

Scuola di Ingegneria Industriale e dell'Informazione

Corso di Laurea Magistrale in Ingegneria Informatica

Anno Accademico 2013 - 2014

 POLITECNICO DI MILANO



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 systems

RDBMS

Well structured data

Vertical scaling

ACID transactions

Relational model

SQL

NoSQL

Non-structured data

Horizontal scaling

BASE properties

Various data models

Proprietary API

NoSQL Common language approaches

Meta-model

- Apache MetaModel
- SOS platform

SQLification

- Apache Phoenix
- UnQL
- Native support

ORM

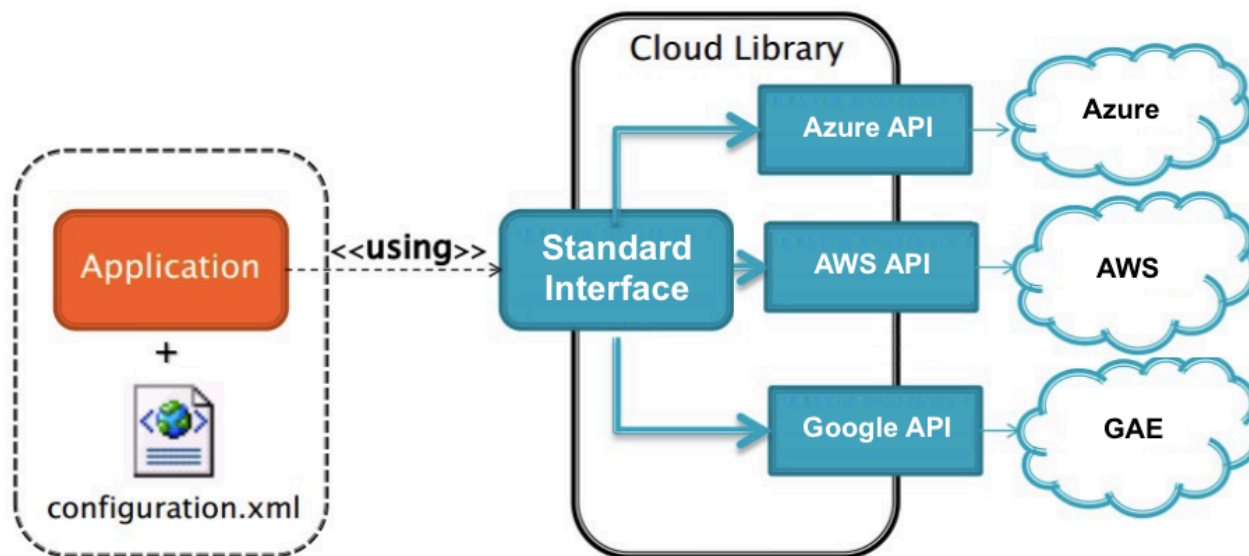
- Kundera
- PlayORM
- Spring-data
- Apache Gora

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

Contribute to the open source project Kundera

Integrate the migration and synchronization system Hegira

Evaluation

Work objectives

Integrate Kundera in the CPIM library

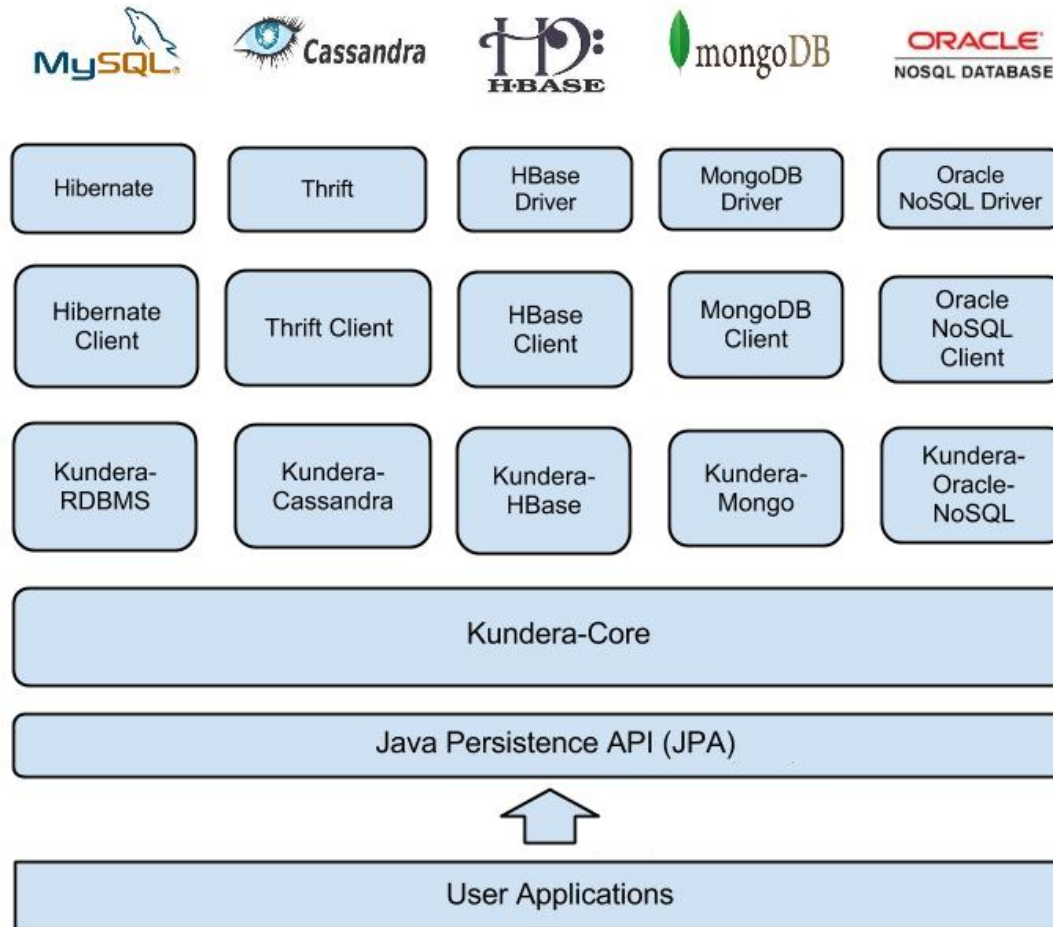
Contribute to the open source project Kundera

Integrate the migration and synchronization system Hegira

Evaluation

Kundera

A JPA 2.1 compliant ORM Library for NoSQL databases



Why Kundera

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

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
- 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

Contribute to the open source project Kundera

Integrate the migration and synchronization system Hegira

Evaluation

Contributions to Kundera

Paradigm shift

- support for DaaS
-  **Merged** Bug fix Kundera deploy on PaaS

Open Two newly developed clients

- Azure Tables¹
- GAE Datastore²

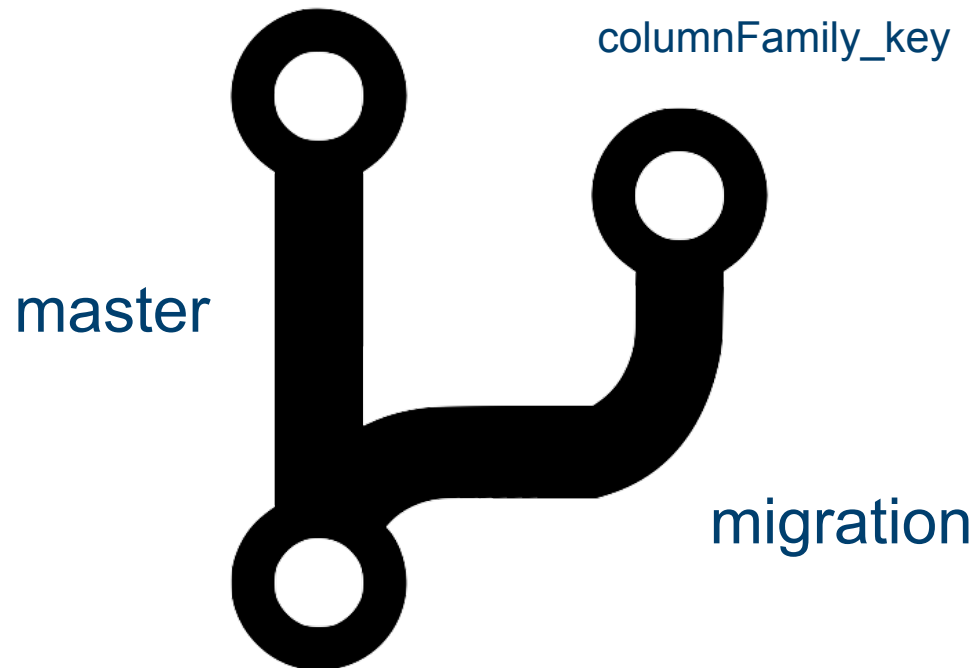
1: <https://github.com/deib-polimi/kundera-azure-table>

2: <https://github.com/deib-polimi/kundera-gae-datastore>

Developed clients

GAE Datastore	Azure Tables
No ancestor path support	Full support to partition key and row key
Key(table, id)	partitionKey_rowKey

GAE Datastore	Azure Tables
No ancestor path support	Partition key bounded to table name
columnFamily_key	columnFamily_key



Work objectives

Integrate Kundera in the CPIM library

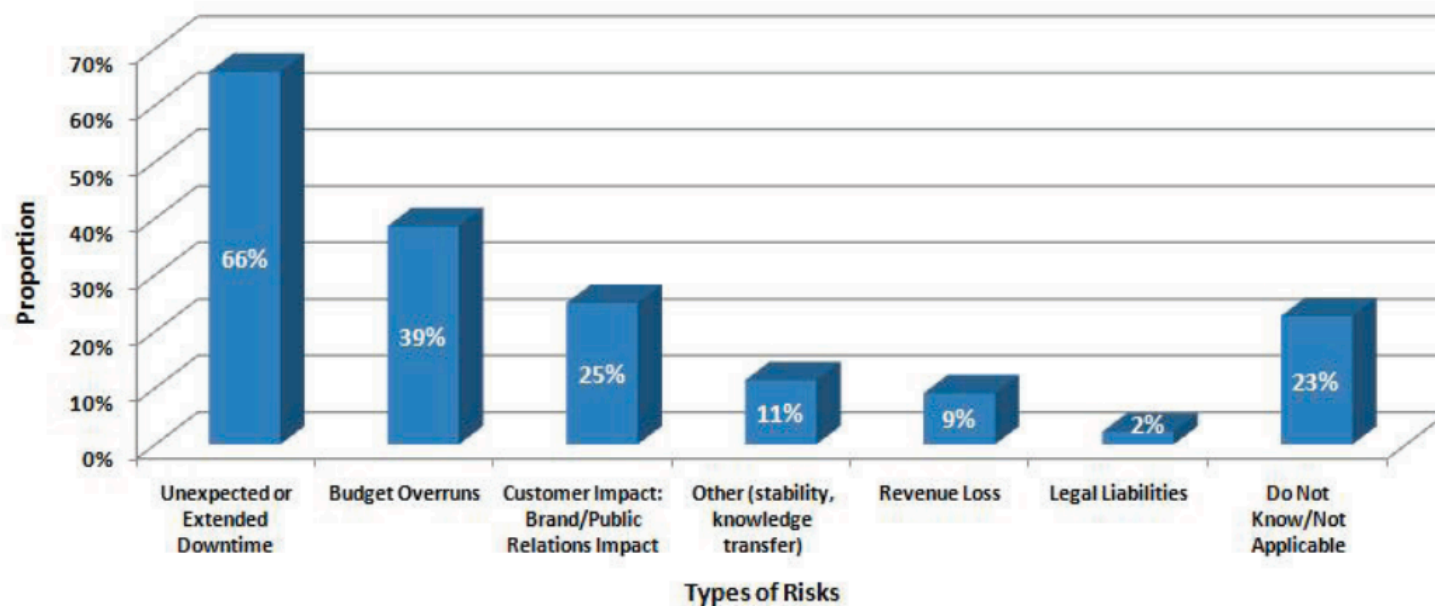
Contribute to the open source project Kundera

Integrate the migration and synchronization system Hegira

Evaluation

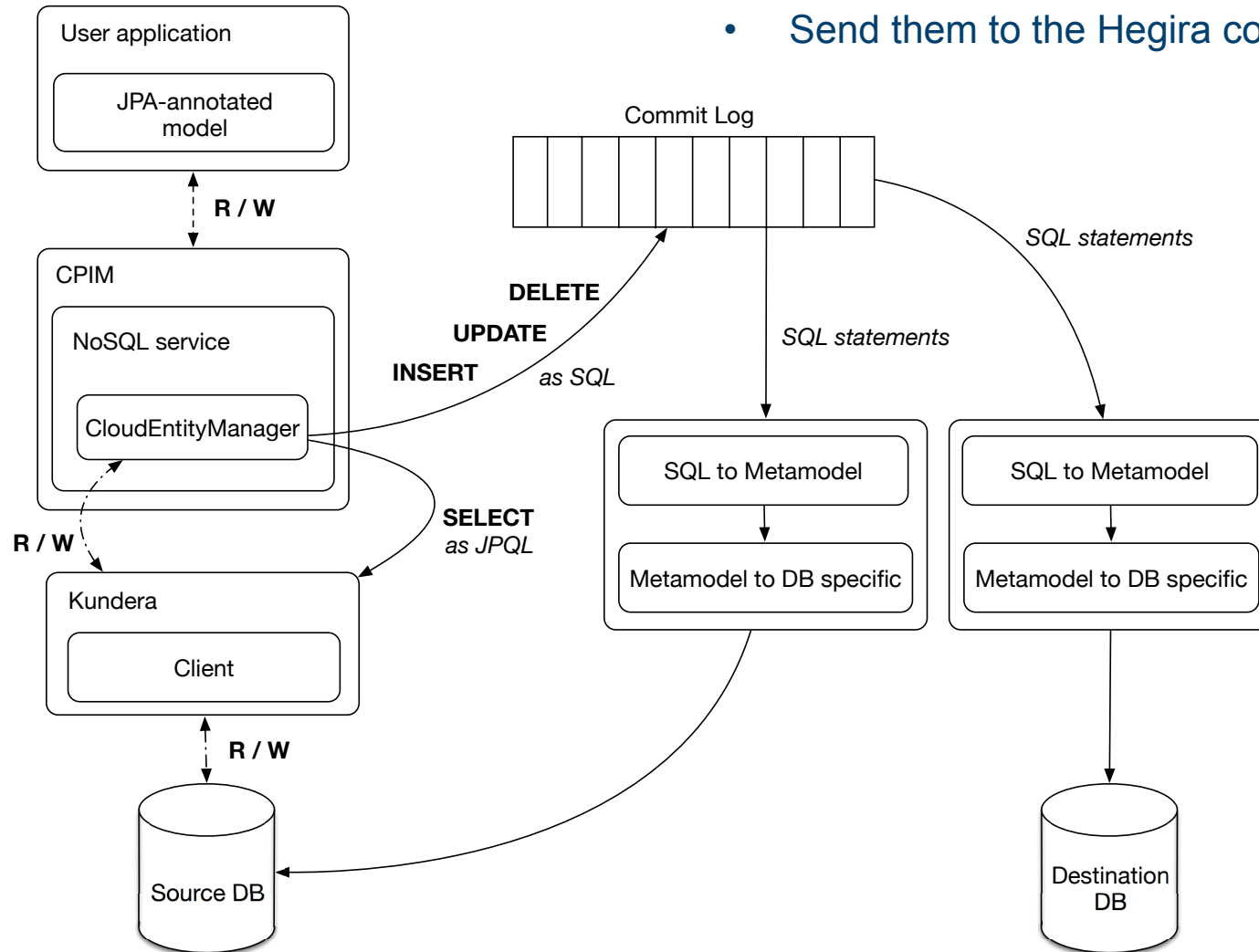
Data migration

- 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

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



Work objectives

Integrate Kundera in the CPIM library

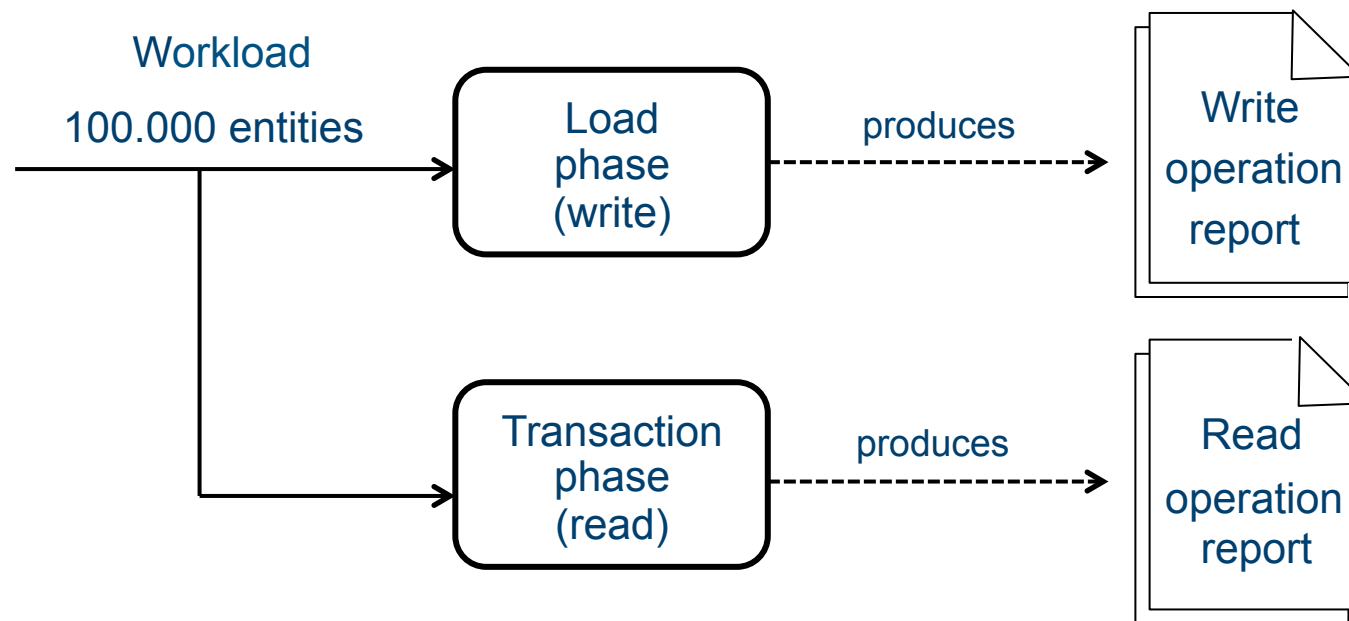
Contribute to the open source project Kundera

Integrate the migration and synchronization system Hegira

Evaluation

YAHOO! Cloud Serving Benchmark

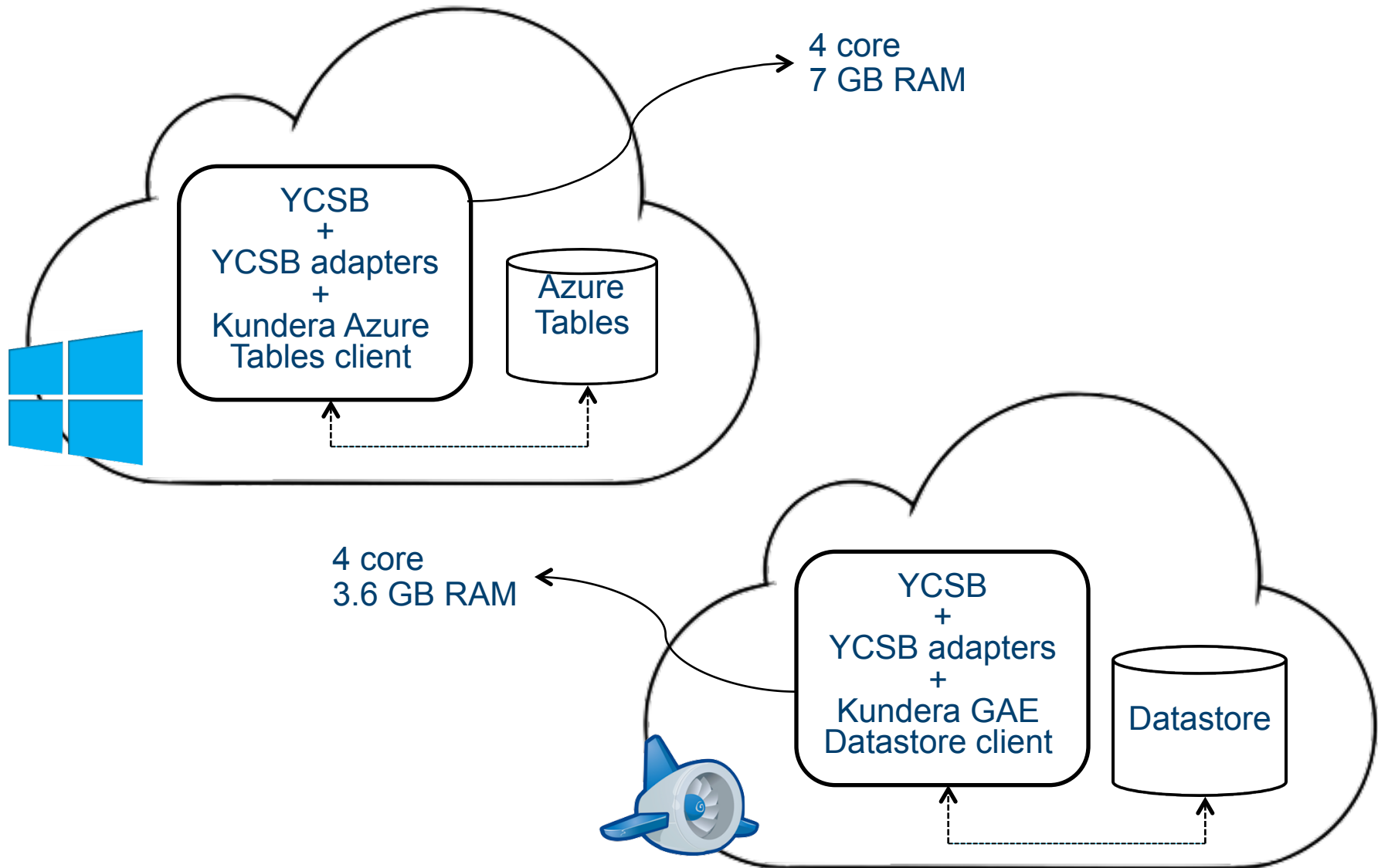
Framework for evaluating the performance of different NoSQL databases



Compare Kundera client w.r.t. the use of low-level API for the same operations

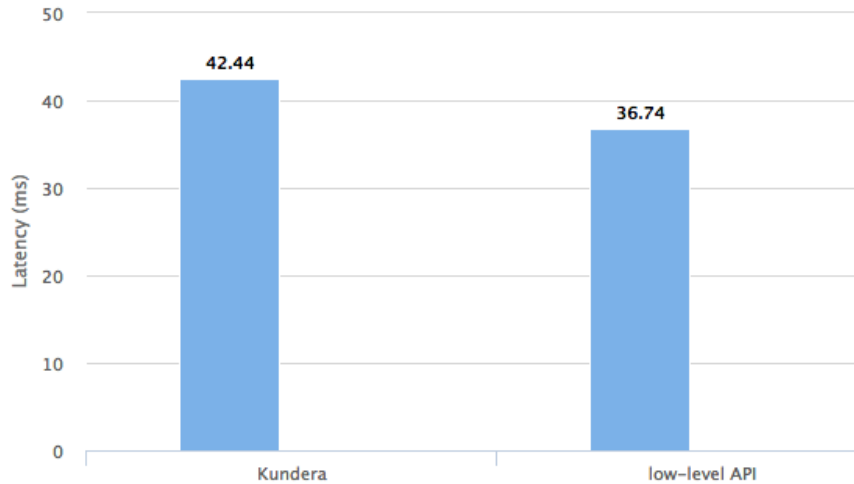
- Development of new adapter for operations through Kundera
- Development of new adapter for operations through the low-level API

Environment setup

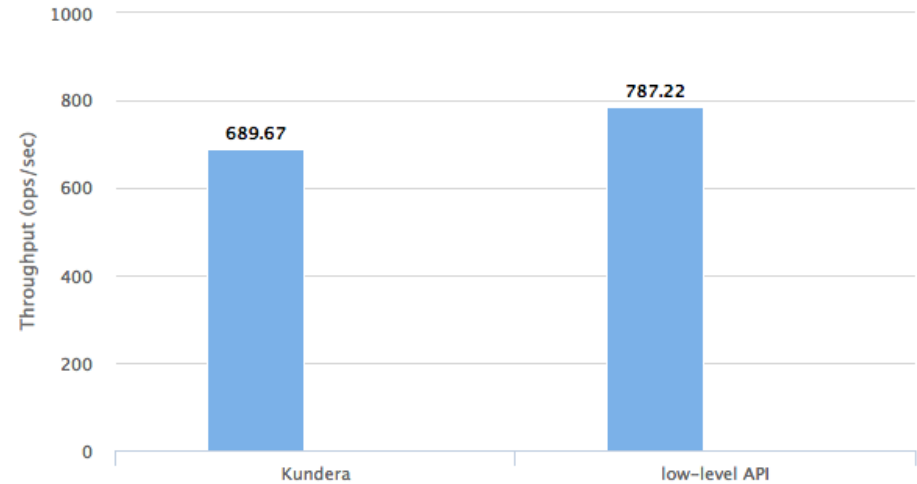


Results - Azure Tables

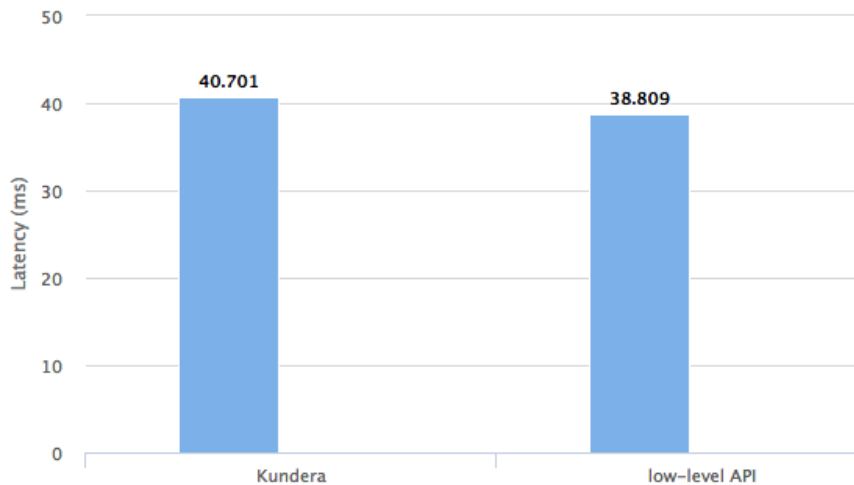
Read latency



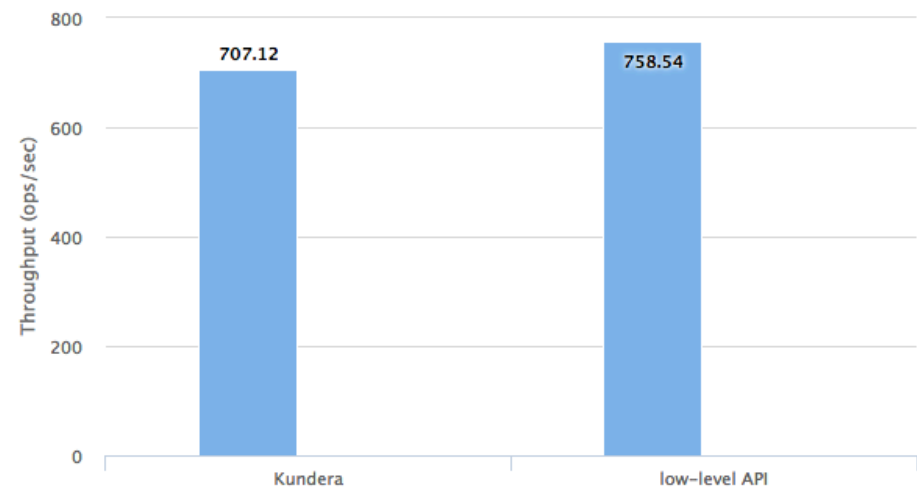
Read throughput



Write latency

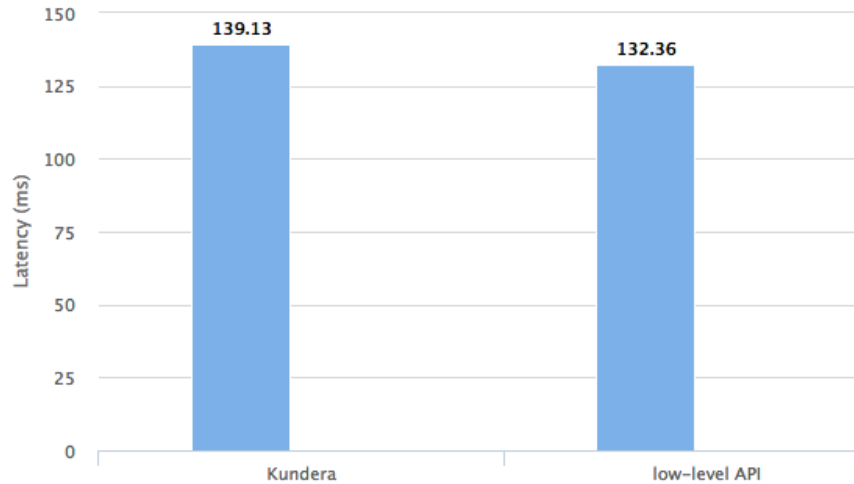


Write throughput

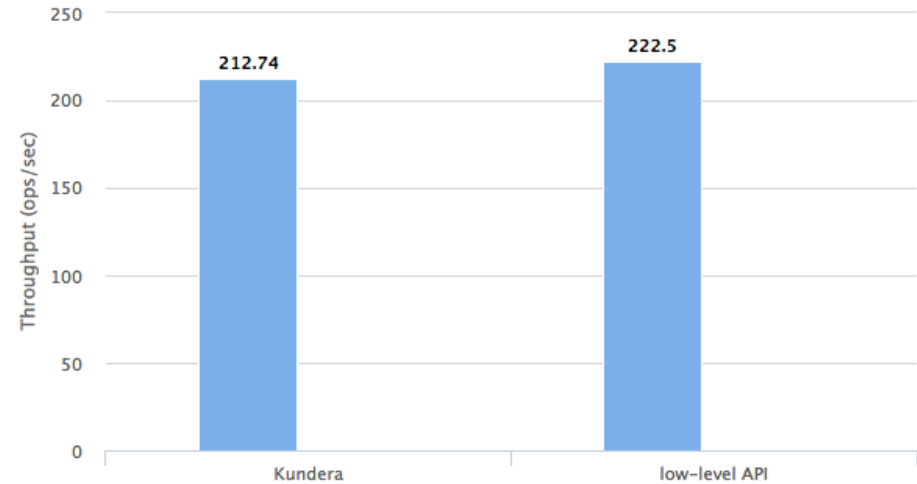


Results - GAE Datastore

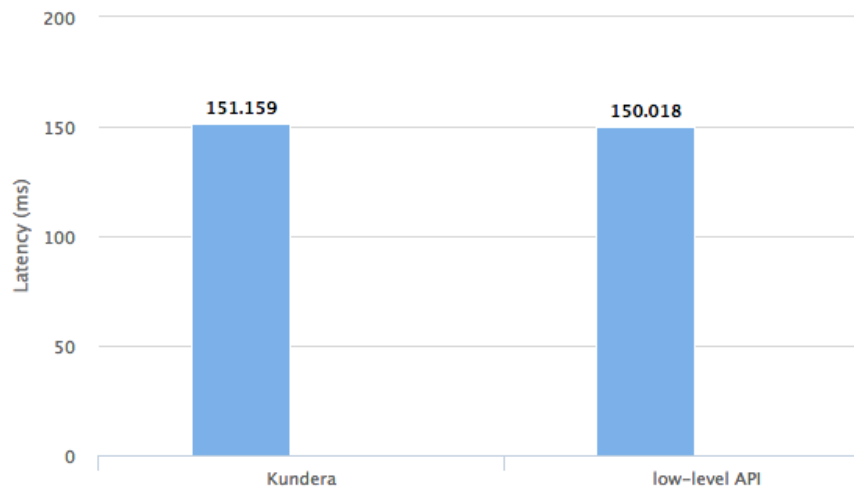
Read Latency



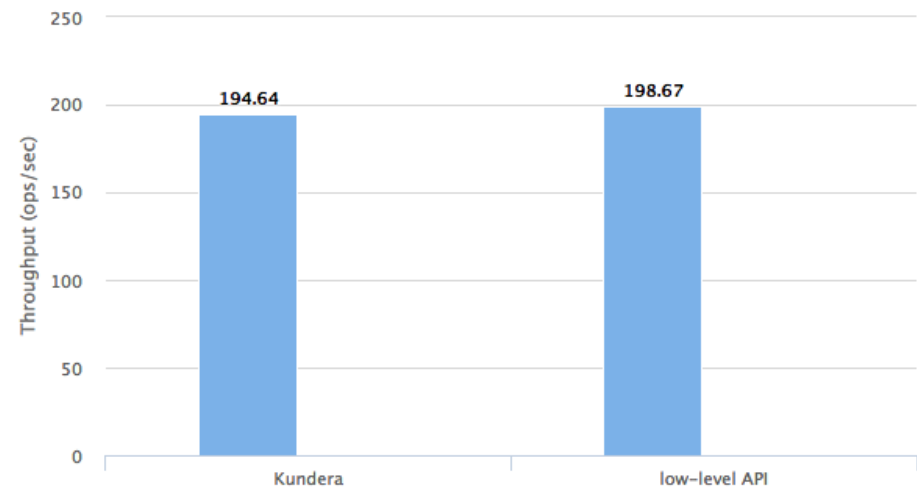
Read Throughput



Write Latency



Write Throughput



Results comparison

Azure Tables

Kunedra overhead 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 overhead w.r.t low-level API

Read latency	Read throughput	Write latency	Write throughput
4,36 %	4,39 %	0,76 %	2,03 %

Conclusions

Contributions:

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

Future work:

- Compare developed client performance with the ones of the other client developed by Kundera team

THANK YOU