004.896

В. С. Белавин[1], *К. Арзыматов*[1], *М. Е. Карпов*[1], *А. А. Сапронов*[1,2], *А. Е. Устюжанин*[1,2]

[1]Национальный исследовательский университет «Высшая школа экономики»
[2]Объединенный институт ядерных исследований
[3]Московский физико-технический институт (государственный университет)

Настройка цифровых двойников системы хранения данных методами обучения подкреплением

В статье рассматривается задача настройки дискретно-событийного симулятора системы хранения данных с помощью нейроморфных алгоритмов и методов обучения с подкреплением. Симулятор обладает некоторым набором входных параметров, на которые можно влиять во время работы. Изменение этих параметров влияет на реалистичность поведения модели. Задача настройки симулятора состоит в поиске оптимальной стратегии воздействия на параметры модели с целью повышения реалистичности. В работе продемонстрирована состоятельность такого подхода и его возможность к обобщению имеющихся закономерностей. Таким образом объединение симулятора и искусственной нейронной сети позволяют значительно упросить разработку точных моделей систем и процессов.

Ключевые слова: система хранения данных, обучение с подкреплением, оптимальное управление, цифровой двойник

1. Введение

Мы предлагаем подход, который объединяет использование симуляторов, основанных на математических моделях и моделях машинного обучения. Первые обычно основываются на экспертном опыте, вторые строятся по накопленным данным. Предполагается, что математическая модель симулятора обладает некоторым набором управляемых параметров, на которые можно оказывать воздействие в процессе симуляции. Управление параметрами делегируется нейронной сети, обученной методами обучения с подкреплением на реальных данных. Тандем из традиционной симуляционной модели и нейроморфной системы управления может стать перспективным направлением для разработки цифровых двойников физических систем или процессов, позволяющих выполнять задачи диагностики и предсказания их поведения [1].

Одним из ограничений симуляторов, основанных на экспертных знаниях, является жёсткость задания математических моделей, лежащих в их основе. Такие модели могут упускать из виду некоторые тонкие моменты в поведении системы, которые, однако, наблюдаются в статистических данных, снятых с моделируемой системы. Кроме того, появление новых данных и/или изменение режимов работы моделируемой системы часто приводит к необходимости пересоздания симулятора.

С другой стороны, методам симуляции в машинном обучения, например, генеративно-состязательным сетям, нужны только данные обучения внутреннего представления симулируемой системы. Неоспоримым плюсом данного подхода является отсутствие какой-либо модели, закладываемой в систему, но это является и проблемой, так как, во-первых, сильно усложняет возможность интерпретации обученной модели и, во-вторых, может быть неадекватна в тех режимах функционирования системы, которые не были представлены в данных.

Наш подход выгодно отличается от байесовской оптимизации или статистических методов оценки параметров тем, что первые дают некоторую точечную оценку параметров симулятора. Однако оптимальные параметры могут быть функцией от профиля нагрузки и/или условий окружающей среды и поэтому могут варьировать во времени. Нейронная сеть же может восстановить функциональную зависимость оптимальных параметров от условий среды.

2. Симулятор

Программный комплекс GoTatlin создан для симуляции системы хранения данных (СХД) ТАТЛИН.

2.1. Моделируемая система

Система Хранения Данных ТАТЛИН (рис. 1) представляет собой комплекс аппаратных вычислительных средств и специального программного обеспечения (СПО), предназначенный для хранения и передачи данных больших объемов.

Благодаря унифицированному протоколу доступа(Hybrid Unified Storage) данная СХД поддерживает широкий ряд накопителей: NVMe/SAS SSD и SAS/NL-SAS/SATA HDD.

Структурно ТАТЛИН состоит из следующих основных компонент:

- PCIe-фабрики — маршрутизатор шины PCI Express, обеспечивающий взаимодействие контроллеров хранения и носителей информации;
- контроллеры хранения — серверная ЭВМ, предназначенная для управления доступом к данным;
- дисковые массивы (JBOD).

Вместо устаревшей технологии RAID целостность данных обеспечивают собственные механизмы защиты на базе кодов Рида-Соломона с минимальной избыточностью. Операция сохранения данных с использованием такой технологии требует больших вычислительных мощностей и может занимать существенное время.

Существенно снизить нагрузку на центральные процессоры в операциях при дедупликации данных позволяет наличие встроенного в контроллеры хранения энергонезависимого (NVRAM) кэша ёмкостью до 2 ТБ.

2.2. Описание симулятора

Симулятор, здесь и далее GoTatlin (рис. 2), представляет собой программу симуляции работы СХД ТАТЛИН реализованную на языке программирования Go [2]. Реализация моделирования осуществляется на основе дискретно-событийной модели, где изменения системы представляются хронологической последовательностью событий. События происходит в определенные моменты времени и знаменуют собой изменение состояния системы.

2.3. Выходные данные симулятора

Ниже представлен неполный список выходных данных симуляции, которые используются в процессе обучения нейросетей входящих в систему управления параметров симулятора.

- параметры контроллеров хранения;
  - трафик контроллера хранения;
  - загрузка контроллера хранения;

Рис. 1. СХД ТАТЛИН

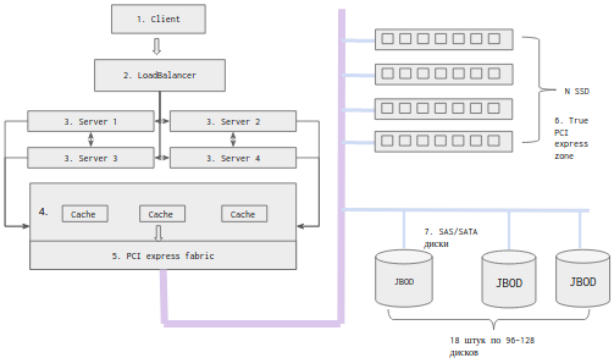

Рис. 2. Схема симулятора GoTatlin

- параметры носителей информации (НИ):
  - занятое место на НИ;
  - средняя скорость чтения/записи;
- параметры функционирования СХД:
  - объем передаваемых данных в режиме чтения/записи;
  - количество запросов на чтение/запись в единицу времени;
  - время отклика на запросы чтения/записи;
  - время обработки запросов на чтение/запись;
  - способ обработки операций ввода-вывода;

Выходные данные генерируются симулятором с фиксированным размером шага, который задаётся заранее.

2.4. Управляющие параметры

Входные параметры симулятора или по-другому управляющие параметры естественно определяются для всех компонент системы:

- носители информации:
  - *read* – скорость на чтение;
  - *write* – скорость на запись;
- дисковые массивы:
  - *speed* – вычислительная мощность;
- контроллер хранения:
  - *speed* – вычислительная мощность;
- фабрика PCIe-фабрика:
  - *speed* – вычислительная мощность;
- сеть соединяющая разные компоненты СХД:
  - *bandwidth* – пропускная способности сети;
  - *latency* – задержка сети.

3. Управляющая система

Управляющая система производит настройку параметров во время работы симулятора.

В основе управляющей системы находится нейронная сеть, которая на вход принимает выходные данные симулятора, а на выходе генерирует управляющие параметры симулятора, которые подаются на вход симулятора.

3.1. Обучение с подкреплением

В обучении с подкреплением вводятся следующие термины:

- среда (s_t) – система, на которую оказывается некоторое воздействие(в нашем случае это симулятор);
- агент – система, которая оказывает управляющее воздействие(a_t) на среду(наш нейросетевой алгоритм);
- выигрыш (r_t) – мера успешности достижения некоторой цели, к примеру, в приведении среды к некоторому состоянию.

В нашем случае выигрышем является то на сколько данные генерируемые симулятором стали более похожими на реальные данные. Более подробно о том как определяется степень похожести будет рассказано в сек. 3.2. Сейчас же рассмотрим как решать задачу максимизации похожести в предположении что выигрыш определён.

Для решения этой проблемы наиболее плодотворным является предположение о марковости зависимостей состояния среды (s_t) и действий агента (a_t). В таком случае модель описывается марковским процессом принятия решений. Для определения марковского процесса нужно задать 4-кортеж $(S, A, P(\cdot,\cdot,R(\cdot,\cdot)))$, где

- $S = \{s_i\}_{i=0}^{N}$ — множество состояний среды;
- $A = \{a_i\}_{i=0}^{K}$ — возможные действия агента;
- $P(s_{t+1}|s_t, a_t)$ — вероятности перехода среды из состояния s_t в момент времени t в состояние s_{t+1} в следующий момент времени $t+1$ при воздействии агента a_t;
- $r(s_{t+1}, s_t) = r(s_t, a_t)$ — выигрыш от перехода из состояния s_t в момент времени t в состояние s_{t+1} в следующий момент времени $t+1$ из-за воздействия a_t.

Развитием данного формализма является Q-обучение.

Наиболее широкое распространения нашёл подход [3] когда агент, т.е. нейронная сеть, генерирует плотность вероятности над возможными действиями при заданном состоянии среды $\pi_\theta(a|s)$ и действие выбирается случайным образом из этой плотности: $a_t \sim \pi_\theta(s)$.

Максимизацию пользы от действий агента во всех возможных состояниях системы можно переформулировать как максимизацию следующего математического ожидания:

$$\mathcal{L}(\pi_\theta) = \mathop{\mathbb{E}}_{s,a}[r(s,a)] = \int\limits_S \rho(s) \int\limits_A \pi_\theta(s) ds da$$

В таком случае обучение производится методом обратного распространения ошибки и градиентного спуска:

$$\nabla_\theta \mathcal{L}(\pi_\theta) = \mathop{\mathbb{E}}_{s,a}[\nabla_\theta \pi_\theta(s) r(s,a)]$$

Однако такой прямой подход даёт очень медленную скорость сходимости. Это вызвано тем что размерность пространства в нашей задаче очень высокая. С этим было решено

бороться использованием алгоритма DDPG(Deep Deterministic Policy Gradients) [4], который позволяет добиться низкой дисперсии при оценке $\mathcal{L}(\pi_\theta)$, а, следовательно, снижение ошибки в расчёте $\nabla_\theta \mathcal{L}(\pi_\theta)$.

В отличии от классической реализации, где нейронная сеть предсказывает плотность вероятности над всеми допустимыми действиями в состоянии s, в DDPG предсказывается сразу конкретное действие. Дополнительно к действию добавляется случайный шум, который генерируется из процесса Орнштейна-Уленбека. Другими словами, шум генерируется следующим образом:

$$n_{t+1} = n_t + \theta(n_0 - n_t) + \sigma W_t,$$

где θ определяет как быстро происходит регрессия к среднему(n_0), W_t — Винеровский процесс, σ — дисперсия процесса. Само же действие, которое выдаёт нейронная сеть модифицируется путём добавления этого шума:

$$\hat{a}_t = a_t + n_t.$$

Такой выбор шума обеспечивает достаточный уровень "любознательности" алгоритма. Кроме того, важной особенностью процесса Орнштейна-Уленбека является его свойство регрессии что обеспечивает большую гладкость в изменении управляемых параметров, а значит более низкую дисперсию в сравнении с семплированием из распределения как предлагается в классической реализации.

3.2. Метрики

Для сравнения выхода реальной системы и симулированной реализован следующий набор метрик, которые выбираются в соответствии с физическими особенностями процесса стоящего за генерацией той или иной величины:

- косинусная метрика близости Фурье-спектров или вейвлетных спектров;
- среднеквадратичная ошибка;
- статистика Колмогорова(KS), расстояние Васерштейна(W), статистика Крамера-Мизеса-Смирнова(CvM) между распределениями статистик посчитанных по реальным данным и симулированным.

Искалась функция которая бы как можно лучше показывала различие, но при этом была бы достаточно робастна к небольшим изменениям в наблюдаемых переменным, чтобы наш алгоритм не пытался выучить неизбежный шум.

В итоге была составлена следующая функция:

$$R = \frac{|S_t^{ref} - S_t^{controlled}|}{10^7} + \frac{CvM(WV_{1,\dots,t}^{ref}, WV_{1,\dots,t}^{controlled})}{20} + \frac{CvM(WQ_{1,\dots,t}^{ref}, WQ_{1,\dots,t}^{controlled})}{2},$$

где

- S_t — суммарный объём данных записанных на диски к моменту времени t;
- $WV_{1,\dots,t}$ — эмпирическая функция распределения объема передаваемых данных в режиме записи с момента времени $t' = 0$ до $t' = t$;
- $WQ_{1,\dots,t}$ — эмпирическая функция распределения количества запросов на запись с момента времени $t' = 0$ до $t' = t$;

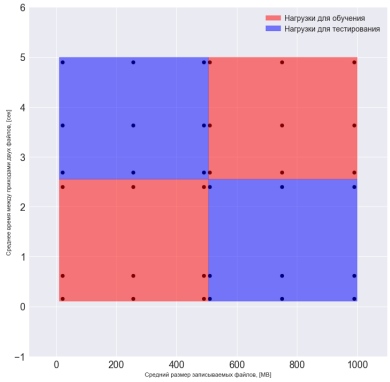

Рис. 3. Область сэмплирования параметров нагрузки. Нагрузки используемые для обучения семплируются из красных областей, а для тестирования алгоритма из синих.

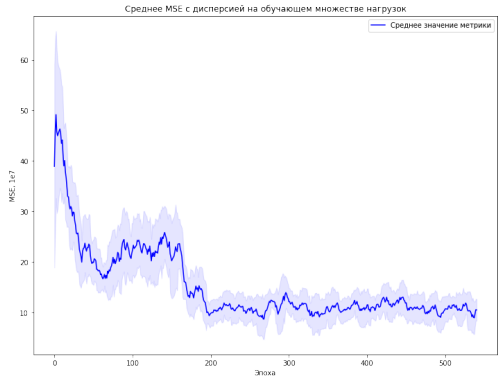

Рис. 4. Среднее значение метрики на множестве нагрузок используемых в обучении с доверительным интервалом. Чем меньше значение метрики, тем ближе наша система к референсной.

4. Настройка параметров управляющей системы

4.1. Постановка экспериментов

Эксперимент ставится следующим образом: берётся две разные конфигурации системы. Первая конфигурация считается референсной, и её выход мы учимся повторять как можно более точно.

Вторая конфигурация является начальным приближением для нашего алгоритма, который пытается так изменять параметры симулятора в ходе работы, чтобы как можно более точно по предложенной метрике повторить выход референсной системы.

Сделано одно упрощение: конфигурации отличаются по двум параметрам: вычислительная мощность контроллера хранения и пропускная способность сети. Это сделано из-за того что изменение именно этих параметров оказывает наиболее сильное влияние на выход симулятора.

Кроме того, в предположении что в реальной системе мы можем не знать хороших начальных приближений параметры для настраиваемой системы нарочно были выбраны сильно отличными от параметров референсной системы.

Нас в первую очередь интересует отклик системы на внешнее воздействие, т.е. внешнюю нагрузку. Нагрузка по времени моделируется как пуассоновский процесс с некоторым средним временем $\bar{\tau}$[сек.], а размер файлов моделируется экспоненциальным распределением с параметром среднего размера файла $\bar{s}$[Мб]. Для настройки алгоритма рассматривается конечное множество возможных нагрузок, которое выразимо через декартово произведение следующим образом: $L = S \times T = \{s_i\}_{i=1}^{N} \times \{\tau_j\} j = 1^K$.

Тренировочные нагрузки при которых мы будем обучаться L_{train} и валидационные нагрузки, которые управляющая система раньше не видела, L_{test} семплируются из равномерных распределений представленных в виде областей в двумерном пространстве(рис. 3). Важно обратить внимание на тот факт, что области семплирования не пересекаются, что очень важно при тестировании алгоритмов. Неформально это можно представить следующим образом: мы учимся повторять поведение симулятора на нагрузках, которые представлены редкими и большими файлами или частыми и маленькими файлами, а проверяем работу нашего алгоритма на больших и часто приходящих файлах и на маленьких и редких файлах.

4.2. Результаты экспериментов

Позитивным результатом является то что и на тренировочной(рис. 4), и на валидационной(рис. 5) выборке параметров нагрузки замечено планомерное снижение значения нашей

метрики.

Это свидетельствует о двух вещах:

- наша нейронная сеть может эффективно управлять параметрами симулятора и сближать симулятор с референсной системой по важным для нас метрикам;
- обученная нейронная сеть способна обобщаться и показывать хорошее качество в условиях отличных от тех в которых она обучалась.

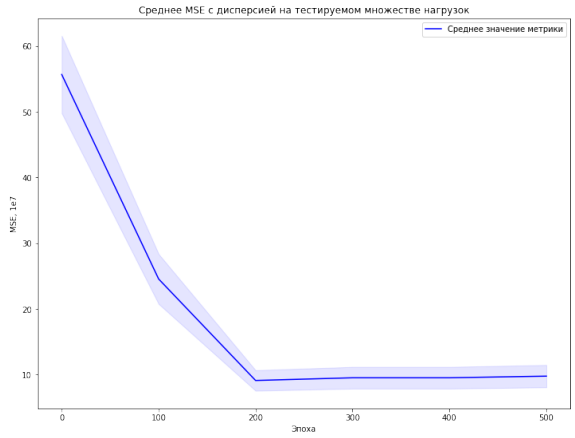

Рис. 5. Среднее значение метрики на множестве нагрузок для тестирования с доверительным интервалом.

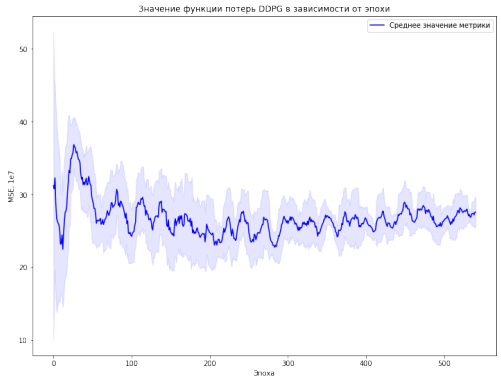

Рис. 6. Значение функционала $\mathcal{L}(\pi_\theta)$ от эпохи.

На рис. 6 показано значение $\mathcal{L}(\pi_\theta)$ на этапе обучения. Так как данный функционал при расчёте включает в себя выход нейронной сети, то поведение его значения не обязано согласоваться со значением метрик. Однако наблюдение данного функционала позволяет судить о стабильности обучения и переобучения. К примеру, наблюдаемые затухающие осцилляции свидетельствуют о сходимости процесса обучения. В пределе данный функционал должен сойтись к некоторой константе.

5. Заключение

Таким образом, мы показали возможность использование связки нейронной сети и реального симулятора посредством обучения с подкреплением для более точной настройки под референсную систему. Данный подход может быть использован для дальнейшего развития технологии цифровых двойников, которые могут быть использованы для диагностики и предсказания состояний реальных систем.

В дальнейшем планируется проведение экспериментов по следующим направлениям:

- увеличение количества управляемых параметров;
- поиск более полно описывающую различие выходов метрику;
- настройка симулятора на реальную систему по реальным данным.

Исследование выполнено при финансовой поддержке Министерства науки и высшего образования Российской Федерации в рамках Федеральной целевой программы «Исследования и разработки по приоритетным направлениям развития научно-технологического комплекса России на 2014-2020 годы». Уникальный идентификатор – RFMEFI58117X0023, соглашение №14.581.21.0023 от 03.10.2017.

Литература

1. *Tao F., Cheng J., Qi Q., et al* Digital twin-driven product design, manufacturing and service with big data // The International Journal of Advanced Manufacturing Technology, 94, 8 — 2018. — P. 3563–3576.
2. *Kincaid, Jason* Google's Go: A New Programming Language That's Python Meets C++ // TechCrunch, 2009.
3. *Mnih V. at al.* Playing Atari With Deep Reinforcement Learning // NIPS Deep Learning Workshop . — 2013.
4. *Silver D. at al.* Deterministic Policy Gradient Algorithms // Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32 . — 2014. — P. I-387–I-395.