AspectWerkz

A deeper view in the new “online mode” architecture

Alexandre Vasseur – alex@gnilux.com
Jonas Bonér – jonas@codehaus.org

Agenda

- Who should read, why
- AspectWerkz prior to 0.8 release
- AspectWerkz new architecture and options
- Choose to suit your needs
- Migration effort for value added projects
- Key advantages

Who should read, why

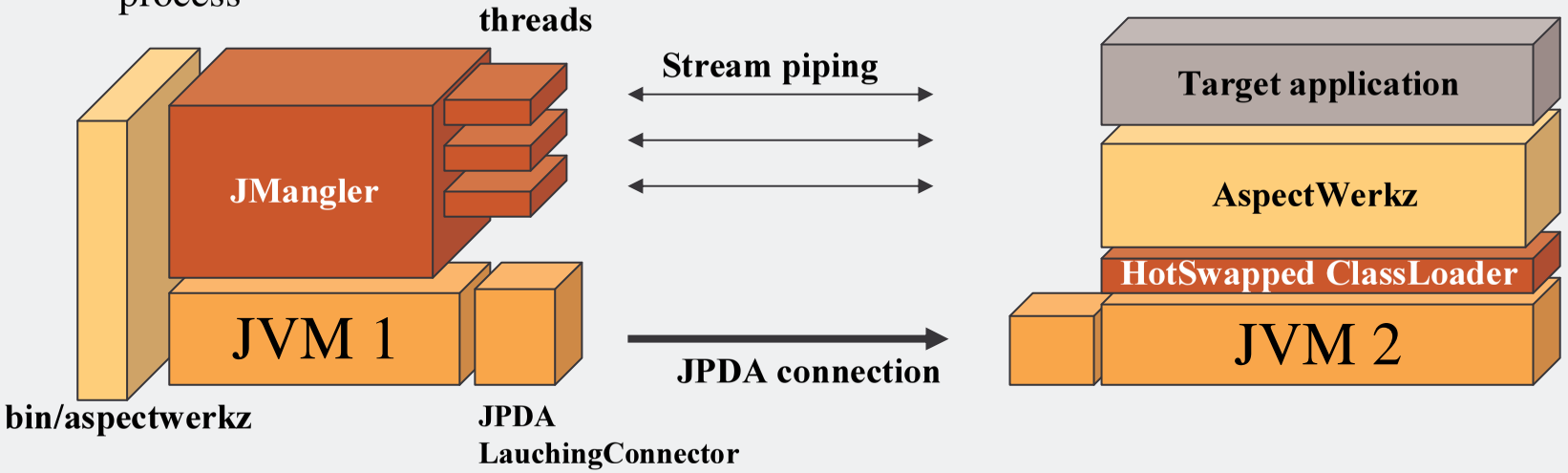
- Who should read
  - AspectWerkz user
  - AspectWerkz value added projects people (Inectis, VirtualMock, Oslo…)
- Learn more
  - About the new features mainly related to online mode
  - How to choose the best option for your needs
  - About the impacts on projects using AspectWerkz prior to 0.8

Prior to 0.8 release (1)

- Online mode (on the fly weaving) based on Jmangler 3
- Online mode requires JVM to run in debug mode
- Java 1.4 only
- LGPL license due to the greedy effect of JMangler license

Prior to 0.8 release (2)

- The aspectwerkz wrapper script (bin/aspectwerkz) launch a first JVM.
- The JVM launchs the target application in a second JVM suspended, in debug mode.
- The java.lang.ClassLoader of the second JVM is replaced by a patched one (HotSwap thru Sun JPDA JDI API)
- The streams of the second JVM are piped thru 3 background threads to appear as normal (stderr, stdout, stdin)
- The second JVM is resumed, loaded class (including « main ») go thru the weaving process



Prior to 0.8 release (3)

- Pros:
  - JVM wide system – all loaded class of all classloaders (system, customs) can be weaved
  - Elegant way (no need to distribute a java.lang.ClassLoader)
- Cons:
  - Only java 1.4, HotSwap compliant JVM
  - Starts two JVMs and background threads
  - Complexity thru wrapper bin/aspectwerkz
  - The JPDA connection of the second JVM is not released : no way to do remote debugging further
  - The mechanism is bogus for heavy classloading scheme (like app server startup sequence)

Prior to 0.8 release (4)

- JMangler 3.x idea of hotswapping java.lang.ClassLoader is really good
- But…
  - Bogus in app server environment
  - JMangler provides lots of other “AOP” features, not used in AspectWerkz
  - Does not provides enough flexibility and does not allow for advanced low level features in AspectWerkz
  - Is bad to integrate seamlessly (config files, big jar)

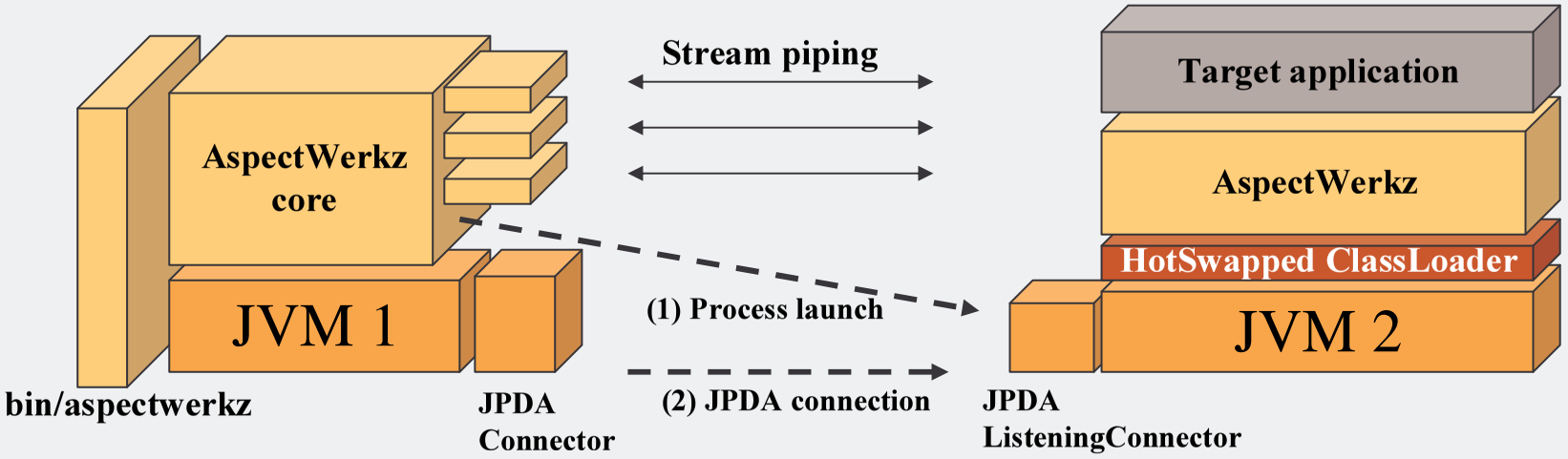
=> Time for new requirements

New architecture (1)
Key points

- Online mode is native to AspectWerkz (no JMangler dependancy, full control over the sources, releases, support and license)
- Online mode does not requires JVM to run in debug mode
- Java 1.3 support
- Several options available to allow online mode – just choose according to your needs:
  - HotSwap (the JMangler idea – but improved new architecture)
  - Transparent bootclasspath
  - Remote HotSwap
  - Prepared bootclasspath
  - Native HotSwap

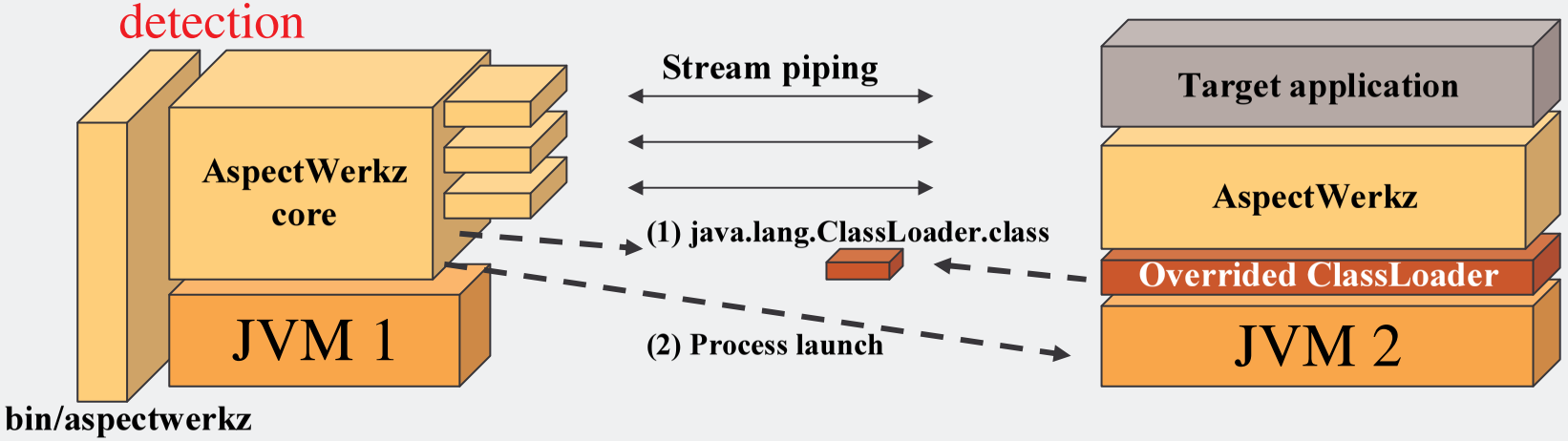
New architecture (2)
HotSwap

- A first JVM launch the application JVM in debug mode, suspended (1)
- The java.lang.ClassLoader is hotswapped to allow weaving (2)
- The streams are piped thru background threads, and the second JVM is resumed
- Key points: the JPDA connection is released, allowing further remote debugging thru 3rd party tools



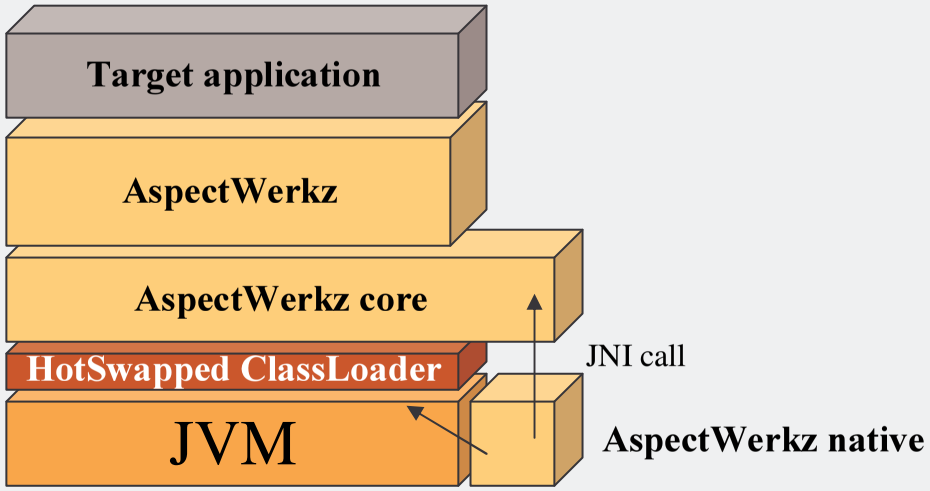
New architecture (3)
Transparent bootclasspath

- A first JVM transforms the java.lang.ClassLoader and stores it on the file system (1)
- The first JMV launch the application JVM specifying to use the transformed java.lang.ClassLoader thru *–Xbootclasspath* option (2)
- The streams are piped thru background threads
- Key points: no debug mode, java 1.3 compliance, java 1.3 auto detection



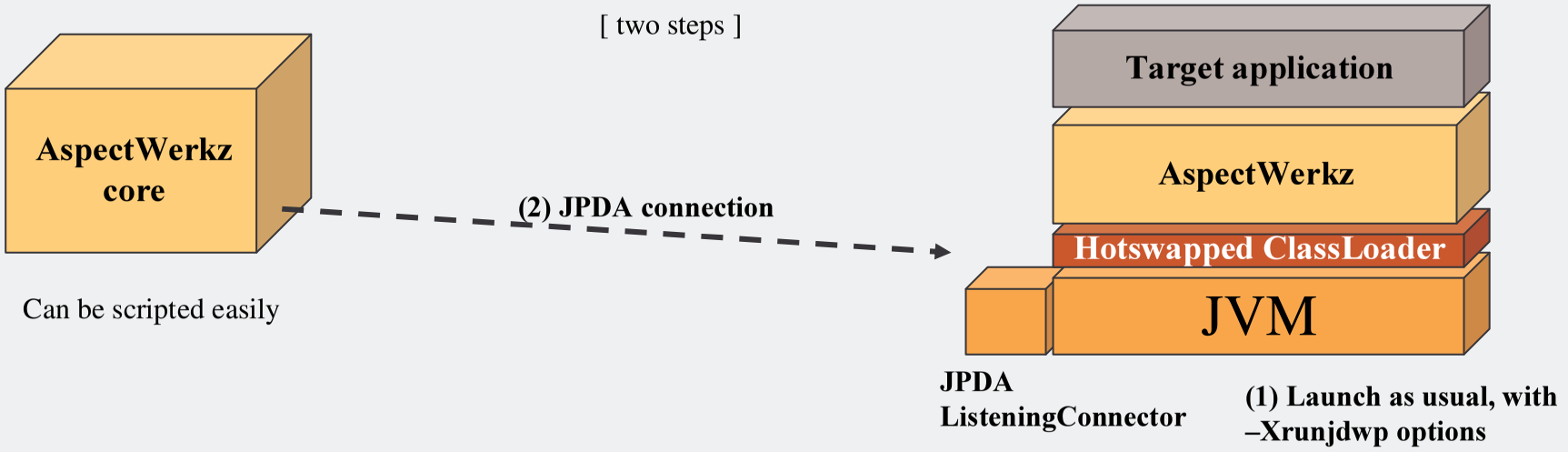
New architecture (4)
Native HotSwap

- The JVM is launched with a native JVMPI extension (-Xrun…)
- This extension - when loads - hotswaps the java.lang.ClassLoader using JNI calls to AspectWerkz core and native JVMDI API
- Key points: a single JVM, a very non intrusive way of plugin AspectWerkz in java 1.4



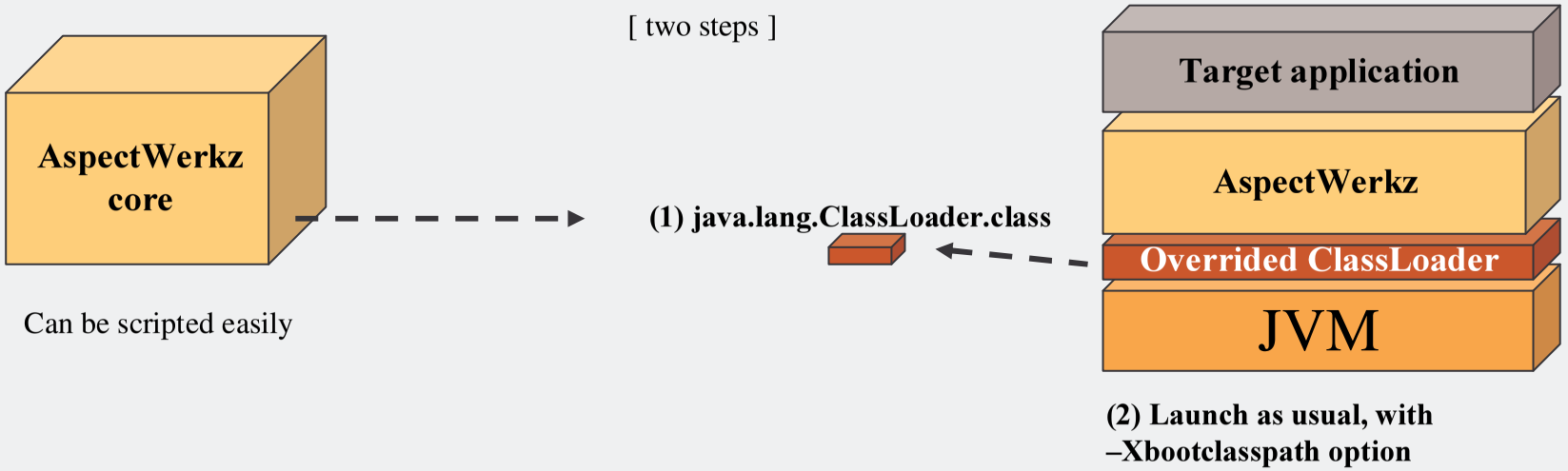
New architecture (5)
Remote HotSwap

- The application JVM is launched in debug mode, with a JPDA listening connector, suspended (1)
- An AspectWerkz « main » connects to the launched JVM thru JPDA, hotswaps the java.lang.ClassLoader and resume the launched JVM (2)
- Key points: a single JVM, no stream piping, no wrapper script, the JPDA connection is released (and can also be used in a running VM).



New architecture (6)
Prepared bootclasspath

- An AspectWerkz « main » transforms the java.lang.ClassLoader and stores it on the file system
- The application JVM is launched with the *–Xbootclasspath* option
- Key points: a single JVM, no stream piping, no wrapper script, java 1.3 support



Choose the right option

Option	Java version	Number of JVM running	Application JVM in debug mode	Usage	Notes
HotSwap	1.4, HotSwap compliant JVM	2	Yes (see (a))	bin/aspectwerkz Default mode for java 1.4 compatible JVM	Auto detection mechanism. Allow to launch an application with AspectWerkz in the same way for java 1.4 and 1.3
Transparent bootclasspath (see (b))	1.3, 1.4	2	Not required	bin/aspectwerkz Default mode for java 1.3 or HotSwap incompatible JVM Can be used for java 1.4 to avoid running JVM in debug mode	
Native HotSwap	1.4, HotSwap compliant JVM	1	Yes (see (a)) (Except 1.4.0)	Standard	Recommended mode for production environment on java 1.4
Unplugged HotSwap	1.4, HotSwap compliant JVM	1	Yes (see (a))	Two steps process	Compliant with running VM
Prepared bootclasspath (see (b))	1.3, 1.4	1	Not required	Two steps process One step process	

(a) Java 1.4 debug mode has a very little overhead on the JVM performance

(b) Not compliant with the Java runtime license – educational / prototyping purpose only

Migration effort

- Reduced to zero for end users
  - bin/aspectwerkz runs the default auto detection mode of AspectWerkz (HotSwap / Transparent bootclasspath)
- A minimal effort for value added projects
  - Update your maven / ant build
  - AspectWerkz is distributed in two jars (aspectwerkz and aspectwerkz-core for the JVM launchers)
  - Read migration FAQ

Key advantages

- Java 1.3 support
- Many options, and a default mode that allow auto detection to ease development and integration efforts for end users and value added projects people
- Totally controlled and innovative architecture allowing performance wise JVM wide AOP
- Questions and fix goes directly in mailing lists, Jira and CVS
- A complete new light architecture with many new features.
- AOP made seamless with the native HotSwap feature

Read more

- http://aspectwerkz.codehaus.org
- http://lists.codehaus.org/mailman/listinfo/aspectwerkz-user
- http://blogs.codehaus.org/people/avasseur
- http://blogs.codehaus.org/people/jboner
- http://java.sun.com/products/jpda/

- http://javalab.cs.uni-bonn.de/research/jmangler/