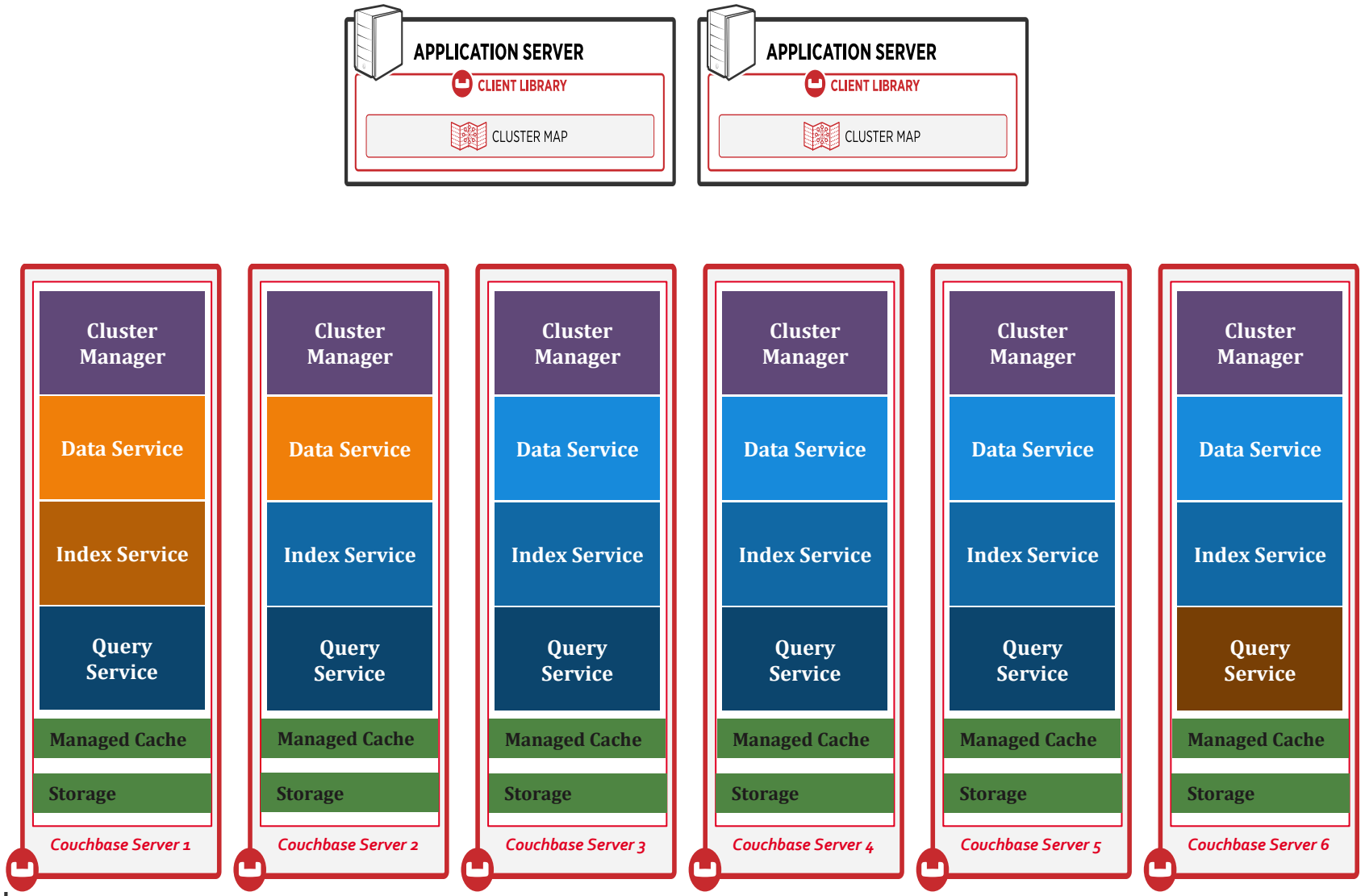
HP

Global Secondary Indexes
new high performance indexer

Couchbase Server Cluster Architecture



HP

Indexing in Couchbase Server 4.0

- **Multiple Indexers**
  - **GSI – Index Service**

New indexing for N1QL for low latency queries without compromising on mutation performance (insert/update/delete)

Independently partitioned and independently scalable indexes in Indexing Service

  - **Map/Reduce Views – Data Service**

Powerful programmable indexer for complex reporting and indexing logic.

Full partition alignment and paired scalability with Data Service.

  - **Spatial View – Data Service**

Incremental R-tree indexing for powerful bounding-box queries

Full partition alignment and paired scalability with Data Service

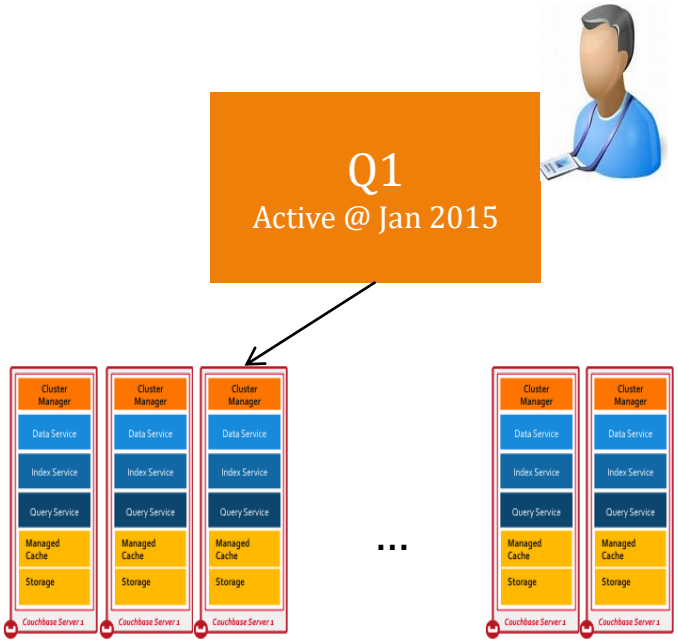
HP

Query and Index Today

Once upon a time in a User Profile System….

- Q1: Find the top 10 most “active” customer by #logins in Jan 2015

```
{…
  “customer_name” : ”Cihan”,
  “total_logins”: {…
                “aug_2015”:100,
                …}
  “type” : “customer_profile”
  …}
```

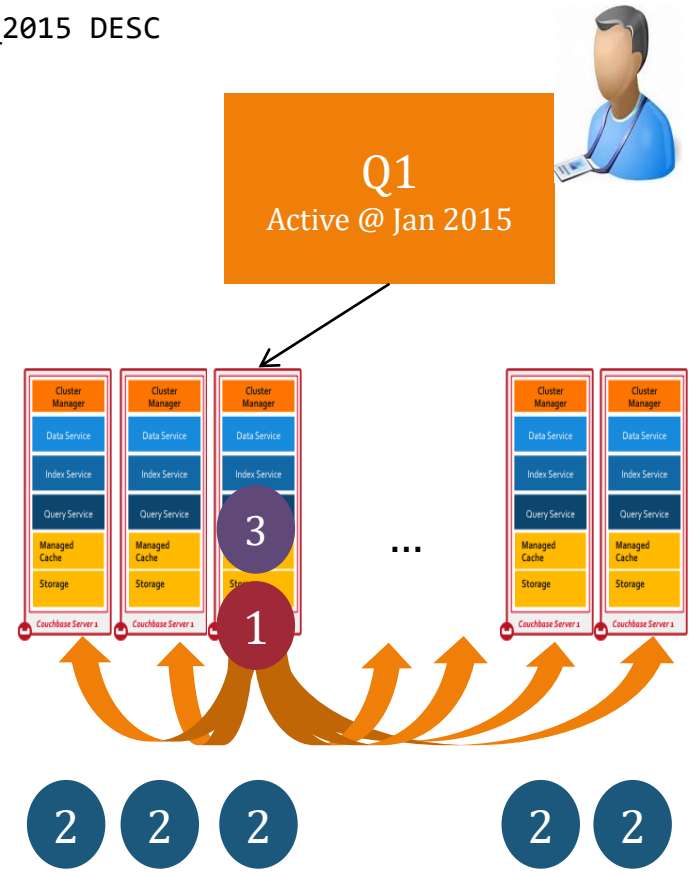


HP

Query and Index Today

```
INDEX ON Customer_bucket(customer_name, total_logins.jan_2015)
WHERE type="customer_profile";

SELECT customer_name, total_logins.jan_2015
FROM customer_bucket
WHERE type="customer_profile"
ORDER BY total_logins.jan_2015 DESC
LIMIT 10;
```



Q1: Execution Plan on N nodes

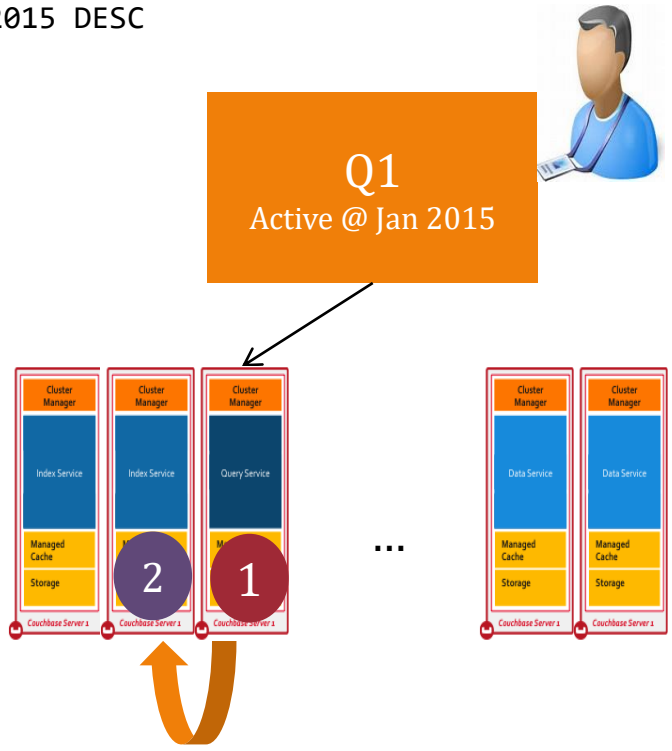
- Scatter: Execute Q1 on N nodes
- Gather: gather N results
- Finalize: Execute Q1 on governor node

HP

Query and Index with GSI

```
INDEX ON Customer_bucket(customer_name, total_logins.jan_2015)
WHERE type="customer_profile";

SELECT customer_name, total_logins.jan_2015
FROM customer_bucket
WHERE type="customer_profile"
ORDER BY total_logins.jan_2015 DESC
LIMIT 10;
```



Q1: Execution Plan on N nodes

- Execute Q1 on N1QL Service node
- Scan index on Index Service node

HP

Introducing Global Secondary Indexes

What are Global Secondary Indexes?

High performance indexes for low latency queries with powerful caching, storage and independent placement.

- Power of GSI
  - **Fully integrated into N1QL** Query Optimization and Execution
  - **Independent Index Distribution** for Limiting scatter-gather
  - **Independent Scalability** with Index Service – *more on this later*
  - **Powerful caching and storage** with ForestDB

HP

Which to choose – GSI vs Views

Workloads	New GSI in v4.0	Map/Reduce Views
Complex Reporting	Just In Time Aggregation	Pre-aggregated
Workload Optimization	Optimized for Scan Latency & Throughput	Optimized for Insertion
Flexible Index Logic	N1QL Functions	Javascript
Secondary Lookups	Single Node Lookup	Scatter-Gather
Tunable Consistency	Staleness false or ok or everything in between	Staleness false or ok

HP

Which to choose – GSI vs Views

Capabilities	New GSI in v4.0	Map/Reduce Views
Partitioning Model	Independent – Indexing Service	Aligned to Data – Data Service
Scale Model	Independently Scale Index Service	Scale with Data Service
Fetch with Index Key	Single Node	Scatter-Gather
Range Scan	Single Node	Scatter-Gather
Grouping, Aggregates	With N1QL	Built-in with Views API
Caching	Managed	Not Managed
Storage	ForestDB	Couchstore

HP

Indexing Lifecycle

- Primary vs Secondary
  - Primary Index is a full list of document keys within a given bucket

```
CREATE PRIMARY INDEX index_name
ON bucket_name
USING GSI|VIEW
WITH `{"nodes": ["node_name"], "defer_build":true}`; //GSI-ONLY
```

  - Secondary Index is an index on a field/expression on a subset of documents for lookups

```
CREATE INDEX index_name
ON bucket_name (field/expression, …)
USING GSI|VIEW
WHERE filter_expressions
WITH `{"nodes": ["node_name"], "defer_build":true}`; //GSI-ONLY
```

HP

Deferred Index Building

- Index building can be deferred to build multiple indexes all at once with greater scan efficiency.

```
CREATE INDEX … WITH {…“defer_build”:true};
CREATE INDEX … WITH {…“defer_build”:true};
…
BUILD INDEX ON bucket_name(index_name1, …) USING GSI;
```

GSI Partitioning and Placement

HP

GSI Placement and Partitioning

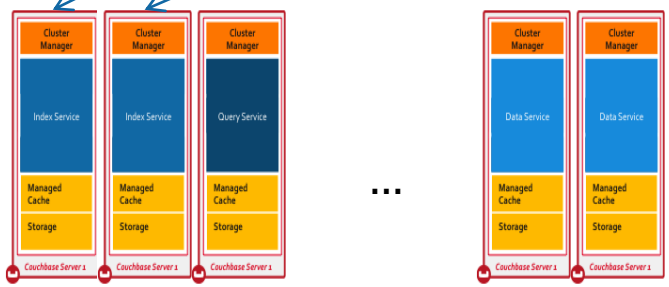
Place GSI Indexes using NODES clause

- *Each GSI reside on 1 node*
  - You can specify the node using nodes clause

```
CREATE INDEX i1 … WITH {“nodes”:”node1”};
```

  - You can scale out the index by creating identical indexes (load balanced)

```
CREATE INDEX i1 … WITH {“nodes”:”node1”};
CREATE INDEX i2 … WITH {“nodes”:”node2”};
```



HP

GSI Placement and Partitioning

Partition Indexes Manually with WHERE clause

- You can partition with the WHERE clause and place on various nodes for scaling out

```
CREATE INDEX i1 … WHERE zipcode between “94000” and “94999” …
CREATE INDEX i2 … WHERE zipcode between “95000” and “96000” …
```

GSI Availability and Rebalance

HP

GSI Availability and Rebalance

Use multiple identical indexes for availability

- GSI Availability
  - Create multiple identical indexes on separate nodes for availability
  - GSI will auto divert traffic if any copy goes down

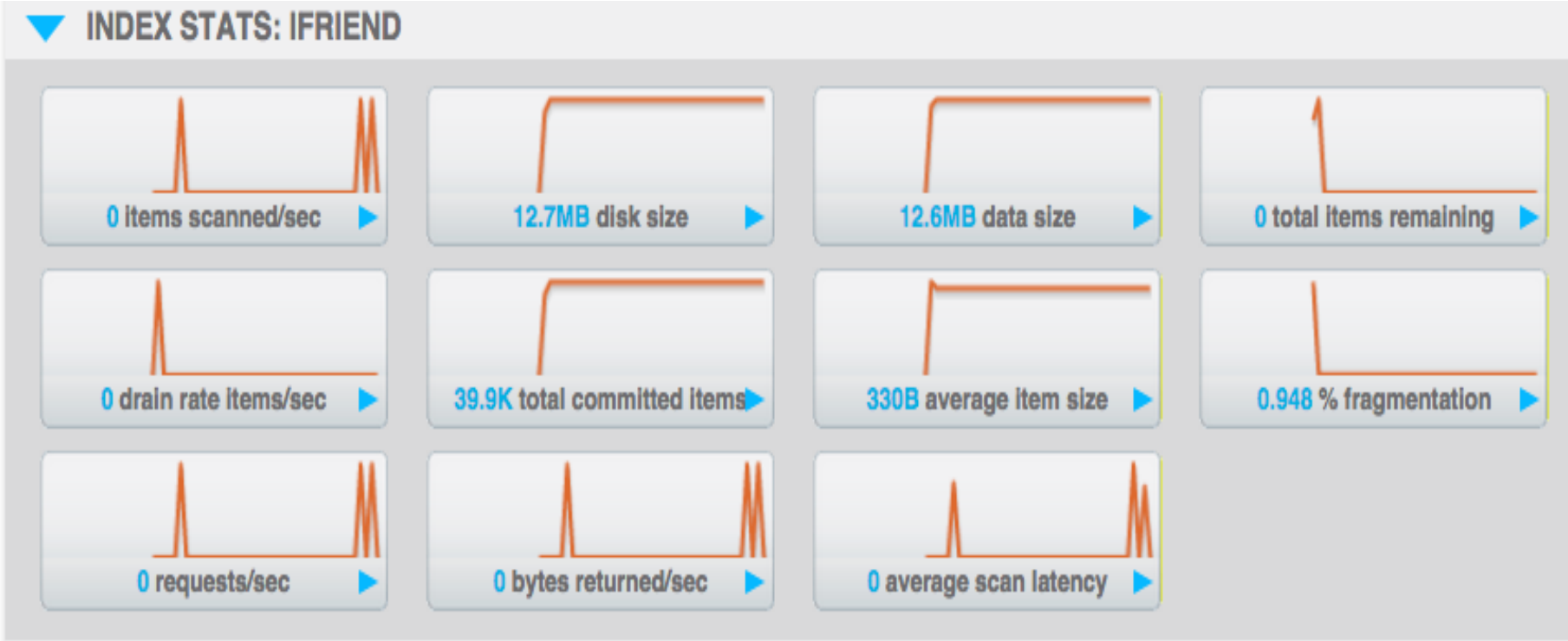
- GSI & Rebalance
  - Removing/Failing a node with index service, remove the GSI indexes on that node
  - Adding a node with index service, won't automatically move some indexes to the node.

Monitoring GSI

HP

Monitoring GSI Indexes

- Index Size and Maintenance Stats
- Index Scan Stats



HP

Complex Types and GSI

- Indexing Complex Types
  - Sub-documents attributes

```
CREATE INDEX ifriend_id ON default(friends.id)
USING GSI;

SELECT * FROM default
WHERE friends.id= "002819”;
```

```
{
 "id": "00000000000001",
 "desc": ”---",
 "type": "friends",
 "tags": [0,1,2,3,4,5,6,7,8,9],
 "friends": {
   "id": "002819",
   "class": "005"
  }
}
```

HP

Complex Types and GSI

- Indexing Complex Types
  - Compound Keys

```
CREATE INDEX ifriends_id_class ON
default(friends.id, friends.class) USING GSI;

SELECT * FROM default
WHERE friends.id="002819" and friends.class="005”;
```

```
{
 "id": "00000000000001",
 "desc": ”---",
 "type": "friends",
 "tags": [0,1,2,3,4,5,6,7,8,9],
 "friends": {
   "id": "002819",
   "class": "005"
  }
}
```

HP

Complex Types and GSI

- Indexing Complex Types
  - Sub-documents

```
CREATE INDEX ifriend ON default(friends) USING GSI;

SELECT * FROM default
WHERE friends= {"class": "005","id":"002819"};
```

```
{
 "id": "00000000000001",
 "desc": ”---",
 "type": "friends",
 "tags": [0,1,2,3,4,5,6,7,8,9],
 "friends": {
   "id": "002819",
   "class": "005"
  }
}
```

Labs: CREATE INDEX