Информационные ресурсы и технологии

НИКОНОВ Эдуард Германович *- доктор физико-математических наук, профессор кафедры системного анализа и управления Международного университета Природы, Общества и Человека "Дубна"*

ФЛОРКО Андрей Борисович *- аспирант кафедры системного анализа и управления Международного университета Природы, Общества и Человека "Дубна", ведущий WEB-программист ЗАО "Финам" (Москва)*

ПОВЫШЕНИЕ ОТКАЗОУСТОЙЧИВОСТИ МНОГОЭТАПНЫХ СОБЫТИЙНО-УПРАВЛЯЕМЫХ СИСТЕМ АНАЛИЗА И ОБРАБОТКИ ИНФОРМАЦИИ В СЦЕНАРИЯХ СТРЕССОВОЙ НАГРУЗКИ

Самый широкий спектр современных информационных систем предполагает поэтапную обработку поступающих в систему запросов. В ответ на поступивший запрос (будь то принятие клиентского пакета из сети, ввод данных пользователем или появление в системе периодического события, например, таймера) такие системы выполняют определённую последовательность работ по обработке данных сообщения, включающую в общем случае несколько этапов. К примеру, WEB-сервер в ответ на запрос статической HTML-страницы последовательно разбирает адрес страницы, выполняет её поиск в локальном кеше, затем (если необходимо) обращается к жёсткому диску, читает данные, подготавливает ответный пакет и отправляет его клиенту.

Один из существующих архитектурных подходов, применяемых для создания масштабируемых многоэтапных событийно-управляемых информационных систем массового обслуживания, рассматривает систему как сеть компонент, обменивающихся сообщениями (рис. 1).

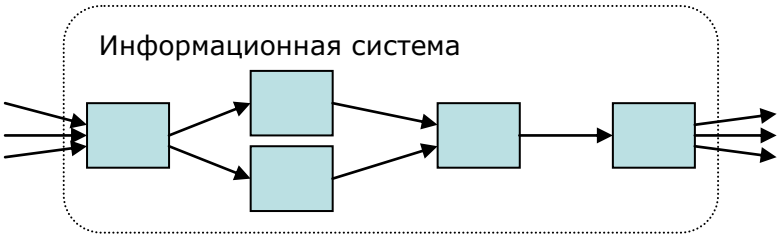


Рис. 1. Системы многоэтапной обработки информации

С каждой компонентой системы ассоциирована очередь входящих сообщений фиксированного размера и пул системных потоков, выделенных для её обработки. В общем случае сообщение проходит маршрут из нескольких компонент (каждая из которых выполняет определённую работу над данными сообщения), прежде чем сообщение удаляется из системы.

Среди информационных систем, использующих рассмотренный архитектурный подход, можно перечислить следующие классы:

- Системы потоковой обработки данных[1]
- WEB сервера[2]
- SCADA-системы
- Математические комплексы, с применением алгоритмов параллельной обработки и некоторые другие.

Поскольку перечисленные классы систем применяются во многих жизненно важных для человеческой жизнедеятельности областях (военная, аэрокосмическая, химическая, транспортная), то особенное внимание при разработке следует уделять отказоустойчивости. Под ***отказоустойчивостью*** понимается способность вычислительной системы продолжать действия, заданные программой, после возникновения неисправностей [8]. Требование отказоустойчивости в последние годы стало особенно

[1] Например, процессоры обработки потоков (stream processing engines) Aurora [5], STREAM [6], TelegraphCQ [7].

[2] Например, Flash[1].

актуальным как в связи с экспоненциальным ростом числа террористических атак по всему миру, так и в связи с ежегодным увеличением числа регистрируемых природных и техногенных катастроф. В то время, как неспособность системы справиться с нагрузкой в случае интернет-сервера может закончиться судебными исками разгневанных покупателей[3], то в случае со SCADA – техногенными авариями и человеческим жертвами.

Среди всевозможных причин, ведущих к отказу системы (неисправность оборудования, исчерпание доступных вычислительных ресурсов), в статье рассматриваются сценарии стрессовой нагрузки, в которых число поступающих в систему сообщений превосходит способности системы к обработке. Сообщения начинают выстраиваться в очередь, и отказ системы возникает при попытке поставить сообщение в очередь, достигшую предельного размера у одной из компонент.

Предпосылки к перегрузке могут складываться на протяжении времени и быть прогнозируемыми. Так, на Уолл-стрит и других мировых биржах объемы электронных торгов возрастают экспоненциально. В потоках рыночных данных каждую секунду генерируются десятки тысяч сообщений. По оценке службы информации о ценах опционов (Options Price Reporting Authority, OPRA), агрегирующей все котировки и сделки, в 2005 г. пиковая скорость генерации сообщений составляла 122,000 сообщений в секунду, и эта скорость ежегодно удваивается.

Но перегрузка может носить и временный, непредсказуемый характер. Так, например, во время событий 11 сентября 2001 года новостной сайт CNN испытывал 20- кратную перегрузку по сравнению с запланированным пиком на протяжении двух с половиной часов. SCADA-системы после ввода в эксплуатацию могут столкнуться с неисправностью оборудования, посылающего на обработку непредусмотренное число команд, а WEB-системы должны быть готовы к атакам, направленным на отказ в обслуживании (denial of service).

В рассмотренных сценариях стрессовой нагрузки под отказоустойчивостью будем рассматривать способность системы продолжать работу максимально продолжительное время, нередко предоставляя человеку возможность вмешаться и выполнить критически важное управляющее воздействие (например, остановка ядерного реактора).

В рассмотренной архитектуре многоэтапных, событийно управляемых информационных систем число системных потоков, ассоциированных с очередью сообщений компоненты, может быть разным. Так, в случае одного системного потока, выделенного на обработку очереди, сообщения будут обрабатываться последовательно, в то время как большее число рабочих потоков позволяет обрабатывать сообщения параллельно. Определяя число системных потоков, выделенных на обработку сообщений каждой компоненты системы, становится возможным распределение вычислительных ресурсов рабочей станции (в первую очередь, процессора) для повышения производительности отдельных компонент за счёт других. Подобное перераспределение ресурсов позволяет отложить время наступления первого отказа системы.

Для многоэтапных событийно-управляемых систем существуют различные подходы к управлению числом системных потоков.

Классическим решением является *фиксированный пул потоков*, где число потоков, выделенных каждой компоненте, определяется администратором эвристически и остаётся неизменным за всё время работы. Подход хорошо зарекомендовал себя при прогнозируемых маршрутах работы системы и известном времени обслуживания каждой из компонент. Однако, если характер поведения системы изменяется (например, изменяется характер появления внешних событий, и, как следствие, изменяется вероятность возникновения тех или иных маршрутов в системе), выбранное однажды число системных потоков может оказаться неоптимальным.

Для преодоления указанного недостатка классического решения *Matthew David Welsh* [3] в своей диссертационной работе предложил адаптивный подход, основанный на наблюдениях системы через равные интервалы времени. По прошествии каждого интервала принимается решение об оптимальном числе системных потоков, выделенных каждой компоненте для минимизации времени обслуживания сообщения компонентой. Для поиска локального минимума времени обслуживания от числа ассоциированных с очередью системных потоков применяются градиентные методы. В дополнение используется эвристика, уменьшающая число системных потоков компоненты в случае их бездействия по таймауту.

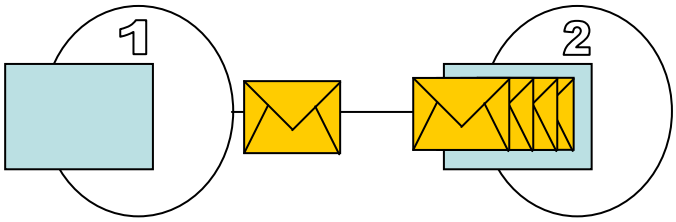
[3] Что имело место против E*Trade [2].

Предложенный Welsh [3] подход позволял системам адаптироваться к изменяющимся условиям функционирования, однако принимаемые эвристикой и градиентными методами решения о числе потоков нередко вступали в противоречия. Сами же градиентные методы не обладали точной оценкой прироста/падения производительности компоненты в зависимости от принятых ранее решений о числе потоков, потому как свой вклад в изменение производительности компоненты вносили абсолютно все решения для всех компонент, так как компоненты разделяют общие вычислительные ресурсы – один процессор и одну оперативную память.

Рассмотренные существующие решения: фиксированный пул потоков и адаптивный подход Welsh [3], стараются достичь максимальной производительности системы, однако для создания систем в жизненно важных для человека областях не менее актуален подход, ориентированный на повышение отказоустойчивости.

Предлагаемый адаптивный подход, как и подход Welsh [3], основан на наблюдениях системы в равные интервалы времени.

Во избежание противоречивых решений о числе системных потоков, решение принимает единственный контроллер из следующих соображений.



$\tau_1(t) = 1$ *сек.* $\tau_2(t) = 4$ *сек.*

рис.2. Время обслуживания двух компонент отличается в 4 раза

Предположим, система состоит из двух компонент, где время обслуживания первой $\tau_1(t) = 1$ *сек.* в 4 раза меньше времени обслуживания второй $\tau_2(t) = 4$ *сек.*, а размер пула потоков *5*. В рассматриваемой системе компонента 1 принимает клиентские сообщения с равным интервалом, передавая на обработку компоненте 2.

В случае, когда обеим компонентам будет выдано по одному системному потоку, первая компонента будет успевать обрабатывать 4 сообщения и передавать их второй за время, пока вторая будет успевать обрабатывать лишь одно. Очередь сообщений второй компоненты начнёт увеличиваться, что приведёт к появлению отказа в системе (рис. 2).

Если второй компоненте выделить 4 системных потока, оставив первой один, то производительность обеих компонент окажется равной – компонента №2 будет успевать обрабатывать весь поток входящих сообщений от первой. Если же частота появления клиентских сообщений в системе превысит возможности системы к обработке, то очереди начнут увеличиваться на обеих компонентах одновременно. Подобный сценарий окажется более предпочтительным, чем первый рассмотренный сценарий (с увеличением очереди только второй компоненты), поскольку отложит время возникновения первого отказа системы за счёт удержания в оперативной памяти большего числа сообщений, ожидающих обработки, чем в первом сценарии.

Обобщая подобные рассуждения на систему с произвольным числом компонент, мы получаем правило, согласно которому число потоков, выделенных отдельной компоненте, определяется пропорциональным суммарному времени обработки сообщений компоненты к суммарному времени обработки сообщений всей сети.

Второй предлагаемый механизм управления (контроллер обратного давления) состоит в приостановке обработки сообщений на компоненте-отправителе, позволяющей компоненте-получателю обработать очередь сообщений, приближающейся к переполнению. В случае нескольких компонент отправителей длительность приостановки назначается пропорционально вкладу каждого отправителя в переполнение очереди получателя. Алгоритм применяется итеративно к каждой компоненте системы.

В основе предложенного механизма лежит алгоритм обратного распространения в обучении многослойного персептрона [4].

По аналогии с нейронными сетями в качестве самой нейросети рассматривается вся совокупность компонент, участвующих во взаимодействии с выбранной. В качестве силы синаптической связи нейронов принимается число сообщений, переданных от компоненты следующей на маршруте компоненте, а в качестве ошибки желаемого сигнала – размер очереди выбранной компоненты.

Алгоритм обратного распространения предполагает изменение синаптических связей с целью минимизации ошибки выходного сигнала. Ослабление синаптической связи в принятой аналогии соответствует уменьшению числа передаваемых сообщений отправителя получателю. В предлагаемом подходе это достигается приостановкой обработки сообщений на отправителе на некоторую штрафную величину миллисекунд. Коэффициент *скорости распространения ошибки* алгоритма обратного распространения в нейросетях применительно к рассматриваемому подходу управления выбирается эвристически с учетом наблюдаемого размера очереди и числа переданных сообщений отправителя получателю. Чем больший вклад в возникновение очереди привнесен отправителем, тем большая величина штрафа тому назначается. Величина штрафа не должна превышать интервал наблюдений системы.

В отличие от алгоритма обратного распространения определение величины штрафа (в рассмотренной аналогии – корректировка синаптических связей) заканчивается на первой итерации. Рассмотренный алгоритм применяется итеративно для каждой компоненты системы.

Литература:

1. V.S. Pai, P.Druschel, W.Zwaenepoel. Flash: An efficient and portable WEB server. Proceedings of the 1999 USENIX Annual Technical Conference, June 1999.

*2. Bloomberg News. E*Trade hit by class-action suit, CNet News.com, February 9, 1999.*

3. M.D. Welsh. An architecture for Highly Concurrent, Well-Conditioned Internet Services. – Phd Thesis, Graduate Division of the University of California at Berkeley, 2002.

4. Bernard Widow, Michael A. Lehr. 30 Years of Adaptive NeuralNetworks: Perceptron, Madaline, and Backpropagation // Artificial Neural Networks: Concepts and Theory, IEEE Computer Society Press, 1922, pp.327-354.

5. D. Carney, U. Cetintemel, M. Cherniack, C. Convey, S. Lee, G. Seidman, M. Stonebraker, N. Tatbul, and S. Zdonik. Monitoring Streams: A New Class of Data Management Applications. In proceedings of the 28th International Conference on Very Large Data Bases (VLDB'02), Hong Kong, China, 2002.

6. A. Arasu, B. Babcock, S. Babu, M. Datar, K. Ito, I. Nizhizawa, J. Rosenstein, and J. Widom. STREAM: The Stanford Stream Data Manager. In ACM SIGMOD Conference, June 2003

7. S. Chandrasekaran, O. Cooper, A. Deshpande, M. J. Franklin, J. M. Hellerstein, W. Hong, S. Krishnamurthy, S. R. Madden, V. Raman, F. Reiss, and M. A. Shah. TelegraphCQ: Continuous Dataflow Processing for an Uncertain World. In Proc. of the 1st CIDR Conference, Asilomar, CA, 2003

8. Богданов А.В., Корхов В.В., Мареев В.В., Станкова Е.Н. Архитектуры и топологии многопроцессорных вычислительных систем. – М.: Интернет-университет информационных технологий – ИНТУИТ.ру, 2004. – 176 с.