OpenMP 発展

岩下武史
（京都大学学術情報メディアセンター）

OpenMP発展の前に・・・・

- 問い： 計算の速度は何できまる？
  - CPU（コア）のクロック？
  - キャッシュの容量？
  - メモリの性能？

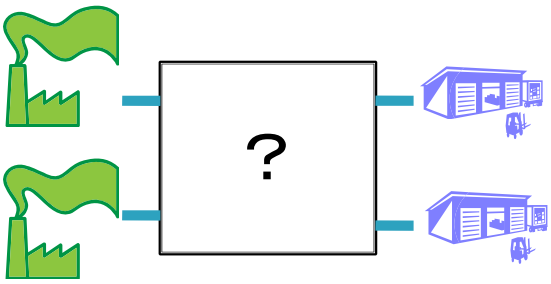
正解： 計算の種類に依存する（一概にいえない）

大規模数値シミュレーション、特に離散化シミュレーションではメモリの性能（キャッシュの構成や性能も含む）に影響されやすい

メモリの性能：バンド幅とレイテンシ

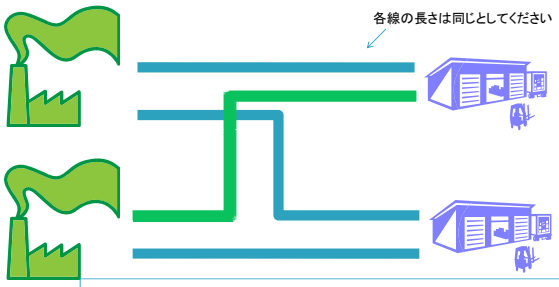
- **バンド幅**：単位時間あたりにどれだけのデータ量を転送できるか
- **レイテンシ**：データを要求してから最初のデータが届くまでの時間

共有メモリ型並列計算機
複数のプロセッサとメモリモジュール



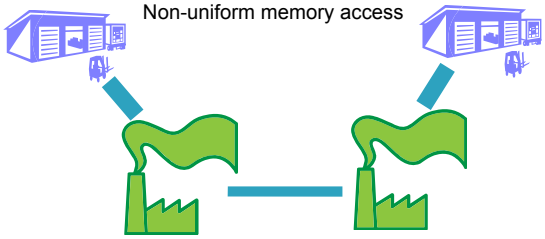
プロセッサとメモリの構成にはいろいろとある

共有メモリ型並列計算機
複数のプロセッサとメモリモジュール



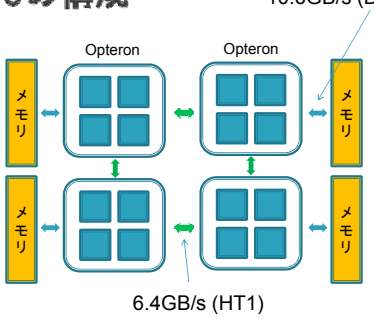
UMA(Uniform memory access)
CPUKからみてどのメモリモジュール上のデータも均等にアクセスできる

NUMAアーキテクチャ



- HX600(Thin)，SE M9000(FAT)のいずれでも採用
- CPUからみた各メモリモジュールに対するアクセスが不均一

HX600の構成



コアで使うデータはコアの近くのメモリに配置する必要がある

メモリインテンシブな計算(1)

例えば、こんな計算を考えてみる

- integer, parameter :: n=4*10**8
- !$OMP PARALLEL DO
- do i=1,n
- b(i)=a(i)*2.0
- enddo

ロード/ストア:2回、演算:1回
Byte per flop: 16

1コアの理論性能: 9.2Gflops
1ソケットのメモリバンド幅: 10.6GB/s

メモリインテンシブな計算(2)

スレッド数(コア数)	計算時間(msec)
1	2130
2	1311
4	927
16	931

HX600(Thin)上での計算結果

計算は単純で並列化可能であるが、
よい台数効果が得られていない

ファーストタッチ(1)

- スレッドのあるコアに近いメモリにそのスレッドが使用するデータを配置する
- 配列は宣言時において物理メモリ上には確保されない。
- 配列にデータを格納した時点で物理的なメモリ上の場所が確定する。
- **データを使うスレッドにより当該データを初期化する**

→ **ファーストタッチ**

ファーストタッチ(2)

- integer, parameter :: n=4*10**8
- !$OMP PARALLEL DO
- do i=1,n
- a(i)=1d0
- b(i)=0d0
- enddo
- !$OMP END PARALLEL DO

この部分がファーストタッチ

- !$OMP PARALLEL DO
- do i=1,n
- b(i)=a(i)*2.0
- enddo

ファーストタッチの効果

スレッド数(コア数)	計算時間(msec) (ファーストタッチなし)	計算時間(msec) (ファーストタッチあり)
1	2130	2058
2	1311	1062
4	927	555
16	931	331

4スレッドまでほぼ理想的な速度向上

疎行列・ベクトル積ベンチマーク

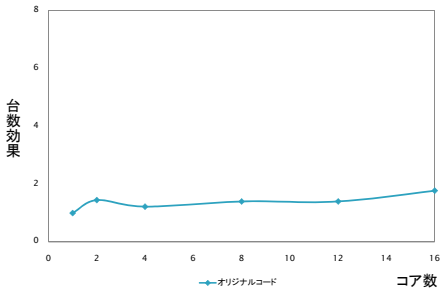
- 有限要素解析等の離散化解法において頻繁に用いられる
- 行列データは一回の行列・ベクトル積演算で一度しか使われない
  - データサイズが大きい場合、キャッシュの有効利用が期待できない
  - メモリスループットにより計算速度が決まる
    - 行列中の非ゼロ要素位置によってはそれ以下の性能となる場合があるが、今回は考えない。
- HX600上でメモリアクセスチューニングの効果を検証

通常のプログラム(オリジナルコード)

```
!$OMP DO PRIVATE(i,j)
 do i=1,n
    do j=ihead(i),ihead(i+1)-1
       bvec(i)=bvec(i)+a(j)*x(icol(j))
    enddo
 enddo
```

行列ベクトル積の並列化は一般的に簡単とされているが・・・

1コア時の経過時間を基準とした台数効果



並列処理による効果が全く得られていない

メモリアクセスチューニング

- ファーストタッチ
  - プログラムにおいて配列を宣言した時点では、物理的なメモリ上に領域は確保されない
  - 各スレッドが計算において扱う配列の領域を事前にタッチ(ダミーの値を格納)する
- 親和的スレッド割り付け
  - スレッドをソケットに分散して割り付け
  - HX600ではジョブスクリプト中の環境変数で指定

ファーストタッチ

```
!$OMP DO
do i=1,n
   bvec(i)=0d0
   ihead(i+1)=0d0
   x(i)=0d0
enddo
!$OMP DO
do i=1,nzero
    a(i)=0d0
    icol(i)=0d0
 enddo
```

疎行列・ベクトル積のデータを読み込む配列に対して、左のプログラムを実行する

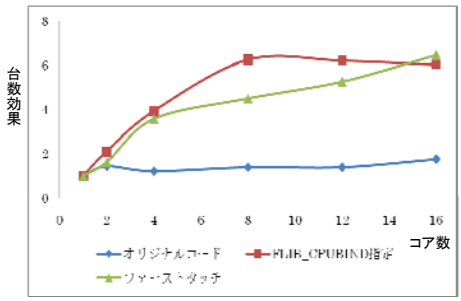
↓

配列要素のうち、各スレッドが初期化した部分は当該のスレッドのあるコアの近くのメモリに配置される

親和的スレッド割り付け

- 京大のシステムでは、通常のジョブスクリプトではどのコアにどのスレッドIDをもつスレッドを割り付けるかの規定ができない
  - 複数のコアを使った場合、どのコアが使われるかわからない
  - numactl(OS)のバージョンが筑波大、東大のシステムと比べて古い
- ただし、スレッドをソケット単位で割り付けることはできる。ジョブスクリプト内で以下を指定する。

```
FLIB_CPUBIND=chip_unpack
export FLIB_CPUBIND
```



1コア時の経過時間を基準とした台数効果

チューニング効果のまとめ

- 通常の疎行列ベクトル積のコードでは、スレッド並列化をしてもほとんど高速化されない
- ファーストタッチにより改善される
- 親和的スレッド割り付けにより、スレッド数が4，8の場合に改善
  - スレッドをソケット毎に割り当てるため、4本のパスを有効に活用可能
- 本ベンチマークの例においても、ソケットあたり1スレッド（1コア使用）とした場合にはメモリのバンド幅を使い切れていないので、ソケットあたり少なくとも2コアを使用した方がよい