Gearman

博升优势 徐长龙

xucl@Bsatinfo.com

此PPT内图片均来自互联网

此PPT介绍内容如下

- 举个现实中的例子(改了栗子俩字)
- Gearman最初用途
- 功能特性
- 性能特征
- 使用方法
- 部署
- 系统架构带来的变革

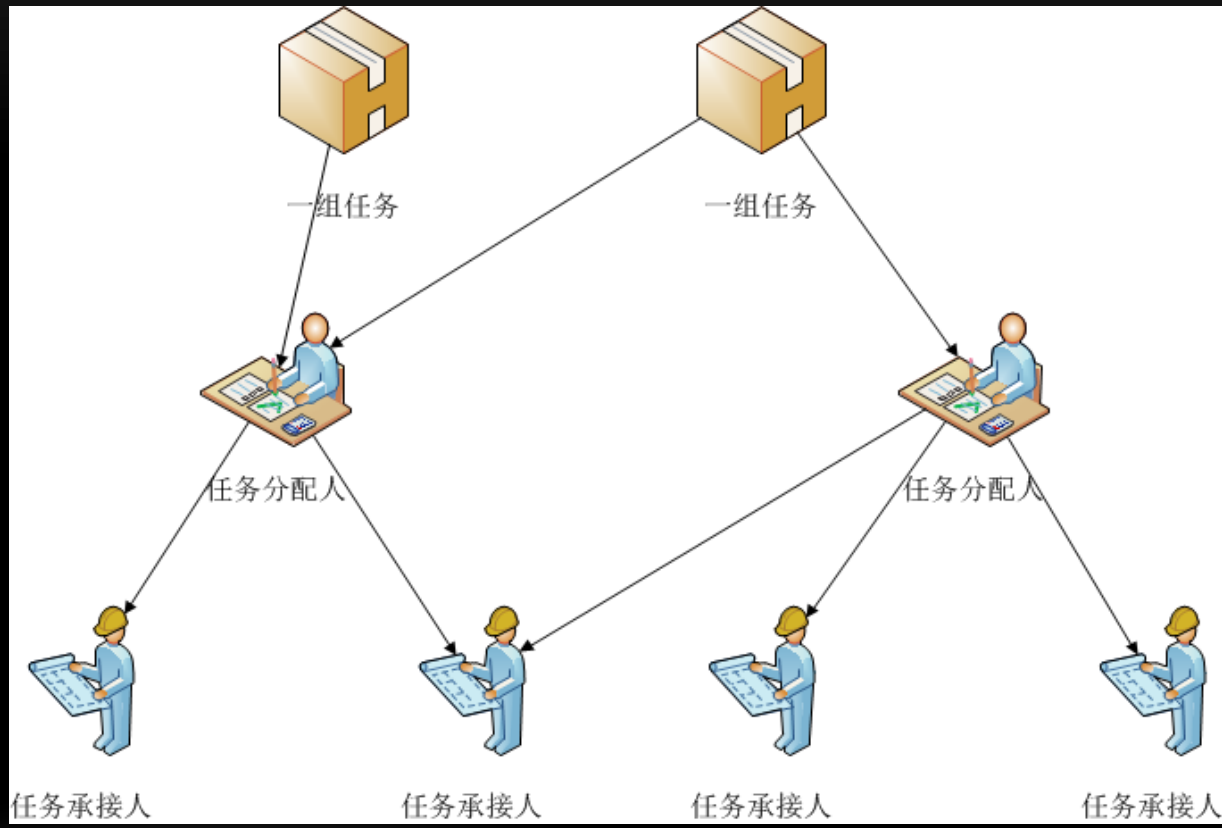
使用Gearman的技术公司

- Yahoo
- SAE
- Digg
- 金山
- 百度
- 新浪

引言-现实中的例子

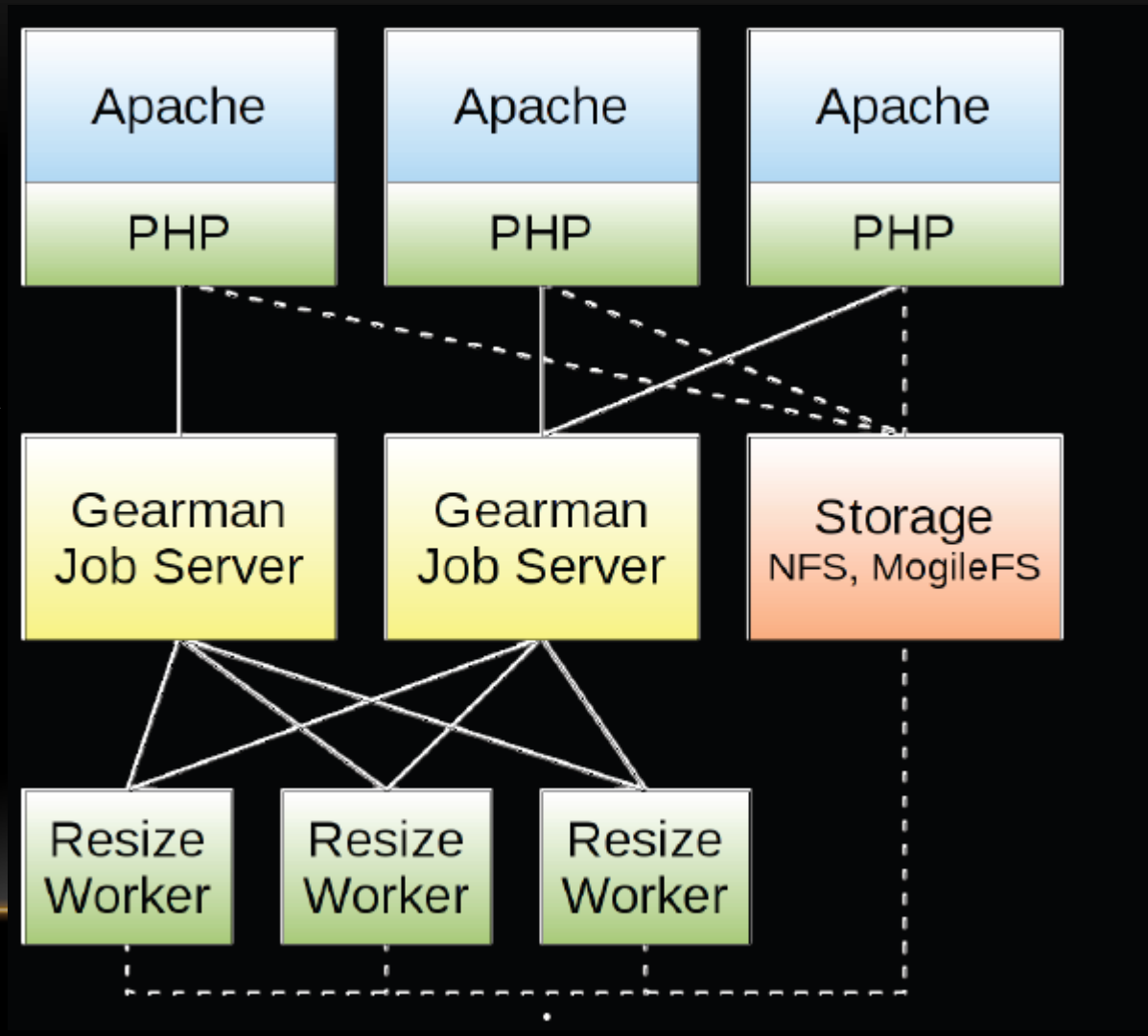
- 当任务很多时需要分配
- 分配规则：
  - 1.承接人不是很忙
  - 2.任务在工作范围内
  - 3.在位置上（非离线
- 完成任务报告结果
- 或完成任务无需回馈

- Gearman主要做了上面的三件事，分配任务，承接任务，抓取任务



Gearman的起家

- 右图为刚开始时用途
- 原用于图片的resize
- 图片的resize很耗时
- 遂做成同步/异步服务
- 由PHP把原始文件保存
- 由PHP分配任务给Job服务
- 由job分发任务给worker
- Work接受到任务并工作
- 完成任务报告



功能特性

- 多方式：支持PHP，Java，Python，数据库，及http通讯
- 持久化：异步任务支持扩展持久化
- 多线程：并发处理请求
- 服务之间通讯使用TCP，比HTTP请求节省系统资源
- 接入及部署简单，接入简单，开发简单
- 负载均衡，自动根据工作量进行均衡
- 稳定：个别服务器死机可自动使用其他冗余节点
- 可代理转发任务到其他Job服务器
- 下发任务可以通过优先级决定执行顺序

性能

- 以下测试数值为虚拟机
- 常规HTTP的Curl：这个受端口最大值及打开文件句柄数限制
  - 当使用HTTP Curl时
    - 同步：2000/sec

- Gearman：未优化情况下
  - 用PHP+gearman插件测试并发批量调用：
    - 同步：4000/sec
    - 异步：10000/sec
  - 以上还有上升空间，官方测试数据为5w/sec
  - 增加持久化插件设置后性能会下降一些

单机实测性能

测试项	环境	平均请求/秒
HTTP压力AB	**AB->Nginx**	**100并发**
	单个html页面http请求	10000 req/sec
	单个PHP页面http请求	2844 req/sec
	上面测试占用大量端口钱	
PHP命令行	**PHP Cli->Gearmand->PHP Worker->随机数**	**单线程**
- 异步	单个50字节消息	11111 req/sec
- 同步	单个50字节消息	4545 req/sec
- 多任务	取决于单个任务执行最大时间	
	上面测试不会占用大量端口，速度很快，load 1.5	
综合测	**ab->Nginx->PHP-fpm->Gearmand->PHP Worker**	**100并发**
	异步	1885 req/sec
	同步	1581 req/sec
	占用端口，load2.7	

GEARMAN使用方法

Work及Client工作角色及职责

- Client
  - 同步调用：调用后阻塞等待结果返回
  - 异步调用：调用后即返回成功，job服务器会根据负载情况慢慢分配。
  - 分优先级调用：调用时可指定当前任务优先级
  - 任务组合调用：支持Task

- Worker
  - 注册处理任务的标识别及回调函数名称
  - 循环等待任务下发（无任务则会阻塞）
  - 处理完毕后根据回调函数返回内容返给Job转发给调用方
  - 报告自己负载情况供Job参考是否继续发放任务
  - 每个Work同时只能处理一个任务（缺点/优点）

- Job:分配任务，持久化保存未完成任务，命令行时执行可做client及work

Linux命令行方式

- Client：发布任务，可接受转向符输入
  - ls | gearman –f abc 命令行管道符发布任务
  - gearman –f abc < xxxxfile 文件发布批量任务
  - gearman –f abc “parameter” 发布单个任务
  - tail –f xxx.log|gearman –n –f loger –h 192.168.0.1 可用于log合并

- Worker：接受任务并执行对应命令或可执行文件
  - gearman –w –f abc -- wc -l 当有abc这个任务时就执行wc –l
  - gearman –w –f abc :/xxxx.sh 当有abc这个任务时就执行xxxx.sh

Mysql插件Client

- mysql下发任务
  - 执行SQL：Select gman_do(‘abc’,”parameter” as result);
  - 有兴趣大家可以研究下，把他写在存储过程内
  - 这里只是简略介绍他的存在……

PHP Gearman Client Job

- Client.php:发布任务
  - $client= new GearmanClient();
  - $client->addServer("127.0.0.1", 4730);
  - $client->addServer(“192.168.0.12”, 4730);//可多个job服务器，任务工作时才连接
  - print $client->doNormal(“abc”, “参数");

  - do后面的描述代表此任务的执行优先级，此为同步阻塞调用
    - doNomal:正常优先级
    - doHigh:最优先执行
    - doLow:最低优先

  - doBackground异步只是派发任务，不等待返回结果
  - 可通过client查询任务执行情况

PHP Gearman Work Job Blocking

- Work.php:主动去取任务，如果没有自动休眠
  - $worker= new GearmanWorker();
  - $worker->addServer(“127.0.0.1”, 4730);//连接job服务器
  - $worker->addFunction(“abc”, “abc对应函数”);//注册支持任务及对应函数
  - while ($worker->work());//循环等待任务，没有时阻塞，循环体内可以放错误处理
  - //work内尽量不要出现资源忘记回收情况
  - //处理任务的回调函数
  - function title_function($job) {
    - $str = $job->workload();//过来的参数}//被调用的函数
    - Return 返回给调用方的数据、
  - }

Non-blocking Gearman Work Job非阻塞方式

- ```
  $worker= new GearmanWorker();
  $worker->addOptions(GEARMAN_WORKER_NON_BLOCKING);  # Make the worker non-blocking
  $worker->addServer(); $worker->addFunction('reverse', 'reverse_fn');
  # Try to grab a job
  while (@$worker->work() ||
         $worker->returnCode() == GEARMAN_IO_WAIT ||
         $worker->returnCode() == GEARMAN_NO_JOBS)
  {
    if ($worker->returnCode() == GEARMAN_SUCCESS)
      continue;

    if (!@$worker->wait())
    {
      if ($worker->returnCode() == GEARMAN_NO_ACTIVE_FDS)
      {
        # We are not connected to any servers, so wait a bit before
        # trying to reconnect.
        sleep(5);
        continue;
      }
      break;
    }
  }
  ```
  ```

Gearman PHP Client Task

- 一次可以调用多个task并行执行,任务会分配到多个work一起执行
- 传结果回来可以使用complete事件获取

```
$client= new GearmanClient();
$client->addServer();

$client->addTask("reverse", "Hello World!");
$client->addTaskHigh("reverse", "Hello High!");
$client->addTaskLow("reverse", "Hello Low!");

$client->setCompleteCallback("complete");
$client->runTasks();

function complete($task)
{
  print $task->data() . "\n";
}
```

Task 与 Job 区别

- Task是一组job，在下发后会并行执行并返回结果给调用方
- Task内的子任务会下发给多个Work并行执行

- Do和dobackground这类函数下发的任务为job，每个任务只会在一个work上执行

- **国内翻译有很多地方对task和job混淆，导致貌似 task=job 特此注意**

同步调用碰到的缺点

- 每启动一个Work，Job服务器会自动创建一个Pid文件，这意味着他会占用文件打开句柄数
- 当Client个数超过Worker个数的时候会出现排队现象，排队时会加长处理时间
  - 弥补方式，启动多个Woker进程，如并发要求300那么启动150个Work
    - 造成后果，Worker空闲会导致闲置占用资源
      - 补救方式，–worker-wakeup 参数，在没有任务时Client休眠，当有任务Job服务器根据需要主动推送唤醒命令
        - 突然发现个工具GearmanManager，可以搞定上面事情~只要简单的使用即可
- 传递的参数必须序列化
- Work占用一定内存和cpu，打开过多会占用很多资源非服务群使用效果不明显

异步调用的缺点

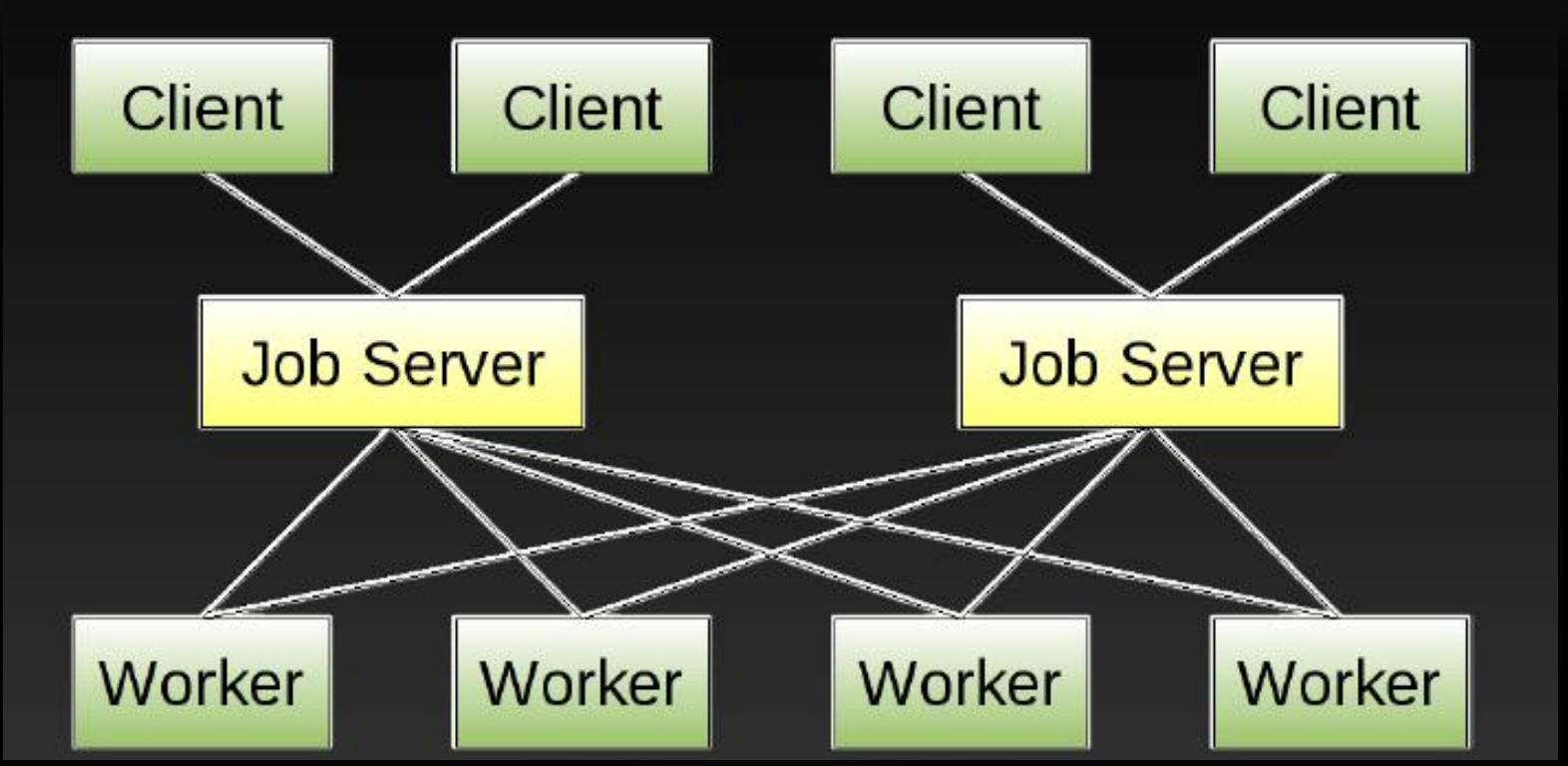
- 多个work处理log记录时，容易出现个别log乱序
- Work启动太少且工作时间太长会导致任务堆积，Job服务器占内存过多
- Work内调用的工作函数错误无法处理或通知，只能通过log查看结果
- 如果worker异常，没有接任务的worker很难发现，只能观察Job的持久服务器内的数据量

Gearman Manager 简单介绍

- basename(__FILE__)." -h | -c CONFIG [-v] [-l LOG_FILE] [-d] [-v] [-a] [-P PID_FILE]
  - -a 定期检测指定目录下是否有新的注册函数
  - -c 指定Worker的配置文件
  - -d 以daemon方式后台运行
  - -D启动时整体启动多少个线程加载work
  - -h Job服务器ip
  - -l log文件保存地址，如果指定syslog可以输出到syslog上
  - -p 给job的function名称前加前缀，只适合pecl方式，pear只能写在代码内
  - -P pid文件地址
  - -u 运行时的用户身份
  - -w 引用函数目录，如果是pecl启动的可以指定多个用逗号隔开
  - -r 每个Worker启动线程个数，越多并发越高，但是吃资源
  - -x work的一个工作超时时间，超过了会关闭
  - -Z 配置文件测试，会输出解析出来的配置结果

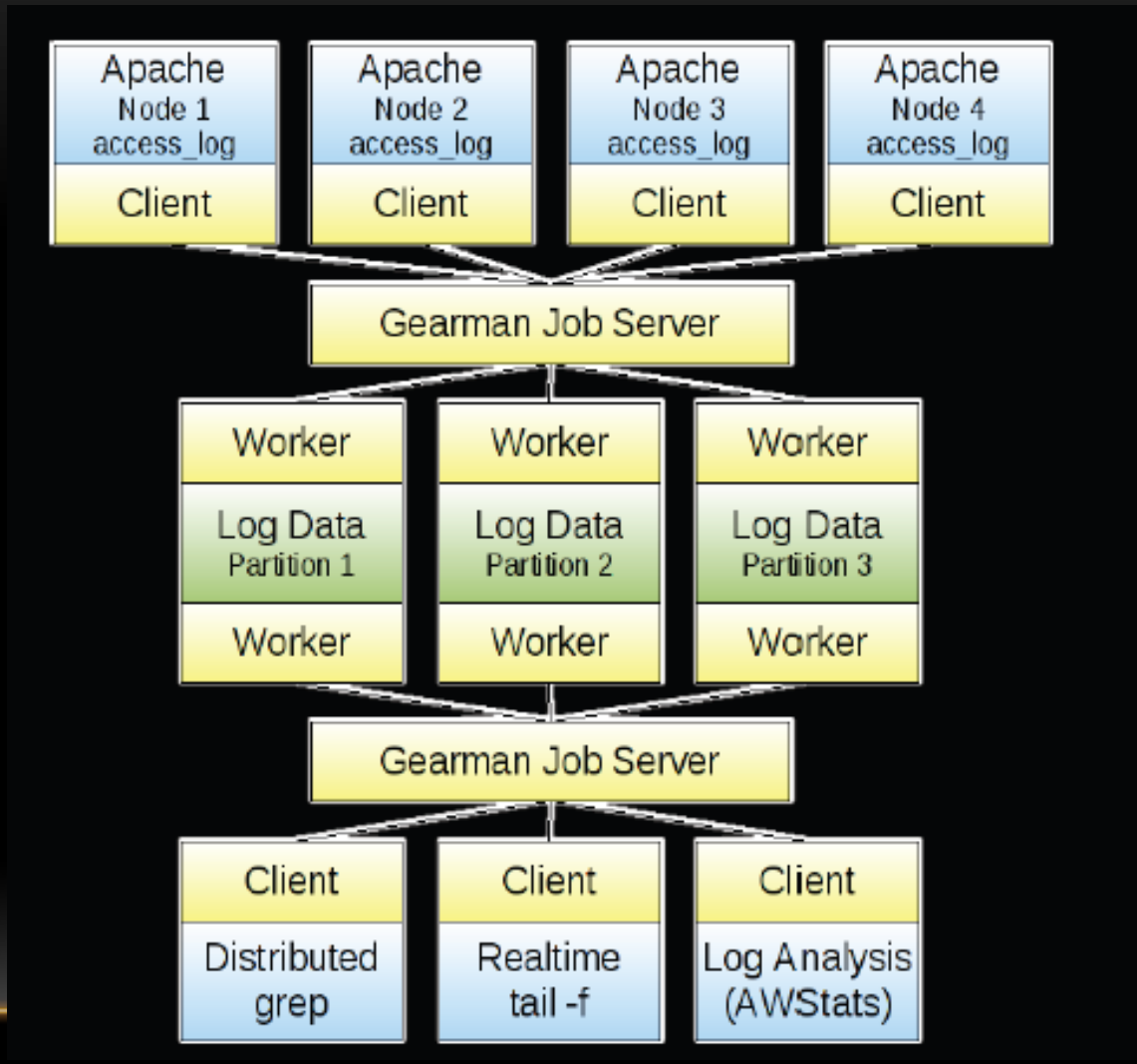
部署方式

部署结构


Client
Client
Client
Client
Job Server
Job Server
Worker
Worker
Worker
Worker

扩展部署结构Map/Reducer

- Woker可以做数据转发任务
- 可以部署瀑布结构分担压力
- 右图为log分发案例在后介绍



应用场景举例

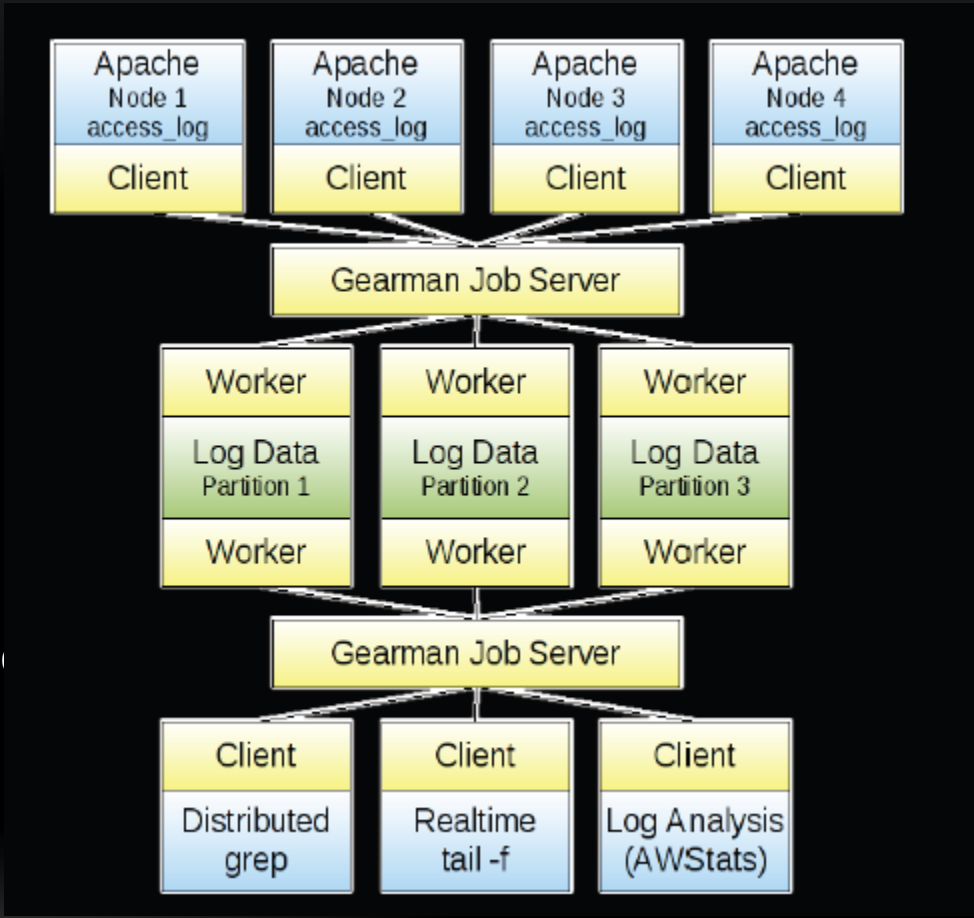
应用场景

- 需要异步操作的缓慢操作：文件生成，图片切割，视频缩略图
- 需要减轻单步操作在单台服务器上的操作：如服务组与服务组之间通讯
- 不同开发语言之间交互调用：php调用java，java调用php
- 同步分步调用：顺序调用函数就像服务在本地一样
- 合并多个系统的log或者数据
- 减轻http请求量节省服务器端口资源：tcp二进制比http纯文本通讯快
- 做驾驶员操作战斗机的事情：一个发起者操作多台机器做一些事情，如查询

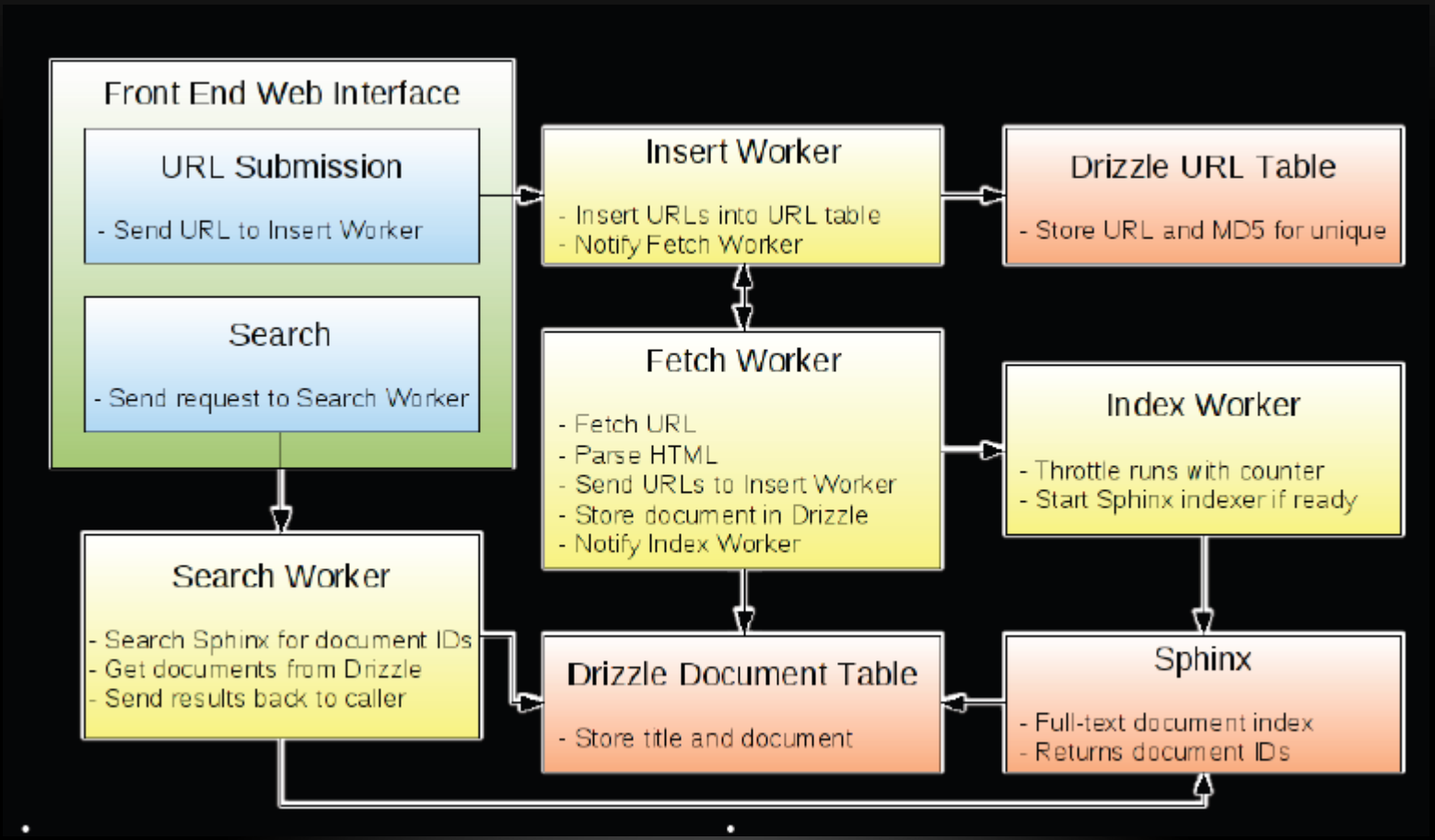
Log记录

- Log文件服务器
- Client发log文件内容
- Job1获取所有任务合并转发给存储服务器
- 存储服务器的Job把任务拆分开并发写入

- Client还可以用于查log
- 如Gearman –h host –f fun1 –f fun2 param
- 并且可以用多个服务器做数据筛选
  如a服务器筛选id1~100000的数据
  b服务器筛选id100000~20000的数据

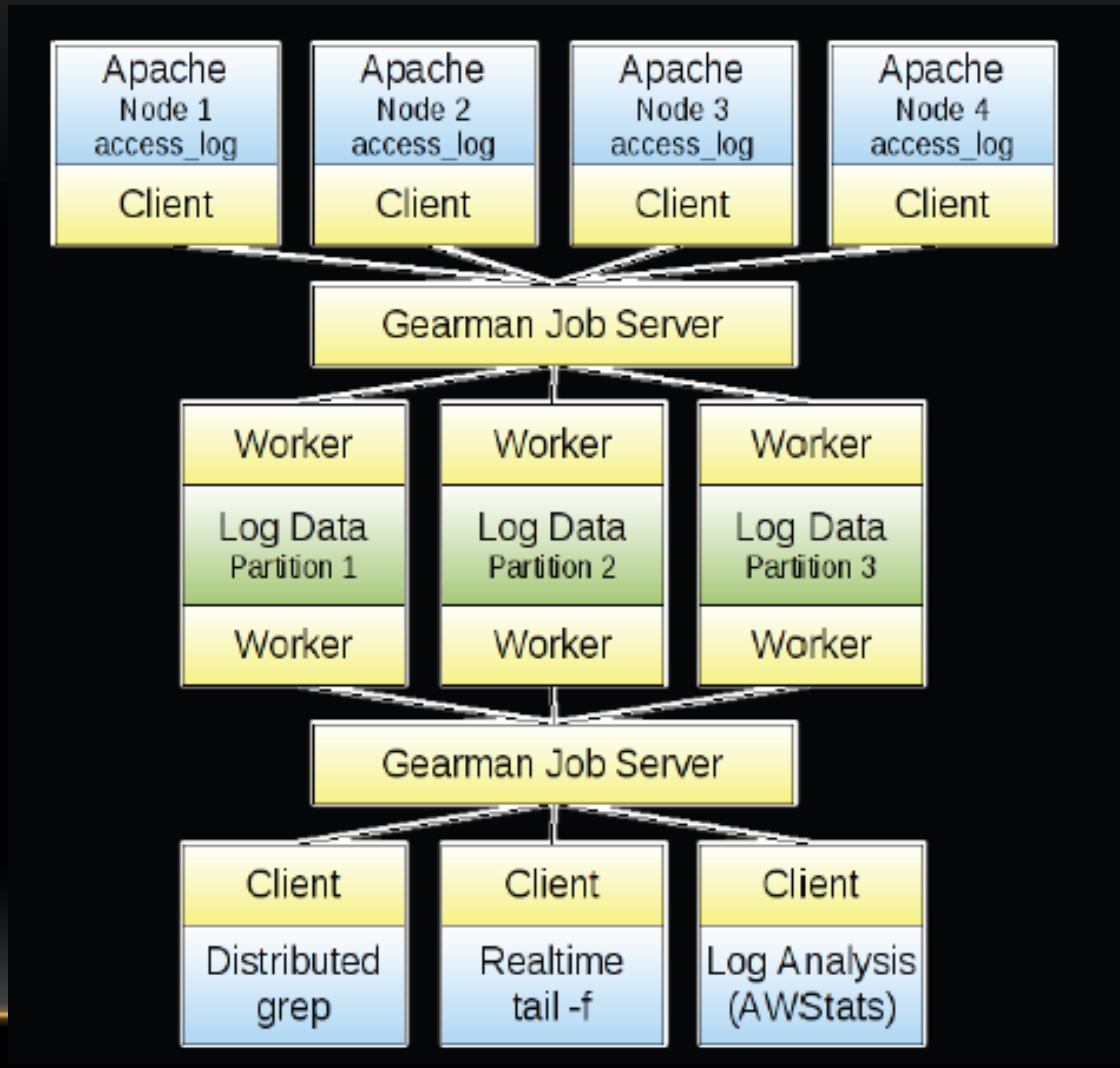


搜索入库


Front End Web Interface
URL Submission
- Send URL to Insert Worker
Search
- Send request to Search Worker
Insert Worker
- Insert URLs into URL table
- Notify Fetch Worker
Drizzle URL Table
- Store URL and MD5 for unique
Fetch Worker
- Fetch URL
- Parse HTML
- Send URLs to Insert Worker
- Store document in Drizzle
- Notify Index Worker
Index Worker
- Throttle runs with counter
- Start Sphinx indexer if ready
Search Worker
- Search Sphinx for document IDs
- Get documents from Drizzle
- Send results back to caller
Drizzle Document Table
- Store title and document
Sphinx
- Full-text document index
- Returns document IDs

消息分发机制

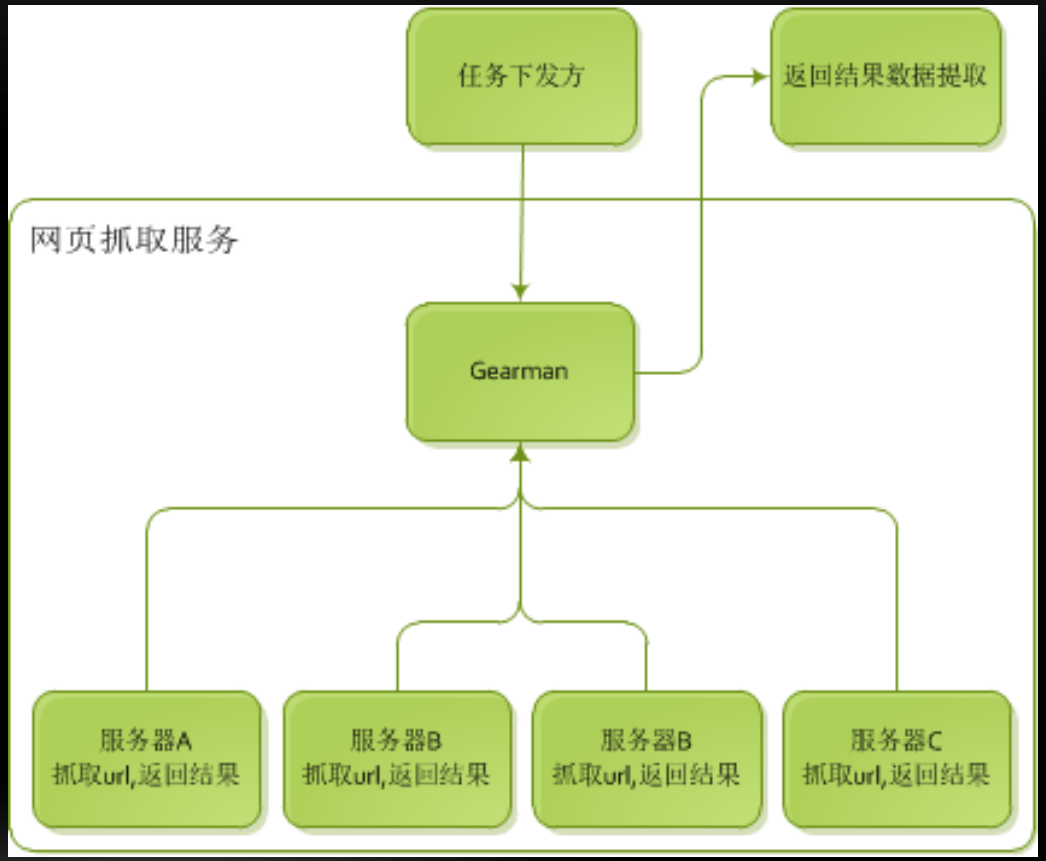
- 右图为引用log记录图
- 中间的work根据订阅关系分发消息即可
- 一个消息分发广播系统完成
- 发布消息使用异步发放若要求很高可以同步



数据抓取

- 抓取网页耗时在http请求上
- 此功能块功能单一
- 需要大量服务器
- 获取到url返回html即可

- 好处
  - 加大并发获取
  - 可多ip抓取
  - 抓取服务不用业务相关代码
  - 数据提取只要考虑提取



其他使用技巧

- 通过一个PHP批量触发多台服务器周期Crontab工作
- 监视服务器存储空间
- 远程控制服务器做常规操作，如在发布服务器发送命令删除某文件夹数据，更新git代码。
- 启发思路：让client作为业务中心点，指挥work工作
- 邮件发送,短信发送
- 视频处理
- 异步log
- 消息分发
- 异步队列

变革

Gearman带来的思考

- 不同语言及开发环境之间的隔阂变小
- 很多事情可以快速通过拼装通用的服务模块做成产品
- 一次http请求，可以用做一件事情的时间处理多个数据
- 一次http请求结果可以从多个系统并发获取数据后，到PHP拼装结果
- fastcgi_finish_request();可以让php把当前所有结果先返回给请求方，后面的事情由php自己处理，但是这个会占用php进程，而异步的调用采用gearman可以降低php外接服务器的压力
- 基础于此我们可以把产品做得松散，通过业务需要进行拼装
- 单个服务节点损坏，服务将不再受影响。

参考资料

- http://gearman.org/presentations 相关扩展使用方法资料
- http://gearman.org/documentation 官方文档
- http://php.net/manual/zh/book.gearman.php php manual扩展插件
- http://blog.csdn.net/linvo/article/details/7721625 gearman优化
- http://blog.chinaunix.net/uid-20357359-id-1963682.html gearman杂谈
- https://github.com/brianlmoon/GearmanManager gearmanager代码

- http://code.google.com/p/shard-query/wiki/HowToUseShardQuery mysql扩展查询

THANKS