POLITECNICO DI MILANO

Scuola di Ingegneria Industriale e dell'Informazione

Corso di Laurea Magistrale in Ingegneria Informatica

Anno Accademico 2013 - 2014

Avoiding CRUD operations lock-in in NoSQL databases: extension of the CPIM library

Candidato: Fabio Arcidiacono (799001)

Relatore: Prof.ssa Elisabetta Di Nitto

Correlatore: Ing. Marco Scavuzzo

Data management techniques

RDBMS	NoSQL
Well structured data	Non-structured data
Vertical scaling	Horizontal scaling
ACID transactions	BASE properties
Relational model	Various data models
SQL	**Proprietary API**

NoSQL Common language approaches

Meta-model

- Apache MetaModel
- SOS platform

SQLification

- Apache Phoenix
- UnQL
- Native support

ORM

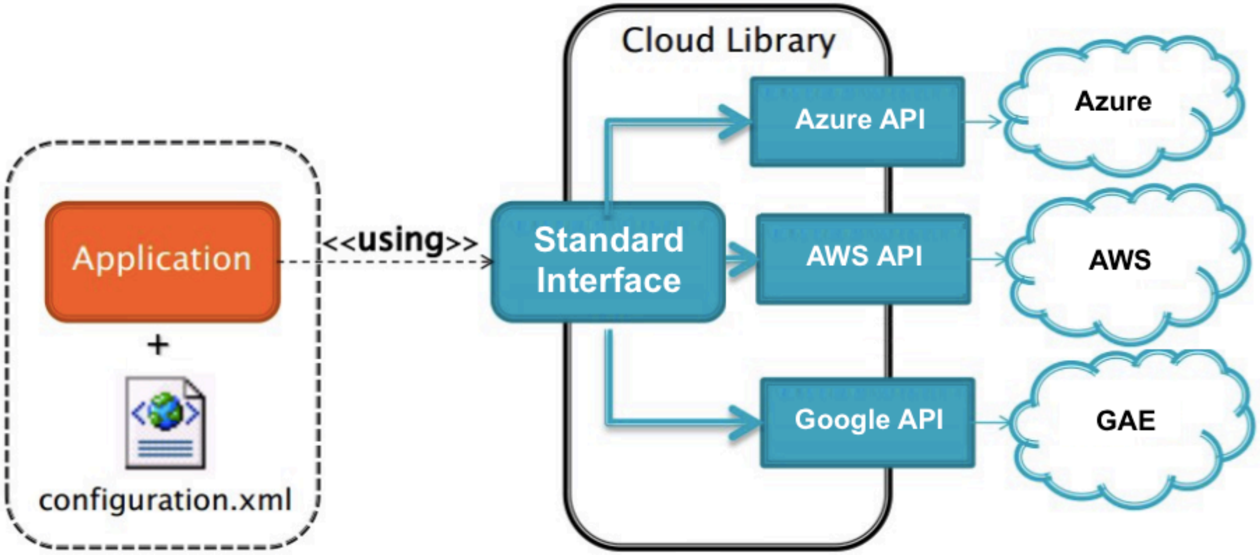
- Kundera
- PlayORM
- Spring-data
- Apache Gora

Cloud Platform Independent Model

Abstract application logic from the specific PaaS Provider to overcome the vendor lock-in

Many supported services:

- Blob
- NoSQL
- Memcache
- Queue
- Mail
- SQL



Work objectives

Integrate Kundera in the CPIM library

- extending the number of NoSQL databases supported
- fixing of the problems of the NoSQL service of CPIM

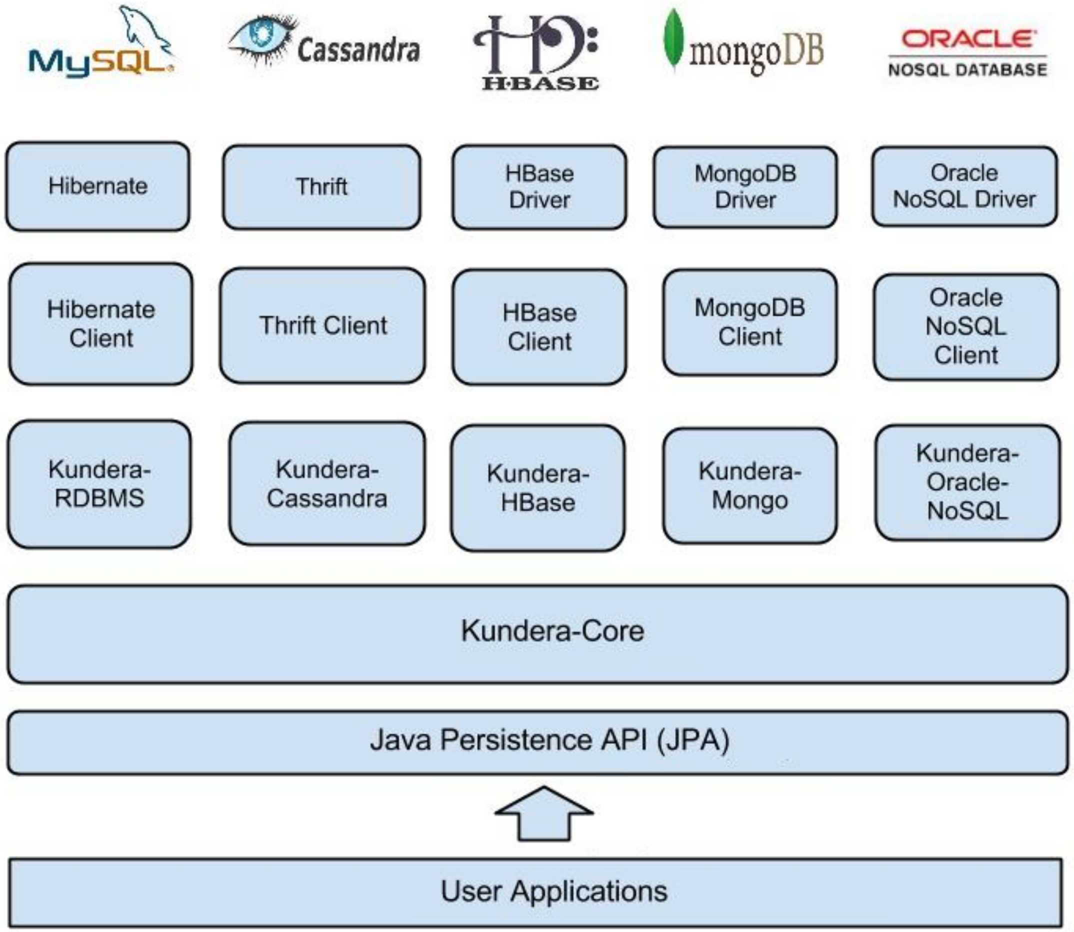
Contribute to the open source project Kudera

- developing a client for GAE Datastore
- developing a client for Azure Tables

Support data migration among NoSQL databases through the migration and synchronization system Hegira

Kundera

A JPA 2.1 compliant ORM Library for NoSQL databases



Hibernate | Thrift | HBase Driver | MongoDB Driver | Oracle NoSQL Driver

Hibernate Client | Thrift Client | HBase Client | MongoDB Client | Oracle NoSQL Client

Kundera-RDBMS | Kundera-Cassandra | Kundera-HBase | Kundera-Mongo | Kundera-Oracle-NoSQL

Kundera-Core

Java Persistence API (JPA)

User Applications

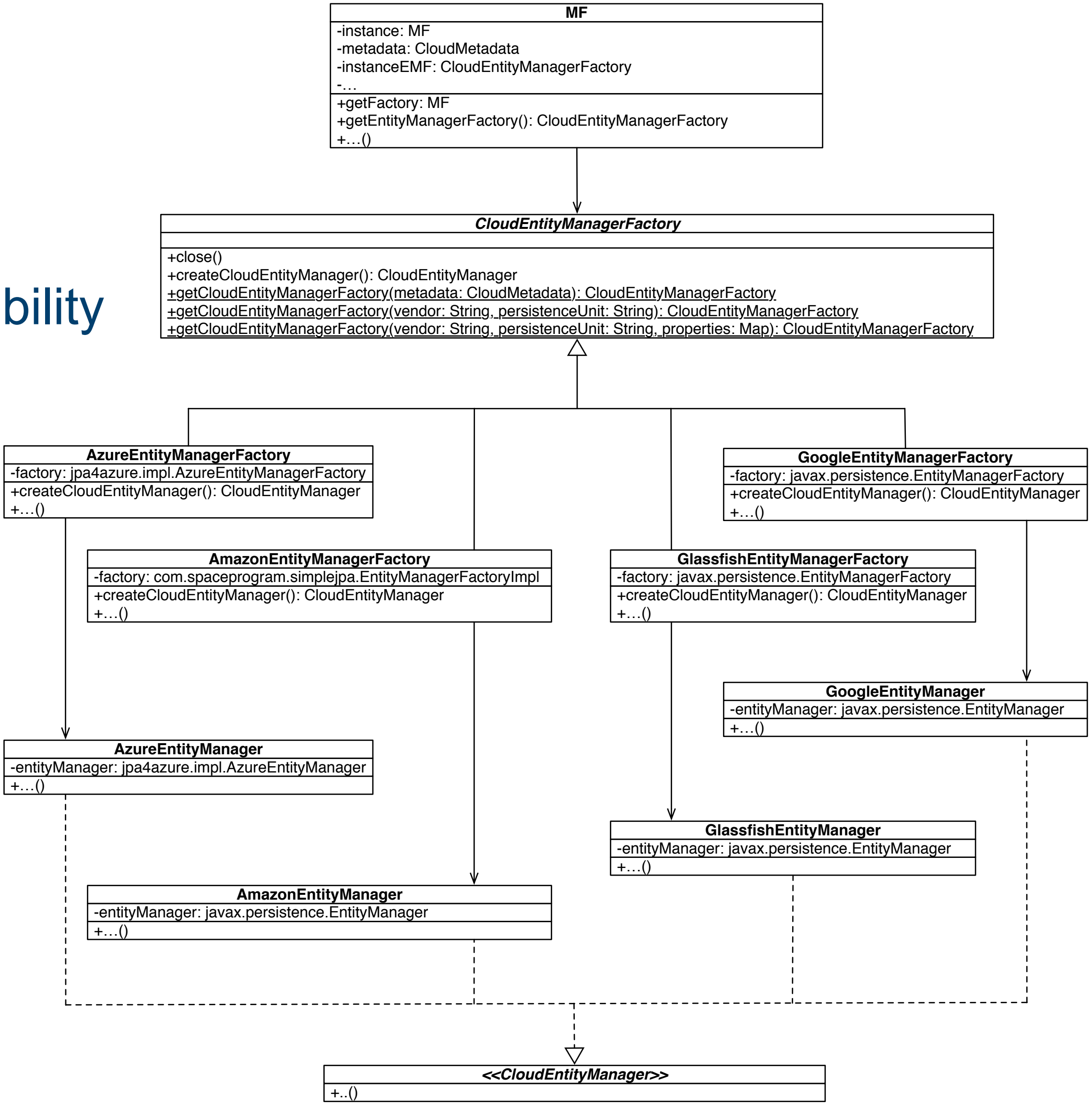
Why Kundera

- Open source
- Developed with extensibility as primary goal
- Ploygglot persistency
- In the field since 2010 with an active community
- Already used in production
- Support to many different NoSQL databases

Kundera integration (1)

Current implementation

- Many JPA provider
- Duplicated code
- No complete code portability



MF
-instance: MF
-metadata: CloudMetadata
-instanceEMF: CloudEntityManagerFactory
-…
+getFactory: MF
+getEntityManagerFactory(): CloudEntityManagerFactory
+…()

CloudEntityManagerFactory
+close()
+createCloudEntityManager(): CloudEntityManager
+getCloudEntityManagerFactory(metadata: CloudMetadata): CloudEntityManagerFactory
+getCloudEntityManagerFactory(vendor: String, persistenceUnit: String): CloudEntityManagerFactory
+getCloudEntityManagerFactory(vendor: String, persistenceUnit: String, properties: Map): CloudEntityManagerFactory

AzureEntityManagerFactory
-factory: jpa4azure.impl.AzureEntityManagerFactory
+createCloudEntityManager(): CloudEntityManager
+…()

GoogleEntityManagerFactory
-factory: javax.persistence.EntityManagerFactory
+createCloudEntityManager(): CloudEntityManager
+…()

AmazonEntityManagerFactory
-factory: com.spaceprogram.simplejpa.EntityManagerFactoryImpl
+createCloudEntityManager(): CloudEntityManager
+…()

GlassfishEntityManagerFactory
-factory: javax.persistence.EntityManagerFactory
+createCloudEntityManager(): CloudEntityManager
+…()

GoogleEntityManager
-entityManager: javax.persistence.EntityManager
+…()

AzureEntityManager
-entityManager: jpa4azure.impl.AzureEntityManager
+…()

GlassfishEntityManager
-entityManager: javax.persistence.EntityManager
+…()

AmazonEntityManager
-entityManager: javax.persistence.EntityManager
+…()

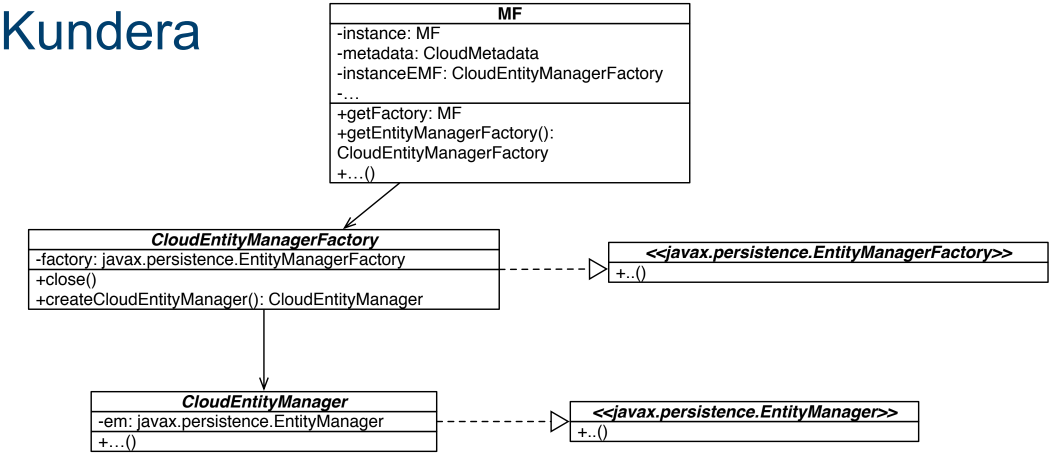
<<CloudEntityManager>>
+..()

Original CPIM NoSQL service implementation

- Many JPA providers
- Duplicated code
- No complete code portability
- Choice of the NoSQL database strictly bounded to the cloud provider (e.g. App Engine → Datastore)
- Limited NoSQL databases support

Kundera integration

- Single persistence provider
- NoSQL support inherited by Kundera



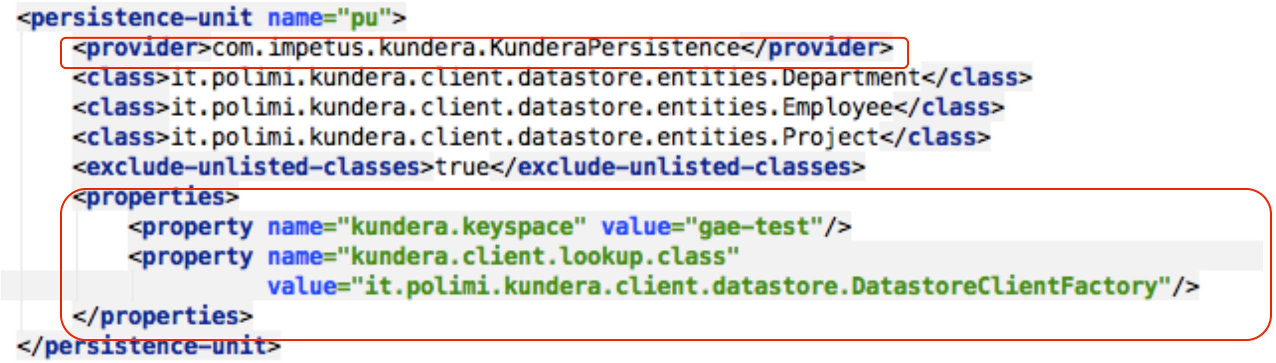
MF
-instance: MF
-metadata: CloudMetadata
-instanceEMF: CloudEntityManagerFactory
-…
+getFactory: MF
+getEntityManagerFactory(): CloudEntityManagerFactory
+…()

CloudEntityManagerFactory
-factory: javax.persistence.EntityManagerFactory
+close()
+createCloudEntityManager(): CloudEntityManager

<<javax.persistence.EntityManagerFactory>>
+..()

CloudEntityManager
-em: javax.persistence.EntityManager
+…()

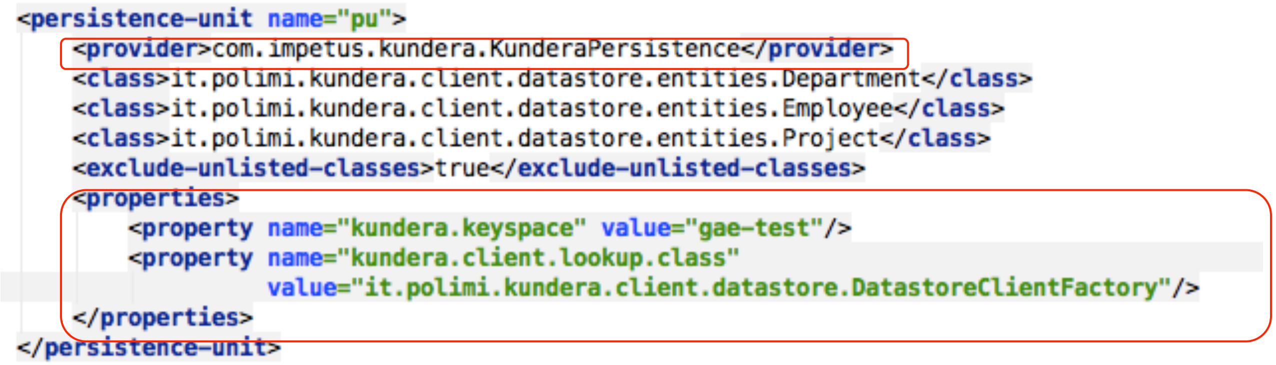
<<javax.persistence.EntityManager>>
+..()



```
<persistence-unit name="pu">
    <provider>com.impetus.kundera.KunderaPersistence</provider>
    <class>it.polimi.kundera.client.datastore.entities.Department</class>
    <class>it.polimi.kundera.client.datastore.entities.Employee</class>
    <class>it.polimi.kundera.client.datastore.entities.Project</class>
    <exclude-unlisted-classes>true</exclude-unlisted-classes>
    <properties>
        <property name="kundera.keyspace" value="gae-test"/>
        <property name="kundera.client.lookup.class"
                  value="it.polimi.kundera.client.datastore.DatastoreClientFactory"/>
    </properties>
</persistence-unit>
```

Kundera integration

- Single persistence provider
- NoSQL support inherited by Kundera
- Easier Configuration through standard persistence.xml



```
<persistence-unit name="pu">
    <provider>com.impetus.kundera.KunderaPersistence</provider>
    <class>it.polimi.kundera.client.datastore.entities.Department</class>
    <class>it.polimi.kundera.client.datastore.entities.Employee</class>
    <class>it.polimi.kundera.client.datastore.entities.Project</class>
    <exclude-unlisted-classes>true</exclude-unlisted-classes>
    <properties>
        <property name="kundera.keyspace" value="gae-test"/>
        <property name="kundera.client.lookup.class"
                  value="it.polimi.kundera.client.datastore.DatastoreClientFactory"/>
    </properties>
</persistence-unit>
```

Work objectives

Integrate Kundera in the CPIM library

- extending the number of NoSQL databases supported
- fixing of the problems of the NoSQL service of CPIM

Contribute to the open source project Kudera

- developing a client for GAE Datastore
- developing a client for Azure Tables

Support data migration among NoSQL databases through the migration and synchronization system Hegira

Contributes to Kundera

Paradigm shift

- support for DaaS
- Merged Kundera deploy on PaaS

Two newly developed client

- Azure Tables[1]
- GAE Datastore[2]

[1] https://github.com/deib-polimi/kundera-azure-table
[2] https://github.com/deib-polimi/kundera-gae-datastore

Developed clients

GAE Datastore	Azure Tables

Work objectives

Integrate Kundera in the CPIM library

- extending the number of NoSQL databases supported
- fixing of the problems of the NoSQL service of CPIM

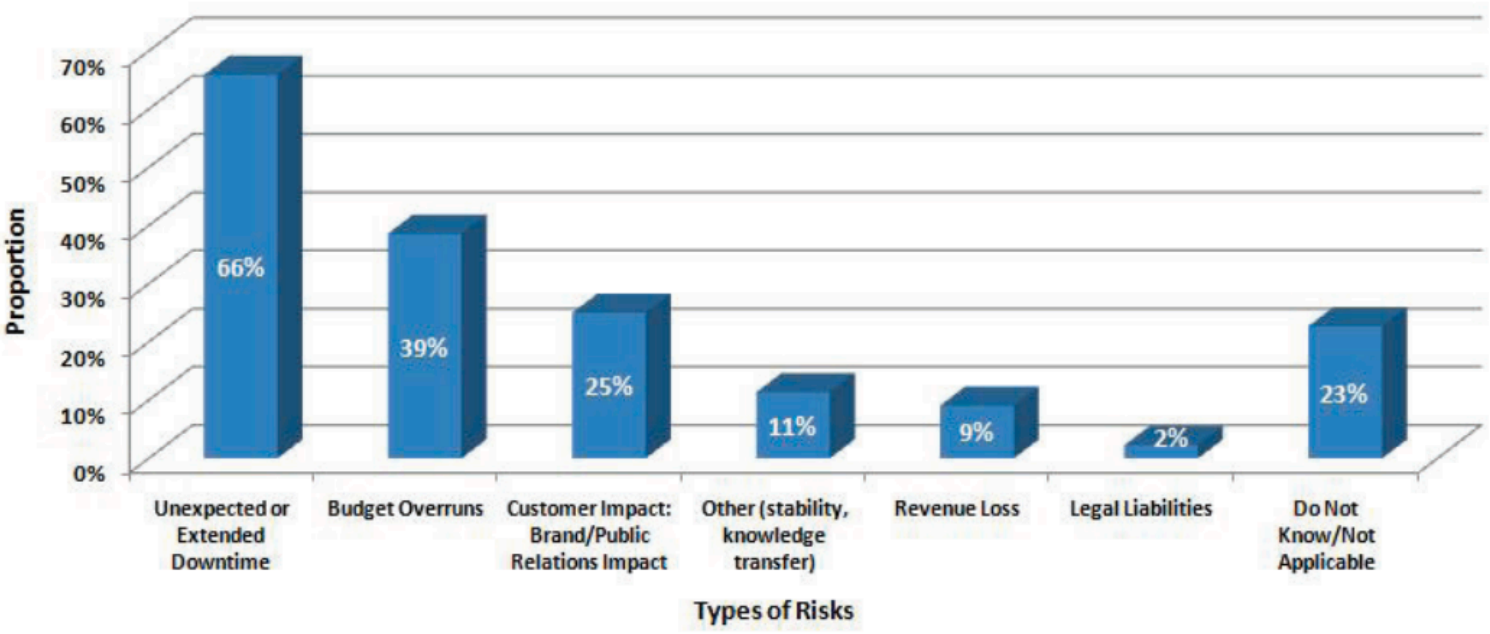
Contribute to the open source project Kudera

- developing a client for GAE Datastore
- developing a client for Azure Tables

Support data migration among NoSQL databases through the migration and synchronization system Hegira

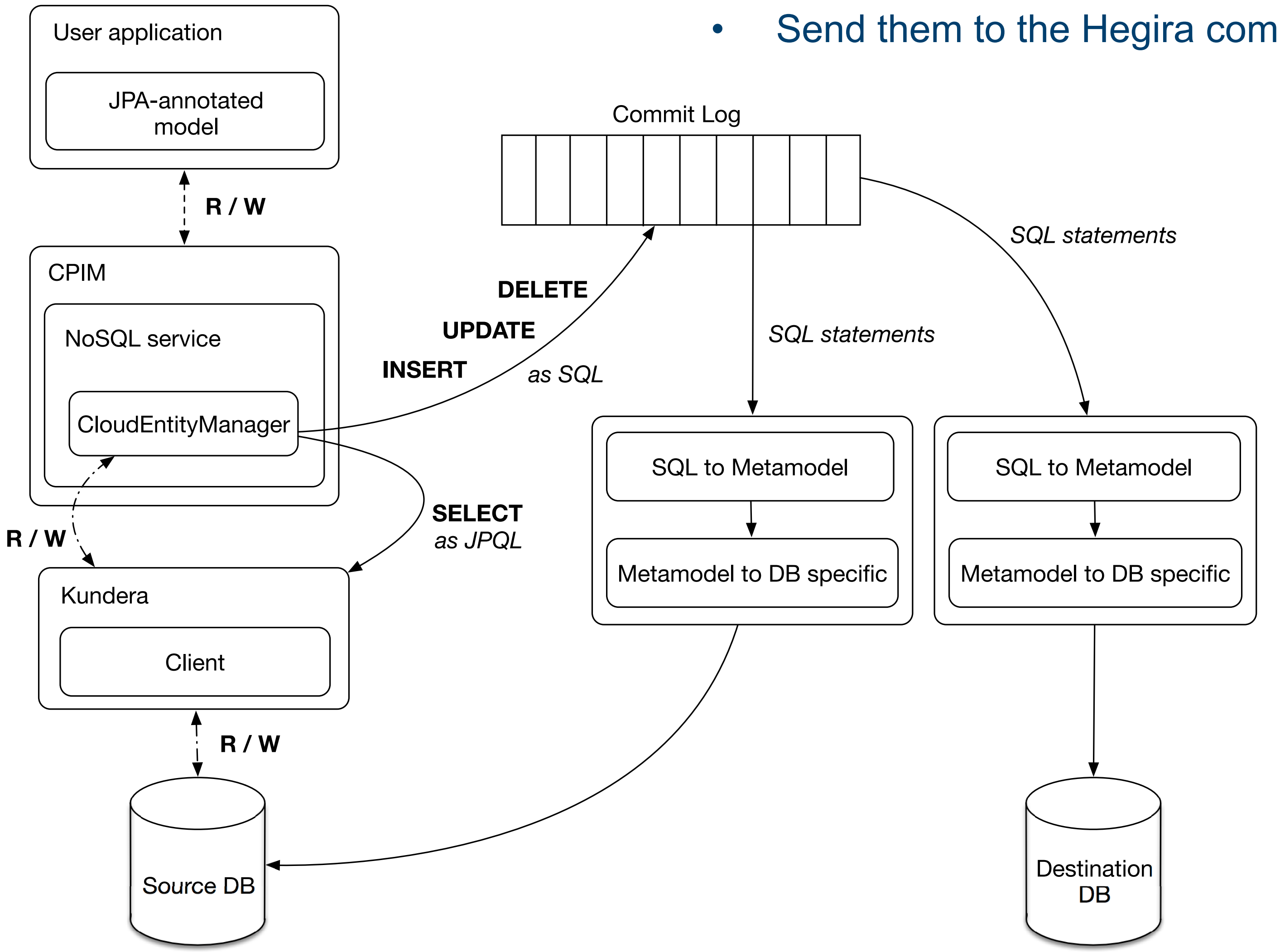
Data migration

- move application to another cloud provider
- move data to a database that better fit requirements
- load balancing, system expansion, failure recovery, costs, etc.
- modern computer systems are expected to be up continuously
- data synchronization between the two involved systems



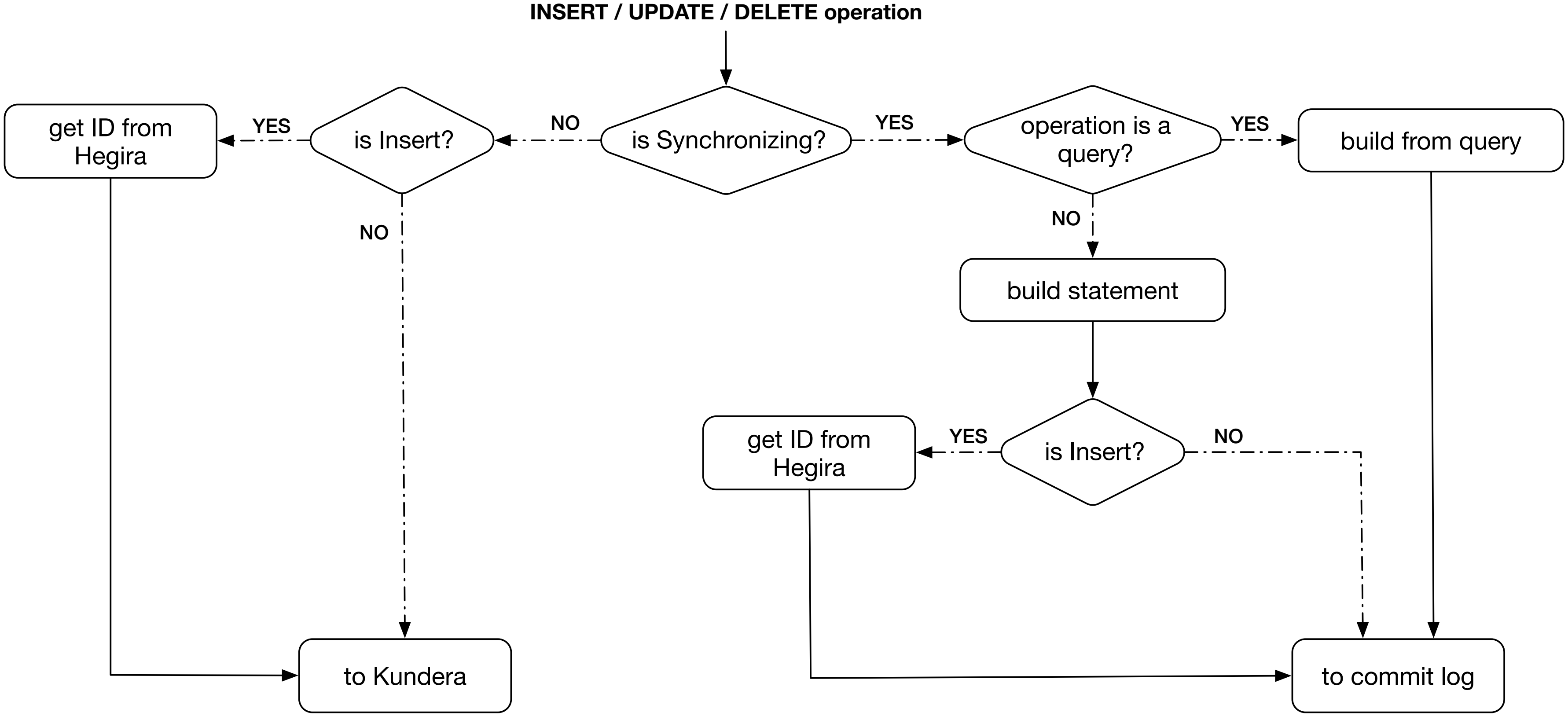
Hegira support (1)

- Intercept transparently user operations (DMQ)
- Translate operations to SQL statements
- Send them to the Hegira commit-log



Hegira support (2)

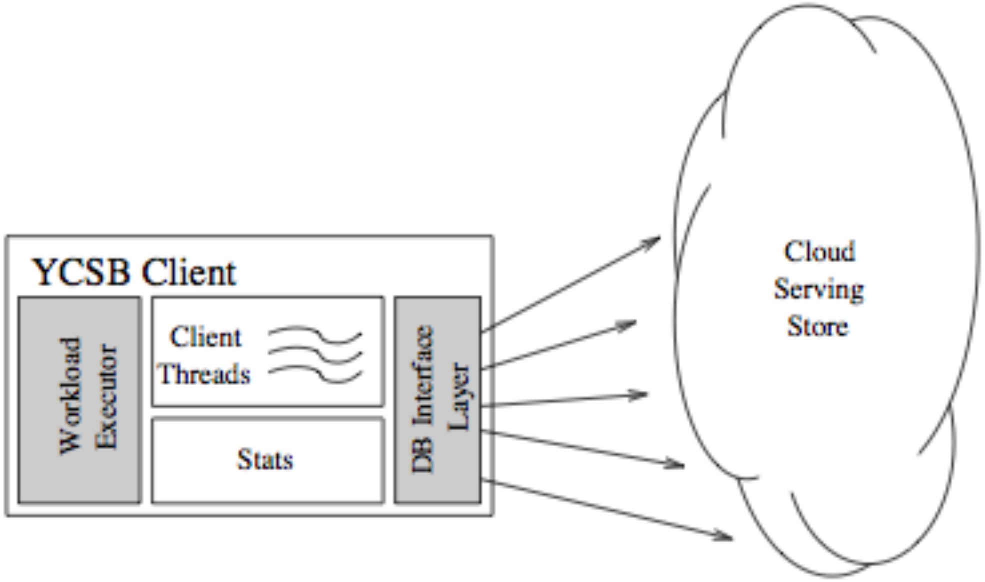
- Guarantee data synchronization
- Translate to SQL intercepted operations
  - JPQL queries (DELETE and UPDATE)
  - ORM operation (through *EntityManager* interface)



Performance

YAHOO! Cloud Serving Benchmark (YCSB)

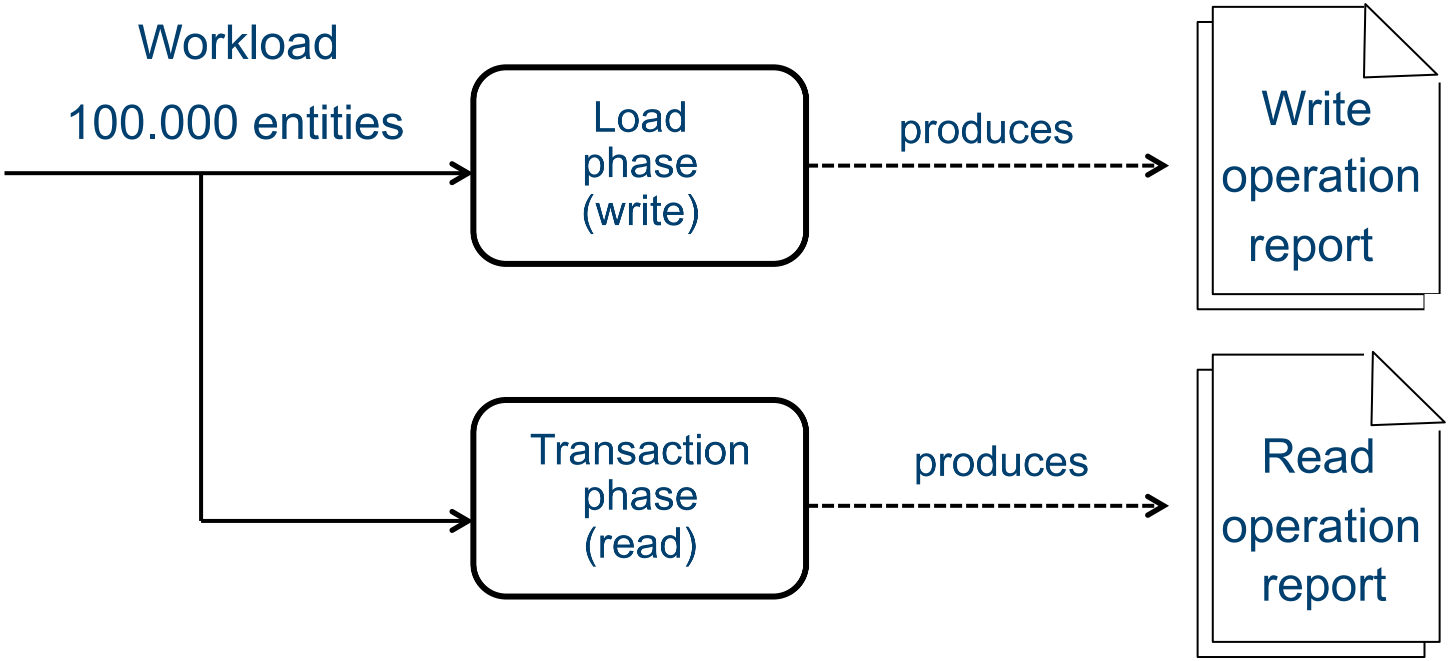
- YCSB client for operations through Kundera
- YCSB client for operations



Workload of 100.000 operations, splitted in two phases (write/read) on remote instances of Google Datastore and Azure Tables.

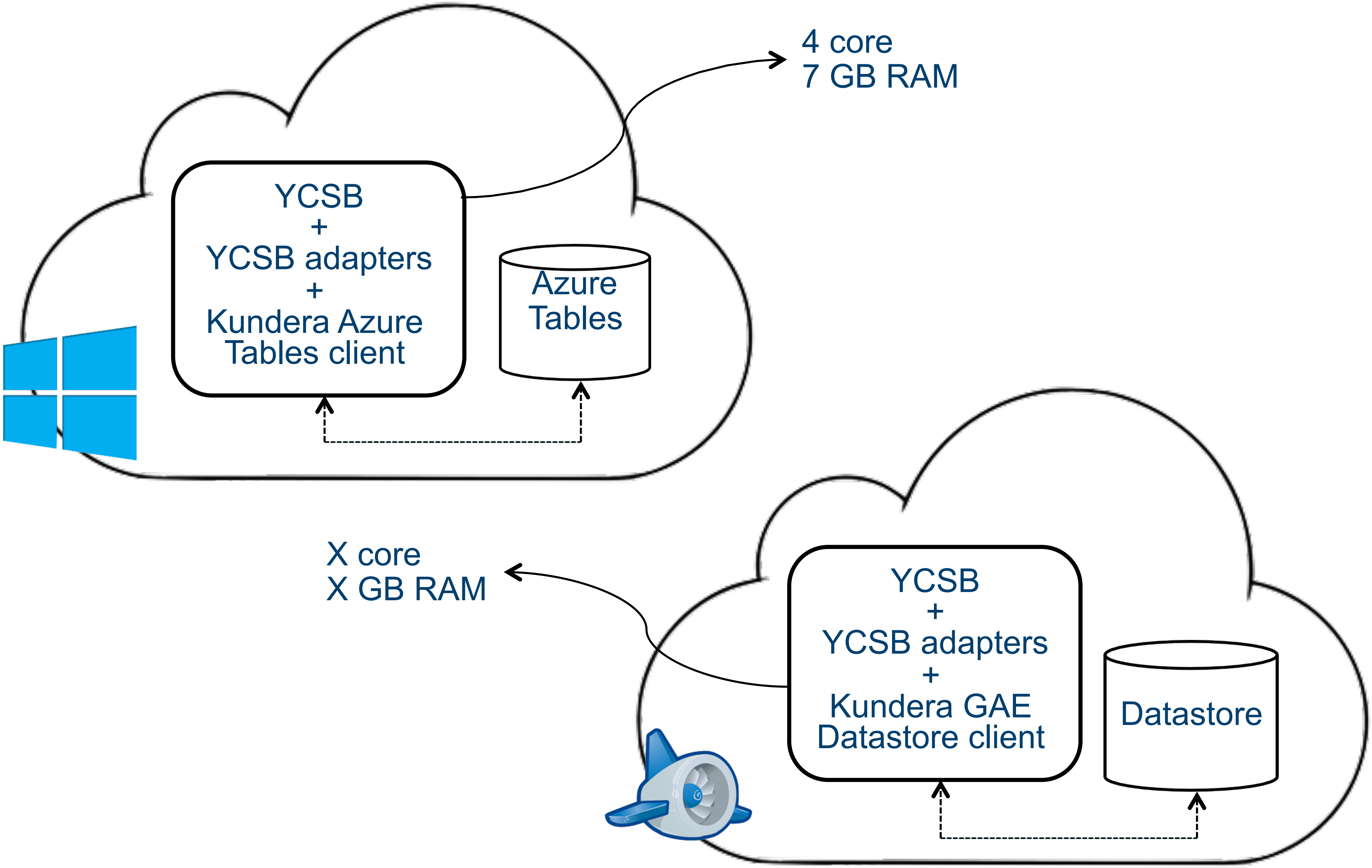
YAHOO! Cloud Serving Benchmark

Framework for evaluating the performance of different NoSQL databases



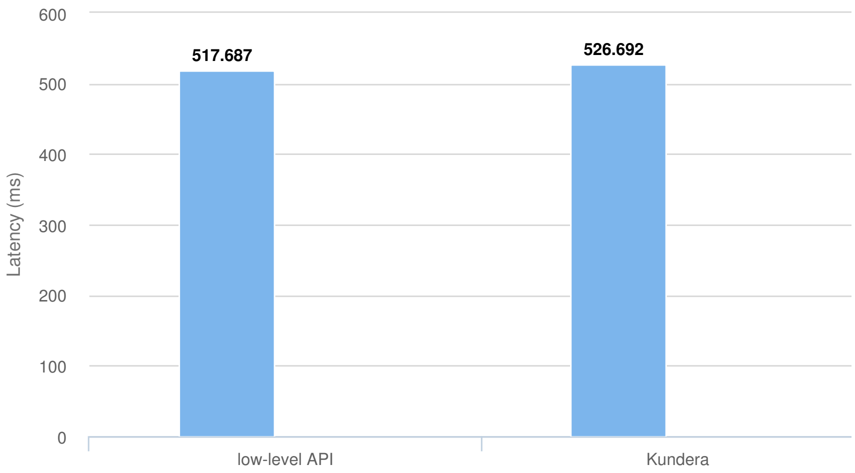
- Development of new adapter for operations through Kundera
- Development of new adapter for operations through the low-level API

Test environment

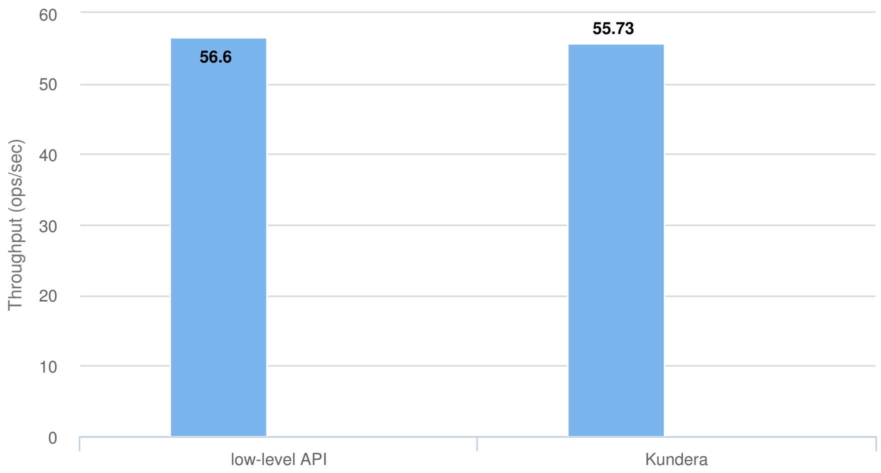


Results - Azure Tables

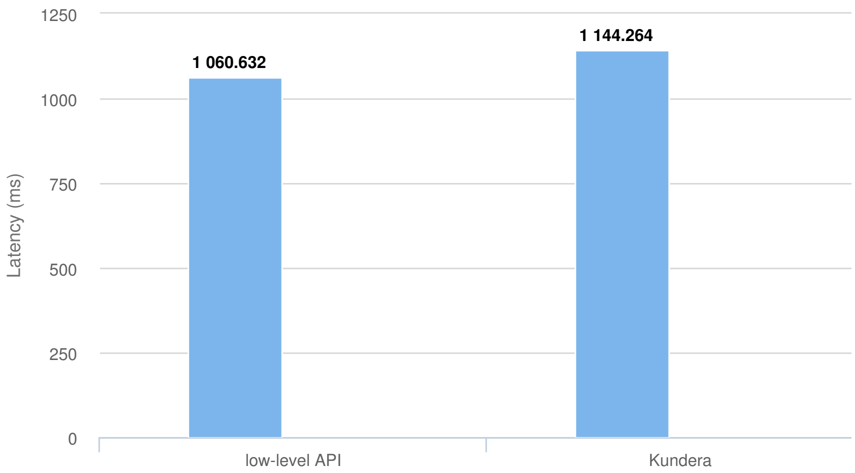
Read latency



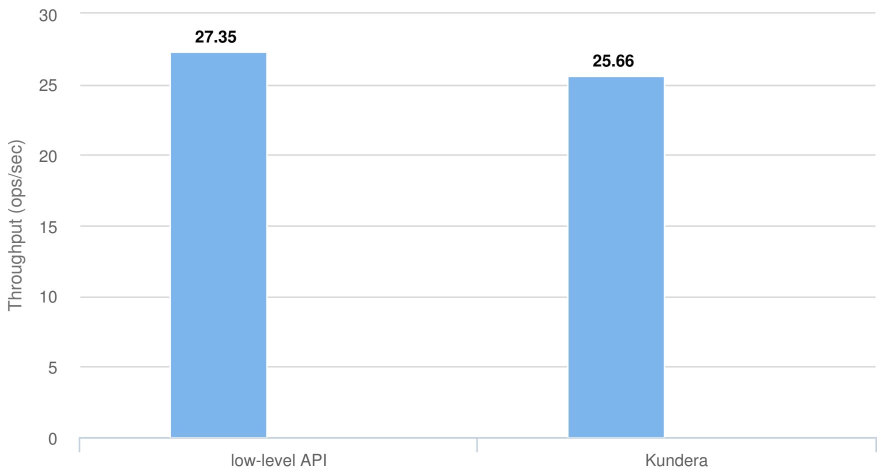
Read throughput



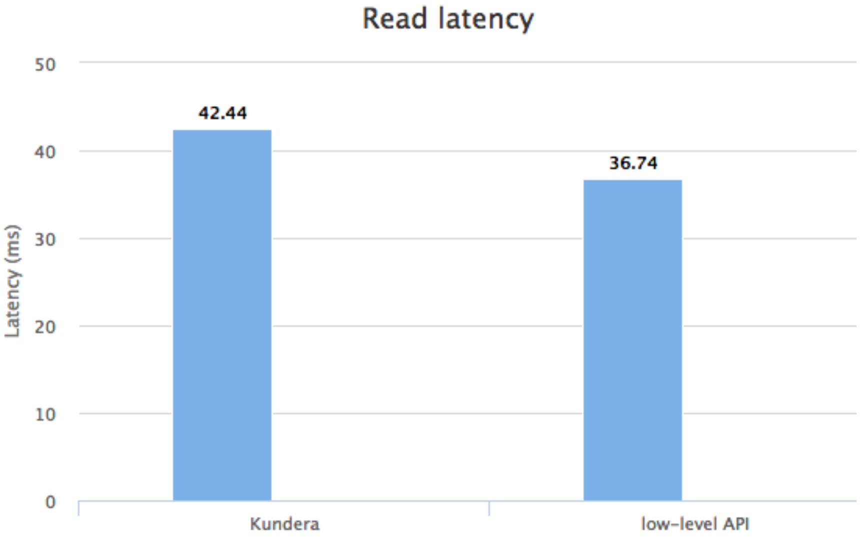
Write latency

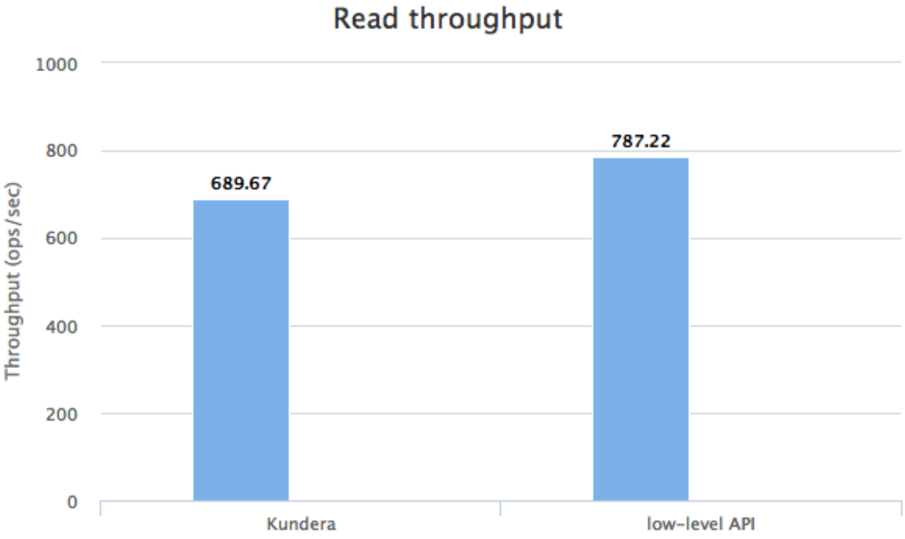


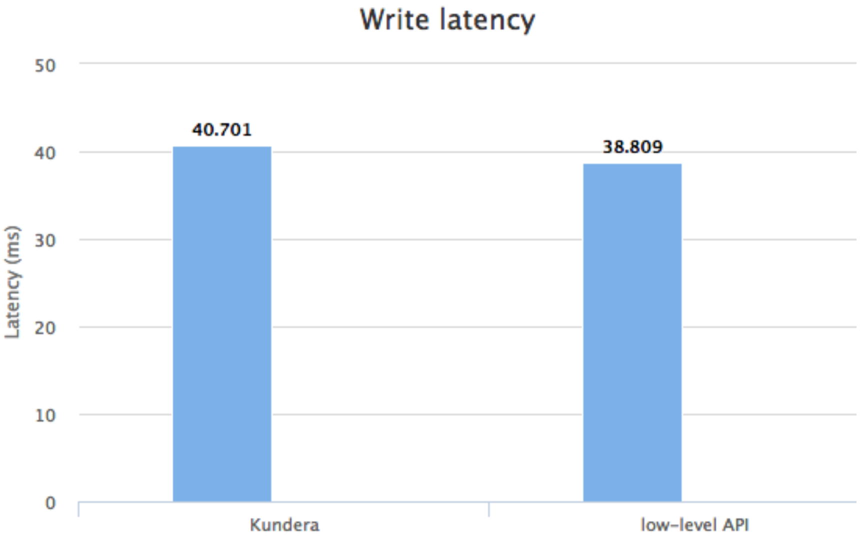
Write throughput

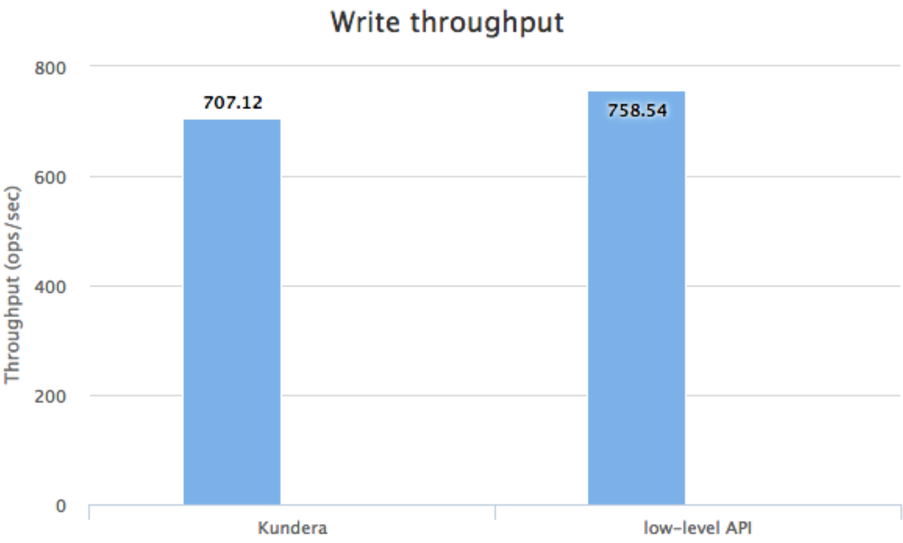


Results - Azure Tables

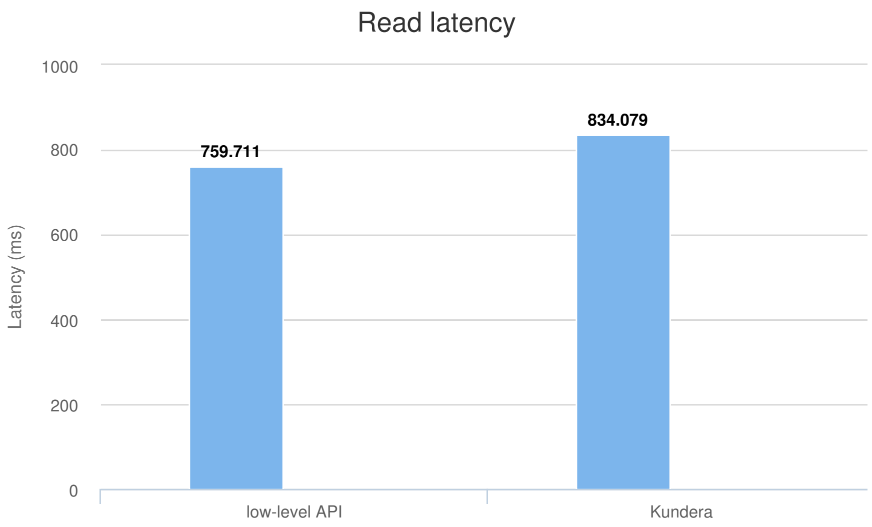


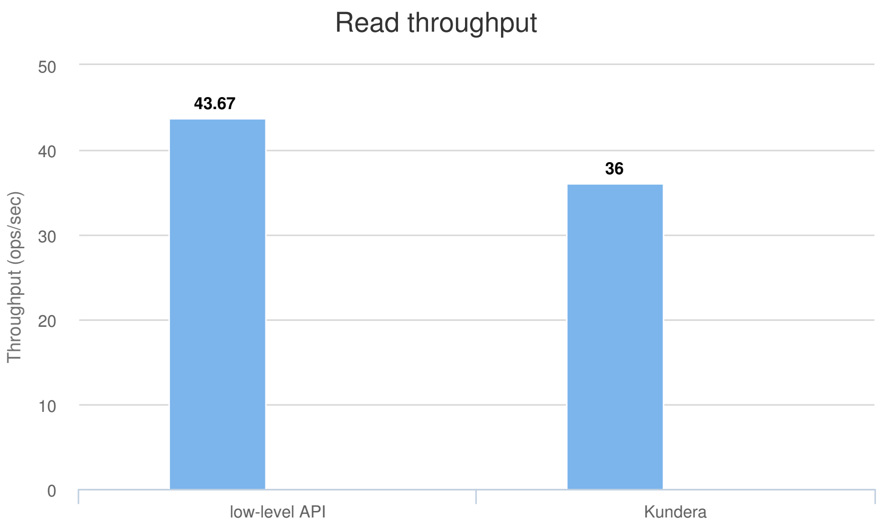


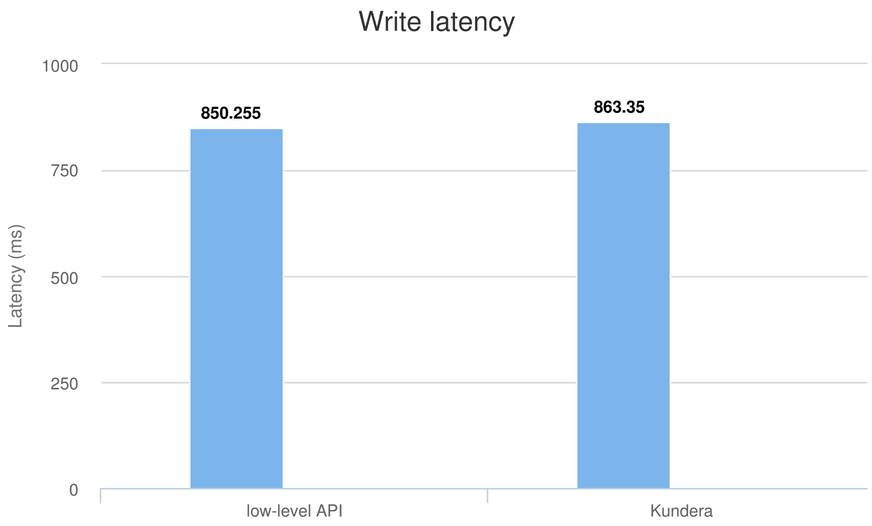


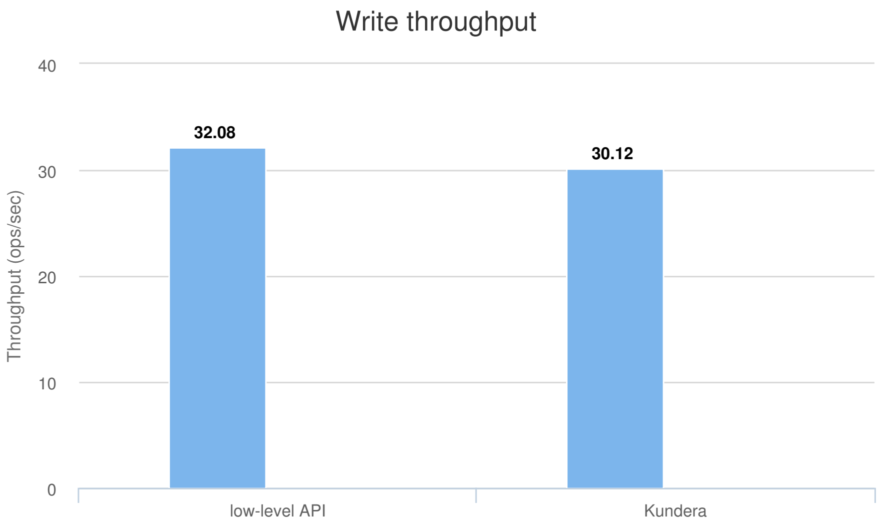


Results - GAE Datastore


Read latency


Read throughput


Write latency


Write throughput

Results - GAE Datastore

Results comparison

Azure Tables

Kunedra w.r.t low-level API

Read latency	Read throughput	Write latency	Write throughput
-13,43 %	- 12,39 %	- 4,75 %	- 6,78 %

Google Datastore

Kunedra w.r.t low-level API

Read latency	Read throughput	Write latency	Write throughput

Conclusions

Contributions:

- New clients for Kundera to support Google Datastore and Azure Tables
- Hegira integration in the CPIM library

Future work:

- Extend the CPIM library to support more cloud providers and/or new cloud services
- Develop new extensions for Kundera to support more NoSQL technologies
- Compare developed client performance with the ones of the other client developed by Kundera team

THANK YOU