Aaron J. Elmore

UC Santa Barbara

A bit about me

Did my majority of PhD research in elasticity primitives for building a DBaaS

Zephyr: live migration

Pythia: automated tenant placement and crisis mitigation

Squall: live reconfiguration

A problem arises

Graduating soon and the ‘cloud’ is no longer

Thinking about OLTP

Transactional databases are king for those who know.

Recording valuable events, such as ad clicks, with consistency and speed.

We are talking:

High Performance Transaction Systems

Living in a real time world.

OLTP is great. Real value is derived from understanding what the transactions are saying.

Many scenarios where we want to react to events in near real time.

(Outlier detections, trend modeling, etc)

Current state of affairs

Separate and insulate OLTP system from analytics.

ETL from OLTP into analytic framework (OLAP, Hadoop, etc).

The extract is often batched to amortize the costs. This introduces latency.

Need one database to rule them all

How can we **integrate** analytics into OLTP?

Generalized solution possible for tightly coupled analytics and transactions?

What analytics do we support?

What does this architecture look like?

Back to the self managed elasticity

We have a fixed amount of resources, partitionable data, and transaction classes with performance SLOs.

Key thought: Flexibly share fixed resources between OLTP and analytics.

Leverage self-managed elasticity and replication to dynamically allocate resources to analytics.

Sharing is caring

Meeting OLTP SLOs = dedicate every last spare drop of resources to analytics.

SLOs start to slip = allocate resources away from analytics. Degrade results, older snapshots, etc on analytics.

Exploit semantics of limited operations (ala stored procedures) for analytics and transactions.

Requires

Good workload modeling

SLO generation

Self managed replication and elasticity

Approximate results (ala Aqua / BlinkDB)

Understand relationship of txns and analytics

Thank You