Adaptively Parallelizing Distributed Range Queries

Ymir Vigfusson, Adam Silberstein, Brain F. Cooper, Rodrigo Fonseca

VLDB ’09

Presented by Anjo Vahldiek
Environment
Environment
Motivation

* What is the right **level of parallelism**?
  * Parallelism too high → Waste of Performance
  * Parallelism too low → Slow acting Service

* How do we **assign** a query to the servers?
  * Which replica should execute one of the tasks?

➢ Difficult to optimize a priori
Engineering a new Database
  - Problem Description & Requirements
  - PNUTS: System Design & Architecture

Optimize Parallelism
  - Adaptive Server Allocation Algorithm
  - Server Scheduling

Evaluation
Agenda

- Engineering a new Database
  - Problem Description & Requirements
  - PNUTS: System Design & Architecture
- Optimize Parallelism
  - Adaptive Server Allocation Algorithm
  - Server Scheduling
- Evaluation
**Problem Description**

* **Range** Queries over *ordered* data
  * Time-ordered data
  * Secondary indexes
  * Hierarchical clustering

* **Example:**
  List all new products (within last 24h) under 10€ in Saarland.
* General distributed systems requirements (fault-tolerance, availability, scalability, geographic scope, response time, …)
* **Statistics-Free**
* Many concurrent clients
* Variety of request characteristics
* Clients with varying capabilities
Evaluation Criterion

- Time until **first result** (latency)
  - Stream of results
- Time to finish the whole query (run time)
* Engineering a new Database
  * Problem Description & Requirements
  * PNUTS: System Design & Architecture
* Optimize Parallelism
  * Adaptive Server Allocation Algorithm
  * Server Scheduling
* Evaluation
* Functions (API):
  * Get
    * Find record of primary key
  * Set
    * Set record of the primary key
  * Delete
    * Delete the record defined by the primary key
  * Getrange
    * Retrieve all records in primary key range passing an (optional) predicate
  * Per-record consistency
PNUTS: Architecture

- Clients
- Query routers
- Storage servers
- Partition controller
- Message broker
- Query router
  - Partition map
  - Scheduler
  - Scan engine
  - Query router
- Storage server
  - Query Processor
  - Update Processor
  - Storage
Executing Range Queries

1. getRange
   Query q

2. Pass q to
   Scan Engine

3. Lookup startq and
   endq partition and set
   of partitions Pq
Executing Range Queries (2)

4. Choose initial $K_q$ (level of parallelism)

5. $K_q$ scan $P_q$. Return set of $SP_q$ (pair of storage server & partition)
Executing Range Queries (3)

6. Forward scan requests of SPq

7. Scan Partitions

8. Return Results

Storage Server

Scan Engine

Client
Agenda

* Engineering a new Database
  * Problem Description & Requirements
  * PNUTS: System Design & Architecture
* Optimize Parallelism
  * Adaptive Server Allocation Algorithm
  * Server Scheduling
* Evaluation
Adaptively change the number of executing server

- Each server returns records at specific rate
- Selectivity of query
- Client consumption rate (bandwidth, processing capacity & current load)
Idealized Adaptive Server Allocation
Important Variables

* Kq: # of assigned servers to query q
* Cq: client consumption of query q
* Tc: static time interval (measuring)
1. Choose an initial number of servers $K_q = 1$
2. Measure client consumption rate $C_q(1)$ for $T_c$ seconds
3. Repeat
   1. Set $K_q(i+1) = K_q(i) + 1$
   2. Assign another server and begin scanning
   3. Measure client consumption rate $C_q(i+1)$ for $T_c$ seconds

   While $C_q(i+1) > C_q(i)$
4. Set $K_q$ to $K_q(i)$
3. Every $T_{flip}$ seconds, flip a coin

4. With probability $P^\uparrow$
   1. Set $K_q(i+1) = K_q(i) + 1$
   2. Allocate another server & start measure $C_q(i+1)$ for $T_c$
   3. If $C_q(i+1) > C_q(i)$, goto 4.1.
   4. Else $K_q(i+2) = K_q(i+1) - 1$
Allocation Algorithm: Adapting Kq

5. With probability $P_{↓}$
   1. Set $Kq(i+1) = Kq(i) - 1$
   2. Revoke a Server & start measure $Cq(i+1)$ for $Tc$
   3. If $Cq(i+1) \geq Cq(i)$, goto 5.1.
   4. Else $Kq(i+2) = Kq(i+1) + 1$
Idealized Adaptive Server Allocation

Find initial $K_q$

Adapting $K_q$ periodically
Agenda

- Engineering a new Database
  - Problem Description & Requirements
  - PNUTS: System Design & Architecture
- Optimize Parallelism
  - Adaptive Server Allocation Algorithm
  - Server Scheduling
- Evaluation
Server Scheduling

* Priority-Baised Round Robin
  * Gold vs. best-effort clients
    * Paid Clients will be preferred (external value)
  * Short/fast vs. long/slow queries
    * Minimum completion time

\[ PBRR = \sum_{q \in Q} \left( C + \frac{\sum_{i \in I(q)} length(i)^2}{P(q)^\alpha} \right) \]
1. For each $q$, insert $q$ into PQ with value $\frac{C + i^2 q}{P(q)^{\alpha}}$ 

2. While servers are idle or PQ is non-empty 
   1. Pop element $q^*$ with highest priority
   2. If $q^*$ saturated, set aside
   3. If $q^*$ cannot use idle server, set aside
   4. Else
      1. Grant $q^*$ one idle server
      2. Set $i_{q^*} = 0$
      3. Insert $q^*$ into PQ with value $\frac{C + i^2 q^*}{P(q^*)^{\alpha}}$
Agenda

* Engineering a new Database
  * Problem Description & Requirements
  * PNUTS: System Design & Architecture
* Optimize Parallelism
  * Adaptive Server Allocation Algorithm
  * Server Scheduling
* Evaluation
Evaluation

- Synthetic data (precise key distribution & selectivity)
- Metadata from Flickr (of 10 million photos)
- Special PNUTS client library measures client consumption rate
- Per-server concurrency Limit = 1
Fixed selectivity (10%) & client rate (10,000 KB/sec)

Fixed range size (1%) & client rate (250KB/sec)
Parallelism Impact & Adaptive Server Allocation

Fixed range size (1%) & selectivity (10%)
Adaptive Server Allocation

Find initial $K_q$  
Adapting $K_q$ periodically

![Graph showing time vs. $K$ and client speed vs. time with different $K$ values.](image)
Scheduling

- Avg. time till first results
- Avg. completion time
- Max. completion time
Scheduling (2)

![Graph showing KB Received vs Time (sec)]
Scheduling (3)
* **Simple** parallel database **design** for **web workload & scalability**
* Adopt parallelism **periodically** to client consumption rate
* Able to prioritize queries based on policies
Thanks for your Attention!

Any Questions?
Backup
Adaptive Server Allocation

![Graph showing time (sec) for K=5 and ASA policies for Slow Client and Fast Client.]

- Time (sec)
- Parallel Policy
- K=5
- ASA
- Slow Client
- Fast Client