

Исследование применимости сопрограмм в параллельных системах обработки данных.

Евгений Пантелеев

Новосибирский государственный университет

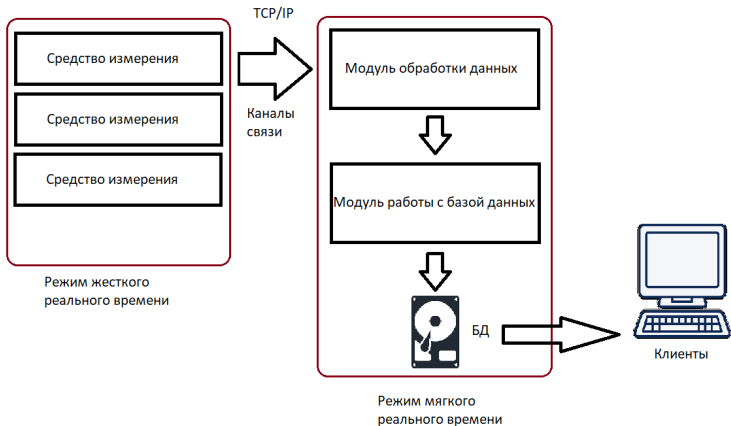
Научный руководитель: Бульонков Михаил Алексеевич,
канд. физ-мат наук
ИСИ СО РАН

Новосибирск
2021г.

В мире существует множество задач, требующих параллельной обработки данных:

- ▶ Параллельные вычисления. Например, библиотека Colt Parallel, используемая CERN для математических расчетов.
- ▶ Системы сбора и обработки данных.

Система обработки и сбора данных



- Для ускорения работы со множеством каналов связи необходим параллелизм.

1. Допустимо применение языка Java в системах мягкого реального времени.
2. Java упрощает написание кода благодаря автоматическому управлению памятью и контролю ошибок при исполнении.

- ▶ 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, который работает *конкурентно* с другими такими модулями. При использовании сопрограммы ведут себя как обычные потоки.
- ▶ Сопрограммы уже реализованы в языках программирования C++20, C#, Go.
- ▶ Модуль сопрограмм в Java находится в стадии работающего прототипа (OpenJDK/Loom)

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 2 параметров.

1. Скорость переключения контекста.
 2. Потребление памяти.
- ▶ Разработанные тесты позволяют оценить применимость сопрограмм в системах сбора данных.
 - ▶ Репозиторий с тестами:
<https://github.com/minium2/coroutines-benchmark>.

Для работы минимального прототипа требуется:

1. Переключение контекста сопрограмм.
2. Сборка мусора.

Переключение сопрограмм может быть реализовано различными способами:

- ▶ 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

Шт.	Резидентная память			
	HuaweiJDK	OpenJDK/"Loom"	Go	Поток
100	18 Мб	130 Мб	3,040 Мб	34 Мб
1000	22 Мб	161 Мб	3,105 Мб	35 Мб
5000	32 Мб	187 Мб	3,156 Мб	37 Мб
10000	37 Мб	193 Мб	3,308 Мб	40 Мб
20000	45 Мб	196 Мб	3,320 Мб	49 Мб
30000	49 Мб	197 Мб	3,350 Мб	56 Мб
40000	51 Мб	200 Мб	3,390 Мб	63 Мб
50000	57 Мб	202 Мб	3,407 Мб	72 Мб

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

Время перемножения матриц 6000x7000

Потоки	Сопрограммы
45.6 ± 1.8 сек.	24.0 ± 5.6 сек.

Время вычисления дискретного преобразования Фурье.

Потоки	Сопрограммы
77.2 ± 0.3 мсек.	77.4 ± 0.4 мсек.

Измерения проводились на CentOS, linux 4.18.0 Intel® Xeon® Gold 6130, 2.10 ГГц.

- ▶ 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 более, чем на 20%.

- ▶ 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 в системах сбора данных.