Introduction to AWS DynamoDB

Omid Vahdaty, Big Data ninja.

Concepts

- Schema less - other than the primary key attributes, you do not need to define any attributes or data types at table creation time.
- Table == collection/Table from OLTP
- Item == "Row in a table".
- Attributes == "col of a row " in a table. May change from item to item.
- Primary key ==a primary key uniquely identifies each item in the table, so that no two items can have the same key. 2 types
  - **Partitions key** - simple **hash** function that maps **item into specific partition**
  - **Partition + sort key -** A composite primary key
    - Hash function maps item into partition. Also called ***hash attribute***
    - The item will be sorted via sort key. Also called **range attribute.**
- **Secondary Index. Non key index.** Another way to read data. **Faster? :)**
  - Global secondary index – an index with a partition key and sort key that can be different from those on the table.
  - Local secondary index – an index that has the same partition key as the table, but a different sort key.

DynamoDB stream

Optional feature : Each event is represented by a *stream record*. If you enable a stream on a table, DynamoDB Streams will write a stream record whenever one of the following events occurs:

- If a new item is added to the table, the stream captures an image of the entire item, including all of its attributes.
- If an item is updated, the stream captures the "before" and "after" image of any attributes that were modified in the item.
- If an item is deleted from the table, the stream captures an image of the entire item before it was deleted.
- Lifetime of 24 hours
- Actions
    a. `ListStreams`
    b. `DescribeStream`
    c. `GetShardIterator`
    d. `GetRecords`

DynamoDB API

- Control Plane: create and manage tables:
    - `CreateTable`
    - `DescribeTable`
    - `ListTables`
    - `UpdateTable`
    - `DeleteTable`
- Data Plane : actions on data:
    - create, read, update, and delete

DynamoDB API

- Data Plane : actions on data:
  - Create
    - putItem - only 1 item, must specify primary key
    - batchWriteItem - write upto 25 item in transaction
  - Read
    - getItem - get only 1 item
    - batchGetItems - get upto 100 items.
    - Query
    - Scan
  - Update - updateItem, 1 item, spesify primary ket, atomic counters to save writes? ;)
  - Delete, deleteItem upto 1 item, BatchDeleteItems. (upto 25)

DynamoDB data types

Scalar Types – A scalar type can represent **exactly one value**. The scalar types are number, string, binary, Boolean, and null.

Document Types – A document type can represent a **complex structure with nested attributes**—such as you would find in a JSON document. The document types are list and map.

Set Types – A set type can represent **multiple scalar values**. The set types are string set, number set, and binary set.

Documents?

- complex data structures up to 32 levels deep.
- size limit (400 KB).
- List:
  - A list type attribute can store an ordered collection of values
  - E.g `FavoriteThings: ["Cookies", "Coffee", 3.14159]`
- Map - A map type attribute can store an unordered collection of name-value pairs, ideal for json.

  -
    ```
    {
        Day: "Monday",
        UnreadEmails: 42,
        ItemsOnMyDesk: [
            "Coffee Cup",
            "Telephone",
            {
                Pens: { Quantity : 3},
                Pencils: { Quantity : 2},
                Erasers: { Quantity : 1}
            }
        ]
    }
    ```

Sets?

- item size limit (400 KB).
- DynamoDB does not support empty sets.
- **Example (String Set, Number Set, and Binary Set)**
-

```
["Black", "Green" ,"Red"]

[42.2, -19, 7.5, 3.14]

["U3Vubnk=", "UmFpbnk=", "U25vd3k="]
```

Provision IO

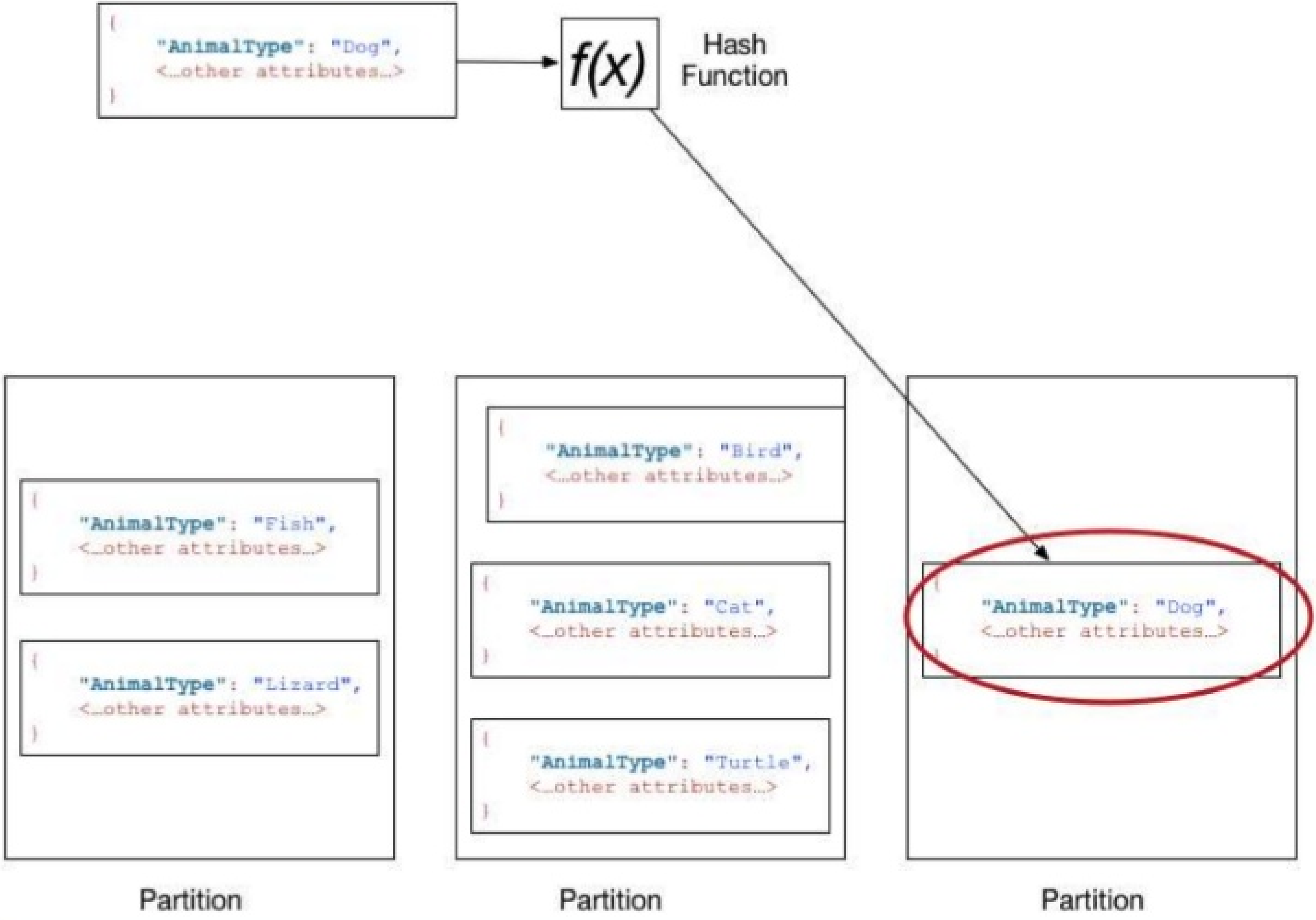
Read carefully:

http://docs.aws.amazon.com/amazondynamodb/latest/developerguide/HowItWorks.ProvisionedThroughput.html

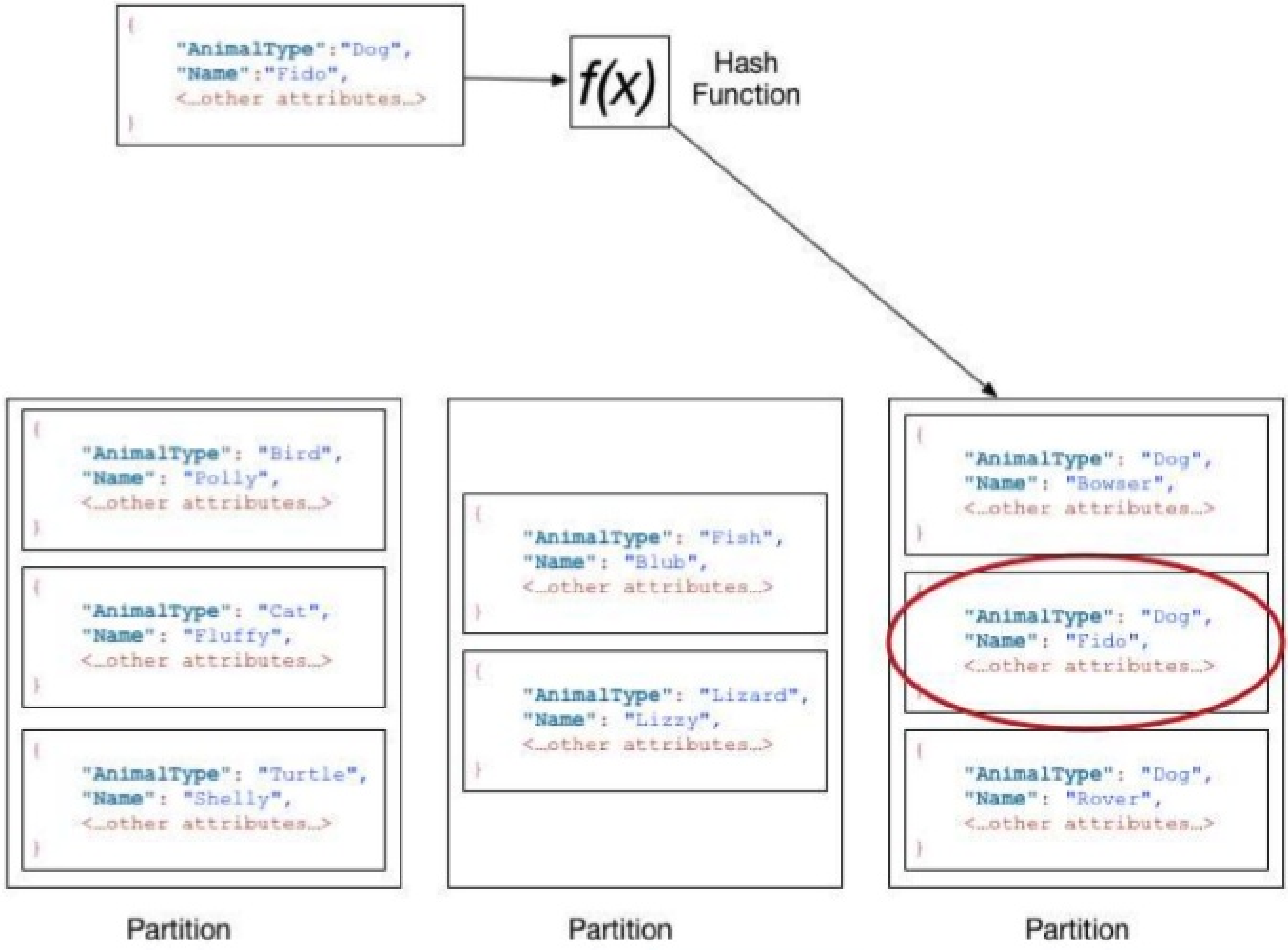
Partitions?

- DynamoDB stores data in partitions.
- **A *partition* is an allocation of storage for a table**
- backed by solid-state drives (SSDs)
- **Multi AZ - automatically** replicated across multiple Availability Zones within an AWS region.
- Partition management is handled automatically by DynamoDB
- Partion key in a table - hash map function to map an item to a partion

Partitions key



Partitions key
Sort Key



```
{
    "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…>
}
```

Getting started

- Download(!) dynamo to your laptop for training: http://docs.aws.amazon.com/amazondynamodb/latest/gettingstartedguide/GettingStarted.Download.html

- `Start server`
  - `java -Djava.library.path=./DynamoDBLocal_lib -jar DynamoDBLocal.jar -sharedDb -inMemory`
  - `Port 8000`

Java and DynamoDB

Setup access key , secret key

Setup JAVA and dynamo DB:
http://docs.aws.amazon.com/sdk-for-java/v1/developer-guide/setup-install.html

Easy to get started with Ecplise, didnt find anything supported for Intellij
https://www.eclipse.org/downloads

Setup AWS tool kit
http://docs.aws.amazon.com/toolkit-for-eclipse/v1/user-guide/setup-install.html

Setup Java SDK:
http://docs.aws.amazon.com/sdk-for-java/v1/developer-guide/setup-install.html

Java and DynamoDB

Using the SDK:
add the full path to the **lib** and **third-party** directories to the dependencies in your build file, and add them to your java **CLASSPATH** to run your code.

Working with local (onsite) installation of dynamodb:
https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/DynamoDBLocal.html

- `java -Djava.library.path=./DynamoDBLocal_lib -jar DynamoDBLocal.jar -sharedDb`

Command line options for local DynamoDB (in memory, path to save files etc):
https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/DynamoDBLocal.html

Specify local end point example:
https://github.com/aws/aws-sdk-java/issues/330

Sample Code

```
final String USAGE = "\n" +"Usage:\n" + "    CreateTable <table>\n\n" +  "Where:\n" +   "    table - the table to create.\n\n" + "Example:\n" + "CreateTable Hell

                              if (args.length < 1) {

                                  System.out.println(USAGE);

                                  System.exit(1);

                              }                        /* Read the name from command args */

                              String table_name = args[0];

                              System.out.format(

                                  "Creating table \"%s\" with a simple primary key: \"Name\".\n",         table_name);
```

Sample Code

```
CreateTableRequest request = new CreateTableRequest()

    .withAttributeDefinitions(new AttributeDefinition("Name", ScalarAttributeType.S))

    .withKeySchema(new KeySchemaElement("Name", KeyType.HASH))

    .withProvisionedThroughput(new ProvisionedThroughput(new Long(1), new Long(1)))

    .withTableName(table_name);



final AmazonDynamoDBClient dynamoDB = new AmazonDynamoDBClient();

dynamoDB.setEndpoint("http://127.0.0.1:8000");

dynamoDB.setSignerRegionOverride("eu-west-1");
```

Sample Code

```
try {
    CreateTableResult result = dynamoDB.createTable(request);
} catch (AmazonServiceException e) {
    System.err.println(e.getErrorMessage());
    System.exit(1);
}
System.out.println("Done!");
```