SmartTV или ConnectedTV - это технология интеграции интернета и цифровых, интерактивных сервисов, в современные телевизоры и ресиверы цифрового телевидения, а также, в техническом симбиозе между компьютерами и телевизорами / ресиверами цифрового телевидения [1].

После анализа платформ и операционных систем современных телевизоров с функцией SmartTV было выявлено, что каждая система предоставляет свой набор характерных настроек для взаимодействия с ней. Это можно увидеть наглядно при реализации воспроизведения видео контента, управления приложением посредством мыши и клавиатуры, взаимодействие приложения с Интернет.

На глубокое изучение и понимание вариантов реализации этих функций в приложении требуется большое количество времени. Это увеличивает процесс разработки и вероятность допущения ошибки в написании приложения под различные операционные системы и платформы в целом.

Каждая операционная система SmartTV, такая, к примеру, как WebOS, Orsey, Tizen, Netcast, требует детального разбора и понимания своих характерных методов для создания видео-плеера и работы с видео-материалом, такой как воспроизведение, пауза, режим промотки в прямом и обратным направлении, на определенную позицию, регулирование громкости, подсчет времени длительности просматриваемого видео-материала и текущего времени, задание качества видео потока и многое другое.

Схожая проблема возникает и при реализации взаимодействия SmartTV телевизора и разрабатываего приложения с использованием клавиш на пульте телевизора. Каждая операционная система SmartTV предоставляет свой собственный набор кодов клавиш, которые приложение должно уметь правильно обрабатывать: определять, к какой операционной системе они относятся и реагировать должным образом. Это, как правило, создает массу неудобств, поскольку большинство кодов клавиш разные и это ведет к путаницы в реализации различных интерфейсов. Примеры разнообразия кодов клавиш можно наглядно увидеть на ресурсах [2] и [3].

Часто при реализации веб-приложений для SmartTV приходится делать проверки на наличие интернет-соединения, определение готовности телевизора к взаимодействию с приложением или готовности осуществить выход из приложения. Поскольку многие телевизоры по-разному это осуществляют, разработчикам приходится писать немало программного кода, учитывающего все особенности каждой поддерживаемой теле-платформы для успешной реализации этих функций.

Решения этих многих проблем получилось достичь путем реализации унифицированного интерфейса, который инкапсулирует логику взаимодействия приложения с телевизором, распознаёт текущую платформу SmartTV, и предоставляет разработчикам открытый интерфейс, позволяющий реализовать функции для работы со SmartTV телевизорами без углубления в работу самого интерфейса.

Унифицированный интерфейс реализует сервис, который позволяет создать видеоплеер, управлять приложением при помощи контроллеров мыши и клавиатуры, осуществлять различные проверки телевизора на работоспособность, доступ к сети и многое другое для каждой платформы SmartTV. Затем при его использовании, сервис определяет внутри себя текущую платформу и операционную систему, на которой выполняется приложение, и на основании этих определений производит, характерные для данного телевизора инструкции.

Разработанный унифицированный SmartTV интерфейс позволяет абстрагироваться от знаний о конкретных интерфейсах различных платформ и операционных систем телевизоров с технлогией SmartTV и позволяет полностью сконцентрироваться на решении бизнес-проблем, что является важной его особенностью.

Список использованных источников:

1. Википедия [Электронный ресурс]. - Электронные данные. - Режим доступа: https://ru.wikipedia.org/wiki/Smart_TV.
2. HandlingControlKeyEvents | SamsungDevelopers [Электронный ресурс]. - Электронные данные. -Режим доступа: http://developer.samsung.com/tv/develop/legacy-platform-library/art00046/index.
3. LG | webOSTVDeveloper | RemoteControl [Электронный ресурс]. - Электронные данные. - Режим доступа: http://webostv.developer.lge.com/design/webos-tv-system-ui/remote-control.

БАЛАНСИРОВАНИЕ НАГРУЗКИ В WEB ПРИЛОЖЕНИЯХ

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

Малич К.В.

Куликов С. С. – к.т.н., доцент

В настоящее время современные приложения работают с постоянно растущими объемами входящих данных. Для обеспечения поддержки нормального функционирования приложения существуют различные методы. К таким методам можно отнести балансирование нагрузки.

Балансирование нагрузки представляет собой распределение входящих запросов между несколькими сетевыми устройствами. Это позволяет поддерживать отказоустойчивость, предоставляет возможности для горизонтального масштабирования.

В Web приложениях в качестве балансировщиков нагрузки используют несколько Web серверов, все используемые сервера называются фермой серверов. Пример топологии с несколькими серверами

представлен на рисунке 1:

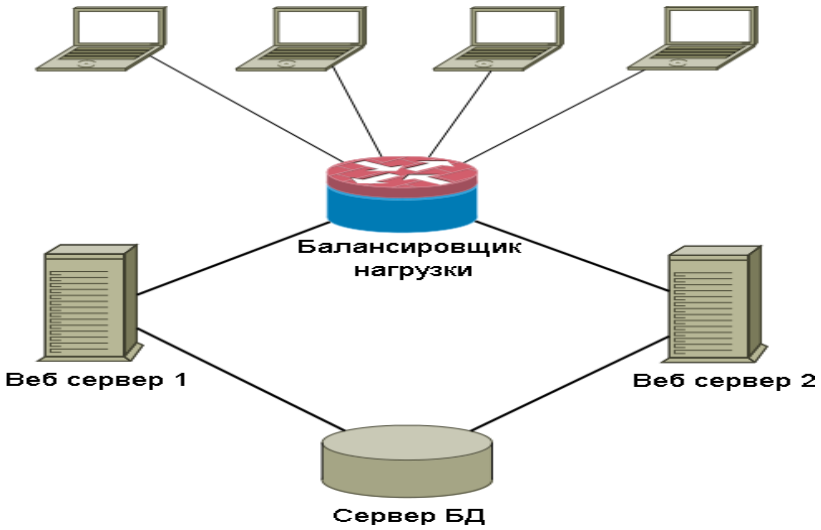

Рис. 1 – использование балансировщика нагрузки.

При такой топологии вводится понятие Прокси, который позволяет распределять нагрузку между серверами. Прокси – сервис, который принимает все входящие запросы и распределяет их между серверами в ферме в соответствии с их загруженностью. Как правило, прокси использует очереди для хранения входящих запросов.

Также использование балансировщиков нагрузки позволяет справиться с проблемой отказа одного из серверов. В таком случае вся нагрузки ложится на остальные сервера, но приложение продолжает свою работу, когда как в случае с единственным сервером, приложение бы прекратило свое корректное функционирование.

Также к положительным сторонам использования балансировщиков агрузки можно отнести простоту горизонтального масштабирования, это значит, что для сокращения нагрузки на существующие сервера достаточно добавить новый сервер, который также будет обрабатывать входящие запросы.

Данный подход является актуальным в наши дни, и большинство крупных систем используют балансирование нагрузки для поддержания работоспособности.

ЭМПИРИЧЕСКАЯ МОДОВАЯ ДЕКОМПОЗИЦИЯ С ИСПОЛЬЗОВАНИЕМ ГРАФИЧЕСКИХ ПРОЦЕССОРОВ NVIDIA, ПОДДЕРЖИВАЮЩИХ ПРОГРАММНО-АППАРАТНУЮ АРХИТЕКТУРУ CUDA

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

Мелещеня Д.В.

Бранцевич П.Ю. – к.т.н., доцент

Эмпирическая модовая декомпозиция является ключевой частью преобразования Гильберта-Хуанга. В силу того, что этапы преобразования включают построение огибающих с помощью интерполяционных полиномов, вся процедура представляет достаточно сложную вычислительную задачу. Для сокращения времени декомпозиции предлагается использовать графические процессоры, реализующих CUDA-совместимую архитектуру, а также, предлагаются подходы по работе с памятью, методы глобальной синхронизации и полной редукции на графическом процессоре, позволяющие ускорить декомпозицию сигнала на модовые функции.

Эмпирическая модовая декомпозиция (EMD – Empirical Mode Decomposition) – итеративная процедура, ставящая в соответствие исходному сигналу набор эмпирических мод (IMF – Intrinsic Mode Functions). Модовая декомпозиция, или просеивание, сводится к последовательности следующих этапов. По локальным экстремумам строятся верхняя и нижняя огибающие. После этого, вычисляется разность между средним значением огибающих и исходным сигналом. Далее, если остаток удовлетворяет критерию остановки, он считается очередной модовой функцией, в противном случае разность принимают за исходный сигнал и алгоритм повторяется. После нахождения IMF, ее вычитают из исходного сигнала и, если разность не является монотонной функцией либо меньше некоторого порогового значения, то алгоритм просеивания