openGauss AI特性

孙佶

Gauss松鼠会

openGauss

https://opengauss.org

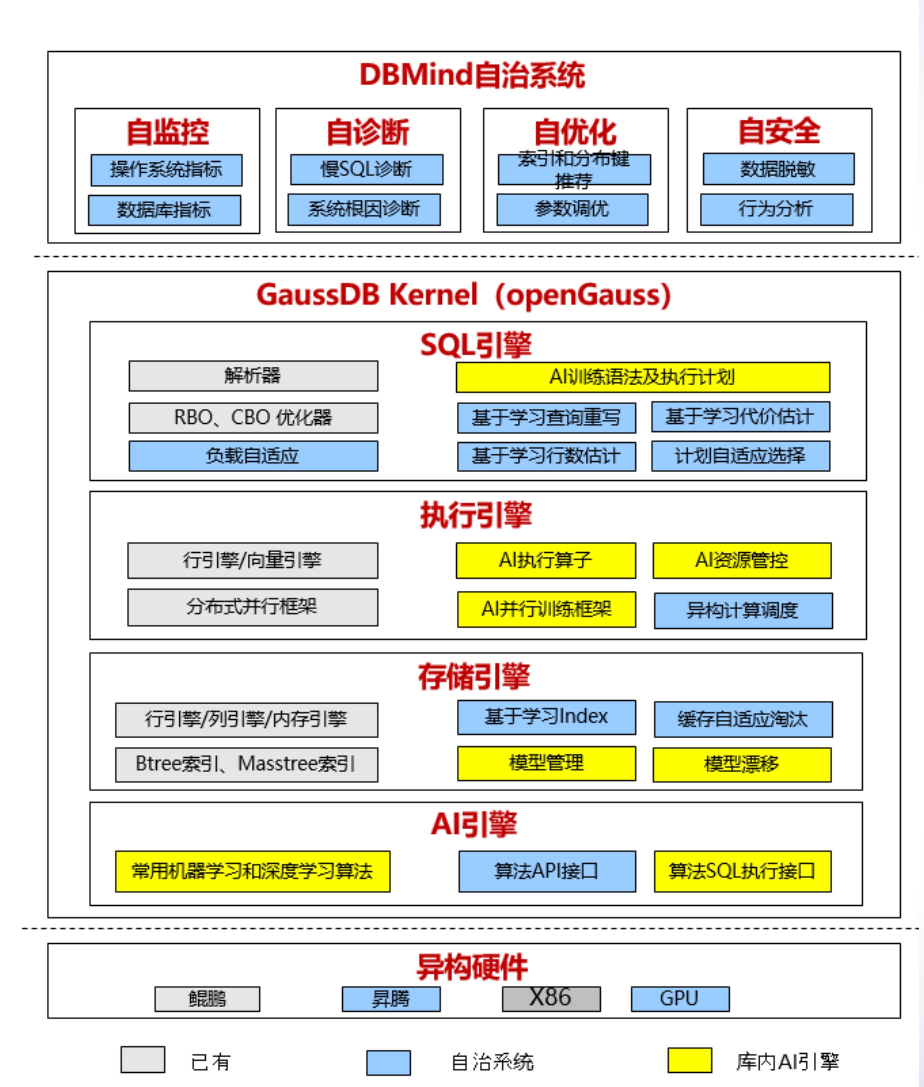
01 openGauss AI整体架构

02 openGauss 原生AI底座

03 openGauss AI自治运维

04 openGauss 智能内核

openGauss AI整体架构



DBMind自治系统

自监控: 操作系统指标, 数据库指标

自诊断: 慢SQL诊断, 系统根因诊断

自优化: 索引和分布键推荐, 参数调优

自安全: 数据脱敏, 行为分析

GaussDB Kernel（openGauss)

SQL引擎: 解析器, RBO、CBO 优化器, 负载自适应, AI训练语法及执行计划, 基于学习查询重写, 基于学习代价估计, 基于学习行数估计, 计划自适应选择

执行引擎: 行引擎/向量引擎, 分布式并行框架, AI执行算子, AI资源管控, AI并行训练框架, 异构计算调度

存储引擎: 行引擎/列引擎/内存引擎, Btree索引、Masstree索引, 基于学习Index, 缓存自适应淘汰, 模型管理, 模型漂移

AI引擎: 常用机器学习和深度学习算法, 算法API接口, 算法SQL执行接口

异构硬件: 鲲鹏, 昇腾, X86, GPU

□ 已有　🟦 自治系统　🟨 库内AI引擎

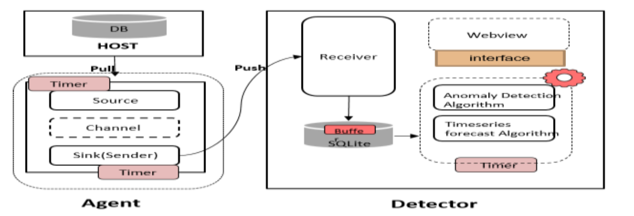
openGauss AI

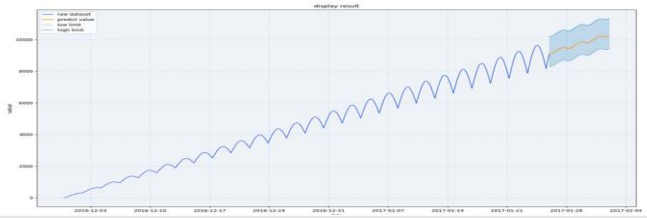
AI自治运维系统	✓ 自监控：采集OS、数据库、日志、SQL等多维指标，进行趋势预测、异常检测 ✓ 自诊断：慢SQL诊断、系统亚健康诊断、集群故障诊断 ✓ 自优化：索引智能推荐、分布键推荐、参数调优、SQL智能重写 ✓ 自安全：敏感信息发现、数据智能脱敏、行为异常分析
AI智能内核	✓ AI优化器：代价估计、行数估计、计划自适应选择 ✓ 存储引擎：自学习索引、缓存自适应淘汰 ✓ 系统调度：负载自适应调度
DB4AI库内AI引擎	✓ 训练和推理SQL语法、训练语句执行计划及代价 ✓ AI执行算子、并行训练框架、分布式训练 ✓ 模型管理、模型漂移、超参优化
	✓ 20+常用机器学习算法 ✓ 算法API接口，供学习型内核组件调用 ✓ 算法SQL执行接口，供AI执行算子调用

Gauss松鼠会　openGauss　https://opengauss.org

数据库自治：全面提升数据库智能化水平，应对不同行业多样化负载

1 自监控，异常检测





信息收集

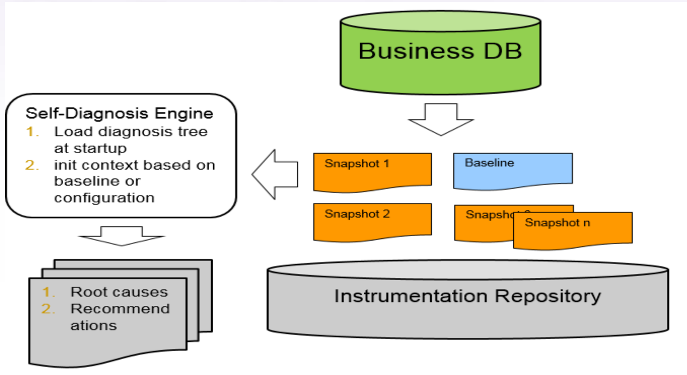
- **信息采集**：**OS**、数据库、慢**SQL**信息、安全等多维度信息采集，支持用户自定义采集策略

异常检测

- **趋势预测：**基于**AI**算法推测数据趋势，提前给出告警或者用于用户资源编排，支撑用户高效编排业务、智能负载调度
- **异常检测**：多维度信息异常检测，支持阈值或者趋势异常，进行时间推测及问题影响评估

2 自诊断自愈，Always Online

自愈自诊断架



Business DB
Self-Diagnosis Engine
1. Load diagnosis tree at startup
2. init context based on baseline or configuration
Snapshot 1
Baseline
Snapshot 2
Snapshot
Snapshot n
Root causes
Recommendations
Instrumentation Repository

自诊断

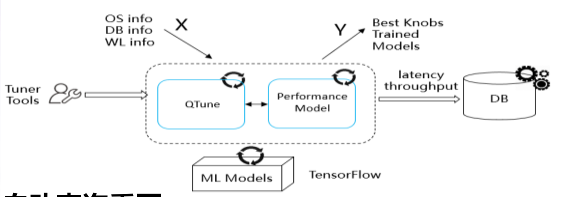
- **诊断引擎**：支持**AI**和**Rule**引擎结合的基于**Trace**智能诊断系统，推断问题根因
- **诊断模式库**：支持常见故障模式库，智能可扩展的模式库

自愈&故障预警

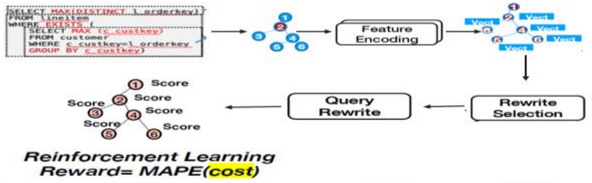
- **统计信息恢复：**根据系统查询数据**STAT**老化程度自动修复统计信息
- **故障预警：**预测软件&硬件故障，故障检测时间小于**5s**

3 自调优，持续高性能

参数自调优&推荐



自动查询重写



云化智能调优

- **参数自调优**：调优时间：天→分钟级；相比普通**DBA**调参，**DB**性能提升**30%**以上
- **索引推荐**：基于用户的单条语句或批量负载，推荐合适索引

云化在线调优

- **在线调参**：结合数据库负载预测模型动态调参，让数据库执行性能最大化
- **自动查询重写**：自动查询重写，利用最优序列的优化问题，基于蒙特卡洛树搜索算法实现语句重写

Gauss松鼠会

openGauss

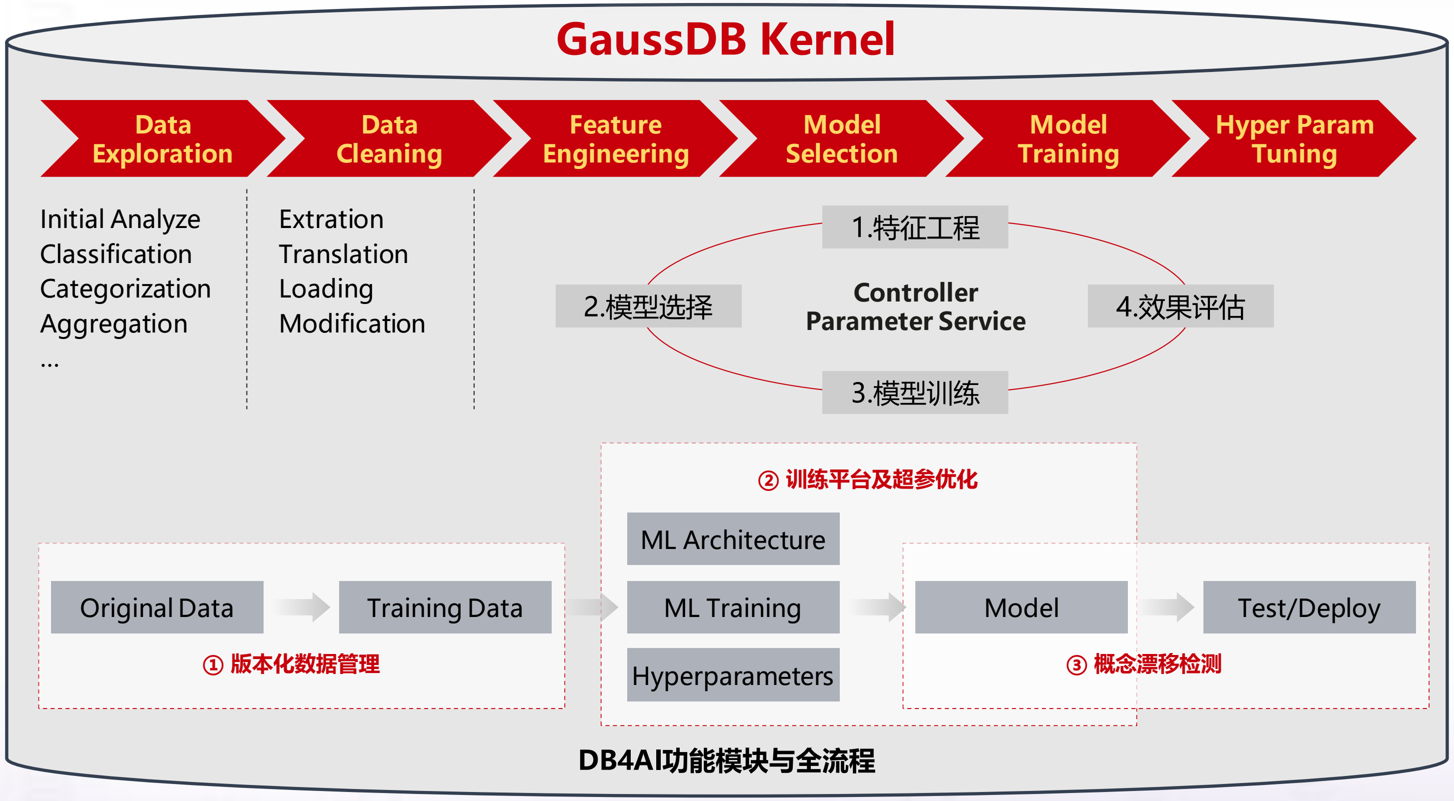
https://opengauss.org

全栈原生AI库内算子训练框架，一站式，会SQL就能用AI

```
CREATE MODEL price_model USING
logistic_regression
FEATURES size, lot
TARGET price
FROM HOUSES;
```

```
SELECT id, PREDICT
BY price_model (FEATURES size, lot), size
FROM houses;
```

原生SQL语法



GaussDB Kernel

Data Exploration → Data Cleaning → Feature Engineering → Model Selection → Model Training → Hyper Param Tuning

Initial Analyze
Classification
Categorization
Aggregation
…

Extration
Translation
Loading
Modification

1.特征工程
2.模型选择
3.模型训练
4.效果评估
Controller Parameter Service

② 训练平台及超参优化

ML Architecture
ML Training
Hyperparameters

Original Data → Training Data → ML Training → Model → Test/Deploy

① 版本化数据管理

③ 概念漂移检测

DB4AI功能模块与全流程

In-Database AI引擎

模型丰富

- 库内原生支持20多种常用AI算子，满足超过95%的机器学习使用场景

高性能

- 性能相比开源产品提升5-10倍

低门槛

- 极简SQL语法，易学习、易使用

安全高效

- 库内全流程AI框架，数据不出库，E2E完成数据清洗、特征工程、模型选择和模型训练，安全可靠、简单高效

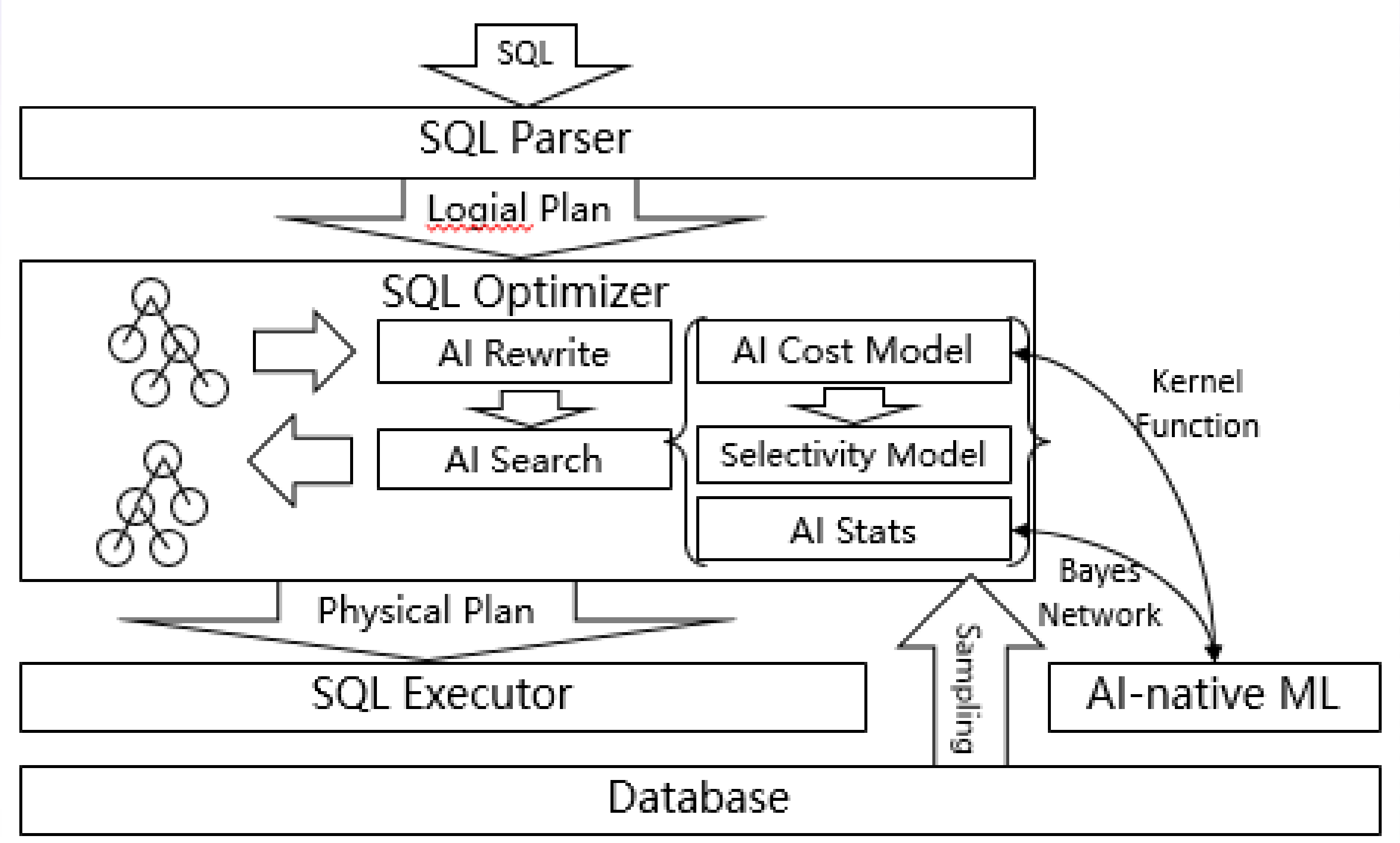
Gauss松鼠会

openGauss

https://opengauss.org

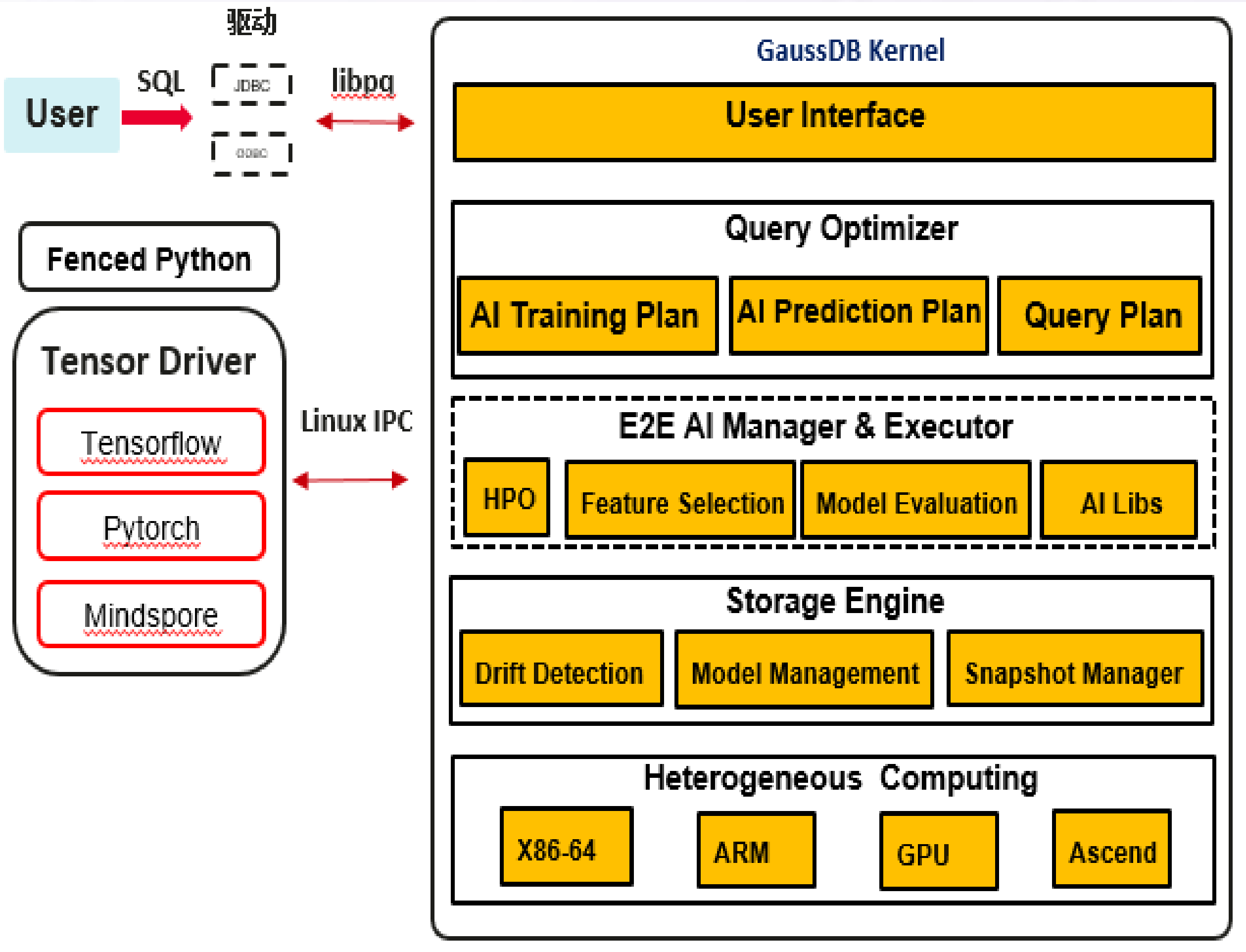
openGauss AI-based 优化器，实现性能倍增

传统优化器关键问题	
行数估算问题	1. 多列相关性问题 2. 表达式统计信息问题 3. 中间结果集统计信息问题 4. 数据分布问题
代价模型问题	1. 算子间代价失衡问题 2. 算子并行化代价问题 3. 分布式计划并行代价问题 4. 代价-时间换算问题 5. 跨平台泛化问题
路径搜索问题	1. 搜索空间与搜索代价的平衡（NP问题） 2. 局部最优解与全局最优解问题



- ➢ **解决传统优化器的行数估算、代价模型、路径搜索等经典问题。**
- ➢ **推动优化器朝智能化方向发展，实现自适应、自优化。**

openGauss 原生AI底座



驱动

User

SQL

JDBC

ODBC

libpq

Fenced Python

Tensor Driver

Tensorflow

Pytorch

Mindspore

Linux IPC

GaussDB Kernel

User Interface

Query Optimizer

AI Training Plan

AI Prediction Plan

Query Plan

E2E AI Manager & Executor

HPO

Feature Selection

Model Evaluation

AI Libs

Storage Engine

Drift Detection

Model Management

Snapshot Manager

Heterogeneous Computing

X86-64

ARM

GPU

Ascend

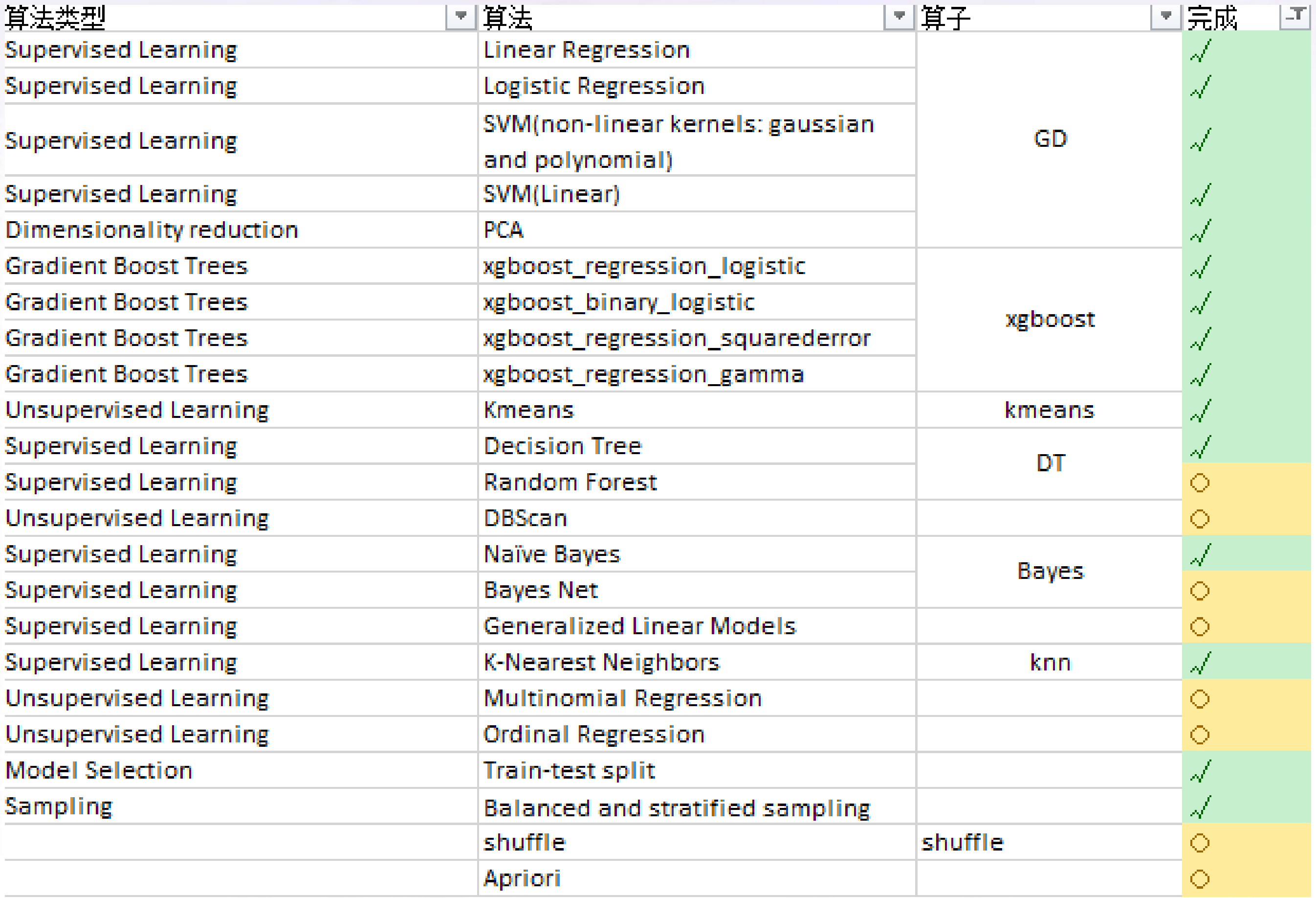
- **用户接口层**：实现SQL-like 语法，提供PREDICT、MODEL等关键字，支持模型的训练、预测以及管理等；
- **语句优化层**：实现业内首创的原生AI算子，优化器生成包含原生AI算子的执行计划；实现代价估计，支持通过EXPLAIN语句查看详细的执行开销，并提供可能的路径选择能力（ShuffleScan 下推）。
- **AI全流程管理**：支持对模型的管理、评估，支持模型的定期、定点更新与fine-tune等；支持AutoML能力，具有超参数调优（Hyper Parameter Optimization, HPO）能力；具备特征选择、特征处理、与数据清洗能力。
- **AI执行器**：负责执行AI执行计划，支持算法的并行执行。
- **存储层**：支持为数据创建快照并进行管理，负责模型文件的存储与组织。
- **异构计算能力**：支持多种计算平台，包括X86架构、ARM架构、以及具备GPU的环境。
- **AI底座**：提供数据库内部API，支持数据库内AI能力，如AI优化器、AI buffer、AI索引等。

Gauss松鼠会

openGauss

https://opengauss.org

openGauss AI算法



算法类型	算法	算子	完成
Supervised Learning	Linear Regression	GD	✓
Supervised Learning	Logistic Regression		✓
Supervised Learning	SVM(non-linear kernels: gaussian and polynomial)		✓
Supervised Learning	SVM(Linear)		✓
Dimensionality reduction	PCA		✓
Gradient Boost Trees	xgboost_regression_logistic	xgboost	✓
Gradient Boost Trees	xgboost_binary_logistic		✓
Gradient Boost Trees	xgboost_regression_squarederror		✓
Gradient Boost Trees	xgboost_regression_gamma		✓
Unsupervised Learning	Kmeans	kmeans	✓
Supervised Learning	Decision Tree	DT	✓
Supervised Learning	Random Forest		○
Unsupervised Learning	DBScan		○
Supervised Learning	Naïve Bayes	Bayes	✓
Supervised Learning	Bayes Net		○
Supervised Learning	Generalized Linear Models		○
Supervised Learning	K-Nearest Neighbors	knn	✓
Unsupervised Learning	Multinomial Regression		○
Unsupervised Learning	Ordinal Regression		○
Model Selection	Train-test split		✓
Sampling	Balanced and stratified sampling		✓
	shuffle	shuffle	○
	Apriori		○

Gauss松鼠会

openGauss

https://opengauss.org

openGauss SQL方式训练和推理的流程

任务	语法
训练	**CREATE MODEL** model_name **USING** architecture_name, [**FEATURES** {attribute_list}, **TARGET** attribute_name, [,attribute_name]*], **FROM** ([schema.]table_name | subquery) **WITH** (hyper_parameter_name [= {hp_value}]) [, ...]*]
推测	**PREDICT BY** model_name [(FEATURES attribute [, attribute] +])]
解释	**EXPLAIN MODEL** model
删除	**DROP MODEL** model

openGauss SQL方式训练和推理的流程

- **> explain CREATE MODEL patient_bayes USING naive_bayes FEATURES second_attack,treatment TARGET trait_anxiety > 50 FROM patients;**
- QUERY PLAN
- --
- Train Model - naive_bayes (cost=0.00..0.00 rows=0 width=0)
- -> Seq Scan on patients (cost=0.00..32.20 rows=1776 width=12)
- (2 rows)

- **> CREATE MODEL patient_bayes USING naive_bayes FEATURES second_attack,treatment TARGET trait_anxiety > 50 FROM patients;**
- MODEL CREATED. PROCESSED 1

- **> SELECT PREDICT BY patient_bayes (FEATURES second_attack,treatment) FROM patients LIMIT 3;**
- patient_bayes_pred
- --------------------
- f
- t
- t
- (3 rows)

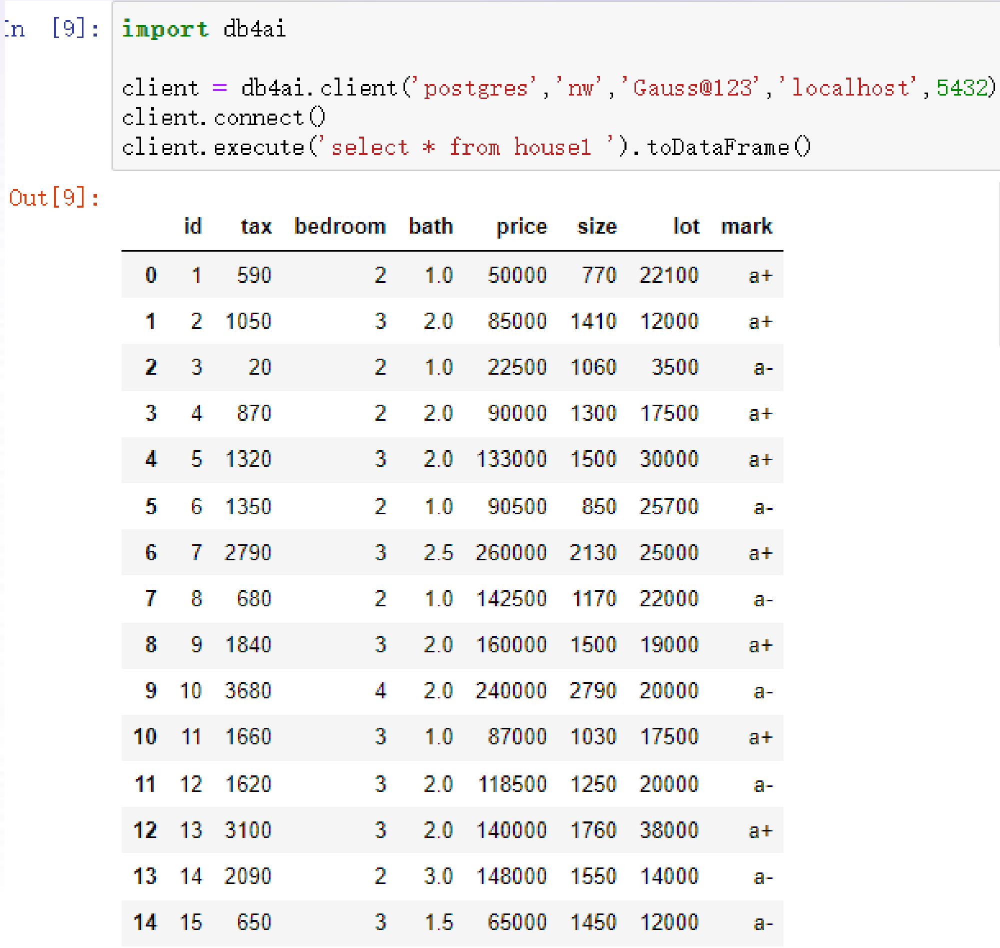
> EXPLAIN MODEL patient_bayes;

```
                                          DB4AI MODEL
--------------------------------------------------------------------------------------------
 Name: patient_bayes
 Algorithm: naive_bayes
 Query: CREATE MODEL patient_bayes USING naive_bayes FEATURES
second_attack,treatment TARGET trait_anxiety > 50 FR
OM patients;
 Return type: Boolean
 Pre-processing time: 0.000000
 Execution time: 0.000186
 Processed tuples: 22
 Discarded tuples: 2
 prob:
{"{.590909090909091,.409090909090909}","{.384615384615385,.61538461538461
5,.777777777777778,.22222222222222
2}","{.307692307692308,.692307692307692,.666666666666667,.333333333333333
}"}
 features: {"{true,false}","{0,1}","{1,0}"}
(11 rows)
```

SQL示例

openGauss SQL方式训练和推理的流程

- 使用者可以直接使用Python版db4ai库，借助本库和Pandas的集成，结合声明性SQL和命令式Python代码，以执行数据分析、可视化等任务。

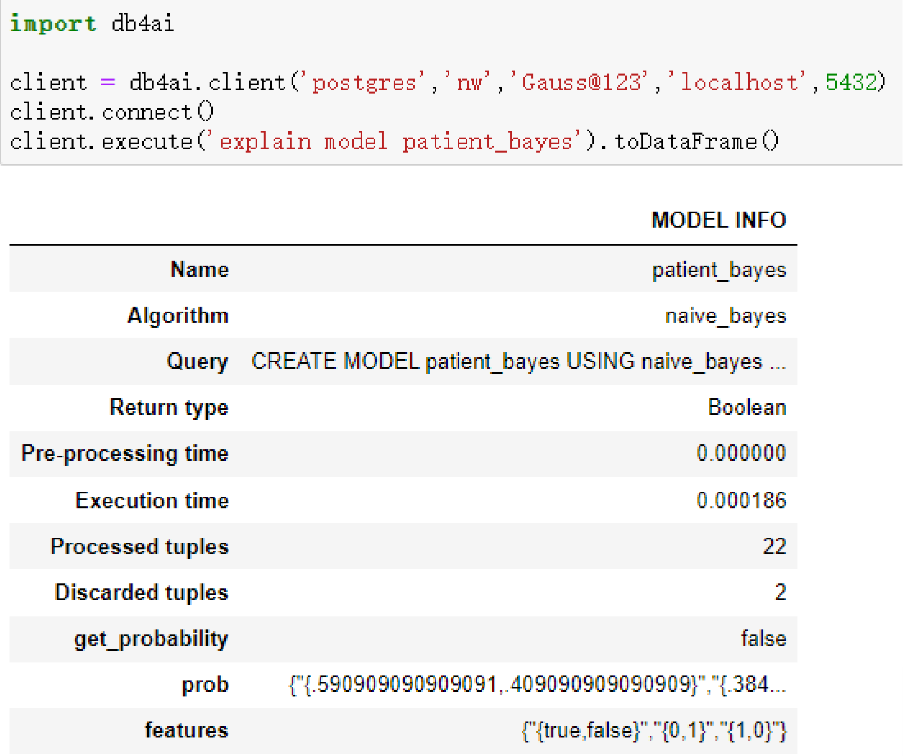


In [9]:
```
import db4ai

client = db4ai.client('postgres','nw','Gauss@123','localhost',5432)
client.connect()
client.execute('select * from house1 ').toDataFrame()
```

Out[9]:

	id	tax	bedroom	bath	price	size	lot	mark
0	1	590	2	1.0	50000	770	22100	a+
1	2	1050	3	2.0	85000	1410	12000	a+
2	3	20	2	1.0	22500	1060	3500	a-
3	4	870	2	2.0	90000	1300	17500	a+
4	5	1320	3	2.0	133000	1500	30000	a+
5	6	1350	2	1.0	90500	850	25700	a-
6	7	2790	3	2.5	260000	2130	25000	a+
7	8	680	2	1.0	142500	1170	22000	a-
8	9	1840	3	2.0	160000	1500	19000	a+
9	10	3680	4	2.0	240000	2790	20000	a-
10	11	1660	3	1.0	87000	1030	17500	a+
11	12	1620	3	2.0	118500	1250	20000	a-
12	13	3100	3	2.0	140000	1760	38000	a+
13	14	2090	2	3.0	148000	1550	14000	a-
14	15	650	3	1.5	65000	1450	12000	a-



```
import db4ai

client = db4ai.client('postgres','nw','Gauss@123','localhost',5432)
client.connect()
client.execute('explain model patient_bayes').toDataFrame()
```

	MODEL INFO
Name	patient_bayes
Algorithm	naive_bayes
Query	CREATE MODEL patient_bayes USING naive_bayes ...
Return type	Boolean
Pre-processing time	0.000000
Execution time	0.000186
Processed tuples	22
Discarded tuples	2
get_probability	false
prob	{"{.590909090909091,.409090909090909}","{.384...
features	{"{true,false}","{0,1}","{1,0}"}

Gauss松鼠会

openGauss

https://opengauss.org

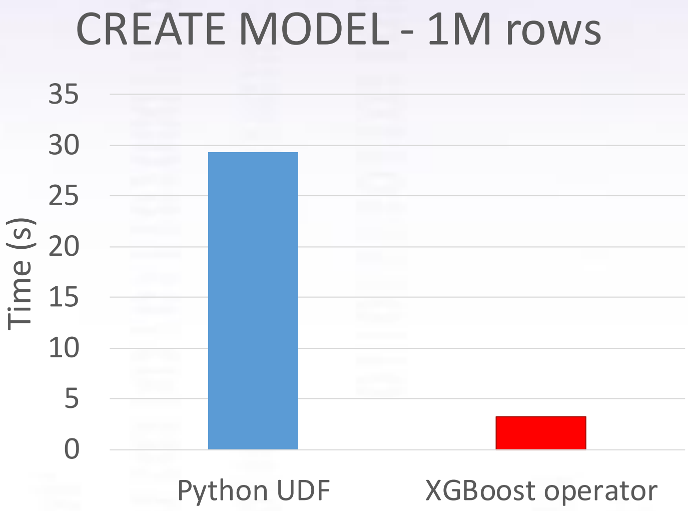
openGauss 性能测试(MADlib VS openGauss XGboost)

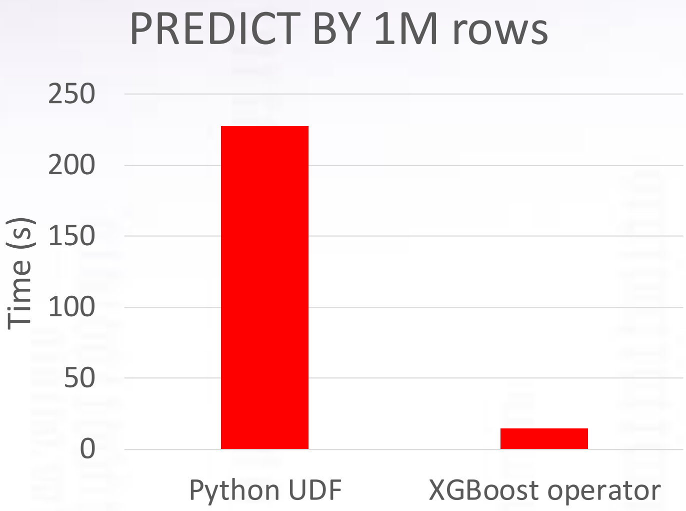
结果分析：

1、数据读取(批次读取)

2、MADlib在计算过程中存在大量的数据转写环节

3、DB4AI存在指令集加速:(option)





Results for dataset with 1m rows:

	Classification (s)	Prediction (s)
Python UDF	29.29	227.36
In-operator	3.18	14.46
KPI Speedup	**9.2x**	**15.7x**

Results for dataset with 10m rows:

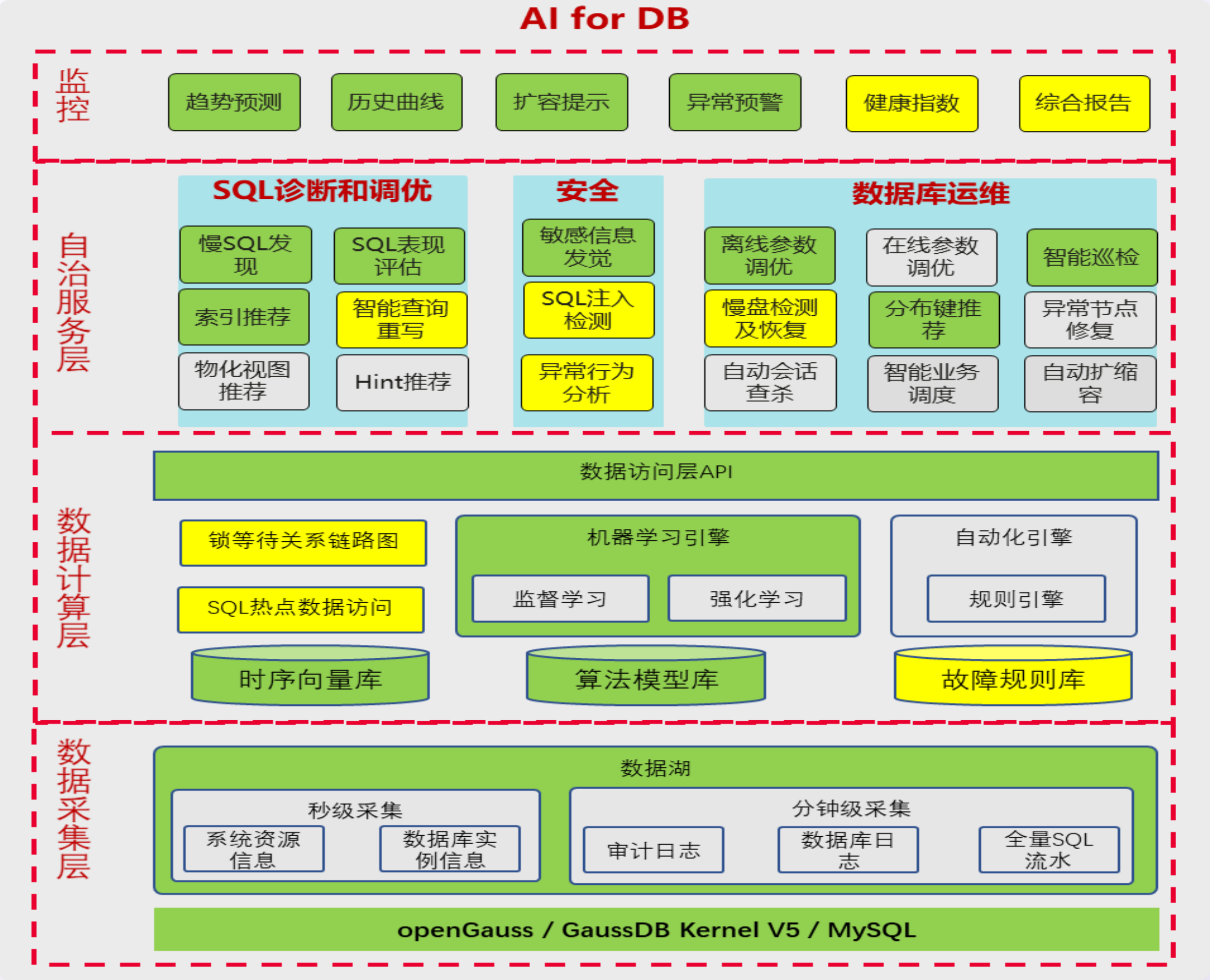
	Classification (s)	Prediction (s)
Python UDF	N/A (Error)	N/A (Error)
In-operator	43.0	204.7

xgboost
with hyperparams: “n_iter=10, max_depth=5, min_child_weight=1, nthread=10,
eta=0.1, booster='gbtree'”

Gauss松鼠会

https://opengauss.org

openGauss DBMind自治运维平台



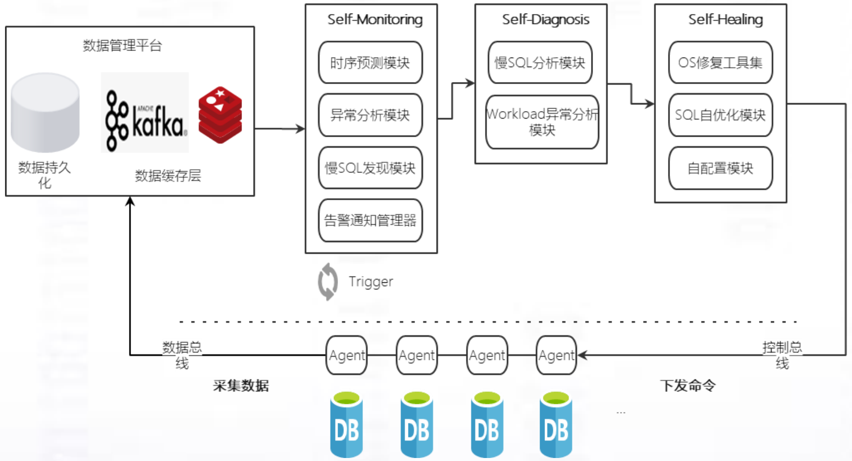
Agent： 从DB上采集数据、负责执行AI Service下发的命令

数据管理平台： 负责数据的缓存、持久化，包括时序数据、日志数据、SQL流水等

AI Service: 包括自监控、自诊断、自愈三个部分，利用机器学习（深度学习）算法、故障模式库等进行问题的发现、分析与解决

数据总线： 用于将从DB上收集的数据传送到数据管理平台

控制总线： 用于下发AI Service的运维命令



Gauss松鼠会

openGauss

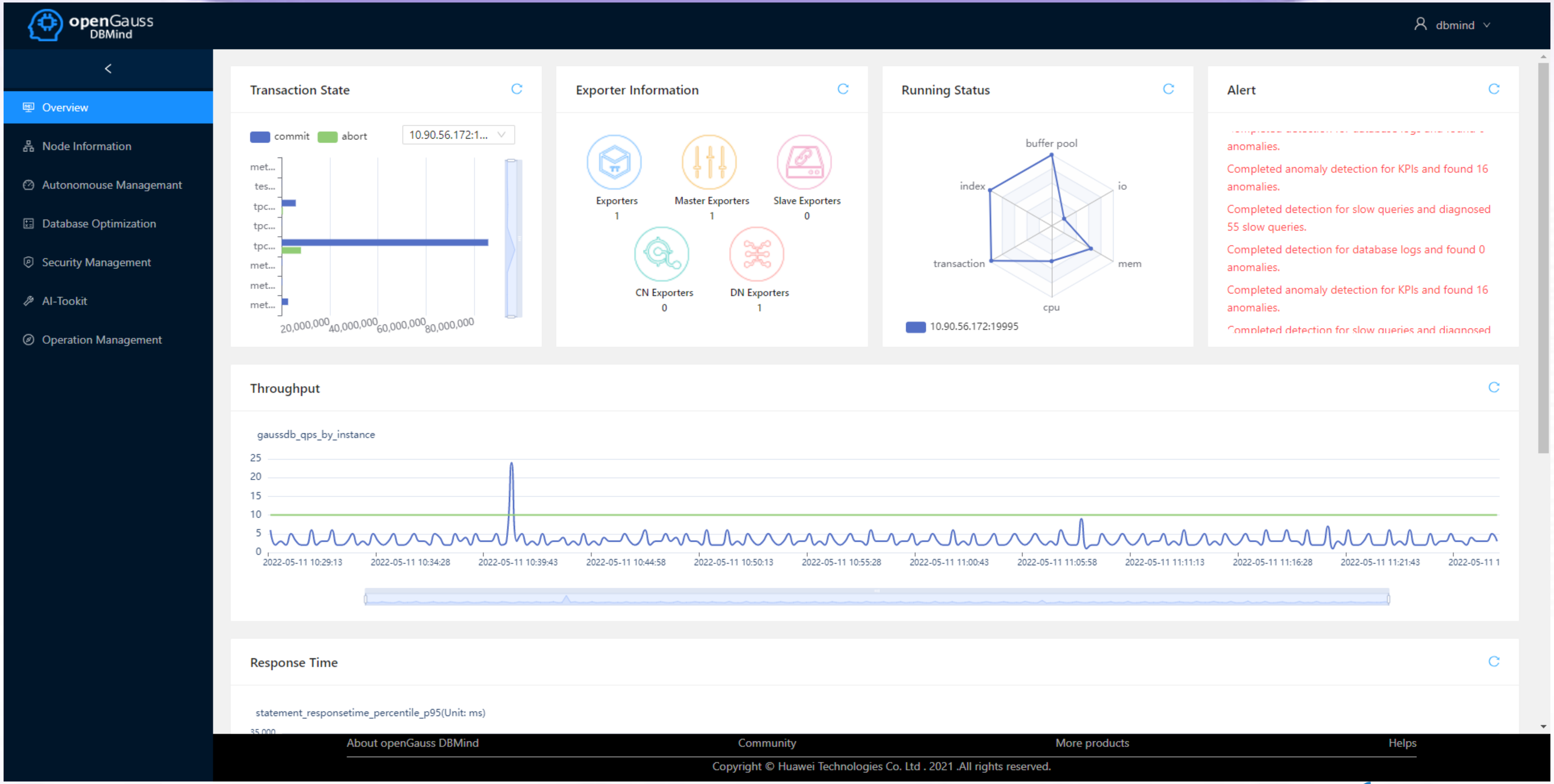
https://opengauss.org

已发布AI自治功能

场景	功能	算法
分布式	分布键推荐	最大图匹配、模板化
单机主备	慢SQL根因诊断	KNN
ALL	系统亚健康根因诊断	聚类、决策树
	集群故障根因诊断	知识图谱、决策树
	参数调优	强化学习、全局搜索（PSO）
	索引推荐	启发式算法
	异常检测	差分、相位图

```
./gs_dbmind service -c config --only-run help
usage:  service [-h] -c DIRECTORY [--only-run
{slow_query_diagnosis,forecast,anomaly_detection,alarm_log_diagnosis,index_recommendation,knob_recommendation}]
[--dry-run] [-f] [--interactive | --initialize] {setup,start,stop,restart}
 service: error: argument --only-run: invalid choice: 'help' (choose from 'slow_query_diagnosis', 'forecast',
'anomaly_detection', 'alarm_log_diagnosis', 'index_recommendation', 'knob_recommendation')
```

自监控-关键指标监控



anomalies.

Completed anomaly detection for KPIs and found 16 anomalies.

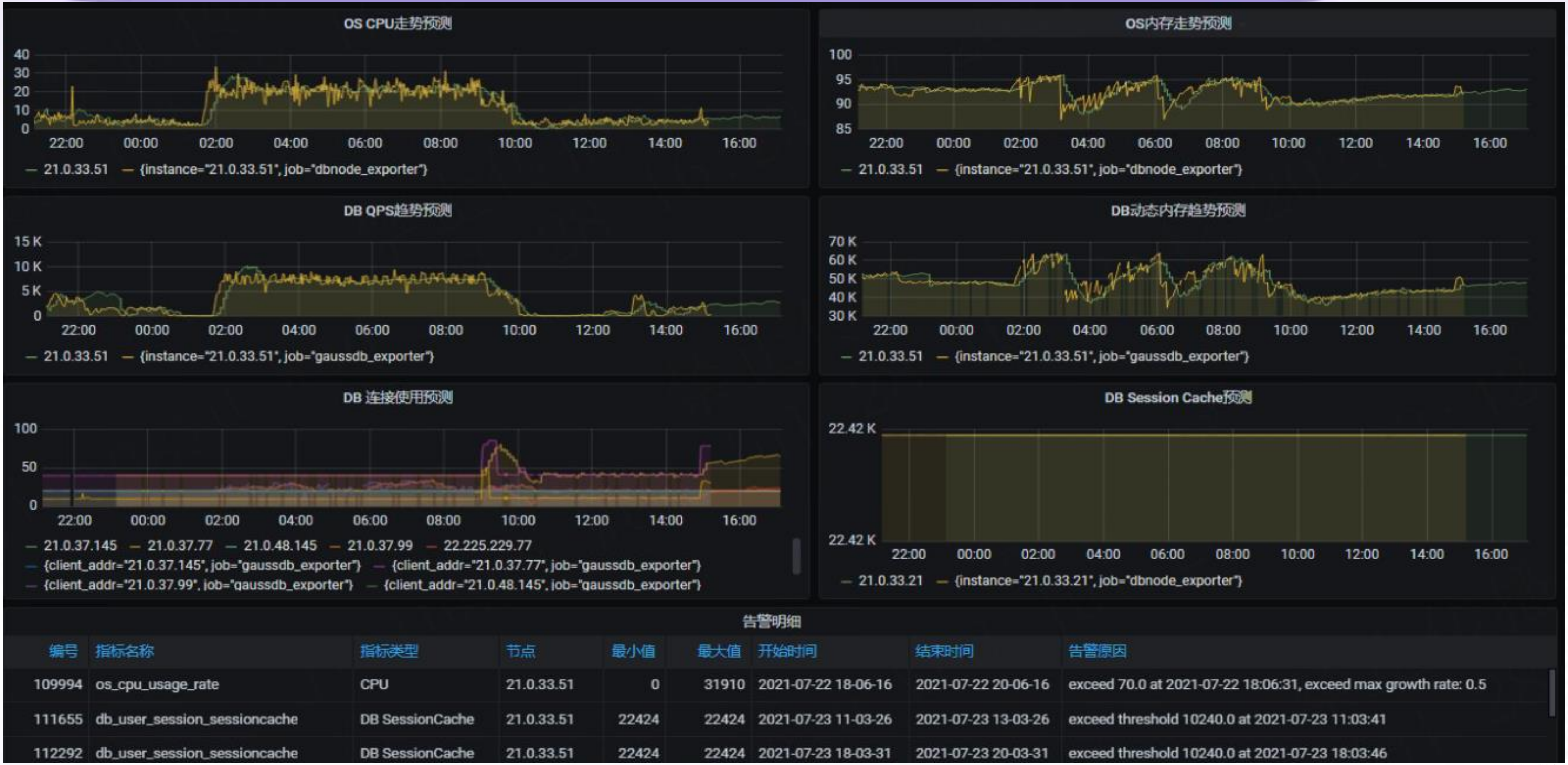
Completed detection for slow queries and diagnosed 55 slow queries.

Completed detection for database logs and found 0 anomalies.

Completed anomaly detection for KPIs and found 16 anomalies.

Completed detection for slow queries and diagnosed

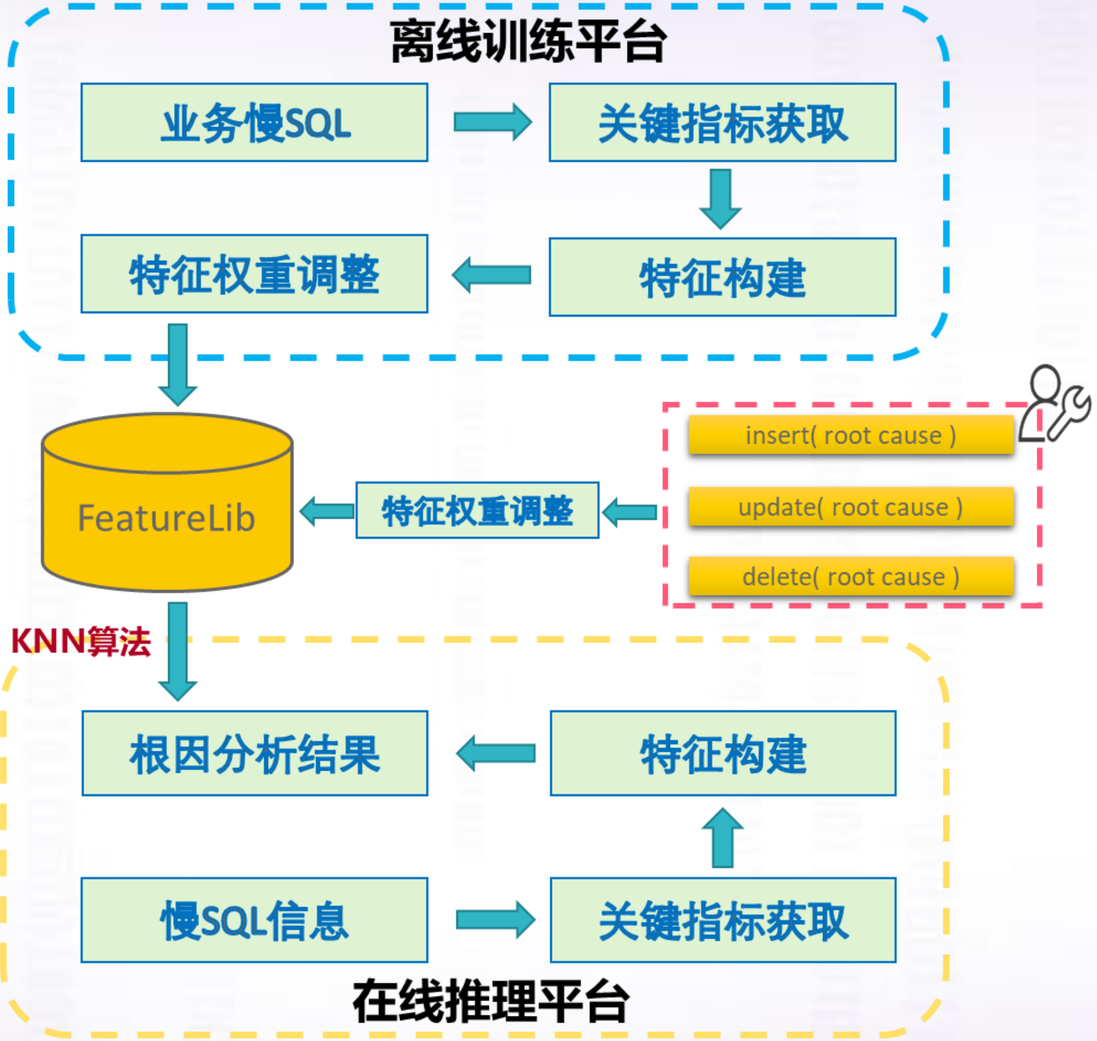
自监控-趋势预测及告警



告警明细

编号	指标名称	指标类型	节点	最小值	最大值	开始时间	结束时间	告警原因
109994	os_cpu_usage_rate	CPU	21.0.33.51	0	31910	2021-07-22 18-06-16	2021-07-22 20-06-16	exceed 70.0 at 2021-07-22 18:06:31, exceed max growth rate: 0.5
111655	db_user_session_sessioncache	DB SessionCache	21.0.33.51	22424	22424	2021-07-23 11-03-26	2021-07-23 13-03-26	exceed threshold 10240.0 at 2021-07-23 11:03:41
112292	db_user_session_sessioncache	DB SessionCache	21.0.33.51	22424	22424	2021-07-23 18-03-31	2021-07-23 20-03-31	exceed threshold 10240.0 at 2021-07-23 18:03:46

慢SQL诊断和建议



- **业务慢SQL来源：**

  慢SQL数据从openGauss系统视图dbe_perf.statement_history，该视图包含慢SQL文本、开始和结束执行时间、行数等相关信息。

- **慢SQL关键指标来源：**

  慢SQL关键指标指能表征慢SQL执行状态的指标，其可分为数据库相关和系统相关，数据库相关指标从系统表和系统视图中获取，例如返回行数、索引信息等；系统相关指标则直接从当前系统中获取，例如CPU USAGE、IOWAIT等。

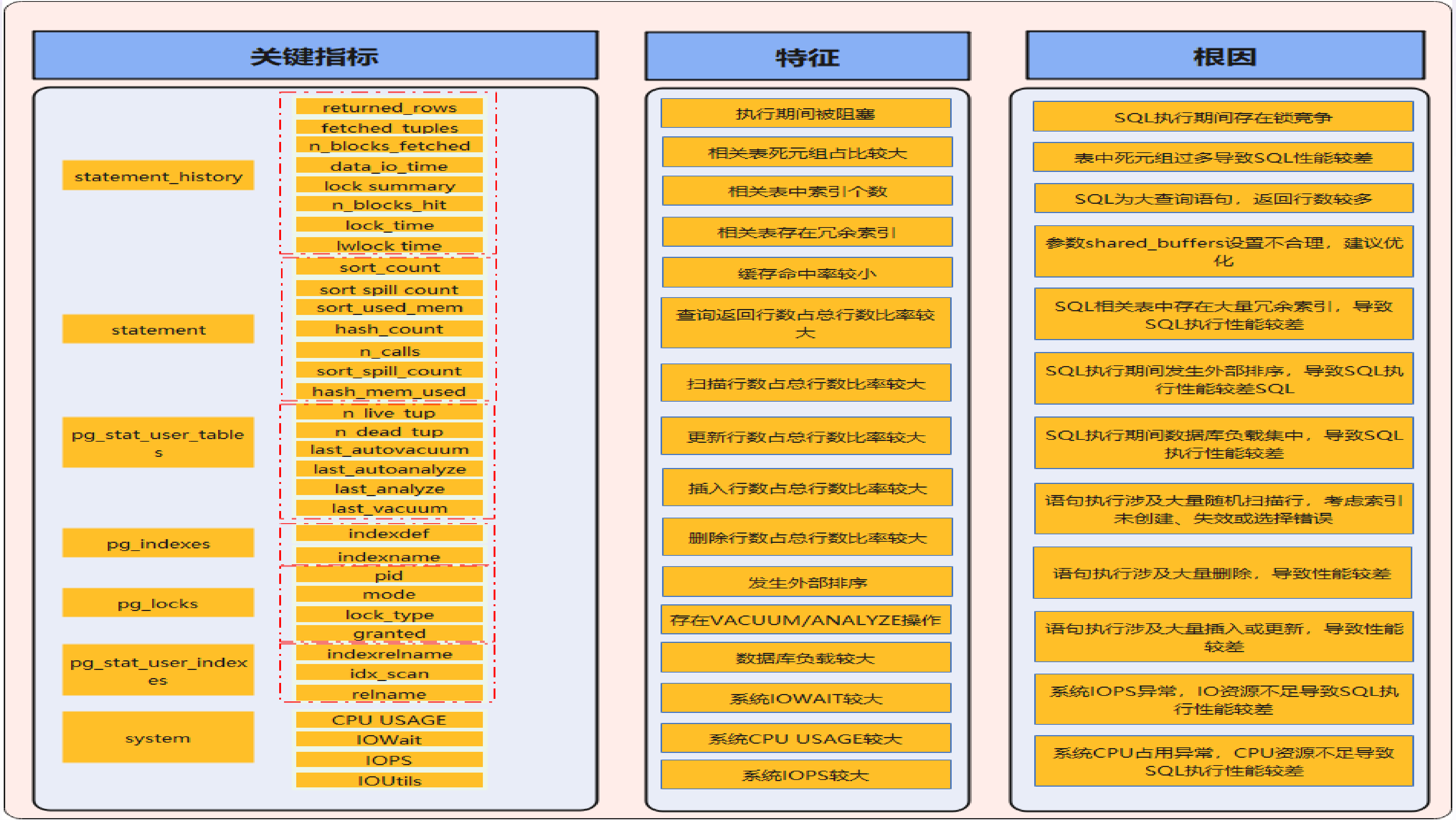
- **特征构建:**

  特征构建主要指在上述指标的基础上，构建分析慢SQL根因所需的特征，例如通过SQL执行的返回行数和总行数，判断慢SQL是否具有大查询的特征，这样可以将每个异常构建成对应的特征向量。

- **特征权重调整：**

  诊断框架使用KNN进行根因定位，即找寻与当前慢SQL特征向量最相似的topK个根因，再基于多数表决的方式确定根因类别。

慢SQL诊断和建议



关键指标

- statement_history
  - returned_rows
  - fetched tuples
  - n_blocks_fetched
  - data_io_time
  - lock summary
  - n_blocks_hit
  - lock_time
  - lwlock time
- statement
  - sort_count
  - sort spill count
  - sort_used_mem
  - hash_count
  - n_calls
  - sort_spill_count
  - hash_mem_used
- pg_stat_user_tables
  - n live tup
  - n dead tup
  - last_autovacuum
  - last_autoanalyze
  - last_analyze
  - last_vacuum
- pg_indexes
  - indexdef
  - indexname
- pg_locks
  - pid
  - mode
  - lock_type
  - granted
- pg_stat_user_indexes
  - indexrelname
  - idx_scan
  - relname
- system
  - CPU USAGE
  - IOWait
  - IOPS
  - IOUtils

特征

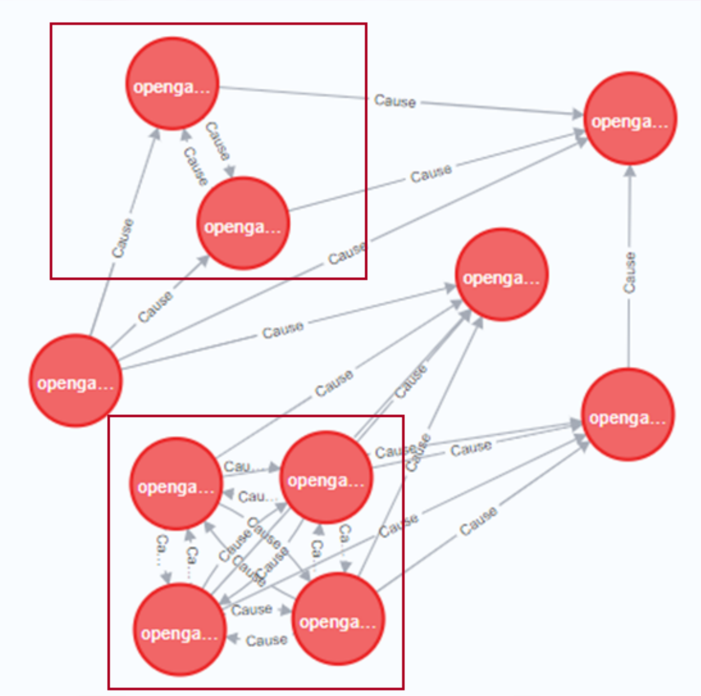
- 执行期间被阻塞
- 相关表死元组占比较大
- 相关表中索引个数
- 相关表存在冗余索引
- 缓存命中率较小
- 查询返回行数占总行数比率较大
- 扫描行数占总行数比率较大
- 更新行数占总行数比率较大
- 插入行数占总行数比率较大
- 删除行数占总行数比率较大
- 发生外部排序
- 存在VACUUM/ANALYZE操作
- 数据库负载较大
- 系统IOWAIT较大
- 系统CPU USAGE较大
- 系统IOPS较大

根因

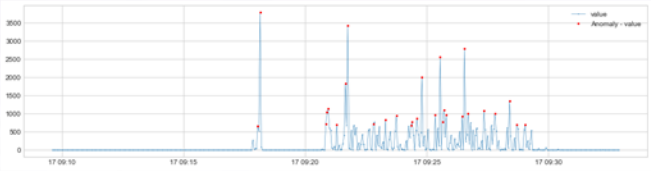
- SQL执行期间存在锁竞争
- 表中死元组过多导致SQL性能较差
- SQL为大查询语句，返回行数较多
- 参数shared_buffers设置不合理，建议优化
- SQL相关表中存在大量冗余索引，导致SQL执行性能较差
- SQL执行期间发生外部排序，导致SQL执行性能较差SQL
- SQL执行期间数据库负载集中，导致SQL执行性能较差
- 语句执行涉及大量随机扫描行，考虑索引未创建、失效或选择错误
- 语句执行涉及大量删除，导致性能较差
- 语句执行涉及大量插入或更新，导致性能较差
- 系统IOPS异常，IO资源不足导致SQL执行性能较差
- 系统CPU占用异常，CPU资源不足导致SQL执行性能较差

系统亚健康诊断

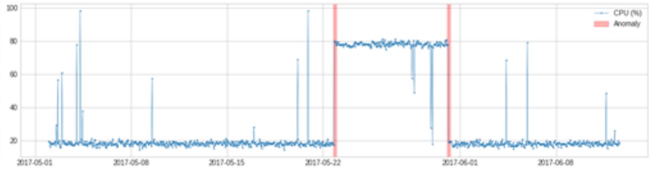
监控指标关联分析（聚类）



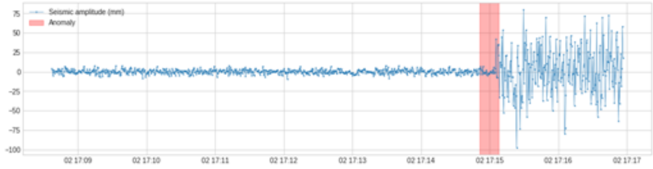
监控指标异常检测



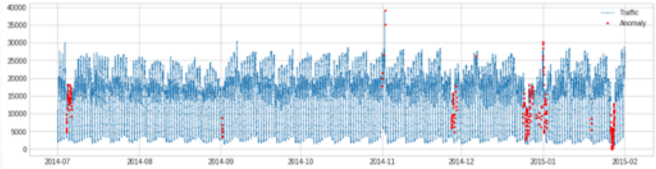
Spike



Level Shift

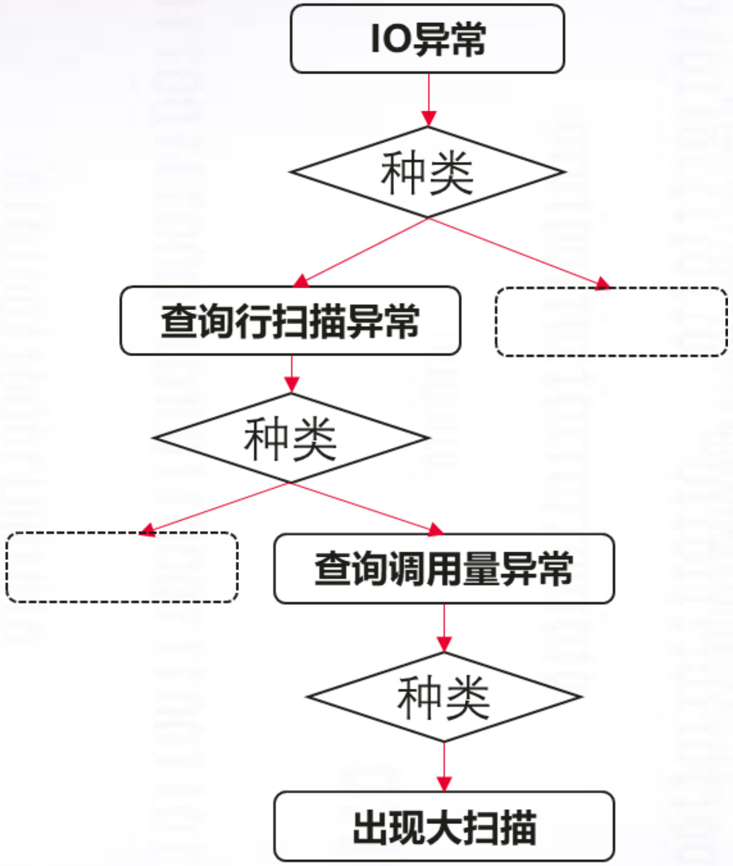


Volatility Shift



Seasonal Violation

根因诊断路径学习



IO异常

种类

查询行扫描异常

种类

查询调用量异常

种类

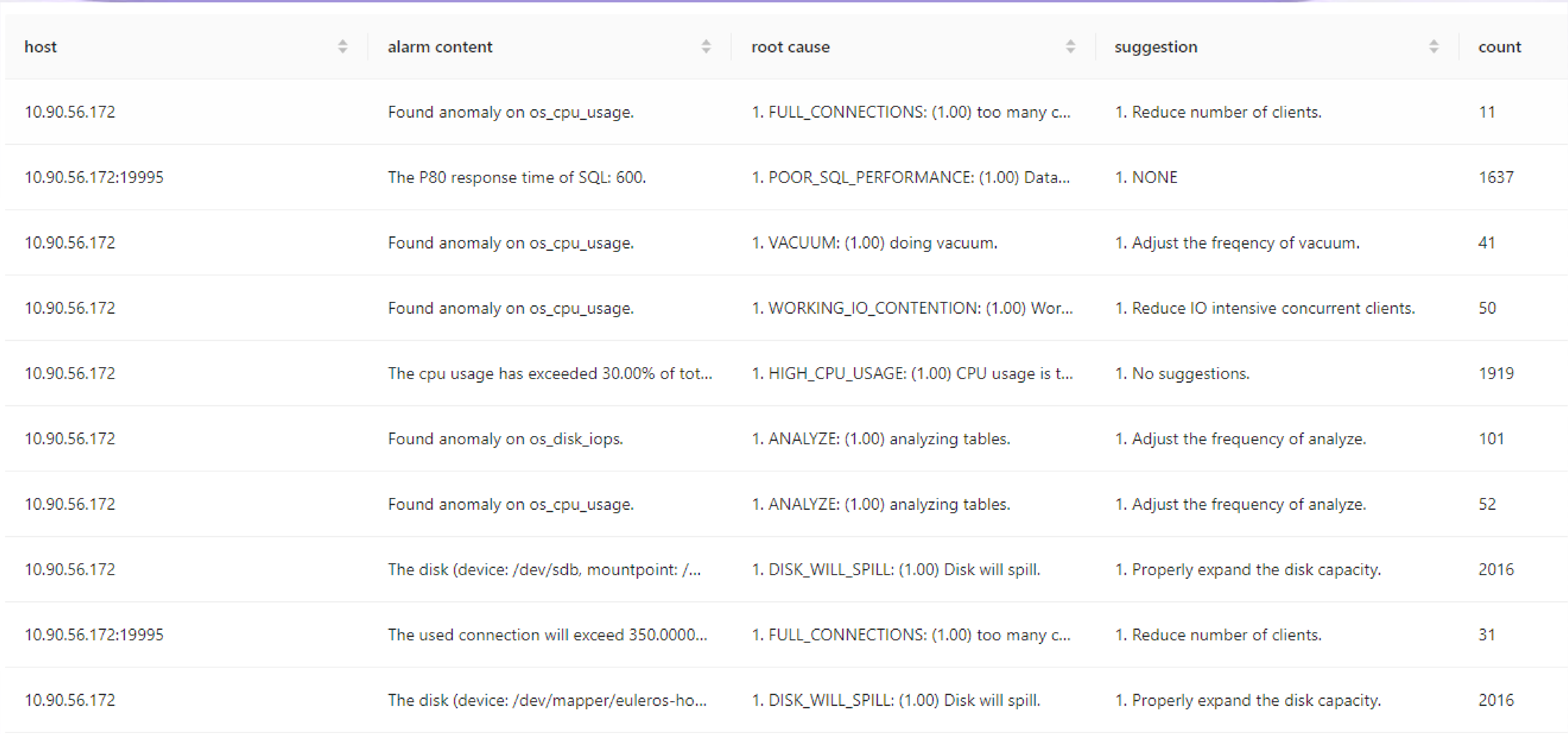
出现大扫描

Gauss松鼠会

openGauss

https://opengauss.org

系统亚健康诊断

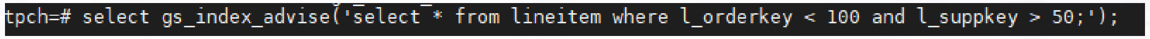


host	alarm content	root cause	suggestion	count
10.90.56.172	Found anomaly on os_cpu_usage.	1. FULL_CONNECTIONS: (1.00) too many c...	1. Reduce number of clients.	11
10.90.56.172:19995	The P80 response time of SQL: 600.	1. POOR_SQL_PERFORMANCE: (1.00) Data...	1. NONE	1637
10.90.56.172	Found anomaly on os_cpu_usage.	1. VACUUM: (1.00) doing vacuum.	1. Adjust the freqency of vacuum.	41
10.90.56.172	Found anomaly on os_cpu_usage.	1. WORKING_IO_CONTENTION: (1.00) Wor...	1. Reduce IO intensive concurrent clients.	50
10.90.56.172	The cpu usage has exceeded 30.00% of tot...	1. HIGH_CPU_USAGE: (1.00) CPU usage is t...	1. No suggestions.	1919
10.90.56.172	Found anomaly on os_disk_iops.	1. ANALYZE: (1.00) analyzing tables.	1. Adjust the frequency of analyze.	101
10.90.56.172	Found anomaly on os_cpu_usage.	1. ANALYZE: (1.00) analyzing tables.	1. Adjust the frequency of analyze.	52
10.90.56.172	The disk (device: /dev/sdb, mountpoint: /...	1. DISK_WILL_SPILL: (1.00) Disk will spill.	1. Properly expand the disk capacity.	2016
10.90.56.172:19995	The used connection will exceed 350.0000...	1. FULL_CONNECTIONS: (1.00) too many c...	1. Reduce number of clients.	31
10.90.56.172	The disk (device: /dev/mapper/euleros-ho...	1. DISK_WILL_SPILL: (1.00) Disk will spill.	1. Properly expand the disk capacity.	2016

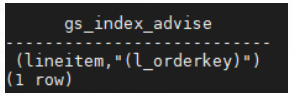
智能索引推荐

- 使用场景介绍：
- 根据用户的Workload整体信息，为用户推荐需要创建的索引。

使用示例：使用**gs_index_advise()**推荐索引，相比无索引情况下，性能提高**10000**倍！

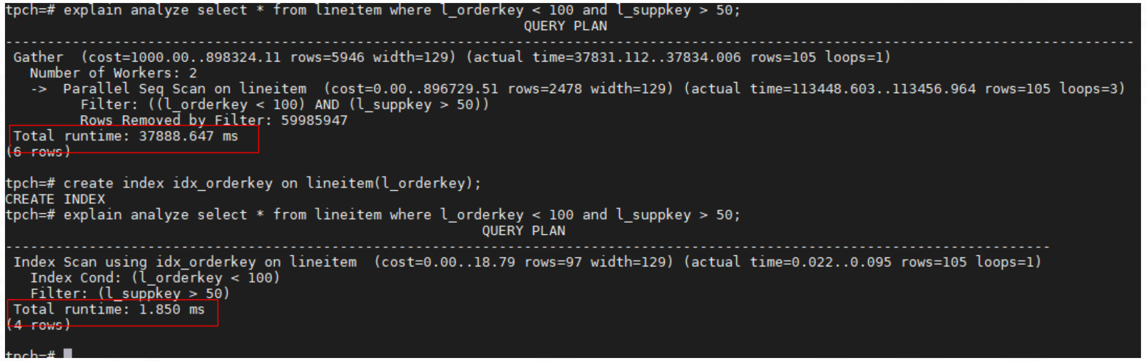


```
tpch=# select gs_index_advise('select * from lineitem where l_orderkey < 100 and l_suppkey > 50;');
```



```
      gs_index_advise
----------------------------
 (lineitem,"(l_orderkey)")
(1 row)
```

← 建议在**lineitem**表的**l_orderkey**列上创建索引。



```
tpch=# explain analyze select * from lineitem where l_orderkey < 100 and l_suppkey > 50;
                                                                 QUERY PLAN
-------------------------------------------------------------------------------------------------------------------------------------
 Gather  (cost=1000.00..898324.11 rows=5946 width=129) (actual time=37831.112..37834.006 rows=105 loops=1)
   Number of Workers: 2
   ->  Parallel Seq Scan on lineitem  (cost=0.00..896729.51 rows=2478 width=129) (actual time=113448.603..113456.964 rows=105 loops=3)
         Filter: ((l_orderkey < 100) AND (l_suppkey > 50))
         Rows Removed by Filter: 59985947
 Total runtime: 37888.647 ms
(6 rows)

tpch=# create index idx_orderkey on lineitem(l_orderkey);
CREATE INDEX
tpch=# explain analyze select * from lineitem where l_orderkey < 100 and l_suppkey > 50;
                                                          QUERY PLAN
-----------------------------------------------------------------------------------------------------------------------------
 Index Scan using idx_orderkey on lineitem  (cost=0.00..18.79 rows=97 width=129) (actual time=0.022..0.095 rows=105 loops=1)
   Index Cond: (l_orderkey < 100)
   Filter: (l_suppkey > 50)
 Total runtime: 1.850 ms
(4 rows)

tpch=# 
```

对索引推荐效果的整体评估：

TPC-DS/TPC-H: 部分语句均有不同程度的执行时间缩短；
TPC-C: 与原生索引基本持平，比无索引有巨大性能提升。

	无索引	原索引	算法一	算法二
tpmC	3.22	135.64	134.27	134.49
tpmTOTAL	8.06	299.64	299.24	298.69
Transaction Count	120.67	4495.00	4489.00	4480.67

TPC-C

端到端索引推荐的原理细节

单Query索引推荐的核心方法：

采用索引设计和优化的相关理论，基于原生的词法和语法解析，对查询语句中的子句和谓词进行分析和处理，再结合字段选择度、聚合条件、多表Join关系等输出最终建议。

索引性能验证的方法：

通过修改优化器相应的数据结构，利用**优化器评估**，进而判断创建该索引后，对优化器生成执行计划的影响。该过程可以不用真正创建索引，即业内所谓的"**假设索引**"，业内也多采用此种方法。

Workload级别索引推荐的核心方法：

通过用户输入（或自主采集）得到的workload信息，根据预设模型，进一步评估创建索引**对整体Workload的影响**，从而从候选索引中筛选出若核心索引。该过程中通过对SQL语句进行模板化，将将具体的value值用统一的占位符替代，同时采用水库抽样的方法采样和保留部分value的值。通过特征过程，对SQL中可能作为索引的列名进行提取作为特征，并进行向量化聚类。最终提取把生成的聚类中心中的语句作为压缩后的工作负载，进行索引推荐。

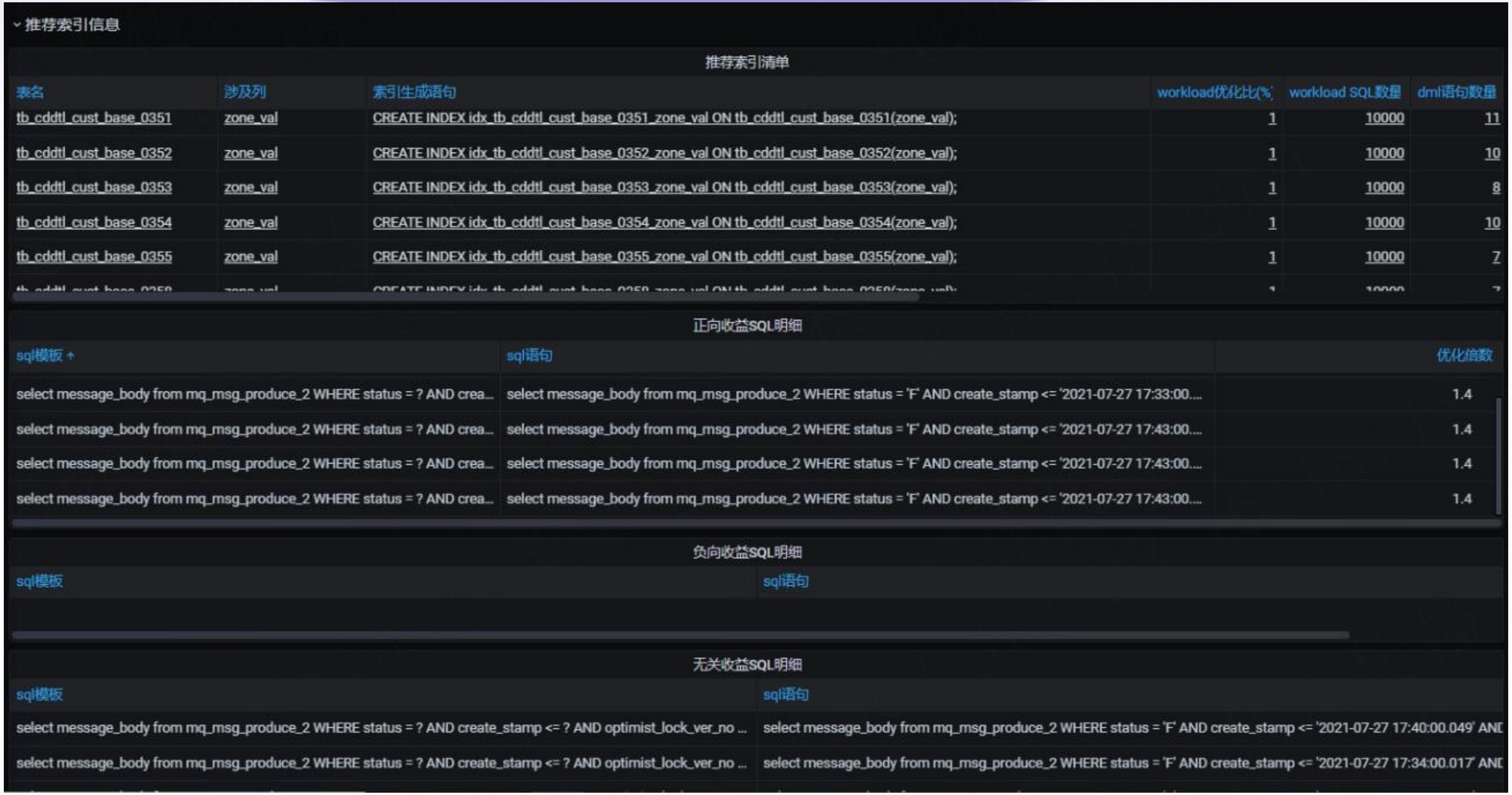
索引推荐
演练视频

Gauss松鼠会

openGauss

https://opengauss.org

智能索引推荐



推荐索引信息

推荐索引清单

表名	涉及列	索引生成语句	workload优化比(%)	workload SQL数量	dml语句数量
tb_cddtl_cust_base_0351	zone_val	CREATE INDEX idx_tb_cddtl_cust_base_0351_zone_val ON tb_cddtl_cust_base_0351(zone_val);	1	10000	11
tb_cddtl_cust_base_0352	zone_val	CREATE INDEX idx_tb_cddtl_cust_base_0352_zone_val ON tb_cddtl_cust_base_0352(zone_val);	1	10000	10
tb_cddtl_cust_base_0353	zone_val	CREATE INDEX idx_tb_cddtl_cust_base_0353_zone_val ON tb_cddtl_cust_base_0353(zone_val);	1	10000	8
tb_cddtl_cust_base_0354	zone_val	CREATE INDEX idx_tb_cddtl_cust_base_0354_zone_val ON tb_cddtl_cust_base_0354(zone_val);	1	10000	10
tb_cddtl_cust_base_0355	zone_val	CREATE INDEX idx_tb_cddtl_cust_base_0355_zone_val ON tb_cddtl_cust_base_0355(zone_val);	1	10000	7
tb_cddtl_cust_base_0358	zone_val	CREATE INDEX idx_tb_cddtl_cust_base_0358_zone_val ON tb_cddtl_cust_base_0358(zone_val);	1	10000	7

正向收益SQL明细

sql模板 ↑	sql语句	优化倍数
select message_body from mq_msg_produce_2 WHERE status = ? AND crea...	select message_body from mq_msg_produce_2 WHERE status = 'F' AND create_stamp <= '2021-07-27 17:33:00....	1.4
select message_body from mq_msg_produce_2 WHERE status = ? AND crea...	select message_body from mq_msg_produce_2 WHERE status = 'F' AND create_stamp <= '2021-07-27 17:43:00....	1.4
select message_body from mq_msg_produce_2 WHERE status = ? AND crea...	select message_body from mq_msg_produce_2 WHERE status = 'F' AND create_stamp <= '2021-07-27 17:43:00....	1.4
select message_body from mq_msg_produce_2 WHERE status = ? AND crea...	select message_body from mq_msg_produce_2 WHERE status = 'F' AND create_stamp <= '2021-07-27 17:43:00....	1.4

负向收益SQL明细

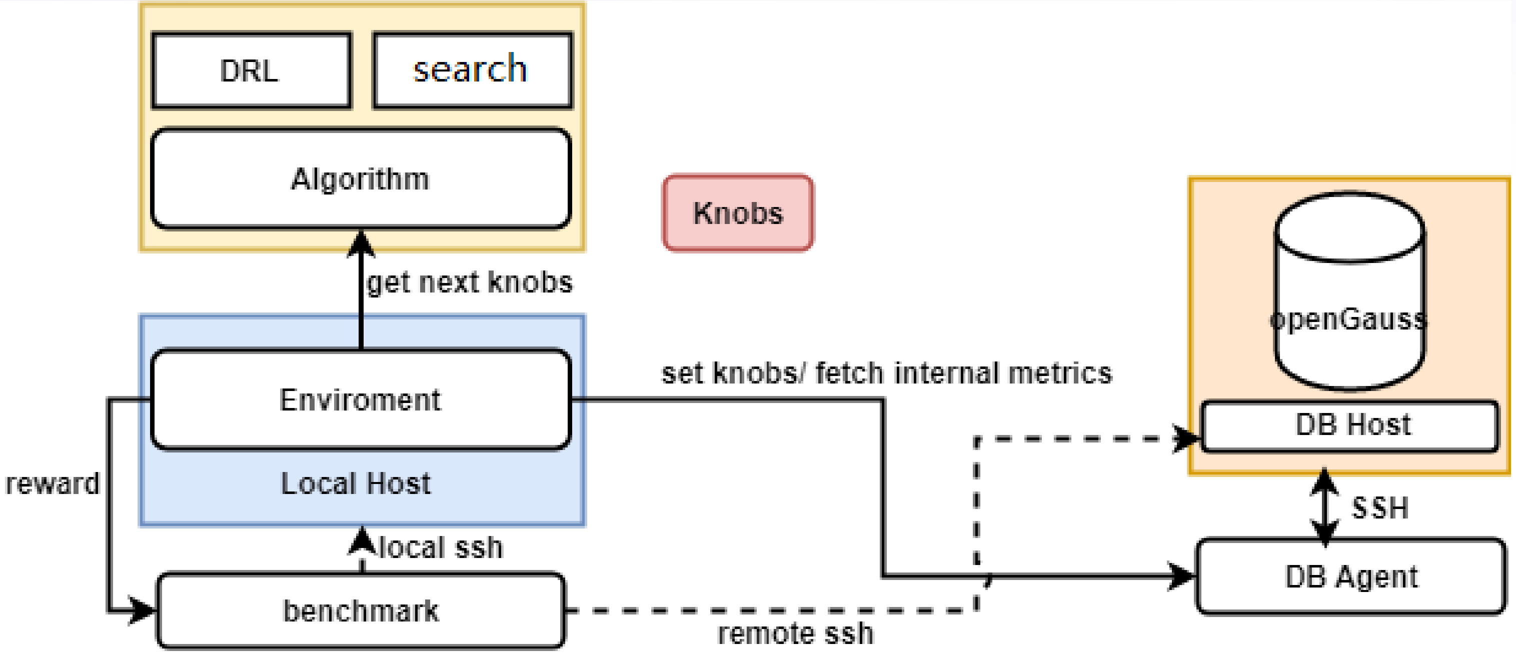
sql模板	sql语句

无关收益SQL明细

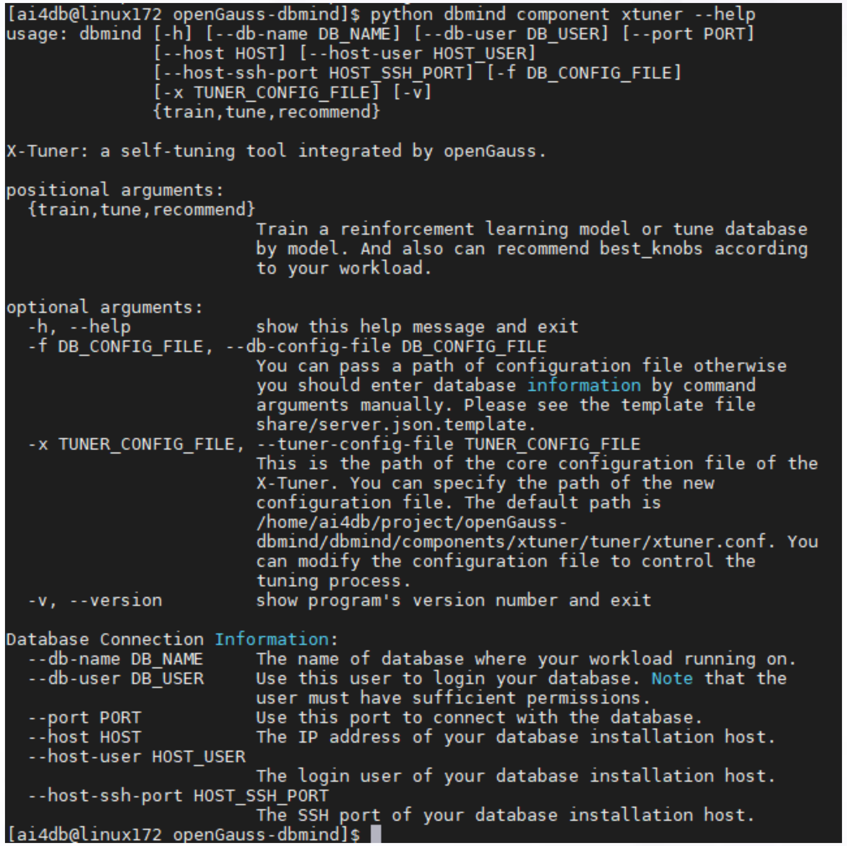
sql模板	sql语句
select message_body from mq_msg_produce_2 WHERE status = ? AND create_stamp <= ? AND optimist_lock_ver_no ...	select message_body from mq_msg_produce_2 WHERE status = 'F' AND create_stamp <= '2021-07-27 17:40:00.049' AN
select message_body from mq_msg_produce_2 WHERE status = ? AND create_stamp <= ? AND optimist_lock_ver_no ...	select message_body from mq_msg_produce_2 WHERE status = 'F' AND create_stamp <= '2021-07-27 17:34:00.017' AN

智能参数调优

1.利用长期参数调优总结出的先验规则进行参数配置诊断与生成数据库workload报告；
2.根据系统的workload、环境信息推荐初始参数配置，包括推荐参数值、建议最大值和最小值（用以保证稳定性，供用户结合自身经验进行选择）；
3.利用训练好的强化学习模型进行调优，或者使用全局优化算法在给定的参数空间内进行搜索；
4.常规评价调优效果好坏的方法是跑benchmark获得反馈，调优框架除支持常规benchmark如TPC-C、TPC-H等，还为用户提供了自定义benchmark的框架，用户只需要进行少量工作进行适配即可；目前还在演进支持Performance Model的参数调优，将更进一步加快调优速度；



智能参数调优用法



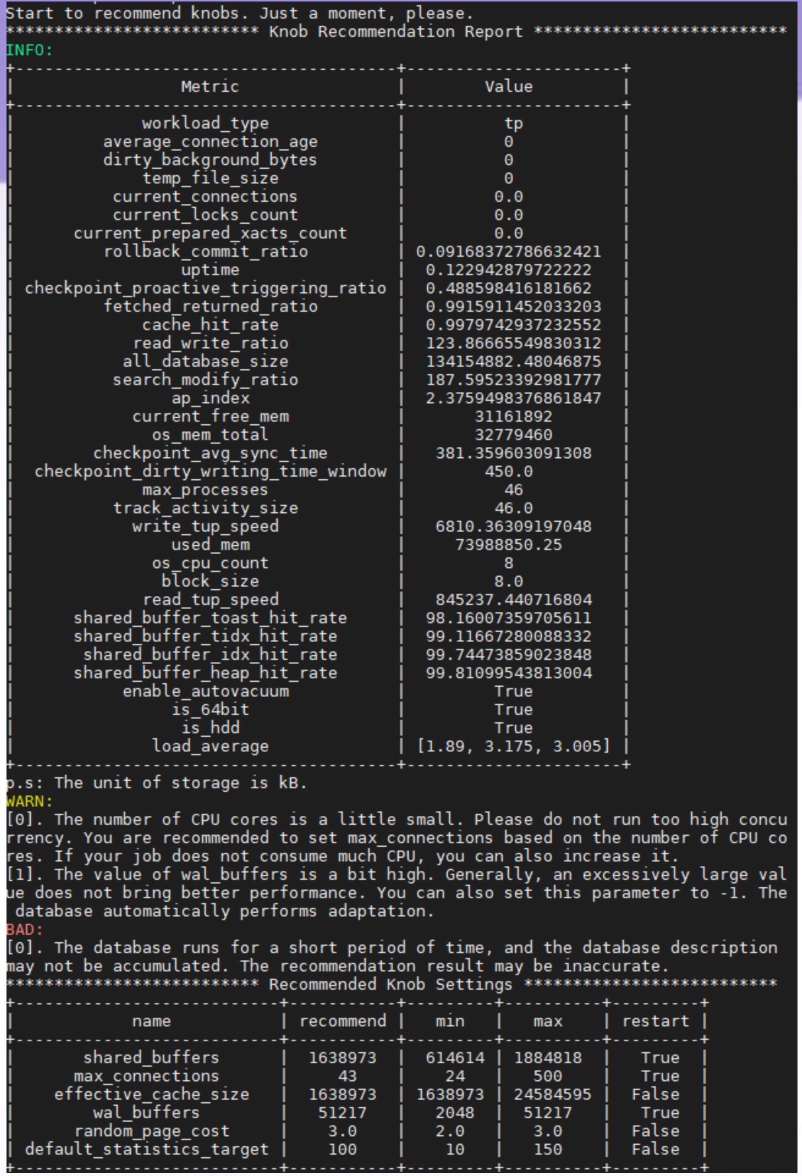
```
[ai4db@linux172 openGauss-dbmind]$ python dbmind component xtuner --help
usage: dbmind [-h] [--db-name DB_NAME] [--db-user DB_USER] [--port PORT]
              [--host HOST] [--host-user HOST_USER]
              [--host-ssh-port HOST_SSH_PORT] [-f DB_CONFIG_FILE]
              [-x TUNER_CONFIG_FILE] [-v]
              {train,tune,recommend}

X-Tuner: a self-tuning tool integrated by openGauss.

positional arguments:
  {train,tune,recommend}
                        Train a reinforcement learning model or tune database
                        by model. And also can recommend best_knobs according
                        to your workload.

optional arguments:
  -h, --help            show this help message and exit
  -f DB_CONFIG_FILE, --db-config-file DB_CONFIG_FILE
                        You can pass a path of configuration file otherwise
                        you should enter database information by command
                        arguments manually. Please see the template file
                        share/server.json.template.
  -x TUNER_CONFIG_FILE, --tuner-config-file TUNER_CONFIG_FILE
                        This is the path of the core configuration file of the
                        X-Tuner. You can specify the path of the new
                        configuration file. The default path is
                        /home/ai4db/project/openGauss-
                        dbmind/dbmind/components/xtuner/tuner/xtuner.conf. You
                        can modify the configuration file to control the
                        tuning process.
  -v, --version         show program's version number and exit

Database Connection Information:
  --db-name DB_NAME     The name of database where your workload running on.
  --db-user DB_USER     Use this user to login your database. Note that the
                        user must have sufficient permissions.
  --port PORT           Use this port to connect with the database.
  --host HOST           The IP address of your database installation host.
  --host-user HOST_USER
                        The login user of your database installation host.
  --host-ssh-port HOST_SSH_PORT
                        The SSH port of your database installation host.
[ai4db@linux172 openGauss-dbmind]$
```



```
Start to recommend knobs. Just a moment, please.
************************** Knob Recommendation Report **************************
INFO:
+---------------------------------------+----------------------+
|                 Metric                |        Value         |
+---------------------------------------+----------------------+
|             workload_type             |          tp          |
|         average_connection_age        |          0           |
|         dirty_background_bytes        |          0           |
|             temp_file_size            |          0           |
|          current_connections          |         0.0          |
|          current_locks_count          |         0.0          |
|      current_prepared_xacts_count     |         0.0          |
|         rollback_commit_ratio         | 0.09168372786632421  |
|                 uptime                |  0.122942879722222   |
| checkpoint_proactive_triggering_ratio |  0.488598416181662   |
|         fetched_returned_ratio        |  0.9915911452033203  |
|             cache_hit_rate            |  0.9979742937232552  |
|            read_write_ratio           |  123.86665549830312  |
|           all_database_size           |  134154882.48046875  |
|          search_modify_ratio          |  187.59523392981777  |
|                ap_index               |  2.3759498376861847  |
|            current_free_mem           |       31161892       |
|              os_mem_total             |       32779460       |
|        checkpoint_avg_sync_time       |   381.359603091308   |
|  checkpoint_dirty_writing_time_window |        450.0         |
|             max_processes             |          46          |
|          track_activity_size          |         46.0         |
|            write_tup_speed            |   6810.36309197048   |
|                used_mem               |     73988850.25      |
|              os_cpu_count             |          8           |
|               block_size              |         8.0          |
|             read_tup_speed            |   845237.440716804   |
|      shared_buffer_toast_hit_rate     |  98.16007359705611   |
|      shared_buffer_tidx_hit_rate      |  99.11667280088332   |
|       shared_buffer_idx_hit_rate      |  99.74473859023848   |
|      shared_buffer_heap_hit_rate      |  99.81099543813004   |
|           enable_autovacuum           |         True         |
|                is_64bit               |         True         |
|                 is_hdd                |         True         |
|              load_average             | [1.89, 3.175, 3.005] |
+---------------------------------------+----------------------+
p.s: The unit of storage is kB.
WARN:
[0]. The number of CPU cores is a little small. Please do not run too high concu
rrency. You are recommended to set max_connections based on the number of CPU co
res. If your job does not consume much CPU, you can also increase it.
[1]. The value of wal_buffers is a bit high. Generally, an excessively large val
ue does not bring better performance. You can also set this parameter to -1. The
 database automatically performs adaptation.
BAD:
[0]. The database runs for a short period of time, and the database description
may not be accumulated. The recommendation result may be inaccurate.
************************** Recommended Knob Settings **************************
+----------------------------+-----------+---------+----------+---------+
|            name            | recommend |   min   |   max    | restart |
+----------------------------+-----------+---------+----------+---------+
|       shared_buffers       |  1638973  |  614614 | 1884818  |   True  |
|      max_connections       |     43    |    24   |   500    |   True  |
|    effective_cache_size    |  1638973  | 1638973 | 24584595 |  False  |
|        wal_buffers         |   51217   |   2048  |  51217   |   True  |
|      random_page_cost      |    3.0    |   2.0   |   3.0    |  False  |
| default_statistics_target  |    100    |    10   |   150    |  False  |
+----------------------------+-----------+---------+----------+---------+
```

openGauss 智能优化器

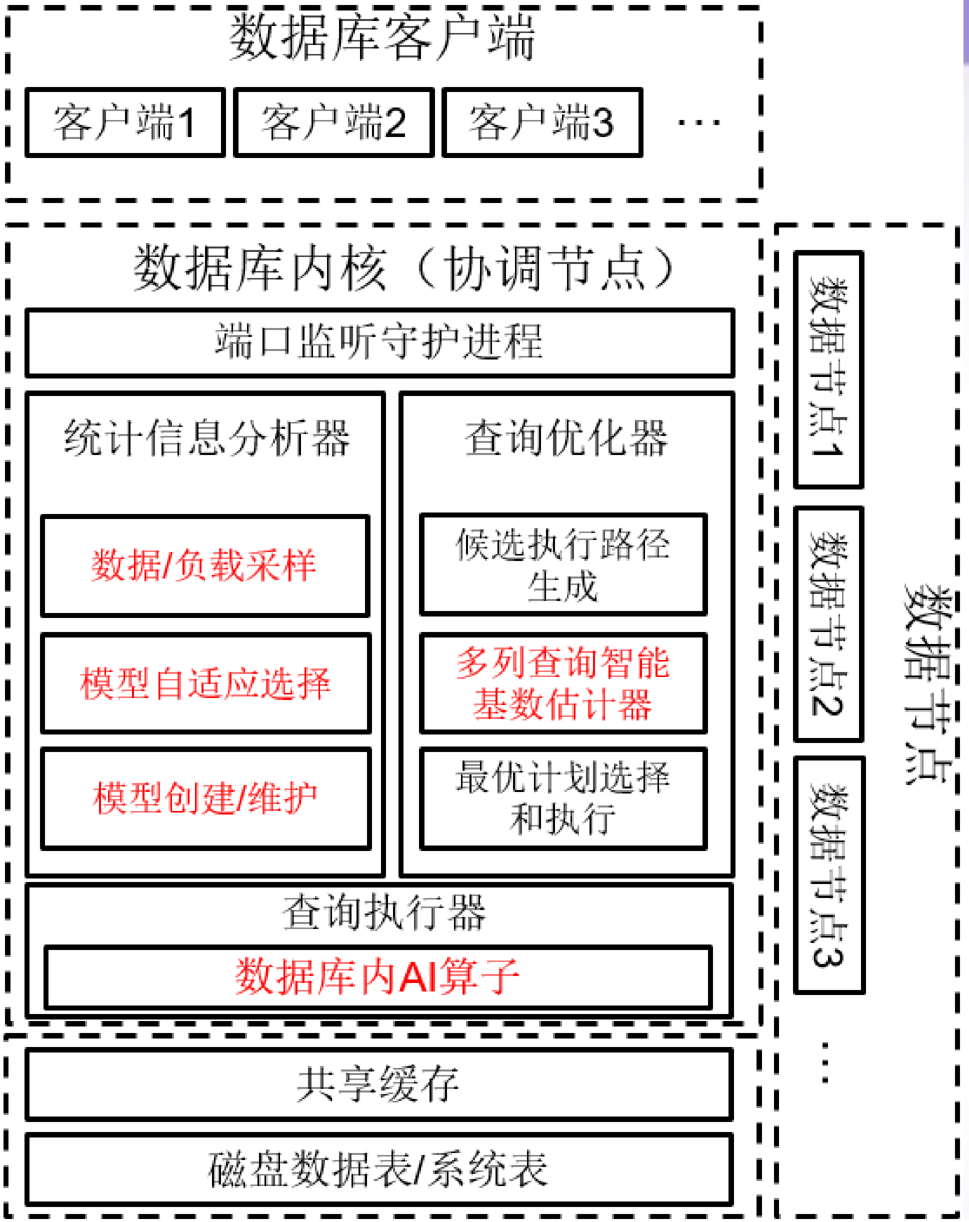
（1）**数据/负载采样：**对于全量数据进行均匀采样，采样对象包括基表、多表内连接结果以及历史查询表；采样出来的数据会作为训练样本输入到基数估计模型中；

（2）**模型自适应选择：**根据数据/负载的分布特征从模型库中选择最合适的模型对于数据/负载基数分布进行建模，以便增加智能基数估计的应用场景；

（3）**模型创建/维护：**调用原生AI算子，利用数据样本进行训练完成模型创建；用新创建的模型从共享缓存以及磁盘系统表中替换旧模型完成模型更新；

（4）**多列查询智能基数估计：**该模块识别出包含多列谓词查询的子查询计划，然后匹配负载这些列的基数估计模型，调用AI算子执行模型获得估计的查询基数。

（5）**数据库内AI框架：**该模块提供原生数据库算子，提供给智能基数估计进行模型训练、推理和维护。

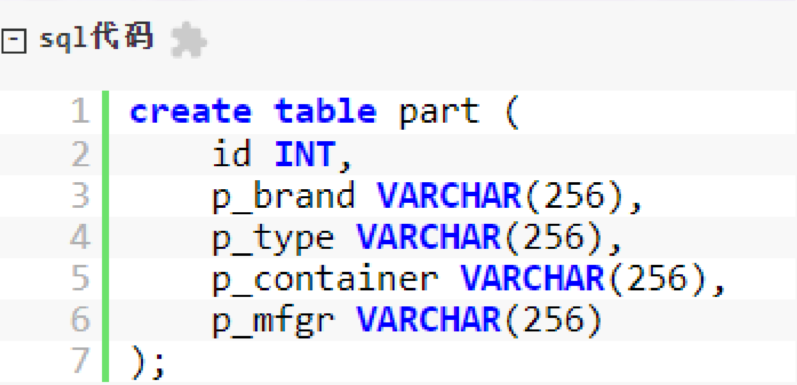


Gauss松鼠会

openGauss

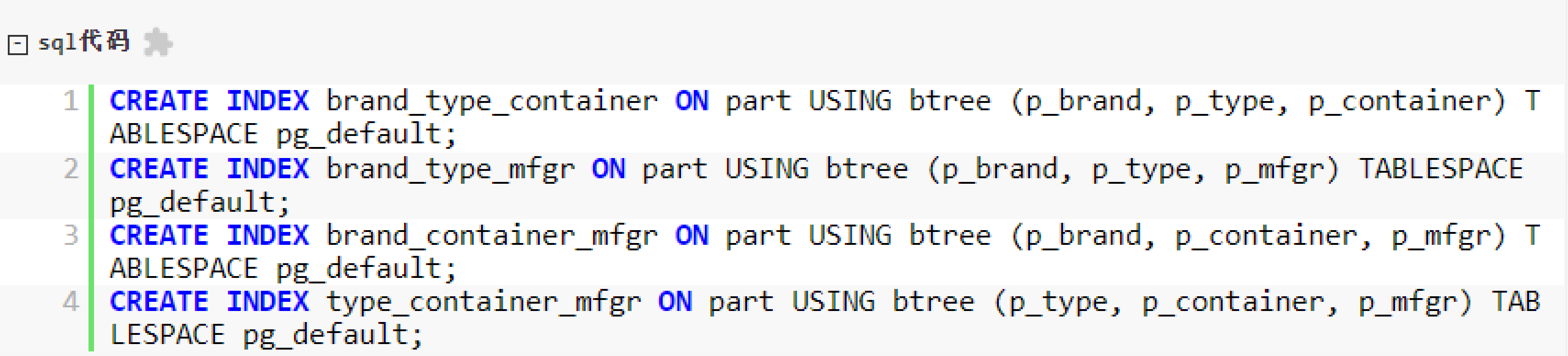
https://opengauss.org

智能优化器使用场景



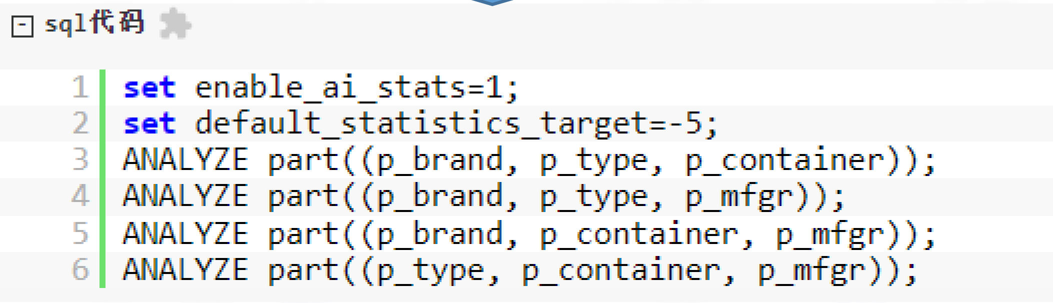
sql代码

```sql
create table part (
	id INT,
	p_brand VARCHAR(256),
	p_type VARCHAR(256),
	p_container VARCHAR(256),
	p_mfgr VARCHAR(256)
);
```



sql代码

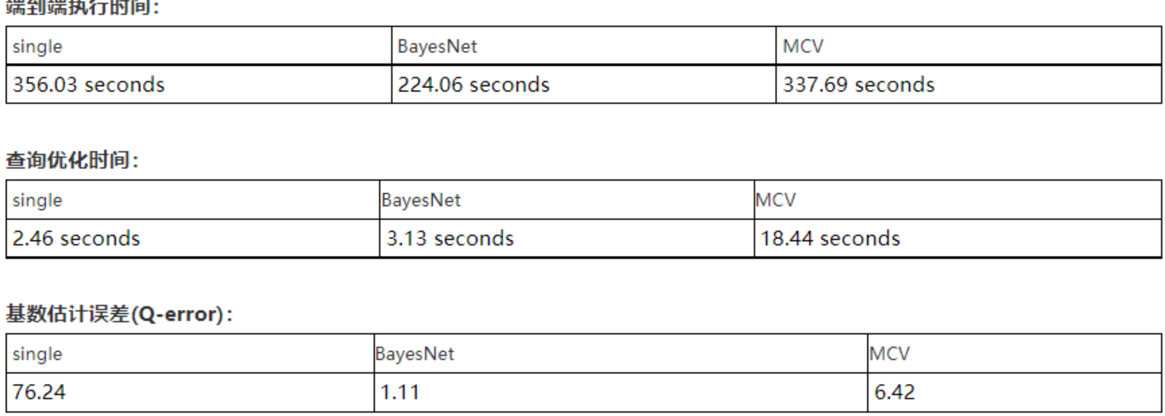
```sql
CREATE INDEX brand_type_container ON part USING btree (p_brand, p_type, p_container) TABLESPACE pg_default;
CREATE INDEX brand_type_mfgr ON part USING btree (p_brand, p_type, p_mfgr) TABLESPACE pg_default;
CREATE INDEX brand_container_mfgr ON part USING btree (p_brand, p_container, p_mfgr) TABLESPACE pg_default;
CREATE INDEX type_container_mfgr ON part USING btree (p_type, p_container, p_mfgr) TABLESPACE pg_default;
```



sql代码

```sql
set enable_ai_stats=1;
set default_statistics_target=-5;
ANALYZE part((p_brand, p_type, p_container));
ANALYZE part((p_brand, p_type, p_mfgr));
ANALYZE part((p_brand, p_container, p_mfgr));
ANALYZE part((p_type, p_container, p_mfgr));
```

智能优化器效果



端到端执行时间：

single	BayesNet	MCV
356.03 seconds	224.06 seconds	337.69 seconds

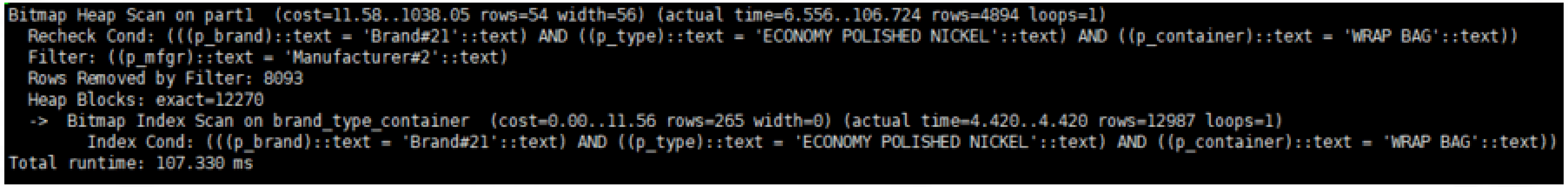
查询优化时间：

single	BayesNet	MCV
2.46 seconds	3.13 seconds	18.44 seconds

基数估计误差(Q-error)：

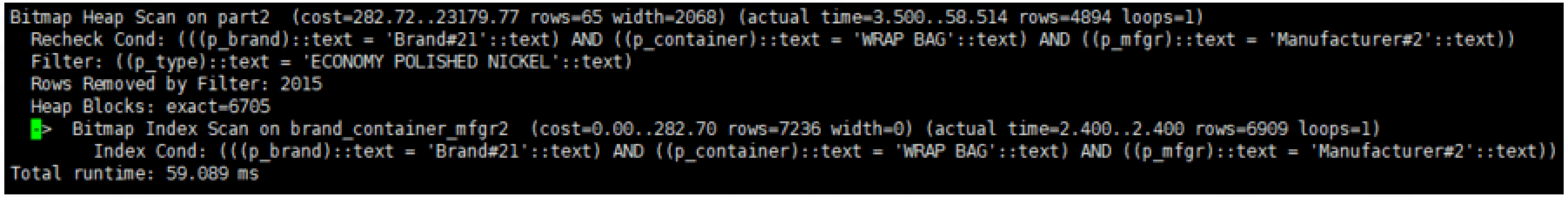
single	BayesNet	MCV
76.24	1.11	6.42

传统单列统计信息



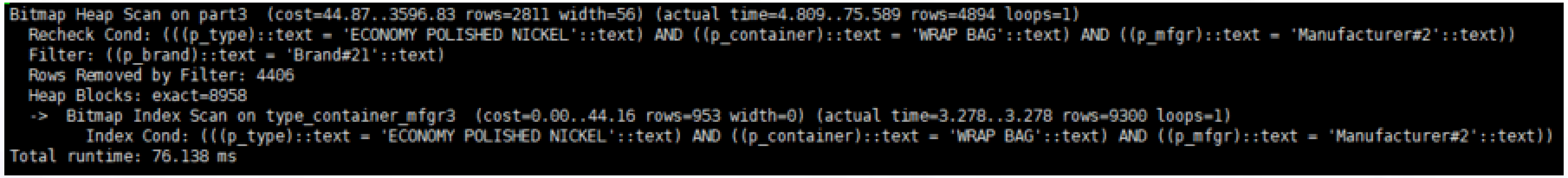
```
Bitmap Heap Scan on part1  (cost=11.58..1038.05 rows=54 width=56) (actual time=6.556..106.724 rows=4894 loops=1)
  Recheck Cond: (((p_brand)::text = 'Brand#21'::text) AND ((p_type)::text = 'ECONOMY POLISHED NICKEL'::text) AND ((p_container)::text = 'WRAP BAG'::text))
  Filter: ((p_mfgr)::text = 'Manufacturer#2'::text)
  Rows Removed by Filter: 8093
  Heap Blocks: exact=12270
  ->  Bitmap Index Scan on brand_type_container  (cost=0.00..11.56 rows=265 width=0) (actual time=4.420..4.420 rows=12987 loops=1)
        Index Cond: (((p_brand)::text = 'Brand#21'::text) AND ((p_type)::text = 'ECONOMY POLISHED NICKEL'::text) AND ((p_container)::text = 'WRAP BAG'::text))
Total runtime: 107.330 ms
```

传统MCV统计信息



```
Bitmap Heap Scan on part2  (cost=282.72..23179.77 rows=65 width=2068) (actual time=3.500..58.514 rows=4894 loops=1)
  Recheck Cond: (((p_brand)::text = 'Brand#21'::text) AND ((p_container)::text = 'WRAP BAG'::text) AND ((p_mfgr)::text = 'Manufacturer#2'::text))
  Filter: ((p_type)::text = 'ECONOMY POLISHED NICKEL'::text)
  Rows Removed by Filter: 2015
  Heap Blocks: exact=6705
  ->  Bitmap Index Scan on brand_container_mfgr2  (cost=0.00..282.70 rows=7236 width=0) (actual time=2.400..2.400 rows=6909 loops=1)
        Index Cond: (((p_brand)::text = 'Brand#21'::text) AND ((p_container)::text = 'WRAP BAG'::text) AND ((p_mfgr)::text = 'Manufacturer#2'::text))
Total runtime: 59.089 ms
```

智能基数估计



```
Bitmap Heap Scan on part3  (cost=44.87..3596.83 rows=2811 width=56) (actual time=4.809..75.589 rows=4894 loops=1)
  Recheck Cond: (((p_type)::text = 'ECONOMY POLISHED NICKEL'::text) AND ((p_container)::text = 'WRAP BAG'::text) AND ((p_mfgr)::text = 'Manufacturer#2'::text))
  Filter: ((p_brand)::text = 'Brand#21'::text)
  Rows Removed by Filter: 4406
  Heap Blocks: exact=8958
  ->  Bitmap Index Scan on type_container_mfgr3  (cost=0.00..44.16 rows=953 width=0) (actual time=3.278..3.278 rows=9300 loops=1)
        Index Cond: (((p_type)::text = 'ECONOMY POLISHED NICKEL'::text) AND ((p_container)::text = 'WRAP BAG'::text) AND ((p_mfgr)::text = 'Manufacturer#2'::text))
Total runtime: 76.138 ms
```

Thank you!

Gauss松鼠会

openGauss

https://opengauss.org