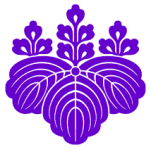


筑波大学
University of Tsukuba

東京大学
THE UNIVERSITY OF TOKYO

JCAHPC

最先端共同HPC基盤施設
ポストT2Kプロジェクト
進捗報告

佐藤三久

筑波大学　計算科学研究センター
最先端共同HPC基盤施設・施設長

最先端共同HPC基盤施設

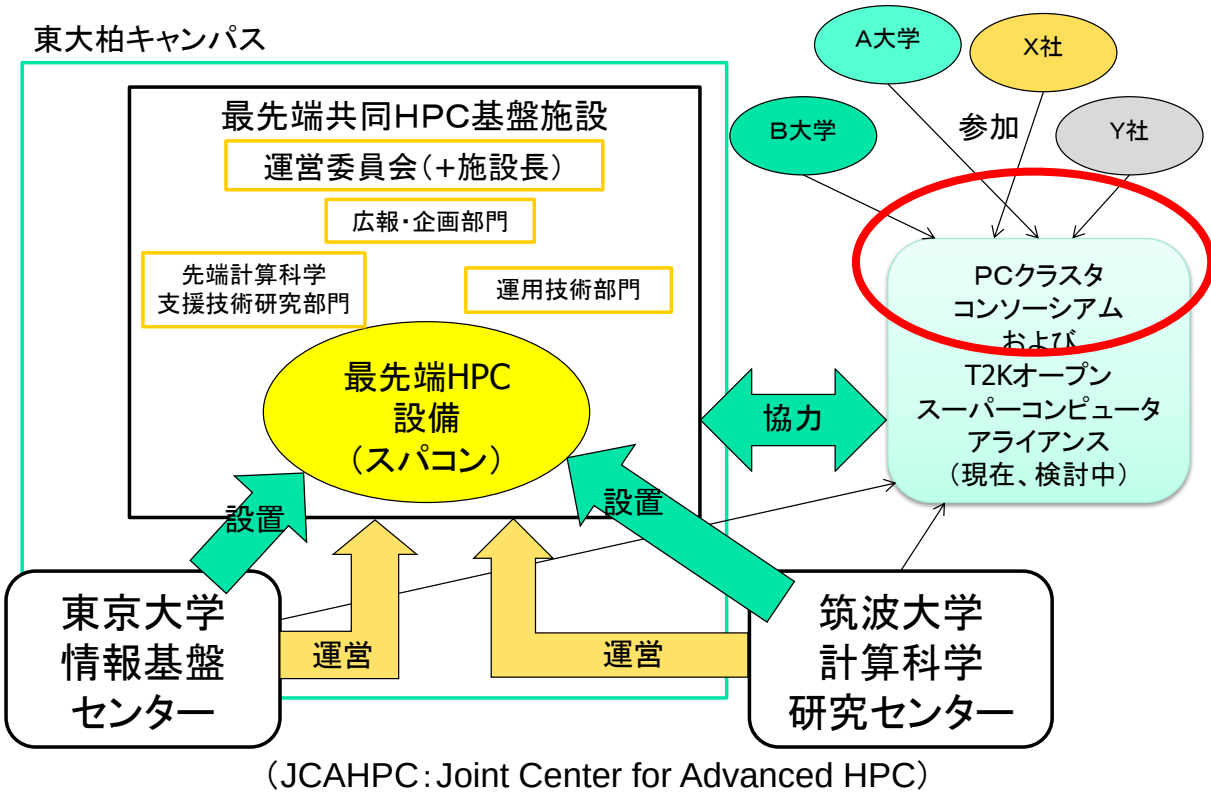
- 最先端共同HPC 基盤施設は、東京大学柏キャンパスの東京大学情報基盤センター内に、筑波大学計算科学研究センターと東京大学情報基盤センターの両機関の教職員が中心となって設計するスーパーコンピュータシステムを設置し、最先端の大規模高性能計算基盤を構築・運営するための組織

- 本施設を連携・協力して運営することにより、最先端の計算科学を推進し、我が国の学術及び科学技術の振興に寄与

- 平成25年3月、筑波大学と東京大学は、「計算科学・工学及びその推進のための計算機科学・工学の発展に資するための連携・協力推進に関する協定」を締結
- 本協定の下に、筑波大学計算科学研究センターと東京大学情報基盤センターは、「最先端共同HPC 基盤施設（JCAHPC：Joint Center for Advanced High Performance Computing）」を設置
- 平成25年度から活動を開始。

最先端共同HPC基盤施設の組織

- 施設長・副施設長
- 運営委員会
- 部門
  - 先端計算科学支援技術研究部門
  - 運用技術部門
  - 広報企画部門
- 協力
  - PCクラスタコンソーシアム
  - T2Kオープンスーパーコンピュータアライアンス



（JCAHPC：Joint Center for Advanced HPC）

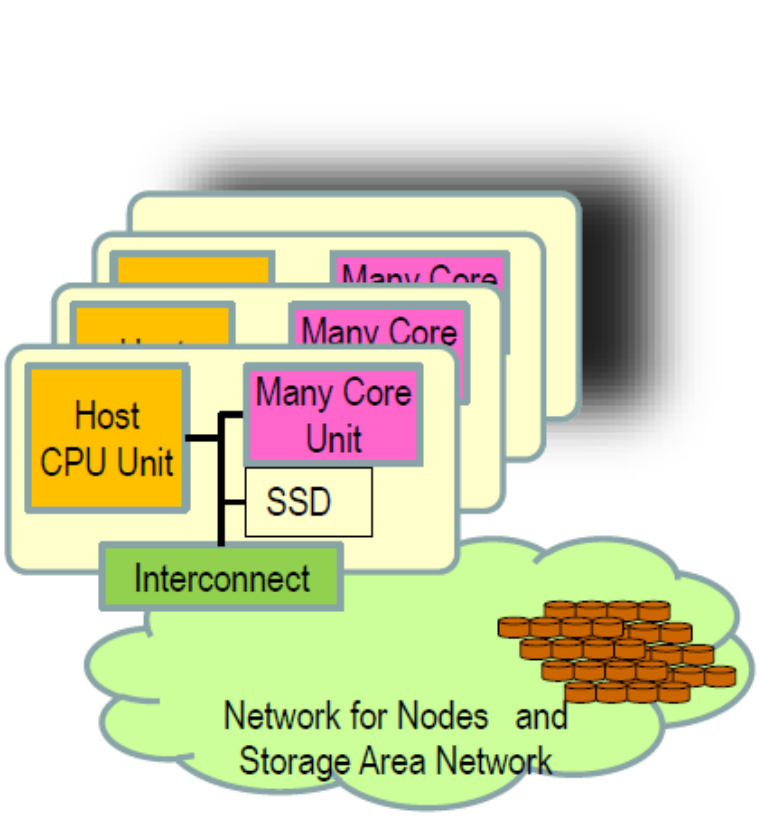
ミッション①　大規模HPCシステムの研究開発

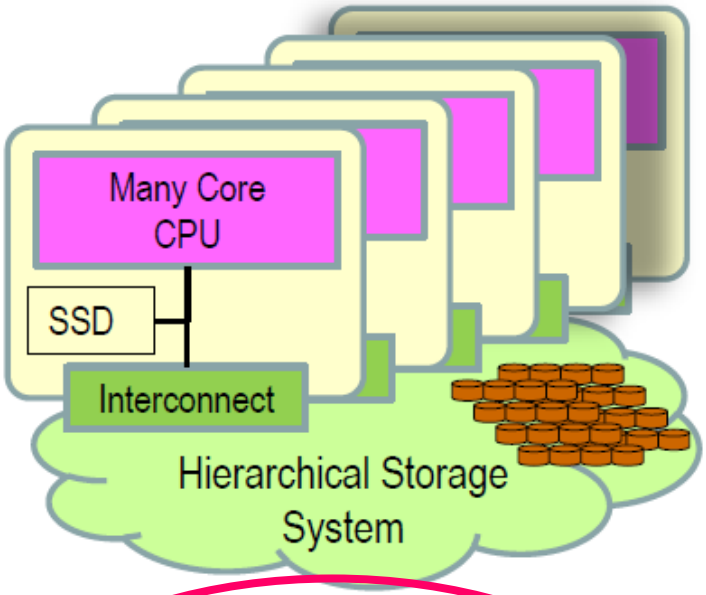
- 先端技術をタイムリーに取り込んだ大規模HPCシステムを設計
  - 現在のHPCシステムの「開発」の重点は、利用可能な高性能なコンポーネントを最適になるように構成すること。
  - これからのHPCシステムの中心的なテクノロジーであるメニーコアを利用し、最先端のシステムを構築

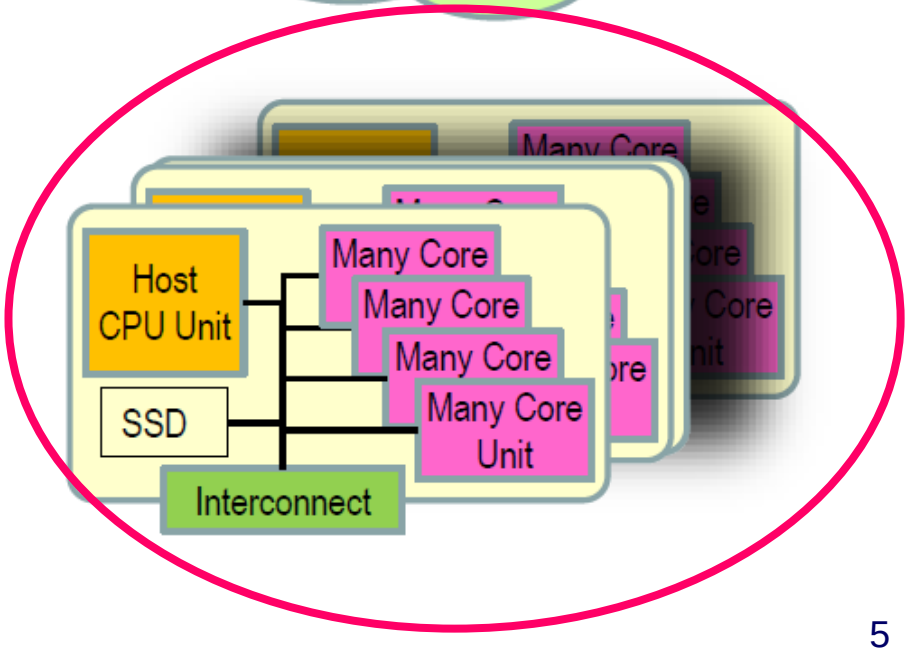
- システムソフトウエアの核となる技術である、OS、プログラミング言語、数値計算ライブラリなどの利用技術を研究開発
  - メニーコア向けのOSカーネル　McKernel
  - メニーコア向けプログラミング言語　XcalableMP
  - その他、検討中、…

- 他の組織とも連携しながら研究開発

アーキテクチャ候補群


Many Core
Many Core
Host
CPU Unit
Many Core
Unit
SSD
Interconnect
Network for Nodes and
Storage Area Network


Many Core
CPU
SSD
Interconnect
Hierarchical Storage
System


Many Core
Host
CPU Unit
Many Core
Many Core
Many Core
SSD
Many Core
Unit
Interconnect

基本仕様（資料招請、導入説明書から）

- (3)　各ノードの汎用CPUは多数の高性能CPUコアを持つメニーコアアーキテクチャによるもので、全てのCPUコアはハードウェア共有メモリ機構により相互に接続されること。
- (4)　各ノードのCPUコアは均一アーキテクチャであり、64ビット拡張されたIA32アーキテクチャに基づくものであること。演算性能は同CPUのみで提供され、それ以外の補助的な演算加速装置等は持たないこと。
- (5)　各ノードのピーク演算性能は3.0 TFLOPS（倍精度浮動小数点）以上であること。
- …
- (9)　全ノードを結合する高性能相互結合網は以下のような階層構造を持つこと。全ノードを2～3程度のグループに分割し、各グループ内の全ノードはフルバイセクションバンド幅のネットワークで結合されること。グループ間についてはグループ内バイセクションバンド幅の20%以上のバイセクションバンド幅を持つこと。
- ….
- 電源容量は、4MW
- ….

システムのデザインポイント(1)

- ノードのプロセッサの選択
  - メニーコア
- 規模
  - 制約条件
    - 予算 (???)
    - 電力　4MW
  - 期待としては、20～30PF

米国NERSC, Coriシステム

- ノード性能3～3.5TFとすると、全体性能は約30PF

運用開始	2016年半ば
CPU Architecture	Knights Landing
Peak Performance	3TF/Node以上
Number of Node	9,300 Node以上
I/O Bandwidth	400 GB/sec以上
Storage	28 PB
Price	70 M$ (70億円)


TODAY AT BERKELEY LAB

News Center | Cafeteria Menu | Flea Market | Contact | Submit | Awards

NERSC Signs Deal for the Next-Generation Supercomputer

April 30, 2014

NERSC has signed a $70 million contract with Cray Inc. for its forthcoming XC supercomputer, which is slated for delivery in 2016 at the new Computational Research and Theory Facility now under construction at Berkeley Lab. The new system will be named "Cori" in honor of biochemist and Nobel Laureate Gerty Cori (left), the first

システムのデザインポイント(2)

- ネットワーク
  - ネットワークトポロジー（IBの場合はFat-Tree？, ...）
  - 階層的な構成（クラスタごとにバンド幅を与える）
- ベンチマークは何にするのか。どのようにベンチマークするのか
- ストレージ
  - グローバルストレージの構成
  - ステージングが必要か？
  - ノードごとのローカルストレージは必要か？
  - ノードへの接続形態
- ジョブスケジューラ
  - ネットワークトポロジーを考慮したスケジューリング
  - 2センターの特性を考慮したマネージメント（課金、リソース割り当て）
  - プロビジョニング（省電力制御、OSの入れ替えmcKernel, 占有利用）

COMA (PACS-IX)

- 筑波大学計算科学研究センターが平成26年4月より運用を開始するスーパーコンピュータシステム（HA-PACSと平行して運用）
- 「高性能汎用計算機高度利用事業 「京」を中核とするHPCIの産業利用支援・裾野拡大のための設備拡充」(平成24年度補正予算）で整備
- JCAHPCシステムのパイロットシステム
- T2K-Tsukubaの後継プラットフォーム
- システム構成
  - 計算ノード：汎用CPU ＋メニーコアプロセッサ (Xeon Phi, MIC)
  - ノード構成
    - CPU x 2: Intel Xeon E5-2670v2
    - MIC x 2: Intel Xeon Phi 7110P
    - Memory: CPU=64GB MIC=16GB
    - Network: IB FDR Full-bisection b/w Fat Tree
  - ノード数： 393
  - ピーク性能： CPU=157.2 TFlops
    MIC=843.8 TFlops
    TOTAL: 1001 TFlops = **1.001 PFLOPS**
- **システムベンダー: Cray Inc.**

東京大学のXeon Phiクラスタ

- メニーコアXeon Phiのシステムソフトウエアの開発、性能評価
- KNCC クラスタ

ノード	SGI Rackable C2110G-RP5
CPU	Intel Xeon E5-2670 v2 x 2 socket (IvyBridge-EP, 2.5GHz 10 core) x2
メモリ	DDR3-1600 128 GB
MIC	Intel Xeon Phi 5110P 8GB
ノード数	32
インタコネクト	Mellanox InfiniBand FDR (Connect X-3) Full bisection
ピーク性能	12.8 TFlops + 32.4 TFlops

- KNSCクラスタ

ノード	NEC LX104Re-G
CPU	Intel Xeon E5-2680 v2 x 2 socket (IvyBridge-EP, 2.8GHz 10 core) x2
メモリ	DDR3-1866 64 GB
MIC	Intel Xeon Phi 5110P 8GB
ノード数	64
インタコネクト	InfiniBand FDR x2 (Mellanox Connect-IB 2port), Full bisection
ストレージ	DDN SFA7700, 96TByte
ピーク性能	28.6 TFlops + 64.7 TFlops

XcalableMP on Xeon Phi

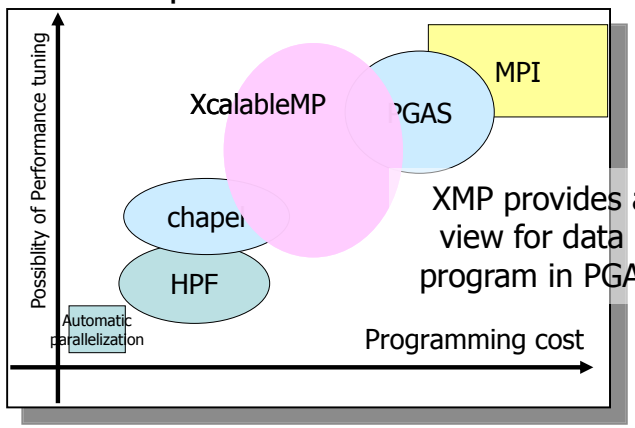
- 現在、筑波大のHPCS研究室において、XcalableMP処理系のXeon Phi (KNC)への設計・実装を進めている。

- XcalableMPの利点
  - メニーコアでは、ノードのコアが多いため(Xeon Phiの場合は60)、MPI+OpenMPのHybridプログラミングが必須になり、2重のプログラミングのコストが高くなる。XcalalableMPにより、1つのプログラミングモデルで、カバーできる。
  - ランタイムを効率的にデータにアクセス・通信できるように設計することにより、単一のモデルでスケーラブルな実装が可能。
  - 各コアでの局所性を高めるプログラミングモデルの提供

- 科学技術計算で典型的な計算パターンであるステンシル計算を対象に、評価実験
  - CCGrid2014で、発表 （by 池井さん@筑波大＆Intel）

XcalableMP(XMP) http://www.xcalablemp.org

- What's XcalableMP (XMP for short)?
  - A PGAS programming model and language for distributed memory , proposed by **XMP Spec WG**
  - XMP Spec WG is a special interest group to design and draft the specification of XcalableMP language. It is now organized under **PC Cluster Consortium**, Japan. Mainly active in Japan, but open for everybody.
- Project status (as of Nov. 2013)
  - XMP Spec **Version 1.2** is available at XMP site. new features: mixed OpenMP and OpenACC , libraries for collective communications.
  - Reference implementation by U. Tsukuba and Riken AICS: **Version 0.7 (C and Fortran90)** is available for PC clusters, Cray XT and K computer. Source-to- Source compiler to code with the runtime on top of MPI and GasNet.



- Language Features
  - Directive-based language extensions for Fortran and C for PGAS model
  - Global view programming with global-view distributed data structures for data parallelism
    - SPMD execution model as MPI
    - pragmas for data distribution of global array.
    - Work mapping constructs to map works and iteration with affinity to data explicitly.
    - Rich communication and sync directives such as "gmove" and "shadow".
    - Many concepts are inherited from HPF
  - Co-array feature of CAF is adopted as a part of the language spec for local view programming (also defined in C).

Code example

```
int array[YMAX][XMAX];

#pragma xmp nodes p(4)
#pragma xmp template t(YMAX)
#pragma xmp distribute t(block) on p
#pragma xmp align array[i][*] to t(i)

main(){
  int i, j, res;
  res = 0;

#pragma xmp loop on t(i)  reduction(+:res)
  for(i = 0; i < 10; i++)
    for(j = 0; j < 10; j++){
      array[i][j] = func(i, j);
      res += array[i][j];
    }
}
```

data distribution

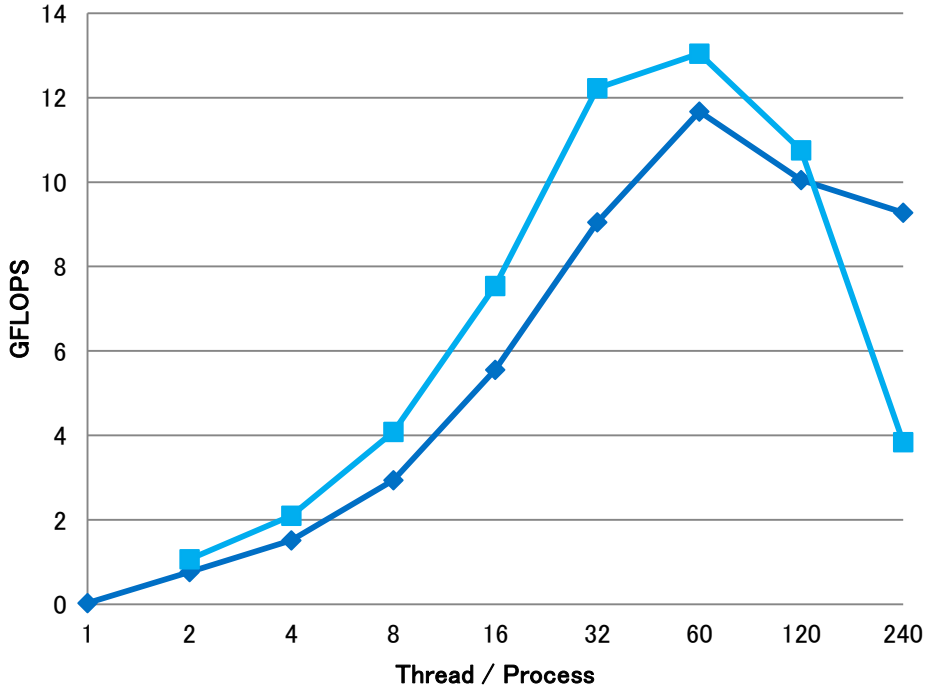
add to the serial code : incremental parallelization

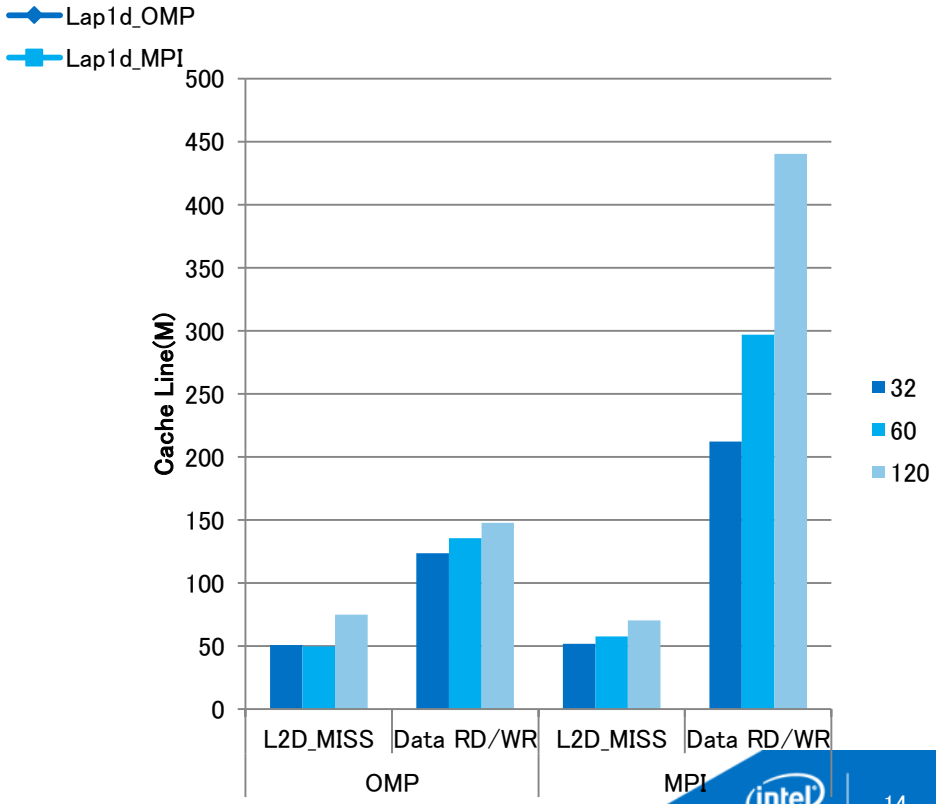
work sharing and data synchronization

OpenMP Laplace Benchmark

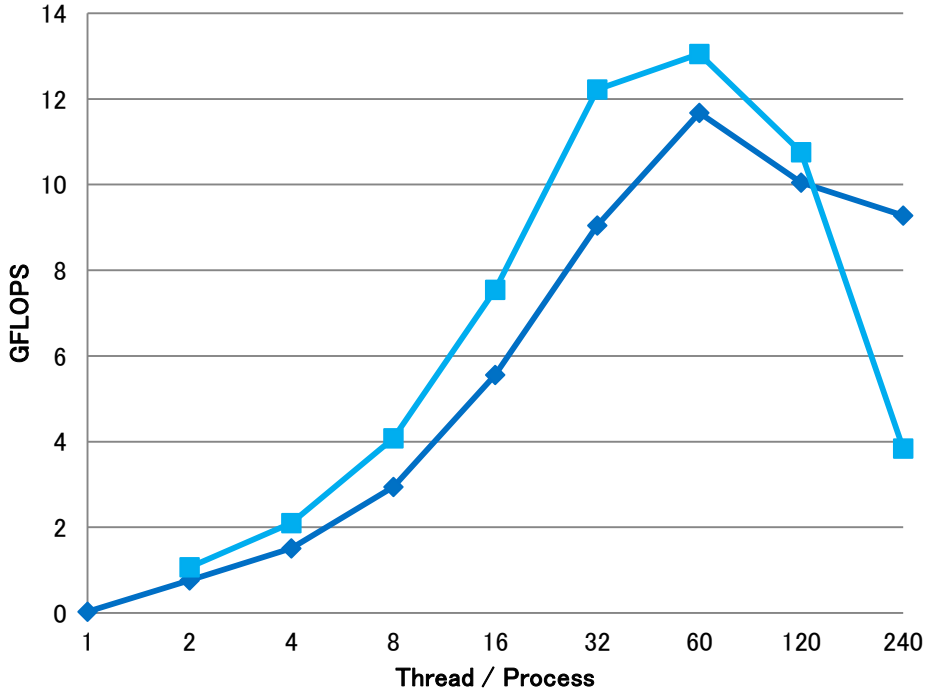
```
 48
 49 #pragma omp parallel private(k,x,y)
 50 {
 51     for(k = 0; k < NITER; k++){
 52         /* old <- new */
 53 #pragma omp for
 54         for(x = 1; x <= XSIZE; x++)
 55           for(y = 1; y <= YSIZE; y++)
 56             uu[x][y] = u[x][y];
 57         /* update */
 58 #pragma omp for
 59         for(x = 1; x <= XSIZE; x++)
 60           for(y = 1; y <= YSIZE; y++)
 61             u[x][y] = (uu[x-1][y] + uu[x+1][y]
+ uu[x][y-1] + uu[x][y+1])/4.0;
 62     }
 63 }
```

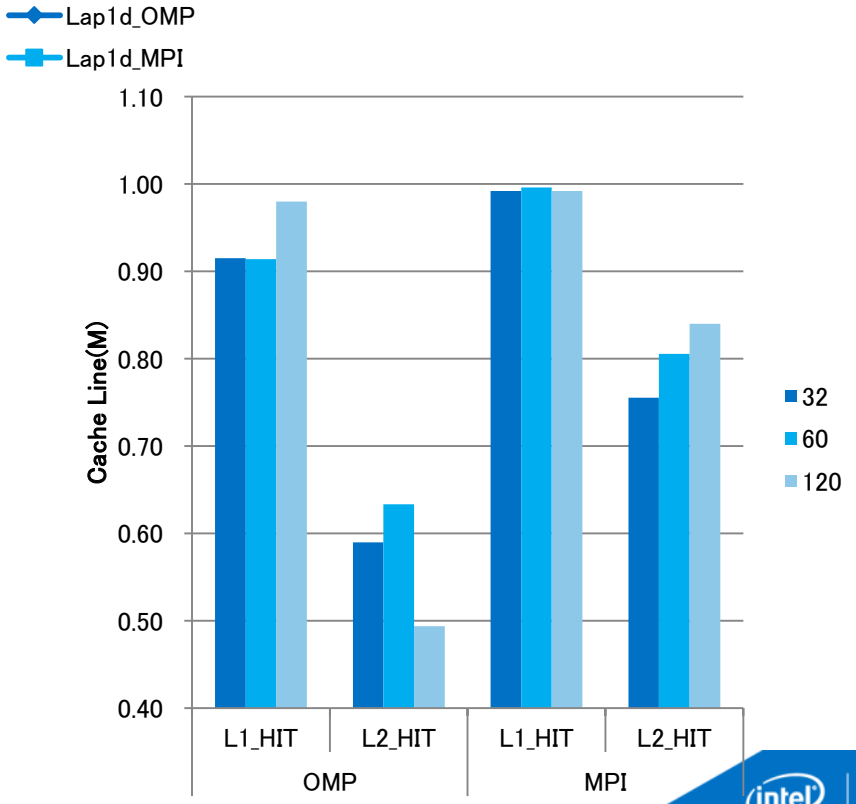
Laplace Result OpenMP vs MPI on Phi





Laplace Result OpenMP vs MPI on Phi





Xeon Phi向けXcalableMPの実装

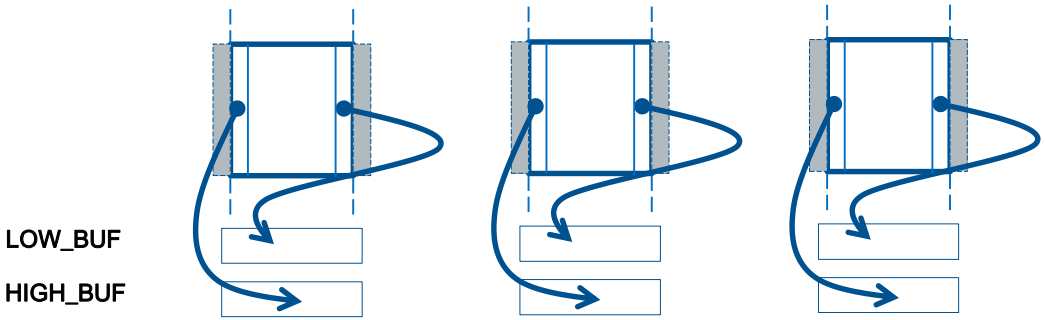
- Xeon Phiのノード内での実装の選択肢
  - MPIを使う（現状、何もしない）
  - mmapによる共有メモリを確保を行い、これを通して通信（←今回）
  - プログラム変換によるスレッドモデルへのトランスレート（UPCでやっている。現在、実装計画中）
  - PVASによる実装（理研AICS グループと共同研究）

XMP version of Laplace Benchmark

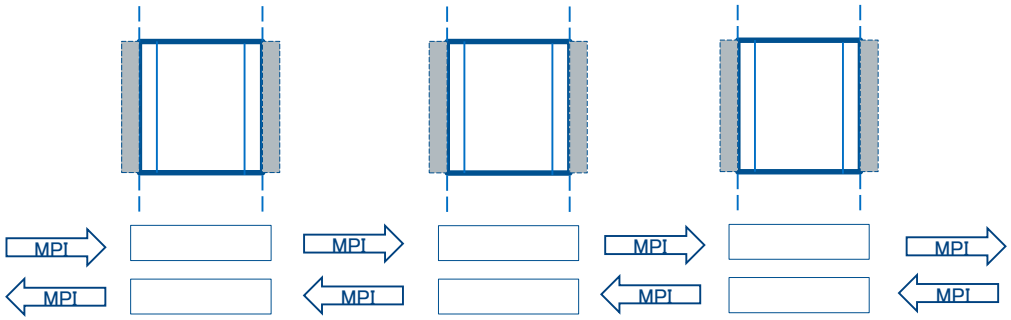
```
16 double u[XSIZE][YSIZE], uu[XSIZE][YSIZE];

19 #pragma xmp nodes p(*)
20 #pragma xmp template t(0:(10000)-1)
21 #pragma xmp distribute t(block) onto p
22 #pragma xmp align u[j][*] with t(j)
23 #pragma xmp align uu[j][*] with t(j)
24 #pragma xmp shadow uu[1][0]

...

86      /* old <- new */
87 #pragma xmp loop (x) on t(x)
88      for(x = 1; x < XSIZE-1; x++)
89        for(y = 1; y < YSIZE-1; y++)
90          uu[x][y] = u[x][y];
91
92 #pragma xmp reflect (uu)
93
94      /* update */
95 #pragma xmp loop (x) on t(x)
96      for(x = 1; x < XSIZE-1; x++)
97        for(y = 1; y < YSIZE-1; y++)
98          u[x][y] = (uu[x-1][y] + uu[x+1][y] + uu[x][y-1] + uu[x][y+1])/4.0;
```

Ghost region update using MPI sendrecv

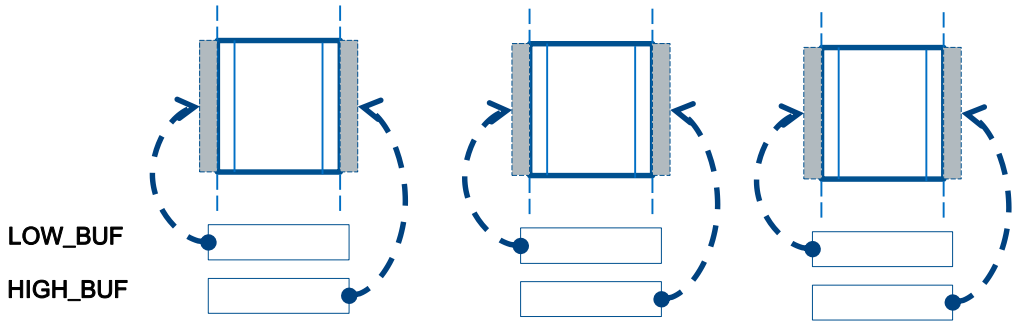
(1) Pack



(2) Exchange

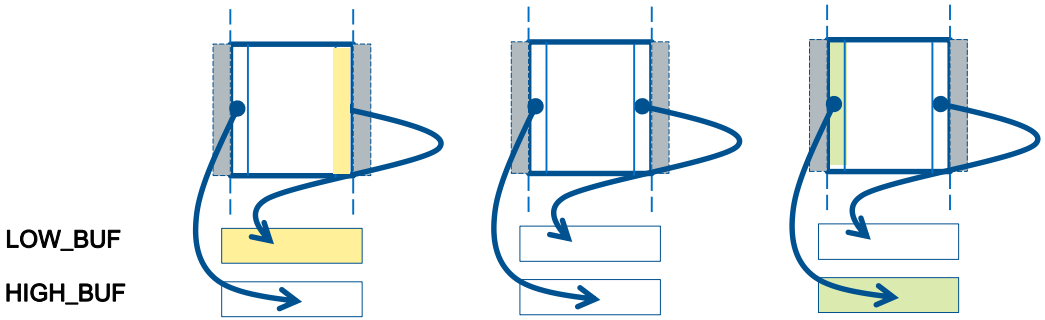


(3) Unpack

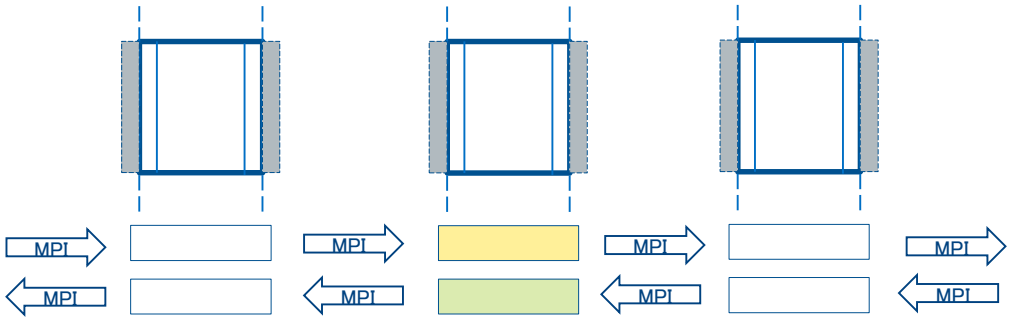


Ghost region update using MPI sendrecv

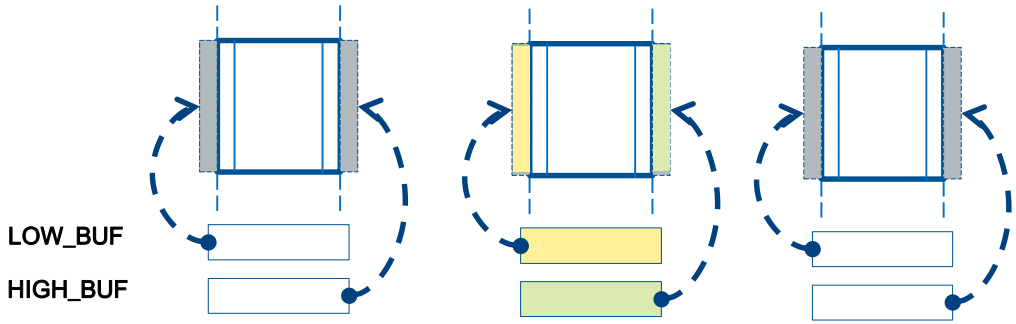
(1) Pack



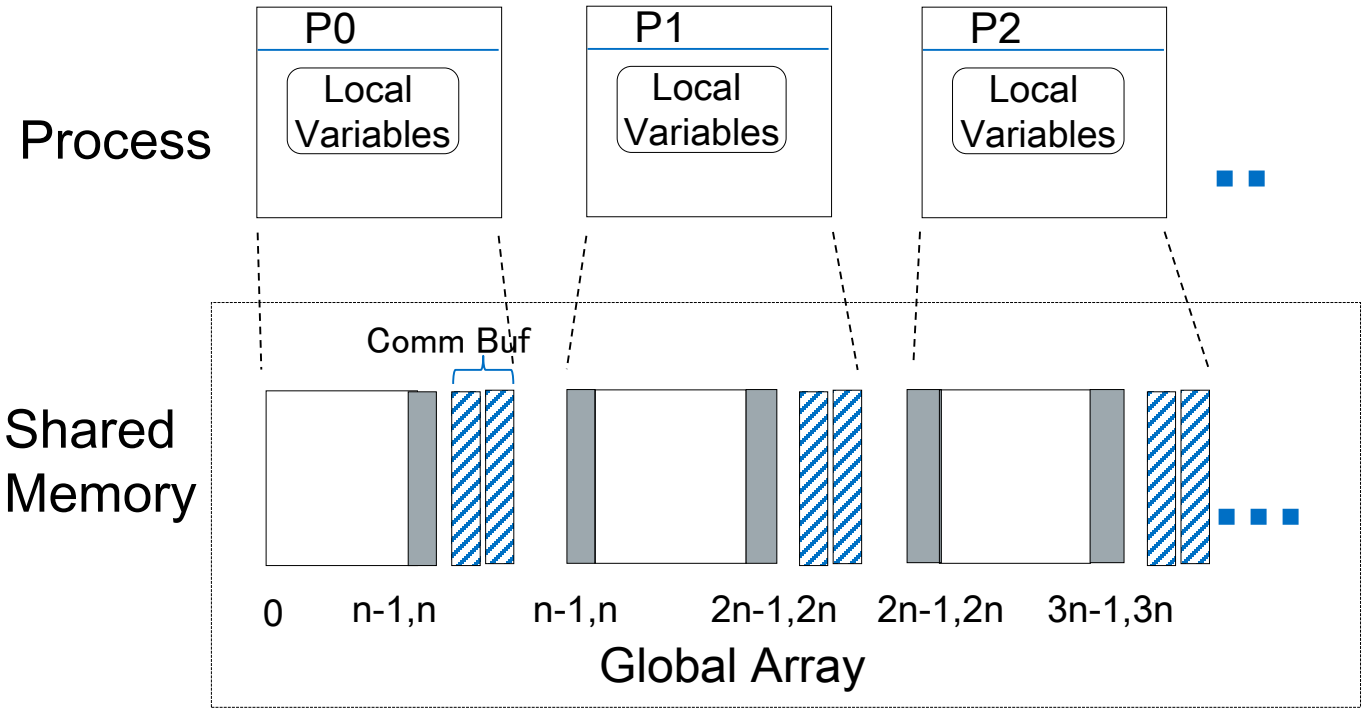
(2) Exchange



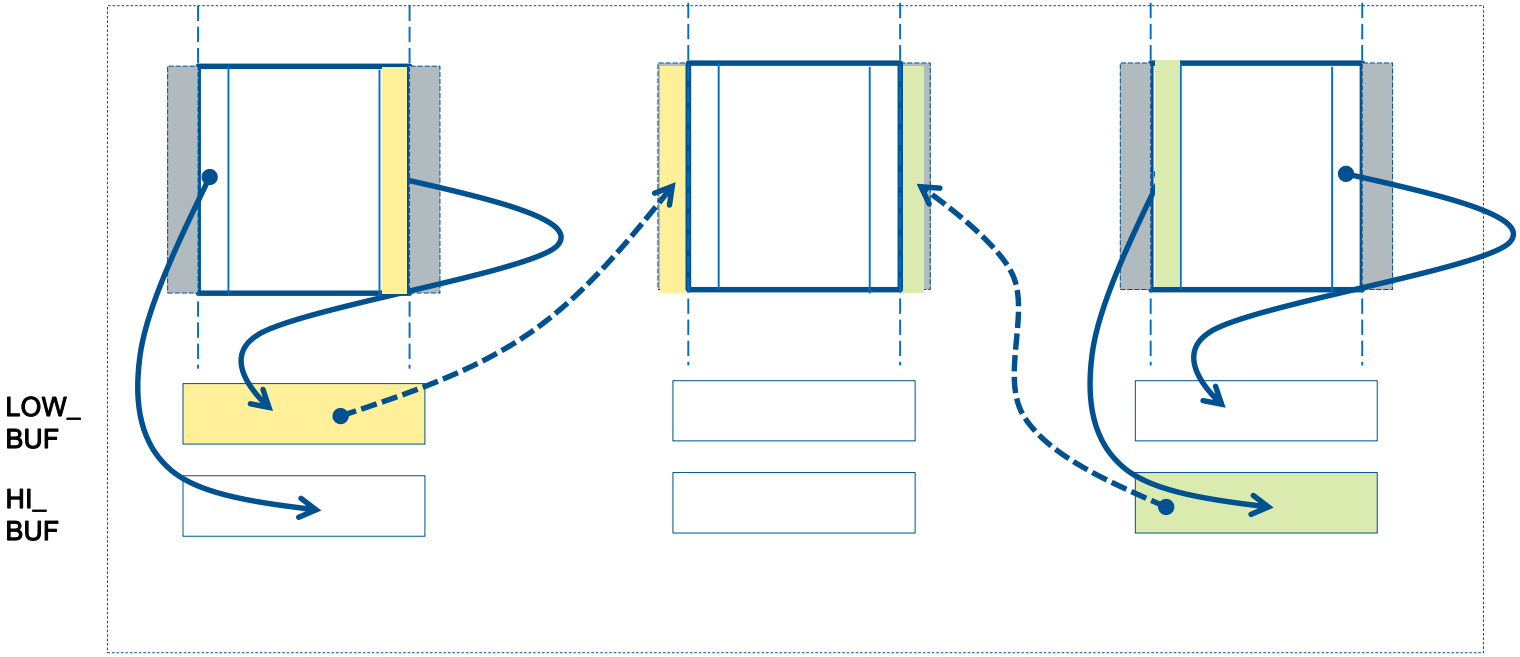
(3) Unpack



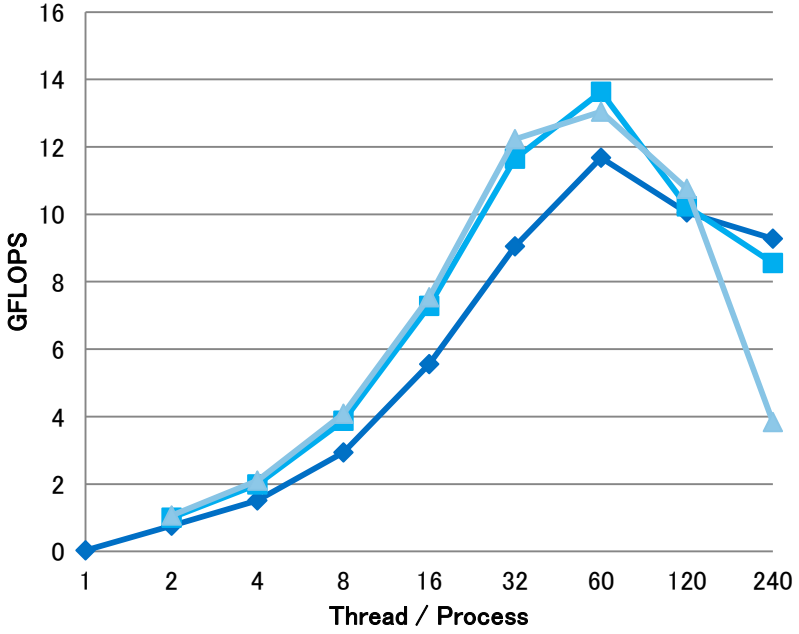
XMP Global Array Allocation

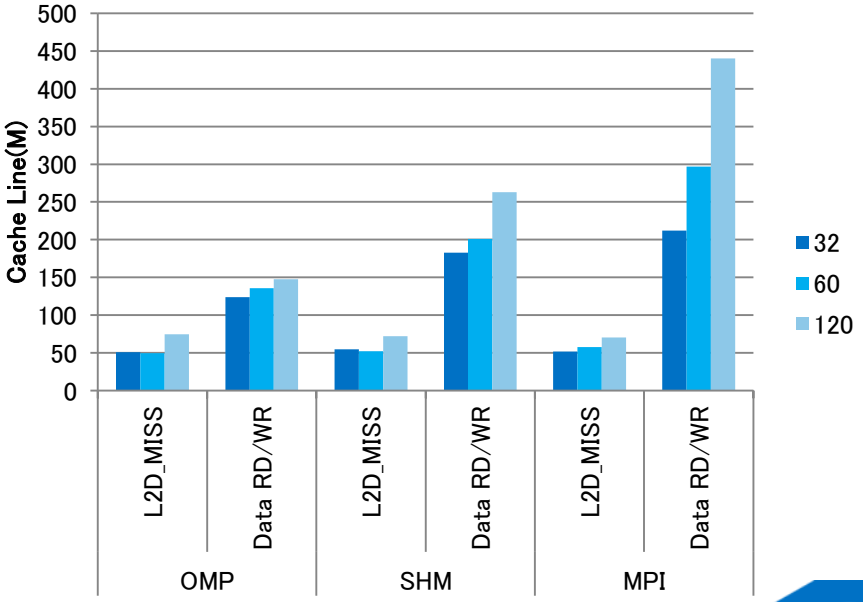


Ghost region update on SHM



Laplace 1D Benchmark Results

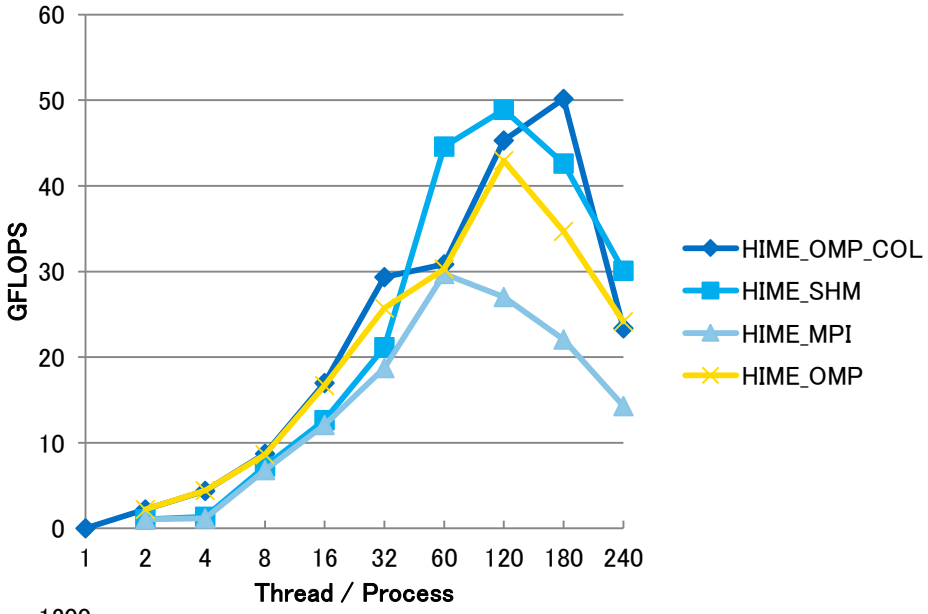


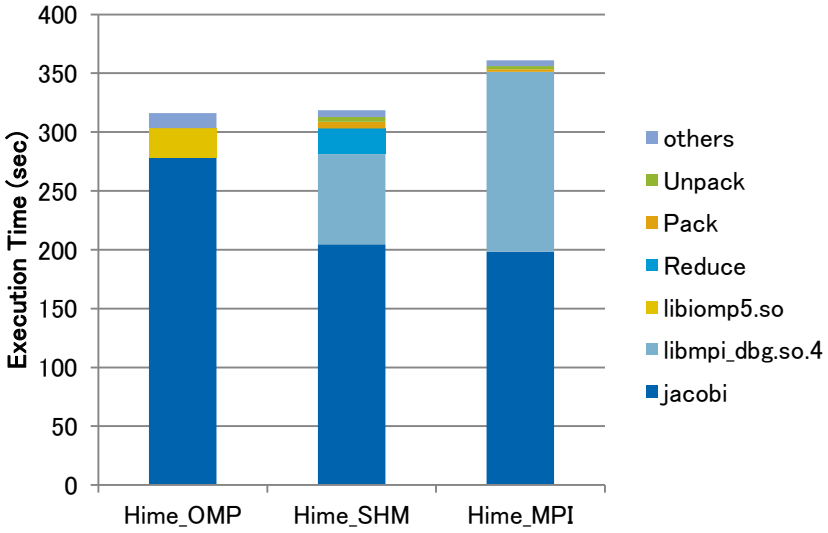


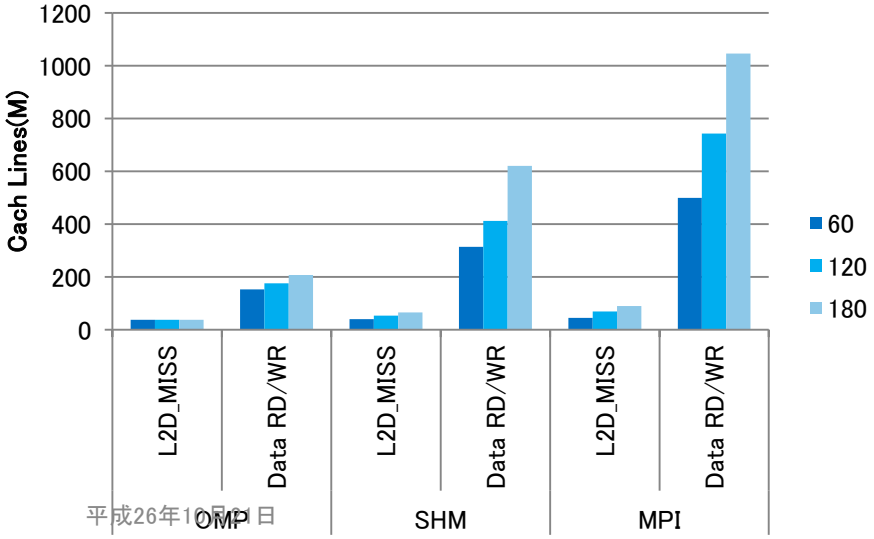
2D Distribution Himeno Benchmark

```
82 static float  p[257][257][513];
83 static float  a[4][257][257][513], b[3][257][257][513],
84              c[3][257][257][513];
85 static float  bnd[257][257][513];
86 static float  wrk1[257][257][513], wrk2[257][257][513];
87
88 #pragma xmp nodes n(6,10)
89 #pragma xmp template t(0:512,0:256,0:256)
90 #pragma xmp distribute t(*,block, block) onto n
91 #pragma xmp align [i][j][*] with t(*, j, i) :: p,bnd,wrk1,wrk2
92 #pragma xmp align [*][i][j][*] with t(*, j, i) :: a,b,c
93 #pragma xmp shadow p[1][1][0]
...
231 #pragma xmp reflect (p)
232 #pragma xmp loop(i,j) on t(*,j,i) reduction (+:gosa)
233     for(i=1 ; i<imax-1 ; i++)
234       for(j=1 ; j<jmax-1 ; j++)
235         for(k=1 ; k<kmax-1 ; k++){
236           s0 = a[0][i][j][k] * p[i+1][j  ][k  ]
237              + a[1][i][j][k] * p[i  ][j+1][k  ]
238              + a[2][i][j][k] * p[i  ][j  ][k+1]
239
             + b[0][i][j][k] * ...
```

Himeno Benchmark Results







コメント・これからの計画

- Our overall performance advantage against OpenMP is 16.8 % for Lap1D and 13.8 % for Hime2D.

  ① **Blocking Effect of Global Arrays**: By dividing global arrays into sub-arrays detached each others, we could reduce 22.2 % and 27.3 % of CPU calculation time on Laplace and Himeno respectively.

  ② **Efficient Ghost Region Exchange**: By changing MPI sendrecvs to direct memory copy on SHM, we could reduce memory traffics about 32.3% to 44.6% and get the maximum performance gain 62.9 %.

- KNCは、なかなか謎のところが多いので、KNLでの性能を期待。
- 現在、ノード間も含めたXcalableMPの性能評価中
- Xeon Phi向けの新しいRuntimeの設計・リリース
  - ベクトル化支援も含めて

ミッション② 大規模HPCシステムの調達・設置・運転

- 東大の柏キャンパスに施設を設置し、設計したスパコンシステムを共同調達・共同運転を行う。

- 筑波大と東大の両センターは計算利用量（計算時間×ノード数）をもってスパコンシステムを案分し、センターごとに運用、利用プログラムを実施。
  - 管理等のコストが削減されるだけでなく、各センターが単独でスーパーコンピュータシステムを保有する場合より大規模な計算が可能

- このような施設を作り、スーパーコンピュータを共同運営・管理するのは国内初めての試み

スケジュール

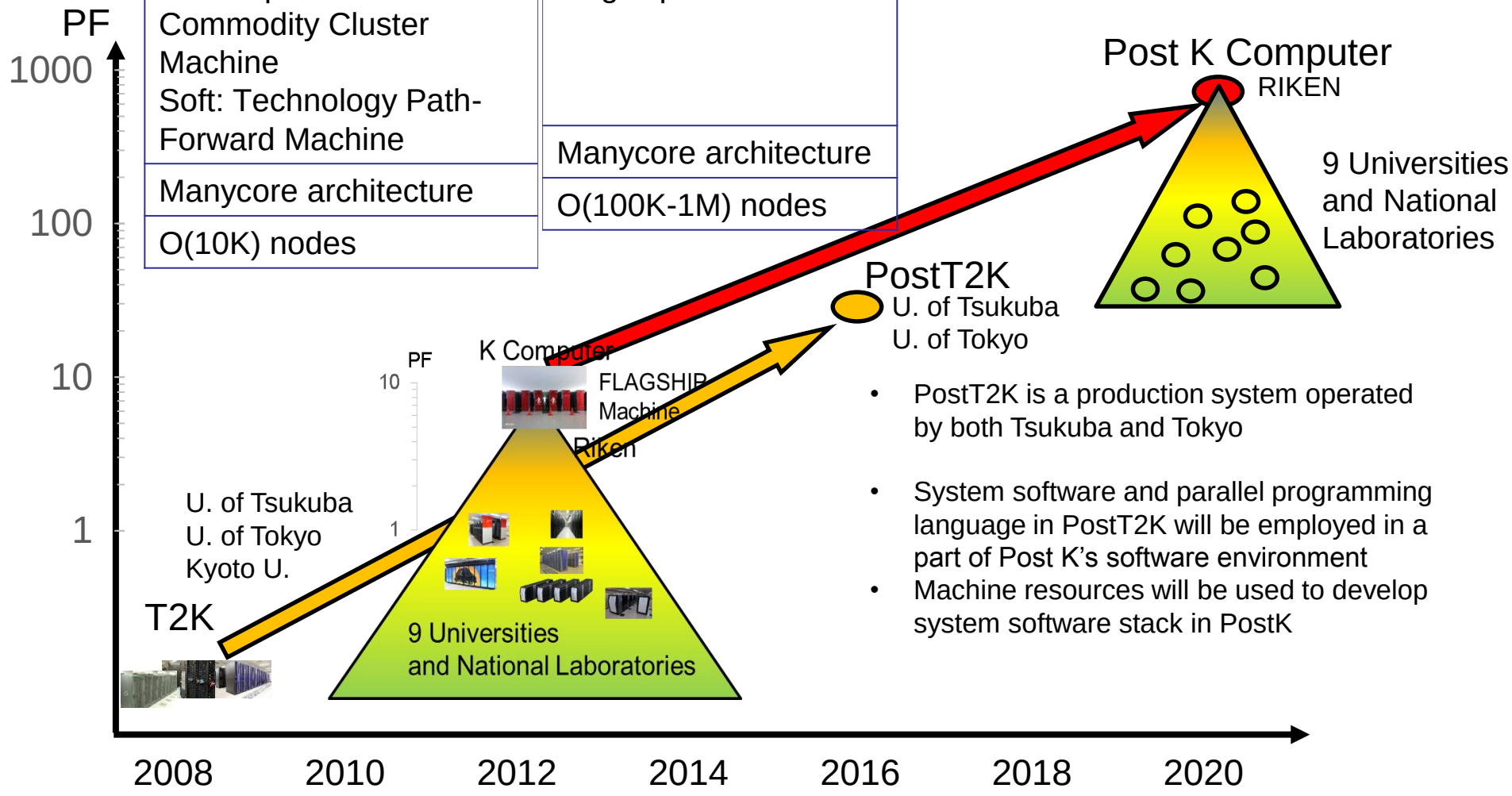
- 残念ながら、仕様を満たすプロセッサのデリバリーの遅れで、スケジュールが遅延の見込み

- 10月　仕様書原案説明会　-> 1月中
- 2月　入札説明会　-> 6月
- 3月上旬　応札〆切　-> 7月
- 4月上旬　開札　-> 8月
- 2016年　6月　運用開始

これは以前のスケジュール

Towards the Next Flagship Machine

PostT2K	PostK
Arch: Upscale Commodity Cluster Machine Soft: Technology Path-Forward Machine	Flagship Machine
Manycore architecture	Manycore architecture
O(10K) nodes	O(100K-1M) nodes



- PostT2K is a production system operated by both Tsukuba and Tokyo
- System software and parallel programming language in PostT2K will be employed in a part of Post K's software environment
- Machine resources will be used to develop system software stack in PostK

おわりに

- 残念ながら、仕様を満たすプロセッサのデリバリーの遅れで、スケジュールが遅延の見込み
  - 実機での一部ベンチマークの可能性
- 仕様については、最終段階にある。
  - 特にファイルシステム関連が焦点
- McKernelおよびXcalableMPの研究開発を行っている。
  - ポスト京につながるソフトウエア
- そろそろ、運用形態についての議論を始めるべき
  - 運用技術部門を動かす時期

Thank you for your attention