システムソフトウェアと
オペレーティング・システム 70−11
（１９９５．８．２３）

ＲＴＳのメモリ使用の分散的分析手法と実施例

杉村　康
yas@shownews.cipl.cae.ntt.jp
ＮＴＴ情報通信研究所
〒238-03 横須賀市武１－２３５６

オンライン中に動作するプログラム等の殆どがメモリ常駐であるリアルタイムシステム（ＲＴＳ）では，そのメモリ使用の最適化のために，メモリ使用の詳細分析の実現が重要である。しかしながら，従来手法では，カーネルからアプリケーションプログラムまでの全てのメモリ使用の詳細分析の実現には至っていない。そのために，そのメモリ使用量が予想を越えているときの原因究明が困難であるだけでなく，分析範囲の網羅性と分析結果の正当性を確認できないという問題があった。本論文では，オンライン時にメモリ使用の最低限の情報を蓄積し，そのダンプ情報をオフライン処理により分散的に解析し，オペレーティングシステムの統計情報が提供する全メモリ使用量と当手法の分析結果との一致を確認することにより分析の正当性と分析範囲の網羅性を検証する手法，メモリ使用を大まかな用途毎等に徐々に細分して全ての使用主等を見い出す手法，およびそれらの効果の例を明らかにする。

METHODS FOR DISTRIBUTED ANALYSYS OF MEMORY USED IN RTS AND EXAMPLES OF THE EFECT

Yasushi Sugimura

NTT Network Information Systems Laboratories,
1-2356 Take Yokosuka City, Kanagawa, 238-03 Japan

To optimize the memory used in an Real Time System (RTS) whose programs and data are resident in the memory it is important to analyse the memory used in detail. However, the detailed analysis of all the memory used is not possible with previous methods. Therefore, it is not only difficult to find the cause when the amount of memory used is beyond expecta tion, but also impossible to verify the correctness of the analysis. This paper will present a method for verifying the correctness of the analysis and a method for finding all users of the memory. Examples of the effect of these methods will also be presented.

1. はじめに

オンライン中に動作するプログラム等の殆どがメモリ常駐であるリアルタイムシステム（ＲＴＳ）では，そのメモリ使用の最適化のために，メモリ使用の詳細分析が重要である。 しかしながら従来手法ではカーネルからアプリケーションプログラムまでの全てのメモリ使用の詳細分析の実現に至っていない。 そのために，メモリ使用量が予想を越えているときの原因究明が困難であるだけでなく，分析範囲の網羅性と分析結果の正当性を確認できないという問題があった。

本論文ではオンライン時にメモリ使用の最低限の情報を蓄積し，そのダンプ情報をオフライン処理により分散的に解析し，オペレーティングシステムの統計情報が提供する全メモリ使用量と当手法の分析結果との一致を確認することにより，分析の正当性と分析範囲の網羅性を検証する手法，メモリ使用を大まかな用途毎等に徐々に細分して全ての使用主等を見い出す手法，およびその効果例を明らかにする。

2. 背景

評価対象がＯＳ（オペレーティングシステム）のカーネルからＡＰ（アプリケーションプログラム）までの全プログラムの階層からなる場合，その実稼働時の実装メモリ量を決定するには，各階層のプログラムの予定メモリ使用量をプログラムステップやデータ量等より概算し，実測によりそれらが達成可能かを確認する。 もし実測結果が予定値を大きく越えるときにはメモリ増設によるコストアップを甘受するか，メモリ使用量の削減を行なう必要がある。 近年ＬＳＩの規模増大によりメモリ増設単位は大きい（例えば４MB）ので，実稼働台数が多数である場合はコスト面からメモリ増設が許容されない。 よって各プログラムの開発過程で各予定値の達成が不可欠であり，予定値を越えるときには原因追求と対策実施のためにメモリ使用の詳細分析が必須となる。

しかしながら従来の分析手法では，いずれもカーネルからＡＰ迄の全体の解析には至っておらず[1][2]，また分析範囲の網羅性に関する検証が見られない[3]

そのために分析結果が示す全メモリ使用量とＯＳが提供する統計情報中の全メモリ使用量との間に隔たりが存在し，分析範囲の網羅性と分析結果の正当性を確認できないだけでなく，全メモリ使用量が予定と異なる場合の原因の究明が困難であるという問題があった。

本論文においては，以下の手法により上記の問題点を解決する。

・メモリ使用中の全てのページの数をカウントして「分析対象の全メモリ使用量（以下全分析メモリ使用量と略記）」を求め，それと「ＯＳが提供する統計情報の全メモリ使用量」との一致の確認により分析範囲の網羅性を検証する。

・上記全分析メモリ使用量を大まかな用途毎のメモリ管理単位から細かなメモリ管理単位へと細分し，最終的にメモリ確保単位毎の全てのメモリ使用量とその使用主とその用途を明らかにすることにより，メモリ使用の概要の把握と全メモリ使用量が予定と異なる場合の詳細な原因究明の両者を可能とする。

・上記の細分において細分化の前後のメモリ使用量の対応関係を明らかにし，それらの一致の確認により，分析結果の正当性を検証する。

更に現在実稼中であるトランザクション型のリアルタイムシステムにおける分析例により，それらの手法が有効であることを示す。

但し，当分析手法の対象は，リアルタイムシステムを実現する小型計算機システムであることを想定しており，前提条件は以下のとおりである。

・システムの処理能力を最大とするためにオンライン中に必要なデータやプログラムの殆どはメモリ上に存在する[4][5]。 その場合オンライン中のメモリ使用量の変動は極めて僅かである。 例えばタイムシェアリングシステムのようにメモリ使用量の変動が大きいシステムは本論文の適用範囲外である。

・システムのＣＰＵはアドレス変換機構（論理アドレスを物理アドレスへ変換する機構[6][7][8]）を有し，ＯＳはメモリ使用の有無をページング[6][9]（詳細下記４．１参照）により管理する。 該変換機構を持たないシステムは本論文の適用範囲外である。

3. 情報収集手法

情報収集は，予め分析用の情報をＯＳの内部情報（以下内部情報と略記）内に格納しておき，ＯＳ提供の

ダンプコマンドを起動して論理アドレス空間の情報（ダンプ情報）を二次記憶媒体上に格納することにより行う。（図１参照） 該コマンドは一般に障害解析用に用意されている既存機能[3]である。 更にダンプ情報から「内部情報が存在する論理アドレス空間の内容」のみを同図中の内容ファイルに抽出し，その内容をページ毎に区切り，各ページ毎にファイル先頭からのオフセットとその先頭論理アドレスを同図中のアドレスファイルに書き出し（同図中のPMEMx），直後の分析処理（同図中のPMEMt)でそれらを活用する。

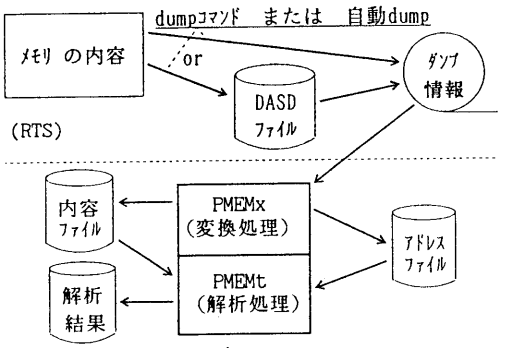

図１： 当手法の流れ

４． 分析手法

分析手法は，大別して以下の三つの手法よりなる。

- ＯＳ統計情報との照合手法
- メモリ使用詳細分析手法
- 分析結果の正当性の確認手法

これらの手法ではＯＳがページングを行うことを前提とするので，その特性を下記４．１で述べる。 またＯＳ統計情報との照合手法を下記４．２で述べる。メモリ使用詳細分析手法と分析結果の正当性の確認手法はメモリ使用の用途により大きく異なるので，その用途毎にまとめて下記４．３で述べる。

４．１ ＯＳのページングの特性

ページングを行うＯＳでは，メモリ使用の最小単位はページ（通常4096バイト）であり，ページには論理アドレス等により決定される論理ページと，実際のメモリ上のページを表す物理ページとがある。

ＯＳは，論理ページの１ページ毎に図２のように，メモリ使用の有無を示す情報と，メモリ使用有のときの物理ページ番号の情報とをＰＴ（ページテーブル）という内部情報の中の一エントリ（以下ＰＴＥと略記）内に保持する。 論理ページの集合は，より大きい単位であるセグメント[6][9]（セクション[7]または空間[8]ともいう）に属す。 セグメントは，全ての論理アドレス空間（論理アドレスを介して利用できる記憶領域）をＭ個（後述の評価例ではM=16で均等）に分割した単位であり，０からＭ－１のセグメント番号で区別される。 セグメント番号は，論理アドレスより一意に求めることができる。

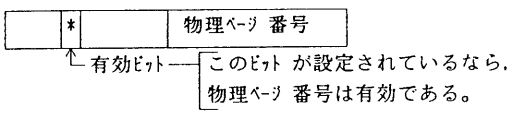

図２：ページテーブルの一エントリの形式の例

４．２ ＯＳ統計情報との照合手法

当手法は，下記（１）でＯＳ統計情報の全メモリ使用量を得，下記（２）で全分析メモリ使用量を得，それらの一致の確認により分析範囲の網羅性を検証する。

（１）ＯＳ統計情報の全メモリ使用量（M_{USE-OS}）の獲得手法

一般にＯＳは資源の使用状況を通知する統計情報用システムコール[10]を提供する。 該システムコールが与えるメモリ関連情報は，ＳＧ（システムジェネレーション）時に決まる実装メモリ量（M_{IMP}）と，メモリ確保・開放システムコール等の実行時にカウントダウン・カウントアップされる空メモリ量（M_{SPA}）である。 それらを予め内部情報内に格納しておき，下式によってM_{USE-OS}を得る。

$$M_{USE-OS} = M_{IMP} - M_{SPA} \qquad (1)$$

（２）全分析メモリ使用量（M_{USE-AN}）の獲得手法

一般にページは，その使用形態より下記(A)～(C)の空間クラスのいずれかに属す[10]。

(A) システム空間： 全タスクから共通にアクセス可能な空間（当空間のＰＴは全タスクで同一である。論理アドレスより一意に対応のＰＴＥが決まる。）

(B) タスク固有空間： 自タスクからのみアクセス可能な空間（当空間のＰＴはタスク毎に異なり論理アドレスとタスクにより一意に対応のＰＴＥが決まる。）

(C) 共用空間： 一部のタスク群からアクセス可能

な空間（当空間のＰＴはタスク間で異なる場合と同一の場合があり，論理アドレスとタスクにより一意に対応のＰＴＥが決まる。）

今，メモリ使用中のＰＴＥに限って，任意の一タスクのシステム空間のＰＴＥの数をカウントしてシステム空間の使用ページ数の合計（P_{SYS}）を得，全タスクのタスク固有空間と共用空間の全ＰＴＥの数をカウントしてそれらの論理アドレス空間の使用ページ数の合計（ΣP_{TAS}）を得，共用空間によりΣP_{TAS}内に多重にカウントされる物理ページのページ数をP_{MUL}とすると，M_{USE-AN}は下式により得ることができる。

$$M_{USE-AN} = P_{SYS} + \Sigma P_{TAS} - P_{MUL} \quad (2)$$

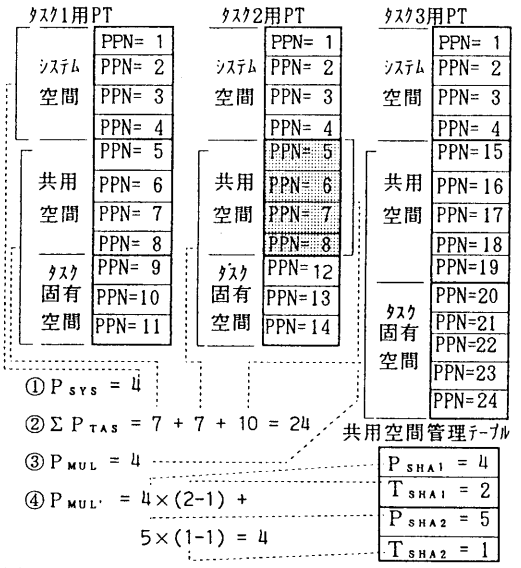

図３：ΣP_{TAS}により多重カウントされるページ数と共用ページ数とタスク数の関係の概念例

例えば図３のように，タスクは3 個，ＰＴＥは全てメモリ使用中であり，ＰＴＥ中の物理ページ番号はＰＰＮ＝ｎで表すと，P_{SYS}（任意の一タスクのシステム空間のＰＴＥの数）＝4 （同図中①），ΣP_{TAS}（各タスク毎のタスク固有空間と共用空間の全ＰＴＥの数）＝24（同図中②），P_{MUL}〔共用空間の全ＰＴＥのうち重複した物理ページ番号を持つＰＴＥ（同図網掛部分）の数〕＝4 （同図中③）である。

当手法では物理ページ番号の重複の判定（分析処理時間が長い）処理を避けるためにP_{MUL}は「各共用単位の使用ページ数（P_{SHA}）」と「各共用単位を共用しているタスク数（T_{SHA}）」を予め内部情報内に設定しておき次式より$P_{MUL'}$として求める。（図中④）

$$P_{MUL'} = \sum_{I=1}^{N} (P_{SHAI} \times (T_{SHAI} - 1)) \quad (3)$$

（但し，式中のＮは共用単位の数である。）

４．３　メモリ使用詳細分析と正当性確認手法

当手法では，先ず下記４．３．１で，上記で求めたM_{USE-AN}を大まかな用途毎の管理単位である表１に示すエリアクラス毎に分解する。

これらの各エリアクラスは，同表に示すように，プログラムの作業域として使用されるトランジェント域（同表項番６～項番10）と，そうではないノントランジェント域（同表項番１～項番５）に大別でき，更にトランジェント域は，その獲得時の属性の違いにより，表2 に示す獲得クラスに大別できる。

以下４．３．２にノントランジェント域の分析手法を，４．３．３にトランジェント域の分析手法を，また，それらで共通に使用される「タスクＩＤ（タスク識別子）等より使用主等を特定する手法」を４．３．４タスク分析手法で述べる。

表１　エリアクラスの種類の概念と空間クラスの例

項番	分類	エリアクラス	説明　（空間ｸﾗｽ: SY = ｼｽﾃﾑ空間，SH= 共用空間，TO:ﾀｽｸ 固有空間）
1	ノントランジェント域	基本OS ﾚｼﾞﾃﾞﾝﾄ	基本OS用の常駐プログラム，内部情報，非常駐システムコールの格納域。（SY）
2		拡張OS ﾚｼﾞﾃﾞﾝﾄ	拡張OS用の常駐プログラム，内部情報，非常駐システムコールの格納域。（SY）
3		ﾌﾟﾛｸﾞﾗﾑ ﾃｷｽﾄ	APやミドルソフトウエアのプログラムのテキストの格納域。（SH）
4		ﾌﾟﾛｸﾞﾗﾑ ﾃﾞｰﾀ	APやミドルソフトウエアのプログラムのデータの格納域。（TO）
5		ｽﾀｯｸ	プログラムの変数等の格納域。（TO）
6	トランジェント域	基本OS1	基本OSが使用する作業域。（SY）
7		拡張OS1	拡張OSが主に使用する作業域。（SY）
8		拡張OS2	拡張OSが主に使用する作業域。（TO）
9		ﾕｰｻﾞ 1	AP等が使用する作業域。（SY）
10		ﾕｰｻﾞ 2	AP等が使用する作業域。（TO）

表2　獲得クラスの種類と概念

項番	獲得ｸﾗｽ	説明
1	保証域	該域の管理元が予め必要な最大ﾒﾓﾘ 使用量と切出し単位を宣言（ﾌﾞｰﾙ の生成[11]）し，その時PTE と物理ﾍﾟｰｼﾞ を確保。
2	準保証域	該域の管理元が予め必要な最大ﾒﾓﾘ 使用量と切出し単位を宣言し，その時PTE を確保。
3	自由域	上記の宣言は無く，ﾒﾓﾘ 確保のｼｽﾃﾑｺｰﾙ の発行時にPTE の作成と物理ﾍﾟｰｼﾞ の確保を行うｴﾘｱ。　当ｴﾘｱ は予め大きさが限定できない比較的少量の作業域の場合の使用に限定することによりｵﾝﾗｲﾝ 中の処理能力低下を防止。

4．3．1　セグメントとエリアクラス等の構成とエリアクラス分析手法

当手法では，上記で求めたMUSE-ANを大まかな用途毎の管理単位のメモリ使用量に分解する。　その分解を容易にするために，予め各管理単位の構成に以下の工夫を行う。

・セグメントは，極力表1中の一つエリアクラスしか含まないように構成する。

・共用空間は一つのセグメントのみで構成する。

なお，セグメントが複数のエリアクラスを含む場合は，それらの分離のための情報が必要であり，当評価例では，セグメント数の節約のために図4のように，セグメント0のみが二つのエリアクラスを含み，それらの分離用情報として基本ＯＳトランジェント域の先頭論理ページ番号*1と同域の論理ページ数*2を予め内部情報内に設ける。　また，共用空間の使用は表1項番3のプログラムテキスト域のみとし，共用単位はプログラムテキストのプログラム名とする。

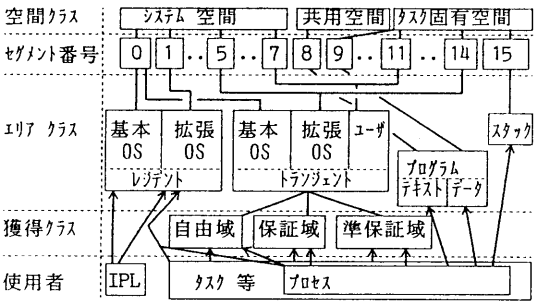

図4：　メモリ体系の概念例

以上の構成により，以下の手法で，エリアクラス毎のメモリ使用量を分析する。

・上記4．2（2）全分析メモリ使用量（MUSE-AN）の獲得手法のＰＴＥのサーチ時に，論理アドレス空間をＭ個に分割して，各ＰＴＥのセグメント番号毎のカウントを行う。　その際に，共用空間以外のセグメントでは，該セグメントのメモリ使用中のＰＴＥの数をカウントした結果をセグメント分析結果（PSEGJ；J=セグメント番号）とする。　共用空間のセグメントでは，該セグメントのメモリ使用中のＰＴＥの数をカウントした結果からPMULを差し引いたものをセグメント分析結果とする。　次式の成立確認により当分析結果の正当性を確認する。

$$M_{USE-AN} = \sum_{J=0}^{M-1} P_{SEGJ} \qquad (4)$$

またセグメント0では，メモリ使用中のＰＴＥをカウントする際に，上記*1と*2を参照して，基本ＯＳトランジェント域と基本ＯＳレジデント域に分けて使用ページ数をカウントして，それぞれをサブセグメント分析結果（PSUBk：k=サブセグメント番号）とし，それらの結果の合計とセグメント0のセグメント分析結果との一致確認により，サブセグメント分析結果の正当性を確認する。

この場合セグメント0以外のセグメント分析とサブセグメント分析は各エリアクラスのメモリ使用量の分析と等価である。　それらを以下まとめてエリアクラス分析と呼び，それらの分析結果をエリアクラス分析結果と呼ぶ。　エリアクラスは大まかな用途を規定するので，エリアクラス分析結果より，メモリ使用量の内訳の概要を知ることができる。

4．3．2　ノントランジェント域の分析手法

（1）基本ＯＳレジデント域の分析手法

当域は一般に各プログラムから共通に参照されシステム空間内にある。　当域では各プログラムやデータの論理アドレスの重複を回避するために論理ページ番号毎に予め使用主と用途を登録制とする。　上記エリアクラス分析時に，当域の「(A) メモリ使用中である連続した論理ページの先頭論理アドレス」と「(B) 上

記(A) の連続使用ページ数」の一覧を得，それらと事前登録情報との照合により使用主と用途と使用主毎のメモリ使用量を確定する。　また上記(B) の合計と当域のエリアクラス分析結果との一致の確認により，上記(A) と(B) の正当性確認を行う。

（2）拡張ＯＳレジデント域の分析手法

当域の分析は，以下に述べる点を除いて上記（1）基本ＯＳレジデント域の分析手法と同様である。

・セグメント番号が異なる。

・「拡張ＯＳのサービス初期設定時にメモリ常駐化したシステムコール」についてシステムコール番号毎に予め内部情報内に表3の情報を格納しておき，該番号と該名称と使用ページ数の一覧を分析結果とする。

表3　メモリ常駐化したシステムコールの分析のためのシステムコール番号毎のＯＳの内部情報

項番	内部情報の詳細
1	非常駐システムコールを示すフラグ
2	メモリ上にロードしたことを示すフラグ
3	使用ページ数
4	システムコール名

（3）プログラムテキスト域の分析手法

当域は共用空間内にある。　当域の用途はＡＰ等のプログラムテキストである。　プログラムテキストは複数のプロセス（タスクの一形態でありタスクに種々の資源を追加した処理単位[11]）により共用され得る。

プロセスのタスクＩＤから使用主（プログラム名）を確定する（詳細4.3.4 参照）。　一プロセスの当域の使用ページ数は当域のセグメントの「そのタスクのメモリ使用中のＰＴＥの数のカウント」により得る。

その結果を「互いに異なるプログラム名を持つプロセス」についてのみ合計したものをP_{TEX} とし，下式の成立確認により上記分析の正当性確認を行う。

$$P_{TEX} = P_{SEG_P} - P_{MUL} \quad (5)$$

（P_{SEG_P}の$_P$は，当域のセグメント番号である。）

（4）プログラムデータ域とスタック域の分析手法

一般にプログラムデータ域とスタック域は，タスク固有空間内にある。　一つのプロセスの該域のメモリ使用量の分析は，該域のセグメントの「そのタスクのメモリ使用中のＰＴＥの数のカウント」により行い，エリアクラス分析結果と同じである。　使用主の確定手法は上記（3）と同様である。　但しスタック域は上記で確定したプログラムからだけでなく該プログラムの内部発行のシステムコールから使用される場合がある。　その場合使用主はそのシステムコールのプログラムであるが，各システムコール毎の使用スタックの量は，別途そのロードモジュールの解析等により予め把握が可能であり，且つ通常一つのプロセスのスタック域の全使用量が数ページを越えることは殆どないので本手法では更なる分析は割愛した。

4．3．3　トランジェント域の分析手法

トランジェント域には表1のように5種類があり，その論理アドレスよりどのエリアクラスの使用であるかを判別できる。　以下エリアクラス毎の獲得クラス毎に，それらの使用主の分析手法を述べる。

（1）保証域の分析手法

当域では，当域の各宣言時に各宣言毎のメモリ使用量が決定する。　各宣言時毎に表4の情報を個別の内部情報内に作成する。　この内部情報をプール情報と呼ぶ。　個々のプール情報内の先頭論理アドレス（表4項番2）よりセグメント番号とエリアクラスを，同最大バイト数（同項番3）をページ換算したものよりメモリ使用量を，同タスクＩＤ（同項番1）から管理元（宣言を行ったプログラム）を得る（詳細4.3.4 参照）。　なお上記の使用率は各宣言毎のメモリ使用量のうち実際に使用されている割合を表す。　それらの分析結果と下記（2）と（3）の分析結果のメモリ使用量のエリアクラス毎の合計と，対応する上記エリアクラス分析結果との一致の確認により，当分析の正当

表4　保証域と準保証域の使用主等の確定のためのＯＳの内部情報

項番	内部情報の詳細
1	当域を宣言したタスクＩＤ
2	当域の先頭論理アドレス
3	当域の最大バイト数
4	当域の切りだし最小単位のバイト数
5	当域の使用率

性を確認する。 更に必要に応じて，各管理元や最大バイト数や切りだし最小単位のバイト数より，該管理元の製造担当者が用途の詳細を特定する。

（2）準保証域の分析手法

当域では上記（1）と同様に当域の各宣言を行うが，各メモリ確保・開放のシステムコールの発行時にメモリ使用量が決定する。 当域の分析方法は，以下を除いて（1）と同様である。

・最大バイト数（表4項番3）と使用率（表4項番5）の乗算結果のページ換算値よりメモリ使用量の概算値を得る。 この概算値は，確保と開放を繰り返す場合は不連続なバイトの確保となるので（実際のメモリ使用量に等しいかまたは小さくなり）誤差を含み得るため，正当性の確認には利用できないので以下の手法により当分析の正当性の確認等を行う。

・当域の各宣言の実行時におけるＰＴＥ作成時に，ＰＴＥ内の特別なビット（仮割当ビットと呼ぶ）をONとする。 上記エリアクラス分析時に，該ビットがONでメモリ使用中のＰＴＥの数をカウントして，当域のエリアクラス毎のメモリ使用量（P_{SEM_q}: q=エリアクラスのセグメント番号）を得る。 また，該ビットがONでメモリ使用中でないＰＴＥの数をエリアクラス毎にカウントしてR_{SEM_q}とし，$P_{SEM_q}+R_{SEM_q}$と「上記の各最大バイト数のページ換算値のエリアクラス毎の合計」との一致の確認により，当域の分析結果の正当性を確認する。

（3）自由域の分析手法

当域では上記（1）や（2）で述べた宣言は存在しない。 分析に必要な情報はメモリ確保・開放のシステムコールの実行毎に，表5に示す情報をメモリ使用履歴域（以下履歴域と略記）という内部情報に，時系列順に格納する。 履歴域はｍ個のエントリｉ（ｉ＝0～m-1）からなり，一つのエントリｉが一つのメモリ確保・開放システムコールの情報を保持する。 ｍは全ての自由域に関する情報を格納するために十分な数とする。 解析は，履歴域内の全ての有効な情報をサーチして「メモリ確保が行われたがメモリ開放が行われていない情報」をメモリ使用中の情報として抽出を行い，その後，抽出されたメモリ使用中の情報内の先頭論理アドレス（表5項番2）よりセグメント番号とエリアクラスを，同確保・開放ページ数（同項番3）よりメモリ使用量を得る。 以上の分析結果と上記（1）と（2）の分析結果のそれぞれのメモリ使用量のエリアクラス毎の合計が対応するエリアクラス分析結果と一致することの確認により上記分析結果の正当性を確認する。 また抽出されたメモリ使用中の情報内のシステムコールの発行元論理アドレス〔とそのアドレスがシステム空間でない場合は更にＴＣＢ（タスク制御ブロック）アドレス（詳細4.3.4 参照）〕から使用主を特定し，それらとプログラムリスト等の照合により用途を明らかにする。

表5 自由域の確保・開放のシステムコールの発行毎にメモリ使用履歴域に格納されるＯＳの内部情報

項番	内部情報の詳細
1	確保・開放のいずれであるかを示すフラグ
2	確保・開放域の先頭論理アドレス
3	確保・開放ページ数
4	確保・開放システムコールの発行時のＴＣＢ（タスク制御ブロック）の論理アドレス
5	確保・開放システムコールの発行元論理アドレス

4．3．4 タスク分析手法

一般にタスクＩＤ（とそれと一意に対応するＴＣＢアドレス、以下省略）は，生成されたタスクに付与されるＯＳ内で一意なＩＤであるが，タスクによって実行されるプログラム名と直接的な関係を持たないので以下により，その関係を特定する。

（A）プロセスを形成しないタスクは，システム空間上で実行させることにより，そのタスクによって実行されるプログラムのスタート論理アドレスをＴＣＢ内に格納しておき，それと事前登録情報の照合によりプログラム名を確定する。

（B）プロセスを形成するタスクでは，プロセスの生成時にプログラム名とタスクＩＤを内部情報内に格納しておくことにより，プログラム名を特定する。

5． 評価例

5．1 前提条件の確認

当手法においては上記1．で述べたように「オンライン中にはメモリ使用量は時間的に殆ど変動しない」

ことを前提とした。 この確認を，文献[5]でモデルとした現在実稼働中のトランザクション型のリアルタイムシステムにおいて，ＡＰのサービス初期設定直後と15万件／時の負荷を掛けたときのそれぞれの分析結果の比較により実施した。 その結果，システム全体でスタック域が10ページ，その他７ページ，合計17ページ（全領域の0.4%）の増加のみであり，前提条件がほぼ満足されることを確認した。

5．2 当手法の補助のためのプログラムツール（PMEM）のメモリ使用分析工数削減効果

当分析手法のうち，登録情報との照合等を除く部分をプログラムツールとして実現した。 同様のことをダンプ情報をもとに専門家の目視により実施した結果は１分析当たり７日強であり，当ツールでは専門家なしで数時間のマシン走行（20分の１未満の工数）で，正確に行うことができた。

5．3 実ＯＳの分析での成果例

（１）当手法を，上記４．１で述べた現在実稼働中のＲＴＳのＯＳ開発初期の段階および実際のＡＰを搭載した段階で適用し，メモリ使用のチューニング等を実施した結果，前者においてＯＳのメモリ使用の３０％の削減，後者においてシステム全体のメモリ使用の１３％の削減を実現した。

（２）上記４．１の前提条件の確認の検討において，そこで述べた17ページ以外に，基本ＯＳトランジェント域の自由域６ページとユーザトランジェント域の自由域20ページのメモリ増加を検出した。 それらの使用主の分析結果より，上記の原因が図１のＯＳ提供のダンプコマンドのメモリ返却ミス（バグ）であることを指摘し，当手法が文献[12]の『メモリ返却し忘れ』の検出にも有効なことを実証した。

（３）当手法を上記４．１で述べたＲＴＳの次期ＯＳにおいて適用中に，マニュアル変更無しにＯＳのメモリ関連の内部情報が変更になったことを当手法の正当性確認により指摘し，マニュアル変更を実施させ，当手法の正当性確認が有効であることを実証した。

6． むすび

本論文では，ＯＳの統計情報が提供する全メモリ使用量と当手法の分析結果との一致等を確認することにより分析の正当性を検証する手法，およびメモリ使用を大まかな用途毎等に徐々に細分して全ての使用主等を見い出す手法，及びその正当性確認例や，メモリの削減の実例，並びに当手法の補助のためのプログラムツールの工数削減効果例を示し，当初の目的を十分果たすことができた。

文 献

(1) 吉本安男，高橋雅宏：オペレーテイングシステムの性能評価と監視効率，情処全大，5V-4，(S61年後期)．

(2) 辻井敦：処理プログラムの仮想メモリ使用量測定方法，情処全大，7P-2（S63年後期）．

(3) 清水和夫：メモリ使用状況分析の一手法，情処全大，6C-3（S60年後期）．

(4) 世羅孝文，他5 名：V60/V70 リアルタイムUNIX，情処全大，3D-4（S62年後期）．

(5) 杉村康：マルチプロセッサソフトオーバヘッド評価におけるロックモデル解析の一手法，信学論(D-I)，J75-D-I ，10，pp.917-925(1992)．

(6) 石田晴久，村田健一：超大型コンピュータシステム，p235，産業図書株式会社（1975）．

(7) 坂村健，榎本龍弥：TRON 仕様に基く32ビット・マイクロプロセッサ，情報処理，Vol.30，No.5，pp.565～573（1989）．

(8) HEWLETT PACKARD:HP3000 and 9000 Series Computers,Precision Architecture & Instruction Set Reference Manual,p.341,HEWLETT PACKARD (1989).

(9) 斉藤信男訳：オペレーティング・システムの基礎，p.360，日本コンピュータ協会（1978）．

(10) 坂村健監修，トロン協会編：原典CTRON 大系２カーネルインタフェース，p.436，オーム社，pp.107，pp.121，pp.122，pp.161，pp.163（1988）．

(11) 坂村健監修，トロン協会編：原典CRTON 大系１CTRON 概説，p174，オーム社，pp.65（1988）．

(12) 吉本安男：仮想メモリの返却忘れ検出機構の試作，情処全大，7P-3（S63年後期）．