HITACHI
Inspire the Next

システム安定稼働を実現するための JavaVMメモリサイジング

2008年6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
中島 恵

uVALUE

HITACHI
Inspire the Next

Contents

uVALUE

HITACHI
Inspire the Next

1

安定したシステムを構築するために

uVALUE

1. 安定したシステムを構築するために

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
3. バックエンドシステムによるネック

1. 安定したシステムを構築するために

HITACHI
Inspire the Next

- **性能設計の方針**

1. CPUネック

・性能要件を満たすCPU数にする　→マシンサイジング

・適切な同時実行数を設定する　→流量制御

2. メモリネック

・性能要件を満たすメモリにする　→メモリサイジング

・適切な同時実行数を設定する　→流量制御

3. バックエンドシステム（DBサーバなど）のネック

・バックエンドシステムの増強

1. 安定したシステムを構築するために

HITACHI
Inspire the Next

メモリサイズのトラブルで一番多い例：

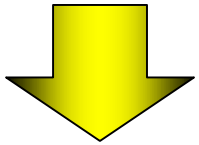
- エンドユーザ数の増加に伴い、トランザクションも増加

  →GC多発によるレスポンス遅延（スローダウン）

これまでの対応

Javaの世界ではメモリ見積りは困難。

負荷テストをやってみないと分からない。



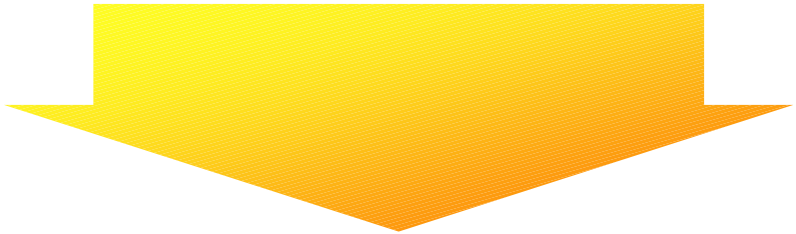
これからの対応

性能単価が判れば、メモリ（GC発生頻度）による
業務への影響度が見積もれる。

1. 安定したシステムを構築するために

HITACHI
Inspire the Next

・性能要件を満たすメモリにする　→メモリサイジング



JavaVMのメモリ管理の仕組み(ガベージコレクション)を
理解することにより、効率的なメモリサイズの見積もりを
行なうことが可能となる

HITACHI
Inspire the Next

2

GC（ガベージコレクション）とは

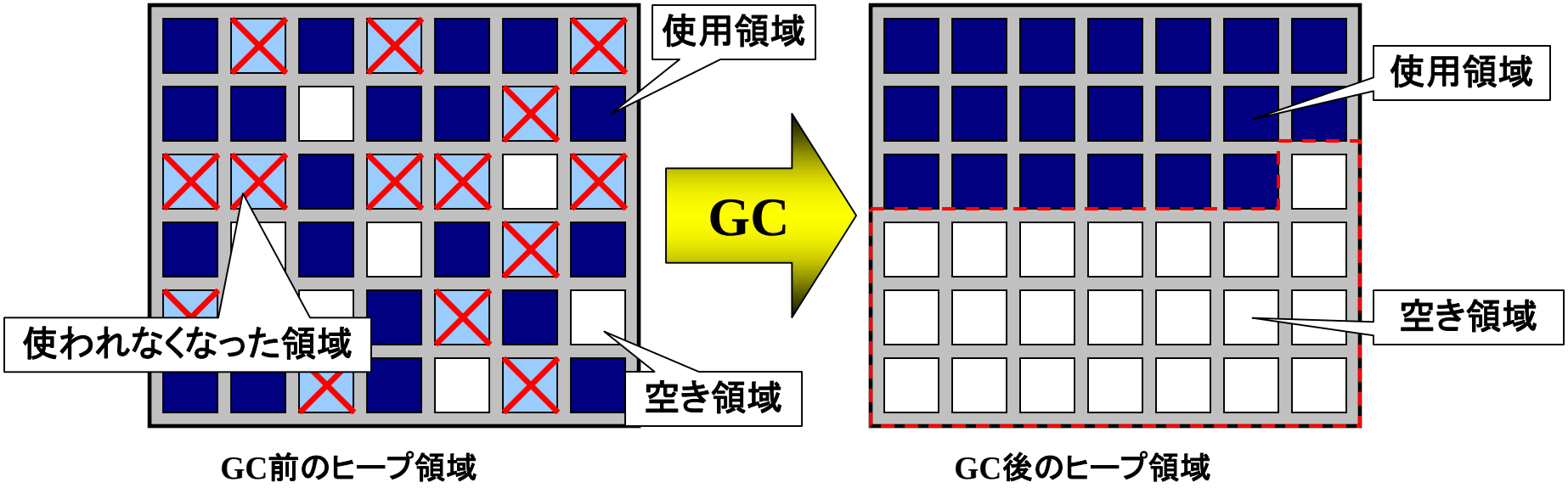
uVALUE

2. ガベージコレクションとは

HITACHI
Inspire the Next

- **ガベージコレクション（GC）とは…**

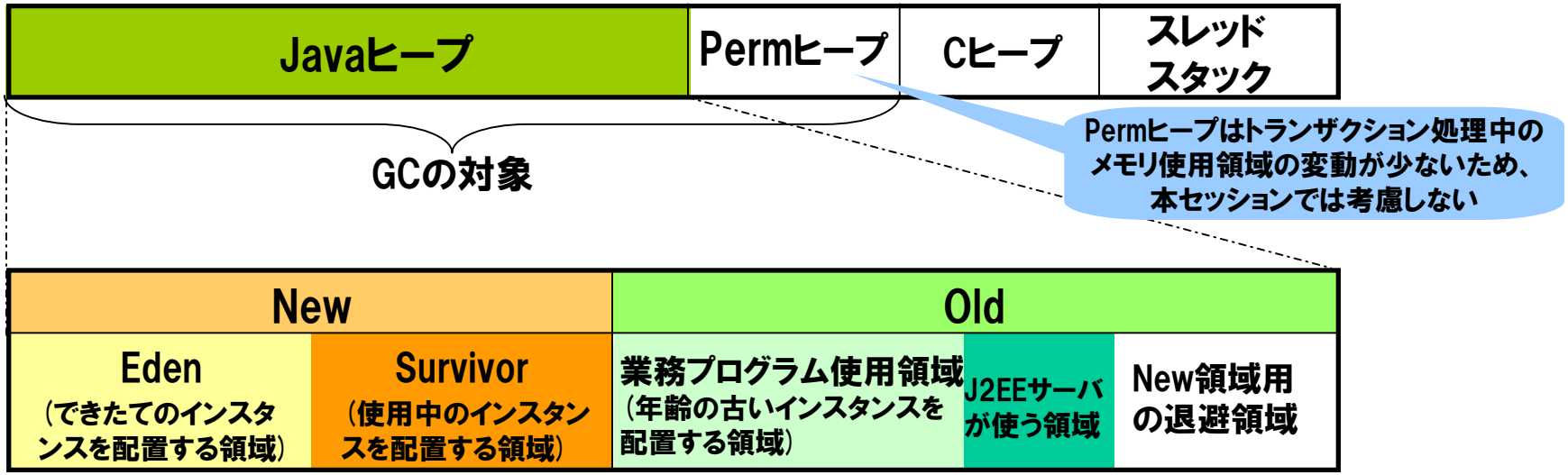
  JavaVMが管理するメモリ領域中の使用済みのメモリ領域を破棄し、空き領域を作ること



2. ガベージコレクションとは
-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

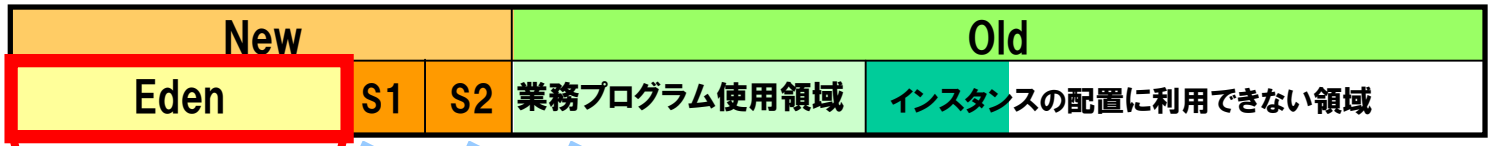
2. ガベージコレクションとは
-CopyGCとFullGC-

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とは
-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

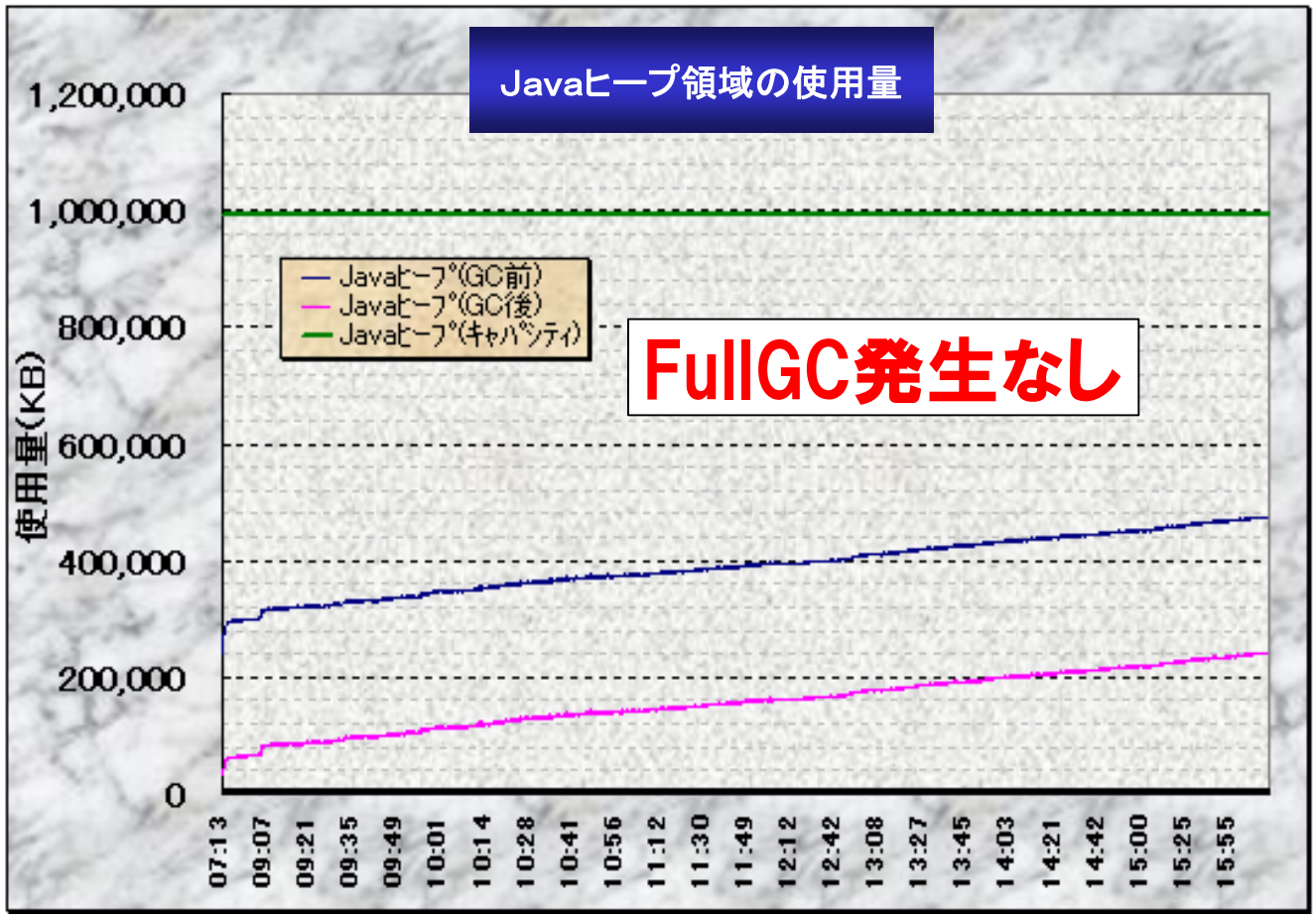
しかし、GCが発生すると、GC実行中は業務処理が停止する

レスポンス遅延を防ぐためには、ピーク時間帯に許容できる
FullGCの頻度を決め、必要なメモリサイズを確保することが重要

設定例

FullGCの許容発生間隔　　2時間に1回

2. ガベージコレクションとは
-安定時のメモリ使用状況(グラフ)-



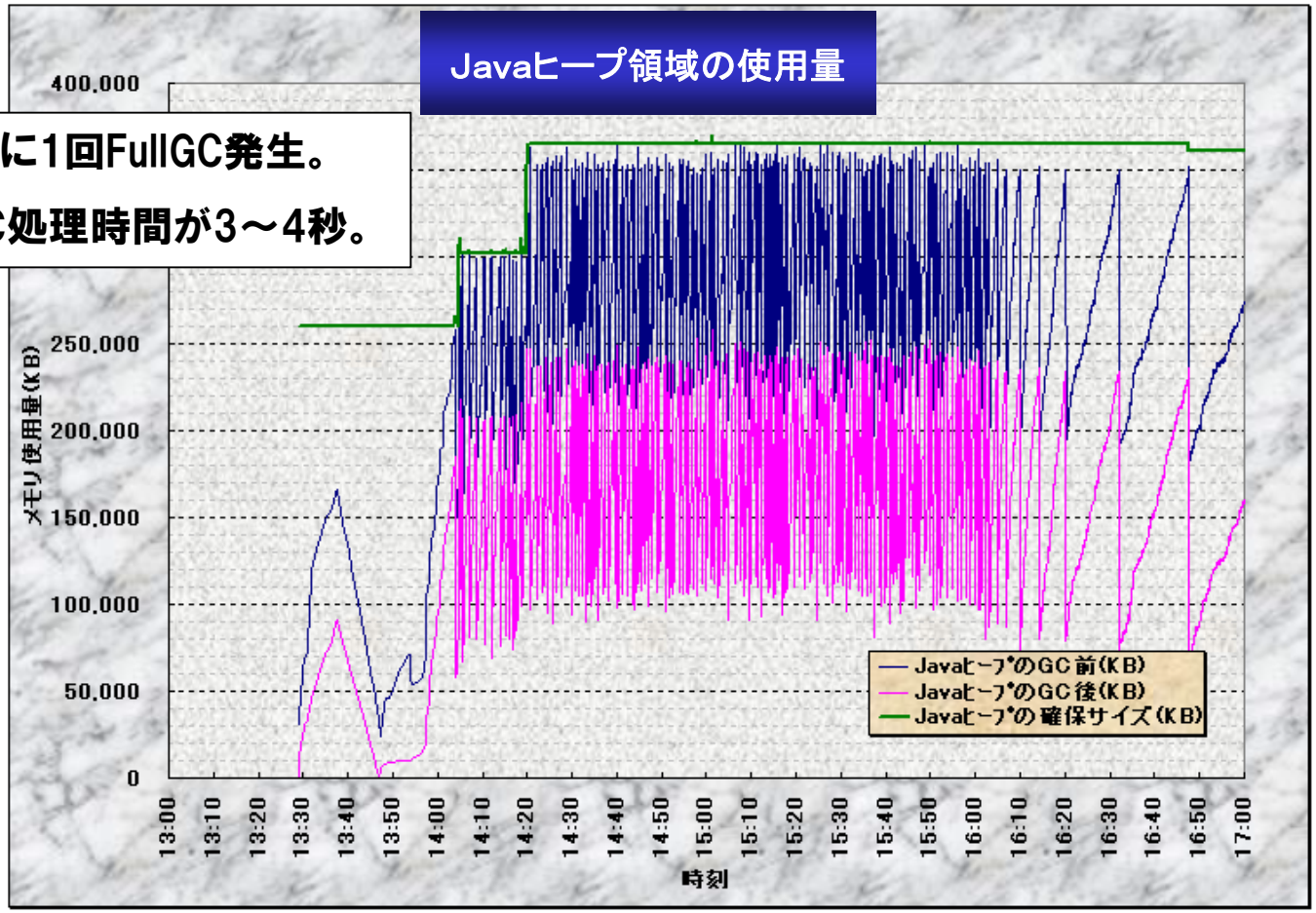
安定時のグラフ（非月末）

2. ガベージコレクションとは
-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
メモリサイジング

uVALUE

HITACHI
Inspire the Next

メモリサイズ見積もりの前提

- **要件**

  ピーク時、レスポンス遅延をしない

- **問題点**

  レスポンス遅延の大きな要因はFullGC

  FullGC実行中は業務処理が秒オーダーで停止する

- **性能設計の方針**

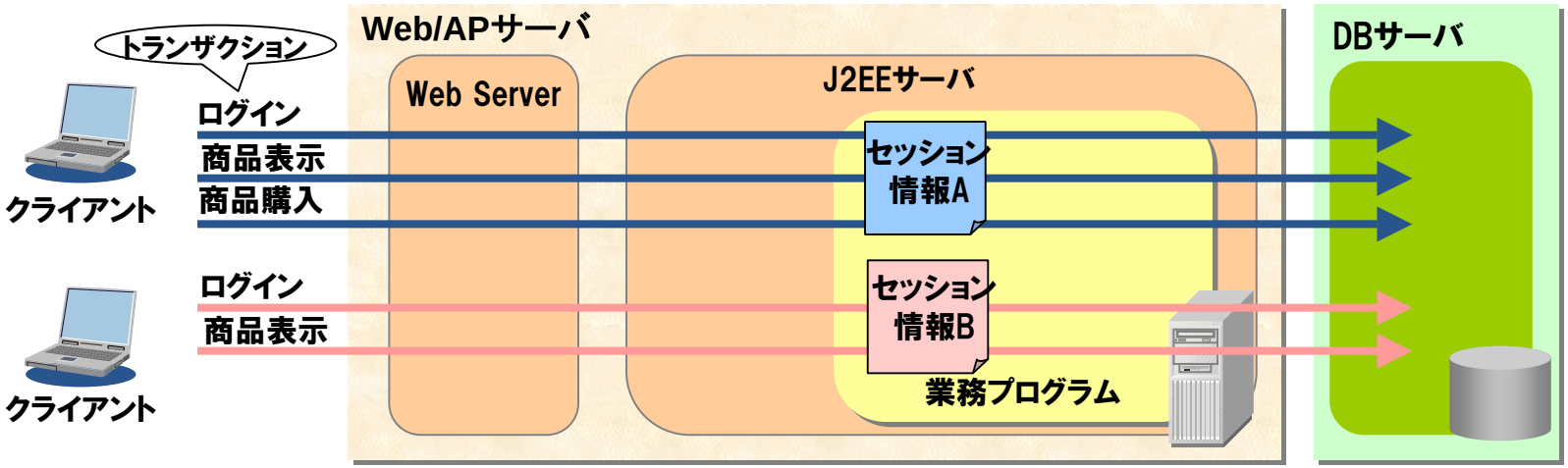
  FullGCをなるべく起こさないメモリサイズを確保する

3. メモリサイジング
-FullGC発生間隔見積もりの考え方-

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

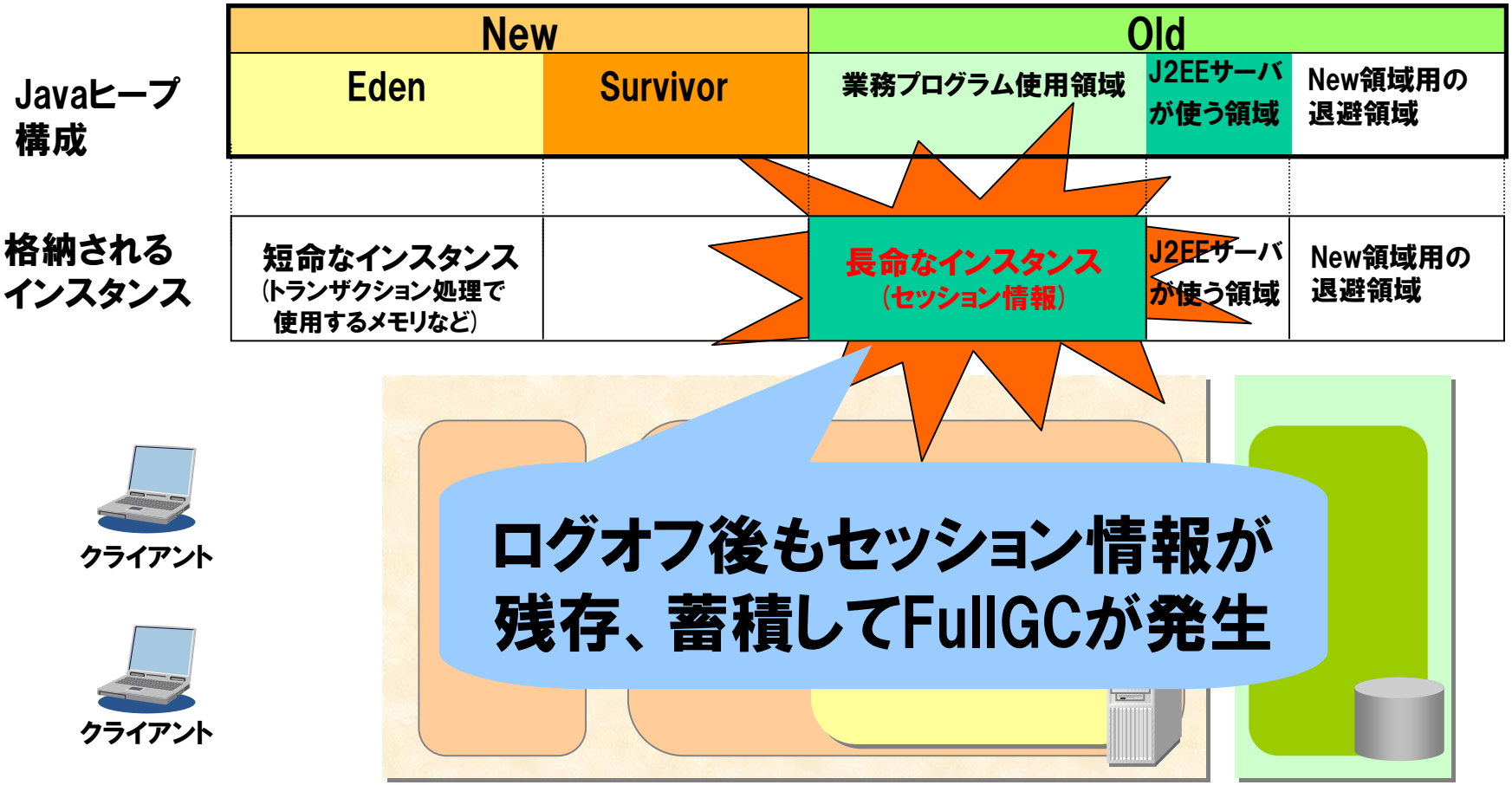
	New		Old		
Javaヒープ構成	Eden	Survivor	業務プログラム使用領域	J2EEサーバが使う領域	New領域用の退避領域
格納されるインスタンス	短命なインスタンス（トランザクション処理で使用するメモリなど）		長命なインスタンス（セッション情報）	J2EEサーバが使う領域	New領域用の退避領域



3. メモリサイジング
-FullGC発生間隔見積もりの考え方-

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



HITACHI
Inspire the Next

3.1
FullGC発生間隔の見積もり

uVALUE

3. 1 FullGC発生間隔の見積もり

HITACHI
Inspire the Next

- メモリの算出に必要な情報
  - 1セッションの処理情報
    - セッション情報サイズ
    - 1秒あたりのログイン数
    - 最大同時ログイン数
  - FullGCの許容発生間隔（要件）
- メモリサイズの見積もり手順
  1. セッション情報の蓄積によるFullGC発生間隔の算出
  2. 業務プログラム使用領域サイズの算出
  3. 最大同時ログイン数の考慮

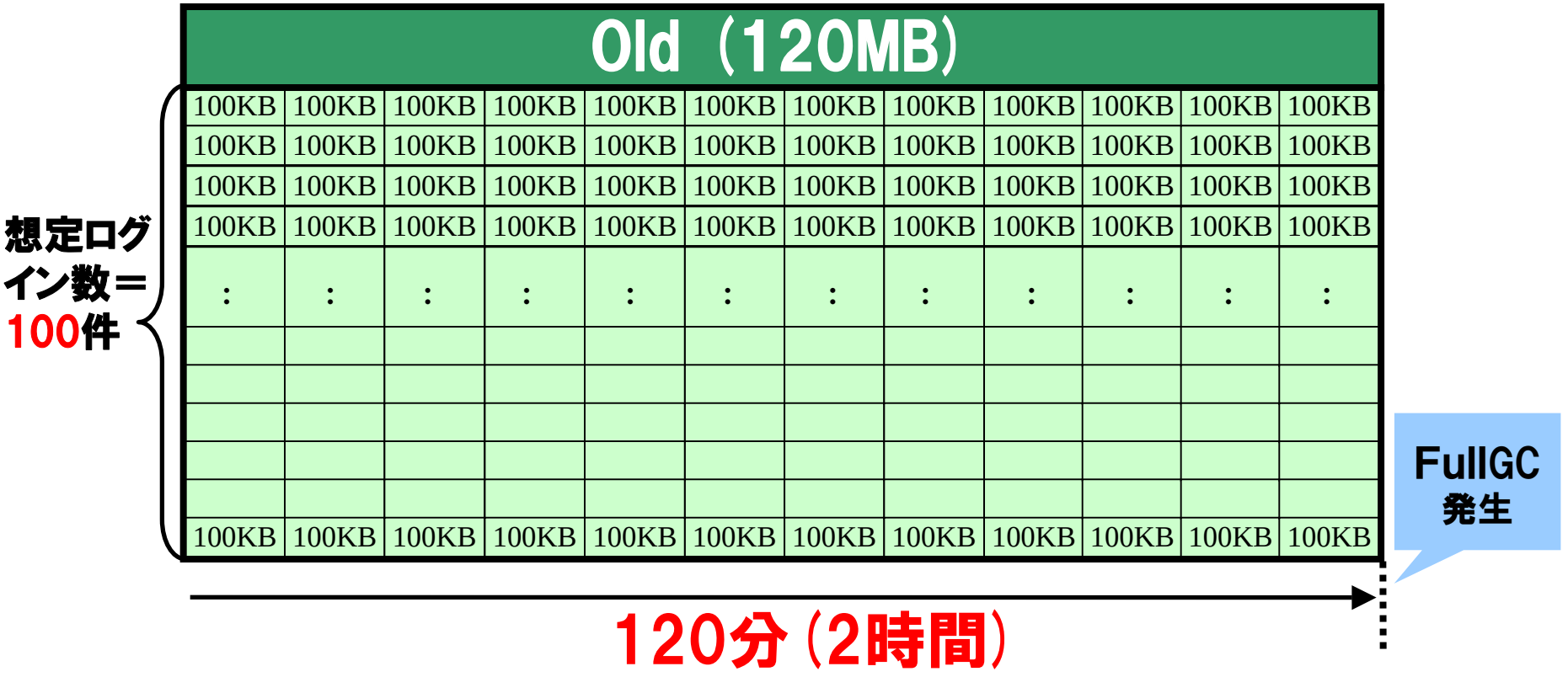
3. 1 FullGC発生間隔の見積もり

HITACHI
Inspire the Next

1. セッション情報の蓄積によるFullGC発生間隔の算出

<例>Oldの業務プログラム使用領域=120MB、セッション情報サイズ=100KB/セッション

1時間の想定ログイン数は600件（10分間では100件）



HITACHI
Inspire the Next

2. 業務プログラム使用領域サイズの算出

- FullGCの発生間隔

  Oldの業務プログラム使用領域サイズ：Old
  1秒あたりの想定ログイン数: Login
  セッション情報サイズ：M

FullGCの発生間隔　= Old／(Login×M)

<計算例>

Oldの業務プログラム使用領域サイズ：120 (MB)
1秒あたりの想定ログイン数: 600/3600 (件/秒)
セッション情報サイズ：0.1 (MB)
FullGCの発生間隔　= 120MB／($\frac{600}{3600}$ ×0.1MB) = 7200秒　→　120分

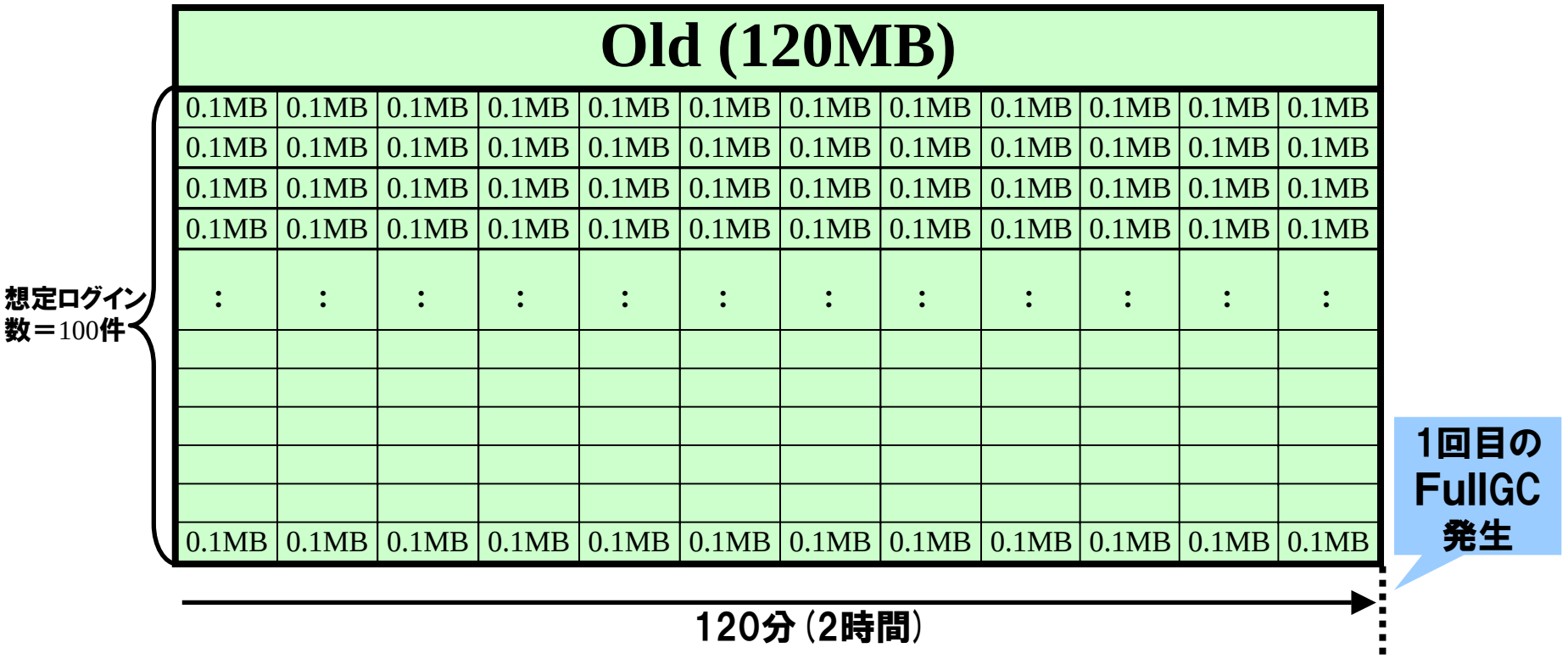
←上記の条件で計算した場合、Oldの業務プログラム使用領域サイズが120MBでは120分に1回発生

3. 1 FullGC発生間隔の見積もり

HITACHI
Inspire the Next

3.最大同時ログイン数の考慮

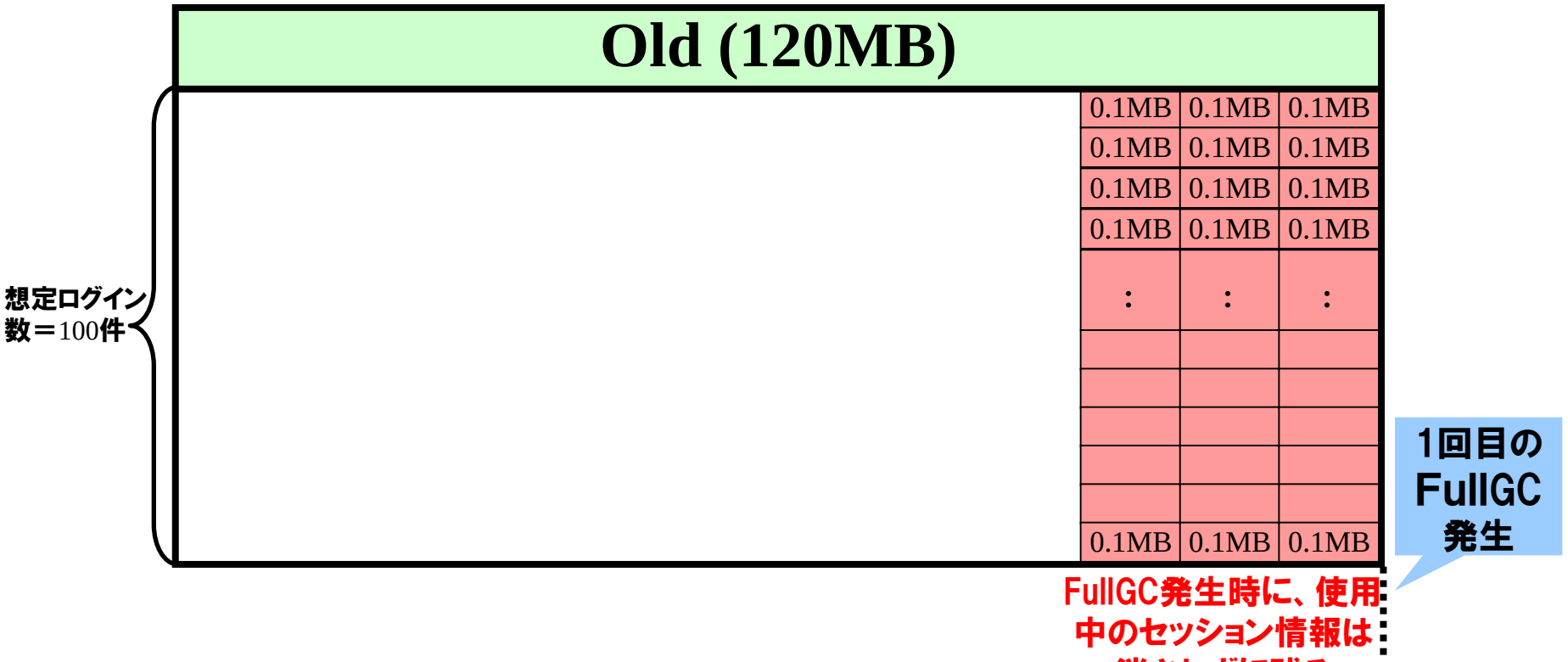
<例>最大同時ログイン数=300件、Oldの業務プログラム使用領域=120MB、1時間の想定ログイン数は600件（10分間では100件）、セッション情報サイズ=0.1MB/セッション



HITACHI
Inspire the Next

3.最大同時ログイン数の考慮

<例>**最大同時ログイン数=300件**、Oldの業務プログラム使用領域=120MB、1時間の想定ログイン数は600件（10分間では100件）、セッション情報サイズ=0.1MB/セッション



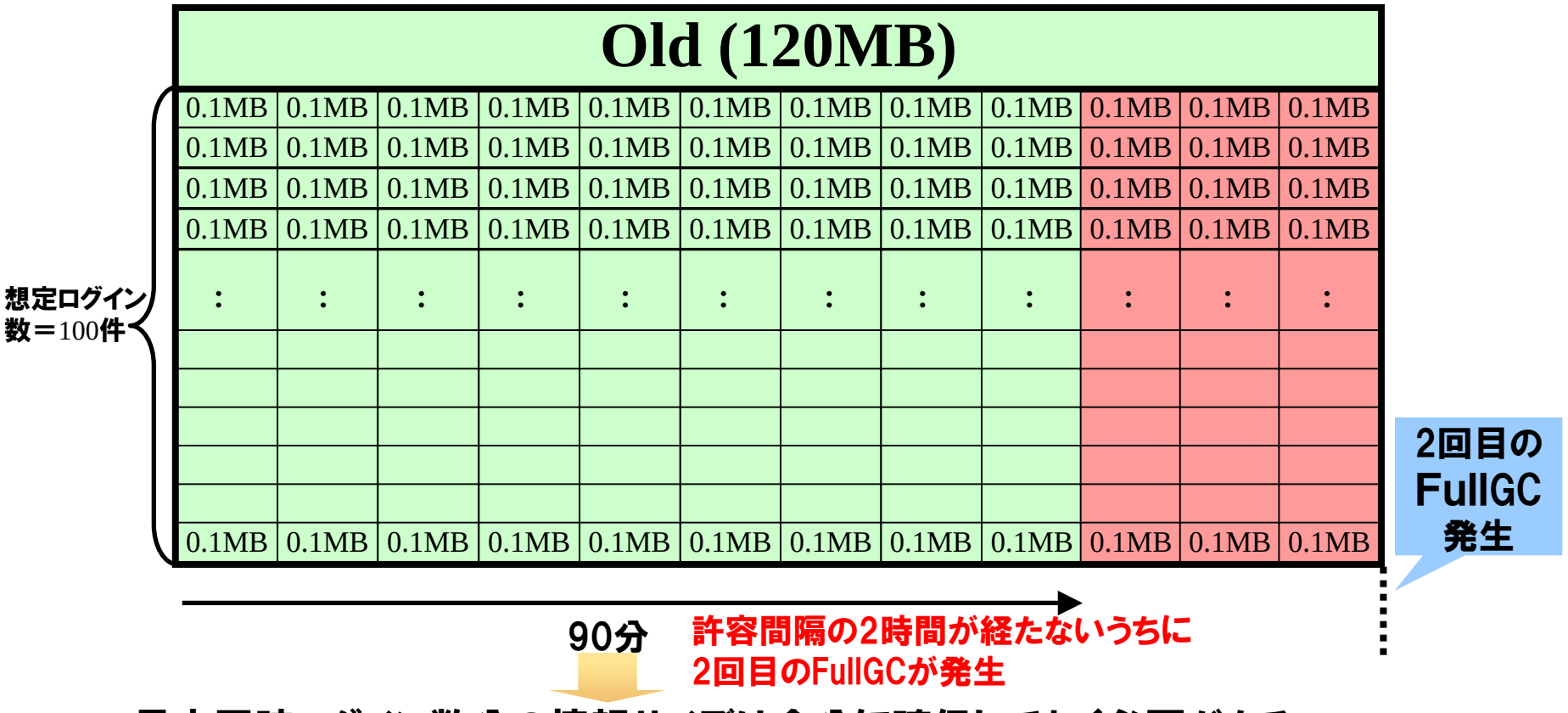
※最大同時ログイン数（300）残る可能性がある

3. 1 FullGC発生間隔の見積もり

HITACHI
Inspire the Next

3.最大同時ログイン数の考慮

<例>最大同時ログイン数=300件、Oldの業務プログラム使用領域=120MB、1時間の想定ログイン数は600件（10分間では100件）、セッション情報サイズ=0.1MB/セッション



最大同時ログイン数分の情報サイズは余分に確保しておく必要がある

HITACHI Inspire the Next

3.最大同時ログイン数の考慮

「最大同時ログイン数×セッション情報サイズ」を加えたサイズを確保する

<計算例>最大同時ログイン数分の情報サイズ=300×0.1MB=30MB

Old(120MB)												Old(30MB)		
0.1MB	0.1MB	0.1MB	0.1MB	0.1MB	0.1MB	0.1MB	0.1MB	0.1MB	0.1MB	0.1MB	0.1MB	0.1MB	0.1MB	0.1MB
0.1MB	0.1MB	0.1MB	0.1MB	0.1MB	0.1MB	0.1MB	0.1MB	0.1MB	0.1MB	0.1MB	0.1MB	0.1MB	0.1MB	0.1MB
0.1MB	0.1MB	0.1MB	0.1MB	0.1MB	0.1MB	0.1MB	0.1MB	0.1MB	0.1MB	0.1MB	0.1MB	0.1MB	0.1MB	0.1MB
0.1MB	0.1MB	0.1MB	0.1MB	0.1MB	0.1MB	0.1MB	0.1MB	0.1MB	0.1MB	0.1MB	0.1MB	0.1MB	0.1MB	0.1MB
:	:	:	:	:	:	:	:	:	:	:	:	:	:	:
0.1MB	0.1MB	0.1MB	0.1MB	0.1MB	0.1MB	0.1MB	0.1MB	0.1MB	0.1MB	0.1MB	0.1MB	0.1MB	0.1MB	0.1MB

必要なOldの業務プログラム使用領域サイズ

=セッション情報の蓄積から求めた必要サイズ＋最大同時ログイン数分の情報サイズ

ここでは、120MB+30MB=150MB

150MBの業務プログラム使用領域サイズが必要

HITACHI
Inspire the Next

3.2
Javaヒープサイズの見積もり

uVALUE

3. 2 Javaヒープサイズの見積もり

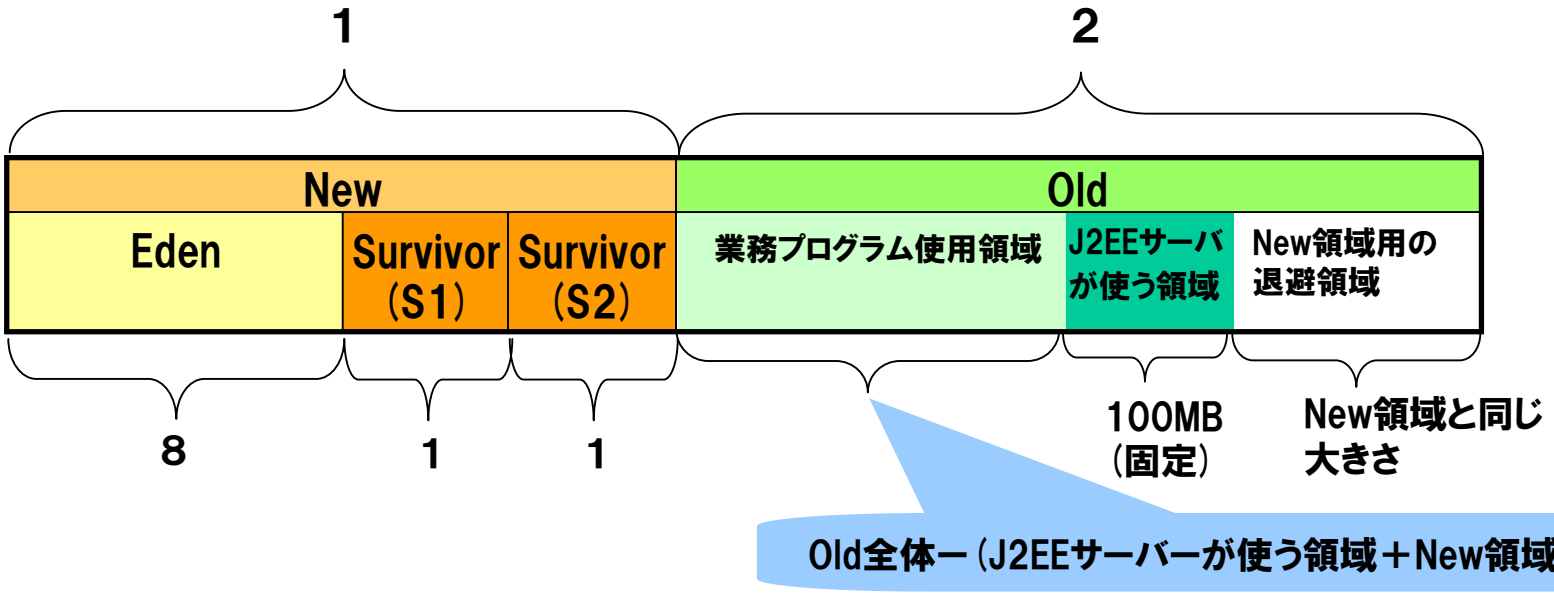
HITACHI
Inspire the Next

●Javaヒープサイズの設定

必要なOld領域サイズが確保できるJavaヒープのサイズを見積もる

**Javaヒープの内訳は比率が決まっているため、
必要なOld領域サイズから，必要なJavaヒープサイズを求められる**

・Javaヒープの内訳



※この比率はデフォルト値である

※New領域の比率のデフォルト値はプラットフォームによって違う

3. 2 Javaヒープサイズの見積もり

HITACHI
Inspire the Next

計算例

要件

FullGCの許容発生間隔	2時間に1回

要件を満たす領域サイズ

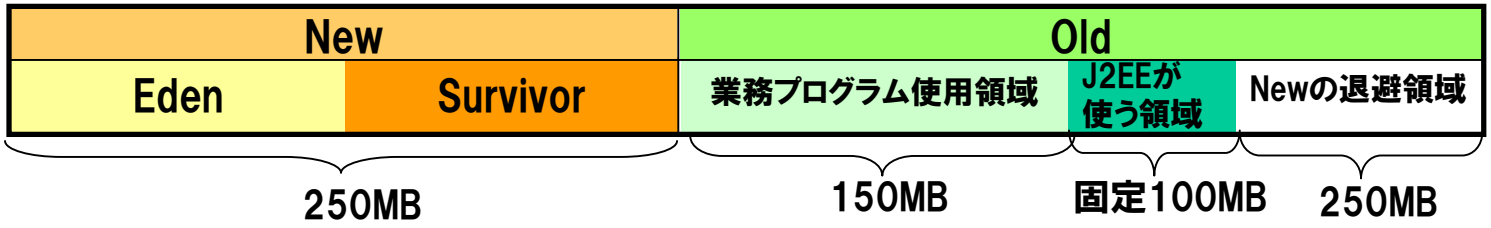
Oldの業務プログラム使用領域サイズ	150MB

Old使用領域サイズから、必要なJavaヒープサイズを求める

3. 2 Javaヒープサイズの見積もり

HITACHI
Inspire the Next

- Oldの業務プログラム使用領域を150MB確保できるJavaヒープサイズは？



Old領域 = （150MB+100MB＋ New領域と同じ大きさの退避領域）

New領域:Old領域 = 1:2

（150MB+100MB）はNew領域と同じ大きさ

Javaヒープサイズ = 250MB+500MB = **750MB**

HITACHI
Inspire the Next

4

Cosminexus(コズミネクサス)での取り組み

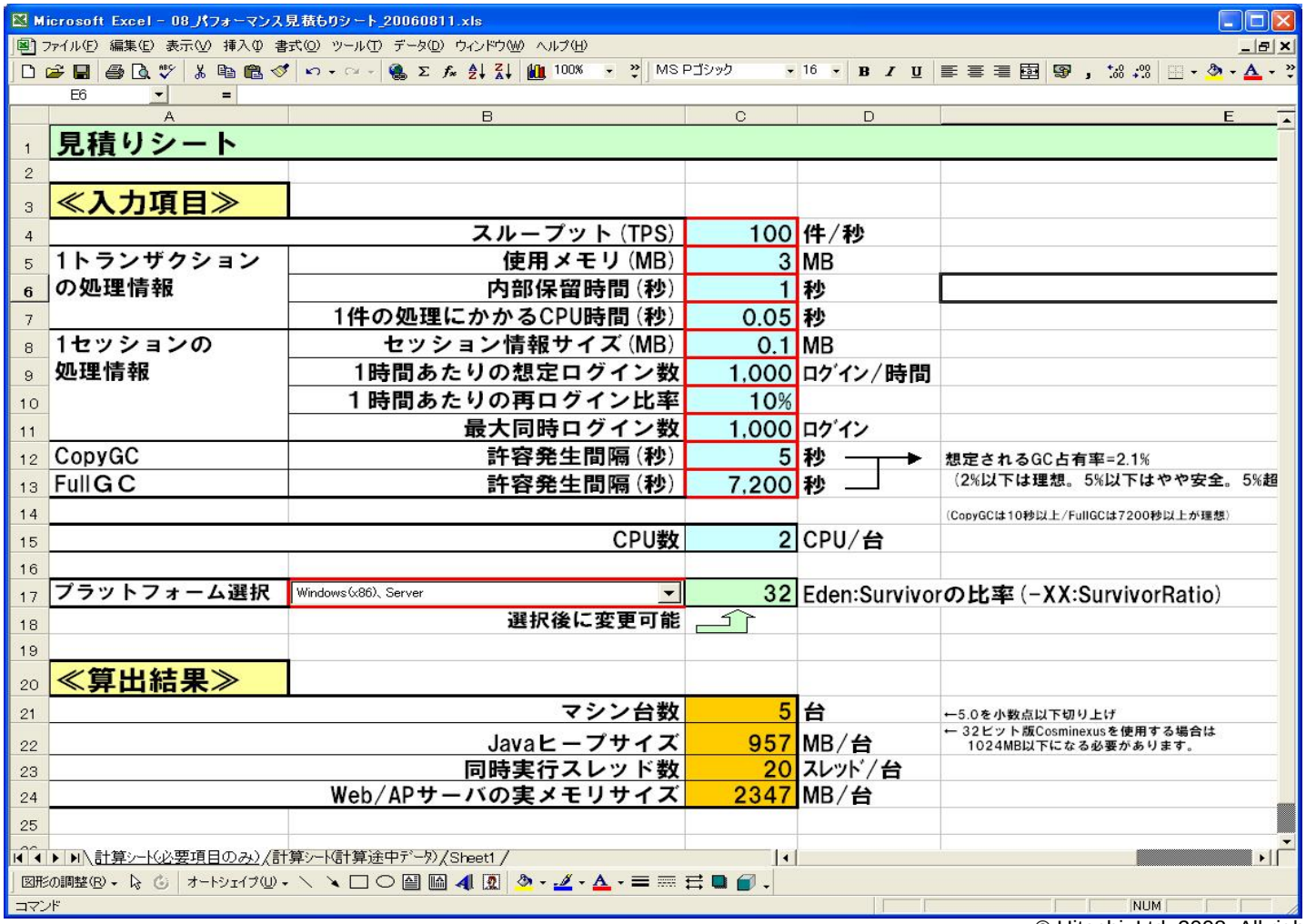
uVALUE

4. Cosminexusでの取り組み(1)
-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」を用いた見積もり数値の検討

赤枠の項目の一部を入力し、サイジングした数値を検討・確認する



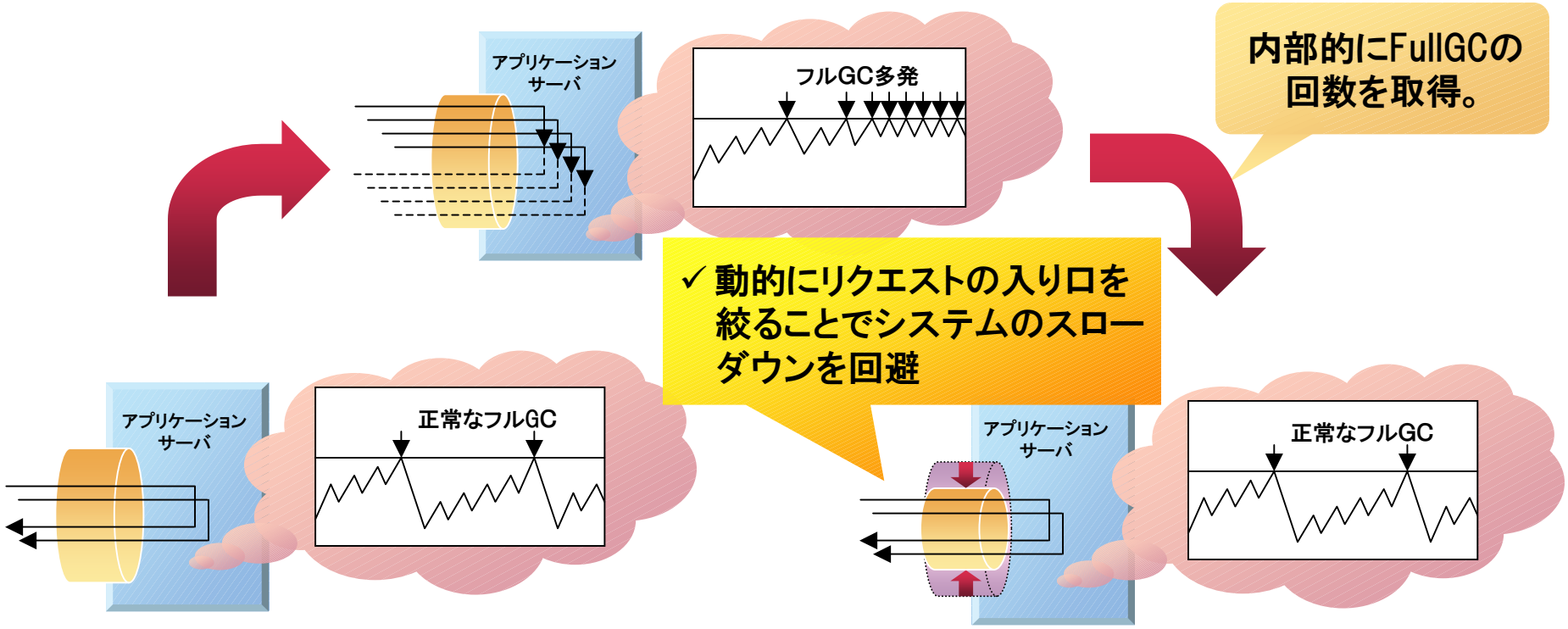
4. Cosminexusでの取り組み(2)
-予期せぬFullGC多発の検知-

HITACHI
Inspire the Next

- リクエスト集中により使い捨てオブジェクトが大量に生成されFullGCが多発しシステムがスローダウンしてしまう問題を何とかしたい・・・

ポイント

FullGCの多発を検知して自動的に回避アクションを取ることが可能。
例えば，FullGCの多発を検知し、自動的にリクエストの入口を絞り、
システムのスローダウンを防ぐことができる。



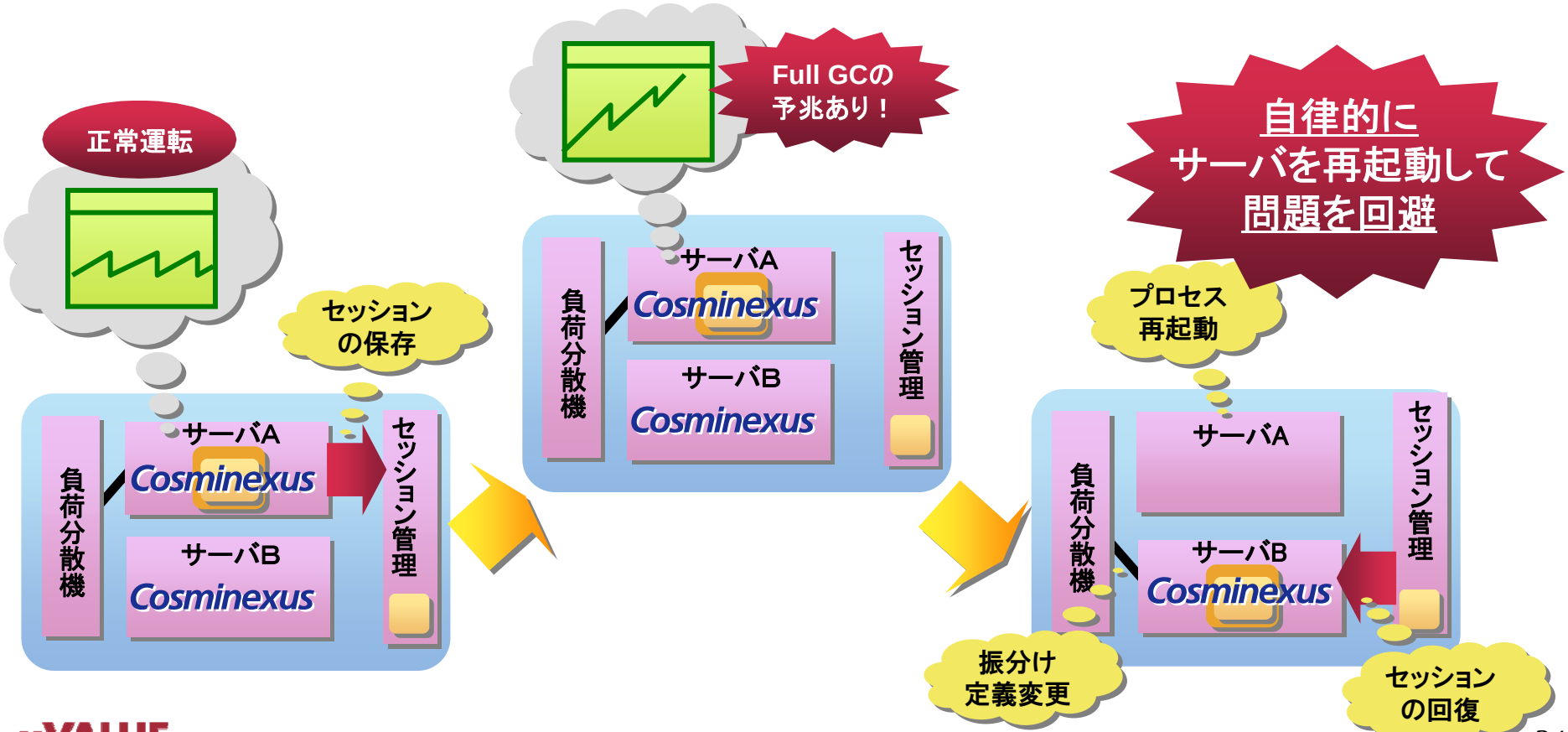
4. Cosminexusでの取り組み(3)
-FullGC発生によるスローダウンの回避-

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。



4. Cosminexusでの取り組み(4)
-メモリリーク原因の特定-

HITACHI
Inspire the Next

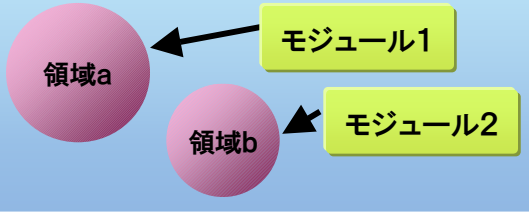
- メモリリークしているが、原因が特定できない。

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。本番環境で発生した時点でサーバーを再起動することなく情報を取得することができるため、テスト環境では再現が困難なメモリリークの不具合を早期に解決。

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。



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

```
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
  classX
------
classB
  classC
    classD
      classX
------
classE
  classA
    classX
```

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る

HITACHI
Inspire the Next

5

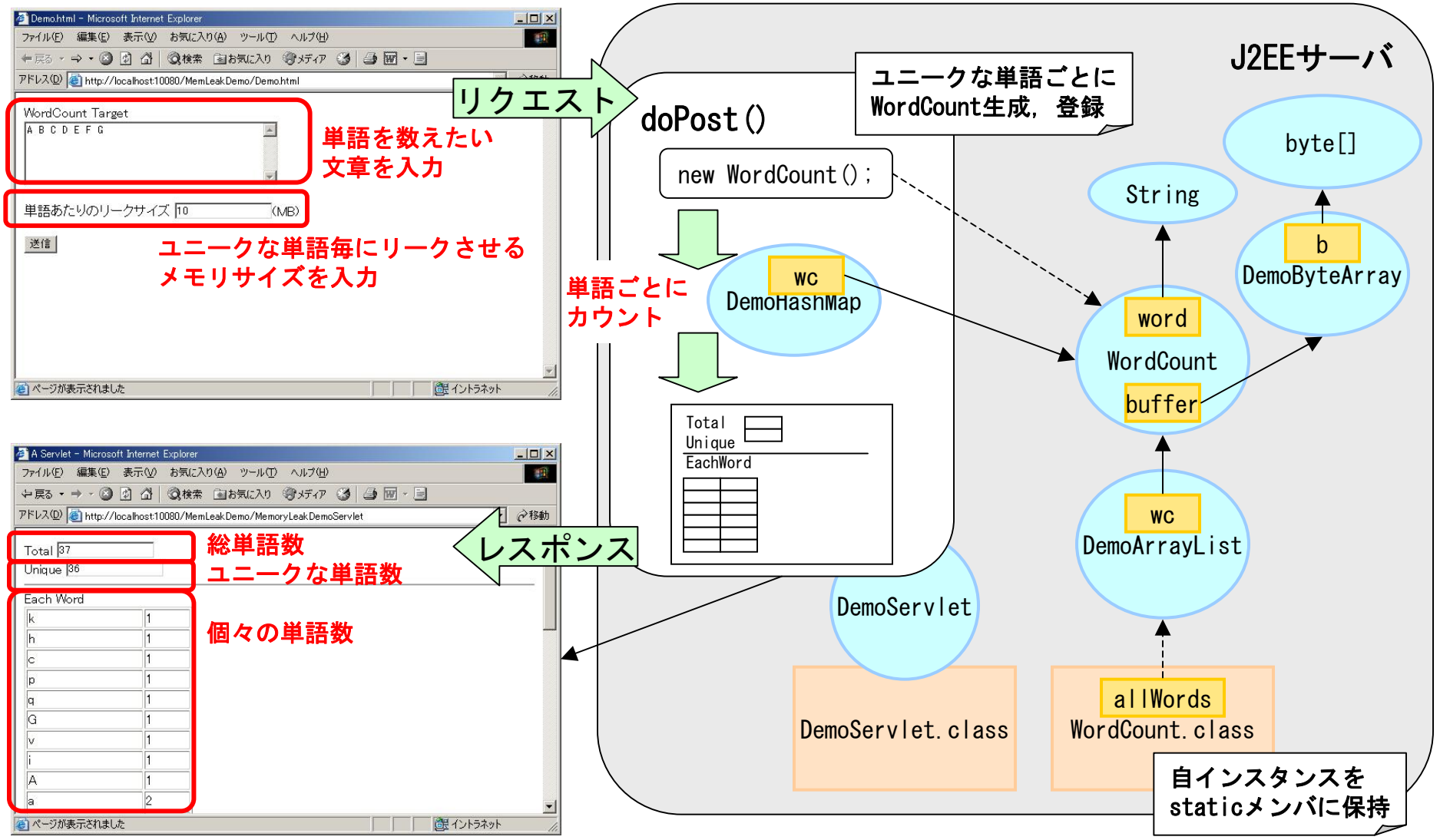
デモンストレーション

uVALUE

5. メモリリーク調査デモ

HITACHI
Inspire the Next

入力した文章の単語数をカウントするデモプログラム



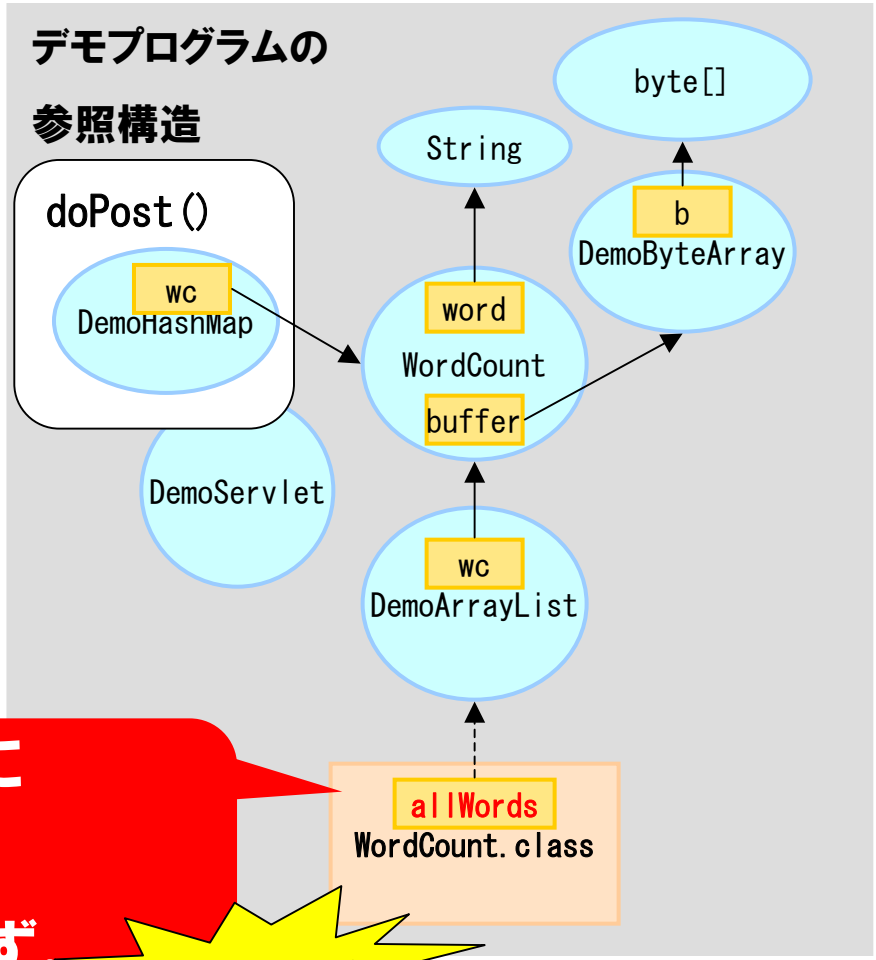
5. メモリリーク調査デモ

HITACHI
Inspire the Next

WordCountクラスの分析

WordCountクラス

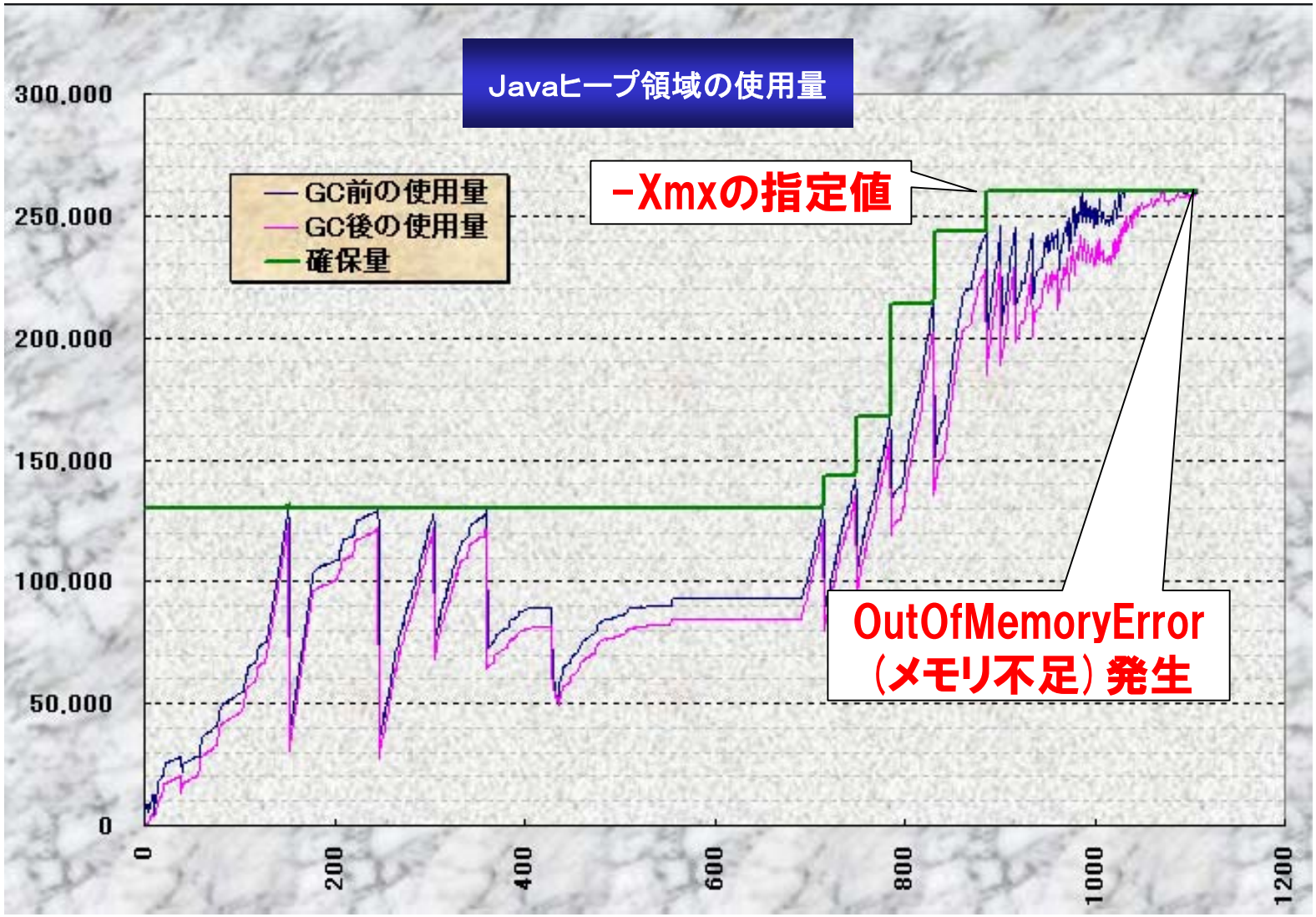
```
package demo;

import java.util.List;

import demo.base.DemoArrayList;
import demo.base.DemoByteArray;

public class WordCount {
    private static List allWords = new DemoArrayList();

    private String word;
    private int count;
    private DemoByteArray buffer;

    public WordCount(String word, int size) {
        allWords.add(this);
        this.buffer = new DemoByteArray(size);
        this.word = word;
        this.count = 1;
    }
                    （以下省略）
```



- StaticメンバのDemoArrayListに自インスタンスを保持。
- 保持したインスタンスを解放せず。

見直し対象

5. メモリリークの事例



まとめ

- 適切なメモリサイズを見積もることで、
  システムスローダウンの要因であるFullGCを
  コントロールすることが可能
- 見積もりよりも過剰なリクエストが殺到した時の
  緊急避難的な処置は、アプリケーションサーバー側で
  対処する仕組みがある(とはいえ万能ではない)
- メモリリークなどの不良によって、メモリが圧迫される
  場合もある。
  これについての調査方法は確立されている。

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。

uVALUE

HITACHI

Inspire the Next