PROJECT IDEA:

The idea behind the project is to take a streaming feed and store all of the data points in a mock distributed database, while at the same time processing a random sample of the data points to estimate the features (mean and variance) of the data set. It would be based on a main activeMQ queue, many of the EIP's in camel, and then a secondary queue that would process a randomly assigned subset of the data and send it to a proxy for an application.

The data would originally come from standard in, a file, or possibly a real-life stream of data (from an ftp type site?) if it is easy enough to find and attach that. It would be some sort of data that was time-stamped, but still had a logical grouping. (Like I said, real life data like that would be nice, but if it is hard to find I could simulate temperature data or something similar.)

A message translator would be used to assign a hash value after entry to the main activeMQ queue. This would determine where in distributed storage (left side of diagram) it would go using the pub-sub pattern. The distributed storage would consist of identical objects that would hold java lists and be proxies for computers that would actually exist elsewhere if the system were real. It would be routed appropriately using a content-based router based on the hash value. This could also implement a content filter to remove the hash value or reformat the message. This portion would be asynchronous.

If the hash value of the message was within a range where it would be randomly selected for processing, it would go to a secondary queue. This queue would be responsible with providing up-to-date estimation values for mean and variance to an application based on the randomly selected data points. This sampled data used for real-time processing would use a splitter to send it to two separate objects for processing and then an aggregator to join the results. It is still a little unclear how the type of data used will fit this splitter/aggregator part of the model.

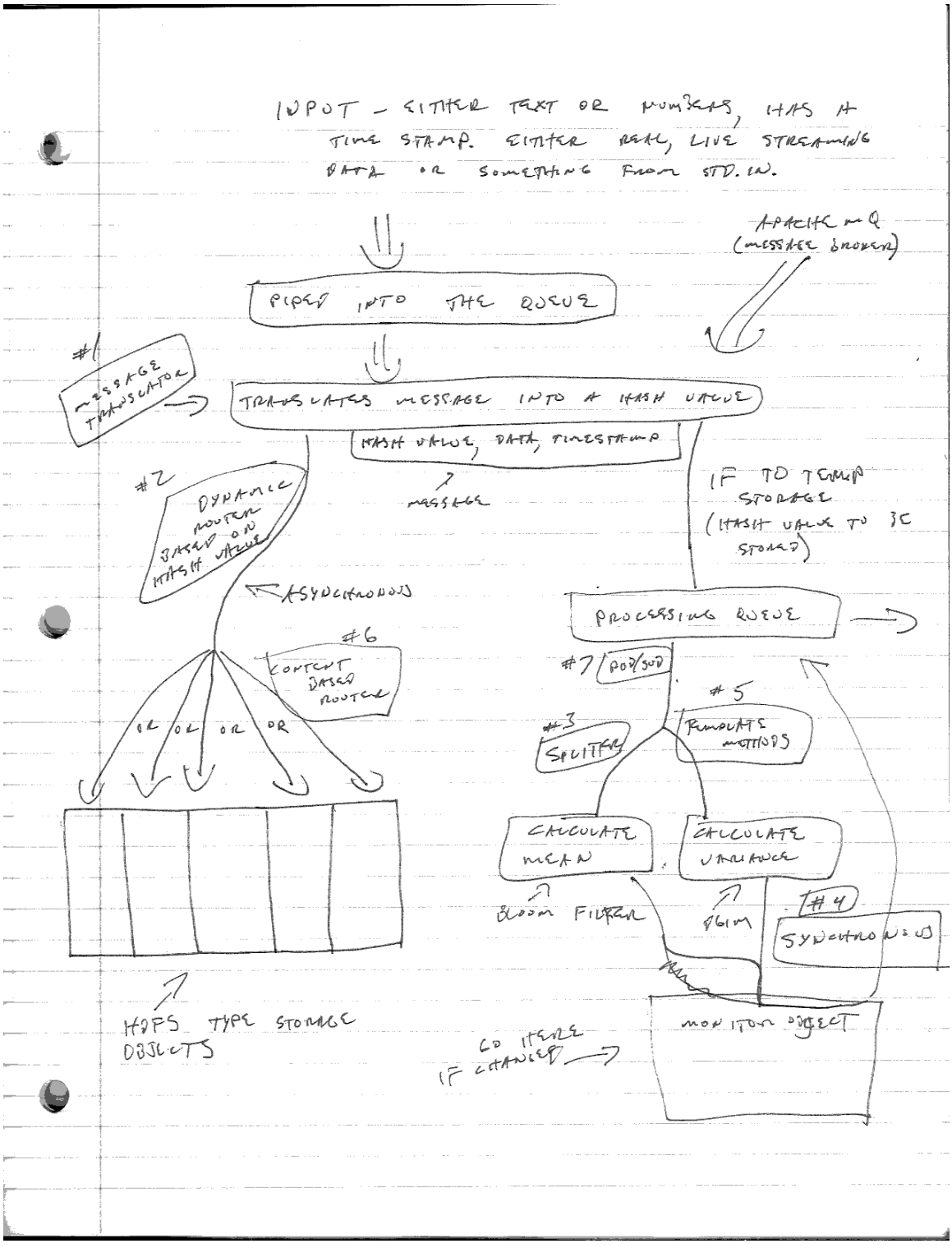
The new estimate of the parameters from the data would then be sent to an object (probably a file or standard out) in a point to point fashion, because this application in our example here would need to confirm that it had received the estimate.

Aside from the patterns mentioned above, all of this would be accomplished using activeMQ and Camel so the patterns involved in the message queue and communication built in to MQ would be there. Also, I plan to use the template method for my objects that calculate in real-time.

Let me know if this makes sense and is of the correct scope and uses the correct amount of patterns. The details might change a bit in implementation, but this seems to cover most of the patterns from the last portion of class and a few from the first part.

Concept on next page….

CONCEPT:


INPUT - EITHER TEXT OR NUMBERS, HAS A TIME STAMP. EITHER REAL, LIVE STREAMING DATA OR SOMETHING FROM STD. IN.
APACHE mQ (MESSAGE BROKER)
PIPES INTO THE QUEUE
#1 MESSAGE TRANSLATOR
TRANSLATES MESSAGE INTO A HASH VALUE
HASH VALUE, DATA, TIMESTAMP
MESSAGE
#2 DYNAMIC ROUTER BASED ON HASH VALUE
IF TO TEMP STORAGE (HASH VALUE TO BE STORED)
ASYNCHRONOUS
PROCESSING QUEUE
#6 CONTENT BASED ROUTER
#7 POP/SUB
#5 REPLICATE METHODS
#3 SPLITTER
OR OR OR OR
CALCULATE MEAN
CALCULATE VARIANCE
BLOOM FILTER
PGIM
#4 SYNCHRONIZED
MONITOR OBJECT
GO HERE IF CHANGED
HDFS TYPE STORAGE OBJECTS