

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 переключение 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”).

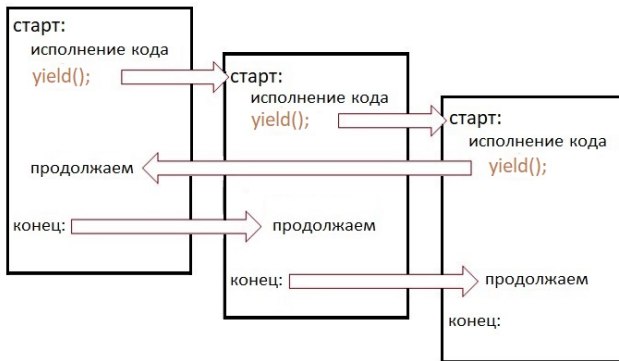
Тесты создавались для измерения 2 параметров.

- ▶ Скорость переключения контекста.
- ▶ Потребление памяти.

Репозиторий с тестами:

<https://github.com/minium2/coroutines-benchmark>

Переключение сопрограмм



- ▶ Функция `yield()` переключает управление от одной сопрограммы другой.
- ▶ Завершение выполнения сопрограммы приводит к переключению на другую.

Подходы к реализации переключения сопрограмм

- ▶ OpenJDK/"Loom": копирование стека сопрограммы при переключении.
- ▶ Go: изменение указателя стека.

В HuaweiJDK выбран подход языка Go, поскольку он более эффективен.

- ▶ 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.

Измерение скорости переключения сопрограмм

Ubuntu, kernel 4.15, Intel Core i7-8700, 4.6 ГГц, 32 Гб ОЗУ
Каждое значение усреднено по 100 измерениям.

<i>Шт.</i>	<i>Число переключений, тыс./сек.</i>		
	<i>HuaweiJDK</i>	<i>OpenJDK/"Loom"</i>	<i>Go</i>
100	1 956 ± 38	1 900 ± 20	18 187 ± 219
1 000	1 829 ± 12	1 775 ± 20	17 934 ± 332
5 000	1 578 ± 39	1 703 ± 30	12 892 ± 339
10 000	1 316 ± 20	1 924 ± 235	8 307 ± 80
20 000	1226 ± 8	1 863 ± 217	7 045 ± 72
30 000	1068 ± 7	1 772 ± 182	6 391 ± 94
40 000	928 ± 7	1 606 ± 194	5 790 ± 67
50 000	881 ± 5	1 503 ± 157	5 292 ± 122

Функции для переключения контекста

- ▶ Первый прототип использовал функции для переключения контекста `getcontext/setcontext` из `glibc`.

Функции для переключения	Число переключений, дол. ед.
Из библиотеки <code>Cu tbox</code>	7.8
<code>Boost.Context</code>	2.2
<code>getcontext/setcontext</code> из <code>glibc</code>	1

К сожалению, эти функции нельзя использовать, поскольку они не учитывают внутренние особенности JVM.

Измерение скорости переключения сопрограмм в HuaweiJDK с новыми функциями переключения контекста

Ubuntu, kernel 4.15, Intel Core i7-8700, 4.6 ГГц, 32 Гб ОЗУ
Каждое значение усреднено по 100 измерениям.
Для измерения используется только одно ядро ЦП.

Шт.	Число переключений, тыс./сек.	
	<i>getcontext/setcontext</i>	Новые функции
100	1 956 ± 38	12 980 ± 540
1 000	1 829 ± 12	11 420 ± 694
5 000	1 578 ± 39	5 875 ± 183
10 000	1 316 ± 20	4 459 ± 162
20 000	1226 ± 8	3 604 ± 93
30 000	1068 ± 7	3 031 ± 94
40 000	928 ± 7	2 653 ± 87
50 000	881 ± 5	2 315 ± 60

Измерение скорости переключения сопрограмм в управляемых средах

Ubuntu, kernel 4.15, Intel Core i7-8700, 4.6 ГГц, 32 Гб ОЗУ
Каждое значение усреднено по 100 измерениям.

<i>Шт.</i>	<i>Число переключений, тыс./сек.</i>		
	<i>HuaweiJDK</i>	<i>OpenJDK/"Loom"</i>	<i>Go</i>
<i>100</i>	<i>12 980 ± 540</i>	<i>1 900 ± 20</i>	<i>18 187 ± 219</i>
<i>1 000</i>	<i>11 420 ± 694</i>	<i>1 775 ± 20</i>	<i>17 934 ± 332</i>
<i>5 000</i>	<i>5 875 ± 183</i>	<i>1 703 ± 30</i>	<i>12 892 ± 339</i>
<i>10 000</i>	<i>4 459 ± 162</i>	<i>1 924 ± 235</i>	<i>8 307 ± 80</i>
<i>20 000</i>	<i>3 604 ± 93</i>	<i>1 863 ± 217</i>	<i>7 045 ± 72</i>
<i>30 000</i>	<i>3 031 ± 94</i>	<i>1 772 ± 182</i>	<i>6 391 ± 94</i>
<i>40 000</i>	<i>2 653 ± 87</i>	<i>1 606 ± 194</i>	<i>5 790 ± 67</i>
<i>50 000</i>	<i>2 315 ± 60</i>	<i>1 503 ± 157</i>	<i>5 292 ± 122</i>

Измерение скорости переключения потоков и сопрограмм

Ubuntu, kernel 4.15, Intel Core i7-8700, 4.6 ГГц, 32 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>
100	12 980 ± 540	2 306 ± 50
1 000	11 420 ± 694	2 300 ± 27
5 000	5 875 ± 183	1 554 ± 37
10 000	4 459 ± 162	1 016 ± 29
20 000	3 604 ± 93	753 ± 28
30 000	3 031 ± 94	556 ± 16
40 000	2 653 ± 87	436 ± 12
50 000	2 315 ± 60	361 ± 8

Измерение потребление памяти сопрограмм в управляемых средах

Ubuntu, kernel 4.15, Intel Core i7-8700, 4.6 ГГц, 32 Гб ОЗУ

<i>Шт.</i>	<i>Резидентная память</i>		
	<i>HuaweiJDK</i>	<i>OpenJDK"J9"</i>	<i>Go</i>
<i>100</i>	<i>18 Мб</i>	<i>130 Мб</i>	<i>3,040 Мб</i>
<i>1000</i>	<i>22 Мб</i>	<i>161 Мб</i>	<i>3,105 Мб</i>
<i>5000</i>	<i>32 Мб</i>	<i>187 Мб</i>	<i>3,156 Мб</i>
<i>10000</i>	<i>37 Мб</i>	<i>193 Мб</i>	<i>3,308 Мб</i>
<i>20000</i>	<i>45 Мб</i>	<i>196 Мб</i>	<i>3,320 Мб</i>
<i>30000</i>	<i>49 Мб</i>	<i>197 Мб</i>	<i>3,350 Мб</i>
<i>40000</i>	<i>51 Мб</i>	<i>200 Мб</i>	<i>3,390 Мб</i>
<i>50000</i>	<i>57 Мб</i>	<i>202 Мб</i>	<i>3,407 Мб</i>

Измерение потребление памяти потоков

Ubuntu, kernel 4.15, Intel Core i7-8700, 4.6 ГГц, 32 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>22 Мб</i>	<i>35 Мб</i>
<i>5000</i>	<i>32 Мб</i>	<i>37 Мб</i>
<i>10000</i>	<i>37 Мб</i>	<i>40 Мб</i>
<i>20000</i>	<i>45 Мб</i>	<i>49 Мб</i>
<i>30000</i>	<i>49 Мб</i>	<i>56 Мб</i>
<i>40000</i>	<i>51 Мб</i>	<i>63 Мб</i>
<i>50000</i>	<i>57 Мб</i>	<i>72 Мб</i>

- ▶ 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.
- ▶ Оптимизировано переключение контекста сопрограмм более чем в 3 раза.
- ▶ Проведен анализ результатов тестов производительности.