AI ProCon 开发者大会

CSDN

快速构建高性能AI应用

AI特征数据库技术实践

2019-09-07

第四范式资深架构师 陈迪豪

Agenda

1. AI场景的计算存储一体化趋势介绍
2. AI全流程解决方案——AI特征数据库架构设计
3. AI特征数据库的特征计算和核心存储组件剖析
4. AI特征数据库快速构建AI应用及真实场景实践

Agenda

1. AI场景的计算存储一体化趋势介绍
2. AI全流程解决方案——AI特征数据库架构设计
3. AI特征数据库的特征计算和核心存储组件剖析
4. AI特征数据库快速构建AI应用及真实场景实践

AI Landing

理想的AI场景落地步骤：

1. 数据收集
2. 模型训练
3. 应用上线

建模门槛高

数据预处理

精通SQL、Pyhton等

熟悉业务场景

时序特征构建

灵活调试迭代

实时场景上线成本高

数据对接存储

线上方案转换

一致性校验

高性能硬实时

场景痛点

人员瓶颈

数据工程师

数据科学家

研发工程师

AI系统对接开发

工程量大

数据管理、数据回流等

实时特征计算

运维监控等

读写分离

AI Landing

以某银行事中反欺诈场景为例：

- 实时性要求强，P99响应时间在20ms以内，包括特征生成和模型预估端到端过程
- 建模特征较多，包括时序和单行特征1000多个，需要保证线上线下特征计算一致性
- 要求在线划窗，部分特征包含前序数据进行聚合，只能通过时序存储和预计算实现
- 模型迭代周期短，需要定期全量数据重新训练，预估结果需要回流形成自学习闭环

AI Landing

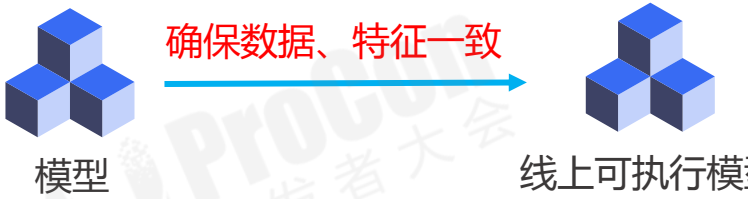
开源的AI“解决方案”：

- 数据处理框架：Spark / Featuretools / ... ≠ 离线在线特征一致性
- 机器学习框架：TensorFlow / Sklearn ... ≠ 高维机器学习场景模型
- 在线预估服务：TensorFlow Serving / ... ≠ 端到端模型预估服务
- 数据存储服务：HDFS / MySQL / ... ≠ AI特征计算存储服务
- 机器学习平台：TFX / Google CloudML / ... ≠ 模型管理和自学习平台

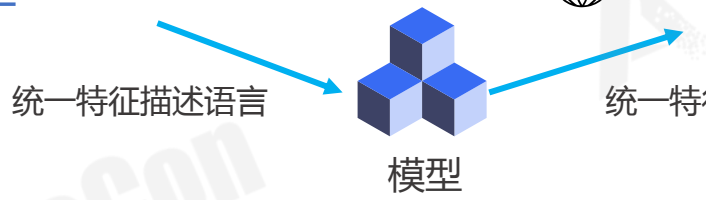
AI Landing

一致性的重要性

离线环境：AI建模　　线上环境：AI应用

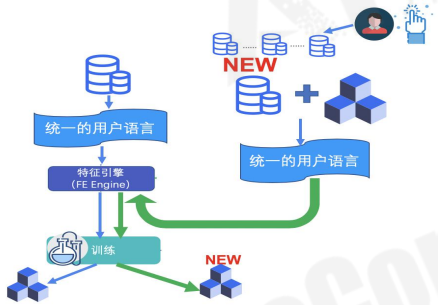


离线环境：AI建模　　线上环境：AI应用



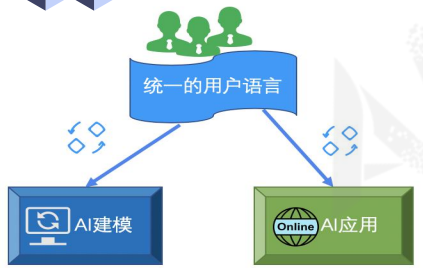
解决方案

数据一致性



将用户新行为数据迅速加入闭环
发现的用户行为特征立即进入模型

计算一致性

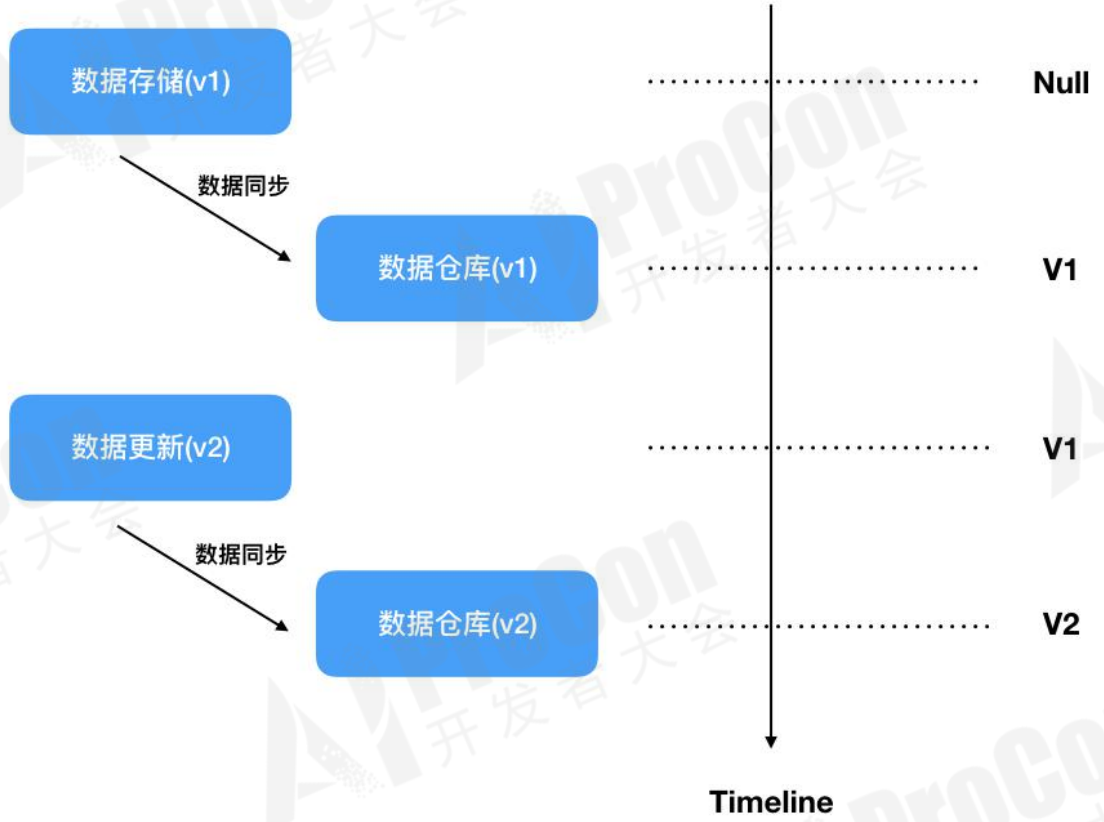


将统一语言自动转译为线上线下代码
线下与线上的模型计算逻辑一致

AI Landing

计算存储一体化重要性：

- 数据时效性
- 特征预聚合
- 软硬一体优化
- 离线在线一致性



AI Landing

计算存储一体化趋势：

Agenda

1. AI场景的计算存储一体化趋势介绍
2. AI全流程解决方案——AI特征数据库架构设计
3. AI特征数据库的特征计算和核心存储组件剖析
4. AI特征数据库快速构建AI应用及真实场景实践

AI特征数据库

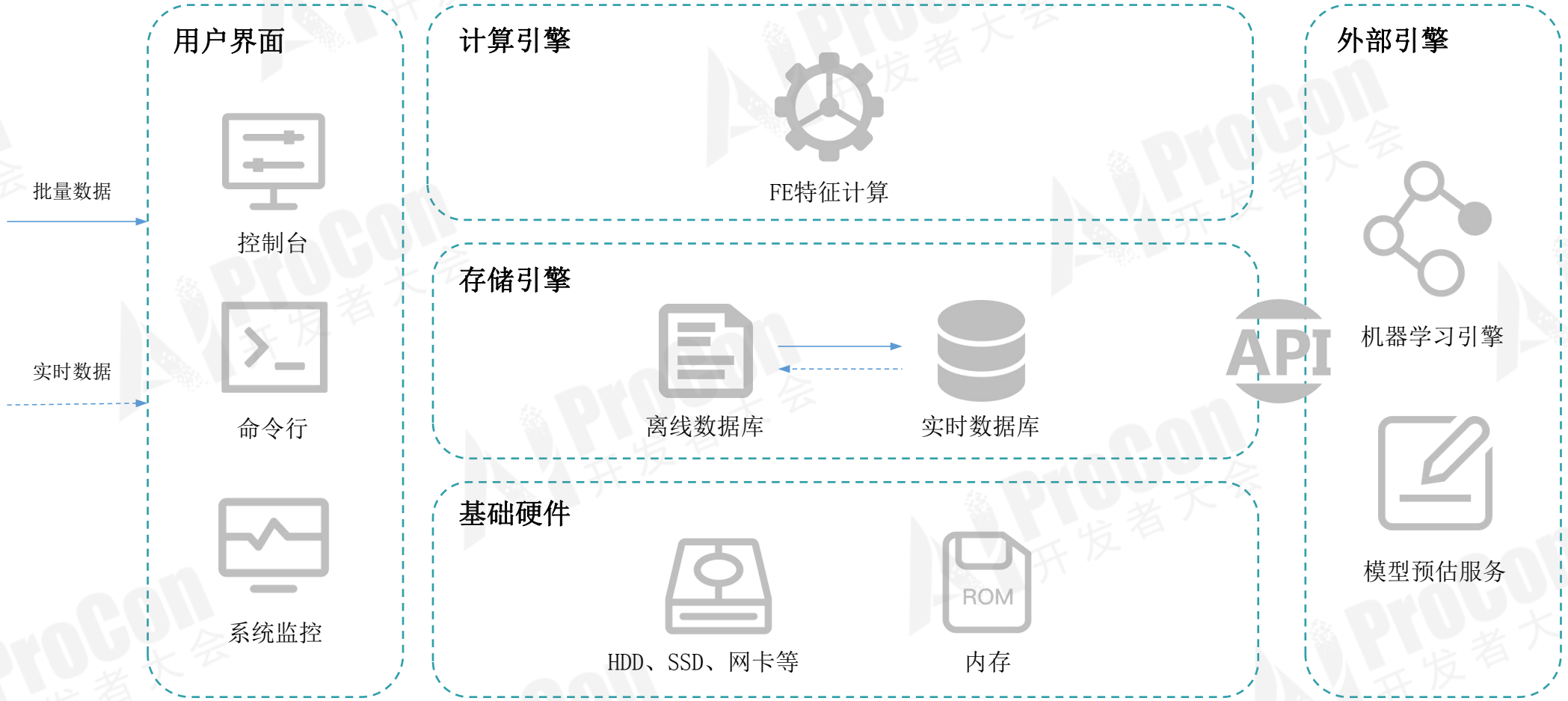
- 解决计算存储一体化问题
- 计算：离现在线一致性计算（特征描述语言）

  高性能特征抽取与数据预处理

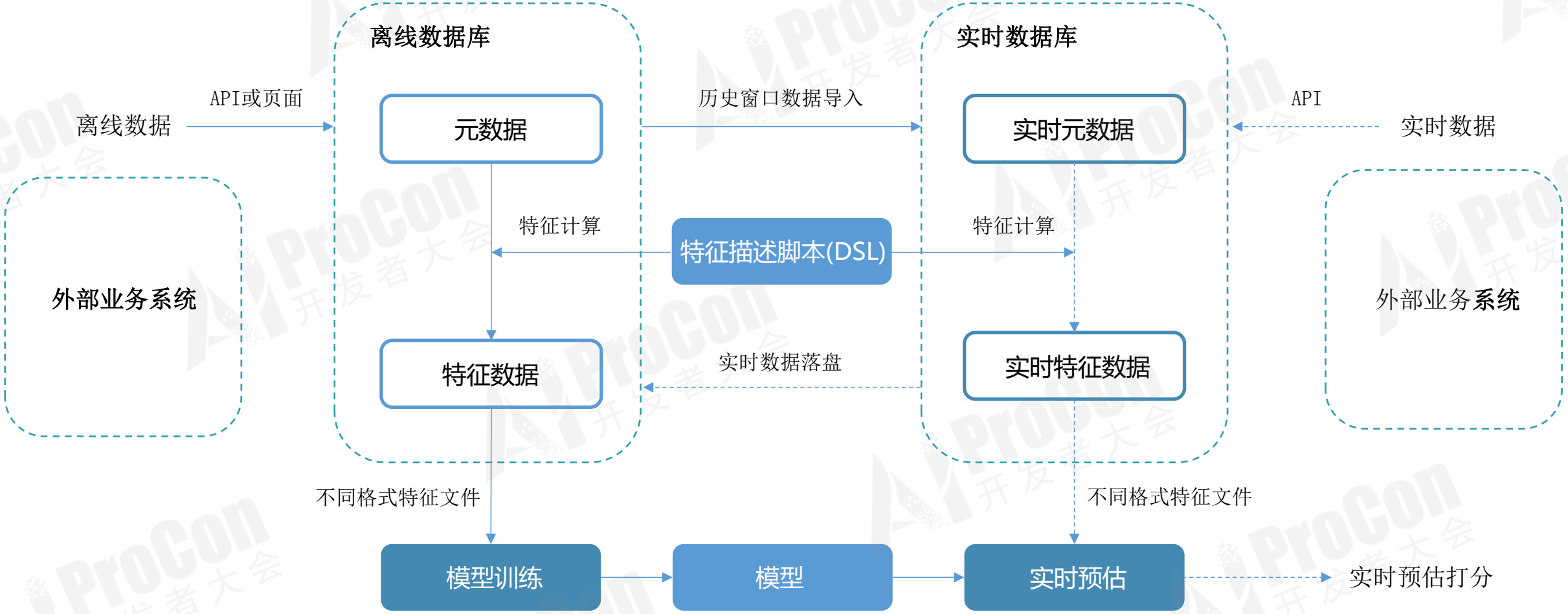
  端到端的模型预估全流程
- 存储：高性能时序特征存储

  预估数据回流与持久化

AI特征数据库：业务架构



AI特征数据库：系统架构



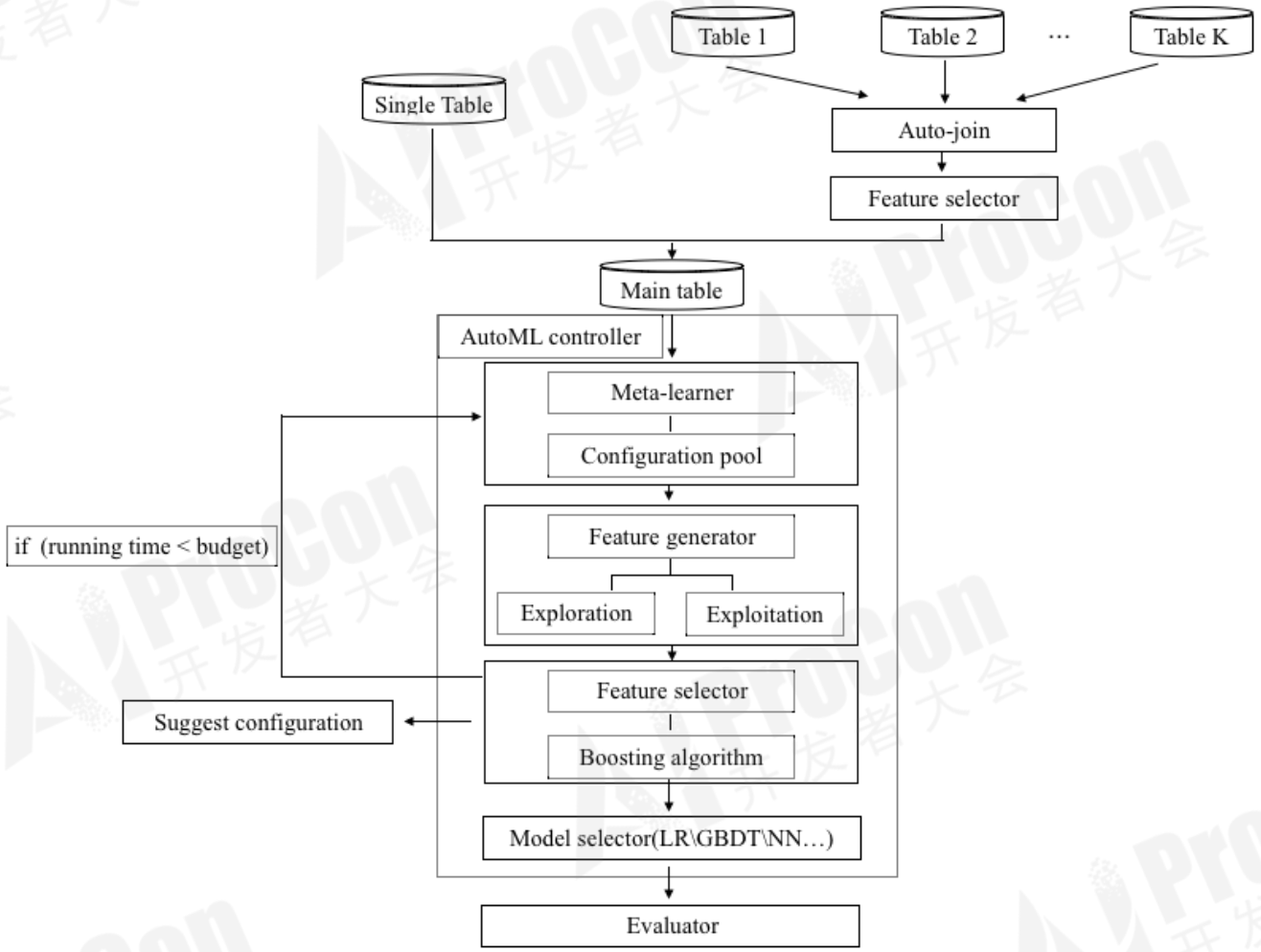
AI特征数据库：特性

特征描述语言（DSL）

```
# 创建窗口特征
w_ef_t1_test = window(t1, "test")
# 单行计算特征
int_ef_0 = column(int(w_ef_t1_test.Float[0]))
log_ef_7 = column(log(w_ef_t1_test.Float[0]))
mapping_ef_9 = column(mapping(w_ef_t1_test.Float[0], "1.4=1"))
# 日期计算特征
month_ef_6 = column(month(w_ef_t1_test.Date[0]))
weekday_ef_8 = column(dayofweek(w_ef_t1_test.Date[0]))
# 条件计算特征
weekend_ef_9 = column(case when(dayofweek(w_ef_t1_test.Date[0]) in (6, 7)) then 1 else 0 end)
# 特殊窗口特征
w1 = window(table=t1, other_table =[t2], keys=[int_CK], order=timestamp_asc_ts, max_size=5, offset=3000)
f1 = column(sum(w1.amt) / count(w1.amt))
# 拼表计算特征
out_table1 = left_join(w1_out, w2_out, "w1_out.merchant = w2_out.merchant and w1_out.ts >= w2_out.f5")
# 特殊拼表特征
out_table2 = last_join(out_table1, w3_out, "out_table1.f_city=w3.f8 and out_table1.ts >= w3.f7")
```

AI特征数据库：特性

自动特征工程（AutoML）



AI特征数据库：特性

- 实用性，解决机器学习特征抽取及上线痛点的计算存储一体化平台
- 高性能，保证线上特征的模型效果与离线相同，保证线上预估服务高性能
- 拓展性，支持单行或多行特征计算，支持时序特征、日期和地点特征计算等
- 低成本，提供标准化的特征抽取语言，减少二次开发成本，上线效率提升30%-50%
- 灵活性，支持API和多语言SDK，特征抽取结果可用于自研或开源机器学习框架
- 自动化，支持高效的AutoML算法，可生成有效特征组合并直接应用于线上服务

Agenda

1. AI场景的计算存储一体化趋势介绍
2. AI全流程解决方案——AI特征数据库架构设计
3. AI特征数据库的特征计算和核心存储组件剖析
4. AI特征数据库快速构建AI应用及真实场景实践

特征数据库计算引擎

User

API接口 | SDK类库 | CLI命令行 | Workflow等

Core

特征描述语言 | 脚本校验服务 | 可视化特征生成 | 自动特征组合算法

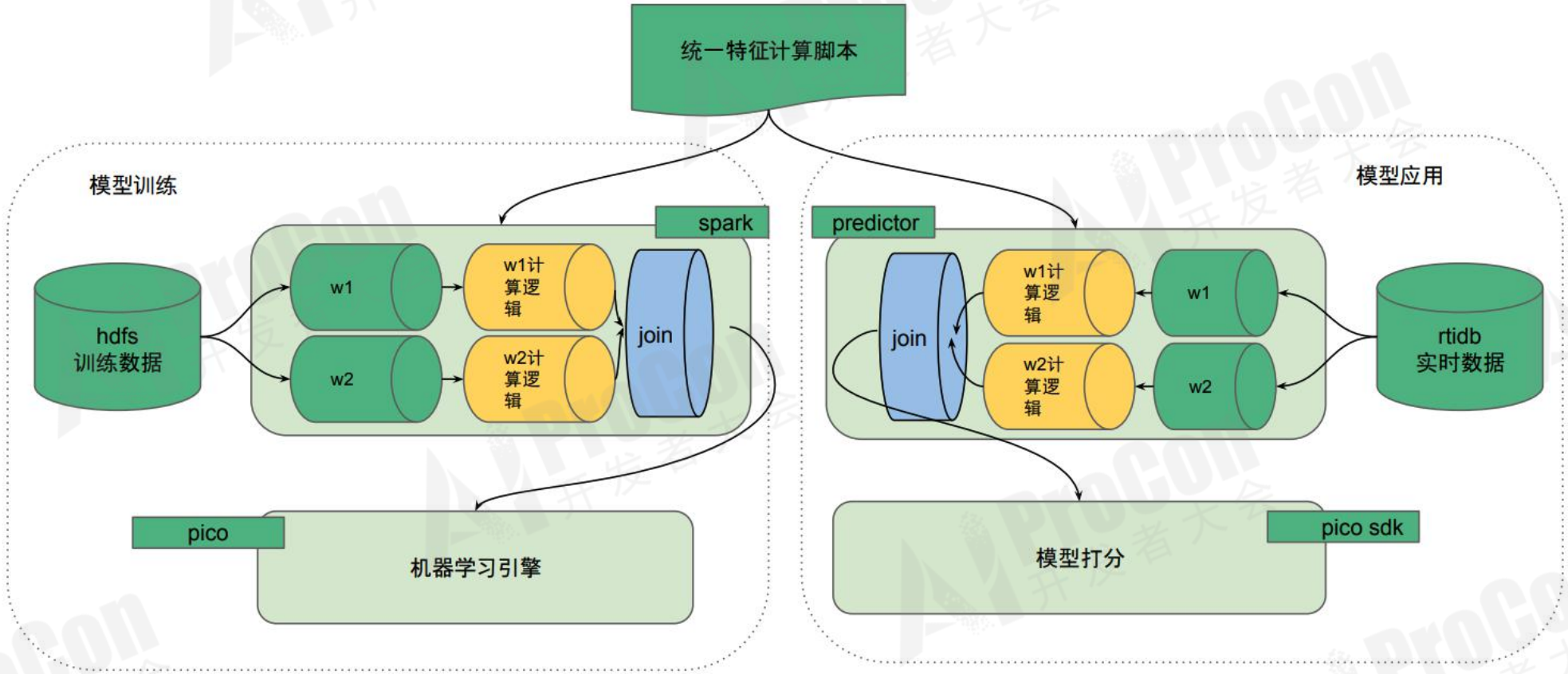
Executor

离线计算（Yarn等） | 在线计算（Predictor等）

特征数据库计算引擎

- 统一的在线和离线特征描述语言（DSL）
- 支持多行时序特征、多表拼接等复杂特征计算
- 强类型检查，支持Map、List等复杂数据结构
- 实现Last Join功能，实现超高性能多表单行拼接功能
- 实现跨表Window功能，实现高效的跨表划窗特征生成
- 支持TensorFlow、LightGBM和自研机器学习框架等

特征数据库计算引擎：一致性



特征数据库计算引擎：拓展性

- 预热编译以及内存编译器
- 自洽的DSL语法定义和解析程序
- 动态加载UDF、UDAF等拓展函数和功能
- 优化Spark Window实现，支持CodeGen函数以及跨表窗口计算

特征数据库计算引擎：性能

- 原地生成用户代码并在内存编译，充分利用Java编译器优化生成高性能byte code
- 提供特征级别的异常处理和隔离性，单行单特征的异常数据不影响其他特征生成结果
- 重写优化，对常量或重复表达式进行优化，非导出表达式进行剪枝，可复用中间变量
- 在反欺诈机器学习场景，相同特征比PySpark/SparkSQL提升10倍性能
- 支持RDMA和FPGA硬件加速，高性能版本比通用服务器版本提升3倍性能

特征数据库存储引擎

高性能

高可用

多维度

高吞吐

多用途

分布式可扩展

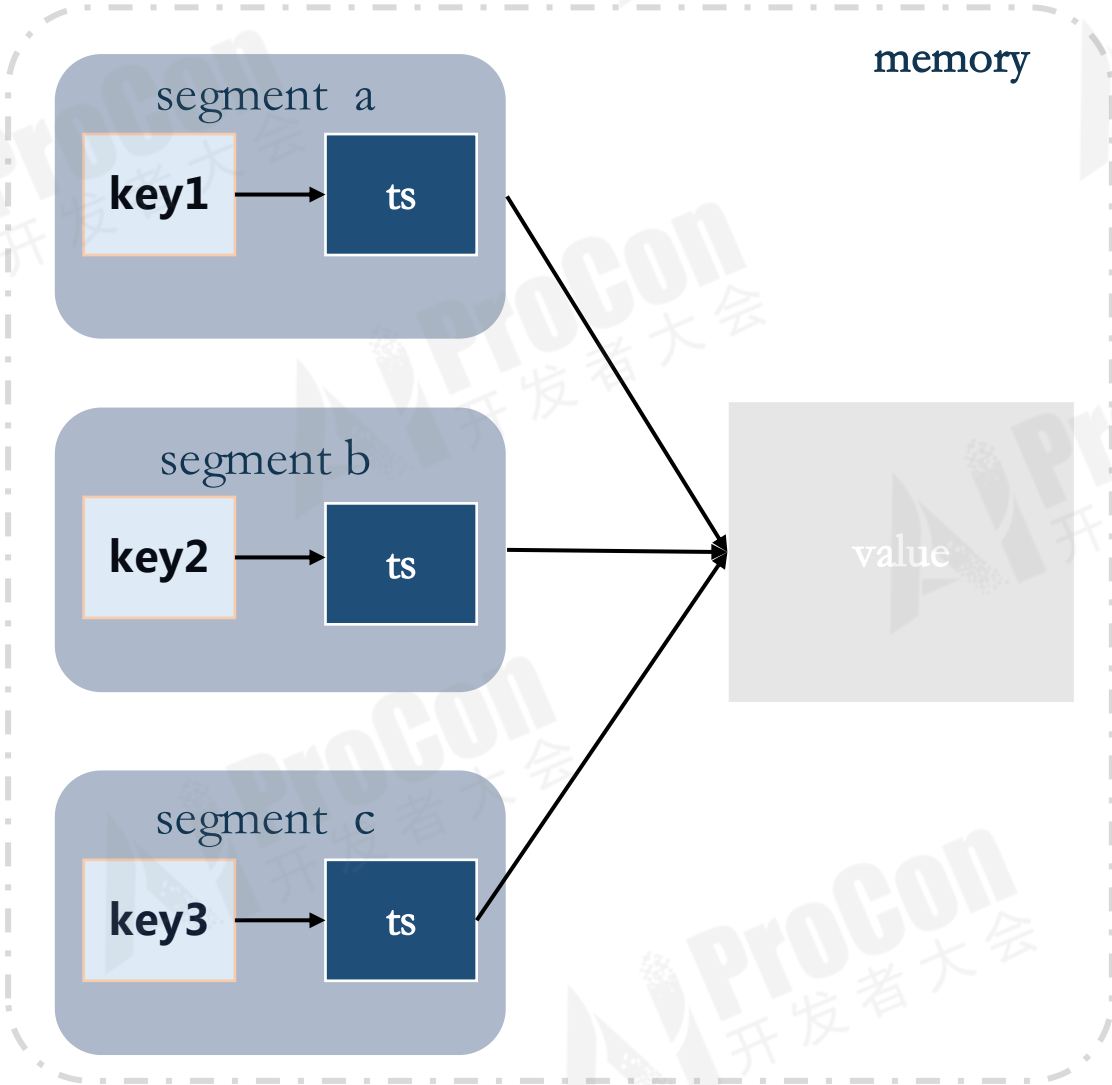
高存储效率

特征数据库存储引擎

- 全内存分布式时序数据库：高性能、高并发、时序优化
- 面向AI的时序数据，存储用户交易行为、IoT设备历史记录等
- 支持按时间维度查询和聚合，支持多维度特征查询
- 实现按时间窗维度或按行数的数据淘汰策略（TTL）
- 基于英特尔傲腾持久内存（AEP）优化，服务器数量可降低3倍以上

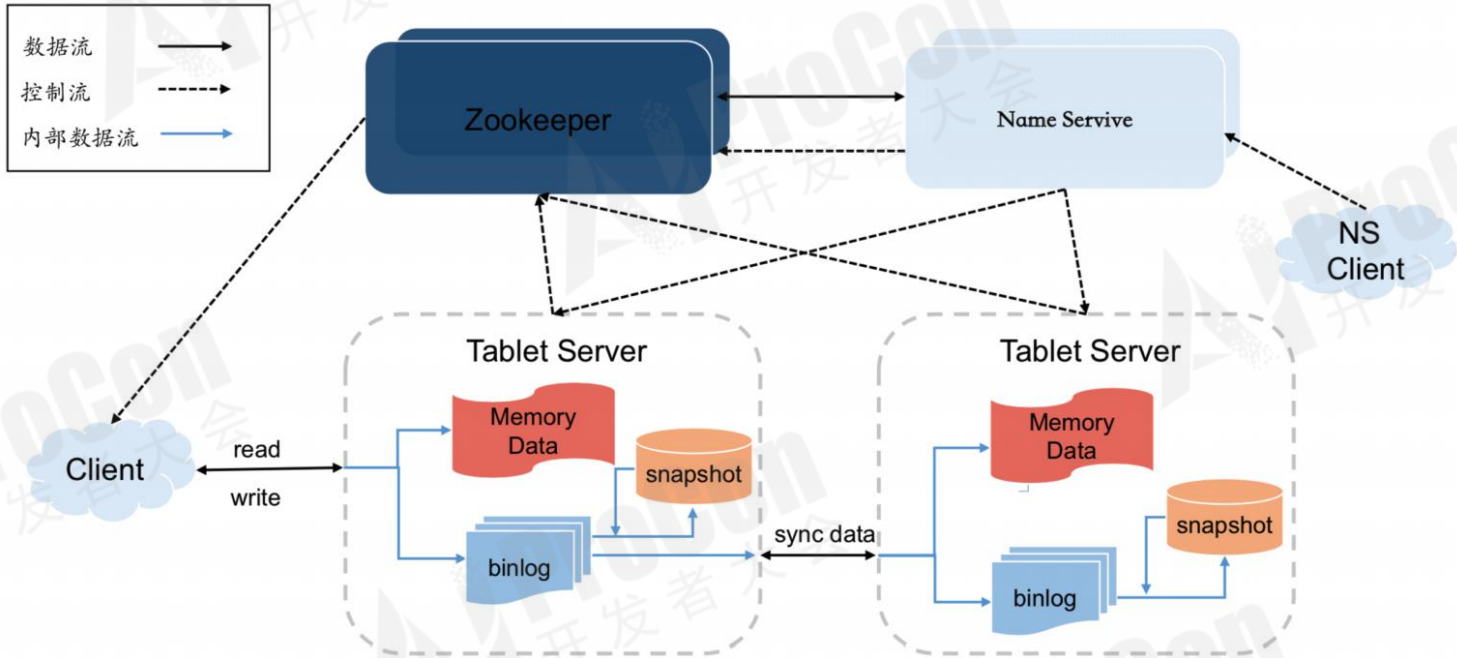
特征数据库存储引擎：设计

- 纯C++实现，GC-free
- 全内存查询，数据可持久化
- 高性能Skip list数据结构
- 多级索引支持多维度查询



特征数据库存储引擎：分布式

- 高可用：分布式锁服务，多节点多备份
- 主从架构，支持动态增加删除节点
- 自动Failover，生产级灾备恢复
- 持久化存储能力，snapshot+binlog



特征数据库存储引擎：性能

性能对比 – RTIDB vs VoltDB

VoltDB是一款流行的内存数据库，具有高性能延时低等优点，有时用于时序类数据的场景中：

内存对比	占用内存	数据量(条)	存储结构
VoltDB	251GB	30,661,433	每条数据16个字段，所有字段值累加之后为256B
RTIDB	22GB	31,169,350	每条数据的value为256B

存储等量的数据，RTIDB对内存的消耗约为VoltDB的 **10%**

特征数据库存储引擎：性能

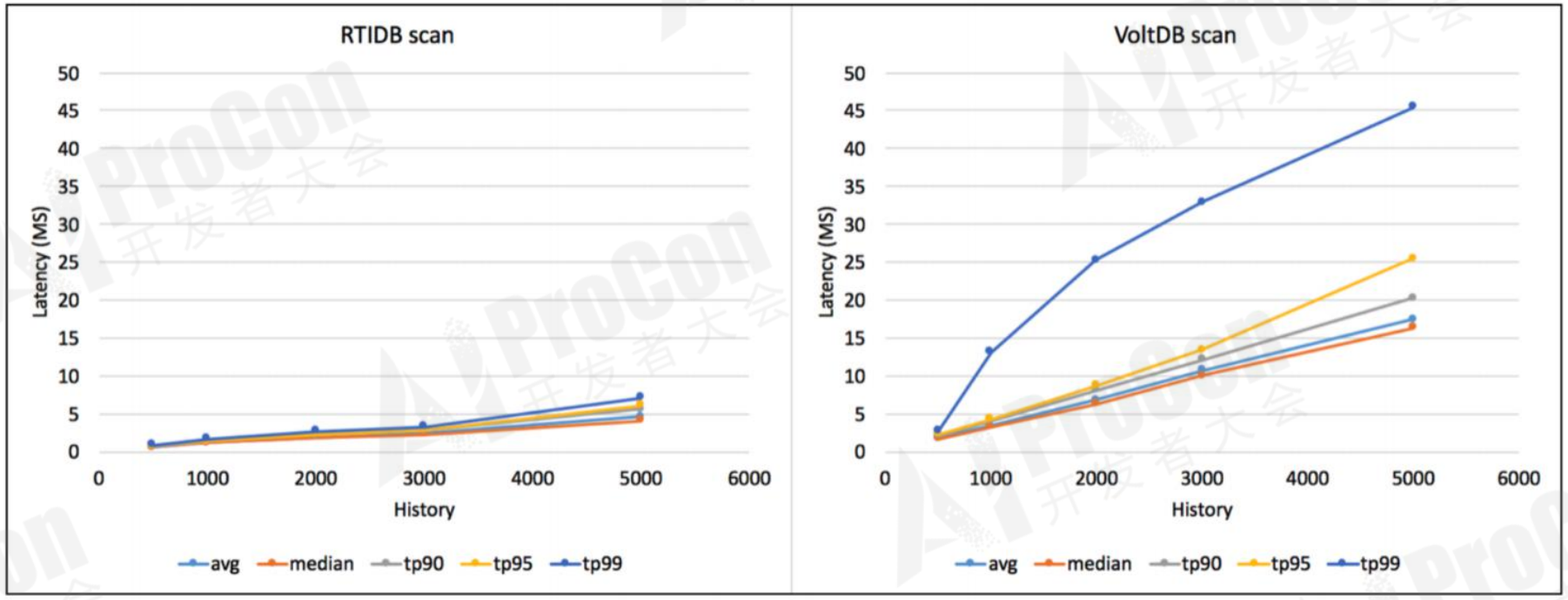
性能对比 – RTIDB vs VoltDB

RTIDB的灌入(put)性能约为VoltDB的 2X

特征数据库存储引擎：性能

性能对比 – RTIDB vs VoltDB

- 数据库至少有3000万条以上数据，数据大小为256Bytes，对数据进行时序查询



RTIDB的查询(select)性能约为VoltDB的 6X

Agenda

1. AI场景的计算存储一体化趋势介绍
2. AI全流程解决方案——AI特征数据库架构设计
3. AI特征数据库的特征计算和核心存储组件剖析
4. AI特征数据库快速构建AI应用及真实场景实践

AI特征数据库实践

- 离线、在线脚本运行
- 特征数据、特征计算脚本管理
- 可视化调试，支持Notebook
- 一键上线，集成高性能预估服务

AI特征数据库实践：Kaggle Competition

1. 下载Kaggle训练数据
2. 使用DSL定义特征计算逻辑 / 使用AutoML自动特征抽取
3. 使用Scikit-learn或TensorFlow进行模型训练
4. 使用Python运行批量预估（可提交Kaggle评分）
5. 使用AI特征数据库上线预估服务

```
// 1. 启动fedb
bash start-fedb.sh

// 2. 模型训练
python model-train.py

// 3. 模型部署
python model-deploy.py

// 4. 启动推荐服务
python fedb-server.py
```

AI特征数据库实践：Expedia Hotel Recommendations

```
feql = """
t1 = left_join(train,destinations,"train.srch_destination_id=destinations
w = window(t1, "user_id", "date_time", 10, 100)\n
stay_span = column(datediff(w.srch_co[0], w.srch_ci[0]))\n
year = column(year(w.date_time[0]))\n
day_of_week = column(dayofweek(w.date_time[0]))\n
site_name = column(w.site_name[0])\n
posa_continent = column(w.posa_continent[0])\n
user_location_city = column(w.user_location_city[0])\n
user_id = column(w.user_id[0])\n
dateis_mobile_time = column(w.is_mobile[0])\n
is_package = column(w.is_package[0])\n
channel = column(w.channel[0])\n
srch_adults_cnt = column(w.srch_adults_cnt[0])\n
srch_destination_type_id = column(w.srch_destination_type_id[0])\n
is_booking = column(w.is_booking[0])\n
cnt = column(w.cnt[0])\n
hotel_continent = column(w.hotel_continent[0])\n
date_hotel_markettime = column(w.hotel_market[0])\n
hotel_cluster = column(w.hotel_cluster[0])\n
"""

status, msg = client.runFeQL("demo", feql)
if not status:
    print("run feql failed: " + msg)
    exit()
print("run feql success")
print(msg)
```

1. 特征抽取

```
X = np.array(data.drop(["hotel_cluster"], 1))
y = np.array(data['hotel_cluster'])
X_train, X_test, y_train ,y_test = model_selection.train_test_split(X,y,test_size=0.2)

clf = RandomForestClassifier(n_estimators=10)
clf.fit(X_train,y_train)
accuracy = clf.score(X_test, y_test)
print(accuracy)
```

2. 模型训练

```
print("Starting deploy...")
feql = util.read_file("model/hotel_fe.fql")
status, msg = client.deployFeQL(deploy_name, ns, feql, False, True, 2, 2)
if not status:
    print("deploy feql failed: " + msg)
    exit()
print("deploy feql success")

status, msg = client.showDeploy(ns)
if not status:
    print("show deploy failed: " + msg)
    exit()
print("show deploy success")
print(msg)
```

3. 模型部署

AI特征数据库实践：B2C Antifraud

交易表原始数据表（2张交易表 + 7张属性表）：

card_no	mac	trx_time	trx_date	amt	merchant	account_id
card1	mac1	1547713539000	2019-01-17	2.0	4451	acct1
card1	mac1	1547713500000	2019-01-17	19.2	3321	acct1
card0	mac3	1547601539000	2019-01-16	2.1	4431	acct3

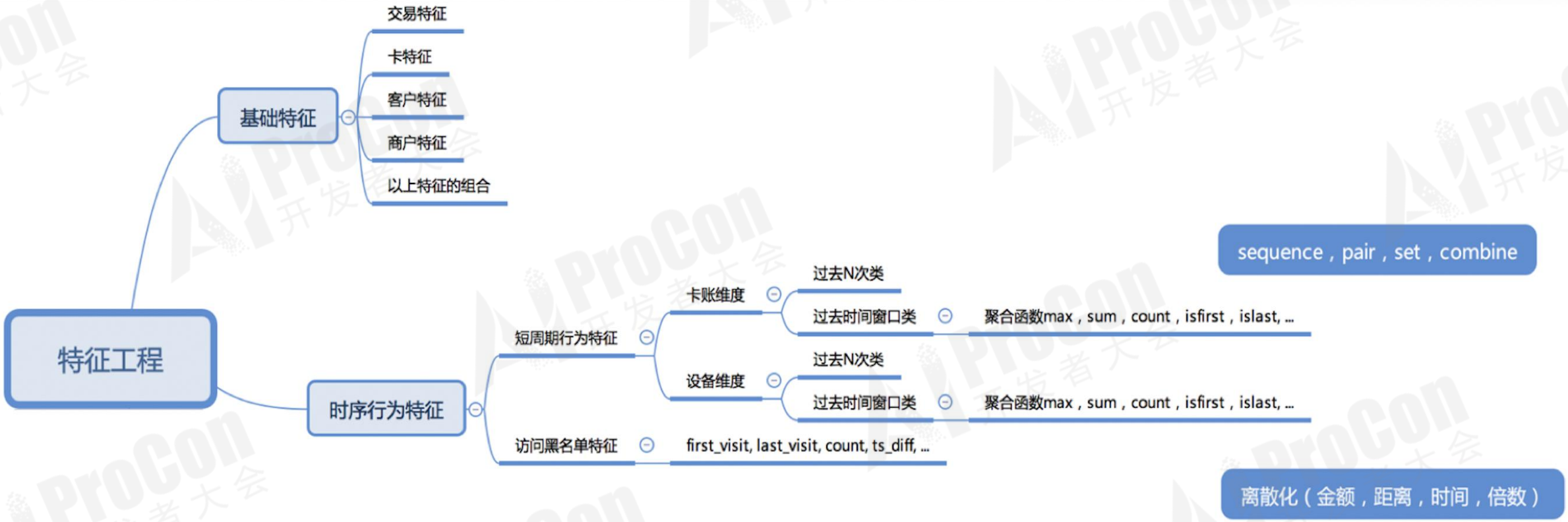
AI特征数据库实践：B2C Antifraud

定义窗口提取时窗特征（超过800组特征）：

```
# 卡号维度窗口，并且输出到card_output表
w_card = window(trx, "cardno", "t_time", 10000, 5d, "card_output")
# 设备号维度窗口，并且输出mac_output表
w_mac = window(trx, "mac", "t_time", 10000, 5d, "mac_output")
```

AI特征数据库实践：B2C Antifraud

- 5000万数据，19亿维，需要3T内存



AI特征数据库实践：B2C Antifraud

```
w_card = window(trx_flow, "card_no", "trx_time", 1000, 5d, "card_fea_output")
w_mac = window(trx_flow, "mac", "trx_time", 1000, 5d, "mac_fea_output")
log_ef_7 = column(log(w_card.Float[0]))
mapping_ef_9 = column(mapping(w_card.Float[0], "1.4=1"))
month_ef_6 = column(month(w_card.Date[0]))
weekday_ef_8 = column(dayofweek(w_mac.Date[0]))
weekend_ef_9 = column(case when(dayofweek(w_mac.Date[0]) in (6, 7)) then 1 else 0 end)
out_table1 = left_join(w_card, w_mac, "w_card.merchant = w_mac.merchant and w_card.ts >= w_mac.f5")
```

1. 特征抽取

```
X = np.array(data.drop(["hotel_cluster"], 1))
y = np.array(data['hotel_cluster'])
X_train, X_test, y_train ,y_test = model_selection.train_test_split(X,y,test_size=0.2)

clf = RandomForestClassifier(n_estimators=10)
clf.fit(X_train,y_train)
accuracy = clf.score(X_test, y_test)
print(accuracy)
```

2. 模型训练

```
for index,row in test.iterrows():
    sample = {
      "index": row[1],
      "date_time": row[2],
      "cnt": 0,
      "hotel_cluster": 0
    }
    print(sample)
    response = requests.post('http://localhost:5000/predict', json = json.loads(json.dumps(sample, default=util.numpyEncoder)))
    print(response.json())
```

4. 在线预估

```
print("Starting deploy...")
feql = util.read_file("model/hotel_fe.fql")
status, msg = client.deployFeQL(deploy_name, ns, feql, False, True, 2, 2)
if not status:
    print("deploy feql failed: " + msg)
    exit()
print("deploy feql success")

status, msg = client.showDeploy(ns)
if not status:
    print("show deploy failed: " + msg)
    exit()
print("show deploy success")
print(msg)
```

3. 模型部署

Conclusion

1. AI场景的计算存储一体化趋势介绍
2. AI全流程解决方案——AI特征数据库架构设计
3. AI特征数据库的特征计算和核心存储组件剖析
4. AI特征数据库快速构建AI应用及真实场景实践

欢迎关注第四范式公众号

简历投递邮箱：

talent@4paradigm.com

CSDN

AI ProCon 开发者大会

Thanks