РЕАЛИЗАЦИЯ АЛГОРИТМА «ВЗВЕШЕННЫХ НАИМЕНЬШИХ СОЕДИНЕНИЙ» КОНТРОЛЯ НАГРУЗКИ НА СЕРВЕРЫ

Д.А. Хлебест, Е.С Омелюсик
Белорусский государственный университет информатики и радиоэлектроники
г. Минск, Республика Беларусь
Шаталова В.В. – канд.техн.наук, доцент

Алгоритм взвешенных наименьших соединений поддерживает взвешенный список серверов приложений с их числом активных соединений. Служба переадресует новое соединение с сервером на основе следующей комбинации: его пропорция к весу или предпочтению, количество активных подключений.

Этот алгоритм использует больше времени вычислений, чем алгоритм наименьшего соединения, однако дополнительные вычисления приводят к более эффективному распределению трафика на сервер [1].

В алгоритме взвешенного наименьшего планирования соединений (Weighted Least Connections, WLC) каждому серверу может быть присвоен различный вес производительности. Алгоритм планирования взвешенного наименьшего количества соединений вводит «вес», основанный на спецификациях каждого сервера [3].

Балансировщик нагрузки, который реализует алгоритм WLC, теперь учитывает две вещи: вес (емкость каждого сервера) и текущее количество клиентов, в настоящее время подключенных к каждому серверу.

Алгоритм может быть применён в «диспетчере» – распределителе нагрузки на серверы и единой точке входа в распределённую систему, для последующего распределения клиентов на различные API-шлюзы (application programming interface, программный интерфейс приложения).

Формула расчёта веса для «API-шлюза»:

$$W = \sqrt{(\rho_{mem} \cdot Q_{mem} \cdot R_{mem})^2 \cdot (\rho_{cpu} \cdot Q_{cpu} \cdot R_{cpu})^2},$$

где R_{mem} – простой памяти;
R_{cpu} – простой процессора;
Q_{mem} – объём памяти (Кб);
V_{cpu} – скорость процессора (МГц);
ρ_{mem} и ρ_{cpu} – коэффициенты пропорциональности.

Формула расчёта приоритета для «API-шлюза»:

$$P_i = \frac{C_i}{W_i},$$

где n – количество узлов;
W_i – веса узлов (i = 1..n);
C_i – количество текущих подключений (i = 1..n).

Схематичная реализация алгоритма в нотации языка Node.js. Сокращённые обозначения объектов и параметров:

[O] – оптимальный (следующий) «API-шлюз»;
[C] – количество текущих подключений «API-шлюза»;
[W] – вес «API-шлюза»;
[P] – приоритет «API-шлюза»;
[I] – любая дополнительная текстовая информация о «API-шлюзе».

Взаимодействие с сервисом «Диспетчера» сводится к 4-м операциям CRUD (создание, чтение, обновление, удаление).

Запрос на создание инициирует новый «API-шлюз», готовый принимать на себя последующие подключения клиентов. В запросе он должен передать начальное количество подключений и параметры для расчёта веса. В ответ он получит личный уникальный идентификатор во всей системе.

Формат запроса: { P: { C, W: { R_{mem}, R_{cpu}, Q_{mem}, V_{cpu} } }, I: {...} }

Формат ответа → { id }

Запрос на обновление инициирует уже работающий «API-шлюз» по событию изменения числа обрабатывающих подключений (подключение или отключение клиентов). В запросе он должен передать новое число открытых соединений. В ответ он получит статус выполнения запроса.

Формат запроса: { C }

Формат ответа → { … }

Запрос на удаление инициирует «API-шлюз», завершающий работу по каким-либо причинам. В запросе и ответе параметры не требуются.

Формат запроса: { … }

Формат ответа → { … }

Запрос на чтение инициирует новый клиент, желающий получить начать работу с распределённой системой. Для выполнения запроса параметры не требуются. В ответ он получит оптимальный «API-шлюз» с минимальной нагрузкой.

Формат запроса: { … }

Формат ответа → { O }

Для оптимального выполнения вышеописанных запросов, сервис «Диспетчера» в памяти должны кэшироваться «активный пул» соединений с форматом: **activePool**: { “id_1”: { P_1: { C_1, W_1 }, I_1: { … } }, … }, а также переменная-объект, содержащая оптимальный на текущий момент «API-шлюз»: **optimalChoice**: { id, P, I }.

Алгоритм взвешенных наименьших соединений один из самых распространённых выборов при разработке больших проектов с распределёнными системами, реализация которого варьируется от особенностей системы, а разработка требует наиболее оптимального решения.

Список использованных источников:

1. Алгоритмы принятия решения по балансировке нагрузки [Электронный ресурс]. – Режим доступа: https://www.ibm.com/support/knowledgecenter/SS9H2Y_7.6.0/com.ibm.dp.doc/lbg_algorithms.html. – Дата доступа: 25.11.2019.

2. S.Sharma, S.Singh, M.Sharma. Анализ производительности алгоритмов балансировки нагрузки: всемирная академия наук, инженерии и технологии, 2008.