Leveraging Intra-Function Parallelism in Serverless Machine Learning

Ionut Predoaia, Pedro García-López

UNIVERSITY of York

UNIVERSITAT ROVIRA i VIRGILI

December 2023

Stateful Machine Learning Algorithms

- As serverless functions are not directly network-addressable, they cannot communicate with each other to share the state related to an iterative machine learning algorithm

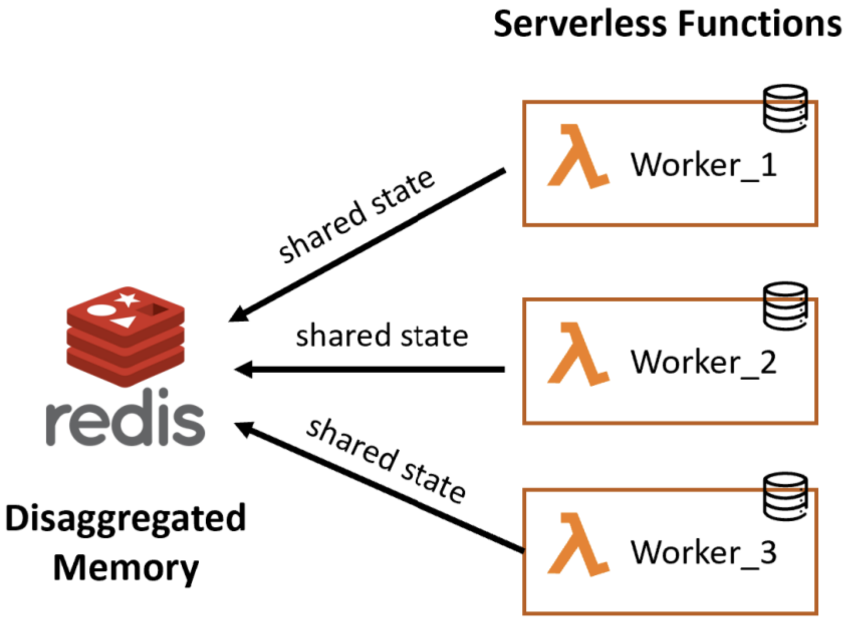
Shared State:
- ➤ the centroids in k-means clustering
- ➤ the gradients in logistic regression

- One must rely on a remote storage service for storing the shared state => large overheads when running hundreds of iterations

Can intra-function parallelism hide the access latency to the remote storage by taking advantage of multiple vCPUs?

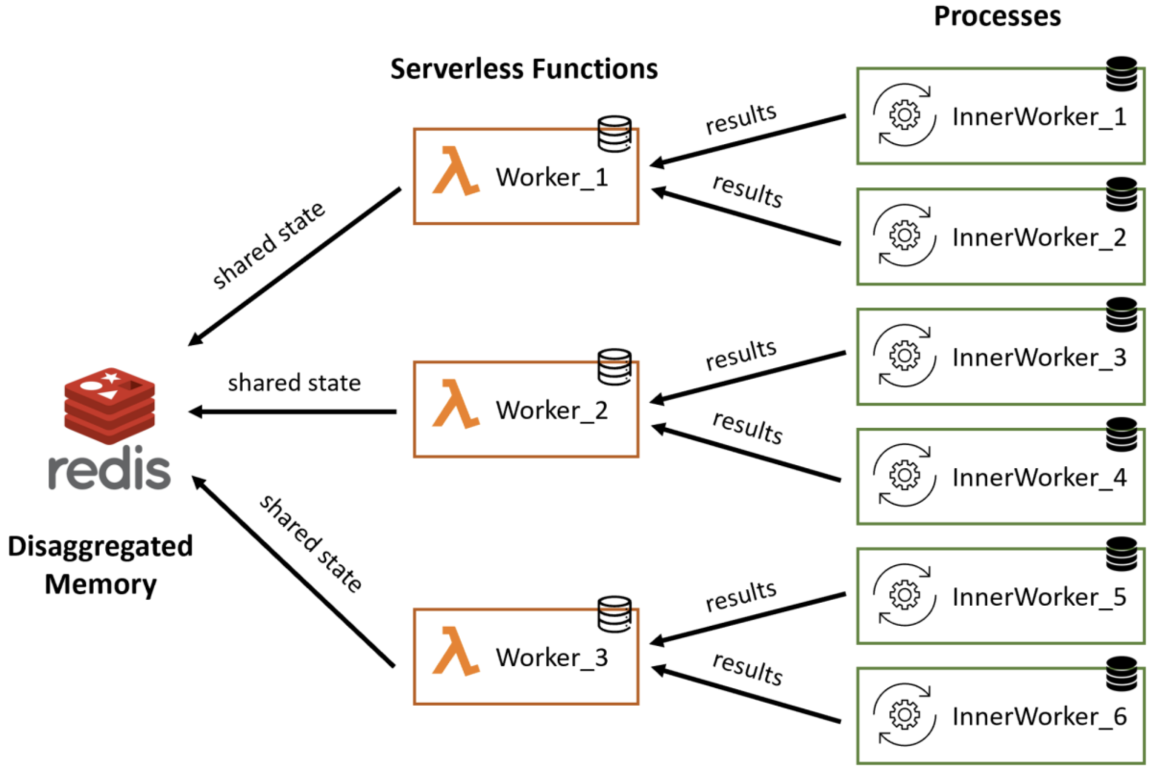
Portage to Serverless

- Lithops has been used for porting k-means clustering and logistic regression to serverless
- Lithops provides a Multiprocessing module containing abstractions that enable sharing state among serverless functions



Adopting Intra-Function Parallelism

- The computation phase of the algorithms is carried out by inner workers, rather than workers
- The number of inner workers is dictated by the number of vCPUs of a serverless function
- A higher level of parallelism can be achieved with a fewer number of connection points to Redis => reduced synchronization overheads



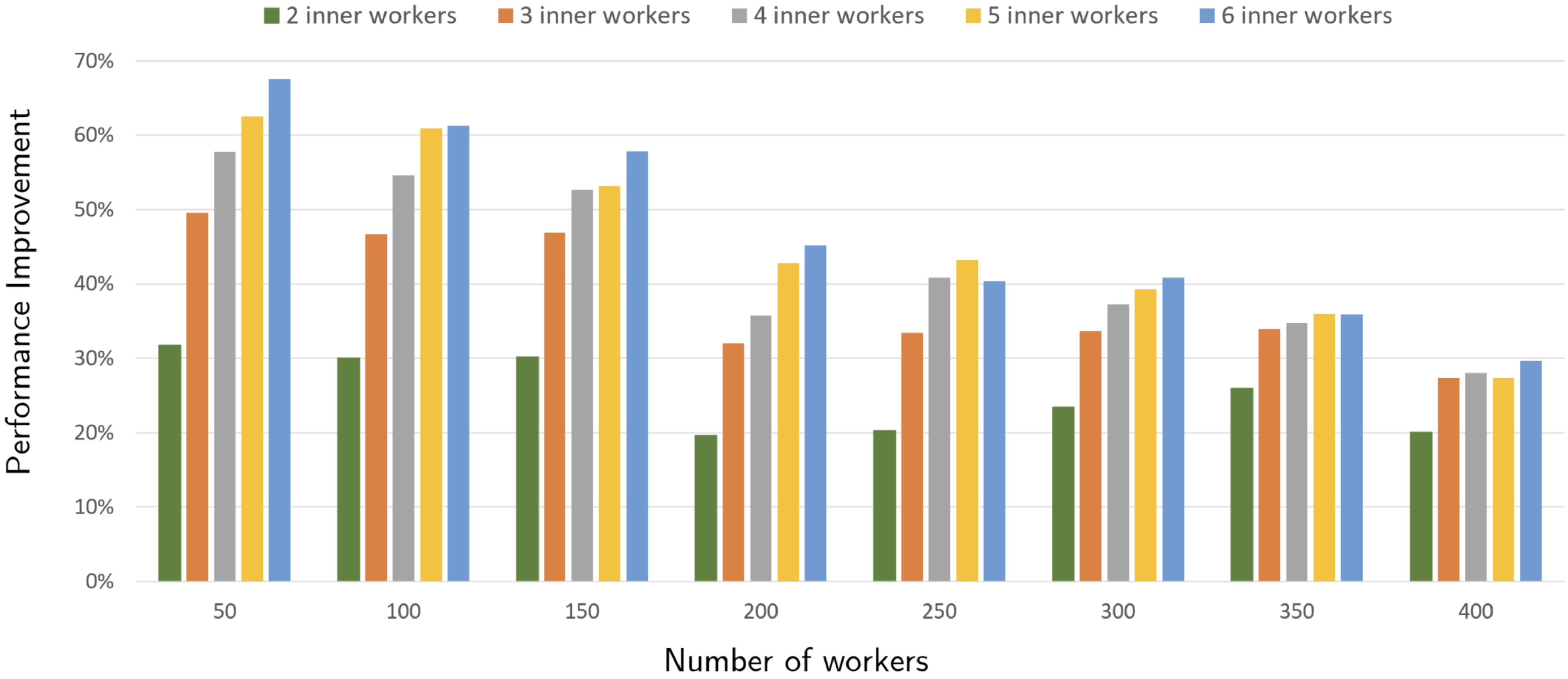
Experimental Evaluations

Experiment 1 - Description

- The k-means algorithm has been executed with a data set of 8GB and with a growing number of serverless functions, from 50 up to 400
- Each serverless function had 6 vCPUs allocated
- In the first (baseline) instance, the k-means algorithm was executed without employing intra-function parallelism
- Then the k-means algorithm was executed by employing intra-function parallelism, by using 2 up to 6 vCPUs of each serverless function

Experimental Evaluations

Experiment 1 - Results



Experimental Evaluations

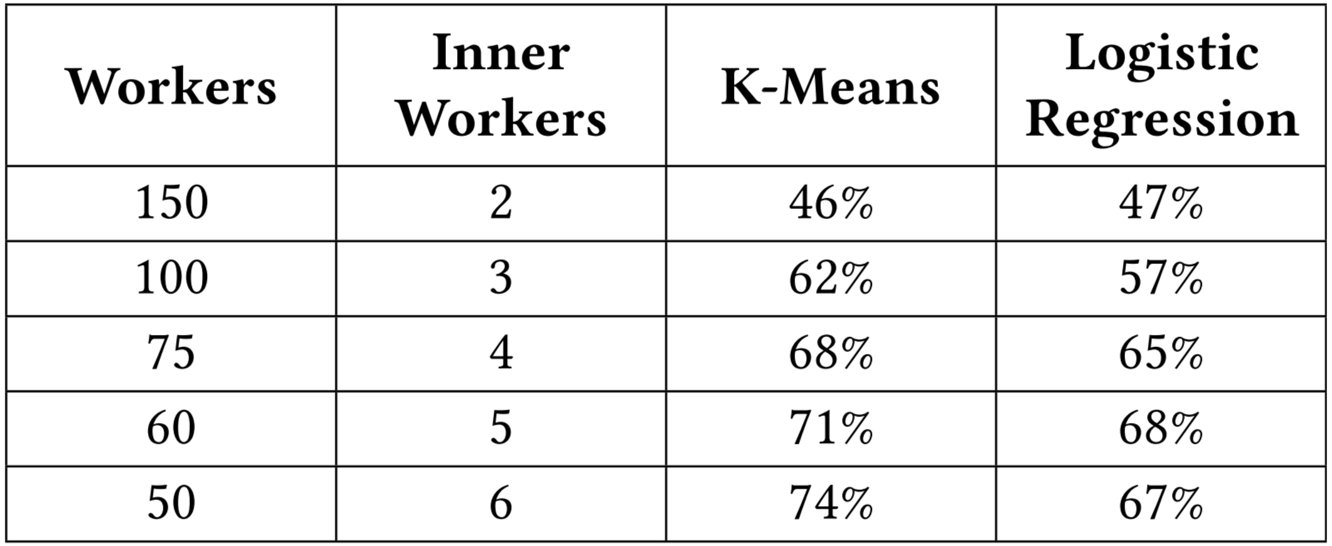
Experiment 2 - Description

- One may argue that the previous experiment does not carry out a fair evaluation because different levels of parallelism were achieved
- This experiment aims to evaluate the performance improvement obtained when leveraging intra-function parallelism, whilst maintaining the same level of parallelism
- For example, it is to be determined whether a better performance can be obtained when invoking 50 serverless functions, each with 6 vCPUs, that leverage intra-function parallelism, compared to when invoking 300 serverless functions, each with only 1 vCPU

Experimental Evaluations

Experiment 2 - Results

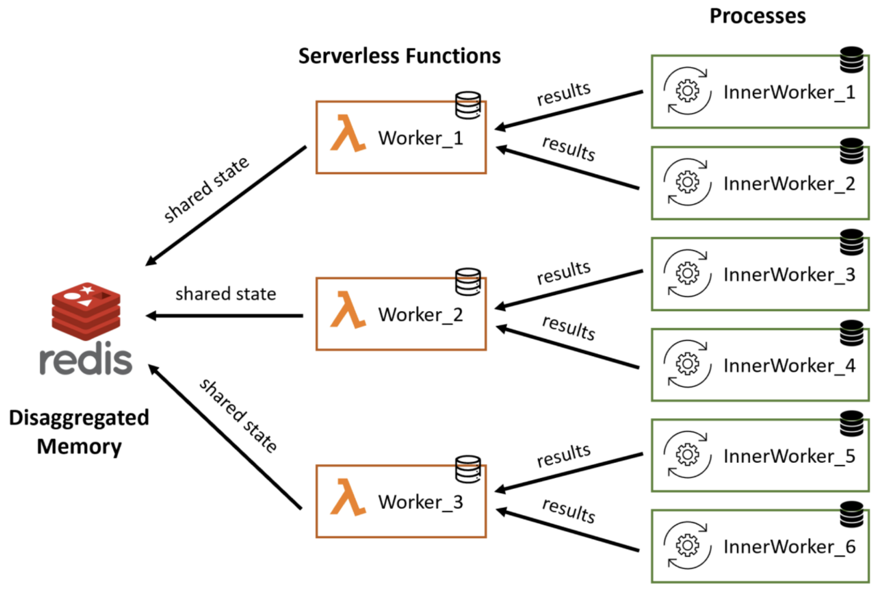
- As a baseline, the algorithms were executed with 300 workers, where each serverless function had 1 vCPU allocated with 1500MB of memory
- The memory of the serverless functions has been proportionally increased for each additional vCPU



Workers	Inner Workers	K-Means	Logistic Regression
150	2	46%	47%
100	3	62%	57%
75	4	68%	65%
60	5	71%	68%
50	6	74%	67%

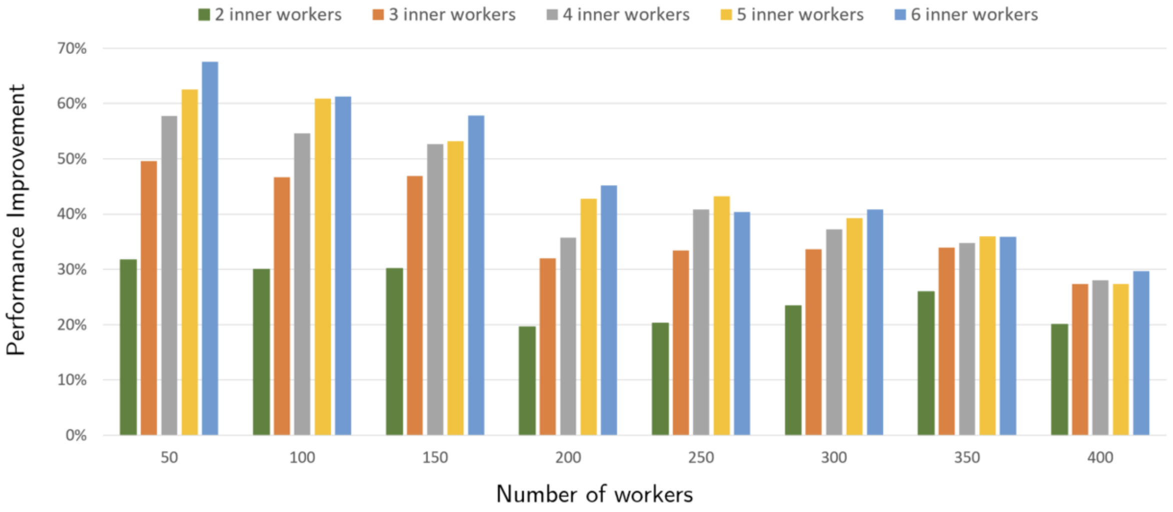
Summary

- We ported two stateful machine learning algorithms to serverless, k-means clustering and logistic regression, and then adopted intra-function parallelism
- Improved performances of up to 68% have been achieved by leveraging intra-function parallelism
- We demonstrated that from a performance perspective, it is preferable to execute a smaller number of multiple-vCPUs workers than a larger number of single-vCPU workers, due to decreased synchronization overheads



Experimental Evaluations

Experiment 1 - Results



Experimental Evaluations

Experiment 2 - Results

- As a baseline, the algorithms were executed with 300 workers, where each serverless function had 1 vCPU allocated with 1500MB of memory
- The memory of the serverless functions has been proportionally increased for each additional vCPU

Workers	Inner Workers	K-Means	Logistic Regression
150	2	46%	47%
100	3	62%	57%
75	4	68%	65%
60	5	71%	68%
50	6	74%	67%

Experimental Evaluations

Configuration Setup

- All experiments have been conducted in AWS in the same VPC
- The serverless functions are running via AWS Lambda
- The data sets are stored in Amazon S3

Parameter	**Value**
Resources Region	eu-west2 (Europe - London)
Redis Node Instance Type	r5.large (memory optimized - 2 vCPU, 16GB RAM)
Client Machine Instance Type	t2.2xlarge (general purpose - 8 vCPU, 32GB RAM)
AWS Lambda Timeout	15 minutes

Limitations

- The amount of memory determines the number of vCPUs available to a serverless function => one may want to employ intra-function parallelism to leverage the multiple vCPUs, but may not need the additional amount of memory which brings additional costs
- AWS Lambda does not provide shared memory for processes, therefore we had to rely on pipes for sending the output of the inner workers to the (parent) workers => large transfer overheads may be induced