15th ANNUAL WORKSHOP 2019

A DATA STORE FOR RESOURCE EFFICIENT SERVERLESS COMPUTING

Bernard Metzler, Adrian Schuepbach, Patrick Stuedi, Jonas Pfefferle

IBM Zurich Research

[March, 2019]

Agenda

- Why Serverless Computing

- Current limitations and challenges

- Handling ephemeral data

- How to integrate an ephemeral data store?

- A prototype integration

- Summary

A Decade ago: From Managed Clusters to Cloud Computing

The promise of Cloud Computing

- ✓ Virtually infinite computing resources on demand.
- ✓ No up-front commitment by cloud users.
- ✓ Pay for use of computing resources as needed.
- ✓ Economies of scale (large data centers).
- ✓ Simplified operation via resource virtualization.
- ✓ Higher hardware utilization by multiplexing workloads, even from different organizations.

…Cloud simplified operation?

- o From managing physical resources to managing virtual resources
- o Scaling for changing load requirements
- o Resource idling/overprovisioning for stateful services like data bases
- o Monitoring resource health
- o System upgrades,
- o ….

"The data center is now the computer"

Serverless Computing

User

- Writes Cloud Function
- Defines event to trigger running the function

Serverless system

- Server instantiation
- Resource scaling/elasticity … auto-configuration
- Fault tolerance
- Logging
- Maintenance

✓User: No cluster setup/management
✓User: Fine grained, sub-seconds billing
✓Provider: optimize resource utilization

CARL… This is NOT what I meant when I said go Serverless!

AWS Lambda, IBM Cloud Functions, Google Functions, Azure Functions, databricks serverless …

Fundamental Differences to conventional Cloud Computing

The abstraction of executing a piece of code instead of first allocating resources on which to execute that code then.
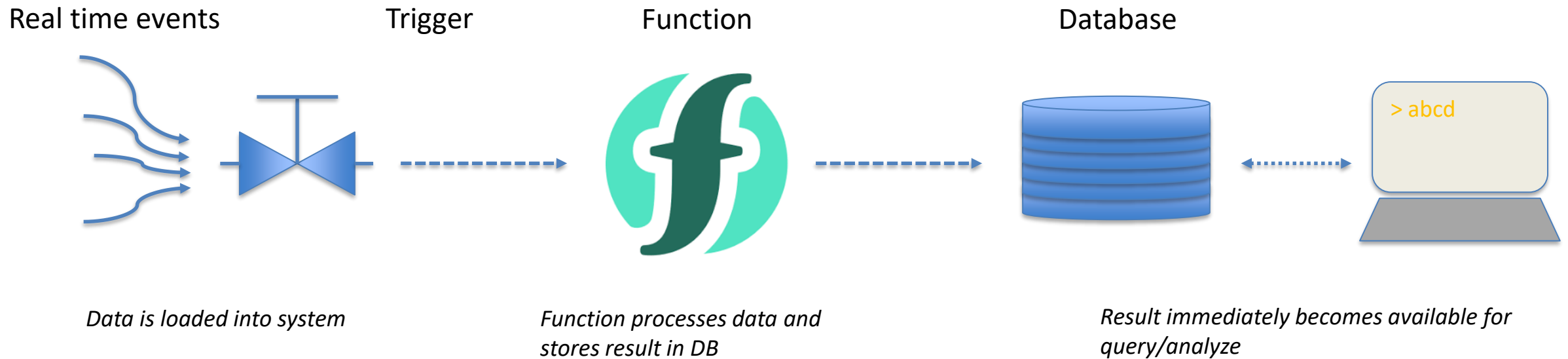
.

.

Paying for the code execution instead of paying for resources allocated to executing the code.

.

.

Decoupling of computation and storage; they scale separately and are priced independently.

A simple Serverless Function

Real time events   Trigger   Function   Database

> abcd

Data is loaded into system   *Function processes data and stores result in DB*   *Result immediately becomes available for query/analyze*

Web applications/backends (IoT, chatbots, …)
Real time data processing (image transcoding, …)

OpenFabrics Alliance Workshop 2019

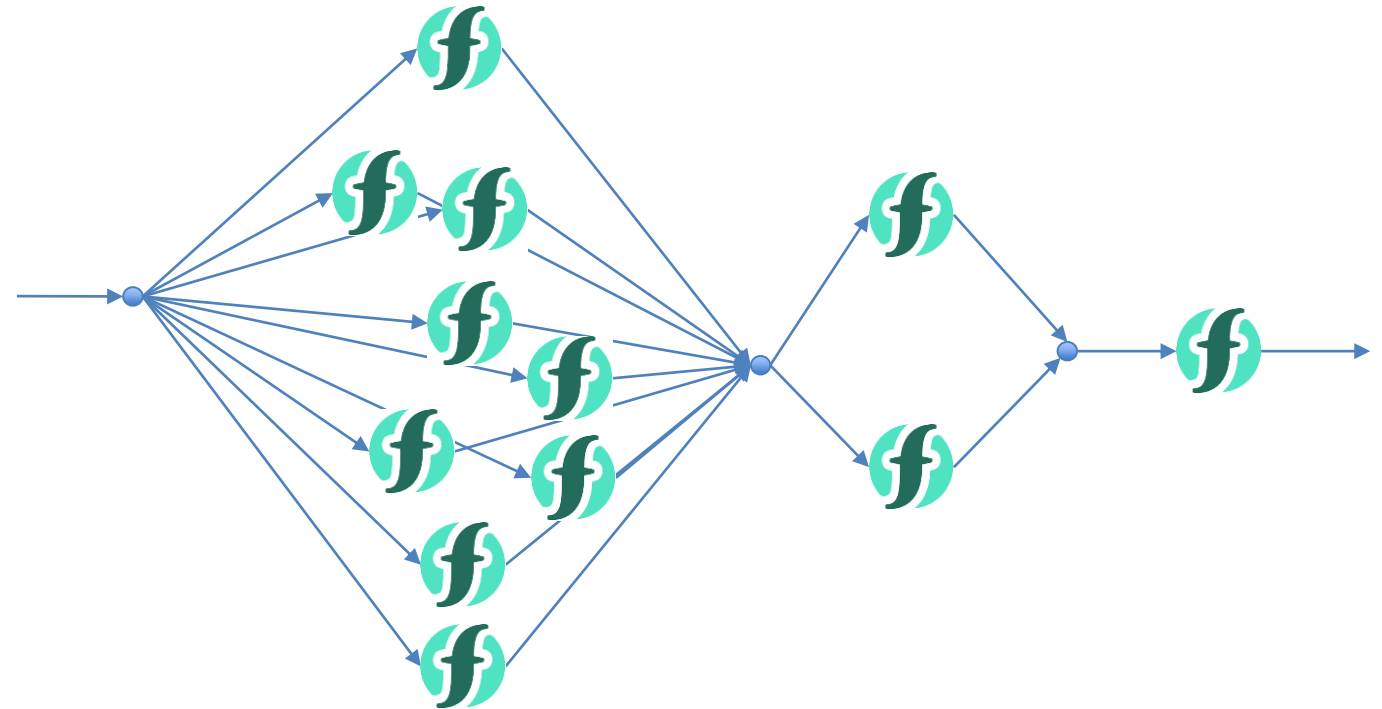# Can we use Serverless Computing for Analytics and ML?

While preserving Serverless Model:

- ✓ Exploit massive parallelism
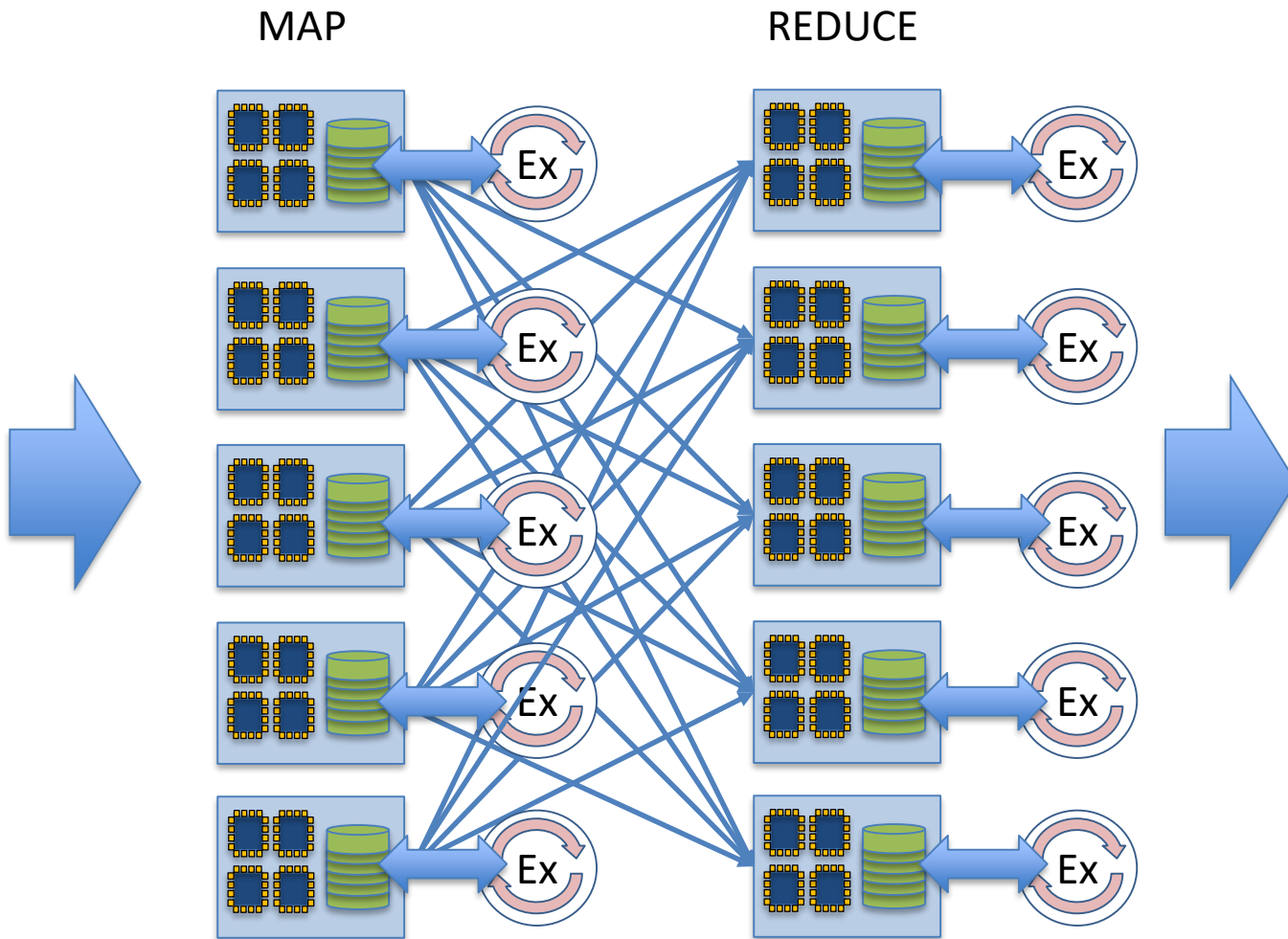- ✓ Compose multiple stages reflecting changing resource requirements

Challenges:

- o Efficient function instantiation
- o Efficient and fine grained coordination between functions and stages (pub/sub, DQ)
- o Efficient communication of ephemeral data between stages

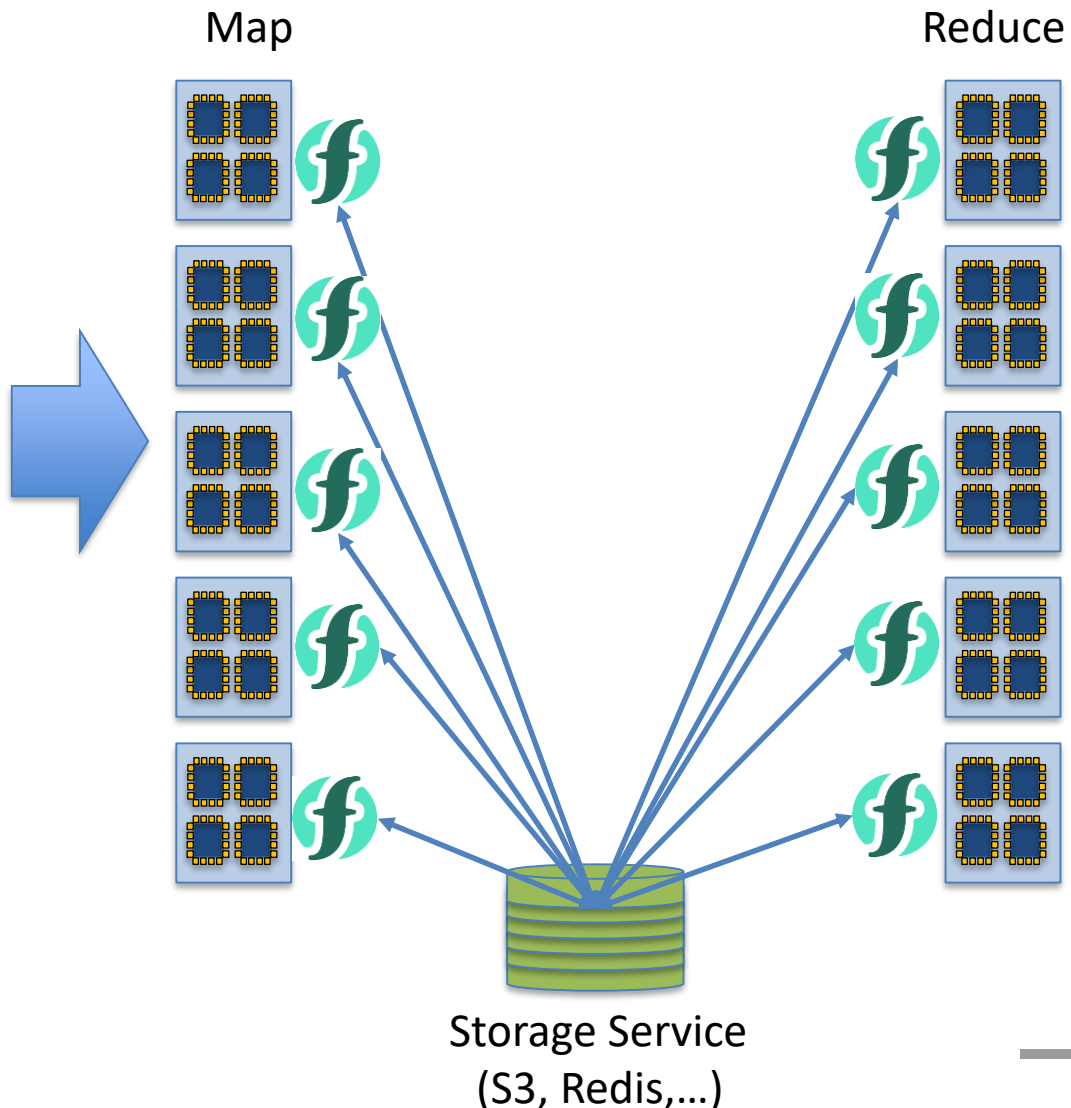# Why is the handling of those Ephemeral Data so difficult?

A MapReduce Job on a managed Cluster

MAP

REDUCE

- Ephemeral data mostly written and read **locally**
- Ephemeral data **directly** exchanged between tasks

MapReduce in a Serverless World

Map   Reduce

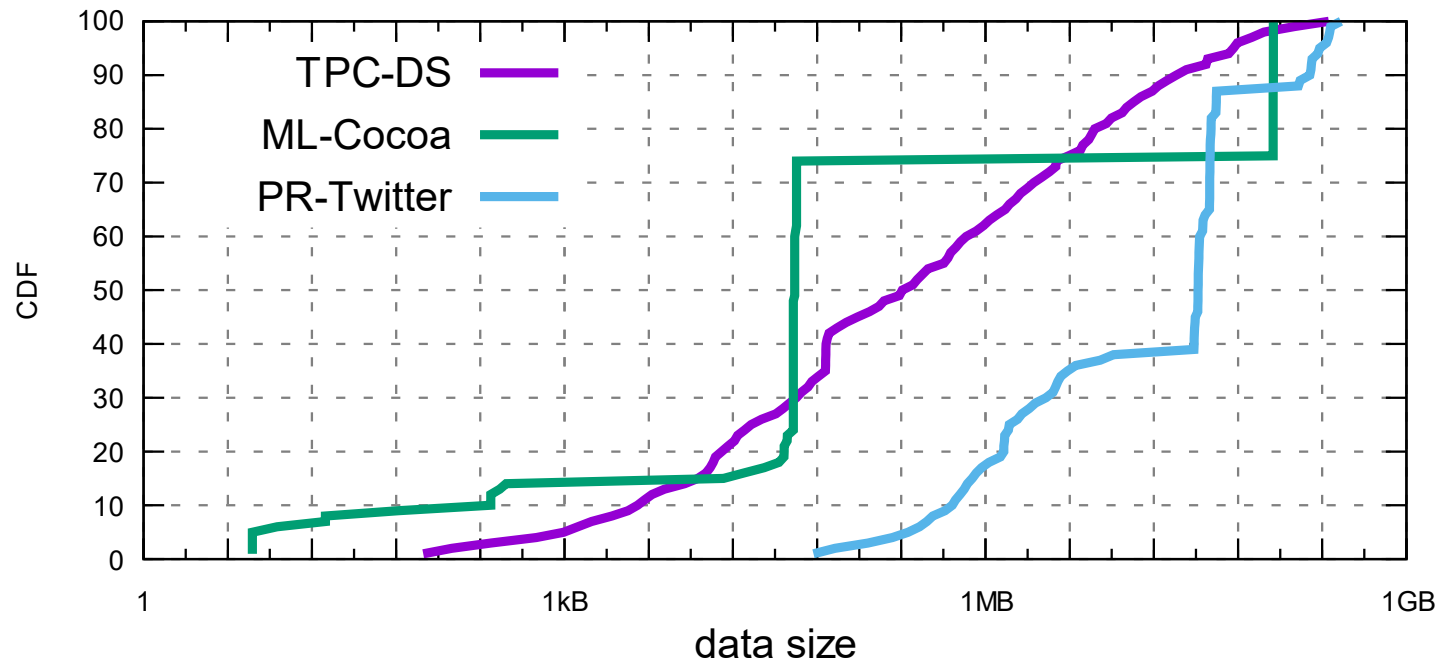

Storage Service
(S3, Redis,…)

- No direct communication between tasks
- Tasks are short living
  - Tasks are stateless
  - Read and write from distributed storage service
  - Cannot directly pass state and data from function to function

Efficiency of job execution highly dependent on storage service's
  - **Performance**, and
  - **Elasticity**

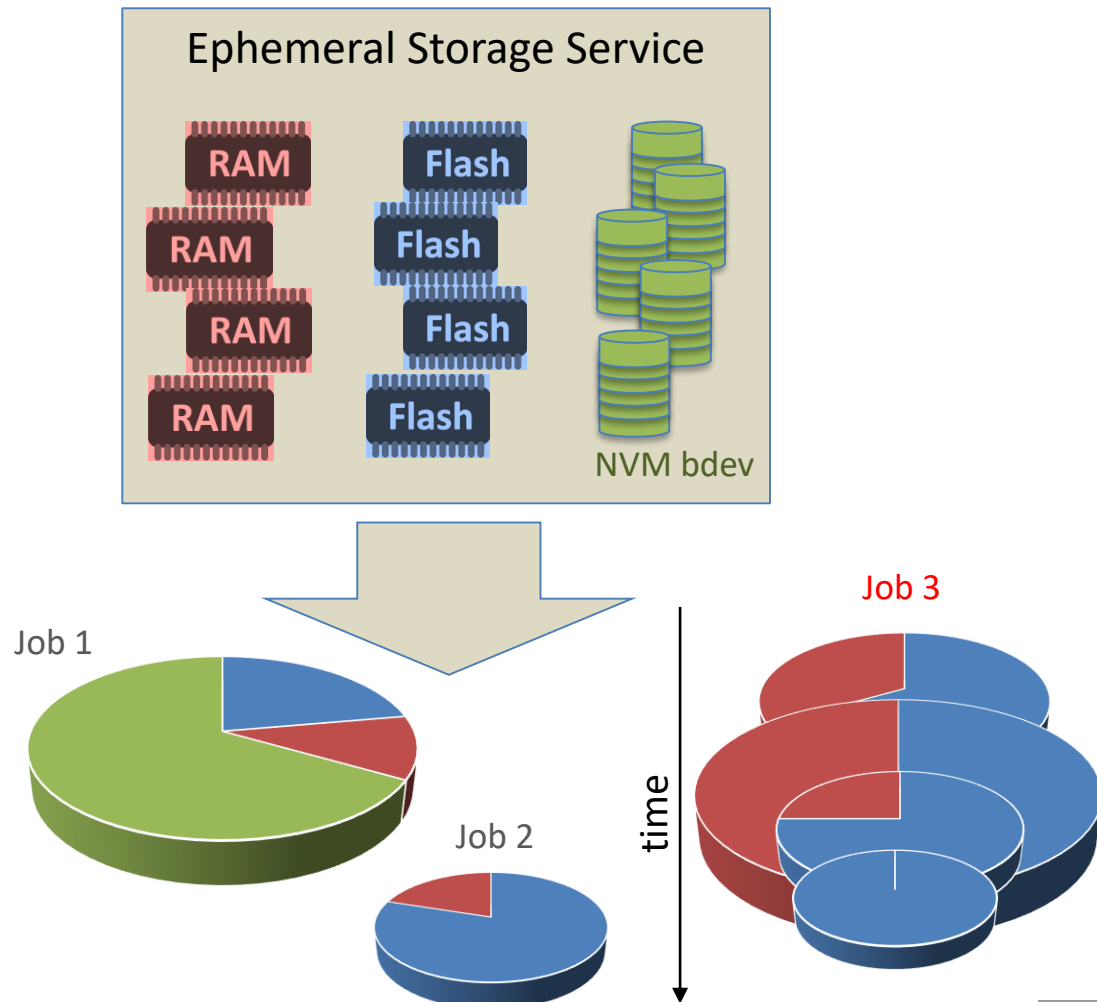# Requirements for Serverless Ephemeral Storage: Performance

CDF distribution of different types of workloads

High performance for a wide range of object sizes

- Todays KV stores have value size limits
- Object stores have access delay limits

Requirements for Serverless Ephemeral Storage: Elasticity

Ephemeral Storage Service

RAM · RAM · RAM · RAM

Flash · Flash · Flash · Flash

NVM bdev

Job 1

Job 2

Job 3

time

Fine grained billing reflecting used resources:

- Dynamically decide on storage service requirements
    1. Amount of storage required
    2. Storage performance required (including storage media type)
    3. Data persistency/lifetime (inter-function vs. intra-function)

- **Efficiently support jobs with varying storage requirements *during* job exceution**

✓ User benefit:
  - Allows for fine grained billing
  - Allows for cost optimization

✓ Provider benefit
  - Allows for optimal resource usage

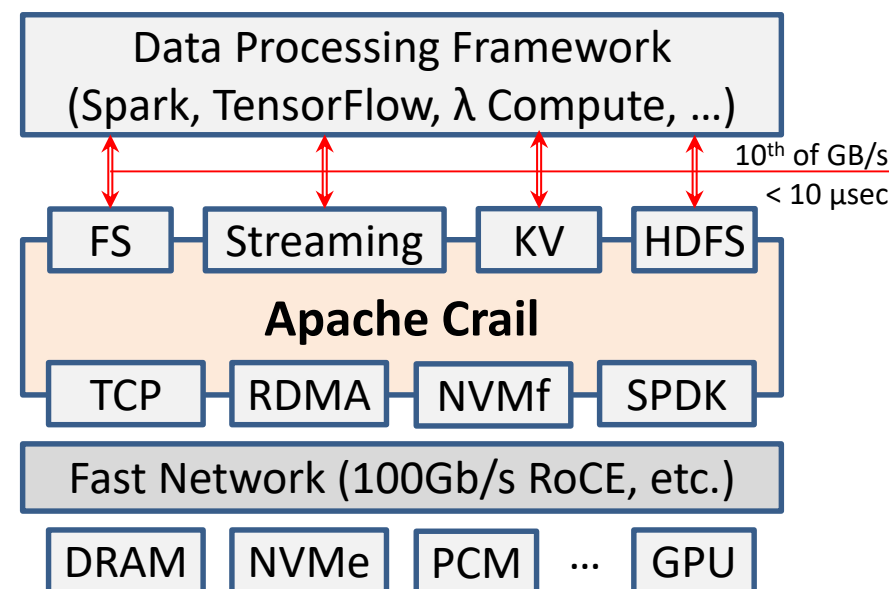# Which Data Store for Ephemeral Data?

- **Object Store, such as S3**
  - + Good for large data sets
  - − Inefficient for small data sets
- **KV Store, such as Redis, Memcached, AWS ElastiCache**
  - + Good for small data sets
  - − Inconvenient to store large data sets (indirection)
  - − No dynamic scaling
  - − DRAM only (costly)
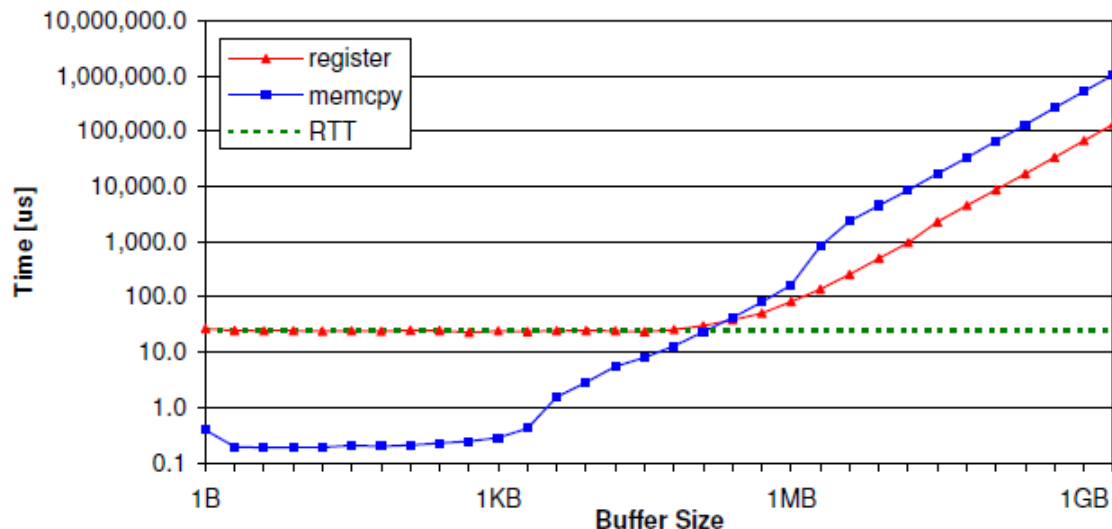- **Apache Crail (incubating) data store**
  - + Lowest delay for small data sets
  - + Highest throughput for large data sets
  - + Multi-tier (DRAM RDMA/TCP, NVMeF, block store) in unified namespace
  - + Explicit data location control possible
  - + Can use RDMA
  - − No dynamic scaling (yet)

Is RDMA a good fit for Serverless Computing?

PRO: Known RDMA Usage Benefits

- **Low delay access**
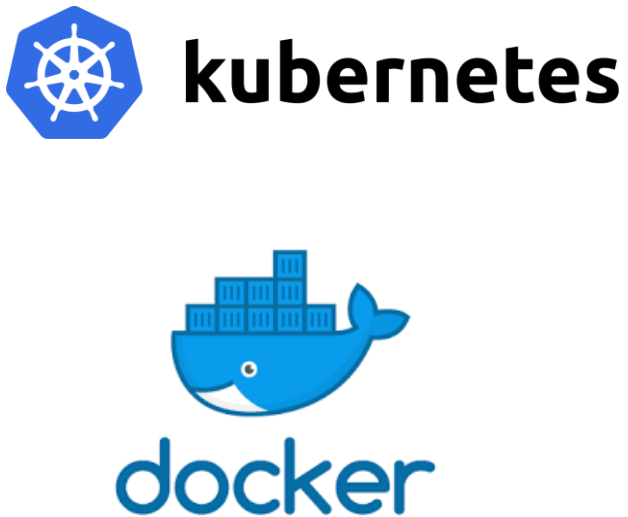- **Arbitrary sized objects**
- **Low CPU load**

CONTRA: The RDMA Setup Costs

- **Function resource setup costs**
  - RDMA objects (QP, PD, CQ, …)
  - Memory Registration
    - Local memory pinning and translation
- **Remote service setup costs**
  - RDMA objects (some RDMA objects may be reusable)
- **Both sides (function + storage service)**
  - Connection setup
    - Think about connectionless RDMA
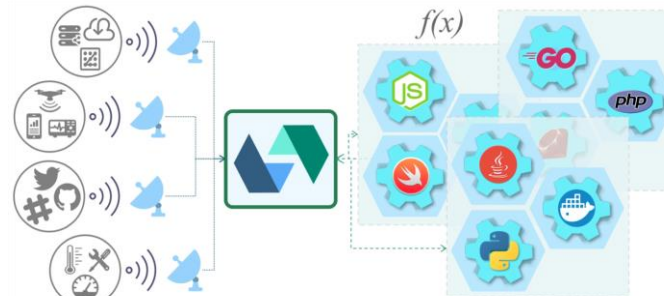 - Credential exchange with peer (for RDMA Read/Write)

The Development Environment

- Runtime environment: **Kubernetes, Docker**
- Cloud Functions: **Apache OpenWhisk**
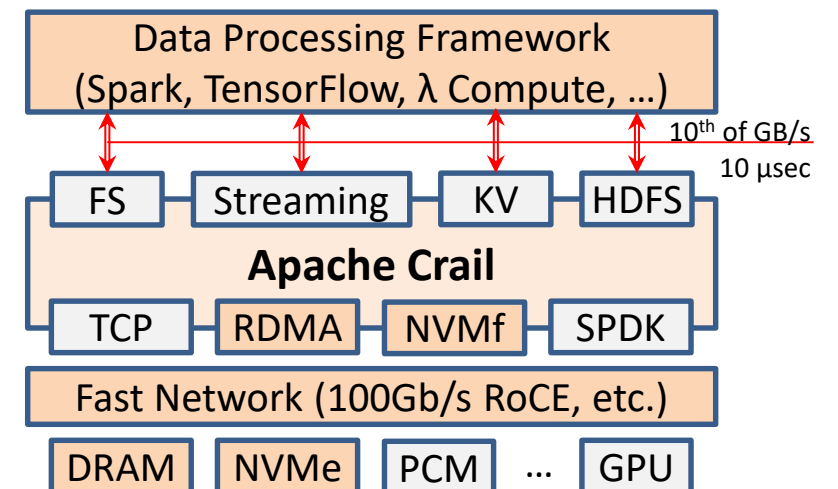- Ephemeral data store: **Apache Crail**

- Physics:
  - 4 nodes x86 cluster
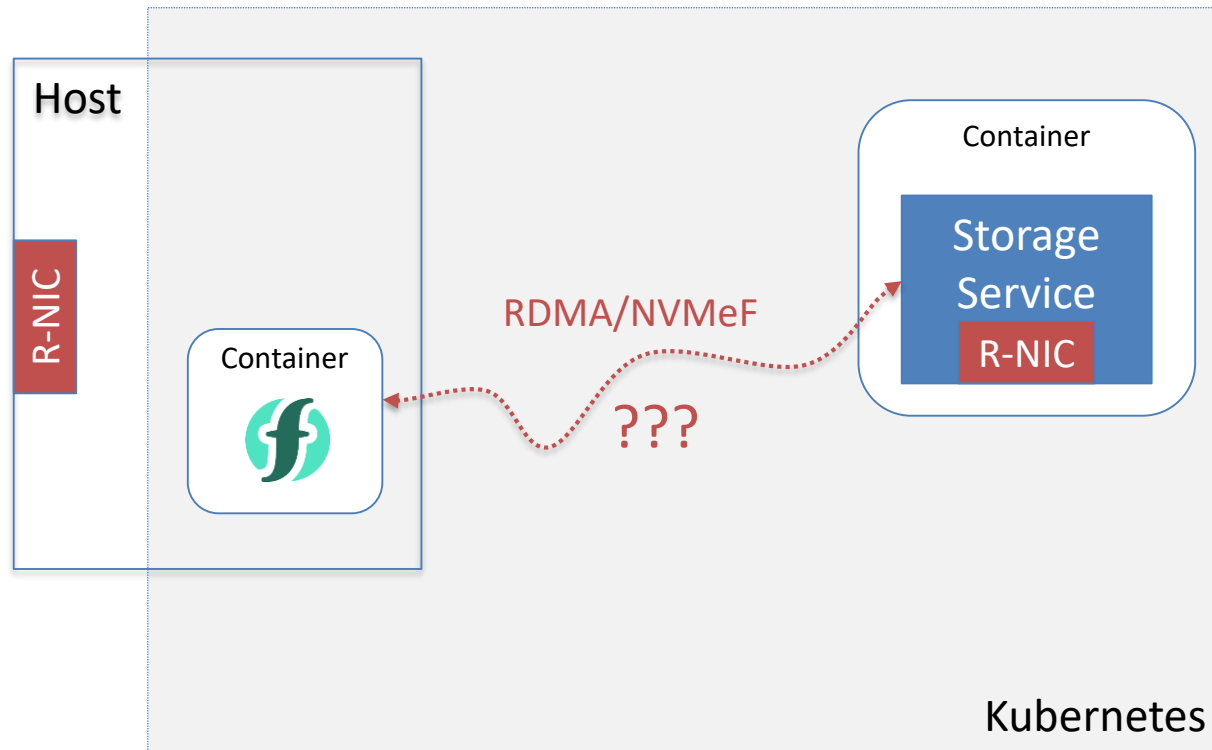 - 100Gbs RoCE
  - Samsung 960 pro NVMe m.2

`https://openwhisk.apache.org/`

`http://crail.apache.org/`

Data Processing Framework (Spark, TensorFlow, λ Compute, …)			

10th of GB/s

10 µsec

FS	Streaming	KV	HDFS

Apache Crail

TCP	RDMA	NVMf	SPDK

Fast Network (100Gb/s RoCE, etc.)

DRAM	NVMe	PCM	…	GPU

How to attach Serverless Function to an RDMA enabled Ephemeral Data Store?

Host

R-NIC

Container

RDMA/NVMeF

???

Container

Storage Service

R-NIC

Kubernetes

- **How to balance performance and elasticity?**
- **How to make integration as seamless as possible?**

Function directly talks RDMA to Data Store

Host

Container

Storage Service

R-NIC

Container

R-NIC

RDMA/NVMeF

Kubernetes

PRO

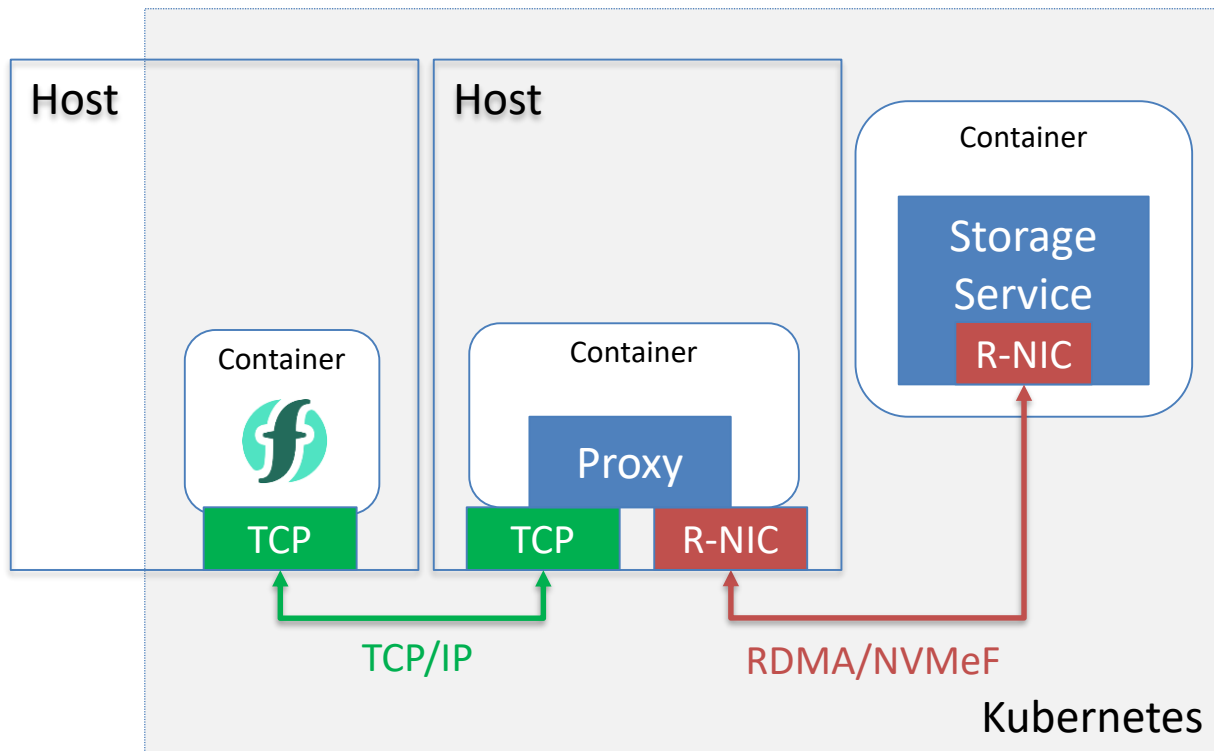- Native RDMA integration
- Fast communication

CONTRA

- Unacceptable resource setup cost on each function invocation
- Pass-through of RDMA device (Kubernetes, Docker)

POTENTIAL SOLUTION

- Warm container caches:
  - Functions
  - Connections
  - RDMA resources (memory registration etc.)
- Connectionless RDMA service

Function talks via remote Proxy to Data Store

PRO

- Serverless framework does not need to integrate with RDMA
- Elastic: no persistent state in function container
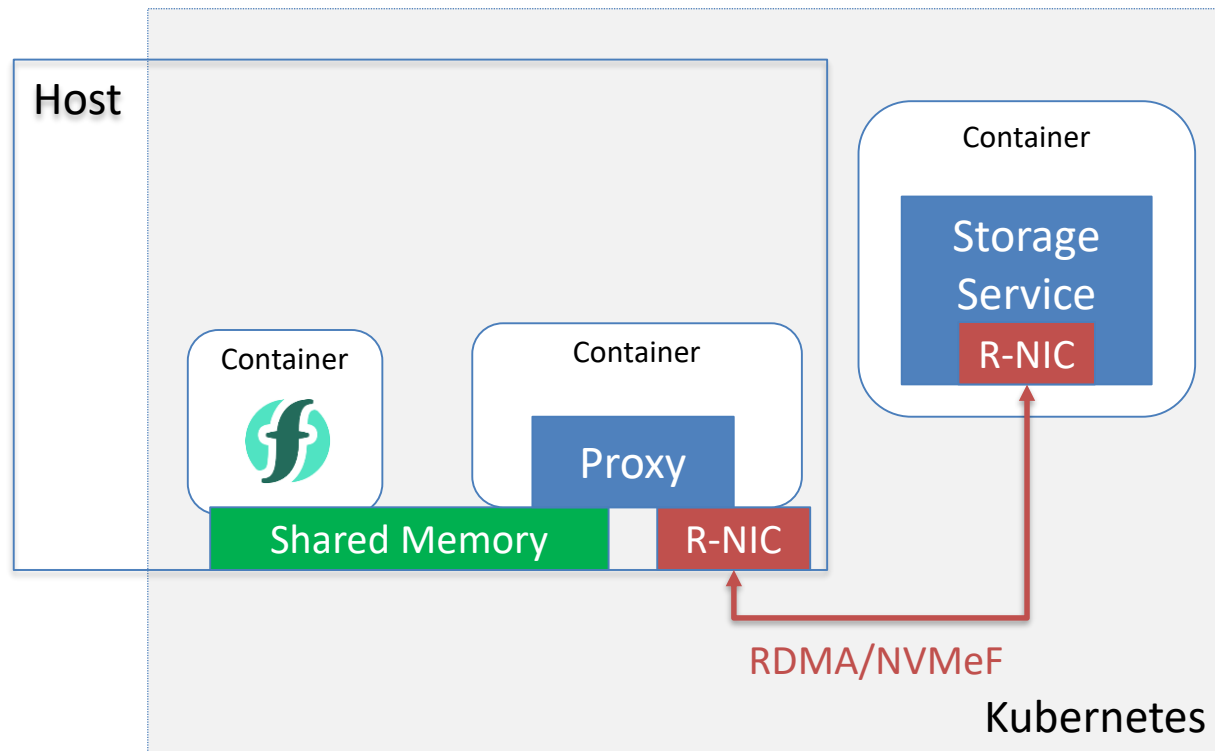- Proxy can be persistent, keeping RDMA resources and storage state

CONTRA

- Performance penalty
  - Data copy
  - Data two times on the wire

POTENTIAL SOLUTION

- Try co-locating function and proxy containers on same physical host

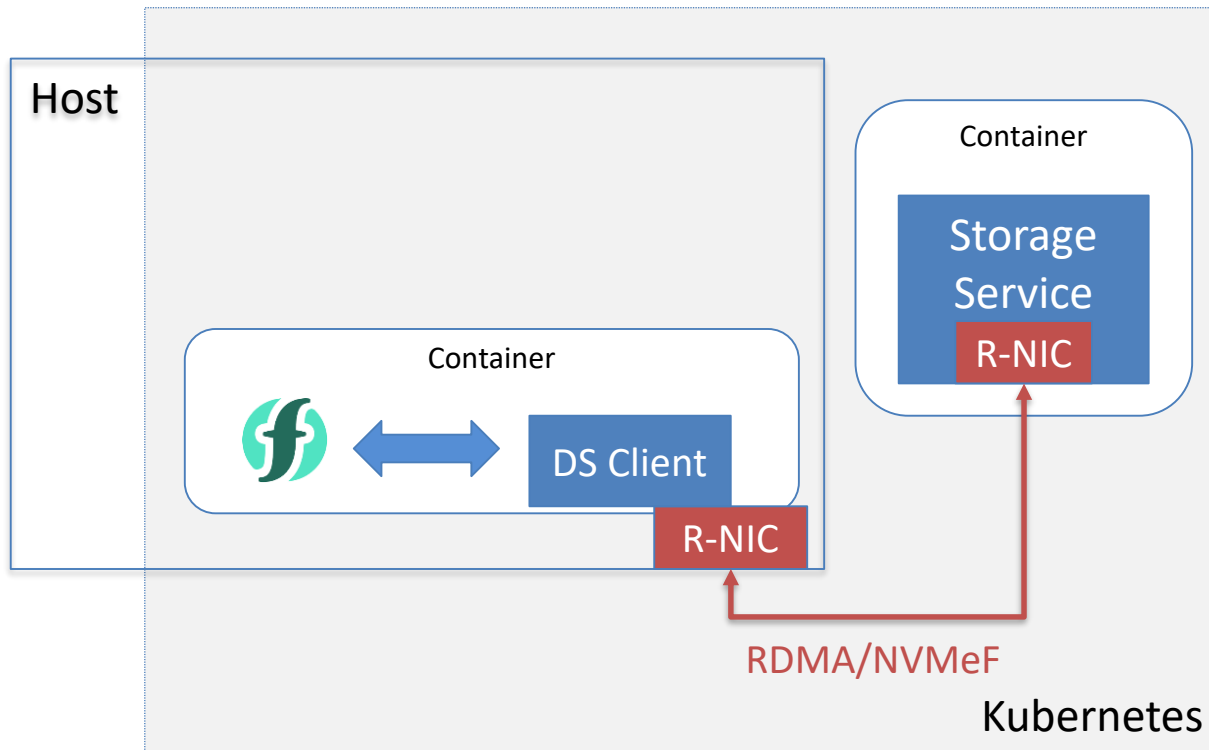# Function talks via host local Proxy to Data Store

PRO

- Serverless framework does not need to integrate with RDMA
- Elastic: no persistent state in function container
- Proxy can be persistent, keeping RDMA resources and storage state
- Shared memory: zero copy possible
  - Application handles data in shared memory
  - Shared memory is RDMA registered

CONTRA

- One storage proxy per physical host

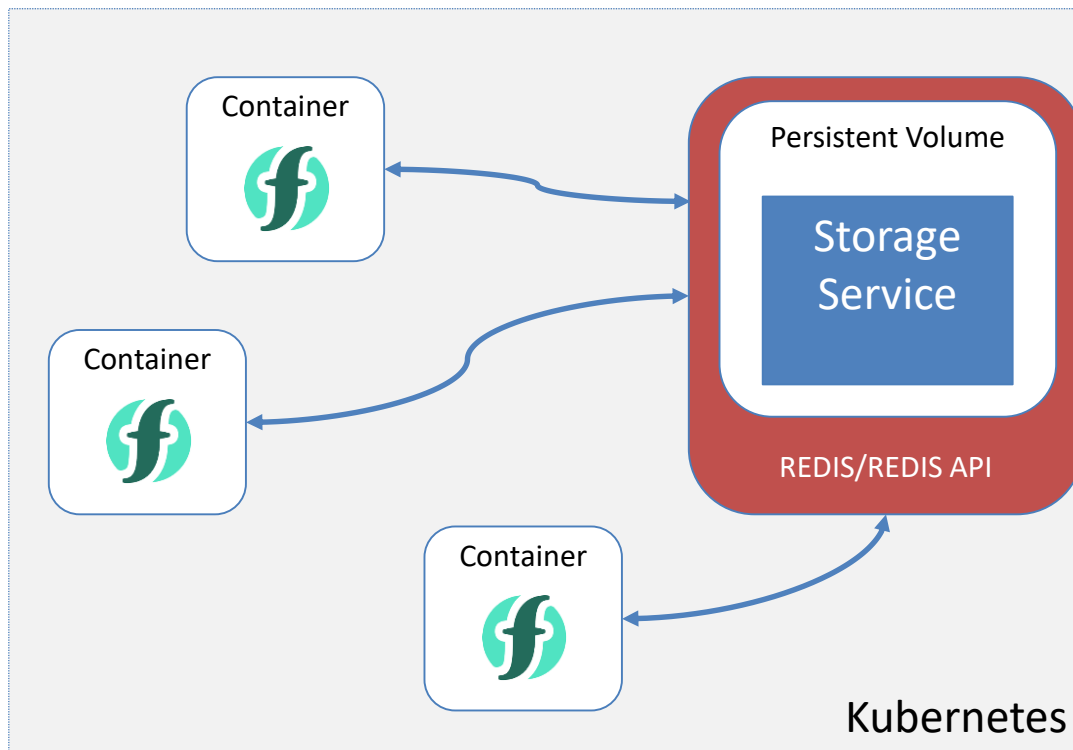# RDMA Storage Client as Service of Function's Container

PRO

- Serverless framework does not need to integrate with RDMA
- Direct communication between function and storage client
- RDMA resources and storage state can be kept persistent for 'hot' container

CONTRA

- Resource setup efficiency limited to lifetime of 'hot' container
- Pass-through of RDMA device (Kubernetes, Docker)

RDMA Storage as attachable Persistent Volume

PRO

- Clean integration like any other Volume (Redis, NFS, S3, …)
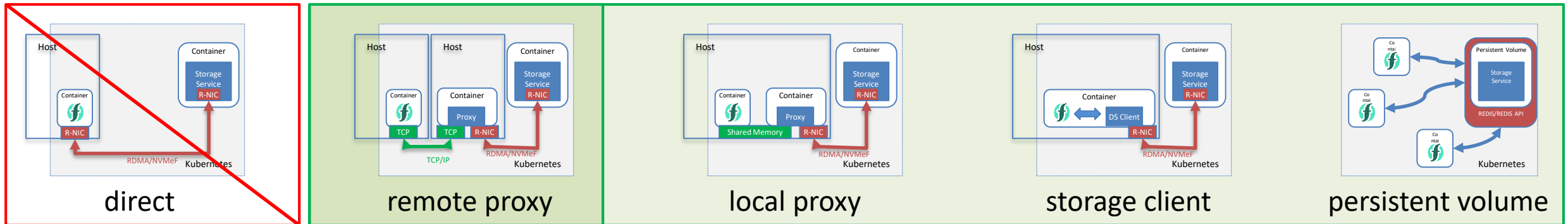- RDMA resource caching at own discretion

CONTRA

- Potentially more layers of software hiding RDMA benefits (FUSE, …)
- Serverless framework today unaware of persistent volumes
  - Containers are instantiated w/o persistent volumes
  - No Function's API to access mounted volume
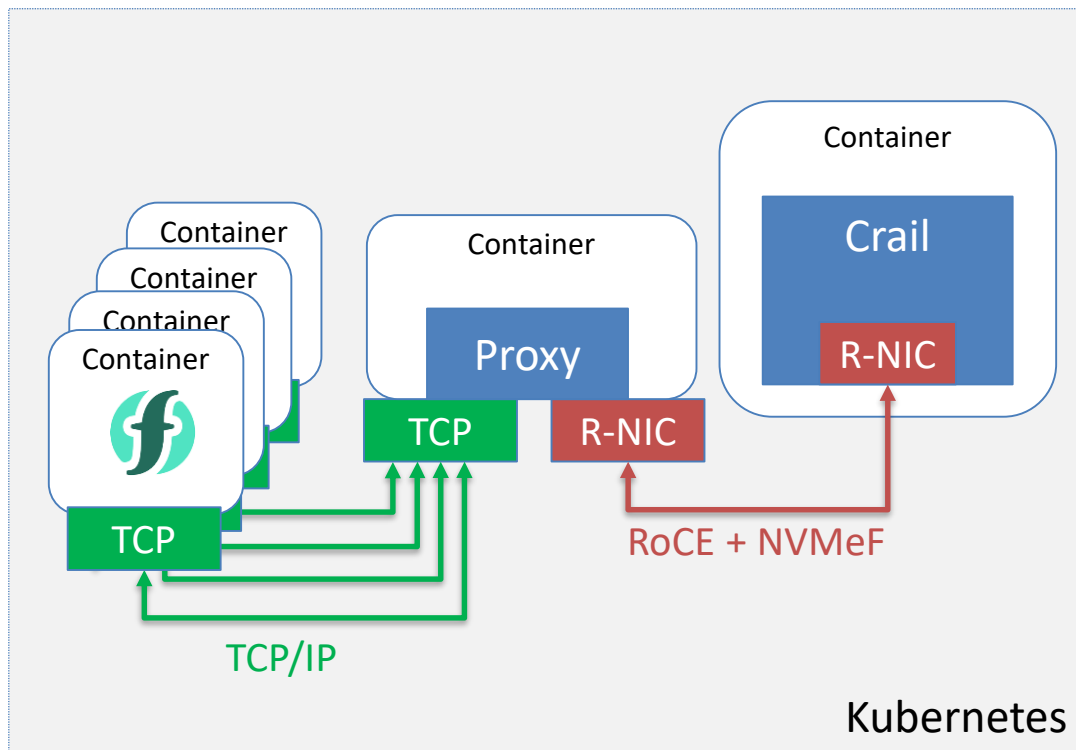
POTENTIAL SOLUTION

- Extend serverless framework to work with persistent volumes
  - Mounting
  - Storage API

Our (tentative) Choice

direct · remote proxy · local proxy · storage client · persistent volume

- **Direct Cloud Function attached RDMA interface seems unreasonable**
- **Investigating storage proxy, storage client, and persistent volume variants**
- **Persistent Volume too ambitious…for now…work in progress…**

- **Started evaluation with remote proxy**

Detailed System Setup

- **Containerized Apache Crail instance**
  - RDMA/DRAM tier
  - NVMeF tier
- **Physically distributed data store**
  - Data nodes on all machines (DRAM + NVMeF)
- **Single containerized storage proxy**
  - Runs Crail storage client on some physical node
  - Maintains RDMA connections to Crail name node and data nodes
  - Accepts connections from remote container's functions via TCP/IP
  - Converts storage requests
  - Ready to move closer to function's container

Our first test Application: MapReduce Bucket Sorting

- **MapReduce Sorting:**
  1. Read data from persistent store into map functions
  2. Write intermediate data from map functions to Ephemeral Data store
  3. Read intermediate data from Ephemeral Data store into reduce functions
  4. Merge, reduce and write back to long term storage

- **Bucket sort algorithm**
  - Map and Reduce functions written for Apache OpenWhisk
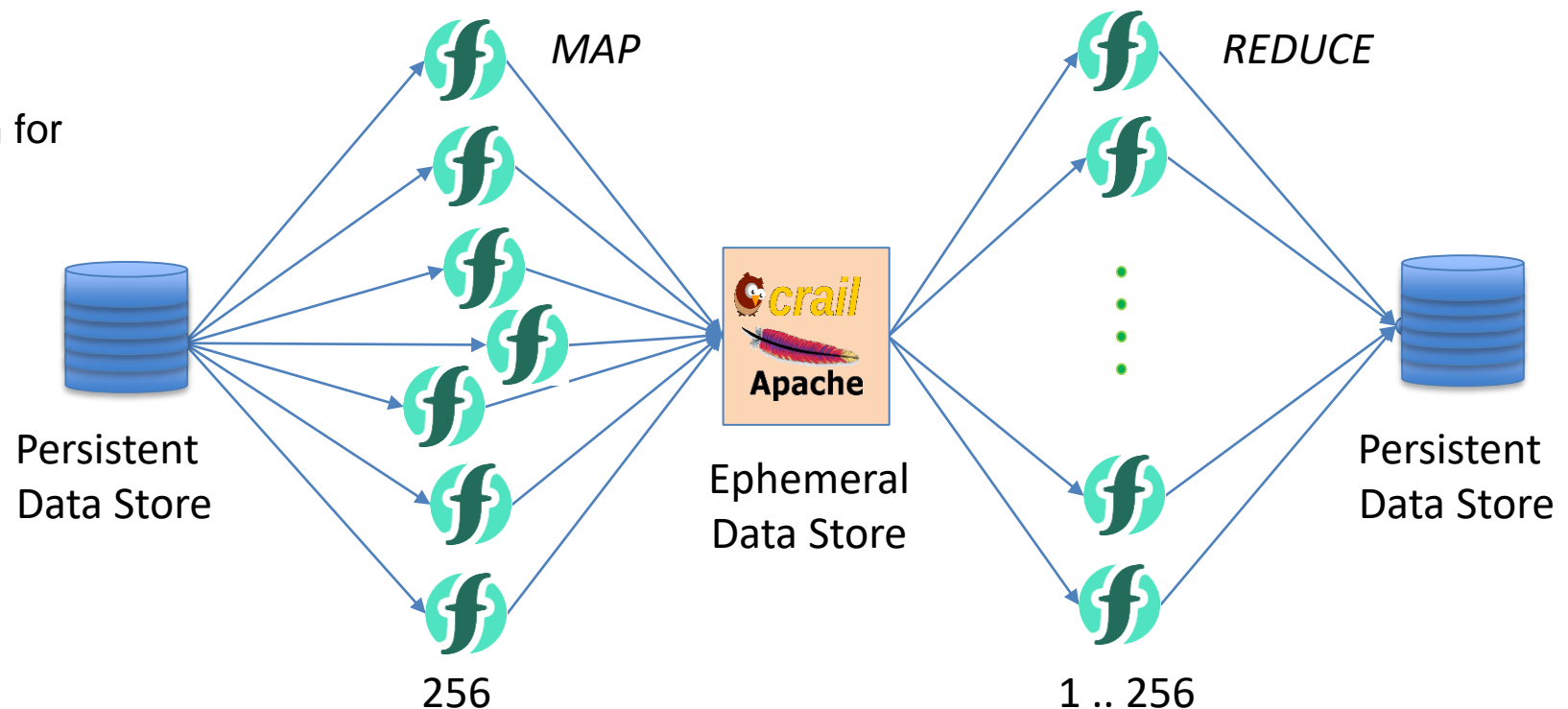
- **Sorting 100 GB**
- **Two storage tiers**
  - NVMeF
  - DRAM
- **4 Systems**
- **256 map functions**
- **1 ... 256 reduce functions**
- **Work in progress**

MAP

REDUCE

Persistent
Data Store

Ephemeral
Data Store

Persistent
Data Store

256

1 .. 256

23

Conclusions

- **Serverless Computing**
  - Promising technology, the next natural step after Cloud computing
    - Ease of programming
    - Automated resource management
    - Pay-as-you-go
  - Many unsolved issues (security, ephemeral and durable storage, networking, management, …)
  - Solving those issues will make Serverless Computing attractive for almost all applications
- **Our current effort:**
  - Focus on the Ephemeral Storage aspect
  - Integrate Apache Crail backed Ephemeral Data Store
    - Promising high performance, cost efficiency
    - May use RDMA if available
- **Other related activities:**
  - Check 'Pocket' project with architectural similarities for very promising results using Crail, even w/o RDMA
    `https://www.usenix.org/conference/osdi18/presentation/klimovic`
  - Flexible provisioning/autoscaling with Apache Crail
  - Add distributed communication primitives to the store

15th ANNUAL WORKSHOP 2019

THANK YOU