Потоковая обработка данных с помощью Actor Model

Вадим Цесько
incubos@yandex.ru
twitter.com/incubos

Яндекс

12 мая 2012 г.

Application Developer Days

Класс систем

Мы разрабатываем

- **Яндекс.Авто**
- Яндекс.Недвижимость
- Яндекс.Работа

Application Developer Days

Класс систем

Мы разрабатываем

- **Яндекс.Авто**
- Яндекс.Недвижимость
- Яндекс.Работа

ETL-процесс:

Application Developer Days

Класс систем

Мы разрабатываем

- **Яндекс.Авто**
- Яндекс.Недвижимость
- Яндекс.Работа

ETL-процесс:

1. **Extract** — загрузка внешних данных

Application Developer Days

Класс систем

Мы разрабатываем

- **Яндекс.Авто**
- Яндекс.Недвижимость
- Яндекс.Работа

ETL-процесс:

1. **Extract** — загрузка внешних данных
2. **Transform** — унификация, кластеризация, ...

Application Developer Days

Класс систем

Мы разрабатываем

- **Яндекс.Авто**
- Яндекс.Недвижимость
- Яндекс.Работа

ETL-процесс:

1. **Extract** — загрузка внешних данных
2. **Transform** — унификация, кластеризация, ...
3. **Load** — построение и раскладка индекса

Application Developer Days

Статистика

Данные:

- До 10^4 источников
- До 10^7 сущностей
- 20% обновляется ежедневно
- $2x$ рост за год

Application Developer Days

[1] `http://stat.yandex.ru`

Статистика

Данные:

- До 10^4 источников
- До 10^7 сущностей
- 20% обновляется ежедневно
- $2x$ рост за год

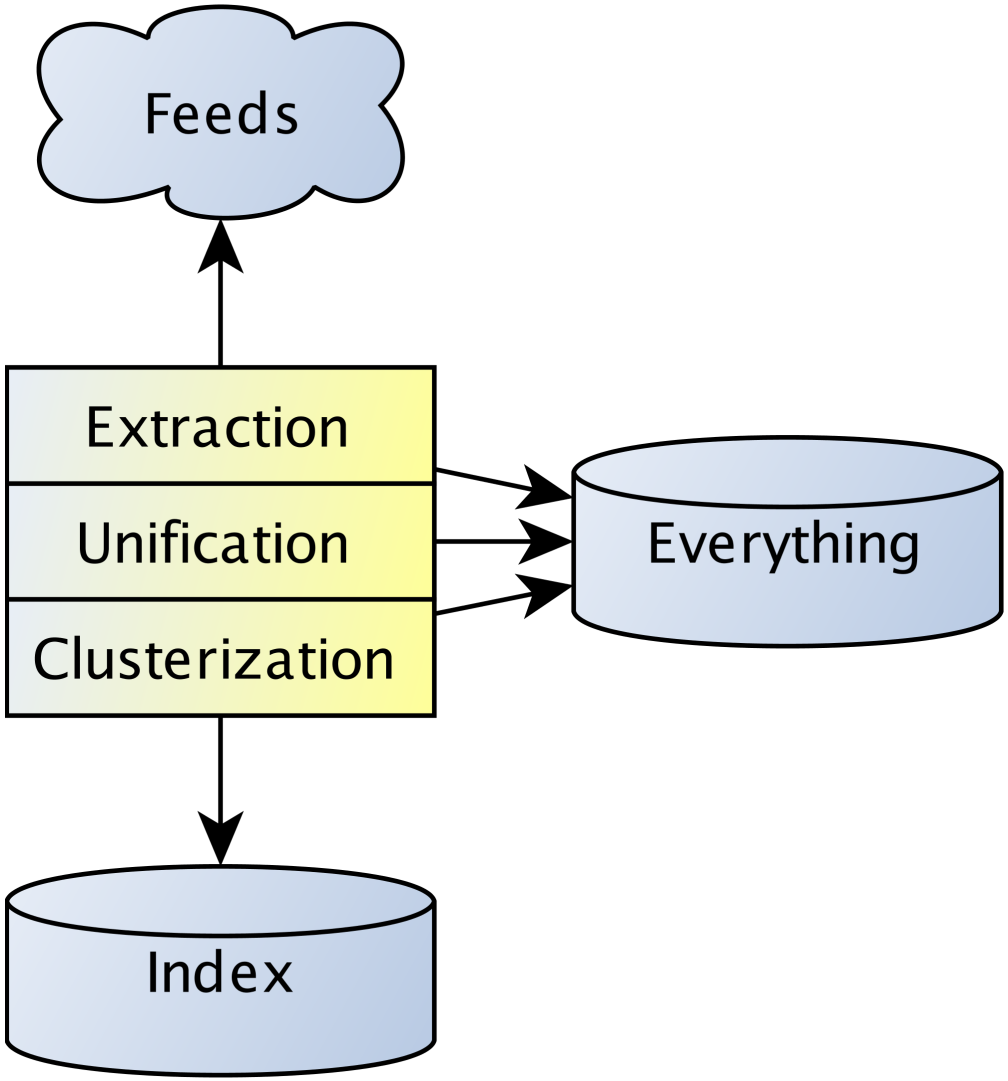
Пользователи[1]:

- Яндекс.Авто — 4.5 млн. чел.
- Яндекс.Недвижимость — 2 млн. чел.
- Яндекс.Работа — 3.3 млн. чел.

Application Developer Days

[1] `http://stat.yandex.ru`

Было



Application Developer Days

За кадром

- Кластер поисковых машин
  - Раскладка
  - Балансировка
- Дополнительные процессы и данные
  - Статистика
  - Картинки
  - Цены
  - Качество
  - Админки/партнёрки
  - История

Application Developer Days

Мотивация

Application Developer Days

Мотивация

- Масштабируемость с ростом объёма данных

Application Developer Days

Мотивация

- Масштабируемость с ростом объёма данных
- Инкрементальная индексация

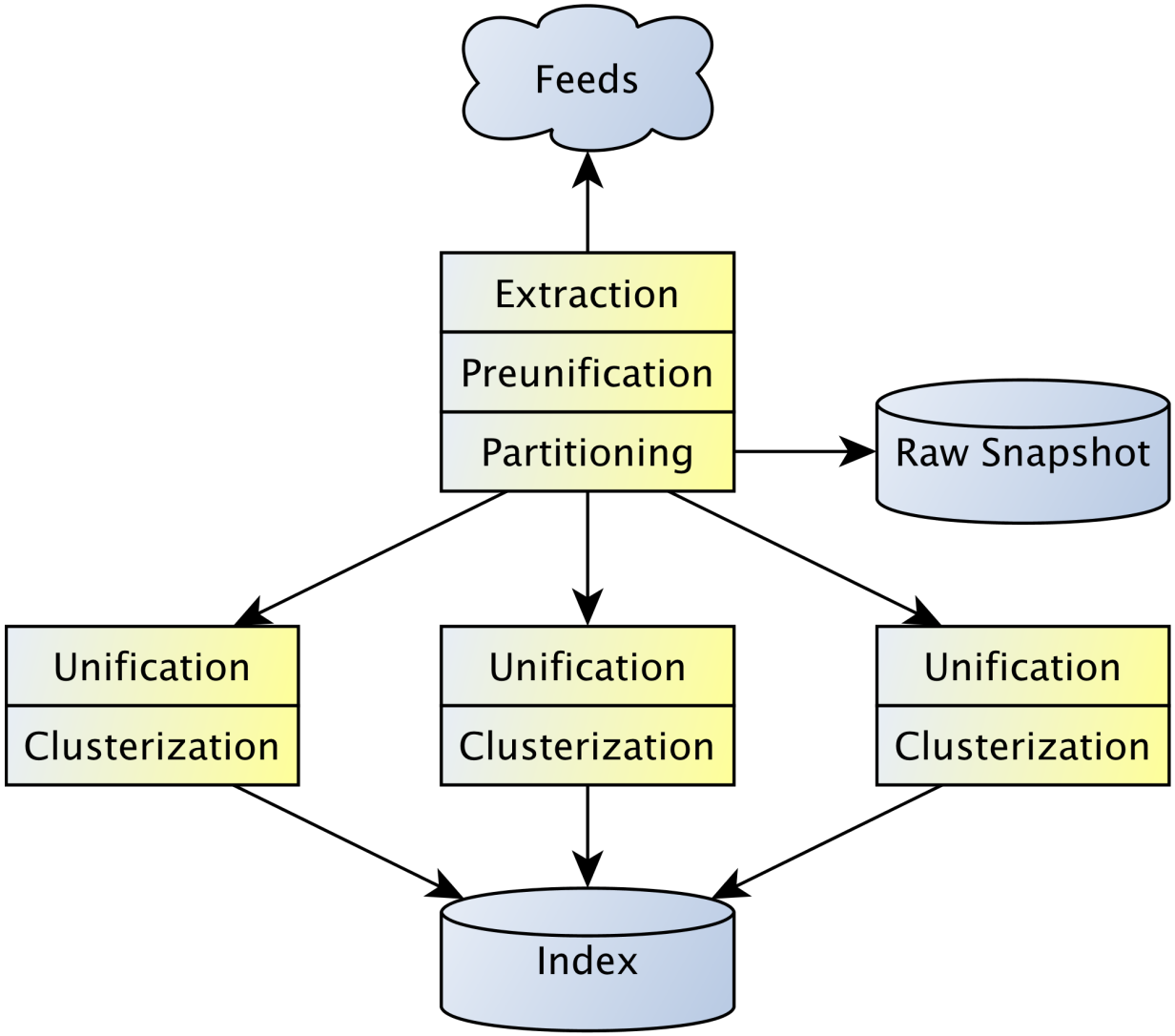
Application Developer Days

Мотивация

- Масштабируемость с ростом объёма данных
- Инкрементальная индексация
- (мягкий) real-time

Application Developer Days

Стало


Feeds
Extraction
Preunification
Partitioning
Raw Snapshot
Unification
Clusterization
Unification
Clusterization
Unification
Clusterization
Index


Application Developer Days

Свойства must have

Application Developer Days

Свойства must have

- Масштабируемость

Application Developer Days

Свойства must have

- Масштабируемость
- Гибкость

Application Developer Days

Свойства must have

- Масштабируемость
- Гибкость
- Устойчивость к сбоям

Application Developer Days

Свойства must have

- Масштабируемость
- Гибкость
- Устойчивость к сбоям
- Тестируемость

Application Developer Days

Свойства must have

- Масштабируемость
- Гибкость
- Устойчивость к сбоям
- Тестируемость
- Наблюдаемость

Application Developer Days

Происхождение

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

Application Developer Days

Происхождение

- Carl Hewitt, Peter Bishop and Richard Steiger. A Universal Modular Actor Formalism for Artificial Intelligence. 1973.
- Изначально для описания параллельных вычислений

Application Developer Days

Происхождение

- Carl Hewitt, Peter Bishop and Richard Steiger. A Universal Modular Actor Formalism for Artificial Intelligence. 1973.
- Изначально для описания параллельных вычислений
- Позднее основа для реализаций

Application Developer Days

Происхождение

- Carl Hewitt, Peter Bishop and Richard Steiger. A Universal Modular Actor Formalism for Artificial Intelligence. 1973.
- Изначально для описания параллельных вычислений
- Позднее основа для реализаций

Don't panic

Алгебры, логики, семантики и остальное оставим за кадром.

Что такое Actor

- Всё это актор[2]

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

Application Developer Days

Что такое Actor

- Всё это актор[2]
- Функционируют параллельно

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

Application Developer Days

Что такое Actor

- Всё это актор[2]
- Функционируют параллельно
- Асинхронно обмениваются сообщениями

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

Application Developer Days

Что такое Actor

- Всё это актор[2]
- 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

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

Application Developer Days

Что такое Actor

- Всё это актор[2]
- 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

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

Application Developer Days

Что такое Actor

- Всё это актор[2]
- 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»

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

Application Developer Days

Реализации Actor Model

Application Developer Days

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

Реализации Actor Model

Языки[3] с «родной» поддержкой:

- Erlang
- Scala
- ...

Application Developer Days

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

Реализации Actor Model

Языки[3] с «родной» поддержкой:

- Erlang
- Scala
- ...

Библиотеки для языков:

- Scala
- Java
- F#
- ...

Application Developer Days

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

Реализации Actor Model

Языки[3] с «родной» поддержкой:

- Erlang
- Scala
- ...

Библиотеки для языков:

- Scala
- Java
- F#
- ...

Мы выбрали

Akka (Scala API).

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

О проекте

Jonas Bonér:

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

Application Developer Days

О проекте

Jonas Bonér:

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

Ресурсы:

- `http://akka.io/docs/`
- `http://letitcrash.com/`

Application Developer Days

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

Application Developer Days

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

Application Developer Days

Производительность

Внимание

Синтетические тесты ☺

Erlang R14B04 vs **Akka** 2.0-SNAPSHOT[4]:

[4] http://letitcrash.com/post/14783691760/akka-vs-erlang

[5] http://letitcrash.com/post/20397701710/50-million-messages-per-second-on-a-single-machine

Application Developer Days

Производительность

Внимание

Синтетические тесты ☺

Erlang R14B04 vs **Akka** 2.0-SNAPSHOT[4]:

- 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

Application Developer Days

Производительность

Внимание

Синтетические тесты ☺

Erlang R14B04 vs **Akka** 2.0-SNAPSHOT[4]:

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

Akka 2.0[5]:

[4] http://letitcrash.com/post/14783691760/akka-vs-erlang

[5] http://letitcrash.com/post/20397701710/50-million-messages-per-second-on-a-single-machine

Application Developer Days

Производительность

Внимание

Синтетические тесты ☺

Erlang R14B04 vs **Akka** 2.0-SNAPSHOT[4]:

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

Akka 2.0[5]:

- **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

Application Developer Days

Производительность

Внимание

Синтетические тесты ☺

Erlang R14B04 vs **Akka** 2.0-SNAPSHOT[4]:

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

Akka 2.0[5]:

- **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`

Application Developer Days

Особенности реализации Actor Model

Application Developer Days

Особенности реализации Actor Model

- **300 байт** на актор

Application Developer Days

Особенности реализации Actor Model

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

Application Developer Days

Особенности реализации Actor Model

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

Application Developer Days

Особенности реализации Actor Model

- **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

Application Developer Days

Особенности реализации Actor Model

- **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**

Application Developer Days

Особенности реализации Actor Model

- **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

Application Developer Days

Особенности реализации Actor Model

- **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`

Application Developer Days

Ссылки

- Чисто локальные
- Локальные ссылки
- Ссылки для маршрутизации (`Router`)
- Ссылки на удалённых акторов
- Особые: `PromiseActorRef`, `DeadLetterActorRef`, `EmptyLocalActorRef` (`DeadLetterActorRef`)

Application Developer Days

Ссылки

- Чисто локальные
- Локальные ссылки
- Ссылки для маршрутизации (`Router`)
- Ссылки на удалённых акторов
- Особые: `PromiseActorRef`, `DeadLetterActorRef`, `EmptyLocalActorRef` (`DeadLetterActorRef`)

Примеры путей

- `akka://system/user/service/worker`
- `akka://system@server.yandex.ru:2552/user/service`

Application Developer Days

Конструирование ссылок

- Создание акторов: `ActorSystem.actorOf` или `ActorContext.actorOf`
- Поиск акторов: `ActorSystem.actorFor` или `ActorContext.actorFor`
- Каждый актор знает себя, родителя и детей

Application Developer Days

Конструирование ссылок

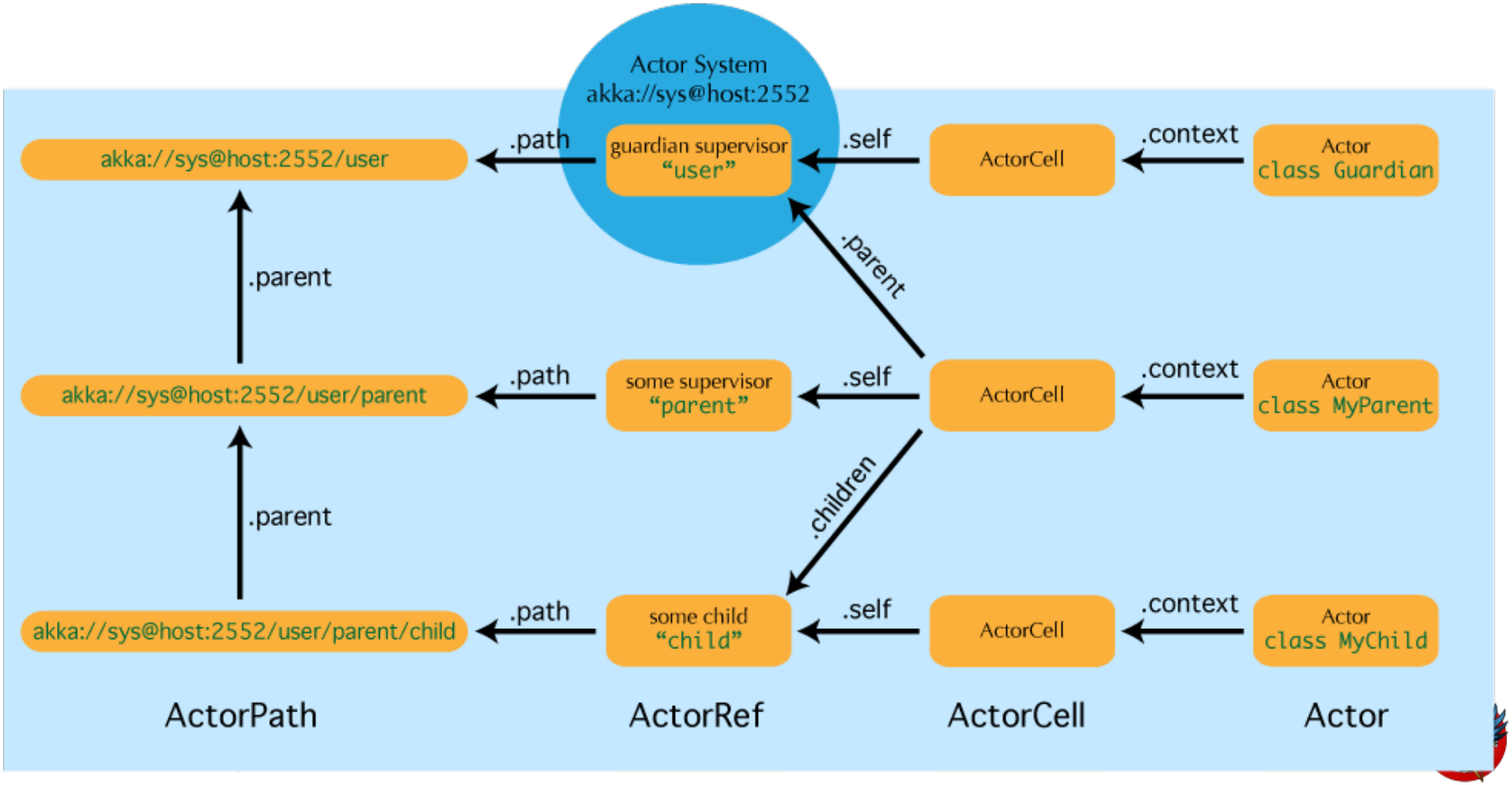
- Создание акторов: `ActorSystem.actorOf` или `ActorContext.actorOf`
- Поиск акторов: `ActorSystem.actorFor` или `ActorContext.actorFor`
- Каждый актор знает себя, родителя и детей

Можно делать так:

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

Application Developer Days

Именование в локальной системе



Пути

Типы путей:

- Логический
- Физический

Application Developer Days

Пути

Типы путей:

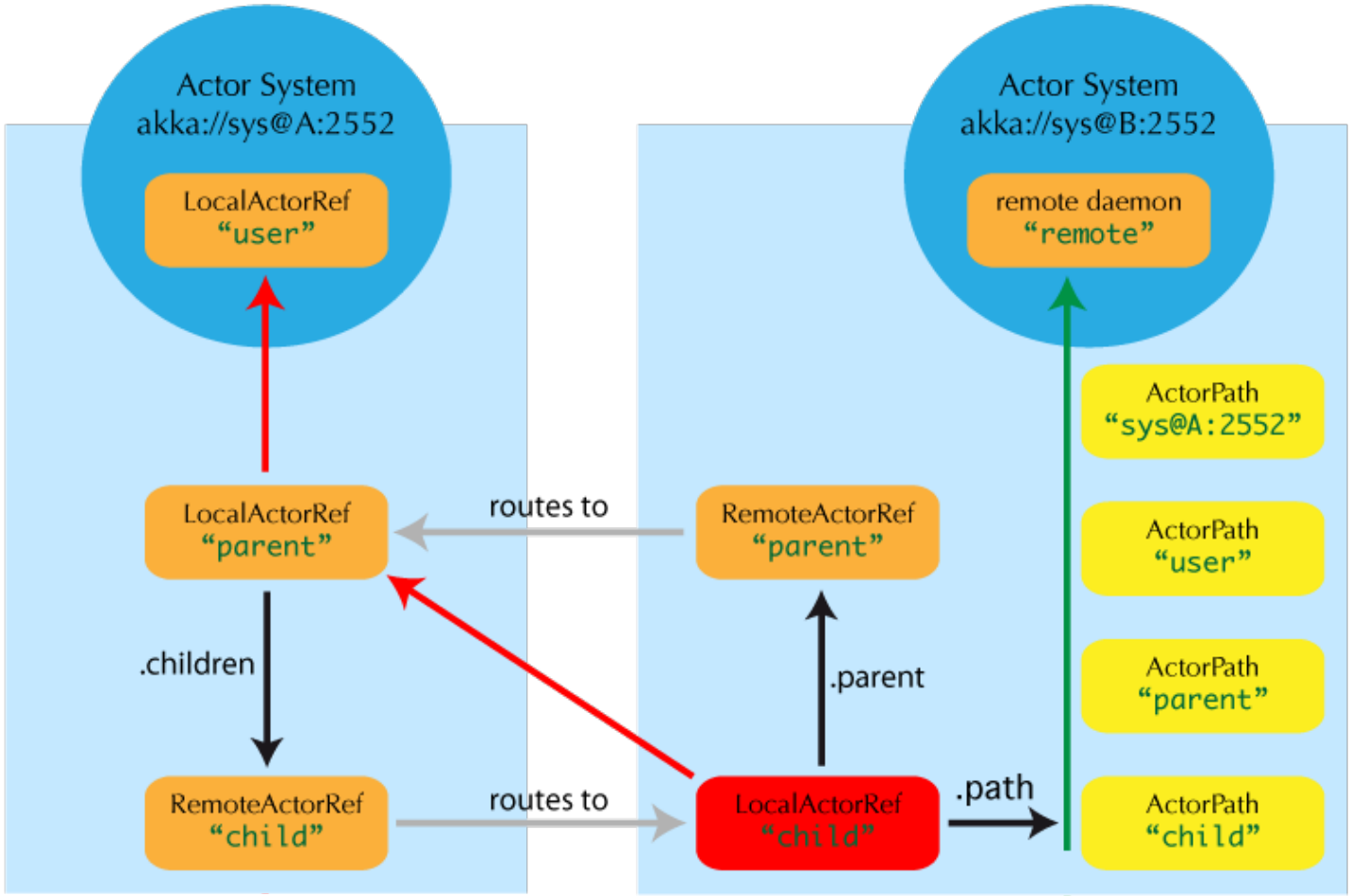
- Логический
- Физический

Особые пути:

- `/user`
- `/system`
- `/deadLetters`
- `/temp`
- `/remote`

Application Developer Days

Удалённое развёртывание



logical actor path: akka://sys@A:2552/user/parent/child

physical actor path: akka://sys@B:2552/remote/sys@A:2552/user/parent/child

Application Developer Days

Actor Best Practices

Application Developer Days

Actor Best Practices

- Не блокироваться

Application Developer Days

Actor Best Practices

- Не блокироваться
- Неизменяемые сообщения

Application Developer Days

Actor Best Practices

- Не блокироваться
- Неизменяемые сообщения
- Нет разделяемому состоянию

Application Developer Days

Actor Best Practices

- Не блокироваться
- Неизменяемые сообщения
- Нет разделяемому состоянию
- «Опасные» подзадачи в дочерние акторы

Application Developer Days

Actor Best Practices

- Не блокироваться
- Неизменяемые сообщения
- Нет разделяемому состоянию
- «Опасные» подзадачи в дочерние акторы
- События жизненного цикла

Application Developer Days

Используемые фичи

- Actors
- Logging
- Scheduler
- Dispatchers
- Routing
- Remoting
- Serialization
- Testing
- FSM
- Fault Tolerance

Application Developer Days

Неиспользуемые фичи

- Typed Actors
- Event Bus
- Futures
- Dataflow Concurrency
- STM
- Agents
- Transactors
- Durable Mailboxes (file, Redis, ZooKeeper, Mongo)
- **Akka Cluster**

Application Developer Days

Определение актора

```
1  class Partitioner(partitionStorage: ActorRef) extends Actor {
2
3    lazy val log = Logging(context.system, this)
4
5    def receive = {
6      case PartitionFeed(partner, offers) =>
7          partition(partner, offers)
8      case msg =>
9          log.error("Unsupported message received: {}", msg)
10   }
11
12   def partition(partner: Partner, offers: Traversable[Offer]) {
13     ...
14
15     partitionStorage ! UpdatePartitions(partner, partitioning)
16   }
17 }
```

Application Developer Days

Создание актора

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

Application Developer Days

Конфигурация диспетчера

Конфигурация в HOCON[6]:

```
dispatcher {
  cpu {
    type = Dispatcher
    executor = "fork-join-executor"
    mailbox-capacity = 4
    mailbox-push-timeout-time = 5m

    fork-join-executor {
      parallelism-min = 4
      parallelism-factor = 1.0
    }
  }
}
```

[6]Human-Optimized Config Object Notation: https://github.com/typesafehub/config

Application Developer Days

Диспетчеры и ящики

Диспетчеры:

- `Dispatcher`
- `PinnedDispatcher`
- `BalancingDispatcher`
- `CallingThreadDispatcher`

Почтовые ящики:

- `UnboundedMailbox`
- `BoundedMailbox`
- `UnboundedPriorityMailbox`
- `BoundedPriorityMailbox`
- Durable Mailboxes

Application Developer Days

Конфигурация маршрутизатора

```
akka.actor.deployment {
  /partitioner {
    router = smallest-mailbox
    nr-of-instances = 4
  }
}
```

```
akka.actor.deployment {
  /unifier {
    router = round-robin
    resizer {
      lower-bound = 2
      upper-bound = 16
    }
  }
}
```

Application Developer Days

Маршрутизаторы

- `RoundRobinRouter`
- `RandomRouter`
- `SmallestMailboxRouter`
- `BroadcastRouter`
- `ScatterGatherFirstCompletedRouter`

Application Developer Days

Конфигурирование из кода

```
1 val shardActors =
2   shards.map(system.actorFor(_)).toIndexedSeq
3
4 val shard = system.actorOf(
5   Props[PartitionReceiver]
6       .withRouter(RoundRobinRouter(shardActors))
7       .withDispatcher("dispatcher.shard"),
8   "shard")
```

Application Developer Days

Распределённые акторы

```
akka {
  actor {
    provider = "akka.remote.RemoteActorRefProvider"
  }

  remote {
    transport = "akka.remote.netty.NettyRemoteTransport"

    netty {
      hostname = "server.yandex.ru"
      port = 2553
    }
  }
}
```

Application Developer Days

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 }
```

Application Developer Days

Тестирование акторов

Application Developer Days

Тестирование акторов

- Модульное тестирование с `TestActorRef`

Application Developer Days

Тестирование акторов

- Модульное тестирование с `TestActorRef`
- Интеграционное тестирование с `Probe`

Application Developer Days

Тестирование акторов

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

Application Developer Days

Тестирование акторов

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

Application Developer Days

Тестирование акторов

- Модульное тестирование с `TestActorRef`
- Интеграционное тестирование с `Probe`
- Проверки с сопоставлением по шаблону
- Замедление времени
- Детализированные логи `akka.actor.debug.*`

Application Developer Days

Тестирование акторов

- Модульное тестирование с `TestActorRef`
- Интеграционное тестирование с `Probe`
- Проверки с сопоставлением по шаблону
- Замедление времени
- Детализированные логи `akka.actor.debug.*`
- `CallingThreadDispatcher`

Application Developer Days

Супервизор

Решение при сбое:

1. `Resume`
2. `Restart`
3. `Stop`
4. `Escalate`

Application Developer Days

Супервизор

Решение при сбое:

1. `Resume`
2. `Restart`
3. `Stop`
4. `Escalate`

- Принятое решение (1-3) действует рекурсивно

Application Developer Days

Супервизор

Решение при сбое:

1. `Resume`
2. `Restart`
3. `Stop`
4. `Escalate`

- Принятое решение (1-3) действует рекурсивно
- Функция `Exception` $\Rightarrow$ `Directive`

Application Developer Days

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`

Application Developer Days

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`

Application Developer Days

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

Application Developer Days

Супервизор по умолчанию

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

Application Developer Days

Наблюдаемость

- Typesafe Console для мониторинга

Application Developer Days

Наблюдаемость

- Typesafe Console для мониторинга
- **Graphite** + `statsd` + `statsd-over-slf4j`

Application Developer Days

Свойства must have

Application Developer Days

Свойства must have

- Масштабируемость

Application Developer Days

Свойства must have

- Масштабируемость
- Гибкость

Application Developer Days

Свойства must have

- Масштабируемость
- Гибкость
- Устойчивость к сбоям

Application Developer Days

Свойства must have

- Масштабируемость
- Гибкость
- Устойчивость к сбоям
- Тестируемость

Application Developer Days

Свойства must have

- Масштабируемость
- Гибкость
- Устойчивость к сбоям
- Тестируемость
- Наблюдаемость

Application Developer Days

Акторы

Application Developer Days

Акторы

- Храните состояние вовне

Application Developer Days

Акторы

- Храните состояние вовне
- Стройте **всю систему** на акторах

Application Developer Days

Акторы

- Храните состояние вовне
- Стройте **всю систему** на акторах
- Пишите асинхронный код

Application Developer Days

Акторы

- Храните состояние вовне
- Стройте **всю систему** на акторах
- Пишите асинхронный код
- Избегайте косвенного взаимодействия акторов

Application Developer Days

Акторы

- Храните состояние вовне
- Стройте **всю систему** на акторах
- Пишите асинхронный код
- Избегайте косвенного взаимодействия акторов
- Баги есть, но быстро чинят

Application Developer Days

Память и ящики

[7] http://letitcrash.com/post/17707262394/why-no-mailboxsize-in-akka-2

Application Developer Days

Память и ящики

- Неограниченные ящики $\Rightarrow$ неограниченная память при перегрузке

[7] http://letitcrash.com/post/17707262394/why-no-mailboxsize-in-akka-2

Память и ящики

- Неограниченные ящики $\Rightarrow$ неограниченная память при перегрузке
- Ограниченные ящики $\Rightarrow$ возможные deadlock'и при наличии циклов

[7] http://letitcrash.com/post/17707262394/why-no-mailboxsize-in-akka-2

Application Developer Days

Память и ящики

- Неограниченные ящики ⇒ неограниченная память при перегрузке
- Ограниченные ящики ⇒ возможные deadlock'и при наличии циклов
- ⇒ стройте системы без циклов ☺

[7] http://letitcrash.com/post/17707262394/why-no-mailboxsize-in-akka-2

Память и ящики

- Неограниченные ящики ⇒ неограниченная память при перегрузке
- Ограниченные ящики ⇒ возможные deadlock'и при наличии циклов
- ⇒ стройте системы без циклов ☺
- Существует диспетчер по умолчанию

[7] http://letitcrash.com/post/17707262394/why-no-mailboxsize-in-akka-2

Application Developer Days

Память и ящики

- Неограниченные ящики ⇒ неограниченная память при перегрузке
- Ограниченные ящики ⇒ возможные deadlock'и при наличии циклов
- ⇒ стройте системы без циклов ☺
- Существует диспетчер по умолчанию
- Нет доступа к размеру ящика[7]

[7] http://letitcrash.com/post/17707262394/why-no-mailboxsize-in-akka-2

Application Developer Days

Удалённые акторы

Application Developer Days

Удалённые акторы

- Remote deploy пока не работает в полной мере

Application Developer Days

Удалённые акторы

- Remote deploy пока не работает в полной мере
- На клиенте нужно иметь код удалённо развёртываемого актора (WTF?!)

Application Developer Days

Удалённые акторы

- Remote deploy пока не работает в полной мере
- На клиенте нужно иметь код удалённо развёртываемого актора (WTF?!)
- Маршрутизатор `smallest-mailbox` не работает для удалённых акторов

Application Developer Days

Недостатки реализации

Application Developer Days

Недостатки реализации

- Отсутствует явная структура системы

Application Developer Days

Недостатки реализации

- Отсутствует явная структура системы
- Нет статической проверки протоколов

Application Developer Days

Недостатки реализации

- Отсутствует явная структура системы
- Нет статической проверки протоколов
- Пока неполноценная реализация удалённого деплоя и апгрейда

Application Developer Days

Недостатки реализации

- Отсутствует явная структура системы
- Нет статической проверки протоколов
- Пока неполноценная реализация удалённого деплоя и апгрейда
- Нет развитых средств анализа производительности системы

Application Developer Days

Достоинства подхода

Application Developer Days

Достоинства подхода

- Выразительное описание поведения и взаимодействия

Application Developer Days

Достоинства подхода

- Выразительное описание поведения и взаимодействия
- Простое конфигурирование

Application Developer Days

Достоинства подхода

- Выразительное описание поведения и взаимодействия
- Простое конфигурирование
- Гибкое управления выполнением

Application Developer Days

Достоинства подхода

- Выразительное описание поведения и взаимодействия
- Простое конфигурирование
- Гибкое управления выполнением
- Масштабируется локально и распределённо

Application Developer Days

Достоинства подхода

- Выразительное описание поведения и взаимодействия
- Простое конфигурирование
- Гибкое управления выполнением
- Масштабируется локально и распределённо
- Работает ☺

Application Developer Days

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.

Application Developer Days