPICH-Tofuの概要

国立情報学研究所
石川　裕

MPICH-Tofu

- なぜ、MPICH-Tofuを開発したか？
  - オープンソースな通信ライブラリを提供したかった
    - 研究者が自由に改変して研究できる研究ツールを提供したかった
- ソースリポジトリ
  - https://github.com/yutaka-ishikawa/mpich-tofu
- インストールの仕方
  - git clone https://github.com/yutaka-ishikawa/mpich-tofu.git
  - 00README.txtを読んでください

Wisteriaでの使い方

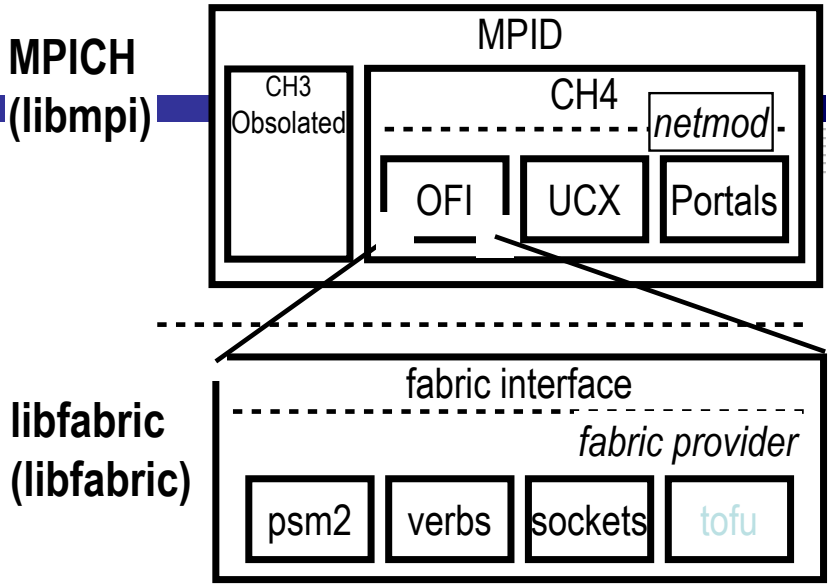
```
#!/bin/bash
#------ pjsub option --------#
#PJM -N “MPICH256"
#PJM -S
#PJM --spath "results/%n.%j.stat"
#PJM -o "results/%n.%j.out"
#PJM -e "results/%n.%j.err"
#PJM -L "node=256"
#PJM -g “XXXX"
#PJM -L "rscgrp=regular-o"
#PJM --mpi "max-proc-per-node=4"
#PJM -L "elapse=00:50:00"
#PJM -L proc-core=unlimited
#------- Program execution -------#
```

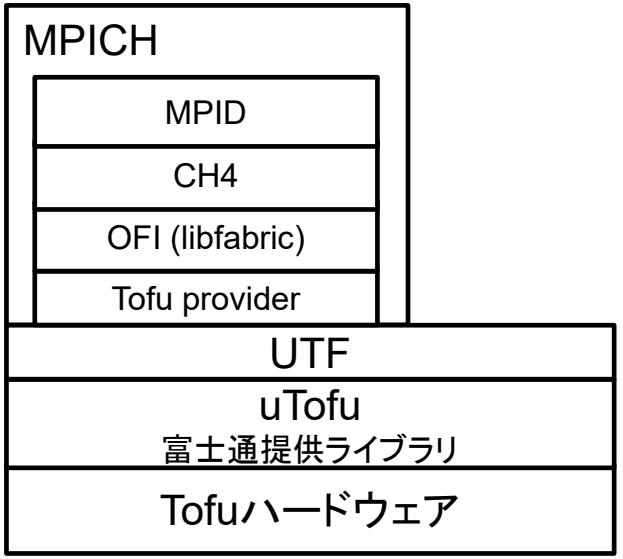
プロセス数が127以下であれば、
UTF_TRANSMODE=1
とした方が通信性能は良い

```
MPIOPT="-of results/%n.%j.out -oferr results/%n.%j.err"
export MPICH_HOME=/data/XXXX/mpich-tofu/
export PATH=$PATH:$MPICH_HOME/bin:$PATH
which mpich_exec
export MPIR_CVAR_CH4_OFI_CAPABILITY_SETS_DEBUG=1
#
# UTF_MSGMODE (0: Eager, 1: Rendezous)
#               Eager in default
# UTF_TRANSMODE (0: Chained, 1: Aggressive)
#               Aggressive in default
#
export MPICH_TOFU_SHOW_PARAMS=1
export UTF_MSGMODE=1
export UTF_TRANSMODE=0
export UTF_INFO=1
#export UTF_TOFU_SHOW_RCOUNT=1
export UTOFU_SWAP_PROTECT=1

mpich_exec -n 256 $MPIOPT $CMD -npmin $NPMIN $BENCH

exit
```

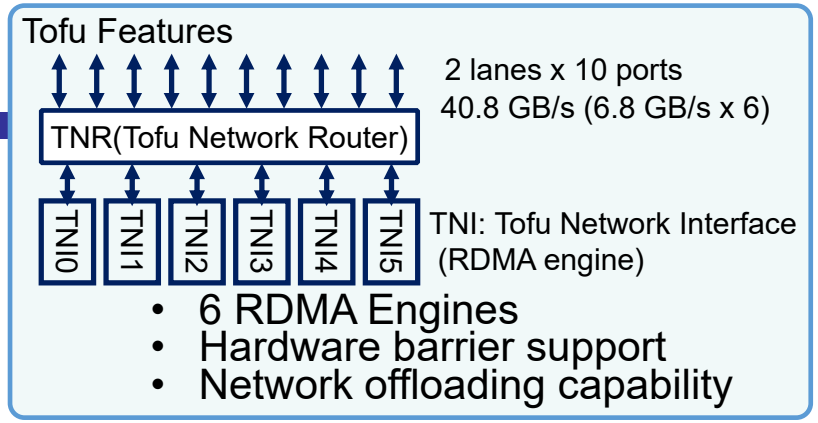
MPICH-Tofu

- 米国アルゴンヌ国立研究所が中心になって開発しているMPI通信ライブラリ実装であるMPICHを「富岳」および富士通FX-1000のTofu通信向けに移植
- MPICHはネットワークデバイス層としてOFI, UCX, Portalsを想定
- OFI (Open Fabric Interface)をTofu通信向けに実装
  - Fabric providerとしてtofuを実装





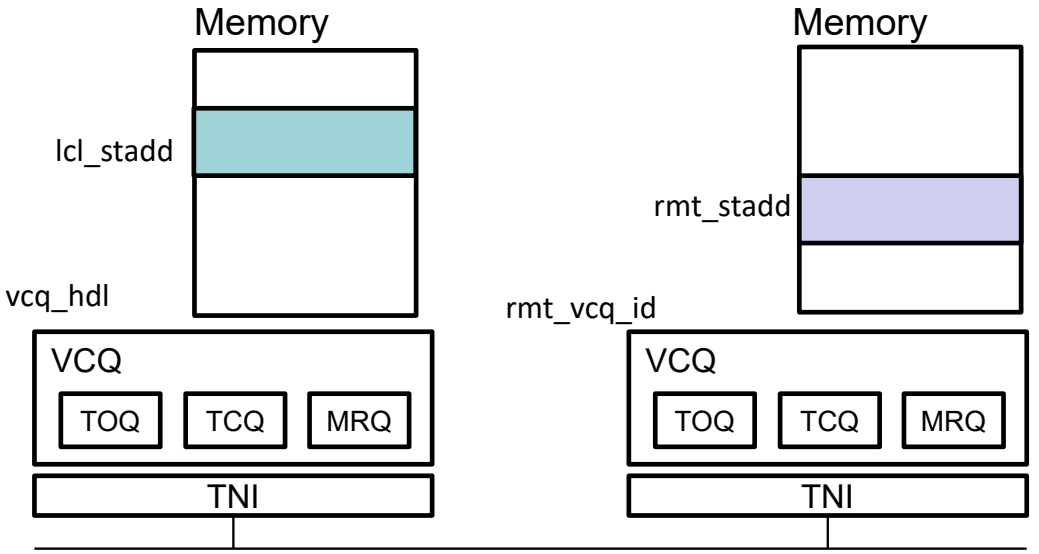
uTofuインターフェイス



- RDMAベースの通信
  - put/get
  - atomic_read_write
    - swap, add, xor, and, or
- 通信領域は予め登録しないといけない
  - Steering Address
- VCQを使って通信
- バリア通信
  - リダクション演算
    - BAND, BOR, BXOR, MAX, MAXLOC, SUM, BFPSUM

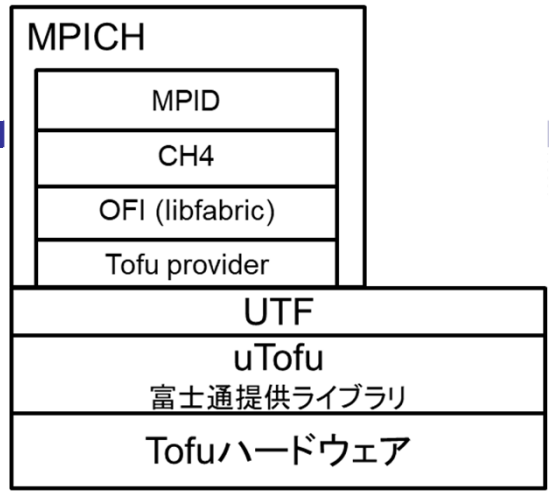
    - VCQ: Virtual Control Queue
    - TOQ: Transmit Order Queue
      - 送信指示キュー
    - TCQ: Transmit Complete Queue
      - 送信完了キュー
    - MRQ: Message Receive Queue
      - 受信キュー

tofu_put(vcq_hdl, rmt_vcq_id, lcl_stadd, rmt_stadd, len, …)



詳しくは以下手引書参照
FUJITSU Software Technical Computing Suite
Development Studio uTofu使用手引書

UTF

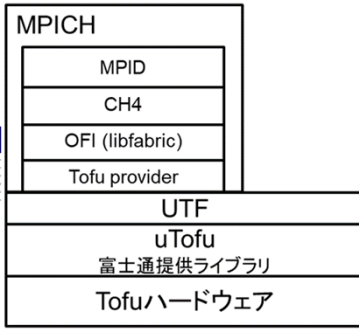


- Point-to-pointメッセージ通信を実現
  - uTofuはRemote Memory Access機能のみを提供
- 2タイプの通信プロトコル
  - UTF_MSGMODE
    - 0: Eger, 1: Rendezvous
    - Eager in default
- 2タイプの通信バッファ管理
  - UTF_TRANSMODE
    - 0: Chained, 1: Aggressive
    - Aggressive in default

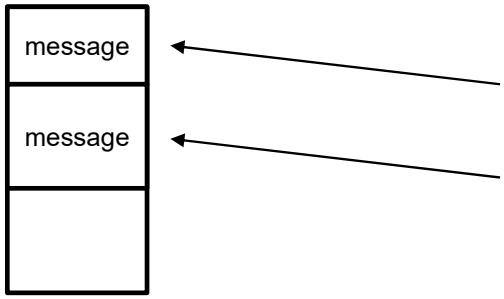
プロセス数が127以下であれば、
UTF_TRANSMOD=1
とした方が通信性能は良い

libfabric (or OFI)の通信

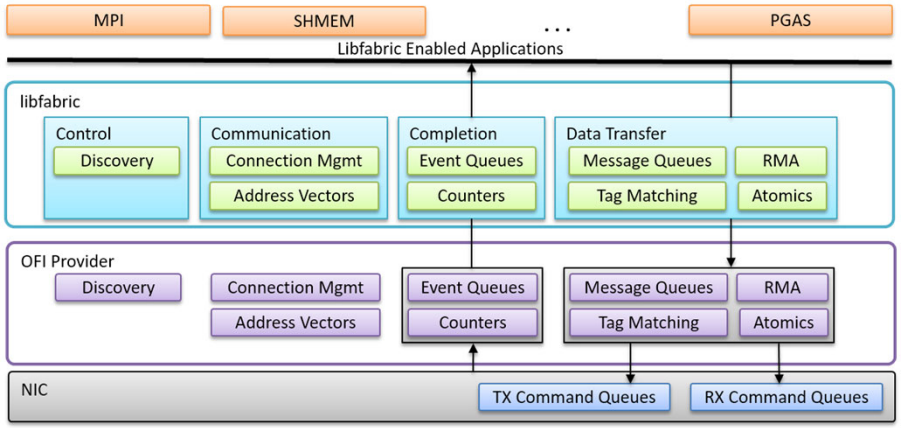
- fi_msg (Message Queue)
  - Point-to-point通信機能
  - FIFOオーダーが保証されている
  - Multi-Receive機能が特徴的
- fi_tagged (Tagged Message)
  - MPI通信ライブラリにおけるタグと同じコンセプト
  - 送信時メッセージにタグを付与することができる
  - 受信時にタグを指定することにより選択的にメッセージを受信できる
- fi_rma (Remote Memory Access)
  - Remote memory get, put
- Fi_atomic (Atomic)
  - atomic-add, atomic-compareなど
- CQ (Completion Queue)
  - 上記操作の完了を受け取る機能



fi_recvmsg(ep, buf, FI_MULTI_RECV);



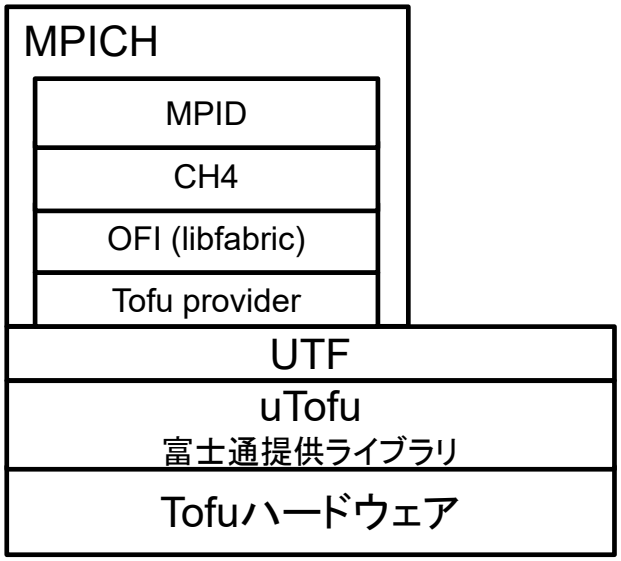
- FI_MULT_RECVフラグをONにすると、receive操作を発行しなくても複数のメッセージを指定したバッファ領域に受信
- CQ経由で受信したメッセージ情報を受け取ることができる



https://ofiwg.github.io/libfabric/

MPICH

- OFIを使った実装では2つの通信方法がある
  - OFIのタグ通信を使った実装
    - MPIR_CVAR_CH4_OFI_ENABLE_TAGGED環境変数で制御
      - defaultは-1、enabled
  - OFIのMulti-receive機能を使ったActive Message実装
    - MPIR_CVAR_CH4_OFI_ENABLE_AM環境変数で制御
      - Defaultは-1、enabled



ハードウェアバリア機能を使った高速化

- 以下のCollectiveはTofuのハードウェアバリア機能を使った実装も提供
  - MPI_Barrier： 4プロセス以上の時有効
  - MPI_Bcast： 128バイト以下のデータサイズの時有効
  - MPI_Reduce：128バイト以下のデータサイズの時有効
  - MPI_Allreduce：128バイト以下のデータサイズの時有効
- 以下の環境変数を設定すると有効になる
  - export UTF_BG_LOAD=1
- 以下の環境変数はバリア機能が正常に初期化されるか確認する目的で提供
  - export UTF_BG_CONFIRM=2

おわりに

- なぜ、MPICH-Tofuを開発したか？
  - オープンソースな通信ライブラリを提供したかった
    - 研究者が自由に改変して研究できる研究ツールを提供したかった
- ソースリポジトリ
  - https://githu
- インストール
  - git clone https://github.com/yutaka-ishikawa/mpich-tofu.git
  - 00README.txtを読んでください

一緒にMPICH-tofu使って研究開発したい人募集してます
yutaka_ishikawa@nii.a.jp まで