МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

имени М.В. Ломоносова

Факультет вычислительной математики и кибернетики

Отчет по учебному курсу

"Имитационное моделирование в исследовании и разработке распределенных информационных систем "

Отчет

студентки 321 группы

факультета ВМК МГУ

Задорожной Юлии Андреевны

2021

**1. Наблюдение за работой ВС и измерение характеристик производительности**

1.1. Организован сбор трассы адресов выполнения программы для последующего использования этой трассы в качестве рабочей нагрузки для имитационной модели подсистемы подкачки страниц виртуальной памяти. Как можно сократить объём трассы без ущерба для правильности работы модели?

Ответ: Смещение страниц происходит относительно реальной памяти, поэтому для уменьшения объёма трассы можно хранить лишь номера страниц для их подкачки, что не нарушит правильность работы модели.

1.2. Вопрос аналогичный 1.1, но собирается трасса кадров Ethernet на портах маршрутизатора (уровня L3) для последующей подачи на вход модели, рассчитывающей длины очередей на портах маршрутизатора. Какие части пакета следует записывать?

Ответ: Для модели, рассчитывающей длины очередей, важно определить лишь то, на какой порт и в какую очередь отправить очередной кадр. Следовательно, в кадре можно оставить лишь адреса получателя и отправителя, а также длину самого переданного кадра.

1.3. Предложите сценарий измерения времени на подкачку страницы виртуальной памяти в вычислительной системе под управлением ОС Linux.

Ответ: Во-первых, для подобной цели можно использовать какой-либо профилировщик (например, perf). Они позволяют отслеживать, какое время затрачивается внутри тех или иных функций, в том числе вызванных ядром. Можно таким образом отслеживать функцию подкачки страницы виртуальной памяти, вызываемой ядром, и засечь нужное нам время.

Во-вторых, чтобы гарантировать подкачку страниц, необходимо написать программу, где мы используем одну лишь функцию 𝑎𝑙𝑙𝑜𝑐\_𝑝𝑎𝑔𝑒(), которая будет выделять память, превышающую объем ОЗУ ВС на несколько виртуальных размеров страницы, и заполнить весь этот объем произвольными данными. В конце мы обращаемся к странице, находящейся в файле подкачки (а нужная страница точно будет там, потому что объем ОЗУ ВС заполнен последними виртуальными страницами из выделенного блока), то тогда подкачка произошла (т.к. нужной страницы нет в ОЗУ) и можно начать измерение времени доступа.

1.4. Предложите сценарий измерения времени запуска файла на выполнение в ОС Linux.

Ответ: Можно использовать системный вызов time(). Пусть файл называется file, при этом ему на вход подают параметры argc. Тогда выполнение утилиты time file argc даст нам информацию о времени, затраченном на запуск файла от начала до конца (real), о процессорном времени в режиме пользователя (user) и в режиме ядра (sys).

Также можно добавить в самое начало файла команду вывода текущего времени с помощью написанного bash-скрипта, зафиксировав предварительно время начала выполнения программы, используя далее функцию clock() узнать время начала и конца работы программы, после чего вычесть первое из последнего, но этот процесс может быть несколько проблематичным в случае сложно редактируемого файла (например, бинарного файла) – для этого придется заглянуть в ассемблерный код, поэтому лучше использовать первый вариант.

**2. Понятие модели, понятие производительности**

2.1. Приведите пример модели (любого вида), адекватной при одних условиях и неадекватной при других. Укажите также эти условия адекватности. Примеры моделей, приведённые на лекциях, в счёт не идут.

Ответ: Приведем в пример модель математического маятника. Пусть отклонение от положения равновесия 𝑥 описывается физической формулой:

𝑥=𝐴∗sin(𝜔𝑡)

где 𝐴− амплитуда колебаний, 𝜔−циклическая частота колебаний, 𝑡− время.

При достаточно малом времени наблюдения 𝑡 и при достаточно малых отклонениях 𝑥 данная модель хорошо описывает поведение маятника, является адекватной, но реальный маятник со временем будет отклоняться всё меньше и меньше (затухающие колебания), а затем и вовсе остановится. Следовательно, при больших 𝑡 данная модель не отражает поведение маятника и является неадекватной.

2.2. Приведите примеры вычислительной системы a) для которой пропускная способность важнее, чем время отклика; б) для которой время отклика важнее пропускной способности.

Ответ: В данном случае **пропускная способность** – **это** максимально допустимая скорость обработки трафика, которая определяется стандартами **сети**. Она показывает, какой максимальный объем может быть передан в единицу времени.

Понятно, что пропускная способность будет важнее для систем, обрабатывающей большое количество данных, а время отклика – для систем реального времени.

а) сетевой процессор

б) вычислительная система, связанная с управлением атомным реактором.

2.3. Сформулируйте определения показателей «пропускная способность» и «время отклика» для сетевого коммутатора L2.

Ответ:

**Пропускная способность коммутатора** - количество данных, переданных через порты коммутатора в единицу времени. Пропускная способность коммутатора измеряется пропускной способностью портов на «вход» и «выход», а они в свою очередь измеряются производительностью передачи информации через себя.

**Время отклика** - время между началом записи кадра в буфер входного порта и появлением этого кадра на выходном порту (то есть получением первого байта ответа).

2.4. При выполнении некоторой программы на вашем ПК его производительность вас не удовлетворяет. Предложите сценарий определения того, какую часть аппаратуры нужно обновить – поставить более быстрый процессор или больший объём памяти?

Ответ: Сначала необходимо проверить ПК на наличие вредоносных программ, а также удалить все ненужные программы. Далее необходимо проверить загруженность оперативной памяти при работе на ПК. Более того, при выполнении программы можно посмотреть на данные мониторинга процессов (какую часть ЦП и памяти занимает программа, к примеру, через диспетчер задач) и определить, какую аппаратуру она больше нагружает (затрачивает больший объем относительно стандартных имеющихся ресурсов): процессор или память. Ту аппаратуру, которая не справляется с нагрузкой созданных программой процессов и необходимо заменить.

**3. Аналитические модели производительности**

3.1. Пусть для предсказания времени выполнения программы используются эталоны (бенчмарки). Дайте математическую формулировку задачи предсказания времени выполнения.

Ответ: Предположим, что нам известно время выполнения программы (𝑇0) на некоторой другой системе, известны оценки бенчмарков нашей системы (𝑆) и другой системы (𝑆0). В таком случае необходимо найти время выполнения программы (𝑇) на нашей системе, это

можно сделать по формуле:

𝑇=𝑇1∗(𝑆1/𝑆0)

3.2. Программа выполняется на некотором вычислителе 10 единиц времени. Известно, что 20% вычислений программы должно быть выполнено строго последовательно, а остальные 80% могут быть распараллелены на 10 ветвей одинаковой длины. За какое минимальное время программа может быть выполнена на системе из 5 аналогичных вычислителей? Временем передачи данных между вычислителями пренебречь.

Ответ: На последовательную часть программы уйдёт 2 единицы времени (те самые 20 процентов) – это никаким образом это ускорить не получится. Остальные вычисления на 8 секунд можно распараллелить на 5 вычислителей, что даст нам 1,6 секунды. Следовательно, общее

время выполнения – 3,6 секунды.

10 \* 0.2 + 10 \* 0.8 / 5 = 3.6

3.3. Приведите пример аналитической модели производительности из курса «Введение в компьютерные сети».

Ответ: Аналитическая модель производительности - это функция характеристики производительности от параметров системы и рабочей нагрузки. Примером может послужить теорема Найквиста-Котельникова, связывающая эти характеристики:

Rmax \_𝑑𝑎𝑡𝑎\_𝑟𝑎𝑡𝑒=2𝐷∗log2𝐿,

где 𝐷− ширина полосы пропускания канала, L – количество значений сигнала, Rmax \_𝑑𝑎𝑡𝑎\_𝑟𝑎𝑡𝑒 – максимально возможная пропускная способность канала.

**4. Средства имитационного моделирования**

4.1. Какие возможности языков и библиотек имитационного моделирования (по материалам курса) используются для представления структуры моделируемой системы?

Ответ:

Язык NED для описания структуры:

* Графическое редактирование структуры
* Доступ к структуре модели из компонента
* Возможность построения массивов компонентов и массивов портов;
* Доступ к порту по имени, по номеру, по ссылке

OMNet++ :

* Конфигурационные файлы;
* Графический интерфейс для прогона модели;
* Возможность анимации состояния компонентов;
* Трассировка

4.2. Какие возможности языков и библиотек имитационного моделирования (по материалам курса) используются для описания поведения моделируемой системы? Приведите как можно больше примеров (в рамках материалов курса, разумеется).

Ответ:

* Массивы не только из чисел, но и из моделей (например, массивы резисторов).
* Сменные подмодели
* Языковые элементы для удобного определения разрывных и переменных   
  структур систем
* Математические функции с переменным числом входных / выходных   
  аргументов
* Удобный вызов функций C, Fortran и Java.
* Мощная концепция библиотеки (автоматический поиск модели в файловой системе, обработка версий, преобразование между версиями и т. д.).
* Средства планирования событий

4.3. Какие возможности системы моделирования OMNet++ ориентированы на наблюдение за работой модели, сбор и обработку результатов эксперимента?

Ответ:

* Наличие collection под названием statistic, которая позволяет с помощью макроса WATCH использовать сигналы
* графический интерфейс прогона модели
* анимации работы компонентов ориентированы на наблюдение за работой модели
* анализ данных в различных средах Excel, R, Python
* генерация отчета

4.4. Какие возможности системы моделирования OMNet++ ориентированы на многократное проведение экспериментов?

Ответ:

Возможность изменения указанных параметров при каждом запуске симуляции без необходимости перестраивать систему снова (например, создание массива компонентов или портов и указание его размера (как параметр) при каждом запуске)

4.5.Какие возможности системы моделирования OMNet++ ориентированы на моделирование компьютерных сетей?

Ответ:

**OMNeT++** (**O**bjective **M**odular **N**etwork **T**estbed in C**++**) Discrete Event Simulator – это модульная, [компонентно‑ориентированная](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D0%BE%D0%BD%D0%B5%D0%BD%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5?stable=1) C++ библиотека и фреймворк для **дискретно‑событийного** [**моделирования**](https://habr.com/post/415707/), используемая прежде всего для создания **симуляторов сетей**. Попросту говоря это “симулятор дискретных событий”, включающий: IDE для создания моделей, и сам симулятор (GUI). В частности, **INET** Framework – [“библиотека” сетевых моделей](https://inet.omnetpp.org/Introduction.html) для OMNeT++.

**5. Прочее**

5.1. Сравните подходы статической и динамической двоичной трансляции в эмуляторах CPU. Укажите достоинства и недостатки.

Ответ: Статическая трансляция предполагает перевод всего кода исполняемого файла в код, выполняемый в целевой архитектуре, а динамическая же разделяет этот код на отдельные части, переводит их отдельно и кэширует данные. **Основные преимущества** **статической трансляции**: отсутствие медленного старта, небольшие накладные расходы при выполнении коротких программ, отсутствие временных ограничений.**Недостатками** будут являться невозможность анализа кода (неизвестно, где находится команда, а где данные), задержка запуска больших программ.

**Преимущества динамической трансляции** состоят в возможности адаптивной оптимизации, отсутствии задержки при запуске больших программ, возможности анализа кода (другими словами – прозрачность оптимизируемой программы). **Её недостатками** являются медленный старт, невыгодность выполнения маленьких программ, временные ограничения (время выполнения должно быть меньше выигрыша, получаемого вследствие этих оптимизаций). Более того, динамическая двоичная трансляция отличается от простой эмуляции удалением основного цикла эмулятора «считывание-декодирование-исполнение» (что является основным его узким местом), расплачиваясь большими накладными нагрузками во время процесса трансляции.

5.2. Какие возможности языка моделирования аппаратных средств (на примере рассмотренного на лекции Verilog) ориентированы на специфику предметной области?

Ответ:

Verilog используется на уровне вентилей и уровне регистровых передач

Учёт специфики предметной области:

• Описание структуры системы:

– модули;

– входные и выходные порты;

– соединения

• Описание поведения системы:

– сигналы;

– время;

– события;

– операции;

– параллелизм

5.3. Какие компоненты, по-вашему, должны входить в развитую и удобную для применения систему имитационного моделирования?

Ответ:

1. Наличие библиотечных примеров из основной области применения для быстрой сборки модели;
2. Наличие дебаггера для полаговой отладки и готовый для быстрой установки IDE загрузчик
3. Удобное и приятное визуальное представление работы модели;
4. Возможность осуществить взаимодействие модели с внешними системами;
5. User-friendly интерфейс, интуитивная его понятность;
6. Наличие понятной документации, удобной и простой системы обучения;
7. Возможность собирать и обрабатывать результаты эксперимента;
8. Возможность многократного проведения эксперимента;
9. Удобная интегрированная среда разработки.

5.4. Описание системы на языке описания аппаратных средств может использоваться для следующих целей: …. Приведите перечень целей.

Ответ:

1. Выявление недостатков и ошибок в системе;
2. Отражение характеристик требуемой системы и примерная оценка стоимости ресурсов;
3. Поиск оптимальных параметров требуемой системы;
4. Автоматизация тестирования проектируемой системы;
5. Подготовка документации;
6. Удешевление и ускорение работы

Язык описания аппаратуры (HDL) может использоваться для описания структуры и поведения электронных схем, чаще всего применяется для цифровых логических синтезов схем. Описание аппаратуры может производиться на уровнях потоков данных (dataflow), структуры, поведения.