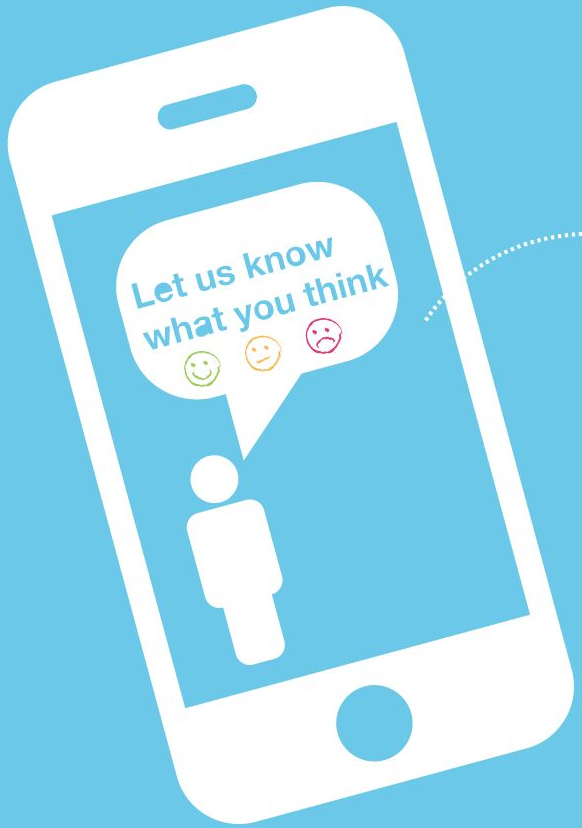
Monitoring Reactive Applications

Duncan DeVore (@ironfish)
Henrik Engstrom (@h3nk3)
Software Engineers at Lightbend



Please use the
Scala Days app
to rate sessions.

Agenda

- What is Monitoring?
- Traditional Monitoring
- Architecture Shift
- Monitoring Reactive Applications
- Trends in Monitoring
- Best Practice
- Lightbend Monitoring

What is Monitoring?

“to be aware of the state of a system, to observe a situation for any changes which may occur over time, using a monitor or measuring device of some sort” - WikiPedia

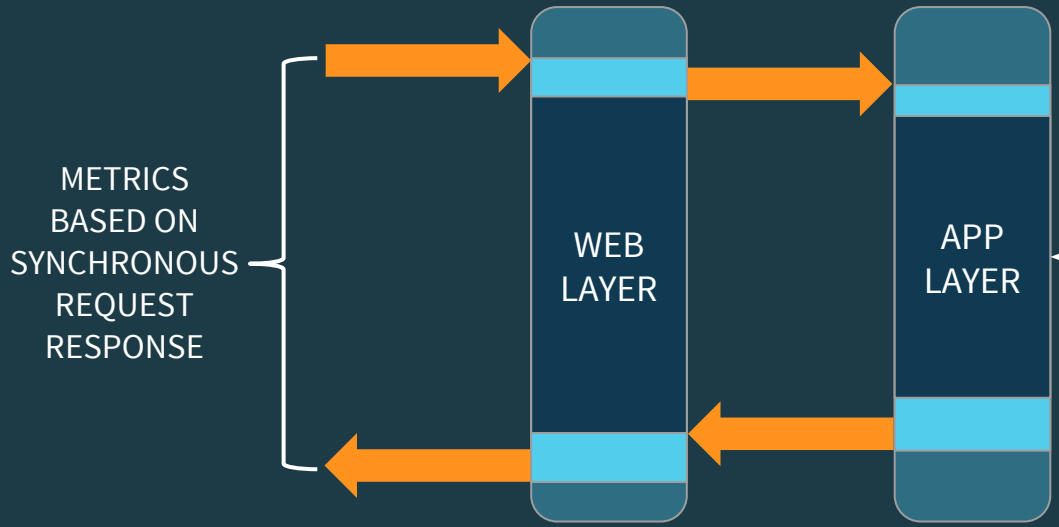
Monitoring Categories

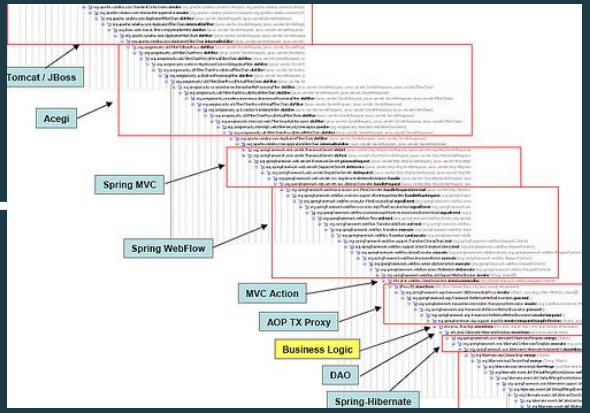
- **Business Process Monitoring**
  - Focus on business process level
- **Functional Monitoring**
  - Focus on use-case level
- **Technical Monitoring**
  - Focus on individual pieces

Types of Monitoring

- System Monitoring
- Tracing/Span/Transaction Monitoring
- Performance Monitoring
- Integration Monitoring
- Application Monitoring

Traditional Monitoring: Metrics and Debugging





Architecture Shift : The Internet

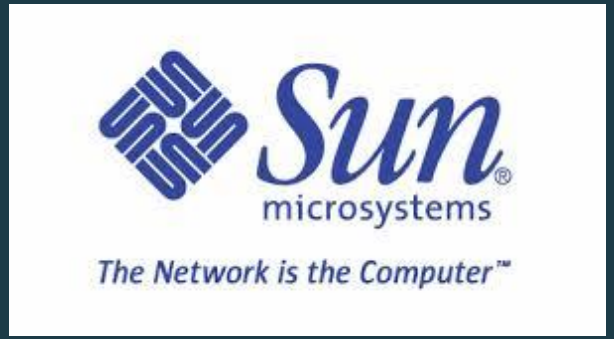
- The **Internet** has been one of the most significant inventions in the last 50 years
- Designed to allow access to **data** from **anywhere** in the world
- Started as research to build a **robust, fault-tolerant** network
- A **global** interconnected network of computers
- Commissioned by the US under J.C.R. Licklider
- By **ARPA**, what we know today as **DARPA**

Architecture Shift : Distributed Computing

- Supporting this concept **required** a new computing model
- Backed by **Big Iron** and midrange hardware like AS/400
- The **distributed computing** model was born
- Still **educational, military** and **government** focus
- Then in 1990 **ARPANET** was decommissioned
- By 1995 the Internet was fully **commissioned** in the US
- This significant event opened the **doors**

Architecture Shift : Cloud Computing

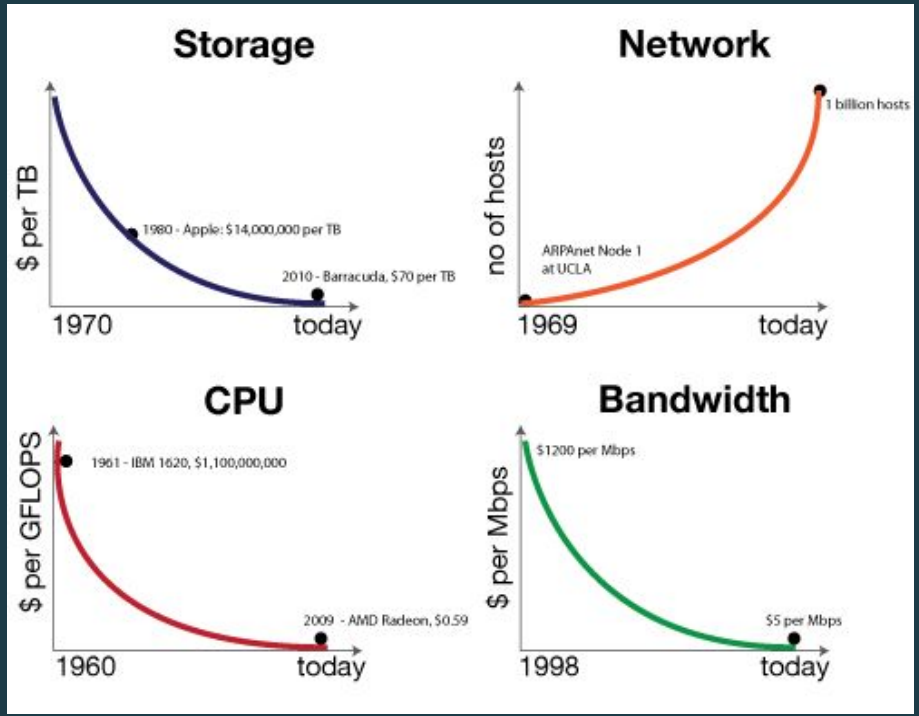
- The result became **cloud computing**
- Distributed computing focused on the **technical**
- While cloud computing focused on the **economics**
- The difference may seem inconsequential
- But nothing could be further from the **truth**
- The **economics** shift changed the **face** of business



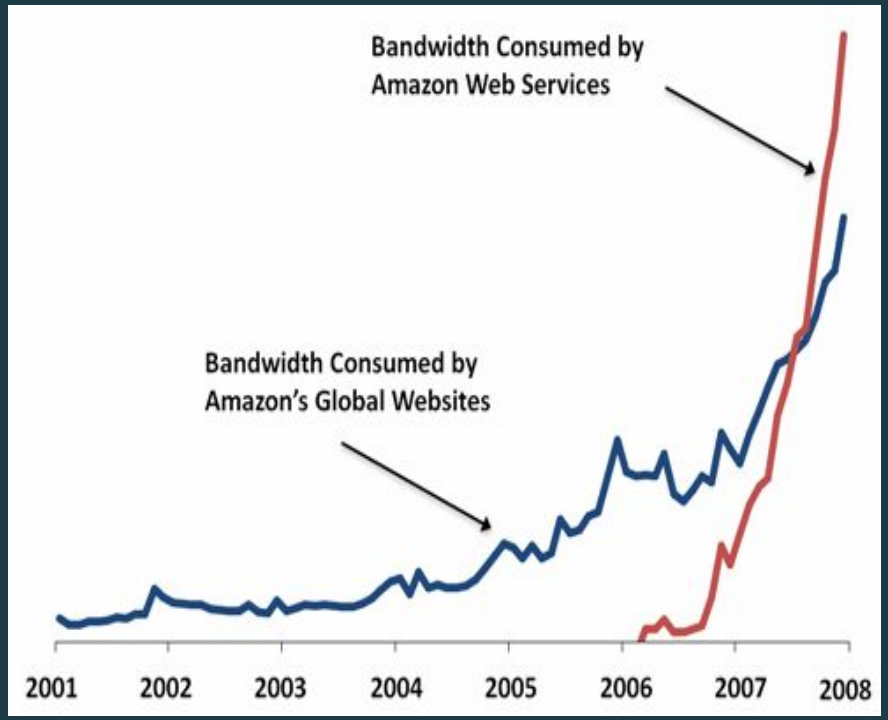
Architecture Shift : The Real World

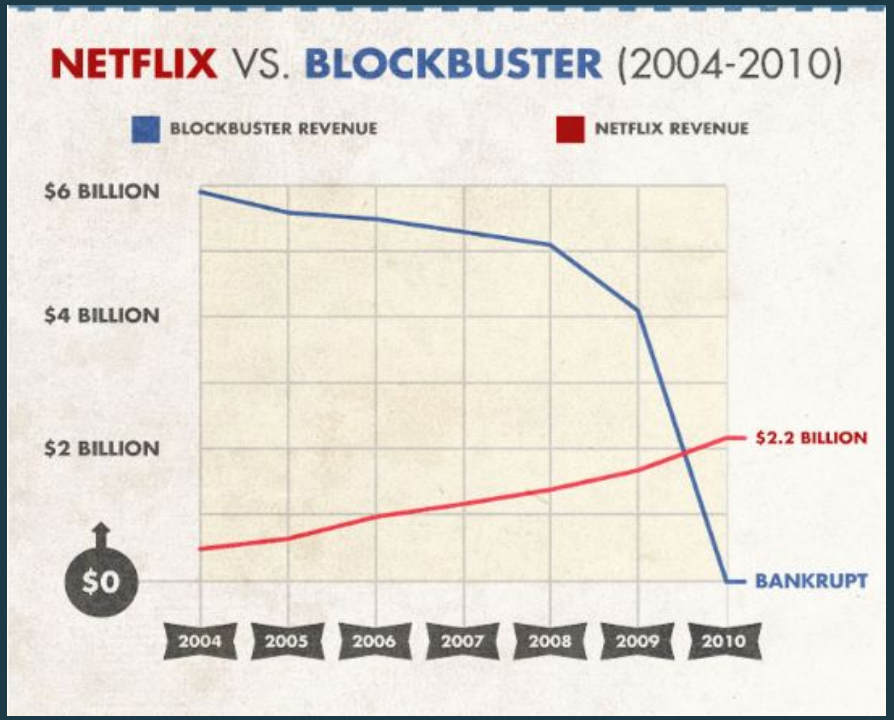
- **Reducing** storage costs
- **Reducing** CPU costs
- **Reducing** Bandwidth costs
- **Increasing** Network traffic

The Business landscape was **totally** reshaped.

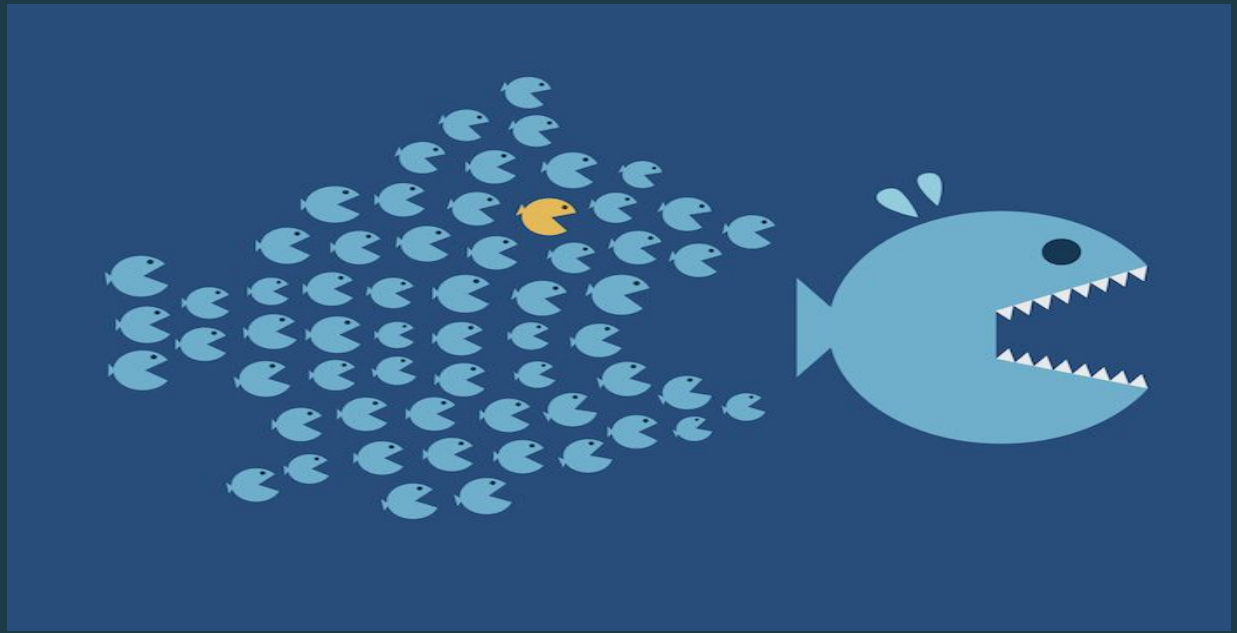


Architecture Shift : Business Impact





Architecture Shift : The Result



In the new world, it is not the big fish which eats the small fish, it’s the **fast fish** which eats the **slow fish** -- *Klaus Schwab*

Architecture Shift : Enter Reactive

Architecture Shift : Enter Reactive

- **Responsive**: reacting in a desired or positive way; quick to react or respond
- **Elastic**: return to original shape/size after being stretched, squeezed, etc.;
- **Resilient**: become strong, or successful again after something bad happens
- **Message**: recorded communication sent to a recipient who cannot be contacted
- **Driven**: operated, moved, or controlled by a specified person or source of power

-- Merriam Webster Dictionary

Architecture Shift : Freedom for all!

Finally, asynchronous, parallel, distributed computing that works for the common person

Distributed Computing Disclaimer: The availability and accuracy of the information delivered by these services and information they portray are contingent on a distributed computing environment comprised of several unique asynchronous, parallel and non-deterministic applications within the said companies multi-clustered system. As a result, the availability, accuracy and access to these services and the data they provide, may or may not be available at any given time. Additionally, the availability, precision and access to dynamic data provided on these multi-clustered services may or may not be dependent on other systems being available at the time of use. The user is cautioned that these services: may or may not be available at any time; may become unavailable without notice; for any reason and for any length of time. Why? Well, because the stack traces produced are pretty much useless, and the supporting development staff must fall back on print-line statements to isoloate the cause of said error.

Monitoring Reactive Applications: The Challenge

Asynchronous and Distributed => traditional ways of monitoring no longer valid...

```
[info]  at cinnamon.sample.failure.B$$anonfun$receive$2.applyOrElse(FailureDemo.scala:102)
[info]  at akka.actor.Actor$class.aroundReceive(Actor.scala:467)
[info]  at cinnamon.sample.failure.B.aroundReceive(FailureDemo.scala:86)
[info]  at akka.actor.ActorCell.receiveMessage(ActorCell.scala:516)
[info]  at akka.actor.ActorCell.invoke(ActorCell.scala)
[info]  at akka.dispatch.Mailbox.processMailbox(Mailbox.scala:238)
[info]  at akka.dispatch.Mailbox.run$$original(Mailbox.scala:220)
[info]  at akka.dispatch.Mailbox.run(Mailbox.scala:29)
[info]  at akka.dispatch.ForkJoinExecutorConfigurator$AkkaForkJoinTask.exec(AbstractDispatcher.scala:397)
[info]  at scala.concurrent.forkjoin.ForkJoinTask.doExec(ForkJoinTask.java:260)
[info]  at scala.concurrent.forkjoin.ForkJoinPool$WorkQueue.runTask(ForkJoinPool.java:1339)
[info]  at scala.concurrent.forkjoin.ForkJoinPool.runWorker(ForkJoinPool.java:1979)
[info]  at scala.concurrent.forkjoin.ForkJoinWorkerThread.run(ForkJoinWorkerThread.java:107)
```

Monitoring Reactive Applications: The Challenge

Monitoring Reactive Applications: The Challenge

Asynchronous and Distributed means:

- The path to where you are is no longer available
- You don’t know where your code will execute
- The epitome of non-determinism

What to do??

Monitoring Reactive Applications: Tracing

- Traces follow asynchronous message flows
- Create a unique entry point
- Flows through multiple actors via message sends
- Create a unique ending point
- The result is a trace span or “hot path”
- From this we can:
  - Gather metrics on time-to-complete
  - Derive latency metrics
  - Establish a base for flow visualization

Monitoring Reactive Applications: Tracing

```
import akka.actor.{ Actor, Props }
import com.lightbend.cinnamon.akka.Tracer

class ActorA extends Actor {
  val actorB = context.actorOf(Props[ActorB], "b")
  def receive = {
    case "begin" =>
      Tracer(context.system).start("myTraceName") {
        actorB forward "doWork"
      }
  }
}
```

Lightbend

Monitoring Reactive Applications: Tracing

```
class ActorB extends Actor {
  def receive = {
    case "doWork" =>
      doSomeWork()
      Tracer(context.system).end("myTraceName")
  }

  def doSomeWork(): Unit = ???
}
```

Output:

```
[metrics-logger-reporter-1-thread-1] INFO  output-sink - type=HISTOGRAM, name=traces.myTraceName.
trace-histogram, count=109, min=266598801, max=4344502206, mean=2.132377393674398E9, stddev=1.
3304223746691678E9, median=1.84076406E9, p75=3.410501492E9, p95=4.157100469E9, p98=4.276913089E9,
p99=4.283057143E9, p999=4.344502206E9
```

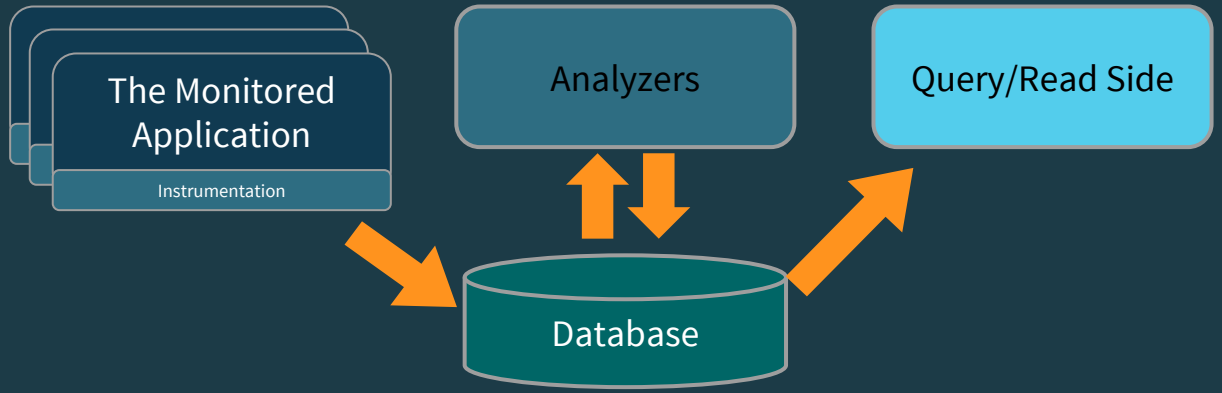
Monitoring Reactive Applications: Tracing

Trace flow: wouldn't it be nice?

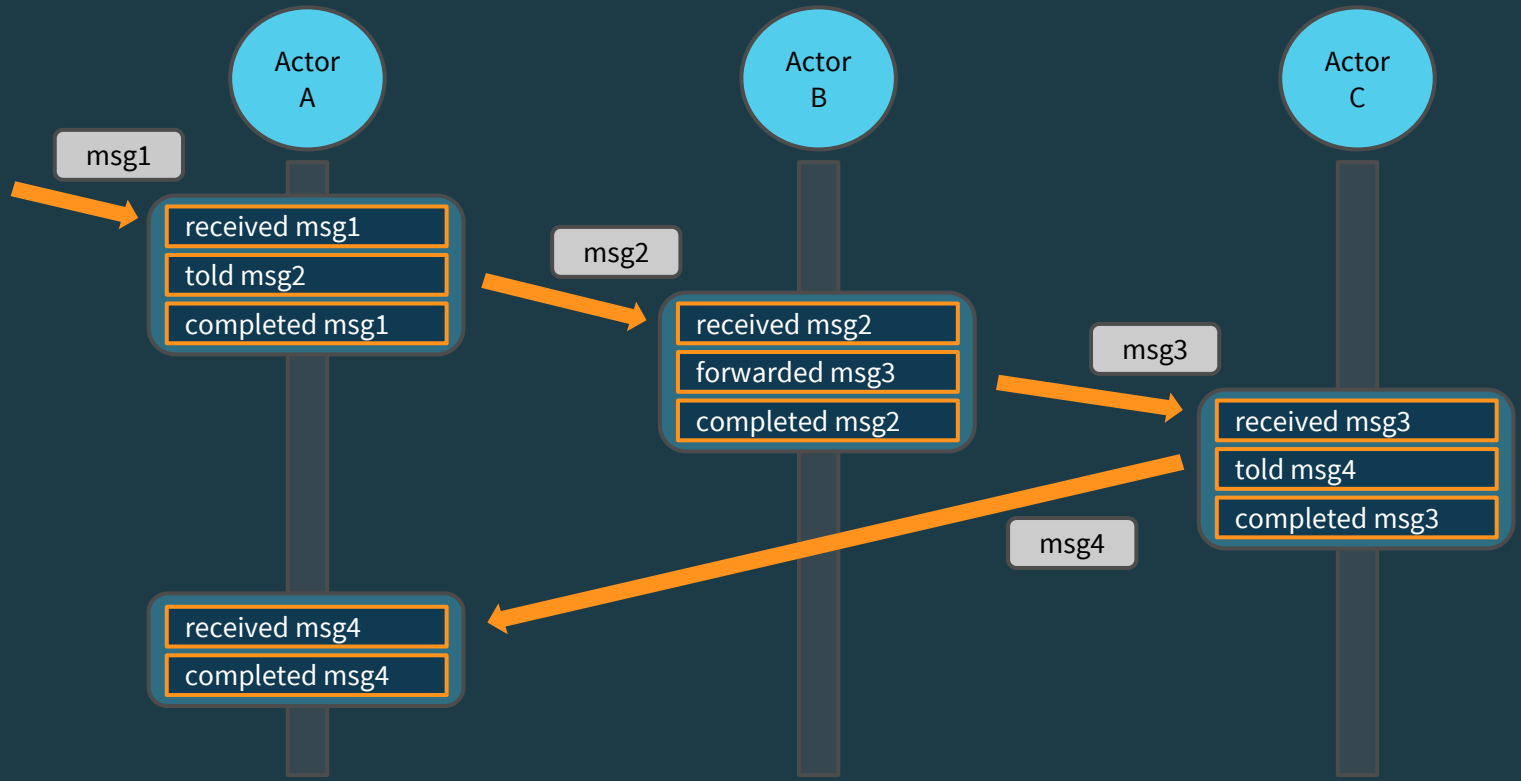
- `/user/a  =>  /user/b  - initiate`
- `/user/b  =>  /user/b/c1  - data`
- `/user/b  =>  /user/b/c2  - data`
- `/user/b  =>  /user/b/c3  - data`
- `/user/c1  =>  /user/b  - answer-1`
- `/user/c3  =>  /user/b  - answer-3`
- `/user/c2  =>  /user/b  - answer-2`
- `/user/b  =>  /user/a  - composed-answer`

Monitoring Reactive Applications: Dapper

- Dapper paper from Google in 2010 - Highly Influential
- Typesafe's implementation of the Dapper paper was called Typesafe Console (2012 - 2013)



Monitoring Reactive Applications: Trace Info.



Monitoring Reactive Applications: Trace Events

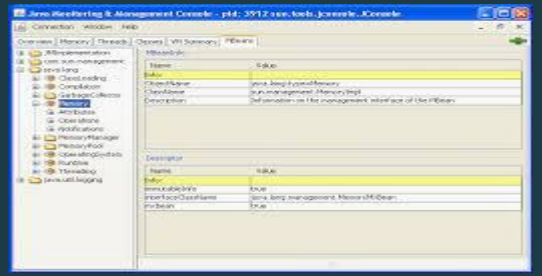
```
"id" : "4RHh8WFt3T6JkhKx9rj2qQ==",
"trace" : "4RHh8ZFx2z6JkhKx9rj2qQ==",
"parent" : "4RHh8UAf3T6JkhKx9rj2qQ==",
"host" : "192.168.99.100",
"nanoTime" : "13462485522467120000",
"annotation" : {
    "type" : "received",
    "actor" : "akka://SomeActorSystem/user/ActorA"
}
```

Monitoring Reactive Applications: Lightbend Monitoring

The Monitored Application — Instr. & Metrics Events Traces

The Monitored Application — Instr. & Metrics Events Traces

The Monitored Application — Instr. & Metrics Events Traces



Monitoring Reactive Applications: Associated Cost

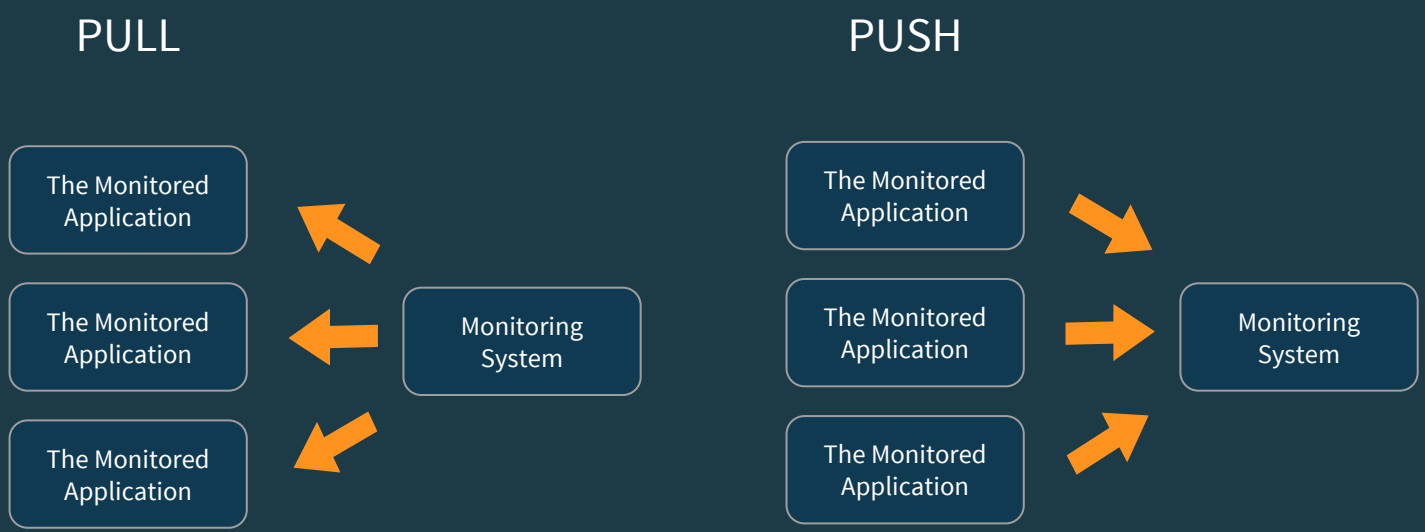
- Each application has its own set of characteristics
- Which Akka application is more costly to monitor?
  - Application with few actors with lots of business logic
  - Application with several temporary, short-lived, actors
- Cost associated with each “event” in the application
- Performance overhead is therefore **very difficult** to predict!

Monitoring Reactive Applications: Cost Mitigation

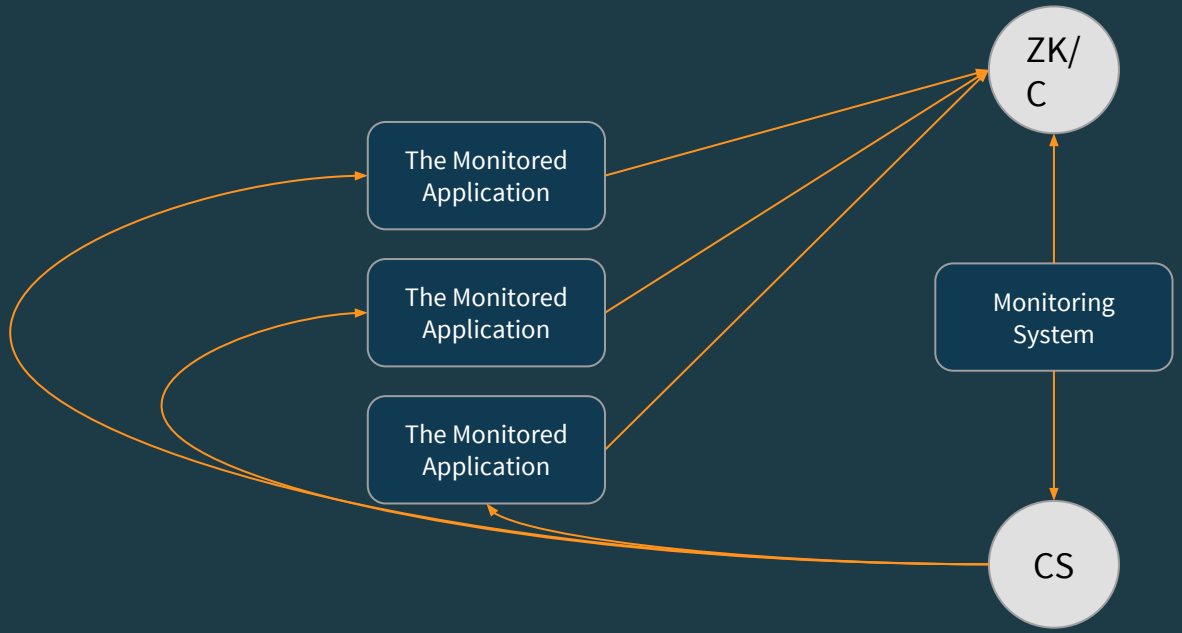
- Flexible Configuration
- Sampling Capabilities
- Delta Approach

Each cost mitigation has its drawbacks: less insight or higher complexity. When it comes to monitoring; ***lunch != free***

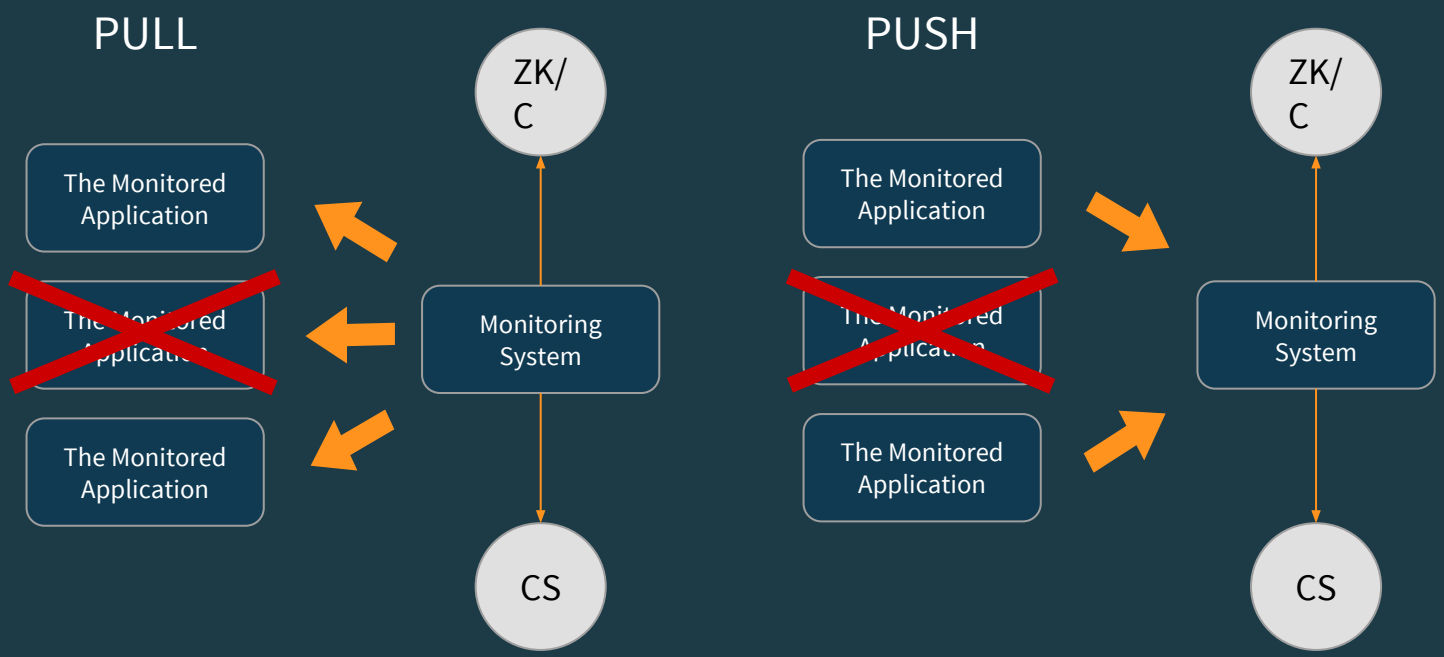
Trends in Monitoring: Pull/Push - Metrics Transfer



Trends in Monitoring: Pull/Push - Discoverability



Trends in Monitoring: Pull/Push - System Status



Trends in Monitoring: Pull/Push

- **Data Volumes and detail**
- Example of Pull Based Systems: Prometheus
- Example of Push Based System: StatsD

Trends in Monitoring: Anomaly Detection

- *“A set of techniques and systems to find unusual behaviors and/or states in systems and their observable signals”*
- Static thresholds on metrics are too crude
- Misconception: anomalies are *not* equal to outliers
- What is it good for?
    - Identify unusual metric values that hides underlying issues
    - Find changes in metrics so humans can investigate why
    - Help reduce surface area or search space
    - Lessen need to calibrate threshold values for alerting
- Example providers: VividCortex, Ruxit, AppDynamics

Trends in Monitoring: Configuration

Configuration Based:

- Class selection
- Path selection
- Wildcard selection
- Report-by: class, instance or path
- Includes support
- Excludes support
- Group support

```
cinnamon.akka {
  actors {
    "com.example.a.A" {
      report-by = class
    }
    "com.example.a.b.*" {
      report-by = instance
    }
    "/user/x/*" {
      report-by = instance
      excludes = "/user/x/y/z"
    }
    "/user/x/y/*" {
      report-by = class
    }
  }
}
```

Trends in Monitoring: Dynamic Configuration

Configuration is great but it's immutable. How about dynamic grouping?

The following configuration will create groups for every actor below the **/user/e/** path but exclude the path named **/user/e/r2/***:

```
cinnamon.akka {
  actors {
    "/user/e/?/*" {
      report-by = group
      excludes = "./r2/*"
    }
  }
}
```

Trends in Monitoring: Dynamic Configuration

Dynamic grouping is nice, but what about fully dynamic configuration in **runtime**?

- Dynamic typed properties
- Subscription based notification of property changes
- Event based trigger for property changes
- Property change conflict resolution
- Automatic reload for subscribers
- Wrappers for sources such as URL's, JDBC, etc.

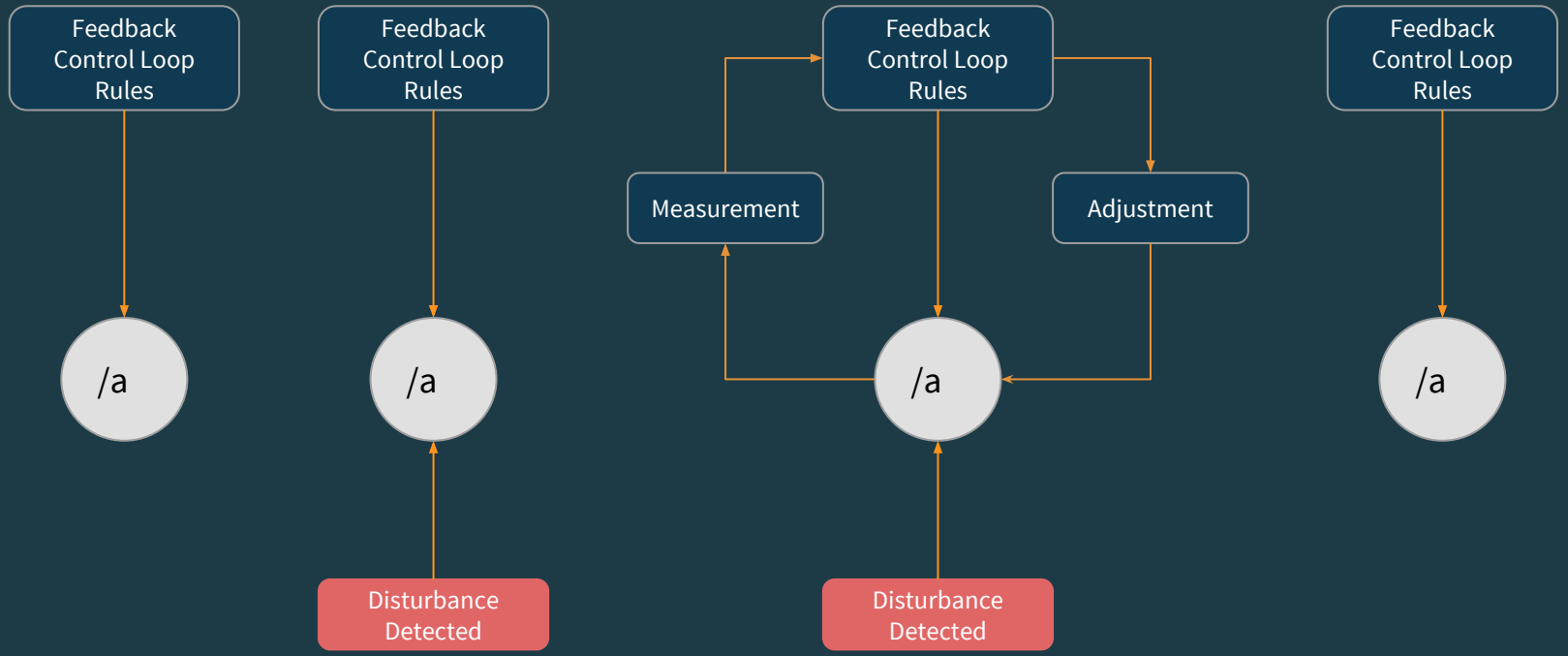
Trends in Monitoring: Self Monitoring

Imagine if we had our own doctor constantly checking our vital.

- Quick detection of problems
- Quick resolution
- Waste of resources
- Can be very expensive

This is kind of what most monitoring solutions are like today.

Trends in Monitoring: Self Monitoring



Best Practice

1. Understand your system, monitoring is not one-size-fits-all
2. Understand your monitoring tool - RT_M
3. Your monitoring tool should be configurable and elastic
4. Capture delta's, the ELK approach can deplete resources
5. Use sampling to capture and reason about trends
6. Capture enough info for relevant troubleshooting
7. Provide multiple channels for notification
8. Provide daily or weekly report for DevOps
9. Run your monitoring tool on a different machine

Lightbend Monitoring

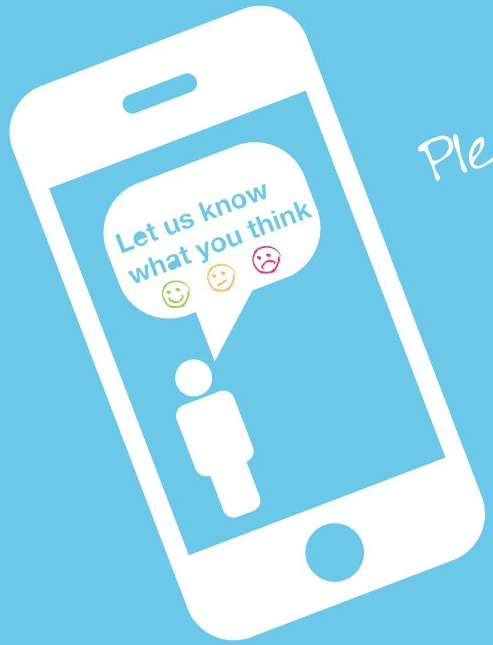
- Project started 1 year ago
- Integrates with Akka actors and Lagom circuit breakers generating metrics, events and traces
- Support for various backends
  - StatsD
  - Takipi
  - Miscellaneous Coda Hale reporters
- Monitoring requires subscription with Lightbend

Lightbend Monitoring: What is next?

- Dispatchers and Thread Pools
- “Flows”
  - Futures and Streams
  - Distributed Tracing (Zipkin style)
- Increased Lagom support
- Akka Http

Thank you for listening

Come by the Lightbend booth



Please

Remember to rate this session

Thank you!

Join the conversation #scaladays

Join the conversation #scaladays