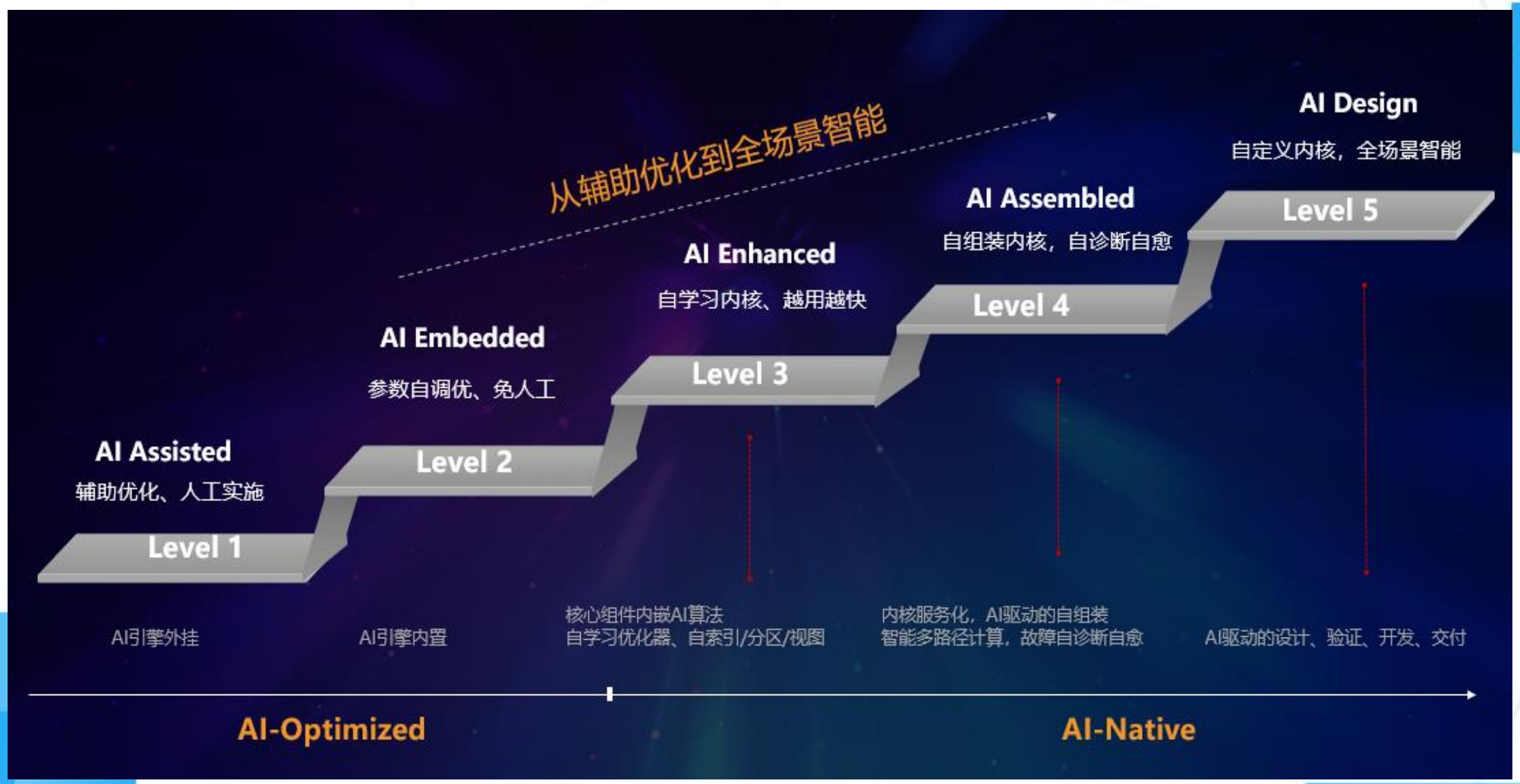
AI在openGauss的实现解析

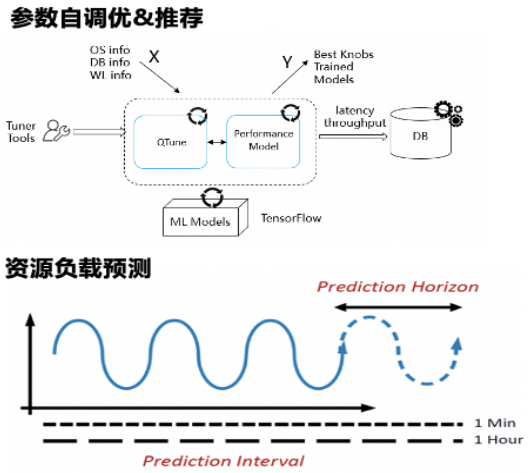
高云龙@云和恩墨

数据库结合人工智能的不同阶段



提升数据库智能化场景

1 自调优，持续高性能



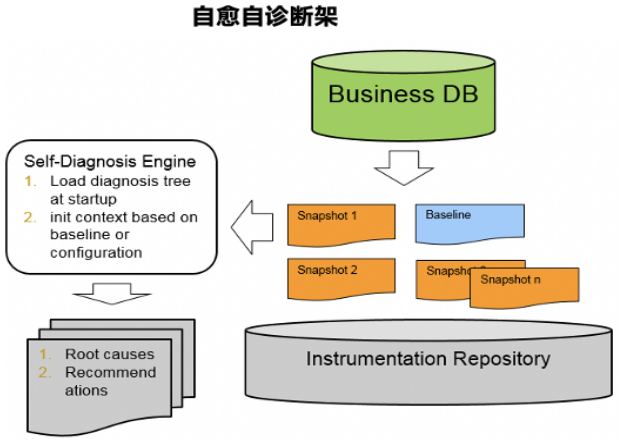
云化智能调优

- **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20%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

云化负载预测&在线调参

- **负载预测**：准确预测负载性能，支撑用户高效编排业务、智能负载调度
- **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

2 自诊断自愈，Always Online



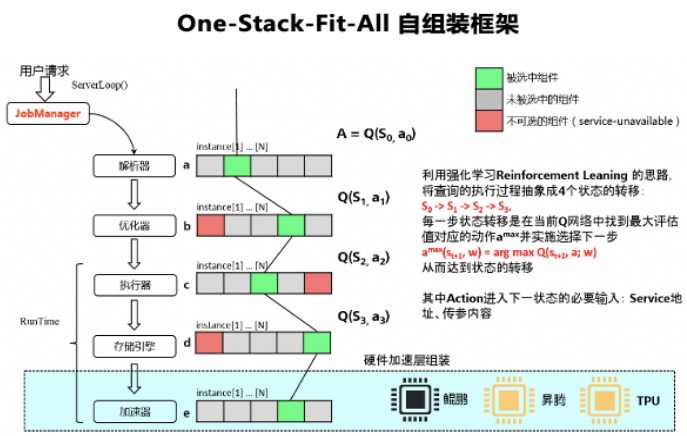
自诊断

- **诊断引擎**：支持AI和Rule引擎结合的基于Trace智能诊断系统
- **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 自组装，全场景数据



丰富的行业应用、支持多样化负载

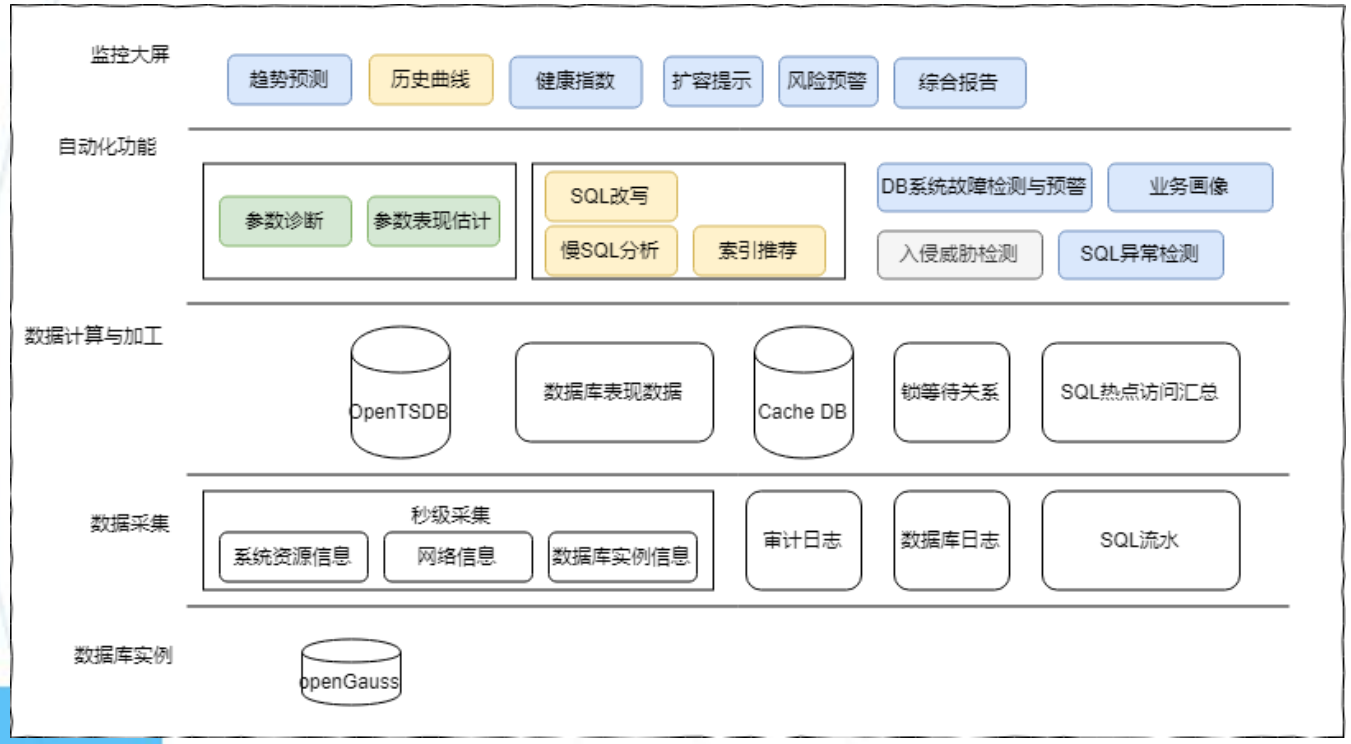
- **业务场景和负载多样化**，单一通用数据库平台无法满足
- 现有的产品研发模式无法满足定制化交付

组件化自组装

- **组件微服务化**：基于搜索优化理论，突破组合空间爆炸问题
- **自组装技术**：基于最优控制论，软件执行栈轻量化、组件组合最优化

墨天轮

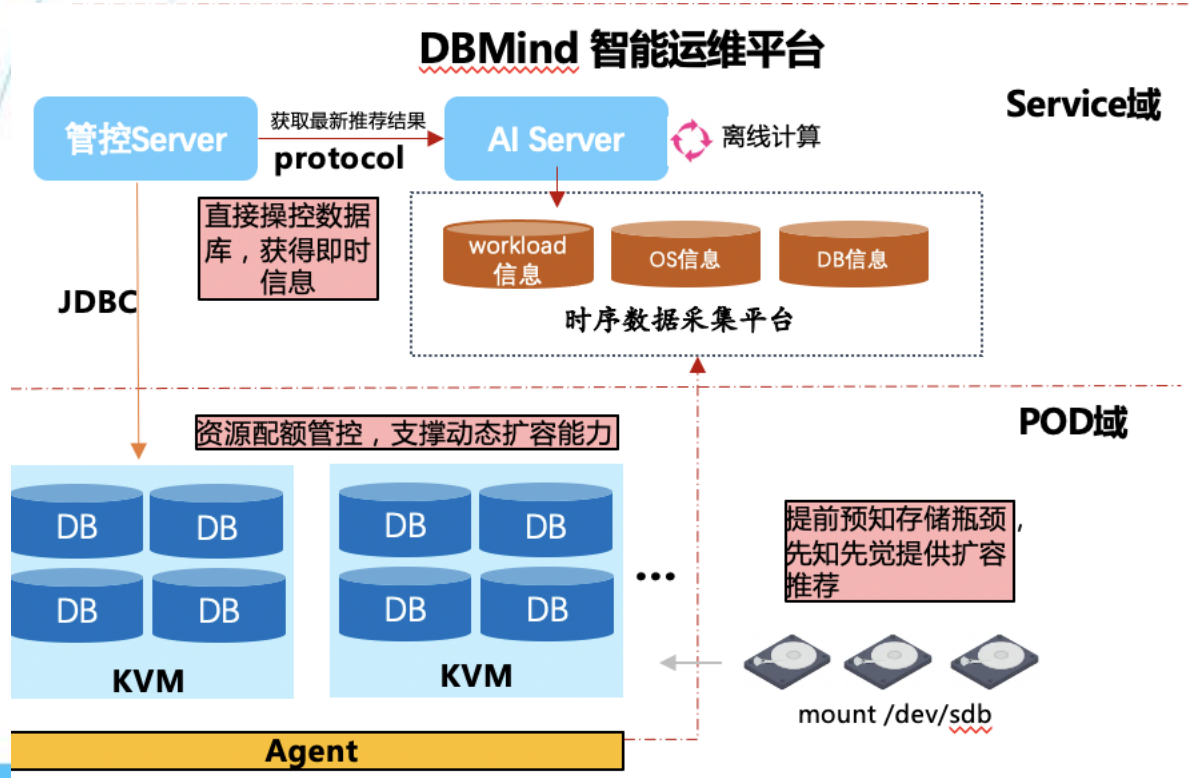
DBMind智能管理平台

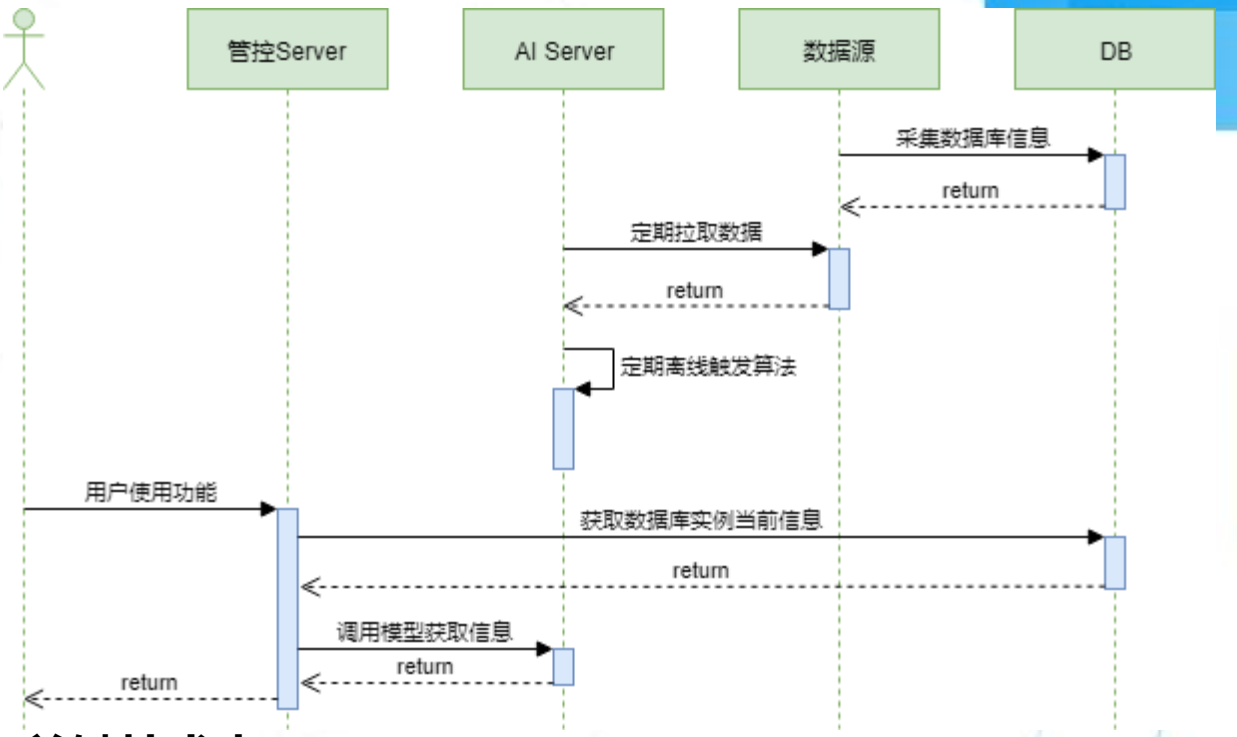


功能列表

历史信息收集与可视化
SQL改写
慢SQL诊断与建议
索引推荐
参数推荐
参数表现估计
趋势预测（AI负载、性能预测）
扩容提示（AI存储空间预测）
健康指数
风险预警
综合报告
DB系统故障检测与预警
业务画像
入侵威胁检测
……

DBMind智能管理平台的典型架构





特点

① 智能化：定义自治能力级别，提供自助式，智能化的服务

② 强调“预测”的特性：提供先知先觉的能力

③ 主要面向workload级别调优：参数调优、索引推荐等功能聚焦在workload上

关键技术点

① 基于机器学习/ 深度学习的SQL优化，如索引推荐、Partition Key选择

② 提供先知先觉的异常检测、扩容（存储）提醒

③ 提供AI性能趋势预测，开启用户运维窗口

④ 数据库内置AI算法或函数，如单条语句索引推荐，提供用户易用运维接口

参数调优与诊断能力

调优参数列表：根据不同的场景预设，用户也可以根据经验配置；

调优方法概括：结合**深度强化学习**与**全局优化算法**，针对不同类别的参数进行细粒度调优。

调优效果评估：观察benchmark的跑分结果，预置的benchmark丰富，极简可扩展。

离线参数调优流程概述：

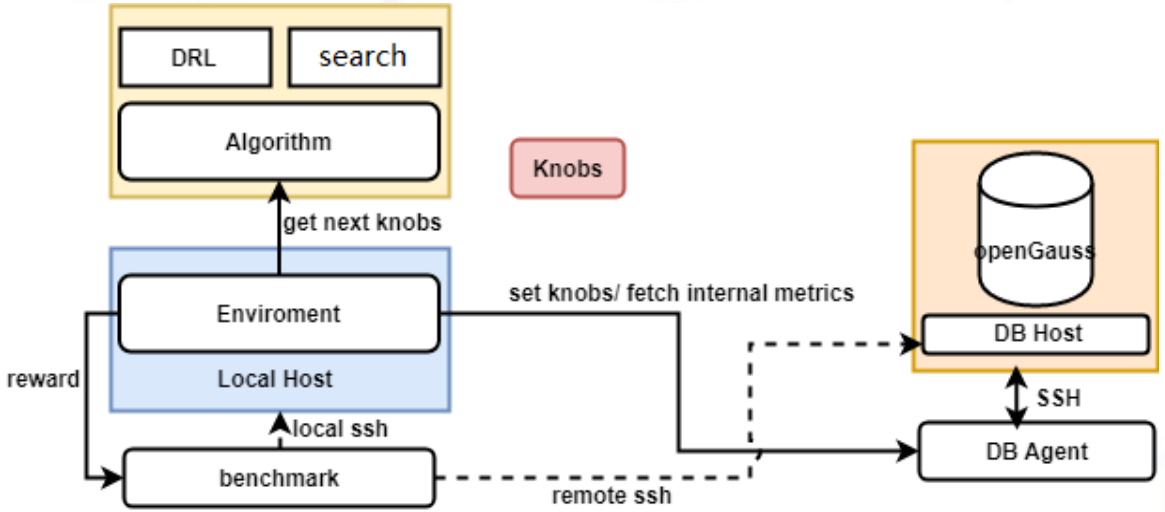
1. 利用长期参数调优总结出的先验规则进行参数配置诊断与生成数据库workload报告；
2. 根据系统的workload、环境信息推荐初始参数配置，包括推荐参数值、建议最大值和最小值（用以保证稳定性，供用户结合自身经验进行选择）；
3. 利用训练好的强化学习模型进行调优，或者使用全局优化算法在给定的参数空间内进行搜索；
4. 常规评价调优效果好坏的方法是跑benchmark获得反馈，调优框架除支持常规benchmark如TPC-C、TPC-H等，还为用户提供了自定义benchmark的框架，用户只需要进行少量工作进行适配即可；目前还在演进支持Performance Model的参数调优，将更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

在线参数调优流程概述：

采集用户系统内的统计信息和workload，根据训练好的监督学习模型和先验规则，推荐给用户对应的参数配置。

参数调优工具X-Tuner

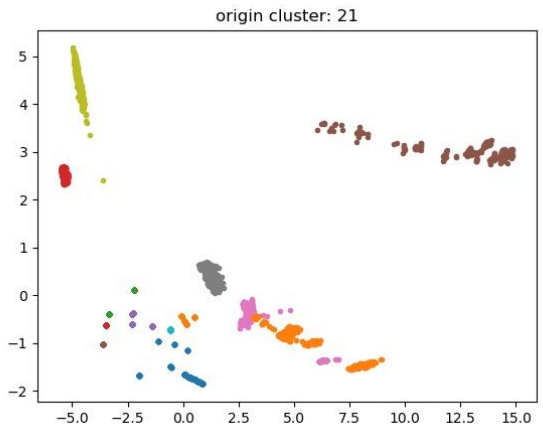
- recommend: 通过用户指定的用户名等信息登录到数据库环境中，获取当前正在运行的workload特征信息，根据上述特征信息生成参数推荐报告。报告当前数据库中不合理的参数配置和潜在风险等；输出根据当前正在运行的workload行为和特征；输出推荐的参数配置。**该模式是秒级的，不涉及数据库的重启操作，其他模式可能需要反复重启数据库。**
- train: 通过用户提供的benchmark信息，不断地进行参数修改和benchmark的执行。通过反复的迭代过程，训练强化学习模型，以便用户在后面通过tune模式加载该模型进行调优。
- tune: 使用优化算法进行数据库参数的调优，当前支持两大类算法，一种是深度强化学习，另一种是全局搜索算法（全局优化算法）。深度强化学习模式要求先运行train模式，生成训练后的调优模型，而使用全局搜索算法则不需要提前进行训练，可以直接进行搜索调优。



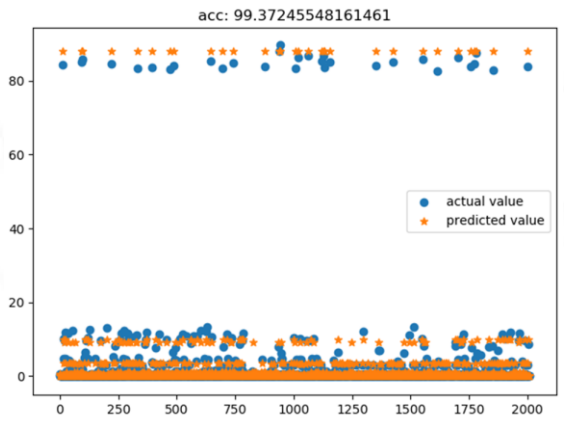
慢SQL发现能力

1. 上线业务预检测：上线一批新业务前，使用SQL诊断功能评估此次上线业务的预估执行时长，便于用户参考是否应该修改上线业务。
2. Workload分析：能够对现有workload进行分析，将现有workload自动分为若干类别，并依次分析此类别SQL语句执行代价，以及各个类别之间的相似程度；
3. SQL透视：能够将workload进行可视化，通过不同颜色和距离远近判断SQL语句之间的相似性，从而供用户直观地分析SQL语句的特点。

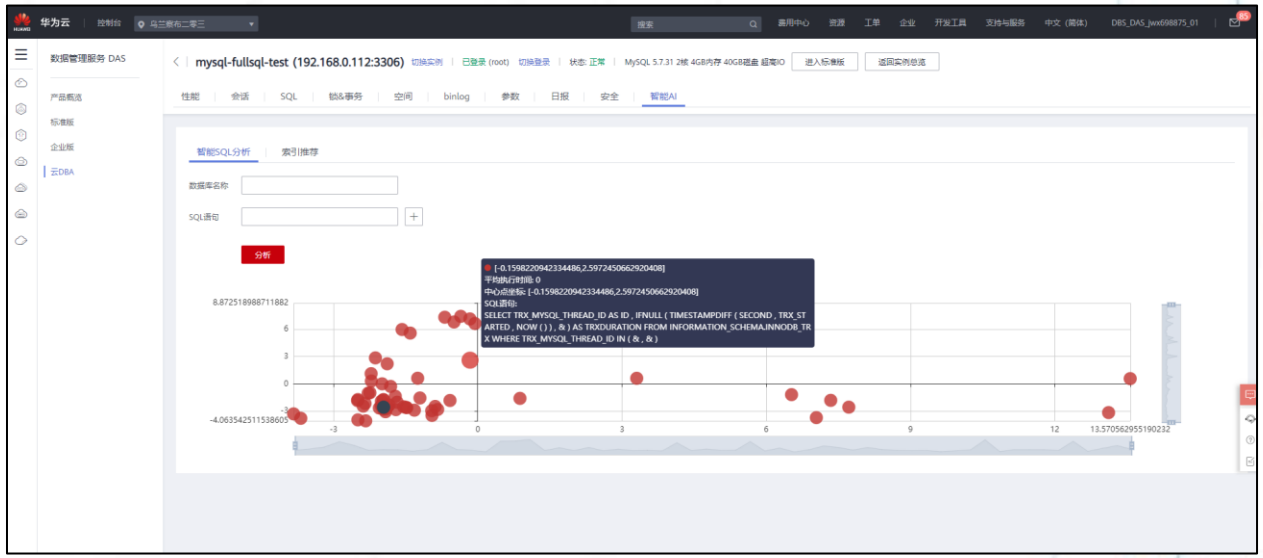
SQL透视效果



具备较高的预测准确率



华为云DAS上基于海量数据训练后的SQL透视效果



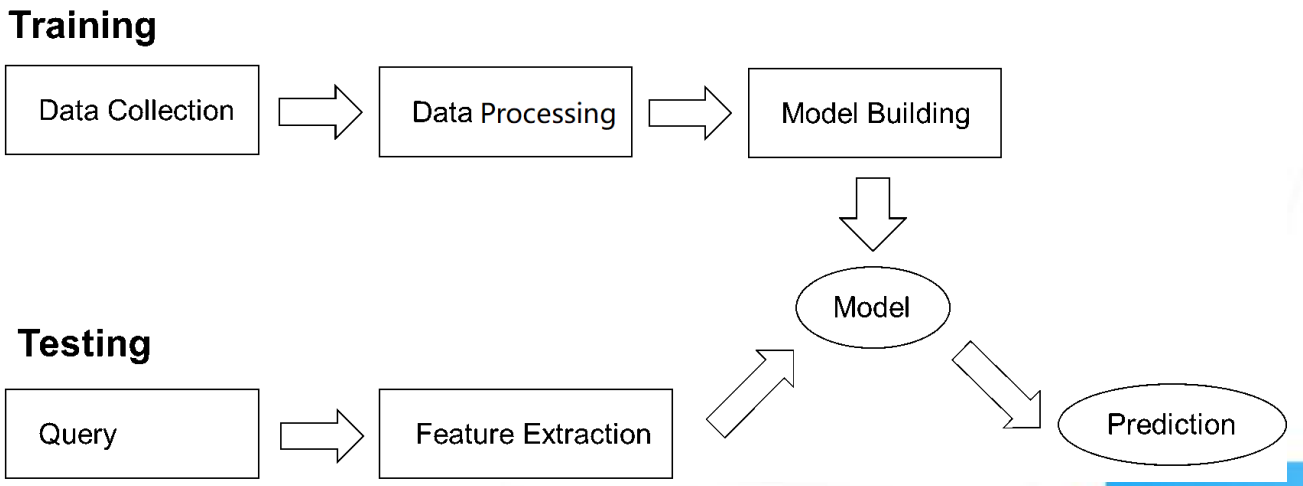
SQL执行时长预测工具SQLdiag

SQLdiag是一个SQL语句执行时间预测工具，通过模板化方法，实现在不获取SQL语句执行计划的前提下，依据语句逻辑相似度与历史执行记录，预测SQL语句的执行时间。

SQLdiag着眼于数据库的历史SQL语句，通过对历史SQL语句的执行表现进行总结归纳，将之再用于推断新的未知业务上。由于短时间内数据库SQL语句执行时长不会有太大的差距，SQLdiag可以从历史数据中检测出与已执行SQL语句相似的语句结果集，并基于SQL向量化技术和模板化方法预测SQL语句执行时长。

特点：

1、不需要SQL语句的执行计划，对数据库性能不会有任何的影响。
2、使用场景广泛，目前业内的很多算法局限性比较高，比如只适用于OLTP或者OLAP，而SQLdiag使用场景广泛。
3、该框架鲁棒性强，容易理解，只需要简单的操作，就可以训练出自己的预测模型。



单query索引推荐

单索引推荐函数

函数名	参数	功能
gs_index_advise	SQL语句字符串	针对单条查询语句生成推荐索引。

获取针对该query生成的推荐索引，推荐结果由索引的表名和列名组成

```
postgres=> select * from gs_index_advise('SELECT c_discount from bmsql_customer where c_w_id = 10');
     table      |  column
----------------+----------
 bmsql_customer | (c_w_id)
 (1 row)
```

应用价值

01 索引设计工具化、标准化

02 提升作业查询效率

03 减少DBA运维操作

虚拟索引

函数名	参数	功能
hypopg_create_index	创建索引语句的字符串	创建虚拟索引。
hypopg_display_index	无	显示所有创建的虚拟索引信息。
hypopg_drop_index	索引的oid	删除指定的虚拟索引。
hypopg_reset_index	无	清除所有虚拟索引。
hypopg_estimate_size	索引的oid	估计指定索引创建所需的空间大小。

1、支持用户在数据库中直接进行操作

2、避免真实索引创建所需的时间和空间开销

3、基于虚拟索引，可通过优化器评估该索引对指定查询语句的代价影响。

虚拟索引功能的GUC参数

参数名	功能	默认值
enable_hypo_index	是否开启虚拟索引功能	off

虚拟索引使用示例

使用函数hypopg_create_index创建虚拟索引

```
mogdb=> select * from hypopg_create_index('create index on bmsql_customer(c_w_id)');
 indexrelid |              indexname
------------+-------------------------------------
     329726 | <329726>btree_bmsql_customer_c_w_id
(1 row)
```

开启GUC参数前

```
mogdb=> explain SELECT c_discount from bmsql_customer where c_w_id = 10;
                              QUERY PLAN
-----------------------------------------------------------------------
 Seq Scan on bmsql_customer  (cost=0.00..52963.06 rows=31224 width=4)
   Filter: (c_w_id = 10)
(2 rows)
```

开启GUC参数后

```
mogdb=> explain SELECT c_discount from bmsql_customer where c_w_id = 10;
                                                  QUERY PLAN
-------------------------------------------------------------------------------------------------------------
 [Bypass]
 Index Scan using <329726>btree_bmsql_customer_c_w_id on bmsql_customer  (cost=0.00..39678.69 rows=31224 width=4)
   Index Cond: (c_w_id = 10)
(3 rows)
```

开启GUC参数enable_hypo_index

```
mogdb=> set enable_hypo_index = on;
SET
```

墨天轮

虚拟索引使用示例

展示所有创建过的虚拟索引

```
mogdb=> select * from hypopg_display_index();
                indexname                 | indexrelid |     table      |      column
------------------------------------------+------------+----------------+------------------
 <329726>btree_bmsql_customer_c_w_id      |     329726 | bmsql_customer | (c_w_id)
 <329729>btree_bmsql_customer_c_d_id_c_w_id |     329729 | bmsql_customer | (c_d_id, c_w_id)
(2 rows)
```

预估虚拟索引创建所需的空间大小（单位：字节）

```
mogdb=> select * from hypopg_estimate_size(329730);
 hypopg_estimate_size
----------------------
             15687680
(1 row)
```

删除虚拟索引

使用函数hypopg_drop_index删除指定oid的虚拟索引。例如：

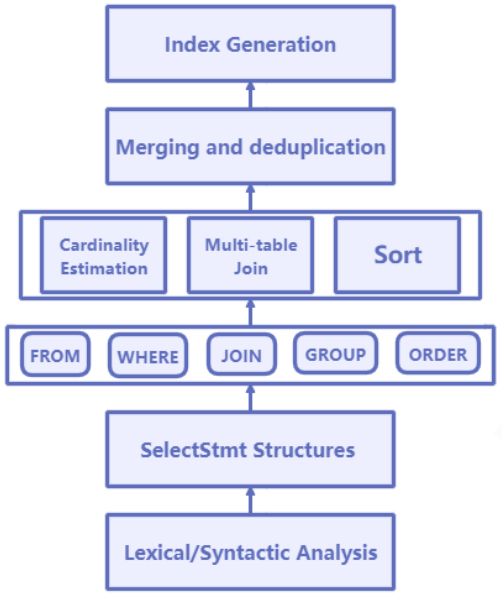
```
mogdb=> select * from hypopg_drop_index(329726);
 hypopg_drop_index
-------------------
 t
(1 row)
```

使用函数hypopg_reset_index一次性清除所有创建的虚拟索引。例如：

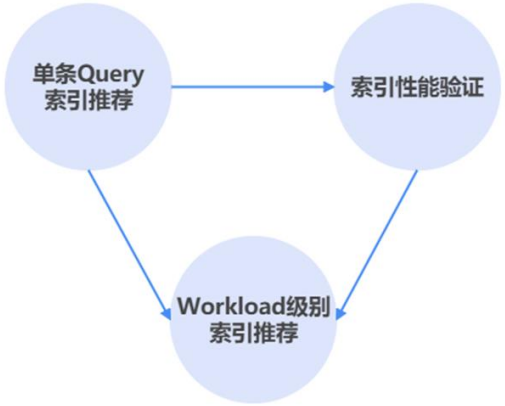
```
mogdb=> select * from hypopg_reset_index();
 hypopg_reset_index
--------------------

(1 row)
```

Workload级别索引推荐



从单条到workload



python脚本index_advisor_workload.py

```
python index_advisor_workload.py [p PORT] [d DATABASE] [f FILE] [--h HOST] [-U USERNAME] [-W PASSWORD]
[--max_index_num MAX_INDEX_NUM] [--multi_iter_mode]
```

其中的输入参数依次为：

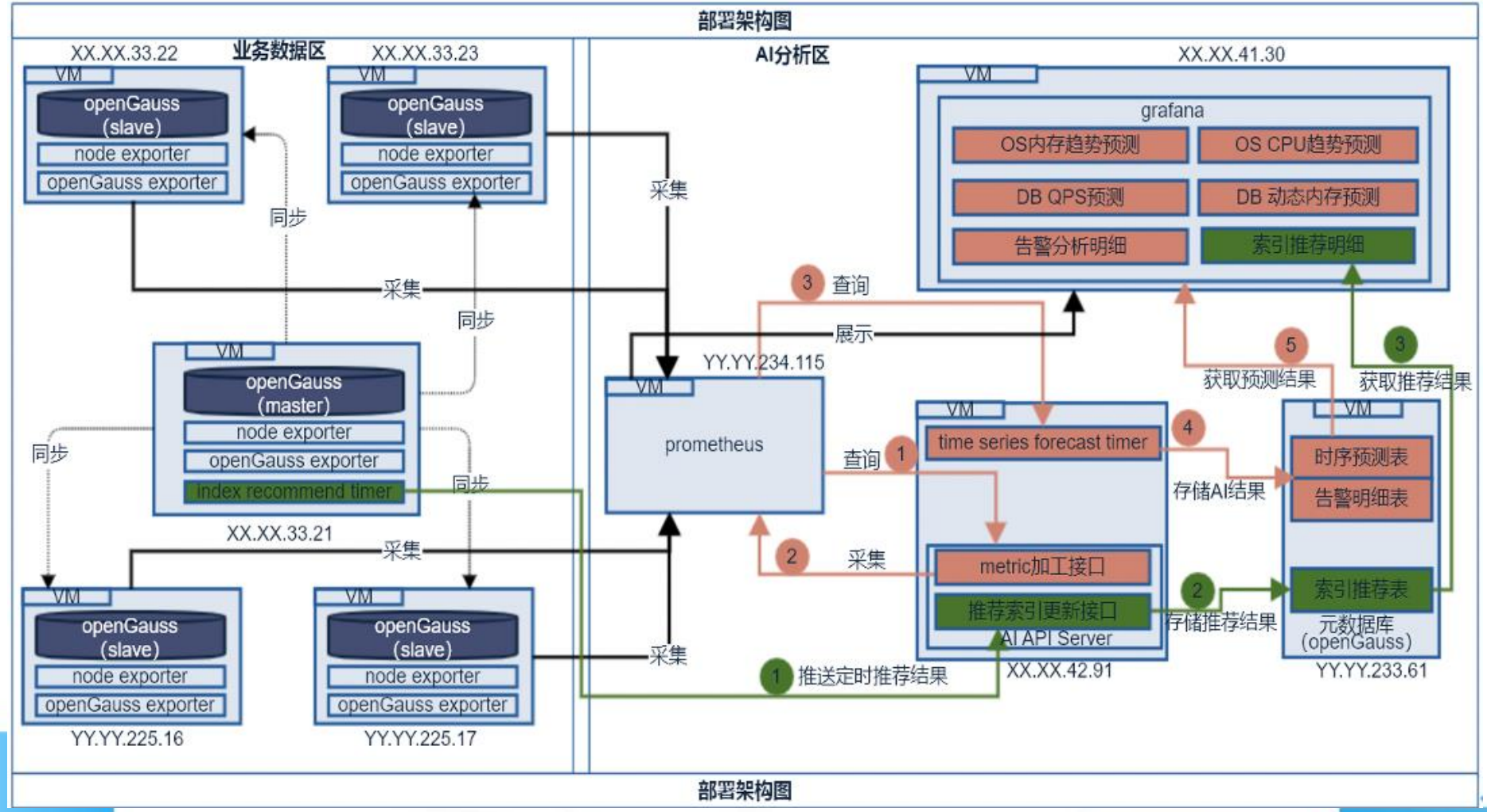
1. PORT：连接数据库的端口号。
2. DATABASE：连接数据库的名字。
3. FILE：包含workload语句的文件路径。
4. HOST：（可选）连接数据库的主机号。
5. USERNAME：（可选）连接数据库的用户名。
6. PASSWORD：（可选）连接数据库用户的密码。
7. MAX_INDEX_NUM：（可选）最大的索引推荐数目。
8. multi_iter_mode：（可选）算法模式，可通过是否设置该参数来切换算法。例如：

```
python index_advisor_workload.py 6001 postgres tpcc_log.txt --max_index_num 10 --multi_iter_mode
```

推荐结果为一批索引，以多个创建索引语句的格式显示在屏幕上，结果示例。

```
create index ind0 on bmsql_stock(s_i_id,s_w_id);
create index ind1 on bmsql_customer(c_w_id,c_id,c_d_id);
create index ind2 on bmsql_order_line(ol_w_id,ol_o_id,ol_d_id);
create index ind3 on bmsql_item(i_id);
create index ind4 on bmsql_oorder(o_w_id,o_id,o_d_id);
create index ind5 on bmsql_new_order(no_w_id,no_d_id,no_o_id);
create index ind6 on bmsql_customer(c_w_id,c_d_id,c_last,c_first);
create index ind7 on bmsql_new_order(no_w_id);
create index ind8 on bmsql_oorder(o_w_id,o_c_id,o_d_id);
create index ind9 on bmsql_district(d_w_id);
```

部署架构图


部署架构图
业务数据区
AI分析区
XX.XX.33.22
XX.XX.33.23
XX.XX.41.30
VM
openGauss (slave)
node exporter
openGauss exporter
openGauss (master)
index recommend timer
grafana
OS内存趋势预测
OS CPU趋势预测
DB QPS预测
DB 动态内存预测
告警分析明细
索引推荐明细
prometheus
YY.YY.234.115
time series forecast timer
metric加工接口
推荐索引更新接口
AI API Server
XX.XX.42.91
时序预测表
告警明细表
索引推荐表
元数据库 (openGauss)
YY.YY.233.61
XX.XX.33.21
YY.YY.225.16
YY.YY.225.17
同步
采集
查询
展示
获取预测结果
获取推荐结果
存储AI结果
存储推荐结果
推送定时推荐结果
1
2
3
4
5
部署架构图

趋势整体预测

OS指标预测

1、CPU趋势预测
2、内存趋势预测

DB指标预测

3、QPS趋势预测
4、动态内存趋势预测
5、连接数使用预测
6、DB Session Cache 预测

告警分析

7、告警明细

索引推荐

	表名	列名		优化前代价	优化后代价	代价降低倍数
2	tb t_cust_relat	prodt_	o, zone_v	2694.22	20.37	132
3	tb e_service_serial	accour	zone_val,	er_tech_sri_no	1083.47	10.99	99
4	tb e_service_serial	zone_v	nner_tech_		187531.71	1188.22	158
5	tb ch_tran_online_comb_succ	accour	main_bat		127186.31	155.27	819
6	tb onc_acc_status	accour	accti_sri_	sta_cd	956.35	8.29	115

索引优化前后代价对比统计

经典索引推荐对比

```
table: tb_trans_base_service_serial_6  columns: zone_val, core_inner_tech_sri_no
187531.71 -> 1188.22  =99%
dcdpdb=# explain select core_inner_tech_sri_no,pos_nega_tx_flag_cd,exec_step_seq_no,global_busi_track_no,subtx_no,
resended_times,serv_no,medium_no,main_contr_no,ledger_no,saccno_seq_no,tx_curr_cd,tx_amt,dmst_ovsea_flag_code,term_
rem_clrnet_cd,ibank_flag,tx_region_scope_cd,trbou_flag,prodt_no,rem_no,psbc_oth_bank_flag,stop_pay_no,open_acc_inst
mer_tp_cd,downstr_sys_subtx_no,downstr_sys_accti_dt,downstr_sys_serv_no,downstr_sys_or_cmpt_no,downstr_sys_resp_cd,
personal_cert_no,no_card_pay_tx_tp_cd,card_kind_cd,dcard_categ_no,agt_pers_name,agter_cert_tp_cd,agenter_cer_no,wu_
resved_input_info,span_library_flag,busi_sri_vatg_cd,pay_seq_no,unionpay_tx_dt,entrust_date,msgrpt_flag_no,busi_sta
outbound_pbs_flag,lmt_amt_dedicate_field,create_stamp,last_mod_stamp,zone_val from tb_trans_base_service_serial_6 W
core_inner_tech_sri_no = '035820210628101737102219900030701776 8' AND zone_val = '70000251032051' limit  '10001'  o
                                        QUERY PLAN
-------------------------2-------------------------------------------------------------------------------------------
 Limit  (cost=4197.10..187531.71 rows=297 width=3638)
   ->  Bitmap Heap Scan on tb_trans_base_service_serial_6  (cost=4197.10..187531.71 rows=297 width=3638)
         Recheck Cond: ((core_inner_tech_sri_no)::text = '035820210628101737102219900030701776 8'::text)
         Filter: ((zone_val)::text = '70000251032051'::text)
         ->  Bitmap Index Scan on tb_trans_base_service_serial_6_pkey  (cost=0.00..4197.02 rows=59372 width=0)
               Index Cond: ((core_inner_tech_sri_no)::text = '035820210628101737102219900030701776 8'::text)
(6 rows)

                                        QUERY PLAN
-------------------------3-------------------------------------------------------------------------------------------
 Limit  (cost=13.11..1188.22 rows=297 width=3638)
   ->  Bitmap Heap Scan on tb_trans_base_service_serial_6  (cost=13.11..1188.22 rows=297 width=3638)
         Recheck Cond: (((core_inner_tech_sri_no)::text = '035820210628101737102219900030701776 8'::text) AND
         ((zone_val)::text = '70000251032051'::text))
         ->  Bitmap Index Scan on <64957444>btree_tb_trans_base_service_serial_6_core_inner_tech_
         (cost=0.00..13.04 rows=297 width=0)
               Index Cond: (((core_inner_tech_sri_no)::text = '035820210628101737102219900030701776 8'::text) AND
```

云和恩墨
ENMOTECH

数据驱动 成就未来
Make Your Data Dance