МОДЕЛИРОВАНИЕ ВЕБ-ПРОЦЕССОВ СЕТЯМИ ПЕТРИ С РАСПРЕДЕЛЕНИЕМ ЗАДАЧ МЕЖДУ АГЕНТАМИ ЧЕРЕЗ WEBSOCKET

Хаджинова Н. В.
Кафедра информационных технологий автоматизированных систем,
Белорусский государственный университет информатики и радиоэлектороники
Минск, Республика Беларусь
E-mail: khajynova@bsuir.by

Современные веб-приложения характеризуются сложной архитектурой, множеством параллельных взаимодействий и необходимостью эффективного распределения ресурсов. Предложенный подход с использованием сетей Петри позволяет моделировать взаимодействие агентов через постоянные WebSocket-соединения, анализировать производительность, выявлять узкие места и предотвращать конфликты при распределении задач. Моделирование также помогает улучшить отказоустойчивость системы и оптимизировать распределение запросов.

Введение

Протокол связи WebSocket обеспечивает эффективное, двустороннее и долговременное соединение, что критично для систем, где требуется мгновенный обмен данными между сервером и клиентом. В современных веб-приложениях WebSocket активно используется, и интеграция его с сетями Петри может значительно оптимизировать и автоматизировать процессы. Сети Петри позволяют описывать поведение параллельных процессов, их синхронизацию и распределение задач, а также управление общими ресурсами, что делает их эффективным средством для анализа и оптимизации веб-процессов.

I. Распределенные процессы в веб-среде

Распределённые процессы в веб-среде – это процессы, которые выполняются на нескольких узлах, взаимодействующих через сеть для достижения общей цели [1, 2]. В контексте моделирования веб-процессов с распределением задач через WebSocket, агентами выступают автономные программные сущности, которые взаимодействуют друг с другом и сервером для выполнения задач. Агентами могут быть: пользовательские сессии, сервисные или служебные агенты, боты, микросервисы, брокеры или каналы данных, роли, закреплённые за пользователями. WebSocket обеспечивает двунаправленное постоянное соединение, что снижает задержки и упрощает управление задачами.

Использование сетей Петри позволяет моделировать взаимодействие агентов, анализировать распределение задач и управлять ресурсами с высокой степенью надёжности [3].

II. Оптимизация процесса обработки запросов

Постановка задачи: необходимо оптимизировать процесс обработки клиентских запросов сервером с доступом к базе данных (см. рис. 1). Диаграмма представляет взаимодействие между клиентами (браузерами), сервером и базой данных [4].

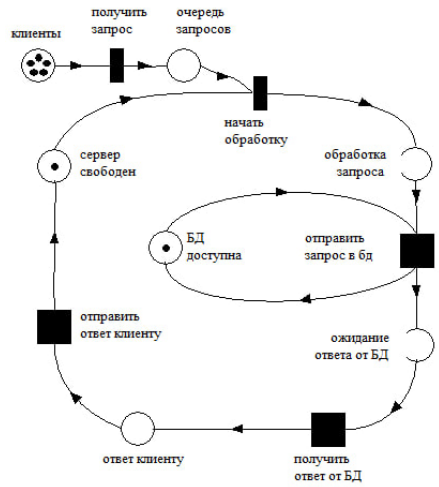

Рис. 1 – Диаграмма процесса до оптимизации

Каждый браузер, подключенный к серверу, может представлять отдельную сессию. Процесс начинается с отправки браузером запросов на сервер через WebSocket, которые попадают в очередь на обработку. Сервер, используя WebSocket, может отслеживать, какой пользователь подключён к какому браузеру, управлять сессиями и обменом данными между ними. Сервер обрабатывает запросы последовательно: получает запрос, отправляет его в базу данных, ожидает ответа и только потом возвращает результат клиенту. Сервер асинхронно обрабатывает запрос и уведомляет о прогрессе. После обработки результат отправляется обратно через WebSocket, что исключает необходимость повторных запросов. Для множества агентов WebSocket обеспечивает постоянное соединение, минимизируя задержки. Для моделирования выполнения процесса обработки сервером запросов клиентов написана функция, которая последовательно обрабатывает запросы (см. рис. 2, 3). Для событий устанавливается приблизительное время, и высчитывается время вы-

полнения запроса. Например, время обработки сервером тысячи запросов равно 8.1 с.

```
function processRequest(clientNumber, callback) {
    const receiveTime = 0.1;
    const processTime = 0.01;
    const sendDbTime = 0.1;
    const dbResponseTime = 0.4;
    const sendClientTime = 0.2;
    let totalTime = 0;
    totalTime += receiveTime;
    totalTime += processTime;
    totalTime += sendDbTime;
    totalTime += dbResponseTime;
    totalTime += sendClientTime;
    setTimeout(() => {
        callback(totalTime);
    }, totalTime);
}
```

Рис. 2 – Функция имитации процесса выполнения запроса

```
function simulateSync() {
    let totalExecTime = 0;
    let completedRequests = 0;
    const totalRequests = 1000;
    let index = 0;
    function processNextRequest(index) {
        if (index < totalRequests) {
            processRequest(index, (time) => {
                totalExecTime += time;
                completedRequests++;
                processNextRequest(index + 1);
            });
        } else {
            console.log(`Итоговое время: ${totalExecTime}`);
        }
    }
    processNextRequest(index);
}
```

Рис. 3 – Функция имитации синхронной обработки запросов сервером

Основная проблема этого подхода – длительное время ожидания. Чтобы сократить время, сервер может отправлять запросы в базу данных без ожидания завершения текущего, сразу готовясь к следующему запросу [5, 6]. Это уменьшает общее время выполнения, так как ожидание от базы данных совмещается с другими операциями (см. рис 4).

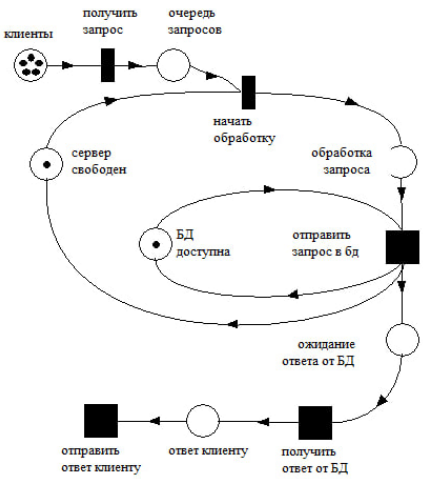

Рис. 4 – Диаграмма процесса после оптимизации

Для моделирования асинхронного процесса обработки запросов сервером написана функция, которая обрабатывает запросы не ожидая от них ответа (см. рис. 5). Время обработки сервером тысячи запросов в данном случае сокращается до 0,81 с, что доказывает эффективность оптимизации.

```
async function simulateAsync() {
    let totalExecTime = 0;
    const totalRequests = 1000;
    const promises = [];
    for (let i = 0; i < totalRequests; i++) {
        promises.push(simulateAsyncRequest(
            new Promise((resolve) => {
            processRequest(i, resolve);
        })));
    }
    const results = await Promise.all(promises);
    totalExecTime = results.reduce((acc, time) => acc + time, 0);
    console.log(`Итоговое время: ${totalExecTime / totalRequests}`);
}
```

Рис. 5 – Функция имитации асинхронной обработки запросов сервером

III. Заключение

Оптимизация распределения задач через WebSocket значительно повышает производительность веб-приложений, устраняя задержки при установке новых соединений и сокращая время выполнения процессов. Постоянное соединение позволяет серверу обрабатывать несколько задач параллельно, эффективно используя ресурсы и снижая время простоя при ожидании откликов от базы данных.

Сети Петри, применяемые для моделирования веб-процессов, позволяют анализировать параллельные процессы, выявлять узкие места и сбалансировать нагрузку на сервер. Благодаря формальному математическому аппарату сетей Петри можно точно оценить задержки, определить источники конфликтов при доступе к ресурсам и предложить способы их устранения. Этот метод также помогает моделировать отказоустойчивость системы, прогнозировать её поведение при сбоях и минимизировать влияние этих сбоев на производительность.

IV. Список литературы

1. Котов В. Е. Сети Петри: Наука, 1984.
2. Питерсон Дж. Теория сетей Петри и моделирование систем. – М.: Мир, 1984. – 264 с.
3. Ломазова И. А. Моделирование мультиагентных динамических системвложенными сетями Петри // Программные системы: Теоретические осно-вы и приложения: Наука. Физматлит, 1999, с. 143–156.
4. Шелестов А. Ю. Моделирование GRID-узла на основе сетей Петри: Наука, 2009.
5. Бусленко Н. П. Моделирование сложных систем. – М.: Наука, 1978. – 400 с.
6. Бенькович Е. С., Колесов Ю. Б., Сениченков Ю. Б. Практическое моделирование динамических систем. – СПб.: БХВ-Петербург, 2002. – 464 с.