

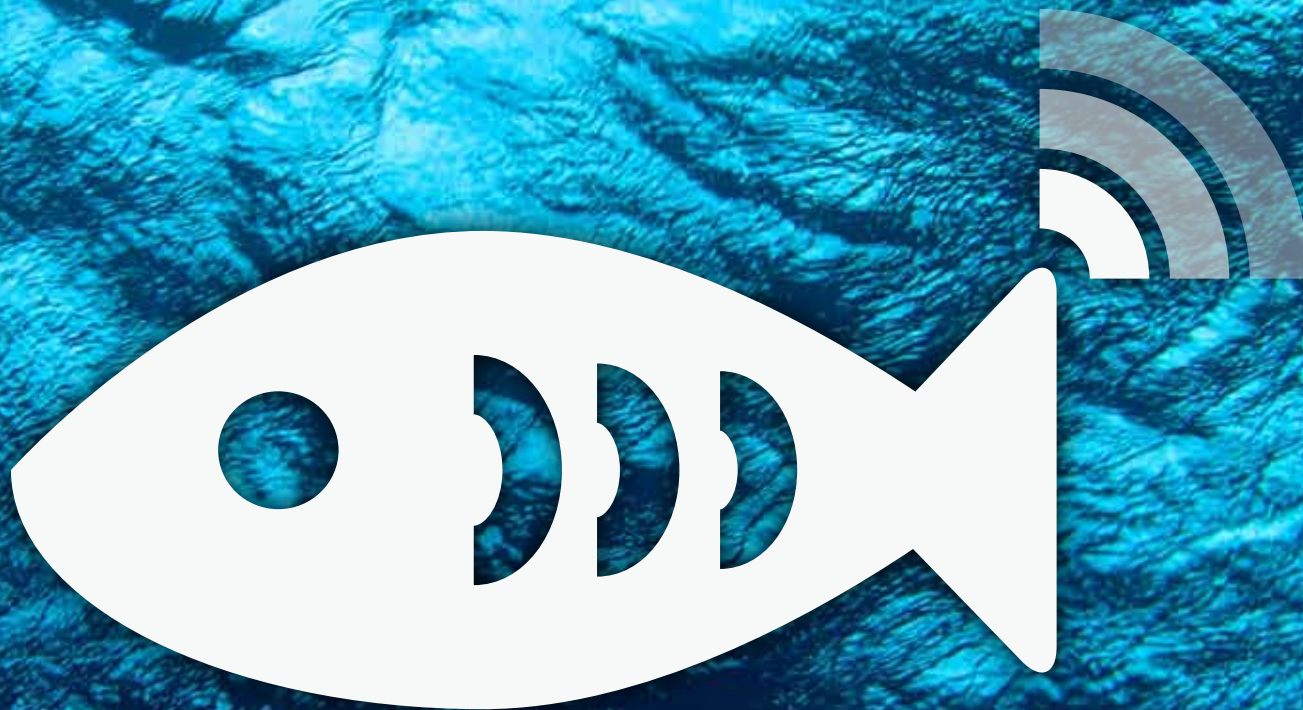
Glassfish で即習！ Java EE パフォーマンスチューニングと トラブルシューティング

Glassfish ユーザグループジャパン
岡崎 隆之 (watermint.org)



Takayuki Okazaki
@watermint
watermint.org

Glassfish.jp



Since 2008.1

本日のゴール

パフォーマンスチューニング、トラブル

シューティングへの取り組み方を紹介。

今まで出来ていなかったこと、これから

実践していくべきことの紹介。

パフォーマンスチューニングへの

取り組み方



パフォーマンスと一言に言っても ...

一つの処理が完了するまでの時間 (レイテンシ)。

一定時間内に処理できるトランザクション数 (スループット)。

一つの処理が完了するまでにかかるコスト。



「チューニング作業だけ」で解決する問題は限定的。

例) 燃費を良くしたい。

→ エンジンを新設計する vs. オイル交換をする



- プロジェクトの企画
 - そのサービスの本質はなにか？

- 要件定義、特に非機能要件
 - 例) 95% の処理が 2 秒以内に完了



サービスの本質について

クライアントデバイスの高度化によって、ユーザエクスペリエンスのハードルが高くなった。

気持ちよく使えることが当たり前、
ひっかかりや、待ち時間は気持ち悪い。

なにが「気持ちよさ」を構成するか → 要件定義

例) 読むスピードと同じぐらいのスピードでダウンロードされる

→ 読むスピードは？頻度は？

→ どれぐらいの人が同時に使うのか？

理想のパフォーマンスチューニング

設計

トランザクションの性質と制約は何か。

実装

継続的なパフォーマンスチューニング。

維持

非機能要件は変化していないか。

トランザクションの性質と制約を発見することからスタートする。

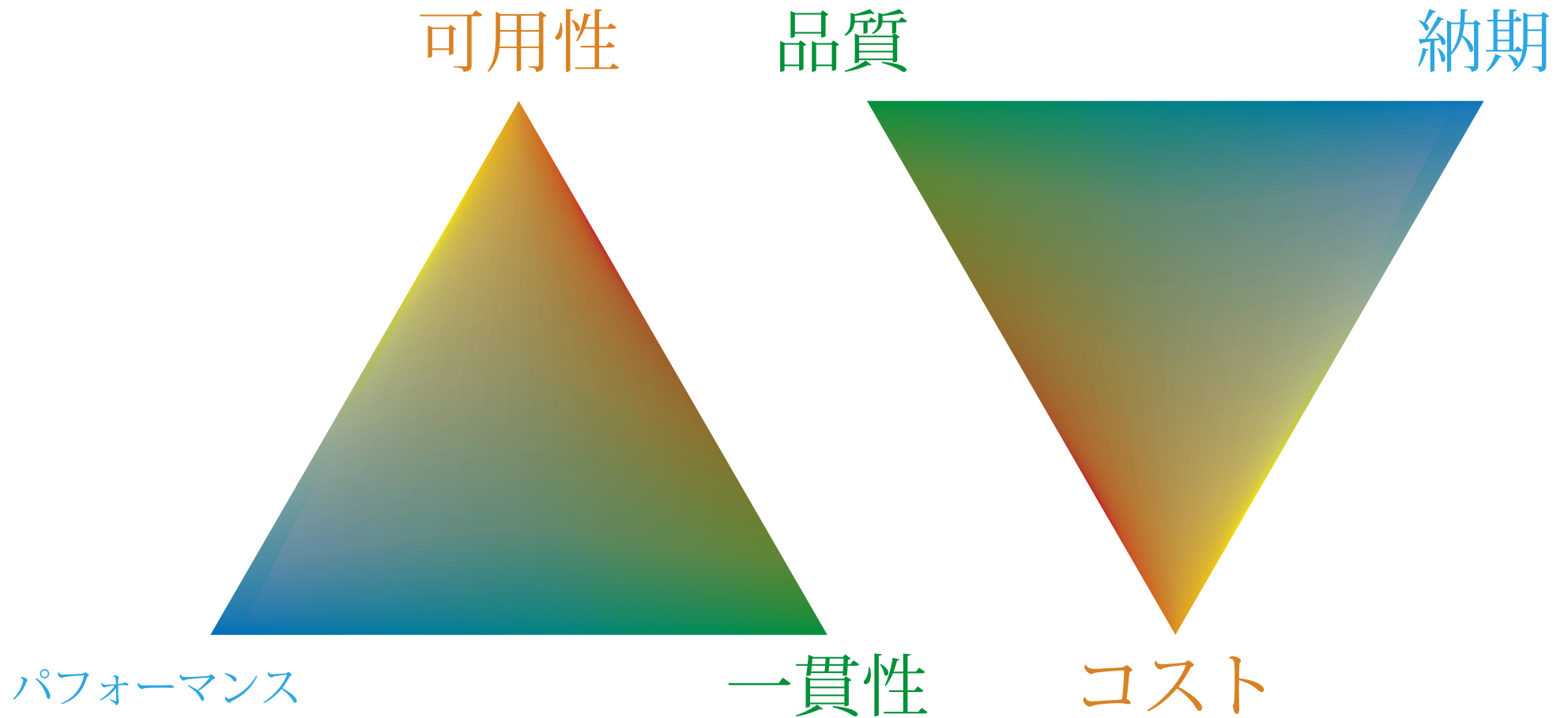
この処理は分散処理可能か？

この処理を実行するために使ってよいコストはいくらか？

同時に使用する資源はいくつか？

それらの資源はいくつまで使えるか？

要件と制約からバランスを見つける



継続的なパフォーマンス チューニング



理想のパフォーマンスチューニング

継続的インテグレーションツールの活用


<http://hudson-ci.org>



非機能要件は変化していないか？

処理しなければならないトランザクション数

レイテンシに対する要求 (2 秒→ 0.7 秒)

A school of approximately 15-20 small, bright orange and yellow fish swimming in a dark blue-green tank. The fish are of various sizes and are scattered throughout the frame, with some in sharp focus and others blurred in the background. The lighting is soft, highlighting the vibrant colors of the fish against the dark water.

パフォーマンスチューニングの実践

パフォーマンスチューニングの実践

1. ボトルネックを探す
2. 原因を探る
3. 設計や実装方式を再考する
 - 3.1. テスト実装などで検証する
4. 修正を実装し、テストする
 - 4.1. テストを自動化し、継続的にテストする

ボトルネックを探す

1. 全体処理の過程で、どのような処理が行われているか洗い出す。
2. 個々の処理に対してそれぞれがどの程度時間、リソースを消費しているか洗い出す。

テスト実装等で検証する

なるべく本番環境に近い、環境、規模、負荷を再現する。

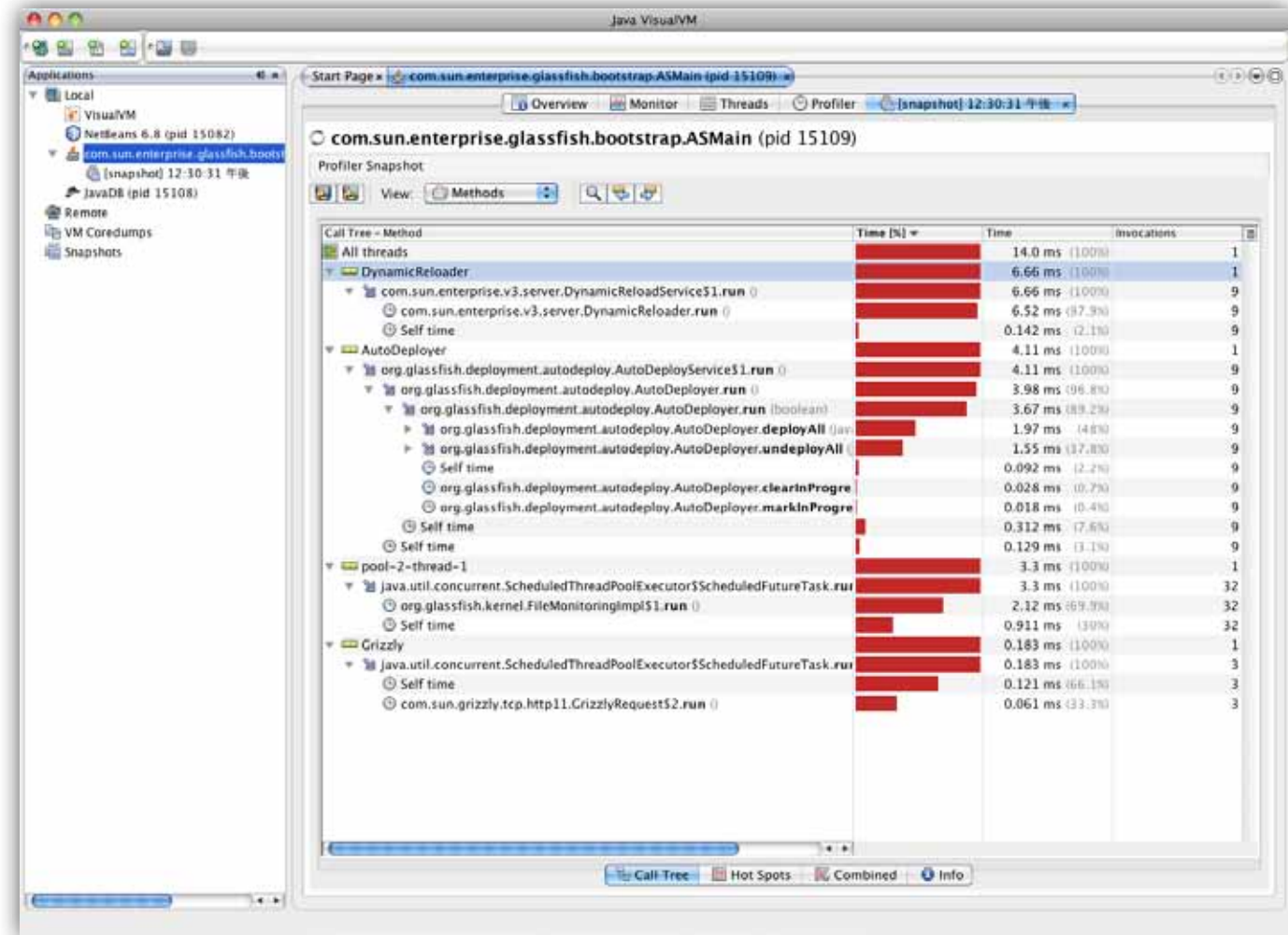
CPUの種類、コア数、スレッド数、メモリ容量、etc

一時的に Amazon EC2 等を使って検証をする方法がリーズナブル

パフォーマンスチューニングの実践

ボトルネックを探す

① visualvm



パフォーマンスチューニングの実践

ボトルネックを探す【上級】

② dtrace

より詳しくピンポイントで

調べたい場合に。

動作環境

Solaris 10, OpenSolaris

FreeBSD 7.1 以降

Mac OS X (Leopard 以降)

```
target$1:::method-entry
{
    time[depth] = timestamp;
    depth++;
    includeMeasurement[depth] = 0;
}

target$1:::method-return
/depth > 0/
{
    depth--;
    delta = timestamp - time[depth];
    inclusive += delta;
    includeMeasurement[depth] += delta;
    delta -= includeMeasurement[depth+1];
    includeMeasurement[depth+1] = 0;
    exclusive += delta;
}
```

トラブルシューティングへの
取り組み方

トラブルシューティングの実践

パフォーマンス劣化

メモリリーク

文字化け

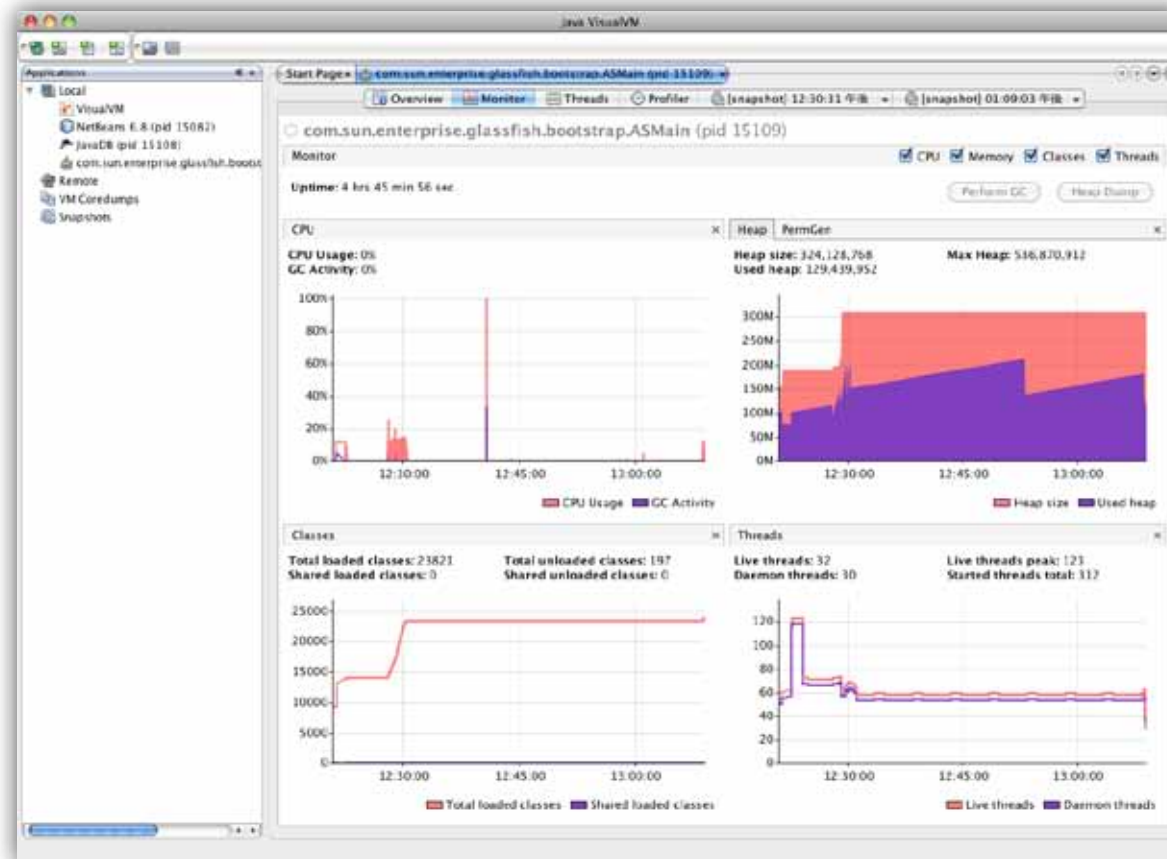
アプリケーションが停止、クラッシュ

セキュリティ

トラブルシューティングの実践

パフォーマンス劣化やメモリリーク

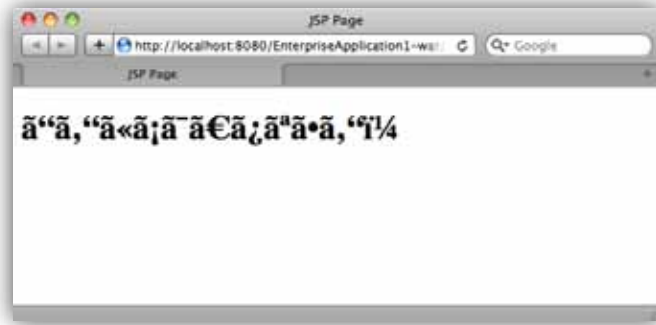
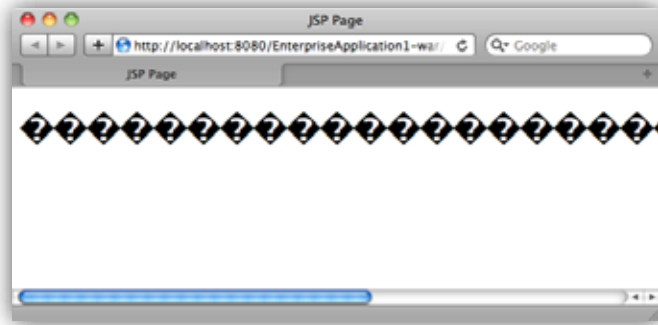
プロファイラで対象 Java VM に接続し統計を分析



トラブルシューティングの実践

文字化け

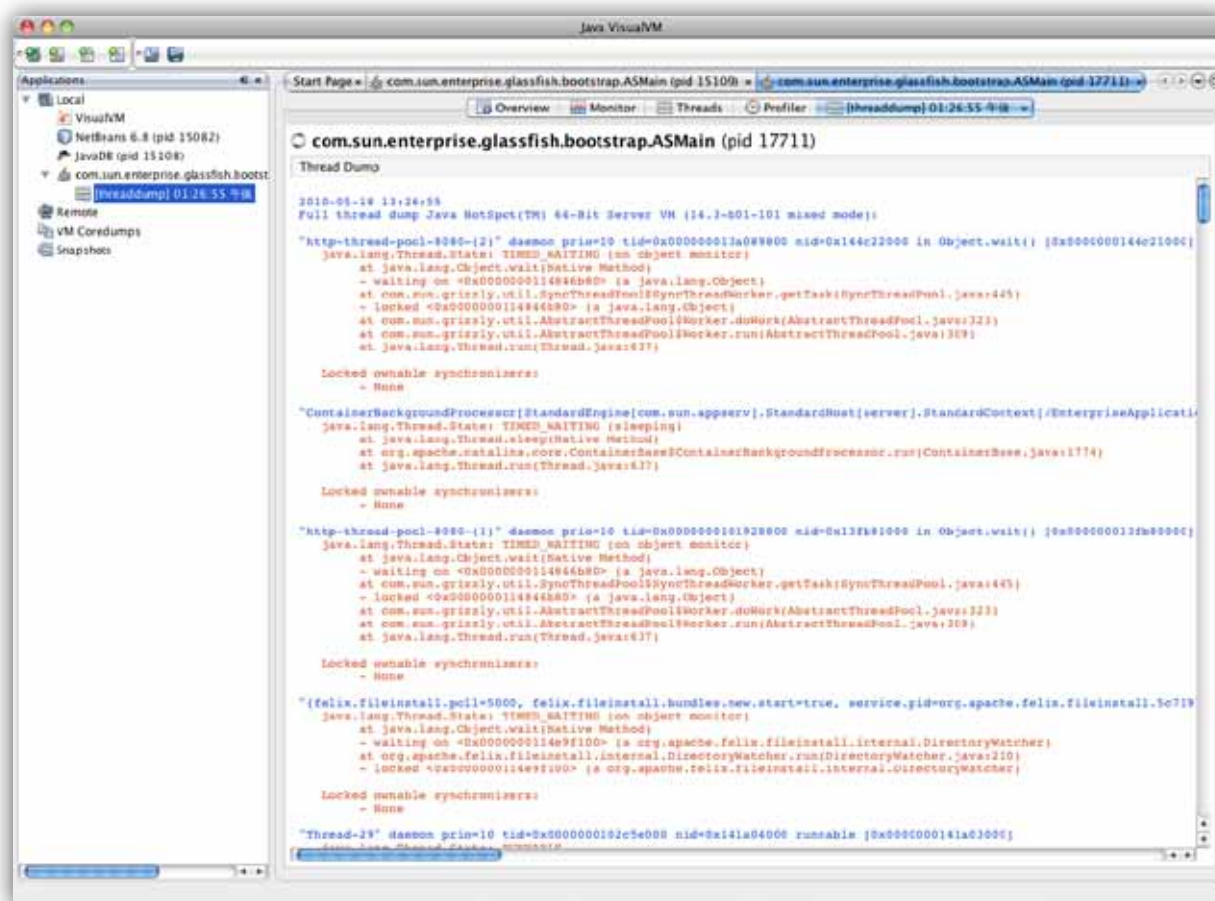
パターンを覚えるのが近道



トラブルシューティングの実践

アプリケーションが停止、クラッシュ

ログやダンプを解析



トラブルシューティングの実践

セキュリティ

日頃の情報収集

APPLICATION CLIENT SECURITY

Application Client Container, which can host first tier client for enterprise applications, contacts the authentication by itself and when communicating with the EJBs, sends the authenticated subject along with the call. In the standard deployment descriptor we can configure the callback handler which collect the user credentials for authentication and all other measures are configured in the vendor specific deployment descriptor.

For example, the following snippet specifies the callback handler to collect user identity information.

```
<callback-handler>  
  org.jboss.seam.security.IdentityCallbackHandler  
</callback-handler>
```

Hot Tip The callback handler element specifies the name of a callback class provided by the application for J2EE authentication. This class must implement the java.security.AuthCallback interface.

```
<application-client>  
  <security-domain>my-app-security-domain</security-domain>  
</application-client>
```

Application Client Container will use the authentication realm specified in the application.xml file to authenticate the users when a request for a container EJB is placed.

Security enforcement in Geronimo ACC

Similar to GlassFish, Geronimo provides some configuration elements in the vendor specific file named `geronimo-application-client.xml`.

Notable configuration elements are `default-subject`, `realm-name` and `callback-handler`. For example to configure the callback handler we can use the following snippet.

```
<callback-handler>  
  org.jboss.seam.security.IdentityCallbackHandler  
</callback-handler>
```

Security enforcement in JBoss ACC

Configuration for the JBoss ACC container are stored in the `jbosseci-client.xml` file. This configuration file provides no further security customization for the application client.

DEFINING SECURITY IN ENTERPRISE APPLICATION LEVEL

As we see in the figure 1, the enterprise application archive (EAR) file can contain multiple modules intended to be deployed into different containers like Web, EJB or ACC. This EAR module has the deployment descriptor of its own which we can use to include the shared deployment plan details in addition to EAR specific declarations.

We can use the application level deployment descriptors to define roles, include the required role mappings and to specify the default security realm of all included modules.

We can use the standard deployment descriptor to define security roles. The syntax is the same as what we used in the `web.xml` and the `ejb-jar.xml`.

Similar to other vendor specific deployment descriptors, we can use the application level descriptor to define the application wide role mapping and also to define the default security realm for all included modules.

The following table shows how we can use different vendor specific deployment descriptors for role mapping and specifying the default security realm and shows what security measures are accessible through the vendor specific enterprise application deployment descriptors:

Application Server	Description
GlassFish <code>sun-application.xml</code>	Role mapping can be done for vendor specific deployment descriptors. Defining the default security realm is as follows: - <code>role</code> , <code>realm</code> - <code>callback-handler</code> The element is an immediate child of the <code>application</code> element.
Geronimo <code>geronimo-application.xml</code>	Role mapping can be done in other Geronimo specific deployment descriptors. Specifying the default security realm is as follows: - <code>subject</code> - <code>realm-name</code> - <code>callback-handler</code> The element is an immediate child of the <code>application</code> element.
JBoss <code>jbosseci-client.xml</code>	Role mapping and specifying the security realm of the same is as follows: - <code>role</code> - <code>realm-name</code> - <code>callback-handler</code> The element is an immediate child of the <code>application</code> element.

SECURING JAVA EE WEB SERVICES

In the Java EE specification, Web services can be deployed as a part of a Web module or an Enterprise module.

In the web module we can protect the Web service endpoint the same way we protect any other resource. We can define a resource collection and enforce access management and authentication on it. The most common form of protecting a Web service is using the HTTP Basic or HTTP Digest authentication.

For example if we use the HTTP basic authentication and our Web service client uses the Dispatch Client API to access the web service we can use a snippet like the following one to include the username and password with the right access role to invoke a Web service.

```
<web-service-client>  
  <username>my-app-user</username>  
  <password>my-app-pass</password>  
</web-service-client>
```

The user and the password should be valid in the configured realm of Web application and should have access right in the endpoint URL.

Hot Tip Another way of authenticating the client is to pass in HTTP header, using the `Authorization` header which provides more flexibility to web services. For more information about the authentication class check `http://java.sun.com/docs/tutorial/security/defining-security.html`.

Web Services Security in EJB Modules

We can create a `WebService` bean as a `Web Service` and therefore we can use all security annotations like `@RoleAllowed`, `@PermitAll` and their corresponding deployment descriptor elements to define its security plan.

But the authentication enforcement of the Web Services is vendor specific and each vendor uses its own method to define the authentication, security realm and so on.

Web Services Authentication in GlassFish

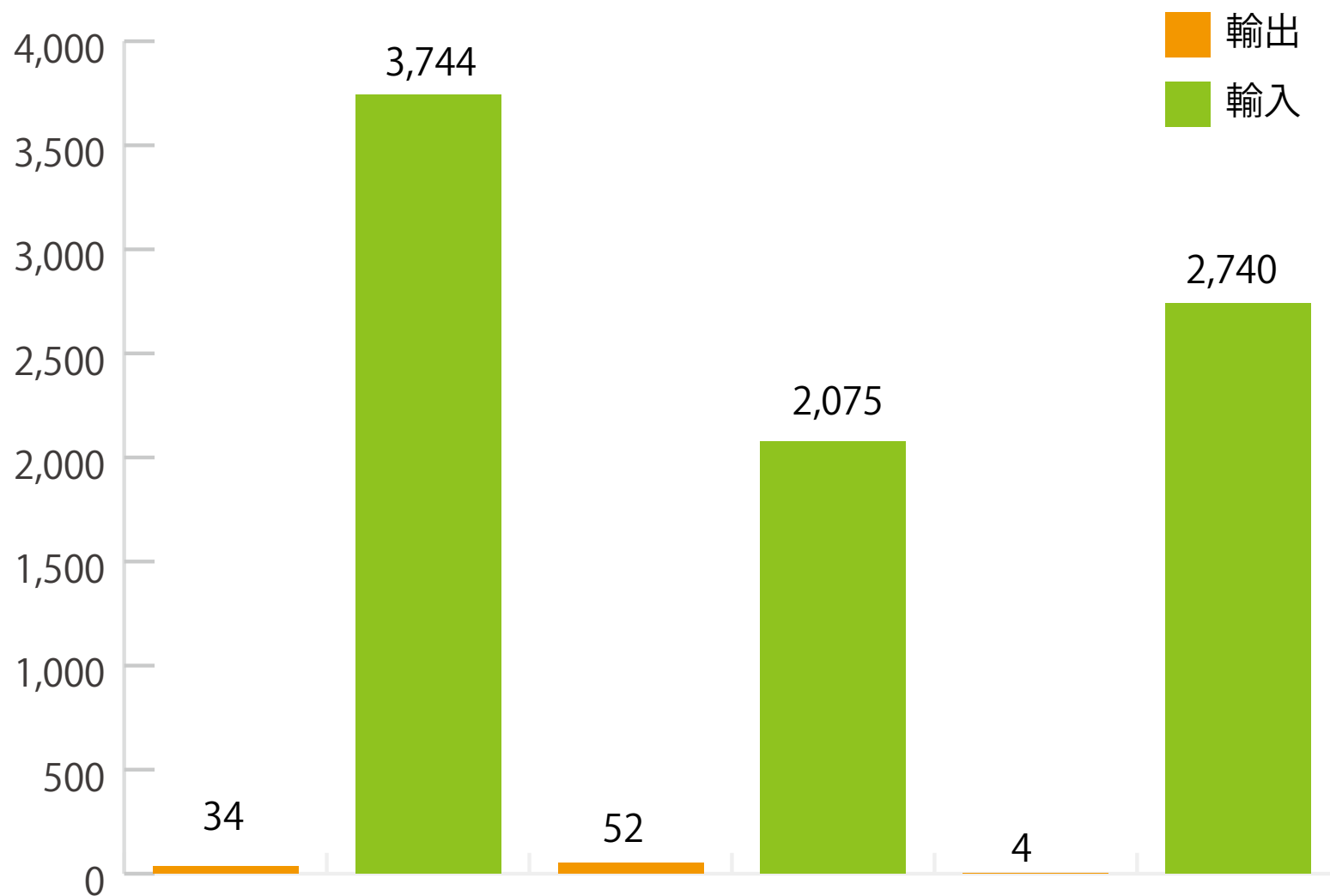
For GlassFish we should specify the authentication method and the security realm in the `sun-ejb-jar.xml`. For example, to specify HTTP Basic authentication method and a realm named `my_realm` as the security realm for a web service called `Echo`, we will have something similar to the following snippet.

<http://refcardz.dzone.com/refcardz/getting-started-java-ee>

パフォーマンスチューニングや
トラブルシューティングに対する

これから
必要な
取り組み。

これから必要な取り組み



JEITA ソフトウェア輸出入統計調査, 2000年度より作成
<http://it.jeita.or.jp/statistics/software/2000/>

これから必要な取り組み

「〇〇だからできない」を無くしていく

科学的根拠の無いまま、ツールやライブラリの導入が禁止される一方で、他国のエンジニアはそれらを使い、より安く、より短期間で仕事をこなしている。

これから必要な取り組み

本番環境も含め、動的にツールを接続する

ダンプ、ログだけでは再現しづらいトラブルはなかなか解決できない。高信頼性のツールを使えば、問題箇所のみ限定して操作を加えることも可能で、圧倒的に効率が良い。



ご清聴ありがとうございました。

@watermint

watermint.org