Оптимизация на основе статистических данных асинхронной распределенной системы, устойчивой к произвольным отказам*

А.Н. Фирсов

В настоящее время существует большое количество различных алгоритмов, обеспечивающих отказоустойчивость асинхронных распределенных систем. Эффективность использования того или иного алгоритма зависит от сети, на основе которой работает система, и требований пользователя. Многие существующие системы этого факта не учитывают. В данной работе предлагается асинхронная отказоустойчивая распределенная система, которая на основе собираемой статистики и требований пользователя, принимает решения об использовании той или иной оптимизации. Кроме того, собираемая информация позволяет системе подбирать оптимальные параметры используемых алгоритмов, механизма для определения которых в других системах не существует.

1. Введение

С течением времени зависимость людей от техники только увеличивается, а сама техника усложняется. Поэтому остро встает проблема обеспечения надежности и отказоустойчивости техники. Так если вероятность сбоя отдельно взятого компьютера невелика, то в распределенных системах, состоящих из тысяч или даже миллионов таких компьютеров, сбои происходят постоянно.

Византийская модель сбоев [1] (или модель произвольных сбоев) является наиболее общей, включающей в себя все остальные, моделью. Она не накладывает никаких ограничений на действия сбойных процессов и, благодаря этому, позволяет моделировать такие типы сбоев, как отказы оборудования, потеря связи, неисправность оборудования, программные сбои, ошибки операторов и действия злоумышленников.

Особый интерес представляют асинхронные распределенные системы, т.к. они не накладывают никаких временных ограничений, и, благодаря этому, идеально подходят для работы в сети интернет. Наиболее универсальной моделью вычислительного узла является машина с конечным числом состояний (state machine).

В данной работе рассматриваются асинхронные распределенные системы, которые, во-первых, устойчивы к произвольным отказам, во-вторых, позволяют выполнять прикладные программы, требующие сохранять свое состояние между обращениями к ним, в-третьих, обеспечивают отказоустойчивость на программном уровне без необходимости в какой-либо специальной аппаратуре и, в-четвертых, делают все это прозрачно для прикладных программ.

В настоящее время существует несколько таких систем: Rampart [2], SecureRing [3], BFT [4] и т.д. Из-за своей универсальности эти системы очень требовательны к ресурсам (временным, вычислительным, линиям связи), поэтому каждая из них предлагает некоторый набор оптимизаций или модификаций ранее использовавшихся алгоритмов, позволяющий улучшить тот или иной параметр функционирования системы. Однако часто встречаются случаи, когда один параметр улучшается за счет ухудшения другого (а это может быть не в интересах пользователя) или оптимизация не приносит желательного эффекта из-за особенностей аппаратного обеспечения, на основе которого работает распределенная система. Поэтому в разных ситуациях наиболее эффективными могут оказаться разные системы. Однако авторы этих систем в своих работах не очерчивают область, где их система более эффективна по сравнению с аналогами, а лишь приводят результаты тестов наиболее ярко демонстрирующих эффективность предложенных ими оптимизаций.

* Работа выполнена при финансовой поддержке гранта РФФИ 08_07_90005 Бел_а.

Единственный известный метод защиты от произвольных сбоев заключается в активной репликации, т.е. в параллельном выполнении нескольких копий (или, как их еще называют, реплик) одной и той же программы. Поэтому, как правило, для обеспечения отказоустойчивости этими системами предоставляется стек протоколов, обеспечивающий корректное взаимодействие реплик. Стек протоколов включает в себя протокол членства в группе, протокол надежной групповой рассылки и протокол атомарной групповой рассылки. Это позволяет комбинировать протоколы из разных систем и в результате получать оптимальный стек протоколов для каждой конкретной ситуации, применение которого может оказаться эффективнее, чем использование не комбинированного стека существующий системы. Возможно также комбинирование вспомогательных алгоритмов.

В данной работе предлагается отказоустойчивая распределенная система, которая на основе требований пользователя и собранной статистики о прикладной программе и об аппаратуре, на основе которой работает распределенная система, принимает решение об использовании той или иной оптимизации. Кроме того, эти данные позволяют найти оптимальные значения параметров используемых алгоритмов.

2. Параметры эффективности

Различные пользователи и приложения предъявляют отличающиеся друг от друга требования к отказоустойчивым распределенным системам. Так, например, если одна большая задача разделена на множество независимых подзадач и их количество много больше, чем число доступных вычислительных узлов (например, как в проекте SETI@home), то эффективное использование вычислительных ресурсов гораздо важнее, чем общее время выполнения отдельной подзадачи. Противоположная ситуация возникает при выполнении прикладных программ, требующих реакции в реальном времени, в этом случае быстрота реакции на сбой, а значит и общее время работы программы много важнее, чем эффективное использование ресурсов.

Формализуем все описанное выше. Рассмотрим два параметра эффективности: To – общее время работы программы и Tp – количество единиц процессорного времени, потраченного на выполнение программы всеми процессорами. В случае не отказоустойчивой программы эти значения совпадают (здесь рассматриваются последовательные программы; если требуется обеспечить отказоустойчивость параллельной программы, то каждая последовательно выполняющаяся подзадача защищается от сбоев независимо от других). При использовании активной репликации, которая лежит в основе всех существующих устойчивых к произвольным отказам систем, отношение Tp/To примерно равно количеству активных реплик. Это равенство является очень приблизительным из-за возможности возникновения таких ситуаций как задержка сообщения или отказ одной из реплик. Например, в зависимости от используемых протоколов задержка в передаче сообщения может увеличить общее время работы, но не повлиять на количество затраченных единиц процессорного времени, т.к. во время ожидания сообщения процессор может выполнять другие задачи.

Пусть некоторая модификация в обеспечивающей отказоустойчивость системе позволяет выполнить программу за время To', потратив Tp' единиц процессорного времени. В случае если $To - To' \geq 0$ и $Tp - Tp' \geq 0$, то использование этой модификации оправдано независимо от требований пользователя, т.к. оба параметра функционирования улучшаются. Аналогично, если $To - To' \leq 0$ и $Tp - Tp' \leq 0$, то независимо от требований пользователя эту модификацию использовать не следует. Эти два случая однозначны и трудностей в вопросе об эффективности использования модификации не вызывают. Проблема возникает, когда при использовании модификации один из параметров улучшается за счет ухудшения другого, т.е. $(To - To') * (Tp - Tp') < 0$. В системах Rampart, SecureRing и BFT в таких ситуациях разработчики сами принимают решение об использовании модификации и не предоставляют пользователю возможность изменить его в случае возникновения необходимости.

В предлагаемой в данной работе статистически оптимизируемой отказоустойчивой системе пользователю предоставляется возможность задать весовые коэффициенты $o > 0$ и $p > 0$, которые отражают степень важности соответствующих параметров эффективности. Тогда ис-

пользовать модификацию следует только в случае, если $(To - To') * o + (Tp - Tp') * p > 0$. Как видно из формулы, абсолютно значение параметров o и p роли не играют, а важно их отношение o / p. Например, значение отношения $\frac{o}{p} = 2$ означает, что если существует оптимизация позволяющая сократить общее время выполнения программ на t_1 единиц, за счет увеличения использования процессорного времени не более чем на $2 * t_1$ единиц, то эту оптимизацию следует использовать. Точно так же следует использовать оптимизацию, позволяющую сократить используемое процессорное время на t_2 за счет увеличения общего времени выполнения программы не более чем на $2 * t_2$ единиц. Общую эффективность системы можно оценить по формуле $E = To * o + Tp * p$, только в этом случае необходимо потребовать нормализации значений o и p, т.е. $o + p = 1$.

Общее время выполнения программы и количество затраченных единиц процессорного времени являются наиболее важными, но не единственными параметрами эффективности распределенной отказоустойчивой системы. Некоторым пользователям могут оказаться важны такие параметры как объем переданных по сети данных, количество переданных сообщений и т.д. Работа с этими параметрами происходит аналогичным образом. В общем виде формула вычисления эффекта от модификации будет выглядеть так: $O = \sum_i p_i * (P_i - P_i')$, где P_i – параметры эффективности, P_i' – параметры эффективности при использовании некоторой модификации, а p_i – соответствующие весовые коэффициенты. И использовать модификацию следует только в случае если $O > 0$.

3. Статистические данные

Вероятность сбоя отдельно взятого вычислительного узла за единицу времени β является наиболее важной характеристикой аппаратуры, на основе которой работает отказоустойчивая распределенная система. Вычисление этого параметра не представляет никакой сложности, он равен отношению общего количества сбоев некоторого вычислительного узла ко времени работы этого узла. В общем случае в гетерогенной системе вероятности сбоев разных вычислительных узлов могут отличаться, однако в данной статье для упрощения формул будем рассматривать усредненную по всем вычислительным узлам вероятность сбоя.

Практически во всех работах, связанных с отказоустойчивостью [1-8], фиксируется некоторое число f – максимальное количество процессоров, которое может выйти из строя из имеющихся n процессоров, и уже на основе него доказываются все утверждения. Это удобно для проектирования и доказательства корректности алгоритмов, однако в действительности не может быть никакой гарантии, что из строя не выйдет большее число процессоров, просто такая вероятность считается пренебрежимо малой. Обозначим ее через α, тогда при условии, что сбои процессоров независимы (а именно в этом и состоит одна из задач подсистемы балансировки нагрузки), эти величины связаны формулой: $\alpha = \sum_{i=f+1}^{n} C_n^i \beta^i * (1-\beta)^{n-i}$ (1). Каждое слагаемое из правой части формулы представляет собой вероятность того, что именно i процессоров откажут. Вся сумма же берется по i от f+1 до n, т.е. сумма всех вероятностей, что откажут более f процессоров. Параметр α должен определяться пользователем, в зависимости от того какие у него требования к надежности системы в целом, β может быть получен на основе статистики о сбоях или некоторых эмпирических заключений о надежности отдельно взятого вычислительного узла, f вычисляется на основе формулы (1).

Еще одной характеристикой сети, от которой зависят эффективность системы в целом и различных оптимизаций, является задержка при доставке сообщений. Обозначим за P(x) вероятность того, что доставка сообщения от одного процессора к другому занимает меньше времени x. Средняя задержка будет равна математическому ожиданию P(x). Распределение этой слу-

чайной величины сильно изменяется от сети к сети, и может быть получено на основе статистических данных, собранных в сети, в которой работает система. Как и с вероятностью сбоя, эта величина может сильно отличаться для разных пар процессоров, но в данной статье для упрощения формул будем использовать усредненное значение.

В асинхронных системах время, требующееся на доставку сообщения не ограничено, поэтому P(x) может быть меньше 1 для сколь угодно большого, но конечного x. Однако все отказоустойчивы асинхронные распределенные системы требуют, чтобы все сообщения посланные одним корректно работающим процессором другому рано или поздно были доставлены. Это означает, что $\lim_{t->\infty} P(t) = 1$.

На решение об использовании некоторых оптимизаций могут влиять следующие параметры прикладной программы: T_{chp} – время, затрачиваемое на создание контрольной точки, как правило, прямопропорционально объему памяти, который использует прикладная программа, μ – частота обмена сообщениями.

Обозначим за T время работы не отказоустойчивого варианта некоторой программы пользователя на одном процессоре. Тогда при использовании активной репликации для обеспечения отказоустойчивости этой программы от произвольных сбоев для ее выполнения потребуется как минимум n*T единиц процессорного времени, где n – количество активных реплик. В [5] доказано, что для работы любого протокола надежной групповой рассылки требуется как минимум $3*f+1$ процессоров, f из которых могут оказаться сбойными. Поэтому в системах Rampart, SecureRing и BFT используется $n = 3*f+1$ активных реплик. Следовательно, требуется $Tp = (3*f+1)*T$ единиц процессорного времени.

4. Пример оптимизации

В работе [6] был предложен асинхронный протокол надежной групповой рассылки с переменным числом активных реплик, специально создававшийся для статистически оптимизируемой отказоустойчивой системы. Основное его отличие от протокола, приведенного в работе [5] и реализованного в системе BFT [4], состоит в том, что реплики разделены на активные и пассивные (наблюдателей). Результат, доказанный в [5], говорит о том, что для реализации асинхронного протокола групповой рассылки требуется как минимум 3f+1 реплика, однако он не требует, чтобы все они были активными, в том смысле, что на них на всех выполнялась копия пользовательской программы. Для обнаружения сбоя достаточно f+1 активной реплики. И только в случае сбоя возникает необходимость в дополнительных активных репликах. В работе [6] показано, что для маскировки сбоя достаточно наличия 2f+1 активной реплики и f наблюдателей.

Сравним используемый в системе BFT протокол надежной групповой рассылки и предлагаемый протокол с переменным числом активных реплик, использующий простейшую оптимистичную стратегию, описанную в работе [6]. Все остальные используемые протоколы и оптимизации будем считать одинаковыми.

Эффективность протокола надежной групповой рассылки из системы BFT зависит от частоты создания контрольных точек γ. Этот параметр оказывает неоднозначное влияние на эффективность. С одной стороны создание контрольной точки требует определенного времени, и чем чаще они производятся, тем большая часть процессорного времени уходит на это. Однако слишком редкое их создание приводит к медленному восстановлению после сбоев.

Эффективность протокола надежной групповой рассылки так же зависит от частоты создания контрольных точек γ'. Но, кроме того, на нее влияет параметр δ – время, в течение которого активные реплики пытаются прийти к соглашению без привлечения дополнительных процессов. Его влияние так же неоднозначно. Так при уменьшении значения δ активные реплики не успевают прийти к соглашению без привлечения дополнительных процессов, из-за чего увеличивается количество ложных срабатываний системы восстановления, что требует дополнительных вычислительных ресурсов. А слишком большое значение δ приводит к замедлению реакции на действительной произошедший сбой, в результате которого отсутствуют реакции на

сообщения. Невозможно отличит очень медленно работающий процесс от сбойного, не отвечающего на запросы, процесса.

Формулы для вычисления параметров эффективности выглядят следующим образом:

$$Tp = (3 * f + 1) * (T + \gamma * T_{chp} + \frac{\beta * T}{2 * \gamma})$$

$$Tp' = (f + 1) * (T + \gamma' * T_{chp} + \frac{T * f * (\beta + (1 - P(\delta))}{2 * \gamma'})$$

$$To = T * (1 + \mu * 3 * M[P])$$

$$To' = T * (1 + \mu * (3 * M[P] + \frac{\beta * \delta}{2 * \gamma'}))$$

Оптимальные значения γ’ и δ находятся при поиске минимума функции $E(\gamma', \delta) = o * To'(\gamma', \delta) + p * Tp'(\gamma', \delta)$. Для этого решается система из двух уравнений:

$$\begin{cases} \frac{d}{d\gamma'} E(\gamma', \delta) = 0 \\ \frac{d}{d\delta} E(\gamma', \delta) = 0 \end{cases}$$

Как говорилось выше, использование этой оптимизации не всегда может оказаться эффективным. Прежде необходимо проверить выполняется ли условие $(To - To') * o + (Tp - Tp') * p > 0$

Следует также отметить, что если пользователю эффективное использование вычислительных ресурсов гораздо важнее общего времени выполнения программы ($p >> 0$), то в идеале можно получить трехкратную экономию ресурсов: $\lim_{f->\infty, \beta->0, \delta->\infty} \frac{Tp}{Tp'} = 3$. Это достигается за счет того, что общее время работы программы в предлагаемой системе будет больше на величину $T * \mu * \frac{\beta * \delta}{2 * \gamma'}$, которая прямо пропорциональна вероятности сбоя. Поэтому если вероятность сбоя невелика и допустима задержка при его ликвидации, то использование асинхронного протокола групповой рассылки с переменным числом активных реплик предпочтительно, т.к. он обеспечивает гораздо более эффективное использование ресурсов.

Приведенный пример является далеко не единственным случаем, когда для выполнения какой-либо задачи из области обеспечения отказоустойчивости в распределенных системах существует несколько различных алгоритмов, каждый из которых обладает своими достоинствами и недостатками. Опишем вкратце еще несколько таких ситуаций.

В системе BFT [4] для защиты сообщений от фальсификации их сбойными репликами используется симметричное шифрование. Оно дает выигрыш в производительности и требует меньше вычислительных ресурсов, чем цифровые подписи, которые реализованы в системе Rampart [2] с той же целью. Однако алгоритм с использованием цифровых подписей для одной надежной групповой рассылки требует всего лишь порядка O(n) сообщений, в то время как при симметричном шифровании необходимо $O(n^2)$, что становится критичным при росте n. В разных ситуациях оптимальными могут оказаться как цифровые подписи, так и симметричное шифрование, но в каждой из систем реализован только один алгоритм, не говоря уж о механизме выбора оптимального для данной ситуации варианта, который ими вообще не предоставляется.

В работе [3] описаны два различных варианта протокола атомарной групповой рассылки. Первый вариант можно назвать пессимистичным в силу того, что при его использовании система находится в постоянной готовности к возникновению сбоя и, благодаря этому, реагирует на сбой быстрее, чем в случае использования оптимистичного варианта. Но это достигается за счет передачи большего объема сообщений. Первоначально в системе Rampart использовался пессимистичный вариант протокола, однако разработчиками было принято решение заменить его на оптимистичный, т.к. в большинстве ситуаций он оказывается предпочтительнее. Но в

некоторых особо критичных системах реального времени задержка в оптимистичном протоколе при возникновении сбоя может оказаться неприемлемой или значительно более нежелательной, чем передача большого объема сообщений, а возможности использовать пессимистичный вариант пользователю не предоставляется.

В большинстве существующих асинхронных отказоустойчивых систем в основе протокола атомарной групповой рассылки в явном или не явном виде используются ненадежные детекторы отказов [7]. Принципиально другой подход заключается в использовании случайной последовательности чисел и криптографических протоколов разделения секретов. Один из алгоритмов на основе этого протокола описан в [8]. Такие протоколы требуют больше вычислительных ресурсов и, иногда, передачи большего количества сообщений, но их использование может принести выигрыш в надежности.

Проблема выбора оптимального значения различных параметров так же возникает довольно часто. Так, например, ненадежные детекторы отказов для того, чтобы избежать бесконечного ожидания в случае поломки используют таймауты. Если реплика в течение определенного промежутка времени не отвечает на запрос, то она считается сбойной. В асинхронной системе корректно работающая реплика может задержаться с ответом на произвольно большое, но конечное, время. Поэтому при малых значениях задержки корректные реплики часто будут приниматься за сбойные, а при больших значениях реакция на поломку будет замедленной. В существующих системах, использующих ненадежные детекторы отказов в явном (SecureRing) или не явном (BFT, Rampart) виде, величина задержки задается программистом. Механизма для определения оптимального значения в этих системах не предоставляется.

5. Заключение

В работе предложен подход, позволяющий подобрать оптимальный набор протоколов, обеспечивающих защиту от произвольных отказов, для каждой конкретной ситуации. А так же определить значения параметров этих протоколов, при которых будет достигнута максимальная эффективность системы. Продемонстрировано применение этого подхода на конкретном примере.

В настоящее время ведутся работы по созданию системы SOFT (Statistically Optimized Fault Tolerant system), в основе которой лежит данный подход.

Литература

1. Lamport L., Shostak R. and Pease M. The Byzantine Generals Problem. // ACM Transactions on Programming Languages and Systems –July 1982. –Vol. 4, N.3. –P. 382-401.
2. Reiter M. The Rampart Toolkit for Building High-Integrity Services // Selected Papers from the International Workshop on Theory and Practice in Distributed Systems –Springer-Verlag, 1994. – P. 99-110
3. Kihlstrom K. P., Moser, L. E. and Mellar-Smith, P. M. The SecureRing group communication system // ACM Transactions on Information and System Security (TISSEC), –November 2001. – Vol. 4, N. 4. –P. 371-406.
4. Castro M. Practical Byzantine Fault Tolerance // Massachusetts Institute of Technology, 2001. PhD thesis. -172 p.
5. Bracha G. and Toueg S. Asynchronous Consensus and Broadcast Protocols // Journal of the ACM. –October 1985. –Vol. 32, N.4. –P. 824-840.
6. Фирсов А.Н. Асинхронный протокол надежной групповой рассылки с переменным числом активных реплик // Математика программных систем: Межвузовский сборник научных трудов – Пермский университет – 2008. – С.166-177.
7. Chandra T. D. and Toueg S. Unreliable failure detectors for reliable distributed systems // Journal of the ACM. – March 1996. –Vol. 43, N. 2. –P. 225-267.

8. Cachin C., Kursawe K. and Shoup V. Random Oracles in Constantinople. Practical Asynchronous Byzantine Agreement using Cryptography // In Proceedings of 19th ACM Symposium on Principles of Distributed Computing (PODC) – 2000. – P. 123-132.