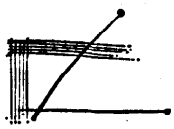
展　望



計算機複合体のオペレーティングシステム†

田中哲男†† 前川　守†††

1. 序　論

1970年代における計算機アーキテクチャ研究の一つの大きな潮流として計算機複合体の研究を挙げることができる．70年代においてこの種の研究が盛んになった最大の理由は，1万ドル計算機といわれる安価なミニコンピュータが登場したことによる．現在ではLSI技術の進展により数千円，数百円のマイクロプロセッサの入手が可能であり，結合のためのハードウェアが依然として高価であるとはいえ，80年代においてもやはり計算機複合体の研究，実用化が推進されていくことになろう．

計算機複合体のアーキテクチャについては，すでに情報処理誌上において詳細な解説[1),24)]がおこなわれている．更に分散処理の観点からは，高平によるオペレーティングシステムの課題についての解説[35)]がある．本稿では計算機複合体のオペレーティングシステムについて，70年代の成果をふり返り，残された問題は何か，ということを論じる．なおここで取扱う計算機複合体とは元岡の定義[24)]のうち，マルチプロセッサシステムおよび狭義のコンピュータコンプレクス（比較的近距離に散在する計算機を，そのシステム独自の通信手段で結合したシステム）をいう．

第2章では，計算機複合体に寄せられる期待を分析し，第3,4,5章ではシステムの分割と統合，プロセス間通信，信頼性と再構成について論じる．計算機複合体の研究開発の例として，ACE[11)]，PPS[29)]，KOCOS[16)]，PPS-1[25)]，EPOS[21),22)]，MICS-II[26)]，TMCS[30)]，C.mmp[39)]，Cm*[34)]，HXDP[13)] などがあげられるが，第6章でいくつかのシステムを紹介する．

† Operating Systems for Computer Complex by Akio TANAKA (Information Systems Lab., Toshiba Research and Development Center, Tokyo Shibaura Electric Co., Ltd.) and Mamoru MAEKAWA (Department of Information Science, Faculty of Science, University of Tokyo).

†† 東京芝浦電気(株)総合研究所情報システム研究所
††† 東京大学理学部情報科学科

2. 計算機複合体に対する期待

計算機複合体の最大の特徴は，プロセッサが複数個存在することである．このことは処理の段階において何らかの分散がおこなわれることを意味する．分散される対象の主なものとして，次のものが考えられる．

①負荷
②機能
③データ
④制御

この分散の種類，度合により計算機複合体の能力は大きく左右される．C.mmp に代表されるマルチプロセッサシステムは，負荷の分散だけを可能にするものと考えられるし，EPOS のようなポリプロセッサシステムでは，負荷および機能の分散を可能とする．データ，制御の一方あるいは両方を分散するには，より高度の技術を必要とし，現時点においてはこれらを完全に分散して成功したシステムは存在しないと思われる．その問題点については次章で述べる．

さて，計算機複合体に寄せられる期待は次の3項目にまとめることができよう．すなわち，①高性能および高価格性能比 (high performance and high cost/performance)，②高信頼性 (high reliability)，③高適応性(high adaptability) である．この章ではこれらの期待を検討し，そこから複合体オペレーティングシステムに対して，どのような課題がひき出されるかを考える．

2.1 高性能および高価格性能比

この期待は，複合体は複数個の処理要素で構成されているのだから，それらを総計した処理能力は，1台の処理装置で構成される一般の計算機システムより大きくなるであろうということと，個々の処理要素を最近の低価格高性能のミニ・コンピュータあるいはマイクロ・コンピュータで構成することにより，高価格性能比を得ようとするものである．この期待に応えるためには，オペレーティングシステムは，まずプログラ

ム可能な環境を提供しなければならない．すなわち利用者が応用システムを記述するためのプログラミングシステムである．このための問題として，①記述言語を提供すること，②記述された応用システムを並列に動作可能な単位（プロセス）に分割すること，③分割されたプロセスが協同作業をおこなうための通信手段（プロセス間通信）を提供すること，④プロセス群を能率よく動作させること，などがあげられる．

計算機システムの性能を向上させる一つの方法としてマイクロプログラムの活用を考えることができる．計算機複合体では，ハードウェア的には同一の安価なプロセッサを，マイクロプログラムによって専用機械化して高速化を図り，それらに機能を分散し，専用機械の集合体として価格性能比の良いシステムを構築できよう．EPOSはこの専用機械化をおこない，更にマイクロプログラムの動的書換えによって負荷の変動に対応しようとしているシステムの例である．

2.2 高信頼性

計算機複合体の高信頼性に対する期待は，次の2点に要約できよう．すなわち，①構成の冗長性により1部の故障に対して残った処理要素が仕事の肩替りをすることにより，全体として正しく動作できる，②処理要素が複数個存在することから，必然的にモジュール化が促され，また従来ソフトウェアだけではオーバヘッド等のために事実上不可能であったエラー検出（例えばcapabilityによる保護機能）等が可能になり，それによって信頼度を高めることが可能となる，の2点である．

ところで信頼できるシステムとは，仕様通り動作するシステムのことである[27]．計算機システムが仕様通り動作しない原因として，①ハードウェアが仕様通り動作しない（例えば記憶装置がパリティエラーを起す），②オペレーティングシステムに虫があって仕様通り動作しない，③利用者のソフトウェアに虫があって，それが他の利用者あるいはオペレーティングシステムの誤動作をもたらす，の3点を挙げることができよう．③については，それが他の要素に影響を及ぼす前に検出することができれば，問題は解決するであろう．②についてもオペレーティングシステムのできるだけ多くを，利用者プログラムと同じレベルで動作させることによってある程度までは検出可能であろう[6]．しかし，検出したということだけでは，仕様を満足するような動作を回復することはできない．故障の原因，および検出された時点までにその故障が及ぼした影響の範囲が局限されなければならない．計算機複合体でのモジュール化は，この局所化を助ける役目を果たすであろう．なお，故障からの回復の手法として，システムの状態を，故障が検出される以前のCheck pointで保存した状態に戻すバックアップがある．計算機複合体上でのプロセス間の関係として重要なものの一つである生産者—消費者のモデルにこれを適用しようとするとドミノ効果[27]と名ずけられるように，結局は一番始めの通信から始めなければならなくなることがある．

Russellはバックアップの回数を少なくするようなモデルを提案している[28]．信頼性に対する要請を少し低いレベルにとどめれば，計算機複合体が寄与できる分野がある．故障が検出された段階で，それを含むハードウェアモジュールをシステムから切り離し（このためのシステムの運転を一時停止するとしても），残りの部分でシステム全体としての動作を続けることができる．この問題は，高適応性の項で論じる．

2.3 高適応性

高適応性に対する要請は次の二つに分けることができよう．一つは静的な適応であり，もう一つは比較的短い時間のうちにおこなわれる動的な適応である．静的な適応とは，計算機システム導入後の業務の増大に対して，単に計算機複合体の構成要素を追加するだけで，あるいは構成要素を機能的にまたは性能的により高性能な要素で置き換えることによって適応しようとすることである．ポリプロセッサシステムEPOSのTSSは，あらかじめTSSを構成する各機能を分割し，それぞれ複数個のモジュールによって実現することが可能な設計がなされており，開発も小さな構成のハードウェア上でおこなわれた[22]．動的な適応には二つの種類がある．一つは故障発生時に，故障を起したモジュールを切り離し，直ちに残されたモジュールだけで立ち上り，システムの動作を再開する方法である[31]．Hydraの場合について，第5章で紹介する．他の一つはシステムに対する負荷に応じて，複合体の構成要素の役割を動的に変更するものである．EPOSにおけるファームウェアの動的書換えによる，専用機械の役割の変更がこれに対応する．このようなさまざまな適応を可能にするためには，システムの個々の機能の利用がそれぞれの機能が存在する場所を陽に指定することによっておこなわれるのではなく，その機能の名前を指定することによって利用できるように設計されていなければならない[8]．

3. 分割と統合

3.1 Jensen のモデル[12]

計算機複合体上で動作するプログラムは，相互に協調しながら並列に動作する．このときの大きな問題は分割 (partitioning) と分割されたものをプロセッサに割り付けること (assignment) である．計算機複合体の設計者は，①誰が，いつ，どのような基準で分割，割りつけをおこなうのか，②分割はどの程度まで細かくおこなうのか，③割りつけはどれだけの間存続するのかということを定めなければならない．

一方システムの中で，この分割と割り付けを担当する者として，以下のようなものがあげられる．ⓐプログラマ，ⓑトランスレータ，ⓒ配置者 (configurator) ―分割の最終責任者，ⓓ割り付け者 (assigner)― プロセスをプロセッサに割り付ける者，ⓔディスパッチャ ―ある一つのプロセッサに割りあてられた複数個のプロセスを一つの制御の流れに変換する者，ⓕプロセッサ―一つの制御の流れを受取って，それを動作 (action) あるいは結果 (result) に変換する者．分割，割り付け，処理のそれぞれの過程を，どの担当者がおこなうかを示したものが図-1である．

計算機複合体のソフトウェアは，ほとんど図の左側の流れで取り扱われ，プログラマ，配置者を人間が担当している．一方並列演算機械と称されるシステム，例えば ILLIAC IV, Data Flow Machine[7] などは右側の流れで取扱われる．

計算機複合体のソフトウェアを特徴づけるものは，

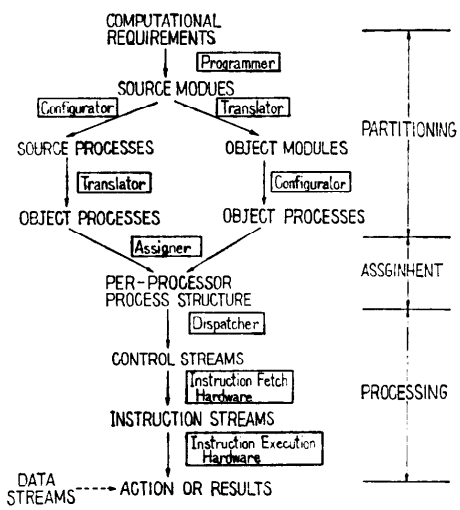


図-1 Jensen のモデル[12]

ユニプロセッサシステムでは存在しなかった配置者と割り付け者の存在である．図-1の流れの中で①～③の問題が個々の複合体の目的に合う形で解答されなければならない．この分割の問題は，現在計算機複合体を実用化する上で最大の問題であり，分割があらかじめ簡単におこなわれる場合に限って実際への応用が可能となっている．しかし，その応用分野は必ずしも小さいものではなく，メッセージ処理を中心とするシステムたとえば多くの実時間処理システム，座席予約システム，銀行システム，タイムシェアリングシステム，等は十分その応用分野となりうる．

3.2 統　合

前節ではシステムに課せられる仕事を分割することについて述べたがそれらは全体としての目的に沿うよう統合された形で，処理がおこなわれなければならない．分散される対象は，この負荷の他に前章で述べたように，機能，データ，制御がある．これらはやはり全体として統合された形で処理がおこなわれなければならない．

（1） 負荷の分散と統合

負荷の分散と統合は，従来のマルチプロセッサシステムでおこなわれてきたものであり，必要な機能は，同期とメッセージの交信である．負荷の分散が小さな仕事の単位でおこなわれる場合には，特にこれらの操作のためのオーバヘッドが問題となる．プロセス間の通信の問題は第4章で述べる．

（2） 機能の分散と統合

機能分散はしばしば話題に上るが，その研究は特に日本において盛んである．機能分散を実現しているシステムとして EPOS[21),22)] がある．また，PPS[29)] もこの範ちゅうにはいる．機能分散の前提として専用プロセッサ化の手法として，最も広く用いられているのは，ダイナミックマイクロプログラミングである．ハードウェア自身を専用化することは，他の面，特に信頼性，価格の面から不適であろう．信頼性その他を犠牲にせずに専用化を期待できるレベルとしては，マイクロプログラムレベルが最適であろう．EPOS ではオペレーティングシステムの各機能および言語プロセッサの機能が専用化されている．特に APL については高度の専用化がおこなわれている[23)]．機能分散の場合のプロセス間の同期，メッセージのやりとり等は，負荷分散の場合と同様である．

（3） データおよび制御の分散と統合

計算機複合体では，機能分散や負荷分散をおこな

い，さらには，ローカルメモリを設けて処理の局所化を図るが，共通情報や全体の制御は依然として共通なものとして残る．これらを完全に分散できれば(例えば，共通データに対しては，特定のプロセッサの管理の下におかずすべてのプロセッサの管理下におき，かつデータの統一性を保つこと)，システムの自由度，信頼性を飛躍的に増すことが期待される．このためのデータおよび制御の分散について盛んに研究がおこなわれている[4),18),19)]．しかしこれらはまだ基礎研究の域を出ておらず，実用化にはもう一段の発展が必要である．

現在のところデータまたは制御の分散を行って動作しているシステムはみあたらない．

4. プロセス間通信

プロセス間の同期，通信の問題の重要性は古くから認識されているからここで再論することはやめる．また通信のための命令の方式等も，3.2(2) で述べた部分を除いては通常のオペレーティング・システムと同じなので省略する．この章では，複合体に関連すると思われる四つの問題をとりあげて紹介する[41)]．

(1) 複数個のサーバ

メッセージを送信することの目的は，受け手（サーバ）に仕事を依頼することである．その仕事の処理がシステムの隘路になる場合には，サーバの数を増してやる必要がある．メッセージの送信がサーバ名を指定するものであれば，サーバを増すのは困難になる．一つの解は集信プロセスを一つだけ用意し，このプロセスが，スケジューリングをおこなって，各サーバへ配信すればよい．メッセージの送信が一つのメッセージバッファに対しておこなわれ，すいているサーバが自分でメッセージバッファから取り出すのであれば，この問題は解けていることになる．放送 (Broadcast) による方式は，必ず一つのサーバがその仕事を処理してくれるという保障がなければならない．

(2) 保 護

仕事を依頼するということは，メッセージの中にある引数を入れておくということを意味する．このために，保護に関連して二つの問題が生じる．一つは，悪意のあるサーバが，あるいは誤動作によって，依頼した仕事以外のことまで，その引数に対して行うかもしれないということである．もう一つは，悪意のあるユーザプロセスが，メッセージを偽造し，自分が権利をもっていない対象に対する仕事を依頼するかもしれないということである．この二つの問題を解決する方法として，完全な，capability による保護機構を導入し，引数とともに，その引数に対する capability を送信する方法がある．capability による保護機構が完全であれば，ユーザプロセスは，自分がもっている capability だけしか送信できないし，サーバプロセスは，受け取った capability の範囲内でしかその対象を操作できない．

(3) 2方向通信

ユーザプロセスは，自分が送出したメッセージに対する応答を取け取ることができなければならない．応答の目的は二つある．一つはサーバからの受理通知または処理終了通知であり，もう一つは，サーバが存在しない場合あるいは，サーバによる処理が一定時間内に終了しない（サーバの誤動作によって，例えばサーバが入出力機器である場合，あるいはサーバのソフトウェアの虫による場合）ことを通知するためのものである．後者の場合には別の通信経路によってこれを通知するという方法はとれない．通信の管理をおこなう核は，応答のための通信経路の存在を知らないからである．

(4) ランデブーサイトの問題[38)]

共有メモリによる通信をおこなわない場合に，この問題が生じる．送信と受信の同期を送信サイトまたは受信サイトのどちらの側で確立するべきかという問題である．信頼性の観点からいえば，ランデブーサイトは送信側であることが原則である．すなわちメッセージは送信側のバッファに止めおき，それに対する受取り要求が発生したときに同期が確立し，メッセージの転送がおこなわれる．ランデブーサイトが受信側にあり，メッセージが受信側のバッファにおかれている場合に，受信側においてエラーが発生すると，送信側が既に送り出したメッセージを回復して代替サーバに送ることは困難である．一方，送出側にランデブーサイトがあり，送出側でエラーが検出されたときには，受信側がそのメッセージを処理する必要はないのだから，メッセージが失われてもよい．

5. 信頼性と再構成

計算機複合体は，信頼性の向上に寄与しうるであろうか？ この問題を検討するためにまず信頼性を確保するための三つの条件をのべ，しかるのち複合体が寄与し得る点について考えてみたい．

5.1 Wulf のモデル

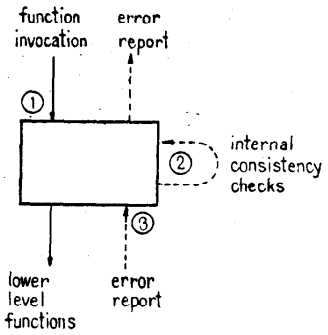


図-2 Wulf のモデル
Module Reliability Responsibilities[42]

ハードウェア，ソフトウェアを含めて計算機システムは多くのモジュールで成り立っている．モジュールはそれぞれ何らかの抽象データすなわち，データの表現とそれに対する操作を実現したものであると考えられる．Wulf は，一つのモジュールの働きを**図-2**のようにモデル化し，信頼性を高めるためにそれが備えるべき機能を明らかにした[42]．実線の矢印はそのモジュールに含まれる操作の呼び出しである．これは更に他のモジュールに属する操作の呼び出しによって実現されるであろう．点線で示された矢印は，誤りを検出し，それを修正するための動作が三つの点において必要であることを示している．第1はパラメータの誤りの検出とその通知，第2はそのモジュール自身が管理しているデータ構造の正しさの検査あるいは修正であり，第3は，自分が呼び出した操作からの誤り通知に対する処置である．例えば記憶装置におけるエラーを考えると，エラーを検出した記憶装置は，エラー訂正機構を備えていれば，故障箇所を修正する．2ビット以上の誤りがあって修理できなければ，自分を呼び出したモジュールに対してエラーを通知するであろう．

このように信頼性を確保するためには，三つの条件が必要である．第1はある操作を実行する前に，その操作要求の正しさを検証がおこなわれること，第2にある操作を実現するもの自身の内部の無矛盾性が確認され，矛盾が訂正可能であれば，修復がおこなわれること，第3に故障の通知がおこなわれることである．

5.2 計算機複合体による故障検出と再構成

複合体であることによって信頼性の向上に寄与し得る点があるとすれば，それは複合体のモジュール性と相互の監視が可能であることに依る．

モジュール性とは，物理的な処理要素が複数個あって，相互のインタフェースが明確に定義されているということである．これによって第1に，ソフトウェアのモジュール化が強制され，第2にモジュール間の不用意な干渉を防ぐことができる．

第2の相互監視による故障検出は一つのプロセッサ／メモリ対によるエラー検出，修復機能が十分働かず，破局的な情況になったときに有効である．もし，相互監視による検出の前に，故障したプロセッサがシステムデータを破壊していれば修理は不可能に近く，再構成が必要になる場合が多い．しかし信頼性の向上という言葉の意味を広くとらえて，故障によるシステム停止を短い時間に限る，と解釈すれば十分に有効な方法になり得る．以下に C. mmp における被疑者／監視者モデル[31]を説明する．なお Pluribus[17] においても相互監視が実現されている．

被疑者／監視者のメカニズムは，あるプロセッサ上で異常割り込みが起ったとき，そのプロセッサが被疑者となって動作を開始する．あるいは一方のプロセッサが番犬 (watchdog) ルーチンによって他プロセッサの異常を発見して起動する．後者の場合には，発見したプロセッサが監視者となり，相手に対して被疑者であることを通知する．被疑者と監視者は決められた処理系列を同期を取りながら動作する．この過程の中で被疑者は，①故障が発生したときのプロセッサの状態を保存し，②自分のローカルメモリの内容をコピーし，③短い診断プログラムを実行し故障の解析をおこなう．監視者は，被疑者の応答を監視し，応答が得られない場合には，再ロードをおこなう．すべての応答が得られた場合には，監視者はその応答の内容によって①変更なしに動作を続けさせる，②被疑者を停止させておいてシステムの運用を継続する，③システムの再ロードをおこない被疑者をシステムの構成からはずす，などの処置の選択をおこなう．

6. 複合体オペレーティングシステムの例

ここでは Hydra, StarOS, EPOS, HXDP をとりあげて紹介する．Hydra は共有メモリに依るマルチプロセッサシステムのオペレーティング・システムであり，StarOS は，更に大規模なマルチプロセッサシステムの例である．EPOS は専用機械で構成される機能分散型複合体の例である．HXDP は共有メモリをもたない分散処理システムである．

6.1 Hydra[41]

Hydra は C. mmp[39] のためのオペレーティングシステムの核である．C. mmp については，情報処理誌

上ですでに紹介されている[10),32)]ので詳論しないが，32 MB の共有メモリに 16 台の PDP 11 を接続した，典型的なマルチプロセッサシステムである．

Hydra の目標は，複合体オペレーティングシステムの研究をするための基本的な核を提供することにあり，以下のような特徴をもっている．①政策 (Policy) と機構の分離：プロセッササイクル，メモリページといったシステム資源を，どのプロセスにどれだけ与えるかという政策は，Hydra の上で動作するオペレーティングシステム（政策モジュール）が定めるものである．Hydra はその政策を実現する機構だけを提供する．② capability による保護機構：Hydra はその上で動作するオペレーティングシステムにマスタ／スレーブとか，リングといった構造上の制限を設けない．かわりに capability による保護機構[5)]だけを提供する．③拡張性：Hydra は抽象データ型を定義する機構を提供する．これによって，例えば，利用者が自分の要求にあったファイルシステムを生成（ファイル，ディレクトリ等の表現の定義とそれらに対する操作の定義）をおこなうことができる．Hydra は，複合体のオペレーティングシステムの例というだけでなく一般のオペレーティングシステムの構成法として有力な方法論を示したものといえよう．

6.2 StarOS kernel[14),15)]

カーネギーメロン大学では，C. mmp にひきつづき DEC LSI-11 を多数結合した，Cm* システム[9),33),34)]を開発している（図-3）．Cm* のアーキテクチャ上の特徴は，一つのクラスタを集中制御する Kmap の存在である．Kmap はマイクロプログラム制御のプロセッサであり，そのクラスタ内の LSI-11 から発せられるメモリ参照のアドレス変換をおこない，他のクラスタに属するセグメントへのアクセスも，相手 Kmapと通信することによって実現する．したがって各クラスタにローカルなメモリを，論理的には，全体として共通メモリとみなすことができる．また Kmap は，LSI-11 上で動作しているプロセスの capability リストを管理し，保護機構を実現している．更にプロセス間通信も kmap による capability の送信という形で実現される．StarOS kernel は LSI-11 上のソフトウェアと kmap 上のマイクロコードによって実装されたオペレーティング・システムの核である．kernel は capability による保護機構，アドレス空間の管理，協同して働くプロセスの集合体であるタスクフォースの管理，プロセス間通信などの機能を実現している．

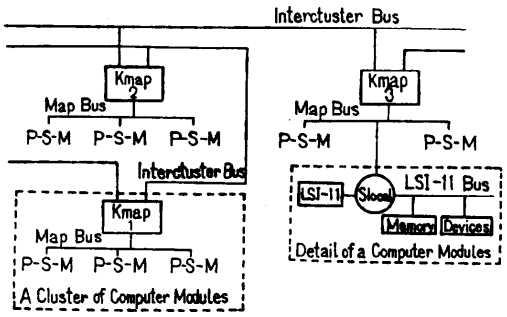


図-3 A three cluster Cm* Configuration[14)]

6.3 ポリプロセッサシステム EPOS[21),22),36)]

EPOS は，マイクロプログラム制御のコンピュータモジュール (CM) を最大32台まで結合可能な計算機複合体である．各 CM にはローカルメモリが付属し，大部分の仕事はローカルメモリ上でおこなわれる．この他に主として通信の目的のための共有メモリも用意されている．EPOS のオペレーティングシステムは，核およびその核の上で動作する応用システムとから構成される．応用システムとして現在タイムシェアリングシステムが実装されている．EPOS の特徴は，各 CM をファームウェアにより専用機械化し，専用機械の集合として一つの計算機システムを構成することにある．

TSS は，端末制御機械，ファイル専用機械[37)]，APL マシン[23)]，PASCAL マシン[43)]等の問題処理専用機械群，および利用者のコマンドを解析して問題処理専用機械にわりあてるジョブ制御機械などから構成される．ジョブ制御機械による割りあて方によって問題処理機械の役割が変化する．例えばすべての利用者が APL を要求すれば，すべての問題処理機械が APL マシンに変化する．核は，マイクロプログラムで実装され，すべての CM に同一のコピーがおかれる．核が提供する機能は，プロセッサタイムの割りあてとプロセス間通信機能であり，上記の各専用機械は，核の

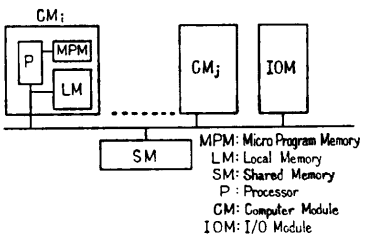


図-4 EPOS の構成

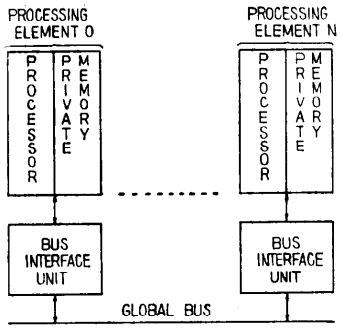


図-5 HXDP hardware architecture[13]

下でそれぞれ一つのプロセスとして動作する.

6.4 HXDP[13]

HXDP は分散処理をめざした計算機複合体である. 分散処理での中心課題は, システム全体にわたるグローバルな変数をもちいず, マスタ/スレーブといった階層関係をもたない, 非集中化 (Decentralized) 高レベルオペレーティングシステムの構築にある. このようなシステムの好例として, ARPA ネットワークの通信サブシステムをあげることができるが, 計算機システムとしての分散処理を実現するようなモデルはいまだ提案されていない[8].

HXDP は, 最大64個のプロセッサ/メモリ対をバスインタフェースユニット (BIU) を介してビットシリアルバスで結合したシステムである (**図-5**). BIU は, 同一プロセッサ/メモリ対内での場合も含めて, すべてのプロセス間メッセージの伝送およびキューイングをおこなう. HXDP のオペレーティングシステムは, プロセス制御という応用分野に対象を限って, 故障を起した要素の隔離 (Fault isolation) を第1目標として設計が進められている[2),3)].

7. まとめ

計算機複合体のオペレーティングシステムについて, その問題点, 70年代の実験システムが与えた解の例, 将来に向けての動向について論じた. オペレーティングシステムが取り扱わなければならない対象は, 極めて広範囲にわたるため, すべての問題を十分に取り上げられなかった. 特に性能の評価については, 紙数の都合上割愛した (例えば MICS についての報告[26]を参照されたい).

今後の計算機複合体に対して, 著者は次の4点が重要であると考えている.

(1) 複合体計算機システムの全体を表現しうるモデルの構築: 問題の分割, 割り付け, 故障からの回復, 再構成等を統一的に取り扱えるモデルが必要である.

(2) ハードウェア主導から, ソフトウェア主導へ: 70年代の計算機複合体は依然として, ハードウェア主導 (安くなったから, 結合してみよう) であったと思える. (1)のモデルに基づくソフトウェア主導の開発が必要になろう. ソフトウェアが必要とするハードウェア (例えばプロセス間通信, 保護等々) を開発するという体制を整えなければならない. さらにはハードウェアのモデル化も必要である.

(3) 応用システムおよびオペレーティングシステムを記述するための, (1)のモデルに基づいた言語の開発.

(4) 分散のモデル, および諸問題の解決: 分散は特に制御, 機能, データの3点についての検討がなされなければならない.

参考文献

1) 相磯秀夫: マイクロコンピュータのアーキテクチャとシステム構成, 情報処理, Vol. 17, No. 4, pp. 259-269 (1976, 4月).
2) Boebert, W. E., Franta, W. R., Jensen, E. D., and Kain, R. Y.: Design Issues in a Distributed Executive, Proc. IEEE COMPSAC Conference (Nov. 13-16 1978).
3) Boebert, W. E., Franta, W. R., Jensen, E. D., and Kain, R. Y.: KERNEL PRIMITIVES OF THE HXDP EXECUTIVE, Proc. IEEE COMPSAC Conference (Nov. 13-16 1978).
4) Brinch Hansen, P. Distributed Processes: A Concurrent Programming Concept, CACM Vol. 221, No. 11 pp. 934-941 (Nov. 1978).
5) Cohen, E. and Jefferson, D.: Protection in the Hydra Operating System, Proc. of the 5-th Symposium on Operating System Principles pp. 141-160 (Nov. 1975).
6) Denning, P. J.: Fault Tolerant Operating Systems, Computing Survey Vol. 8, No. 4 pp. 359-389 (Dec. 1976).
7) Dennis, J. B. and Misunas, D. P.: A Preliminary Architecture for a Basic Data Flow Processor, Proc. 2nd Ann. Symp. on Computer Architecture pp. 126-132 (1975).
8) Enslow, Jr. P. H.: What is a "Distributed" Data Processing System?, IEEE Computer pp. 13-21 (Jan. 1978).
9) Fuller, S. H., Ousterhout, J. K., Raskin, L., Rubinfeld, P. I., Sindhu, P. J., and Swan, R. J.: Multi-Microprocessor: An Overview and

Working Example, Proceedings of the IEEE Vol. 66, No. 2, pp. 216-228 (Feb. 1978).
10) 飯塚, 古谷: C. mmp マルチミニプロセッサについて, 情報処理, Vol. 15, No. 7, pp. 550-556 (1974, 7月).
11) 飯塚, 藤井, 古谷: モジュール型複合計算機(ACE)の試み, 情報処理, Vol. 20, No. 4, pp. 314-318 (1979, 4月).
12) Jensen, E. D. and Boebert, W. E.: Partitioning and assignment of distributed software, COMPCOM '76 EAST
13) Jensen, E. D.: The Honeywell Experimental Distributed Processor—an Overview, IEEE Computer pp. 28-38 (Jan. 1978).
14) Jones, A. K., Chansler, R. J., Durhan, I., Feiler, P., and Schwans, K.: Software management of Cm*—A distributed multiprocessor, AFIPS Conference Proceedings Vol. 46, pp. 657-663 NCC (1977).
15) Jones, A. K., Chansler, Jr. R. J., Durham, I., Feiler, P. H., Scelza, D. A., Schwans, K., and Vegdahl, S. R.: Programming Issues Raised by a Multiprocessor, Proceedings of the IEEE Vol. 66, No. 2, pp. 229-237 (Feb. 1978).
16) 上林, 相磯: プロセス間通信機能指向ミニコンピュータ複合体 KOCOS のアーキテクチャ, 情報処理, Vol. 20, No. 4, pp. 307-313 (1979, 4月).
17) Katsuki, D., Elson, E. S., and Mann, W. F. et al.: Pluribus—An Operational Fault-Tolerant Multiprocessor, Proc. IEEE Vol. 66, No. 1, pp. 1146-1159 (Oct. 1978).
18) Lamport, L.: Time, Clocks, and the Ordering of Events in a Distributed System, CACM Vol. 21, No. 7, pp. 558-565 (July 1978).
19) Le Lann, G.: Distributed Systems-Towards a Formal Approach, Proc. IFIP Congress 77 pp. 155-160.
20) Lomet, D. B.: Process Structuring, synchr Nouizaton and recovery using atomic actions, Proc. of ACM Conf. Language Design for Reliable Software SIGPLAN Notices Vol. 12, No. 3, pp. 128-137 (March 1977).
21) Maekawa, M., Yamazaki, I., and Maeda, A. et al.: Experimental Polyprocessor System (EPOS)—Architecture, Proc. 6th Annual Symp. on Computer Architecture (April 1979).
22) Maekawa, M., Yamazaki, I., and Tanaka, A. et al.: Experimental Polyprocessor System (EPOS)—Operating System, Proc. 6th Ann. Symp. on Computer Archtecture (April 1979).
23) Maekawa, M. and Morimoto, Y.: Performance Adjustment of an APL Interpreter, Proc. EUROMICRO 79 Conference (Aug. 1979).
24) 元岡 達: コンピュータコンプレクスの展望, 情報処理, Vol. 15, No. 7, pp. 525-533 (1974, 7月).
25) 元岡, 山村: ポリプロセッサシステム PPS-1, 情報処理, Vol. 15, No. 7, pp. 557-564 (1974, 7月).
26) 大森, 小池, 山崎, 大宮: 計算機複合体 MICS-II のシステム評価, 情報処理学会論文誌, Vol. 20, No. 2, pp. 130-137 (1979, 3月).
27) Randell, B., LEE, P. A., and Trlraven, P. C.: Reliability Issues in Computing System Design, Computing Surveys Vol. 10, No. 2, pp. 123-165 (June 1978).
28) Russell, D. L.: Process Backup in Producer-Consumer Systems, Proc. of the 6-th ACM Symposium on Operating Systems Principles pp. 151-157 (Nov. 1977).
29) Sato, M., Nishimura, S., Murakami, K., and Takahira, S.: Dynamic Function Exchange Mechanism in Polyprocessor, Proc. 6th Ann. Symp. on Computer Architecture, pp. 130-136 (April 1979).
30) 白男川, 三木, 森, 松本: 負荷分散型マルチミニコンシステム, 電気四学会連合大会予稿集 (昭和53年).
31) Siewiorek, D. P., Kini, V., Mashburn, H., McConnel, S., and Tsao, M.: A Case Study of C. mmp, Cm*, and C, vmp: Part I—Experience, with Fault Tolerance in Multiprocessor Systems, Proc. of the IEEE Vol. 66, No. 10, pp. 1178-1199 (Oct. 1978).
32) 杉本正勝: C. mmp の評価, 情報処理, Vol. 20, No. 4, pp. 301-306 (1979, 4月).
33) Swan, P. J., Fuller, S. H., and Siewiorek, D. P.: Cm*—A moduler, Multi-minimicroprocessor, AFIPS Conference Proceedings Vol. 46, pp. 637-644, 1977 NCC
34) Swan, R. J., Bechtolsheim, A., Lai, K. W., and Ousterhout, J. K.: The implementation of the Cm* multi-microprocessor, AFIPS Conference Proceedings Vol. 46, pp. 645-655, 1977 NCC
35) 高平 叡: 分散処理用オペレーティングシステムの課題, 情報処理, Vol. 20, No. 4, pp. 284-288 (1979, 4月).
36) 田中, 若原, 菊地: ポリプロセッサオペレーティングシステム: EPOS の構造, 第17回情報処理学会全国大会 p. 169 (1976).
37) 田中, 若原: ポリプロセッサシステム EPOS におけるファイルプロセッサ, 昭和54年電子通信学会総合全国大会 p. 1458 (1979).
38) Walden, D. C.: A System for Interprocess Communication in a Resource Sharing Computer Network, CACM Vol. 15, No. 4, pp. 221-230 (April 1972).
39) Wulf, W. A. and Bell, C. G.: C. mmp—A Multi mini processor, Proc. FJCC Vol. 39, pp. 779-790 (1972).

40) Wulf, W. A., Cohen, E., Corwin, W., Jones, A., Levin, R., Pierson, C., and Pollack, F.: Hydra: The Kernel of a Multiprocessor Operating System, CACM Vol. 17, No. 6, pp. 337-345 (June 1974).
41) Wulf, W. A. and Levin, R.: The Hydra Operating System (Preliminary-Draft) Department of Computer Science Carnegie-Mellon Univ. (June 1975).
42) Wulf, W. A.: Reliable Hardware/Software Architecture, SIGPLAN Notices Vol. 10, No. 6 pp. 122-130 (June 1975).
43) 吉村，藤田：ポリプロセッサシステム EPOS におけるパスカルプロセッサ，昭和54年電子通信学会総合全国大会，p. 1457 (1979)．

(昭和54年6月8日受付)