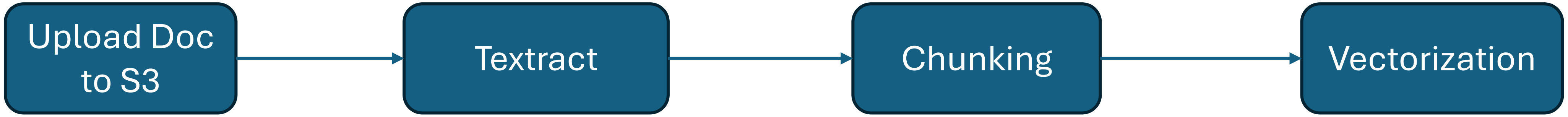
PDF Parsing Process Overview



Upload Doc to S3

Document in uploaded in s3 Bucket.

Textract

Texract is called to process document along with its s3 location

Chunking

Extract required text from JSON & Create chunks.

Vectorization

Vectorize Chunks and store in Milvus VectorDB

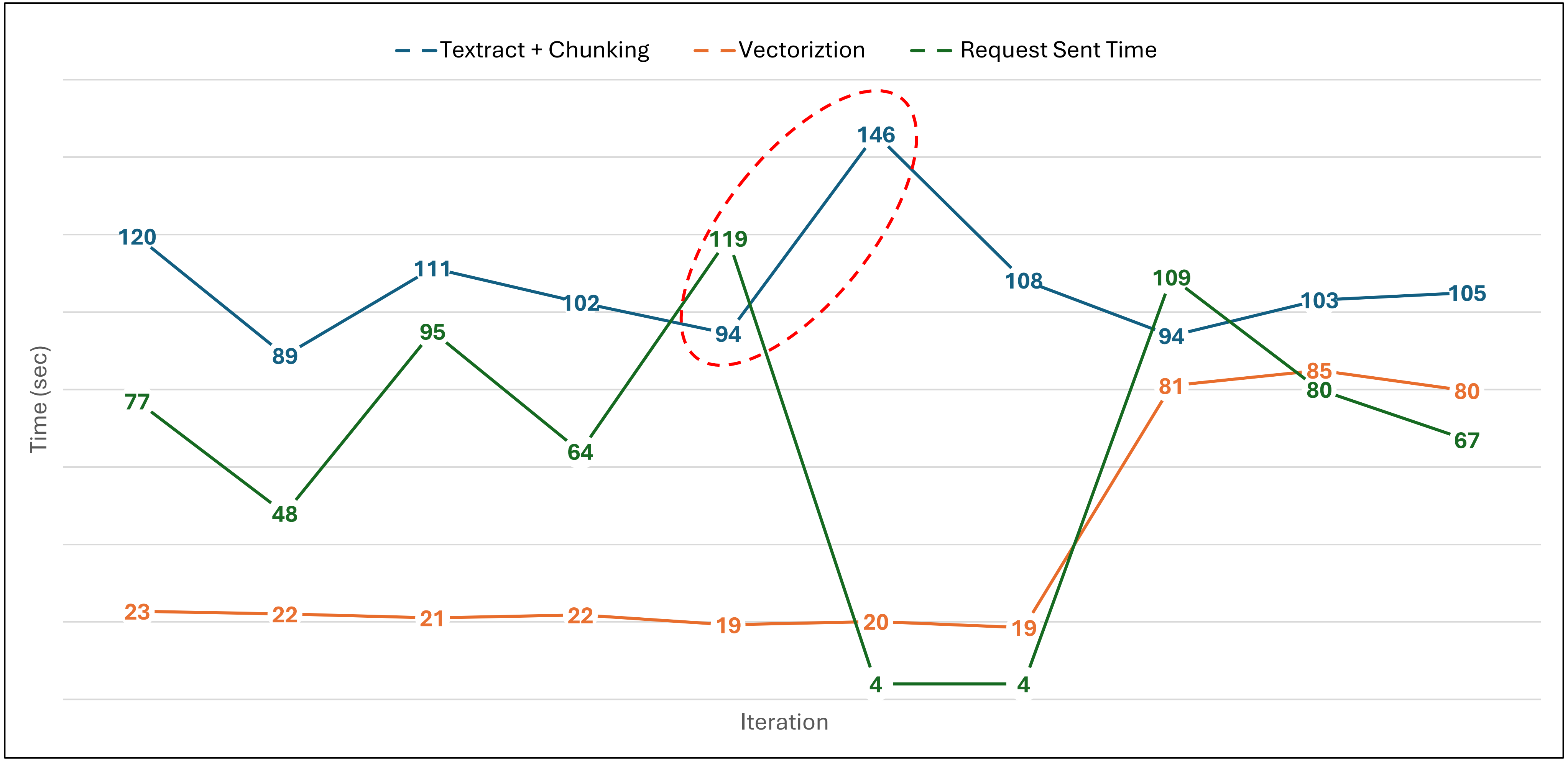
Issues Identified -

1. Inconsistency in time taken by Textract to process documents.
2. Varying request sent time for same document.
3. Textract Time increases with increase in complexity and number of pages.
4. Vectorization time increases as number of pages increases.

Inconsistency in Textract Processing Time –

Example 1 (02-TED 2022 JCEM supplement.pdf)

64 Pages, 48 MB Document



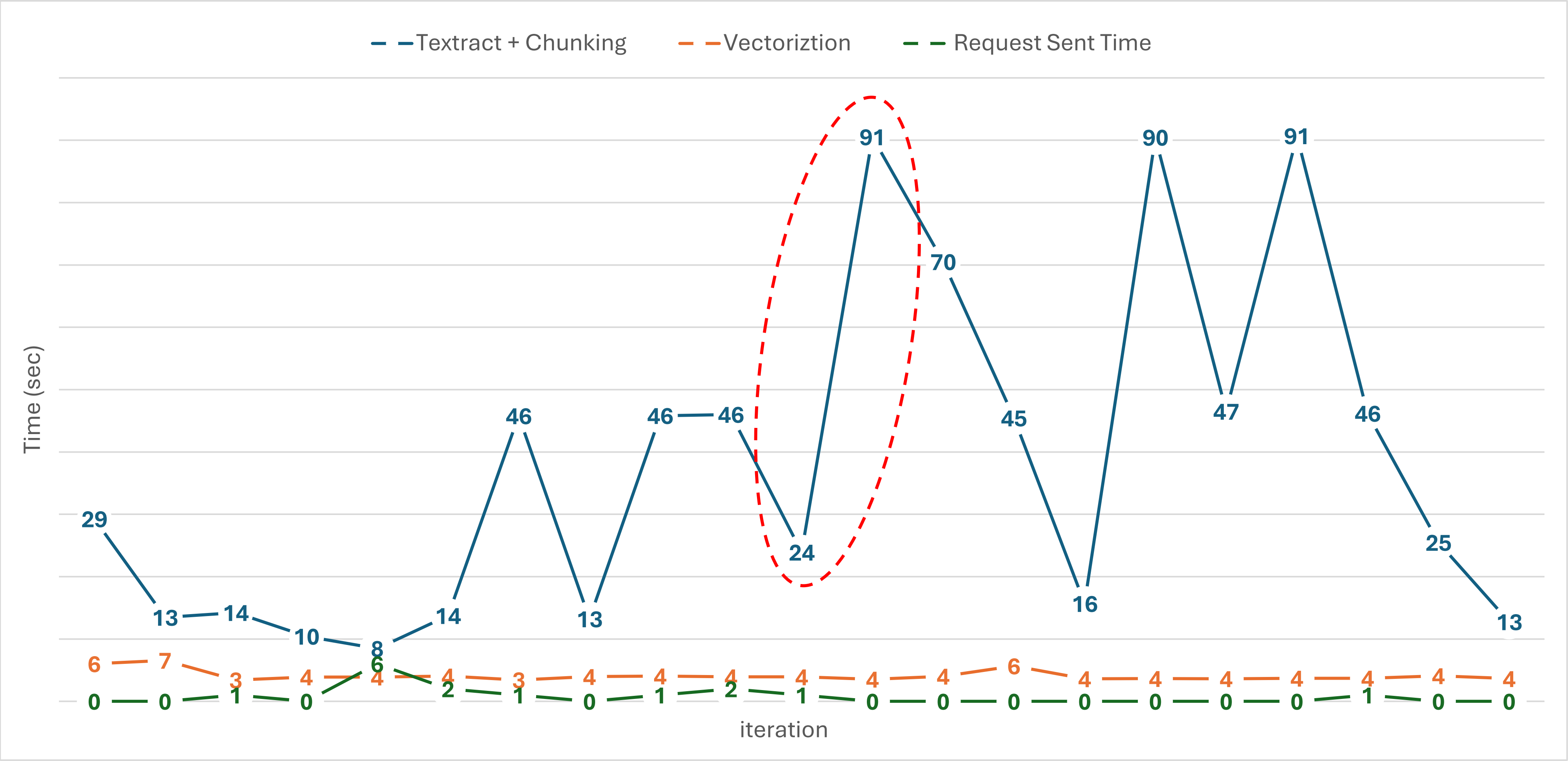
Adobe Acrobat Document

1. Textract takes 90 to 146 seconds to process this document.
2. Inconsistency seen in request sent time as well.

Inconsistency in Textract Processing Time –

Example 2 (Yen_Thyroid Eye Disease - Current Concepts and State of the Art.pdf)

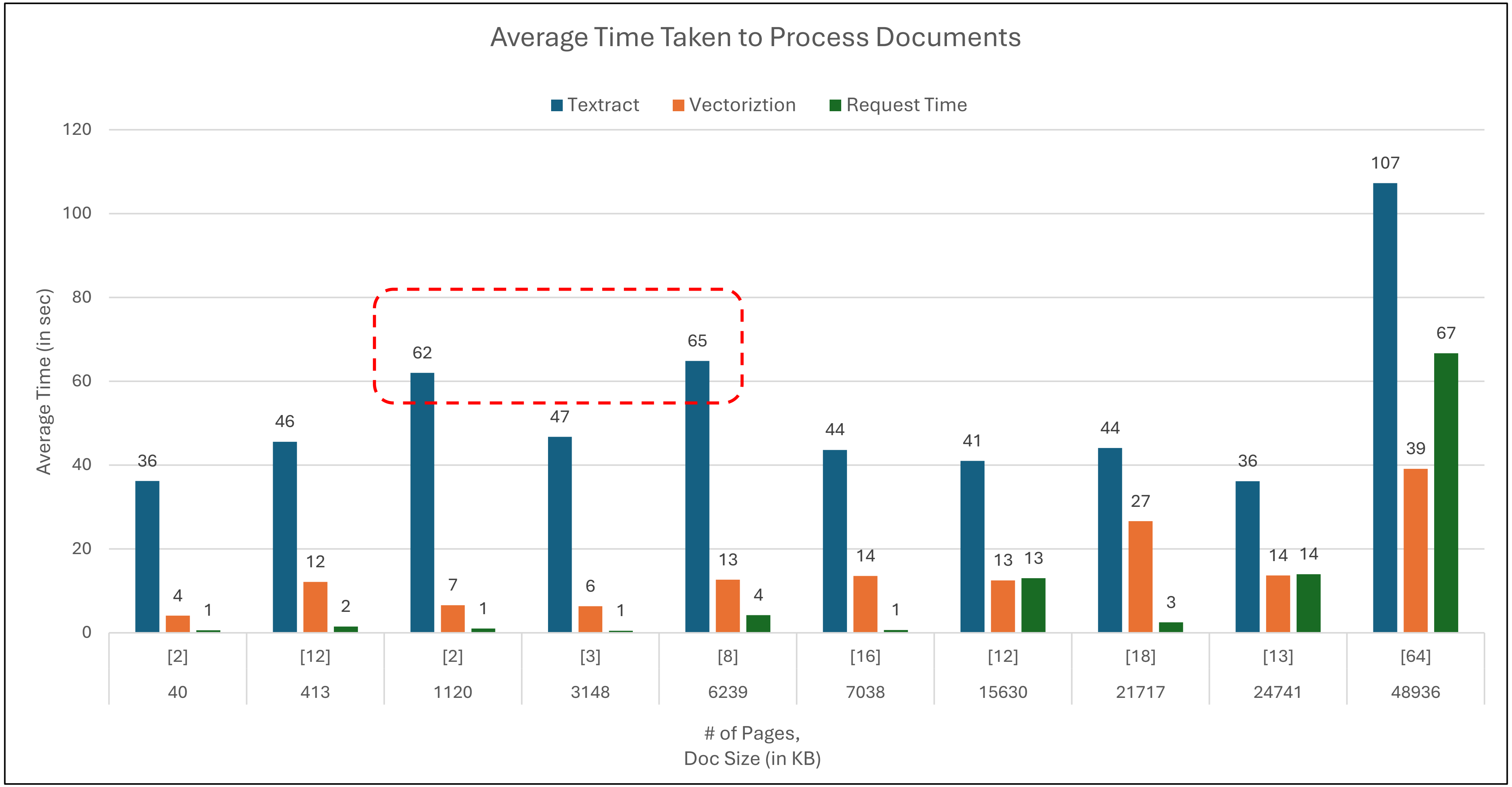
2 Pages, 40 KB Document



Adobe Acrobat Document

1. 40 KB document takes about 20 to 90 seconds.

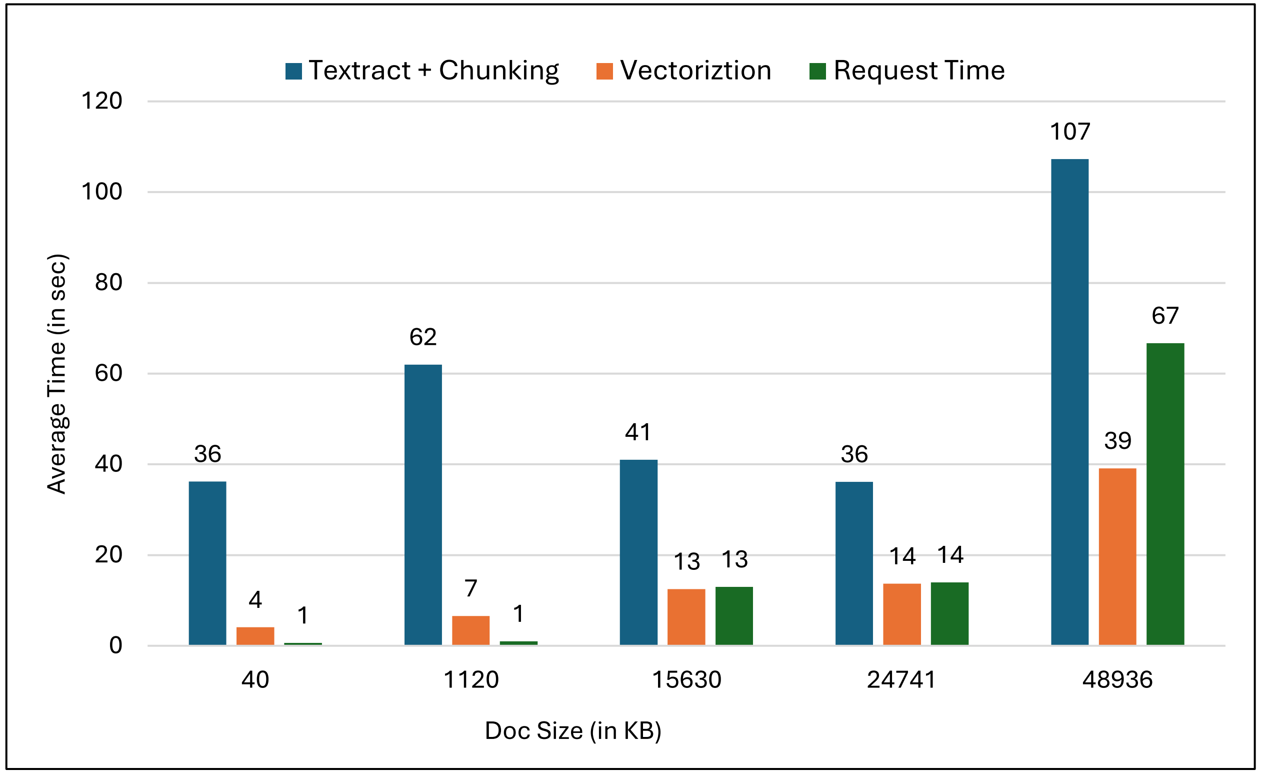
Increase in Document Processing Time –



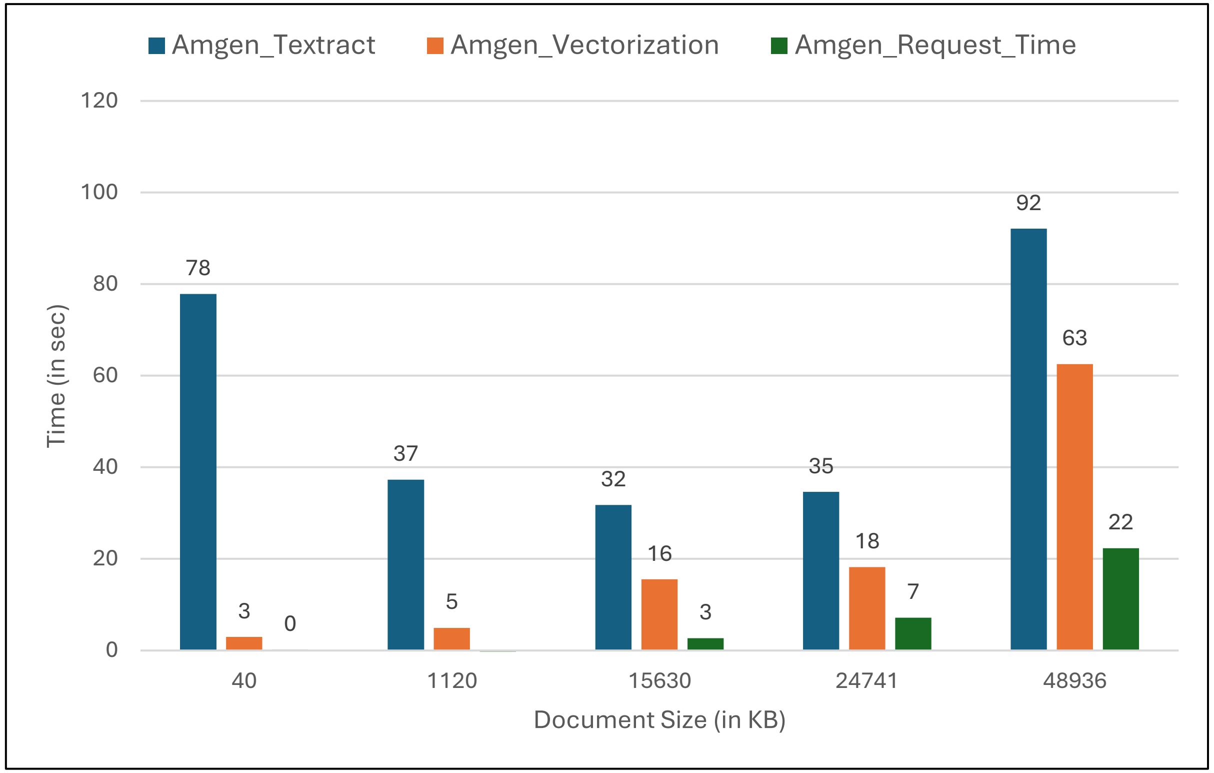
1. Textract processing time depends on document complexity as well as number of pages.
2. Vectorization time increases as number of pages increase

Comparing Time for document uploaded using Amgen Machine vs Cloud PC -

Cloud PC Upload



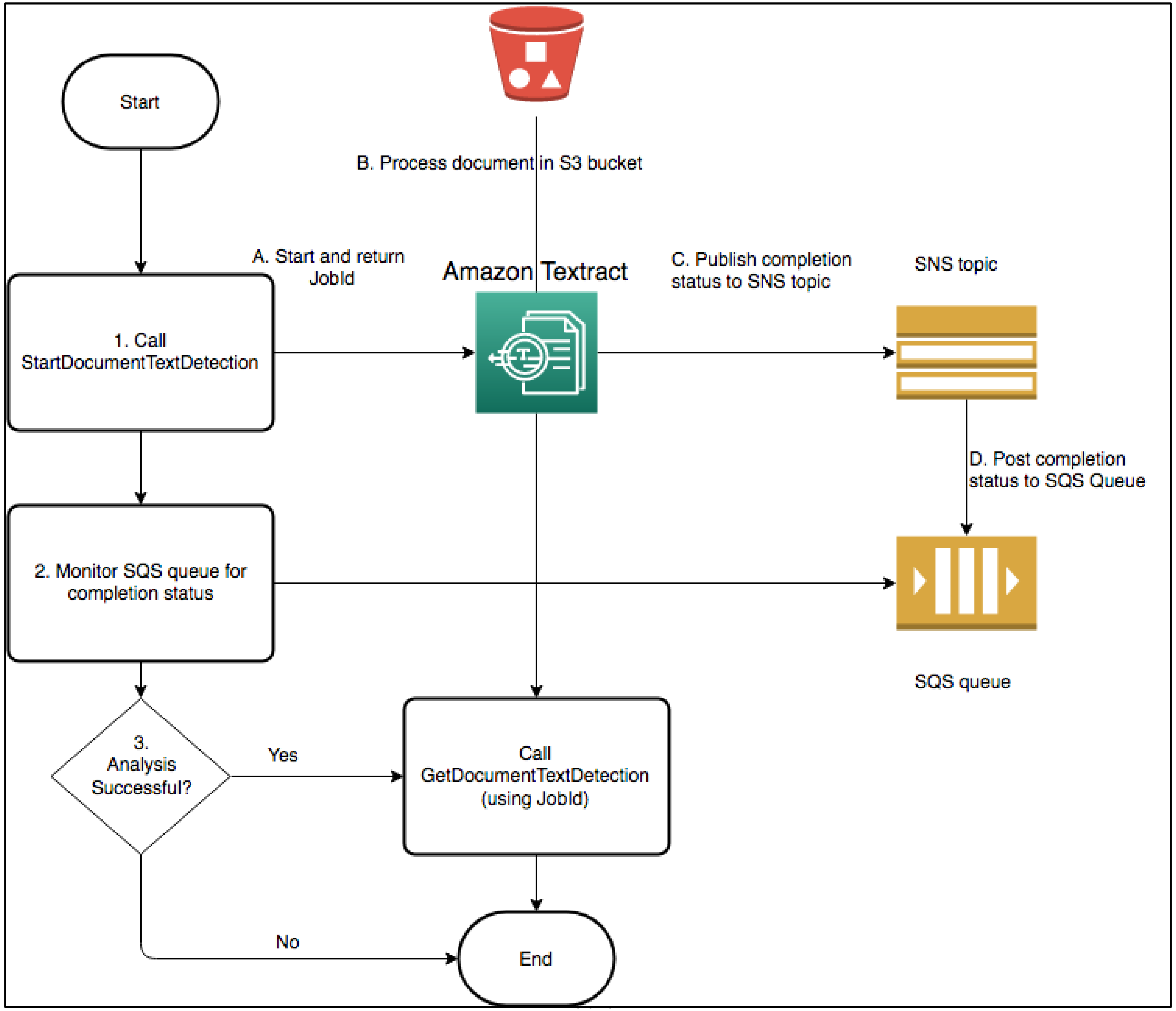
Amgen Machine Upload



1. The time taken to upload & process a document from Amgen Machine is not significantly different from the time taken to upload it from cloud PC.

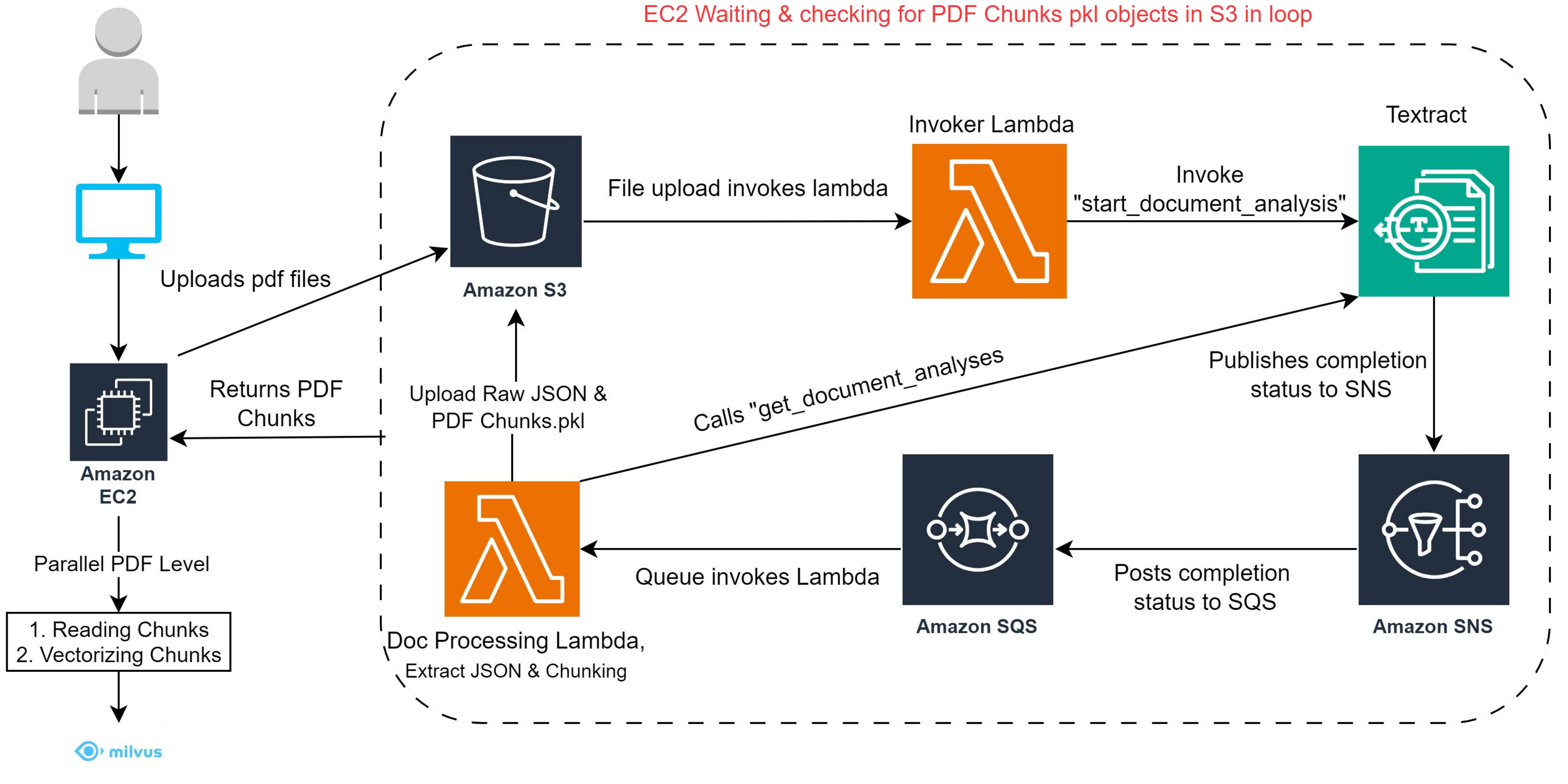
Textract Asynchronous Implementation in EC2

Current Architecture



Start
B. Process document in S3 bucket
A. Start and return JobId
Amazon Textract
C. Publish completion status to SNS topic
SNS topic
1. Call StartDocumentTextDetection
D. Post completion status to SQS Queue
2. Monitor SQS queue for completion status
SQS queue
3. Analysis Successful?
Yes
Call GetDocumentTextDetection (using JobId)
No
End

Textract Asynchronous Implementation using Lambda


EC2 Waiting & checking for PDF Chunks pkl objects in S3 in loop
Uploads pdf files
Amazon S3
File upload invokes lambda
Invoker Lambda
Invoke "start_document_analysis"
Textract
Returns PDF Chunks
Upload Raw JSON & PDF Chunks.pkl
Calls "get_document_analyses"
Publishes completion status to SNS
Amazon EC2
Parallel PDF Level
1. Reading Chunks
2. Vectorizing Chunks
milvus
Queue invokes Lambda
Amazon SQS
Posts completion status to SQS
Amazon SNS
Doc Processing Lambda,
Extract JSON & Chunking

PRO’s –

1. Better for multi pdf upload because of concurrent calls.
2. Partially independent of server load.

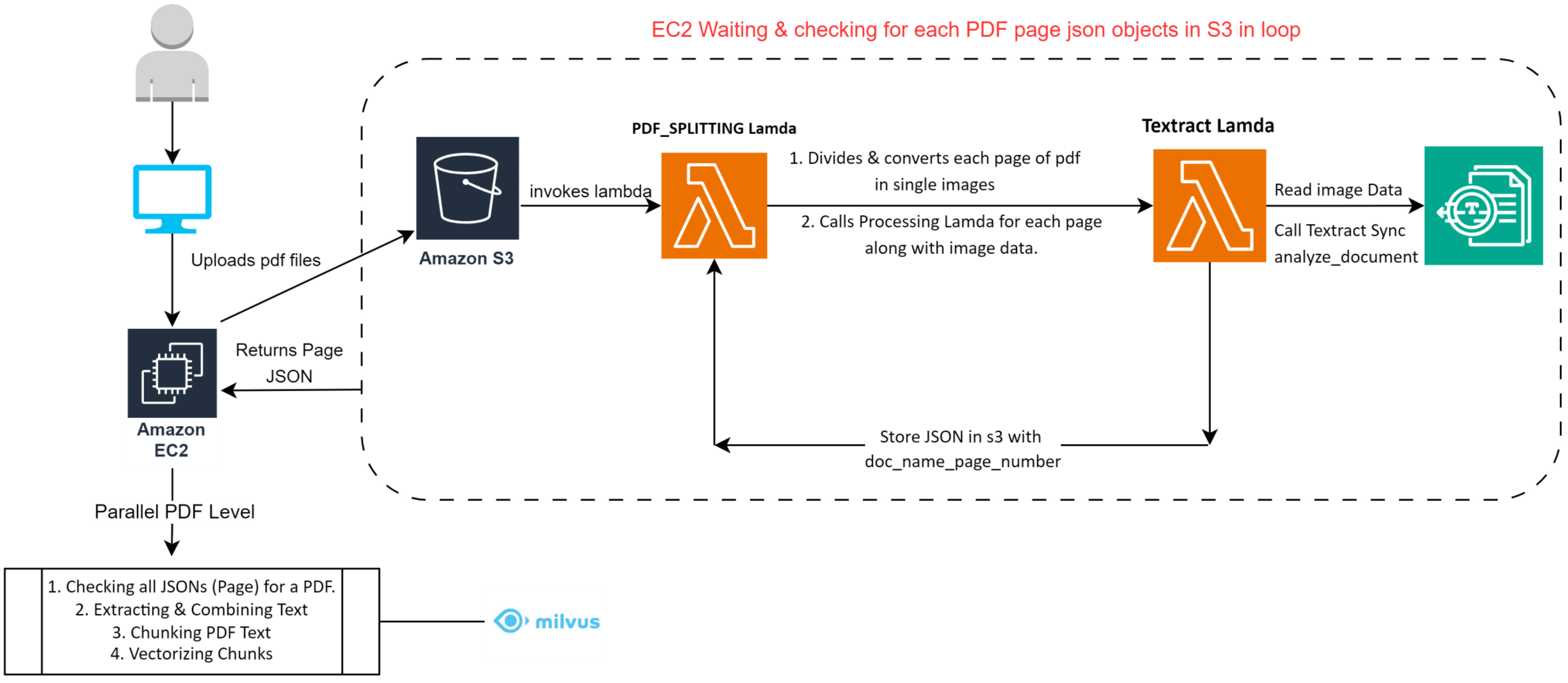
CON’s –

1. Server constantly checks the s3 & waits for PDF pkl files to be available.
2. No major decrease in time taken to process documents.

Questions –

1. Lambda Warmup time
2. Any better implementation to check if for a pdf all Json objects are present in s3.
3. FIFO vs Standard Queue, if a doc with large number of pages starts first then the other docs would have to wait for processing in 2nd lambda.

Textract Synchronous Implementation using Lambda



EC2 Waiting & checking for each PDF page json objects in S3 in loop

Uploads pdf files

Amazon S3

invokes lambda

PDF_SPLITTING Lamda

1. Divides & converts each page of pdf in single images

2. Calls Processing Lamda for each page along with image data.

Textract Lamda

Read image Data

Call Textract Sync analyze_document

Store JSON in s3 with doc_name_page_number

Returns Page JSON

Amazon EC2

Parallel PDF Level

1. Checking all JSONs (Page) for a PDF.
2. Extracting & Combining Text
3. Chunking PDF Text
4. Vectorizing Chunks

PRO's –

1. Quick for single document upload
2. Flexibility to Increase number of pages without significant increase in time

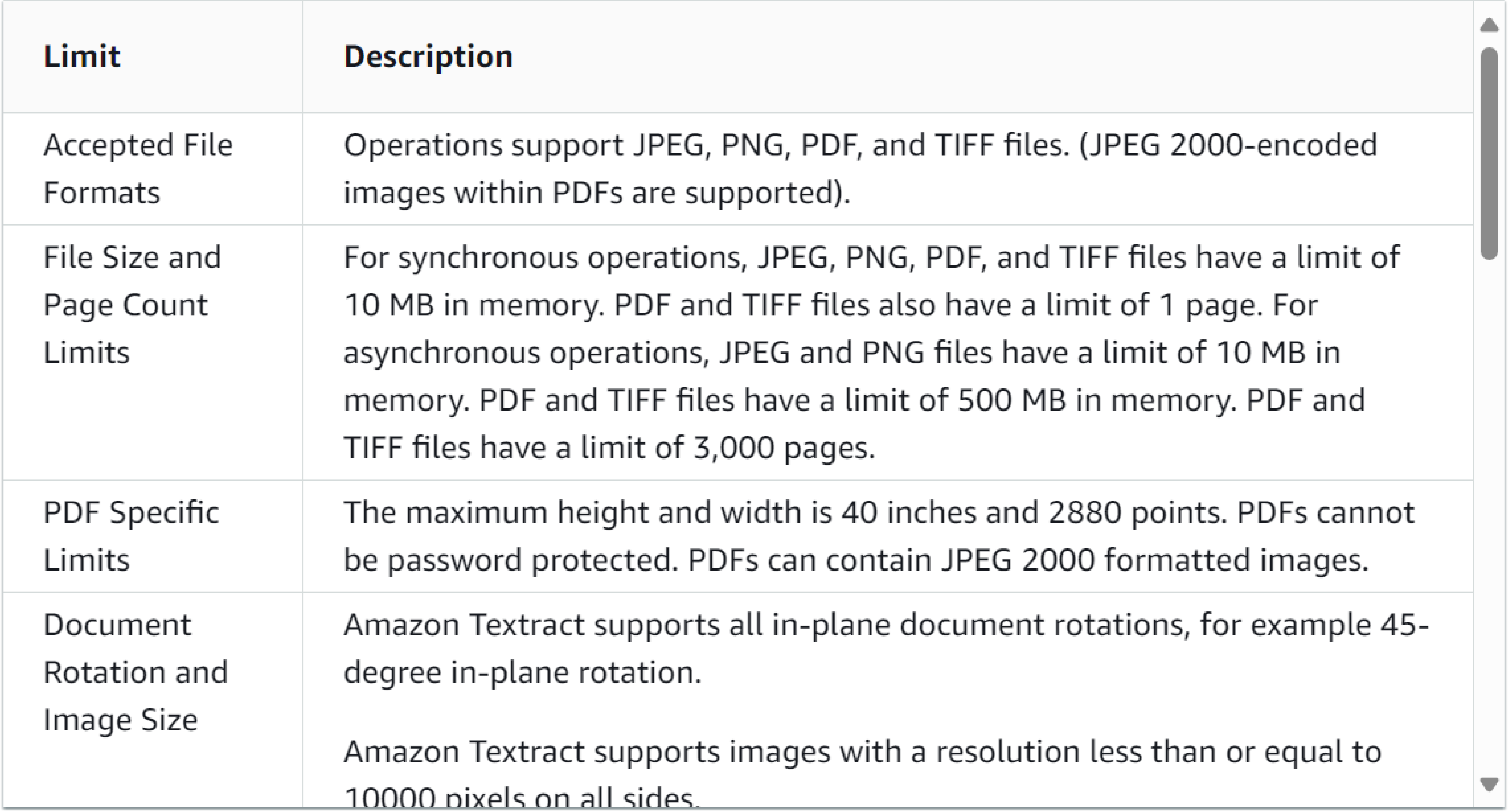
CON's –

1. Server constantly checks the s3 & waits for the single page json's to be available.
2. Throttling analyze_document API.
3. Complex implementation

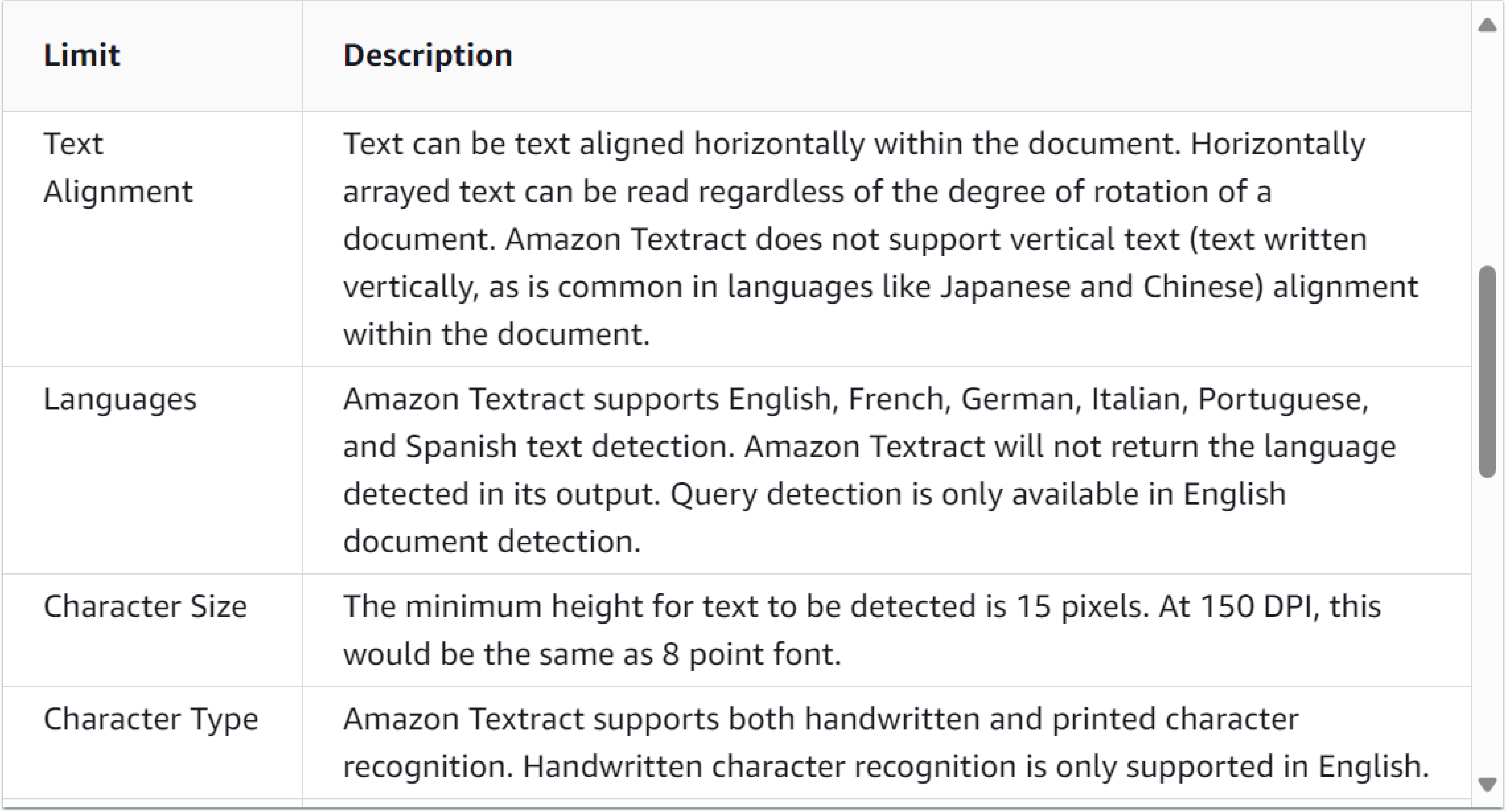
Questions –

1. Should we implement this using EC2 or Lambda?
2. Any better implementation to check if for a pdf all json objects are present in s3.
3. How many max number of pages we can have within 100 MB.

The following is a list of set quotas in Amazon Textract, which cannot be changed. For information about limitations in default quotas you can change, see the section Default Quotas in Amazon Textract.



Limit	Description
Accepted File Formats	Operations support JPEG, PNG, PDF, and TIFF files. (JPEG 2000-encoded images within PDFs are supported).
File Size and Page Count Limits	For synchronous operations, JPEG, PNG, PDF, and TIFF files have a limit of 10 MB in memory. PDF and TIFF files also have a limit of 1 page. For asynchronous operations, JPEG and PNG files have a limit of 10 MB in memory. PDF and TIFF files have a limit of 500 MB in memory. PDF and TIFF files have a limit of 3,000 pages.
PDF Specific Limits	The maximum height and width is 40 inches and 2880 points. PDFs cannot be password protected. PDFs can contain JPEG 2000 formatted images.
Document Rotation and Image Size	Amazon Textract supports all in-plane document rotations, for example 45-degree in-plane rotation. Amazon Textract supports images with a resolution less than or equal to 10000 pixels on all sides.



Limit	Description
Text Alignment	Text can be text aligned horizontally within the document. Horizontally arrayed text can be read regardless of the degree of rotation of a document. Amazon Textract does not support vertical text (text written vertically, as is common in languages like Japanese and Chinese) alignment within the document.
Languages	Amazon Textract supports English, French, German, Italian, Portuguese, and Spanish text detection. Amazon Textract will not return the language detected in its output. Query detection is only available in English document detection.
Character Size	The minimum height for text to be detected is 15 pixels. At 150 DPI, this would be the same as 8 point font.
Character Type	Amazon Textract supports both handwritten and printed character recognition. Handwritten character recognition is only supported in English.