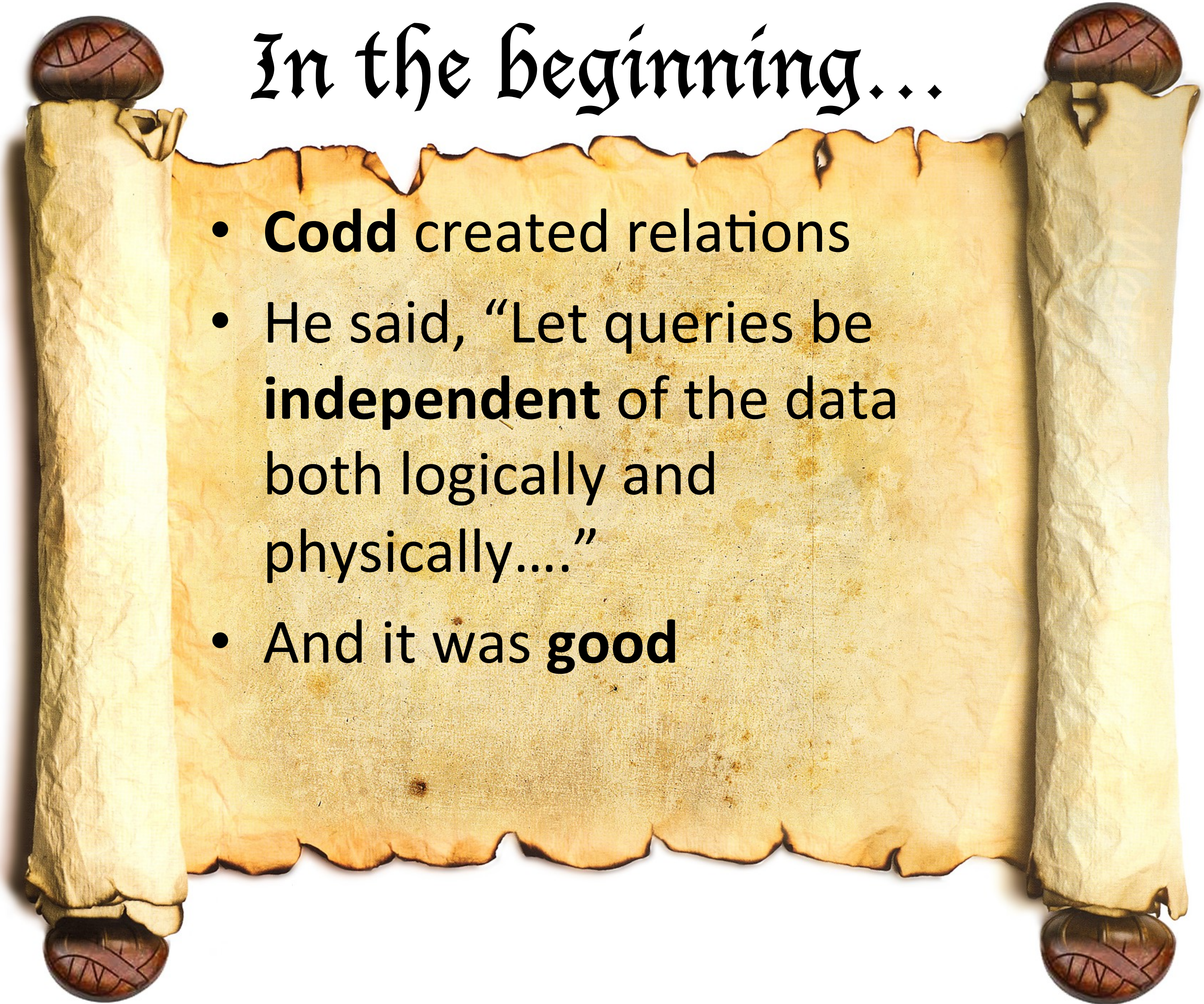
amplab
UC BERKELEY

Bridging the NoSQL Gap with Scale Independence

Michael Armbrust, Kristal Curtis, Tim Kraska
Armando Fox, Mike Franklin, David Patterson

HPTS 2011 – Poster Session

In the beginning…



- **Codd** created relations
- He said, “Let queries be **independent** of the data both logically and physically….”
- And it was **good**

The Good Years

- Ingres begat NonStop, Sybase and Postgres
- Sybase begat SQL Server
- Postgres begat PostgreSQL and Illustra
- PostgreSQL begat Netezza and Greenplum
- and so on for many years…

Web2.0: The NoSQL Heresies

“SQL databases are fundamentally non-scalable, and there is no magical pixie dust that we, or anyone, can sprinkle on them to suddenly make them scale.”

-Adam Wiggins - Heroku

http://www.newlife.org/images/blog/20110211/goldencalf.jpg

The NoSQL 'Solution'

- Throw out declarative queries, let developers write get/put calls themselves.
- Trivial to reason about performance!

http://stravaganzastravaganza.blogspot.com/2011/06/some-interesting-word-origins.html

Common Ground?

http://www.flickr.com/photos/wolfgangstaudt/2252688630

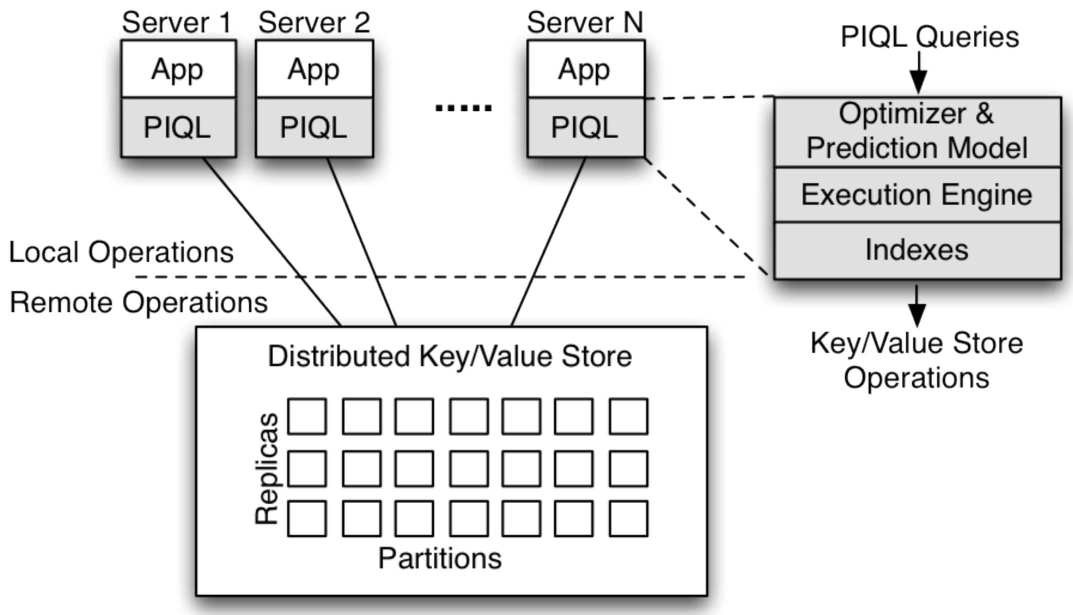
A New Kind of Data Independence

Scale Independence: Queries must perform a constant number of constant time operations ***independent of the size of the database***

http://www.flickr.com/photos/hopefoote/1394792726/

PIQL: A Performance Insightful Query Language

- Designed for **interactive web applications**
- Library leverages K/V stores for:
  - Predictable performance
  - Scalability
  - Consistency



Preserving Performance Predictability

- **Scale-Independent Optimization**
  - Choose query plans that bound the # storage ops **in the worst case**
  - Automatically calculate required indexes / materialized views
- **Query Language Extensions**
  - Queries over unbounded amounts of data with PAGINATE and LIMIT
  - Relationship cardinality constraints
- **SLO Compliance Modeling**
  - Use performance models to predict query response time distribution

New Objective Function for Optimization

- Example: twitter subscriber intersection query

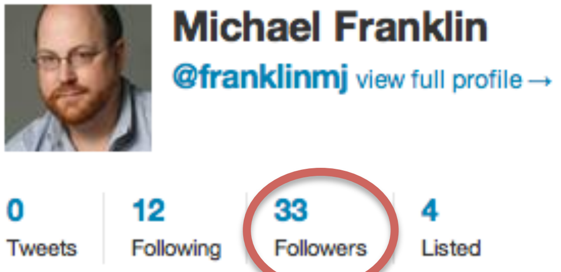
```
SELECT * FROM SUBSCRIPTIONS
WHERE target = <target user> AND
owner IN <friends of current user>
```

Cost-based optimizer: Average user has 126 followers. Do an index scan!

(even if smart engineers create a composite index)

Performance of Cost-based Plan

- Great for unpopular users:

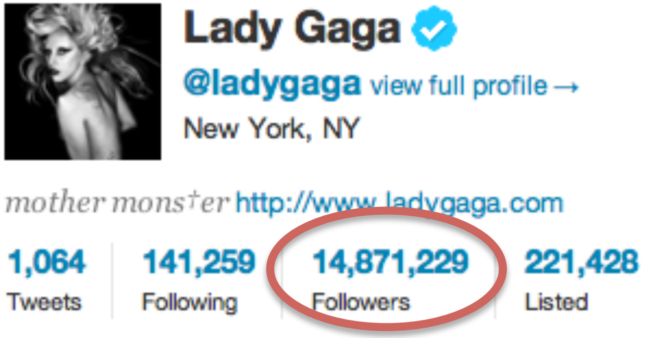


Michael Franklin

@franklinmj view full profile →

0	12	33	4
Tweets	Following	Followers	Listed

- Sometimes disastrous:



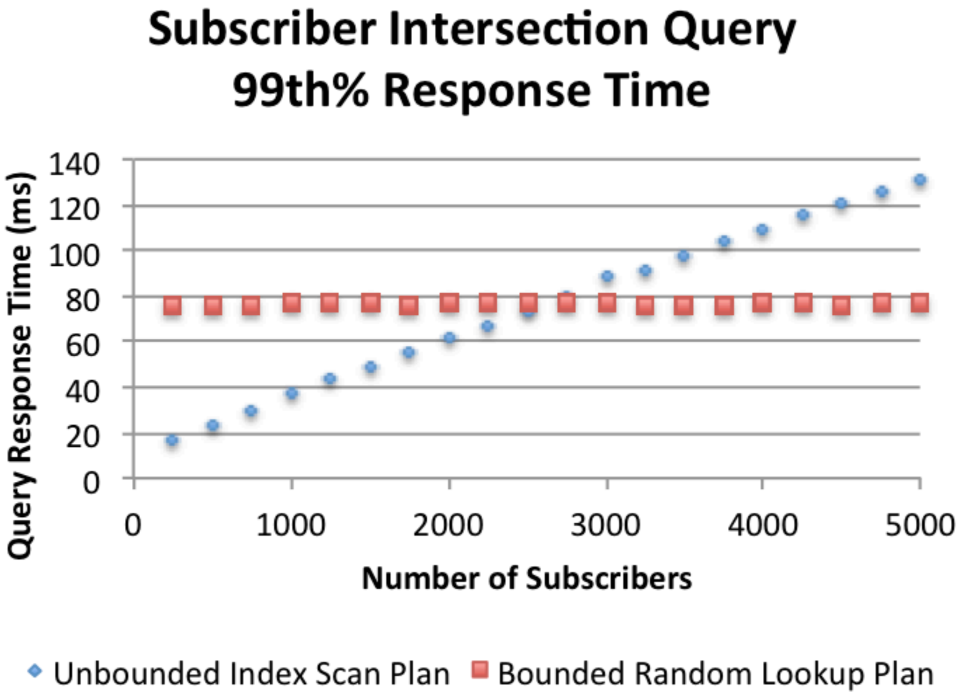
Lady Gaga

@ladygaga view full profile →

New York, NY

mother monster http://www.ladygaga.com

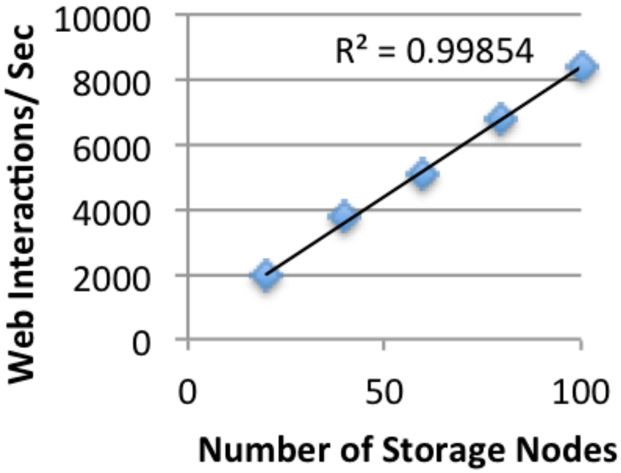
1,064	141,259	14,871,229	221,428
Tweets	Following	Followers	Listed

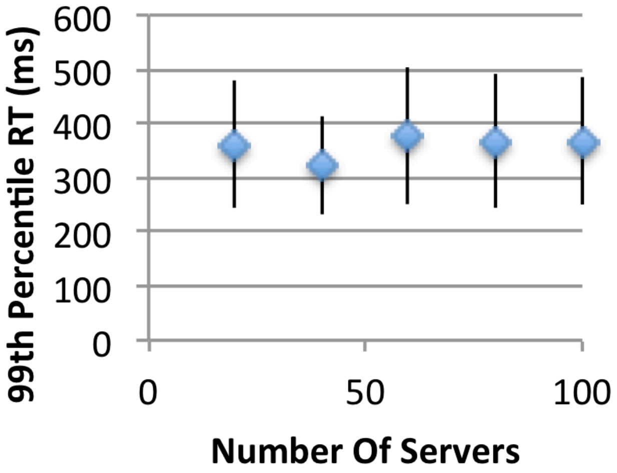


Subscriber Intersection Query 99th% Response Time

Project Status

- Prototype optimizer w/ simple secondary indexes [VLDB 12]
  - TPC-W scales linearly with 150+ machines
- Future Work: Leverage materialized views
  - Bound computation for incremental maintenance





Questions?

- Want to learn more?
  - amplab UC BERKELEY

    http://amplab.cs.berkeley.edu/blog
  - github SOCIAL CODING

    http://github.com/radlab/scads

Google SAP®

amazon web services™ ERICSSON imagination at work HUAWEI IBM intel®

Microsoft® NEC NetApp™ ORACLE® vmware®

cloudera MarkLogic® splunk>

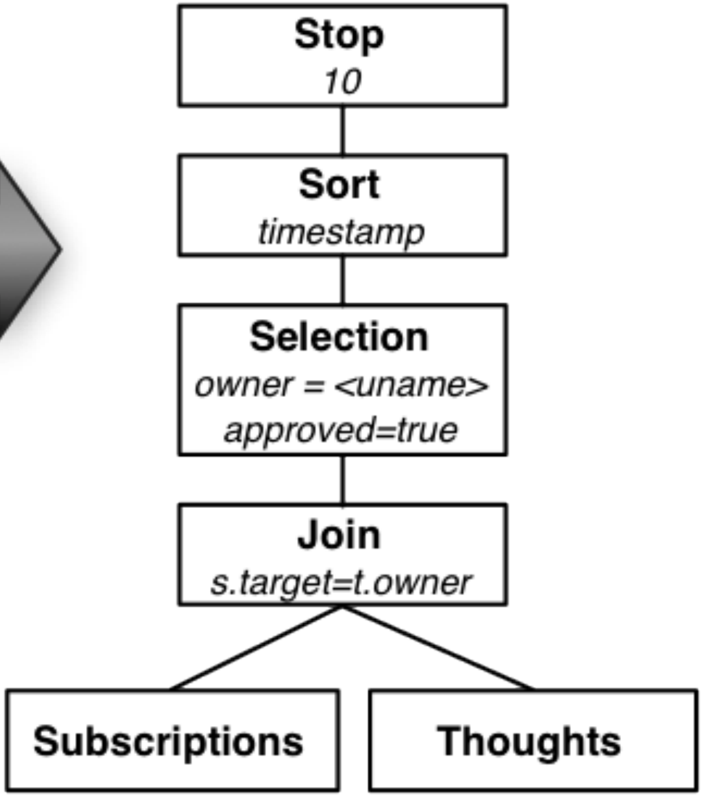
DARPA

Scale Independent Optimization

(a) Query

```
SELECT thoughts.*
FROM subscriptions s JOIN
   thoughts  t
WHERE t.owner = s.target
 AND s.owner = <uname>
 AND s.approved = true
ORDER BY t.timestamp DESC
LIMIT 10
```

(b) Logical Query Plan



Stop
10

Sort
timestamp

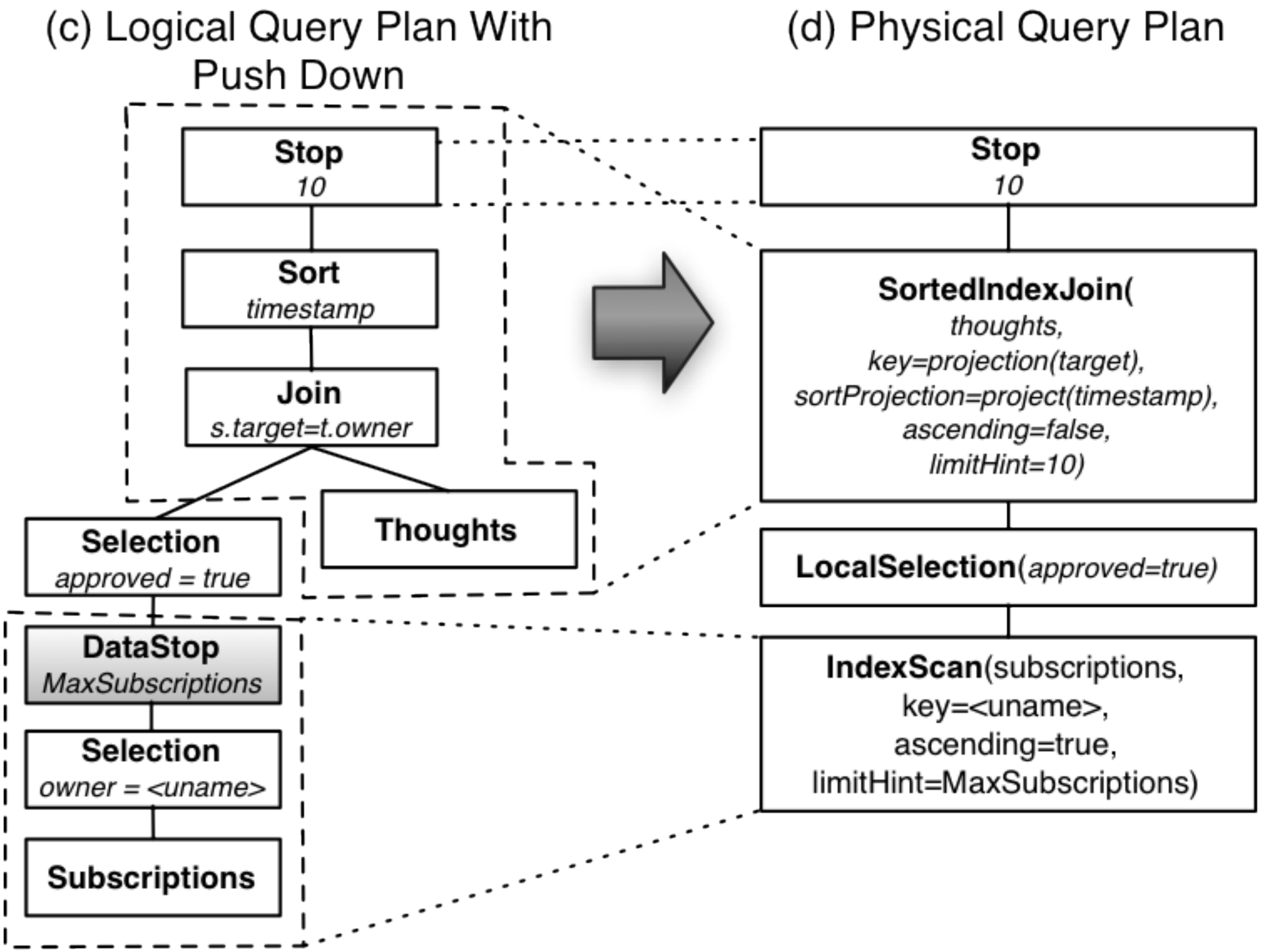
Selection
owner = <uname>
approved=true

Join
s.target=t.owner

Subscriptions

Thoughts

Scale Independent Optimization



(c) Logical Query Plan With Push Down

Stop *10*

Sort *timestamp*

Join *s.target=t.owner*

Selection *approved = true*

Thoughts

DataStop *MaxSubscriptions*

Selection *owner = <uname>*

Subscriptions

(d) Physical Query Plan

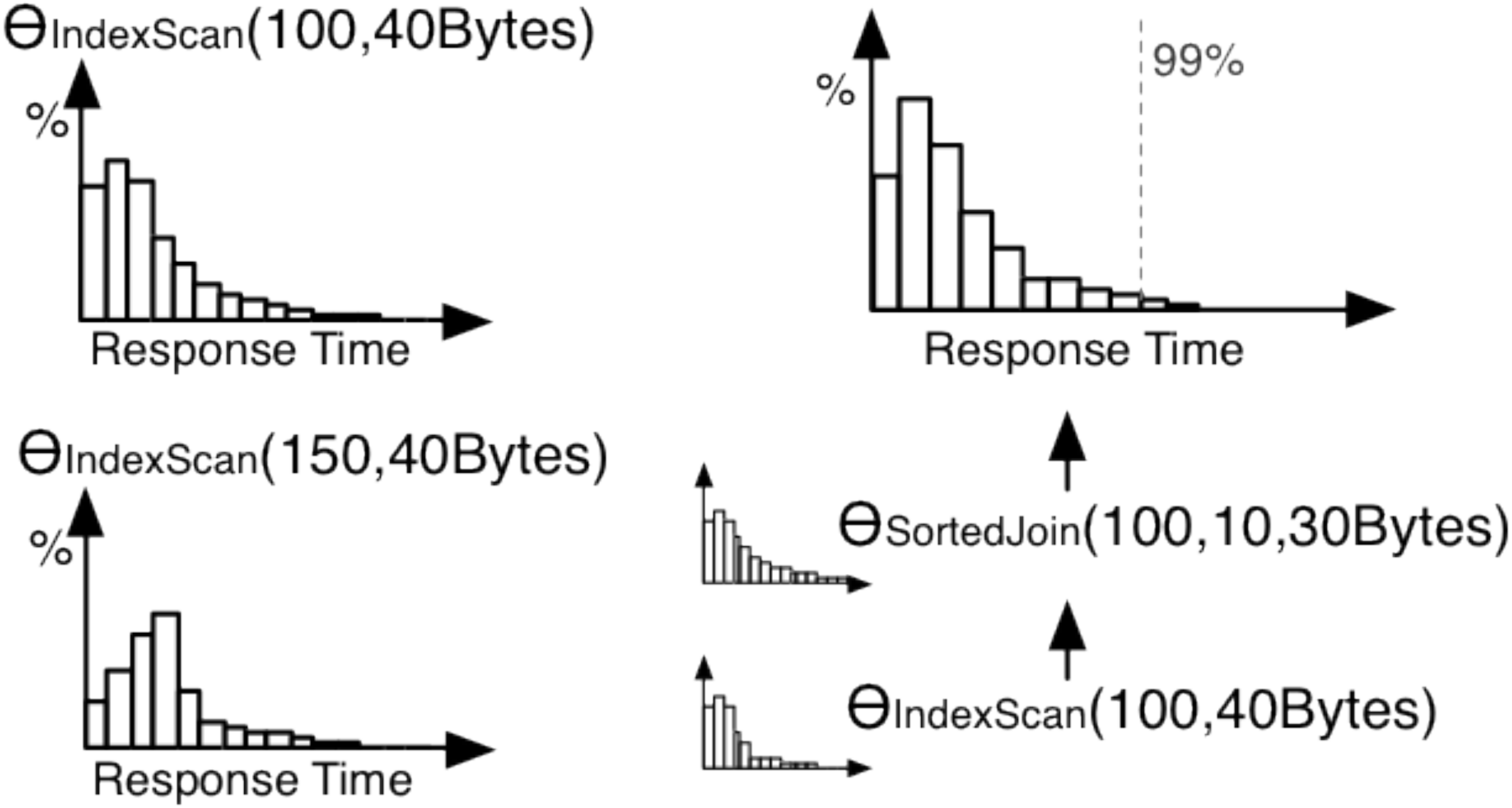
Stop *10*

SortedIndexJoin(*thoughts, key=projection(target), sortProjection=project(timestamp), ascending=false, limitHint=10)*

LocalSelection(*approved=true)*

IndexScan(subscriptions, key=<uname>, ascending=true, limitHint=MaxSubscriptions)

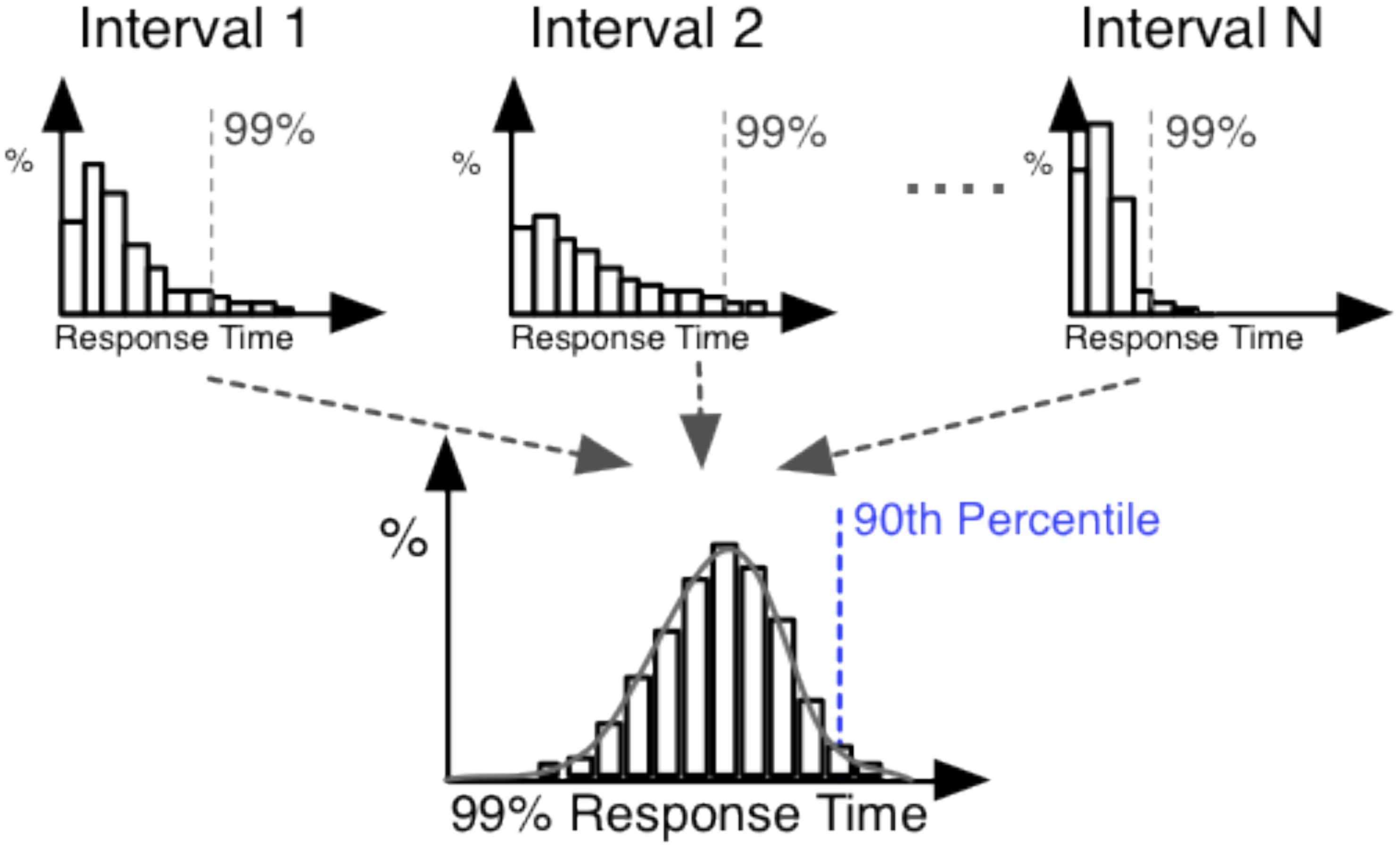
SLO Compliance Prediction



(a) Single Operator Prediction

(b) Query Plan Prediction (single interval)

SLO Compliance Prediction



(c) SLO Violation Risk