11-791 Design and Engineering of Intelligence Information System Homework 3

Execution Architecture with CPE and Deployment Architecture with UIMA-AS

Lab Report

Andrew id: xchu
email: xchu@cs.cmu.edu

1 General Description

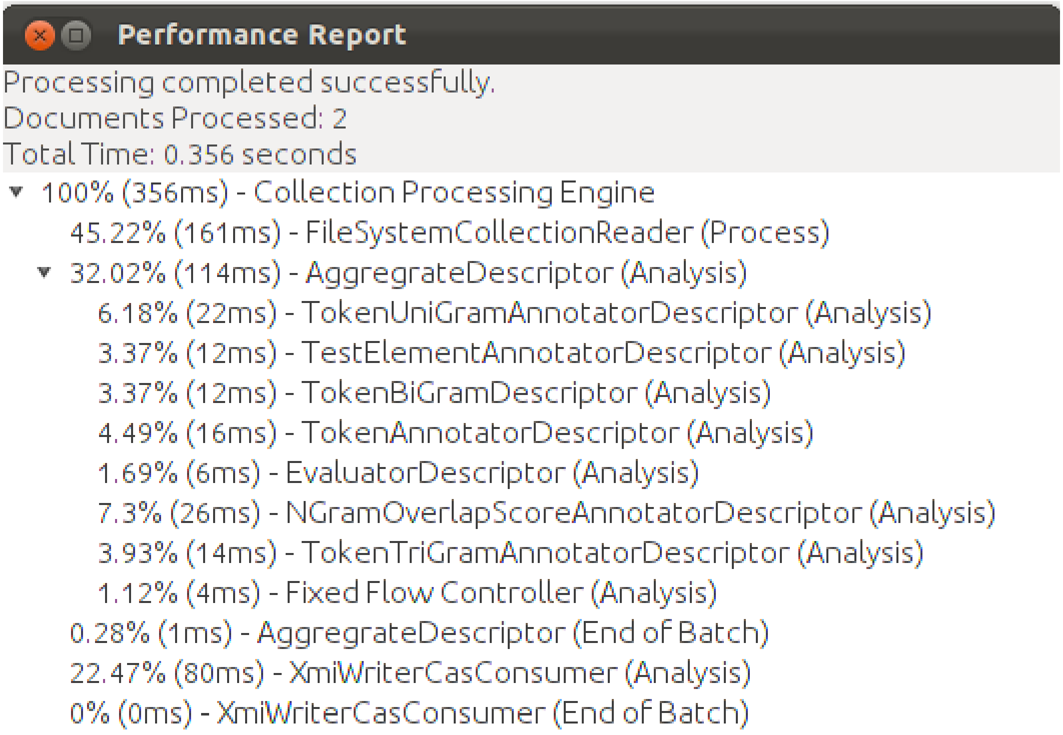
In this lab, we mainly excuted the popleline from howework 2 with a CPE and deploy it as a service with UIMA-AS. Following the UIMA collection process engine developer's guide and the Apache UIMA asynchronous scaleout, we first creating the CPE descriptor based on homework 2 pipeline. Then, by designing the UIMA-AS client descriptor, we can use the remote UIMA-AS service and integrate with our own CPE pipeline. Then, I deploy the aggregate analysis engine and Stanford CoreNLP in my own machine and called the service locally.

2 Experiment

Task 1

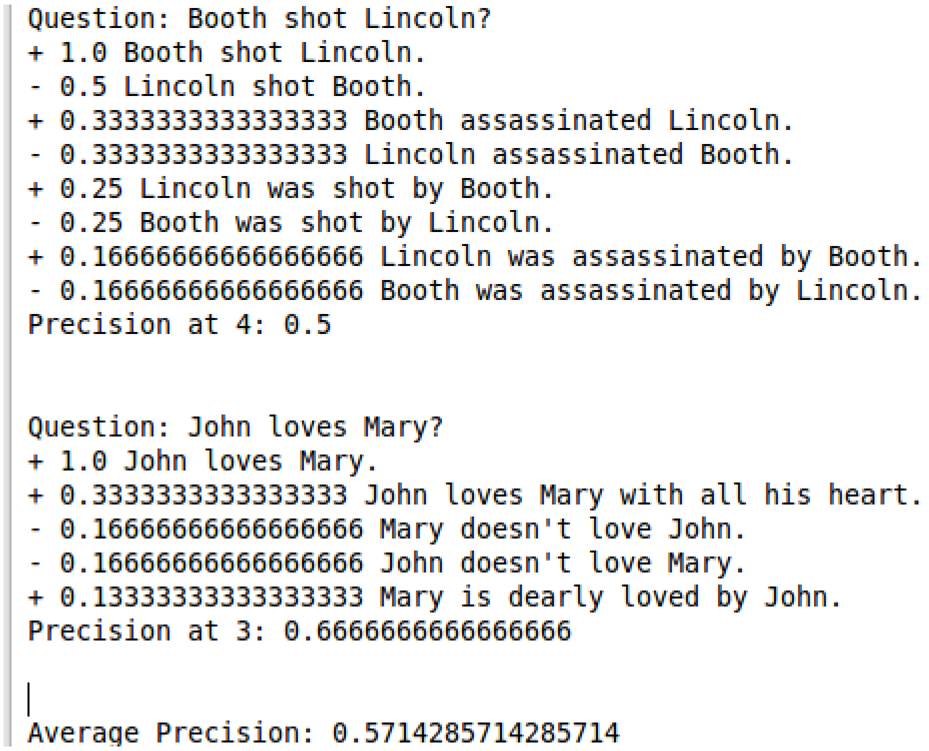
Task 1.2

by using the CPE GUI, this task is somewhat easy to implement. All we need to do is to design a collection reader and a cas consumer. I have created both of them in java code in the edu.cmu.deiis.cpe package. After this step, we need to create collectionreader descriptor as well as casconsumer descriptor. Then by running the CPE GUI, we can specify the collection reader and consumer as well as the analysis engine in homework 2, then we can get the CPE file. Also, the execution result of this task is:



```
Processing completed successfully.
Documents Processed: 2
Total Time: 0.356 seconds
▼ 100% (356ms) - Collection Processing Engine
    45.22% (161ms) - FileSystemCollectionReader (Process)
  ▼ 32.02% (114ms) - AggregrateDescriptor (Analysis)
      6.18% (22ms) - TokenUniGramAnnotatorDescriptor (Analysis)
      3.37% (12ms) - TestElementAnnotatorDescriptor (Analysis)
      3.37% (12ms) - TokenBiGramDescriptor (Analysis)
      4.49% (16ms) - TokenAnnotatorDescriptor (Analysis)
      1.69% (6ms) - EvaluatorDescriptor (Analysis)
      7.3% (26ms) - NGramOverlapScoreAnnotatorDescriptor (Analysis)
      3.93% (14ms) - TokenTriGramAnnotatorDescriptor (Analysis)
      1.12% (4ms) - Fixed Flow Controller (Analysis)
    0.28% (1ms) - AggregrateDescriptor (End of Batch)
    22.47% (80ms) - XmiWriterCasConsumer (Analysis)
    0% (0ms) - XmiWriterCasConsumer (End of Batch)
```

and the analysis engine gives the following result:

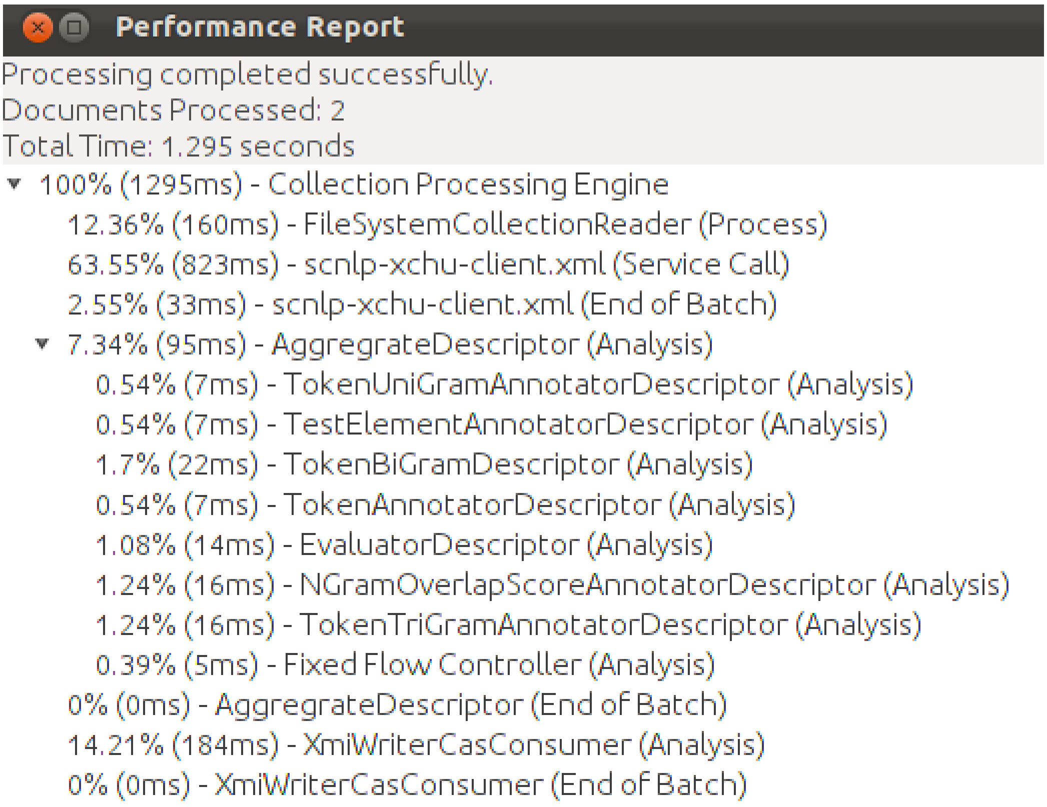


```
Question: Booth shot Lincoln?
+ 1.0 Booth shot Lincoln.
- 0.5 Lincoln shot Booth.
+ 0.3333333333333333 Booth assassinated Lincoln.
- 0.3333333333333333 Lincoln assassinated Booth.
+ 0.25 Lincoln was shot by Booth.
- 0.25 Booth was shot by Lincoln.
+ 0.16666666666666666 Lincoln was assassinated by Booth.
- 0.16666666666666666 Booth was assassinated by Lincoln.
Precision at 4: 0.5


Question: John loves Mary?
+ 1.0 John loves Mary.
+ 0.3333333333333333 John loves Mary with all his heart.
- 0.16666666666666666 Mary doesn't love John.
- 0.16666666666666666 John doesn't love Mary.
+ 0.13333333333333333 Mary is dearly loved by John.
Precision at 3: 0.6666666666666666


Average Precision: 0.5714285714285714
```

Task 2

Task 2.2

In this task, we use the UIMA-AS to integrate a remote UIMA-AS service(stanford CoreNLP) into our CPE pipeline. First of all, we have to create an UIMA-AS client descriptor for a remote UIMA-AS service. In order to call the remote service, we have to add the brokerURL: tcp://mu.lti.cs.cmu.edu:61616 and the endpoint scnlpQueue. After these steps, we should start the activeMQ broker and specify them in the CPE GUI. Then we can get the result for this task:

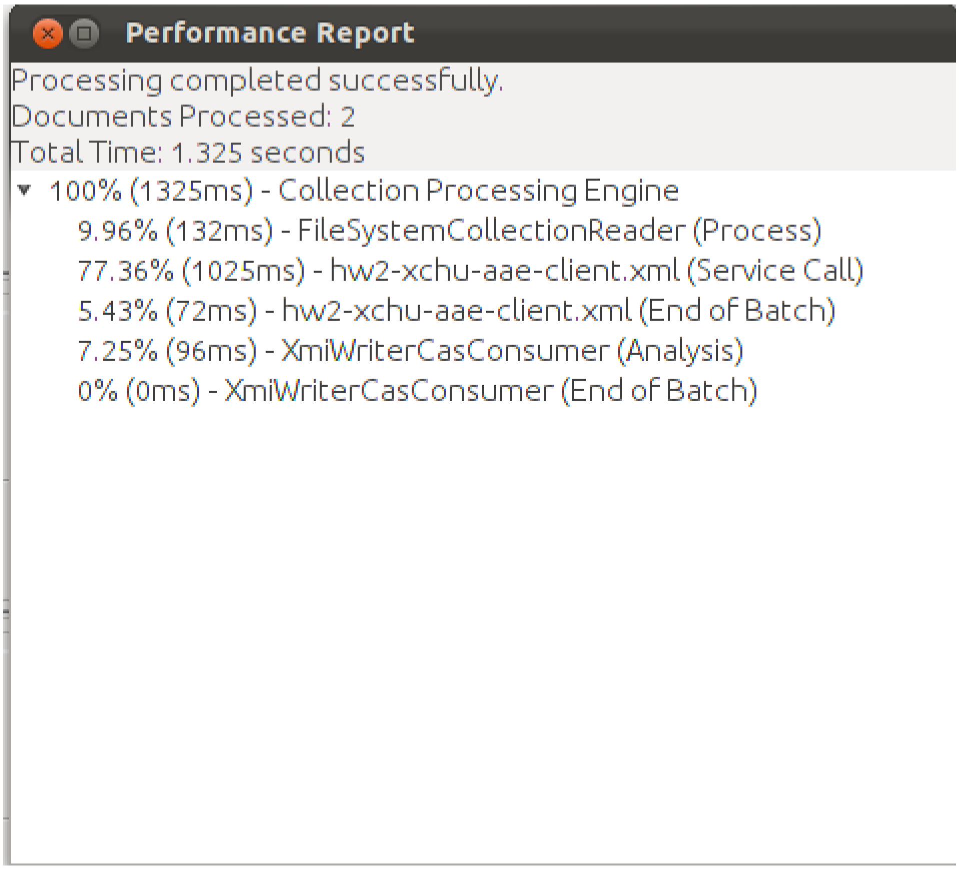


```
Processing completed successfully.
Documents Processed: 2
Total Time: 1.295 seconds
▼ 100% (1295ms) - Collection Processing Engine
    12.36% (160ms) - FileSystemCollectionReader (Process)
    63.55% (823ms) - scnlp-xchu-client.xml (Service Call)
    2.55% (33ms) - scnlp-xchu-client.xml (End of Batch)
  ▼ 7.34% (95ms) - AggregrateDescriptor (Analysis)
      0.54% (7ms) - TokenUniGramAnnotatorDescriptor (Analysis)
      0.54% (7ms) - TestElementAnnotatorDescriptor (Analysis)
      1.7% (22ms) - TokenBiGramDescriptor (Analysis)
      0.54% (7ms) - TokenAnnotatorDescriptor (Analysis)
      1.08% (14ms) - EvaluatorDescriptor (Analysis)
      1.24% (16ms) - NGramOverlapScoreAnnotatorDescriptor (Analysis)
      1.24% (16ms) - TokenTriGramAnnotatorDescriptor (Analysis)
      0.39% (5ms) - Fixed Flow Controller (Analysis)
    0% (0ms) - AggregrateDescriptor (End of Batch)
    14.21% (184ms) - XmiWriterCasConsumer (Analysis)
    0% (0ms) - XmiWriterCasConsumer (End of Batch)
```

As for the performance in time, we can clearly see that by calling the remote service, it will cost more time.

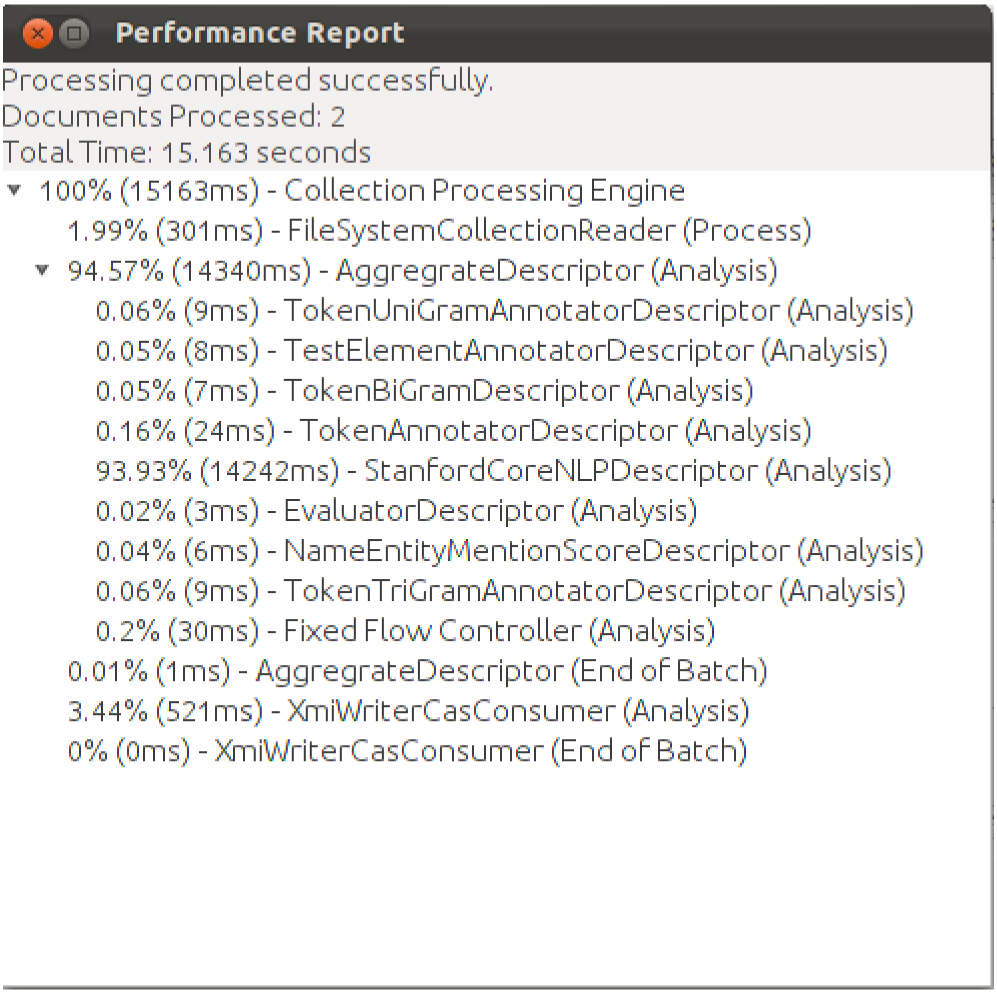
Task 2.3:

In this task, we have to deploy the aggregate analysis engine in homework 2 and call the service locally. This step is much similar to the task in 2.2 except that we should modify the broker URL and the endpoint. The result for this part is:



```
Processing completed successfully.
Documents Processed: 2
Total Time: 1.325 seconds
▾ 100% (1325ms) - Collection Processing Engine
    9.96% (132ms) - FileSystemCollectionReader (Process)
    77.36% (1025ms) - hw2-xchu-aae-client.xml (Service Call)
    5.43% (72ms) - hw2-xchu-aae-client.xml (End of Batch)
    7.25% (96ms) - XmiWriterCasConsumer (Analysis)
    0% (0ms) - XmiWriterCasConsumer (End of Batch)
```

bonus part:
For the bonus part, I implemented the Stanford CoreNLP annotator in my own machine. Using the aggregate analysis engine descriptor and the CPE GUI, we can get the following result:



```
Processing completed successfully.
Documents Processed: 2
Total Time: 15.163 seconds
▼ 100% (15163ms) - Collection Processing Engine
    1.99% (301ms) - FileSystemCollectionReader (Process)
  ▼ 94.57% (14340ms) - AggregrateDescriptor (Analysis)
      0.06% (9ms) - TokenUniGramAnnotatorDescriptor (Analysis)
      0.05% (8ms) - TestElementAnnotatorDescriptor (Analysis)
      0.05% (7ms) - TokenBiGramDescriptor (Analysis)
      0.16% (24ms) - TokenAnnotatorDescriptor (Analysis)
      93.93% (14242ms) - StanfordCoreNLPDescriptor (Analysis)
      0.02% (3ms) - EvaluatorDescriptor (Analysis)
      0.04% (6ms) - NameEntityMentionScoreDescriptor (Analysis)
      0.06% (9ms) - TokenTriGramAnnotatorDescriptor (Analysis)
      0.2% (30ms) - Fixed Flow Controller (Analysis)
    0.01% (1ms) - AggregrateDescriptor (End of Batch)
    3.44% (521ms) - XmiWriterCasConsumer (Analysis)
    0% (0ms) - XmiWriterCasConsumer (End of Batch)
```

obviously, we can see that running the Stanford CoreNLP annotator is much slower in our own machine. The reason is that the program costs a lot of time to load the classifier in my computer.

3 conclusion

From this homework, we have used the CPE to excuted the pipeline for the homework 2 and deploy it as a service with UIMA-AS by calling the remote service and locally. We know that it is very convenient for us just to design the collection reader, analysis engine and consumers in order to implement all the service.