

Акка в Яндекс

JPoint 2014

Вадим Цесько
<http://incubos.org>
@incubos

Яндекс

18 апреля 2014 г.



Предыстория

2 года назад

См. доклад «Потоковая обработка данных с помощью модели акторов (Actor Model)» на ADD-3^a

^ahttp://addconf.ru/talk.sdf/add/add_3/talks/12823



Содержание



Содержание

- Actor Model на примере Akka



Содержание

- Actor Model на примере Akka
 - Происхождение



Содержание

- Actor Model на примере Akka
 - Происхождение
 - Концепции и API



Содержание

- Actor Model на примере Akka
 - Происхождение
 - Концепции и API
 - Примеры кода



Содержание

- Actor Model на примере Akka
 - Происхождение
 - Концепции и API
 - Примеры кода
- Примеры систем в Яндекс



Содержание

- Actor Model на примере Akka
 - Происхождение
 - Концепции и API
 - Примеры кода
- Примеры систем в Яндекс
 - Конвейерная обработка данных



Содержание

- Actor Model на примере Akka
 - Происхождение
 - Концепции и API
 - Примеры кода
- Примеры систем в Яндекс
 - Конвейерная обработка данных
 - Реактивные иерархические системы



Содержание

- Actor Model на примере Akka
 - Происхождение
 - Концепции и API
 - Примеры кода
- Примеры систем в Яндекс
 - Конвейерная обработка данных
 - Реактивные иерархические системы
- Опыт разработки и эксплуатации



Содержание

- Actor Model на примере Akka
 - Происхождение
 - Концепции и API
 - Примеры кода
- Примеры систем в Яндекс
 - Конвейерная обработка данных
 - Реактивные иерархические системы
- Опыт разработки и эксплуатации
 - Подводные камни



Содержание

- Actor Model на примере Akka
 - Происхождение
 - Концепции и API
 - Примеры кода
- Примеры систем в Яндекс
 - Конвейерная обработка данных
 - Реактивные иерархические системы
- Опыт разработки и эксплуатации
 - Подводные камни
 - Проблемы и некоторые решения



Содержание

- Actor Model на примере Akka
 - Происхождение
 - Концепции и API
 - Примеры кода
- Примеры систем в Яндекс
 - Конвейерная обработка данных
 - Реактивные иерархические системы
- Опыт разработки и эксплуатации
 - Подводные камни
 - Проблемы и некоторые решения
 - Дополнительные тулы



Происхождение

¹[http://letitcrash.com/post/20964174345/
carl-hewitt-explains-the-essence-of-the-actor](http://letitcrash.com/post/20964174345/carl-hewitt-explains-the-essence-of-the-actor)



Происхождение

- Carl Hewitt¹, Peter Bishop and Richard Steiger. A Universal Modular Actor Formalism for Artificial Intelligence. 1973.

¹<http://letitcrash.com/post/20964174345/carl-hewitt-explains-the-essence-of-the-actor>

Происхождение

- Carl Hewitt¹, Peter Bishop and Richard Steiger. A Universal Modular Actor Formalism for Artificial Intelligence. 1973.
- Gul Agha. Actors: A Model of Concurrent Computation in Distributed Systems. 1986.

¹<http://letitcrash.com/post/20964174345/carl-hewitt-explains-the-essence-of-the-actor>

Происхождение

- Carl Hewitt¹, Peter Bishop and Richard Steiger. A Universal Modular Actor Formalism for Artificial Intelligence. 1973.
- Gul Agha. Actors: A Model of Concurrent Computation in Distributed Systems. 1986.
- Изначально для описания параллельных вычислений

¹<http://letitcrash.com/post/20964174345/carl-hewitt-explains-the-essence-of-the-actor>

Происхождение

- Carl Hewitt¹, Peter Bishop and Richard Steiger. A Universal Modular Actor Formalism for Artificial Intelligence. 1973.
- Gul Agha. Actors: A Model of Concurrent Computation in Distributed Systems. 1986.
- Изначально для описания параллельных вычислений
- Позднее в качестве основы для многочисленных реализаций

¹<http://letitcrash.com/post/20964174345/carl-hewitt-explains-the-essence-of-the-actor>

Actor

- Всё это актор



Actor

- Всё это актор
- Функционируют параллельно



Actor

- Всё это актор
- Функционируют параллельно
- Асинхронно обмениваются сообщениями



Actor

- Всё это актор
- Функционируют параллельно
- Асинхронно обмениваются сообщениями
- При обработке сообщения актор может
 - отправить конечное число сообщений другим акторам
 - создать конечное число новых акторов
 - назначить поведение для обработки следующего сообщения



Actor

- Всё это актор
- Функционируют параллельно
- Асинхронно обмениваются сообщениями
- При обработке сообщения актор может
 - отправить конечное число сообщений другим акторам
 - создать конечное число новых акторов
 - назначить поведение для обработки следующего сообщения
- Порядок доставки сообщений не специфицирован



Actor

- Всё это актор
- Функционируют параллельно
- Асинхронно обмениваются сообщениями
- При обработке сообщения актор может
 - отправить конечное число сообщений другим акторам
 - создать конечное число новых акторов
 - назначить поведение для обработки следующего сообщения
- Порядок доставки сообщений не специфицирован
- Акторы имеют «адреса»



Concurrency vs Parallelism

- Actor Model не про Parallelism, а про Concurrency²

²Rob Pike. Concurrency is not Parallelism:
<https://player.vimeo.com/video/49718712>



Concurrency vs Parallelism

- Actor Model не про Parallelism, а про Concurrency²

Concurrency

Способность выполняться параллельно.

²Rob Pike. Concurrency is not Parallelism:
<https://player.vimeo.com/video/49718712>

Concurrency vs Parallelism

- Actor Model не про Parallelism, а про Concurrency²

Concurrency

Способность выполняться параллельно.

Parallelism

Действительно параллельное выполнение.

²Rob Pike. Concurrency is not Parallelism:

<https://player.vimeo.com/video/49718712>

Реализации



³http://en.wikipedia.org/wiki/Actor_model



Реализации

- Языки³ с «родной» поддержкой: Erlang, Scala, ...



³http://en.wikipedia.org/wiki/Actor_model

Реализации

- Языки³ с «родной» поддержкой: Erlang, Scala, ...
- Библиотеки для языков: Scala, Java, F#, ...



³http://en.wikipedia.org/wiki/Actor_model

Реализации

- Языки³ с «родной» поддержкой: Erlang, Scala, ...
- Библиотеки для языков: Scala, Java, F#, ...

Будем рассматривать
Akka (Scala API).



³http://en.wikipedia.org/wiki/Actor_model

О проекте

Jonas Bonér:

- Java Champion
- Terracotta JVM clustering, JRockit JVM, AspectWerkz AOP, Eclipse AspectJ



О проекте

Jonas Bonér:

- Java Champion
- Terracotta JVM clustering, JRockit JVM, AspectWerkz AOP, Eclipse AspectJ

Ресурсы:

- <http://akka.io/docs/>
- <http://letitcrash.com/>



О проекте

Jonas Bonér:

- Java Champion
- Terracotta JVM clustering, JRockit JVM, AspectWerkz AOP, Eclipse AspectJ

Ресурсы:

- <http://akka.io/docs/>
- <http://letitcrash.com/>

Код:

- <https://github.com/akka/akka>
- Apache V2 license



Производительность

Внимание

Синтетические тесты ☺

⁴<http://letitcrash.com/post/14783691760/akka-vs-erlang>

⁵<http://letitcrash.com/post/20397701710/>

50-million-messages-per-second-on-a-single-machine



Производительность

Внимание

Синтетические тесты ☺

Erlang R14B04 vs **Akka** 2.0-SNAPSHOT⁴:

⁴<http://letitcrash.com/post/14783691760/akka-vs-erlang>

⁵<http://letitcrash.com/post/20397701710/>

50-million-messages-per-second-on-a-single-machine



Производительность

Внимание

Синтетические тесты ☺

Erlang R14B04 vs **Akka** 2.0-SNAPSHOT⁴:

- 1M mps vs **2.1M mps**

⁴<http://letitcrash.com/post/14783691760/akka-vs-erlang>

⁵<http://letitcrash.com/post/20397701710/>

50-million-messages-per-second-on-a-single-machine



Производительность

Внимание

Синтетические тесты ☺

Erlang R14B04 vs **Akka** 2.0-SNAPSHOT⁴:

- 1M mps vs **2.1M mps**

Akka 2.0⁵:

⁴<http://letitcrash.com/post/14783691760/akka-vs-erlang>

⁵<http://letitcrash.com/post/20397701710/>

50-million-messages-per-second-on-a-single-machine



Производительность

Внимание

Синтетические тесты ☺

Erlang R14B04 vs **Akka** 2.0-SNAPSHOT⁴:

- 1M mps vs **2.1M mps**

Akka 2.0⁵:

- **50M mps**

⁴<http://letitcrash.com/post/14783691760/akka-vs-erlang>

⁵<http://letitcrash.com/post/20397701710/>

50-million-messages-per-second-on-a-single-machine



Производительность

Внимание

Синтетические тесты ☺

Erlang R14B04 vs **Akka** 2.0-SNAPSHOT⁴:

- 1M mps vs **2.1M mps**

Akka 2.0⁵:

- **50M mps**
- 48-core, 128 GB, ForkJoinPool

⁴<http://letitcrash.com/post/14783691760/akka-vs-erlang>

⁵<http://letitcrash.com/post/20397701710/>

50-million-messages-per-second-on-a-single-machine



Особенности реализации



Особенности реализации

- **300 байт** на актор



Особенности реализации

- **300 байт** на актор
- Актор: состояние, поведение, почтовый ящик, список детей, стратегия супервизора



Особенности реализации

- **300 байт** на актор
- Актор: состояние, поведение, почтовый ящик, список детей, стратегия супервизора
- Множество акторов на множестве нитей



Особенности реализации

- **300 байт** на актор
- Актор: состояние, поведение, почтовый ящик, список детей, стратегия супервизора
- Множество акторов на множестве нитей
- Нет гарантированной доставки, семантика **at-most-once**, порядок сохраняется



Особенности реализации

- **300 байт** на актор
- Актор: состояние, поведение, почтовый ящик, список детей, стратегия супервизора
- Множество акторов на множестве нитей
- Нет гарантированной доставки, семантика **at-most-once**, порядок сохраняется
- Сообщения обрабатываются **строго по порядку**



Особенности реализации

- **300 байт** на актор
- Актор: состояние, поведение, почтовый ящик, список детей, стратегия супервизора
- Множество акторов на множестве нитей
- Нет гарантированной доставки, семантика **at-most-once**, порядок сохраняется
- Сообщения обрабатываются **строго по порядку**
- Иерархия: создаваемые акторы — дети, родитель — супервизор



Особенности реализации

- **300 байт** на актор
- Актор: состояние, поведение, почтовый ящик, список детей, стратегия супервизора
- Множество акторов на множестве нитей
- Нет гарантированной доставки, семантика **at-most-once**, порядок сохраняется
- Сообщения обрабатываются **строго по порядку**
- Иерархия: создаваемые акторы — дети, родитель — супервизор
- Актор скрыт за **переносимой** ActorRef



Особенности реализации

- **300 байт** на актор
- Актор: состояние, поведение, почтовый ящик, список детей, стратегия супервизора
- Множество акторов на множестве нитей
- Нет гарантированной доставки, семантика **at-most-once**, порядок сохраняется
- Сообщения обрабатываются **строго по порядку**
- Иерархия: создаваемые акторы — дети, родитель — супервизор
- Актор скрыт за **переносимой** ActorRef
- Подход «Let it crash»



Пути

Примеры

- akka://system/user/a/b
- akka.tcp://system@server.yandex.ru:2552/user/a/b



Конструирование ссылок

- Создание акторов: `ActorRefProvider.actorOf`
- Поиск акторов:
`ActorRefProvider.actorSelection`
- Каждый актор знает себя, родителя и детей



Конструирование ссылок

- Создание акторов: `ActorRefProvider.actorOf`
- Поиск акторов:
`ActorRefProvider.actorSelection`
- Каждый актор знает себя, родителя и детей

Можно делать так:

```
1 context.actorSelection("../brother") ! msg
2 context.actorSelection("/user/service") ! msg
3 context.actorSelection("../*") ! msg
```



Определение актора

```
1 class Partitioner(partitionStorage: ActorRef) extends Actor {
2
3   def receive = {
4     case PartitionFeed(partner, offers) =>
5       partition(partner, offers)
6     case msg =>
7       log.error("Unsupported message received: {}", msg)
8   }
9
10  def partition(partner: Partner, offers: Traversable[Offer]) {
11    ...
12
13    partitionStorage ! UpdatePartitions(partner, partitioning)
14  }
15 }
```



Создание актора

```
1 val system = ActorSystem("sharder")
2
3 val partitionStorage = ...
4
5 val partitioner =
6     system.actorOf(
7         FromConfig.props(
8             Props[Partitioner](new Partitioner(partitionStorage))
9                 .withDispatcher("dispatcher.cpu")),
10         "partitioner")
```



Конфигурация диспетчера

Конфигурация в HOCON⁶:

```
1 dispatchers.cpu {  
2   type = Dispatcher  
3   executor = "fork-join-executor"  
4  
5   fork-join-executor {  
6     parallelism-min = 4  
7     parallelism-factor = 1.0  
8   }  
9 }
```

⁶Human-Optimized Config Object Notation:
<https://github.com/typesafehub/config>

HOCON

Независимая самоценная библиотека:



HOCON

Независимая самоценная библиотека:

- Чистая Java без зависимостей



HOCON

Независимая самоценная библиотека:

- Чистая Java без зависимостей
- Поддерживает Java properties и JSON superset



HOCON

Независимая самоценная библиотека:

- Чистая Java без зависимостей
- Поддерживает Java properties и JSON superset
- Merge и include из файлов, URL, classpath



HOCON

Независимая самоценная библиотека:

- Чистая Java без зависимостей
- Поддерживает Java properties и JSON superset
- Merge и include из файлов, URL, classpath
- Мощная поддержка вложенности



HOCON

Независимая самоценная библиотека:

- Чистая Java без зависимостей
- Поддерживает Java properties и JSON superset
- Merge и include из файлов, URL, classpath
- Мощная поддержка вложенности
- Поддержка duration (10 seconds) и size (512K)



HOCON

Независимая самоценная библиотека:

- Чистая Java без зависимостей
- Поддерживает Java properties и JSON superset
- Merge и include из файлов, URL, classpath
- Мощная поддержка вложенности
- Поддержка duration (10 seconds) и size (512K)
- Конвертация типов



Диспетчеры

- Dispatcher
- PinnedDispatcher
- BalancingDispatcher
- CallingThreadDispatcher

на

- fork-join-executor
- thread-pool-executor



Почтовые ящики

- UnboundedMailbox
- BoundedMailbox
- UnboundedPriorityMailbox
- BoundedPriorityMailbox



Почтовые ящики

- UnboundedMailbox
- BoundedMailbox
- UnboundedPriorityMailbox
- BoundedPriorityMailbox

```
1 trait MessageQueue {  
2   def enqueue(receiver: ActorRef, handle: Envelope): Unit  
3   def dequeue(): Envelope  
4   def numberOfMessages: Int  
5   def hasMessages: Boolean  
6   def cleanUp(owner: ActorRef, deadLetters: MessageQueue): Unit  
7 }
```



Пулы акторов

```
1 akka.actor.deployment {  
2   /partitioner {  
3     router = smallest-mailbox-pool  
4     nr-of-instances = 4  
5   }  
6 }
```



Пулы акторов

```
1 akka.actor.deployment {  
2   /partitioner {  
3     router = smallest-mailbox-pool  
4     nr-of-instances = 4  
5   }  
6 }
```

```
1 akka.actor.deployment {  
2   /unifier {  
3     router = round-robin-pool  
4     resizer {  
5       lower-bound = 2  
6       upper-bound = 16  
7     }  
8   }  
9 }
```



Типы роутеров

Реализации Pool и Group:

- RoundRobin
- Random
- SmallestMailbox
- Broadcast
- ScatterGatherFirstCompleted



Типы роутеров

Реализации Pool и Group:

- RoundRobin
- Random
- SmallestMailbox
- Broadcast
- ScatterGatherFirstCompleted
- Самописные



Конфигурирование из кода

```
1 val shardActorPaths: immutable.Seq[String] = ...  
2  
3 val shard = system.actorOf(  
4   RoundRobinGroup(shardActorPaths).props(),  
5   "shard")
```



Удалённые акторы

```
1 akka {  
2   actor {  
3     provider = "akka.remote.RemoteActorRefProvider"  
4   }  
5  
6   remote {  
7     enabled-transport = ["akka.remote.netty.tcp"]  
8  
9     netty.tcp {  
10      hostname = "server.yandex.ru"  
11      port = 2553  
12    }  
13  }  
14 }
```



Тестирование акторов



Тестирование акторов

- Модульное тестирование с TestActorRef



Тестирование акторов

- Модульное тестирование с TestActorRef
- Интеграционное тестирование с Probe



Тестирование акторов

- Модульное тестирование с `TestActorRef`
- Интеграционное тестирование с `Probe`
- Проверки с сопоставлением по шаблону



Пример теста

```
1 val probe = TestProbe()
2 val burstScaler = TestActorRef(new BurstScaler(probe.ref))
3
4 before {
5     burstScaler.underlyingActor.sent =
6         burstScaler.underlyingActor.sent.empty
7 }
8
9 "A BurstScaler" should {
10     "always forward the first message" in {
11         probe.within(1 second) {
12             burstScaler ! 1
13             probe.expectMsg(1)
14         }
15     }
16 }
```



Супервизор

Решение при сбое:

- 1 Resume
- 2 Restart
- 3 Stop
- 4 Escalate



Супервизор

Решение при сбое:

- 1 Resume
 - 2 Restart
 - 3 Stop
 - 4 Escalate
- Принятое решение (1-3) действует рекурсивно



Супервизор

Решение при сбое:

- 1 Resume
 - 2 Restart
 - 3 Stop
 - 4 Escalate
- Принятое решение (1-3) действует рекурсивно
 - Функция Exception $\Rightarrow$ Directive



Супервизор

Решение при сбое:

- 1 Resume
 - 2 Restart
 - 3 Stop
 - 4 Escalate
- Принятое решение (1-3) действует рекурсивно
 - Функция Exception $\Rightarrow$ Directive
 - Terminated, preStart, preRestart, postStop, postRestart



Супервизор

Решение при сбое:

- 1 Resume
 - 2 Restart
 - 3 Stop
 - 4 Escalate
- Принятое решение (1-3) действует рекурсивно
 - Функция `Exception` $\Rightarrow$ `Directive`
 - `Terminated`, `preStart`, `preRestart`, `postStop`, `postRestart`
 - `OneForOneStrategy` и `AllForOneStrategy`



Супервизор

Решение при сбое:

- 1 Resume
 - 2 Restart
 - 3 Stop
 - 4 Escalate
- Принятое решение (1-3) действует рекурсивно
 - Функция `Exception` $\Rightarrow$ `Directive`
 - `Terminated`, `preStart`, `preRestart`, `postStop`, `postRestart`
 - `OneForOneStrategy` и `AllForOneStrategy`
 - Ограничение количества перезапусков



Супервизор по умолчанию

```
1 final val defaultDecider: Decider = {  
2   case _: ActorInitializationException => Stop  
3   case _: ActorKilledException       => Stop  
4   case _: DeathPactException         => Stop  
5   case _: Exception                 => Restart  
6 }  
7  
8 final val defaultStrategy: SupervisorStrategy =  
9   OneForOneStrategy()(defaultDecider)
```



Классификация



⁷<http://spray.io>

Классификация

- Конвейерная индексация
 - Яндекс.{Авто, Недвижимость, Работа}
 - etc.



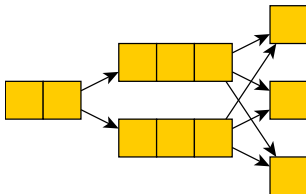
⁷<http://spray.io>

Классификация

- Конвейерная индексация
 - Яндекс.{Авто, Недвижимость, Работа}
 - etc.
- Иерархическая обработка пользовательских запросов
 - Spray⁷
 - Маршрутизация и обработка кликов и показов
 - Матчинг объявлений на подписки
 - etc.

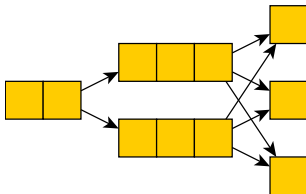
⁷<http://spray.io>

Конвейер

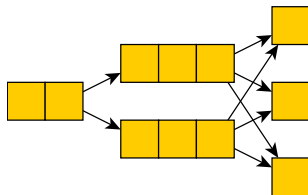


Конвейер

- Масштабируемость

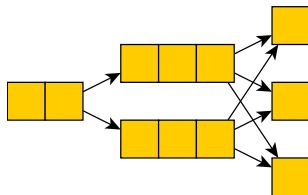


Конвейер



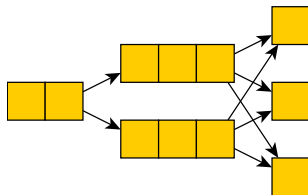
- Масштабируемость
- Устойчивость к сбоям

Конвейер



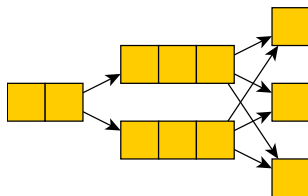
- Масштабируемость
- Устойчивость к сбоям
- Ограниченные очереди

Конвейер



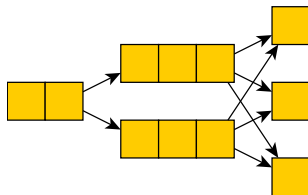
- Масштабируемость
- Устойчивость к сбоям
- Ограниченные очереди
- Back pressure

Конвейер



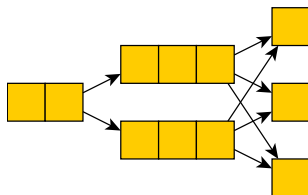
- Масштабируемость
- Устойчивость к сбоям
- Ограниченные очереди
- Back pressure
- Выделенные диспетчеры

Конвейер



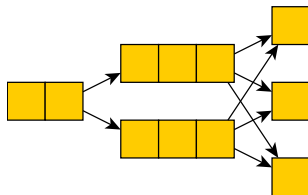
- Масштабируемость
- Устойчивость к сбоям
- Ограниченные очереди
- Back pressure
- Выделенные диспетчеры
- Детерминированное потребление памяти

Конвейер



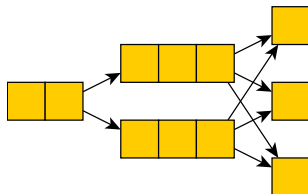
- Масштабируемость
- Устойчивость к сбоям
- Ограниченные очереди
- Back pressure
- Выделенные диспетчеры
- Детерминированное потребление памяти
- Графики размеров очередей

Конвейер



- Масштабируемость
- Устойчивость к сбоям
- Ограниченные очереди
- Back pressure
- Выделенные диспетчеры
- Детерминированное потребление памяти
- Графики размеров очередей
- Декомпозиция стадий

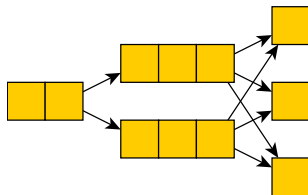
Конвейер



- Масштабируемость
- Устойчивость к сбоям
- Ограниченные очереди
- Back pressure
- Выделенные диспетчеры
- Детерминированное потребление памяти
- Графики размеров очередей
- Декомпозиция стадий
- Масштабирование пулами



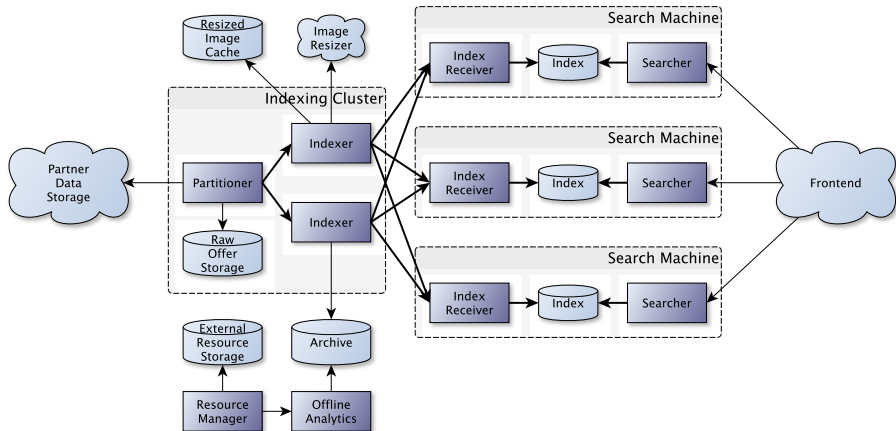
Конвейер



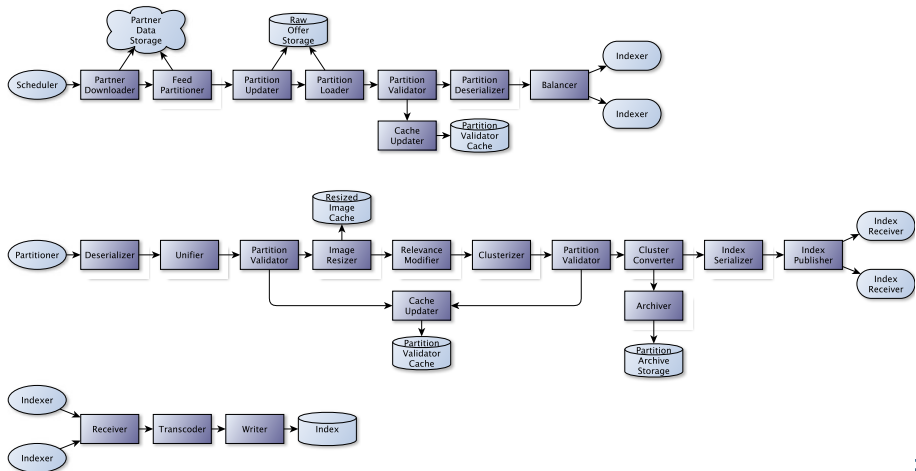
- Масштабируемость
- Устойчивость к сбоям
- Ограниченные очереди
- Back pressure
- Выделенные диспетчеры
- Детерминированное потребление памяти
- Графики размеров очередей
- Декомпозиция стадий
- Масштабирование пулами
- Загрузка CPU 100%



Яндекс.Авто: Архитектура



Яндекс.Авто: Гиперконвейер



Яндекс.Авто: Цифры



Яндекс.Авто: Цифры

- **4 ДЦ, 4 машины** для индексации, **60 машин** для поиска (общие для разных сервисов)



Яндекс.Авто: Цифры

- 4 ДЦ, 4 машины для индексации, 60 машин для поиска (общие для разных сервисов)
- Больше 2М объявлений



Яндекс.Авто: Цифры

- 4 ДЦ, 4 машины для индексации, 60 машин для поиска (общие для разных сервисов)
- Больше 2М объявлений
- Время в конвейере — до 5 мин



Яндекс.Авто: Цифры

- 4 ДЦ, 4 машины для индексации, 60 машин для поиска (общие для разных сервисов)
- Больше 2М объявлений
- Время в конвейере — до 5 мин
- Принудительная переиндексация — каждые 15 мин



Яндекс.Авто: Цифры

- 4 ДЦ, 4 машины для индексации, 60 машин для поиска (общие для разных сервисов)
- Больше 2М объявлений
- Время в конвейере — до 5 мин
- Принудительная переиндексация — каждые 15 мин
- Объём поискового индекса — больше 2 ГБ



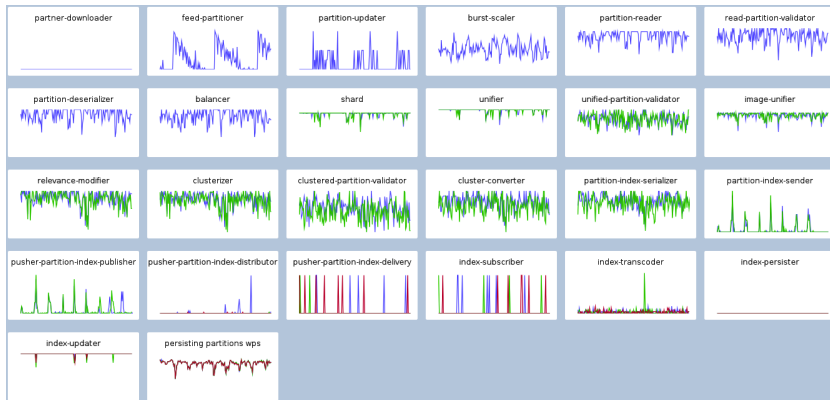
Яндекс.Авто: Цифры

- **4 ДЦ, 4 машины** для индексации, **60 машин** для поиска (общие для разных сервисов)
- Больше **2М объявлений**
- Время в конвейере — до **5 мин**
- Принудительная переиндексация — каждые **15 мин**
- Объём поискового индекса — больше **2 ГБ**
- Scala — **8 KLOC**, Java — **210 KLOC**



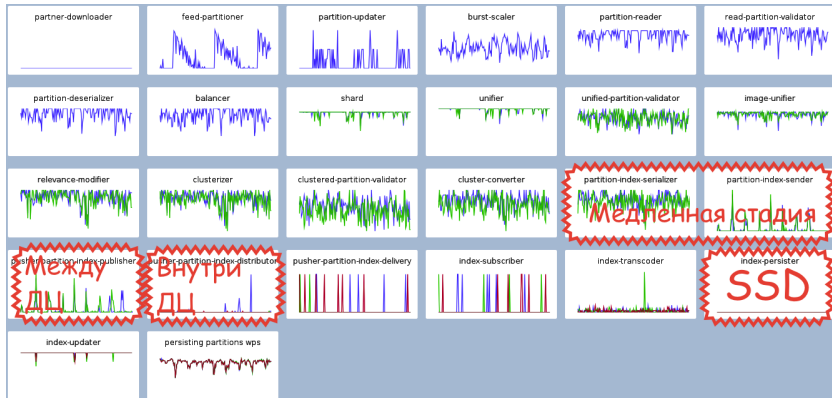
Инструментированные очереди

Metrics⁸ + Graphite:



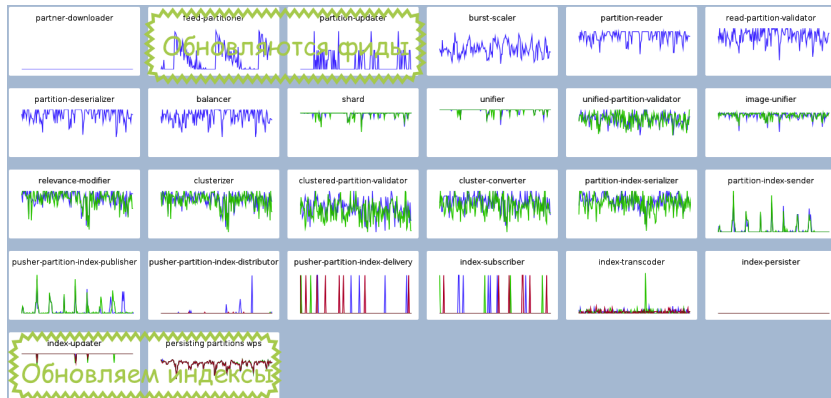
⁸<http://metrics.codahale.com/>

Диагностика



jpoint

Интересное



Специфика

Для прореживания потока задач:



Специфика

Для прореживания потока задач:

- BurstScaler



Специфика

Для прореживания потока задач:

- BurstScaler
- InstrumentedUnboundedSkipClonesMailbox



Специфика

Для прореживания потока задач:

- BurstScaler
- InstrumentedUnboundedSkipClonesMailbox
- InstrumentedSkippingBoundedMailbox



Горизонтальное масштабирование (1)

Планировали масштабировать индексацию так:



Горизонтальное масштабирование (1)

Планировали масштабировать индексацию так:

- Добавляем машины-indexer'ы



Горизонтальное масштабирование (1)

Планировали масштабировать индексацию так:

- Добавляем машины-indexer'ы
- Используем RoundRobin- или Random-пул



Горизонтальное масштабирование (1)

Планировали масштабировать индексацию так:

- Добавляем машины-indexer'ы
- Используем RoundRobin- или Random-пул

Не работает

Одна из машин патологически тупит — сильно тормозит весь конвейер



Горизонтальное масштабирование (1)

Планировали масштабировать индексацию так:

- Добавляем машины-indexer'ы
- Используем RoundRobin- или Random-пул

Не работает

Одна из машин патологически тупит — сильно тормозит весь конвейер

AdaptiveBalancer

Измеряем время пропихивания данных и штрафуем на некоторое время втупливающие ноды

Горизонтальное масштабирование (2)

Обновление данных на поисковых машинах:



Горизонтальное масштабирование (2)

Обновление данных на поисковых машинах:

- Несколько ДЦ + *ненадёжная* сеть



Горизонтальное масштабирование (2)

Обновление данных на поисковых машинах:

- Несколько ДЦ + *ненадёжная* сеть

Эволюция



Горизонтальное масштабирование (2)

Обновление данных на поисковых машинах:

- Несколько ДЦ + *ненадёжная* сеть

Эволюция

- BroadcastPool



Горизонтальное масштабирование (2)

Обновление данных на поисковых машинах:

- Несколько ДЦ + *ненадёжная* сеть

Эволюция

- BroadcastPool
- ZeroMQ (EPGM, TCP)



Горизонтальное масштабирование (2)

Обновление данных на поисковых машинах:

- Несколько ДЦ + *ненадёжная* сеть

Эволюция

- BroadcastPool
- ZeroMQ (EPGM, TCP)
- JGroups (Multicast + Relay)



Горизонтальное масштабирование (2)

Обновление данных на поисковых машинах:

- Несколько ДЦ + *ненадёжная* сеть

Эволюция

- BroadcastPool
- ZeroMQ (EPGM, TCP)
- JGroups (Multicast + Relay)

Akka-based Spanning Tree

Горизонтальное масштабирование (2)

Обновление данных на поисковых машинах:

- Несколько ДЦ + *ненадёжная* сеть

Эволюция

- BroadcastPool
- ZeroMQ (EPGM, TCP)
- JGroups (Multicast + Relay)

Akka-based Spanning Tree

- Доставляем хотя бы одной ноде в каждом ДЦ

Горизонтальное масштабирование (2)

Обновление данных на поисковых машинах:

- Несколько ДЦ + *ненадёжная* сеть

Эволюция

- BroadcastPool
- ZeroMQ (EPGM, TCP)
- JGroups (Multicast + Relay)

Akka-based Spanning Tree

- Доставляем хотя бы одной ноде в каждом ДЦ
- Нода распространяет данные внутри ДЦ

Горизонтальное масштабирование (2)

Обновление данных на поисковых машинах:

- Несколько ДЦ + *ненадёжная* сеть

Эволюция

- BroadcastPool
- ZeroMQ (EPGM, TCP)
- JGroups (Multicast + Relay)

Akka-based Spanning Tree

- Доставляем хотя бы одной ноде в каждом ДЦ
- Нода распространяет данные внутри ДЦ
- Всё на akka-remote + ZooKeeper

Горизонтальное масштабирование (2)

Обновление данных на поисковых машинах:

- Несколько ДЦ + *ненадёжная* сеть

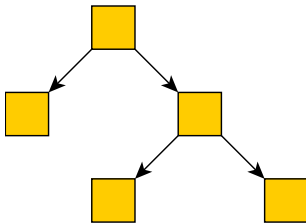
Эволюция

- BroadcastPool
- ZeroMQ (EPGM, TCP)
- JGroups (Multicast + Relay)

Akka-based Spanning Tree

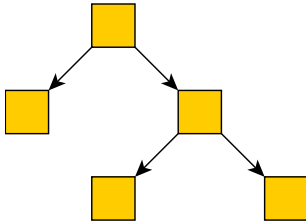
- Доставляем хотя бы одной ноде в каждом ДЦ
- Нода распространяет данные внутри ДЦ
- Всё на akka-remote + ZooKeeper
- Тюнинг и патчинг akka-remote

Иерархическая обработка

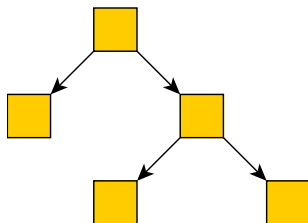


Иерархическая обработка

- Масштабируемость

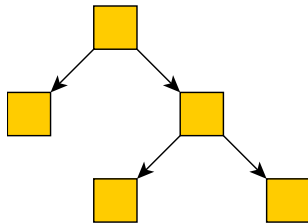


Иерархическая обработка



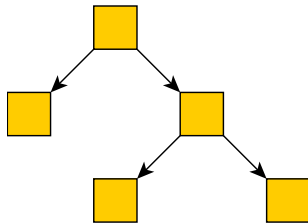
- Масштабируемость
- Высокая доступность

Иерархическая обработка



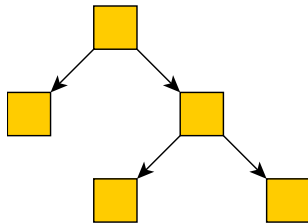
- Масштабируемость
- Высокая доступность
- Естественные подсистемы

Иерархическая обработка



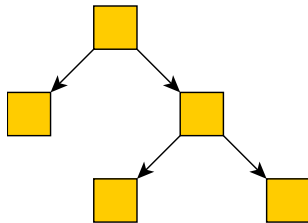
- Масштабируемость
- Высокая доступность
- Естественные подсистемы
- Часто динамическая топология

Иерархическая обработка



- Масштабируемость
- Высокая доступность
- Естественные подсистемы
- Часто динамическая топология
- Actor per request

Иерархическая обработка



- Масштабируемость
- Высокая доступность
- Естественные подсистемы
- Часто динамическая топология
- Actor per request
- Request-reply

Иерархические системы

⁹<http://curator.apache.org>

¹⁰http://en.wikipedia.org/wiki/Consistent_hashing



Иерархические системы

- Внутренние сервисы

⁹<http://curator.apache.org>

¹⁰http://en.wikipedia.org/wiki/Consistent_hashing

Иерархические системы

- Внутренние сервисы
- Иерархия акторов для каждого активного пользователя/подписки/визита/запроса

⁹<http://curator.apache.org>

¹⁰http://en.wikipedia.org/wiki/Consistent_hashing

Иерархические системы

- Внутренние сервисы
- Иерархия акторов для каждого активного пользователя/подписки/визита/запроса
- Динамическая выгрузка/подгрузка сущностей

⁹<http://curator.apache.org>

¹⁰http://en.wikipedia.org/wiki/Consistent_hashing

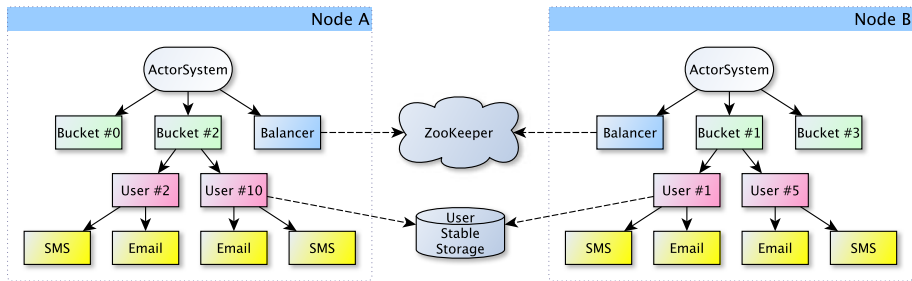
Иерархические системы

- Внутренние сервисы
- Иерархия акторов для каждого активного пользователя/подписки/визита/запроса
- Динамическая выгрузка/подгрузка сущностей
- Распределение пользователей на бакеты (и ноды) через ZooKeeper + Curator⁹ (Consistent Hashing¹⁰)

⁹<http://curator.apache.org>

¹⁰http://en.wikipedia.org/wiki/Consistent_hashing

Компонент сервиса уведомлений



О чём нужно думать



О чём нужно думать

- Наблюдаемость



О чём нужно думать

- Наблюдаемость
 - Логгирование



О чём нужно думать

- Наблюдаемость
 - Логгирование
 - Агрегатные графики



О чём нужно думать

- Наблюдаемость
 - Логгирование
 - Агрегатные графики
 - Мониторинги



О чём нужно думать

- Наблюдаемость
 - Логгирование
 - Агрегатные графики
 - Мониторинги
- Поведение при перегрузке



О чём нужно думать

- Наблюдаемость
 - Логгирование
 - Агрегатные графики
 - Мониторинги
- Поведение при перегрузке
 - Детектирование перегрузки



О чём нужно думать

- Наблюдаемость
 - Логгирование
 - Агрегатные графики
 - Мониторинги
- Поведение при перегрузке
 - Детектирование перегрузки
 - Ограничение ресурсов



О чём нужно думать

- Наблюдаемость
 - Логгирование
 - Агрегатные графики
 - Мониторинги
- Поведение при перегрузке
 - Детектирование перегрузки
 - Ограничение ресурсов
 - Превентивный reject



О чём нужно думать

- Наблюдаемость
 - Логгирование
 - Агрегатные графики
 - Мониторинги
- Поведение при перегрузке
 - Детектирование перегрузки
 - Ограничение ресурсов
 - Превентивный reject
 - Восстановление после перегрузки



О чём нужно думать

- Наблюдаемость
 - Логгирование
 - Агрегатные графики
 - Мониторинги
- Поведение при перегрузке
 - Детектирование перегрузки
 - Ограничение ресурсов
 - Превентивный reject
 - Восстановление после перегрузки
 - Выравнивание нагрузки



Deadlocks

Ограниченные ресурсы:



Deadlocks

Ограниченные ресурсы:

- Очереди



Deadlocks

Ограниченные ресурсы:

- Очереди
- Диспетчеры



Deadlocks

Ограниченные ресурсы:

- Очереди
- Диспетчеры
- Пулы нитей (в т. ч. в akka-remote)



Deadlocks

Ограниченные ресурсы:

- Очереди
- Диспетчеры
- Пулы нитей (в т. ч. в akka-remote)
- Косвенная синхронизация между акторами



Deadlocks

Ограниченные ресурсы:

- Очереди
- Диспетчеры
- Пулы нитей (в т. ч. в akka-remote)
- Косвенная синхронизация между акторами

Типы:



Deadlocks

Ограниченные ресурсы:

- Очереди
- Диспетчеры
- Пулы нитей (в т. ч. в akka-remote)
- Косвенная синхронизация между акторами

Типы:

- Локальные



Deadlocks

Ограниченные ресурсы:

- Очереди
- Диспетчеры
- Пулы нитей (в т. ч. в akka-remote)
- Косвенная синхронизация между акторами

Типы:

- Локальные
- **Распределённые**



OOMs

Возможные причины:



OOMs

Возможные причины:

- Перегрузка + неограниченные очереди



OOMs

Возможные причины:

- Перегрузка + неограниченные очереди
- Логические утечки памяти (в т. ч. через замыкания, см. `ActorContext.sender`)



Logging



Logging

ActorLogging



Logging

ActorLogging

- Slf4jLogger (Logback) vs akka.loglevel



Logging

ActorLogging

- Slf4jLogger (Logback) vs akka.loglevel
- Перегрузка EventBus



Logging

ActorLogging

- Slf4jLogger (Logback) vs akka.loglevel
- Перегрузка EventBus
- OOM



Logging

ActorLogging

- Slf4jLogger (Logback) vs akka.loglevel
- Перегрузка EventBus
- OOM

Решение

Logging

ActorLogging

- Slf4jLogger (Logback) vs akka.loglevel
- Перегрузка EventBus
- OOM

Решение

- Использовать slf4j-api напрямую

Logging

ActorLogging

- Slf4jLogger (Logback) vs akka.loglevel
- Перегрузка EventBus
- OOM

Решение

- Использовать slf4j-api напрямую
- Обёртки для наполнения MDC спецификой Akka

Dead Letters



Dead Letters

Внимание

В любой непонятной ситуации сообщение пойдёт в DeadLetter



Dead Letters

Внимание

В любой непонятной ситуации сообщение пойдёт в DeadLetter

Рекомендация

Подписываться на DeadLetter, логгировать, вылавливать и мониторить «интересные» сообщения



Blocking



Blocking

- Актор на время обработки сообщения берёт нитку из диспетчера



Blocking

- Актор на время обработки сообщения берёт нитку из диспетчера
- Thread starvation при блокировании в акторе



Blocking

- Актор на время обработки сообщения берёт нитку из диспетчера
- Thread starvation при блокировании в акторе
- Особенно «интересные» результаты на default-dispatcher



Blocking

- Актор на время обработки сообщения берёт нитку из диспетчера
- Thread starvation при блокировании в акторе
- Особенно «интересные» результаты на default-dispatcher

Рекомендация

Запускайте блокирующихся акторов на собственном выделенном диспетчере

Akka и JMM

¹¹Chris Okasaki. *Purely Functional Data Structures*. 1999.

<http://www.cs.cmu.edu/~rwh/theses/okasaki.pdf>



Akka и JMM

Happens before

¹¹Chris Okasaki. *Purely Functional Data Structures*. 1999.

<http://www.cs.cmu.edu/~rwh/theses/okasaki.pdf>



Akka и JMM

Happens before

- Message send happens before receive

¹¹Chris Okasaki. *Purely Functional Data Structures*. 1999.
<http://www.cs.cmu.edu/~rwh/theses/okasaki.pdf>



Akka и JMM

Happens before

- Message send happens before receive
- Actor receive happens before previous receive

¹¹Chris Okasaki. *Purely Functional Data Structures*. 1999.
<http://www.cs.cmu.edu/~rwh/theses/okasaki.pdf>

Akka и JMM

Happens before

- Message send happens before receive
- Actor receive happens before previous receive

Следствия:

¹¹Chris Okasaki. *Purely Functional Data Structures*. 1999.
<http://www.cs.cmu.edu/~rwh/theses/okasaki.pdf>

Akka и JMM

Happens before

- Message send happens before receive
- Actor receive happens before previous receive

Следствия:

- Не нужно защищать состояние актора

¹¹Chris Okasaki. *Purely Functional Data Structures*. 1999.
<http://www.cs.cmu.edu/~rwh/theses/okasaki.pdf>

Akka и JMM

Happens before

- Message send happens before receive
- Actor receive happens before previous receive

Следствия:

- Не нужно защищать состояние актора
- Возможны гонки на изменяемых сообщениях

¹¹Chris Okasaki. *Purely Functional Data Structures*. 1999.
<http://www.cs.cmu.edu/~rwh/theses/okasaki.pdf>

Akka и JMM

Happens before

- Message send happens before receive
- Actor receive happens before previous receive

Следствия:

- Не нужно защищать состояние актора
- Возможны гонки на изменяемых сообщениях
- Используйте неизменяемые структуры данных¹¹

¹¹Chris Okasaki. *Purely Functional Data Structures*. 1999.
<http://www.cs.cmu.edu/~rwh/theses/okasaki.pdf>

Наблюдаемость



Наблюдаемость

Статическая топология

Никаких проблем



Наблюдаемость

Статическая топология

Никаких проблем

Динамическая топология

Пока только с привлечением мозга



Akka



Akka

- Бинарная несовместимость на пустом месте



Akka

- Бинарная несовместимость на пустом месте
- API иногда колбасит (диспетчеры и очереди)



Akka

- Бинарная несовместимость на пустом месте
- API иногда колбасит (диспетчеры и очереди)
- До многих вещей не дотянуться — приходится копипастить и патчить



Scala



Scala

- spray-routing (shapeless) крепкий орешек для IDEA



Scala

- spray-routing (shapeless) крепкий орешек для IDEA
- Closures (call-by-name, laziness) — утечки памяти, гонки и гейзенбаги
 - Незащищённое состояние актора
 - В т. ч. `sender()`



Рассмотрели



Рассмотрели

- Actor Model в реализации Akka



Рассмотрели

- Actor Model в реализации Akka
- Конвейерные и иерархические системы на Akka



Рассмотрели

- Actor Model в реализации Akka
- Конвейерные и иерархические системы на Akka
- Некоторые проблемы и решения



В итоге



В итоге

- Прошли путь от Akka 1.x до 2.3.x



В итоге

- Прошли путь от Akka 1.x до 2.3.x
- 1.5 года в production



В итоге

- Прошли путь от Akka 1.x до 2.3.x
- 1.5 года в production
- Множество граблей, но жить можно



В итоге

- Прошли путь от Akka 1.x до 2.3.x
- 1.5 года в production
- Множество граблей, но жить можно
- Исключительно простой и выразительный формализм



В итоге

- Прошли путь от Akka 1.x до 2.3.x
- 1.5 года в production
- Множество граблей, но жить можно
- Исключительно простой и выразительный формализм
- Удобно проектировать, разрабатывать и тестировать



Ссылки

¹²http://addconf.ru/talk.sdf/add/add_3/talks/12823

¹³<http://typesafe.com/activator>

¹⁴<https://class.coursera.org/reactive-001/class>

¹⁵<http://letitcrash.com/post/64667109914/>

series-the-neophytes-guide-to-scala

¹⁶<http://letitcrash.com/post/59190266995/>



Ссылки

- «Потоковая обработка данных с помощью модели акторов (Actor Model)» на ADD-3¹²

¹²http://addconf.ru/talk.sdf/add/add_3/talks/12823

¹³<http://typesafe.com/activator>

¹⁴<https://class.coursera.org/reactive-001/class>

¹⁵<http://letitcrash.com/post/64667109914/>
series-the-neophytes-guide-to-scala

¹⁶<http://letitcrash.com/post/59190266995/>



Ссылки

- «Потоковая обработка данных с помощью модели акторов (Actor Model)» на ADD-3¹²
- Typesafe Activator¹³

¹²http://addconf.ru/talk.sdf/add/add_3/talks/12823

¹³<http://typesafe.com/activator>

¹⁴<https://class.coursera.org/reactive-001/class>

¹⁵<http://letitcrash.com/post/64667109914/>
series-the-neophytes-guide-to-scala

¹⁶<http://letitcrash.com/post/59190266995/>



Ссылки

- «Потоковая обработка данных с помощью модели акторов (Actor Model)» на ADD-3¹²
- Typesafe Activator¹³
- Principles of Reactive Programming¹⁴

¹²http://addconf.ru/talk.sdf/add/add_3/talks/12823

¹³<http://typesafe.com/activator>

¹⁴<https://class.coursera.org/reactive-001/class>

¹⁵<http://letitcrash.com/post/64667109914/>
series-the-neophytes-guide-to-scala

¹⁶<http://letitcrash.com/post/59190266995/>



Ссылки

- «Потоковая обработка данных с помощью модели акторов (Actor Model)» на ADD-3¹²
- Typesafe Activator¹³
- Principles of Reactive Programming¹⁴
- Книги:

¹²http://addconf.ru/talk.sdf/add/add_3/talks/12823

¹³<http://typesafe.com/activator>

¹⁴<https://class.coursera.org/reactive-001/class>

¹⁵<http://letitcrash.com/post/64667109914/>
series-the-neophytes-guide-to-scala

¹⁶<http://letitcrash.com/post/59190266995/>



Ссылки

- «Потоковая обработка данных с помощью модели акторов (Actor Model)» на ADD-3¹²
- Typesafe Activator¹³
- Principles of Reactive Programming¹⁴
- Книги:
 - Jamie Allen. *Effective Akka*. 2013

¹²http://addconf.ru/talk.sdf/add/add_3/talks/12823

¹³<http://typesafe.com/activator>

¹⁴<https://class.coursera.org/reactive-001/class>

¹⁵<http://letitcrash.com/post/64667109914/>
series-the-neophytes-guide-to-scala

¹⁶<http://letitcrash.com/post/59190266995/>



Ссылки

- «Потоковая обработка данных с помощью модели акторов (Actor Model)» на ADD-3¹²
- Typesafe Activator¹³
- Principles of Reactive Programming¹⁴
- Книги:
 - Jamie Allen. *Effective Akka*. 2013
 - Derek Wyatt. *Akka Concurrency*. 2013

¹²http://addconf.ru/talk.sdf/add/add_3/talks/12823

¹³<http://typesafe.com/activator>

¹⁴<https://class.coursera.org/reactive-001/class>

¹⁵<http://letitcrash.com/post/64667109914/>
series-the-neophytes-guide-to-scala

¹⁶<http://letitcrash.com/post/59190266995/>

Ссылки

- «Потоковая обработка данных с помощью модели акторов (Actor Model)» на ADD-3¹²
- Typesafe Activator¹³
- Principles of Reactive Programming¹⁴
- Книги:
 - Jamie Allen. *Effective Akka*. 2013
 - Derek Wyatt. *Akka Concurrency*. 2013
 - Series: The Neophyte's Guide to Scala¹⁵

¹²http://addconf.ru/talk.sdf/add/add_3/talks/12823

¹³<http://typesafe.com/activator>

¹⁴<https://class.coursera.org/reactive-001/class>

¹⁵<http://letitcrash.com/post/64667109914/series-the-neophytes-guide-to-scala>

¹⁶<http://letitcrash.com/post/59190266995/>



Ссылки

- «Потоковая обработка данных с помощью модели акторов (Actor Model)» на ADD-3¹²
- Typesafe Activator¹³
- Principles of Reactive Programming¹⁴
- Книги:
 - Jamie Allen. *Effective Akka*. 2013
 - Derek Wyatt. *Akka Concurrency*. 2013
 - Series: The Neophyte's Guide to Scala¹⁵
 - EAI Patterns Series¹⁶

¹²http://addconf.ru/talk.sdf/add/add_3/talks/12823

¹³<http://typesafe.com/activator>

¹⁴<https://class.coursera.org/reactive-001/class>

¹⁵<http://letitcrash.com/post/64667109914/>

[series-the-neophytes-guide-to-scala](http://letitcrash.com/post/64667109914/series-the-neophytes-guide-to-scala)

¹⁶<http://letitcrash.com/post/59190266995/>



Вопросы?

- <http://incubos.org/contacts/>
- @incubos
- vadim.tsesko@gmail.com



Alan Kay on OOP, 1967

I thought of objects being like biological cells and/or individual computers on a network, only able to communicate with messages (so messaging came at the very beginning – it took a while to see how to do messaging in a programming language efficiently enough to be useful).

OOP to me means only messaging, local retention and protection and hiding of state-process, and extreme late-binding of all things. It can be done in Smalltalk and in LISP. There are possibly other systems in which this is possible, but I'm not aware of them.

