УДК 004.89

Н.В. Пацей, доц., канд. техн. наук;
Д.В. Шиман, доц., канд. техн. наук;
И.Г. Сухорукова, ст. преп.;
А.С. Наркевич, ст. преп.
(БГТУ, г. Минск)

МНОГОУРОВНЕВАЯ АРХИТЕКТУРА ОРГАНИЗАЦИИ РАСПРЕДЕЛЕННЫХ ИНФОРМАЦИОННЫХ СИСТЕМ

Распределенные информационные системы обеспечивают масштабируемость, высокую доступность, отказоустойчивость, репликацию и избыточность, которые обычно недоступны в централизованных информационных системах. Параллельное распределение задач способствует их быстрому выполнению. Обеспечение безопасности, аутентификации и рабочих процессов авторизации выполняется просто, система слабо связана [1–2].

В архитектуре распределенной системы можно выделить разные уровни, которые разбивают обработку на несколько разных частей.

Уровень сбора и подготовки данных *(Data collection and preparation layer)* принимает на вход данные из разных внешних источников. Часто потоки данных не имеют стандартной структуры или формата, поэтому это сырой, неструктурированный или полуструктурированный набор [3–5].

Перемещение данных уязвимо с точки зрения нарушений безопасности. Уровень безопасности данных *(Data security layer)* должен гарантировать целостность и конфиденциальность при передаче данных, выполняя наблюдения и применяя протоколы безопасности, шифрования и прочее механизмы [1–3].

Уровень хранения данных *(Data storage layer)* использует разные подходы. Если аналитика выполняется в реальном времени (потоковая обработка), то при хранении используется размещенный в памяти кеш. Если данные обрабатываются традиционным способом, например, при пакетной обработке, для хранения данных используются распределенные базы данных.

Распределенный кеш, является предпочтительным выбором для облачных вычислений. *Google Cloud* использует *Memcache* для кеширования данных на своей публичной облачной платформе. *Redis* использует хранилища данных *NoSQL* и другие варианты.

Распределенный кеш представляет распределенную хеш-таблицу, которая отвечает за сопоставление значений объекта с ключами, распределенными по нескольким узлам. Распределенная таблица хеширования позволяет масштабировать кеш на лету, она управля-

ет добавлением, удалением и отказом узлов постоянно, пока служба находится в оперативном режиме.

Что касается параллелизма, с ним связано несколько концепций, таких как конечная согласованность, строгая согласованность, распределенные блокировки, журналы фиксации и прочее. Распределенный кеш часто работает с координаторами распределенных систем, такими как *Zookeeper*. Это облегчает обмен данными и помогает поддерживать согласованное состояние между несколькими работающими узлами кеша.

Существуют разные виды стратегий кеширования, которые подходят для конкретных случаев использования. Это *Cache Aside, Read-through cache, Write-through cache* и *Write-back* [6]. Популярные распределенные системы кеширования – *Eh-cache, Memcache, Redis, Riak, Hazelcast*.

Уровень обработки данных *(Data processing layer)* содержит реальную логику и отвечает за обработку и извлечение значимой информации. Для этого используются технологии машинного обучения, прогнозирования, описания и моделирование.

Уровень визуализации данных *(Data visualization layer)* обычно содержит информационные панели на основе браузера, которые отображают информацию в виде графиков, диаграмм, инфографики. *Kibana* – один из примеров инструмента визуализации данных, популярного в отрасли [7].

В настоящее время популярными технологиями распределенной обработки являются – *MapReduce (Apache Hadoop), Apache Spark, Apache Storm, Apache Kafka. Kafka* используется для платформ уведомлений, управления потоками больших объемов данных, мониторинга активности и показателей веб-ресурсов, обмена сообщениями и агрегирования журналов. *Hadoop* предпочтительнее для пакетной обработки данных, тогда как *Spark*, *Kafka* и *Storm* больше подходит для обработки потоковых данных в реальном времени.

Также, существуют две основные архитектуры, используемые в распределенной обработке данных, это – *Lambda* и *Kappa*.

Lambda – распределенная архитектура обработки данных, которая использует как пакетный, так и потоковый подход к обработке данных в реальном времени. Он объединяет результаты перед тем, как представить их пользователю (рис.1).

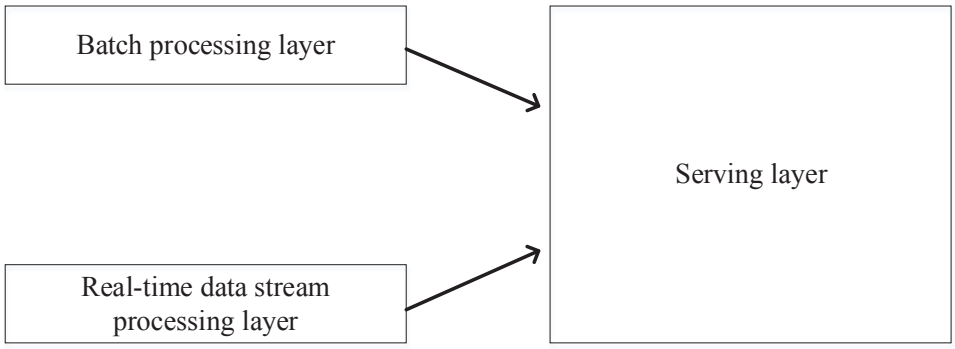

Рисунок 1 – Lambda архитектура системы обработки данных

Пакетная обработка требует времени, учитывая огромное количество данных, но точность подхода высока.

Обработка потоковых данных в реальном времени обеспечивает быстрый доступ к аналитической информации. При этом сценарии аналитика выполняется для небольшой части данных, поэтому результаты не такие точные и исчерпывающие по сравнению с результатами пакетного подхода.

В *Kappa* архитектуре все данные проходят через один конвейер потоковой передачи данных, что снижает сложность управления отдельными уровнями. *Kappa* содержит только два уровня: *Speed*, который является слоем потоковой обработки, и *Serving* – слой обслуживания [8]. *Kappa* не является альтернативой *Lambda*. Обе архитектуры имеют свои варианты использования. Обе архитектуры могут быть реализованы с использованием технологий распределенной обработки данных.

Проектирование и разработка распределенных систем является трудной задачей по следующим причинам: необходимо учитывать множество сочетаний ошибок и сбоев; проблемы возникают на всех рассмотренных уровнях распределенной системы и могут распространяться, в частности проблемы усугубляются на верхних уровнях системы из-за рекурсий; многие проблемы обусловлены законами физики сети.

ЛИТЕРАТУРА

1. Distributed system URL: https://www.scaleyourapp.com/category/distributed-systems/ (дата обращения 12.02.2022).

2. Архитектура современных распределённых систем [Электронный ресурс] : электрон. учеб.- метод. комплекс по дисциплине в LMS Moodle / Мин-во образования и науки РФ, Самар. гос. аэрокосм. ун-т им. С. П. Королева (нац. исслед. ун-т); авт.-сост. С. В. Востокин. - Электрон. текстовые и граф. дан. - (дата обращения 12.09.2021).

3. Таненбаум Э. Распределенные системы. Принципы и парадигмы / М. ван Стеен. – СПб. [и др.] : Питер , 2003. – 877 с.

4. Востокин С.В. Графическая объектная модель параллельных процессов и ее применение в задачах численного моделирования / С.В. Востокин. Изд-во Самарского научного центра РАН – Самара, 2007. – 186 с.

5. Мосин С.В., Зыкин С.В. Кэширование запросов к реляционной базе данных с использованием областей истинности. Моделирование и анализ информационных систем, 22(2), 2015 – С. 248-258.

6. Read-Through, Write-Through, Write-Behind, and Refresh-Ahead Caching URL: https://docs.oracle.com/cd/E15357_01/coh.360/e15723/ cache_rtwtwbra.htm (дата обращения 12.01.2022).

7. Kibana: Explore, Visualize, Discover Data URL: https://www.elastic.co/kibana// (дата обращения 12.01.2022).

8. Kappa архитектура URL: https://www.bigdataschool.ru/blog/kappa-architecture.html (дата обращения 03.01.2022).

UDC 316.776

N.V. Patsei, PhD.; G. Jaber, PhD stud. (BSTU, Minsk)

ROUTING OF NAMED DATA OBJECT WITH SEMANTIC PART OF NAME IN INFORMATION-CENTRIC NETWORKING

According to previous research it was proposed the three-dimension address for NDO (Name data objects) [1–3]. As follows, the three tables design. The first table matches semantic addresses to publisher ID address. The second table matches publisher ID address to geographical address. The third table matches semantic address to geographical address.

Let's walk through the process of adding, removing and matching an entry to the routing table on an NDO request.

If a subscriber sends Interest Request Message (IRM) and it reaches the router, the router will search its tables for an entry match. There are three cases:

Case 1: If there is no positive match between IRM and any addressing table in the router, the router will broadcast IRM to the network from all its interfaces in a spanning tree technique to avoid loops. When IRM reaches the match in any router caching the content or even in the original publisher cache, the latest will send a reply message to all requested interfaces, which sent IRM to it containing the data with the three publisher addressing dimensions. The reply message CRM (Content Reply Message)