並列言語 XcalableMP の生産性と性能評価

中尾昌広（筑波大学計算科学研究センター），小田嶋哲哉，朴泰祐，佐藤三久

研究背景

分散メモリアプリケーションの多くは MPI を用いて作成されている

- データの送受信を関数を用いて詳細に記述（MPI_Send/Recv）
- データ分散やループの並列処理を手動で記述

} 生産性の低さが問題となっている

並列プログラミング言語 XcalableMP

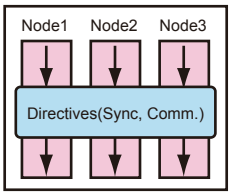
- C99 と Fortran95 の並列拡張
  - 指示文を用いることで逐次プログラムから簡易な並列化が可能
  - 片側通信の記法を提供（Fortran 版 XcalableMP は Coarray Fortran の上位互換）
  - コードの書き換えや教育に要するコストの削減
- MPI および OpenMP，OpenACC との相互運用性
- PC クラスタコンソーシアムの XcalableMP 規格部会が仕様を決定
  - 大学，研究機関，メーカの有志で構成

仕様と実装の進捗について

- XcalableMP 仕様書 version 1.2 を 11 月にリリース
  - OpenMP 指示文と XcalableMP 指示文の混在利用の動作を定義
  - 組み込み関数を導入（分散配列の転置など）
  - *http://www.xcalablemp.org*
- Omni XcalableMP Compiler version 0.7 を 11 月にリリース
  - 筑波大学 計算科学研究センターと理化学研究所 計算科学研究機構が開発
  - *http://www.hpcs.cs.tsukuba.ac.jp/omni-compiler/xcalablemp/*
  - プロファイリングツール Scalasca，tlog をサポート
  - Linux クラスタ，Cray マシン（XK6，XE6），京，FX10 などで動作
    - 京への最適化：高速 RDMA インターコネクトを利用
    - SX，地球シミュレータ，SR16000，Blue Gene/Q などにも移植中
- Omni OpenACC Compiler を 11 月にリリース

言語の特徴

- Single Program, Multiple Data streams



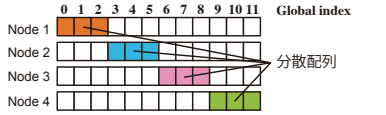
- データの送受信や同期，ループ文の並列化などの処理は，指示文もしくは拡張構文の箇所で発生する

Global-view

- 配列a[12]を4ノードに割り当てる場合

```
int a[12];
#pragma xmp nodes p(4)
#pragma xmp template t(0:11)
#pragma xmp distribute t(block) onto p
#pragma xmp align a[i] with t(i)
```

Data mapping




```
#pragma xmp loop on t(i) reduction(+s)
for(i = 0; i < 12; i++) {
    a[i] = func(i);
    s += a[i];}
```

Work mapping

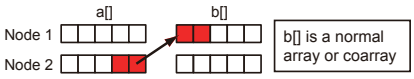
Local-view

- 片側通信のためのC言語の拡張構文

```
array_name[start:length[:step]]:[node_number]
```

- Coarrayの利用方法

```
double a[5]:[*];                    // Declaration
    :
if(me == 2)
   b[0:2]:[1] = a[3:2];             // Put Operation
```



生産性と性能について

生産性の評価

- 各ベンチマーク実装に対する行数の比較
- XcalableMP の方が少ない行数で記述可能
- 姫野ベンチマークのプログラム例（Fortran 版 XcalableMP）
  - 近傍の値を用いて自身の値を更新するアプリケーション
  - 逐次ソースに指示文を追加するのみで並列化が可能

	Laplace Solver	姫野ベンチマーク	NAS Parallel Benchmarks			HPC Challenge Benchmarks			
			Embarrassingly Parallel	Integer Sort	Conjugate Gradient	HPL	RandomAccess	FFT (only kernel)	STREAM
XcalableMP	45	137	141	704	567	306	250	70	66
MPI	158	380	187	711	1,129	8,800	938	101	329

```
!$xmp nodes n(2,2)
!$xmp template t(mimax,mjmax,mkmax)          分散配列の定義
!$xmp distribute t(*,block,block) onto n
!$xmp align (*,j,k) with t(*,j,k) :: p
!$xmp shadow p(0,1,1)                        袖領域の定義
   :
!$xmp reflect (p)                            袖領域の同期
!$xmp loop (J,K) on t(*,J,K)                 ループ文の並列化
do K = 2, kmax-1
   do J = 2, jmax-1
      do I = 2, imax-1
         S0 = a(I,J,K,1) * p(I+1,J,K) + ...
         SS = (S0*a(I,J,K,4)-p(I,J,K)) * ...
         GOSA = GOSA + SS * SS
            :
      enddo
   enddo
enddo
!$xmp reduction (+:GOSA)                     集約演算
```

性能の評価 on HA-PACS