djigger

An open-source performance analysis solution

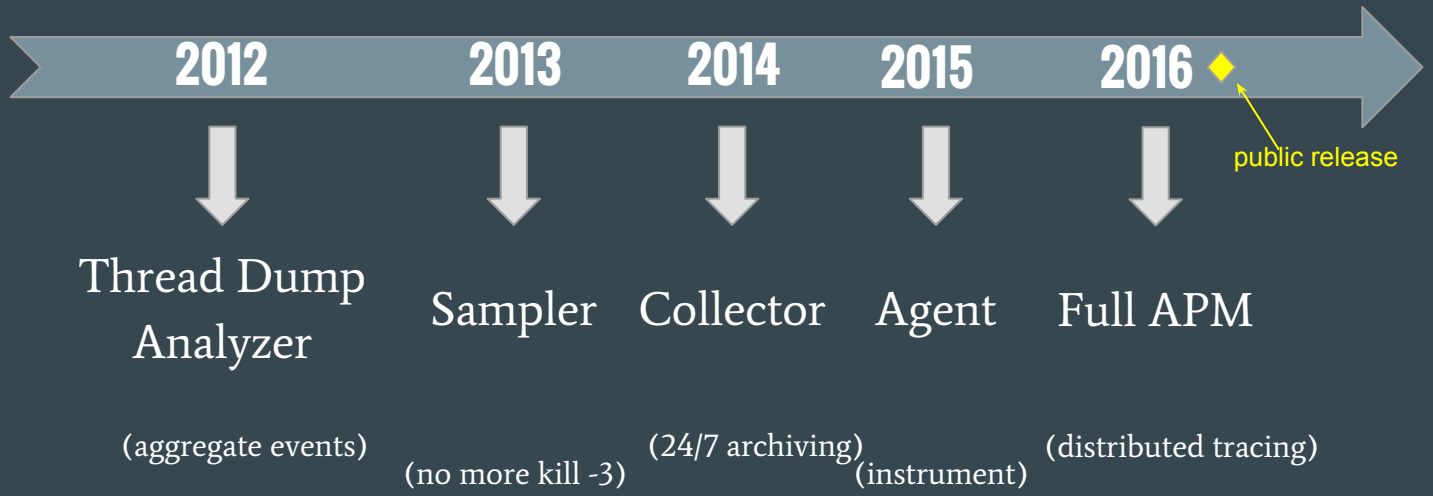
Context

2012

- Performance Testing & Analysis @ several companies
- Depending on project : often no tools or tools that can't be used
- Thread dumps are available : ***while (true) do kill -3 PID done***
- Analyzing thread dumps manually is a pain

➡ **Let's build our own thread dump analyzer !**

Development



➡ **~ 10 companies use djigger in France and Switzerland**

About performance analysis

…

My definition

Performance Analysis : gathering and interpreting **necessary** **&** **sufficient** data to understand and optimize a system or solve a performance problem.

Necessary and sufficient conditions

Performance Analysis : gathering and interpreting necessary & sufficient data to understand and optimize a system or solve a performance problem.

Necessary : without the necessary data, we can't understand nor solve the problem
Sufficient : runtimes are complex and we can't afford to harvest every detail

It's not just about tools

Performance Analysis : gathering and interpreting necessary & sufficient data to understand and optimize a system or solve a performance problem.

Necessary : without the necessary data, we can't understand nor solve the problem
Sufficient : runtimes are complex and we can't afford to harvest every detail

➡ Many factors affect our ability to do this correctly, <u>not **just** tooling</u>

Many factors are at play...

- Permissions & environment
- Monitoring maturity
- Knowledge of the stack
- Problem inputs

Many factors are at play...

Permissions & environment

Monitoring maturity

Knowledge of the stack

Problem inputs

who owns the code?
may I access the system?
may I change things?

Many factors are at play...

Permissions & environment

Monitoring maturity

Knowledge of the stack

Problem inputs

do we have proper tooling?
are all environments monitored?
do I have the necessary data?

Many factors are at play...

Permissions & environment

Monitoring maturity

Knowledge of the stack

Problem inputs

have I already seen this pattern?
are components closed/proprietary?
can I understand this runtime?

Many factors are at play...

Permissions & environment

Monitoring maturity

Knowledge of the stack

Problem inputs

what's the occurrence pattern?
what's the desired behaviour?
what are the actual symptoms?

About metrics

…

There's a ton of metrics out there

Pool usage

User CPU

Cache hit ratio

Memory

Logs

Kern CPU

Net I/O

AWR / v$

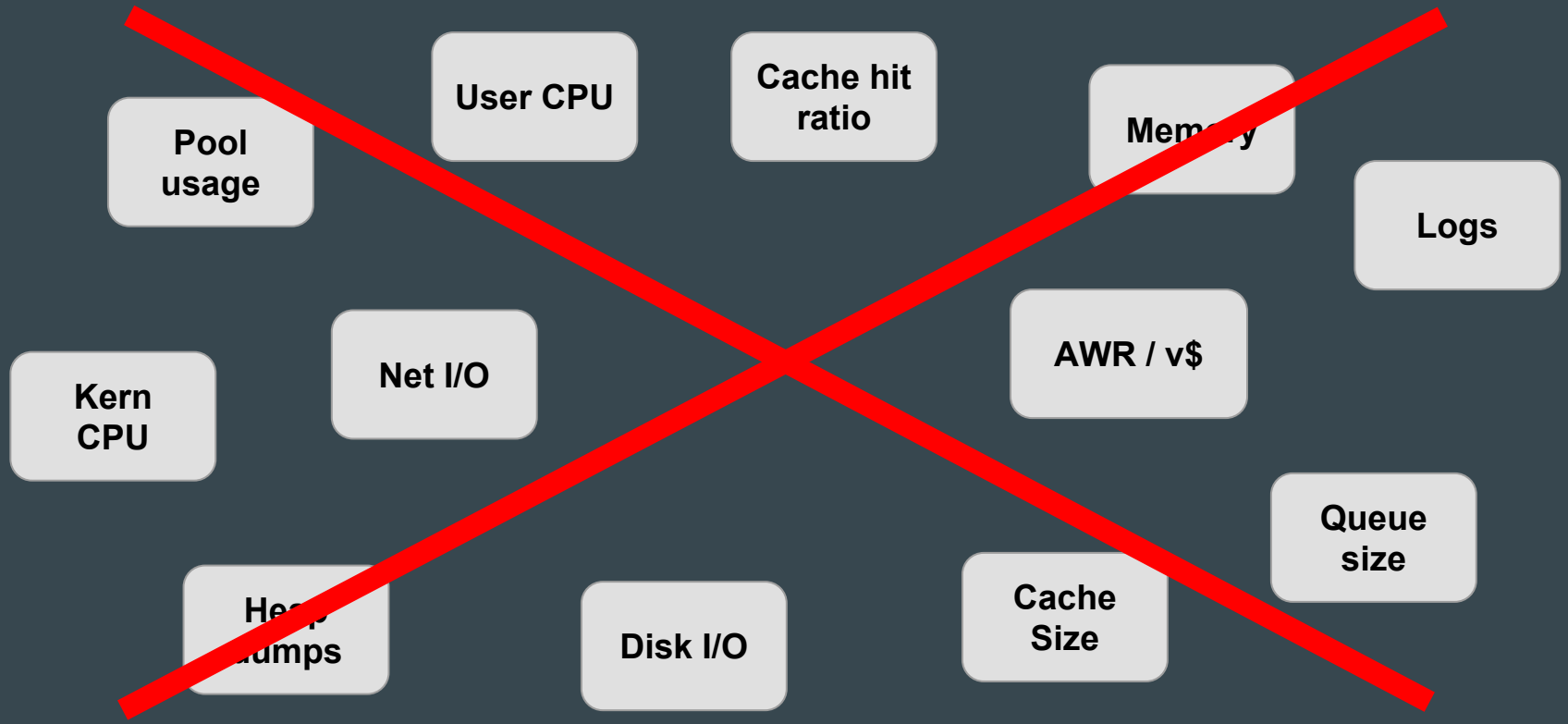
Queue size

Heap dumps

Disk I/O

Cache Size

I don’t play the elimination game (anymore)



Let’s look at what the program is doing

What are the main actors of a program’s execution?

Let’s look at what the program is doing

What are the main actors of a program’s execution?
Threads.

What’s the most important information about a thread?

Let's look at what the program is doing

What are the main actors of a program's execution?
Threads.

What's the most important information about a thread?
Its stack state (in particular, method calls).

..but what are java stacks blind to?

Let's look at what the program is doing

What are the main actors of a program's execution?
Threads.

What's the most important information about a thread?
Its stack state (in particular, method calls).

..but what are java stacks blind to?
GC pauses.

Look at what the program is doing

- I check thread stacks and GC overhead first.

Analysis process

…

A 3-step approach to analyzing latency issues

WHAT → WHERE → WHY

A 3-step approach to analyzing latency issues

WHAT

ex.: a servlet call

A 3-step approach to analyzing latency issues

WHAT → WHERE

ex.: a servlet call

ex.: time is spent in DB

A 3-step approach to analyzing latency issues

WHAT → WHERE → WHY

ex.: a servlet call

ex.: time is spent in DB

ex.: 1-n pattern and query can be cached

A 3-step approach to analyzing latency issues

WHAT → WHERE → WHY

Find out **which events** are problematic (transaction, method, click..)

ex.: a servlet call

Identify **top consumers** in the execution trees

ex.: time is spent in DB

Read stacks & object data to identify **faulty** or **optimizable behaviour**

ex.: 1-n pattern and query can be cached

Collecting events

sampling

instrumentation

Collecting events

sampling

Thread-dump events,
approximation of reality

instrumentation

Concrete measurements
and object capture

Collecting events

sampling

instrumentation

Thread-dump events,
approximation of reality

Concrete measurements
and object capture

without agent

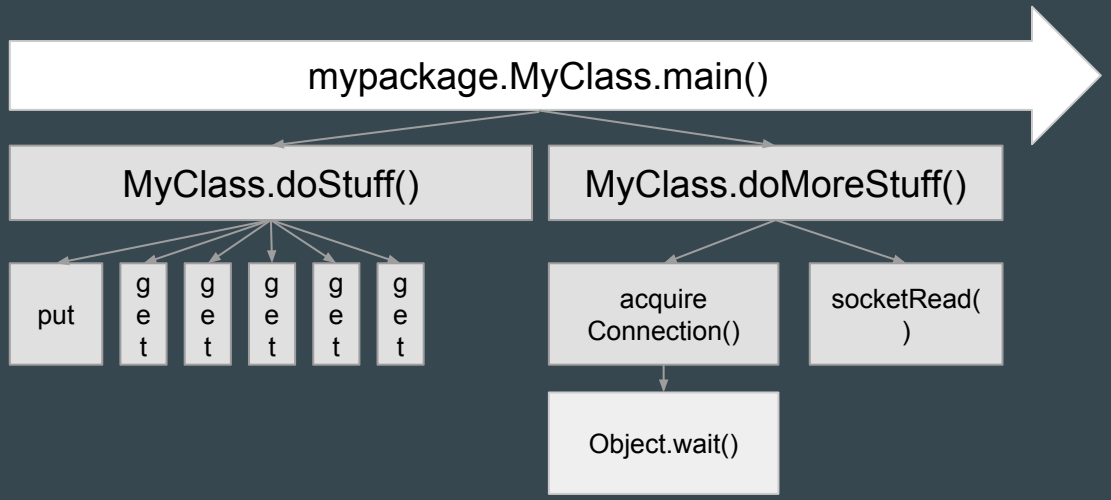
with agent

(for BCI)

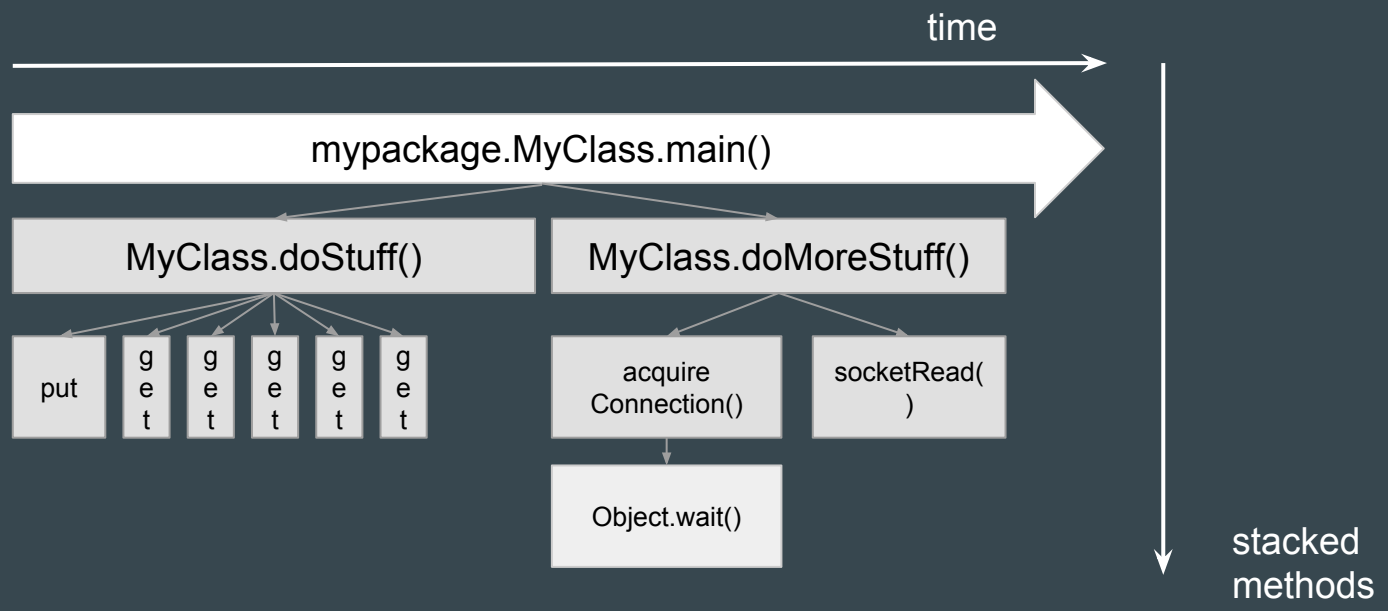
Stacktrace Sampling

...

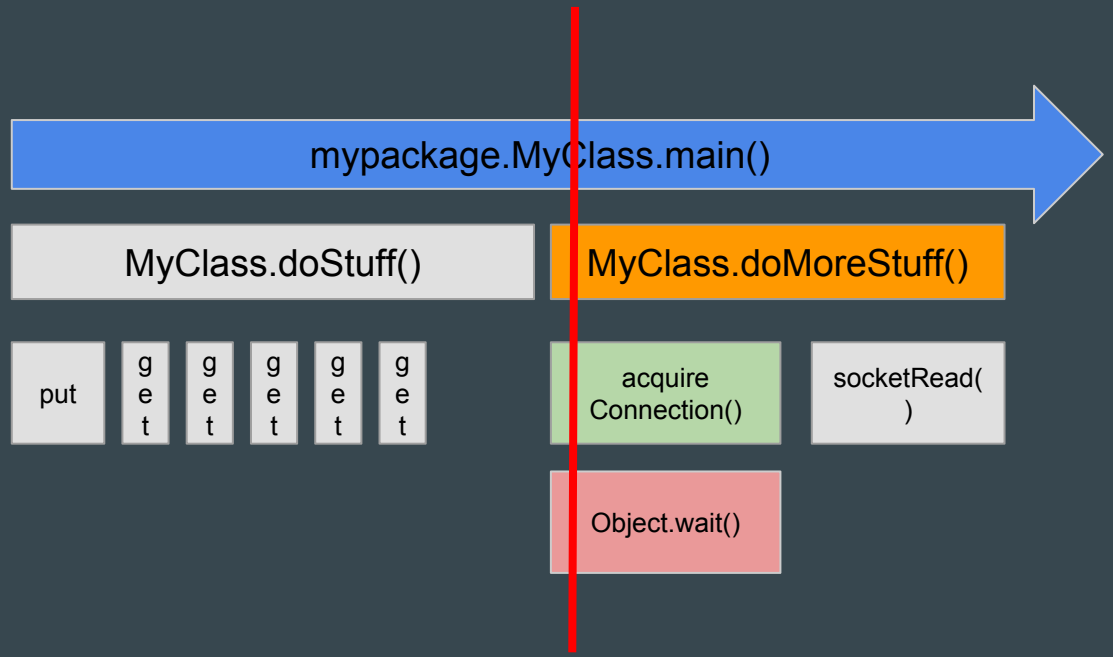
A dummy thread at runtime



A dummy thread at runtime

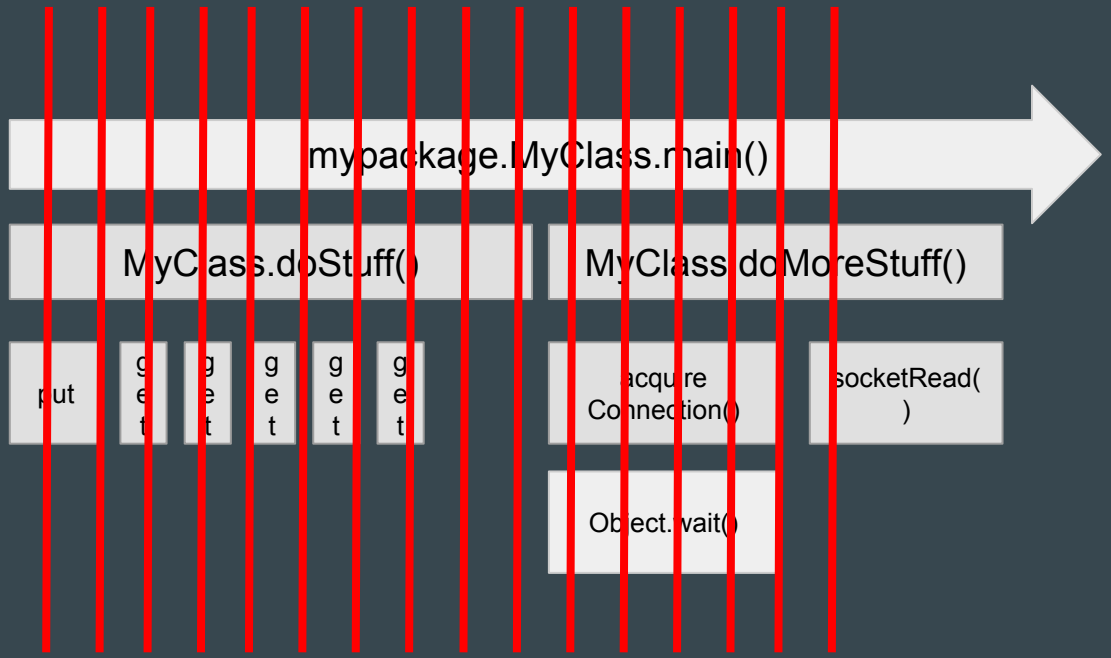


A random thread dump

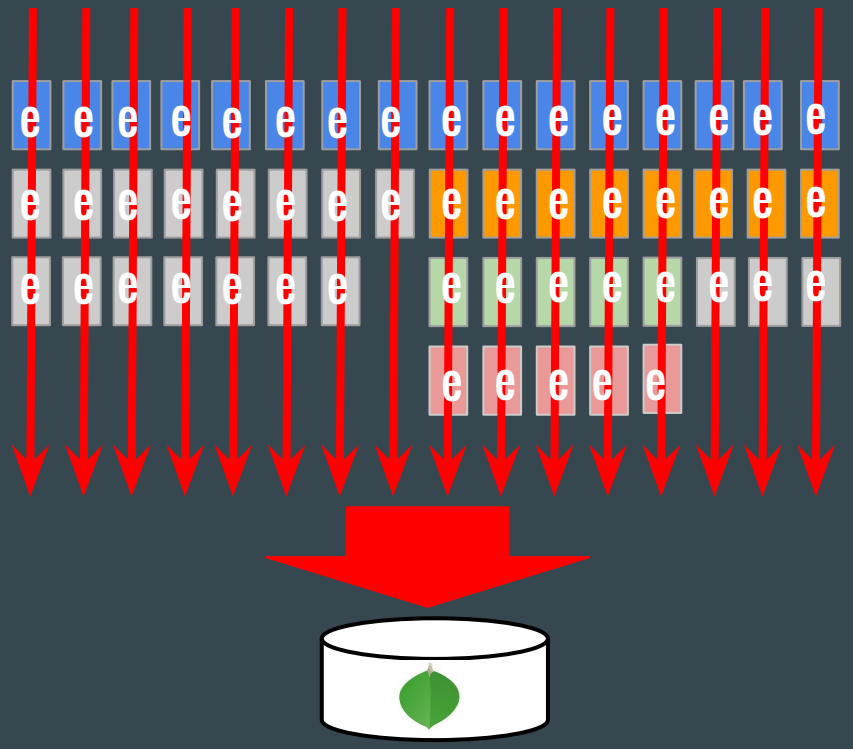



```
at    java.lang.Object.wait()
at    mypackage.datasource.acquireConnection()
at    mypackage.Myclass.doMoreStuff()
at    mypackage.MyClass.main()
```

Sampling = periodical thread dumps

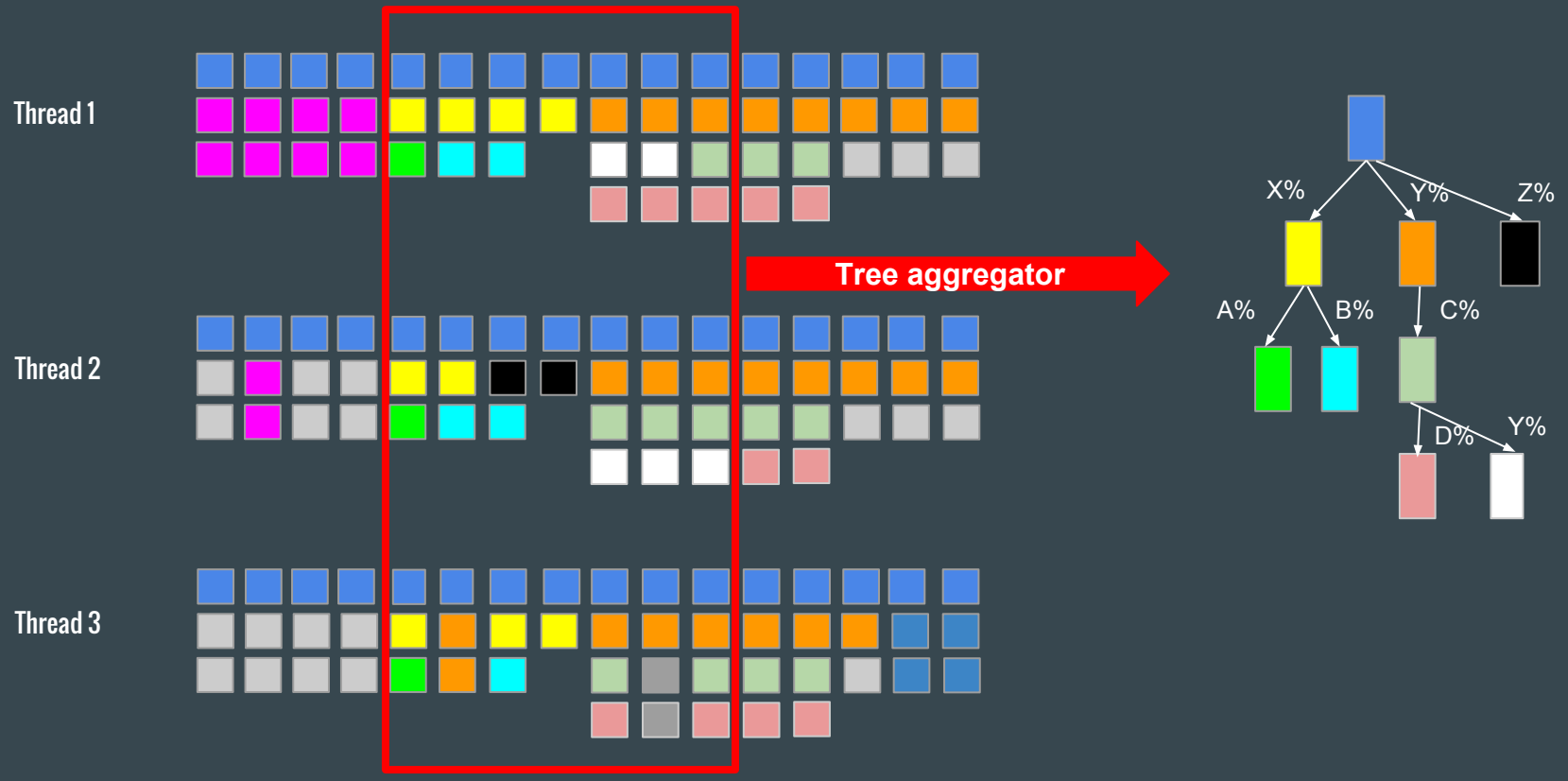


Time-based events

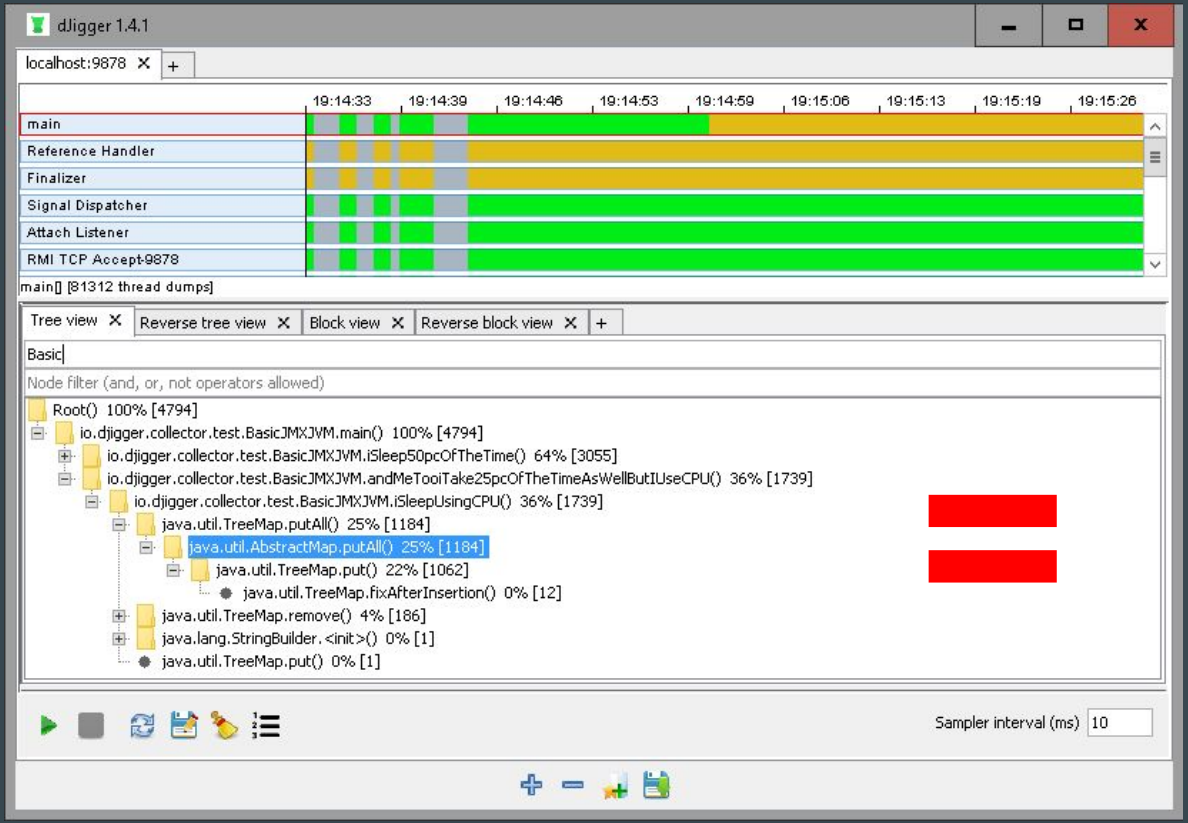


Time-based aggregation

Thread-based aggregation

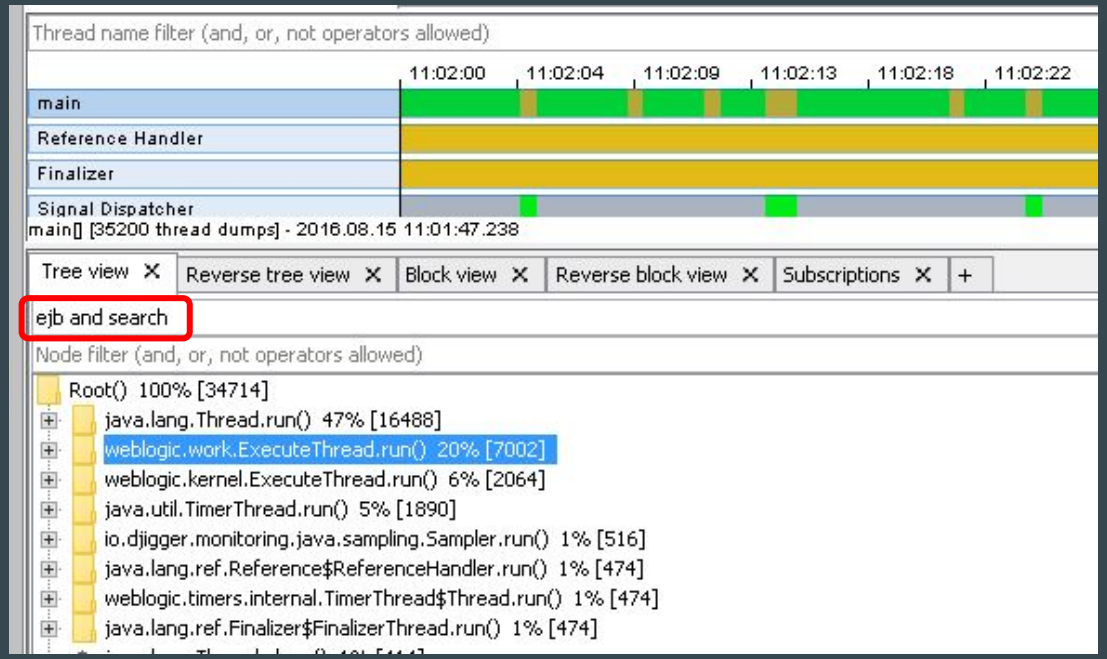


What does it look like in djigger?



3-step approach with sampling

Example


Thread name filter (and, or, not operators allowed)
11:02:00
11:02:04
11:02:09
11:02:13
11:02:18
11:02:22
main
Reference Handler
Finalizer
Signal Dispatcher
main[] [35200 thread dumps] - 2016.08.15 11:01:47.238
Tree view
Reverse tree view
Block view
Reverse block view
Subscriptions
ejb and search
Node filter (and, or, not operators allowed)
Root() 100% [34714]
java.lang.Thread.run() 47% [16488]
weblogic.work.ExecuteThread.run() 20% [7002]
weblogic.kernel.ExecuteThread.run() 6% [2064]
java.util.TimerThread.run() 5% [1890]
io.djigger.monitoring.java.sampling.Sampler.run() 1% [516]
java.lang.ref.Reference$ReferenceHandler.run() 1% [474]
weblogic.timers.internal.TimerThread$Thread.run() 1% [474]
java.lang.ref.Finalizer$FinalizerThread.run() 1% [474]

Example

Root() 100% [251]
weblogic.work.ExecuteThread.run() 100% [251]
weblogic.work.ExecuteThread.execute() 100% [251]
weblogic.rmi.internal.wls.WLSExecuteRequest.run() 100% [251]
weblogic.rmi.internal.BasicServerRef.handleRequest() 100% [251]
weblogic.security.service.SecurityManager.runAs() 100% [251]
weblogic.security.acl.internal.AuthenticatedSubject.doAs() 100% [251]
weblogic.rmi.internal.BasicServerRef$1.run() 100% [251]
weblogic.rmi.cluster.ClusterableServerRef.invoke() 100% [251]
weblogic.rmi.internal.BasicServerRef.invoke() 100% [251]
[redacted]Bean_289q9t_EOImpl.[redacted]_search[redacted]iste() 100% [251]
[redacted]Bean_289q9t_EOImpl.__WL_invoke() 100% [251]
[redacted]Bean.rem_[redacted]Searcher_search[redacted]) 100% [251]
[redacted]SearcherProxy.search[redacted]iste() 100% [251]
[redacted]SearcherImpl.search[redacted]) 100% [251]
[redacted]SearcherImpl.filterRowList() 100% [251]
[redacted]SearcherImpl.evaluate[redacted]ow() 100% [251]
[redacted]DBSessionImpl.execQuery() 99% [249]
[redacted]DAOBean.execQuery() 99% [249]
[redacted]dao.GenericStatementGenerator.query() 99% [248]
[redacted]dao.GenericStatementGenerator.query() 99% [248]
com.sun.proxy.$Proxy88.next() 91% [229]
[redacted]Handler.invoke() 91% [229]
java.lang.reflect.Method.invoke() 91% [228]
sun.reflect.DelegatingMethodAccessorImpl.invoke() 91% [228]
sun.reflect.GeneratedMethodAccessor174.invoke() 91% [228]
weblogic.jdbc.wrapper.ResultSet_oracle_jdbc_driver_OracleResultSetImpl.next() 91% [228]
oracle.jdbc.driver.OracleResultSetImpl.next() 91% [228]
oracle.jdbc.driver.OracleResultSetImpl.close_or_fetch_from_next() 91% [228]
oracle.jdbc.driver.T4CPreparedStatement.fetch() 91% [228]
oracle.jdbc.driver.T4CPreparedStatement.doOall8() 91% [228]
oracle.jdbc.driver.T4C8Oall.doOALL() 91% [228]
oracle.jdbc.driver.T4CTTIfun.doRPC() 91% [228]
oracle.jdbc.driver.T4CTTIfun.receive() 91% [228]
oracle.jdbc.driver.T4CMAREngine.unmarshalUB1() 90% [227]
oracle.jdbc.driver.T4CSocketInputStreamWrapper.read() 90% [227]
oracle.jdbc.driver.T4CSocketInputStreamWrapper.readNextPacket() 90% [227]
oracle.net.ns.NetInputStream.read() 90% [226]
oracle.net.ns.NetInputStream.read() 90% [226]
oracle.net.ns.NetInputStream.read() 90% [226]
oracle.net.ns.NetInputStream.getNextPacket() 90% [226]
oracle.net.ns.DataPacket.receive() 68% [171]
oracle.net.ns.NetOutputStream.flush() 22% [55]
oracle.jdbc.driver.T4CSocketOutputStreamWrapper.flush() 0% [1]
oracle.jdbc.driver.T4C8Oall.readBVC() 0% [1]

Example

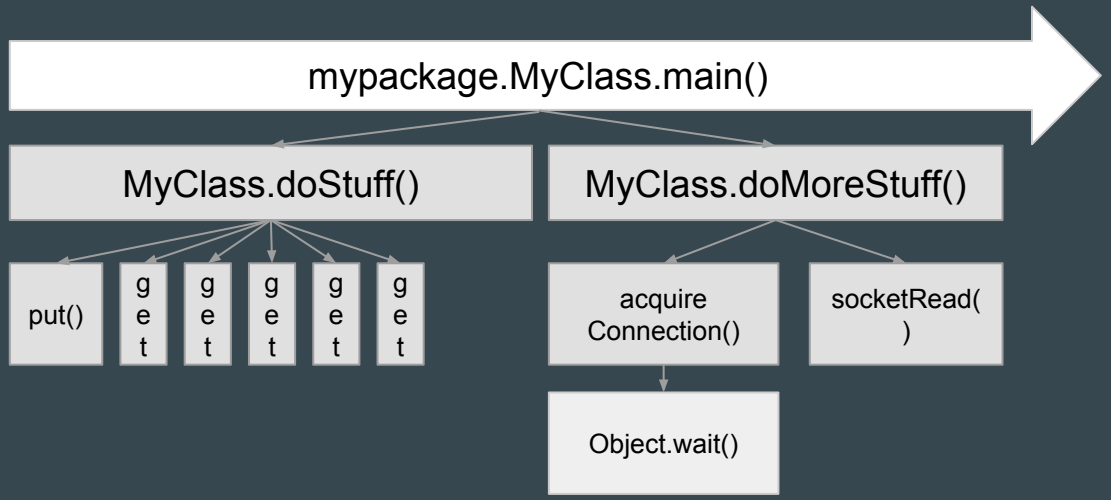
Root() 100% [251]
weblogic.work.ExecuteThread.run() 100% [251]
weblogic.work.ExecuteThread.execute() 100% [251]
weblogic.rmi.internal.wls.WLSExecuteRequest.run() 100% [251]
weblogic.rmi.internal.BasicServerRef.handleRequest() 100% [251]
weblogic.security.service.SecurityManager.runAs() 100% [251]
weblogic.security.acl.internal.AuthenticatedSubject.doAs() 100% [251]
weblogic.rmi.internal.BasicServerRef$1.run() 100% [251]
weblogic.rmi.cluster.ClusterableServerRef.invoke() 100% [251]
weblogic.rmi.internal.BasicServerRef.invoke() 100% [251]
[redacted]Bean_289q9t_EOImpl_WLSkel.invoke() 100% [251]
[redacted]Bean_289q9t_EOImpl.[redacted]_search[redacted]iste() 100% [251]
weblogic.ejb.container.internal.SessionRemoteMethodInvoker.invoke() 100% [251]
[redacted]Bean_289q9t_EOImpl.__WL_invoke() 100% [251]
[redacted]Bean.rem_[redacted]Searcher_search[redacted]) 100% [251]
[redacted]SearcherProxy.search[redacted]iste() 100% [251]
[redacted]SearcherImpl.search[redacted]) 100% [251]
[redacted]SearcherImpl.filterRowList() 100% [251]
[redacted]SearcherImpl.evaluate[redacted]ow() 100% [251]
[redacted]DBSessionImpl.execQuery() 99% [249]
[redacted]DAOBean.execQuery() 99% [249]
[redacted]dao.GenericStatementGenerator.query() 99% [248]
[redacted]dao.GenericStatementGenerator.query() 99% [248]
com.sun.proxy.$Proxy88.next() 91% [229]
[redacted]Handler.invoke() 91% [229]
java.lang.reflect.Method.invoke() 91% [228]
sun.reflect.DelegatingMethodAccessorImpl.invoke() 91% [228]
weblogic.jdbc.wrapper.ResultSet_oracle_jdbc_driver_OracleResultSetImpl.next() 91% [228]
oracle.jdbc.driver.OracleResultSetImpl.next() 91% [228]
oracle.jdbc.driver.OracleResultSetImpl.close_or_fetch_from_next() 91% [228]
oracle.jdbc.driver.T4CPreparedStatement.fetch() 91% [228]
oracle.jdbc.driver.T4CPreparedStatement.doOall8() 91% [228]
oracle.jdbc.driver.T4C[redacted]all.doOALL() 91% [228]
oracle.jdbc.driver.[redacted]CTTIfun.doRPC() 91% [228]
oracle.jdbc.dri[redacted].T4CTTIfun.receive() 91% [228]
oracle.jdbc[redacted]iver.T4CMAREngine.unmarshalUB1() 90% [227]
oracle.[redacted]c.driver.T4CSocketInputStreamWrapper.read() 90% [227]
ora[redacted].jdbc.driver.T4CSocketInputStreamWrapper.readNextPacket() 90% [227]
[redacted]acle.net.ns.NetInputStream.read() 90% [226]
[redacted]oracle.net.ns.NetInputStream.read() 90% [226]
oracle.net.ns.NetInputStream.read() 90% [226]
oracle.net.ns.NetInputStream.getNextPacket() 90% [226]
oracle.net.ns.DataPacket.receive() 68% [171]
oracle.net.ns.NetOutputStream.flush() 22% [55]
oracle.jdbc.driver.T4CSocketOutputStreamWrapper.flush() 0% [1]
oracle.jdbc.driver.T4C8Oall.readBVC() 0% [1]

Instrumentation

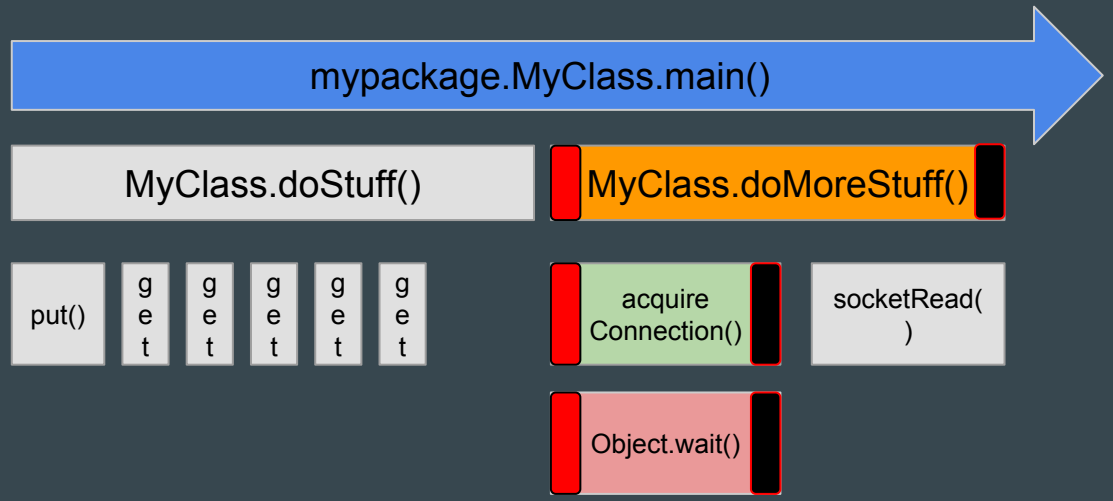
…

|⟷|

A dummy thread at runtime (again)



Subscriptions

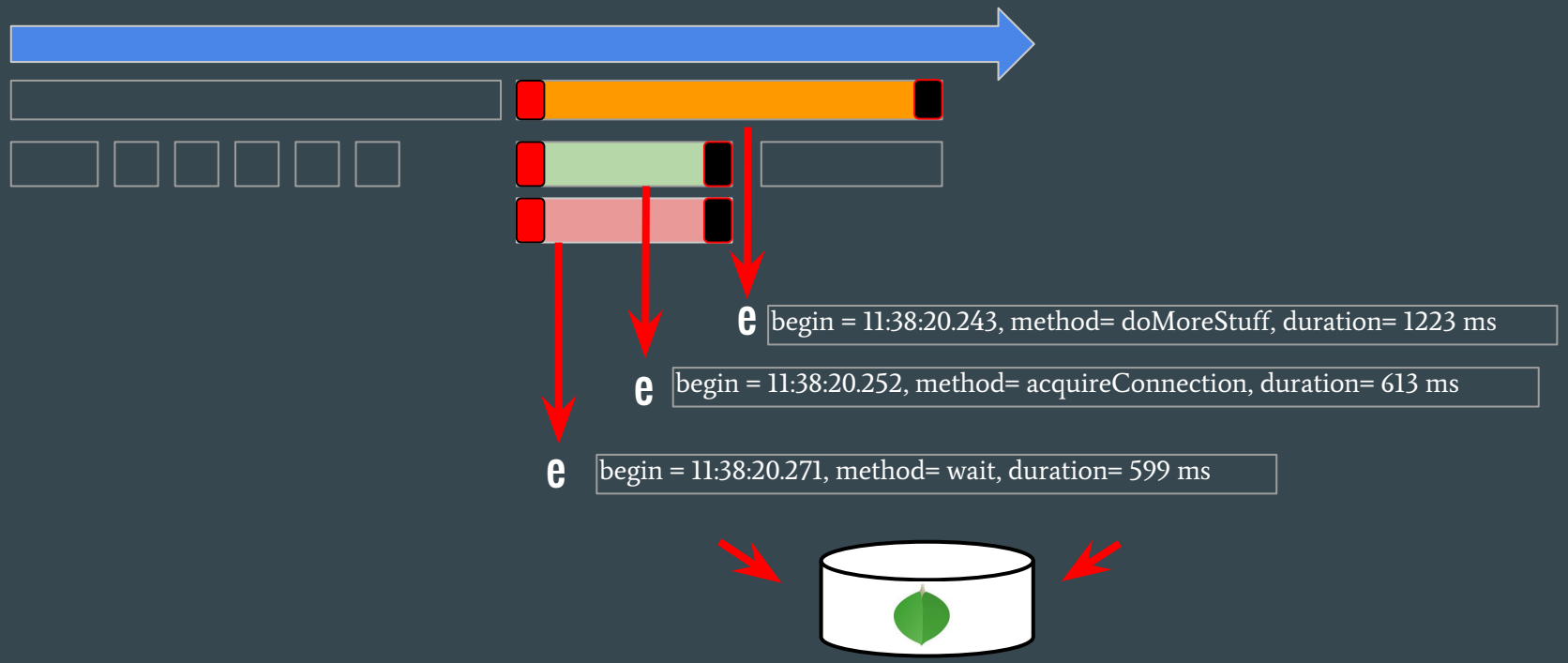


Active subscriptions:

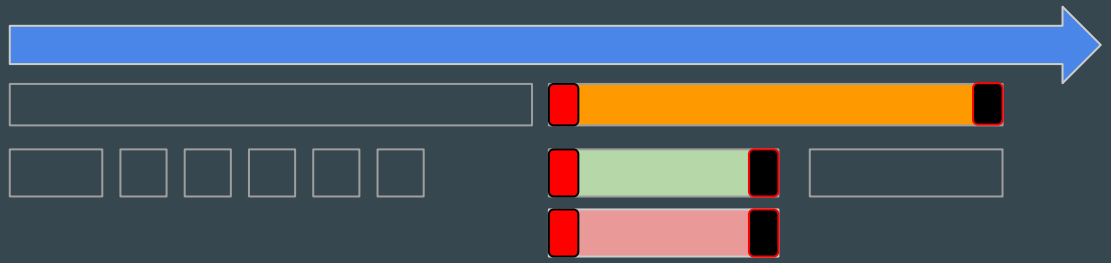
Start event:

End event:

Subscription-based events



Transaction flags



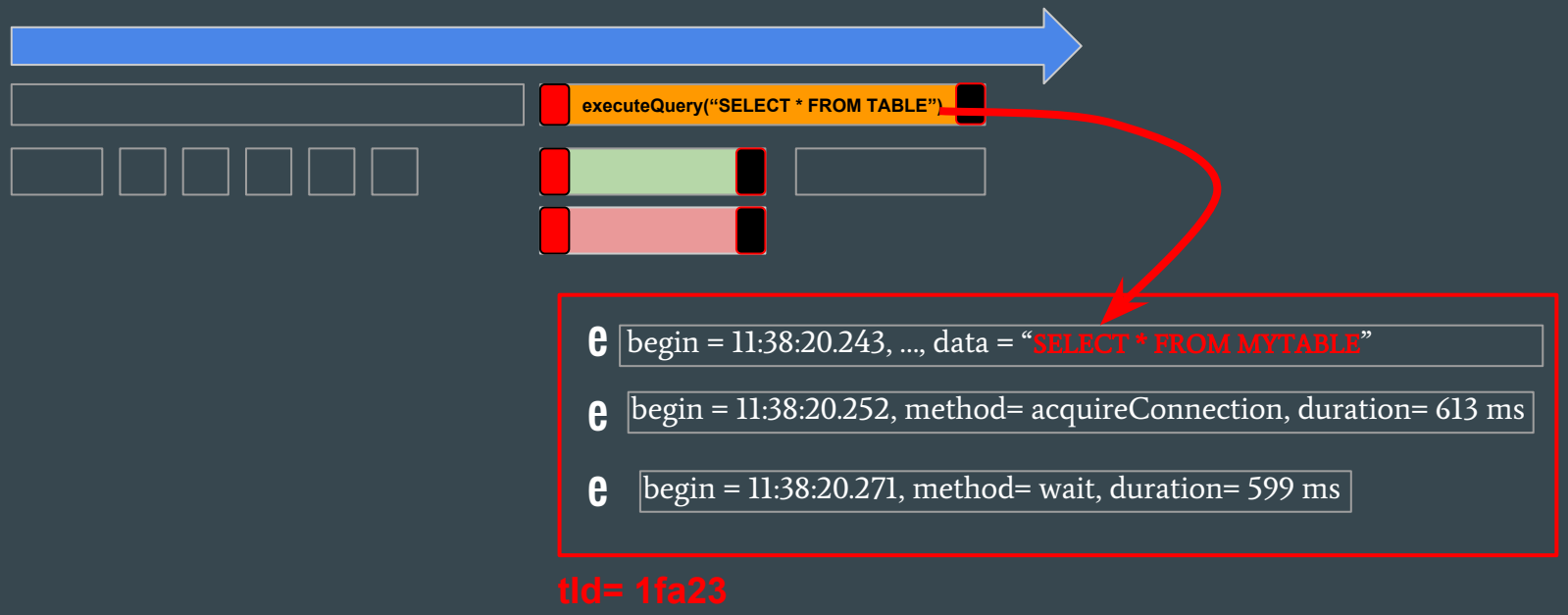
e begin = 11:38:20.243, method= doMoreStuff, duration= 1223 ms

e begin = 11:38:20.252, method= acquireConnection, duration= 613 ms

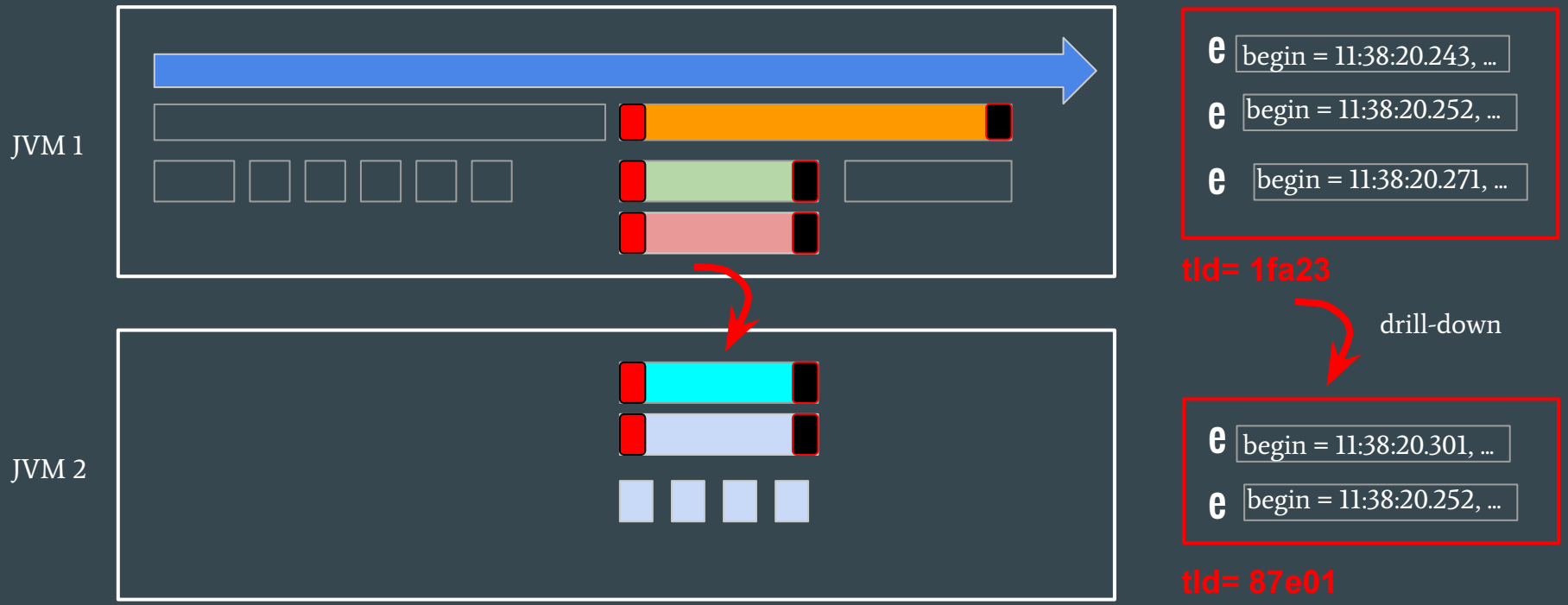
e begin = 11:38:20.271, method= wait, duration= 599 ms

tId= 1fa23

Object capture

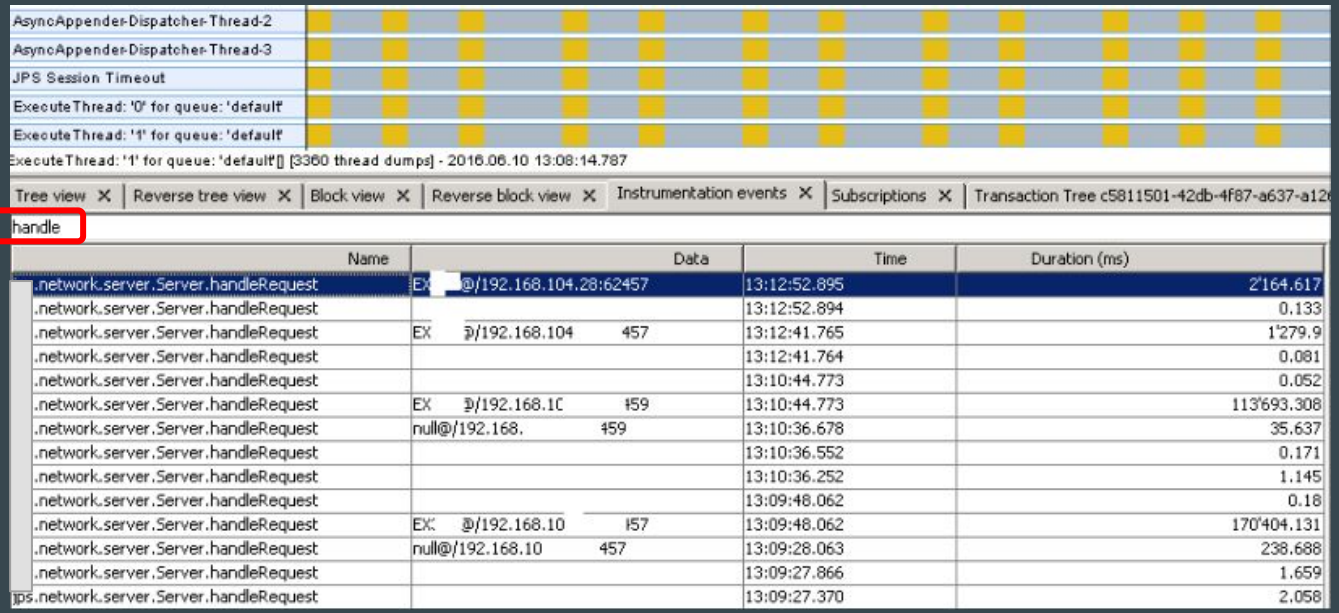


Distributed transactions



3-step approach with instrumentation

What does it look like in djigger?



Name	Data	Time	Duration (ms)
.network.server.Server.handleRequest	EX @/192.168.104.28:62457	13:12:52.895	2'164.617
.network.server.Server.handleRequest		13:12:52.894	0.133
.network.server.Server.handleRequest	EX @/192.168.104 457	13:12:41.765	1'279.9
.network.server.Server.handleRequest		13:12:41.764	0.081
.network.server.Server.handleRequest		13:10:44.773	0.052
.network.server.Server.handleRequest	EX @/192.168.1(459	13:10:44.773	113'693.308
.network.server.Server.handleRequest	null@/192.168. 459	13:10:36.678	35.637
.network.server.Server.handleRequest		13:10:36.552	0.171
.network.server.Server.handleRequest		13:10:36.252	1.145
.network.server.Server.handleRequest		13:09:48.062	0.18
.network.server.Server.handleRequest	EX @/192.168.10 457	13:09:48.062	170'404.131
.network.server.Server.handleRequest	null@/192.168.10 457	13:09:28.063	238.688
.network.server.Server.handleRequest		13:09:27.866	1.659
jps.network.server.Server.handleRequest		13:09:27.370	2.058

What does it look like in djigger?

Tree view | Reverse tree view | Block view | Reverse block view | Instrumentation events | Subscriptions | Transaction Tree c5

Stacktrace filter (and, or, not operators allowed)

Node filter (and, or, not operators allowed)

- [redacted].network.server.Server.handleRequest() - 113693ms
  - weblogic.rmi.cluster.ClusterableRemoteRef.invoke() - 110526
    - weblogic.rmi.cluster.ClusterableR[illegible]ms
  - weblogic.rmi.cluster.ClusterableRemo[illegible]
  - weblogic.rmi.cluster.ClusterableRemo[illegible]
  - weblogic.rmi.cluster.ClusterableRemo[illegible]
  - weblogic.rmi.cluster.ClusterableRemo[illegible]
  - weblogic.rmi.cluster.ClusterableRemo[illegible]
  - weblogic.rmi.cluster.ClusterableRemoteRef.invoke() - 3ms
  - weblogic.rmi.cluster.ClusterableRemoteRef.invoke() - 39ms
  - weblogic.rmi.cluster.ClusterableRemoteRef.invoke() - 4ms
  - weblogic.rmi.cluster.ClusterableRemoteRef.invoke() - 2ms

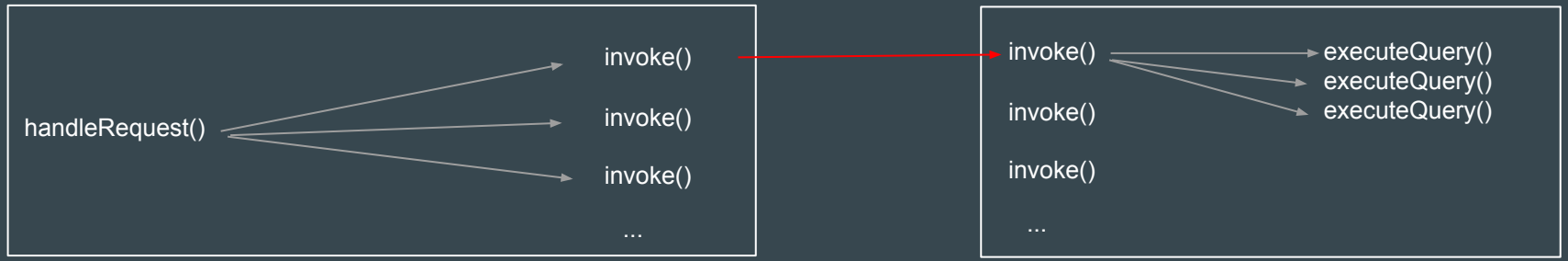
Drill-down
Expand all
Expand first branch
Collapse all

Tree view | Reverse tree view | Block view | Reverse block vie

Stacktrace filter (and, or, not operators allowed)

Node filter (and, or, not operators allowed)

- weblogic.rmi.cluster.ClusterableServerRef.invoke() - 110502ms
  - oracle.jdbc.driver.PhysicalConnection.prepareStatement() - 0ms
  - com.sun.proxy.$Proxy142.prepareStatement() - 0ms
  - com.sun.proxy.$Proxy90.executeQuery() - 14ms
  - com.sun.proxy.$Proxy142.prepareStatement() - 0ms
  - com.sun.proxy.$Proxy90.executeQuery() - 0ms
  - com.sun.proxy.$Proxy142.prepareStatement() - 0ms
  - com.sun.proxy.$Proxy86.prepareStatement() - 0ms
  - com.sun.proxy.$Proxy90.executeQuery() - 1ms
  - com.sun.proxy.$Proxy86.prepareStatement() - 0ms
  - com.sun.proxy.$Proxy90.executeQuery() - 2ms
  - com.sun.proxy.$Proxy86.prepareStatement() - 0ms
  - com.sun.proxy.$Proxy90.executeQuery() - 1ms

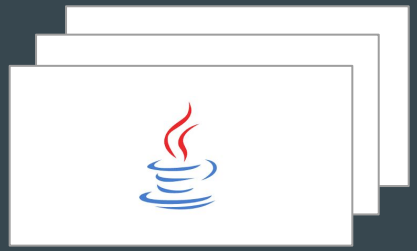


What does it look like in djigger?

Name	Data	Duration (ms)
com.sun.proxy.$Proxy90.executeQuery	SELECT cast(systimestamp as timestamp) from dual	0,638
oracle.jdbc.driver.OraclePreparedStatementWra...	SELECT 1 FROM DUAL	0,508
oracle.jdbc.driver.OraclePreparedStatementWra...	SELECT PACKAGENAME FROM RELEASE WH...	1,133
com.sun.proxy.$Proxy90.executeQuery	SELECT PACKAGENAME FROM RELEASE WH...	1,163
oracle.jdbc.driver.OraclePreparedStatementWra...	SELECT 1 FROM DUAL	0,793
com.sun.proxy.$Proxy90.executeQuery	SELECT /*n8BZpQJnr3OLqd+6D b79...	1,241
com.sun.proxy.$Proxy90.executeQuery	SELECT /*a7MTiLKfes3hkmOhJc b130...	0,91
com.sun.proxy.$Proxy90.executeQuery	SELECT DISTINCT target.I	0,65
com.sun.proxy.$Proxy90.executeQuery	SELECT	0,561
com.sun.proxy.$Proxy90.executeQuery	SELECT /*a7MTiLKfes3hkmOh / b130...	1,253
oracle.jdbc.driver.OraclePreparedStatementWra...	SELECT 1 FROM DUAL	0,324
com.sun.proxy.$Proxy90.executeQuery	SELECT cast(systimestamp as timestamp) from dual	0,463
oracle.jdbc.driver.OraclePreparedStatementWra...	SELECT 1 FROM DUAL	0,536
com.sun.proxy.$Proxy90.executeQuery	SELECT /*nKT1sBdK9 / b4...	0,552
oracle.jdbc.driver.OraclePreparedStatementWra...	SELECT 1 FROM DUAL	0,586
com.sun.proxy.$Proxy90.executeQuery	SELECT /*nKT1sBdK96 b4...	0,767
oracle.jdbc.driver.OraclePreparedStatementWra...	SELECT 1 FROM DUAL	0,341
com.sun.proxy.$Proxy90.executeQuery	SELECT /*+ FIRST_ROWS(1) uNiOZivPU6gsUTv...	0,854
oracle.jdbc.driver.OraclePreparedStatementWra...	SELECT 1 FROM DUAL	0,621
oracle.jdbc.driver.OraclePreparedStatementWra...	SELECT /*n8BZpQJnr 79...	1,822
com.sun.proxy.$Proxy90.executeQuery	SELECT /*n8BZpQJnr3C 79...	1,855

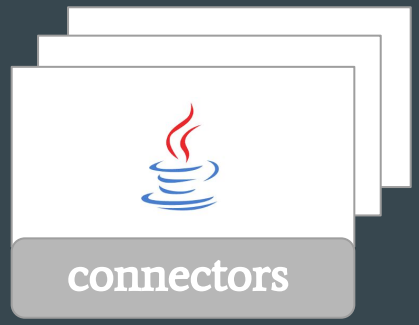
Component overview

…



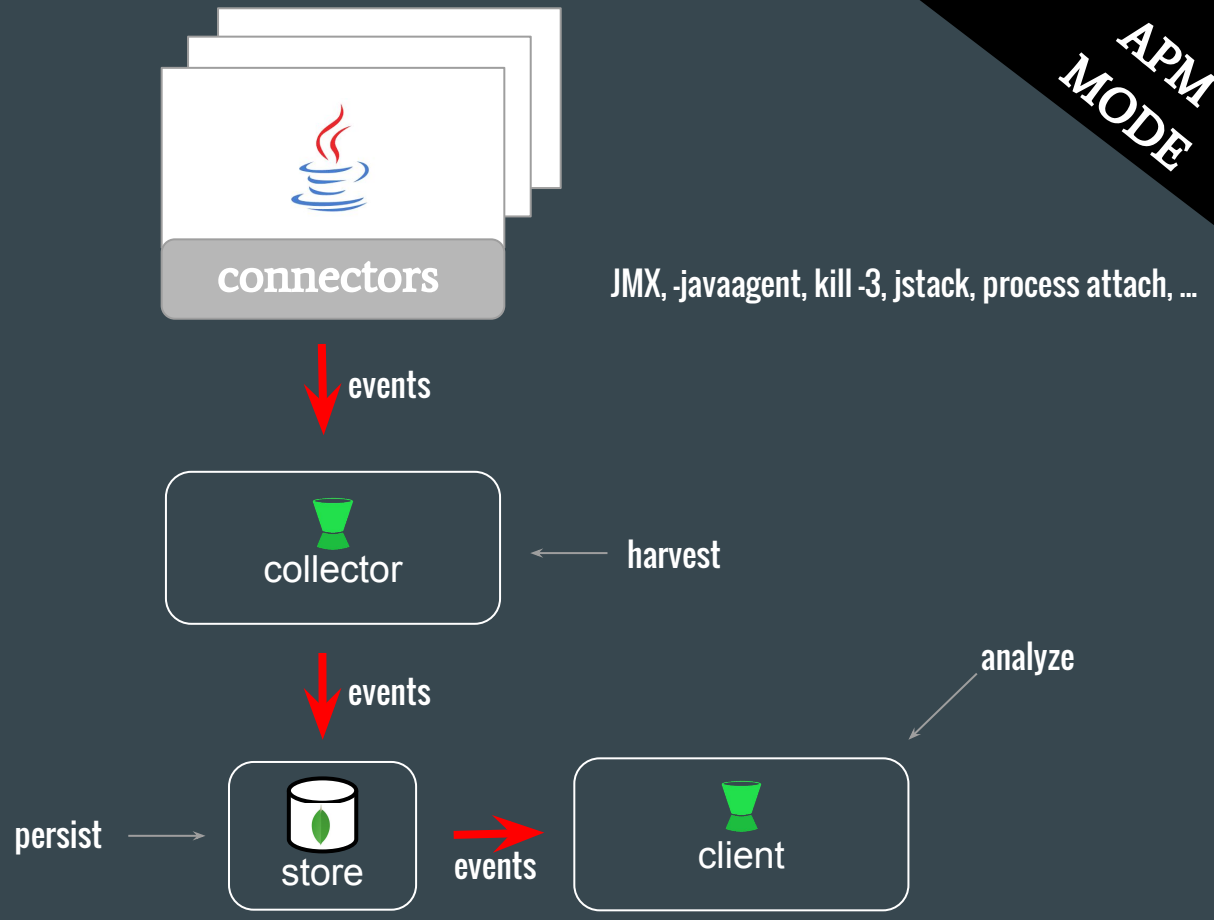
JMX, -javaagent, kill -3, jstack, process attach, ...

PROFILER MODE



JMX, -javaagent, kill -3, jstack, process attach, ...

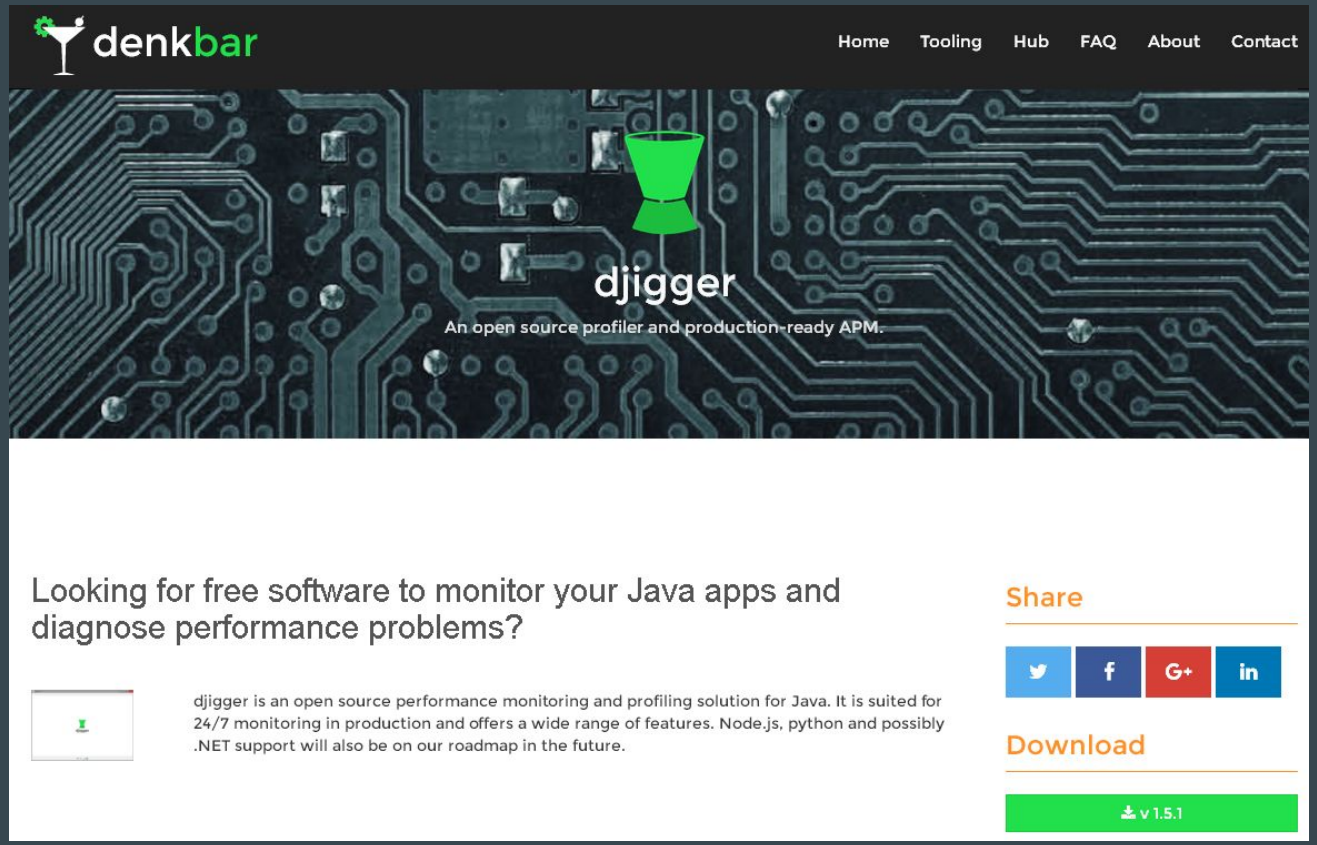
harvest & analyze


APM MODE
connectors
JMX, -javaagent, kill -3, jstack, process attach, ...
events
collector
harvest
events
analyze
persist
store
events
client

Download and try out djigger !

...

Download djigger at http://denkbar.io



Thanks for your attention

…