Поединок:
.NET Core против Java

Виталий Езепчук, программист Fast Reports

witaly@fast-report.com

Делаем ставки: Java или Core?

1. Сканируй
2. Голосуй
3. Жди результаты

https://goo.gl/SGE4DL

Почему .NET Core и Java?

1. Я разработчик решений под обе платформы
2. Позволяют разрабатывать серверные приложения
3. Работают на Windows, Linux и Mac OS

Версии и системы

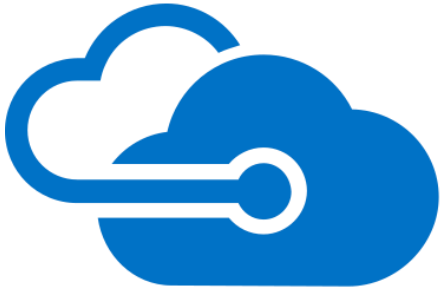
Платформы:

.NET Core, Версия: 2.0.0
от 14 августа 2017

Java от Oracle, Версия 8 Update 144
от 26 июля 2017

Операционные системы:
Windows Server 2016
Ubuntu Server 16.04

DS2_V2 Standard



Настройки платформ

Core

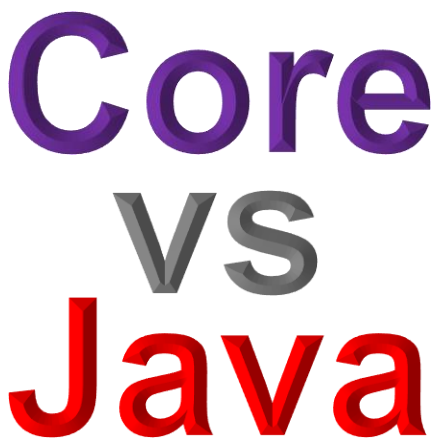
```
gcServer = true
gcConcurrent = true
```

Java

```
-Xms50m -Xmx4g
-XX:+UseG1GC
-XX:+PrintGCDetails
-XX:+UseStringDeduplication
-XX:CompileThreshold=15
```

Список раундов

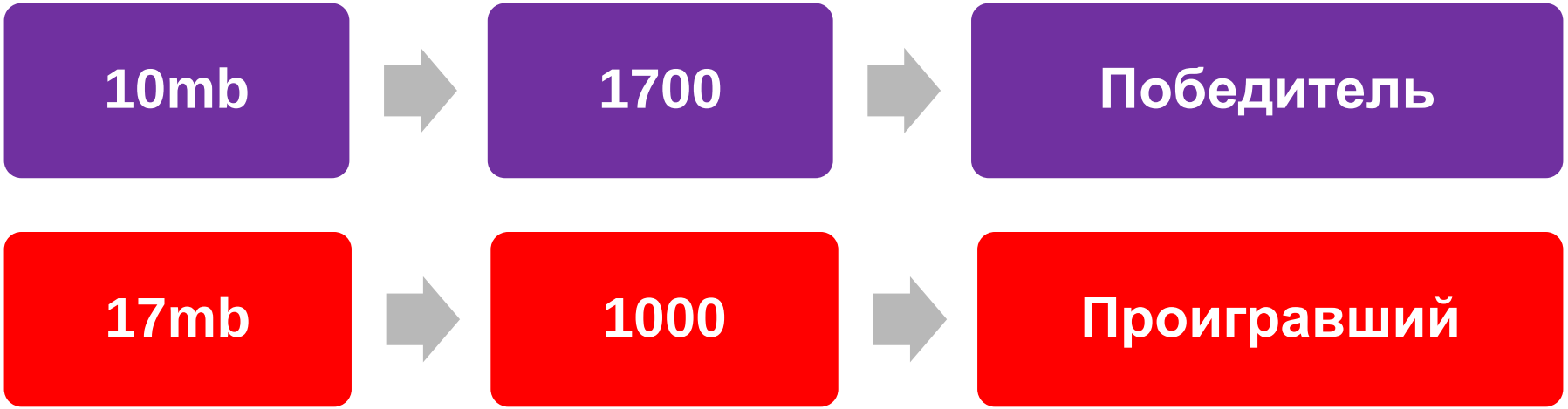
1. Работа с памятью
2. Производительность
3. Потоки и задачи
4. Другие тесты



Система оценок

- Вычисляем время и трафик управляемой памяти
- Две платформы, значит 2 результата
- Худшему результату присваиваем 1000 баллов
- Лучшему результату присваиваем пропорционально с худшим, значение выше 1000 баллов
- Для выбора победителя вычисляем среднее геометрическое

Пример подсчёта



Исходники бенчмарков

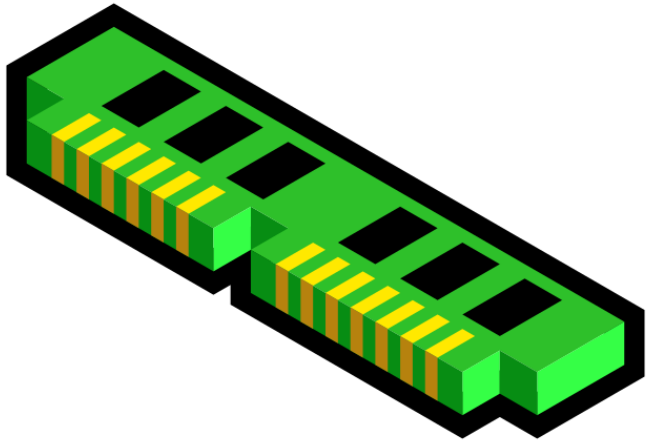
Все исходники бенчмарков доступны на моём GitHub:

https://github.com/detrav/dotnext2017

Работа с памятью

Работа с памятью

1. Строки
2. StringBuilder
3. Объекты
4. MemoryStream



Строки + StringBuilder

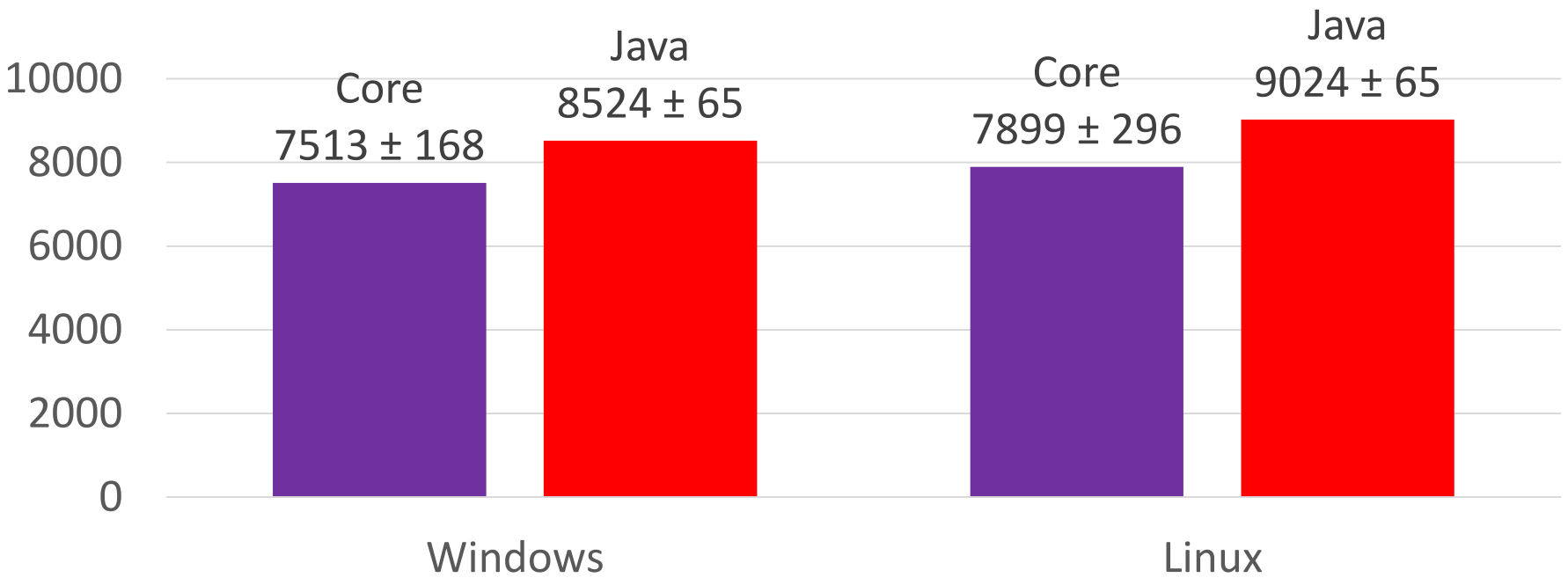
Проверим работу со строками

Три группы тестов:

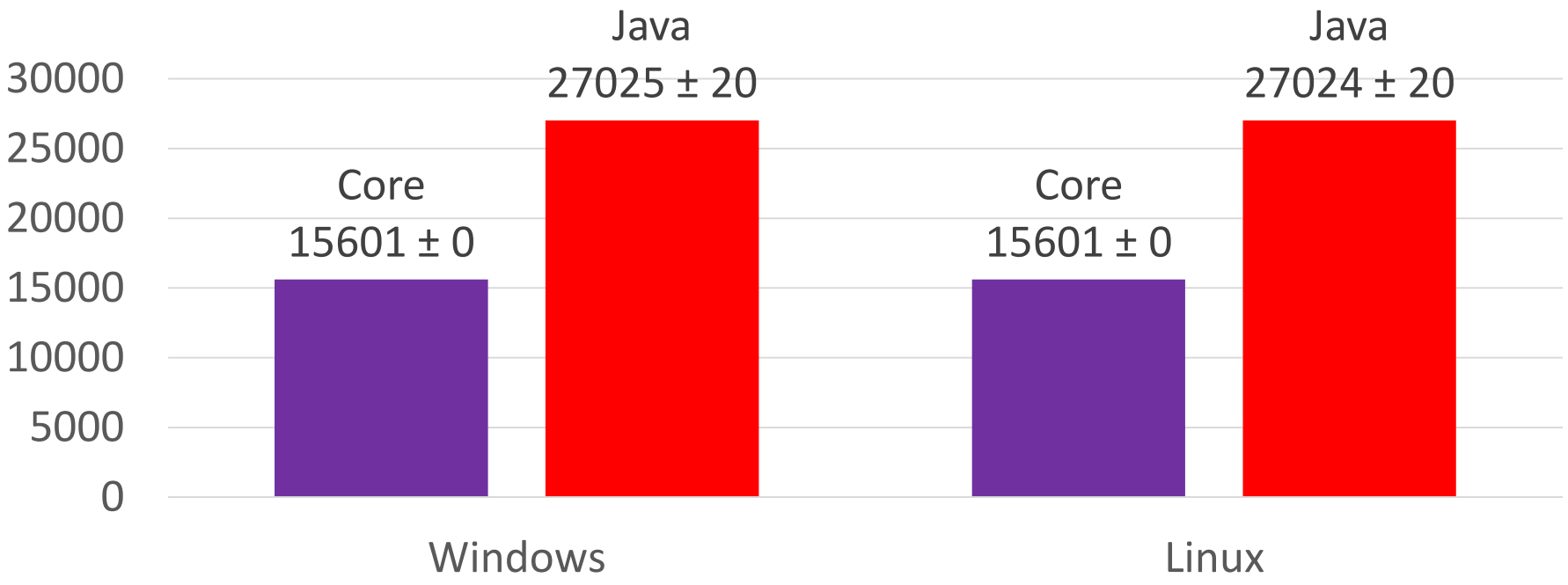
- Конкатенация, 2500 сложений,
  в результате строка длиной 1 280 000 символов
- Стандартные методы: IndexOf, Substring, Split, ToUpper, ToLower
- StringBuilder, 250 000 итераций

Используется подготовленная строка 512 символов

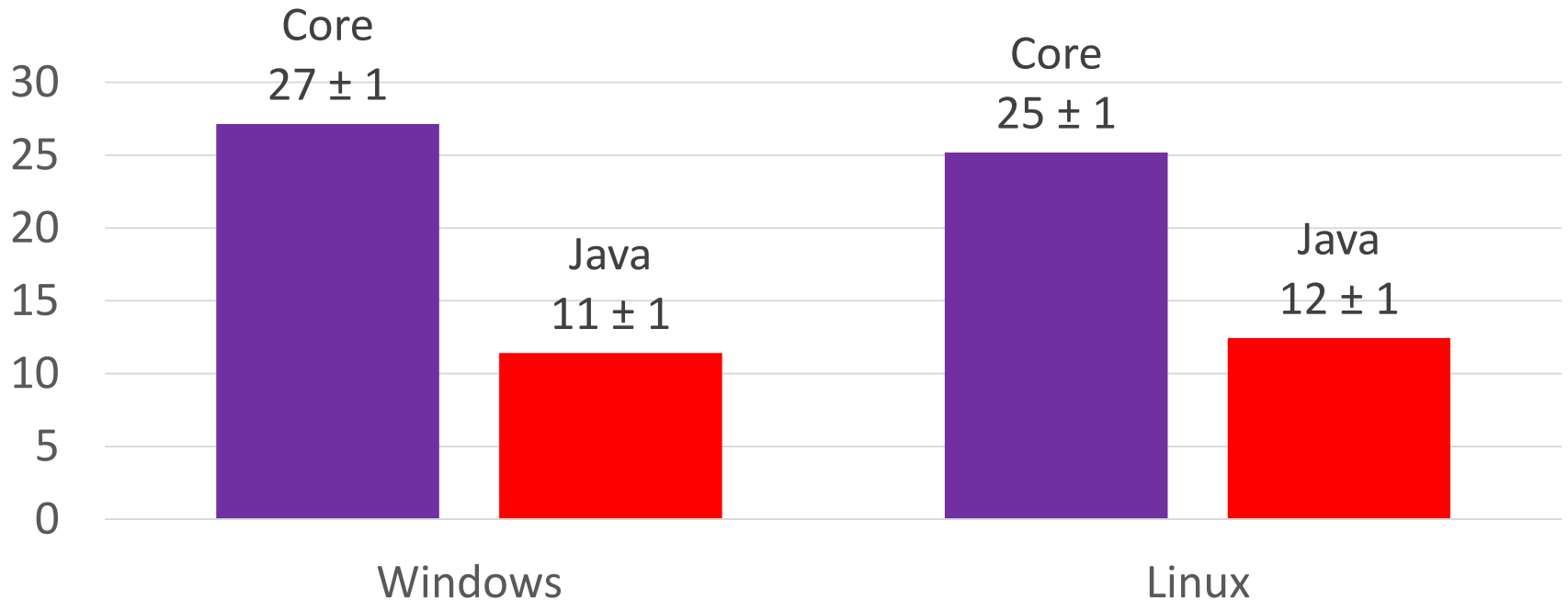
Строки, конкатенация: время (мс)



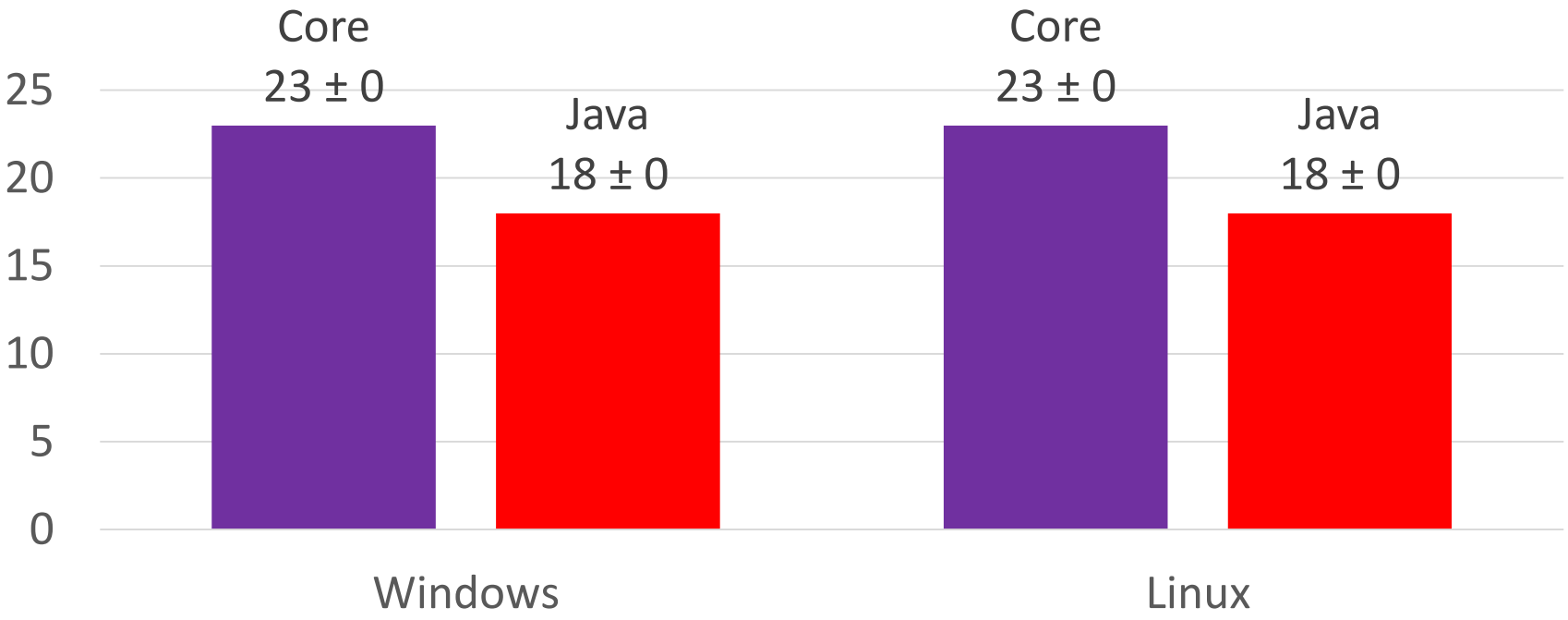
Строки, конкатенация: память (мб)



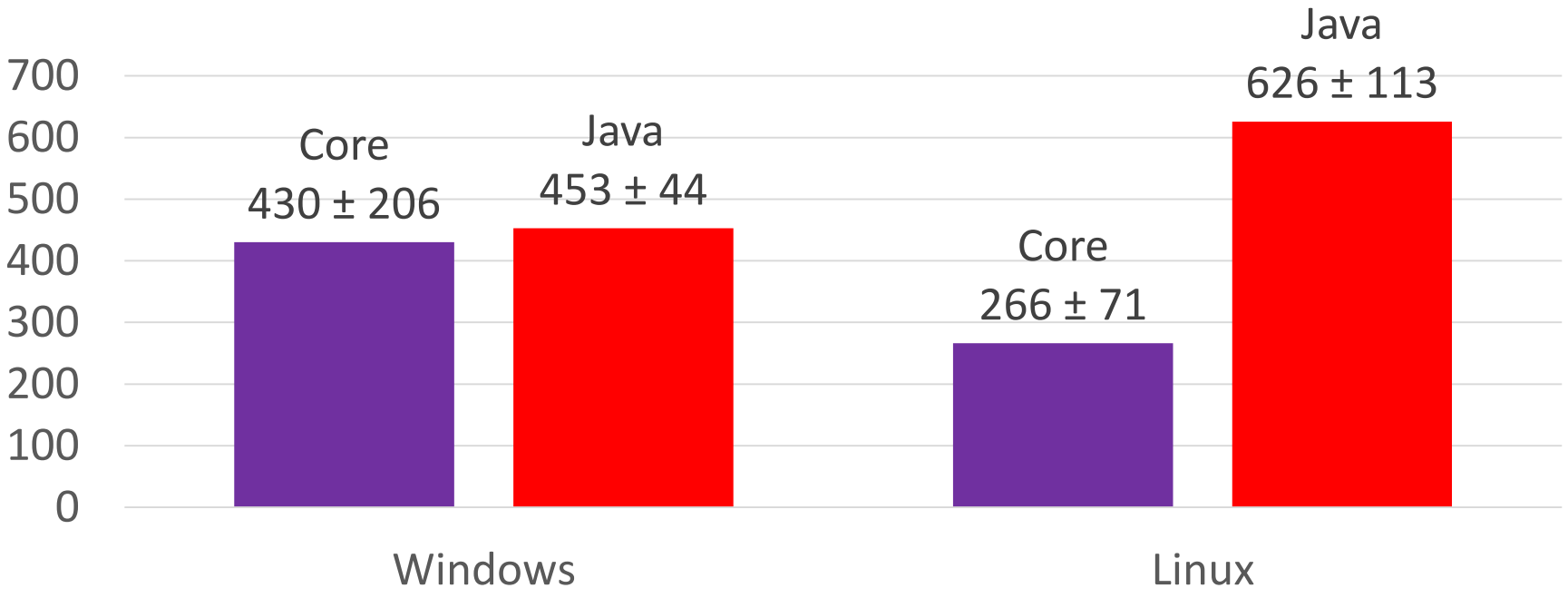
Строки, стандартные методы: время (мс)



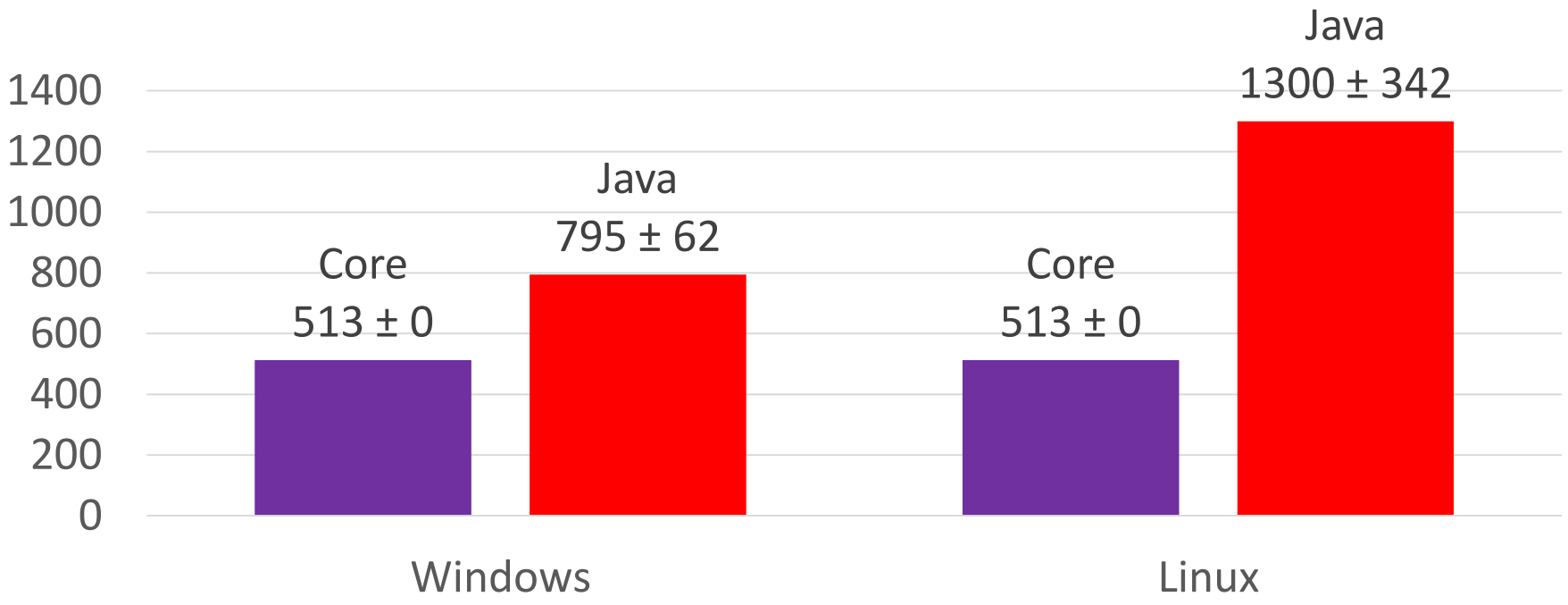
Строки, стандартные методы: память (мб)



Строки, StringBuilder: время (мс)



Строки, StringBuilder: память (мб)

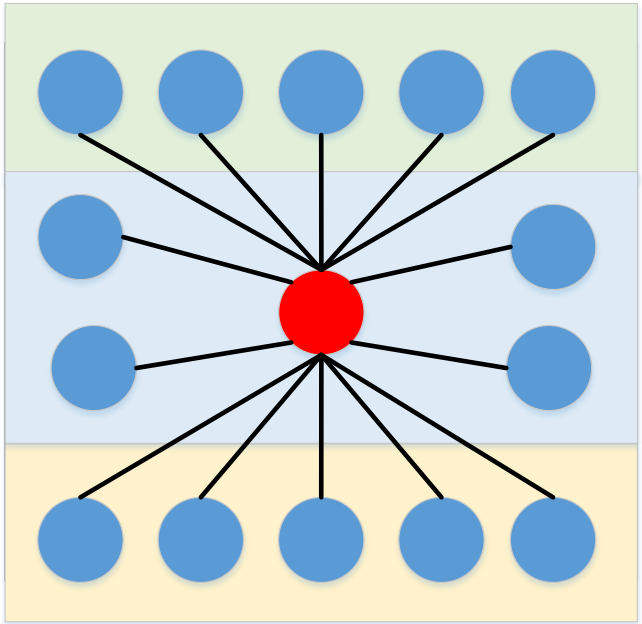


Объекты

Протестируем аллокацию и большое количество объектов

Используется регулярный граф 14-ой степени

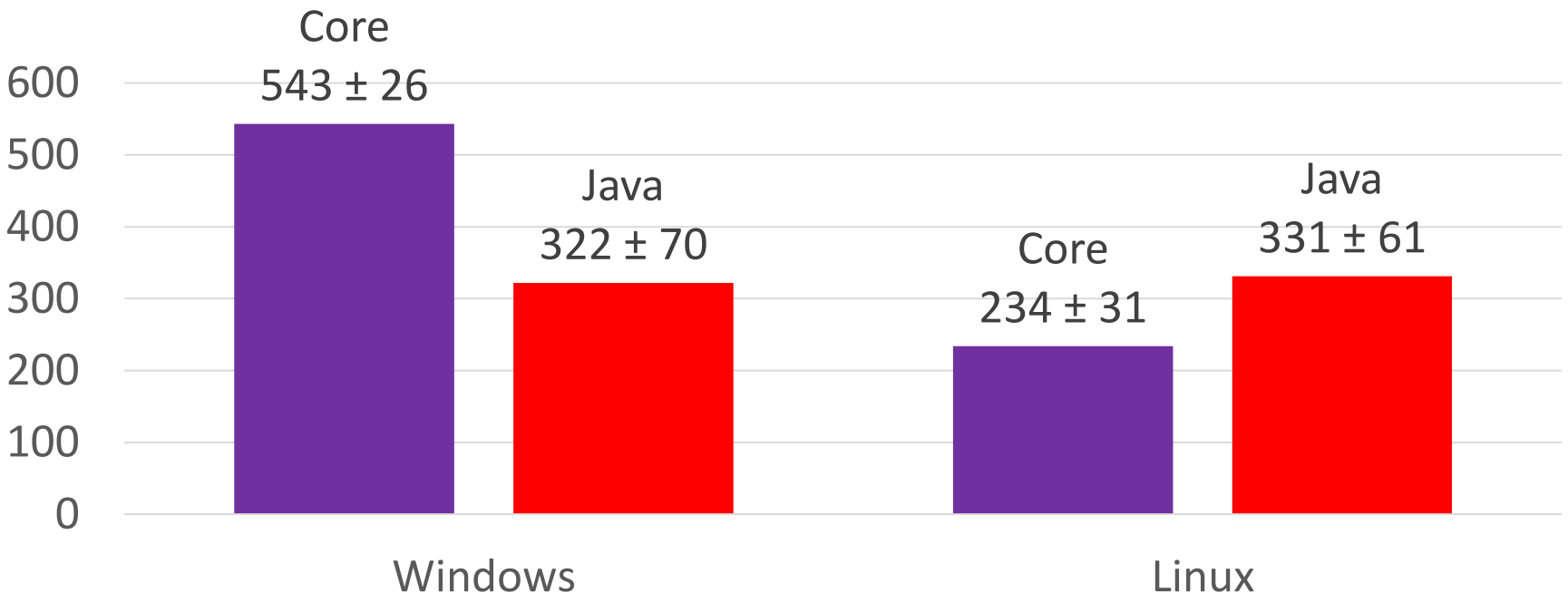
Пустой массив, список, строка, StringBuilder и Dictionary в каждом узле



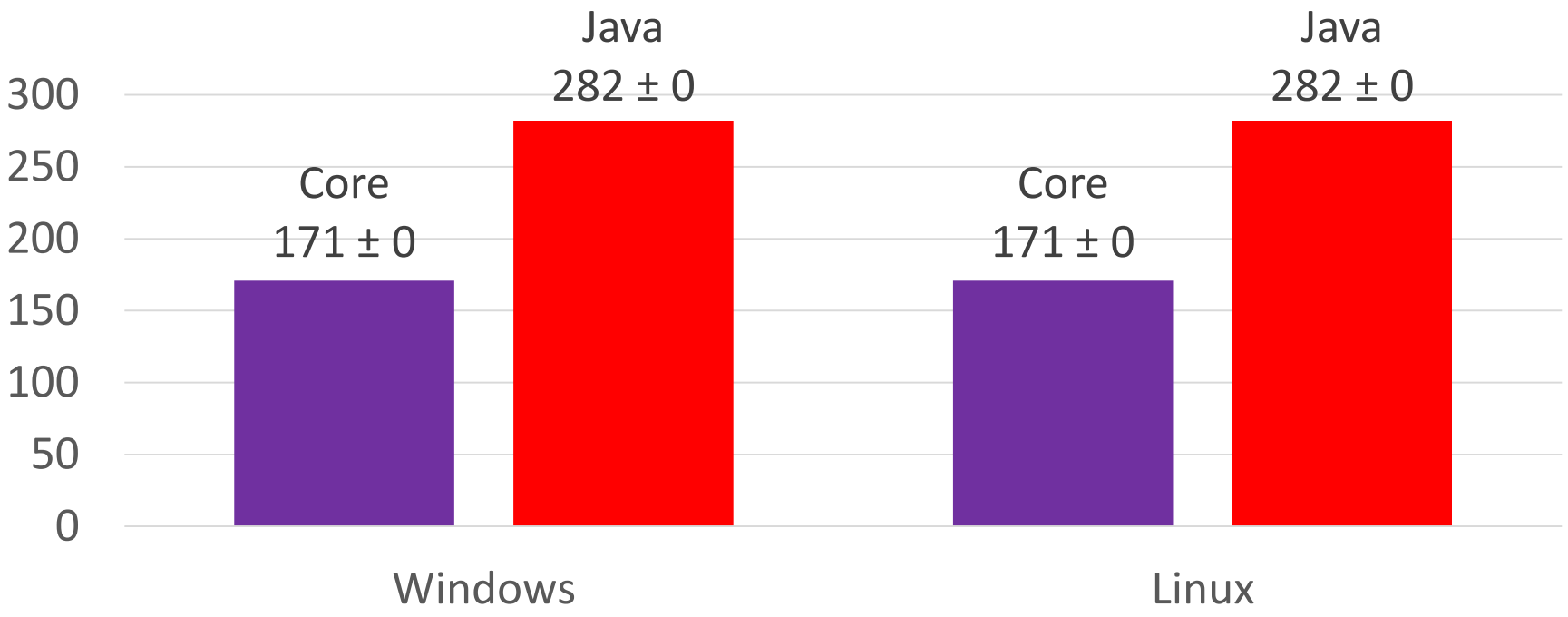
Объекты, описание тестов

1. Создаём новую сеть 100 000 слоёв (2 теста: добавляем «мусор», не добавляем «мусор»)
2. Заменяем 4-й объект в каждом слое на новый
3. Заменяем каждый 3-й слой на новый
4. Удаляем каждый 2-й слой
5. Заполняем до полной сети
6. Реверс сети

Объекты: время (мс)



Объекты: память (мб)



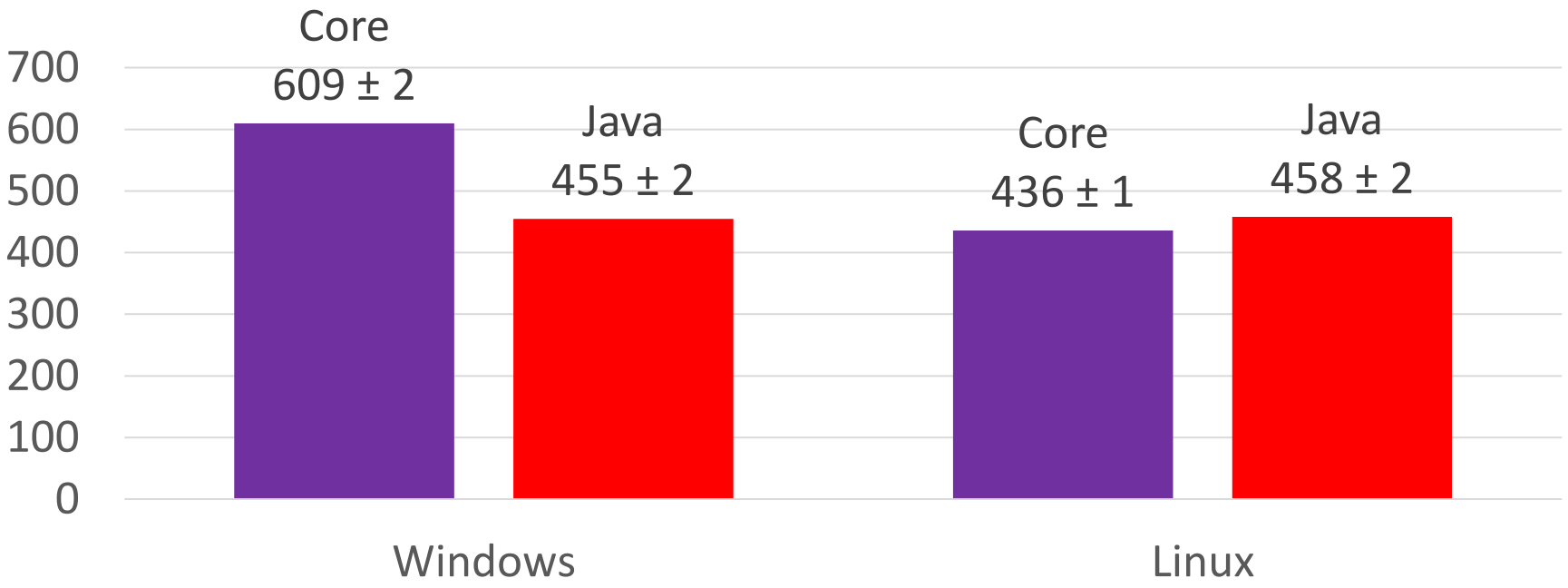
MemoryStream

Сравним скорость работы с MemoryStream

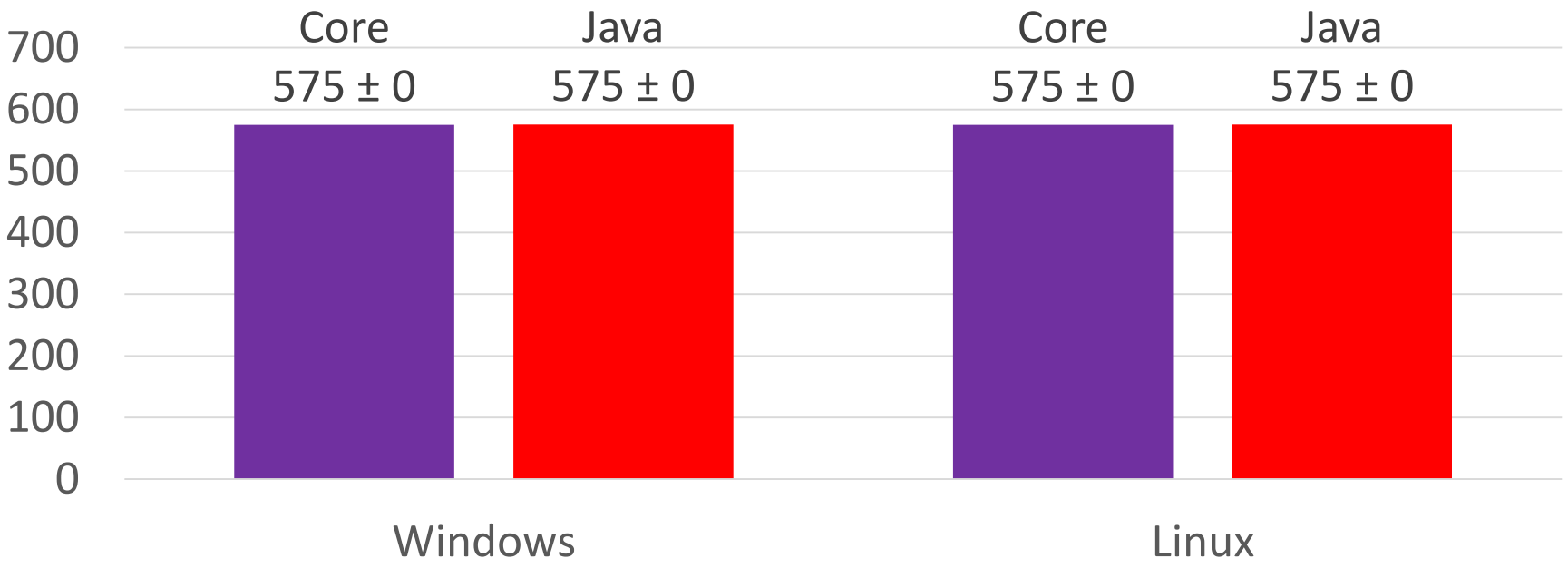
Заранее подготовленный файл на 1мб

- Создание и запись в поток по 1 байту, 1кб и 1 мб
- Копирование из одного потока в другой
- Генерация массива байт из потока
- Переливание Int32 значений из одного потока в другой

MemoryStream: время (мс)



MemoryStream: память (мб)



Первый раунд окончен, Windows

	Время		Память	
	Core	Java	Core	Java
Конкатенация	1135	1000	1732	1000
Строки	1000	2379	1000	1278
StringBuilder	1053	1000	1549	1000
Объекты	1000	1687	1651	1000
MemoryStream	1000	1340	1000	1000
Итого	1036	1400	1347	1050

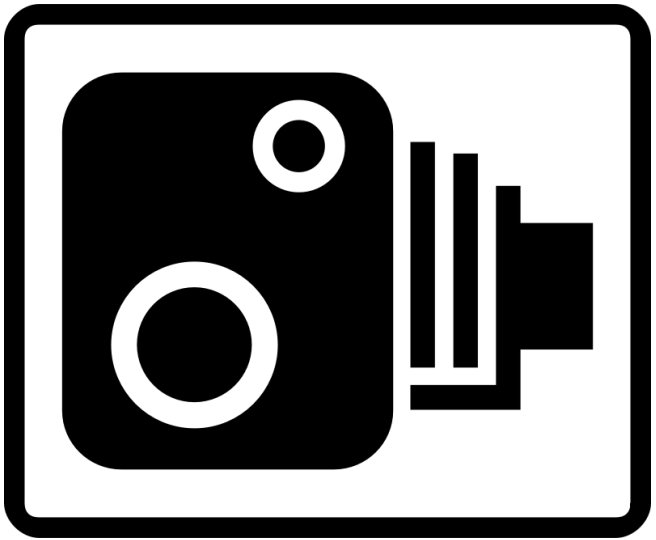
Первый раунд окончен, Linux

	Время		Память	
	Core	Java	Core	Java
Конкатенация	1143	1000	1732	1000
Строки	1000	2024	1000	1278
StringBuilder	2353	1000	2533	1000
Объекты	1416	1000	1651	1000
MemoryStream	1050	1000	1000	1000
Итого	1319	1151	1486	1050

Производительность

Производительность

1. Переменные, математика и рандом
2. Рекурсия
3. Регулярные выражения
4. Коллекции

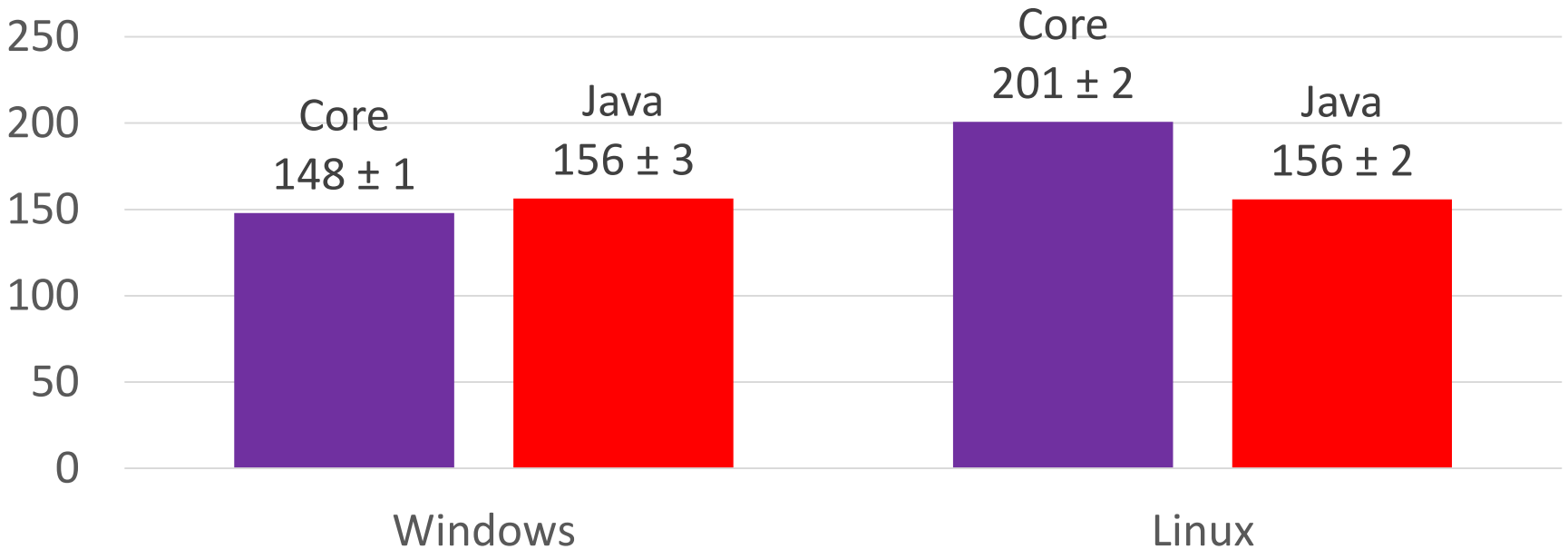


Переменные, математика и рандом

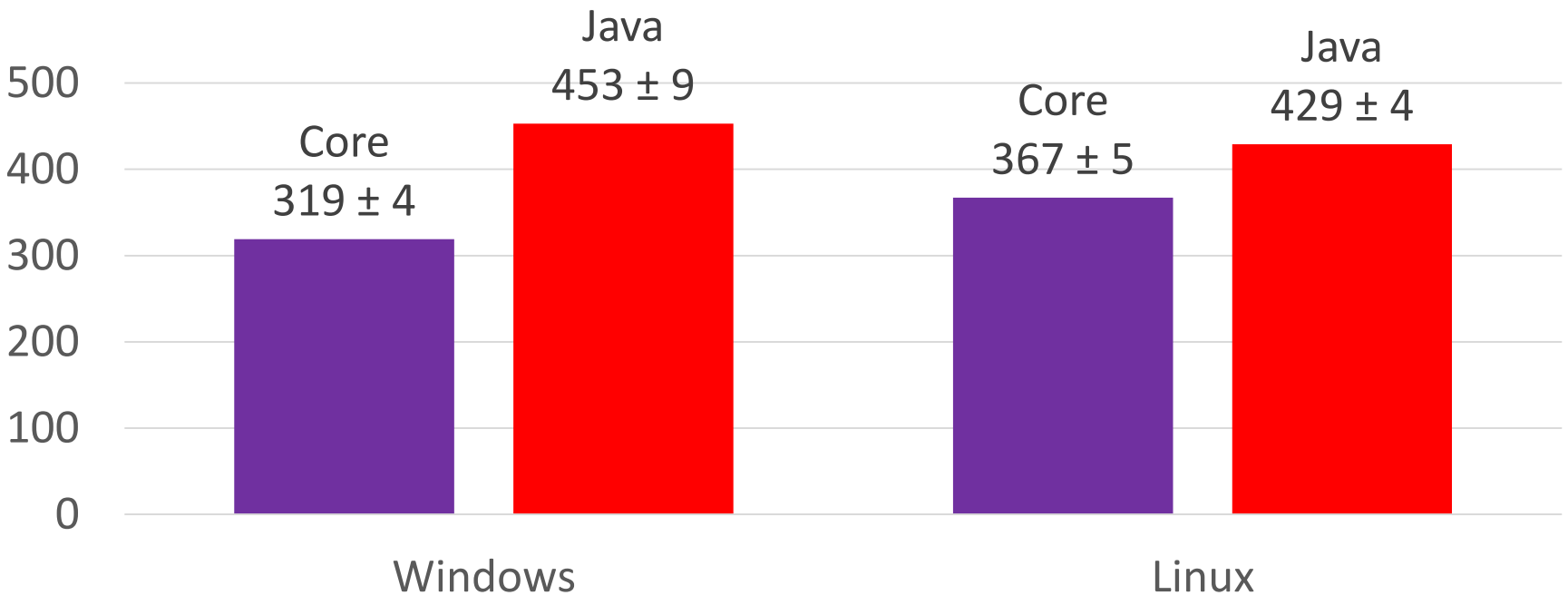
Сравним простейшую математику и генератор случайных чисел

- 10 000 000 итераций
- Сложение и умножение переменных: Integer, Long, Float, Double
- Random: Integer, Double, Byte[1024]
- Стандартная математика: возведение в степень, корень, арктангенс, округление.
- Комплексная математика, суперпозиция нескольких функций

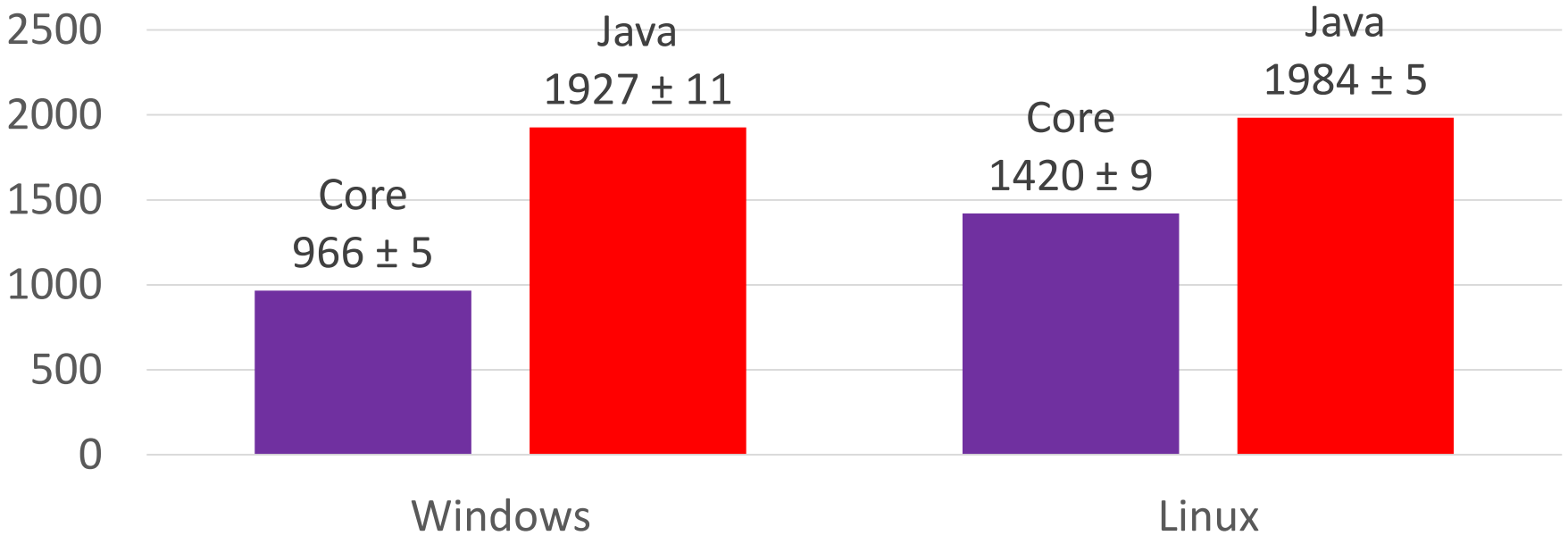
Переменные: время (мс)



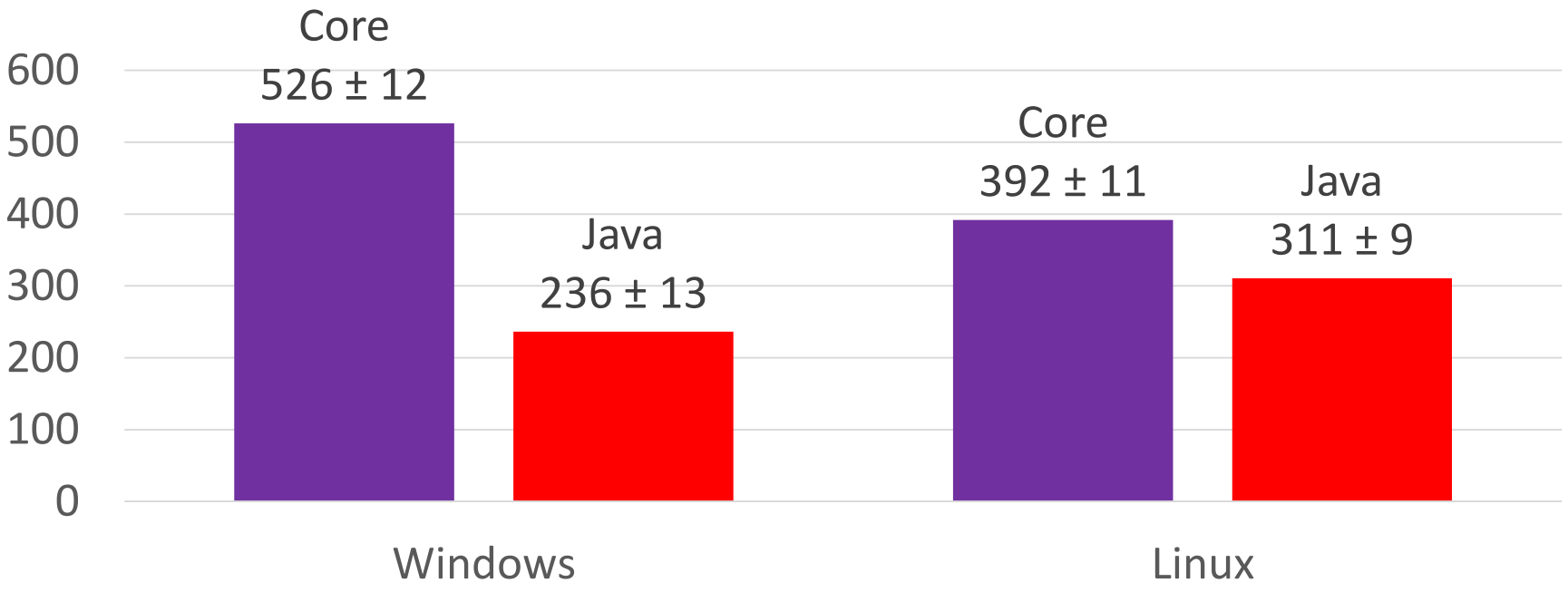
Random: время (мс)



Стандартная математика: время (мс)



Суперпозиция: время (мс)



Рекурсия

Сравним простую (хвостовую) рекурсию и глубокую рекурсию

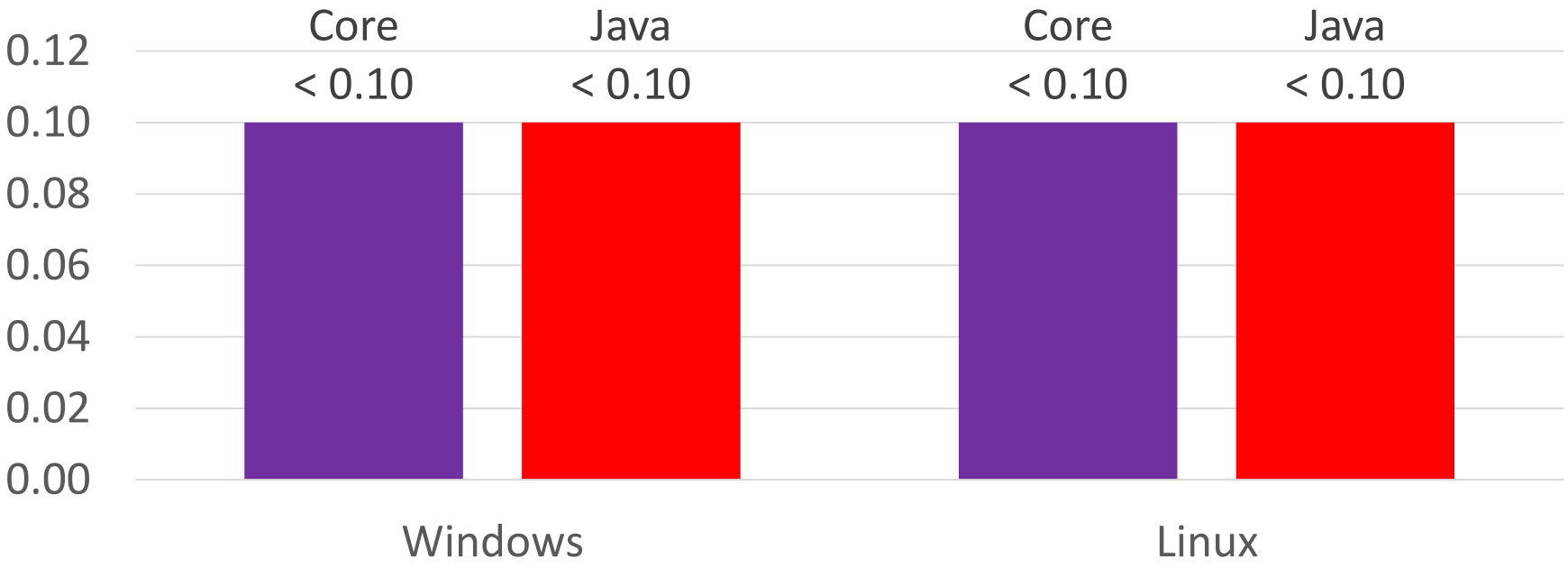
- Простая рекурсия, 10 000 вложений.
- Задача: игровое поле на **n** последовательных ячеек, кубик на 6 граней, бросаем кубик, идём на выпавшее число — сколько возможных вариантов завершения игры.
- Рекурсия Аккермана

Функция Аккермана

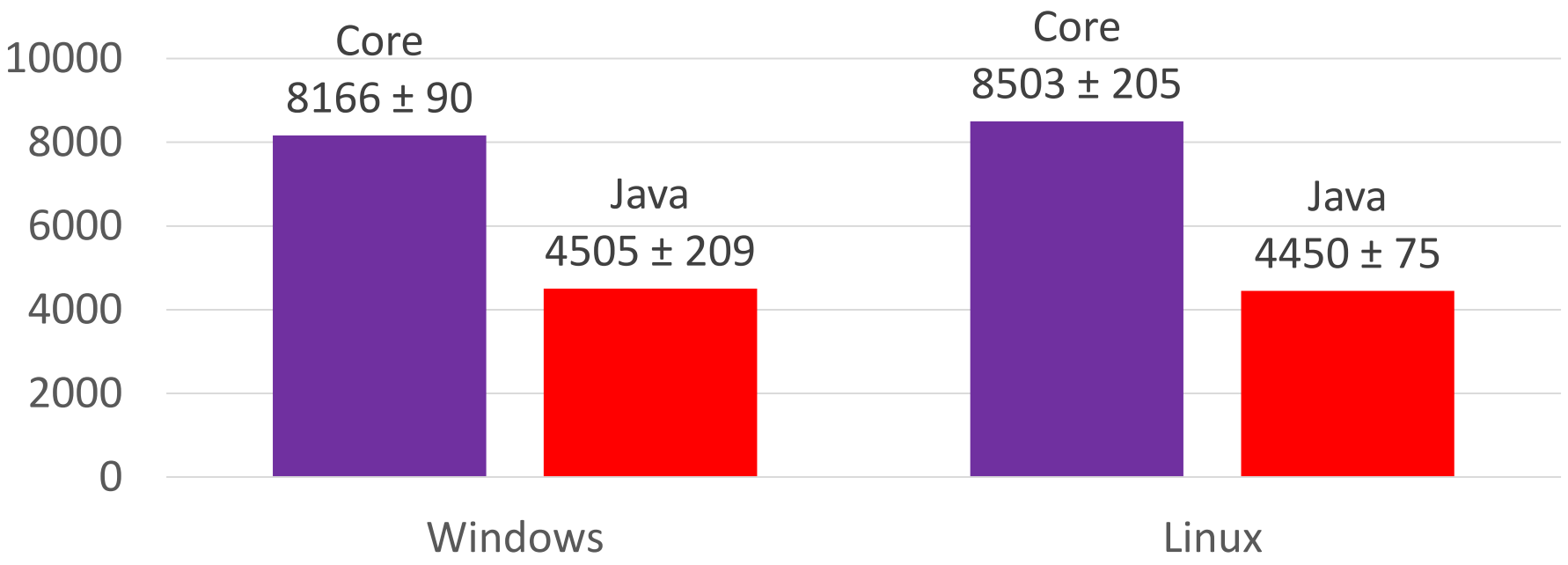
$$A(m,\ n) = \begin{cases} n+1, & m = 0; \\ A(m-1,\ 1), & m > 0,\ n = 0; \\ A(m-1,\ A(m,\ n-1)), & m > 0,\ n > 0. \end{cases}$$

```
long Ackermann(long m, long n) {
    if (m == 0) return n + 1;
    if (n == 0) return Ackermann(m - 1, 1);
    return Ackermann(m - 1, Ackermann(m, n - 1));
}
```

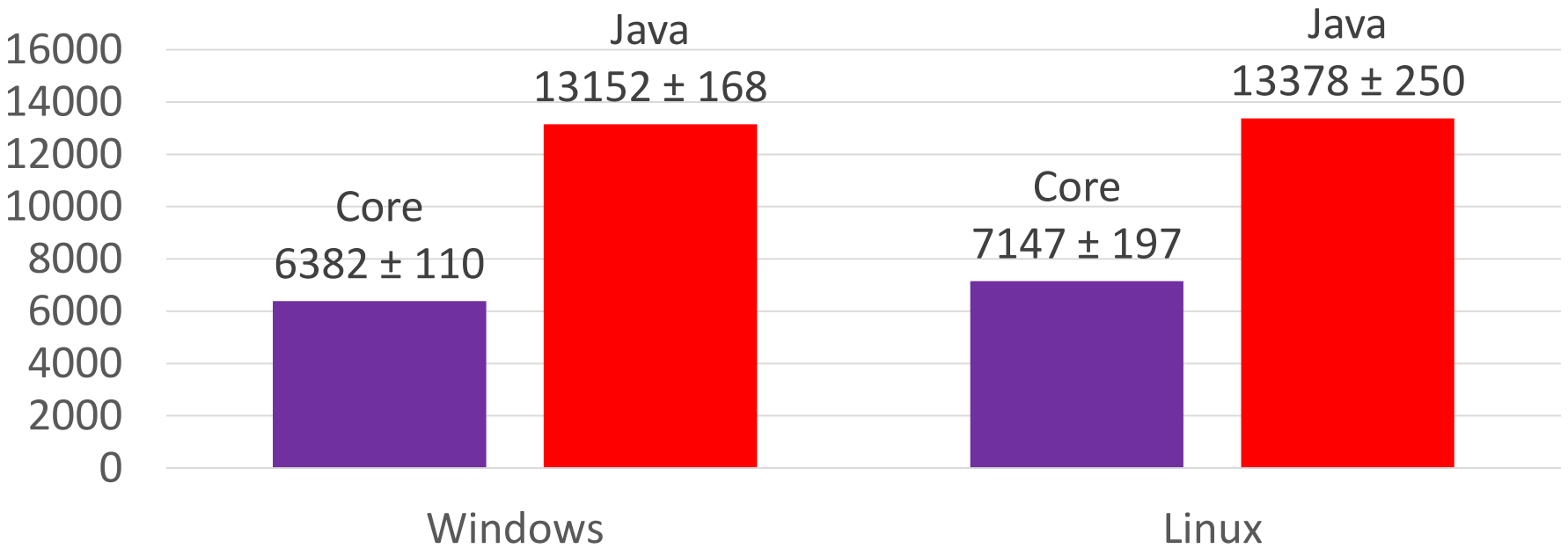
Простая рекурсия: время (мс)



Задача с полем и кубиком: время (мс)



Рекурсия Аккермана: время (мс)

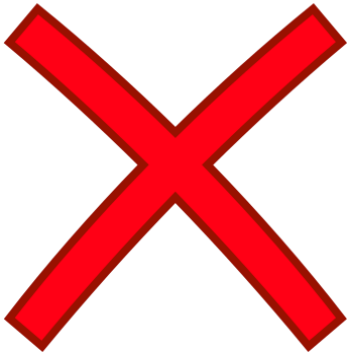


Регулярные выражения

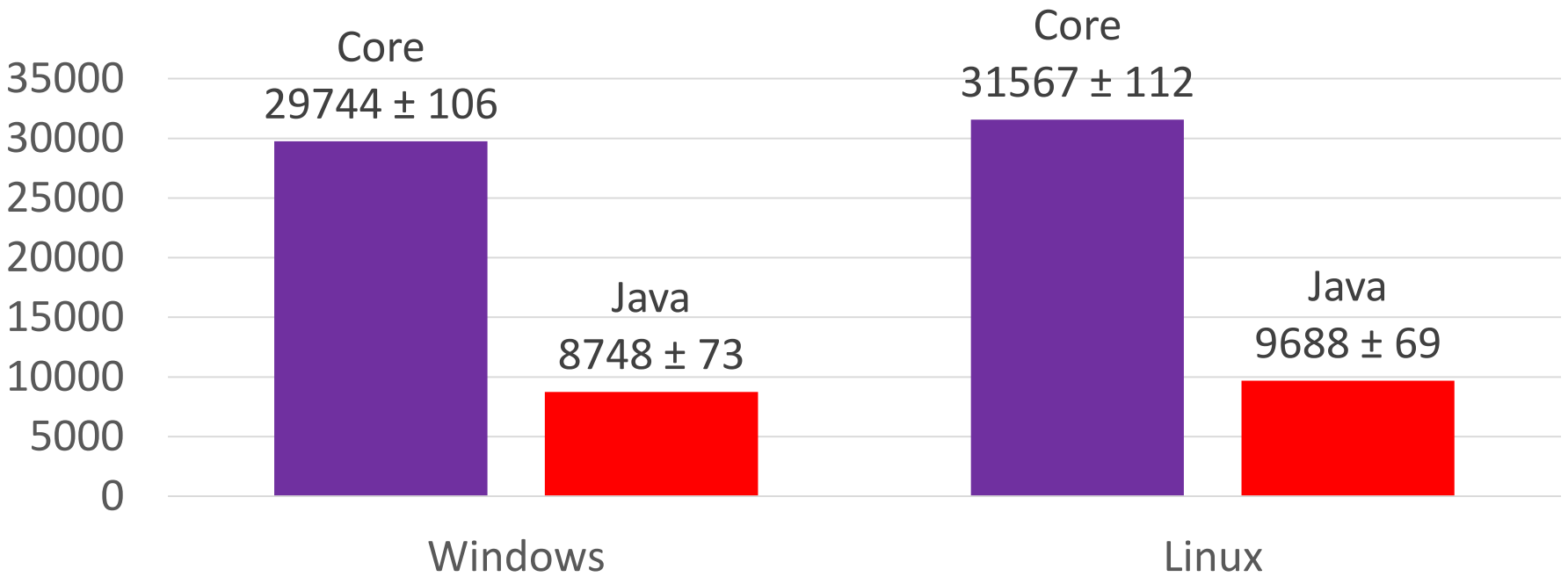
Сравним скорость регулярных выражений

Используем мануал для Linux объёмом 31мб

- URI — протокол://домен
- Email — имя@домен.домен
- Date — 00/00/0000
- URI или Email



Регулярные выражения: время (мс)



Коллекции

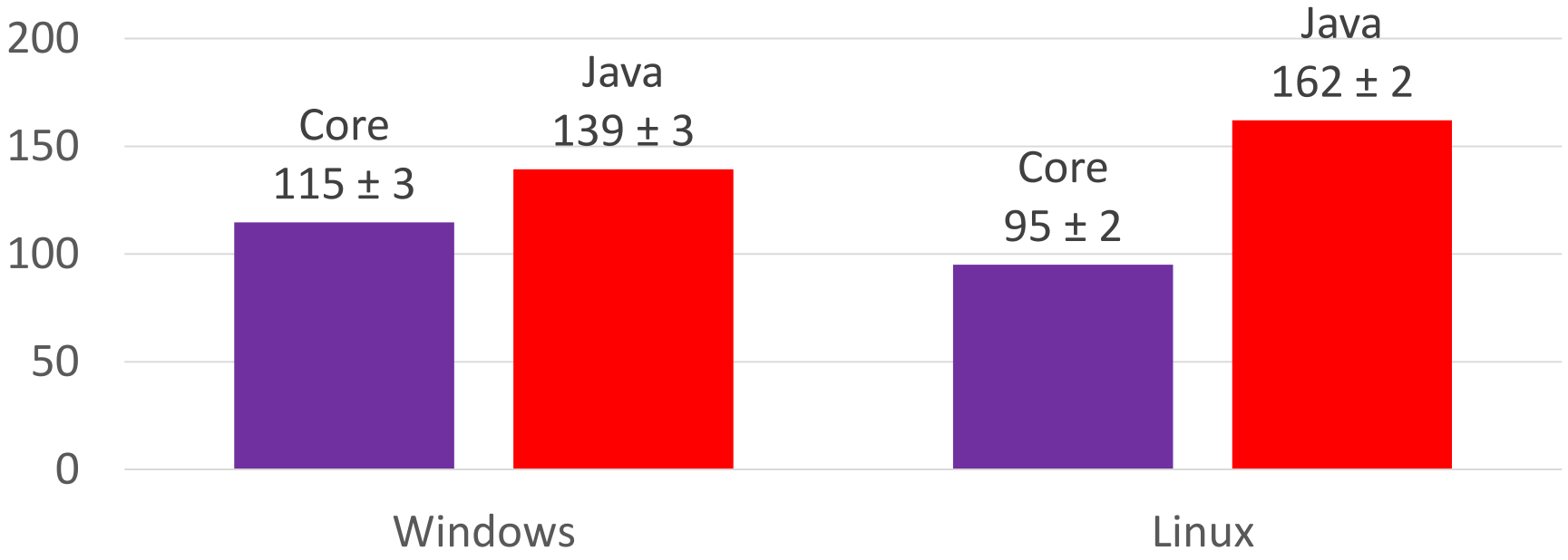
Массив

- 1 000 000 элементов
- Создание
- Реверс
- Изменение размера
- Перетасовка элементов

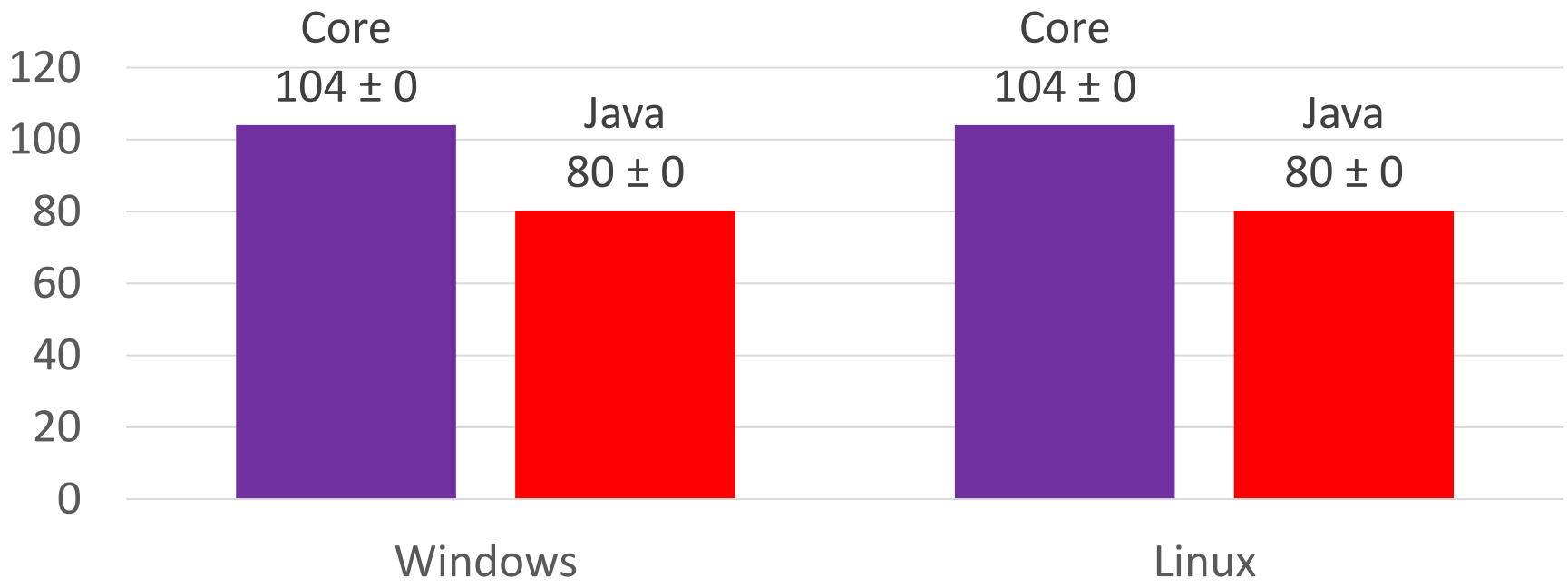
Dictionary и HashMap

- 1 000 000 элементов
- Создание
- Перетасовка элементов

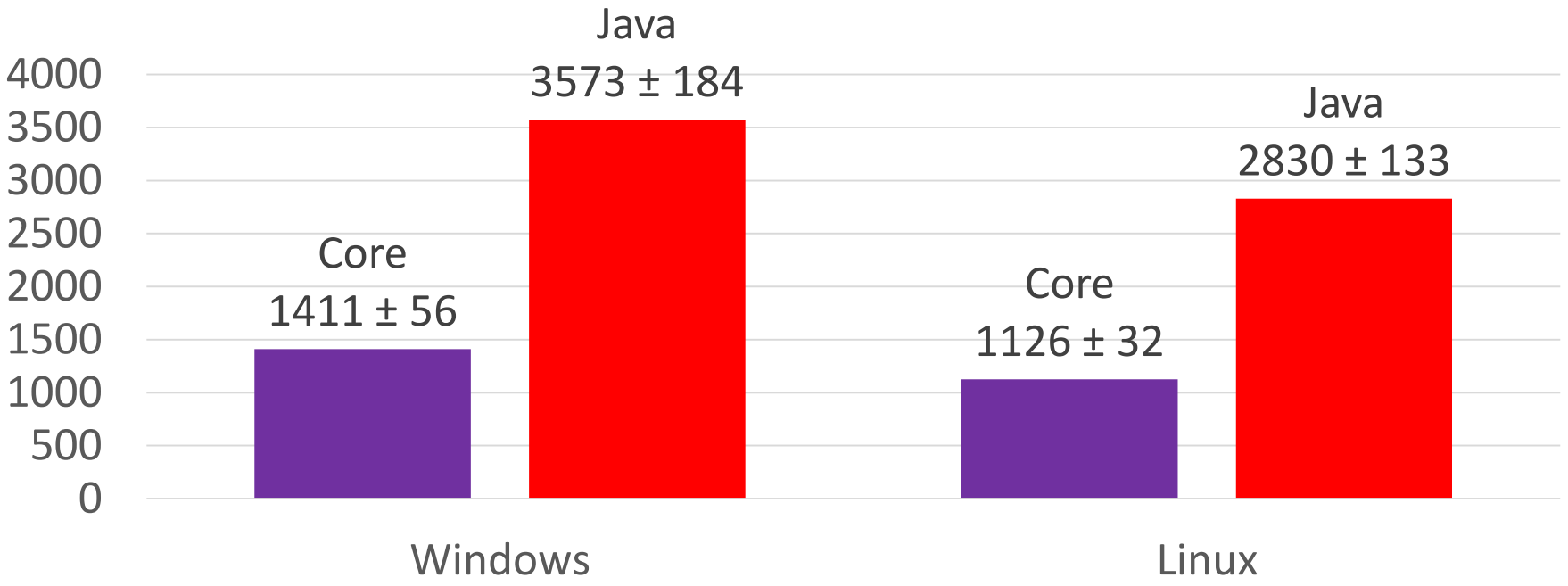
Массивы: время (мс)



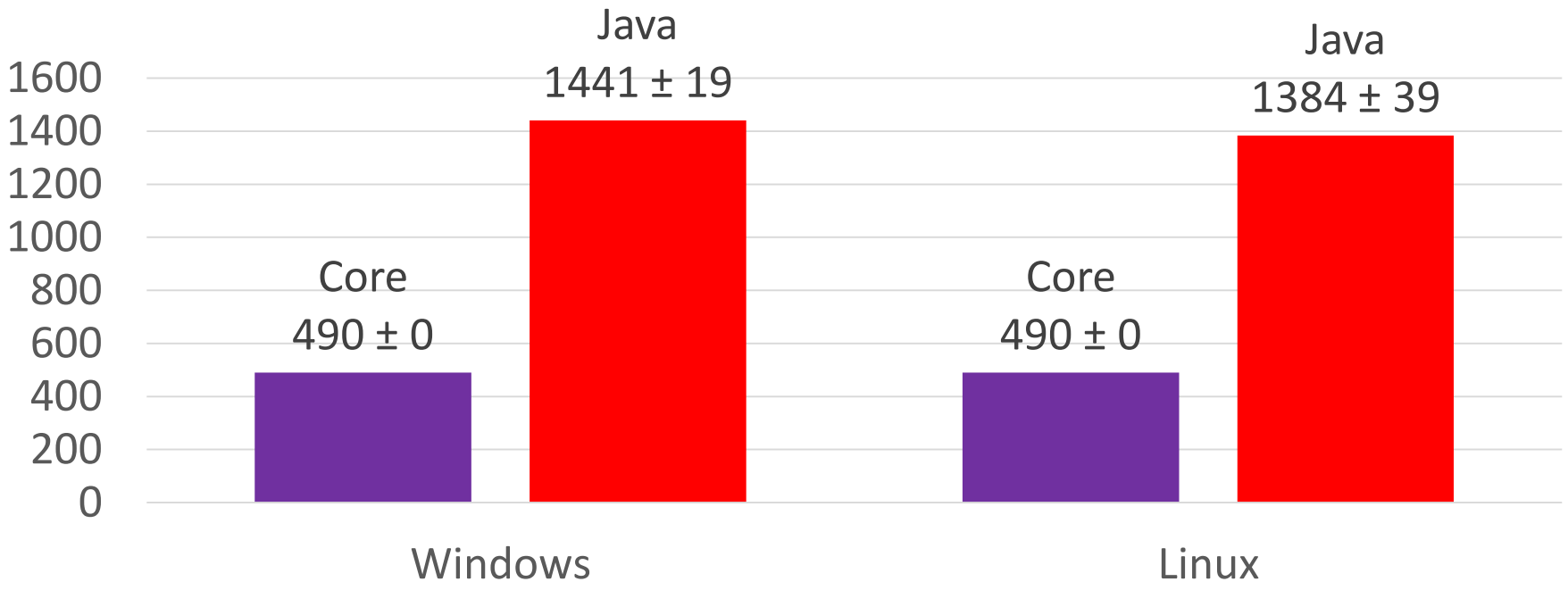
Массивы: память (мб)



Dictionary и HashMap: время (мс)



Dictionary и HashMap: память (мб)



Второй раунд окончен, Windows

	Время		Память	
	Core	Java	Core	Java
Переменные и математика	1315	1227		
Рекурсии	1272	1219		
Регулярные выражения	1000	3399		
Коллекции	1754	1000	1715	1138
Итого	1309	1500	1715	1138

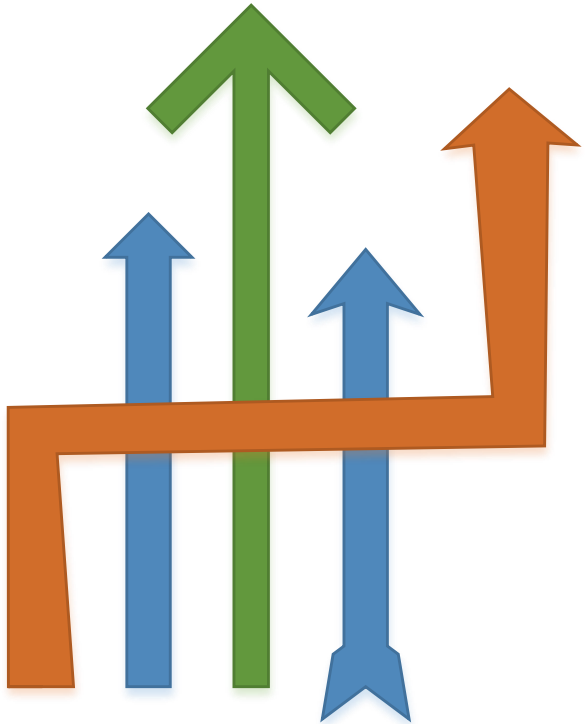
Второй раунд окончен, Linux

	Время		Память	
	Core	Java	Core	Java
Переменные и математика	1130	1128		
Рекурсии	1232	1240		
Регулярные выражения	1000	3258		
Коллекции	2069	1000	1681	1138
Итого	1303	1461	1681	1138

Потоки и задачи

Потоки и задачи

1. Thread
    - Синхронные
    - Асинхронные
2. Task
    - Синхронные
    - Асинхронные

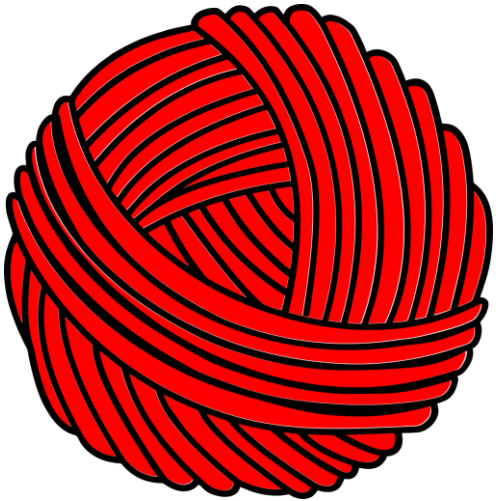


Thread

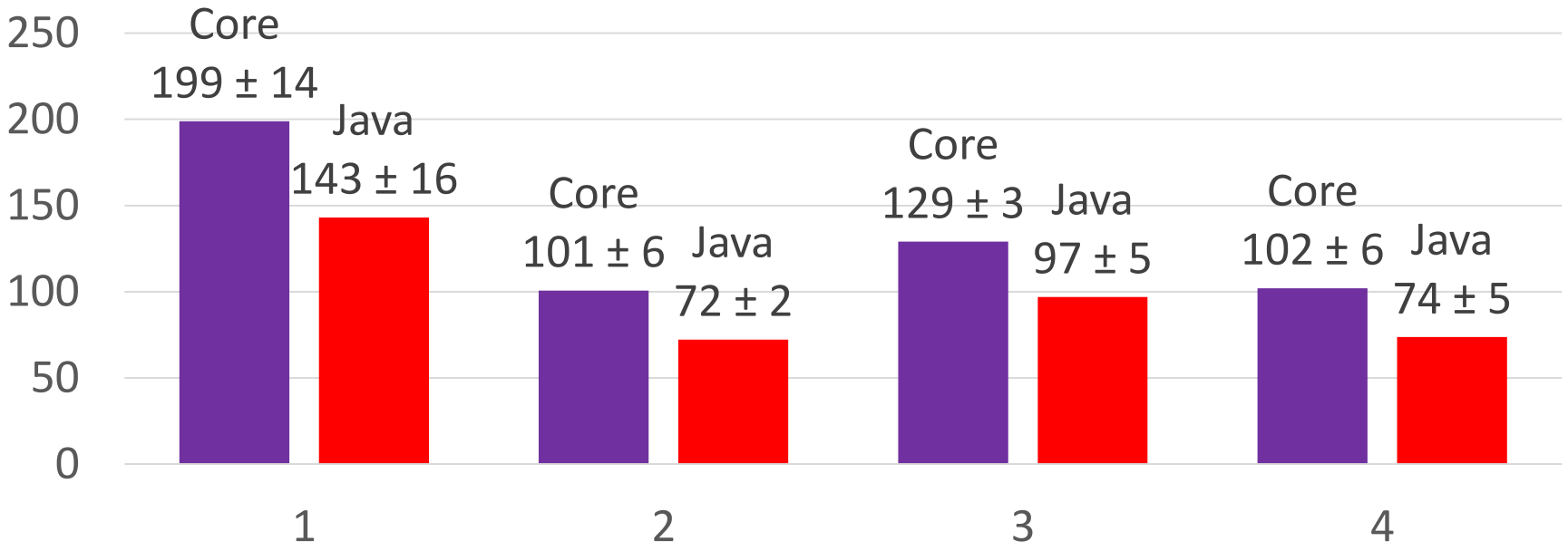
Сравним преимущество от использования потоков

- Создаём 2, 3 или 4 потока
- Каждый поток делает 4 типа вычислений
- Тест заканчивается, после всех вычислений
- 2 теста: синхронный и асинхронный

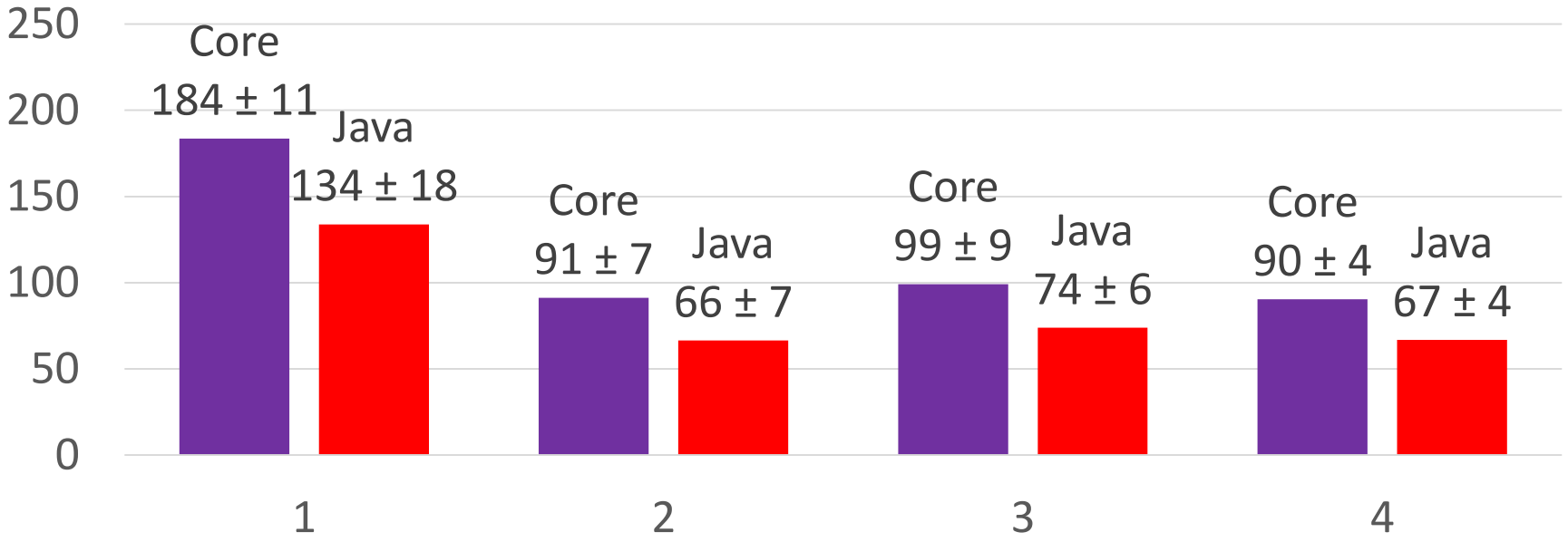
Синхронные потоки не могут выполнять одинаковые типы вычислений



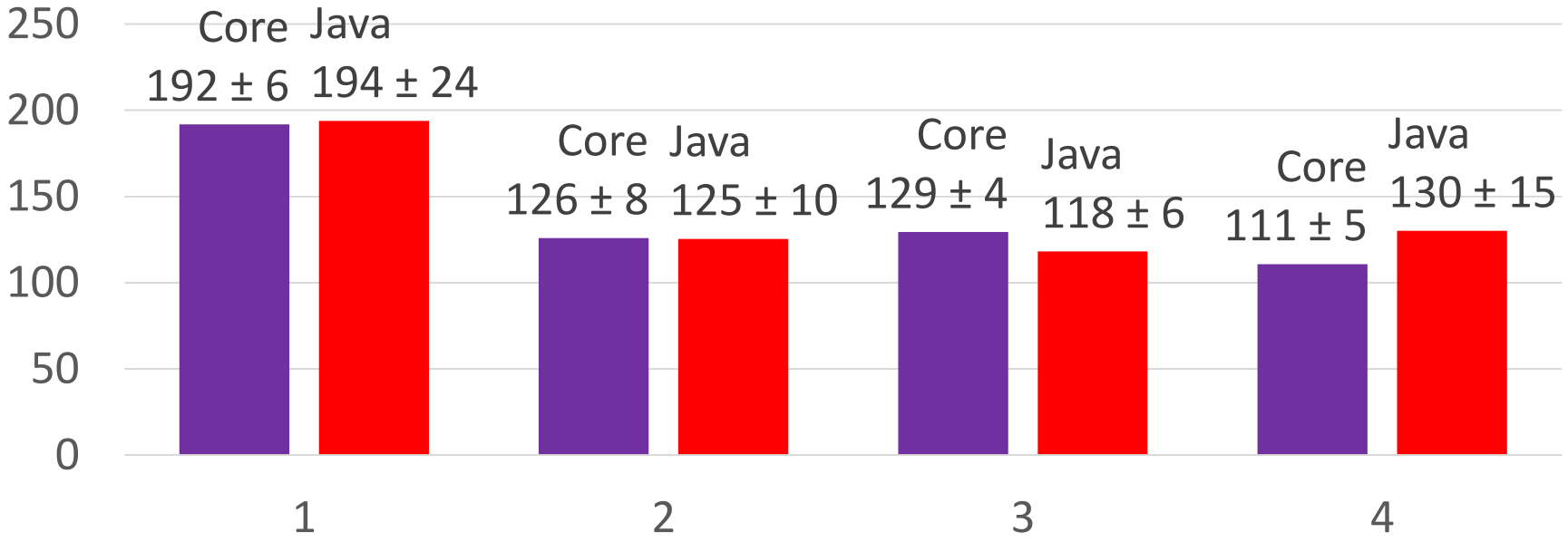
Thread, асинхронный, Windows: время (мс)



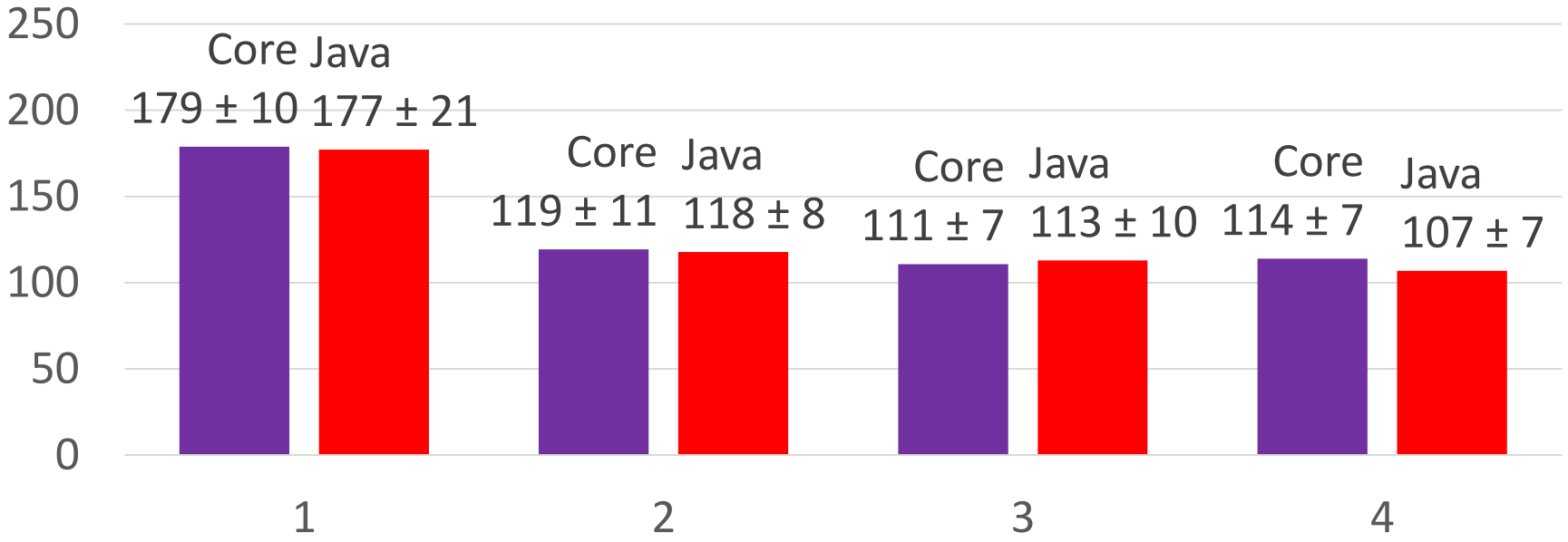
Thread, асинхронный, Linux: время (мс)



Thread, синхронный, Windows: время (мс)



Thread, синхронный, Linux: время (мс)



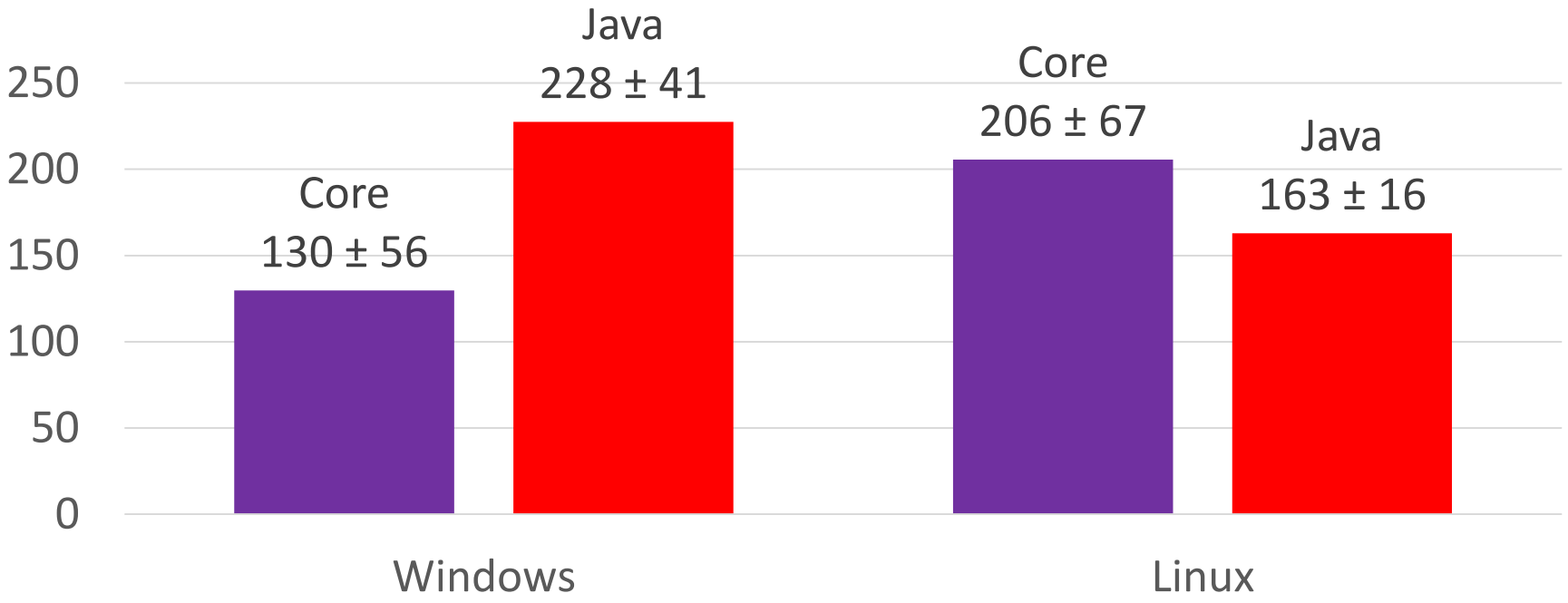
Task

Сравним пул потоков,

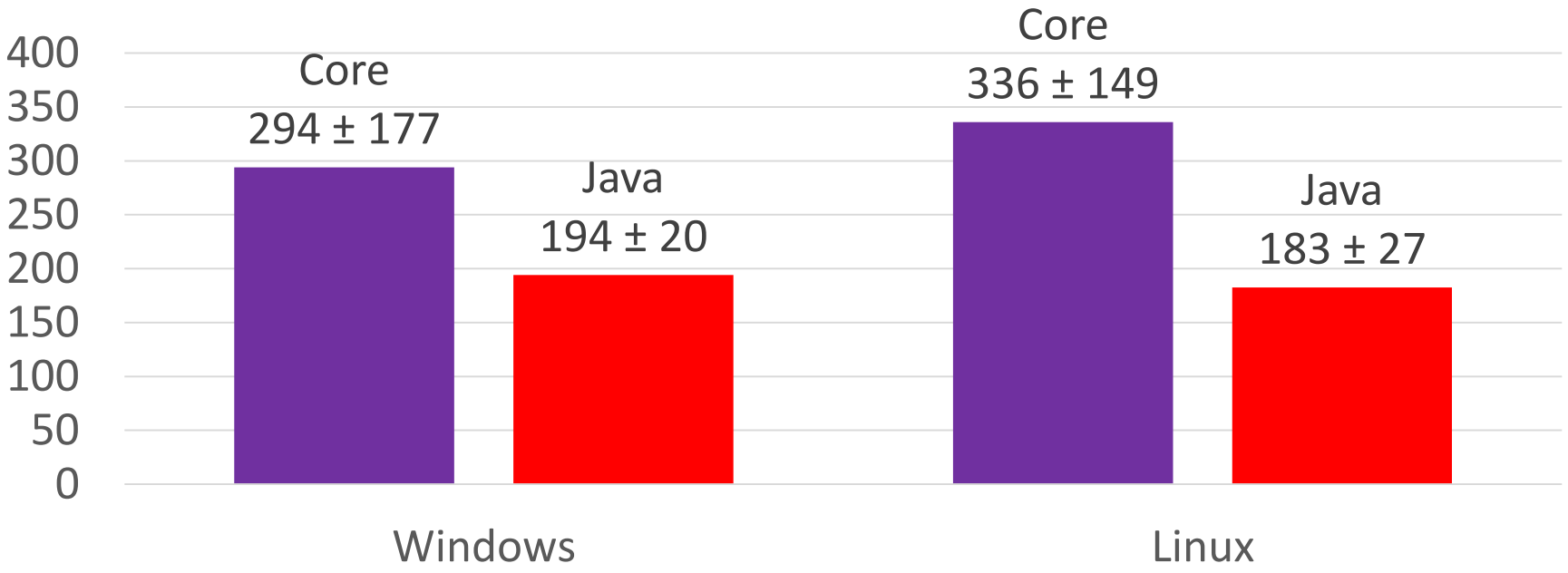
1. Создаём 500 000 задач
2. Каждая задача увеличивает переменную на единицу
3. Добавляем все задачи в общий пул, для Core – это Task, для Java – ExecutorService.

В синхронном тесте к переменной не могут обратиться две задачи, когда та кратна 3

Task, асинхронный тест, время (мс)



Task, синхронный тест, время (мс)



Третий раунд окончен

	Windows		Linux	
	Core	Java	Core	Java
Потоки, асинхронный тест	1000	1375	1000	1361
Потоки, синхронный тест	1044	1024	1005	1022
Задачи, асинхронный тест	1754	1000	1000	1262
Задачи, синхронный тест	1000	1514	1000	1839
Итого	1163	1208	1001	1340

Другие тесты и удобство использования

Другие тесты и удобство использования

Ещё бенчмарки:

1. Определитель матрицы
2. Игры с XML

Моё мнение по удобству!

Определитель матрицы

Применим знания из предыдущих бенчмарков и подтвердим результаты

Используется:

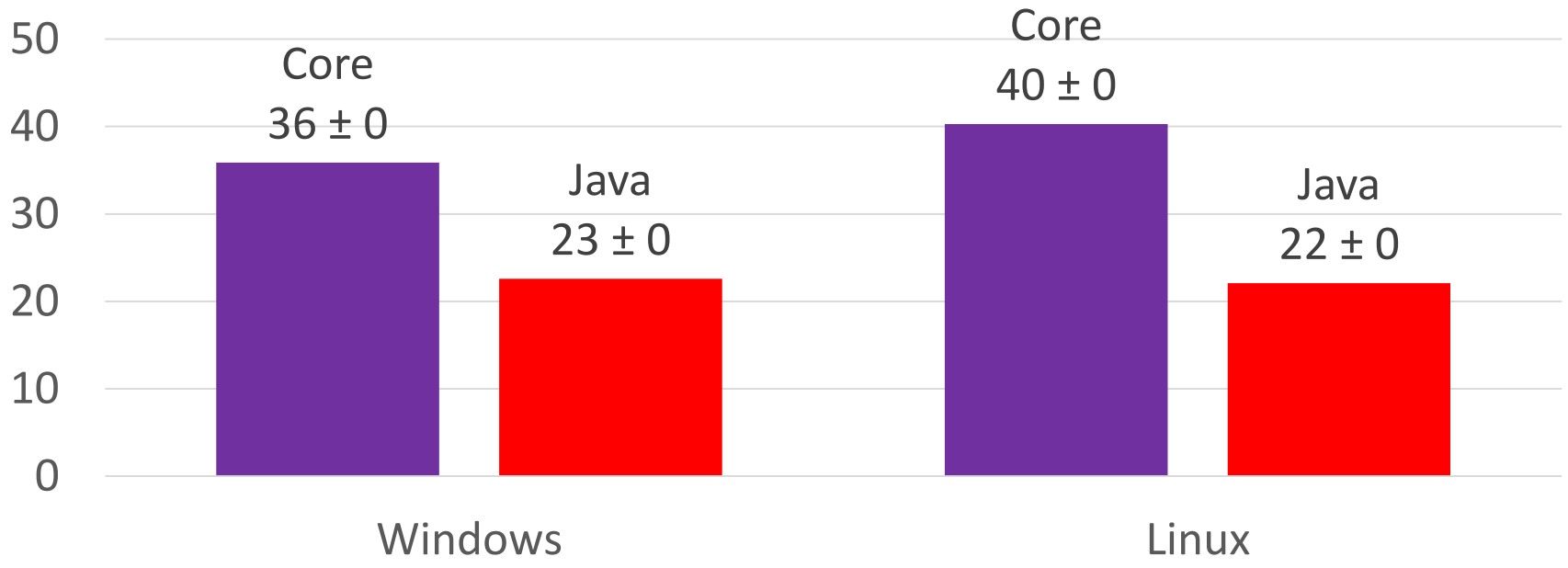
1. Рекурсия
2. Математика
3. Аллокация объектов
4. Массивы

Используем метод разложения по первой строке:

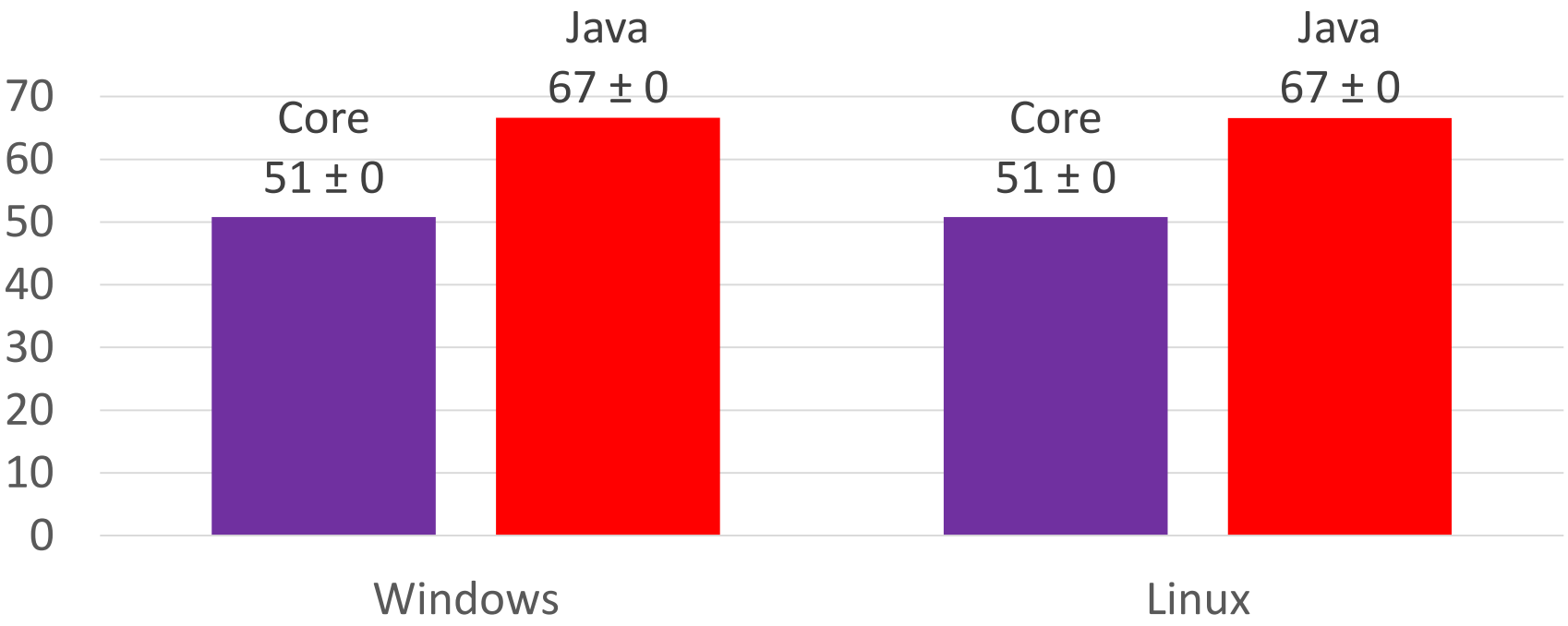
$$\Delta = \sum_{j=1}^{n} (-1)^{1+j} a_{1j} \bar{M}_j^1$$

Размер матрицы 6x6

Определитель матрицы, время (мс)



Определитель матрицы, память (мб)



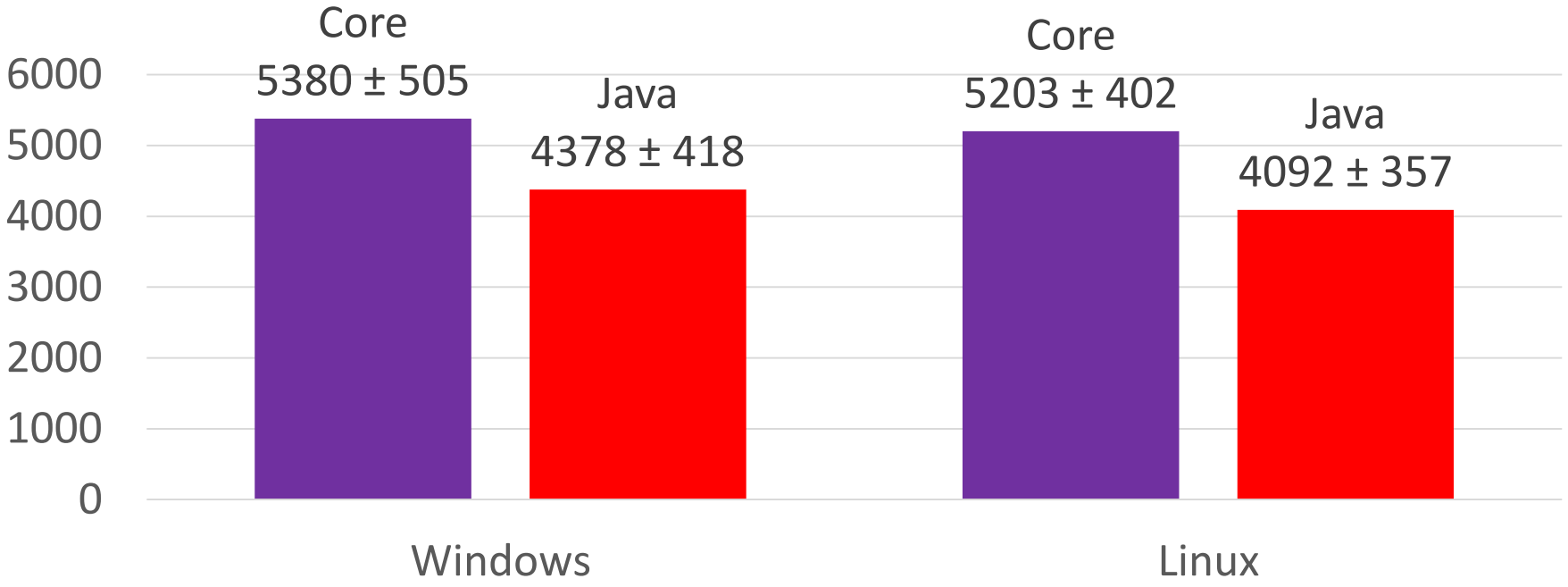
Игры с XML

Применим знания из предыдущих бенчмарков

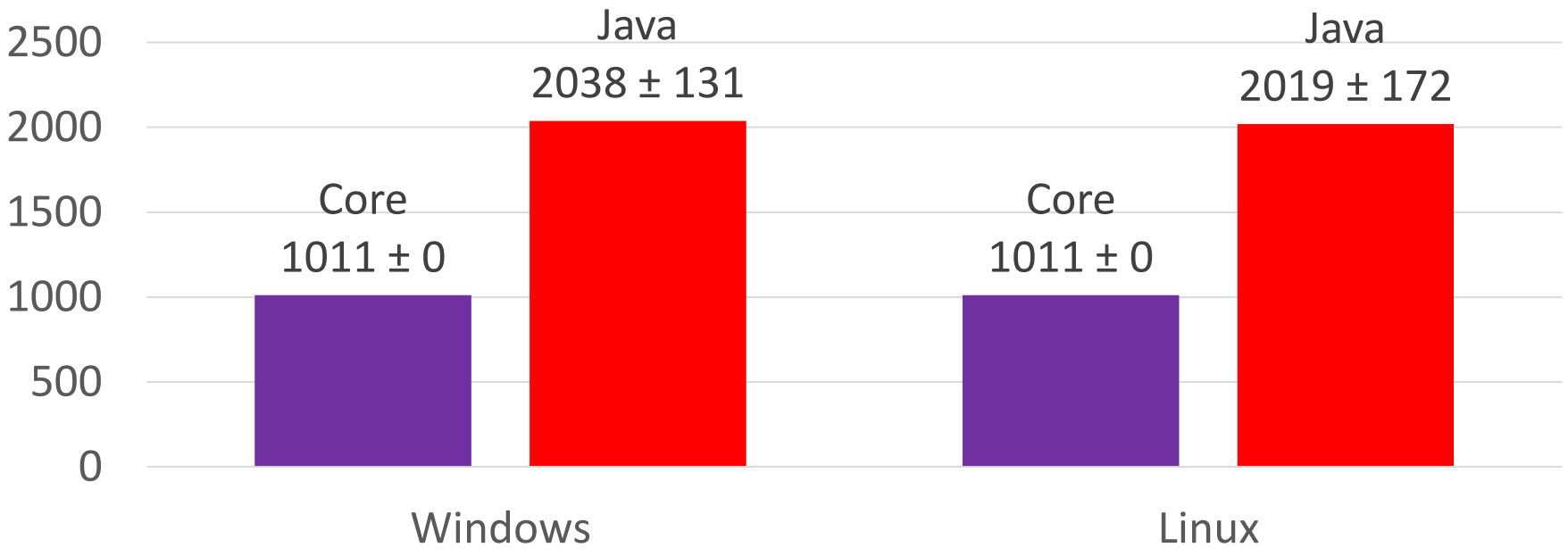
XML: 130 мб, простая структура, не более 10 вложений

- Парсинг XML в XmlDocument (для Java отключена валидация)
- Используется несколько GetElementsByTagName
- Аналитика текста в текстовых элементах с использованием Regex
- Результат работы записывается в Dictionary

Игры с XML, время (мс)



Игры с XML, память (мб)



Четвертый раунд окончен, Windows

	Время		Память	
	Core	Java	Core	Java
Определитель матрицы	1000	1588	1312	1000
Игры с XML	1000	1229	2015	1000
Итого	1000	1397	1626	1000

Четвертый раунд окончен, Linux

	Время		Память	
	Core	Java	Core	Java
Определитель матрицы	1000	1824	1310	1000
Игры с XML	1000	1271	1997	1000
Итого	1000	1523	1617	1000

Итоговый счёт

	Windows		Linux	
	Core	Java	Core	Java
Работа с памятью, время	1036	1400	1319	1151
Работа с памятью, память	1347	1050	1486	1050
Производительность, время	1309	1500	1303	1461
Производительность, память	1715	1138	1681	1138
Потоки и задачи	1163	1208	1001	1340
Другие тесты, время	1000	1397	1000	1523
Другие тесты, память	1626	1000	1617	1000
Итого	9196	8693	9407	8663

Подведем итоги!

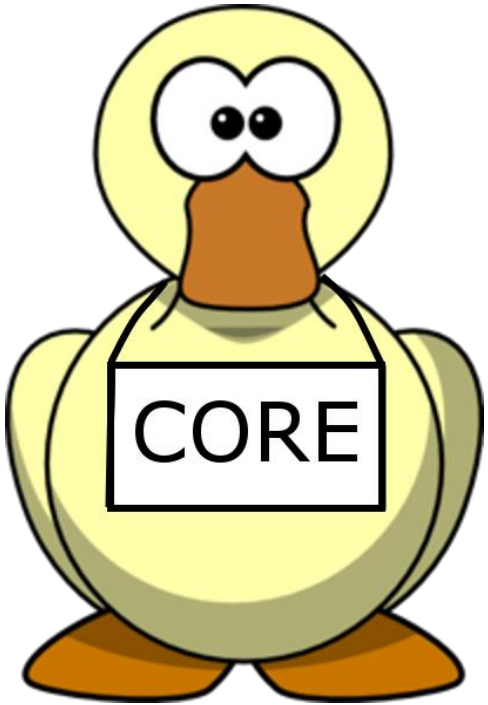
Core:

- Генерация документов
- Сервер с большим потоком клиентов

Java:

- OLAP
- Дата майнинг
- Игровые сервера

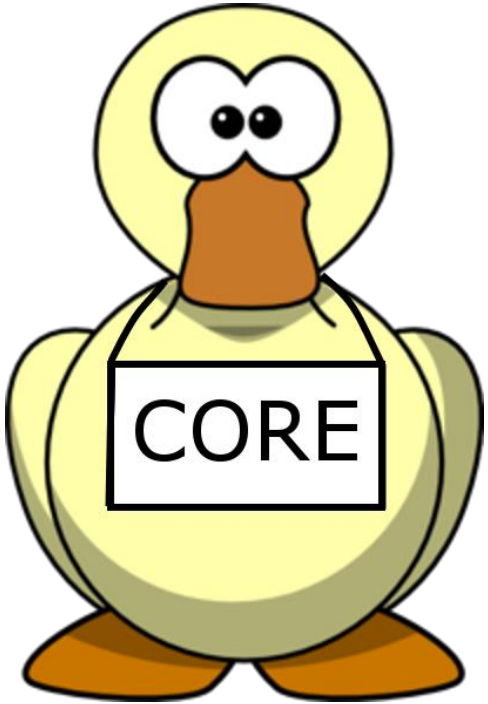
Подведем итоги!

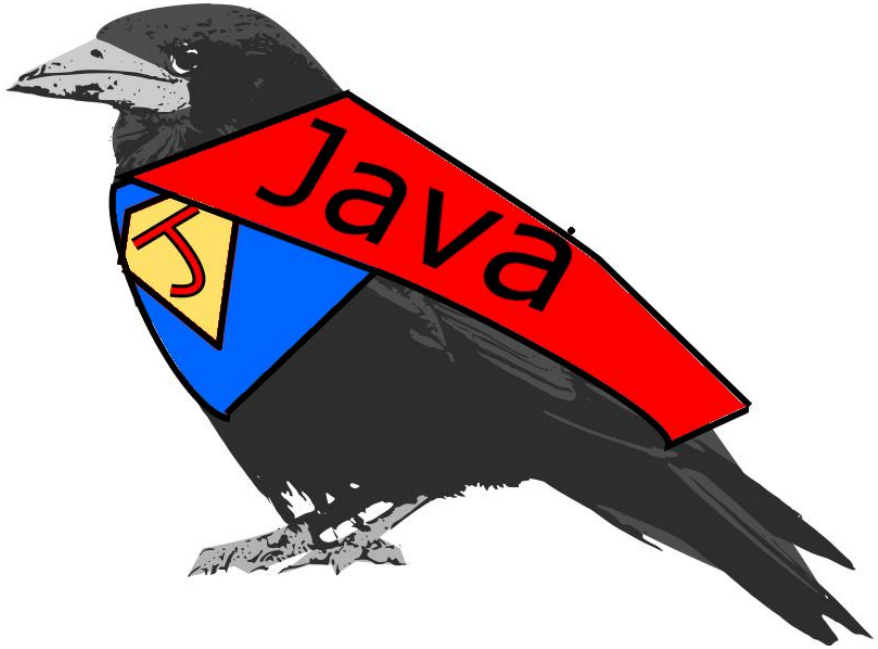


Java:

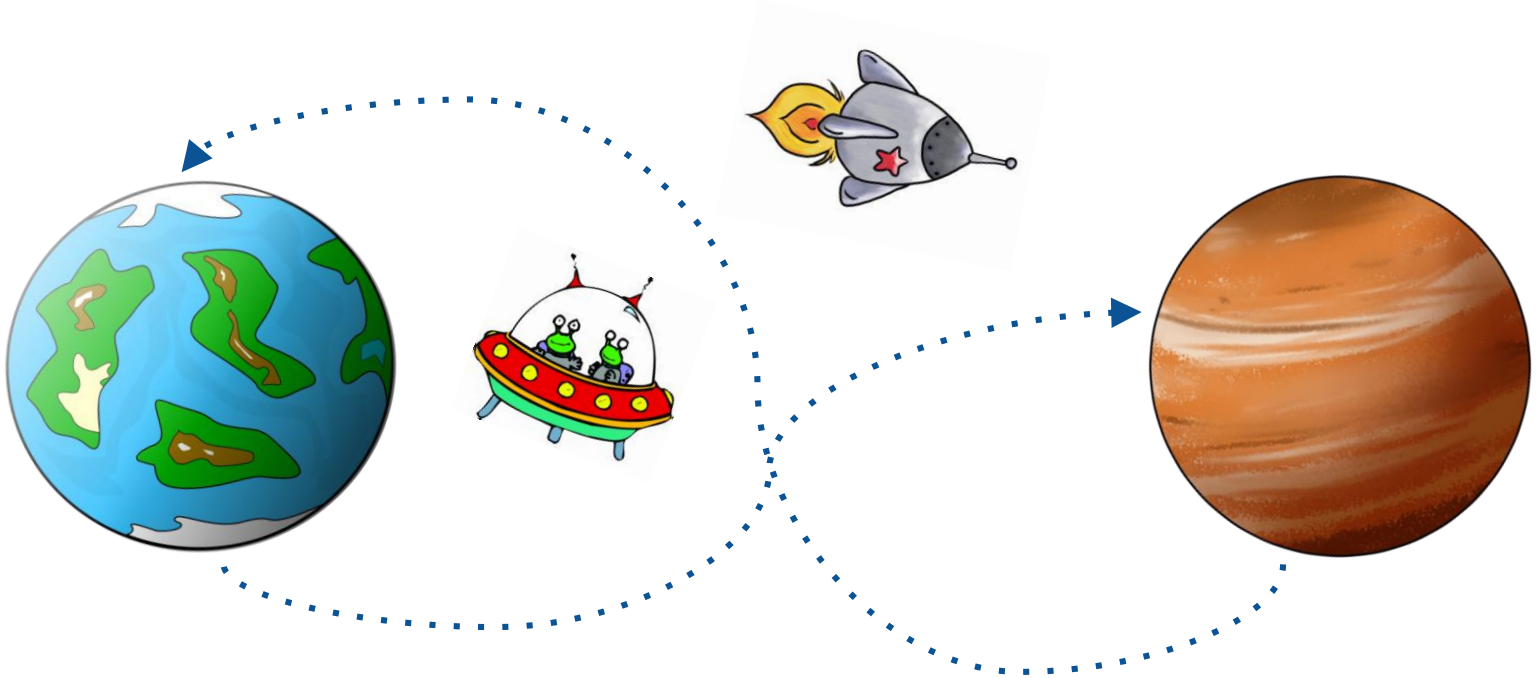
- OLAP
- Дата майнинг
- Игровые сервера

Подведем итоги!





Они так похожи...



Вопросы


Witaly_Ezepchuk


witaly@fast-report.com


https://github.com/detrav/dotnext2017

Виталий Езепчук
Программист Fast Reports
Автор DocumentSprint