Эффективная реализация сопрограмм в управляемой среде исполнения

Евгений Пантелеев

Новосибирский государственный университет

Научный руководитель: Бульонков Михаил Алексеевич,
канд. физ-мат наук
ИСИ СО РАН

Новосибирск
2021г.

(a) Серверы.

Существует множество задач, в которых необходимо обрабатывать много независимых событий.

Сопрограммы

- **Сопрограмма** (англ. coroutine) - программный модуль, организованный для обеспечения взаимодействия с другими модулями по принципу кооперативной многозадачности.
- Сопрограммы способны приостанавливать свое выполнение, сохраняя *контекст* (программный стек и регистры), и передавать управление другой.

Ключевые отличия от потоков ОС

- Переключение контекста сопрограммы требует меньше накладных расходов, чем потока.
- Как правило меньший размер стека, а значит, потребление памяти так же меньше.

Поддержка в языках программирования

(a) C++20

(b) C#

(c) Go

В языке Java сопрограммы не реализованы.

Project Loom

Fibers and Continuations

- Project Loom – проект на базе OpenJDK, целью которого является разработка сопрограмм для языка Java.
- На данный момент уже доступна ранняя версия проекта.

Цели и задачи

Цель: реализация прототипа сопрограмм в Java.
Поставленные задачи:

- Разработать тесты для сравнения производительности потоков и сопрограмм.
- Реализовать переключение сопрограмм.
- Реализовать трассировку ссылок объектов на стеках сопрограмм для сборки мусора.
- Сравнить производительность сопрограмм и потоков.

Работа проводится на базе Huawei JDK.

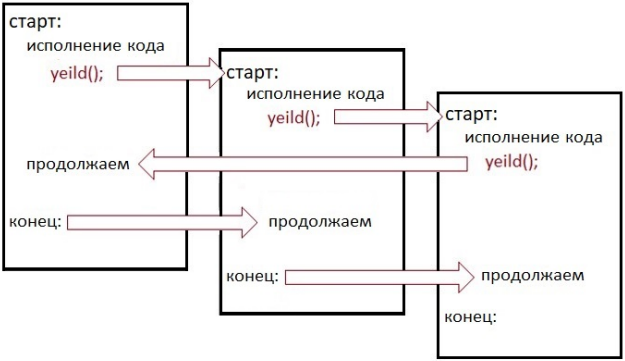
Тесты производительности

Был создан набор тестов производительности сопрограмм для языков Go, Java (с “Loom Project”).
Тесты создавались для измерения 2 параметров.

- Скорость переключения контекста.
- Потребление памяти.

Репозиторий с тестами: https://github.com/minium2/coroutines-benchmark

Переключение сопрограмм



Подходы к реализации:

- OpenJDK(Проект ”Loom”): копирование стека сопрограммы при переключении.
- Go и HuaweiJDK: изменение указателя стека.

Трассировка стеков

- Для работы сборщика мусора необходимо хранить адрес начала и конца стека каждой сопрограммы.
- При сборке мусора сканируются все стеки сопрограмм для поиска корневого множества живых объектов.

Результаты: скорости переключения потоков и сопрограмм

Ubuntu, Intel Core i7-8700, 31 Гб ОЗУ, HuaweiJDK
Каждое значение усреднено по 100 измерениям.
Для измерения используется только одно ядро ЦП.

Шт.	*Число переключений, тыс./сек.*	
	Сопрограммы	*Потоки*
100	*1 247 ± 13*	*2 306 ± 50*
1’000	*1 199 ± 12*	*2 300 ± 27*
5’000	*1 076 ± 59*	*1 554 ± 37*
10’000	*1 017 ± 10*	*1 016 ± 29*
20’000	*917 ± 8*	*753 ± 28*
30’000	*859 ± 4*	*556 ± 16*
40’000	*790 ± 8*	*436 ± 12*
50’000	*756 ± 8*	*361 ± 8*

Результаты: скорости переключения сопрограмм в управляемых средах

Ubuntu, Intel Core i7-8700, 31 Гб ОЗУ, HuaweiJDK
Каждое значение усреднено по 100 измерениям.

Шт.	*Число переключений, тыс./сек.*		
	HuaweiJDK	*OpenJDK("Loom Project")*	*Go*
100	*1 246 ± 13*	*1 900 ± 20*	*18 187 ± 219*
1 000	*1 199 ± 12*	*1 775 ± 20*	*17 934 ± 332*
5 000	*1 075 ± 59*	*1 703 ± 30*	*12 892 ± 339*
10 000	*1 016 ± 10*	*1 924 ± 235*	*8 307 ± 80*
20 000	*916 ± 8*	*1 863 ± 217*	*7 045 ± 72*
30 000	*858 ± 4*	*1 772 ± 182*	*6 391 ± 94*
40 000	*790 ± 8*	*1 606 ± 194*	*5 790 ± 67*
50 000	*756 ± 8*	*1 503 ± 157*	*5 292 ± 122*

Причина худшего результата

- Причина неэффективного переключения сопрограмм – использование медленной функции для переключения контекста.
- Сейчас применяется getcontext/setcontext из glibc, потому что их проще использовать.

Функции для переключения	*Число переключений, дол. ед.*
Из библиотеки Си ***tbox***	*7.8*
Boost.Context	*2.2*
getcontext/setcontext из ***glibc***	*1*

Результаты: потребление памяти

Ubuntu, Intel Core i7-8700, 31 Гб ОЗУ

Шт.	*Резидентная память*		
	HuaweiJDK	*OpenJDK*	*Go*
100	*18 Мб*	*130 Мб*	*3040 Кб*
1000	*23 Мб*	*161 Мб*	*3105 Кб*
5000	*30 Мб*	*187 Мб*	*3156 Кб*
10000	*35 Мб*	*193 Мб*	*3308 Кб*
20000	*40 Мб*	*196 Мб*	*3320 Кб*
30000	*45 Мб*	*197 Мб*	*3350 Кб*
40000	*49 Мб*	*200 Мб*	*3390 Кб*
50000	*55 Мб*	*202 Мб*	*3407 Кб*

Результаты: потребление памяти

Ubuntu, Intel Core i7-8700, 31 Гб ОЗУ, HuaweiJDK

Шт.	*Размер физической памяти*	
	Сопрограммы	*Потоки*
100	*18 Мб*	*34 Мб*
1000	*23 Мб*	*35 Мб*
5000	*30 Мб*	*37 Мб*
10000	*35 Мб*	*40 Мб*
20000	*40 Мб*	*49 Мб*
30000	*45 Мб*	*56 Мб*
40000	*49 Мб*	*63 Мб*
50000	*55 Мб*	*72 Мб*

План дальнейших работ

- Реализовать функцию переключения контекста.
- Поддержка synchronized блоков.
- Переключение сопрограммы при вызове ввода вывода.

Выводы

- Создан набор тестов для сравнения производительности потоков и сопрограмм.
- Реализовано переключение контекста сопрограмм.
- Разработана трассировка ссылок объектов на стеках сопрограмм.
- Проведено сравнение результаты тестов производительности.