gen-i

SMART ENERGY

AWS, some small hints for Dynamo DB and Lambda

David Grgič

2023-06-15

gen-i

- Sun (Sonce)…
- Retail…
- Trading:
  - gen-i do not owning production or storage assets.
  - Data driven business model.
  - Trading based on:
    - Technical Analysis (RSI, MACD, etc.)
    - Fundamental Models:
      - …
      - **Market Simulator**
      - …

gen-i

gen-i Market Simulator

- Input:
  - Power consumption.
  - Power plants with their technical parameters, availability, etc.
  - Fuel and CO2 prices.
  - Connection between countries (price zones).
  - Flow Based Market Coupling data.
- Output:
  - Future prices for most European countries (price zones).
- Challenges:
  - MILP type of a problem with several thousands of optimisation variables per each simulated hour.
  - Simulating up to end of next year delivery periods, e.g. more than ten thousand hours.
  - Scenarios to estimate risk associated with several fundamental parameters: $\prod n_i$
  - Compute intensive task, where results are required ASAP, but simulations are run sporadically:

    $\Rightarrow$ AWS: Lambda, DynamoDB, Step Functions

gen-i

AWS DynamoDB, introduction

- Fully managed.
- NoSQL database.
- Document database, data is stored in a flexible JSON-like format.
- Less structured than traditional SQL databases.
- Fast and predictable performance.
- Seamless scalability.

gen-i

DynamoDB read (write) capacity

- Why it is important:
  - Related to costs!
  - If exceeded, operation (read or write) can be rejected.
- Reading data via „Query“ and „Scan“:
  - „ProjectionExpresion“ to limit retrieved attributes.

- Findings:
  - „ProjectionExpression“ does not decrease read capacity, since consumed read capacity is derived from total item size!
  - Read capacity is not derived from read items but from largest item in the database, even if that particular item is not part of read items!

gen-i

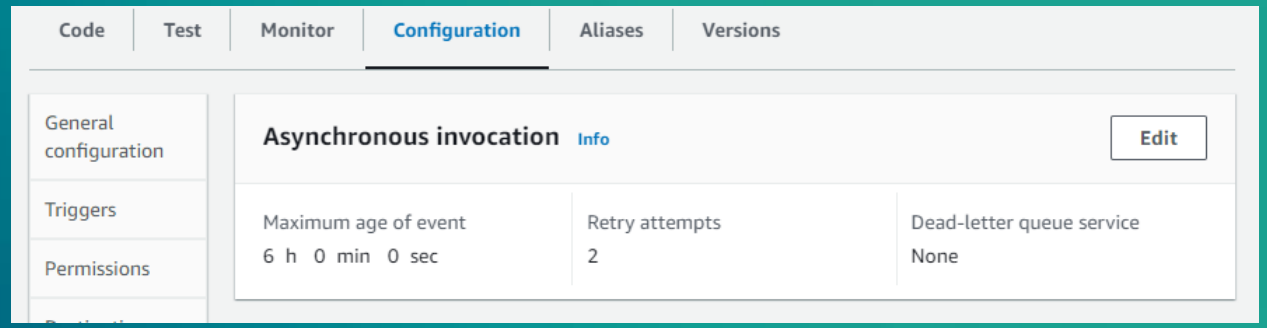
AWS Lambda, introduction

- Serverless compute service:
  - Run your code without provisioning or managing servers.
- Can be event triggered: S3, HTML request, DynamoDB, etc.
- Supported variety of programming languages:
  - Python,
  - Java,
  - .NET,
  - etc.
- High availability.
- Automatic scaling.

gen-i

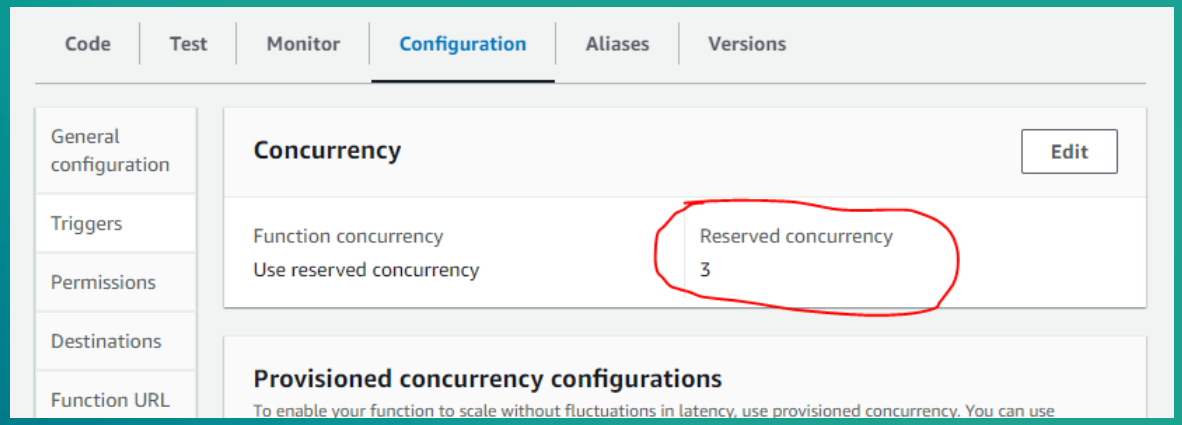
Lambda provisioning

- Available lambdas:
  - Each AWS account has some number of Lambdas, available per AWS location:
    - This number of Lambdas can be increased by request, without costs.
- Retry for asynchronous invocations:
  - When throttled due to:
    - "Too Many Requests" or
    - "ThrottlingException"
  - Up to two times (1x initial + 2x retry)
  - Exponential backoff delay between retries.



Lambda provisioning

- Reserved concurrency:
  - If function has defined reserved concurrency, than:
    - This function can use only up to reserved number of lambdas!
    - Remaining lambdas for other functions are decreased accordingly.
  - Max: account lambdas - 100



- Burst concurrency (lambda scaling limitation):
  - What does it mean:
    - Even if you have ten thousands of lambdas per account, they are not available instantly.
    - First minute you have, e.g. 1000* lambdas, each consequent minute you have 500 more lambdas: 1000; 1500; 2000; 2500; 3000; …
    - If load decrease, e.g. number of used lambdas, you again have a limited ramp for increase (500 more each minute).
  - * Initial burst concurrency is location dependent (3000: us-east-1, us-west-2, eu-west-1; 1000: **eu-central-1**, etc.; 500: other)
  - Reserved concurrency does not save you from burst concurrency limitation!
  - Can not be changed!

gen-i