How to debug slow Lambda response times

Yan Cui @theburningmonk
Developer Advocate, Lumigo
AWS Serverless Hero
Author of Production-Ready Serverless

lumigo

Amazon Found Every 100ms of Latency Cost them 1% in Sales

Yoav Einav
January 20, 2019

3 minutes read

10 years ago, Amazon found that every 100ms of latency cost them 1% in sales. Google found an extra .5 seconds in search page generation time dropped traffic by 20%. A broker could lose $4 million in revenues per millisecond if their electronic trading platform is 5 milliseconds behind the competition.

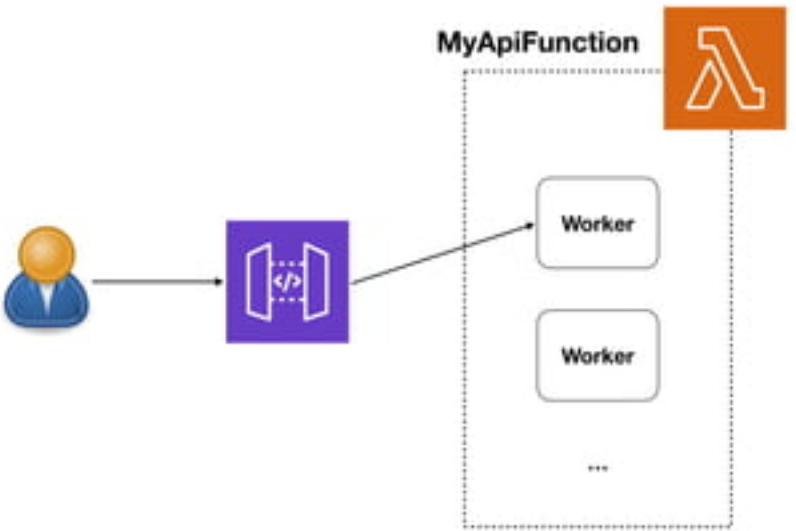
The expectations of today's NOW customers continue to grow and the amount of data generated and accessed is mind boggling. Bernard Marr, in his article in Forbes describes how 2.5 quintillion bytes of data are generated every day and that over the last two years alone 90 percent of the data in the world was generated.

It is clear, that the need for speed and scale are escalating and enterprises need to understand how they can support current and future applications to remain competitive from all aspects: optimized operations, regulation adherence and enhanced customer experience.

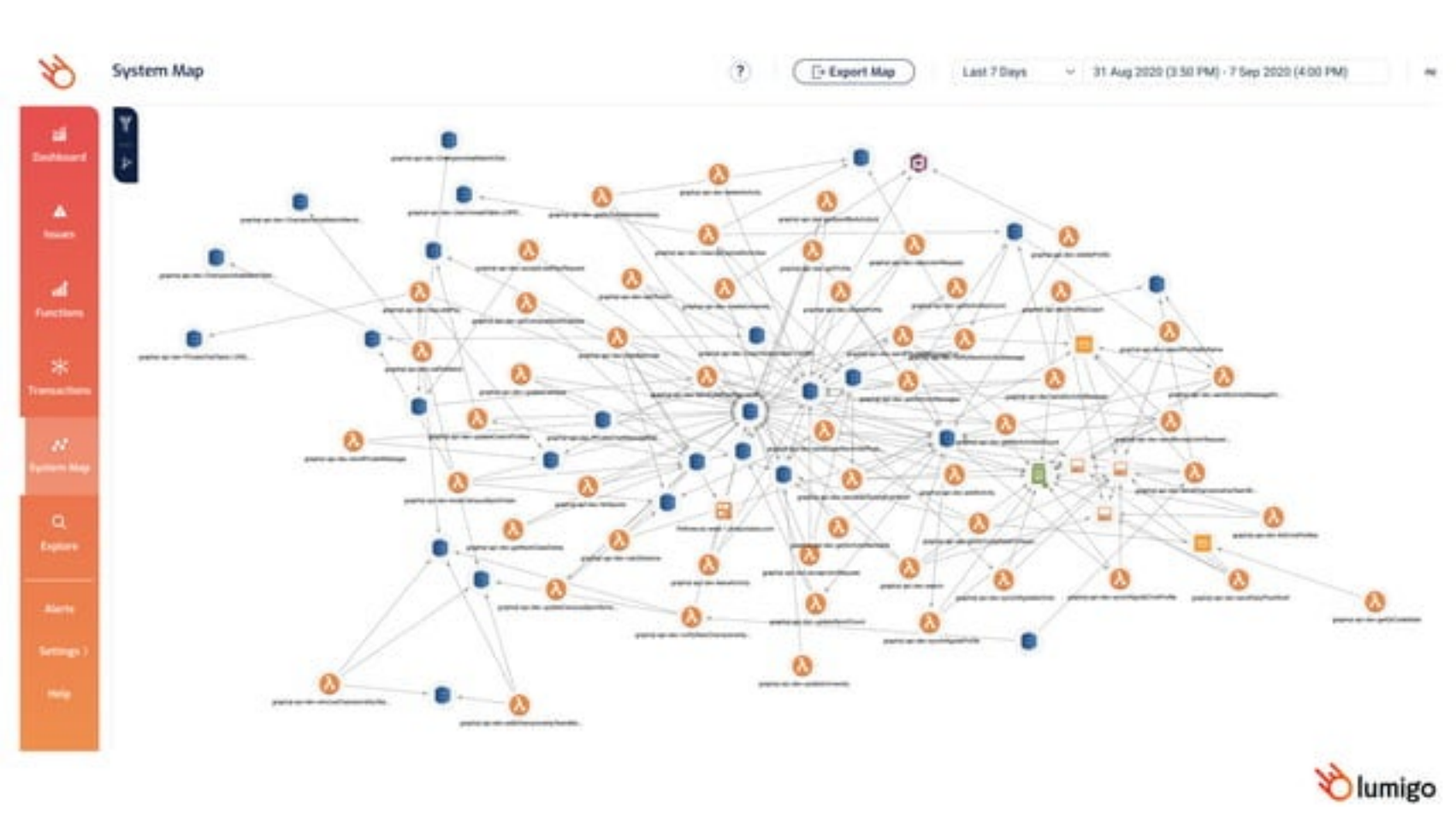
So, we've decided to put together some of the latest statistics discussing not just the cost of

Lambda autoscales by traffic

multi-AZ by default



overloaded servers are a thing of the past


System Map
Export Map
Last 7 Days
31 Aug 2020 (3:50 PM) - 7 Sep 2020 (4:00 PM)
Dashboard
Issues
Functions
Transactions
System Map
Explore
Alerts
Settings
Help
lumigo

observation

majority of performance problems originates from a function's integration points

macro

how well is this service performing in general?

micro

why did this transaction perform poorly?

macro

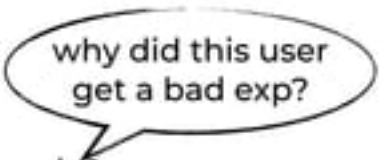
how well is this service performing in general?

micro

why did this transaction perform poorly?

macro

how well is this service performing in general?

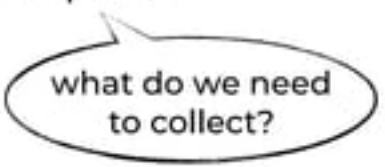


micro

why did this transaction perform poorly?

In control theory, observability is a measure of how well **internal states** of a system can be inferred from knowledge of its external outputs.

In control theory, observability is a measure of how well **internal states** of a system can be inferred from knowledge of its external outputs.



Yan Cui
@theburningmonk
http://theburningmonk.com

AWS user since 2009

Developer Advocate @ lumigo

Yan Cui

@theburningmonk

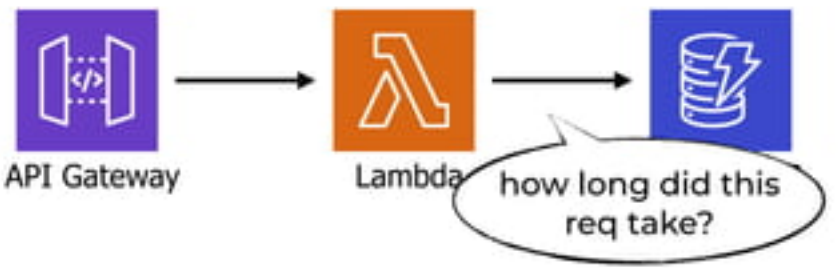
http://theburningmonk.com

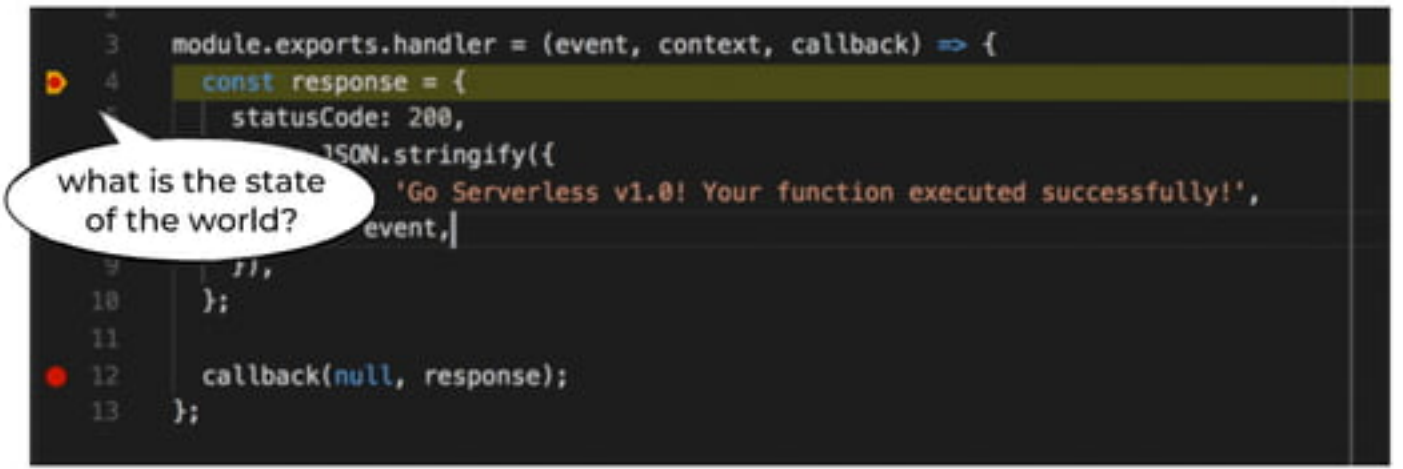
Independent Consultant

training

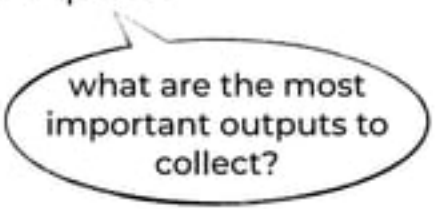
advise

delivery




module.exports.handler = (event, context, callback) => {
const response = {
statusCode: 200,
'Go Serverless v1.0! Your function executed successfully!',
event,
};
callback(null, response);
};
what is the state of the world?

In control theory, observability is a measure of how well **internal states** of a system can be inferred from knowledge of its external outputs.

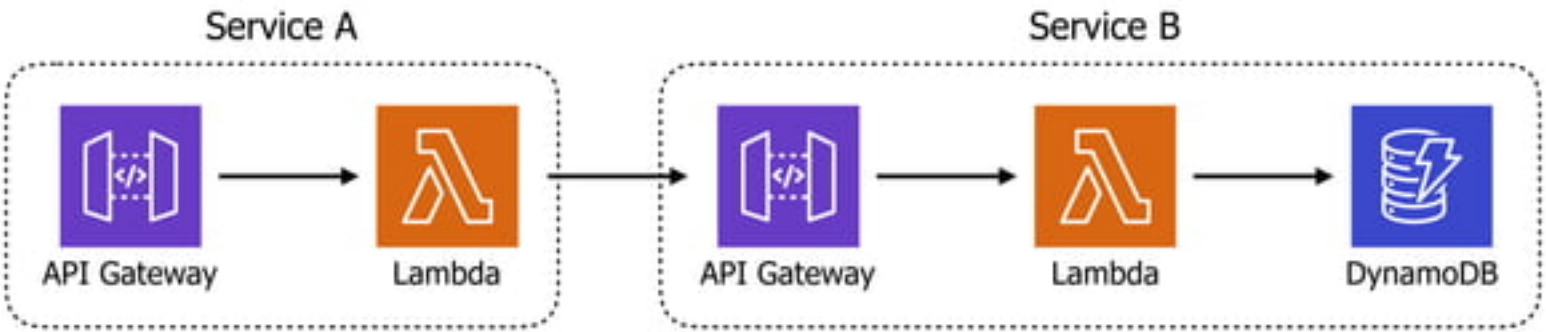


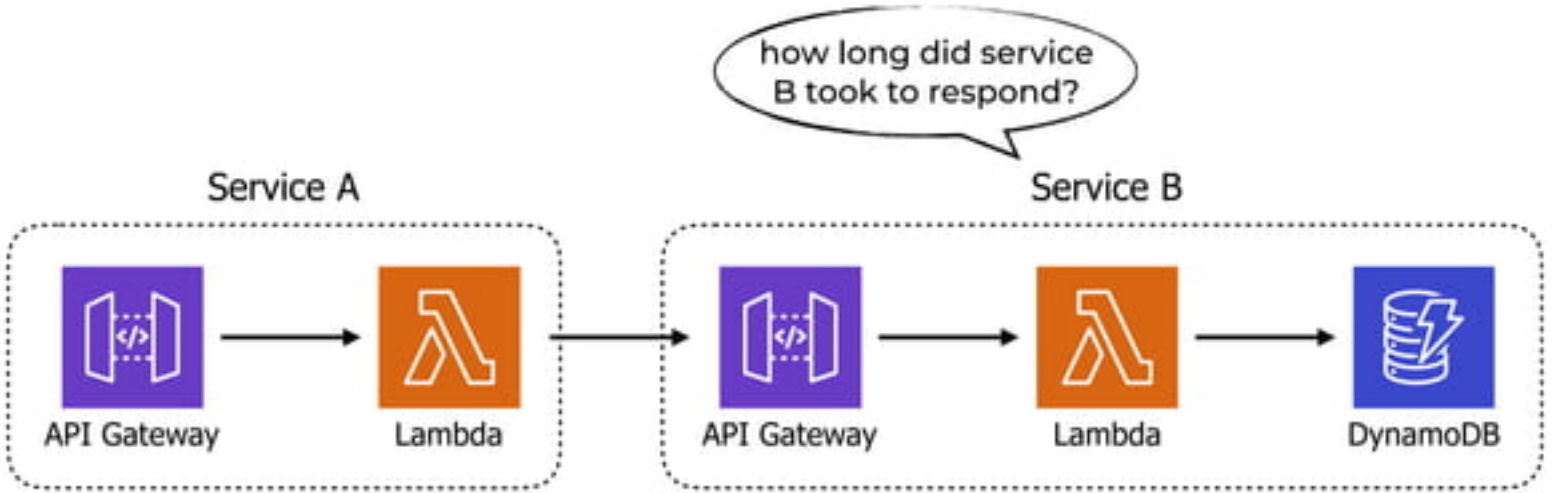
macro

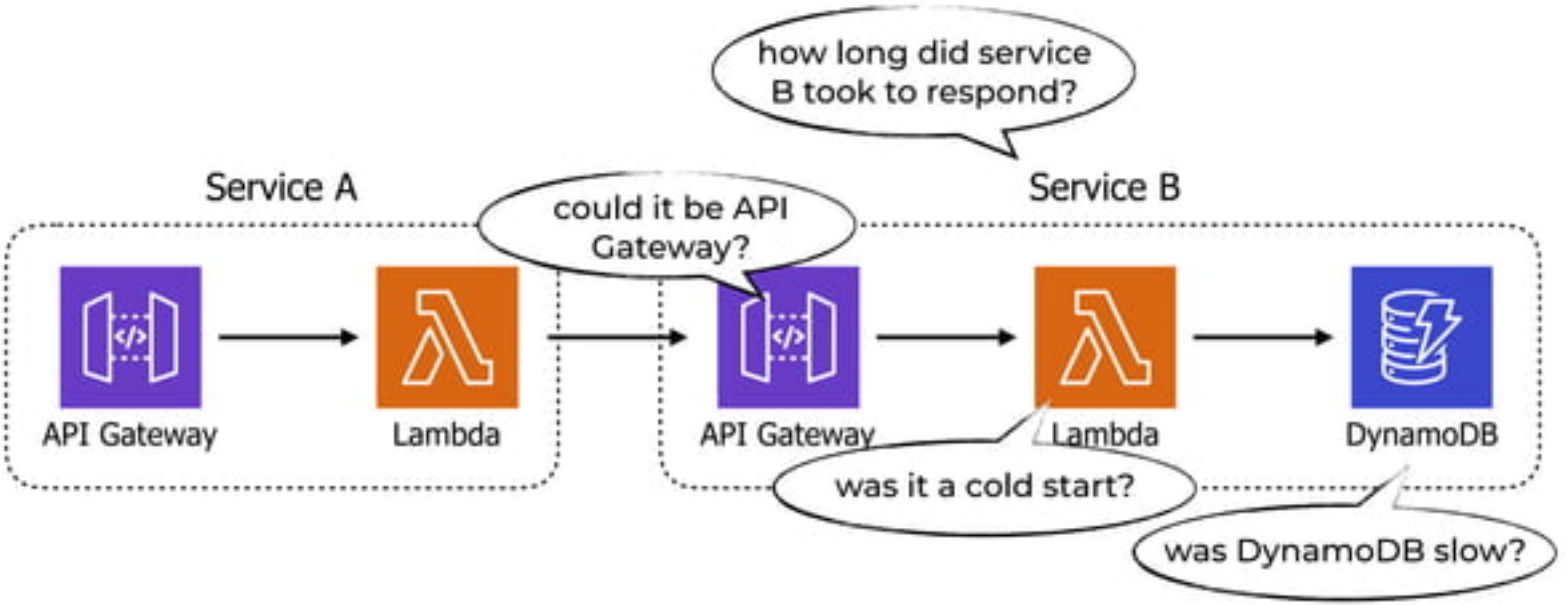
how well is this service performing in general?

micro

why did this transaction perform poorly?


Service A
Service B
API Gateway
Lambda
API Gateway
Lambda
DynamoDB


how long did service B took to respond?
Service A
Service B
API Gateway
Lambda
API Gateway
Lambda
DynamoDB


how long did service B took to respond?
Service A
could it be API Gateway?
Service B
API Gateway
Lambda
API Gateway
Lambda
DynamoDB
was it a cold start?
was DynamoDB slow?

API Gateway

IntegrationLatency	The time between when API Gateway relays a request to the backend and when it receives a response from the backend. Unit: Millisecond
Latency	The time between when API Gateway receives a request from a client and when it returns a response to the client. The latency includes the integration latency and other API Gateway overhead. Unit: Millisecond

Using performance metrics

Performance metrics provide performance details about a single invocation. For example, the `Duration` metric indicates the amount of time in milliseconds that your function spends processing an event. To get a sense of how fast your function processes events, view these metrics with the `Average` or `Max` statistic.

Performance metrics

- `Duration` – The amount of time that your function code spends processing an event. For the first event processed by an instance of your function, this includes initialization time. The billed duration for an invocation is the value of `Duration` rounded up to the nearest 100 milliseconds.
- `IteratorAge` – For event source mappings that read from streams, the age of the last record in the event. The age is the amount of time between when the stream receives the record and when the event source mapping sends the event to the function.

`Duration` also supports percentile statistics. Use percentiles to exclude outlier values that skew average and maximum statistics. For example, the P95 statistic shows the maximum duration of 95 percent of executions, excluding the slowest 5 percent.

Traces › Details

Timeline | Raw data

Method	Response	Duration	Age	ID
--	200	1.1 sec	1.1 min (2020-02-05 08:02:32 UTC)	1-5e3a7698-2addxmplecc5009a66f9ea1f

▸ Trace Map

Name	Res.	Duration	Status
▾ my-function AWS::Lambda			
my-function	200	1.1 sec	✓
▾ my-function AWS::Lambda::Function			
my-function	-	665 ms	✓
Initialization	-	328 ms	✓
Invocation	-	664 ms	✓
Overhead	-	0.5 ms	✓

Duration

lumigo

Lambda

Traces › Details

Timeline | Raw data

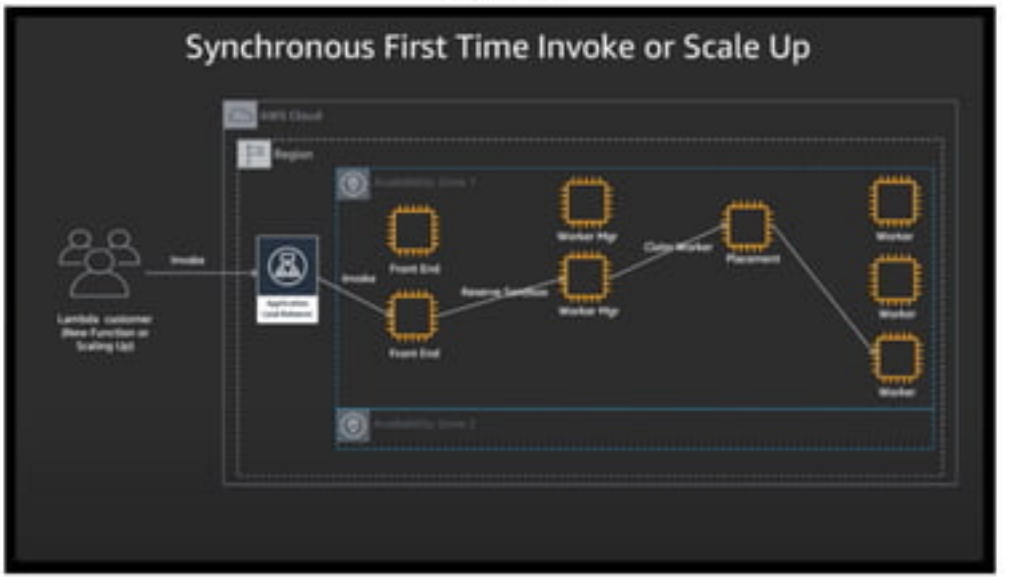
Method	Response	Duration	Age	ID
--	200	1.1 sec	1.1 min (2020-02-05 08:02:32 UTC)	1-5e3a7698-2addxmpleccc5009a66f9ea1f

▸ Trace Map

Name	Res.	Duration	Status
▾ my-function AWS::Lambda			
my-function	200	1.1 sec	
▾ my-function AWS::Lambda::Function			
my-function	-	665 ms	
Initialization	-	328 ms	
Invocation	-	664 ms	
Overhead	-	0.5 ms	

time to create and initialize the worker instance

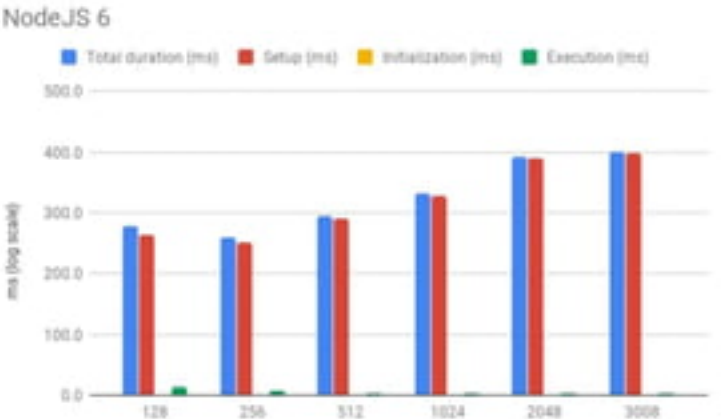
Lambda



bit.ly/2QXNVwc

lumigo

Lambda



bit.ly/2WL1uj0

Lambda

Traces › Details

Timeline | Raw data

Method	Response	Duration	Age	ID
--	200	1.1 sec	1.1 min (2020-02-05 08:02:32 UTC)	1-5e3a7698-2addxmpleccc5009a66f9ea1f

▸ Trace Map

Name	Res.	Duration	Status
▾ my-function AWS::Lambda			
my-function	200	1.1 sec	
▾ my-function AWS::Lambda::Function			
my-function	-	665 ms	
Initialization	-	328 ms	
Invocation	-	664 ms	
Overhead	-	0.5 ms	

time to create and initialize the worker instance

for API functions, use API Gateway's **IntegrationLatency**
as a proxy for "total response time from Lambda"

DynamoDB

DynamoDB Metrics

Note

Amazon CloudWatch aggregates the following DynamoDB metrics at one-minute intervals:

- `ConditionalCheckFailedRequests`
- `ConsumedReadCapacityUnits`
- `ConsumedWriteCapacityUnits`
- `ReadThrottleEvents`
- `ReturnedBytes`
- `ReturnedItemCount`
- `ReturnedRecordsCount`
- `SuccessfulRequestLatency`
- `SystemErrors`
- `TimeToLiveDeletedItemCount`
- `ThrottledRequests`
- `TransactionConflict`
- `UserErrors`
- `WriteThrottleEvents`

For all other DynamoDB metrics, the aggregation granularity is five minutes.

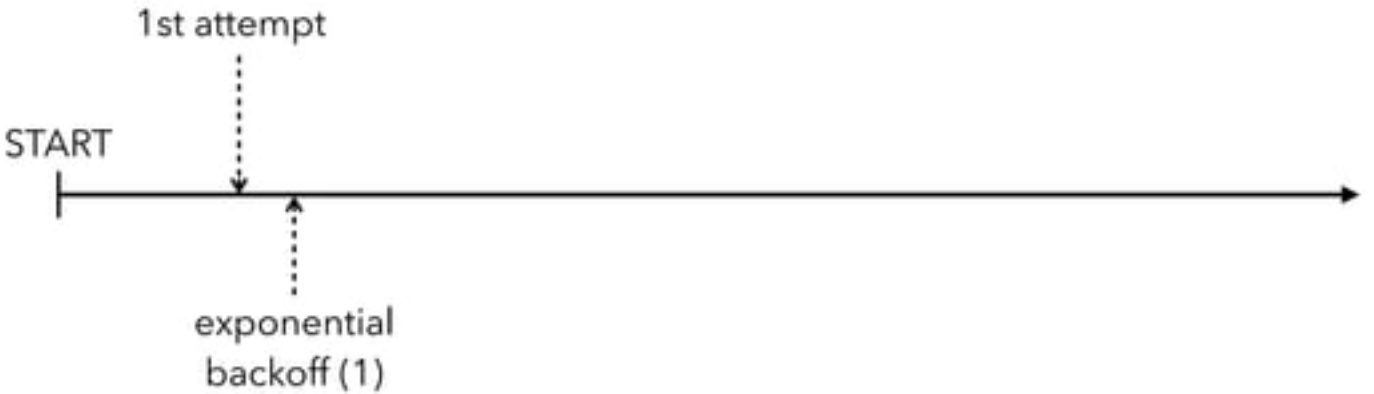
DynamoDB

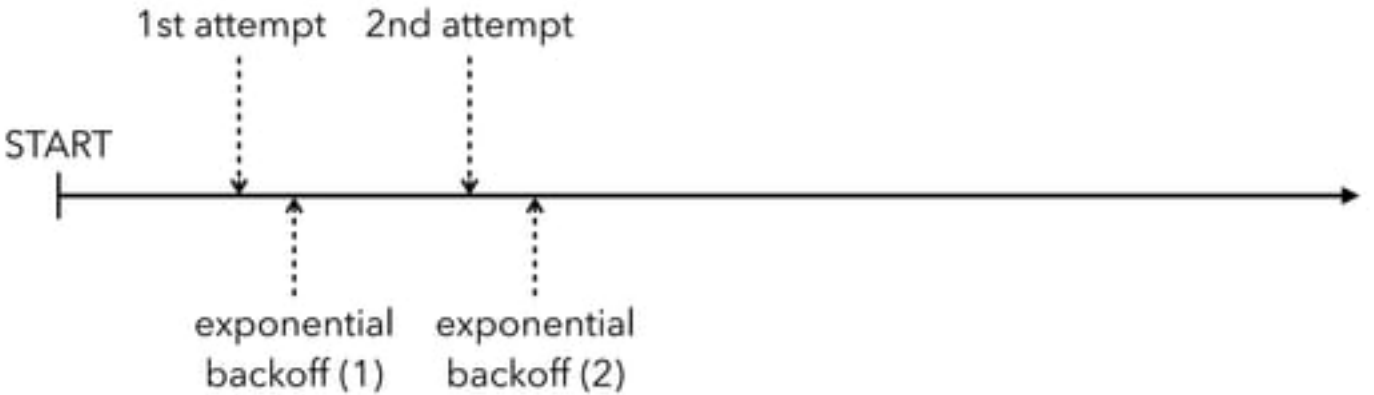
DynamoDB Metrics

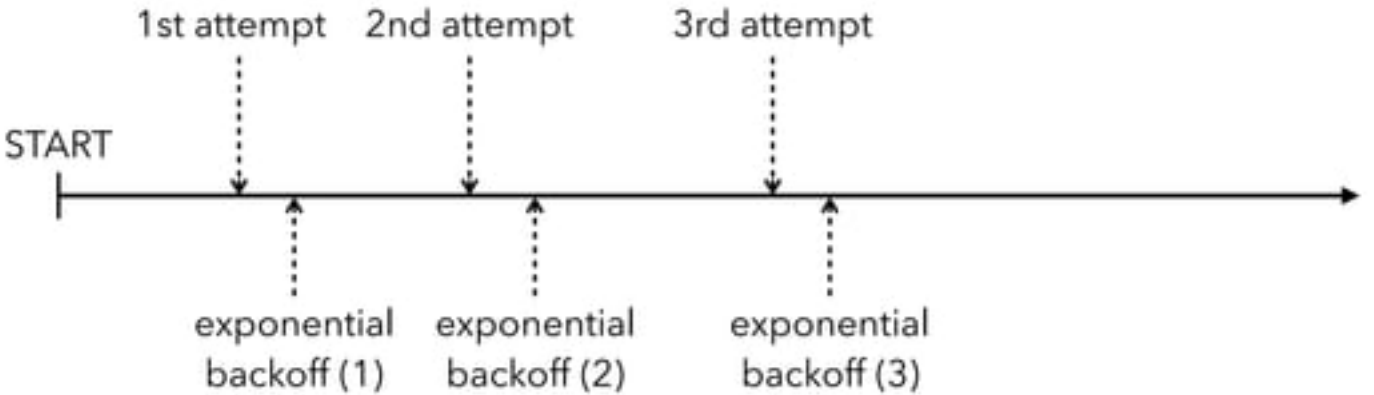
- ConditionalCheckFailedRequests
- ConsumedReadCapacityUnits
- ConsumedWriteCapacityUnits
- ReadThrottleEvents
- ReturnedBytes
- ReturnedItemCount
- ReturnedRecordsCount
- SuccessfulRequestLatency
- SystemErrors
- TimeToLiveDeletedItemCount
- ThrottledRequests
- TransactionConflict
- UserErrors
- WriteThrottleEvents

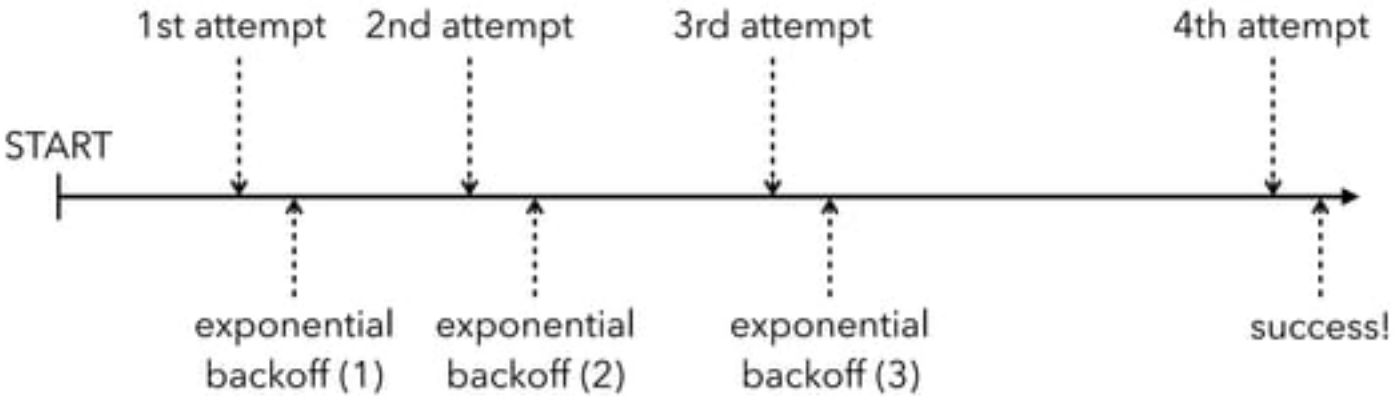
SuccessfulRequestLatency

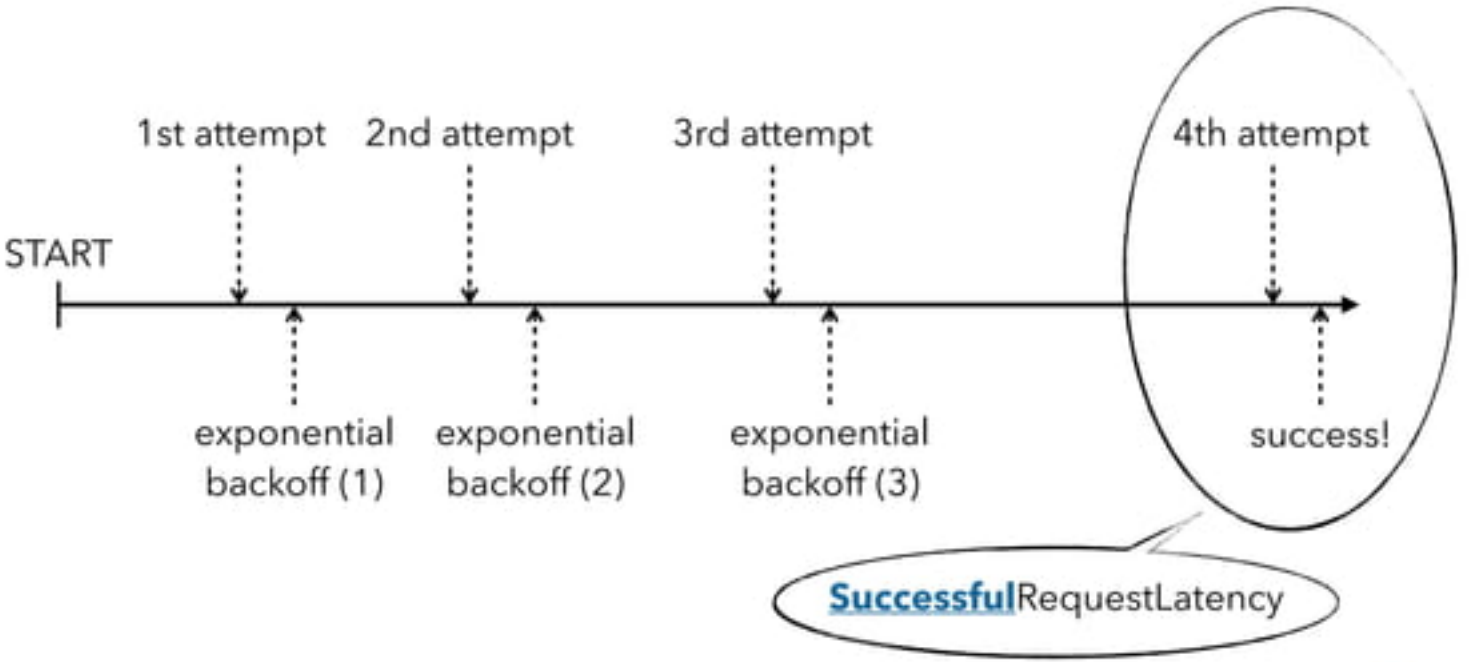
"I'm facing this problem now with a lambda that usually takes 25 ms but once a week or so takes > 6000 ms and times out. The lambda's first step is to load a DynamoDB table that only has 8 items. I'm at a loss to understand how such a simple query could take so long."




1st attempt
2nd attempt
START
exponential backoff (1)
exponential backoff (2)


1st attempt
2nd attempt
3rd attempt
START
exponential backoff (1)
exponential backoff (2)
exponential backoff (3)


1st attempt
2nd attempt
3rd attempt
4th attempt
START
exponential backoff (1)
exponential backoff (2)
exponential backoff (3)
success!

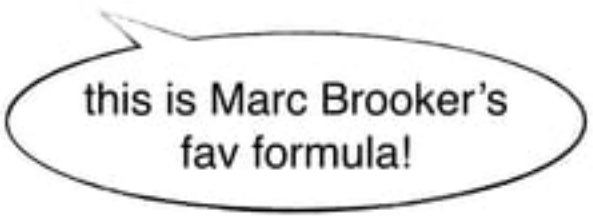

1st attempt
2nd attempt
3rd attempt
4th attempt
START
exponential backoff (1)
exponential backoff (2)
exponential backoff (3)
success!
SuccessfulRequestLatency

JavaScript AWS SDK

10 retries

Initial exponential backoff of 50ms

```
delay = Math.random() * (Math.pow(2, retryCount) * base)
```



JavaScript AWS SDK

10 retries

Initial exponential backoff of 50ms

delay = Math.random() * (Math.pow(2, retryCount) * base)

retry count	max delay
1	100
2	200
3	400
4	800
5	1600
6	3200
7	6400
8	12800
9	25600
10	51200

JavaScript AWS SDK

10 retries

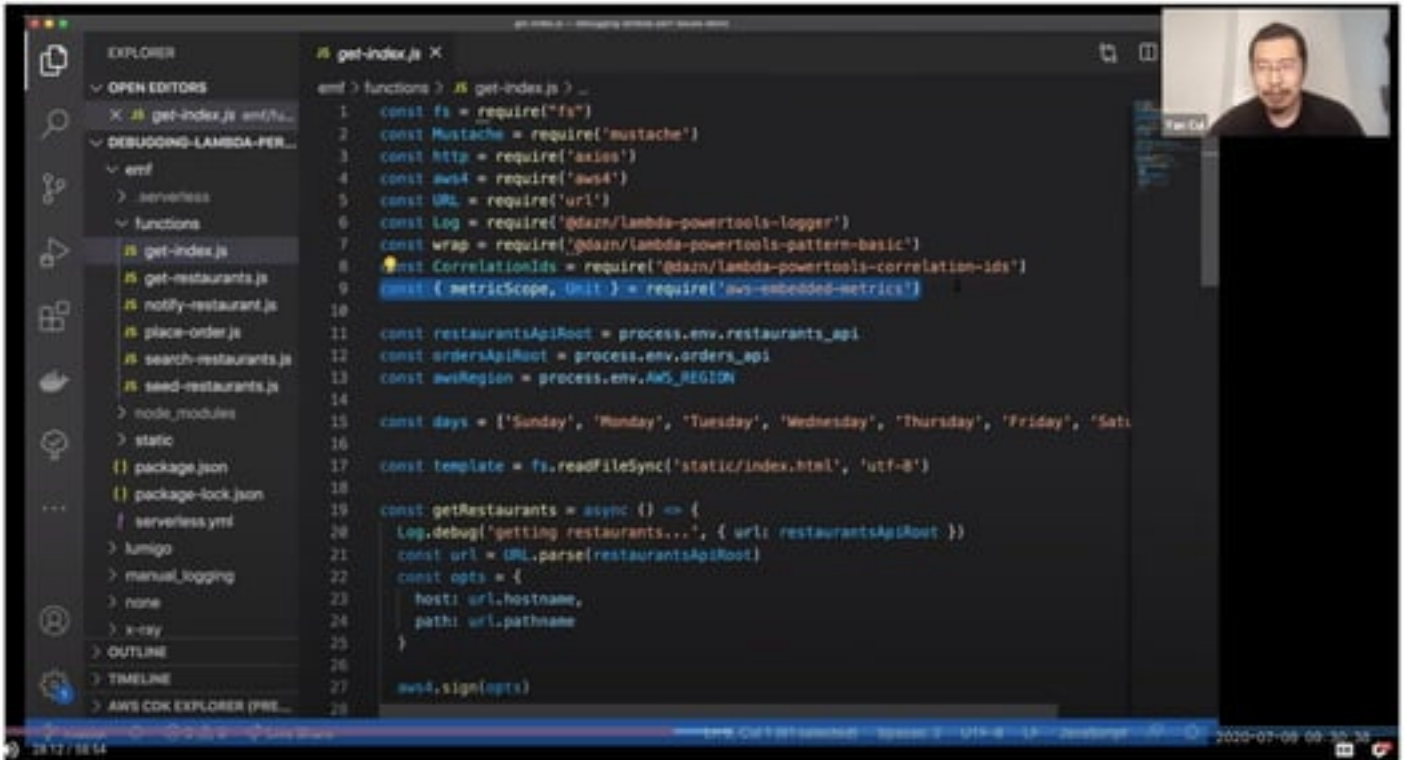
Initial exponential backoff of 50ms

delay = Math.random() * (Math.pow(2, retryCount) * base)

retry count	max delay
1	100
2	200
3	400
4	800
5	1600
6	3200
7	6400
8	12800
9	25600
10	51200

danger zone!

Record client-side latency metrics for IO operations



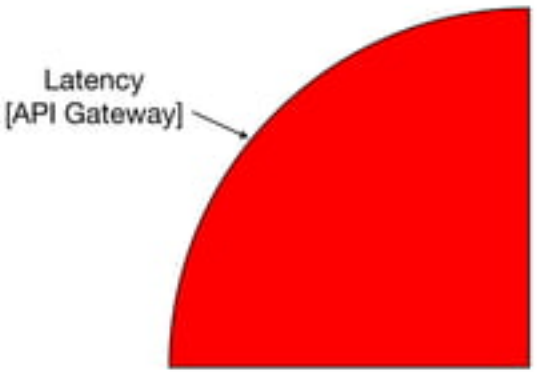
www.youtube.com/watch?v=adtCwnKApWI

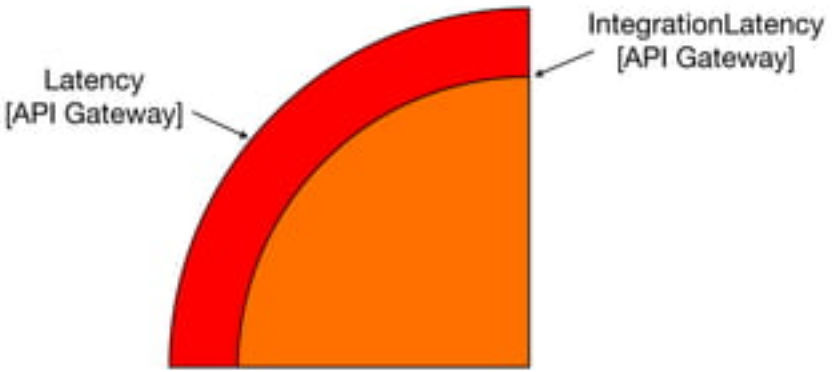
lumigo

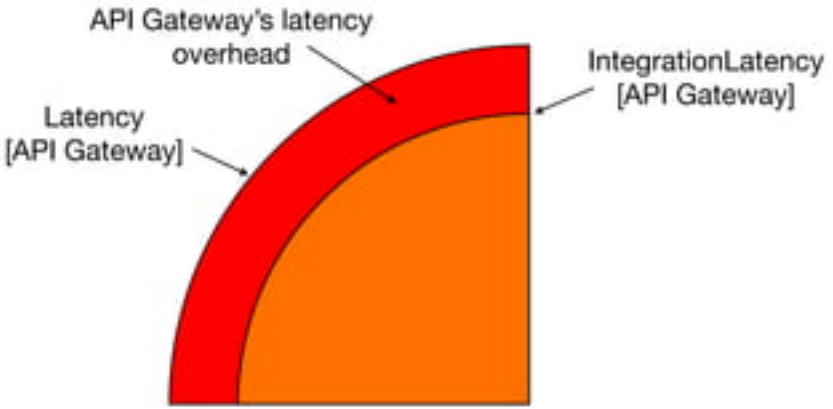
Embedded Metric Format (EMF)

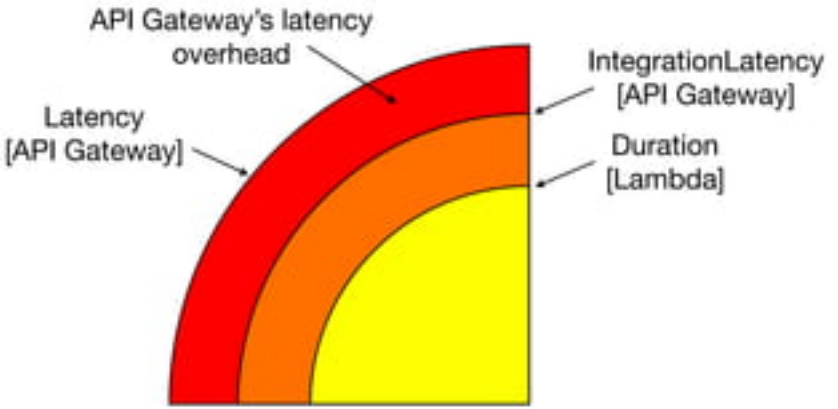
```
2020-07-09T14:39:05.616Z        bf6c45d4-38a8-4003-962d-0f9da54a8982    INFO
{
    "LogGroup": "debug-perf-issues-emf-demo-dev-get-index",
    "ServiceName": "debug-perf-issues-emf-demo-dev-get-index",
    "ServiceType": "AWS::Lambda::Function",
    "RequestId": "bf6c45d4-38a8-4003-962d-0f9da54a8982",
    "executionEnvironment": "AWS_Lambda_nodejs12.x",
    "memorySize": "1024",
    "functionVersion": "$LATEST",
    "logStreamId": "2020/07/09/[$LATEST]e8ba9df804464a2f8de196abe630ad5e",
    "_aws": {
        "Timestamp": 1594305545477,
        "CloudWatchMetrics": [
            {
                "Dimensions": [
                    [
                        "LogGroup",
                        "ServiceName",
                        "ServiceType"
                    ]
                ],
                "Metrics": [
                    {
                        "Name": "latency.HTTP.getRestaurants",
                        "Unit": "Milliseconds"
                    }
                ],
                "Namespace": "emf-demo"
            }
        ]
    },
    "latency.HTTP.getRestaurants": 137
}
```

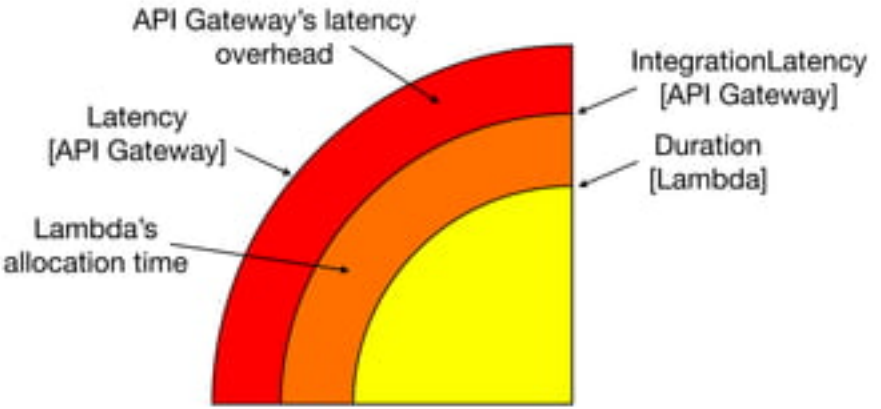
lumigo

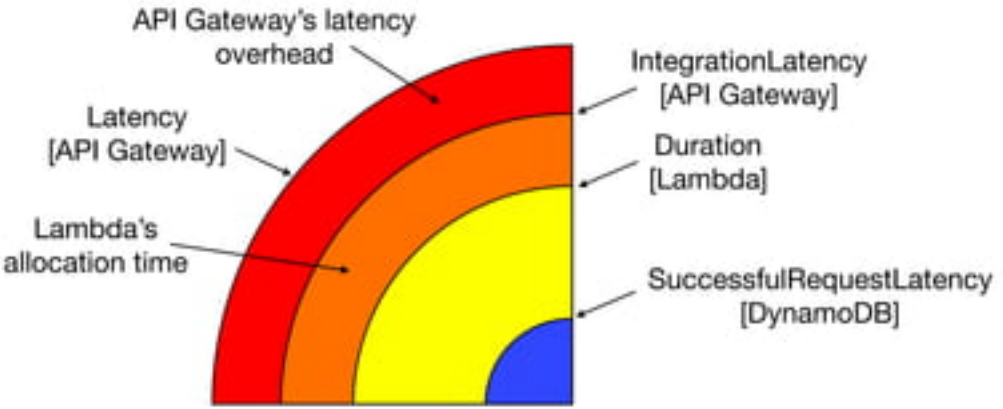


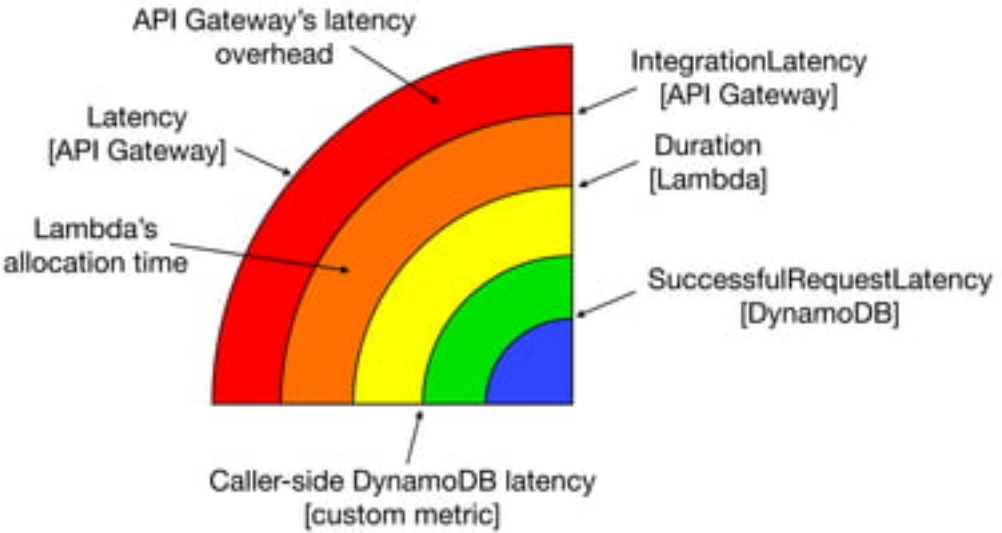


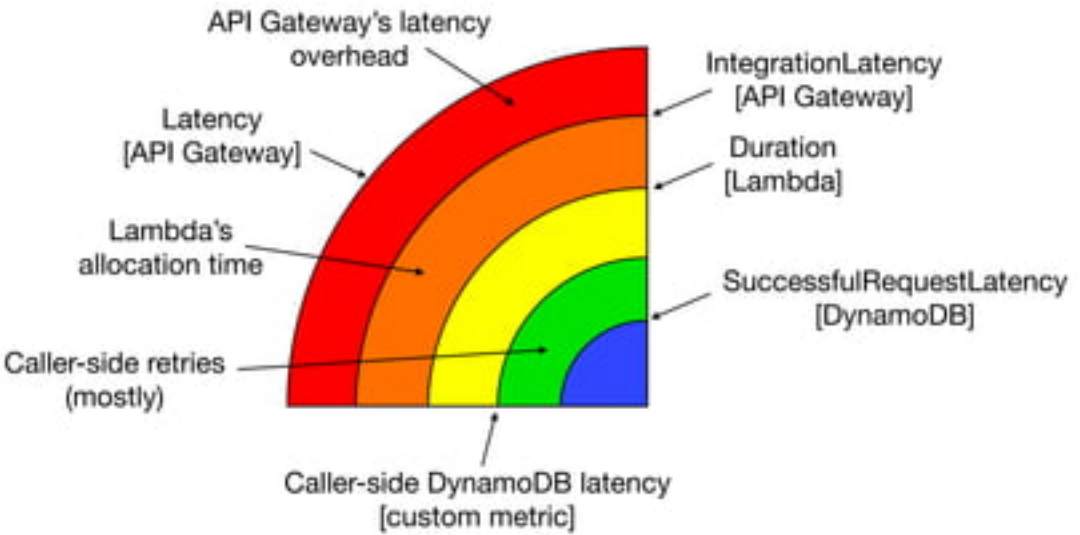

API Gateway's latency
overhead
IntegrationLatency
[API Gateway]
Latency
[API Gateway]

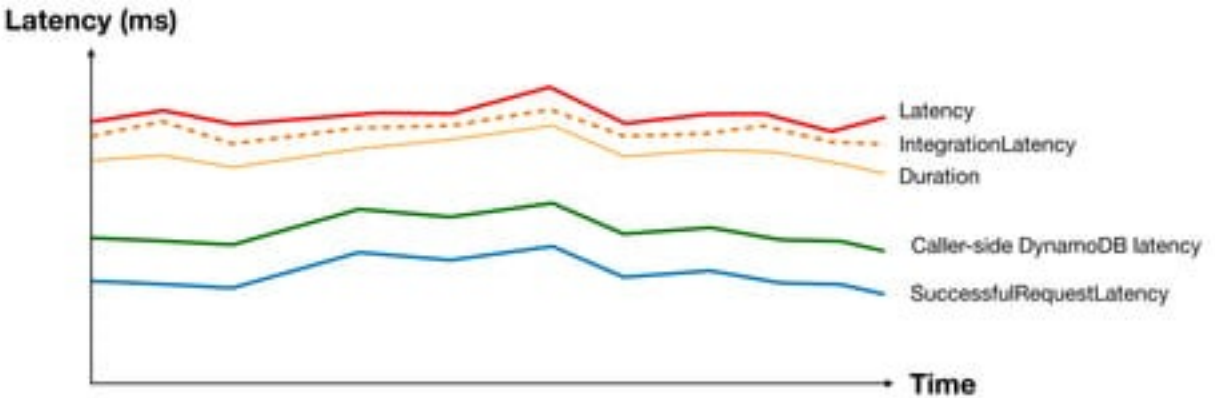

API Gateway's latency overhead
IntegrationLatency [API Gateway]
Latency [API Gateway]
Duration [Lambda]

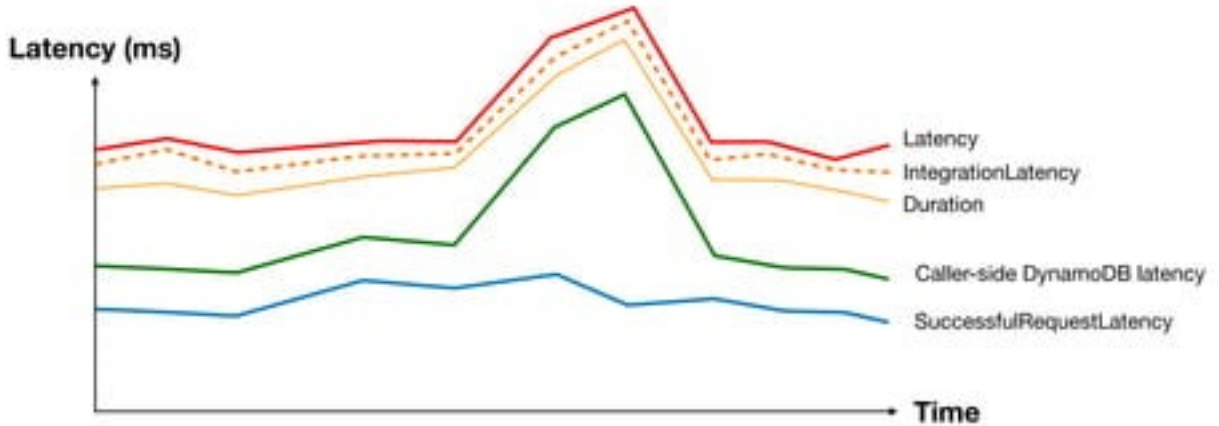

API Gateway's latency overhead
IntegrationLatency [API Gateway]
Latency [API Gateway]
Duration [Lambda]
Lambda's allocation time


API Gateway's latency overhead
IntegrationLatency [API Gateway]
Latency [API Gateway]
Duration [Lambda]
Lambda's allocation time
SuccessfulRequestLatency [DynamoDB]


API Gateway's latency overhead
IntegrationLatency [API Gateway]
Latency [API Gateway]
Duration [Lambda]
Lambda's allocation time
SuccessfulRequestLatency [DynamoDB]
Caller-side DynamoDB latency [custom metric]


API Gateway's latency overhead
IntegrationLatency [API Gateway]
Latency [API Gateway]
Duration [Lambda]
Lambda's allocation time
SuccessfulRequestLatency [DynamoDB]
Caller-side retries (mostly)
Caller-side DynamoDB latency [custom metric]


Latency (ms)
Latency
IntegrationLatency
Duration
Caller-side DynamoDB latency
SuccessfulRequestLatency
Time


Latency (ms)
Latency
IntegrationLatency
Duration
Caller-side DynamoDB latency
SuccessfulRequestLatency
Time

macro

how well is this service performing in general?

micro

why did this transaction perform poorly?

X-Ray

X-Ray

```
const XRay = require('aws-xray-sdk-core')
const AWS = AWSXRay.captureAWS(require('aws-sdk'))
AWSXRay.captureHTTPsGlobal(require('https'))
```

X-Ray

```
const XRay = require('aws-xray-sdk-core')
const AWS = AWSXRay.captureAWS(require('aws-sdk'))
AWSXRay.captureHTTPsGlobal(require('https'))
```

can be encapsulated
in custom modules

X-Ray

✓ doesn't add latency

X-Ray

- ✓ doesn't add latency
- ✓ can see "system" overhead (e.g. allocation time)

X-Ray

- ✓ doesn't add latency
- ✓ can see "system" overhead (e.g. allocation time)
- ✓ built-in sampling

X-Ray

- ✓ doesn't add latency
- ✓ can see "system" overhead (e.g. allocation time)
- ✓ built-in sampling
- ✗ X-Ray SDK adds significant overhead

X-Ray

- ✓ doesn't add latency
- ✓ can see "system" overhead (e.g. allocation time)
- ✓ built-in sampling
- ✗ X-Ray SDK adds significant overhead
- ✗ doesn't trace TCP traffic (RDS/Elasticache)

X-Ray

- ✓ doesn't add latency
- ✓ can see "system" overhead (e.g. allocation time)
- ✓ built-in sampling
- ✗ X-Ray SDK adds significant overhead
- ✗ doesn't trace TCP traffic (RDS/Elasticache)
- ✗ poor support for saync event sources (only SNS)

X-Ray

- ✓ doesn't add latency
- ✓ can see "system" overhead (e.g. allocation time)
- ✓ built-in sampling
- ✗ X-Ray SDK adds significant overhead
- ✗ doesn't trace TCP traffic (RDS/Elasticache)
- ✗ poor support for saync event sources (only SNS)
- ✗ doesn't capture request & response data

X-Ray

- ✓ doesn't add latency
- ✓ can see "system" overhead (e.g. allocation time)
- ✓ built-in sampling
- ✗ X-Ray SDK adds significant overhead
- ✗ doesn't trace TCP traffic (RDS/Elasticache)
- ✗ poor support for saync event sources (only SNS)
- ✗ doesn't capture request & response data
- ✗ logs and traces are separate

X-Ray

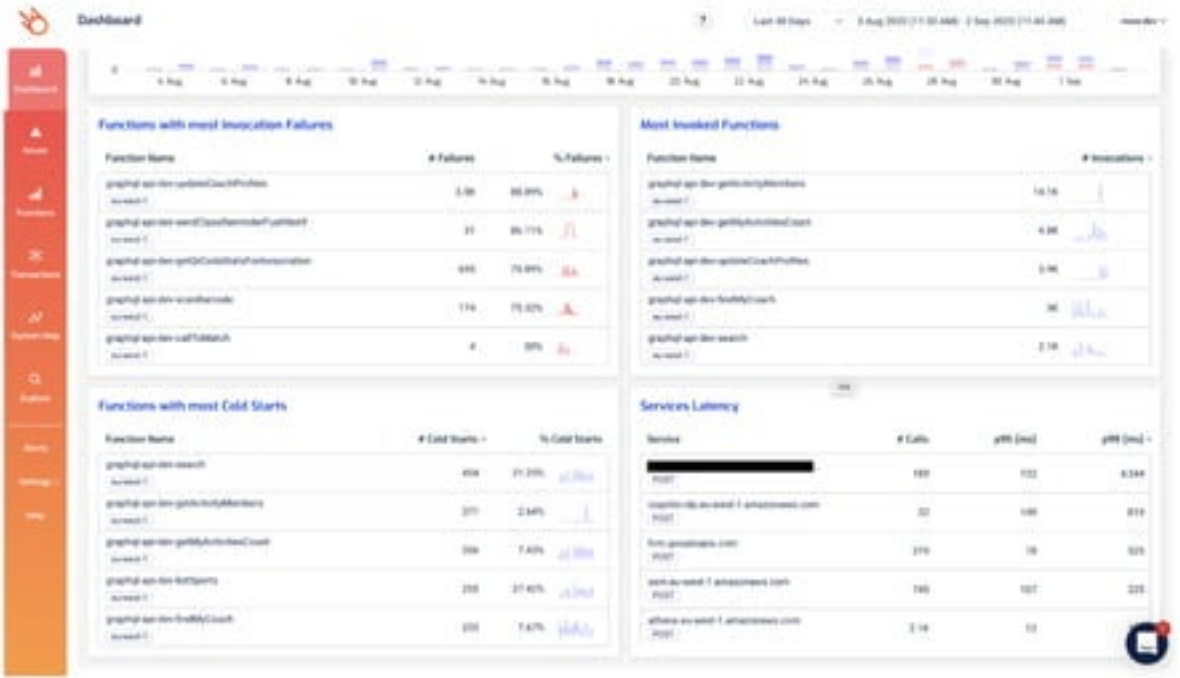
- ✓ doesn't add latency
- ✓ can see "system" overhead (e.g. allocation time)
- ✓ built-in sampling
- ✗ X-Ray SDK adds significant overhead
- ✗ doesn't trace TCP traffic (RDS/Elasticache)
- ✗ poor support for saync event sources (only SNS)
- ✗ doesn't capture request & response data
- ✗ logs and traces are separate
- ✗ difficult to search

good enough for simple workloads

when you outgrow X-Ray, look for a 3rd-party tool


lumigo

lumigo



lumigo

lumigo

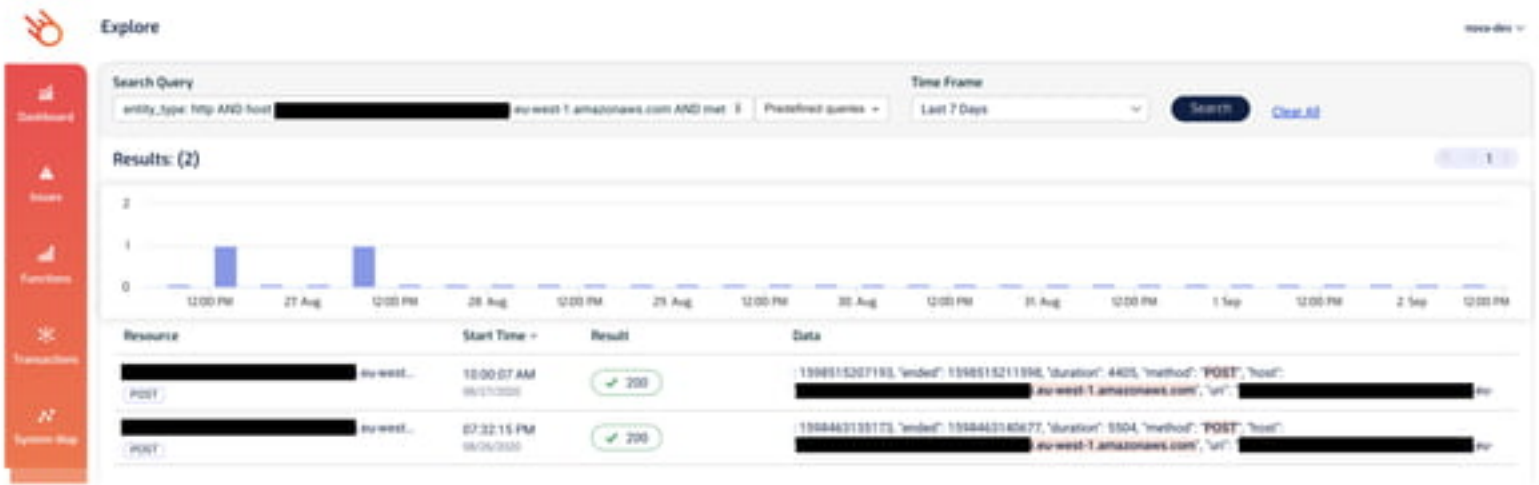
Services Latency

Service	# Calls	p95 (ms)	p99 (ms)
[redacted] POST	189	132	4,344
cognito-idp.eu-west-1.amazonaws.com POST	32	149	819
fcm.googleapis.com POST	319	78	525
ssm.eu-west-1.amazonaws.com POST	745	107	225
athena.eu-west-1.amazonaws.com POST	2.1K	12	

lumigo

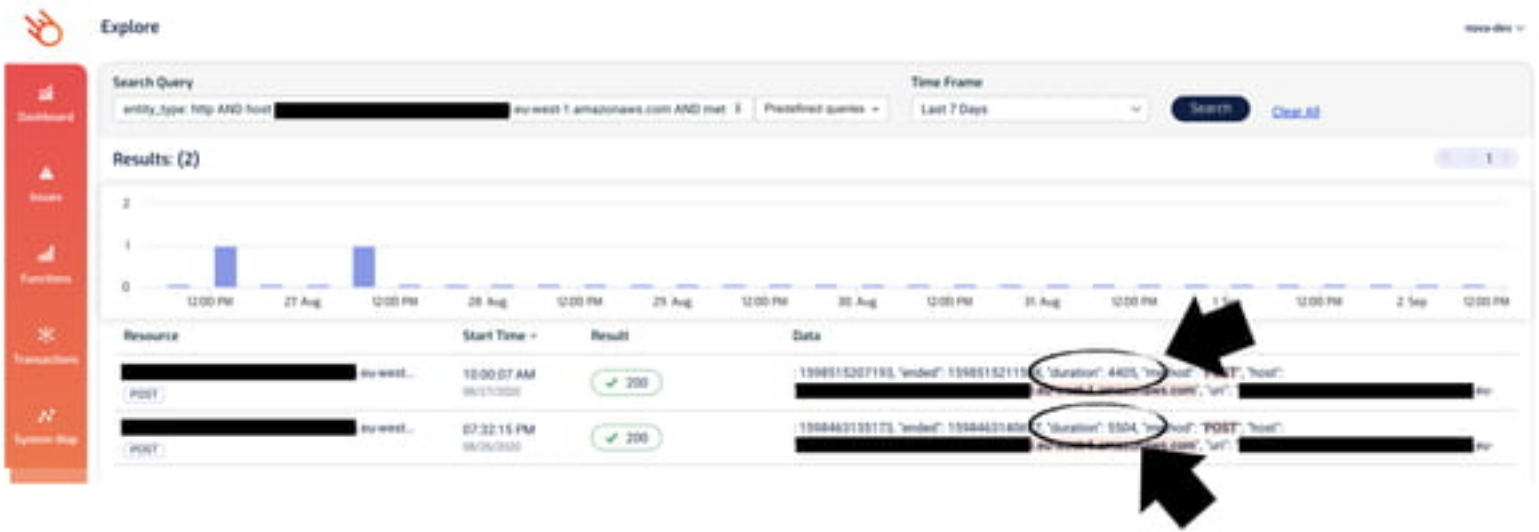
lumigo

lumigo

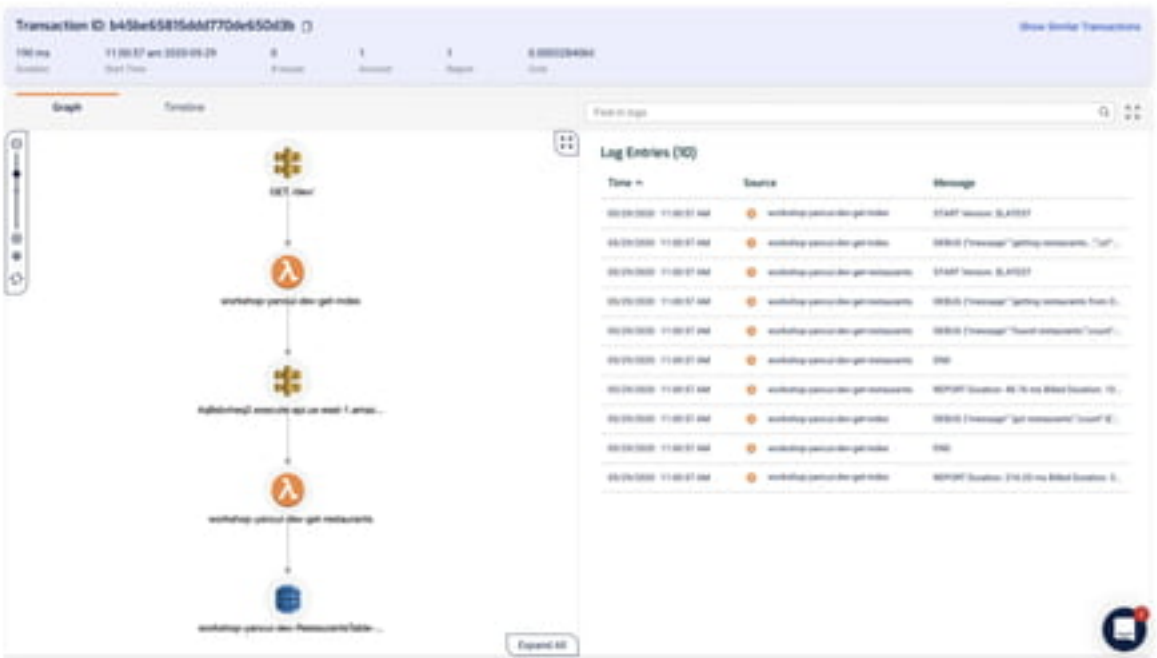


lumigo

lumigo



lumigo


lumigo
Transaction ID: b45be65815ddd770de650d3b
Show Similar Transactions
Graph
Timeline
Log Entries (10)
Time
Source
Message
Expand All

lumigo

lumigo

lumigo

lumigo

answer both macro and micro level questions
in just **a few clicks!**

Functions with most Cold Starts

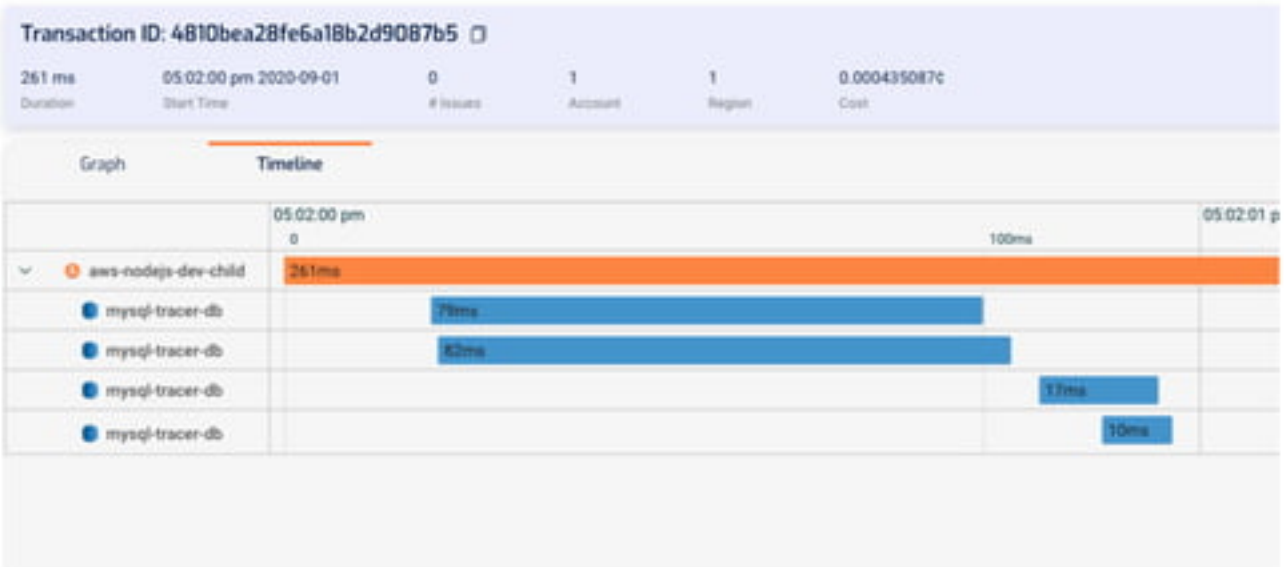
Function Name	# Cold Starts	% Cold Starts
graphql-api-dev-search eu-west-1	454	21.25%
graphql-api-dev-getActivityMembers eu-west-1	371	2.64%
graphql-api-dev-getMyActivitiesCount eu-west-1	356	7.43%
graphql-api-dev-listSports eu-west-1	255	27.42%
graphql-api-dev-findMyCoach eu-west-1	233	7.67%

lumigo

Support async event sources such as Kinesis, DynamoDB streams and SNS

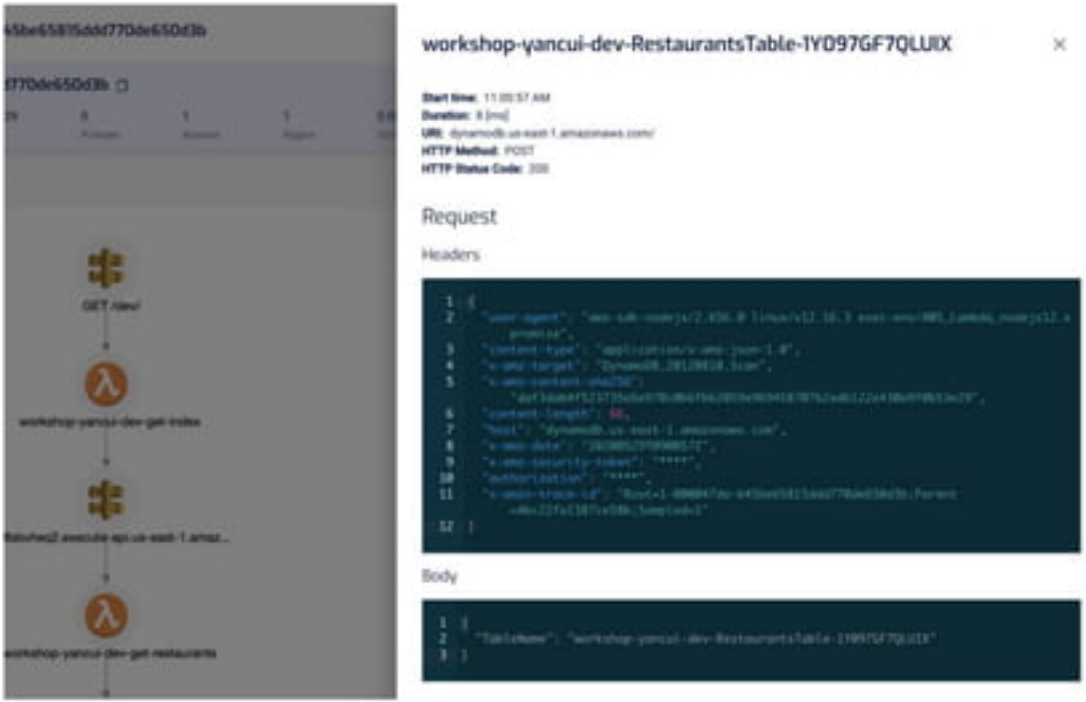
lumigo

Support TCP traffic - e.g. RDS and Elasticache



lumigo

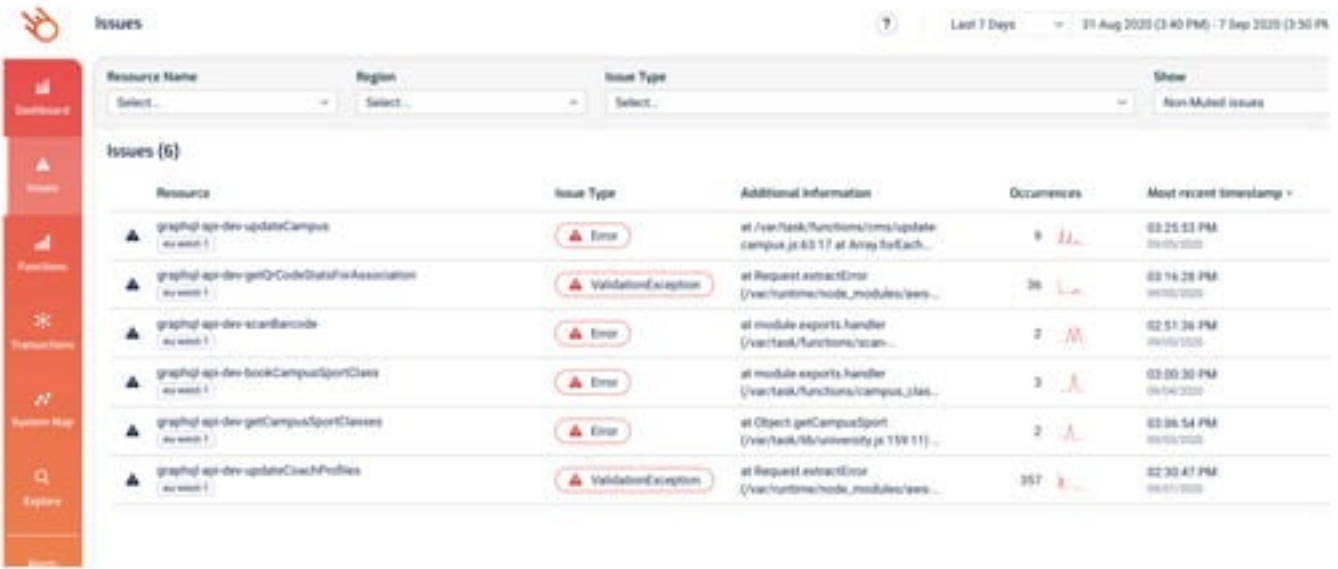
lumigo



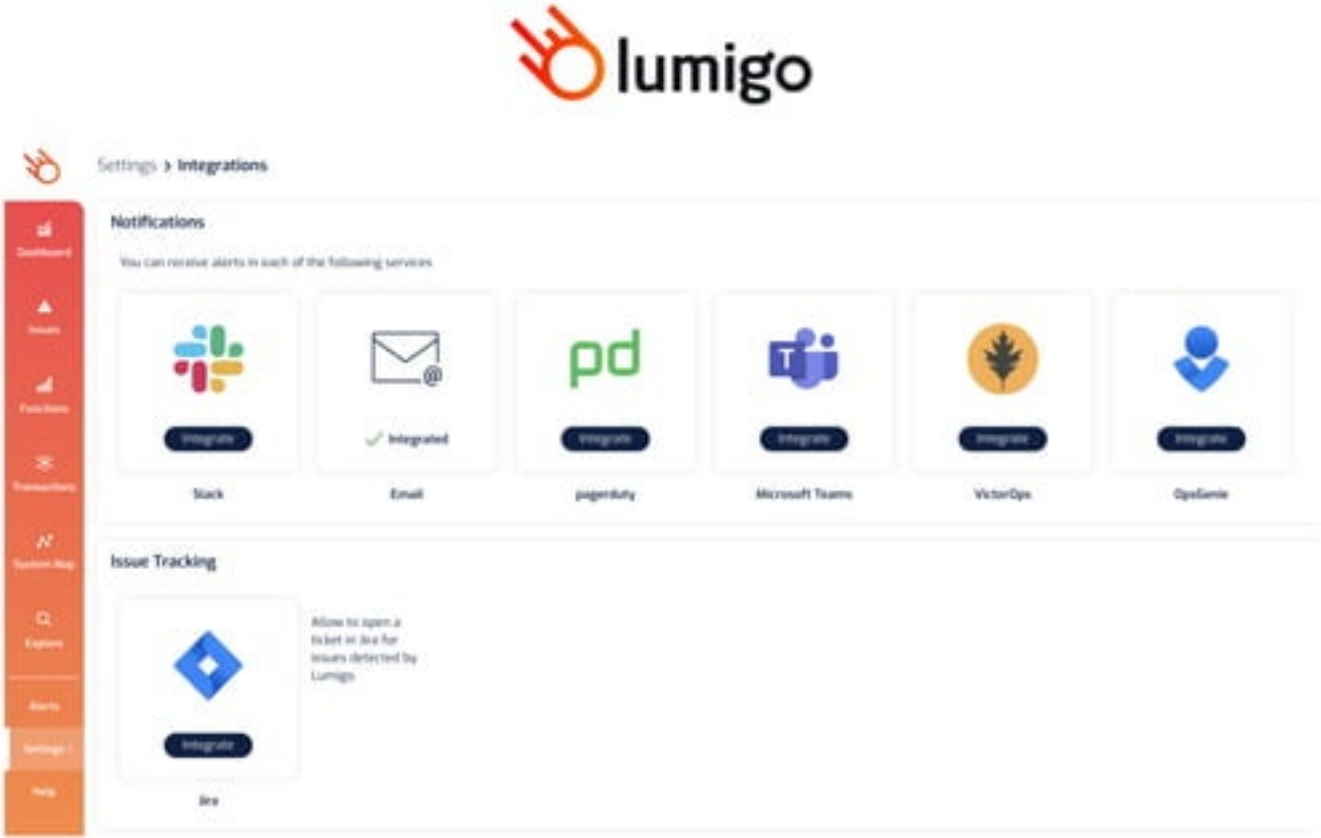
lumigo


mysql-tracer-db
5:02:00 PM
Start time
10 ms
Duration
mysql-tracer-db.ci1...
URI
200
HTTP Status Code
Request
Body
{ "query": "SELECT * FROM `Users` WHERE `favLang` = ?", "values": "[\"Java\"]" }
Response
Body
{ "rows": "[{\"id\":1,\"name\":\"Uri\",\"favLang\":\"Java\"}]" }

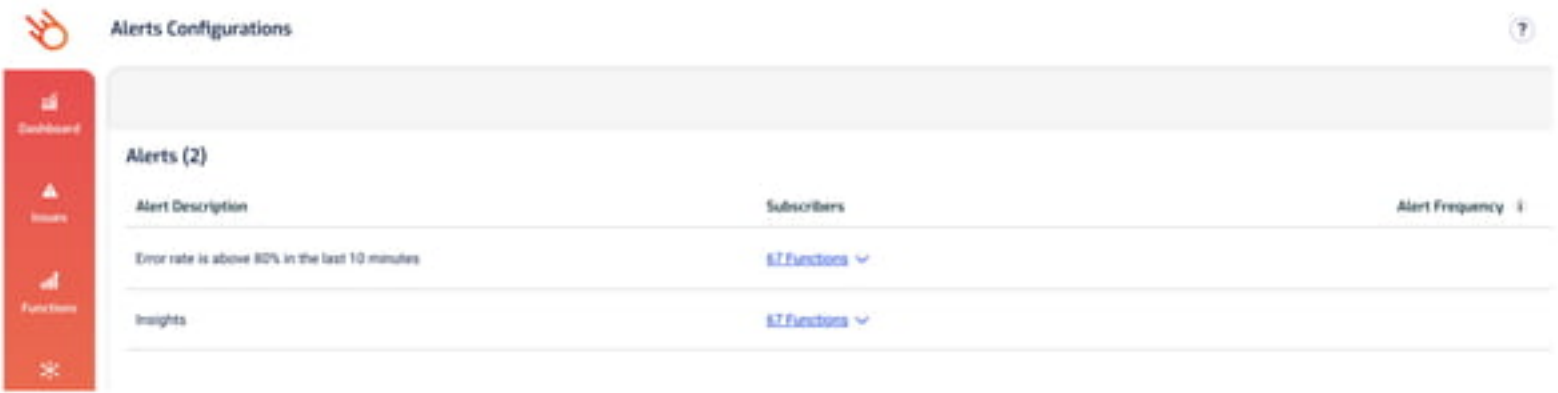
lumigo



lumigo


lumigo
Settings > Integrations
Notifications
You can receive alerts in each of the following services
Integrate
Integrated
Slack
Email
pagerduty
Microsoft Teams
VictorOps
OpsGenie
Issue Tracking
Allow to open a ticket in Jira for issues detected by Lumigo
Jira
lumigo

lumigo



lumigo

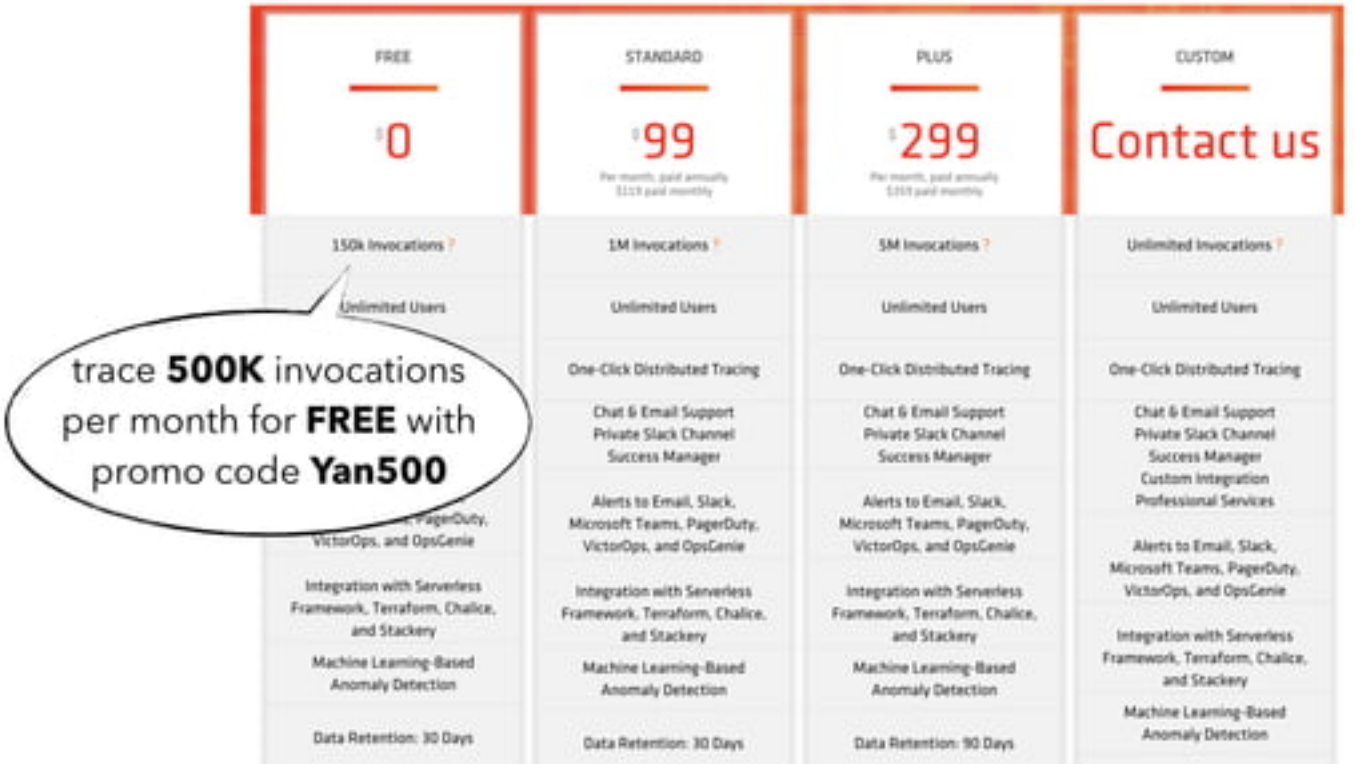
lumigo

lumigo

FREE	STANDARD	PLUS	CUSTOM
$0	$99 Per month, paid annually $119 paid monthly	$299 Per month, paid annually $359 paid monthly	Contact us
150k Invocations	1M Invocations	5M Invocations	Unlimited Invocations
Unlimited Users	Unlimited Users	Unlimited Users	Unlimited Users
One-Click Distributed Tracing	One-Click Distributed Tracing	One-Click Distributed Tracing	One-Click Distributed Tracing
Chat & Email Support Community Slack Channel	Chat & Email Support Private Slack Channel Success Manager	Chat & Email Support Private Slack Channel Success Manager	Chat & Email Support Private Slack Channel Success Manager Custom Integration Professional Services
Alerts to Email, Slack, Microsoft Teams, PagerDuty, VictorOps, and OpsGenie	Alerts to Email, Slack, Microsoft Teams, PagerDuty, VictorOps, and OpsGenie	Alerts to Email, Slack, Microsoft Teams, PagerDuty, VictorOps, and OpsGenie	Alerts to Email, Slack, Microsoft Teams, PagerDuty, VictorOps, and OpsGenie
Integration with Serverless Framework, Terraform, Chalice, and Stackery	Integration with Serverless Framework, Terraform, Chalice, and Stackery	Integration with Serverless Framework, Terraform, Chalice, and Stackery	Integration with Serverless Framework, Terraform, Chalice, and Stackery
Machine Learning-Based Anomaly Detection	Machine Learning-Based Anomaly Detection	Machine Learning-Based Anomaly Detection	Machine Learning-Based Anomaly Detection
Data Retention: 30 Days	Data Retention: 30 Days	Data Retention: 90 Days	

platform.lumigo.io/signup

lumigo



platform.lumigo.io/signup

lumigo

How to mitigate slow dependencies?

it depends…

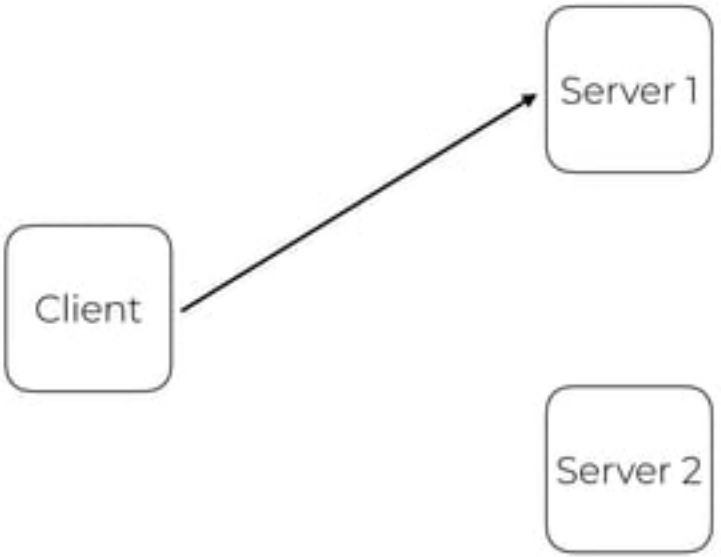
can you use another service?

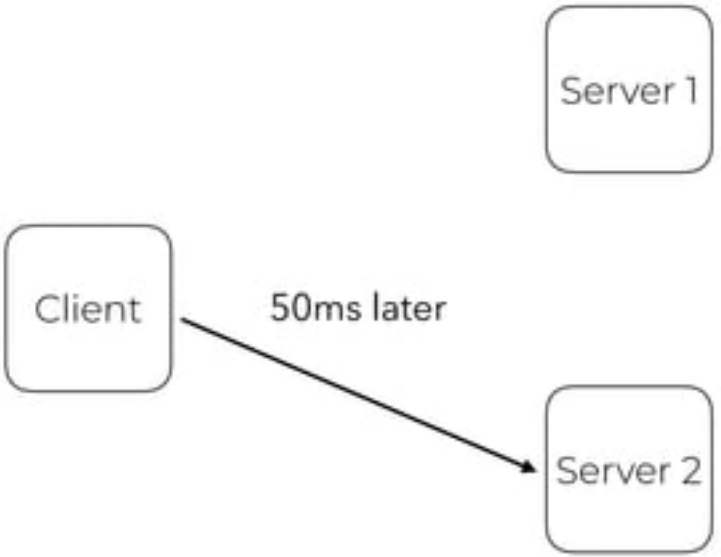
if not, a good caching strategy often helps

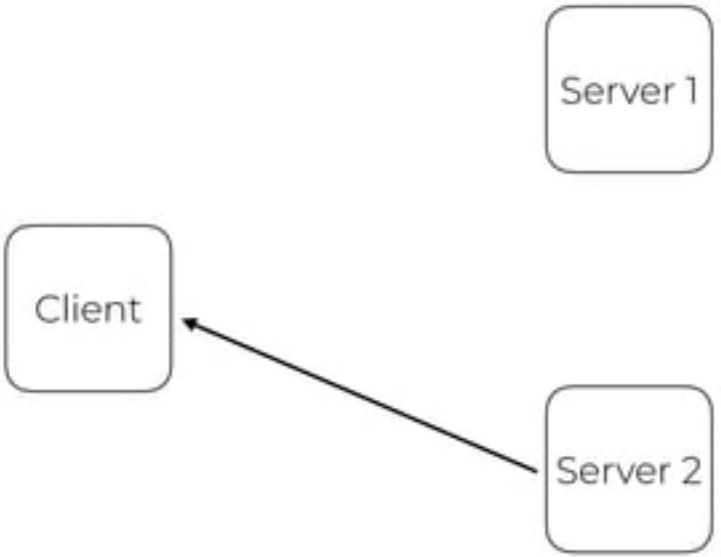
Achieving Rapid Response Times in Large Online Services

Jeff Dean
Google Fellow
jeff@google.com

bit.ly/3h7Bo41







Backup Requests Effects

- In-memory BigTable lookups
  - data replicated in two in-memory tables
  - issue requests for 1000 keys spread across 100 tablets
  - measure elapsed time until data for last key arrives

	Avg	Std Dev	95%ile	99%ile	99.9%ile
No backups	33 ms	1524 ms	24 ms	52 ms	994 ms
Backup after 10 ms	14 ms	4 ms	20 ms	23 ms	50 ms
Backup after 50 ms	16 ms	12 ms	57 ms	63 ms	68 ms

- Modest increase in request load:
  - 10 ms delay: <5% extra requests; 50 ms delay: <1%

Google

Backup Requests Effects

- In-memory BigTable lookups
  - data replicated in two in-memory tables
  - issue requests for 1000 keys spread across 100 tablets
  - measure elapsed time until data for last key arrives

	Avg	Std Dev	95%ile	99%ile	99.9%ile
No backups	33 ms	1524 ms	24 ms	52 ms	994 ms
Backup after 10 ms	14 ms	4 ms	20 ms	23 ms	50 ms
Backup after 50 ms	16 ms	12 ms	57 ms	63 ms	

runing required for each service

- Modest increase in request load:
  - 10 ms delay: <5% extra requests; 50 ms delay: <1%

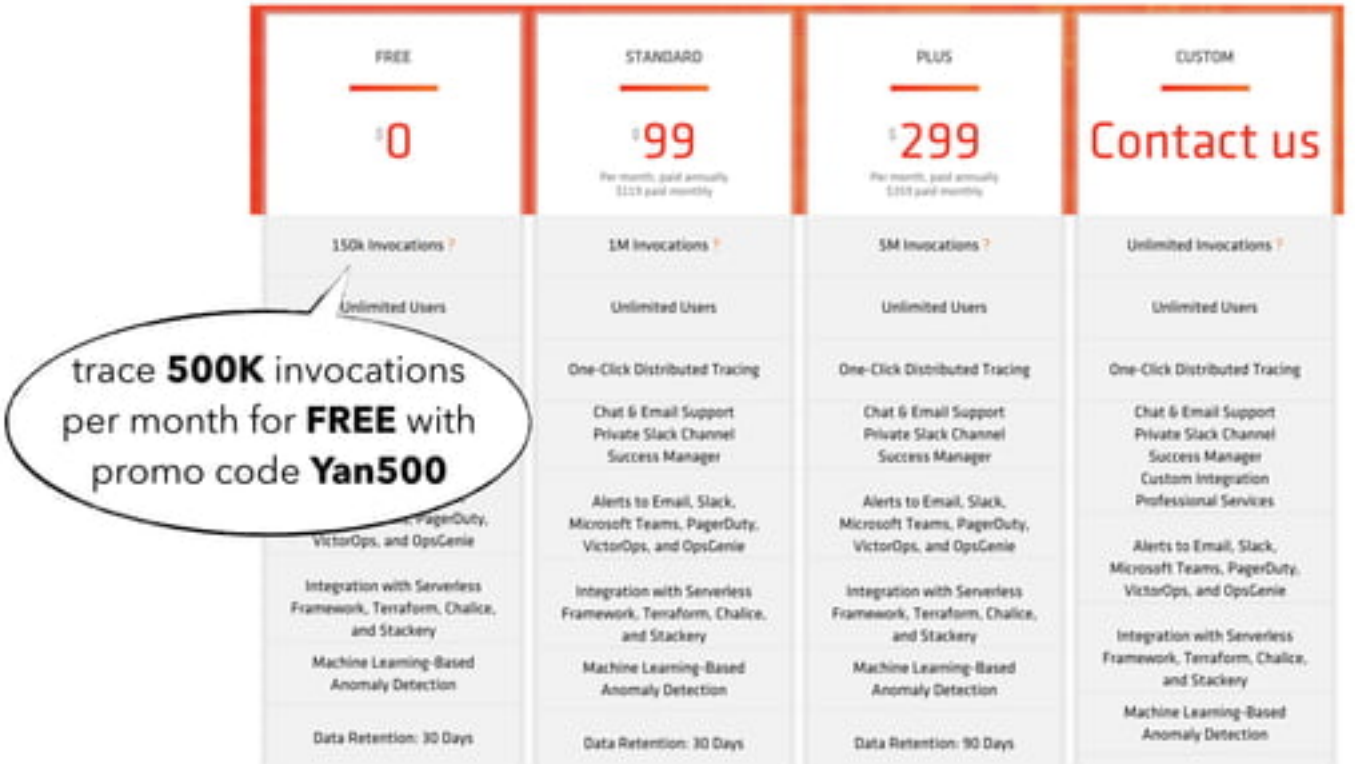
Google

helps in some cases

but can exaspate the problem in other cases

can you use another service?


"That's all Folks!"



platform.lumigo.io/signup

lumigo

@theburningmonk
theburningmonk.com
github.com/theburningmonk
yan@lumigo.io

lumigo