（並列プログラミング言語XcalableMP規格部会）
「XMP規格部会の活動報告」

PGASプログラミング言語（XcalableMP）の課題

XcalableMP規格部会
佐藤 三久
理研 AICS & 筑波大学 計算科学研究センター

並列プログラミング言語XcalableMP規格部会　2017年度中間報告

- 今年度は1回の部会を開催 (8/1@福岡)
  - 新仕様V2.0に向けた新しい議論
    - メニーコア対応のため、動的なタスクモデルの実現検討、プロトタイプ実装の状況
  - 現仕様V1系の詰めのための議論
    - ループ並列化の強化 (expand, margin, peel_and_wait), シャドウのリダクション (reduce_shadow), reflect の orthogonal 節
    - [C] 多次元分散配列の動的割付け, ノード配列やテンプレートにおける [ ] 記法
    - 組込み関数の追加 (xmp_exit, xmpc_node_num など), その他 errata
- 第5回XMPワークショップ開催（10/31@秋葉原）
  - 共催：ポストペタCREST　筑波大　朴チーム、理化学研究所
  - Invited Talk ２件　(AICSサポート)
    - Valentin Clément (C2SM, ETH Zurich / MeteoSwiss)
    - Joseph Schuchart (High Performance Computing Center Stuttgart, HRLS)
- XMP講習会を開催
  - 9月：CEA@フランス、東大 for OFP、福岡@PCC WS（予定）
- McKernelと合同で5分程度のビデオを作製
- リファレンス実装（Omni compiler）
  - V1.2.2 リリース（12/4)
    - Add Python module., Support KNL : "./configure --target=KNL-linux-gnu"., Improve performance of XMP/C on for-loop statement, *Fixed lots of bugs.
  - Fortran2008向け対応を実装中、C++向け対応を検討中 (LLVM clangベースフロントエンド)
  - Mixed Language feature (with MP and python)
  - XMP 2.0のプロトタイプ実装・評価中

XcalableMP

- 分散メモリ環境を対象とした指示文ベースの並列言語
- 次世代並列プログラミング言語検討委員会 → 当部会において仕様を検討、提案。
- 2つの並列プログラミングモデルをサポート
  - グローバルビューモデルによる定型的な並列化
  - ローカルビューモデルによる自由度の高い並列化

XcalableMPの現況

- PCクラスタコンソーシアムで規格を議論
  - 2017/April Version 1.3仕様を公開。
    - Version 1の系統はほぼ「収束」
    - ループ並列化の強化 (expand, margin, peel_and_wait), シャドウのリダクション (reduce_shadow), reflect の orthogonal 節
    - [C] 多次元分散配列の動的割付け, ノード配列やテンプレートにおける [ ] 記法
    - 組込み関数の追加 (xmp_exit, xmpc_node_num など), その他 errata
  - 2015年より、次期仕様「XMP2.0」の検討を開始。
    - PGAS + Multitasking for Multicore
    - Code transformation for Optimization
    - (Accelerator)

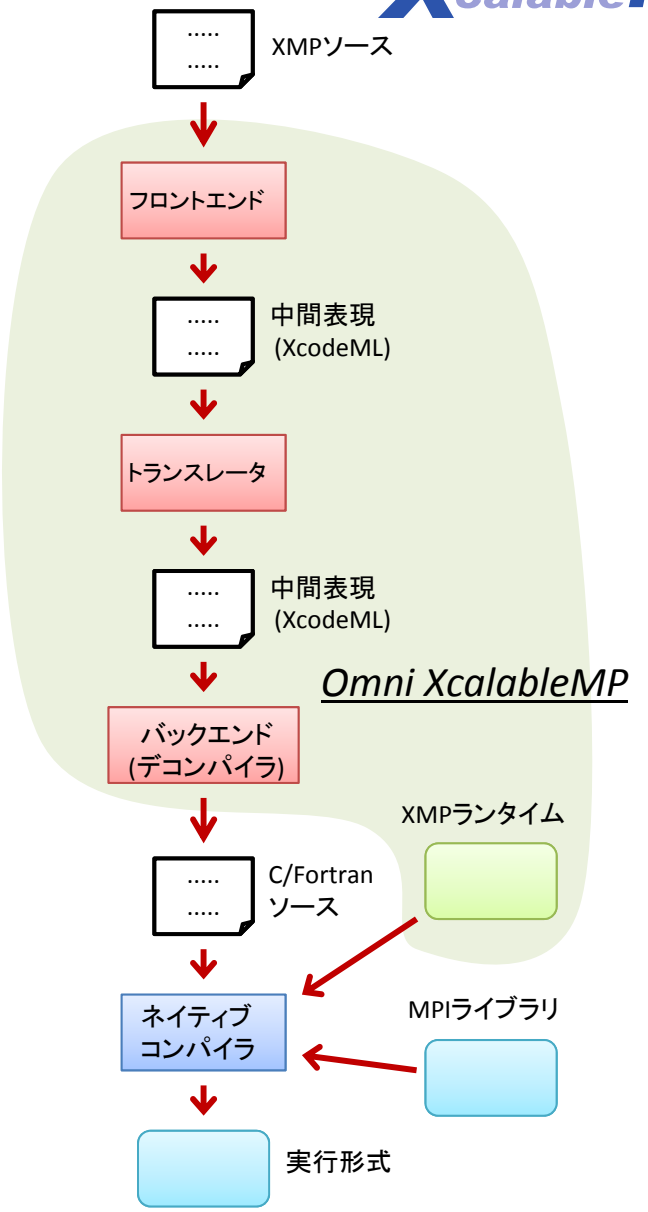
www.xcalablemp.org

- 理研・筑波大で、レファレンス実装
  - Omni XMP コンパイラ

omni-compiler.org

Omni XcalableMP

- 理研AICSと筑波大で開発中のXMP処理系
  - XMP/C
  - XMP/Fortran
- オープンソース
- トランスレータ + ランタイム(MPIベース)
- OpenACC、XcalableACC対応



XcalableMPプロジェクト(＆規格部会)の これからの方向性

- 言語、コンパイラ研究ともに、使ってもらうことが大切
  - Global viewについては継続して、良さをアピール！
  - 多様な使い方を提案する必要がある。
    - 多言語(e.g. python) との組み合わせ。
- プログラムの最適化・高速化に貢献できるモデル・仕組みを提案
  - MPIよりも速いプログラムを！　PGASの使い方の追求。
  - PGASとマルチタスクの統合のモデル
- 「規格化」することは重要だが、XMP 2.0ではもう少し、研究指向もありか!?
  - GPU/FPGAも！

XcalableMP

How does PGAS “collaborate” with MPI+X?

Contributors: members of XMP-WG
Hitoshi Murai (RIKEN), Masahiro Nakao (RIKEN), Jinpil Lee (RIKEN), Tesuya Odajima (RIKEN), Hidetoshi Iwashita(Fujitsu), Hitoshi Sakagami (NIFS), Takeshi Nanri (Kyushu U), Atsushi Hori (RIKEN), Taisuke Boku (U. Tsukuba), Akihiro Tabuchi (U. Tsukuba), Keisuke Tsugane (U. Tsukuba)

Mitsuhisa Sato

Deputy Project Leader and Team Leader, Architecture Development Team, Flagship 2020 Project

Team Leader, Programming Environment Research Team

Advanced Institute for Computational Science, RIKEN

Second Annual PGAS Applications Workshop @ SC17
Keynote talk by M. Sato

第5回 XcalableMPワークショップ
岩下英俊（富士通）

FUJITSU
shaping tomorrow with you

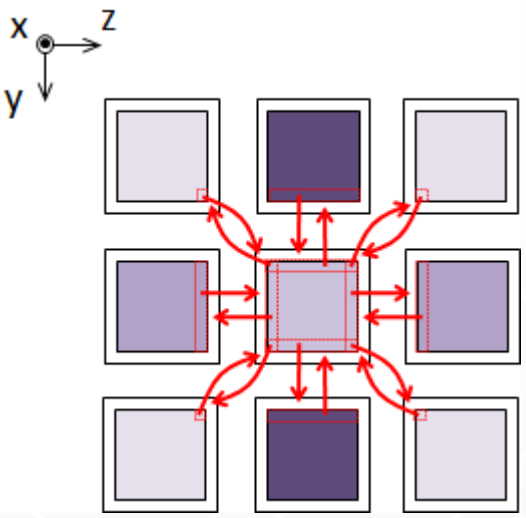
Coarrayは性能でMPIを超えられるのか？
～Omni XcalableMPの開発経験から～

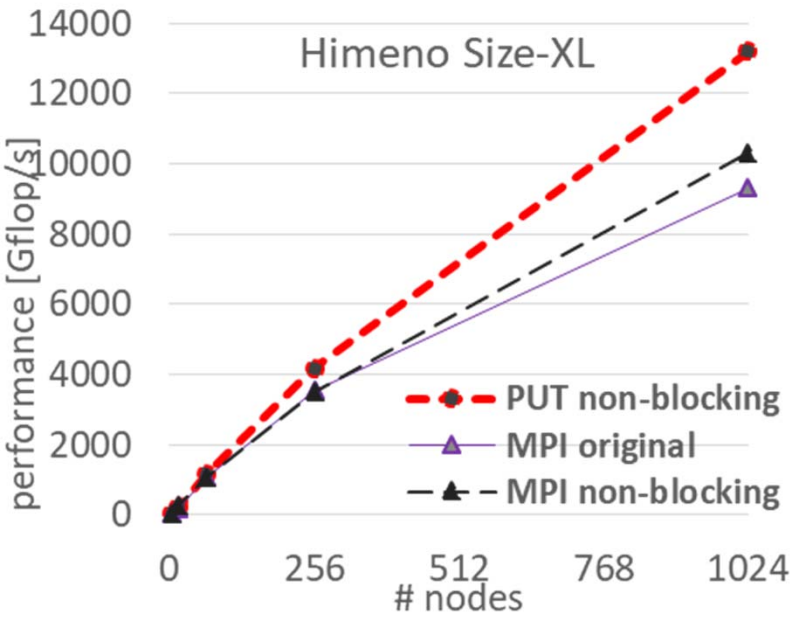
2017/10/31
富士通（株）次世代TC開発本部
岩下 英俊

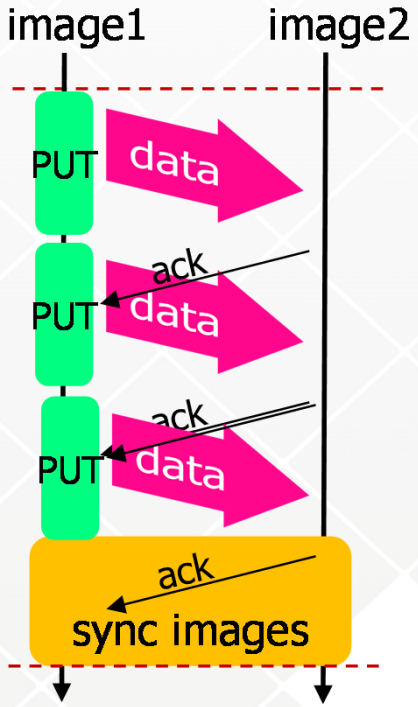
※この内容は、理化学研究所計算科学研究機構（AICS）
アーキテクチャ開発チームで開発研究員として行った活動の成果です.

Case study: stencil communication

- Typical communication pattern in domain-decomposition.
- Advantage of PGAS: Multiple data transfers with a single synchronization operation at end
- PUT non-blocking outperforms MPI in Himeno Benchmark!
  - Don’t wait ack before sending the next data (by FJ-RDMA)







NOTE: The detail of this results is to be presented in HPCAisa 2018: Hidetoshi Iwashita, Masahiro Nakao, Hitoshi Murai, Mitsuhisa Sato, “A Source-to-Source Translation of Coarray Fortran with MPI for High Performance”

Support communication patterns in PGAS

- When the communication pattern is expressed in some defined forms, PGAS comm. can be optimized.
  - Coarray assignment by array section syntax. Runtime select better stride comm, pack/unack or direct RDMA.

```
A(1, 1:10)[1] = B(1:10)    // put from image2 to image1
```

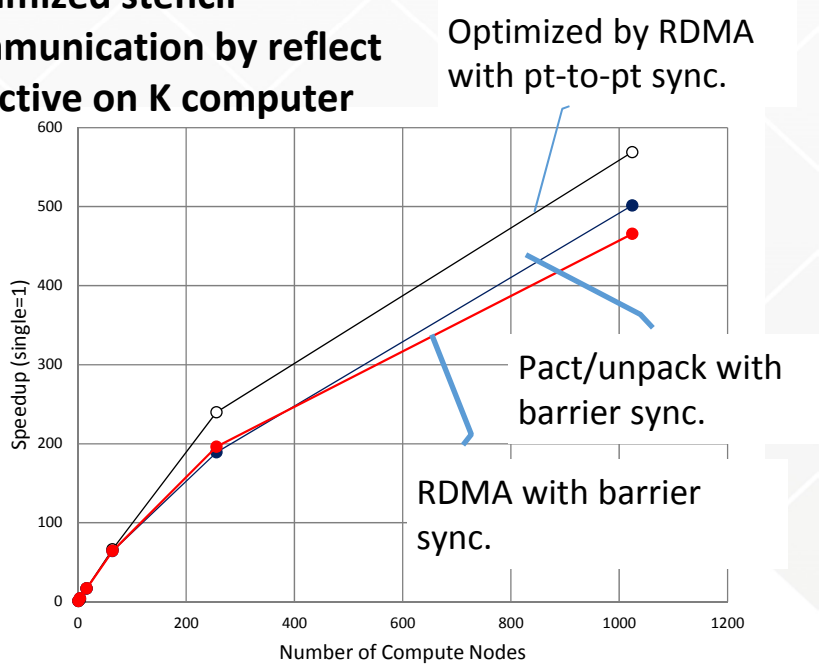
  - Reflect operation in XMP global view model (stencil neighbor comm)

- Irregular Communication Patterns:
  - If communication *pattern* is not known *a priori*, but the data locations are known, send-recv program needs an extra step to make pairs of send-recv. BUT, RMA can handle it easily because only the source or destination process needs to perform the put or get call

```
!$xmp shadow a(2:2, 1:1)
!$xmp reflect (a) width (/periodic/1:1, 0:0) async (0)
!$xmp wait_async (0)
```

- **shadow: declares width of shadow**
- **reflect: executes stencil comm.**
  - width: specifies shadow width to be updated. /periodic/: updates shadow periodically. async: asynchronous comm.
- **wait_async: completes async. reflects**

Optimized stencil communication by reflect directive on K computer

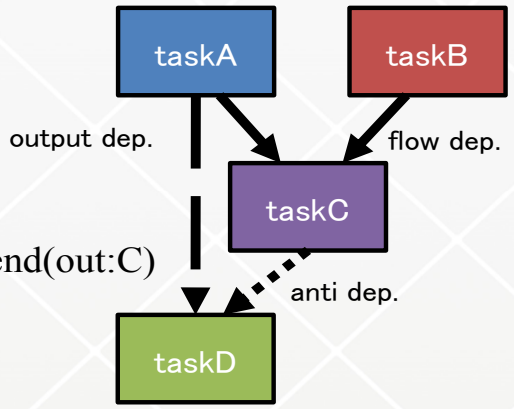


Target: a prototype dynamical core of a climate model SCALE-LES

Task in OpenMP and extension to between nodes

- Task directive in OpenMP4.0 creates a task with dependency specified “depend” clause
- The task dependency depends on the order of reading and writing to data based on the sequential execution.
  - OpenMP multi-tasking model cannot be applied to tasks running in different nodes since threads of each nodes are running in parallel.
  - In OmpSs, interactions between nodes are described through the MPI task that is executing MPI communications (MPI task).
  - Otherwise, run the same task program in all nodes to resolve dep. (StarPU)

```
#pragma omp parallel
#pragma omp single
{
   int A, B, C;
#pragma omp task depend(out:A)
   A = 1;          /* taskA */
#pragma omp task depend(out:B)
   B = 2;          /* taskB */
#pragma omp task depend(in:A, B) depend(out:C)
   C = A + B;  /* taskC */
#pragma omp task depend(out:A)
   A = 3;          /* taskD */
}
```



- *Flow dependency: The flow dependency occurs between dependence-type out and in with same variables, similar to read after write (RAW) consistency. It is shown in between taskA and taskC with variable A, or taskB and taskC with variable B.*
- *Anti dependency: The anti dependency occurs between dependence-type in and out with same variables, similar to write after read (WAR) consistency. It is shown in between taskC and taskD with variable A.*
- *Output dependency: The output dependency occurs between dependence-type out and out with same variables, similar to write after write (WAW) consistency. It is shown in between taskA and taskC with variable A.*

Multitasking in XMP: our proposal

- **For Intra-node**
  - Tasklet directive creates a task with dependency specified “in/out/inout” clause in sequential order
  - Same as in OpenMP4.0 and OMPss
- **For Inter-node**
  - Describe communications by PGAS operations (Coarray and gmove)
  - Annotated by get/put and get_ready/put_ready clauses to specify dependency.

`#pragma xmp tasklet` *tasklet-clause[, tasklet-clause[, ...]]* on {*node-ref* | *template-ref*}
(structured-block)

where *tasklet-clause* is :
{`in` | `out` | `inout`} (*variable[, variable, ...]*)
or
{`put` | `get`} (*tag*)
or
{`put_ready` | `get_ready`} (*variable*, {*node-ref* | *template-ref*}, *tag*)

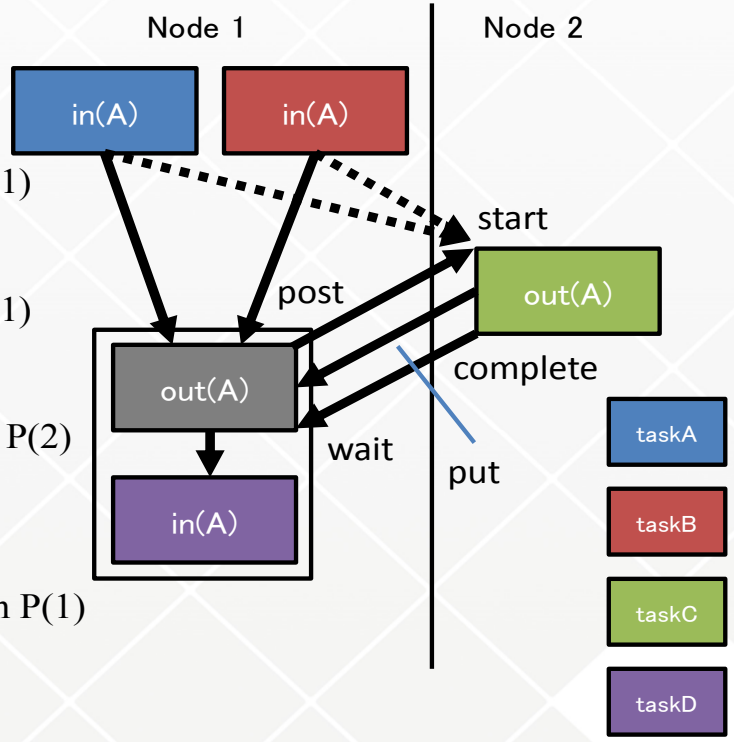
`#pragma xmp taskletwait` *[on* {*node-ref* | *template-ref*}*]*

Put operation in tasklet

- put_ready clause: indicates that the specified data may be written by the associated PUT operation
  - This clause has the dependence-type out for the specified data on a node since its values are overwritten by the remote node.
- put clause: indicates that the PUT operation may be performed in the associated structured block.
  - At the beginning of the block, the task waits to receive the post notification with the tag by the put_ready clause to indicates that the data is exposed in the target node for the PUT operations.

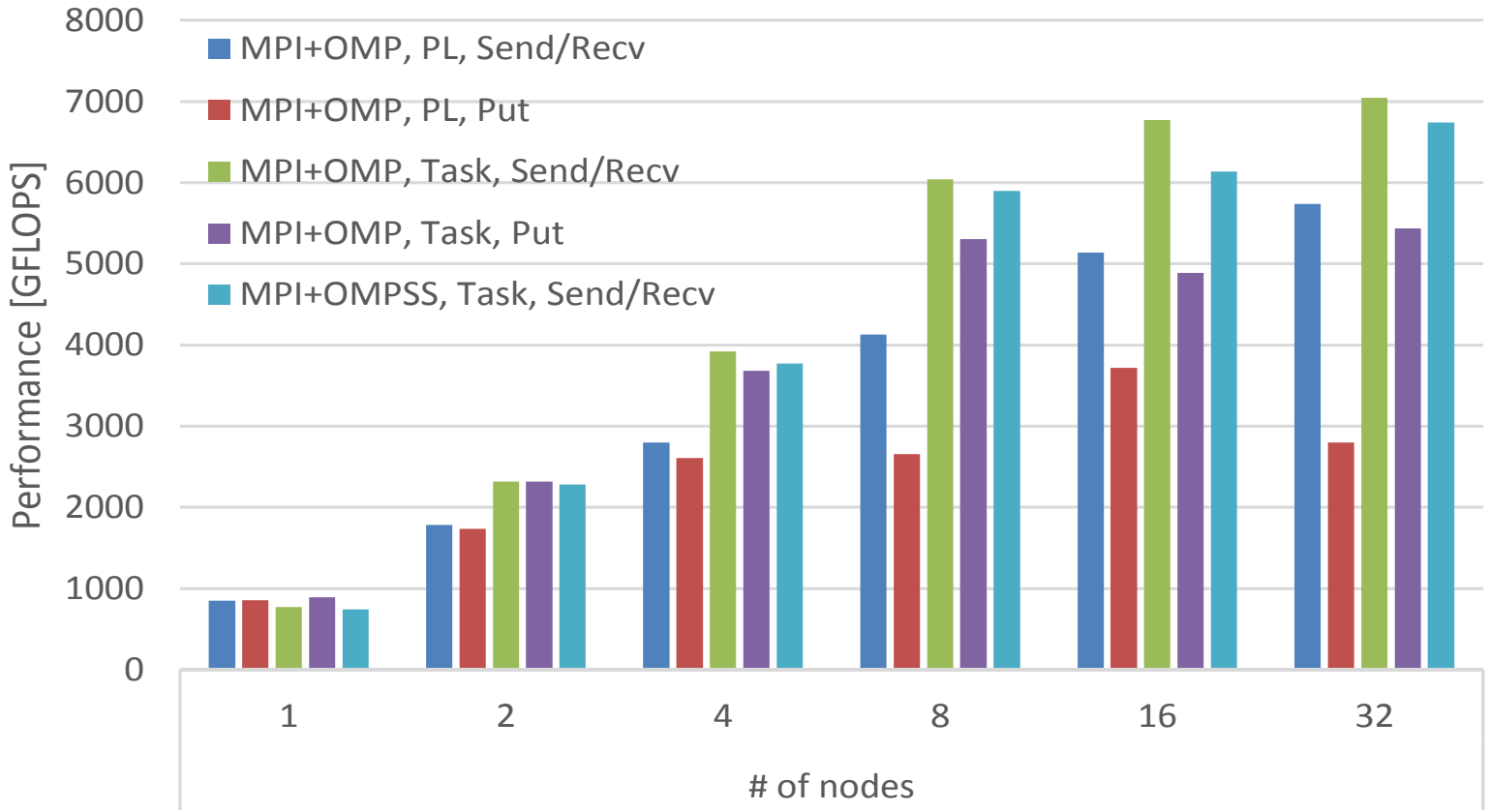
- When output dependencies for the data are satisfied before executing the block, the clause exposes the data for the PUT operation from the specified set of nodes by sending the post notifications to these nodes, starting the PUT operations eventually in remote nodes. Then, it waits until remote operations are done. When the task receives the completion notification of the PUT operation, the block is immediately scheduled.
- When the post notification is received, the task is scheduled to execute the calculation and PUT operation in the block. When the execution of the block is finished, the data written by the PUT operation is flushed and the completion notification is sent to the node matched by the tag.

```
#pragma xmp nodes P(2)
   int A:[*], B, C, D, tag;

#pragma xmp tasklet in(A) out(B) on P(1)
   B = A;        /* taskA */

#pragma xmp tasklet in(A) out(C) on P(1)
   C = A;        /* taskB */

#pragma xmp tasklet out(A) put(tag) on P(2)
   A:[1] = 1;    /* taskC */

#pragma xmp tasklet in(A) out(D) ¥
                  put_ready(A, P(1), tag) on P(1)
   D = A;        /* taskD */
```



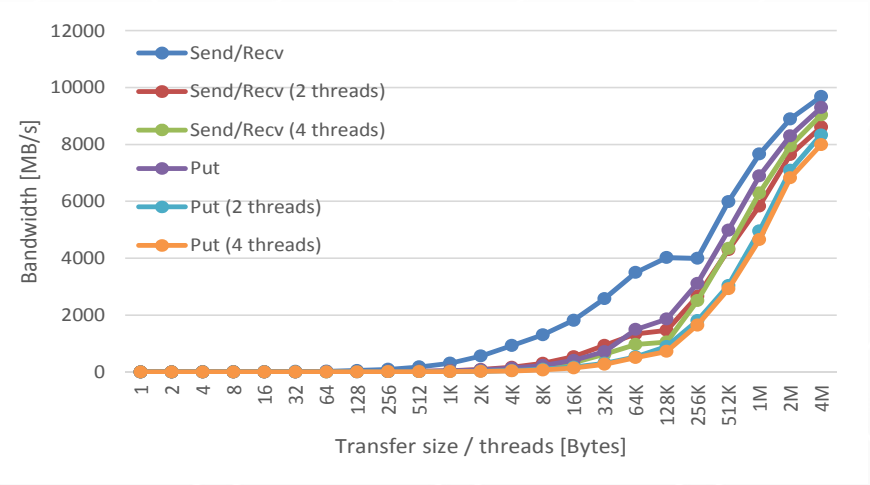
Results on OFP

- Comparison with “Parallel Loop” (PL) and Task-based
- “Put” and “Send/Recv”
- (OMPSS)



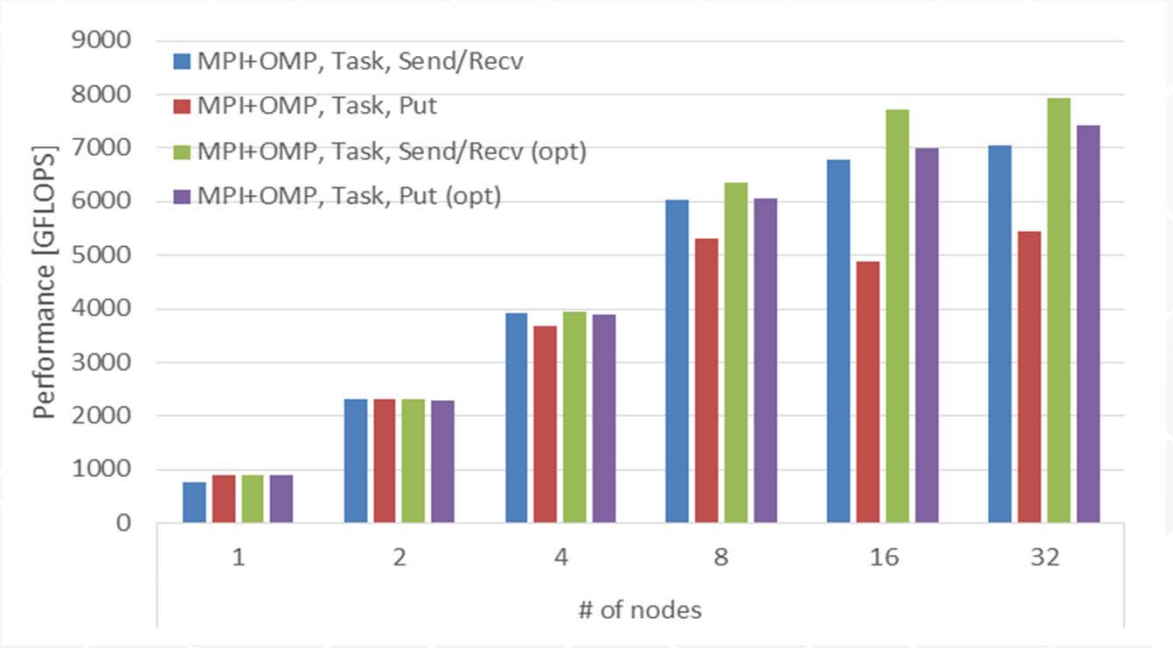
Performance Improvements on OFP

- **Optimization**
  - The communication performance may be improved if all communications are to be delegated to the communication thread.
  - If communication pattern is simple enough to be analyzed by the compiler, some patterns of Put can be transformed into Send/Recv.



- (Opt): all communications are delegated to the communication thread
  - Still, Send/recv

  is better



現在のメニーコア・プログラミングの問題点

- 今のところ、MPI+OpenMPがstandard
  - ノード上に複数プロセス
  - (flat-MPIのほうが速い？)
- コアが多いところでは、taskモデルで書いたほうが効率のいいプログラムがある。
  - Global　syncよりも効率的(な、ときがある) 特に、大規模では。
  - 計算と通信のオーバーラップが可能(であるはず)

- MPIのMPI_THREAD_Multipleが速くない。
  - “End point per process”の問題？
- PGAS(one-sided comm.)がsimpleなはず！？
- ノード間のタスクモデルをどう書くか。
  - OpenMP 4の拡張は、簡単ではない。

並列プログラミング言語XcalableMP規格部会　2018年度活動計画案

- 体制　（継続）

  部会長:　　佐藤 三久(理化学研究所)
  副部会長:　岩下 英俊(富士通)、林 康晴(日本電気)

- 課題
  - 魅力的なプログラミング環境:　使いやすさだけでなく、性能も。部分的な利用。
  - 次世代のメニーコア・演算加速機構に対応した拡張案
    - XcalableMP 2.0, XcalableACC …
- 活動予定
  - 部会（4か月に1度の開催）
    - V2.0の議論を継続:メニーコア向けの動的なタスクモデルへの対応、C++対応
    - 演算加速デバイスのためのXACCも規格化の対象にする。
  - 第6回XMPワークショップの開催
    - 時期　例年通り、予算50万
  - XMP講習会
    - 魅力ある内容を模索しつつ継続する.
    - XACCや部分的な利用を含む、advancedなコースも検討。
  - リファレンス実装
    - Fortran2008対応、C++対応の継続
    - V2仕様の試作評価など
    - ドキュメンテーションが必要
    - ベンチマーク・データの収集、等