Java VMへの処方箋

～先進のメモリ管理技術とは～

13-B-4

中島 恵

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

Developers Summit

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

Developers Summit

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

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

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

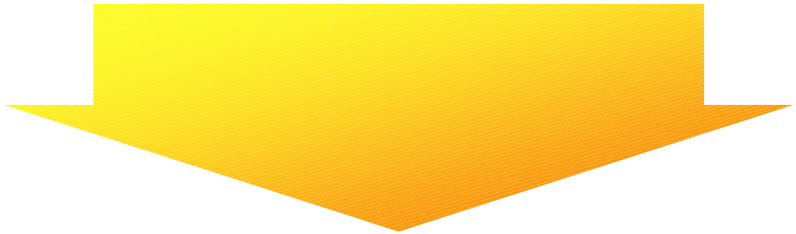
➔６４ビット化で対処はできるが、メモリ領域サイズの
増大に伴ってＧＣでの停止時間が長時間し、
システム性能劣化に陥る

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？

Developers Summit

Contents

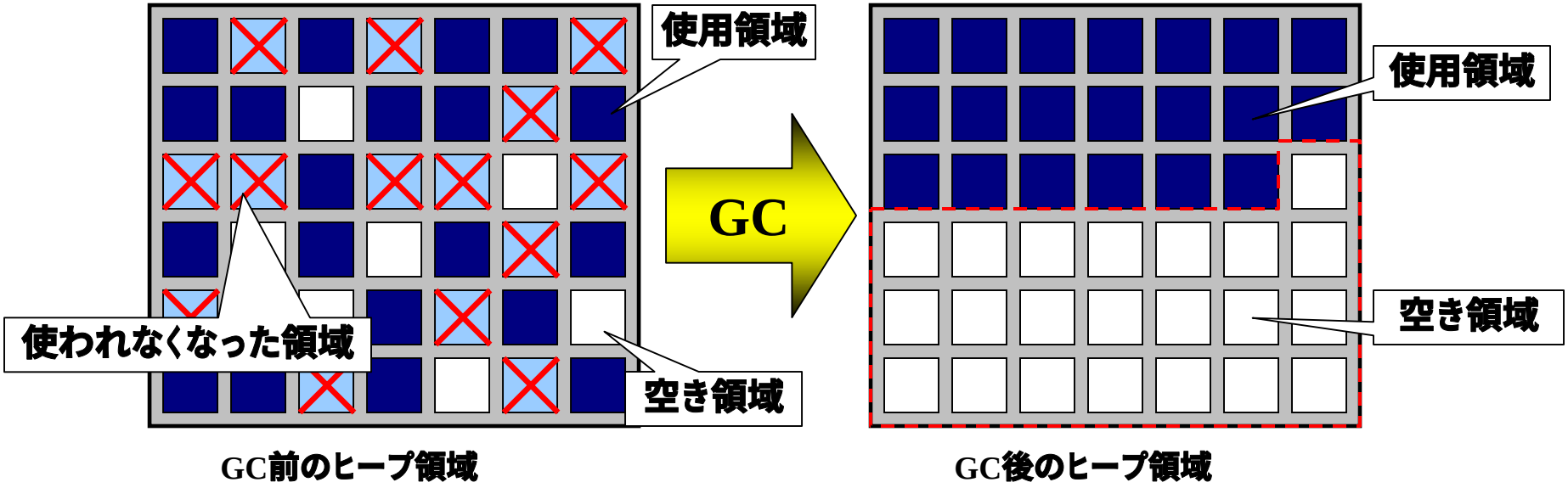
1. Javaのメモリ管理の課題とは
2. ＧＣ（ガベージコレクション）の仕組み
3. 「FullGCレス」を実現する最新技術
4. 「FullGCレス」の効果（ﾃﾞﾓﾝｽﾄﾚｰｼｮﾝ）
5. メモリトラブルを起こさないためには

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...

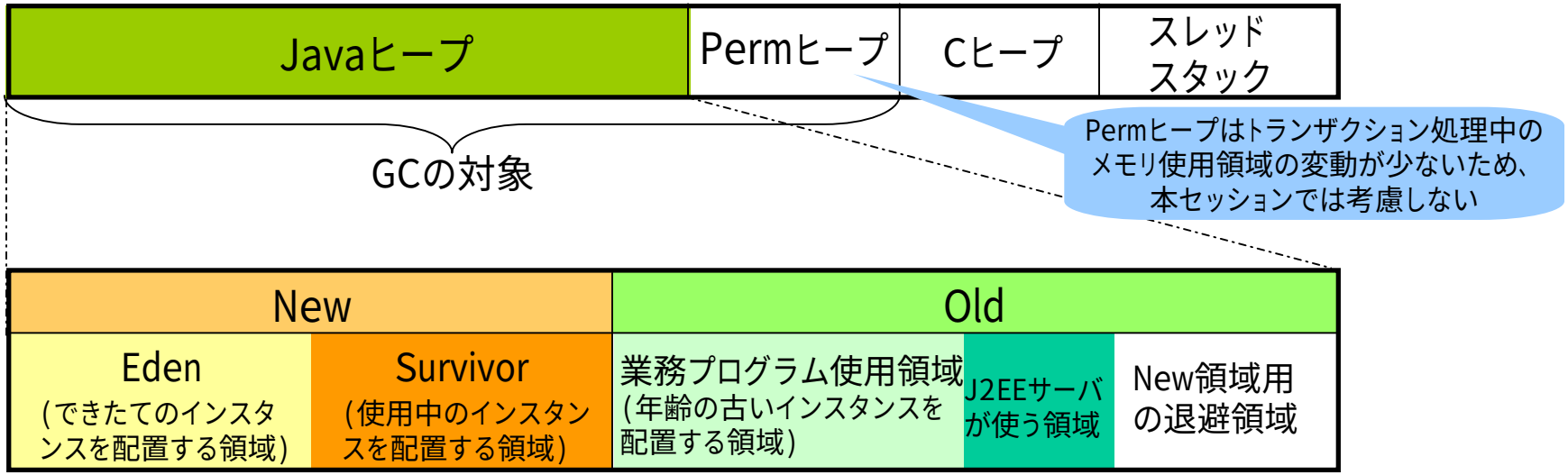
  JavaVMが管理するメモリ領域中の使用済みのメモリ領域を破棄し、
  空き領域を作ること



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

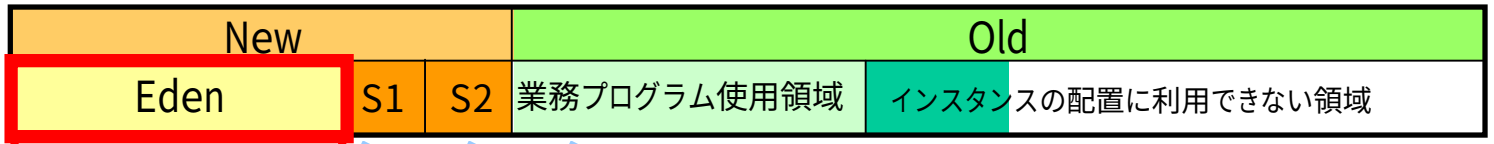
2.ガベージコレクションの仕組み -CopyGCとFullGC-

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



使用中のものは隣の領域へ

Edenの領域がいっぱいになるとCopyGCが起こる

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

Oldの業務プログラム使用領域がいっぱいになるとFullGCが起こる

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アルゴリズムは
複雑になるほど
コントロール不可と
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の位置付け-

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

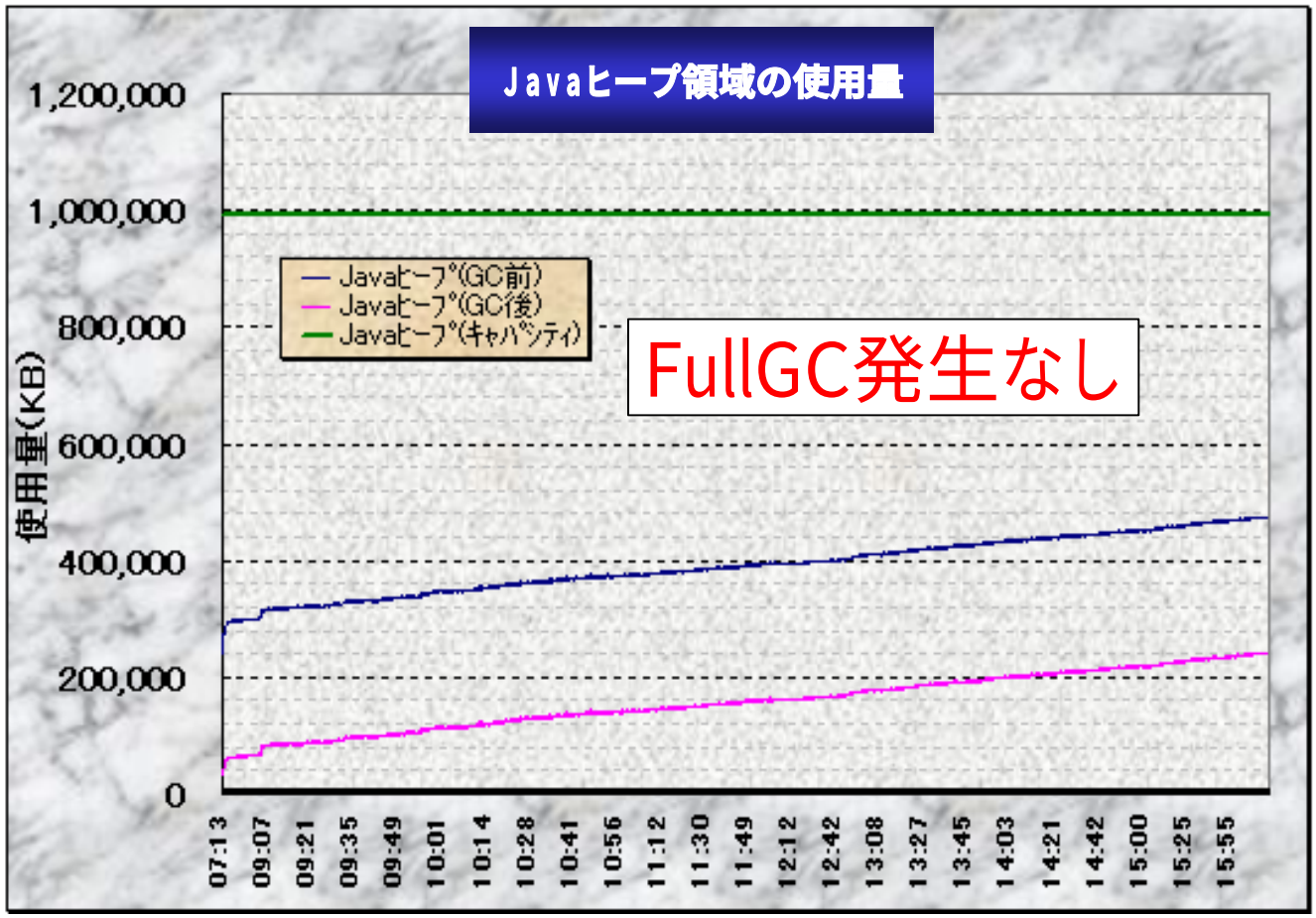
パラレルGC

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)-

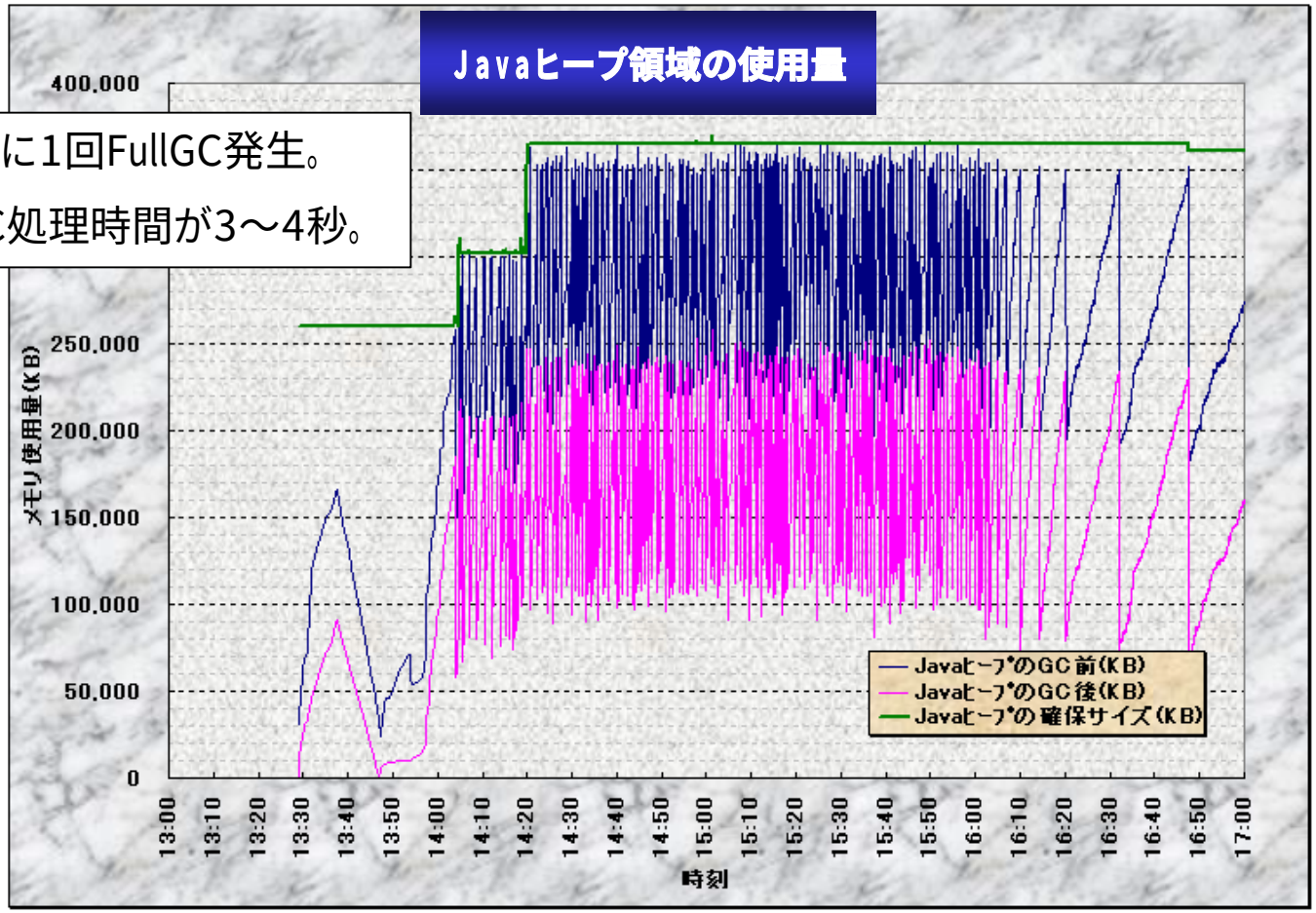


安定時のグラフ（非月末）

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)-

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)

Developers Summit

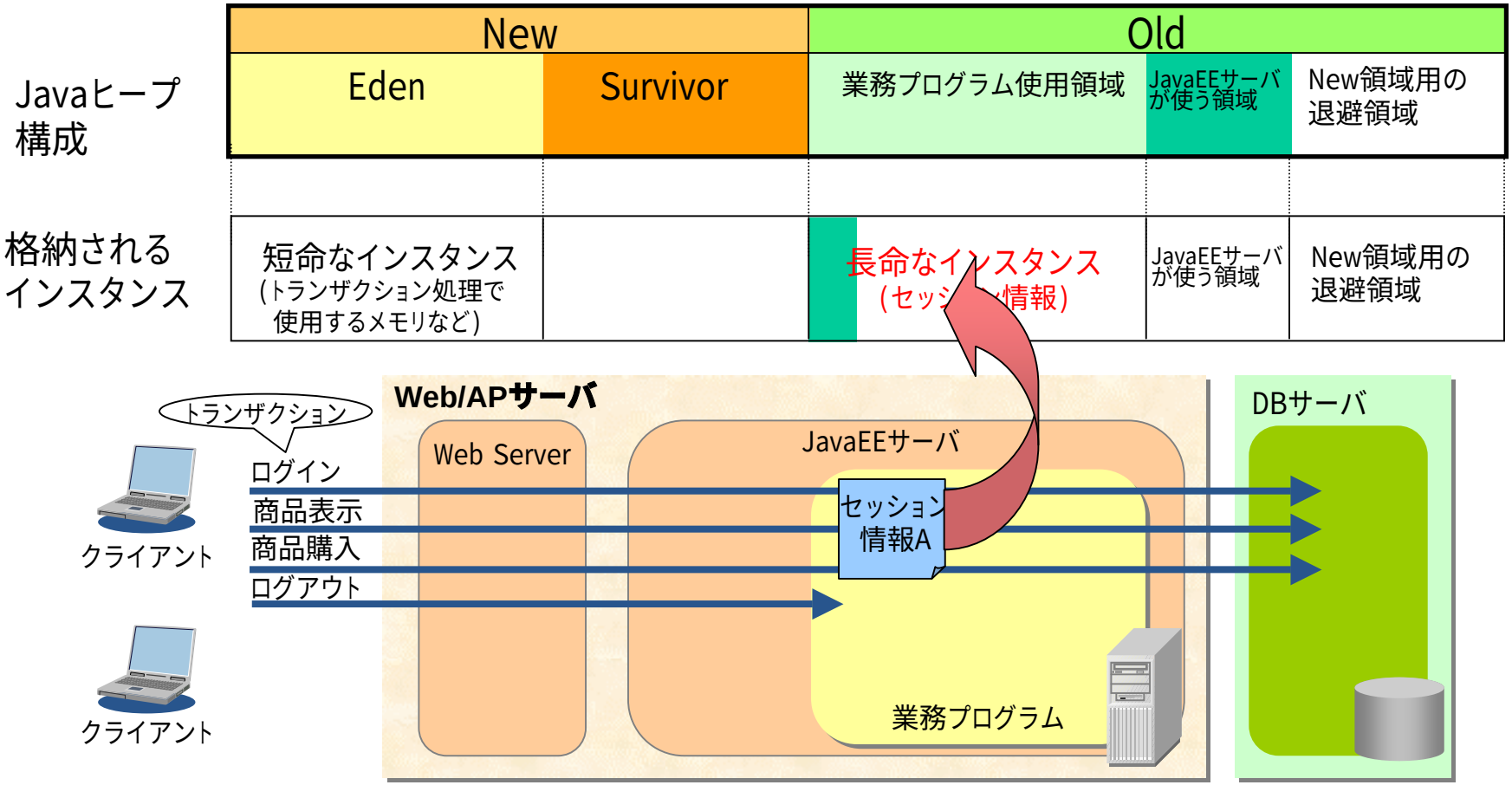
Contents

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

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

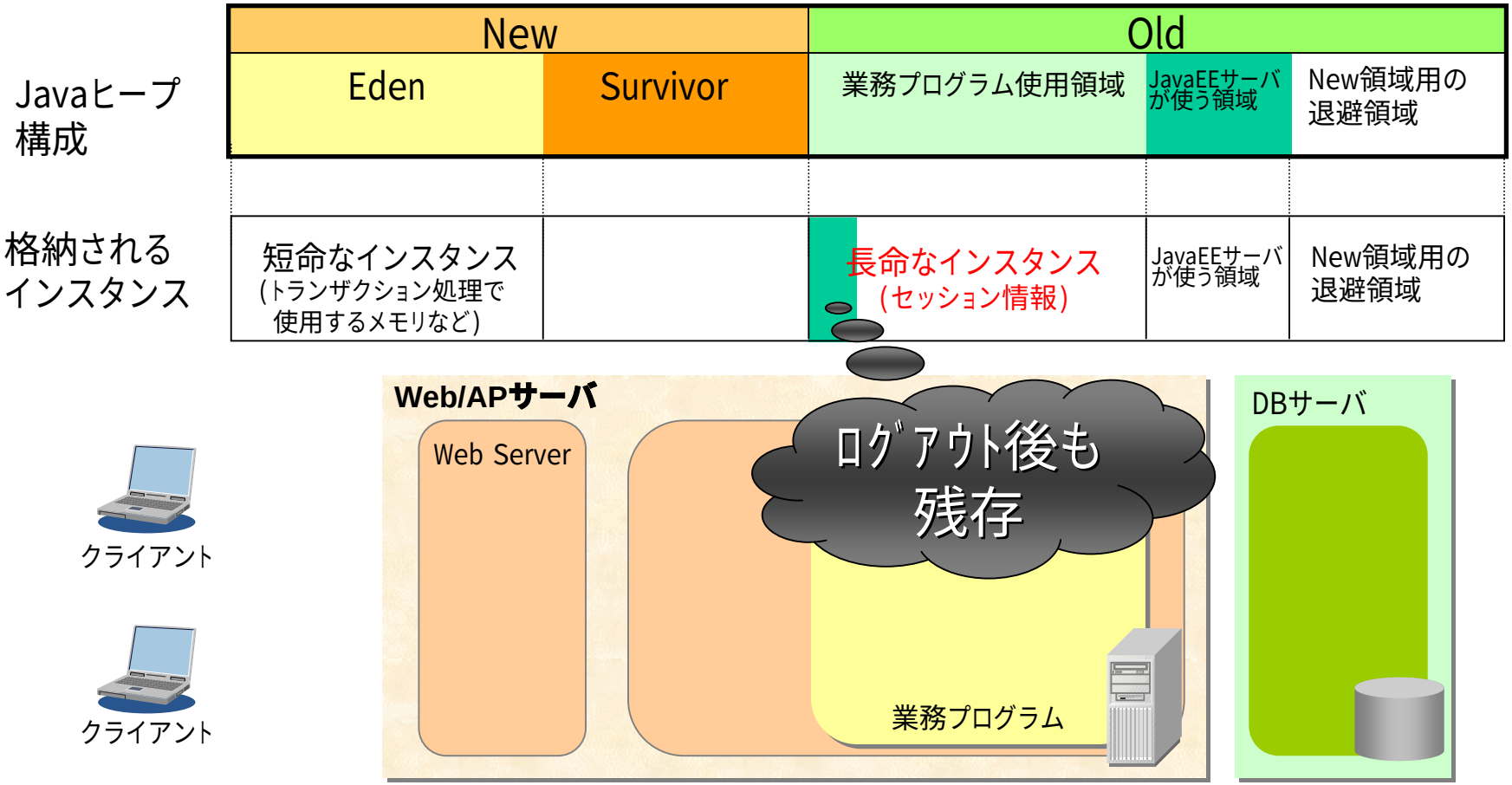
- New領域およびOld領域に格納されるインスタンスの違いに着目



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

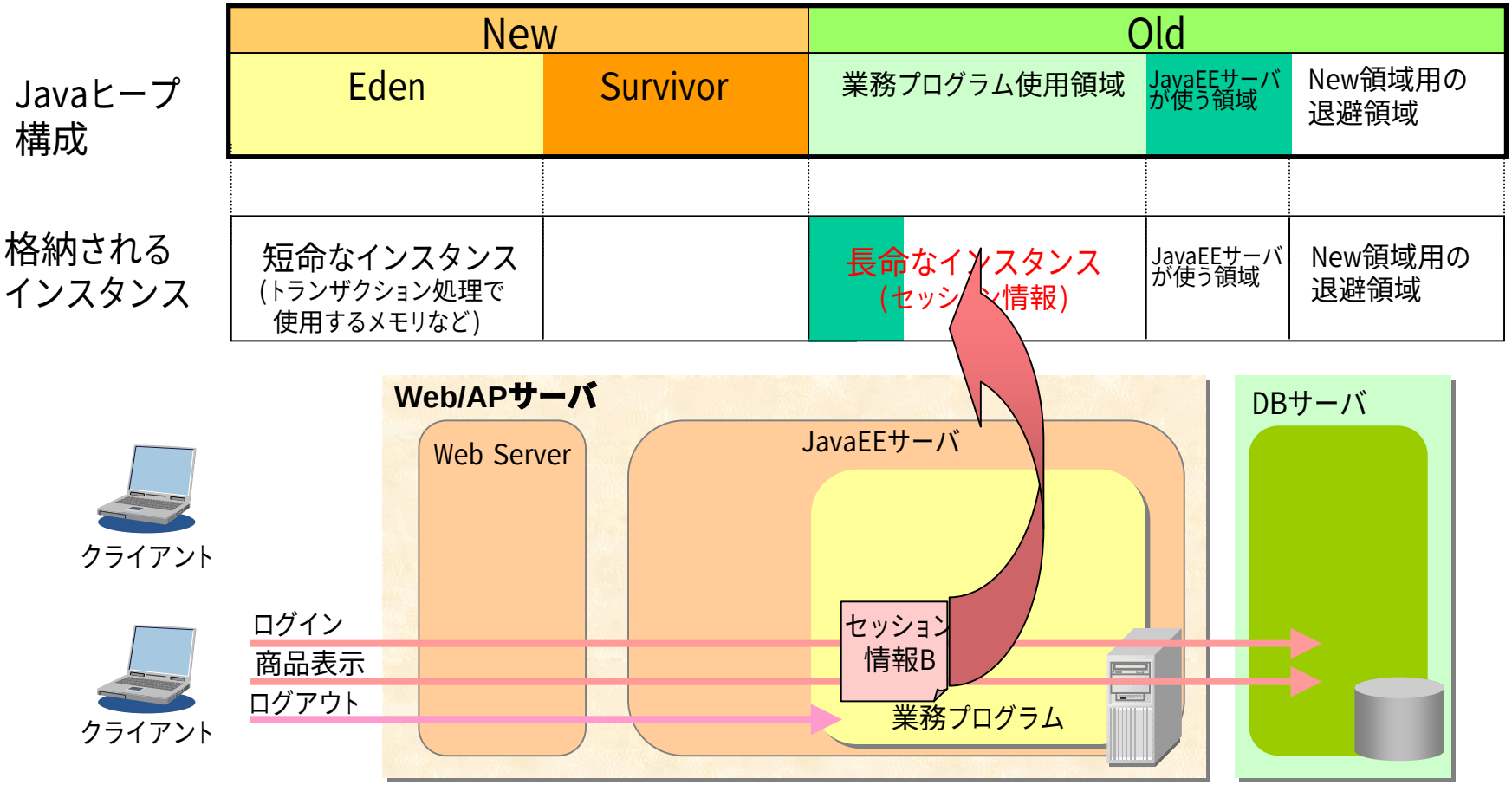
- New領域およびOld領域に格納されるインスタンスの違いに着目



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

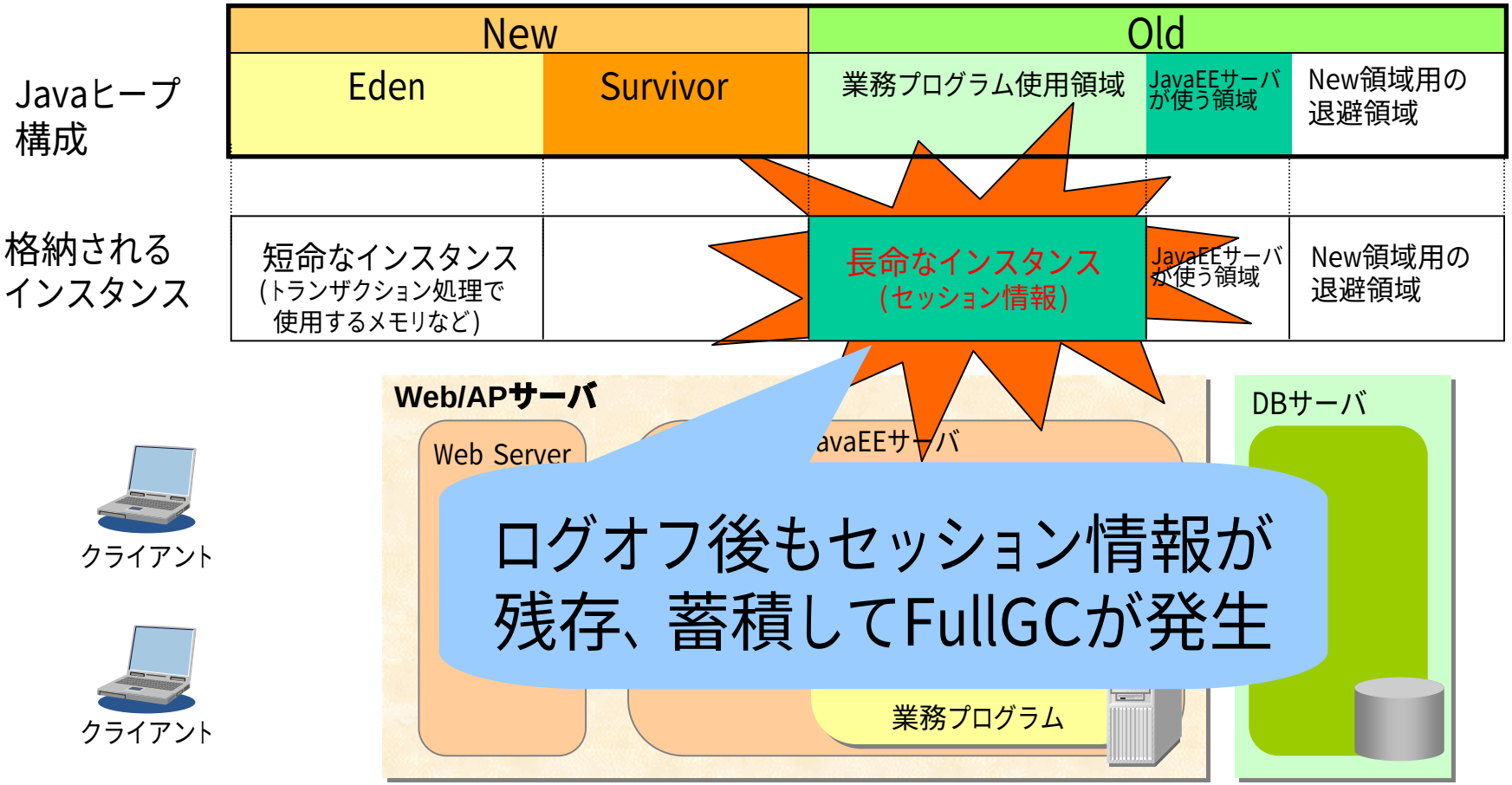
- New領域およびOld領域に格納されるインスタンスの違いに着目



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！-

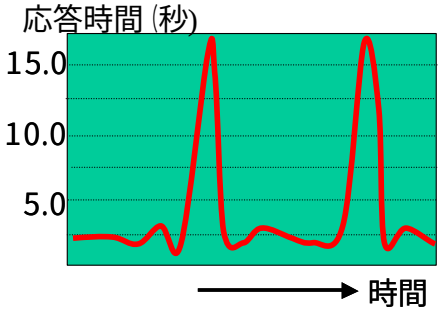
HITACHI
Inspire the Next

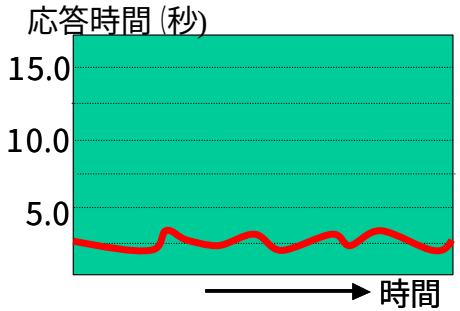
Full GCによるオンライン業務の一時的な停止は、
何とかならないものだろうか・・・

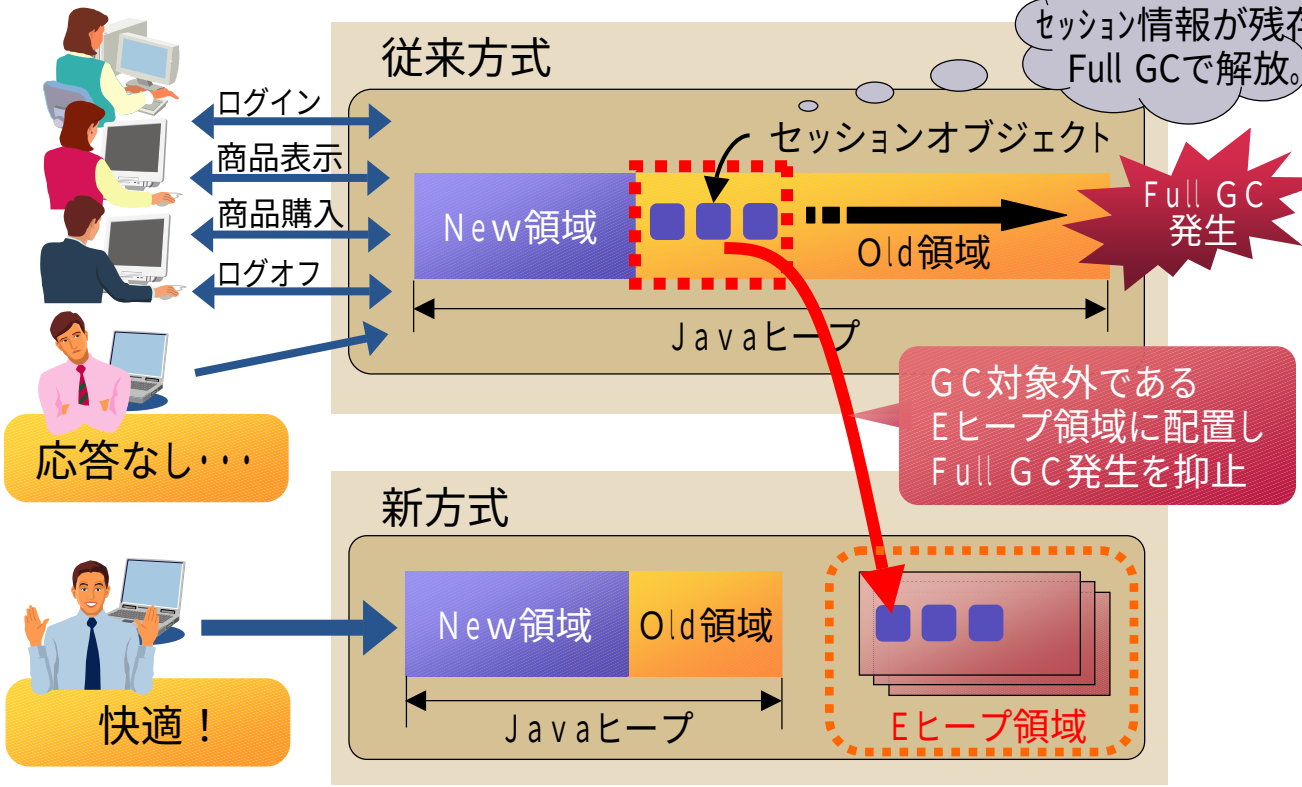
業界初！(*)
Full GCレスを実現

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。

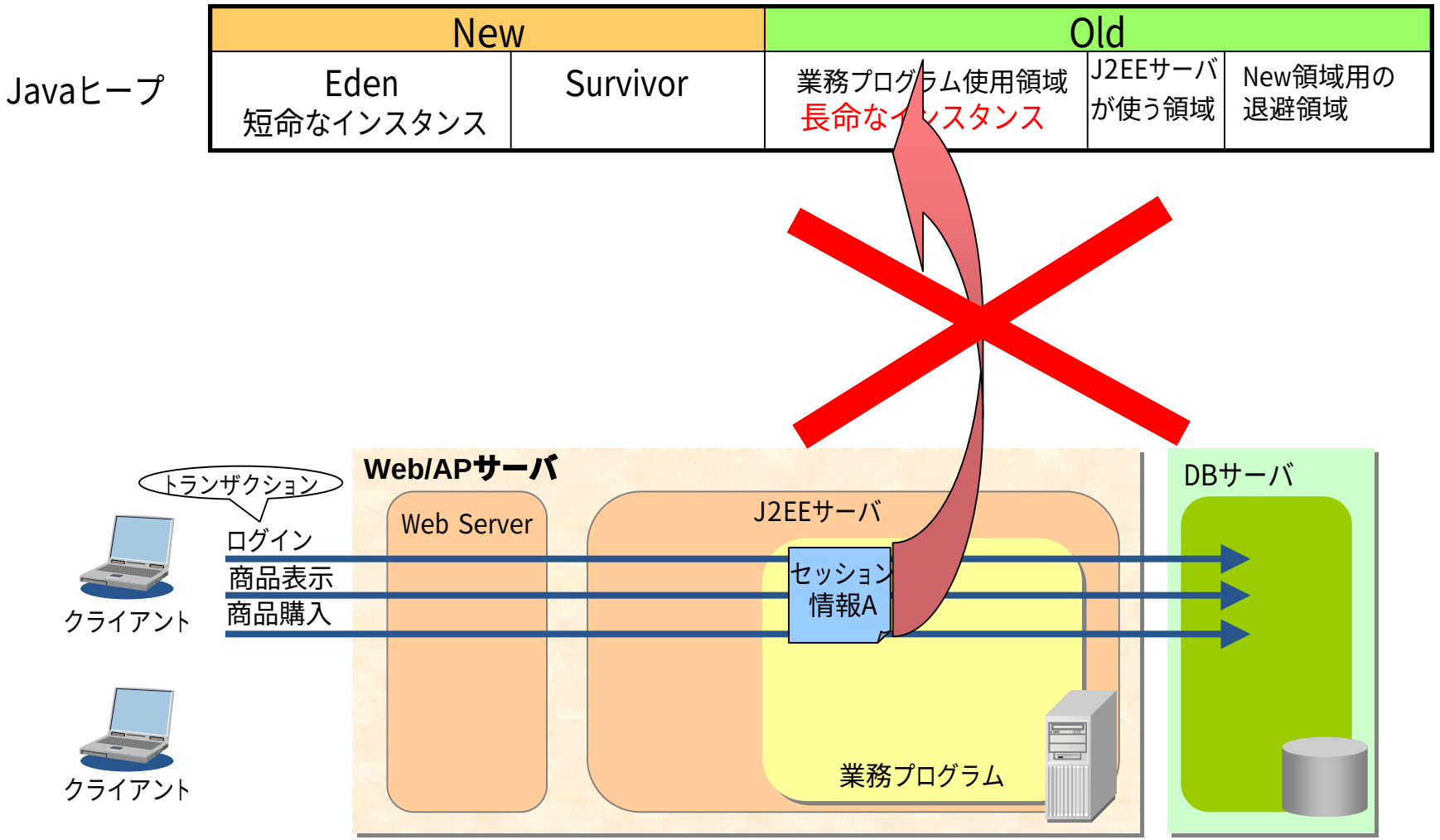




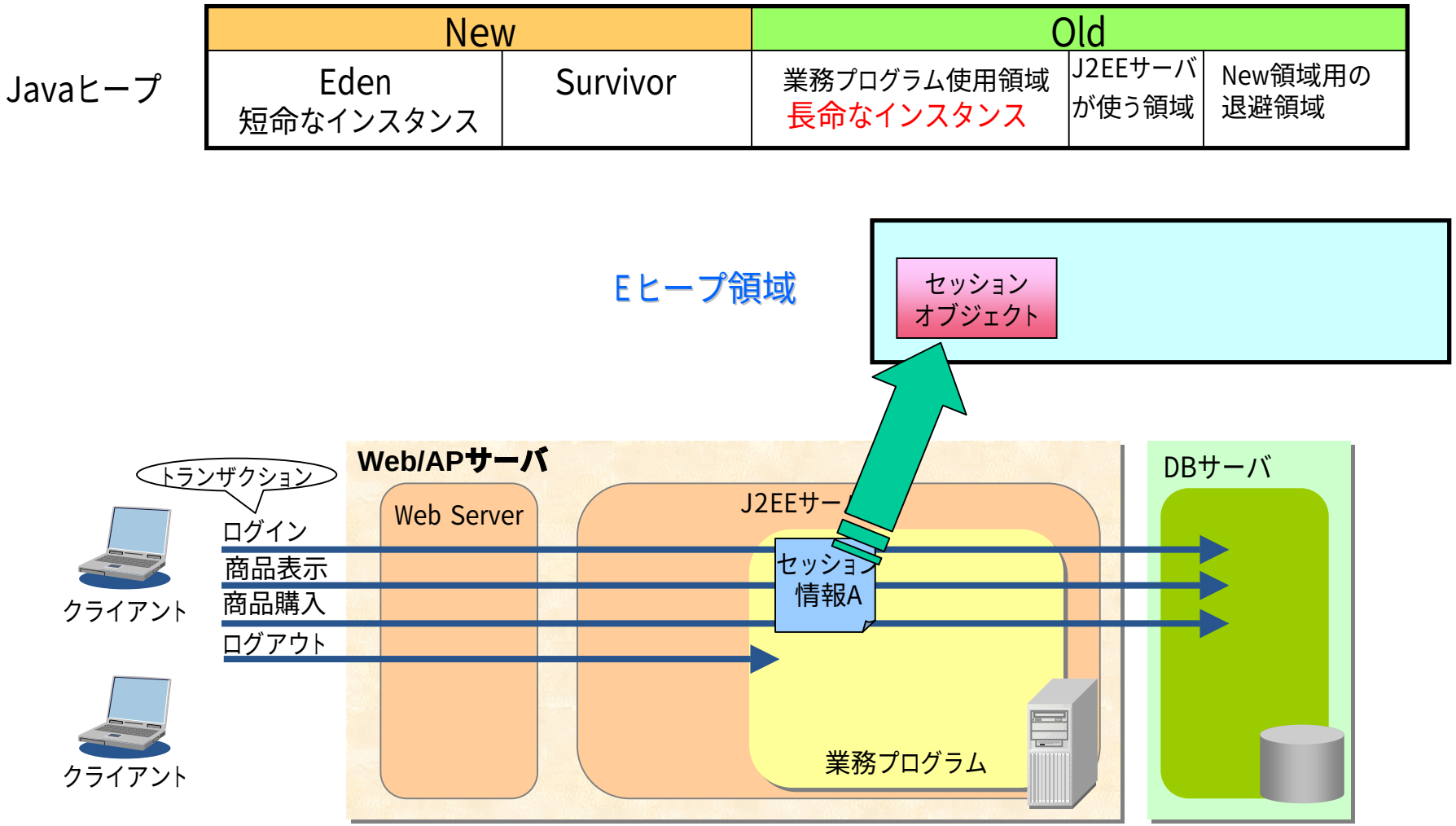


(*)・・・弊社調べ　2008年10月時点

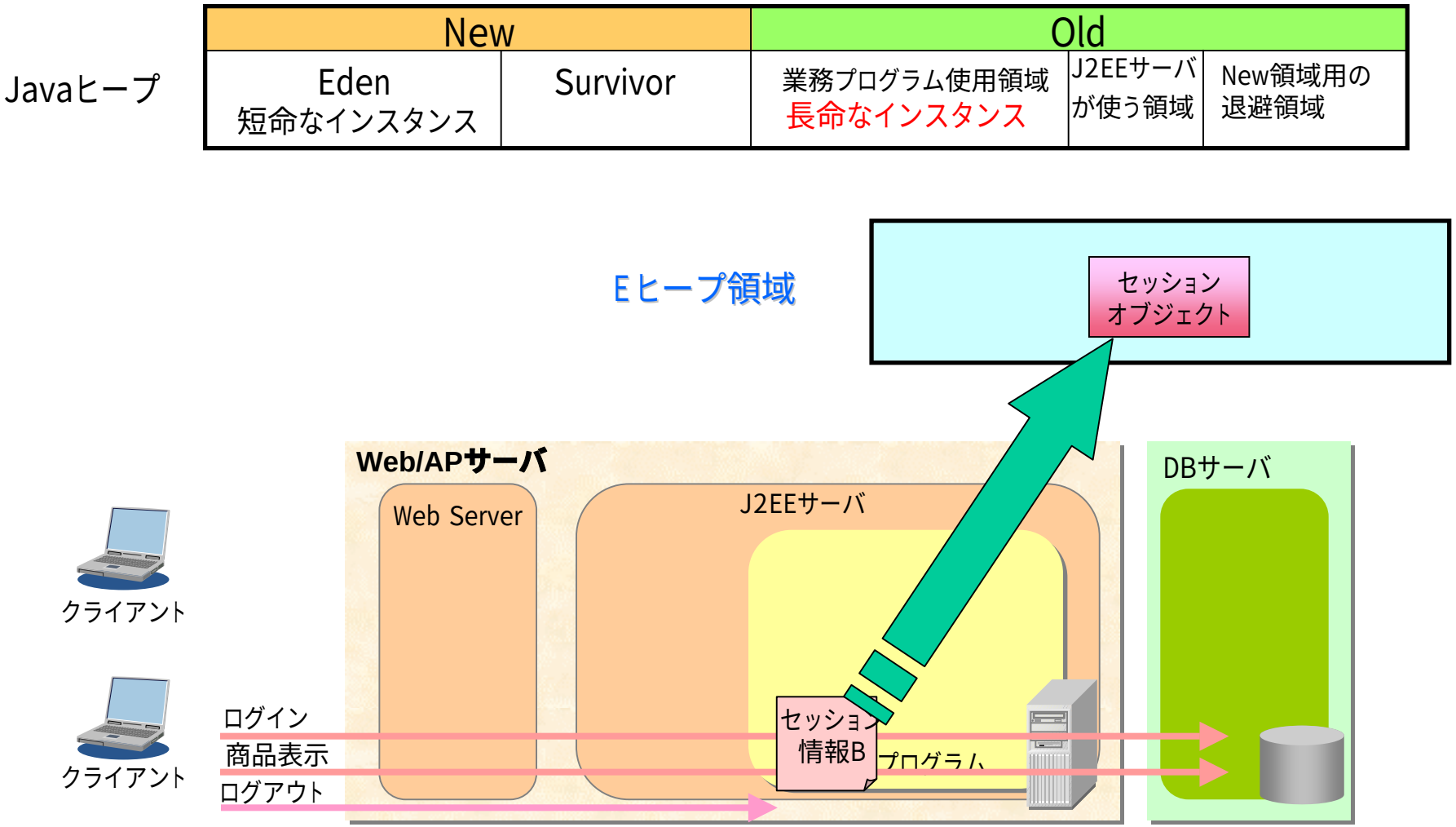
3.「FullGCレス」を実現する最新技術
-FullGCレス機能適用時の動き-



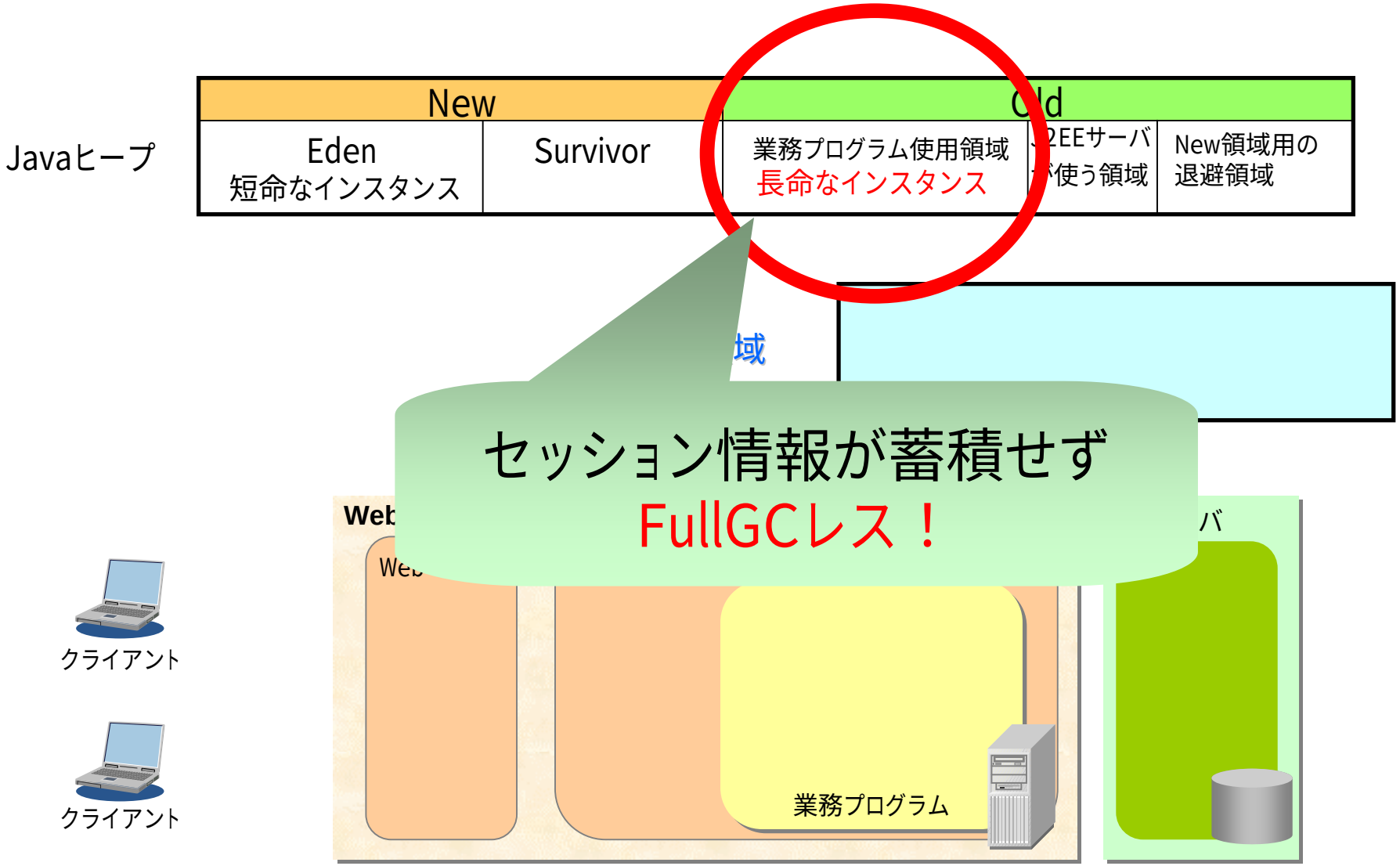
3.「FullGCレス」を実現する最新技術
-FullGCレス機能適用時の動き-



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

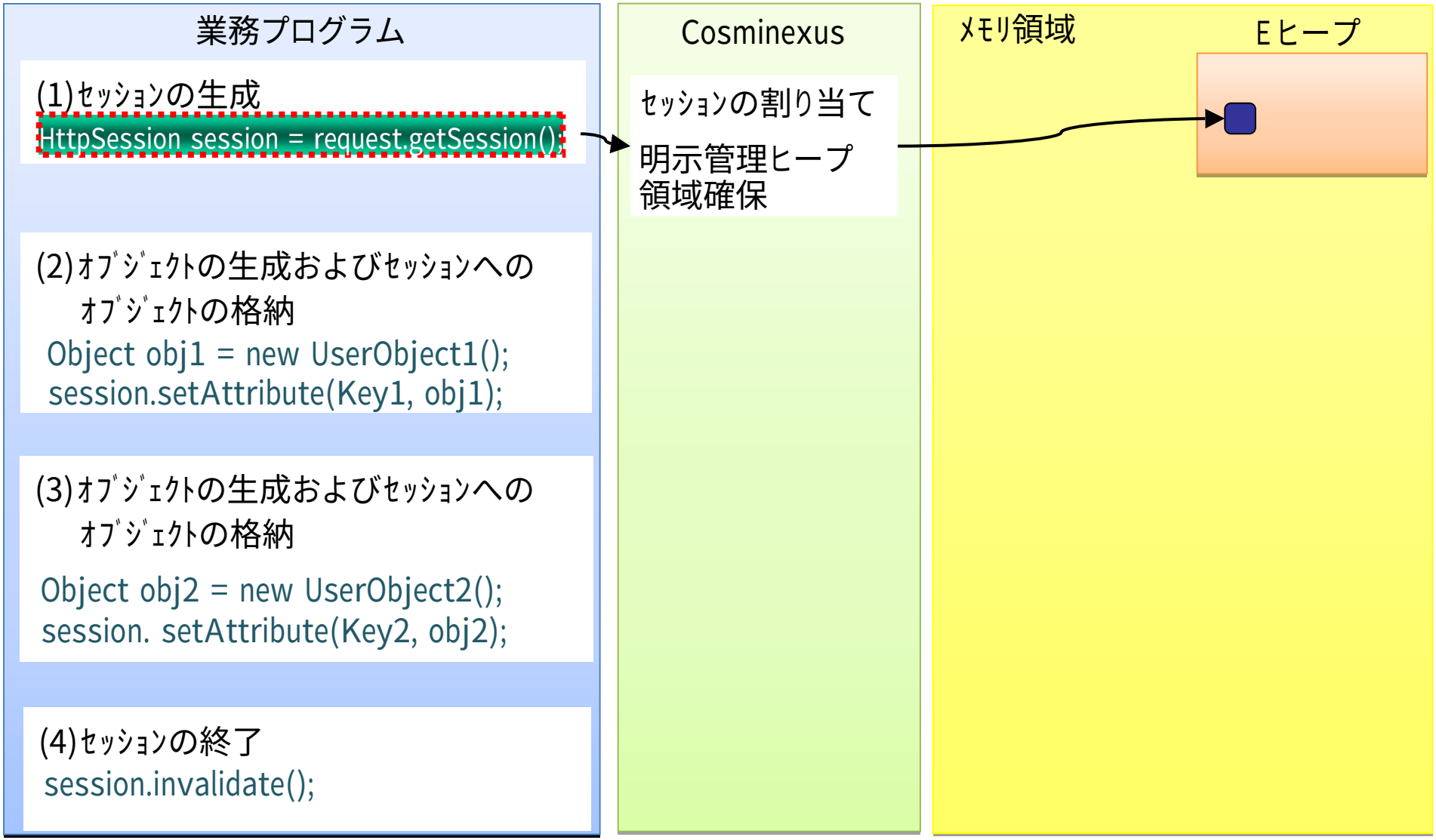


3.「FullGCレス」を実現する最新技術
-FullGCレス機能適用時の動き-



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

業務プログラム

(1)ｾｯｼｮﾝの生成

```
HttpSession session = request.getSession();
```

(2)ｵﾌﾞｼﾞｪｸﾄの生成およびｾｯｼｮﾝへの
ｵﾌﾞｼﾞｪｸﾄの格納

```
Object obj1 = new UserObject1();
session.setAttribute(Key1, obj1);
```

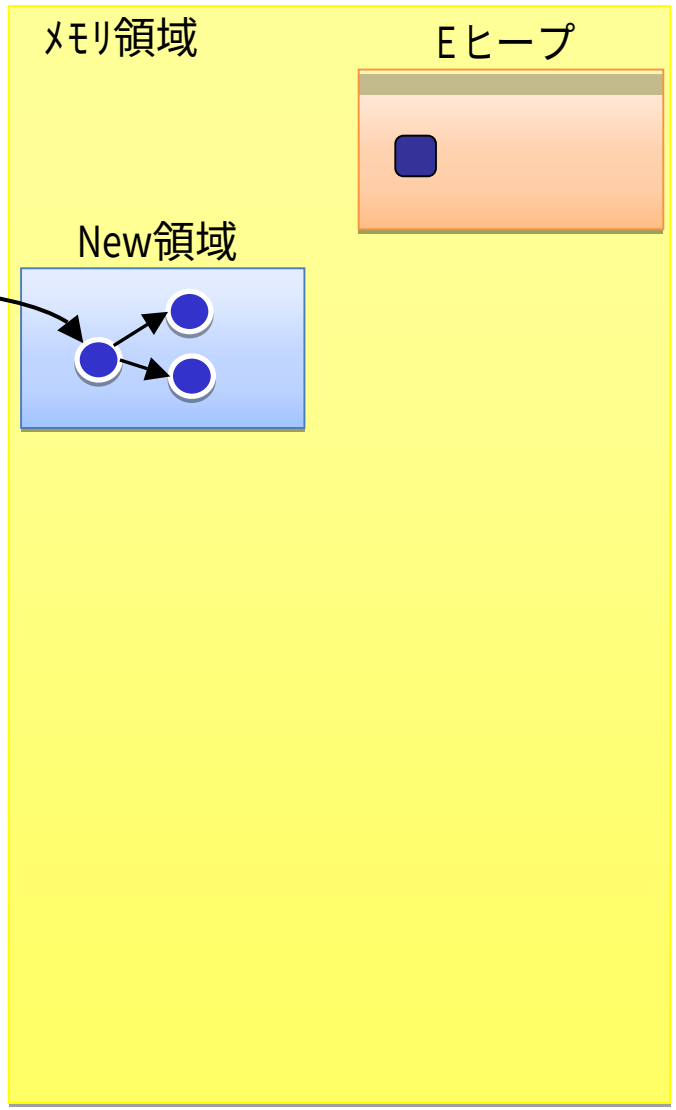
(3)ｵﾌﾞｼﾞｪｸﾄの生成およびｾｯｼｮﾝへの
ｵﾌﾞｼﾞｪｸﾄの格納

```
Object obj2 = new UserObject2();
session. setAttribute(Key2, obj2);
```

(4)ｾｯｼｮﾝの終了

```
session.invalidate();
```

Cosminexus



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

業務プログラム

(1)ｾｯｼｮﾝの生成
HttpSession session = request.getSession();

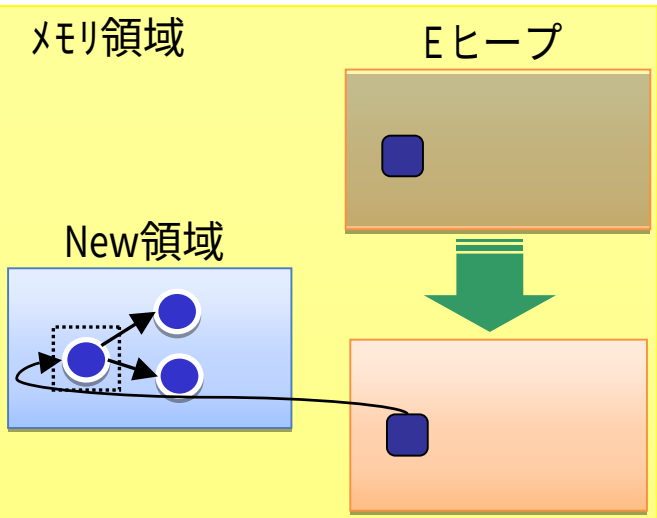
(2)ｵﾌﾞｼﾞｪｸﾄの生成およびｾｯｼｮﾝへの
ｵﾌﾞｼﾞｪｸﾄの格納
Object obj1 = new UserObject1();
session.setAttribute(Key1, obj1);

(3)ｵﾌﾞｼﾞｪｸﾄの生成およびｾｯｼｮﾝへの
ｵﾌﾞｼﾞｪｸﾄの格納
Object obj2 = new UserObject2();
session. setAttribute(Key2, obj2);

(4)ｾｯｼｮﾝの終了
session.invalidate();

Cosminexus

Obj1とセッション
を関連付け



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

業務プログラム

(1)ｾｯｼｮﾝの生成

```
HttpSession session = request.getSession();
```

(2)ｵﾌﾞｼﾞｪｸﾄの生成およびｾｯｼｮﾝへの
ｵﾌﾞｼﾞｪｸﾄの格納

```
Object obj1 = new UserObject1();
session.setAttribute(Key1, obj1);
```

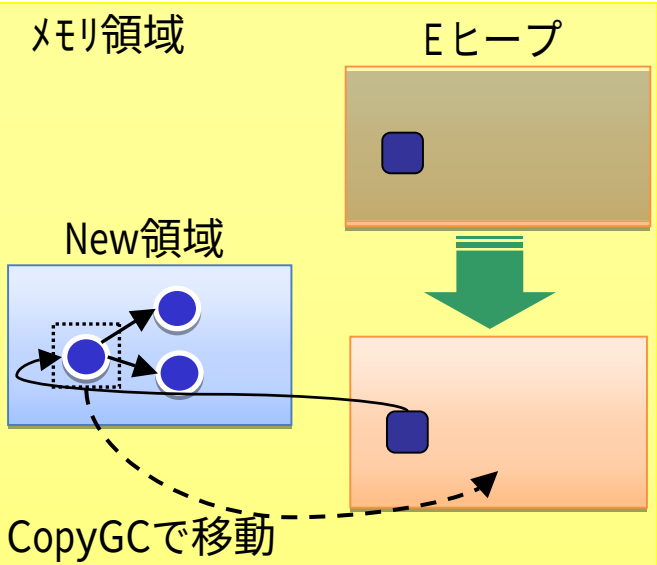
(3)ｵﾌﾞｼﾞｪｸﾄの生成およびｾｯｼｮﾝへの
ｵﾌﾞｼﾞｪｸﾄの格納

```
Object obj2 = new UserObject2();
session. setAttribute(Key2, obj2);
```

(4)ｾｯｼｮﾝの終了

```
session.invalidate();
```

Cosminexus



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

業務プログラム

(1)ｾｯｼｮﾝの生成

```
HttpSession session = request.getSession();
```

(2)ｵﾌﾞｼﾞｪｸﾄの生成およびｾｯｼｮﾝへの
ｵﾌﾞｼﾞｪｸﾄの格納

```
Object obj1 = new UserObject1();
session.setAttribute(Key1, obj1);
```

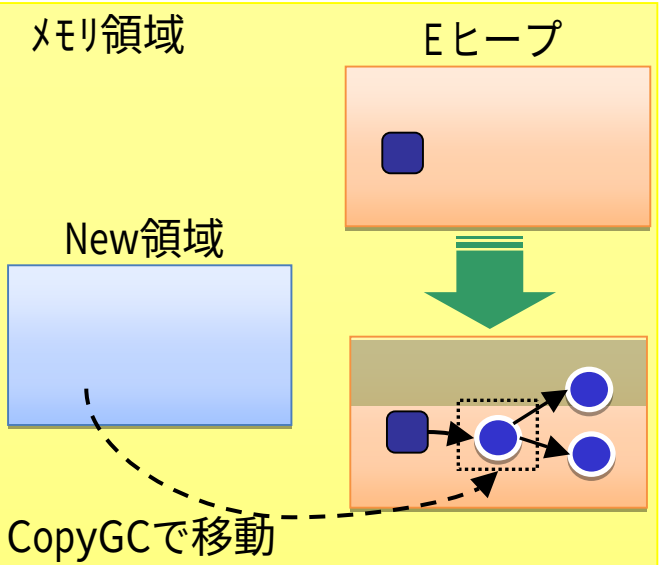
(3)ｵﾌﾞｼﾞｪｸﾄの生成およびｾｯｼｮﾝへの
ｵﾌﾞｼﾞｪｸﾄの格納

```
Object obj2 = new UserObject2();
session. setAttribute(Key2, obj2);
```

(4)ｾｯｼｮﾝの終了

```
session.invalidate();
```

Cosminexus



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

業務プログラム

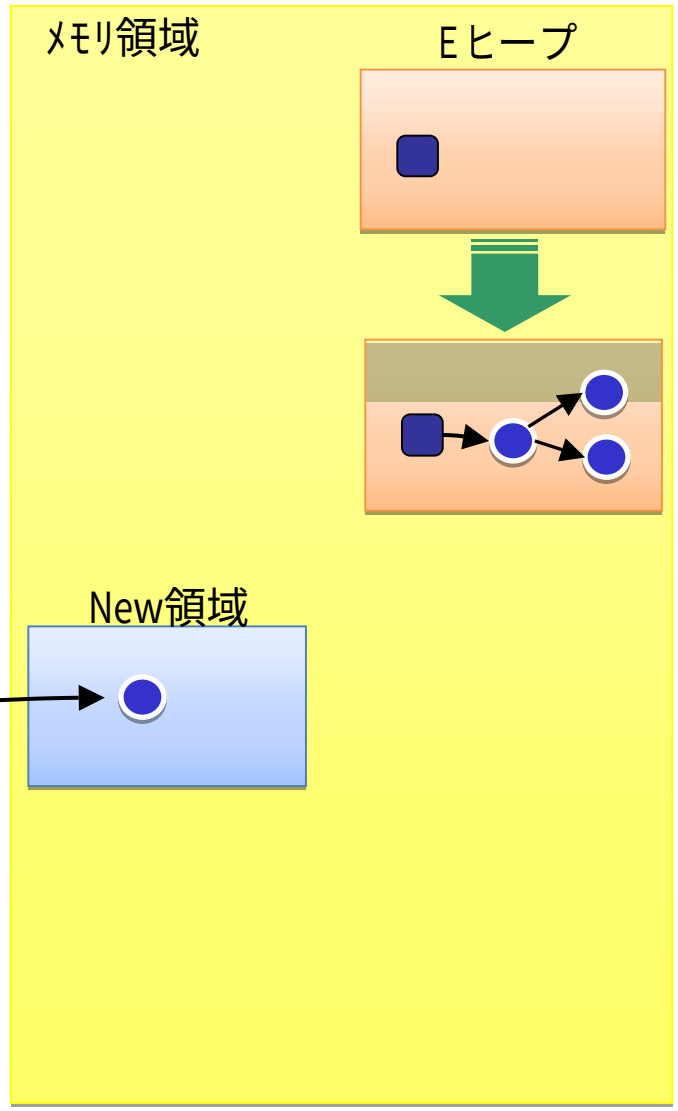
(1)ｾｯｼｮﾝの生成
```
HttpSession session = request.getSession();
```

(2)ｵﾌﾞｼﾞｪｸﾄの生成およびｾｯｼｮﾝへの
ｵﾌﾞｼﾞｪｸﾄの格納
```
Object obj1 = new UserObject1();
session.setAttribute(Key1, obj1);
```

(3)ｵﾌﾞｼﾞｪｸﾄの生成およびｾｯｼｮﾝへの
ｵﾌﾞｼﾞｪｸﾄの格納
```
Object obj2 = new UserObject2();
session. setAttribute(Key2, obj2);
```

(4)ｾｯｼｮﾝの終了
```
session.invalidate();
```

Cosminexus



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

業務プログラム

(1)ｾｯｼｮﾝの生成
```
HttpSession session = request.getSession();
```

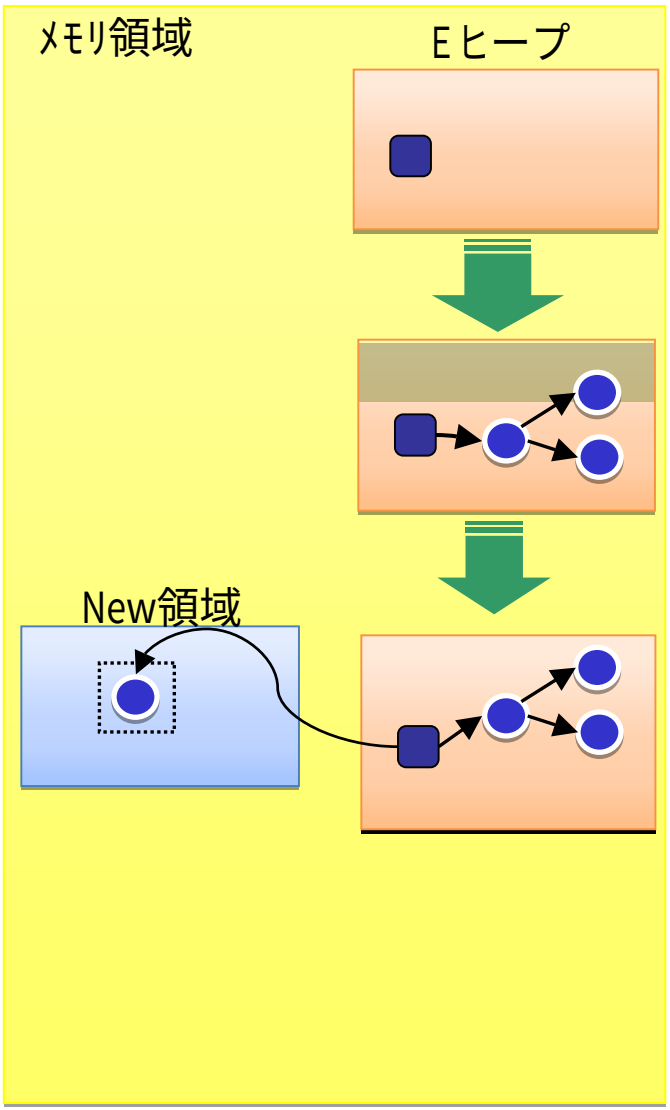
(2)ｵﾌﾞｼﾞｪｸﾄの生成およびｾｯｼｮﾝへの
ｵﾌﾞｼﾞｪｸﾄの格納
```
Object obj1 = new UserObject1();
session.setAttribute(Key1, obj1);
```

(3)ｵﾌﾞｼﾞｪｸﾄの生成およびｾｯｼｮﾝへの
ｵﾌﾞｼﾞｪｸﾄの格納
```
Object obj2 = new UserObject2();
session.setAttribute(Key2, obj2);
```

(4)ｾｯｼｮﾝの終了
```
session.invalidate();
```

Cosminexus

Obj2とセッション
を関連付け



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

業務プログラム

(1)ｾｯｼｮﾝの生成

```
HttpSession session = request.getSession();
```

(2)ｵﾌﾞｼﾞｪｸﾄの生成およびｾｯｼｮﾝへの
ｵﾌﾞｼﾞｪｸﾄの格納

```
Object obj1 = new UserObject1();
session.setAttribute(Key1, obj1);
```

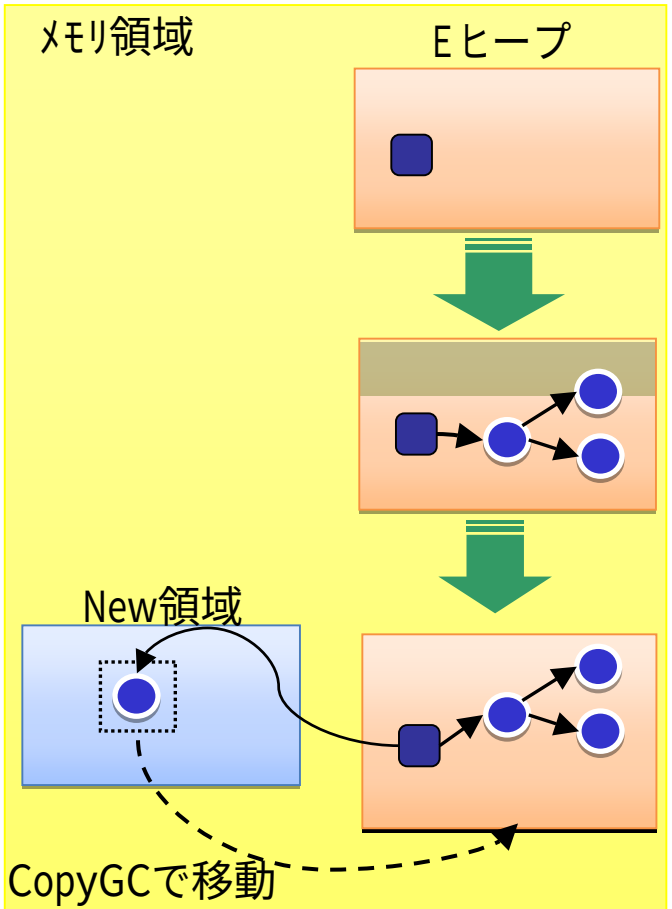
(3)ｵﾌﾞｼﾞｪｸﾄの生成およびｾｯｼｮﾝへの
ｵﾌﾞｼﾞｪｸﾄの格納

```
Object obj2 = new UserObject2();
session. setAttribute(Key2, obj2);
```

(4)ｾｯｼｮﾝの終了

```
session.invalidate();
```

Cosminexus



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

業務プログラム

(1)ｾｯｼｮﾝの生成

```
HttpSession session = request.getSession();
```

(2)ｵﾌﾞｼﾞｪｸﾄの生成およびｾｯｼｮﾝへの
ｵﾌﾞｼﾞｪｸﾄの格納

```
Object obj1 = new UserObject1();
session.setAttribute(Key1, obj1);
```

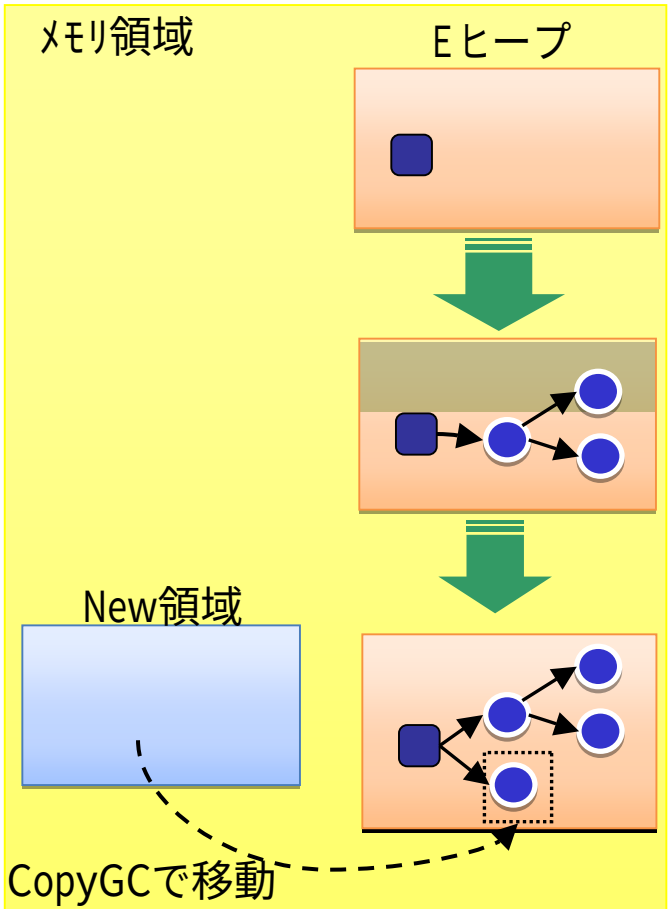
(3)ｵﾌﾞｼﾞｪｸﾄの生成およびｾｯｼｮﾝへの
ｵﾌﾞｼﾞｪｸﾄの格納

```
Object obj2 = new UserObject2();
session. setAttribute(Key2, obj2);
```

(4)ｾｯｼｮﾝの終了

```
session.invalidate();
```

Cosminexus



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

業務プログラム

(1)ｾｯｼｮﾝの生成
HttpSession session = request.getSession();

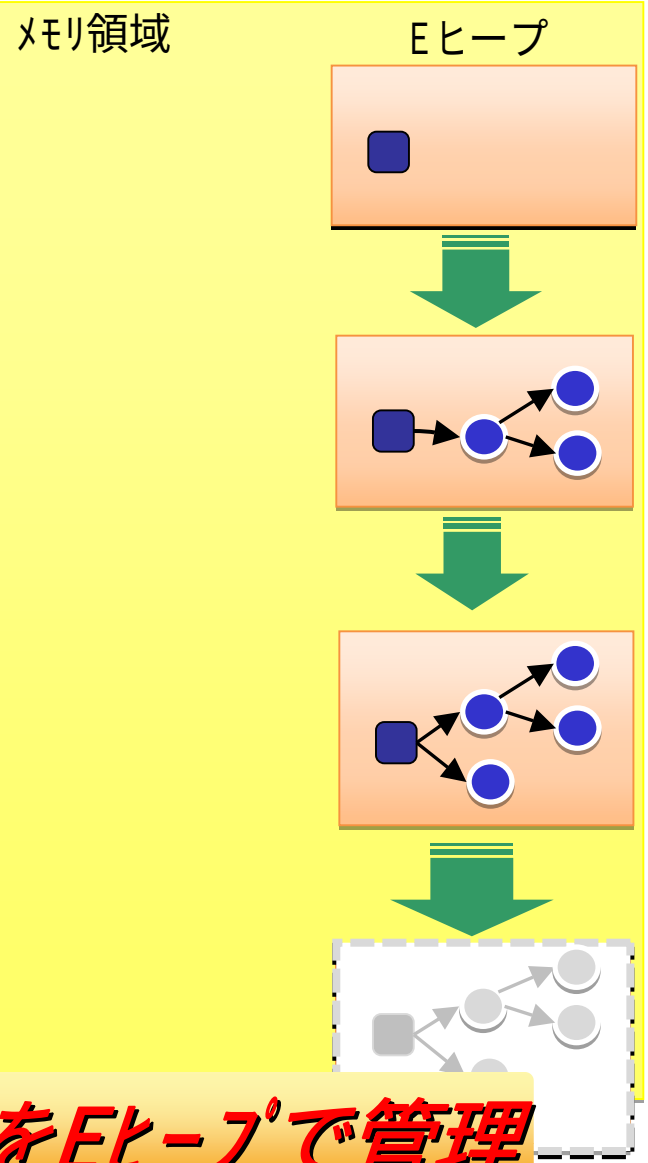
(2)ｵﾌﾞｼﾞｪｸﾄの生成およびｾｯｼｮﾝへの
ｵﾌﾞｼﾞｪｸﾄの格納
Object obj1 = new UserObject1();
session.setAttribute(Key1, obj1);

(3)ｵﾌﾞｼﾞｪｸﾄの生成およびｾｯｼｮﾝへの
ｵﾌﾞｼﾞｪｸﾄの格納
Object obj2 = new UserObject2();
session. setAttribute(Key2, obj2);

(4)ｾｯｼｮﾝの終了
session.invalidate();

Cosminexus

明示管理ヒープ
領域削除

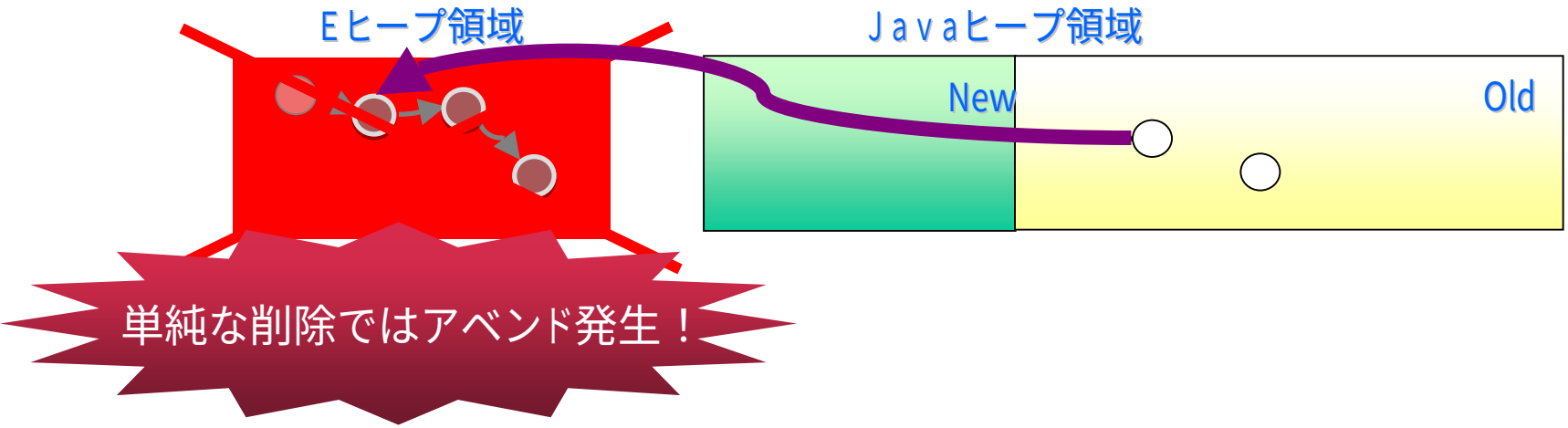


アプリ変更無しにセッションオブジェクトをEヒープで管理

3.「FullGCレス」を実現する最新技術
-「Eヒープ領域削除」の堅牢化-

特許出願済

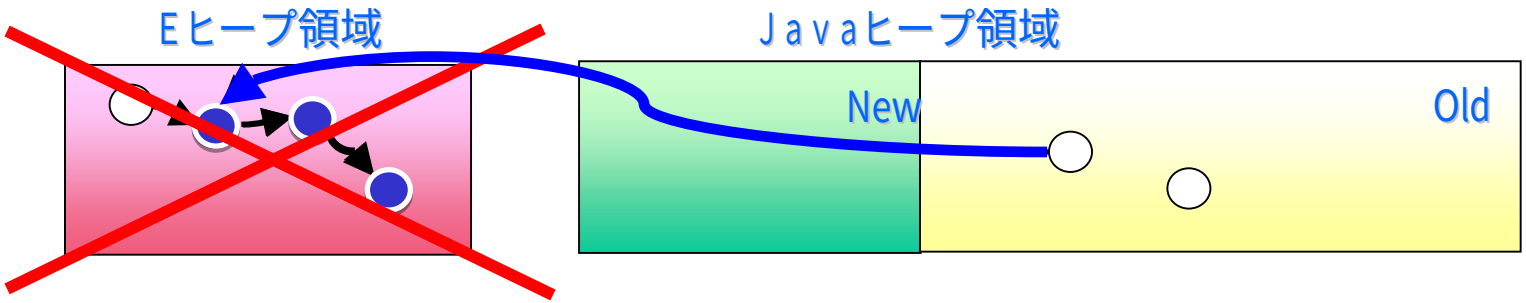
- 削除するEヒープ内に共通データなど使用中データが存在する場合
  →単純な削除ではトラブル(領域外参照)が発生



3.「FullGCレス」を実現する最新技術
-「Eヒープ領域削除」の堅牢化-

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

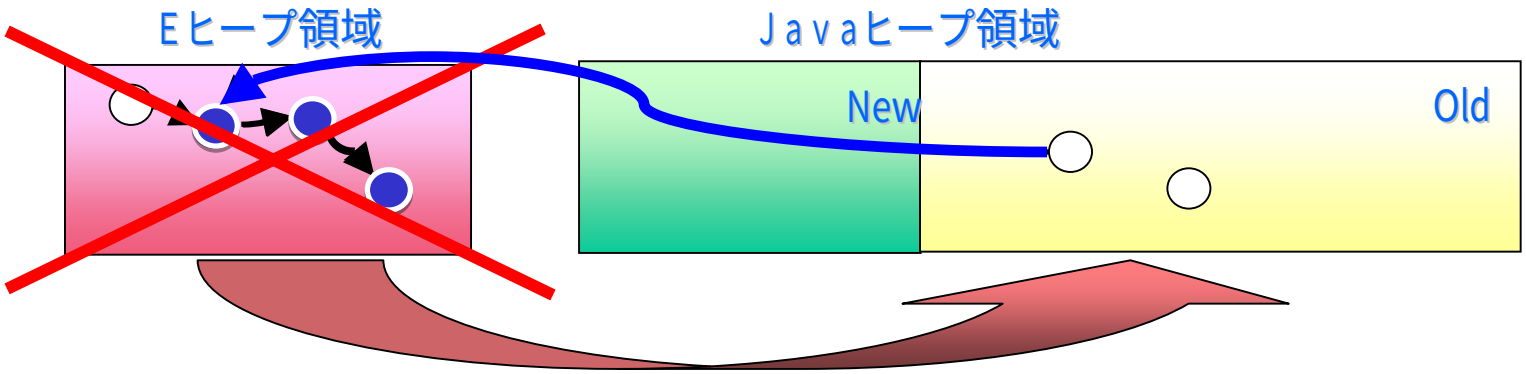


領域削除時に使用中
オブジェクトを検出して移動

3.「FullGCレス」を実現する最新技術
-「Eヒープ領域削除」の堅牢化-

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

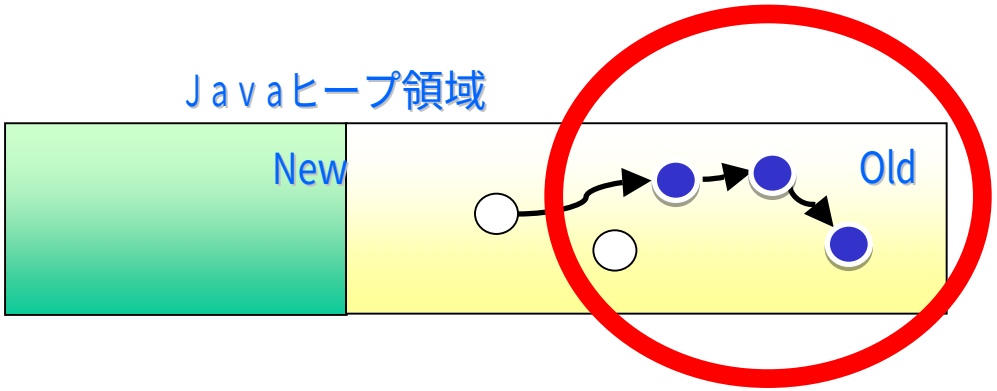


領域削除時に使用中
オブジェクトを検出して移動

3.「FullGCレス」を実現する最新技術
-「Eヒープ領域削除」の堅牢化-

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



領域削除時に使用中
オブジェクトを検出して移動

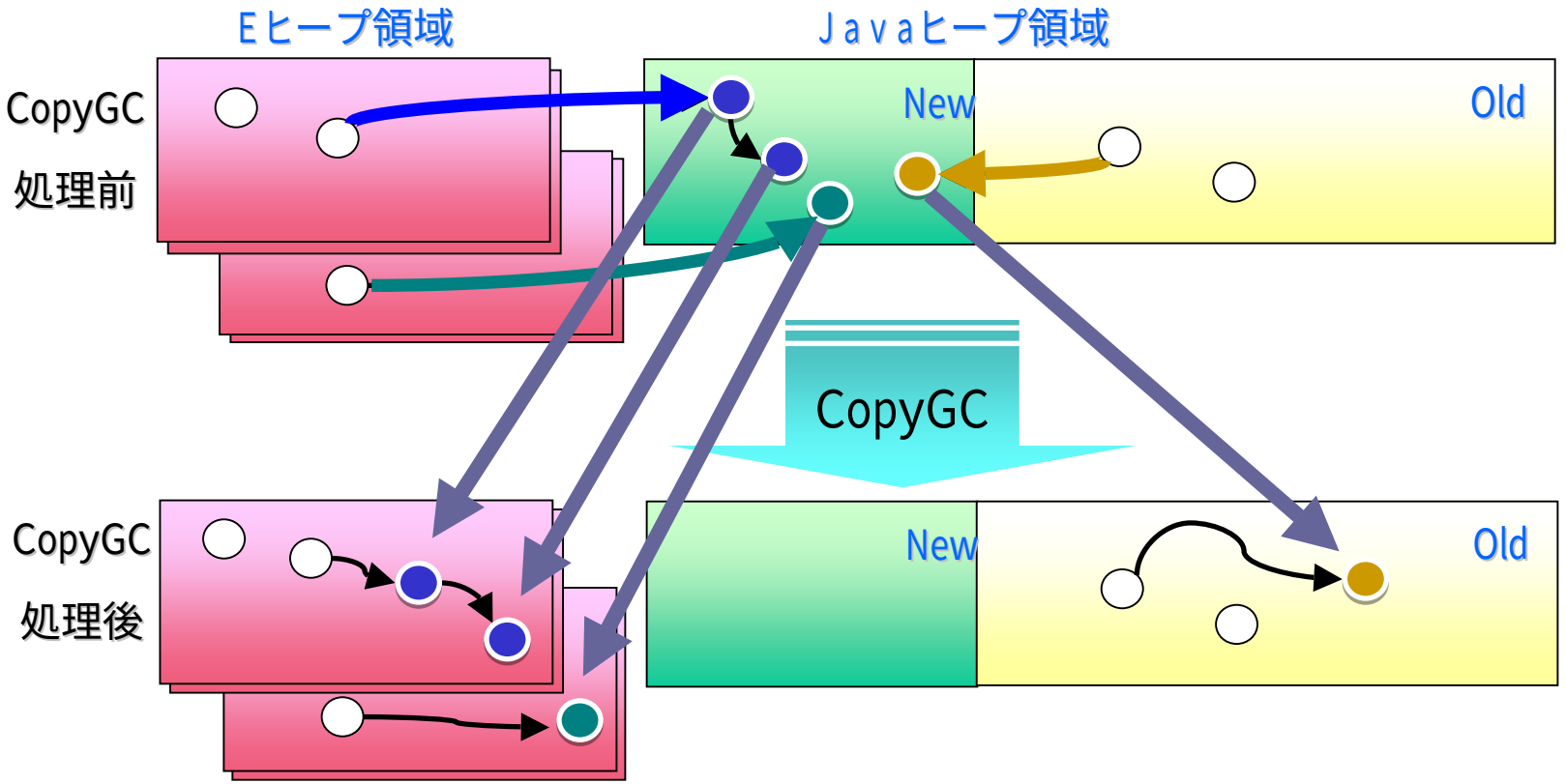
Eヒープ内に使用中データが存在する場合にはJavaヒープへの移動を行い、そのEヒープを削除してしまっても問題なく動作

→ ***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。

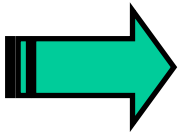
[CopyGC時間増によるスループット劣化の試算例]

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)

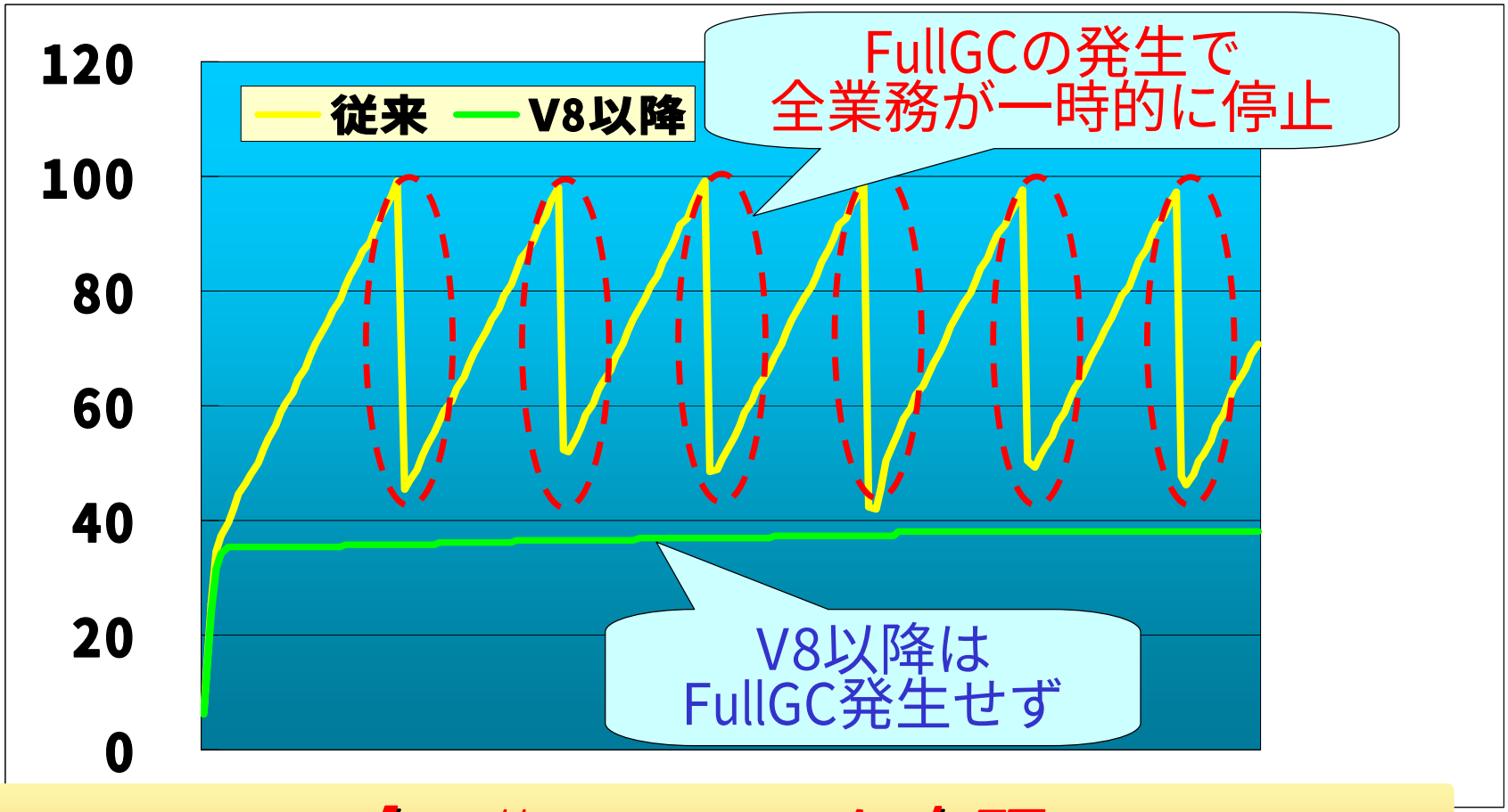
→処理性能への影響

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**



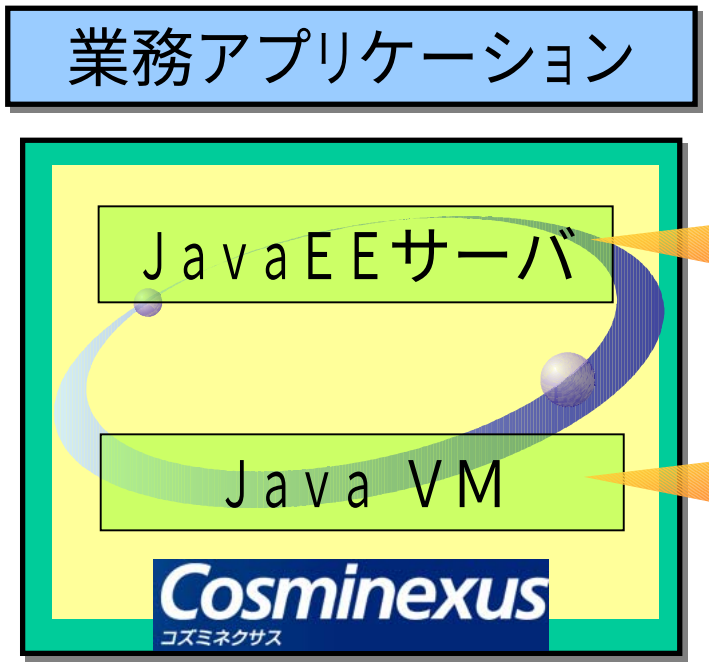
「FullGCレス」を実現！

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-

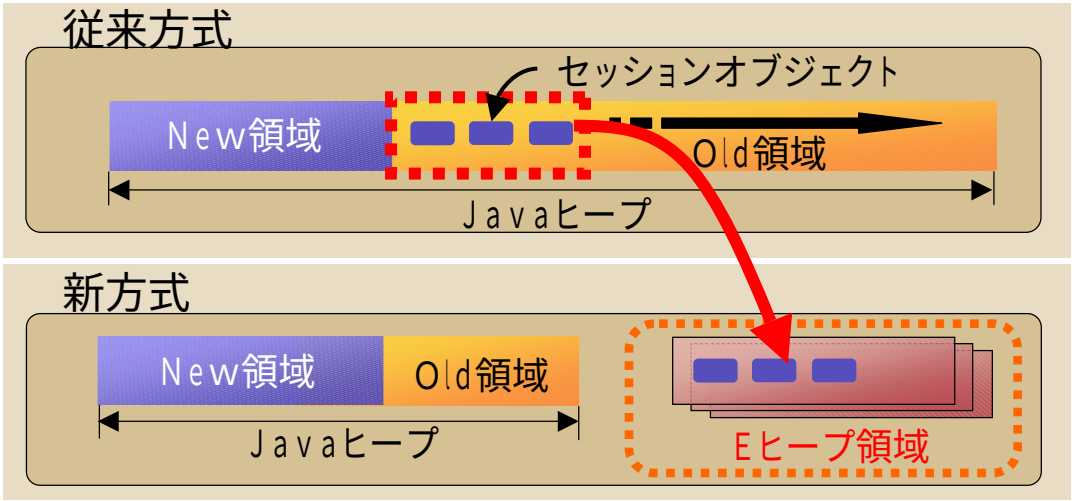
HITACHI
Inspire the Next

JavaEEｻｰﾊﾞと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！

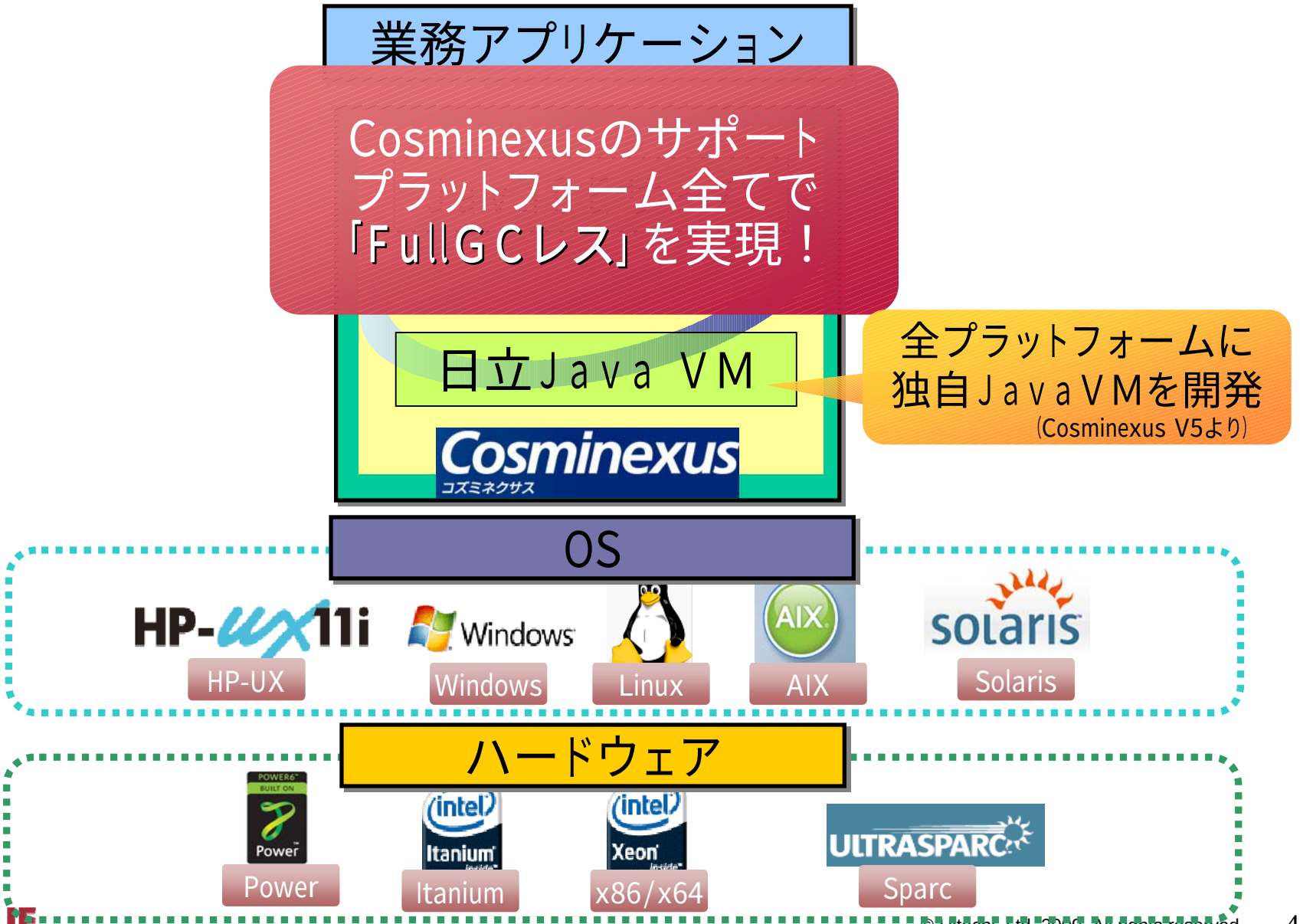


セッションオブジェクトを自動Eヒープ化

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-



Developers Summit

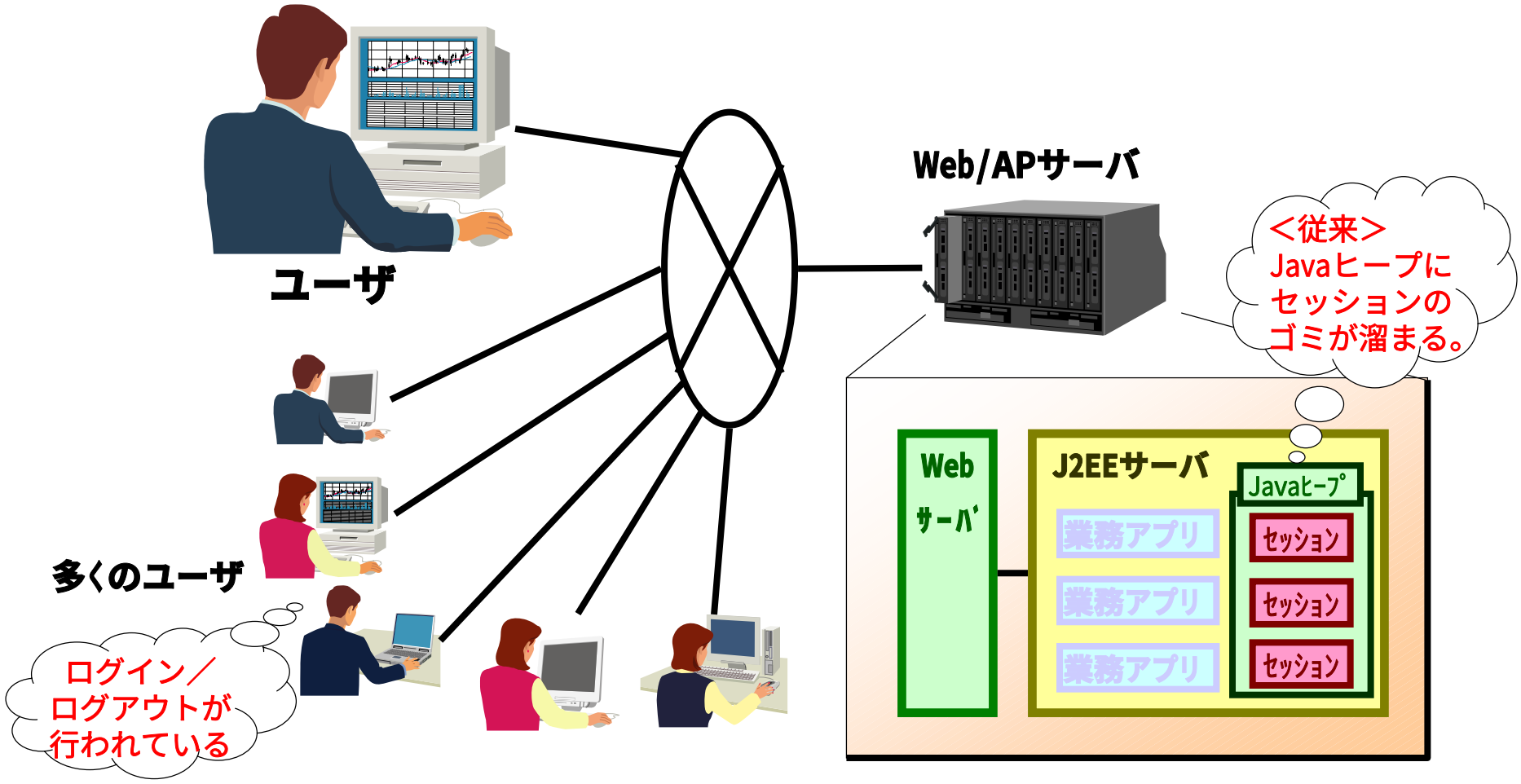
Contents

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

4.「FullGCレス」の効果（ﾃﾞﾓﾝｽﾄﾚｰｼｮﾝ）
-デモンストレーションの構成-

HITACHI
Inspire the Next

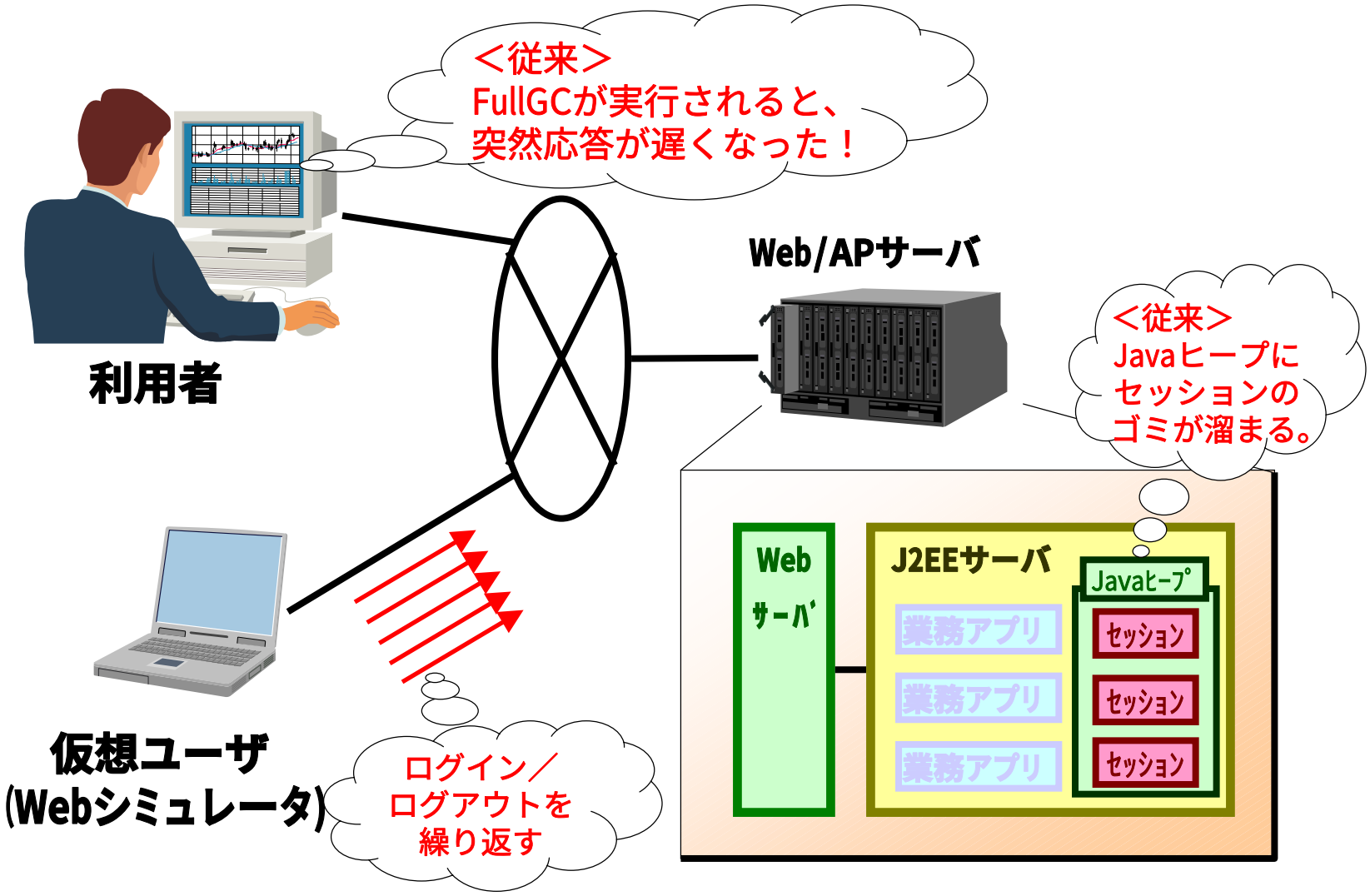
現実のシステムでは、多くのユーザがサーバにログインします。



4.「FullGCレス」の効果（ﾃﾞﾓﾝｽﾄﾚｰｼｮﾝ）
-デモンストレーションの構成-

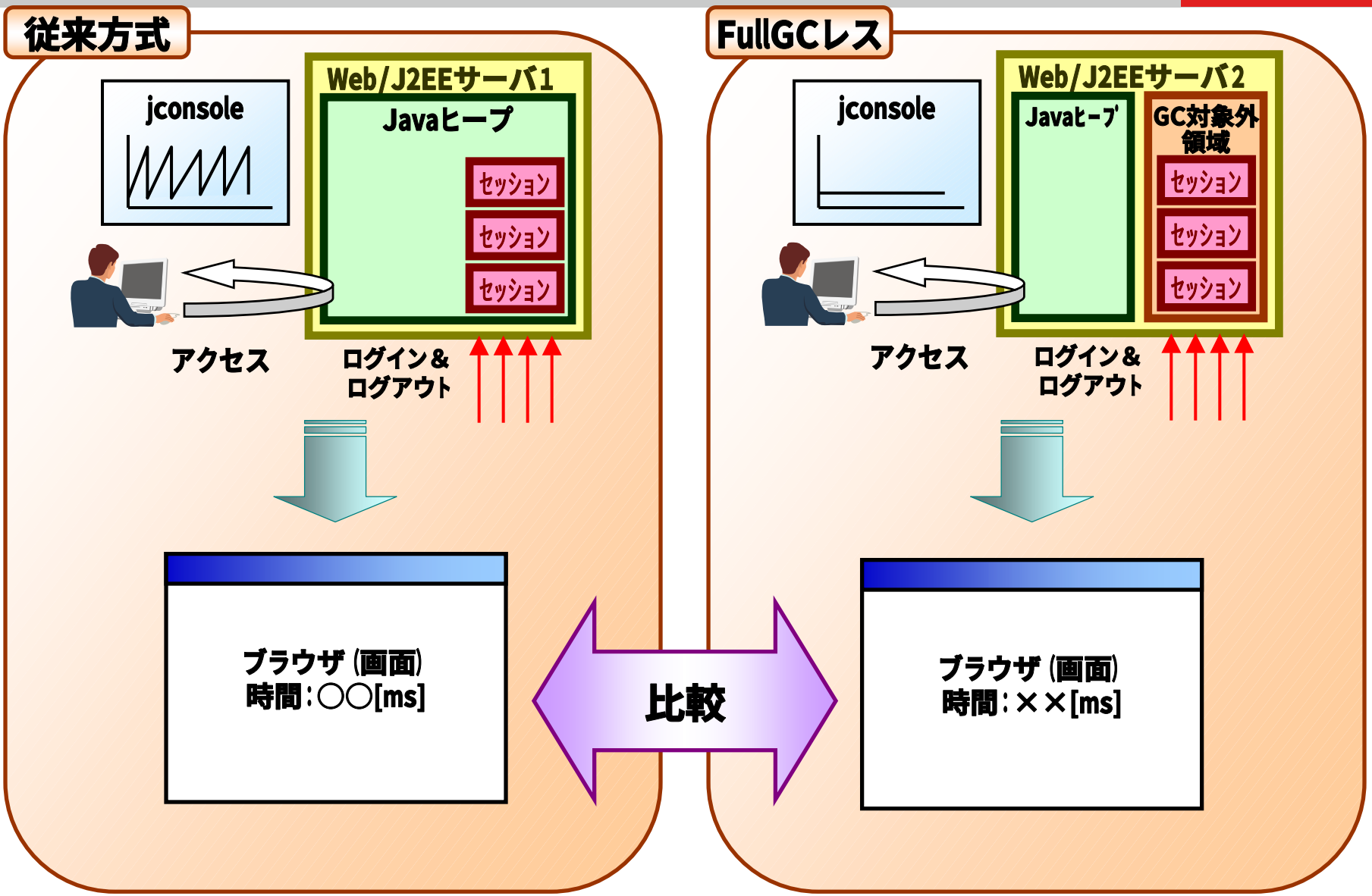
HITACHI
Inspire the Next

ここでは、シミュレータを使って多くの仮想ユーザでログインします。



4.「FullGCレス」の効果（ﾃﾞﾓﾝｽﾄﾚｰｼｮﾝ）

-デモンストレーションの内容-



4.「FullGCレス」の効果（ﾃﾞﾓﾝｽﾄﾚｰｼｮﾝ）

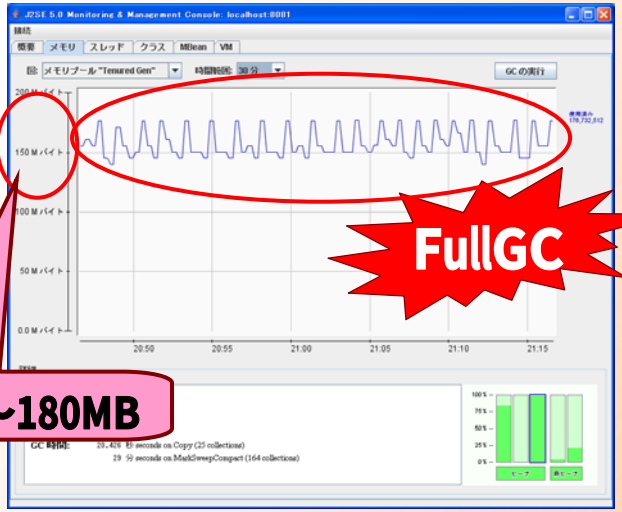
HITACHI
Inspire the Next

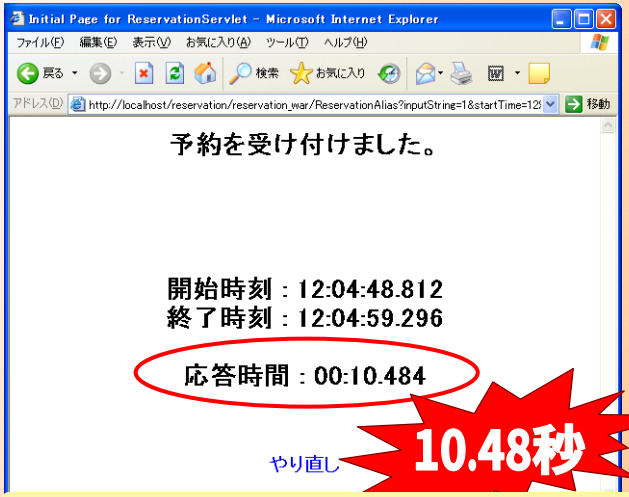
スクリーンをご覧ください。

4.「FullGCレス」の効果（ﾃﾞﾓﾝｽﾄﾚｰｼｮﾝ）
-デモンストレーションの結果まとめ-

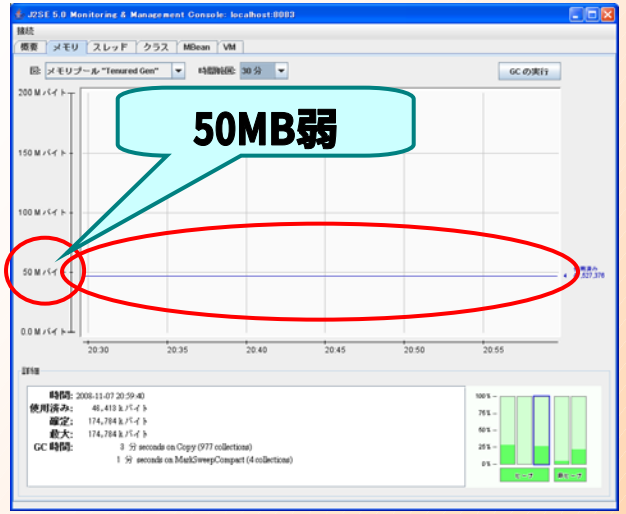
HITACHI
Inspire the Next

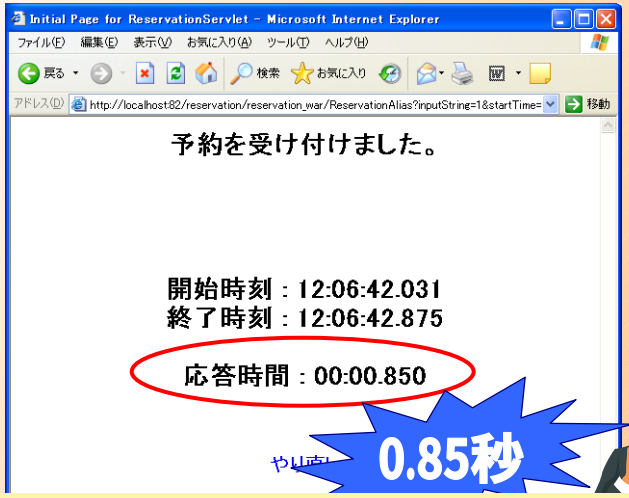
従来方式





FullGCレス





「FullGCレス」によりレスポンス遅延解消！

uVALUE

Developers Summit

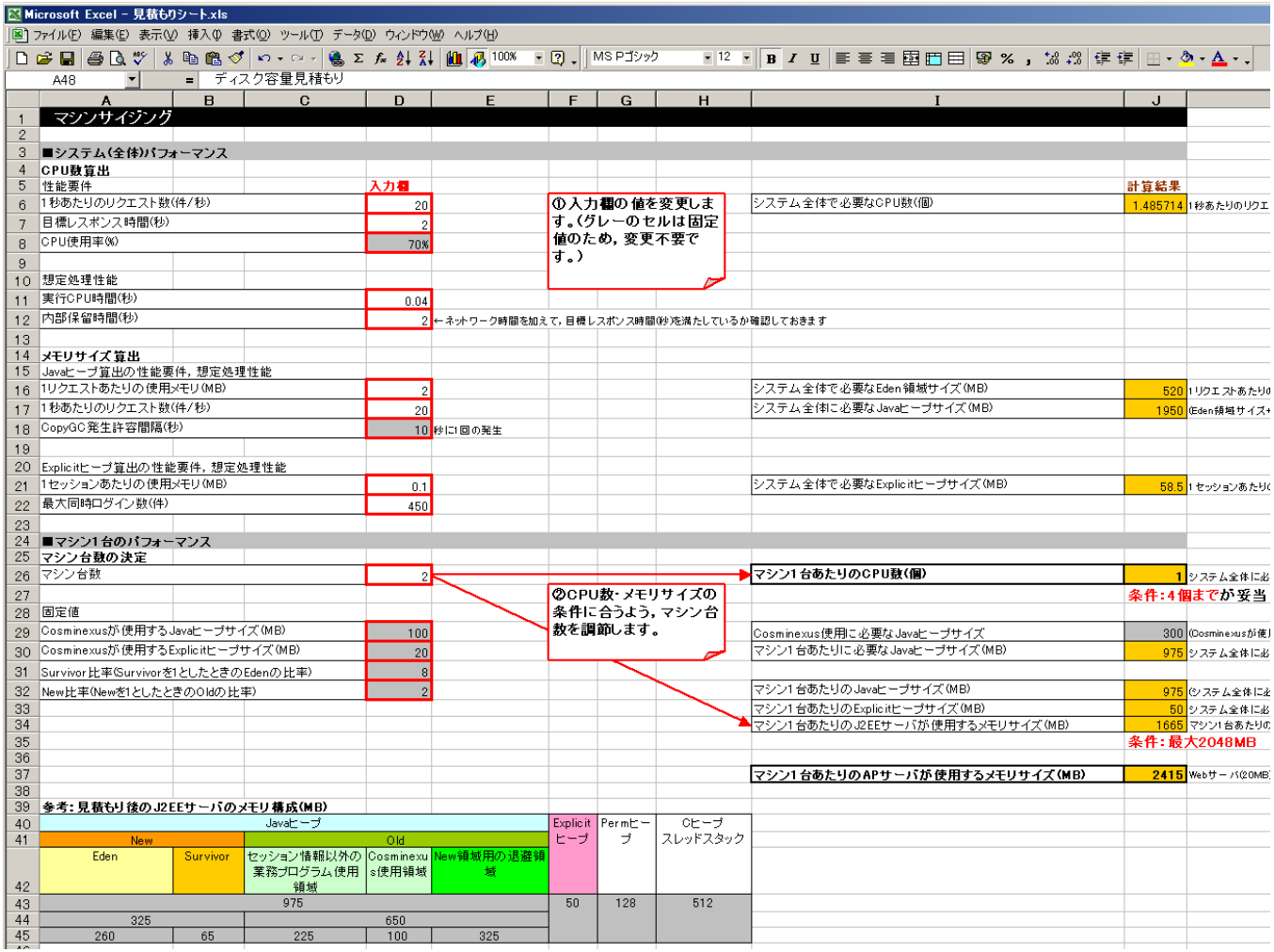
Contents

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

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



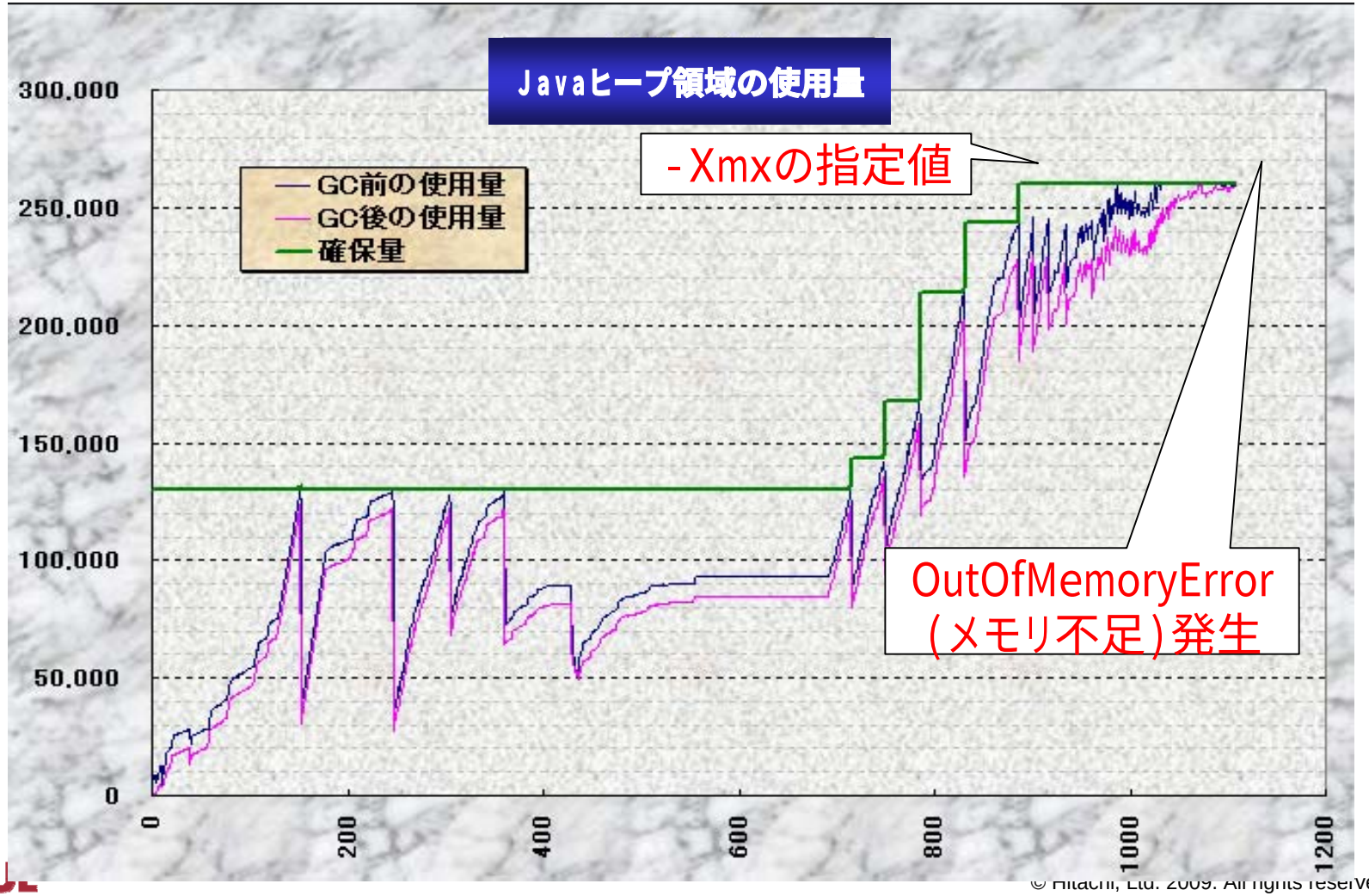
Cosminexus 認定資格講座より
http://www.hitachi.co.jp/Prod/comp/soft1/certification/cosminexus/kouza.html

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、
  ＧＣを行っても解放されないオブジェクトが継続的に増加してしまうこと



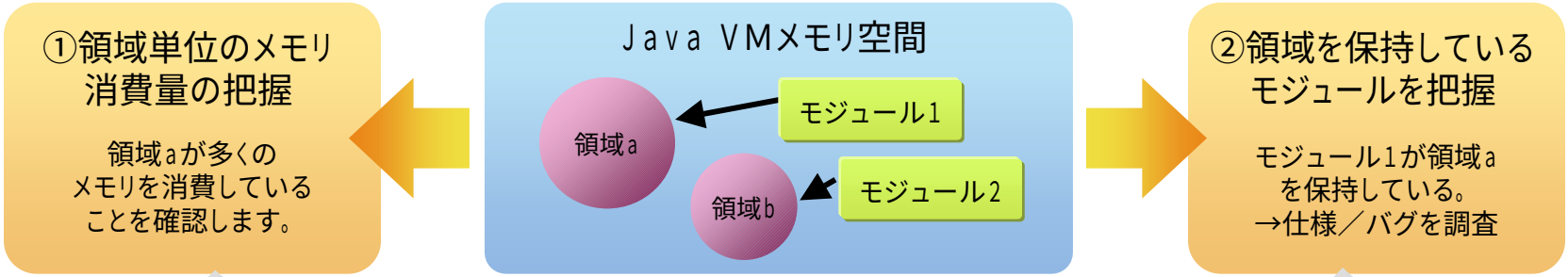
5.メモリトラブルを起こさないためには
-メモリリーク原因の特定- 「ヒーププロファイル機能」

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。



※ 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

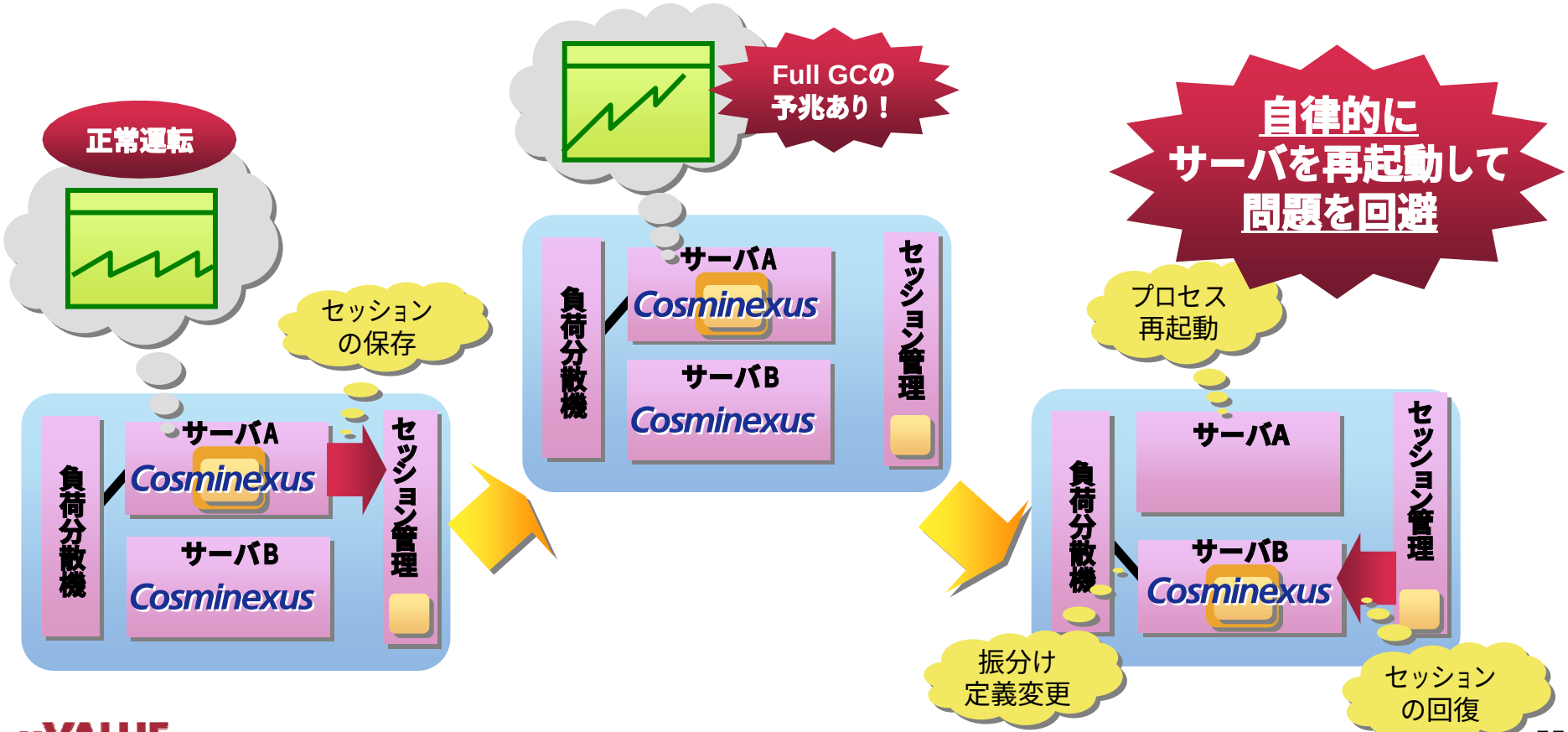
5.メモリトラブルを起こさないためには
-FullGC発生によるスローダウンの回避-

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

- 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。
- Cosminexus 日立JavaVMの明示管理ヒープと、ｱﾌﾟﾘｹｰｼｮﾝｻｰﾊﾞ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/
Cosminexus 認定資格講座
http://www.hitachi.co.jp/Prod/comp/soft1/certification/cosminexus/kouza.html

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。

END