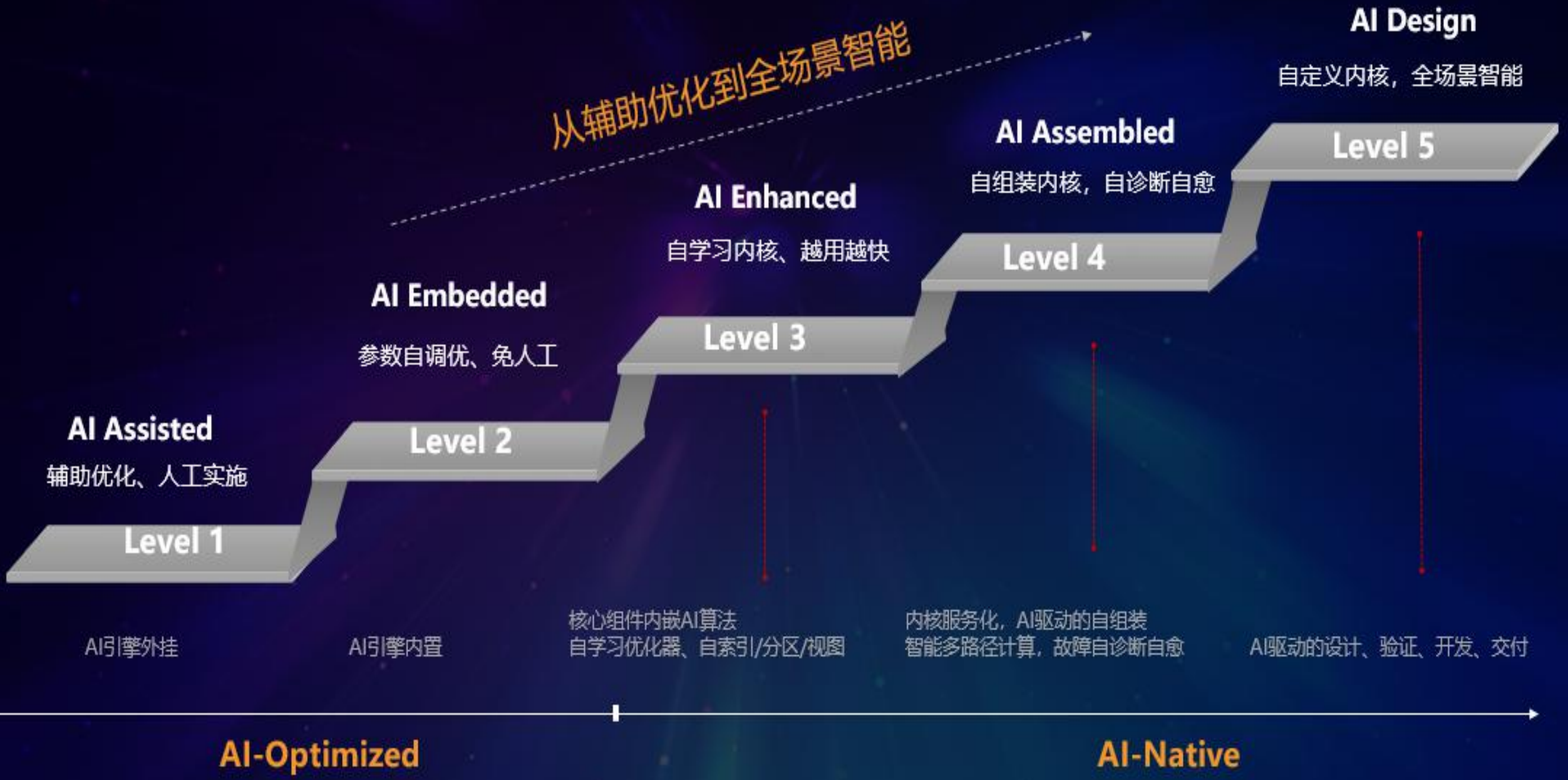
# openGauss在AI自治数据库上的演进

华为高斯实验室
王天庆

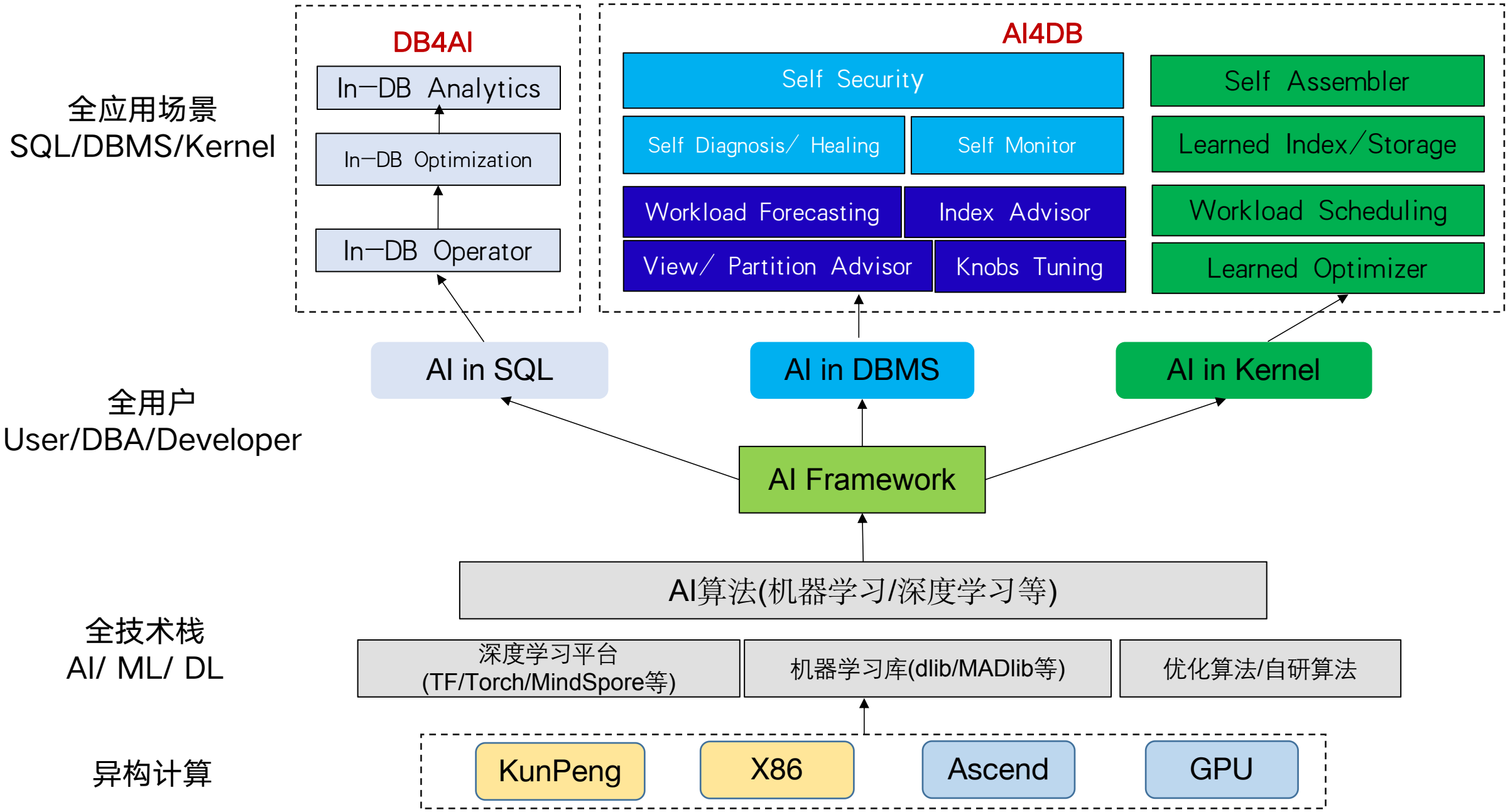
openGauss

# PART 1 AI能力整体规划

# 数据库结合人工智能的不同阶段



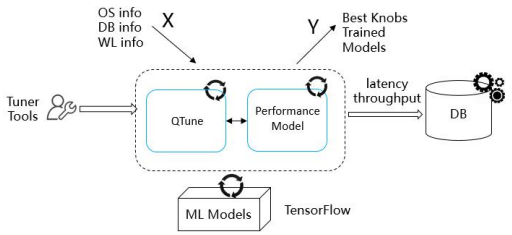
# 高斯数据库AI发展全景 - AI Framework for Database



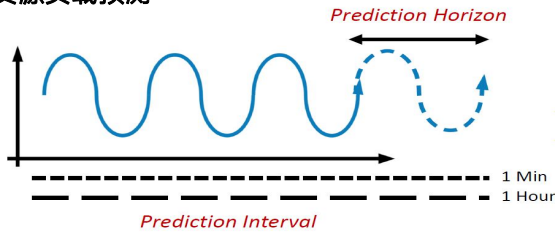
# AI in DBMS：全面提升数据库智能化水平，应对不同行业多样化负载

## 1 自调优，持续高性能

**参数自调优&推荐**

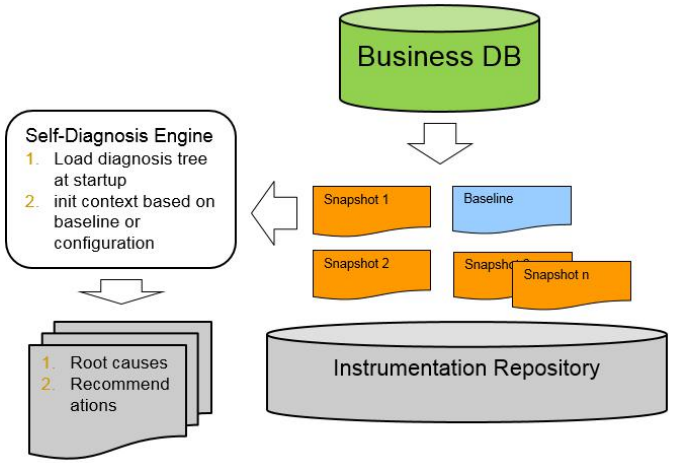



**资源负载预测**



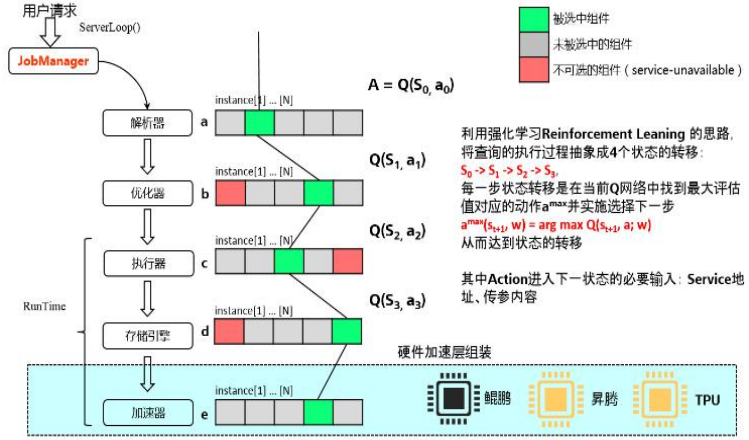

## 2 自诊断自愈，Always Online

**自愈自诊断架**




## 3 自组装，全场景数据

**One-Stack-Fit-All 自组装框架**




### 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

### 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

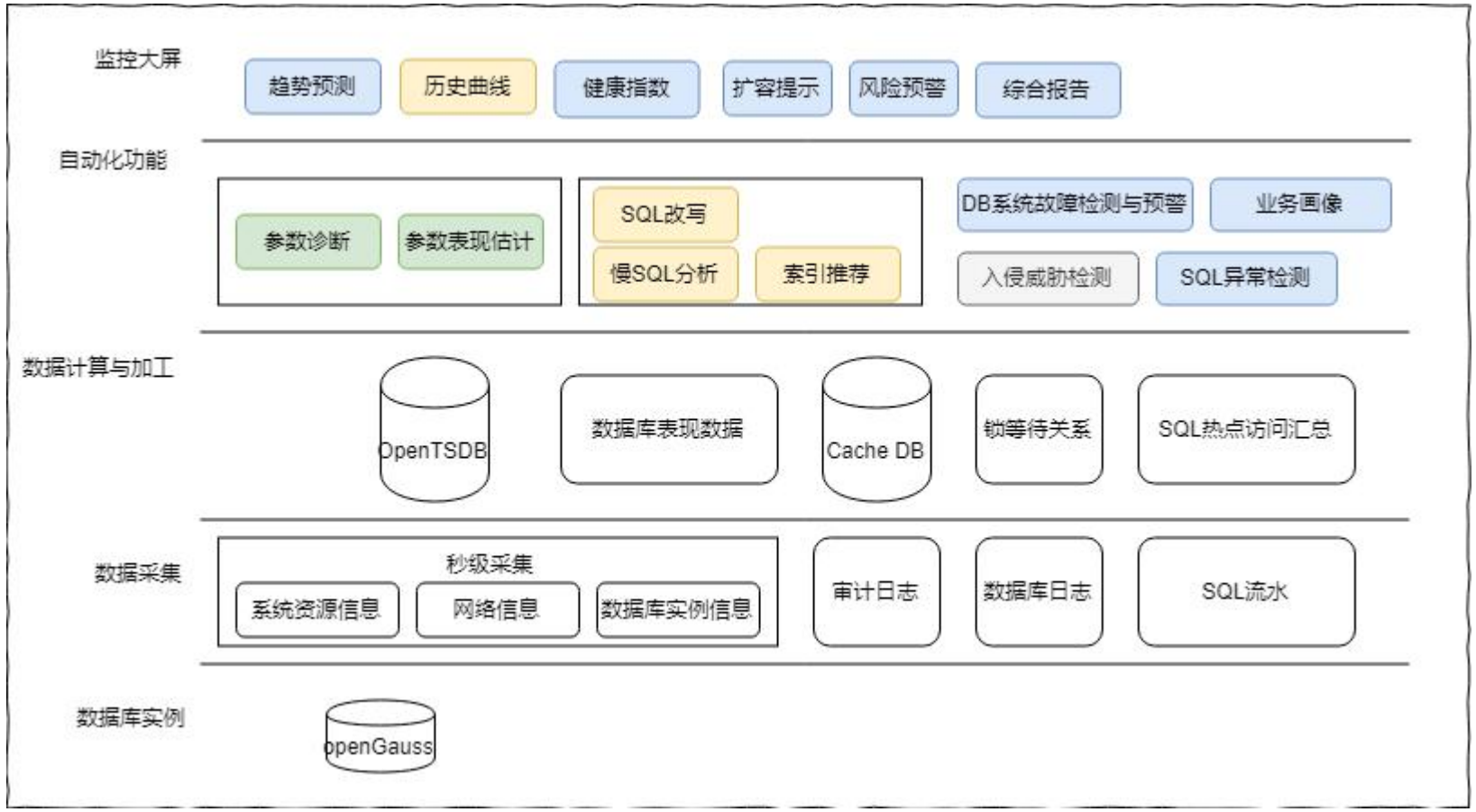
### 丰富的行业应用、支持多样化负载

- **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

# AI in DBMS：DBMind智能管理平台的能力



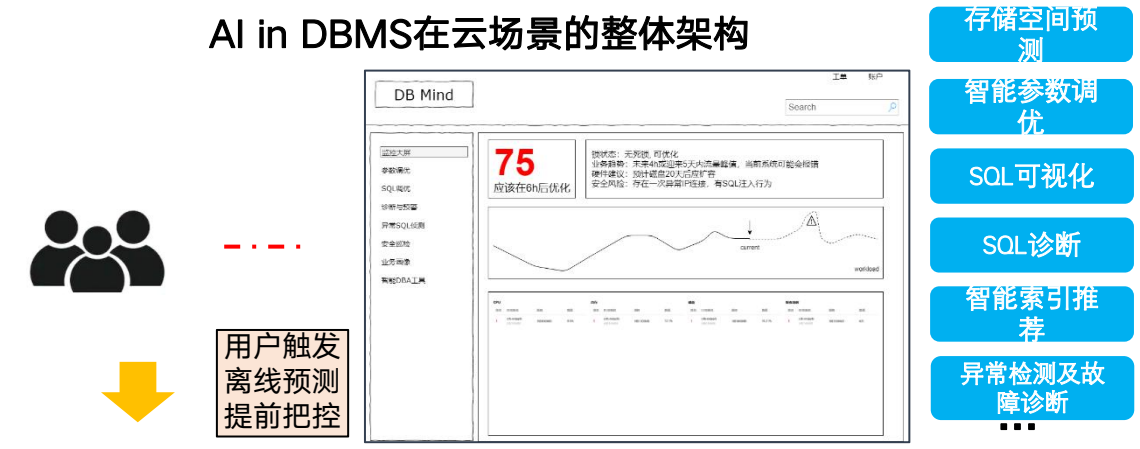

**功能列表**

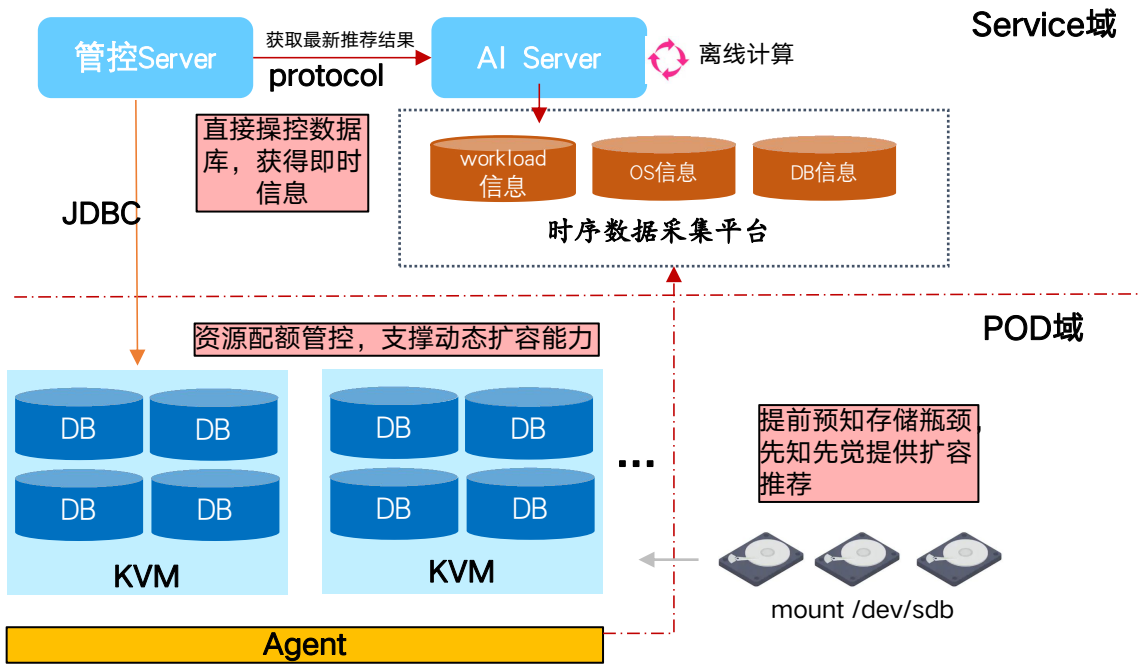
历史信息收集与可视化
SQL改写
慢SQL诊断与建议
索引推荐
参数推荐
参数表现估计
趋势预测（AI负载、性能预测）
扩容提示（AI存储空间预测）
健康指数
风险预警
综合报告
DB系统故障检测与预警
业务画像
入侵威胁检测
……

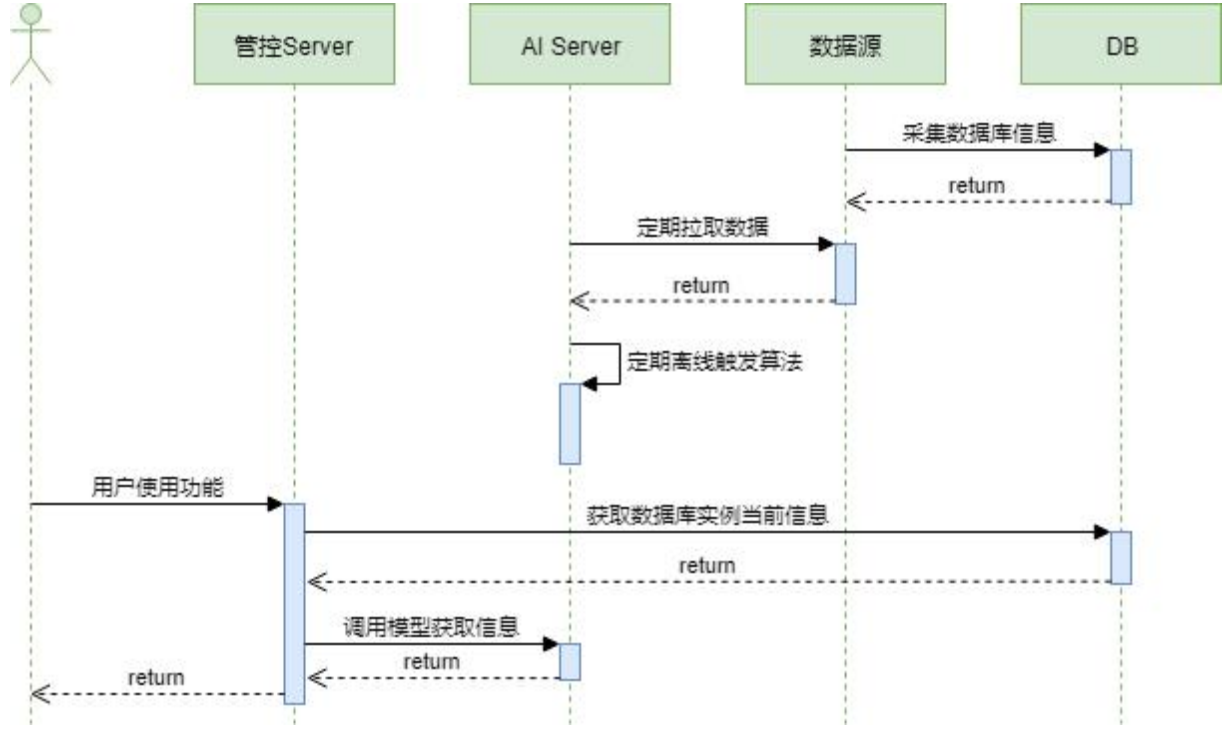
**两部分工作：**

- 数据采集与计算：获取数据信息进行分析和处理，为AI运维服务，提供用户多维度展现。
- 智能管理服务：提供精确预测、智能推荐、优化SQL功能，提升用户体验，增强粘性。

# 一种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

openGauss

# PART 2 AI能力介绍

# AI in DBMS：参数调优与诊断能力介绍

**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。

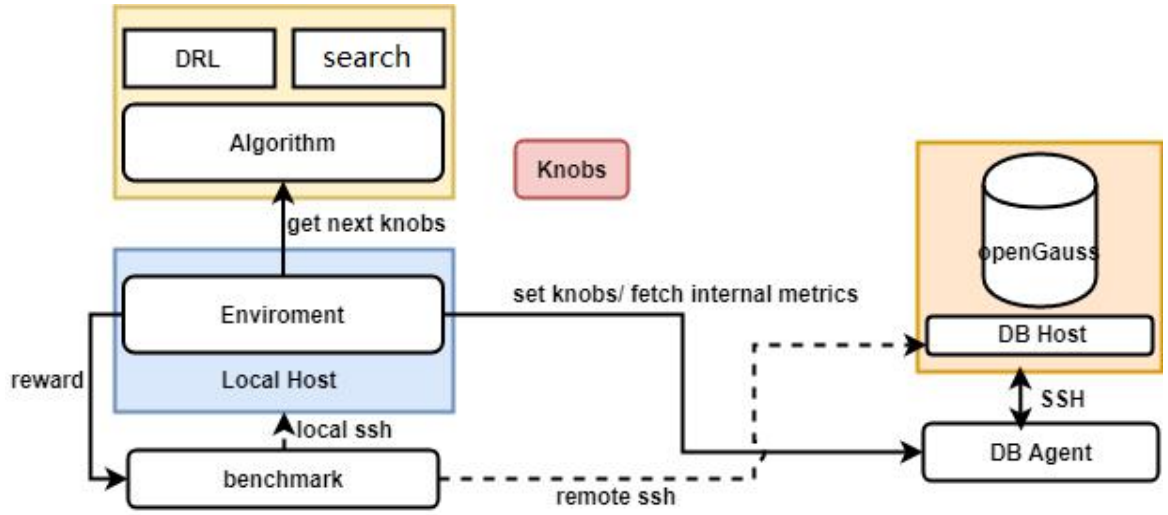
**调优效果评估：** 观察benchmark的跑分结果，预置的benchmark丰富，极简可扩展。

**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；.

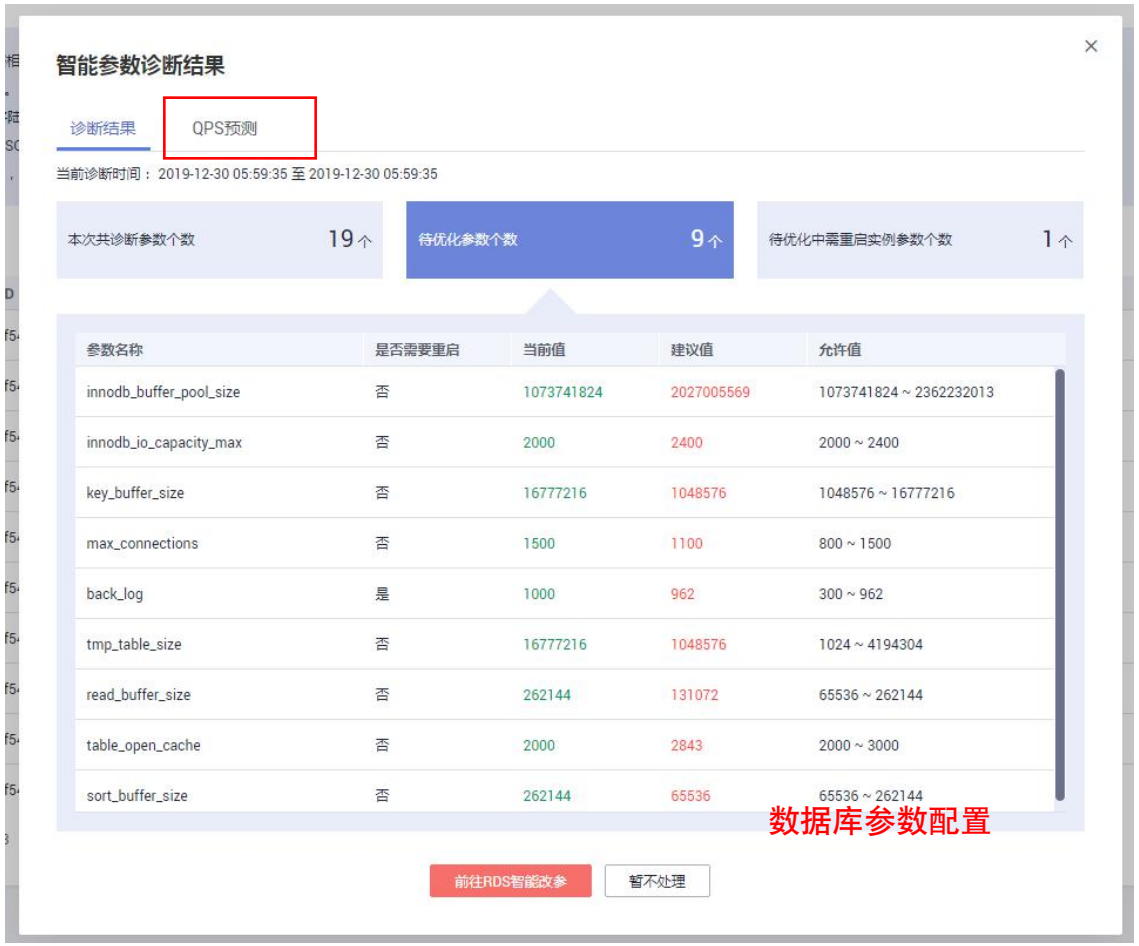
**在线参数调优流程概述：**

1. 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。




```
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  |
+---------------------------+-----------+---------+----------+---------+
```

# AI in DBMS：华为云上参数调优能力介绍（华为云DAS）



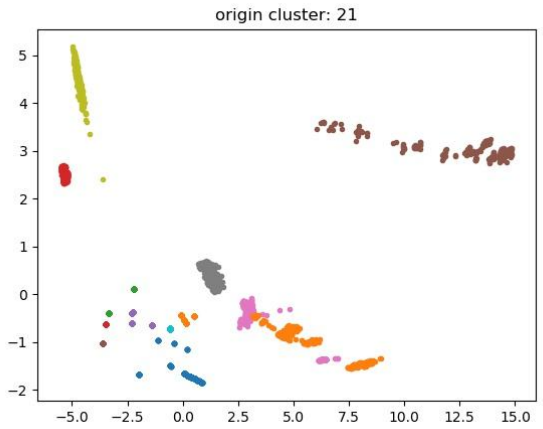

数据库参数配置

# AI in DBMS：慢SQL发现能力介绍

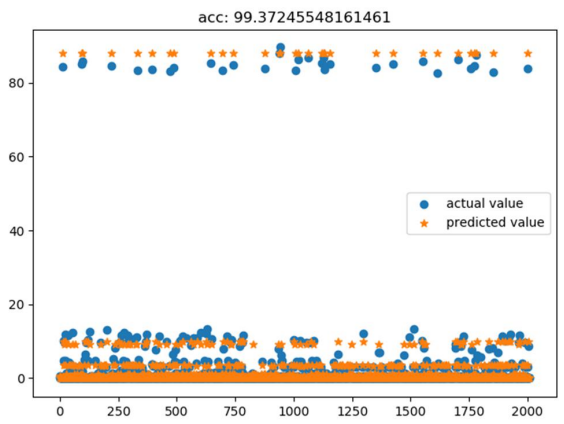
**使用场景介绍：**

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。

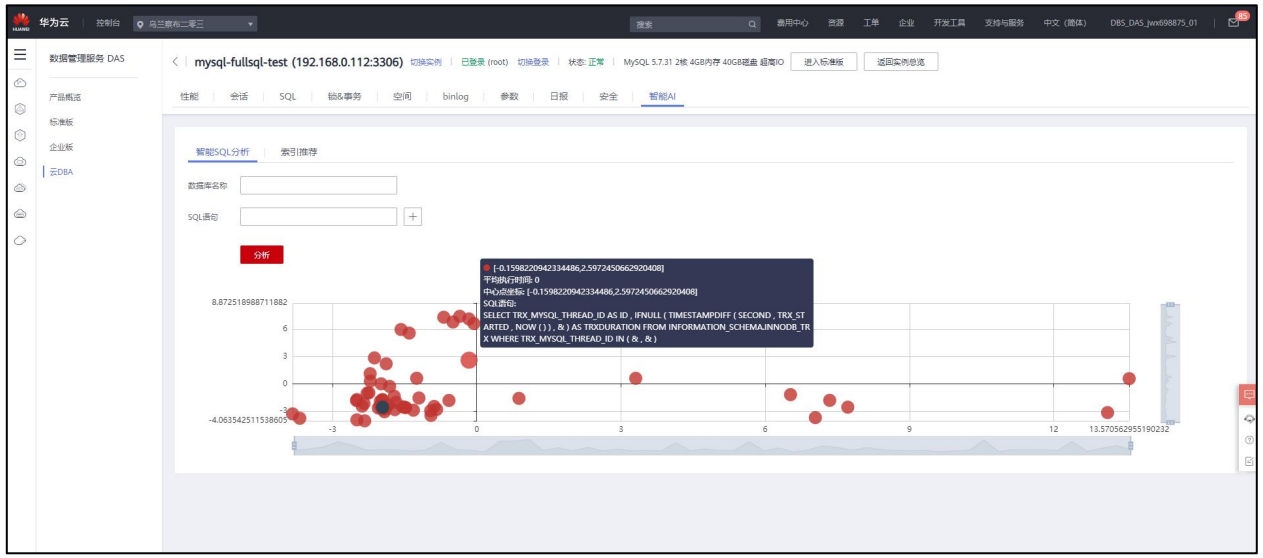
## SQL透视效果



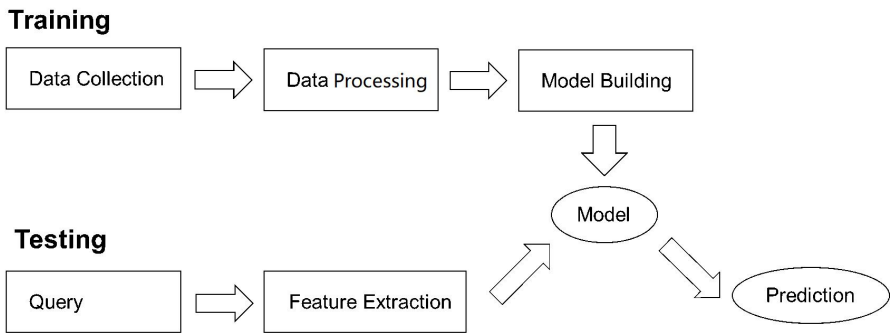

## 具备较高的预测准确率



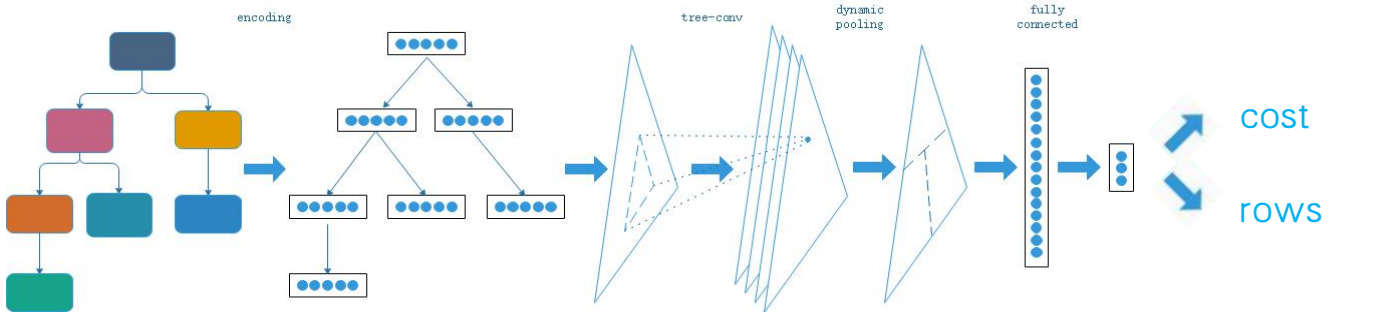

## 华为云DAS上基于海量数据训练后的SQL透视效果

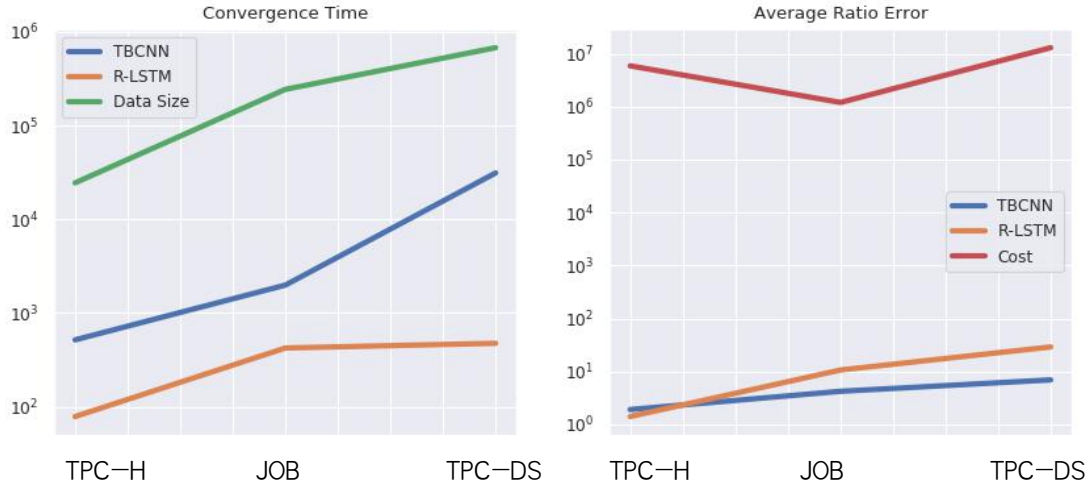


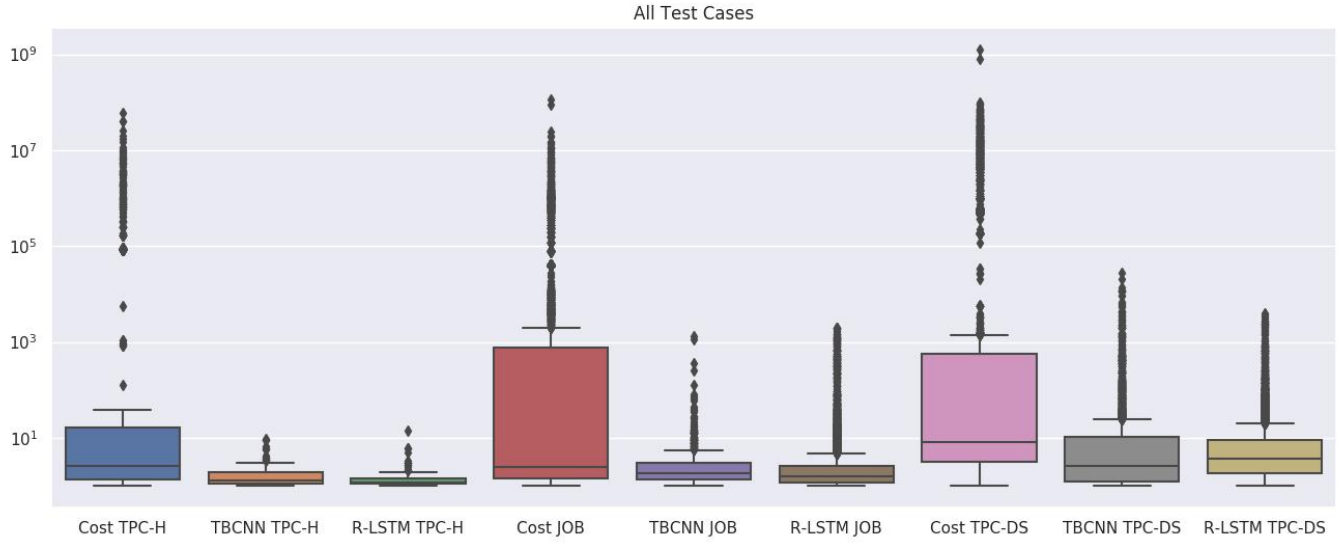

## 流程示意：



# AI in Kernel：AI优化器之基于在线学习的查询性能预测




| | TPC-H | | | JOB | | | TPC-DS | | |
|---|---|---|---|---|---|---|---|---|---|
| | Cost | TBCNN | R-LSTM | Cost | TBCNN | R-LSTM | Cost | TBCNN | R-LSTM |
| 训练耗时(s) | | 590 | 79 | | 1990 | 425 | | 31204.5 | 477 |
| 平均准确率 | 5.90E+06 | 1.9(1.8) | 1.4(1.3) | 1.20E+06 | 4.2(4.0) | 10.6(4.1) | 1.30E+07 | 6.9(4.3) | 28.9(7.8) |
| <10准确率 | 74% | 99% | 99% | 62% | 93% | 91% | 53% | 74% | 74% |
| <2准确率 | 42% | 77% | 91% | 43% | 53% | 66% | 12% | 44% | 29% |

**实验结论：**

Cost为使用代价线性拟合预测时间，拟合参数通过历史数据回归获得。

深度学习模型比Cost线性模型在预测准确率上有显著提升。

R-LSTM模型收敛速度比TBCNN模型显著提升，但在较复杂场景下(如TPC-DS)容易过拟合。

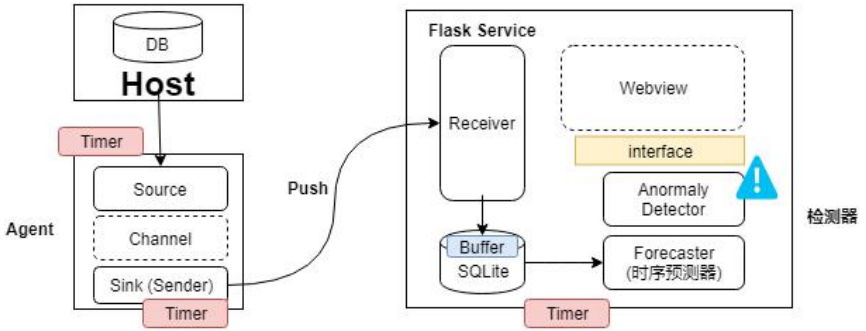
TBCNN在简单场景下和R-LSTM准确率比较接近，在复杂场景下表现出较强的泛化能力，但训练速度过慢。

# AI in DBMS：数据库监控与异常检测

**数据库指标**是数据库与用户行为健康的重要标志，数据库中的异常行为可能导致数据库指标产生异常，因此对指标进行有效的监控显得十分必要。

**数据库状态监控**，指对数据库运行指标进行全方位实时监控。系统能够发现和识别数据库异常以及潜在的性能问题，并及时将数据库异常报告给用户，通过针对各项运行指标的统计分析报告，帮助管理员、运维人员、决策者多视角了解数据库的运行状态，从而更好的应对数据库的需求及规划。

## 1 A-Detection 架构示意图





## 2 各个子模块的说明

Agent由Source、Channel以及Sink组成。部署在数据库环境上的，用于采集数据库中的性能指标，并通过网络，将其传送给远端检测器模块。

Detector由数据收集Agent推送的数据，并将数据存储在本地磁盘，同时Detector基于时序预测和异常检测等算法对数据库指标进行监控和异常检测。

Receiver：数据接收器，接受Agent.HttpSink数据。同时将接受的数据推送到储存模块。

储存模块：对Receiver中的数据进行储存，目前主要支持数据库方式储存，默认是SQLite数据库。

算法模块：该模块是Detector的核心模块，实现了时序预测、异常检测、告警推送等功能。同时用户可以按照接口要求实现自定义的算法。

```
[l  .n@linux173 anomaly_detection]$ python main.py forecast --metric-name disk_space --forecast-periods 30M
+-------------+---------------------------------------+--------------------+--------------------+--------------------+
| Metric name |               Date range              |       Minimum      |       Maximum      |       Average      |
+-------------+---------------------------------------+--------------------+--------------------+--------------------+
|  disk_space | 2020-11-24 11:41:53~2020-11-24 12:11:52 | 102.35646415444381 | 105.60445111891163 | 103.83065385316687 |
+-------------+---------------------------------------+--------------------+--------------------+--------------------+
[l   .n@linux173 anomaly_detection]$ python main.py forecast --metric-name disk_space --forecast-periods 1H
+-------------+---------------------------------------+--------------------+--------------------+--------------------+
| Metric name |               Date range              |       Minimum      |       Maximum      |       Average      |
+-------------+---------------------------------------+--------------------+--------------------+--------------------+
|  disk_space | 2020-11-24 11:41:53~2020-11-24 12:41:52 | 102.35646415444381 | 110.55916605998928 | 105.89202214450336 |
+-------------+---------------------------------------+--------------------+--------------------+--------------------+
[likun@linux173 anomaly_detection]$ python main.py forecast --metric-name disk_space --forecast-periods 1D
+-------------+---------------------------------------+--------------------+--------------------+--------------------+
| Metric name |               Date range              |       Minimum      |       Maximum      |       Average      |
+-------------+---------------------------------------+--------------------+--------------------+--------------------+
|  disk_space | 2020-11-24 11:41:53~2020-11-25 11:41:52 | 102.35646415444381 | 306.9882691991314  | 211.11922600898393 |
+-------------+---------------------------------------+--------------------+--------------------+--------------------+
[l.   .@linux173 anomaly_detection]$
```

# AI in Kernel：单Query/ Workload级别索引推荐能力介绍

**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，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。

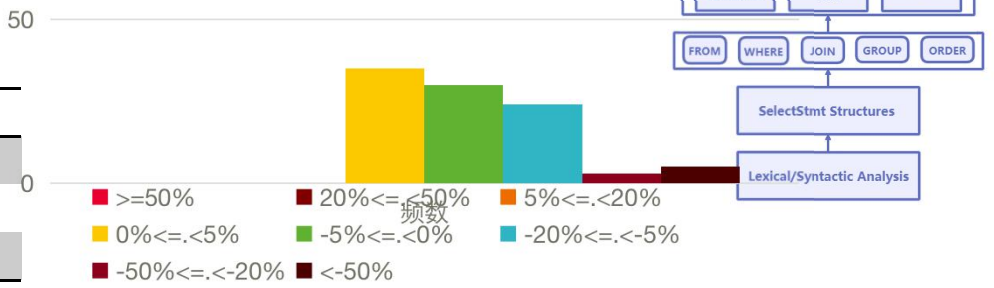
```
tpch=# explain analyze select * from lineitem where l_orderkey < 100 and l_suppkey > 50;
                                                                                  QUERY PLAN
-----------------------------------------------------------------------------------------------------------------------------------------------------------
 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：除40%左右无明显变化外，其余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




TPC-DS

## 端到端索引推荐的原理细节

**单Query索引推荐的核心方法：**

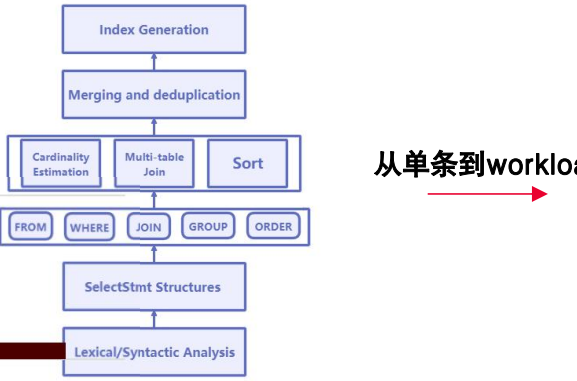
采用索引设计和优化的相关理论，基于原生的词法和语法解析，对查询语句中的子句和谓词进行分析和处理，再结合字段选择度、聚合条件、多表Join关系等输出最终建议。

**索引性能验证的方法：**

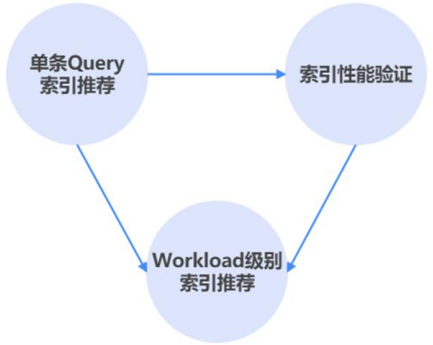
通过修改优化器相应的数据结构，利用**优化器评估**，进而判断创建该索引后，对优化器生成执行计划的影响。该过程可以不用真正创建索引，即业内所谓的“**假设索引**”，业内也多采用此种方法。

**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。




从单条到workload



# AI in SQL：全流程AI，构筑数据库内端到端DB4AI系统，无需人工参与

**关键组件说明：**

1．**自动特征工程**：从相关的数据表中自动提取有用且有意义的特征，减少了特征工程所需的时间。将周级的处理时间缩短至天级。

2．**模型最优选择**：在多个模型中，选择出最适合本数据的模型。

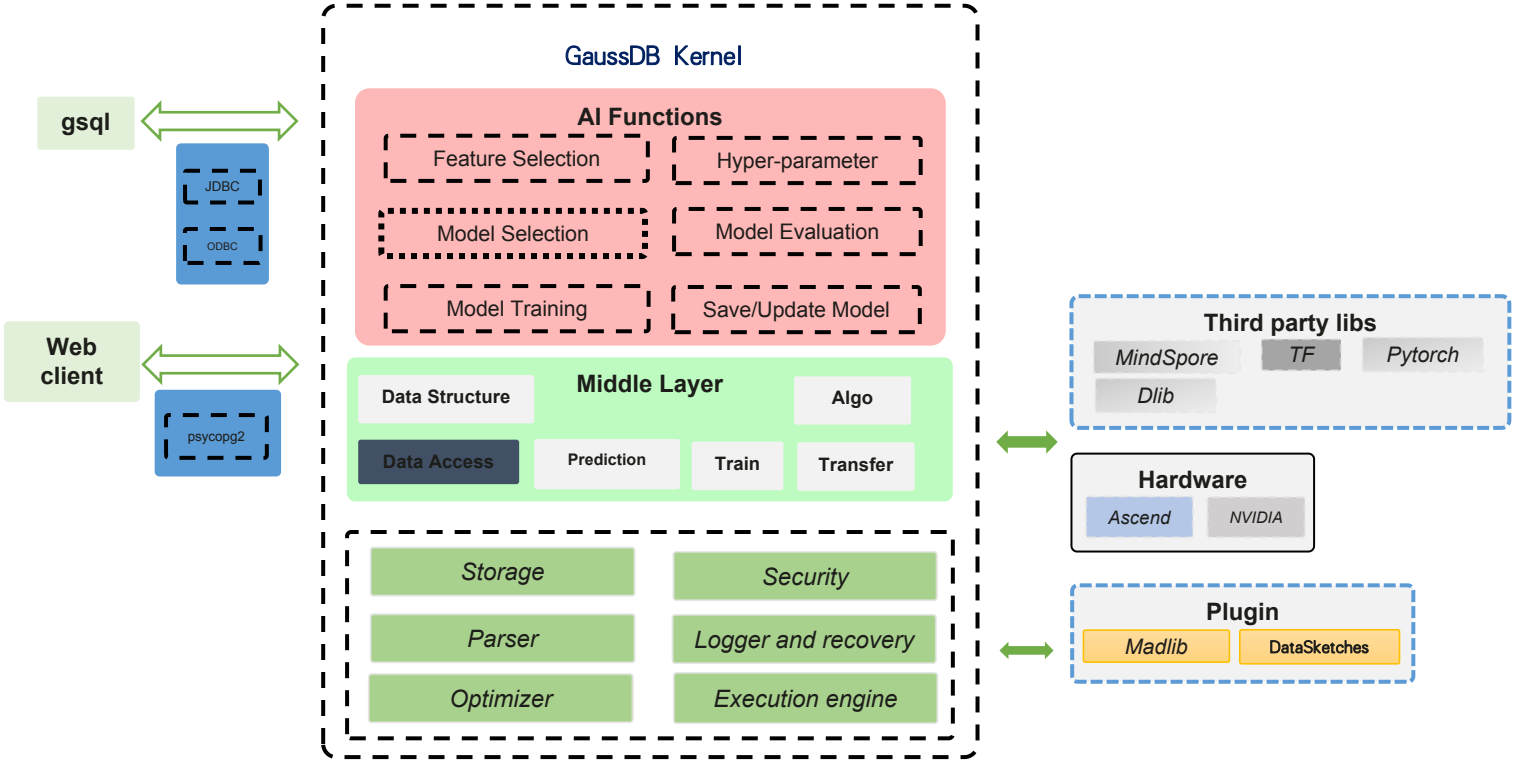
3．**超参数优化**：选出合适模型后，并且能够设定好它的最优参数。

4．**优化算法选择**：自动地选择出一个优化算法（如SGD、L-BFGS、GD等），以便能够达到效率和精度的平衡。

5．**支持深度学习**：支持主流深度学习框架。并自动地选择出一个优化算法，以便能够达到效率和精度的平衡。

6．**模型管理**：记录模型准确度等相关信息。增加模型生命周期管理，支持增量训练，模型更新，支持模型导入和导出。

兼容PostgreSQL生态的MADlib，相比原生MADlib on PG性能提升10~20%，支持更多客户需求的算法，如时序预测Prophet、XGBoost、梯度提升树（GBDT）等。



# THANKS!

谢谢观看