

Interstage / ENdoSnipe 動作検証結果報告書(Solaris)

平成21年6月5日

株式会社ヒューマンクレスト

品質コンサルティング部

第1.0版

動作検証結果報告書 目次

■ 1. 本プロジェクトの目的及び方針	・・・P 2
■ 2. 検証環境	・・・P 3
■ 3. 検証内容	・・・P 5
■ 4. 検証結果	・・・P 6
◆ メモリリーク	・・・P 7
◆ マルチプロセス	・・・P 11
◆ 負荷試験	・・・P 14
◆ ENdoSnipe動作の有無によるリソース使用量比較	・・・P 18
■ 5. 「見える化」でシステムの問題解決を効率化する	・・・P 19
■ 6. 問い合わせ先	・・・P 20
■ 別紙 検証内容	・・・P 21

1. 本プロジェクトの目的及び方針

本プロジェクトは以下の目的及び基本方針に基づいて実施する。

■ 1. 本プロジェクト発足の経緯

- ◆ ソフトウェア事業本部・サポート技術統括本部様における、ENdoSnipe動作確認作業において、高評価をいただき、Interstageパートナー登録の運びとなった。
- ◆ Interstageパートナー登録にあたって、ENdoSnipeの機能が正しく動作するかを再度確認するために、今回のプロジェクトが発足した。

■ 2. 本プロジェクトの実施目的

- ◆ Interstage Application Server上で動作するJavaアプリケーションに対し、ENdoSnipeの機能が正しく動作するかを確認する。

■ 3. 基本方針

- ◆ 上記実施目的を達成するために、弊社にて準備したJavaアプリケーションを使用し、動作検証を実施する。

■ 4. 本プロジェクトの達成目標

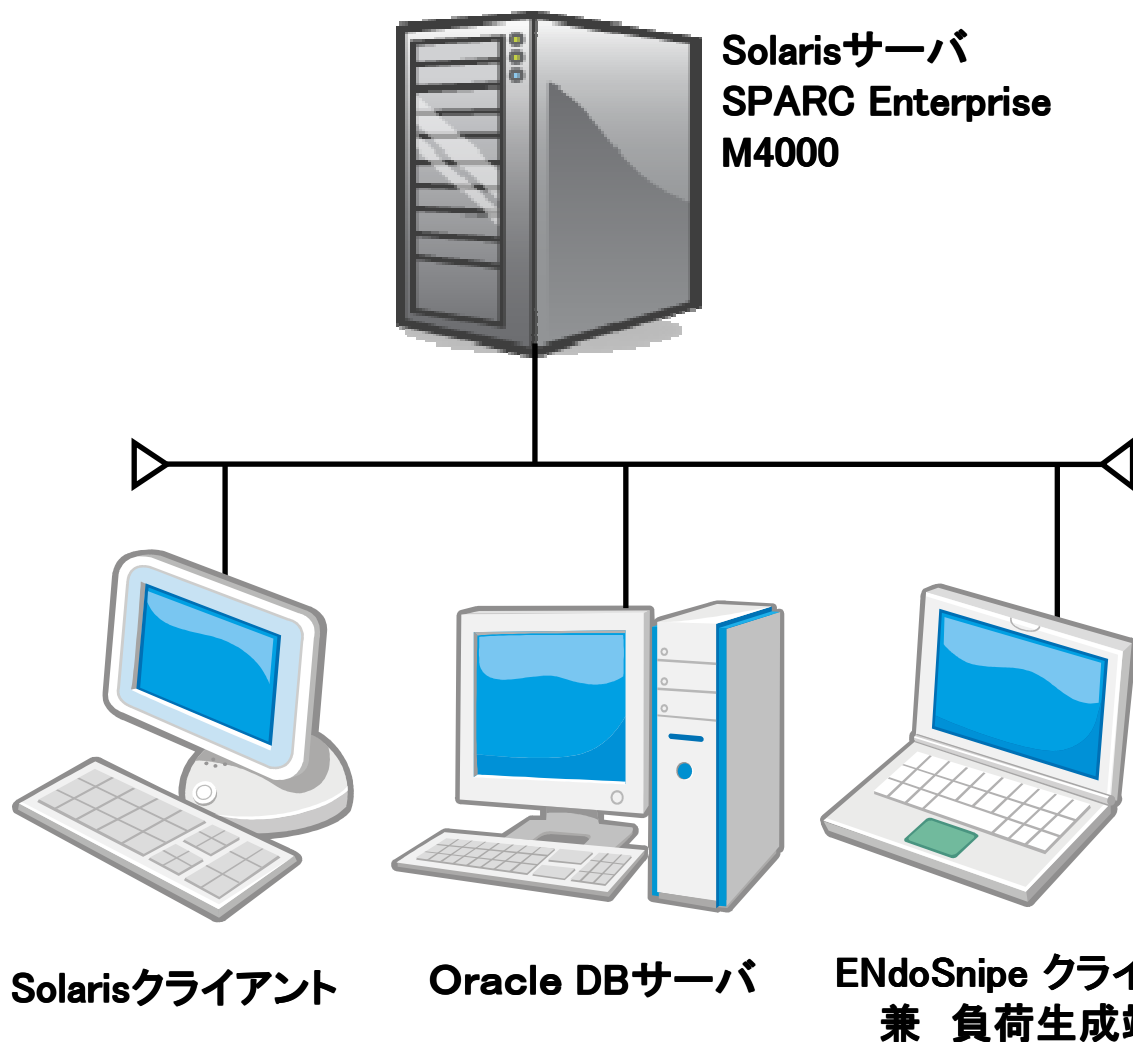
- ◆ ENdoSnipeの機能が正しく動作することを明らかにする。

■ 5. 検証のポイント

- ◆ Interstage Application Server上で動作するJavaアプリケーションに対し、ENdoSnipeの機能が正しく動作すること。
- ◆ マルチコンテナ(マルチプロセス)で動作するJavaアプリケーションに対し、ENdoSnipeの機能が正しく動作すること。

2. 検証環境 (1/2)

■ 1. ハードウェア構成



マシン	仕様	
Solarisサーバ SPARC Enterprise M4000	CPU	SPARC64 VI: 2.15GHz(2core/5MB) × 4CPU
	MEM	16GB
	HDD	73GB/10Krpm × 2
Solarisクライアント SunBlade150	CPU	UltraSPARC-IIe 650MHz × 1
	MEM	1GB
	HDD	80GB × 1
OracleDBサーバ FMV830NA	CPU	Pentium 4 3.2GHz × 1
	MEM	2GB
	HDD	80GB × 1(EIDE)
ENdoSnipeクライアント ThinkPad X31	CPU	Pentium M 1.7GHz × 1
	MEM	2GB
	HDD	70GB × 1

2. 検証環境 (2/2)

■ 2. ソフトウェア構成

マシン名	機種	OS	Application	DB	Java Ver
評価用サーバ APサーバ	SPARC Enterprise M4000	Solaris 10 OS	・Interstage Application Server Enterprise Edition V9.1.0 ・ENdoSnipeエージェント(Javelin) ・検証プログラム(Javaアプリ)	—	1.5.0_13 (Fujitsu)
DBサーバ	FMV830NA	Windows XP	—	Oracle Database 11g (11.1.0)	—
Solarisクライアント	SunBlade150	Solaris 9 OS	—	—	—
評価用クライアント ENdoSnipeクライアント 兼 負荷生成端末	ThinkPad X31	Windows XP SP3	・Eclipse 3.4.2 ・ENdoSnipeプラグイン ・Jmeter(負荷生成ツール)	—	1.6.0_13

※Java verについて Interstageサーバは富士通JDKのバージョンを、ENdoSnipeクライアントは、SunJDKのバージョンを記載

※DBサーバについて 通常 Windows XP で Oracle を運用することは無いと思いますが、今回は Oracle に性能を求めていない為サーバ機ではなく、Windows XP 搭載のPCにて動作検証を行いました。

3. 検証内容

■以下の観点で検証を実施

機能	観点
BottleneckEye	アプリケーションの実行状況を、クラス図表示、グラフ表示できること
	メソッドの呼び出し回数、平均処理時間などの統計情報を参照できること
PerformanceDoctor	メモリリークの発生を検知できること
	DatabaseのSQLを解析できること
	Databaseの実行計画を解析できること
	スレッドのCPU占有、スリープを検知できること
ArrowVision	アプリケーションの処理の流れをシーケンスダイアグラムとして表示できること
	Javaなどの複数ログの情報が1つのシーケンス図で表示できること
	ボトルネック解析として、時間が掛かった処理部分を表示できること

その他の観点として、以下のものを実施。

- ・マルチプロセスで動作しているアプリの、プロセス毎の情報を取得できること
- ・Serverに負荷が掛かった状態でも正常に動作ログが出力されること
- ・ENdoSnipeの動作有無によるリソース差異を確認する

4. 検証結果 (1/13)

Interstage Application Server上で動作するJavaアプリケーションに対し、ENdoSnipeの機能が正しく動作することが確認できた。(※1)

結果として、代表的なものを以降に示す。

- ◆ メモリリーク
メモリリークの発生を検知できること
- ◆ マルチプロセス
マルチプロセスで動作しているJavaアプリケーションに対して、プロセス毎の情報を取得できること
- ◆ 負荷試験
Serverに負荷が掛かった状態でも正常に動作ログが出力されること
- ◆ ENdoSnipeの動作有無によるリソース使用量比較
ENdoSnipeの動作有無によるリソース差異を確認する

(※1)ただし、以下については確認できていない。

- ・スリープ回数、時間は測定不能のため、スリープ時間の診断はできない。(ENdoSnipe次期バージョン(Ver4.12009/09リリース予定)にて対応予定)
ただし、アイドル時間の診断ルールを用いて、CPUを使用していない時間を診断する事で代用が可能。

4. 検証結果 (2/13)

●メモリーリーク

▪BottleneckEye画面

jdbc:oracle:thin:@(DESCRIPTION=(ADDRESS=(PF...
SELECT NULL AS table_cat, towner AS tab
SELECT DEPT deptno, DEPT dname, DEPT loc, DE
SELECT NULL AS table_cat, c_owner AS tab
select count(*) from emp e left outer join dept d o
select dname from dept where deptno = ?(850:292
select empno,ename,job,mer,hiredate, sal,comm,e
select empno,ename,job,mer,hiredate, sal,com
select sysdate from dual(1171:3324)

examples.jsf
logic.impl.EmployeeLogicImpl(78010:6540)
util.RequestDumpFilter(119255:160)
sample.MultiAccessThread(11255:9029:500)
dao.DepartmentDtoDao(8121:2442)
dao.EmployeeDtoDao(7369:3394)
util.RequestDumpUtil(6187:0:742)
logic.impl.EmployeeLogicImpl\$Stopper(6029:6015:000)
action.impl.EmployeeConfirmInitActionImpl(5264:0:638)
action.impl.EmployeeEditInitActionImpl(5093:1:123)
action.impl.EmployeeEditActionImpl(5030:0:274)

/Servlet_JSP_APP
/dbac01(48:48:000)
/dbac02(3257:3257:000)
/style.org.css(1:0:000)

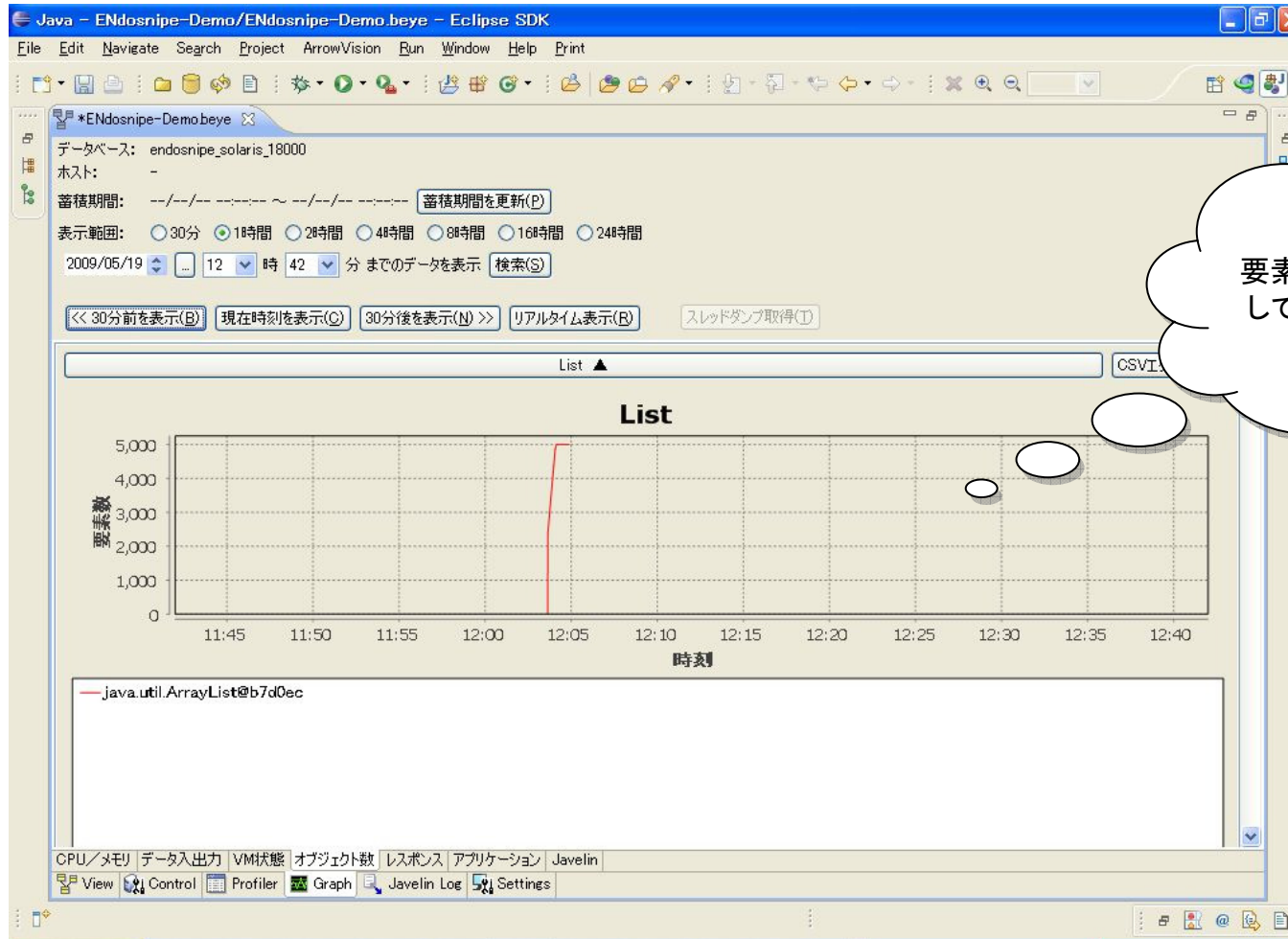
メモリリーク検知時、関連するクラス部分が点滅

Time	Node	Name	Duration
2009/05/19 17:50:14.903	EmployeeEditInitActionIm...	initialize	11037
2009/05/19 17:50:14.903	EmployeeLogicImpl	getEmployeeDto	11029
2009/05/19 17:50:14.933	EmployeeLogicImpl	proceed	11019
2009/05/19 17:50:14.933	EmployeeLogicImpl	proceed	11009
2009/05/19 17:50:14.933	EmployeeLogicImpl	getEmployeeDto	11009
2009/05/19 17:50:14.933	EmployeeLogicImpl	sleep	11001

4. 検証結果 (3/13)

●メモリーリーク

▪BottleneckEye画面



要素のインスタンスが存在していることがわかります

4. 検証結果 (4/13)

●メモリーリーク

▪ ArrowVision画面

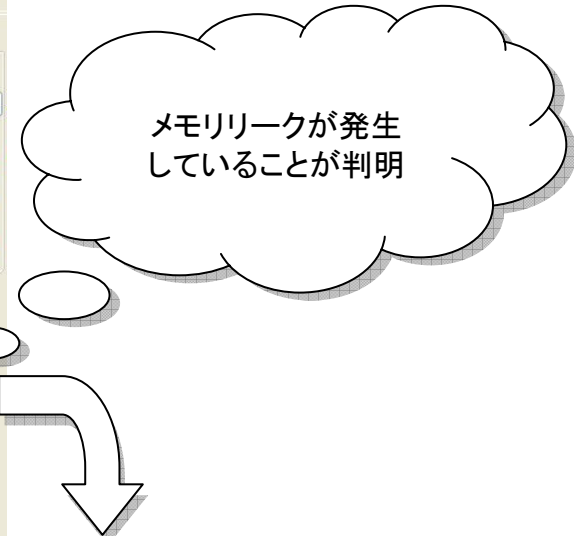
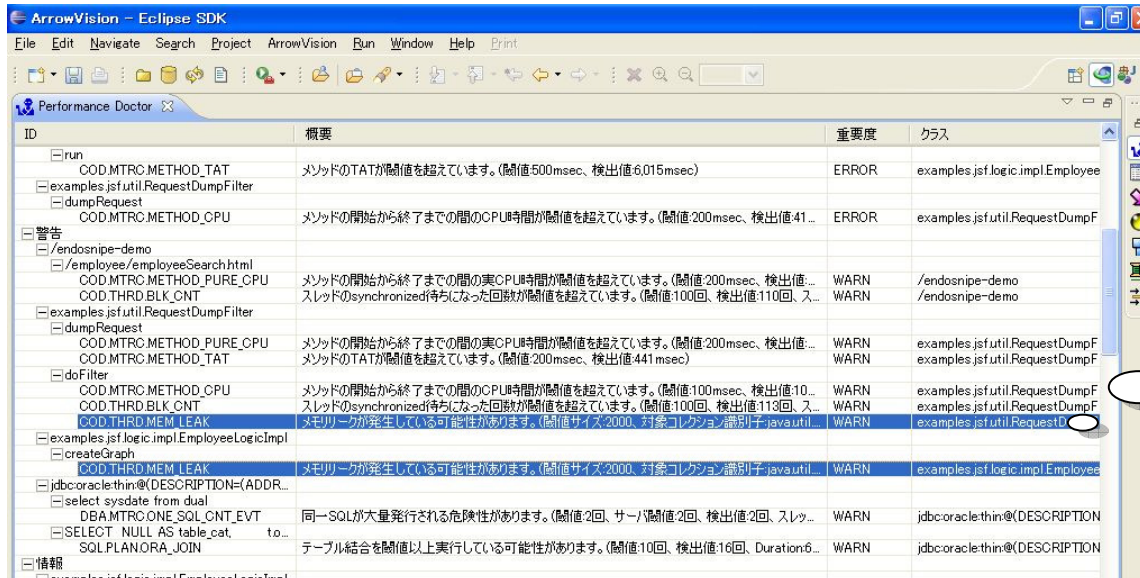
The screenshot shows the ArrowVision interface within Eclipse SDK. The main window displays a sequence diagram with several lifelines: root, examples.jsf.util.RequestDumpFilter, examples.jsf.dto.EmployeeSearchDto, /endosnipe-demo, examples.jsf.action.impl.EmployeeListActionImpl, and examples.jsf.action.impl.EmployeeEditInitActionImpl. The diagram shows a call from root to RequestDumpFilter, followed by a call to EmployeeSearchDto, and then a call to EmployeeListActionImpl. A yellow warning icon labeled 'warn: Leak Detected' is positioned on the RequestDumpFilter lifeline. A callout bubble points to this event with the text: "LeakDetectedイベントが発生していることから、メモリーリークが発生していることが判明". The bottom panel shows the event details in the Journal:

```
Event :2009/05/19 12:08:35.250,"LeakDetected","doFilter","examples.jsf.util.RequestDumpFilter","WARN","http-9003-Processor15@59 (org.apache.tomcat.util.threads.ThreadWithAttributes@1b4e557)"
<<Javelin.EventInfo_START>>
javelin.leak.identifier = java.util.HashMap@1ce154c
javelin.leak.threshold = 2000
javelin.leak.stackTrace =   at java.util.HashMap.put (HashMap.java)
                           at com.fujitsu.interstage.j2ee.ijserver.loader.ApplicationClassLoader.findResourceInternal (ApplicationClassLoader.java:986)
                           at com.fujitsu.interstage.j2ee.ijserver.loader.ApplicationClassLoader.findResource (ApplicationClassLoader.java:1054)
```

4. 検証結果 (5/13)

●メモリーク

▪PerformanceDoctor画面



ID	概要	重要度
COD.THRD.MEM_LEAK	メモリークが発生している可能性があります。 (閾値サイズ:2000、対象コレクション識別子:java.util.HashMap@1ce154c、オブジェクトサイズ:nullbytes、スレッド:http-9003-Processor15@59(org.apache.tomcat.util.threads.ThreadWithAttributes@1b4e557))	WARN
COD.THRD.MEM_LEAK	メモリークが発生している可能性があります。 (閾値サイズ:2000、対象コレクション識別子:java.util.HashMap@5b13dd、オブジェクトサイズ:nullbytes、スレッド:http-9001-Processor15@55(org.apache.tomcat.util.threads.ThreadWithAttributes@1fc1696))	WARN

4. 検証結果 (6/13)

- マルチテナ (マルチプロセス)
- BottleneckEye画面

↓ プロセス2側の画面
(10.20.20.106:18001)

The screenshot displays two Eclipse IDE windows side-by-side, representing different processes. The left window shows a project named 'ENdoSnipe-Demo' with a file explorer on the left and a code editor showing 'examples.jsf'. The right window shows a similar project but with a 'jdbc:' connection string in the code editor. Both windows have a 'Problems' view at the bottom showing various errors. A central text box with arrows pointing to the error logs in both windows contains the text: 'それぞれのプロセスで問題が発生したことがわかります' (It is clear that problems occurred in each process). At the bottom right, a 'Journal' view displays a table of execution events.

Time	Node	Name	Duration
2009/05/19 12:19:39.962	EmployeeLogicImpl	proceed	11014
2009/05/19 12:19:39.962	EmployeeLogicImpl	proceed	11014
2009/05/19 12:19:39.962	EmployeeLogicImpl	getEmployeeDto	11014
2009/05/19 12:19:39.962	EmployeeLogicImpl	sleep	11006
2009/05/19 12:23:16.543	RequestDumpFilter	doFilter	418
2009/05/19 12:23:16.543	/endosnipe-demo	/employee/employeeSearch.html	415

↑ プロセス1側の画面
(10.20.20.106:18000)

4. 検証結果 (7/13)

●マルチコンテナ(マルチプロセス)

▪ ArrowVision画面

↓プロセス2側の画面
(10.20.20.106:18001)

それぞれのプロセスで
問題が発生したことがわかります

↑プロセス1側の画面
(10.20.20.106:18000)

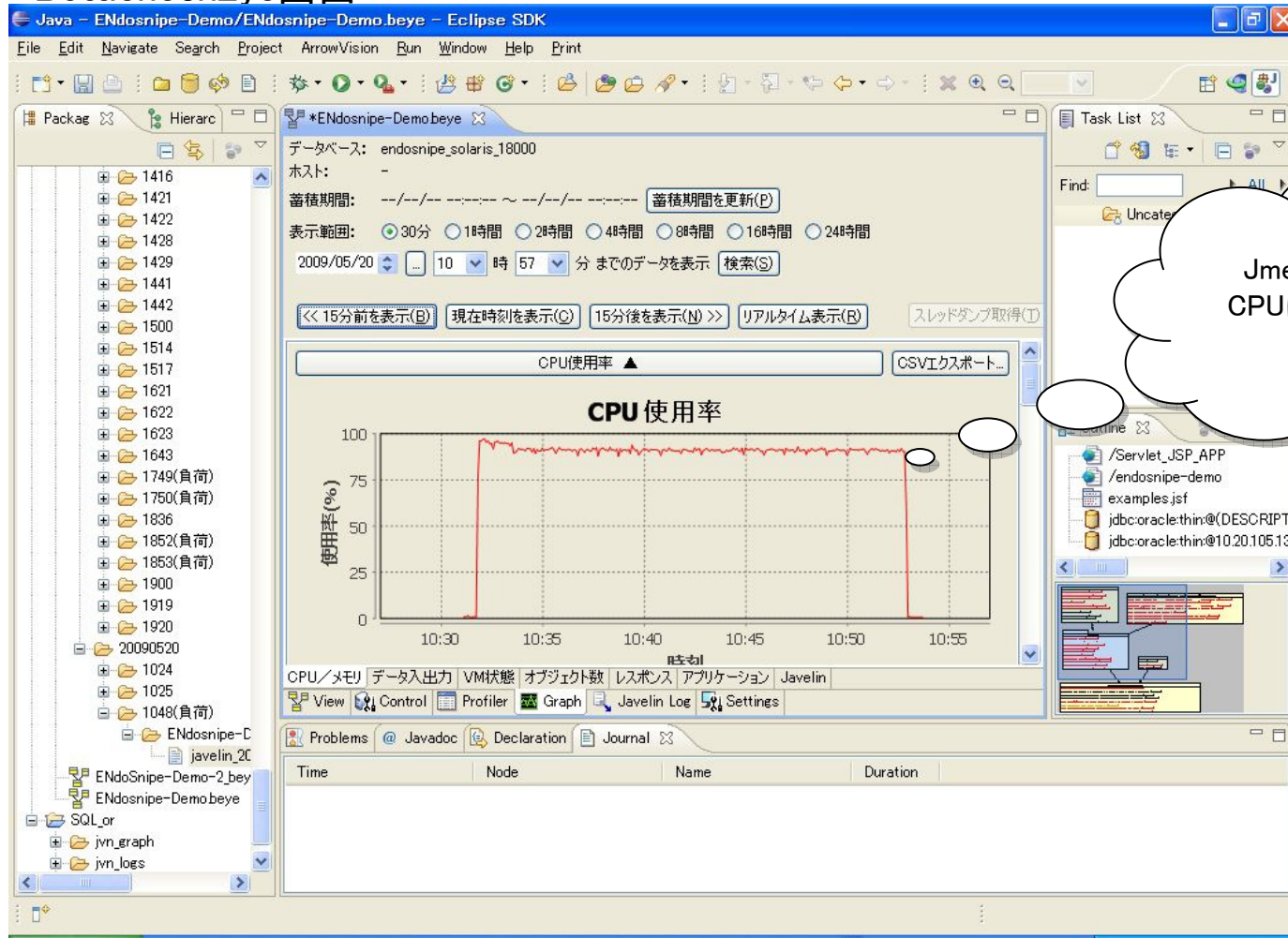
```
Call : 2009/05/19  
12:19:25.069, "getEmployeeDto", "examples.jsf.logic.impl.EmployeeLogicImpl",  
"unknown", "http-9001-Processor13@53(org.apache.tomcat.util.threads.Thre  
<<Javelin.Args_START>>  
args[0] = 7566  
<<Javelin.Args_END>>  
<<Javelin.JMXInfo_START>>  
thread.currentThreadCpuTime.delta = 7.1933
```

```
Event : 2009/05/19 12:23:12.343, "LeakDetected", "proceed", "examples.jsf.dao.EmployeeDtoDao", "WARN", "http-9003-Processor@052  
(org.apache.tomcat.util.threads.ThreadWithAttributes@13da08)"  
<<Javelin.EventInfo_START>>  
javelin.leak.identifier = java.util.HashMap@1ce154c  
javelin.leak.threshold = 2000  
javelin.leak.stackTrace = at java.util.HashMap.put (HashMap.java)  
at com.fujitsu.interstage.j2ee.ijserver.loader.ApplicationClassLoader.findResourceInternal (ApplicationClassLoader.java:986)  
at com.fujitsu.interstage.j2ee.ijserver.loader.ApplicationClassLoader.findResource (ApplicationClassLoader.java:1054)
```


4. 検証結果 (9/13)

● 負荷試験

▪ BottleneckEye画面



Jmeterを使用して
CPUに負荷を掛けて
いる状態

4. 検証結果 (10/13)

● 負荷試験

▪ BottleneckEye画面

The screenshot shows the BottleneckEye tool interface. The main window displays a performance profile of a Java application. The profile is organized into a hierarchy of components, with the most time-consuming components highlighted in red. The components include:

- jdbc:oracle:thin@... (DESCRIPTION=(ADDRESS=(P...))
- SELECT NULL AS table_cat, towner AS ta...
- SELECT DEPT,deptno,DEPT,dname,DEPT,loc,...
- SELECT NULL AS table_cat, cowner AS tal...
- select count(*) from emp e left outer join dept d...
- select dname from dept where deptno = '(850:29...
- select empno,ename,job,mgr,hiredate, sal,comm,e...
- select empno,ename,job,mgr,hiredate, sal,comm,e...
- select sysdate from dual(1306:8399)
- examples.jsf
- logic.impl.EmployeeLogicImpl(780105.582)
- util.RequestDumpFilter(119255.160)
- sample.MultiAccessThread(11255:9029.500)
- dao.DepartmentDaoImpl(81219.633)
- dao.EmployeeDtoDao(7369:3417)
- util.RequestDumpUtil(61870.742)
- logic.impl.EmployeeLogicImpl\$Stopper(6029:6015.000)
- action.impl.EmployeeConfirmInitActionImpl(5264:0.608)
- action.impl.EmployeeEditInitActionImpl(5050:1.164)
- action.impl.EmployeeEditActionImpl(5030:0.263)
- /Servlet_JSP_APP
- /dbac01(48:48.000)
- /dbac02(3257:3257.000)
- /style.org.css(1:0.000)

A callout box on the right side of the screenshot contains the following text:

負荷が掛かっている状態で
FILE I/Oを起こしています

負荷が掛かっている状態でも、
FILE I/Oが閾値を越えて
発生したことが取得できて
います

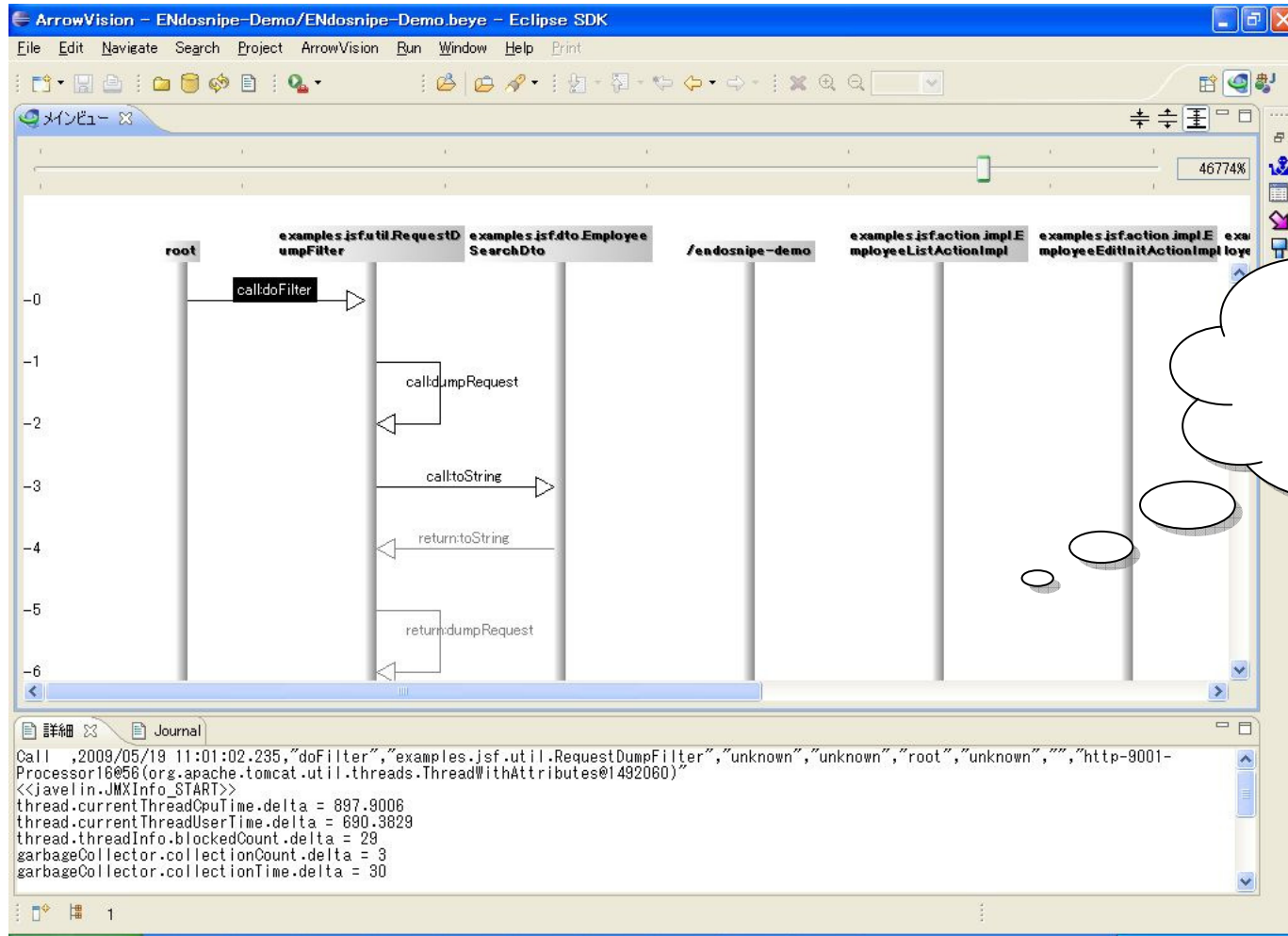
The bottom of the screenshot shows a table with the following data:

Time	Node	Name	Duration
2009/05/19 18:53:09.511	EmployeeEditInitActionIm...	initialize	11018
2009/05/19 18:53:09.511	EmployeeLogicImpl	getEmployeeDto	11014
2009/05/19 18:53:09.511	EmployeeLogicImpl	proceed	11014
2009/05/19 18:53:09.511	EmployeeLogicImpl	proceed	11014
2009/05/19 18:53:09.511	EmployeeLogicImpl	getEmployeeDto	11014
2009/05/19 18:53:09.511	EmployeeLogicImpl	sleep	11010

4. 検証結果 (11/13)

● 負荷試験

▪ ArrowVision画面

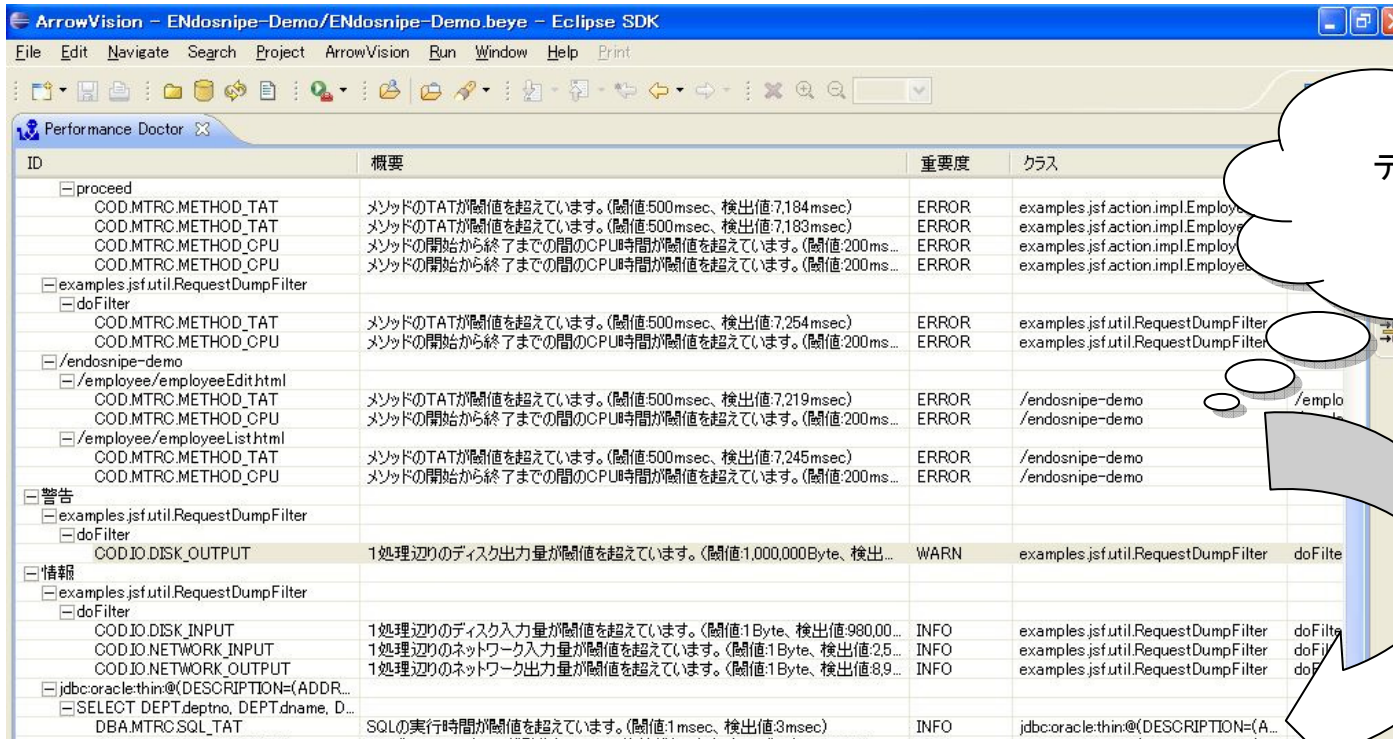


ディスクI/Oの情報が
表示されています

4. 検証結果 (12/13)

● 負荷試験

・ PerformanceDoctor画面



ディスクI/Oの情報が表示されています

ID	概要	重要度
COD.IO.DISK_OUTPUT	1 処理辺りのディスク出力量が閾値を超えています。(閾値:1,000,000Byte、検出値:2,280,383Byte)	WARN
COD.IO.DISK_INPUT	1 処理辺りのディスク入力量が閾値を超えています。(閾値:1Byte、検出値:980,000Byte)	INFO

4. 検証結果 (13/13)

●ENdoSnipeの動作有無によるリソース差異

	物理メモリ空き容量	CPU使用率
ENdoSnipe適用下	14,454,328 [Kbytes]	1.0%
ENdoSnipe非適用下	14,479,640 [KBytes]	1.0%

※ENdoSnipe適用下 : Javelinがログを取得する状態で、「数秒間のスリープ発生」をさせる

※ENdoSnipe非適用下 : Javelinがログを取得しない状態で、「数秒間のスリープ発生」をさせる

物理メモリ空き容量は多少差があるものの、CPU使用率はともに1%程度であることから、ENdoSnipeの動作有無によるリソース使用量に大きな違いは見られない。

よって、アプリケーション実行時に致命的なパフォーマンス低下は発生しないと考えられる。

5. 「見える化」でシステムの問題解決を効率化する

解析

ArrowVision

システムの開発・保守効率を向上させたい！

▶ ArrowVision とは

ArrowVisionは、アプリケーションのソースコードを一切変更することなく、その動作をシーケンスダイアグラムによって「見える化」します。

- ・システムの処理の流れを、自動的にシーケンスダイアグラムにします。
- ・Javaシステム、ネットワークなど、複数のログを1つのシーケンス図にまとめて表示します。
- ・時間の掛かる処理がすぐにわかるため、大量のログを調査する必要がありません。

診断

PerformanceDoctor

手早く品質を診断したい！

▶ PerformanceDoctor とは

PerformanceDoctorは、取得したログからシステムに潜む性能問題の芽を「見える化」します。

- ・性能問題の原因となる処理を自動的に見つけます。
- ・スレッドのCPU占有や頻繁なスリープなども見つけます。

監視

BottleneckEye

予期せず発生した障害をすぐに解決したい！

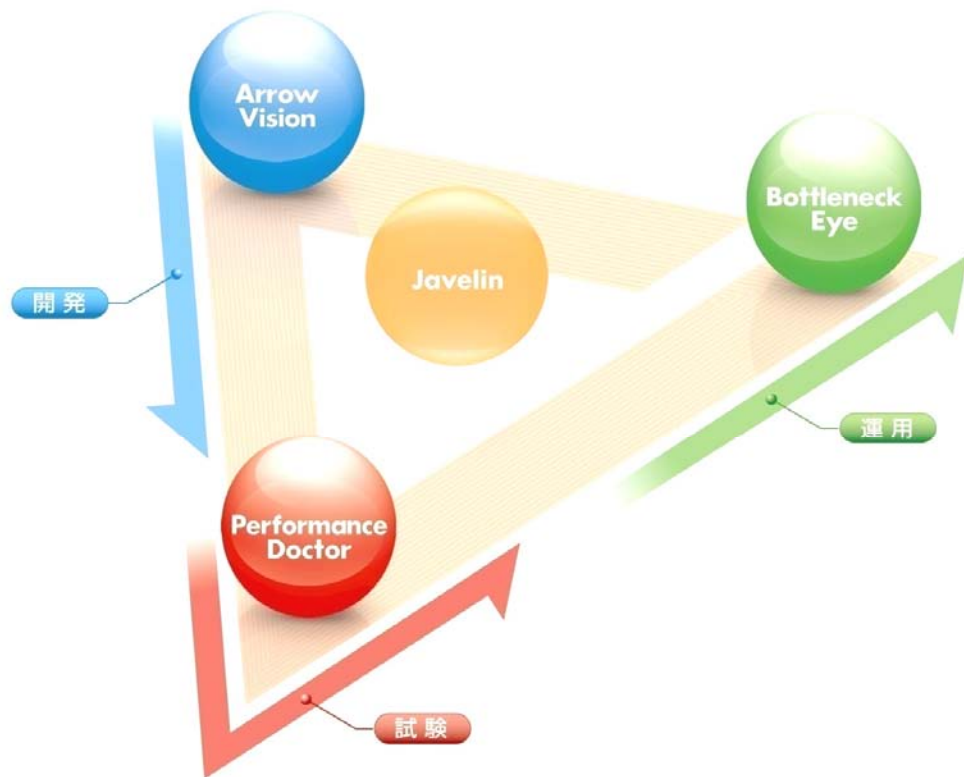
▶ BottleneckEye とは

BottleneckEyeは、運用中のシステムを監視し、「アプリケーションの構造」と「システムのボトルネック」を「見える化」します。

- ・異常や性能問題の発生を即座に知らせます。
- ・アプリケーションの構造を自動的にクラス図化します。
- ・アプリケーション中のメモリリークを発見できます。

ENdoSnipeがソフトウェアの解析／診断／監視に「革命的効率化」をもたらします

6. 問い合わせ先



変革と研鑽

Human Crest

株式会社ヒューマンクレスト

TEL:045-226-0714

URL:<http://www.humancrest.co.jp>

MAIL: endosnipe@humancrest.co.jp



エスエムジー株式会社

TEL:045-476-3171

URL:<http://www.smg.co.jp>

MAIL: endosnipe@smg.co.jp

別紙. 検証内容(1/3)

◆ BottleneckEye: システムの構造をクラス図で「見える化」

アプリケーションの実行状況を、 クラス図表示、グラフ表示できること	CPU使用率
	物理メモリ使用量
	仮想マシンメモリ使用量
	ヒープメモリ
	非ヒープメモリ
	GC停止時間
	スレッド数
	スワップ領域使用量
	ネットワーク経由でのデータ受信量
	ネットワーク経由でのデータ送信量
	ファイル出力量
ファイル入力量	
メソッドの呼び出し回数、平均処理時間などの統計情報を参照できること	

別紙. 検証内容(2/3)

◆ PerformanceDoctor: 性能問題の原因を「見える化」

メモリーリークの発生を検知できること	
DataBaseのSQLを解析できること	SQL実行回数と時間の閾値チェック
	同一SQL実行回数の閾値チェック
	SQL中のUNION句登場回数の閾値チェック
	SQL中のOR句登場回数の閾値チェック
Databaseの実行計画を解析できること	SQL実行計画中のテーブル・フルスキャン
	SQL実行計画中のコスト閾値チェック
	SQL中の結合利用回数の閾値チェック
スレッドのCPU占有、スリープを検知できること	メソッド処理時間の閾値チェック
	同一メソッド呼び出し回数の閾値チェック
	Threadの平均CPU使用率の閾値チェック
	Threadスリープ回数と時間の閾値チェック
	Thread同期待ち回数と時間の閾値チェック
	GC発生回数と時間の閾値チェック
	通信量(バイトサイズ)の閾値チェック
	ファイルアクセス量(バイトサイズ)の閾値チェック
	ハンドリングされない例外の発生のチェック

別紙. 検証内容(3/3)

◆ ArrowVision: システムの動作をシーケンス図で「見える化」

アプリケーションの処理の流れをシーケンスダイアグラムとして表示できること
Javaなどの複数ログの情報が1つのシーケンス図で表示できること
ボトルネック解析として、時間が掛かった処理部分を表示できること

その他の観点として、以下のものを実施。

- ・マルチプロセスで動作しているアプリの、プロセス毎の情報を取得できること
- ・Serverに負荷が掛かった状態でも正常に動作ログが出力されること
- ・ENdoSnipeの動作有無によるリソース差異を確認する