

Java应用程序故障解析
支援工具
Excat for Java Version 3.0



2009年12月17日



株式会社アイ・プライド



针对Java 应用程序故障原因进行解析 获取并表示故障信息的工具



发生故障时，及时取得应用程序等执行状态，并将其可视化

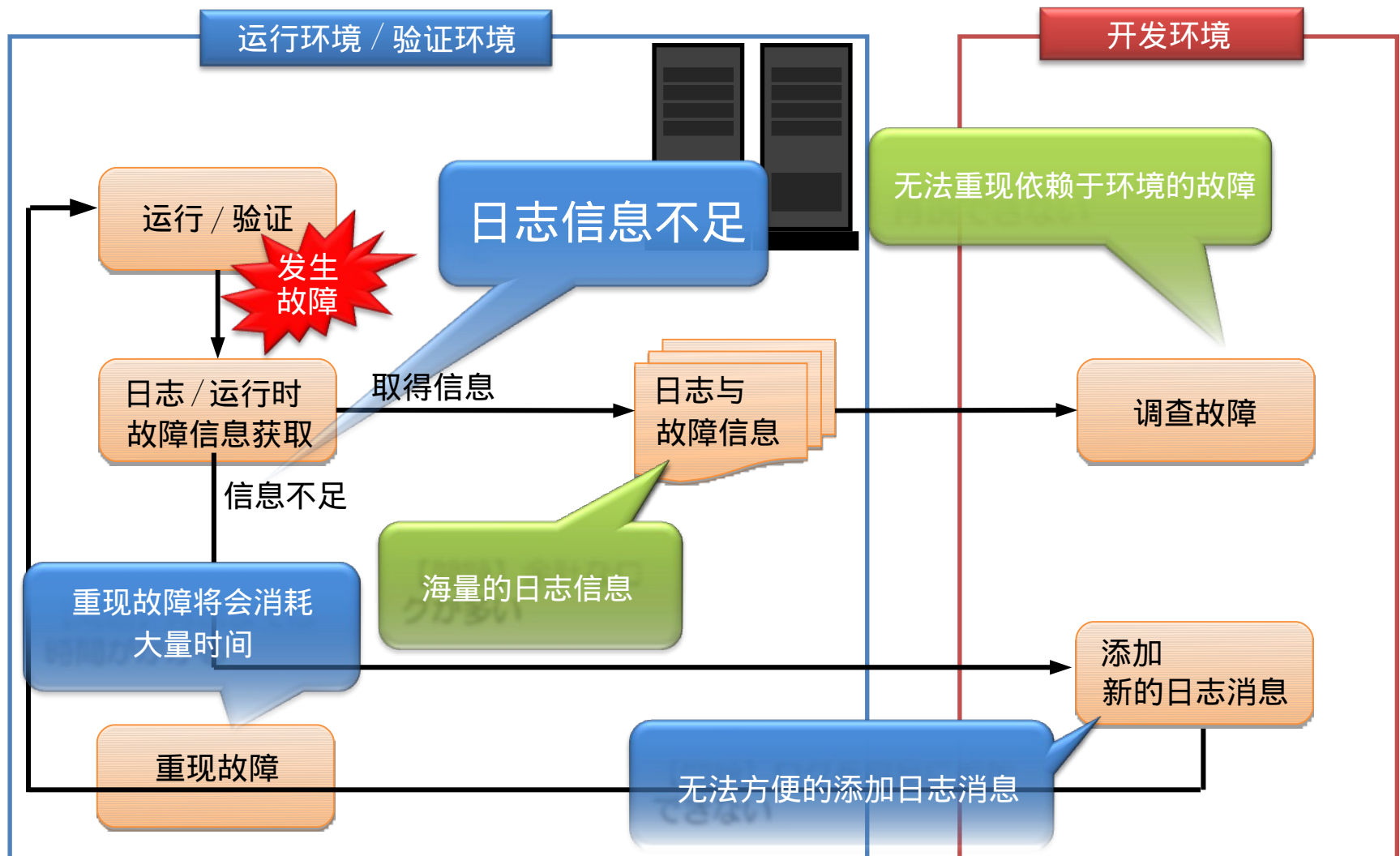


大幅度提高结合・综合测试发生的故障调查的效率



使得在实际运行环境中的故障调查变得更迅速

排查故障的各种问题



排查故障的各种问题



日志信息不足

- 日志设计不完善
- 考虑到运行效率而受到限制的日志输出优先级

无法方便的添加日志消息

- 运行或验证环境中无法方便的添加日志信息

环境依赖

- 依赖于数据
- 依赖于设定
- 依赖于系统结构
- 依赖于发生时点

海量的日志信息

难以重现故障

难以获取有用信息

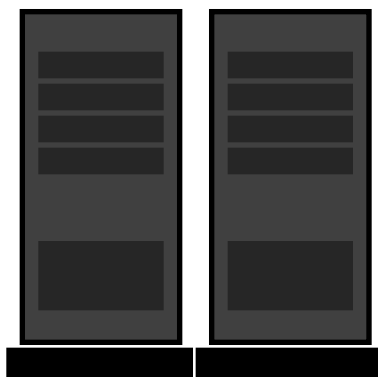
无法及时的从服务器端获取相关信息

难以高效的排查故障发生的原因

弊社の解决方案



在故障发生时，通过及时的获取故障信息，提高排查故障的效率。



```
Thread
  java.lang.Thread.run
  com.evermind.util.ReleasableResourcePooledExecutor$MyWorker.run
  oracle.oc4j.network.ServerSocketReadHandler$SafeRunnable.run
  com.evermind.server.http.HttpRequestHandler.run
  com.evermind.server.http.HttpRequestHandler.serveOneRequest
  com.evermind.server.http.HttpRequestHandler.processRequest
  com.evermind.server.http.HttpRequestHandler.doProcessRequest
  this=com.evermind.server.http.HttpRequestHandler
  thread=com.evermind.server.ApplicationServerThread
  request=com.evermind.server.http.EvermindHttpServletRequest
  site=com.evermind.server.http.HttpSite
  application=com.evermind.server.http.HttpApplication
  socket=sun.nio.ch.SocketAdaptor
  input=byte[]
  contentType="application/x-www-form-urlencoded"
  cookies=javax.servlet.http.Cookie[]
  method="POST"
  session=com.evermind.server.http.EvermindHttpSession
  application=com.evermind.server.http.HttpApplication
  values=java.lang.Object[]
    [0]="activeSubCategoryId"
    [1]=java.lang.Integer
    [2]="activeSubCategory"
    [3]="コンピュータ・インターネット"
    [4]="activeCategoryId"
    [5]=java.lang.Integer
    [6]="activeCategory"
```

Excat for Javaの概要



获取信息的时点

- 当发生指定的异常时
- 当指定的包(package)中发生异常时
- 当指定的方法被调用时
- 接受到外部的信号量时(在Windows系统中, 按下[Ctrl]+[Break]键)

能够取得的信息

- 堆栈情报
- 方法的输入参数、局部变量和返回值
- 方法的This对象和其他指定对象信息
- 异常对象的信息
- 造成线程等待的对象信息

其他

- 在获取故障情报信息时发送电子邮件到指定的邮箱地址来通知用户

Excat for Javaの特长



故障发生时将应用程序的状态以快照的形式进行完整的保存

- 解决了日志文件信息不足的问题
- 由于只保存了故障情报信息，因此解决了日志文件中多余信息过多的问题

可以即时更改获取快照的设定

- 不需要追加任何输出日志信息的代码
- 不需要重新启应用程序随时可将设定文件的变更内容进行应用

从实际运行/测试环境中获取准确的信息

- 针对特定环境才发生的故障，大大提高了不在开发环境中重现即可确定其故障原因的可能性

显示故障信息



- ・ 按照执行顺序显示故障信息
- ・ 显示与该信息对应的源代码

故障情報信息文件视图

堆栈试图

代码视图

属性视图

Java 障害分析支援ツール

ファイル(F) 編集(E) 表示(V) 設定(O) ヘルプ(H)

フィルター

log [C:\excat\log]

20081117

- java.lang.NumberFormatException
- test_IPRIDE-2_1_150_12_183629...
- test_IPRIDE-2_1_150_12_183629...
- test_IPRIDE-2_1_150_12_183629...
- test_IPRIDE-2_1_150_12_183629...

20081119

- com.ibatis.common.jdbc.exception.N...
- tutorial_IPRIDE-2_1_150_12_1849
- tutorial_IPRIDE-2_1_150_12_1849
- tutorial_IPRIDE-2_1_150_12_1849
- java.sql.SQLException
- tutorial_IPRIDE-2_1_150_12_1849
- tutorial_IPRIDE-2_1_150_12_1849
- tutorial_IPRIDE-2_1_150_12_1849
- execute
- jp.co.ipride.excat.sample.blogic.Inpu...
- tutorial_IPRIDE-2_1_150_12_1
- tutorial_IPRIDE-2_1_150_12_1

20081120

RIDE-2_1_150_12_1849

RIDE-2_1_150_12_1849

RIDE-2_1_150_12_1849

RIDE-2_1_150_12_1849

プロパティ

種類	内容
スレッド	jp.co.ipride.excat.sample.blogic.InputBLogic.execute
メソッド名	jp.co.ipride.excat.sample.blogic.InputBLogic.execute
バイトコード行番号	2
修飾子	public

ソース・パス: C:\Documents and Settings\chu\Desktop\workspace\Sample2\src\jp\co\ipride\excat\sample\blogic\inputBLogic.java

```
private UpdateDAO updateDao = null;

/**
 * 業務ロジック
 */
public BLogicResult execute(Input param) {
    // insert実行
    updateDao.execute("insertClient", param);

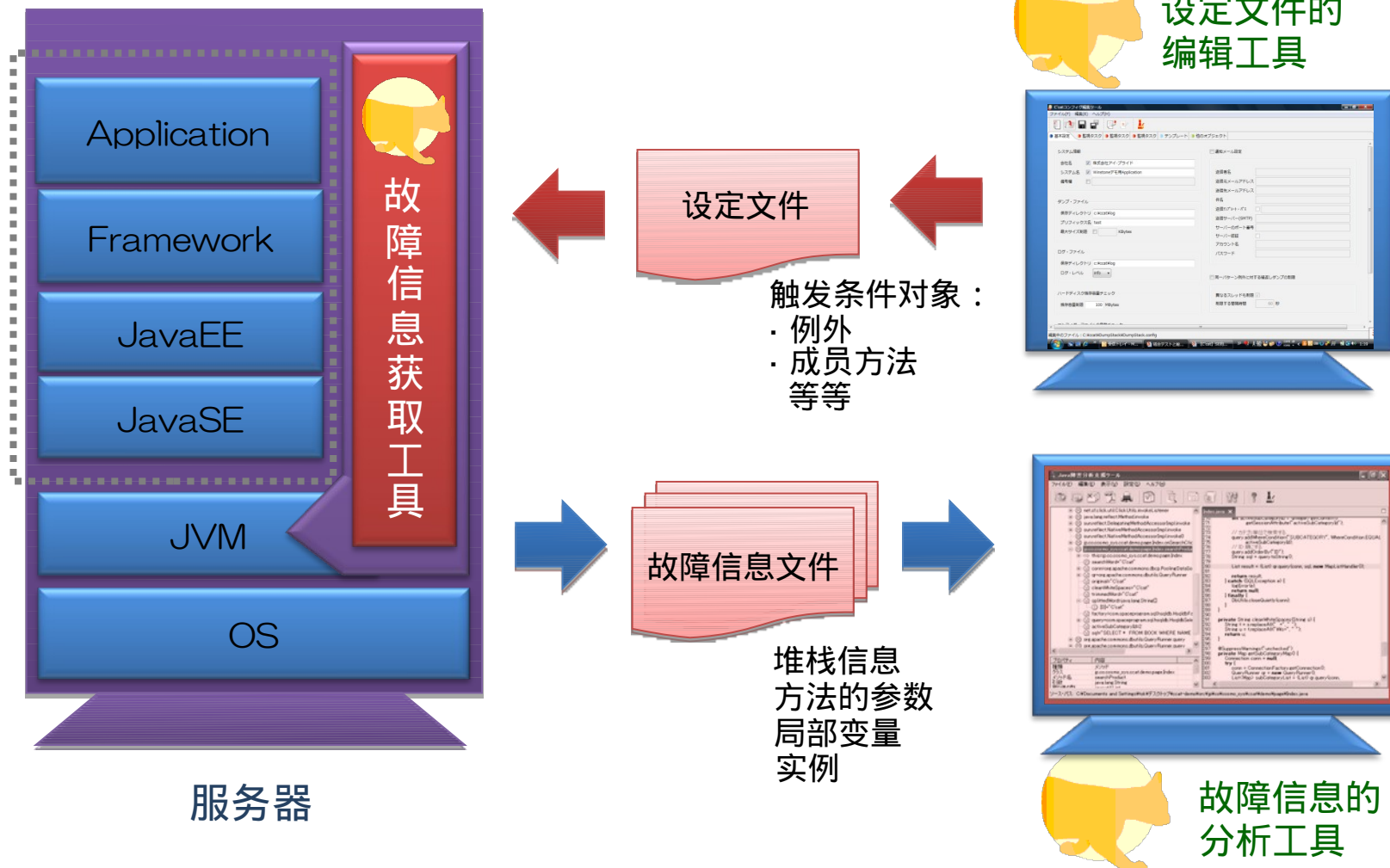
    //結果設定
    BLogicResult result = new BLogicResult();
    result.setResultString("success");
    return result;
}

/**
 * 更新用DAOも設定する。
 * @param updateDao 更新用DAO
 */
public void setUpdateDao(UpdateDAO updateDao) {
    this.updateDao = updateDao;
}

/**
 * 半角英数字のチェック
 * @param charstr チェック用文字列
 */
public boolean halfck(String charstr) {
    boolean rs = true;
    for (int i=0; i<charstr.length(); i++) {
        if (@charstr.charAt(i) > 0x0020 && charstr.charAt(i) < 0x007F) {
            rs = false;
            break;
        }
    }
    return rs;
}

/**
 * 文字列のチェック
 * @param charstr チェック用文字列
 */
public boolean tabck(String charstr) {
    boolean rs = true;
    String pattern0 = " ";
    String pattern1 = " ";
    int charcon0 = charstr.indexOf(pattern0);
    int charcon1 = charstr.indexOf(pattern1);
    if (charcon0 > charcon1) {
        rs = false;
        return rs;
    }
    return rs;
}
}
```


Excat for Java 的构成和运行方式





- 故障信息获取工具

适用于以下平台

- Windows 服务器 CPU: x86 32/64bit)
- Linux (CPU: x86 32/64bit)
- Solarisサーバ (CPU: SPARC, x86)
- HP-UXサーバ (CPU: Itanium 2)
- IBM AIXサーバ (CPU: POWER)

在以下的Web应用服务器已进行过运行验证

- Cosminexus
- Interstage
- Weblogic
- Websphere
- Oracle Application Server
- Adobe ColdFusion
- Tomcat
- JBoss

- 设定文件编辑工具和故障信息分析工具

适用于以下平台

- Windows、Linux



实例演示



① 系统构成

- ・ J2EEサーバ：Tomcat
- ・ フレームワーク：Terasoluna、SpringFramework、Struts、iBATIS等
- ・ DB：HSQLDB
- ・ AP：Web系统

② 故障现象

- ・ 用户正常登录时，发生错误。客户端显示错误页面。
- ・ 日志信息显示，由于没有保证数据的唯一性，而发生了SQLException例外。但具体原因不明。

③ Excat设定

- ・ 方法1：对引起该故障的例外进行监视。
- ・ 方法2：对执行该业务逻辑的方法进行监视

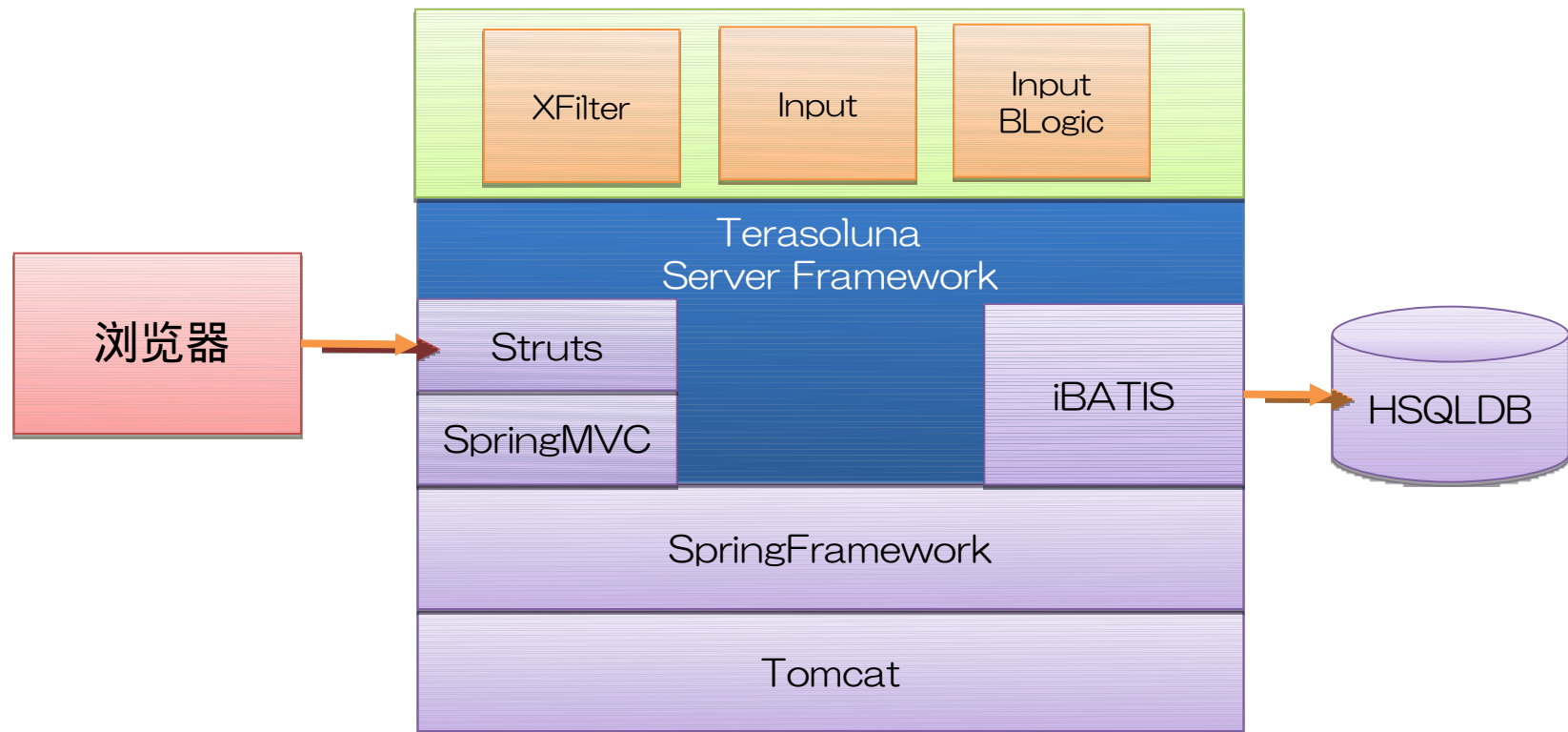
④ Excat解析结果

- ・ 掌握了未保持唯一性的数据以及相关数据流的详细信息。

演示系统的构成



演示系统构成应用程序





Excat for Java 的适用案例



1. 系统构成
 - ・ J2EE服务器：Tomcat
 - ・ DB：Postgres
 - ・ AP：Web系统
2. 故障现象
 - ・ 不明原因的异常中止
3. Excat设定
 - ・ 例外监视（使用与Tomcat对应的设定文件模板）
4. 再现故障
 - ・ 通过Excat获得以下信息：
 - ① 发生SQLException例外的线程信息
5. Excat解析结果
 - ・ 针对该SQLException的处理中含有bug
6. 原因与分析
 - ・ 由于在后续处理中，发生了预想外的情况，导致程序异常中止



1. 系统构成

- J2EE服务器：Weblogic
- DB：Postgres
- AP：基础系统

2. 故障现象

- 数据库时常不明原因的恢复数据
- 数据恢复的同时，必然伴随发生`java.sql.SQLException`和`XXXException`

3. Excat设定

- 对Transaction中的数据恢复方法进行监视成员
- 监视`XXXException`

4. 再现故障

- 通过Excat获得以下信息：
 - ① 执行数据恢复的方法的线程信息（线程A）
 - ② 同时，发生`XXXException`的线程的信息（线程B）



5. Excat解析结果

- 通过分析线程A获得以下结果
 - Timeout被设定为4小时
 - 数据恢复的原因在于Transaction的Timeout设定
- 通过分析线程B获得以下结果
 - 对通过JMS传输来的CSV文件的每一行记录进行处理
 - 在Transaction处理中，每次都会上传CSV文件
 - CSV文件内需要处理的记录共有4500条
 - 在这之中，一共处理了4000件，没有全部完成
 - 该处理因为XXXException而中止
 - XXXException是由于Timeout导致的SQLException而发生

6. 原因与对策

- 数据恢复的原因在于Transaction操作超时
- 可以通过控制CSV文件的最大记录数，或延长Transaction的超时限制来解决。



1. 系统构成
 - J2EE服务器：无
 - 框架：不明
 - DB：Postgres
 - AP：批处理包
2. 故障现象
 - AP处理时间过长
 - 根据性能解析工具的分析结果，A方法的执行时间过长，B方法的执行次数过多。
 - 然而，由于该批处理包的构造过于复杂，无法了解其工作原理，故而无法进行改善。
3. Excat设定
 - 对A方法和B方法进行监视
4. 再现运行逻辑关系
 - 完整的获取到了对A方法的调用路径
 - 完整的获取到了对B方法的调用路径
5. Excat解析结果
 - 查明了A方法和B方法的调用路径以及数据流



1. 系统构成

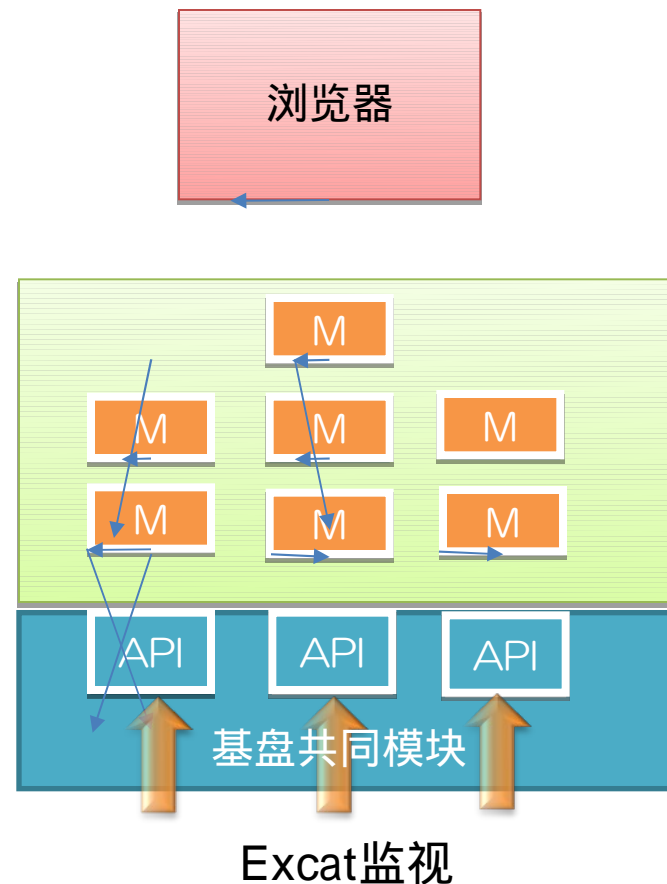
- J2EE服务器：Tomcat
- 框架：Struts
- 基盘：基盘共同模块
- AP：某业务系统

2. 应用程序构造

- 业务AP调用了基盘共同模块的API。
- 基盘共同模块调用了其他AP，并将结果返回给业务AP。
- 业务AP将从基盘共同模块获取的结果，表示在客户端的浏览器上。

3. Excat的适用内容

- 对基盘共同模块的API进行监视。
- 故障发生时，对Excat获取的数据进行比较，确认该调用该API的迁移与数据流。





1. 系统构成
 - J2EE服务器：不明
 - 框架：不明
 - DB：不明
 - AP：业务系统
2. 故障现象
 - 由log4j.error生成的错误级别的日志消息
 - 由于信息不足而无法确定发生原因
3. Excat设定
 - 对log4j.error方法进行监视
4. 再现错误
 - 获取错误发生点的执行路径和数据
5. Excat解析结果
 - 查明错误发生点的执行路径和数据流



感谢您的垂听