ПРОГРАММНЫЙ МЕНЕДЖЕР РАСПРЕДЕЛЕННЫХ ВЫЧИСЛЕНИЙ В МУЛЬТИАГЕНТНОЙ СРЕДЕ

Северин К. М., Парамонов А. И.
кафедра программного обеспечения информационных технологий, кафедра информационных систем и технологий Института информационных технологий, Белорусский государственный университет информатики и радиоэлектроники
Минск, Республика Беларусь
E-mail: severin.klim@yandex.by, a.paramonov@bsuir.by

Рассмотрена проблема балансировки нагрузки или выравнивания нагрузки в вычислительной сети. Выполнен обзор существующих методов балансировки задач. Описана реализация менеджера распределенных вычислений на основе алгоритма Round Robin. Выполнен проект мультиагентой среды для организации распределенных вычислений без простоев.

Введение

В связи с большей доступностью и широким применением вычислительных систем становится актуальной проблема их эффективного использования. Одним из аспектов данной проблемы является эффективное распределение задач внутри распределенных серверов внутри вычислительных систем с целью сокращения общего времени вычисления. В этой связи при организации компьютерных сетей и систем возникают потребности в балансировке нагрузки или выравнивании нагрузки. Суть метода состоит в распределении заданий между несколькими сетевыми устройствами с целью оптимизации использования ресурсов вычислительного кластера, сокращения времени обслуживания запросов, горизонтального масштабирования кластера, а также обеспечения отказоустойчивости.

В работе рассматривается проблема балансировки нагрузки на прикладном уровне и предлагается прикладное решение задачи распределения вычислений в мультиагентной среде. Для этого реализован программный менеджер, который распределяет поток запросов на вычисления по сети агентов. Менеджер должен обеспечивать следующие возможности:

- поддерживать подключение с несколькими агентами;
- распределять полученные задачи согласно выбранному алгоритму;
- корректно обрабатывать ситуации при отсутствии связи с агентом или клиентом (не заканчивать критически свою работу при неработающем агенте или при неработающем клиенте);
- гарантированное выполнение принятой задачи от клиента.

I. Существующие решения задачи балансировки

На сегодня существует множество алгоритмов планирования, которые используются балансировщиками нагрузки для определения «какому исполнителю послать запрос», и каждый из них имеет свои особенности. В числе широко используемых: Round Robin, Least Connections, Destination Hash Scheduling, Sticky Sessions и их модификации [1].

Round Robin, или алгоритм кругового обслуживания, представляет собой перебор по круговому циклу: первый запрос передаётся одному серверу, затем следующий запрос передаётся другому и так до достижения последнего сервера, а затем всё начинается сначала. Weighted Round Robin – усовершенствованная версия алгоритма Round Robin. Суть усовершенствований заключается в следующем: каждому серверу присваивается весовой коэффициент в соответствии с его производительностью и мощностью. Это помогает распределять нагрузку более гибко: серверы с большим весом обрабатывают больше запросов.

При распределении запросов по алгоритму Least Connections, каждый следующий запрос передаётся серверу с наименьшим количеством активных подключений. Weighted Least Connections – усовершенствованная версия Least Connections. Этот алгоритм учитывает при распределении нагрузки не только количество активных подключений, но и весовой коэффициент серверов. Locality-Based Least Connection Scheduling был создан специально для кэширующих прокси-серверов. В алгоритме Locality-Based Least Connection Scheduling with Replication Scheduling каждый IP-адрес или группа IP-адресов закрепляется не за отдельным сервером, а за целой группой серверов (кластерами).

Алгоритм Destination Hash Scheduling был создан для работы с кластером кэширующих прокси-серверов, но он часто используется и в других случаях. В этом алгоритме сервер, обрабатывающий запрос, выбирается из статической таблицы по IP-адресу получателя.

Sticky Sessions – алгоритм распределения входящих запросов, при котором соединения передаются на один и тот же сервер группы. Сессии пользователя могут быть закреплены за кон-

кретным сервером с помощью метода IP hash. С помощью этого метода запросы распределяются по серверам на основе IP-адреса клиента.

II. Программный комплекс распределения нагрузки в мультиагентной среде

В программной реализации менеджера в качестве базового алгоритма балансировки задач используется модифицированный алгоритм Round Robin.

Разработанный программный комплекс реализован на .Net платформе, с использованием встроенных коллекций: SortedSet – для хранения задач в отсортированном виде по времени, Dictionary – для хранения пары (Id задачи, Сокет клиента) и последующей отправки клиенту задачи по окончании ее выполнения. Для работы с JSON использовался nuget из пакета System.Text.Json. Для работы с сетевыми функциями использовался класс System.Net.Sockets.Socket.

Алгоритм работы менеджера мультиагентной вычислительной среды таков:

Первым в сети запускается менеджер, который слушает указанные в аргументах коммандной строки портах входящие соединения от агентов и клиентом. Общение в сети происходит по собственному протоколу поверх TCP. Сам протокол состоит из двух полей: размер JSON данных и сами данные. JSON данные описывают задачу на разных этапах ее жизни и включают в себя максимум 6 полей: id – уникальный идентификатор задачи (устанавливается балансировщиком), time – время получения задачи (устанавливается балансировщиком), command – команда, которую необходимо выполнить (отправляется клиентом), arguments – аргументы к команде (отправляется клиентом), exitcode – код возврата (устанавливается агентом), result – программный вывод (устанавливается агентом). Такой протокол общения позволяет передавать только необходимые данные и не нагружать сеть. В качестве протокола транспортного уровня используется TCP, что гарантирует целостность передаваемых данных и уведомление отправителя о результатах передачи [2].

Получив задачу от клиента, менеджер добавляет ее в глобальную приоритетную очередь нераспределенных задач. Приоритетом в данной очереди является время получения задачи. Далее в отдельном потоке эти задачи распределяются между подключенными агентами согласно алгоритму балансировки и добавляются в очереди активных задач агентов. На стороне агента присутствует также очередь задач на выполнение. Агент выполняет задачи в отдельном процессе и контролирует ее выполнение. Далее агент отправляет результат выполнения обратно менеджеру, который в свою очередь отправляет результат клиенту.

Однако следует отметить, что по причине потери связи с агентом задача может быть не выполнена. Потеря связи может произойти в силу таких причин как: отключение сетевого оборудования, используемого для связи с агентом, аварийное завершение агента из-за отключения питания компьютера и др. Для детектирования наличия соединения с агентом менеджер один раз в 5 секунд отправляет «пустую» задачу, ответ на которую агент должен выслать сразу же. Если ответ не поступил или соединение было прервано менеджер перемещает все активные задачи из очереди «проблемного» агента в глобальную очередь нераспределенных задач.

В ходе тестирования разработанного программного комплекса была подтверждена стабильная работа менеджера и гарантированное выполнение всех поставленных задач. Вместе с тем, в ходе тестирования был выявлен один из недостатков реализованного алгоритма балансировки: при распределении не учитывается различия в сложности обрабатываемых агентами задач, а также различия в мощности самих агентов. Таким образом при определенных условиях наблюдается простой одних агентов и загруженность других агентов. Эта проблема предполагает внедрение дополнительного мониторинга ресурсов в вычислительной сети.

III. Заключение

Предложенное решение может применяться в различных системах распределенного выполнения задач, таких как обработка серверных запросов, компьютерные, создание сложных математических моделей. Одной из актуальных областей применения предложенного подхода является балансировка нагрузки при организации дистанционного образования, в частности проведение большого числа единовременных высоконагруженных видеоконференций [3]. Рассмотренная проблемная область в ходе эксперимента показала, что важным аспектом достижения высокой производительности при распределенных вычислениях можно достичь не только путем масштабирования вычислительного кластера, но и за счет рационального использования возможностей его агентов.

1. RHEL5: Linux Virtual Server (LVS) / Red Hat, Inc. [Электронный ресурс] – режим доступа: https://web.mit.edu/rhel-doc/5/RHEL-5-manual/Virtual_Server_Administration/s2-lvs-sched-VSA.html. - Дата доступа: 14.10.2022.
2. Программная спецификация протокола DARPA INTERNET [Электронный ресурс] – режим доступа: https://rfc.com.ru/rfc793.htm
3. Markov, A. Parameters of load testing models: approaches to estimation / A. Markov, A. Paramonov // Nano-Desing, Tehnology, Computer Simulations (NDTCS–2021) : тезисы докладов XIX Международного симпозиума, Минск, 28–29 октября 2021 года / БГУИР ; редкол.: В. А. Богуш [и др.]. – Минск, 2021. – P. 122—123.