Lites に対する並列分散データストリーム機能拡張の設計†

4Z－3 遠山 緑生[1] 安田 絹子[1] 平川 泰之[1] 斎藤 鉄也[1] 服部 隆志[2]
[1]慶應義塾大学大学院 メディア研究科 [2]慶應義塾大学 環境情報学部

1 はじめに

複数マシンから成る並列分散システム上でアプリケーションを開発するためには，複数の計算機資源を有効に扱うための使い易いインターフェースが求められる．我々は，複数マシンで動作する Mach マイクロカーネル上においてアプリケーションの開発・実行環境を提供する Lites (4.4BSD Lite Server)[1] の並列分散拡張を行っている．

これはネットワーク接続された同種の PC あるいはワークステーションから構成される並列分散コンピュータシステム上で動作し，UNIX のプロセス，パイプ，ファイルなどを並列分散拡張することで，従来のプログラミングスタイルを保持したまま並列分散プログラムに必要なサービスを提供する．本稿では，このうち並列分散データストリーム機能拡張の設計について述べる．

2 並列分散データストリーム機構

2.1 並列分散データストリームの概要

従来の UNIX においては，全てのデータ入出力をバイトストリームとして抽象化する事で，アプリケーションに対して非常に汎用的な入出力インターフェースを提供している．特にプロセス間の通信路として，単一方向の FIFO 型バイトストリームであるパイプが広く利用されている [2]．

本稿で提案する並列分散データストリーム機能は，パイプの概念を拡張する形での並列分散パイプを利用可能なインターフェースを提供することにより，従来の UNIX におけるパイプ処理の自然な拡張として，並列分散プロセス群を組み合わせた処理を複数マシン上に分散して構成することを可能とする．

まず実際のアプリケーション構築時に想定される，並列分散パイプの利用方法とその処理形態を下図に示す．

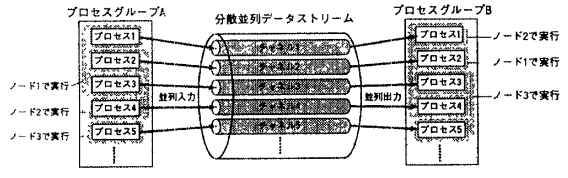

図 1: 並列分散データストリームの利用形態

Design of Parallel and Distributed Data Stream Extensions for Lites
Norio Touyama[1], Kinuko Yasuda[1], Tetsuya Saitou[1], Yasuyuki Hirakawa[1], and Takashi Hattori[2]
[1]Graduate School of Media and Governance, Keio University
[2]Faculty of Environmental Information, Keio University
E-Mail: <next@sfc.keio.ac.jp>

†本研究は，情報処理振興事業協会 (IPA) の創造的ソフトウェア育成事業「並列・分散処理基盤ソフトウェアの開発」の一部として実施している．

従来の UNIX におけるパイプに相当する並列分散パイプは，複数のチャネルに分割する事が可能となる．それぞれの分割されたチャネルは同時に並行して入出力可能である．この機能によって，二つのアプリケーションがそれぞれ n 個のプロセスから成り立つプロセス群であり，処理内容を n 個に分割して並列処理を行ない得る時に，その間で受け渡されるデータも n 個に分割して並列に通信可能となる．

このような場合には，対応する n 組のプロセスの間に n 本のチャネルを設定する．各チャネル毎に，パイプへの出力側の処理を行なうプロセス群に属するプロセスによって並列に書き込みが行なわれ，また入力側の処理を行なうプロセス群に属するプロセスによって並列に読み込みを行なうことが可能となる．

2.2 管理機構

並列分散 Lites では，並列分散プロセスの生成は他の機構と密接に関連して行なわれる．ここでは，並列分散 Lites の各機構のうち並列分散データストリームの管理機構の特徴について述べる．

並列分散データストリーム拡張は，具体的には以下の 3 つの特徴を持つデータストリームを提供する処理を組み合わせる事で，従来の UNIX パイプの持つ利便性と汎用性の高さを極力失わない形で利用可能な並列分散パイプ機能を提供するものである．

- ノード間に分散したプロセス間においても，パイプ型のデータストリームを利用可能とする分散データストリーム機能
- 各アプリケーションを構成するプロセス群に属する複数の処理プロセスが，一つのデータストリームに対して同時に入出力可能とする並列データストリーム機能
- 各データストリームごとの並列性やノード位置情報など，データストリームが持つ属性をパイプの管理情報に付加し，対応する入出力のプロセス群の間で通信形態の情報を伝達可能にする属性管理機能

並列分散データストリーム生成機構は，これらの特徴を持つ並列分散パイプの生成，管理を行う．UNIX のパイプと同様にな read(), write() といった標準的な入出力機構に対する操作を，複数ノード上に分散されたプロセス群の間を結合するデータストリームに対して提供する．

また，並列分散 Lites の最大の特徴は，システム内に分散するデータの位置に従って並列分散プロセスを生成できるという点である．このため，並列分散プロセス管理機構との協調によって，データを受けとる側のプロセスを，データを出力する側のプロセスのノード位置に合わせて生成することができるように，プロセスの位置情報をパイプを通じて伝える事ができる属性管理機能を提供する．これにより，ノード間通信によるコストを削減した効率的なプロセスの結合を可能とする．

2.3 利用操作

並列分散データストリーム機能は，具体的には次のような手順で利用される．

1. 一つの並列分散パイプは，従来の UNIX における pipe() システムコールと同様なインターフェースを持つ，pdpipe() によって作成される．これは，実際の利用においては，並列分散拡張されたシェルによって行われる事を想定している．

この時に並列分散パイプ用の管理エントリが作成され，ファイル構造体から参照が設定される．このエントリは，チャネル数やノードテーブルなどのパイプの属性，実際の入出力時に利用される各チャネルのエンドポイントの情報などを保持するものであり，以後のパイプに対する操作によって利用，設定される．pdpipe() が行われたノードにおいて管理されるが，一つのパイプについて以後に fork(), pfork(), pvfork() された全ノード上の関連プロセス間で共有される．

2. pdpipe() を行ったプロセスは，UNIX パイプの一般的な利用方法と同様に，次に fork() によって並列分散パイプに対する入力側のプロセスと出力側のプロセスを生成する．この時点で生成されるプロセスは，入出力それぞれのプロセス群の親プロセスとなり，この後で生成される子プロセス群の管理を行う．

3. パイプに対する出力側の親プロセスは，pdsetattr() によってチャネル数や，パイプに対して出力を行う側の子プロセス群のノード位置を示すノードテーブルなどから成る並列分散パイプの属性をパイプ管理エントリに対して設定する．

4. pdsetattr() によって出力側が設定したパイプの属性は，データストリームから入力を行う側の親プロセスが pdgetattr() によってパイプ管理エントリから得る事ができる．

5. pdsetattr(), pdgetattr() に利用したのと同じノードテーブルを引数として，入出力側の親プロセスがそれぞれ pfork() または pvfork() を行い，プロセス群を構成する子プロセスを生成する．

6. 生成された各子プロセス群は，プロセスの初期化操作の一貫として，親プロセスから継承されてオープンされている並列分散パイプの入出力位置の再設定 pdsetfp() を行なう．これによって，入出力を行うチャネルを設定することになる．

2.4 プロセス管理機構との協調

並列分散データストリーム機構は，全体に並列分散プロセス管理機構との協調を行うが，特に上記の 3〜6 におけるノードテーブルなどの並列属性の指定は，プロセス管理機構との協調が重要となる．

例えば，並列分散パイプによって結合された複数のプロセス群によってパイプラインを構成する場合，まず最初のプロセス群の親プロセスが，自分自身の pfork(), pvfork() を行うのに利用するノードテーブルを用いてパイプに対する属性指定 pdsetattr() を行う．この際，最初のプロセスは自分でノードテーブルを生成するか，もしくは入力として用いる並列分散ファイルから得られるノードテーブルを利用する．

パイプからデータを受け取る次のプロセスは，最初のプロセスが行った並列属性の指定を pdgetattr() を用いて読み出す．この中には最初のプロセスが設定を行ったノードテーブルの情報が含まれているので，その情報に応じて自分自身の pfork(), pvfork() を行うことができ，生成された各子プロセスは自分が走っているノードと同一ノードにある出力側プロセスとチャネルを結合する事が可能となる．

このプロセス群がさらに出力を持つ場合には，並列分散ファイルに対してノードテーブルに対応した出力を行うか，出力が次のパイプに結合されているならば，再びノードテーブルを用いた並列属性をそのパイプによって設定することによって，さらに次のプロセス群に対してノードテーブルを伝える事ができる．

このように，各プロセス群を構成するプロセスの位置情報をパイプを通じて次のプロセス群に伝播させていくことができ，各子プロセスは位置情報に応じた生成が可能になるため，理想的な場合では各システムコール内で転送される内部情報以外のデータは全てローカルノード上で処理を行う事が可能となる．

また，実際にノードテーブルの情報を利用して pfork(), pvfork() を行う親プロセスは，次に示す 3 つのパターンがあり得る．

1. プロセス群を構成する親プロセスが並列分散 Lites に合わせて書かれたものである場合，親プロセスは自分で明示的に pdsetattr(), pfork() などの操作を行う．
2. grep などの既存のフィルタプログラムなどを単純に変更なしに利用したい時は，親プロセスはシェルの組み込み機能として実行し，既存のプログラムは複数の子プロセスとして並列に実行される．この場合，ノードテーブル関連の一連の操作はシェル側によって行われることになる．
3. パイプに対する出力を行うプロセスが複数チャネルに並列に出力するのに対し，受け取る側のプロセスが非並列で 1 本にまとめて受け取る必要がある場合 (sort コマンドなど) には，親プロセスとして用途に応じた wrapper プログラムを利用する．この場合には一連の操作は wrapper によって行われる．

3 まとめ

本稿では，Lites に対する複数計算機資源を有効に扱うための並列分散機能拡張のうち，並列分散データストリーム機構の設計について述べた．UNIX におけるパイプの概念を拡張した並列分散パイプを提供する事で，従来のプログラミングスタイルを保持したまま並列分散プログラムを構成し，複数のプロセス群を効率的に結合する事を可能とする．

参考文献

[1] J. Helander, "Unix under Mach - The Lites Server", Master Thesis, Helsinki University, Dec 30, 1994.

[2] Marshall Kirk McKusick, Keith Bostic, Micheal J. karels, John S. Quarterman, "The Design and Implementation of the 4.4 BSD Operating System", Addison-Wesley publishing, 1996.

[3] Erik P. DeBenedictis, Stephen C. Johnson, "Extending Unix for Scalbale Computing", IEEE Computer, Nov. 1993.