Elastic Query Processing on Function as a Service Platforms

Thomas Bodner

VLDB 2020 PhD Workshop
31 August 2020
Target Analytics Workloads

- Interactive ad-hoc analytics on cold data
  - Interactive – require query latencies in seconds
  - Ad-hoc
    - Every query is different
    - Infrequent query bursts
  - Cold data – infrequently accessed data
  - Example: Bring ERP and HCM data together to answer questions like "For every cost center, how has the revenue per employee changed over time?"
Target Analytics Workloads

• Interactive ad-hoc analytics on cold data
  • Interactive – require query latencies in seconds
  • Ad-hoc
    • Every query is different
    • Infrequent query bursts
  • Cold data – infrequently accessed data
  • Example: Bring ERP and HCM data together to answer questions like „For every cost center, how has the revenue per employee changed over time?“

• These dynamic workloads are hard to predict
  • Difficult to provision infrastructure and to optimize query execution upfront
  • Difficult to achieve adequate performance and cost efficiency
Target Analytics Workloads

• Interactive ad-hoc analytics on cold data
  • Interactive – require query latencies in seconds
  • Ad-hoc
    • Every query is different
    • Infrequent query bursts
  • Cold data – infrequently accessed data
  • Example: Bring ERP and HCM data together to answer questions like „For every cost center, how has the revenue per employee changed over time?“

• These dynamic workloads are hard to predict
  • Difficult to provision infrastructure and to optimize query execution upfront
  • Difficult to achieve adequate performance and cost efficiency

▷ Database systems need to adapt to them quickly
Traditional Database Architectures

• Shared-everything
  • Limited compute scalability
  • Storage scalability via data tiering to larger/cheaper/slower storage until too slow

• Shared-nothing
  • Expensive data shuffles and loads on workload changes
  • Non-interactive performance during transition periods

• Shared-disk (with regular VMs)
  • Separate compute and storage resources, matching modern cloud infrastructures
  • Compute scalability via adding/removing nodes
  • Configuring and launching VMs takes minutes at best, and cannot be part of interactive query response
Traditional Database Architectures

• Shared-everything
  • Limited compute scalability
  • Storage scalability via data tiering to larger/cheaper/slower storage until too slow

• Shared-nothing
  • Expensive data shuffles and loads on workload changes
  • Non-interactive performance during transition periods

• Shared-disk (with regular VMs)
  • Separate compute and storage resources, matching modern cloud infrastructures
  • Compute scalability via adding/removing nodes
  • Configuring and launching VMs takes minutes at best, and cannot be part of interactive query response

▸ Current approaches do not scale fast enough and are prone to under- or over-provisioning
Function as a Service Platforms

• Allocate and bill fine-grained units of compute resources that launch in milliseconds
Function as a Service Platforms

- Allocate and bill fine-grained units of compute resources that launch in milliseconds
  - Allocation fast enough to be part of interactive query response
Function as a Service Platforms

• Allocate and bill fine-grained units of compute resources that spawn in milliseconds
  ▸ Allocation fast enough to be part of interactive query response

• Let users write pieces of code in (almost) any programming language

• Run user code on tiny, short-lived, and stateless workers

• Transparently schedule, load balance, and scale user code across 10,000s of workers
Function as a Service Platforms

• Allocate and bill fine-grained units of compute resources that spawn in milliseconds
  ▸ Allocation fast enough to be part of interactive query response

• Let users write pieces of code in (almost) any programming language

• Run user code on tiny, short-lived, and stateless workers

• Transparently schedule, load balance, and scale user code across 10,000s of workers
  ▸ Combined performance high enough for large-scale query processing
Function as a Service Platforms

• Allocate and bill fine-grained units of compute resources that spawn in milliseconds
  ▸ Allocation fast enough to be part of interactive query response

• Let users write pieces of code in (almost) any programming language

• Run user code on tiny, short-lived, and stateless workers

• Transparently schedule, load balance, and scale user code across 10,000s of workers
  ▸ Combined performance high enough for large-scale query processing

• Economically viable for users, when moderately utilized (2-8X the costs of VMs)
Challenges in FaaS-based Query Execution: Cloud Functions

1. Tight resource limits (2 vCPUs, 3GB RAM and 15min runtime)

2. Launch overheads (potentially 10s of seconds)
   - Invocation via web-based REST API
   - Initialization including host provisioning, worker placement and runtime setup

3. Observability for blackbox cloud function services

4. Fault tolerance via transparent re-execution

5. Indirect communication due to disabled inbound network connections
Challenges in FaaS-based Query Execution: Object Storage

6. Inefficiencies
   • High request latencies
   • Significant per-request costs

7. Weak data consistency guarantees
   • No read-your-own-write
   • No multi-key write
8. Cost-awareness
   • Cloud service pricing models
   • Cost-performance tradeoffs

9. Parallel plans
   • Exploit parallelism of underlying platform
   • Avoid data shuffles
# Cloud Data Analysis Systems

<table>
<thead>
<tr>
<th>FaaS-based Data Analysis Systems</th>
<th>Disaggregated Storage</th>
<th>FaaS-based Compute</th>
<th>Relational OLAP</th>
<th>Query Cost-Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>PyWren</td>
<td>✓</td>
<td>✓</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>Flint</td>
<td>✓</td>
<td>✓</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>Locus</td>
<td>✓</td>
<td>✓</td>
<td>×</td>
<td>✓</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Cloud-based OLAP Database Systems</th>
<th>Disaggregated Storage</th>
<th>FaaS-based Compute</th>
<th>Relational OLAP</th>
<th>Query Cost-Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amazon Redshift</td>
<td>×</td>
<td>×</td>
<td>✓</td>
<td>×</td>
</tr>
<tr>
<td>Redshift Spectrum</td>
<td>✓</td>
<td>×*</td>
<td>✓</td>
<td>×</td>
</tr>
<tr>
<td>Snowflake</td>
<td>✓</td>
<td>×</td>
<td>✓</td>
<td>×</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>FaaS-based OLAP Database Systems</th>
<th>Disaggregated Storage</th>
<th>FaaS-based Compute</th>
<th>Relational OLAP</th>
<th>Query Cost-Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lambada</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Starling</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Skyrise</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>
Skyrise Target Architecture

- FaaS-based, shared-disk architecture
- Coordinator compiles SQL queries to optimized plans
- Optimization incorporates statistics and prices
- Coordinator schedules operators on function service
- Coordinator observes operator execution
- Operators interact with storage service
- Build on AWS cloud services
Skyrise Query Engine

- Query operators
  - C++ for efficient resource management
  - Minimal deployment package for fast launches
  - Idempotence for correct behavior under failure

- Scheduler
  - Parallel function invocation
  - Function pre-warming
Skyrise Query Engine II

- Operator communication
  - Operator collocation
  - Interleaved and late materialization

- Access to Persistent and Intermediate Data
  - Columnar and compressed file formats
  - Statistics-based pruning
  - Wait for convergence of eventual consistent storage
  - Metadata layer for MVCC
Skyrise Query Optimizer

• FaaS-based execution: Limits and degrees of freedom

• Query cost-performance: Multi-objective optimization

• Parallel plans: Maximize parallelism and minimize data exchange

• Parallel optimization: Cope with complex cost function and large search space
Conclusion

• Interactive ad-hoc analytics on cold data require elastic query processing capabilities

• Modern cloud infrastructure (i.e., FaaS platforms) represents a promising foundation

• We identify challenges of building a query processing system on FaaS platforms

• We propose approaches to address these challenges

• We report on our progress towards building these concepts into our research prototype

• We further provide an outlook of what is still planned in this thread of research
Thank you. Questions?