扩展Spark引擎支持MPP计算场景
——替换大规模企业级传统数据仓库

张成松

zhangcs2@Lenovo.com

May 12, 2017

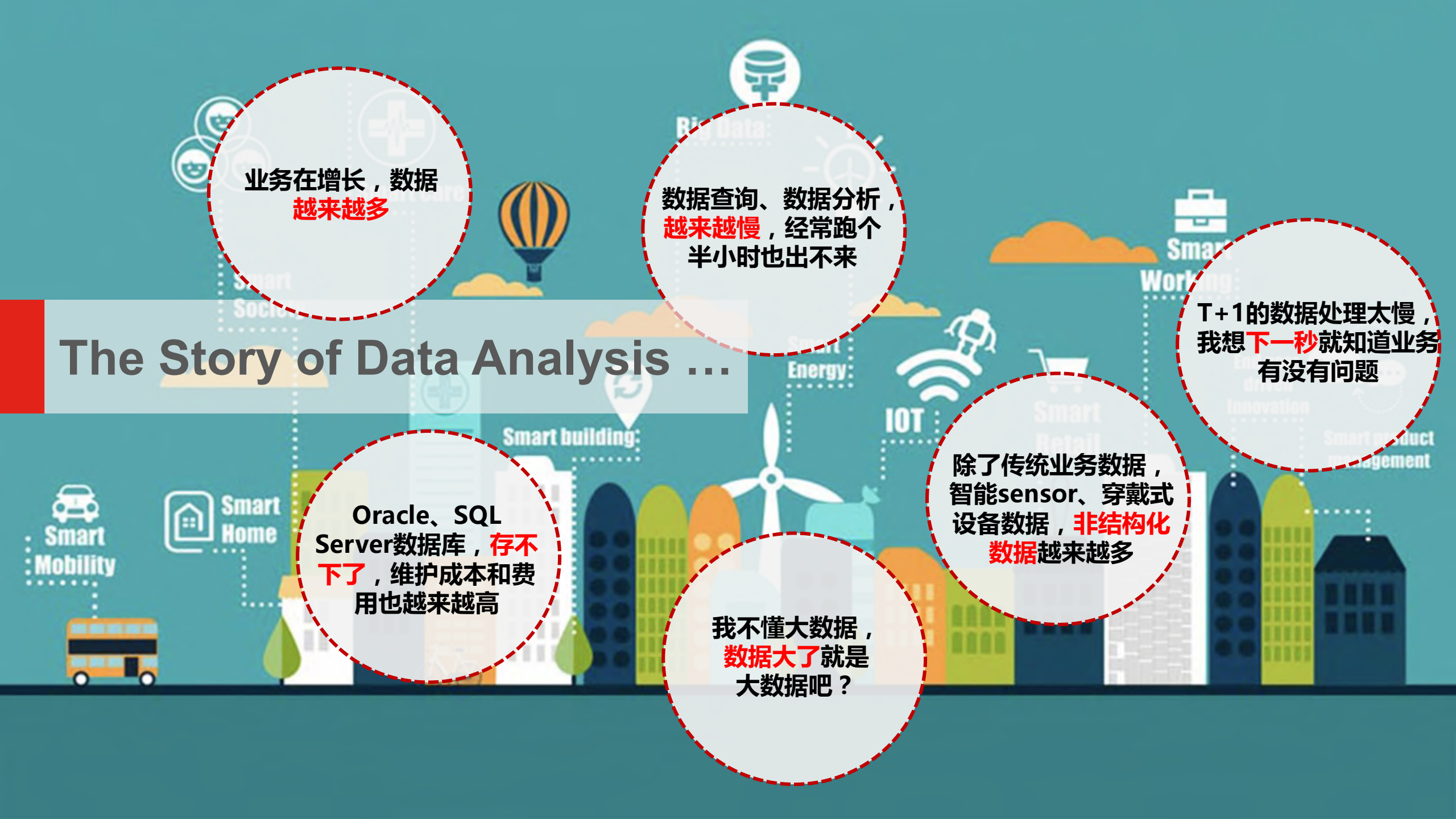
个人简介--张成松

- 联想大数据研发总监
- 联想大数据平台首席架构师
  （从无到有，全球化大数据平台，9 IDC、2000+节点)
- 互联网&开源技术爱好者
- 非专业驴友

演讲内容

- 一个故事引出的业务场景
- 企业级数据仓库+大数据，如何碰撞出火花？
- 来点干货，如何扩展Spark支持MPP计算场景？
- Spark MPP在联想大数据企业级分析平台中的应用

一个故事引出的业务场景



The Story of Data Analysis …

传统企业级数据仓库向大数据平台转型

解决传统数据应用问题

01

- 历史数据迁移
- 与传统数据仓库&应用工具无缝集成
- 适配原有业务处理逻辑，支持数据CRUD
- 支持存储过程

解决大数据计算问题

02

- PB 级数据计算
- 结构化&非结构化数据存储、应用
- 大数据场景数据分析
- 大数据环境下的数据治理

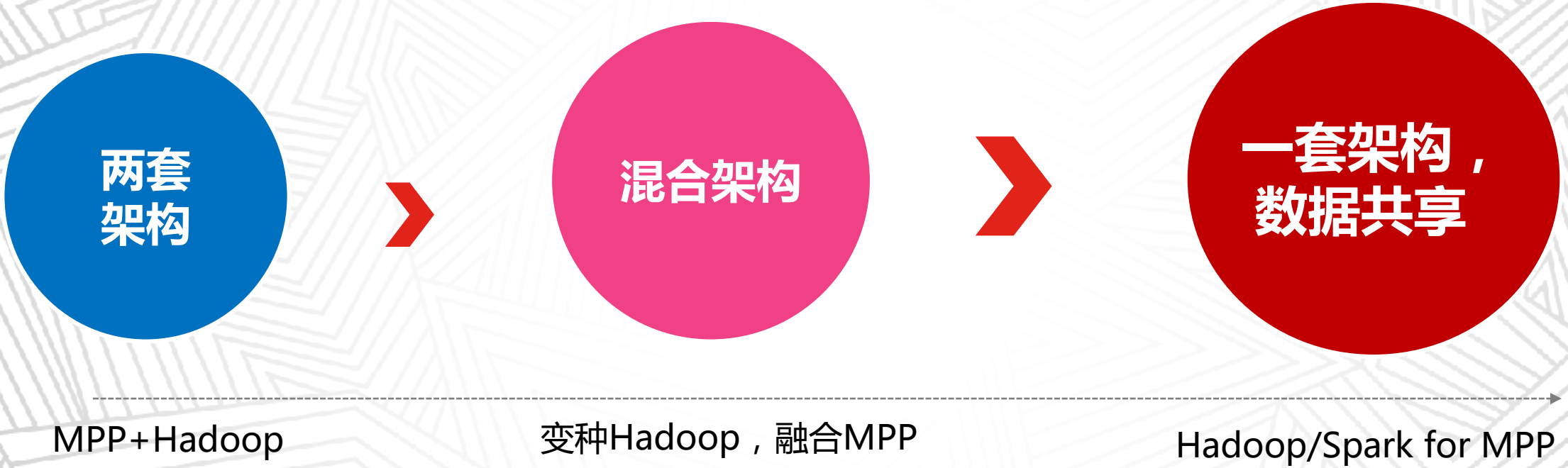
解决数据实时性处理问题

03

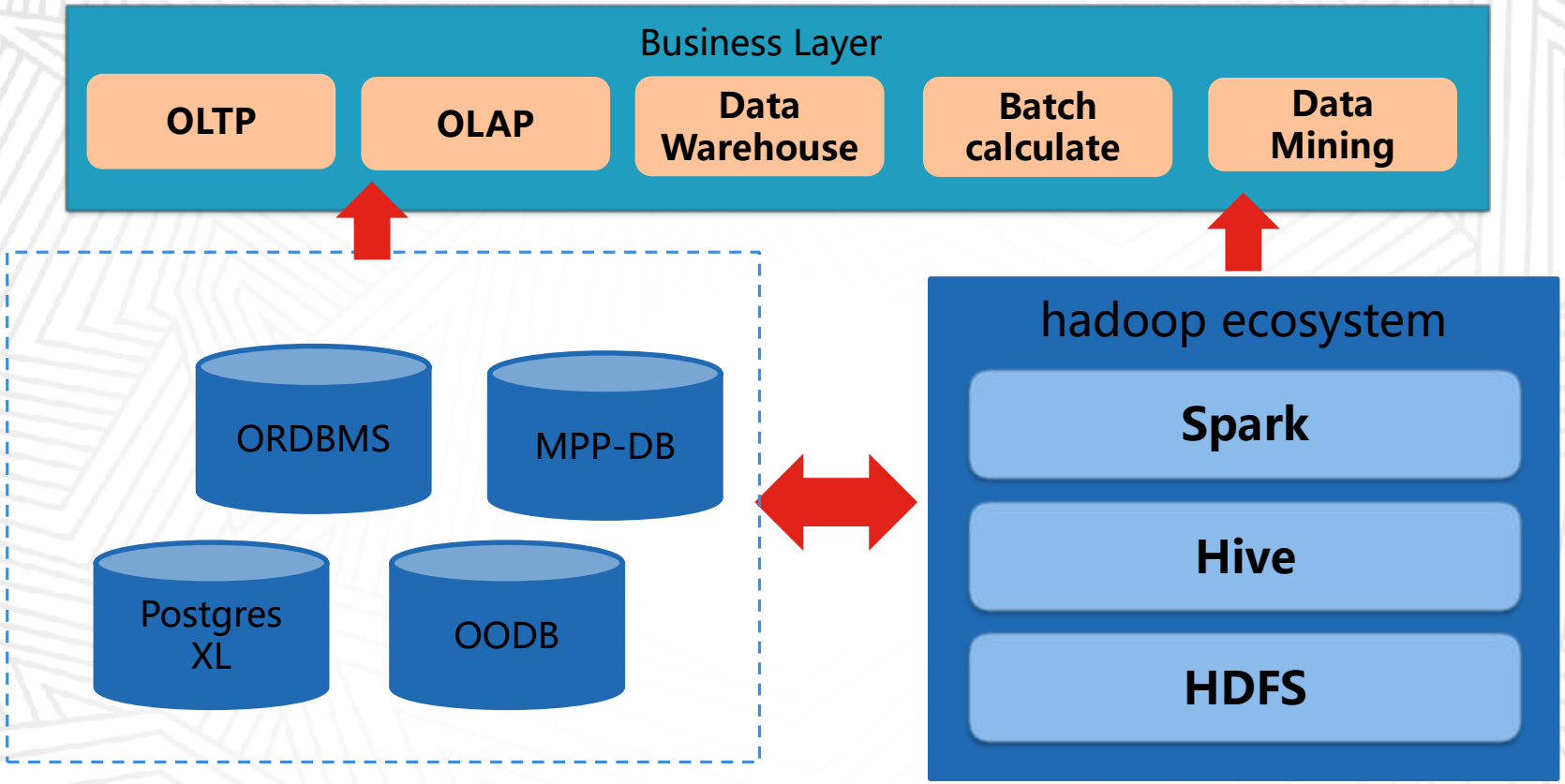
- 实时数据采集
- 流式计算、实时计算
- 交互、探索式多维数据分析

企业传统数据仓库+大数据，如何碰撞出火花？

企业级大数据平台实施方案

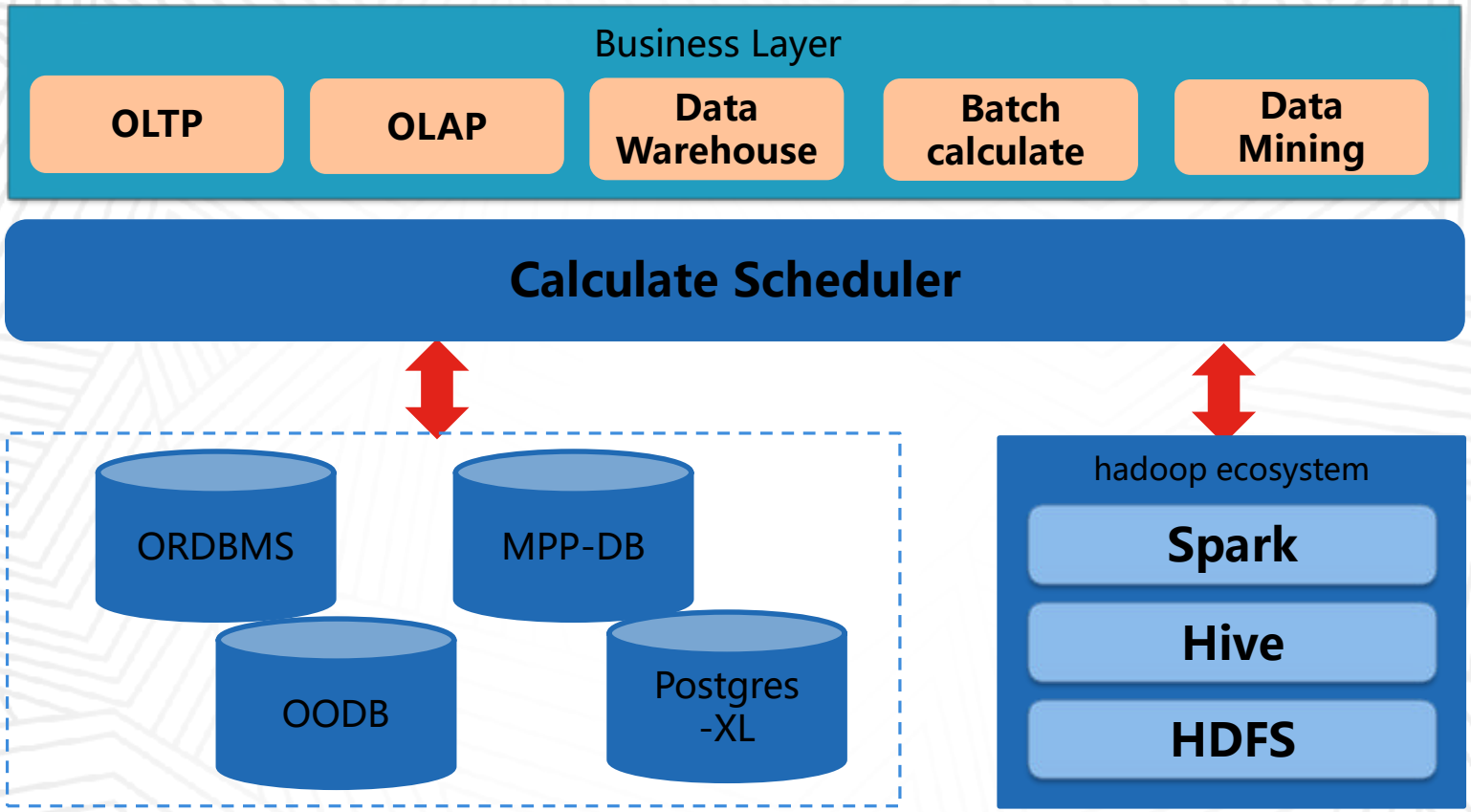


方案一：MPP+Hadoop



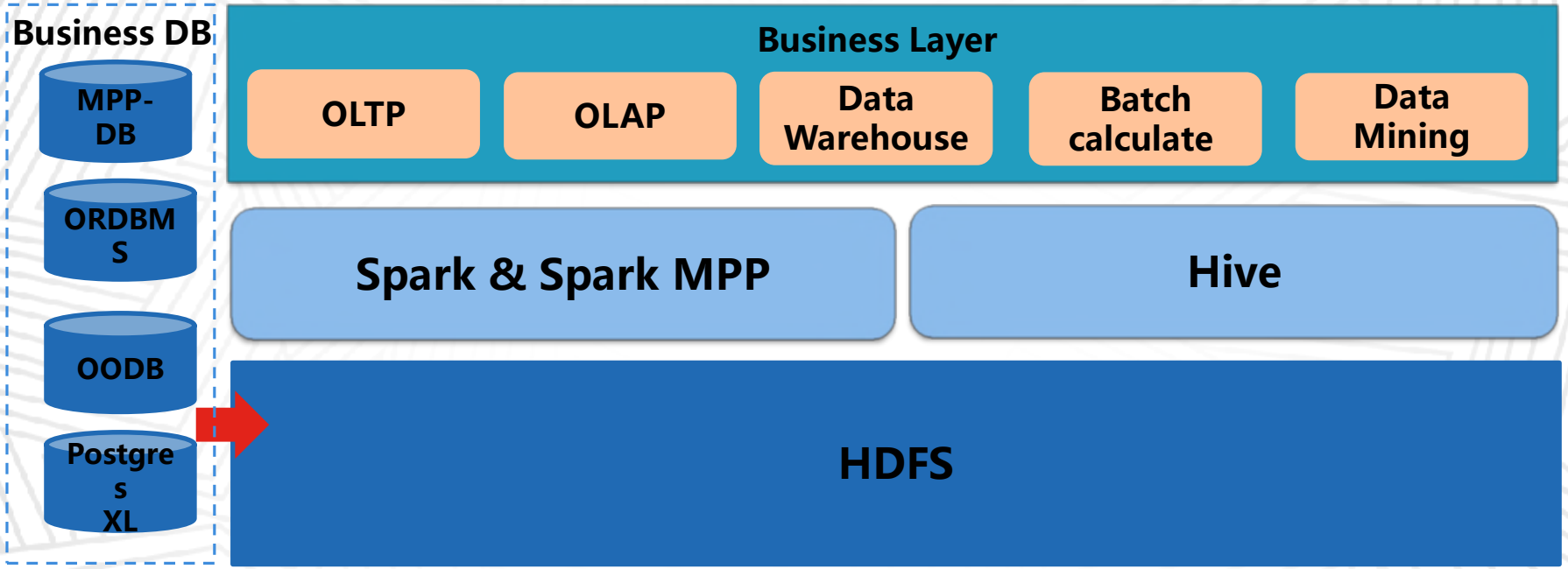
两套系统，数据独立存储，集群间数据同步

方案二：变种Hadoop，融合MPP



两套架构融合，数据各自存储，抽象存取&计算服务，按需选择计算引擎

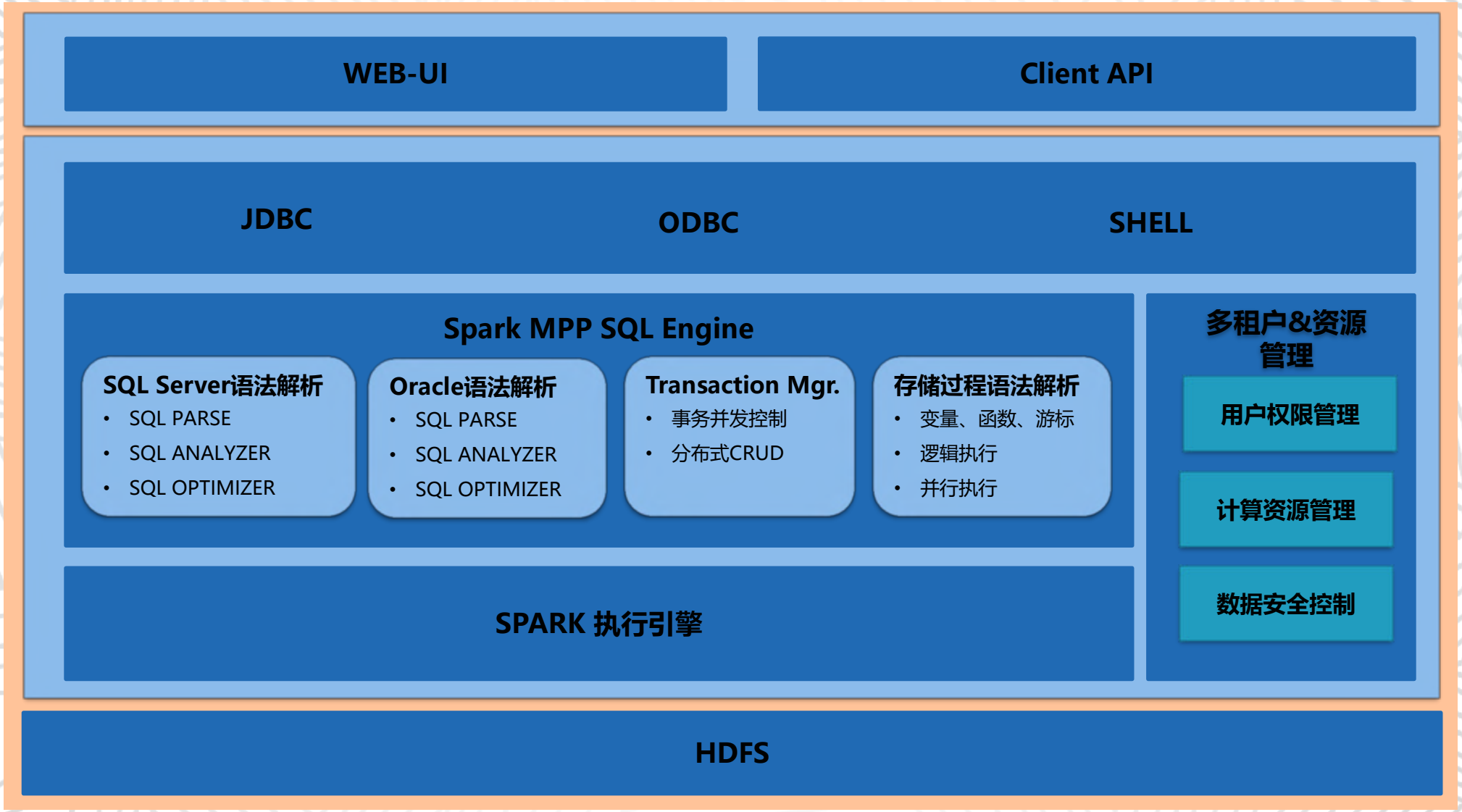
方案三： Hadoop/Spark for MPP



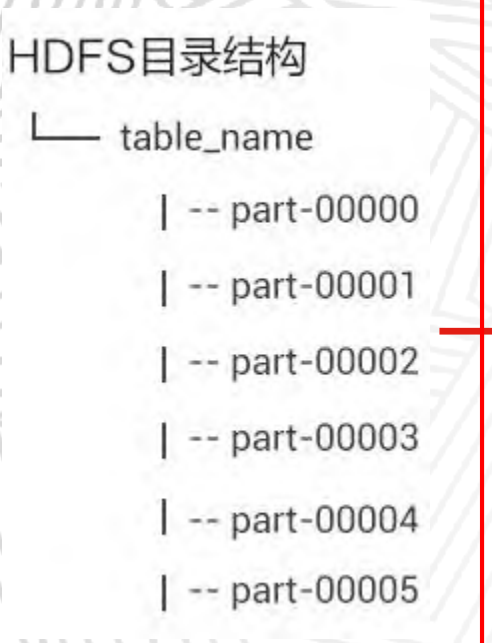
一套系统，同时支持大数据计算和传统数据仓库，导入历史数据后，数据统一存储

For 方案 3, 如何扩展Spark支持MPP计算场景？

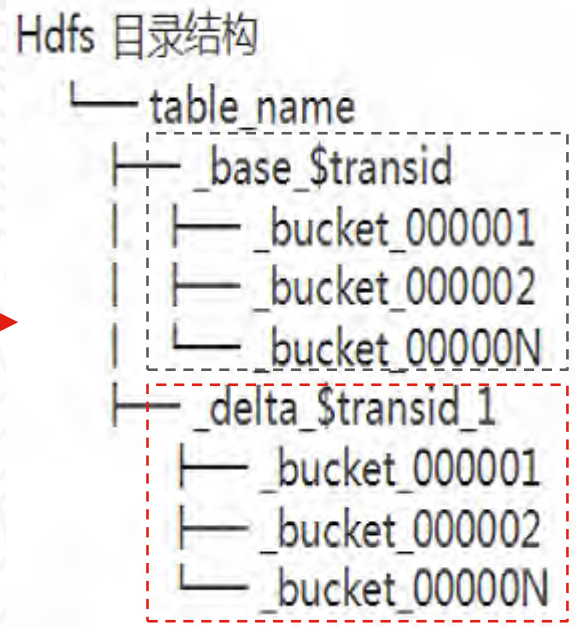
Spark MPP： 系统架构



Spark MPP： 实现数据CRUD原理



HDFS原始存储结构



上次事务，数据合并后的存储目录

新的事务，变化数据存储目录

Spark MPP方案HDFS存储结构

数据变化时

- 记录增加 → 将**新增记录**直接写入新的delta_$transid_x目录内bucket文件；
- 记录更新 → 将**更新内容**做为新记录，写入新的delta_$transid_x目录内bucket文件；
- 记录删除 → 将删除内容做为**null记录**，写入新的delta_$transid_x目录内bucket文件；

每条记录增加的信息：

{ 操作类型,
原始事务id,
rowId,
当前的事务ID }

周期性Compaction

- 新增文件数量大小监控， 当达到一定数据时自动合并
- 定时控制，周期性合并

- 仅对特定格式的ORC表有事务处理功能，事务处理的主要操作为增删改；
- 操作时开启事务模式，SET transaction.type=SparkMPP;
- ORC表必须是分桶表，在表属性里需要加上TBLPROPERTIES（“transactional” = “true”），以标识这是一个要用作事务操作的表；

Spark MPP： insert、update、delete

CREATE TABLE user_info (name STRING, age INT, sex STRING)
CLUSTERED BY (**name**) INTO 1 BUCKETS STORED AS ORC **TBLPROPERTIES ("transactional"="true");**

Sample：

1. insert into user_info values('Lilei', 6 ,'M') *;--事务id:1*
2. update user_info set age = 7 where name = 'Jim' *;--事务id:2*
3. delete user_info where name = 'Lucy' *;--事务id:3*

数据存储：

hdfs://warehouse/user_info/**base_0**/bucket_0001

Name	Age	Sex	extra_info
Lucy	5	F	{…}
Jim	6	M	{…}

数据操作

新增数据存储：

hdfs://warehouse/user_info/**delta_1**/bucket_0001

Name	Age	Sex	extra_info
Lilei	6	M	{…}

构造新记录，写入文件

hdfs://warehouse/user_info/**delta_2**/bucket_0001

Name	Age	Sex	extra_info
Jim	7	M	{…}

查询出来最大transaction_id的整条数据，修改字段后，写入文件

hdfs://warehouse/user_info/**delta_3**/bucket_0001

Name	Age	Sex	extra_info
			{…}

构造新记录(字段null)，写入文件

数据合并后

hdfs://warehouse/user_info/**base_3**/bucket_0001

Name	Age	Sex	extra_info
			{…}
Jim	7	M	{…}
Lilei	6	M	{…}

Spark MPP： select 实现逻辑

原始文件存储：

hdfs://warehouse/user_info/base_1/bucket_0001

Name	Age	Sex	extra_info
Lucy	5	F	{…}
Jim	6	M	{…}

新增文件存储：

hdfs://warehouse/user_info/**delta_2**/bucket_0001

Name	Age	Sex	extra_info
Lilei	6	M	{…}

hdfs://warehouse/user_info/**delta_3**/bucket_0001

Name	Age	Sex	extra_info
Jim	7	M	{…}

hdfs://warehouse/user_info/**delta_4**/bucket_0001

Name	Age	Sex	extra_info
			{…}

数据查询(select)处理逻辑：

1. rowsets = sort([tid asc, rowid asc , **max**(ctid)], data)

   --从base和所有delta文件中，查询符合where条件的， transaction_id最大的数据集；

2. foreach row in rowsets *--构造返回结果集*

   2.1 if row.optype = **insert or update**： **read_lines** += row

   --如果操作标志为inser or update，把记录写入返回结果集；

   2.2 if row.optype = **delete** : continue

   --如果操作标志为delete，继续；

3. return **read_lines**

   --返回构造的结果集；

Spark MPP：测试结果

Insert测试：

```
0: jdbc:hive2://10.0.64.72:10000/> insert into file_orc values('name001',1),('name002',2);
+---------+--+
| Result  |
+---------+--+
+---------+--+
No rows selected (2.767 seconds)
0: jdbc:hive2://10.0.64.72:10000/> select * from file_orc;
+----------+------+--+
|   name   | age  |
+----------+------+--+
| name002  | 2    |
| name001  | 1    |
+----------+------+--+
2 rows selected (2.657 seconds)
```

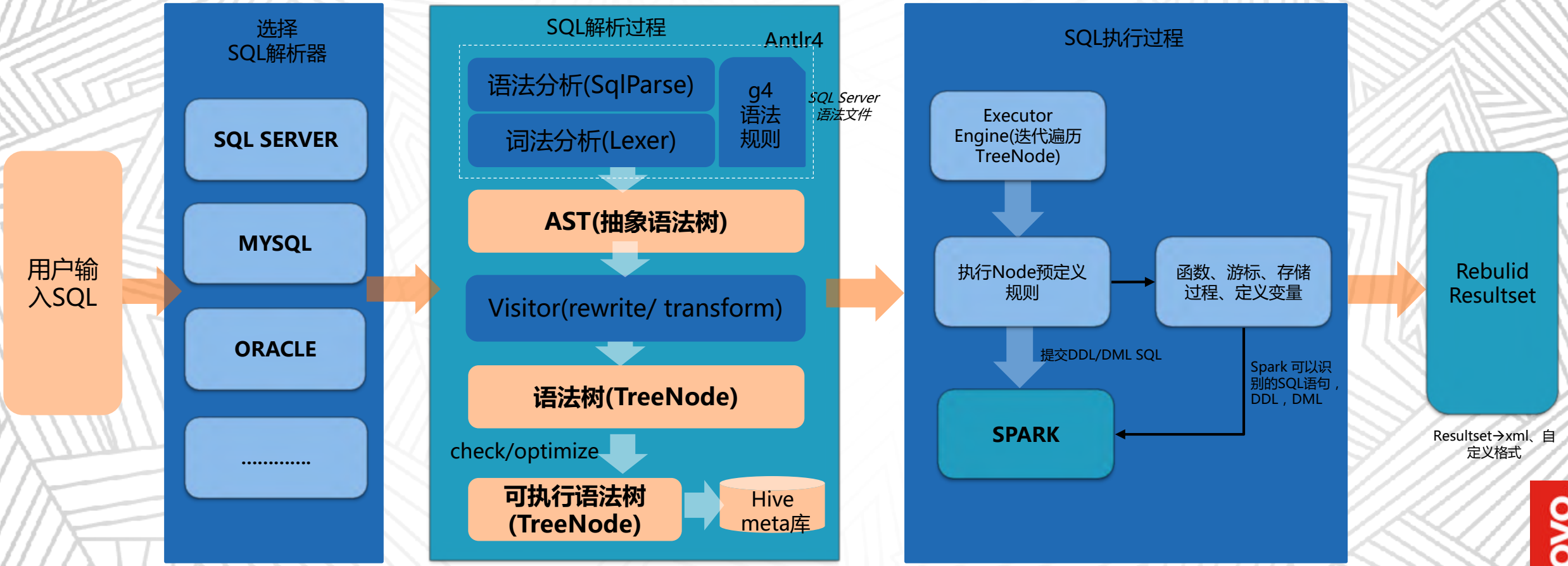
Update测试：

```
0: jdbc:hive2://          10000/> select * from file_orc;
+-------------+------+--+
|     name    | age  |
+-------------+------+--+
| athmodeus   | 29   |
| leon        | 56   |
| kakath      | 30   |
| eve         | 25   |
+-------------+------+--+
4 rows selected (0.127 seconds)
0: jdbc:hive2://          10000/> update file_orc set name='athmodeus' where age='30';
+---------+--+
| Result  |
+---------+--+
+---------+--+
No rows selected (0.519 seconds)
0: jdbc:hive2://          10000/> select * from file_orc;
+-------------+------+--+
|     name    | age  |
+-------------+------+--+
| leon        | 56   |
| athmodeus   | 30   |
| eve         | 25   |
+-------------+------+--+
```

Delete测试：

```
0: jdbc:hive2://          0000/> delete from file_orc where name='athmodeus';
+---------+--+
| Result  |
+---------+--+
+---------+--+
No rows selected (0.549 seconds)
0: jdbc:hive2://          0000/> select * from file_orc;
+-------+------+--+
| name  | age  |
+-------+------+--+
| leon  | 56   |
| eve   | 25   |
+-------+------+--+
2 rows selected (0.149 seconds)
```

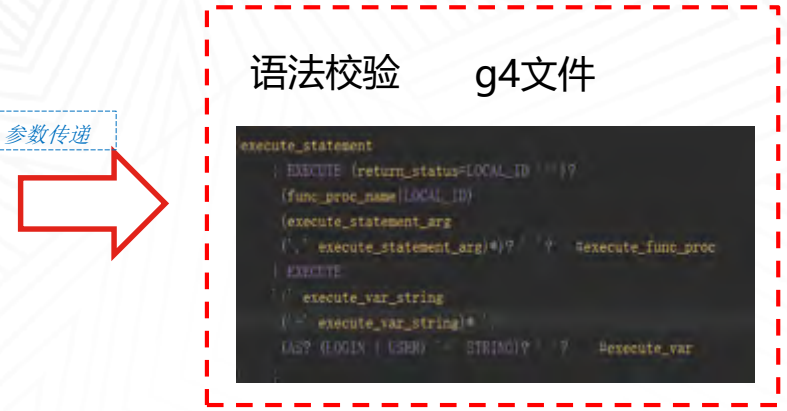
Spark MPP ：存储过程&SQL语句执行逻辑



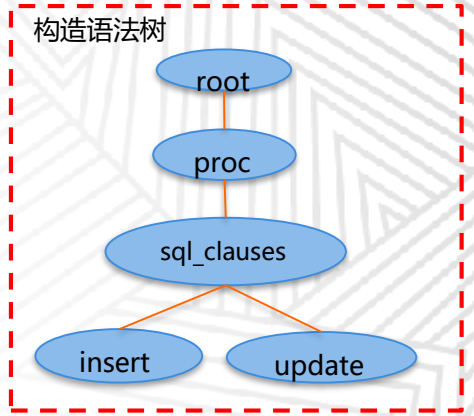
Spark MPP： 创建存储过程

```
drop PROCEDURE test_a_out
GO
CREATE PROCEDURE test_a_out
@name VARCHAR(50) = 'default_value' output,
@id int = 27
AS
print @name
print @id
set @name = 'zhong'
INSERT INTO b1(name) VALUES(@name)
set @name = 'deyin'
UPDATE b1 SET name = @name where name = 'zhong'
DELETE from b1 where name = @id
return 20170206
go

DECLARE @nm_value varchar(50) = 'nm_value'
DECLARE @return_status varchar(50)
EXEC @return_status = test_a_out @nm_value OUTPUT,
1
print @return_status
print @nm_value
go
```

支持Input/output/return 参数传递

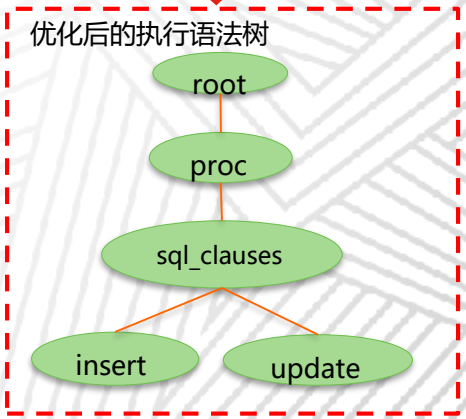


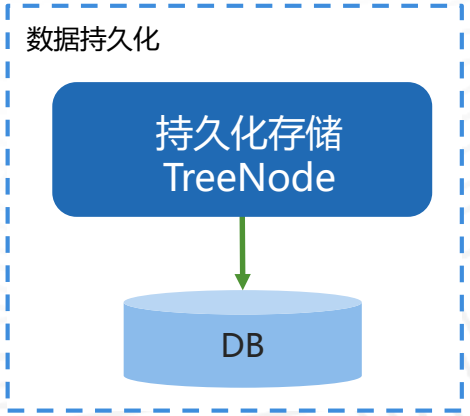
语法解析

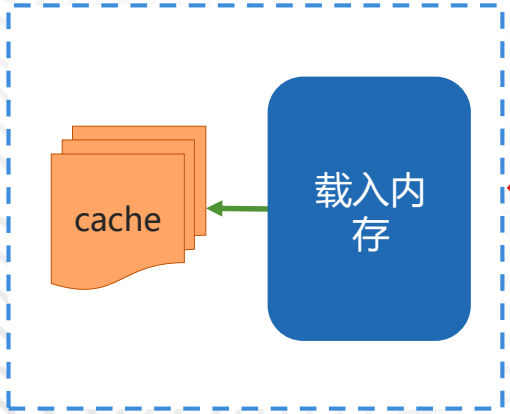


存储过程是否存在？
(名称/MD5)

optimize

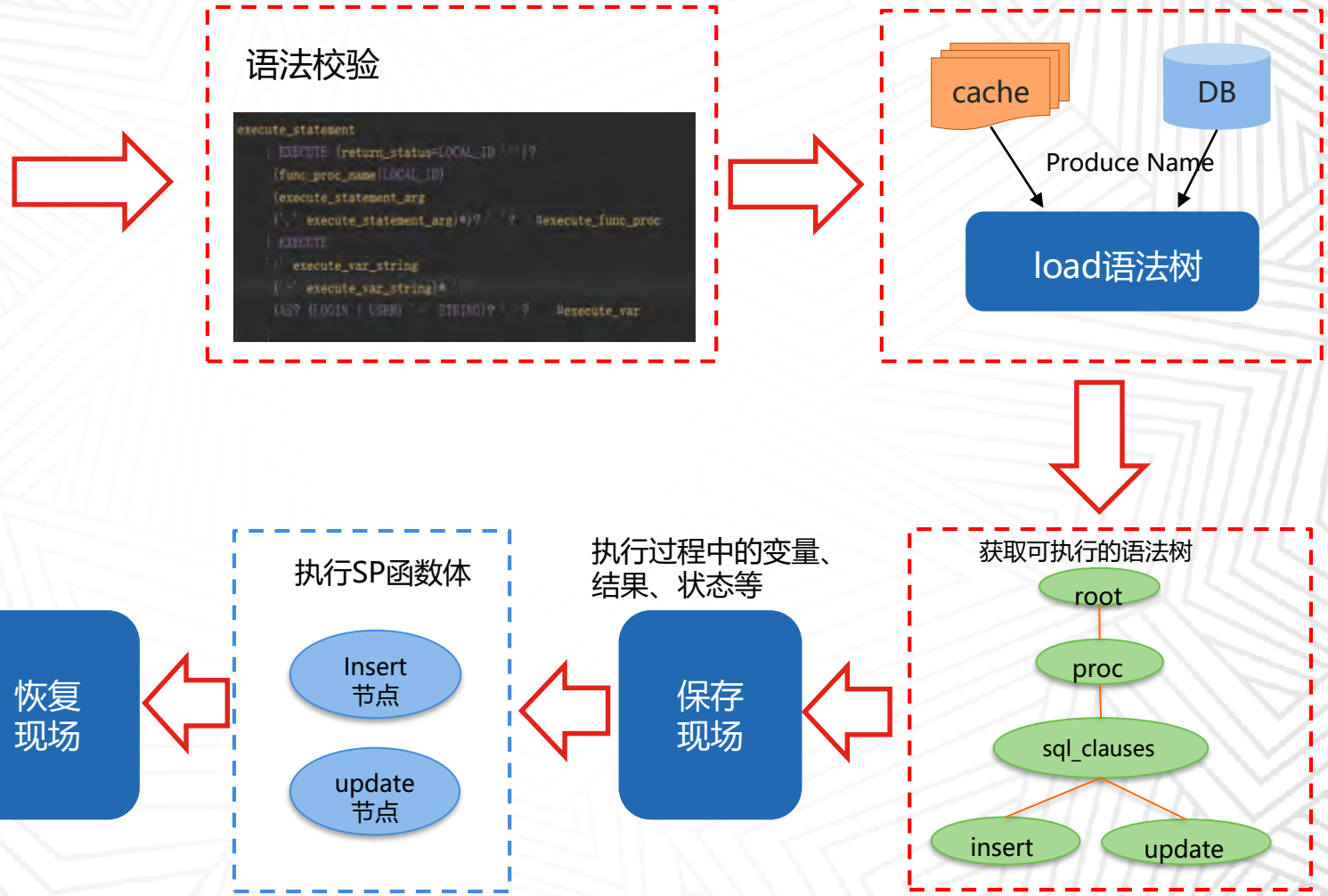






Spark MPP： 执行存储过程

```
DECLARE @nm_value
varchar(50) = 'nm_value'
DECLARE @return_status
varchar(50)
EXEC @return_status =
test_a_out @nm_value
OUTPUT, 1
print @return_status
print @nm_value
go
```

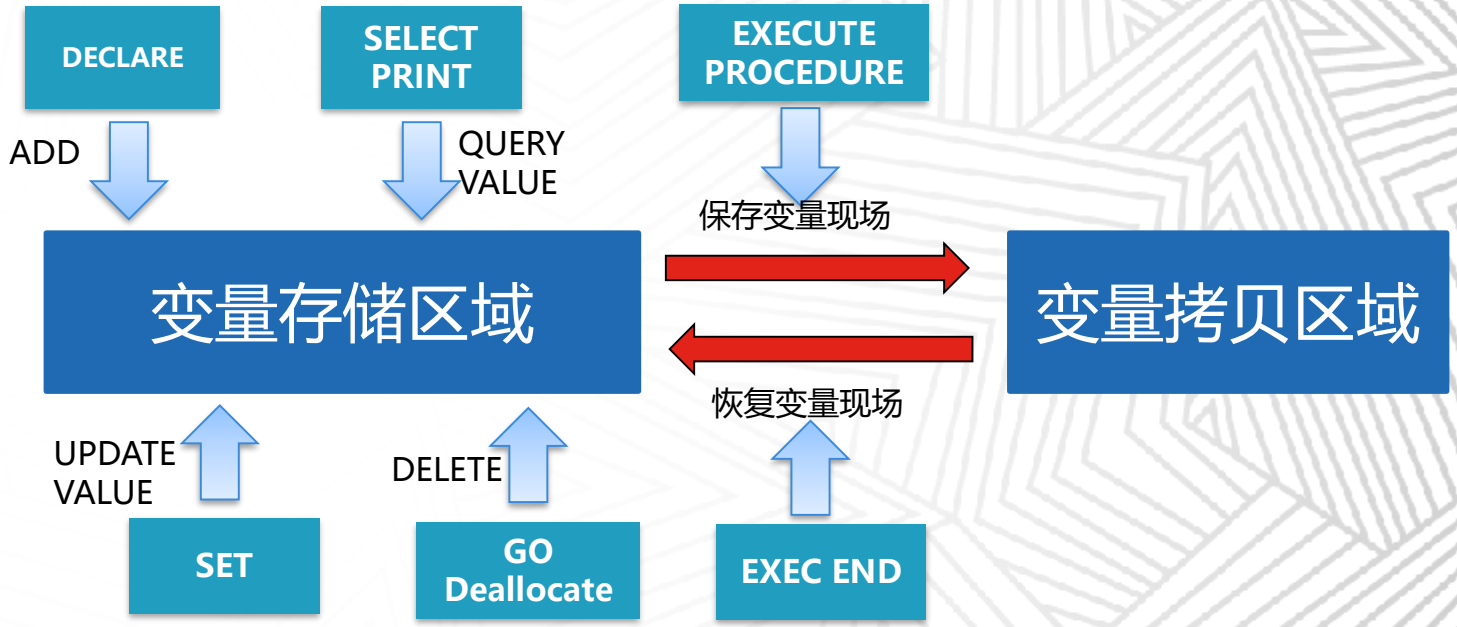


Spark MPP： 变量

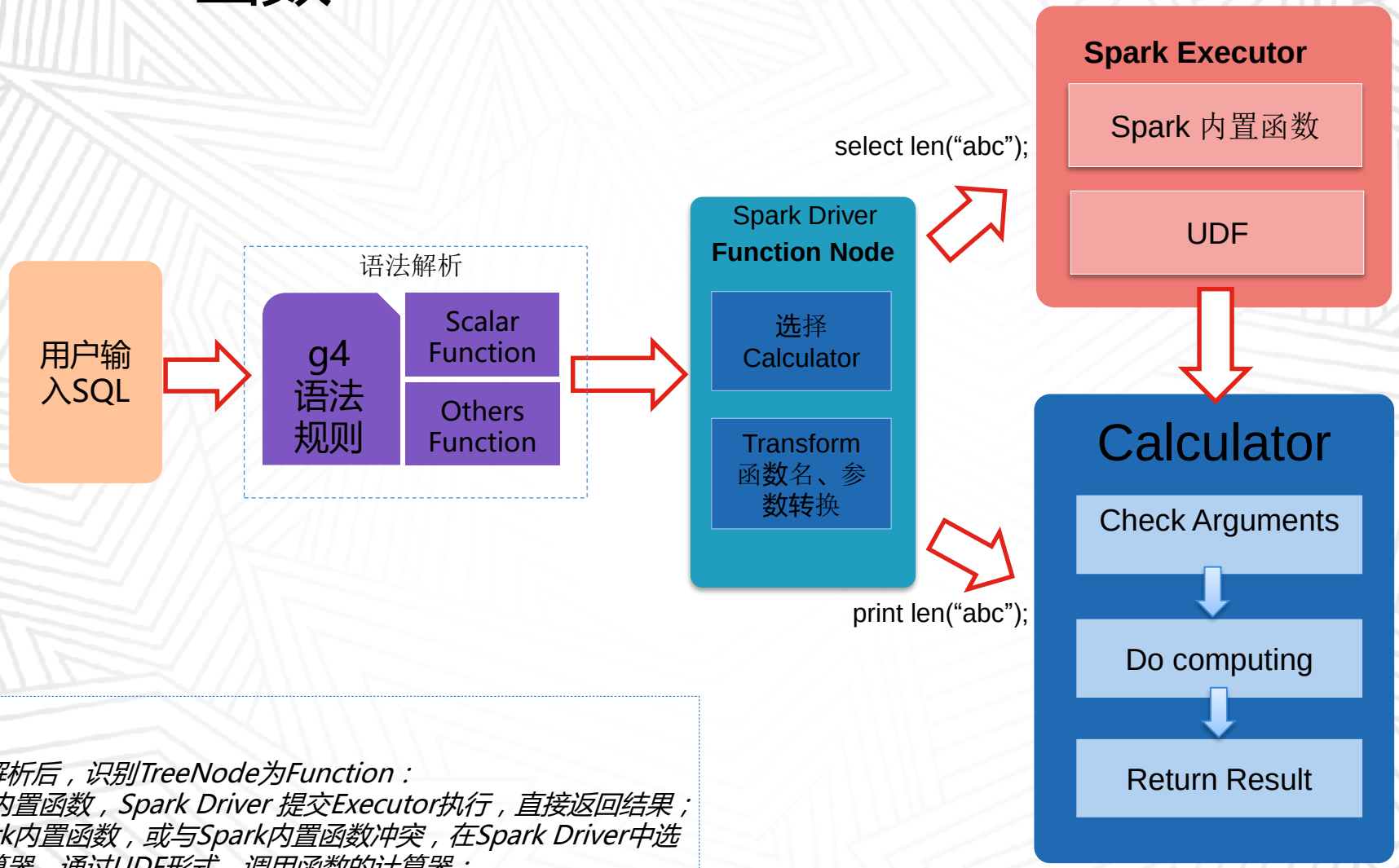
变量隔离：

GO

存储过程函数体与外部变量



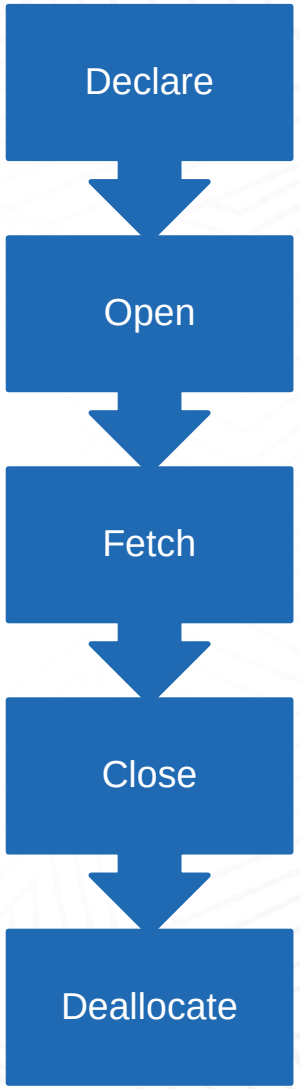
Spark MPP： 函数



注：语法解析后，识别TreeNode为Function：
1、Spark内置函数，Spark Driver 提交Executor执行，直接返回结果；
2、非Spark内置函数，或与Spark内置函数冲突，在Spark Driver中选择函数计算器，通过UDF形式，调用函数的计算器；

Spark MPP：游标

```
Examples:
DECLARE @id bigint, @name nvarchar(100), @age int

DECLARE cursor_name SCROLL CURSOR FOR select id,name,age
from tmp.tb_cursor
OPEN cursor_name

FETCH next FROM cursor_name into @id, @name, @age
insert into tmp.tb_cursor_result values(@id, @name, @age)

while @@fetch_status = 0
BEGIN
    FETCH next FROM cursor_name into @id, @name, @age
        if @id % 2 <> 0
        begin
            insert into tmp.tb_cursor_result values(@id, @name, @age)
        end
end

close cursor_name
deallocate cursor_name
go

select * from tmp.tb_cursor_result
```



1. Global/Local/Variable
2. Save to Variable Container

1. Find cusor by name
2. Execute cursor(select statement)
3. Save Resultset to cursor

1. Find cusor by name
2. Move pointer in cursor result
3. Fetch row from resultset
4. Assign value to into variables

1. Set the cursor status to CLOSE
2. Clear cursor resultset

1. Delete cursor from Variable Container

Spark MPP： 测试结果

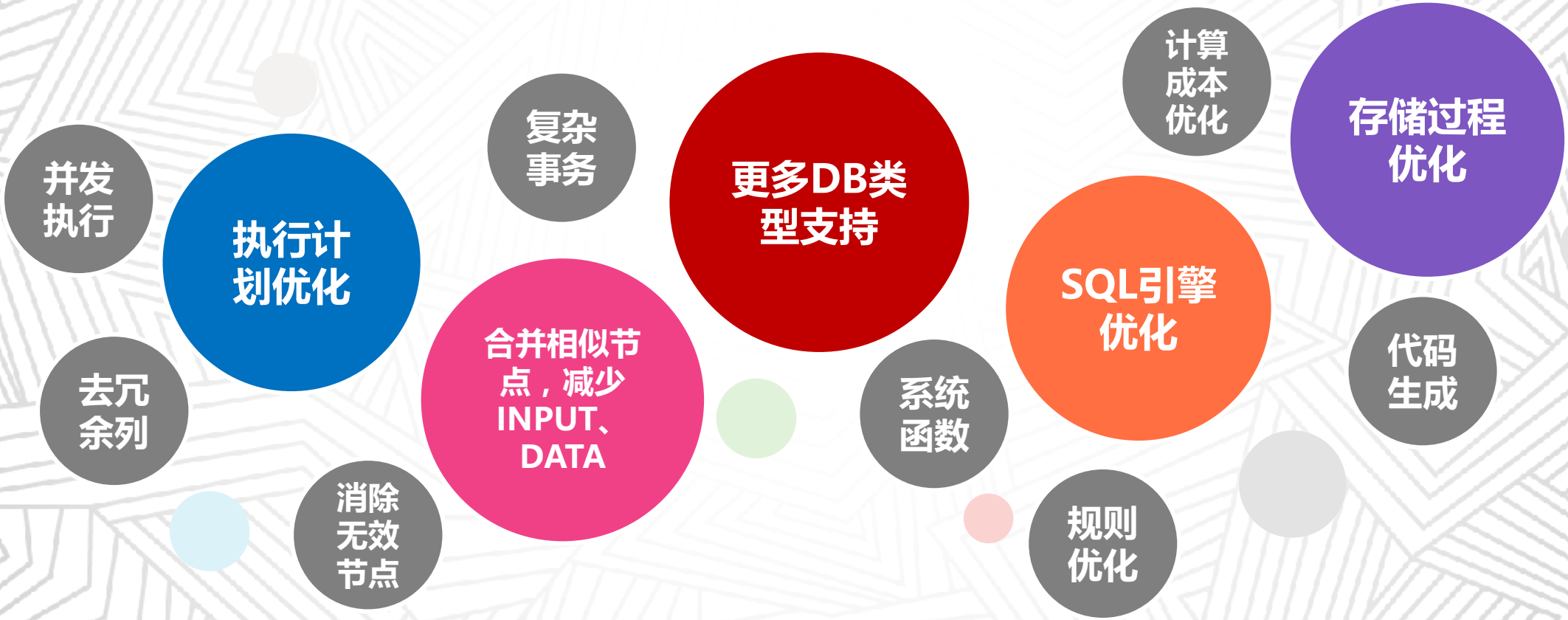
```
0: jdbc:hive2://10.0.64.72:10000/> -- 如果存储过程存在，删除
0: jdbc:hive2://10.0.64.72:10000/> if object_id('test_proc', N'p') is not null
0: jdbc:hive2://10.0.64.72:10000/> drop PROCEDURE test_proc
0: jdbc:hive2://10.0.64.72:10000/> GO
0: jdbc:hive2://10.0.64.72:10000/>
0: jdbc:hive2://10.0.64.72:10000/> -- 创建存储过程
0: jdbc:hive2://10.0.64.72:10000/> -- name中含有e，并且年龄大于25
0: jdbc:hive2://10.0.64.72:10000/> CREATE PROCEDURE test_proc @name VARCHAR(50) = 'e', @age int = 25
0: jdbc:hive2://10.0.64.72:10000/> AS
0: jdbc:hive2://10.0.64.72:10000/> insert into tmp.tb_proc_result(id, name, age)
0: jdbc:hive2://10.0.64.72:10000/> select id, name, age from tmp.tb_proc
0: jdbc:hive2://10.0.64.72:10000/> where name like concat('%', @name, '%') and age >= @age
0: jdbc:hive2://10.0.64.72:10000/>
0: jdbc:hive2://10.0.64.72:10000/> return 20170321
0: jdbc:hive2://10.0.64.72:10000/> go;
+---------+--+
| value   |
+---------+--+
+---------+--+
No rows selected (0.051 seconds)
0: jdbc:hive2://10.0.64.72:10000/>
```

```
0: jdbc:hive2://10.0.64.72:10000/> -- 执行存储过程
0: jdbc:hive2://10.0.64.72:10000/> declare @name nvarchar(50) = 'i'
0: jdbc:hive2://10.0.64.72:10000/> declare @age int = 27
0: jdbc:hive2://10.0.64.72:10000/> DECLARE @return_status int
0: jdbc:hive2://10.0.64.72:10000/>
0: jdbc:hive2://10.0.64.72:10000/> EXEC @return_status = test_proc @age = @age, @name = @name
0: jdbc:hive2://10.0.64.72:10000/> insert into tmp.tb_proc_result values(@return_status, 'success', 0)
0: jdbc:hive2://10.0.64.72:10000/> go
0: jdbc:hive2://10.0.64.72:10000/>
0: jdbc:hive2://10.0.64.72:10000/> -- 查看结果
0: jdbc:hive2://10.0.64.72:10000/> select * from tmp.tb_proc_result;
+-----------+-----------+------+--+
|    id     |   name    | age  |
+-----------+-----------+------+--+
| 20170321  | success   | 0    |
| 8         | eight     | 28   |
| 9         | nine      | 29   |
+-----------+-----------+------+--+
3 rows selected (1.906 seconds)
0: jdbc:hive2://10.0.64.72:10000/>
```

```
0: jdbc:hive2://10.0.64.72:10000/> -- 定义并使用游标
0: jdbc:hive2://10.0.64.72:10000/> DECLARE @id bigint, @name nvarchar(100), @age int
0: jdbc:hive2://10.0.64.72:10000/>
0: jdbc:hive2://10.0.64.72:10000/> DECLARE cursor_name SCROLL CURSOR FOR select id,name,age from tmp.tb_cursor
0: jdbc:hive2://10.0.64.72:10000/> OPEN cursor_name
0: jdbc:hive2://10.0.64.72:10000/>
0: jdbc:hive2://10.0.64.72:10000/> FETCH next FROM cursor_name into @id, @name, @age
0: jdbc:hive2://10.0.64.72:10000/> insert into tmp.tb_cursor_result values(@id, @name, @age)
0: jdbc:hive2://10.0.64.72:10000/>
0: jdbc:hive2://10.0.64.72:10000/> while @@fetch_status = 0
0: jdbc:hive2://10.0.64.72:10000/> BEGIN
0: jdbc:hive2://10.0.64.72:10000/> FETCH next FROM cursor_name into @id, @name, @age
0: jdbc:hive2://10.0.64.72:10000/>
0: jdbc:hive2://10.0.64.72:10000/> if @id % 2 <> 0
0: jdbc:hive2://10.0.64.72:10000/> begin
0: jdbc:hive2://10.0.64.72:10000/> insert into tmp.tb_cursor_result values(@id, @name, @age)
0: jdbc:hive2://10.0.64.72:10000/> end
0: jdbc:hive2://10.0.64.72:10000/> end
0: jdbc:hive2://10.0.64.72:10000/>
0: jdbc:hive2://10.0.64.72:10000/> close cursor_name
0: jdbc:hive2://10.0.64.72:10000/> deallocate cursor_name
0: jdbc:hive2://10.0.64.72:10000/> go
0: jdbc:hive2://10.0.64.72:10000/>
0: jdbc:hive2://10.0.64.72:10000/> -- 查看结果
0: jdbc:hive2://10.0.64.72:10000/> select * from tmp.tb_cursor_result;
+-----+---------+------+--+
| id  |  name   | age  |
+-----+---------+------+--+
| 1   | one     | 21   |
| 5   | five    | 25   |
| 7   | seven   | 27   |
| 3   | three   | 23   |
| 9   | nine    | 29   |
+-----+---------+------+--+
5 rows selected (2.046 seconds)
0: jdbc:hive2://10.0.64.72:10000/>
```

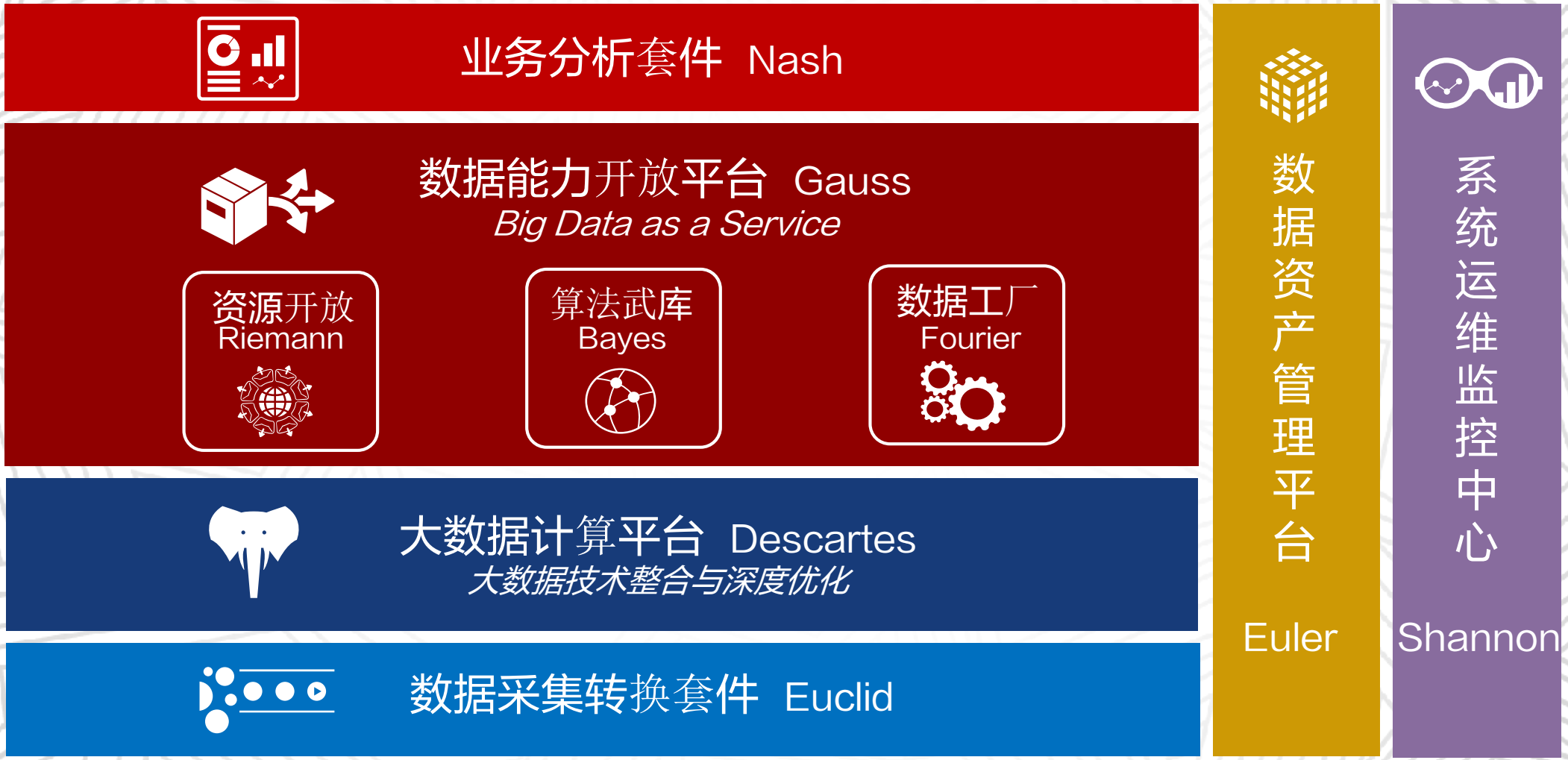
优化Spark MPP，不断提升性能

实现了数据的CRUD操作，支持存储过程、游标、函数、变量等逻辑，仅仅是Spark架构实现MPP功能的第一步。为了提升执行效率和性能，还需要从不同层面对Spark MPP引擎进行优化。

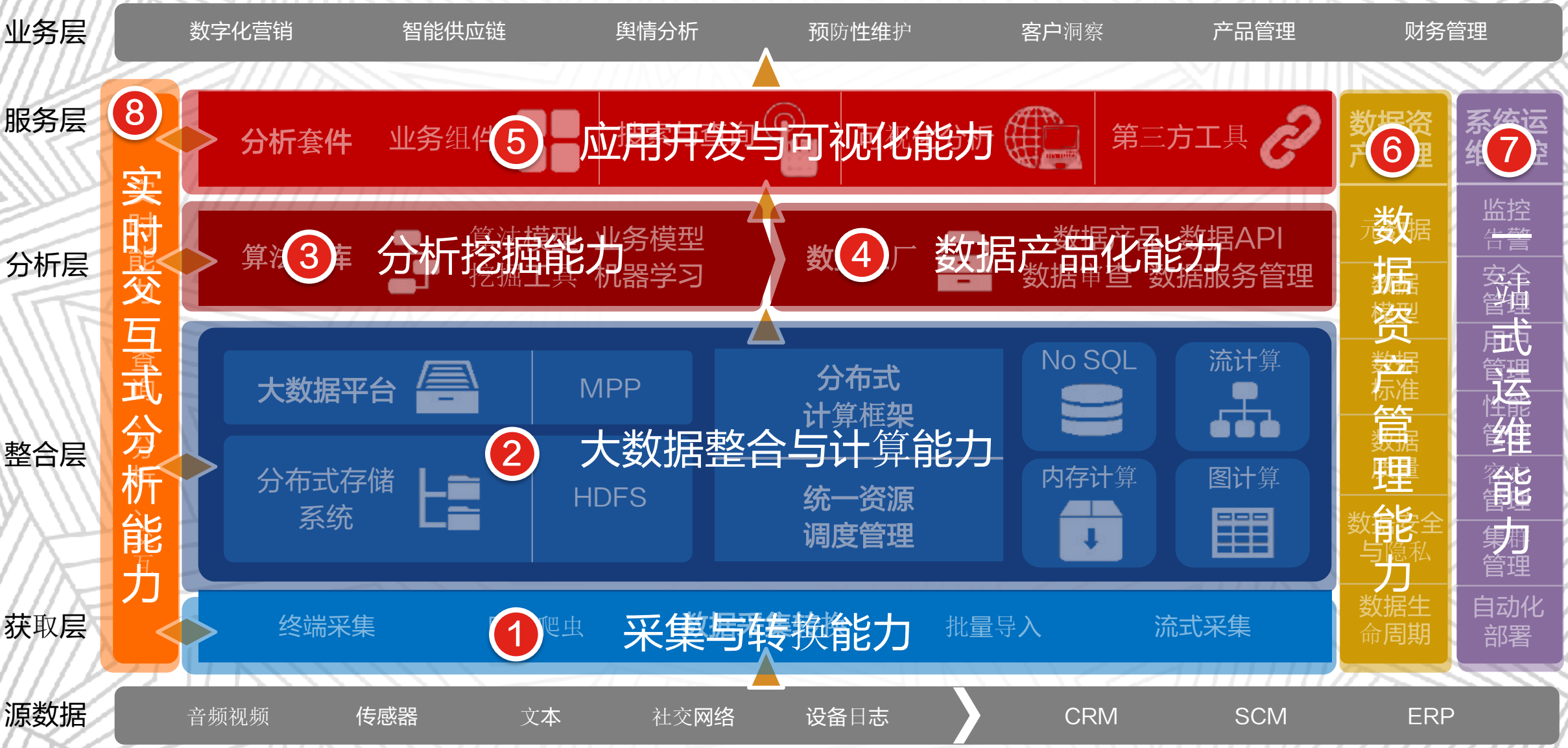


联想大数据企业级分析平台（LEAP）

联想大数据企业级分析平台解决方案（LEAP）



联想大数据企业级分析平台解决方案（LEAP）



联想大数据具有强大的研发和实施能力

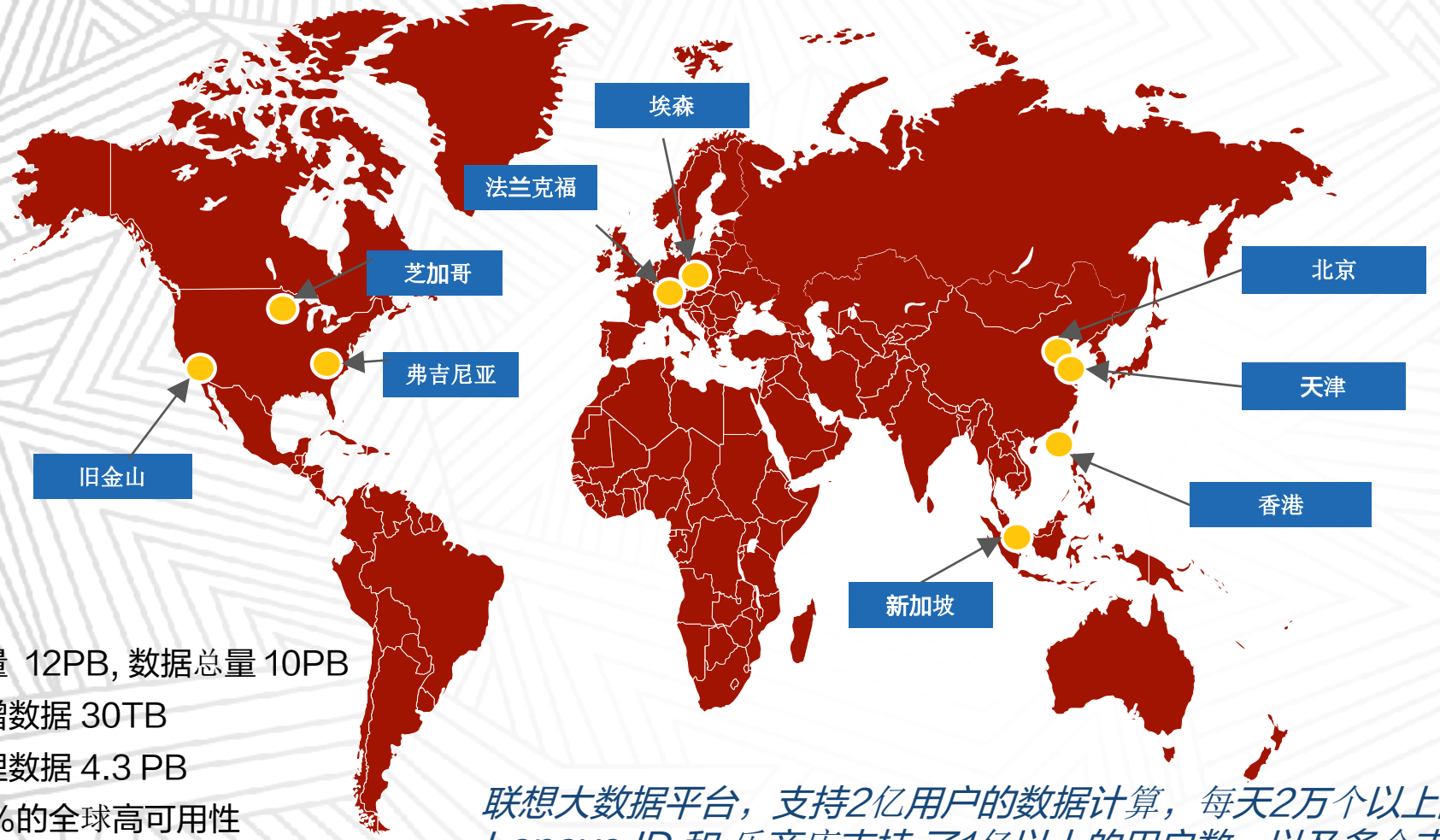
6年！
300+研发人员持续投入

三个研发中心
200+名大数据研发工程师
60+名大数据平台运维工程师

40+名数据科学家
来自中科院、清华、北大、牛津、港大、港科大以及美国、澳洲等著名学府的博士和硕士人才
博士与海归比例超过80%
在顶级期刊和会议中发表学术论文近百篇，国内和美国专利数十余项。

20+名大数据领域专家
数据专家：平台架构、数据架构、数据标准、数据治理、数据管控等
业务专家：制造业、零售业、政府、医疗行业、能源与公共事业、通讯业、金融业等

全球部署的超大规模集群，PB级与复杂业务生态的实战锤炼



- 全球化多中心部署，2000台服务器，3000名操作用户
- 在实践中充分验证系统的高可靠性
- 企业数据本地化收集和存储
- 完全合规各国数据保护和隐私保护法律

- 总容量 12PB, 数据总量 10PB
- 日新增数据 30TB
- 日处理数据 4.3 PB
- 99.5%的全球高可用性

联想大数据平台，支持2亿用户的数据计算，每天2万个以上的计算任务，Lenovo ID 和 乐商店支持 了1亿以上的用户数，以及多个支持百万以上用户数的应用和服务

每隔5秒，联想全球的新增设备信息会被实时提取和计算，实时优化全球业务流程


THANK YOU