CloudButton

WP4 - Support for Mutable Shared Data in Serverless Computing

Final review

September 15, 2022

Pierre Sutra

Associate Professor @ Institut Mines-Télécom

Challenges

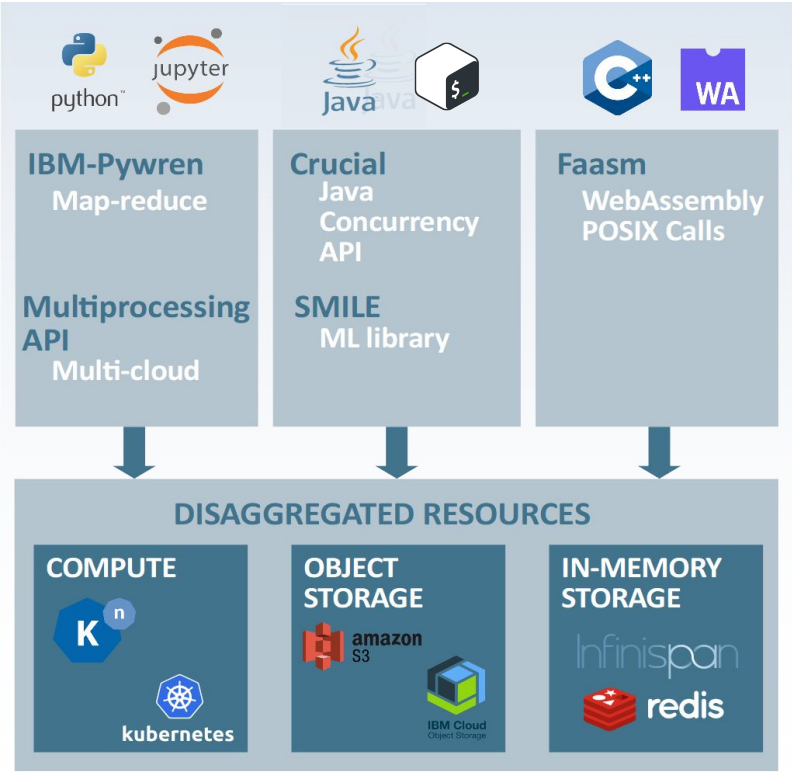
Support for Mutable Shared Data in Serverless Computing

- client tier = programming framework for stateful serverless computing

  high-level constructs (OOP)

- server tier = distributed data store

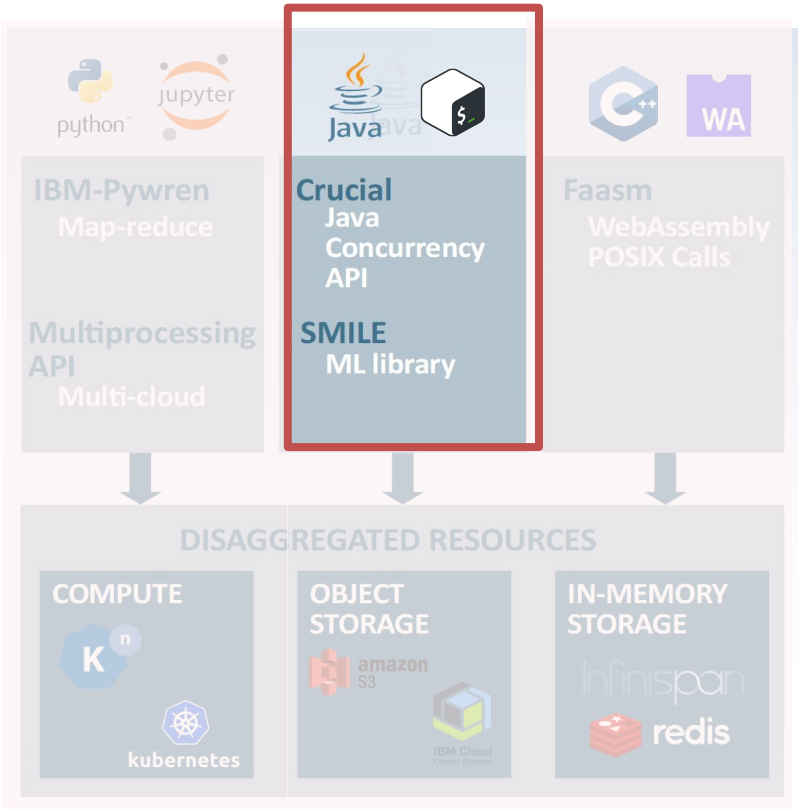
  elastic, dependable, ephemeral/durable, just-right consistent, dynamic & auto-adaptable (data locality, replication)



Client tier (T4.1)

Offer means to manipulate mutable shared data with serverless functions

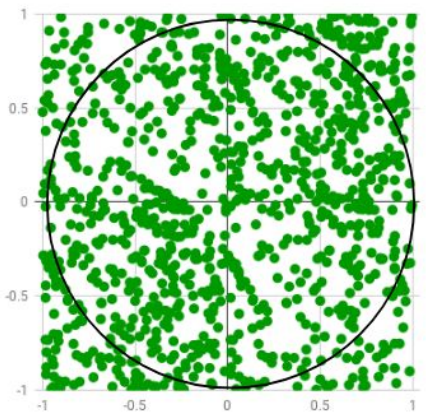
- API = multithreaded programming
- synchronization primitives (e.g., lock, barrier)
- callable @shared objects (Java)
- support for Python *(Lithops)* and shell
- application to ML, web crawling, big data benchmarks (amplab, YCSB)



Crucial*

multithreaded programming for serverless


```
int counter = 0;
for (long i = 0L; i < 100; i++)
    if ( Math.pow(Math.random(),2) +
            Math.pow(Math.random(),2)<= 1.0) counter++;
```



** formally, Blossom in DoW*

Crucial

multithreaded programming for serverless


```
ExecutorService service = Executors.newCachedThreadPool();
AtomicInteger cnt = new AtomicInteger();
service.submit((Callable) () -> {
    for (long i = 0L; i < 100; i++)
        if ( Math.pow(Math.random(),2) +
                Math.pow(Math.random(),2)<= 1.0) cnt.getAndIncrement();
    return null;
});
```

Crucial

multithreaded programming for serverless


```
ExecutorService service = Executors.newCachedThreadPool();
AtomicInteger cnt = new AtomicInteger();
service.invokeAll(
        (Collection<? extends Callable<Void>>)
        IntStream.range(0,100).mapToObj(n ->
            (Callable<Void>) () -> {
                for (long i = 0L; i < 100; i++)
                    if (Math.pow(Math.random(), 2) +
                            Math.pow(Math.random(), 2) <= 1.0) cnt.getAndIncrement();
```

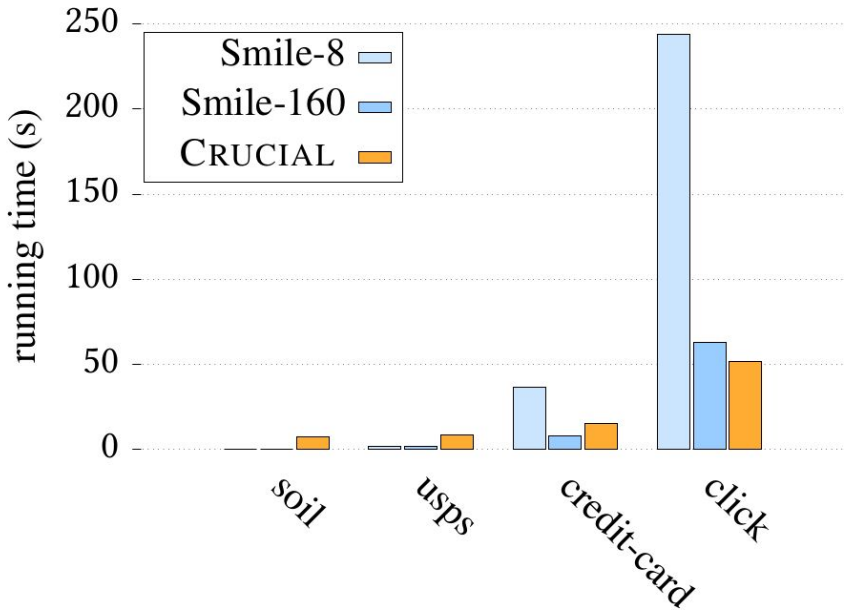
Crucial

multithreaded programming for serverless


```
service = new AWSLambdaExecutorService();
org.crucial.dso.AtomicLong count= new org.crucial.dso.AtomicLong(0);
service.invokeAll(
        (Collection<? extends Callable<Void>>)
        IntStream.range(0,100).mapToObj(n ->
            (Callable<Void>) () -> {
                for (long i = 0L; i < 100; i++)
                    if (Math.pow(Math.random(), 2) +
                            Math.pow(Math.random(), 2) <= 1.0) cnt.getAndIncrement();
```

[TOSEM'22, Middleware'19]

Smile ML library



random forest classification - training phase

- one task per decision tree
- smile-160 = *160-threads server-class machine*
- smile-8 = *laptop*
- crucial = *serverless*

Takeaways:

- serverless performance on par w. high-end server
- few modifications (<4%)

Serverless shell

example: what is the average web page size?

Serverless shell

Common Crawl

example: what is the average web page size?

```
average(){
    while read l; do
        sshell "curl -s ${RANGE} ${CCBASE}/${l} | zcat -q | grep ^Content-Length " &
    done < ${TMP_DIR}/index | awk '{ sum += $2 } END { if (NR > 0) print int(sum / NR) }'
}
```

web index (PBs, +4B pages/month)

Serverless shell

example: what is the average web page size?

```
average(){
    while read l; do
        sshell "curl -s ${RANGE} ${CCBASE}/${l} | zcat -q | grep ^Content-Length " &
    done < ${TMP_DIR}/index | awk '{ sum += $2 } END { if (NR > 0) print int(sum / NR) }'
}
```

cloud function (AWS lambda, knative, k8s job, etc.)

Serverless shell

example: what is the average web page size?

```
average(){
    while read l; do
        sshell "curl -s ${RANGE} ${CCBASE}/${l} | zcat -q | grep ^Content-Length " &
    done < ${TMP_DIR}/index | awk '{ sum += $2 } END { if (NR > 0) print int(sum / NR) }'
}
```

```
$> average
130411
```

[Middleware'21]

Serverless shell

example: what are the most popular web domain?

- original LinkRun solution

- sshell vs. LinkRun

	SLOC	Pricing	Time	Dataset (compressed)
Linkrun	716	$200-260	26-48h	17.62 TB
sshell	51	$19	28mn	20.17 TB

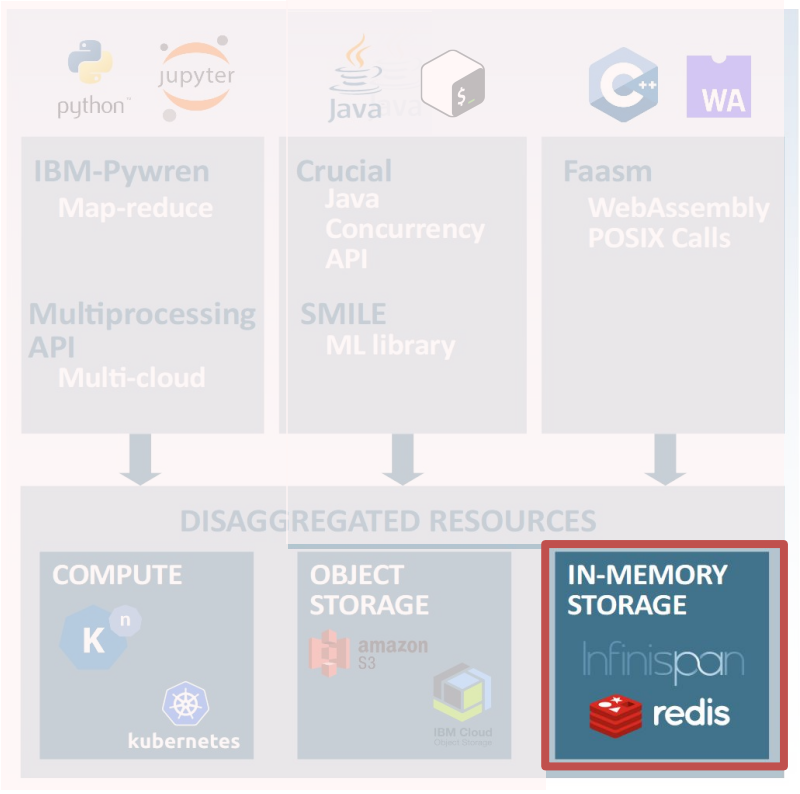
Takeaways:

- on-demand + scalable resources allocation
- faster (cheaper) than original solution

Server tier

In-memory storage with smart capabilities

- elastic, dependable, ephemeral/durable, just-right consistent, dynamic & auto-adaptable (data locality, replication)



Toward zero-friction storage (T4.4)

Objective: adjust the storage tier to serverless computing

Results:

- anchored keys
  (avoid rebalancing data when scaling up)

- compilation to native
  (kick start a server in milliseconds)

- kubernetes operators
  (drive elastic scaling based on demand)

Data sharing, synchronization (T4.3)

Objective: reduce cost of data sharing & synchronization

Results:

- leaderless state-machine replication protocols
  [Eurosys'20, IPL'20, DISC'20]

- partial replication
  [Eurosys'21, DSN'19, PODC'22, DISC'22]

Data persistence (T4.4)

Objective: add language support for NVRAM (Intel Optane)

Results:

- J-NVM = new programming framework for Java
- durable linearizable data types

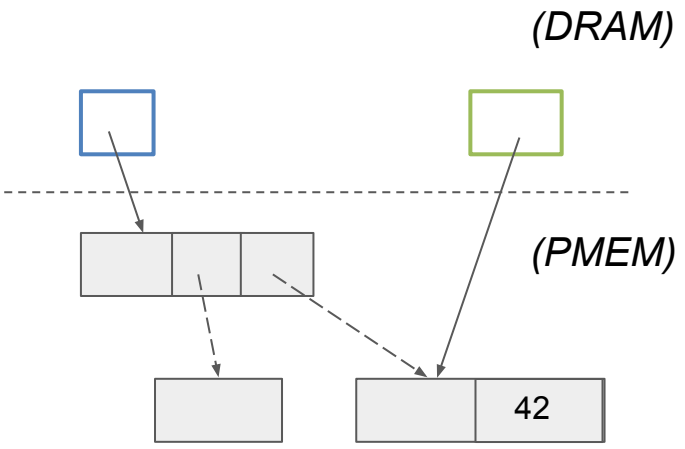
[SOSP'21]

Data persistence

Persistent object is
- a persistent data structure *(off-heap)*
  - holds object fields
- a proxy *(on-heap)*
  - holds object methods
  - implements PObject interface
  - intermediates access to pers. data structure
  - instantiated lazily (low GC pressure)

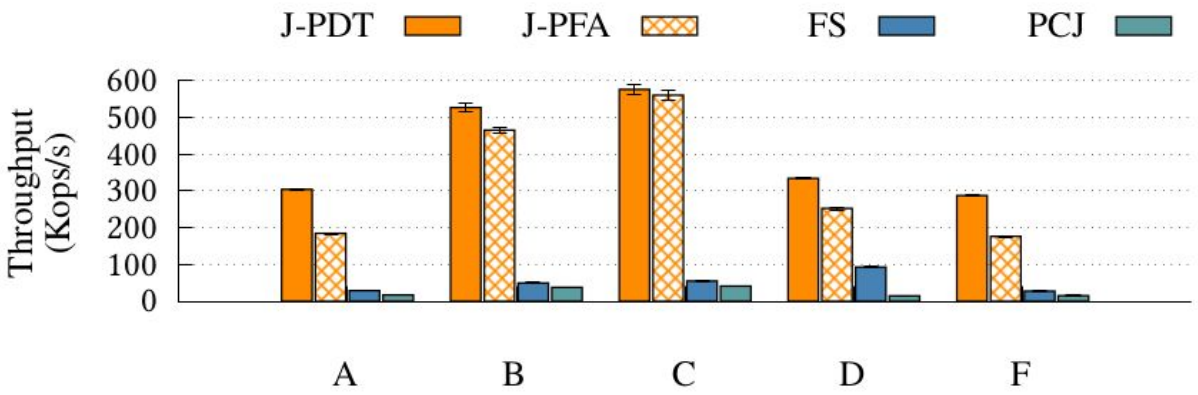
Alive when reachable (from persistent root)

Class-centric model
- safe references thanks to the type system




```
Map root = JNVM.root();
Simple s = root.get("Simple");
s.setX(42);
```

Data persistence



Durable backends for Infinispan:

- PCJ = HashMap from Persistent Collections Java (JNI + PMDK)
- FS: ext4-dax

Hardware used:
4 Intel CLX 6230 HT 80-core
128GB DDR4,
4x128GB Optane (gen1)

Takeaways:

- J-NVM up to *10.5x* (resp. *22.7x)* than FS (resp. PCJ)
- no need for volatile cache

Software

List of contributed softwares

- Infinispan *(anchored keys, k8s operator, native compilation)*
  https://infinispan.org

- Crucial *(w. serverless shell)*
  https://github.com/crucial-project

- J-NVM *(pmem for Java)*
  https://github.com/jnvm-project

- Synchronization protocols
  https://github.com/vitorenesduarte/fantoch

CloudButton

THANK YOU!

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825184.