Инженерный вестник Дона, №5 (2019)
ivdon.ru/ru/magazine/archive/n5y2019/6004

Реализация приложений Интернета вещей агентной моделью *Akka*

А.М. Лященко, А.Н. Пачев, Л.Х. Манучарян

Ростовский государственный университет путей сообщения

Аннотация: В статье рассматриваются технологии, работающие с интернетом вещей, и их применении в различных отраслях. Интернет вещей рассматривается как автоматизация более высокого уровня, следовательно, становится возможным автоматизированное принятие сложных логических решений, связанных с выбором структуры процесса и операций, назначением технологических баз и других подобных задач. Рассмотрена модель, основанная на универсальном примитиве, называемом Актором для параллельных и распределенных вычислений. Освещены вопросы реализации агентного подхода для приложений интернета вещей. Установлено необходимое условие реализации модели *IoT*, это возможность масштабируемости и наличие параметров отказоустойчивости. Чтобы удовлетворить реальную потребность в масштабируемости, а также отказоустойчивости, *Akka* предоставляет комплексные функции в маршрутизации, кластеризации, окантовке и сохранении агентов. Проведен анализ использования фреймворка Akka для реализации агентов.
Ключевые слова: автоматизация, управление, технология, анализ, интернет вещей, агент, примитив, реализация, инструментарий, масштабируемость, *Akka*, *IoT*.

Для облегчения целенаправленной деятельности человека, окруженного множеством объектов, на которые направлена его деятельность, широко разработана система автоматизации процессов. В связи с быстрым развитием в последнее время глобальной сети Интернет стала возможным концепция Интернета вещей (*Internet of Things* или (*IoT*)), которая базируется на принципе межмашинного общения. В таком случае отпадает необходимость вмешательства человека, и электронные устройства могут самостоятельно связываться между собой, т.е. Интернет вещей можно рассматривать как автоматизацию более высокого уровня. Автоматизация высокого уровня может быть достигнута при заполнении базы знаний. В этом случае становится возможным автоматизированное принятие сложных логических решений, связанных, например, с выбором структуры процесса и операций, назначением технологических баз и другие подобные задачи. Процесс принятия таких решений полностью автоматизировать не удается, поэтому режим диалога между агентами остается даже на самом высоком

уровне автоматизации [1–5].

На сегодняшний день гибкость и скорость внедрения инноваций являются ключевыми факторами успеха не только любого производства, но и экономики в целом. В этом смысле «умным» средам принадлежит особая роль: по сути, они выполняют функцию каркаса, на который в ближайшем будущем будет крепиться и тем самым обеспечивать новое качество продукции не только сама промышленность, но и транспортная и энергетическая инфраструктура.

Большинство технологий «умных» сред находится на достаточно раннем этапе своего развития, им еще предстоит преодолеть многие ограничения технического и регулятивного характера. При этом уже сегодня существуют отдельные примеры их эффективного применения в промышленности, на транспорте и в энергетике. Об этом свидетельствуют соответствующие программы и проекты в США, Европейском союзе, прочих странах ОЭСР, гибкие инструменты государственной политики, созданная нормативно-правовая база, включающая необходимые стандарты.

По мере того, как интерес к Интернету вещей (*IoT*) проявляется по всем отраслям, компании от небольших стартапов до промышленных гигантов спешат запускать свои продукты *IoT*. Быстрые успехи в Интернет-инфраструктуре, облачных вычислениях, пропускной способности соединения и мобильных устройствах на протяжении многих лет помогли сделать *IoT* реальным. Учитывая обилие постоянно развивающихся вычислительных технологий, существует множество вариантов вычислительных моделей и платформ для проектирования и внедрения продукта *IoT* [6].

Для построения полностью взаимосвязанного будущего Интернета (*IoT*) агентная вычислительная модель выделяется из остальных. Модель основывается на универсальном примитиве, называемого Актором [7] для

параллельных и распределенных вычислений. Он обеспечивает альтернативу более традиционной модели параллелизма, которая основана на синхронизации общего измененного состояния с использованием блокировок. В частности, управляемый сообщениями стиль неблокирующих взаимодействий посредством непосредственных сообщений между агентами хорошо связан с современными подходами к программированию на сложных распределенных платформах.

Одной из основных характеристик типичной системы *IoT* является то, что она включает в себя большое количество управляемых устройств, в каждом из которых подразумеваются изменения внутреннего состояния. Во многих случаях эти устройства являются примитивным оборудованием, работающим на каком-то простом сетевом протоколе из-за ограничений в стоимости мощности. Например, термостаты, управляемые *SaaS* от *EcoFactor*, сочетают использование не очень дорогостоящего оборудования термостата и маломощного протокола *WPAN* (*Wireless Personal Area Network*) под названием *ZigBee*.

В агентной модели разбиение бизнес-логики на минимальные задачи для отдельных действующих лиц является частью основополагающего принципа модели. Агенты очень просты в проектировании; следовательно, могут масштабироваться, не потребляя чрезмерных вычислительных ресурсов. Другим ключевым свойством агентов является их неблокирующая связность посредством передачи сообщений. Эти атрибуты делают их пригодными для построения распределенных вычислительных систем [8].

Каждый агент может порождать дочерних агентов с программируемыми стратегиями наблюдения, подходящими для имитации менеджеров устройств и иерархических групп устройств *IoT*, при этом главной особенностью агента является способность поддерживать свое внутреннее состояние в практически собственном потоке, изолированном от

остальной системы. Все эти характеристики делают агентов весьма подходящими для имитации устройств *IoT*, каждый из которых состоит из собственного состояния и логики управления.

На сегодняшний день агенты *Akka* являются наиболее совершенной агентской реализацией, предоставляя надежный инструментарий на *JVM* (виртуальная машина *Java*). Написанная в *Scala*, *Akka* обладает всеми достоинствами, которые может предложить стандартная модель агентов. У всех участников есть четко определенный жизненный цикл с усовершенствованными методами, такими как *preStart*, *postRestart* и *postStop* для управления логикой жизненного цикла. Благодаря этим навыкам управления жизненным циклом и всесторонним стратегиям надзора за агентами можно полностью настроить поведение агента на протяжении всего жизненного цикла. В случае имитации устройства *IoT* можно легко привязать к соответствующим обработчикам сообщений.

Приложения *Akka* могут выглядеть немного «нетрадиционными» без знания событийно-ориентированного (*event-driven*) программирования. Тем не менее, с точки зрения высокого уровня, стандартный фрагмент кода актора может быть совершенно понятным. Например, примитив устройства, написанный на *Scala* представлен на рис. 1.

В приведенном выше фрагменте показано, как можно создать примитив-актор в *Scala – Akka*, который публикует подписчикам свою информацию о состоянии работы с использованием стандартного протокола обмена сообщениями для подписки на подписку на сообщения *MQTT* (*Message Queue Telemetry Transport*). Цель здесь заключается не в том, чтобы изучить, как программировать в *Scala* или *Akka*, а в том, чтобы представить пример жизненного цикла агента *Акка*.

```
object Device {
  def props(deviceType: String, mqttPubSub: ActorRef) = // ...
}

class Device(deviceType: String, mqttPubSub: ActorRef) extends Actor
  import Device._

  private var opState: OpState = InitialState(deviceType)
  override def preStart(): Unit = // Initialize device's op-state ...
  override def postStop(): Unit = // Reset/Shutdown device ...

  def receive = {
    case ReportOpState =>
      // Assemble report data with opState ...
      mqttPubSub ! new Publish(Mqtt.topic_report, reportData)
    case UpdateOpState(newState) =>
      // Update opState with newState ...
      mqttPubSub ! new Publish(Mqtt.topic_update, updateResult)
    case PowerOff =>
      // Shutdown device ...
  }
}
```

Рис. 1. – Пример кода примитива, написанный на *Scala*

Агенты *Akka* предназначены не только для выполнения упрощенных задач, они способны использовать полный набор функциональных возможностей, предоставляемый *Akka*. Такая гибкость позволяет регулировать рабочую нагрузку агента в соответствии с бизнес-требованиями. Например, агент может быть отдельным устройством *IoT*, или он может представлять собой набор устройств, поддерживаемых в какой-то коллекции карт значений ключа.

Установлено, что сообщения, которые отправляются через участников, всегда должны быть неизменными [9,10]. Тем не менее, поскольку агенты работают практически в частных потоках, обычно безопасно поддерживать состояние агента, используя охраняемые частные переменные в классе агента. Но если возникает необходимость исключить использование изменяемых объектов, агенты *Akka* также оснащены функциональными возможностями логического потока, который эмулирует конечный автомат посредством горячей замены.

Akka предоставляет метод *context.become*, позволяющий внутреннему

состоянию подвергаться «горячей» замене внутри цикла сообщений агента. На рис. 2 приведен фрагмент примера, иллюстрирующий, как «горячая» замена имитирует конечный автомат без необходимости использования изменяемой переменной для поддержания состояния агента.

```
object Worker {
  def props(workerId, String, clusterClient: ActorRef) = // ...
}

class Worker(workerId, String, clusterClient: ActorRef) extends Acto
  import Worker._

  override def preStart(): Unit = // Initialize worker's operational s
  override def postStop(): Unit = // Terminate worker ...

  def sendToMaster(msg: Any): Unit = {
    clusterClient ! SendToAll("/user/master/singleton", msg)
  }

  def receive = idle

  def idle: Receive = {
    case WorkIsReady =>
      sendToMaster(WorkerIsFree(workerId))

    case Work =>
      // Process work ...
      workProcessor ! work
      context.become(working)
  }

  def working: Receive = {
    case WorkProcessed(workResult) =>
      sendToMaster(WorkIsDone(workerId, workResult))
      context.become(idle)

    case Work =>
      sendToMaster(WorkerIsBusy(workerId))
  }
}
```

Рис. 2. – Реализация конечного автомата

Необходимым условием реализации модели *IoT* является возможность масштабируемости и наличие параметров отказоустойчивости. Чтобы удовлетворить реальную потребность в масштабируемости, а также отказоустойчивости, *Akka* предоставляет комплексные функции в маршрутизации, кластеризации, окантовке и сохранении агентов.

Между тем, другие части системы *Akka* адресуют конкретные потребности, такие как потоковая передача или *REST API* в системе *IoT*.

«*Акка* потоки» – это полнофункциональный потоковый *API*, созданный на основе *Akka. Akka HTTP,* построенный поверх потоков *Akka*, обеспечивает потоковый интерфейс *REST / HTTP API*.

В заключении, если имеется необходимость построения масштабируемой системы *IoT*, которая включает отслеживание состояния устройства, следует обратиться к использованию стека технологий на основе агентов, такого как *Akka*.

Литература

1 Карташов, О.О., Бутакова М.А., Чернов А.В., Костюков А.В., Жарков Ю.И. Средства представления знаний и извлечения данных для интеллектуального анализа ситуаций // Инженерный вестник Дона, 2018, №4. URL: ivdon.ru/ru/magazine/archive/n4y2018/5421

2 Розин, М.Д., Свечкарев В.П. Анализ принципов формирования общедоступного интернет пространства в сфере инженерной деятельности // Инженерный вестник Дона, 2018, №1. URL: ivdon.ru/ru/magazine/archive/n1y2018/4750

3. Smith, P.C. On the representation of spatial uncertainty // Int. J. Robot. Res., Vol. vol. 5, 1987. pp. 56–68.

4. Aztiria A., Izaguirre A., Augusto J.C. Learning patterns in ambient intelligence environments// Artificial. Intelligence, Vol. 34, №1, pp. 35–51, 2010.

5. Лященко, А.М., Ковалев С.М. Гибридная модель слабоформализованного динамического процесса на основе нечеткой продукционной системы// Сб. науч. трудов. «Итоги и перспективы научных исследований». – Краснодар: Изд. Априори, 2014. – С. 183–192.

6. McEwen A., Cassimally H. Designing the Internet of Things. John Wiley and Sons, Ltd. – 2014. – 336 p.

7. Mozzaquatro, B.A., Jardim-Goncalves R., Agostinho C. Situation

awareness in the Internet of Things // 2017 International Conference on Engineering, Technology and Innovation (ICE/ITMC), Madeira Island, Portugal, 2017. – pp. 982–990. 8. Федотов И.Е., Модели параллельного программирования. – М.: СОЛОН-ПРЕСС, 2012. – 384 с.

9. Mozer, M.C. Lessons from an Adaptive Home // Wiley, New York, pp. 271–294, 2005.

10. Weiser M., Ubiquitous computing // Computer, Vol.26 №10, pp. 71–72, 1993.

References

1 Kartashov, O.O., Butakova M.A., CHernov A.V., Kostyukov A.V., ZHarkov YU.I. Inženernyj vestnik Dona (Rus), 2018, №4. URL: ivdon.ru/ru/magazine/archive/n4y2018/5421

2 Rozin, M.D., Svechkarev V.P. Inženernyj vestnik Dona (Rus), 2018, №1 URL: ivdon.ru/ru/magazine/archive/n1y2018/47503. A. Aztiria A., Izaguirre A., Augusto J.C. Artificial. Intelligence, Vol. 34, №1, 2010, pp. 35-51.

3. Smith, P.C. On the representation of spatial uncertainty. Int. J. Robot. Res., Vol. vol. 5, 1987. pp. 56-68.

4. Aztiria A., Izaguirre A., Augusto J.C. Learning patterns in ambient intelligence environments. Artificial. Intelligence, Vol. 34, №1, pp. 35–51, 2010.

5. Lyashchenko, A.M., Kovalev S.M. Sb. nauch. trudov. «Itogi i perspektivy nauchnyh issledovanij». Krasnodar: Izd. Apriori, 2014. pp. 183-192.

6. McEwen A., Cassimally H. Designing the Internet of Things. John Wiley and Sons, Ltd. 2014, 336 p.

7. Mozzaquatro, B.A., Jardim-Goncalves R., Agostinho C. 2017 International Conference on Engineering, Technology and Innovation (ICE/ITMC), Madeira Island, Portugal, 2017. pp. 982–990. 8. Fedotov I.E., Modeli parallel'nogo programmirovaniya [Parallel Programming Models]. M.: SOLON-PRESS, 2012. 84 p.

9. Mozer, M.C. Lessons from an Adaptive Home: Wiley, New York, 2005, pp. 271-294

1. 10. Weiser M., Ubiquitous computing: Computer, Vol.26 №10, 1993, pp. 71-72,