



Aalto University  
School of Science

# GenAI/LLMs and Big Data Platforms

*Hong-Linh Truong*

*Department of Computer Science*

*[linh.truong@aalto.fi](mailto:linh.truong@aalto.fi)*, *<https://rdsea.github.io>*

# Learning objectives

- **Understand the role of GenAI/LLMs as services for big data platforms**
- **Able to support the development of GenAI/LLMs using techniques and methods studied from big data platforms**

# Up to now: what you have learned is very much about the past and the present

- **Big data platforms are complex**
  - we have studied basic, foundational techniques and models
  - we have practiced with existing platforms, and developed our own use cases/solutions
- **Trends and new requirements**
  - real time data and analytics
  - new concepts and platforms for data mesh, lakehouse, design patterns like medallion architectures
  - composibility and interoperability of modern data platforms

# GenAI/LLMs as a service

- **GenAI/Large Language Models (LLMs)**

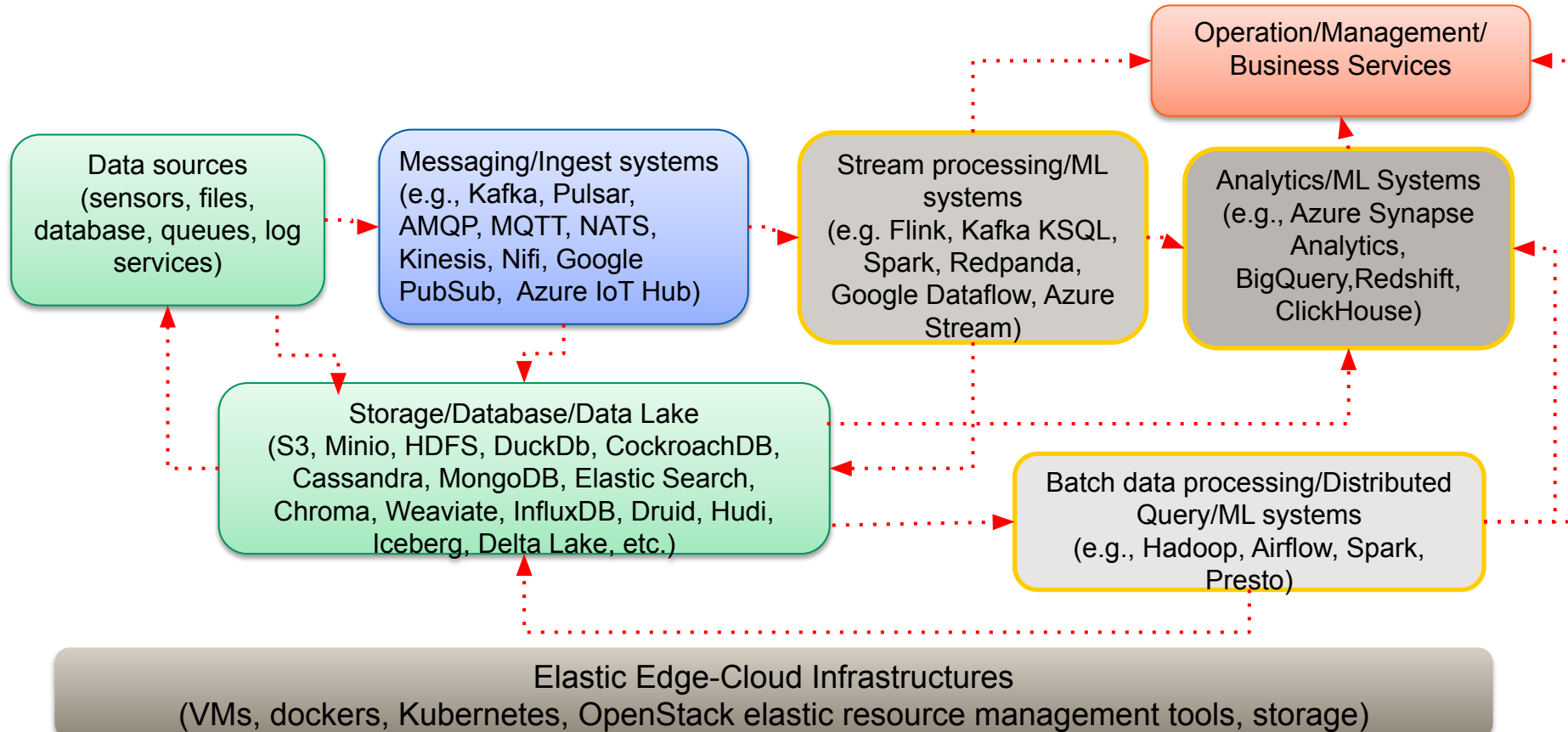
- using generative models to analyze and generate text, code, images, data, etc.
- generative models learn patterns and build their capabilities based on data
- LLMs: foundational models trained with a lot of data, able to handle and generate natural languages and other contents

- **AI Agents**

- software acting on behalf of the user/customer to plan and complete tasks for the user (goal-oriented)
- agent models built around/powered by GenAI/LLMs
  - *autonomy, proactiveness, collaboration, environment sensing, memorization, uncertainty*

# GenAI/LLMs for Data Platforms VS Data Platforms for GenAI/LLMs

# Where could GenAI/LLMs play a role?



# GenAI/LLMs & data platforms (1)

- **Exploring the benefits of GenAI/LLMs**
  - for the development and use cases in big data platforms
- **Integrating GenAI/LLMs services**
  - for platform requirement analysis, design and development
- **Providing GenAI/LLMs services as a service in the platform**
  - for data analysis, visualization, and transformation

# GenAI/LLMs & data platforms (2)

- **Big data platforms as backbone for AI/ML/LLMs**
  - building LLMs requires strong support from big data techniques, models and platforms
- **Leveraging our expertise in big data platforms for building GenAI/LLMs**
  - data pipeline, data services, data transformation, and data quality/governance



# GenAI/LLMs for low code in big data programming

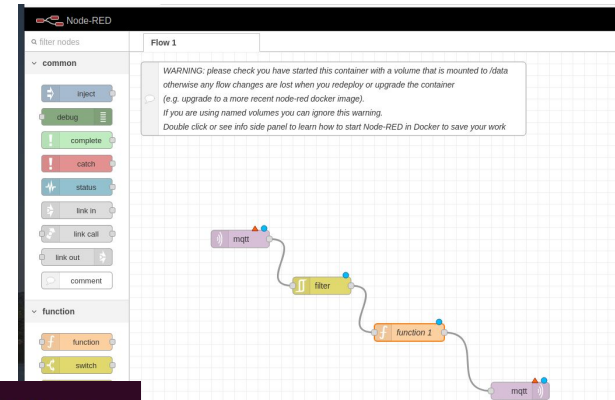
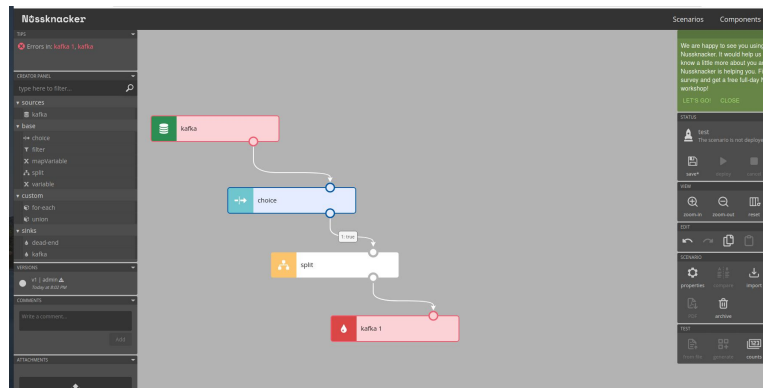
# What have we had? look back your code and examples in the tutorials

- **Developer's daily tasks:**
  - building lot of ETL, data analytics, etc.
- **The Nifi way:**
  - defining flows using drag and drop features
- **The Airflow-alike way:**
  - programing workflows with APIs and annotations
- **The Flink/Spark way:**
  - low-level data processing API
- **Multiple runtime systems and microservices:**
  - utilizing containers (e.g., Airbyte), mapping to different underlying runtimes (e.g., Apache Beam)

# Some snapshots

(a snapshot from Beam)

```
24 Returns:
25     The processed element.
26 """
27 text_line = element.strip()
28 if not text_line:
29     self.empty_line_counter.inc(1)
30 words = re.findall("[a-zA-Z]+", text_line, re.UNICODE)
31 for w in words:
32     self.words_counter.inc(1)
33     self.word_lengths_counter.inc(len(w))
34     self.word_lengths_dist.update(len(w))
35 return words
36
37 def main(argv=None, save_main_session=True):
38     """Main entry point; defines and runs the wordcount pipeline."""
39     parser = argparse.ArgumentParser()
40     parser.add_argument(
41         "--input",
42         default="gs://dataflow-samples/shakespeare/kinglear.txt",
43         help="Input file to process.",
44     )
45     parser.add_argument(
46         "--output",
47         default="output",
48         help="Output file to write.",
49     )
50     parser.add_argument(
51         "--runner",
52         default="DataflowRunner",
53         help="Runner to use.",
54     )
55     args = parser.parse_args(argv)
56     pipeline_options = PipelineOptions(args.runner, args.input, args.output, args.runner)
57     pipeline = Pipeline.create(pipeline_options)
58     pipeline.read(argv[0]).map(wordcount).write(argv[1])
59     pipeline.run()
```



```
- name: onutraffic
description: "onu traffic"
columns:
  - name: speedout
    description: "the speed"
  tests:
    - unique
    - not_null
(dbt) truong@aaltosea22:~/myprojects
```

```
-- Use the 'ref' function to select
select max(speedout)
from {{ ref('onutraffic') }}
```

# Difficulties

- **Nice UI, simple drag and drop, and easy composition**
  - difficult to solve complex processing and quality control requirements
- **Programming solutions**
  - powerful, but low-level programming, mostly difficult for users who do not know programming languages/technologies well
- **Handling streaming data is still hard**
- **How could low-code and GenAI/LLMs techniques help to generate powerful transformation code?**
  - lowering cognitive load for users and empowering generalists (who can work on code, data and ML models)

# GenAI/LLMs as a solution?

- **Can GenAI/LLMs reduce programming tasks?**
- **Can GenAI/LLMs help for speeding up designs of big data platforms?**
- **Can GenAI/LLMs do the analytics of data by invoking suitable algorithms?**
- **etc.**

# Write the question seeking for a solution, but how much does the answer help?

pls. tell me how can I determine which type of data and which algorithms that can be used to detect anomalies of electricity grid usages?

## Aalto AI Assistant

Detecting anomalies in electricity grid usage involves monitoring the electricity consumption data to identify unusual patterns that might indicate problems such as equipment failures, unauthorized usage, or other irregularities. Here's how you can approach this task:

### Types of Data

The data you need typically includes:

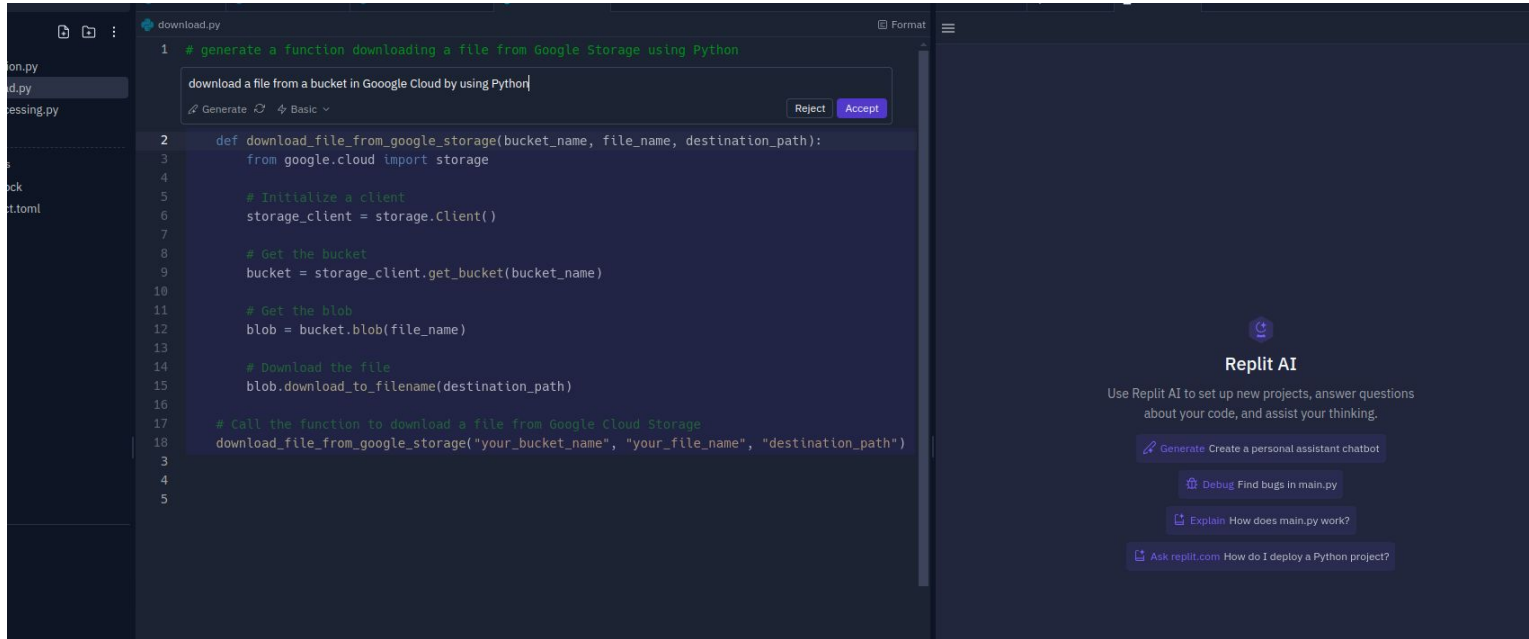
- **Timestamp:** The time at which the measurement was taken.
- **Usage Data:** The amount of electricity consumed during the timestamp (e.g., kilowatt-hours).
- **Voltage and Current:** Measurements of voltage and current, which can help identify issues related to power quality.
- **Frequency Data:** The frequency of the grid, deviations from normal frequency could be an indicator of anomalies.
- **Device/Infrastructure Data:** Information about the devices or infrastructure being monitored, such as sensor IDs, geographical location, and device health metrics.
- **Weather Data:** Optional external data such as weather conditions, which can be correlated with usage patterns.

### Algorithms for Anomaly Detection

Several algorithms can be used for anomaly detection in electricity usage. The choice of algorithm depends on the nature of the data and the specific types of anomalies you want to detect.

**Source:** GPT4o integrated with Aalto AI Assistant, 19.03.2025

# GenAI/LLMs for generating code



The screenshot displays a code editor with a Python file named `download.py`. The code is as follows:

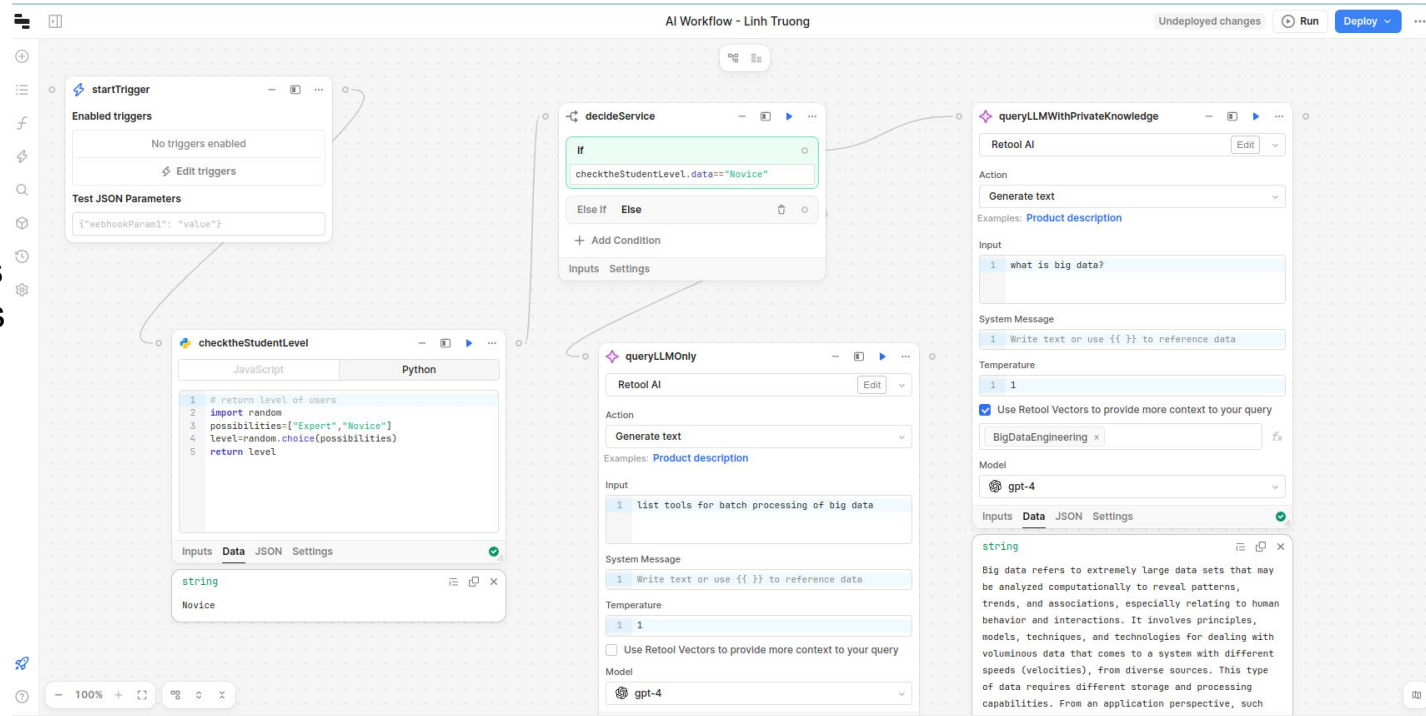
```
1 # generate a function downloading a file from Google Storage using Python
2
3 download a file from a bucket in Google Cloud by using Python
4
5 def download_file_from_google_storage(bucket_name, file_name, destination_path):
6     from google.cloud import storage
7
8     # Initialize a client
9     storage_client = storage.Client()
10
11     # Get the bucket
12     bucket = storage_client.get_bucket(bucket_name)
13
14     # Get the blob
15     blob = bucket.blob(file_name)
16
17     # Download the file
18     blob.download_to_filename(destination_path)
19
20 # Call the function to download a file from Google Cloud Storage
21 download_file_from_google_storage("your_bucket_name", "your_file_name", "destination_path")
22
23
24
25
```

On the right side of the editor, the Replit AI chatbot interface is visible. It includes the text "Use Replit AI to set up new projects, answer questions about your code, and assist your thinking." and several interactive buttons: "Generate Create a personal assistant chatbot", "Debug Find bugs in main.py", "Explain How does main.py work?", and "Ask replit.com How do I deploy a Python project?".

Note: Not necessary for big data platforms only!

# Workflow of conventional tasks and LLMs-based tasks

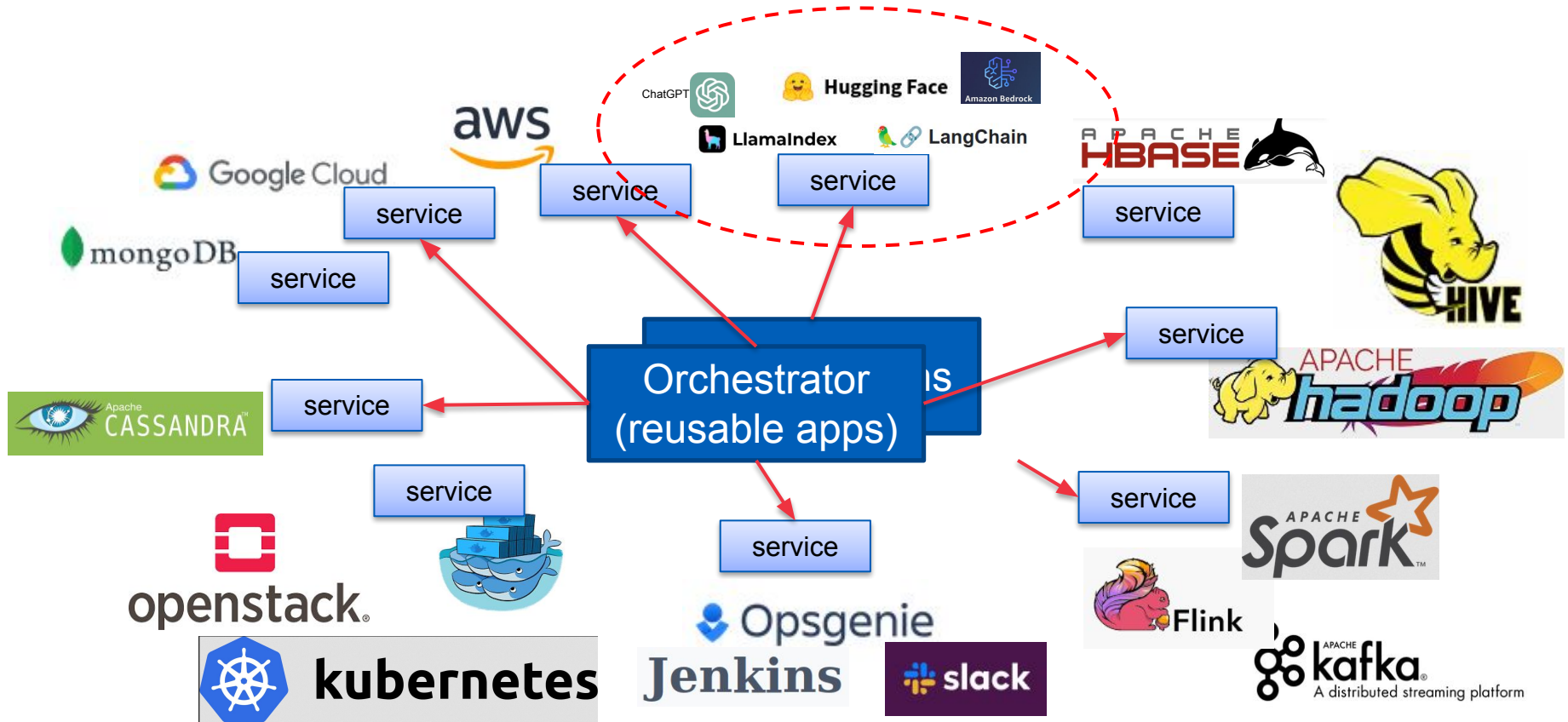
Examples with retool,  
some knowledge is  
trained based on our  
basic big data materials  
(<https://github.com/linhsolar/basicbigdata>)





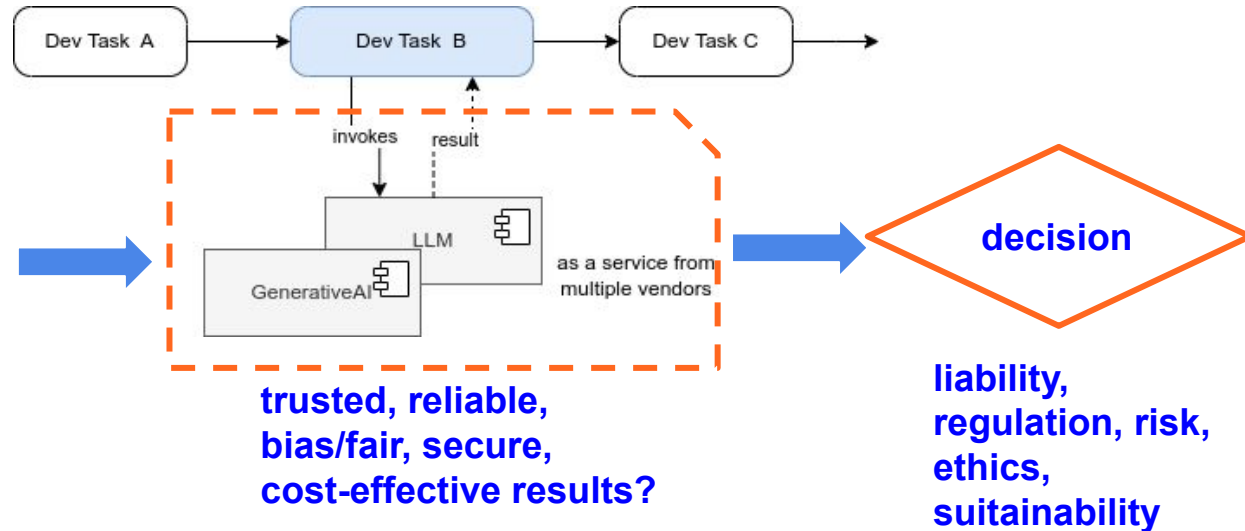
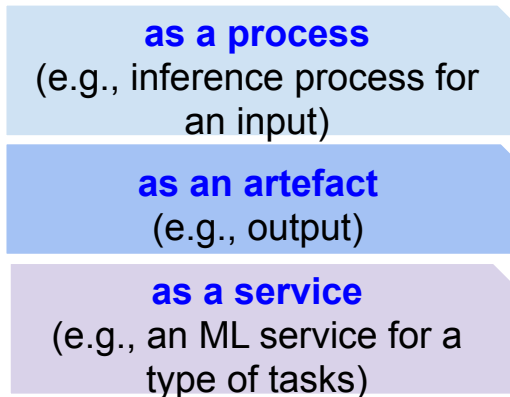
If we want to include  
GenAI/LLMs as service in a  
big data platform

# Recall: service orchestration in data platforms



# Understand the expected quality of results from GenAI/LLMs

## Aspects/Views



# Understand the need to coordinate LLMs and services for development tasks

- **Targets**
  - focused tasks and customized environments for the developer in big data
- **Encapsulating knowledge about the big data platform and application domains**
  - knowledge not easily found in the public space/open sources
  - a kind of “internal” data services, not available in LLMs

# Understand the need to coordinate LLMs and services for development tasks

- **Composition of LLMs with other development tools/knowledge sources**
  - integrated with conventional highly accurate and relevant services
  - cloud pay-per-use and on-prem LLMs services
- **Human-in-loop as workflow tasks**
  - for uncertainty handling and quality feedback
- **Adaptive coordination workflows**
  - based on quality associated with individual LLM capabilities

# Employing LLMs: service selection and integration problems

- **Big LLMs**
  - mostly from well-known providers
  - which tasks can we use as we may not know how the GenAI/LLMs services deal with our own data
- **Small LLMs**
  - can be hosted within a platform, dedicated for individual tenants
- **Diversity of LLMs**
  - functionality: generating analytics reports, code, sample of data
- **Engineering challenges**
  - scalability, data privacy/governance, and multitenant models

# Tools

- **Code generation**

- Many cloud services (with close and open source LLM models)
- ChatGPT, Azure Mistral, together.ai

- **Text-to-SQL for analytics**

- Vanna - <https://vanna.ai/docs/>
- [https://github.com/HKUSTDial/NL2SQL\\_Handbook](https://github.com/HKUSTDial/NL2SQL_Handbook)

- **LLMs for data management**

- <https://github.com/weAIDB/aw-some-data-llm>

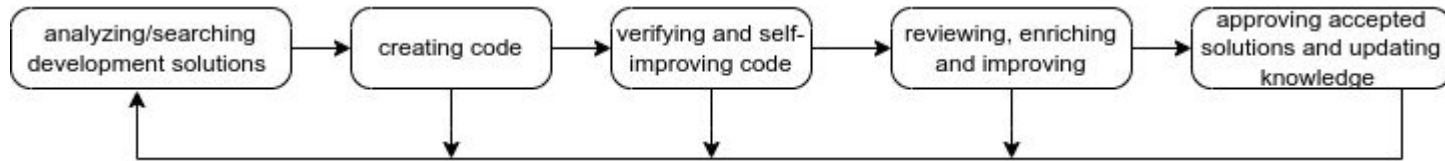
Table 4: Pass@1 performance of pretrained code models (top), instruction finetuned code models (middle), in comparison with some of the best general language models (bottom), with models in each category ordered chronologically. The sources of these figures can be found in Section 5.3, Section 5.4, and Table 1.

Model	Size	HumanEval	MBPP
PolyCoder	2.7B	5.6	-
CodeGen-Mono	16.1B	29.3	35.3
InCoder	6.7B	15.2	19.4
PyCodeGPT	110M	8.3	-
Pangu-Coder	2.6B	23.8	23.0
SantaCoder	1.1B	14.0	35.0
CodeGeeX	13B	22.9	24.4
StarCoder	15.5B	33.6	52.7
CodeT5+	16B	30.9	-
Phi-1	1.3B	50.6	55.5
CodeFuse	13B	24.8	-
DeepSeek Coder	33B	56.1	66.0
InstructCodeT5+	16B	35.0	-
WizardCoder	15.5B	57.3	51.8
Pangu-Coder 2	15.5B	61.6	-
OctoCoder	15.5B	46.2	-
CodeFuse	34B	74.4	-
DeepSeek Coder-Instruct	33B	79.3	70.0
GPT-4	-	67.0/82	-
PaLM 2*	S	37.6	50.0
Code LLaMA	34B	53.7	56.2
Phi-1.5	1.3B	41.4	43.5

**Source:** Zhang et al., “Unifying the Perspectives of NLP and Software Engineering: A Survey on Language Models for Code”,  
<https://arxiv.org/abs/2311.07989>

# Collms example: for edge-cloud software modernization & new development

- **AI/ML/LLMs as services in development workflows**

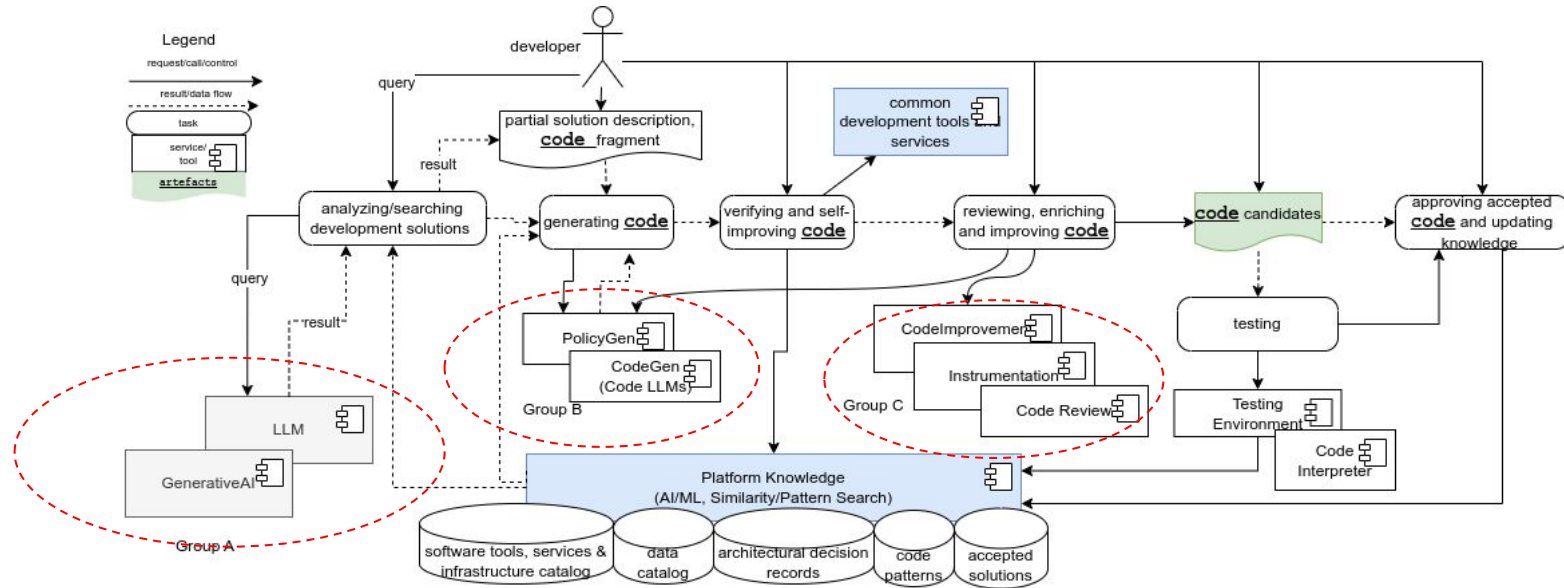


- **Cloud, on-prem and personal LLMs services**
  - composition of LLMs with other development tools/knowledge sources
- **Adaptive coordination workflows based on quality associated with concrete aspects/views of individual LLM capabilities**
  - explainability; robustness, reliability, resilience & elasticity

H. -L. Truong, M. Vukovic and R. Pavuluri, "On Coordinating LLMs and Platform Knowledge for Software Modernization and New Developments," *2024 IEEE International Conference on Software Services Engineering (SSE)* doi: 10.1109/SSE62657.2024.00036.

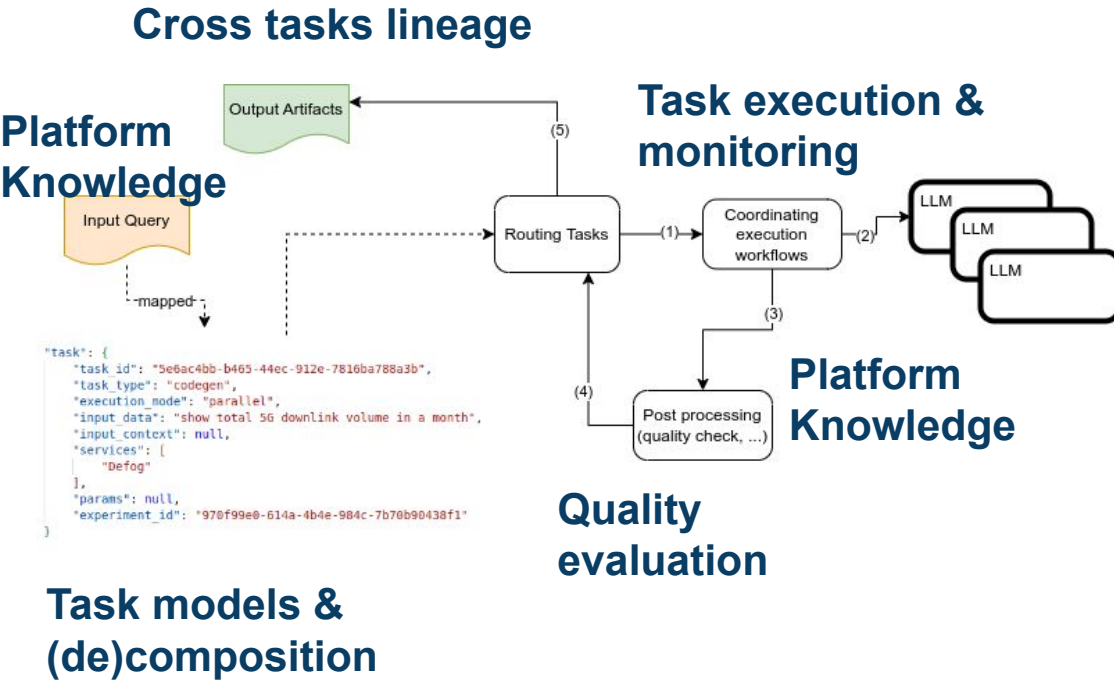


# Collms: a high level view



**Service group:** for specific type of tasks, at runtime an ensemble from the group will be used

# Collms: a high level view

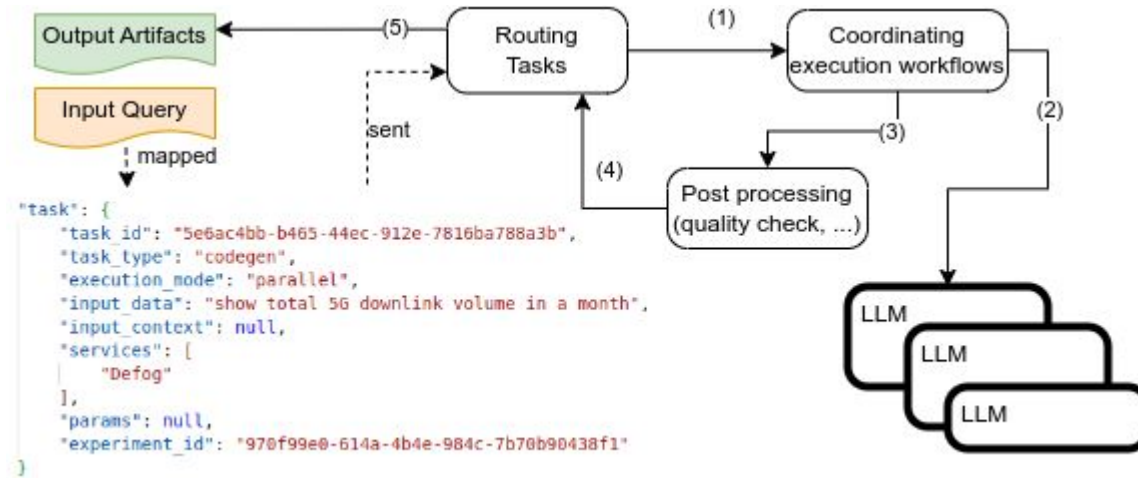


## Trails of task execution & quality

```
{
  "task_instance_id": "2f25cd24-e5dc-4437-8949-95788371c353",
  "start_ts": 1698430168.4745324,
  "end_ts": 1698430184.2641177,
  "service_name": "Defog",
  "service_qoa": {
    "total_token": null,
    "responsetime": 15.789584357000422,
    "cost": null
  },
  "task": {
    "task_id": "5e6ac4bb-b465-44ec-912e-7816ba788a3b",
    "task_type": "codegen",
    "execution_mode": "parallel",
    "input_data": "show total 5G downlink volume in a month",
    "input_context": null,
    "services": [
      "Defog"
    ],
    "params": null,
    "experiment_id": "970f99e0-614a-4b4e-984c-7b70b90438f1"
  }
}
```

# Colms: configuring and run-time quality-aware scheduling of tasks execution

Patterns in coordination: at the ensemble level (composition/group coordination) and at the individual LLM level (RAG and LLM parameters)



## LLM service management complexity

- controllable parameter spaces for coordination
- customized invocation pipelines of individual LLMs with in a group
- scalable service communication handling (REST and AMQP)

# Collms: Observability

- **End-to-end tracing and logging of cross LLMs and platforms**
  - allowing for detail explainability
- **Linkages/trails between accepted solutions/results and the goal of the developments**
  - types of tasks, types of target solutions, and development contexts

---

```
"task_instance_id": "354c6c20-66df-4390-...",  
"start_ts": 1698768167.5257006,  
"end_ts": 1698768167.8629,  
"service_name": "InCoder",  
"service_goa": {  
  "responsetime": 0.3371994800399989  
},  
"task": {  
  "task_id": "30fbfed7-bd9f-4760-9fcb-...",  
  "task_type": "codegen",  
  "execution_mode": "parallel",  
  "input_data": "def read_data_as_df ...",  
  "services": ["StarCoder", "CodeLlama", "  
    InCoder"],  
  "experiment_id": "473fc800-6c60-449b-..."  
}
```

---

# Collms: few snapshots of a prototype

## Coordinating LLMs and Platform Knowledge for Application Modernization

### Showing some thoughts and concepts

Platform Knowledge Search Solution Search/Refinement **Code Generation** Testing and Verification Enrichment/Improvement Platform Knowledge Update

### Code Generation with Code LMs

Services

StarCoder x CodeLlama x InCoder x

Select executor ensembles

☒ parallel Parallel service invocations ☐ quorum A majority of services ☐ n\_random N of random services ☐ first best The first best service ☐ top\_k The top k services

Input:

```
def read_data_as_df(self, table_name, size, fields):
    if (self.conn==None):
        return None
    join_fields=", ".join(fields)
    join_fields=", ".join(fields)
```

Generate

### Result from StarCoder

```
def read_data_as_df(self, table_name, size, fields):
    if (self.conn==None):
        return None
    join_fields=", ".join(fields)
    if (size > 0):
        sql=f'SELECT {join_fields} from {table_name} LIMIT {size}'
    else:
        sql=f'SELECT {join_fields} from {table_name}'
    print(f'Running {sql}')
    try:
        with self.conn.cursor() as cur:
            cur.execute(sql)
            rows=cur.fetchall()
            df=pd.DataFrame
```

### Result from CodeLlama

```
def read_data_as_df(self, table_name, size, fields):
    if (self.conn==None):
        return None
    join_fields=", ".join(fields)
    if (size > 0):
        sql=f'SELECT {join_fields} from {table_name} LIMIT {size}'
    else:
        sql=f'SELECT {join_fields} from {table_name}'
    print(f'Running {sql}')
    try:
        with self.conn.cursor() as cur:
            cur.execute(sql)
            rows=cur.fetchall()
            columns
```

### Result from InCoder

Model facebook/incode-6B is currently loading

Platform Knowledge Search Solution Search/Refinement Code Generation **Testing and Verification** Enrichment/Improvement Platform Knowledge Update

### Code, Policy and Configuration Testing and Verification with Virtual Testing Environments

Services

CodePolicyValidation x

Type:

OPA

Input:

```
Violation[input_client_info, "ResponseTime violation on small resource machine"]
input_service_info.machine_ipes == "small"
input_service_info.metric_1 == "ResponseTime"
some.i.j
```

Check

### Result from CodePolicyValidation

```
{
  "errors": [
    {
      "message": "unexpected eof token expected '\\n or ; or }'",
      "code": "rego_parse_error",
      "location": {
        "file": "/tmp/tmp4kclagpg/policy_file",
        "row": 34,

```

# Colms: few snapshots of a prototype

## Coordinating LLMs and Platform Knowledge for Application Modernization

### Showing some thoughts and concepts

Platform Knowledge Search Solution Search/Refinement Code Generation Testing and Verification Enrichment/Improvement **Platform Knowledge Update**

#### Platform Knowledge Update

Case info:

expect table aggregation to equal other table

Query (for knowledge update):

Except an (optionally grouped) expression to match the same (or optionally other) expression in a different table.

Answer (for adding accepted solutions):

```
group_by: [date_column]
compare_group_by: [some_other_date_column]
row_condition: some_flag=true
compare_row_condition: some_flag=false
```

Reference doc(for adding accepted solutions):

[https://github.com/calogica/dbt-expectations/tree/0.8.2/#expect\\_column\\_to\\_exist](https://github.com/calogica/dbt-expectations/tree/0.8.2/#expect_column_to_exist)

Add Accepted Solution

## Coordinating LLMs and Platform Knowledge for Application Modernization

### Showing some thoughts and concepts

**Platform Knowledge Search** Solution Search/Refinement Code Generation Testing and Verification Enrichment/Improvement Platform Knowledge Update

#### Platform Knowledge

Input (for query/description):

find dbt expectation table aggregation

Search Software/Platform Catalog

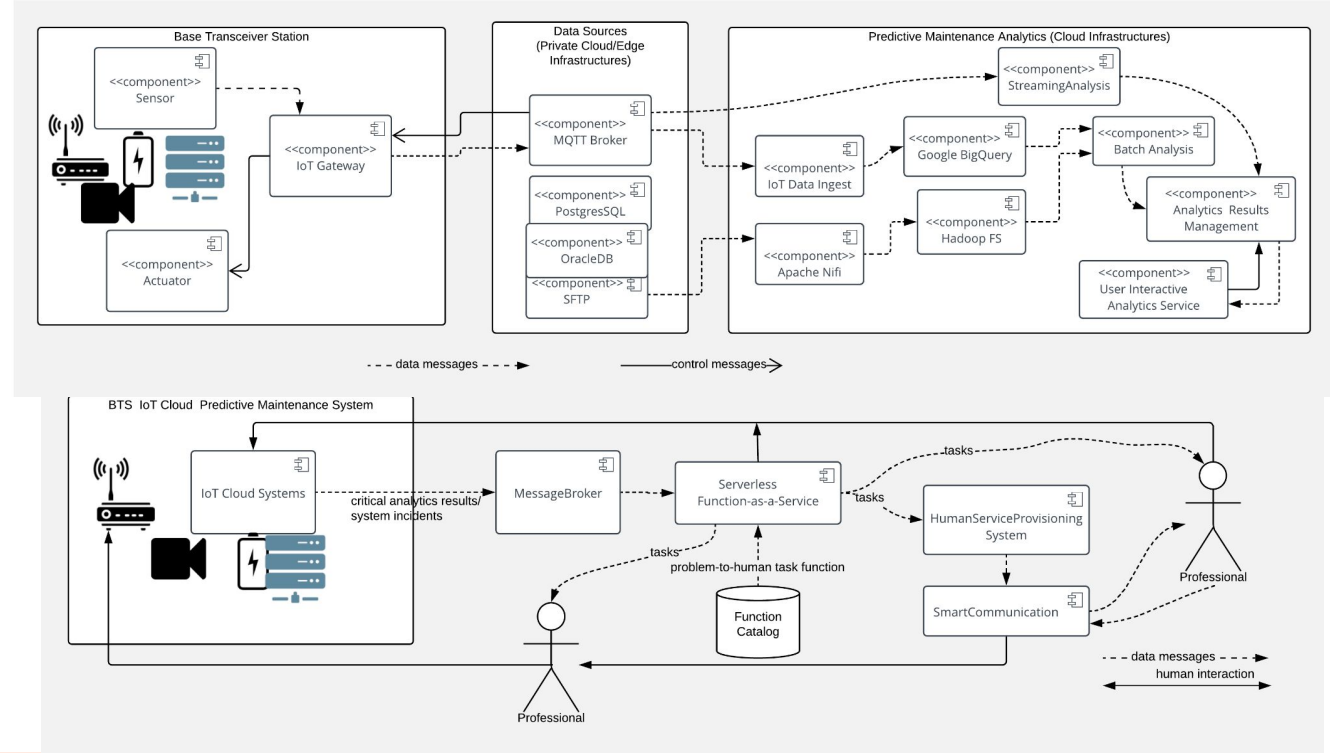
Search Solution

tests:

```
- dbt_expectations.expect_table_aggregation_to_equal_other_table:
  expression: max(column_a)
  compare_model: ref("other_model")
  compare_expression: max(column_b)
  group_by: [date_column]
  compare_group_by: [some_other_date_column]
  row_condition: some_flag=true
  compare_row_condition: some_flag=false
```

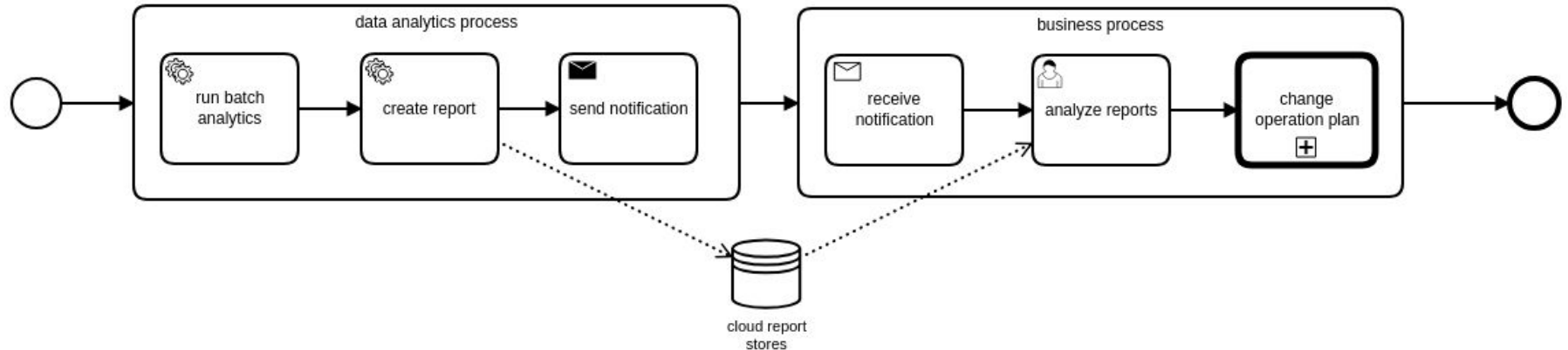
# Where could workflows of LLMs calls and AI Agents help? (1)

**Complex platform tasks for predictive maintenance or security orchestration**



# Where could workflows of LLMs calls and AI Agents help? (2)

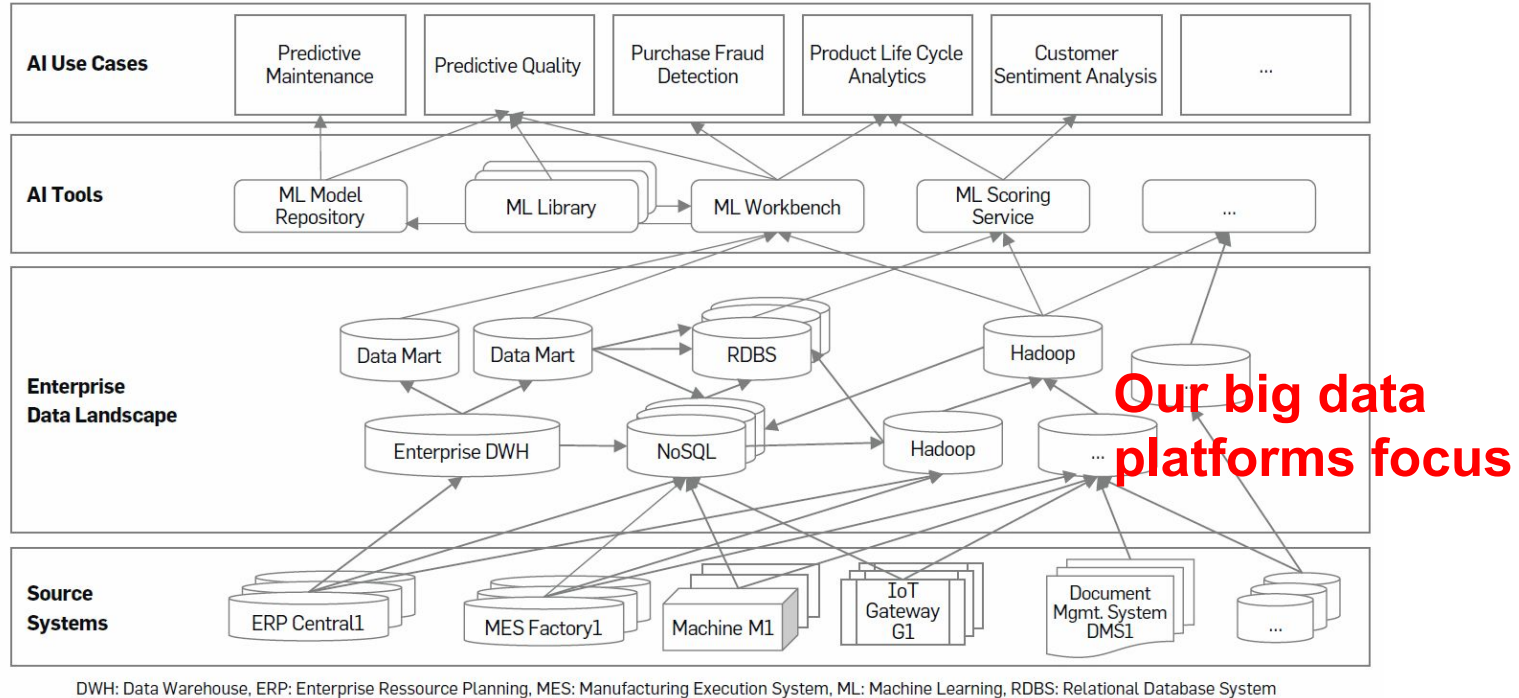
- **data analytics** processes and **business** processes
  - include human-in-the-loop





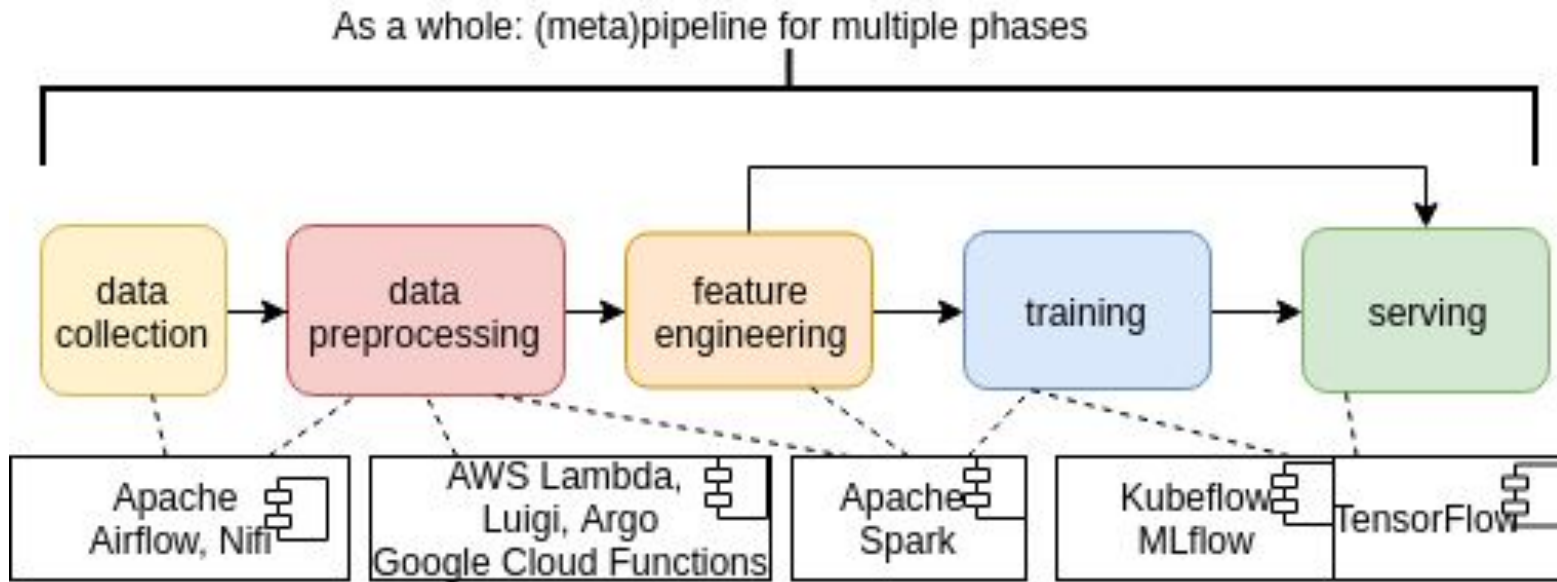
# Using techniques/methods studied in Big Data Platforms for GenAI/LLMs

# “No AI Without Data”: before LLMs



**Figure source:** There Is No AI Without Data, by Christoph Gröger, Communications of the ACM, November 2021, Vol. 64 No. 11, Pages 98-108, 0.1145/3448247

# Big data and ML pipelines



# Big data platforms and ML

## Example: Michelangelo ML from Uber

It is easy to see the role of big data components we study in this figure

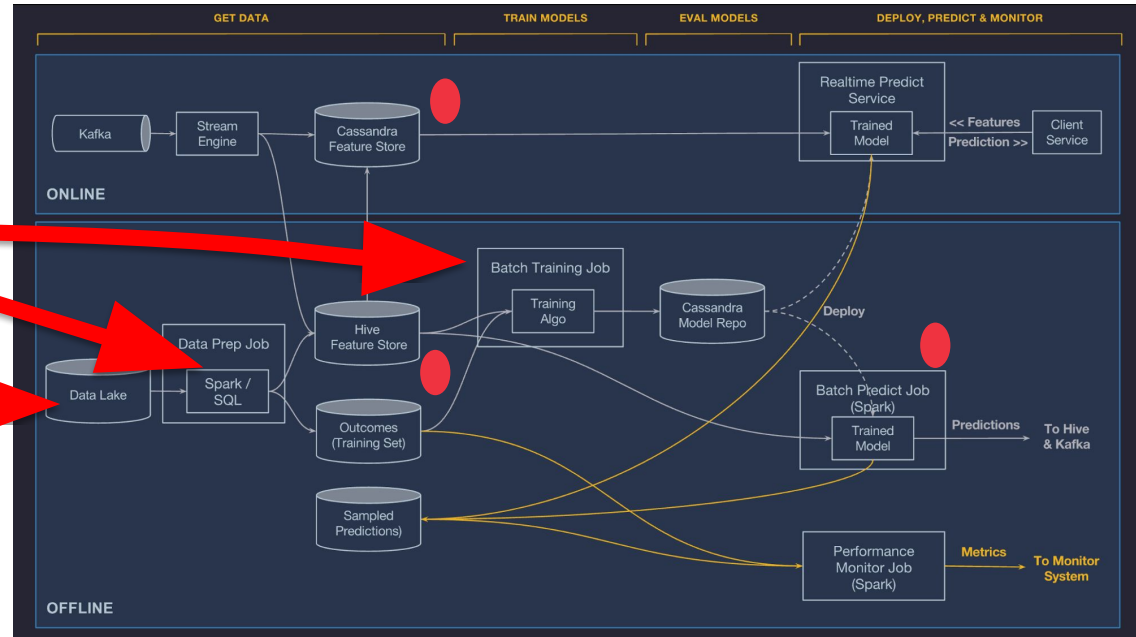


Figure source: <https://eng.uber.com/michelangelo-machine-learning-platform/>

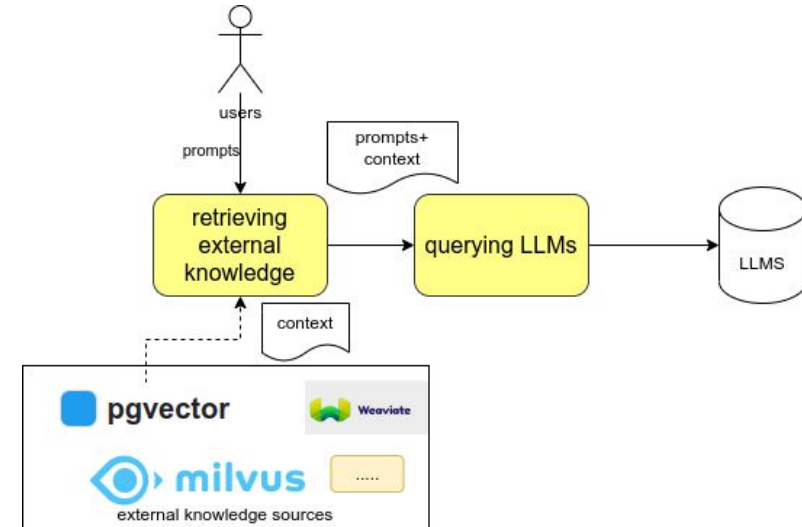
# Techniques and methods in big data platforms help to build GenAI/LLMs

- **GenAI/LLM as a key part that big data platforms must support**
  - analytics workflows and service integration
- **Data management for GenAI/LLMs**
  - enable sharing, retrieving and managing big data
- **Techniques and methods for GenAI/LLMs development**
  - connectors, transformation, data pipelines, data governance

# Support LLMs: the big data pipeline for RAG in GenAI/LLMs

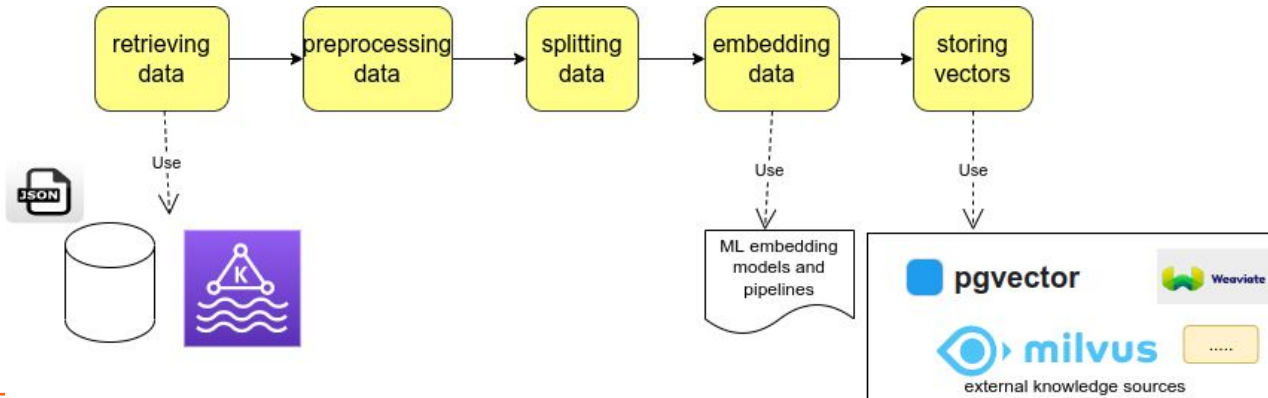
- **RAG: Retrieval-Augmented Generation**
  - getting facts/relevant accurate knowledge from an external source for LLMs
  - providing context for improving LLM inferences
- **Key components**
  - pipelines and databases

## RAG for LLMs

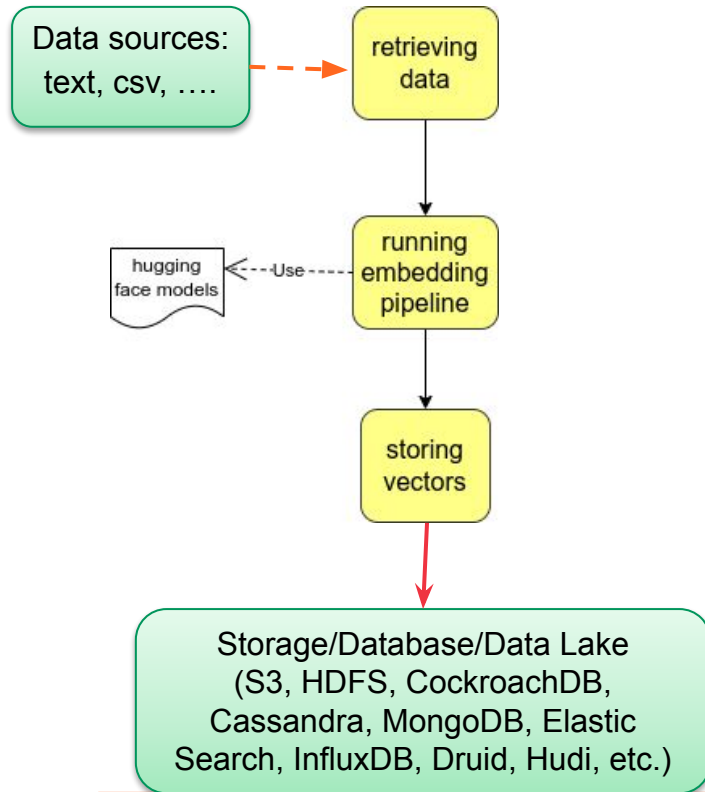


# Support LLMs: the big data pipeline for RAG in GenAI/LLMs

- The process to build RAG is a “big data” pipeline
  - getting data and determining vectors as indexes for data
  - storing data and indexes into databases
  - using common vector databases → **data services**
    - *Postgres with pgvector, ChromaDB, Milvus, Weaviate, ...*



# An example with LlamaIndex



```
from llama_index.core import VectorStoreIndex, SimpleDirectoryReader, Settings
from llama_index.core.embeddings import resolve_embed_model

#assume the data is under "data" (e.g., csv and text file)
documents = SimpleDirectoryReader("data").load_data()

#specify embedding model for extracting vectors
Settings.embed_model = resolve_embed_model("local:BAAI/bge-small-en-v1.5")

#configure LLM, set None as it illustrates the RAG
Settings.llm = None
# specify the embedding pipeline/task
index = VectorStoreIndex.from_documents(
    documents,
)

#Now we can query Vector to get a context
retriever = index.as_retriever()
query=input("Enter your query:")
contexts = retriever.retrieve(query.strip())
# show context, assume that it context=NodeWithScore with node=TextNode
for context in contexts:
    print(context.node.text)
# also query LLM (but none in this code)
query_engine = index.as_query_engine()
result = query_engine.query(query.strip())
print(result)
```



# Summary

- **Complex design and engineering issues in big data platforms**
  - many big systems, very complex designs and requirements
  - require us to continuously upskill
- **New GenAI/LLMs can help simplifying certain tasks**
  - GenAI/LLMs and AI Agents are services and paradigms that can be considered as **fundamental elements for a programming model**
- **Your expertises from Big Data Platforms can help**
  - integrate GenAI/LLM services
  - build GenAI/LLMs capabilities for big data problems

# Thanks!

**Hong-Linh Truong**  
**Department of Computer Science**

**rdsea.github.io**