Повышение достоверности обнаружения состояний гонки данных многопоточных системах

А.Ю. Лобачев
Самарский государственный университет путей сообщения
Самара, Россия
lobachevandrei1996@gmail.com

В.А. Засов
Самарский государственный университет путей сообщения
Самара, Россия
vzasov@mail.ru

***Аннотация*—Предложена методика коллективного тестирования группой детекторов гонки произвольных программ для обнаружения состояний гонки данных многопоточных системах. Разработан алгоритм для генерации программ для тестирования детекторов гонки данных. Предложенные методика и алгоритм позволяют повысить достоверность обнаружения состояний гонки данных в многопоточных системах.**

Ключевые слова— системы, многопоточные, гонки, данные, обнаружение, алгоритмы, достоверность.

1. Введение

Состояния гонки данных возникают в многопоточных параллельных системах, когда несколько потоков одновременно обращаются к одним тем же данным и хотя бы один из потоков выполняет операцию записи, причем порядок этих обращений точно не определен [1].

В этих ситуациях результаты работы программы не детерминированы и зависят от времени или порядка исполнения потоков [1, 2]. Проблема гонки данных приводит ошибкам, которые при попытке обнаружения классическими методиками отладки могут никак не проявлять себя и оставаться скрытыми. Это может оказывать серьезное негативное воздействие на надежность программного обеспечения.

Алгоритмы и реализующие эти алгоритмы программы обнаружения состояний гонки данных, в дальнейшем детекторы гонки данных, разделяются на статические и динамические.

Статические детекторы гонки анализируют исходный код программы и строят её модель, основывающуюся, как правило, на графе потока выполнения и графе использования объектов синхронизации [1, 2, 3].

Динамические детекторы гонки анализируют программу во время её работы, собирая информацию при трассировке программы об истории обращений к памяти и выполнении операций синхронизации [4].

В настоящее время обобщенный универсальный алгоритм обнаружения гонки данных для произвольных программ не создан. Разработанные детекторы гонки являются специализированными и используют оригинальные методы, основанные на особенностях реализуемых в многопоточных системах вычислительных процессов, применяемых структур данных и др.

Например, алгоритм, ESC/Java использует расширенный подход статической проверки ограничений времени компиляции программного кода основанный на автоматическом доказательстве теорем [5]. Алгоритм Chord основан на методике многофазного потокозавимого анализа состояний гонки, который в своей основе применяет языковые средства рефлексии для анализа работы потоков и состояния данных [6]. Алгоритм RccJava основан на расширении системы типов Java для большого числа популярных паттернов проектирования многопоточного кода, что позволяется обнаруживать несинхронизированные операции над данными [7]. Алгоритм RecPlay собирает трассу выполнения потоков в рантайме, а затем на основании собранных данных проверяет отношения happens-before на предмет возникновения гонки используя механизм векторных часов [6].

При индивидуальном использовании детекторов их специализация, а также стохастическая природа возникновений гонки данных в произвольных программах являются причинами, снижающими достоверность результатов анализа о количестве и местах обнаружения гонки данных в программах. Технология применения разработанных детекторов недостаточно автоматизирована и ориентирована на квалифицированных специалистов, что усложняет этот вид тестирования программного обеспечения.

Целью работы является разработка методики и инструментов для ее реализации, позволяющие повысить достоверность обнаружения состояний гонки данных в многопоточных системах.

2. Коллективное тестирование для обнаружения состояний гонки данных в многопоточных системах

Для повышения достоверности обнаружения состояний гонки данных в произвольных программах предлагается производить тестирование группой детекторов и результаты тестирования определять на основе интегрированной обработки результатов анализа полученных группой детекторов. Реализацию коллективного тестирования предлагается производить на разработанном автоматизированном программном комплексе.

Программный комплекс позволяет вычислять основные показатели, определяющие полноту и точность информации о состояниях гонки. Это общее количество обнаруженных состояний гонки, количество ложных состояний гонки, количество пропущенных состояний гонки и соотношения этих показателей.

Пример результатов коллективного тестирования клиент-серверного приложения, представляющего

многопоточный сервис формирования xml документов и содержащего примерно 20000 строк кода, приведен в таблице 1.

ТАБЛИЦА 1. РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ КЛИЕНТ-СЕРВЕРНОГО ПРИЛОЖЕНИЯ

Применяемые алгоритмы	Обнаруженные гонки	Ложные гонки	Время выполнения (с)
ESC/Java	24	11	442
Chord	23	7	384
RccJava	27	9	470
RecPlay	11	0	301(с накладными расходами 342)

На рис. 1 приведены результаты тестирования вышеуказанного приложения, показывающие количество детекторов (всего 4) обнаруживших состояния гонки на каждом из 20 анализируемых участков программы. Если под ложным состоянием гонки считать состояния гонки обнаруженное только одним детектором из группы, то количество ложных состояний гонки 5 (4, 5, 10, 13 и 19 участки программы).

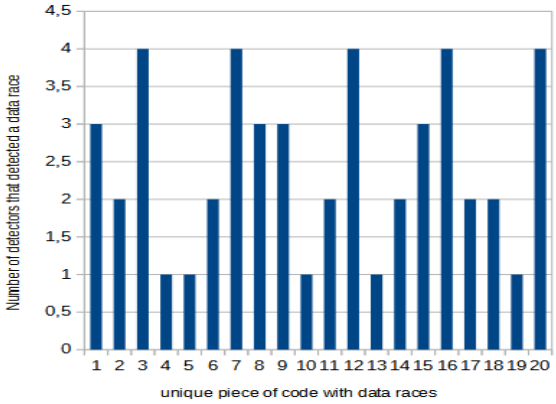

Рис. 1. Результаты тестирования приложения: количество детекторов, обнаруживших гонки данных на каждом из участков кода программы

Устранение состояний гонки данных осуществляется применением средств синхронизации потоков (семафоров, мониторов и др.), что снижает быстродействие программ из-за уменьшения степени параллелизма выполнения потоков. Поэтому игнорирование ложных состояний гонки позволяет избегать ненужную синхронизацию и, следовательно, не снижать быстродействие программ.

Для определения количества пропущенных состояний гонки данных в работе предлагается алгоритм псевдослучайной генерации тестовых программ с заранее известным наличием или отсутствием секций с конкурентным изменением данных. Программный комплекс позволяет задавать размер тестовых программ, количество критических секций в отдельной сгенерированной программе. После завершения тестирования детекторов гонки можно определить количество обнаруженных состояний гонки, соотношения обнаруженных, пропущенных и ложных состояний гонки. Результаты тестирования можно определять по всему набору программ и детально.

На рис. 2 приведены результаты тестирования вышеуказанных 4 детекторов гонки на тестовой программе состоящей примерно из 8500 строк кода и содержащей 37 участков кода с возможными состояниями гонки данных. Тестирование показало 5 пропущенных группой детекторов состояний гонки.

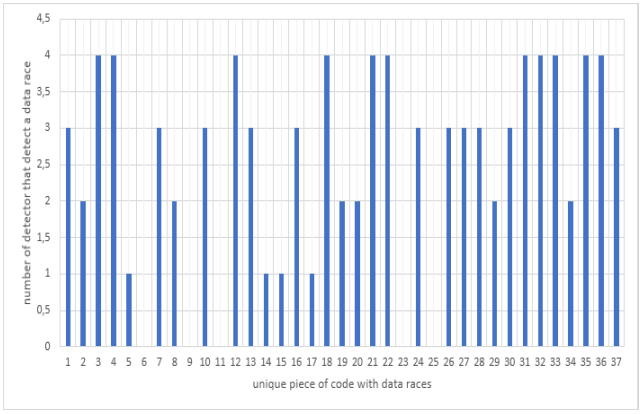

Рис. 2. Результаты тестирования группы детекторов гонки с целью определения количества пропущенный состояний гонки

3. ЗАКЛЮЧЕНИЕ

Предложена методика и программный комплекс для коллективного тестирования группой детекторов гонки произвольных программ для обнаружения состояний гонки данных многопоточных системах. Разработан алгоритм для генерации программ для тестирования детекторов гонки данных.

Предложенные методика и алгоритм позволяют повысить достоверность обнаружения состояний гонки данных в многопоточных системах и надежность разрабатываемого программного обеспечения.

ЛИТЕРАТУРА

[1] Битнер, В.А. Построение универсального линеаризованного графа потока управления для использования в статическом анализе кода алгоритмов / В.А. Битнер, Н.В. Заборовский // Модел. и анализ информ. систем. – 2013. – Т. 20, № 2. – С. 166-177.

[2] Barabanova, P. Modeling and Investigating a Race Condition Detection Algorithm for Multithread Computational Systems / P. Barabanova, V. Zasov // XXI International Conference Complex Systems: Control and Modeling Problems (CSCMP). – Samara, 2019. – P. 356-359. DOI: 10.1109/CSCMP45713.2019.8976855.

[3] Барабанова, П.С. Моделирование и исследование алгоритма обнаружения состояний гонки в многопоточных вычислительных системах / П.С. Барабанова, В.А. Засов // Труды XXI международной конференции «Проблемы управления и моделирования в сложных системах». – Самара: ООО «Офорт», 2019. – С. 195-198.

[4] Трифанов, В.Ю. Динамические средства обнаружения гонок в параллельных программах / В.Ю. Трифанов, Д.И. Цителов // Компьютерные инструменты в образовании. – 2011. – № 5. – С. 3-15.

[5] Flanagan, C. Extended static checking for Java / C. Flanagan, K.R.M. Leino, M. Lillibridge, G. Nelson, J.B. Saxe, R. Stata // Proceedings of the ACM SIGPLAN Conference on Programming Language Design and Implementation. – 2002. – Vol. 37(5). – P. 234-245. DOI: 10.1145/512529.512558.

[6] Java Programm Representation [Electronic resourse]. – Access mode: https://www.seas.upenn.edu/~mhnaik/chord/user_guide/program.html

[7] Flanagan, C. Type-Based Race Detection for Java / C. Flanagan, S.N. Freund // Proceedings of the ACM SIGPLAN Conference on Programming Language Design and Implementation. – 2000. – Vol. 35(5). – P. 219-232. DOI:10.1145/349299.349328.