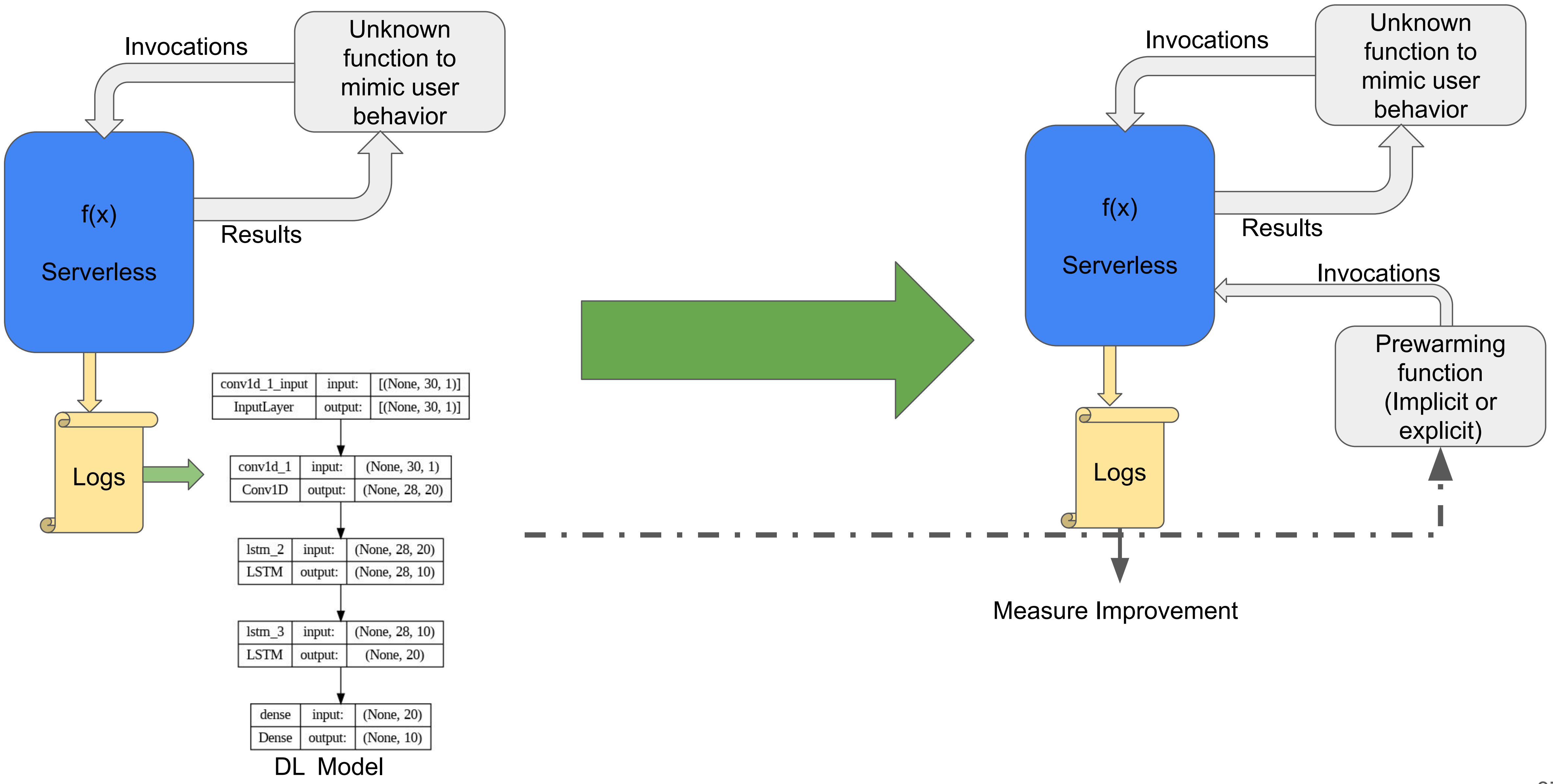
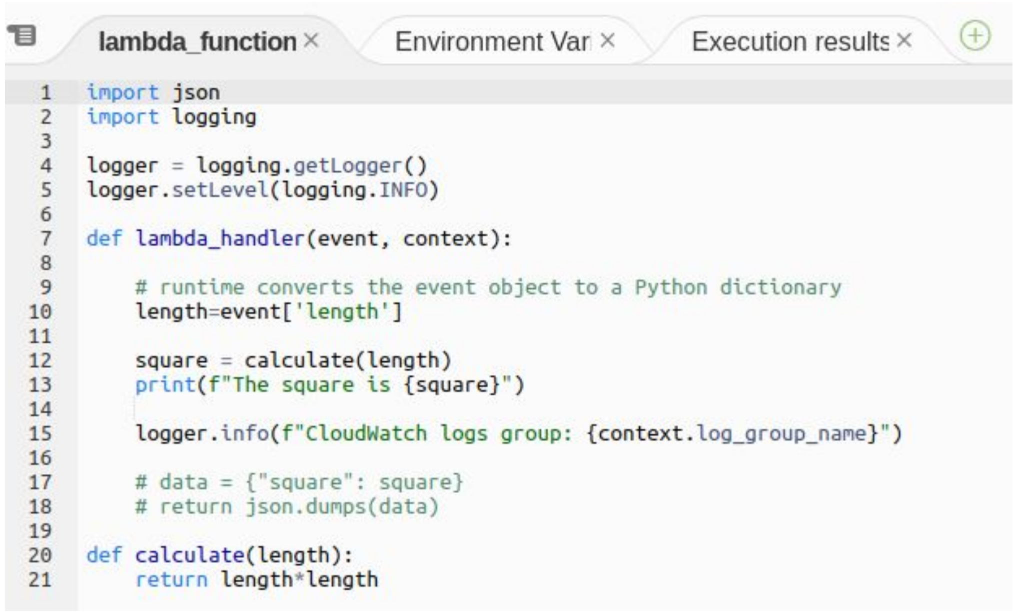
Experimentation Setup


Invocations
Unknown function to mimic user behavior
f(x)
Serverless
Results
Logs
conv1d_1_input | input: | [(None, 30, 1)]
InputLayer | output: | [(None, 30, 1)]
conv1d_1 | input: | (None, 30, 1)
Conv1D | output: | (None, 28, 20)
lstm_2 | input: | (None, 28, 20)
LSTM | output: | (None, 28, 10)
lstm_3 | input: | (None, 28, 10)
LSTM | output: | (None, 20)
dense | input: | (None, 20)
Dense | output: | (None, 10)
DL Model
Invocations
Unknown function to mimic user behavior
f(x)
Serverless
Results
Invocations
Prewarming function (Implicit or explicit)
Logs
Measure Improvement

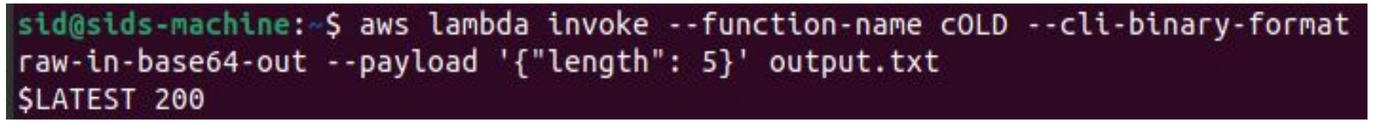
AWS Lambda

- The function



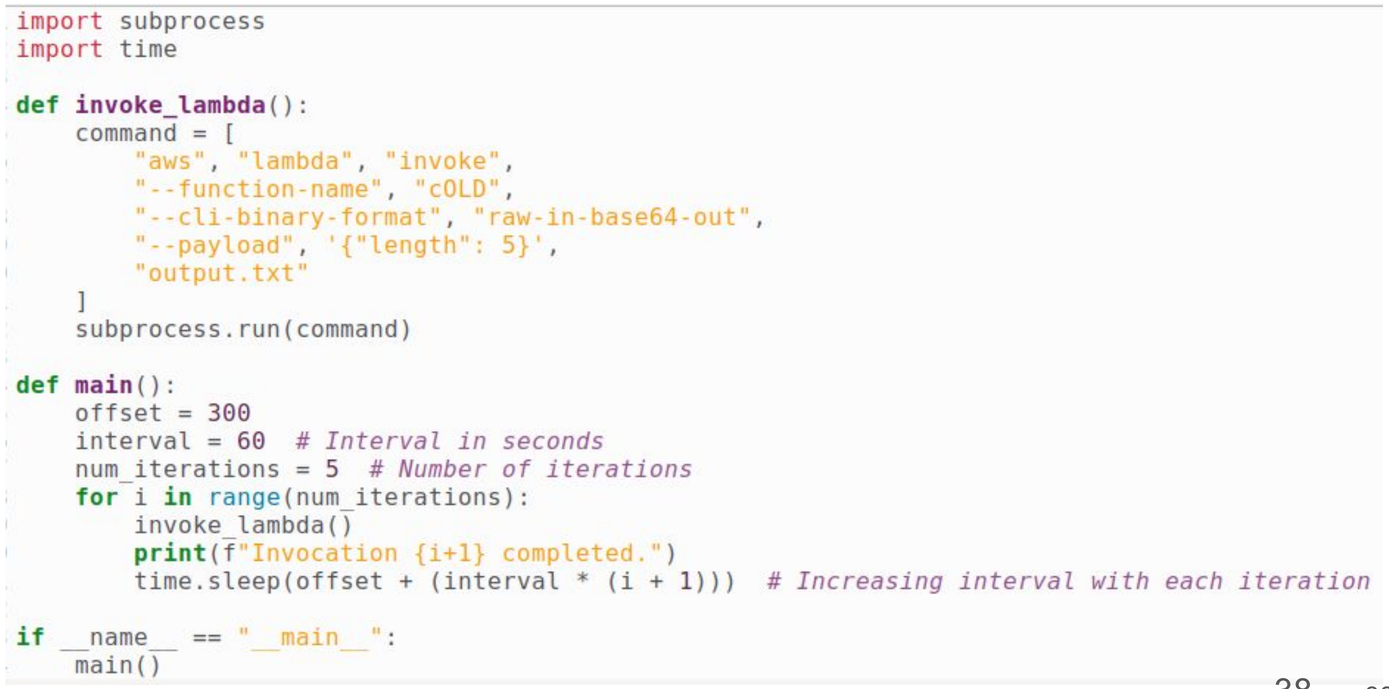
```python
import json
import logging

logger = logging.getLogger()
logger.setLevel(logging.INFO)

def lambda_handler(event, context):

    # runtime converts the event object to a Python dictionary
    length=event['length']

    square = calculate(length)
    print(f"The square is {square}")

    logger.info(f"CloudWatch logs group: {context.log_group_name}")

    # data = {"square": square}
    # return json.dumps(data)

def calculate(length):
    return length*length
```

- CLI integration with local system to run function invocation scripts



```
sid@sids-machine:~$ aws lambda invoke --function-name cOLD --cli-binary-format
raw-in-base64-out --payload '{"length": 5}' output.txt
$LATEST 200
```

- Python Script for invocation

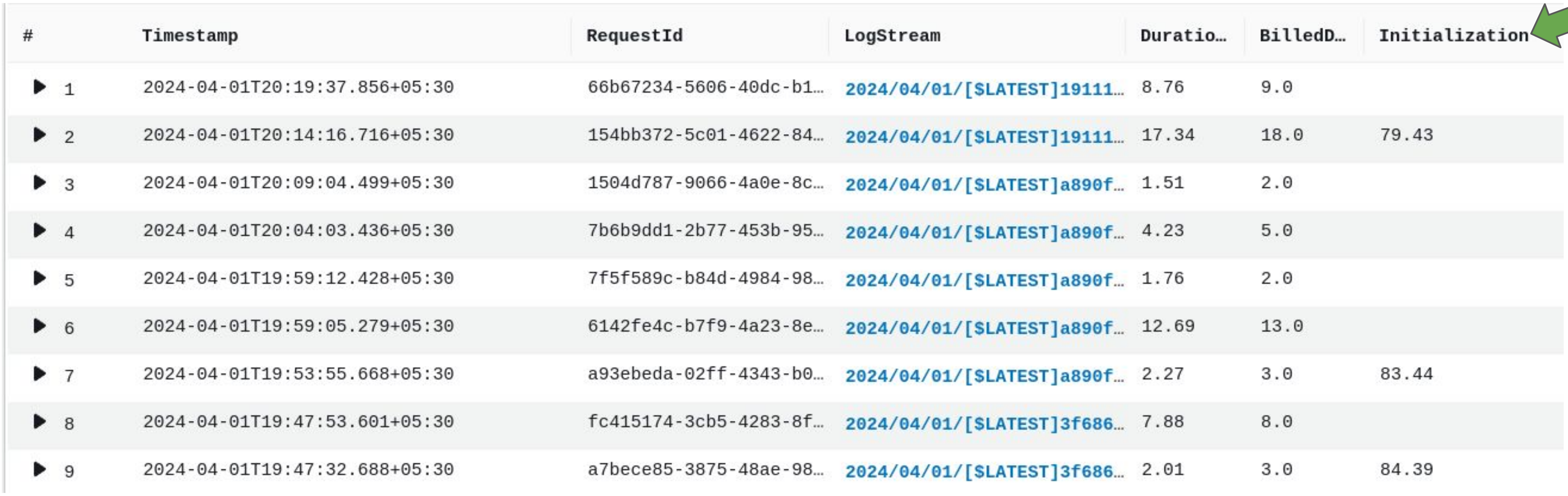


```python
import subprocess
import time

def invoke_lambda():
    command = [
        "aws", "lambda", "invoke",
        "--function-name", "cOLD",
        "--cli-binary-format", "raw-in-base64-out",
        "--payload", '{"length": 5}',
        "output.txt"
    ]
    subprocess.run(command)

def main():
    offset = 300
    interval = 60  # Interval in seconds
    num_iterations = 5  # Number of iterations
    for i in range(num_iterations):
        invoke_lambda()
        print(f"Invocation {i+1} completed.")
        time.sleep(offset + (interval * (i + 1)))  # Increasing interval with each iteration

if __name__ == "__main__":
    main()
```

AWS Lambda

- 1-D grid search for identifying cold -start
- Adding Cloudwatch for logs with appropriate parameters

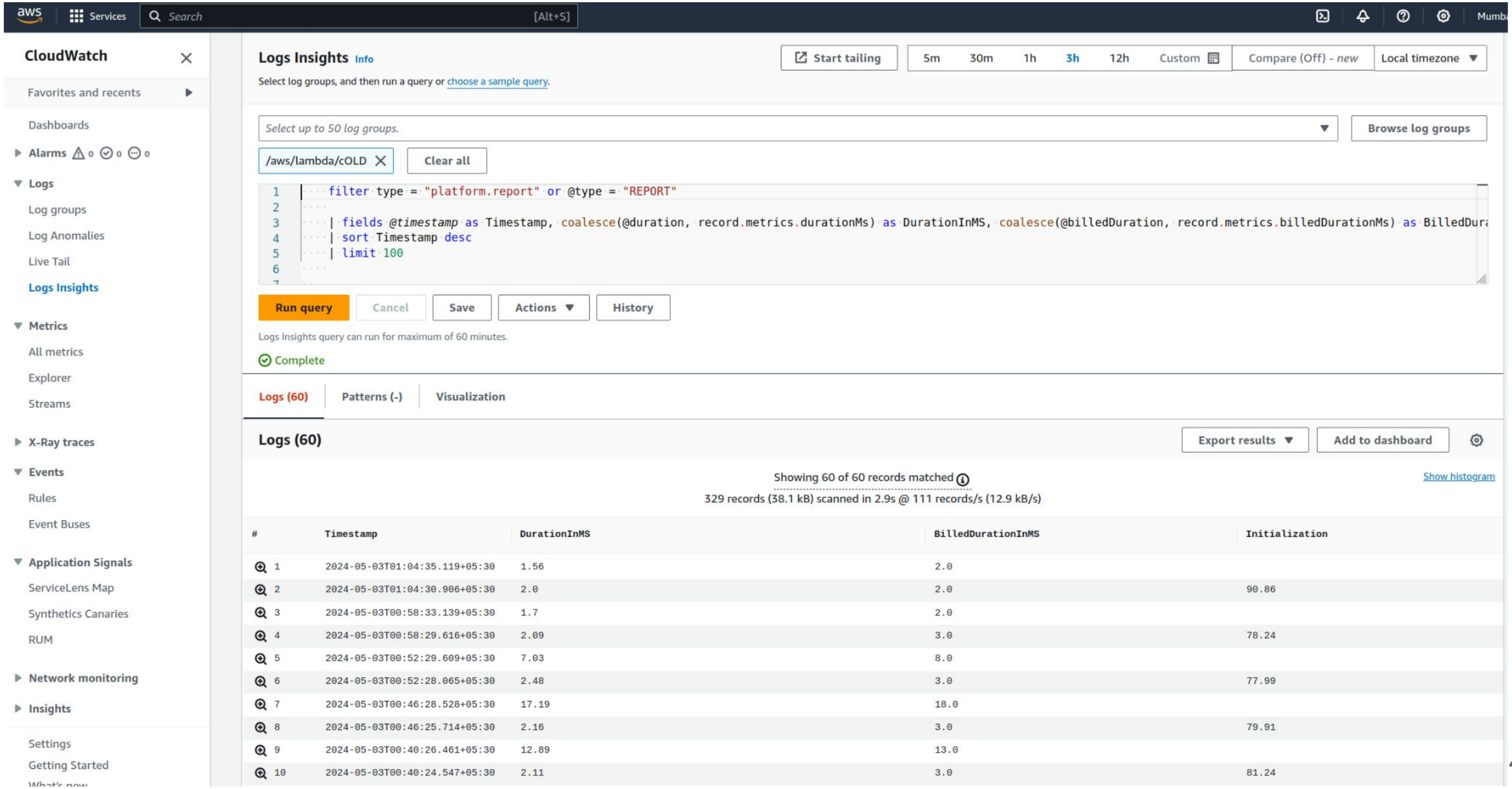
Cold-start Indicator



#	Timestamp	RequestId	LogStream	Duratio…	BilledD…	Initialization
▶ 1	2024-04-01T20:19:37.856+05:30	66b67234-5606-40dc-b1…	2024/04/01/[$LATEST]19111…	8.76	9.0	
▶ 2	2024-04-01T20:14:16.716+05:30	154bb372-5c01-4622-84…	2024/04/01/[$LATEST]19111…	17.34	18.0	79.43
▶ 3	2024-04-01T20:09:04.499+05:30	1504d787-9066-4a0e-8c…	2024/04/01/[$LATEST]a890f…	1.51	2.0	
▶ 4	2024-04-01T20:04:03.436+05:30	7b6b9dd1-2b77-453b-95…	2024/04/01/[$LATEST]a890f…	4.23	5.0	
▶ 5	2024-04-01T19:59:12.428+05:30	7f5f589c-b84d-4984-98…	2024/04/01/[$LATEST]a890f…	1.76	2.0	
▶ 6	2024-04-01T19:59:05.279+05:30	6142fe4c-b7f9-4a23-8e…	2024/04/01/[$LATEST]a890f…	12.69	13.0	
▶ 7	2024-04-01T19:53:55.668+05:30	a93ebeda-02ff-4343-b0…	2024/04/01/[$LATEST]a890f…	2.27	3.0	83.44
▶ 8	2024-04-01T19:47:53.601+05:30	fc415174-3cb5-4283-8f…	2024/04/01/[$LATEST]3f686…	7.88	8.0	
▶ 9	2024-04-01T19:47:32.688+05:30	a7bece85-3875-48ae-98…	2024/04/01/[$LATEST]3f686…	2.01	3.0	84.39

Container teardown time is ~ 300 seconds

Log-monitoring



Logs Insights Info

Select log groups, and then run a query or choose a sample query.

Select up to 50 log groups.

/aws/lambda/cOLD

```
filter type = "platform.report" or @type = "REPORT"

    | fields @timestamp as Timestamp, coalesce(@duration, record.metrics.durationMs) as DurationInMS, coalesce(@billedDuration, record.metrics.billedDurationMs) as BilledDura
    | sort Timestamp desc
    | limit 100
```

Logs Insights query can run for maximum of 60 minutes.

Complete

Logs (60)

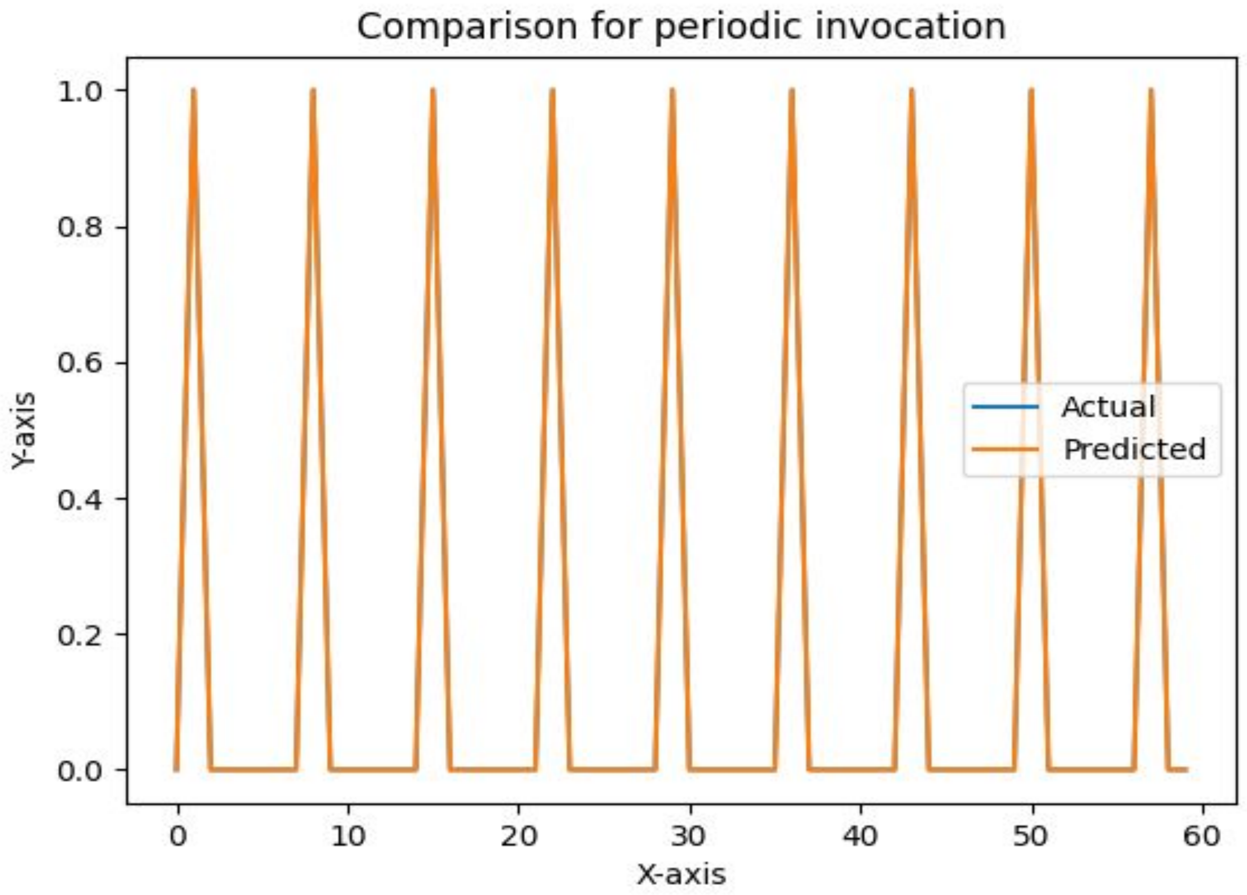
Showing 60 of 60 records matched

329 records (38.1 kB) scanned in 2.9s @ 111 records/s (12.9 kB/s)

#	Timestamp	DurationInMS	BilledDurationInMS	Initialization
1	2024-05-03T01:04:35.119+05:30	1.56	2.0	
2	2024-05-03T01:04:30.906+05:30	2.0	2.0	90.86
3	2024-05-03T00:58:33.139+05:30	1.7	2.0	
4	2024-05-03T00:58:29.616+05:30	2.09	3.0	78.24
5	2024-05-03T00:52:29.609+05:30	7.03	8.0	
6	2024-05-03T00:52:28.065+05:30	2.48	3.0	77.99
7	2024-05-03T00:46:28.528+05:30	17.19	18.0	
8	2024-05-03T00:46:25.714+05:30	2.16	3.0	79.91
9	2024-05-03T00:40:26.461+05:30	12.89	13.0	
10	2024-05-03T00:40:24.547+05:30	2.11	3.0	81.24

Results

- For periodic Invocation (Time period = 6 minutes)
  - Test-time (3+ hours)



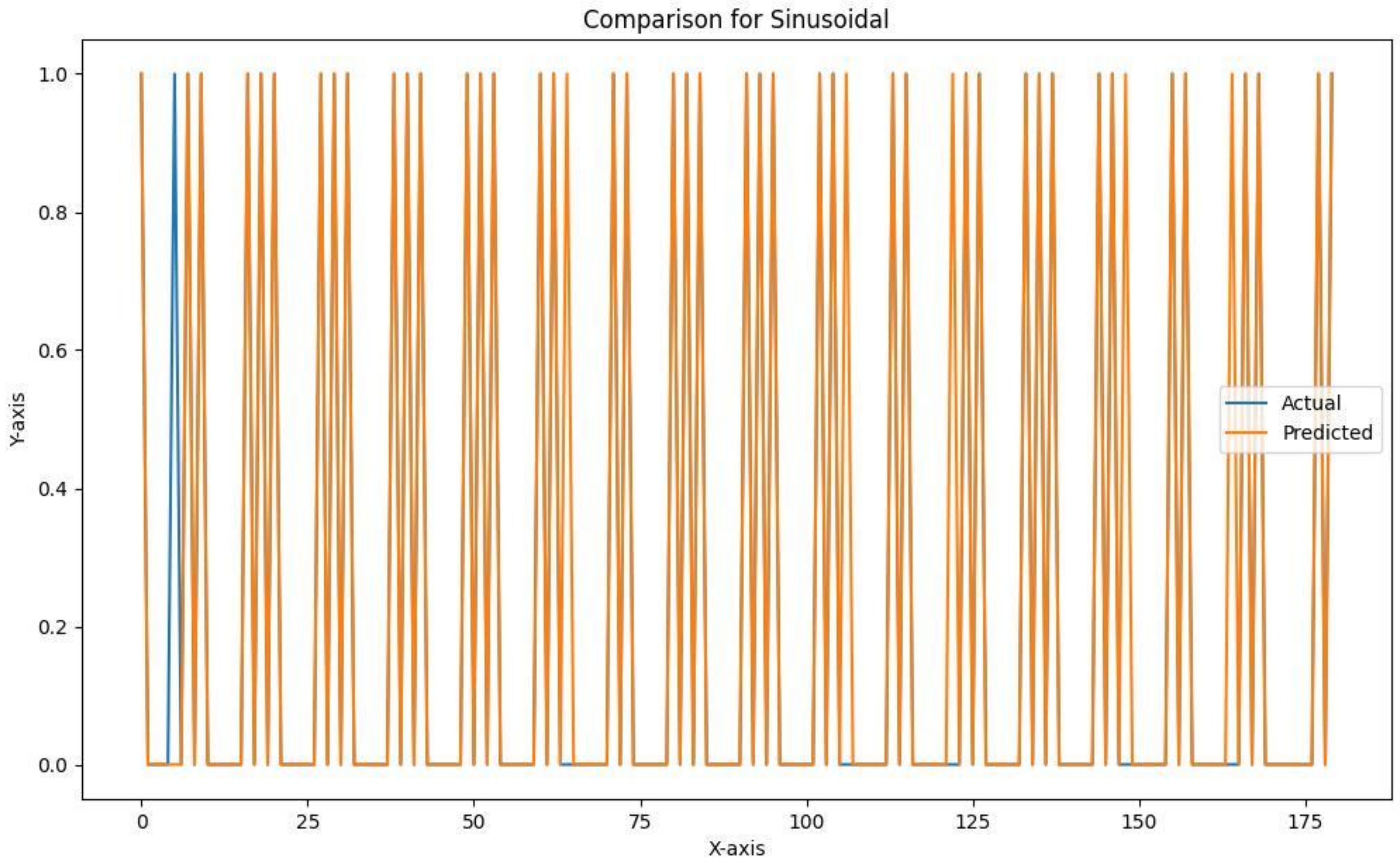
Results

- For periodic Invocation
  - Test-time (3+ hours)

	Without Prewarming(in ms)	With Pre-warming(in ms)	improvement	Extra billing per invocation(average)	Mishits per invocation
Periodic (Average execution time)	95.23	6.21	15x	6.25 ms	0
Periodic (Median execution time)	90.83	2.1	45x		

Results

- For sinusoidal invocation with thresholding
  - Test-time (3+ hours)



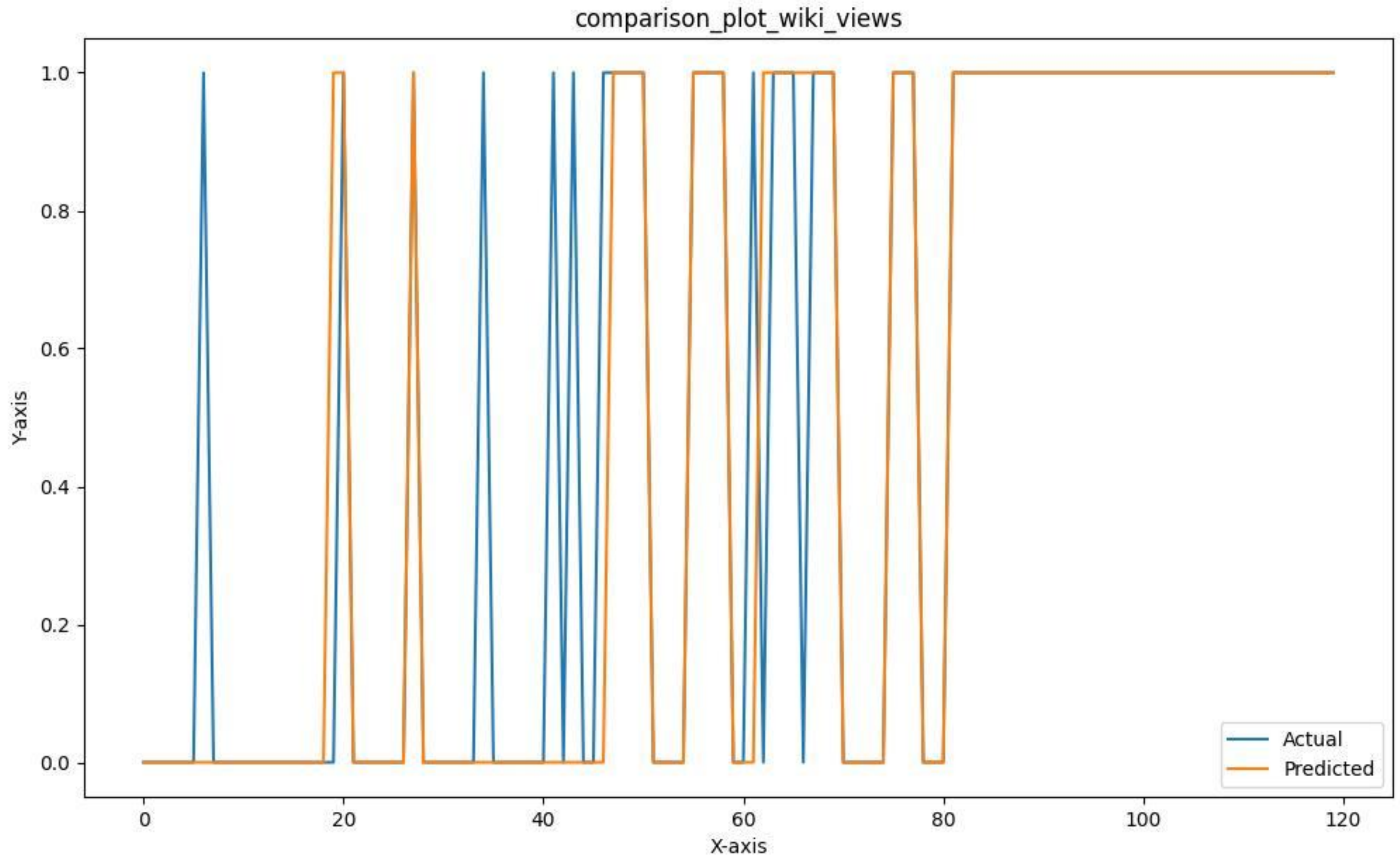
Results

- For sinusoidal invocation with thresholding
  - Test-time (3+ hours)

	Without Prewarming(in ms)	With Pre-warming(in ms)	improvement	Extra billing per invocation(average)	Mishits per invocation
Periodic (Average execution time)	65.59	8	8x	6.36 ms	0.03 (6 in 180)
Periodic (Median execution time)	85.18	4.32	20x		

Results

- For wikipedia traffic data with thresholding
  - Test-time (3+ hours)



Results

- For wikipedia traffic data with thresholding
  - Test-time (2+ hours)

	Without Prewarming(in ms)	With Pre-warming(in ms)	improvement	Extra billing per invocation(average)	Mishits per invocation
Periodic (Average execution time)	12.12	6.69	2x	3.07 ms	0.06
Periodic (Median execution time)	1.55	1.56	-		

Conclusion & Future Scope

- Server prewarming via time-series forecasting gives a significant improvement in reducing the function execution time.
- Especially useful where the micro-service architecture uses function invocations in series i.e the output of one is used by the next until the user request is met
- A drawback of this solution is the extra billing per invocation which leads to higher costs

<u>Future Scope</u>

- Server - aware prewarming to reduce extra billing per invocation i.e even if the program anticipates an invocation it should send the prewarming request only if the function container is in cold state
    - Requires server side integration

Thank you !