Building an Elastic Query Engine on Disaggregated Storage

Midhul	Justin	Rachit	Dan	Ashish	Thierry
Traditional Shared-Nothing Architectures
Traditional Shared-Nothing Architectures

- Data partitioned across servers
- Each server handles its own partition
Traditional Shared-Nothing Architectures

- Data partitioned across servers
- Each server handles its own partition

Workload A

50x 🗄️ 20x 🟡
Traditional Shared-Nothing Architectures

- Data partitioned across servers
- Each server handles its own partition

Workload A

50x 20x ➔ 25 VMs

80x wasted!
Traditional Shared-Nothing Architectures

- Data partitioned across servers
- Each server handles its own partition

**Workload A**
- 50x
- 20x
- 25 VMs
- 80% wasted!

**Workload B**
- 5x
- 24x
Traditional Shared-Nothing Architectures

- Data partitioned across servers
- Each server handles its own partition

**Workload A**
- 50x
- 20x
- 25 VMs
- 80x wasted!

**Workload B**
- 5x
- 24x
- 6 VMs
- 7x wasted!
Traditional Shared-Nothing Architectures

- Data partitioned across servers
- Each server handles its own partition

**Workload A**
- 50% CPU
- 20% Memory
- 25 VMs
- 80% wasted!

**Workload B**
- 5% CPU
- 24% Memory
- 6 VMs
- 7% wasted!

Hardware-workload mismatch!
Traditional Shared-Nothing Architectures

- Data partitioned across servers
- Each server handles its own partition

Hardware-workload mismatch!
Traditional Shared-Nothing Architectures

- Data partitioned across servers
- Each server handles its own partition

Hardware-workload mismatch!
Traditional Shared-Nothing Architectures

- Data partitioned across servers
- Each server handles its own partition

Hardware-workload mismatch!

Data re-shuffle during elasticity!
Traditional Shared-Nothing Architectures

- Data partitioned across servers
- Each server handles its own partition

Hardware-workload mismatch!

Data re-shuffle during elasticity!

Fundamental issue in shared-nothing architectures: **Tight coupling of compute & storage**
Query Engines on Disaggregated Storage

- Decouple compute and persistent storage
- Independent scaling of resources
Query Engines on Disaggregated Storage

- Decouple compute and persistent storage
- Independent scaling of resources
Query Engines on Disaggregated Storage

- Decouple compute and persistent storage
- Independent scaling of resources
Query Engines on Disaggregated Storage

- Decouple compute and persistent storage
- Independent scaling of resources

Design aspects    Data-driven insights    Future Directions
Query Engines on Disaggregated Storage

- Decouple compute and persistent storage
- Independent scaling of resources

Design aspects
- Data-driven insights
- Future Directions

- Warehousing as a service
- In production for over 5 years
- 1000s of customers, millions of queries / day
Query Engines on Disaggregated Storage

- Decouple compute and persistent storage
- Independent scaling of resources

Design aspects

Data-driven insights

Future Directions

- Warehousing as a service
- In production for over 5 years
- 1000s of customers, millions of queries / day

Statistics from 70 million queries over 14 day period
Diversity of Queries
Diversity of Queries

- Read-Only -> 28%
Diversity of Queries

- Read-Only -> 28%
- Write-Only -> 13%
Diversity of Queries

- Read-Only -> 28%
- Write-Only -> 13%
- Read-Write -> 59%
Diversity of Queries

- Read-Only -> 28%
- Write-Only -> 13%
- Read-Write -> 59%

Three distinct query classes

Persistent data read/written varies over several orders of magnitude within each class
Query distribution over time

Read-Only query load varies significantly over time
High-level architecture

Virtual Warehouse
- Abstraction for computational resources
- Under the hood -> Set of VMs
- Distributed execution of queries
High-level architecture

Virtual Warehouse 1

Virtual Warehouse 2
High-level architecture

Virtual Warehouse 1

Persistent Data

Persistent Storage (S3)

Virtual Warehouse 2
High-level architecture

Persistent Data

Intermediate Data

Virtual Warehouse 1

Virtual Warehouse 2

Persistent Storage (S3)
High-level architecture

Persistent Storage (S3)

Virtual Warehouse 1
- Persistent Data
- Intermediate Data
- Ephemeral Storage

Virtual Warehouse 2
- Ephemeral Storage

Persistent Data

Intermediate Data
Ephemeral Storage System

Key Features

Intermediate Data

Persistent Storage (S3)
Ephemeral Storage System

Intermediate Data

Key Features

- Co-located with compute in VWs
Ephemeral Storage System

Key Features

- Co-located with compute in VWs
Ephemeral Storage System

Key Features

- Co-located with compute in VWs
- Opportunistic caching of persistent data
Ephemeral Storage System

Key Features

- Co-located with compute in VWs
- Opportunistic caching of persistent data
- Elasticity without data re-shuffle

Intermediate Data

Persistent Storage (S3)
Ephemeral Storage System

Key Features

- Co-located with compute in VWs
- Opportunistic caching of persistent data
- Elasticity without data re-shuffle

Diagram:

Intermediate Data

- DRAM
- Local SSD

- DRAM
- Local SSD

- DRAM
- Local SSD

Persistent Storage (S3)
Intermediate Data Characteristics
Intermediate Data Characteristics

![Graph showing the relationship between total CPU time and intermediate data exchanged. The x-axis represents total CPU time in micro-seconds, ranging from $10^0$ to $10^{14}$. The y-axis represents intermediate data exchanged in bytes, ranging from $10^0$ to $10^{14}$. The graph is color-coded, with different colors indicating different data交换 quantities, ranging from $10^0$ to $10^6$. A vertical line is drawn at $10^{10}$ on the x-axis, indicating a specific threshold for CPU time.]
Intermediate Data Characteristics
Intermediate Data Characteristics

Intermediate data sizes -> variation over 5 orders of magnitude

Difficult to predict intermediate data sizes upfront

Decouple compute & ephemeral storage?
Persistent Data Caching
Persistent Data Caching

- Intermediate data volume -> Peak Average
- Opportunistic caching of persistent data in ephemeral storage system

Hides latency of S3 access
Persistent Data Caching

- Intermediate data volume -> Peak → Average
- Opportunistic caching of persistent data in ephemeral storage system

How to ensure consistency?

Hides latency of S3 access
Persistent Data Caching

- Intermediate data volume -> Peak Average
- Opportunistic caching of persistent data in ephemeral storage system

How to ensure consistency?

- Each file assigned to unique node
  - Consistent hashing
- Write-through caching
Persistent Data Caching

- Intermediate data volume -> Peak Average
- Opportunistic caching of persistent data in ephemeral storage system

How to ensure consistency?

- Each file assigned to unique node
  - Consistent hashing
- Write-through caching

Analysis, Future Directions in paper
Elasticity
Elasticity

• **Persistent storage** – easy, offloaded to S3
Elasticity

- **Persistent storage** – easy, offloaded to S3
- **Compute** – easy, **pre-warmed pool of VMs**
Elasticity

- **Persistent storage** – easy, offloaded to S3
- **Compute** – easy, *pre-warmed pool of VMs*
- **Ephemeral storage** – challenging, due to co-location with compute
  
  Back to shared-nothing architecture problem (data re-shuffle)
Elasticity

- **Persistent storage** – easy, offloaded to S3
- **Compute** – easy, **pre-warmed pool of VMs**
- **Ephemeral storage** – challenging, due to co-location with compute
  
  Back to shared-nothing architecture problem (data re-shuffle)
Elasticity

- **Persistent storage** – easy, offloaded to S3
- **Compute** – easy, **pre-warmed pool of VMs**
- **Ephemeral storage** – challenging, due to co-location with compute

Back to shared-nothing architecture problem (data re-shuffle)
Elasticity

- **Persistent storage** – easy, offloaded to S3
- **Compute** – easy, *pre-warmed pool of VMs*
- **Ephemeral storage** – challenging, due to co-location with compute
  
  Back to shared-nothing architecture problem (data re-shuffle)
Lazy Consistent Hashing

Scheduler

Node 1

Node 2

Node 3

Persistent Storage (S3)
Lazy Consistent Hashing

Locality aware task scheduling

Scheduler

Node 1

Node 2

Node 3

Persistent Storage (S3)
Lazy Consistent Hashing

Locality aware task scheduling

Scheduler

Node 1  Node 2  Node 3

Persistent Storage (S3)
Lazy Consistent Hashing

Scheduler

Node 1

Node 2

Node 3

Persistent Storage (S3)
Lazy Consistent Hashing

Elasticity without data re-shuffle

Scheduler

Node 1
Node 2
Node 3

Persistent Storage (S3)
Lazy Consistent Hashing

Elasticity without data re-shuffle

Scheduler

Node 1
Node 2
Node 3
Node 4

Persistent Storage (S3)
Lazy Consistent Hashing

Elasticity without data re-shuffle

Scheduler

Node 1
Node 2
Node 3
Node 4

Persistent Storage (S3)
Lazy Consistent Hashing

Elasticity without data re-shuffle

Scheduler

Node 1

Node 2

Node 3

Node 4

Persistent Storage (S3)
Do customers exploit elasticity in the wild?
Do customers exploit elasticity in the wild?
Do customers exploit elasticity in the wild?

Sometimes by up to 100x

Resource scaling by up to 100x needed at times

20% warehouses exploit elasticity
At what time-scales are warehouses resized?
At what time-scales are warehouses resized?
At what time-scales are warehouses resized?
At what time-scales are warehouses resized?

Granularity of warehouse elasticity

>>

Changes in query load
At what time-scales are warehouses resized?

Granularity of warehouse elasticity

Changes in query load

Need finer-grained elasticity in order to better match demand
Resource Utilization
Resource Utilization

System-wide resource utilizations

Utilization

CPU  Memory  Network-Tx  Network-Rx
Significant room for improvement in resource utilizations
Resource Utilization

Virtual Warehouse abstraction
- Good performance isolation
- Trade-off: Low resource utilization

Significant room for improvement in resource utilizations
Resource Utilization

Virtual Warehouse abstraction
- Good performance isolation
- Trade-off: Low resource utilization

Solution #1
Finer-grained elasticity with current design

Solution #2
Move to resource shared model

Significant room for improvement in resource utilizations
Alternate design: Resource Sharing

Customer \(\xrightarrow{\text{per-hour}}\) snowflake \(\xrightarrow{\text{per-hour}}\) AWS
Alternate design: Resource Sharing

Customer → per-hour → snowflake → per-second → aws
Alternate design: Resource Sharing
Alternate design: Resource Sharing

Move to per-second pricing -> pre-warmed pool not cost effective
Alternate design: Resource Sharing

Move to per-second pricing -> pre-warmed pool not cost effective

Solution #1
Finer-grained elasticity with current design
Alternate design: Resource Sharing

Move to per-second pricing -> pre-warmed pool not cost effective

Solution #1
Finer-grained elasticity with current design

Solution #2
Move to resource shared model
Alternate design: Resource Sharing

Move to per-second pricing -> pre-warmed pool not cost effective

Solution #1
Finer-grained elasticity with current design

Solution #2
Move to resource shared model

Statistical Multiplexing
• Better resource utilization
• Helps support elasticity
Alternate design: Resource Sharing

Move to per-second pricing -> pre-warmed pool not cost effective

Solution #1
Finer-grained elasticity with current design

Solution #2
Move to resource shared model

Statistical Multiplexing
- Better resource utilization
- Helps support elasticity

For 30% of warehouses
Standard deviation $\geq$ Mean
Resource sharing challenges
Resource sharing challenges

- Challenges of moving to a resource shared architecture
  - Maintaining *isolation guarantees*
  - *Shared Ephemeral Storage System*
Resource sharing challenges

• Challenges of moving to a resource shared architecture
  • Maintaining isolation guarantees
  • Shared Ephemeral Storage System

Tenant A

Tenant B

Shared Warehouse
Resource sharing challenges

• Challenges of moving to a resource shared architecture
  • Maintaining isolation guarantees
  • Shared Ephemeral Storage System

• Sharing cache
  • No pre-determined lifetime
  • Co-existence with int. data
Resource sharing challenges

- Challenges of moving to a resource shared architecture
  - Maintaining isolation guarantees
  - Shared Ephemeral Storage System

- Sharing cache
  - No pre-determined lifetime
  - Co-existence with int. data

- Elasticity without violating isolation
  - Possible cross-tenant interference
  - Need private address-spaces for tenants
Resource sharing challenges

• Challenges of moving to a resource shared architecture
  • Maintaining **isolation guarantees**
  • **Shared Ephemeral Storage System**

• Sharing cache
  • No pre-determined lifetime
  • Co-existence with int. data

• Elasticity without violating isolation
  • Possible cross-tenant interference
  • Need private address-spaces for tenants
Resource sharing challenges

- Challenges of moving to a resource shared architecture
  - Maintaining isolation guarantees
  - Shared Ephemeral Storage System

- Sharing cache
  - No pre-determined lifetime
  - Co-existence with int. data

- Elasticity without violating isolation
  - Possible cross-tenant interference
  - Need private address-spaces for tenants

[Diagram showing resource sharing between Tenant A and Tenant B, with a shared warehouse]
Conclusion

Design aspects  
Data-driven insights  
Future Directions
Conclusion

Design aspects  Data-driven insights  Future Directions
Conclusion

- Design aspects
- Data-driven insights
- Future Directions

• Dataset publicly released
  • https://github.com/resource-disaggregation/snowset
Thank You

Questions?