Эффективная реализация сопрограмм в управляемой среде исполнения

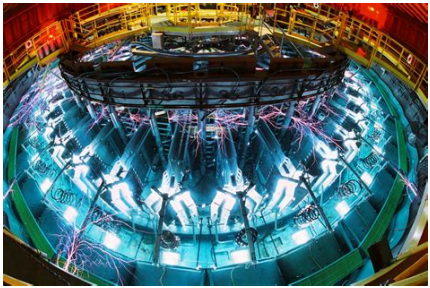
Евгений Пантелеев

Новосибирский государственный университет

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

Новосибирск
2021г.

(a) Серверы.



(b) Ускорители.

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

Плюсы сопрограмм

- 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.

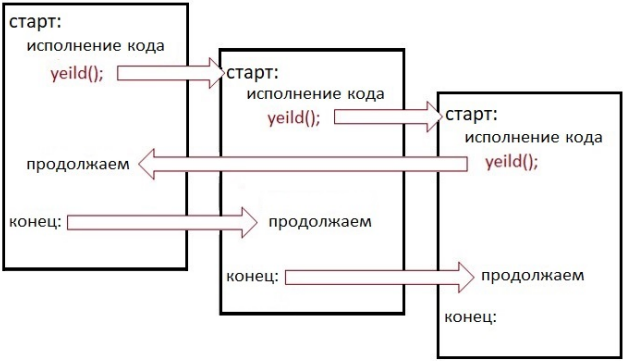
Тесты производительности

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, 1/сек.	
	Сопрограммы	Потоки
100	1’246’756 $\pm$ 12’961	2’306’346 $\pm$ 49’831
1’000	1’199’142 $\pm$ 11’803	2’300’279 $\pm$ 27’180
5’000	1’075’559 $\pm$ 59’328	1’553’872 $\pm$ 36’832
10’000	1’016’802 $\pm$ 9’990	1’015’976 $\pm$ 29’096
20’000	916’809 $\pm$ 8’354	753’123 $\pm$ 28’248
30’000	858’994 $\pm$ 4’307	555’720 $\pm$ 16’102
40’000	790’015 $\pm$ 8’033	436’529 $\pm$ 12’334
50’000	756’523 $\pm$ 8’232	361’088 $\pm$ 7’853

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, 1/сек.		
	HuaweiJDK	OpenJDK(”Loom Project”)	Go
100	1’246’756 ± 12’961	1’900’009 ± 19’732	18’187’799 ± 219’367
1’000	1’199’142 ± 11’803	1’775’239 ± 20’491	17’934’078 ± 332’778
5’000	1’075’559 ± 59’328	1’703’631 ± 30’498	12’892’417 ± 339’410
10’000	1’016’802 ± 9’990	1’924’971 ± 234’982	8’307’791 ± 79’652
20’000	916’809 ± 8’354	1’863’342 ± 217’482	7’045’984 ± 72’584
30’000	858’994 ± 4’307	1’772’720 ± 182’023	6’391’629 ± 94’370
40’000	790’015 ± 8’033	1’606’534 ± 194’728	5’790’831 ± 66’910
50’000	756’523 ± 8’232	1’503’444 ± 157’186	5’292’780 ± 121’844

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

- Переделать 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.