Amino - A Distributed Runtime for Applications Running Dynamically Across Device, Edge and Cloud

Symposium on Edge Computing 2018

Ying Xiong, *Isaac Ackerman, Quinton Hoole et al*

Seattle Cloud Lab, Huawei R&D USA, Bellevue WA

Problem to solve

How to help develop apps running dynamically across device, edge, and cloud

Related Works

Cuckoo: A framework offers code offloading capability to Android app. Limited to Android; Only offloading is provided.

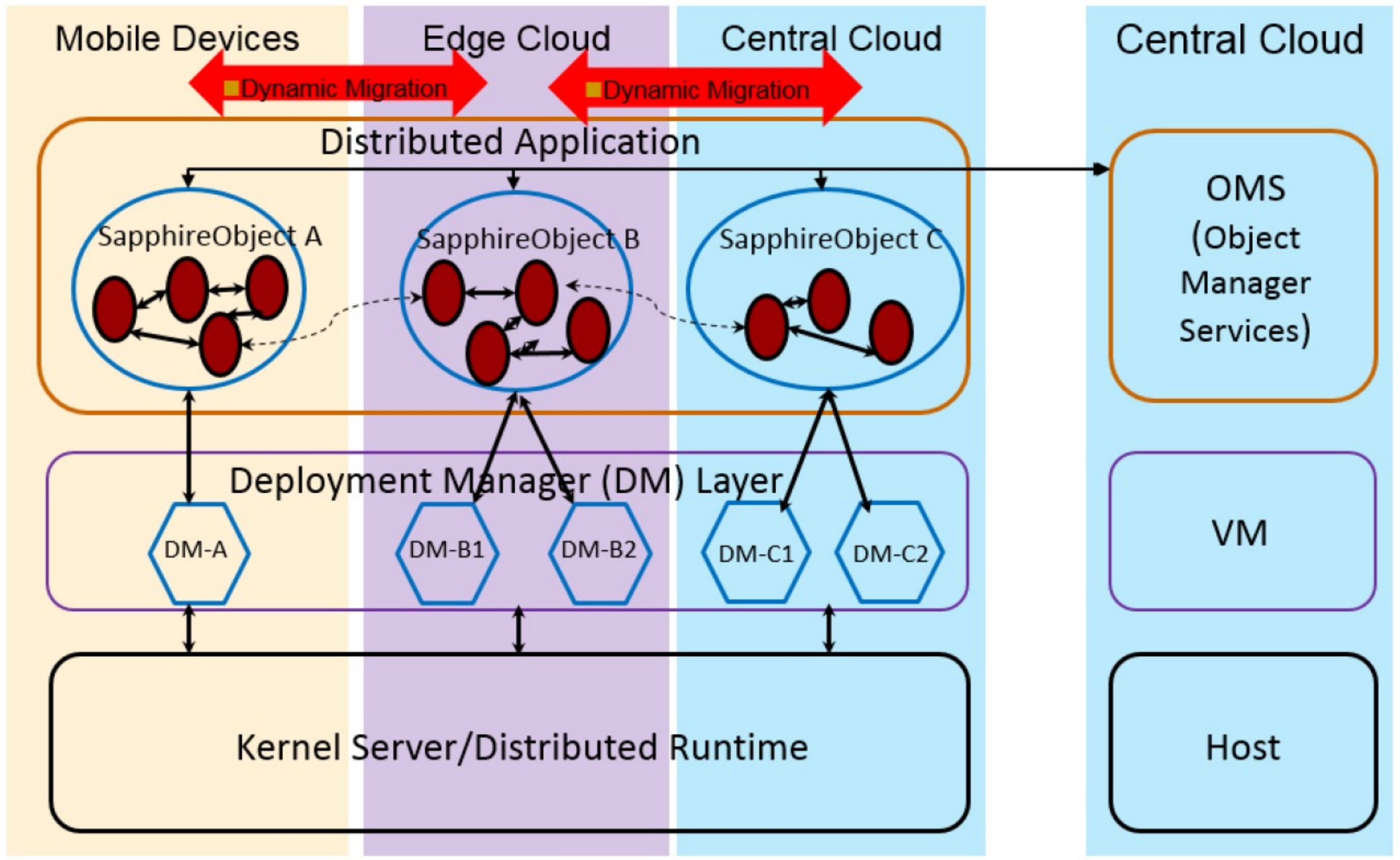
Mobile Fog: A programming model to support apps dynamically scale at runtime. Static partitioning is needed; Does not solve problems like leader election.

CloneCloud:…

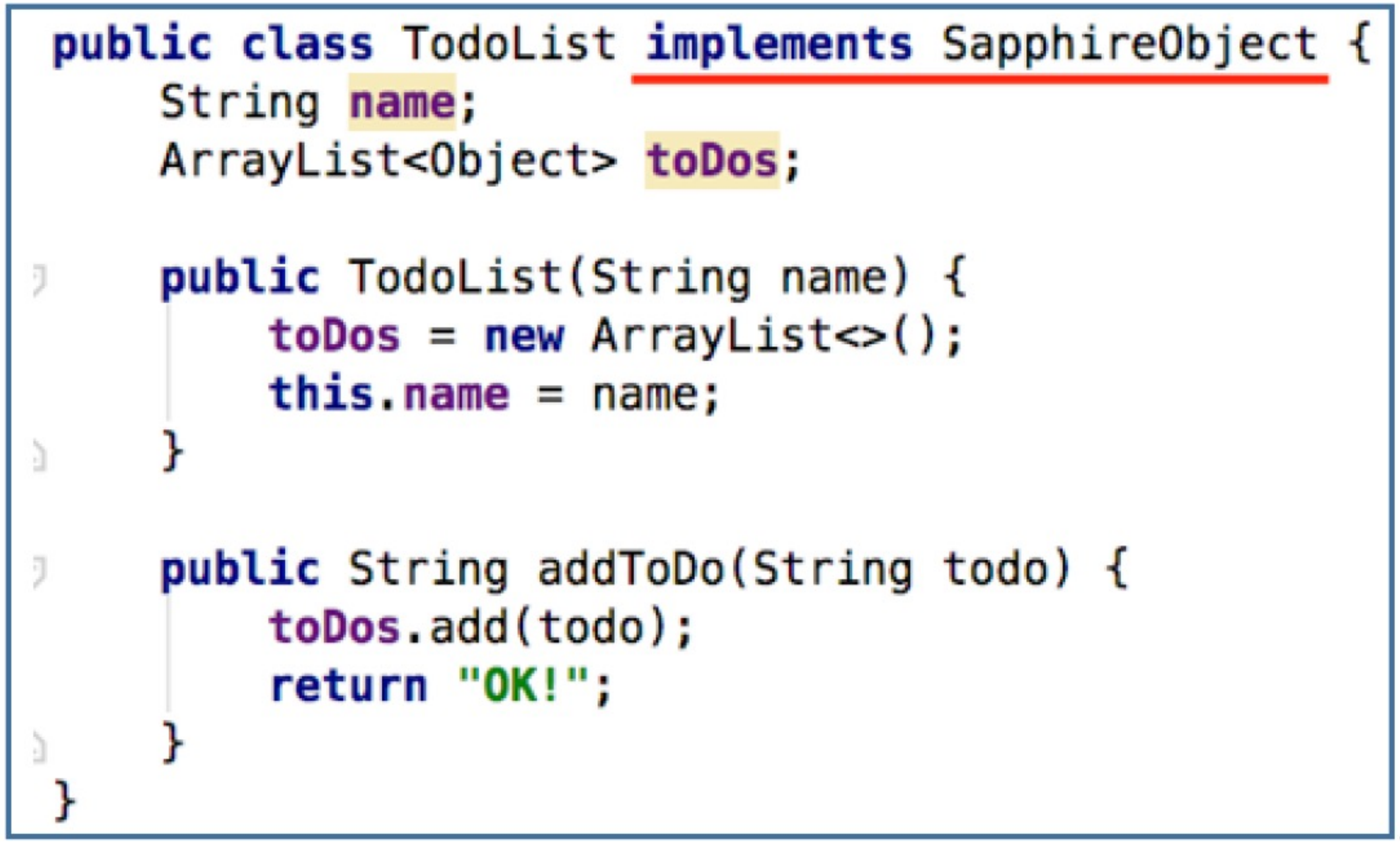
Features of Amino

1. Code offloading.
2. Multiple languages supported.
3. Solving common distributed system problems for programmers: fault-tolerance, caching, …

Layered Architecture of Amino



Define a Sapphire Object

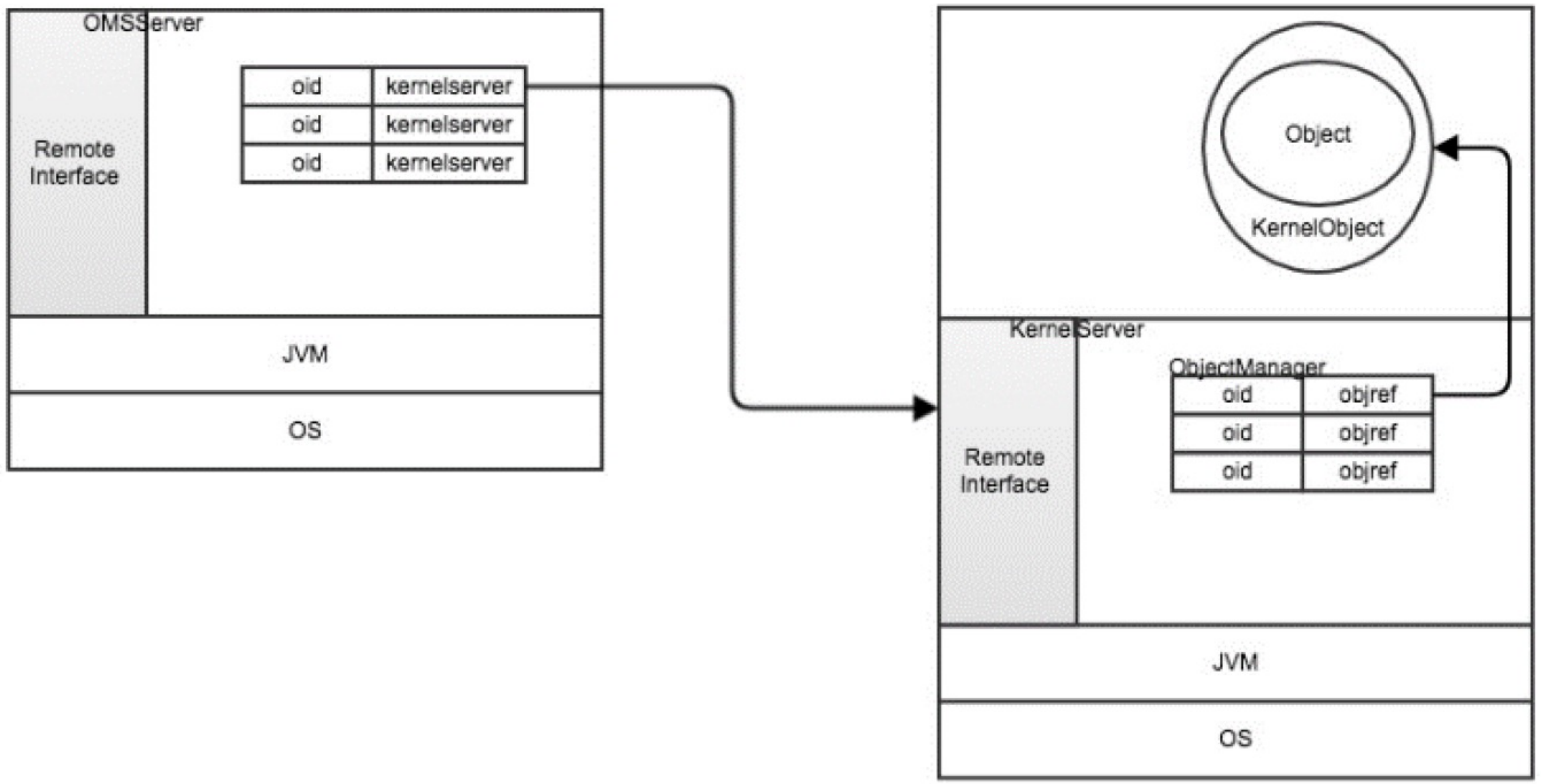


```
public class TodoList implements SapphireObject {
    String name;
    ArrayList<Object> toDos;

    public TodoList(String name) {
        toDos = new ArrayList<>();
        this.name = name;
    }

    public String addToDo(String todo) {
        toDos.add(todo);
        return "OK!";
    }
}
```

Kernel Server

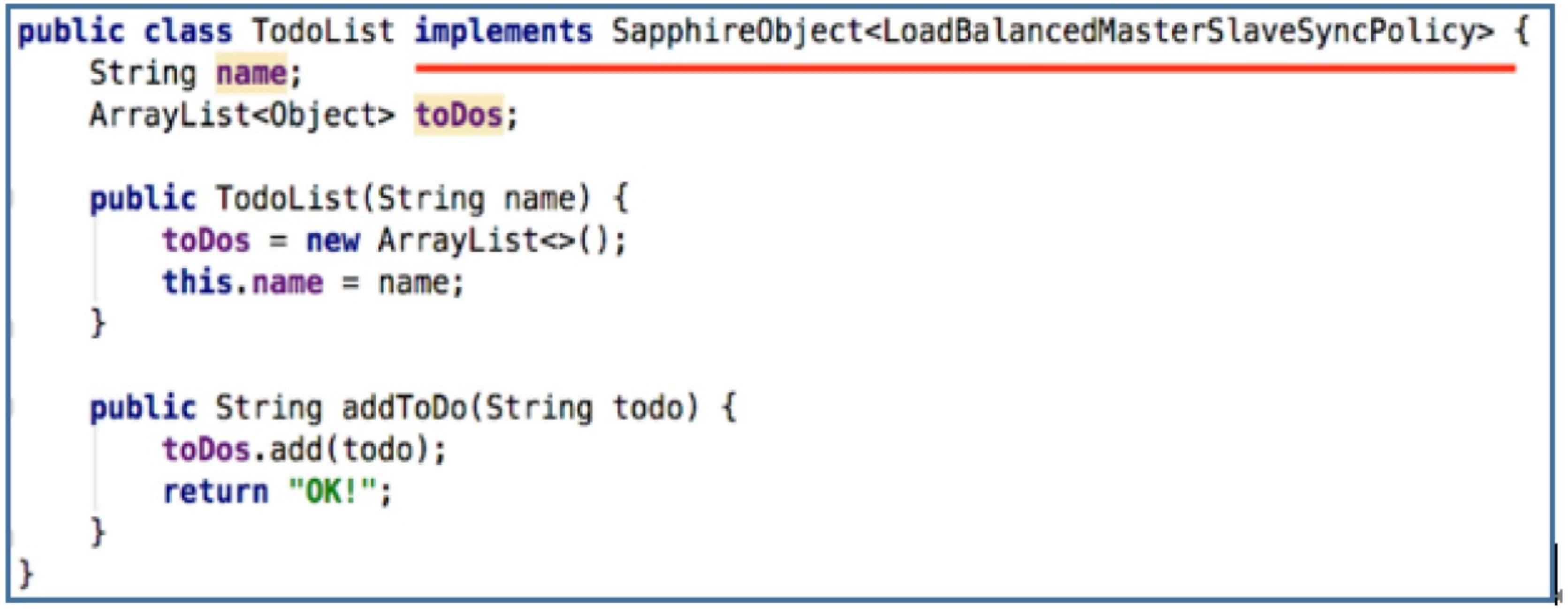
- Expose a set of remote API
- Use Sapphire.new_() to create SO: unique ID
- OMS keeps track of SO

OMS



OMSServer
Remote Interface
oid
kernelserver
oid
kernelserver
oid
kernelserver
JVM
OS
Object
KernelObject
KernelServer
ObjectManager
oid
objref
oid
objref
oid
objref
Remote Interface
JVM
OS

DM



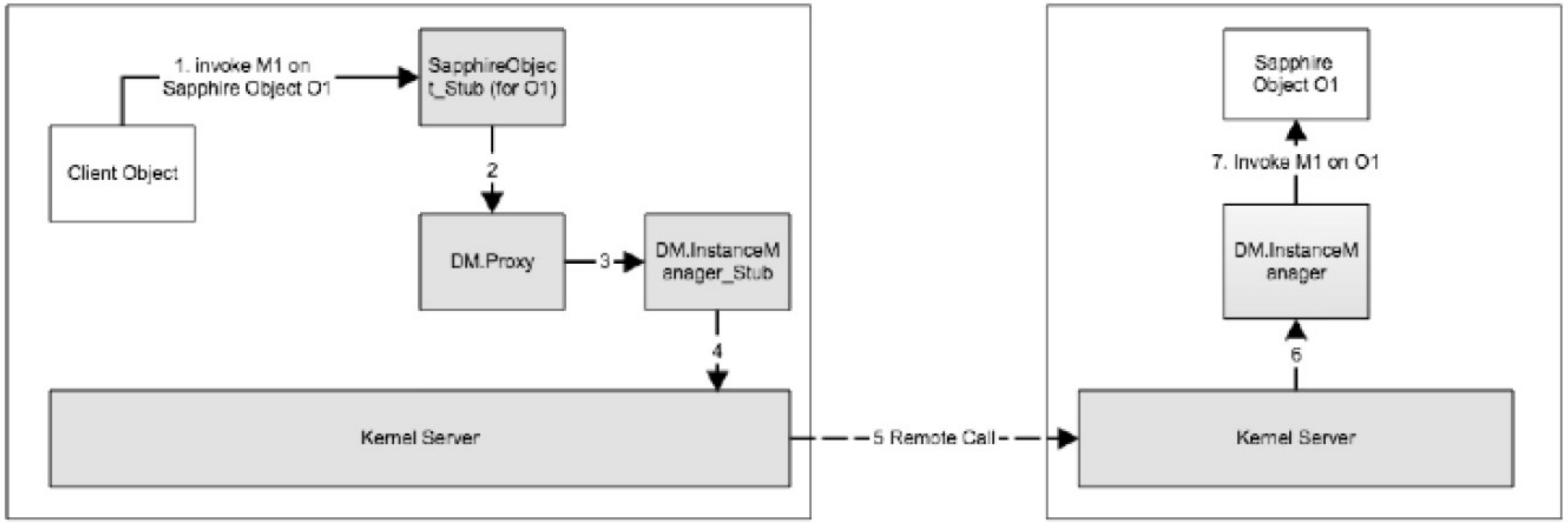
```
public class TodoList implements SapphireObject<LoadBalancedMasterSlaveSyncPolicy> {
    String name;
    ArrayList<Object> toDos;

    public TodoList(String name) {
        toDos = new ArrayList<>();
        this.name = name;
    }

    public String addToDo(String todo) {
        toDos.add(todo);
        return "OK!";
    }
}
```

DM

Three components:

1. Proxy
2. Instance manager
3. coordinator

DM


1. invoke M1 on Sapphire Object O1
SapphireObject_Stub (for O1)
Client Object
2
DM.Proxy
3
DM.InstanceManager_Stub
4
Kernel Server
5 Remote Call
Kernel Server
6
DM.InstanceManager
7. Invoke M1 on O1
Sapphire Object O1

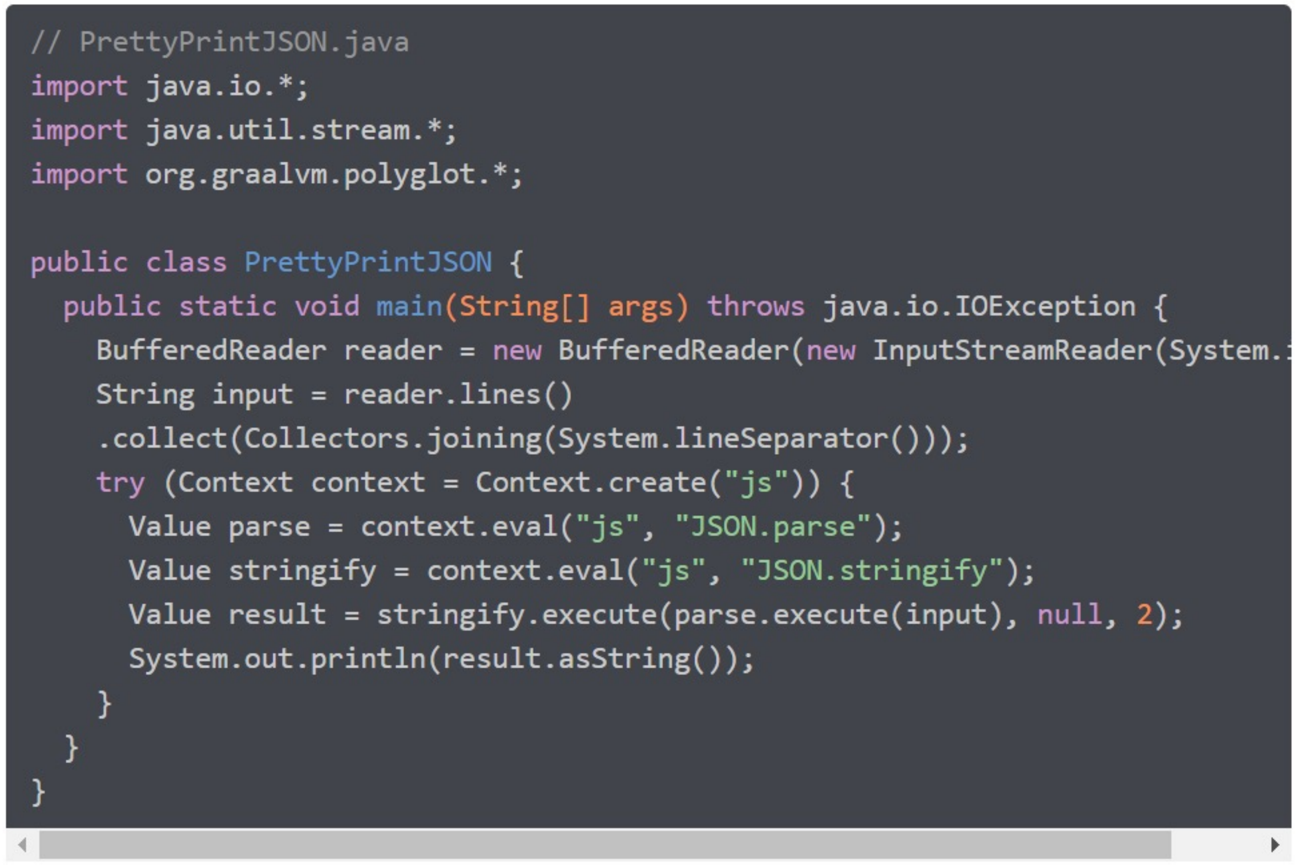
Multi-Language Support

GraalVM

- https://www.graalvm.org/

- A high-performance JDK distribution. It is designed to accelerate the execution of applications written in Java and other JVM languages while also providing runtimes for JavaScript, Ruby, Python, and a number of other popular languages.

Example of using GraalVM



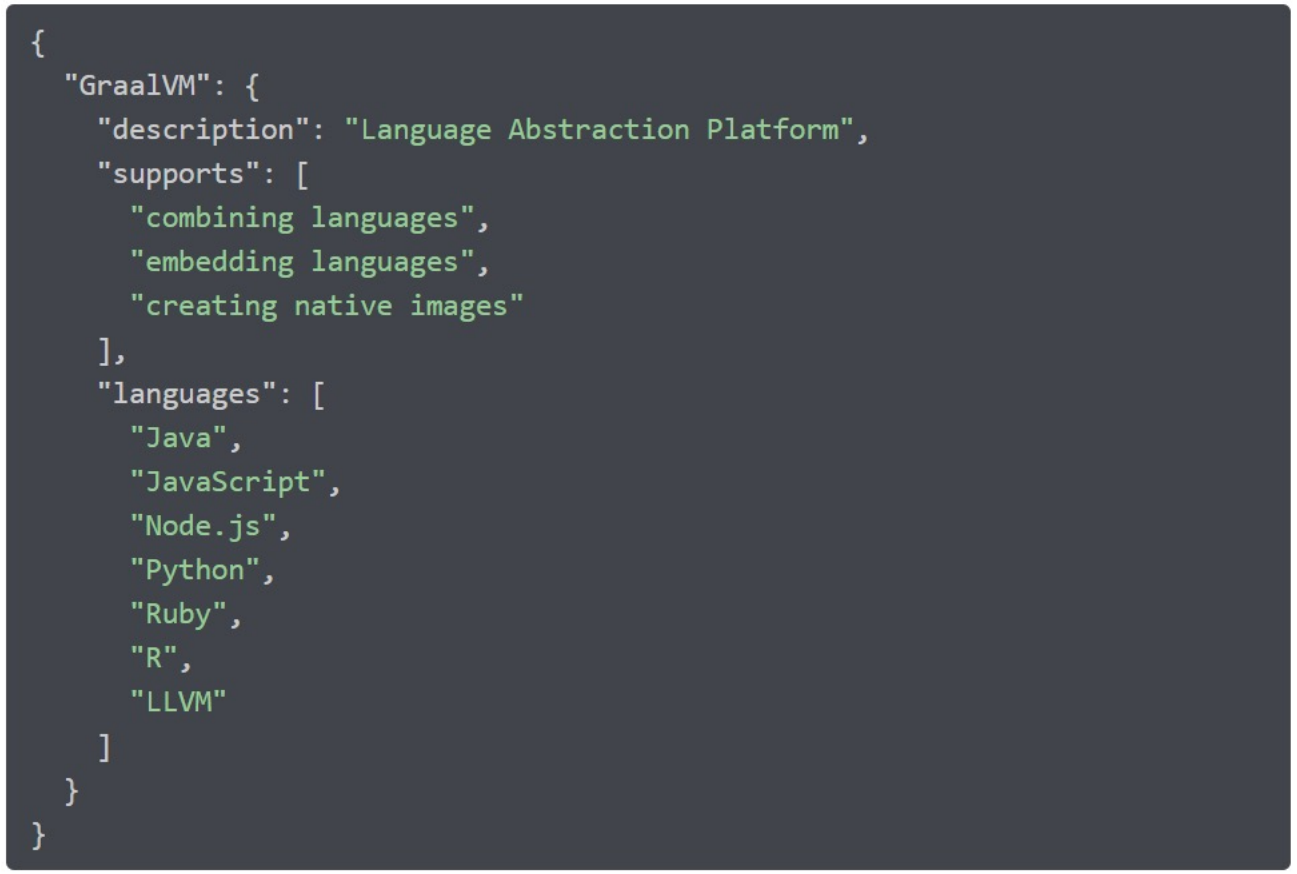
```java
// PrettyPrintJSON.java
import java.io.*;
import java.util.stream.*;
import org.graalvm.polyglot.*;

public class PrettyPrintJSON {
  public static void main(String[] args) throws java.io.IOException {
    BufferedReader reader = new BufferedReader(new InputStreamReader(System.
    String input = reader.lines()
    .collect(Collectors.joining(System.lineSeparator()));
    try (Context context = Context.create("js")) {
      Value parse = context.eval("js", "JSON.parse");
      Value stringify = context.eval("js", "JSON.stringify");
      Value result = stringify.execute(parse.execute(input), null, 2);
      System.out.println(result.asString());
    }
  }
}
```

Example of using GraalVM

```
javac PrettyPrintJSON.java
native-image --language:js --initialize-at-build-time PrettyPrintJSON
```

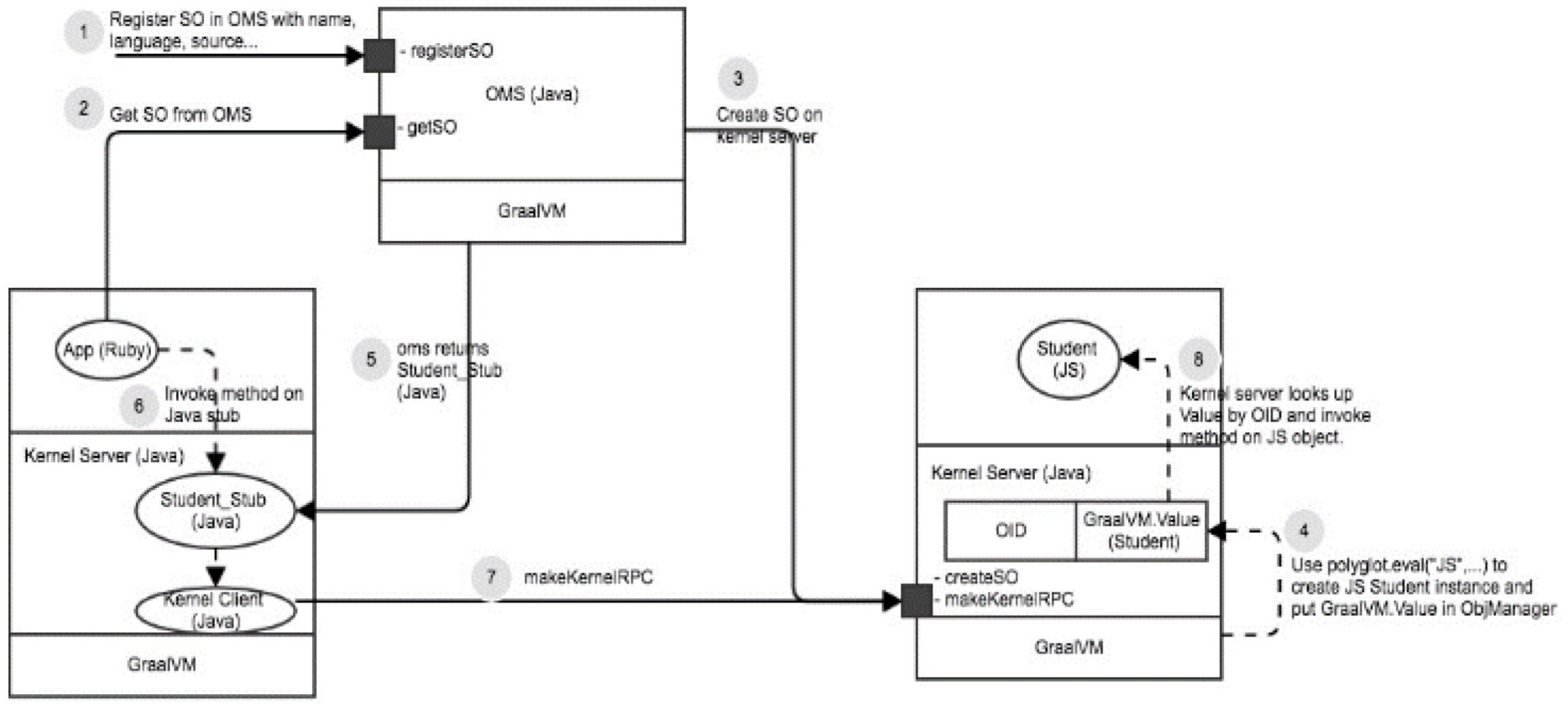
```
./prettyprintjson <<EOF
{"GraalVM":{"description":"Language Abstraction
Platform","supports":["combining languages","embedding
languages","creating native images"],"languages":
["Java","JavaScript","Node.js", "Python", "Ruby","R","LLVM"]}}
EOF
```

Example of using GraalVM



```
{
  "GraalVM": {
    "description": "Language Abstraction Platform",
    "supports": [
      "combining languages",
      "embedding languages",
      "creating native images"
    ],
    "languages": [
      "Java",
      "JavaScript",
      "Node.js",
      "Python",
      "Ruby",
      "R",
      "LLVM"
    ]
  }
}
```

GraalVM in Amino



1
Register SO in OMS with name, language, source...
- registerSO
OMS (Java)
2
Get SO from OMS
- getSO
GraalVM
3
Create SO on kernel server
App (Ruby)
5
oms returns Student_Stub (Java)
6
Invoke method on Java stub
Kernel Server (Java)
Student_Stub (Java)
Kernel Client (Java)
GraalVM
7
makeKernelRPC
Student (JS)
8
Kernel server looks up Value by OID and invoke method on JS object.
Kernel Server (Java)
OID
GraalVM.Value (Student)
4
Use polyglot.eval("JS",...) to create JS Student instance and put GraalVM.Value in ObjManager
- createSO
- makeKernelRPC
GraalVM

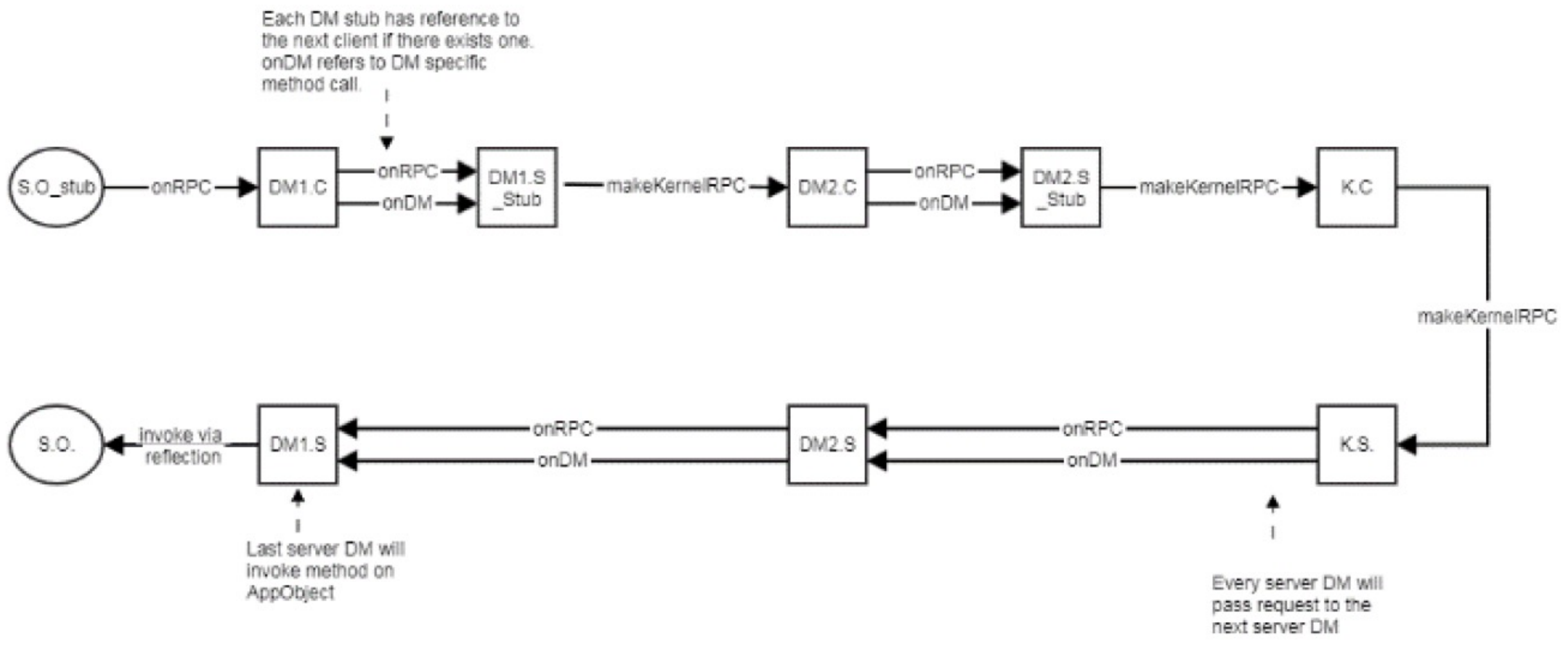
GraalVM in Amino

- Kernel Server, OMS and all DMs are still in Java
- Sapphire objects can be written in different languages, but corresponding stubs are in Java
- Kernel Server uses Graal API polyglot.eval() to create Sapphire object instances, and saves sapphire object as Graal Value instance (polyglot.Value) in Object Manage
- It is possible for Object Manager to store server policies that refer to the Graal value
- Kernel Server uses Graal API(polyglot.Value.getMembers(…).execute()) to invoke methods on the Sapphire object.

DM Chain

- Distributed Hashing Table (DHT) DM
- Consensus DM
- DHT & Consensus
- …

DM Chain


Each DM stub has reference to the next client if there exists one. onDM refers to DM specific method call.
S.O_stub
onRPC
DM1.C
onRPC
onDM
DM1.S _Stub
makeKernelRPC
DM2.C
onRPC
onDM
DM2.S _Stub
makeKernelRPC
K.C
makeKernelRPC
K.S.
onRPC
onDM
DM2.S
onRPC
onDM
DM1.S
invoke via reflection
S.O.
Last server DM will invoke method on AppObject
Every server DM will pass request to the next server DM

Code Offloading

- For each Sapphire object associated with offloading DM, OMS generates client and server stubs that calculate the execution time of each method invocation on the object instance running on a given host (edge device or cloud server), t(Object, host).

- For each object invocation on a server from a client, the generated DM stubs measure the latency and bandwidth (transmit time) of the link between calling client to the object host, l(object, client, host).

Code Offloading

- For each Sapphire object (configured with Offloading DM) invocation, the generated server DM stub measures the cpu, memory and IO resource consumption as well as energy consumption on a given host, r(object, host) = f(cpu, mem, disk io, network io) and e(object, host) = f (r), where r() and e() are the resource utilization and energy consumption of a given object on a given host respectively.

Code Offloading

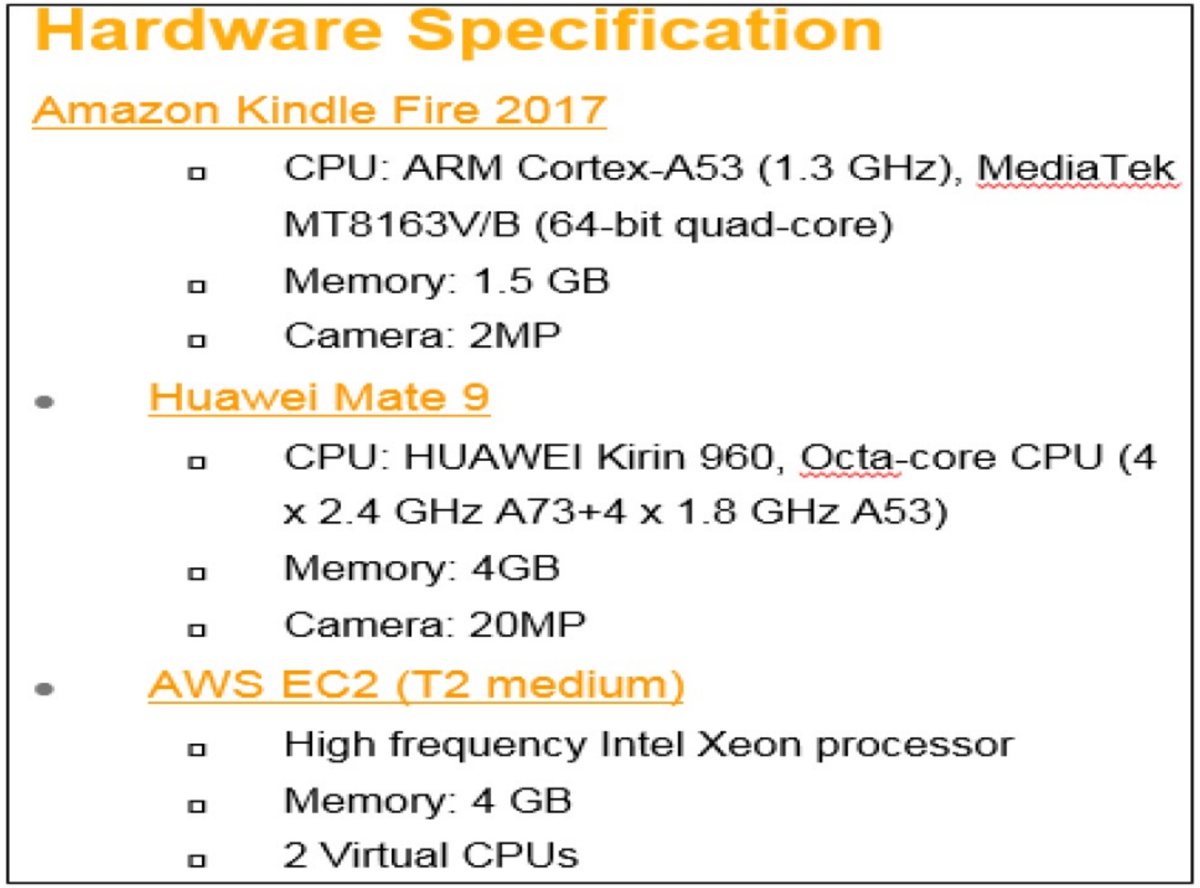
$$V(Host_i) = \sum_{k=1}^{n} \Big(t\big(Obj_{k,}\ Host_i\big) + l\big(Obj_{k,}\ Client, Host_i\big) + e(Obj_{k,}\ Host_i)\Big)$$

Experiments

- Go Game on Android: 5x faster

- License plate recognition

License plate recognition



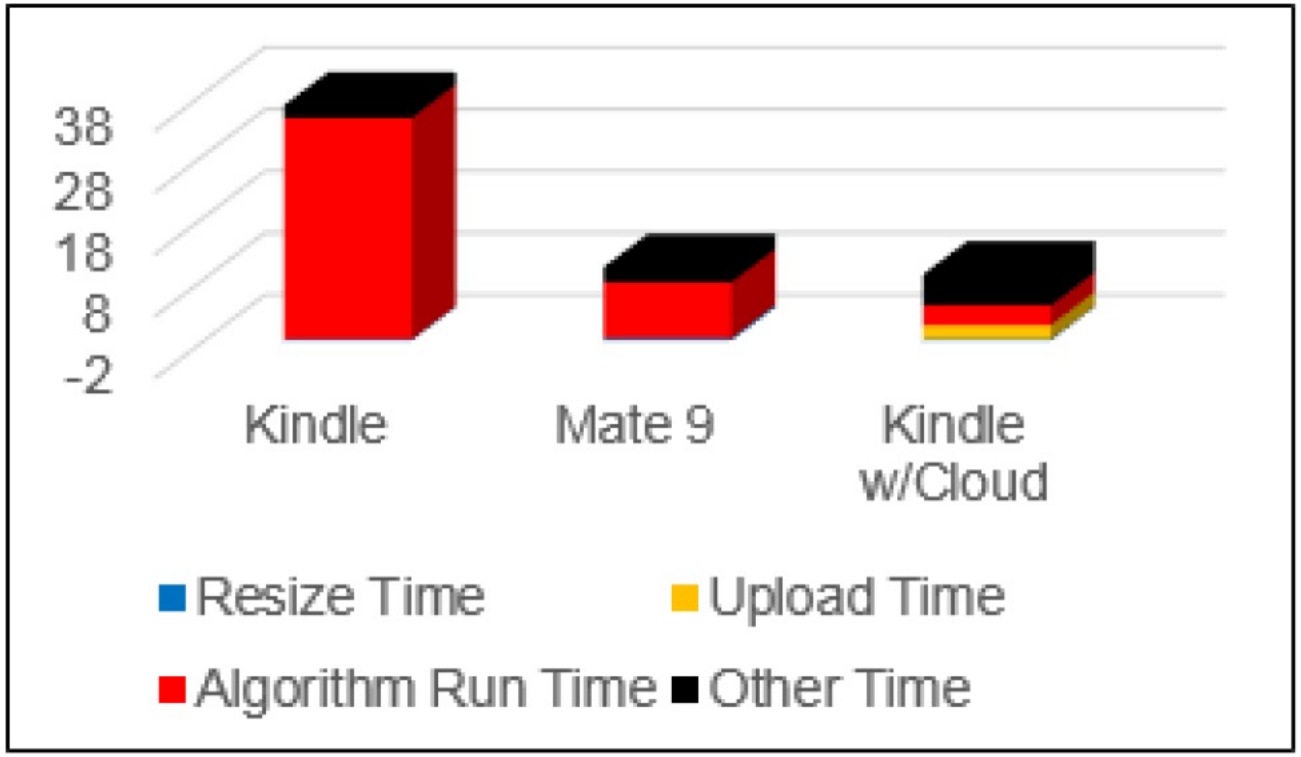
Hardware Specification

Amazon Kindle Fire 2017

- CPU: ARM Cortex-A53 (1.3 GHz), MediaTek MT8163V/B (64-bit quad-core)
- Memory: 1.5 GB
- Camera: 2MP

- Huawei Mate 9
  - CPU: HUAWEI Kirin 960, Octa-core CPU (4 x 2.4 GHz A73+4 x 1.8 GHz A53)
  - Memory: 4GB
  - Camera: 20MP
- AWS EC2 (T2 medium)
  - High frequency Intel Xeon processor
  - Memory: 4 GB
  - 2 Virtual CPUs

License plate recognition


Kindle
Mate 9
Kindle w/Cloud
Resize Time
Upload Time
Algorithm Run Time
Other Time