オフコン／分散プロセッサと ワークステーションの機能分散

棚橋 賢一 （東芝 青梅工場 基本ソフトウェア第二部）

１．はじめに

最近のマイクロプロセッサ・高集積メモリをはじめとするＬＳＩ技術の進歩は、大量の低価格な処理システムを生み出し、システム方式として、ホスト集中処理方式から、各種の分散処理方式に発展してきた。

分散処理方式も、従来の端末接続に近いシステムから水平・垂直分散を組合わせた高度なシステムまで、非常に異なったレベルとなっている。例えばホスト対フロントエンドという主従関係の中での機能分散と、フロントエンド側主体の機能分散、各システム間の対等な関係を持つ水平分散ネットワークなど、システムを構成する各機器の処理能力に応じた機能分散が行われている。

一方、システムの使用者から見ると、従来の機能を中心としたニーズから操作性、応答性、使いやすさなどフロントエンド側のマンマシンインタフェースの充実が強く求められるようになってきている。これらのニーズもフロントエンド側の機能分散を強める大きな要因となっている。

前記のような動向の中で、パソコンやワークステーションの処理能力の高度化と他の処理システムとの分散方式は一つの重要な技術分野となっている。

以下では、これらの分散方式の特徴と具体的な実現例について紹介し、それぞれの方式について考察したい。

２．分散方式の分類

端末・パソコン・ワークステーションなどのフロントエンドを処理するシステムと、これらのシステムより上位の能力をもつ、オフコン・分散プロセッサ・ミニコン・汎用機などのホストシステムとの処理分散方式をフロントエンド側の機能により分類してみた。（図１）

このうち従来型の端末タイプについては論を待たないので省略し、その他の方式について次章以降に記す。

タイプ	基本ハードウェア構成	フロントエンド機能	代　表　例
A　端末	ディスプレイ／キーボード	文字グラフィック基本表示 物理属性制御 キーボード制御	VT－100、I－3270、T－4014
B　専用ワークステーション	ディスプレイ／キーボード	上記＋論理属性制御	J－5010
C　高機能ワークステーション	ディスプレイ／キーボード マウス、ファイルデバイス	上記＋拡張機能	J－5070
D　パソコン	同上	端末・ワークステーション エミュレーション、自己処理 ファイル転送	J－5030 パソピア1600
E　ネットワーク	同上	端末・ワークステーション エミュレーション、ファイル転送 ネットワークファイルアクセス	オフコン・分散プロセッサ （DPシリーズ、Qシリーズ）

図１　フロントエンドからみた分散方式の分類
ハードウェア構成については、通信モジュールは除いている。

３．専用ワークステーションでの機能分散

オフコンや分散プロセッサに接続されるＪ－５０１０は、効率良いマルチワークステーションシステム構築のため、応答性、機能面で強化されており、機能分散についても、端末タイプより高度な処理能力をもっているため、一歩進んだ形になっている。

３．１　本体との機能分散方式

本ワークステーションと本体との接続方式は、次の特徴をもっている。

(1) 本体とワークステーションの接続は、高速回線を利用し、通信手順としては、ＨＤＬＣ手順をベースにした方式をとって効率的なデータ伝送を可能としている。

(2) 前記プロトコル上に、論理チャネル方式を実現し、本体とワークステーション間で、複数のデータストリームを同時に交換できるようにしている。
ワークステーション内の制御構造として、マルチタスク構造をとり、次のような論理チャネルを同時に制御している。

① コントロールチャネル
本体からのファームウェアダウンラインロード／アップワードダンプなど最優先のチャネルである。

② モニタチャネル
本体ＯＳ系からのメッセージを扱うチャネルである。

③ ユーザチャネル
一般のアプリケーションプログラムやユーティリティ・サブシステム等のディスプレイ／キーボード入出力を処理する。

④ 漢字制御チャネル
外字・第二水準の表示時や、かな漢字変換等で本体ＯＳとパターンや候補文字をやりとりするチャネルである。本チャネルはモニタチャネル、ユーザチャネル使用時に優先的に制御される。

⑤ デバイスチャネル
プリンタやＯＣＲなどステーションにデバイスが接続されている場合に設定されるデバイス制御用の論理チャネルである。

上記、論理チャネルを経由して本体側ソフトウェアとの機能分散の一例が 図２ である。

これらの論理チャネルは一般に本体側ソフトウェアから起動する入出力チャネルとして制御されるが、ステーション側からトリガをかけるアテンション割込み制御も実現している。この双方向多重の論理チャネルを制御することにより、相互に必要な機能を呼出すことができる。

論理チャネル名	本体ソフトウェア	本　体　機　能	ステーション機能
コントロール	システム初期化タスク	ＲＡＭファームウェアのロード　セション設定	システム初期化
モニタ	モニタライン制御タスク	アテンション受付 メッセージ入出力	メッセージ表示入力
ユーザ	・ユーザタスク下のステーション入出力制御 ・画面オブジェクト作成ユーティリティ	・ステーション論理入出力 ・画面オブジェクト生成・出力	・入力制御 ・表示編集 ・画面オブジェクト処理 ・グラフィック表示
漢字制御	漢字処理タスク	辞書アクセス 候補送出	・不在漢字管理 ・候補選択 ・ローマ字変換 ・学習機能
デバイス	・ユーザタスク下のデバイス入出力	デバイス論理入出力	・物理デバイス制御 ・デバイス固有論理制御

図２　論理チャネルと機能分散

この方式を実現するため、それぞれのソフトウェア構造は階層化され各層ごとに対応するモジュール群と交信し、効率良い機能分散を図っている。このうちシステムの負荷分散として最も重要な技術は、ユーザチャネルにおける画面オブジェクトである。

画面オブジェクトは、本体側ユーティリティを利用してアプリケーションプログラムの一部として作成されるが、いったんステーションに送られると一画面の入力処理中には本体側の負荷はほとんど発生しない。画面オブジェクトの構成要素はフィールドレイアウト定義・フィールドの物理属性・論理属性・入力時の編集情報等がある。

このような制御情報群によりステーション操作時のデータ交信回数やデータ量を最小にできるため、システム性能や応答性を良くすることができる。

4．高機能ワークステーションでの機能分散

最近のOA・EA・FAなどのシステム利用の拡大につれて、さらに強力な処理能力をもつ高機能ワークステーションが登場してきた。これらは、高精細のビットマップディスプレイや高速のCPU及び専用ハードウェアを装備し、機能分散方式もさらに高度になってきている。

高機能ワークステーションの実施例として、J－5070を中心として、特徴及び機能分散方式を説明する。

4．1　高機能ワークステーションの機能

J－5070では、前記J－5010の機能に加え、次の拡張機能をもっている。

(1)　マルチウインドウ／マルチビューポート

論理画面と物理画面制御を分離し、その間をウインドウ／ビューポート変換という手法を用いて、複数の論理画面の全部、または、一部を同時に物理画面上に表示可能としている。

(2)　表示情報の拡張

文字・グラフィック・イメージの各種情報が表示でき、それぞれに拡大・縮小・回転などを行なえる。

(3)　イメージ処理

イメージ処理を高速化するための専用ハードウェアを装備し、イメージの圧縮・伸長（MH・MR・M2Rいずれも可）、切り出し、密度変換、拡大、縮小などの機能をもつ。

(4)　複合文書処理

文章・図形・イメージなどから構成される複雑な文書を作成するための統合ソフトウェアを実現するため、文書編集に必要な機能やアイコン・プルダウンメニュー・マウスポインテング・リアルタイムラバーバンド機能などのマンマシンインタフェース用の機能を実現している。

4．2　本体と高機能ワークステーションの機能分散方式

前記の拡張機能は、いずれも負荷が高いため、本体系での処理よりもワークステーション側での処理を中心にする方式が、システム性能を向上させるために必要である。図3 には、拡張機能について、本体とワークステーションの機能分担を記す。

拡張機能を実現するために、本体とワークステーショク間のプロトコルにも強化がされている。これらは、前記論理チャネル上のコマンド群の拡張で対応できる単機能のワークステーションへの指示と、

機　能	本体処理	ワークステーション
マルチウインドウ／マルチビューポート	論理画面入出力 ウインドウ／ビューポート設定	・論理画面制御 ・ビットマップディスプレイ制御 ・ウインドウ／ビューポート変換 ・オーバラップビューポート表示
表示情報の拡張	論理画面設定 表示コマンド・データ送出	論理画面制御 拡大・縮小・回転
イメージ処理	イメージ入出力 イメージ蓄積	圧縮・伸長・切り出し・拡大 縮小・回転・バッファリング
複合文書処理	文書保管、文書送出などのファイルサーバー	論理画面作成 編集 マンマシンインタフェース機能

図3　高機能ワークステーションと本体の機能分散例

複合的な機能指示に分けられる。後者のインタフェースとして、以下のような例がある。

(1) イメージ属性やウインドウ／ビューポート変換などの複雑な制御情報
(2) 文書転送時の論理的プロトコル（文書交換規約）
(3) コード・図形イメージ等で構成される統合文書データ構造
(4) 論理チャネルの拡張

5．パーソナルコンピュータ

ハードウェアの低価格化、流通ＯＳの浸透により、パーソナルコンピュータや多機能ワークステーションと呼ばれる大量のシステムが市場に出されている。これらのシステムとより上位のシステムの融合が今後のシステム技術の発展に重要な位置を占めている。

一般に実現されている上位システムとパソコンとの機能分散やインタフェースには次のものがある。

(1) パソコンによる端末・専用ワークステーションのエミュレーション

Ｉ－３２７０などの端末エミュレーションは、ほとんどのシステムで実現されている。また、専用ワークステーションなどのエミュレーションなども実現されていこともある。

(2) 本体系における端末エミュレーション

本方式は異なるシステムの端末を本体系に接続する時に実現される方法であり、端末側は一切の変更なしに、本体側のソフトウェア・ハードウェアにより、接続する端末の通信手順やデータストリーム変換を実現する手法である。オフコン・分散プロセッサなどの内部制御手順は、より高度な専用ワークステーションを前提とした機能分散を行なっている。（３項参照）。これらの方式をシステム全体の変更なしに、一部のハードウェアやソフトウェアにより、エミュレーションすることによって専用ワークステーションに近い形の機能分散を実現できる。

(3) ファイル転送

本体とパソコン・ワークステーション間で、データファイルを相互に交換する機能である。このシステムでは、一般には上位システムとフロントエンド側のコードやファイル構造の違いなどが問題となる。これらの変換をいずれのシステムで処理するかは、システム設計上重要な点である。
本機能は従来の異なるシステム間の通信システムと同等の機能分散であり、アプリケーションシステムによっては本体側データベースのリアルタイム検索なども実現可能である。

(4) ファイルサーバー

フロントエンド側には、一般に少量・低速のファイル装置しかもたないため、本体側の大容量め高速のファイル装置を有効に利用するためのシステムである。本システムでは、フロントエンド側のソフトウェアはローカルなファイル装置と同等のインタフェースを提供しており、ファイルサーバー側で、フロントエンドからのファイルアクセスに対し、ローカルファイル装置と同等機能を果たす。

上記の（１）（２）のエミュレーションでは、いずれもディスプレイ／キーボードを中心とした処理の機能分散であり、このレベルだけでみると端末接続における機能分散と同等である。またこれらのエミュレーションでは、ディスプレイのハードウェア仕様やキーボードの配置・刻印・キー数などの違いからマンマシンインタフェースや機能面で多少異なることもある。

この問題を抱えたとしても尚システム的メリットを出すためには、フロント側のプログラム実行やファイルアクセスなどの自システム内の処理系と本体の処理系との機能分担や融合が必要となる。これらの実例としてファイル転送・ファイルサーバーのほかに本体側でのクロス開発環境・プリントサーバーなどがある。

一般に本体側システムとフロント側システムは、システムアーキテクチャが異なっているので機能分散としては難しい問題が多い。この点については７項にまとめる。

6．ネットワーク

各種の機能分散方式のうち最も徹底した方式が、ネットワークを利用した水平分散システムである。本ネットワークの実例としてオフコン・分散処理プロセッサのネットワーク（TOPNET，DPNET）の特徴と機能分散方式について記す。

6．1　機能の概要

本ネットワークは以下の主要な機能を提供している。

(1) NSF（Network Station Facility）
ワークステーションをネットワーク内の任意のノードに接続し、他ノードの処理システムのすべてを自由にアクセスできる。

(2) NAF（Network File Access Facility）
アプリケーションプログラム及びユーティリティ等から任意の他ノードのファイルを自ノードのファイルとまったく同等のインタフェースでアクセスできる。

(3) NPF（Network Path through Facility）
他ノードのシステムが他系のシステムと通常のオンライン接続されている場合、そのノードを経由して他系システムと通信できる。

(4) NSU（Network Service Utility）
ネットワーク内での管理・運用を実現するユーティリティ群であり、ネットワーク内のファイル転送や構成変更等を行なう。

6．2　ネットワークソフトウェアの構造

本ネットワークの各ノードでの処理システムは、いずれも同一のアーキテクチャを採用しているため、システム的な機能分散も非常に優れており、かつ柔軟なシステムを構築できる。

各システムのソフトウェア構造は、階層化されたモジュール構造となっており、お互いに対応したモジュール間で交信することによって、効率的な機能分散がはかれる。以下には各機能利用時の両ノード間の機能分担及びモジュール構成を記す。

(1) 通信制御
OSI参照モデルにおけるトランスポート層以下を処理するモジュールであり、各ノード間に同一モジュールが存在する。本モジュールは各種の回線や通信制御手順に適用できるようにさらにモジュール化されている。サポートしている通信回線と通信速度は以下のようになっている。

①特定回線	～9600BPS
②公衆回線	～2400BPS
③DDXパケット網	～48KBPS
④高速構内回線	250KBPS
⑤TOTAL－LAN	10MBPS

このように多様な接続形態をとっているが、上位のプロトコルはまったく同一である。

(2) NSF
本機能はネットワーク内での他ノードのステーションエミュレーションにあたり、以下のパスを経由して実現されている。

① 利用者からの接続コマンド入力により、セッションが開始され、他ノードの仮想的なステーションとして論理的に接続される。
② 他ノード側の仮想ステーションに対する入出力はすべて論理的に接続された自ノードに渡される。その他の処理はすべて他ノード内のOSやに処理プログラムで実行される。
③ 自ノード側のNSFモジュールは他ノードから渡されたステーション入出力を代行し、結果を他ノードに返す。
④ 結果を受けた他ノード側はステーション入出力を発行した処理プログラムやシステムモジュールに結果を返す。
⑤ 利用者から切断コマンドが入力されるまで②③④の機能分散状態が継続される。切断コマンド入力時には論理的切断やセッション閉鎖が行なわれる。

(3) NAF
他ノードのファイルアクセス機能であり、処理プログラム側はノード指定以外は、自システム内のファイルアクセスと同等に処理でき

る。本機能における制御パスは以下のようになっている。

① 処理プログラムのリソース割付け時に、他ノードの指定があると、他ノードとセッションが開設され、ＮＡＦモジュールが起動される。
② 処理プログラムから該当のファイルに対する入出力要求が出されると、本要求は他ノードのＮＡＦモジュールに渡される。ＮＡＦモジュールは、他ノードのファイル入出力を代行し、要求ノードに結果が返される。
③ リソース解放時にセッションの閉鎖が行なわれる。

この間のノード間のデータ転送量を小さくするため、ＮＡＦモジュールではファイル共用制御、バッファ管理、エラー処理などを実行ノード内でのＯＳと連係をとりながら処理し、必要データのみが回線上を流れるようにしている。

上記処理例は、基本的な機能を示しているが、一つの処理プログラムが複数の他ノードのファイルを同時にアクセスしたり、前記ＮＳＦを組合わせて利用すると、操作は自ノード、処理プログラムの実行は他ノード、ファイルは自ノードなどという柔軟な運用も可能となる。

(4) ＮＰＦ

ＮＰＦは自ノード内の通信処理プログラムインタフェースと同一インタフェースで他ノードに接続されている他系システムとの通信を提供する。本機能では、ＮＡＦモジュールと同等位置付けで、他ノード中にＮＰＦモジュールが起動され、自ノードの処理プログラムの通信処理を代行する。

本機能及び複数の端末エミュレータ等を利用すると、一つのワークステーションが自システムには接続されていないいくつかの違うシステムの端末として機能することになる。

本機能により、他ノード、自ノード本体、自ノードワークステーションの四つの処理システムに渡る垂直・水平分散システムを構築できる。

7．機能分散の問題点と今後の動向

前項までに、種々の機能分散レベルを紹介してきたが、ここでその相違及び今後の問題点をまとめてみる。従来の端末接続に代表される主従関係をもった機能分散とフロントエンド側に自己完結の処理能力（プログラムの実行ファイルアクセスなど）をもつ機能分散とでは、大きく考え方が異なってくる。

前者はあくまで内部処理を効率化するために複数のシステムを利用しているので、アプリケーションシステムよりも基本システムを提供する側の責任によるところが大である。後者は、基本システムとしては道具を提供する形となり、実際の効率的な機能分散を実現するには、アプリケーション側のシステム設計に依存するところが多い。

また、後者の場合でも、機能分散をはかるシステムが同系のシステムの場合と異種システムの場合を比べると、現状ではアプリケーションへの負担がかなり異なる。同系システムの場合には６項のネットワーク実施例のように、密結合かつ、柔軟なシステムが構築しやすいが、異系システムの場合には数多くの問題が残されている。

異なるシステム、例えば、パソコンとオフコンあるいは汎用機をとってみるとハード、ソフト全般のシステムアークテクチャが一般にはまったく異なっている。 これらの機能分散と融合は、システム仕様全体にかかわる問題であり、簡単には解決できない。機能分散のレベルにしたがって解決すべき相違点を揚げると以下のようになる。

① 端末エミュレーション
- ディスプレイ・キーボードのハードウェア仕様
- コマンド・データストリーム

② ファイル転送
- コード
- データ形式

③ ファイルサーバー
- ファイルシステム
- データ管理機能

④ プリントサーバー
- コマンド体系
- コード

⑤ パススルー機能
- 通信機能

⑥　互換言語・互換サブシステム
⑦　ＯＳ機能エミュレーション
　　（バーチャルマシン、マルチＯＳサポート）

一方トータルシステムとして、機能分散を効率的に行なうには、特に回線性能と相互のシステムの負荷状況及び、処理能力を十分に評価する必要がある。

今後は回線能力、フロントエンド側の処理能力もますます強くなる傾向にあり、機能分散とシステム間の融合もしやすくなってきている。従来は比較的処理能力の高い本体側で、変換や実際の処理が行なわれていたが、今後は、ニューメディアの普及と合せてフロントエンド側主体に、たとえば、どの本体ともコミュニケートできるようなワークステーションの台頭が予想される。パススルー機能はその一つの実施例であるが、このエミュレーションが端末機能レベルから、それ以上の機能分散に向かうことと考えられる。

インテリジェンスをもったシステム間の結合は、異文化をもつ民族間の交流にも似た難しさを含んでいるが、システム技術の発展と標準化の浸透により、着実に解決されるであろう。