分散並列OS「Orion」の試作

2P−2　並列計算におけるネームサーバの課題とその解決

宇都宮直樹，岩嵜正明，吉澤聡，千葉寛之，薗田浩二，山内雅彦
(株)日立製作所　中央研究所

1. はじめに

本稿では分散システム中のネームサーバに注目し、これを並列計算の世界に導入する場合の課題と解決法について述べる。

ネームサーバを導入する目的は、プログラムを実行環境に依存しない形で記述することである。分散システムで既知のとおり、ネームサーバは位置透過性を提供する。この位置透過性により、ソースコード中で書かれた資源をアクセスするために用いる名前は、このプログラムがどのような環境で実行されようとも常に同じ資源を指し示す。従って、実行環境に依存しないプログラミングが可能となる。

2. ネームサーバの概要

Orionシステムのネームサーバはシステム定義名とユーザ定義名の二種類の名前を提供する。システム定義名はOSが資源に与えるID (OID: Object IDentifer) でOS内でその資源を管理する目的で使用される。ユーザ定義名は使用する資源に対してユーザが任意に与える。

目的の資源にアクセスするためには、ネームサーバから位置情報(ロケーション)を取得して、どのノードのサーバに処理を依頼するかを決定する必要がある。この名前の解決は、以下の手順で行われる。まずロケーションを要求するユーザはネームサーバに名前を与える。ネームサーバは与えられた名前がユーザ定義名の場合は、IDへの解決ルーチンにより対応するシステム定義名に変換する。次に、システム定義名を用いてシステム定義名とロケーションの対応表(ロケーションテーブル)を検索し、目的のロケーションとその属性をユーザに返す。

3. ネームサーバの実装

Orionシステムでは以下の理由によりネームサーバを分散形態で実装している。

- ●ネームサーバは頻繁にアクセスされるので、アクセスはなるべくローカルな通信で行う必要がある。
- ●ノード数が多い場合、1つのネームサーバで集中的に処理を行うと、そのサーバが性能上のボトルネックになる。

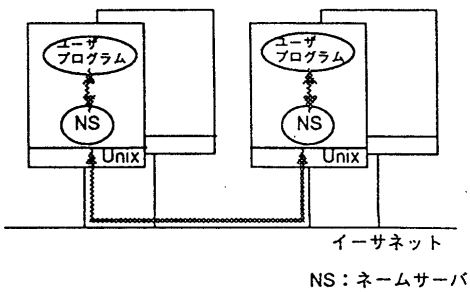
しかし、このような分散形態のネームサーバでは、ロケーションテーブルの複製の内容の一貫性を取る必要がある。そこで、この課題に対してOrionでは、Lamport [1] が提案したevent orderingに基づく分散排他制御アルゴリズムを用いて、ロケーションテーブルの一貫性制御を行うことにした。

次に、排他制御をどのような形で各サービスに適応するかについて述べる。ネームサーバはユーザに対して次のようなサービスを提供する。(1) OIDの取得、(2) OIDとロケーションの登録、(3) 属性情報の変更、(4) OIDをキーとしたロケーションの問い合わせ。各ネームサーバは、各ノードで並列に動作し、ローカルなユーザに対して上記のサービスを行う。ロケーションテーブルの一貫性を損なわずにこのサービスを遂行するためには、一貫性を損なう可能性のあるサービスを排他的に実行する必要がある。ロケーションテーブルの一貫性を損なう可能性のあるサービスは、(2) ロケーションの登録と(3) 属性情報の変更である。さらに、(4) ロケーションの問い合わせに関しても、別のノードで並行してロケーションテーブルが変更されている可能性があるので、排他的にサービスする必要がある。

排他制御を行ってサービスを逐次的に行う場合、新たな要求がサービスを受けるためには、その時点で既に発行されていた要求のサービスが全て完了する必要がある。従って、並列にサービスを要求しているプロセスの数(競合するプロセスの数)によってネームサービスの性能が左右される。特に、ロケーションの問い合わせは、資源にアクセスする度に使用され、この性能が全体の性能を大きく左右する。性能向上のためには、問い合わせ処理の排他制御を省略する必要があるが、問い合わせの結果の厳密さ、即ち、ロケーション情報の一貫性が崩れてしまう。従って、性能と一貫性のバランスを取ることが重要である。

4. 性能の測定

性能と排他制御の関係を定量的に実測評価した。実測は図1に示す構成で、排他制御のオーバヘッドを測るため



NS：ネームサーバ

図1．ネームサーバの構成

Prototyping of Distributed-Parallel Operating System "Orion"
- Issues of Name Server and their Solutions in Parallel Computation -
Naoki UTSUNOMIYA, Masaaki IWASAKI, Satoshi YOSHIZAWA, Hiroyuki CHIBA, Kouji SONODA, Masahiko YAMAUCHI
Central Research Laboratory, Hitachi, Ltd.

に、排他制御付きと排他制御無しの問い合わせ処理時間を測定した。

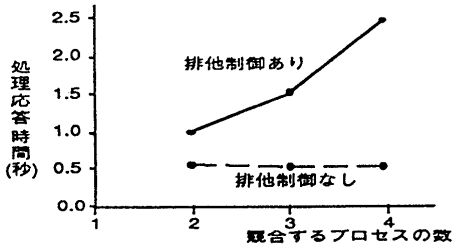


図2．ネームサーバの性能測定結果

この性能を測定した結果が図2である。これによると、排他制御を行わない場合は、応答時間が、競合するプロセスの数に依存しないが、排他制御処理を行うと、応答時間が、競合するプロセス数に依存して長くなることが分かる。この結果から、性能と一貫性のバランスを取る必要性が確認できる。

5. 排他制御処理の課題とその解決

性能と一貫性のバランスをとるために、まず、ネームサーバの高速化の課題について以下の側面を考える。

(1) サーバとの通信処理のオーバヘッドの削減

(2) 分散排他制御処理に必要なメッセージの数の削減

(3) 競合が起きたときの待ち時間の削減

(1) についてはキャッシュを用いて対処することができる[5]。(2), (3) の課題に対処するためには、二つのアプローチがある。一つは、厳密な一貫性を保ちながらアルゴリズムを改善する方法である。即ち、分散並行制御アルゴリズムの改良であり、さまざまな方式が提案されている[2]。これに対して、別のアプローチは、一貫性をある程度崩すことによって、高速化する方式である。

Orionシステムでは後者のアプローチを取る。アプリケーションとして並列数値計算を考える場合、複雑なアルゴリズムはかえって性能の低下を招くからであるのと、数値計算の実行環境は比較的静的であり、厳密な一貫性はかならずしも必要でない場合が多いからである。そこで、数値計算アプリケーションに限らず、一般的なアプリケーションが一貫性の厳密さと性能とのバランスを制御できるように、アプリケーションプログラムインタフェースを定める[3]。その指標として一貫性の強さという概念を導入する。

一貫性の強さは時間的な側面と空間的な側面がある。時間的な側面を定めるパラメータとして、以下のものを考える。

(A) ある項目 (ネームサーバの場合はロケーション情報) が変更される頻度

(B) ある項目が有効である期間

(C) 変更時刻

空間的なパラメータとしては以下のものを考える。

(a) ある項目について一貫性を保つ必要のあるノードの集合

このようなパラメータをアプリケーションプログラムが定め、ヒントとしてネームサーバに渡すことによって、アプリケーションの性質を活かした一貫性の緩和を行うことができる。ここでは、SPMD (Single Program Multiple Data) 方式[4]に、一貫性の緩和を適用したときの例について述べる。

SPMD 方式の特徴は、一旦プログラムが各ノード上に割り付けられると、後は計算が終わるまでノードとプログラムの対応は変化しないことである。この静的な性質は時間的なパラメータ (A) 変更の頻度が0であることを意味し、これにより、参照時の排他制御を省略することができるとネームサーバは判断する。

さらに、この数値計算における通信は全てのノードと行うわけではなく、通常は、データを分割したときの境界を挟んで隣接するノード (隣接ノード) とのみ通信を行う。従って、空間的な側面で一貫性を弛めることができる。つまり、空間的なパラメータの (a) として隣接ノードの集合を与えることにより、一貫性を保つノードの数を制限することができる。プログラムをプロセスとして複数のノードに分配するとき、即ち、ロケーションテーブルのエントリを作成するときは、エントリの複製を (a) で示されるノードにのみ分配すればよく、通信量を減らすことができる。

6. おわりに

分散環境上で並列計算を行うとき、ネームサーバはプログラムの移植性や開発の容易さの面で有利になるが、他方、性能の面で問題がある。そこで、実装したネームサーバのデータから、性能向上の見通しを立て、解決策を提示した。今後は、この解決策を実装しその効果を評価する予定である。

参考文献

[1] Lamport, L., Time, Clocks, and the Ordering of Events in a Distributed Systems, CACM, Vol. 21, No.7, pp. 558-565 (1978).

[2] Bernstein, P.A., Hadzilcos, V., and Goodman, N., "Concurrency Control and Recovery in Database Systems," Addison-Wesley, 1987.

[3] Sinha, P.K., Supporting Data Sharing Facilities in the GALAXY Distributed Operating System, Doctor thesis, The University of Tokyo, 1990.

[4] 岩嵜, 他, 分散並列OS「Orion」の試作－システムの概要, 情報処理学会第45回全国大会予稿集, 2P-01, 1992.

[5] 薗田, 他, 分散並列OS「Orion」の試作－メッセージサーバの実装, 情報処理学会第45回全国大会予稿集, 2P-04, 1992.