

Быть в **10** раз  
эффективнее  
благодаря  
**Groovy**

Евгений  
Компаниец



# Smart1: система бронирования ТВ-рекламы

- Вся реклама на телеканалах 1+1, 2+2, ТЕТ, СІТІ продается через Smart1
- Месячный оборот 00 000 000 гр.
- Информация о 1 300 000 размещениях рекламы
- Сложная модель продаж - аукцион
- Отчеты
- Интеграция с внешними системами:  
GFK Mark Data Media Workstation, 1С
- 2 разработчика; 1,5 года; внедрено на втором месяце разработки

Период: **апреля 2011**  
 Статус: **Открыт**  
 Канал: **1+1**  
 Показатель: **Взвешенные GRP**

Нарpic **Шоплист** 68  
**A - Нарpic Hands+10 Max.tag v.1(16.10.2010) 30"** 68  
 Nurofen **Шоплист** 29  
**B - Nurofen** 29

Выбрать кампании

Стандарт	Низкий
-	68

НЕ ФИКС	1-я	3-я	ППосп
	-	-	-
	2-я	4-я	Посп
	-	-	-

Несколько копий кампании в блоке

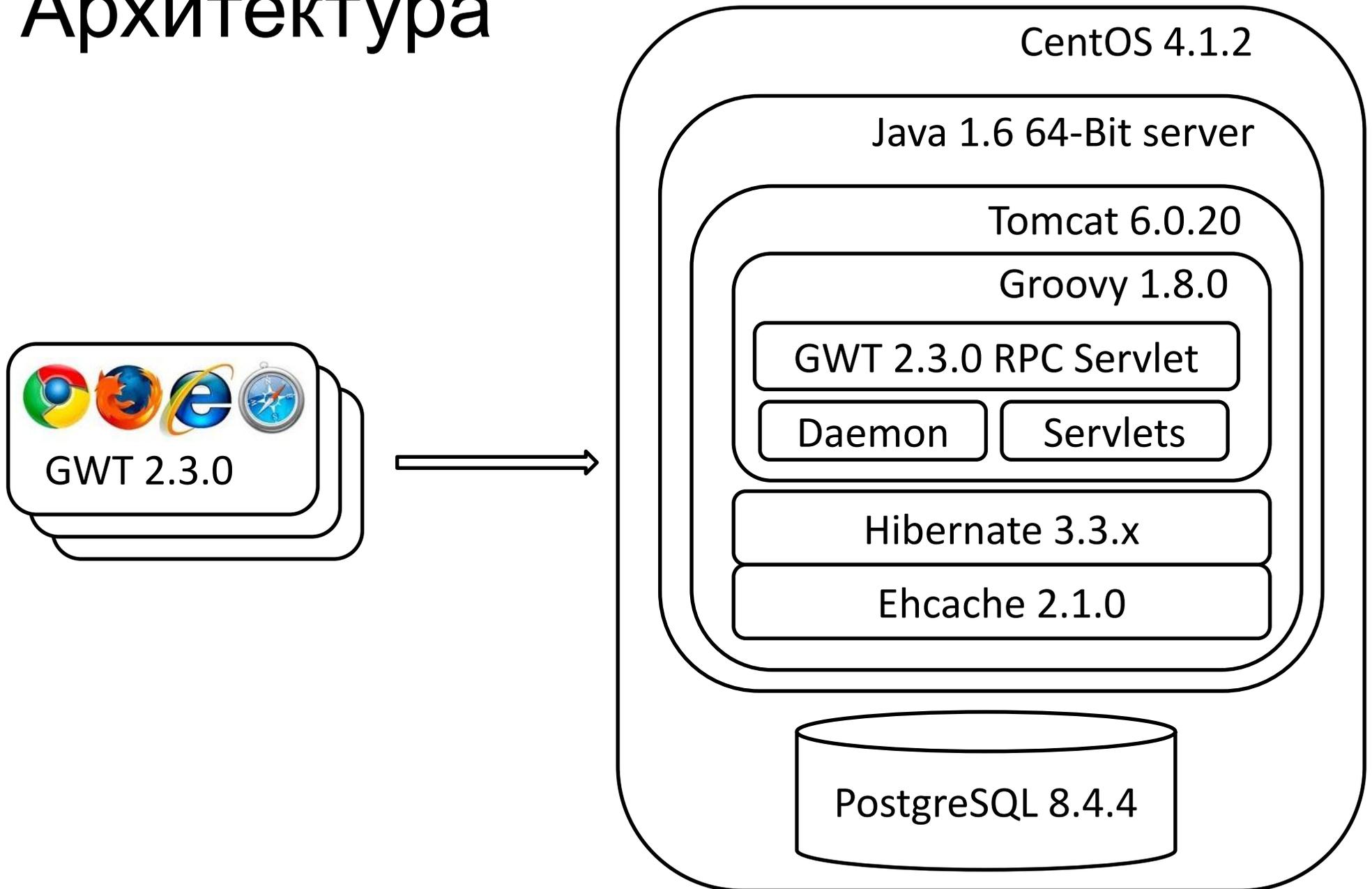
	Пн 25 Был в эфире	Вт 26 Закрыт	Ср 27 Закрыт	Чт 28 Заморожен	Пт 29 Заморожен	Сб 30 Открыт
			а			а
	16:25 ХФ Деловая женщина					16:40 ХФ Необычайные приключения Адель
17:00	17:00 внутри 0,71 а	17:30 Новости 17:43 после 1,06	17:00 Новости 17:15 Пр Шесть кадров 17:30 внутри 1,13	17:00 Новости 17:15 Пр Шесть кадров 17:30 внутри 0,92 а	17:00 Новости 17:15 Пр Шесть кадров 17:30 внутри 0,84 а	17:05 внутри 1,77
18:00	18:00 внутри 1,43	18:27 внутри 2,51	18:27 внутри 2,08	18:27 внутри 1,7 а	18:27 внутри 1,51	18:00 внутри 2,36
	18:30 Пр Суперняня	18:45 Пр Не ври мне	18:45 Пр Не ври мне	18:45 Пр Не ври мне	18:45 Пр Не ври мне	18:35 Пр Деньги
19:00	19:02 внутри 1,73	19:05 внутри 2,44 +5%	19:05 внутри 1,94 +5%	19:05 внутри 1,7 +5%	19:05 внутри 1,51 а +5%	19:05 внутри 2,29
	19:30 Новости	19:30 Новости	19:30 Новости	19:30 Новости	19:30 Новости	19:30 Новости
20:00	20:00 Пр Звезда + звезда 20:25 внутри 1,73 +5%	20:10 Пр Меняю жену 20:25 внутри 3,96	20:00 Пр Адская кухня 20:25 внутри 3,09 +5%	20:00 ХФ Кавказская пленница 20:25 внутри 2,9	20:15 Концерт Стаса Михайлова в Кремле 20:35 внутри 2,13	20:00 ХФ Я буду жить 20:16 внутри 2,16

- свободный
- проходит станд/низк
- в очереди станд/низк
- проходит другими станд/низк
- в очереди другими станд/низк
- телепередача
- день заморожен
- день закрыт

-> load: 2 rq/s, blocks: 0

ОСТАВЬ ОТЗЫВ

# Архитектура



# Разработка



IntelliJ IDEA 10



GIT



Maven 2



Jetbrains TeamCity 6.0.3



Selenium 1.0.2

- 710 Тестов
- Время сборки 40 мин
- Установка на рабочий сервер 2 раза в неделю

# Производительность

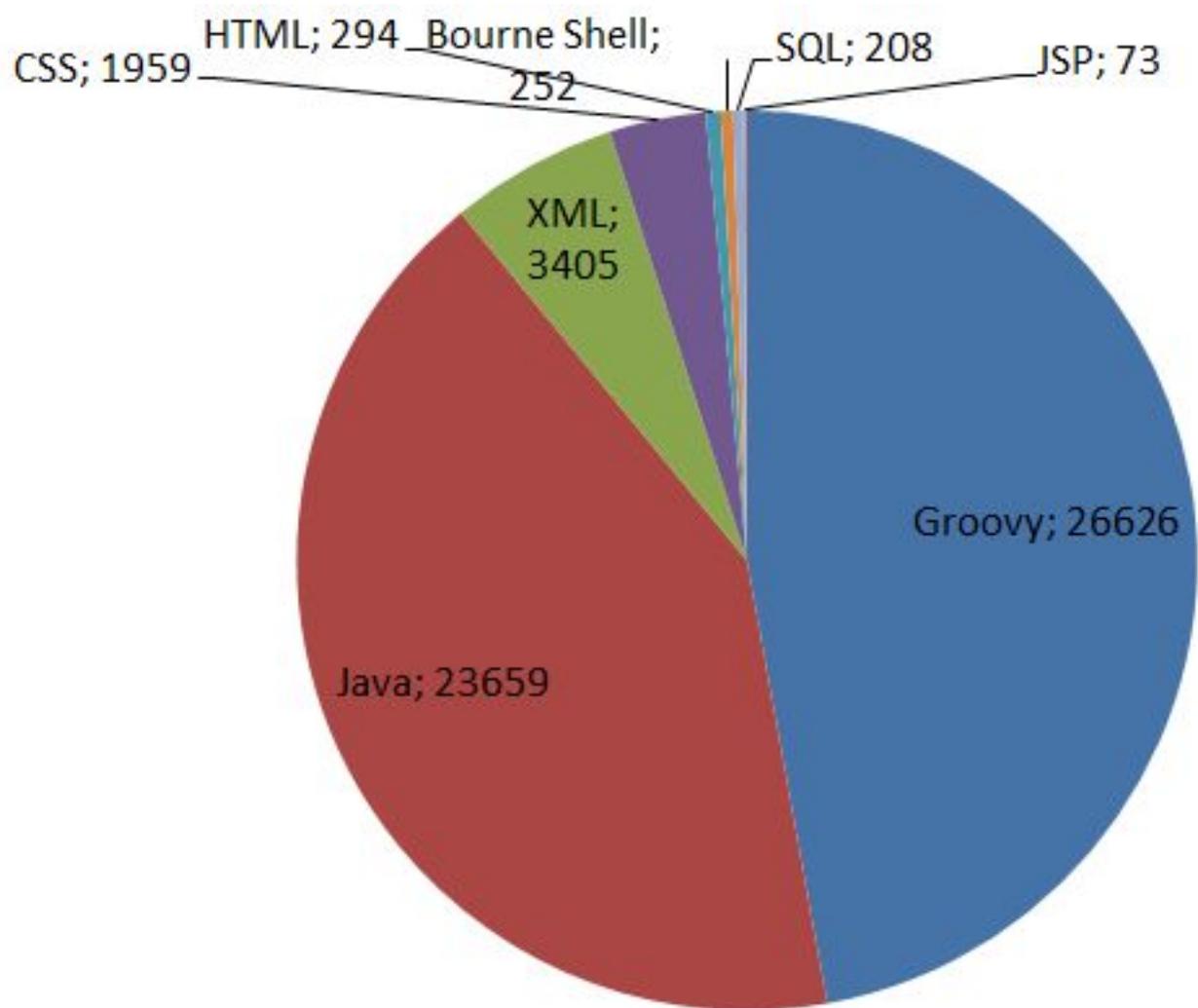
Пиковая нагрузка 40 gwt rps  
запросов в секунду

HP Proliant DL360G6  
2xQuad CPU  
12G RAM



- Денормализация структуры БД
- Тяжелые отчеты обновляются по расписанию
- Ряд задач выполняется только ночью

# Строки кода



# От Java к Groovy

- Smart1 - наш второй groovy проект
- До перехода сомнения:
  - что такого принципиального может дать groovy?
  - зачем терять часть возможностей IDE?
  - огромный тормоз
- После перехода:
  - сожаление, что gwt не позволяет использовать groovy, чтобы полностью отказаться от java

# Опрос: Насколько Groovy эффективнее Java?

- 4-6 раз, коллеги
- Я бы сказал 2-3 раза, Алекс Ткачман
- Я обычно продуктивнее в 2 с лишним. Иногда groovy действительно упрощает проблему и я становлюсь в 3-5 раз продуктивнее. Давид Кларк
- Моя продуктивность легко достигает 10 раз. Jochen Theodorou

# Groovy - это гораздо больше, чем убрать из Java ; и типы!

- значительно меньше кода
- код значительно читабельнее
- значительно выше повторное использование
- легко создаются DSL
- не нужен псевдокод

# Коротко и выразительно!

Взять все проходящие размещения и отсортировать сначала по цене, потом по дате создания

```
placements.findAll { it.booked }  
  .sort {p1, p2 ->  
    p2.wPrice <=> p1.wPrice ?:  
    p1.creationDate <=> p2.creationDate }
```

```
List bookedPlacements = new ArrayList();
for (Placement placement : placements) {
    if (placement.isBooked()) {
        bookedPlacements.add(placement);
    }
}
Collections.sort(bookedPlacements, new Comparator<Placement>() {
    public int compare(Placement p1, Placement p2) {
        int r = p1.getwPrice().compareTo(p2.getwPrice());
        if (r == 0) {
            r = p1.getCreationDate()
                .compareTo(p2.getCreationDate());
        }
        return r;
    }
});
```

# Коротко и выразительно!

Вернуть короткие названия бюджетных месяцев

```
def monthNames = budgets*.month*.shortName
```

```
List monthNames = new ArrayList();  
for (MonthBudget budget: budgets) {  
    monthNames.add(budget.getMonth().getShortName());  
}
```

# Коротко и выразительно!

Эфирное время конца программы – это время начала первого из послепрограммных блоков, либо время конца программы

```
blocks.findAll { it.position == AFTER }*.startTime.min() ?:  
                endTime
```

```
List afterBlocks = new ArrayList ();
for (Block block : blocks) {
    if (block.getPosition() == AFTER) {
        afterBlocks.add(block);
    }
}
if (afterBlocks.isEmpty()) {
    return endTime;
}
Time minTime = new Time(0);
for (Block block : afterBlocks) {
    if (block.getStartTime().isBefore(minTime)) {
        minTime = block.getStartTime();
    }
}
return minTime;
```

# Коротко и выразительно!

Если плательщик задан, то вернуть его, иначе взять плательщика из прошлого периода. Если в прошлом периоде нет плательщиков, то взять любого из агентства.

```
payee ? : prevInYear?.payee ? : (agency.payees as List)[0]
```

```
if (payee != null) {  
    return payee;  
}  
if (getPrevInYear() != null  
    && getPrevInYear().getPayee() != null) {  
    return prevInYear.getPayee();  
}  
return getAgency().getPayees().iterator().next();
```

# Немного сложнее?

Взять размещения из самой популярной категории

```
placements.groupBy { it.category }.collect {it}  
            .sort {it.value.size()}.last().value
```

## Java, с использованием «библиотечных» groupBy и last:

```
List groupsList = new ArrayList((Util.groupBy(placements,
    new GroupSelector<CopyCategory, Placement>() {
        public CopyCategory getProperty(Placement p) {
            return p.getCopyCategory();
        }
    })).entrySet());
Collections.sort(groupsList,
    new Comparator<Map.Entry<CopyCategory, List<Placement>>>() {
        public int compare(Map.Entry<CopyCategory, List<Placement>> o1,
            Map.Entry<CopyCategory, List<Placement>> o2) {
            return ((Integer) o1.getValue().size())
                .compareTo(o2.getValue().size());
        }
    });
return (List<Placement>) ((Map.Entry) Util.last(groupsList)).getValue();
```

## Java, прямая реализация:

```
Map<CopyCategory, List<Placement>> categoryPlacements = new
HashMap<CopyCategory, List<Placement>>();
for (Placement placement : placements) {
    List _placements = categoryPlacements.get(placement.getCategory());
    if (_placements == null) {
        _placements = new ArrayList();
        categoryPlacements.put(placement.getCategory(), _placements);
    }
    _placements.add(placement);
}
CopyCategory popularCategory = null;
int maxSize = 0;
for (CopyCategory category : categoryPlacements.keySet()) {
    if (categoryPlacements.get(category).size() > maxSize) {
        maxSize = categoryPlacements.get(category).size();
        popularCategory = category;
    }
}
return categoryPlacements.get(popularCategory);
```

# Сила Closure

Настоящие возможности открываются, когда мы понимаем что такое Closure

sort, findAll, groupBy и т.п – все навсего методы принимающие Closure и мы можем делать такие свои

# Сила Closure

Получить Map время, на название (названия уникальны для времени)

```
placements.mapUnique('time') { it.name }
```

```
Map<Time, String> timePlacements =  
    new HashMap<Time, String>();  
for (GfkPlacement placement: placements) {  
    timePlacements.put(placement.time, placement.name);  
}
```

# Расширение существующих классов

Мы можем добавлять методы и поля к уже написанным классам без наследования.

- Наш `mapUnique` можно вызывать на любой коллекции
- `robot.grp = 22.cent`
- `scheduleMonth.month = 2009.jan`
- `block.startTime = /17:59/.time`

# Расширение существующих классов

Методы у Object дают нам следующий синтаксис:

```
transaction {  
    new User( 'New User' ).dbStore()  
}
```

# Расширение существующих классов

Сделаем немного удобнее Hibernate Criteria API:

```
def clientGroups =  
  RobotGroup.dbQuery.with {  
    eq('deleted', false)  
    createCriteria('copy').eq('_client', client)  
  }.list()
```

# DSL делается легко

```
count = 0
new SwingBuilder().edt {
    frame(title: 'Frame', size: [300, 300], show: true) {
        BorderLayout()
        textlabel = label(text: "Click the button!",
            constraints: NORTH)
        button(text: 'Click Me',
            actionPerformed: {
                count++
                textlabel.text = "Clicked ${count} time(s)."
            }, constraints: SOUTH)
    }
}
```

```
JFrame frame = new JFrame("Frame");
frame.setSize(300, 300);
frame.setLayout(new BorderLayout());
final JLabel label = new JLabel("Click the button");
frame.add(label, NORTH);
final JButton button = new JButton("Click Me");
final int[] counter = {0};
button.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        counter[0]++;
        label.setText("Clicked " + counter[0] + " time(s).");
    }
});
frame.add(button, SOUTH);
frame.setVisible(true);
```

# DSL делается легко

```
shopList(client, 2011, 'Test ShopList', primePercent: 70) {  
    discount(offPrime: -10.disc)  
    channelSl(channel, lowDiscount: -30.disc) {  
        monthSl(JAN, seasonal: -30.disc) {  
            stdSl(100.centi)  
            lowSl(400.centi)  
        }  
    }  
}
```

# Selenium junit TECT

```
void testPlace_SpotClient() {
    runTest(
        agent, {
            chooseCampaigns 'XC'
            placeClick getBlock(1), 'XC'
            waitForError 'No Shop List for Volvo in 2009'
        },
        sale, clients, {
            changeToPerSpot 'Volvo'
        },
        ...
    )
}
```

# Динамика



Динамическое программирование позволяет нам понять что такое повторное использование по настоящему!

Например давайте перестанем каждый раз делать одно и тоже для Bidirectional Association и Lazy Initialization:

# Bidirectional Association

```
class Program {
    @OneToMany (mappedBy = "_program")
    Set<Block> _blocks = []
}
class Block {
    @ManyToOne
    Program _program
}
```

# Bidirectional Association

И теперь мы сразу можем работать:

```
def p = new Program()
def b = new Block()
p << b
p.addBlock(b)
p.removeBlock(b)

//valid properties: p.blocks (unmodifiable set), b.program
```

# Bidirectional Association

**Этого писать не нужно:**

```
class Program {  
    ...  
    void addBlock(Block b) {  
        b._program = this  
    }  
  
    void removeBlock(Block b) {  
        b._program = null  
    }  
}
```

```
class Block {  
    ...  
    void setProgram(Program p) {  
        if (_program != null) {  
            _program.friendBlocks  
                .remove(this)  
        }  
        _program = p  
        if (_program != null) {  
            _program.friendBlocks  
                .add(this)  
        }  
    }  
}
```

# Lazy initialization

```
class MonthBudget {  
    ...  
    Centi __actualBudget() {  
        ... calculation  
    }  
}
```

**Этого писать не нужно:**

```
class MonthBudget {  
    ...  
    Centi _actualBudget  
  
    def getActualBudget() {  
        if (_actualBudget == null) {  
            _actualBudget = ...  
        }  
    }  
}
```

```
println new MonthBudget().actualBudget
```

# Но не все так хорошо

- Скорость?
- IDE?

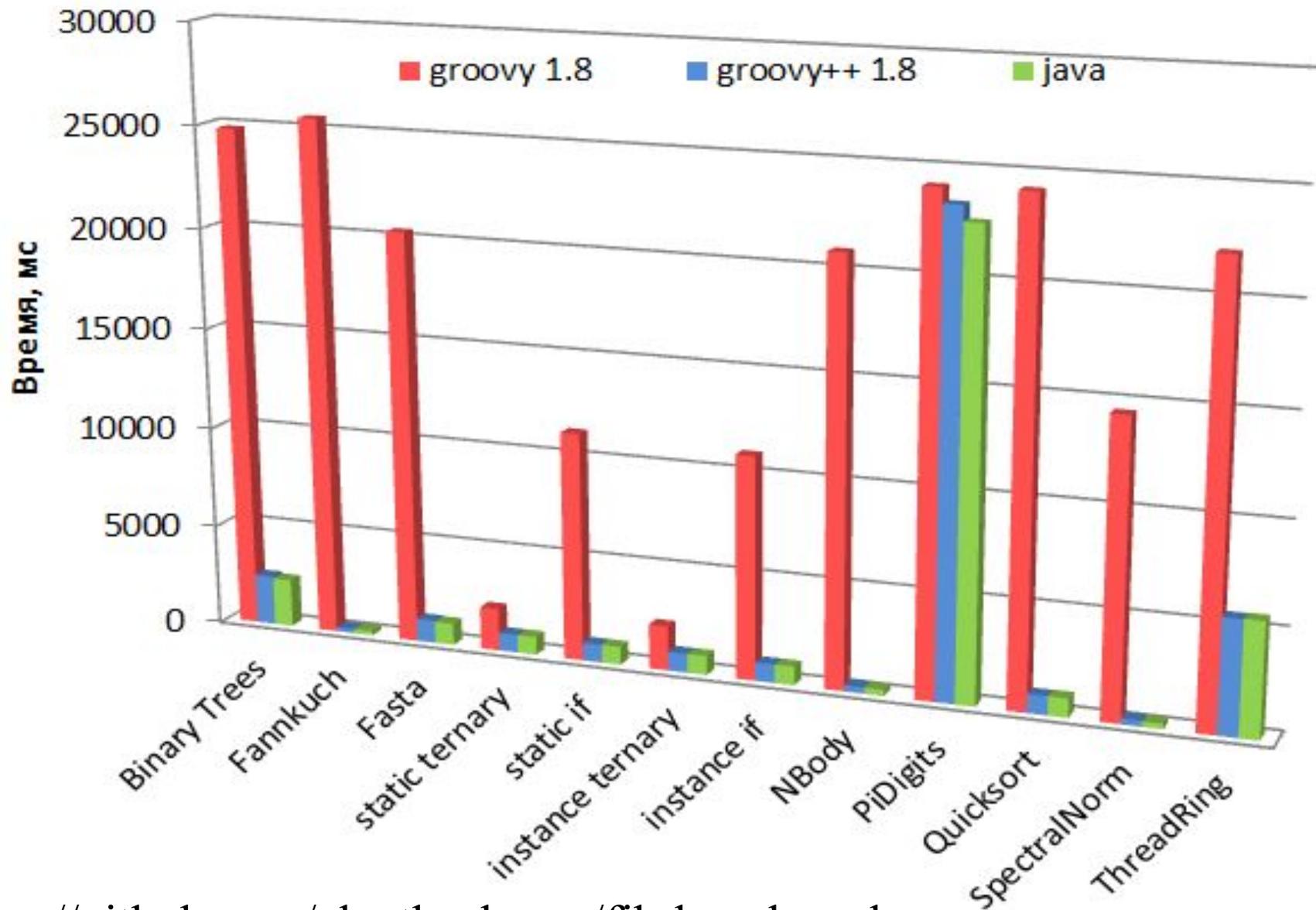


# Реально тормоз!



Groovy работает в 10 раз медленнее Java

# Benchmark Groovy, Groovy++, Java



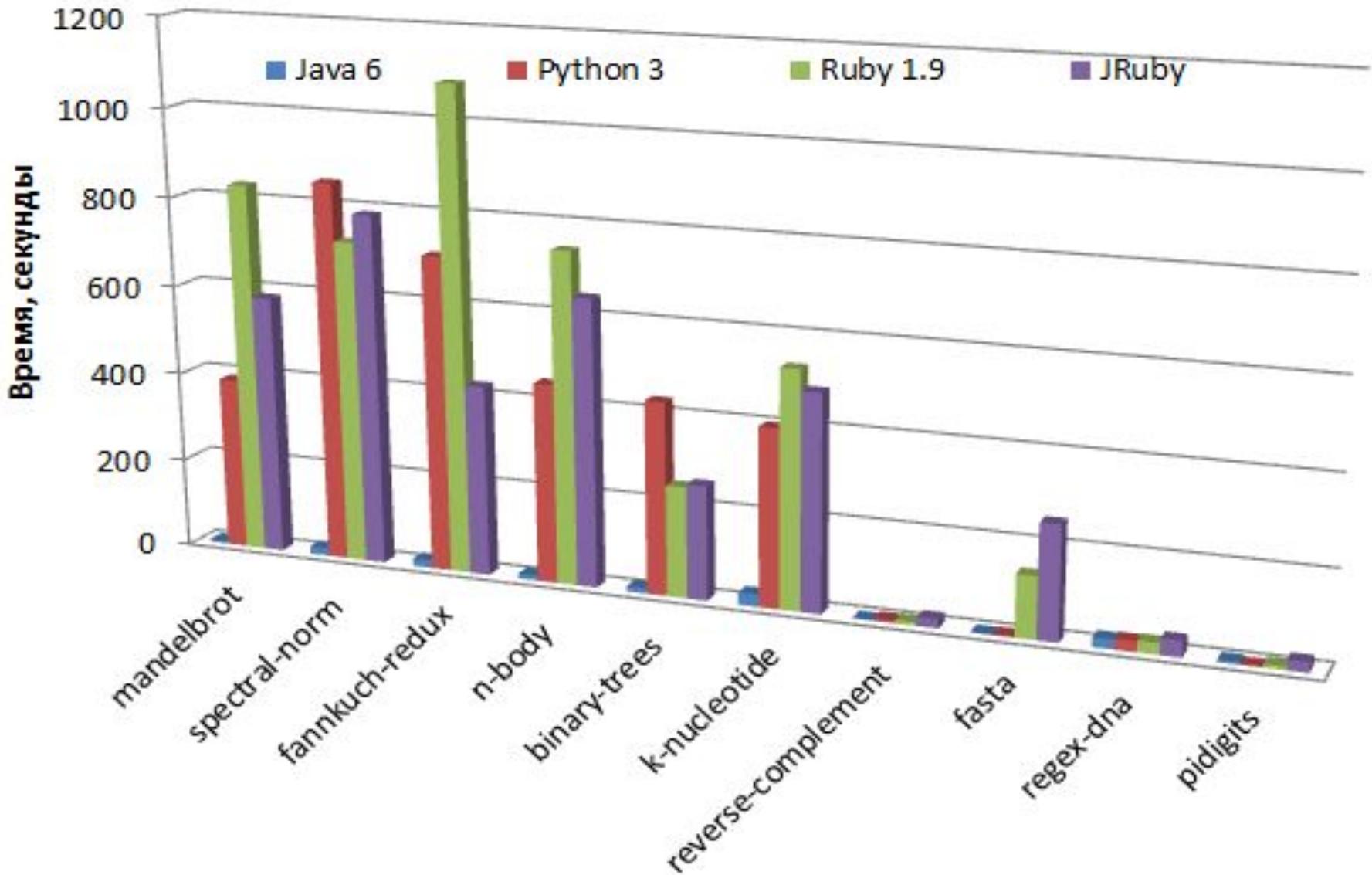
<https://github.com/alextkachman/fib-benchmark>

# Но на этом можно работать



Groovy работает также как Python, Ruby, PHP и т.п.

# Benchmark Java, Python, Ruby



# Скорость Groovy

- не забываем, что часто узкое место база данных
- любой фрагмент можно переписать на java
- любой фрагмент можно ~~переписать~~ сделать Groovy++

# Groovy++

- Статически типизированное расширение Groovy
- По скорости выполнения почти не уступает Java
- Может рассматриваться как альтернатива Scala
- Пишется небольшой группой энтузиастов (один хакер?), мало используется

# IDEA

IDEA в целом очень хорошо поддерживает groovy:

- Для работы с динамическими методами и полями в IDEA есть Dynamic properties
- Работает выведение типов, в основном 😊

Тем не менее:

- Для динамики мы теряем автоматический рефакторинг и высокоуровневый поиск (findUsages)
- В отладчике иногда сильно тормозит Step Into

Спасибо

evgenyk@gmail.com

