DTCC

2015中国数据库技术大会

DATABASE TECHNOLOGY CONFERENCE CHINA 2015

大数据技术探索和价值发现

Wing - 新一代百度大数据查询引擎

刘成 百度大数据部QE团队

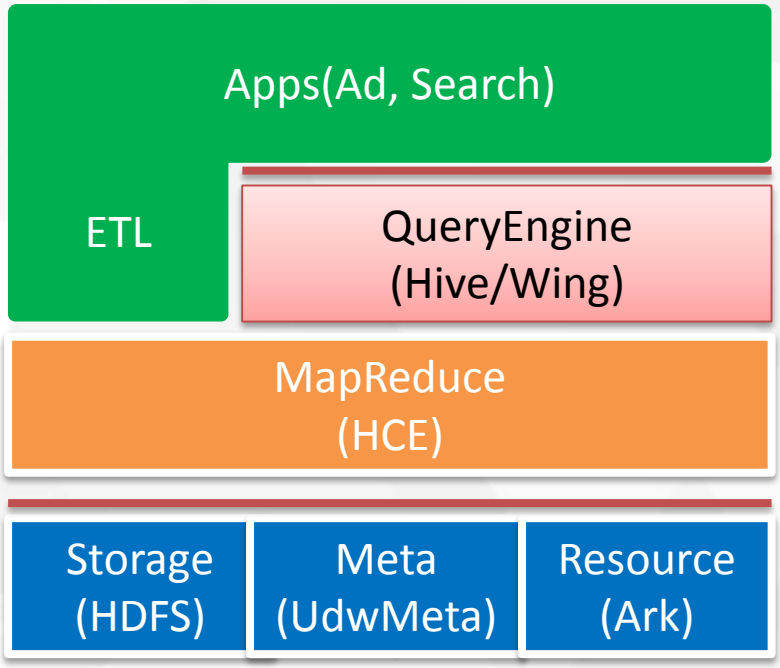
2015-04-13

概览

- QueryEngine技术介绍
  - 技术发展、架构介绍
- Wing的语义接口
  - 语法、语义系统
- 执行优化
  - 降低数据传输、提升计算效率
- 性能结果
  - runtime性能要比hive提高30%
  - 百度线上query性能提升达4倍

QueryEngine

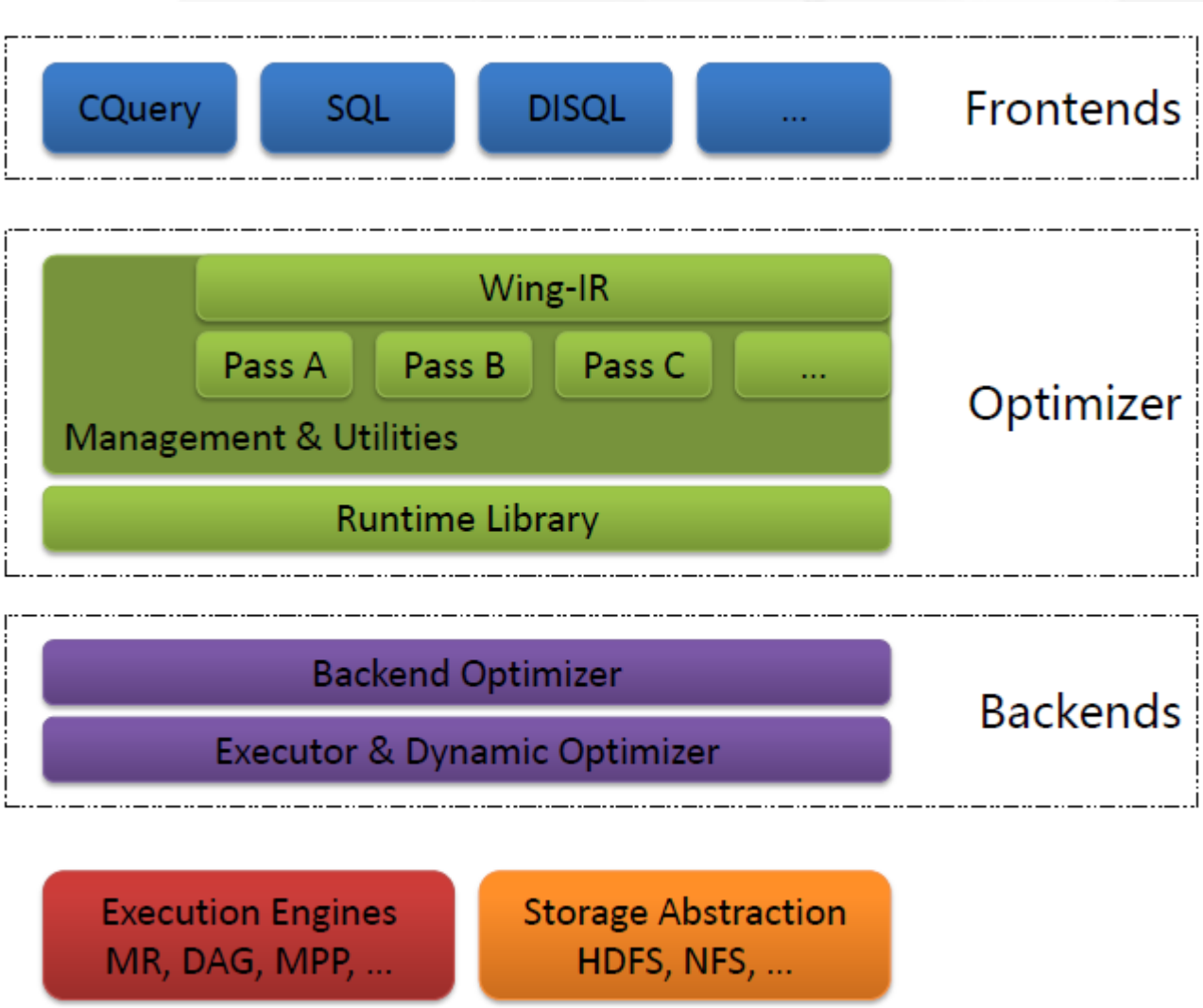
- QueryEngine是一个编译器
  - 编译：高级语言描述查询(HQL, Pig Latin)
  - 优化：理解查询，优化用户计算逻辑
- 大量QueryEngine系统涌现
  - 批处理：Hive/Pig
  - 交互式：Spark SQL/Dremel/Impala/Apache Drill
  - 流式计算：Storm
- QueryEngine与MR/Tez/Spark关系
  - 更高层的抽象
  - 更易用接口，充分利用不同框架计算能力



百度QueryEngine 2.0

- QueryEngine 1.0： Hive
  - 与社区版本严重脱节 0.8.1 vs 0.13.1
  - 代码可维护性差，不易做定制性优化，bugfix困难
  - HQL不易嵌入用户自定义逻辑
- QueryEngine 2.0： Wing
  - 目标：结构化数据处理引擎的公共组件
  - 接口：HQL、CQuery
  - 优化：基于关系模型的优化、基于llvm分析数据流优化
  - 维护性：完全由QueryEngine团队开发和维护
- 百度QueryEngine服务
  - 每天session数量: 14~15w
  - 每天处理数据量： 2P
  - 运行场景：例行任务

Wing架构设计



- Frontends: 不同的Query描述语言，产生一致的中间表示
- 语义分析：类型检查、列引用检查、函数检查、关系算子语义检查
- Optimizer: 变换中间表示，关系代数优化
- Runtime：实际的计算逻辑，表达式求值，算子求值，输入和输出
- Backend: 执行框架，驱动runtime算子执行

概览

- QueryEngine技术介绍
  - 技术发展、架构介绍
- Wing的语义接口
  - 语法、语义系统
- 执行优化
  - 降低数据传输、提升计算效率
- 性能结果
  - runtime性能要比hive提高30%
  - 百度线上query性能提升达4倍

HQL语义

- 数据组织
  - Namespace, Database, Table
  - Partition, Bucket
- 存储抽象
  - InputFormat/OutputFormat: 输入输出为二进制Record
  - Serializer/Deserializer：输入输出为Tuple
- 类型系统
  - 基本类型: boolean, int, float, double, string, binary
  - 复杂类型: enum, list, struct, map
  - 支持递归结构(指针)：
    - struct TreeNode { TreeNode* left; TreeNode* right; }

HQL：多后端meta和SessionDB

- 会话临时表
  - session = DATABASE 'session:/';
  - USE session;
  - CREATE TABLE cnt_distinct_20130310 AS SELECT count(DISTINCT baiduapp_uid) FROM default.udwetl_bd_input WHERE event_day=20130310;

- 多存储后端
  - 方便跨后端查询数据
  - Hive = DATABASE '…';
  - UDW = DATABASE '…'
  - MYSQL = DATABASE '…'
  - Select * from Hive.t1 join UDW.t2 on cond1 join MYSQL.t3 on cond2;

CQuery接口

- 数据抽象：Table
- 操作抽象：
  - Load、Sink、Select、Aggregate等

- 示例代码

```
int main(int argc, char **argv) {
    DECLARE_FUNCTION(trim_heading_space);
    DECLARE_AGGREGATE_FUNCTION(count, CountUDAF);

    CQuery::init(argc, argv);

    Table user = Table::load("user.data", "user_property.proto", "User", "SequenceFile");

    Table props = Table::load("user_property.data", "user_property.proto",
                  "UserProperty", "SequenceFile");

    Table teenager = user->filter("age >= 10 && age < 20")->select("name, id");

    teenager->output_overwrite_file("teenager.data", "teenager.proto", "Teenager");
```

概览

- QueryEngine技术介绍
  - 技术发展、架构介绍
- Wing的语义接口
  - 语法、语义系统
- 执行优化
  - 降低数据传输、提升计算效率
- 性能结果
  - runtime性能要比hive提高30%
  - 百度线上query性能提升达4倍

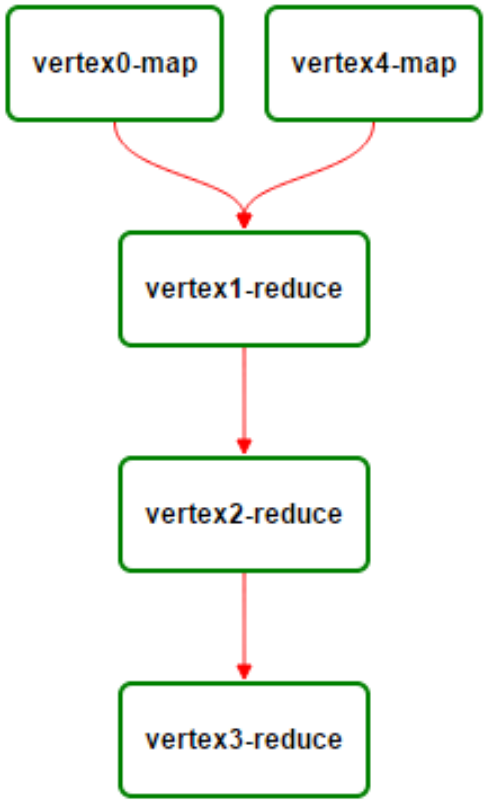
优化

- 减少数据流读取/传输
  - Dag优化
  - 条件表达式下推
    - 减少中间传输行数
  - Partition裁剪、列裁剪
    - 减少读取数据量
    - 减少中间传输记录大小
  - 数据源合并
    - 减少重复读取的数据
- 节省CPU计算
  - 使用llvm直接产生汇编指令，优化计算
  - 避免/优化记录反解效率

优化-多种运行模式

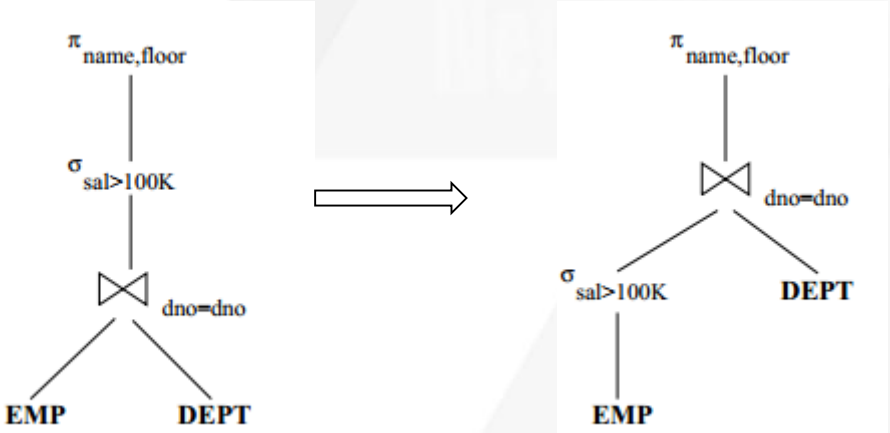
Local模式运行：任务在本地运行，调试方便（codex)

DAG模式运行
Join + Agg



优化-谓词下推/Partition裁剪

- 谓词下推



- Partition裁剪
  - 数据按照确定粒度组织
    - /tables/event_action=tieba_view/event_day=20140802/event_hour=12/…
    - /tables/event_action=tieba_view/event_day=20140802/event_hour=13/…
  - 依赖于条件表达式下推
    - 只引用到partition列的表达式即可被用于裁剪
  - Partition裁剪极大减少输入数据量
    - 不可用 -> 可用

优化-减少IO数据量

- 列裁剪
  - 分析每个算子用到的列，没有用到的列不需要读取

```
select '20150101', count(*)
from pb_access
where event_day  = '20150101';
```

  - 作用：减少读取RCFile/ORCFile的IO数据量，减少中间传输数据量

- 中间数据序列化格式优化
  - Null bit和数据编码进一个byte

null bit	实际数据	has more

  - 简单聚集case，shuffle数据量减半

优化-数据源合并

- 应用场景
  - 不同sql语句重复操作同一数据集
  - 同一sql语句中通过Union操作多次

```
--1. os + 大版本 +版本    (os->1~9,    app版本\w开头)
insert overwrite directory '{TMP_PATH}/searcherrno/e1'
select count(1) from audio_central_ts_server
WHERE event_day={DATE}  and pid != 832 and pid != 833 and pid!=1025 and pid != 1027 app'

insert overwrite directory '{TMP_PATH}/searcherrno/e99'
select count(1) from audio_central_ts_server
WHERE event_day={DATE} and pid !=0 and logid!= 0 and pid != 832 and pid != 833 and p

insert overwrite directory '{TMP_PATH}/searcherrno/e100'
select count(1) from audio_central_ts_server
WHERE event_day={DATE}  and pid != 832 and pid != 833 and pid!=1025 and pid != 1027

    udw.udw_event
where
    event_action='tieba_view' and event_day='{DATE}' and event_client_type='mobile_app'
    and event_isinternalip=0 and event_isspider=0
    and event_os_not_from_ua rlike '^[1-9]$'
    and event_os_version_not_from_ua rlike '^\\w+'
group by
```

执行流程-引入LLVM优化

- 表达式求值
  - 使用llvm直接生成指令

```
define double @BinaryExpression(i8* %row_batch, i32 row_index) {
eval_left:
```

```
virtual Datum BinaryExpression::Eval(Tuple* tuple) {
    typedef Datum (* FUNCTION) (const Datum& left, const Datum& right);
    FUNCTION f = NULL;
    Switch (m_op) {
    case '+': f = Add; break;
    case '-': f = Sub; break;
    case '*': f = Mul; break;
    case '/': f = Div; break;
    }

    const Datum& left = m_left->Eval(tuple);
    const Datum& right = m_right->Eval(tuple);
    if (left.is_null() || right.is_null()) {
        return NullDatum;
    }
    return f(left, right);
}
```

```
ret_null_block:                                   ; preds = %eval_left, %eval_right, %check_operand
    ret {i1,i64} {false, 0}
}
```

优化JNI调用

- 为什么要用JNI
  - 使用java编写的InputFormat、OutputFormat、以及Hive udf
- Batch方式一次处理多条记录，降低JNI调用开销

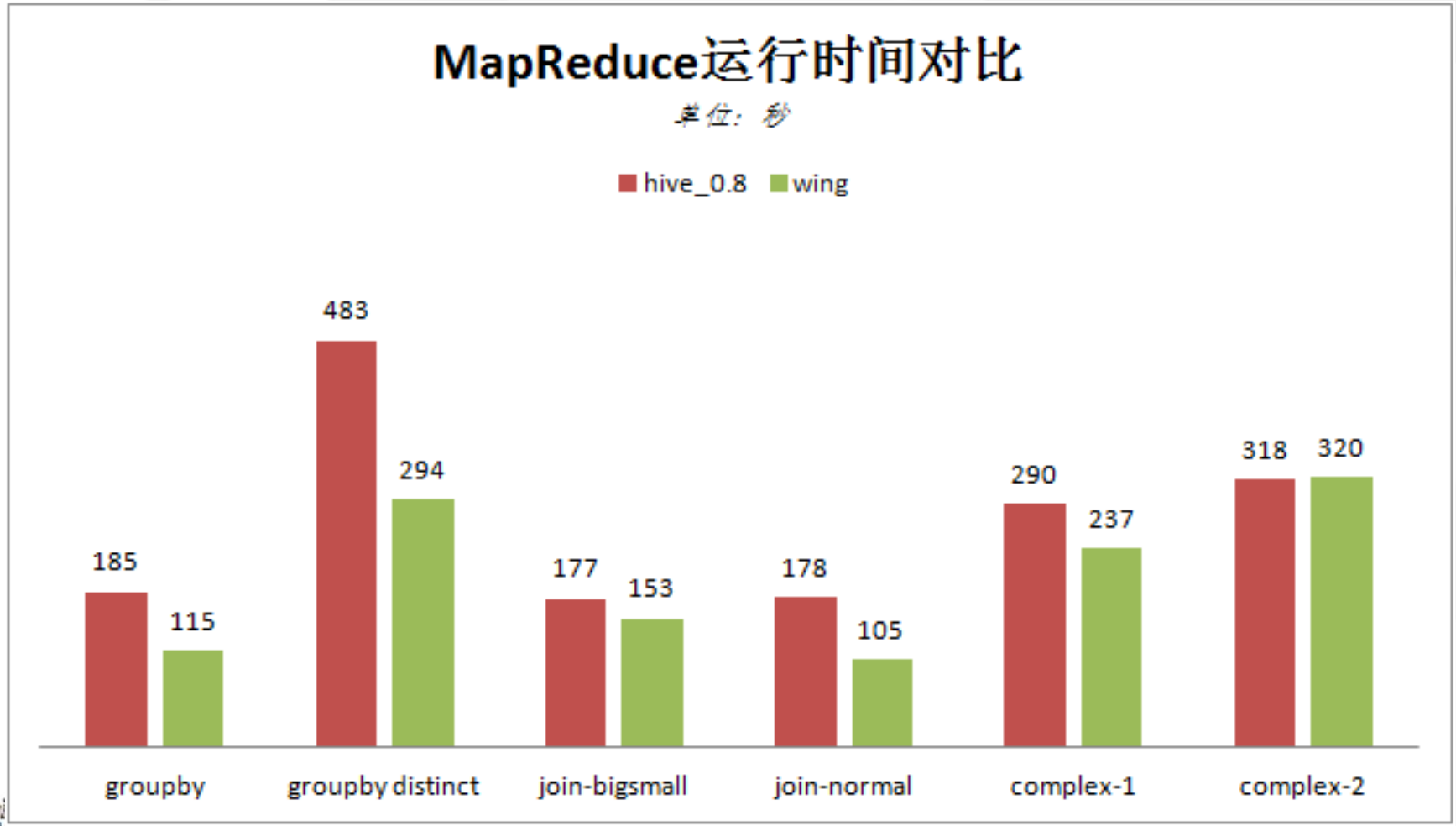
```
virtual bool write_row_batch(const runtime::RowBatch* row_batch);
```

  - 1次C++到Java的JNI调用大约0.2us
  - 解决：运行时算子尽量缓存输出记录，批量发送给下游算子
  - 针对输出数据量大的case，优化后reduce端执行性能是原来的**2倍**以上
- 避免java的String和byte[]的转换
  - 一次转换大约0.2 – 0.3us（短字符串）
  - 解决：JNI接口避免传递String；修复hive代码中的性能问题

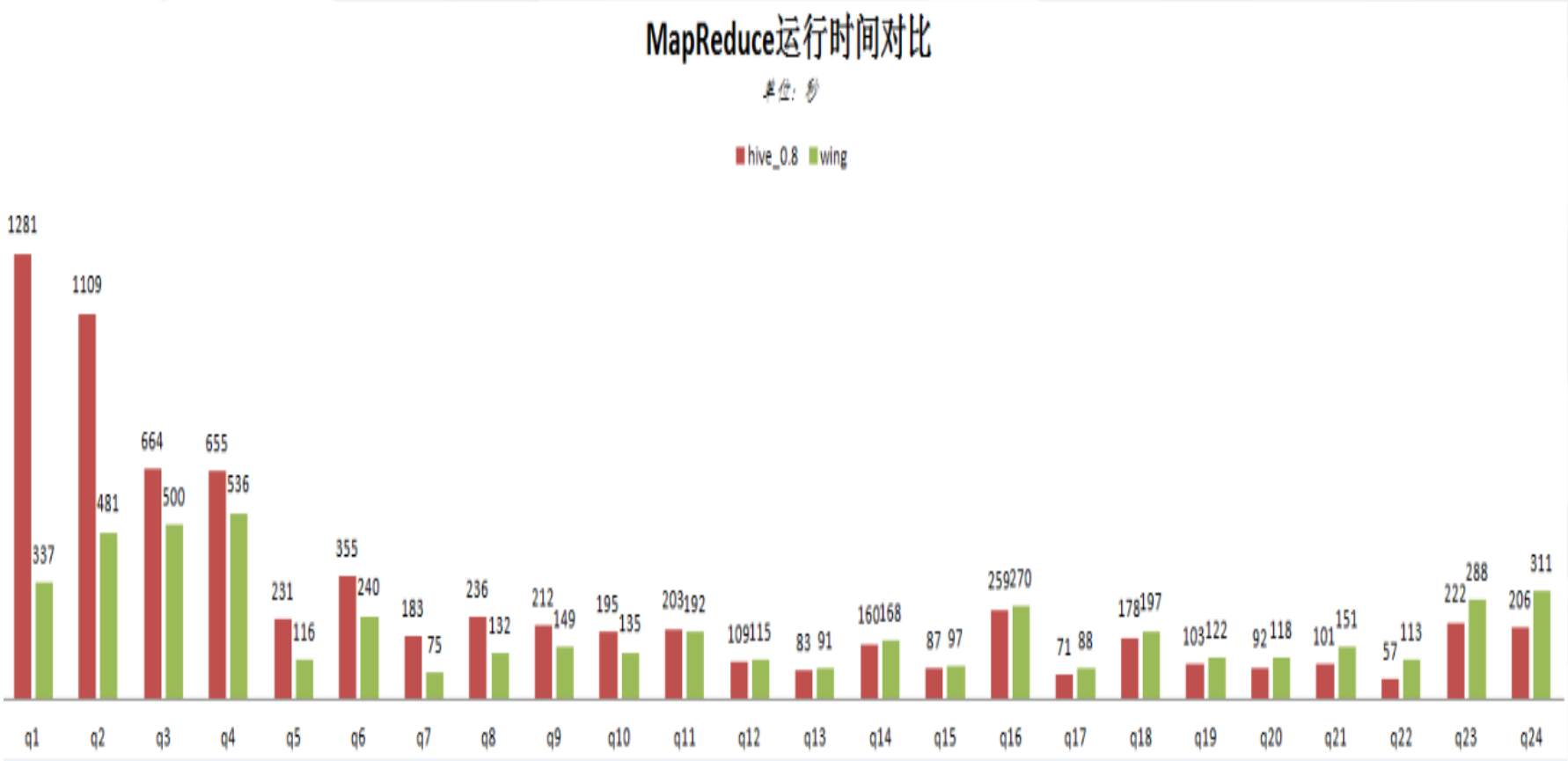
概览

- QueryEngine技术介绍
  - 技术发展、架构介绍
- Wing的语义接口
  - 语法、语义系统
- 执行优化
  - 降低数据传输、提升计算效率
- 性能结果
  - runtime性能要比hive提高30%
  - 百度线上query性能提升达4倍

6种典型场景Query



实际在线业务Query



开发流程

- 主干开发、持续集成、分支发布
  - 全部feature在主干上开发，分支只做bug fix
  - 单元测试保证代码基本质量
  - 开发 -> 持续集成系统测试 -> 代码review -> 入库

- 集成测试：数据diff
  - 选取线上实际查询作为diff case
  - 将部分线上数据导入线下集群，节约测试时间
  - 版本发布之前进行diff测试

- 开发量
  - 代码共计17w行
  - 测试代码：6万行，单测用例：200+个
  - 测试行覆盖率：平均85%，核心代码约11w行85%
  - 核心开发：3 ~ 4人 1.5年

踩过的许多坑：Tears

- 代码完全用C++实现，不适应Hadoop生态
  - 开发效率、运行效率都受影响
- Join算子谓词下推、隐含条件下推
  - A inner join B on A.a = B.a and A.a > 1
  - and B.a > 1
- JNI调用代价最小化
  - Java -> C++, 20ns, C++->Java 200ns
  - 通过Jni读取OrcFile
  - Batch方式处理数据
- llvm引入的坑
  - 集群Cpu型号不支持llvm的SSE指令
  - 引入llvm后导致C++异常机制失效
  - Signal Handler线程不安全

总结：新一代百度大数据查询引擎

- 更丰富的功能和接口
  - Session db、多后端、CQuery
- 更好的扩展性
  - 适配local、MR、Spark等多种运行模式
- 更高的执行效率
  - llvm优化、结合百度场景优化
- 更稳定的系统
  - 高覆盖率的单元测试
  - 完全独立开发的代码


ChinaUnix
THANKS