vert.x

Bartek Zdanowski
TouK

vert.x

Bartek Zdanowski

@BartekZdanowski

- developer @ Touk make IT simple
- współorganizator Confitury
- członek Warszawa JUG
- dr Zdanek
- tata i mąż :)

vert.x

czym jest vert.x?

vert.x

- w czym problem?
  - rośnie liczba użytkowników mobilnych*
    - 2,1 mld w 2012
    - 7 mld w 2018
  - rośnie liczba urządzeń inteligentnych
    - Internet of things - 30 mld do 2020!**
    - IPv6 czeka na nich!

* http://mobithinking.com/mobile-marketing-tools/latest-mobile-stats/b

** http://en.wikipedia.org/wiki/Internet_of_Things

vert.x

- w czym problem?
  - Tomcat - 200 wątków = 200 połączeń
  - reszta połączeń czeka…

vert.x

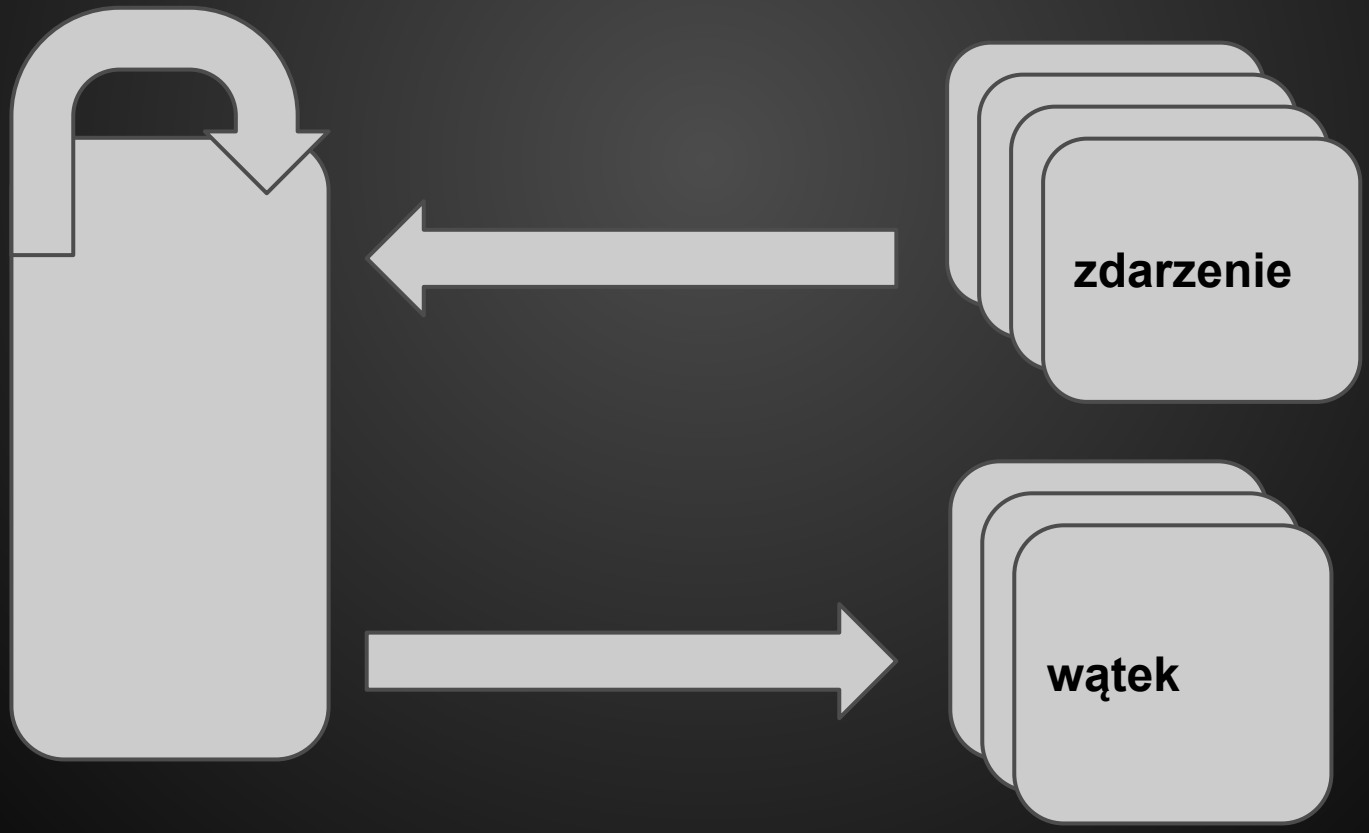
- w czym problem?
- 1 wątek per zadanie
  - jeśli wątek czeka na wynik pracy, stoi cała kolejka zadań
  - tradycyjne programowanie synchroniczne
  - oczekiwanie na wynik (pętla!)

vert.x

- jak można inaczej?
  - 1 zadanie = seria zdarzeń luźno powiązanych
  - asynchronicznie
  - Zostać poinformowanym o danych do przetworzenia
  - oddać kontrolę do wątku, zamiast czekać na wynik operacji

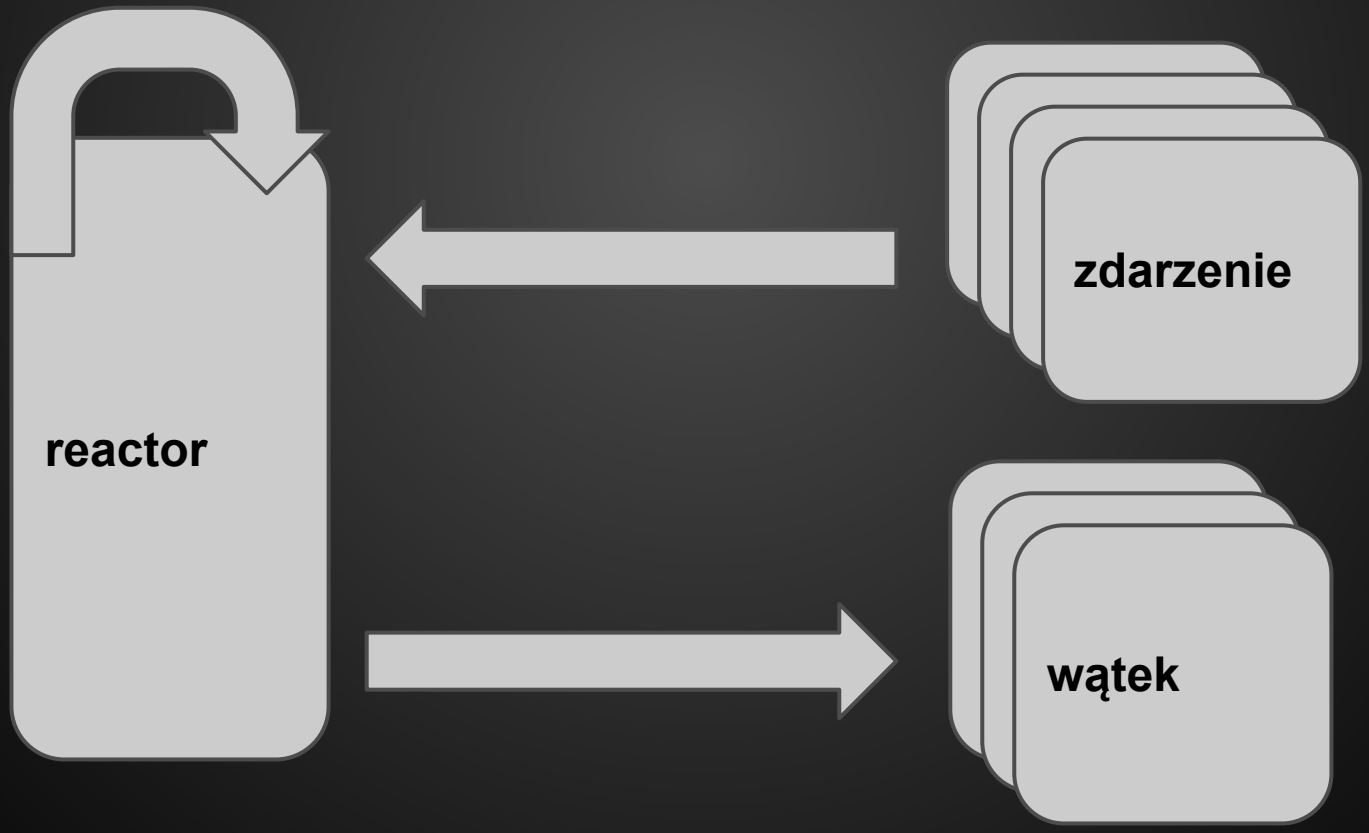
vert.x

- jak można inaczej?
- trzeba czekać! ale może to robić 1 wątek!



vert.x

- jak można inaczej?
- trzeba czekać! ale może to robić 1 wątek!



vert.x

- jak to robi vert.x?
  - multi-reactor pattern
  - pulą wątków równa ilości core’ów

vert.x

- jak to robi vert.x?
  - od początku jest asynchroniczny
  - sterowany zdarzeniowo
  - rozproszony
  - skalowalny
  - bezpieczny wątkowo
  - używa modelu aktora

vert.x

- demo

vert.x

- bezpieczna wielowątkowość?

```
class MyService {
      public synchronized Result doSomething(Data data) {
            //do some critical stuff
      }
}
```

vert.x

- bezpieczna wielowątkowość?

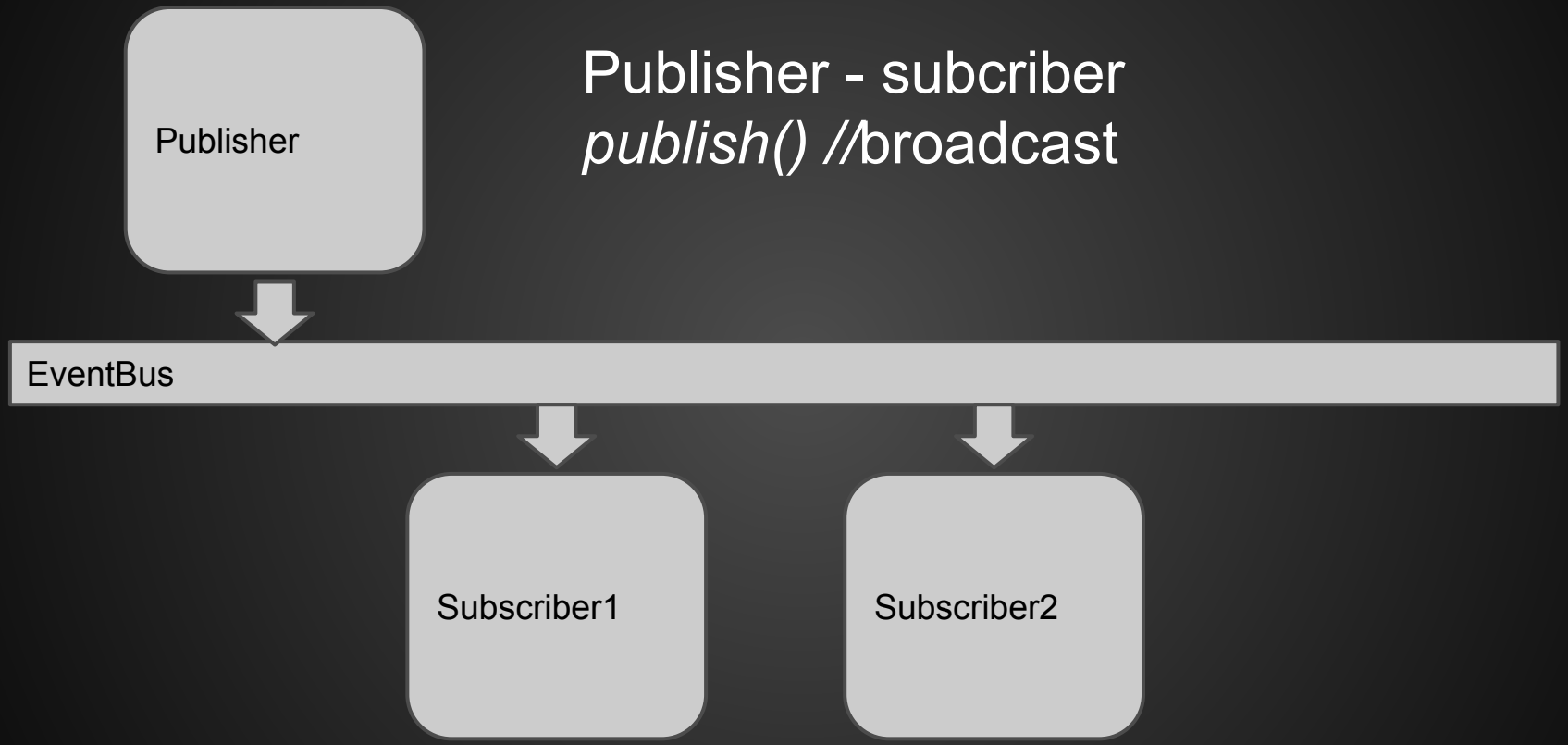
```
class MyService {
      public synchronized Result doSomething(Data data) {
            //do some critical stuff
      }
}
```

- 1 wątek!

vert.x

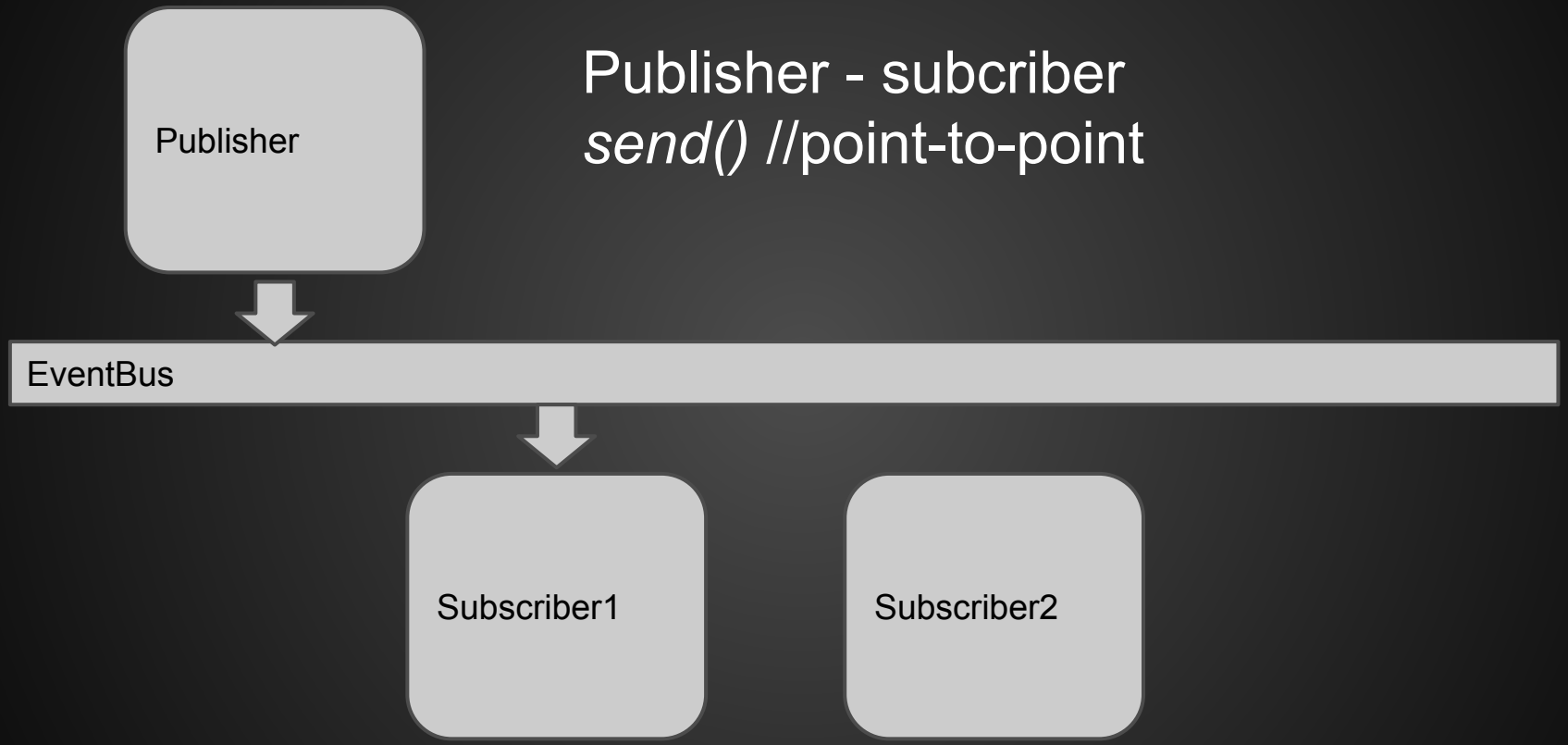
- bezpieczna wielowątkowość!
  - dany verticle uruchmiony zawsze w tym samym wątku
  - oddzielne classloadery
  - event bus
  - shared data: maps, sets

vert.x



Publisher - subcriber
*publish() //*broadcast

vert.x



Publisher - subcriber
send() //point-to-point

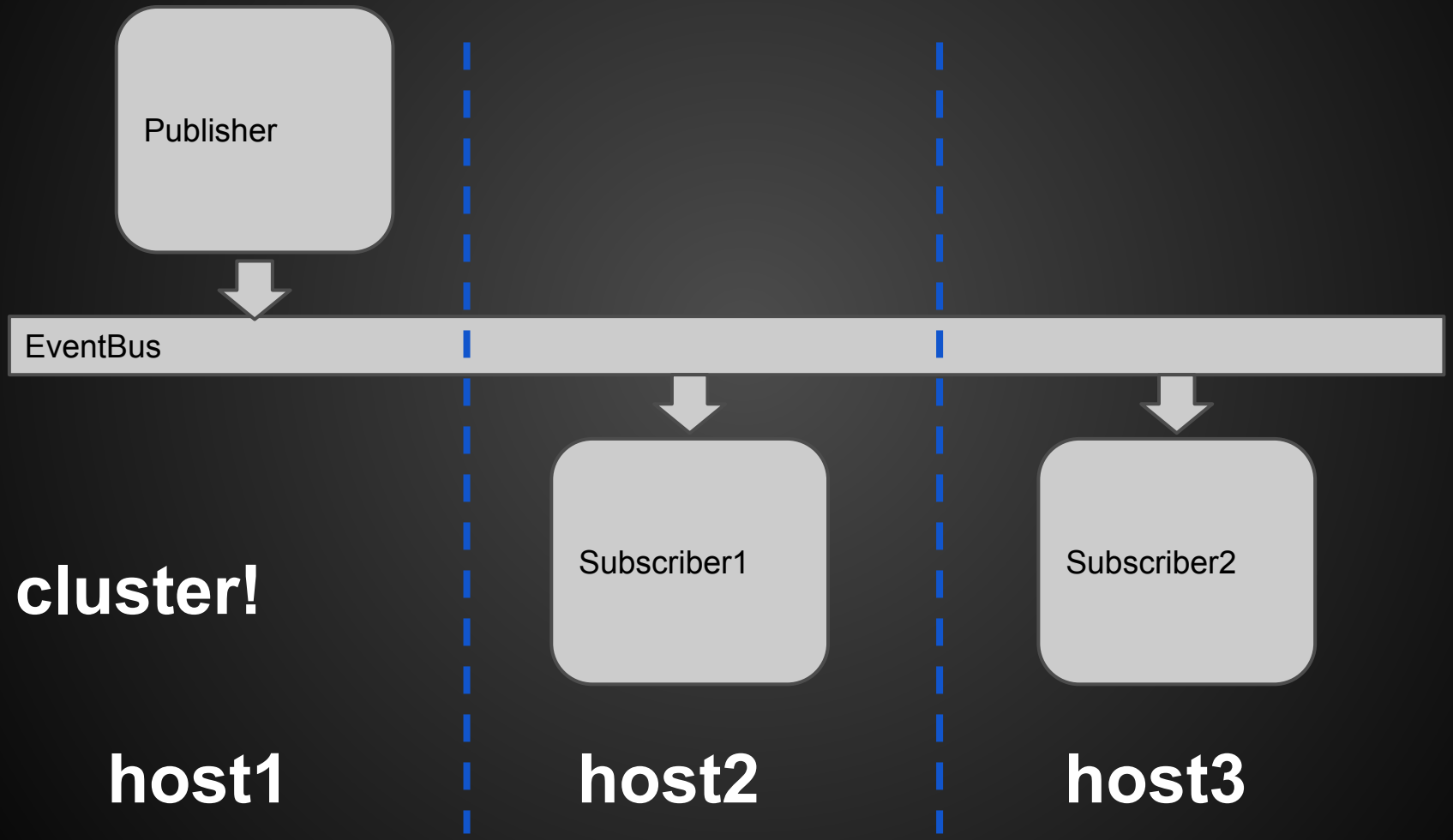
vert.x

- EventBus
  - publish - wszyscy
  - send - jeden, round robin
  - kolejkowanie
  - brak potwierdzenia
  - wiadomości tylko w pamięci - ulotne!

vert.x

- EventBus
  - distributed - cluster
  - sięga klienta przez SockJS

vert.x


Publisher
EventBus
Subscriber1
Subscriber2
cluster!
host1
host2
host3

vert.x

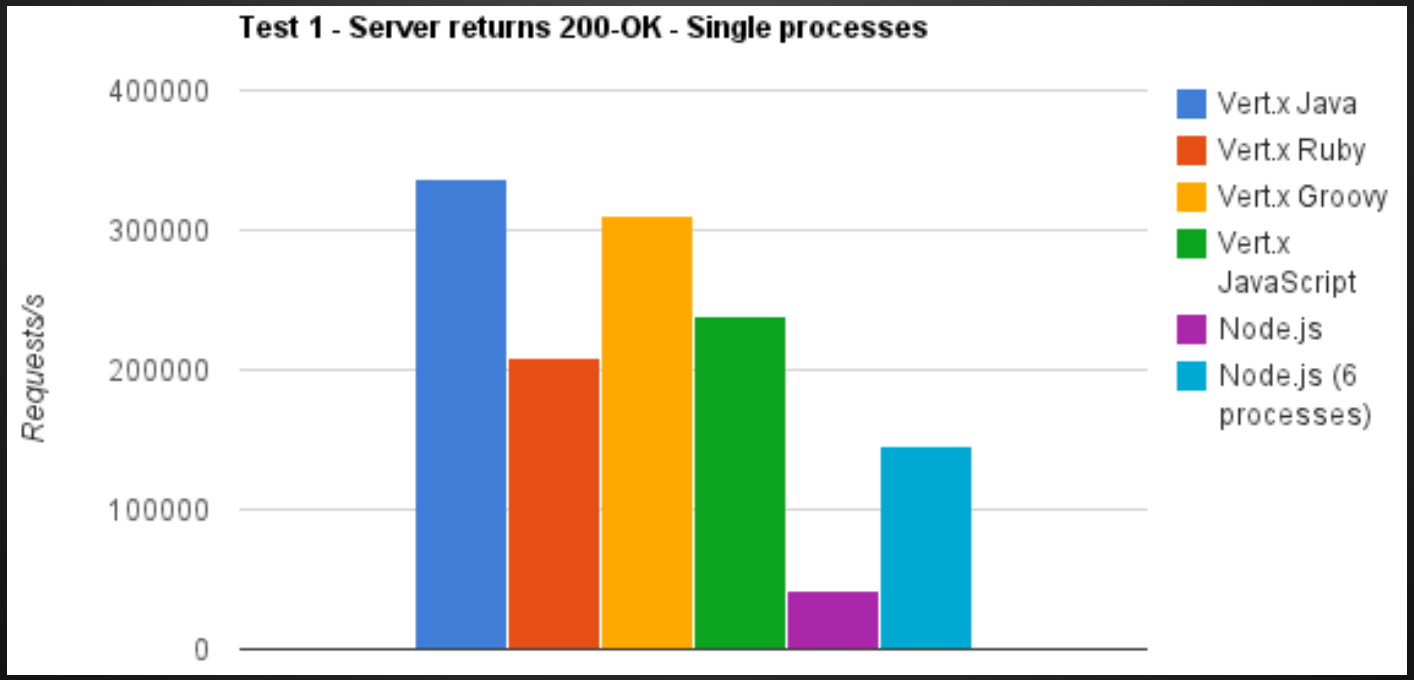
- Jak vert.x się skaluje
  - wiele instancji jednego verticle'a
  - cluster (automagicznie!)
  - eventbus rozpina się pośród node'ów clustra
  - wykorzystuje wszystkie core'y

vert.x

- Inne właściwości
  - Polyglot
    - Java, JavaScript, CoffeeScript, Ruby, Python, Groovy + Scala, Clojure i...PHP
  - Moduły + publiczne repo
  - Osadzanie vert.x w aplikacji
  - Server Http + Sockety
  - Filesystem API

vert.x

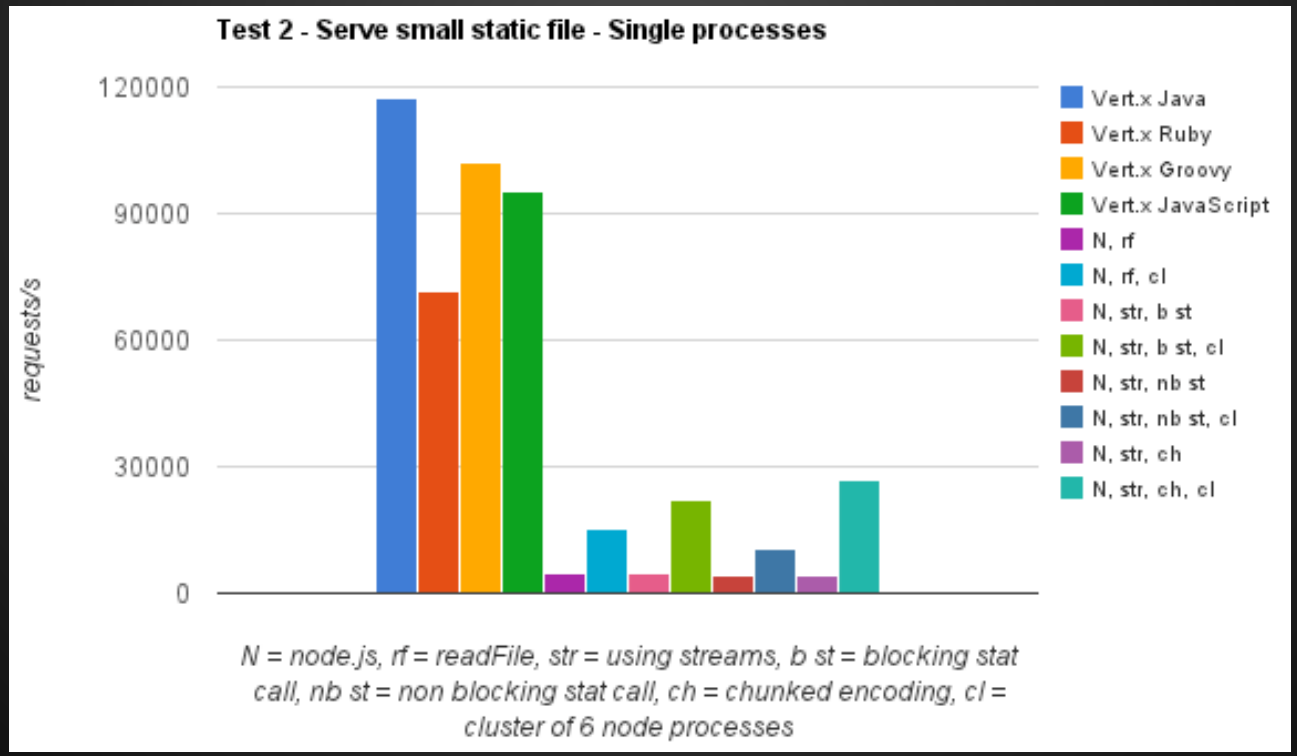
- Konkurencja?
- node.js! really?!



http://www.cubrid.org/blog/dev-platform/inside-vertx-comparison-with-nodejs/

vert.x

- Konkurencja?
- node.js! really?!



http://www.cubrid.org/blog/dev-platform/inside-vertx-comparison-with-nodejs/

vert.x

- Podsumowanie
  - wysoce skalowalny
  - bezpieczny i wielowątkowy
  - rozproszony EventBus
  - wielojęzyczny (polyglot)
  - gotowy na 2020 - 30 mld urządzeń ;)

vert.x

vert.x na Raspberry Pi!

vert.x

demo

vert.x

Dziękuję!