解　説

PCサーバ・クラスタ—アベイラビリティ（可用性）向上をねらって—

PC Server Cluster - Focusing on Availability Cluster - by Ryoya MORI (Information & Communications Systems Lab.,Toshiba Corp.), Shigeru KOBAYASHI (Information & Communications Systems Lab.,Toshiba Corp.), Tetsuo KANEKO (Ome Works.,Toshiba Corp.) and Shuichi HARA (Information & Communications Systems Lab.,Toshiba Corp.).

森　良哉[1]　小林　茂[1]　金子　哲夫[2]　原　修一[1]

1 (株) 東芝情報・通信システム技術研究所
2 (株) 東芝青梅工場

1. はじめに

複数のコンピュータを連携させて，1つのシステムとして機能，運用する技術は「クラスタ」技術と呼ばれている．PCサーバ・クラスタとは，複数のPCサーバを連携させ，システムとして利用することをいう．

「クラスタ」は，1980年代前半にミッドレンジコンピュータを用いて行われ始めた．共有メモリや共有ディスクを専用ハードウェアで密結合したり，標準のネットワークで疎結合したもので，計算機複合体，コンピュータコンプレックスなどとも呼ばれていた．

1990年代になって，汎用大型機やミッドレンジコンピュータで構築されていた大規模企業情報システム，基幹業務システム，制御システムがオープンなUNIXサーバで構築されるようになった．この時期，処理量や可用性の面で大規模あるいは基幹業務システムに適用できるようにUNIXサーバのクラスタ技術の開発が進んだ．

最近，イントラネット，グループウェア，データウェアハウスといった新しい情報システムにおいてPCサーバが急速に普及してきたが，これらのシステムでも可用性が求められるようになってきている．加えて，UNIXサーバのクラスタでも構築されるようになった基幹システムに，さらにPCサーバを適用する試みも増えてきており，PCサーバ・クラスタが重要視されてきている．

PCサーバ・クラスタのねらいは次の3つに分けられる．もちろん，複数のねらいを兼ねている場合もある．

- パフォーマンス・スケーラビリティ（拡張性）

クラスタにPCサーバを追加していくことにより全体の処理量を増やしていく．

- ロードシェア（負荷分散）

クラスタ内のPCサーバ同士で負荷を分散させることにより，ピーク時の処理能力（応答性）を保つ．

- アベイラビリティ（可用性）

クラスタ内のPCサーバ同士で障害時にバックアップして，全体での処理の可用性を高める．

PCサーバ・クラスタでは，現時点では，アベイラビリティ・クラスタとよばれる可用性向上をねらったクラスタのみが，実用レベルまで確立した技術である．本稿では，以下アベイラビリティ・クラスタについて解説する．

2. アベイラビリティを向上させるクラスタシステムとは

2.1 アベイラビリティとクラスタとの関係

アベイラビリティ（可用性）とは，特定の時間区間にシステムが稼働している時間の割合をいう．システムとして考えた場合，次のように表される．

$$\text{アベイラビリティ} = \frac{\text{正しくサービスを提供する時間}}{\text{サービス提供が期待される時間}}$$

システム全体としてアベイラビリティを高める一般的な手法は「分散化」と「多重化（あるいは冗長化）」である．分散化は，システム機能をネットワーク上に分散させることにより，単一障害の影響をシステム全体に及ぼしにくくする．しかし，障害発生部分のサー

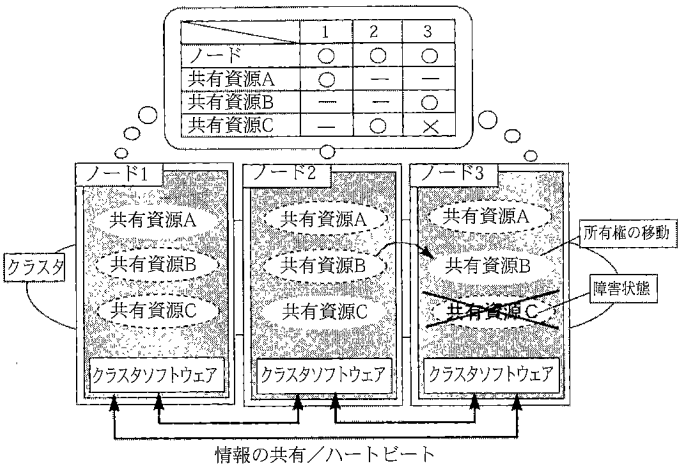


図-1　クラスタの基本動作

ビスは停止してしまう．そこで，サービスの提供に必要な機能要素を多重化することにより，さらにアベイラビリティを高めることができる．

分散化と多重化を兼ね備えた方法として，アベイラビリティ・クラスタがある．

2.2 アベイラビリティ・クラスタの基本動作

アベイラビリティ・クラスタでは，まずクラスタに含まれるべき各PCサーバ（以下ノードと呼ぶ）が相互に情報を交換しクラスタを形成する．クラスタ制御用のノード間通信路をインターコネクトと呼び，専用LAN，SCSI，あるいは機種固有のハードウェアなどを使用する．サービスを提供するパブリックLANを兼用する場合もある．ノード間の情報のやりとりは他のノードが正常に稼働しているかどうかの判断も含み，これをハートビートと呼ぶ．

クラスタが形成されると，各ノードはクラスタ内に存在する共有資源の管理を共同で行う．管理対象となる共有資源には，一般に（1）ネットワーク上の識別子（IPアドレスなど）（2）共有ディスク（3）業務処理（アプリケーションプログラム）などがある．

共有資源を管理するために，クラスタソフトウェア（以下CSWと略す）には資源に対する障害検出機能および各資源を有効化（組み込み），無効化（切り離し）する機能が必要となる．クラスタを構成する各ノードが，現在のそれぞれのノード状態や共有資源状態などのクラスタ状態について全ノードにわたって同じ情報を持つことによりノード内だけでなく，ノード間での共有資源の管理が可能となる．あるノードで障害が発

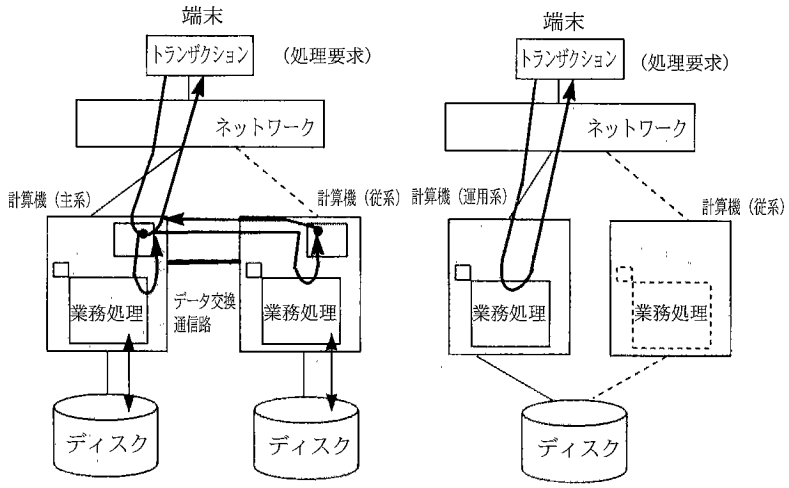


図-2 デュアルシステムとデュプレックスシステム

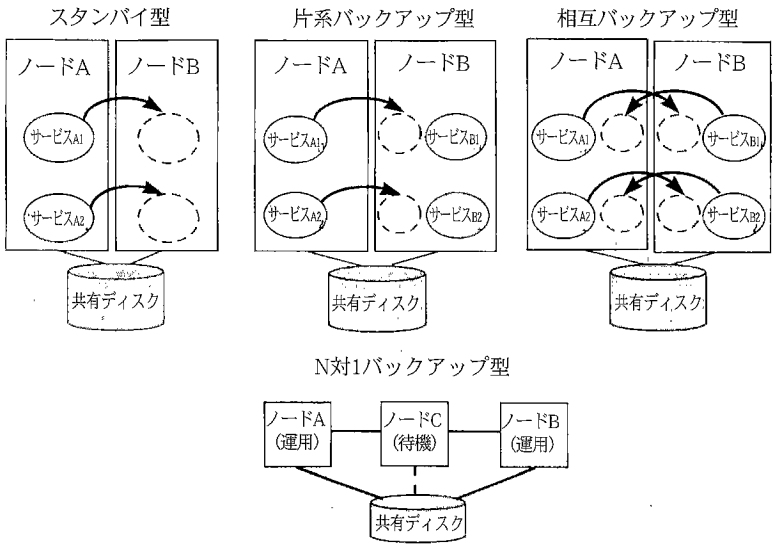


図-3 業務処理の引き継ぎ形態

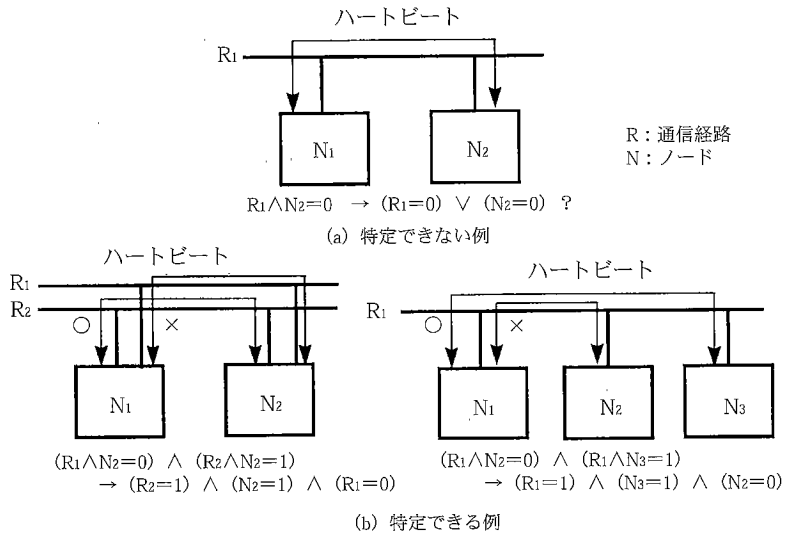


図-4 システム構成と故障個所の特定

生したときに，CSWは障害が発生したノードを安全に切り離し，他の正常なノードに共有資源の所有権を移動することでアベイラビリティを高める（図-1）．

2.3 システム形態について

複数のコンピュータを多重化してアベイラビリティを高めることは1960年代からアプリケーションソフトウェアでの作り込みで行われてきた．基本は2系列の処理系を組みにした，デュアルシステム型とデュプレックスシステム型である（図-2）．

現在のPCサーバ・クラスタのアベイラビリティ・クラスタの多くはデュプレックスシステム型である．デュプレックスシステム型では，正常時は運用系で処理をし，待機系は障害に備えて待機している．運用系に障害が発生したときには待機系に切り替えられる．待機系は正常時には，他の優先度の低い処理に用いることもでき，サーバの使用効率が良く，アプリケーションシステムの構築が容易でコスト効率も良い．ただし，切り替えのために，デュアルシステム型に比べて業務処理の中断時間が長くなる．

PCサーバ・クラスタではさまざまな形態が容易にとれることがのぞましい．すなわち，業務運用PCサーバ1台に対して安価なPCサーバを1台増やしてスタンバイ専用とする形態（スタンバイ型），数台の個別の業務を行うPCサーバ群に対して1台のバックアップノードを置く形態（N対1バックアップ型）などがある（図-3）．さらには別業務を行う数台のPCサーバが互いにバックアップできるようにしておき優先度の高い業務をバックアップし，優先度の低い業務は縮退する形態もある．

3. アベイラビリティ・クラスタの機能

CSWの基本機能は，クラスタの稼働状態を監視し，あるノードでの障害を検出したら，そのノードで実行していた業務処理を他のノードで再起動して引き継ぐことである．

本章では障害の検出方法，クラスタの運用状態の制御，および業務処理の引き継ぎに伴う共有資源の引き継ぎについて述べる．

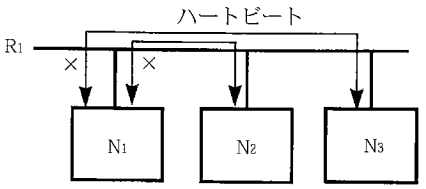
3.1 監視，障害検出

3.1.1 ノード内障害監視・検出機構

自ノード内の障害は，主にソフトウェアの状態として検出される．たとえば，プロセスが異常コードで終了する，あるいはプロセスが正常処理時間を過ぎても終了しない，などである．ディスクデータや周辺機器などの障害も，プロセスの実行結果を通して検出できる．

しかし多くの業務処理プロセスは，バックグラウンドで常駐して動作する．これを終了コードや時間のみで監視するのでは十分ではない．デッドロックや無限ループに陥っているなどの状態を検出できないからである．このような場合には，業務処理プロセスに付属してその動作の正否を判定するプロセスを作り，その結果をCSWが得るようにする．この方法は，簡単で比較的信頼性が高い検出法である．

PCサーバの機種によってはサーバのハードウェア異常，たとえば電源や温度の異常，ドライブの故障などをソフトウェアに通知するハードウェアを備えているものもある．その場合は，致命的な障害が発生して業務処理が止まる前に，より安全な手順を踏んでノード間引き継ぎを行うことが可能になる．



$(R_1 \wedge N_2=0) \wedge (R_1 \wedge N_3=0) \rightarrow (R_1=0) \vee ((N_2=0) \wedge (N_3=0))$?

図-5　故障個所を特定できない例

3.1.2　ノード間障害監視・検出機構

他ノードの障害やノード間通信経路の障害は，ハートビートによって検出される．通常，ハートビートはクラスタ内に含まれるノードのすべてのペアについて行われる．図-4は，ノードN1がハートビートによって特定できる故障情報の，システム構成による違いを示す．

個々のハートビートによって直接に判定できるのは，相手ノード自体と相手ノードに到る経路のいずれもが正常であるか否かということまでである．図-4（a）は，障害個所を特定できない場合の例を示している．この例において，ノードN1はハートビートによって障害を検出しても，通信経路R1かノードN2の少なくとも一方に障害があることしか分からない．

これに対して図-4（b）の例は，いずれもより詳細に障害個所を特定できる場合を示している．ノードN1は2つのハートビートを調べることによって，ノードと通信経路のうち，正常なものと障害を生じているものを区別できる．

冗長性の大きなクラスタシステムは障害を起こした要素の代替だけでなく，障害箇所を特定する精度を高め，的確な処置をも可能にする．

3.1.3　障害ノードの確実な停止

CSWの認識するクラスタシステムの状態から曖昧さを完全に取り除くことはできない．たとえばノード数3・通信経路数1の構成において，あるノードから見て他のいずれのノードからもハートビートの応答がない場合，経路に問題があるのか，たまたま両方のノードが障害を起こしたのか特定できない（図-5）．

この例のように通信経路を絶たれたノード同士は，互いの状態を明確に把握できなくなり，クラスタ全体の一貫性が失われる．この状態はスプリット・ブレインと呼ばれている．この状態に陥ると，複数のノードがいずれも自分が運用系であると判断して同一ディスクデータに同時書き込みを行うなど，重大な問題を生じる．

また，障害を起こしているノードは，通信経路がつながっていても正常ノードからの停止指示の受信やその処理を正しく行う保証がない．

障害状態と判断されたノードが誤動作を続けることを避ける典型的な方法は，業務処理の扱うディスクデータを，共有ディスク装置上に置くことである．業務処理を開始するノードは，それに先だってディスク装置を占有状態とし，データのチェック・修復を行うことによってデータ破壊を回避する．

また，専用ハードウェアでノード間を結び，異常であると判定されたノードを強制終了させる機能を持ったPCサーバもある．

3.2　クラスタの状態制御，構成制御

3.2.1　クラスタ全体の状態遷移

業務処理の引き継ぎ処理は，障害状態の他に業務処理の実行状態，システム管理者による操作情報にも依存する．これらの状態情報はクラスタ内のノード間で共有される必要がある．ノード間で状態情報を共有する簡単な方法は，各ノードが検出した事象を他のノードに対して配信するものである．その際，すべてのノードが，発生した事象と発生順序を共通に認識しなければならない．

事象発生順序の認識がノード間で食い違うことによる問題の例を図-6に示す．3ノードのクラスタにおいて，システム管理者がノードN1から，N3上で実行している業務処理Aの停止操作を行った直後にN3がダウンしたとする．このとき，事象の順序をN1が「停止操作→N3ダウン」と認識し，一方で，N2が「N3ダウン→停止操作」と認識したとする．すると，N2がダウン後の停止操作は無効であるとみなして業務処理Aの引き継ぎをすべきであると判断してしまい，クラスタ全体の状態に対するN1とN2の情報に矛盾を生じることになる．

ノード間の事象順序の認識を正しく行い，状態を遷移させる代表的な方法として，2フェーズコミットをベースとした放送プロトコルがある[1),2)]．

3.2.2　業務処理の引き継ぎ

障害ノードの業務処理を待機系ノードが引き継ぐことを「フェイルオーバ」といい，逆に障害復旧したノードに業務処理を引き戻すことを「フェイルバック」という．これらを完全に自動的に行うことによって，システムのアベイラビリティを高めることができる．しかし重要なデータを扱う基幹業務システムなどでは引き継ぎ時に人の手によるチェックを必須とする場合がある．システム管理者が操作によって業務処理をノード間で移動することを「スイッチオーバ」という．

3.2.3　ノードの切離しと組み込み

障害が発生していなくても，ノードをクラスタから切り離したり，組み込んだりすることが必要になる場合がある．たとえばノードの保守である．保守対象ノ

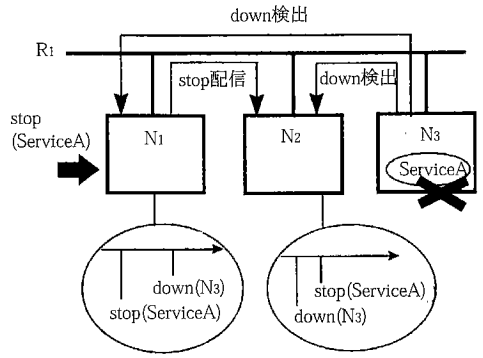


図-6　事象発生順序の認識が異なる例

ードを切り離すことによって，他のノードで障害が発生しても，保守中のノードに業務処理が引き継がれることを防ぐことができる．

3.3　共有資源の引き継ぎ

業務処理とともに引き継ぐべき代表的な共有資源には，ネットワーク識別子，データ，プロセスがある．

3.3.1　ネットワーク識別子の引き継ぎ

アベイラビリティ・クラスタで実行される業務処理には，ネットワークを経由してクライアントにサービスを提供するものが多い．このような業務処理では，ノードが処理を引き継いだときにクライアントが新しいノードにアクセスできるようにネットワーク識別子を業務処理ごとに割り当て，これを処理とともに引き継ぐ．

PCサーバでは，NetBIOS名の割当て・引き継ぎによってネットワーク識別子の引き継ぎを実現する場合が多い．この場合クライアント側のソフトウェアは，サーバに対するアクセスをNetBIOS名によって行うように作られていなければならない．最近では，OSの内部機能を使ってIPアドレスを直接引き継げるCSWもある．

3.3.2　データの引き継ぎ

データ引き継ぎの主要な対象は，ディスク上のデータ（データベースなど）である．ディスクデータの引き継ぎには，共有ディスク方式と，リモートミラー方式がある．

共有ディスク方式は，複数のノードに接続可能な（複数のSCSIポートを備えた）ディスクを用いたり，ノードとディスクをデイジーチェーン接続して，これを稼働ノードからアクセスする．**3.1.3**項で述べたように，誤って複数のノードからアクセスしないように処理中のノードはディスクを占有し，終了したら解放する．共有ディスク内のデータそのものの保全性を高めるため，共有ディスク装置としてはRAID構成を採ったものを用いる．

リモートミラー方式は，待機系ノードに，運用系ノードのデータのコピーを持つ．このデータは通常，業務処理にともない更新されるので，業務処理と並行してコピー処理も行う必要がある．コピーの方法には，変更の履歴をSQLなどの命令レベルで渡す方法や，変更されたブロックのデータをそのまま転送する方法などがある．データのコピーは，処理を引き継ぐノードすべてが持たなければならない．また，障害を起こしたノードが復旧したときには，障害中にバックアップノードで更新された内容を復旧ノードに反映する必要がある．

3.3.3　プロセスの引き継ぎ

プロセスや実行形式ファイルも，業務処理という抽象的な対象の使用する共有資源の1つと考えることができる．実際には，業務処理を構成するプロセスの実行形式ファイルは，その業務処理を引き継ぐ可能性のあるすべてのノードに，あらかじめインストールしておく必要がある．通常はそれぞれのノードのローカルディスクに実行形式ファイルを格納するが，共有ディスクに置くこともあり得る．

3.4　業務処理のクラスタ対応

アベイラビリティ・クラスタは，業務処理を停止させないのではなく，停止した業務を短い時間で復旧するものである．したがって各業務処理は，障害による停止と再開に対し，単体のノードで実行される場合と同様にデータの一貫性に対する注意が必要である．具体的には，適切な処理単位ごとにデータをディスクに反映すること，障害による業務処理停止によってディスクデータに不整合が生じる場合には処理の実行ステップやデータ復旧に必要な情報をディスク上に常時書き出しておくこと，そして業務処理は起動されたらまず以前の終了状況を調べてデータ復旧処理を行うことなどが求められる．障害からの復旧後もデータの整合性を保証する特性を「クラッシュ・トレランシ」という．

4．実システム適用への支援機能

アベイラビリティ・クラスタを実システムに適用するための支援機能について述べる．

4.1　さまざまなシステム形態への対応

2.3節で述べたように，業務処理の引き継ぎには，スタンバイ型，片系バックアップ型，相互バックアップ型のようにいくつかの形態がある．しかもこれは引き継ぎの方向のみによる分類であり，負荷の集中するバックアップノードでの縮退運転や，個々の業務処理の引き継ぎにおけるシステム管理者の介在の有無など，細かな違いまで考慮するとクラスタシステムの挙

動は非常に多様である．3ノード以上の構成ではとりわけ複雑になる．

システム形態の多様性に対する解として，最も典型的な形態（1つ，あるいは数種）を固定的にサポートするものと，アプリケーションシステムの開発者が動作をカスタマイズできる手段をサポートするものとがある．

4.2 システム構築支援，運用管理支援

アベイラビリティ・クラスタの設計における誤りのためにシステムの信頼性を低下させることがあってはならない．そのために，CSWとともにシステム構築支援機能が必要である．固定的な形態のみをサポートするCSWでは，設計はパラメータ設定を主とする比較的簡単な作業となる．カスタマイズ可能な場合には，それを誤りなく行うために，抽象的な表現からのコンパイラやデバッガなどの支援が必要になる．

システム稼働時には，システム管理者がシステム状態を的確にモニタリングし，また操作するために，運用管理支援機能も重要である．そのためのツールや，ツール作成のためのAPIが用意される場合もある．

5. おわりに

紙面の関係で，PCサーバ・クラスタのアベイラビリティ向上をねらったクラスタに絞って解説した．現在，実際のアプリケーション・システムに適用されているのはほとんどこのアベイラビリティ・クラスタである．OSレベルのクラスタプラットフォームAPIも，まずこの機能から提供され始めている．

パフォーマンス・スケーラビリティ，ロードシェアをねらったクラスタは，まずパラレルデータベースが，これから実システムでの評価を受け始めよう．技術的にはPCサーバをつなぐ高速なインターコネクトが，いくつか出そろってきている．また，OSレベルのAPIも，将来パフォーマンス・スケーラビリティ，ロードシェアに対応するであろう．

しかし，**4**章で述べたように実システムへの適用を増し普及させるには，アベイラビリティ・クラスタの場合よりさらに高度な負荷分散やクラスタの構成制御の面で支援するミドルウェアレベルのソフトウェアと，クラスタ上のアプリケーションソフトウェアやシステム開発を支援する開発環境が必要と考えられる．

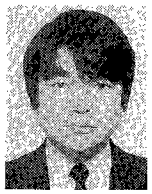
パフォーマンス・スケーラビリティ，ロードシェアの技術の研究・開発が加速されており，ここ1年位の間にはこれらの技術が確立するであろう．次の機会に解説したい．

参考文献
1）J. グレイ他，渡辺榮一編訳：フォールト・トレラント・システム，マグロウヒル出版（1986）．
2）Jalote, P.：FAULT TOLERANCE IN DISTRIBUTED SYSTEMS, PTR Prentice Hall（1994）．

（平成9年11月19日受付）

森　良哉（正会員）

1981年京都大学大学院工学研究科情報工学専攻修士課程修了．同年（株）東芝入社．以来，OS，UNIX仮想計算機，VLIW計算機基本ソフトウェア，高可用システム，分散ノード連携ソフトウェアの研究開発に従事．現在，東芝情報・通信システム技術研究所開発第三担当主査．IEEE会員．



小林　茂（正会員）

1982年北海道大学大学院工学研究科情報工学専攻修士課程修了．同年（株）東芝入社．以来，ミニコン・ワークステーションのシステム開発支援ソフトウェア，高可用システムの研究開発に従事．現在，東芝情報・通信システム技術研究所開発第三担当．

金子　哲夫（正会員）

1970年（株）東芝入社．以来，クラスタシステムをはじめ高信頼性システム，耐障害性システムの開発に従事．現在，東芝青梅工場ミドルウェア設計部ミドルウェア担当グループ長．

原　修一（正会員）

1980年神戸大学大学院工学部電子工学専攻修士課程修了．同年（株）東芝入社．以来，OAコンピュータ，並列・分散処理コンピュータ・アーキテクチャの研究開発に従事．現在，東芝情報・通信システム技術研究所開発第一担当主査．