

# TPCH-Q1性能优化实践

付新



**达梦数据库概述**

**TPCH-Q1优化实践**

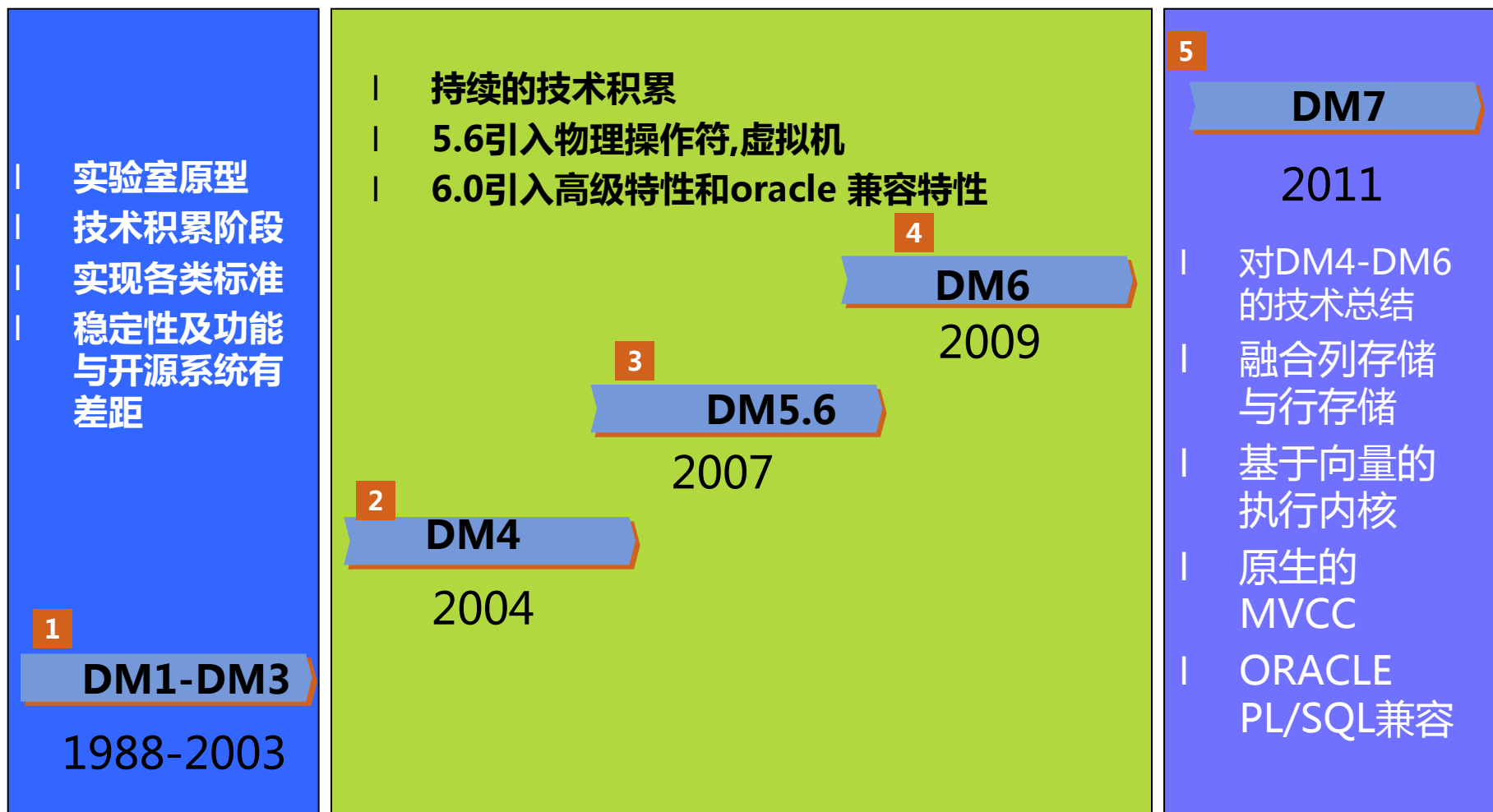
**将来的工作**

# 达梦数据库概述

---

- 完全的自主知识产权
- 大型、通用、安全
- 传统架构
  - 基于磁盘
  - 行列混合式存储
  - 擅长OLTP处理
- 关系型数据库管理系统

# 达梦数据库概述-系统研制历程





**达梦数据库概述**

**TPCH-Q1优化实践**

**将来的工作**

# 语句和执行计划

- **TPCH的Q1语句：**




- select l\_returnflag,
- l\_linestatus,
- sum(l\_quantity) as sum\_qty,
- sum(l\_extendedprice) as sum\_base\_price,
- sum(l\_extendedprice \* (1 - l\_discount)) as sum\_disc\_price,
- sum(l\_extendedprice \* (1 - l\_discount) \* (1 + l\_tax)) as sum\_charge,
- avg(l\_quantity) as avg\_qty,
- avg(l\_extendedprice) as avg\_price,
- avg(l\_discount) as avg\_disc,
- count(\*) as count\_order
- from lineitem
- where l\_shipdate <= date
- group by
- l\_returnflag, l\_line
- order by
- l\_returnflag, l\_line

- **其查询计划：**




- #NSET2;
- #PRJT2; exp\_num(10), is\_atom(FALSE)
- #SORT2; key\_num(2), is\_distinct(FALSE)
- #HAGR2; grp\_num(2), sfun\_num(8)
- #PRJT2; exp\_num(7), is\_atom(FALSE)
- #SLCT2; LINEITEM.L\_SHIPDATE <= exp44
- #CSCN2; INDEX33555444(LINEITEM)

# TPC官网TPC-H性能榜(单机)



## 100 GB Results

Rank	Company	System	QphH	Price/QphH	Watts/KQphH	System Availability	Database	Operating System	Date Submitted
1		Lenovo ThinkServer RD630	420,092	.11 USD	NR	05/13/13	VectorWise 3.0.0	Red Hat Enterprise Linux 6.4	05/13/13
2		Dell PowerEdge R720	403,230	.12 USD	NR	05/08/12	Action VectorWise 2.0.1	Red Hat Enterprise Linux 6.1	05/13/12
3		Cisco UCS C250 M2 Extended-Memory Server	332,481	.15 USD	NR	02/14/12	Action VectorWise 2.0.1	Red Hat Enterprise Linux 6.0	02/14/12

## 300 GB Results

Rank	Company	System	QphH	Price/QphH	Watts/KQphH	System Availability	Database	Operating System	Date Submitted
1		Lenovo ThinkServer RD630	434,353	.24 USD	NR	05/10/13	VectorWise 3.0.0	Red Hat Enterprise Linux 6.4	05/10/13
2		Dell PowerEdge R720	410,594	.28 USD	NR	05/08/12	Action VectorWise 2.0.1	Red Hat Enterprise Linux 6.1	05/13/12
3		Cisco UCS C250 M2 Extended-Memory Server	331,658	.34 USD	NR	02/13/12	Action VectorWise 2.0.1	Red Hat Enterprise Linux 6.0	02/13/12

## 1,000 GB Results

Rank	Company	System	QphH	Price/QphH	Watts/KQphH	System Availability	Database	Operating System	Date Submitted
1		IBM System x3850 X6	519,976	1.36 USD	NR	04/16/14	Microsoft SQL Server 2014 Enterprise Edition	Microsoft Windows Server 2012 R2 Standard	04/15/14
2		INSPUR K1	485,242	4.03 CNY	NR	06/04/14	Action Vector 3.0.0	K-UX2.2	06/03/14
3		Dell PowerEdge R820	445,529	.75 USD	NR	06/01/12	Action VectorWise 2.0.1	Red Hat Enterprise Linux 6.1	06/01/12

# 性能结果对比

SCALE = 10 (LINEITEM=6000W) , CPU:3.0GHz 8 Cores

VECTOR WISE: **360ms** (并行度8)

DM6: >50s(无并行)

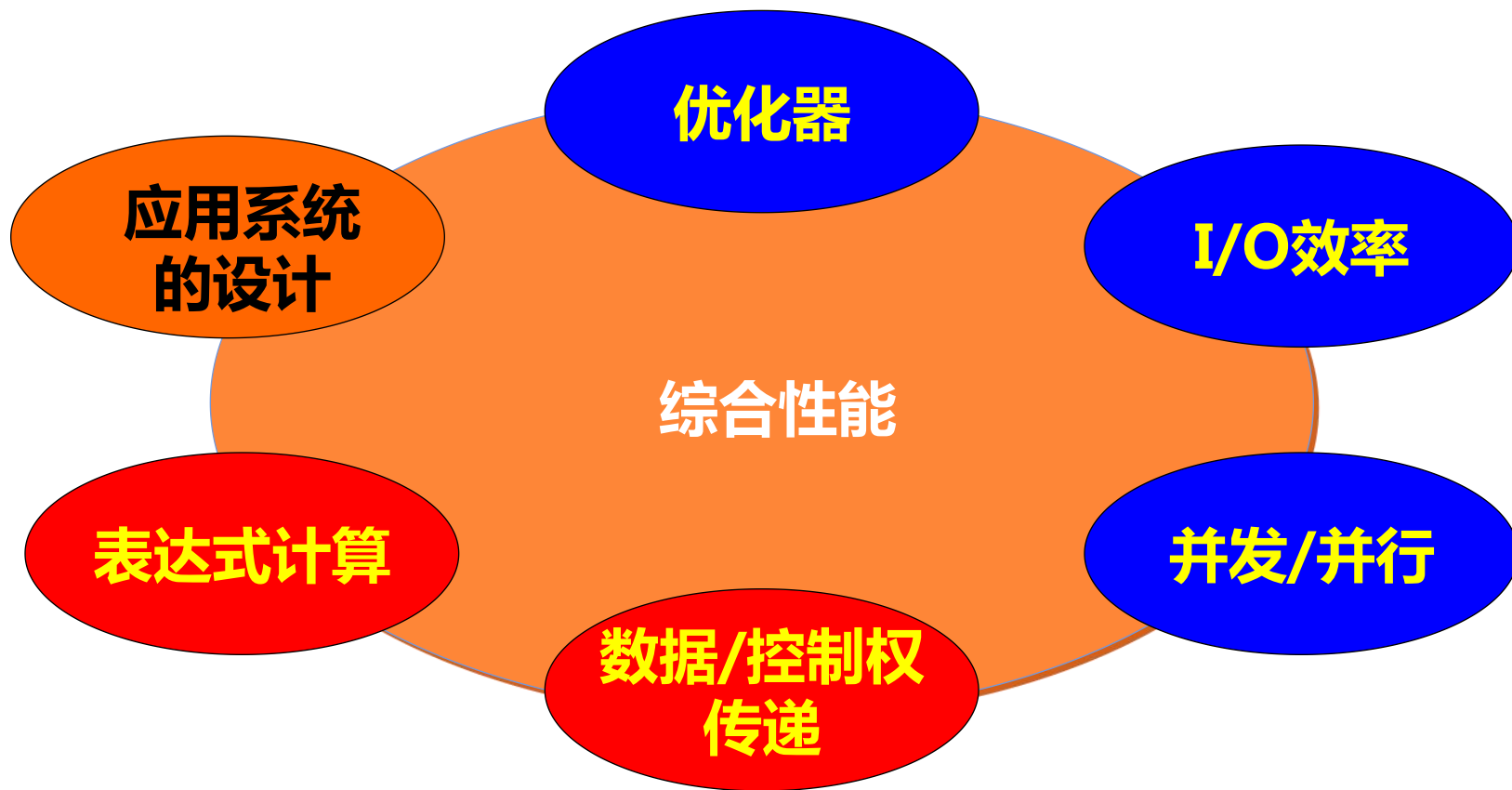
DM7(2011年): 5000ms (并行度8)

DM7(2014年): 750ms (并行度8)

DM7实验版(2015年): **<500ms** (并行度8)



# 关于性能的认识



## 智能代价优化器

- 基于多趟分析的代价优化策略
- 语义分析、代价优化过程分离
- 灵活的计划变换控制
- 以时间为单位的代价模型
- 智能自适应的查询优化器

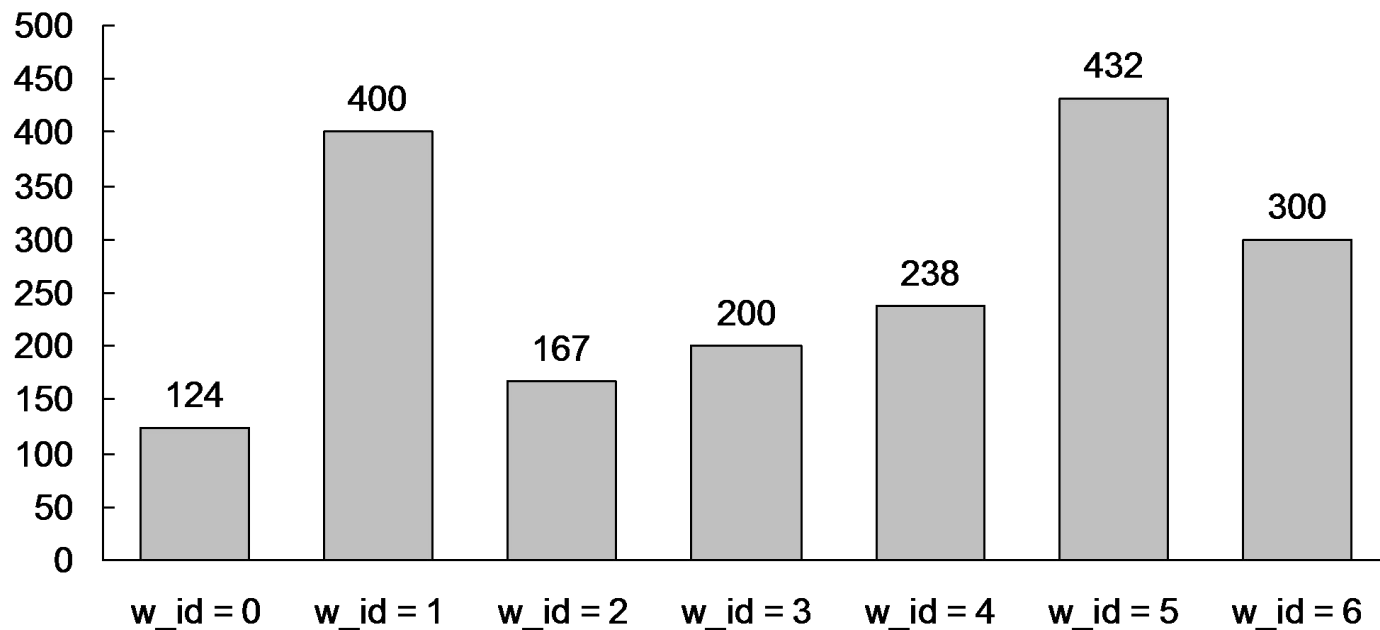
# 优化器-统计信息

---

- 记录数据分布情况
- 用于精确行数估计，特别是数据分布不规则的情况
- 对基数及代价计算有重大影响
- 直方图分类
  - 频率直方图
  - 等高直方图

# 优化器-频率直方图

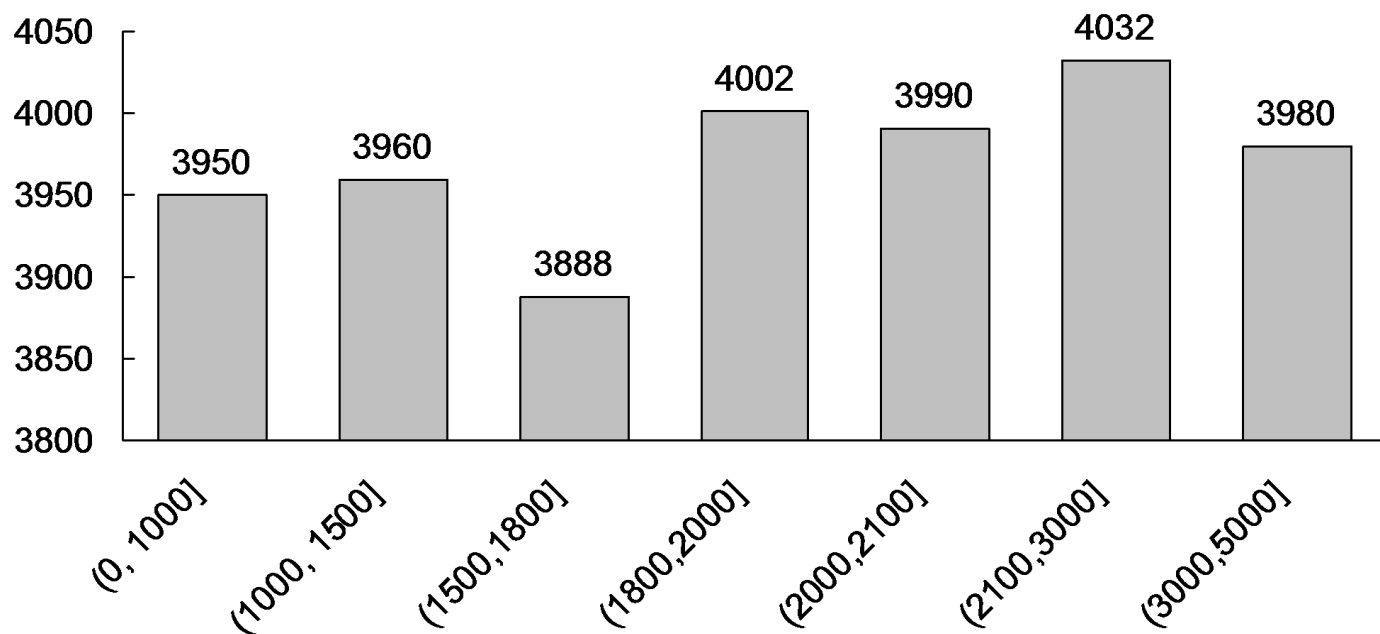
- 不同值较少的情况
- 记录每个值出现的次数
- 能精确计算每个值的行数



各仓库对应的有效记录数

# 优化器-等高直方图

- 不同值个数很多情况
- 横轴的值是非均匀分布
- 纵轴基本等值



不同区间对应的有效记录数基本一致

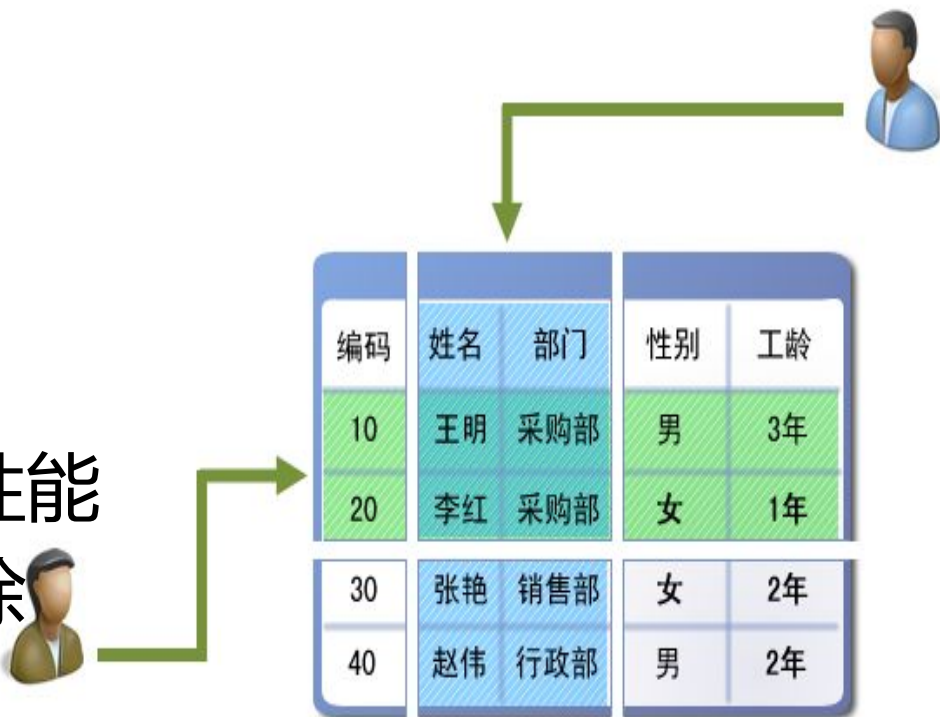
# IO效率-分区、压缩、列存储

## I/O性能优化

- Ø 数据分区
- Ø 数据动态透明压缩
- Ø 快速批量数据装载

## 提供按列存储选项

- Ø 特别适合OLAP应用
- Ø 大幅提升扫描查询性能
- Ø 适合批量装载与删除



编码	姓名	部门	性别	工龄
10	王明	采购部	男	3年
20	李红	采购部	女	1年
30	张艳	销售部	女	2年
40	赵伟	行政部	男	2年

## 列存储和行存储技术的融合

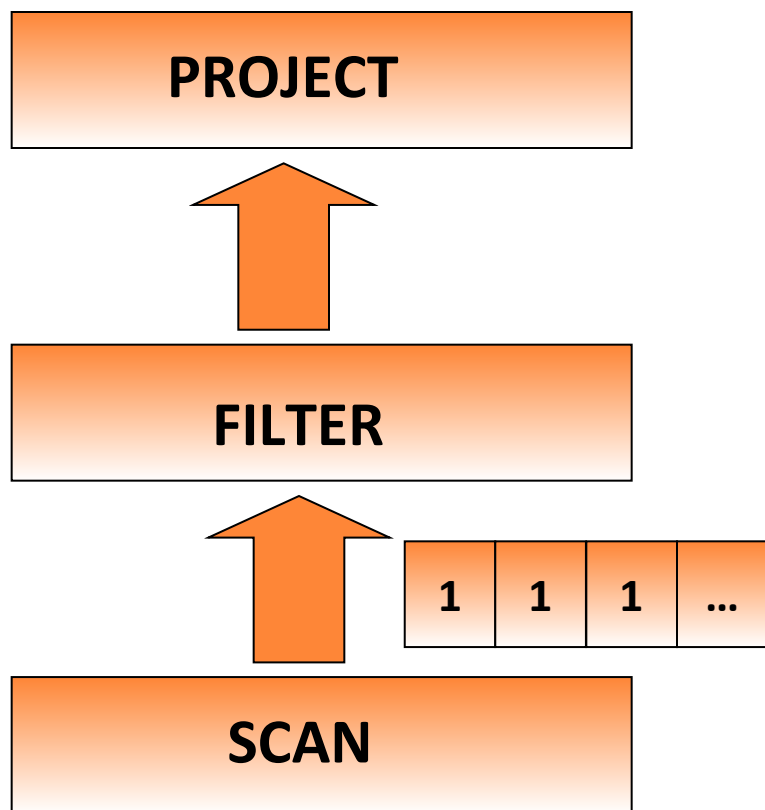
- 列存储:
  - 数据按列存储，结合自适应压缩技术。
- 行存储:
  - 简化物理记录格式，字段物理次序与逻辑次序分离；提升记录的解析速度。
- 结合向量处理技术，同时适应OLAP和OLTP应用需求。

## 向量数据处理

- 在数据泵一次传送一批数据
- 减少控制转移的CPU损耗；
- 结合多版本并发控制
- 批量的表达式计算
- 适应OLAP应用

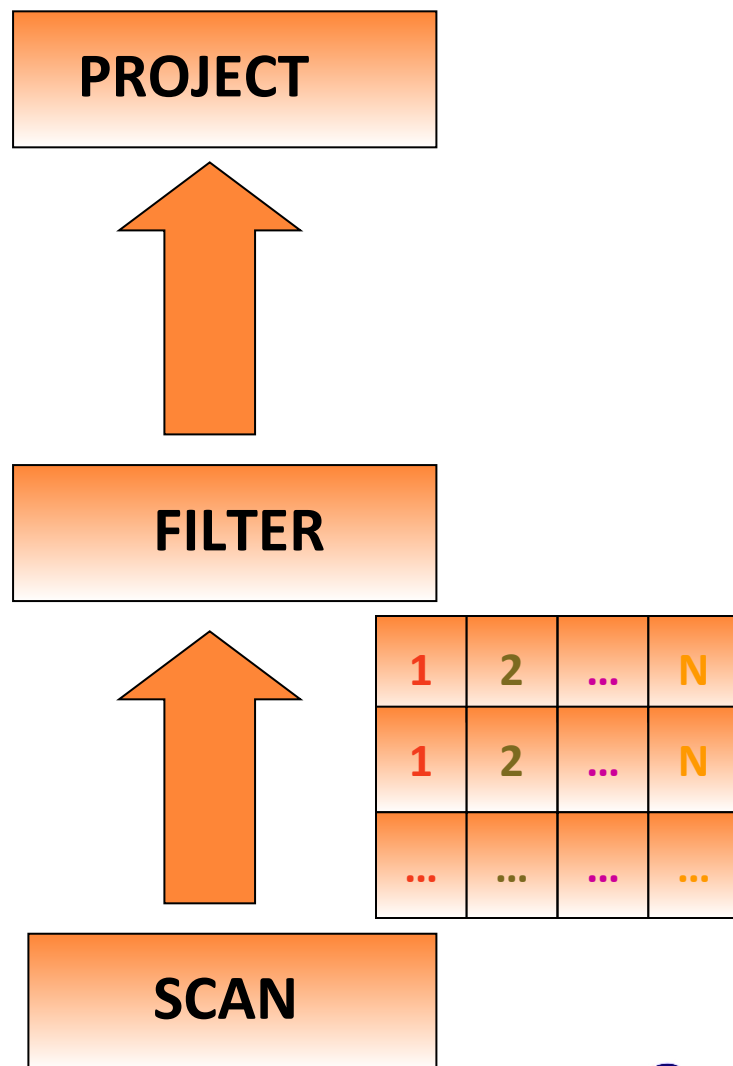


# 数据/控制权传递-传统的数据传递



- | 一次只传递一条记录；
- | 每个操作符一次只处理一行记录。
- | 控制权需要反复传递

# 数据/控制权传递-向量式的数据传递



- | 减少控制权限的反复传递
- | 提升CPU的有效利用率
- | 便于表达式批量计算

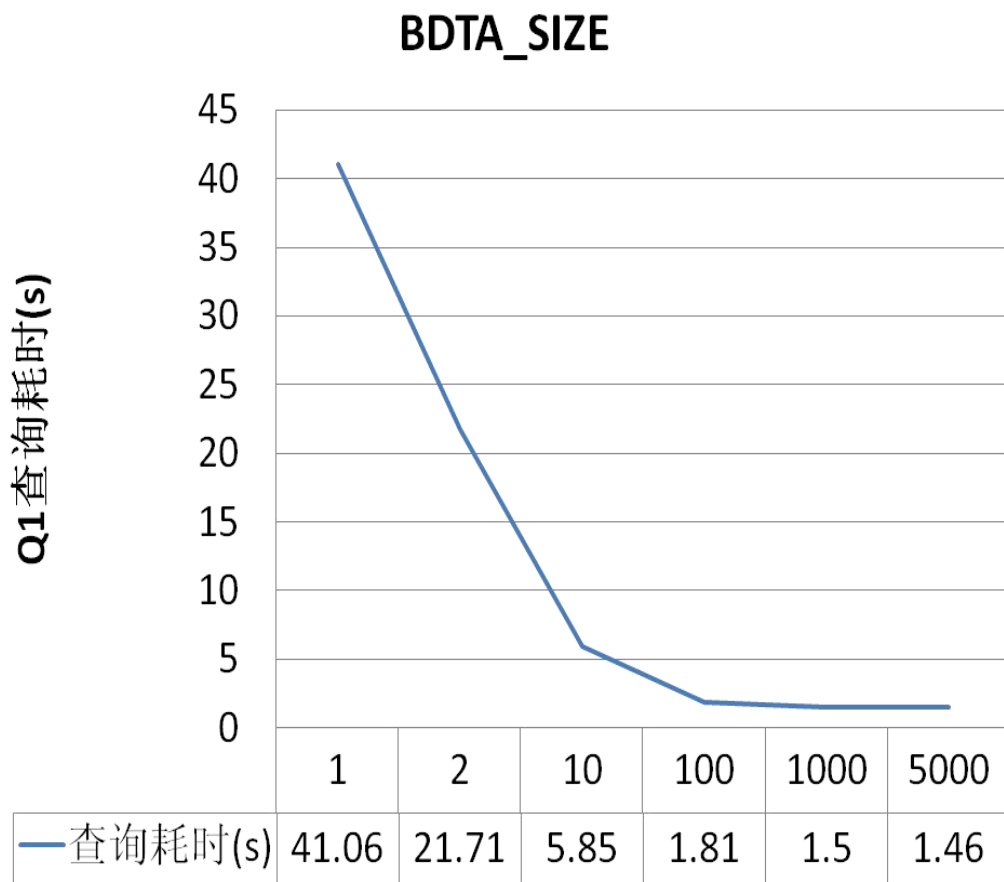
# 数据/控制权传递-批量表达式计算

```
for (i = 0; i < n; i++)  
{  
    r = (int64)opr1[i] + opr2;  
  
    if (r != (int)r)  
        return EC_DATA_OVERFLOW;  
  
    res[i] = (int)r;  
}
```

- 虚拟机支持批量计算指令
- 一次计算一批数据
- 利用CPU的CACHE
- 利用CPU的SIMD特性
- 避免传统DBMS的函数反复调用代价
- 接近于C的效率
- 比一次一行模式快10-100倍以上

# 数据/控制权传递-批量尺寸对性能的影响

- SF=1, TPCH Q1
- BDTA\_SIZE: 可配置的批量大小参数
- 增大BDTA\_SIZE可以有效地提高执行效率



# 表达式计算-中间结果重用

- 一个复杂查询在一条sql语句中使用多次的情况。
- 将这个复杂查询提取出来，结果缓存，构建一次，使用多次。
  - 例:
    - `Select *from v1 where c2 = (select max(c2) from v1)`

# 表达式计算-提取重用表达式

- 一个表达式出现多次，只计算一次
- `Select sum(1 * c1), sum(2 * (1 * c1))  
from t`
- $v1 = 1 * c1$
- `Select sum(v1), sum(2 * v1) from t`

# 表达式计算-LIKE谓词性能

- select count(\*) from orders where  
o\_comment not like  
'%special%requests% '
- DBMS 'O' 11g: 3.3
- DBMS 'S' 2005: 10
- DM7: 0.4

orders : 1,500,000记录

cpu 2.2G,多次执行

# 并行执行-单机与多机

```
•1 #NSET2: [859, 59154, 140]
•2 #PRJT2: [859, 59154, 140]; exp_num(10), is_atom(FALSE)
•3 #SORT3: [859, 59154, 140]; key_num(2), is_distinct(FALSE), top_flag(0)
•4 #HAGR2: [859, 59154, 140]; grp_num(2), sfun_num(8)
•5 #PRJT2: [859, 5915419, 140]; exp_num(7), is_atom(FALSE)
•6 #SLCT2: [859, 5915419, 140]; LINEITEM.L_SHIPDATE <= var3
•7 #HFSCN: [859, 6001215, 140]; (LINEITEM)
```

```
•1 #NSET2: [859, 59154, 140]
•2 #PRJT2: [859, 59154, 140]; exp_num(10), is_atom(FALSE)
•3 #SORT3: [859, 59154, 140]; key_num(2), is_distinct(FALSE), top_flag(0)
•4 #HAGR2: [859, 59154, 140]; grp_num(2), sfun_num(8)
•5 #LOCAL GATHER: [859, 59154, 140]; op_id(1) n_grp_by (0) n_cols(0) n_keys(0)
•6 #HAGR2: [859, 59154, 140]; grp_num(2), sfun_num(9)
•7 #PRJT2: [859, 5915419, 140]; exp_num(7), is_atom(FALSE)
•8 #SLCT2: [859, 5915419, 140]; LINEITEM.L_SHIPDATE <= var3
•9 #HFSCN: [859, 6001215, 140]; (LINEITEM)
```

```
•1 #NSET2: [429, 29578, 140]
•2 #PRJT2: [429, 29578, 140]; exp_num(10), is_atom(FALSE)
•3 #SORT3: [429, 29578, 140]; key_num(2), is_distinct(FALSE), top_flag(0)
•4 #HAGR2: [429, 29578, 140]; grp_num(2), sfun_num(8)
•5 #MPP COLLECT: [429, 29578, 140]; op_id(1) n_grp_by (0) n_cols(0) n_keys(0)
•6 #HAGR2: [429, 29578, 140]; grp_num(2), sfun_num(9)
•7 #PRJT2: [429, 2957854, 140]; exp_num(7), is_atom(FALSE)
•8 #SLCT2: [429, 2957854, 140]; LINEITEM.L_SHIPDATE <= var3
•9 #HFSCN: [429, 3000586, 140]; (LINEITEM)
```





- 执行器利用统计信息
  - Q1: 3600ms->750ms

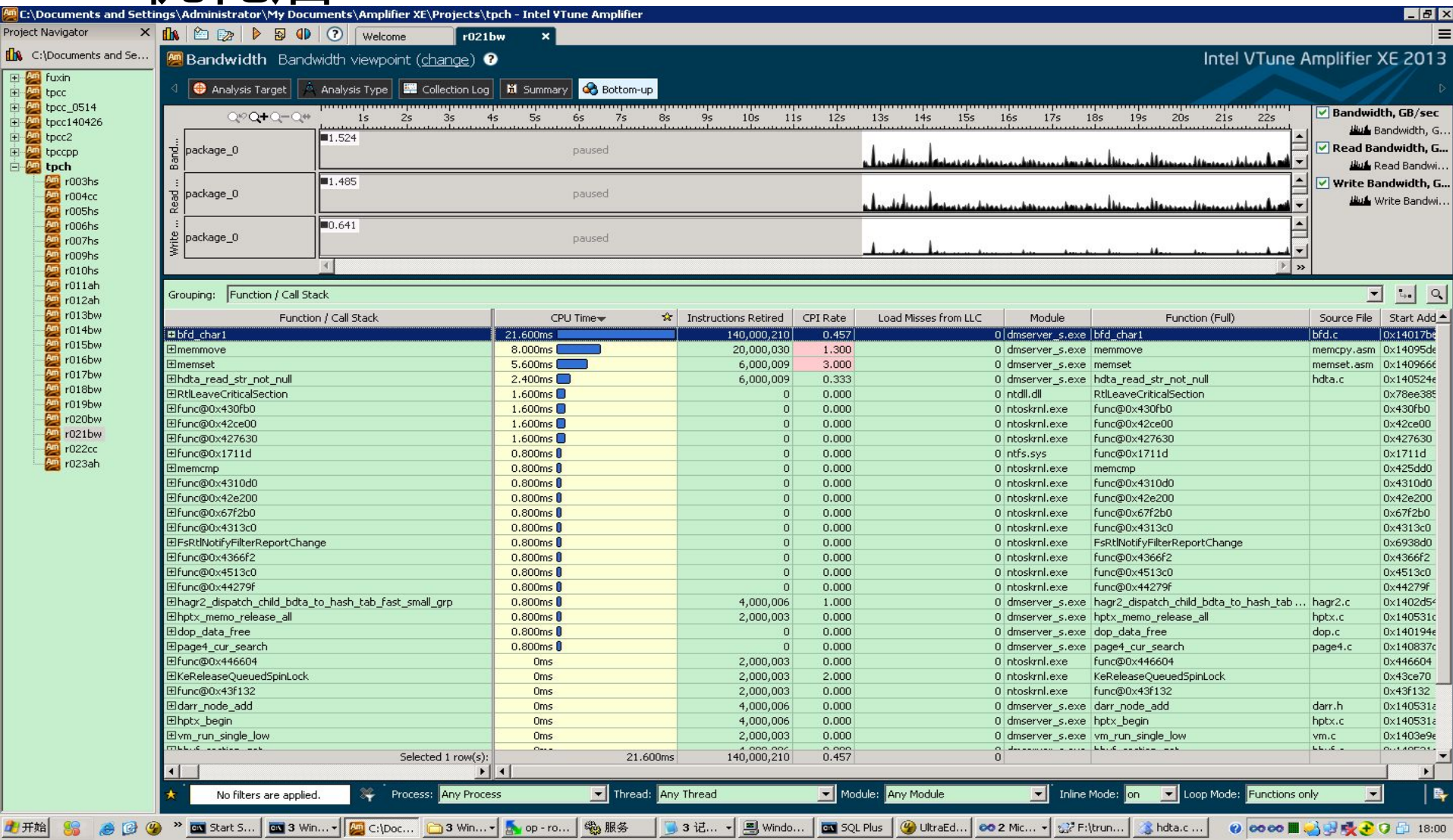
- 循环展开
  - 1100ms->375ms

```
• for(k = 0; k < 100000; k++)
• {
•     for(i = 0; i < 8192; i+=16) //1 2 4 8 16 32 64 128 256
•     {
•         null_arr[i] = null_arr[i] == 0?0:1;           //sum[i] = a[i]+ b[i];
•         null_arr[i+1] = null_arr[i+1] == 0?0:1;
•         null_arr[i+2] = null_arr[i+2] == 0?0:1;
•         null_arr[i+3] = null_arr[i+3] == 0?0:1;
•         null_arr[i+4] = null_arr[i+4] == 0?0:1;
•         .....
•         null_arr[i+14] = null_arr[i+14] == 0?0:1;
•         null_arr[i+15] = null_arr[i+15] == 0?0:1;    //sum[i+15] = a[i+15]+ b[i+15];
•     }
• }
```

- 优化前
- `select sum(l_tax) from h_lineitem;`
  - DM7 **2ms** VW3.5 **43ms**
- `Select 1 from h_lineitem group by l_returnflag;` --待优化的点
  - DM7 **137ms** VW3.5 **36ms**
- `select 1 from h_lineitem group by upper(l_returnflag);`
  - DM7 **344ms** VW3.5 **80ms**
- `select 1 from h_lineitem group by l_returnflag||l_linestatus;`
  - DM7 **374ms** VW3.5 **371ms**

# 其他-2

## • 优化后





**系统的研制历程**

**系统架构**

**将来的工作**

## 分析型场景性能未来可能的优化方向

- MPP环境下资源合理利用
- GPU计算资源的利用
- 统计信息的完善和充分利用
- 基于维度的预计算

The end

谢谢！