Best Practices for Application Management in Introscope

Abhijit Sawant
Terms of This Presentation

This presentation was based on current information and resource allocations as of October 2009 and is subject to change or withdrawal by CA at any time without notice. Notwithstanding anything in this presentation to the contrary, this presentation shall not serve to (i) affect the rights and/or obligations of CA or its licensees under any existing or future written license agreement or services agreement relating to any CA software product; or (ii) amend any product documentation or specifications for any CA software product. The development, release and timing of any features or functionality described in this presentation remain at CA’s sole discretion. Notwithstanding anything in this presentation to the contrary, upon the general availability of any future CA product release referenced in this presentation, CA will make such release available (i) for sale to new licensees of such product; and (ii) to existing licensees of such product on a when and if-available basis as part of CA maintenance and support, and in the form of a regularly scheduled major product release. Such releases may be made available to current licensees of such product who are current subscribers to CA maintenance and support on a when and if-available basis. In the event of a conflict between the terms of this paragraph and any other information contained in this presentation, the terms of this paragraph shall govern.
For Informational Purposes Only

Certain information in this presentation may outline CA’s general product direction. All information in this presentation is for your informational purposes only and may not be incorporated into any contract. CA assumes no responsibility for the accuracy or completeness of the information. To the extent permitted by applicable law, CA provides this document “as is” without warranty of any kind, including without limitation, any implied warranties or merchantability, fitness for a particular purpose, or non-infringement. In no event will CA be liable for any loss or damage, direct or indirect, from the use of this document, including, without limitation, lost profits, lost investment, business interruption, goodwill, or lost data, even if CA is expressly advised of the possibility of such damages.
Abstract

In this session, you will learn how to enhance and enrich your CA Wily APM investment. Discussions will cover such topics as enhanced scalability, installation and configuration; administration, and optimizing your application monitoring environment with new security features, and new interface features. By attending this session, you will be able to take full advantage of these improvements with the latest release to implement a better application performance management solution. Highlights include problem diagnosis 101, customising Introscope, real world examples, and tips and tricks.
Agenda

> Best Practices on Monitoring with Introscope Metrics
  - Metric Definition
  - Metrics created out of the box

> Best Practices on Introscope Java Instrumentation
  - How to use a new pbd file
  - How can you modify existing out of the box instrumentation

> Best Practices on Introscope Boundary Blame

> Best Practices on Avoiding Introscope Metric Explosion
  - What is special about SQL Metrics
  - SQL Agent Regex Normalizer
What are we monitoring?
Monitoring is Useful

> Failures can occur without warning
> Communication breaks down into the blame game amongst different groups
> Time to restore service is unpredictable
Controlling the Complexity

- 24x7 Application Monitoring
- Incident detection and monitoring
- Rapid triage
- Root cause diagnosis
- Trend Analysis
- Platform Independence
How is Introscope setup?

> Java/.NET Agent -> Byte Code Instrumentation
> Enterprise Manager
> Visualising the Data: Workstation (Thick Client), Webview (Thin Client)
Introscope Investigator Tree
Diagnosing with Transaction Trace

Visualizing a Transaction—Drill Down
Transaction Tracer provides multiple views of same data.
Tree view provides quick drill down.

Visualizing a Transaction—Client vs. Server
App Server Time
Network Time
End-to-End Transaction Time

Visualizing a Transaction—Back-End Database
Transaction Tracer shows the whole transaction from front ends to back-end SQL interaction.
Best Practices on Monitoring with Introscope Metrics
Metric Definition

- Represents a single unit of measurement from an Application Component
- Uniquely defined by its Metric Type and Metric Name
- A Data Point in Time
- Metric Name Example
  - Backends | Database Drive Name | SQL | SQL Query : Average Response Time (ms)
  - Frontends | Apps | Application Name : Responses Per Interval (ms)
Metric Types

> Types
  - String Constants
  - Integer Fluctuating Counters
  - Integer/Long Duration
  - Integer/Long Counters
  - Integer/Long Rate

> Possible Metrics
  - Average Response Time
  - Responses Per Interval
  - Errors Per Interval
  - Stall Count
  - Concurrent Count Per Interval
BlamePointTracer

> Tracer comprises of 5 metrics
> Out of the Box: BlamePointTracer is used for Servlet, EJB, Frontend, Backend metrics
> Common Tracer used in Introscope
Viewing data...

Drill down into specific data
Data is grouped by hierarchy
Best Practices on Introscope Boundary Blame
Create Effective Monitoring

> See specific data points, but...
> Where do I begin monitoring
> How do I troubleshoot an issue
> How do I know when an issue occurs
Boundary Blame

J2EE System

Transaction Request

Response

Host Systems

Web Services

Databases
Boundary Components in Introscope

> Frontend Components
   - Monitor data that enters the JVM
   - Out of the box: Servlets are marked as Frontends

> Backend Components
   - Monitor data that leaves the JVM
   - Out of the box: Sockets, JDBC are marked as Backends
Boundary Components in Introscope

> High level know quickly how your system is behaving
> Setup alerts and dashboards based on the Boundary data
> Called Backends provide granular data points specifically based on the Frontend caller
Best Practices on Introscope Java Instrumentation
There is an issue

> Alert gets fired – we are having an issue

> Alert Message 05:03:33 Service Responsiveness

Application A is reporting slow response greater than 2 seconds

How can we use Introscope to pinpoint the problem?
View Boundary Metrics

Why am I seeing high response times?
Run a Transaction Trace
Scenario: Diagnosing An Issue

- Frontend App is detecting a stall somewhere before we make our JDBC calls – where is it occurring?
- Need to Instrument a middle-layer class to detect if stalls are occurring
- Instrument Class Name is MuckInTheMiddle
- Method Name and Signature is muck()
Let’s find the Culprit: Adding Custom Pbd

> Add a new pbd file in the default pbl file
> Reference the custom.pbd file in the IntroscopeAgent.profile
> Add the custom.pbd file in the hotdeploy folder (reference to Dynamic Instrumentation)
HotDeploy Folder Example
Adding Instrumentation

> Need to provide the Class and Method name you want to instrument
> How do you monitor the data? -&gt; Which metrics do you want to create -&gt; Which tracer to you select?
> Introscope 9.0 will provide this interface by using a Dynamic Instrumentation UI

> SetFlag: MuckInTheMiddleTracing
> TurnOn: MuckInTheMiddleTracing


> IdentifyClassAs: com.wily.str.muckinmiddle.MuckInTheMiddle MuckInTheMiddleTracing
> TraceOneMethodIfFlagged: MuckInTheMiddleTracing muck MuckBlamePointTracer "{classname}|{method}"
After Instrumentation

<table>
<thead>
<tr>
<th>Workstation</th>
<th>Edit</th>
<th>Trace</th>
<th>Properties</th>
<th>Help</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Domain</td>
<td>Host</td>
<td>Process</td>
<td>Agent</td>
</tr>
<tr>
<td>T</td>
<td>&quot;Super Domain&quot;</td>
<td>DESNE01-PRE...</td>
<td>Unknown Process</td>
<td>Unnamed Agent</td>
</tr>
<tr>
<td>T</td>
<td>&quot;Super Domain&quot;</td>
<td>DESNE01-PRE...</td>
<td>Unknown Process</td>
<td>Unnamed Agent</td>
</tr>
</tbody>
</table>

Summary View → Trace View → Tree View
Agent: "Super Domain"|DESNE01-PRE490|Unknown Process|Unnamed Agent
Timestamp: 09/27/09 19:57:48 PDT
Duration: 91243 ms

0 ms 4000 8000 12000 16000 20000 24000 28000 32000 36000 40000 44000 48000 52000 56000 60000 64000 68000 72000 76000 80000 84000 88000 92000
Frontends|Apps|Application A|URLs|Default
Sandals|Dhro_ming_Sandals
MuckInTheMiddle|muck

Component Details
Identification
Type: Frontends
Name: Default
Path: Frontends|Apps|Application A|URLs|Default

Performance
Duration: 91243 ms
Configure Tracing

> Design Instrumentation to easily configurable using pbd syntax
> Concept of toggling flags to turn on and off tracing

> SetFlag: MuckInTheMiddleTracing
> #TurnOn: MuckInTheMiddleTracing


> IdentifyClassAs: com.wily.str.muckinmiddle.MuckInTheMiddle MuckInTheMiddleTracing
> TraceOneMethodIfFlagged: MuckInTheMiddleTracing muck MuckBlamePointTracer "\{classname\}|\{method\}"
Adding a New Flag into the Existing Pbd Set

> # EJB Configuration
> # ================
> TurnOn: SessionBeanTracing
> TurnOn: EntityBeanTracing
> TurnOn: MessageDrivenBeanTracing
> # TurnOn: EJBMethodLevelTracing

> # HTTP Servlets Configuration
> # ================
> TurnOn: HTTPServletTracing

> # XML Configuration
> # ================
> # TurnOn: XMLSAXTracing
> # TurnOn: XSLTTracing
> TurnOn: MuckInTheMiddleTracing

Ability to turn on and off instrumentation points
Be careful not to Over-Instrument

> Instrument key components -> Boundary components that provide a good overview of your system to do quick diagnosis
> Do not instrument POJO Classes and Methods
> Do not instrument methods that get called thousands of times in a 15 second interval
> Agent is not designed to be a profiler -> creating too many metrics becomes expensive
Best Practices on Avoiding Introscope Metric Explosion
## Collector Sample Recommended Data

### Sizing and Performance Guide

<table>
<thead>
<tr>
<th>Operating System</th>
<th>Hardware</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Physical RAM</td>
</tr>
<tr>
<td>Solaris</td>
<td>2 CPU UltraSPARC III, Clock speed ~ 1.2 GHz</td>
</tr>
<tr>
<td>Solaris</td>
<td>4 CPU UltraSPARC III, Clock speed ~ 1.2 GHz</td>
</tr>
<tr>
<td>Solaris</td>
<td>4 CPU UltraSPARC III, Clock speed ~ 1.2 GHz</td>
</tr>
<tr>
<td>Red Hat Linux</td>
<td>2 CPU Xeon or Opteron, Clock Speed ~ 3 GHz</td>
</tr>
</tbody>
</table>
### Collector Sample Recommended Data

<table>
<thead>
<tr>
<th>Operating System</th>
<th>Hardware</th>
<th>Physical RAM</th>
<th>JVM Heap</th>
<th>Max # Agents /EM</th>
<th>Max # Metrics*</th>
<th>Max # Applications /EM</th>
<th>Max # Events/minute</th>
<th>Max # Virtual Agent Matched Metrics</th>
<th>Max # Workstation Connections per Standalone EM</th>
<th>Max # EMs/machine</th>
<th>Max metrics in metric groupings/Standalone EM</th>
</tr>
</thead>
<tbody>
<tr>
<td>AIX 5.3</td>
<td>4 CPU</td>
<td>4 GB</td>
<td>1.5 GB</td>
<td>250</td>
<td><strong>400,000</strong></td>
<td>1500</td>
<td>850</td>
<td>4000</td>
<td>3500</td>
<td>50</td>
<td>15% of Max # metrics (60,000)</td>
</tr>
<tr>
<td>Windows 2000/2003</td>
<td>2 CPU</td>
<td>4 GB</td>
<td>1.5 GB</td>
<td>300</td>
<td><strong>500,000</strong></td>
<td>1500</td>
<td>1000</td>
<td>5000</td>
<td>5000</td>
<td>50</td>
<td>15% of Max # metrics (75,000)</td>
</tr>
<tr>
<td>Windows 2000/2003</td>
<td>4 CPU</td>
<td>4 GB</td>
<td>1.5 GB</td>
<td>400</td>
<td><strong>500,000</strong></td>
<td>3000</td>
<td>1000</td>
<td>5000</td>
<td>5000</td>
<td>50</td>
<td>30% of Max # metrics (150,000)</td>
</tr>
<tr>
<td>Windows 2000/2003</td>
<td>4 CPU</td>
<td>8 GB</td>
<td>1.5 GB</td>
<td>300</td>
<td><strong>500,000</strong></td>
<td>1500</td>
<td>1000</td>
<td>5000</td>
<td>5000</td>
<td>50</td>
<td>15% of Max # metrics (75,000)</td>
</tr>
</tbody>
</table>
Creating Too Many Metrics

> Custom Instrumenting: a new framework, custom application, etc...
> Prevent unnecessary tracing by turning off flags
> Even if instrumented methods are not invoked metrics are still running
SQL Metric Case
SQL Metric Case

> Important to monitor SQL metrics
> How can we minimise the number of SQL metrics
> Use a SQL Normalizer
  - Write a SQL Normalizer Extension
  - Regex SQL Normalizer
RegexSQLNormalizer Agent Extension

- RegexSqlNormalizer.jar shipped under wily/ext
- Implementation
  - com.wily.introscope.agent.extensions.sqlagent.RegexSqlNormalizer
- An actual extension that normalises based on regular expressions
  - Java – valid syntax in java.regex package
  - Dotnet – valid syntax in System.Text.RegularExpressions
- Regex properties specified in agent profile
- No need for writing normalisation code
- Normalisation controlled by regex properties
RegexpSQLNormalizer Agent Extension

> Agent profile properties - are “HOT”
  > introscope.agent.sqlagent.normalizer.extension=RegexpSQLNormalizer
  > introscope.agent.sqlagent.normalizer.regex.keys=default
  > introscope.agent.sqlagent.normalizer.regex.default.pattern
    ▪ Specify the regex pattern to match in the sql string
  > introscope.agent.sqlagent.normalizer.regex.default.replaceAll=
  > introscope.agent.sqlagent.normalizer.regex.default.replaceFormat
    ▪ Specify the replacement format for the matched pattern
  > introscope.agent.sqlagent.normalizer.regex.default.caseSensitive
  > introscope.agent.sqlagent.normalizer.regex.matchFallThrough
    ▪ Are all keys evaluated/First match wins
RegexSQLNormalization Agent Extension

> Solving using RegexSqlNormalization - example scenarios

> Input SQL:

- SELECT * FROM TMP_123981398210381920912 WHERE ROW_ID =

> Desired Normalized SQL:

- SELECT * FROM TMP_ WHERE ROW_ID =

> Configuration needed to achieve the above normalized sql:

- introscope.agent.sqlagent.normalizer.extension=RegexSqlNormalization
- introscope.agent.sqlagent.normalizer.regex.matchFallThrough=true
- introscope.agent.sqlagent.normalizer.regex.keys=key1
- introscope.agent.sqlagent.normalizer.regex.key1.pattern=(TMP_)[1-9]*
- introscope.agent.sqlagent.normalizer.regex.key1.replaceAll=false
- introscope.agent.sqlagent.normalizer.regex.key1.replaceFormat=$1
- introscope.agent.sqlagent.normalizer.regex.key1.caseSensitive=false
Summary and Q&A
Help Us Help You!

<table>
<thead>
<tr>
<th>Feedback and Additional Requirements</th>
<th>Customer Name</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Development Exchange Summit 2009
CA User Community Forums

> Discuss CA product related topics
> Share best practices
> Exchange tips & tricks.

Some notable features of the forums include: user ranking based on contributions, tagging, posting attachments, a powerful search feature, RSS notifications, profile preferences, and private messaging.