ЛР. Диспетчер корутин

Задание

В данной лабораторной работе необходимо реализовать программу (комплекс программ), осуществляющую вычисление алгоритма полезной нагрузки с использованием примитивов кооперативной многозадачности (корутин). Необходимо разработать диспетчер корутин, осуществляющий поддержку асинхронного запуска и выполнения корутин, передачу параметров.

Разрабатываемый диспетчер корутин основывается на исполнителе (пуле потоков) по варианту, и осуществляет планирование корутин в соответствии с алгоритмом по варианту. Диспетчер должен собирать статистику выполнения корутин и поддерживать ее в актуальном состоянии по мере исполнения программы. Статистика должна обновляться при передаче управления корутиной диспетчеру (любой вызов await, exit, yield). Периодически статистика должна передаваться во внешний мир.

Метрики, которые необходимо собирать:

1. Для каждой корутины
    1. Суммарное время пребывания в состоянии running
    2. Суммарное время пребывания в состоянии runnable
    3. Суммарное время пребывания в состоянии blocked
2. Для диспетчера в целом
    1. Количество корутин в каждом из состояний
    2. Суммарное время пребывания всех корутин в каждом из состояний
    3. Количество завершенных корутин
    4. Распределение время создания корутины

Реализуйте программу прикладного уровня, использующую ваш планировщик. Запустите ее, соберите ваши метрики, а также полученные средствами мониторинга Linux, визуализируйте их, проанализируйте.

Вы можете воспользоваться готовым шаблоном: https://github.com/vityaman-edu/coroed

Ограничения

Программа (комплекс программ) должна быть реализован на языке C/C++.

Разрешается реализовать как stackless, так и stackfull корутины (файберы).

Все циклы в корутинах, обрабатывающие какие-либо обособленные блоки данных, должны быть реализованы в виде генераторов (yield), или доказана их нецелесообразность, или невозможность использования для каждой конкретной ситуации.

Все IO обращения к ядру операционной системы из корутин должны считаться блокирующими и выполняться асинхронно с использованием соответствующих асинхронных функций-адаптеров (non-blocking IO), разрабатываемых самостоятельно.

Необходимо реализовать прикладную программу, исполнителя (executor) и алгоритм планирования корутин в соответствии с вашим вариантом.

Справочные материалы

- Шаблон проекта: Coroed
- Курс Concurrency Романа Липовского
- Онлайн курс про ОС от CSC