オペレーティング・システム 38 － 6
（１９８８．２．２６）

性能監視エキスパートシステム：ＥＸＰＥＣＴ

住田　宏己＊　村井　孝＊　大庭　剛＊＊　加藤　一義＊＊　中谷　智紀＊＊
＊富士通（株）　＊＊（株）富士通愛知エンジニアリング

計算機システムの性能トラブルを監視し、チューニング方法をアドバイスするエキスパートシステムを開発した。大規模な知識を取り扱う難しさを避けるため、チューニング項目ごとに知識を分割して管理した。分割された知識は、制御部分から呼び出す構造とした。また、実用的なエキスパートシステムを構築する際には、既存ツールとの連携が重要である。本システムでは待ち行列網解説ツールや統計解析グラフィックパーケッジなどと連携した。本稿では、知識の分割を中心にシステムの概要を述べ、具体的な知識の記述については仮想記憶不足の問題の例を用いて述べる。

An Expert System of Performance Consulting and Tuning

Hiroki SUMIDA *,　Takashi MURAI *,
Tsuyoshi OOBA **,　Kazuyoshi KATO **,　Tomonori NAKATANI **
* FUJITSU Ltd.　** FUJITSU AICHI ENGINEERING Ltd.

FUJITSU LIMITED, 140 MIYAMOTO, NUMAZU-SHI SHIZUOKA, 410-03 JAPAN

We have developed an expert system EXPECT. It can watch status of a computer system performance, and advise tuning. Large scale knowledge base is divided into functional groups based on tuning item. And these functions are called by a main module. For practical use, It is important to use any other ordinary tool in the expert system. In this system, queueing network model solver and analytical graphic system are used. This paper describes an outline of EXPECT and example knowledge to solve lack of virtual storage space.

1. はじめに

計算機システムの性能評価ツールを、エキスパートシステムとして実現することを試み、EXPECT (Expert System of Performance Consulting and Tuning) を製作した。EXPECTは、FACOM OSIV/F4 MSP E20 で動作する。既存のソフトウェアモニタ PDL / PDA (Performance Data Logger / Performanc Data Analyser) で測定されたデータを、熟練者と同様な手順で分析し、チューニング項目をアドバイスするシステムである。

我々は自分たちのノウハウをツールとして流通させる際に、どのようにすれば実用的なエキスパートシステムを構築できるかという点に関心がある。

我々のシステムの特徴は、知識を機能モジュールに分割して整理した点である。これにより、膨大な量に及ぶ熟練者の分析手順を忠実に辿ることができた。一般にエキスパートシステムを構築する際のポイントは知識の獲得にある。同時に、複数の熟練者から取り出した知識のマージ方法も解決されていない課題である。異なる２人の熟練者の知識が同じ構造を持っているとは考えにくく、このことが熟練者同士の共同開発を難しくしているからである。我々は、性能関連作業における知識がチューニング対象によって分類できることを利用してシステムを機能モジュールに分割した。複数の熟練者の知識をマージする時の難しさ、巨大な知識を扱う難しさを避けた。機能モジュールは相互に情報をやりとりせずに独立に動作するものである。このやり方で手順は統合できたが総合的な分析を得にくい欠点が残る。機能モジュールとして整理された知識を元にした総合的な判断を可能にするのは、今後の課題である。

本稿では、機能モジュールの分割という特徴を中心に、EXPECTの概要について報告する。また、他の論文で取り上げられていない仮想記憶不足の問題を例に、知識の詳細を述べる。

2. 解析対象

本稿の報告は対象を以下の範囲に絞る。

１つめは、既存の知識を利用している点である。我々は、新しい知識を発見して新しいツールを作るというのではなく、既存の熟練者のノウハウをシステム化することを狙っている。熟練者は現在の計算機にはどのような性能評価項目があるか、それを調べるためにどのようなソフトウェアモニタが利用できるかを知っている。またモニタされたデータの解析手法、原因究明の手順、チューニングパラメタの設計方法、チューニング効果の予測手法を知っている。しかし、新しいハードウェアについては、データの採取方法もチューニングパラメタも分からない。我々は、既存の知識をシステム化することに最初の興味を持っている。熟練者のノウハウを活かすために既存ツールを有効に利用することが重要である。知識ベースシステムは、エキスパートシステムの構築するための手段の一部であり全てではないと考えている。

例えば、チューニング効果の予測では待ち行列網解析ツールを利用している。

ソフトウェアモニタについても既存のものを利用している。このシステムとソフトウェアモニタとの関連を、図１に示す。計算機センターの負荷がピークを示す時間帯に数十分から１時間程度の間、ソフトウェアモニタで測定する。その後、測定・編集されたデータを取り込み、データベースに蓄積する。蓄積されたデータをもとに、資源使用状況をチェックし、問題があれば、対応する分析機能を起動して対策をアドバイスする。

２つめは、システム性能をマクロに捕らえた場合の問題解決のアプローチである。つまり、ＣＰＵやチャネルなどのハードウェアの平均的な使用率から、システム全体の処理能力を高めたりコマンドの平均的な応答時間を短縮するための方法をアドバイスする手法である。これに対するミクロな分析としては、個々のジョブやコマンドの処理が遅い原因を詳細に分析する場合がある。我々のシステムでは、まずマクロに捕らえてシステム全体の処理能力を高めるための分析を行っている。

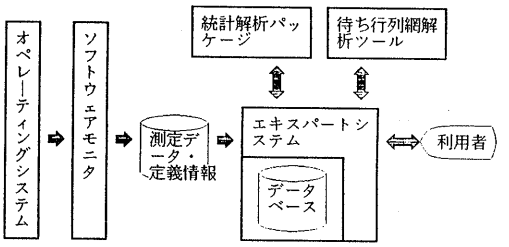

図1. 性能データの流れ

3. 機能モジュールの分割と制御

経験により得られた知識は、プロダクションルールやフレームなどが扱えてオブジェクト指向の記述ができるツール ESHELL/X と、Common Lisp に準拠した Lisp システムを利用して記述している。しかし、性能関連作業では、待ち行列網モデルやグラフによる解析など、様々な手法を複合して用いることも多く、既存のツールと結合してトータルなエキスパートシステムとすることを考えた。

実際の性能関連作業では、性能データを測定するためのソフトウェアモニタ、待ち行列網モデルを解析するサブルーチン、解析結果をグラフ表示するための統計解析パッケージなどを利用している。

ソフトウェアモニタはＯＳ内部と連携して動作するが、我々の製作したシステムは今のところＴＳＳ配下で動作するアプリケーションプログラムであり、ＯＳとインタフェースを持っていない。ソフトウェアモニタをエキスパートシステムには結合するのではなく、ソフトウェアモニタが測定し編集した結果を、ファイルを経由してエキスパートシステムが読み込むことで実現した。

待ち行列網解析サブルーチンや統計解析パッケージはFORTRANで記述してあり、大きな仮想記憶領域を使うツールである。一方、推論を行うためのエキスパートシステム構築ツールであるESHELL/XやLISPも、一般的に大きい仮想記憶領域を必要とする。そこで、機能単位にモジュールを独立させ、モジュール構成の頂点に位置する制御モジュールから、各機能モジュールを呼び出す構造とした。熟練者の作業手順の大きな流れは、この制御モジュールで実現した。

機能モジュールは表１のように分割した。機能モジュールに分割することにより、チューニング項目ごとに独立に知識を追加できた。

表1. 機能モジュール一覧表

機能名	機能概要
性能データ蓄積	ソフトウェアモニタによる測定データを入力し、データベースに蓄積する。
データベースアクセス	データベースのレコードを取り出す。
ピーク時監視	資源使用率をチェックし、性能トラブルの有無を調べる。
設備見積り	ＣＰＵ・磁気ディスク装置・ＴＳＳ端末などの機種の変更や装置の追加・削除を行う前に、システム変更による資源使用率やサービスレベルへの影響を予測する
ＤＡＳＤ分析 ＤＡＳＤとは直接アクセス装置、即ち磁気ディスク装置などのこと。	特定の磁気ディスク装置に入出力要求が集中している場合、磁気ディスク装置内のいくつかのファイルを他の磁気ディスク装置に移動することにより、負荷を均等化する。
チャネル分析	特定のチャネルに入出力要求が集中している場合、そのチャネル配下のＤＡＳＤボリュームを負荷の低いチャネルの配下のものと交換することにより、チャネルの使用率を均等化する。
ファイル分析	特定のファイルの入出力要求頻度が高いために全体の応答時間を悪化させている場合、ロードモジュールライブラリや、システムファイルなど、ファイルの特性に合った固有のチューニング方法をアドバイスする。
仮想記憶分析	空間固有域やシステム共通域などの仮想記憶領域の過不足をチェックし、バランス良く再配置する方法をアドバイスする
仮想記憶見積り	システム共通域を多く消費している場合に、消費しているコンポーネントを調べることにより原因を分析し、ログオン端末数を削減するなどの運用制限を立案しその効果を見積もる。
ＳＤＭ分析	システムチューニングパラメタの定義体を静的に分析し、矛盾のない指定かどうか、妥当性をチェックする。

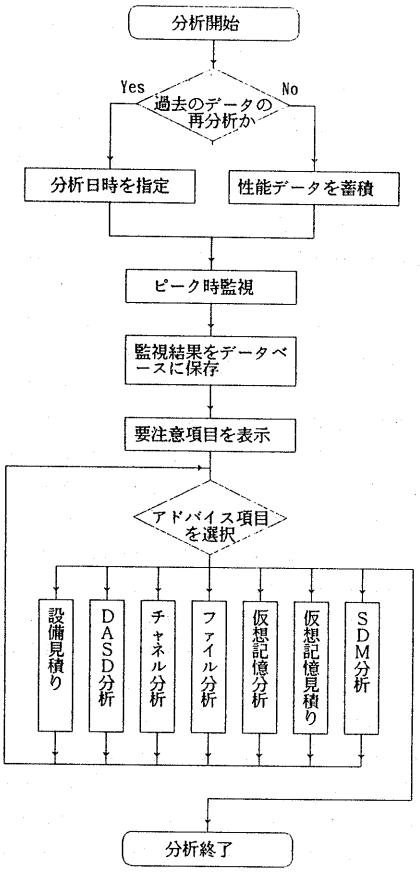

図2. 機能モジュールの流れ

表１のピーク時監視機能が問題点を検出した後、設備見積り以下の各分析機能を起動すると、自動的に対策を立案する。また、端末からの操作により、別案を示すことができる。いずれも分析結果を日本語とグラフで表現する。

4. ピーク時監視の手順

１日の中でシステム負荷がピークになる時間帯に、性能データを測定・蓄積し、それをもとにシステム性能を監視するものである。熟練者の場合、まず初めに問題があるかどうか診断を下す。そのやり方は、ソフトウェアモニタで測定し編集されたデータを眺め、表２に示すように、経験的な知識として記憶している評価基準値と比較し、特に悪化している項目がないか、また、特定の装置に負荷が集中するような偏りがないかチェックするものである。その結果、例えば、応答時間が遅い場合に、実メモリ不足によるものか、ＣＰＵの過負荷によるものかなど、原因を絞り込むものである。このようなピーク

時の監視機能はいわゆる診断型エキスパートシステムである。

AならばBという推論が経験だけから得られる場合を「浅い推論」と呼ぶことにすると、表2に示すように、日常的な監視作業においては「浅い推論」を用いていると仮定してよさそうである。例えば、ページング率が高すぎるという評価基準値は、ページング率が高い時は、応答時間か悪化しているという経験から得られた値である。

このような知識を知識ベースとして蓄えるようとすると、実際には知識にかなり漏れがあることが分かる。例えば、存在するすべてのＣＰＵ機種についてチューニングの経験があるような熟練者はめったにいない。経験のないＣＰＵ機種についてページング率が高いかどうか判定するために、熟練者は、ＣＰＵやページングデバイスのハードウェア性能の相対値を知っている。あるいは、相対値の一覧表の在りかを知っている。このような一覧表は経験的に得られた値ではなく、ハードウェアを実際に測定した結果であったり、理論的に計算された値であったりする。これらはプロダクションルールで推論するような知識とは異質である。性能関連作業ではこの種の知識を多用する。

我々は、ピーク時の監視の部分のプロトタイプをエキスパートシステム構築ツールで開発した後、監視項目を追加することを計画した。この追加は、いわば様々な一覧表を追加する作業である。この一覧表はルールやフレームで記述するよりも、従来の手続型言語の方が記述しやすく実行時のオーバーヘッドも小さいと考え、Ｃ言語で作りなおした。

この機能において、熟練者の知識の一部である評価基準値は、ハードェアの機種や、業務の形態に応じて異なるものである。ハードウェアの機種や、業務の形態にどのように依存するかという点まで知識として整理できればよいが、それは困難である。計算機センターの個々の事情によって決まるものもある。例えば、コマンドの応答時間の目標値などは個々の計算機センターのシステム設計時に設定するものである。評価基準値をデータベース内に保存し、計算機センター管理者が持っている経験値によっていつでも修正できる仕組みを組み込んだ。

5. 知識例　——仮想記憶分析の手順——

5.1 問題の定義

我々は、この、仮想記憶分析に関してプロトタイピングを行った結果をすでに報告している［1］。今回、知識を整理しなおし、知識の強化を図った。本稿では、整理しなおした知識の構造について詳細に述べる。はじめに仮想記憶不足の問題を定義する。

我々が分析の対象としているＯＳでは、仮想記憶領域が用途ごとに図3のように分割されている。各領域は配置される相対関係が決まっているだけで、その大きさは計算機システムの構成に応じて可変である。

図3に示す領域のうち、核の部分には、入出力装置の情報や、システムに常駐しなければならない制御プログ

表2. 日常的な監視項目と基準

	監視項目	判断基準：実測値を評価基準値と比較する	評価基準値を決定する上での要因
ハードウェア	ＣＰＵ使用率	定常的に高い値になっていないか。スーパバイザモードの使用率の割合が高くなっていないか。	大型機か、中型機か、などの機種に依存。オンライン業務が主体か、科学技術計算主体かといった処理形態にも依存する
	チャネル使用率	定常的に高いチャネルはないか。使用率に偏りはないか。	プリンタ、磁気ディスク装置、磁気テープ装置など、チャネルの配下に接続されている装置の種別に依存する。
	磁気ディスク装置使用率・Ｉ／Ｏレスポンスタイム	定常的に高い値になっていないか。特定ボリュームが特に悪化していることはないか	通常の磁気ディスク装置か、半導体ディスク装置かなど、装置の種別に依存する。ディスク装置内のファイルの種別にも依存する。
	ページング回数	定常的に高い値になっていないか。	ＣＰＵ機種、磁気ディスク装置種類にも依存する。
	実メモリ使用量	余裕がなくなっていないか。	ＣＰＵ機種に依存する。
ソフトウェア	空間固有域サイズ	空間固有域が以前より狭くなってはいないか。	ユーザジョブが使用する最大空間固有域サイズに依存する。
	システム共通域	システム共通域の余裕がなくなってはいないか。	計算機センターのシステム規模・運用形態全般に依存する。過去の実績にも基づく。
	タスク使用率・タスク処理待ち時間	オンラインシステムで、タスクの使用率が高くて待ち時間が増加していないか。	トランザクションの処理内容（業務）に依存する。
サービスレベル	ＴＳＳコマンド応答時間・オンライントランザクション処理時間	平均的にみて基準値以内に収まっているか悪化していないか。	業務内容に依存する。システム設計時の目標値に依存する。

ラムがローディングされる。

ＬＰＡはＯＳのモジュールを常駐させる領域である。

ＢＬＤＬは、システム共通のロードモジュールライブラリを検索する時間を削減するために、ディレクトリ情報を実記憶内に展開したものである。ディレクトリ情報とは、ロードモジュールライブラリに入っているメンバについて、メンバ名とライブラリ上の物理的な位置の対応を一覧表にしたものである。

ＳＱＡ・ＣＳＡはＯＳが使用するデータ領域であり空間共通のデータを扱うために使用している。この領域の大きさは、計算機システムを立ち上げる時に予約する。

例えば、ＴＳＳセッションが１台開設されるたびに、ＣＳＡから一定量を切り出して使用する。あるいは、空間に共通なファイルをオープンするたびに切り出して使用される。この領域が不足すると、新たにＴＳＳ端末を開設できなかったり、極端な場合はシステムダウンにつながる事態に陥る。このため、この領域の使用状況を常時監視することが重要になる。ＣＰＵ資源の使用量と同様に、徐々に増加する性質のものだからである。

一方、空間固有域は、ユーザジョブが使用する領域である。プログラムの巨大化により、空間固有域を最大限度まで確保したいという要求がある。

限られた仮想記憶領域の中では、システム共通域の確保と、空間固有域の拡大は、相反する課題である。ＯＳ本来の機能として、これらの問題を解決する必要があるが、当面の回避手段が存在する場合がある。経験者の持っているそういった回避手段を有効に利用することが、この機能の狙いである。

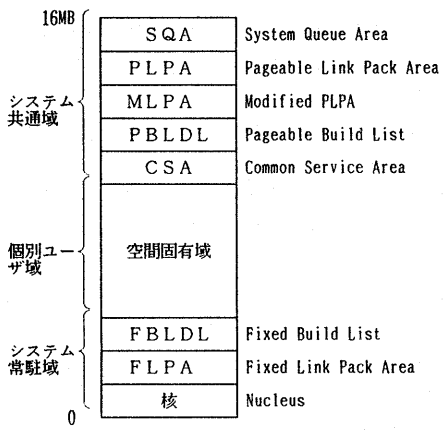

図3. 仮想記憶領域の配置例

5.2 問題の有無を検出

システム共通域、空間固有域が不足しているかどうかを、評価基準値との比較によって判定する。計算機センターごとの事情で、システム共通域や、空間固有域の最大値が異なる。そこで、システム共通域については、使用率を、空間固有域については最大ジョブの使用量を、設定する。設定された値は、データベース内に保存される。

問題点の検出は、ピーク時監視機能が実施しており、その結果がデータベース内に保存されている。仮想記憶分析機能は、見付けられている問題点に基づき、対策を立案するところから推論を開始する。

5.3 仮想記憶領域の構造

ＳＱＡ、ＣＳＡは、ＯＳを立ち上げる際にパラメタで領域を予約しておく。稼働中にこの領域から必要量を切り出して使われる。不要になると返却される。

ＢＬＤＬ、ＬＰＡ、核の領域にローディングするものは、ＯＳを立ち上げる時にパラメタで指定する。立ち上げた後み、これらの領域の使用量は変化しない。

空間固有域は、16MBから、ＳＱＡとＣＳＡの予約量、ＬＰＡ、ＢＬＤＬ、核の領域を除外した残りが割当てられる。

空間固有域は、ＣＳＡと核にはさまれるが、その境界は 1MBまたは 64KB の単位になるようにそろえられる。1MB の境界の場合、核・ＦＬＰＡ・ＦＢＬＤＬの領域が 0.8MB しか使用しなくても、空間固有域は 1MBのところから始まるため、0.2MB 分は使用されない領域となる。

ＳＱＡやＣＳＡはシステム立ち上げ時に指定した予約量の中から切り出される。予約量が使用量に比べ大きすぎると、領域が無駄に予約されたことになる。

仮想記憶分析の目標は、空間固有域の両側の境界調整領域と、ＳＱＡ・ＣＳＡの未使用量をバランスよく最小限度に収めることである。

空間固有域の配置方式と境界調整の方式をＬＩＳＰで記述したものを図４に示す。空間固有域の大きさは、境界調整の影響を考慮しながら、16MBから空間固有域の両側の領域の大きさを引くことで求まる。境界調整については、仮想記憶領域を変更する前の境界との差を記述している。空間固有域の両側のそれぞれの領域の大きさは、フレームのスロット値を参照している。

仮想記憶領域の配置自体は図５のようにフレームで表現している。我々が利用したＥＳＨＥＬＬ／Ｘでは、フレームをオブジェクトで実現した。仮想記憶マップを一つのオブジェクトとしてとらえ、ＳＱＡやＬＰＡのサイズをスロットとして保持する。仮想記憶領域を再配置することは、仮想記憶マップのフレームに基づいて新しい

仮想記憶マップを創成することで記述する。新しい仮想記憶マップを創成した後、仮想記憶領域の配置の変更点に合わせてスロット値を変更することになる。

空間固有域や境界調整領域については他のスロット値を元に決まるものであるから、他のスロット値がすべて決定した後で、値を参照する時に正しい値が得られれば十分である。空間固有域や境界調整領域のスロット値の設定では欠測値デモンと呼ぶ手続きを利用し、図4の関数を指定している。¥IF-NEEDEDという指定は、スロットを参照した時点でそのスロットにまだ値が設定されていなかった場合に、値を獲得する手続を指定するものである。

```
(DEFUN  空間固有域の配置 ()
  (- 16  (境界調整単位に切上げ
           (+  (! *OBJECT* 'REF 'ＳＱＡ予約量)
               (! *OBJECT* 'REF 'ＰＬＰＡ)
               (! *OBJECT* 'REF 'ＭＬＰＡ)
               (! *OBJECT* 'REF 'ＰＢＬＤＬ)
               (! *OBJECT* 'REF 'ＣＳＡ予約量)))
         (境界調整単位に切上げ
           (+  (! *OBJECT* 'REF 'ＦＢＬＤＬ)
               (! *OBJECT* 'REF 'ＦＬＰＡ)
               (! *OBJECT* 'REF '核)))
))

(DEFUN  低アドレス側の境界調整 ()
  (-   (境界調整単位に切上げ
         (+  (! '仮想記憶マップ  'REF 'ＦＢＬＤＬ)
             (! '仮想記憶マップ  'REF 'ＦＬＰＡ)
             (! '仮想記憶マップ  'REF '核)))
       (+  (! *OBJECT* 'REF 'ＦＢＬＤＬ) )
           (! *OBJECT* 'REF 'ＦＬＰＡ)
           (! *OBJECT* 'REF '核)))
))

(DEFUN  高アドレス側の境界調整 ()
  (-   (境界調整単位に切上げ
         (+  (! '仮想記憶マップ  'REF 'ＳＱＡ予約量)
             (! '仮想記憶マップ  'REF 'ＰＬＰＡ)
             (! '仮想記憶マップ  'REF 'ＭＬＰＡ)
             (! '仮想記憶マップ  'REF 'ＰＢＬＤＬ)
             (! '仮想記憶マップ  'REF 'ＣＳＡ予約量)))
       (+  (! *OBJECT* 'REF 'ＳＱＡ予約量)
           (! *OBJECT* 'REF 'ＰＬＰＡ)
           (! *OBJECT* 'REF 'ＭＬＰＡ)
           (! *OBJECT* 'REF 'ＰＢＬＤＬ)
           (! *OBJECT* 'REF 'ＣＳＡ予約量))
))
```

図4.　空間固有域の配置の規則

5.4　仮想記憶領域の変更

空間固有域が不足した場合に可能な配置換えの例として、次のような方法がある。

① ＦＢＬＤＬとＰＢＬＤＬの違いは、配置する場所だけの違いである。ＦＢＬＤＬをＰＢＬＤＬに変更したり、その逆を行うことが可能である。ＢＬＤＬを削除できる場合もある。

② ＦＬＰＡ内のモジュールはＭＬＰＡ内に移すことができる。ＰＬＰＡ内に移すより簡単である。逆に、ＰＬＰＡ内のモジュールをＦＬＰＡに移すこともある。また、ＢＬＤＬと同様に、モジュールをＬＰＡから追い出せる場合もある。

③ ＣＳＡ・ＳＱＡの使用率が小さい場合は、予約量を削減できる。

このような配置換えは、図5のようにフレームのメソッドとして、手続を記述している。厳密には、例えば、ＬＰＡについては、ＦＬＰＡ内の一部のモジュールを移動することになるから、移動候補となるモジュールを選びだす必要があるが、ここでは省略する。

5.5　対策立案

仮想記憶領域の構造や変更に関する記述は、空間固有域を拡張する対策を立案するための基礎知識となる。これらは経験則ではなく、ＯＳの構造から決まるものである。つまり、この問題はいわゆる設計・計画型の問題であり、すべての変更方法の組合せを試し効果を比較すれば、空間固有域を拡張する対策を立案できる。仮想記憶内の配置に関係する要因は多数存在するが、空間固有域の問題を解決するために有効なものを絞り込んだもの、また、採用する際の優先順位を付与したものが経験則だと言える。

例えば、入出力装置を追加したことにより、核の部分が増加したとする。この時、核・ＦＬＰＡ・ＦＢＬＤＬの領域の和が 1MBを越えると、空間固有域との境界調整の結果、空間固有域は 1MB小さくなることになる。もし、ＦＢＬＤＬを除くと 1MBに収まり、また、ＣＳＡの使用率が低く、ＢＬＤＬの大きさ程度ならＣＳＡの予約量を小さくしても問題がないのであれば、ＦＢＬＤＬをＰＢＬＤＬに移し、かつその分だけＣＳＡの予約量を変更することで問題は解決する。

この問題に適合する解決手順は次のようなものである。

① 空間固有域が評価基準値より 1MBだけ小さく、かつ核＋ＦＬＰＡ＋ＦＢＬＤＬが 1MBを越えている場合、ＦＬＰＡやＦＢＬＤＬを移動すれば空間固有域との境界調整領域が元どおりになって問題が解決する可能性がある。

② ＦＢＬＤＬをＰＢＬＤＬに変更するだけで解決しないか試みる。

③ ただし、このときＦＢＬＤＬがＰＢＬＤＬに移動することによって高アドレス側の境界が移動して空間固有域の高アドレス側を圧迫する危険がある。高アドレス側の境界調整領域が十分に空いているか、または、ＣＳＡ・ＳＱＡの使用率からみてＣＳＡ・ＳＱＡの予約量を削減できる場合でなければ適用できない。

④ 高アドレス側の空き領域がどうなるかは、ＯＳの構造の基礎知識であり、図5に示したデモンによって自動的に評価できる。

⑤ 高アドレス側に空きがなければ、ＣＳＡ・ＳＱＡの予約量の削減の可能性を調べることになる。ＣＳＡ・

ＳＱＡの予約量を幾らまで縮小できるかという知識はやはり図５に記述したメソッドで得られる。

⑥　ＦＢＬＤＬだけでは解決しないならば、ＦＬＰＡ内の一部のモジュールも移動することを試みる。このとき移動するモジュールとしては、目標を満たす最小の部分集合を見付ければよい。

このように後ろ向き推論で記述すると見通しがよくなる。記述例を図６に示す。ただし、図６では、ＦＬＰＡ内の一部のモジュールを移す仕組みや、利用者との応答部分は除き、手順の概要だけを記述している。

以上は、核が増加したことによる空間固有域の圧迫を、ＦＢＬＤＬとＦＬＰＡの配置換えで解決する手順である。ＣＳＡやＳＱＡの使用率が増加して空間固有域の圧迫した場合は、核の側の境界調整領域はすでに最小になっている可能性が高いから、ＦＬＰＡやＦＢＬＤＬの移動で解決できる可能性は少なく、ＰＬＰＡやＭＬＰＡ内のモジュールの削除を検討することから始める方が早い。このように、経験から得られる優先順位に基づいて記述することと、何が起こったのかという性能の変化も判断材料として利用するのが有効である。

5.6　競合の解消

この記述により、同時に、変更方法を採用する際の優先順位も記述していることになる。ＦＢＬＤＬを移動する方がＦＬＰＡを移動するよりも簡単にできるので、通常はまずＦＢＬＤＬの移動から試みる、というような順位である。ここでは、方法の簡単さと、移動の影響の少なさを基準に優先順位を決め、優先度の高いものから適用して、解が見付かったところで推論を中止するやり方をとっている。

この優先順位だけでは満足できない場合、別案を立てられる仕組みを組み込んでいる。解を求める過程で、各々の移動のルールごとにルールが適用可能と推論された時点で、その移動方法を採用してよいかどうか利用者に確認を促すというインタフェースで実現している。採用してはいけないと指示した場合は、その案を破棄し、次善の策を次々と求めることが可能になった。

6.　おわりに

本稿では、性能関連作業を支援するエキスパートシステムの構築例を報告した。記述言語にとらわれず、複数のツールを組み合わせることによって、実用性が高まると考える。各機能は独立しており、一つの機能を一人の開発者が担当することで、知識のマージの問題を回避した。制御部が各々の機能を起動する方式なので、チューニング項目を追加する際は、制御部からの呼び出しを追加するだけでよい。

現在は、システムのデバグ中であり、実際の計算機センターでの適用試験はこれから行うところである。

参照文献

［１］　村井　他，“システム性能監視へのエキスパートシステムの適用”，情報処理学会全国大会，第35回，昭和62年後期．


```
( オブジェクト定義　評価基準値
  システム　レベル　クラス
  属性
    ＳＱＡ使用率　0.8
    ＣＳＡ使用率　0.8
)

( オブジェクト定義　仮想記憶マップ
    システム　レベル　クラス
  属性
    ＳＱＡ予約量　＊
    ＳＱＡ使用量　＊
    ＰＬＰＡ　　＊
    ＭＬＰＡ　　＊
    ＰＢＬＤＬ　＊
    ＣＳＡ予約量　＊
    ＣＳＡ使用量　＊
    高アドレス側の空き　＊ ¥IF-NEEDED 高アドレス側の境界調整
    空間固有域　　＊ ¥IF-NEEDED 空間固有域の配置
    低アドレス側の空き　＊ ¥IF-NEEDED 低アドレス側の境界調整
    ＦＢＬＤＬ　＊
    ＦＬＰＡ　　＊
    核　　＊
  メソッド
    ＦＢＬＤＬのＰＢＬＤＬ化
      (LAMBDA (OBJ)
        (COND ((> (! OBJ 'REF 'ＦＢＬＤＬ) 0.0)
               (! OBJ 'SET 'ＰＢＬＤＬ
                  (! OBJ 'REF 'ＦＢＬＤＬ))
               (! OBJ 'SET 'ＦＢＬＤＬ 0) T)))
    ＰＢＬＤＬのＦＢＬＤＬ化
      (LAMBDA (OBJ)
        (COND ((> (! OBJ 'REF 'ＰＢＬＤＬ) 0.0)
               (! OBJ 'SET 'ＦＢＬＤＬ
                  (! OBJ 'REF 'ＰＢＬＤＬ))
               (! OBJ 'SET 'ＰＢＬＤＬ 0) T)))
    ＦＢＬＤＬの削除
      (LAMBDA (OBJ)
        (! OBJ 'SET 'ＦＢＬＤＬ 0) T)
    ＰＢＬＤＬの削除
      (LAMBDA (OBJ)
        (! OBJ 'SET 'ＰＢＬＤＬ 0) T)
    ＦＬＰＡのＭＬＰＡ化
      (LAMBDA (OBJ)
        (COND ((> (! OBJ 'REF 'ＦＬＰＡ) 0.0)
               (! OBJ 'SET 'ＭＬＰＡ
                  (+ (! OBJ 'REF 'ＭＬＰＡ)
                     (! OBJ 'REF 'ＦＬＰＡ)))
               (! OBJ 'SET 'ＦＬＰＡ 0) T)))
    ＳＱＡ予約量の縮小
      (LAMBDA (OBJ)
        (COND ((< (/ (! OBJ 'REF 'ＳＱＡ使用量)
                     (! OBJ 'REF 'ＳＱＡ予約量))
                  (! '評価基準値 'REF 'ＳＱＡ使用率))
               (! OBJ 'SET 'ＳＱＡ予約量
                  (/ (! OBJ 'REF 'ＳＱＡ使用量)
                     (! '評価基準値 'REF 'ＳＱＡ使用率)))
               T)))
    ＳＱＡ予約量の初期化
      (LAMBDA (OBJ)
        (! OBJ 'SET 'ＳＱＡ予約量
           (! '仮想記憶マップ 'REF 'ＳＱＡ予約量)) T)
    ＣＳＡ予約量の縮小
      (LAMBDA (OBJ)
        (COND ((< (/ (! OBJ 'REF 'ＣＳＡ使用量)
                     (! OBJ 'REF 'ＣＳＡ予約量))
                  (! '評価基準値 'REF 'ＣＳＡ使用率))
               (! OBJ 'SET 'ＳＱＡ予約量
                  (/ (! OBJ 'REF 'ＣＳＡ使用量)
                     (! '評価基準値 'REF 'ＣＳＡ使用率)))
               T)))
    ＣＳＡ予約量の初期化
      (LAMBDA (OBJ)
        (! OBJ 'SET 'ＣＳＡ予約量
           (! '仮想記憶マップ 'REF 'ＣＳＡ予約量)) T)
```

図5.　仮想記憶マップと評価基準値のフレーム

```
( オブジェクト定義　判定
  システム
    レベル　クラス
    下層　判定結果
  メソッド
    空間固有域不足の対策　ルール集合１
)

( オブジェクト定義　判定結果
  システム
    レベル　インスタンス
    上層　判定
  属性
    問題解決                          SINGLE *
    ＦＢＬＤＬをＰＢＬＤＬにする    SINGLE *
    ＦＢＬＤＬの移動で解決           SINGLE *
    ＦＢＬＤＬの移動で元に戻る       SINGLE *
    ＦＢＬＤＬの量だけ空きがある     SINGLE *
    ＦＬＰＡの移動で解決             SINGLE *
    ＦＬＰＡの移動で元に戻る         SINGLE *
    ＦＬＰＡの量だけ空きがある       SINGLE *
    ＳＱＡ予約量を縮小できる         SINGLE *
    ＣＳＡ予約量を縮小できる         SINGLE *
)

( ルール集合定義　ルール集合１　BACKWARD
  実行戦略　SINGLE　GOAL '問題解決
  前処理
    <EXECUTE>
      EXPRESSION　(! '仮想記憶マップ　'NEW '新マップ)
  後処理
    <EXECUTE>
      EXPRESSION　(IF (! '判定結果　'REF '問題解決)
                      T
                      (! '新マップ 'DESTROY))
  ルール
    ルール名　ルール１
      もし
        (OR
          (!　判定　'REFBC 'ＦＢＬＤＬの移動で解決)
          (!　判定　'REFBC 'ＦＬＰＡの移動で解決))
      ならば
      <CHAIN>
        PROCESS SET
        OBJECT　判定
        SLOTS　問題解決 = T

    ルール名　ルール２
      もし
        (AND
          (!　判定　'REFBC 'ＦＢＬＤＬの量だけ空きがある)
          (!　判定　'REFBC 'ＦＢＬＤＬの移動で元に戻る))
      ならば
      <CHAIN>
        PROCESS SET
        OBJECT　判定
        SLOTS　ＦＢＬＤＬの移動で解決 = T

    ルール名　ルール３
      もし
        (AND
          (!　判定　'REFBC 'ＦＬＰＡの量だけ空きがある)
          (!　判定　'REFBC 'ＦＬＰＡの移動で元に戻る))
      ならば
      <CHAIN>
        PROCESS SET
        OBJECT　判定
        SLOTS　ＦＬＰＡの移動で解決 = T
      <EXECUTE>
        EXPRESSION　(! '新マップ 'ＦＬＰＡのＭＬＰＡ化)

    ルール名　ルール４
      もし
        (AND
          (!　判定　'REFBC 'ＦＢＬＤＬをＰＢＬＤＬにする)
          (>　(+　(! '新マップ　'REF '核)
                  (! '新マップ　'REF 'ＦＬＰＡ)
                  (! '新マップ　'REF 'ＦＢＬＤＬ)) 1)
            (<= (+　(! '新マップ　'REF '核)
                    (! '新マップ　'REF 'ＦＬＰＡ)) 1))
      ならば
      <CHAIN>
        PROCESS SET
        OBJECT　判定
        SLOTS　ＦＢＬＤＬの移動で元に戻る = T

    ルール名　ルール５
      もし
        (OR
          (>　(! '新マップ　'REF '高アドレス側の空き '(N))
              (! '新マップ　'REF 'ＦＢＬＤＬ))
          (AND
            (!　判定　'REFBC 'ＳＱＡ予約量を縮小できる)
            (>　(! '新マップ　'REF '高アドレス側の空き '(N))
                (! '新マップ　'REF 'ＦＢＬＤＬ)))
          (AND
            (!　判定　'REFBC 'ＣＳＡ予約量を縮小できる)
            (>　(! '新マップ　'REF '高アドレス側の空き '(N))
                (! '新マップ　'REF 'ＦＢＬＤＬ)))
        )
      ならば
      <CHAIN>
        PROCESS SET
        OBJECT　判定
        SLOTS　ＦＢＬＤＬの量だけ空きがある = T

    ルール名　ルール６
      もし
        (AND
          (>　(! '新マップ　'REF 'ＦＬＰＡ) 0)
          (>　(+　(! '新マップ　'REF '核)
                  (! '新マップ　'REF 'ＦＬＰＡ)
                  (! '新マップ　'REF 'ＦＢＬＤＬ)) 1)
            (<= (+　(! '新マップ　'REF '核)
                    (! '新マップ　'REF 'ＦＢＬＤＬ)) 1))
      ならば
      <CHAIN>
        PROCESS SET
        OBJECT　判定
        SLOTS　ＦＬＰＡの移動で元に戻る = T

    ルール名　ルール７
      もし
        (OR
          (>　(! '新マップ　'REF '高アドレス側の空き '(N))
              (! '新マップ　'REF 'ＦＬＰＡ))
          (AND
            (!　判定　'REFBC 'ＳＱＡ予約量を縮小できる)
            (>　(! '新マップ　'REF '高アドレス側の空き '(N))
                (! '新マップ　'REF 'ＦＬＰＡ)))
          (AND
            (!　判定　'REFBC 'ＣＳＡ予約量を縮小できる)
            (>　(! '新マップ　'REF '高アドレス側の空き '(N))
                (! '新マップ　'REF 'ＦＬＰＡ)))
        )
      ならば
      <CHAIN>
        PROCESS SET
        OBJECT　判定
        SLOTS　ＦＬＰＡの量だけ空きがある = T

    ルール名　ルール８
      もし
        (! '新マップ 'ＦＢＬＤＬのＰＢＬＤＬ化)
      ならば
      <CHAIN>
        PROCESS SET
        OBJECT　判定
        SLOTS　ＦＢＬＤＬをＰＢＬＤＬにする = T

    ルール名　ルール９
      もし
        (! '新マップ 'ＳＱＡ予約量の縮小)
      ならば
      <CHAIN>
        PROCESS SET
        OBJECT　判定
        SLOTS　ＳＱＡ予約量を縮小できる = T

    ルール名　ルール１０
      もし
        (! '新マップ 'ＣＳＡ予約量の縮小)
      ならば
      <CHAIN>
        PROCESS SET
        OBJECT　判定
        SLOTS　ＣＳＡ予約量を縮小できる = T
)

(!　' 判定結果 '空間固有域不足の対策 '問題解決)
```

図6.　対策立案のための後ろ向き推論