
# SCADS: Performance Safe Queries for Interactive Web Applications

HTPS

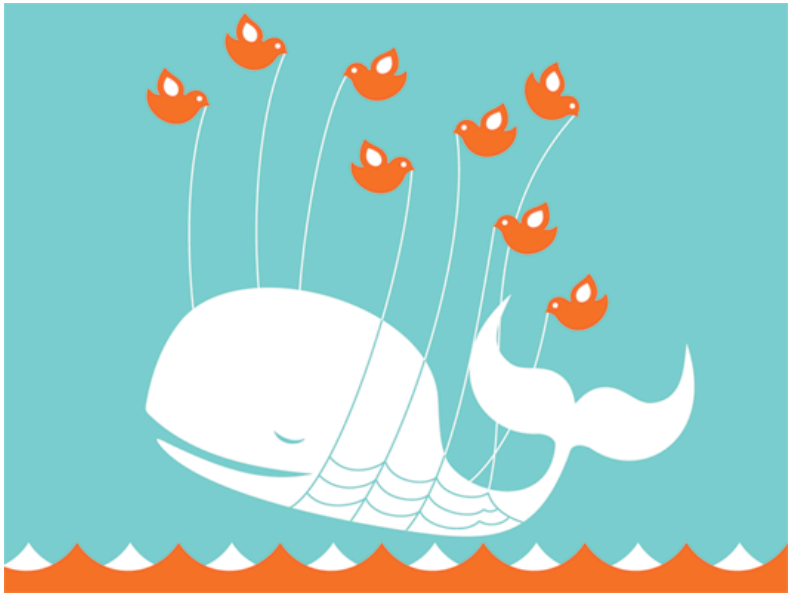
October 2009

Michael Armbrust, Armando Fox, Michael Franklin, David Patterson, Nick Lanham, Beth Trushkowsky, Jesse Trutna,

Stephen Tu

# Motivation

- Most popular websites follow the same pattern
  - Rapidly (one weekend) developed on a relational database
  - Become popular and realize scaling limitations
  - Build large, complicated ad-hoc systems to deal with scaling limitations as they arise
- Websites that can't scale fast enough lose customers




# NoSQL?

- Recently lots of buzz about key/value stores
  - Trivial scaling
  - Predictable performance
  - Great performance through relaxed consistency
- Some nonstandard data models (column families, etc)
- End up implementing many things by hand
  - Imperative queries
  - Indexes
  - Optimization
  - Parallelism
  - Session guarantees


# SCADS: YeSQL

- Developers specify ahead of time:
  - Entities
  - Relationships and cardinalities
  - Queries
- Constraints
  - All queries are specified ahead of time
  - Only allow equ-joins over pre-specified relationships with fixed cardinalities
  - Require that all intermediate steps and final results are bounded


# Implementation

- SCADS compiles this to a library that allows them to interact with the underlying key/value store
  - entities -> classes
  - queries -> methods
- Queries that return an unbounded number of results can either return top-k or use efficient pagination


# Benefits

- Guaranteed scaling for all queries
- Automatic index selection and maintenance
- Library ensures eventual consistency in the face of failures
- Automatic parallelization of queries
- Session guarantees on a per query basis


# Where are we going?

- Status
  - We have a simple heuristic optimizer that can answer many queries
  - We hope to be able to implement Twitter /Facebook / E-Bay by December
- Future Work
  - Tunable consistency per query
  - Predicting performance
  - Aggregates


# Questions?


# Declaring Entities

```
ENTITY user
{
   string name,
   string password,
   string email,
   string profileData
   PRIMARY(name)
}

ENTITY subscription
{
   bool approved
   PRIMARY(following,
   target)
}
```

```
ENTITY thought
{
   date time,
   string thought
   PRIMARY(owner,
   time)
}

ENTITY topic
{
   string name
   PRIMARY(reference,
   name)
}
```


# Relationships

```
RELATIONSHIP owner FROM user TO MANY thought
RELATIONSHIP following FROM user TO 5000
   subscription
RELATIONSHIP target FROM subscription TO ONE user
RELATIONSHIP hashtag FROM thought TO 10 topic
RELATIONSHIP references FROM thought TO 10 user
```


# Queries

```
QUERY userByName
FETCH user
WHERE user.name = [1:name]


QUERY myThoughts
FETCH thought
   OF user BY owner
WHERE user=[this]
PAGINATE [1: numperpage] MAX 10
```


# Complicated Queries

```
QUERY thoughtstream
FETCH thought
  OF user friend BY owner
  OF subscription BY target
  OF user me BY following
WHERE me=[this] AND
  subscription.approved = true
ORDER BY timestamp
LIMIT [1:count] MAX 100
```