阿里巴巴 Alibaba.com | 广告和搜索中心 Advertising and Search Center

分布式流数据实时计算平台
iprocess

强琦
搜索平台

ASC

Outline

- 业界的分布式计算产品
- 实时计算的背景
- 当前产品的应用场景和局限性
- 搜索，广告业务实时计算的需求
- 分布式流数据实时计算平台iprocess简介
- iprocess的架构和计算模型
- 如何编写iprocess的job
- 应用和进展
- 实际例子
- 未来规划
- Q&A

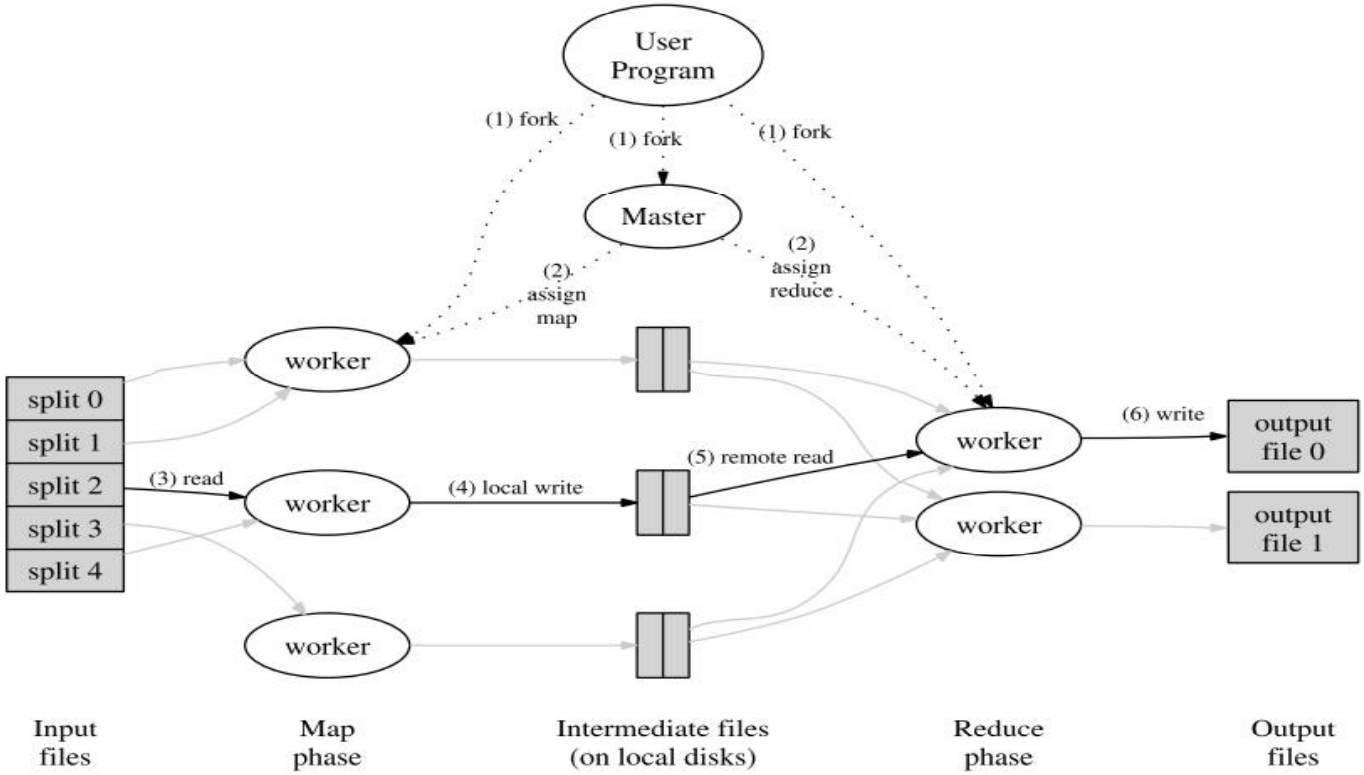
业界的分布式计算产品

- GFS2,Bigtable,Megastore,Spanner
- Mapreduce,Hadoop Online,Mapreduce merge
- S4,Dryad,Pregel,Dremel
- Cloudscale, perccolator, Caffeine
- …

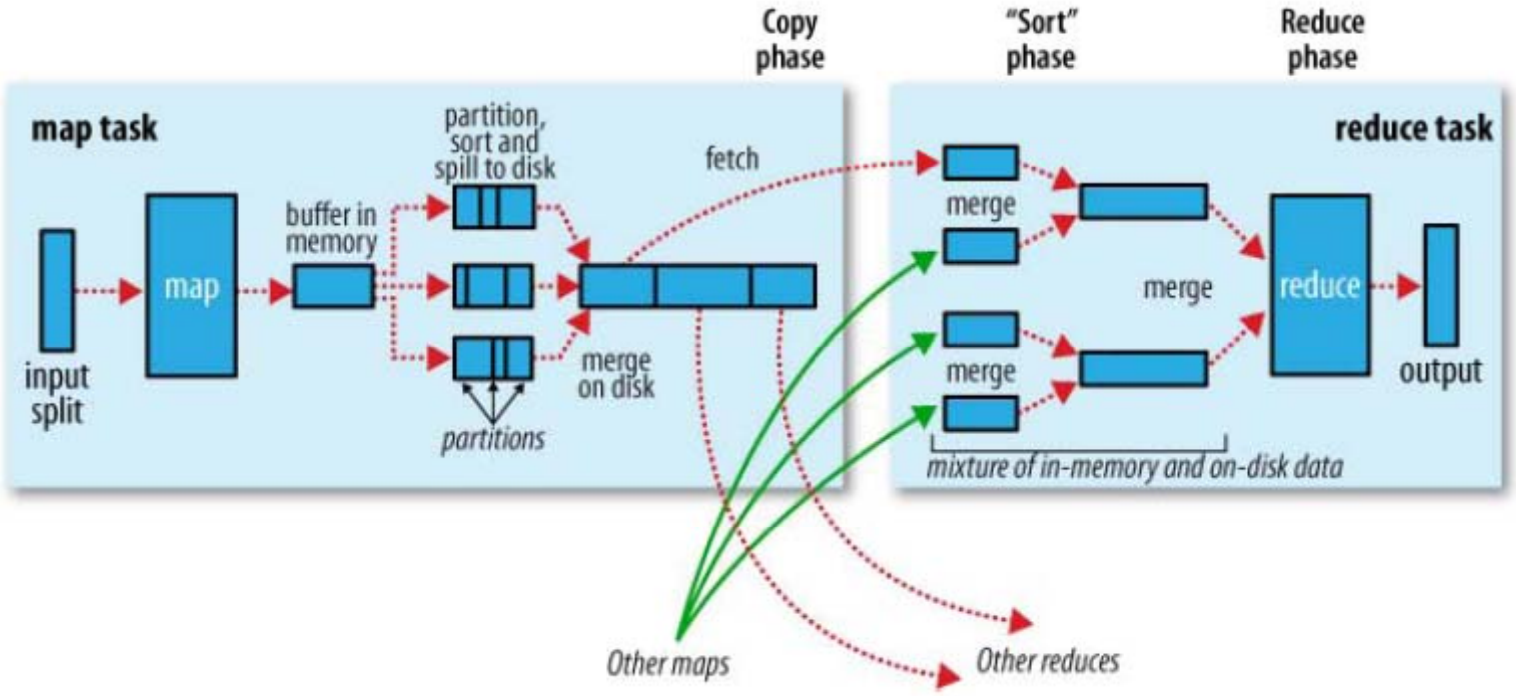
方法论

- What?是什么？
- Why？为什么？
- Why not？为什么我没有想到这一点？
- What？它有什么缺点？
- Better？有没有更好的？
- WWWWB会贯穿整个分享。

Mapreduce



Mapreduce-shuffle



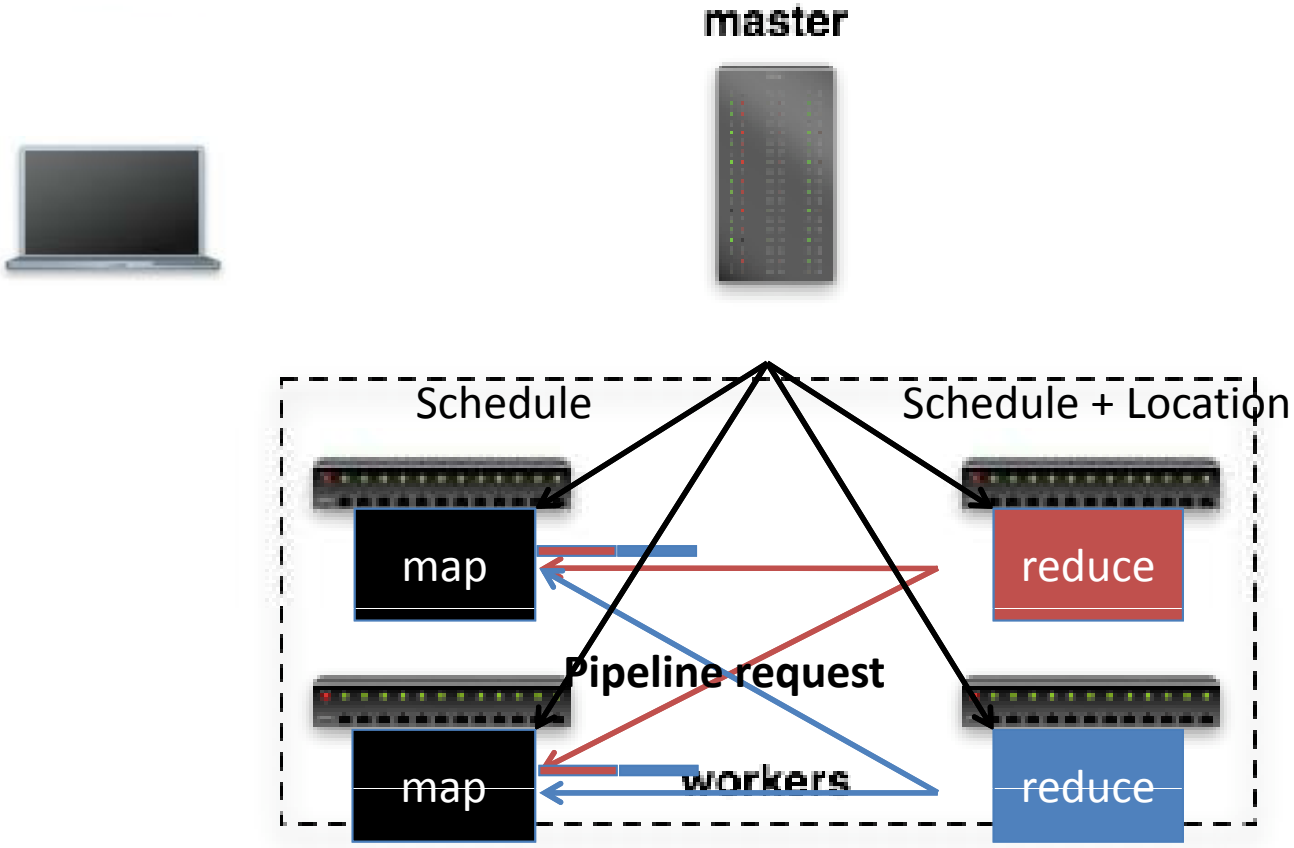
Mapreduce-接口

- Map
- Reduce
- Combine
- Partition
- Group

Mapreduce特点和应用场景

- 特点
  - LOCAL
  - 单job内串行
  - 高吞吐量
  - 模型简单
  - 。。。
- 应用场景
  - “重”计算（批量计算）
  - Data balance = computation balance
  - 非图运算
  - 非迭代运算

Hadoop Online



Hadoop online

- WWWWB?

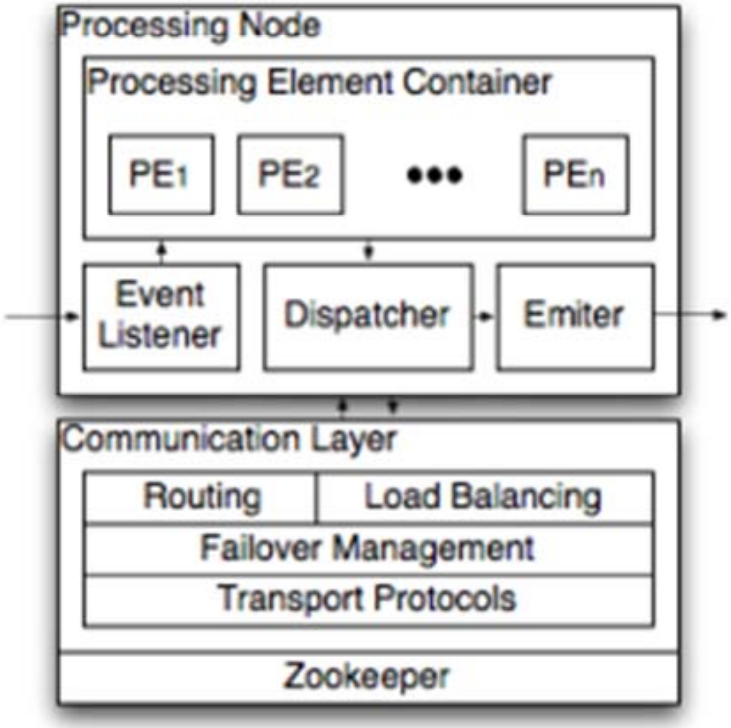
Mapreduce merge

$$\begin{aligned}
\text{map:}\quad & (k_1, v_1)_\alpha \rightarrow [(k_2, v_2)]_\alpha \\
\text{reduce:}\quad & (k_2, [v_2])_\alpha \rightarrow (k_2, [v_3])_\alpha \\
\text{merge:}\quad & ((k_2, [v_3])_\alpha, (k_3, [v_4])_\beta) \rightarrow [(k_4, v_5)]_\gamma
\end{aligned}$$

map: (k1, v1) → [(k2, v2)]
reduce: (k2, [v2]) → [v3]

- WWWWB?

S4



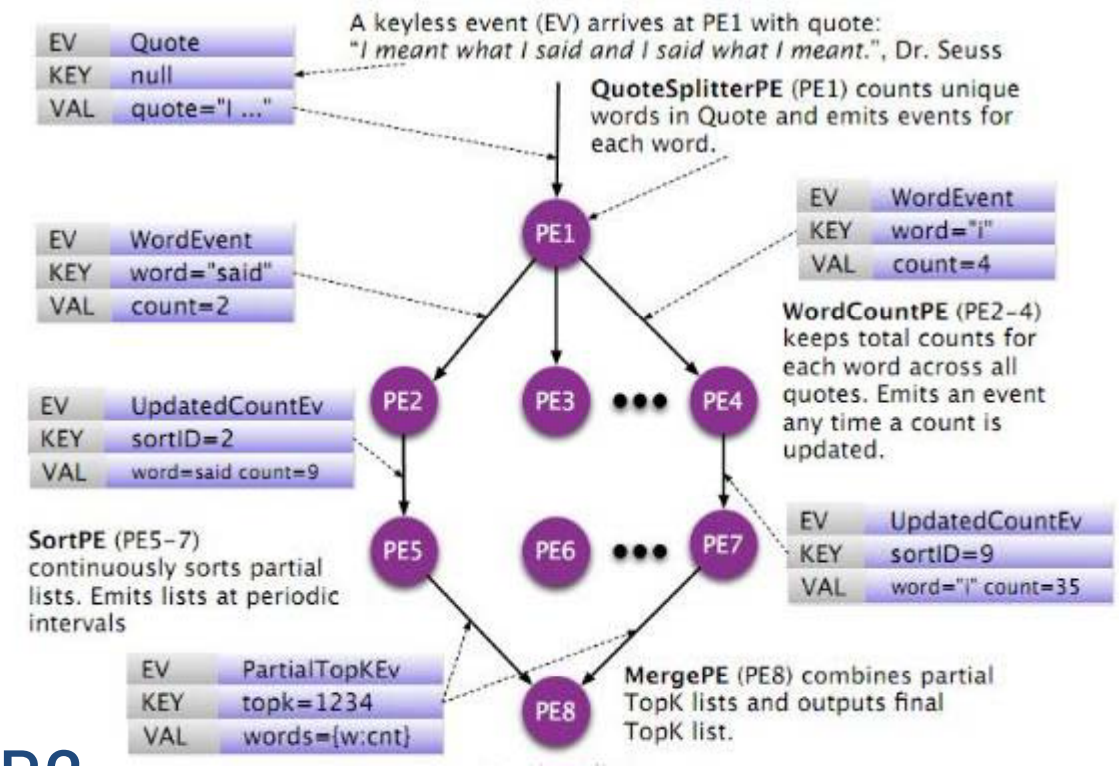

```
private queryCount = 0;

public void processEvent(Event event)
{
  queryCount ++;
}

public void output()
{
  String query = (String) this.getKeyValue().get(0);
  persister.set(query, queryCount);
}
```

```
<bean id="queryCounterPE"
      class="com.company.s4.processor.QueryCounterPE">
  <property name="keys">
    <list>
      <value>QueryEvent queryString</value>
    </list>
  </property>
  <property name="persister" ref="externalPersister">
  <property name="outputFrequencyByTimeBoundary"
            value="600"/>
</bean>
```

S4

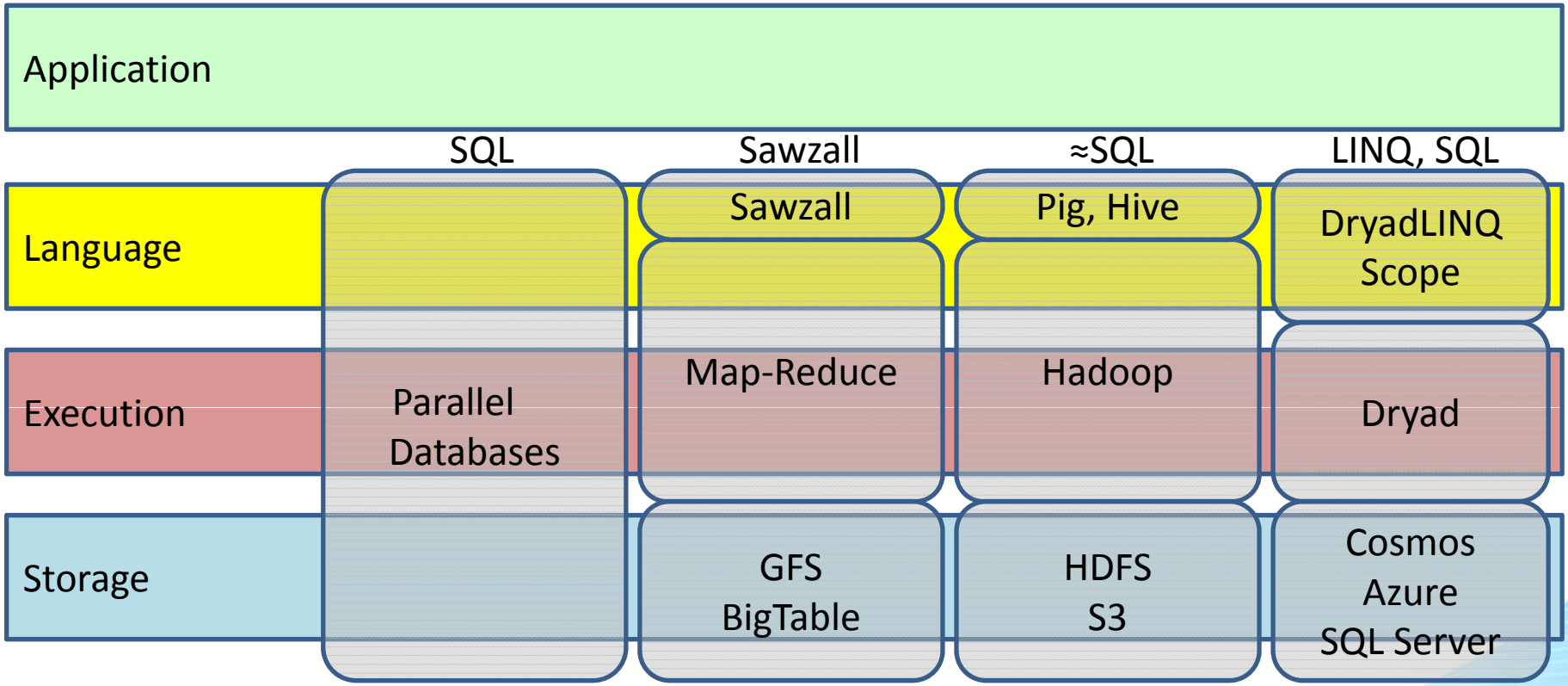


- WWWWB?

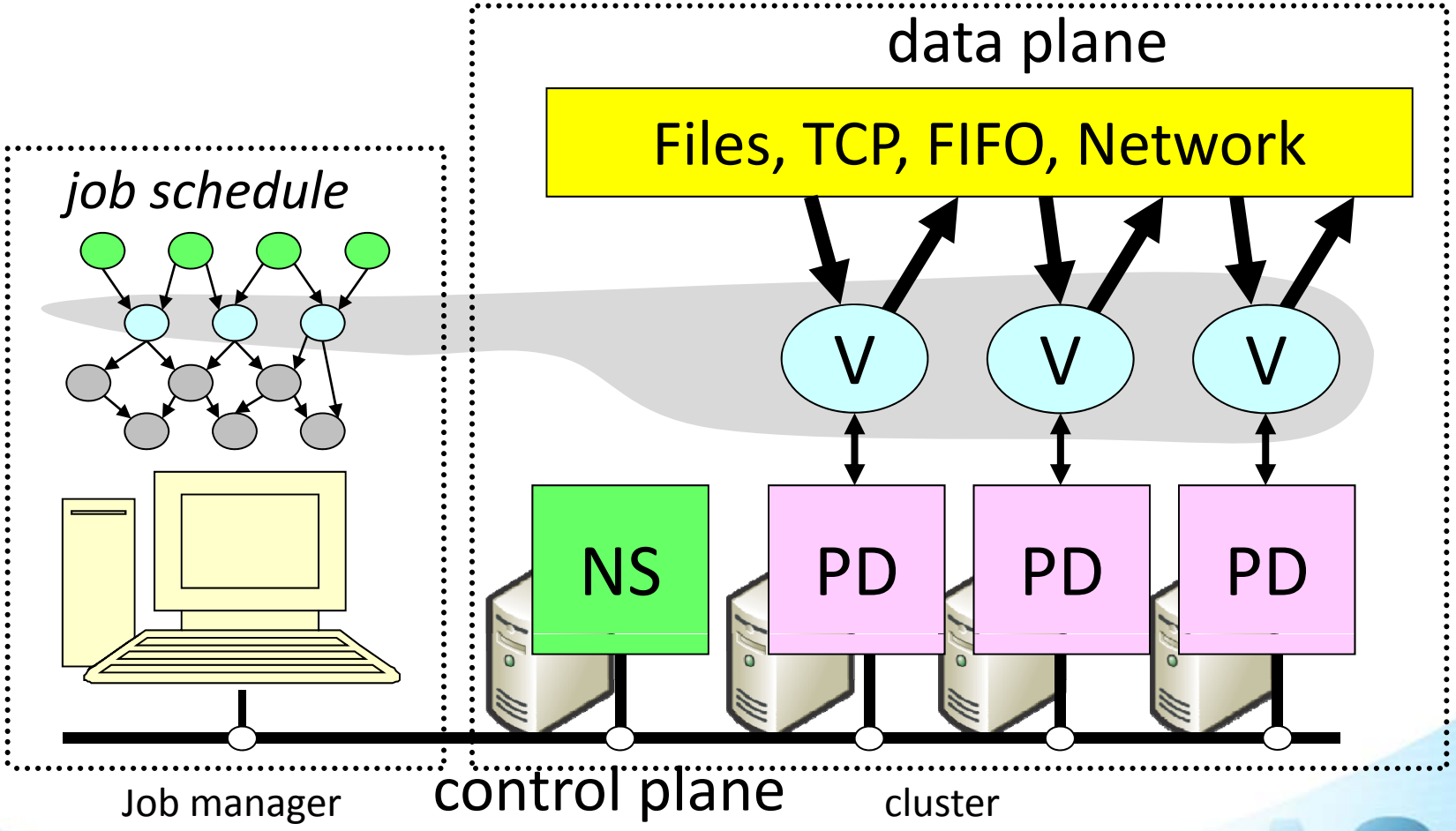
Dryad &&DryadLINQ

- Dryad is a general-purpose distributed execution engine for coarse-grain data-parallel applications. A Dryad application combines computational “vertices” with communication “channels” to form a dataflow graph. Dryad runs the application by executing the vertices of this graph on a set of available computers, communicating as appropriate through files, TCP pipes, and shared-memory FIFOs

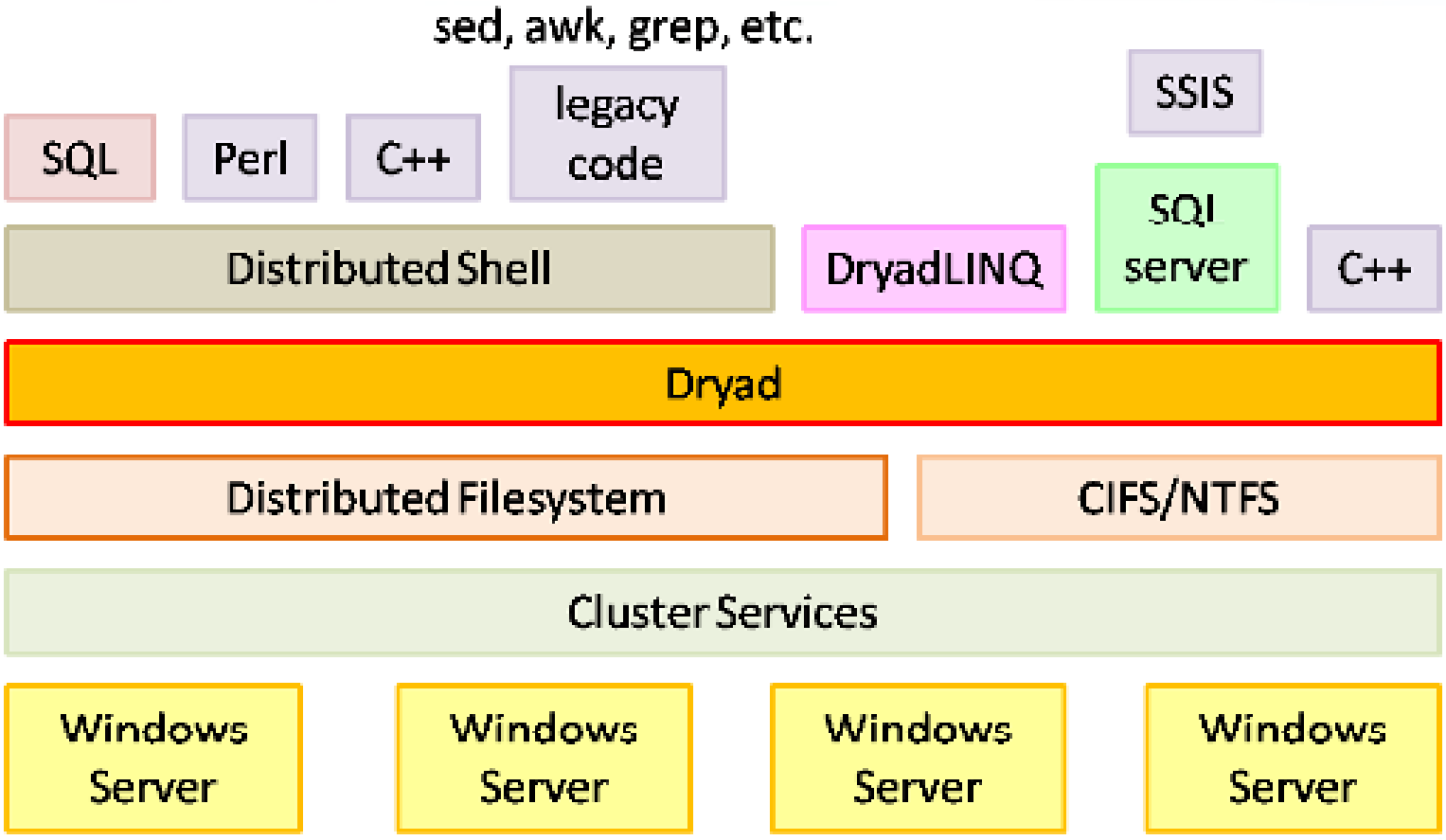
Dryad



Dryad System Architecture



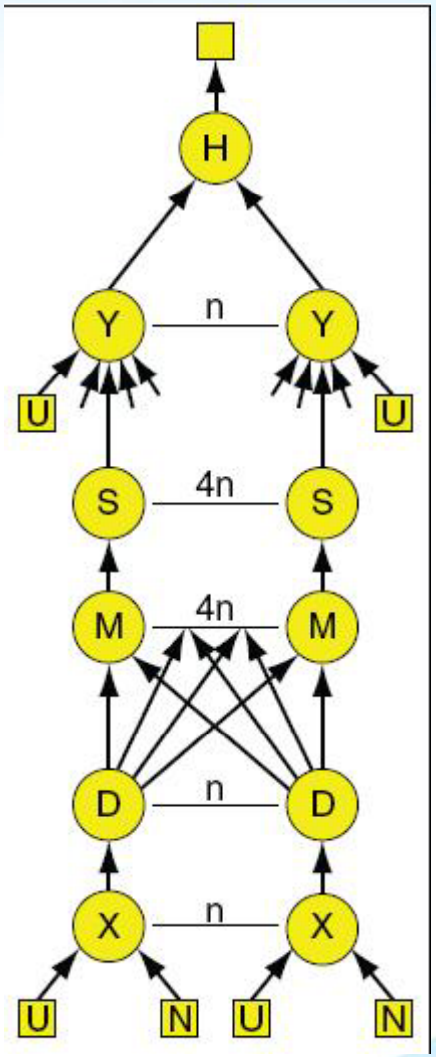
Dryad



Dryad

- select distinct p.objID
- from photoObjAll p
- join neighbors n — *call this join "X"*
- on p.objID = n.objID
- and n.objID < n.neighborObjID
- and p.mode = 1
- join photoObjAll l — *call this join "Y"*
- on l.objid = n.neighborObjID
- and l.mode = 1
- and abs((p.u-p.g)-(l.u-l.g))<0.05
- and abs((p.g-p.r)-(l.g-l.r))<0.05
- and abs((p.r-p.i)-(l.r-l.i))<0.05
- and abs((p.i-p.z)-(l.i-l.z))<0.05

- **WWWWB?**



Pregel

- 图算法可以被写成是一系列的链式mapreduce作业。
- 可用性和性能。Pregel将顶点和边在本地机器进行运算，而仅仅利用网络来传输信息，而不是传输数据。
- MapReduce本质上是面向函数的，所以将图算法用mapreduce来实现就需要将整个图的状态从一个阶段传输到另外一个阶段，这样就需要许多的通信和随之而来的序列化和反序列化的开销。
- mapreduce作业中各阶段需要协同工作也给编程增加了难度，这样的情况能够在Pregel的各轮superstep的迭代中避免。

Pregel

```
template <typename VertexValue,
          typename EdgeValue,
          typename MessageValue>
class Vertex {
 public:
  virtual void Compute(MessageIterator* msgs) = 0;

  const string& vertex_id() const;
  int64 superstep() const;

  const VertexValue& GetValue();
  VertexValue* MutableValue();
  OutEdgeIterator GetOutEdgeIterator();

  void SendMessageTo(const string& dest_vertex,
                     const MessageValue& message);
  void VoteToHalt();
};
```

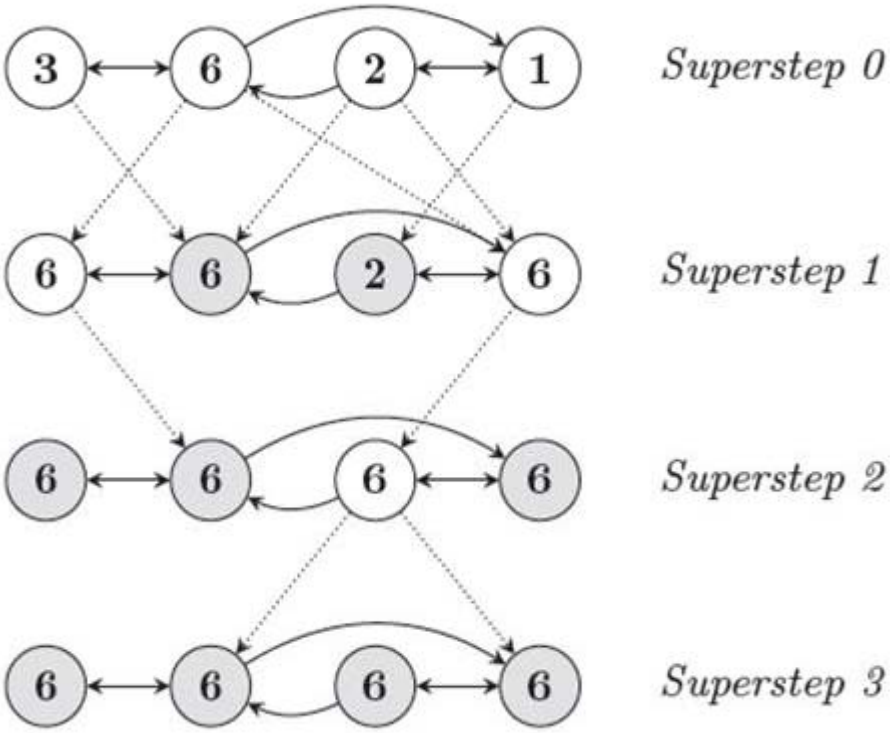
Pregel

```
class PageRankVertex
    : public Vertex<double, void, double> {
 public:
  virtual void Compute(MessageIterator* msgs) {
    if (superstep() >= 1) {
      double sum = 0;
      for (; !msgs->Done(); msgs->Next())
        sum += msgs->Value();
      *MutableValue() =
          0.15 / NumVertices() + 0.85 * sum;
    }

    if (superstep() < 30) {
      const int64 n = GetOutEdgeIterator().size();
      SendMessageToAllNeighbors(GetValue() / n);
    } else {
      VoteToHalt();
    }
  }
};
```

Figure 4: PageRank implemented in Pregel.

Pregel



- WWWWB?

Dremel

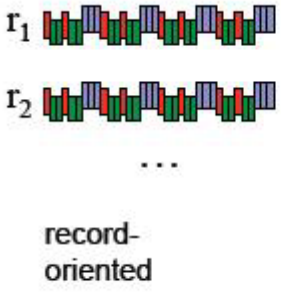
- 应用场景只读，不会涉及到迁移等。DW。
- 完全的按列存储，没有附加字段，如果需要返回正行则时间比较多。
- 嵌套结构，展平为row不太容易，按列存储需要恢复结构。近似值，因为可以早停。

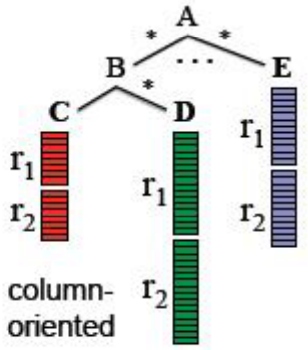
Dremel

```
DocId: 10                    r1
Links
  Forward: 20
  Forward: 40
  Forward: 60
Name
  Language
    Code: 'en-us'
    Country: 'us'
  Language
    Code: 'en'
  Url: 'http://A'
Name
  Url: 'http://B'
Name
  Language
    Code: 'en-gb'
    Country: 'gb'
```

```
message Document {
  required int64 DocId;
  optional group Links {
    repeated int64 Backward;
    repeated int64 Forward; }
  repeated group Name {
    repeated group Language {
      required string Code;
      optional string Country; }
    optional string Url; }}
```

```
DocId: 20                    r2
Links
  Backward: 10
  Backward: 30
  Forward:  80
Name
  Url: 'http://C'
```

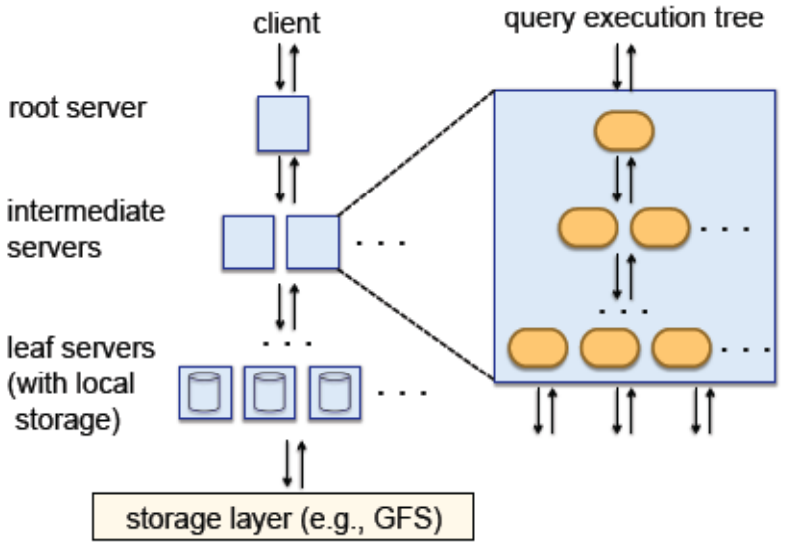




Dremel

- Splitting Records into Columns
- Record Assembly
- Distributed
- QUERY EXECUTION
- mr可以从column获利

Table name	Number of records	Size (unrepl., compressed)	Number of fields	Data center	Repl. factor
T1	85 billion	87 TB	270	A	3×
T2	24 billion	13 TB	530	A	3×
T3	4 billion	70 TB	1200	A	3×
T4	1+ trillion	105 TB	50	B	3×
T5	1+ trillion	20 TB	30	B	2×



- WWWWB?

others

- Cloudscale
- Perccolator
- Caffeine

Bill McColl

Bill left Oxford University to found Cloudscale, a
Computer Science, Head of the Parallel Compu
Faculty. Along with Les Valiant of Harvard, he d
research, product, and business teams, in a nu
supercomputing, parallel programming languag
and cloud computing. He lives in Palo Alto, CA.

Tony Faustini

Tony spent 10 years at Microsoft Silicon Valley
Windows, WebTV, and Legal and Corporate Af
Standards. Prior to Microsoft, Tony spent three
developed the award-winning Java Studio prod
Laboratory at SRI International and a Tenured
Science, specializing in parallel dataflow syster

实时计算的背景

- 交互越来越频繁
- 利用用户反馈越来越模切
- 商业模式越来越实时（实时搜索、高频率交易、社交网络）
- 用户需求越来越个性化，实时化
- 海量流数据的产生
- Mapreduce应用于批量计算场景-强调吞吐量
- 实时计算则强调单个记录的latency(freshness)

当前产品的不足和局限性

- Mapreduce：批处理，高吞吐量，LOCAL，DATA BALANCE !=COMPUTATION BALANCE,同构数据(join),无法实时，无法并行，etc…
- mapreduce online：取巧的方法
- mapreduce merge：将异构数据装入mr的框里
- S4：容错，业务逻辑复杂，checkpoin实现复杂
- Dryad：使用复杂，模型不比mr强大
- Pregel：专款专用
- Dremel：专款专用

搜索，广告业务实时计算的需求

- 不同于网页搜索
- 付费用户，对实时性要求更高
- 实时状态
- 收益更依赖平台，效果驱动，小修改大动作
- 广告数据修改非常频繁
- 网站特殊业务逻辑

分布式流数据实时计算平台iprocess简介

- 一个分布式流数据实时计算引擎和平台。基于该引擎的应用系统可以建模为有向图（非DAG)，其中“每个节点”为一个用户编写的插件，而”边“为插件定义的事件。
- 基于iprocess的服务为：一张流程图，以及对应的节点插件和注册事件。iprocess保证了上层系统的可扩展性，一致性和高可用性，并实现了实时计算，增量/批量，全量计算的并行和一致性。
- 服务化
- 目前可支持常驻任务，迭代任务等，支持多场景，多任务隔离，层级目录结构，每个目录结构都可以定制一致性策略和事务类型。可支持增量MAP/REDUCE,支持灵活的事务冲突管理机制,分布式跨行跨表事务。支持物化视图和二级索引，支持高效的增量JOIN。iprocess的内核实现高度的插件化，后续的系统功能扩充只需编写系统级插件即可，例如：join功能，二级索引，迭代任务等，只需编写相应的内核级插件。

Iprocess-应用场景和特点

- 应用场景
  - 流数据
  - 实时计算
  - 离线计算
- 特点
  - 基于事件
  - 完全不同于MR的计算模型
  - 兼容MR的模型
  - 兼容pregel模型
  - 简单的说是s4,hadoop online，percolator，pregel，dryad的混合体

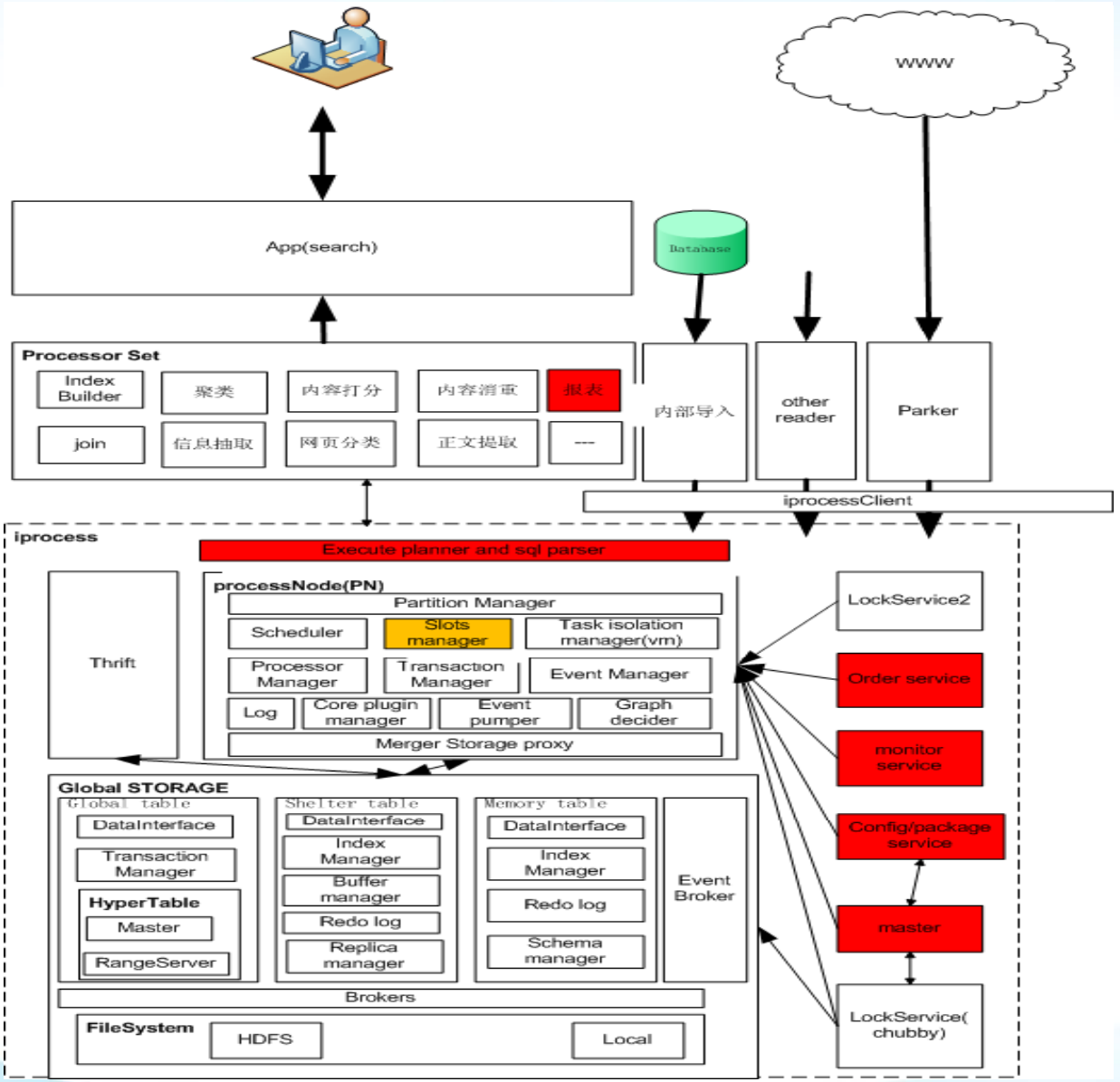
简介–基本事件

- Record
  - add record
  - batch add record
  - timer record
  - full record
  - del record
  - modify record
- Segment
  - add segment
  - del segment
  - modify segment
  - occupy segment

简介–基本接口

- int init(Context context);
- int process(DataItem doc);
- int resolve(List<Value> valueSet);
- int uninit();
- 扩展接口…

iprocess的架构和计算模型



Stream realtime computing

Hadoop online

s4

iprocess主要角色

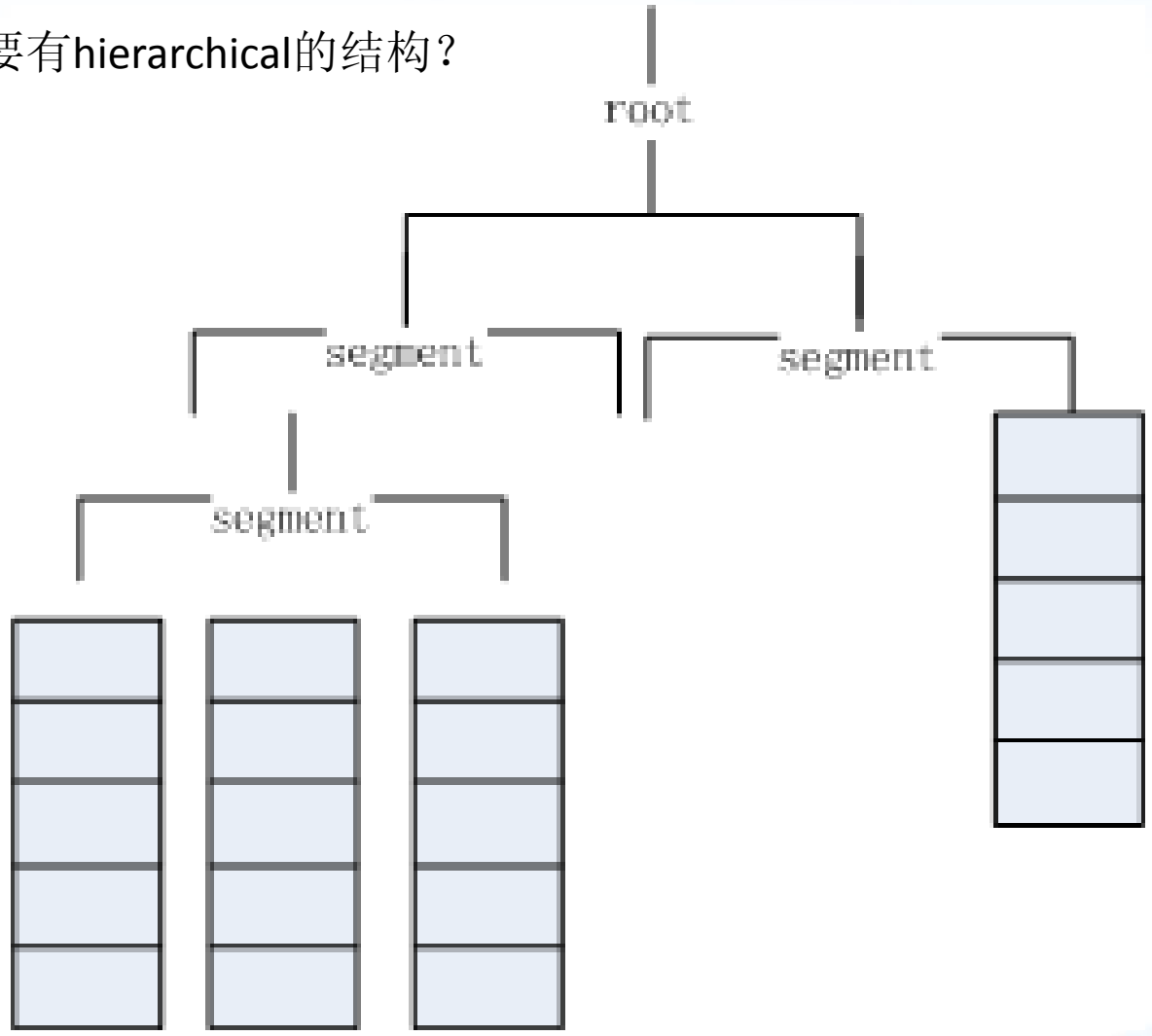
- Master：总控，任务加载配置
- PN:实际的执行节点。
- 松粒度lockservice与高性能lockservice
- 时钟服务
- 配管，监控
- IDE，模拟环境，调试环境

iprocess-实时

- MR为什么达不到实时？
- iprocess如何做到实时？
  - 记录级
  - 流数据
  - 事件触发
  - 实时条件判断

iprocess-逻辑存储结构

为什么要有hierarchical的结构？



iprocess的架构

- Memory table:s4，
  - 优点：快，
  - 缺点：容错，共享
- Shelter table：hadoop online
  - 优点：吞吐量，容错
  - 缺点：实时性
- Global table：stream realtime computing
  - 优点：全局共享，realtime，容错
  - 缺点：吞吐量

iprocess平滑

- 三者之间平滑过度
- 可以根据业务逻辑，数据特点制定参数
- 可以根据SLA来在一套系统下同时运行

只有mt时，蜕变为s4

只有st时，蜕变为mr

只有gt时，类似咖啡因后台系统

三者可以任意组合，加以不同权重则显示出不同特点：1.响应时间；2.吞吐量；3.容错　的组合

Iprocess-主要过程

- 插入数据
- 判断条件是否满足
- 满足条件，则触发相应事件

- 响应事件
- 查询注册processor
- 调度相应slots
- 任务隔离模块加载
- 执行

- 死锁检测
- 异常检测
- 异常数据清理

iprocess扩展接口

- Map
- Reduce:同构数据ok，异构数据？Join例子
  Reduce(key, list<v>)→Reduce(key, tree<v>)
- Combine
- Partition：1对1，多对1，1对多
- No group

- 均为基础接口扩展而来

如何编写iprocess的job-输入

输入

- **Table类：**
  Table::open();
  Table::addSegment(Segment* pParent, string name, segInfo*);
  ScanStream* Table::startScan(schema);
  int32_t Table::endScan();
- Segment类
  Row* Segment::get(key, schema);
  int32_t Segment::set(key, row);
  ScanStream* Segment ::startScan(schema);
  int32_t Segment ::endScan();
  Segment* Segment::getParent();
- **Row类:**
  Row::get(column);
  Row::set(column, pValue);
- **ScanStream类：**
  Row* ScanStream::Next();

如何编写iprocess的job响应

- 响应事件：
  - int process(DataItem doc);
  - int resolve(list<Value> valueSet);
  - vector<VM> Partition(doc);
  - 扩展接口都是process函数的再定制
    - Reduce(key, list<v>)
    - Reduce(key, tree<v>)
    - Reduce(key, tree<v>)

Iprocess-locality

- Mt
- St,gt与pn部署在相同物理机器
- 数据的局部性由master来监控st，gt的master或者lockservice的metainfo，根据策略进行动态的processor迁移或者暂时组织数据的迁移或者容忍数据的迁移
- Push computation to the data

计算能力

- Join是搜索广告应用最重头的离线计算
- 以前的join都是批量，大大制约搜索的实时性
- 接下来会主要以join为例，说明mapreduce为代表的批量计算和iprocess的计算区别。。。

Job-join

- Two-way Joins; Multi-way Joins

$$P \bowtie_{a=b} Q \qquad\qquad P_1 \bowtie_{a_1=a_2} P_2 \bowtie_{a_2=a_3} \cdots \bowtie_{a_{n-1}=a_n} P_n$$

- 星形；链式；混合
- 执行计划，优化

Job-join

- Hadoop玩法
  - Reduce-Side Join，注意group

```
void map(Text key, Text values,
    OutputCollector<TextPair, TextPair> output, Reporter
       reporter) throws IOException {
    output.collect(new TextPair(key.toString(), tag),
          new TextPair(values.toString(), tag));
}
int getPartition(TextPair key, TextPair value, int
   numPartitions) {
  return (key.getFirst().hashCode() & Integer.MAX_VALUE)
     % numPartitions;
}
```

Job-join

```
void reduce(TextPair key, Iterator<TextPair> values,
      OutputCollector<Text, Text> output,  Reporter
         reporter)
      throws IOException {
  ArrayList<Text> T1 = new ArrayList<Text>();
  Text tag = key.getSecond();
  TextPair value = null;
  while(values.hasNext())
  {
    value = values.next();
    if(value.getSecond().compareTo(tag)==0)
    {
      T1.add(value.getFirst());
    }
    else
    {
      for(Text val : T1)
      {
        output.collect(key.getFirst(),
          new Text(val.toString() + "\t"
          + value.getFirst().toString()));
      }
```

Job-join

- Hadoop玩法
  - **Map-Side Join**
- All datasets must be sorted
- All datasets must be partitioned
- The number of partitions in
- the datasets must be identical.
- A given key has to be in the same partition in each dataset

Job-join

- Hadoop 玩法
  - Broadcast Join
  - Hash join
  - Part join
  - Etc…

iprocess-join

- 物化view. A.2=B.1 AND B.2=C.1

```
View
{
    table a;
    view
    {
            table b;
            table c;
            joinFiled b.2, c.1;
    }bc
    joinFiled a.2, bc.1;
}
```

iprocess-join

- 执行优化
- Reduce(key, tree<v>, output)

```
{
        for(ancestor)
        …
        for(parent)
        {
                output.commbit(compoundkey(key, parent), ancestor.getvalue(), getvalue(key));
        }
}
```

笛卡尔积

iprocess-join

- 两层
- Reduce(key, tree<v>, output)

```
{
        for(parent)
        {
                output.commbit(compoundkey(key,
parent), (parent.getvalue(), getvalue(key);
        }
}
```

笛卡尔积，当然是offer表，parent是member表。

iprocess-join

- 以两层join为例
- Reduce(sKey,tree<v>)

```
{
        for(childs)
        {

        output.commbit(compoundkey(getkey(for(child
  s), sKey), getvalue(sKey), getvalue(childs));
        }
  }
```

笛卡尔积，当然是offer表，parent是member表。

iprocess-join

- 多路join，迭代进行
- 已经实现缺省的JOIN，例如ADD,LIST,笛卡尔积(**Cartesian**)。
- 类似的wordcount，简单倒排索引，join直接制定EVENT.BIND(**Cartesian**);无需编写代码。

Table join

```
Table* table = Open(“/qstore/jointable”);
While(1){
Record_a = readRecord(A);
If(record_a == NULL)
Break;
Row* r1 = new Row(T, getJoinKey(Record_a));
r1->set(“content”, “table_a”, record_a);
}
//读取表B
Table* table = Open(“/qstore/jointable”);
While(1){
Record_b = readRecord(B);
If(record_b == NULL)
Break;
Row* r1 = new Row(T, getJoinKey(Record_b));
r1->set(“content”, “table_b”, record_b);
}

Int32_t process(DataItem doc)
{
  Return buildIndex(doc);
}
```

iprocess-join

- Reduce-side,map-side,hash-side,bc-side,etc
- Partition: 1:1;1:1,1:n, part

iprocess-join特点

- 实时join即join好一条记录则产出一条
- 同时支持实时，增量，全量join
- 保持原表关系，例如：假设member表修改某些字段，则系统会自动维持一致性，而无需重新build join。Mapreduce无法做到。
- Hadoop join分为：原始数据上传hdfs，hdfs下载为本地（locality），切块，map，sort，combine，传输，sort，merge，reduce，上传至hdfs，使用则还需要下载。以上过程虽有部分优化但本质上这些步骤都是串行。
- iprocess的join过程，从原始数据从客户端插入iprocess（通讯），运算，结果存储。这些都是并行。试验结果表明，join的时间就是插入到存储的时间。

iprocess-mr模型优势

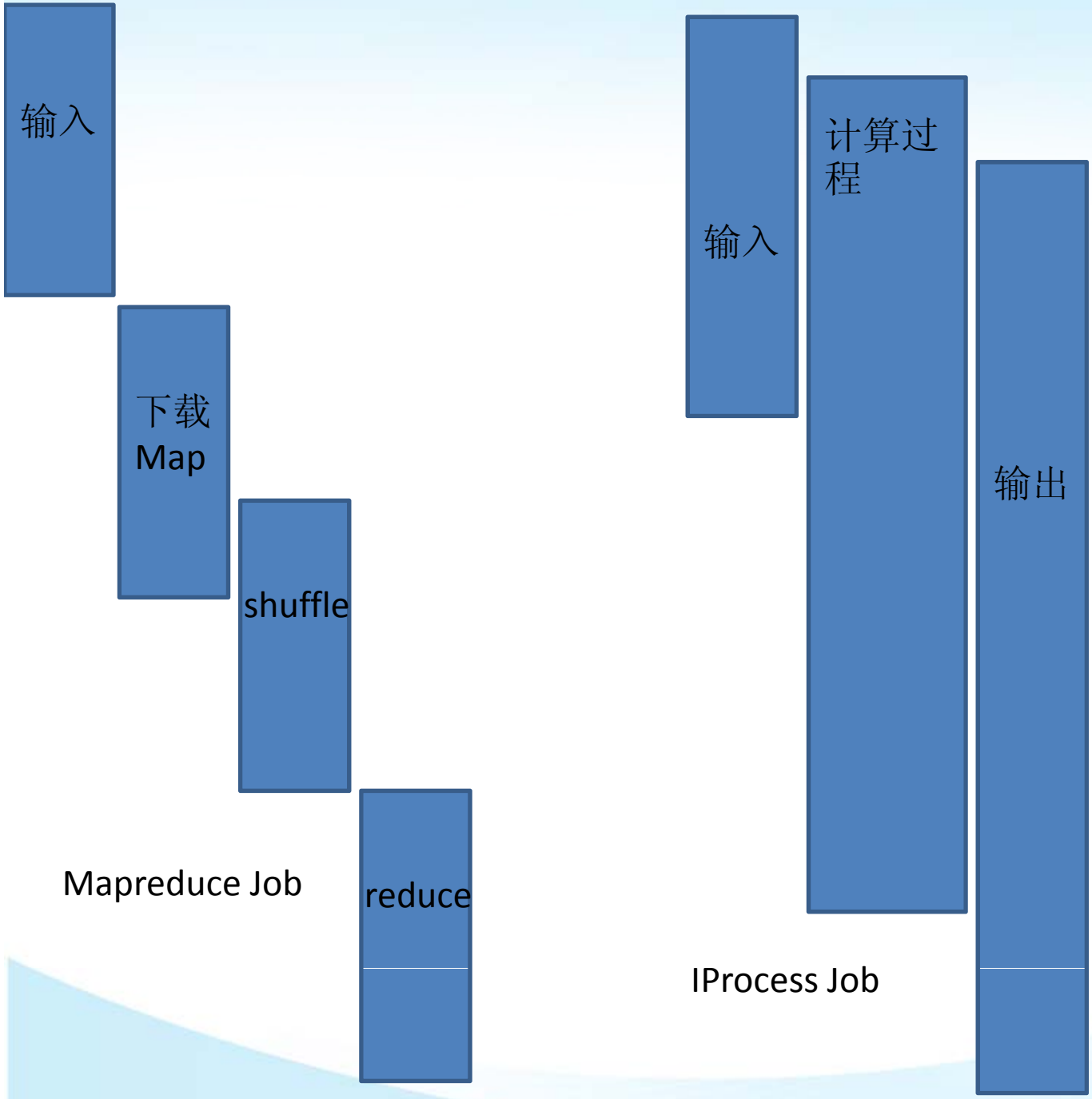
- 完全兼容MR模型
- 可以做到实时，增量，定时的MR
- 对于图计算可加速迭代过程(并且支持聚类等需要global信息的操作)
- 对于图运算无数据反复拷贝
- 对于图运算不同迭代之间可以并行运算
- 对于mr比较难受的异构数据reduce，有着天然的理论基础
- 可早停，对于某些数据挖掘算法（频繁项）有优势
- 大规模矩阵运算和机器学习（有待试验）
- 可支持并行（分支），汇聚
- 对主表，附表顺序无任何要求。克服hadoop广播方法处理不了两个大表的情况

iprocess-mr模型优势

- Reduce可尽早启动，短任务及早完成对于整个集群利用率有好处
- 高优先级任务粒度可以到segment级别。
- 这意味这更adaptive的负载策略和调度策略。
- 更容易检测和处理“straggler” 任务。
- 抢占更有效。
- 可以天然的利用columnar优势。想想dremel

iprocess-mr模型缺点

- 过于复杂
- 依赖组件多
- 任务调度更复杂
- 容错精细化到记录，所以控制更复杂
- 。 。 。



iprocess-事务

- 多版本
  - 经典的多版本事务
  - 分布式跨行跨表
  - 代价带大
  - Segment内部直接锁
  - 跨segment两阶段提交
  - tradeoff

- Lazy resolve
  - 应用场景
  - 事务冲突时不是回滚，而是
  - 调用resolve，让用户自己
  - 解决

key	bal:data	bal:lock	bal:write
Bob	6: 5: $10	6: 5:	6: data @ 5 5:
Joe	6: 5: $2	6: 5:	6: data @ 5 5:

1. Initial state: Joe's account contains $2 dollars, Bob's $10.

Bob	7:**$3** 6: 5: $10	7: **I am primary** 6: 5:	7: 6: data @ 5 5:
Joe	6: 5: $2	6: 5:	6: data @ 5 5:

2. The transfer transaction begins by locking Bob's account balance by writing the lock column. This lock is the primary for the transaction. The transaction also writes data at its start timestamp, 7.

Bob	7: $3 6: 5: $10	7: I am primary 6: 5:	7: 6: data @ 5 5:
Joe	7: **$9** 6: 5: $2	7: **primary @ Bob.bal** 6: 5:	7: 6: data @ 5 5:

3. The transaction now locks Joe's account and writes Joe's new balance (again, at the start timestamp). The lock is a secondary for the transaction and contains a reference to the primary lock (stored in row "Bob," column "bal"); in case this lock is stranded due to a crash, a transaction that wishes to clean up the lock needs the location of the primary to synchronize the cleanup.

Bob	8: 7: $3 6: 5: $10	8: 7: 6: 5:	8: **data @ 7** 7: 6: data @ 5 5:
Joe	7: $9 6: 5: $2	7: primary @ Bob.bal 6: 5:	7: 6:data @ 5 5:

4. The transaction has now reached the commit point: it erases the primary lock and replaces it with a write record at a new timestamp (called the commit timestamp): 8. The write record contains a pointer to the timestamp where the data is stored. Future readers of the column "bal" in row "Bob" will now see the value $3.

iprocess实现pagerank

```
int32_t process(Table(PageID ,double) curr ,Table(PageID ,double) next ,Table(PageID ,[ PageID ]) graph_partition )
{
        for page , outlinks in get_iterator (my_partition()) {
                rank = curr[page]
                update = PropagationFactor * rank / len(outlinks)
                for target in outlinks:
                        next.write(target , update)
        }
}

int32_t PRControl(Config conf )
{
        graph = Table(PageID ,[ PageID ]). init("/dfs/graph")
        curr = Table(PageID , double ). init(graph. numPartitions (),DefaultSumProccessor, BATCH);
        next = Table(PageID , double ). init(graph. numPartitions (),DefaultSumProccessor, BATCH);
        GroupTables (curr , next , graph);
        last_iter = 0;

        for i in range(last_iter , 50) {
                launch(PRProcess ,instances=curr_pr. numPartitions (),locality= LOC_REQUIRED (curr),args =(curr , next , graph ));
                Synchronized ();
                 swap(curr ,next);
        }
}
```

Iprocess将任务内的调度开放给开发者（如不制定，iprocess会用缺省实现的调度，本例开发者自己实现了迭代的任务调度）。Iprocess负责任务间的调度。

iprocess-主要特性

- Segment级别一致性和事务
- 任务隔离系统，MPM类似apache但是引入了关联事件。
- 引入物化视图的概念
- 搜索的宽表
- 维护原始表最终一致性
- 维护内核稳定性，插件分级

iprocess-features

v0.1 100% done

1. 支持有向图的插件系统
2. 支持实时计算，全量计算并行，支持常驻型任务
3. 支持多场景
4. 数据补偿
5. 每个场景下支持多任务
6. 支持插件隔离，优先级，进程，线程，混合模型
7. 支持trigger/触发功能
8. 支持场景，任务，目录动态创建
9. 事件池动态创建，自动迁移

iprocess-features

v0.2 80% done

1.可定制事件保序
2.支持层级目录 支持物化view
3.支持join和增量join
4.单事件触发优化
5.immune cache支持，优化access
6.memtable支持，与table一致性，写优化。兼容S4
7.分离系统级插件与应用插件接口和管理方式
8.支持定义在任意目录上的事件触发
9.增加occupy事件，支持多优先级,支持增量事件和时间触发事件
10.local preference第一次优化
11.跨行跨表事务，目录级事务定制，transaction conflict managment
12.二级索引
13.增加column多版本触发事件，支持增量mapreduce，增加merge函数。（兼容mapreduce模型，增强型mapreduce-增量reduce)

iprocess-features

v0.3 部分预研和开发

1.local preference第二次优化
2.支持迭代任务
3.目录级别一致性定制
4.引入master(chubby,paoxs)，支持任务调度，任务内关键路径调度
5.3种服务即big lock service, lightweight lock service, Oscillator service的优化和稳定性，可用性
6.插件迁移，负载均衡
7.full scan优化（无需字典序）
8.compact optimization
9.增加synchronized功能。
10.增加iteratorVersion支持
11.增加processor新的触发类型，即主动触发，查询状态。想
12.完整的监控以及任务分析
13.增加checkpoint点支持。增加用户checkpoint函数。
14.本地化第三次优化
15.sql表达式支持和优化
16.graph computing支持
17.virtual table支持
18.各层级的任务调度，即多场景调度，多任务调度，图的关键路径调度等。
19.死锁检测以及生命期维护
20.插件管理以及部署系统
21.引入虚拟机
22.引入完全的扩展接口

iprocess-features

- v0.4
  - 服务化建设
  - 完善多场景，多任务，多segment，图关键路径调度
  - 性能调优
- v0.5
  - 完善监控系统
  - 完善运维系统
  - 支持ad-hoc sql优化
  - 系统完全的流动化

iprocess应用和进展

- Clipper
  - 分布式网页处理平台。前端用户标注，后台分析

clipper

```
<result>
  <url>http://www.taobeng.com/htab/215438_845504.html</url>
  <time>2011/04/27 11:46:48 Wednesday </time>
  <root>
    <产品名称>产品名称：星三角启动控制柜/星三角水泵控制柜/星三角启动控制柜/水泵控</产品名称>
    <产品分类>产品分类：泵 -&gt; 控制柜 -&gt; 星三角启动控制柜</产品分类>
    <所在地>所 在 地：上海-上海</所在地>
    <发布公司>发布公司：上海奥力泵阀制造有限公司</发布公司>
    <联合节点1>
      <data>
        <型号>ALK-2X</型号>
        <进口口径 (mm)>18.5</进口口径 (mm)>
        <出口口径 (mm)>2</出口口径 (mm)>
        <流量 (m3/h)>星三角</流量 (m3/h)>
        <最大供气压 (MPa)>2240元</最大供气压 (MPa)>
      </data>
      <data>
        <型号>ALK-2X</型号>
        <进口口径 (mm)>22</进口口径 (mm)>
        <出口口径 (mm)>2</出口口径 (mm)>
        <流量 (m3/h)>星三角</流量 (m3/h)>
        <最大供气压 (MPa)>2415元</最大供气压 (MPa)>
      </data>
      <data>
        <型号>ALK-2X</型号>
        <进口口径 (mm)>30</进口口径 (mm)>
        <出口口径 (mm)>2</出口口径 (mm)>
        <流量 (m3/h)>星三角</流量 (m3/h)>
        <最大供气压 (MPa)>2700元</最大供气压 (MPa)>
      </data>
...
```

clipper

clipper
你好，admin 修改资料 安全退出 用户管理 Query管理 模板类型 任务列表

你所在位置：任务列表

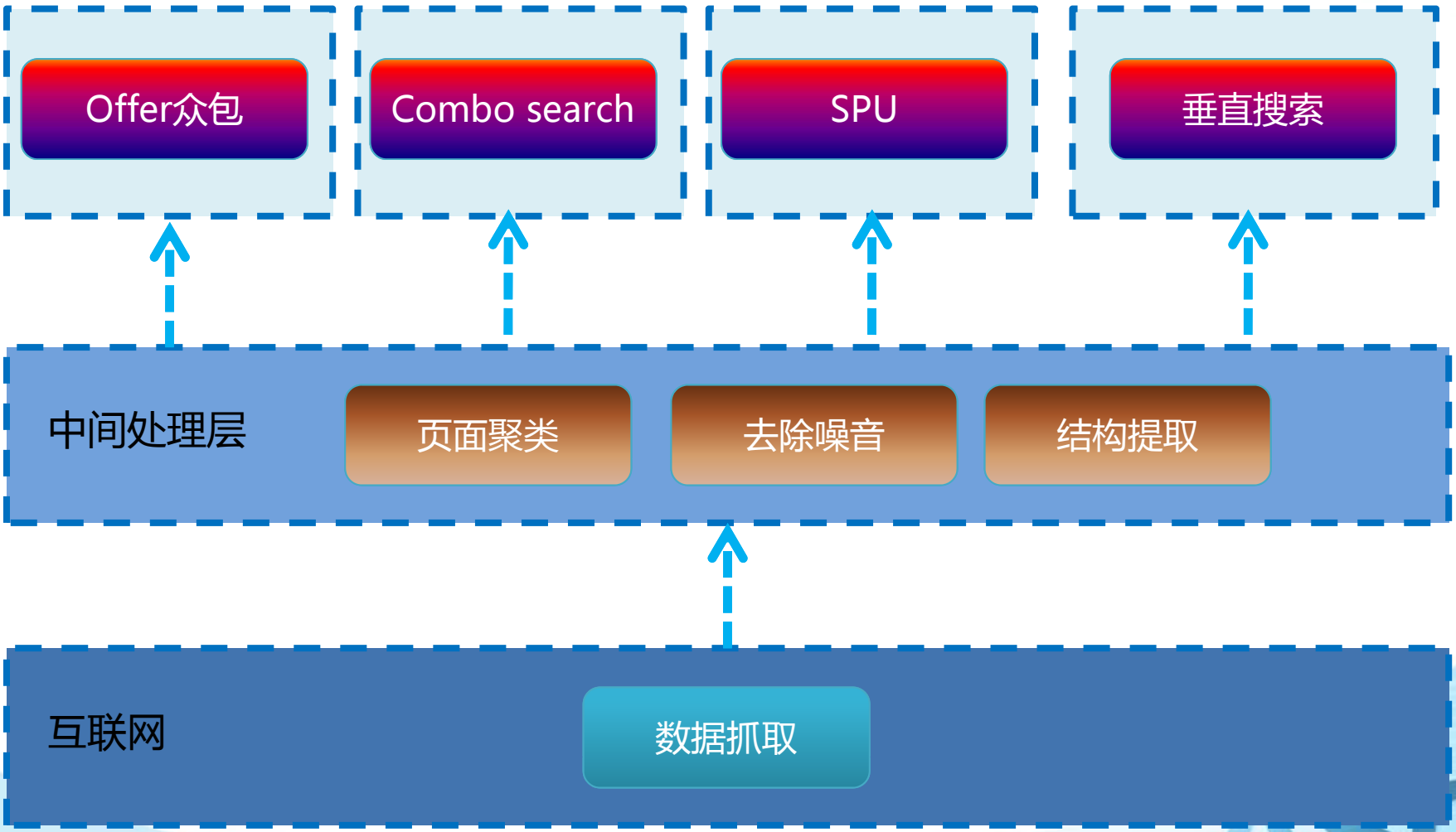
创建异步任务 创建同步任务

任务名	任务组 全部	任务类型	任务状态	创建时间	操作	查看任务	提交人
数码_mobile	数码	一次性任务	运行中	2011-04-27 10:20	编辑 启动 停止 复制 删除	查看	mazhifeng
机械-泵_2	机械	一次性任务	运行中	2011-04-27 10:01	编辑 启动 停止 复制 删除	查看	mazhifeng
zgc-数码相机		一次性任务	编辑中	2011-04-25 11:00	编辑 启动 停止 复制 删除	查看	admin
量盒	中关村	一次性任务	编辑中	2011-04-19 18:34	编辑 启动 停止 复制 删除	查看	ice.liuxq
中关村	中关村	一次性任务	运行中	2011-04-18 11:01	编辑 启动 停止 复制 删除	查看	ice.liuxq
全球IC网-品牌库_copy_copy	电子行业	一次性任务	运行中	2011-04-15 12:34	编辑 启动 停止 复制 删除	查看	nili.huangnl
info_chemnet_fxps	行业资讯信息抓取	一次性任务	运行中	2011-04-12 10:38	编辑 启动 停止 复制 删除	查看	huanqing.lihq
info_l-zzz_plastic_copy	行业资讯信息抓取	一次性任务	运行中	2011-04-12 10:34	编辑 启动 停止 复制 删除	查看	huanqing.lihq
saic_ok_copy		一次性任务	编辑中	2011-04-12 09:43	编辑 启动 停止 复制 删除	查看	admin
news_test_zongyuan		一次性任务	运行中	2011-04-11 18:30	编辑 启动 停止 复制 删除	查看	admin
全球IC网-品牌库_copy	电子行业	一次性任务	已停止	2011-04-11 16:46	编辑 启动 停止 复制 删除	查看	nili.huangnl
全球IC网-品牌库_2_copy	电子行业	一次性任务	已停止	2011-04-11 16:46	编辑 启动 停止 复制 删除	查看	nili.huangnl
机械-泵_copy_copy_copy_copy_copy_copy_copy	机械	一次性任务	运行中	2011-04-11 15:21	编辑 启动 停止 复制 删除	查看	mazhifeng
数码-图片_copy_copy	数码	一次性任务	运行中	2011-04-11 11:07	编辑 启动 停止 复制 删除	查看	mazhifeng
数码_copy_copy	数码	一次性任务	已停止	2011-04-11 11:06	编辑 启动 停止 复制 删除	查看	mazhifeng

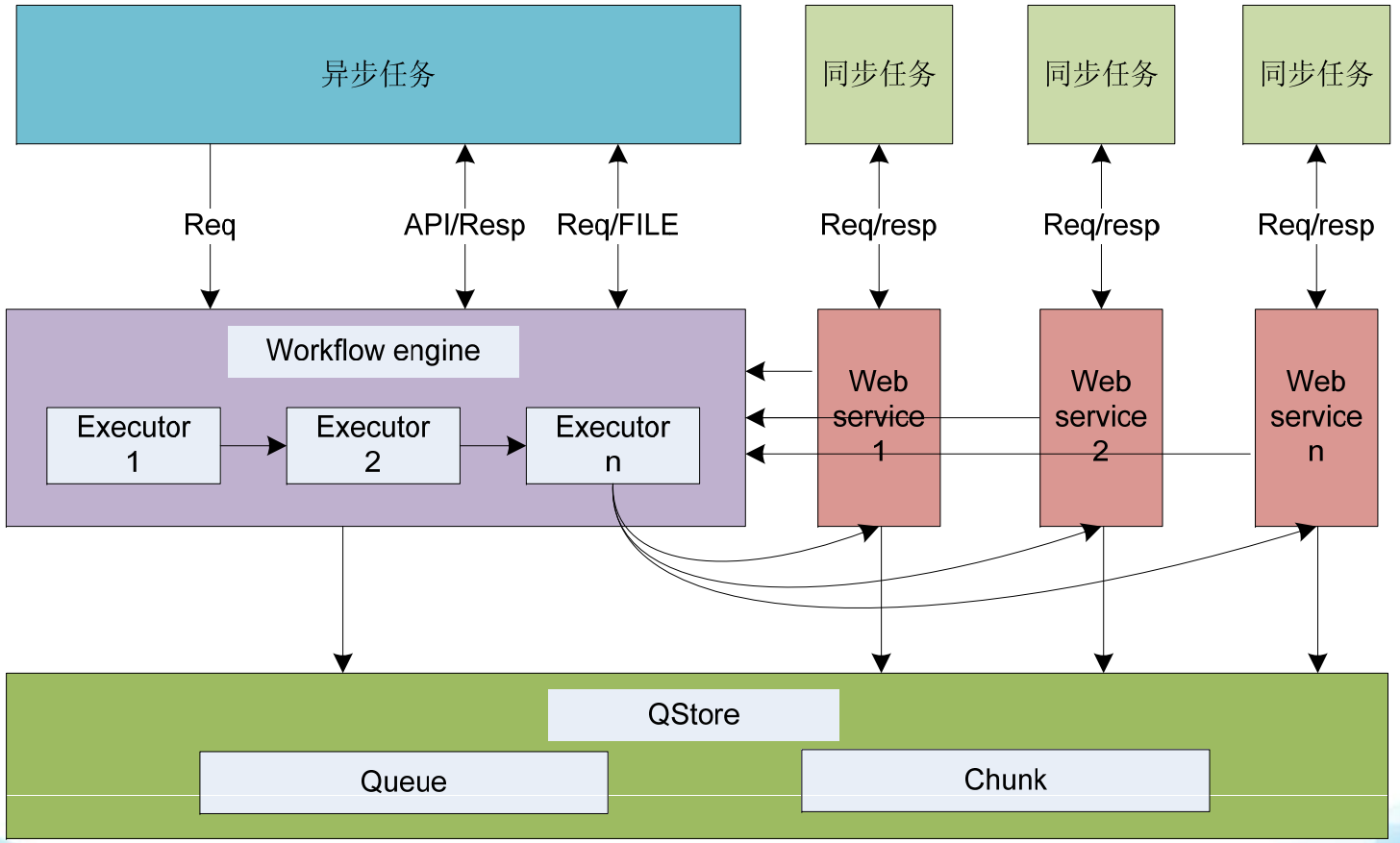
clipper


可立拍
clipper:网页信息处理平台
字段名称
字段列表
字段视图
字段列表
产品名称
产品分类
所在地
发布公司
联合节点1
型号
进口口径 (mm)
出口口径 (mm)
流量 (m3/h)
扬程 (m)
字段列表[根节点]
标注方式
框选
划选
标注内容
根节点代表着整个文档对象,不需要
保存模板
所选字段抽取信息预览(点击查看完整数据)
设为首页 | 添加到收藏夹 | 帮助 | 淘泵宣言 | 联系我们 | 提建议赚积分 | 推广服务
用户名：
密码：
登录
注册
淘泵网
www.taobeng.com
产品
选型
公司
找一下
首页
公司库
行业资讯
求购信息
代理合作
技术转让
展会
招标
标准
文章
人才中心
检测中心
科研院校
您所在的位置：首页 -> 泵 -> 隔膜泵 -> 气动隔膜泵 -> 参数报价表
沈阳三维QBY系列气动隔膜泵/耐腐蚀隔膜泵
网络营销 首选淘泵网!
产品名称：沈阳三维QBY系列气动隔膜泵/耐腐蚀隔膜泵
产品分类：泵 -> 隔膜泵 -> 气动隔膜泵
询价留言
更新时间：2010-8-12 11:59:53
所在地：辽宁-沈阳
发布公司：沈阳世运河机械设备有限公司
查看更多 "沈阳三维QBY系列气动隔膜泵/耐腐蚀隔膜泵"
沈阳世运河机械设备有限公司
进入旺铺
VIP
联系人：关福海
手 机：13898100551
联系电话：024-82711065
传 真：024-89837066
电子邮箱：zhaolivino1975@163.com
公司网址：HTTP://WWW.SYSYH.COM
沈阳三维W型往复式
沈阳三维WY型往复式
沈阳三维SZ系列水环
沈阳三维SK系列水环
沈阳三维XD系列单级
沈阳三维SZB系列水
XZQB系列水泵自动控

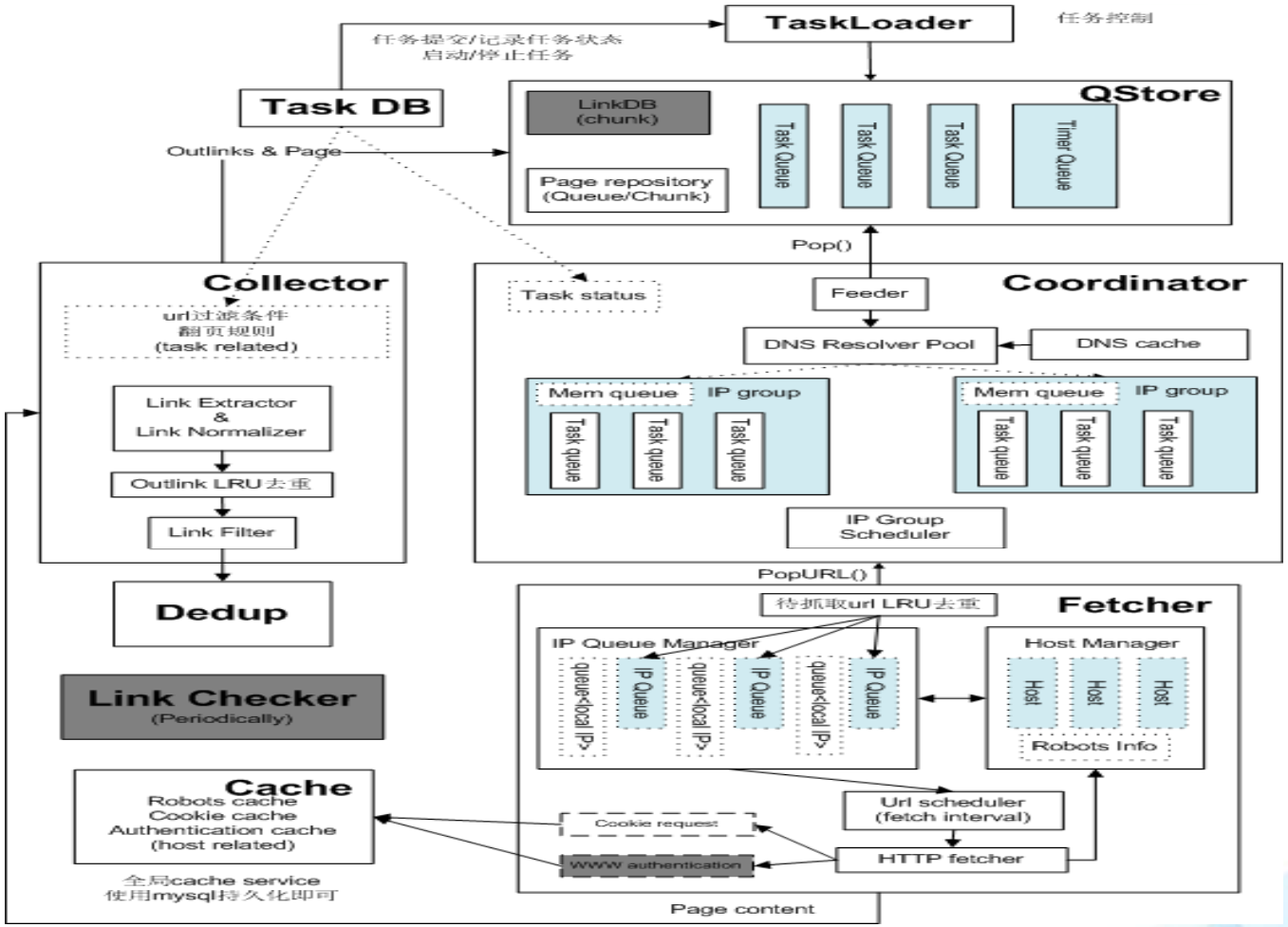
clipper



clipper



parker


TaskLoader
任务控制
任务提交/记录任务状态
启动/停止任务
Task DB
Outlinks & Page
QStore
LinkDB
(chunk)
Page repository
(Queue/Chunk)
Task Queue
Task Queue
Task Queue
Timer Queue
Pop()
Collector
url过滤条件
翻页规则
(task related)
Link Extractor
&
Link Normalizer
Outlink LRU去重
Link Filter
Dedup
Task status
Feeder
Coordinator
DNS Resolver Pool
DNS cache
Mem queue
IP group
Task queue
Task queue
Task queue
Mem queue
IP group
Task queue
Task queue
Task queue
IP Group
Scheduler
PopURL()
待抓取url LRU去重
Fetcher
IP Queue Manager
queue<local IP>
IP Queue
queue<local IP>
IP Queue
queue<local IP>
IP Queue
Host Manager
Host
Host
Host
Robots Info
Link Checker
(Periodically)
Cache
Robots cache
Cookie cache
Authentication cache
(host related)
全局cache service
使用mysql持久化即可
Cookie request
WWW authentication
Url scheduler
(fetch interval)
HTTP fetcher
Page content

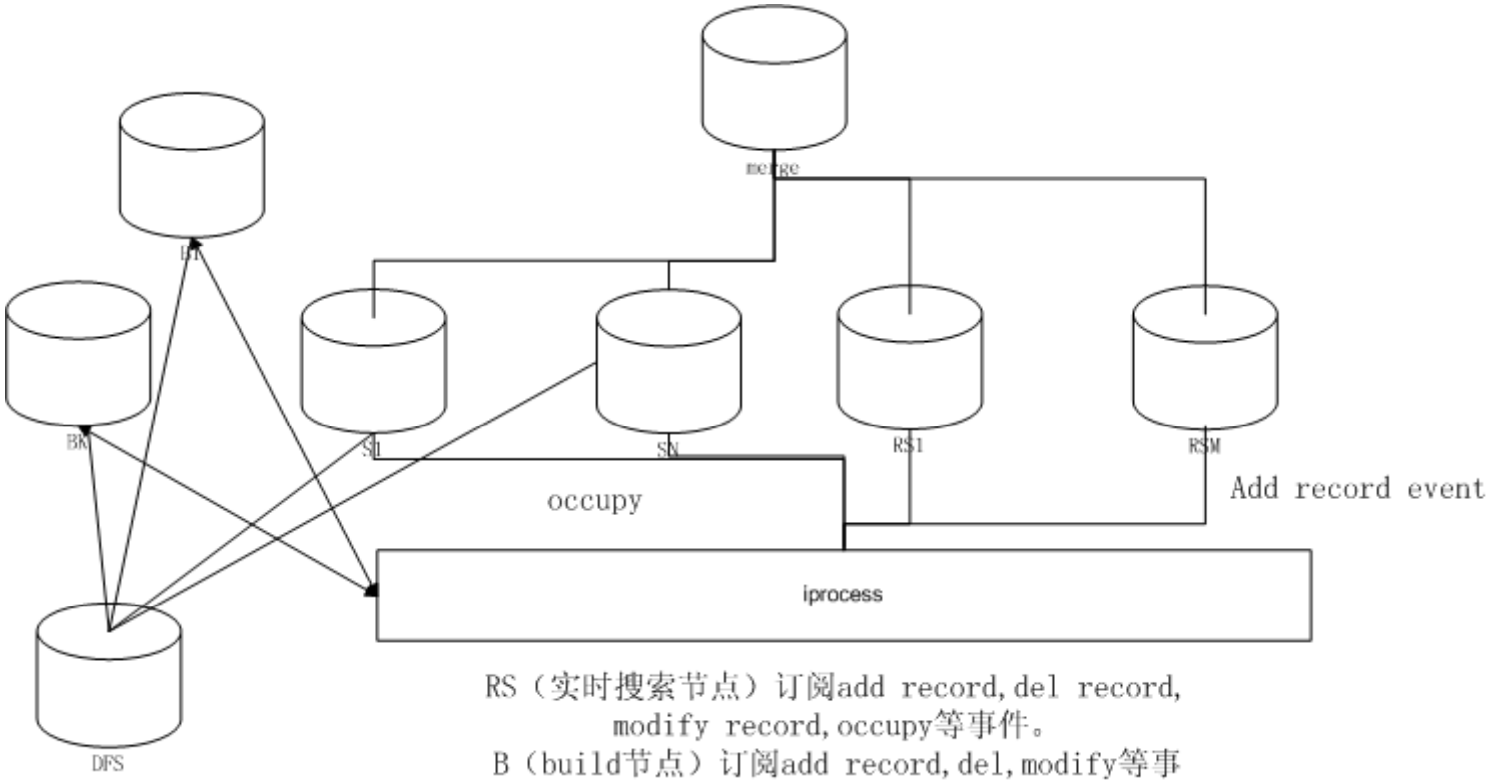
iprocess应用和进展

- 搜索数据中心
  - 搜索需要宽表
  - Db， dump，join，很多预处理，切分，准备build索引
  - 实时，增量，全量不同逻辑，不能并行
  - 晚上很辛苦

  - 解决并行问题，实时，增量，全量统一逻辑。
  - 对于小修改大变化自动响应
  - 删除自动响应
  - 运营更加方便

iprocess应用和进展

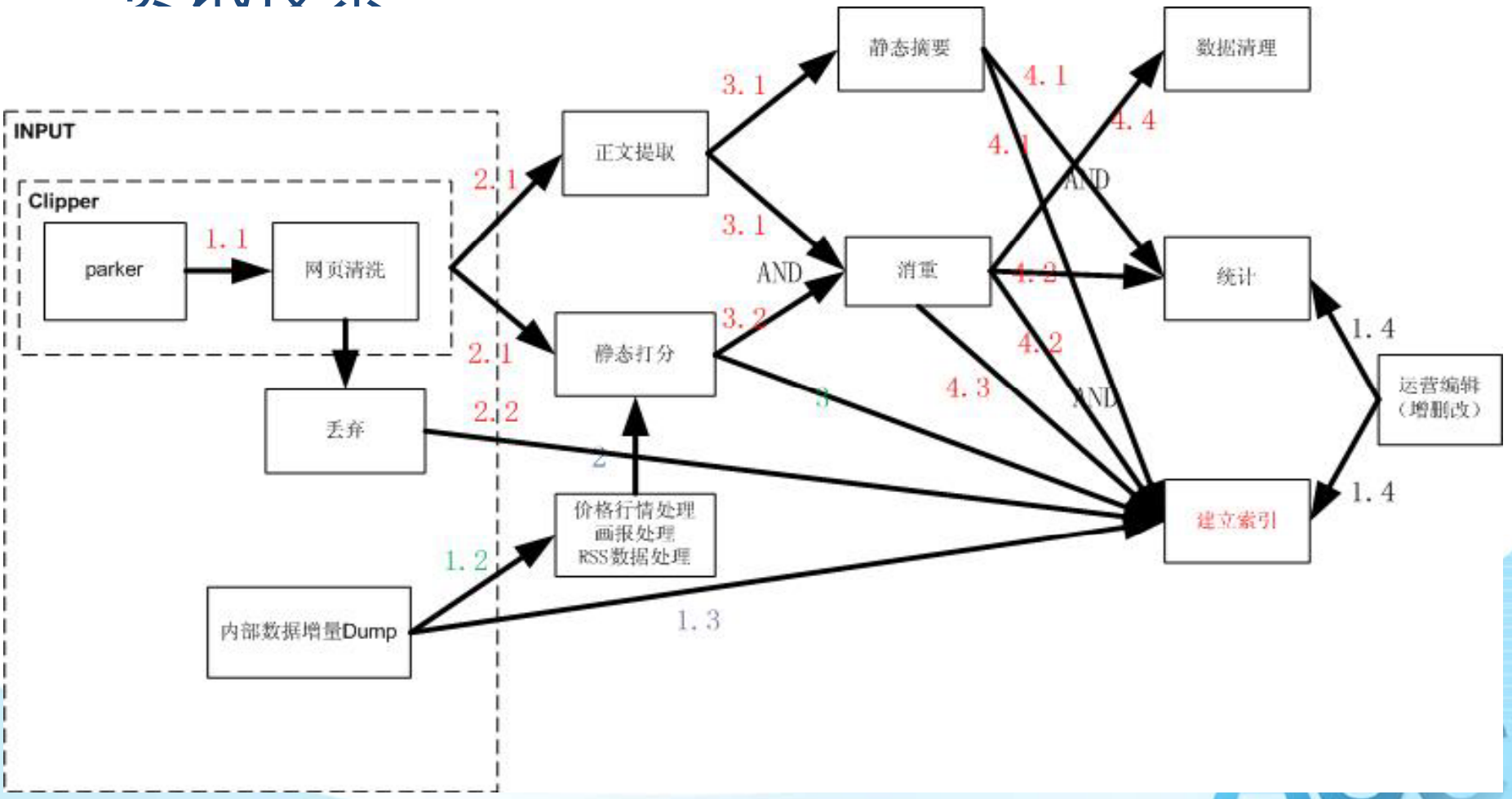
- isearch新架构



RS（实时搜索节点）订阅add record,del record, modify record,occupy等事件。
B（build节点）订阅add record,del,modify等事件
S（搜索节点）订阅add segment，del record, modify，发出occupy等事件

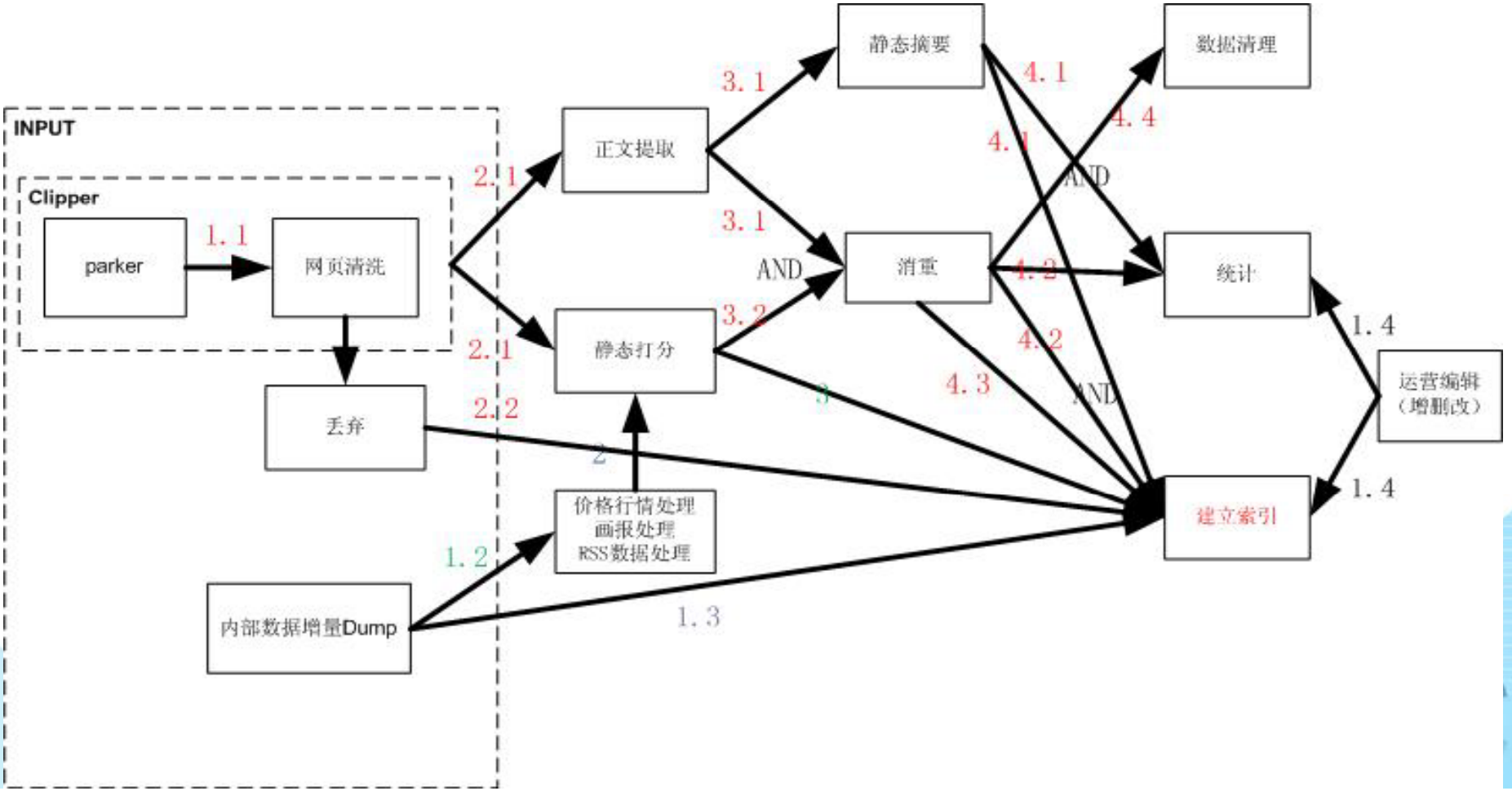
iprocess应用和进展

- 资讯搜索

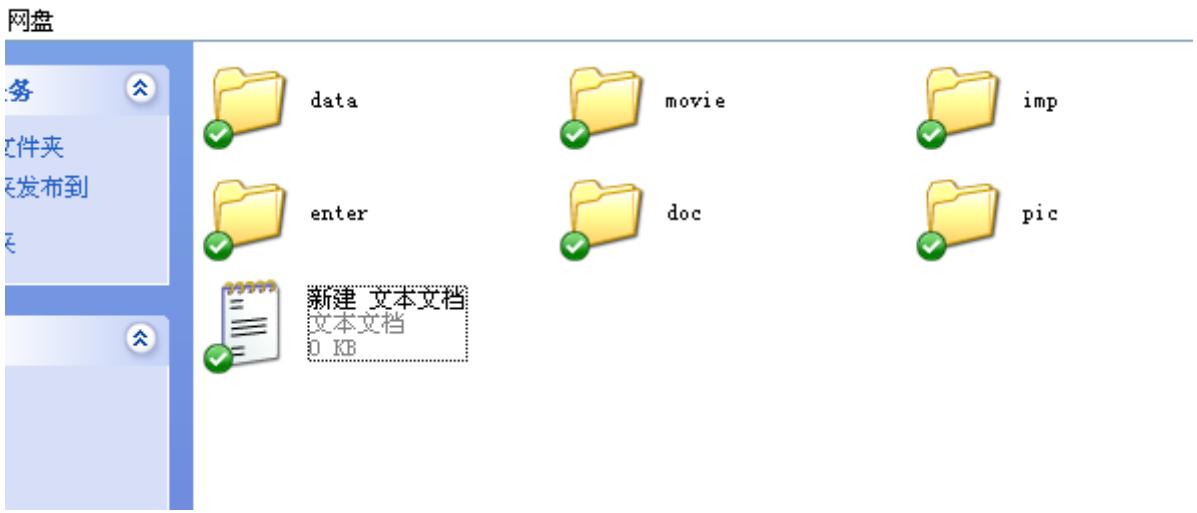


实际例子

- 资讯搜索


INPUT
Clipper
parker
网页清洗
丢弃
正文提取
静态打分
价格行情处理
画报处理
RSS数据处理
内部数据增量Dump
静态摘要
消重
数据清理
统计
运营编辑
（增删改）
建立索引
1.1
1.2
1.3
1.4
2.1
2.2
3.1
3.2
4.1
4.2
4.3
4.4
AND
2
3

Iprocess未来应用--网盘



iprocess未来规划

- 高性能，高响应时间，高容错，动态扩展的实时计算平台
- 服务化
- 流动计算
- 虚拟化
- 易用性提升
- 推广：（实时）搜索，计算广告，数据挖掘，机器学习，图计算，任何离线实时计算。。。

iprocess

- 刚刚走出第一步
- 期待大家能提出批评，建议
- 共同建设
- 0.3
- 金晓军，潘岳

Thanks!

Q&A