講演番号: 35

Erlang によるスケーラブルなマルチコア対応分散プログラミング
Scalable distributed programming on multi-core using Erlang

清澤　豪　　小林　洋
Suguru Kiyosawa　Hiromi Kobayashi

東海大学工学研究科情報理工学専攻
Tokai University, Graduate School of Engineering

1. はじめに

Erlang は，マルチコア CPU 搭載の PC や多数の PC ネットワーク上で処理の並行化を言語レベルで可能にする言語であり，通信分野などでは普及しているが，他の分野ではアプリケーションの開発の必要性が言われている．マルチコア PC で Erlang の特徴を生かすには，コア数が変化しても，プログラムを書き変えることなく対応できるというスケーラビリティが重要である．本研究では，ビジネス系システムでは頻繁に出てくる処理であるソーティング（sorting）を取り上げ，Erlang での処理の分散最適化に適していると考えられるバイトニックソート（Bitonic sort）を用い，コア数にフレキシブルに対応できるプログラムを開発した．また，最近では OS 自体がマルチコア CPU に対応しており処理の最適分散化が行われていると言われているが，言語記述による分散化と OS による分散化とは意味が違うはずなので，その違いについても検証を行った．

2. 実装の工夫

バイトニックソート（Bitonic sort）は，マルチプロセッサによる並列処理用のソーティングアルゴリズムとして開発された。このアルゴリズムは，昇順と降順からなる双調列（bitonic）の性質を利用したもので，シメトリックな構造を持つために，コア数が増加した際にも，負荷の自動均等化が行いやすいと予測される．Erlang[1]は，単一代入しか可能でない関数型言語であるため，C や Java とはプログラミングスタイルが異なる．for 文や while 文のような繰り返し表現がないので関数を使って表現しなければならない．並行処理を実現するためには，プロセス間のメッセージの送受信を，spawn()関数を使って実装することになる．開発にあたっては，データ数の変更をプログラム上で容易に変更可能にするというスケーラビリティのために，プログラミング上の工夫が必要であり，その際，配列が存在しないためリストを用いた．そして CPU の各コアに処理を分散化して割り当てるのが容易なように，本処理でのペア探索部分を二分割した．また，以上の並列処理用プログラム（以下，cbito）の他，性能比較のため直列処理用プログラム（以下，bito）も作成した．

3. 評価実験

評価実験には，以下のコア数 2 (duo)と 4 (quad)の PC を用いた．OS は Windows7 である．

・HP dx7400:　Inter(R)Core2 Duo CPU E4400 2.00GHz
・HPv7780jp/CT:Intel(R)Core2QuadCPU Q9650 3.00GHz

両者の性能比を表す MIPS 比は integer MIPS(Dhrystone)で，4258/6283=0.68 であり，両者でアプリケーションの

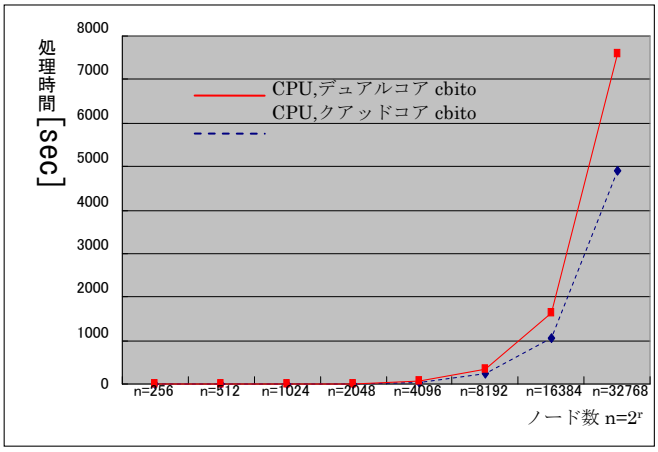

図 1 データ数 $n=2^r$[個]と処理時間[sec] (r=8~15)10 回の平均

実行時にこの値に近いほどマルチコアの性能を生かしているということができる．評価実験では，bito と cbito について，データ数（$n=2^r$[個]）を変化させながら処理時間の計測を行った．またその時の CPU 負荷のバランスも観測した．図 1 は，cbito の duo と quad でのデータ数を変化させた場合での処理時間の比較である．処理時間比は，データ数 $n=2^{15}=32768$ 個で，

(quad での処理時間)/(duo での処理時間)=0.65

であり，上記の MIPS 比との比較から，cbito ではマルチコアの性能を引き出すためのアプリケーションの並行化を行うことが出きたと言える．次に，Erlang によるマルチコア対応と，OS による対応を比較するために，cbito と bito（並行処理機能未使用）の実行時の比較を行ったところ，処理時間比は(cbito での処理時間)/(bito での処理時間)=0.75 であった．また，CPU 負荷の分散状況を観測したところ，cbito 実行時には負荷が均等に分散していたが，これに比べ bito では負荷の均等化が不十分であることが観測された．

4. おわりに

Erlang でのマルチコア対応を生かすためのアプリケーションとして，バイトニックソートの実装を行った．これにより，コア数が増えても並行化に柔軟に対応できるプログラムを開発することができた．今後の課題としては，他のアプリケーションの実装や，プログラムの洗練化が考えられる．

参考文献

[1] ジョー・アームストロング，榊原訳：プログラミング Erlang，オーム社(2008).