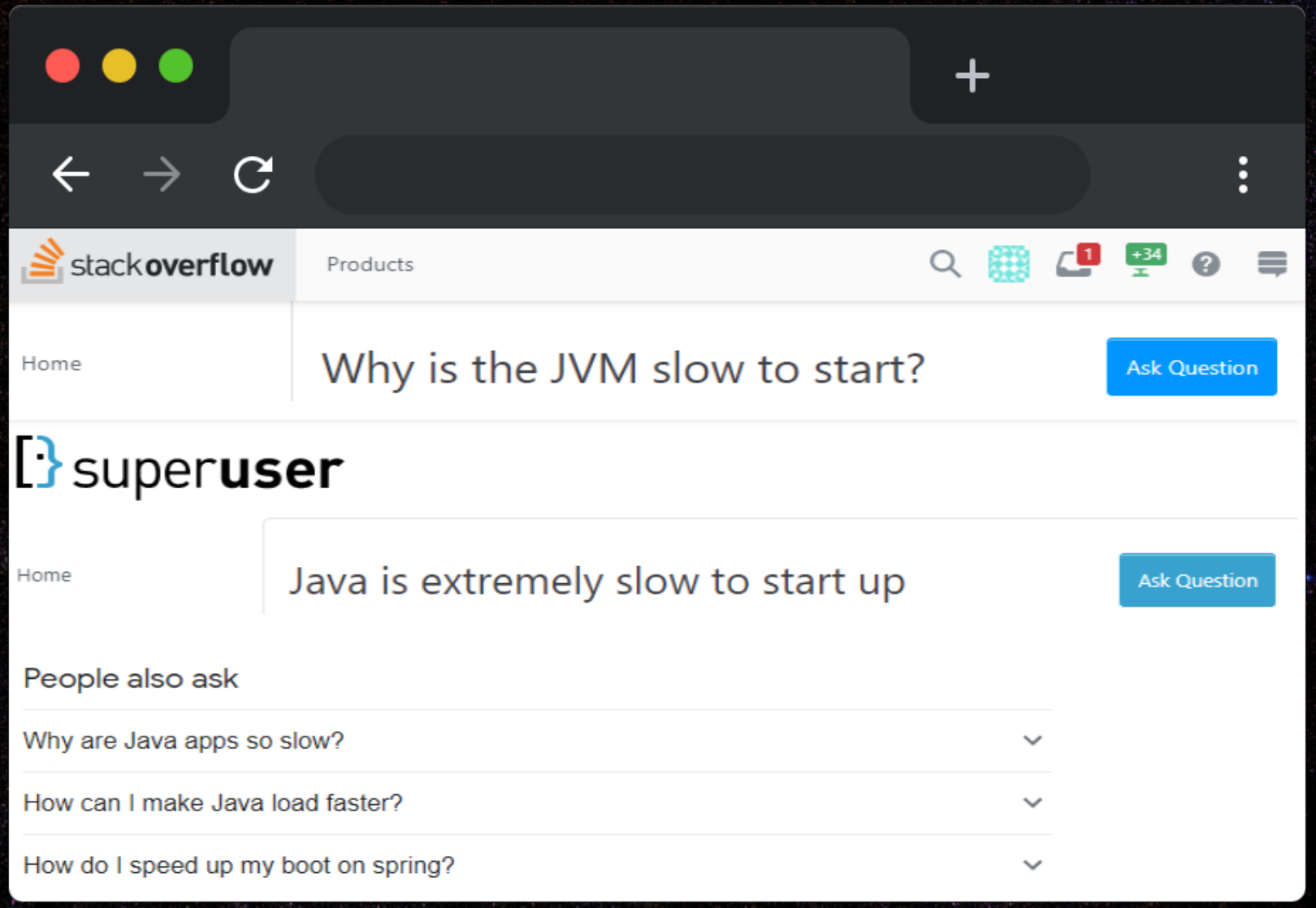
Konumuz

GraalVM™

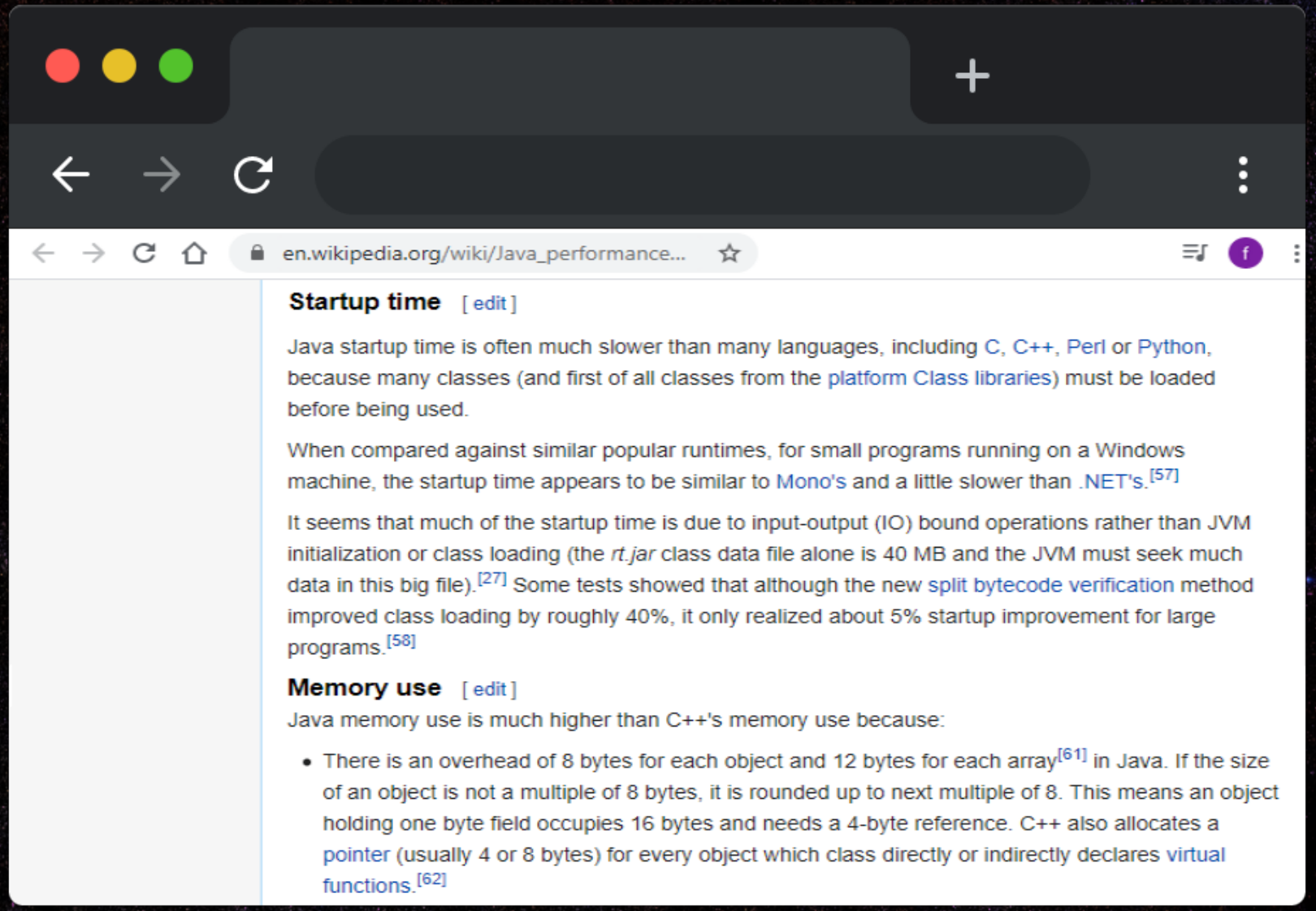
adesso

Fırat GÜRSOY - Kıdemli Yazılım Geliştirici / Mayıs 2021

Soru ve problem neydi?



Ne yapabiliriz?



Startup time [edit]

Java startup time is often much slower than many languages, including C, C++, Perl or Python, because many classes (and first of all classes from the platform Class libraries) must be loaded before being used.

When compared against similar popular runtimes, for small programs running on a Windows machine, the startup time appears to be similar to Mono's and a little slower than .NET's.[57]

It seems that much of the startup time is due to input-output (IO) bound operations rather than JVM initialization or class loading (the *rt.jar* class data file alone is 40 MB and the JVM must seek much data in this big file).[27] Some tests showed that although the new split bytecode verification method improved class loading by roughly 40%, it only realized about 5% startup improvement for large programs.[58]

Memory use [edit]

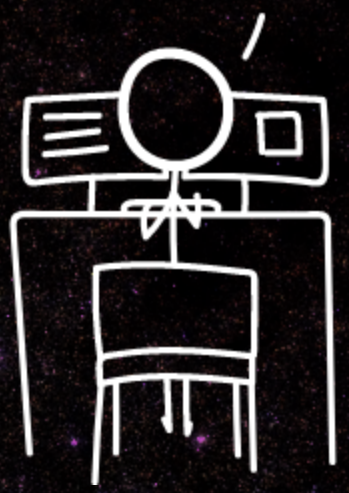
Java memory use is much higher than C++'s memory use because:

- There is an overhead of 8 bytes for each object and 12 bytes for each array[61] in Java. If the size of an object is not a multiple of 8 bytes, it is rounded up to next multiple of 8. This means an object holding one byte field occupies 16 bytes and needs a 4-byte reference. C++ also allocates a pointer (usually 4 or 8 bytes) for every object which class directly or indirectly declares virtual functions.[62]

Peki çözüm nedir?

Wait.
So you just save it,
And your code is running?
And it's Java?!

I know, right?
SUPERSONIC JAVA, FTW!



GraalVM nedir?

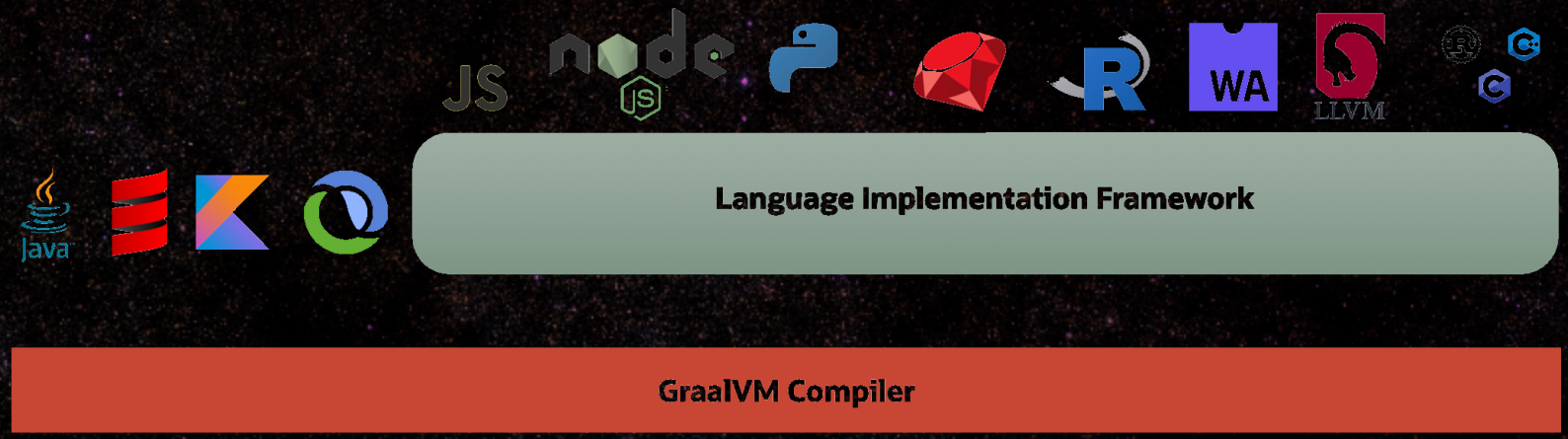
'GraalVM', 'Java' ve diğer JVM dillerinde yazılmış **uygulamaların yürütülmesini hızlandırmak** için tasarlanmış **yüksek performanslı** bir JDK dağıtımıdır ve 'JavaScript', 'Ruby', 'Python' ve ***bir dizi başka popüler dil***. `GraalVM` nin çok dilli yetenekleri, ***yabancı dil arama maliyetlerini ortadan kaldırırken*** birden çok programlama dilini tek bir uygulamada karıştırmayı mümkün kılar.

`Oracle` tarafından oluşturulan `GraalVM`. Üretime hazır ilk sürüm olan `GraalVM 19.0`, `Mayıs 2019`da yayınlandı. GraalVM'nin yaptığına eşdeğer bir başka teknoloji yoktur.

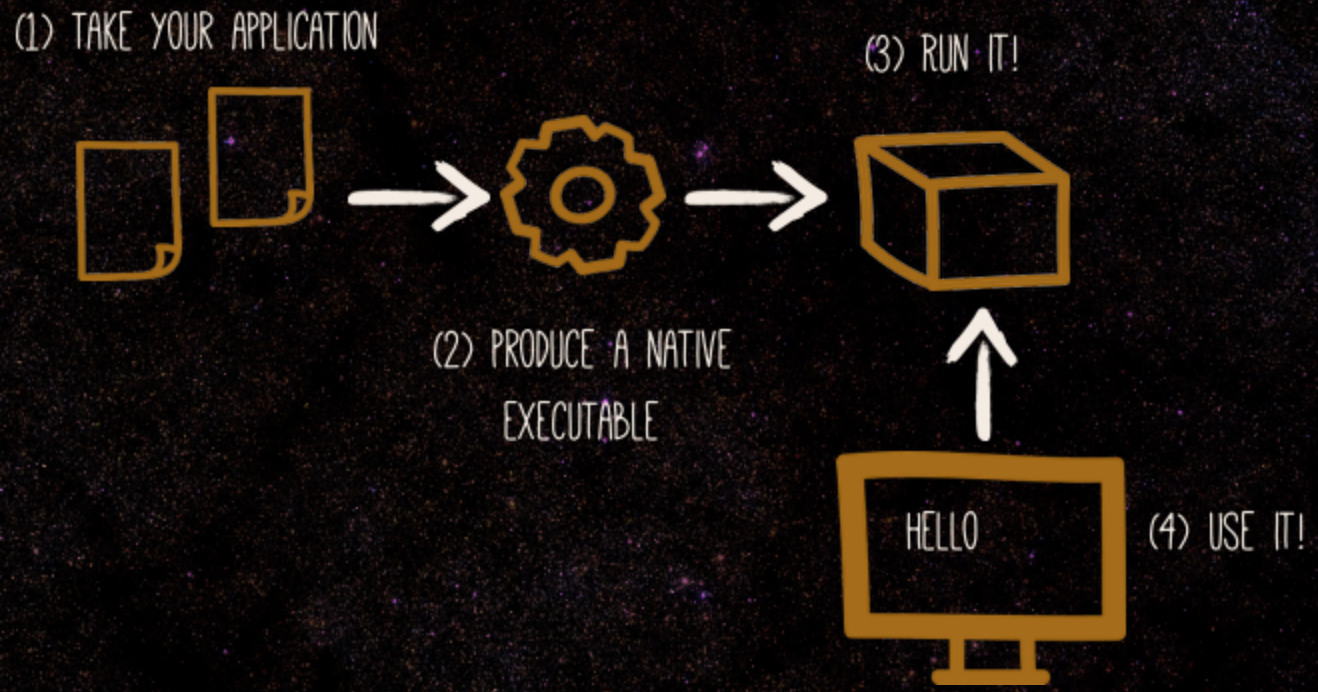
GraalVM bana ne verebilir?

- Tek bir uygulamada birden fazla programlama dilini karıştırmak
- Yerel yürütülebilir dosyalar
- `İnanılmaz` hızlı önyükleme süresi
- İnanılmaz derecede `düşük RSS belleği` (yalnızca yığın boyutu değil!)
- `Kubernetes` gibi kapsayıcı düzenleme platformlarında anında (nispeten) ölçek büyütme ve yüksek yoğunluklu bellek kullanımı.

GraalVM Mimarisi



Yerel yürütülebilir dosyalar



Uygulamamız için `yerel yürütülebilir dosya`, `uygulama kodu`, `gerekli kütüphaneler`, `Java API'leri` ve `bir sanal makinenin küçültülmüş sürümü`nü içerecektir. `. Daha küçük VM tabanı, uygulamanın `başlangıç zamanını` iyileştirir ve `minimum disk ayak izi` üretir.

Truffle dil uygulama çatısı

Truffle dil uygulama çatısı kendi kendini değiştiren Soyut Sözdizimi Ağaçları için yorumlayıcılar olarak araçlar ve programlama dilleri uygulamaları oluşturmaya yönelik açık kaynaklı bir kitaplıktır. Açık kaynak GraalVM derleyicisi ile birlikte,

```
Truffle, mevcut dinamik diller çağında programlama dili uygulama
teknolojisinde önemli bir adımı temsil ediyor.
```

Desteklenen çatılar ve teknolojiler

- Spring Framework (Deneyimsel)
- Quarkus
- Play Framework
- Camel
- Prometheus
- JavaFX

  ...

Dezavantajlar

- Bu yaklaşımın ana dezavantajı, platforma bağlı yerel koddur.

Bu, linux/windows vb. için kaynak kodunu derlemeniz gerektiği anlamına gelir.

O esnada

`SystemAdmin01:` server_api.so sunucumda çalışmıyor !

`Müşteri$$$__:` customer_api.exe bilgisayarımda çalışmıyor!

`DevGuy42_____:` api.jar iş istasyonumda çalışıyor!

`Fırat________:` Bazı durumlarda kullanılan kitaplıklardan veya derleme komut dosyalarından sorunlar çıkabilir. Yani testlere dikkat etmek gerekiyor.

`SystemAdmin01:` hımm, server_api.so benim sunucumda çalışmıyor !

.....

`DevGuy42 yazıyor ...`

Örnek Uygulama : İlk adım

Oracle indirmelerinden GraalVM'yi indirin ve kurun. Bir `Example.java` dosyası oluşturun.

```
public class Example {

    public static void main(String[] args) {
        System.out.println("Hello GraalVM");
    }
}
```

```
#Aşağıdaki komutları çalıştırın:
javac Example.java
native-image Example
```

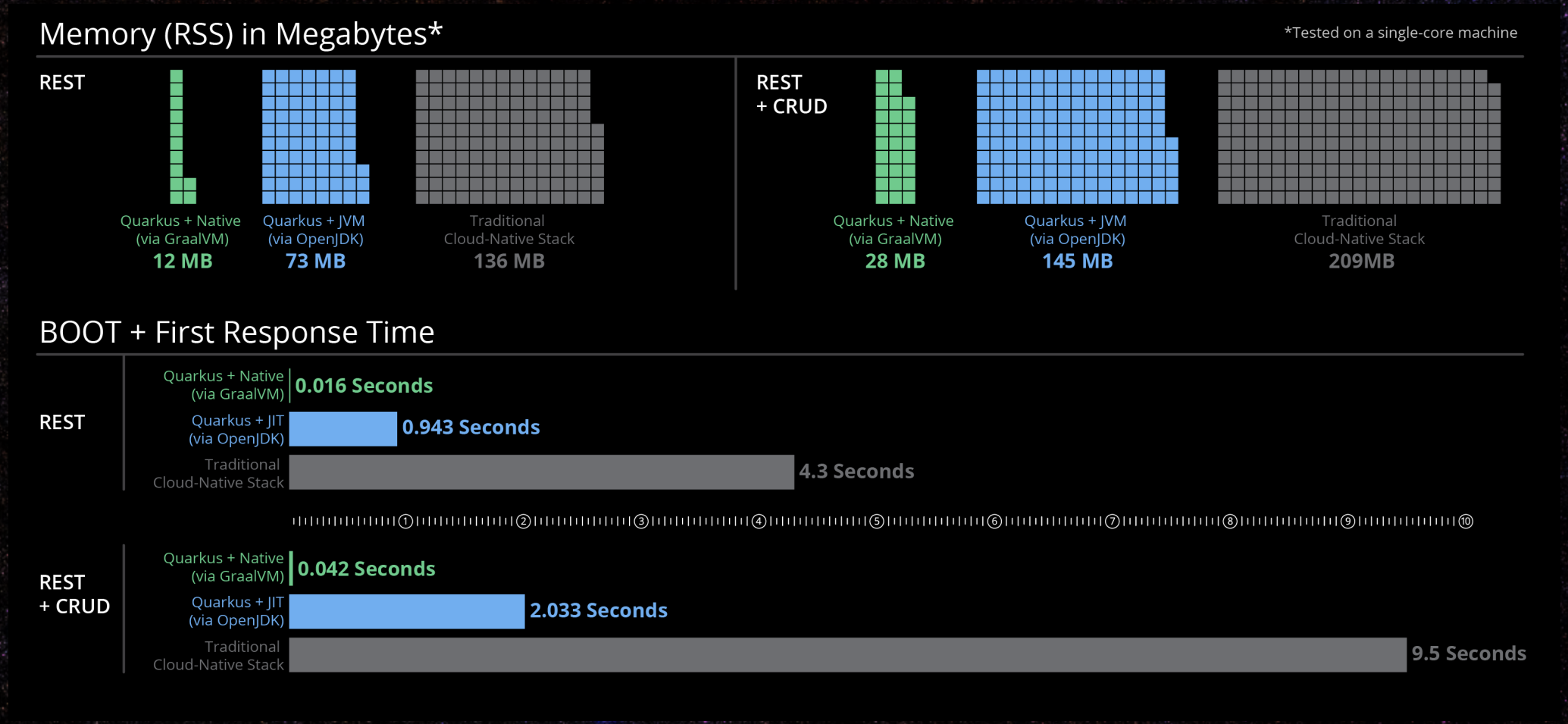
Testler : Tabiki

```java
package org.acme.quickstart;


import io.quarkus.test.junit.NativeImageTest;

@NativeImageTest
public class NativeGreetingResourceIT extends GreetingResourceTest {
    // Run the same tests
}
```

Performans



Quarkus Testi

Şuan Kullanılabilir mi?

GraalVM teknolojileri, üretime hazır ve deneysel olarak dağıtılır.

GraalVM'nin gelecekteki sürümleri için deneysel özellikler değerlendirilmektedir ve üretimde kullanılması amaçlanmamıştır. Geliştirme ekibi, deneysel özelliklerle ilgili geri bildirimleri memnuniyetle karşılar, ancak kullanıcılar, deneysel özelliklerin hiçbir zaman son sürüme dahil edilmeyebileceğini veya üretime hazır kabul edilmeden önce önemli ölçüde değişebileceğini bilmelidir.

Bu bilgiden sonra bireysel ihtiyacım için bir demo uygulaması yapmaya çalıştım.

Öncelikle şık kullanıcı arayüzüne sahip bir masaüstü uygulamasını tercih ettim.
(Nispeten:))

Bu, uygulamanın ne yazık ki çok fazla bellek kullandığı anlamına gelir. Bu yüzden performansı artırmak için `JavaFX` ve ayrıca `Gluon` (yerel bir JavaFX çözümü) kullanmam gerekiyor.

Graalvm entegrasyonu için ne yaptım?

Bu entegrasyon için sadece maven pom'uma bir eklenti ekledim.
Şaşırtıcı bir şekilde, entegrasyon için bu yeterlidir.

```
<plugin>
        <groupId>com.gluonhq</groupId>
        <artifactId>gluonfx-maven-plugin</artifactId>
        <version>1.0.2</version>
        <configuration>
                <target>host</target>
                <mainClass>com.gnosis.cuteoverlay.Application</mainClass>
                <reflectionList>
                        <list>com.gnosis.cuteoverlay.Application</list>
                        <list>com.gnosis.cuteoverlay.DataToSend</list>
                        <list>com.gnosis.cuteoverlay.DataToSendContainer</list>
                </reflectionList>
                <graalvmHome>C:\dev\tools\graalvm-svm-windows-gluon-21.2.0-dev</graalvmHome>
                <nativeImageArgs>
                        <imageArg>-Dio.netty.noUnsafe=true</imageArg>
                        <imageArg>--report-unsupported-elements-at-runtime</imageArg>
                        <imageArg>--allow-incomplete-classpath --enable-all-security-services</imageArg>
                        <imageArg>--enable-url-protocols=https -H:EnableURLProtocols=http</imageArg>
                        <imageArg>-H:TraceClassInitialization=true</imageArg>
                        <imageArg>--initialize-at-build-time=${buildtime-classes}</imageArg>
                        <imageArg>--initialize-at-run-time=${runtime-classes}</imageArg>
                </nativeImageArgs>
        </configuration>
</plugin>
```

Ana zaman harcama sorunu, derleme ve çalışma zamanı sınıflarını ayrıca Java'nın Reflection ı tarafından kullanılanları sınıflara ayırmaktır. Ayrıca bu listeler hangi kütüphaneleri kullandığınıza bağlıdır.

Birkaç denemeden sonra exe başarıyla hazırlandı.

```
#yerel çalıştıralabilir dosyayı hazırlar
mvn gluonfx:build
```

Bundan sonra, exeyi ve iş mantığını çalıştırmak için ihtiyaç duyduğum diğer tüm yürütülebilir dosyaları kopyaladığım bir dağıtım klasörü oluşturdum.

ve sonunda

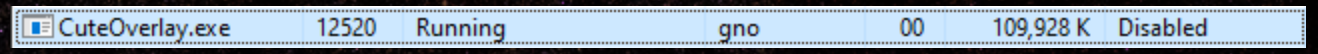

firatgursoy
CUTE OVERLAY
CPU TEMP 55C
GPU TEMP 46C
CPU LOAD 07%
GPU LOAD 00%
MEM LOAD 72%
GPU FAN 99%
Stockfish seviye 5
1 d4 e6
2 c4 ♘c6
3 e3 h5
4 ♘c3 d5
½
firatgursoy 1500?
Sıra sizde

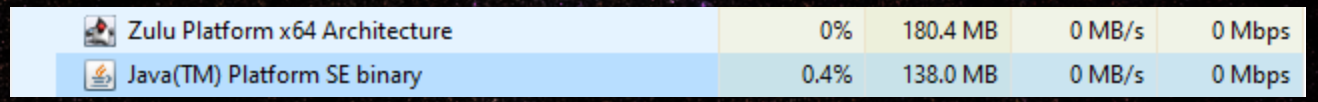
Uygulamadan başlangıç zamanı metrikleri

```
# Jar Log
Uptime:781ms
StartTime:Wed Jun 30 02:02:09 EET 2021
Max heap memory is 4086 MBytes
Used non-heap memory is 32 MBytes
# Native Image(exe) Log
Uptime:1ms
StartTime:Wed Jun 30 02:06:14 TRT 2021
Max heap memory is 0 MBytes
Used non-heap memory is 0 MBytes
```

Yıldırım hızında jvm başlangıç zamanı!

İşletim sisteminde gözüken bellek kullanımları





Bellek kullanımından 200MB tasarruf sağladık!

Özellikle Teşekkürler !

- Oracle, Quarkus, JavaFX / GluonHQ, Spring
- Google and Wiki
- adesso

Kaynak koduna ve windows buildlerine git üzerinden ulaşabilirsiniz.

https://github.com/firatgursoy/CuteOverlay