Actor Bug: Concurrency - Creation

The actor model has actors being called and created via parent actors and while this process is integral to the actor model it can lead to errors if improperly implemented. These bugs represent the errors that occur as a result of this actor creation process.

Kamon had difficulties with running multiple actor systems within a single cluster. The actor systems could be passed around and in doing so one would be created in another system and leave behind a null pointer error in it's wake. This was a result of accessing the actor systems directly and was solved by utilizing a dispatcher to trade actor systems and only access those present within a system.

The trouble was solved by adding code to the DispatcherTracing class that would handle a situation by more stringently pairing a Dispatcher to an ActorSystem instead of relying on the default case.

Link: https://github.com/kamon-io/Kamon/issues/38

Code:

instrumentation/DispatcherTracing.scala

```
...
 class DispatcherTracing {

-  private[this] var actorSystem: ActorSystemImpl = _
+  @Pointcut("execution(akka.dispatch.Dispatchers.new(..)) && this(dispatchers) &&
cflow(execution(akka.actor.ActorSystemImpl.new(..)) && this(system))")
+  def onActorSystemStartup(dispatchers: Dispatchers, system: ActorSystemImpl) = {}

-  @Pointcut("execution(akka.actor.ActorSystemImpl.new(..)) && this(system)")
-  def onActorSystemStartup(system: ActorSystemImpl) = {}
+  @Before("onActorSystemStartup(dispatchers, system)")
+  def beforeActorSystemStartup(dispatchers: Dispatchers, system: ActorSystemImpl): Unit = {
+    val currentDispatchers = dispatchers.asInstanceOf[DispatchersWithActorSystem]
+    currentDispatchers.actorSystem = system
+  }
+
+  @Pointcut("execution(* akka.dispatch.Dispatchers.lookup(..)) && this(dispatchers)")
+  def onDispatchersLookup(dispatchers: Dispatchers) = {}
+
+  @AfterReturning(pointcut = "onDispatchersLookup(dispatchers)", returning = "dispatcher")
+  def afterReturningLookup(dispatchers: Dispatchers, dispatcher: Dispatcher): Unit = {
```

```
+    val dispatchersWithActorSystem = dispatchers.asInstanceOf[DispatchersWithActorSystem]
+    val dispatcherWithMetrics = dispatcher.asInstanceOf[DispatcherMessageMetrics]

-  @Before("onActorSystemStartup(system)")
-  def beforeActorSystemStartup(system: ActorSystemImpl): Unit = {
-    actorSystem = system
+    dispatcherWithMetrics.actorSystem = dispatchersWithActorSystem.actorSystem
   }
...
+  var actorSystem: ActorSystemImpl = _
+}
+
+trait DispatchersWithActorSystem {
+  var actorSystem: ActorSystemImpl = _
 }
...
```