A-029

SPEサーバを用いたプログラミング環境の構築

出宮 健彦† 高山 征大† 境 隆二†
†（株）東芝 デジタルメディアネットワーク社 コアテクノロジーセンター

1 はじめに

我々は Cell プロセッサをはじめとするマルチコア向けのソフトウェア技術開発を行ってきた．その中で *Molatomium* と呼んでいる並列実行環境を開発している．またパーソナルコンピュータ (PC) で並列実行環境向けのプログラムをビジュアルに開発が行えるプログラミング環境も用意した．しかし，PC 上のプログラミング環境からは直接 Cell プロセッサ向けプログラムの実行はできていなかった．そこで，ネットワーク上の Synergistic Processor Element (SPE) サーバを利用して，PC 上のビジュアルツールから Cell 向けのプログラムを実行するプログラミング環境の構築を行った．*Molatomium* でのプログラムは各プラットフォーム共通であるため，マルチプラットフォーム開発が効率的に行えるようになった．

2 マルチコア向け並列実行環境: Molatomium

ヘテロジニアスマルチコアも含めたプログラミングモデルとして，これまで我々は *Molatomium* と呼んでいる並列実行環境の開発を行ってきた [1]．並列プログラムの実装を容易にしつつ，コア数に比例した性能が得られることを目指している．本実行環境では特定のコアでスケジューラを実行せず，各コアがランタイムを持ち回りで実行していく．あるコアがランタイムを実行し始めると，C 言語風の *Mol* と呼んでいる並列記述言語から変換されたバイトコードを解釈し，データとタスクの依存関係グラフの生成，ランタイムのコンテキスト更新を行う．実行可能な C/C++ 言語で記述された *Atom* と呼んでいるタスクが見つかると，その *Atom* に処理を切り替え，ランタイムの実行権限を解放する．*Atom* 処理が完了するとランタイムの実行権限の取得を試みる．他のコアも同様にして処理を行っていく．この手順によって，各コアが自律的に処理を選択して実行できるようになり，並列動作の記述容易性と，高い並列性能を両立することが可能となる．*Mol* と *Atom* は各プラットフォーム共通であり，ランタイムがプラットフォームの差を吸収する．加えて，並列記述の容易性を高めるため，GUI を用いてビジュアルにプログラミングが出来る環境も用意した． この Multicore Editor で GUI パターンを使った *Mol* のプログラミング (図 1)，並列的な処理フローを検証するための展開図閲覧 (図 2)，ビルド，実行およびデバッグが可能である．

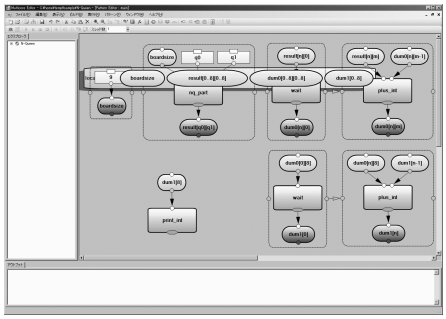

図 1: ビジュアルプログラミング環境 (GUI 編集モード)

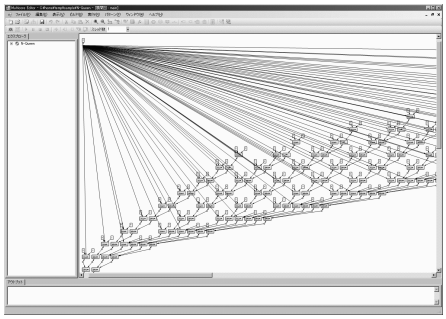

図 2: ビジュアルプログラミング環境 (並列処理フロー展開図)

3 SPE サーバを用いたプログラミング環境

前節で述べたビジュアルプログラミング環境は PC(OS: Microsoft(R) Windows(R) XP) 上で動作している．他のプロセッサでも *Mol* と *Atom* は共通であるが，ビジュアルプログラミング環境から直接他のプラッ

Implementation of Programming Environment using SPE Server
Takehiko DEMIYA†, Motohiro TAKAYAMA†, and Ryuji SAKAI†
†Core Technology Center, Digital Media Network Company, Toshiba Corporation
{takehiko.demiya, motohiro.takayama, ryuji.sakai}@toshiba.co.jp

トフォーム用のプログラムの動作確認が出来ていなかった．そこで今回，ビジュアルプログラミング環境から直接他のプラットフォーム用のプログラムの実行も出来るように，ファイルシステムを利用したリモートクロスプラットフォーム開発環境の構築を行った．リモートクロスプラットフォーム開発環境の構築には従来からデバッガのリモート・デバッグ機能が利用されてきた．本報告ではファイルシステムを利用して，ターゲットマシン上のスレッドといったデータとは異なるオブジェクトも含めてファイルとして見せようとするものである．ターゲットとした Cell には SPE を操作するための SPUFS と呼ばれるファイルシステムが Linux で実装されており，これをエミュレートしリモートでも Cell プロセッサ内部にある SPE と同様の使い勝手の提供を試みている．関連する研究として，ACCFS (ACCelerator File System)[2] などもあるが，Linux のカーネルコードの修正を必要とし，ユーザの自由度が低いと言える．そこで，一般ユーザがカーネルコードを修正することなく独自のファイルシステムを作成できる Filesystem in Userspace (FUSE)[3] を用いる．FUSE は，UNIX 系オペレーティングシステムのローダブル・カーネル・モジュールの一種であるため，ホスト PC には UNIX ライクな環境である Cygwin 環境を用いて導入する．本報告では FUSE の API を用いてリモートで Cell のプログラムを動作させるための独自のファイルシステム CBEFS を作成した．

ネットワーク上の SPE サーバを利用して，PC 上のビジュアルツールから Cell 向けのプログラムを実行する環境の構築を行った (図 3)．ターゲットマシンとなる SPE サーバと Multicore Editor が動作している開発ホストは相互にアクセス可能なネットワークに接続する．今回は同一の Local Area Network(LAN) に接続している．また，Cell 向けの *Molatomium* を適用した実行ファイルを生成するために，クロスコンパイル環境を導入した．Barcelona Supercomputing Center によって配布されている Cell の開発環境 [4] を Cygwin で再コンパイルし導入した．どのプラットフォーム向けのビルドを行うかの指定は，Multicore Editor で行う．Multicore Editor はプログラムのビルドに利用する Makefile の生成を行っているので，各プラットフォーム用の make ルールを環境変数の指定によって切り替えることで，プラットフォーム毎の実行ファイルのコンパイルを行う．

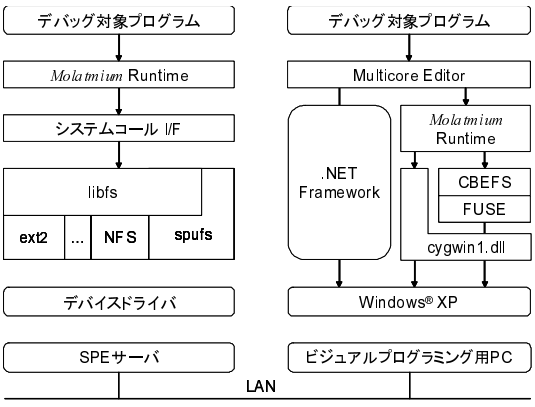

図 3: SPE サーバを用いたプログラミング環境

上記で構築した環境を用いることで，*Molatomium* の *Mol* と *Atom* が共通という特徴を活かしたマルチプラットフォーム開発ができる．また，CBEFS によって，リモートの SPE リソースをローカルマシンのリソースと同様に扱えるので分散環境の構築も容易になる．

4 おわりに

今回，ネットワーク上の SPE サーバを利用して，PC 上のビジュアルツールから Cell プロセッサ向けのプログラムを実行するプログラミング環境の構築を行った．構築した環境を用いることで，*Molatomium* の *Mol* と *Atom* が共通という特徴を活かしたマルチプラットフォーム開発が可能になる．また，プログラムの動作確認としてターゲットとは異なるプラットフォーム上で動作確認ができることによって，柔軟かつ効率的な開発が可能になると考えられる．今後は，Cell 以外のマルチコアプロセッサへの対応や，CBEFS の機能拡張を行う．

参考文献

[1] 境 隆二, 加藤 宣弘, 保科 聡, 島田 智文, “マルチコア向け並列プログラミングモデルの設計と実装,” 情報処理学会プログラミング研究会, 2008.

[2] Andreas Heinig, René Oertel, Jochen Strunk, Wolfgang Rehm, Heiko J. Schick, “Generalizing the SPUFS concept - a case study towards a common accelerator interface,” in *Proc. of the MRSC2008*, 2008.

[3] “FUSE: Filesystem in Userspace”, http://fuse.sourceforge.net/, 2009 年 6 月 29 日アクセス.

[4] “Barcelona Supercomputing Center (BSC) Linux on Cell”, http://www.bsc.es/projects/deepcomputing/linuxoncell/, 2009 年 6 月 29 日アクセス.