DynamoDB in Real Life

Jonathon Hill - Upstate PHP

@compwright

compwright.com

**For every advantage there is
an equal and opposite disadvantage.**

Cost-efficient

Unlimited scaling

Stateless

Transactions

Parallel operations

Atomic counters

Conditional writes

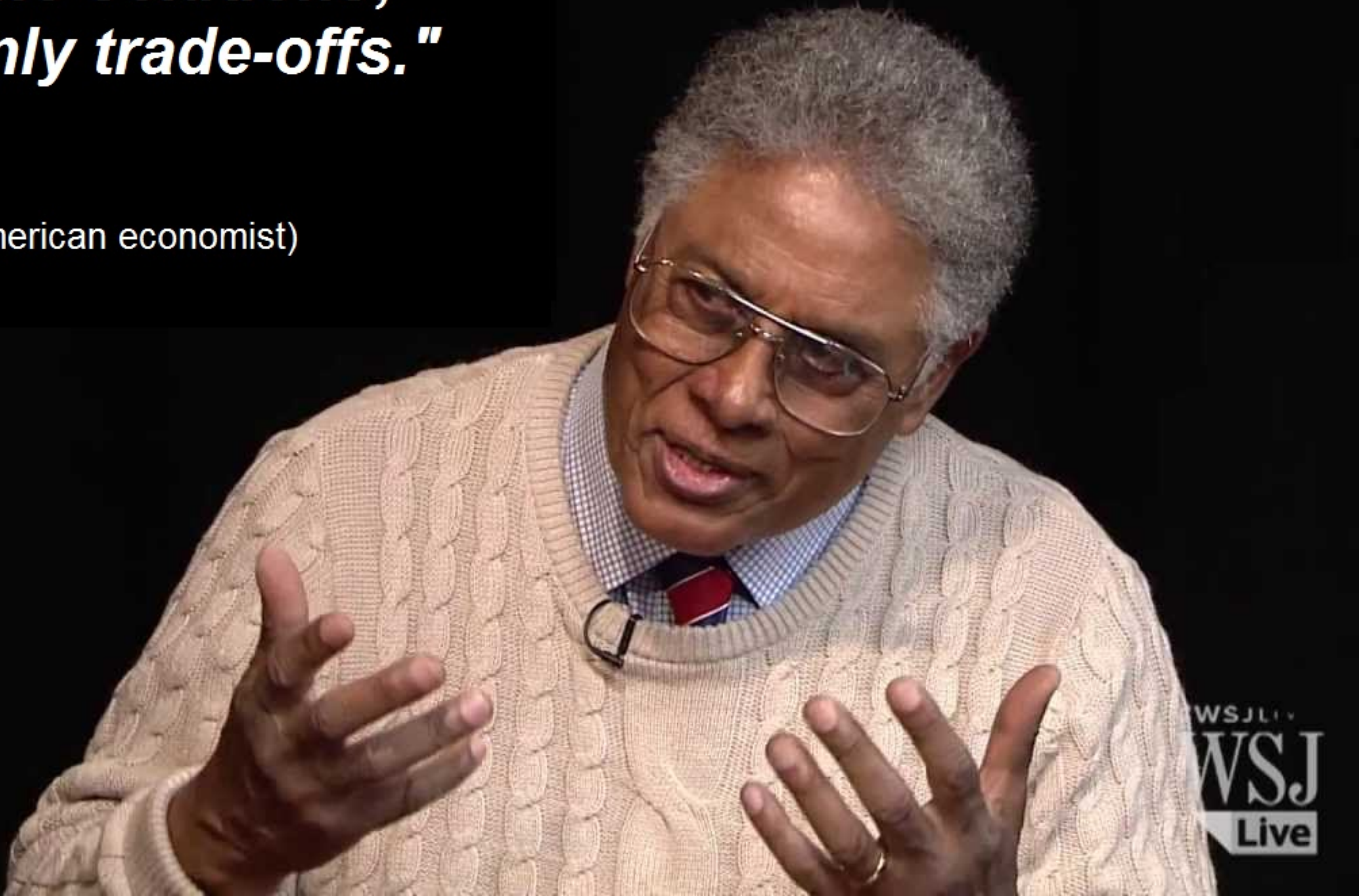
Expiration

Encryption

Streaming

"There are no solutions; there are only trade-offs."

(*Thomas Sowell*, American economist)



No indexes*

No date support

Small records

Slow console

Bad design = $$$

Idiosyncrasies

Query   Scan

GetItem   PutItem   UpdateItem   DeleteItem

TransactGetItems   TransactWriteItems

BatchGetItem   BatchWriteItem

…and more

```php
$operation = [
  'TableName' => $events_table,
  'Key' => [
    'messageId' => $id
  ],
  'UpdateExpression' => 'SET #status = :newStatus, #updatedAt = :timestamp',
  'ExpressionAttributeNames' => [
    '#status' => 'status',
    '#updatedAt' => 'updatedAt'
  ],
  'ExpressionAttributeValues' => [
    ':newStatus' => 'finished',
    ':timestamp' => $timestamp
  ],
  'ReturnValues' => 'UPDATED_NEW'
];
```

500 InternalServerError

400 ProvisionedThroughputExceededException

400 RequestLimitExceeded

400 ResourceNotFoundException

```json
{
    "ConsumedCapacity": {
        //...
    },
    "Count": 100,
    "Items": [
        //...
    ],
    "LastEvaluatedKey": {
        //...
    },
    "ScannedCount": 100
}
```

```php
function dynamodbPaginator(callable $next, callable $done): array {
  $allItems = [];
  $lastResult = [];

  do {
    $result = $next($lastResult);

    if ($next) {
      $next($result);
    } elseif ($result['Count'] > 0) {
      $allItems = array_merge($allItems, $result['Items']);
    }

    $lastResult = $result;
  } while ($lastResult['LastEvaluatedKey']);

  if (!$next) {
    return $allItems;
  }
}
```

```
[
  {
    "ReplyDateTime": {"S": "2015-02-18T20:27:36.165Z"},
    "PostedBy": {"S": "User A"},
    "Id": {"S": "Amazon DynamoDB#DynamoDB Thread 1"}
  },
  {
    "ReplyDateTime": {"S": "2015-02-25T20:27:36.165Z"},
    "PostedBy": {"S": "User B"},
    "Id": {"S": "Amazon DynamoDB#DynamoDB Thread 1"}
  }
]
```

NULL: null   **BOOL**: boolean

S: string   **N**: number   **B**: binary

SS: string set   **NS**: number set   **BS**: binary set

L: list   **M**: map

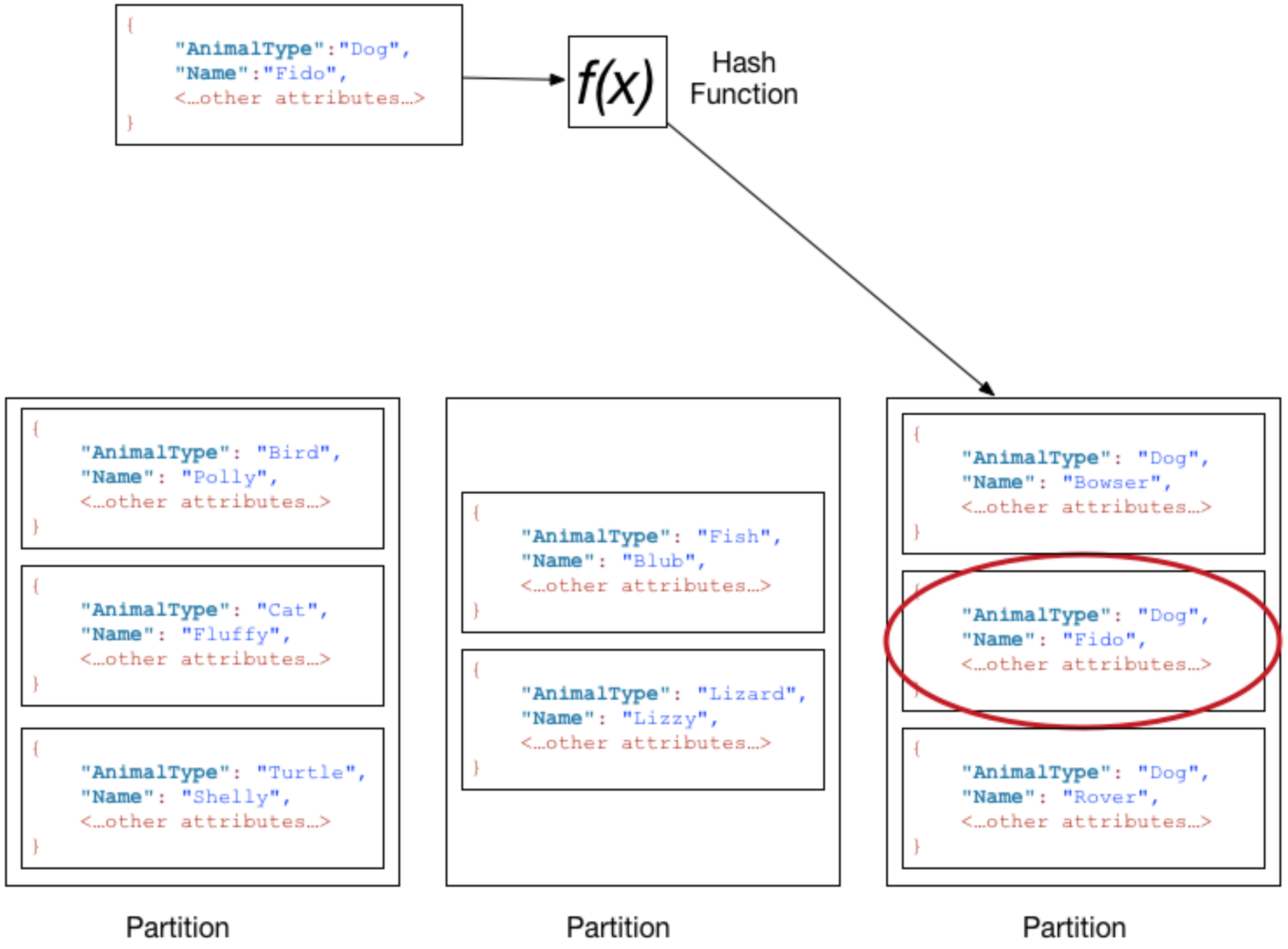
```php
$marshaler = new Aws\DynamoDb\Marshaler();

// Marshal into query
$query = [
  'TableName' => 'OrderEvents',
  'Key' => [
    'messageId' => $marshaler->marshalValue($id)
  ]
];

// Unmarshal query result item
$item = $marshaler->unmarshalItem($result['Item']);
```

	Eventually Consistent	Strongly Consistent	Transactional
Read per 4k	1/2 RCU	1 RCU	2 RCU
Write per 1k	1 WCU		2 WCU

Dual-key sharded design



```
{
    "AnimalType":"Dog",
    "Name":"Fido",
    <…other attributes…>
}
```

```
{
    "AnimalType": "Bird",
    "Name": "Polly",
    <…other attributes…>
}
```

```
{
    "AnimalType": "Cat",
    "Name": "Fluffy",
    <…other attributes…>
}
```

```
{
    "AnimalType": "Turtle",
    "Name": "Shelly",
    <…other attributes…>
}
```

```
{
    "AnimalType": "Fish",
    "Name": "Blub",
    <…other attributes…>
}
```

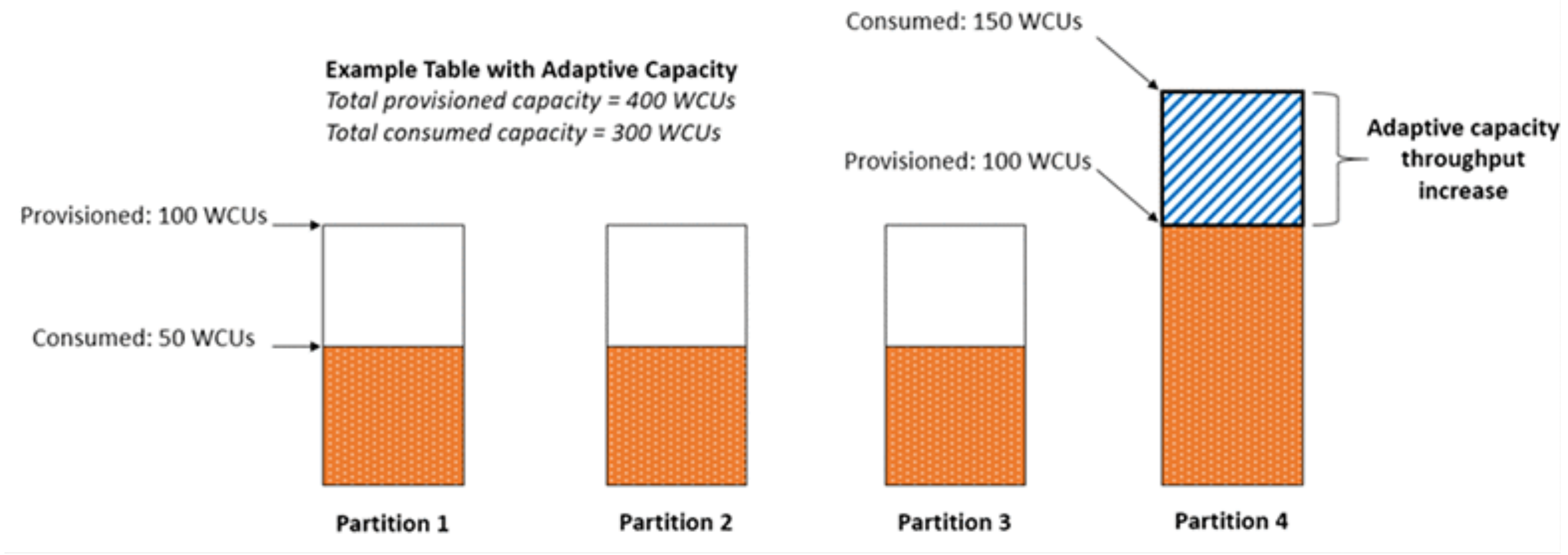
```
{
    "AnimalType": "Lizard",
    "Name": "Lizzy",
    <…other attributes…>
}
```

```
{
    "AnimalType": "Dog",
    "Name": "Bowser",
    <…other attributes…>
}
```

```
{
    "AnimalType": "Dog",
    "Name": "Fido",
    <…other attributes…>
}
```

```
{
    "AnimalType": "Dog",
    "Name": "Rover",
    <…other attributes…>
}
```

Partition key value	Possible values	Uniformity
User ID	Many	Good
Status code	Few	Bad
Timestamp, rounded to day, hour, etc	Few	Bad
Device ID, uniform traffic	Many	Good
Device ID, a few “hot” devices	Many	Bad


Example Table with Adaptive Capacity
Total provisioned capacity = 400 WCUs
Total consumed capacity = 300 WCUs
Consumed: 150 WCUs
Provisioned: 100 WCUs
Adaptive capacity throughput increase
Provisioned: 100 WCUs
Consumed: 50 WCUs
Partition 1
Partition 2
Partition 3
Partition 4

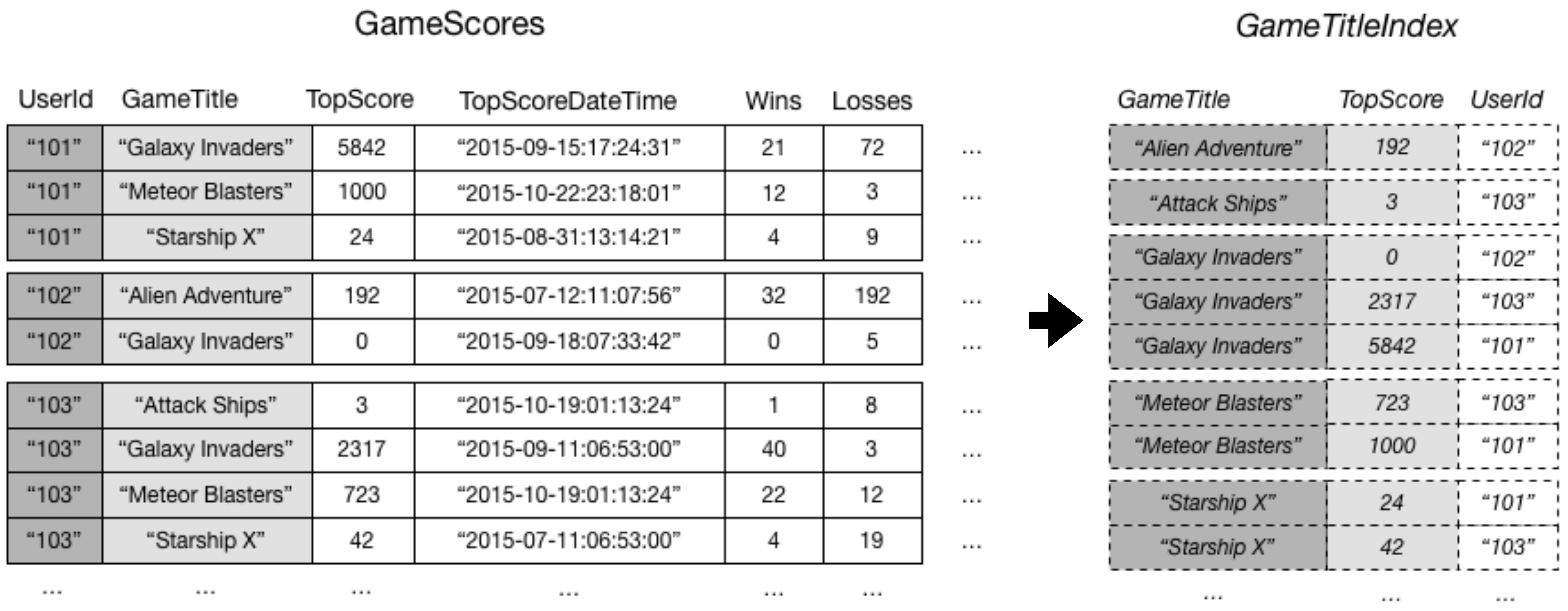
Query by key

vs.

Scan all

Secondary Indexes

Global Secondary Index



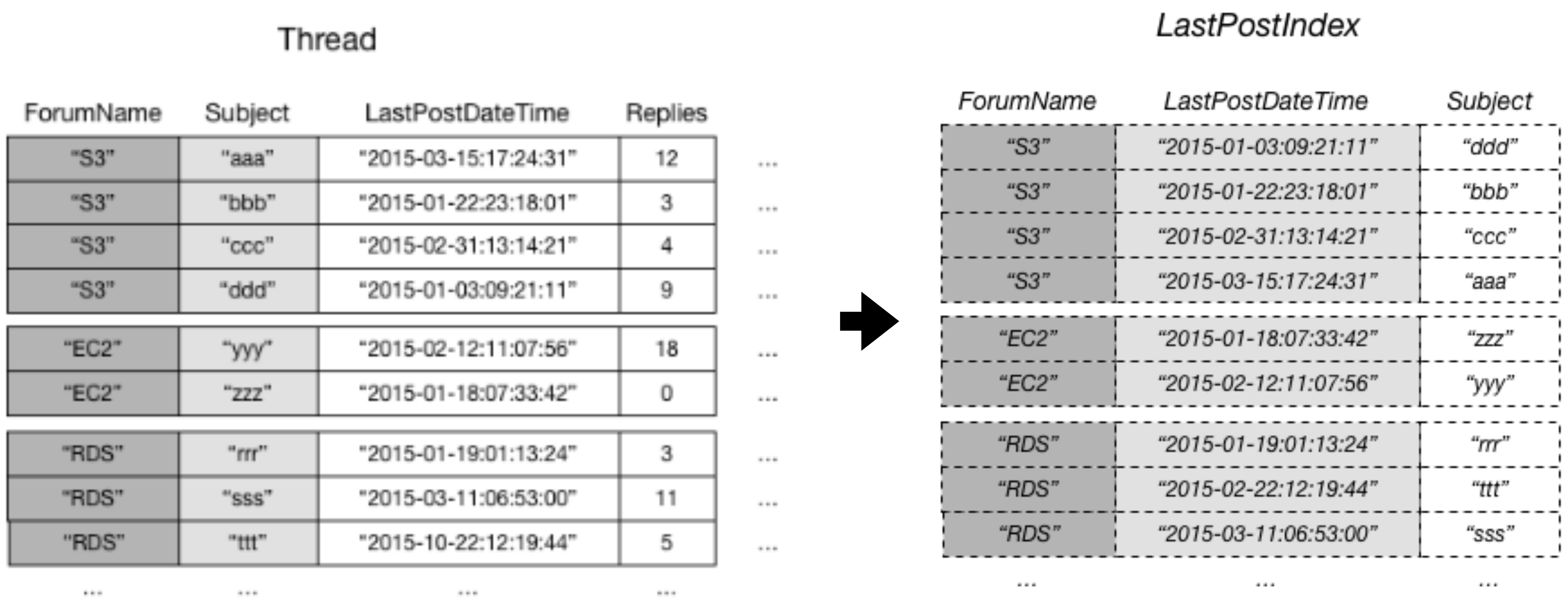
GameScores

UserId	GameTitle	TopScore	TopScoreDateTime	Wins	Losses	
“101”	“Galaxy Invaders”	5842	“2015-09-15:17:24:31”	21	72	…
“101”	“Meteor Blasters”	1000	“2015-10-22:23:18:01”	12	3	…
“101”	“Starship X”	24	“2015-08-31:13:14:21”	4	9	…
“102”	“Alien Adventure”	192	“2015-07-12:11:07:56”	32	192	…
“102”	“Galaxy Invaders”	0	“2015-09-18:07:33:42”	0	5	…
“103”	“Attack Ships”	3	“2015-10-19:01:13:24”	1	8	…
“103”	“Galaxy Invaders”	2317	“2015-09-11:06:53:00”	40	3	…
“103”	“Meteor Blasters”	723	“2015-10-19:01:13:24”	22	12	…
“103”	“Starship X”	42	“2015-07-11:06:53:00”	4	19	…
…	…	…	…	…	…	

GameTitleIndex

GameTitle	*TopScore*	*UserId*
“Alien Adventure”	*192*	*“102”*
“Attack Ships”	*3*	*“103”*
“Galaxy Invaders”	*0*	*“102”*
“Galaxy Invaders”	*2317*	*“103”*
“Galaxy Invaders”	*5842*	*“101”*
“Meteor Blasters”	*723*	*“103”*
“Meteor Blasters”	*1000*	*“101”*
“Starship X”	*24*	*“101”*
“Starship X”	*42*	*“103”*
…	…	…

Local Secondary Index



Thread

ForumName	Subject	LastPostDateTime	Replies	
"S3"	"aaa"	"2015-03-15:17:24:31"	12	...
"S3"	"bbb"	"2015-01-22:23:18:01"	3	...
"S3"	"ccc"	"2015-02-31:13:14:21"	4	...
"S3"	"ddd"	"2015-01-03:09:21:11"	9	...
"EC2"	"yyy"	"2015-02-12:11:07:56"	18	...
"EC2"	"zzz"	"2015-01-18:07:33:42"	0	...
"RDS"	"rrr"	"2015-01-19:01:13:24"	3	...
"RDS"	"sss"	"2015-03-11:06:53:00"	11	...
"RDS"	"ttt"	"2015-10-22:12:19:44"	5	...
...	...	...	...	

LastPostIndex

ForumName	*LastPostDateTime*	*Subject*
"S3"	*"2015-01-03:09:21:11"*	*"ddd"*
"S3"	*"2015-01-22:23:18:01"*	*"bbb"*
"S3"	*"2015-02-31:13:14:21"*	*"ccc"*
"S3"	*"2015-03-15:17:24:31"*	*"aaa"*
"EC2"	*"2015-01-18:07:33:42"*	*"zzz"*
"EC2"	*"2015-02-12:11:07:56"*	*"yyy"*
"RDS"	*"2015-01-19:01:13:24"*	*"rrr"*
"RDS"	*"2015-02-22:12:19:44"*	*"ttt"*
"RDS"	*"2015-03-11:06:53:00"*	*"sss"*
...	...	...

Size

Shape

Velocity

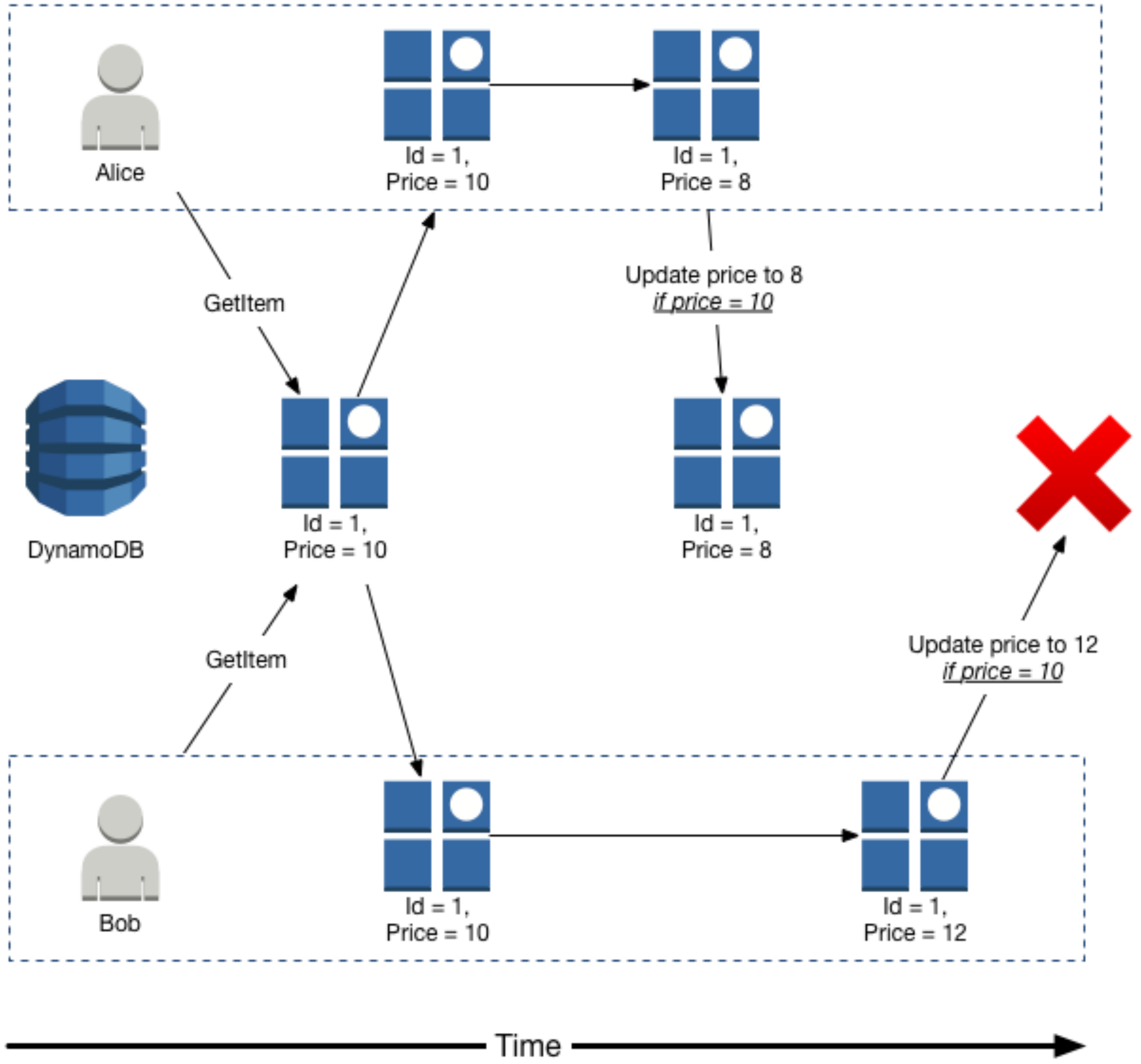
Idempotence and Concurrency

Webhooks

Amazon SQS

Amazon SNS

Amazon Lambda


Alice
Id = 1, Price = 10
Id = 1, Price = 8
GetItem
Update price to 8
if price = 10
DynamoDB
Id = 1, Price = 10
Id = 1, Price = 8
GetItem
Update price to 12
if price = 10
Bob
Id = 1, Price = 10
Id = 1, Price = 12
Time

Real Life

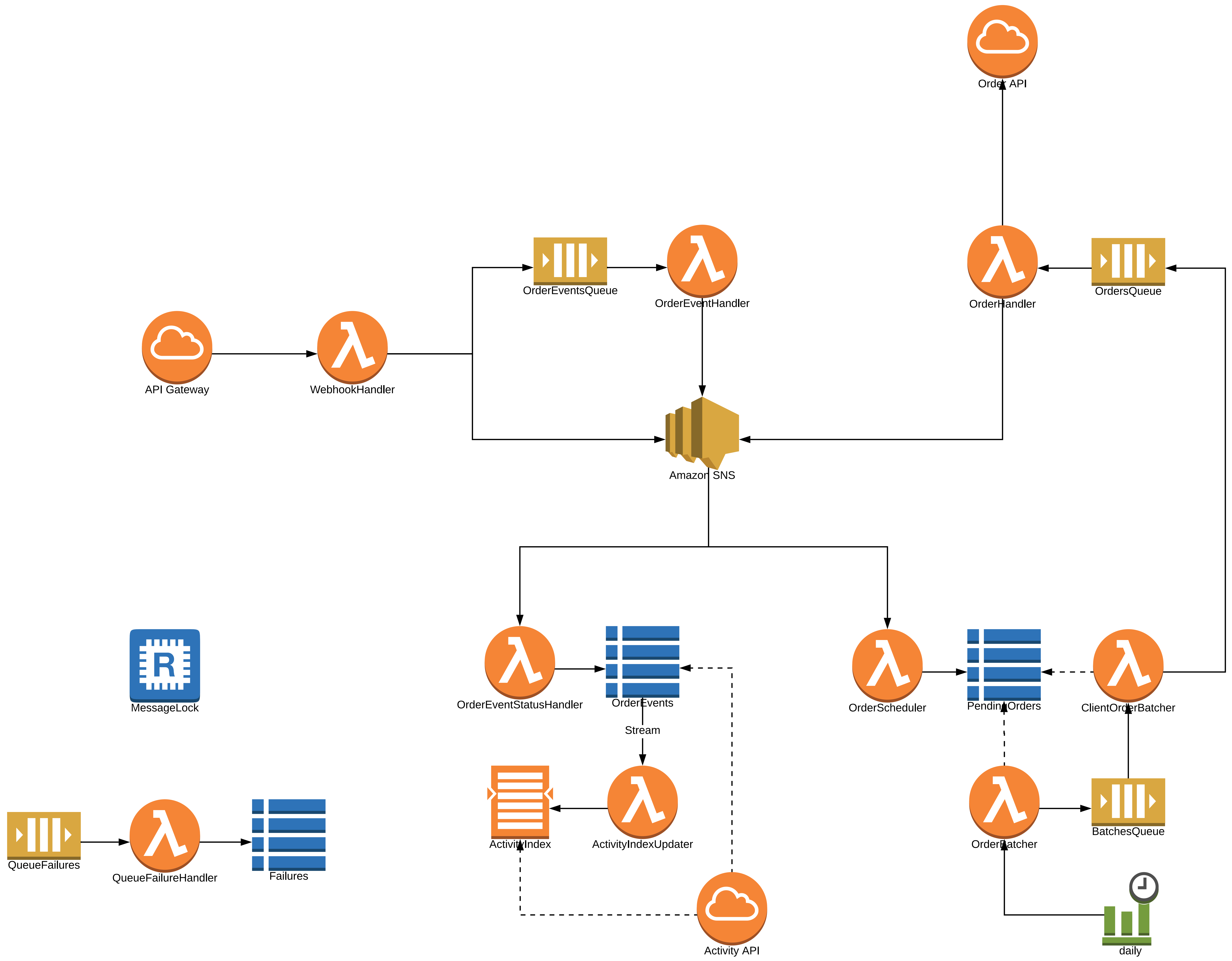
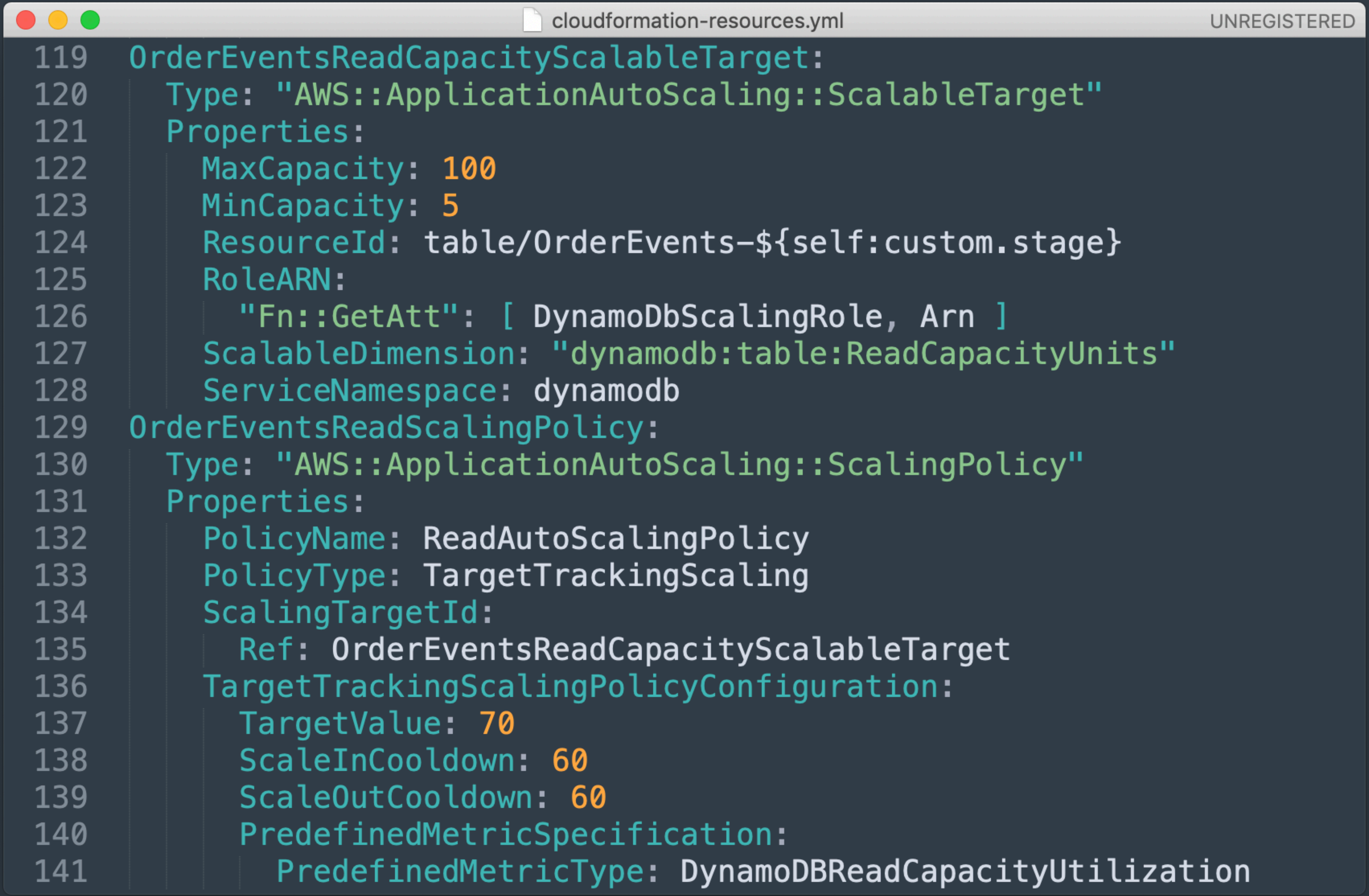

Order API
OrderEventsQueue
OrderEventHandler
OrderHandler
OrdersQueue
API Gateway
WebhookHandler
Amazon SNS
MessageLock
OrderEventStatusHandler
OrderEvents
Stream
OrderScheduler
PendingOrders
ClientOrderBatcher
ActivityIndex
ActivityIndexUpdater
OrderBatcher
BatchesQueue
QueueFailures
QueueFailureHandler
Failures
Activity API
daily

Table	Partition Key	Sort Key	Size	RCUs	WCUs
OrderEvents	messageId		3GB	5-100	5-500
PendingOrders	clientId	messageId	56KB	5-500	5-500
Failures	messageId		68KB	5-50	5-50

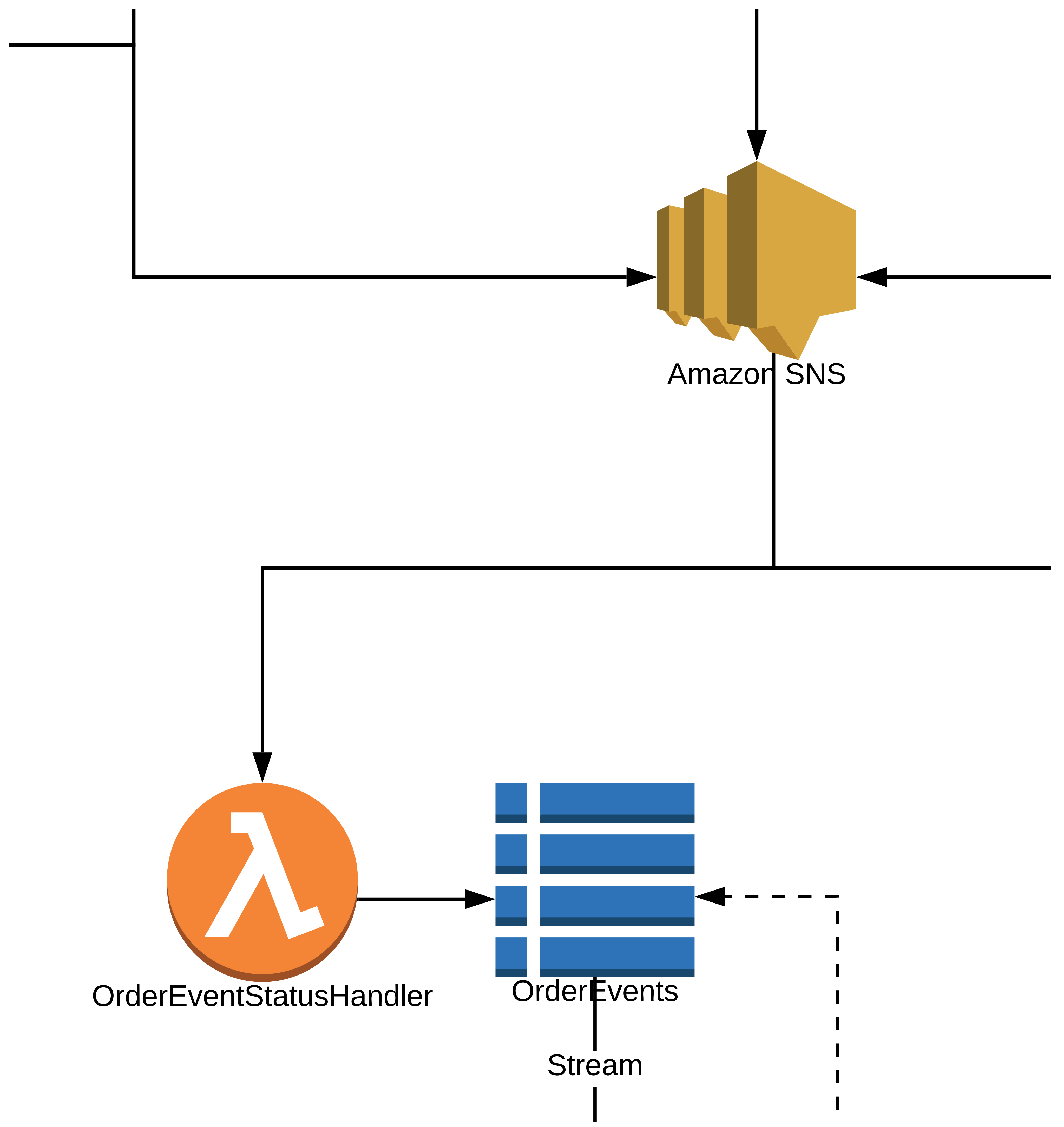
Ran out of
compute units



```yaml
OrderEventsReadCapacityScalableTarget:
  Type: "AWS::ApplicationAutoScaling::ScalableTarget"
  Properties:
    MaxCapacity: 100
    MinCapacity: 5
    ResourceId: table/OrderEvents-${self:custom.stage}
    RoleARN:
      "Fn::GetAtt": [ DynamoDbScalingRole, Arn ]
    ScalableDimension: "dynamodb:table:ReadCapacityUnits"
    ServiceNamespace: dynamodb
OrderEventsReadScalingPolicy:
  Type: "AWS::ApplicationAutoScaling::ScalingPolicy"
  Properties:
    PolicyName: ReadAutoScalingPolicy
    PolicyType: TargetTrackingScaling
    ScalingTargetId:
      Ref: OrderEventsReadCapacityScalableTarget
    TargetTrackingScalingPolicyConfiguration:
      TargetValue: 70
      ScaleInCooldown: 60
      ScaleOutCooldown: 60
      PredefinedMetricSpecification:
        PredefinedMetricType: DynamoDBReadCapacityUtilization
```

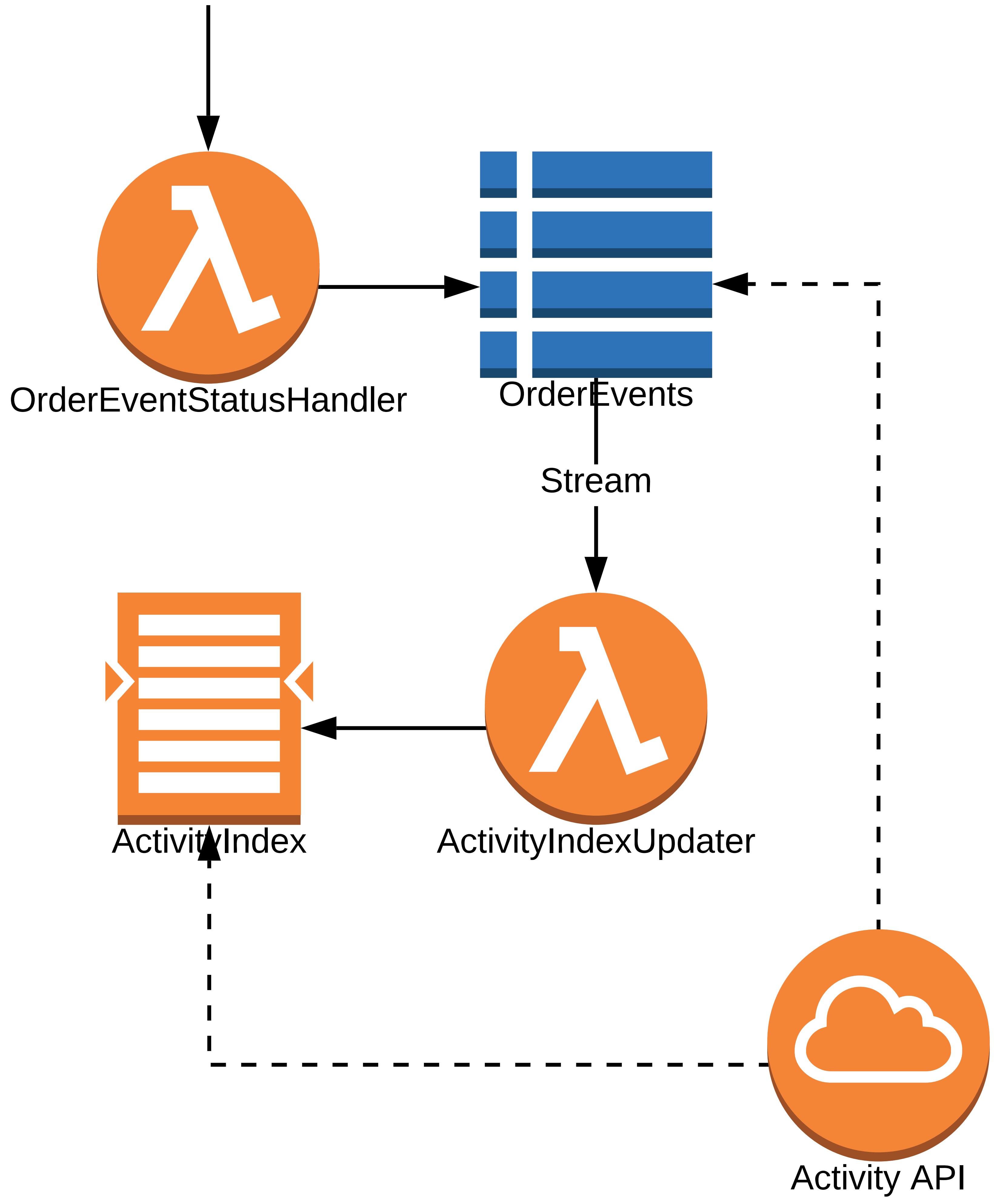
Not enough time to retry

Race conditions


Amazon SNS
OrderEventStatusHandler
OrderEvents
Stream

Status	ConditionExpression
created	#id <> :id and #did <> :did
started	#status = :current and #status <> :newStatus
queued	#status <> :final and #status <> :newStatus
finished	

Search and filter


OrderEventStatusHandler
OrderEvents
Stream
ActivityIndex
ActivityIndexUpdater
Activity API

More race conditions

Status	Step	UpdateExpression
created	1	
started	2	SET #status = :newStatus, #step = :step
queued	4	SET #status = :newStatus, #step = :step
error	128	SET #status = :newStatus, #step = :step
finished	+1	SET #status = :newStatus, #step = #step + :step

Hard 10GB limit

Tips

RTFM!

It ain’t a lock

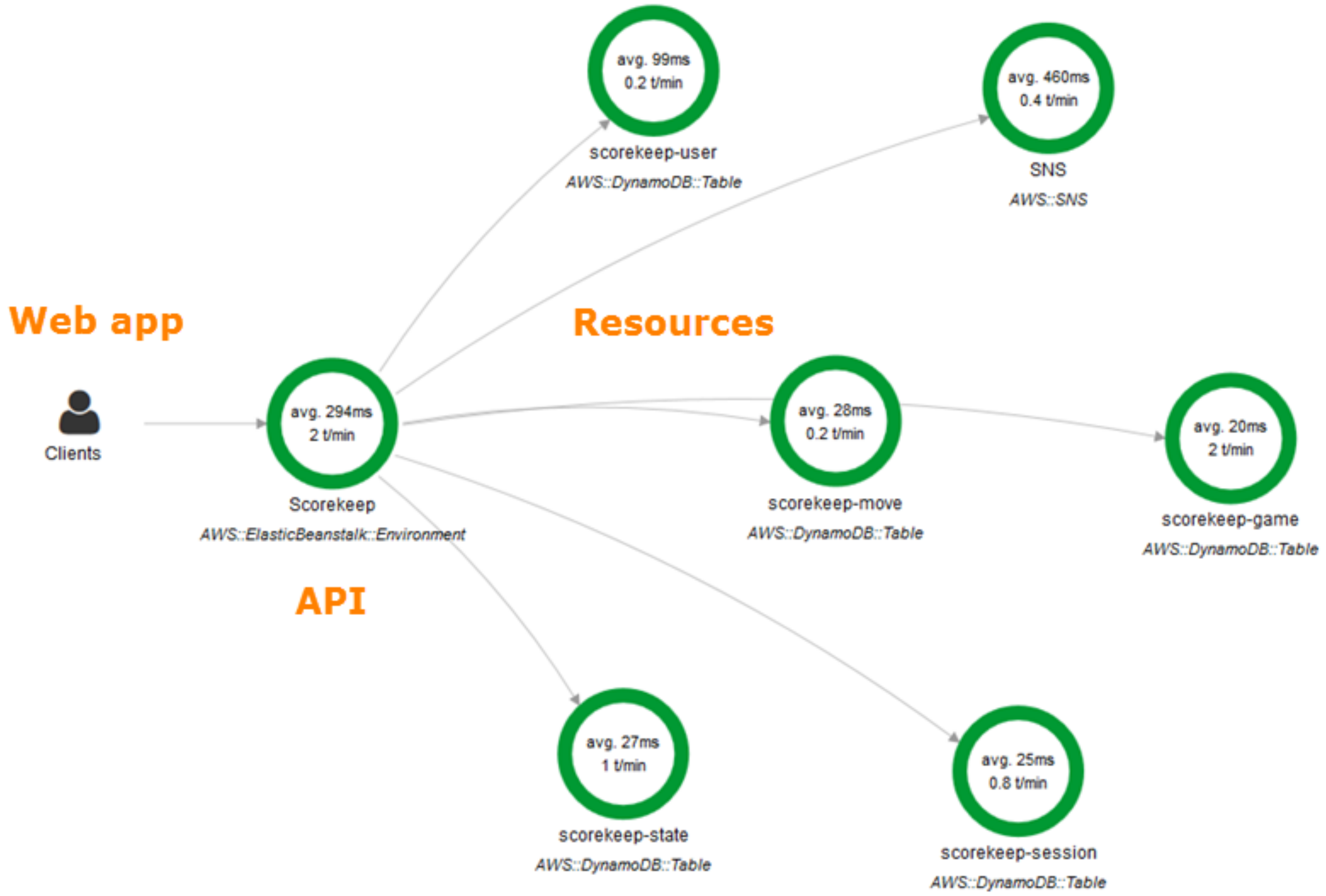
Bind attribute names

Handle ConditionalCheckFailed errors and retry

Use CloudFormation

Auto-scale

X-Ray


Web app
Resources
API
Clients
avg. 294ms
2 t/min
Scorekeep
AWS::ElasticBeanstalk::Environment
avg. 99ms
0.2 t/min
scorekeep-user
AWS::DynamoDB::Table
avg. 460ms
0.4 t/min
SNS
AWS::SNS
avg. 28ms
0.2 t/min
scorekeep-move
AWS::DynamoDB::Table
avg. 20ms
2 t/min
scorekeep-game
AWS::DynamoDB::Table
avg. 27ms
1 t/min
scorekeep-state
AWS::DynamoDB::Table
avg. 25ms
0.8 t/min
scorekeep-session
AWS::DynamoDB::Table

Questions?

Thank You!

http://compwright.com/talks/dynamodb-in-real-life

@compwright
jonathon@compwright.com
864-245-5885