「並列Fortranの現状と展望」
~Coarrayは救世主なのか?~

話題提供

核融合科学研究所
基礎物理シミュレーション研究系
坂上仁志

Coarrayが使えるコンパイラ

- 富士通
  - ようやくリリースされる見込み.
- Intel
  - ifort –coarray=shared：共有メモリ上でのみ動作する.
  - ifort –coarray=distributed：分散メモリ上で動作する.
- GCC Fortran
  - GCC 5でサポートされた.
- Omni XcalableMP/Fortran
  - version 1.0が2016/3/10にリリースされた.
  - version 0.7でテストした.

MPIとCoarrayプログラム

```
integer parameter :: lx=2048, ly=2048, lZ=2048
integer parameter :: ny=16, nz=16
integer parameter :: lsy=ly/ny, lsz=lz/nz
real*8 :: physval(6,lx,0:lsy+1,0:lsz+1)

do iz = 1, lsz
  do iy = 1, lsy
    do ix = 1, lx
      ...
end do
```

Z neighboring communication

```
call MPI_ISEND( physval(1,1,1,1), ..., lrkzm,
call MPI_ISEND( physval(1,1,1,lsz), ..., lrkzp,
call MPI_IRECV( physval(1,1,1,lsz+1), ..., lrkzp,
call MPI_IRECV( physval(1,1,1,0), ..., lrkzm,
call MPI_WAITALL( ...
```

Y neighboring communication

```
buff1(:,:,:) = physval(:,:,1,:)
buff2(:,:,:) = physval(:,:,lsy,:)
call MPI_ISEND( buff1, ..., lrkym,
call MPI_ISEND( buff2, ..., lrkyp,
call MPI_IRECV( buff1, ..., lrkyp,
call MPI_IRECV( buff2, ..., lrkym,
call MPI_WAITALL( ...
physval(:,:,lsy+1,:) = buff1(:,:,:)
physval(:,:,0,:) = buff2(:,:,:)
```

```
integer parameter :: lx=2048, ly=2048, lZ=2048
integer parameter :: ny=16, nz=16
integer parameter :: lsy=ly/ny, lsz=lz/nz
real*8 :: physval(6,lx,0:lsy+1,0:lsz+1)[:]

do iz = 1, lsz
  do iy = 1, lsy
    do ix = 1, lx
      ...
end do
```

PUT communication

```
physval(:,:,:,lsz+1)[linzm] = physval1(:,:,:,1)
physval(:,:,:,0)[linzp] = physval1(:,:,:,lsz)
sync all

physval(:,:,lsy+1,:)[linym] = physval1(:,:,1,:)
physval(:,:,0,:)[linyp] = physval1(:,:,lsy,:)
sync all
```

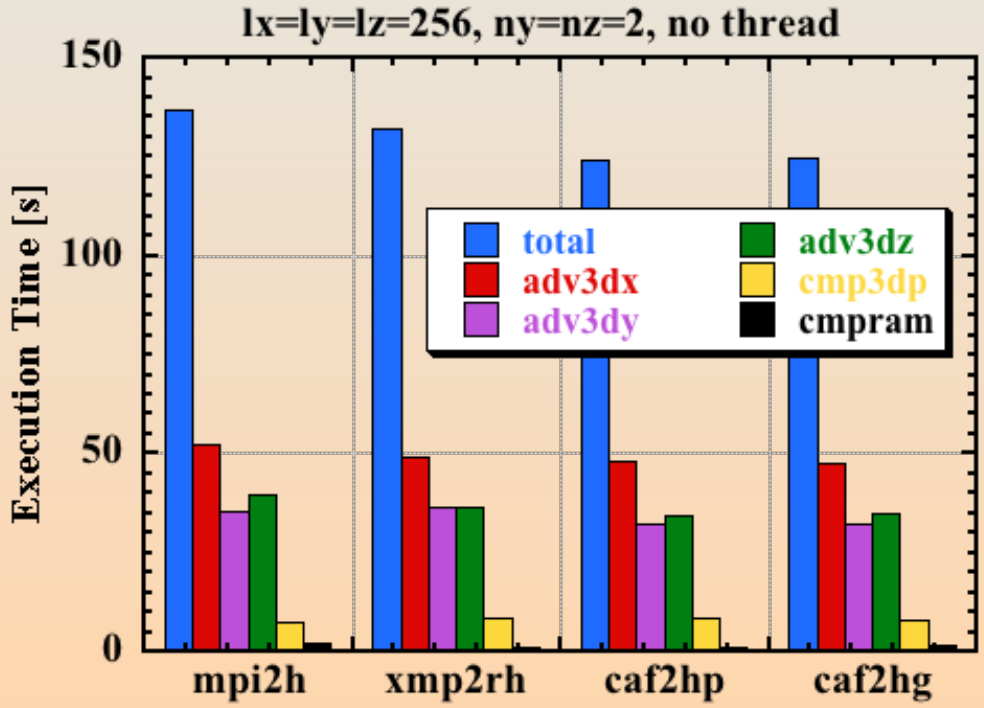
GET communication

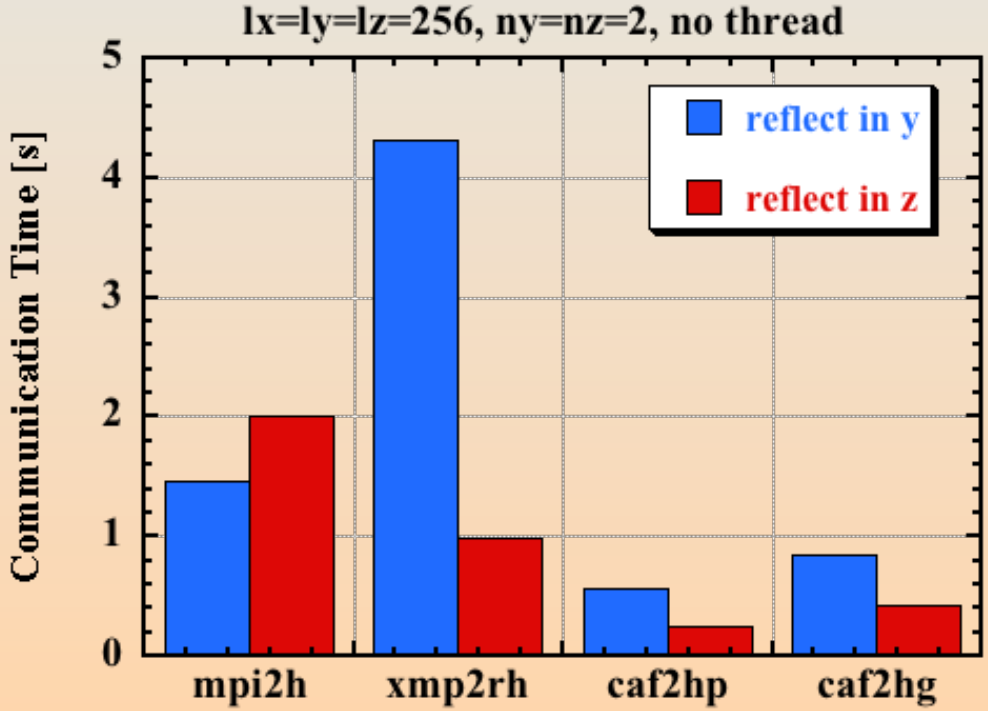
```
sync all
physval(:,:,:,lsz+1)= physval1(:,:,:,1)[linzp]
physval(:,:,:,0)= physval1(:,:,:,lsz)[linzm]

sync all
physval(:,:,lsy+1,:) = physval1(:,:,1,:)[linyp]
physval(:,:,0,:) = physval1(:,:,lsy,:)[linym]
```

PCクラスターでの実行結果

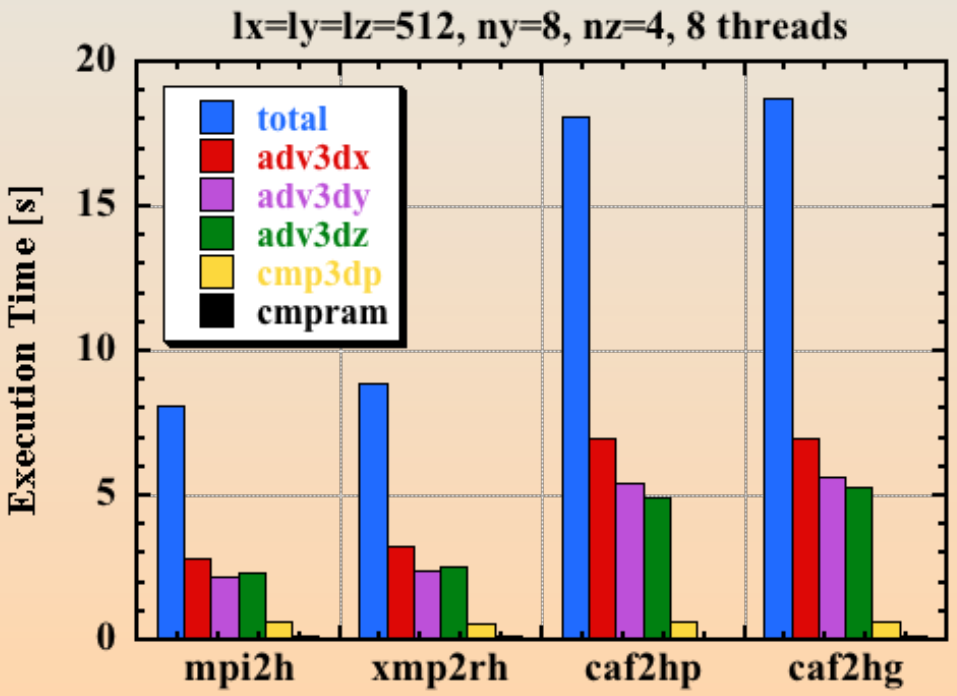
- XMP/Fのcoarray通信は,GASNetライブラリで実装されているが,GASNetはMPI conduitを使っている.
- MPI版,XMPグローバルビュー版およびcoarray版は,ほぼ同等の性能である.
  - むしろ,通信性能は改善されている.

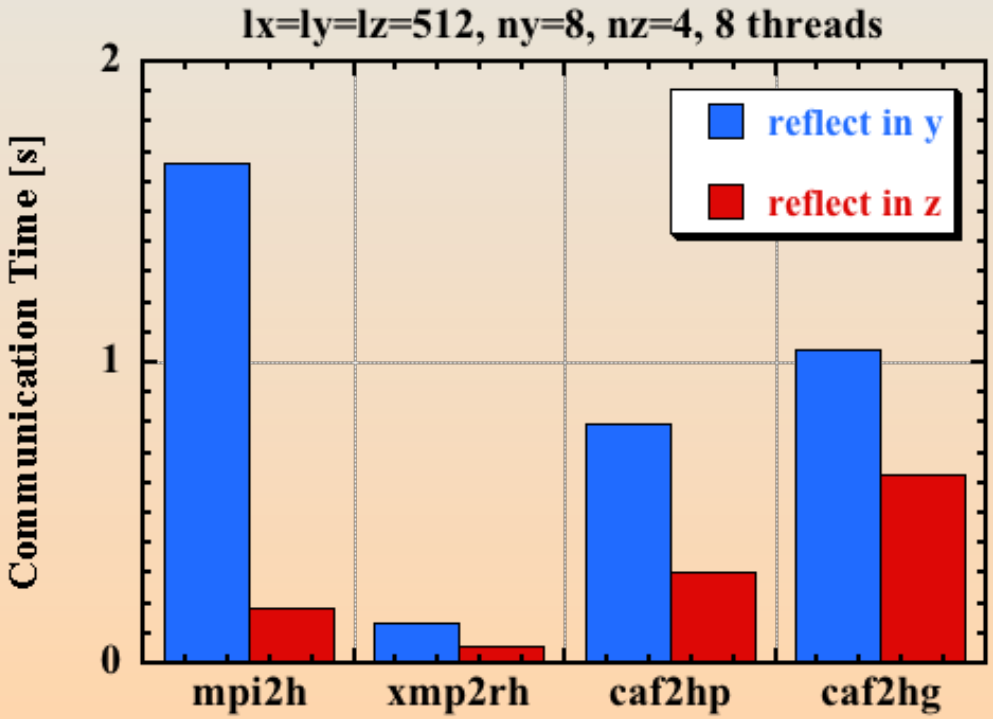




「京」(32ノード)での実行結果

- XMP/Fのcoarray通信は,Fujitsu RDMAで実装されている.
- すべての版で通信性能は同等であるが,coarray版は計算性能が半分程度に低下している.
  - coarrayは配列pointerで実装されたため,ループ実行のコンパイラ最適化が阻害されたことが原因であった.





一見便利な機能だが…

- 配列pointer,形状引き継ぎ配列
  - コンパイル時に配列形状等が確定しないので,最適化が難しい.
    - 起こりうる全ての場合について,計算結果は正しくなければならない.
  - 高性能なコンピュータほど性能差は大きくなる.
- 部分配列の実引数(例えば,a(11:20,11:20,11:20))
  - ナイーブな実装では,コンパイラが一時領域にコピーする.
    - 無駄なコピーの発生／メモリ利用制限値の超過
  - 賢い実装では配列をディスクリプタで渡したり引数を増やしたり
    - 連続アクセスにならない／互換性問題
- 割付け配列の自動再割付け
  - 配列サイズのチェックが必要になり,大域的な最適化が難しい.

サブセット／ガイドラインの必要性

- コンパイラ屋さんに頑張れと言うのは簡単だけど…
  - 状況を論理的に考えると難しいのは,よくわかる.
- 大規模な数値計算が高速に実行できることを優先すべきであり,それを阻害する機能は削ってもいいのでは?
  - そのサブセットを推奨することはできる.

- 多次元配列／多重ループにおける参照順序
  - Fortranでの連続アクセスになっていない.(C教育のせいか…)
- implicit noneが推奨されているが…
  - DOループのインデックス変数をサブルーチン毎に宣言するのが邪魔くさいので,moduleで定義してuseしていた.