# 6Q−6　ホスト・ワークステーション連携処理における共通通信インタフェースの適用効果

斎藤　博史，半澤　正樹，○大八木　誠
富士通株式会社

## 1. はじめに

高機能ワークステーション（以下ＷＳと略す）の普及に伴い，ホストコンピュータ（以下ホストと略す）との連携においてもＷＳ機能を活かしたきめ細かな連携処理が望まれている．その一つに，ホストとＷＳそれぞれの利用者プログラム（以下プログラムと略す）の間でデータをやりとりするプログラム間通信機能を用いたシステム構築がある．

本稿では，富士通におけるホスト・ＷＳ間のプログラム間通信機能の進展状況を踏まえ，システム統合アーキテクチャSIA(Systems Integration Architecture)の共通通信インタフェースとして提供している分散プログラム間通信機能（以下DPCF:Distributed Program to program Communication Facilityと略す）の概要と適用効果について述べる．

## 2. ホスト・ＷＳ間のプログラム間通信機能の進展

### 2.1 『アプリケーション会話機能』

ホスト・ＷＳ間のプログラム間通信機能として最初に提供したのが『アプリケーション会話機能』である．これは回線手順を意識した通信システムであり，端末エミュレータとの回線共用ができず，専用の業務分野で利用されてきた．

### 2.2 『会話型データ転送機能』と課題

1985年にホスト・ＷＳ連携システム『LINKDUET』を提供し，その中で汎用的な通信機能として『会話型データ転送機能』を実現した．通信手段として端末エミュレータ機能をベースとしているため，利用者は回線手順に依存しないシステムを構築できる．

## 3. 『DPCF』の概要

### 3.1 プログラムインタフェースの進展

さらに汎用化を進めるために，通信処理を行うプログラムインタフェースの整備を行った．LINKDUETからDPCFへと表-1に示すように進展している．

LINKDUETの初期の段階では，ホスト側は転送データの先頭に特殊な制御情報を付加し，それをCOBOLのREAD/WRITE命令によってレコード処理として扱い，ＷＳ側はCALL命令によってサブルーチンを呼び出し，パラメタ処理として扱っていた．その後，ホスト側は制御情報をレコードから分離することにより，一般のファイル入出力と同様に処理することを可能とした．これは「表示ファイル」と呼ぶ仮想ファイルへの入出力処理で共通化し，実際にデータを送受信する相手（プログラム／ディスプレイ／プリンタ）に関わらず，全て共通のREAD/WRITE命令による入出力インタフェースとして統一し拡張したことによる．

表-1　プログラムインタフェースの進展

| | LINKDUET | | DPCF | |
|---|---|---|---|---|
| | STEP1 | STEP2 | STEP3 | STEP4 |
| ホスト | READ/WRITE 特殊レコード | READ/WRITE 一般レコード | READ/WRITE 一般レコード | READ/WRITE 一般レコード |
| WS | CALL パラメタ | CALL パラメタ | CALL パラメタ | READ/WRITE 一般レコード |

さらに，プログラム間通信の機能強化を行い，会話型データ転送機能における以下の機能制限を解除し，『DPCF』として製品化した．

- ●データ転送量　最大４KB／回⇨最大32KB／回
- ●送信・受信がペア⇨連続送信，連続受信可能
- ●通信相手は１つ⇨同時に複数相手との通信可能

1990年２月にＷＳ側にもホスト側と同一のインタフェースを提供し，両者共にREAD/WRITEの単純でかつ共通の入出力命令による通信が可能となった．

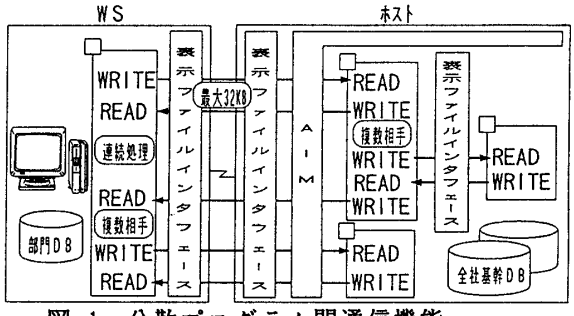


図-1　分散プログラム間通信機能

---

An effect of application system using common communication interface in Micro-Mainframe Link
Hiroshi Saito , Masaki Hanzawa , Makoto Ohyagi
FUJITSU LIMITED

3.2　DPCFの適用

ホスト単体で動作している業務システムに対しての主な機能要望を示す.

①HCI(Human Computer Interface) の優れた端末を使いたい.

②入力データのチェックのような本来の処理ではない部分では高速かつ一定レスポンスが欲しい.

これらの要望を実現するために，ホストとＷＳを連携した分散処理システムの構築が必要となる.

DPCFは，既存のプログラムインタフェースの拡張で実現したことにより，分散処理用に新たにシステム設計を行う必要はない．既存システムに適用することにより，分散処理型に変更することができる.

実際に，評価システムとしてホスト上で動作しているオンライントランザクション処理の一部を図-2に示すように分散型に変更した．この結果，HCI の向上や入力データチェック処理のローカル処理化が実現できた.

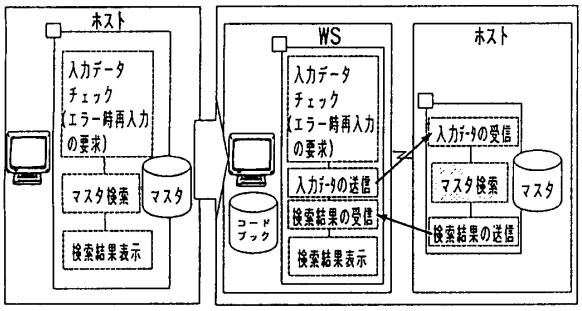


図-2　ホスト単体処理の分散型への再構築例

ホスト上のプログラムの「画面入出力」と「マスタ検索」という二つの処理部を分離し，前者をＷＳ側にオフロードし，ＷＳ側・ホスト側それぞれに通信処理を追加すればよい．この時も特別な処理を追加するのではなく，ＷＳ側ではマスタ検索のREAD命令をホストからのデータ受信のREAD命令とし，ホスト側では画面からのデータ入力／検索結果表示のREAD/WRITE命令をＷＳからのデータ受信／送信のREAD/WRITE命令に変更するだけで実現できた.

さらに，ＷＳ側についてはHCI 向上とホスト負荷軽減のために以下の処理を追加した.

①HCI 向上⇨・マウスによる選択操作の実現.
　　・背景色，強調文字の使用.
　　・メッセージ出力のサブウィンドウ化.

②負荷軽減⇨・入力データの正当性チェック

ホスト側はＷＳから送信されてくる「チェック済の正しいデータ」を元にマスタを検索するだけの処理となりプログラムを単純化することもできた.

このようにプログラムを機能単位に分割し，機能を実現するのに最も適したコンピュータに振り分け，その間をプログラム間通信機能で繋ぐことにより分散型にすることが可能となった.

3.3　DPCFの適用効果

DPCFの適用により分散システムを構築した場合の効果を評価結果をもとに述べる.

まず第一に，プログラム間通信機能を共通通信インタフェースにより実現することで，システム構築が容易となった．即ち，ホスト・ＷＳ共に共通言語(COBOL) の共通命令(READ/WRITE)で通信処理が可能であり，しかも画面入出力処理と同様に扱うことができるため，特別な知識習得は不要であった.

図-2に示した分散型への再構築の場合，通信処理部については前述のように僅かな追加・変更だけであり，従来の会話型データ転送のように全てを追加記述する場合と比較すると生産性が向上した.

第二に，機能分担によりコンピュータ資源の有効利用が可能となった．例えば，図-2のようなオンラインシステムでは入力データの正当性チェックが処理の大半を占め，さらにホスト集中で行うためにレスポンスの悪化やホストCPU の負荷増大となる.

そこで，チェック処理やエラー表示，入力すべきデータを示すコードブックの表示といった処理についてはＷＳ側で行うことによって，ホストに依存せず一定の高速レスポンスが得られた.

第三に，利用者からみて使い勝手の良いシステム構築が可能となった．ＷＳ側の豊富な入力機能（マウス・音声等）及び出力機能（マルチウィンドウ，マルチメディア）の利用により，親しみやすい優れたHCI が実現できた.

4.　おわりに

プログラム間通信機能は，今後さらに利用されるケースが増えると考えられる．このような分散システム構築のための基盤技術の整備・充実について，今後も継続的に努力していく所存である.

5.　参考文献

・太田昭一：『情報処理学会第33回（昭和61年後期）全国大会 2T-2 ホスト～パソコン連携　アプリケーション会話処理の応用システム』

・『富士通　SIA 解説書』

・『FACOM OSⅣ AIM/DPCF 使用手引書』