



JET

A Distributed Test Engine

Vemund Østgaard
MySQL System QA
Sun Microsystems, Inc.



Goals for this presentation

- Describe what JET can do and how it is done
- Answer if JET could be useful for you
- Give ideas and inspiration to those that are working on similar frameworks

Agenda

- **Introduction**
- Overview
- Components
 - > JET
 - > JETBatch
 - > JAG
- Use cases and examples

Introduction

- JET = Java Engine for Testing
- Developed by Sun's Database Group QA team
- Development started in 2001.
- Open sourced during the summer of 2009
- Maintained on Kenai:
<http://kenai.com/projects/jet>

Agenda

- Introduction
- **Overview**
- Components
 - > JET
 - > JETBatch
 - > JAG
- Use cases and examples

Overview

- So what exactly is it?
 - > It is a test framework written in Java
 - > It is designed for testing distributed products
 - > It is built on top of Junit

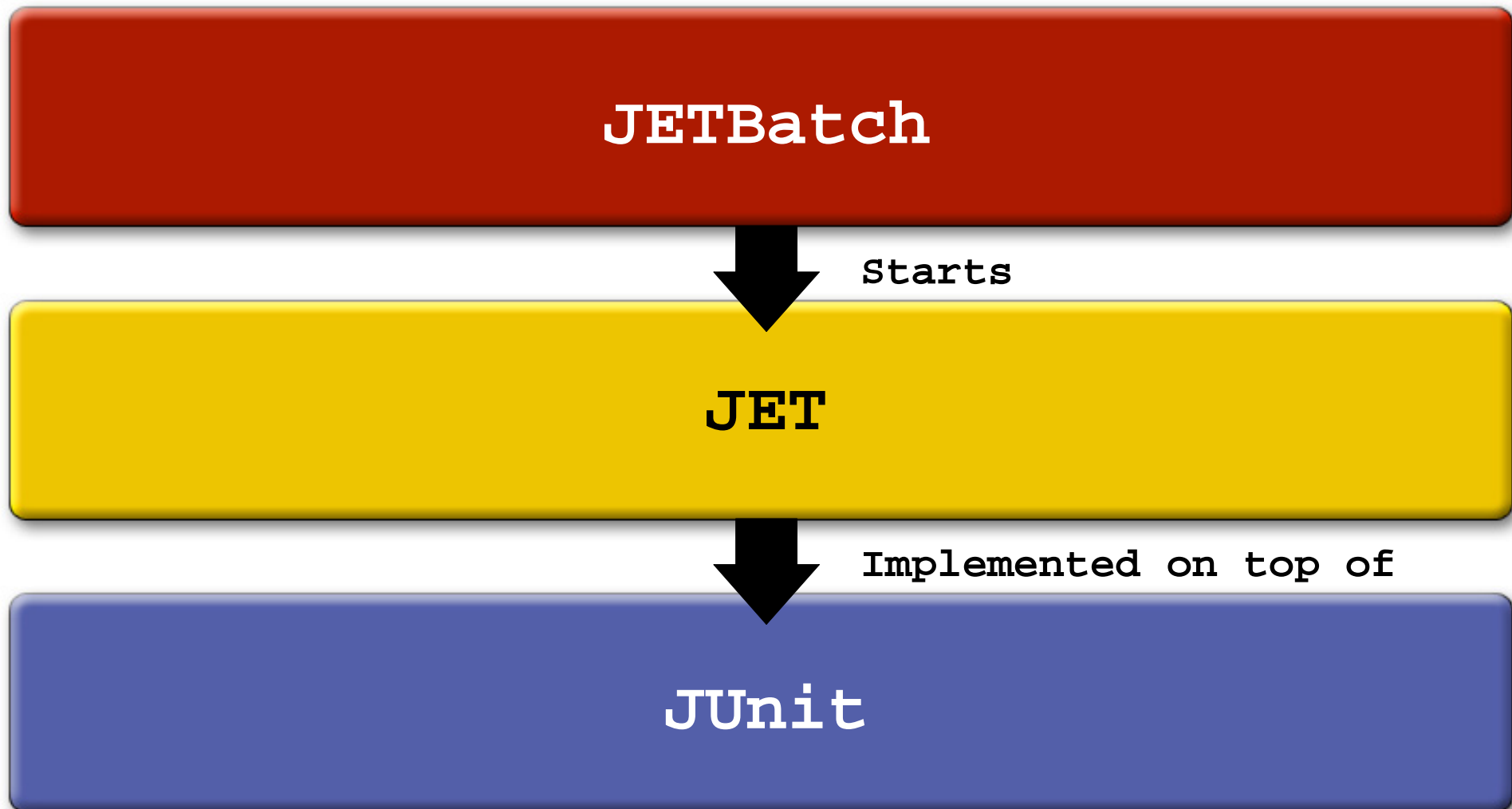
Some facts

- JET has been run on Solaris, Linux, Windows, Mac OS X and HP-UX.
- JET has been used to test HADB, PostgreSQL, Memcached, Java DB, MySQL Server and MySQL Cluster Mgmt.
- JET has been used for functional testing as well as many types of non-functional testing: stress, robustness, performance, scalability, longevity, large volume, etc.

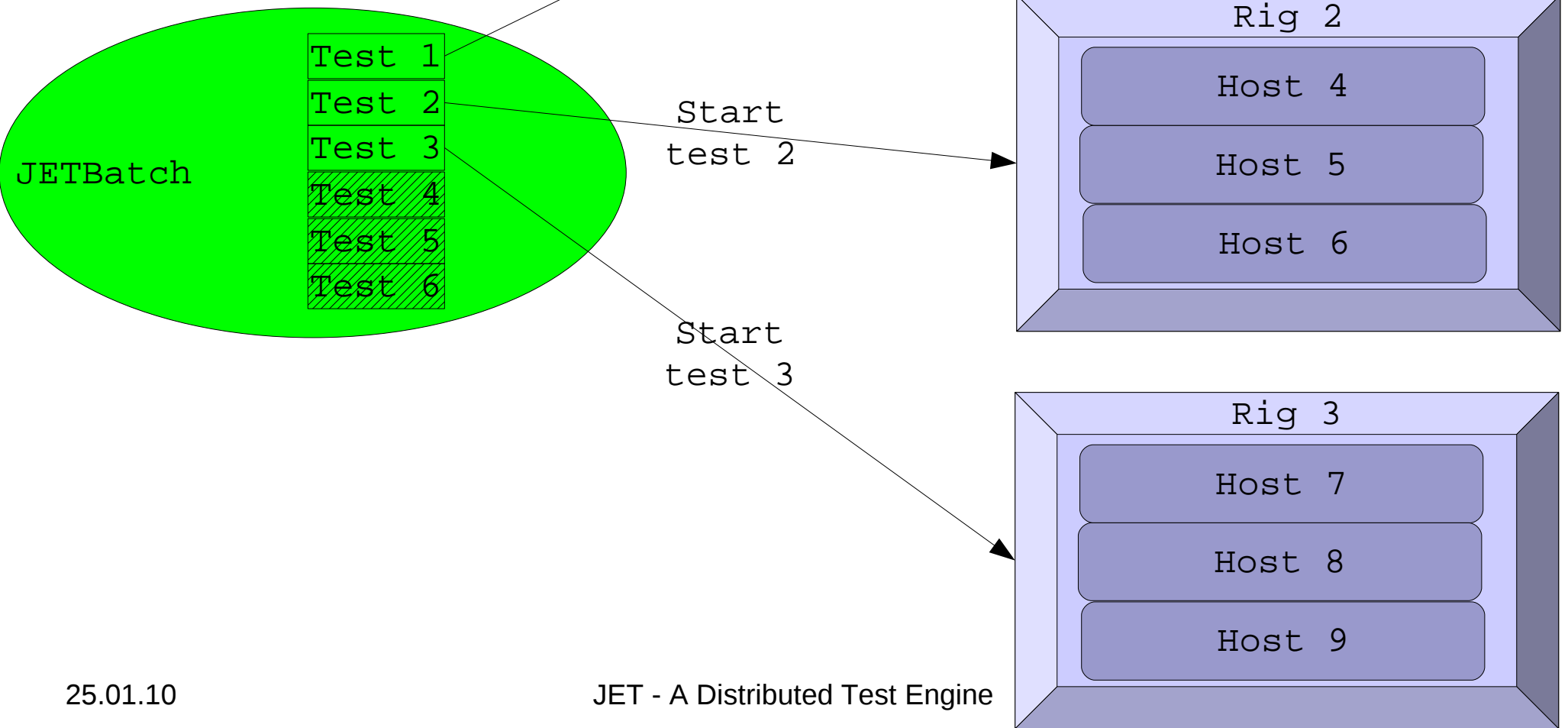
Components

- JET framework is 3 fairly separate components:
 - > JET engine: Takes an XML-file as input, constructs a test and executes it.
 - > JETBatch engine: Executes a set of JET tests on a set of machines.
 - > JAG agent: Used by JET and JETBatch for remote access to machines.

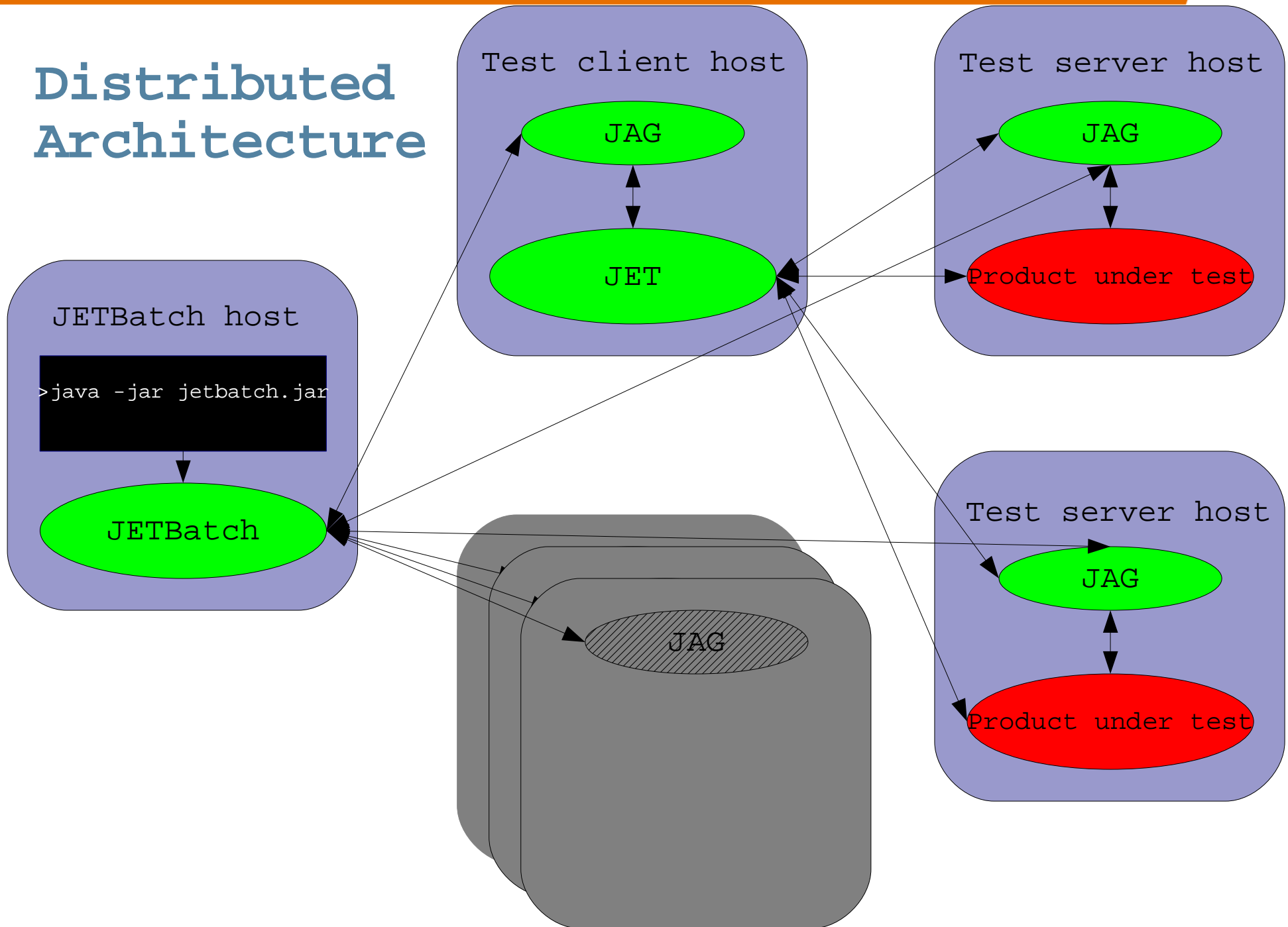
Layered architecture



Distributed Architecture



Distributed Architecture



Agenda

- Introduction
- Overview
- **Components**
 - > **JET**
 - > JETBatch
 - > JAG
- Use cases and examples

JET Engine features

- Easily portable framework and tests, 100% Java
- Written on top of JUnit: TestCases & TestSetups, assert based
- Tests created as XML-files describing test classes to use
- Test code written as reusable building blocks
- Property based configuration of tests and framework
- Can run standalone without using JAG and JETBatch

JET Engine logic

Step 1: Read runtime settings from users property file

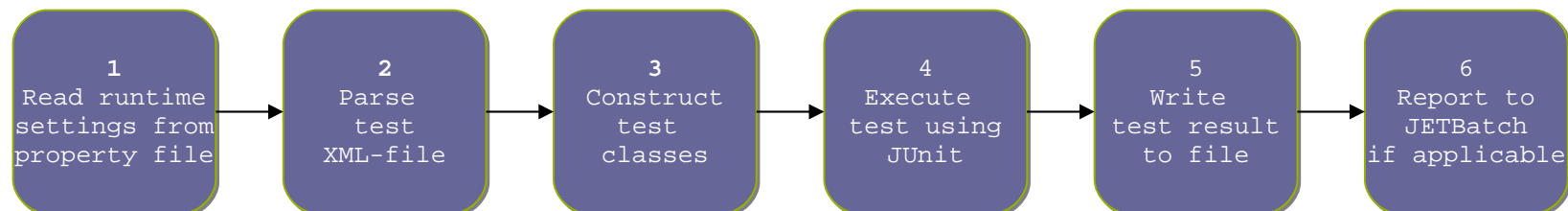
Step 2: Parse the test XML-file

Step 3: Construct test classes from parsed information

Step 4: Execute test using the test classes and JUnit

Step 5: Write test result to file

Step 6: Report back to JETBatch if applicable



Configuration of JET and tests

- JET, JETBatch and tests share the same sources for configuration
- Configuration determined by hierarchy
 - 1) XML-file (hierarchical in itself)
 - 2) User supplied property file
 - 3) System properties
 - 4) Defaults property file
- Test and user supplied properties verified as legal against defaults property file and alternatively using validators.

Agenda

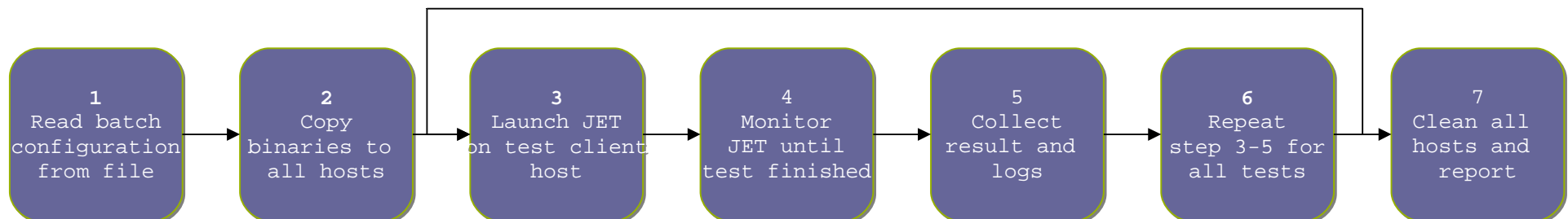
- Introduction
- Overview
- **Components**
 - > JET
 - > **JETBatch**
 - > JAG
- Use cases and examples

JETBatch features

- Runs batches of JET tests
- Can distribute tests across many groups of machines concurrently
- Handles distribution of binaries to test hosts
- Monitors/pings JET during test execution
- Pluggable support for different reports (mail, file)

JETBatch logic

- Step 1:** Read configuration from property/xml-file
- Step 2:** Copy binaries to all test hosts to be used
- Step 3:** Start JET on client host (per set of hosts)
- Step 4:** Wait until test is finished and start the next
- Step 5:** Collect result and logs from JET execution
- Step 6:** Repeat step 3-5 until no more tests to run
- Step 7:** Clean all hosts used and produce report/mail



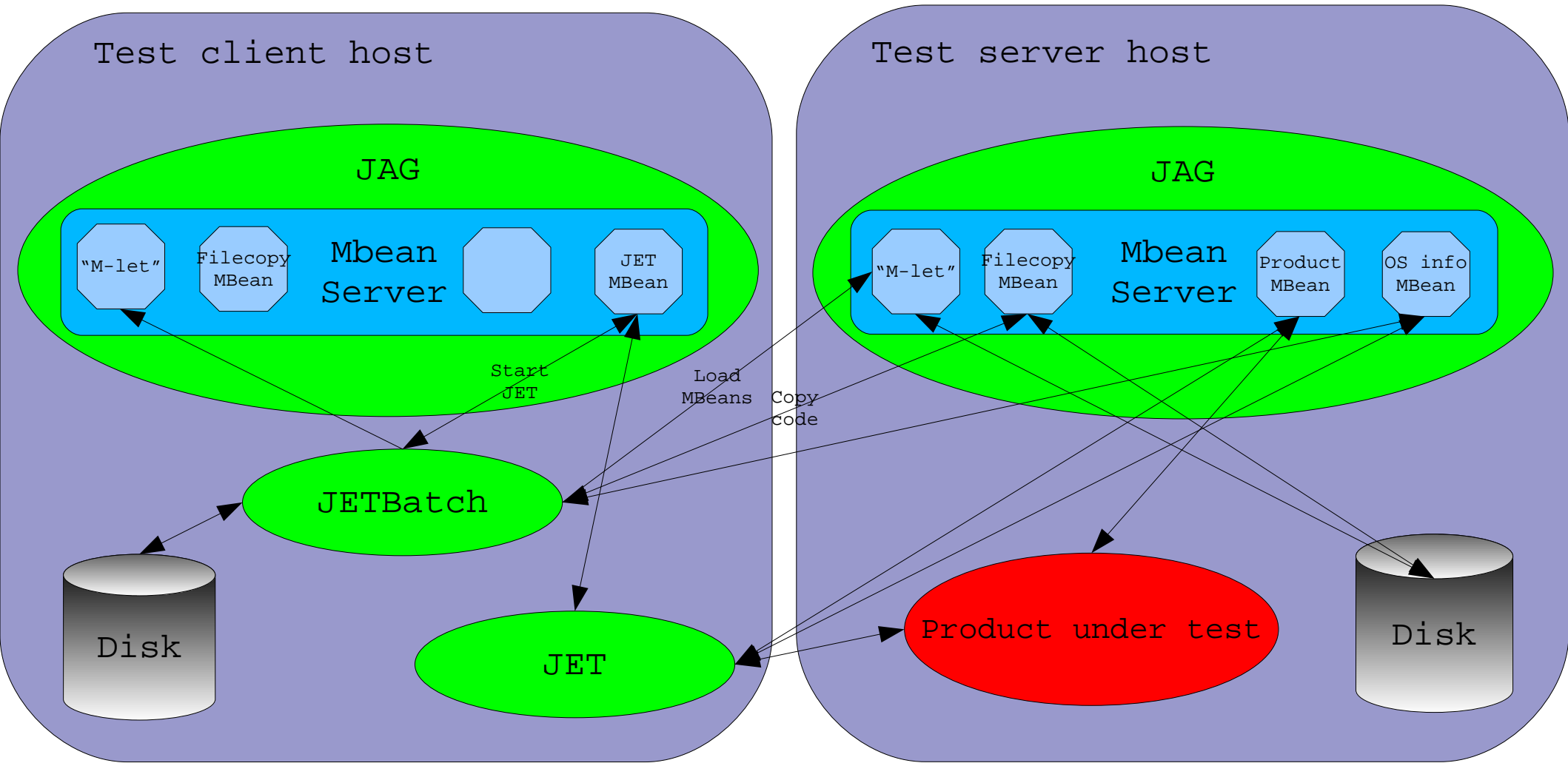
Agenda

- Introduction
- Overview
- **Components**
 - > JET
 - > JETBatch
 - > **JAG**
- Use cases and examples

JAG features

- Remote management (JMX): tests, resources, JAG
- MBeans can also be accessed from web browser
- Monitors and cleans resources: MBeans, processes, disk
- Tests executed as the user running JAG
- Provide an OS-transparent view of test hosts
- Tests use MBeans of compatible version (dynamic loading)
- Different versions of same MBean may coexist
- Supports timers and alerts

JAG Architecture



Agenda

- Introduction
- Overview
- Components
 - > JET
 - > JETBatch
 - > JAG
- **Use cases and examples**

Tests described in xml

- Assemble test code into actual tests.
- Comment attribute, Documentation tags
- Can be pretty printed in html

- Test tags

- > TestSetup

- > TestCase

- > TestSuite

- > LoadClient

- > Binding

```

<TestSetup class="com.sun.jet.dbmgt.dbSetup"
  comment="Do DB Setup">
  <Binding key="actives" value="2" comment="Starting with 2 active nodes"/>
  <Binding key="spares" value="2" comment="Starting with 2 spare nodes" />
  <Binding key="bloblength" value="14000" />
  <Binding key="datadevicesize" value = "300"
    comment="Use smaller devices" />
  <Binding key="cleanup" value="false"
    comment="Do not delete historyfiles if setup fails" />
  <Binding key="logbuffersize" value="96" />

  <TestSuite>
  <!-- Basic Management tests -->
    <TestCase class="com.sun.jet.dbmgt.dbmgtTestCase" method="testGetAll"/>
    <TestCase class="com.sun.jet.dbmgt.dbmgtTestCase" method="testGet"/>
    <TestCase class="com.sun.jet.dbmgt.dbmgtTestCase" method="testList"/>
    <TestCase class="com.sun.jet.dbmgt.configTestCase" method="testGetHistorypath"/>
    <TestCase class="com.sun.jet.dbmgt.configTestCase" method="testGetDevicepath"/>
    <TestCase class="com.sun.jet.dbmgt.configTestCase" method="testGetStateOnRunning"/>
  </TestSuite>
</TestSetup>

```

Remote calls to JAG in test code

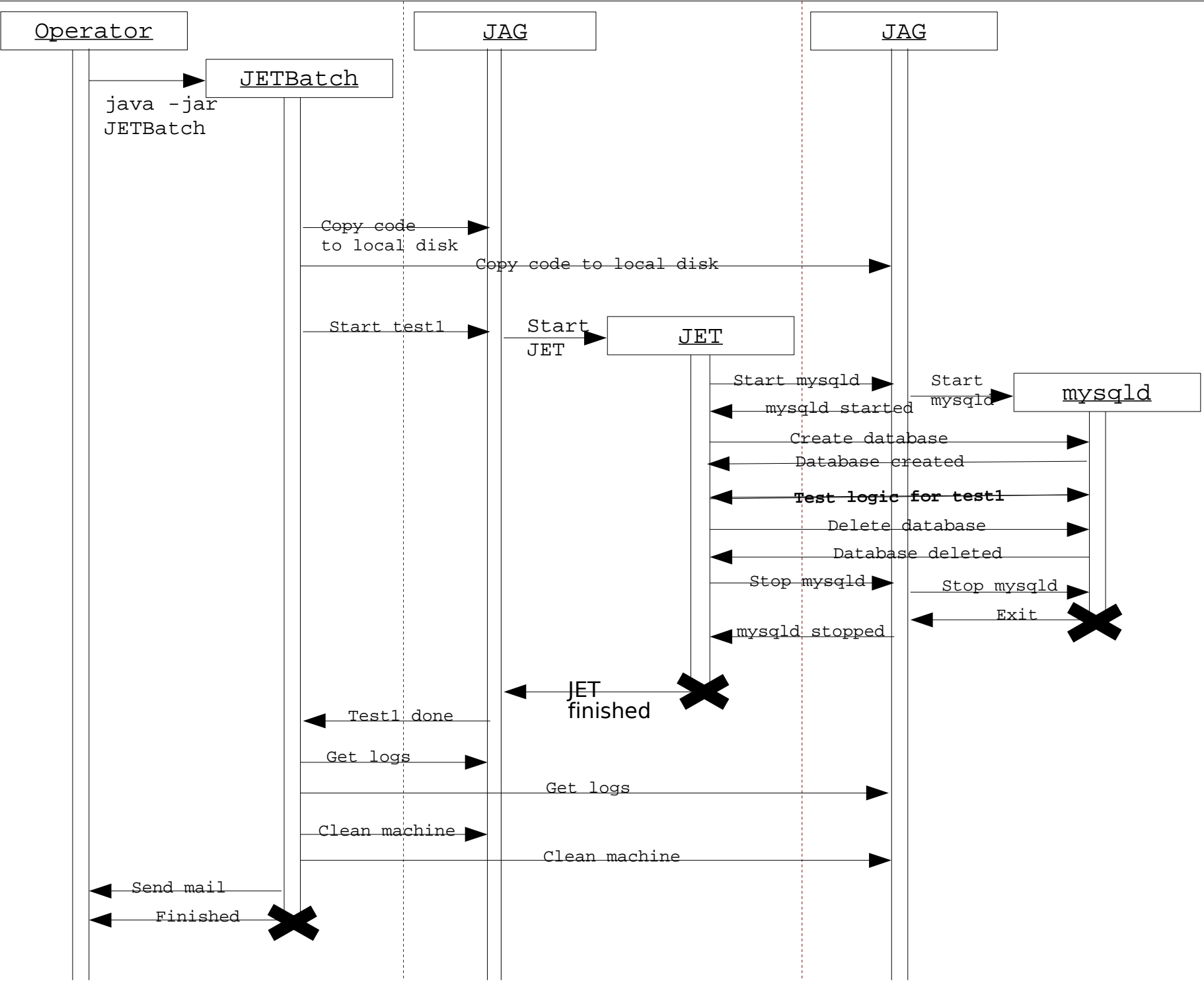
- Remote methods called like regular methods

```
JAGMachine jagma =  
    JAGHandles.createServerMachineFileClient(props,host);  
  
String filesep = jagma.getFileSeparator();  
  
String localtestlogpath = jagma.getLocalTestLogPath();  
  
log.fine("Testlogpath on slave " + host + "  
    "+localtestlogpath);  
  
jagma.unregister();
```

Developer Machine

Test Client Machines

Test Server Machines



Could JET be useful for you?

- Maybe, if:
 - > You want to test software distributed on several machines
 - > You want to write tests in Java
 - > You want to run the tests on different platforms
 - > You are willing to invest in your tests when writing them to get easier test maintenance in return

References

- `http://kenai.com/projects/jet`



JET

A Distributed Test Engine

Vemund Østgaard
MySQL System QA
Sun Microsystems, Inc.

