Средства симуляции ЦП и ОС и изучение поведения программ

Лекция №7

Державин Андрей
Шурыгин Антон

- **Квиз**
- Моделирование с использованием трасс
- Применение трасс
- Практическая часть
- Домашнее задание №4

Код викторины:

00712895

- Квиз
- **Моделирование с использованием трасс**
- Применение трасс
- Практическая часть
- Домашнее задание №4

Моделирование на основе трасс

- Симуляция на основе трасс (англ. trace driven simulation) базируется на возможности использования истории дискретных событий для нужд симуляций
- Трасса – истории событий, произошедших в системе за определенный период времени и сохраненные в порядке их возникновения в файл.

История событий в симуляции

- Текстовое представление трассы может иметь следующий вид:

```
time=10  read addr=0x45df4 result=0x0455
time=14  write addr=0x35df4 data=0xffff
time=20  interrupt number=10
time=25  port write addr=0x10 data=0xabcd
```

История событий в симуляции

- Текстовое представление трассы может иметь следующий вид:

```
time=10  read addr=0x45df4 result=0x0455
time=14  write addr=0x35df4 data=0xffff
time=20  interrupt number=10
time=25  port write addr=0x10 data=0xabcd
```

В данном примере отражены характерные составляющие составляющие:

- time – моменты времени возникновения событий;
- read, write, interrupt, port write – описание типа события;
- addr, number – параметры события;
- result – результаты выполнения.

История событий в симуляции

- Текстовое представление трассы может иметь следующий вид:

```
time=10  read addr=0x45df4 result=0x0455
time=14  write addr=0x35df4 data=0xffff
time=20  interrupt number=10
time=25  port write addr=0x10 data=0xabcd
```

- Экономия вычислительных ресурсов при повторном исполнении за счет увеличения потребления памяти.

Процесс сбора трасс

- ➢ Данные собранные в процессе трассировки должныы достаточно точно отражать поведение исходной системы
- Обычно процесс сбора трасс разбивается на несколько стадий

Процесс сбора трасс

- Данные собранные в процессе трассировки должныы достаточно точно отражать поведение исходной системы

➢ Обычно процесс сбора трасс разбивается на несколько стадий

Процесс сбора трасс

- Данные собранные в процессе трассировки должныы достаточно точно отражать поведение исходной системы
- Обычно процесс сбора трасс разбивается на несколько стадий
  - ➢ Характеризация изучаемого сценария – анализ наиболее интересных фаз для исследования, особенностей изучаемого сценария в целом.

Процесс сбора трасс

- Данные собранные в процессе трассировки должныы достаточно точно отражать поведение исходной системы
- Обычно процесс сбора трасс разбивается на несколько стадий
  1. Характеризация изучаемого сценария
  ➢ Выбор технологии записи трасс – определение методов детектирования происходящих событий в системе и места хранения собранных трасс (данных).

Процесс сбора трасс

- Данные собранные в процессе трассировки должныы достаточно точно отражать поведение исходной системы
- Обычно процесс сбора трасс разбивается на несколько стадий
  1. Характеризация изучаемого сценария
  2. Выбор технологии записи трасс
  - ➢ Выбор интервалов трассировки – определение интервалов в работе сценария, для которых записанные данные будут достаточно точно отражать поведение системы

Процесс сбора трасс

- Данные собранные в процессе трассировки должныы достаточно точно отражать поведение исходной системы
- Обычно процесс сбора трасс разбивается на несколько стадий
  1. Характеризация изучаемого сценария
  2. Выбор технологии записи трасс
  3. Выбор интервалов трассировки
  - ➢ Подготовка сценария – модификация изумачемого ПО/оборудования с целью иметь возможность снимать интересующие события/сигналы.

Процесс сбора трасс

- Данные собранные в процессе трассировки должныы достаточно точно отражать поведение исходной системы
- Обычно процесс сбора трасс разбивается на несколько стадий
  1. Характеризация изучаемого сценария
  2. Выбор технологии записи трасс
  3. Выбор интервалов трассировки
  4. Подготовка сценария
  - ➢ Валидация – проверка, что полученная трасса верно отражает поведения сценария с допустимой погрешностью

Процесс сбора трасс

- Данные собранные в процессе трассировки должныы достаточно точно отражать поведение исходной системы
- Обычно процесс сбора трасс разбивается на несколько стадий
  1. Характеризация изучаемого сценария
  2. Выбор технологии записи трасс
  3. Выбор интервалов трассировки
  4. Подготовка сценария
  5. Валидация
  - ➢ Публикация трассы

Процесс сбора трасс

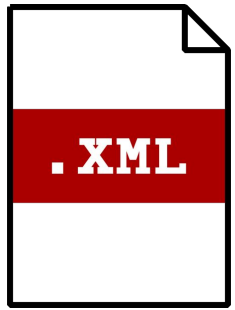
- Данные собранные в процессе трассировки должныы достаточно точно отражать поведение исходной системы
- Обычно процесс сбора трасс разбивается на несколько стадий
  1. Характеризация изучаемого сценария
  2. Выбор технологии записи трасс
  3. Выбор интервалов трассировки
  4. Подготовка сценария
  5. Валидация
  6. Публикация трассы

Форматы хранения трасс

- Обычно трассы хранятся прямо на жестком диске в файле.
- Файлы разделяют на две категории:
  - Текстовый формат
  - Бинарный формат

Форматы хранения трасс

- Обычно трассы хранятся прямо на жестком диске в файле.
- Файлы разделяют на две категории:
  - ➢ Текстовый формат
    - ■ XML, JSON, CSV
    - ■ Содержимое файла без труда может быть изучено человеком
  - ○ Бинарный формат



Форматы хранения трасс

- Обычно трассы хранятся прямо на жестком диске в файле.
- Файлы разделяют на две категории:
  - Текстовый формат
  - ➢ Бинарный формат
    - Обработка двоичного формата для симулятора более удобная и быстрая
    - Общепринятого формата нет

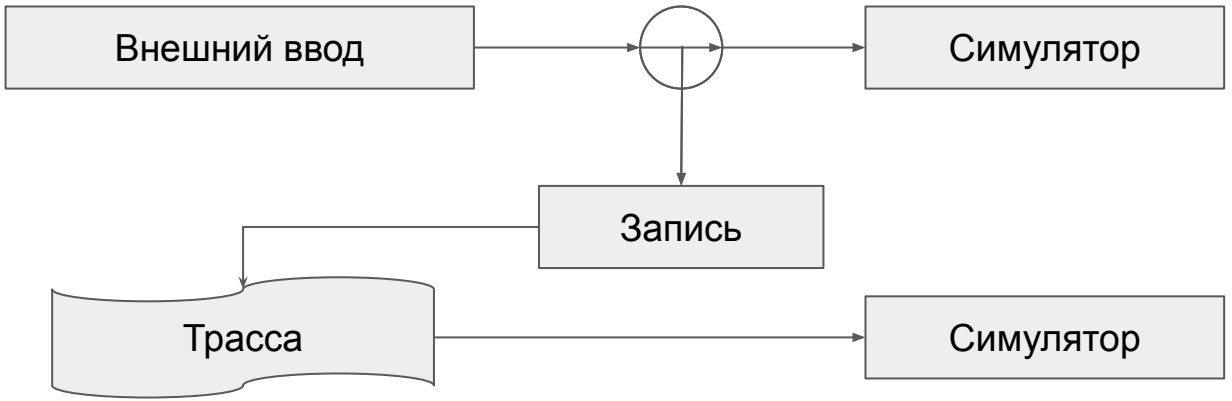
- Квиз
- Моделирование с использованием трасс
- **Применение трасс**
- Практическая часть
- Домашнее задание №4

Применение трасс

- Рассмотрим области применения симуляции на основе трасс

Применение трасс

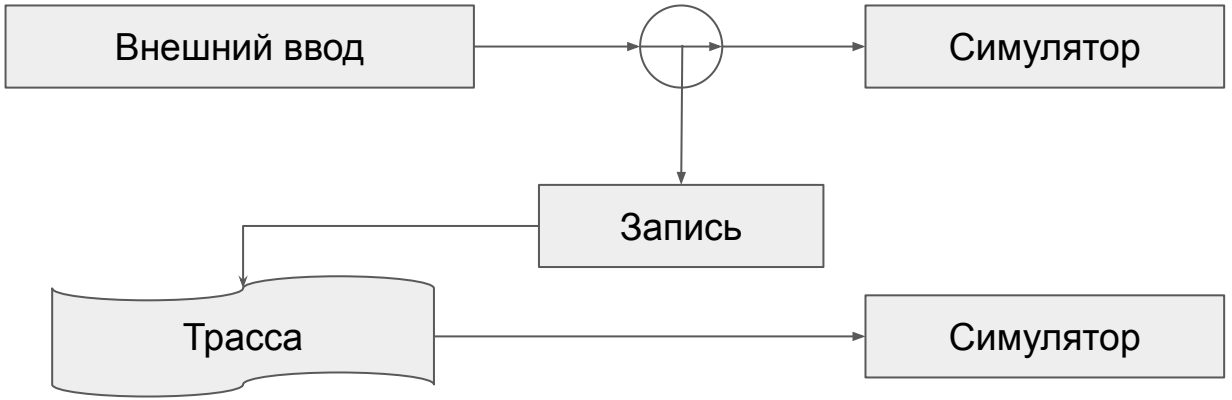
- Рассмотрим области применения симуляции на основе трасс
  - ➢ Детерминистический ввод



Этапы сбора и использования трассы

Применение трасс

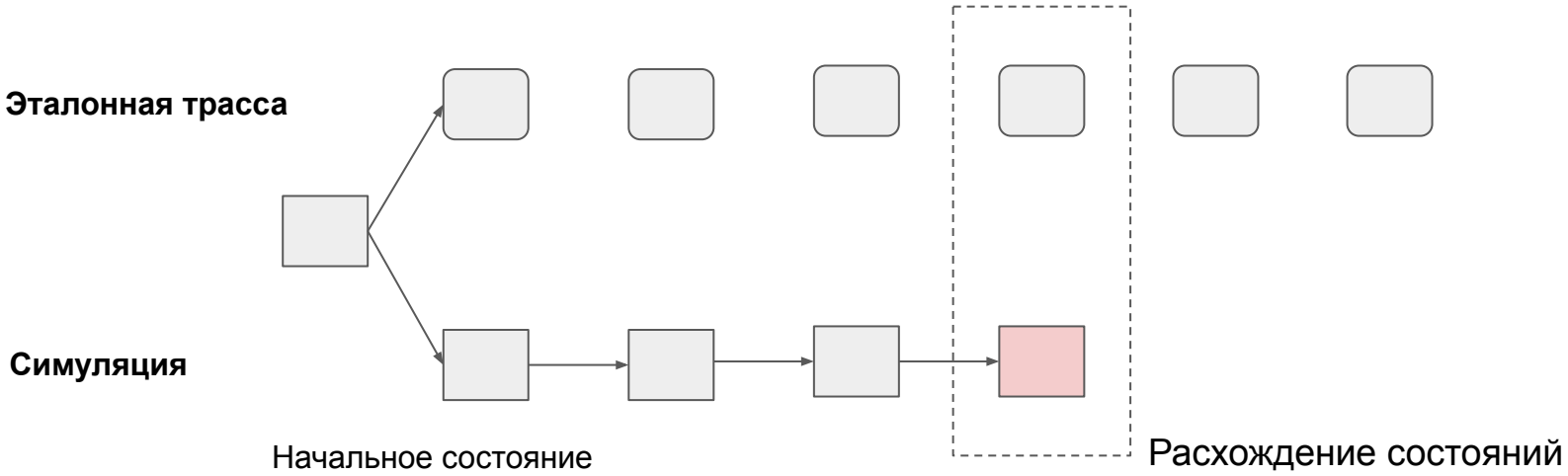
- Рассмотрим области применения симуляции на основе трасс
  - ➢ Детерминистический ввод
    - Ввод пользователя с использованием клавиатуры
    - Сетевое взаимодействие



Этапы сбора и использования трассы

Применение трасс

- Рассмотрим области применения симуляции на основе трасс
  - ➢ Валидация симулятора



Верификация модели с помощью эталонной трассы на каждом шаге симуляции

Применение трасс

- Моделировать одно и то же явление, получая один и тот же результат, сомнительная идея

Применение трасс

- Моделировать одно и то же явление, получая один и тот же результат, сомнительная идея

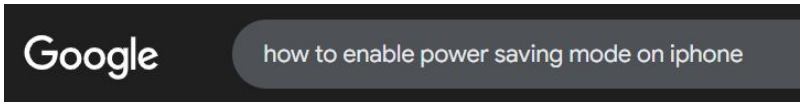
➢ **Предложение**: менять характеристики модели, а трассу оставлять неизменной

Применение трасс

- Моделировать одно и то же явление, получая один и тот же результат, сомнительная идея
- Порядок и структура событий в трассе будут одинаковыми при каждом запуске симулятора
- Нет необходимости много раз запускать модель для сборки трассы

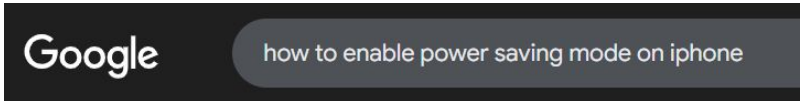
Применение трасс

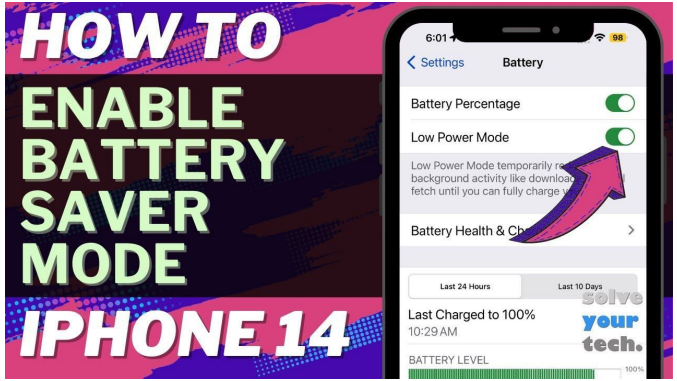
- **Пример №1:** оптимизация потребления электроэнергии.



Применение трасс

- **Пример №1:** оптимизация потребления электроэнергии.
  1. Допустим, хотим изменить модель потребления электроэнергии системы.
  2. История доступов в память не изменяется ⇒ обращения в память можно сохранить в трассу
  3. Записанную трассу используем для прогонов на модифицированной модели для получения новых значений





Применение трасс

- **Пример №2:** изучение поведения программы на процессоре с новой микроархитектурой.

Применение трасс

- **Пример №2:** изучение поведения программы на процессоре с новой микроархитектурой.
- Важно, что оба процессора совместимы на уровне машинных команд

Применение трасс

- **Пример №2:** изучение поведения программы на процессоре с новой микроархитектурой.
  - ➢ Трасса собирается на существующем процессоре
    - Все доступы в регистр с результатами отдельных чтений/записей
    - Все внешние и внутренние события (прерывания, исключения)

Применение трасс

- **Пример №2:** изучение поведения программы на процессоре с новой микроархитектурой.
  - Трасса собирается на существующем процессоре
  - ➢ Трасса подаются на модель
    - Результаты используются как история взаимодействия с внешним миром
    - Изменяется внутреннее состояние соответственно трассе, сообщая возникающие задержки

Применение трасс

- **Пример №2:** изучение поведения программы на процессоре с новой микроархитектурой.
  - Трасса собирается на существующем процессоре
  - Трасса подаются на модель
  - ➢ Анализ полученной статистики
    - Нет необходимости писать точную модель оптимизированной аппаратуры
    - Достаточно обладать упрощенной системой задержек

Применение трасс

- **Пример №2:** изучение поведения программы на процессоре с новой микроархитектурой.
  - Трасса собирается на существующем процессоре
  - Трасса подаются на модель
  - Анализ полученной статистики
- При таком подходе нет необходимости иметь доступ ни к исходному коду, ни к исполняемым файлам изучаемого приложения

Применение трасс

- **Пример №2:** изучение поведения программы на процессоре с новой микроархитектурой.
  - Трасса собирается на существующем процессоре
  - Трасса подаются на модель
  - Анализ полученной статистики
- При таком подходе нет необходимости иметь доступ ни к исходному коду, ни к исполняемым файлам изучаемого приложения
- Удобно изучать закрытые или ограниченные в распространении приложения, работая лишь с трассой

- Квиз
- Моделирование с использованием трасс
- Применение трасс
- ➢ **Практическая часть**
- Домашнее задание №4

От слов – к делу

- ChampSim — симулятор с открытым исходным кодом, основанный на трассировке, поддерживаемый Техасским университетом A&M

От слов – к делу

- ChampSim — симулятор с открытым исходным кодом, основанный на трассировке, поддерживаемый Техасским университетом A&M
- Первоначально ChampSim был разработан как платформа для соревнований по микроархитектуре (DPC3, DPC2, CRC2, IPC1 и т. д.)

От слов – к делу

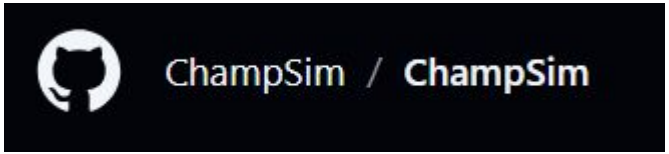
- ChampSim — симулятор с открытым исходным кодом, основанный на трассировке, поддерживаемый Техасским университетом A&M
- Первоначально ChampSim был разработан как платформа для соревнований по микроархитектуре (DPC3, DPC2, CRC2, IPC1 и т. д.)
- Активно используется для разработки множества современных политик замены кэша и префетч-политик

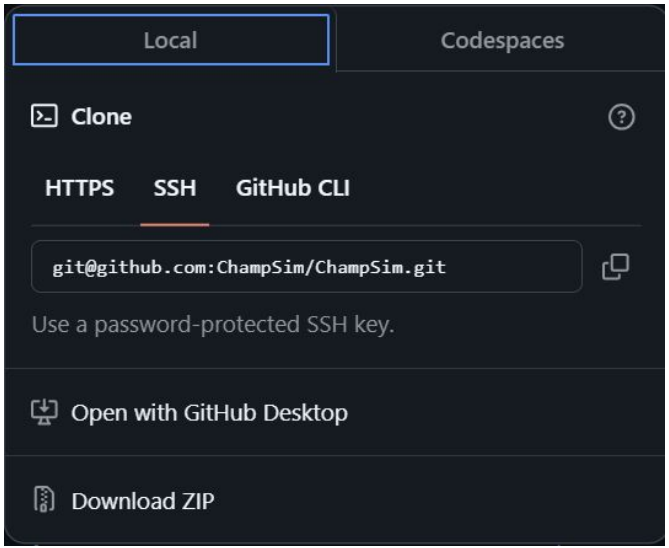
От слов – к делу

- Почему именно ChampSim?
  - Быстрый trace-based симулятор
  - Можно приступить проводить исследования без глубокого погружения в микроархитектуру
  - Модульность
  - Разнообразие сред моделирования

От слов – к делу

- Внимательно читаем README и не делаем ничего лишнего:
  - ➢ Клонируем проект
    - Устанавливаем сторонние зависимости
    - Собираем проект





От слов – к делу

- Внимательно читаем README и не делаем ничего лишнего:
  - Клонируем проект
  - ➢ Устанавливаем сторонние зависимости
  - Собираем проект

```
git submodule update --init
vcpkg/bootstrap-vcpkg.sh
vcpkg/vcpkg install
```

От слов – к делу

- Внимательно читаем README и не делаем ничего лишнего:
  - Клонируем проект
  - Устанавливаем сторонние зависимости
  - ➢ Собираем проект

```
$ ./config.sh <configuration file>
$ make
```

От слов – к делу

- Внимательно читаем README и не делаем ничего лишнего:
  - Клонируем проект
  - Устанавливаем сторонние зависимости
  - Собираем проект
- ChampSim при сборке требует конфигурационный файл.
- Все параметры в файле являются опциональными

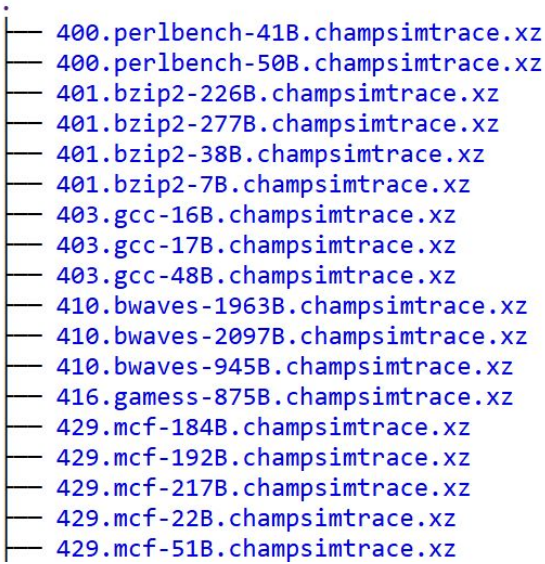
```
{
    "executable_name": "champsim",
    "block_size": 64,
    "page_size": 4096,
    "heartbeat_frequency": 10000000,
    "num_cores": 1,

    "ooo_cpu": [
        {
            "frequency": 4000,
            "ifetch_buffer_size":64,
            "decode_buffer_size":32,
            "dispatch_buffer_size":32,
            "rob_size": 352,
            "lq_size": 128,
            "sq_size": 72,
            "fetch_width": 6,
            "decode_width": 6,
            "dispatch_width": 6,
            "execute_width": 4,
            "lq_width": 2,
            "sq_width": 2,
            "retire_width": 5,
            "mispredict_penalty": 1,
            "scheduler_size": 128,
            "decode_latency": 1,
            "dispatch_latency": 1,
            "schedule_latency": 0,
            "execute_latency": 0,
            "branch_predictor": "bimodal",
            "btb": "basic_btb"
        }
    ],
```

Конфигурационный файл champs config.json

От слов – к делу

- ✓ Внимательно читаем README и не делаем ничего лишнего:
- Затем переходим в хранилище и скачиваем трассы с бенчмарков SPEC CPU 2017
  - Например, скачаем самую первую, `400.perlbench-41B.champsimtrace.xz`



Хранилище трасс бенчмарков SPEC CPU 2017

От слов – к делу

- ✓ Внимательно читаем README и не делаем ничего лишнего:
- ✓ Затем переходим в хранилище и скачиваем трассы с бенчмарков SPEC CPU 2017
- Тут вы должны задаться вопросом…

Дополнение к примеру №2

- Говоря о закрытых приложениях и “обезличенных” трассах, мы должны понимать, что:
  - В трассе отражены лишь внешние события
  - Изменения во внутреннем состоянии отслеживаются моделью

Дополнение к примеру №2

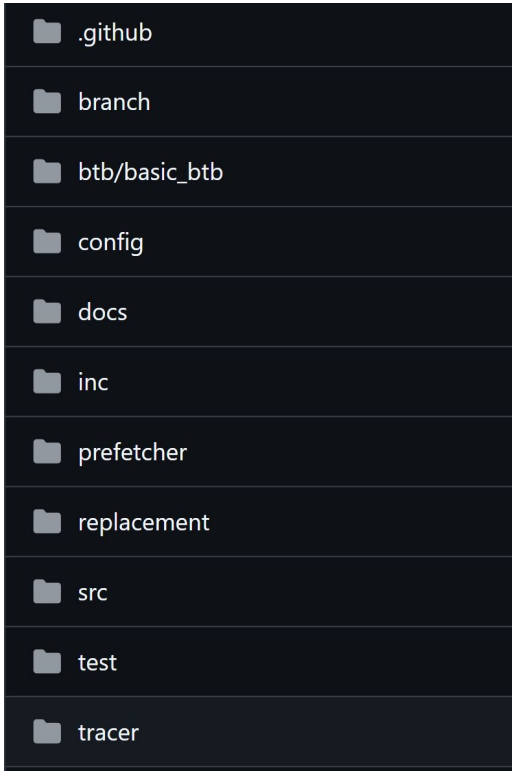
- Говоря о закрытых приложениях и “обезличенных” трассах, мы должны понимать, что:
  - В трассе отражены лишь внешние события
  - Изменения во внутреннем состоянии отслеживаются моделью
- Рассмотрим более конкретный анализ – изучение производительности кэшей.
  - Трасса содержит информацию о последовательности, типах, адресах доступов
  - Количество линий, емкость и прочие микроархитектурные аспекты определяются моделью

От слов – к делу

- Вспоминаем про модульность и простоту конфигурации ChampSim.

От слов – к делу

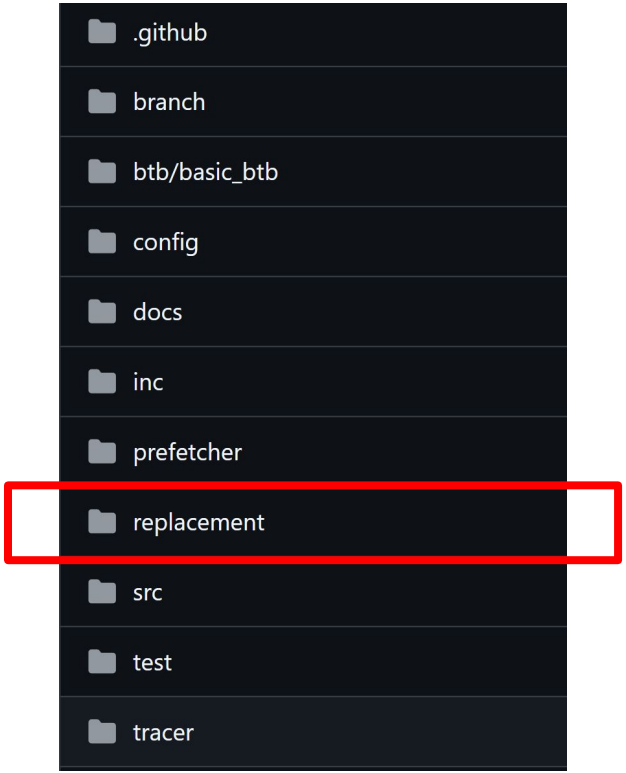
- Вспоминаем про модульность и простоту конфигурации ChampSim
- Смотрим на структуру проекта и видим:
  - Branch Predictors
  - Branch Target Buffers
  - Memory Prefetchers
  - Replacement Policies



Структура проекта ChampSim

От слов – к делу

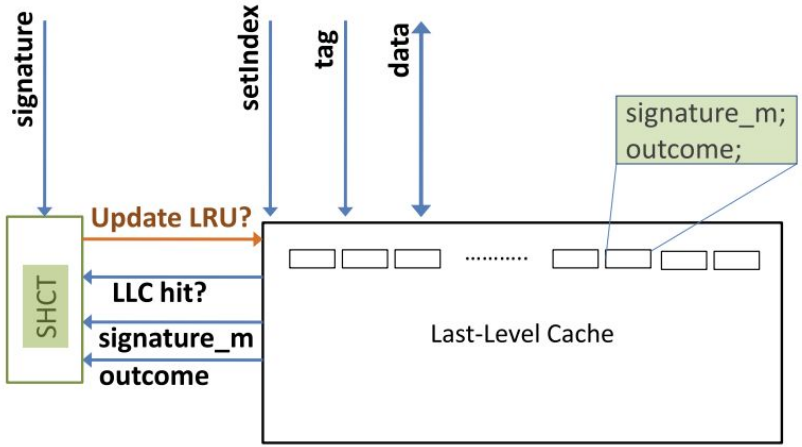
- Вспоминаем про модульность и простоту конфигурации ChampSim
- Смотрим на структуру проекта и видим:
  - Branch Predictors
  - Branch Target Buffers
  - Memory Prefetchers
  - **Replacement Policies**



Структура проекта ChampSim

Встроенные политики замещения

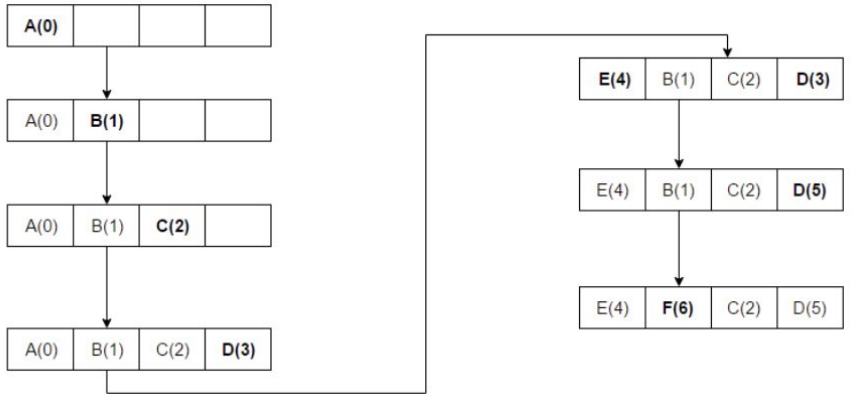
- На мастере нам сразу доступны четыре политики
  - LRU – Least Recently Used
  - SRRIP – Static Re-Reference Interval Prediction
  - DRRIP – Dynamic Re-Reference Interval Prediction
  - SHIP – Signature-Based Hit Prediction



Архитектура SHiP

Встроенные политики замещения

- На мастере нам сразу доступны четыре политики
  - **LRU – Least Recently Used**
  - SRRIP – Static Re-Reference Interval Prediction
  - DRRIP – Dynamic Re-Reference Interval Prediction
  - SHIP – Signature-Based Hit Prediction



LRU

Объявление новой политики замещения

- Модуль политики замещения кэшей должен определять четыре функции:
  - `initialize_replacement();`
  - `find_victim(...);`
  - `update_replacement_state(...);`
  - `replacement_final_stats()`

Объявление новой политики замещения

- Модуль политики замещения кэшей должен определять четыре функции:
  - `void CACHE::initialize_replacement();`
  - `uint32_t CACHE::find_victim(...);`
  - `void CACHE::update_replacement_state(...);`
  - `void CACHE::replacement_final_stats();`
- Подробнее можно ознакомиться в документации ChampSim.

От слов – к делу.

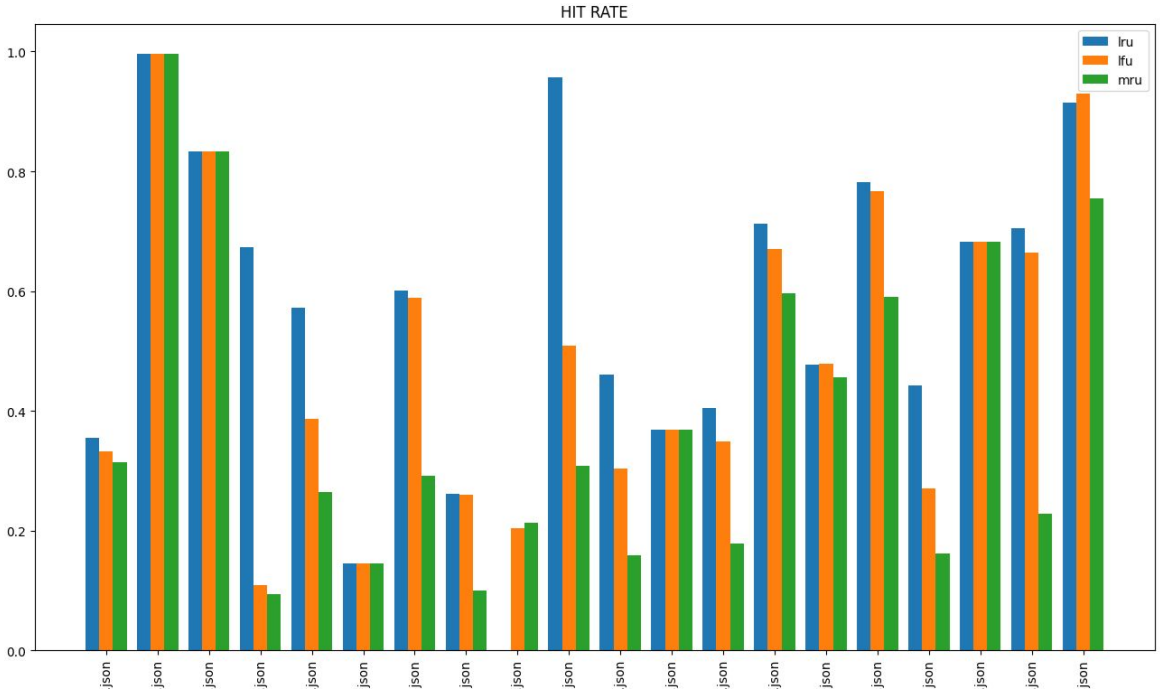
- В дополнение к LRU определим еще две простые политики замещения:
  - LFU – Least Frequently Used
  - MRU – Most Recently Used

От слов – к делу.

- В дополнение к LRU определим еще две простые политики замещения:
  - LFU – Least Frequently Used
  - MRU – Most Recently Used
- Давайте проведем небольшую исследовательскую работу и изучим пространство конфигураций модели ChampSim

Анализ результатов

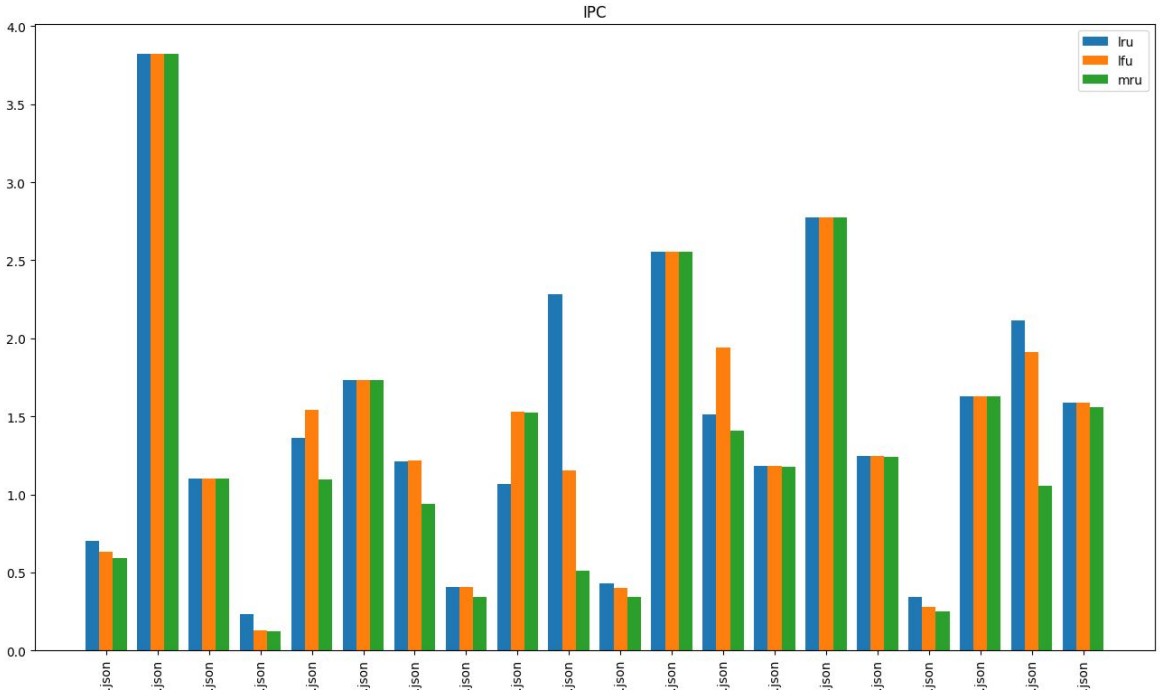
- Из выходного JSON файла с результатами подсчитываем HIT RATE по L2C



HIT RATE трех разных политик на наборе трасс

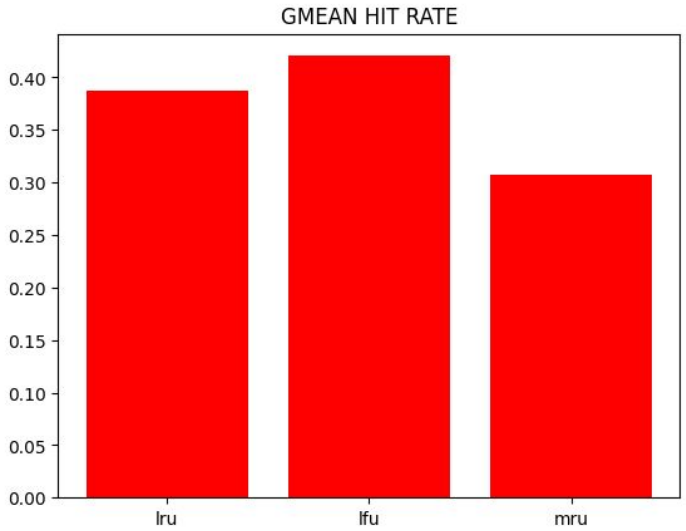
Анализ результатов

- Из выходного JSON файла с результатами собираем статистику IPC

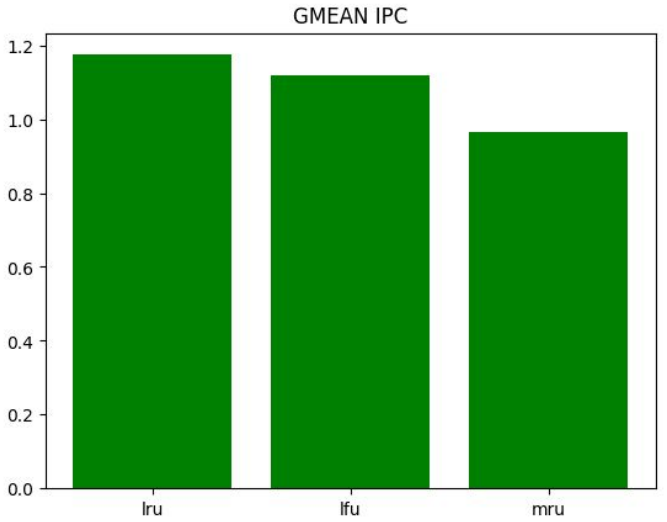


IPC трех разных политик на наборе трасс

Анализ результатов



Среднее геометрическое HIT RATE для трех разных политик

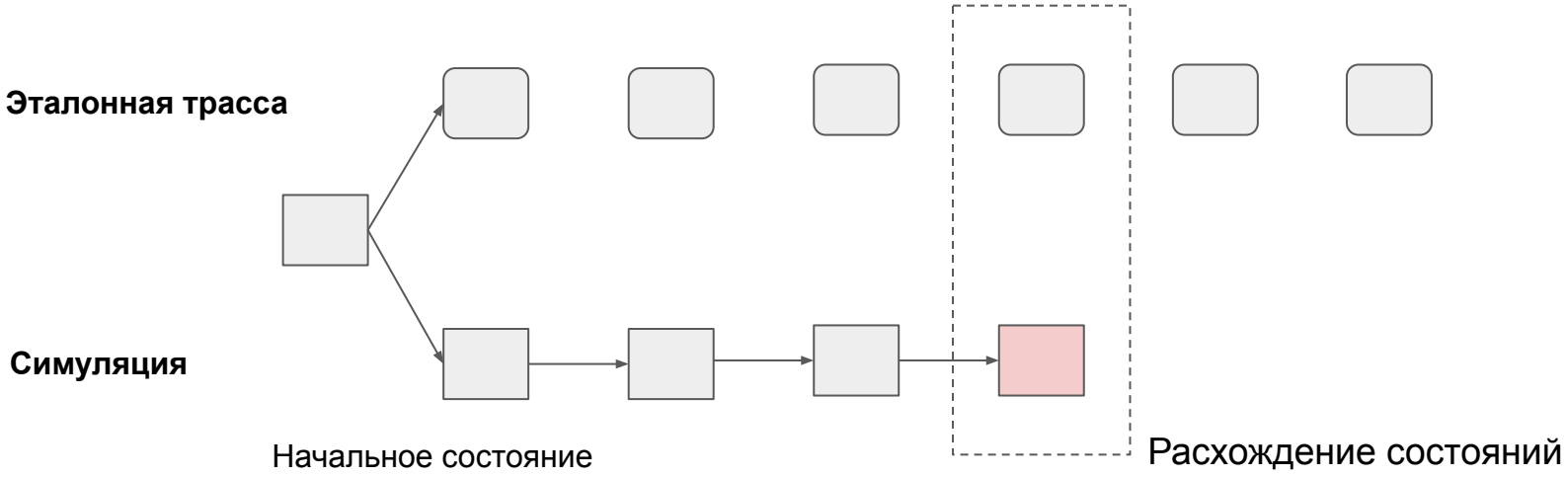


Среднее геометрическое IPC для трех разных политик

- Квиз
- Моделирование с использованием трасс
- Применение трасс
- Практическая часть
- **Домашнее задание №4**

Домашнее задание №4

1. Реализуйте в вашем симуляторе модуль, трассирующий исполнения модели.
2. Реализуйте косимуляционный скрипт, позволяющий проводить валидацию симулятора с эталонной моделью



Литература

1. Precise and Accurate Processor Simulation
2. The Championship Simulator: Architectural Simulation for Education and Competition
3. SHiP: Signature-based Hit Predictor for High Performance Caching