姫野ベンチマークの調査

中尾 昌広

1　今月行ったこと

- 姫野ベンチマークの調査

2　調査結果

2.1　姫野ベンチマークの特性

姫野ベンチマークは 3 次元配列[1]を用いてヤコビの反復法の計算を行っている．複数のノードで並列計算を行う場合はその 3 次元配列をどのように分割するかをユーザーが決定できる．

2.2　計算回数の測り方

ベンチマークとは計算処理の速度を測定したものであり，色々なハードウェアとソフトウェアのシステム設定の比較に用いられる．単位にはよく Mflops が用いらる．この単位は浮動小数点演算を一秒間に何百万回計算できるかを示している．

姫野ベンチマークではプログラム中のヤコビの反復法を一秒間にどれだけ計算できるかを算出して性能評価を行っている．ここでヤコビの反復法の関数内で行われている計算回数を A とおき，3 次元配列のそれぞれの要素数を x，y，z とおくと，

$$A = (x-2)(y-2)(z-2) \times 34 \qquad (1)$$

で表すことができる[2]．次に A を用いてベンチマークの値を出す．ベンチマークの値を B とおくと次式で表すことができる．

$$B = \frac{A}{\text{ヤコビの反復法の実行時間}} 1.e-6 \qquad (2)$$

最後にベンチマークの精度を上げるために一度行った実行時間で 60 を割り，その値の回数だけ繰り返し計算させ，最後にその平均の値を算出する[3]．

2.3　分割の方法

複数のノードに分けた場合，分割したそれぞれの配列が各ノードに送られ，計算過程である値をそれぞれ別ノードに通信[4]を行うことによって，プログラムの並列化を行っている．またその場合，実行時間の算出方法はすべてのノードの実行時間から一番大きな値を調べ，それを実行時間としてベンチマークの値を算出している．

[1]それぞれの配列の要素数はユーザが定義できる
[2]ここで 34 とはこの関数内での四則演算の数である
[3]つまり約 1 分間で姫野ベンチの平均値が算出できる
[4]ちなみにノンブロッキング通信である

3　Gregor と Cambria との比較

3.1　分割数について

姫野ベンチでは 3 次元配列のそれぞれをある値に分割することができる．3 次元配列の要素の分割数を x，y，z(≥ 1) とおくと通信回数は以下の式のようになる．

$$\acute{x}, \acute{y}, \acute{z} = \begin{cases} 1 & (x, y, z \geq 2 \text{ のとき}) \\ 0 & (x, y, z = 1 \text{ のとき}) \end{cases} \qquad (3)$$

$$\text{通信回数} = (4\acute{x} + 4\acute{y} + 4\acute{z}) + 2xyz \qquad (4)$$

3.2　速度の比較

式 (4) よりノードが増えるほど通信負荷が増えることが予測できる．そこで Ethernet を用いている Cambria と Myrinet を用いている Gregor とでノードの数が増えれば姫野ベンチの値がどう変化するかを観察する．結果は Fig. 1 の通り．

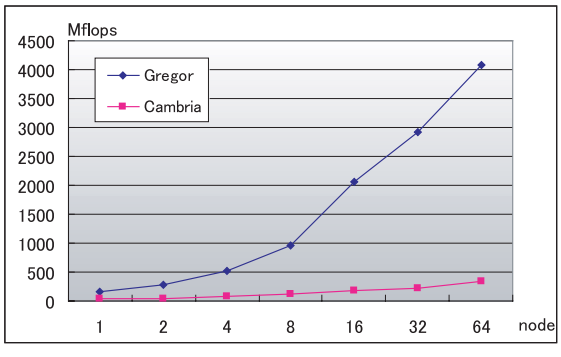

Fig. 1　姫野ベンチマークの結果

3.3　考察

Fig. 1 を見ればわかる通り，Gregor に比べて Cambria はノードの数が増えれば速度の上昇もなだらかになっていく．これより姫野ベンチは通信速度がとても重要であることがわかる[5]．

4　今後の課題

- Linpack の測定

5　参考文献

姫野ベンチマーク `http://w3cic.riken.go.jp/HPC/HimenoBMT/result.htm`

[5]一般に CPU の処理時間に比べて，通信時間の方が圧倒的に多い