グリッドコンピューティング (並列分散計算)

分散システムは、大規模な問題を解く際に優れた利点を発揮します。計算負荷を共有することで、単一のコンピュータでは処理できない大規模な問題を解くことができます。また、単一コンピュータで問題を解くよりも大幅に時間を短縮できます。Maple 16 では Grid Computing Toolbox のプラットフォームサポートが拡張され、世界最大級のクラスタ上で並列分散されたプログラムを容易に作成し、テストできます。

MPICH2

MPICH2 は、Argonne National Laboratory (http://www.mcs.anl.gov/research/projects/mpich2/) により配布されている、メッセージ パッシング インターフェイス (MPI) 規格の高性能実装です。MPICH2 の目的は次のとおりです。

1. コモディティクラスタ (デスクトップシステム、共有メモリーシステム、マルチコアアーキテクチャ)、高速ネットワーク (10 Gigabit Ethernet、InfiniBand、Myrinet、Quadrics)、専用ハイエンドコンピューティングシステム (Blue Gene、Cray、SiCortex) を含むさまざまな計算プラットフォームおよび通信プラットフォームを効率的にサポートする MPI 実装を提供します。
2. その他のデバイス実装に対しては、拡張が容易なモジュール形式のフレームワークを介して MPI で最先端の研究を可能にします。

Maple 16 の Grid Computing Toolbox には、マルチマシン計算やクラスタ並列計算を行うために MPICH2 とのインターフェイスを取るマルチプロセス計算を簡単に設定できる機能があります。

例： モンテカルト積分

以下の公式に従って定積分の近似を計算するために、x のランダム値を使用できます。

$$Area = \int_a^b f(x)\,dx = \lim_{N\to\infty} \frac{1}{N}\sum_{i=1}^{N} f(r)\,(b-a)$$

この手順は、上記の公式 (r は f に対するランダム入力) を使用して 1 変数積分を効率的に計算します。

```
#simple monte-carlo integrator

approxint := proc( expr, r::name=numeric..numeric, { numsamples::integer := 1000 } )
    local f, randvals;

    f := `if`( type(expr,procedure), expr, unapply(expr,lhs(r)) );

    randomize();
    randvals := LinearAlgebra:-RandomVector(numsamples,generator=evalf(rhs(r)),datatype
    (add( f(randvals[i]), i=1..numsamples )/numsamples) * (rhs(rhs(r)) - lhs(rhs(r)));
end proc:
```

1000 データ点を使用した場合のサンプル実行で、これがどのように機能するかを示します。

$$approxint\left(x^2, x = 1..3 \right);$$

$$8.63278051029565 \qquad (1.1.1)$$

Maple での計算結果は正確で、上記の近似が概算であることがわかります。ただし、アプリケーションによっては、これで十分です。

$$\int_1^3 x^2 \, dx$$

$$\frac{26}{3} \quad \textbf{(1.1.2)}$$

$$\xrightarrow{\text{at 10 digits}}$$

$$8.666666667 \quad \textbf{(1.1.3)}$$

並列実装により、以下のコードが追加されます。このコードは、問題を利用可能なすべてのノード間で分割し、部分解をノード 0 に戻します。ここではヘッドノード 0 を使用して結果を累算していますが、このノードでは計算はまったく実行されていないことに注意してください。

```
parallelApproxint := proc( expr, lim::name=numeric..numeric, { numSamples::integer := 1
 )
    uses Grid;
    local me, numNodes, r, n;

    me := MyNode();
    numNodes := NumNodes();
    n := trunc(numSamples/numNodes);
    r := approxint( expr, lim, numsamples = n );
    printf("Node %d computed result %a using %d samples\n", me, r, n );

    if me = 0 then
        r := (r + add( Receive(), i=1..numNodes-1 )) / numNodes;
    else
        Send(0,r):
    end if;
end proc:
```

N サンプルを使用して範囲 lim を積分します。ご使用のクラスタで利用可能なノードをいくつでも使用できます。

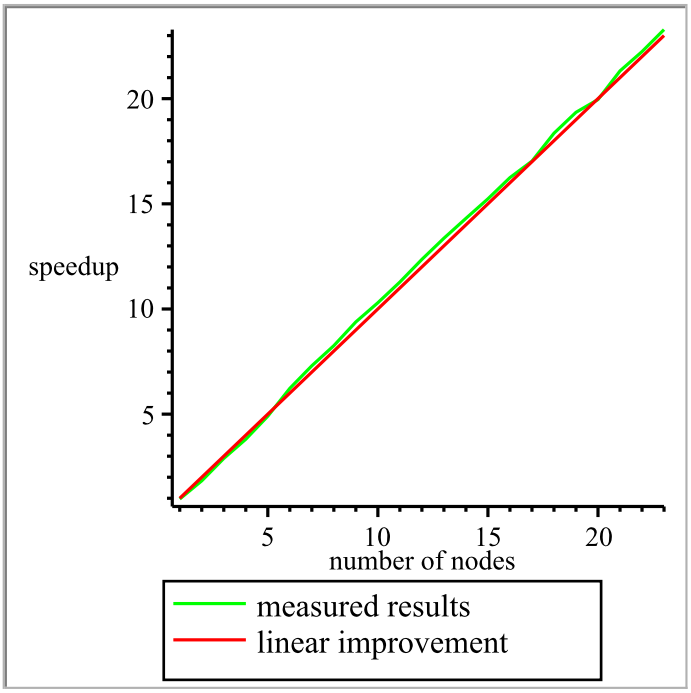
Grid:-*Setup*("hpc");

```
Grid:-Launch(parallelApproxint, x^2, x = 1 ..3, numSamples = 10^7, imports = ['approxint'],
    numnodes = 16);
Node 12 computed result 8.66806767157001 using 625000 samples
Node 10 computed result 8.66806767157001 using 625000 samples
Node 4 computed result 8.66806767157001 using 625000 samples
Node 2 computed result 8.66806767157001 using 625000 samples
Node 6 computed result 8.66806767157001 using 625000 samples
Node 5 computed result 8.66806767157001 using 625000 samples
Node 0 computed result 8.66806767157001 using 625000 samples
Node 13 computed result 8.66806767157001 using 625000 samples
Node 11 computed result 8.66806767157001 using 625000 samples
Node 1 computed result 8.66806767157001 using 625000 samples
Node 9 computed result 8.66806767157001 using 625000 samples
Node 8 computed result 8.66806767157001 using 625000 samples
Node 3 computed result 8.66806767157001 using 625000 samples
Node 15 computed result 8.66806767157001 using 625000 samples
Node 7 computed result 8.66806767157001 using 625000 samples
Node 14 computed result 8.66806767157001 using 625000 samples
                                  8.66806767157001                                  (1.1.4)
```

実行時間は次のように要約されます。計算は、6 つのクアッドコア AMD Opteron 2378/2.4GHz プロセッサと CPU ペアあたり 8GM のメモリーを持ち、Windows HPC Server 2008 を実行する 3 ブレードクラスタ上で実行されました。

計算ノード数	解を得るまでの実時間	性能
1 (逐次コードを使用)	356.25 s	スピードアップは、以下の尺度で判断できます。 $\frac{T_1}{T_p}$ ここで、T_1 は逐次アルゴリズムの実行時間、T_p は p プロセスを使用した並列アルゴリズムの実行時間です。
1 (Grid を使用)	369.18	
2	194.63	
3	123.12	
4	93.57	
5	72.67	
6	57.22	

7	48.80
8	43.15
9	37.96
10	34.60
11	31.58
12	28.83
13	26.66
14	24.90
15	23.38
16	21.93
17	20.92
18	19.41
19	18.41
20	17.85
21	16.71
22	16.02
23	15.30 *s*



> MapleGrid を使用しない場合の Maple の計算時間は、表の最初の数字 (6 分間) です。残りは、コアの数をさまざまに変えて MapleGrid を使用した場合の時間です。グラフは、コアを追加すると線形に時間が短縮されることを示しています。同じ例で 23 のコアが使用された場合、計算には 15.3 秒しかかかりません。

詳細は、Grid パッケージのドキュメントを参照してください。