SCALABLE MACHINE LEARNING WITH APACHE IGNITE, PYTHON, AND JULIA: FROM PROTOTYPE TO PRODUCTION

Created by Peter Gagarinov and Ilya Roublev

May 21, 2021

ALLIEDIUM AISSISTANT[1]: ABOUT THE PROJECT

- Makes the project management easier via automating the ticket assignment, labeling, ranking by priority
- Uses ML to infer rules from existing Jira tickets

OVERVIEW

- What is JIRA app?
- Alliedium AIssistant backend design paradigms, requirements to the underlying database
- The legacy backend architecture vs the current backend architecture
- PostgresSQL + Celery vs Apache Ignite + Ray Serve as both the database and computing grid: cons and pros for our use case
- Alliedium Apache Ignite Migration Tool: features and assumptions
- Python vs Julia as Apache Ignite ML alternative

WHY JIRA?

- Profitable for plugin developers: license cost depends on number of all users even if they do not use the plugin
- Very popular — millions of users around the globe

ALLIEDIUM AISSISTANT BACKEND DESIGN PARADIGMS

- SaaS built using microservice architecture
- Container orchestration
- Cloud-based fail-safe distributed architecture
- Scalable key-value database with SQL layer
- Multitenancy
- Background task manager
- Internal ML engine as a service
- Should support both cloud and on-premise deployment

DATABASE REQUIREMENTS

- integrates with Java natively
- highly available and horizontally scalable
- fault-tolerant and distributed
- supports distributed ACID transactions
- provides both persistent and in-memory storage
- supports SQL for distributed data

DATABASE REQUIREMENTS (CONTINUED...)

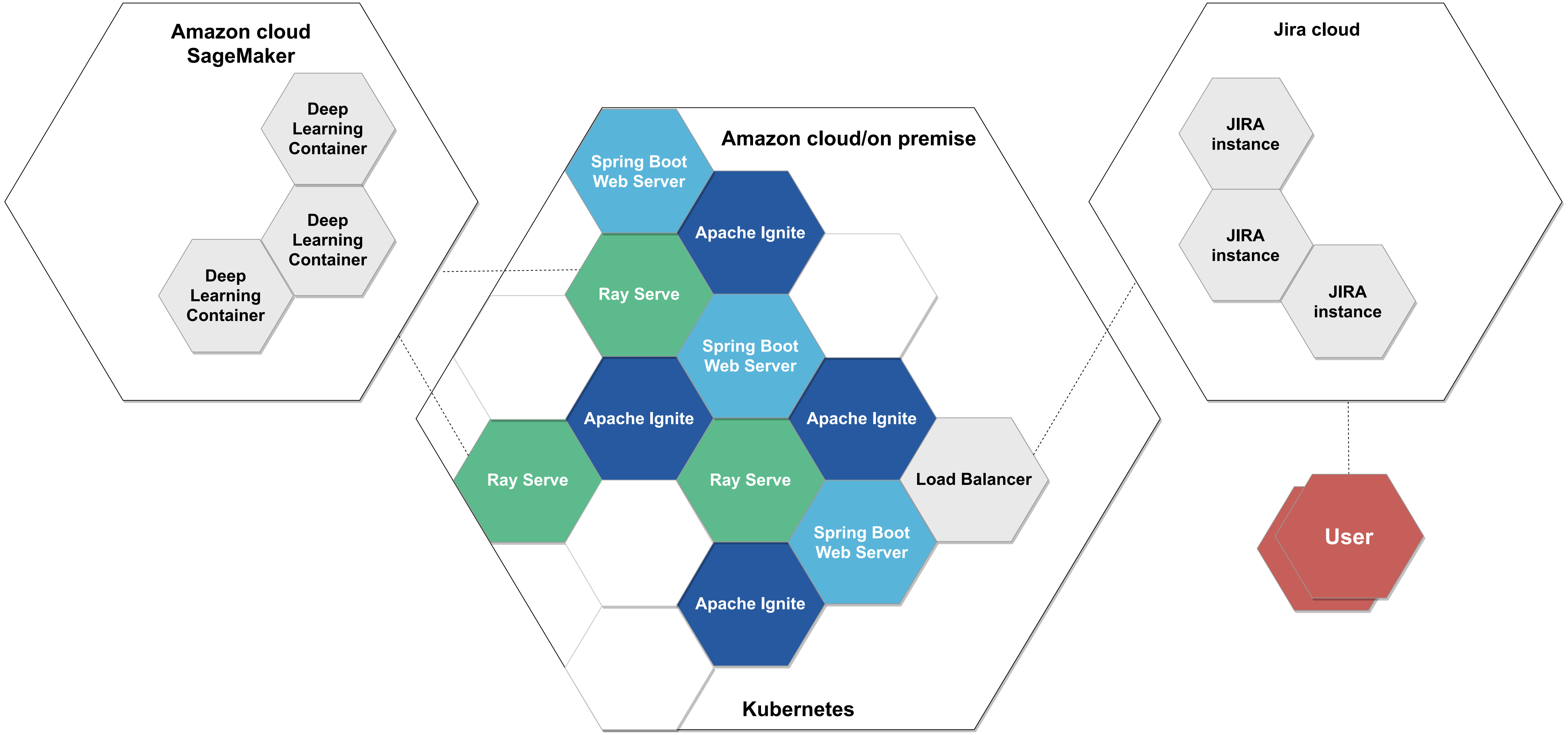
- supports user-defined distributed jobs
- provides automatic failover (jobs and db connections)
- provides Transparent Data Encryption for safety reasons
- supports native configurations for deployment in Kubernetes
- free and open-source

INITIAL TECHNOLOGY STACK

- Spring Boot as a web framework
- PostgreSQL as a database
- Hibernate as an ORM tool
- Celery + RabbitMQ as a computing grid[2]
- Scikit-Learn as an ML framework (runs inside Celery)[3]

CURRENT TECHNOLOGY STACK

- Spring Boot as a web framework
- Apache Ignite as a distributed database, no ORM is used
- Celery + RabbitMQ → Apache Ignite + Ray Serve[4][5][6]
- Scikit-Learn + PyTorch[7][8]


Amazon cloud
SageMaker
Deep Learning Container
Deep Learning Container
Deep Learning Container
Amazon cloud/on premise
Spring Boot Web Server
Apache Ignite
Ray Serve
Spring Boot Web Server
Apache Ignite
Apache Ignite
Ray Serve
Ray Serve
Load Balancer
Spring Boot Web Server
Apache Ignite
Kubernetes
Jira cloud
JIRA instance
JIRA instance
JIRA instance
User

POSTGRESQL: GOOD

- Easy to deploy[9]
- Easy to integrate with Atlassian Connect Spring Boot[10]
- Easy to version track schema changes and perform data migrations[11][12]
- supports most of the major features of ANSI SQL:2016 (starting with PostgreSQL 12) [13] [14]
- Full support for ACID transactions

POSTGRESQL: NOT SO GOOD

- Not horizontally scalable (unless some PostgreSQL-derivative database is used) [15] [16] [17] [18]
- Requires more efforts for mapping objects to tables
- Key-value API needs to be imitated via

```
select value from some_table where key = some_key
```

- Transparent Data Encryption is available only via an unofficial patch[19][20]
- In-memory tables: approximation only (RAM disk, UNLOGGED)[21][22][23][24]

APACHE IGNITE AS A DATABASE: GOOD

- Thick client for Java providing a full set of APIs
- Both key-value and SQL API
- Distributed
- Native persistence
- Full support for distributed ACID transactions[25]
- Built-in Transparent Data Encryption
- In-memory caches
- Good integration with Kubernetes
- Automatic connection failover for both thick and thin clients

APACHE IGNITE AS A DATABASE: NOT SO GOOD

- No open-source schema version tracking and data migration tools
- Database backup/restore is difficult[26][27]
- Still supports only a subset of ANSI SQL:1999 (e.g. no foreign keys)[28]
- SQL transactions are still in beta[29]
- Doesn't play nicely with Spring Boot DevTools[30][31][32]
- Requires network isolation for development purposes: https://github.com/Alliedium/arch-network-isolator[33]
- Python thin client doesn't yet support transactions[34]
- Using the thick client API[35] from Python requires Py4J Python-Java bridge[36]

ALLIEDIUM APACHE IGNITE MIGRATION TOOL: FEATURES

- Open-source (Apache License 2.0): https://github.com/Alliedium/ignite-migration-tool
- The data migration is performed in 3 stages:
  - exporting data and meta data from a live Apache Ignite cluster into an isolated filesystem directory in form of Avro files
  - applying database schema transformations to the exported data and writing the transformed data into a separate filesystem directory
  - uploading the transformed Avro files to the new cluster
- Data and metadata transformations are defined in a way that is Avro format agnostic

ALLIEDIUM APACHE IGNITE MIGRATION TOOL: FEATURES

- Data and metadata transformations are applied to Avro files on disk and do not require a live Apache Ignite cluster
- The tool can be used for creating data backups that are Apache Ignite version independent (assuming definitions of QueryEntity, CacheConfiguration and AffinityKey classes are stable)
- List of supported cache value field datatypes is limited by those allowed in QueryEntity (see https://ignite.apache.org/docs/latest/sql-reference/data-types)
- Cache keys can be of arbitrary non-user defined Java types and AffinityKey of such
- Source and target cluster topologies do not have to be the same
- Encrypted caches are supported

ALLIEDIUM APACHE IGNITE MIGRATION TOOL: ASSUMPTIONS

- Source and target should be different clusters
- Transformed data class definitions should be available at all target cluster nodes
- Each cache is configured with QueryEntity (field not present in QueryEntity definition are invisible to the tool)
- In-memory caches are backed up along with the persisted caches

CELERY: GOOD

- Python-based — easier to integrate with Python-based ML frameworks
- "At Least Once" delivery guarantee for Celery message queues (implemented via RabbitMQ)[38]

CELERY: NOT SO GOOD

- Requires a separate message broker (RabbitMQ) for submitting tasks[39]
- Requires a separate results backend for large results[39]
- No out-of-the-box pure Java API[40]
- If not run inside K8s a special care is needed for RabbitMQ auto-failover implementation[41]
- Automatic connection failover is available only inside Kubernetes

APACHE IGNITE AS A COMPUTING GRID: GOOD

- Native Java API for messages and distributed computing tasks
- Built-in distributed basic ML models
- Automatic connection failover for both thick and thin clients

APACHE IGNITE AS A COMPUTING GRID: NOT SO GOOD

- Weaker delivery guarantees — not suitable for important messages (in finance e.g.)[42]
- Built-in ML models lack certain features for our use case
- Python thin client doesn't support neither message nor computing API[34][43]
- Using the thick client API from Python requires Py4J Python-Java bridge[36]

PYTHON VS JULIA AS APACHE IGNITE ML ALTERNATIVE

- ML in both Julia and Python is much faster than Apache Ignite ML exactly for our case (~8-10k observations)

Apache Ignite ML

```
LogisticRegressionModel lrClassifier =
    new LogisticRegressionSGDTrainer()
    .fit(...);
DecisionTreeModel dtClassifier =
    new DecisionTreeClassificationTrainer()
    .fit(...);
```

MLJ[48] (Julia)

```
lr_classifier = LogisticClassifier(...)
lr_mach = machine(
    lr_classifier, ...) |> fit!
dt_classifier = (
    @load DecisionTreeClassifier pkg=DecisionTree)(...)
dt_mach = machine(
    dt_classifier, ...) |> fit!
```

scikit-learn[3] (Python)

```
lr_classifier = LogisticRegression(...)
lr_classifier.fit(...)
dt_classifier = DecisionTreeClassifier(...)
dt_classifier.fit(...)
```

Fit time + Cross-validation time (10 folds)

	Apache Ignite ML	MLJ	scikit-learn
Linear Regression (SAG)	5.438 sec+40.237 sec		0.066 sec+0.534 sec
Linear Regression (LBFGS)		0.196 sec+1.372 sec	0.082 sec+0.551 sec
Decision Tree	1.664 sec+12.259 sec	0.146 sec+1.465 sec	0.197 sec+1.755 sec

Ignite ver. 2.10.0 (1 Ignite node), Julia ver. 1.6.1 (MLJ v0.16.4, MLJLinearModels v0.5.4, DecisionTree v0.10.10), Python ver. 3.8.6 (scikit-learn v0.23.2), Windows 10, 32 GB RAM, Intel(R) Core(TM) i7-8700K CPU @ 3.70GHz, Data: a subset of Fraud Detection dataset[49][50] (7936 rows, 30 columns, 2 classes)

- A limited set of opt. solvers in Apache Ignite ML (e.g. LogisticRegressionSGDTrainer for LogisticRegressionModel, in scikit-learn — 5 solvers)
- No nested cross-validation[51], no stratified cross-validation[52] in Apache Ignite ML

PYTHON VS JULIA AS APACHE IGNITE ML ALTERNATIVE

WHERE PYTHON > JULIA

- Python Ignite thin client, no such client for Julia
- Ray Serve[4][5][6] (e.g. Genie.jl + Dagger.jl is not an equivalent replacement)
- Python has a much more mature ML ecosystem comparing to Julia
- scikit-learn is sometimes faster than MLJ

WHERE PYTHON = JULIA

- Calling Apache Ignite thick client Java API: Py4J[36] (Python) vs JavaCall.jl[53] (Julia)
- Calling Apache Ignite thick client C++ API: Cython[54] (Python) vs CxxWrap.jl[55] (Julia)

WHERE JULIA < PYTHON

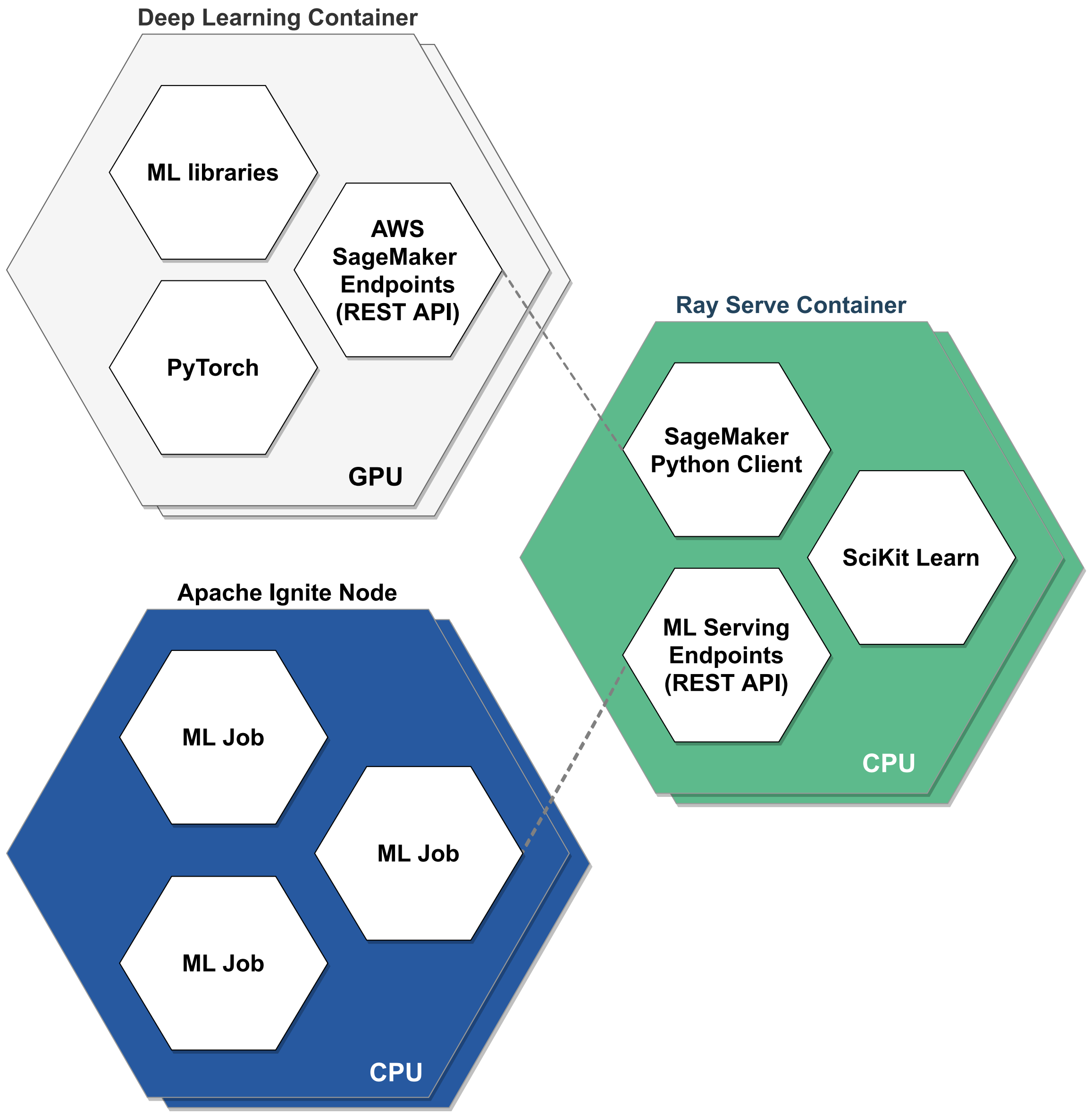
- Julia is more flexible
- Easier parallelism (native threads in Julia vs GIL[56] in Python)

POSTGRESQL → APACHE IGNITE: MIGRATION DIFFICULTIES

- Apache Ignite cache imitating `atlassian_host` table needs to be created prior to starting Atlassian Connect Spring Boot[44]
- Fields having non-SQL datatypes (custom class-valued fields) need to be stored as XML (via Binarylizable[45] and QueryEntity[46]) to be readable in SQL client tools such as DBeaver and DataGrip
- Still not possible to get the list of all atomics names inside the cluster[47]

CELERY + RABBITMQ → APACHE IGNITE: MIGRATION DIFFICULTIES

- Still need a place to run Python-based ML calculations, that is why Ray Serve
- More care on the front-end is required due to no delivery guarantees


Deep Learning Container
ML libraries
AWS SageMaker Endpoints (REST API)
PyTorch
GPU
Ray Serve Container
SageMaker Python Client
SciKit Learn
ML Serving Endpoints (REST API)
CPU
Apache Ignite Node
ML Job
ML Job
ML Job
CPU

QUESTIONS?

REFERENCES

[1] Alliedium AIssistant Jira App, ver. 1.2, 2020

[2] Johansson Lovisa, Running Celery with RabbitMQ, www.cloudampq.com, 2019

[3] scikit-learn: Machine Learning in Python, scikit-learn.org, ver. 0.24, 2020

[4] Ray Serve: Scalable and Programmable Serving, docs.ray.io, ver. 1.2.0, 2021

[5] Mo Simon, Machine Learning Serving is Broken: And How Ray Serve Can Fix it, medium.com, 2020

[6] Oakes Edward, The Simplest Way to Serve your NLP Model in Production with Pure Python, medium.com, 2020

[7] How would you compare Scikit-learn with PyTorch?, www.quora.com, 2020

[8] K Dhiraj, Why PyTorch Is the Deep Learning Framework of the Future, medium.com, 2019

[9] Chiniara Dan, Installing PostgreSQL for Mac, Linux, and Windows, medium.com, 2019

REFERENCES

[10] Atlassian Connect Spring Boot, bitbucket.org, ver. 2.1.2, 2020

[11] Oliveira Junior, The best and easy way to handle database migrations (version control), medium.com, 2019

[12] Gopal Vineet, Move fast and migrate things: how we automated migrations in Postgres, medium.com, 2019

[13] PostgreSQL: Appendix D. SQL Conformance, www.postgresql.org, ver. 13.2, 2021

[14] PostgreSQL vs SQL Standard, wiki.postgresql.org, 2020

[15] Kuizinas Gajus, Lessons learned scaling PostgreSQL database to 1.2bn records/month: Choosing where to host the database, materialising data and using database as a job queue, medium.com, 2019

[16] Slot Marco, Why the RDBMS is the future of distributed databases, ft. Postgres and Citus, www.citiusdata.com, 2018

REFERENCES

[17] Knoldus Inc., Want to know about Greenplum?, medium.com, 2020

[18] TimescaleDB 2.0: A multi-node, petabyte-scale, completely free relational database for time-series, blog.timescale.com, 2020

[19] Chen Neil, Rise and Fall for an expected feature in PostgreSQL — Transparent Data Encryption, highgo.ca, 2020

[20] PostgreSQL Transparent Data Encryption, www.cybertec-postgresql.com, 2021

[21] Huang Cary, Approaches to Achieve in-Memory Table Storage with PostgreSQL Pluggable API, highgo.ca, 2020

[22] Westermann Daniel, Can I put my temporary tablespaces on a RAM disk with PostgreSQL?, blog.dbi-services.com, 2020

[23] PostgreSQL: 22.6. Tablespaces, www.postgresql.org, ver. 13.2, 2021

[24] Ringer Craig, Putting a PostgreSQL tablespace on a ramdisk risks ALL your data, 2014

REFERENCES

[25] Apache Ignite: ACID Transactions with Apache Ignite, ignite.apache.org, ver. 2.9.1, 2020

[26] Apache Ignite: Cluster Snapshots: Current Limitations, ignite.apache.org, ver. 2.9.1, 2020

[27] Ignite in-memory + other SQL store without fully loading all data into Ignite, Apache Ignite Users, 2020

[28] SQL Conformance, ignite.apache.org, ver. 2.9.1, 2020

[29] Apache Ignite: SQL Transactions, ignite.apache.org, ver. 2.9.1, 2020

[30] Using Spring Boot: 8. Developer Tools, 8.2.7. Known Limitations, docs.spring.io, ver. 2.4.3, 2021

[31] ClassCastException while fetching data from IgniteCache (with custom persistent store), Apache Ignite Users, 2016

REFERENCES

[32] Spring Session and Dev Tools Cause ClassCastException, github.com/spring-projects/spring-boot, 2017

[33] Bhuiyan Shamim, A Simple Checklist for Apache Ignite Beginners (5. Ghost Nodes), dzone.com, 2019

[34] Apache Ignite: Thin Clients Overview, ignite.apache.org, ver. 2.9.1, 2020

[35] Kulichenko Valentin, Apache Ignite: Client Connectors Variety, dzone.com, 2020

[36] Py4J — A Bridge between Python and Java, py4j.org, ver. 0.10.9.2, 2021

[37] MavenRepository: Ignite Spring, ver. 2.9.1, 2020

[38] RabbitMQ: Reliability Guide, Acknowledgements and Confirms, ver. 3.8.13, 2021

[39] First Steps with Celery: Configuration, docs.celeryproject.org, ver. 5.0.5, 2020

[40] Celery: Message Protocol, docs.celeryproject.org, ver. 5.0.5, 2020

REFERENCES

[41] Paudice Genny, High availability with RabbitMQ, blexin.com, 2019

[42] Messaging Reliability, Apache Ignite Users, 2016

[43] Apache Ignite: Python Thin Client, ignite.apache.org, ver. 2.9.1, 2020

[44] Gagarinov Peter & Roublev Ilya, Boosting Jira Cloud app development with Apache Ignite, medium.com, 2020

[45] Apache Ignite JavaDoc: Interface Binarylizable, ignite.apache.org, ver. 2.9.1, 2020

[46] Apache Ignite: SQL API, Query Entities, ignite.apache.org, ver. 2.9.1, 2020

[47] Unable to query system cache through Visor console, Apache Ignite Users, 2017

[48] A Machine Learning Framework for Julia, alan-turing-institute.github.io, ver. 0.16.4, 2021

[49] Credit Card Fraud Detection dataset, kaggle.com, 2018

REFERENCES

[50] Chaudhri Akmal, Using Apache Ignite's Machine Learning for Fraud Detection at Scale, dzone.com, 2018

[51] Brownlee Jason, Nested Cross-Validation for Machine Learning with Python, machinelearningmastery.com, 2021

[52] Brownlee Jason, How to Fix k-Fold Cross-Validation for Imbalanced Classification, machinelearningmastery.com, 2020

[53] Call Java programs from Julia, juliainterop.github.io, ver. 0.7.7, 2021

[54] Using C++ in Cython, cython.readthedocs.io, ver. 0.29.6, 2019

[55] Janssens Bart, Wrapping a C++ library using CxxWrap.jl, JuliaCon, 2020

[56] Ajitsaria Abhinav, What Is the Python Global Interpreter Lock (GIL)?, realpython.com, 2018