CloudButton

WP2. Architecture

Pedro Garcia Lopez
Coordinator @ Universitat Rovira i Virgili

BIG DATA VALUE
PUBLIC-PRIVATE PARTNERSHIP

UNIVERSITAT ROVIRA i VIRGILI

redhat

IBM

EMBL

Institut Mines-Télécom

Imperial College London

The James Hutton Institute

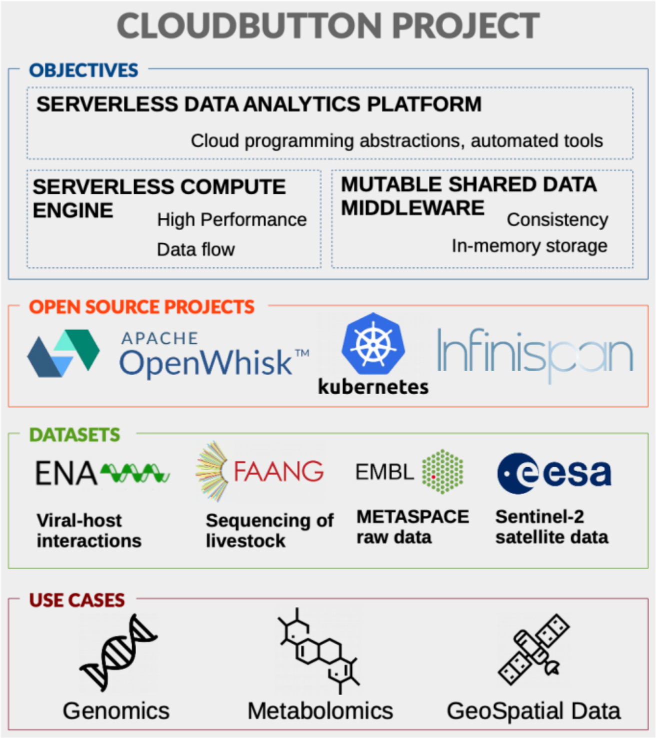
Atos

fundaciónMATRIX
INVESTIGACIÓN Y DESARROLLO SOSTENIBLE

http://cloudbutton.eu

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

CloudButton Goals and tasks



WP2
URV

UNIVERSITAT ROVIRA i VIRGILI

- **Global Architecture** – URV
- **Prototyping and Software Evaluation** – URV
- **Testbed and Validation** – ATOS
- **Use cases** – EMBL, ANSWARE, JHI, MATRIX

EMBL · The James Hutton Institute

WP5
Imperial

Imperial College London

- **Progr. Abstr. Stateful Serverless Comp.** – Imperial
- **Consistency & fault tolerance models** – Imperial
- **Progr. tools porting existing applications** – Imperial
- **Application patterns and libraries** – Imperial

WP3
IBM

IBM · AtoS

- **High Perf. Stateful Compute Engine** – IBM
- **CLOUDBUTTON Operations Support** – IBM
- **Instrumentation and QoS** – ATOS
- **Big Data Serverless Execution Framework** – IBM

WP4
IMT

Institut Mines-Télécom · Red Hat

- **Progr. abstractions Mutable Shared Data** – IMT
- **Sup. for Consistency Degradation** – IMT
- **Just-right Synchronization** – RHAT
- **In-memory Data Storage** – RHAT

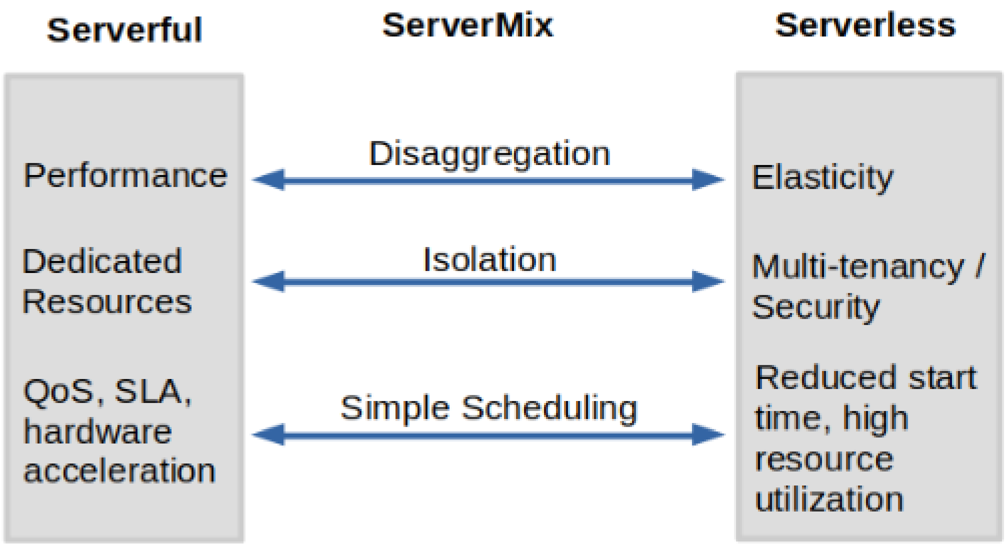
Global Architecture

CloudButton Requirements and KPIs

- **Simplicity/Productivity**
  - Front-end engineers and data analysts without Cloud knowledge
  - Hide Resource provisioning (data-driven approach)
  - Semi-transparent (Python notebooks) or fully transparent transition to the Cloud (CloudButton)
- **Performance**
  - Show performance improvements compared to cluster technologies like Spark
  - Low overheads in the transition to the Cloud
- **Scalability**
  - Proof that we scale to large data volumes using massive parallel computing power
  - Big Data pipelines (Use Cases)
- **Elasticity**
  - Demonstrate elastic workloads that benefit from the Serverless model
- **Cost**
  - Provide adequate cost/performance tradeoffs and offer alternatives services for batch analytics

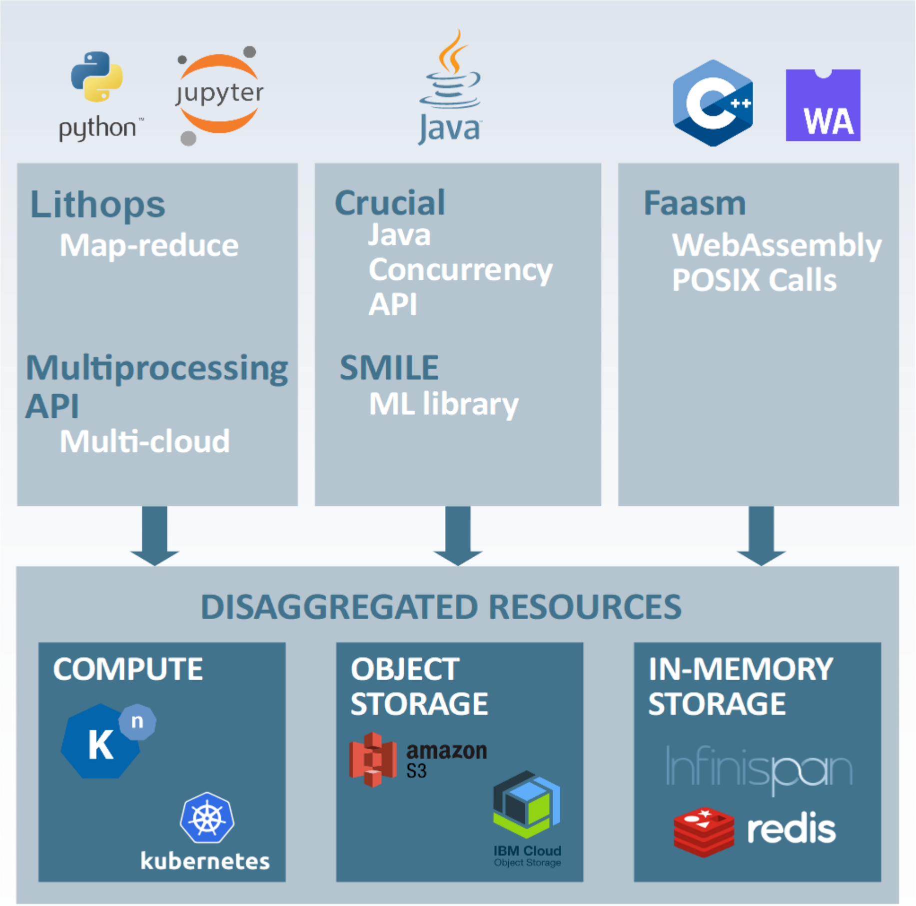
Serverless Challenges

- Complexity
- Stateful Computing
- Direct Communication
- Intermediate data
- Cost !
- Smart Data-driven provisioning



Pedro García López, Marc Sánchez Artigas, Simon Shillaker, Peter R. Pietzuch, David Breitgand, Gil Vernik, Pierre Sutra, Tristan Tarrant, Ana Juan Ferrer, Gerard París:
Trade-Offs and Challenges of Serverless Data Analytics. Technologies and Applications for Big Data Value 2022: 41-61

Transparency



We advocate for **access transparency**: **enabling local and remote resources to be accessed using identical operations.** Transparency means concealing the complexities of distributed programming like remote locations, failures or scaling.

For us, **full transparency** implies that we can **run unmodified single-machine code** over effectively **unlimited** compute, storage, and memory **resources**.

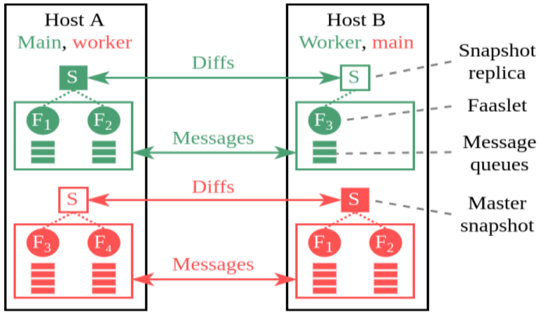
- Serverless End Game: Dissaggregation Enabling Transparency
- Toward Multicloud Access Transparency in Serverless Computing
- Efficient portage of Java and shell applications to serverless.
- Transparently running OpenMP/MPI applications on Serverless
- Transparent Serverless execution of Python multiprocessing applications

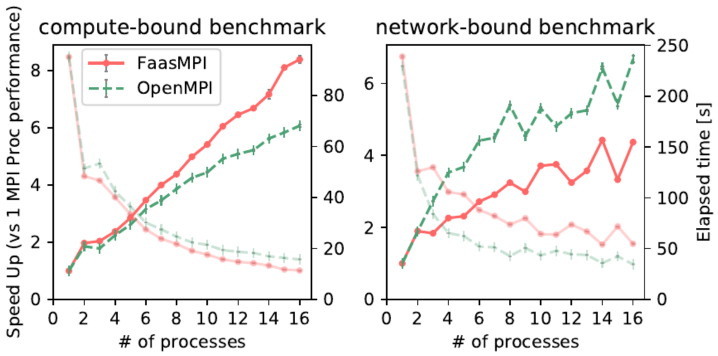
Transparent Support for HPC Applications (WP5)

- Supports **shared memory** & **message passing** abstractions for serverless
- Implements **OpenMP** & **MPI** interfaces for data & compute intensive HPC applications
- Provides **scheduling**, **state management**, and **snapshots** using Faasm

Faabric library for distributed shared memory & message passing on serverless

- Two key abstractions: **Snapshots** & **FGroups**
- For **shared memory**, Faaslets are forked and their state is synchronised using diffs
- For **message passing**, Faaslets have group address for asynchronous communication

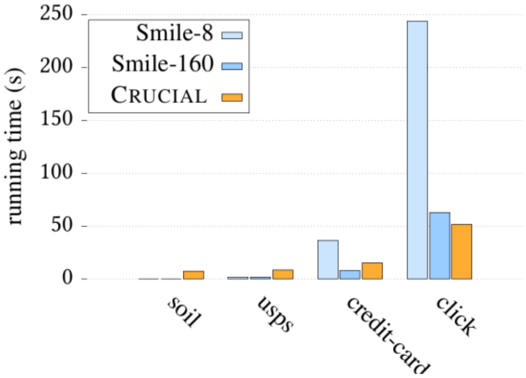




- Available on **GitHub**: https://github.com/faasm/faabric
- Better **performance** compared to commercial batch cloud solutions
- Published in USENIX ATC 2020, more papers submitted

Java transparency (WP4)

```java
service = new AWSLambdaExecutorService();
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();
});
```



SMILE Machine Learning Library

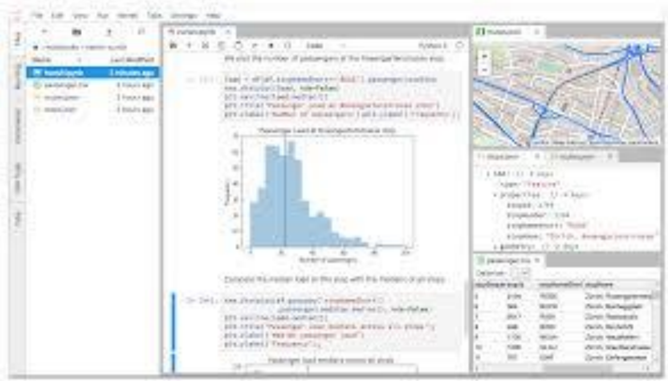
Main features
- multithreaded-like API
- easy portage to serverless of legacy applications
- support for multiple FaaS platforms
- serverless-ready storage *(elastic & dependable, NVM support, kubernetes operator)*

Infinispan

Scientific contributions published in

[TOSEM'22, SOSP'21, Eurosys'21/20, Middleware'21/19]

Python Transparency (WP2,WP3)



NumPy · pandas · scikit-learn · OpenAI · DASK

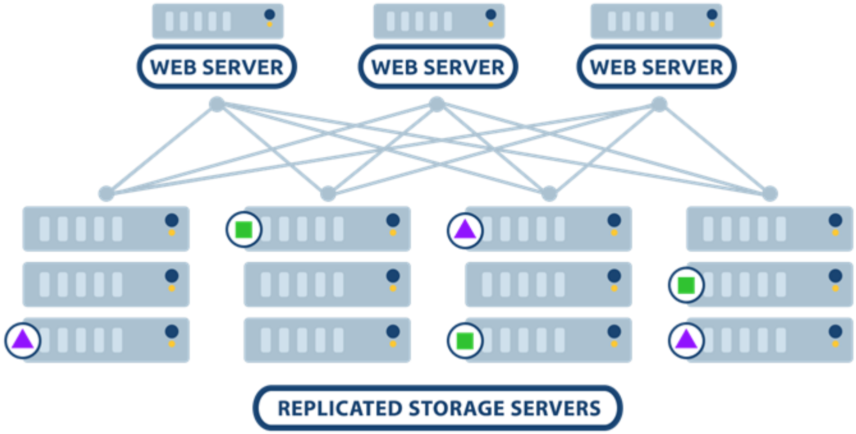
LITHOPS

IBM Cloud	aws	Microsoft Azure	Google Cloud	Alibaba Cloud	Red Hat	kubernetes
Cloud Functions Code Engine VPC Cloud Object Storage	AWS Lambda AWS S3	Functions Blob Storage	Cloud Functions Cloud Run Cloud Storage	Functions Compute Object Storage Service	OpenShift	Batch/Job Knative OpenWhisk OpenStack Swift Redis Ceph

What is Lithops

Lithops is a

- Map-Reduce Serverless Platform
- Orchestrator of Cloud Resources
- Data Staging platform
- Python Cloud Computing toolkit
- Especially good for large **unstructured** data



Examples
Data Preprocessing
Extract, Transform and Load (ETL)
Parallelization of Python code in the cloud
Python notebooks in the Cloud
Hyper-parameter tuning
Montecarlo Simulation
Image processing
Text filtering and queries
Data reduction

Python Multiprocessing transparency

```python
import lithops
import random

def is_inside(n):
    count = 0
    for i in range(n):
        x = random.random()
        y = random.random()
        if x*x + y*y < 1:
            count += 1
    return count

if __name__ == '__main__':
    np, n = 10, 15000000
    part_count = [int(n/np)] * np
    fexec = lithops.FunctionExecutor()
    fexec.map(is_inside, part_count)
    results = fexec.get_result()
    pi = sum(results)/n*4
    print("Esitmated Pi: {}".format(pi))
```

Usage
- Application-level transparency
- Scale applications using serverless

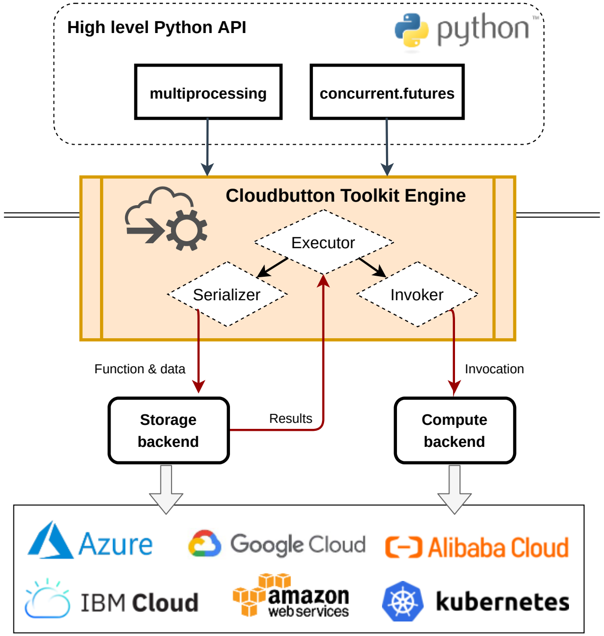
Full multiprocessing API implemented
- Process, Pool, Pipe, Queue, Manager …

Achievements
- Migrate legacy applications to the cloud
- Vertical scaling of a VM
- Hiding complexities of distributed systems

LITHOPS

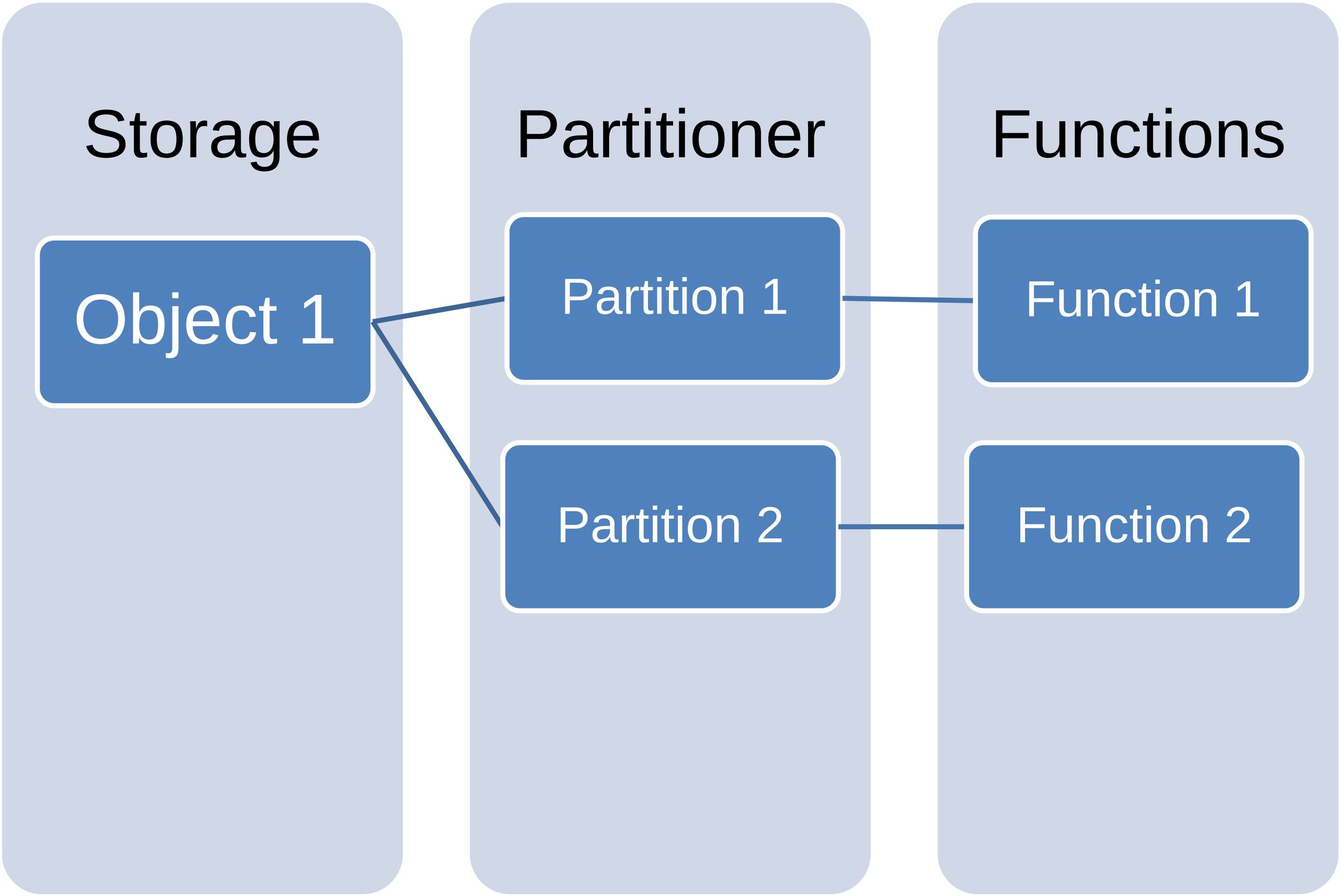
Lithops MapReduce API (Data-driven)

```python
import lithops

# Bucket with prefix
data_location = 'cos://lithops-sample-data/test/'  # Change-me


def my_map_function(obj):
    print('Bucket: {}'.format(obj.bucket))
    print('Key: {}'.format(obj.key))
    print('Partition num: {}'.format(obj.part))
    counter = {}

    data = obj.data_stream.read()

    for line in data.splitlines():
        for word in line.decode('utf-8').split():
            if word not in counter:
                counter[word] = 1
            else:
                counter[word] += 1

    return counter


if __name__ == "__main__":
    fexec = lithops.FunctionExecutor(log_level='DEBUG')
    fexec.map(my_map_function, data_location)
    print(fexec.get_result())
```



IBM Cloud	aws	Microsoft Azure	Google Cloud	Alibaba Cloud	kubernetes OPENSHIFT
Cloud Functions Code Engine VPC Gen2 --- Cloud Object Storage	AWS Lambda AWS Batch AWS EC2 --- AWS S3	Functions --- Blob Storage	Cloud Functions Cloud Run --- Cloud Storage	Functions Compute --- Object Storage Service	Batch/Job - OpenWhisk Knative --- OpenStack Swift - Ceph MinIO - Redis - Infinispan

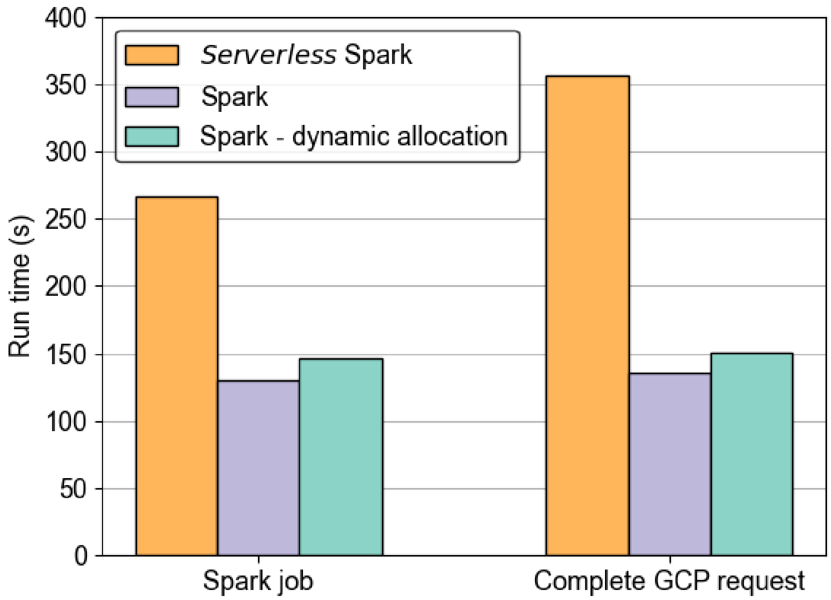
Lithops Partitioner (Data-driven)

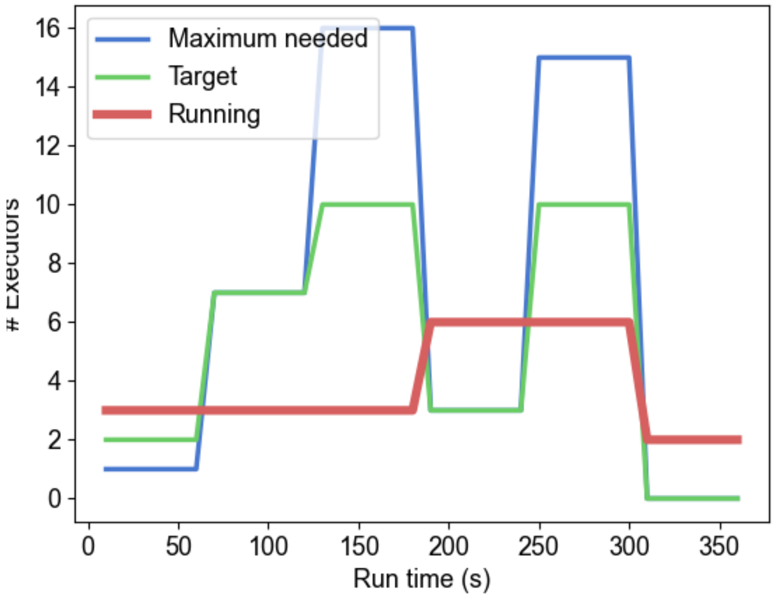


- Preprocessing
  - LIDAR
- On the fly
  - mlMZ
  - Text
  - GZIP
  - COPC
  - COG*

Benchmarks and Validation

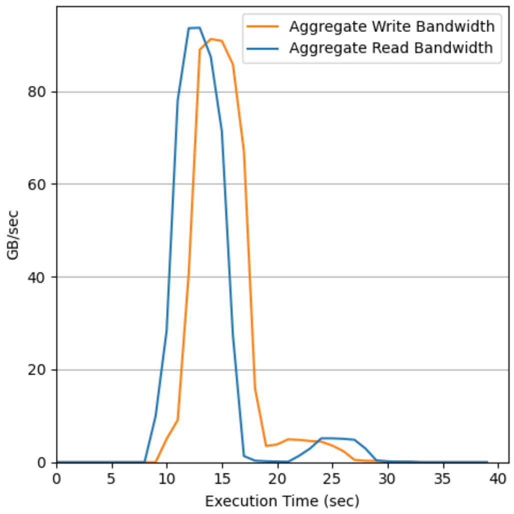
Elasticity (Spark vs Cluster Spark)

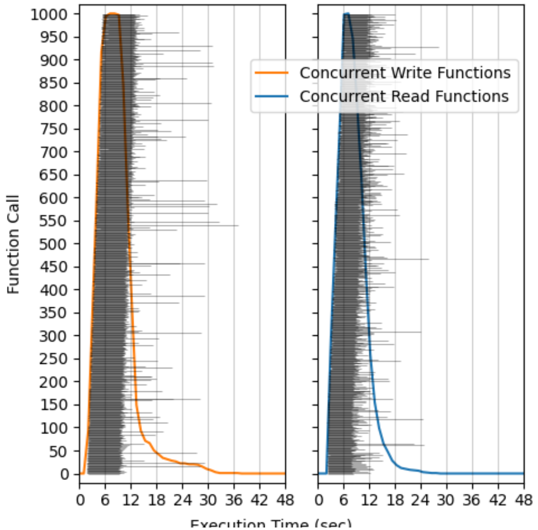




Lithops benchmarks

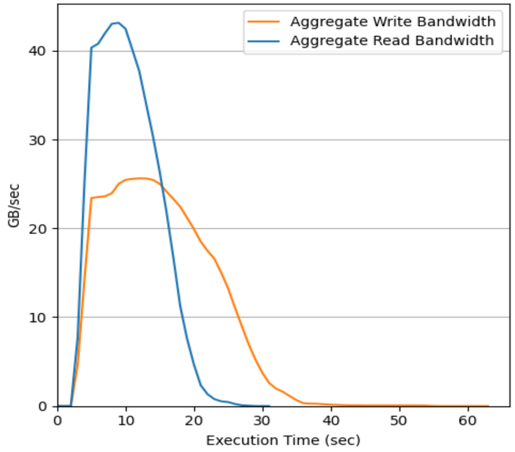
Lithops Multicloud FaaS Benchmark.. [*https://github.com/lithops-cloud/applications/tree/master/benchmarks*](https://github.com/lithops-cloud/applications/tree/master/benchmarks)

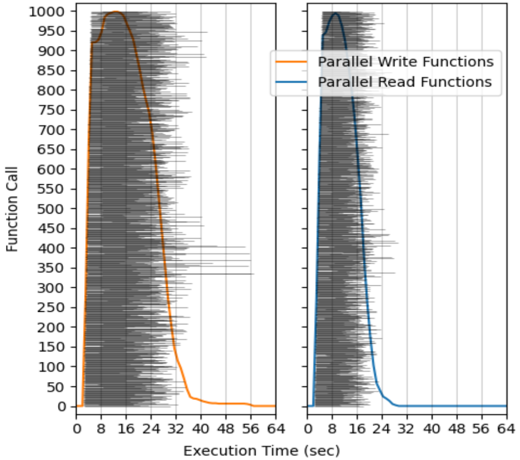




Amazon Lambda

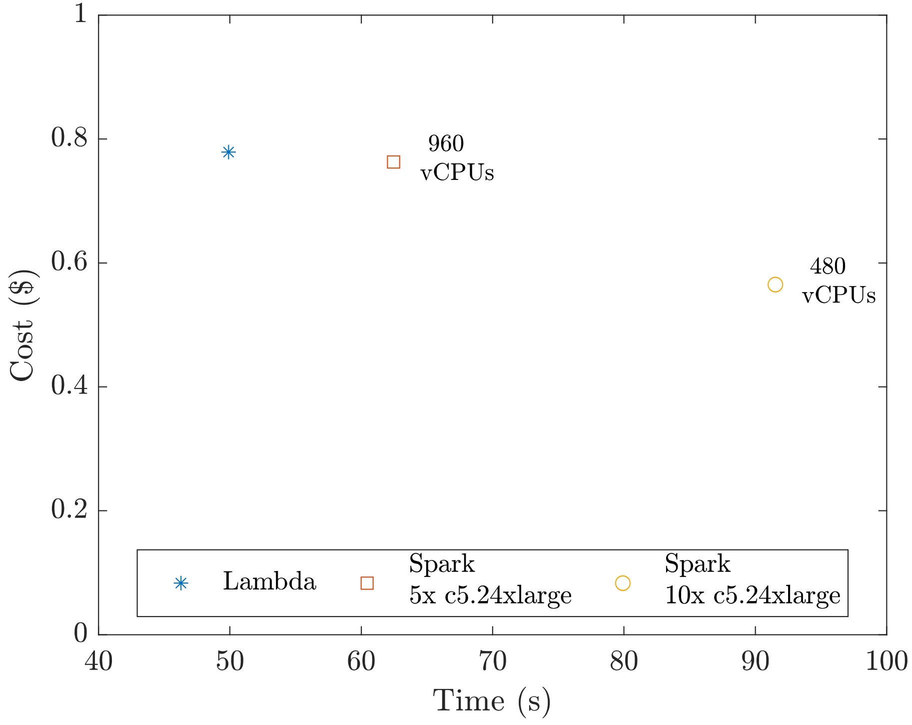
Amazon S3

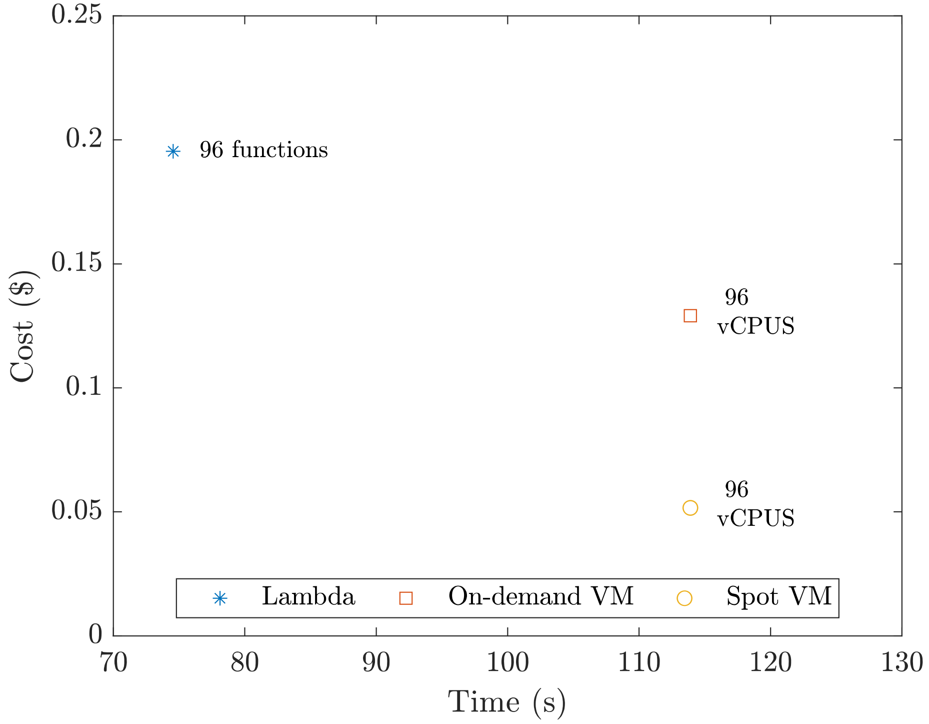




IBM Cloud Object Storage

Elasticity compute-intensive (Spark vs Serverless)





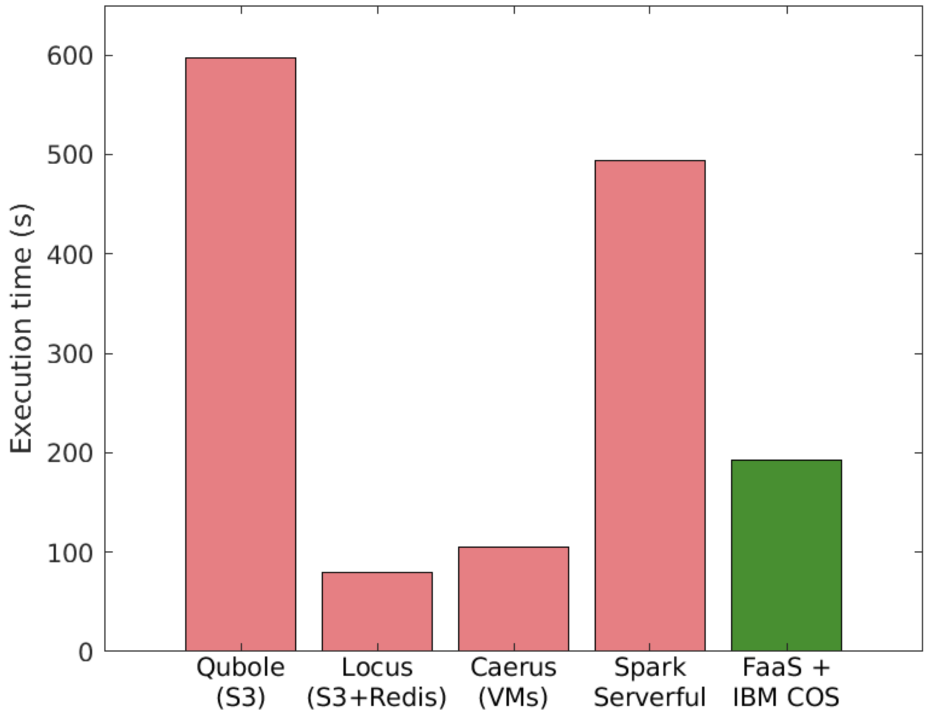
Reference: *Serverless Elastic Exploration of Unbalanced Algorithms*

Performance data–intensive (100GB Terasort)

A **completely serverless architecture** (cloud functions + object storage) in the IBM Cloud

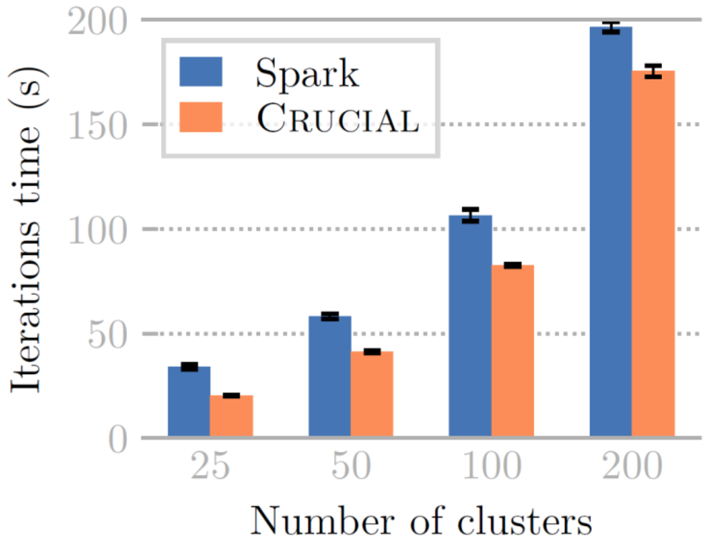
vs

State-of-the-art **serverful and partially serverful** solutions

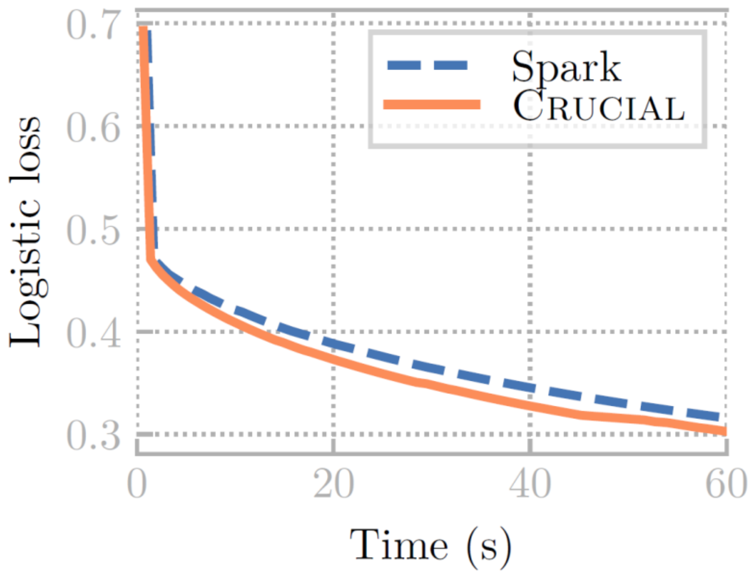


Reference: Primula: a Practical Shuffle/Sort Operator for Serverless Computing

Performance comm-intensive (Machine learning)



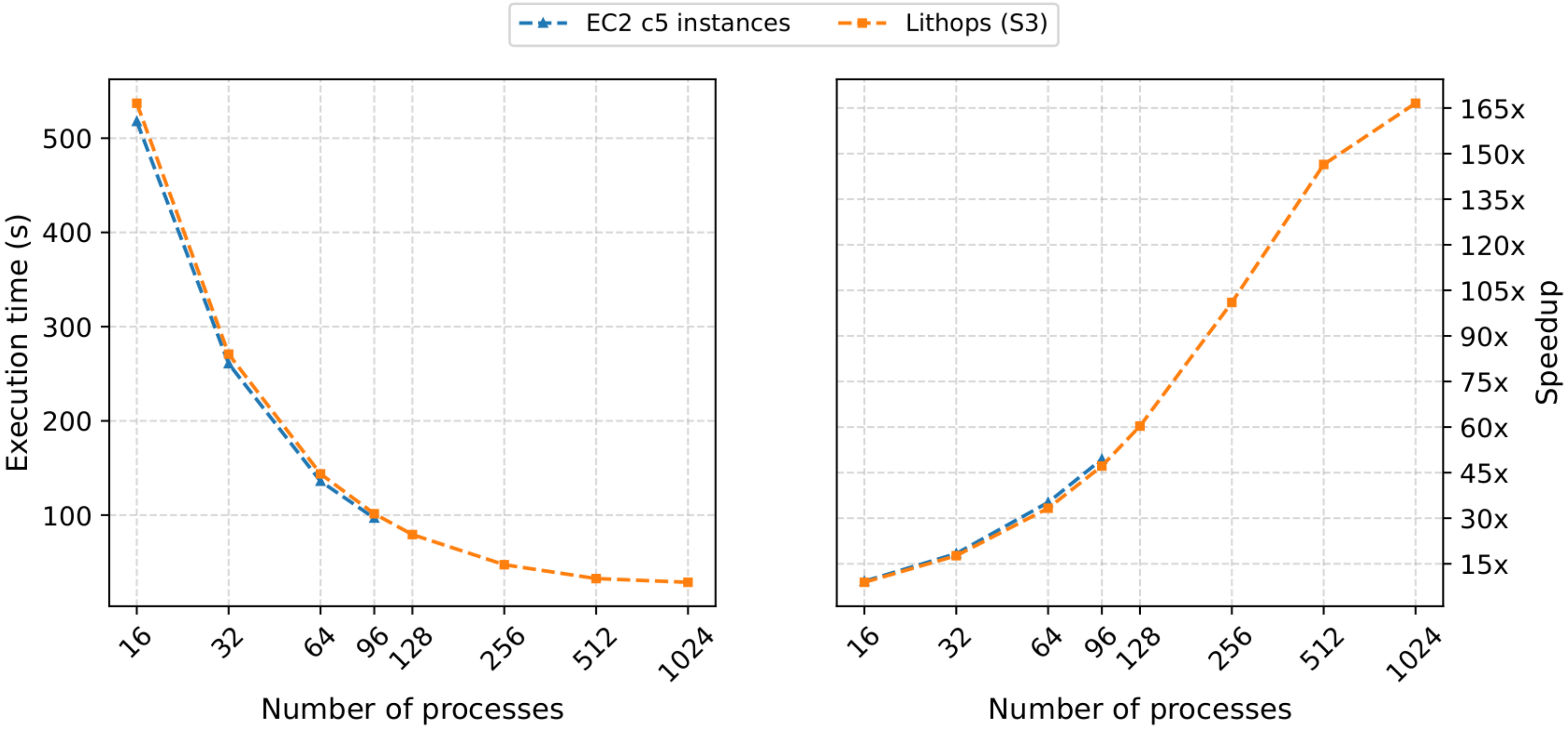
KMeans



Logistic Regression

Reference: *Stateful Serverless Computing with Crucial*

Performance (Hyperparameter tuning)



Reference: Transparent Serverless execution of Python multiprocessing applications.

Performance (Machine learning)

- For fixed budgets, MLLess (Lithops) is capable of outperforming
  Pytorch’s serverful architecture in all cases
- FaaS implementations can be more cost-efficient than
  serverful solutions for fast-converging algorithms when
  the right optimisations are applied.

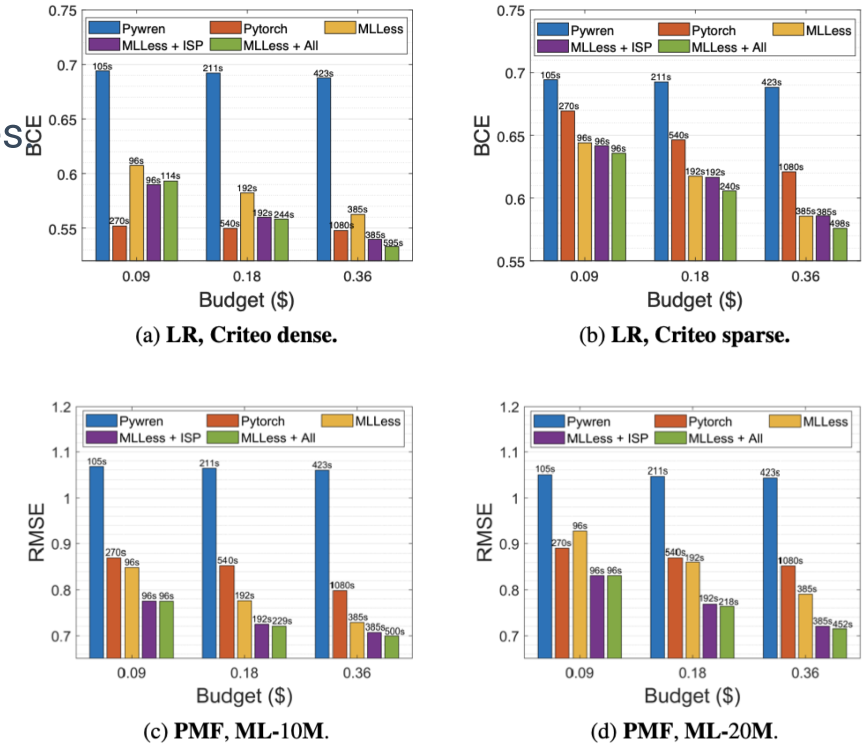
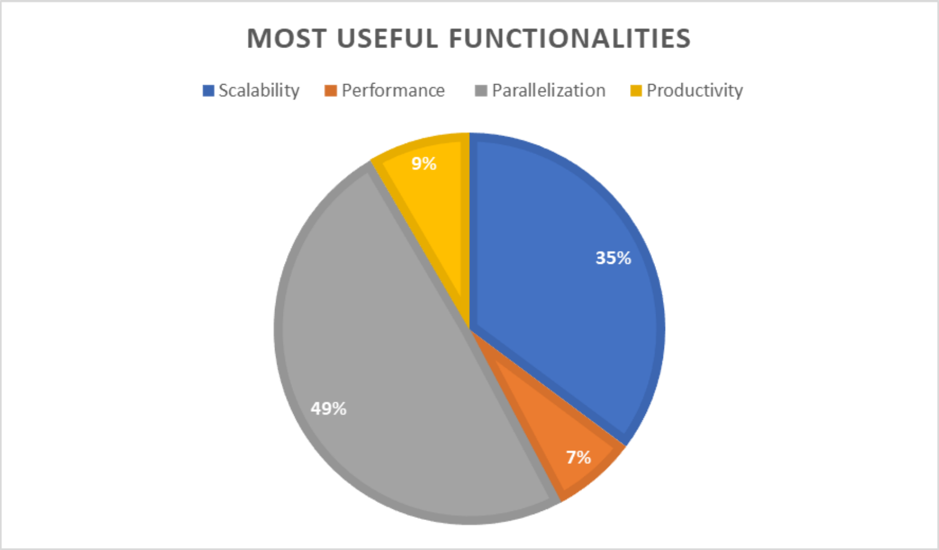


Figure 8: Cost vs. loss comparison between PyTorch, PyWren-IBM and MLLess with different variants: BSP synchronization (**MLLess**), ISP synchronization (**MLLess + ISP**) and ISP synchronization + auto-tuner (**MLLess + All**), for 24 workers. The numbers above the bars report the maximum execution time affordable with each possible budget.

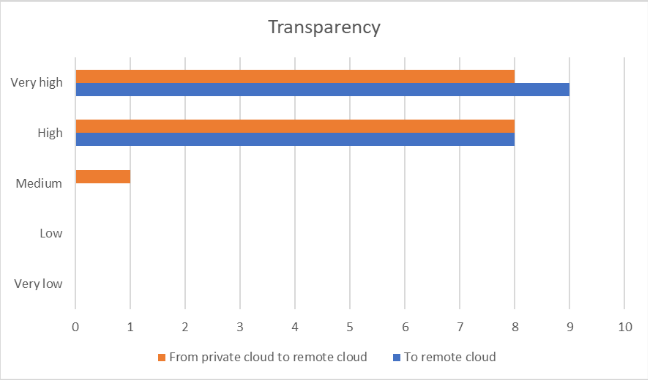
Reference: MLLess: Achieving Cost Efficiency in Serverless Machine Learning Training

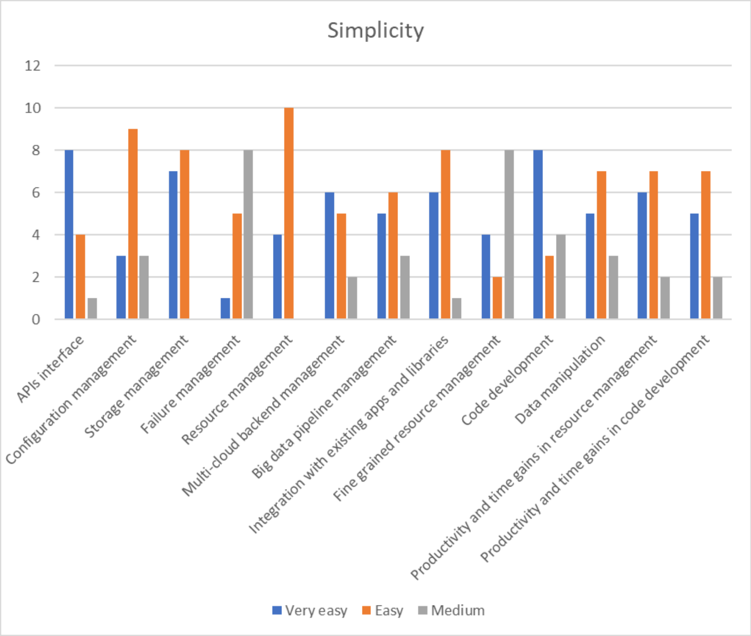
Simplicity KPI (Lithops User Questionnaire)

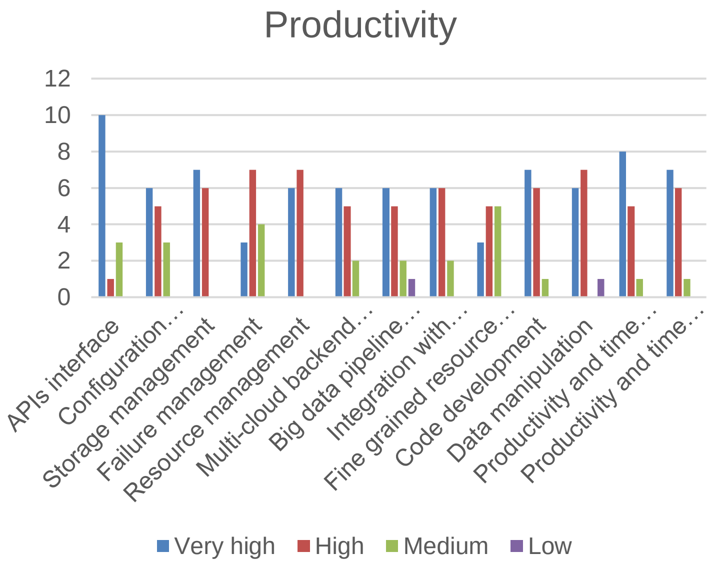
- 16 people testing Lithops for their apps
- UCs representatives but not involved in the project
- 7 aspects evaluated: i) Applicability, ii) Simplicity, iii) Productivity, iv) Scalability, Elasticity and Performance, v) Cost, vi) Learning and documentation, and vii) Overall system evaluation.

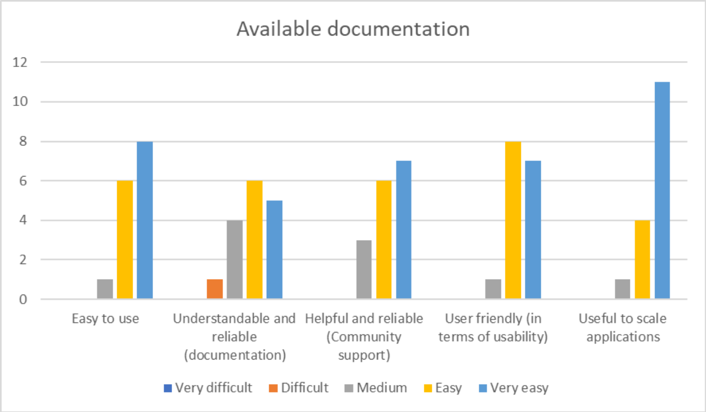


Main outcomes (cont'd)









Use cases

Metabolomics Use Case

CloudButton project impact

New, serverless implementation of METASPACE

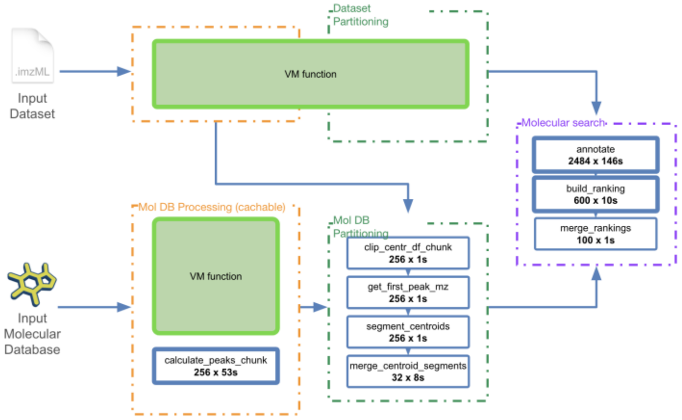
- Using **Lithops** Serverless Data Analytics Platform
- Using **hybrid approach** combining Lithops and Virtual Machines
- By EMBL with the help from IBM Research: 280+ commits

Lithops-METASPACE is used in production since March 2021

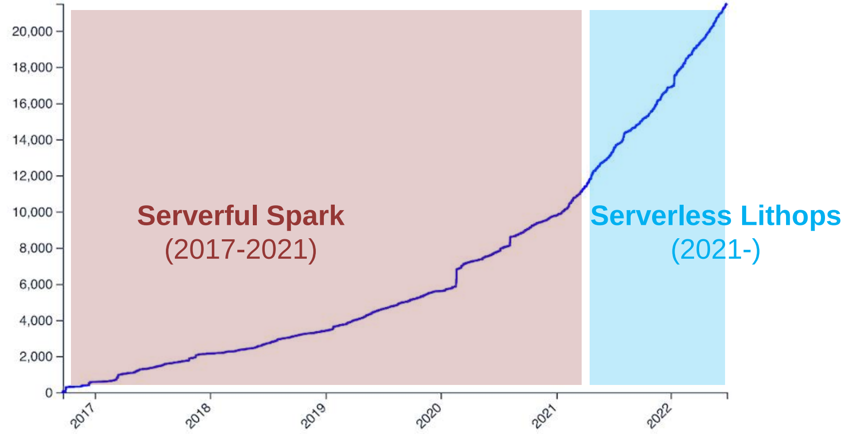
- Replacing (old) serverful Apache Spark implementation
- Lithops-METASPACE was benchmarked (KPIs in the next slide)
- Lithops-METASPACE started to be used in METASPACE production since March 2021
- Since then, in 1.5 years, we have processed ~50% of all submissions (compared to 2017-2021)

Increased exploitation & sustainability

- Helped create the startup SpaceM on spatial single-cell metabolomics
- Helped secure other grants (EU Cloud Computing, ERC PoC, NIH and other grants) to fund METASPACE till 2027



Datasets processed on METASPACE production



KPIs of Lithops-METASPACE: reduced runtime and costs

- Lithops implementation outperforms the (previous) Spark implementation in runtime and cost (for all but one very large dataset)
- Lithops provides a competitive alternative to Apache Spark in code readability and ease of development

Benchmarking the Lithops implementation of METASPACE as compared to the (previous) serverful Apache Spark implementation

Dataset	Size (MB)	Spark time	Lithops time	Relative time	Spark $USD	Lithops $USD	Relative cost
20190228_Rhodamine_Well3_p70s50_POS	63	709.633	331.423	-53%	0.342	0.094	-73%
NPC_179_pos	102	480.642	249.911	-48%	0.226	0.068	-70%
FDtest_exp7_mixed_slide_DAN_Slice2	592	496.158	265.631	-46%	0.234	0.079	-66%
DESI HEART SYNAPT-XS RES-MODE	1,245	630.096	576.766	-8%	0.301	0.266	-12%
150618-RatBrain-DHA-NEG-centroid	1,539	570.610	765.351	34%	0.271	0.251	-7%
2020-02-05_SlideD_DHB_POS_110x280_150umSS_31at	1,994	536.161	648.503	21%	0.254	0.258	2%
MPI//MPIMM_011_FT_P_KM	34,465	610.401	515.240	-16%	0.291	0.190	-35%
region1	39,733	744.605	839.450	13%	0.359	0.210	-42%
2019-12-19_DDN_microbe-spotting_exp2_45_400x900_30	41,532	3703.384	3544.381	-4%	1.853	5.777	212%
k233_combined three datasets	42,653	868.984	590.230	-32%	0.422	0.185	-56%

KPIs of Lithops-METASPACE: elasticity and simplicity

Elasticity: adjustment to the varying datasets sizes

- Submitted datasets size vary: 0.05 GB – 300 GB (4 orders of magnitude!)
- Lithops help handle dataset sizes without reconfiguration by running 1000s parallel jobs (2000 in real applications) per dataset
- Hybrid elastic approach balancing IBM CodeEngine (with RAM up to 32 GB) and VM (RAM of 128+ GB)

Simplicity of implementation

- Ease of Lithops configuring
- Lithops-METASPACE can be run from a notebook (see our GitHub)
- Able to run METASPACE at the same capacity with a reduced team (2 software developers in 2022 vs 3 software developers previously)

Summary

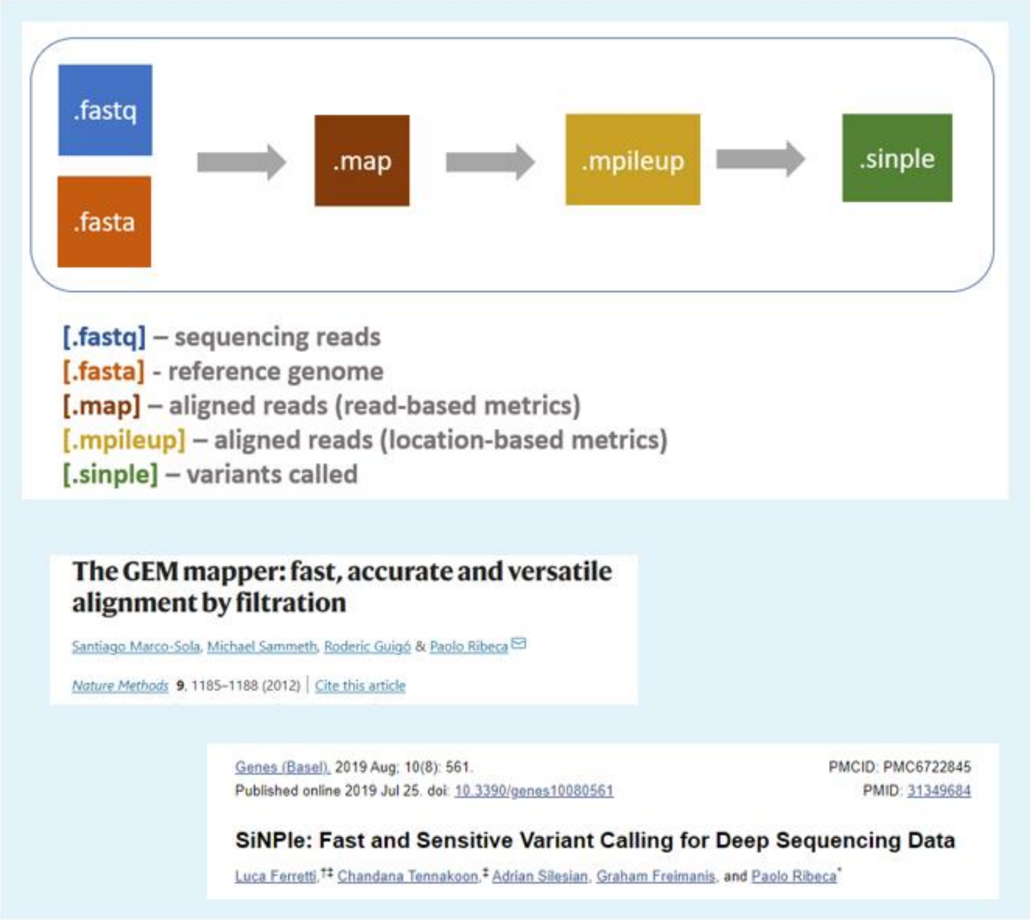
- **METASPACE** is a critically important platform for the scientific community
- With increasing popularity, the previous Spark version became a bottleneck
- Lithops-METASPACE was implemented and is used in production since 2021
- Lithops provides reduced runtime and costs, yet increasing elasticity and simplicity

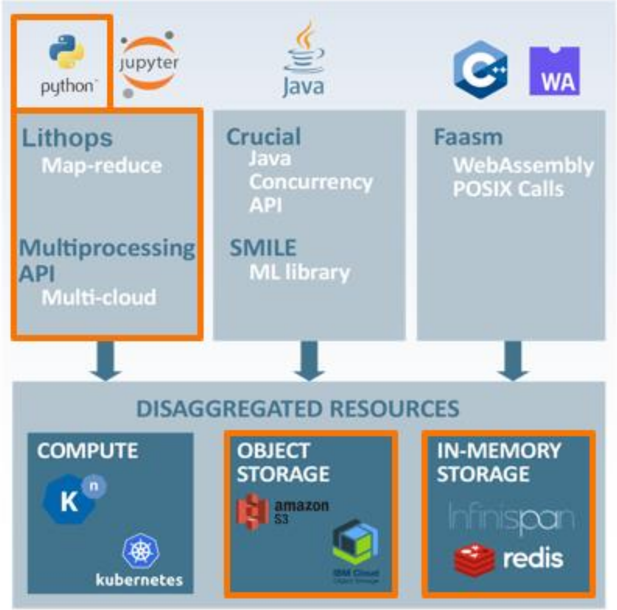
CloudButton

Genomics Use Case

BIG DATA VALUE
PUBLIC-PRIVATE PARTNERSHIP

Variant Calling pipeline



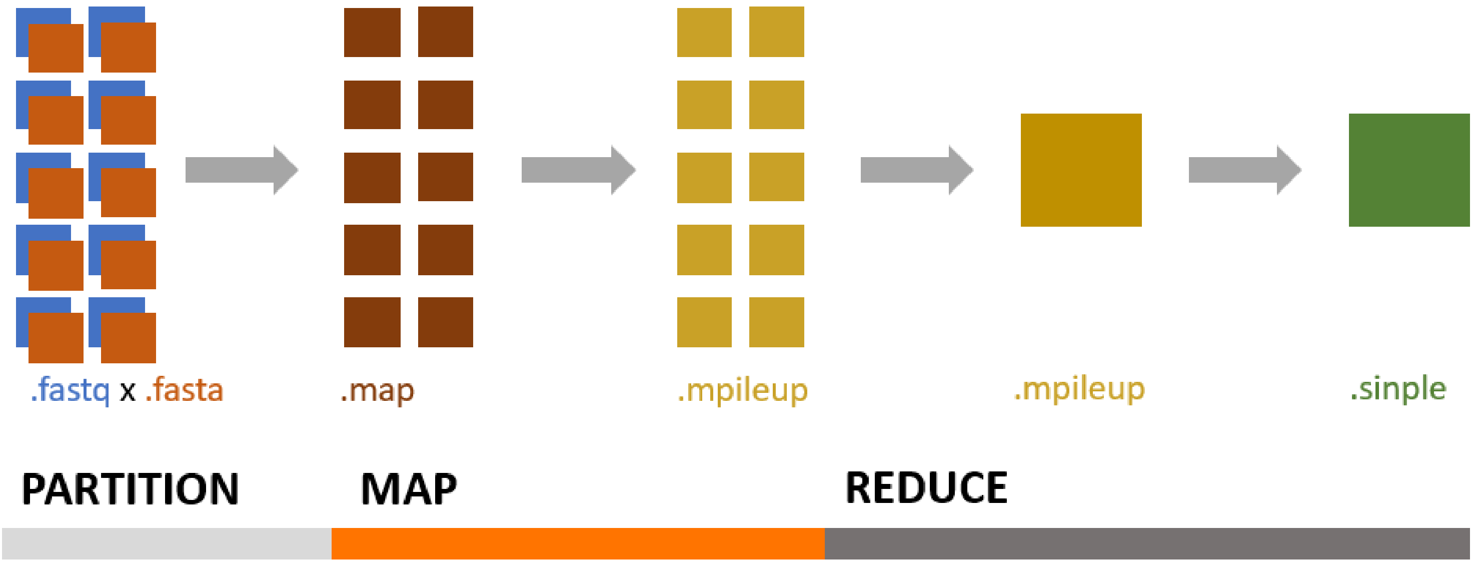


- The requirements of the Genomic Use Case mandate the use of a complex, heterogeneous platform such as the one developed by CloudButton.

Variant Calling pipeline map reduce

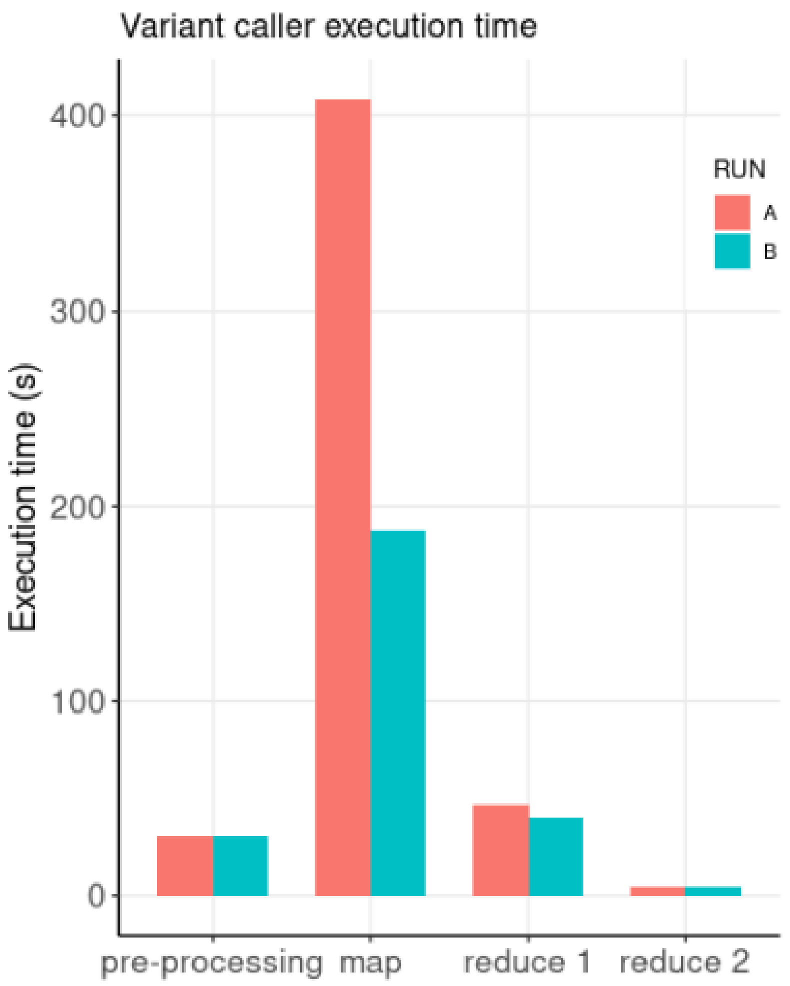
LITHOPS

Multi-cloud python computing framework
Parallelisation of local applications using stateless cloud functions



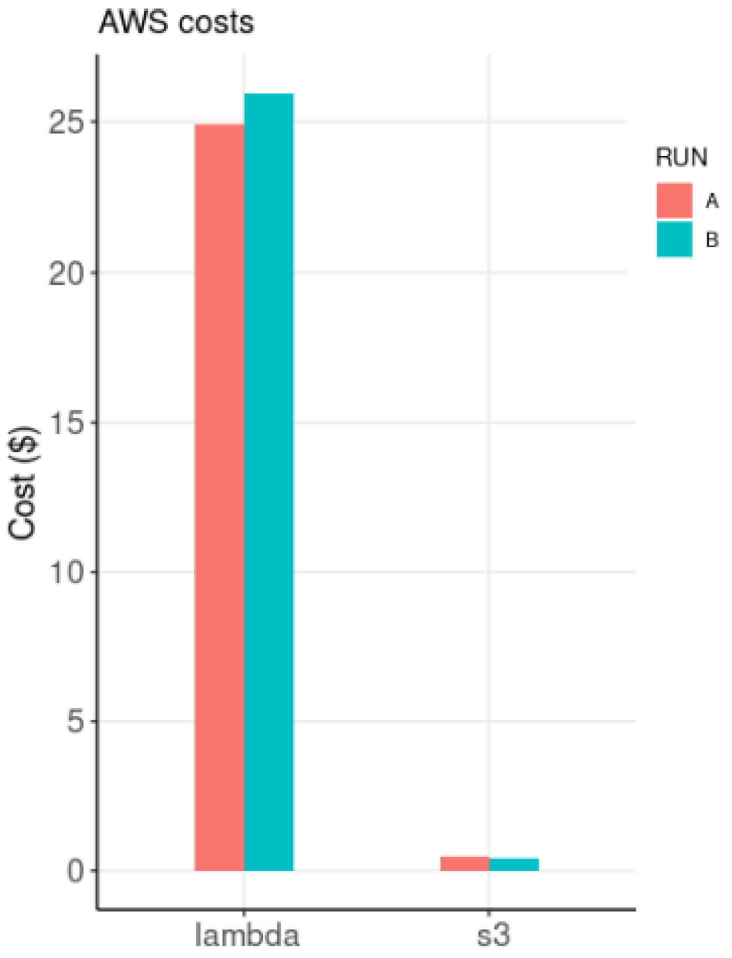
KPIs: Performance

- 101 Gb FASTQ tested (ERR9856489)
- Genomics use case runs A/**B**:
  - 260/**225** FASTQ chunks [385/**444**Mb]
  - 4680/**4050** lambda functions
  - 4096/**5120** Mb mem per function
- **Execution time**:
  - **7/3 min** (up to 33 Gb/min)
  - HPC run ~7 hours
  - Illumina DRAGEN : ~30 min (est.)
- **Huge wall-clock time improvement!**



KPIs: Cost

- 101 Gb FASTQ tested (ERR9856489)
- Genomics use case runs A/**B**:
  - 260/**225** FASTQ chunks [385/**444**Mb]
  - 4680/**4050** lambda functions
  - 4096/**5120** Mb mem per function
- **Cost**:
  - **~25$** for both runs
  - Illumina DRAGEN : **~15$** (est.)
- **Higher cost than Illumina DRAGEN – but <2-fold cost increase to deliver up to 10-fold speed-up, and further optimisations and/or parameter choices possible.**



KPIs: Scalability, elasticity, simplicity

- **Scalability**: Stateful processing of alignment indices across functions poses scalability challenge: other non-Redis approaches under investigation
- **Elasticity**: Time-sensitive sequencing analysis workloads can benefit from the high performance of the serverless architecture developed
- **Simplicity**: Current pipeline design partially data-driven, but with the potential for full automation of partition choices based on cost considerations and input data size.

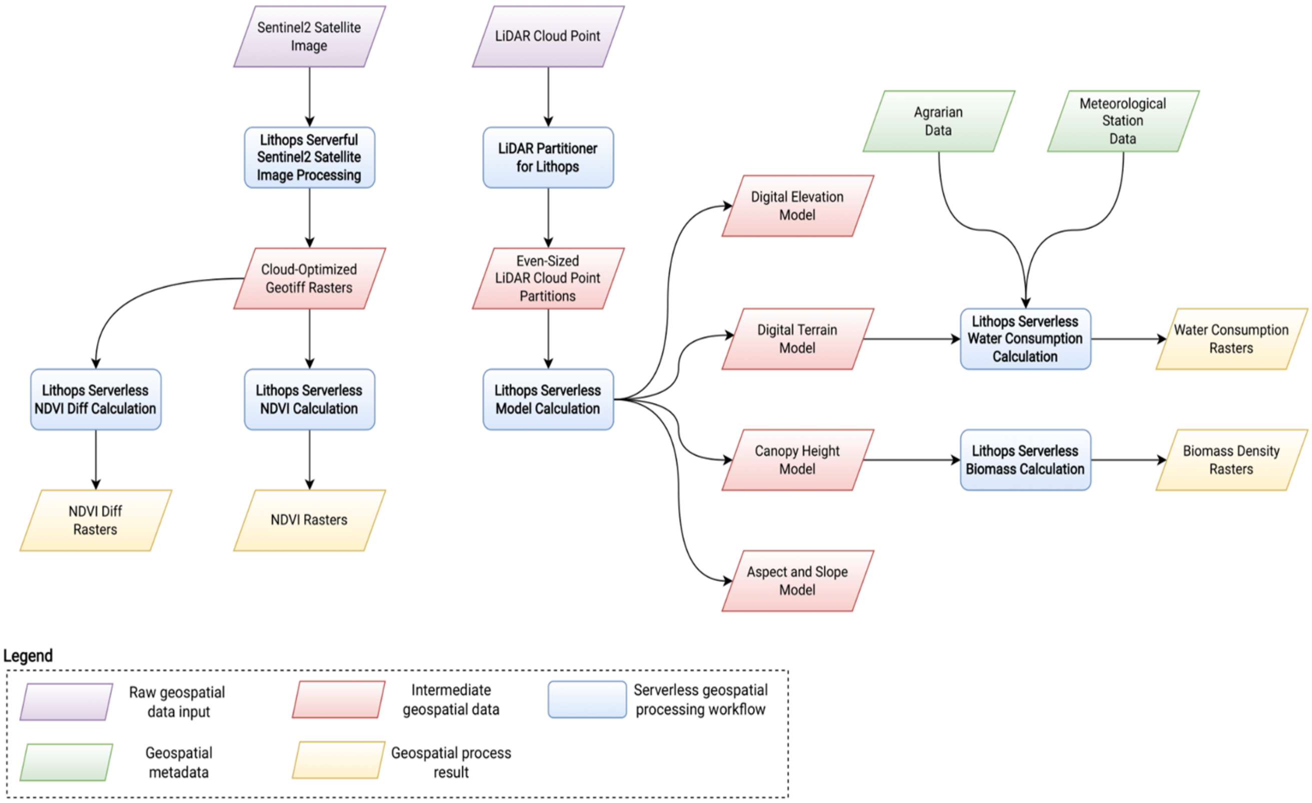
Conclusions

- Porting genomic workflows to serverless cloud can be challenging, but powerful components such as Lithops make the process possible
- Our approach to the Genomics Use Case shows superior wall-clock performance, and probably a record in the field
- The variant calling pipeline developed by the project is expected to be a useful software product in the genomics field
- Further genomics pipelines can be built applying the design principles developed in this use case.

Geospatial Use Case

Overview

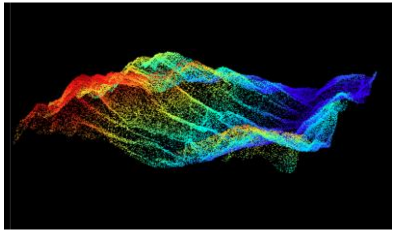
- **Six interconnected serverless geospatial workflows** (3 preprocessing, 3 analytics) - Serverless is capable to preprocess but also to perform analytics

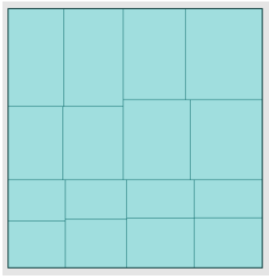


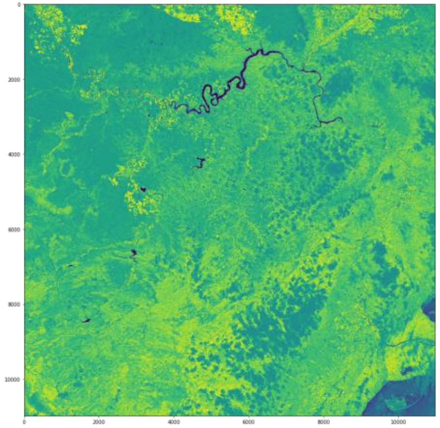
- **E1: High-resolution hybrid land-cover mapping**
  - Sentinel2 Image Processing
  - NDVI Workflow
- **E2: 3D fuel mapping for forest risk assessment**
  - LiDAR partitioner
  - Digital Terrain Model Calculation
  - *Biomass calculation**
- **E3: Water Consumption**
  - Water Consumption Workflow

Preprocessing Workflows

1. **LiDAR point-cloud partitioner tool**
    - Novel density-based LiDAR point cloud partitioner
    - Produces even-sized partitions for efficient serverless computation
2. **Serverless Digital Models Calculation**
    - Generate many geospatial surface models at scale: from LiDAR to GeoTIFF
    - Improved performance thanks to density-based partitioning
3. **Sentinel2 satellite image preprocessing**
    - Preprocess and apply atmospheric correction to Sentinel2 JPEG2000 images and transform to Cloud-Optimized GeoTIFF
    - Lithops allows to seamlessly combine serverful and serverless resources for resource-demanding processes

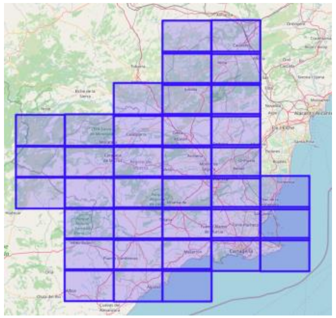


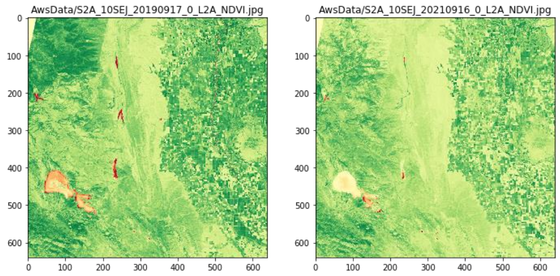


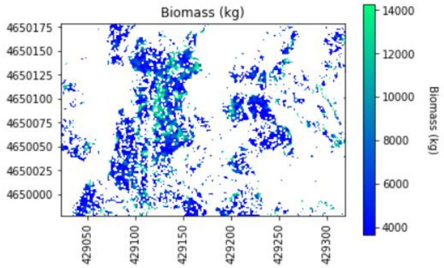


Computing Workflows

4. **NDVI calculation**
   - Calculate NDVI from Sentinel2 preprocessed images to asses deforestation caused by wildfires
   - Cloud-Optimized GeoTIFFs enable to efficiently process satellite images using many parallel serverless functions
5. **Water Consumption calculation**
   - Calculate crop water consumption for better water management in intensive cultivation areas
   - Workflow with great variability in task granularity – serverless is effective for short-running tasks
6. **Biomass calculation**
   - Calculate tree volume from canopy height models to assess biomass
   - Lithops simplifies scaling code from local to the Cloud

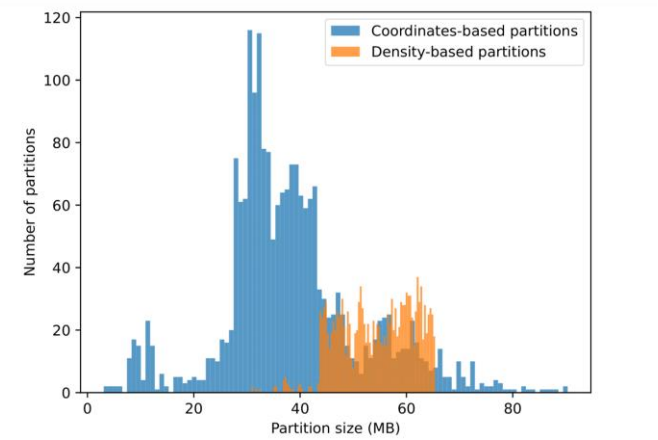


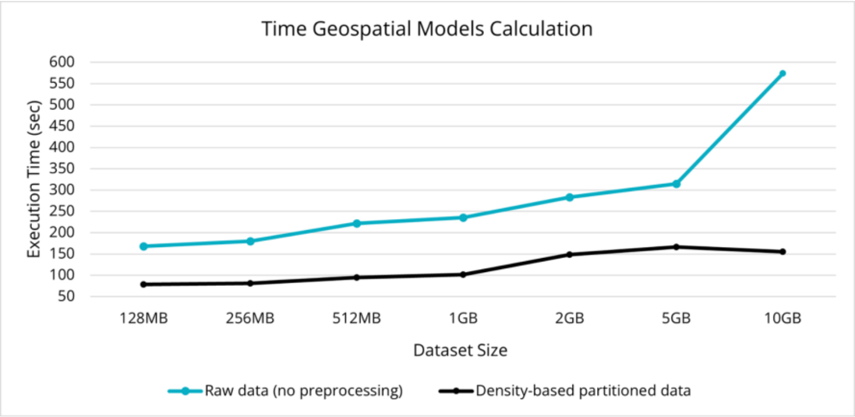




KPIs: Performance

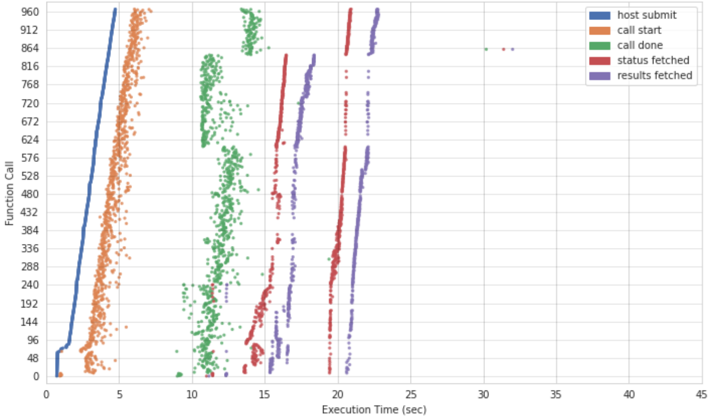
- **LiDAR partitioner tool**
  - Even-sized partitions is beneficial for serverless computation: load balancing is key to gain performance from greater parallelism.
  - Preprocess and partition 516 files, **80 GB** in 2 minutes.
- **Serverless Digital Models Calculation**
  - A finer granularity partitioning allows to exploit parallelism of serverless functions ⇒ Processing a dataset of 10GB is **72% faster** using proper partitioning.

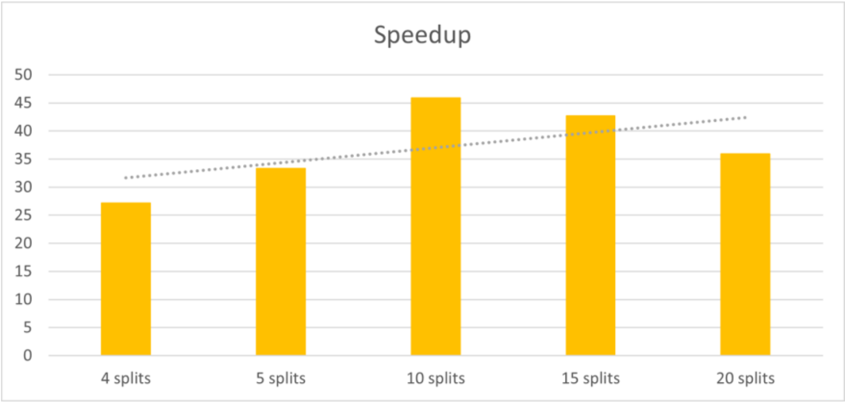




KPIs: Performance, Simplicity

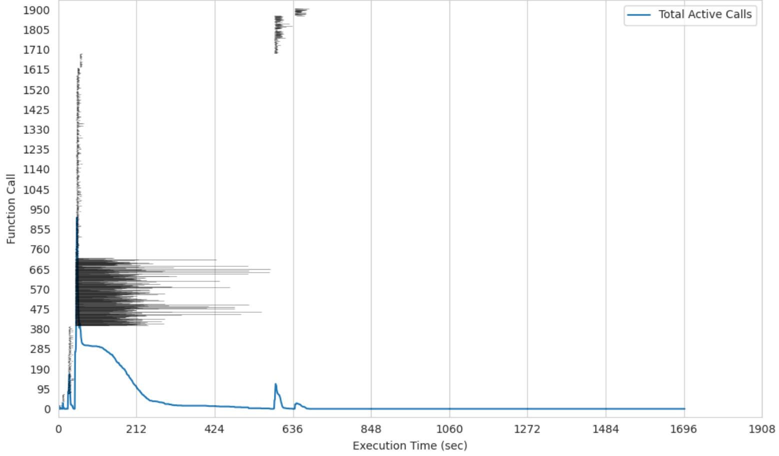
- **NDVI Calculation**
  - Proper pre-processing with cloud-optimized data types enables to use **968 functions** to get **1.25 GB/s processing throughput**.
- **Biomass Calculation**
  - Compute-intensive tasks runs for **4 hours and 20 minutes** on the user's laptop and in **5 minutes and 41 seconds** using **400** parallel serverless functions.
  - Lithops and serverless enables simple Jupyter Notebook task parallelization and scaling for a **speedup gain of 46x**.





KPIs: Elasticity, simplicity

- **Sentinel2 satellite image preprocessing**
  - Lithops allows to seamlessly combine serverful and serverless resources.
- **Water Consumption workflow**
  - Data volume of 6.07 GB which cover **>11.313 km²** of surface area
  - Flexible and efficient resource allocation: from **1296 to 36** functions between steps.



Job ID	Function	Invocations	Memory (MB)	Avg Run time (s)	Cost (USD)
M000	asc_to_geotiff	36	73728	0.85	0.001
M001	get_tile_meta	36	73728	1.27	0.001
M002	split_blocks	324	663552	2.63	0.029
M003	radiation_interpolation	324	663552	149.70	1.649
M004	temperature_interpolation	324	663552	3.35	0.036
M005	humidity_interpolation	324	663552	3.12	0.034
M006	wind_interpolation	324	663552	3.12	0.034
M007	merge_blocks	180	368640	8.33	0.051
M008	combine_calculations	36	73728	14.46	0.017
Summary		**1908**	**3907584 MB**	**28.60 s**	**1.85 $**

Conclusions

Conclusions

- Global architecture with **Lithops as Serverless Data Analytics platform**. All subprojects integrate with Lithops. Metaspace is built on Lithops and it runs in production in IBM Cloud. Infinispan is a Lithops Storage Backend, FaasM is a Lithops Compute Backend.
- **Three transparency efforts** (Python, Java, WebAssembly) weave the project results (WP3, WP4, WP5)
- Lithops was successfully used in the **three use cases** demonstrating **KPIs such as Simplicity, Performance, Scalability and Elasticity**.
- Lithops is **NOT** a competitor of Apache Spark or Ray. It is not an in-memory cluster technology.
- Lithops can be used as Data Staging platform (Preprocessing) but also as simple orchestrator of heterogeneous Cloud services (backends). Lithops is also a perfect tool to parallelize and migrate Big Data applications to the Cloud (DATOMA Cloud)

THANK YOU!

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