PNUTS: Yahoo!’s Hosted Data Serving Platform


Yahoo! Research
Motivation

To design a distributed database for Yahoo!’s web applications, which is scalable, flexible, available.

(Colored animation slides borrowed from openly available presentation on PNUTS by Yahoo!)
What does Yahoo! need?

• Web applications demand
  – Scalability
  – Response time and Geographic scope
  – High Availability and Fault Tolerance

• Characteristic of Web traffic
  – Simple query needs
  – Manipulate one record at a time.
  – Relaxed Consistency Guarantees
PNUTS

Structure

Features

Data and Query Model

System Architecture

Data Storage and Retrieval

Consistency (Yahoo! Message Broker)

Recovery

Query Processing

Experiments

Future Work
Features

• Record-level operations
• Asynchronous operations
• Novel Consistency model
• Massively Parallel
• Geographically distributed
• Flexible access: Hashed or ordered, indexes, views; flexible schemas.
• Centrally managed
• Delivery of data management as hosted service.
PNUTS

<table>
<thead>
<tr>
<th>Structure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Features</td>
</tr>
<tr>
<td>Data and Query Model</td>
</tr>
<tr>
<td>System Architecture</td>
</tr>
<tr>
<td>Data Storage and Retrieval</td>
</tr>
<tr>
<td>Consistency (Yahoo! Message Broker)</td>
</tr>
<tr>
<td>Recovery</td>
</tr>
<tr>
<td>Query Processing</td>
</tr>
<tr>
<td>Experiments</td>
</tr>
<tr>
<td>Future Work</td>
</tr>
</tbody>
</table>
Data and Query Model

• Data representation
  – Table of records with attributes
  – Additional data types: Blob
• Flexible Schemas
• Point Access Vs Range Access
• Hash tables Vs Ordered tables
PNUTS

Structure

- Features
- Data and Query Model
- System Architecture
- Data Storage and Retrieval
- Consistency (Yahoo! Message Broker)
- Recovery
- Query Processing
- Experiments
- Future Work
System Architecture

- System is divided into regions.
- Each region contains a full complement of system components.
- Each region has a complete copy of each table.

- Tablets: Horizontal partitions of data tables.
- At least one tablet per region.)
PNUTS

Structure

- Features
- Data and Query Model
- System Architecture
  - Data Storage and Retrieval
    - Consistency (Yahoo! Message Broker)
  - Recovery
  - Query Processing
  - Experiments
  - Future Work
Data Storage and Retrieval

• Storage unit
  – Store tablets
  – Respond to get(), scan() and set() requests.

• Tablet controller
  – Owns the mapping
  – Routers poll periodically to get mapping updates.
  – Performs load-balancing and recovery
Data Storage and Retrieval

• Router
  – Determines which tablet contains the record
  – Determines which storage unit has the tablet
  – Interval mapping- Binary search of B+ tree.
  – Soft state
  – Tablet controller does not become bottleneck.
PNUTS

Structure

- Features
- Data and Query Model
- System Architecture
- Data Storage and Retrieval
- **Consistency (Yahoo! Message Broker)**
- Recovery
- Query Processing
- Experiments
- Future Work
Consistency model

- Per-record timeline consistency
  - All replicas apply updates to the record in same order.
  - Events (insert/update/delete) are for a particular primary key.
  - Reads return a consistent version from this timeline.
  - One of the replicas is the master. All updates forwarded to the master.
  - Replica master chosen based on the locality of majority of write requests.
  - Every Record has sequence number: Generation of record (New insert), Version of record (Update).

Related Question
Consistency model

• API Calls
  – Read-any
  – Read-critical (required version)
  – Read/latest
  – Write
  – Test-and-set-write (required version)
Yahoo! Message broker

- Topic based Publish/subscribe system
- Used for logging and replication
- PNUTS + YMB = Sherpa data services platform
- Data updates considered committed when published to YMB.
- Updates asynchronously propagated to different regions (post-publishing).
- Message purged after applied to all replicas.
- Per-record mastership mechanism.
Yahoo! Message broker

• Mastership is assigned on a record-by-record basis.
• All requests directed to master.
• Different records in same table can be mastered in different clusters.
• Basis: Write requests locality
• Record stores its master as metadata.
• Tablet master for primary key constraints
  – Multiple values for primary keys.

Related Question (Write locality)  Related Question (Tablet Master)
PNUTS

Structure

- Features
- Data and Query Model
- System Architecture
- Data Storage and Retrieval
- Consistency (Yahoo! Message Broker)

Recovery

- Query Processing
- Experiments
- Future Work
Recovery

- Any committed update is recoverable from a remote replica.

- Three step recovery
  - Tablet controller requests copy from remote (source) replica.
  - “Checkpoint message” published to YMB, for in-flight updates.
  - Source tablet is coped to destination region.

- Support for recovery
  - Synchronized tablet boundaries
  - Tablet splits at the same time (two-phase commit)
  - Backup regions within the same region.

Related Question
PNUTS

Structure

- Features
- Data and Query Model
- System Architecture
- Data Storage and Retrieval
- Consistency (Yahoo! Message Broker)
- Recovery
- **Query Processing**
- Experiments
- Future Work
Bulk load

• Upload large blocks into the database.
• Bulk inserts done in parallel to multiple storage units.
• Hash table – natural load balancing.
• Ordered table – Avoid hot spots.

Avoiding hot spots in ordered table
Query Processing

• Scatter-gather engine
  – Receives a multi-record request
  – Splits it into multiple individual requests for single records/tablet scans
  – Initiates requests in parallel.
  – Gather the result and passes to client.

• Server-side design?
  – Prevent multiple parallel client requests.
  – Server side optimization (group requests to same storage)

• Range scan - Continuation object.
Notifications

• Service to notify external systems of updates to data.
• Example: popular a keyword search engine index.
• Clients subscribe to all topics (tablets) for table
  – Client need no knowledge of tablet organization.
  – Creation of new topic (tablet split) - automatic subscription
• Break subscription of slow notification clients.
PNUTS

Structure

- Features
- Data and Query Model
- System Architecture
- Data Storage and Retrieval
- Consistency (Yahoo! Message Broker)
- Recovery
- Query Processing
- Experiments
- Future Work
Experimental setup

• Production PNUTS code
  – Enhanced with ordered table type

• Three PNUTS regions
  – 2 west coast, 1 east coast
  – 5 storage units, 2 message brokers, 1 router
  – West: Dual 2.8 GHz Xeon, 4GB RAM, 6 disk RAID 5 array
  – East: Quad 2.13 GHz Xeon, 4GB RAM, 1 SATA disk

• Workload
  – 1200-3600 requests/second
  – 0-50% writes
  – 80% locality
Experiments

• Inserts
  – required 75.6 ms per insert in West 1 (tablet master)
  – 131.5 ms per insert into the non-master West 2, and
  – 315.5 ms per insert into the non-master East.

Figure 3: Impact of varying request rate on the average request latency.
Experiments

Figure 4: Impact of varying read:write ratio on the average request latency.

Figure 5: Impact of varying the skew of requests on the average request latency.
Experiments

Figure 6: Impact of varying the number of storage units on the average request latency.

Figure 7: Impact of varying the size of range scans on the average request latency.
Bottlenecks

• Disk seek capacity on storage units
• Message Brokers

Different PNUTS customers are assigned different clusters of storage units and message broker machines. Can share routers and tablet controllers.
PNUTS

Structure

- Features
- Data and Query Model
- System Architecture
- Data Storage and Retrieval
- Consistency (Yahoo! Message Broker)
- Recovery
- Query Processing
- Experiments
- Future Work
Future Work

• Consistency
  – Referential integrity
  – Bundled update
  – Relaxed consistency  Related Question

• Data Storage and Retrieval:
  – Fair sharing of storage units and message brokers

• Query Processing
  – Query optimization: Maintain statistics
  – Expansion of query language: join/aggregation
  – Batch-query processing

• Indexes and Materialized views
Question 1  (Dolphia Nandi)

Q.  In section 3.3.2:Consistency via YMB and mastership~~
It is saying that 85% of writes to a record originate from the same data center which obviously justify locating the master at that point. Now my question is remaining 15% (with timeline consistency) must go across the wide area to the master and making it difficult to enforce SLA which is often set to 99%. What is your thought about this?

A.  SLA’s define a requirement of 99% availability. The remaining 15% of traffic that needs wide-area communciation increases latency and does not effect availability.
Q. Can we develop a strong consistent system which will deliver acceptable performance and maintains availability in the face of any single replica failure? Or is it still a fancy dream?

A. Strong consistency means ACID. ACID properties are in trade-off with availability, graceful degradation and performance. See references 1, 2 and 17. This tradeoff is fundamental.
   • CAP theorem (2000) outlined by Eric Brewer. It says that a distributed database system can only have at most two of the following three characteristics:
     – Consistency
     – Availability
     – Partition tolerance
   • Also see the NRW notation that describes at a high level how a distributed database will trade off consistency, read performance and write performance
Q. Why would users need table-scans, scatter-gather operations, bulk loading? Are these just best effort operations or is some consistency provided?

A. Some social applications may need these functionalities for instance, searching through all the available users to find the ones that reside in Buffalo, NY. Bulk loading may be needed in user database like applications, for uploading statistics of page views.

• At the time this paper was written
  – Technology was there to optimize bulk loading. However, for consistency, the paper says, “In particular, we make no guarantees as to consistency for multi-record transactions. Our model can provide serializability on a per-record basis. In particular, if an application reads or writes the same record multiple times in the same “transaction,” the application must use record versions to validate its own reads and writes to ensure serializability for the “transaction.”
  – Scatter-gather are best effort operations. They propose bundled updates-like consistency operations as future work.

• No major literature available since then.
Q. How does primary replica recovery and replica recovery work?
A. In slides.

Q. Is PNUTS (or components of PNUTS) available as open source?
A. No. (Source: http://wiki.apache.org/hadoop/Hbase/PNUTS)
Q. Can PNUTS work in a partitioned network? How does conflict resolution work when the network is united again?

A. In this mentioned as future work. “Under normal operation, if the master copy of a record fails, our system has protocols to fail over to another replica. However, if there are major outages, e.g., the entire region that had the master copy for a record becomes unreachable, updates cannot continue at another replica without potentially violating record-timeline consistency. We will allow applications to indicate, per-table, whether they want updates to continue in the presence of major outages, potentially branching the record timeline. If so, we will provide automatic conflict resolution and notifications thereof. The application will also be able to choose from several conflict resolution policies: e.g., discarding one branch, or merging updates from branches, etc.
Q. How do you compare PNUTS algorithm with the Paxos algorithm?

A. PNUTS though uses leadership is not entirely based on Paxos algorithm. It does not need a quorum to agree before deciding to execute/commit a query. The query is first executed and then applied to rest of the replicas. That being said, it does however seem to use it in publishing to the YMB (unless it uses chain replication, or something else).
Question 6 (Fatih)

Q. Is there any comparison with other data storage systems like Cassandra of Facebook, because it seems like latency is quite a bit high for figures (e.g. Figure3). In addition, when I was reading I was surprised with the quality of machines used in the experiment (Table 1). Could it be a reason for high latency?

A. SLA guarantees generally start from 50 ms and extend up to 150 ms across geographically diverse regions. So the first few graphs are within the SLA agreement, unless the number of clients is increased (in which case, a production environment can load balance to provide better SLA). It is an experimental setup, hence the limitations.

• Here is a link to such comparisons (Cassandra, Hbase, Sherpa and MySQL): http://www.brianfrankcooper.net/pubs/ycsb-v4.pdf
• These systems are not similar, for instance, Cassandra is P2P.
Question 7 (Hanifi Güneş)

Q. PNUTS leverages locality of updates. Given the typical applications running over PNUTS, we can say that the system is designed for single-write-multiple-read applications, say, in case of Flickr or even Facebook where only a single user is permissible to write and all others to see. Locality of updates, thus, seems a good cut for this design. How can we translate what we have learned from PNUTS in to multiple-read-multiple-write applications like a wide-area file system? Any ideas?

A. PNUTS is not strictly single-read-multiple-write, but good conflict resolution is the key if it needs to handle multiple-writes more effectively. PNUTS serves a different function than a wide-area file system. It is designed for simple queries of web traffic.

Q. Also, in case of a primary replica failure or partition, where/how exactly are the new coming requests routed? And how is the data consistency ensured in case of a 'branching' (the term, branching, is used in the same context as mentioned in the paper, pg 4)? Can you throw some light on?

A. Record has soft state metadata. Resolving branching is future work.
Q. In section 2.2 about per-record timeline consistency, it seems that consistency is serializability. All updates are forwards to master, suppose there are two replica who update their record at the same time, according to the paper, they would forward update record to master, since their timeline is similar (update at the same time), how the master recognize that scenario?

A. Tablet masters resolve this scenario.
Question 9  (Santosh)

Q.  Sec 3.21: How are changes in data handled while the previous data is yet to be updated to all the systems?
A.  Versioning of updates for timeline consistency.

Q.  How are hotspots related to user profiles handled?
A.  In slides.
Q. The author states in section 3.2.1 "While stronger ordering guarantees would simplify this protocol (published messaging), global ordering is too expensive to provide when different brokers are located in geographically separated datacenters." Since this paper was published in 2008, has any work been done to provide for global ordering and to mitigate the disadvantage of being too expensive? Wouldn't simplifying the protocol make it easy for robustness and further advancement?

A. They have done optimizations of Sherpa but have not really changed the basic PNUTS design, or at least have not published. Global ordering being expensive translates to the fundamental trade-off of consistency vs performance. Simplifying how? This is the simplest Yahoo! tried to keep it while still meeting the demands of their applications.
Thank you !!
Additional Definitions

- **JSON** (JavaScript Object Notation) is a lightweight data-interchange format. It is easy for humans to read and write. It is easy for machines to parse and generate.
- **InnoDB** is a transactional storage engine for the MySQL open source database.
- **Soft state** expires unless it is refreshed.
- A transaction schedule has the **Serializability** property, if its outcome (e.g., the resulting database state, the values of the database's data) is equal to the outcome of its transactions executed serially, i.e., sequentially without overlapping in time.

Zipfian distribution

- A distribution of probabilities of occurrence that follows Zipf’s Law.
- **Zipf’s Law:** The probability of occurrence of words or other items starts high and tapers off. Thus, a few occur very often while many others occur rarely.
- **Formal Definition:** $P_n = 1/n^a$, where $P_n$ is the frequency of occurrence of the $n^{th}$ ranked item and $a$ is close to 1.

Source: [http://xw2k.nist.gov](http://xw2k.nist.gov)
Bulk loading support

- An approach in which a planning phase is invoked before the actual insertions. By creating new partitions and intelligently distributing partitions across machines, the planning phase ensures that the insertion load will be well-balanced.

**Consistency model**

- Goal: make it easier for applications to reason about updates and cope with asynchrony

- What happens to a record with primary key “Brian”? 

![Diagram showing record updates and deletions over time](image-url)
Consistency model
Consistency model

Time

Read up-to-date

Stale version

Stale version

Current version

v. 1  v. 2  v. 3  v. 4  v. 5  v. 6  v. 7  v. 8

Generation 1
Consistency model

Read-critical (required version):

- Read ≥ v.6
- Stale version
- Stale version
- Current version

Time
Consistency model

Write

Stale version

Stale version

Current version

Time

v. 1 v. 2 v. 3 v. 4 v. 5 v. 6 v. 7 v. 8

Generation 1
Consistency model

Test-and-set-write(required version)

Write if = v.7

Stale version

Stale version

Current version

Time

Generation 1
Consistency model

Mechanism: per record mastership

Write if = v.7

ERROR

Stale version

Stale version

Current version

Time

Generation 1
What is PNUTS?

- Parallel database
- Geographic replication
- Structured, flexible schema
- Hosted, managed infrastructure
Detailed architecture
Detailed architecture

Local region

Remote regions

Clients

Tablet controller

Routers

REST API

YMB

Storage units
Accessing data

1. Get key k
2. Get key k
3. Record for key k
4. Record for key k
Bulk read

\[ \{k_1, k_2, \ldots, k_n\} \]

1. Get \( k_1 \)
2. Get \( k_2 \)
3. Get \( k_3 \)
Updates

1. Write key k
2. Write key k
3. Write key k
4. SUCCESS
5. Write key k
6. Write key k
7. Sequence # for key k
8. Sequence # for key k

Routers

Message brokers
Asynchronous replication