マルチコア向け統合開発環境における デバッグ効率向上のための視覚化機能の開発

鈴木　邦彦†　　山本　修一郎†　　前島　英雄†

東京工業大学　大学院総合理工学研究科　物理情報システム専攻†

1. まえがき

近年、CPU に対し、消費電力削減と性能向上の両立や、分散処理に対するリアルタイム処理性能の向上・既存環境との互換性による開発コストの低減といったことが求められ、その結果 CPU のマルチコア化が行われている。しかし、マルチコアでのアプリケーション開発では、複数のコアの同時動作や CPU コア間の同期、メモリのコヒーレンス、資源に対するアクセスコントロールといったことを考慮する必要があり、シングルコアでの開発に比べ、難易度が高くなっている。そのため、マルチコアでの開発の難易度を下げるために、統合開発環境の改善が求められている。

2. 機能の検討

本研究ではマルチコアでの開発におけるデバッグ効率の改善を目標とした。まず、デバッグ対象のプログラムについて考察を行った。マルチコアで動作するプログラムは複数の CPU コアが相互に連携し処理を行う。そのため、これらのプログラムのデバッグでは、ある CPU コアについてデバッグを行う際にも他の CPU コアの状態を知っておく必要がある。ところが、従来の統合開発環境では 1 つずつ CPU の動作状態を逐一デバッグ者が調査しなければならず、効率的なソフトウェア開発の障害となっていた。そこで、本研究では CPU コアの状態を一覧できる視覚化機能を開発することにした。

上記の CPU コア間の連携の例として、分散処理を図 1 に示す。分散処理では一般に、前処理で行う計算を複数に分割し、それぞれ別の CPU コアに割り当てて分散処理させ、最後に後処理で計算結果を統合させる。分散処理では、分散処理の前後で CPU コア間の同期を取る必要がある。

他の例として、ミューテックスを図 2 に示す。

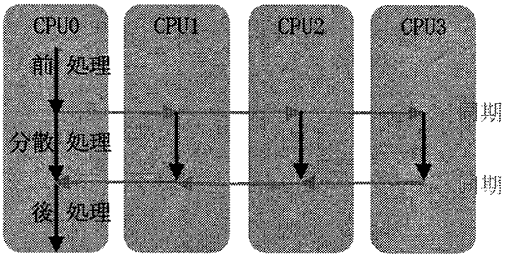

図 1　分散処理

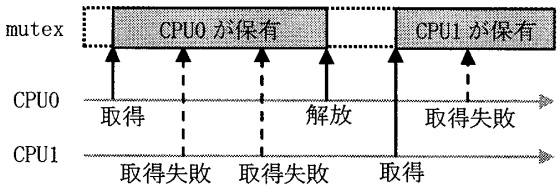

図 2 ミューテックス

ミューテックスは同時に 1 つしか取得できないオブジェクトで、複数の CPU コアからのアクセスが禁止されている資源へのアクセスコントロールに用いられる。本研究では、上で述べた 2 種類の CPU コア間の同期状態の視覚化を行うこととした。

さらに、上記の分散処理やミューテックス使用の結果として、あるメモリ空間に複数の CPU コアが書き込むことがある。従来このような処理のデバッグを行う際は、どの CPU コアがどこに書き込んだかを知るためには、CPU コアごとの処理を確認する必要がある。そこで、メモリ表示機能を拡張し、従来のメモリ変更点の視覚化に加えて、どの CPU コアが書き込んだかの視覚化も行うことにした。

3. CPU 動作状態の検出

次に、CPU 動作状態の検出法について論じる。一般に CPU コア間の同期処理やミューテックス処理は、複数の CPU 命令を組み合わせて実現されている。そのため、CPU コア状態の取得には各 CPU の命令実行を把握する必要がある。この各 CPU の命令実行の把握には、次節で述べるハードウェアによるトレースを用いた。その命令を解析し、同期処理およびミューテックス処理を検出する。

まず、一般的なミューテックス処理の例を擬似コードで以下に示す。

Development of Visualization Features on IDE for a Multi-core Processor to Improve Debugging Efficiency

†Kunihiko Suzuki, Shuuichirou Yamamoto, Hideo Maejima

Department of Information Processing, Interdisciplinary Graduate School of Science and Engineering, Tokyo Institute of Technology

```
while (!IsTestAndSetSuccessful()) {}
```

TestAndSet 命令を用いて、ミューテックスの取得を試み、成功するまでループし続けるようになっている。このため、上述のミューテックス処理は次の 2 点で検出できる。

- ループ中
- そのループ内に TestAndSet 命令が存在

次に、一般的な同期処理の例を擬似コードで以下に示す。

```
PrepareSyncing();
while (!AreAllTargetsSyncing()) {}
```

最初に同期待ちの準備を行い、その後対象全てが同期待ちになるまでループし続けるようになっている。また、このループ内の処理では書き込みは行わない。このため、上述の同期処理の検出は次の 2 点で検出できる。

- ループ中
- そのループ内にメモリ書き込み処理がない

しかし、この 2 点だけでは上記のミューテックス処理を含め同期待ちでないループも検出してしまうため、この 2 点に加えて、デバッグ者によるサブルーチン名指定を可能にし、誤検出を減らすことにした。

また、メモリ変更の視覚化のために、CPU によるメモリ変更を検出する必要があるが、これにも次節で述べるハードウェアによるトレースを用いた。

4. 視覚化機能の実装

本研究では、商用 32 ビット CPU コアを 8 個搭載した開発ボードおよび Linux 上のその統合開発環境 Eclipse において視覚化機能を開発した。

各 CPU の命令実行の把握とメモリ変更の検出には、開発ボードに実装されているトレース機能を用いた。トレース機能はトレース対象のイベントが発生すると CPU からハードウェアに内蔵されているトレースバッファに書き込みが行われる。イベントには分岐やメモリアクセスなどが用意されている。本研究では、統合開発環境上で、このバッファからトレースログを読み出すトレースビュー機能と、その読み出したトレースログを用いて CPU コア間の同期の視覚化機能とメモリ変更の視覚化機能を開発した。

トレースビュー機能と CPU コア間の同期の視覚化機能、メモリ変更の視覚化機能は、Eclipse へのプラグインとして開発した。[1][2]

5. 動作例

例として CPU8 個で分散処理を行うプログラムをデバッグした。その際の CPU 状態の視覚化機能を図 3 に示す。このプログラムは、各 CPU コアが（ 0xFFFF0000 + [コア番号] * 0x1111 ）を連続したメモリアドレスに書き込み、全 CPU コアで同期を取った後、CPU0 が結果を合算する。図では、CPU2～7 が処理を終え同期待ち、CPU1 がミューテックスを得て処理中、CPU0 が処理前のミューテックス待ちであることが視覚化されている。

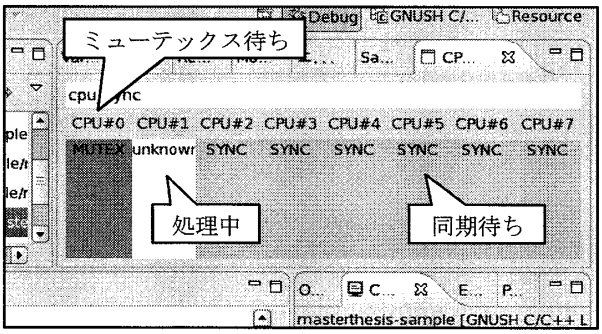

図 3 CPU 状態の視覚化

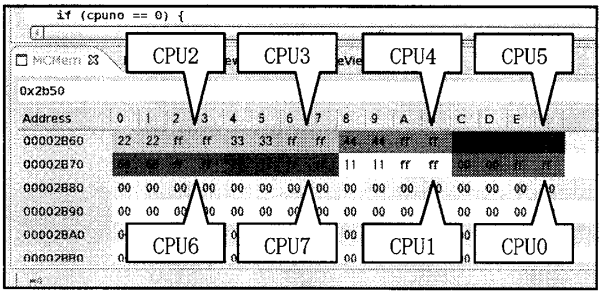

図 4 メモリ変更の視覚化

また、この際のメモリ変更の視覚化機能を図 4 に示す。各 CPU が連続したメモリアドレスにそれぞれの処理結果を書き込んでいることが視覚化されている。

6. まとめ

本研究では、デバッグ効率向上のために、マルチコア向け統合開発環境において、CPU 状態とメモリ変更の視覚化機能を開発した。

謝辞

本研究は NEDO 受託研究により実施された。

参考文献

[1] Eric Clayberg, Dan Rubel: Eclipse Plug-ins, Thrid Edition, Addison-Wesley, 2008

[2] 竹添直樹，志田隆弘，奥畑祐樹，里見知宏，野沢智也: Eclipse 3.4 プラグイン 開発徹底攻略，(株)毎日コミュニケーションズ，2009