AiDD AI+ 软件研发数字峰会
AI+ software Development Digital summit

AI驱动软件研发
全面进入数字化时代

中国·深圳 11.24-25

AI+
software
Development
Digital
summit

Byzer：以数据方式管理大模型全生命周期

祝海林 Kyligence

科技生态圈峰会 + 深度研习

——1000 + 技术团队的共同选择

KEYLINK ing

K+峰会

K+峰会 深圳站

K+ 全球软件研发行业创新峰会

会议时间：2024.05.24-25

K+峰会 上海站

K+ 全球软件研发行业创新峰会

会议时间：2024.09.20-21

AiDD 峰会

AiDD峰会 深圳站

AI+ 软件研发数字峰会

会议时间：2023.11.24-25

AiDD峰会 北京站

AI+ 软件研发数字峰会

会议时间：2024.07.19-20

AiDD峰会 深圳站

AI+ 软件研发数字峰会

会议时间：2024.11.15-16

演讲嘉宾

祝海林

Byzer 社区 PMC/资深数据架构师/Kyligence 技术合伙人

拥有 15 年研发经验，一直专注于 Data+AI 融合，致力于帮助企业更好的落地 Data+AI。个人热衷于开源产品的设计和研发，主要开源作品：Byzer/MLSQL 。

最新项目：Byzer-LLM 可帮助企业快速落地私有化大模型；Byzer-retrieval 旨在作为 LLM RAG（检索增强生成）检索后端。

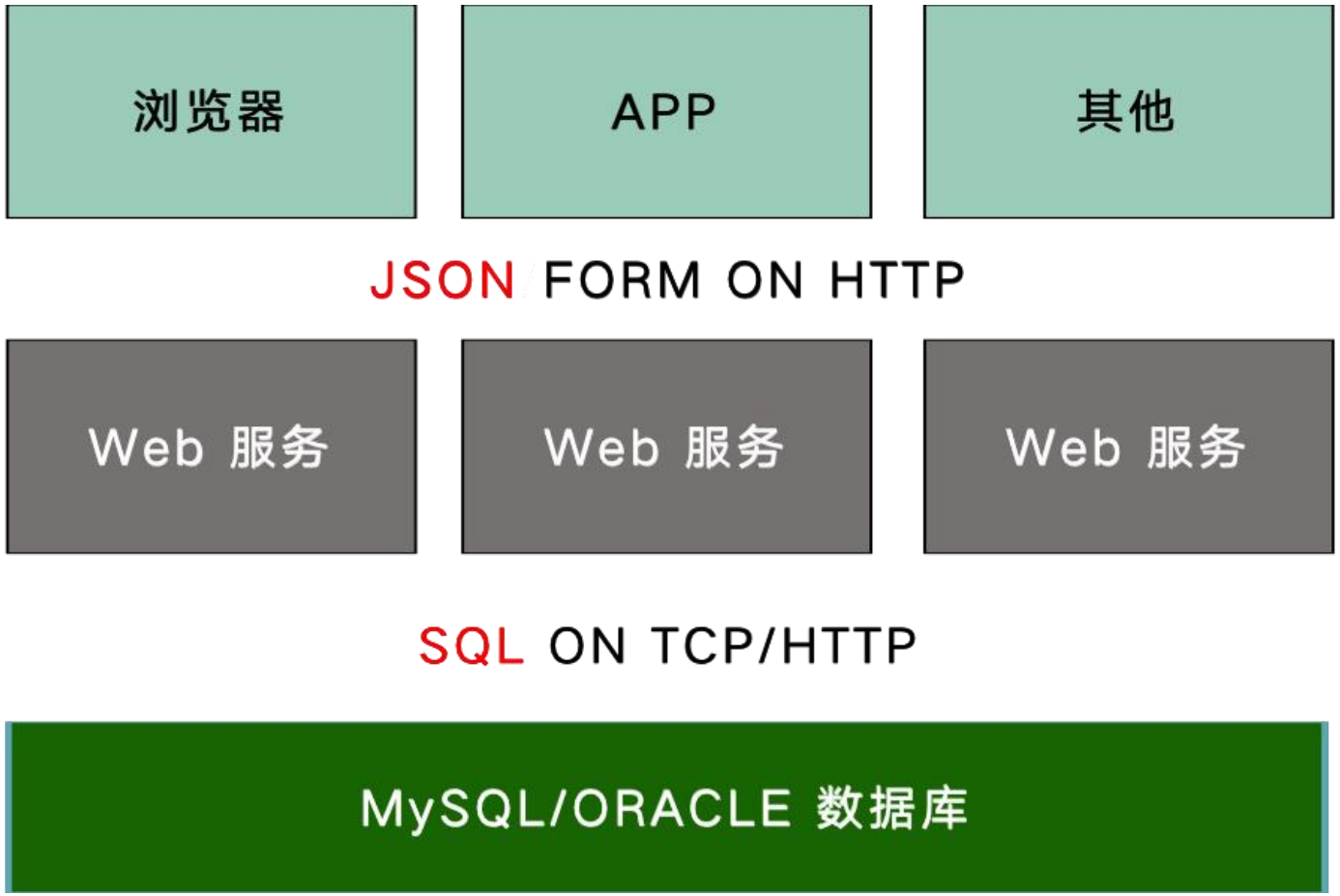
Byzer 2022 年获得中国开源创新大赛二等奖；2023 年获得浦东新区人工智能创新大赛一等奖。个人入选「中国 2022 年开源先锋 33 人」，荣获 2023 年全球人工智能开发者先锋大会「开发者先锋」称号。

目录
CONTENTS

1. Byzer 数据库的开发背景
2. 为什么我们称 Byzer 是AI 数据库
3. 我们是如何实现这个 AI 数据库的
4. 使用 SQL 完成预训练 微调 部署及调用
5. 快速将 Byzer 数据库应用于企业业务中
6. Byzer 数据库现状和未来的发展

PART 01
Byzer 数据库的开发背景

以数据库为中心的传统 Web 开发模式

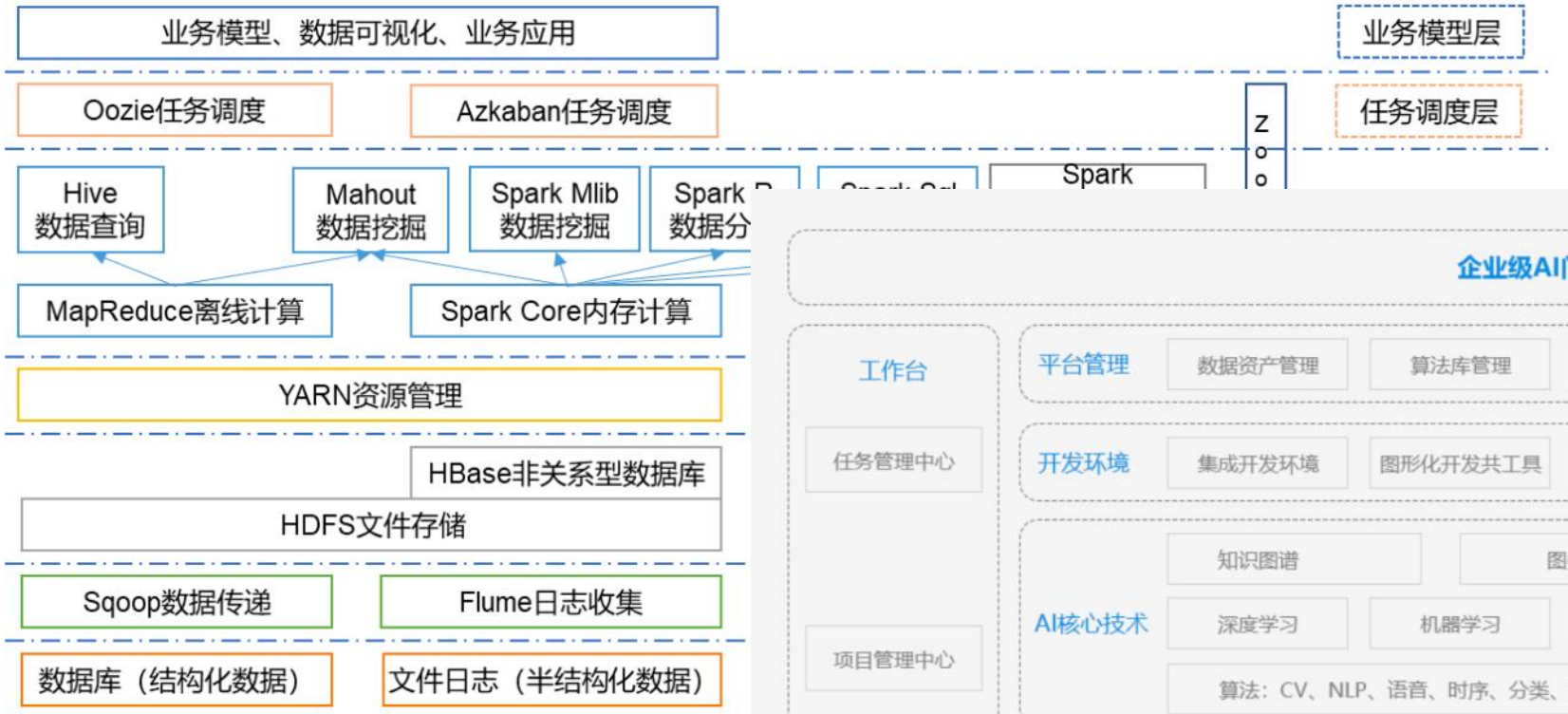


前端/APP/业务产品

后端/业务逻辑

存储/计算/数据

现有 Data/AI 开发模式

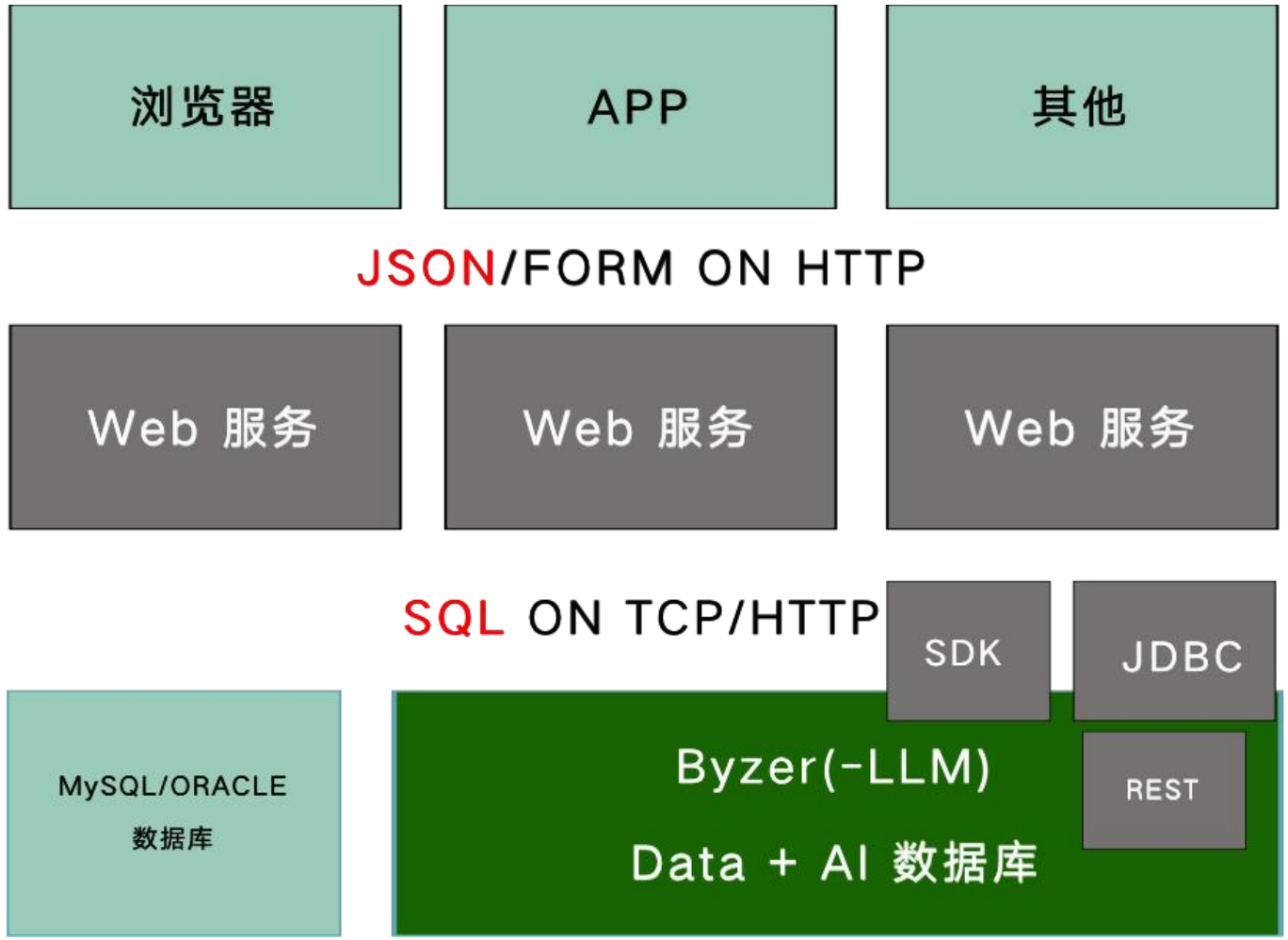


专有组件繁多，学习、开发、维护成本极高，**需要大量专家。**

复杂度太高，团队无法专注业务

Data/AI 平台割裂，难以统一

以 Byzer Data+AI 数据库为中心的开发模式



前端/APP/业务产品

后端/业务逻辑

存储/计算/数据/大模型

Byzer 数据库的价值

依托于久经验证的Data+AI 基础设施

- **复用 Web 开发模式开发 D+AI产品**

企业可以保留原有的开发模式和流程保持不变

- **仅需懂得 Prompt 和 SQL 的人才**

几乎所有开发者都懂得 SQL ,而 Prompt 的学习门槛也非常低

- **复用庞大的开发者生态，专注于业务**

Web开发是当前最庞大的开发者生态，我们可以让他们开发 Data + AI 应用

- **功能强大且覆盖面广**

Byzer AI 数据库增强了 SQL ， 几乎满足 Data + AI/LLM 方方面面的需求

Byzer 数据库的价值

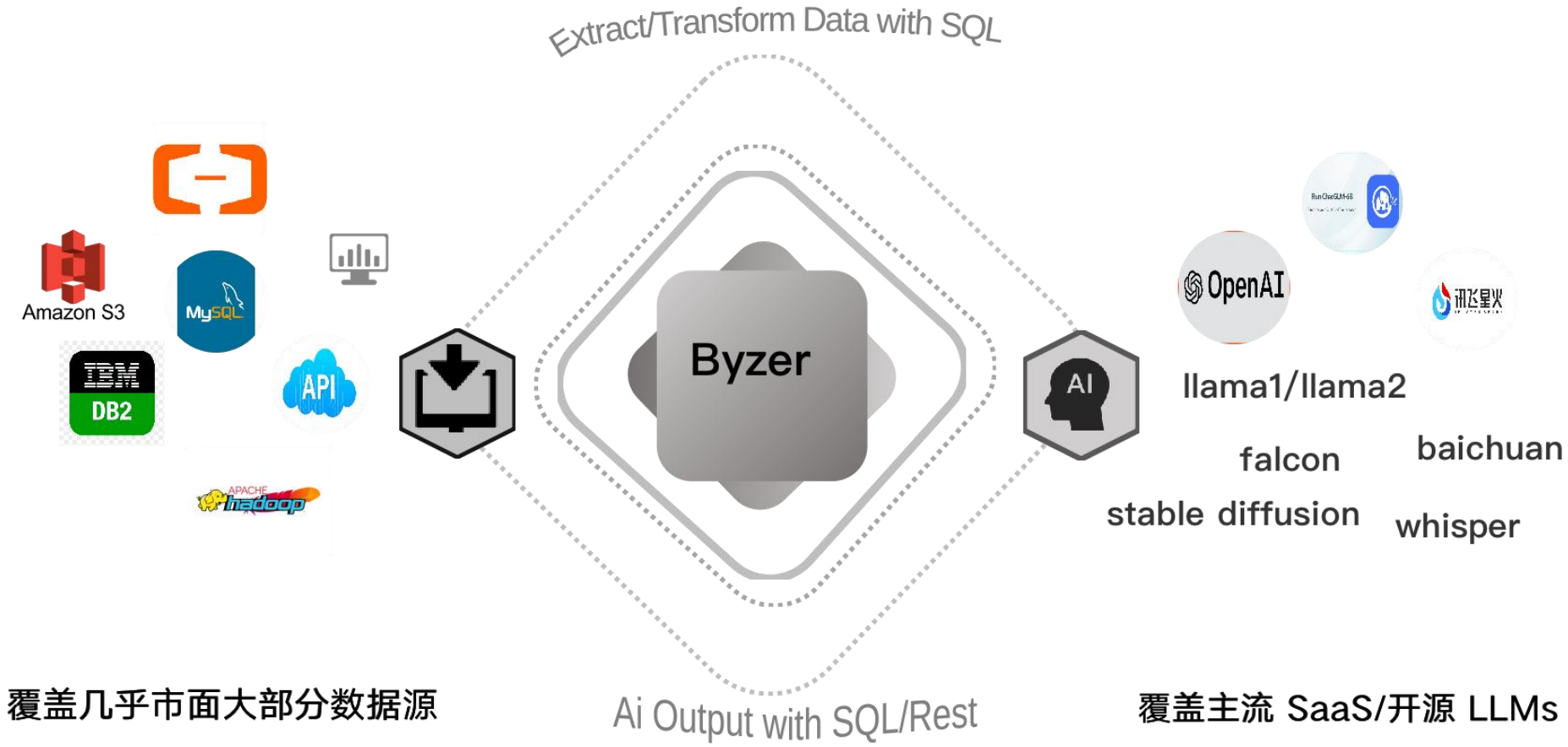
Data + AI(LLM) 的应用开发不再高不可攀

有懂 Web 开发的同学就够了

PART 02

为什么我们称 Byzer 是 AI 数据库

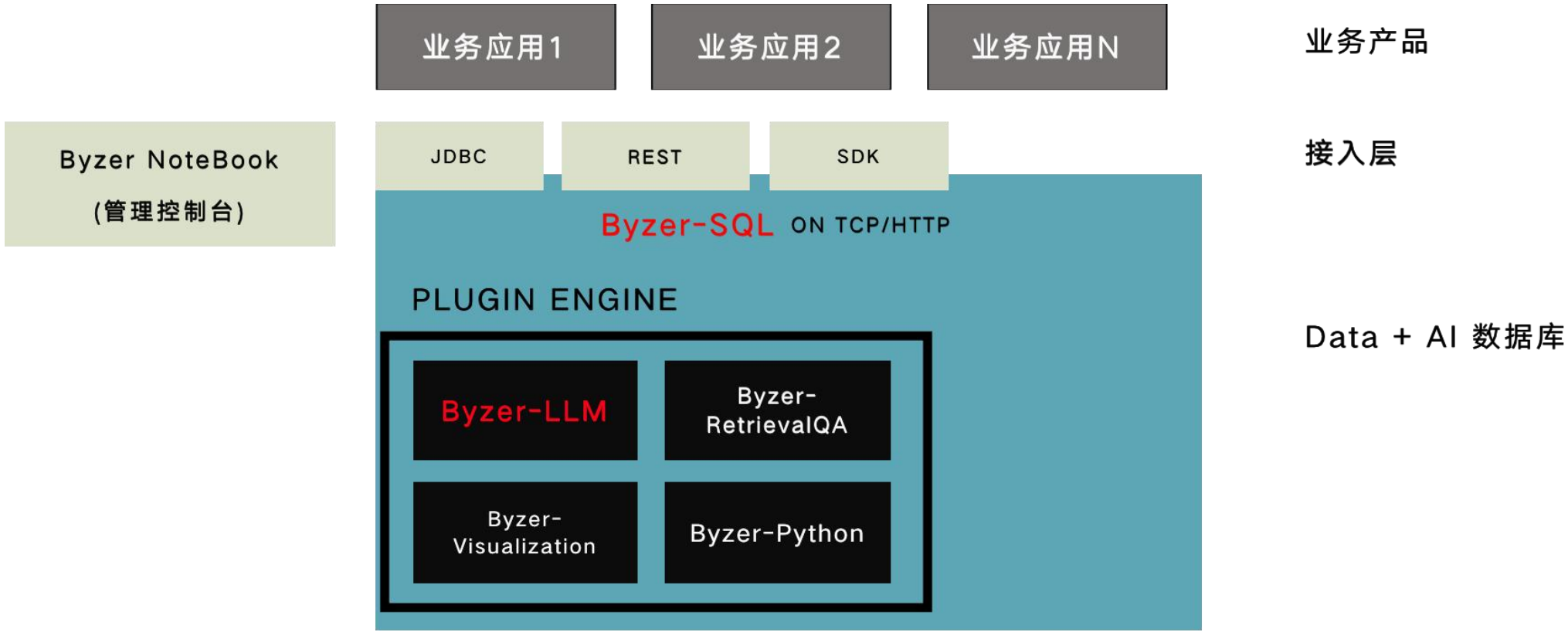
Data+AI 从业者的视角下的Byzer 数据库



覆盖几乎市面大部分数据源

覆盖主流 SaaS/开源 LLMs

传统 Web 开发视角下的 Byzer 数据库

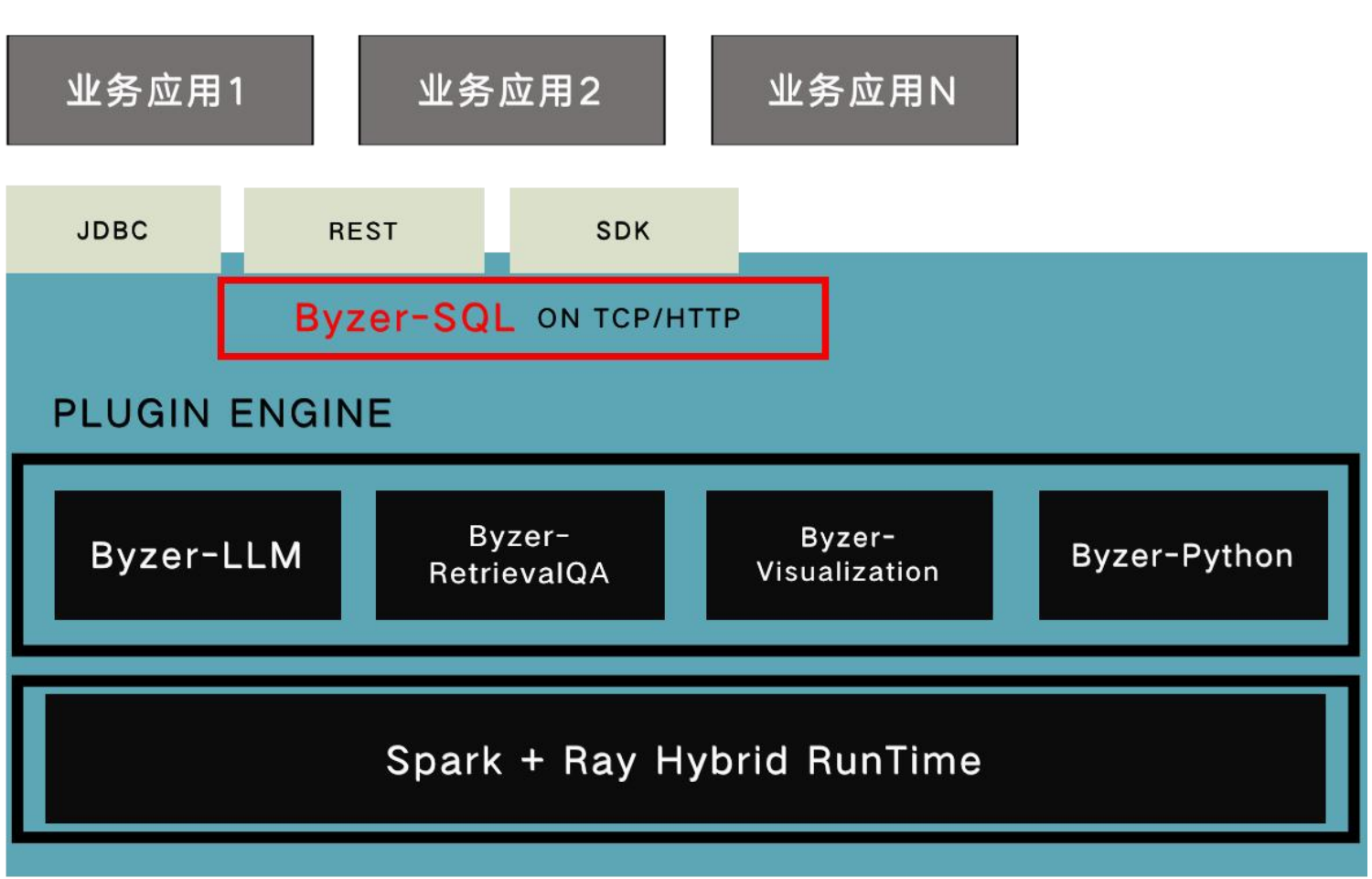


大模型时代下，Data+AI(LLM) 的开发会重回数据库时代

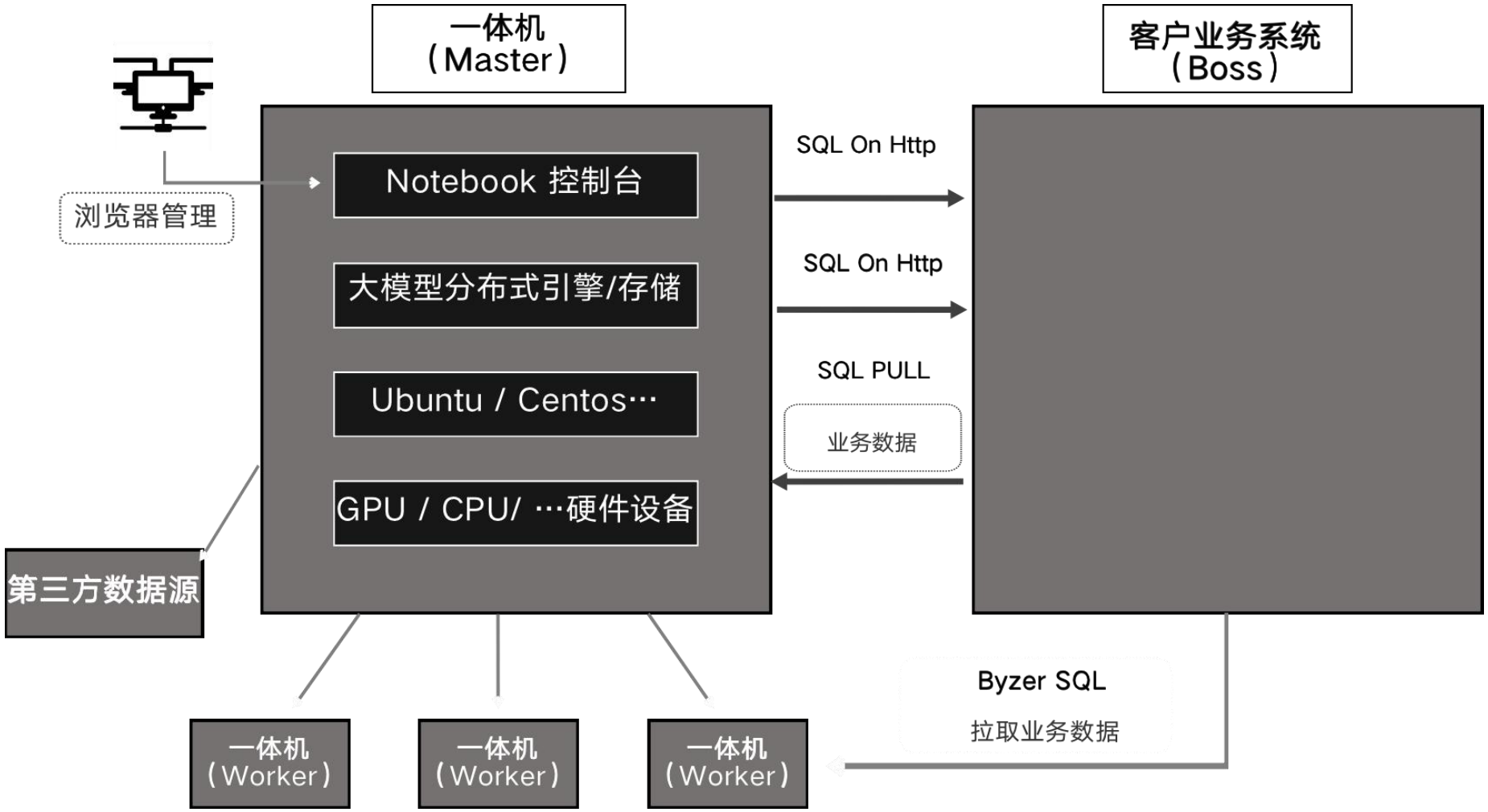
PART 03
我们是如何实现这个 AI 数据库的

我们是如何实现 Byzer 数据库的

1. 多种形态的接入层
2. 重新设计的 SQL 语言
3. 插件式的执行引擎
4. 工业级的运行时



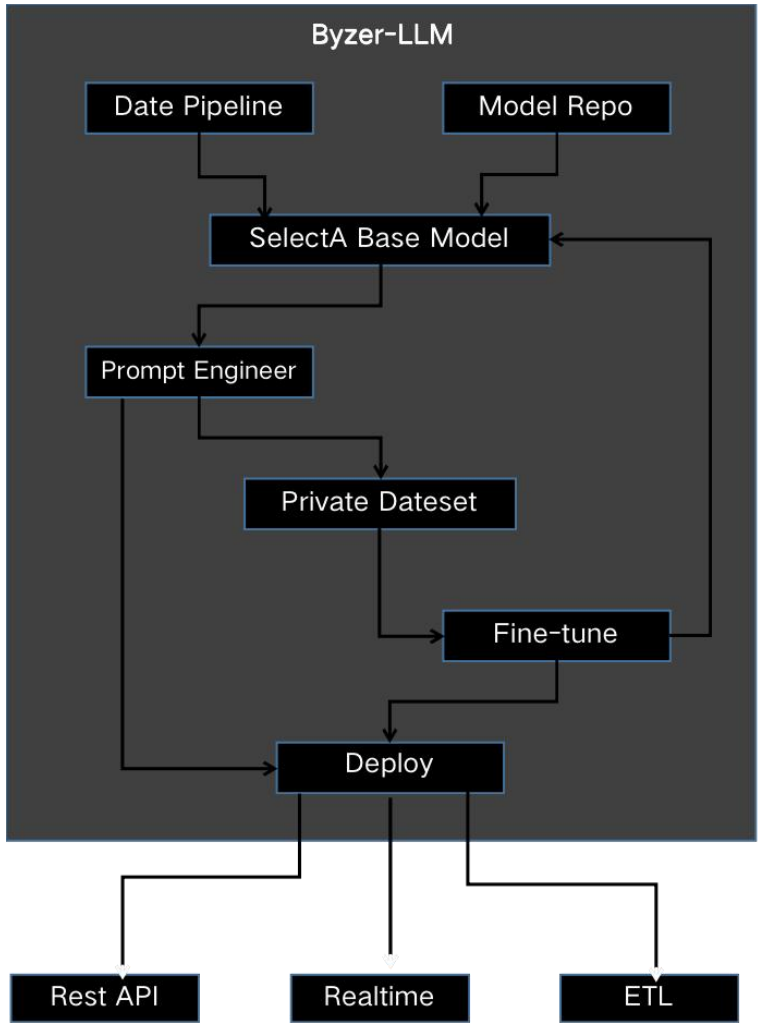
数据库软硬一体 插电可用



PART 04

使用 SQL 完成预训练 微调 部署和调用

大模型全生命周期管理

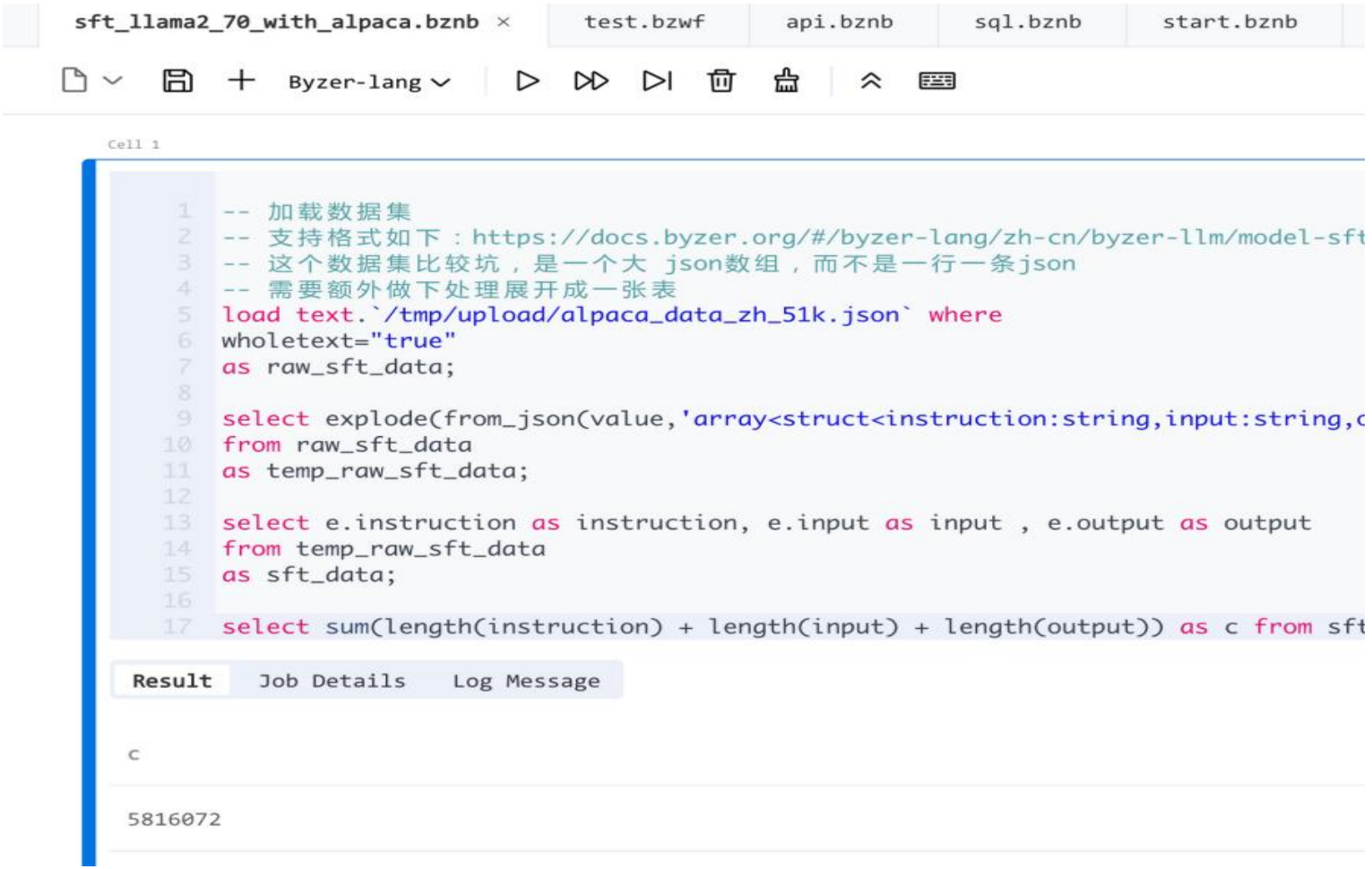


- 使用 SQL 管理大模型预训练，微调和部署
- 支持 SQL 调用大模型
- 提供了大模型的存储设施
- 在 BYZER NOTEBOOK 中可以完成左侧全流程

大模型全生命周期管理

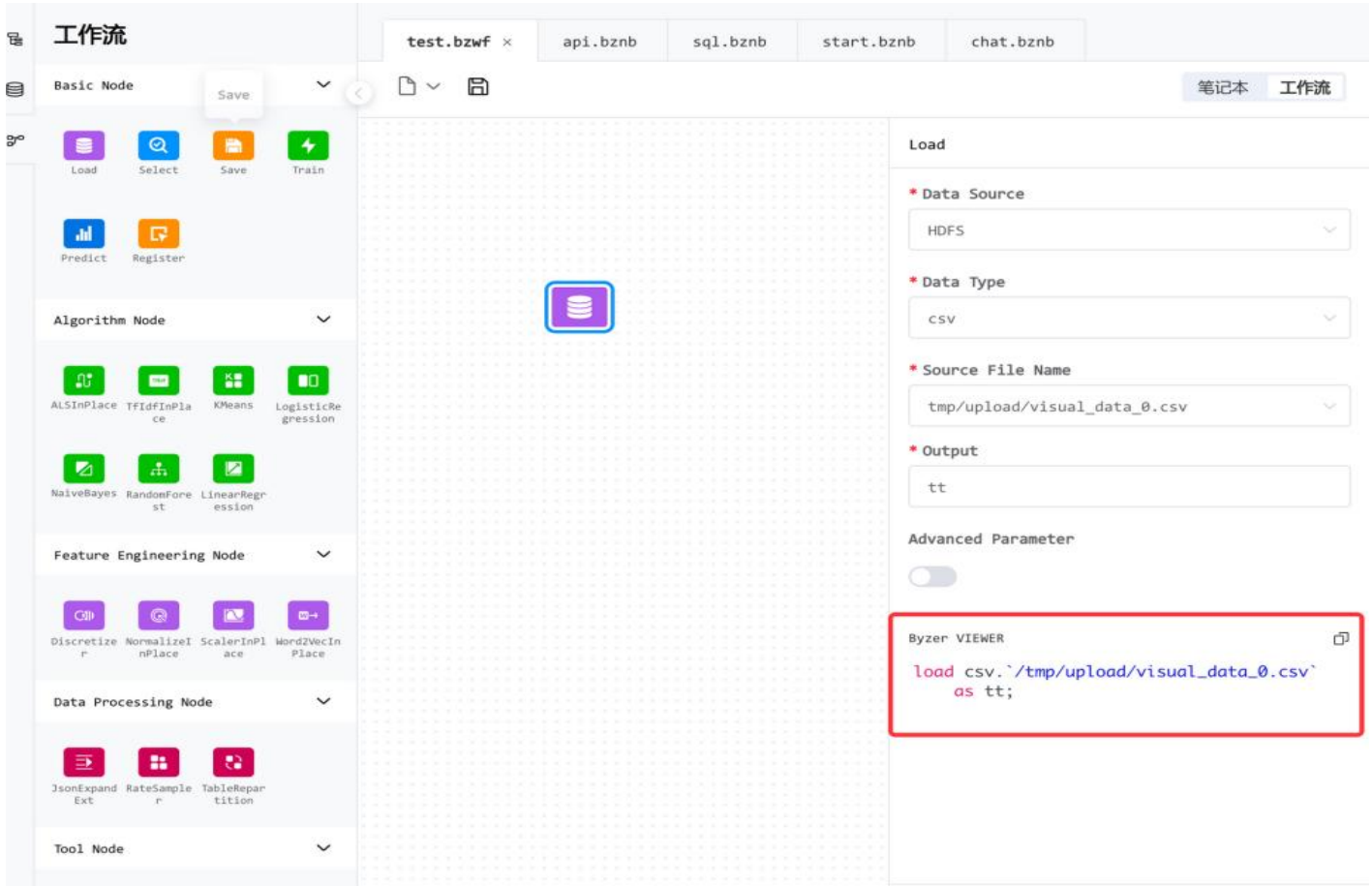
微调数据处理

NoteBook 模式数据处理



专家模式：直接编写 SQL / 可以利用大模型 Copilot / 有强大的代码提示

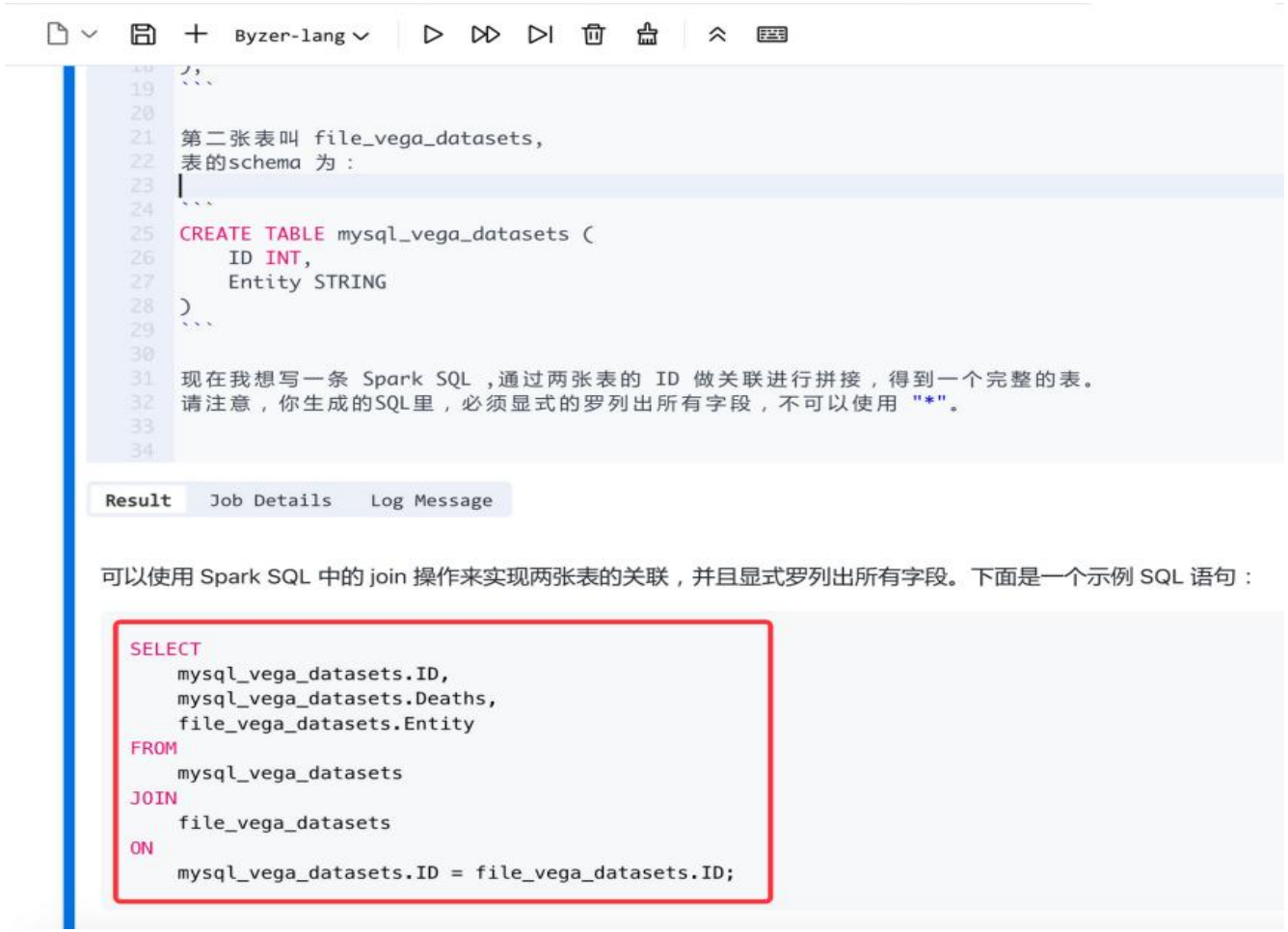
Workflow 模式数据处理



完全可视化交互 / 实时预览自动生成 SQL / 涵盖数据和算法处理算子 / 后续可增加 copilot 算子

数据处理的 AI Copilot

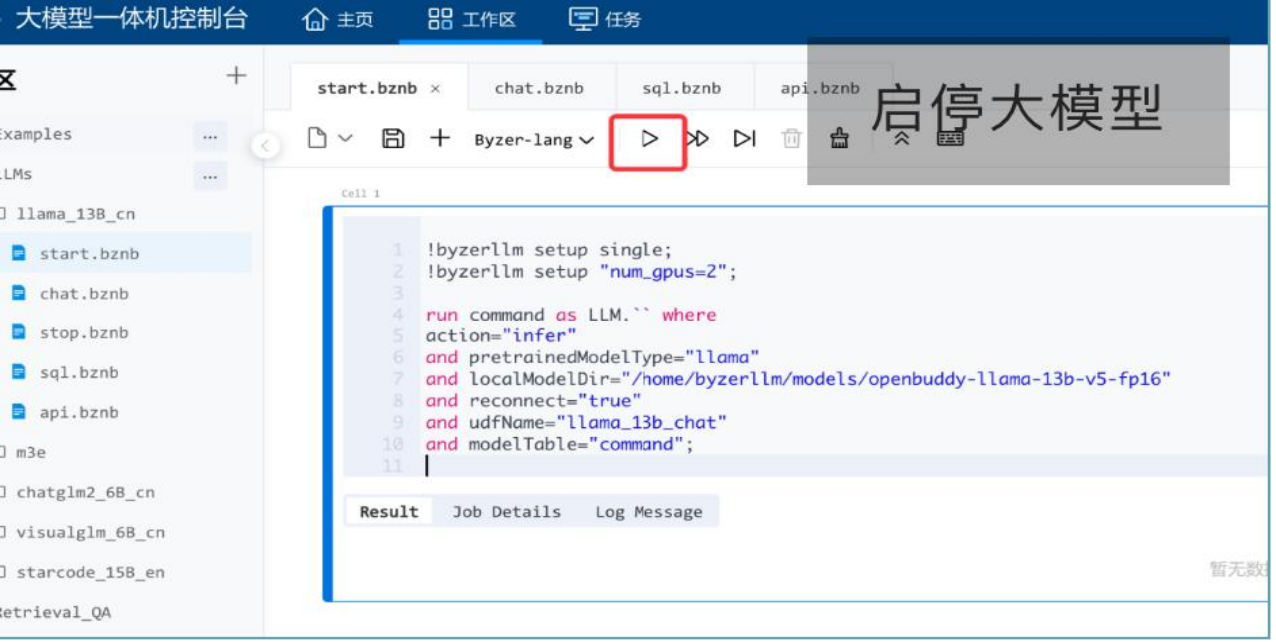
用户不会写 SQL 也没问题

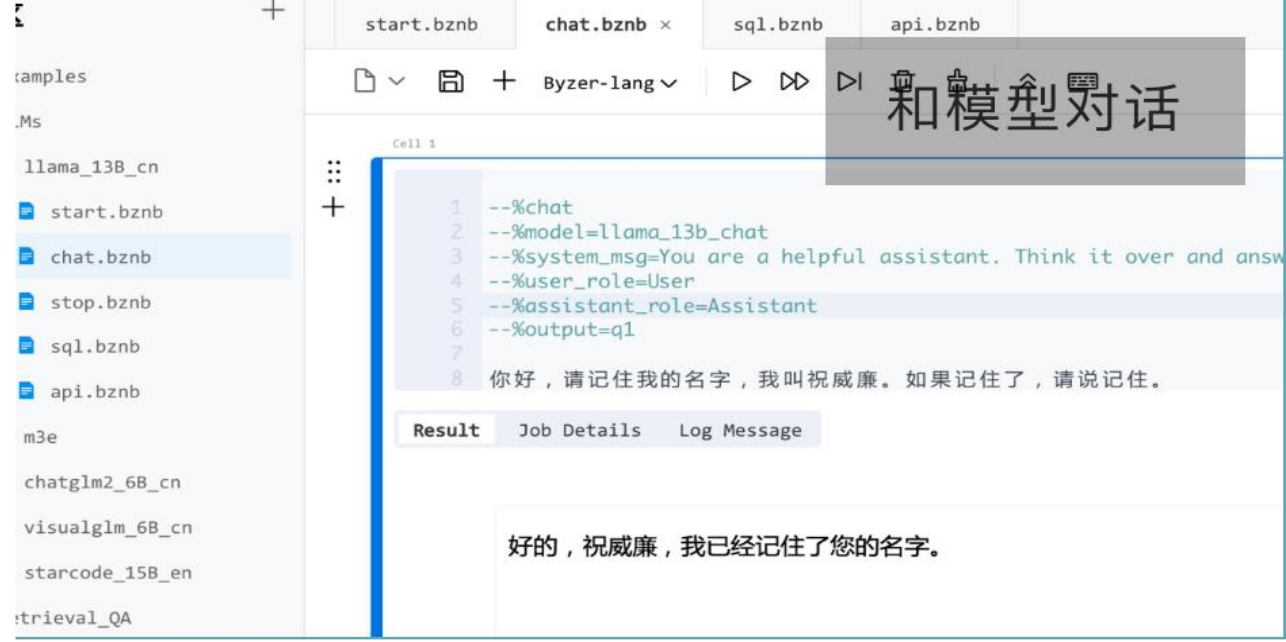


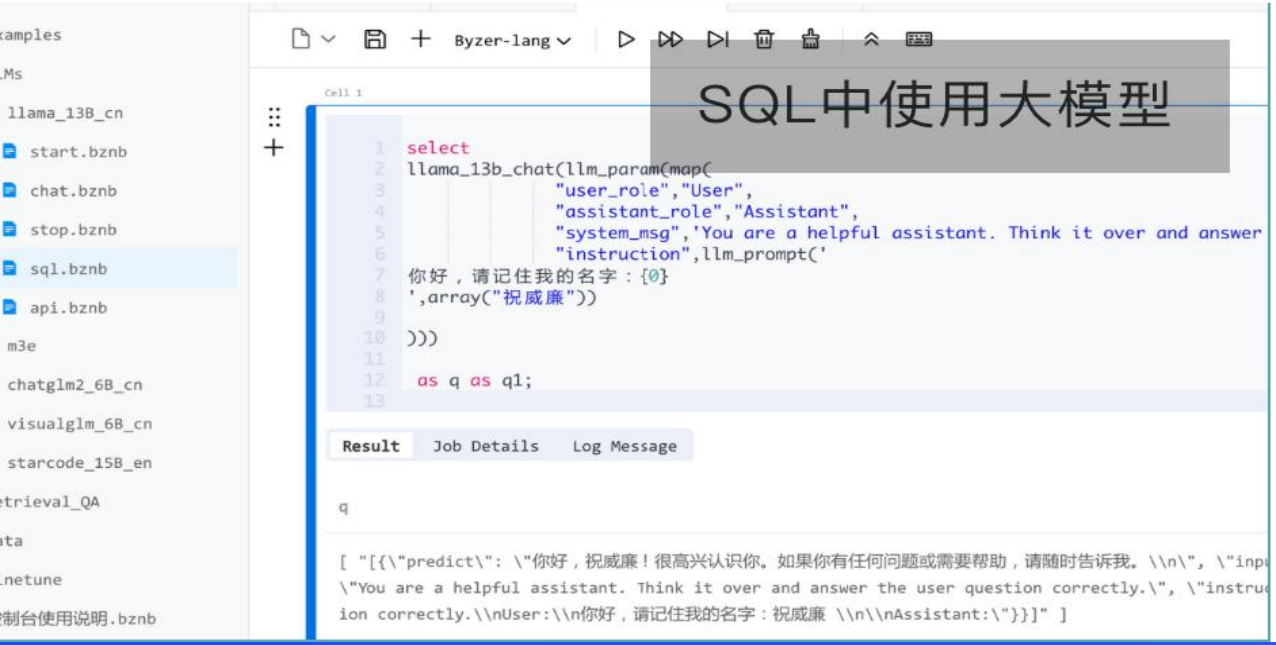
随时获取表 Schema / 支持各种开源/Saas 模型 / 可在 NoteBook 中直接问询 / 支持多轮对话 / 生成SQL 可直接运行

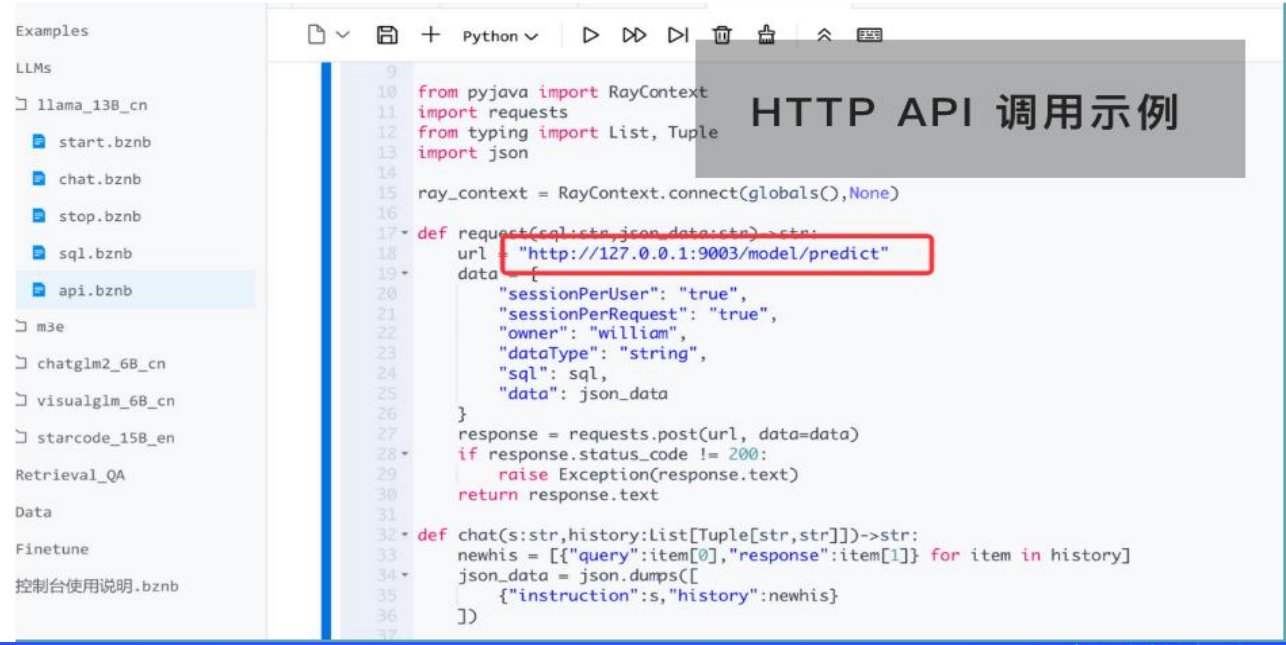
大模型全生命周期管理

大数据管理


大模型一体机控制台
主页
工作区
任务
start.bznb
chat.bznb
sql.bznb
api.bznb
Byzer-lang
!byzerllm setup single;
!byzerllm setup "num_gpus=2";
run command as LLM.`` where
action="infer"
and pretrainedModelType="llama"
and localModelDir="/home/byzerllm/models/openbuddy-llama-13b-v5-fp16"
and reconnect="true"
and udfName="llama_13b_chat"
and modelTable="command";
Result
Job Details
Log Message
启停大模型


start.bznb
chat.bznb
sql.bznb
api.bznb
Byzer-lang
--%chat
--%model=llama_13b_chat
--%system_msg=You are a helpful assistant. Think it over and answ
--%user_role=User
--%assistant_role=Assistant
--%output=q1
你好，请记住我的名字，我叫祝威廉。如果记住了，请说记住。
Result
Job Details
Log Message
好的，祝威廉，我已经记住了您的名字。
和模型对话


Byzer-lang
select
llama_13b_chat(llm_param(map(
"user_role","User",
"assistant_role","Assistant",
"system_msg",'You are a helpful assistant. Think it over and answer
"instruction",llm_prompt('
你好，请记住我的名字：{0}
',array("祝威廉"))
)))
as q as q1;
Result
Job Details
Log Message
q
SQL中使用大模型


Python
from pyjava import RayContext
import requests
from typing import List, Tuple
import json
ray_context = RayContext.connect(globals(),None)
def request(sql:str,json_data:str)->str:
url = "http://127.0.0.1:9003/model/predict"
data = {
"sessionPerUser": "true",
"sessionPerRequest": "true",
"owner": "william",
"dataType": "string",
"sql": sql,
"data": json_data
}
response = requests.post(url, data=data)
if response.status_code != 200:
raise Exception(response.text)
return response.text
def chat(s:str,history:List[Tuple[str,str]])->str:
newhis = [{"query":item[0],"response":item[1]} for item in history]
json_data = json.dumps([
{"instruction":s,"history":newhis}
])
HTTP API 调用示例

如何部署一个模型

Byzer-lang 代码	代码描述
load delta.`ai_model.llama2_70b_model` as llama2_70b_model;	模型和数据集一样，都是采用库表来管理 这里加载模型表
!byzerllm setup single; run command as LLM.`` where action="infer" and pretrainedModelType="custom/llama2" and udfName="llama2_70b_chat" and modelTable="llama2_70b_model";	注册预训练大模型类型为 llama2, 并且将其转换为函数 llama2_70b_chat, 模型来自前面加载的 llama2_70b_model 模型表

如何使用模型

Byzer-lang 代码	代码描述
select llama_30b_chat(llm_param(map("instruction",llm_prompt(' 你好，请记住我的名字：{0} ',array("祝威廉")))))as q as q1;	在ETL、流式计算中 作为一个普通 UDF 函数来使用
set q = """{\"instruction\":\"类型#裤*版型#宽松*风格#性感*图案#线条*裤型#阔腿裤 \",\"output\":\"NAN\"}"""; !sh curl -XPOST 'http://127.0.0.1:9003/model/predict' -d 'sessionPerUser=true&owner=william&dataType=string&sql=select finetune_model_predict(array(feature)) as a &data=[${q}]';	Byzer-LLM 提供了 model/predict endpoint 支持 HTTP API调用

如何 Finetune 一个模型

Byzer-lang 代码	代码描述
load jdbc.`business.tunningData` where url="jdbc:mysql://127.0.0.1:3306/business" and driver="com.mysql.jdbc.Driver" and user="root" and password="${PASSWORD}" as tunningData; select content as instruction,"" as input,summary as output from tunningData as formatTunningData;	从 MySQL 加载业务数据库，并且修改字段名称
!byzerllm setup sft; !byzerllm setup "num_gpus=4"; run command as LLM.`` where pretrainedModelType="sft/llama2" and localModelDir="/home/byzerllm/models/Llama-2-7b-chat-hf" and model="command" and detached="true" and inputTable="sft_data" and outputTable="llama2_300" and `sft.int.max_seq_length`="512" and `sft.int.logging_steps`="1";	指定资源用 QLora 进行 fintune 最后得到的模型重新保存到数据湖里

并发及资源控制

控制模型单实例 GPU 资源

部署前执行：!byzerllm setup "num_gpus=0.4";

系统会完成自动调度

控制模型的部署实例数

部署前执行：!byzerllm setup "maxConcurrency=10";

表示可以同时支持10个并发请求

资源池管理

部署前执行： !byzerllm setup "rayAddress=127.0.0.1:10001";

表示会连接特定的一个资源池，从而得到诸如 GPU/CPU等资源

Byzer 使用 Hrid Runtime，使用 Ray 来完成 GPU/CPU 资源的管理和调度

在 SQL 和大模型融合中的多项设计创新

Model as UDF

- 首推模型即 SQL 函数概念
- 无需开发，一键注册主流私有化和SaaS模型
- 从几KB的模型到几百G的大模型

SQL 支持多轮对话

- 每张临时表就是一个对话，对话使用临时表进行上下文衔接
- 使用实体表完成持久化

NoteBook Copilot

- 将注释和SQL之间实现互相转换

优雅的扩展 SQL 语法

- 一行 SQL 完成大模型预训练
- 一行 SQL 完成大模型微调
- 一行 SQL 完成大模型 UDF 部署

模型和数据统一按表管理

- 模型和数据，都以表形态管理
- 模型和数据，都可以存储在内置数据湖中

在 SQL 和大模型融合中实现多项技术打磨

先进的模型x集群部署

- 模型部署实现 UDFMaster-> UDFWorker 集群模式
- 可分别控制单个模型的 GPU 数和模型实例数
- 基于 Ray 完成了 Transformers, vLLM，,DeepSpeed Inference 以及 Aviary 多种backend 的集成，并且融合进我们设计的 UDFMaster/UDFWorker 模式而非使用 Ray serve
- 同时提供 SQL /Python API

先进的模型预训练/微调能力

- 首次实现基于 Ray 的完全自动化的 Deepspeed 分布式预训练
- 默认实现了多种模型的 QLora 单机多卡的微调能力
- 同时提供 SQL 和 Python API

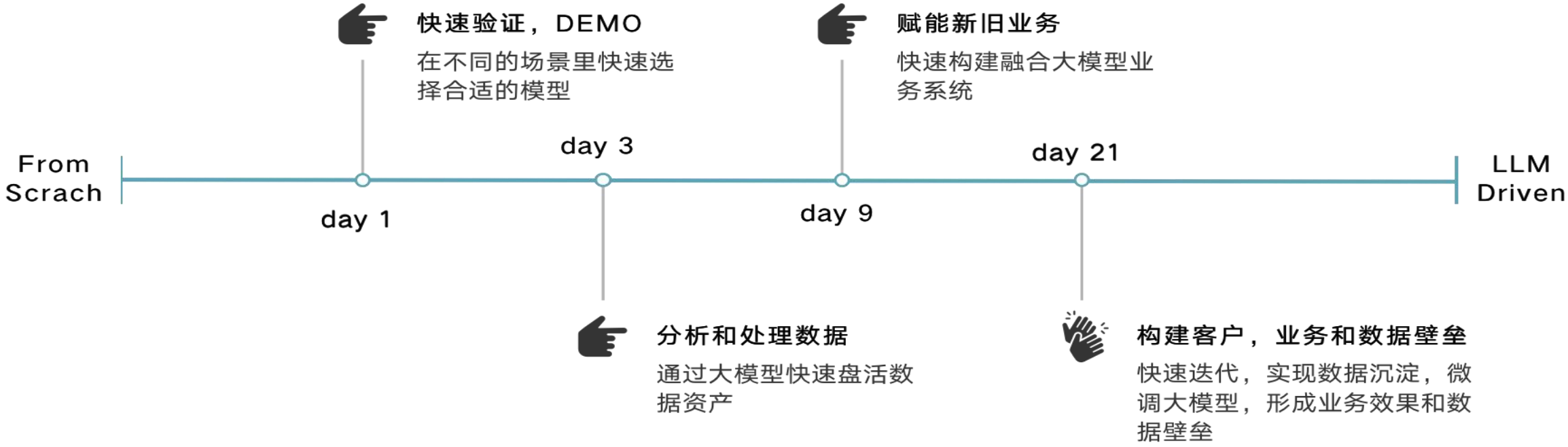
更加成熟稳定的 Data + AI 能力

- 强大的大模型数据处理能力
- 大模型训练的数据分发能力
- 模型和数据都支持在数据湖中存储

SQL 和 Python 融合

- Byzer-python 使得 SQL 和 Python 之间可以互相访问数据，实现无缝互通
- 可以在 Java/Scala 中轻松调
- 支持 Arrow 格式数据传输

PART 05
如何快速将 Byzer 应用于企业业务中


快速验证，DEMO
在不同的场景里快速选择合适的模型
赋能新旧业务
快速构建融合大模型业务系统
From Scrach
day 1
day 3
分析和处理数据
通过大模型快速盘活数据资产
day 9
day 21
构建客户，业务和数据壁垒
快速迭代，实现数据沉淀，微调大模型，形成业务效果和数据壁垒
LLM Driven

大模型时代，唯快不破

SDK 访问

SDK 辅助生成 SQL

```
val byzer = Byzer()
val jsonString = byzer.cluster.engine.url("http://127.0.0.1:9003/run/script").owner("jack").end.
backendStrategy(new JobNumAwareStrategy("")).
end.
load.format("csv").path("/tmp/jack").
options().add("header", "true").end.tag("table1").end.filter.
and.add(Or(Expr(Some("a>1")), Expr(Some("b>1")))).add(Expr(Some("c==1"))).end.end.
toJson(true)

var byzer2 = Byzer()
byzer2 = byzer2.fromJson(jsonString)
assert(byzer2.toJson(true) == jsonString)
```

Byzer-client-sdk 项目：Java/Scala SDK 快速生成 SQL 代码

Rast 接口访问

Rast 接口调用 SQL

```python
def request(sql:str,json_data:str)->str:
    url = "http://127.0.0.1:9003/model/predict"
    data = {
        "sessionPerUser": "true",
        "sessionPerRequest": "true",
        "owner": "william",
        "dataType": "string",
        "sql": sql,
        "data": json_data
    }
    response = requests.post(url, data=data)
    if response.status_code != 200:
        raise Exception(response.text)
    return response.text

def chat(s:str,history:List[Tuple[str,str]])->str:
    newhis = [{"query":item[0],"response":item[1]} for item in history]
    json_data = json.dumps([
        {"instruction":s,"history":newhis}
    ])

    response = request("select llama_30b_chat(array(feature)) as value",json_data)


    t = json.loads(response)
    t2 = json.loads(t[0]["value"][0])
    return t2[0]["predict"]
```

用户直接调用 HTTP 接口传递 SQL 语句

JDBC 接口访问

JDBC 调用 SQL（实验 不可用）

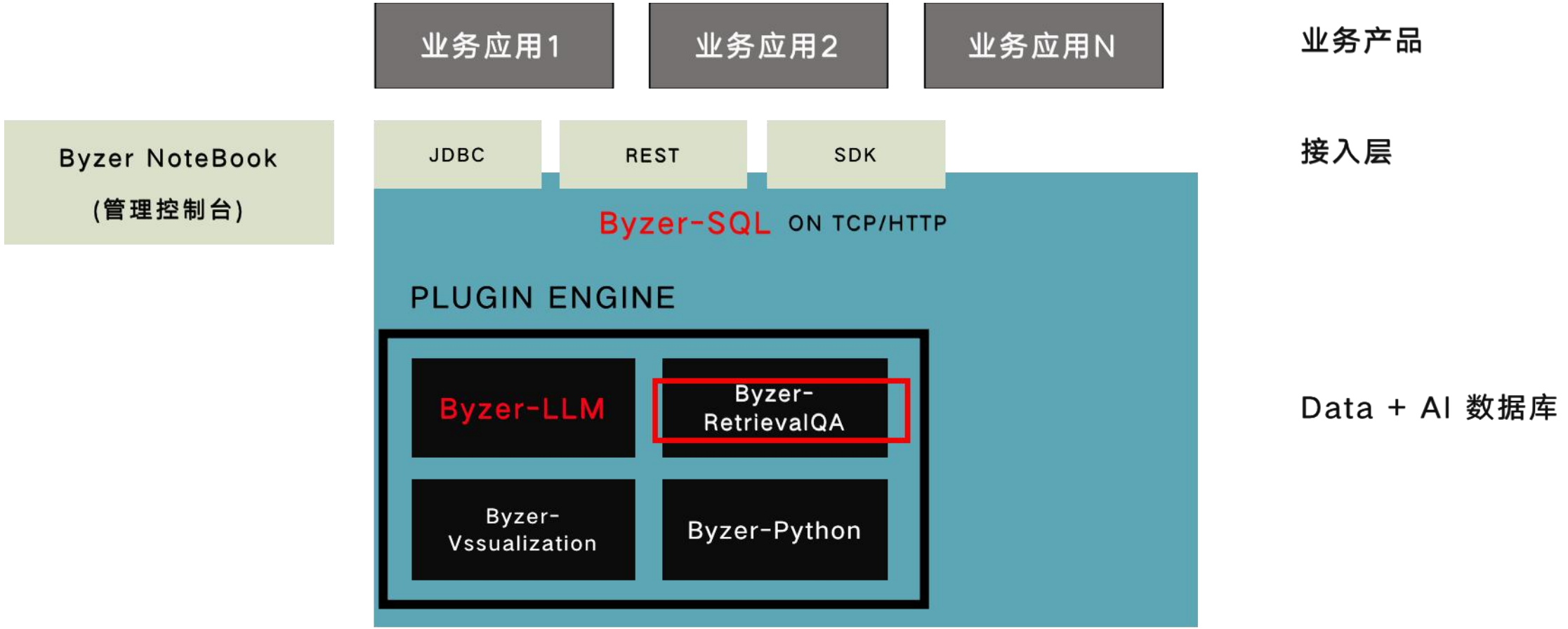
```
Class.forName("tech.mlsql.api.jdbc.MLSQLDriver")
val properties = new Properties()
properties.put("sqlType", "mlsql")
val conn = DriverManager.getConnection("jdbc:mlsql://127.0.0.1:9003/test?user=william&password=)

val stat = conn.prepareStatement(
  """
    |select 1 as a,TIMESTAMP("2017-07-23 00:00:00") as k, current_date() as v, "wow'" as jack as
    |select * from b where a=? and jack=? as output;
    |""".stripMargin)
stat.setInt(1, 1)
stat.setString(2, "wow'")
val rs = stat.executeQuery()
while (rs.next()) {
  println(rs.getString("a"))
  println(rs.getTimestamp("k"))
  println(rs.getDate("v"))
}
rs.close()
stat.close()
conn.close()
}
```

mlsql-jdbc 驱动，使用标准的 JDBC 协议传递 SQL 语句

Byzer-RetrievalQA:

基于大模型的问答知识库引擎插件

所在位置


业务应用1
业务应用2
业务应用N
业务产品
Byzer NoteBook
(管理控制台)
JDBC
REST
SDK
接入层
Byzer-SQL ON TCP/HTTP
PLUGIN ENGINE
Byzer-LLM
Byzer-RetrievalQA
Byzer-Vssualization
Byzer-Python
Data + AI 数据库

使用私有数据构建基于大模型的问答知识库

基于 Byzer 大模型全生命周期管理能力上的知识库引擎

构建和查询全部分布式

可以应对大规模的私有数据

EMBEDDING服务成本可控

Eembedding服务可能是最耗时和消耗成本的部分，我们支持使用内置开源模型作为可选Embedding服务

完整，成熟，稳定

依托于Byzer自身成熟的稳定的以及支持的海量的的数据连接和处理能力，我们可以快速稳定的从业务数据构建知识库

Byzer-LLM 可以快速帮助用户将私有数据提供基于大模型的问答能力

四步纯 SQL 快速构建知识库示例

第一步

获取业务数据

```
load jdbc.`business.ByzerDoc` where
url="jdbc:mysql://127.0.0.1:3306/busines
as ByzerDoc;

select file as source,
value as page_content
from ByzerDoc as newData;
```

第二步

部署 Embedding服务

```
run command as LLM.`` where
action="infer"
and pretrainedModelType="chatglm"
and udfName="chat"
and modelTable="chatglm-6b-int4";
```

第三步

构建向量数据库

```
run command as
LLMQABuilder.``
where inputTable="newData2"
and outputTable="qaIndex";

save overwrite qaIndex as
delta.`ai_model.qa3`;
```

第四步

部署加持向量外挂的大模型服务

```
run command as LLM.`` where
action="infer"
and pretrainedModelType="qa"
and udfName="qa"
and modelTable="qa_model";
```

加持业务数据后的大模型效果

```
1  select llm_response_predict(chatglm_chat(llm_param(map(
2  "instruction","Byzer 是什么",
3  "temperature",0.1
4  )))) as response as output;
```

Result | Job Details | Log Message

response

"Byzer" 不是一个常见的单词，可能是一个拼写错误或一个特定的术语。如果能提供更多上下文信息，我将尽力回答您的问题。

共 1 条 < 1 >

原始模型效果： 要么不知道，要么完全一本正经胡编乱造

加持业务数据后的大模型效果

Cell 11

Byzer-lang

```
--查看完整结果
select docs_qa(array(to_json(map(

"instruction","Byzer 是什么",
"k",3,
"prompt","${template}",
"temperature",0.1
)))) as response as output;
```

Result | Job Details | Log Message

response

["[{\"predict\": \"Byzer 是一种面向大数据和人工智能的分布式语言，提供了强大的可编程能力和语法糖，使得开发者可以像传统编程语言一样组织和管理代码。它支持多种数据存储和处理技术，包括 Apache Spark、Apache Ray 和 Apache Flink 等。同时，Byzer 还支持运行 Python 脚本，使得开发者可以将 Byzer 集成到现有的 Python AI 项目中。Byzer 旨在帮助开发者快速构建高效的分布式计算引擎，从而更好地落地 Data+AI 业务。\", \"input\": {\"instruction\": \"Byzer 是什么\",

共 1 条 < 1 > 前往 1 页

知识库效果：正确回答用户的问题

PART 06
Byzer 数据库现状和未来的发展

Byzer 数据库现状和未来的发展

现状

- 主项目 Byzer 历经 7 年开发，1.8K Star
- ETL / 数据分析广泛落地
- Byzer 数据库（Byzer-LLM）部分正在快速落地

Byzer 数据库现状和未来的发展

Byzer-RETRIEVAL （新项目）

1. Byzer-RETRIEVAL 为 RAG 而生，基于 Ray 的分布式检索系统

2. 项目地址： https://github.com/allwefantasy/Byzer-retrieval

3. 介绍： https://docs.Byzer.org/#/public/blog/en-us/how-to-use-Byzer-sql-to-build-rag-based-application.md

支持全文检索
向量相似度召回

多路召回融合打分

分布式，性能优异

AiDD AI+ 软件研发数字峰会
AI+ software Development Digital summit

THANKS