5T－5　並列分散プログラミング言語 Sushi のアノテーションの実現について

山中英樹
E-mail: yamanaka@flab.fujitsu.co.jp
〒261 千葉市美浜区中瀬 1-9-3, (株) 富士通研究所 情報社会科学研究所

1. はじめに

プログラミング言語 Sushi [UGA94,WAD94,YAM94] は，並列計算機，ネットワーク上の計算機群を使って，エンドユーザが簡易に効率的な並列分散プログラムを記述できるようにすることを目標に現在開発中の言語である．この言語のプログラムは，アルゴリズを記述する部分と，計算機資源を効率的に動的に割り当てるためのアノテーション部分をもつ．アノテーション部分は，プログラムのアルゴリズムを変更することなく，その効率だけを変更するもので，アルゴリズム記述部と似た構文で計算機資源を観察，変更するためのシステム変数群，関数群を使って記述される．

Sushi では，アノテーションの主なターゲットとして，プロセス・マイグレーションと実行優先度の動的変更を考えているので，以降の節で，これらを中心にその実現の構想を述べる．

2. プロセス・マイグレーション

プロセス・マイグレーションは，プログラムの実行の単位であるプロセスを物理的に違う多数の CPU 間を移動させることにより，各 CPU の負荷を動的に分散すると共に，プロセス間の通信遅延を減少させる，並列・分散計算の効率的な実行技法である．この技法は，大まかに二つに分類できる．一つは，プログラミング言語のレベルでプロセス・マイグレーションを実行する構文をもち，プログラムの中で明示的に記述する方法であり，もう一つは，OS のレベルで暗黙の内にプロセスのバイナリ・イメージを他の CPU に移動する方法で，通常，スワップ／ページングを拡張する形で実現される．

どちらの方法でも，共有メモリ型並列計算機上では，比較的安価なコストで実現可能である．しかしながら，分散メモリ型並列計算機（或は，非並列計算機のクラスタ）上では，OS レベルでの実現は，かなり高価なコストが掛かる．それは，一般に分散メモリ型の計算機は，CPU モジュール間の通信速度が，CPU と（局所）メモリ間の速度に比べかなり遅いということと，アプリケーションと転送するデータの種類に依存した形で，CPU モジュール間の最適なページングのブロックサイズが決まるためである．プログラミング言語のレベルでの実現は，各アプリケーションに最適なページングのブロックサイズを，プログラマが何らかの形でプログラムの中に記述できるようになっているので，OS レベルの実現に比べ，安価で効率的な実現が可能になる．

図 1．は，プロセス・マイグレーションの概念図である．各 CPU 1,2,3 は，深鍋状の形をし，それらの中にプロセス proc A, proc B, proc C,･･･, proc H を表す丸が入っている．プロセス間の矢印は，ストリームを通したデータの流れを表している．この図は，CPU 1 のロードが高く，かつ CPU 3 のロードが低い，しかも，proc C から proc G への通信量が多いとき，proc C が，CPU 1 から CPU 3 に，マイグレートしていることを表している．ここで，ロードが高い，低い，通信量が多いという語は，相対的なもので，プログラムの実行の各局面によっても，CPU の速度，ネットワークの構成・バンド幅によっても変わって来る．従って，プログラムの使われる環境，実行状態を予め固定してアノテーションをプログラムの中に埋め込む方法では，汎用性の高く効率の良いプログラムを作ることが困難である．

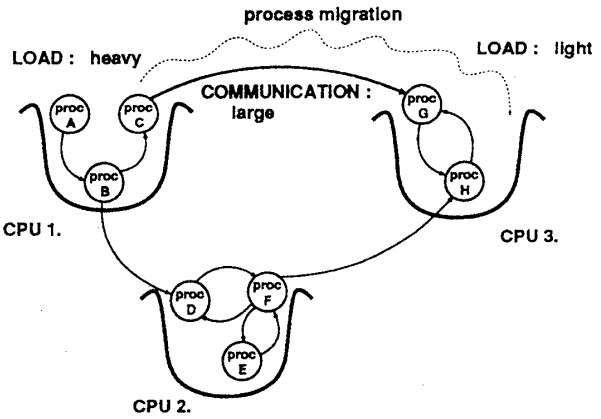

図 1. プロセス・マイグレーション

3. Sushi のアノテーション

プロセスの生成は，spawn 文を使って記述され，そのときアノテーションと呼ぶプログラムのアルゴリズムを変更しない，付加的な記述を付けることができる．このアノテーションは，プロセスのマイグレーションの戦略，実行優先度の動的な変更を記述する特殊なプログラムで，アルゴリズムを記述する言語と独立になっている．各プロセスに一つのアノテーションを付加するこ

On the Annotation Implementation of Parallel and Distributed Programming Language Sushi, Hideki Yamanaka, Fujitsu Laboratories, 1-9-3 Nakase, Mihama-ku, Chiba 261, Japan

とができ，プロセス実行のコンテキスト・スイッチの度にこの各アノテーションが実行される．ただし，各アノテーションの実行がある一定の時間以内に完了しないときには，割り込みによりそのアノテーションの実行を強制的に停止する．これは，アノテーションの実行がアルゴリズムに影響を与えないことと，アノテーションの実行は，一回毎に完結し，前の状態を保持しないことにより正当化される．また，アノテーションの一回目の計算は，特別扱いとし，以降の計算と違うものを指定できる．これは，最初にどのCPU上でプロセスを生成するのかが，プログラムの効率を大きな影響をもつからである．しかも，短命なプロセスで，2回目のコンテキスト・スイッチを必要としない場合，最初のアノテーションの計算だけがプログラムの効率に影響する．

Sushiでは，アノテーションをプロセスの実行中に何度も計算することにより，プログラムの使われる環境，実行状態を細かく反映したプロセス・マイグレーション，実行優先度の変更を可能とする予定である．アノテーションは，基本的な計算機環境（各CPU のロード・位置，各ストリームのリーダ・ライタとその通信量，各プロセスの計算時間，実時間など）を表す数値をシステム変数とし，その変数と各種算術演算を組み合わせた数式を元に，プロセス・マイグレーションと実行優先度を決定する形である．この形のアノテーションは，かなり汎用性の高いものであるが，特定の計算機環境に最適化したいときには，プログラムの中で使われるアノテーションを新しいものと置き換え，リンクし直すことが簡単にできる．

4. Sushi のアノテーションの実現構想

想定している Sushi の利用環境は，分散メモリ型並列計算機（或は，非並列計算機のクラスタ）で，そこに共有メモリとしてアクセスできるメモリ空間を構築する．それは，ある種のデマンド・ページングの仕組みで，Sushiのランタイムライブラリのパケット通信でシミュレートされる．つまり，各CPU上に一個づつタスク（あるいは，OSのプロセス）を立ち上げ，それらに共通のメモリ空間をリザーブし，さらに，その中の一部を必要に応じて動的にどのCPUの領域であるかを登録して行く．各CPUは，自分の領域をローカルなヒープ領域として使い，他のCPUの領域へのアクセスはトラップし，それをパケット通信によりリモート・アクセスに変換する．ただし，アクセスするデータ構造によって，最適になるようにその転送するブロック・サイズを変える．

Sushiのデータ構造は，文字列，ストリーム，連想配列とこれらの組合せである．すべてのデータは，このヒープ領域にその属性付きで取られる．また，Sushiの各プロセスは，タスク中にスレッドとして実現され，それらの使う各スレッド・スタックもこのヒープ上に取られる．そうして，スタック上には，ヒープ領域へのポインタとテキスト領域へのポインタだけが積まれるようにする．（テキストは，プログラムが立ち上がるときに各CPUに同じバイナリを供給し，ローカル・メモリの同じアドレス空間にロードしておく．） こうしておくと，Sushiのプロセスのマイグレーションは，単にそのプロセスのスレッド・スタックを元のCPUのヒープ空間から移動先のCPUのヒープ空間へコピーし，この二つのCPU上でスレッドの実行制御を調整し合うだけで済む．（ただし，スタック・ポインタ，フレーム・ポインタの値は，スタックから取り出されたときに，適当なオフセットが掛けられているものとする．）

アルゴリズム部分とアノテーション部分のSushiのソースを，各々独立にコンパイルし，Sushiの各プロセスは，スレッドに，各アノテーションはスレッド・スケジューラが呼び出すことができるに関数にされる．spawn 文によって結び付けられる，プロセスとアノテーションの組は，spawn 文をコンパイルしたコードの中に，スレッド・スケジューラに登録するコードとして埋め込まれる．最終的に，各々のコンパイル結果は，実行形式ファイルの中に一緒にリンクされる．そして，各アノテーションは，タスク中の一つのスレッドであるスレッド・スケジューラ内で時分割で実行される．

このスレッド・スケジューラは，実行優先度の付いた複数の実行キューとスリープ・キューを持ち，割り込み可能なラウンドロビン型のスケジュールをする．スレッドは，その入出力要求がブロックされると，自動的に割り込み待ちのスリープ・キューに入れられる．スレッド（Sushiのプロセス）に対応付けられたアノテーションは，そのスレッドの実行がタイムアウトで中断されてコンテキストスイッチした直後にタイムアウト付きで実行される．（ただし，最初の一回は，スレッドの作られる直前に実行する．スリープ・キュー内のスレッドのアノテーションは，実行されない．）

最後に，アノテーションの実行には分散環境の様々な統計情報を使うので，スレッド・スケジューラが，その情報の必要になった都度情報を集めていると実行のオーバヘッドが大きいので，適当な時間間隔で定期的に他のCPUのスレッド・スケジューラとの間で統計情報を交換し合うことにする．

5. まとめ

現在開発中の Sushi 処理系のうち，アノテーションの実現構想について，プロセス・マイグレーション部分を中心に述べた．

参考文献

[UGA94] 鵜飼孝典，山中英樹，上田晴康，プロセスの再配置が可能なストリームベース並列プログラミング言語，SWoPP'94, 沖縄, 1994.

[WAD94] 和田裕二，ストリームベースの並列分散プログラミング言語 Sushi における実行状態の可視化の試み，情報処理秋季全国大会，1994.

[YAM94] 山中英樹，鵜飼孝典，上田晴康，菅野博靖，和田裕二，Sushi – プロセスの再配置可能な並列分散プログラミング言語 –，情報処理秋季全国大会，1994.