HITACHI
Inspire the Next

Inside JavaVM

～先進のメモリ管理技術とは～

2008年11月18日
株式会社日立製作所　ソフトウェア事業部
第2AP基盤ソフト設計部　担当部長
中島 恵

uVALUE

HITACHI
Inspire the Next

Contents

1. Javaのメモリ管理の課題とは
2. GC(ガベージコレクション)の仕組み
3. 「FullGCレス」を実現する最新技術
4. 「FullGCレス」の効果(ﾃﾞﾓﾝｽﾄﾚｰｼｮﾝ)
5. メモリトラブルを起こさないためには

uVALUE

HITACHI
Inspire the Next

1

Javaのメモリ管理の課題とは

uVALUE

1. Javaのメモリ管理の課題とは

HITACHI
Inspire the Next

Webシステムの安定性・堅牢性を高めるためには

Webシステムは、アプリケーションサーバおよびJavaによる
開発生産性の向上によって、容易に構築が可能となった

➔業務量ピーク時など、性能上のトラブルが後を絶たない

●システムの性能劣化につながる要因

1. CPUネック
2. メモリネック ← JavaVMのメモリ管理機能（GC）に起因
3. バックエンドシステムによるネック

1. Javaのメモリ管理の課題とは

HITACHI
Inspire the Next

Webシステムの扱うデータサイズの肥大化

Webシステムが参照・更新するデータのサイズが
どんどん大きくなってきている

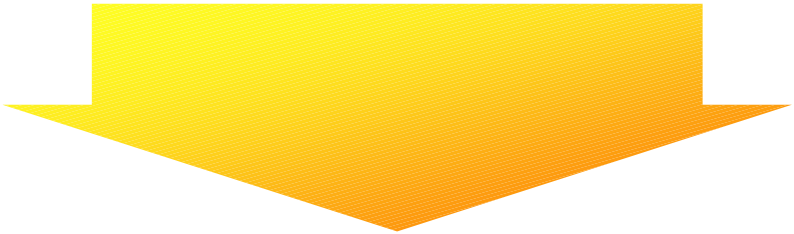
➔64ビット化で対処はできるが、メモリ領域サイズの
増大に伴ってGCでの停止時間が長時間し、
システム性能劣化に陥る

JavaVMのメモリ管理機能（GC）に起因

1. Javaのメモリ管理の課題とは

HITACHI
Inspire the Next

JavaVMのメモリ管理機能（GC）に起因した問題が山積



JavaVMのメモリ管理機能(GC)として、これらの問題を解消する「次の一手」が、課題解決のためには必要

➔ JavaVMを大きく改良することによって、課題を解決するアプローチが見出せるのでは？

HITACHI
Inspire the Next

2

GC(ガベージコレクション)の仕組み

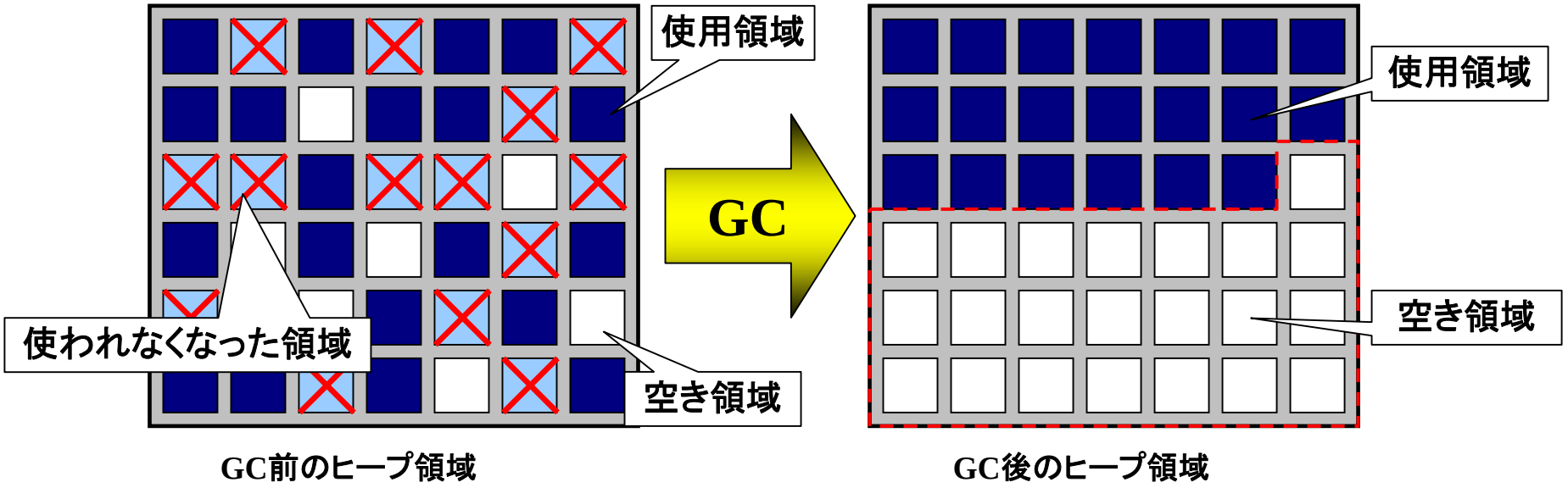
uVALUE

2. ガベージコレクションの仕組み

HITACHI
Inspire the Next

- **ガベージコレクション（GC）とは…**

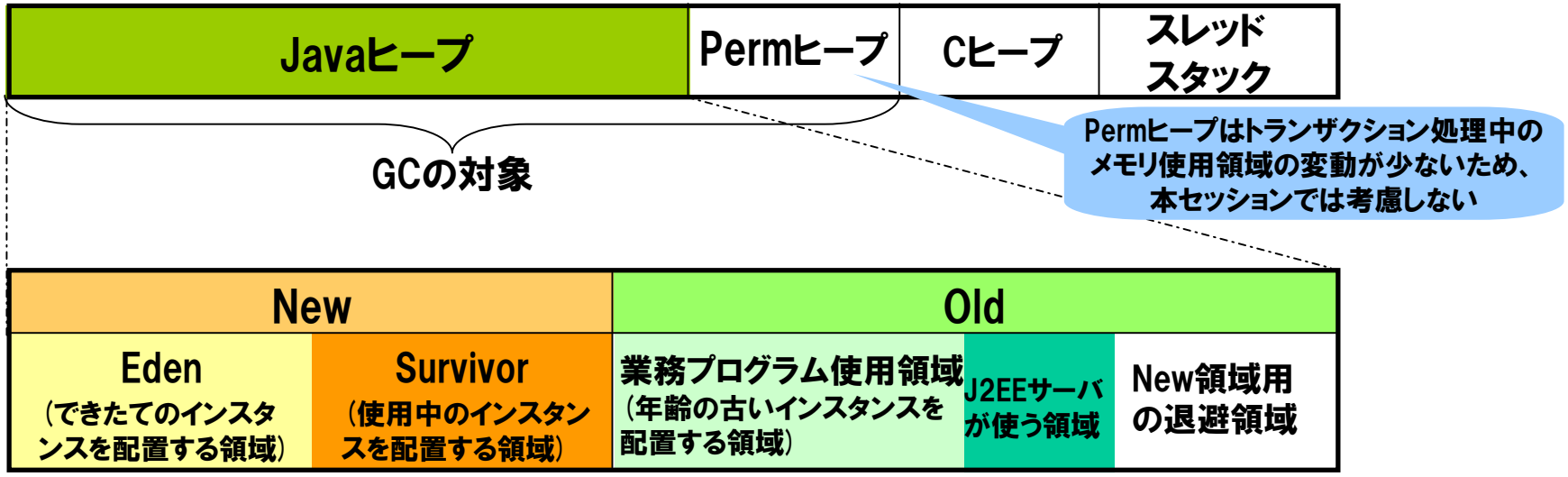
  JavaVMが管理するメモリ領域中の使用済みのメモリ領域を破棄し、
  空き領域を作ること



2.ガベージコレクションの仕組み
-Javaヒープとは-

HITACHI
Inspire the Next

- Javaプロセスのメモリの内訳



- Javaヒープ　Javaアプリケーションが使用するメモリ領域
- Permヒープ　クラスなどのメタ情報を格納する領域
- Cヒープ　JavaVMやCプログラムが使用するヒープ領域
- スレッドスタック　スレッドごとに保持するスタック領域

2.ガベージコレクションの仕組み
-CopyGCとFullGC-

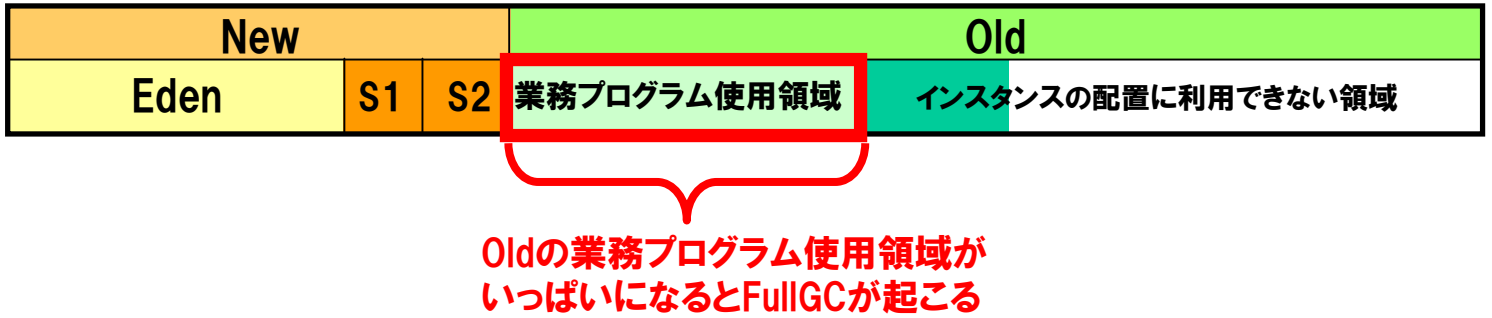
HITACHI
Inspire the Next

●GCには2種類ある

CopyGC…New領域を対象に、使用済みのインスタンスを全て削除する

使用中のインスタンスは隣の領域へ移動する

FullGC…全ての領域を対象に、使用済みのインスタンスを全て削除する



2.ガベージコレクションの仕組み
-GCの問題点-

HITACHI
Inspire the Next

- 2つのGCの比較

	範囲	発生タイミング	実行にかかる時間
CopyGC	New領域	Eden領域がいっぱいになった時	0.01～0.7秒
FullGC	全領域	Old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

・FullGCは特に時間がかかる

・Javaヒープのサイズにより実行にかかる時間は増加する

GCはメモリの空き容量を確保するために必要な処理

しかし、GCが発生すると、GC実行中は業務処理が停止する

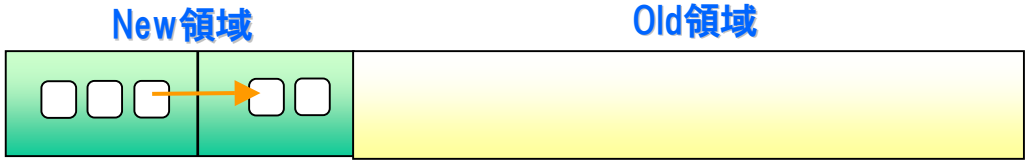
GCのアルゴリズムを変えて
対処してみる？

2.ガベージコレクションの仕組み
世代別GCのアルゴリズム　-ノーマルGC-

HITACHI
Inspire the Next

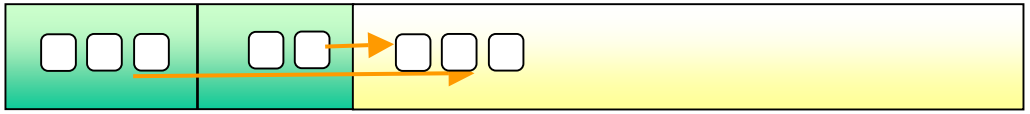
GCの種類	特徴
ノーマルGC	長所 ・スループットが高い ・GC発生頻度の見積もりが容易 短所 ・Copy GC、Full GC時に全スレッド停止する ・Copy GC、Full GCを1つのスレッドで実行する

Copy GC



・New領域におけるオブジェクトの移動処理
・詰め直しは行わないため短時間

Full GC

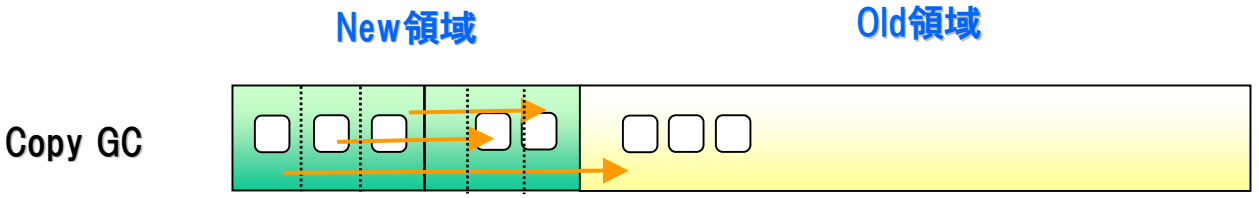


・New領域からOld領域への移動処理
・New,Old全領域のオブジェクト生死判定
・詰め直しを行うために時間がかかる

2.ガベージコレクションの仕組み
世代別GCのアルゴリズム -パラレルGC-

HITACHI
Inspire the Next

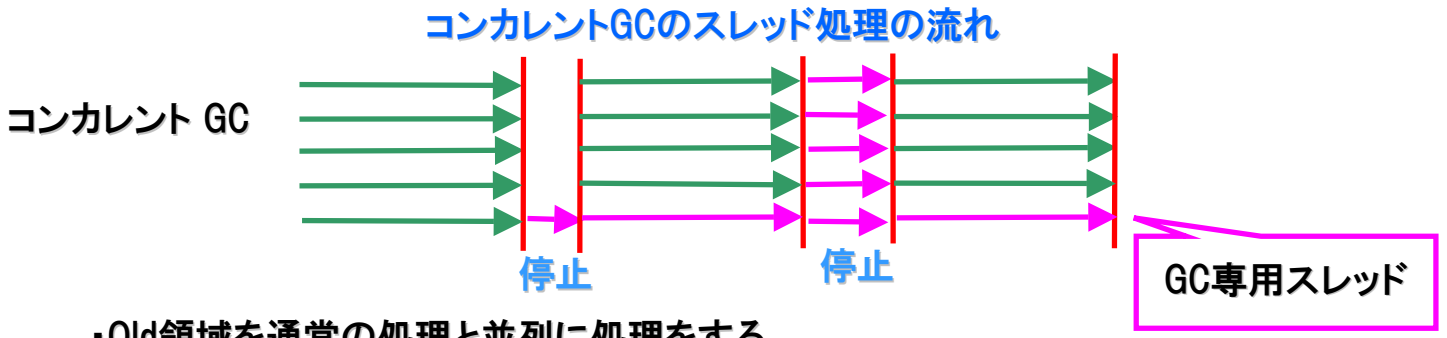
GCの種類	特徴
パラレルGC	長所 ・並列処理するCopy GCが速い 短所 ・Full GCはノーマルGCと同じ ・並列処理用に領域を細分化するためCopy GCの発生頻度は高い



・ノーマルGCのCopy GC同様の処理をNew領域からOld領域への移動処理を並列に行う
・並列処理のため、New領域内は細分化している

2.ガベージコレクションの仕組み
世代別GCのアルゴリズム　-コンカレントGC-

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



・Old領域を通常の処理と並列に処理をする
・GC専用スレッドが一般スレッドで実行中のオブジェクト生死を調査
・Copy GCとは別に定期的に全スレッドを停止させてオブジェクト生死調査の整合性を保つ
・詰め込みは行わず、空き領域はフリーチェインで管理する
・フラグメンテーションなどによりメモリ確保が出来なくなった場合は、ノーマルGCと同じFullGCを行う

2.ガベージコレクションの仕組み
-GCの問題点-

HITACHI
Inspire the Next

- 2つのGCの比較

	範囲	発生タイミング	実行にかかる時間
CopyGC	New領域	Eden領域がいっぱいになった時	0.01～0.7秒
FullGC	全領域	Old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

・FullGCは特に時間がかかる

・Javaヒープのサイズにより実行にかかる時間は増加する

GCはメモリの空き容量を確保するために必要な処理

しかし、GCが発生すると、GC実行中は業務処理が停止する

GCアルゴリズムは
<u>複雑になるほど
コントロール不可</u>と
なる

2.ガベージコレクションの仕組み
-GCの問題点-

HITACHI
Inspire the Next

- 2つのGCの比較

	範囲	発生タイミング	実行にかかる時間
CopyGC	New領域	Eden領域がいっぱいになった時	0.01～0.7秒
FullGC	全領域	Old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

・FullGCは特に時間がかかる

・Javaヒープのサイズにより実行にかかる時間は増加する

GCはメモリの空き容量を確保するために必要な処理

しかし、GCが発生すると、GC実行中は業務処理が停止する

FullGCの発生を抑止することを目的に
発想の転換を行った
➔メモリ領域を1つの枠組みで管理することに
そもそも無理があるのでは？

2.ガベージコレクションの仕組み
-明示管理ヒープ方式-

HITACHI
Inspire the Next

■ヒープ管理方式

世代別ヒープ管理方式
（世代別GC方式）

＋

NEW
明示管理ヒープ方式
特許出願済

■GC方式

ノーマルGC

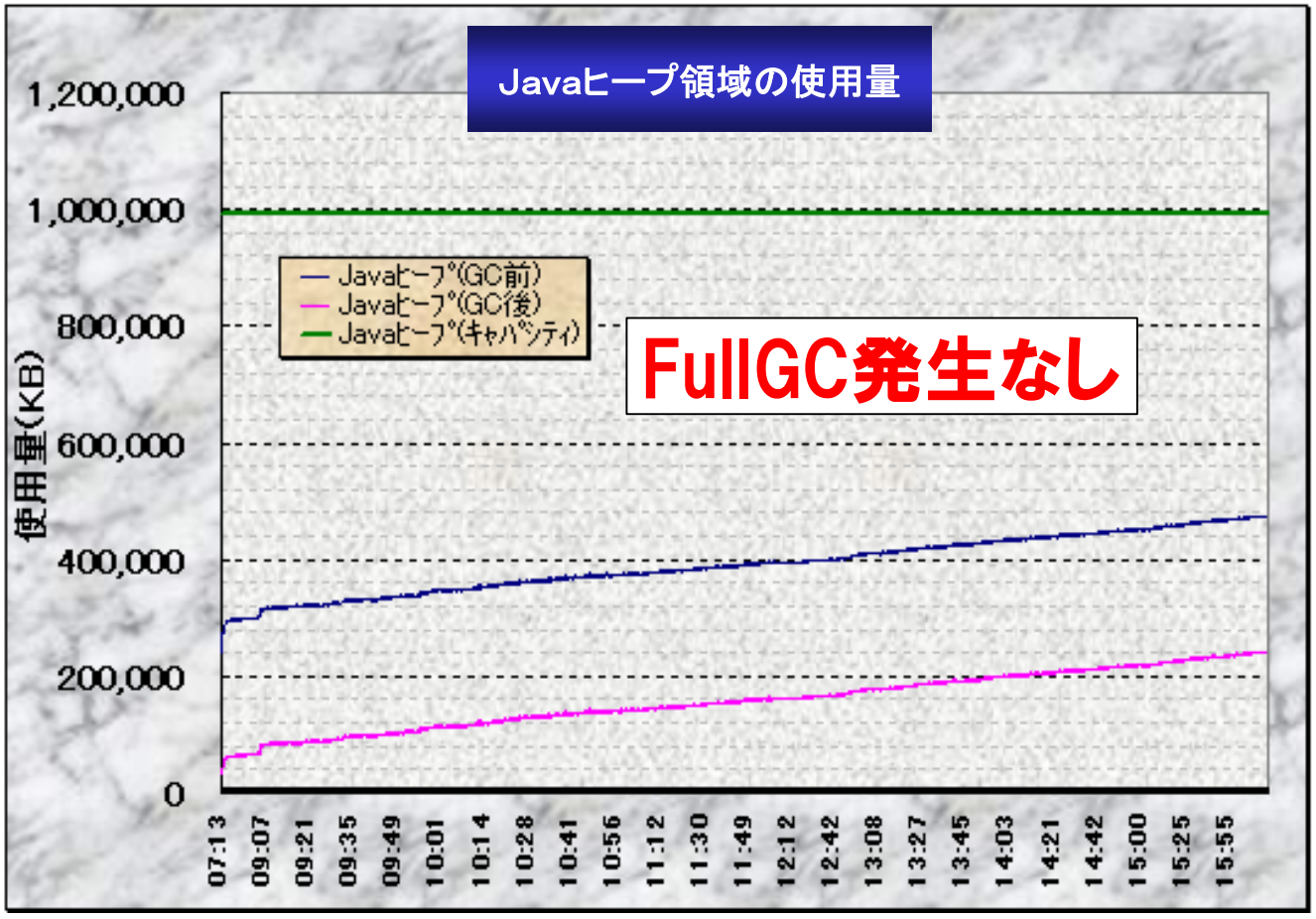
パラレルGC

コンカレントGC

⋮

GCアルゴリズムの変更ではFullGC発生は回避不可のため、
ヒープ管理方式として***明示管理ヒープ方式***
（Explicit Heap方式：略称***Eヒープ方式***）を開発！

2.ガベージコレクションの仕組み
-安定時のメモリ使用状況(グラフ)-



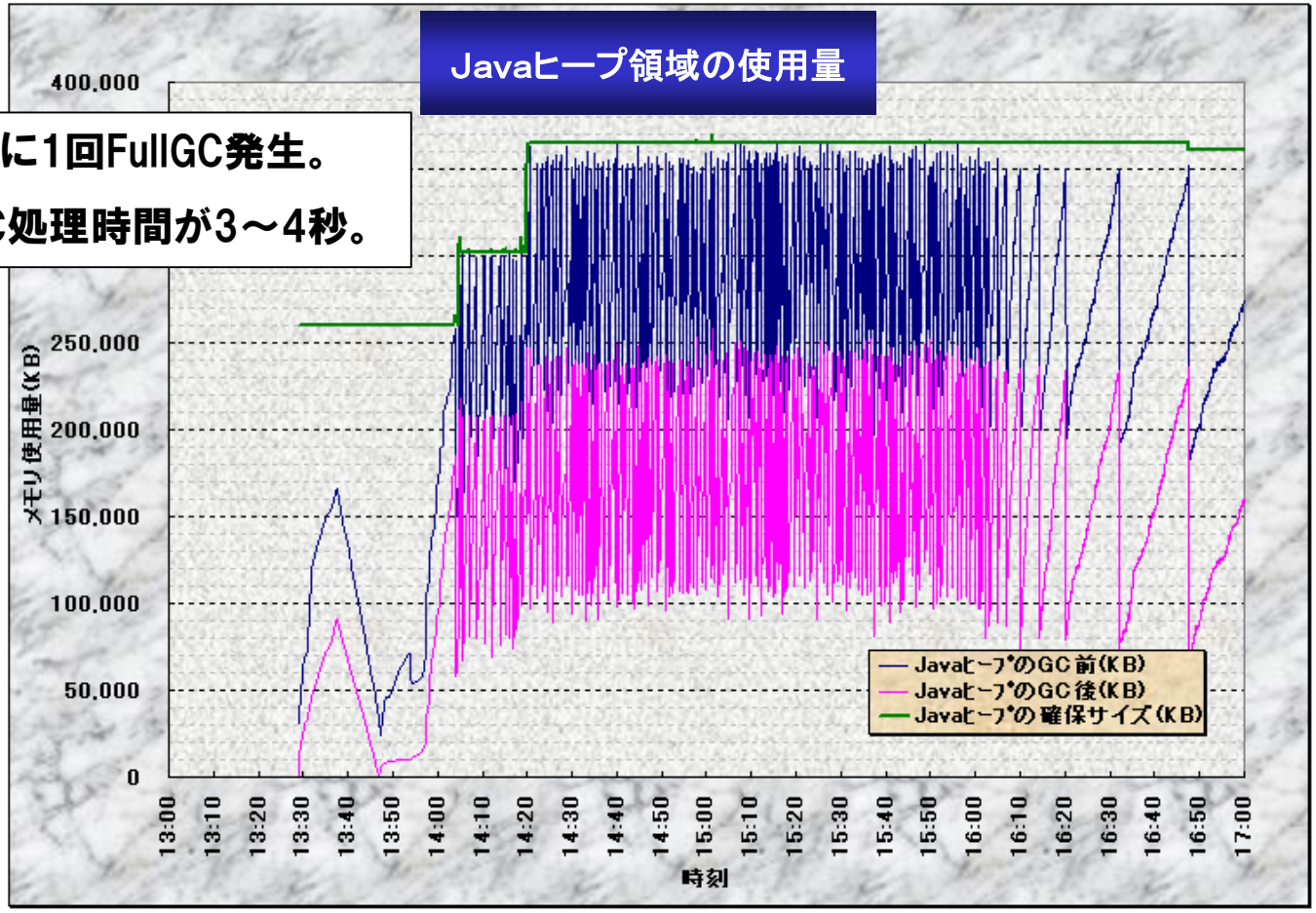
安定時のグラフ（非月末）

2.ガベージコレクションの仕組み
-GC多発時のメモリ使用状況(グラフ)-

HITACHI
Inspire the Next

10～15秒おきに1回FullGC発生。

1回あたりのGC処理時間が3～4秒。



GC多発時のグラフ(月末)

HITACHI
Inspire the Next

3

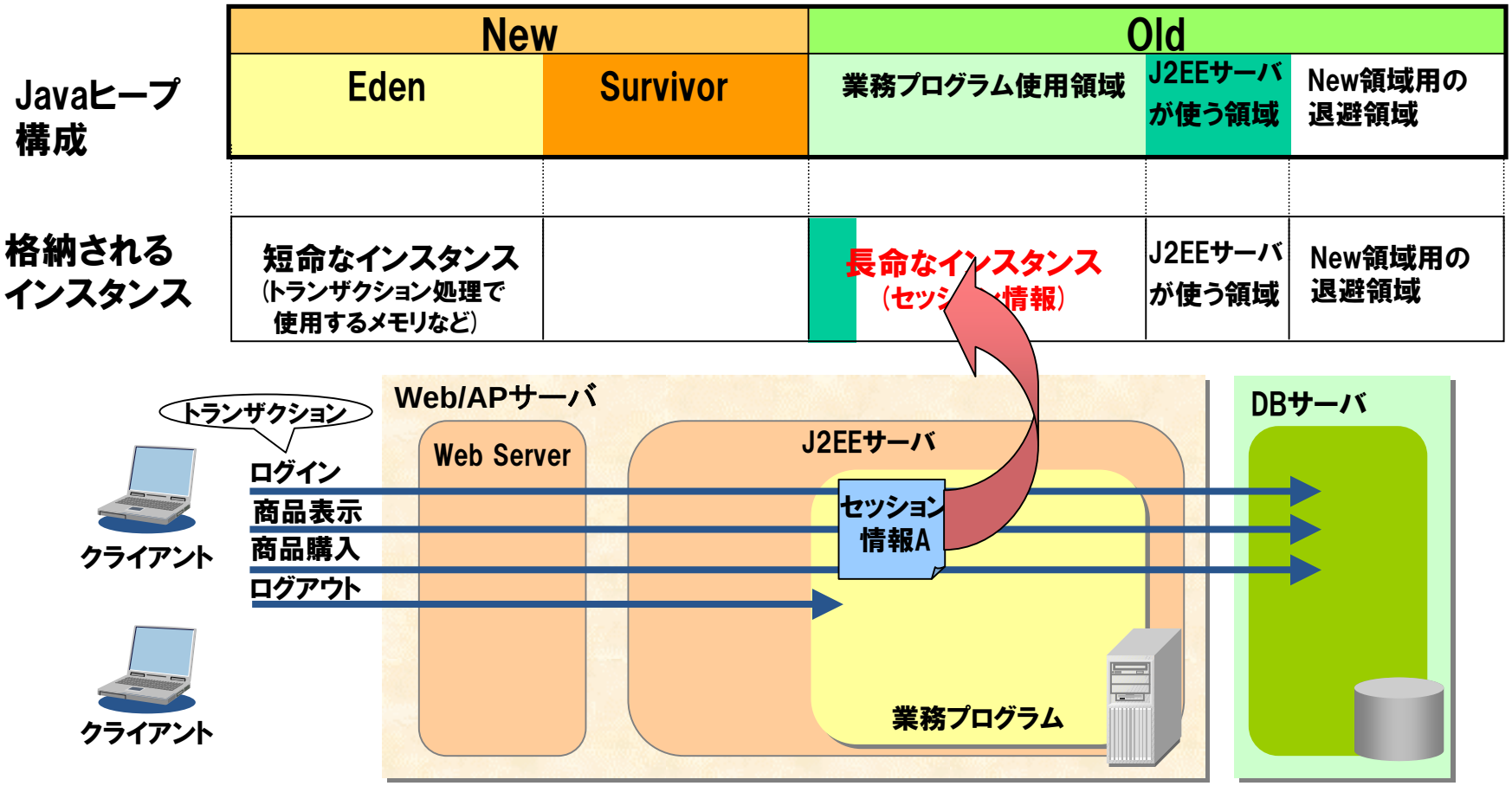
「FullGCレス」を実現する最新技術

uVALUE

3.「FullGCレス」を実現する最新技術
-ヒープ領域ごとの用途-

HITACHI
Inspire the Next

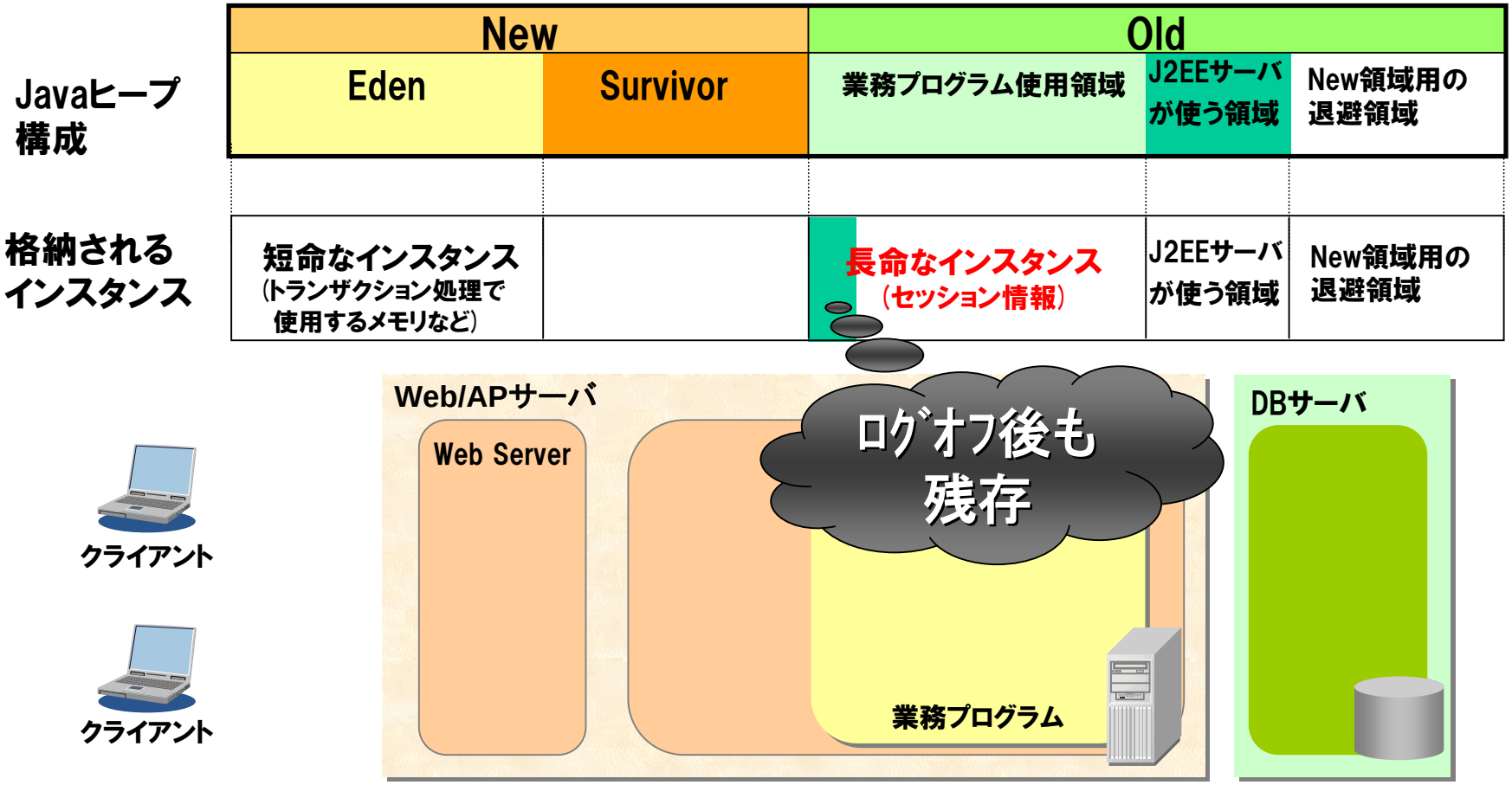
- New領域およびOld領域に格納されるインスタンスの違いに着目



3.「FullGCレス」を実現する最新技術
-ヒープ領域ごとの用途-

HITACHI
Inspire the Next

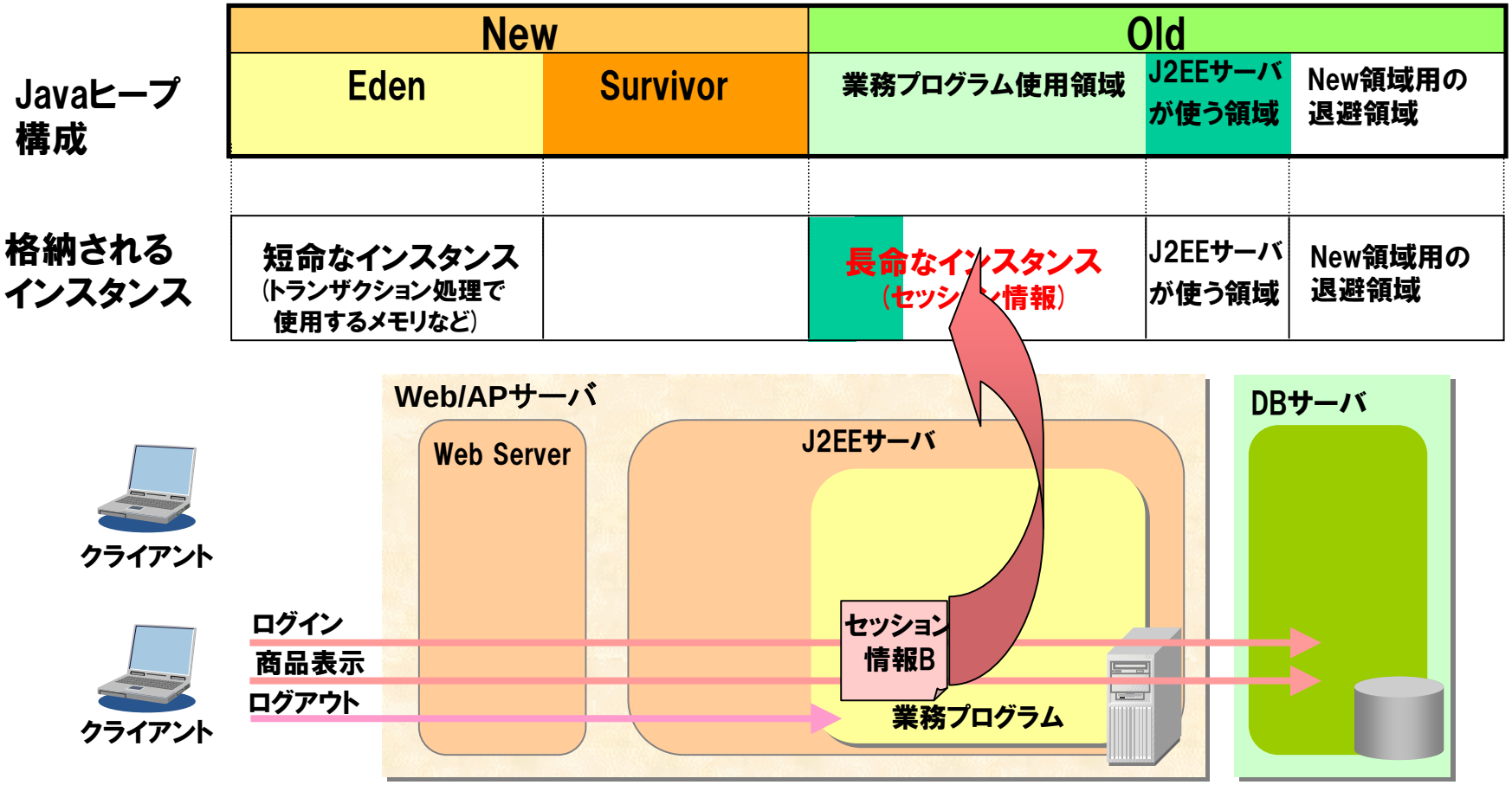
- New領域およびOld領域に格納されるインスタンスの違いに着目



3.「FullGCレス」を実現する最新技術
-ヒープ領域ごとの用途-

HITACHI
Inspire the Next

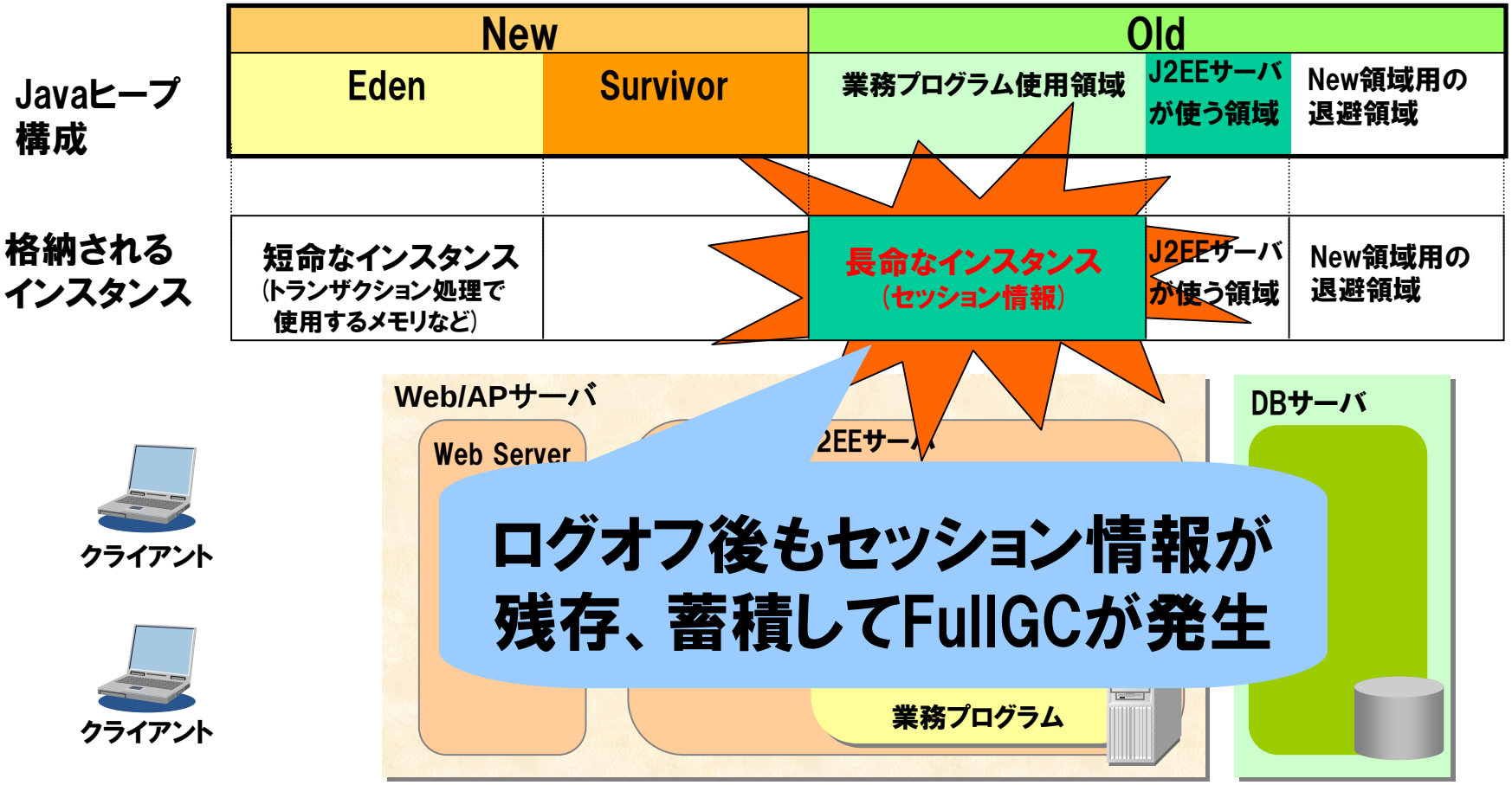
- New領域およびOld領域に格納されるインスタンスの違いに着目



3.「FullGCレス」を実現する最新技術
-ヒープ領域ごとの用途-

HITACHI
Inspire the Next

- New領域およびOld領域に格納されるインスタンスの違いに着目



3.「FullGCレス」を実現する最新技術

-Full GCレスでStop The Worldを解消！-

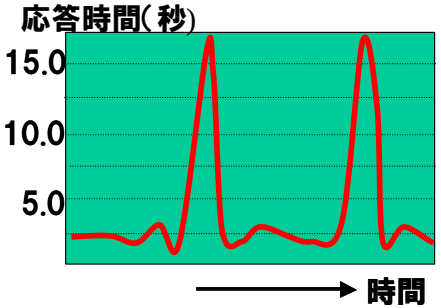
HITACHI
Inspire the Next

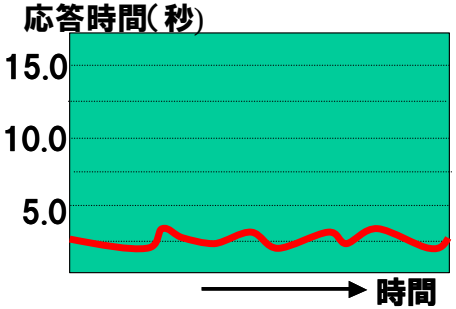
Full GCによるオンライン業務の一時的な停止は、
何とかならないものだろうか・・・

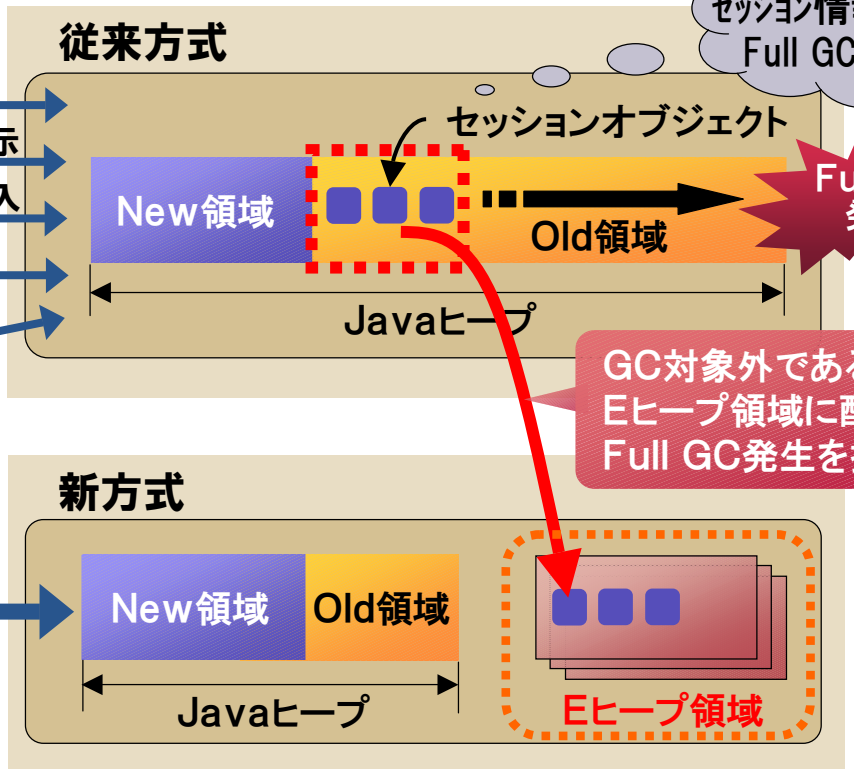
業界初！(*)
Full GCレスを実現

ポイント

- Webアプリケーションの変更無く、Full GCの発生を抑止。
  Full GCによるオンライン業務の一時停止を解消します。





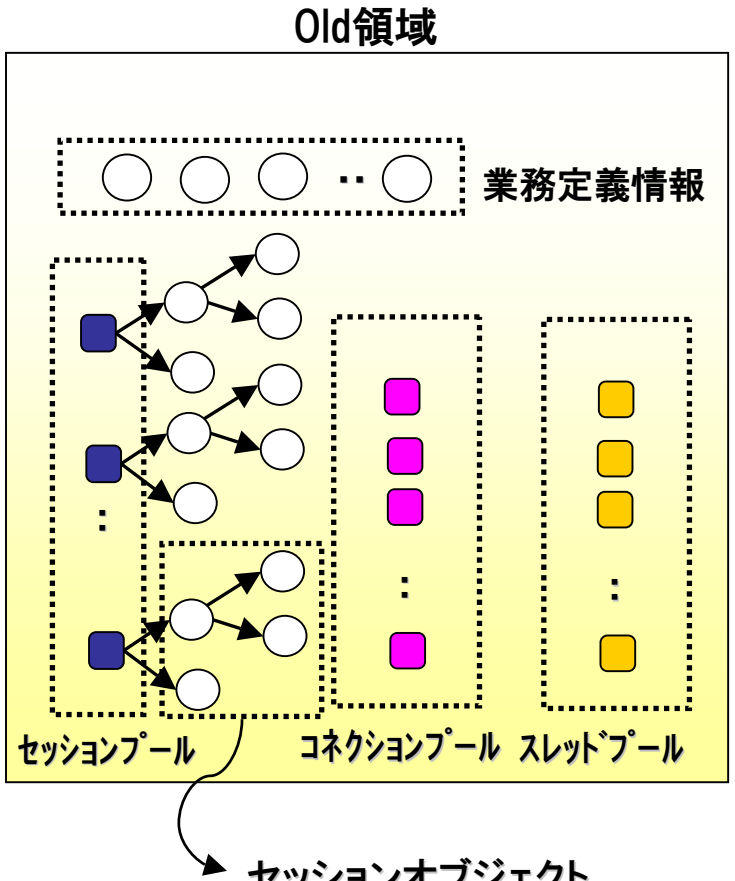


(*)・・・弊社調べ　2008年10月時点

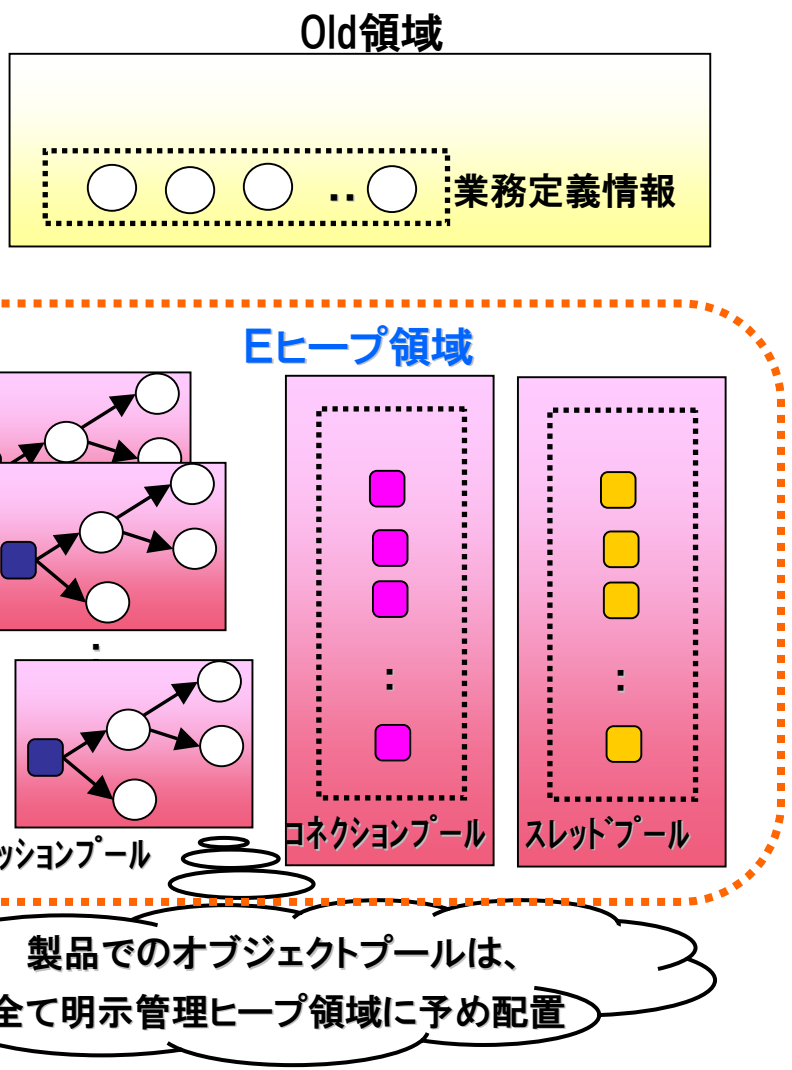
3.「FullGCレス」を実現する最新技術
-Eヒープ領域適用前後でのオブジェクトの配置-

HITACHI
Inspire the Next

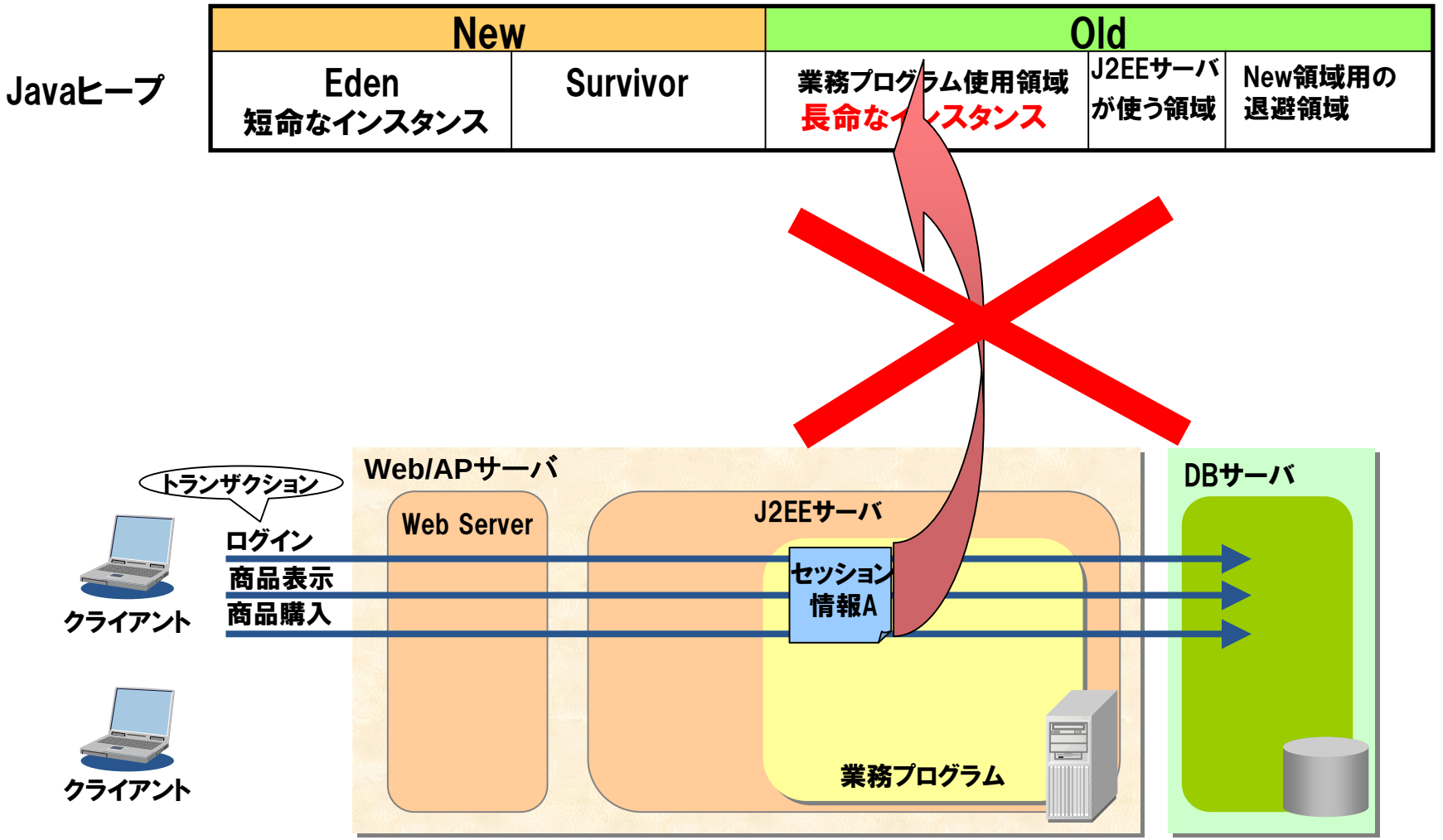
Eヒープ適用前の状態



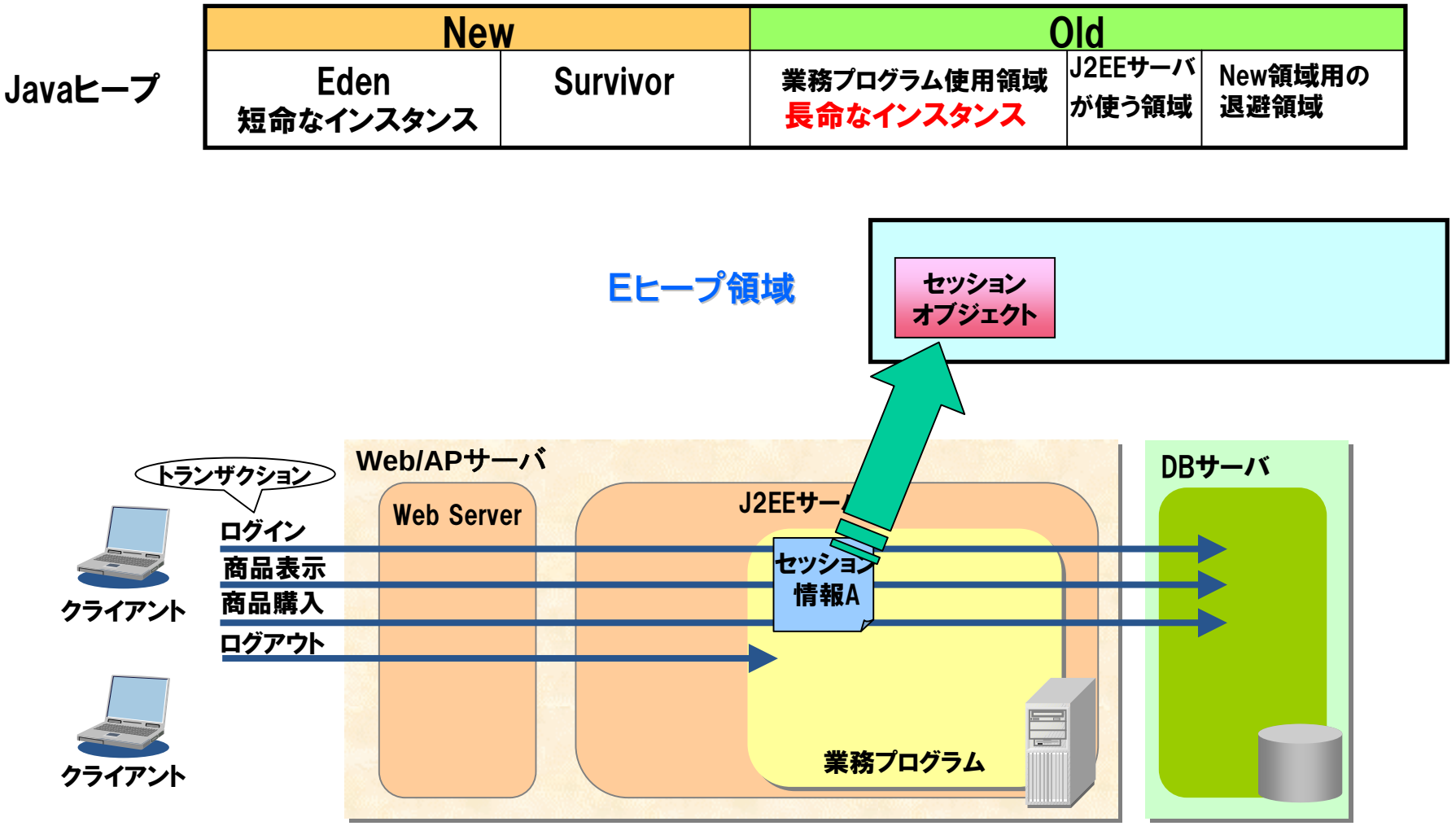
Eヒープ適用後の状態



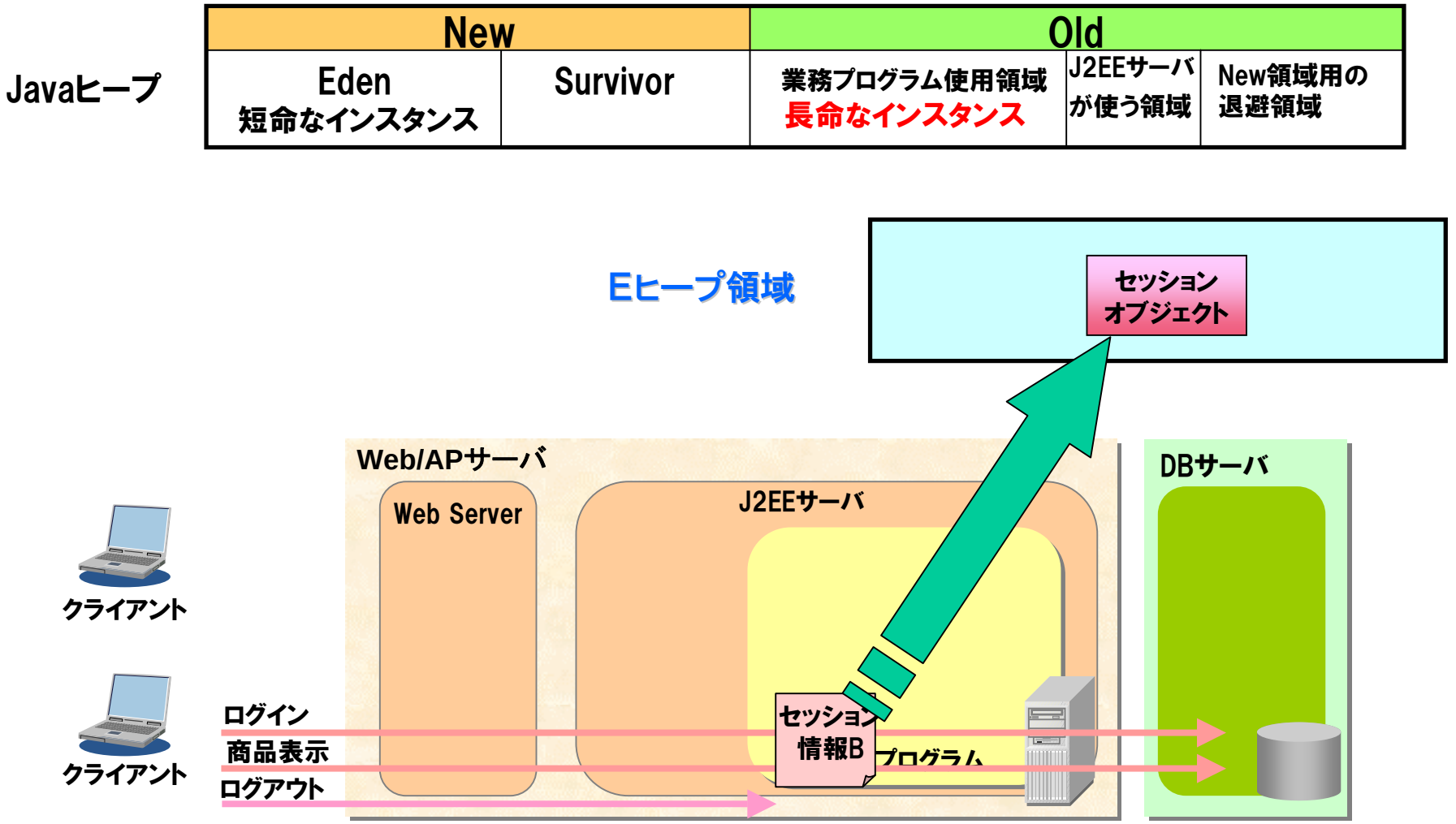
3.「FullGCレス」を実現する最新技術
-FullGCレス機能適用時の動き-



3.「FullGCレス」を実現する最新技術
-FullGCレス機能適用時の動き-



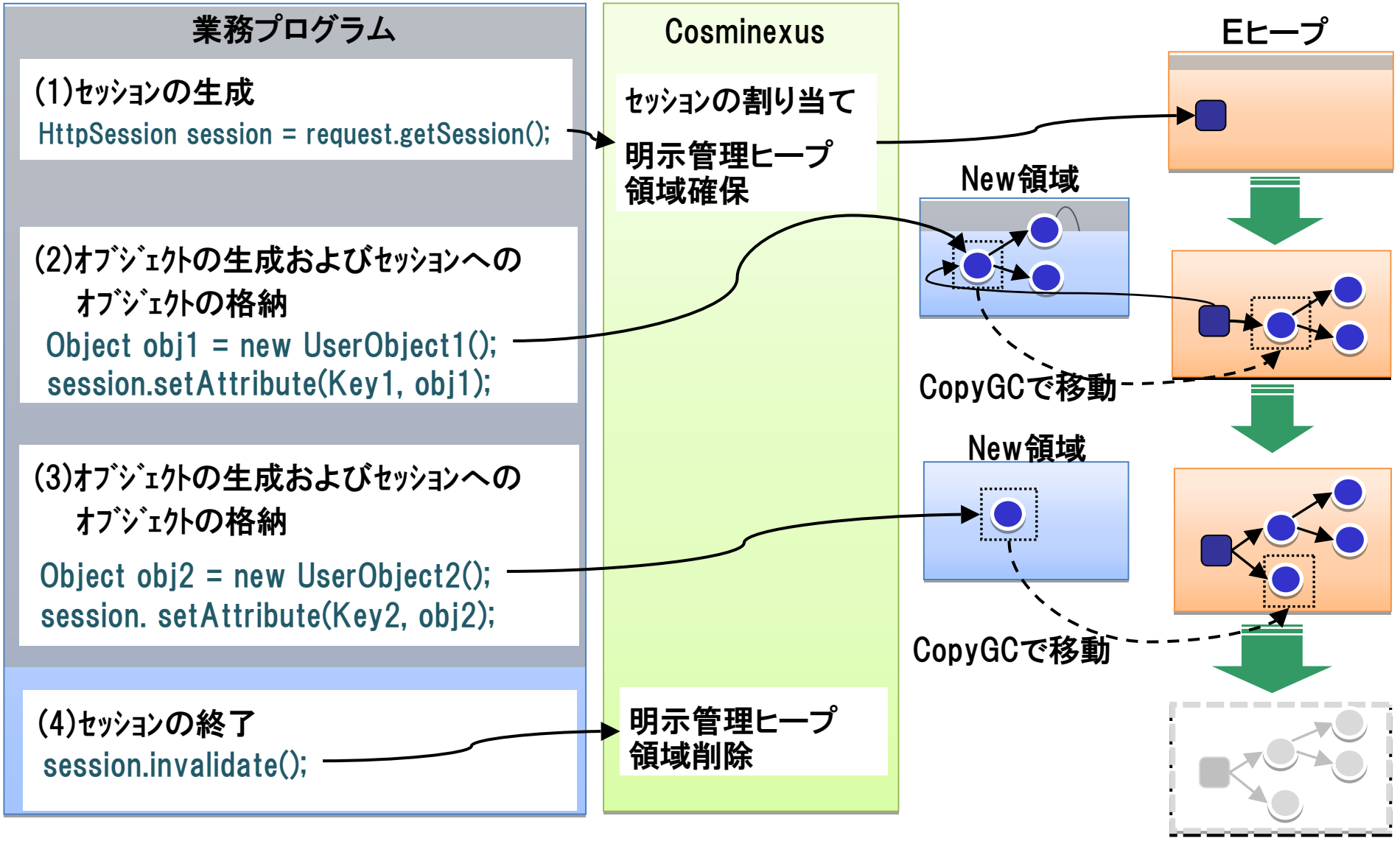
3.「FullGCレス」を実現する最新技術
-FullGCレス機能適用時の動き-



3.「FullGCレス」を実現する最新技術
-FullGCレス機能適用時の動き-

3.「FullGCレス」を実現する最新技術
-FullGCレス機能の仕組みとは-

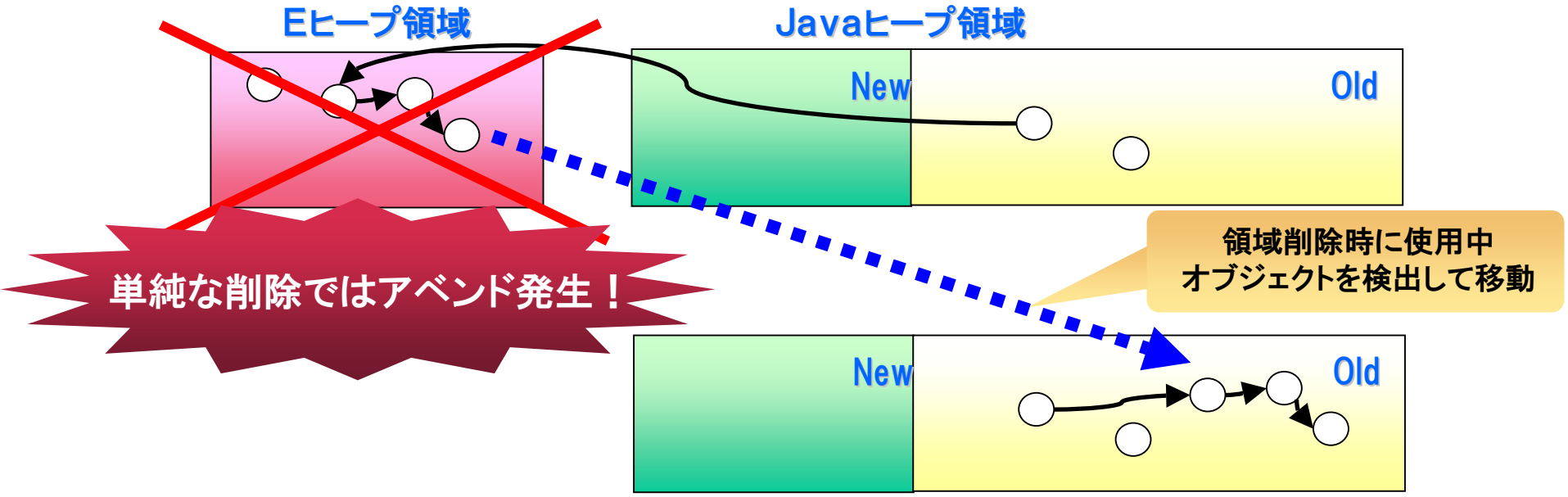
特許出願済



3.「FullGCレス」を実現する最新技術
-FullGCレス機能の仕組みとは-

特許出願済

- 削除するEヒープ内に共通データなど使用中データが存在する場合
  →単純な削除ではトラブル(領域外参照)が発生



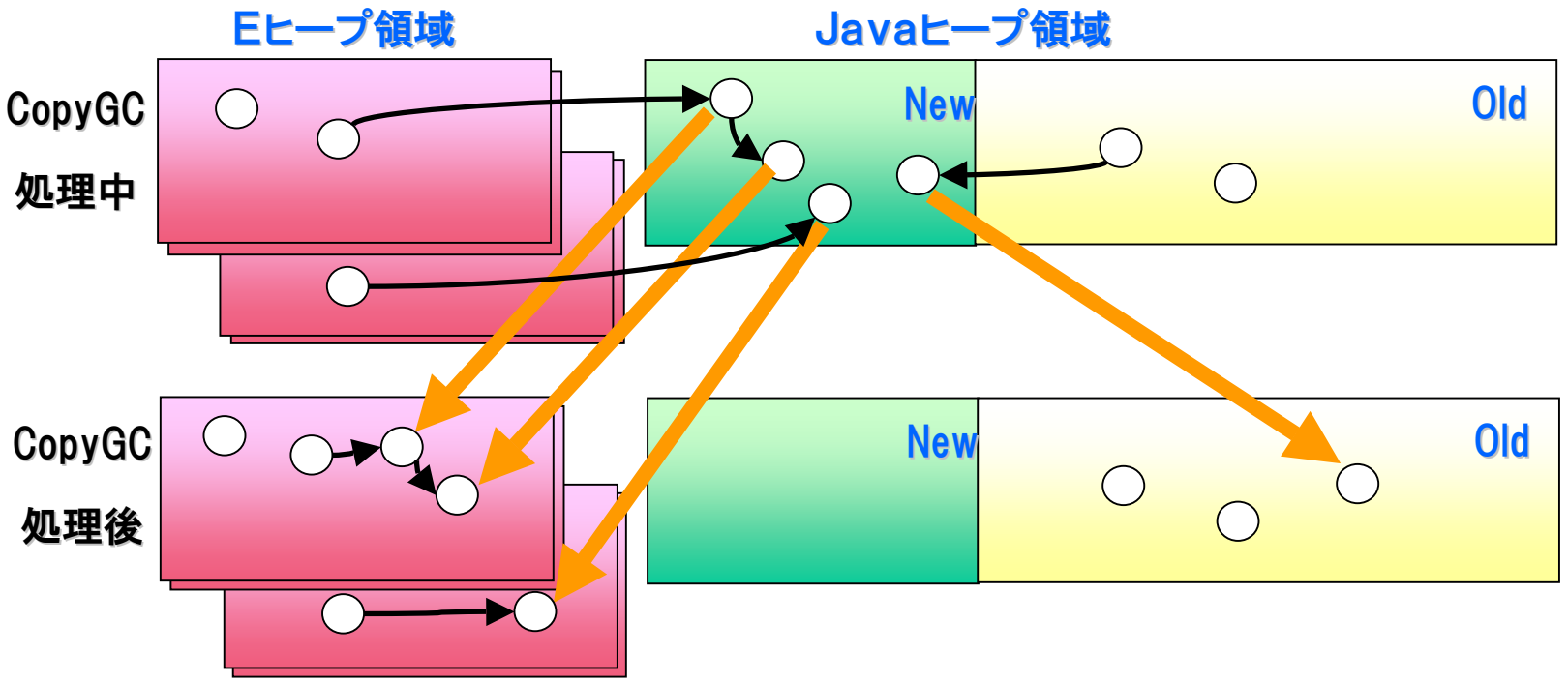
Eヒープ内に使用中データが存在する場合にはJavaヒープへの移動を行い、そのEヒープを削除してしまっても問題なく動作

→ ***システム堅牢性を確保***

3.「FullGCレス」を実現する最新技術
-FullGCレス機能適用時のオーバヘッド-

特許出願済

- FullGCレス機能適用時CopyGC処理のコスト
  → 従来のCopyGC処理時間の最大30%増



CopyGC時にEヒープ領域への移動候補の解析及び
移動先がJavaヒープかEヒープかの移動先判定を行う

3.「FullGCレス」を実現する最新技術
-FullGCレス機能適用時のオーバヘッド-

特許出願済

[単価極小(数十ms～数百ms)のCopyGCに対して30%増のオーバヘッドのみ]

FullGCの停止処理コストを広く浅くオンライン業務処理全域に割り振るのではなく、FullGC対象領域を縮小させる という抜本的な方式のため、処理全体に占める増加オーバーヘッドはCopyGCオーバヘッド増という限定されたものとなる。

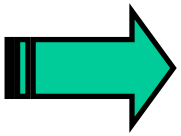
[CopyGC時間増によるスループット劣化の試算例]

測定単位時間:10:00～15:45(20700秒)

CopyGC回数:1658回

平均CopyGC時間:0.253秒/回

処理件数:548819件



平均CopyGC時間:0.328秒/回 (30%増)

処理件数:545486件 (0.61%劣化)

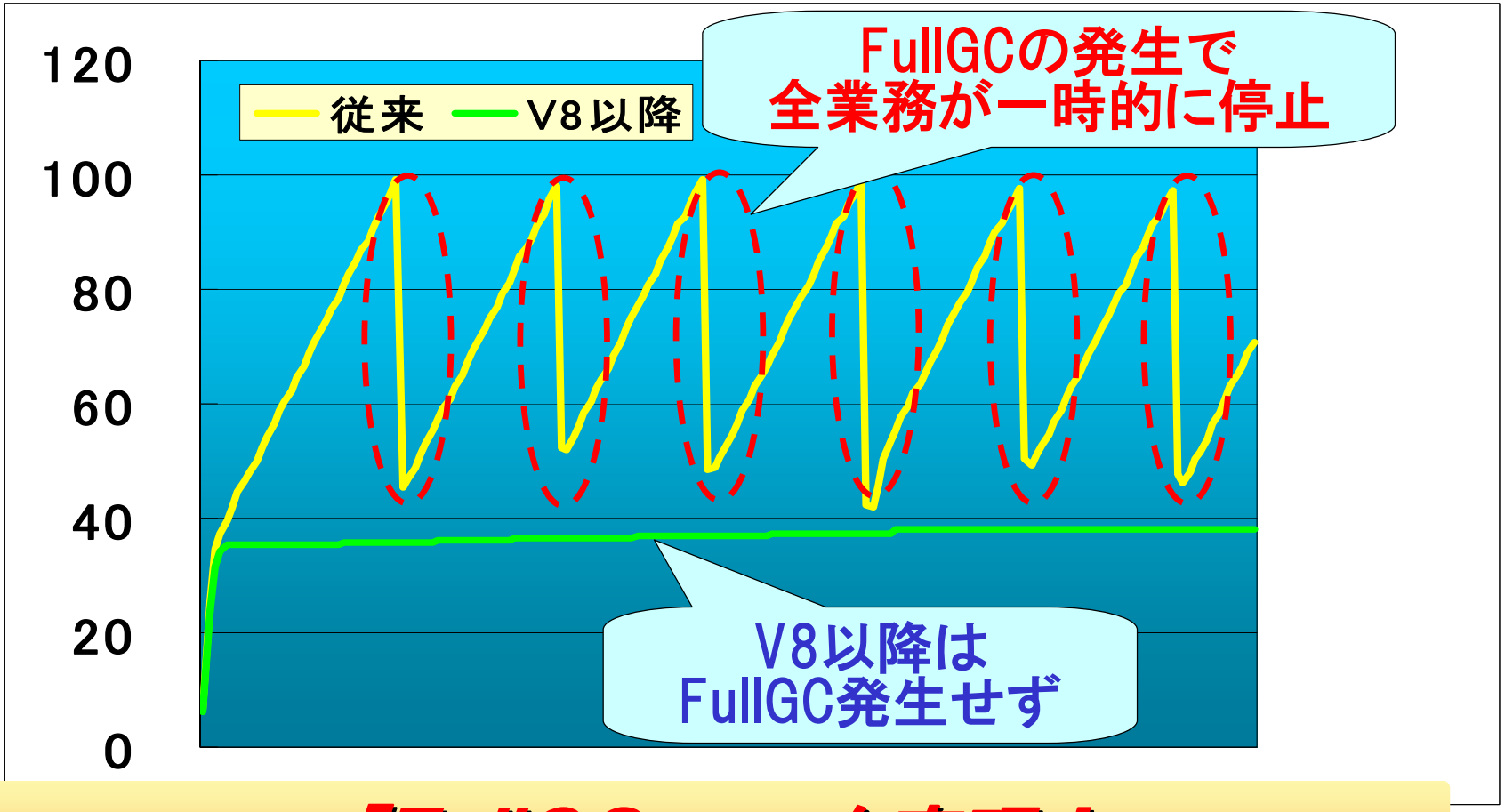
→処理性能への影響

1%未満

3.「FullGCレス」を実現する最新技術
-FullGCレス機能の効果例-

特許出願済

- Cosminexus動作時のメモリ使用量変化: A銀行殿縮小模擬環境



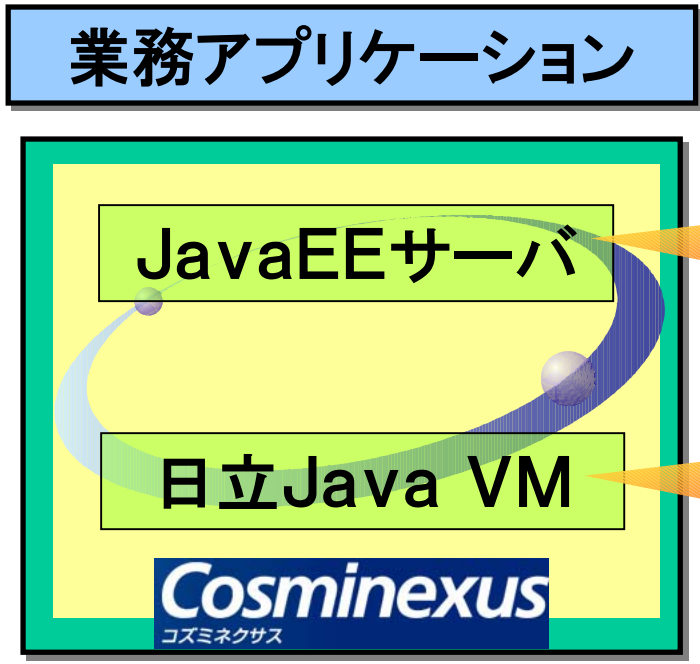
「FullGCレス」を実現！

GC対象外領域でメモリ大容量化対応!

3.「FullGCレス」を実現する最新技術
-Cosminexusでしか実現できない理由-

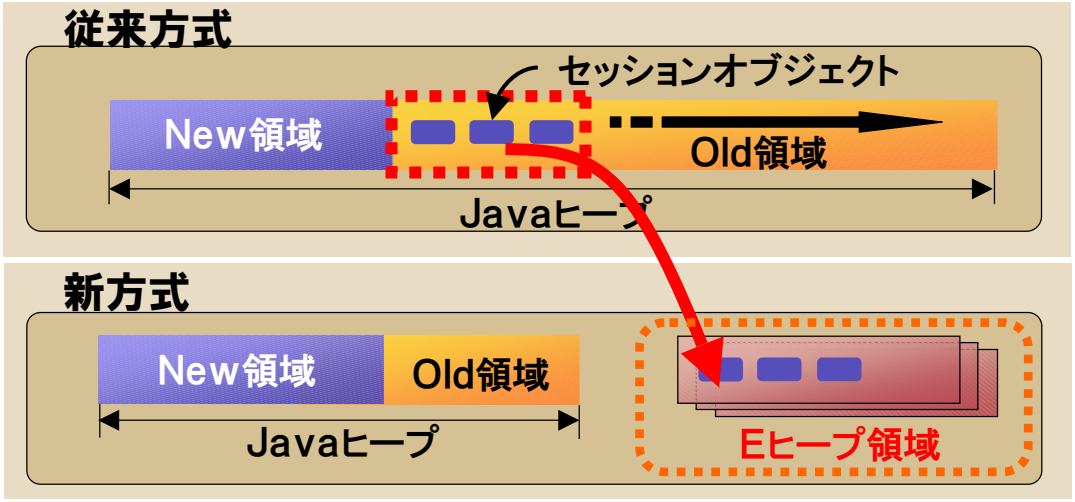
HITACHI
Inspire the Next

JavaEEサーバとJavaVMの連携により業務アプリの変更無しに「FullGCレス」を実現可能に！



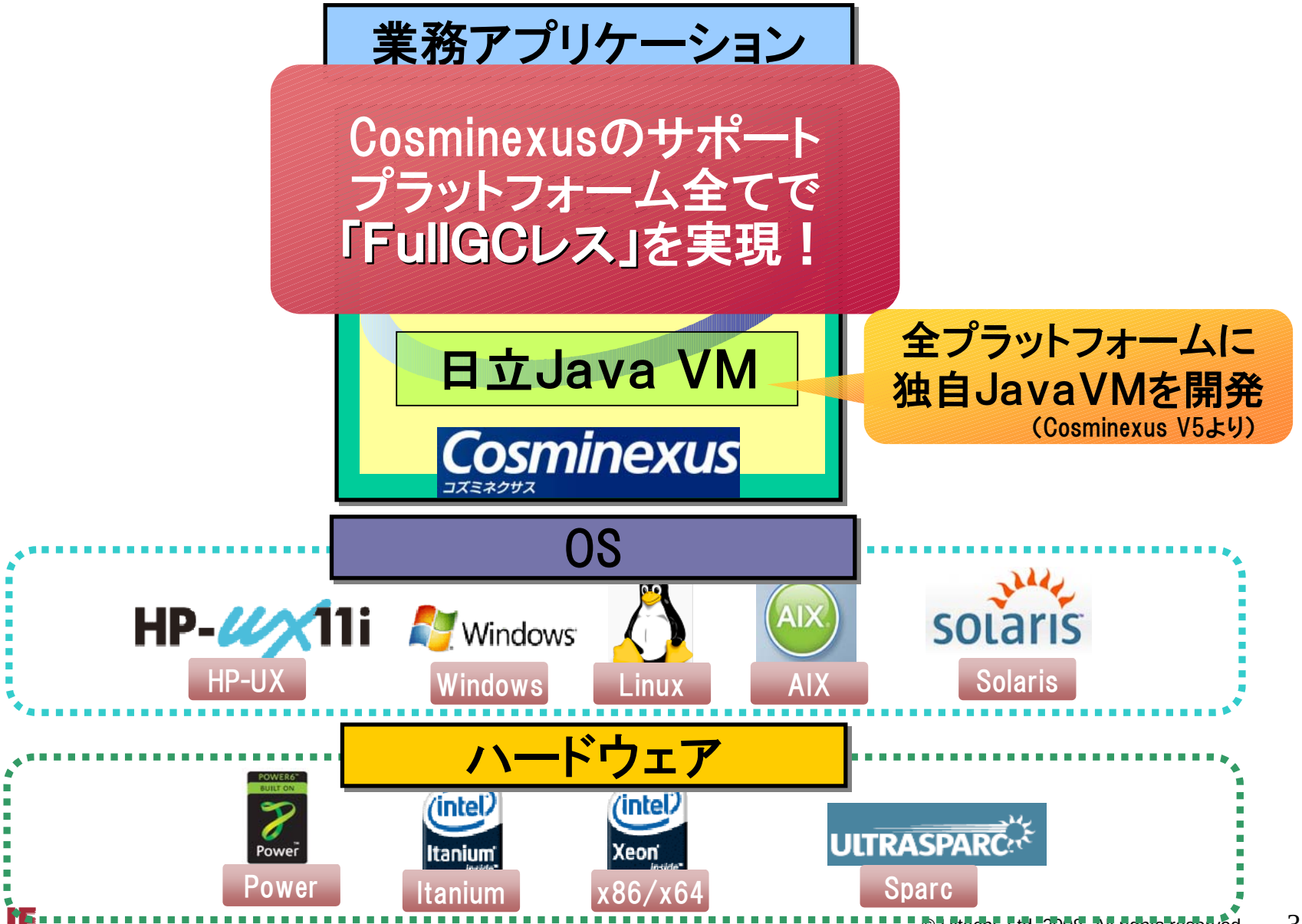
セッションオブジェクトを自動Eヒープ化

Eヒープ領域を追加



3.「FullGCレス」を実現する最新技術
-Cosminexusでしか実現できない理由-

HITACHI
Inspire the Next



HITACHI
Inspire the Next

4

「FullGCレス」の効果(ﾃﾞﾓﾝｽﾄﾚｰｼｮﾝ)

uVALUE

HITACHI
Inspire the Next

5

メモリトラブルを起こさないためには

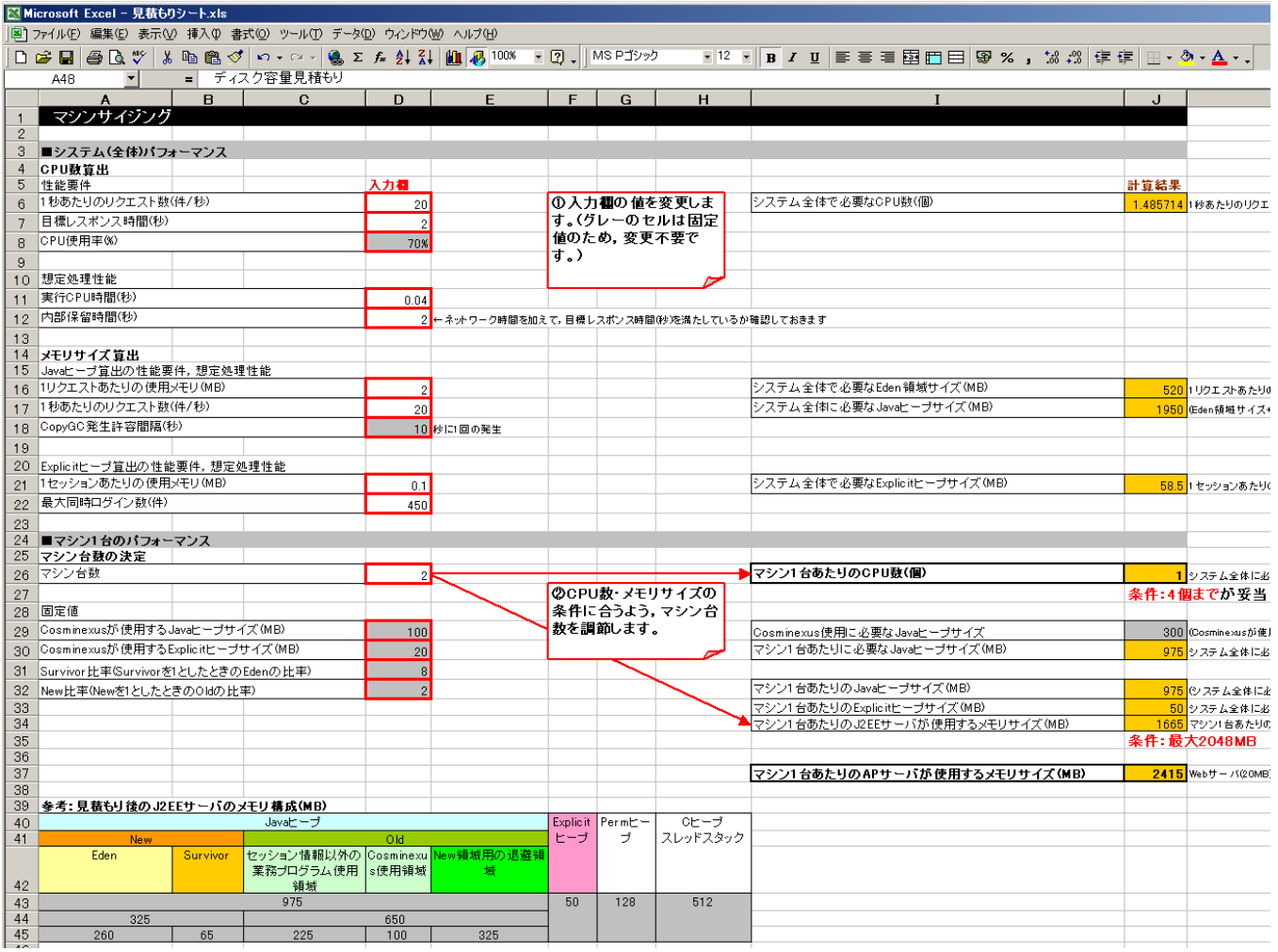
uVALUE

5.メモリトラブルを起こさないためには
-パフォーマンス見積もりシート-

HITACHI
Inspire the Next

- 「パフォーマンス見積もりシート」を用いてラクラク見積もり

赤枠の項目の一部を入力すると、必要なメモリサイズ、CPU数などを自動サイジング



Microsoft Excel - 見積もりシート.xls

ファイル(F) 編集(E) 表示(V) 挿入(I) 書式(O) ツール(T) データ(D) ウィンドウ(W) ヘルプ(H)

A48 = ディスク容量見積もり

	A	B	C	D	E	F	G	H	I	J
1	マシンサイジング									
2										
3	■システム(全体)パフォーマンス									
4	CPU数算出									
5	性能要件			入力欄						計算結果
6	1秒あたりのリクエスト数(件/秒)			20					システム全体で必要なCPU数(個)	1.485714 1秒あたりのリクエ
7	目標レスポンス時間(秒)			2						
8	CPU使用率(%)			70%						
9										
10	想定処理性能									
11	実行CPU時間(秒)			0.04						
12	内部保留時間(秒)			2	←ネットワーク時間を加えて，目標レスポンス時間(秒)を満たしているか確認しておきます					
13										
14	メモリサイズ算出									
15	Javaヒープ算出の性能要件，想定処理性能									
16	1リクエストあたりの使用メモリ(MB)			2					システム全体で必要なEden領域サイズ(MB)	520 1リクエストあたり
17	1秒あたりのリクエスト数(件/秒)			20					システム全体に必要なJavaヒープサイズ(MB)	1950 (Eden領域サイズ+
18	CopyGC発生許容間隔(秒)			10	秒に1回の発生					
19										
20	Explicitヒープ算出の性能要件，想定処理性能									
21	1セッションあたりの使用メモリ(MB)			0.1					システム全体で必要なExplicitヒープサイズ(MB)	58.5 1セッションあたり
22	最大同時ログイン数(件)			450						
23										
24	■マシン1台のパフォーマンス									
25	マシン台数の決定									
26	マシン台数			2					マシン1台あたりのCPU数(個)	1 システム全体に必
27										条件：4個までが妥当
28	固定値									
29	Cosminexusが使用するJavaヒープサイズ(MB)			100					Cosminexus使用に必要なJavaヒープサイズ	300 (Cosminexusが使
30	Cosminexusが使用するExplicitヒープサイズ(MB)			20					マシン1台あたりに必要なJavaヒープサイズ(MB)	975 システム全体に必
31	Survivor比率(Survivorを1としたときのEdenの比率)			8						
32	New比率(Newを1としたときのOldの比率)			2					マシン1台あたりのJavaヒープサイズ(MB)	975 (システム全体に必
33									マシン1台あたりのExplicitヒープサイズ(MB)	50 システム全体に必
34									マシン1台あたりのJ2EEサーバが使用するメモリサイズ(MB)	1665 マシン1台あたりの
35										条件：最大2048MB
36										
37									マシン1台あたりのAPサーバが使用するメモリサイズ(MB)	2415 Webサーバ(20MB)
38										
39	参考：見積もり後のJ2EEサーバのメモリ構成(MB)									

	Javaヒープ					Explicitヒープ	Permヒープ	Cヒープ スレッドスタック
41	New		Old					
42	Eden	Survivor	セッション情報以外の業務プログラム使用領域	Cosminexus使用領域	New領域用の退避領域			
43	975					50	128	512
44	325		650					
45	260	65	225	100	325			

①入力欄の値を変更します。(グレーのセルは固定値のため，変更不要です。)

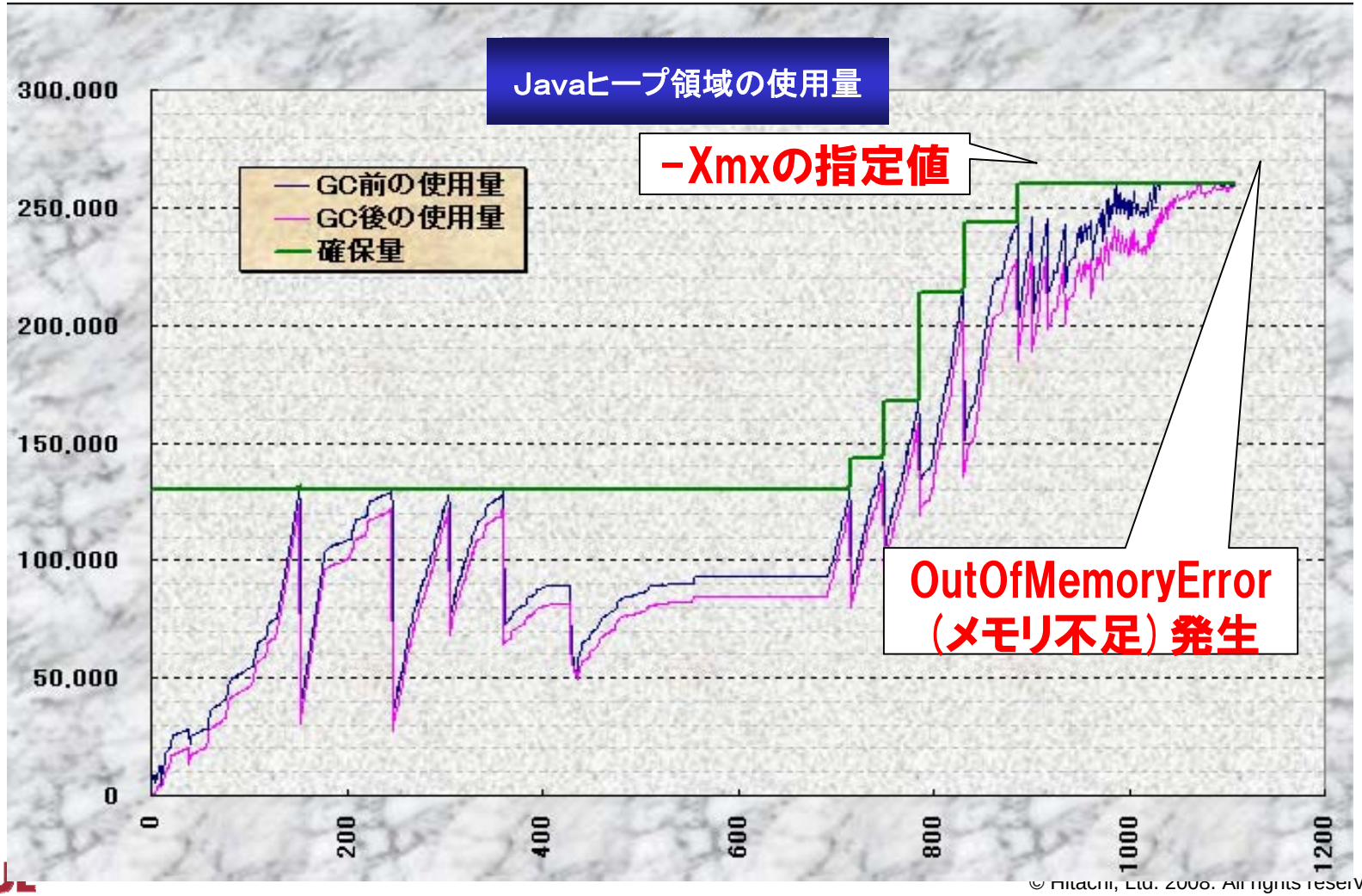
②CPU数・メモリサイズの条件に合うよう，マシン台数を調節します。

5.メモリトラブルを起こさないためには
-メモリリーク-

HITACHI
Inspire the Next

- **メモリリークとは…**

  プログラマが予想しないところでオブジェクトの参照が残ってしまい、解放されないJavaオブジェクトが継続的に増加してしまうこと



5.メモリトラブルを起こさないためには
-メモリリーク原因の特定- 「ヒーププロファイル機能」

HITACHI
Inspire the Next

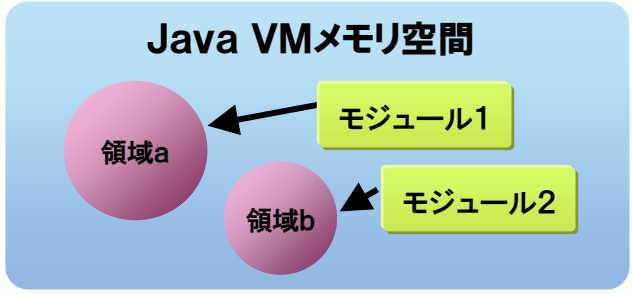
- メモリリークしているが、原因が特定できない。

ポイント

- メモリの解放モレ(オブジェクト参照の解除モレ)の原因を容易に追求することが可能になります。
- JavaVM自身で情報出力を行うため、高い精度で解析を行うことが可能です。
- 通常実行時のオーバヘッドはゼロ。
- 情報取得時のオーバヘッドはFullGC1回分程度の低オーバヘッド。
- サーバーを再起動することなく情報を取得できます。

①領域単位のメモリ消費量の把握

領域aが多くのメモリを消費していることを確認します。



②領域を保持しているモジュールを把握

モジュール1が領域aを保持している。
→仕様／バグを調査

※ 実際は領域／モジュールともJavaのクラスインスタンスになります。

```
Total Size of Instances
-----------------------
___________Size__Instances__Class_______________
        1437424      15809 [Ljava.lang.Class;
         525120       7408 java.util.HashMap$Entry
         502000       7094 java.util.HashMap
         500248       7012 [Ljava.util.HashMap$Entry;
         486336       4017 java.lang.ref.SoftReference
         394760       4658 java.util.HashSet
         394328       4648 java.lang.Shutdown$WrappedHook
…
```

```
Reference of class classC
-------------------------
classA(0x10766840)[Eden]
  classX(0x10766998)[Eden]
------
classB(0x10766840)[Eden]
  classC(0x10766858)[Tenured]
    classD(0x10766968)[Eden]
      classX(0x10766a28)[Survivor]
------
classE(0x10766840)[Eden]
  classA(0x10766920)[Survivor]
    classX(0x06aa0020)[EM(eid=1)]
```

classCをポイントしているすべてのクラスが一覧として出力されます。ヒープ領域名も表示します

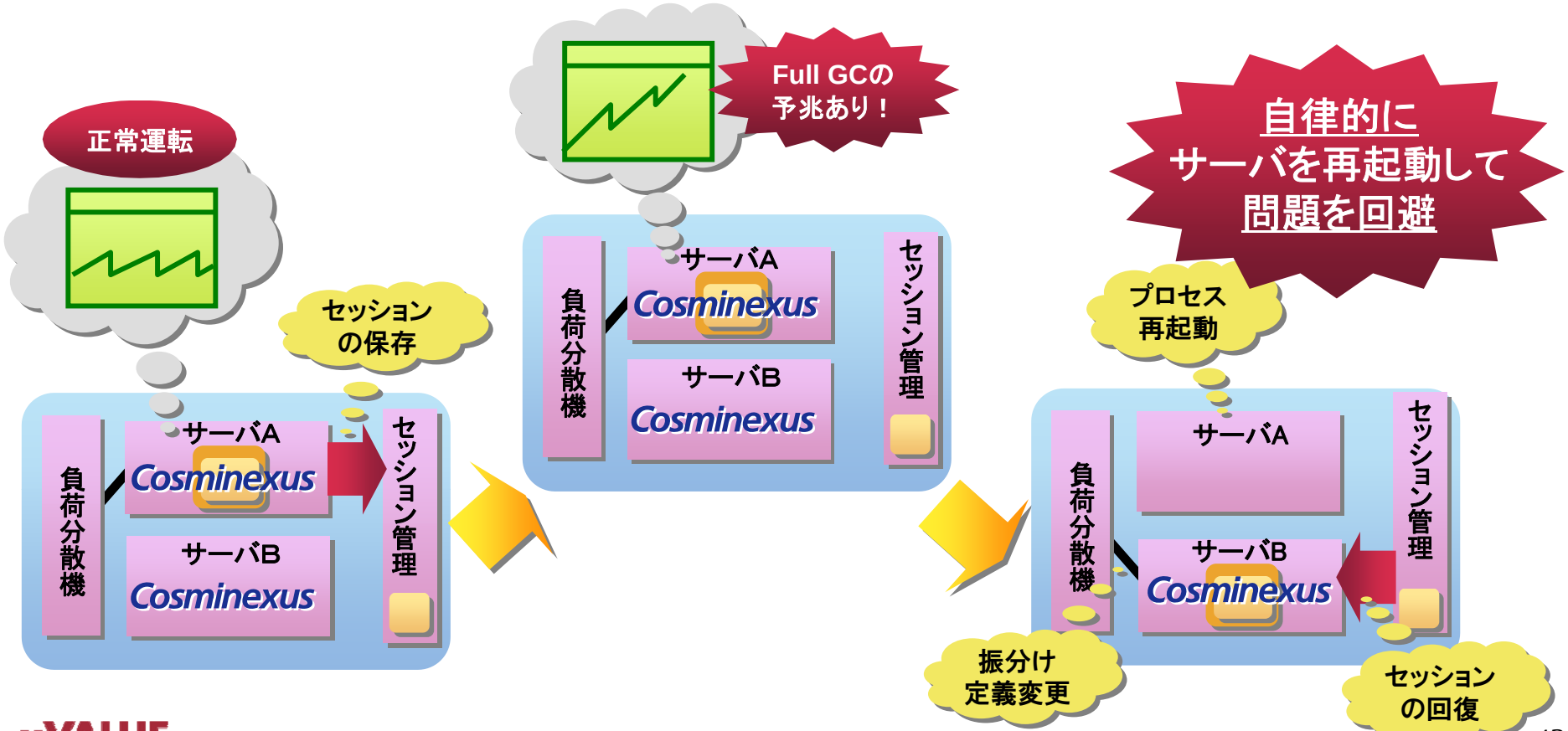
5.メモリトラブルを起こさないためには
-FullGC発生によるスローダウンの回避-

HITACHI
Inspire the Next

- FullGC発生によるスローダウンを事前に検知して回避したい。

ポイント

- FullGCの発生を事前に検知し自律的に再起動することで、トラブルを未然に防止。
- セッション情報を引継ぐため，業務を継続することができる。



まとめ

HITACHI
Inspire the Next

- Cosminexus V8は、Webシステムの“Stop the world”解消のために「FullGCレス」を実現する機能を新たに搭載した。アプリの変更なく、誰でも恩恵を享受できる。
- 日立JavaVMの明示管理ヒープと、Cosminexus JavaEEサーバとの密連携という高度な独自技術の結晶として、「FullGCレス機能」は実現されている。
- メモリリークなどのアプリ不良によって、メモリトラブルが発生する場合もある。これについての調査方法は確立されており、トラブルを暫定回避する機能もある。

他社所有名称に対する表示

HITACHI
Inspire the Next

Cosminexus ホームページ
http://www.hitachi.co.jp/cosminexus/
http://www.cosminexus.com/

謝辞および他社所有名称に対する表示

《他社所有名称に対する表示》

- Java 及びすべてのJava関連の商標及びロゴは，米国及びその他の国における米国Sun Microsystems, Inc.の商標または登録商標です。
- その他記載の会社名、製品名は、それぞれの会社の商号、商標もしくは登録商標です。