数据库技术在IoT场景的应用

公益讲座11：00准时开始，请大家先浏览云技术微信公众号技术文章。资料会在各群同步发布，已入群客户请勿重复入群！

20-22

数据库和云讲座群

甲骨文云技术公众号

B站专家系列课程

* 活动最终解释权归甲骨文公司所有

基于 Oracle 数据库
免费企业数据健康检查

- 及时了解数据库健康状况，发现并解决潜在问题
- 维护数据库系统良好状态，保护数据资产的安全
- 提升数据库性能、稳定性和安全性，降低业务风险

免费咨询热线：
400-699-8888

数据库技术
在 IoT 场景的应用

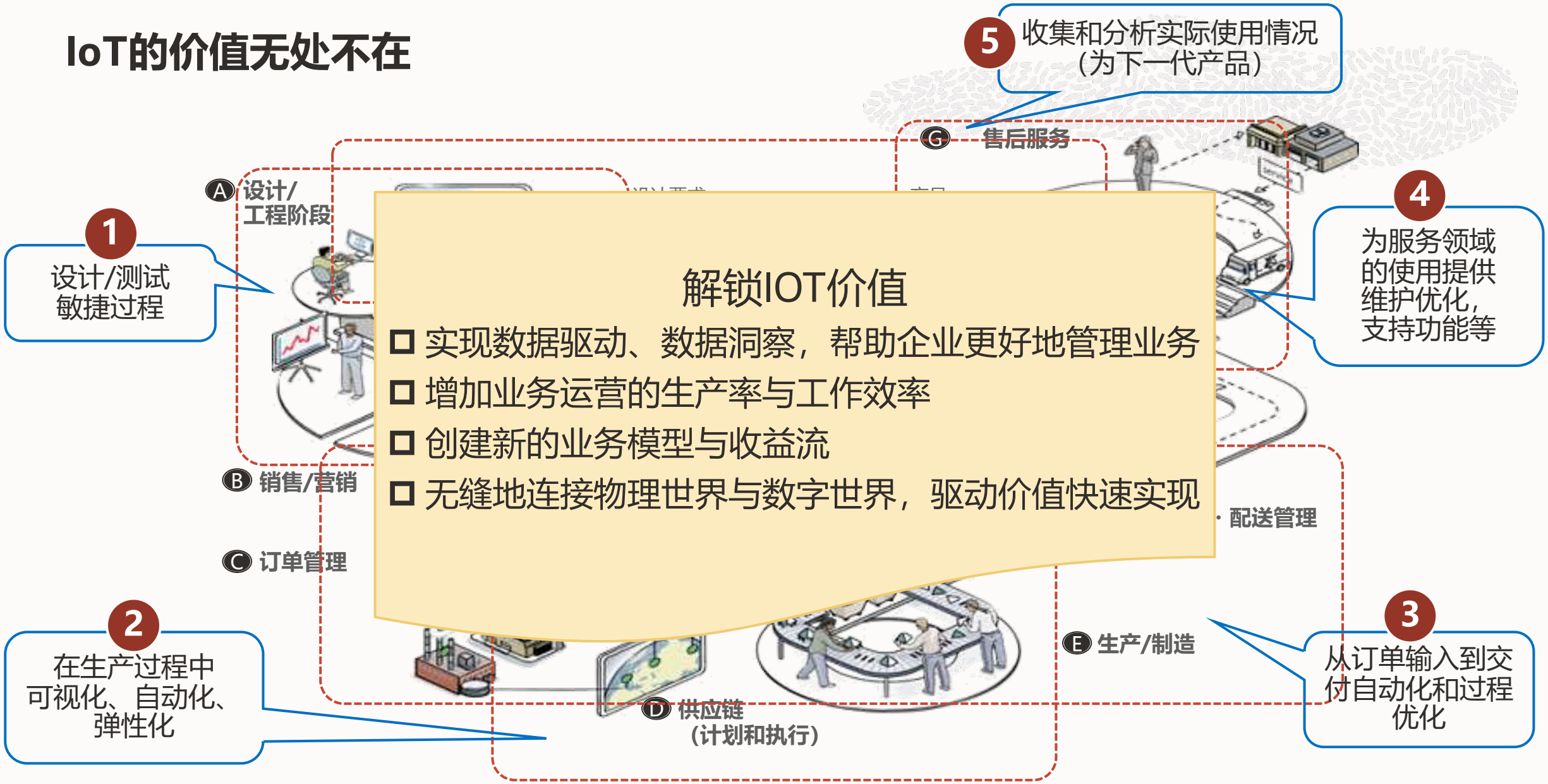
甲骨文技术公益课 - 数据库专场

2023 年 11 月 03 日 11:00

线上直播

张锋

IoT的价值无处不在



IoT工作负载有哪些特征

超高频，但小或轻量级事务

一般包含复杂数据类型，例如空间、非结构化数据等

存在丢失数据的可能性，但又不影响整体业务运行

ACID并非强制要求

数据模型在不断优化和进化中

对数据库而言，从技术角度，IoT真的是一个全新事物吗?

超高吞吐量并非一个全新的概念

Oracle数据库已经在全球大量行业中使用了几十年，处理各种复杂数据工况

- 某电信企业：每天处理650亿笔交易；每天新增数据超过18TB
- 某国土局：每天处理30亿笔交易；每天处理320亿笔查询；总数据量超过1PB
- 某交易所：每天处理10亿笔交易；每天新增数据超过15TB

IoT负载对数据库提出了什么关键需求?

性能与可扩展性

- 能够按比例地满足快速增加的设备数量以及数据量，而维持体验不变

灵活性

- 能够适应由新的数据类型引发的必要的数据模型迭代

实时分析

- 能够实现快速的数据清洗、整合、分析，而无需昂贵的、长时间的ETL流程

性能与可扩展性

如果你要买的东西很多，你是愿意买一样东西交一次钱呢？还是买全了东西再一次性交钱？

Demo1:

传统的数据加载机制效果怎么样?

如果简单的批量提交，效率会有什么变化?

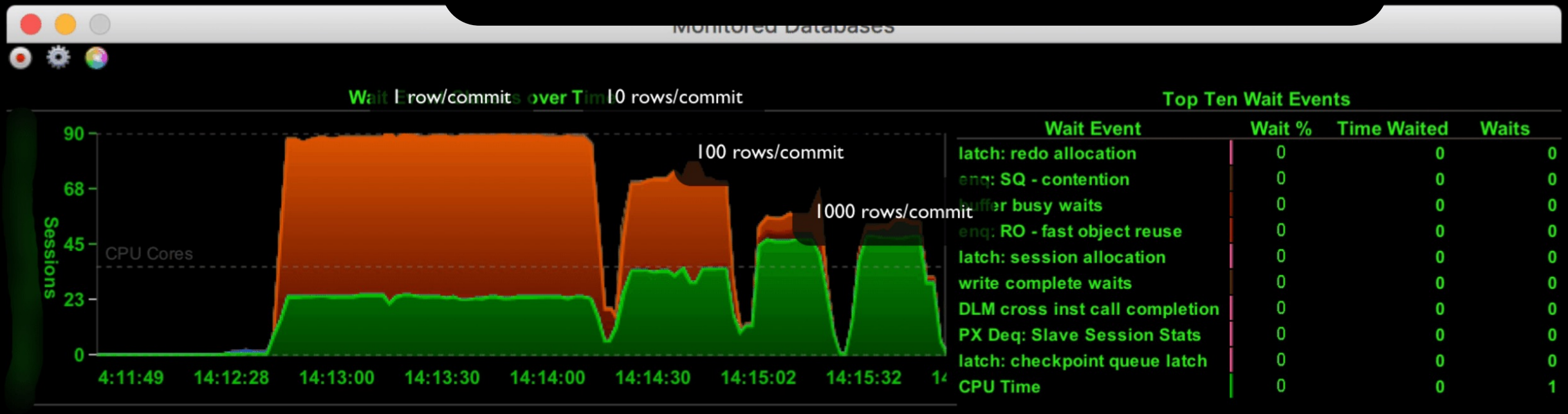
```
(myvirtualenv) [opc@domias OraIngestTests]$ python runtests.py -u anpr -p anpr -cs [redacted] -st light -com 1,10,10
000 -bat 1 -tc 8 -proc 11 -scale 5 -rs clean_up.sql -ss
```

```
(myvirtualenv) [opc@domias OraIngestTests]$ python runtests.py -u anpr -p anpr -cs                                    -st light -com 1,10,100 -bat 1 -tc 8 -proc 11 -scale 5 -rs clean_up.sql -ss
Running script clean_up.sql
Tests Run: 100%|████████████████████████████████████████████████████████████████████████████████████████████████| 4/4 [03:07<00:00, 52.22s/
```

JVMs Started	Thread Count	Commit Size	Batch Size	Image Size	Async	Total Rows Inserted	Real Time Taken	Total Insert Time	Rows/sec Inserted
11	8	1	1	100	False	5500000	94.66	995.00	59,805
11	8	10	1	100	False	5500000	36.06	362.60	163,593
11	8	100	1	100	False	5500000	29.98	281.40	202,109
11	8	1000	1	100	False	5500000	27.18	267.01	220,529

```
(myvirtualenv) [opc@domias OraIngestTests]$
```

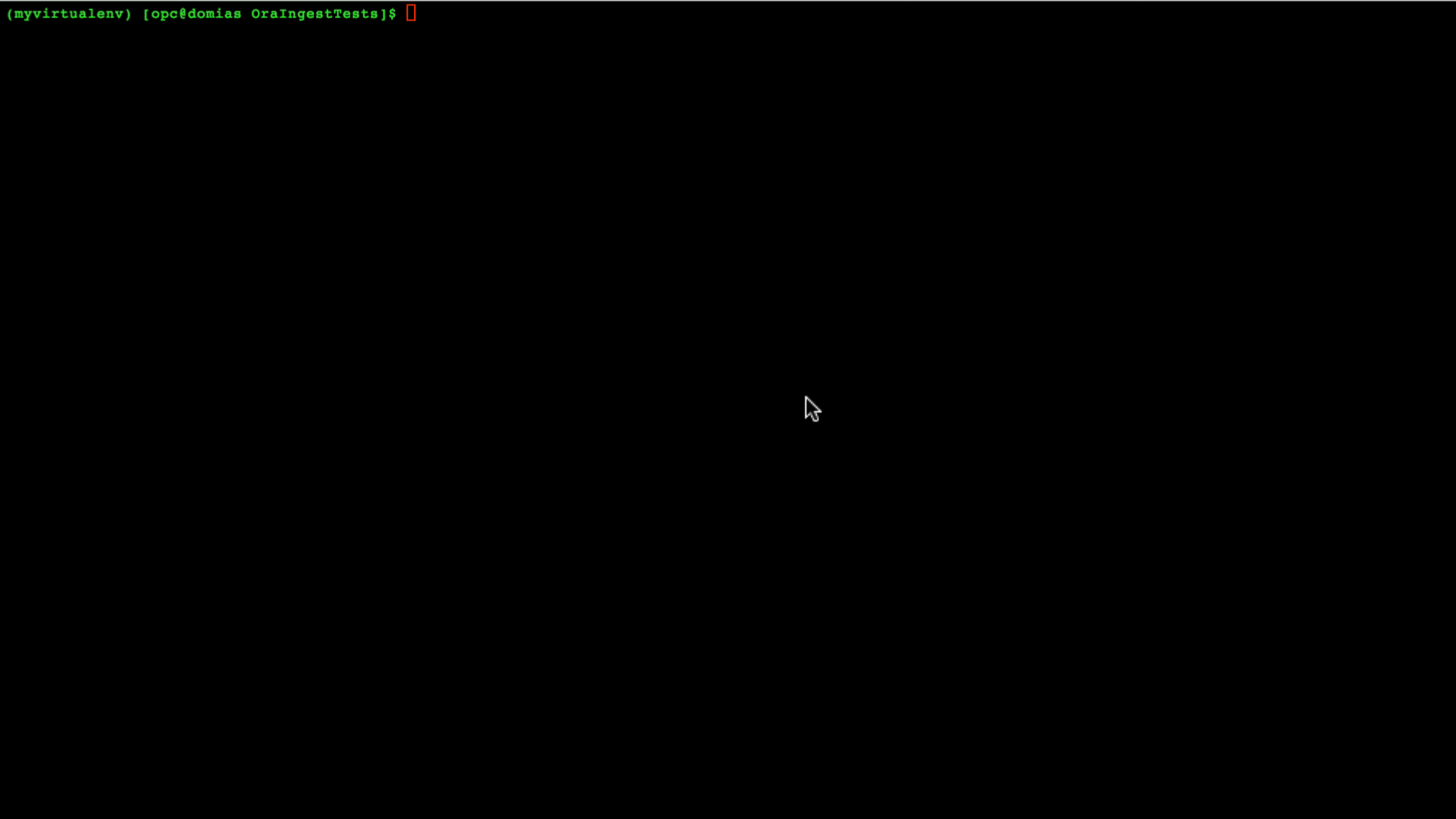
简单地批量提交，吞吐量就能提升若干倍



注意：结果是在特定场景下测试所得，并不代表最大处理能力，仅供参考。

Demo2:

数组批量插入的效果会怎么样?

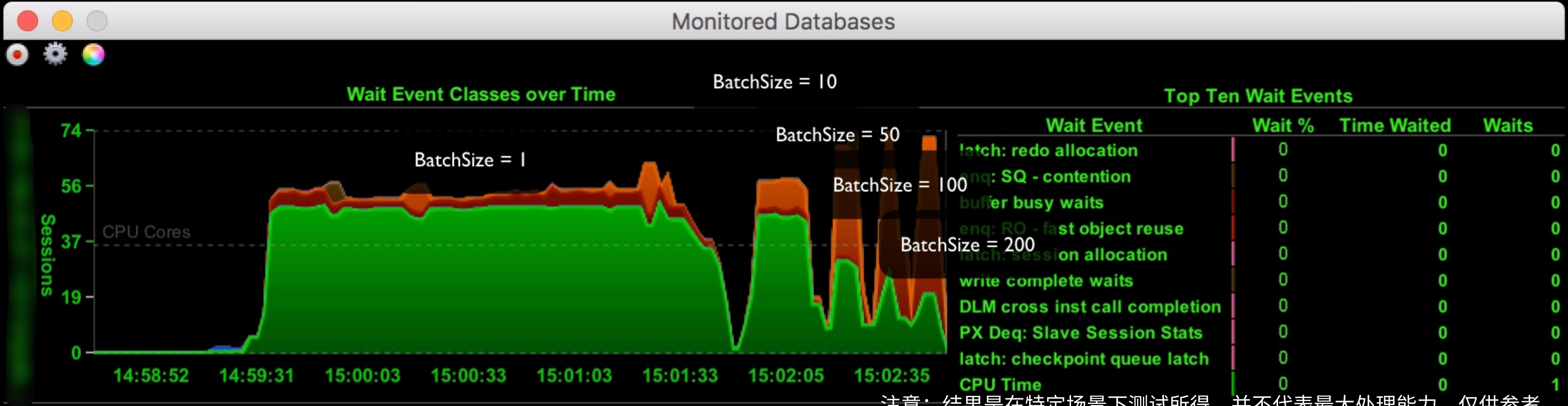

(myvirtualenv) [opc@domias OraIngestTests]$

```
(myvirtualenv) [opc@domias OraIngestTests]$ python runtests.py -u anpr -p anpr -cs [redacted] -st light -com 1000 -b
1,10,50,100,200 -tc 8 -proc 11 -scale 30 -rs clean_up.sql -ss
Running script clean_up.sql
Tests Run: 100%|████████████████████████████████████████| 5/5 [03:12<00:00, 42.26s/
```

JVMs Started	Thread Count	Commit Size	Batch Size	Image Size	Async	Total Rows Inserted	Real Time Taken	Total Insert Time	Rows/sec Inserted
11	8	1000	1	100	False	33000000	135.71	1,445.32	247,413
11	8	1000	10	100	False	33000000	22.37	211.52	1,680,843
11	8	1000	50	100	False	33000000	12.06	102.45	3,460,208
11	8	1000	100	100	False	33000000	11.13	91.25	3,844,810
11	8	1000	200	100	False	33000000	10.87	86.96	3,750,000

```
(myvirtualenv) [opc@domias OraIngestTests]$
```

如果改成数组方式，吞吐量又能提升若干倍



注意：结果是在特定场景下测试所得，并不代表最大处理能力，仅供参考。

Demo3:

还有其他的技巧吗?

```
(myvirtualenv) [opc@domias OraIngestTests]$ python runtests.py -u anpr -p anpr -cs                                    -st light -com 1 -bat
-tc 8 -proc 11 -scale 1 -ss -rs clean_up.sql
```

Monitored Databases

Wait Event Classes over Time

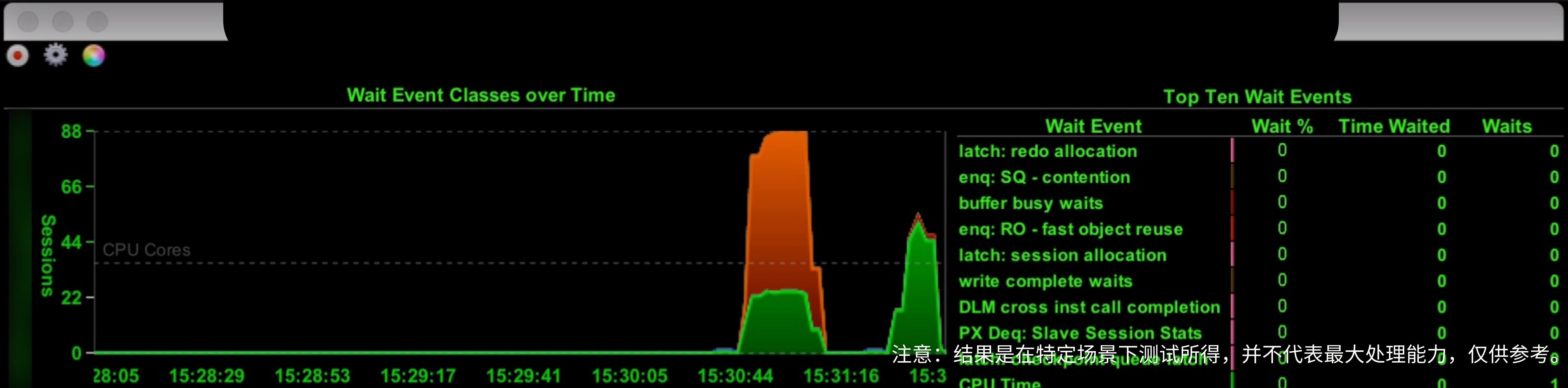
Top Ten Wait Events

Wait Event	Wait %	Time Waited	Waits
latch: redo allocation	0	0	0
enq: SQ - contention	0	0	0
buffer busy waits	0	0	0
enq: RO - fast object reuse	0	0	0
latch: session allocation	0	0	0
write complete waits	0	0	0
DLM cross inst call completion	0	0	0
PX Deq: Slave Session Stats	0	0	0
latch: checkpoint queue latch	0	0	0
CPU Time	0	0	1

```
[(myvirtualenv) [opc@domias OraIngestTests]$ python runtests.py -u anpr -p anpr -cs [redacted] -st light -com 1 -ba
-tc 8 -proc 11 -scale 1 -ss -rs clean_up.sql
Running script clean_up.sql
Tests Run: 100%|██████████████████████████████████████████| 1/1 [00:21<00:00, 21.33s
+--------------+------------+-----------+---------+------------+-------+---------------------+-----------------+-------------------+-----------------
| JVMs Started | Thread Count | Commit Size | Batch Size | Image Size | Async | Total Rows Inserted | Real Time Taken | Total Insert Time | Rows/sec Inserted
+--------------+------------+-----------+---------+------------+-------+---------------------+-----------------+-------------------+-----------------
|           11 |          8 |         1 |       1 |          0 | False |             1100000 |           21.33 |            201.85 |            59,054
+--------------+------------+-----------+---------+------------+-------+---------------------+-----------------+-------------------+-----------------
[(myvirtualenv) [opc@domias OraIngestTests]$ python runtests.py -u anpr -p anpr -cs [redacted] -st light -com 1 -ba
-tc 8 -proc 11 -scale 1 -ss -rs clean_up.sql -async
Running script clean_up.sql
Tests Run: 100%|██████████████████████████████████████████| 1/1 [00:  <00:00, 13.03s
+--------------+------------+-----------+---------+------------+-------+---------------------+-----------------+-------------------+-----------------
| JVMs Started | Thread Count | Commit Size | Batch Size | Image Size | Async | Total Rows Inserted | Real Time Taken | Total Insert Time | Rows/sec Inserted
+--------------+------------+-----------+---------+------------+-------+---------------------+-----------------+-------------------+-----------------
|           11 |          8 |         1 |       1 |          0 |  True |             1100000 |           13.03 |              110. |           100,917
+--------------+------------+-----------+---------+------------+-------+---------------------+-----------------+-------------------+-----------------
(myvirtualenv) [opc@domias OraIngestTests]$
```

如果再做一些配置改变呢？吞吐量又能翻倍！

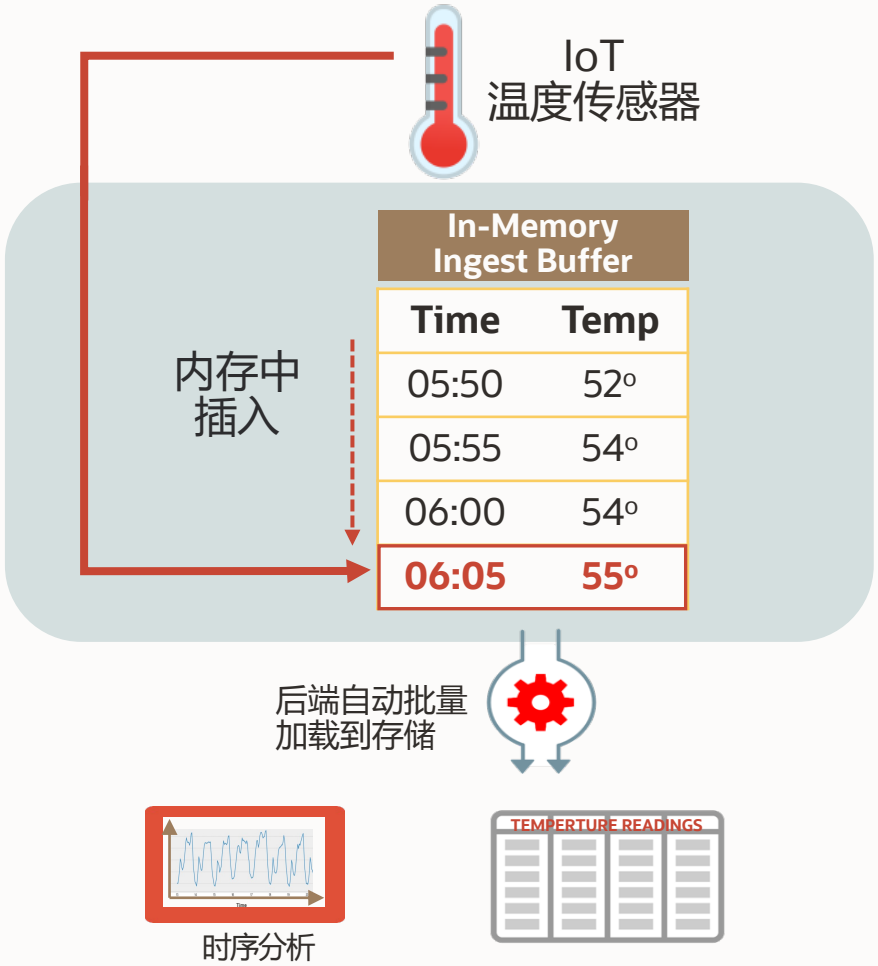
（例如，增加initrans，disable table lock, commit nowait…）



注意：结果是在特定场景下测试所得，并不代表最大处理能力，仅供参考。

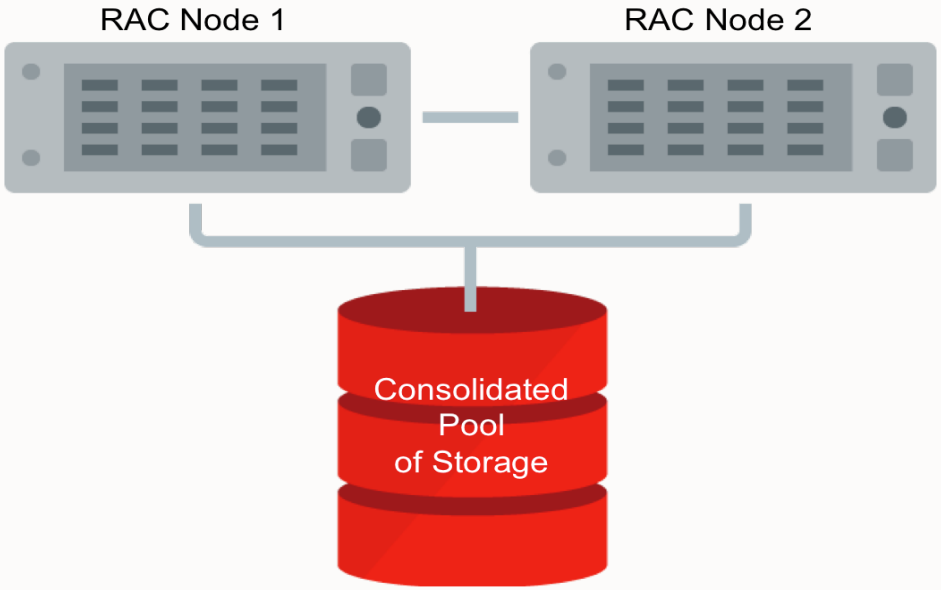
或者，利用新版本中的内存优化行存储，支撑超高频插入

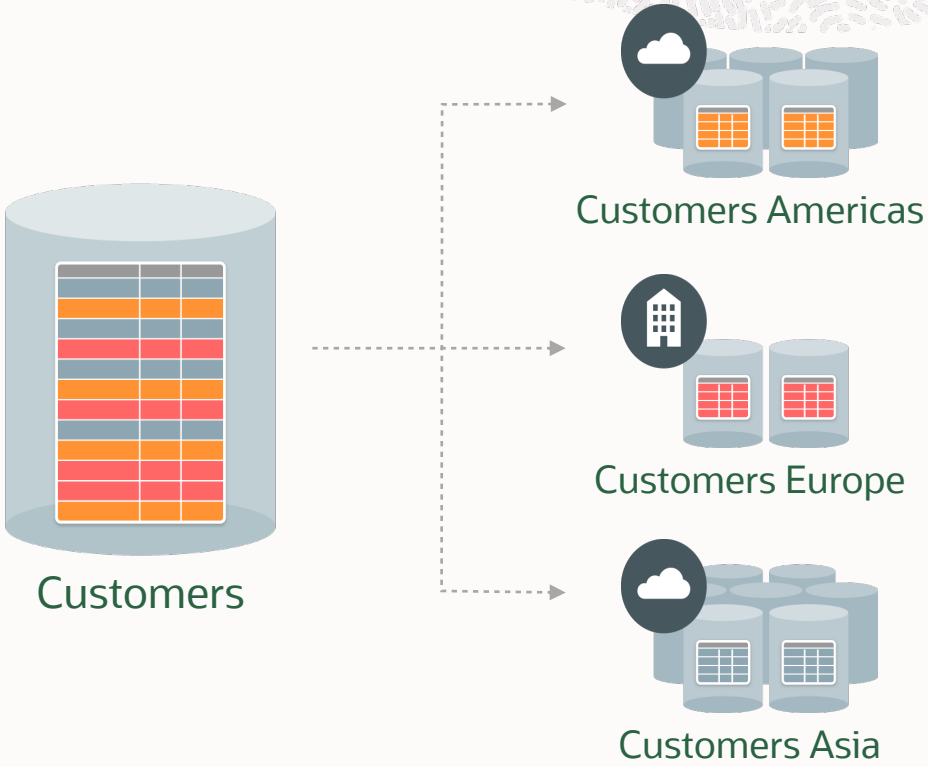
示例：插入温度传感器读数



- 用于将数据插入数据库的内存优化机制
- 理想的物联网（IoT）交易的选择
- 行缓存在内存中，并异步落地到磁盘
- 性能
  - ➢ 在x86双插槽服务器上的测试数据：
    - 每秒执行 2500 万次插入操作
- API允许开发人员检查其插入的持久性

更进一步，从数据库架构讲，集群与分片进一步提升可扩展性





灵活性

为什么IoT需要灵活性?

IoT还正处于高速发展期，随着新的设备不断产生新的用例
适应新的数据格式是必要的
同时也要能够分析并管理超大量数据

一般地，开发者都愿意使用**JSON**灵活处理各种数据需求

```
{
    "Fflat": 0, //正向平电量
    "Fpeak": 28.91, //正向峰电量
    "Fsharp": 105.6, //正向尖电量
    "Ftotal": 105.15, //正向总电量
    "Fvalley": 23.29, //正向谷电量
    "ID": "865650045071202", //无线红外抄表器ID
    "Rflat": 0, //反向平电量
    "Rpeak": 0, //反向峰电量
    "Rsharp": 0, //反向尖电量
    "Rtotal": 0, //反向总电量
    "Rvalley": 0, //反向谷平电量
    "actPower": 13.715, //总有功功率
    "actPowerA": 4.524, //A相有功功率
    "actPowerB": 0, //B相有功功率
    "actPowerC": 9.191, //C相有功功率
    "currentA": 40, //A相电流
    "currentB": 70, //B相电流
    "currentC": 50, //C相电流
    "fac": 0.414, // 总功率因数
    "facA": 0.707, // A相功率因数
    "facB": 0, //B相功率因数
    "facC": 0.707, //C相功率因数
    "lastMRtotal": 0, //上月反向总电量
    "lastMtotal": 0, //上月正向总电量
    "reactPower": 13.715, //总无功功率
    "reactPowerA": 4.524, //A相无功功率
    "reactPowerB": 0, //B相无功功率
```

IoT JSON示例
光伏发电信息（来自网络）

JSON的SQL原生支持

对新的IoT应用而言，JSON是最流行的数据格式

数据库中的JSON能极大地简化应用开发

- 在应用和数据库中使用同样的schema-less数据表达

Oracle将JSON存储为表的列，**支持标准SQL的访问**

- SELECT m.json_column.address.city
  FROM Meter_Readings m;

在数据库中使用JSON，**适用于Oracle数据库的所有特性**

- 分析，加密，列式内存，RAC集群，复制，并行SQL等等
- 并且能够使用SQL管理JSON文档、索引以及元素

Better JSON than JSON Only Databases

```
(myvirtualenv) [opc@domias OraIngestTests]$ python runtests.py -u anpr -p anpr -cs [redacted] -st document_light -c
30000 -bat 50 -tc 5 -proc 11 -scale 20 -ss -rs clean_up.sql
```

Monitored Databases

Wait Event Classes over Time

Top Ten Wait Events

Wait Event	Wait %	Time Waited	Waits
latch: redo allocation	0	0	0
enq: SQ - contention	0	0	0
buffer busy waits	0	0	0
enq: RO - fast object reuse	0	0	0
latch: session allocation	0	0	0
write complete waits	0	0	0
DLM cross inst call completion	0	0	0
PX Deq: Slave Session Stats	0	0	0
latch: checkpoint queue latch	0	0	0
CPU Time			1

```
(myvirtualenv) [opc@domias OraIngestTests]$ python runtests.py -u anpr -p anpr -cs [redacted] -st document_light -
30000 -bat 50 -tc 5 -proc 11 -scale 20 -ss -rs clean_up.sql
Running script clean_up.sql
Tests Run: 100%|████████████████████████████████████████| 1/1 [00:20<00:00, 20.?7s/
```

JVMs Started	Thread Count	Commit Size	Batch Size	Image Size	Async	Total Rows Inserted	Real Time Taken	Total Insert Time	Rows/sec Inserted
11	5	30000	50	100	False	22000000	20.17	193.52	1,173,959

```
(myvirtualenv) [opc@domias OraIngestTests]$
```

一个简单的例子，每秒处理110万JSON文档

Monitored Databases

Wait Event Classes over Time

Top Ten Wait Events

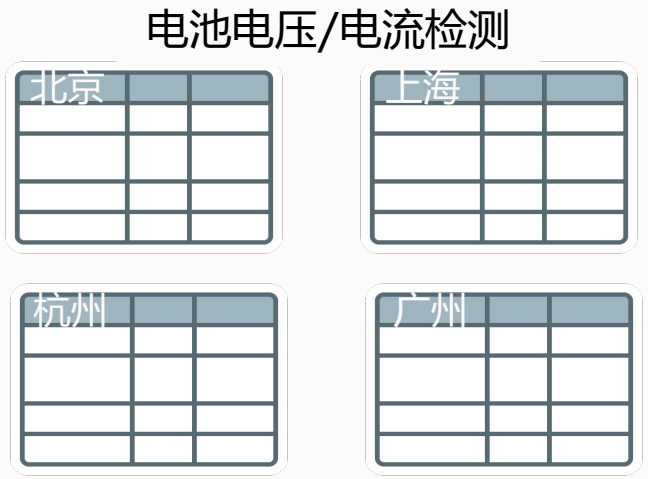
Wait Event	Wait %	Time Waited	Waits
latch: redo allocation	0	0	0
enq: SQ - contention	0	0	0
buffer busy waits	0	0	0
enq: RO - fast object reuse	0	0	0
latch: session allocation	0	0	0
write complete waits	0	0	0
DLM cross inst call completion	0	0	0
PX Deq: Slave Session Stats	0	0	0
latch: checkpoint queue latch	0	0	0

注意：结果是在特定场景下测试所得，并不代表最大处理能力，仅供参考。

数据分区能力也是适应灵活性、提升管理效率的重要手段

单张超大表

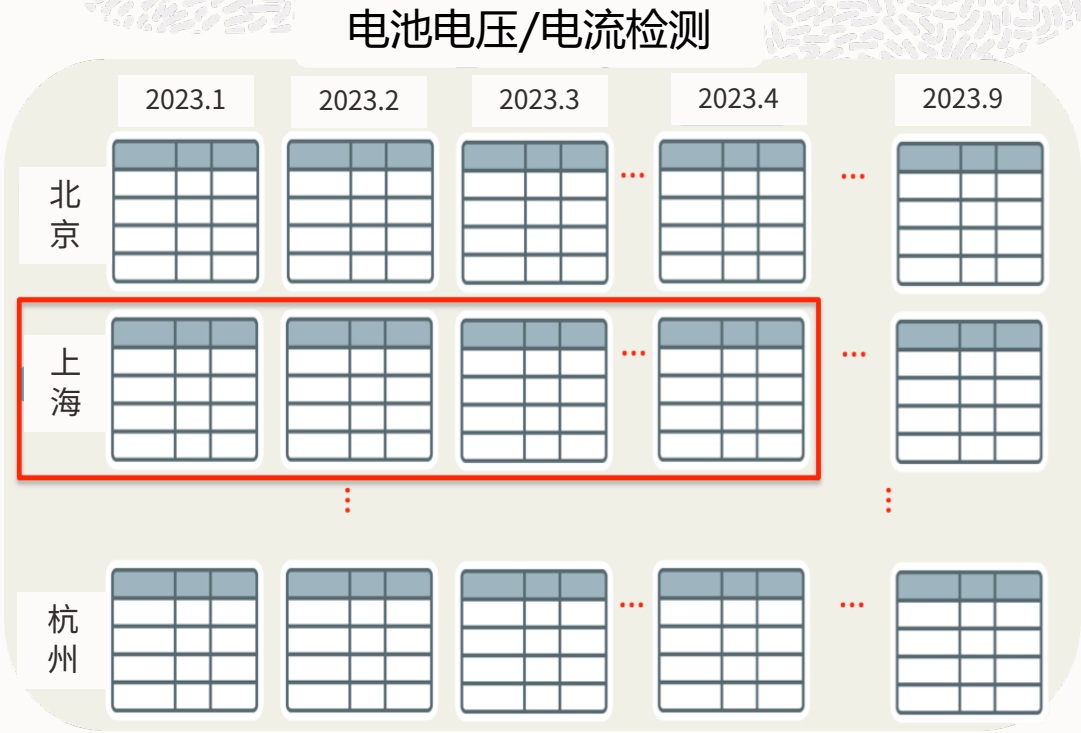
很难管理



分区

分而治之

容易管理，且提升性能

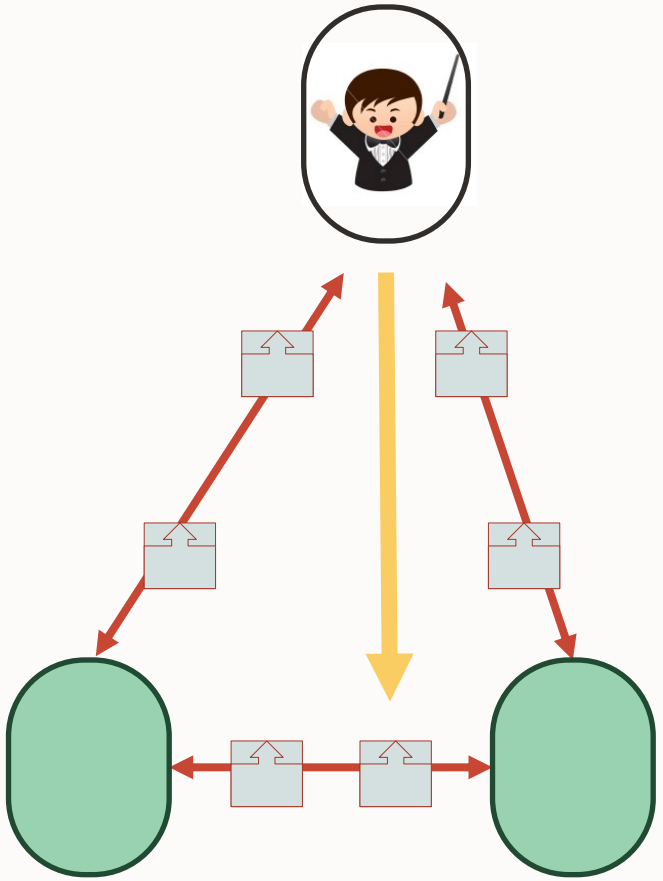


二级分区

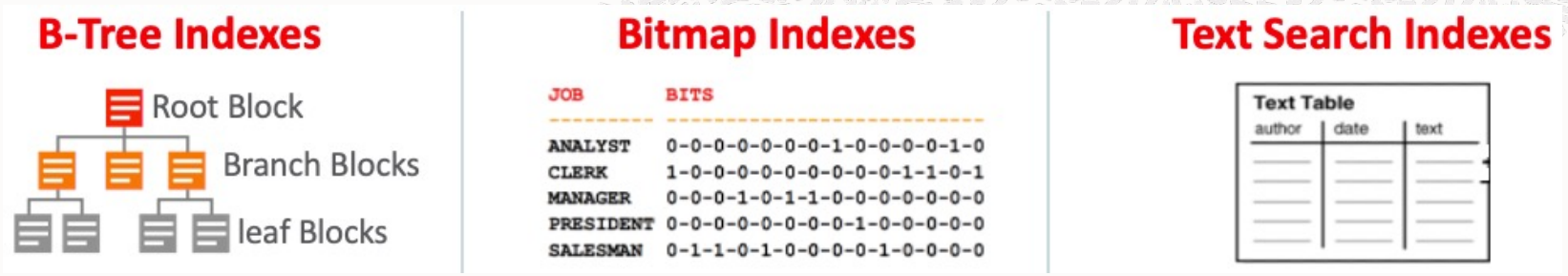
对应用而言，分区是透明的（即，**无需修改代码**）

实时分析

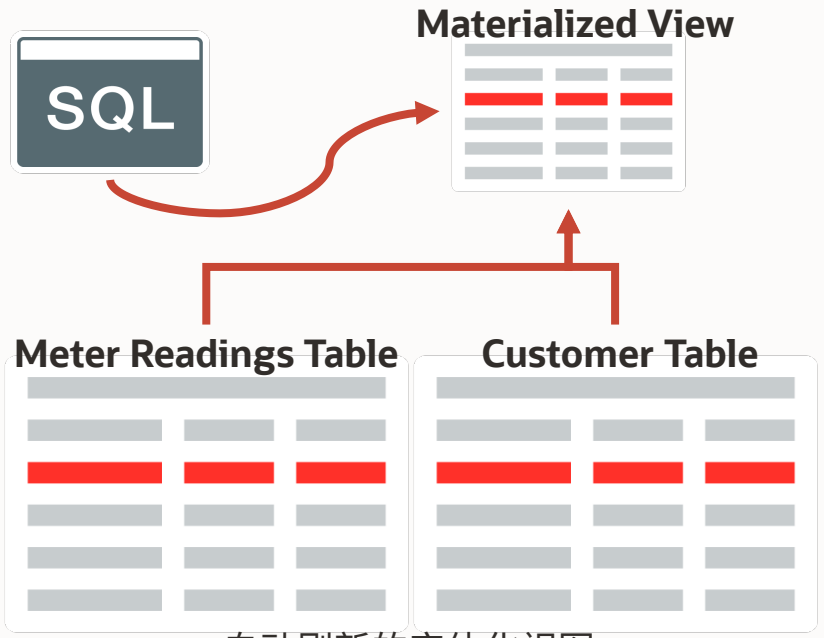
技术一：基本技术



节点内并行
+
节点间并行

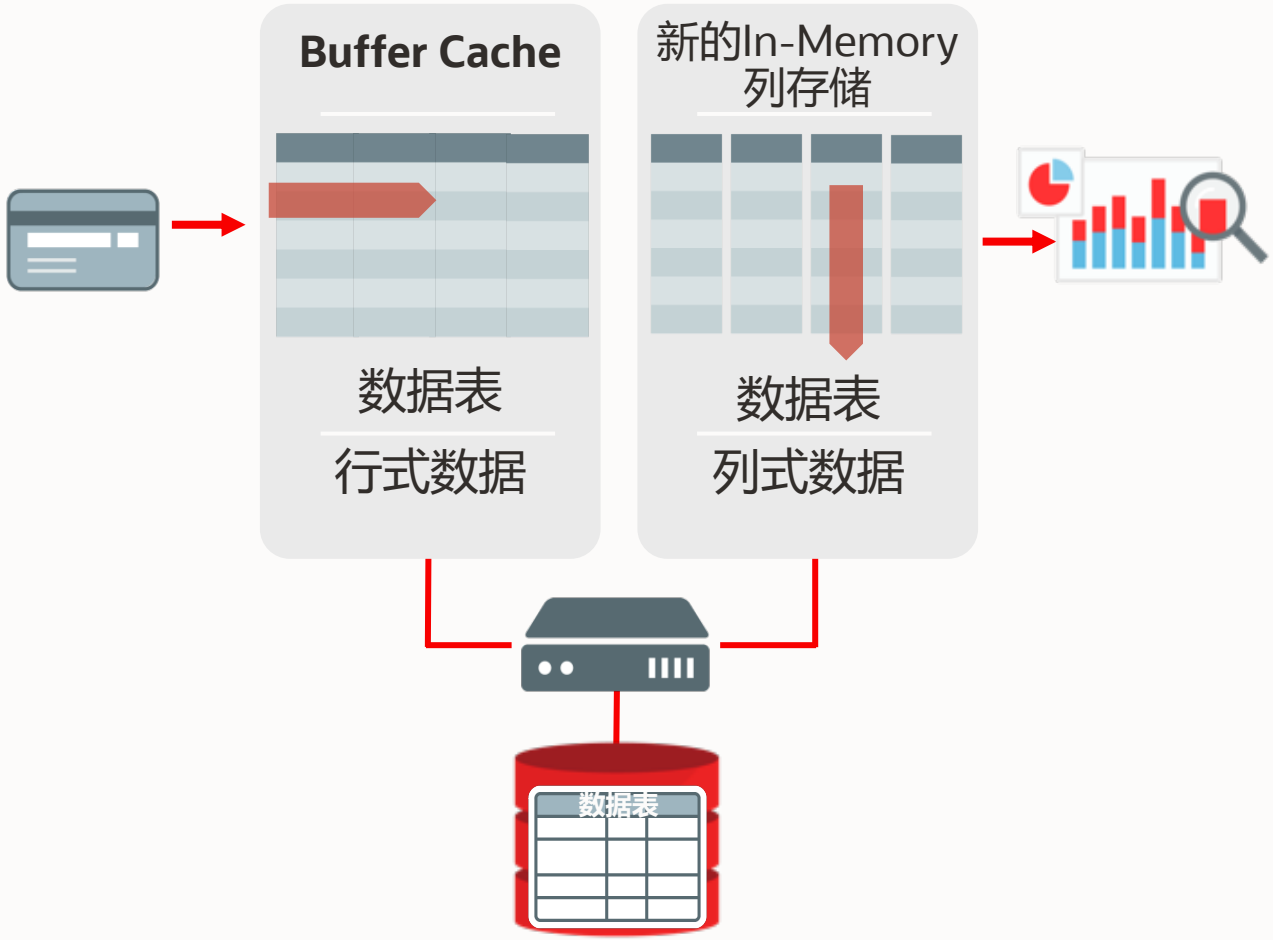


传统B树之外的多种索引技术



自动刷新的实体化视图

技术二：（双模）内存技术

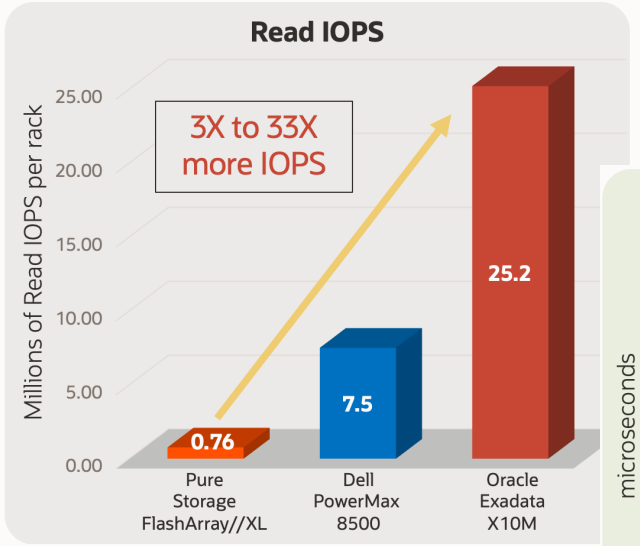


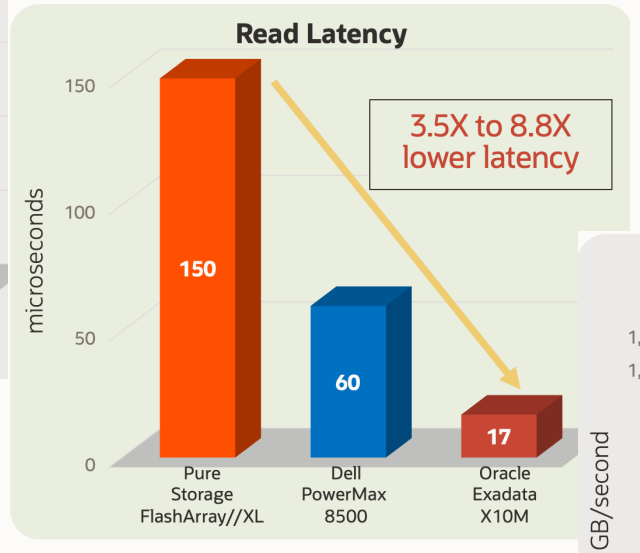
- 同一张表的数据，行式和列式共存
- 同时有效，且数据一致
- 交易类业务使用行式数据提高效率
- 分析和报表类业务使用列式数据提高效率两个数量级

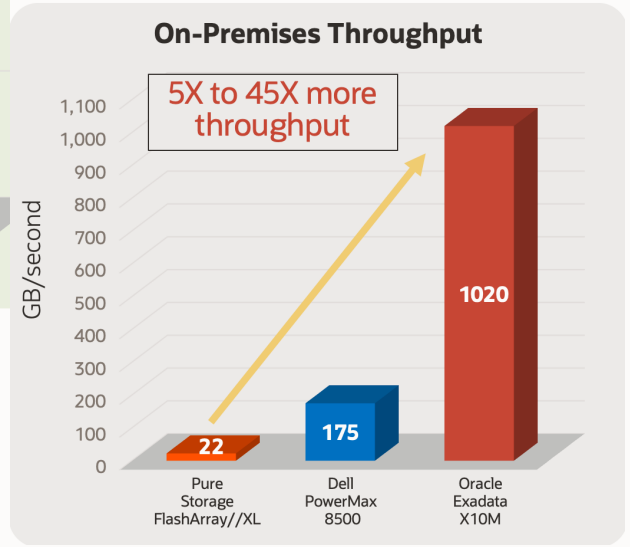
➢优化引擎自动判断，**无需修改应用代码**

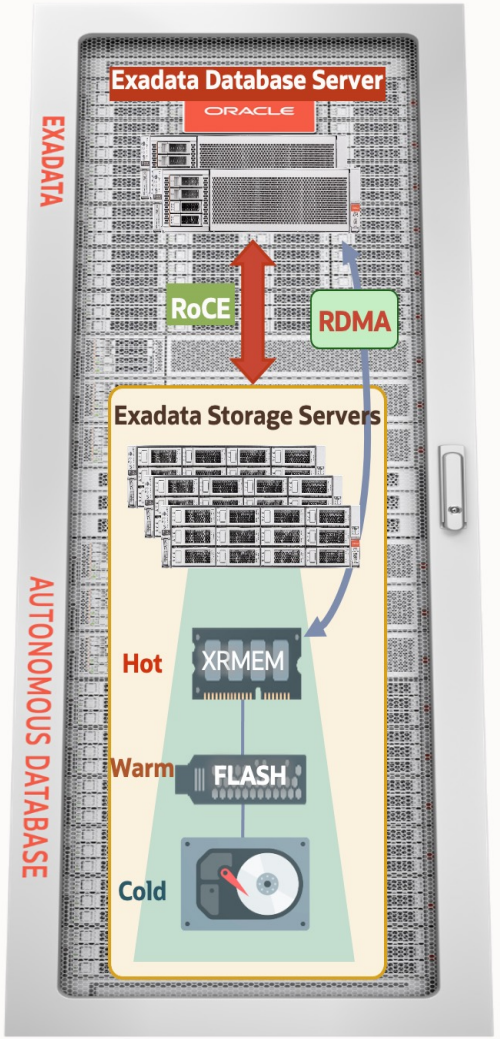
技术三：工程化数据库一体机Exadata

数据库+系统软件+硬件，三方融合；智能存储节点参与运算



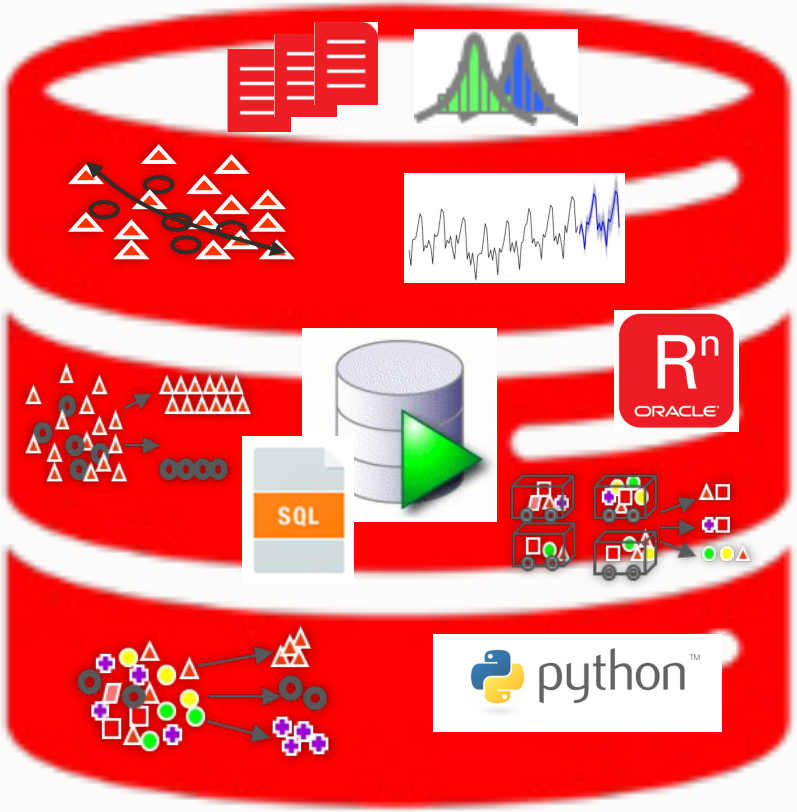


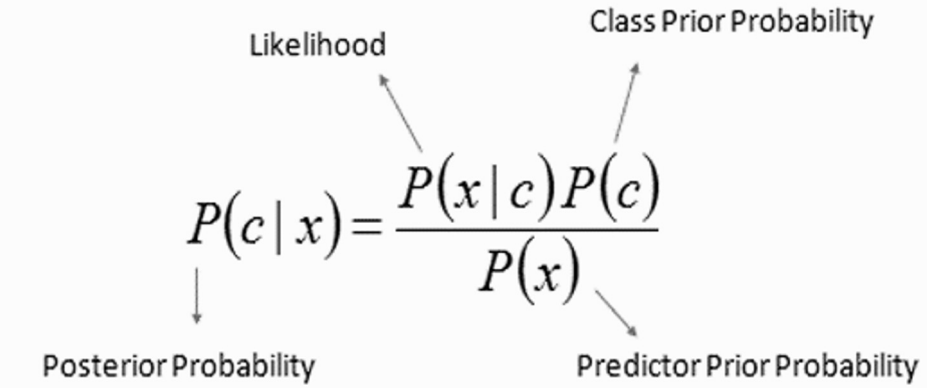




技术四：移动算法，而不是移动数据！

直接在数据库内应用算法，而不是将数据导出到其他平台再去计算——实时性不够



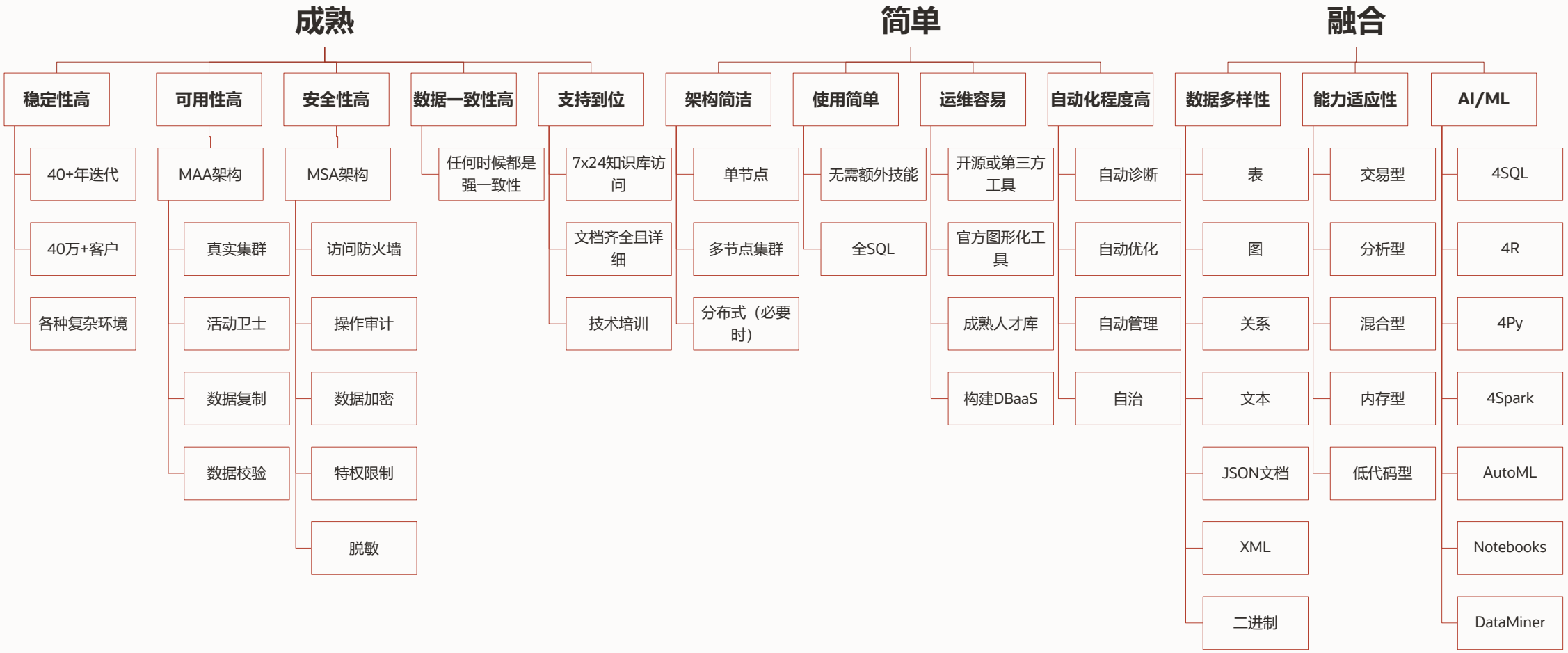


$$P(c|X) = P(x_1|c) \times P(x_2|c) \times \cdots \times P(x_n|c) \times P(c)$$

总结

选择合适的技术，
才能事半功倍

传统数据库的二次革命为适应IoT时代的数据爆炸提供了坚实可靠的基础



数据库和云讲座群

20-22

甲骨文云技术公众号

技术专家1V1深入交流


ORACLE