

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.

- ▶ **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.



(a) Серверы.



(b) Браузеры.

- ▶ Обработка множества независимых событий.
- ▶ Организация асинхронного ввода/вывода.

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



(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.

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.

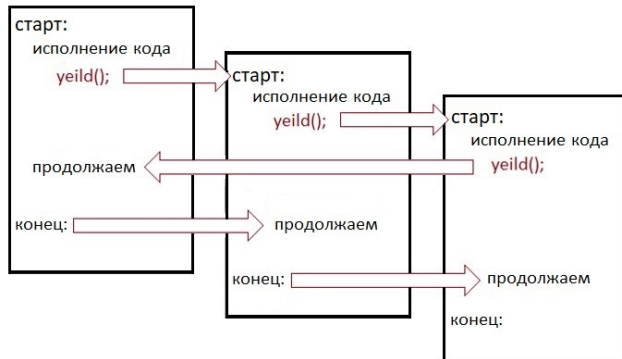
Был создан набор тестов производительности сопрограмм для языков Go, Java (с “Loom Project”).

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.

- ▶ 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 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 <i>tbox</i>	7.8
<i>Boost.Context</i>	2.2
<code>getcontext/setcontext</code> из <i>glibc</i>	1

Результаты: скорости переключения сопрограмм в управляемых средах с новыми функциями переключения контекста

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

Каждое значение усреднено по 100 измерениям.

Для измерения используется только одно ядро ЦП.

<i>Шт.</i>	<i>Число переключений, тыс./сек.</i>	
	<i>getcontext/setcontext</i>	<i>Новые функции</i>
100	1 246 ± 13	5 599 ± 73
1 000	1 199 ± 12	5 375 ± 77
5 000	1 075 ± 59	3 510 ± 75
10 000	1 016 ± 10	3 076 ± 58
20 000	916 ± 8	2 472 ± 39
30 000	858 ± 4	2 106 ± 38
40 000	790 ± 8	1 856 ± 36
50 000	756 ± 8	1 673 ± 28

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.

<i>Шт.</i>	<i>Число переключений, тыс./сек.</i>	
	<i>Сопрограммы</i>	<i>Потоки</i>
<i>100</i>	<i>5 599 ± 73</i>	<i>2 306 ± 50</i>
<i>1 000</i>	<i>5 375 ± 77</i>	<i>2 300 ± 27</i>
<i>5 000</i>	<i>3 510 ± 75</i>	<i>1 554 ± 37</i>
<i>10 000</i>	<i>3 076 ± 58</i>	<i>1 016 ± 29</i>
<i>20 000</i>	<i>2 472 ± 39</i>	<i>753 ± 28</i>
<i>30 000</i>	<i>2 106 ± 38</i>	<i>556 ± 16</i>
<i>40 000</i>	<i>1 856 ± 36</i>	<i>436 ± 12</i>
<i>50 000</i>	<i>1 673 ± 28</i>	<i>361 ± 8</i>

Результаты: потребление памяти

Ubuntu, Intel Core i7-8700, 31 Гб ОЗУ

<i>Шт.</i>	<i>Резидентная память</i>		
	<i>HuaweiJDK</i>	<i>OpenJDK</i>	<i>Go</i>
<i>100</i>	<i>18 Мб</i>	<i>130 Мб</i>	<i>3,04 Мб</i>
<i>1000</i>	<i>23 Мб</i>	<i>161 Мб</i>	<i>3,105 Мб</i>
<i>5000</i>	<i>30 Мб</i>	<i>187 Мб</i>	<i>3,156 Мб</i>
<i>10000</i>	<i>35 Мб</i>	<i>193 Мб</i>	<i>3,308 Мб</i>
<i>20000</i>	<i>40 Мб</i>	<i>196 Мб</i>	<i>3,320 Мб</i>
<i>30000</i>	<i>45 Мб</i>	<i>197 Мб</i>	<i>3,350 Мб</i>
<i>40000</i>	<i>49 Мб</i>	<i>200 Мб</i>	<i>3,390 Мб</i>
<i>50000</i>	<i>55 Мб</i>	<i>202 Мб</i>	<i>3,407 Мб</i>

Результаты: потребление памяти

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

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

План дальнейших работ

- ▶ Поддержка synchronized блоков.
- ▶ Переключение сопрограммы при вызове ввода вывода.

- ▶ 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.