A World without
Out-Of-Memory

a.k.a. Elastic Memory in the Cloud

Jingjing Wang
Magdalena Balazinska
Big-Data Analytics

```plaintext
-- Invariant: val = 2^exp
x = [1 as val, 0 as exp];
do
  x = [from x emit val*2 as val, exp+1 as exp];
while [from x emit exp < 5];
store(x, powersOfTwo);
```
Wait...
Wait...

```plaintext
-- Invariant: val = 2^exp
x = [1 as val, 0 as exp];
do
  x = [from x emit val*2 as val, exp+1 as exp];
while [from x emit exp < 5];
store(x, powersOfTwo);
```
Wait…

```
1547  -- Invariant: val = 2^exp
    x = [1 as val, 0 as exp];
    do
        x = [from x emit val*2 as val, exp+1 as exp];
    while [from x emit exp < 5];
    store(x, powersOfTwo);
```
java.lang.OutOfMemoryError: Java heap space
    at java.util.Arrays.copyOf(Arrays.java:2271)
    at java.io.ByteArrayOutputStream.grow(ByteArrayOutputStream.java:113)
    at java.io.ByteArrayOutputStream.ensureCapacity(ByteArrayOutputStream.java:93)
    at java.io.ByteArrayOutputStream.write(ByteArrayOutputStream.java:140)
    at java.io.ObjectOutputStream$BlockDataOutputStream.drain(ObjectOutputStream.java:1188)
    at java.io.ObjectOutputStream$BlockDataOutputStream.flushBlock(ObjectOutputStream.java:347)
    at akka.serialization.JavaSerializer$$anonfun$toBinary$1.apply$mcVsp(Serializer.scala:57)
    at akka.serialization.JavaSerializer$$anonfun$toBinary$1.apply(Serializer.scala:129)
    at akka.serialization.JavaSerializer$apply$mcVsp(JavaSerializer.scala:129)
    at scala.util.DynamicVariable$$anonfun$apply$mcVsp$1.apply(DynamicVariable.scala:57)
    at scala.util.DynamicVariable$$anonfun$apply$mcVsp$1.apply(DynamicVariable.scala:57)
    at akka.remote.MessageSerializer$$anonfun$apply$mcVsp$1.apply(MessageSerializer.scala:36)
    at akka.remote.MessageSerializer$$anonfun$apply$mcVsp$1.apply(MessageSerializer.scala:845)
    at akka.remote.EndpointActor$apply$mcVsp$1.apply(EndpointActor.scala:722)
    at akka.actor.Actor$class.aroundReceive(Actor.scala:465)
    at akka.remote.EndpointActor.aroundReceive(Endpoint.scala:415)
    at akka.actor.ActorCell.receiveMessage(ActorCell.scala:516)
    at akka.actor.ActorCell.invoke(ActorCell.scala:487)
    at akka.dispatch.Mailbox.processMailbox(Mailbox.scala:238)
    at akka.dispatch.Mailbox.run(Mailbox.scala:220)
Memory: Whose Responsibility

• User’s:
  • Claim resources from service provider
  • Pay for them
  • Run the application

• Provider’s:
  • Shared resources
  • Users/Applications with SLAs
  • Put them in containers, schedule them in a smart way
How Much Memory to Allocate

- Allocate more:
  - Waste resource

- Allocate less:
  - Out-Of-Memory
  - Performance degradation due to Garbage Collections

- Problem: precise estimation before execution is hard
Solution: Be Elastic

• Change memory quota on-the-fly
  • Cloud, flexible resource (within budget)

• Use the best strategy that benefits us
• Real-time performance characteristics
Work-In-Progress

- Decisions to allocate memory among applications
  - Allocate more memory on the same machine
  - Add machines
  - Kill them
- Ability to change the memory quota of a container
  - JVMs are black boxes once launched
    - Need to hack
- Cost model
  - Predict application behavior in terms of resources
  - GC time given memory quota & application state
  - Data analytics
A World without Out-Of-Memory