Исследование применимости сопрограмм в параллельных системах обработки данных.

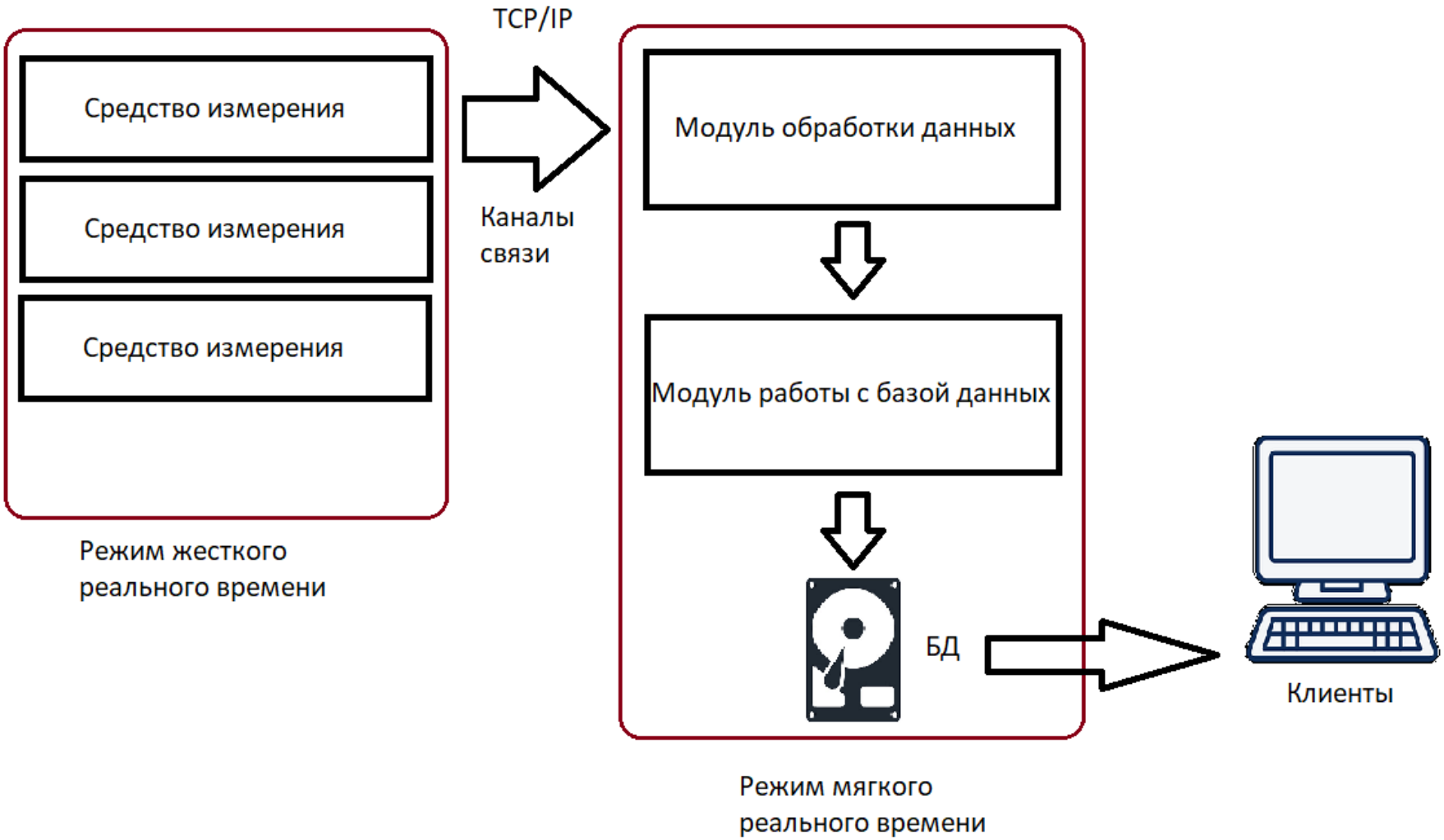
Евгений Пантелеев

Новосибирский государственный университет

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

Новосибирск
2021г.

Система обработки данных



Язык Java

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

Модуль обработки данных

Как можно организовать работу с каналами приема данных?

1. Создать поток на обработку каждого канала.
   - Плюсы: простота реализации.
   - Минусы: высокое потребление системных ресурсов: памяти и процессорного времени.
2. Опрос каналов на доступность данных.
   - Плюсы: меньшее потребление ресурсов системы.
   - Минусы: с ростом числа модулей растет задержка приема данных.

Существует альтернативный способ организации – **сопрограммы**.

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.

Цели и задачи

Цель: изучить применимость сопрограмм вместо потоков в программах Java.
Поставленные задачи:

- Разработать тесты для сравнения производительности потоков и сопрограмм.
- Реализовать прототип модуля сопрограмм.
- Сравнить производительность сопрограмм и потоков.
- Выявить ключевые плюсы использования сопрограмм.

Работа проводится на базе Huawei JDK.

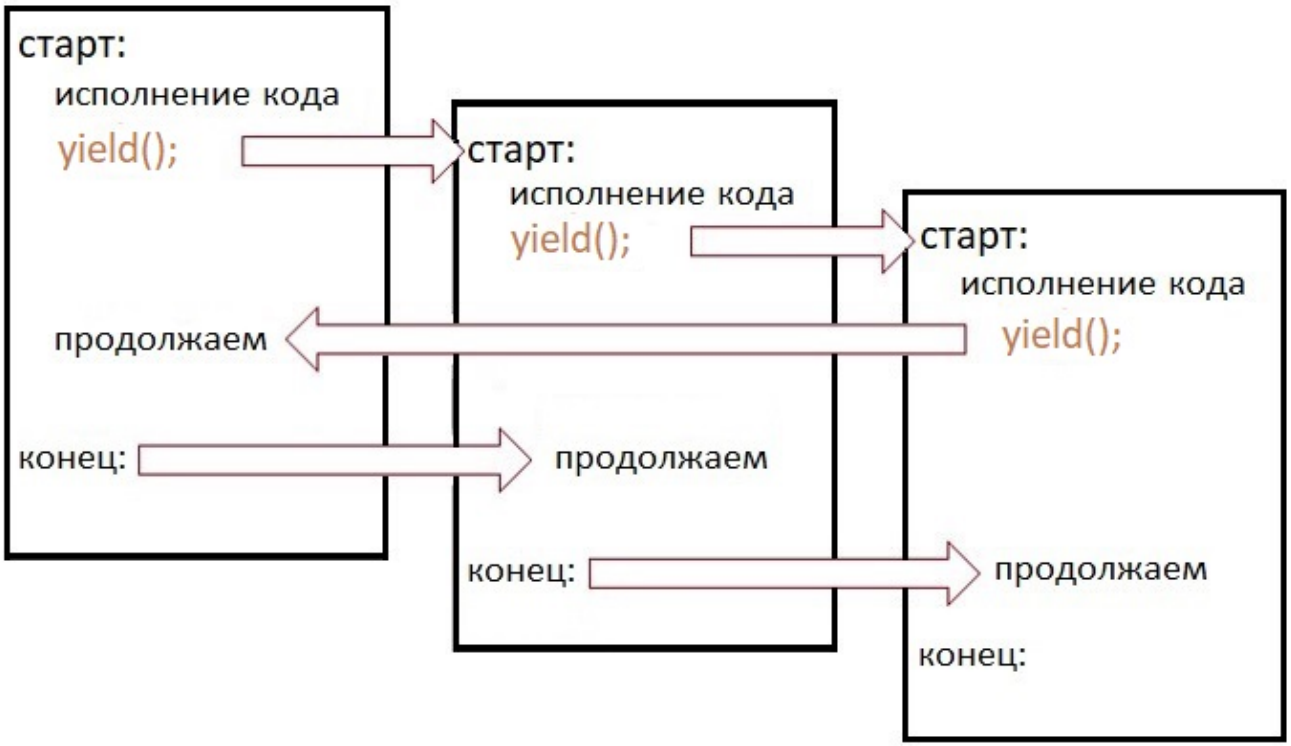
Тесты производительности

Был создан набор тестов производительности сопрограмм для языков Go, Java (с “Loom Project”).
Тесты создавались для измерения 3 параметров.

- Время создания потока/сопрограммы.
- Скорость переключения контекста.
- Потребление памяти.

Репозиторий с тестами:
https://github.com/minium2/coroutines-benchmark

Переключение сопрограмм



- Функция yeild() 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.

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

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.

Шт.	*Число переключений, тыс./сек.*	
	Сопрограммы	*Потоки*
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*
100	*18 Мб*	*130 Мб*	*3,040 Мб*
1000	*22 Мб*	*161 Мб*	*3,105 Мб*
5000	*32 Мб*	*187 Мб*	*3,156 Мб*
10000	*37 Мб*	*193 Мб*	*3,308 Мб*
20000	*45 Мб*	*196 Мб*	*3,320 Мб*
30000	*49 Мб*	*197 Мб*	*3,350 Мб*
40000	*51 Мб*	*200 Мб*	*3,390 Мб*
50000	*57 Мб*	*202 Мб*	*3,407 Мб*

Измерение потребления памяти потоков

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

Шт.	*Размер физической памяти*	
	Сопрограммы	*Потоки*
100	*18 Мб*	*34 Мб*
1000	*22 Мб*	*35 Мб*
5000	*32 Мб*	*37 Мб*
10000	*37 Мб*	*40 Мб*
20000	*45 Мб*	*49 Мб*
30000	*49 Мб*	*56 Мб*
40000	*51 Мб*	*63 Мб*
50000	*57 Мб*	*72 Мб*

Ключевые отличия от потоков ОС

- Переключение контекста сопрограммы 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%.

План дальнейших работ

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

Результаты

- 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.