CSC5355

Project – Real time & batch access log processing

Up to 110 points to collect over 100 !!

Description of the project

In this project you are asked build a pipeline for real-time and batch processing of access to webserver pages (assumed to correspond to products). The pipeline should produce (N=3) most visited pages every (P=5minutes) period of time. N and P should be parameters that can be changed at every start of the pipeline. It should also allow for running a batch process for producing N most visited products/pages for an entire day. Refer to Batch Processing with Unix Tools pp.391-392 of the textbook

For this, build a cluster of webservers with pages /product1 /produvct2 … /product5 running behind an HTTP load balancer. Each webserver is the exact replica of the other webservers. Configure the webservers to forward page access logs to a Kafka broker. Configure 2 Kafka consumers as a fan-out cluster. The first consumer should write every log item into a wide table NoSQL database (Cassandra) while the other consumer produces/writes to the screen aggregates for the N most accessed products/pages every P minutes. The results of the aggregation should be written back into Cassandra (use a different wide-column table). For this you can either leverage the Kafka broker with another topic (PRODUCTS) or write into Cassandra directly.

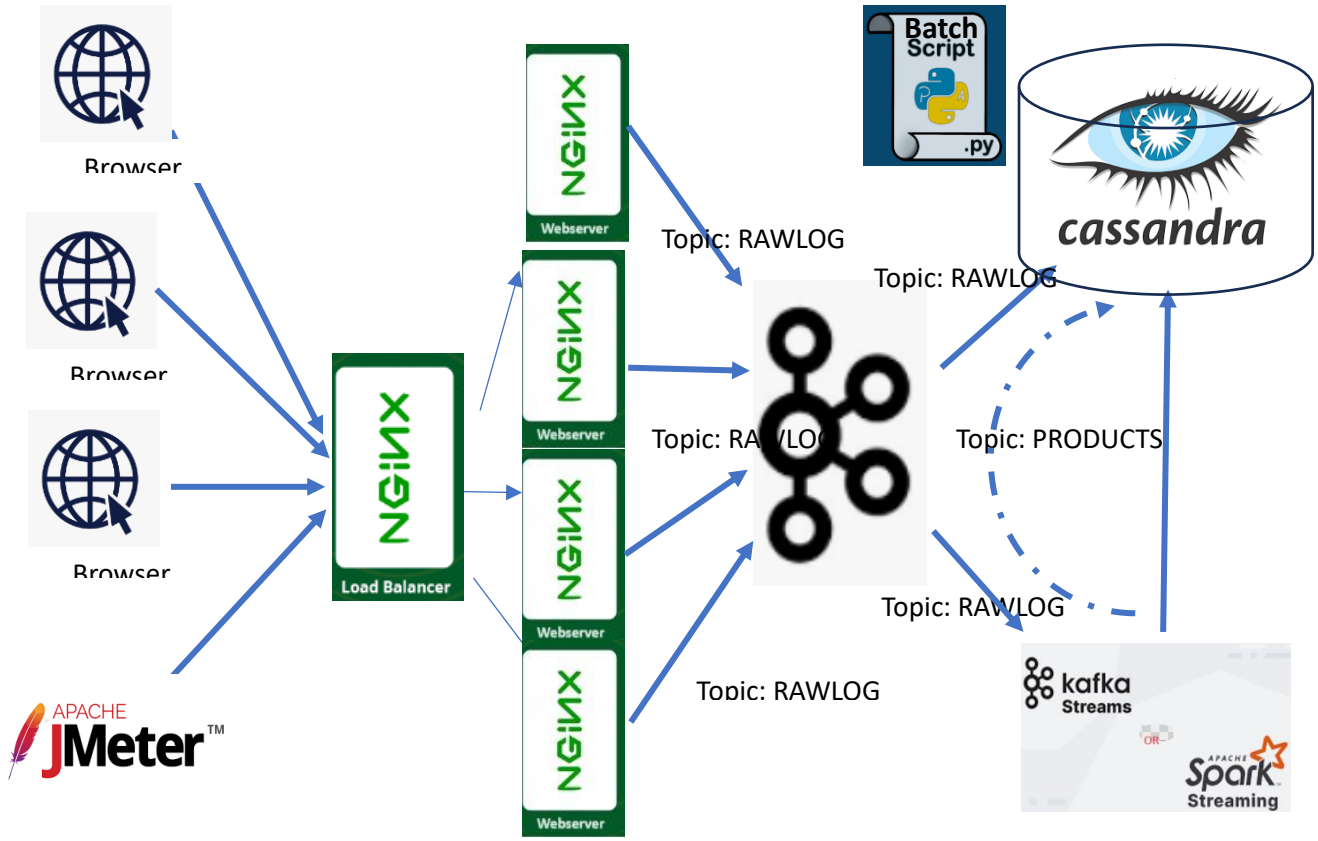
A batch process should be written and set to run ever end of the day at 8pm and should produce the N most visited pages/products for the **entire** day.

The focus in this project is on building a scalable pipeline **that permits batch and real-time processing of access logs.**

To test your application use Jmeter to produce large scale HTTP requests with the following characteristics:

Page	Percentage of requests
/products1	50%
/product2	25%
/product3	15%
/products4	8%
/product5	2%

Proposed Solution Architecture



Proposed pipeline for real-time and batch processing of access logs

Any departure from this architecture needs to be discussed with the professor and justified.

Implementation guidelines/hints

To implement the first version of the project, you will have (among other things) to:

1. Deploy a cluster of NGNIX webservers behind an NGNIX load balancer and configure Jmeter to produce the requests as set above. You can use dockized NGNIX
2. Deploy a Kafka broker cluster. You can use dockized Kafka broker here too.
3. Configure the NGNIX log format. Choose a minimalist JSON format. This will have impact on the design of the sink table (LOG) later.
4. Configure streaming of logs to Kafka (Topic RAWLOG). For this you need to look at NGNIX and Kafka Integration. Use Jmeter to ensure that logs arrive into the Kafka broker.
5. Install and deploy Cassandra.
6. **Design** and Create two wide column tables (LOG and RESULTS): one for the raw logs and one for the results of both batch and real-time processing
7. Configure a consumer pool in Fan -out mode. Use the first one to write the raw logs into Cassandra LOG table
8. For realtime processing of raw logs, use a stream processing engine you select (either Kafka Streams or Spark streaming) to produce the N most visited pages every

P period of time. If you opt for spark streaming engine use Structured streaming API. If you opt for Kafka Streams see Kafka Streams.

9. Write the results of the real time processing into screen and to Cassandra directly into RESULTS table or write them back into Kafka broker using the topic RESULTS then write a Kafka consumer that reads the topic RESULTS and write them into the table RESULTS.
10. Write a Batch program that processes the table LOG and produces the N most visited pages for the day and write the results into the table RESULTS. Configure a cron to start this batch process at 8pm every day.
11. Test everything using Jmeter

Report

The report should not exceed **10 pages** and should have the following sections:

"Project Team". Identify clearly the TEAM and the parts each member worked on. Use a table for this.

"Project definition", that include a brief description of the project goal(s), and the requirements (both functional and non-functional) including the requirements on the architecture.

"Project Design" focusing on key aspects in data intensive applications such as data models, encoding, querying, configurations for **partitioning and replication**, for NoSQL databases. In case a different architecture than the proposed one is used, it should be mentioned here and Justified. Links to concepts in the textbook.

"Project Implementation" Use a separate subsection for the deployment of each component that include other dependent components, key config parameters including clusters/replication/partitioning, libraries used, major code snippets, key data models and encoding if any, and unit tests. Use a subsection for test dataset preparation, unit tests, and integration tests.

"Implementation Limitations" about project limitations (parts not implemented, performance problems, bugs, eventual hardcoded data, etc.)

"Project demo and codebase" contains a link to a YouTube video for the demo. The codebase should be compressed and submitted with the report.

"References" not only to components, but also to key concepts from the textbook.

Demonstration

A 10-15mn or shorter Fullscreen video recording demonstrating successful installation of components, and successful execution of the project. The video should show the date, time of the day, machine names, and shell prompts, where the project is deployed.

Credits

To receive credit, your submission needs to be on time, of graduate coursework quality, and meet the Submission Guidelines in the course syllabus). The grade will be assigned according to the rubric below.

The following rubric will be used for grading.

Grading Rubric

Level of Achievement						
		Excellent (80)	**Good (70)**	**Average (60)**	**Poor (50)**	**Score**
A	**Working project**	All parts implemented with scalability, reliability features with **automation** as proven by the video recording	All parts implemented with scalability, reliability features but **without automation** as proven by the video recording	Many parts implemented but without scalability, reliability, and automation features as proven by the video recording	Project is not working	
B	**Report**	**Excellent (15)**	**Good (12)**	**Average (10)**	**Poor (6)**	**Score**
		Significant attention to concepts (data models, scalability, encoding, etc.)	Fair attention to concepts (data models, scalability, encoding, etc.)	**Some attention to** concepts (data models, scalability, encoding, etc.)	No attention to concepts (data models, scalability, encoding, etc.)	
C	**Ease of deployment**	**Excellent (5)**	**Good (4)**	**Average (3)**	**Poor (2)**	**Score**
		Significant attention to description of prerequisites for deployment (A README file gives deployment details)	Fair attention to description of deployment steps	**Some** attention to description of prerequisites for deployment	No attention to description of prerequisites for deployment	
D	**Video Presentation**	**Excellent (10)**	**Good (8)**	**Average (6)**	**Poor (4)**	**Score**
		• The content of presentation the is appropriate and well delivered. • clear image and voice with good spoken language	• The content of the presentations is appropriate but not well delivered. • Unclear image or voice.	Poor delivery	• Very poor delivery	