#########################################################################

## Author: Jin Li

## Date: Jan. 2015

## This document describes procedure to compile/deploy Prajna in

## a domain joined cluster.

#########################################################################

Dependencies:

The following components are needed to deploy Prajna in a domain joined cluster.

\* Each machine in the cluster should be installed with .Net 4.0.

\* The person in charge of deployment of Prajna cluster needs administrator privilege on the cluster node. (Note, people who later run Prajna job doesn’t need the privilege.)

\* Remote Powershell scripting will need to be enabled on the cluster machine (see 3.)

\* Memory limitation on power shell executed program be lifted for the cluster node.

\* Firewall be opened for port used by Prajna (see 8).

1. Please build the solution

Prajna.sln

with x64 target.

2. Prajna domain join cluster is designed to run on a Windows Server 2012 data center cluster. We believe that it will also run on other server version, but haven't got the time and resource to test and validate deployment on other Windows OS version. If you want to run Prajna on a cluster of other OS system, please give the following deployment procedure a try. If failed, please report the failure back and we can work with you to see if Prajna deployment is feasible on your target cluster.

3. The machine in the cluster needs the capability to enable Remote Powershell execution. This can be achieved by running the Powershell script prepareclient.ps1 with admin privilege on the target machine once.

4. You should lift the memory limitation of the Remote Powershell. Please refer to:

<http://blogs.msdn.com/b/powershell/archive/2010/05/03/configuring-wsman-limits.aspx>

Otherwise, the Prajna program may crash when it reaches the memory limitation (usually below 1GB).

5. If there is a large cluster of machines that you need to prepare, the best practice is to prepare one machine by running the script prepareclient.ps1 with admin privilege once, captures the image of the machine, and then pushes the image to the rest of the nodes.

6. We have observed that many internal Microsoft server cluster have already enabled Remote Powershell and lift the memory limitation. So you may not need to perform 3-5 on your target cluster.

Do be aware that if all Powershell script fails to execute on any cluster node, the root problem is most probably 3. If the Prajna client fails when the node is involved in memory intensive operation, the root problem may be 4.

7. Please use storecred.ps1 to create a credential file (stored at $CredentialFile in config.ps1.sample) that will be used to Launch the Prajna daemon at the remote machine. You may optional type the credential every time to launch/shutdown the daemon.

7. Most of the Prajna Powershell allow you to specify the cluster in the following way:

a. .lst file: the file contains multiple lines, each line is the name of a machine.

b. .inf file: the file contains coded machine configuration for Prajna job.

8. Please run Powershell script

firewall.ps1 machinename

or firewall.ps1 cluster.lst

to open the firewall for port used by Prajna during its execution. Each Prajna client will use one port (specify by variable port in config.ps1.sample) for daemon program, and one port for each attached Prajna program run on the node (with port range specified by jobport in config.ps1.sample). The firewalls of all those ports need to be opened, otherwise, the Prajna program will not be able to get any traffic on the cluster, and will wait for information indefinitely.

10. Please copy config.ps1.sample to config.ps1

copy config.ps1.sample config.ps1

and modify config.ps1 so that the directory and variable corresponding to the directory that you want to test. The $() variable referred in the unit test below will all be defined in the config.ps1 file.

11. We have the following unit test to test Prajna operation. These unit test program also serves as start template which you can modify to develop your own Prajna program, as follows.

1. Write\_unittest.ps1

a.1 Recursively write a folder ($localDirName) to the remote cluster ($remoteDKVname) , each file becomes a key, value pair (key=filename, value=file content). This also serves as test data for other unit test.

a.2 Write a URL list ($uploadfile) to the remote cluster (as DKV $uploadRemote). Each line of the URL list contains a URL (with column number specified by $uploadkey), and a tag (of URL information, with column number specified by $tagKey). The written DKV will be used in the read and download unittest.

a.3 Generate and store to remote cluster a set of random vector (as DKV ($remoteVector) and ($remoteVector)\_NOISE)

a.4 Generate and store to remote cluster a set of random vector (as DKV $remoteVector1000) and perform a map reduce operation to aggregate the random vector set.

1. Read\_unittest.ps1

b.1 Read a remote DKV (with name $remoteDKVName, in which key is the filename, and value is the file content). Store this DKV back to a local folder ($localSaveDir)

b.2 Calculate SHA512 hash of a remote DKV(with name $remoteDKVName, in which key is the filename, and value is the file content), and show the hash.

b.3/b.4 Calculate statistics of the remote DKV written by a.3

b.5 Calculate statistics of the union of the remote DKV written by a.3

b.6 Validate the map reduce result of the unit test a.4.

b.7 Perform a distributed sort.

b.8 Distributed log analysis.

b.9 Distributed hash join to extract URL tags generated by a.2.

1. Map reduce unit test (same as a.4)

Generate and store to remote cluster a set of random vector (as DKV $remoteVector1000) and perform a map reduce operation to aggregate the random vector set.

1. Calculate the intra distance of a vector set via cross join (-dist 1) or the intra distance distribution of a vector set via cross join & fold.
2. Distributed web crawling unittest.

Distributed crawl the URL list specified in a.2, save the crawling result to a DKV in the cluster.