WebTable Use Case Documentation

This document details the WebTable use case implementation for the HBase Project, focusing on the schema, data ingestion, performance tuning, and the rationale behind design choices. It fulfills the requirements for the WebTable use case (section 4) and documentation (section 5), providing setup instructions, configuration, network architecture, security considerations, data modeling decisions, and sample queries. The design is implemented using modular Java classes (`Hbase` , `WebPage` , `utils`) on an HBase 2.5.11 cluster with Hadoop 3.3.6 and ZooKeeper 3.8, deployed in a Dockerized high-availability (HA) environment.

1. WebTable Use Case Implementation

The WebTable use case stores crawled web pages, their outgoing links, and metadata in an HBase table (`webtable`), enabling efficient page retrieval and link analysis for web graph modeling. The implementation leverages the `Hbase` class for `HBaseWebTabele` operations, `WebPage` for crawling, and `utils` for URL processing, ensuring modularity and maintainability.

1.1 Application Requirements

1.1.1 WebTable Schema

The `webtable` schema is designed for scalability and query efficiency:

- **Table Name**: `webtable`
- **Row Key**: Reversed URL (e.g., `org.hbase.www` for `http://www.hbase.org`), generated by `utils.reverseUrl` .
- **Column Families**:
  - `contents` : Stores HTML content.
    - Qualifier: `html`
    - Max Versions: 3 (for historical tracking)
    - Compression: SNAPPY
  - `anchor` : Stores outgoing links.
    - Qualifier: `out:<reversed_target_url>` (e.g., `out:org.apache.www`)
    - Max Versions: 1
    - Compression: SNAPPY
  - `metadata` : Stores page metadata.
    - Qualifier: `language` (e.g., `en`)
    - Max Versions: 1
    - Compression: SNAPPY

Implementation (in `Hbase.java`):

```
TableDescriptorBuilder tableBuilder = TableDescriptorBuilder.newBuilder(TableName.valueOf("webtable"));
tableBuilder.setColumnFamily(
    ColumnFamilyDescriptorBuilder.newBuilder(Bytes.toBytes("contents"))
        .setMaxVersions(3)
        .setCompressionType(Compression.Algorithm.SNAPPY)
        .build()
);
tableBuilder.setColumnFamily(
    ColumnFamilyDescriptorBuilder.newBuilder(Bytes.toBytes("anchor"))
        .setCompressionType(Compression.Algorithm.SNAPPY)
        .build()
);
tableBuilder.setColumnFamily(
    ColumnFamilyDescriptorBuilder.newBuilder(Bytes.toBytes("metadata"))
        .setCompressionType(Compression.Algorithm.SNAPPY)
        .build()
);
admin.createTable(tableBuilder.build());
```

1.1.2 Table and Column Family Creation

The `Hbase.createTable` method creates the `webtable` if it doesn't exist, defining column families with SNAPPY compression and versioning to optimize storage and retrieval. It ensures idempotency by checking table existence, supporting seamless setup.

1.1.3 Performance Tuning

Performance optimizations enhance scalability and efficiency:

- **SNAPPY Compression**: Minimizes storage for HTML and link data while maintaining fast read/write performance.
- **Row Key Design**: Reversed URLs (via `utils.reverseUrl`) ensure even region distribution, optimizing load balancing.
- **Compaction**: Configured with a 7-day interval (`hbase.hregion.majorcompaction=604800000`) to maintain storage efficiency.
- **Connection Management**: Try-with-resources for `Connection` and `Table` ensures resource efficiency.
- **JMX Monitoring**: Enabled on port 10102 for real-time performance metrics.

1.1.4 Data Ingestion Processes

Data ingestion supports flexible crawling and storage:

- **`Hbase.ingestData`**:
  - Crawls a single URL using `WebPage`, extracting HTML, outgoing links, and language.
  - Stores data in `contents:html`, `anchor:out:<reversed_target_url>`, and `metadata:language` using `table.put`.
- **`Hbase.ingestBulkData`**:
  - Processes multiple URLs, creating `WebPage` instances and storing data for each.
  - Uses `table.put` for controlled ingestion, suitable for small to medium datasets.
- **Crawling**: The `WebPage` class uses JSoup to parse HTML, extract `<a href>` links, and determine language from `<html lang>`, defaulting to `en` if unspecified.
- **Input**: URLs from `urls.txt` or programmatic lists.

Example (in `Hbase.java`):

```
String reversedUrl = utils.reverseUrl(url);
WebPage page = new WebPage(url);
long crawlTimestamp = System.currentTimeMillis();
Put put = new Put(Bytes.toBytes(reversedUrl));
put.addColumn(Bytes.toBytes("contents"), Bytes.toBytes("html"), crawlTimestamp, Bytes.toBytes(page.html));
for (String link : page.outLinks) {
    put.addColumn(Bytes.toBytes("anchor"), Bytes.toBytes("out:" + utils.reverseUrl(link)), crawlTimestamp, Bytes.toBytes(link));
}
put.addColumn(Bytes.toBytes("metadata"), Bytes.toBytes("language"), crawlTimestamp, Bytes.toBytes(page.language));
table.put(put);
```

2. Setup Documentation

2.1 Step-by-Step Installation Guide

1. **Prerequisites**:
   - OS: Ubuntu/Debian
   - Tools: Docker, Maven, Java 8
   - Dependencies: HBase 2.5.11, Hadoop 3.3.6, ZooKeeper 3.8
2. **Configure Hostnames**:
   - Edit `/etc/hosts`:

```
# zookeeper
172.20.0.4 master1
172.20.0.2 master2
172.20.0.3 master3

#HbaseMaster
172.20.0.6 hb-master1
172.20.0.7 hb-master2

#Hbase Regions
172.20.0.9 hbase-ha-clustercp-hb-region-servers1-1.hbase-ha-clustercp_cluster_net
172.20.0.8 hbase-ha-clustercp-hb-region-servers2-1.hbase-ha-clustercp_cluster_net
```

3. **Set Up Docker**:
   - **Dockerfile**:

```
FROM an2071497/hadoopimg:1.13

ENV HBASE_VERSION=2.4.9
ENV HBASE_HOME=/usr/local/hbase
ENV PATH=/usr/local/hbase/bin:$PATH

USER root

RUN apt update && sudo apt install -y libsnappy-dev
RUN curl -L https://archive.apache.org/dist/hbase/2.4.9/hbase-2.4.9-bin.tar.gz -o /tmp/hbase-2.4.9-bin.tar.gz && \
    tar -xvzf /tmp/hbase-2.4.9-bin.tar.gz -C /usr/local && \
    mv /usr/local/hbase-2.4.9 /usr/local/hbase && \
    rm /tmp/hbase-2.4.9-bin.tar.gz && \

chown -R hduser:hadoop /usr/local/hbase

USER hduser

COPY ./hbase-site.xml /usr/local/hbase/conf/hbase-site.xml
```

```
RUN cp /usr/local/hadoop/share/hadoop/common/lib/* /usr/local/hbase/lib/
COPY entrypoint.sh /entrypoint.sh

ENTRYPOINT [ "/entrypoint.sh" ]
```

4. **Deploy Cluster**:
   - **entrypoint.sh**:

```
#!/bin/bash
if [ "$HBASE_ROLE" = "master" ]; then
    echo "Starting HBase Master..."
    exec $HBASE_HOME/bin/hbase master start
elif [ "$HBASE_ROLE" = "regionserver" ]; then
    echo "Starting HBase RegionServer..."
    exec $HBASE_HOME/bin/hbase regionserver start
else
    echo "Unknown role: $HBASE_ROLE"
exit 1
fi
```

- **Run**:

```
cd Hbase-HA-Cluster
docker-compose -f docker-compose.yml up
```

5. **Configure Maven**:
   - **pom.xml**:

```
<dependencies>
   <!-- HBase -->
   <dependency>
       <groupId>org.apache.hbase</groupId>
       <artifactId>hbase-client</artifactId>
       <version>2.4.9</version>
       <exclusions>
            <exclusion>
            <groupId>org.apache.hadoop</groupId>
               <artifactId>*</artifactId>
           </exclusion>
       </exclusions>
   </dependency>
    <dependency>
        <groupId>org.apache.hbase</groupId>
       <artifactId>hbase-common</artifactId>
       <version>2.4.9</version>
       <exclusions>
            <exclusion>
                <groupId>org.apache.hadoop</groupId>
                <artifactId>*</artifactId>
           </exclusion>
       </exclusions>
   </dependency>
   <!-- Hadoop -->
   <dependency>
       <groupId>org.apache.hadoop</groupId>
       <artifactId>hadoop-common</artifactId>
       <version>3.2.4</version>
   </dependency>
   <dependency>
        <groupId>org.apache.hadoop</groupId>
       <artifactId>hadoop-hdfs</artifactId>
       <version>3.2.4</version>
    </dependency>
    <dependency>
        <groupId>org.apache.hadoop</groupId>
        <artifactId>hadoop-auth</artifactId>
        <version>3.2.4</version>
    </dependency>
   <!-- Logging -->
   <dependency>
       <groupId>org.slf4j</groupId>
       <artifactId>slf4j-api</artifactId>
       <version>1.7.36</version>
   </dependency>
   <dependency>
        <groupId>org.slf4j</groupId>
```

```
        <artifactId>slf4j-log4j12</artifactId>
        <version>1.7.36</version>
    </dependency>
    <dependency>
         <groupId>org.jsoup</groupId>
        <artifactId>jsoup</artifactId>
        <version>1.17.2</version>
    </dependency>
</dependencies>
```

6. **Run Application**:
   - `src/main/resources/urls.txt` :

```
http://www.hbase.org
http://www.apache.org
http://www.example.com
http://www.github.com
http://www.wikipedia.org
```

- Example main class:

```java
package org.example;
import org.example.hbase.Hbase;
import java.util.Arrays;
public static void main(String[] args) {
    Hbase hbase = new Hbase();
    try {
        hbase.createTable();
        hbase.showConnectionInfo();
        List<String> urls = utils.readUrls("src/main/resources/urls.txt");
        hbase.ingestBulkData(urls);
        hbase.flushTable("webtable");

        String reversedUrl = utils.reverseUrl(urls.get(1));
        WebPage webPage = hbase.getWebPage(reversedUrl);
        utils.writeSep();
        System.out.println(webPage);

    } catch (IOException e) {
        System.err.println("Error: " + e.getMessage());
        e.printStackTrace();
    }
}
```

- Compile and run:

```
mvn clean compile exec:java -Dexec.mainClass="org.example.Main"
```

```
    </div>
  </footer><!-- / Footer -->
  <script src="/js/jquery.min.js"></script>
  <script src="/js/bootstrap.js"></script>
  <script src="/js/slideshow.js"></script>
  <script>
    (function($){
      $(document).ready(function(){
        $('ul.dropdown-menu [data-toggle=dropdown]').on('click', function(event) {
          event.preventDefault();
          event.stopPropagation();
          $(this).parent().siblings().removeClass('open');
          $(this).parent().toggleClass('open');
          console.log('WOrked');
        });
      });
    })(jQuery);
  </script>
 </body>
</html>
Lang: en
OutLinks: [https://www.apache.org/licenses/LICENSE-2.0, https://twitter.com/TheASF, https://github.com/apache, https://www.youtube.com/c/TheApacheFou
```

2.2 Configuration Files

- **hbase-site.xml**:

```
<configuration>
<property>
  <name>hbase.zookeeper.quorum</name>
  <value>172.20.0.4,172.20.0.2,172.20.0.3</value>
  <description>ZooKeeper quorum for HBase coordination.</description>
</property>
<property>
  <name>hbase.zookeeper.property.clientPort</name>
  <value>2181</value>
  <description>ZooKeeper client port.</description>
</property>
<property>
  <name>hbase.master</name>
  <value>172.20.0.4:60000</value>
  <description>HBase Master address.</description>
</property>
<property>
  <name>hbase.rootdir</name>
  <value>hdfs://172.20.0.4:9000/hbase</value>
  <description>HDFS path for HBase data.</description>
</property>
<property>
  <name>hbase.security.authentication</name>
  <value>simple</value>
  <description>Authentication mode (simple for development).</description>
</property>
<property>
  <name>hbase.regionserver.jmx</name>
  <value>true</value>
  <description>Enable JMX for monitoring.</description>
</property>
<property>
  <name>hbase.hregion.majorcompaction</name>
  <value>604800000</value>
  <description>7-day compaction interval for storage optimization.</description>
</property>
</configuration>
```

2.3 Network Architecture

The HA HBase cluster integrates with Hadoop HA on a Docker network (`hbase-net`). Below is a textual representation:

```
+---------------------+       +---------------------+
| Hadoop NameNode     |       | ZooKeeper Quorum    |
| master1:9000        |<----->| master1:2181        |
| master1:9000        |       | master2:2181        |
| master3:9000        |       | master3:2181        |
+---------------------+       +---------------------+
                                      |
                                      v
+---------------------+       +---------------------+
| HBase Master (1)    |<----->| HBase Master (2)    |
| 172.20.0.4:60000    |       | 172.20.0.4:60001    |
| JMX: 10102          |       |                     |
+---------------------+       +---------------------+
          |                           |
          v                           v
+---------------------+       +---------------------+
| RegionServer (1)    |       | RegionServer (2)    |
| 172.20.0.4          |       | 172.20.0.2          |
+---------------------+       +---------------------+
          |                           |
          v                           v
+---------------------+       +---------------------+
| RegionServer (3)    |       | HDFS DataNodes      |
| 172.20.0.3          |<----->| 172.20.0.4, etc.    |
+---------------------+       +---------------------+
```

- **Components**:
  - Hadoop NameNode: Stores HBase data (`hdfs://172.20.0.4:9000/hbase`).
  - ZooKeeper Quorum: Coordinates HBase Master election and RegionServer assignments (3 nodes for HA).
  - HBase Masters: 2 nodes (active/standby) for failover.
  - RegionServers: 2-3 nodes for data storage and queries.
  - HDFS DataNodes: Back HBase data storage.

- **Network**: `hbase-net` isolates components, with IP mappings (`172.20.0.4`).

2.4 Security Considerations

- **Authentication**: Configured as `simple` for development. For production, use Kerberos (`hbase.security.authentication=kerberos`).
- **Network**: Docker network isolates components. For production, use firewalls to restrict ports (60000, 2181, 9000).
- **Data**: No encryption in development. For production, enable HBase encryption.

3. Use Case Documentation

3.1 WebTable Schema Design

- **Row Key**: Reversed URL (e.g., `org.hbase.www`) via `utils.reverseUrl`.
- **Column Families**:
  - `contents`: HTML content (`html`, 3 versions, SNAPPY) for page rendering/indexing.
  - `anchor`: Outgoing links (`out:<reversed_target_url>`, SNAPPY) for web graph analysis.
  - `metadata`: Metadata (`language`, SNAPPY) for filtering/analytics.
- **Compression**: SNAPPY minimizes storage.
- **Versioning**: 3 versions for `contents:html` track page changes.

3.2 Data Modeling Decisions and Design Rationale

The WebTable design was chosen for its alignment with web crawling and analysis needs. Below are the key decisions and reasons:

- **Reversed URL Row Key**:
  - **Why**: Reversing URLs (e.g., `http://www.hbase.org` to `org.hbase.www`) ensures even key distribution across HBase regions, preventing hotspotting. It groups related domains (e.g., `org.hbase.*`) for efficient scans, critical for domain-level analysis (e.g., all pages under `org.*`).
  - **Benefit**: Enhances scalability for millions of URLs and supports fast prefix-based queries.
  - **Alternative Considered**: Hash-prefixed URLs (e.g., `MD5(url):url`) were rejected as they disrupt domain locality, complicating domain scans.
- **Separate Column Families**:
  - **Why**: Three column families (`contents`, `anchor`, `metadata`) separate data by access pattern. `contents` stores large HTML data, `anchor` handles link relationships, and `metadata` supports filtering (e.g., by language).
  - **Benefit**: Reduces I/O by fetching only relevant data (e.g., HTML without links), improving query performance.
  - **Alternative Considered**: A single column family with multiple qualifiers was rejected due to increased I/O for mixed queries.
- **SNAPPY Compression**:
  - **Why**: SNAPPY is applied to all column families to reduce storage for text-heavy HTML and link data while maintaining fast decompression.
  - **Benefit**: Balances storage efficiency and query speed, ideal for large-scale web data.
  - **Alternative Considered**: GZ compression was less favorable due to slower decompression, impacting query latency.
- **Versioning for HTML**:
  - **Why**: `contents:html` supports 3 versions to track page changes over time, useful for historical analysis or auditing.
  - **Benefit**: Enables retrieval of recent page versions without external storage, supporting use cases like change detection.
  - **Alternative Considered**: Single versioning was rejected as it limits historical analysis; more than 3 versions would increase storage overhead.
- **Single Table Design**:
  - **Why**: All data (HTML, links, metadata) is stored in `webtable` to simplify the schema and queries, as they are closely related.
  - **Benefit**: Streamlines data management and reduces complexity compared to multiple tables.
  - **Alternative Considered**: Separate tables for links and content were rejected due to join complexity in HBase.
- **Modular Code Structure**:
  - **Why**: The design separates concerns with `utils` for URL processing, `WebPage` for crawling, and `Hbase` for HBase operations.
  - **Benefit**: Enhances maintainability and reusability, allowing easy extension (e.g., adding incoming links).
  - **Alternative Considered**: A monolithic class was rejected due to reduced flexibility.

3.3 Performance Optimization Techniques

- **Compression**: SNAPPY for all column families reduces storage and speeds up queries.
- **Row Key**: Reversed URLs ensure balanced region distribution.
- **Compaction**: 7-day interval (`hbase.hregion.majorcompaction=604800000`) optimizes storage.
- **Future**: Pre-split regions and batch writes for large-scale ingestion.

3.4 Sample Queries and Expected Results

1. **Retrieve HTML**:

```
get 'webtable', 'org.hbase.www', 'contents:html'
```

Result:

```
COLUMN                    CELL
contents:html             timestamp=1739990400000, value=<html><body>Welcome to HBase</body></html>
```

2. **Retrieve Outgoing Links**:

```
get 'webtable', 'org.hbase.www', 'anchor'
```

Result:

```
COLUMN                      CELL
anchor:out:org.apache.www timestamp=1739990400000, value=http://www.apache.org
```

3. **Retrieve Language**:

```
get 'webtable', 'org.hbase.www', 'metadata:language'
```

Result:

```
COLUMN                      CELL
metadata:language          timestamp=1739990400000, value=en
```

4. **Retrieve HTML Versions**:

```
get 'webtable', 'org.hbase.www', {COLUMN => 'contents:html', VERSIONS => 3}
```

Result:

```
COLUMN                      CELL
contents:html              timestamp=1739990400000, value=<html><body>Welcome to HBase v2</body></html>
contents:html              timestamp=1739986800000, value=<html><body>Welcome to HBase v1</body></html>
contents:html              timestamp=1739983200000, value=<html><body>Welcome to HBase v0</body></html>
```

4. Verification

- **HBase Shell**:

```
scan 'webtable'
get 'webtable', 'org.hbase.www', {COLUMNS => ['contents:html', 'anchor', 'metadata:language'], VERSIONS => 3}
```

- **Monitoring**:
  - HBase UI: `http://172.20.0.4:16010`

5. Future Improvements

- **Incoming Links**: Add `in:<reversed_source_url>` to `anchor` for full web graph modeling.
- **Metadata**: Include qualifiers like content_type`.