B-002

Java 並列処理フレームワークのマルチコア環境における性能評価
Performance Evalution of a Java Framework for Parallel Processing on Multi-Core Processor

古賀 雅伸 †
Masanobu Koga

赤峰 義明 †
Yoshiaki Akamine

1. はじめに

複雑な問題を高速に処理する技術の一つとして並列処理がある。並列処理はハードウェアの変化に伴い，さまざまな手法が研究開発されてきた。並列処理手法の代表的なものとして共有メモリ型のハードウェアで利用される OpenMP[1] や，分散メモリ型のハードウェアで利用される MPI[2]，RMI[3]，などがあげられる。数年前までは PC クラスタによる並列処理が注目されていたが，近年はマルチコア CPU による並列処理に注目が集まっている [4]。

ユーザは多くの並列処理手法から条件に合ったものを選択できる。しかし，並列処理を利用する場合は，手法の特徴やプログラミング方法を理解する必要があり，並列処理の実装は容易ではない。手法を変更する場合は，プログラムを大幅に書き換えなければならない。

我々の研究グループでは，並列処理の実装の効率化を図るため，並列処理手法を実行時に選択可能な並列処理フレームワークを開発した [5]。本稿では，マルチコア環境におけるフレームワークの有効性について述べる。

2. 並列処理フレームワーク

本フレームワークでは，ユーザが GUI を通し並列処理手法，負荷分散手法を選択できる。GUI 画面を図 1 に，並列処理手法選択の概略を図 2 に示す。また，並列処理問題（ホットスポット）を実装することで，容易に並列処理を実行できる。フレームワークの開発は Java 言語で行っているため，実装には Java 言語の知識が必要となる。

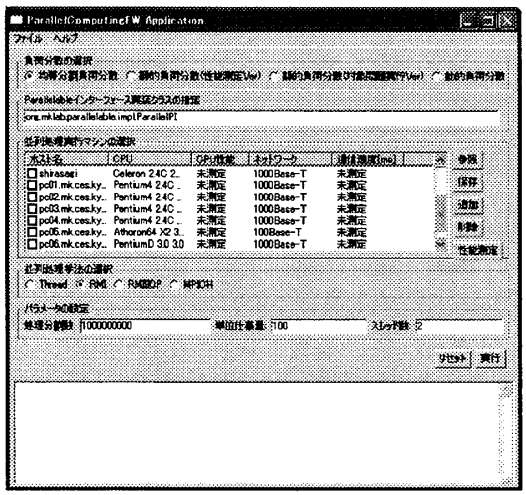

図 1: GUI 画面

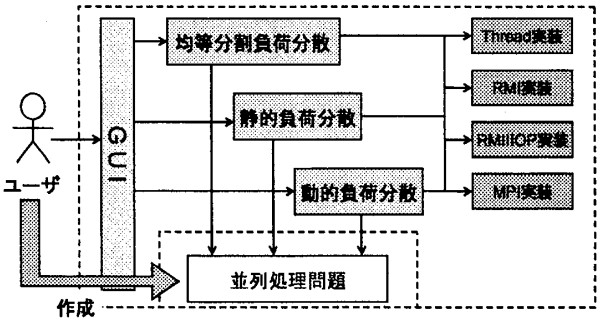

図 2: 並列処理手法選択の概略

2.1 フレームワークの機能

提供する並列処理手法は，スレッド，RMI，RMI-IIOP，MPI の 4 種類である。複数のマシンを利用した並列処理を行う場合は，マシン情報を保存した XML[6] ファイルを読み込むことで，GUI からマシンの選択が可能となる。マシン情報の追加や削除は GUI 上から行える。

提供する負荷分散手法は，均等分割負荷分散，静的負荷分散，動的負荷分散の 3 種類である。静的負荷分散の評価は，計算能力と通信能力を基に行う。計算能力は Scimark2.0[7] の CompositeScore，通信能力は ping よって得られた測定結果を利用する。

パラメーターとして GUI から処理分割数，単位仕事量，ノード数を与えることができる。

2.2 並列処理問題クラス

並列処理問題（ホットスポット）を表す並列処理問題クラスでは，図 3 に示すインターフェースを実装する必要がある。各メソッドでは以下の処理を行う。

- ·setSize : ノード数を設定する。
- ·receive : ノードの計算結果を返す。
- ·reduce : 計算結果を統合する。
- ·task : ノードの計算処理を実行する。
- ·getResult : 並列処理全体の計算結果を返す。

Parallelable インターフェースを実装することでフレームワークが提供する並列処理手法の選択が可能となる。静的負荷分，動的負荷分散を利用するには，それぞれ Parallelable を継承した StaticLoadSharable インターフェース，DynamicLoadSharable インターフェースを実装する。

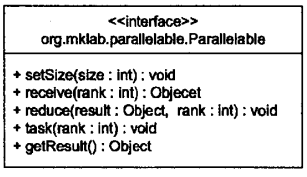

図 3: Parallelable インターフェース

†九州工業大学院, KIT

3. マルチコア環境における性能評価

本研究では，前節で述べたフレームワークの性能評価をマルチコア環境で行った。

3.1 実験環境

クアッドコア CPU であるインテル Xeon5310 を 2 つ搭載した PC を用い 8 コアの環境を構築した。仕様を表 1 に示す。

表 1: 実験環境の仕様

構成要素	仕様
OS	Windows XP
CPU	Intel(R) Xeon 1.6GHz E5310 × 2
Mother Board	Intel(R) 5000X チップセット
Memory	DDR2 667 512MB FBIDIMM × 4

3.2 性能測定実験

π 計算問題，800×800 行列の行列乗算問題を用いて性能測定を行った。実験条件を表 2 に示す。

表 2: 実験条件

対象問題	π 計算	行列乗算
計算量	処理分割数 10 億	20 回
並列処理手法	Thread	Thread
負荷分散手法	均等分割	均等分割
スレッド数	最大 8 つ	最大 8 つ

ノード数 8 の場合のタスクマネージャー画面を図 4 に示す。図 4 をみると，ほぼ同時に各 CPU が動き始め，8 つ全てのコアが使用され CPU 使用率が 100 %であることが確認できる。さらに，各測定結果を図 5 に，性能向上率を図 6 に示す。

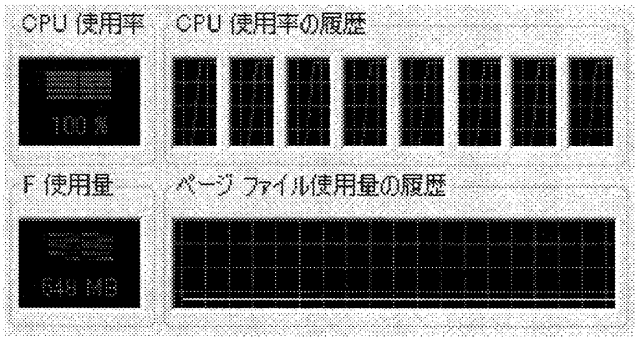

図 4: タスクマネージャー画面

測定結果から，マルチコア環境で十分に並列処理の効果が現れることが確認できた。さらに，アムダールの法則 [8] より並列化された割合の近似値を次式から求める。

$$\text{性能向上率}=1/(F+(1-F)/N)$$

F:並列化できない割合，N:プロセッサ数

π 計算の F を求めると 0.01，行列乗算の F を求めると 0.035 となる。これより並列化された割合は，π 計算の場合は約 99 %，行列乗算では約 96.5 %となった。

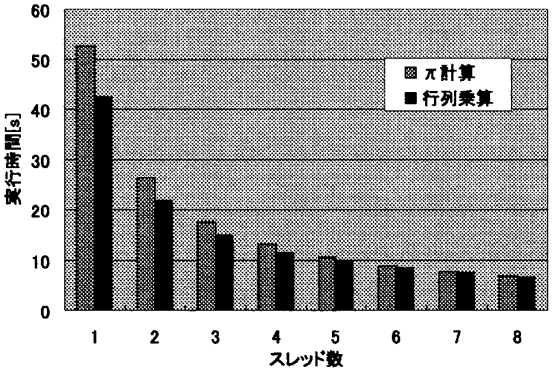

図 5: スレッド数と処理時間

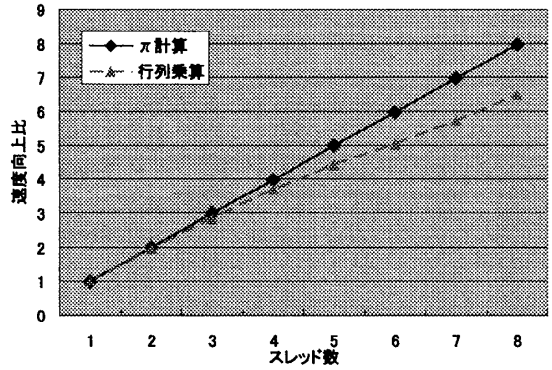

図 6: マルチコア CPU による性能向上率

4. まとめ

本稿では，並列処理手法を選択可能な並列処理フレームワークのマルチコア環境での性能評価を行った。性能評価結果では，8 コアを利用した場合 π 計算では約 7.97 倍，行列乗算では約 6.5 倍の速度向上比が得られ，フレームワークの有効性が確認された。

今回は，ホモなマルチコア環境で均等負荷分散を用いた性能評価を行った。今後のは，ヘテロなマルチコア環境における動的負荷分散，静的負荷分散の有効性を検証したい。

参考文献

[1] 牛島 省, OpenMP による並列プログラミングと数値計算法, 2006.
[2] W.D.Gropp, E.Lusk, User's Guide for mpich, a Portable Implementation of MPI, 1996.
[3] Remote Method Invocation Home, http://java.sun.com/javase/technologies/core/basic/rmi/,
[4] 情報処理 2006, 株式会社 精機通信社, 47 巻 1 号
[5] 稲沢 雄気, 計算機クラスタにおける負荷分散機能を有する並列処理フレームワークの開発, 2004 年度修士論文.
[6] O'RELLY XML.com, http://www.xml.com/
[7] Roldan Pozo and Bruce Miller, Scimark2.0, http://math.nist.gov/scimark2/
[8] Reevaluating Amdahl's Law and Gustafson's Law, http://joda.cis.temple.edu/ shi/docs/amdahl/amdahl.html