Full GC発生を抑止するメモリ管理技術

CosminexusのFull GCレス機能

2010.1 January

Contents

＜本書での表記＞

Java VM: Java Virtual Machine

1. 「Stop the World」−ミッションクリティカルシステムを止めない

ここ数年、銀行・証券・為替といった金融分野などのミッションクリティカルシステム、すなわち止まることを許されないシステムでも、Java を基盤とした Web システムを適用することが多くなってきています。しかし、Java VM のメモリ管理を考慮したプログラム設計とパラメータチューニングを行っても、稼働後、予想以上の高負荷になると性能上のトラブルが発生するという事例が後を絶ちません。

順調に動いていた Web システムが、ある時突然応答を返さなくなり、しばらくするとまた動き出す。そのような事象に不安を感じてはいませんか？ Java を基盤とした Web システム環境では避けられないこの現象は、Java VM のガーベージコレクション(GC)、中でも Full GC と呼ばれるものが原因です。

Java VM には、Java を基盤としたシステムのメモリ管理者としての役割があります。使われなくなったメモリは Java VM がガーベージコレクションを行うことで回収します。ただし、このガーベージコレクションの間、実行中のすべてのアプリケーションプログラムは停止します。これが、「Stop the World」と呼ばれる現象です。「Stop the World」は、Web システムのレスポンスの遅延やスローダウン、スループットの低下の要因となります。

近年、Web システムは大規模化・複雑化し、業務で参照・更新するデータのサイズはさらに大きくなってきています。巨大なデータには、64bit OS を適用することで対応できます。しかし、Java VM が扱うメモリを増やすと、ガーベージコレクションはより長時間化するため、この問題はさらに深刻なものとなります。

本冊子では、まず、Java VM のメモリ管理とガーベージコレクションについて説明します。その後、日立の Cosminexus V8 が可能にした、日立独自のメモリ管理方式による「Full GC レス機能」をご紹介いたします。

2. Java VMのメモリ管理とガーベージコレクション

2.1 ガーベージコレクションとは

Java のプログラムでは、使用済みメモリを明示的に削除する必要がありません。不要になったメモリは、Java VM によって自動的に回収されるためです。

業務処理が進むと、図 2-1 の左図のように、使用済みの領域と使用中の領域が混在し、使用できるメモリ領域が減っていきます。Java VM は、使用できるメモリ領域が少ないことを検知すると、ガーベージコレクションを実行します。ガーベージコレクションでは、使われなくなった領域を回収して、空き領域を作る処理が行われます。これにより、大きな空き領域が作られます。

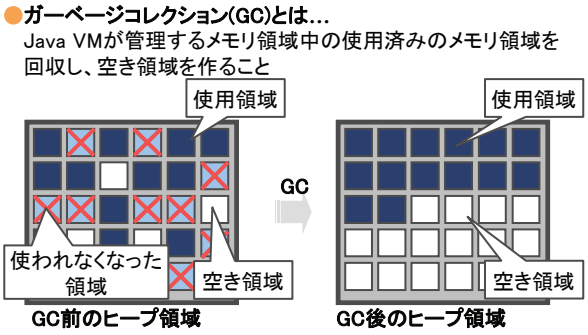


図 2-1 ガーベージコレクションによる使用済み領域の回収と空き領域の作成

2.2 Java VMのメモリ管理方式

ここで、Java VM のメモリ管理方式について説明します。Java VM のプロセスは、Java ヒープ、Perm ヒープ、C ヒープ、およびスレッドスタックという四つのメモリ領域を保持しています。メモリ領域の構成を図 2-2 に示します。ガーベージコレクションは、Java ヒープと Perm ヒープに対して行われます。

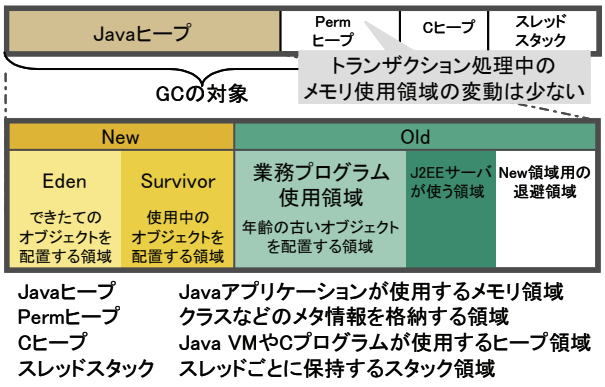


図 2-2 Java VM が管理するメモリ領域の構成

Java VM が管理するメモリ領域の中で、ガーベージコレクションの発生に大きく影響するのは、Java ヒープです。Java ヒープとは、業務処理で使用されるメモリ領域です。Java ヒープは、New 領域と Old 領域の二つに大別できます。New 領域はさらに Eden 領域と Survivor 領域に分けられます。

生成されたオブジェクトは、まず New 領域中の Eden 領域に配置されます。ガーベージコレクションが実行されると、使用中のオブジェクトが Survivor 領域に移動します。使用期間の長いオブジェクトは、何度かのガーベージコレクションを経て Old 領域に移動します。このため、New 領域には生成されてから間もないオブジェクトが、Old 領域には使用期間の長いオブジェクトが存在することになります。

2.3 Copy GCとFull GC

使用できるメモリ領域が少なくなると、Copy GC または Full GC のどちらかのガーベージコレクションが発生します。

Copy GC は New 領域の Eden 領域がいっぱいになると発生します。Copy GC の対象は、New 領域です。New 領域にある使用中のオブジェクトを他の領域に移動し、使用済みのオブジェクトをすべて削除します。そして、一定回数以上の Copy GC を超えて使用され続けている New 領域のオブジェクトを Old 領域に移動します。

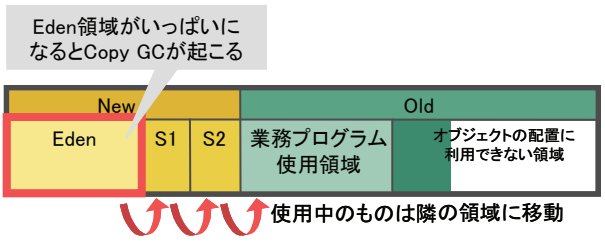


図 2-3 Copy GC

Full GC は Old 領域の業務プログラム使用領域がいっぱいになると発生します。Java ヒープ全体を対象として、使用済みのオブジェクトを削除します。Old 領域にある使用済みのオブジェクトが削除されるのは、Full GC が発生したときだけです。

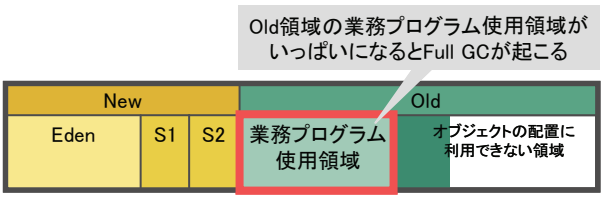


図 2-4 Full GC

Copy GC、Full GC 双方を比較してみると、表 2-1のようになります。

表 2-1　Copy GC と Full GC の比較

	範囲	発生タイミング	実行にかかる時間
Copy GC	New 領域	Eden 領域がいっぱいになった時	0.01～0.7 秒
Full GC	全領域	Old 領域の業務プログラム使用領域がいっぱいになった時	1 秒～数十秒

ここで問題となるのは、GC の実行にかかる時間です。Copy GC の実行にかかる時間は、使用中のオブジェクトの数に依存します。一方、Full GC の実行にかかる時間は、GC の対象となるメモリ領域全体のサイズに依存します。アプリケーションサーバのように、500M バイトから 1G バイト近くのメモリサイズを使用する場合、Copy GC は 0.01 秒から 0.7 秒程度と1秒に満たない間に終了します。しかし、Full GC は、Java ヒープ全体を対象とするため、Copy GC に比べて時間がかかります。その値は1秒から数十秒とされ、Java ヒープが 10G バイトある場合などには、30 秒以上かかることもあります。通常は、だいたい数秒ほどの実行時間がかかります。

2.4 「Stop the World」への従来のアプローチ

従来は、Full GC による数十秒にも及ぶ業務停止を避けるために GC のアルゴリズムを工夫する方法がとられてきました。GC のアルゴリズムには、一般に以下の方法が存在します。

(1) ノーマル GC

デフォルト GC ともいいます。前述した仕組みで Copy GC と Full GC の二つの GC を実施します。GC 発生時点で全実行スレッドを停止して、GC スレッドという一つのスレッドのみで GC を実行するというのが特徴です。

GCの種類	特徴
ノーマル GC	長所 ・スループットが高い ・GC発生頻度の見積もりが容易 短所 ・Copy GC、Full GC時に全スレッドが停止する ・Copy GC、Full GCを一つのスレッドで実行するため、業務処理停止時間が長い

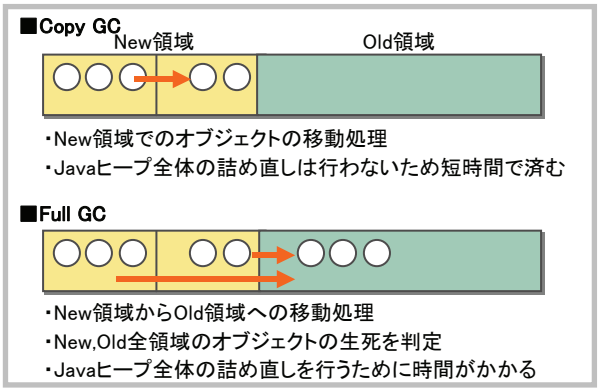


図 2-5　ノーマル GC の特徴

(2) パラレル GC

ノーマル GC とは異なり、GC を複数のスレッドで並列に実行することで GC による処理停止時間を短くしよう、というものです。

ただし、GC を並列化して実行することの副作用が伴います。例えば、スレッドローカルに GC を行うためにヒープ領域内を細分化する必要があります。これが要因で、GC の発生間隔が短くなってしまう場合もあります。

GCの種類	特徴
パラレル GC	長所 ・並列処理するCopy GCが速い 短所 ・Full GCはノーマルGCと同じ ・並列処理用に領域を細分化するためCopy GCの発生頻度は高い

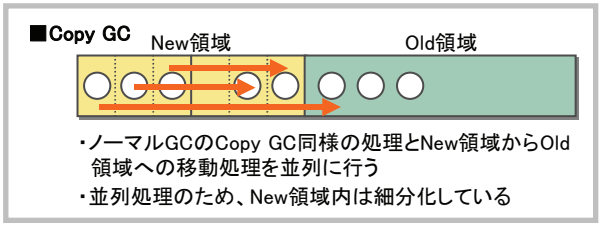


図 2-6　パラレル GC の特徴

(3) コンカレント GC

コンカレント GC では、業務スレッドと GC スレッドの実行を並行して行います。GC が動作している間は業務処理のスループットが低下するというデメリットがありますが、CPU のマルチコア化に伴い、その問題は克服されつつあります。

ただし、業務処理と GC を同時に動作させることによる副作用があります。そのひとつが、フラグメンテーションの問題です。図 2-8 に示すように、GC 処理の間も業務スレッドが並行して動作しているため、使用中の領域の詰め直しができません。そのため、ヒープ領域が断片化してしまい、空きメモリサイズはあるのに連続領域が取れなくなり、結果的に Full GC が発生する場合があります。

GCの種類	特徴
コンカレント GC	長所 ・Full GCの発生頻度が低い 短所 ・GC専用のスレッドが常時処理を行うため スループットが低い ・Copy GCはノーマルGCと同じ ・GC発生頻度の見積もりが困難 ・フラグメンテーションの発生によりメモリの 有効活用が困難

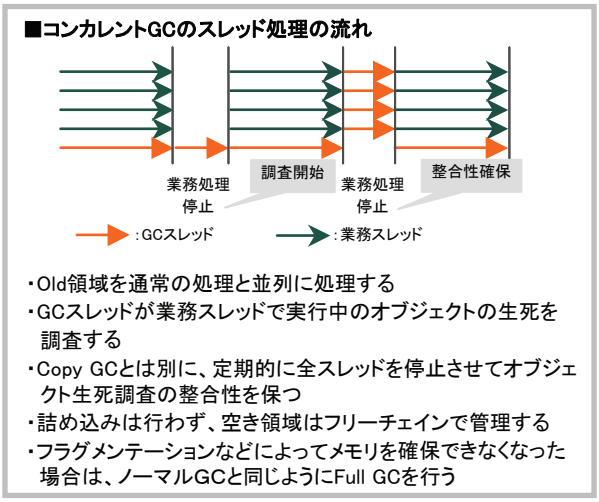
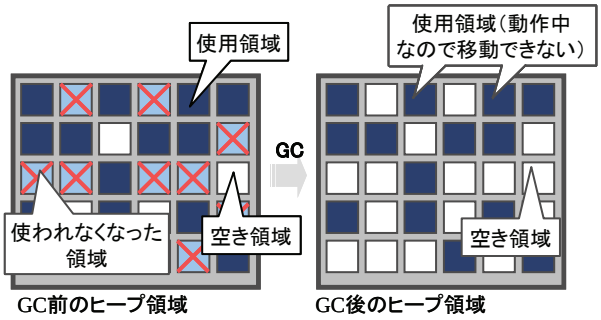


図 2-7 コンカレント GC の特徴



GC処理間も業務スレッドが動作しているため、使用中の領域の詰め直しができない。

図 2-8 コンカレント GC でのフラグメンテーションの問題

以上のように、GC アルゴリズムを使用することで、GC の発生頻度や処理時間を低減できる場合もあります。しかし、Full GC の発生自体は避けられません。これが、「Stop the World」と長年、技術者を悩ませてきた問題の根本原因です。

3. 「Full GCレス」を実現する最新技術

3.1 Full GC発生の要因

ここで、Cosminexus アプリケーションサーバが搭載している、Full GC の発生を抑止する機能（Full GC レス機能）についてご紹介します。Cosminexus アプリケーションサーバでは、従来のアプローチであるガーベージコレクションのアルゴリズムの変更ではなく、Java ヒープの使い方そのものを工夫することで Full GC を抑止します。これによって、「Stop the World」を解消します。

Java ヒープの New 領域に配置されたオブジェクトのうち、使用期間の短いもの（短命なオブジェクト）は New 領域で削除され、使用期間の長いもの（長寿命なオブジェクト）は Old 領域に移動します。これを Web アプリケーションサーバに当てはめて考えると、トランザクションのような短時間で完了する処理で使われるオブジェクトは、ほかの領域に移動する前に使用済みになるため、New 領域で削除されます。一方、セッション情報のように、クライアントユーザーのログインからログアウトまでの長時間にわたって使用されるオブジェクトは、New 領域から Old 領域に移動していきます。

図 3-1 で見ると、クライアントユーザーがログインして、商品表示、商品購入、というトランザクションがそれぞれ実行されていきます。これらのトランザクションは短時間で終了するため、作成したオブジェクトは、New 領域で Copy GC によって削除されます。

一方、ログインからログアウトまでの情報を保持するセッション情報は、Old 領域に移動します。セッション情報は、クライアントユーザーがログインするたびに作成され、Old 領域に移動します。Old 領域に移動したセッション情報は、Copy GC では削除されません。これを繰り返していくと、Old 領域がいっぱいになって Full GC が発生することになります。

つまり、Web アプリケーションサーバの場合、セッション情報のような使用期間が長い特定のオブジェクトの蓄積が、Full GC 発生の要因となっているのです。

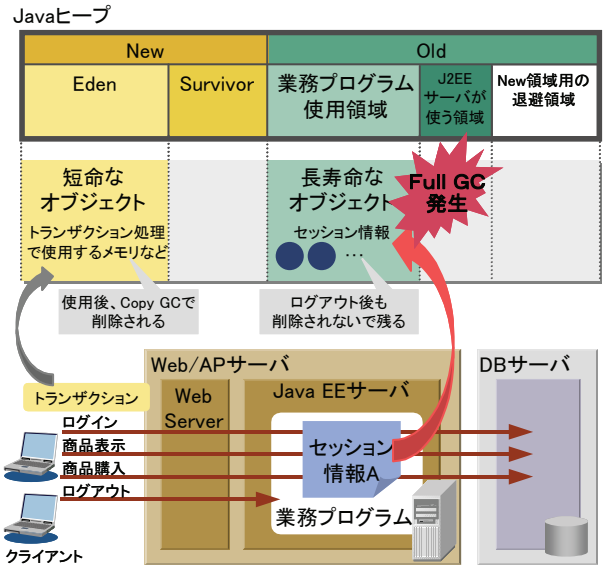


図 3-1 Web アプリケーションサーバでの Full GC 発生の要因

3.2 明示管理ヒープ領域によるFull GCの抑止

Cosminexus アプリケーションサーバではメモリ管理の仕組みそのものを変更することで、Full GC による「Stop the World」を解消しました。

これは、使用期間の長いオブジェクトをガーベージコレクションの対象外の領域で管理するというものです。ガーベージコレクションの対象外にする領域を明示管理ヒープ領域といいます。明示管理ヒープ領域とは、長寿命なオブジェクトを配置する領域です。オブジェクトの使用期間が過ぎると、オブジェクトが配置されていた領域は明示的に削除されます。

オブジェクトの明示管理ヒープ領域への配置は、Cosminexus アプリケーションサーバが自動で行う場合と、ユーザが指定する場合があります。

① Cosminexus アプリケーションサーバが自動で行う場合

Cosminexus アプリケーションサーバは、セッションに関連するオブジェクトを自動的に、明示管理ヒープ領域に配置します。

② ユーザが指定する場合

①以外に、明示管理ヒープ領域で管理したいオブジェクトが有る場合は、ユーザがそのオブジェクトのクラス名を自動配置設定ファイルに記述することで、①と同様に、明示管理ヒープ領域で管理するようになります。

まず、①について、Cosminexus アプリケーションサーバがセッションに関連するオブジェクトを自動的に配置する明示管理ヒープ領域上のオブジェクトの配置先を図 3-2 に示します。セッション情報は自動的に明示管理ヒープ領域に配置されるため、Old 領域にセッション情報が蓄積されることはありません。明示管理ヒープ領域に配置されたセッション情報は、不要になるとCosminexus アプリケーションサーバによって削除されます。

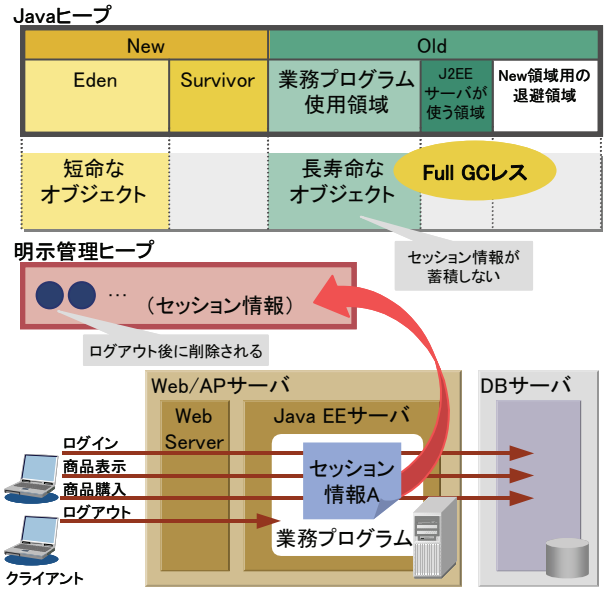


図 3-2　明示管理ヒープ領域を使用した場合のオブジェクトの配置先

明示管理ヒープ領域を適用する前後の Old 領域のイメージを図 3-3 に示します。適用前は、業務定義情報、コネクションプール、スレッドプール、セッション情報など、すべての長寿命なオブジェクトが Old 領域に移動します。一方、適用後は、セッション情報が自動的に明示管理ヒープ領域に移動されるため、Old 領域に移動しません。これによって、不要になったオブジェクトで Old 領域がいっぱいになることを防ぎ、Full GC によるオンライン業務の長時間停止を発生させません。

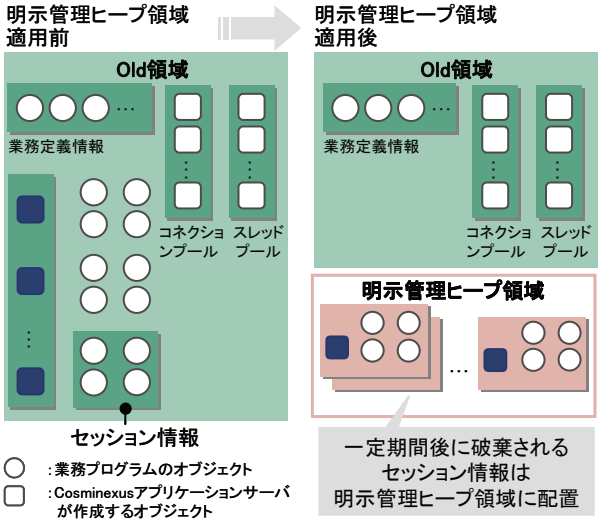


図 3-3　明示管理ヒープ領域適用前後の Old 領域の違い

利用者から見た明示管理ヒープ領域適用の効果を図 3-4 に示します。

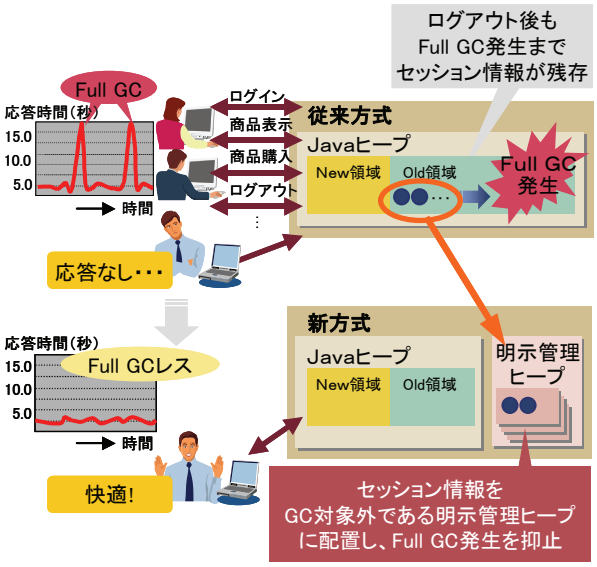


図 3-4　明示管理ヒープ領域適用の効果

次に、②について、ユーザ指定により、Cosminexus アプリケーションサーバが特定のオブジェクトを明示管理ヒープ領域に配置するイメージを図 3-5 に示します。まず、Old 領域増加要因調査機能を使用して Old 領域増加原因クラス名をログファイルに出力します。次に、ユーザがログファイルを参照し、明示管理ヒープ管理領域へ移動するオブジェクトのクラス名を記述する自動配置設定ファイルを編集します。Cosminexus アプリケーションサーバは、自動配置設定ファイル中のクラス名を参照し、Old 領域の外にある明示管理ヒープ領域に配置します。また、明示管理ヒープ領域へ移動したオブジェクトは、使用されなくなった時点で、Cosminexus アプリケーションサーバによって、削除されます。

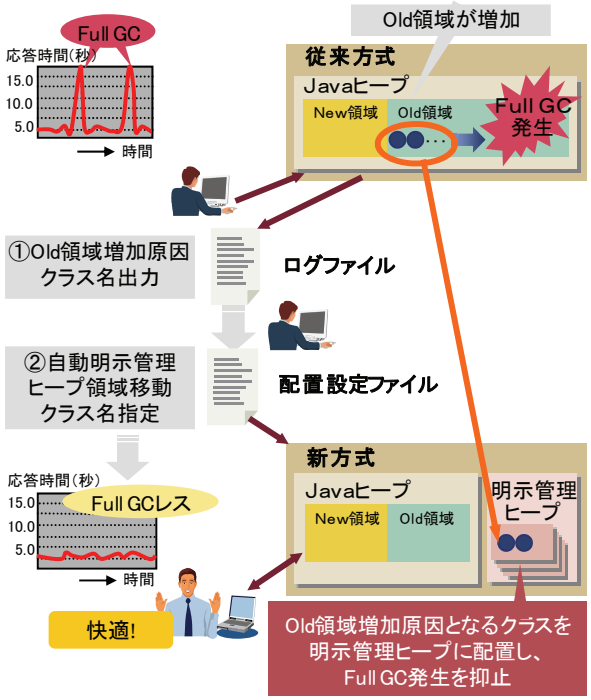


図 3-5 ユーザ指定による明示管理ヒープ領域への配置手順

従来の Java ヒープだけ使用する方式では、セッション情報などの長寿命なオブジェクトの蓄積に伴って定期的に Full GC が発生したため、利用者が「応答なし」と感じる時間がありました。新方式である明示管理ヒープを適用することで、Old 領域へのセッション情報の蓄積がなくなり、Full GC の発生が抑止されます。このため、利用者は常に快適に操作を続けることができます。

このように、Full GC 発生の要因となる長寿命のオブジェクトを GC 対象外の領域に配置し、Full GC の発生を抑止するのが、Cosminexus アプリケーションサーバの Full GC レス機能です。

3.3 Full GCレス機能の仕組み

3.3.1 Cosminexusアプリケーションサーバがセッションに関するオブジェクトを自動的に明示管理ヒープ領域に配置する場合

Cosminexus アプリケーションサーバでは、長寿命なオブジェクトのうち、セッション情報などを自動的に明示管理ヒープ領域に配置します。業務プログラムに対応して、Cosminexus アプリケーションサーバが明示管理ヒープ領域とどのように連携するかの例を図 3-6に示します。

(1) セッションの生成

業務プログラムがセッションを生成すると、Cosminexus アプリケーションサーバでセッションの生成が行われます。この時に明示管理ヒープ領域が確保されます。

(2) オブジェクトの生成およびセッションへのオブジェクトの格納

ユーザがオブジェクトを生成し、セッションにオブジェクトを格納すると、明示管理ヒープ領域上のセッション情報格納用の領域と、オブジェクトが関連付けられます。セッションと関連付いたオブジェクトは、Copy GC が発生したタイミングで明示管理ヒープ領域に移動します。

(3) オブジェクトの生成およびセッションへのオブジェクトの格納

セッションにほかのオブジェクトを格納した場合も(2)と同様に処理されます。

(4) セッションの終了

セッションの利用が終了すると明示管理ヒープ領域は削除されます。

このように、明示管理ヒープ領域の取得、関連付け、削除は、Cosminexus アプリケーションサーバが行います。業務プログラムの変更は不要です。

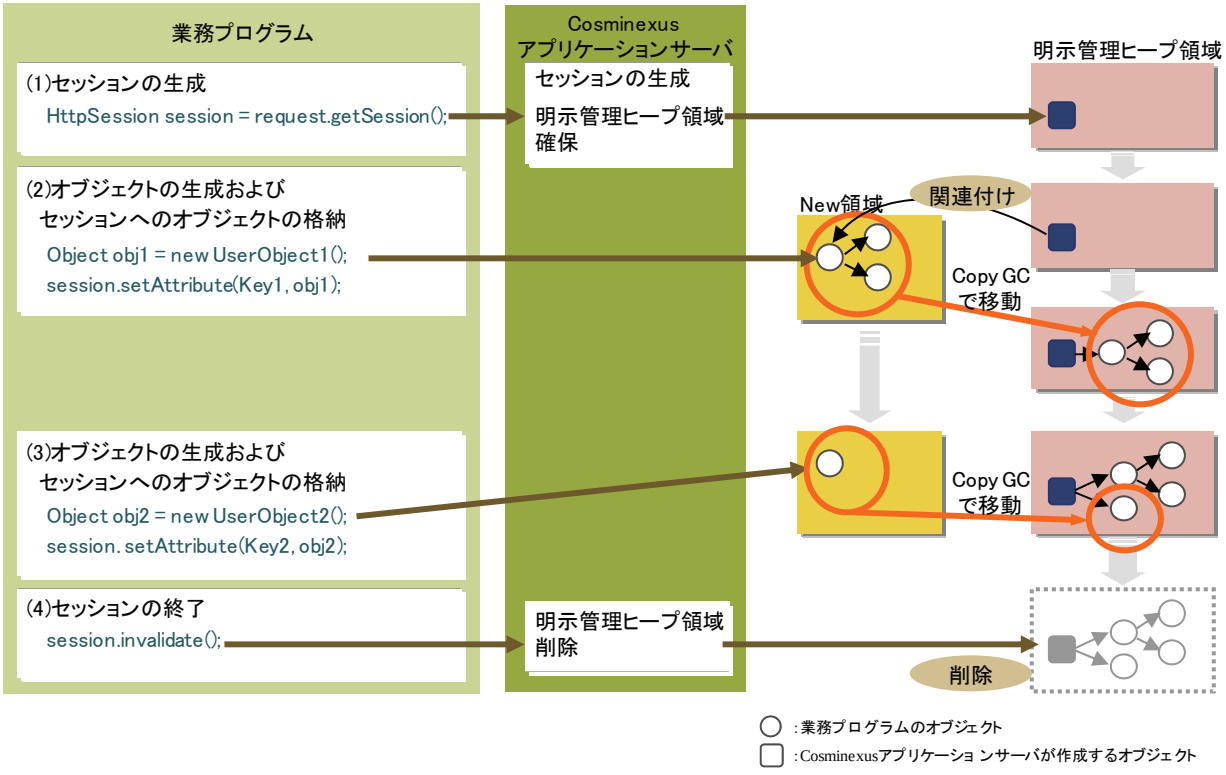


図 3-6 業務プログラムに対応した Cosminexus アプリケーションサーバと明示管理ヒープ領域との連携(その1)

なお、セッションの終了後に削除される明示管理ヒープ領域には、複数のセッションから参照されているオブジェクトなど、まだ使用中のオブジェクトが含まれている可能性があります。明示管理ヒープ領域のオブジェクトが複数のセッションから参照されている例を図 3-7に示します。この場合、単純に明示管理ヒープ領域を削除することができません。

これを防ぐため、明示管理ヒープ領域を削除する際には、使用中のオブジェクトを検出する処理が実行されます。使用中のオブジェクトが検出された場合は、Java ヒープ領域に移動します。この処理によって、明示管理ヒープ領域を削除したあとのシステムが問題なく動作できるようにして、システムの堅牢性を確保しています。

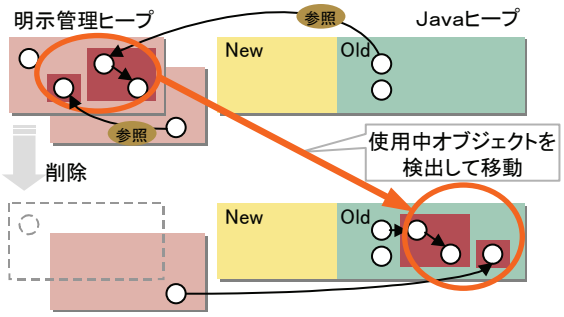


図 3-7　使用中オブジェクトの検出と移動

3.3.2 ユーザが明示管理ヒープ領域に配置するオブジェクトを指定する場合

Cosminexus アプリケーションサーバでは、長寿命なオブジェクトのうち、ユーザが指定したオブジェクトを自動的に明示管理ヒープ領域に配置します。業務プログラムに対応して、Cosminexus アプリケーションサーバが明示管理ヒープ領域とどのように連携するかの例を図 3-8に示します。

(1) 自動配置設定ファイルの入力

Java VM が起動されると、自動配置設定ファイルから、明示管理ヒープ領域へ配置するクラス名の情報を読み込みます。

(2) オブジェクトの生成

ユーザが(1)で読み込んだクラスのオブジェクトを生成した時点で、明示管理ヒープ領域を確保し、オブジェクトを、明示管理ヒープ領域上に生成します。

(3) 関連するオブジェクトの生成および関連付け

(2)で生成したオブジェクトと今回生成したオブジェクトが関連付けられます。このオブジェクトは、Copy GC が発生したタイミングで明示管理ヒープ領域に移動します。

(4) 明示管理ヒープ領域の削除

オブジェクトおよび関連付いたオブジェクトが不要となった時点で、明示管理ヒープ領域は削除されます。明示管理ヒープ領域の削除は、3.3.1 と同様に、使用中のオブジェクトを検出し、使用中のオブジェクトに対しては、Java ヒープ領域に移動することで、明示管理ヒープ領域を削除したあとのシステムが問題なく動作できるようにして、システムの堅牢性を確保しています。

このように、明示管理ヒープ領域の取得、関連付け、削除は、Cosminexus アプリケーションサーバが行います。業務プログラムの変更は不要です。

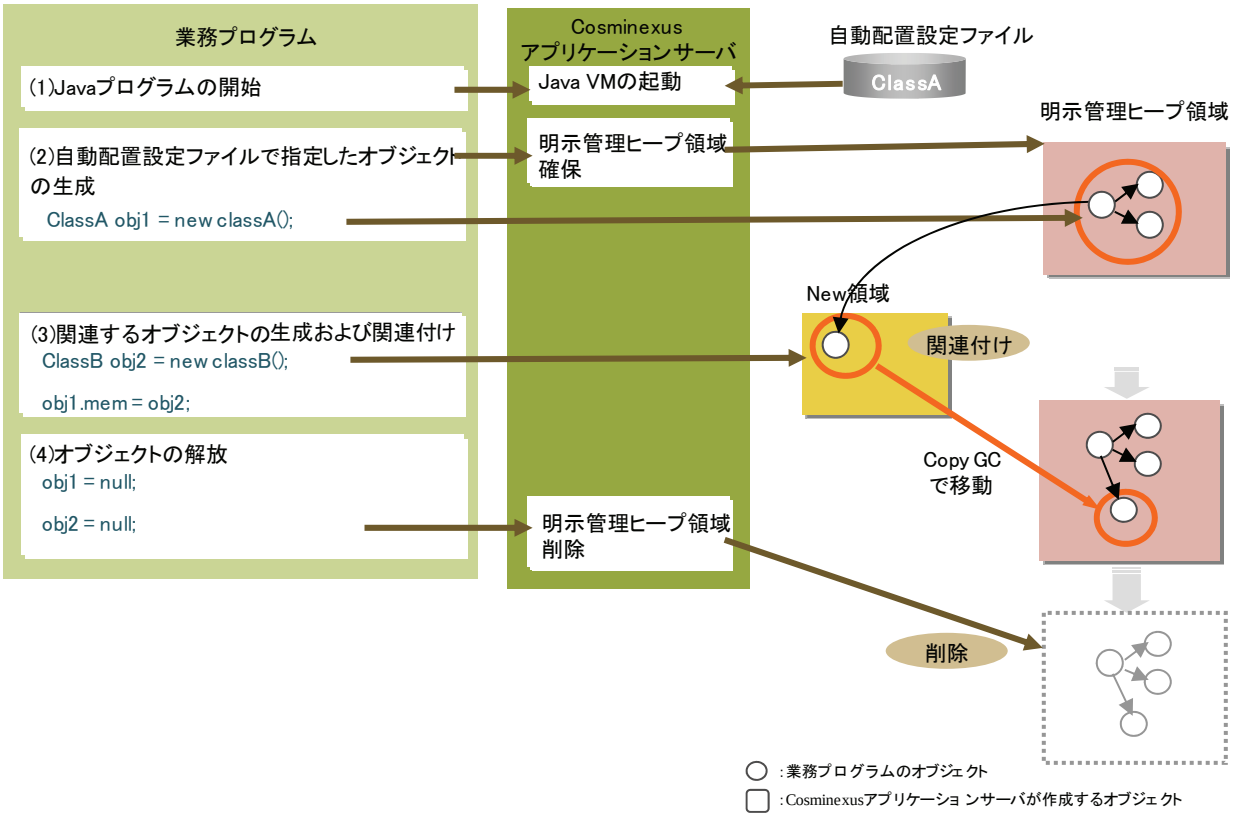


図 3-8 業務プログラムに対応した Cosminexus アプリケーションサーバと明示管理ヒープ領域との連携(その2)

3.4　Javaヒープ使用量とレスポンスタイムの変化

実際に Full GC レス機能を利用した場合の Java ヒープ使用量の変化を図 3-9に示します。これはあるシステムの規模を縮小し、再現した環境での測定結果です。従来はオンライン業務を繰り返すと Full GC が定期的に発生していたのが、Cosminexus アプリケーションサーバの Full GC レス機能により、Full GC 発生が抑止され、Java ヒープ使用量が安定動作していることがわかります。

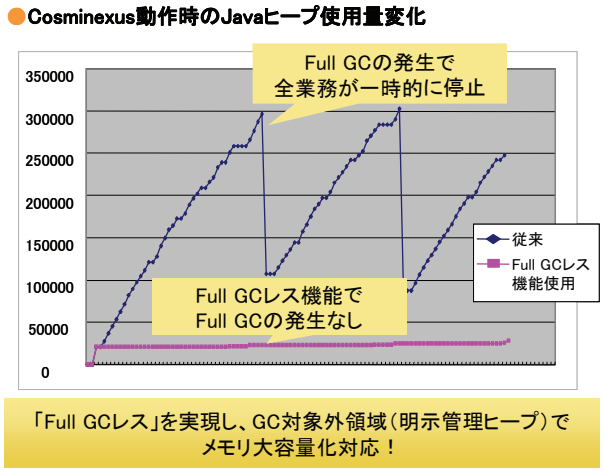


図 3-9　Java ヒープ使用量の変化

Full GC が抑止されたことに伴うレスポンスタイムの変化の例を表 3-1 に示します。この例では、レスポンスタイムとしてログインにかかる時間を比較しています。なお、適用前の環境の Java ヒープサイズと、適用後の環境の Java ヒープと明示管理ヒープの合計サイズは同じです。

表 3-1　レスポンスタイムの変化

	Full GC レス機能適用前	Full GC レス機能適用後
Java ヒープサイズ	1024m	512m
明示管理ヒープサイズ		512m
Full GC 発生回数	17 回	0 回
最大ログイン時間(Sec)	2.343	0.204
平均ログイン時間(Sec)	0.063	0.057

表 3-1 で示した結果から、最大ログイン時間が大幅に改善されていることがわかります。Full GC 発生回数が 0 になったことで、Full GC の影響を受けるリクエストがなくなったためです。これに伴い、平均ログイン時間も短縮されています。

3.5　Copy GC発生時の処理コストの変化

これまでに説明したように、Full GC レス機能を使用することで、オンライン業務の長時間停止を防ぎ、システムの安定稼働を実現できます。ただし、Copy GC 発生時の処理時間という点に絞って見た場合、処理時間は増加します。Copy GC 発生時に、明示管理ヒープ領域に移動するオブジェクトの候補の解析処理と、New 領域中のオブジェクトを Old 領域と明示管理ヒープ領域のどちらに移動するかを判定する処理が発生するためです。

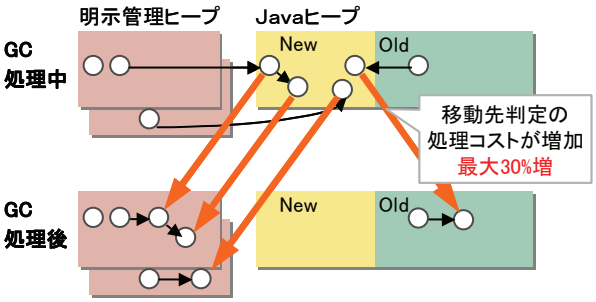


図 3-10　移動先の判定処理による処理コストの増加

この処理によって、Full GC レス機能を適用しない場合に比べて、Copy GC にかかる処理時間が最大で 30%増加することがあります。

ただし、一般的に、1回の Copy GC にかかる時間は、数十 ms～数百 ms 程度と短いものです。Copy GC の処理コスト増加は、処理性能全体に大きく影響はしません。なお、Full GC レス機能の適用によってオーバーヘッドが増加するのは、処理全体を通して Copy GC 時の処理コストだけです。

Copy GC 時間の増加によるスループットの変化を試算した例を表 3-2 に示します。

表 3-2　Copy GC 時間増加によるスループットの変化

	Full GC レス機能適用前	Full GC レス機能適用後
測定単位時間	10:00～15:45(20,700 秒)	
Copy GC 回数	1,658 回	
平均 Copy GC 時間	0.253 秒/回	0.328 秒/回
処理件数	548,819 件	545,486 件

平均 Copy GC 時間が 30%増加したことに伴って減少する処理件数の割合は、0.61%です。つまり、この例の場合、Copy GC の処理コスト増加による処理性能への影響は、1%未満であることがわかります。

このように、Full GC レスという大きな効果に対して、副作用と考えられる影響が非常に小さいことも、Full GC レス機能の特徴です。

4. まとめ

Web システムの課題である「Stop the World」。Cosminexus アプリケーションサーバでは、これを GC アルゴリズムの変更によって短縮するのではなく、Full GC そのものを発生させない「Full GC レス機能」によって解消します。Full GC レス機能は、Cosminexus アプリケーションサーバの高度な独自技術によって実現されています。

Full GC レス機能を使用する場合、Web アプリケーションの変更は不要です。稼働実績のあるアプリケーションを使用しながら、Full GC レスの恩恵を享受し、Web システムの安定稼働を実現できます。

Cosminexus

2010年1月第2版発行
お問い合わせ先：株式会社 日立製作所 ソフトウェア事業部　販売推進部
cosminexus-s@itg.hitachi.co.jp

インターネットで製品情報をご覧いただけます。
http://www.hitachi.co.jp/apserver/
http://www.cosminexus.com/