1T-5

PC グリッドの性能評価を行うための ベンチマークシステムの設計と実装

畑寛之† 西山裕之† 溝口文雄†
東京理科大学理工学部†

1 はじめに

近年，遊休状態にある PC を仮想的な計算資源として使用する PC グリッドが，低コストでグリッド構築が可能であるという点から研究機関や企業など，様々な現場で注目を受けている．PC グリッドを構築する際，そのグリッドの処理能力を測定するベンチマークは，実際の大規模計算問題を解く上での目安として必要である．

現在，グリッドに対するベンチマークソフトとしてデファクトスタンダードとなっているのが，線形代数の LU 分解並列処理にかかる時間からグリッドの性能評価を行う High-Performance Linpack Benchmark[1](以下，HPL)である．HPL は世界のスーパーコンピュータの処理性能のランキングを行うサイト Top500 でも採用されている．しかしながら，HPL はクラスタのようなホモジニアスな環境で使用することが前提であり，PC グリッドのようなヘテロジニアスな環境には適していない[2][3]．そこで本研究では，この問題を解決するために PC グリッドに適した性能評価を行うベンチマークソフトの設計と実装を行う．

2 設計

2.1 タスク分配

既存のベンチマークソフトは，あらかじめ用意した大規模計算問題を，ノード数だけ均等にタスク分配を行う手法をとっている．本システムでは大規模計算問題に対して，ノードの処理性能に応じたタスク分配を行う．各ノードのタスク分配量は，以下の三つのデータから，タスク分配率を計算し，それを基に決める．

- ノードの CPU 周波数
- ノードの平均 CPU 使用率
- 浮動小数点演算のループ計算の処理時間

CPU 周波数はノード性能の理論値であり，ループ計算処理は実測値である．また，CPU の周波数は PC グリッドのノードは，普段 PC として動作しているため，その使用頻度を計る目的がある．以上，三つのデータからタスク分配率を求めることで，汎用的なタスク分散量を算出することが出来る．

次に，タスク分配率を求める計算式を導く．ここでノード i に対して，CPU 周波数を $c_i[MHz]$，平均 CPU 使用率を $u_i[\%]$，ループ計算の処理時間を $t_i[ms]$ とし，ノードの合計を n[個]とする．

はじめに CPU 性能 P_i と演算性能 O_i を求める．

$$P_i = 1 - \frac{c_i(1-u_i/100)}{\sum_{k=1}^{n} c_k(1-u_k/100)} \qquad O_i = \frac{1-t_i \Big/ \sum_{k=1}^{n} t_k}{n-1}$$

そして P_i と O_i から，タスク分配率 D_i は式(1)となる．

$$D_i = \frac{\left(P_i \Big/ \sum_{k=1}^{n} P_k\right) + \left(O_i \Big/ \sum_{k=1}^{n} O_k\right)}{(n-1)} \qquad (1)$$

2.2 大規模計算問題

本システムでは大規模計算問題として積分法による円周率計算を用いる．

$$\pi = \int_0^1 \frac{4}{t^2+1} dt \qquad (2)$$

式(2)に対して，積分区間を任意の数に分割し，式(1)で求めたタスク分配率に沿って，各ノードに分配する．ノードは与えられた積分区間の計算を行う．各ノードが計算を終えた後，最も遅い処理時間をグリッド全体の処理時間とする．

2.3 性能評価

本システムの大規模計算問題は，実装に Java を用いるため(3.1 参照)，演算処理能力を示す単

Design and Implementation of benchmark system to evaluate performance in PC grid
Hiroyuki Hata†, Hiroyuki Nishiyama†, Humio Mizoguchi†
†Faculty of Science and Technology Tokyo University of Science

位である flops を用いた絶対的な評価値による評価を行うことが困難である．よって，本システムの均一なタスク分配による測定から得た結果と，最適化されたノード分配による測定を比較し，相対的な評価を行う．そして，本システムで最適化を行うことで，どのくらいの性能向上が見込めるかを評価の対象とする．

3 実装

3.1 システム構成

本システムはヘテロジニアスな環境に対応させるために，アーキテクチャに非依存かつマルチプラットフォームで実行が可能な Java を用い実装を行う．システムはマスタ部，ワーカ部，UI 部により構成されている．

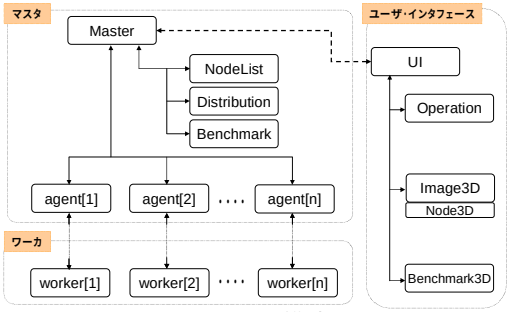

図 1.システム構成

マスタはノード情報の管理，タスク分配率の計算，ベンチマーク測定の実行および評価を行う．各ノードへの命令はエージェントを介し，ソケット通信により行う．ワーカには，ベンチマーク測定のための計算モジュールや，ノード情報取得用の専用モジュールが実装されている．UI は操作処理と視覚化処理に分かれている．特に視覚化では，システムの動作を視覚化する．視覚化の方法として，一定空間内に様々な情報を効果的に表示することが可能な三次元空間を使用し，実装には Java3D を用いる．

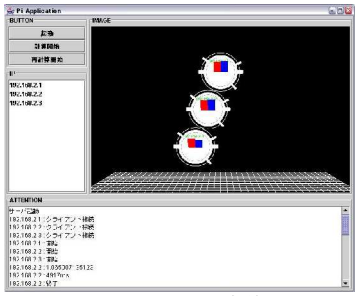

図 2.システムの概観

3.2 ノード情報取得

CPU のようなハードウェア情報は PureJava で取得することが出来ないため， NativeMethod を用い，JNI で呼び出す．そして，浮動小数点演算のループ計算として，モンテカルロ法による円周率計算を行う．また，CPU 周波数とループ計算処理はノードの接続の際に取得する．そして，平均 CPU 使用率は，接続した時点からベンチマーク測定を行う直前まで常時取得し，測定を行う際に，それまでの平均を取るものとする．

4 実験

本システムの評価実験を行った．実験環境として三台のノードにより構成するグリッドを使用した．三台についてのノード情報は以下の通りである．CPU 性能 P_i，演算性能 O_i，タスク分配率 D_i および大規模計算問題の結果は以下の通りである．

表 1.タスク分配率

	P_i	O_i	D_i
Node1	0.46	0.36	0.41
Node2	0.34	0.37	0.35
Node3	0.20	0.27	0.24

表 2.計算結果

	均等分配[ms]	最適化分配[ms]
Node1	1828	2234
Node2	4836	5078
Node3	9396	7862

表 2 から，均等分配に比べて約 20%の性能向上に成功した．

5 おわりに

本システムは，PC グリッドを構成するノードに最適なタスク分配を行うことで，環境に適した性能評価を行うことを可能にした．また，最適化を行う際に，平均 CPU 使用率を考慮することで，実際の PC グリッドに対応した最適化が行えるようになった．

参考文献

[1] High-Performance Linpack Benchmark for Distributed-Memory Computers http://www.netlib.org/benchmark/hpl/

[2] 笹生 健，松岡 聡，建部 修見，「ヘテロなクラスタ環境における並列 LINPACK アルゴリズム」，並列処理シンポジウム JSPP2002 論文集，pp.71-78，(2002)

[3] 岸本芳典，市川周一，「不均一クラスタ上での並列 Linpack の性能に関する検討」JSPP2002 論文集，pp.177-178， (2002)