ОБЗОР МЕТОДОВ БАЛАНСИРОВКИ НАГРУЗКИ В КОМПЬЮТЕРНЫХ СЕТЯХ

В.В. Чемерилов Е.С. Чердынцев
Томский политехнический университет
vchemerilov@gmail.com, es@tpu.ru

Введение

При проектировании компьютерной сети важной задачей является правильное распределение нагрузки между узлами данной сети. Балансировка нагрузки – это метод распределения заданий между несколькими сетевыми устройствами с целью оптимизации использования ресурсов, сокращения времени обслуживания запросов, обеспечения отказоустойчивости (резервирования) и горизонтального масштабирования кластера [1]. В этой работе будет проведена классификация алгоритмов балансировки нагрузки компьютерной сети, будут выделены достоинства и недостатки каждого алгоритма.

Статические алгоритмы распределения нагрузки

Алгоритмы распределения нагрузки в сети можно поделить на статические, динамические и адаптивные.

В статических алгоритмах решения о распределении нагрузки принимается заранее, до начала работы. Они не изменяются в процессе работы. Примером статического алгоритма может служить алгоритм Round Robin [2]. Рассмотрим его подробнее.

Пусть имеются N объектов и M задач, которые должны выполнить эти объекты. Пусть задачи m имеют равный приоритет, а объекты n равны по своим свойствам между собой. Первый объект (n=1) должен выполнить первую задачу (m=1), второй объект вторую и т.д., до достижения последнего объекта (m=N). Следующая задача (m=N+1) будет задана первому объекту (n=1) и т.д. То есть происходит перебор объектов, выполняющих задания, по циклу или по кругу (round), и по достижении последнего объекта следующая задача будет также назначена первому объекту.

В компьютерных сетях в качестве объектов, можно принять IP-адреса web-серверов, а в качестве задач – ответы web-сервера на запросы клиентов. Когда клиент посылает запрос на web-сервер, web-сервер отвечает на запросы не только одним IP-адресом, а списком из нескольких адресов серверов, предоставляющих идентичный сервис, при этом порядок, по которому определяется этот список, основан на алгоритме Round Robin. С каждым ответом последовательность IP-адресов меняется. Таким образом, разным клиентам будут выданы IP-адреса разных серверов, что распределит общую нагрузку между серверами [3].

Статический алгоритм достаточно легко реализовать, но в случае невозможности выполнения объектом поставленной ему задачи (например, если сервис на одном IP-адресе будет недоступен), данный объект не удалится из списка всех объектов, а будет получать новые задачи (клиенты могут повторно получить IP-адрес недоступного сервера).

Динамические и адаптивные алгоритмы распределения нагрузки

В отличие от статических алгоритмов распределения динамические алгоритмы используют информацию о системе для реализации распределения нагрузки. Примером такой информации может служить информация о загруженности узлов. Загруженность узла может определяться, например, как количество задач в очереди узла, либо как доля процессорного времени, затраченная на решение задачи, либо как вероятность обнаружить узел, занятый решением задачи в какой-либо момент времени.

Динамические алгоритмы нагрузки можно разделить по степени централизованности на централизованные, иерархические и полностью распределенные алгоритмы [4]. Также существуют алгоритмы, совмещающие все указанные подходы.

Координатор собирает и поддерживает в актуальном состоянии информацию о загруженности узлов системы, а алгоритм поиска
партнера использует эти данные.

Большой недостаток централизованных алгоритмов состоит в том, что в случае сбоя центрального компонента, произойдет сбой всей системы (центральный компонент является уязвимым местом системы). Централизованные алгоритмы потенциально менее надежны, чем иерархические и полностью распределенные.

Динамический алгоритм балансировки содержит четыре элемента:

политику балансировки;
алгоритм выбора партнера;
алгоритм выбора задачи для передачи;
механизм сбора необходимой информации о состоянии системы.

Политика балансировки определяет, является ли узел объектом балансировки. Алгоритм выбора задачи определяет, какую именно задачу необходимо передать. При выборе задачи алгоритм может учитывать, что накладные расходы, которые связаны с пересылкой задачи, должны быть минимальны, число связей в задаче с локальными ресурсами должно быть минимально, сложность выполнения задачи должна быть большой.

За выбор подходящего узла для операции балансировки нагрузки отвечает алгоритм выбора

партнера. Обычно в распределенных алгоритмах используют опросы узлов. Они могут быть последовательными или параллельными.

Механизм сбора информации о загруженности системы определяет, когда собираются данные о загруженности, где хранятся эти данные и что является их источником. Выделяют несколько классов механизмов сбора информации [4].

1. Сбор данных по необходимости. Когда узел нуждается в балансировке нагрузки, распределенные алгоритмы данного класса собирают информацию о загруженности. Они подразделяются на инициируемые отправителем, получателем и симметрично инициируемые. В алгоритмах, инициируемых отправителем, узел, передающий задачи, ищет получателей, которым он может передать часть задач. В алгоритмах, инициируемых получателем, узел, получающий задачи, заимствует задачи у отправителя. В симметричных алгоритмах используется комбинация упомянутых подходов.
2. Периодический сбор данных. Алгоритмы данного класса могут быть как распределенными, так и централизованными. Алгоритм инициирует балансировку нагрузки в зависимости от собранных данных.
3. Сбор данных по изменению состояния. В системах, реализующих алгоритмы данного класса, при изменении внутреннего состояния, узлы сами распространяют информацию об изменении загруженности.

Под стабильностью алгоритма балансировки или системы, его использующей, обычно понимают одну из двух характеристик:

- системная устойчивость, то есть недопустимость ситуации, когда отдельные узлы системы перегружены, в то время как остальные простаивают или недогружены;
- алгоритмическая стабильность, выраженная в том, что алгоритм не совершает бесполезных действий с ненулевой вероятностью.

Динамические алгоритмы можно использовать, например, для решения задач динамической маршрутизации. Пусть маршрут пакета от отправителя до получателя включает маршрутизаторы нескольких уровней – уровня доступа, распределения и ядра. Маршрутизатору необходимо оптимизировать распределения нагрузки между используемыми каналами связи. Для решения задачи используется динамический алгоритм, который будет распределять нагрузку, основываясь на информации о состоянии загруженности соседних узлов [5].

Адаптивные алгоритмы являются частным случаем динамических алгоритмов и в зависимости от условий и априорного знания о свойствах алгоритмов балансировки выбирают наиболее подходящий из них. Пример - алгоритм адаптивной балансировки через оценку эффективной пропускной способности [6]. Этот алгоритм маршрутизации имеет две цели: оценка стоимости канала и создание «политики для принятия решения об обновлении стоимости канала». Для первой цели все стоимости, которые будут использоваться в будущем кратчайшем или лучшем периоде обновления путей вычисляются на основе предварительной информации о нагрузке каналов (трафика). В то же время для второй, расходы анализируются таким образом, чтобы решение по обновлению расходов могло быть принято окончательно.

Заключение

В результате работы была рассмотрена классификация алгоритмов распределения нагрузки в компьютерной сети. У каждого класса есть свои достоинства и недостатки, например, динамические алгоритмы должны собирать, хранить и анализировать информацию о состоянии системы, им свойственны большие накладные расходы на балансировку по сравнению со статическими алгоритмами, но статические алгоритмы менее эффективны по сравнению с динамическими. Выбор алгоритма распределения нагрузки во многом зависит от того, какие сетевые устройства будут использоваться.

Литература

http://ru.wikipedia.org/wiki/ Балансировка_нагрузки,
http://ru.wikipedia.org/wiki/Round-robin(алгоритм),
http://ru.wikipedia.org/wiki/Round _robin_DNS,
http://www.isa.ru/jitcs/images/stories/2009/03/33_48.pdf,
http://ea.donntu.edu.ua:8080/jspui/bitstream/123456789/7138/1/610_Batyr.PDF
http://www.moluch.ru/archive/63/9880/