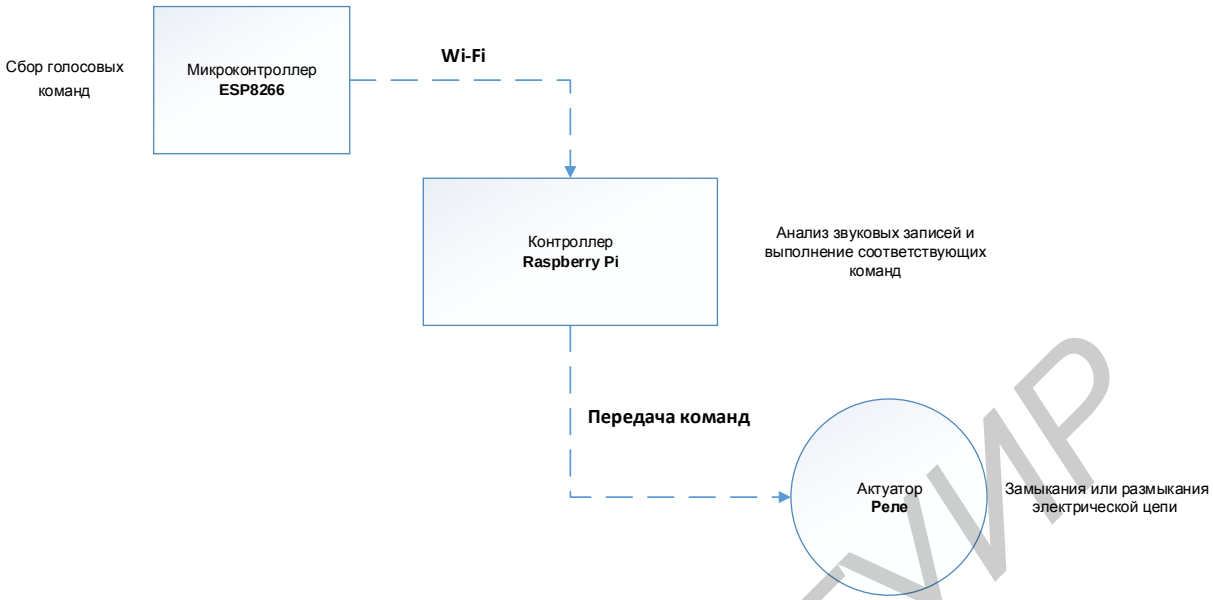

Рис. 2 – Схема аппаратно-программного комплекса

Наряду с преимуществами, которые предоставляет использование упомянутой системы, необходимо помнить о недостатках: необходимости обеспечения защиты элементов системы и общей высокой отказоустойчивости. В случае, если злоумышленнику удастся захватить управление системой, это вызовет угрозу безопасности и сохранности имущества.

Таким образом, описанный аппаратно-программный комплекс может быть внедрен в учебные помещения и лекционные аудитории, что позволит облегчить управление различными устройствами и системами, управление микроклиматом, повысить уровень безопасности помещений в сфере образования.

Список использованных источников:

1. Википедия Свободная Энциклопедия [Электронный ресурс]. - Режим доступа: https://ru.wikipedia.org/wiki/Умное_здание– Дата доступа: 01.04.2017.
2. Википедия Свободная Энциклопедия [Электронный ресурс]. - Режим доступа: https://ru.wikipedia.org/wiki/Домашняя_автоматизация– Дата доступа: 02.04.2017.

ИСПОЛЬЗОВАНИЕ ACTORMODEL ДЛЯ РЕАЛИЗАЦИИ РАСПРЕДЕЛЕННЫХ СИСТЕМ НА ОСНОВЕ ПРИНЦИПОВ DOMAIN-DRIVENDESIGN

Белорусский государственный университет информатики и радиоэлектроники
г. Минск, Республика Беларусь

Костевич А.А.

Блинов И.Н. – канд. физ.-мат. наук, доцент

Несмотря на большое количество доступных архитектурных решений и инструментов для их реализации, разработка распределенных систем, отвечающих высоким требованиям к масштабированию и времени задержки, остается чрезвычайно трудной задачей. Использование архитектуры, построенной на модели параллельных вычислений Actor Model, позволяет не только решить проблемы маштабируемости и производительности, но и предоставляет естественную модель для реализации принципов Domain-Driven Design.

Проектирование масштабируемых распределенных систем на основе традиционной многоуровневой архитектуры является трудновыполнимый задачей, так как их масштабируемость ограничена уровнем доступа к данным (PersistenceLayer). Применение архитектурных принципов Domain-DrivenDesign ставит модель предметной области на первое место в архитектуре системы, накладывая дополнительные требования к уровню хранения данной модели, использующих, как правило, SQL или NoSQL базы данных.

1) Проектирования уровня хранения на основе SQL не позволяет представить агрегаты в виде реляционной модели из за проблемы, известной как Object-RelationalImpedanceMismatch.
2) Проектирование уровня хранения на основе NoSQL позволяет добиться лучшей маштабируемости и решает проблему хранения агрегатов, но возлагает дополнительные обязанности на

разработчика за обеспечения согласованности данных между агрегатами.

Применение архитектурных паттернов CommandQueryResponsibilitySegregation и EventSourcing позволяет частично упростить данные требования к технологиям хранения, однако представляет собой трудную задачу проектирования, сложность которой становится преградой для многих распределенных систем.

Решением проблем маштабируемости и высокой сложности проектирования по модели может стать использования платформы, основанной на идеях ActorModel, изначально предоставляющей распределенную среду для работы модели предметной области.

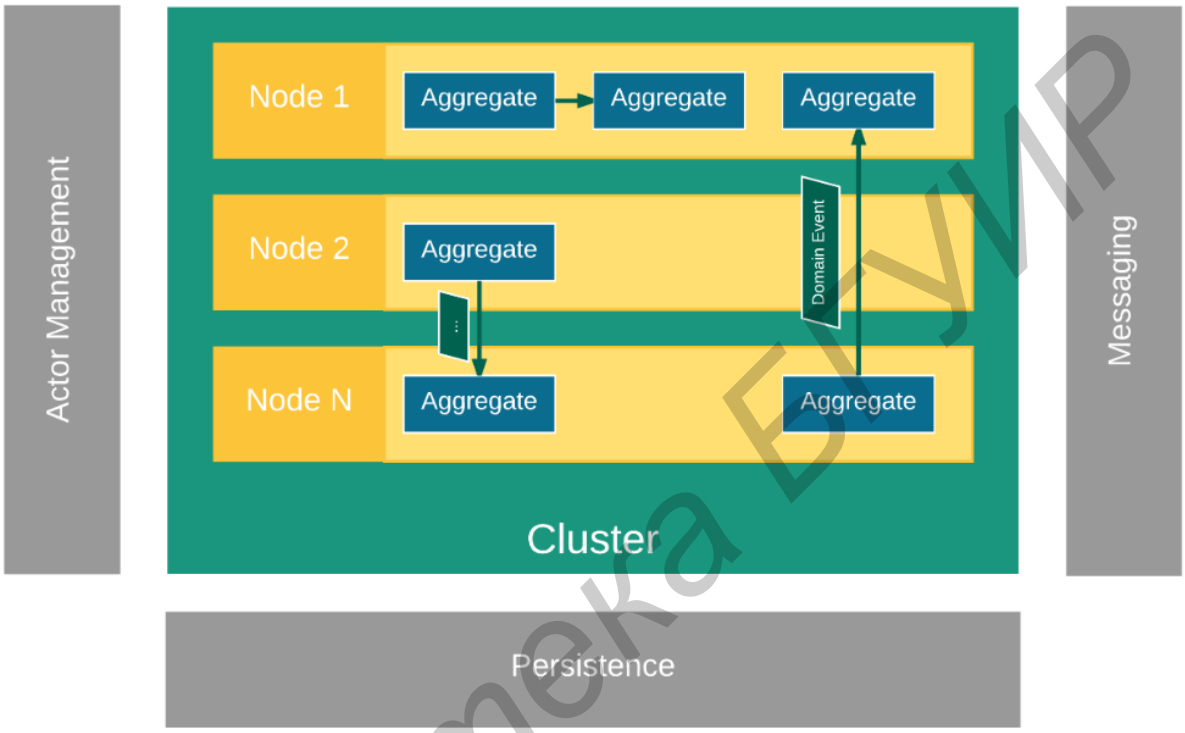

Рис. 1 – Actor Model вконтексте Domain-Driven Design

Ключевыми архитектурными принципами платформы являются следующие:

– каждый агрегат системы представляет собой актора (Actor);
– агрегаты могут находится в памяти системы, представляющей распределенный кластер;
– агрегаты (акторы) общаются между собой на основе сообщений, представляющих собой события данной предметной области (DomainEvents);
– общение между агрегатами происходит асинхронно и управляется на уровне платформы;
– хранение агрегатов реализуется на уровне платформы с обеспечением событийной согласованности (EventualConsistency) вместо транзакционной согласованности (TransactionalConsistency).

Использование исследуемыхMicrosoftOrleans или Akka.NET, предоставляющих реализацию ActorModel для .NET, в качестве основы для последующего процесса проектирования по модели позволяет добиться высокой маштабируемости системы наряду с естественной интеграцией архитектуры с принципами Domain-DrivenDesign.

Список использованных источников:

1. Bernstein P. A., Bykov S., Geller A.,Kliot G., Thelin J. Orleans: Distribute d Virtual Actors for Programmability and Scalability / P. A. Bernstein, S. Bykov, A. Geller, G. Kliot, J. Thelin // Microsoft Research - 2014 - 13 p.
2. Microsoft Orleans Project [Electronic resource] – Mode of access: https://dotnet.github.io/orleans. – Date of access: 12.02.2017.
3. Вернон В., Реализация методов предметно-ориентированного проектирования / В. Вернон – Вильямс, 2015. – 688 с.