Backend Message Flow

Paralellism: *From the message decoder down, this should be easily paralellizable: the retriever would get batches of messages from the local SMTP server, would save them to a single temporal file on disk or a text blob on a DB, and would pass the path or the ID to a message decoder, that could be on the same thread, on another thread, on another process or even a worker on a different machine with access to the message (local file or DB record). From there, any processing of the message batches should not cause any conflict with the other concurrent message processors.*

Note: *For clarity this graph shows as if every message were processed individually and the data structures passed by value, but probably to optimize function call and cache locality the messages would be processed on batches and the data structures/objects included in lists. These list would then be passed by reference and modified in place by the receiving methods.*

Message batches and RAM: *the message batches that the retriever creates should be limited in size to a configurable fraction of the total RAM in the machine, to avoid filling the machine RAM when processing huge emails with attachments. The number of paralell message processors should also be considered. For example, on a 600MB RAM machine, if we configure the system to don't take more than 20% (120MB) or memory for the message processing, then the size of every message batch shouldn't be greater than (120/number of workers), so for 4 workers the message batches should not exceeed 30MB (except for cases where a single message would be bigger than that).*