OverviewDeploymentConfigs

/!\ :This full page requires a complete update to reflect Nutch 1.3 release: (!)

Overview of Deployment Configurations in Nutch 0.8

(11/2005 Paul Baclace)

This page describes a range of deployment configurations, the assumptions involved, and the relevant property settings. The primary focus is on a few canonical deployments scenarios and surrounding issues. Relevant properties are described, but a complete description of all properties is not attempted here.

The process startup sequence is also described in order to see differences between different deployments.

Flexibility of assumptions is noted with MUST (rigid) or SHOULD (highly recommended, but could be different for the adventurous).

Configuration File Overview

When building Nutch, the conf directory has 2 important property files that are put into the classpath for lookup at runtime:

- *nutch-default.xml* the place for universal defaults as set by the Nutch developers.
- *nutch-site.xml* the highest priority properties that override all other.

The Java System Properties are *not* consulted for Nutch properties, so -D style commandline overriding is strongly discouraged. However, System Properties are used when standard properties are to be found.

The bin/nutch sh script places $NUTCH_HOME/conf at the beginning of the classpath so that the xml property files can be found.

Nutch Shell Scripts

A meta-assumption here is that the sh scripts in the nutch bin directory are used to start and control the ensemble of processes across many machines.

The Nutch shell scripts are simple and elegant and they form a call hierarchy, starting at the top level:

1. start_all.sh or stop_all.sh - start and stop whole ensemble. 2. nutch_daemons.sh - run a Nutch command on all slave hosts. 3. slaves.sh - run a shell command on all slave hosts. 4. nutch_daemon.sh - run a Nutch command as a daemon with a start|stop argument like a regular Unix/Linux /etc/rc.local script; the process pid is stored during start and used during stop. At start, runs rsync to master initiated on slave. 5. nutch - run a Nutch command, specified either as a command name or full path to a class, using the JVM.

Depending upon the context of use, any level of these scripts can be handy on the command line.

Configuration Assumptions

For simplicity of configuration, filenames you pass to commands SHOULD be pathnames that work on all hosts. When working with just a few hosts, this seems to be a limitation, but it obviously makes a lot of sense when hundreds or thousands of machines are involved.

1. property settings are meant to be the same across hosts; they are SHOULD not be customized per host (they are not even settable on the commandline, so per-process settings are discouraged). 2. filenames and paths are meant to be the same across hosts (SHOULD). 3. Some file paths are ambivalent about NDFS/local filesystem and are interpreted depending on which kind of filesystem is in use. 4. each machine SHOULD have (including the master) nutch installed in the same filesystem path. 5. The ndfs.data.dir and mapred.local.dir properties list comma separated directories. Only those that exist are used. So not all machines are required to have exactly the same devices.

System Assumptions

1. The env var NUTCH_MASTER is set to the hostname of the master machine. 2. The slave nodes are defined by putting list of hostnames, one per line, in ~/.slaves (alternatively, use NUTCH_SLAVES to refer to a different file). 3. a cluster of machines is managed from a master machine, without a firewall in bewteen any of the machines (MUST, for simplicity). Many tcp/ip ports are used. 4. the master machine MUST have a no-password login (ssh) to all the slave machines, using the same username. 5. set environment variables in ~/.ssh/environment, since ssh does not source your .bash_profile. These include JAVA_HOME, NUTCH_LOG_DIR, NUTCH_SLAVES and NUTCH_MASTER. 6. make sure that your NUTCH_LOG_DIR and the directories named in ndfs.data.dir exist on all slaves. This can be done most easily with bin/slaves.sh.

Deployment and Startup Sequences

1. Cluster deployment with too many machines to customize (probably more than 4; 1000 machines should be possible):
   6. the ensemble starts by running bin/start-all.sh on the master.
   7. start-all.sh uses bin/nutch-daemons.sh which uses nutch-daemon.sh to run rsync (to update jars and conf files from master) and then run one datanode process on each slave (in the background without waiting, one daemon thread is started per comma-separated storage device, non-existent storage devices in the list are ignored).
   8. start-all.sh runs one namenode and one jobtracker on the master.
   9. start-all.sh uses bin/nutch-daemons.sh run one tasktracker process on each slave (in the background without waiting).

   B. Cluster of a few machines:
   While the cluster deployment and startup sequence can apply for 2 more more machines, the idea behind this case is to address experimental

configurations for federations of machines that might have firewalls in between them; 1. *Add more details here...*

C. One developer debugging on one machine:

  a. *Add more details here*

back to FrontPage