2004－EVA－11　(5)
2004／11／4

システム統合・変更に伴うシステム診断の重要性

荻田　光一郎＊、加倉井　宏一＊

アウトソーシングや企業の統合なので複数のシステムを統合、拡張して運用する形態が増加している。一方メインフレームでは機器の性能が向上して単一の筐体で論理分割を行い複数システムを効率よく稼動出来るようになった。この論理分割での稼動形態はスペース、電力などの運用コストを削減できるため複数システムの稼動に際し広く利用されている。複数システムを統合稼動する場合筐体の持つ CPU、ストレージ、IO 等の資源をそれぞれの稼動環境にあわせて最適に分配する必要がある。資源を稼動するシステムの状況に適した分配を行わないとサービスレベルの低下を起こすことがある。すなわちオンラインレスポンスの低下やバッチジョブの遅延などである。そのためにも統合・拡張するシステムの稼動状況を分析し最適に資源を配分する手法が必要となる。本稿ではシステム診断手法を活用して統合に必要なシステム資源を分配・運用する方法を実践し評価している。

The importance of the System Clinic along with the System Integration/Migration

Kohichiroh Ogita*, Kohichi Kakurai*

Because they are outsourcing and the unification of the enterprise,, the form that more than one system is integrated and expanded and employed is on the increase. The performance of the machine improved, and logic division was done in single machine frame. On the other hand learned to work efficiently with the main frame more than one system .Because space, an employment cost such as electric power can be reduced, an operating form by this logic division is being used widely on the occasion of the operation of more than one system. When it works more than one system is integrated, resources such as the CPU which machine body has, storage, IO are fitted to each operating environment, and you must distribute it suitably. The decline of the service level is sometimes caused the resources if the distribution which is suitable for the conditions of a system to work isn't done. In other words, it is in such cases as the decline of online transaction response and the delay of batch jobs. The operating conditions of the system that unification is expanded are analyzed, and the technique which resources are distributed to suitably becomes necessary because of that as well. A system diagnosis technique is made use of, and the way that distribution/ employs the system resources which are necessary for the unification is practiced and evaluated in this paper..

１．はじめに

アウトソーシングによるシステム運用や企業の共同化により、複数拠点や複数システムで稼動していた個別システムを一拠点や、より少ないシステムに統合して運用してゆく形態が増加している。アウトソーシングについていえば、運用拠点が複数あるよりは管理・コスト面で一拠点に集約したほうが有利となる。また稼動する機器も少ないほうが運用・管理がし易いため、一つの筐体内で複数のシステムを稼動させている。アウトソーシングされた企業の複数の個別のシステムを単一拠点に集約して出来るだけ単一筐体で運用するようにしている。また同一企業の枠を超えて複数のアウトソーシングされた別の企業が同一拠点、単一筐体内で運用されているケースもある。

また企業の合併や共同化により、今まで別の拠点で運用されてきた個々の企業システムが共同で運用される単一拠点に統合したり、合併する企業どうしで稼動しているシステムを同一の筐体に統合して運用したりしている。

これらの形態に共通していえることは複数のシステムを単一の筐体に統合して稼動させることにより運用・管理に伴う処理コストを削減できることである。

このような環境を実現するために１筐体で複数の論理的なサーバ環境を実現しているのが、PR/SM(Processor Resource System Manager)による論理分割(LPAR)である。システムの統合前には単一プロセッサーで稼動していた個別のシステムは PR/SM の１区画で稼動するように構成に変更され、実際に稼動・運用されるシステム全体は複数の区画を使用して多くのシステムが稼動するように構成される。

＊日本アイビーエム・システムズエンジニアリング（株）

複数の個別システムを PR/SM により統合稼動させるためには、個々に稼動している個別システムの稼動状況を配慮したシステム資源の適切な配分が不可欠となる。

２．システム統合のための区画分割方法について

２．１区画分割方法の種類と考慮点

システム統合を行うため区画分割方法は３つの方法が提供されていて必要に応じて選択される。

（１）物理分割（フィジカル・パーティショニング）方式：ハードウエアを物理的・電気的な境界で分割する手法。採用される筐体（サーバ）により分割される単位はシステム・ボード、セル・ボードなどさまざまである。区画での独立性が高いためある区画での障害（ハードウエア、ソフトウエア）が他の区画に及ぶ可能性はほとんどないが、区画間でのリソース（CPU,メモリー）の共用は行えない。

（２）論理分割（ロジカル・パーティショニング、PR/SM）：ハードウエアの物理的、電気的な境界に関係なくライセンス内部コード（マイクロコード、ファームウエアなどのハードウエアと OS の中間層で分割を行う方法。物理分割より分割単位が小さくて柔軟であり、ハードウエアに区画やリソースの定義を行う。マイクロコードで分割制御を行っているため、ある区画での障害（ハードウエア、ソフトウエア）が他の区画に及ぼす影響は少なく、区画間でプロセッサー（CP）や IO バスなどのリソースを共用が可能のため共用にあたっては定義が必要となる。マイクロコードで区画管理を行っているのでサーバのリソース（CPU、メモリー）が使用され若干のオーバーヘッドが伴う。分割の数は数十である。

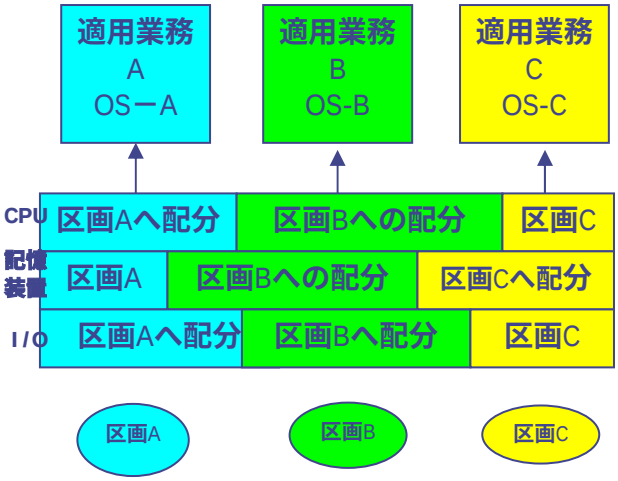
（３）ソフトウエア分割（仮想分割）：システム・ソフトウエアによって複数の OS を稼動させる方法。区画間で CPU、メモリーを含めたリソースの共用が可能であり、リソースのもっとも柔軟かつ効率的な利用ができる。ハードウエア障害がすべての区画に影響を及ぼすことや区画制御を行うシステム・ソフトウエアの障害はサーバ全体の障害となる場合がある。区画管理に使用されるサーバのリソース(CPU、メモリー)や制御のためのオーバーヘッドは他の分割方法にくらべれば大きい。筐体のリソースが十分あれば分割される区画の数に制限はない。

実際にシステム統合に使用する方法には区画の独立性、設定や運用での柔軟性、リソースの共有性、稼動にあたって障害への対応性などを考慮して論理分割（PR/SM）が多く採用されている。

２．２ 論理分割（PR/SM）の特徴と機能

論理分割（PR/SM）は区画の独立性を高くするために筐体の持つハードウエア･リソース(CPU,メモリー、IO）は図１のように区画に配分されます。配分については占有配分される資源と共有配分されるリソースに分けることができる。

図１）論理分割（PR/SM）概念図



リソースの占有、共有について以下に述べる

（１） プロセッサー：占有（物理 CP）と共有(論理 CP)の指定が出来る。ただし１筐体で保有している物理 CP 以上に論理 CP を持つことは出来ない、また一つの区画で占有（物理 CP）と論理（論理 CP）を持つことは出来ない。共有（論理 CP）を持つ区画では重み(Weight)を１－９９９の範囲で指定して相対的なプロセッサー使用率を決定する。最近では Linux のための専用プロセッサー(IFL)を区画に占有させることが出来る。

（２） メモリー：区画ごとに占有となるので区画ごとに使用量（メモリー・サイズ）を設定する必要がある。また設定値は稼動中に区画間で動的に変更することが出来る、但し区画で稼動している OS が対応している必要がある。

（３） IO(チャネル)：：特定の区画のみに占有させる占有チャネル、一時点では１つの区

画からのみ使用可能なチャネルで、運用時に必要であれば他の区画に付け替えることが可能な再構成可能チャネル、複数の区画から同時に使用できる共有チャネルがある。一部のチャネルの種類によっては共有できないタイプがある。

２．３ 論理分割を使用した統合化での課題

筐体が異なる状況で稼動している個別システムに論理分割(PR/SM)を採用して統合稼動する場合、論理分割の特徴（占有、共有など）を考慮した統合化と統合する個別システムの稼動状況を調査・分析してリソースの配分を行う必要がある。

(1) 占有リソースと共有リソースの割り当て

プロセッサー・リソース：プロセッサーには占有の物理 CP と共有の論理 CP がある。プロセッサー・リソースを有効に活用すると言う観点から、実際の統合運用では論理 CP を各区画に必要な数をアサインする。ただプロセッサー資源を保障したい場合（パフォーマンス・テスト）などを行う時に物理 CP を区画に占有する。例えば１０CP のうちある論理区画に３つの物理CPを占有させパフォーマンス・テストを行いデータを取得する。区画に論理 CP を使用する場合には論理CPの数と論理区画ごとに設定される重み（Processing weights）を与える。論理 CP の重みは目標値(能力)の指定であるが、区画に設定するとき Capping を与えると制限値となる。論理 CP の目標は以下の式で与えられる。

目標 CP=重み*N(共有論理 CP)/重みの総和

目標値は筐体の CPU 使用率が上昇した時に CP 資源の配分の目標に近くようになる。

１CP の論理区画は１物理 CP しか使用できないので論理CPの数には注意する必要がある。

メモリー・リソース：メモリーの共有は出来ないので稼動に必要なメモリーを設定する。予め稼動状況を分析して必要なメモリー量を算出しておく必要がある。統合に当たって総メモリー量が不足する場合に備えて統合される個別システムのメモリー使用状況をアドレススペース毎に捉えておくと各区画間でのメモリ設定の調整をし易い。

IO・リソース：各区画で使用する IO（チャネル）は接続されている装置により占有、共用を設定して使用する。

(2) 物理・論理プロセッサーの数の設定

個別システムのプロセッサーの能力の合計と統合するプロセッサーの能力は稼動の余裕を見て統合するプロセッサーの方の能力は大きくするが、技術の進歩により１物理 CP の能力が増加しているため、個別システムでは複数の物理 CP で稼動していたのが統合すると物理 CP の数が減少する状況や統合する筐体の都合により個別システムで稼動していた物理 CP の数が増加してしまう状況が起こりえる。

物理 CP の能力が高くなったため、稼動に際し物理 CP の数が減少すると、図２のように優先度が高く CPU を使用する物が占有して、全体に効率よく CPU リソースが回らない現象が発生する。稼働状況を分析し必要であれば能力が同じでも物理 CP の数は減少させない配慮をする。

図２．物理 CP が少ない場合の動き

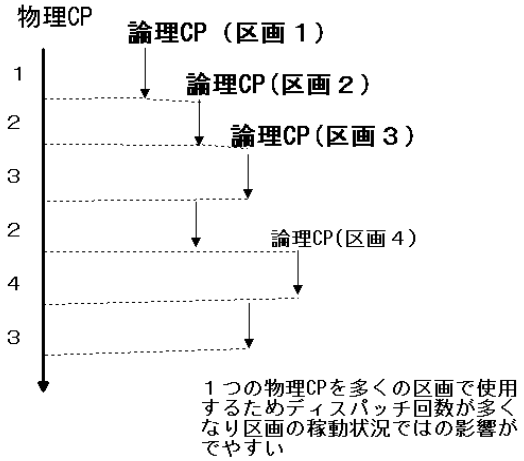
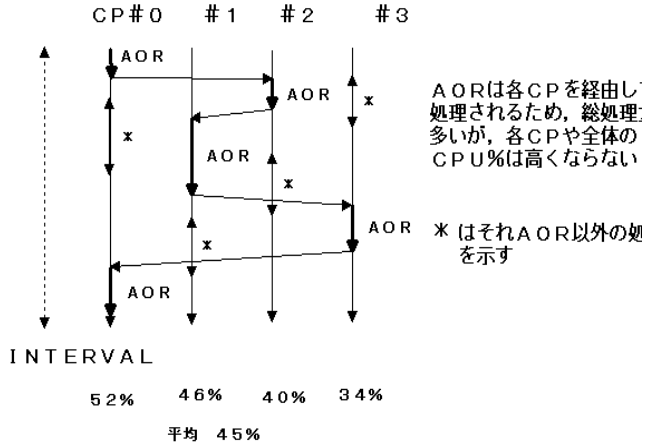


図３．論理 CP が多い場合の動き



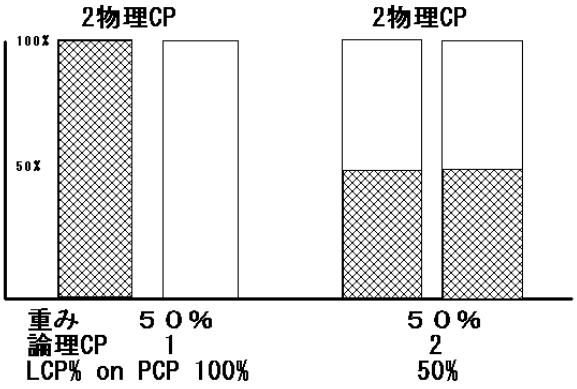
必要以上に論理 CP の数が多いと図２のように、区画の稼動状況により、インターラプションが多い動き（IO 処理など）で CPU のディス

パッチ処理に時間がかかりCPU使用率は高いが効率よく処理されないことが起こる。CPU を多く使用して処理されるタスクがいくつもの論理CP を経過して処理されるからである。

また、図４に示すように物理 CP より論理 CP の数がすくない設定をしている区画では、特定の物理 CP に負担がかかり筐体全体の物理 CP を効率よく使用できなくなる状況が起こる。

図４．物理・論理 CP の数の差による使用状況

論理区画名	論理CP数	重み	CAPPED
LPAR1	1	50%	YES
LPAR2	2	50%	YES



論理区画の相対的な重みは論理 CP 毎に割り振られ、論理 CP 単位に CP 使用制限（Capping）が行われるため物理 CP が１CP の場合を除いて区画への論理 CP の割り当ては２CP 以上にすることが望ましい。

稼動するサブシステムやアプリケーションの性質により論理 CP の数を決めることが大切である。特に複数 CP を効率よく使用している並列稼動処理（DB 処理など）サブシステムが使用されている場合には論理 CP は多く設定する。

3．システム診断によるシステム･リソース分析と配分

統合化をするにあたって区画分割（PR/SM）を使用するためには、個々のシステムのシステム･リソースの使用状況を分析して、使用する区画に最適な占有リソース、共用リソースを配分する必要がある。リソースの配分が不適切な場合、その区画で稼動するサブシステムやアプリケーションに影響を及ぼす。CP 数と重み（Weight）値で同等なプロセッサー･パワーを設定しているにもかかわらず、オンラインレスポンスの低下やバッチジョブの遅延などを引き起こすことがある。システム･リソース（CPU,ﾒﾓﾘｰ、IO）の分析にシステム診断の手法とシステム診断を円滑におこなうために開発した TOOL を使用して効果を得た。システム診断（System Clinic）とは定期的に稼動システムの状況を診断し評価する仕組みである。主にシステム･リソースの使用状況や主要なサブシステムの稼動状況を客観的立場から評価し、問題点や改善点を指摘する。

今回、統合化する個々のシステムについてシステム診断で使用した TOOL を使用してシステム･リソースの使用状況の分析を行い、区画設定のための資料として活用した。

３．１システム・リソース使用状況分析

（１）CPU リソース：稼動しているアドレス・スペースに関して CPU 使用率を取得する。どのようなサブシステムやアプリケーションが CPU 使用率が高いかを調査する。平行処理具合の高いサブシステムについて CPU 使用率を見る。

図５．CPU 使用状況の分析

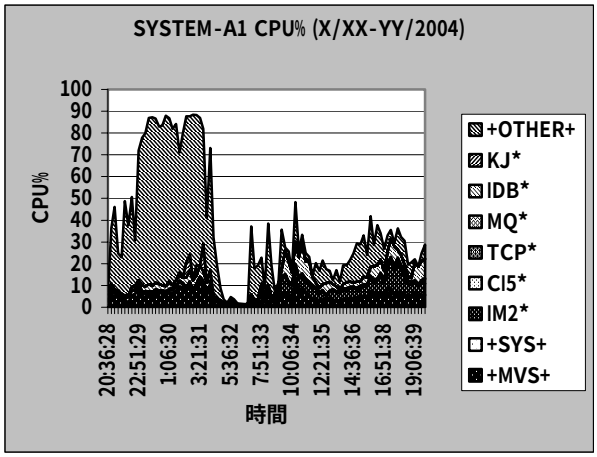
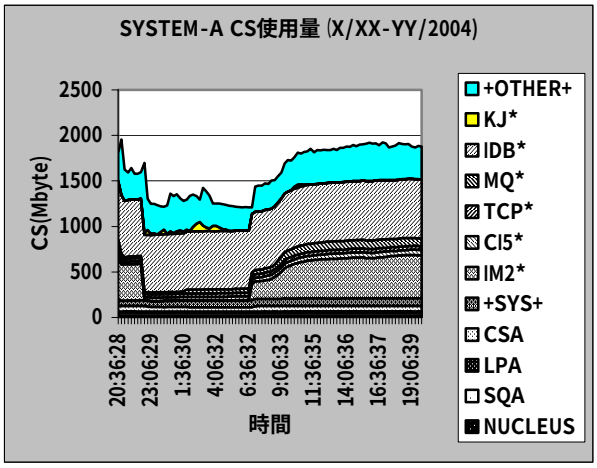


図５は１日のCPU使用率をサブシステム単位で見たものである。夜間の大半はバッチジョブがCPU を使用していることがわかる。昼間の時間帯では各サブシステムがそれぞれCPUを使用しているが夜間には及ばない。特に多く CPU を使用するサブシステムは見られない。このシステムを稼動させるための区画の設定には夜間バッチジョブを円滑に稼動させるためのプロセッサー能力を設定する必要があることがわかる。

（２）ストレージ・リソース：稼動しているアドレス・スペースに関してメモリーの使用状況を取得する。CS(Central Storage) 以外に ES(Expand Storage)を使用している場合は ES の使用状況の調査も行う。

図６は図５と同様に１日のメモリー使用量をサブシステム単位で見たものである。メモリーの使用量が高いのは昼間の時間帯で２つのサブシステムが多くメモリーを使用していることが分かる。CPU 使用率が高かった夜間バッチジョブの稼動する時間帯は逆にメモリーの使用量が減少している。この環境でのバッチジョブは CPU リソースは使用するが、メモリーリソースはそれほど使用していないことが分かる。メモリー・リソースの配分は昼間の各種サブシステムの稼動に十分なサイズの設定を行う。

図６．メモリー使用状況の分析



IO リソースに関しては十分な IO が接続可能であったため特に分析はしていない。

３．２　バッチジョブ稼動状況分析

システム・リソースの分析で夜間のバッチジョブの負荷が高いことが分かったのでさらに詳細にバッチジョブの分析を行う。

バッチジョブの処理傾向について分析する必要がある。図５を参照してみて夜間バッチジョブについてはバッチ・ウインドウが大きな要因となっていることが分かる。オンライン処理の開始までにバッチジョブを終了させる必要がある。CPU リソースを十分与えても設定された論理 CP をバッチジョブが効率よく使用できなければ処理は遅延してしまう。バッチジョブの形態にはいくつかの処理形態がある。バッチジョブは CPU 処理と IO 処理の混在であり、統合して区画で稼動させるバッチジョブ群がどのような処理形態であるかの調査を行う。CPU リソースを多く使用するバッチジョブが多くある場合は論理 CP の数を多く設定する必要がある。とくに個々のジョブの処理時間（Elapse Time）が長い（１０分以上）場合には注意を要する。

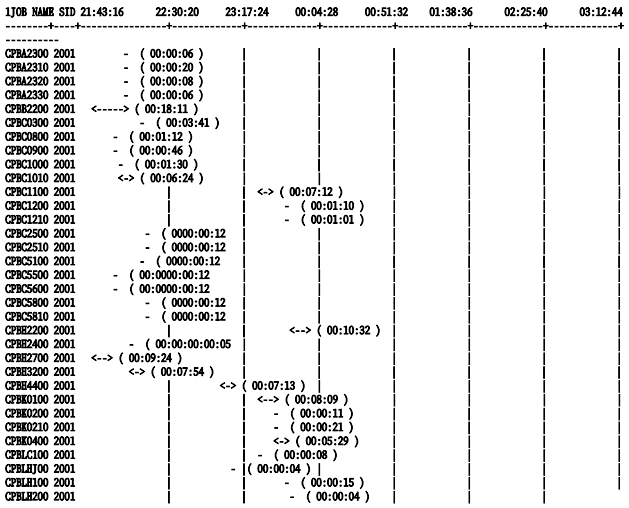
そのため、バッチジョブの分析を行いバッチジョブの特性の把握をおこなった。

表１は図５で示した夜間のバッチジョブの時間帯で稼動したジョブをジョブ群で分けて特性を表示したものである。この時間帯で約９０００のジョブが稼動している。それらのジョブの稼動状況をバーグラフで示したのが図７である。

表１．バッチジョブ群の稼動特性

JOB	NUM	ELAPSE	CPU	CPU/EPS	IO	IO/EPS	EXCP
CPB*	1061	150K	2806	1.9%	2371K	15.7%	11786K
CPE*	110	18978	669	3.5%	271K	14.3%	817K
KDD*	929	179K	9505	5.3%	2541K	14.2%	11397K
KGD*	2107	241K	21195	8.8%	1298K	5.4%	4976K
KJD*	3137	413K	23235	5.6%	6474K	15.7%	17363K
KKD*	584	98362	5870	6.0%	2507K	25.5%	12470K
SED*	221	26163	3094	11.8%	525K	20.1%	1549K
SPC*	882	103K	11857	11.5%	1395K	13.5%	3014K
TOTAL	9031	342:09:23	21:43:50.20	6.4%	48:17:39.12	14.1%	63377K

図７．バッチジョブ稼動状況のバーグラフ



この時間帯の８つのジョブ群でみると約９０００のジョブが稼動している。ジュブ群での処理時間に占めるCPU時間は１．９%から１１．８%であり、平均は６．４%である。バッチジョブの一般的 CPU 時間である。IO の処理時間にしめる割合も平均で１４%である。図７の稼動状況のバーグラフで確認してもほとんどのジョブが間時間（５分以内）で終了していることが分かる。

従って、夜間時間帯には大量のジョブが処理されるが、それぞれは CPU、IO とも比較的少なく処理時間も短いジョブであることがわかる。

論理区画に移行しても統合前と同じ論理 CP 数を設定すれば稼動状況に変化はないと予測できる。

４．システム統合時の評価

統合前の各システムの稼動状況をシステム診断で使用している手法、TOOL を使用して分析を行い、その結果を論理区画の設定の資料として反映した。実際には４筐体、１０システムを２つの筐体への統合をおこなった。事情があり詳しい評価を出来ないのが残念であるが、順調に稼動している。