# Honeycombx数据同步系统

### 技术栈：

* 语言：Java，Scala
* 其它：akka，jetty，curator，bootstrap

### 布署与启动

* 目录说明：

$HONEY\_HOME/conf下为系统配置

$HONEY\_HOME/conf/supervisor-conf下为监控配置

$HONEY\_HOME/lib为依赖包

$HONEY\_HOME/bin下为命令脚本

$HONEY\_HOME/web下为相关ui展现页面

同时在配置中需要指定history与log目录

* 命令说明：

start-supervisor.sh启动监控

stop-supervisor.sh 结束监控

deamons.sh start Queen; deamons.sh start Bee :Queen与Bee的启动

deamons.sh stop Queen; deamons.sh stop Bee :Queen与Bee的停止

honey 作业管理脚本，详细查看honey -h

### 客户端

两种方式提交作业：

* API方式

**object** HoneyClient {

**def** submitJobToQueen(job: SubmitJob): String

**def** killJob(kill:KillJob):String

}

* Cli方式

./honey

Usage: honey -submit [options]

Usage: honey -kill [job ID]

Options:

--output job-output-path

--number job-task-number

--conf job-conf-path

--name job-name

--help, -h

### 作业

* Job

一次同步任务的描述

* Task

Job在提交后会被分成多个task，类似MR 体系中的Map

* Attempt

每个task被分到Bee上的一次执行，当这个Bee执行失败时会在其它Bee上再起一个task的attempt,所以一个task可能有多个attempt生成

### 系统

* Queen（进程）

承担master的角色，负责客户端作业的提交，作业的分片，作业运行时的管理，Bee的管理。

* Bee（进程）

扮演Slave的角色，负责任务的执行，即Attempt被分到Bee上进行实际执行

* JobHistory

对历史作业的管理

* Supervisor（进程）

当作业开始执行与执行结束后会向Supervisor汇报，该模块中可以实现报警，重试等。

* JobManager

作业的管理，作业运行时任务的重试，状态的变化等，目前有task的重试执行，task超时的并行执行。

* BeeManager

负责Bee的注册状态维护等

* Scheduler

负责作业的调度，现在只提供基于优先级的FIFO调度,对资源的分配采取最闲Bee优先分配的策略。

* UI

对Bee与作业提供监控UI

### 协作

数据的同步一般为一个ETL的初始点，对于作业的触发可以使用crontab进行调度，也可以基于api方式与oozie等工作流系统集成。

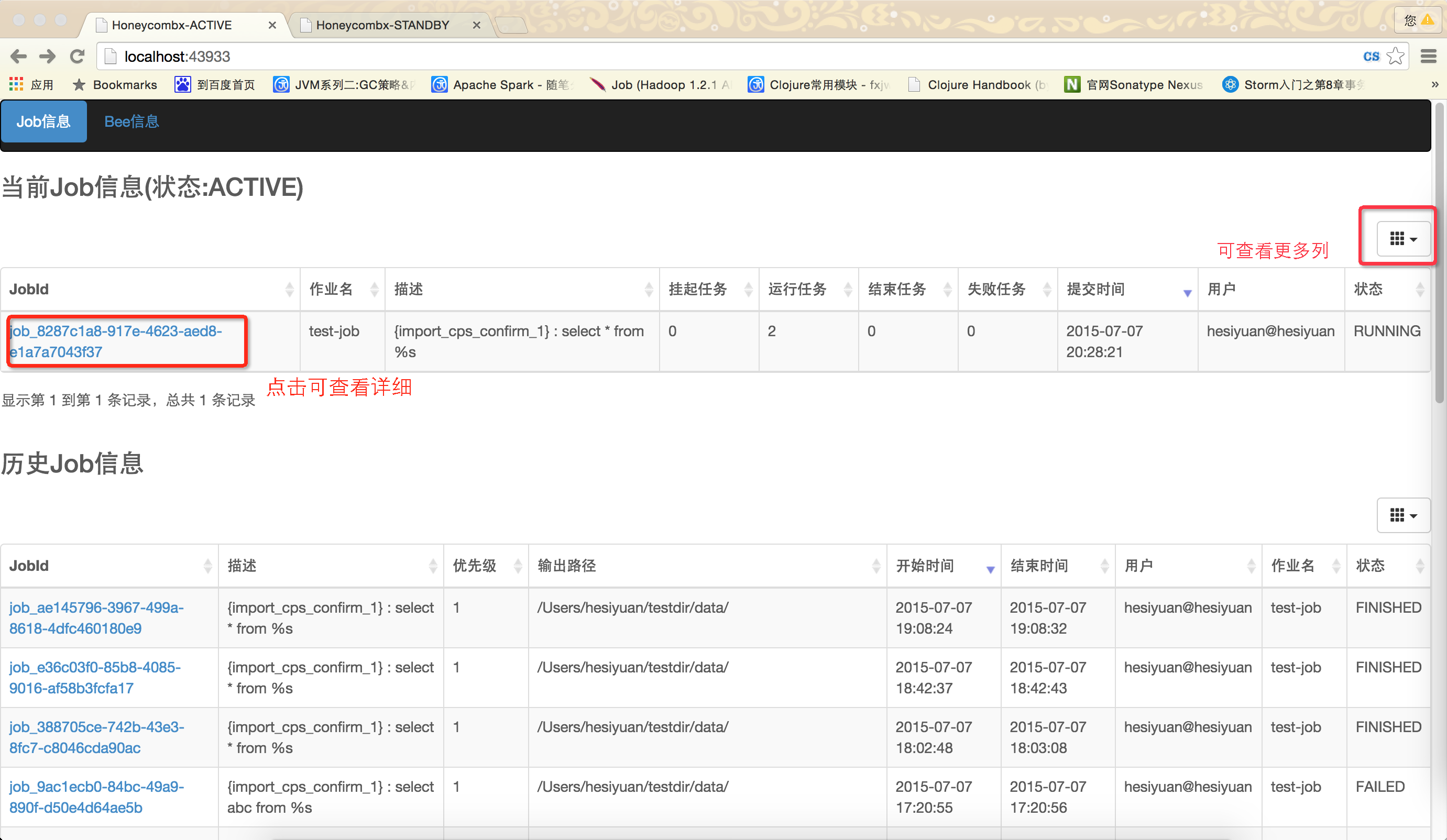
当作业执行完成会时有多种方式进行获知：

1. 完成的作业输出目录下会有\_SUCCESS文件生成
2. 作业提交时可以指定一个回调接口，当作业状态发生变化时会通知该接口
3. 作业提交时可以指定一个完成后执行的脚本
4. 可以通过系统提供的restful接口实时查询作业状态

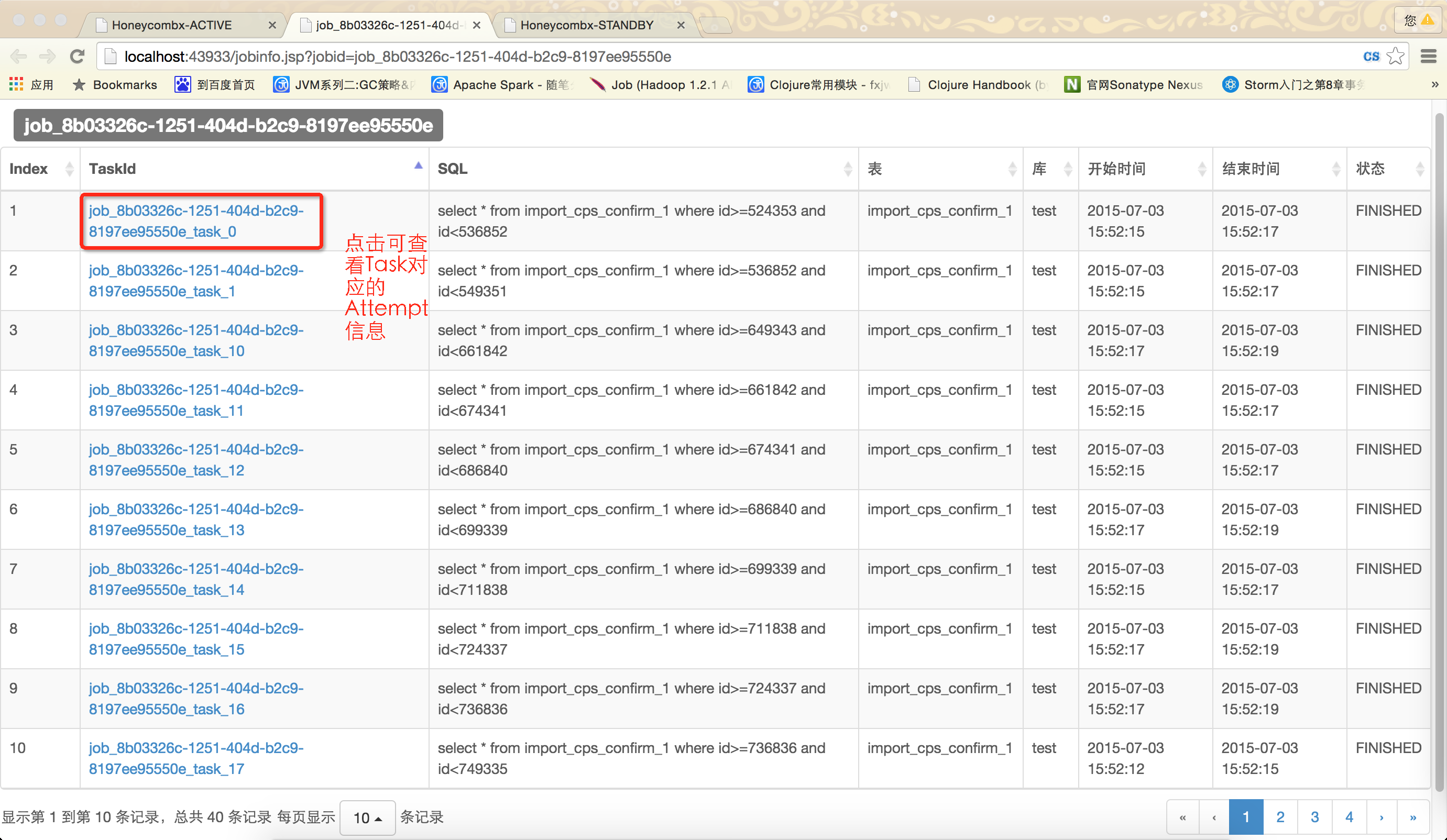
### UI

WebUI主要由以下三个页面组成

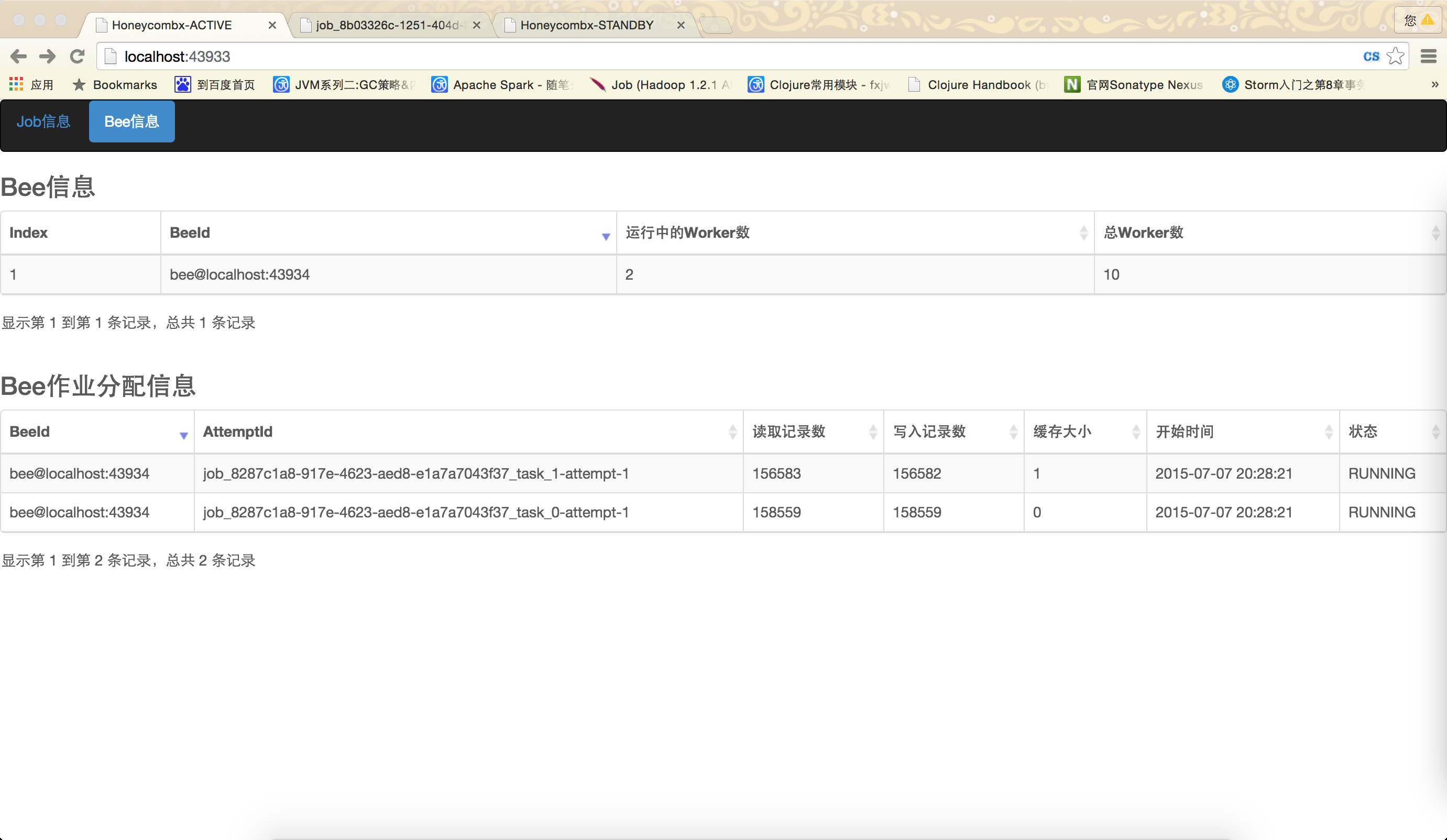
* Job页面



* Job详情页面（即该Job产生的各task）



* Bee信息



### HA

* Queen存在单点故障，借助ZK实现Queen的HA.
* 目前只实现最大粒度的HA,即到作业级别的。
* 用户可以起动多个Queen,但只有一个Queen会被选为leader,客户端与Bee都通过探测的方式与当前leader进行连接，当作业提交时，该作业会被记录到ZK上，在执行完成后被清除，当leader宕机后，会有其它Queen被选为leader,同时客户端与Bee都会与新的leader进行连接，同时新的leader会将未执行完的Job进行恢复。

### 作业执行

* 作业被客户端提交后,Queen根据指定的主键id对作业进行分片，默认使用的分片原则为假定id是连续递增的，找出最大与最小的id然后根据用户指定的分片数来进行均匀的划分。
* 当作业被划分后就产生N个Task,此时集群中若有空闲的worker则生成一个TaskAttempt，然后推给该Worker所属的Bee去执行（并不是hadoop那种用心跳拉的方式，任务启动的代价非常小，所以当数据分布不均时可以加大任务数）。
* 当Bee收到指派给自己的TaskAttempt时会生成一个Fetcher用于拉取数据，一个Sinker用于持久化数据，两者使用Storage来进行数据的交换，即Fetcher从数据库中拉取数据然后写入Storage,Sinker则从storage中读取数据写入hdfs，Fetcher与Sinker会被放入线程池进行执行，即一个Worker会占用两个线程。
* Bee的运行过程中会周期性将自己管理的Worker运行状态汇报给Queen，即Worker的执行进度，错误信息等都会上报给Queen;当Queen得知某个TaskAttempt执行出错时，会在另外一台机器上再启动一个TaskAttempt,但如果已经超过了重试次数则直接将该TaskAttempt所属的Job标记为失败，并杀死所有还在运行的TaskAttempt.
* 当所有task执行完毕后Queen会把数据从临时目录转到目标目录，并将作业的执行过程保存下来， 同时清理所有作业执行过程中所占的内存与磁盘空间