

Cloudia – A MapReduce Framework for Cloud Computing

Especificação

Seguem as User Stories do projeto, com seu estado de implementação:

Como usuário, quero fazer login com minhas credenciais do AWS

Critérios:

- Usuário entrar com sua access key;
- Usuário entrar com sua secret key;
- Os dados ficarem persistidos em um cookie apenas;
- Sem armazenamento permanente destes dados;
- Opção de "desconectar", limpando o cookie.

Implementado

Como usuário, quero iniciar um grupo de execução

Critérios:

- Usuário indica nome do grupo;
- Usuário indica quantas máquinas quer no grupo;
- O sistema abstrai quais são as instâncias iniciadas, mantendo controle de quem elas são;
- O sistema indica o estado do grupo - se as instâncias estão iniciando, paradas, em execução etc.

Implementado

Como usuário, quero encerrar um grupo de execução

Critérios:

- O usuário não precisa parar as instâncias manualmente, abstraindo este aspecto do AWS;
- O sistema corre todas as instâncias do grupo e para elas completamente independente do estado em que estiverem;

Implementado

Como usuário, quero executar a etapa de map num caso simples de exemplo

Critérios:

- O usuário tem acesso a página do grupo de execução;
- A opção de map é acionada
- Como exemplo, o sistema automaticamente usa um bucket em específico - word_count;
- O gerenciador assegura que as instâncias estão preparadas para execução, aplicando a rotinas básica nas instâncias através do fabric, que ao final inicia o webserver remoto;
- Uma das instâncias é escolhida e instruída através da chamada REST ao webserver remoto para fazer a etapa de map;

- Como exemplo é usado o código básico de map do WordCount - cada arquivo no bucket vira uma mensagem no SQS para os workers futuramente processarem;

Implementado

Como usuário, quero processar os dados num caso simples de exemplo

Critérios:

- O usuário aciona a etapa de processamento;
- O gerenciador instrui todas as instâncias através do webserver remoto a executarem a etapa de processamento;
- A camada de modelo do webserver remoto é um worker que vai buscar uma mensagem da fila de trabalho no SQS e, para cada mensagem, chama o código de processamento do WordCount;
- O código de exemplo para o processamento básico do wordcount corre as linhas e palavras do documento especificado na mensagem e retorna o resultado;
- O worker armazena o resultado num array em memória;
- Quando a fila esvaziar, o worker encerra;

Implementado

Como usuário, quero executar a etapa de reduce num caso simples de exemplo

Critérios:

- O usuário aciona a etapa de reduce;
- O gerenciador instrui uma instância através do webserver remoto a executar a etapa de reduce;
- A camada de modelo do webserver remoto busca a lista de resultados de cada instância através dos seus webserver remotos;
- O código de exemplo para o processamento básico soma todos os valores de todos os arrays;
- O gerenciador exibe o resultado, que é o retorno da chamada feita na instrução de reduce;

Em andamento

Como usuário, quero escolher quais dados quero que sejam processados

Critérios:

- O usuário pode escolher qual é o bucket que quer acessar;
- Opcionalmente, ele pode escolher se é um bucket ou um arquivo, indicando um path S3;
- O sistema usa o path indicado, descompactando o arquivo indicado, ou listando todos os arquivos da pasta indicada e passando para a etapa de map;

Não iniciado

Como usuário, quero informar qual é o código específico do meu MapReduce

Critérios:

- O usuário sobe um arquivo que implemente a classe MapReducer do pacote da aplicação;
- O usuário deve implementar os métodos map, process e reduce, que terão como entradas os arquivos indicados no bucket e depois as mensagens indicadas pela etapa de map;

Não iniciado

Como usuário, quero informar quais etapas de preparação de cada máquina do grupo são necessárias

Critérios:

- O usuário sobe junto com o arquivo e o bucket um script fabric para ser executado na preparação de cada instância;

Não iniciado

Como desenvolvedor, quero criar exemplos com um algoritmo de ML

Critérios:

- Implementar o algoritmo do artigo Double Dip Map Reduce como duas tarefa de map reduce usando o sistema;

Não iniciado

Estória 11: Como desenvolvedor, quero criar exemplos com um algoritmo de OCR

Critérios:

- Implementar um esquema de OCR em massa usando o sistema;

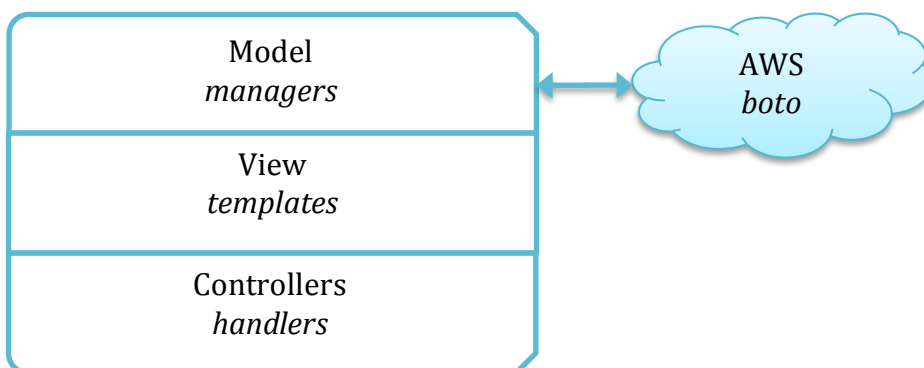
Não iniciado

Projeto

A aplicação se dividirá em duas partes:

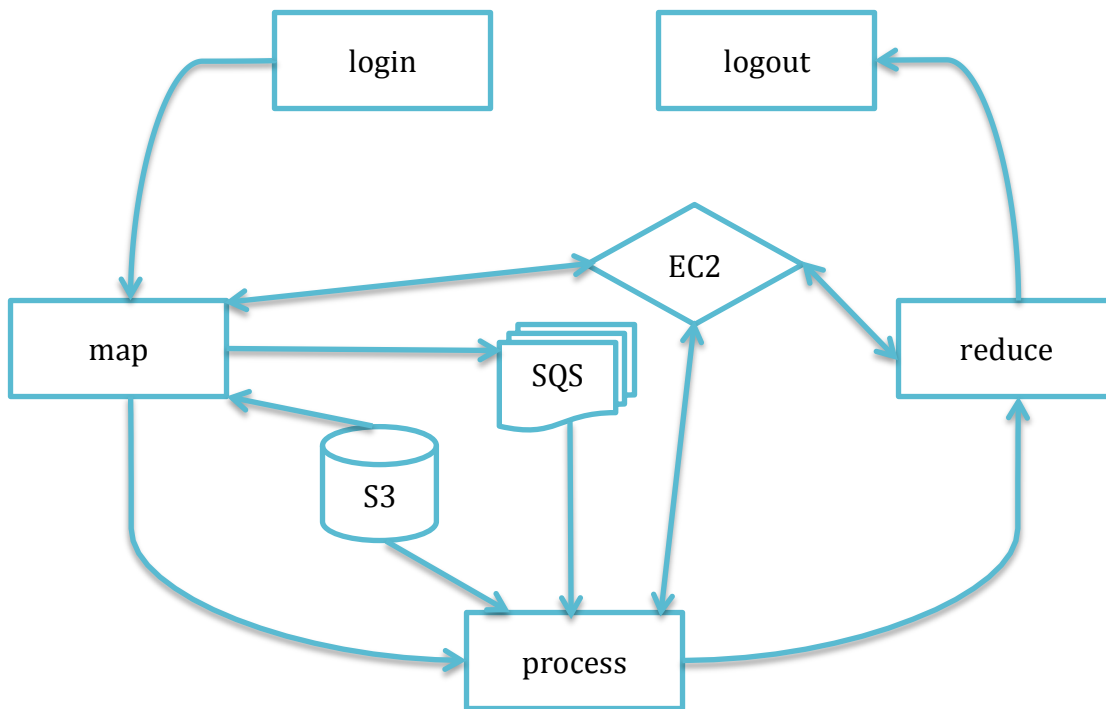
1 – Um servidor web local com a interface gestão a partir do computador do usuário;

Este servidor segue a arquitetura MVC, separando as camadas de modelo - responsável por abstrair o modelo do AWS para a aplicação - das camadas de view e controllers.



2 – Um servidor web remoto para ser instalado em cada instância do grupo de execução;

O servidor web receberá os comandos do sistema de gestão para usar o processamento no EC2 nas etapas do MapReduce



A gestão do MapReduce é feita usando o paradigma de fila de trabalho e de pool de workers - cada instância no EC2 é um worker que vai, após instruído para tal, buscar um pedaço do trabalho na fila do SQS. A etapa de map portanto mapeia os dados iniciais em mensagens na fila de trabalho, e o reduce vai em cada worker buscar os resultados armazenados em memória ao final do processo.

Tecnologias

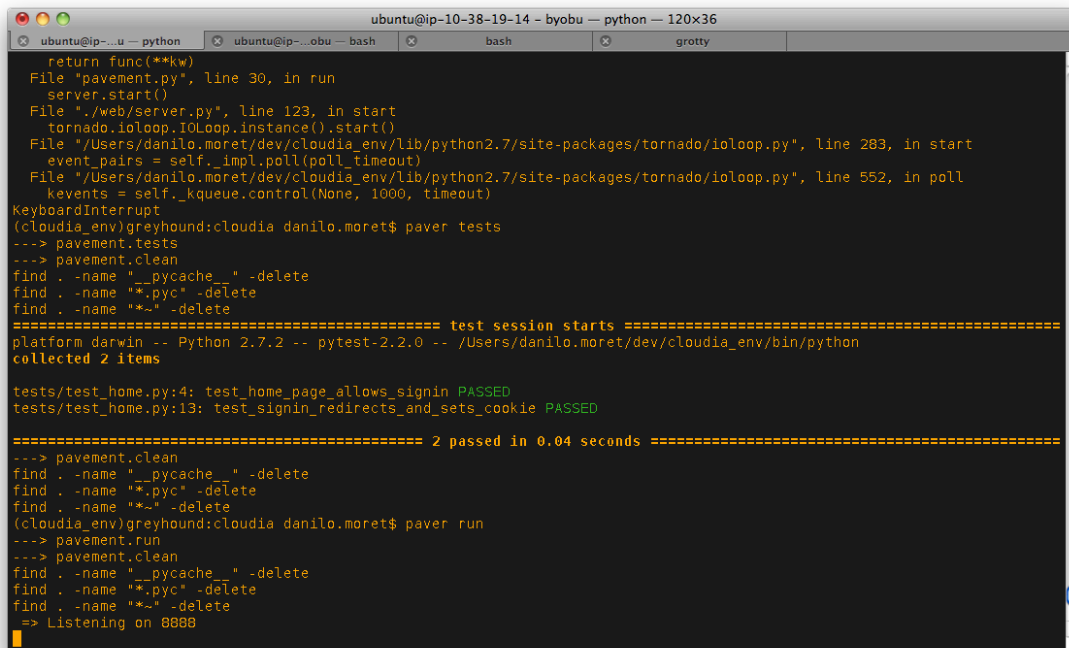
Foi escolhida a linguagem Python pela familiaridade do autor. Além disso, diversas bibliotecas e ferramentas se encontram disponíveis para Python que tomam parte nas funções do sistema e facilitam o projeto:

boto: uma camada de acesso a API do AWS para realização das tarefas de baixo nível dos serviços

fabric: um sistema de automação para configurações de sistemas, possibilitando uma solução nativa que permita usar uma imagem padrão do EC2 sem customizações e prepara-la para a execução do MapReduce sempre da mesma forma

Testes e documentação

Os testes são testes automatizados, que documentam as funções do sistema. Seguem exemplos da execução do sistema e dos testes:

A terminal window titled 'ubuntu@ip-10-38-19-14 - byobu -- python -- 120x36' showing a series of commands and their outputs. The user is in a 'pavement' environment. They run 'paver tests', which triggers a cleanup of pycache files and then runs a pytest session. The pytest session shows two tests passing: 'test_home_page_allows_signin' and 'test_signin_redirects_and_sets_cookie'. After the tests, the user runs 'paver run', which again triggers a cleanup and then starts a server listening on port 8888.

```
return func(**kw)
File "pavement.py", line 30, in run
server.start()
File "../web/server.py", line 123, in start
tornado.ioloop.IOLoop.instance().start()
File "/Users/danilo.moreto/dev/cloudia_env/lib/python2.7/site-packages/tornado/ioloop.py", line 283, in start
event_pairs = self.impl.poll(poll_timeout)
File "/Users/danilo.moreto/dev/cloudia_env/lib/python2.7/site-packages/tornado/ioloop.py", line 552, in poll
kevents = self._queue.control(None, 1000, timeout)
KeyboardInterrupt
(cloudia_env)greyhound:cloudia danilo.moreto$ paver tests
--> pavement.tests
--> pavement.clean
find . -name "__pycache__" -delete
find . -name "*.pyc" -delete
find . -name "*~" -delete
===== test session starts =====
platform darwin -- Python 2.7.2 -- pytest-2.2.0 -- /Users/danilo.moreto/dev/cloudia_env/bin/python
collected 2 items

tests/test_home.py:4: test_home_page_allows_signin PASSED
tests/test_home.py:13: test_signin_redirects_and_sets_cookie PASSED

===== 2 passed in 0.04 seconds =====
--> pavement.clean
find . -name "__pycache__" -delete
find . -name "*.pyc" -delete
find . -name "*~" -delete
(cloudia_env)greyhound:cloudia danilo.moreto$ paver run
--> pavement.run
--> pavement.clean
find . -name "__pycache__" -delete
find . -name "*.pyc" -delete
find . -name "*~" -delete
=> Listening on 8888
```

