Serverless Evolutionary Computation

Background:
Evolutionary algorithms model biological evolution using mechanisms such as reproduction, mutation, recombination, and selection to create candidate solutions for often complex optimisation problems.
They typically produce near-optimal solutions to problems that are far too computationally-intensive to find an exact solution for.
The general process involves random individuals (potential solutions) in an initial population (fixed sized) undergoing the above described operations to improve solution quality (measured by a fitness function) over a series of generations.
After a series of genetic operations performed on each individual they are evaluated to determine the fitness. Before evaluation there are often more individuals than the population permits (Notably; reproduction) therefore evaluation aims to evolve the populations average fitness by removing the weakest individuals at the end of each generation.
This cycle is repeated over many generations until a population produces a candidate solution that best fits the fitness criteria.

Problem:
Evolutionary algorithms are highly parallelizable as genetic operations are typically quick and simple. Each individual does not need to be operated sequentially; instead, they can be distributed across concurrent systems for asynchronous computation. Parallelization allows for a quicker throughput of generations resulting in expediated discovery of candidate solutions.
The parallelization of evolutionary algorithms has scaled with the evolution of concurrent systems. From multi-processor machines to multi-core processors to multi-thread processor cores, evolutionary algorithms have been adapted to exploit architectural improvements in parallel processing.
In recent years processor core counts, clock rates and overall processing power has stagnated. These barriers have led to new innovations such as the Serverless paradigm where architecture is virtualised to the point where hardware is not provisioned. Code is written as functions that are independent from the underlying infrastructure. Therefore, computation power can auto-scale to meet the demand of the executing code. This paradigm provides an opportunity for evolutionary algorithms to exploit degrees of parallelization that cannot be achieved by single servers (or even small clusters of servers).

Scope:
Develop, compare, contrast various serverless evolutionary strategies using modern serverless services (e.g. AWS Lambda, Azure Functions, Google Cloud Functions). Compare performance to already established non-serverless systems.

Extended scope:
Integration of in-memory cached database systems (e.g. Amazon DynamoDB Accelerator (DAX)) or other high performance scalable database systems allowing for data to be quickly fed into a real-time continuous optimisation system exporting solutions at regular intervals.