デペンダブルストレージシステム

喜連川優
情報理工学系研究科電子情報学専攻

概要

ストレージシステムのデペンダビリティに対する期待はとりわけ、９・１１以降、大きい。業界によっては、法制度化により、システム構成への信頼性強化が強いられる状況にもなりつつある。また、e-business における予測不能な負荷変動への対応も強く求められている。

一方、ストレージネットワーキング技術が近年大きく進展し、当該技術を利用した新たなソリューションが模索されている。

本研究では、ファイバチャネルによるストレージネットワーキング技術を利用したクラスタシステムを実験用プラットフォームとして構築し、動的負荷変動に強固なストレージ仮想化技術に関して独自の方式を提案すると同時に、実装により有効性を明らかにする。

1 はじめに

本研究ではストレージエリアネットワーク（SAN）を適用した共有ディスク方式のクラスタシステムに於いて、データインテンシブアプリケーションを対象とした動的資源調節の実現を目的とする。ここで、動的資源調節とは実行時に於いてアプリケーションにIO帯域やCPU演算能力等の資源を必要に応じて配置する機能を示す。従来、クラスタシステムではストレージが個々のストレージを管理する Shared Nothing 方式のストレージアーキテクチャが主であり、実行時にディスク間でデータを移送することが難しいという問題があった。このため、負荷分散における台数効果の改善に限界があり、動的資源配置は困難であった。本研究では、共有ディスク方式のクラスタシステムに、共有読み込みと動的デクラスタリングなる２つの機能を提案する。両機能を用いることにより、データインテンシブアプリケーションにおいて動的資源調節を実現することが可能となる。さらに、並列データマイニングを用いた実装実験の結果を示し、提案手法の有効性を示す。

2 ストレージ仮想化機構

2.1 概要

データインテンシブアプリケーションを対象とした共有ディスク方式のクラスタシステムの概要を図 1 に示す。サーバ間は IP ネットワークによって、サーバ-ディスク間はストレージネットワークによって接続されている。クラスタシステムはサーバ資源の集合であるサーバプール、それを管理する負荷分散器、ストレージ資源の集合であるストレージプール、およびそれを管理するメタサーバから構成される。ストレージプールはストレージ仮想化機構によって管理される。ストレージ仮想化機構は単一のメタサーバと複数の LSM（Logical Storage Manager）から構成される機構であり、共有ディスク方式の IO アクセスを採用するファイルシステムである。ストレージ仮想化機構により、ストレージプールは全サーバから共有され、仮想化ストレージ空間が構成され、ファイルの物理ディスクへの割り当て(物理-論理アドレス変換)が行われる。メタサーバはストレー

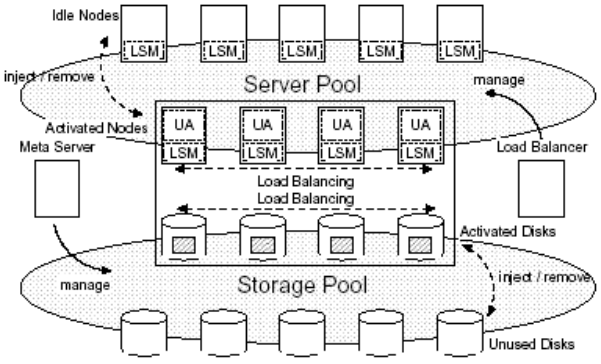

図 1 ストレージ仮想化機構を用いたクラスタシステム

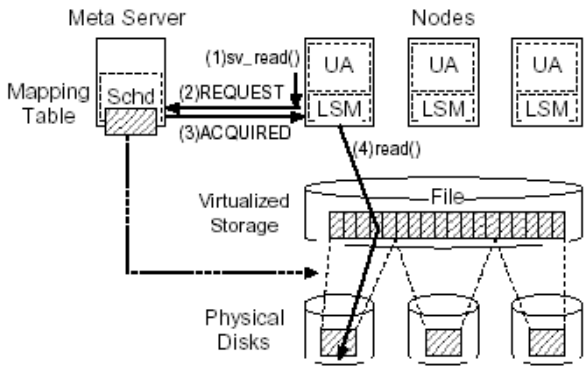

図 2 共有読み込み

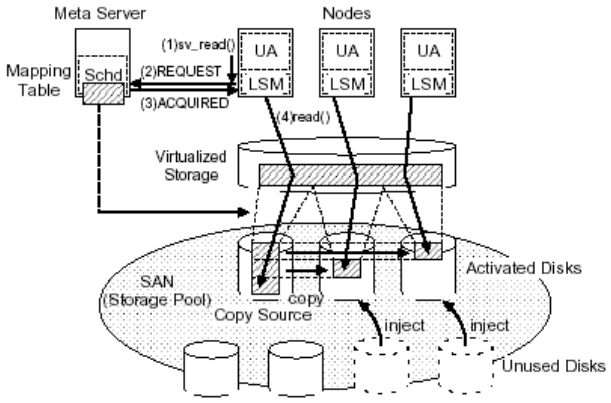

図 3 動的デクラスタリング

ジプールを集中管理しているため、ディスク間の IO 負荷を容易に実現することができる。

一方、サーバプール内では負荷分散器がサーバへのアプリケーションプロセスの割り当てを行い、必要に応じてサーバ間の負荷分散を行う。このとき、負荷分散器はアプリケーション固有の知識を用いることがある。

ストレージ仮想化機構は共有読み込みと動的資源調節なる２つの機能を有する。

2.2 共有読み込み

共有読み込みはユーザの指定する複数のサーバ間で、共有ファイルのシークポインタを共有する機能である。図２に共有読み込みの概要を示す。読み込みシステムコールの呼び出しによって、LSM はメタサーバにメタ情報の問い合わせを行い、メタサーバは読み出すべき領域をサーバ用にロックして通知する、この時、メタサーバはあらかじめ複数のシステムコール呼び出しに相当するメタ情報をまとめてサーバに通知することにより、システムコールの読み出し毎にメタサーバへ問い合わせるオーバヘッドを削減することができる。

巨大なファイルを逐次読み出して処理するアプリケーションでは、高い台数効果を得るため、各サーバの実行時間を等しくする必要がある。このためには、サーバが処理すべきデータ量を適切に制御する必要がある。Shared Nothing 方式のストレージアーキテクチャを用いたクラスタシステムの場合、アプリケーション実行前にあらかじめサーバのディスクにデータを分散させておく必要があるが、この見積もりが正しくない場合や、負荷が動的に変動する場合など、サーバが処理すべきデータ量を適切に制御することは難しい。一方、本研究が提案する共有読み込み機能では、シークポインタを共有することにより、ストレージ仮想化機構がサーバの要求に応じてオンデマンドにデータを分配することが可能となる。データ量調節のための特別な制御をアプリケーションは導入する必要がなくなり、アプリケーションの設計を単純化することが可能になるとともに、台数効果の向上が期待できる。

2.3 動的デクラスタリング

動的デクラスタリングはストレージ仮想化機構内においてユーザの指定したファイル毎の IO 帯域を自動調節する機能である。図３に動的デクラスタリングの概要を示す。動的デクラスタリングは、アプリケーション実行中にデータが存在しているストレージデバイスからデータを分割して未利用のストレージ空間に投機的にコピーを行い、後に並列アクセスを行うことにより、I/O 帯域を拡張する。 投機的コピーが完成したのち、メタサーバはアプリケーションの IO スループットを計測し、アプリケーションがより多くの I/O 帯域を必要としている場合は IO アクセスの並列度を上げることにより IO 帯域を拡張し、反対に IO 帯域が余っている場合には並列度を下げることにより IO 帯域を縮退する。

共有読み込みと動的デクラスタリングを用いることにより、クラスタシステムにおいてデータインテンシブアプリケーションのプロセスは任意のサーバにおいて実行することができる。アプリケーション実行時に統計情報を用い、実行サーバ数を変更し、CPU 演算能力を調節する方式はすでに多くの研究がなされている。このような制御をストレージ仮想化機構上で行うことにより、

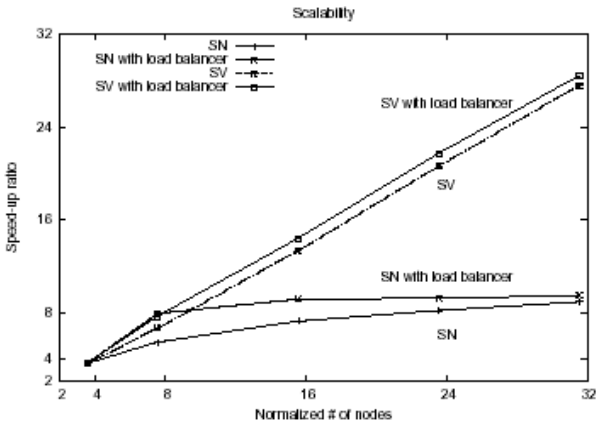

図 4 共有読み込みによる台数効果の向上

クラスタシステムにおいてデータインテンシブアプリケーションのIO帯域とCPU演算能力を動的に調節することが可能となる。

3 並列データマイニングアプリケーションを用いた評価実験

3.1 SAN結合PCクラスタと実装

データインテンシブアプリケーションの代表的な一例として並列データマイニング処理を取り上げ、ファイバチャネルおよびギガビットイーサによって PC とディスクが接続された SAN 結合 PC クラスタにおいて、ストレージ仮想化機構を用いた実装を行い計測した動的資源調節の評価実験結果を示す。並列データマイニング処理では大容量かつスキューのあるトランザクションデータベースを扱う。その処理は通常複数のパスから構成されるが、各パスによってその負荷特性が異なり、CPU 演算能力や I/O 帯域などを適宜調節する必要がある。

3.2 共有読み込みによる負荷分散の台数効果の向上

ストレージ仮想化機構において共有読み込みによる負荷分散における台数効果の向上を調査するために、サーバ数を変更して実行した。この時、クラスタシステムにおいてアプリケーションの知識を用いる負荷分散器を用いない場合、用いる場合それぞれにおいて測定した。さらに比較のため Shared Nothing 方式で負荷分散器を用いない場合、用いる場合を比較した。この時、過酷な負荷の偏りを発生させるために、1 台のみを Pentium Pro 200MHz を搭載するサーバとし、残りは Pentium III 800MHz のサーバとした。サーバ数 4 を起点としたスピードアップ曲線を図 5 に示す。この時、横軸は 800MHz の CPU を 1 とし、総クロック数で正規化を行った。例えば、800MHz のサーバ 3 台に 200MHz のサーバを 1 台用いる場合、正規化サーバ数は 3。25 となる。ストレージ仮想化機構により共有読み込みを行う場合は、負荷分散器の適用に係わらず高い台数効果が得られているが、Shared Nothing 方式では、サーバ数 8 以降の性能改善は見られない。Shared Nothing 方式ではサーバ数が増加してもデータがサーバに依存し、データ量の偏りの影響を受ける為であり、専用の負荷分散制御による改善もわずかである。以上から、共有読み込みを用いることにより、ディスク上のデータ配置に依存せず、高い台数効果が得られることがわかる。

3.3 動的資源調節

32 台のサーバおよび 4 台のディスクの環境における動的資源調節の実行トレースを図 6 に示す。この時、7。3GB のトランザクションデータベースを用いた。パス 1 は I/O バウンドであるため、サーバ数の変化は見られず、投機的な分割コピー作成が行われる。その後、パス 2 が開始し負荷分散器は CPU バウンドを判断し、逐次サーバを追加し、サーバ数は 16 へと段階的に増加する。サーバ数が 16 になった際に、全サーバが I/O バウンドへと移行し、サーバの追加投入は停止する。ディスクのスループットが飽和しているため、これ以上のスループットを見込めないためである。その後、メタサーバは I/O 並列度を上げる。これにより、I/O 帯域が拡張し、ディスクのスループットが増大する。16 台のサーバは再び、CPU バウンドに移行し、最終的にサーバプール内 32 台全てのサーバが用いられる。また、パス 3 以降では CPU 演算能力が余っていると判断し、逐次サーバ数を減少させる。

動的資源調節による性能改善を実行時間により確認するため、29GB のトランザクションを用意し、動的資源調節を行わない場合、CPU 演算能力の動的調節のみを行う場合、および CPU 演算能力・IO 帯域双方の動的調節を行う場合について各パスの実行時間を測定した。結果を図 6 に示す。CPU 演算能力の動的投入のみを行う場合、最大で 16 サーバ 1 ディスクまで拡張が行われ、パス 2 が大幅に改善している一方、パス 3 以降は改善しない。IO 帯域の拡張を併せて用いる場合、最大で 32 サーバ 4 ディスクと予め設定した資源全てを使用した。パス 2 は動的資源拡張を行わない場合と比較し、約 22 倍に改善しており、

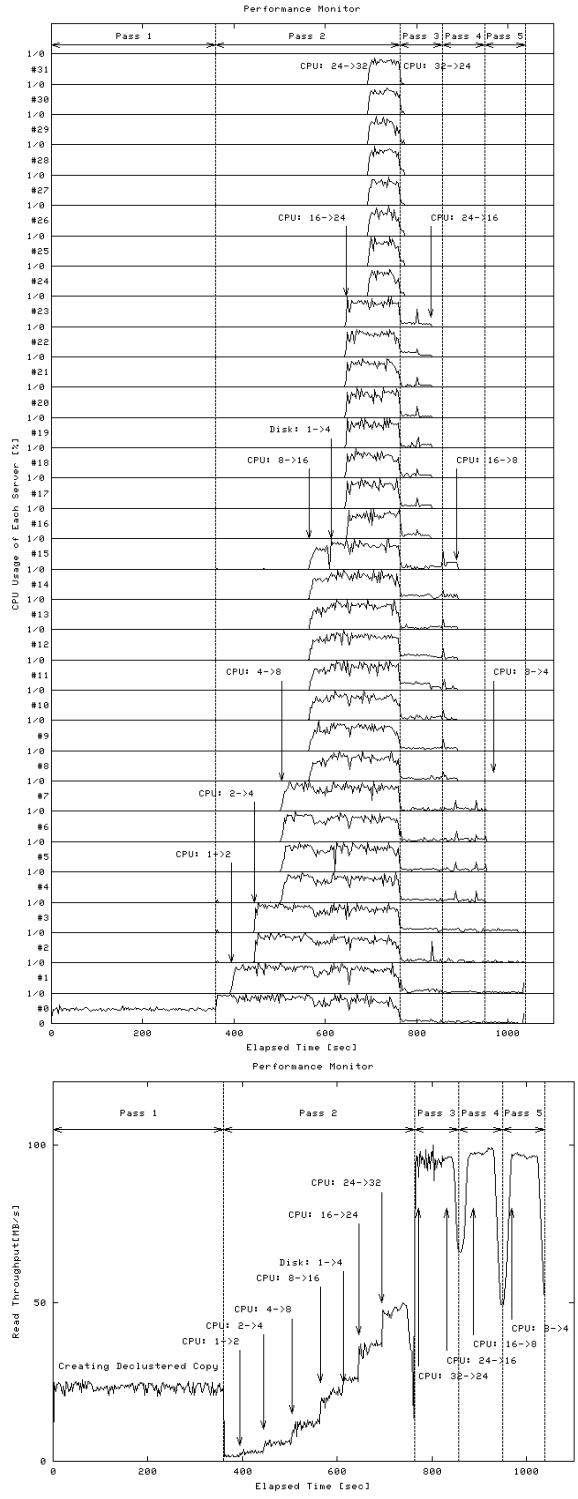

図 5 動的資源調節の実行トレース

これは、逐次サーバを増加させる過程を考慮すると、十分な性能改善値であると言える。パス 1 を含んだ全パスの実行時間の性能改善は約 7 倍であった。

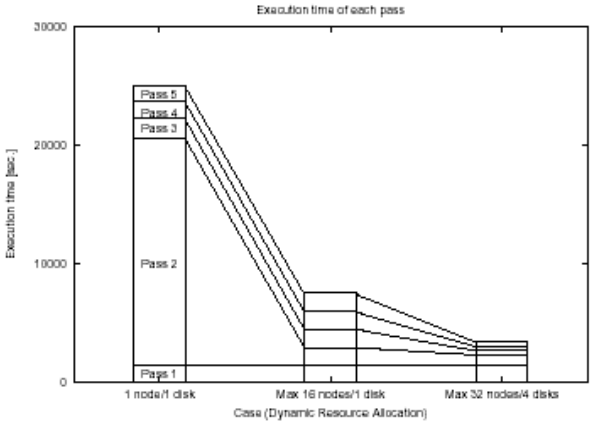

図 6 動的資源調節による実行時間の改善

以上の実験により、並列データマイニングアプリケーションを例に、クラスタシステムにおいて提案手法により、IO 帯域および CPU 演算能力の動的な資源調節が可能になることを実証した。

4 まとめと今後の課題

ストレージネットワークを用いた共有ディスク方式の IO アクセスを行うクラスタシステムにおいて、共有読み込みおよび動的デクラスタリングなる機能を提案した。両機能により負荷分散における台数効果が向上し、動的資源調節を実現可能であることを述べた。提案手法を SAN 結合 PC クラスタにおいて実装し、並列データマイニングアプリケーションを用いた実験により評価した。この結果、提案手法の有効性を示した。

提案手法は並列データマイニングアプリケーションのみならず、多くのアプリケーションにおいて有効である。今後実験により実証する予定である。また、複数のアプリケーションや複数のユーザが混在するなど、より複雑な環境において、アプリケーションやユーザ毎に適切な IO 資源を割り当て、適切なスケジュール調節やキャッシュ制御を行うことにより、ストレージのデペンダビリティ向上が期待できる。当該研究課題に関し、研究をすすめたい。