Librario

Document searching service

Members

Sippakorn Widsankun	5510500140
Aikdanai Sidhikosol	5610500079
Kanitkorn Sujautra	5610500311
Padungkiet Tamasee	5610501008
Jitlada Supreeyaporn	5610503841
Wassapon Watanakeesuntorn	5610503981

Architecture design
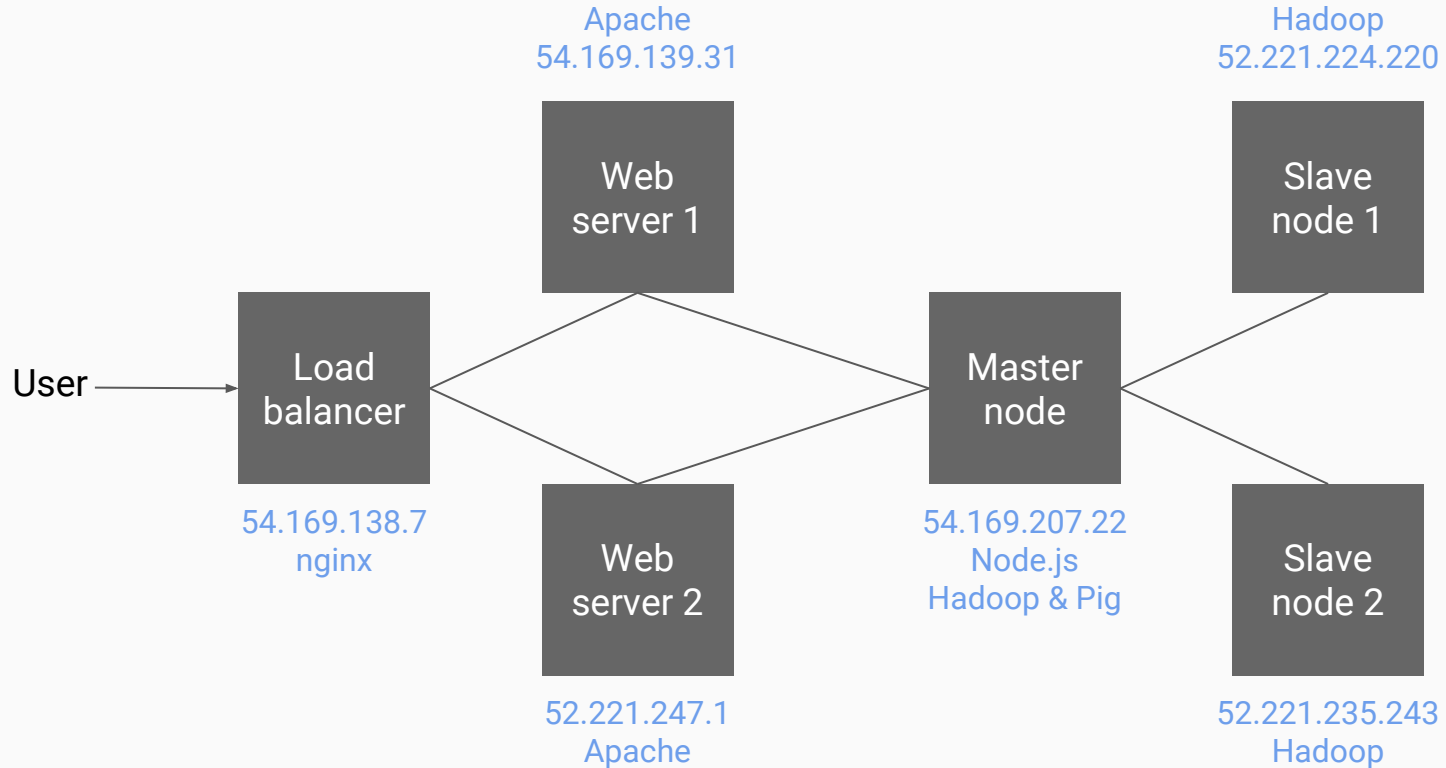
Javascript (on Web server)

```
1 main.js                                                                    X
 function senddata() {
 |    var str = formatString($('#paragraph').val());
 |    var jsondata = JSON.stringify({ "data": str });
 |    console.log(jsondata);
 |    $.ajax({
 2        type: "POST",
 2        url: "http://54.169.207.22:3000",
 2        data: jsondata,
 2        contentType: 'application/json',
 2        cache: false,
 +        beforeSend: function(xhr) {...}--------------------------------------
 +        success: function(reply) {...}--------------------------------------
 +        error: function(xhr, ajaxOptions, thrownError) {...}---------------
 2
 2    });
 |}

 function formatString(str) {
 |    str = str.trim();
 |    str = str.replace("\t", "");
 |    str = str.replace(/(?:\r\n|\r|\n)/g, " ");
 |    str = str.replace(/[ ]+/g, " ");
 |    // str = str.replace(/[.!?][ ]+/g, "\n");
 |    var strings = str.match(/((Jr\.|Mr\.|Ms\.|Mrs\.|Dr\.|[A-Z]\.|(\.\w+)+|\.\w\.|[0-9]*\.[0-9]+)*[^.!?]+?)+[.!?]+/gm );
 -    if (strings != null) {
 2        str = strings.join("\n");
 2    }
 |    // str = str.replace(/\[.!?]$/, "");
 |    return str;
 |}

 function animateWait() {
 |    $('#submitbtn').html('Searching <i class=\"fa fa-spinner fa-pulse fa-fw\"></i>');
 |    $('#submitbtn').attr("disabled", true);
 |}

 function btnReady() {
 |    $('#submitbtn').html('Submit');
 |    $('#submitbtn').attr("disabled", false);
 |}

 +function renderResult(json) {...}--------------------------------------------
 ~
 ~
 ~
 NORMAL   main.js                                                  69%    50:1
"js/main.js" 72L, 2121C
```
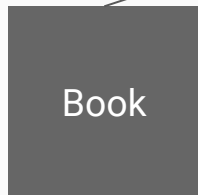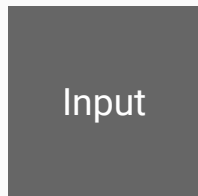
Hadoop endpoint

```javascript
app.post('/', function (req, res) {
  const timestamp = new Date().valueOf()
  const data = req.body.data

  fs.writeFile(`input/message_${timestamp}`, data, function (err) {
    if (err) throw err
    exec(`hadoop fs -put input/message_${timestamp}`, function (err, stdout, stderr) {
      if (err) {
        res.status(300).send(data)
      } else {
        exec(`pig -param input=message_${timestamp} ~/project/main.pig`, function (err, stdout, stderr) {
          if (!err) {
            const results = stdout
              .split('\n')
              .map(function (result) {
                return {
                  'line': result.substring(result.indexOf(',') + 1, result.length - 1),
                  'name': result.substring(1, result.indexOf(','))
                }
              })
            res.status(200).json({'results': results})
          } else {
            res.status(300).send(data)
          }
        })
      }
```
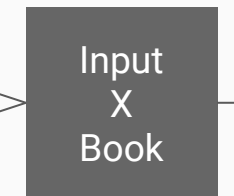
Preprocess for Books

- Reformat each book into one line.
- Remove '\n', '\r', '\t' and multiple instances of ' '
- Add file name in front of line separated by '\t'
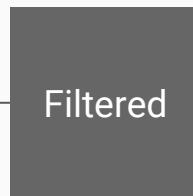
Pig Design
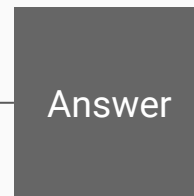
Sentences separated
by newline

Input

Input
X
Book

Cross operation

Filtered

Filtered by
Jython UDFs

Answer

(book , sentences)

Book

preprocessed

Pig

```
GNU nano 2.2.6                    File: main.pig                              Modified

REGISTER '/home/ubuntu/project/udfs.py' USING jython AS myudfs;

B = load 'book1line' USING PigStorage('\t') as  (name:chararray,content:chararray);
I = load '$input' as text:chararray;
crossed = CROSS B, I;
gen_crossed = FOREACH crossed GENERATE B::name as book, I::text as text, B::content as content;
filtered = FILTER gen_crossed BY myudfs.regex_find(content,text) == TRUE;

answer = FOREACH filtered GENERATE book,text;

dump answer;
```

Jython UDFs

```
GNU nano 2.2.6                    File: udfs.py

#!/usr/bin/python

@outputSchema('output_field_name:boolean')
def regex_find(content, regex):
    return regex in content
```

Pig output


```
(Introduction_and_Copyright, Such people always stay fresh in the hearts of those
ove for God is contagious.)
(Introduction_and_Copyright, One such godly man was M. D. Barefield, who learned t
he could make this word his own.)
(Introduction_and_Copyright, Without the goodness of their lives no one would beli
out a Bible.)
(Introduction_and_Copyright, And then there are those who have simply loved these
ghest ideals.)
(Introduction_and_Copyright, For this crime he was strangled and burned at the sta
(Introduction_and_Copyright, By the time of the Reformation, William Tyndale had o
ible available in English to the poorest farmhand in England.)
(Introduction_and_Copyright, Nameless monks sleep in the dust, who faithfully copi
n through the centuries.)
(Introduction_and_Copyright, Those who penned the autographs often faced ridicule
(Introduction_and_Copyright,Although it is not appropriate to dedicate a New Testa
or would like to give his gratitude to the many people who have labored to preserv
```

Librario

Document searching service