DISQL 2.0：
百度海量数据分析语言

陈晓鸣
资深工程师
百度基础架构部
@陈晓鸣在百度
chenxiaoming@baidu.com

Baidu 百度

Baidu 百度

目录

组织进化-使用人数的问题

LSP

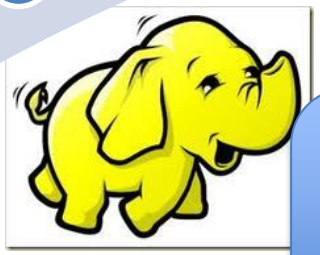
- 成立 了数据团队
- 正在进行中...

- 一站式的日志分析服务
- B-S模式访问
- 私有云
- 统一管理、统一运维

语言进化-数据量的问题

DISQL

DISQL 2.0



- 类SQL的描述式语言
- PHP + C 混合运行时
- 算子的分布式实现
- 用户自定义函数支持（PHP、.so）

- 全新的C++ 运行时
- C++自定义函数支持
- 更多算子支持
- 多种优化

cron awk php

目录

例：新闻站点访问量和广告量统计

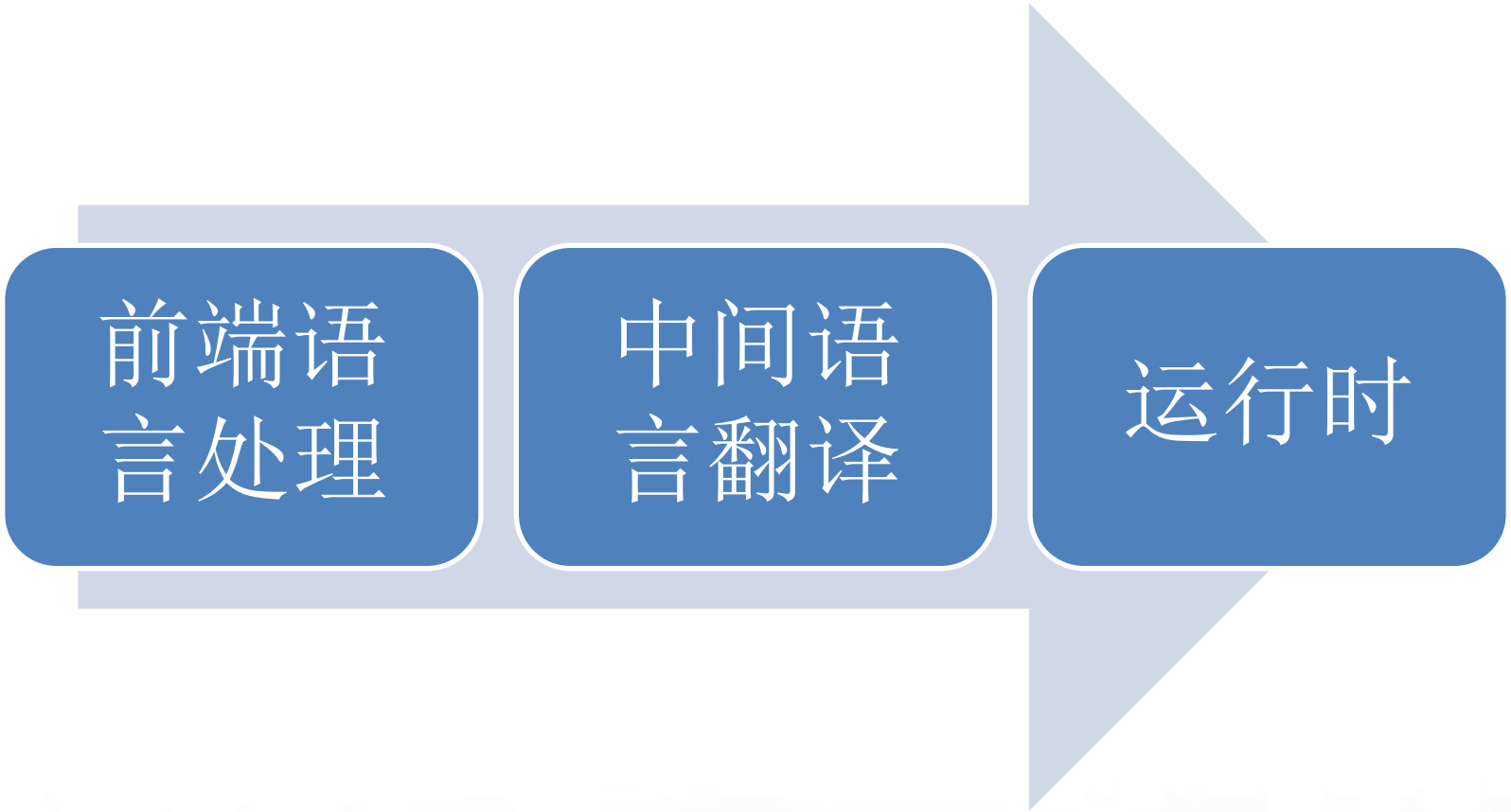
- 执行步骤
  - 读取日志数据
  - 选取出_Url、_Res(广告数)两列
  - 编写一个函数，从_Url中抽取出_Site
  - 用正则表达式过滤出新闻站点的数据
  - 按站点分组，每组做两件事：
    - 计算访问量
    - 将广告数求和
  - 输出数据，每行是一个JSON数据

例：DQuery代码

```
1  function get_site($rec){
2      $parts = explode( '/', $rec['_Url'] );
3      $rec['_Site'] = $parts[0];
4      return $rec;
5  }
6
7  $temp_view = DQuery::input()
8  ->select( array( '_Url','_Res' ) )
9  ->select( 'get_site' )
10 ->filter( array( '_Site', 'match', '/news\.[^.]+\.cn/' ) );
11
12 $temp_view
13 ->group( array('_Site') )
14 ->each(
15     DQuery::count( '*', '_QueryCnt' ),
16     DQuery::sum( '_Res', '_ResSum' )
17 )->outputAsFile( 'query_ad_site
   ','StorerUtils::jsonLine' );
```

- 读取日志数据
- 选取出_Url、_Res(广告数)两列
- 编写一个函数，从_Url中抽取出_Site
- 用正则表达式过滤出新闻站点的数据
- 按站点分组，每组做两件事：
  - 计算访问量
  - 将广告数求和
- 输出数据，每行是一个JSON数据

处理阶段



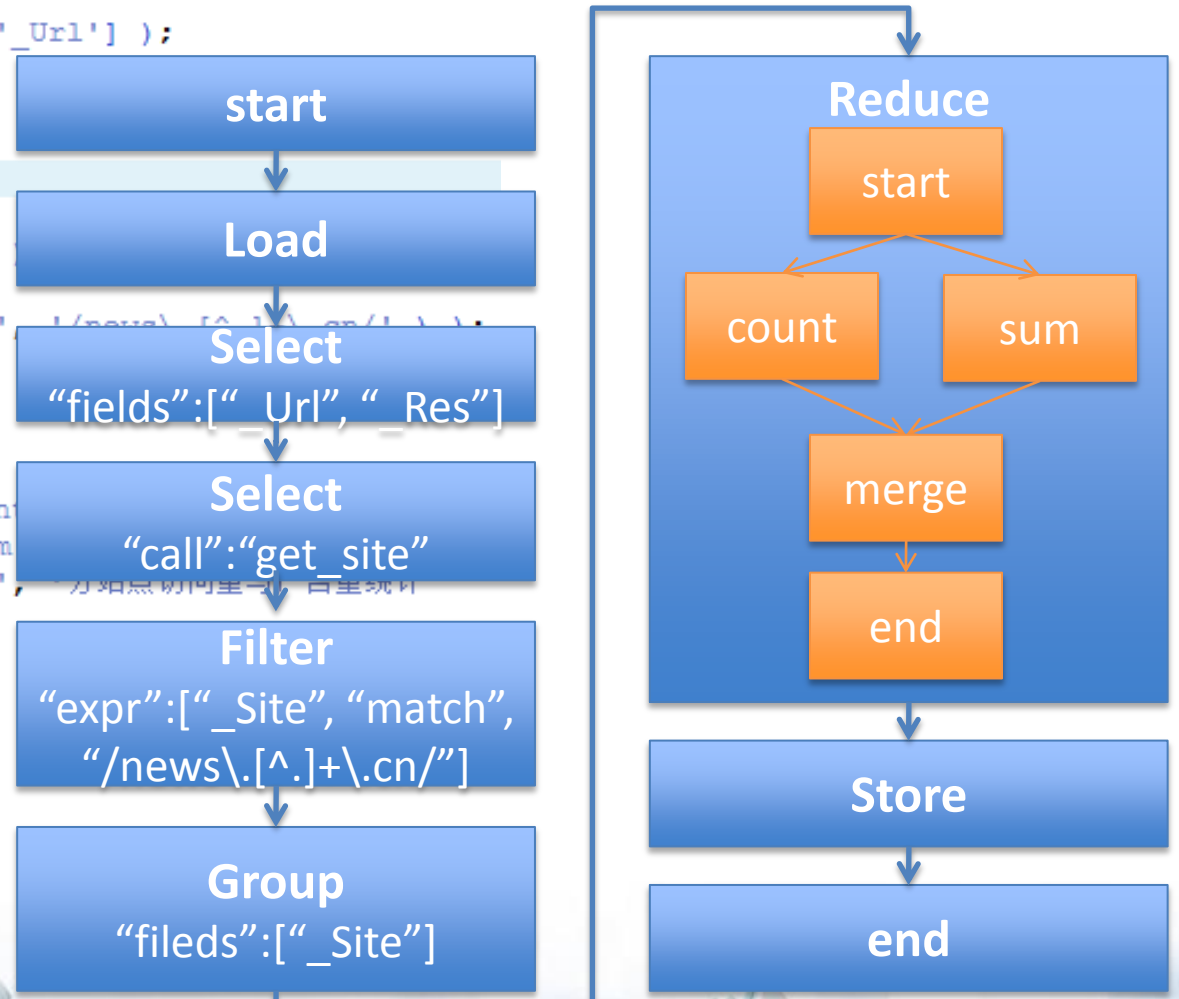
目录

前端处理

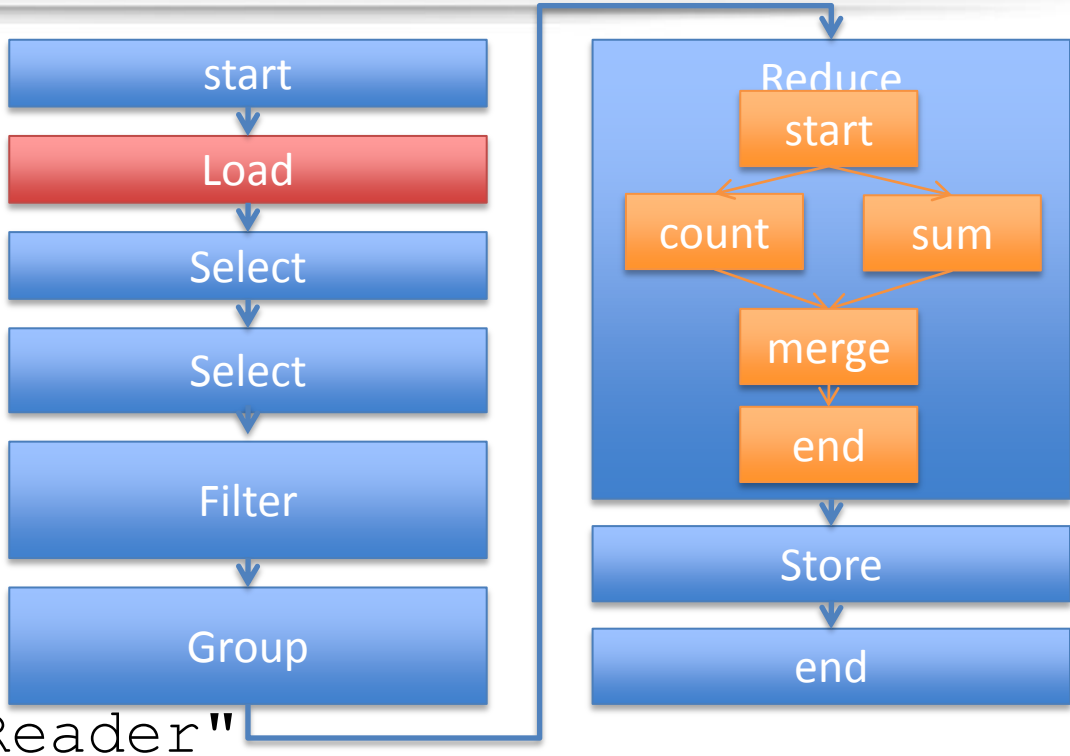
- 把用户编写的计算逻辑翻译为便于编译程序理解的中间码（语法树、数据流图）
- 前端代码运行一遍，产生结果是中间码
- 相当于编译技术中的parser
- 中间码用JSON表示

数据流图

```
function get_site($rec){
    $parts = explode( '/', $rec['_Url'] );
    $rec['_Site'] = $parts[0];
    return $rec;
}

$temp_view = DQuery::input()
->select( array( '_Url','_Res' ) )
->select( 'get_site' )
->filter( array( '_Site', 'match', '/news\.[^.]+\.cn/' ) );

$temp_view
->group( array('_Site') )
->each(
    DQuery::count( '*', '_QueryCnt' ),
    DQuery::sum( '_Res', '_ResSum' )
)->outputAsFile( 'query_ad_shows', '\t', 'StorerUtils::jsonLine' );
```



中间码




```
[
   {
      "cmd": "load",
      "path": null
      "using": "SchemaReader"
      "from": 0
      "options": {"max_item_in_mem": 100000}
      "include": [25]
   }, {"cmd":"filter"…}, {"cmd":"group"…},……
]
```

SQL语言接口？

```
query: SELECT fiel_list FROM form_list WHERE
clause
fiel_list: fiel_list ',' fiel_name | fiel_name |
'*';
fiel_name: ID|FID;
form_list: form_list ',' form_name|form_name;
form_name: ID;
clause: clause AND clause | clause OR clause |
clause_express| '('clause')';
clause_express: element compopera element;
element: element mathopera element | element1 |
'('element')';
compopera: '<' | '>' | '=' | '>=' | '<
=';
mathopera: '+' | '-' | '*' | '/';
element1: ID|ICON|FID|'"'STRI '"';
```

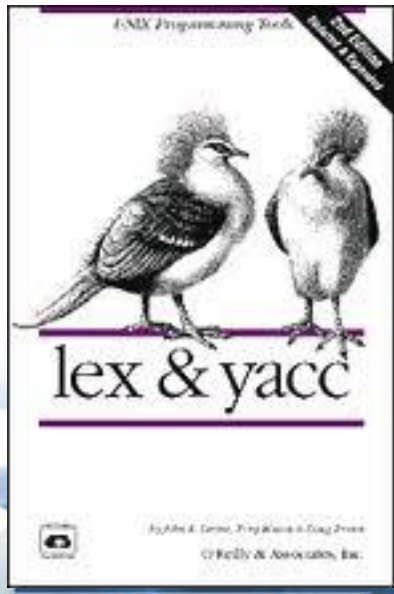
语言定义（词法、语法分析）

动作（生成中间码）

```
%%
SELECT |yylval.str=yytext;return(SELECT);|
";" |yylval.str = yytext;return(ENDMARK);|
/*;为结束符*/
.....
|alpha||alnum| * |yylval.str = yytext; return
(ID);| /*变量名或字段各*/
|digit| + |yylval.str = yytext; return (I-
CON);| /*常数*/
|alpha||alnum|[.]|alpha||alnum| * |yyl-
val.str=yytext;return(FID);|
/*带表或视图名的字段名*/
|alnum| + |yylval.str = yytext;return(STRI);|
/*字符串*/
[/t/n]; /*去除空格、制表符和换行*/
%%
```

```
[
  {
    "cmd": "load",
    "path": null
    "using": "SchemaReader"
    "from": 0
    "options":
    {"max_item_in_mem":
    100000}
    "include": [25]
  }
  , {"cmd":"filter"……},
  {"cmd":"join"……},…… ……
]
```

中间码



Baidu 百度

目录

中间语言翻译

正规化
- 将数据流图变成完整的方便后续处理的数据流图

算子替换
- 将实现复杂的算子等价替换成多个简单算子

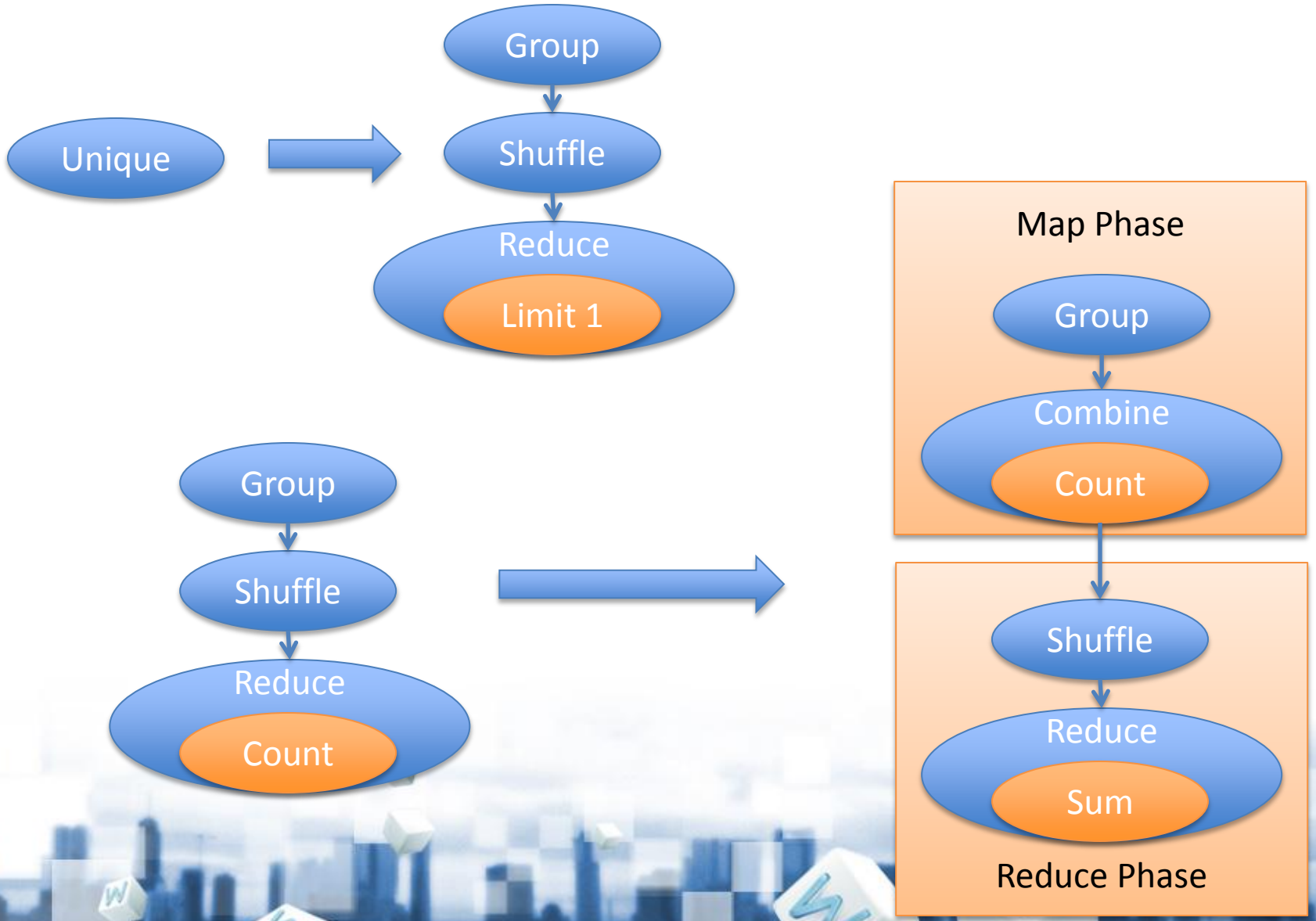
优化
- 对数据流图进行各种优化，使执行效率提高

阶段划分(可选)
- 划分为多个MapReduce执行阶段
- 不划分会生成单机程序

Schema推导、字段偏移量推导
- 推导每一算子产出的表schema，以及字段偏移量

代码生成
- 生成真正可执行的代码或数据流图

算子替换

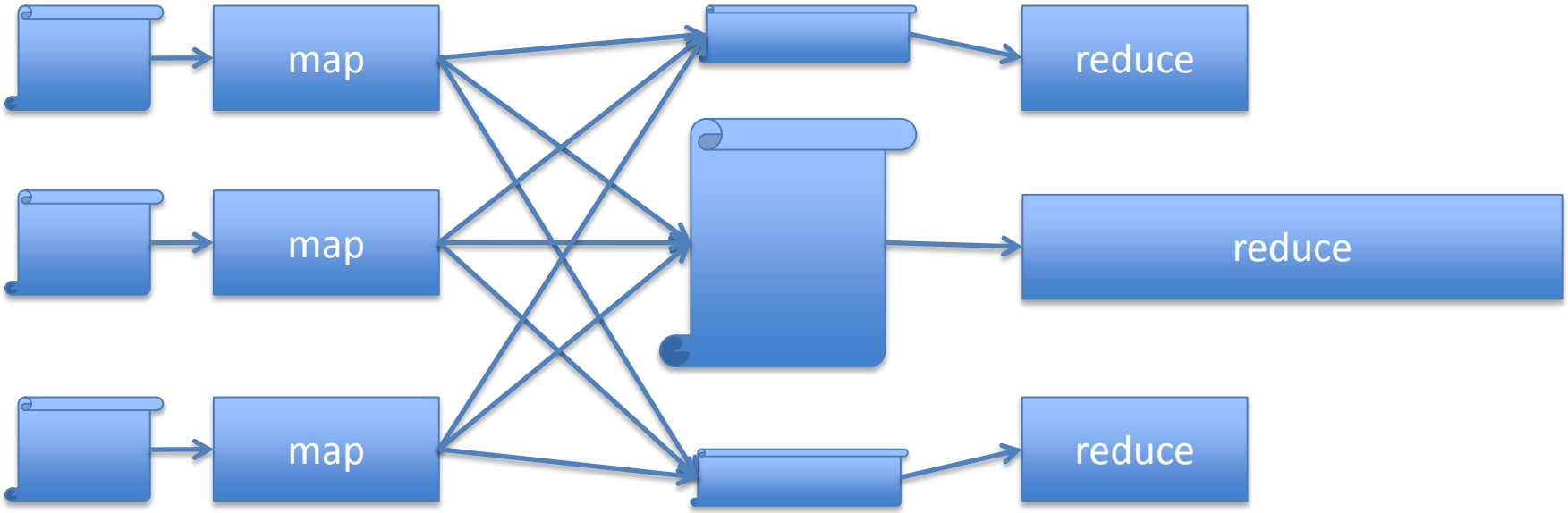

Unique
Group
Shuffle
Reduce
Limit 1
Group
Shuffle
Reduce
Count
Map Phase
Group
Combine
Count
Shuffle
Reduce
Sum
Reduce Phase

优化

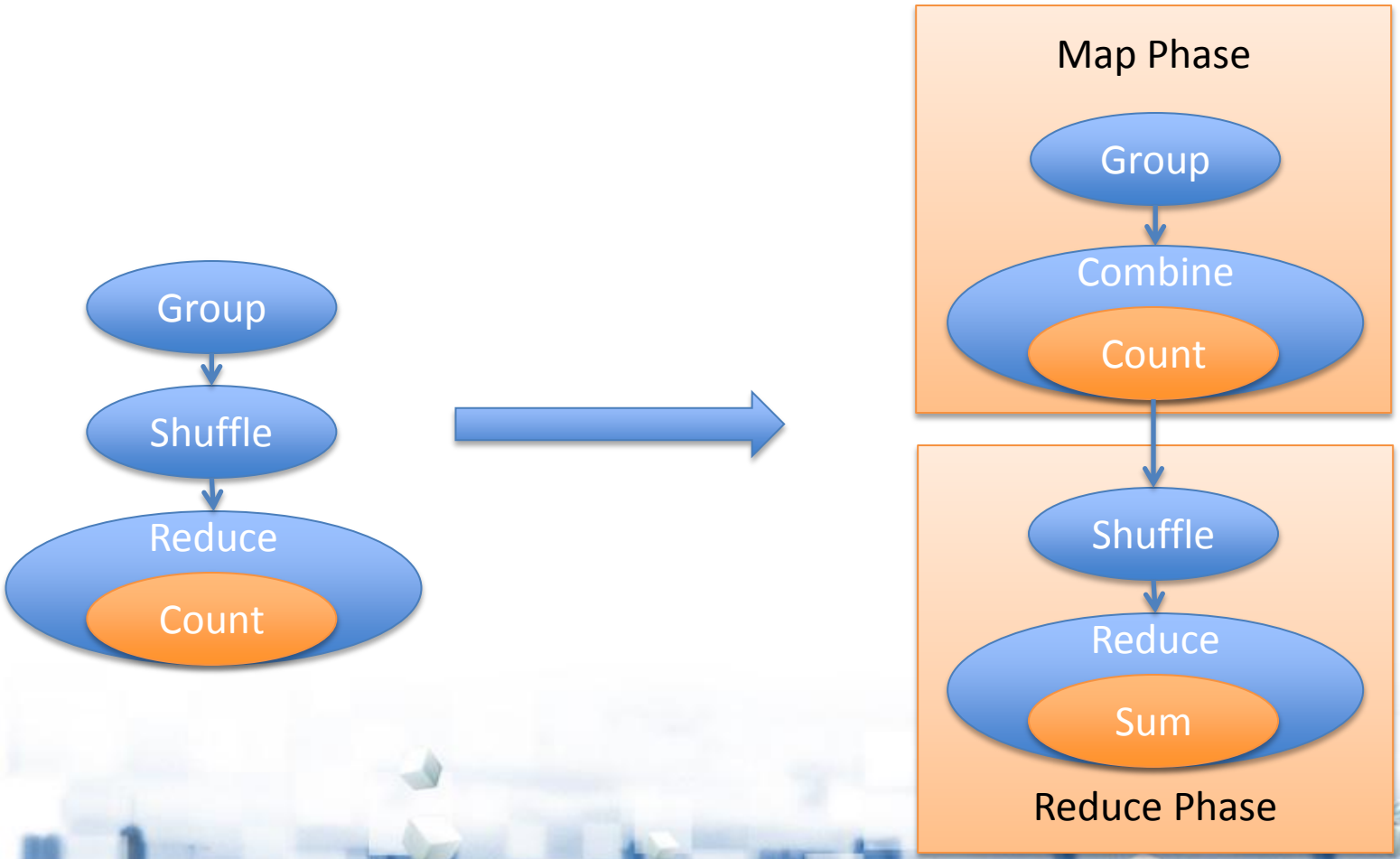
- 多种优化策略
  - 多任务合并
  - 等价算子合并
  - Combiner优化
  - Cached Combiner优化
  - 同key Join合并优化
  - 公共子表达式提取
  - ……
- 核心思想
  - 面向应用特点、减少作业轮数、减少I/O、减少重复计算

优化举例

大Group问题，shuffle I/O问题

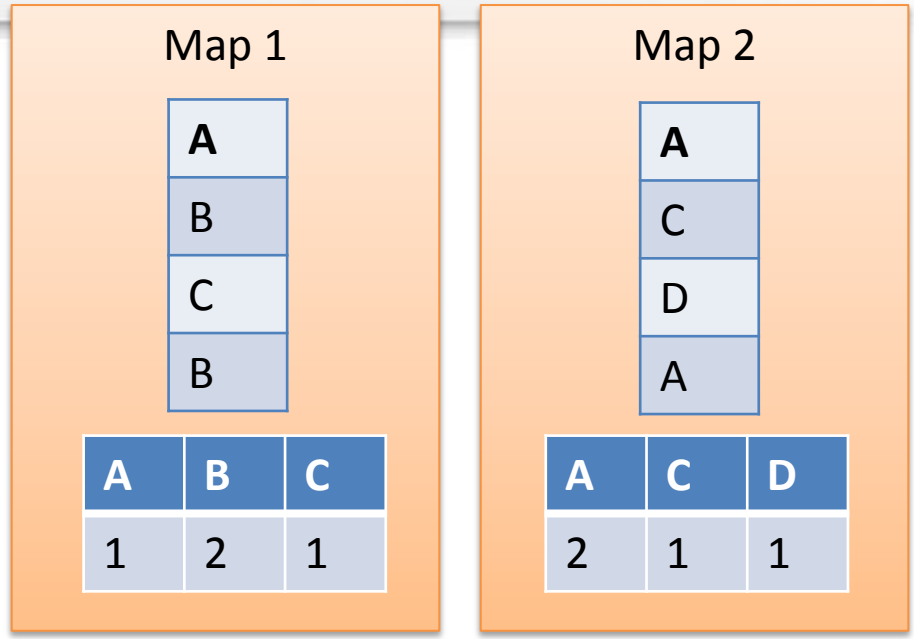


Combiner

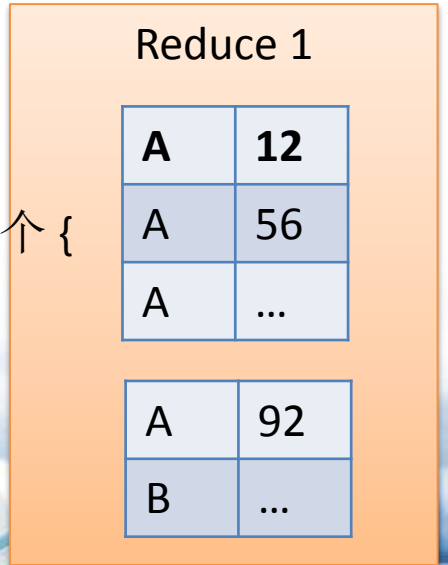

Group
Shuffle
Reduce
Count
Map Phase
Group
Combine
Count
Shuffle
Reduce
Sum
Reduce Phase

Combiner实现

- 不采用系统Combiner
  - 额外磁盘I / O
- 编译到Mapper中实现
  - 内存中的hash字典

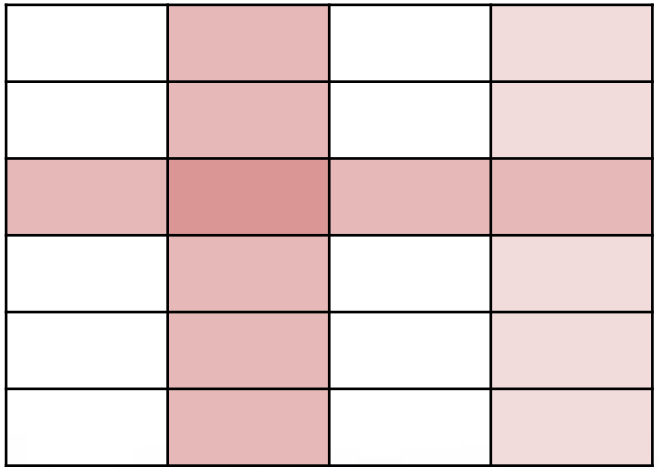


O(mapper)个 {

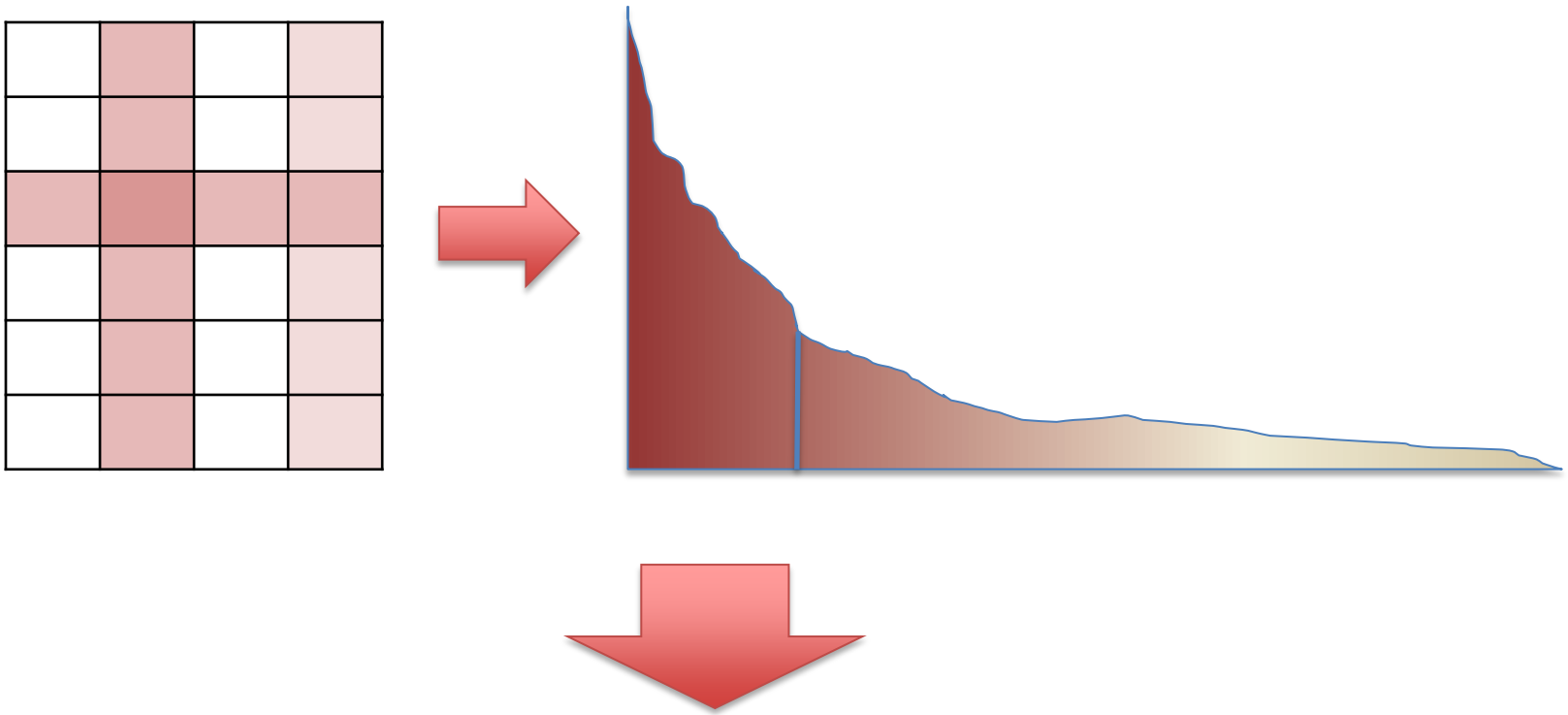


高维诅咒

- group(array('_Site','_Rank','_Strategy',.....))
- 高维诅咒
- 内存溢出

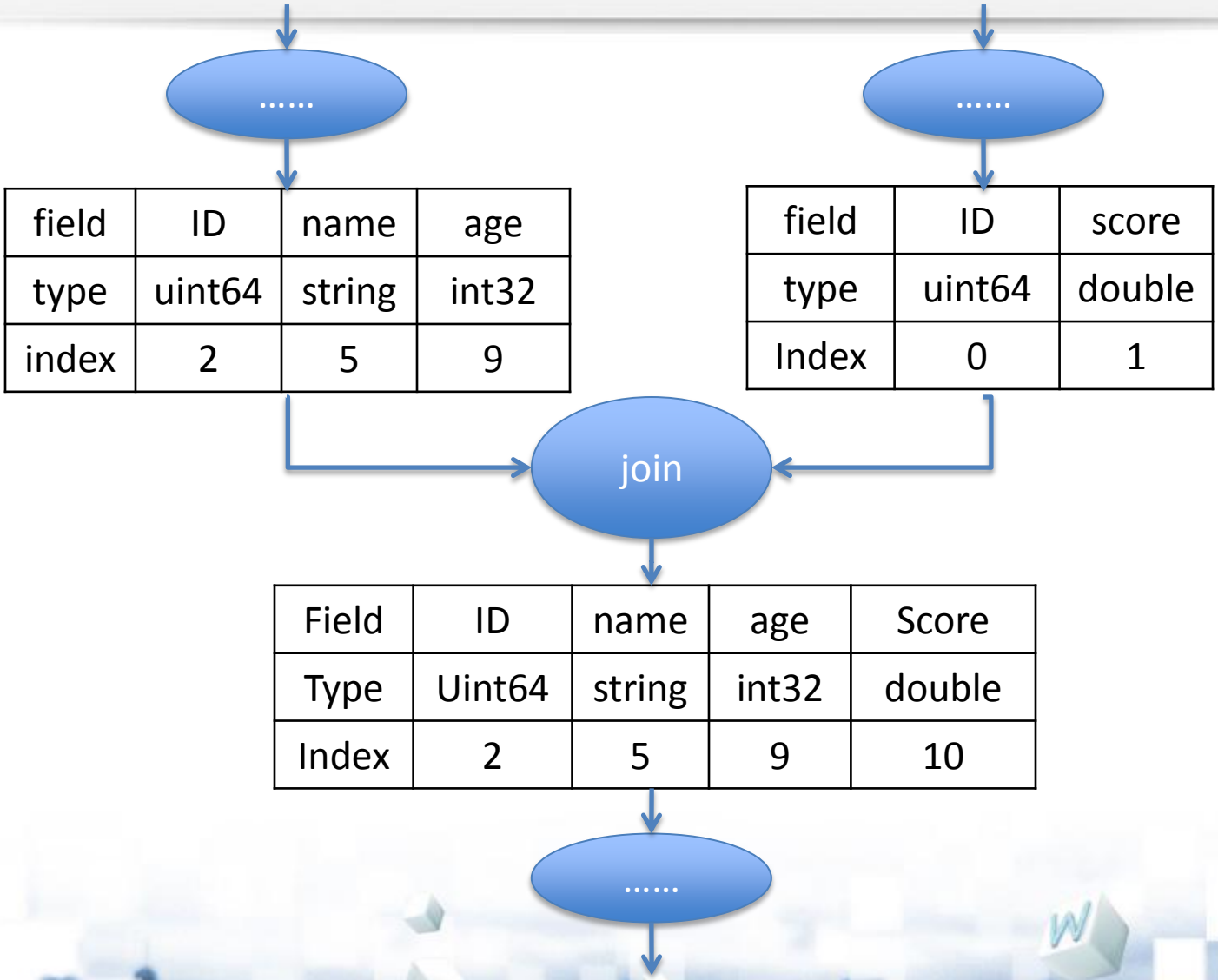


Cached Combiner

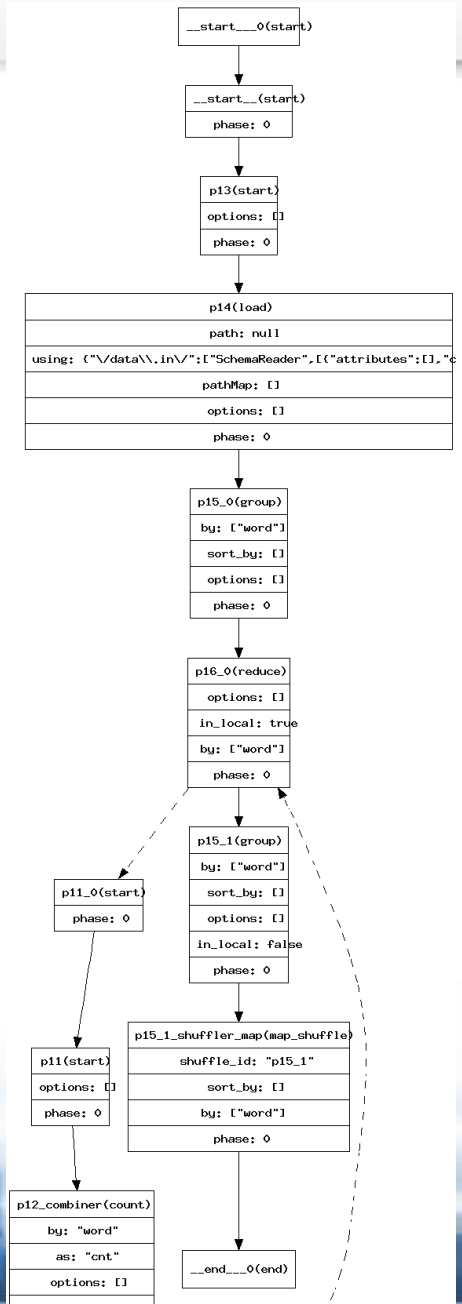


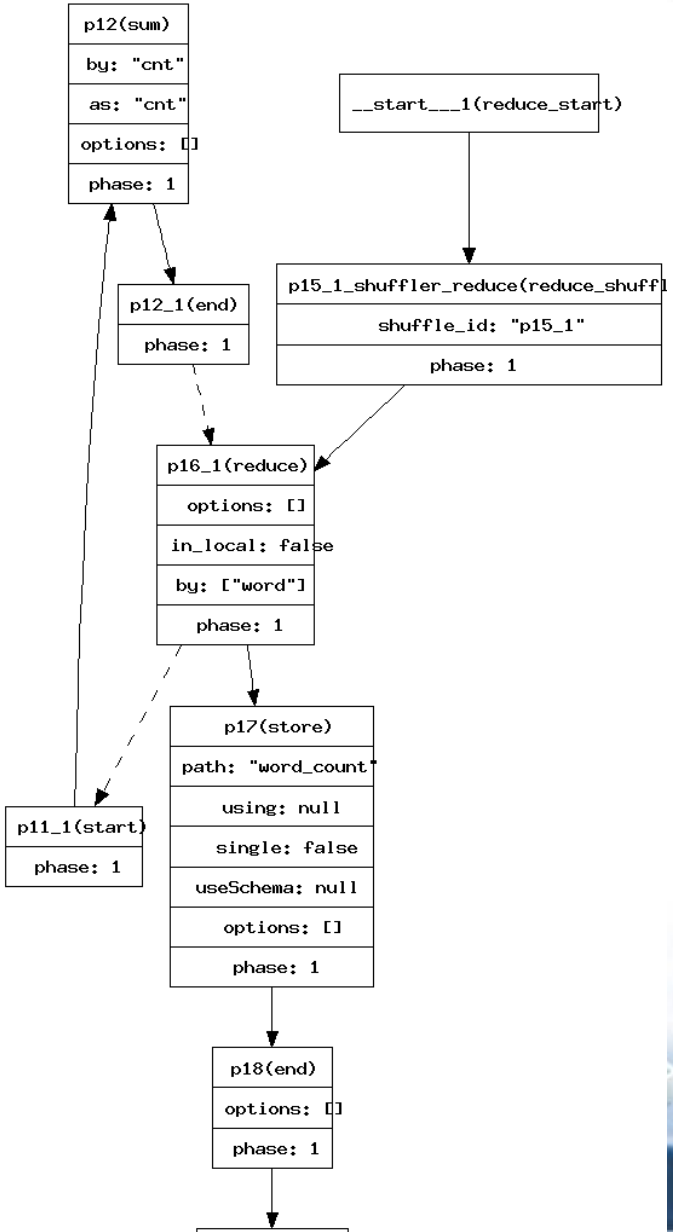
hot key	warm key	warm key	cool key	cool key	cool key

Schema推导、下标推导



数据流图生成


__start___0(start)
__start__(start)
phase: 0
p13(start)
options: []
phase: 0
p14(load)
path: null
using: {"\/data\\.in\/":["SchemaReader",[{"attributes":[],"c
pathMap: []
options: []
phase: 0
p15_0(group)
by: ["word"]
sort_by: []
options: []
phase: 0
p16_0(reduce)
options: []
in_local: true
by: ["word"]
phase: 0
p11_0(start)
phase: 0
p15_1(group)
by: ["word"]
sort_by: []
options: []
in_local: false
phase: 0
p11(start)
options: []
phase: 0
p15_1_shuffler_map(map_shuffle)
shuffle_id: "p15_1"
sort_by: []
by: ["word"]
phase: 0
p12_combiner(count)
by: "word"
as: "cnt"
options: []
__end___0(end)


p12(sum)
by: "cnt"
as: "cnt"
options: []
phase: 1
__start___1(reduce_start)
p12_1(end)
phase: 1
p15_1_shuffler_reduce(reduce_shuffl
shuffle_id: "p15_1"
phase: 1
p16_1(reduce)
options: []
in_local: false
by: ["word"]
phase: 1
p11_1(start)
phase: 1
p17(store)
path: "word_count"
using: null
single: false
useSchema: null
options: []
phase: 1
p18(end)
options: []
phase: 1

代码生成

0.php (~/doc/disql/tech_salon/word_count) - VIM

0.php (~/doc/disql/tech_salon/word_count) - VIM 86

```
/***** vertex section *****/
$__start__ = new Starter;
$__start__->set_id('__start__');

$p13 = new Starter;
$p13->set_id('p13');

$p14 = new Loader(NULL,array (
  '/data\\.in/' =>
  array (
    0 => 'SchemaReader',
    1 =>
    array (
      0 =>
      array (
        'attributes' =>
        array (
        ),
        'children' =>
        array (
          0 =>
          array (
            'attributes' =>
            array (
            ),
            'name' => 'word',
            'type' => 'string',
          ),
        ),
        'name' => 'res',
        'type' => 'struct',
      ),
    ),
  ),
));
$p14->set_id('p14');

$p15_0 = new Grouper(array (
  0 => 'word',
));
```

```
$p12_combiner->set_to('p12'

$p12_0 = new Ender;
$p12_0->set_id('p12_0');

$p15_1 = new Grouper(array
  0 => 'word',
));
$p15_1->set_id('p15_1');

$p15_1_shuffler_map = new M
  0 => 'word',
),NULL,array (
));
$p15_1_shuffler_map->set_id

$__start___0 = new Starter;
$__start___0->set_id('__sta

$__end___0 = new Ender;
$__end___0->set_id('__end_

/***** edge section *****/
$__start__->follow($__start
$p13->follow($__start__);
$p14->follow($p13);
$p15_0->follow($p14);
$p16_0->follow($p15_0);

$p11->follow($p11_0);
$p12_combiner->follow($p11
$p12_0->follow($p12_combine
$p15_1->follow($p16_0);
$p15_1_shuffler_map->follow

$__end___0->follow($p15_1_s
/***** child section *****/
$p16_0->set_children($p11_0
/***** excution section ***
$__start___0->init(0);
```

0.cpp (~/doc/disql/tech_salon/word_count) - VIM

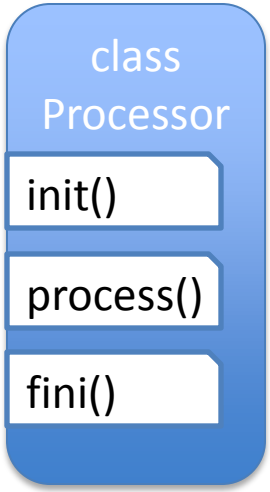
0.cpp (~/doc/disql/tech_salon/word_count) - VIM 78x41

```
#include"Starter___start__.h"
#include"Starter_p13.h"
#include"Loader_p14.h"
#include"Grouper_p15_1.h"
#include"MapShuffler_p15_1_shuffler_ma
#include"Starter___start___0.h"
#include"Ender___end___0.h"
#include<disql/types.h>
#include<disql/Context.h>
#include<disql/Dumper.h>
using namespace disql;

int main(int argc, char *argv[]){
Context::initialize(argc, argv);

bsl::Exception::set_line_delimiter("\n
try{
/*****  vertex section  *****/
Starter___start__ *__start__ = new Sta
Starter_p13 *p13 = new Starter_p13(1);
Loader_p14 *p14 = new Loader_p14(2);
Grouper_p15_1 *p15_1 = new Grouper_p15
MapShuffler_p15_1_shuffler_map *p15_1_
Starter___start___0 *__start___0 = new
Ender___end___0 *__end___0 = new Ender

/*****  child section  *****/

/*****  edge section  *****/
__start__->follow(*__start___0);
p13->follow(*__start__);
p14->follow(*p13);
p15_1->follow(*p14);
p15_1_shuffler_map->follow(*p15_1);
```

```

/*****  edge section  *****/
p12->follow(*p11_1);
p12_1->follow(*p12);

p16_1->follow(*p15_1_shuffler_r
p17->follow(*p16_1);
p18->follow(*p17);
__end__->follow(*p18);
p15_1_shuffler_reduce->follow(*

__end___1->follow(*__end__);

/*****  exec section  *****/
bsl::var::IFactory& factory = C
Value main_record(factory);
__start___1->init(0);
__start___1->process(0, main_re
__start___1->fini(0);
/*****  clean section  *****/
delete p12;
delete p12_1;
delete p11_1;
delete p16_1;
delete p17;
delete p18;
delete __end__;
delete p15_1_shuffler_reduce;
delete __start___1;
delete __end___1;
} catch(bsl::Exception &e){
    fprintf(stderr, "BSL EXCEPT
} catch(std::exception &e){
    fprintf(stderr, "STD EXCEPT
} catch(...){
```

目录

运行时-Processor模型

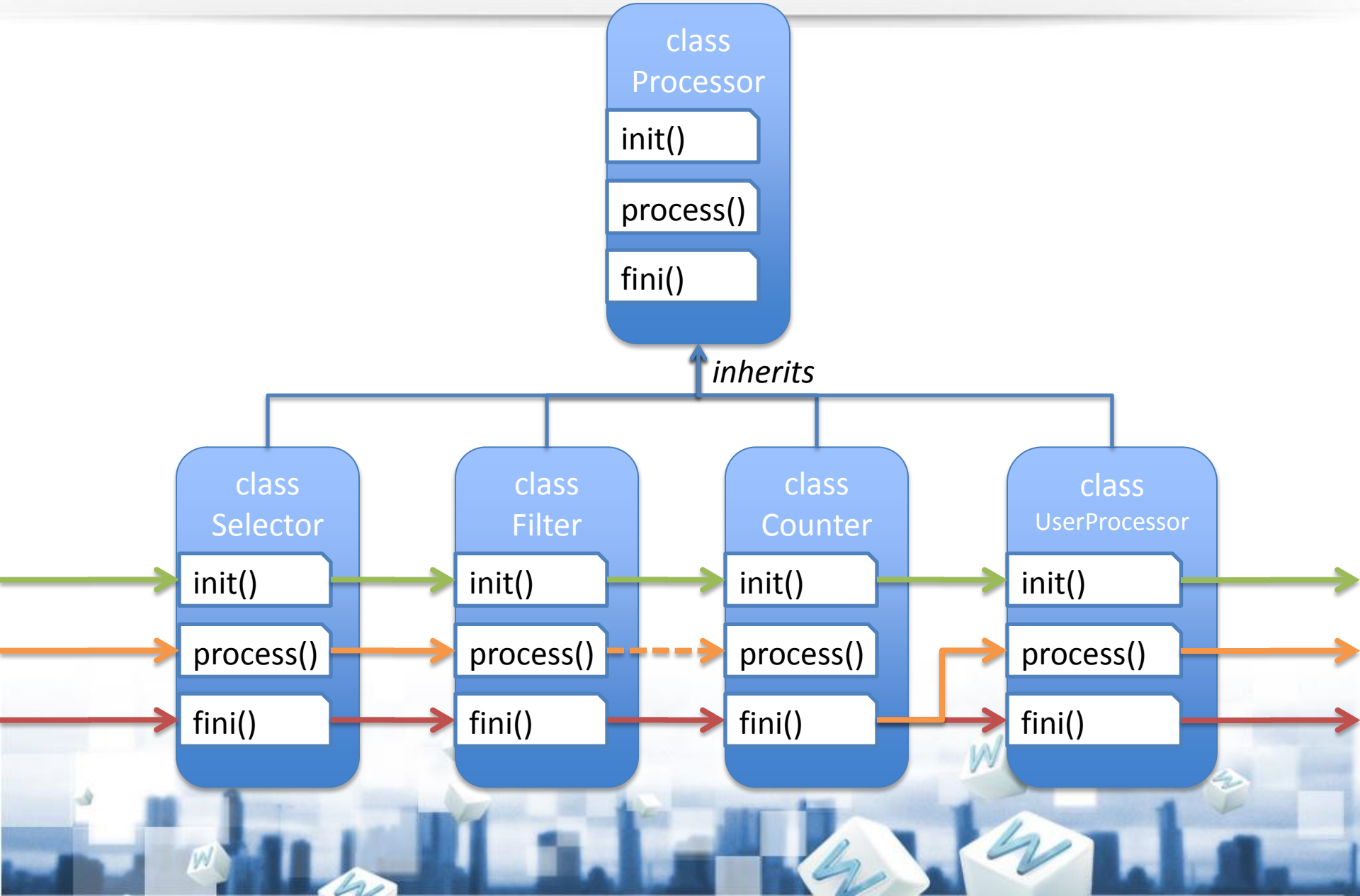


初始化一组数据处理

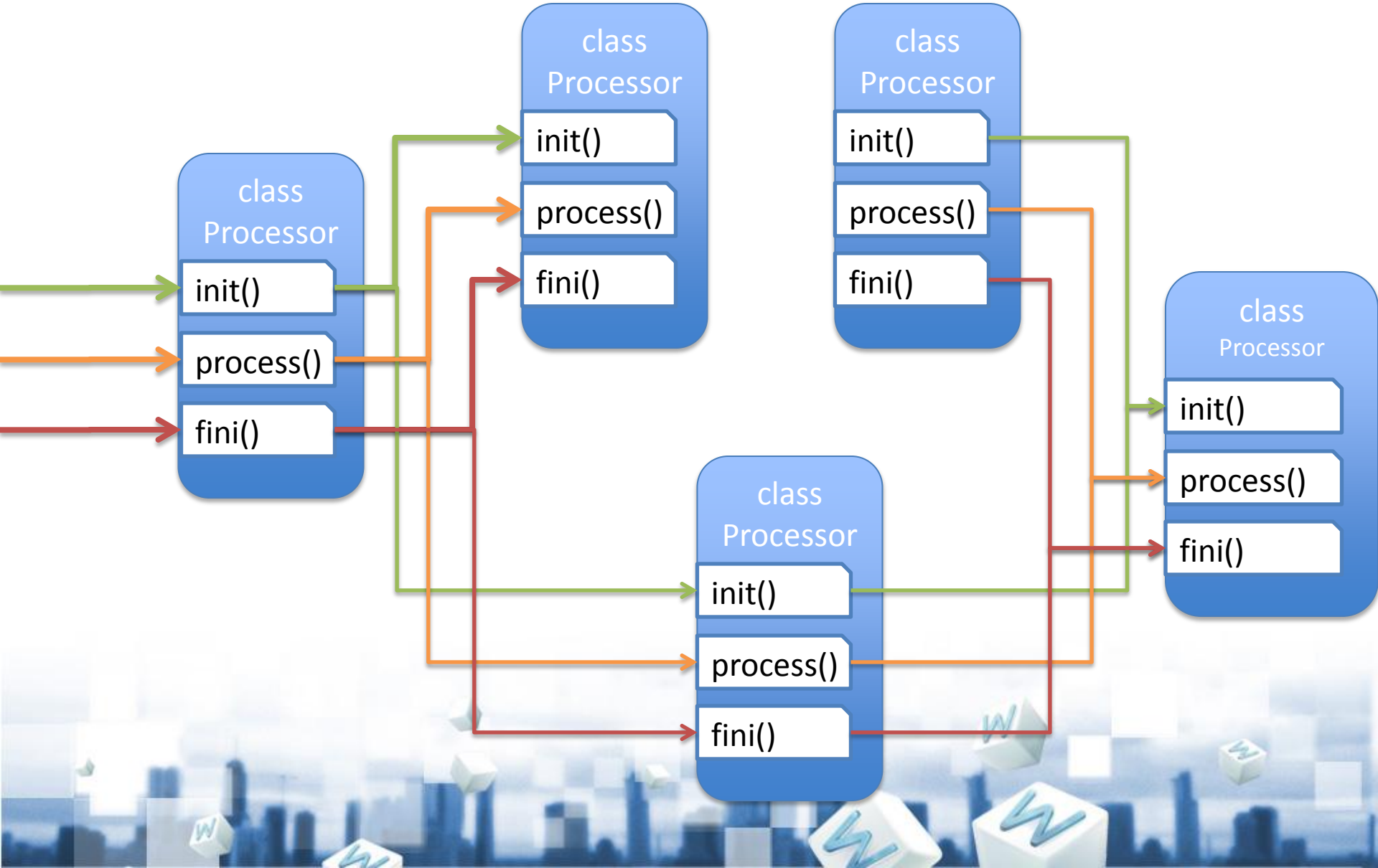
处理一组中的一条数据记录（多次调用）

结束一组数据处理

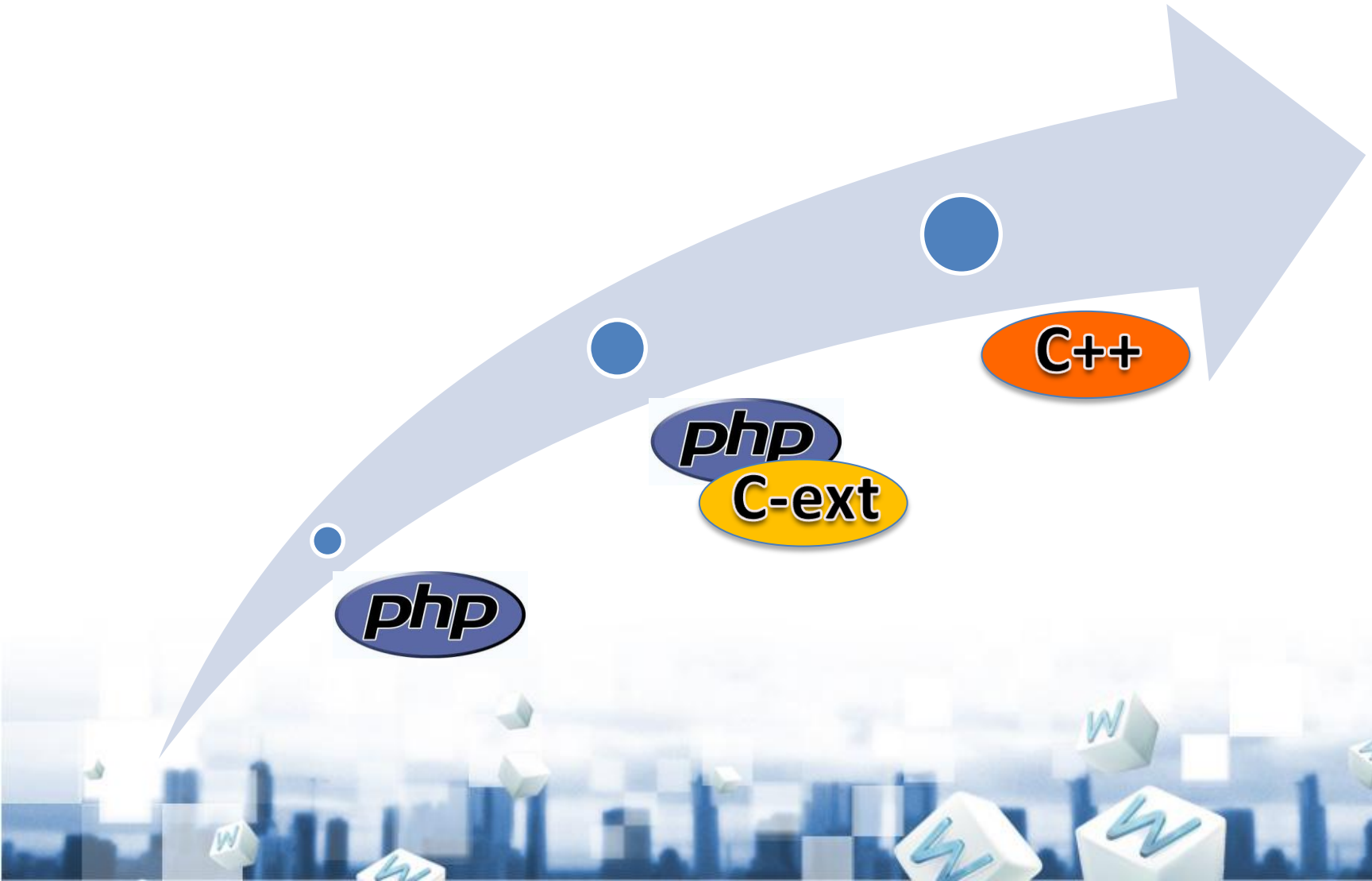
不同算子对应不同的Processor



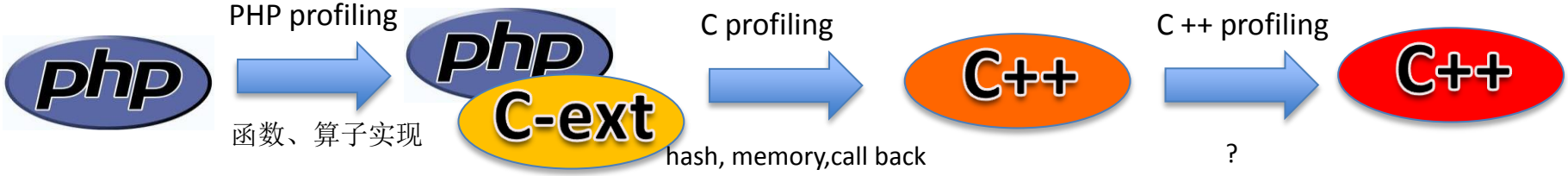
描述DAG数据流图


class
Processor
init()
process()
fini()
class
Processor
init()
process()
fini()
class
Processor
init()
process()
fini()
class
Processor
init()
process()
fini()
class
Processor
init()
process()
fini()

运行时的演化



在profiling指导下优化



C-Profiling发现的三大瓶颈

- Hash操作
- Memory
- User Callback

Hash操作

PHP Runtime

record[“user”][“name”]

C++ Runtime

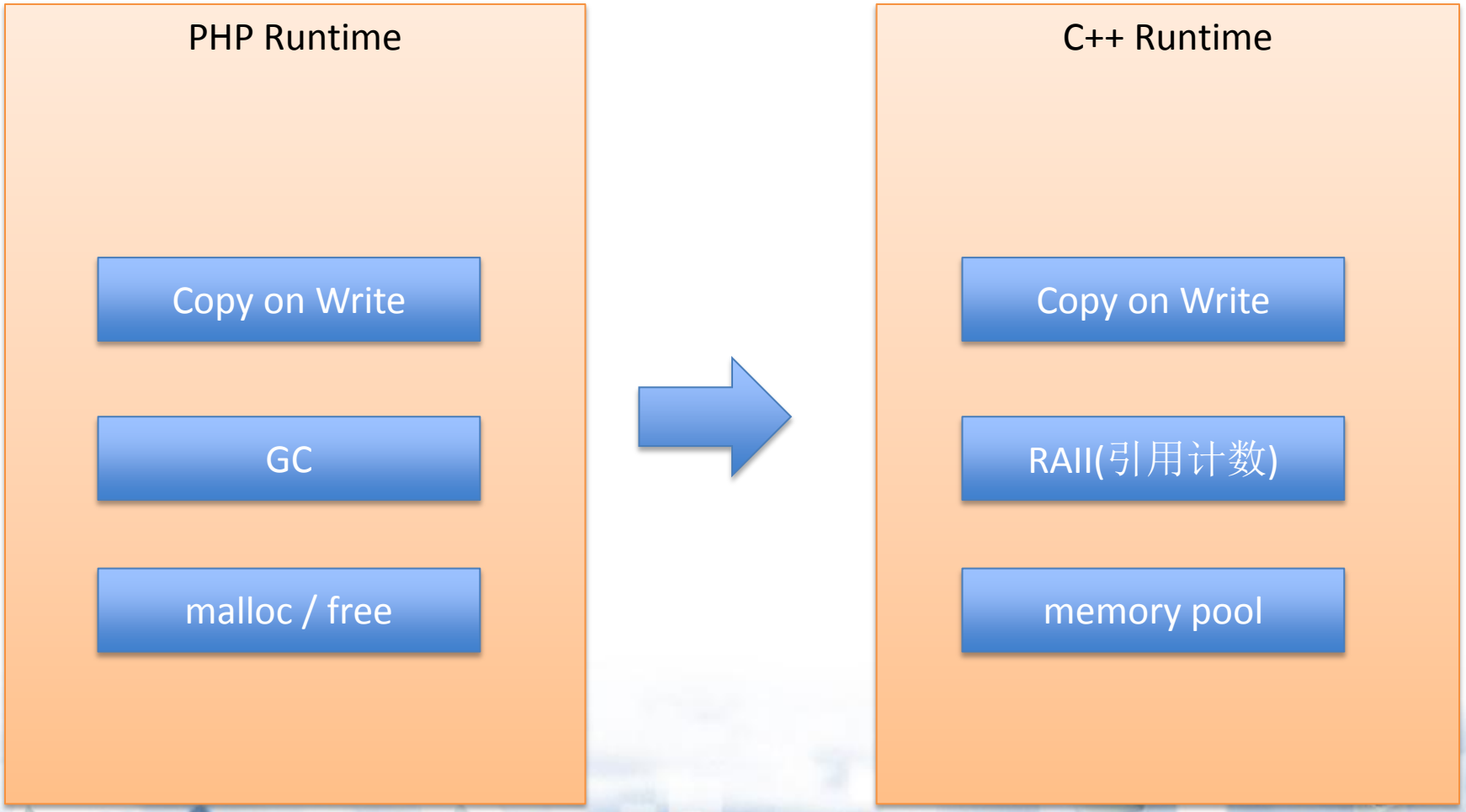
下标推导
const int user= 1;
const int name = 2;

用户代码、算子实现
record[user][name]

compile

record[1][2]

Memory


PHP Runtime
Copy on Write
GC
malloc / free
C++ Runtime
Copy on Write
RAII(引用计数)
memory pool

PHP-Callback

```php
function expand($fields){
    $ret=array();
    for($i=0;$i<count($fields['_Dis'][0]);$i++)    {
        if($fields['_Dis'][0][$i]>0 && $fields['_Cn'][0][$i]!='baidu_fc_gusuan'
   &&$fields['_Cn'][0][$i]!='baidufcidear_pg' && $fields['_Cn'][0][$i]!='baiduadrquery_pg')
{
            for($j=0;$j<$fields['_Dis'][0][$i];$j++){
                $r=array();
                $r['_Srchid']=$fields['_S'];
                $r['_Cmatch']=$fields['_Im_apres'][27][$i][$j];
                $r['_Rank']=(string)$fields['_Absrk'][0][$i][$j];
                $r['query']=$fields['_Query'];
                $r['ip']=$fields['_Ip'];
                $r['cn']=$fields['_Cn'][0][$i];
                $r['bd']=$fields['_Bd'];
                $r['dis']=$fields['_Dis'][0][$i];
                $r['wd']=$fields['_Wd'][0][$i][$j];
                $r['pid']=$fields['_Pid'];
                $r['cid']=$fields['_Cid'];
                $r['tim']=$fields['_Tim'];
                $r['term']=$fields['_Term'][0][$i][$j];
                $r['pres_1']=$fields['_Pres_1'];
                $r['pn']=$fields['_Pn'];
                $r['eq']=$fields['_Eq'];
                $r['qs']=$fields['_Extra']['qs'];
                $r['const1']=1;
                global $valfields;
                foreach($valfields as $val)
                {
                    if($r[$val]==''||$r[$val]==null)
                        $r[$val]='-';
```

位置: 行 1, 字符 26　　总计: 行 41, 字符 1521

C-callback

```
#include "disql_udf.h"
#include "disql_udf_schema.h"

extern "C" void expand(
    bsl::var::LValue& fields, bsl::var::LValue& result, disql::CallbackContext& context) {
    int k = 0;
    if (!fields[ASP_LOG::_Dis].is_array()) return;
    if (!fields[ASP_LOG::_Dis][0].is_array()) return;
    int dis0_size = fields[ASP_LOG::_Dis][0].size();
    for (int i = 0; i < dis0_size; i++) {
        int dis0i;
        const char * cn;
        if (fields[ASP_LOG::_Dis][0][i].is_null() || strcmp(fields[ASP_LOG::_Dis][0]
[i].c_str(), "") == 0) dis0i = 0;
        else dis0i = fields[ASP_LOG::_Dis][0][i].to_int32();
        if (fields[ASP_LOG::_Cn][0][i].is_null()) cn = "";
        else cn = fields[ASP_LOG::_Cn][0][i].c_str();
        if (dis0i > 0 && strcmp("baidu_fc_gusuan", cn) != 0 && strcmp("baidufcidear_pg", cn)
!= 0 && strcmp("baiduadrquery_pg", cn) != 0){
            for (int j = 0; j < dis0i; j++) {
                result[k][EXPAND::_Srchid]      = fields[ASP_LOG::_S];
                result[k][EXPAND::_Cmatch]      = fields[ASP_LOG::_Im_apres][27][i][j];
                result[k][EXPAND::_Rank]        = fields[ASP_LOG::_Absrk][0][i][j];
                result[k][EXPAND::query]        = fields[ASP_LOG::_Query];
                result[k][EXPAND::ip]           = fields[ASP_LOG::_Ip];
                result[k][EXPAND::cn]           = fields[ASP_LOG::_Cn][0][i];
                result[k][EXPAND::bd]           = fields[ASP_LOG::_Bd];
                result[k][EXPAND::dis]          = fields[ASP_LOG::_Dis][0][i];
                result[k][EXPAND::wd]           = fields[ASP_LOG::_Wd][0][i][j];
                result[k][EXPAND::pid]          = fields[ASP_LOG::_Pid];
                result[k][EXPAND::cid]          = fields[ASP_LOG::_Cid];
```

位置: 行 1, 字符 23　　总计: 行 50, 字符 2520

依赖C++库

- 依赖百度在C++编程上的丰富积累，尤其是BSL库
  - BSL = Baidu Standard Library，使用最广泛的基础库
  - 各种高效的泛型容器
  - 内存池
  - 不弱于Java异常的异常支持
  - 动态类型支持
    - 像写PHP那样编写C++代码
  - 格式转换
    - JSON、内部二进制格式……
  - 语言扩展支持
    - PHP、Python、Perl……

目录

代码规模

- 前端
  - DQuery: 2037
- 中间语言翻译
  - 4552
- 运行时
  - PHP Runtime: 7283
    - PHP: 2842
    - C Extension: 4441
  - C++ Runtime: 8318

特点总结

- 轻量级
- 类SQL逻辑(非常简约)
  - 封装所有SQL算子的M/R分布式实现：
    - 分组、聚合、表连接、行列过滤、集合操作、输入输出格式转换
- DAG数据流
  - 自动翻译为一轮或多轮MapReduce
  - 也可翻译为单机计算或数据流图
- 逻辑顺序而非SQL顺序
- 支持PHP自定义函数（简洁）
- 支持C++自定义函数（同样简洁+高效）和C-Runtime NEW!
  - 全自动高效内存管理 （RAII + 内存池）
  - 廉价对象复制（Copy On Write）
  - 字段操作翻译为数组操作，无字典查找 （schema推导）
  - C++的性能，PHP的开发代价！

分析程序数量增长

程序类型	4月1日	10月27日	增长	增长百分比
简单编辑	3540	4761	1221	+34.5%
DQuery模式	1153	3359	2206	+191%
复杂编辑	1569	2963	1394	+88.9%

分析程序输入量

程序类型	占比
Web表单	24%
DQuery模式	43%
裸MapReduce	33%

} 67%

用户数量

角色	数量	占比
PM	1352	47.4%
RD	1174	41.2%
OP	190	6.66%
其他	136	4.77%
总数	2852	100%

Q & A

更多问题，可以向微博@陈晓鸣在百度提问

谢谢！