

# Исследование применимости сопрограмм в параллельных системах обработки данных.

Евгений Пантелеев

Новосибирский государственный университет

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

Новосибирск  
2021г.

В мире существует множество задач, требующих параллельной обработки данных:

- ▶ Вычислительные задачи, вроде операций с матрицами.
- ▶ Системы сбора и обработки данных.

- ▶ Традиционно, параллелизм реализуется внутри операционной системы с помощью механизма потоков.
- ▶ Однако, модель потоков имеет ряд минусов.
- ▶ Потоки – это достаточно ”тяжеловесный” механизм: их создание и переключение несет в себе крупные накладные расходы.
- ▶ Избежать накладных расходов на использование потоков можно, применяя вместо них 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.

# Реализация 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.

Цель: изучение применимости сопрограмм вместо потоков в параллельных системах.

Поставленные задачи:

- ▶ Провести анализ реализаций сопрограмм в других языках.
- ▶ Реализовать прототип модуля сопрограмм.
- ▶ Сравнить производительность сопрограмм и потоков.
- ▶ Выявить ключевые плюсы использования сопрограмм.

Работа проводится на базе Huawei JDK.

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

Тесты создавались для измерения 3 параметров.

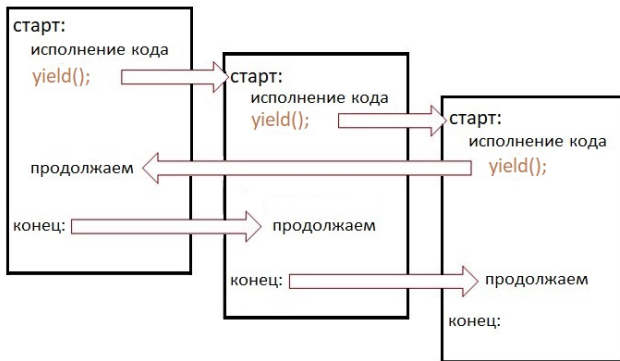
- ▶ Скорость переключения контекста.
- ▶ Потребление памяти.

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.

# 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 потребления 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>

Для измерений была применена библиотека Colt Parallel и ее вариант на сопрограмаах.

Время перемножения матриц 6000x7000

Потоки	Сопрограммы
$45.6 \pm 1.8$ сек.	$24.0 \pm 5.6$ сек.

Время вычисления дискретного преобразования Фурье.

Потоки	Сопрограммы
$77.2 \pm 0.3$ мсек.	$77.4 \pm 0.4$ мсек.

- ▶ Переключение контекста сопрогаммы требует меньше накладных расходов, чем потока. Измерено: скорость переключения сопрограмм в 5–7 раз больше, чем у потоков.
- ▶ Меньшее потребление физической памяти не менее, чем на 30%.



- ▶ Поддержать synchronized блоки.
- ▶ Реализовать переключение сопрограммы при вызове ввода–вывода.
- ▶ Оценить реальный рост производительности от применения сопрограмм в системах обработки данных.

- ▶ Создан набор тестов для сравнения производительности потоков и сопрограмм.
- ▶ Реализован базовый прототип сопрограмм на базе HuaweiJDK.
- ▶ Проведен анализ результатов тестов производительности.
- ▶ Выявлены ключевые отличия сопрограмм от потоков. Можно ожидать прирост производительности при использовании сопрограмм.