Подготовка к разработке модели ИИ прогнозирования поведения программ в ОС

Ключевые слова: нейронные сети, трассировка, системные вызовы, предсказание системных вызовов

Аннотация

Знание поведения программ полезно при разработке более совершенных методов планирования процессов в ОС. В данной статье изучаются подходы к предсказанию поведения программ с целью выявления наиболее эффективного метода для прогнозирования системных вызовов. Проведено исследование различных аналогов для модели прогнозирования. Были рассмотрены предиктор последнего значения, предиктор на основе таблиц, унифицированный метод, предиктор на основе дерева решений, а также предсказание на основе Sequence-to-Sequence модели. По результатам анализа работы каждого аналога с точки зрения их точности и производительности, было принято решение использовать предиктор на основе дерева решений в качестве наиболее точной модели прогнозирования. Для выбранного метода прогнозирования был создан набор данных с помощью следующих утилит трассировки: strace и bpftrace. Полученный набор данных был разделен в соотношении 80% для обучения и 20% для оценки результатов обучения нейронной сети.

Введение

Целью работы является повышение точности нейронной сети и улучшения производительности операционной системы Linux при помощи прогнозирования следующего системного вызова конкретных программ.

Предсказание поведения программ позволяет оптимизировать производительность операционной системы и приложений, путем уменьшения загрузки CPU, улучшения использования памяти и уменьшения времени ожидания при выполнении системных вызовов [1], [2], [5].

Объектом исследования является нейронная сеть прогнозирования системных вызовов, предметом исследования - точность предсказания системных вызовов.

Для достижения поставленной цели необходимо выполнить следующие задачи:

- Реализовать трассировку списка процессов, чтобы получать список последовательных событий (системных вызовов).
- Подобрать архитектуру нейронной сети.
- Планирование последующих шагов в работе с нейросетью

Обзор предметной области

Принцип отбора аналогов

Поиск аналогов производился в Google Scholar. Для поиска использовались запросы с ключевыми словами “predicting the behavior of programs”, “predicting the behavior of programs in the OS”, “prediction with system call”. При отборе аналогов учитывалось прогнозирование системных вызовов.

Предиктор последнего значения

Предиктор последнего значения [1] предсказывает, что значение текущего экземпляра инструкции будет таким же, как значение, полученное предыдущим экземпляром. Этот предиктор применяет функцию идентификации к ранее наблюдаемому значению и использует его в качестве следующего прогноза. Эта стратегия довольно простая, но полезна, когда инструкция выдает постоянные значения.

Предиктор на основе таблиц

Предиктор на основе таблиц [1] использует поведение в прошлых интервалах в качестве индекса в таблице для прогнозирования поведения в следующем интервале. Первоначально таблица прогнозирования заполняется на этапе обучения и хранит прогнозируемое значение. В дополнение к прогнозируемому значению в каждой записи таблицы хранится последнее фактическое значение, которое наблюдалось. После получения нового фактического значения, таблица обновляется путем установки на наиболее частое из трех значений: прогнозируемого и последнего значения, сохраненные в таблице, и нового фактического значения.

Унифицированный метод прогнозирования

“Унифицированный метод прогнозирования” (UPM) [2] – метод, который не зависит от системы и показателей для прогнозирования компьютерных показателей. UPM использует параметрическую модель и основана на статистике и данных.

Предлагаются две реализации UPM: 1) целевая функция среднеквадратичной ошибки (MSE); 2) целевая функция накопленной квадратичной ошибки (ASE).

Первая целевая функция приводит к линейному прогнозированию (LP), а вторая - к прогнозированию методом наименьших квадратов (PLS).

Предиктор на основе дерева решений

Дерево решений [3] построенное при помощи алгоритма C4.5 выявляет наиболее важные статические и динамические атрибуты процессов для прогнозирования времени загрузки процессора, что приводит к минимизации времени выполнения процесса TaT (Turn-around-Time).

Предсказание на основе Sequence-to-Sequence модели

В своей архитектуре Sequence-to-Sequence модель [4] использует рекуррентные нейронные сети (RNN) для обработки последовательностей входных и выходных данных. Рекуррентная нейронная сеть используется для моделирования последовательности системных вызовов, а модель Sequence-to-Sequence применяется для генерации прогнозируемых последовательностей, обусловленных данным контекстом системного вызова. Предложенная прогностическая модель предсказывает, какие системные вызовы будут выполнены впоследствии, что позволит наблюдать состояние системы и прогнозировать атаки.

Критерии сравнения аналогов

Точность прогнозирования

Точность прогнозирования системных вызовов зависит от нескольких факторов, включая используемые алгоритмы прогнозирования, объем доступных данных и качество обучающей выборки. Чтобы оценить точность прогнозирования, данные разделяют на обучающий и тестовый наборы. Одним из распространенных методов оценки точности прогнозирования является процент правильно предсказанных системных вызовов относительно общего числа вызовов.

Производительность

Критерий производительности компьютерной системы оценивает эффективность использования ресурсов ПК, включая центральный процессор (ЦПУ), память и другие, сравнивая уровень использования ЦПУ до и после применения предсказания системных вызовов с использованием параметра Turn Around Time (TaT). Turn-around-Time (TaT) — это показатель производительности, используемый для измерения времени, требуемого для выполнения определенного процесса или задачи, который вычисляется, как интервал времени от начала выполнения процесса до его полного завершения.

Устойчивость к шуму

Устойчивость к шуму нейронной сети означает, насколько хорошо она способна справляться с внешними помехами, сохраняя при этом свою точность. Это является важным свойством нейронной сети, поскольку реальные данные могут содержать шум или ошибки.

Лицензирование ОС

Лицензирование операционной системы — это процесс получения разрешения на использование операционной системы. Существуют два основных типа лицензий: проприетарные и открытые. Проприетарные лицензии обычно требуют оплаты и предоставляют пользователю ограниченные права на использование программного обеспечения. Открытые лицензии, такие как лицензии GNU General Public License (GPL),

предоставляют пользователю более широкие права, включая право на изменение и распространение программного обеспечения.

Таблица сравнения по критериям

Критерий/ Аналог	**Точность**	**Лицензирование ОС**	**Производительность**	**Устойчивость к шуму**
Предиктор последнего значения	88%	Коммерческое	Эксперименты по вычислению производительности не проводились	Небольшое увеличение ошибки прогнозирования до 6% от исходного
Предиктор на основе таблиц	92%	Коммерческое	Эксперименты по вычислению производительности не проводились	Небольшое увеличение ошибки прогнозирования до 6% от исходного
Унифицированный метод прогнозирования	94%	Коммерческое	Эксперименты по вычислению производительности не проводились	Эксперименты по влиянию шума не проводились
Предиктор на основе дерева решений	94%	Коммерческое/ Свободное	уменьшение TaT 1,4-5,8%	Эксперименты по влиянию шума не проводились
Предсказание на основе Sequence-to-Sequence модели	93%	Коммерческое/ Свободное	Эксперименты по вычислению производительности не проводились	Эксперименты по влиянию шума не проводились

Выводы по итогам сравнения

Предиктор последнего значения обладает наименьшей точностью среди остальных методов. Предиктор на основе таблиц, унифицированный метод, предиктор на основе дерева решений, предсказание на основе Sequence-to-Sequence модели обладают практически одинаковой точность. Предиктор последнего значения, предиктор на основе таблиц, унифицированный метод прогнозирования используются в архитектуре IBM Power и ОС AIX, которая является коммерческой проприетарной операционной системой, разработанной компанией IBM для серверов, имеет высокую производительность, надежность и поддержку различных аппаратных компонентов. Предиктор на основе дерева решений, предсказание на основе Sequence-to-Sequence модели используются в

архитектуре x86 и ОС Linux, которая является свободно распространяемой операционной системой с открытым исходным кодом и может быть использована на различных типах компьютеров или серверов, имеет большое сообщество разработчиков и пользователей, что приводит к быстрому внедрению новых технологий и исправлению ошибок. Предиктор на основе дерева решений позволяет снизить TaT в диапозоне от 1,4% до 5,8%, для остальных решений эксперименты по вычислению производительности не проводились. Для предиктора последнего значения и предиктора на основе таблиц проведено два эксперимента по оценке влияния шума, результатом которого является небольшое увеличение ошибки прогнозирования до 6% от исходного, что является не существенным.

Выбор метода решения

На основе сравнения аналогов решение должно удовлетворять следующим условиям:

1. Повышенной точностью прогнозирования, не меньше, чем на существующих решениях, чтобы успешно предсказывать следующие системные вызовы.
2. Обучение нейросети на датасете системных вызовов, содержащих информацию о контексте, включая указание на используемое приложение и выполняемое действие.
3. Обеспечение более точной работы обученной нейросети за счет большого количества тестов.
4. снижение TaT.

Описание метода решения

Для выполнения условия повышенной точности выбрана модель предсказания на основе дерева решений. Указанная модель должна быть обучена на подготовленном наборе данных. Набор данных формируется с помощью утилит трассировки: strace и bpftrace. strace - один из самых популярных инструментов для трассировки системных вызовов. Он позволяет отслеживать все системные вызовы, производимые программой, и выводить подробную информацию о них, такую как тип вызова, аргументы, возвращаемое значение и время выполнения. bfptrace позволяет отслеживать контекст программы, для более точного предсказания системных вызовов. Модель обучается в соотношении 80% для обучения и 20% для оценки результатов обучения нейронной сети.

Для создания и обучения нейронных сетей доступны различные инструменты и библиотеки, и одним из наиболее популярных языков программирования для этой цели является Python [6]. Python предлагает широкий спектр библиотек, таких как TensorFlow, PyTorch и Keras, которые предоставляют необходимые инструменты и функциональность для разработки и обучения нейронных сетей. Благодаря этому разнообразию средств Python является предпочтительным языком программирования для разработки и обучения нейронных сетей.

Заключение

В последние годы прогнозирование поведения программ уделяется все больше внимания и

исследователи решают проблему с помощью различных подходов.

Был проведен сравнительный анализ существующих моделей для предсказания поведения программ.

В результате исследования была выбрана модель предсказания на основе дерева решений, поскольку она обеспечивает наибольшую точность предсказания среди рассмотренных моделей, а также позволяет улучшить производительность ОС.

Для дальнейшего развития этой работы планируется реализовать обучение нейронной сети, провести дополнительный анализ и эксперименты для подтверждения эффективности разработанного алгоритма. Это позволит более полно исследовать потенциал модели на основе дерева решений для прогнозирования поведения программ и дополнительно оптимизировать производительность операционной системы

Список литературы

[1] Duesterwald E., Cascaval C., Dwarkadas S. Characterizing and predicting program behavior and its variability //2003 12th International Conference on Parallel Architectures and Compilation Techniques. – IEEE, 2003. – С. 220-231.

[2] Sarikaya R., Buyuktosunoglu A. A unified prediction method for predicting program behavior //IEEE Transactions on Computers. – 2009. – Т. 59. – №. 2. – С. 272-282.

[3] Negi A., Kumar P. K. Applying machine learning techniques to improve linux process scheduling //TENCON 2005-2005 IEEE Region 10 Conference. – IEEE, 2005. – С. 1-6.

[4] Lv S. H. et al. Intrusion prediction with system-call sequence-to-sequence model //IEEE Access. – 2018. – Т. 6. – С. 71413-71421.

[5] Xu C. et al. Cache contention and application performance prediction for multi-core systems //2010 IEEE International Symposium on Performance Analysis of Systems & Software (ISPASS). – IEEE, 2010. – С. 76-86.

[6] Thaker N., Shukla A. Python as multi paradigm programming language //International Journal of Computer Applications. – 2020. – Т. 177. – №. 31. – С. 38-42.