QCon

www.qconferences.com

Wing - 新一代百度大数据查询引擎

刘成 百度大数据部QE团队

2015-04-21

促进软件开发领域知识与创新的传播

InfoQueue

ArchSummit
全球架构师峰会

[深圳] 2015年7月17日-18日

QCon
全球软件开发大会

[上海] 2015年10月15-17日

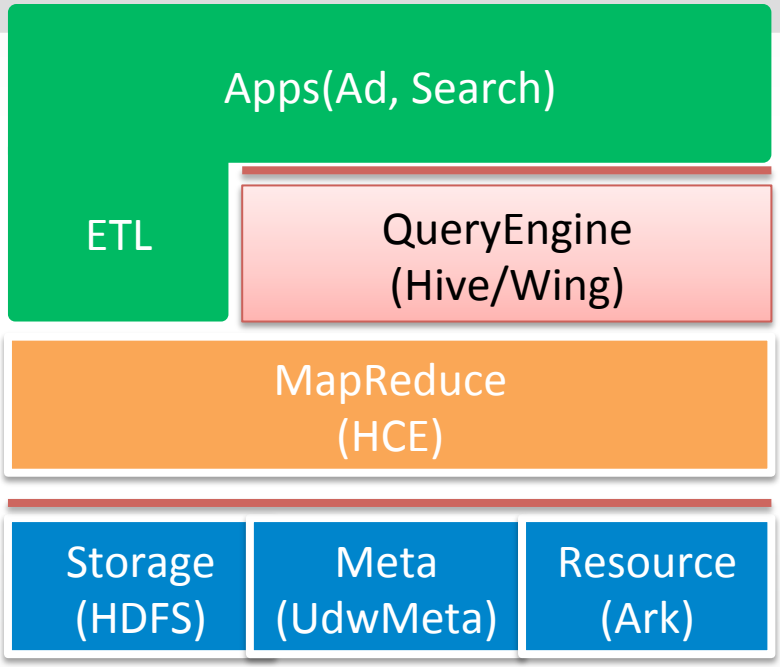
关注InfoQ官方微信
及时获取QCon演讲视频信息

概览

- QueryEngine技术
  - 百度QueryEngine 1.0 -> 2.0
- Wing前端接口
  - HQL + CQuery
- 优化引擎
  - 数据传输
  - 计算效率
- 性能结果
  - runtime性能比hive提高30%
  - 百度线上query性能提升达4倍
- 总结
  - 开发模式、Tears

QueryEngine技术

- 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



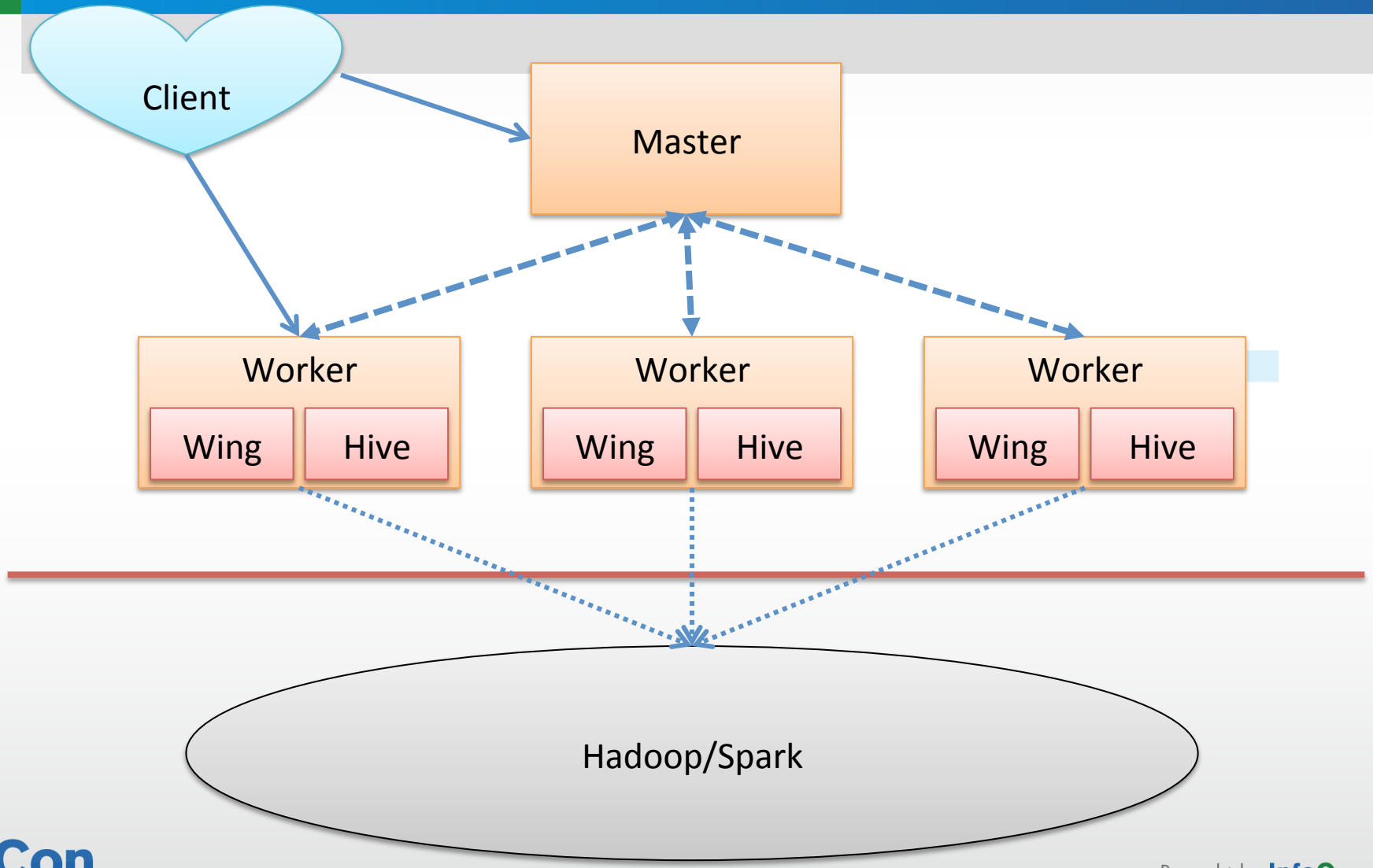
百度QueryEngine服务

- 百度QueryEngine服务
  - 每天session数量：14~15w
  - 每天处理数据量：2P
  - 运行场景：主要是大型HQL统计任务

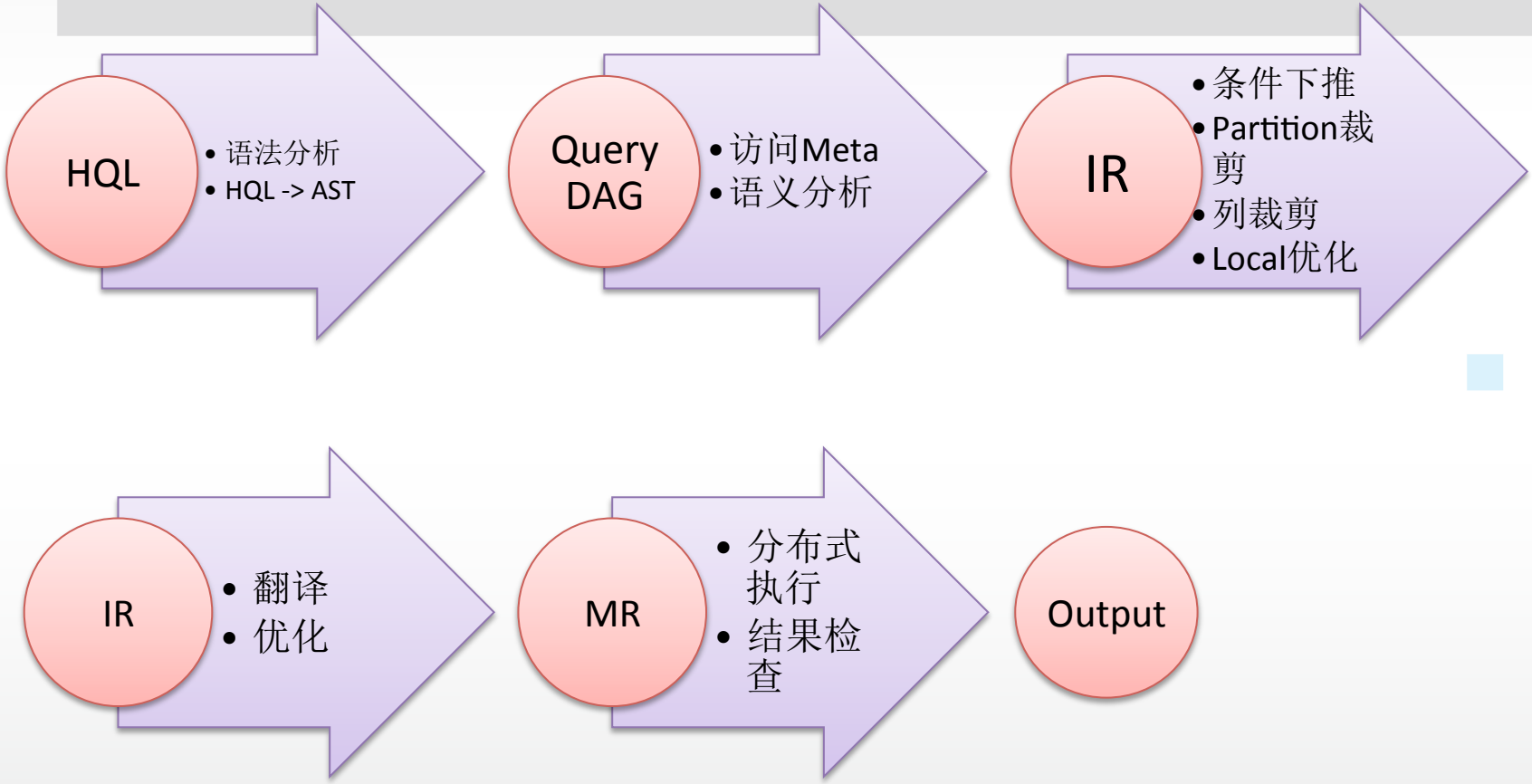
- QueryEngine 1.0：Hive
  - Hive 0.8，与最新版本Hive 0.14脱节
  - 百度需求多样化，不易于社区版保持同步
  - 代码耦合度高，不易模块化移植
  - 缺乏单测，新手接受困难，bugfix困难

- 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



Query执行过程

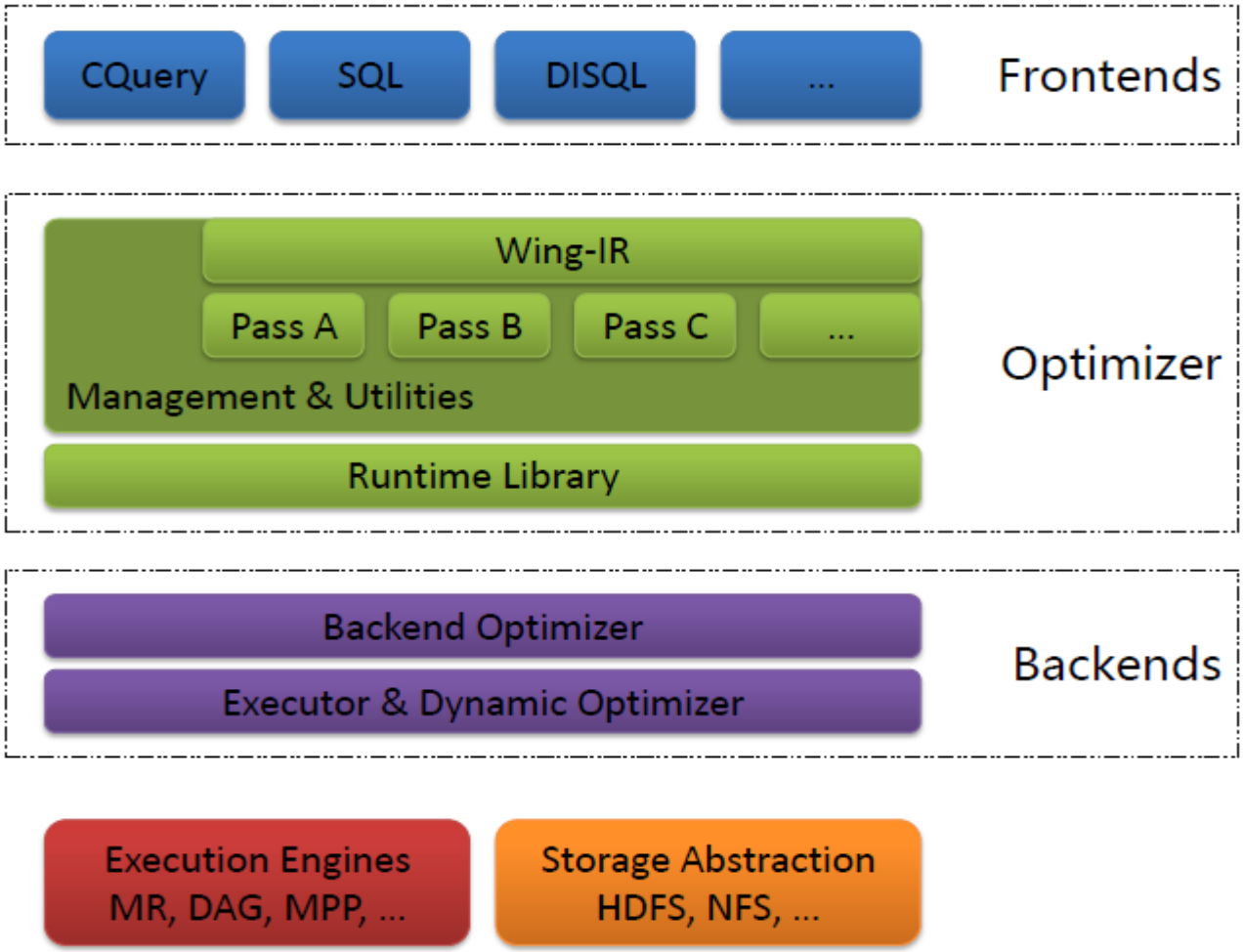

HQL
• 语法分析
• HQL -> AST
Query DAG
• 访问Meta
• 语义分析
IR
• 条件下推
• Partition裁剪
• 列裁剪
• Local优化
IR
• 翻译
• 优化
MR
• 分布式执行
• 结果检查
Output

Query执行过程

- HQL:
  - select a from t where a > 1;
- QueryDAG:
  - LOAD(t) -> FILTER(a > 1) -> PROJECT(a) -> OUTPUT(stdout)
- IR:
  - SOURCE(t { a, b , c}) -> Filter((a > 1), {a, b, c}) -> PROJECT(a {a}) -> OUTPUT(stdout, {a}, serde)
- IR:
  - SOURCE(t {a} (a > 1)) -> SINK(stdout, serde)
- MR:
  - Map only

Wing架构设计

- IR(中间表示)：类型、表达式、算子、执行子拓扑
- PASS(优化器)：列裁剪、条件下推、partition裁剪、Local优化等
- Runtime(执行逻辑)：表达式求值、投影、过滤、聚集、连接



前端接口：HQL

- 支持hive 0.8的所有HQL语法、功能
  - SQL功能: filter/aggregate/join/lateral view/dynamic partition
  - DDL操作: create/drop/show/alter/set
  - 自定义逻辑：UDF、UDAF、UDTF、Transform
- 语义扩展
  - 数据组织：**Namespace**, Database, Table, Partition, Bucket
  - 数据类型：基本类型、list、struct、map、**enum**、**递归结构**
    - Message TreeNode { optional TreeNode left = 1; optional TreeNode right = 2; }
- 功能扩展
  - 多后端Meta：同时操作多个不同数据源
  - 会话临时表(SessionDB)：只在本session内可见
  - Implicit Join

前端接口：HQL

- 多后端Meta
  - 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;

- 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;

前端接口：CQuery

- 数据抽象：Table
- 操作抽象：Load/Sink/Filter/SelectAggregate/Join/Transform等
- 自定义逻辑：直接使用C函数作为UDF
- 示例代码

```
// A udf that trims the heading space of a string.
const char* trim_heading_space(const char* str) {
    int pos = 0;
    while (str[pos] == ' ') {
        ++pos;
    }
    return str + pos;
}

int main(int argc, char **argv) {
    DECLARE_FUNCTION(trim_heading_space);
    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");
    Table teenager_with_props = teenager->join(props, "id == user_id");
    teenager_with_props->select("name, trim_heading_space(favorite_book) as book")
        ->filter("is_not_null(book)")
        ->group("book")
        ->aggregate("book, count(name) as number")
        ->sort("number desc, book asc")
        ->output_overwrite_file("teenager_book.data", "teenager_book.proto", "Book");
```

优化

- 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
- 减少数据流读取/传输
  - 条件下推、Partition裁剪、列裁剪
  - 数据源合并
  - Local优化、Dag优化
  - Shuffle优化
- 其他
  - 访问Meta、Split、JNI调用

优化-LLVM

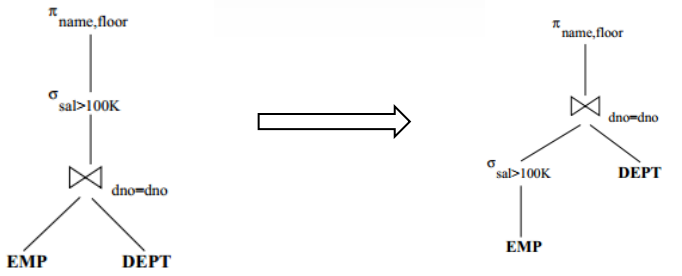
- LLVM(low level virtual machine): JIT
- 表达式求值
  - 使用llvm直接生成指令
  - Select a + b from t;
-

```
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 {i1,i64} %result

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

优化-谓词下推/Partition裁剪/列裁剪

- 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/…
  - Partition裁剪极大减少输入数据量
    - 不可用 -> 可用

- 列裁剪
  - 减少读取数据量，缩小传输记录
  - 嵌套结构

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'
uni
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
```

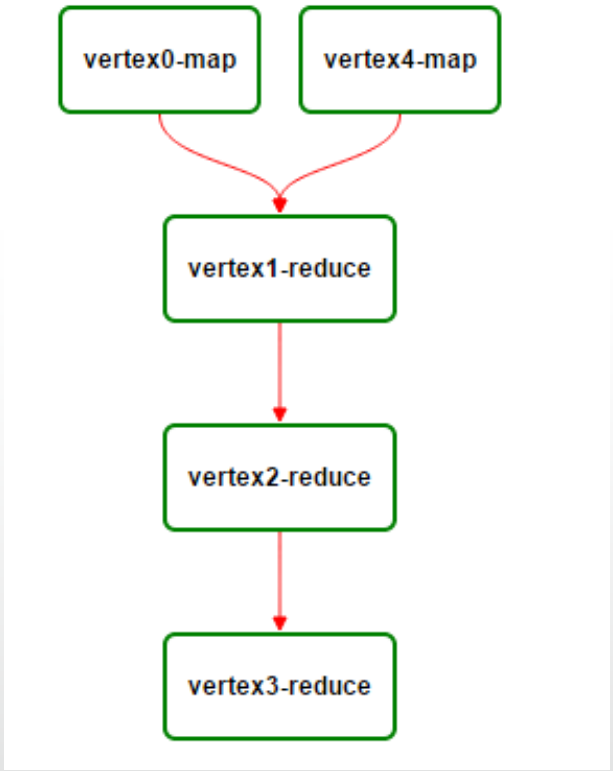
优化-多种运行模式

A、Local模式运行：

1. 小数据集上的简单查询（select a from t(10M) where a > 10);
2. 不带过滤条件的直接读取（select a from t(1T) limit 10;

B、DAG模式运行：

中间输出不写hdfs (e.g. Join + Agg)



优化-减少IO数据量

- 中间数据序列化格式优化
  - Varint编码数值类型
  - Null bit和数据编码进一个byte
  - 多字段聚集case，shuffle数据量减半
    - Select a, b, count(c), count(d) from t group by a, b;

优化:任务执行前

- 访问Meta
  - Wing: ppd -> GetPartitions() -> filter
  - Hive： ppd -> get_partition_names -> filter -> get_partition_by_name()
- 多线程Split
  - Magi： szwg， Hadoop： nmg
  - Wing： 64线程split
  - Hive： 1个线程

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

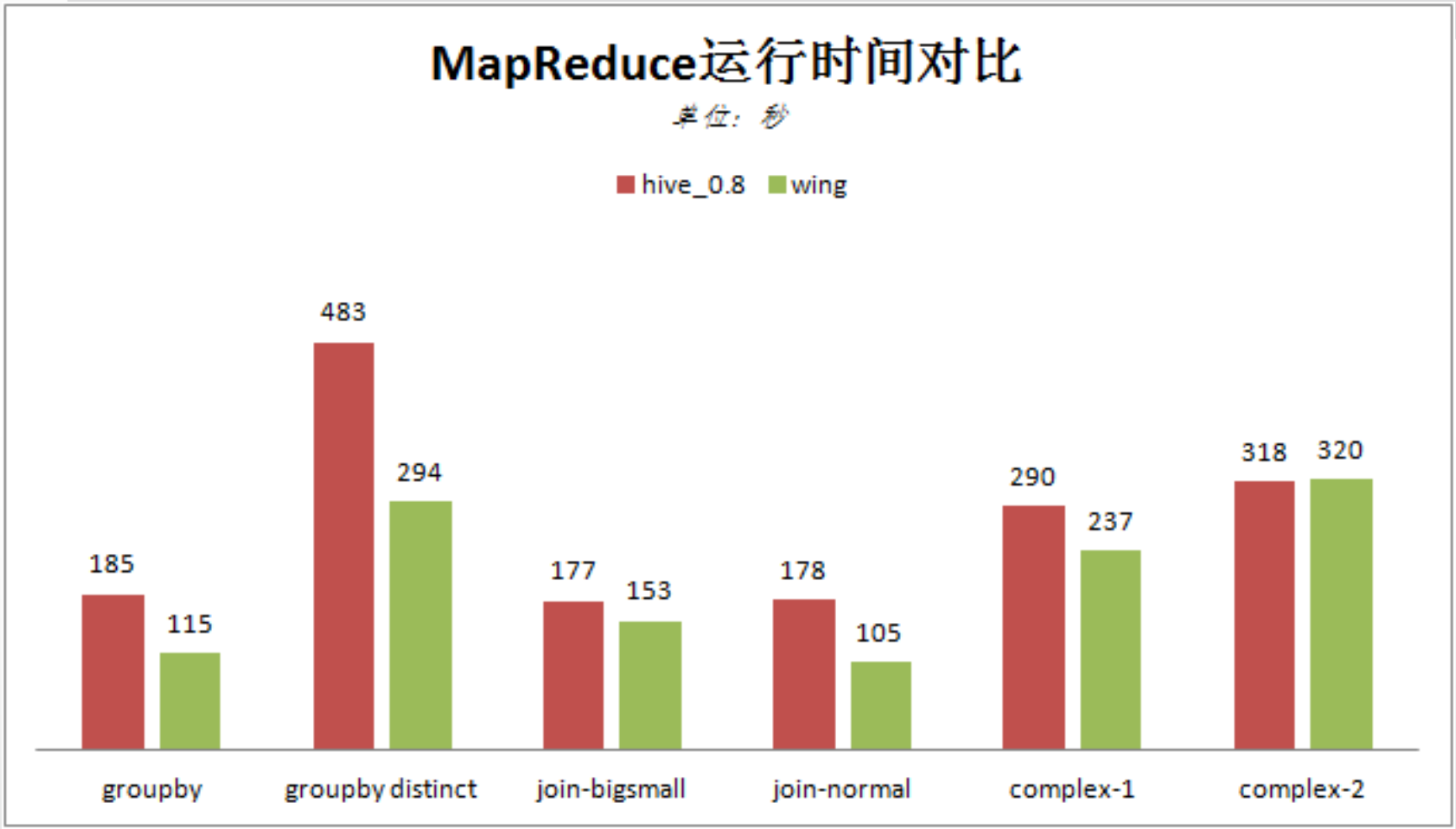
```
virtual bool write_row_batch(const runtime::RowBatch* row_batch);
```

  - 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

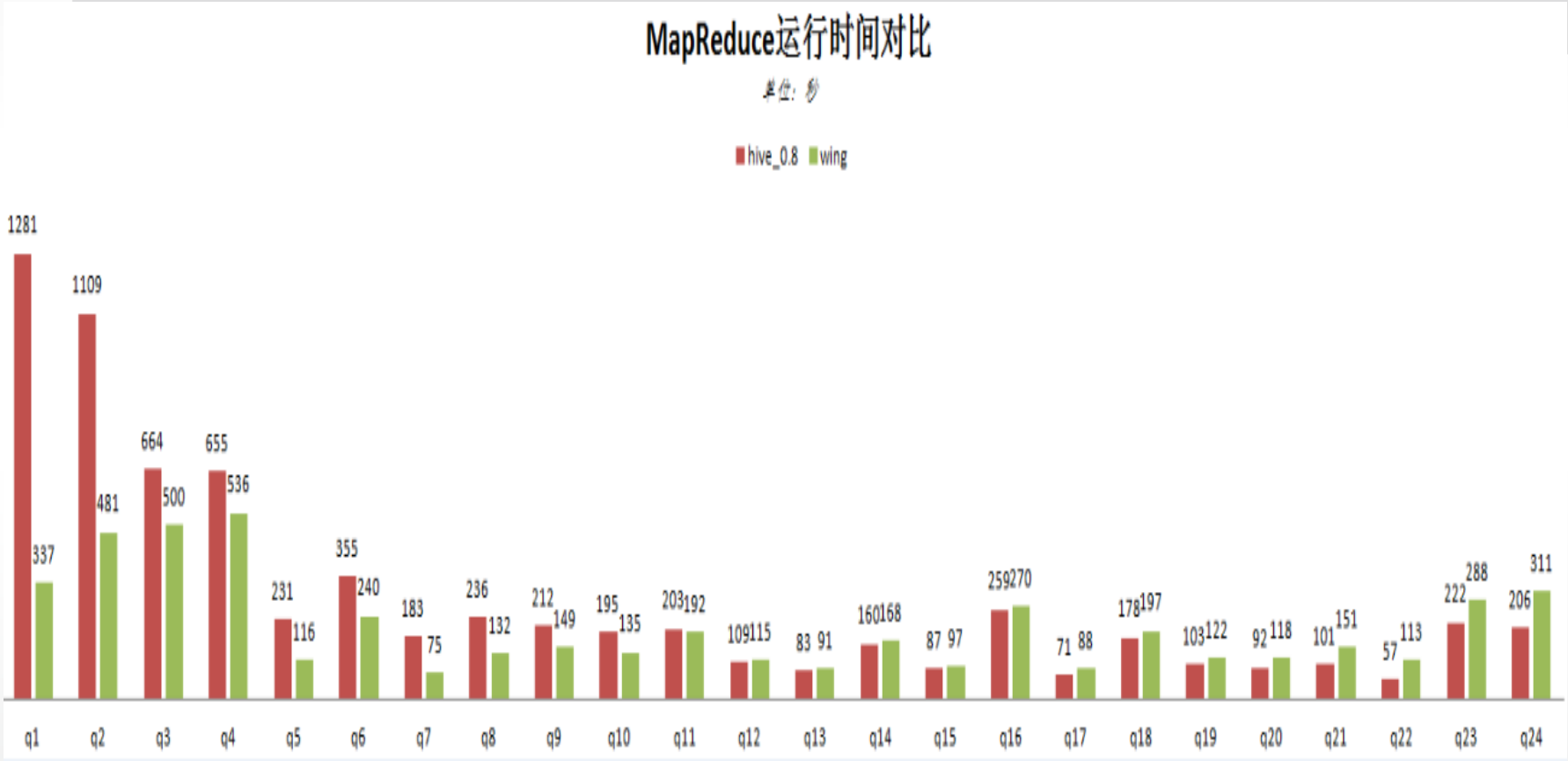
QueryEngine暂时无法优化的

- 1.transform/udf
  - 大量低效计算逻辑在脚本或自定义逻辑里面
- 2.不能下推条件的Outer join
  - Select * from A left outer join B where f(B.event_day);
- 3.跨级群计算
  - 数据存在集群A，计算在集群B
- 4.cross join, order by等

性能结果：6种典型场景Query



性能结果：实际在线业务Query



开发方式

- 开发量
  - 代码共计17w行
  - 测试代码：6万行，单测用例：2000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
- 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

Tears

- 代码完全用C++实现，不适应Hadoop生态
  - Jni，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
- llvm引入的坑
  - 集群Cpu型号不支持llvm的SSE指令
  - Signal Handler中使用线程不安全函数sigprocmask

总结：Wing——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
  - 容易移植和适配local、MR、Spark等多种运行模式
- 更高的执行效率
  - llvm优化、数据源合并
- 更稳定的系统
  - 高覆盖率的单元测试
  - 完全独立开发的代码

QCon

Thanks && QA !

www.infoq.com/cn

InfoQ

@InfoQ

infoqchina

软件
正在改变世界！