Lab 1 -- Test Performance of SAMOA

Li Huang
2014/2/18

Goal:

Test performance(speed) of SAMOA on cluster environment.
Test SAMOA's scalability: check whether SAMOA run faster with more computing nodes.

Experiment Preparation:

- 3 computers:

node1: P4 1.8, 576M ram
node2: P4 1.8, 768M ram
node3: P4 2.35, 495M ram

- D-Link LAN switch:

Connect the three nodes to constitute a cluster.
These three nodes are in a LAN and can communicate with each other.

- Software (for each node):

Ubuntu 12.04 desktop 32-bit
Yahoo S4 0.6 (Stream process platform for SAMOA)
Yahoo SAMOA 0.0.1

- Tools:

SSH Server/Client should be setup up on each node, to let main operating node remote access.
Zookeeper Inspector, Apache Http Server should be setup on main operating node: node 3.

- Configure:
  - Add node names and IPs in /etc/hosts of each node, to let them know each other's IP and hostname.
  - edit the file "etc/profile", to add $SAMOA_HOME, $S4_HOME into environment variables.
  - edit the file "opt/s4/subprojects/s4-core/src/mian/resources/logback.xml", change <root level="debug"> into <root level="info"> to turn off the massive debug information output to screen which slow down the SAMOA performance. Then rebuild S4.

- Turn off the “Screen Saver” of Ubuntu, avoid the performance loss caused by animated screen saver.

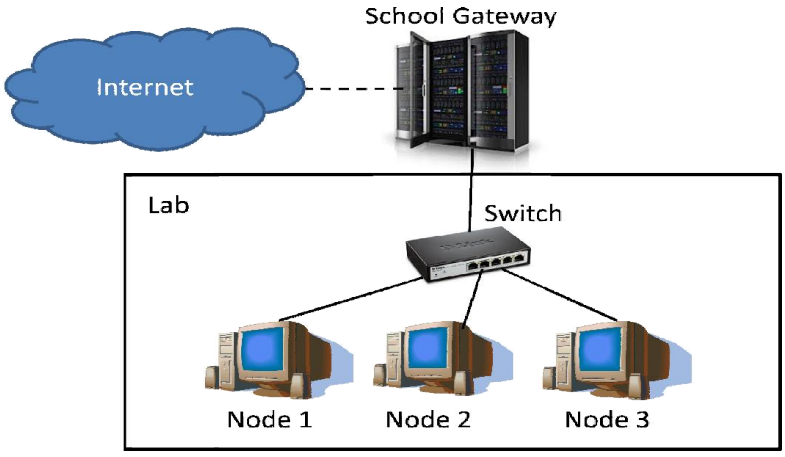

Fig. experiment network topology

Experiment Contents:

Outline

My experiment contains 3 tests:
Decision tree classification and evaluation with
(1) Small amount of train/test data
(2) Medium amount of train/test data
(3) Big amount of train/test data

The different of these 3 tests are:
Different number of test data instance;
Different number of attributes for each instance of test data;

For each test, I compare the performances of different configurations:
(1) 1 node, local mode
(2) 1 node, S4 mode
(3) 2 nodes, S4 mode
(4) 3 nodes, S4 mode

Task:
In the experiments, I run a “Decision Tree Classification and Evaluation” task, which named “**PrequentialEvaluation**” task, on SAMOA, and test the performance (running time) of this application running on different configurations of only one node, two nodes and three nodes.

Zookeeper:

Except running the classification task, one of the nodes needs to run the “Zookeeper server”, which manage the whole cluster. In the whole experiment, I always use node3 as the zookeeper server because node3 has the best CPU.

Operating computer:
All the operations and commands are entered on node 3. To deal with some commands running on node 1 or node2, I use “remote terminal -- SSH” on node 3 to login to node 1 or node 2, and run these commands remotely.

Test data:
Test data is randomly streaming data generated by a “Random Tree Generator” task of SAMOA, each instance of test data contains x nominal attributes and y numeric attributes. The number of attributes and the number of total instances can be set differently in this experiment.

Process

1 node, local mode

(1) boot the node x

(2) start the task on local mode

```
cd $SAMOA_HOME
bin/samoa local target/SAMOA-Local-0.0.1-SNAPSHOT.jar “PrequentialEvaluation \
-d /tmp/dump.csv \
-i 100000 -f 10000 \
-l (com.yahoo.labs.samoa.learners.classifiers.trees.VerticalHoeffdingTree -p 4) \
-s (com.yahoo.labs.samoa.moa.streams.generators.RandomTreeGenerator -c 2 -o 10 -u 10)”
```

Then the node x will start the task and output the result, like:

```
[main] INFO com.yahoo.labs.samoa.evaluation.EvaluatorProcessor - com.yahoo.labs.samoa.evaluation.EvaluatorProcessorid = 0
evaluation instances,classified instances,classifications correct (percent),Kappa Statistic (percent),Kappa Temporal Statistic (percent)
10000.0,10000.0,70.5,38.21395606404887,40.07718870607352
20000.0,20000.0,77.64999999999999,54.202093661021614,54.438895117725
30000.0,30000.0,80.62333333333333,60.46686951831741,60.334356874786764
40000.0,40000.0,82.4825,64.32914017039238,64.01314775820451
50000.0,50000.0,83.83,67.12827133507653,66.75165522062755
60000.0,60000.0,84.77,69.05855287039135,68.6797367699479
70000.0,70000.0,85.45,70.45266688397568,70.09074090388513
80000.0,80000.0,85.9525,71.47643972862416,71.08973039720107
90000.0,90000.0,86.32444444444445,72.22217294674633,71.86549934852677
```

100000.0,100000.0,86.72999999999999,73.0339534093311,72.70052870867534

[main] INFO com.yahoo.labs.samoa.evaluation.EvaluatorProcessor - total evaluation time: 23 seconds for 100000 instances

From the last line of output, we know the total running time of this task is 23 seconds.

n node, cluster mode

(3) At first, boot these three computers.

(4) Then start zookeeper server on node3:

```
cd $S4_HOME
. /s4 zkServer -clean
```

(5) Create a cluster with ***n*** node(s), n can be 1,2 or 3

```
cd $S4_HOME
. /s4 newCluster -c=cluster -flp=12000 -nbTasks=n -zk=node3
```

Parameters

-c: cluster name

-flp: network port number of cluster

-nbTaks: number of nodes in the cluster

-zk: zookeeper server

e.g.

```
./s4 newCluster -c=cluster -flp=12000 -nbTasks=2 -zk=node3
```

means this cluster contains 2 nodes. Currently, S4 0.6 only support predefine static nodes number, not support dynamic nodes configuration.

(6) Deploy "PrequentialEvaluation" task to cluster

```
cd $SAMOA_HOME
bin/samoa S4 target/SAMOA-S4-0.0.1-SNAPSHOT.jar "PrequentialEvaluation \
-d /tmp/dump.csv \
-i 100000 -f 10000 \
-l (com.yahoo.labs.samoa.learners.classifiers.trees.VerticalHoeffdingTree -p 4) \
-s (com.yahoo.labs.samoa.moa.streams.generators.RandomTreeGenerator -c 2 -o 10 -u 10)"
```

Mean of this command:

samoa S4 target/SAMOA-S4-0.0.1-SNAPSHOT.jar: Run SAMOA on S4 platform

-d /tmp/dump.csv: Output(result) file is dump.csv

-i **100000** -f 10000: The number of input data for classification training is 100,000 instances. After training with every 10,000 instances, evaluation task would run once. In our example, evaluation would run 100,000/10,000 = 10 times. This parameter would change in different test configurations.

-l (com.yahoo.labs.samoa.learners.classifiers.trees.VerticalHoeffdingTree -p 4):
learner is "Vertical Hoeffding Tree", and its **parallel level** is **4** meaning that it will use 4 local statistic PIs(process instance).

Vertical Hoeffding Tree
Vertical Hoeffding Tree is a distributed decision tree classifier. It can parallel the computation process onto **n** PIs. The logic structure of this algorithm is below:

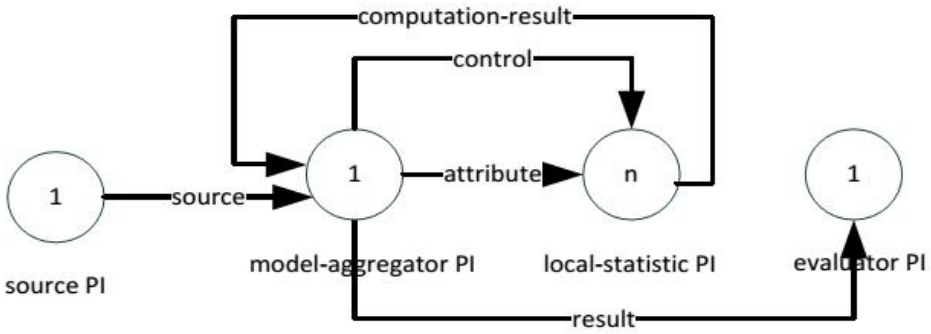

Figure 4.6: Vertical Hoeffding Tree

The detail of this algorithm can be found in the article ***"Distributed Decision Tree Learning for Mining Big Data Streams"*** by **Arinto Murdopo**. "Vertical" mean this algorithm is vertical parallelism, which split the attributes into n PIs. For example, assume the each input sample has 20 attributes, our parallel level is 4, and this algorithm is running on 2 nodes. Then each nodes will run 4/2 = 2 local-statistic PI, and each local-statistic PI will do computation with 20/4=5 attributes. So each node will compute with 10 attributes. At last the algorithm will aggregate the results by the "model-aggregator PI", which runs on one of the two node.

Because this algorithm is "vertical parallelism", we can exploit the advance of parallelism only when the number of attributes is high.

-s (com.yahoo.labs.samoa.moa.streams.generators.RandomTreeGenerator -c 2 **-o 10 -u 10**)
The input data stream is generated by "RandomTreeGenerator".
The data generated has 2 target classes(-c), and each instance has **10 nominal attributes(-o)** and **10 numeric attributes(-u)**. This two parameters (-i, -u) would change in different test configurations.

The "PrequentialEvaluation" task also has the parameter -e EvaluationMethod
Here I did not type this parameter, means I use the default evaluation method (**BasicClassificationPerformanceEvaluator**) which output the "Correction Rate and Kappa Statistics"

The logic structure of “Prequantial Evaluation” in terms of SAMOA API is below:

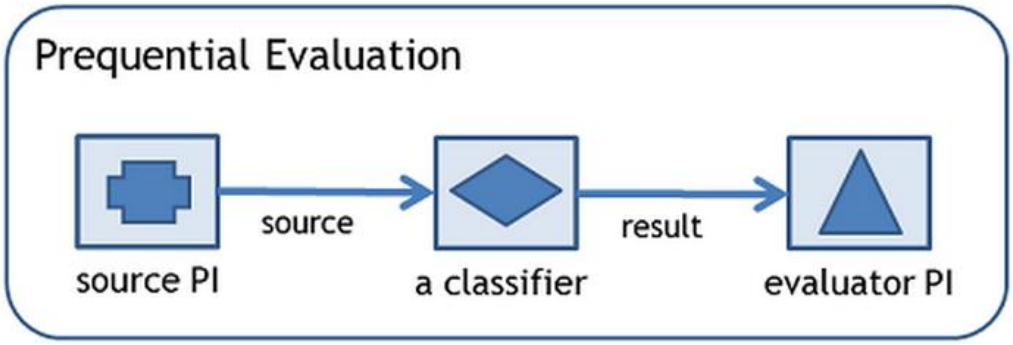

Fig. Process of Prequential Evaluation task in SAMOA

After I enter this command, the “Prequential Evaluation” task has been deployed to the cluster, and is waiting for available nodes started to run this task.

(7) Start a node:

remote login to node1 (from the operating machine - node 3)

```
ssh hl@node1
enter password: ****
cd $S4_HOME
s4 node -c=cluster -zk=node3
```

Parameters:

-c: this node will be added into “cluster”:

-zk: this node will be managed by the zookeeper server “hlnode3”

If we need to start another node, do the same process, but just change the “node1” to “nodeX”. How many nodes need to be start depends on the cluster parameter **-nbTasks=n**. When you create the cluster with n nodes, you need to start n nodes.

After we start all the necessary nodes, the SAMOA task “Prequential Evaluation” would start running.

For example, if we start 3 nodes, It will run “Partition 0” “Partition 1” “Partition 2” on these nodes. At last, on the node running Partition 0, it would output the final result and the total time spent for the task.

An output example:

```
15:53:19.616 [STREAM-1_PROCESSING-ITEM-2] INFO  c.y.l.s.e.EvaluatorProcessor -
com.yahoo.labs.samoa.evaluation.EvaluatorProcessorid = 0
evaluation instances,classified instances,classifications correct (percent),Kappa Statistic
(percent),Kappa Temporal Statistic (percent)
10000.0,10000.0,63.0,18.27226688655109,24.842575665244766
20000.0,20000.0,71.72500000000001,39.896180137068185,42.36061563551118
30000.0,30000.0,75.24333333333333,48.16209849764309,49.32105083589218
```

```
40000.0,40000.0,77.71000000000001,53.682114130964365,54.20882337835755
50000.0,50000.0,79.258,57.159571407151745,57.350824526051724
60000.0,60000.0,80.47999999999999,59.78171783757047,59.857417055113785
70000.0,70000.0,81.44571428571429,61.83842950320394,61.85945437993716
80000.0,80000.0,82.17125,63.38304943338002,63.30777937847293
90000.0,90000.0,82.70111111111112,64.49010463360504,64.41127391592568
100000.0,100000.0,83.191,65.48988889098432,65.41998395358885

15:53:19.624 [STREAM-1_PROCESSING-ITEM-2] INFO    c.y.l.s.e.EvaluatorProcessor - total
evaluation time: 388 seconds for 100000 instances
```

From the last line of the output, we can know the total run time of the test.

Experiment Result

The tables below shows the test configurations and the results. **In each cell is the running time (seconds) of the task.**

Test 1 Small amount data

Input instance: 100,000 (-i)
Attributes: 10 nominal attributes (-o), 10 numerical attributes(-u)
Output information level: **DEBUG** (In DEBUG level, it output a lot of debug information, and slow down the performance, so in the following tests I turn the level to INFO level)

1 node, local mode

Node1	Node2	Node3

note: did not test local mode

n node(s), s4 mode

Node1	Node2	Node3	Node 2,3	Node 1,3	Node 1,2,3
23	27	327	23	21	24
63	49	108	21	29	45
132	54	110	19	20	47
36	158	110	20	19	24
23	53	107	19	29	48
161		112			
135					
155					
Avg=91	Avg=68.2	Avg=145	Avg=**20.4**	Avg=23.6	Avg=**37.6**

Conclusion:

- Running task on two nodes is faster on run task on only one node. On the other side, run task on three nodes is slower than run on two nodes.
- When attributes or instances are not high, the cost of coordinating the cluster exceeds the

efficiency of parallelism.

Note: This test 1 output a lot of DEBUG information to the terminal and it takes lot of time, so the **performance result is not proper**. So later I repeated this test with the "info output level = INFO", as shown below.

Input instance: 100,000 (-i)
Attributes: 10 nominal attributes (-o), 10 numerical attributes(-u)
Output information level: **INFO** (In INFO level, only few important info are output, so the time measure can be more precise)

1 node, local mode

Node1	Node2	Node3
12	12	6
11	12	6
11	12	6
11	12	6
11		6
Avg=11.2	Avg=12	Avg=6

note: did not test local mode

n node(s), s4 mode

Node1	Node2	Node3	Node 1,2	Node 2,3	Node 1,3	Node 1,2,3
124	52	25	36(1,2)	30(2,3)	30(1,3)	39(2,3,1)
132	49	61	36(2,1)	18(3,2)	34(1,3)	40(1,3,2)
43	24	69	34(1,2)	21(3,2)	18(3,1)	25(3,2,1)
117	118	24			18(3,1)	25(3,1,2)
Avg=104	Avg=60.75	Avg=44.75	Avg=35.33	Avg=23	Avg=**25**	Avg=**32.25**

Note:
The number in parenthesis is the start order of nodes.

Test 2-1 Medium amount data(much instances)

Input instance: 500,000(-i)
Attributes: 10 nominal attributes(-o), 10 numerical attributes(-u)
Output information level: **INFO**. (This level only output important information)

1 node, local mode

Node1	Node2	Node3
	57	31
	58	31
	53	31
	52	
	52	
	52	
	Avg=54	Avg=**31**

Note: Node1 should has similar performance as node 2, so I did not test it.

n node(s), s4 mode

Node1	Node2	Node3	Node 2,3	Node 1,2	Node 1,2,3
	657	330	68 (3,2)	125	145(2,3,1)
	515	327	68 (3,2)	223	155(2,1,3)
	659	325	**119 (2,3)**	130	75 (3,1,2)
				118	80 (3,2,1)
				123	80 (3,2,1)
				131	
	Avg=610	Avg=**327**	Avg=**85**	Avg=142	Avg=**78.2**

Note:
The number in parenthesis is the start order of nodes.
Node1 should has similar performance as node 2, so I did not test it.

Conclusion:

- Run task in s4 mode, even though it is paralleled into 2 or 3 nodes, the process speed is slower than run it on single node in local mode.
- The performance of 3 nodes is still a little worse than 2 nodes.
- The start order of the nodes is important. When I run task on nodes 2,3, if I start node 2 first, the total run time is longer (see bold text in the table above). Because the first node started will be assigned “Partition 0”, which run the aggregation PI except the local statistic

PI. If we assign the better computer,node 3, to partition 0, if will do faster aggregation.

Form this “Test 2-1”, I realized the “Medium amount data” is not large enough. To exploit the advance of “Vertical Parallelism”, I should make the input data with more attributes. So I increase the attributes numbers of input data.

Test 2-2 Medium amount data (much attributes)

Input instance: 100,000(-i)
Attributes: 50 nominal attributes(-o), 50 numerical attributes(-u)
Output information level: info

1 node, local mode

Node1	Node2	Node3
42	43	23
42	42	22
42	42	22
Avg=42	Avg=42	Avg=**22**

n node(s), s4 mode

Node1	Node2	Node3	Node 1,2	Node 1,3	Node 2,3	Node 1,2,3
	797	163	27(1,2)	27(1,3)	17(3,2)	36(3,1,2)
			38(1,2)	15(3,1)	16(3,2)	39(3,1,2)
			37(2,1)	17(3,1)	17(3,2)	35(3,1,2)
			42(2,1)	19(3,1)	28(2,3)	36(3,1,2)
					29(2,3)	61(2,3,1)
						73(2,1,3)
						80(2,1,3)
						76(1,3,2)
	Avg=797	Avg=**163**	Avg=36	Avg=**19.5**	Avg=21.4	Avg=**54.5**

Note: The number in parenthesis is the start order of nodes.
Node 1 should has similar performance as node 2.
I only test one time for “node2”,”node3” in s4 mode, because they take long time and they are not important.

Conclusion:

- With medium dataset (more attributes), the speed of running task on two nodes(19 s) exceeds the speed of single node in local mode(22 s). However, the performance improvement is not significant.
- However, running task on three nodes is still slower than running on two nodes

Now it is surely that when I use bigger data set, run classification on two nodes improve the performance than single node, but it is strange that three nodes is slower than two nodes.
I don't know the reason is what, they maybe:
*SAMOA's problem
*Algorithm's problem
*Node is not enough
*or data is not enough big?

In the future test I will try to test with bigger data and more nodes.
I will also try to use different task, such as Sun's movie review classification, and clustering algorithm.

(Future works)

Test 3 Big amount data

Test 4 Higher parallel level (4 nodes)