Подсистема наблюдения, отладки и управления параллельным счетом для объектно-ориентированной системы программирования

Чугунов Арсений

научный руководитель: Илюшин Александр Иванович

Цель работы

Создать систему наблюдения, отладки и управления для системы OST со следующими возможностями:

- Отображение мнемосхемы прикладной модели (объекты, связи, процессоры)
- Мониторинг прикладных объектов в процессе счета (значения существенных переменных, локальное время, процессорное время и т.п.)
- Управление счетом прикладной модели (запуск, приостановка/возобновление счета, вывод данных для оперативного анализа, изменение приоритетов для объектов и т.д.)

Инструментальные средства

- В качестве языка программирования используется язык Python
- Для подключения к системе OST используется библиотека PYRO
- Для выбора графической библиотеки были на практике опробованы библиотеки:
  - Pygame
  - Pyglet
  - VPython
  - WxWidgets
  - Qt

О выборе графической библиотеки

На первом этапе разработки программы выбор был сделан в пользу библиотек, ориентированных только на вывод графики (в том числе, через OpenGL), таких как Pygame и Pyglet.

Но в процессе работы стало ясно, что программе необходим удобный графический интерфейс. Поэтому в рассмотрение были включены библиотеки, ориентированные на создание графического интерфейса. При этом они должны были обладать методами вывода произвольной графики.

В итоге была выбрана библиотека Qt как наиболее универсальная.

Сравнение с другими системами

Существует множество различных систем отладки как текстовых, так и графических, которые поддерживают отладку параллельных программ. Они, в основном, универсальны в рамках тех языков программирования, на которые рассчитаны.

Основные возможности таких программ состоят в следующем:

- Возможность показа переменных по каждому потоку/процессу
- Показ загруженности процессоров
- Иногда, показ схемы потоков

Программа, разрабатываемая в данной работе отличается от других подобного типа тем, что ориентирована на отладку конкретной системы, поэтому учитывает только её особенности. Это позволяет сделать её несколько проще с точки зрения внутреннего устройства.

Перехват вызовов

Перехват вызовов функций можно осуществить с помощью декораторов.

Декораторы в языке Python – удобный способ изменения поведения функции (или целого класса). Его синтаксис выглядит так:

```
@f1
def f(x):
    …
```

что эквивалентно

```
def f(x):
    …
f = f1(f)
```

то есть имя функции f теперь соответствует f1(f), и при каждом вызове f(x) будет реально происходить вызов f1(f(x)).

С помощью декоратора можно отправлять программе мониторинга информацию о вызове и времени его прохождения.

Перехват значений переменных

Для перехвата значений переменных при их изменении можно использовать дескрипторы. Дескрипторы позволяют определить методы доступа к объекту.

```
class desc(object):
    def __get__(self, inst, cls=None):
        return self.x
    def __set__(self, inst, val):
        self.x = val

class A(object):
    x = desc()
```

Здесь атрибут x класса A – дескриптор. Его можно использовать как обычную переменную, но при этом для доступа будут использоваться функции __get__() и __set__().

Используя дескрипторы, можно отправлять значения переменных монитору при их изменении.

Отображение мнемосхемы

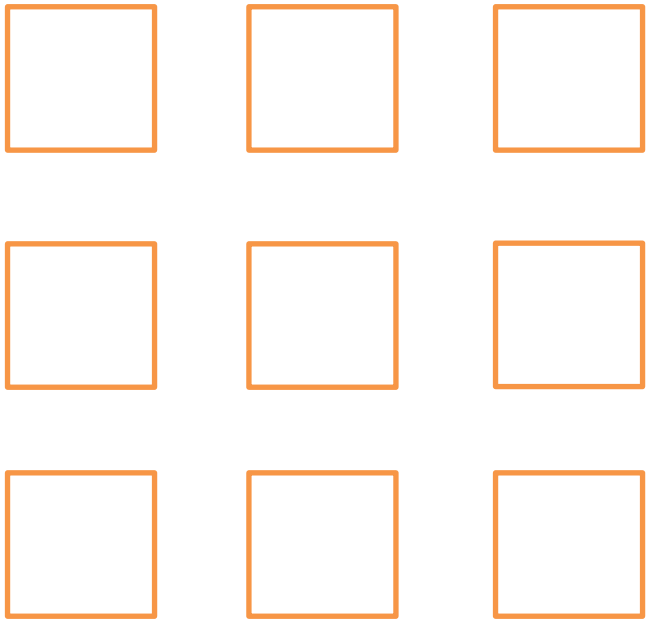
Отображение мнемосхемы задачи с точки зрения расстановки объектов по экрану можно рас-сматривать как задачу изображения произвольного графа.

К решению можно подойти двумя способами:

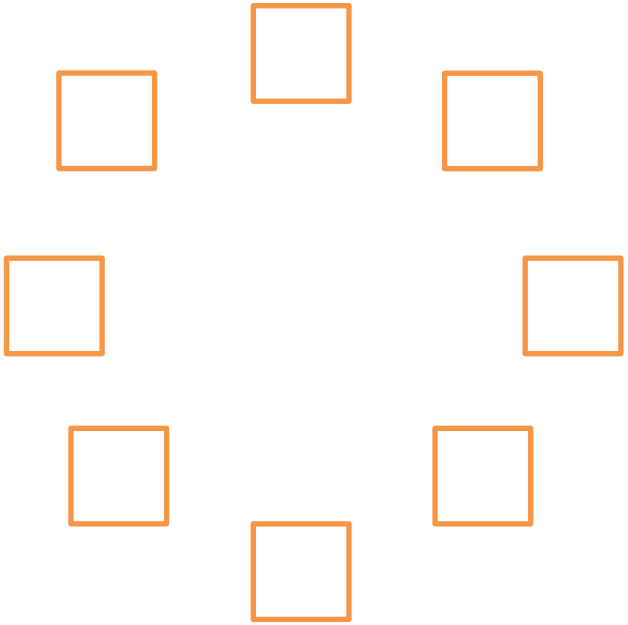
- Изображение графа по шаблону
- Использование приближённого силового метода (Force-based)

Изображение с помощью шаблона

По сетке

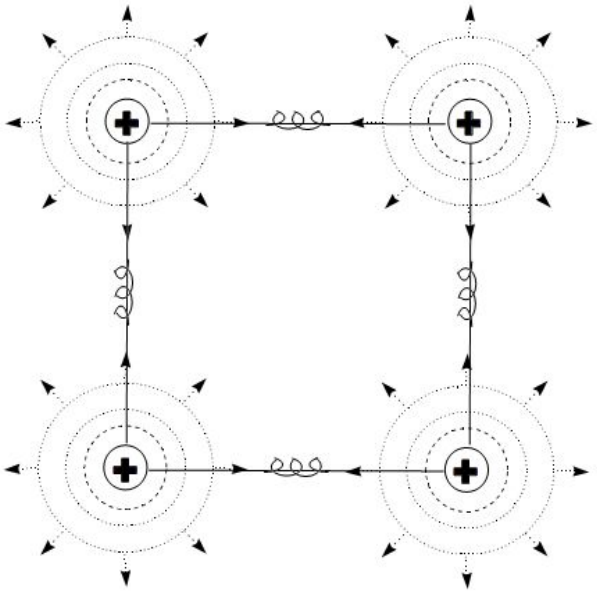


По окружности



Силовой метод

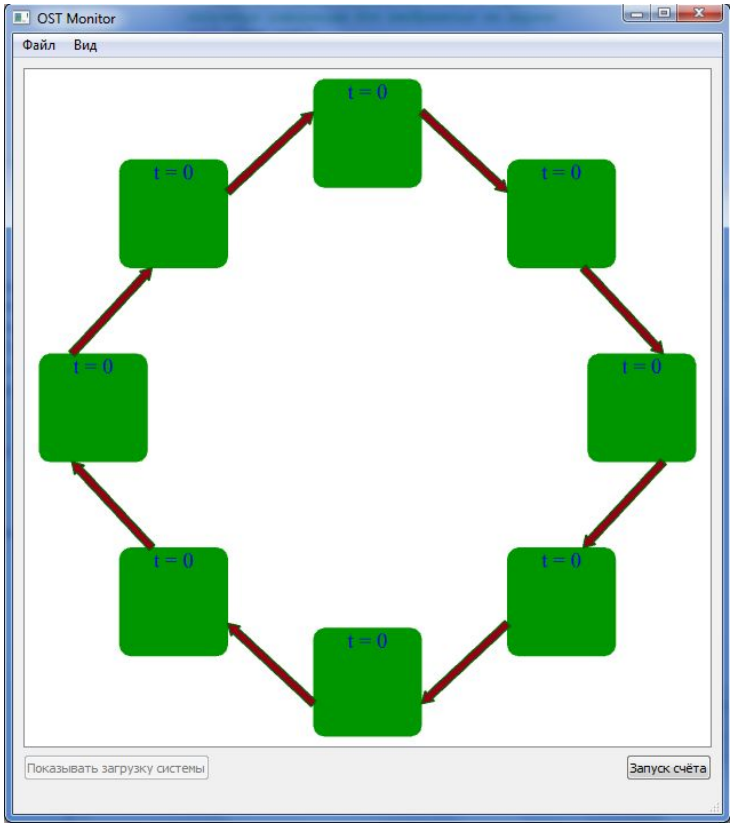
Это целый класс методов, суть которых сводится к рассмотрению графа как физической системы, где вершины – одинаков зараженные шарики, а рёбра – пружины.



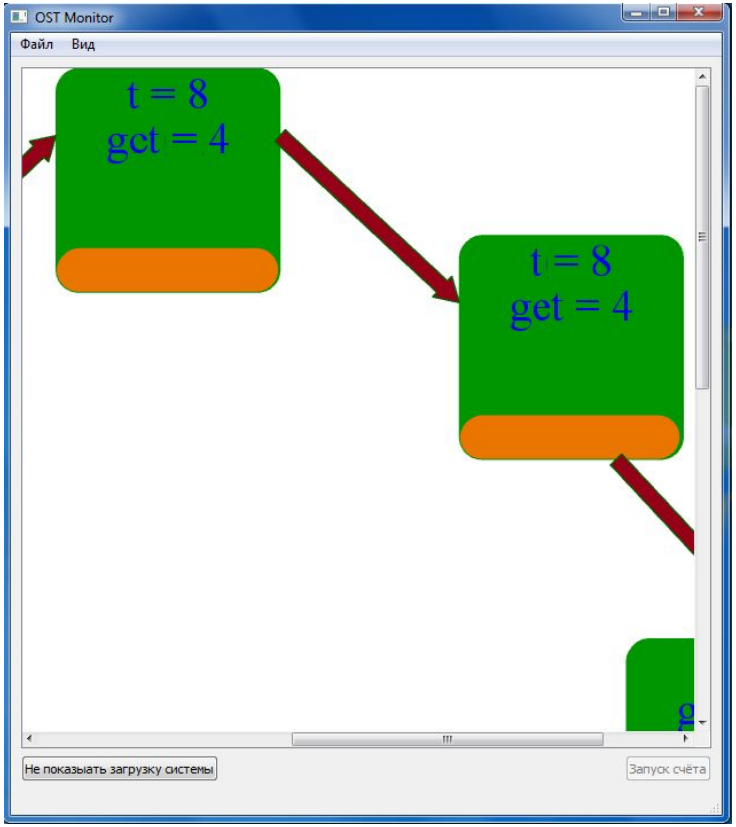
Положение равновесия системы – конечный результат работы алгоритма.
Причём законы взаимодействия не обязательно должны соответствовать законам физики.

Отображение мнемосхемы

Общий вид



Во время счёта

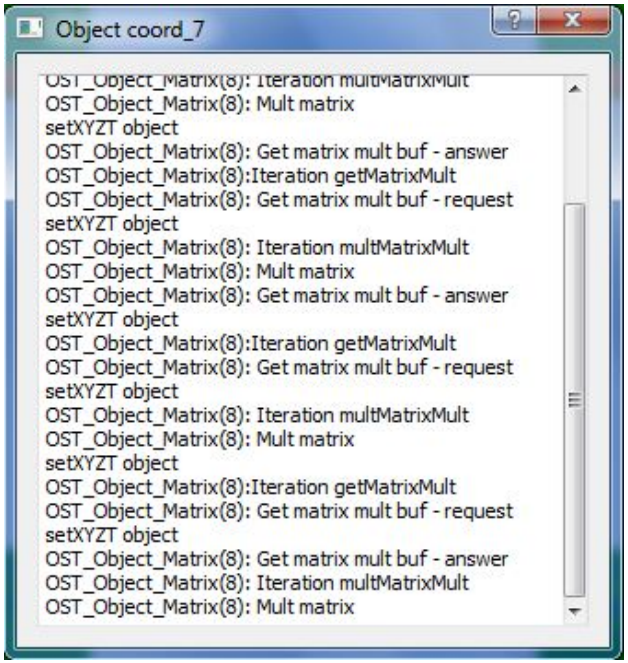


Отображение отладочной информации

Иногда недостаточно видеть схему счёта задачи для выявления всех проблем.

Бывает нужно иметь возможность получать текстовую отладочную информацию от объекта.

В программе такая возможность есть, причём выводить можно не только простой текст, но и html.



Количество кода

Программа занимает около 1150 строк кода. Из которых около 140 приходятся на декораторы, дескрипторы и функции, работающие на стороне системы OST отдельным модулем.

Число строк, необходимых для внесения в код системы OST:

- Около 7 служебных базовых строк
- По одной строке на каждую перехватываемую функцию
- По 2 строки на каждую перехватываемую переменную
- По одной строке на каждый вызов функции показа отладочной информации

Накладные расходы

Для умножения матриц:

Размер матрицы	потоки	Без мониторинга	С мониторингом
600X600	4	21,9с	22,35с (102%)
600X600	8	22с	22,65с (103%)
400X400	4	6,56с	6,83с (104,1%)
400X400	8	6,62с	7,14с (107,9%)
400X400	16	7с	8,3с (118,5%)

Заключение

В рамках данной работы была написана программа мониторинга системы OST со следующими возможностями:

- Подключение к системе OST, запуск счёта
- Отображение мнемосхемы задачи (объекты, связи)
- Перехват удалённых вызовов для их отображения, возможность измерения времени прохождения
- Перехват значений переменных для отображения локального времени и пользовательских переменных
- Отображение отладочной информации по каждому объекту
- Отображение уровня загрузки системы прикладными задачей