PC クラスタを用いたサーバーベースドコンピューティング方式の提案

渡辺 英俊　伊藤 健一　田中 一男
(株)NTT データ 技術開発本部
e-mail: {watanabehdt, itoukn, tanakakza}@nttdata.co.jp

1. はじめに

今日、一般企業においても PC の利用は不可欠なものとなっている。しかし、PC の普及と社内ネットワークの広域化により、分散して存在する多数の端末を管理する負担は増加している。その一つの問題解決策として、ユーザが利用するデータや計算機資源をサーバに集め、ユーザのデスクトップをサーバ上で仮想化し、入出力のみユーザの手許の端末で行なうサーバーベースドコンピューティング(SBC)がある。近年のインターネット回線の高速化・低価格化により、SBC の適用領域は LAN 内だけでなく、遠隔地からの利用にも広がると予想される。

SBC 製品の代表例として、Sun Ray や Windows Based Terminal 等があるが、サーバの構築費用が高い、ユーザ数や処理内容の増加に対応するための柔軟な規模拡大が難しい、サーバの一部に障害が発生した場合の影響範囲が大きい、等の問題を抱えている。そして、長期的な不況により、企業が IT 投資に求める費用対効果は非常に大きくなっている。

我々は、可用性とスケーラビリティに優れた SBC 環境を安価に実現する方式の研究を行なっている。本稿では、PC クラスタによる SBC の実現方式を提案し、また、現時点での実装状況も報告する。

2. 要求条件

PC クラスタは、ノード単体の絶対的な性能や信頼性が低い反面、価格性能比は高い。また手軽にノード数の変更ができる等の利点がある。PC クラスタ上で SBC を実現するにあたり、以下の要求条件を設定した。

(1) システム構築費用を低く抑える
(2) 短期間に効率的に開発を進める
(3) ノンストップサービスを実現する
(4) ユーザや AP から見えるシステム構成が常に同一
(5) ユーザのログイン環境が場所によらず同一

3. 提案方式

上記の要求条件を満たすため、以下の方針で設計を行なう。

A server-based computing architecture by using PC cluster.
Hidetoshi WATANABE, Ken'ichi ITO and Kazuo TANAKA,
Research and Development Headquarters, NTT DATA CORPORATION

3.1. コモディティハードとオープンソースの利用

コモディティな PC を複数台接続したシェアードナッシング型 PC クラスタを用いることにより、システム構築費用を抑える。汎用機やハイエンドサーバに比べてノード単位の性能や信頼性が低い点はソフトウェアによる信頼性確保策を採用することにより補う。

システムの開発コストを抑えるとともに開発期間の短縮を行なう観点から、核となる部分以外はオープンソースソフトウェアを利用する。クラスタ内の各ノードでは Linux ベースの OS 環境を構築し、端末との接続には遠隔デスクトッププログラムである VNC[1]を使用する。

3.2. 構成の自動認識

各ノードは疎結合であるため、ハードウェア的なノードの増減は簡単に行なえる。しかし、そのハードをすぐにシステム資源として使うためにはソフトウェアの対応が不可欠である。本方式では、ノードの増減によるサーバ能力の変化を自動的かつ動的に認識できるようにするため、各ノードの存在情報と負荷状況を「構成情報」と呼ぶ共有ファイルで管理する。新規にクラスタに参加したノードは、自分自身で構成情報を更新する。ノード障害が発生した場合は、残存ノードが処理依頼のタイムアウト等により検知し、当該ノードの構成情報を「障害」とする。

3.3. セッションの静的分散

1 台の端末が複数の AP 起動を並行して要求する場合、同一ノードが全要求の処理を行なうと過負荷状態になりやすい。そのため、セッションを単位とした負荷分散をノード間で行なう。端末と仮想デスクトップ間の通信をデスクトップセッション、仮想デスクトップと AP 間の通信をアプリケーションセッションと定義する。以下にセッションの接続処理の流れを示す。

① クライアントは、デスクトップセッションを介して、ノードにアプリケーションセッションの接続要求を行なう。
② 接続要求を受け取ったノード(要求受付ノード)は、共有ファイル上の構成情報を参照し、自ノードも含めた各ノードの負荷状態から、セッションの処理に最適なノード(最適ノード)を決定する。
③ 選択されたノードが要求受付ノードの場合は、そのままセッションを構築する。それ以外の場合、端末に

通知するとともに選択されたノードにセッションを転送する。

3.4. セッションの動的分散

時間の経過とともにシステムの負荷状態は変化する。そこで、実行中の処理のうち、I/O 操作を伴わない部分(ノード非依存部)については、他ノードの CPU を利用することにより、セッションの動的な負荷分散を実現する。

セッションが接続されているノードは構成情報を参照し、他ノードの負荷が低い場合にはノード非依存部の処理を負荷の軽いノードへ移送する。I/O 操作を伴う部分(ノード依存部)については、自ノードで処理を行なう。

この方式とすることで、ノード非依存部を実行しているノードに障害が発生した場合には、セッションを保持しているノードが異常を検出し、他のノードにノード非依存部の処理を再依頼することによりサービスを継続することができる。また、セッションを保持するノードは何らかの二重化対策を行なう必要があるが、セッションを保持するノードの処理はノード依存部だけになるため、セッションの多重度を上げることが可能になり、二重化する範囲を最小限にすることが可能である。(図 1)

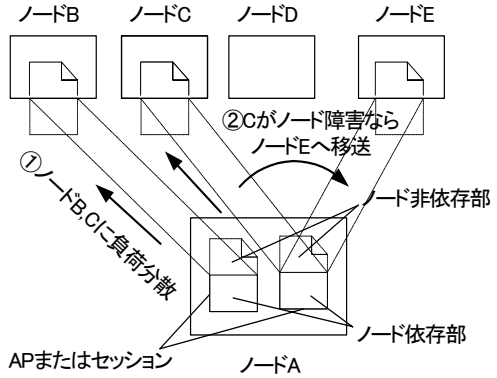

図 1 セッションの負荷分散概念図

3.5. シングルシステムイメージ

利用者の利便性や AP のプログラミングを考えると、システムのノード状態が変化しても、システムのリソースが一つのシステムのように仮想化することが有効である。CPU については上記の方法で仮想化を行っている。ファイルシステムについても相互のノードのファイルシステムを共通のパス名でクロスマウントすることにより、どのノード上でも、同一のパス名でサーバ内の同じファイルにアクセスする環境を実現している。

3.6. セッション永続化

デスクトップがサーバ上で仮想化されるため、ログイン中のセッションを維持したまま端末の変更が可能である。ユーザは、異なる端末から同一の環境にログインできるため、出張や外出先等でも普段と同じデスクトップ環境を使用できる。

4. プロトタイプ

上述の設計に基づいて、PC クラスタによる SBC のプロトタイプを構築している。以下に、そのモジュール構成を示す。(図 2)

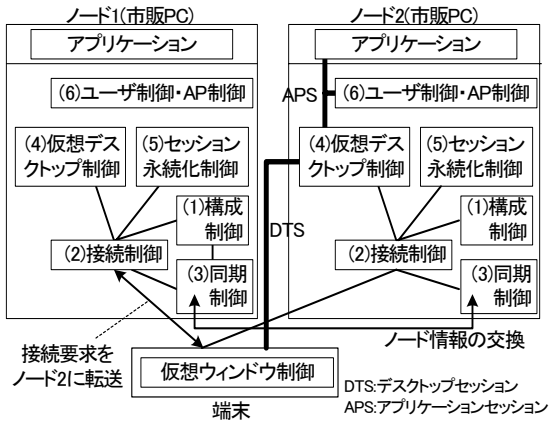

図 2 本方式の構成図

(1) 構成制御
ローカルのノード資源(CPU、メモリ等)の情報を定期的に収集する。

(2) 接続制御
ローカルのノードが保持しているセッションを管理する。また、クライアントからのセッション受付窓口となる。

(3) 同期制御
構成制御や接続制御の情報をノード間で交換する。また、セッションの提供に最適なノードを選定する。

(4) 仮想デスクトップ制御
ユーザ端末と仮想デスクトップのサーバ間のセッション(デスクトップセッション)を管理する。

(5) セッション永続化制御
セッションの監視を行ない、永続化・復帰を処理する。

(6) ユーザ制御・アプリケーション制御
ユーザ認証、AP アクセス制御、ライセンス管理をする。

5. おわりに

本稿では、新しい SBC アーキテクチャを提案した。また、プロトタイプの構築により、デスクトップセッションの仮想化、セッションの永続化・復帰、AP セッションのノード間分散、構成ノードの自動認識の各動作を確認した。

今後は、保守等のために特定ノード上の AP セッションを他ノードに移して処理を継続する機能の実現を行なう予定である。

参考文献

[1] Tristan Richardson, Quentin Stafford-Fraser, Kenneth R. Wood, Andy Hopper, “Virtual Network Computing”, IEEE Internet Computing, 1-2/1998.