

Web performances

@k33g_org
@nicolasleroux
!@RemiForax
@jeffmaury
@ehsavoie

DEVOXXTM France

#devovxfr-webperf

@DevovxFR

Agenda

- Pourquoi
- L'application
- Les stacks utilisés
- L'infrastructure et les outils
- Les tests
- Conclusion

La team

- Philippe Charriere (@k33g_org)
- Remi Forax
- Nicolas Leroux (@nicolasleroux)
- Jeff Maury (@jeffmaury)
- Emmanuel Hugonnet (@ehsavoie)

Un grand merci





DEVVOXXTM

#devoxfr-webperf

DEVVOXXTM France

@DevoxFR

Objectifs

- Revenir sur <http://blog.shinetech.com/2013/10/22/performance-comparison-between-node-js-and-java-ee/>
- Montrer que cela peut être simple
- Pas de backend, cela ralentit
- On vise la prod, les stacks standard sont utilisées
- Le cloud est notre ami
- Faire un état de l'art sur les technos

L'application

- Service de recommandation
- 10000 utilisateurs, 800 films
- Recherche par utilisateur, film
- Les utilisateurs votent pour des films
- Similarité : calcul de distance (euclidienne) entre 2 utilisateurs

Movie Buddy

Users

Movies

...

Thu Apr 17 2014 16:01:34 GMT+0200 (CEST)

Search users by name

bob

Search

id	name	
2164	Bobby Stamm	<div>select</div>
3902	Dr. Bobbie Veum	<div>select</div>
4806	Bobbie Miller	<div>select</div>
5707	Bobby Veum	<div>select</div>
5965	Bobby Har ^a ann	<div>select</div>
6250	Dr. Bobby McLaughlin	<div>select</div>
7056	Bobbie Strosin	<div>select</div>

Movie Buddy

Users

Movies

...

Dr. Bobbie Veum has just rated Police Academy : 4

Search movies

title

Search

Comedy

Search

actors

Search

Title	Genre	Actors	Rating
Rick Santorum Aborts Presidential Campaign	Short, Comedy	Ashley Judd, Michelle Trachtenberg, Katy Mixon, Eliza Coupe	<div><div>0</div><div>1</div><div>2</div><div>3</div><div>4</div><div>5</div></div>
Gary and the Store Button, Howard Addresses Tonight Show Rumor, Howard and Fred Write a Song, and Scott the Engineer's Bowling Ring	Comedy, Talk-Show	Gary Dell'Abate, Robin Quivers, Scott Salem, Howard Stern	<div><div>0</div><div>1</div><div>2</div><div>3</div><div>4</div><div>5</div></div>
Wataru to himiko no achi achi adobenchâ	Animation, Adventure, Comedy	N/A	<div><div>0</div><div>1</div><div>2</div><div>3</div><div>4</div><div>5</div></div>
Police Academy	Comedy, Crime	Steve Guttenberg, Kim Cattrall, G.W. Bailey, Bubba Smith	<div><div>0</div><div>1</div><div>2</div><div>3</div><div>4</div><div>5</div></div>
The Adding Machine	Comedy, Fantasy	Milo O'Shea, Phyllis Diller, Billie Whitelaw, Sydney Chaplin	<div><div>0</div><div>1</div><div>2</div><div>3</div><div>4</div><div>5</div></div>
Acupressure	Short, Comedy	Steve Exeter, Mike Lukey	<div><div>0</div><div>1</div><div>2</div><div>3</div><div>4</div><div>5</div></div>

L'application

- Services REST/JSON :
 - Utilisateurs : liste, lecture, recherche, distance
 - Films : liste, lecture, recherche
 - Vote : vote avec ranking
- Front-end :
 - Bootstrap/JQuery/Backbone/React.JS

Les API : film

- GET /movies : retourne la liste des films
- GET /movies/search/genre/criteria/limit
 - Retourne la liste des films dont le genre satisfait le critère avec une limite

Les API : vote

- POST /rates {« userid » : userid, « movieid »:movieid, « rate » : rate}
 - Enregistre le vote d'un utilisateur pour un vote

Les API : utilisateur

- GET /users/distance/id1/id2 :
 - Retourne la distance entre deux utilisateurs

Implémentations

- JEE/JAX-RS (Java)
- NodeJS/Express (Javascript)
- Play2/Scala (Scala)
- Undertow (Java)
- Vertx (Java)
- Finatra (Scala)

J2EE/JAX-RS

- Endpoints REST basés sur annotations JAX-RS
- Container Tomcat
- Implémentation JAX-RS : Jersey

J2EE/JAX-RS code

```
@GET
@Path("/")
@Produces(MediaType.APPLICATION_JSON)
public void getMovies(@Context HttpServletResponse response) throws IOException {
    PrintWriter writer = response.getWriter();
    writer.write("[");
    for(Movie movie : movieService.listMovies()) {
        writer.write(movie.toString());
        writer.write(",");
    }
    writer.write("]");
}
```

J2EE/JAX-RS code

```
@GET
@Path("search/genre/{genre}/{limit}")
@Produces(MediaType.APPLICATION_JSON)
public void searchByGenre(@Context HttpServletResponse response, @PathParam("genre") String genre,
    @PathParam("limit") String limit) throws IOException {
    PrintWriter writer = response.getWriter();
    writer.write("[");
    for(Movie movie : movieService.findByGenre(genre, Integer.parseInt(limit))) {
        writer.write(movie.toString());
    }
    writer.write("]");
    writer.flush();
}
```

Play 2.2 overview

- Playframework est un framework full stack.
- Support for Form, Validation, Redirection, etc..
- Json → Object Java / Scala
- Librairie pour manipuler les flux de données
- Code élégant

Play 2.2 – Get all movies

Fichier route

GET /movies controllers.Application.movies

Fichier Application.scala

```
def movies = Action {  
    Ok.chunked(Enumerator.fromStream(Play.resourceAsStream("movies.json").get))  
    .as("application/json")  
}
```

Play 2.2 – Get movies per genre

GET /movies/search/genre/:genre/:limit

controllers.Application.movieByGenre(genre: String, limit: Int)

```
def movieByGenre(genre: String, limit: Int) = Action {  
    val pattern = Pattern.compile(genre.toLowerCase);  
    val movies:Seq[Movie] = Repository.movies  
        .filter(movie => pattern.matcher(movie.genre.toLowerCase)  
            .find())  
        .take(limit)  
    }  
    Ok(Json.toJson(movies))  
}
```

Express + Node

- Node : plateforme construite sur la V8 du Javascript Engine de Chrome
- Plateforme js du back au front
- Facilité d'échange avec le navigateur
- Express : Framework pour construire des Webapps
- Avantages : le JSON est natif
- Module cluster ...

Express + Node / Code

```
1 app.get("/movies", function(req, res) {
2   res.send(movies);
3 });
4
5 app.get("/movies/search/genre/:genre/:limit", function(req, res) {
6   res.send(movies.filter(function(movie) {
7     return movie.Genre.toLowerCase().
8       search(new RegExp(req.params.genre.toLowerCase()), "g") != -1;
9   })).slice(0, req.params.limit));
10 });
```

Finatra

- Framework web de chez Twitter, en Scala inspiré par Sinatra
- Ressemble beaucoup à Express, le typage en plus
- Assez simple à mettre en œuvre même pour un débutant Scala
- Librairie standard de Scala pour parser le JSON du **POST** :
`JSON.parseFull(request.getContentString())
 .get.asInstanceOf[Map[String, Double]]`

Finatra / Code

```
1 // moviesList : List[Map[String,Any]]
2
3 get("/movies") { request =>
4   render.json(moviesList).toFuture
5 }
6
7 get("/movies/search/genre/:genre/:limit") { request =>
8   val genre = request.routeParams.getOrElse("genre","?")
9   val limit = request.routeParams.getOrElse("limit",1).toString()
10  val searchedMovies = moviesList.filter(movie => {
11    movie("Genre").toString().toLowerCase().contains(genre)
12  }).slice(0, limit.toInt)
13  render.json(searchedMovies).toFuture
14 }
```

Vertx in one slide

- Polyglot application, heavily inspired by node.js but multi-threaded
- Based on Netty for Networking, Jackson for Json
- Deploy sources, no Maven, no Gradle, no Ant ??
- Based on Java 7

Vertx + Java 8 code

```
route.get("/movies", req -> {
    req.response().sendFile("/db/movies.json");
});

route.get("/movies/search/genre/:genre/:limit", req -> {
    Pattern pattern = compile(req.params().get("genre").toLowerCase());
    req.response().end(
        movies.stream()
            .filter(movie -> pattern.matcher(movie.genre).find())
            .map(Movie::toString)
            .limit(parseInt(req.params().get("limit")))
            .collect(joining(", ", "[", "]")));
    });
```

Vertx @ Jackson vs Java 8

Vertx can use lambda out of the box
All handlers are interfaces with one method

But not support of new Java 8 classes
(`java.util.Stream`)

For Jackson, no automatic mapping json file -> Stream
only 120 lines of code

For Vertx, async read but sync-like write
the Vertx response should take a Stream as parameter

Undertow / in one slide

Use non standard NIO (XNIO)

Flexible

full Java EE servlet 3.1 container

low level specific non blocking handler API

Embeddable

Undertow Servlet / Code

```
DeploymentInfo servletBuilder = Servlets.deployment()  
    .setClassLoader(SearchMoviesServlet.class.getClassLoader())  
    .setContextPath(MYAPP)  
    .addWelcomePage("index.html")  
    .setResourceManager(new ClassPathResourceManager(SearchMoviesServlet.class.getClassLoader()))  
    .setDeploymentName("moviebuddy.war")  
    .addServlets(  
        Servlets.servlet("Movies", SearchMoviesServlet.class)  
            .addMapping("/movies")  
            .addMapping("/movies/*")  
            .setAsyncSupported(true), ...  
    );
```

← Declaring a servlet

```
DeploymentManager manager = Servlets.defaultContainer().addDeployment(servletBuilder);  
manager.deploy();  
PathHandler path = Handlers.path(Handlers.redirect(MYAPP))  
    .addPrefixPath(MYAPP, manager.start());
```

```
Undertow server = Undertow.builder()  
    .addHttpListener(port, hostname).setHandler(path)  
    .setBufferSize(1024 * 16).setWorkerThreads(50).build();  
server.start();
```

← Starting the server

Undertow Handler / Code

```
Undertow server = Undertow.builder()
    .setHandler(Handlers.header(Handlers.path(Handlers.resource(
        new WebappClassPathResourceManager(MYAPP,
        SearchMoviesHandler.class.getClassLoader()))))
        .addPrefixPath(MYAPP + "/rates", new RatesHandler())
        .addPrefixPath(MYAPP + "/users", new SearchUsersHandler())
        .addPrefixPath(MYAPP + "/movies", new SearchMoviesHandler()),
        Headers.SERVER_STRING, "U-tow")).build();
server.start();
```

Undertow Handler / Code

```
protected void doGet(HttpServerExchange exchange) throws Exception {
    final String[] params = URLParser.parse(PREFIX, exchange);
    String result = RateService.INSTANCE.findRateByUser(Integer.parseInt(params[0]));
    //Sending the response in an nio way, ByteBuffer can be used.
    if (result == null) {
        exchange.setResponseCode(StatusCodes.NOT_FOUND);
    } else {
        exchange.getResponseHeaders().add(Headers.CONTENT_TYPE, "application/json");
        exchange.getResponseSender().send(result);
    }
    exchange.endExchange();
}

protected void doPost(HttpServerExchange exchange) throws Exception {
    exchange.startBlocking(); //We are going to read from the request inputstream
    List<JsonItem> items = JsonLoader.load(exchange.getInputStream());
    exchange.setResponseCode(StatusCodes.MOVED_PERMENANTLY);
    exchange.getResponseHeaders().put(Headers.LOCATION, "http://" + exchange.getHostAndPort()
        + MYAPP + RateService.INSTANCE.rateMovie(items.get(0)));
    exchange.endExchange();
}

@Override
public void handleRequest(HttpServerExchange exchange) throws Exception {
    //Only method to implement
    if (Methods.POST.equals(exchange.getRequestMethod())) {
        doPost(exchange);} else { doGet(exchange); }
}
```

L'infrastructure

- Serveur hébergé dans un cloud (Cloudbees)
- Cloudbees fournit les containers appropriés : Tomcat, Vertx, Play, Node
- Finatra et Undertow via le container Java standalone
- Frontal nginx pour la répartition de charge

L'outil de test de charge

- Gatling : gatling-tool.org
- DSL Scala pour décrire les scénarios
- Support très réactif (merci S LANDELLE)
- Architecture réactive permettant de gros scénarios de charge sur des machines classiques
- Les tests sont lancés par des jobs Jenkins : plugin Maven, plugin Jenkins pour l'archivage des résultats

Les scénarios

- Trois scénarios :
 - MovieLoading : on charge la liste des films
 - Vote : recherche de films puis vote
 - Distance : calcul de distance entre 2 paires d'utilisateurs
- Les scénarios simulent 100 utilisateurs en parallèle, sans pause et bouclant 5000 fois

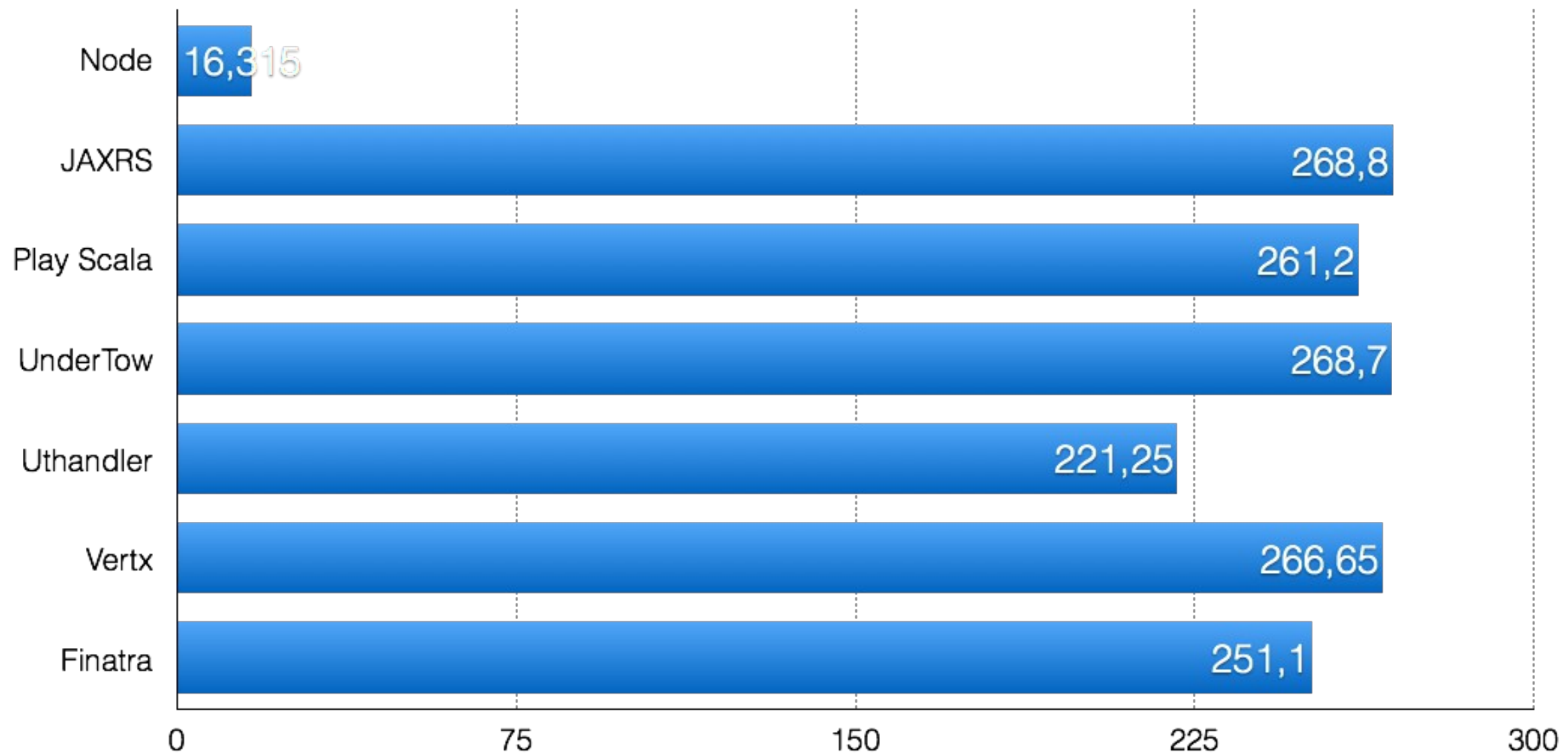
Scénario

```
class DistanceScenario extends Simulation {  
  val server = System.getProperty("buddyserver", "http://localhost:8080")  
  val totalUsers = Integer.getInteger("gatling.users", 100).toInt  
  val loops = Integer.getInteger("gatling.loops", 1000).toInt  
  val protocol = http.disableCaching.disableFollowRedirect  
  
  val scn = scenario(s"Distance ($totalUsers users/$loops loops)")  
    .repeat(loops) {  
      exec(  
        http("Distance 3022 <-> 9649")  
          .get(server + "/users/distance/3022/9649")  
          .check(status.is(200))),  
      exec(  
        http("Distance 2349 <-> 496")  
          .get(server + "/users/distance/2349/496")  
          .check(status.is(200)))  
    }  
  setUp(scn  
    .inject(rampUsers(totalUsers) over (totalUsers seconds)).protocols(protocol))  
}
```

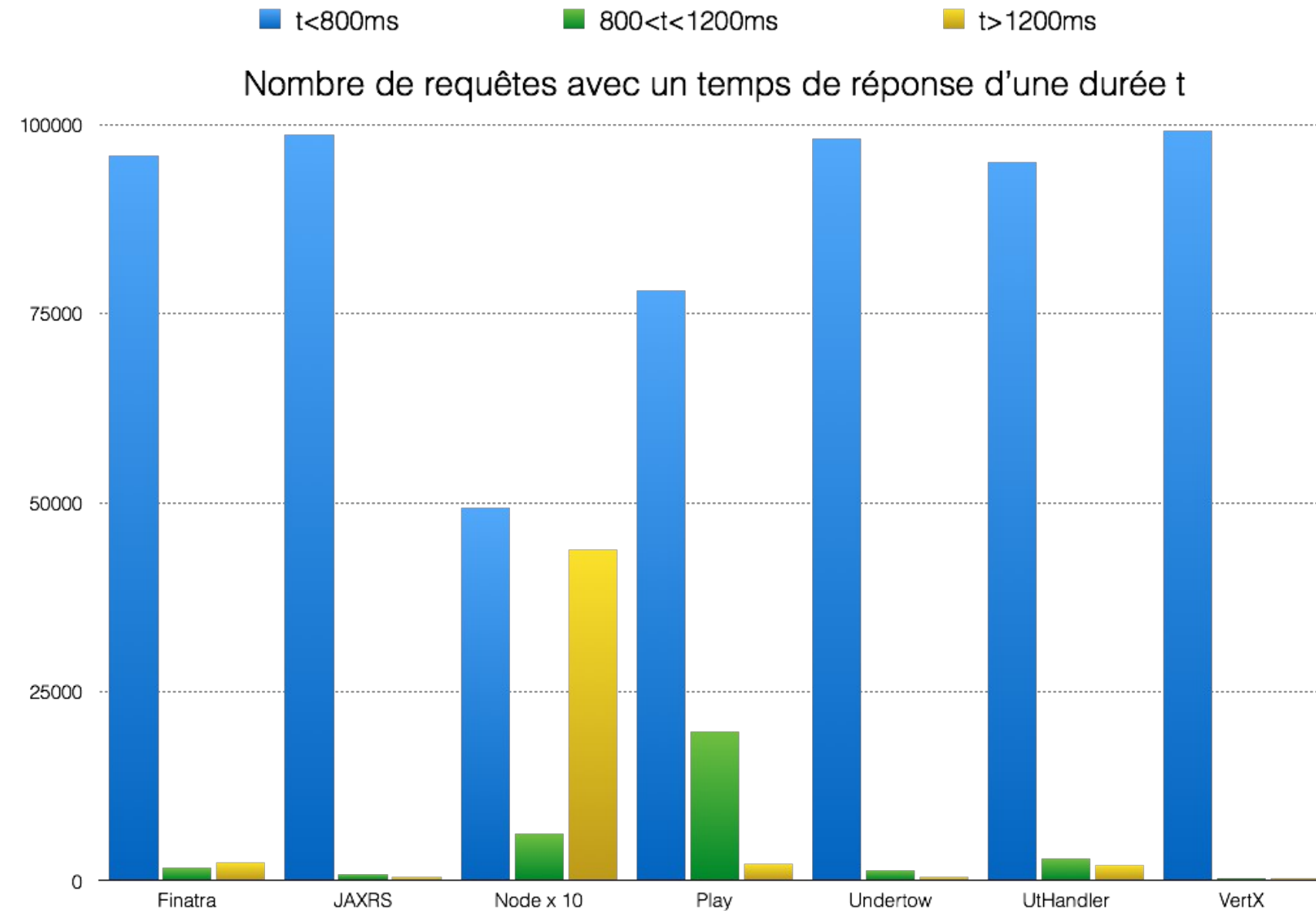
Scénario MovieLoading

- Chargement de la liste de film (400Ko)
- IO bound
- Valide la gestion des IO du framework

Nb Reqs/sec : All Movies



Temps de réponse



Changement de code

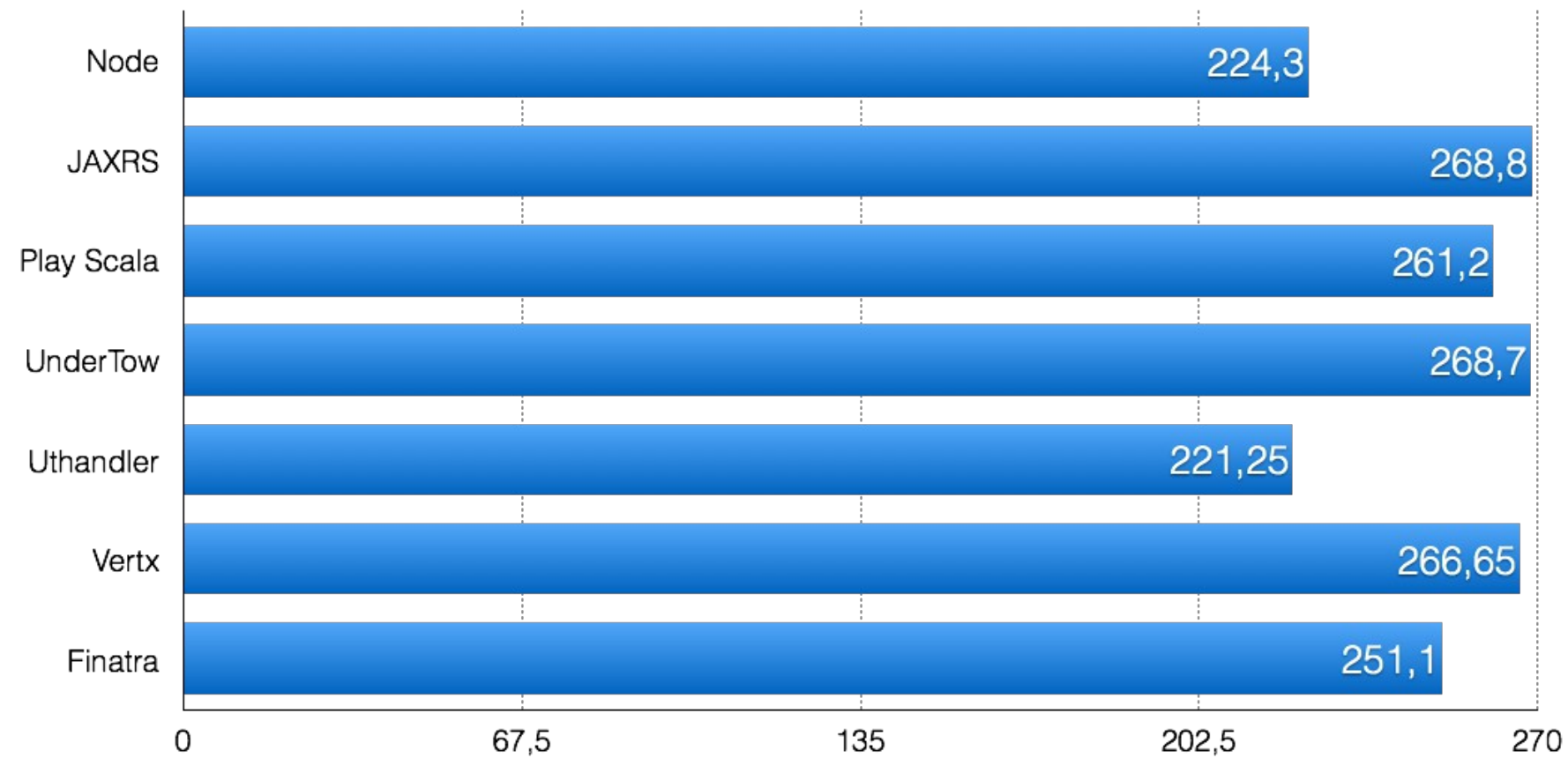
// avant

```
app.get("/movies", function(req, res) {  
  res.send(movies);  
});
```

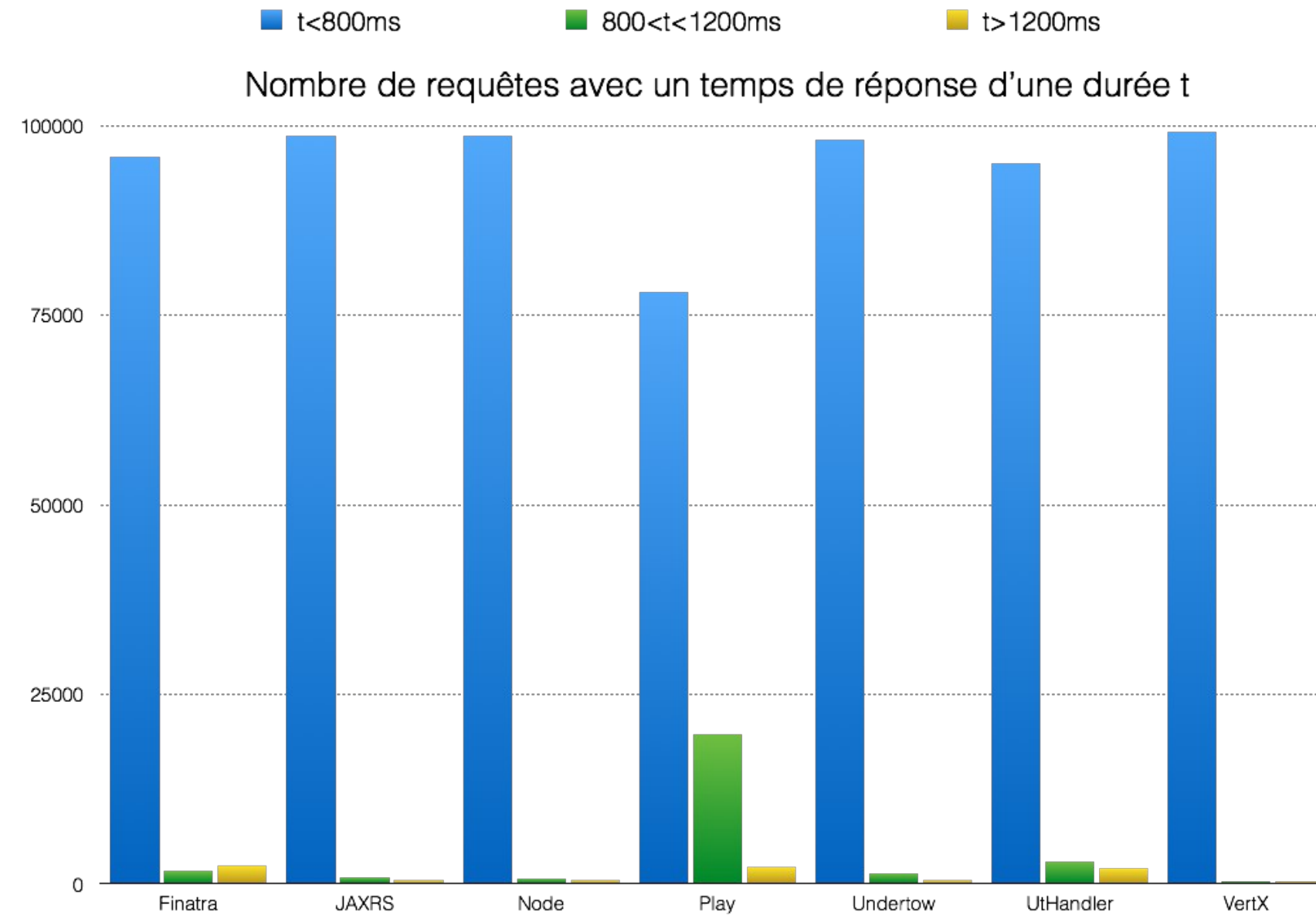
// après

```
app.get("/movies", function(req, res) {  
  res.sendFile("./db/movies.js", "utf8");  
});
```

Mais ...



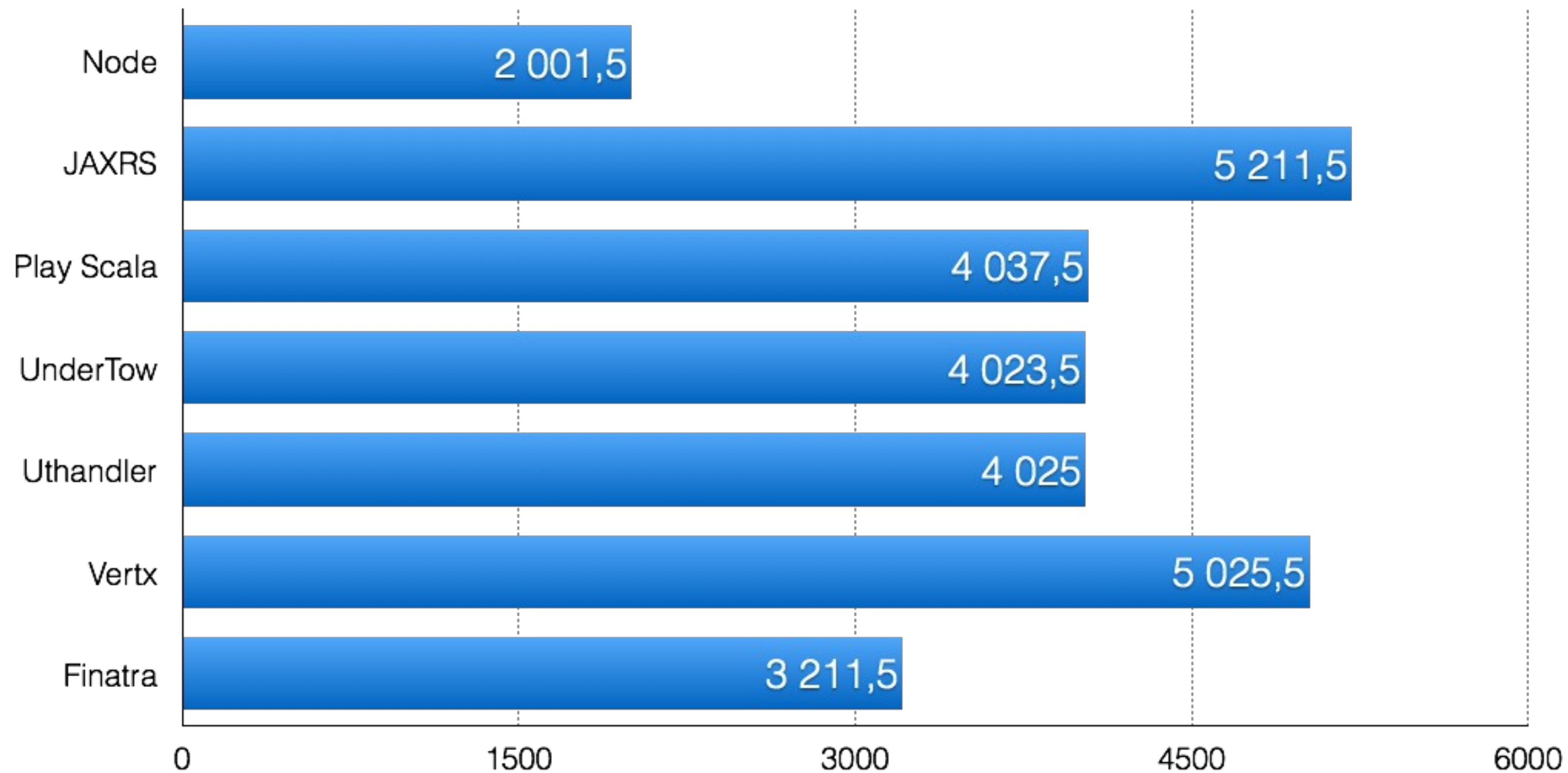
Mais ...



Scénario Vote

- Payload léger
- Plus orienté CPU (recherche via les expressions régulières)

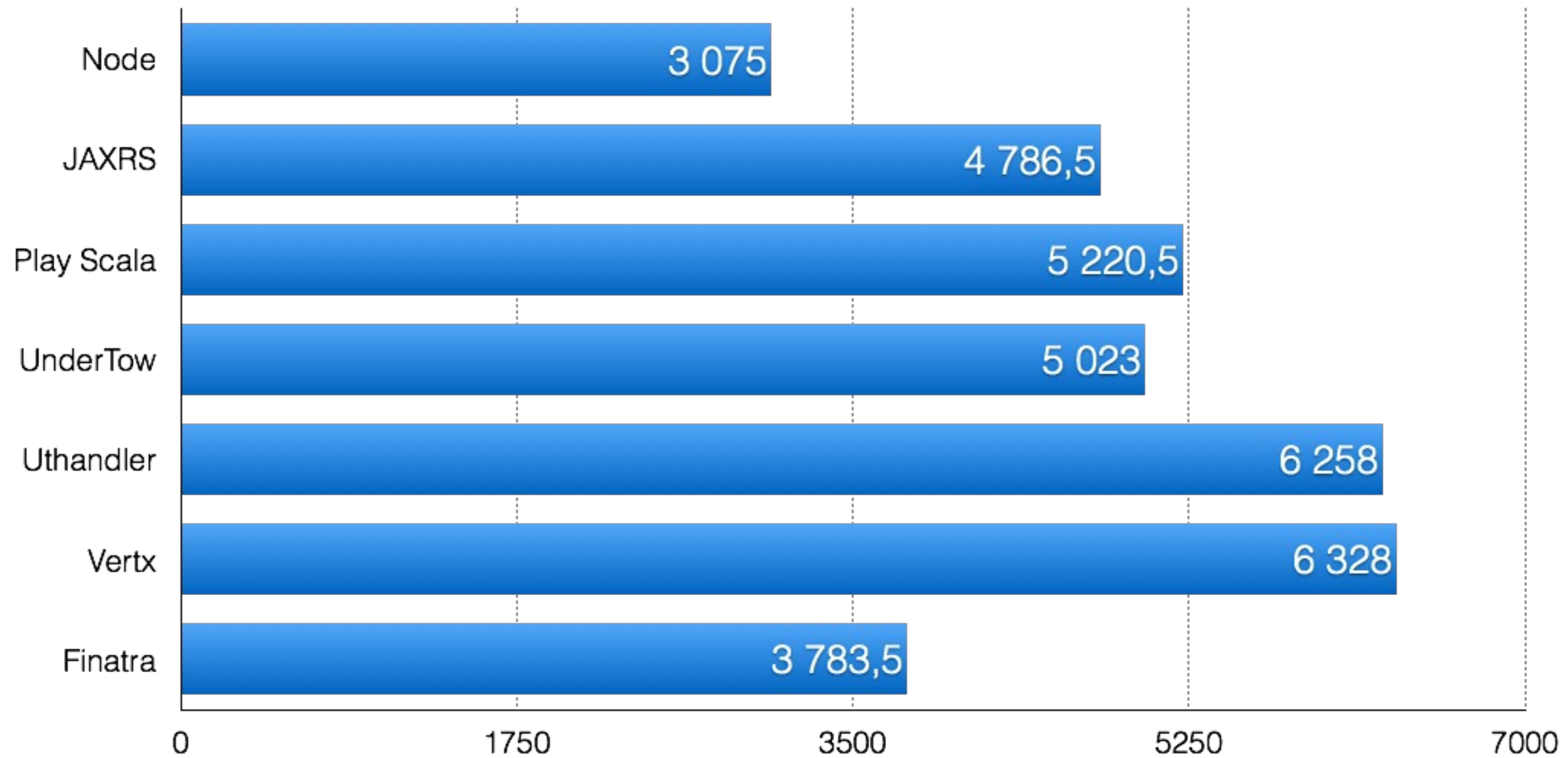
Nb Reqs/sec : Votes



Scénario Distance

- Payload quasi nul
- CPU bound : calcul de distance

Nb Reqs/sec : Distances





Summary

#YourSessionHashtag

DEV^{OX}™ France

@YourTwitterHandle

Conclusion

- Testez la performance, ce n'est pas compliqué
- Le développeur, c'est la clé
- Comprenez l'architecture
- Validez vos choix architecturaux au plus tôt
- To be continued...



#YourSessionHashtag

DEVOXX[™] France

@YourTwitterHandle