MAKE YOUR DATABASE DREAM OF ELECTRIC SHEEP DESIGNING FOR AUTONOMOUS OPERATION
Part #1 - Background
Part #2 - Engineering
Part #3 - Oracle Rant
1970-1990s Self-Adaptive Databases

- Index Selection
- Partitioning / Sharding
- Data Placement
1970-1990s Self-Adaptive Databases

Admin

SELECT * FROM A JOIN B
  ON A.ID = B.ID
  WHERE A.VAL > 123
  AND B.NAME LIKE 'XY%'

Self-Adaptive Databases
1970-1990s Self-Adaptive Databases

SELECT * FROM A JOIN B
ON A.ID = B.ID
WHERE A.VAL > 123
AND B.NAME LIKE 'XY%'

Admin
Tuning Algorithm
Self-Adaptive Databases

1970-1990s

Query:

```
SELECT *
FROM A JOIN B
ON A.ID = B.ID
WHERE A.VAL > 123
AND B.NAME LIKE 'XY%'
```
AUTONOMOUS DBMSs
SELF-ADAPTIVE DATABASES

1970-1990s
Self-Adaptive Databases

SELECT * FROM A JOIN B
ON A.ID = B.ID
WHERE A.VAL > 123
AND B.NAME LIKE 'XY%'

Tuning Algorithm

Admin

A.ID
A.VAL
B.ID
B.NAME

+100
+200
+50
SELECT * FROM A JOIN B
ON A.ID = B.ID
WHERE A.VAL > 123
AND B.NAME LIKE 'XY%'

Admin

Tuning Algorithm

1970-1990s Self-Adaptive Databases

+100 +200 +50
**AUTONOMOUS DBMSs**

**SELF-ADAPTIVE DATABASES**

---

**1970-1990s**

Self-Adaptive Databases

→ Index Selection

→ Partitioning / Sharding

→ Data Placement

---

**INDEX SELECTION IN A SELF-ADAPTIVE DATA BASE MANAGEMENT SYSTEM**

Michael H. Werner

Department of Computer Science, MIT, Cambridge, Massachusetts, USA

---

**INTRODUCTION**

The efficient utilization of a data base is highly dependent on the optimal matching of the physical organization of the data base to the access requirements and other program needs. The physical design problem involves the creation of a physical structure of the data base which is both efficient and effective for execution. A physical design involves identifying the data base characteristics and evaluating strategies for data base organization. The purpose of this paper is to present a self-adaptive data base management system that makes changes to the physical design of the data base based on the current use of the data base. The self-adaptive system is capable of making changes to the data base that are necessary for efficient execution.
1990-2000s Self-Tuning Databases

AUTONOMOUS DBMSs
SELF-TUNING DATABASES

SELECT * FROM A JOIN B
ON A.ID = B.ID
WHERE A.VAL > 123
AND B.NAME LIKE 'XY%

→ Index Selection
→ Partitioning / Sharding
→ Data Placement
SELECT * FROM A JOIN B ON A.ID = B.ID WHERE A.VAL > 123 AND B.NAME LIKE 'XY%'

1990-2000s Self-Tuning Databases

Microsoft AutoAdmin

Optimizer Cost Model

Tuning Algorithm

Admin
1990-2000s Self-Tuning Databases

**ABSTRACT**

In the paper, we describe the evolution of self-tuning database systems over the past two decades, reflecting the advances in database research. This paper provides insights into the evolution of self-tuning algorithms, their applications, and the challenges faced by self-tuning algorithms.

1. **History of Autonomous Systems**

   In 1990, Microsoft Research created the first self-tuning database system called *AutoAdmin*. The system was designed to optimize the performance of queries by automatically tuning the execution plans. In the 1990s, several other research groups also worked on self-tuning algorithms. For example, IBM developed a self-tuning system called *Tuning Algorithm*, which was used to optimize the performance of database queries.

2. **An Introduction to Physical Design**

   The main goal of physical design is to determine the optimal storage and retrieval methods for the database. Physical design involves choosing the appropriate storage structures and indexes to improve query execution time and memory usage.

   **2.1 Importance of Physical Design**

   Physical design is crucial for the efficient execution of database queries. A good physical design can significantly improve the performance of database systems. Physical design involves decisions such as choosing the appropriate storage structures, indexing strategies, and buffer management techniques.

   **2.2 State of the Art in 1999**

   In 1999, the state of the art in physical design was characterized by the following trends: (a) the increasing importance of query processing, (b) the need for efficient data management, and (c) the importance of query optimization.

   **2.3 Recent Advances**

   Recent advances in physical design include the development of new storage structures, indexing techniques, and buffer management algorithms. These advances have significantly improved the performance of database systems.

**SELECT**

* FROM A
JOIN B
ON A.ID = B.ID
WHERE A.VAL > 123
AND B.NAME LIKE 'XY%'

Tuning Algorithm

Admin

MicroAuto

VLDB 2007
Automated DBMSs: Self-Tuning Databases

1990-2000s Self-Tuning Databases

Number of Knobs

→ Knob Configuration
AUTONOMOUS DBMSs
CLOUD MANAGED DATABASES

2010s
Cloud Databases
AUTONOMOUS DBMSs
CLOUD MANAGED DATABASES

2010s Cloud Databases
AUTONOMOUS DBMSs
CLOUD MANAGED DATABASES

2010s
Cloud Databases

→ Initial Placement
→ Tenant Migration
Why is this previous work insufficient?
Problem #1
Human Judgements

Problem #2
Reactionary Measures
What is different this time?
Better hardware. 
Better machine learning tools. 
Better appreciation for data. 

We seek to complete the circle in autonomous databases.
OtterTune
Existing Systems

Peloton
New System
Database Tuning-as-a-Service
→ Automatically generate DBMS knob configurations.
→ Reuse data from previous tuning sessions.

OtterTune
ottertune.cs.cmu.edu

Supported Systems
CONTROLLER

COLLECTOR

TARGET DATABASE
```
mysql> SHOW GLOBAL STATUS;
+--------------------------+----------+
|       METRIC_NAME        |   VALUE |
|--------------------------+----------|
| ABORTED_CLIENTS          | 0        |
| ABORTED_CONNECTS         | 0        |
| INNODB_BUFFER_POOL_BYTES_DATA | 129499136 |
| INNODB_BUFFER_POOL_BYTES_DIRTY | 76070912  |
| INNODB_BUFFER_POOL_PAGES_DATA | 7904     |
| INNODB_BUFFER_POOL_PAGES_DIRTY | 4643     |
| INNODB_BUFFER_POOL_PAGES.Flushed | 25246   |
| INNODB_BUFFER_POOL_PAGES_FREE  | 0    |
| INNODB_BUFFER_POOL_PAGES_MISC | 288     |
| INNODB_BUFFER_POOL_PAGES>Total | 8192 |
| INNODB_BUFFER_POOL_READS    | 15327   |
| INNODB_BUFFER_POOL_READ_AHEAD | 0       |
| INNODB_BUFFER_POOL_READ_AHEAD_EVICT | 0       |
| INNODB_BUFFER_POOL_READ_AHEAD_RND | 0       |
| INNODB_BUFFER_POOL_READ_REQUESTS | 2604302 |
| INNODB_BUFFER_POOL_WAIT_FREE | 0       |
| INNODB_BUFFER_POOL_WRITE_REQUESTS | 562763 |
| INNODB_DATA_FSYNCS         | 2836    |
| INNODB_DATA_PENDING_FSYNCS | 1       |
| INNODB_DATA_WRITES         | 28026   |
+--------------------------+----------+
```
```
mysql> SHOW GLOBAL STATUS;
+-------------------------+------------+
|            NAME         |      VALUE |
|-------------------------+------------|
| INNODB_BUFFER_POOL_BYTES_DATA | 129499136  |
| INNODB_BUFFER_POOL_BYTES_DIRTY   | 76070912   |
| INNODB_BUFFER_POOL_PAGES_DATA   | 7904       |
| INNODB_BUFFER_POOL_PAGES_DIRTY  | 4643       |
| INNODB_BUFFER_POOL_PAGES_FLUSHED | 25246     |
| INNODB_BUFFER_POOL-pages_FREE   | 0          |
| INNODB_BUFFER_POOL-pages_MISC   | 288        |
| INNODB_BUFFER_POOL_pages_TOTAL  | 8192       |
| INNODB_BUFFER_POOL_READS        | 15327      |
| INNODB_BUFFER_POOL_READ_AHEAD   | 0          |
| INNODB_BUFFER_POOL_READ_AHEAD_EVICT | 0       |
| INNODB_BUFFER_POOL_READ_AHEAD_RND | 0        |
| INNODB_BUFFER_POOL_READ_REQUESTS | 2604302   |
| INNODB_BUFFER_POOL_WAIT_FREE    | 0          |
| INNODB_BUFFER_POOL_WRITE_REQUESTS | 562763    |
| INNODB_DATA_FSYNCS             | 2836       |
| INNODB_DATA_PENDING_FSYNCS     | 1          |
| INNODB_DATA_WRITES             | 28026      |
| UPTIME                      | 5996       |
| UPTIME_SINCE_FLUSH_STATUS    | 5996       |
+-------------------------+------------+
```
mysql> SHOW GLOBAL VARIABLES;

<table>
<thead>
<tr>
<th>KNOB_NAME</th>
<th>KNOB_VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>AUTOCOMMIT</td>
<td>ON</td>
</tr>
<tr>
<td>AUTOMATIC_SP_PRIVILEGES</td>
<td>ON</td>
</tr>
<tr>
<td>INNODB_BUFFER_POOL_SIZE</td>
<td>134217728</td>
</tr>
<tr>
<td>INNODB_CHANGE_BUFFERING</td>
<td>all</td>
</tr>
<tr>
<td>INNODB_FLUSH_LOG_AT_TRX_COMMIT</td>
<td>1</td>
</tr>
<tr>
<td>INNODB_FLUSH_METHOD</td>
<td></td>
</tr>
<tr>
<td>INNODB_FORCE_LOAD_CORRUPTED</td>
<td>OFF</td>
</tr>
<tr>
<td>INNODB_FORCE_RECOVERY</td>
<td>0</td>
</tr>
<tr>
<td>INNODB_IO_CAPACITY</td>
<td>200</td>
</tr>
<tr>
<td>INNODB_LARGE_PREFIX</td>
<td></td>
</tr>
<tr>
<td>INNODB_LOCKSUnsafe_FOR_BINLOG</td>
<td>OFF</td>
</tr>
<tr>
<td>INNODB_LOCK_WAIT_TIMEOUT</td>
<td>500</td>
</tr>
<tr>
<td>INNODB_LOG_BUFFER_SIZE</td>
<td>8388608</td>
</tr>
<tr>
<td>INNODB_LOG_FILES_IN_GROUP</td>
<td>2</td>
</tr>
<tr>
<td>INNODB_LOG_FILE_SIZE</td>
<td>5242880</td>
</tr>
<tr>
<td>SORT_BUFFER_SIZE</td>
<td>2097152</td>
</tr>
<tr>
<td>SQL_AUTO_IS_NULL</td>
<td>OFF</td>
</tr>
<tr>
<td>TIMED_MUTEXES</td>
<td>OFF</td>
</tr>
<tr>
<td>VERSION_COMPILE_OS</td>
<td>debian-linux-gn</td>
</tr>
<tr>
<td>WAIT_TIMEOUT</td>
<td>28800</td>
</tr>
</tbody>
</table>
### SHOW GLOBAL VARIABLES

<table>
<thead>
<tr>
<th>KNOB_NAME</th>
<th>KNOB_VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>AUTOCOMMIT</td>
<td>ON</td>
</tr>
<tr>
<td>AUTOMATIC_SP_PRIVILEGES</td>
<td>ON</td>
</tr>
<tr>
<td>INNODB_Buffer_Pool_Size</td>
<td>134217728</td>
</tr>
<tr>
<td>INNODB_CHANGE_BUFFERING</td>
<td>all</td>
</tr>
<tr>
<td>INNODB_FLUSH_LOG_AT_TRX_COMMIT</td>
<td>1</td>
</tr>
<tr>
<td>INNODB_FLUSH_METHOD</td>
<td>OFF</td>
</tr>
<tr>
<td>INNODB_FORCE_LOAD_CORRUPTED</td>
<td>OFF</td>
</tr>
<tr>
<td>INNODB_FORCE_RECOVERY</td>
<td>0</td>
</tr>
<tr>
<td>INNODB_IO_CAPACITY</td>
<td>200</td>
</tr>
<tr>
<td>INNODB_LARGE_PREFIX</td>
<td>OFF</td>
</tr>
<tr>
<td>INNODB_LOCKSUnsafe_For_BINLOG</td>
<td>OFF</td>
</tr>
<tr>
<td>INNODB_LOCK_WAIT_TIMEOUT</td>
<td>500</td>
</tr>
<tr>
<td>INNODB_LOG_BUFFER_SIZE</td>
<td>8388608</td>
</tr>
<tr>
<td>INNODB_LOG_FILES_IN_GROUP</td>
<td>2</td>
</tr>
<tr>
<td>INNODB_LOG_FILE_SIZE</td>
<td>5242880</td>
</tr>
<tr>
<td>SORT_BUFFER_SIZE</td>
<td>2097152</td>
</tr>
<tr>
<td>SQL_AUTO_IS_NULL</td>
<td>OFF</td>
</tr>
<tr>
<td>TIMED_MUTEXES</td>
<td>OFF</td>
</tr>
<tr>
<td>VERSION_COMPILE_OS</td>
<td>debian-linux-gn</td>
</tr>
<tr>
<td>WAIT_TIMEOUT</td>
<td>28800</td>
</tr>
</tbody>
</table>
OTTERTUNE
AUTOMATIC DBMS TUNING SERVICE

TARGET DATABASE

CONTROLLER

COLLECTOR

TUNING MANAGER

TensorFlow

Internal Repository

Configuration Recommender

Metric Analyzer

Knob Analyzer

Internal Repository
OTTERTUNE
AUTOMATIC DBMS TUNING SERVICE

CONTROLLER

COLLECTOR

INSTALL AGENT

TUNING MANAGER

TensorFlow

Configuration Recommender

Metric Analyzer

Knob Analyzer

Internal Repository

TARGET DATABASE

INSTALL AGENT
Demonstration
Postgres v9.3
TPC-C Benchmark
Throughput (txn/sec)

- **MySQL**
  - Default: 165
  - Scripts: 508
  - RDS: 562
  - DBA: 736
  - OtterTune: 686

- **PostgreSQL**
  - Default: 426
  - Scripts: 845
  - RDS: 714
  - DBA: 843
  - OtterTune: 946
Self-Driving Database System

→ In-memory DBMS with integrated ML/RL framework.

→ Designed for autonomous operations.

Peloton
pelotondb.io
PELOTON
THE SELF-DRIVING DBMS

TARGET DATABASE

WORKLOAD HISTORY
PELOTON
THE SELF-DRIVING DBMS

WORKLOAD HISTORY

TARGET DATABASE

FORECAST MODELS
PELOTON
THE SELF-DRIVING DBMS

WORKLOAD HISTORY

TARGET DATABASE

FORECAST MODELS

Search Tree

ACTION CATALOG

ACTION SEQUENCE

"THE BRAIN"

TensorFlow
PELTON
THE SELF-DRIVING DBMS

"THE BRAIN"

TARGET DATABASE

WORKLOAD HISTORY

FORECAST MODELS

Search Tree

ACTION CATALOG

ACTION SEQUENCE

TensorFlow
PELOTON
THE SELF-DRIVING DBMS

"THE BRAIN"

WORKLOAD HISTORY

TARGET DATABASE

FORECAST MODELS

ACTION SEQUENCE

Search Tree

ACTION CATALOG

TensorFlow
PELOTON
BUS TRACKING APP WITH ONE-HOUR HORIZON

QUERY-BASED WORKLOAD FORECASTING FOR SELF-DRIVING
DATABASE MANAGEMENT SYSTEM
SIGMOD 2018

Ensemble (LR+RNN)

Actual  Predicted

Queries Per Hour

9-Jan  11-Jan  13-Jan  15-Jan  17-Jan

0  15000  30000  45000  60000
PELOTON
ADMISSIONS APP WITH THREE-DAY HORIZON

- Actual
- Predicted

**Ensemble (LR+RNN)**

Queries Per Hour

Ensemble (LR+RNN)
PELOTON
ADMISSIONS APP WITH THREE-DAY HORIZON

### Queries Per Hour

**Ensemble (LR+RNN)**

- **Actual**
- **Predicted**

<table>
<thead>
<tr>
<th>Date</th>
<th>Actual</th>
<th>Predicted</th>
</tr>
</thead>
<tbody>
<tr>
<td>26-Nov</td>
<td></td>
<td></td>
</tr>
<tr>
<td>30-Nov</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4-Dec</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8-Dec</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12-Dec</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16-Dec</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Millions**

- **Queries Per Hour**

- **Figures**

- **Dates**
  - 26-Nov
  - 30-Nov
  - 4-Dec
  - 8-Dec
  - 12-Dec
  - 16-Dec
PELOTON
ADMISSIONS APP WITH THREE-DAY HORIZON

- **Actual**
- **Predicted**

**Ensemble (LR+RNN)**

**Hybrid (LR+RNN+KR)**

Queries Per Hour

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>5</th>
<th>10</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Millions</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>26-Nov</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>30-Nov</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4-Dec</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8-Dec</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12-Dec</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16-Dec</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Ensemble (LR+RNN) and Hybrid (LR+RNN+KR) predictions compared to actual values. Red arrows indicate significant predictions.
Let's on check the demo...
Design Considerations for Autonomous Operation
AUTONOMOUS DBMS
DESIGN CONSIDERATIONS

Configuration Knobs

Internal Metrics

Action Engineering
Anything that requires a human value judgement should be marked as off-limits to autonomous components.

– *File Paths*
– *Network Addresses*
– *Durability / Isolation Levels*
The autonomous components need hints about how to change a knob

– *Min/max ranges.*
– *Separate knobs to enable/disable a feature.*
– *Non-uniform deltas.*
The autonomous components need hints about how to change a knob

- *Min/max ranges.*
- *Separate knobs to enable/disable a feature.*
- *Non-uniform deltas.*
The autonomous components need hints about how to change a knob:

- Min/max ranges.
- Separate knobs to enable/disable a feature.
- Non-uniform deltas.
Indicate which knobs are constrained by hardware resources.

– *The sum of all buffers cannot exceed the total amount of available memory.*

The problem is that sometimes it makes sense to overprovision.
Expose DBMS's hardware capabilities:

- CPU, Memory, Disk, Network

<table>
<thead>
<tr>
<th>id</th>
<th>model</th>
<th>mhz</th>
<th>cache</th>
<th>boqomips</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Intel Xeon E7-4830</td>
<td>3472.718</td>
<td>4096 KB</td>
<td>5615.84</td>
</tr>
<tr>
<td>1</td>
<td>Intel Xeon E7-4830</td>
<td>3472.718</td>
<td>4096 KB</td>
<td>5615.84</td>
</tr>
<tr>
<td>2</td>
<td>Intel Xeon E7-4830</td>
<td>3472.718</td>
<td>4096 KB</td>
<td>5615.84</td>
</tr>
<tr>
<td>3</td>
<td>Intel Xeon E7-4830</td>
<td>3472.718</td>
<td>4096 KB</td>
<td>5615.84</td>
</tr>
<tr>
<td>4</td>
<td>Intel Xeon E7-4830</td>
<td>3472.718</td>
<td>4096 KB</td>
<td>5615.84</td>
</tr>
<tr>
<td>5</td>
<td>Intel Xeon E7-4830</td>
<td>3472.718</td>
<td>4096 KB</td>
<td>5615.84</td>
</tr>
<tr>
<td>6</td>
<td>Intel Xeon E7-4830</td>
<td>3472.718</td>
<td>4096 KB</td>
<td>5615.84</td>
</tr>
<tr>
<td>7</td>
<td>Intel Xeon E7-4830</td>
<td>3472.718</td>
<td>4096 KB</td>
<td>5615.84</td>
</tr>
</tbody>
</table>
Expose DBMS's hardware capabilities: 
– CPU, Memory, Disk, Network

Otherwise you have to come up with clever ways to approximate this...
If the DBMS has sub-components that are tunable, then it must expose separate metrics for those components.

Bad Example: RocksDB
RocksDB Column Family Knobs

```
rocksdb_override_cf_options=
  cf_link_pk={prefix_extractor=capped:20}
```

Column Family Metrics

```
mysql> SELECT * FROM INFORMATION_SCHEMA.ROCKSDB_CFSTAT;
+-----------+-----------------+----------+
<table>
<thead>
<tr>
<th>CF_NAME</th>
<th>METRIC_NAME</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>default</td>
<td>COMPACTION_PENDING</td>
<td>1</td>
</tr>
<tr>
<td>default</td>
<td>CUR_SIZE_ACTIVE_MEM_TABLE</td>
<td>21672</td>
</tr>
<tr>
<td>default</td>
<td>CUR_SIZE_ALL_MEM_TABLES</td>
<td>21672</td>
</tr>
<tr>
<td>default</td>
<td>MEM_TABLE_FLUSH_PENDING</td>
<td>0</td>
</tr>
<tr>
<td>default</td>
<td>NON_BLOCK_CACHE_SST_MEM_USAGE</td>
<td>0</td>
</tr>
<tr>
<td>default</td>
<td>NUM_ENTRIES_ACTIVE_MEM_TABLE</td>
<td>18</td>
</tr>
<tr>
<td>default</td>
<td>NUM_ENTRIES_IMM_MEM_TABLES</td>
<td>0</td>
</tr>
<tr>
<td>default</td>
<td>NUM_IMMUTABLE_MEM_TABLE</td>
<td>0</td>
</tr>
<tr>
<td>default</td>
<td>NUM_LIVE_VERSIONS</td>
<td>2</td>
</tr>
</tbody>
</table>
```
INTERNAL METRICS

SUB-COMPONENTS

RocksDB Column Family Knobs

```c
rocksdb_override_cf_options=
    cf_link_pk={prefix_extractor=capped:20}
```

Global Metrics

```sql
mysql> SHOW GLOBAL STATUS;
+----------------+----------------+--------+
| METRIC_NAME     | VALUE           | STATUS |
+----------------+----------------+--------+
| ABORTEDCLIENTS | 0               |        |
|                 |                 |        |
| ROCKSDB_BLOCKCACHEBYTES_READ | 295700537      |        |
| ROCKSDB_BLOCKCACHEBYTES_WRITE  | 709562185      |        |
| ROCKSDB_BLOCKCACHEDATA_HIT     | 64184          |        |
| ROCKSDB_BLOCKCACHEDATA_MISS    | 1001083        |        |
| ROCKSDB_BYTES_READ            | 5573794        |        |
| ROCKSDB_BYTES_WRITTEN         | 5817440        |        |
| ROCKSDB_FLUSH_WRITE_BYTES     | 2906847        |        |
| UPTIME_SINCE_FLUSH_STATUS     | 5996           |        |
+----------------+----------------+--------+
```
No action should ever require the DBMS to restart in order for it to take affect. The commercial systems are much better than this than the open-source systems.
Provide a notification callback to indicate when an action starts and when it completes.

Harder for changes that can be used before the action completes.
Support executing the same action with different resource usage levels.
Allow replica configurations to diverge from each other.
Allow replica configurations to diverge from each other.
Allow replica configurations to diverge from each other.
Allow replica configurations to diverge from each other.
Allow replica configurations to diverge from each other.
What About Oracle's Self-Driving DBMS?
World’s First “Self-Driving” Database

Oracle Autonomous Database

No Human Labor – Half the Cost
No Human Error – 100x More Reliable

ORACLE

oracle.com/selfdrivindb

September 2017

Self-Driving Database Management Systems

Andrew Pavlo, Guvanan Arumugan, Jawahar, Hailen Lin, Jiejie Lin, Lin Ma, Prasanth Menon, Todd C. Milan, Matthew Pernier, Ian Pugh, Siddharta Rajurkar, Anthony Tomasic, Shyake Tsiir, Dana Van Aken, Zhi Wang, Yingjun Wu, Fan Xian, Tiebing Zhang

Carnegie Mellon University, National University of Singapore

ABSTRACT

In the last two decades, both researchers and vendors have built auxiliary tools to assist database administrators (DBAs) in various aspects of system tuning and physical design. Most of these tools have been designed to make the third decision about any changes to the database and are reactive, meaning that they get invoked after they occur. What is needed is a truly “self-driving” database management system (DBMS) that can autonomously take appropriate action, without any human intervention, to optimize itself in accordance with the overall business goals and constraints. Oracle’s Autonomous Database (AD) is designed to be a self-driving database that is able to continuously monitor the health of the system and take the necessary action to keep the system running optimally. This paper describes the efforts by Oracle to build an AD that guides the user through the system setup process to a state where DBAs can no longer be required. The AD’s machine learning approach is based on autonomic computing principles. The AD’s machine learning approach is based on autonomic computing principles. The AD’s machine learning approach is based on autonomic computing principles.

1. INTRODUCTION

The idea of a DBMS on a chip on the horizon of data management vision was not in the original planning phase of the traditional model and declarative query language from the 1970s [12]. With this approach, a database with a query that specifies what data they want to see. The ADMS can take the most efficient way to view the data in question, which is generally, a combination of the query data, the context of the situation, and the data store available to the database at the chip level to order to optimize the performance of the database. The ADMS can also take the most efficient way to view the data in question, which is generally, a combination of the query data, the context of the situation, and the data store available to the database at the chip level to order to optimize the performance of the database.

This paper presents the architecture of the ADMS, which is designed to work in conjunction with the database system to provide a seamless, end-to-end solution. The ADMS is designed to be a self-driving database that is able to continuously monitor the health of the system and take the necessary action to keep the system running optimally. This paper describes the efforts by Oracle to build an ADMS that guides the user through the system setup process to a state where DBAs can no longer be required. The ADMS’s machine learning approach is based on autonomic computing principles. The ADMS’s machine learning approach is based on autonomic computing principles. The ADMS’s machine learning approach is based on autonomic computing principles.

2. PROBLEM DESCRIPTION

The first challenge in a self-driving DBMS is to understand an application’s workload. The most hard part is to understand the workload on the application DBMS, which can tell us the database needs of a workload, and how they can be deployed for a better performance. The ADMS is designed to be a self-driving database that is able to continuously monitor the health of the system and take the necessary action to keep the system running optimally. This paper describes the efforts by Oracle to build an ADMS that guides the user through the system setup process to a state where DBAs can no longer be required. The ADMS’s machine learning approach is based on autonomic computing principles. The ADMS’s machine learning approach is based on autonomic computing principles. The ADMS’s machine learning approach is based on autonomic computing principles.

Much of the previous work on self-driving systems is focused on the problem of selecting and deploying a self-driving database. Oracle’s Autonomous Database (AD) is designed to be a self-driving database that is able to continuously monitor the health of the system and take the necessary action to keep the system running optimally. This paper describes the efforts by Oracle to build an ADMS that guides the user through the system setup process to a state where DBAs can no longer be required. The ADMS’s machine learning approach is based on autonomic computing principles. The ADMS’s machine learning approach is based on autonomic computing principles. The ADMS’s machine learning approach is based on autonomic computing principles.

Much of the previous work on self-driving systems is focused on the problem of selecting and deploying a self-driving database. Oracle’s Autonomous Database (AD) is designed to be a self-driving database that is able to continuously monitor the health of the system and take the necessary action to keep the system running optimally. This paper describes the efforts by Oracle to build an ADMS that guides the user through the system setup process to a state where DBAs can no longer be required. The ADMS’s machine learning approach is based on autonomic computing principles. The ADMS’s machine learning approach is based on autonomic computing principles. The ADMS’s machine learning approach is based on autonomic computing principles.
Automatic Indexing
Automatic Recovery
Automatic Scaling
Automatic Query Tuning
Automatic Indexing
Automatic Recovery
Automatic Scaling
Automatic Query Tuning

Problem #2
Reactionary Measures

Oracle Autonomous Database
No Human Labor – Half the Cost
No Human Error – 100x More Reliable

oracle.com/selfdrivingdb

September 2017
Automatic Indexing
Automatic Recovery
Automatic Scaling
Automatic Query Tuning

Problem #2
Reactionary Measures

Oracle Autonomous Database
No Human Labor – Half the Cost
No Human Error – 100x More Reliable

September 2017
Automatic Indexing
Automatic Recovery
Automatic Scaling
Automatic Query Tuning

Microsoft  IBM  DB2  INGRES

Problem #2
Reactionary Measures

Oracle Autonomous Database
No Human Labor – Half the Cost
No Human Error – 100x More Reliable

September 2017
True autonomous DBMSs are achievable in the next decade.

You should think about how each new feature can be controlled by a machine.
Demo Results
END

@andy_pavlo