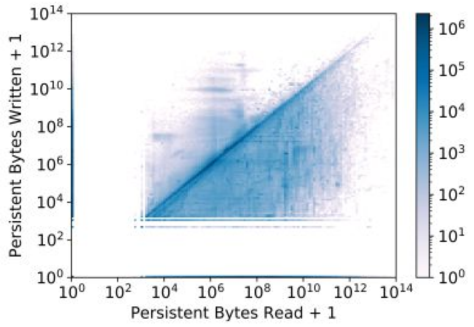
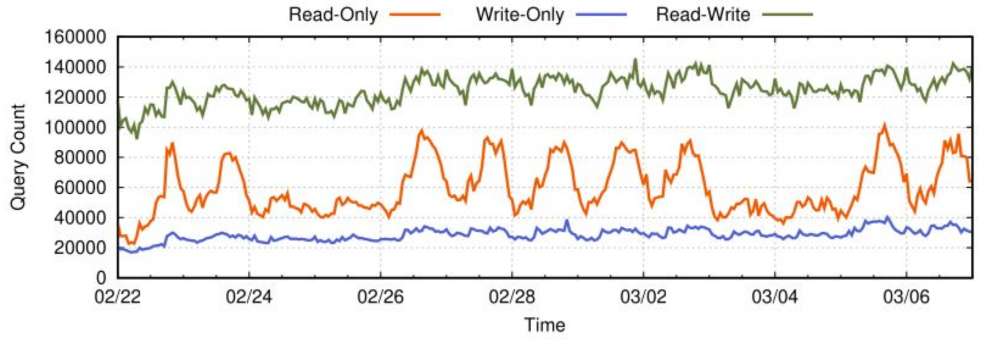
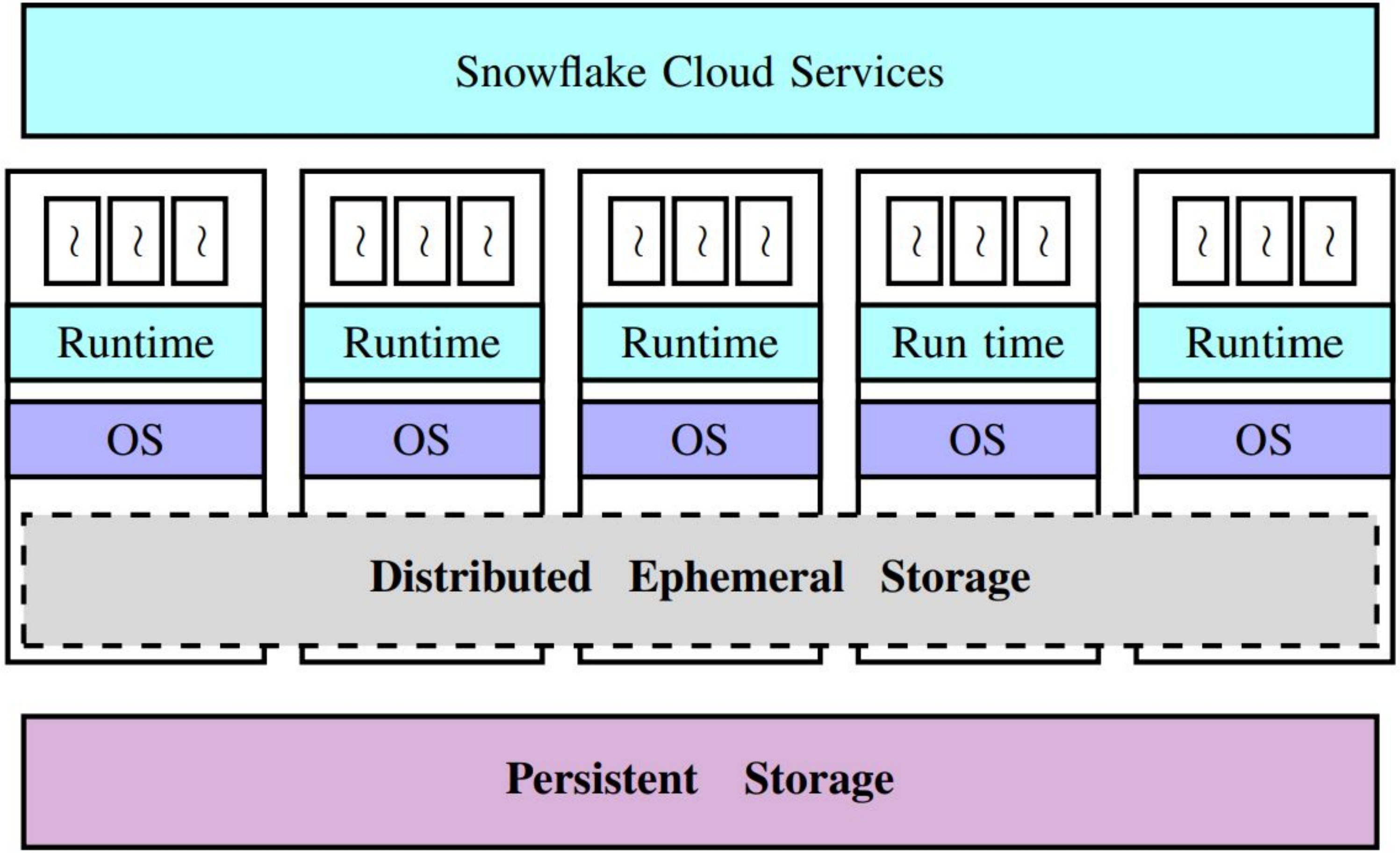
Building An Elastic Query Engine on Disaggregated Storage

- Performance numbers based on ~70 million queries over a period of 14 days.
- Explore potential aspects of the design that can be relooked.
    - Especially isolation and ephemeral storage design.





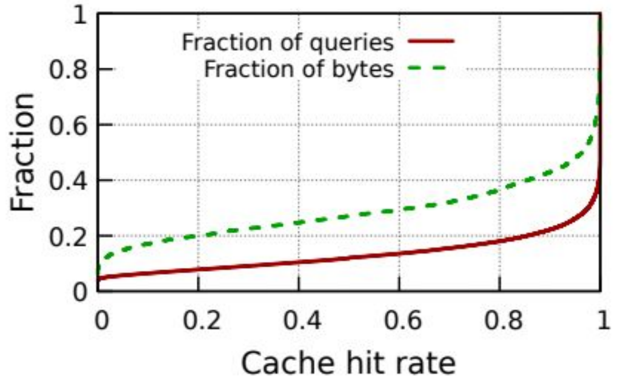
- ~28% of all customer queries were read only
- ~13% were write only (Data ingestion queries)
- ~59%, a majority were read-write queries (ETL workloads largely)


Snowflake Cloud Services
Runtime
Runtime
Runtime
Run time
Runtime
OS
OS
OS
OS
OS
Distributed Ephemeral Storage
Persistent Storage

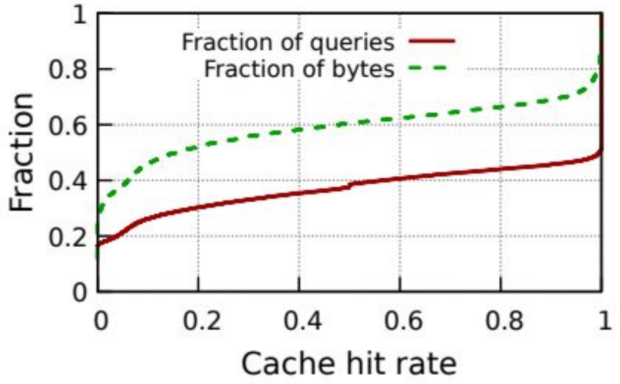
Ephemeral storage - cache access patterns

- Intermediate data has high peak sizes but a low average size.
- Persistent data access is highly skewed.
- LRU Cache replacement policy while prioritising intermediate data
- We can multiplex and get pretty good cache hit rates.

- Upto 80% hit rate for read only queries



- Upto 60% for read-write queries.

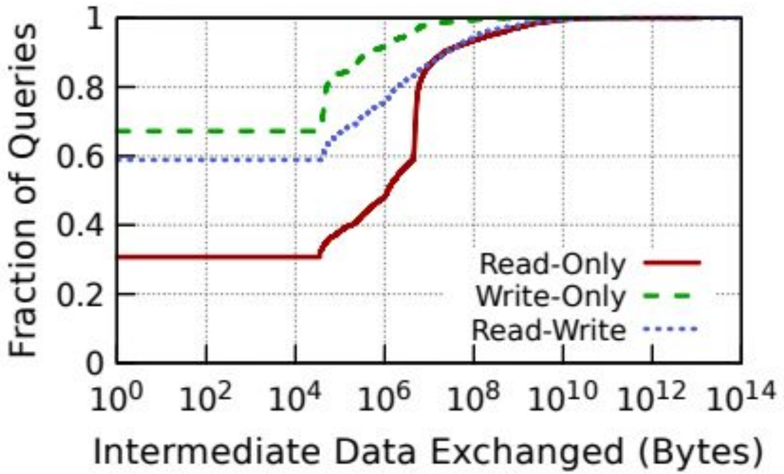


Scope for Improvement

- Improve cache eviction policies.
  - Can you do better than prioritising intermediate data.
- Infiniswap - Spill cache data to other nodes in the network via RDMA
  - Helpful in case of skew?

Do we fully have Storage Disaggregation?

- Intermediate data exchanged can vary by order of magnitudes upto 8.
- Hard to predict how much you will need

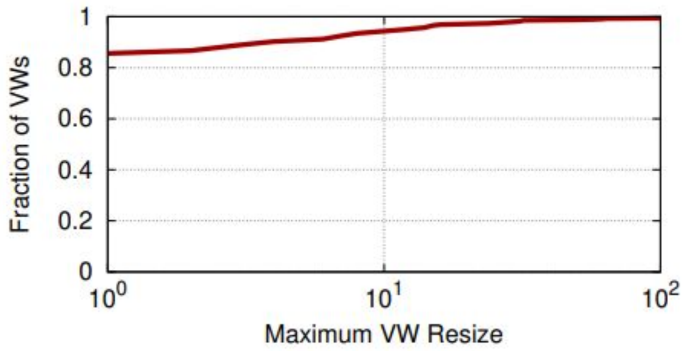


Solutions?

- Also disaggregate ephemeral storage.
- Should support fine grained elasticity to handle unpredictable Intermediate data.
- Existing work such as Pocket or Locus could provide a base to build upon.

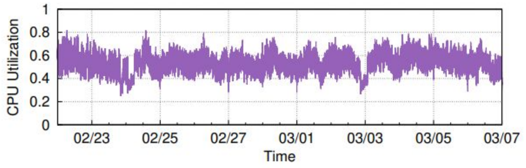
Elasticity

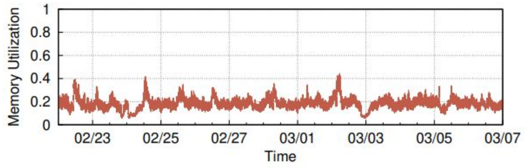
- Only 20% of Snowflake's customers utilized coarse grained scaling.
- Lack of visibility for when resizing the VW is necessary.
- Resource utilization can even vary within the lifetime of a query, but such fine grain elasticity is not supported.

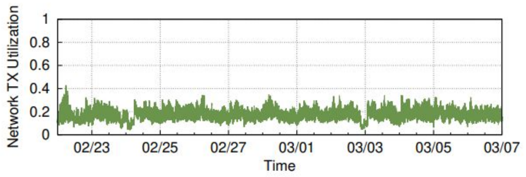


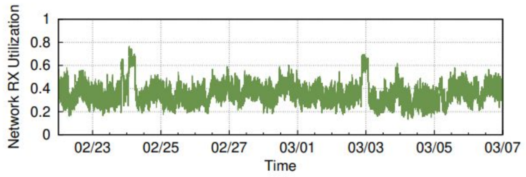
Utilization

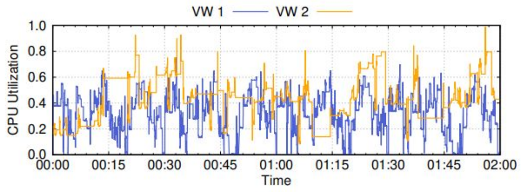
- Average CPU, Memory, Network TX and RX utilizations are roughly 51%, 19%, 11%, 32%, respectively.

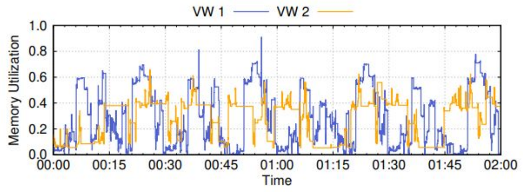






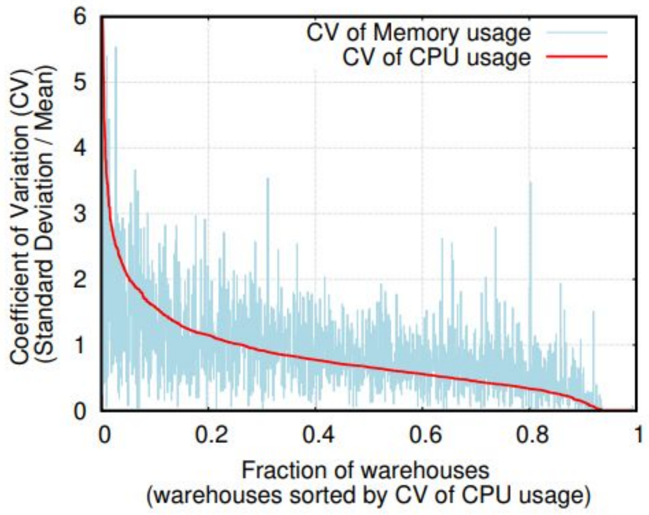






Utilization

- 30% of VWs have CPU utilization with the standard deviation greater than the mean.
- Isolated VWs can hence lead to underutilization.
- Additionally per second pricing makes them Financially inefficient.



Multi-Tenancy

- Improves utilization over isolated virtual warehouses, but you lose out on the trivial isolation guarantee.
- Can impact on cache policy and fairness.
    - Tasks running for a different user can affect cache hit rates for another user

Thoughts?