**Twitter Storm: 搭建storm集群**

本文翻译自： <https://github.com/nathanmarz/storm/wiki/Setting-up-a-Storm-cluster>。

这篇文章介绍搭建storm集群并运行的步骤。如果你使用AWS, 那么你可以看一下[storm-deploy](https://github.com/nathanmarz/storm-deploy/wiki)项目，storm-deploy项目使得在Amazon EC2上安装,配置storm集群完全自动化。这篇文章同时也帮你配置好Ganglia以监控CPU， 硬盘以及网络资源的使用情况。

下面是搭建storm集群的一些主要步骤：

1. 搭建ZooKeeper集群。
2. 在Nimbus和所有工作机器上面安装所有的软件。
3. 在Nimbus和所有工作机器上下载并解压storm的发行版。
4. 对storm.yaml进行一些必要的配置。
5. 使用storm脚本来启动所有的必要的进程(nimbus, supervisor, worker)。

搭建ZooKeeper集群

storm使用zookeeper来协调整个集群， 但是要注意的是storm并不用zookeeper来传递消息。所以zookeeper上的负载是非常低的，单个节点的zookeeper在大多数情况下都已经足够了， 但是如果你要部署大一点的storm集群， 那么你需要的zookeeper也要大一点。关于如何部署zookeeper，可以看[这里](http://zookeeper.apache.org/doc/r3.3.3/zookeeperAdmin.html)。

关于如何部署zookeeper有些需要注意的地方：

1. 对zookeeper做好监控非常重要， zookeeper是fail-fast的系统，只要出现什么错误就会退出， 所以一定要监控，更多细节看[这里](http://zookeeper.apache.org/doc/r3.3.3/zookeeperAdmin.html#sc_supervision)。
2. 一定要配置一个cron job来压缩zookeeper的数据和业务日志。zookeeper自己是不会去压缩这些的，所以你如果不设置一个cron job, 那么你很快就会磁盘不够用了，更多细节看[这里](http://zookeeper.apache.org/doc/r3.3.3/zookeeperAdmin.html#sc_maintenance)。

在Nimbus和工作机器上安装必要软件

接下来需要安装Nimbus和工作机器上面的一些storm所依赖的软件。

1. [ZeroMQ 2.1.7](http://www.zeromq.org/area:download)
2. [JZMQ](http://github.com/nathanmarz/jzmq)
3. Java 6
4. Python 2.6.6
5. unzip

上面同时也列出了storm所依赖的软件的版本， 如果版本不一样可能运行不了。

如果安装ZeroMQ和JZMQ的时候有问题，可以看下[安装依赖](https://github.com/nathanmarz/storm/wiki/Installing-native-dependencies)。

在Nimbus和工作机器上下载并解压storm发行版

接下来， 下载storm的发行版，然后解压。storm的发行版可以在[这里](http://github.com/nathanmarz/storm/downloads)找到。

配置storm.yaml

storm发行版在conf/storm.yaml包含了一些配置信息。你可以在[这里](https://github.com/nathanmarz/storm/blob/master/conf/defaults.yaml)看到默认配置。storm.yaml里面的配置比default.xml的优先级要高， 下面是要运行storm集群所必须的配置:

1. **storm.zookeeper.servers**这个配置storm集群使用的zookeeper集群的地址，比如:

|  |  |
| --- | --- |
| 1  2  3 | storm.zookeeper.servers:    - "111.222.333.444"    - "555.666.777.888" |

2.**storm.local.dir** Nimbus和Supervisor在本地磁盘上需要一个目录来存储一些状态信息(jar包， 配置文件之类的东西）你应该在每台机器上创建那个目录，分配正确的权限，比如:

|  |  |
| --- | --- |
| 1 | storm.local.dir: "/mnt/storm" |

3. **java.library.path** 这是storm所依赖的本地依赖(ZeroMQ和JZMQ)的加载地址, 默认的是:/usr/local/lib:/opt/local/lib:/usr/lib, 大多情况下是对的，所以你应该不用更改这个配置。

4. **nimbus.host** 所有工作机器需要nimbus机器的地址， 这样它们才知道去哪里获取jar包和配置文件:

|  |  |
| --- | --- |
| 1 | nimbus.host: "111.222.333.44" |

5. **supervisor.slots.ports** 对于每一台工作机器，这个配置指定在这台工作机器上运行多少工作进程， 每个进程使用一个独立端口来接收消息，这个配置同时也指定使用哪些端口。如果你在这里定义5个端口， storm会在这个机器上最多分配5个工作进程。如果分配3个端口，那么最多分配3个进程。默认的配置是4个：

|  |  |
| --- | --- |
| 1  2  3  4  5 | supervisor.slots.ports:      - 6700      - 6701      - 6702      - 6703 |

启动守护进程

最后一步是启动storm的所有守护进程。还是前面说的那样，对于每个进程都要有监控！storm是一个fail-fast系统，出现什么不可预知的错误的时候它都会退出的。storm被设计成在任何时候都可以安全退出， 在任何时候都能正确重启， 这就是storm为什么不在线程内存储状态 — 如果Nimbus和Supervisor重启的话，不会影响正在运行的topology。下面介绍如何启动这些线程：

1. **Nimbus** 在Nimbus机器上运行bin/storm nimbus
2. **Supervisor** 在每台工作机器上运行bin/storm supervisor, supervisor负责启动和终止工作机器上的工作进程。
3. **UI** storm UI是一个可以查看storm运行状态的的一个网站，通过bin/storm ui运行，访问地址: http://{nimbus.host}:8080/。

就像你能看到的一样，运行storm集群很简单。这些进程会往你解压storm发行版的目录的子目录logs目录里面打日志。

Strom 是Twitter最新的开源系统，下面就test它：  
  
1）安装  
  
Storn使用大量的Clojure动态编程语言，Clojure会将clj或者javascript编译成JVM下的bytecode。  
  
Storm使用了Clojure

## Building Clojure Projects with Leiningen

[[](http://zef.me/wp-content/uploads/2009/11/clojure-icon.gif)](http://zef.me/wp-content/uploads/2009/11/clojure-icon.gif)Everybody who once used Java, struggled with Java’s classpath at some point during their career. You have to put all the right paths in there, the right .jar files and so on, both when compiling and running your Java project. To make this somewhat simpler you typically end up doing it either in an IDE, or using a tool like [Ant](http://ant.apache.org/) or[Maven](http://maven.apache.org/). These are pretty heavy weight tools, and the latter too involve writing XML, which hardly anybody does for fun anymore.

[Leiningen](http://github.com/technomancy/leiningen) is a simple build tool for [Clojure](http://www.clojure.org/), based on Maven (I’m pretty sure). It offers a simple, Clojuresque way of constructing build files for your Clojure projects (which run on the JVM).

To install Leiningen you only have to [download one file](http://github.com/technomancy/leiningen/raw/master/bin/lein) and put it in some directory that’s on your PATH:

cd ~/bin  
wget <http://github.com/technomancy/leiningen/raw/stable/bin/lein>  
chmod +x lein

You then do a self-install:

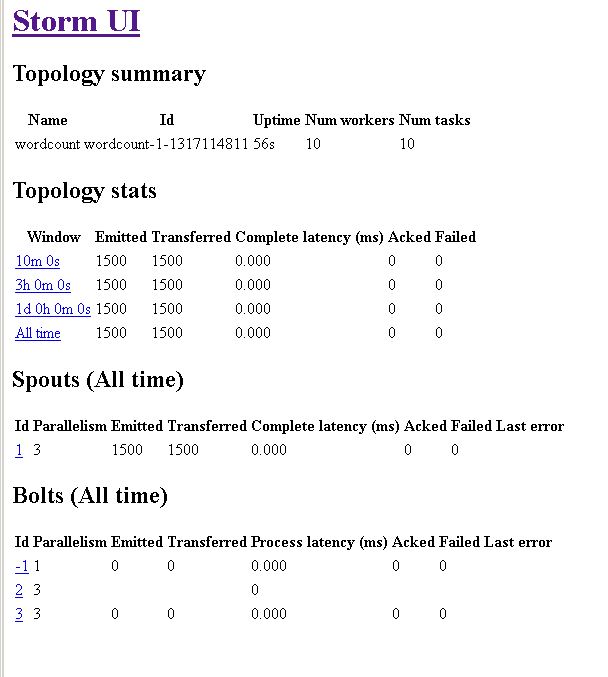
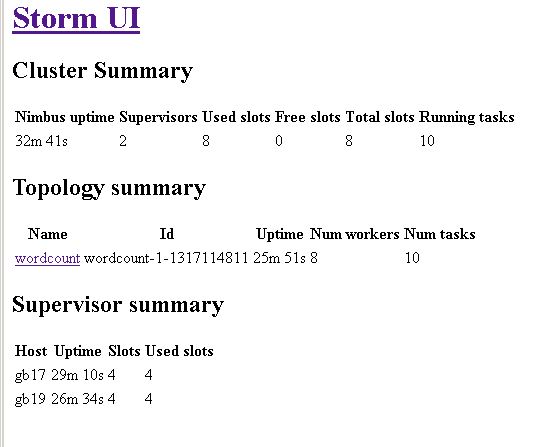
lein self-install

This will dowload a number of jar files, including Clojure itself, so you do not even have to have Clojure installed at this point.

leiningen是在mvn的基础上，实现了一个更简洁的项目管理工具。安装的过程为：  
git clone <http://github.com/technomancy/leiningen.git>  
mvn install  
将leiningen/bin目录加入PATH中  
lein self-install 下载jar，包括clojure，所以不用单独下载安装Clojure.  
回到storm的目录  
lein deps  下载storm中project.clj需要依赖的jar包  
lein compile 出现了如下的错误：  
Compiling backtype.storm.messaging.zmq  
1    [main] ERROR org.apache.zookeeper.server.NIOServerCnxn  - Thread Thread[main,5,main] died  
java.lang.UnsatisfiedLinkError: no jzmq in java.library.path (zmq.clj:1)  
    at clojure.lang.Compiler$InvokeExpr.eval\_r(Compiler.java:2911)  
    at clojure.lang.Compiler.compile1(Compiler.java:5933)  
    at clojure.lang.Compiler.compile1(Compiler.java:5923)  
    at clojure.lang.Compiler.compile(Compiler.java:5992)  
    at clojure.lang.RT.compile(RT.java:368)  
    at clojure.lang.RT.load(RT.java:407)  
    at clojure.lang.RT.load(RT.java:381)  
    at clojure.core$load$fn\_\_4511.invoke(core.clj:4905)  
    at clojure.core$load.doInvoke(core.clj:4904)  
    at clojure.lang.RestFn.invoke(RestFn.java:409)  
    at clojure.core$load\_one.invoke(core.clj:4729)  
    at clojure.core$compile$fn\_\_4516.invoke(core.clj:4916)  
    at clojure.core$compile.invoke(core.clj:4915)  
    at user$eval27.invoke(NO\_SOURCE\_FILE:1)  
    at clojure.lang.Compiler.eval\_r(Compiler.java:5424)  
    at clojure.lang.Compiler.eval\_r(Compiler.java:5415)  
    at clojure.lang.Compiler.eval\_r(Compiler.java:5391)  
    at clojure.core$eval.invoke(core.clj:2382)  
    at clojure.main$eval\_opt.invoke(main.clj:235)  
    at clojure.main$initialize.invoke(main.clj:254)  
    at clojure.main$null\_opt.invoke(main.clj:279)  
    at clojure.main$main.doInvoke(main.clj:354)  
    at clojure.lang.RestFn.invoke(RestFn.java:422)  
    at clojure.lang.Var.invoke(Var.java:369)  
    at clojure.lang.AFn.applyToHelper(AFn.java:165)  
    at clojure.lang.Var.applyTo(Var.java:482)  
    at clojure.main.main(main.java:37)  
Caused by: java.lang.UnsatisfiedLinkError: no jzmq in java.library.path  
    at java.lang.ClassLoader.loadLibrary(ClassLoader.java:1689)  
    at java.lang.Runtime.loadLibrary0(Runtime.java:823)  
    at java.lang.System.loadLibrary(System.java:1028)  
    at org.zeromq.ZMQ.<clinit>(ZMQ.java:34)  
    at java.lang.Class.forName0(Native Method)  
    at java.lang.Class.forName(Class.java:247)  
    at clojure.lang.RT.classForName(RT.java:1566)  
    at clojure.lang.Compiler$ImportExpr.eval\_r(Compiler.java:604)  
    at clojure.lang.Compiler.compile1(Compiler.java:5933)  
    at clojure.lang.Compiler.compile1(Compiler.java:5923)  
    at clojure.lang.Compiler.compile1(Compiler.java:5923)  
    at clojure.lang.Compiler.compile(Compiler.java:5992)  
    at clojure.lang.RT.compile(RT.java:368)  
    at clojure.lang.RT.load(RT.java:407)  
    at clojure.lang.RT.load(RT.java:381)  
    at clojure.core$load$fn\_\_4511.invoke(core.clj:4905)  
    at clojure.core$load.doInvoke(core.clj:4904)  
    at clojure.lang.RestFn.invoke(RestFn.java:409)  
    at clojure.core$load\_one.invoke(core.clj:4729)  
    at clojure.core$load\_lib.doInvoke(core.clj:4766)  
    at clojure.lang.RestFn.applyTo(RestFn.java:143)  
    at clojure.core$apply.invoke(core.clj:542)  
    at clojure.core$load\_libs.doInvoke(core.clj:4800)  
    at clojure.lang.RestFn.applyTo(RestFn.java:138)  
    at clojure.core$apply.invoke(core.clj:542)  
    at clojure.core$require.doInvoke(core.clj:4869)  
    at clojure.lang.RestFn.invoke(RestFn.java:422)  
    at backtype.storm.messaging.zmq$loading\_\_4410\_\_auto\_\_.invoke(zmq.clj:1)  
    at clojure.lang.AFn.applyToHelper(AFn.java:159)  
    at clojure.lang.AFn.applyTo(AFn.java:151)  
    at clojure.lang.Compiler$InvokeExpr.eval\_r(Compiler.java:2906)  
    ... 26 more  
Compilation failed.  
  
Storm的通信库使用ZeroMQ，以及JZMQ（Java API),需要安装这两个包：  
1）ZeroMQ：  
wget <http://download.zeromq.org/historic/zeromq-2.1.7.tar.gz>  
tar -xzf zeromq-2.1.7.tar.gz  
cd zeromq-2.1.7  
./configure  
make  
sudo make install  
2）jzmq  
#install jzmq  
git clone <https://github.com/nathanmarz/jzmq.git>

cd jzmq  
./autogen.sh  
./configure  
make  
sudo make install  
2） 编译  
重新使用 lein compile 报错：  
Compiling backtype.storm.messaging.zmq  
0 [main] ERROR org.apache.zookeeper.server.NIOServerCnxn - Thread Thread[main,5,main] died  
java.lang.UnsatisfiedLinkError: /usr/local/lib/libjzmq.so.0.0.0: libzmq.so.1: cannot open shared object file: No such file or directory (zmq.clj:1)  
at clojure.lang.Compiler$InvokeExpr.eval\_r(Compiler.java:2911)  
at clojure.lang.Compiler.compile1(Compiler.java:5933)  
at clojure.lang.Compiler.compile1(Compiler.java:5923)  
at clojure.lang.Compiler.compile(Compiler.java:5992)  
at clojure.lang.RT.compile(RT.java:368)  
at clojure.lang.RT.load(RT.java:407)  
at clojure.lang.RT.load(RT.java:381)  
at clojure.core$load$fn\_\_4511.invoke(core.clj:4905)  
at clojure.core$load.doInvoke(core.clj:4904)  
at clojure.lang.RestFn.invoke(RestFn.java:409)  
at clojure.core$load\_one.invoke(core.clj:4729)  
at clojure.core$compile$fn\_\_4516.invoke(core.clj:4916)  
at clojure.core$compile.invoke(core.clj:4915)  
at user$eval27.invoke(NO\_SOURCE\_FILE:1)  
at clojure.lang.Compiler.eval\_r(Compiler.java:5424)  
at clojure.lang.Compiler.eval\_r(Compiler.java:5415)  
at clojure.lang.Compiler.eval\_r(Compiler.java:5391)  
at clojure.core$eval.invoke(core.clj:2382)  
at clojure.main$eval\_opt.invoke(main.clj:235)  
at clojure.main$initialize.invoke(main.clj:254)  
at clojure.main$null\_opt.invoke(main.clj:279)  
at clojure.main$main.doInvoke(main.clj:354)  
at clojure.lang.RestFn.invoke(RestFn.java:422)  
at clojure.lang.Var.invoke(Var.java:369)  
at clojure.lang.AFn.applyToHelper(AFn.java:165)  
at clojure.lang.Var.applyTo(Var.java:482)  
at clojure.main.main(main.java:37)  
Caused by: java.lang.UnsatisfiedLinkError: /usr/local/lib/libjzmq.so.0.0.0: libzmq.so.1: cannot open shared object file: No such file or directory  
at java.lang.ClassLoader$NativeLibrary.load(Native Method)  
at java.lang.ClassLoader.loadLibrary0(ClassLoader.java:1758)  
at java.lang.ClassLoader.loadLibrary(ClassLoader.java:1683)  
at java.lang.Runtime.loadLibrary0(Runtime.java:823)  
at java.lang.System.loadLibrary(System.java:1028)  
at org.zeromq.ZMQ.<clinit>(ZMQ.java:34)  
at java.lang.Class.forName0(Native Method)  
at java.lang.Class.forName(Class.java:247)  
at clojure.lang.RT.classForName(RT.java:1566)  
at clojure.lang.Compiler$ImportExpr.eval\_r(Compiler.java:604)  
at clojure.lang.Compiler.compile1(Compiler.java:5933)  
at clojure.lang.Compiler.compile1(Compiler.java:5923)  
at clojure.lang.Compiler.compile1(Compiler.java:5923)  
at clojure.lang.Compiler.compile(Compiler.java:5992)  
at clojure.lang.RT.compile(RT.java:368)  
at clojure.lang.RT.load(RT.java:407)  
at clojure.lang.RT.load(RT.java:381)  
at clojure.core$load$fn\_\_4511.invoke(core.clj:4905)  
at clojure.core$load.doInvoke(core.clj:4904)  
at clojure.lang.RestFn.invoke(RestFn.java:409)  
at clojure.core$load\_one.invoke(core.clj:4729)  
at clojure.core$load\_lib.doInvoke(core.clj:4766)  
at clojure.lang.RestFn.applyTo(RestFn.java:143)  
at clojure.core$apply.invoke(core.clj:542)  
at clojure.core$load\_libs.doInvoke(core.clj:4800)  
at clojure.lang.RestFn.applyTo(RestFn.java:138)  
at clojure.core$apply.invoke(core.clj:542)  
at clojure.core$require.doInvoke(core.clj:4869)  
at clojure.lang.RestFn.invoke(RestFn.java:422)  
at backtype.storm.messaging.zmq$loading\_\_4410\_\_auto\_\_.invoke(zmq.clj:1)  
at clojure.lang.AFn.applyToHelper(AFn.java:159)  
at clojure.lang.AFn.applyTo(AFn.java:151)  
at clojure.lang.Compiler$InvokeExpr.eval\_r(Compiler.java:2906)  
... 26 more  
Compilation failed.  
  
排查是找不到对应的so文件，因为so文件被放到了/usr/local/lib下，需要更新以下搜索路径，  
sudo ldconfig  
  
再次编译，  
lein compile  
通过。  
  
获取storm-starter，主要是一些Example；  
git clone <http://github.com/nathanmarz/storm-starter.git>  
  
启动Zookeeper集群，具体的配置可以参考<http://zookeeper.apache.org/doc/r3.3.3/zookeeperAdmin.html>  
  
配置Zookeeper集群之后，如果Zookeeper的集群设置了多个节点，可以提供多个容灾的能力。  
  
然后配置storm/conf/storm.yaml,默认配置为storm/conf/defaults.yaml  
  
遇到的主要问题是配置conf/storm.yaml的某些选项没有作用。与Storm的作者进行了沟通的结果是，它说如果你设置成Local的情况会ignore这些配置选项，可是我是按照Distributed来进行的。  
  
后来发现这种问题在启动Nimbus的时候，仍然存在，可能的原因是在backtype.command.conf\_load.clj文件出了错误。每次没能加载最新的value值。经过与Storm的作者Nathan沟通，通过源码编译的项目，与release  
版本有一定的差别，后来从<https://github.com/nathanmarz/storm/downloads> 获取了一个release版本，我将storm/storm\*.jar 替换成release包中的storm\*.jar,问题即可解决。  
  
注意，版本在变化，上面的提到的trick可能在将来的版本中消失。  
  
3）部署  
ok，下面配置集群环境的Storm：  
  
#配置conf/storm.yaml  
  
#由于我的zeromq以及jzmq没有安装到默认的/usr/local/lib, /opt/local/lib, /usr/lib, 因此需要加入java.library.path  
 java.library.path: "/usr/local/lib:/opt/local/lib:/usr/lib:/home/jiangbing/storm/zeromq/lib"  
  
#设置nimbus.host,相当于Hadoop-Jobtracker的内容，gb17是选中的机器的hostname.  
 nimbus.host: "gb17"  
  
#设置storm工作目录  
 storm.local.dir: "/opt/jiangbing/storm"  
  
#设置storm依赖的Zookeeper集群的信息  
 storm.zookeeper.servers:  
- "gb17"  
- "gb18"  
- "gb19"  
  
#设置storm链接Zookeeper Leader的端口号  
 storm.zookeeper.port: 2284  
  
（特别注意使用yaml文件在解析的时候，每一个选项在前面一定要有一个空格，不然yaml解析器将读不到配置参数.)  
  
启动nimbus，  
bin/storm nimbus  
打印出的-Djava.library.path仍然是默认配置，查看bin/storm的内容，发现在confvalue函数中获取java.library.path的时候，没有加载$storm\_home/conf目录，于是修改bin/storm程序内容如下：  
  
25 def confvalue(name):  
26     cp = get\_classpath([])  
27     command = ["java", "-client", "-cp", cp +":"+ STORM\_DIR+ "/conf", "backtype.storm.command.config\_value", name] // add the directory 'storm/conf' into the path , and original :  
//command = ["java", "-client", "-cp", cp , "backtype.storm.command.config\_value", name]   
  
 28     p = sub.Popen(command,stdout=sub.PIPE)  
 29     output, errors = p.communicate()

 30     lines = output.split("\n")  
 31     for line in lines:  
 32         tokens = line.split(" ")  
 33         if tokens[0] == "VALUE:":  
 34             return tokens[1]

再次启动bin/storm nimbus  
选择不同的worknode启动supervisor， bin/storm supervisor  
run第一个WordCountTopology程序，github上有一个子项目叫做storm-start,里面包含了WordCount等应用，需要修改一下：下面仅列出需要修改的函数(storm.starter.WordCountTopology)。  
public static void main(String[] args) throws Exception {  
          
        TopologyBuilder builder = new TopologyBuilder();  
          
        builder.setSpout(1, new RandomSentenceSpout(), 5);  
          
        builder.setBolt(2, new SplitSentence(), 8)  
                 .shuffleGrouping(1);  
        builder.setBolt(3, new WordCount(), 12)  
                 .fieldsGrouping(2, new Fields("word"));          
        Config conf = new Config();  
        conf.setDebug(true);  
        conf.setMaxTaskParallelism(3);  
       
          
        //Config conf = new Config();  
        conf.setNumWorkers(20);  
        conf.setMaxSpoutPending(5000);  
        StormSubmitter.submitTopology("wordcount", conf, builder.createTopology());  
  
        //LocalCluster cluster = new LocalCluster();  
        //cluster.submitTopology("word-count", conf, builder.createTopology());  
        Thread.sleep(10000);          
    }  
然后，编译打jar包，/tmp/storm-start.jar  
  
向Nimbus提交jar，执行。  
bin/storm jar /tmp/storm-start.jar storm.starter.WordCountTopology   
此时会报错，由于执行bin/storm jar \*\*\* 命令时，没有载入正确的$STORM\_HOME/conf,所以再次修改bin/storm：  
 1 #!/usr/bin/python  
  2   
  3 import os  
  4 import sys  
  5 import random  
  6 import subprocess as sub  
  7   
  8 CONF\_DIR = os.path.expanduser("~/.storm")  
  9 STORM\_DIR = "/".join(os.path.abspath( \_\_file\_\_ ).split("/")[:-2])  
 10 CONF\_DIR = STORM\_DIR + "/conf" #确定目录conf文件目录，这样在执行bin/storm jar的时候可以将正确的conf目录加入classpath中。  
  
后来再次执行  
bin/storm jar /tmp/storm-start.jar storm.starter.WordCountTopology   
终于通过作业提交.  
上传一下Storm的ui吧  
[](http://photo.blog.sina.com.cn/showpic.html#blogid=4a1f59bf0100xn8l&url=http://s8.sinaimg.cn/orignal/4a1f59bftade8fabb2397&690)  
  
[](http://photo.blog.sina.com.cn/showpic.html#blogid=4a1f59bf0100xn8l&url=http://s6.sinaimg.cn/orignal/4a1f59bftade8faabbde5&690)