Alfresco Sizing Guide 1.0

Alfresco Content Services 7.0

Hyland™ | Alfresco™

Alfresco Sizing Guide 1.0

Alfresco Content Services 7.0

This whitepaper provides information around things that need to be considered when deploying and sizing an Alfresco Content Services environment. This document is intended for Alfresco architects, administrators, and developers that are interested in the sizing factors that will help them size their Alfresco projects.

The scope of this document is constrained to the Alfresco stack including the Alfresco repository, user interface, and search applications. This document excludes any tuning advice specific to a particular operating system, virtualization platform, storage device, or network hardware. This document is merely a guide to sizing and Alfresco environment and Hyland is not responsible for inadequate sizing done by customers without guidance from Hyland.

Alfresco Sizing Guide 1.0

Alfresco Content Services 7.0

Table of Contents

1. Introduction

One of the first things to accomplish when implementing a new software application is to determine how to size the deployment for optimal performance. Each deployment will have specifics that will require consideration when estimating the sizing of your Alfresco Content Services (ACS) environment. Understanding the factors that influence resource usage is key to providing more accurate sizing estimates.

This document provides both test results with certain node volume levels (80 million, 500 million and 1 billion) and provides insight in how to interpret this information so that you can use the data provided to answer questions you might have around system requirements for your own installation of ACS.

Performance results are hard to describe and understand – they can vary significantly depending on what can appear to be small differences in test definition and how those tests are run. In this document, Alfresco is providing the reader with objective data using tests and environments that reflect as close to real-world scenarios as feasible. The systems under test were not specifically tuned for performance – they were configured using as close to the out-of-the-box settings as possible. While these results can be used to guide the sizing of an Alfresco system, it is strongly advised that they are not used in any comparison with other test results published either by Alfresco or by any other vendor.

2. Executive Summary

Alfresco Content Services (ACS) is proven to be a highly scalable and elastic ECM solution and version 7.0 continues to expand to support customer content growth. The information used for this document has been acquired in working with various customer and benchmarking our software to determine optimal performance guidelines.

This document details the results of benchmark testing for Alfresco Content Services version 7.0 on Amazon Web Services (AWS) infrastructure. This document's purpose is to provide examples of deployments of various sizes as a reference.

3. Terminology

To understand these performance metrics and the data provided in this document, it is important to get a clear understanding of the terminology related to performance that is used.

- **Throughput** – How many defined operations a given system can execute in a given period of time
- **Response Time** – The time between a client system sending a request and receiving the response to that request (in its entirety).
- **Performance** – Generally used as a generic term that relates to either throughput, response time, or both. A commonly used term, but ill defined.
- **Scalability** – An expression of how throughput and response time change as the system is given greater resources. A system that scales well will provide twice the throughput when given twice the resources while retaining response times.
- **Latency** – The time from a when a message is sent from one device across a network to when it is received by another device.
- **Scale Out** – Also known as horizontal scaling. Increasing the resources available to a software system by adding new instances of software components that run on new hardware along existing instances. This usually entails running the system by expanding operations to multiple machines.

- **Scale Up** – Also known as vertical scaling. Increasing the resources available to a software system without adding any new instances. This usually entails running the system on larger, bigger and/or faster machines.
- **CPUs or Cores** – A Central Processing Unit (CPU) is a physical “chip” on a motherboard. A Core is an independent process unity built into a CPU. A typical CPU will have four or more Cores.
- **IOPS** – Input/Output Operations per Second (IOPS) are commonly used to express the performance of a storage system as a number that indicates the average throughput of that system loaded with a blend of read and write operations.
- **Resilience** – Tangential to performance and scalability, resilience describes how tolerant a system is to failure of one or more of its components. Mentioned here as it relates to “scale up” versus “scale out” decisions.

4. Alfresco Content Services

There are many components that make up the Alfresco Content Services solution and each of these components can work with other components within the solution. Recognizing these individual components and services and their interoperability will help in clarifying where throughput can be improved to achieve better performance.

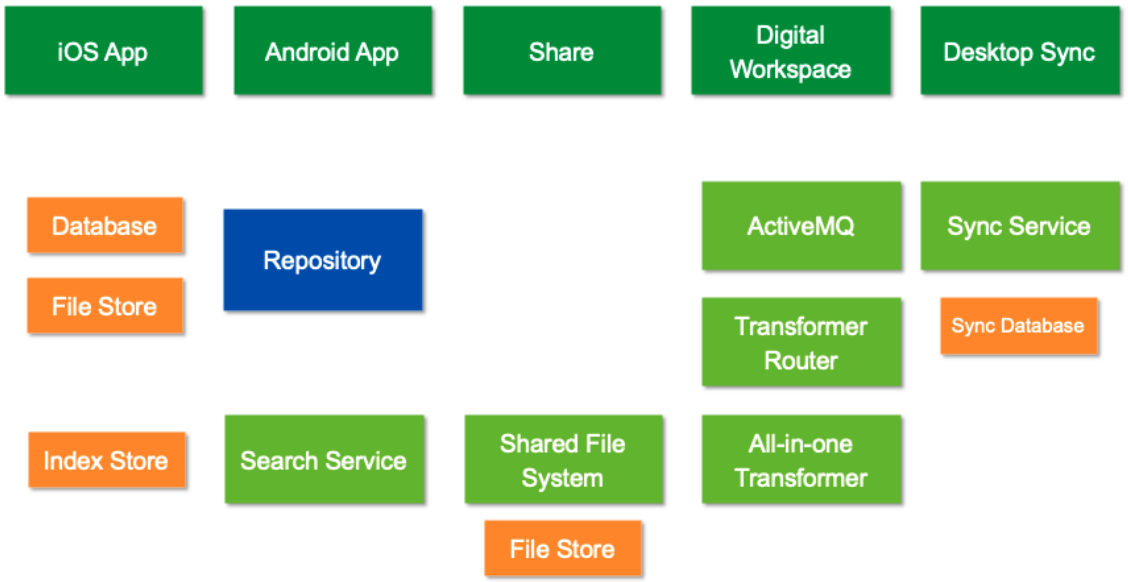

Figure 1. ACS Components and Services

Essentially, how you deploy these components together can have an effect on performance. It is important to understand the relationships between components so that you can avoid latency and optimize performance. For example, separating the Transformer and the Sync Services, two components that have a high consumption of CPU resources. This provides the ability to improve scalability and flexibility of the system as well as the resilience and fault tolerance as well. Alfresco recommends the separation of these components to provide customers with significant flexibility around scaling the system.

4.1. Component Connections Overview

There are many key communications that take place within the system and some are critical to performance.

- ***Repository to database*** – The repository must communicate with both the database as well as the file store. This is the most important connection in the system. This needs to be

optimized for low latency, especially to the database. This means that when the repository sends requests to/from the database, the request has to reach the database very quickly. The response time will depend on the compute time available to the database. This should be tuned for single digit millisecond response.

- Latency to the Database – for example, you do not want to put your database in a different availability zone than your repository if you are running in AWS.

- ***Repository to Repository*** – There is another critical connection between repository instances. Within the repository, there is a cache layer that sits just above the object relational mapping layer that sits above the database tier. This cache is synced across all members of a repository cluster, and that syncing mechanism is sensitive to latency.
- ***Repository to File Store*** – Finally, there is the repository to the file store. You do not need any super performing file store, but operations should be around 3,000 Input/Output Operations per Second (IOPS).

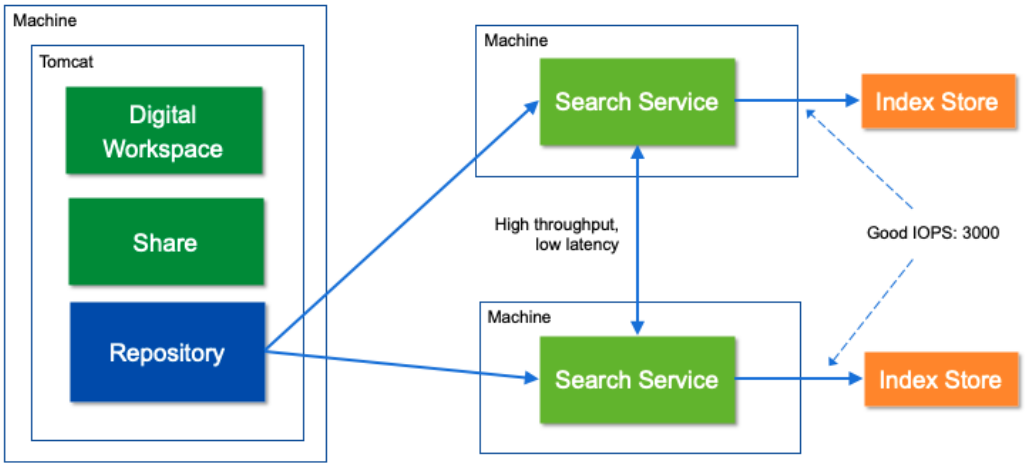

Figure 2. Repository and Search

- ***Search Cluster Connections*** – Another area in which to consider latency and configuration is that around Alfresco Search Services and Solr, if that is your configuration. If sharded, the connection between the shards is very important. The index store is also important but is not the limiting factor. The main factor is the connection between shards – so any query that must gather results from many shards has to communicate across the Solr cluster to put together a response.

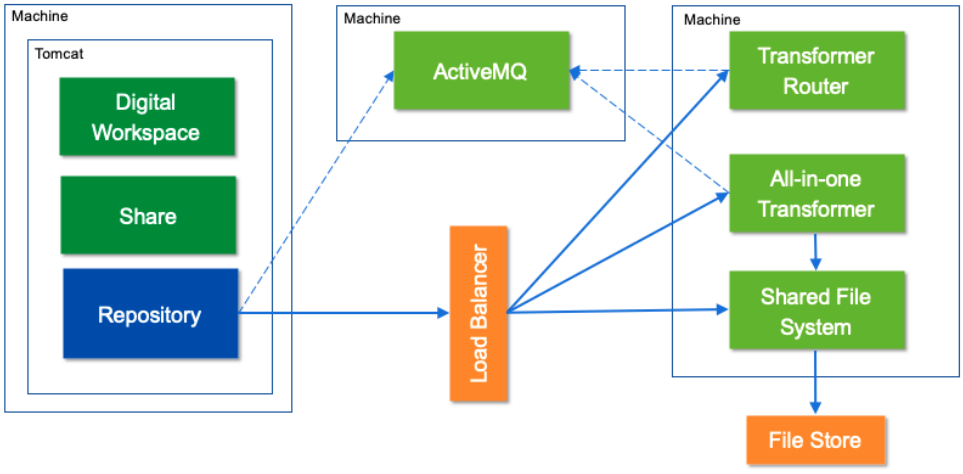

Figure 3. Repository and Transform Service

- ***Transform Service*** – The transform service enables asynchronous through messages using ActiveMQ as a broker. The All-in-one Transform Engine shown in the above figure is a grouping of all the individual transform engines encapsulating into a single deployed engine. The solid blue lines in the figure above represent *http* connections. The repository, transform router and the all-in-one transformer communicate through messaging via ActiveMQ.

  The repository connects to all of these components through a load balancer using *http*. When the repository wants a transformation to be done, the file is sent to the shared file system which then puts it on the file store. Then a message is sent to ActiveMQ to queue the transform router to transform the content into another format. The appropriate transform engine(s) required for that transformation are messaged through ActiveMQ. Once the transformation is complete, the transformer will message back to the repository which then goes back to the shared file system to get the transformed file.

  These messages and actions do not massively change the performance of the system, but this is important for scalability. It is important to understand that actions related to Solr may send requests to the transform router to obtain text from a PDF document, for example, to enable that for full text search. This configuration provides the ability to scale as needed to handle these requests.

4.1.1. Summary of Connection Points

To summarize, the connection points to play close attention to are:

- Repository to Database
- Repository to Repository
- Repository to File Store
- Search Cluster Connections

For scalability considerations, the following also needs to be noted:

- Transform Service

4.2. Sizing Considerations

There are many sizing elements to study consider deploying Alfresco. The table below represents major areas of interest and data that should be gathered to help guide the sizing exercise. These are listed in alphabetical order, not in order of importance. It is essential to not only locate and assemble this information, but to document it as well as a baseline for future reference.

Area of Consideration	Information to Gather
Architecture	Document the architecture requirements as well as the associated Alfresco components and their role in the overall architecture. Be sure to note requirements like high availability, optimized sizing, failover, virtualization, 3rd party components, protocols, etc. A diagram of the proposed architecture is optimal.
Content Ingestion	It is important to know any required batch uploads and volumes as well as other jobs that might update other systems. These types of things can include workflows, scheduled batch jobs, transformations, uploads, etc.
Customizations and Integrations	Gather the details around any required customizations and document the external system integrations. If code already exists to access the other systems, then it should be reviewed for proper API integration and code optimization.
Operations and Peak Loads	Understanding the read and write operations of your application needs and splitting those into operations such as browse, download and search will help with sizing. Determining if searches will be full text, global, custom searches will also assist in determining search loads and requirements. Be sure to document peak time frames as well as average loads.
Repository Content*	Assemble information about document types to be managed including sizes, number of versions, daily volumes, number of properties, distribution ratios, etc. This will yield information for the database and storage requirements of your deployment.

Area of Consideration	Information to Gather
Response Times	If there is an expected response time for certain operations for the system, be sure this is well documented. This will help determine what can be single threaded or must be multi-threaded, etc.
Security/Authority	One of the most important aspects is how security will be deployed in your application including the application of groups. It is also important to dissect how content will be accessed and used to further define the security configuration for the Alfresco deployment. In addition, extensive auditing can impact the database size and performance.
Use Case*	Collect information about the use case for the Alfresco deployment, the solution's intended use case as well as end user requirements and integration with other systems.
Users*	Determine the number of concurrent users including those that will be system users. Include detail around average time interacting with the system and the peak usage times as well as the average times.

Figure 4. Areas for Sizing Considerations

* *Denotes an area of essential information for sizing considerations. No sizing can be done without this information.*

5. Testing Overview

5.1. Common Use Cases

Alfresco deployments vary considerably because of the flexibility of the platform and although we can enumerate some generic common use cases, the details on which each real implementation differs may be very important from architecture and sizing perspective. We consider that generally Alfresco solutions can be classified on one of the 2 following cases:

- Collaboration
- Backend Repository

The more information you know about how Alfresco will be deployed and used before you implement, the more accurate the system sizing can be to help with deployment decisions. Understanding your content use cases helps you to avoid overlooking subtle details that may be hidden in your numbers. Each application and organization is different as are the individual use cases which can make sizing more difficult especially if the repository is serving more than one application.

For example, if you are using Alfresco as a backend repository that will be accessed from one or more different applications, the needs and stresses on the environment can be quite different than if you have a repository for a single department or application. The use case can also affect your overall folder structure, site structure and more. Some applications reference documents by their unique identifier in the backend repository which is very efficient. However, other applications may rely heavily on searches including full text searching which will impact system performance at many layers. This extends to the protocols accessing the system (CMIS, REST, etc.) as well as network connectivity between applications. This is another example of how high-volume content ingestion on the same instances being utilized by end users may result in performance impacts.

In general, know your use case which includes how users will access the content so that you can make the appropriate sizing adjustments for each application tier of Alfresco.

5.2. Background

Alfresco has invested heavily in performance and scale testing. We have a performance team with extensive experience in performance testing and analysis. This team has developed a performance testing system that provides a contiguous program of performance investigation and improvement.

5.3. Content Mixture

While every customer has a different mixture of file types, Alfresco has detailed the file types in the repositories used for this testing in Appendix 2 - Test Content Profile under Content Files beginning on page 44. Customers with different mixtures should perform testing with their own sets and use this guide to help guide their deployment.

5.4. Testing Structure

The results provided in this document were obtained using these steps:

Four (4) environments – Small, Medium, Large and Extra-Large – were created. These environments are described in more detail in Appendix 1 - Environment Information starting on page 36 of this document.

1. The appropriate node volume number of files (80 million, 500 million and 1 billion are covered in this paper) were imported into each of the environments. The profile of these files is described in Appendix 2 - Test Content Profile beginning on page 44.
2. A sequence of different operations was executed on each environment. The operations were:
    - In-place File Ingestion
    - File Upload
    - Metadata Update
    - File Download
    - Full Text Search within a site
    - Search by Date Range within a site
    - Search by Text Property within a site

In each case, the operations were executed against the REST API endpoints exposed by the system.

Initially, each operation was invoked with just one concurrent request, and this concurrency was gradually increased until no further increase in throughput was observed. Within this document, this point is referred to as the "saturation point".

Note that this document illustrates the average number of requests per minute being handled by the system under certain concurrent load. This load has been applied programmatically without wait times between requests which means that these are the results you should expect to see if loading the system with client applications that are making requests as fast as they possibly can.

In a system being used by humans through a user interface, with the increased pauses that entails, one should expect the system to be able to handle between six and twelve times the amount. That is to say:

Max concurrent users ~= 12x max concurrent requests

Naturally, this can vary quite widely depending on how the system is used, so this calculation is just a guide. In some cases where a user clicks and then reads, the multiplier may be as high as thirty.

5.5. Server Details and Configuration

Alfresco testing utilized an Amazon Web Services (AWS) environment to performance this performance testing. As mentioned in the previous section, Different sized environments were used for the executed tests: Small, Medium, Large and Extra-Large. The server configuration for the environments included an AWS instance is provided in two availability zones and storage of

documents/content on Amazon S3[1] (AWS Simple Cloud Storage). The high-level AWS components for each environment can be found in the following figure with the EC2 instance types for each component provided.

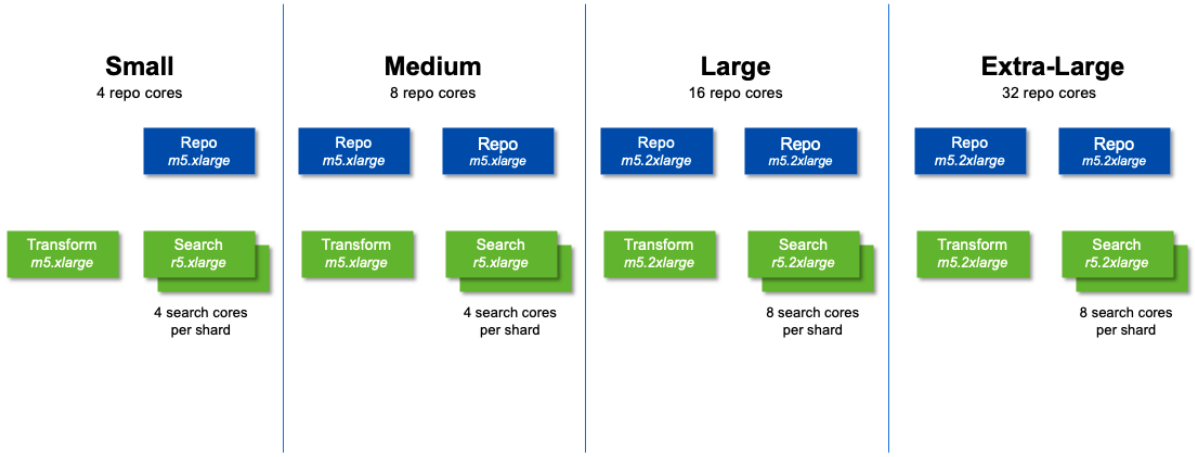

Figure 5. Test Environment AWS Instances

In these configurations, there is one (1) search service for every 60 million documents. Another item to note is that our experience has shown that performance is best when the repository (repo) is scaled up until you reach *m5.4xlarge* and then it should be scaled out by adding additional repo instances. A more detailed configuration can be found in Appendix 1 - Environment Information beginning on page 36. For more information about these instance types, please see the Amazon Web Services website on https://aws.amazon.com/ec2/instance-types/.

6. Performance Results

This section provides the results of key tests for each configuration (small, medium, large, and extra-large). Results are separated as follows:

- 80 million nodes
- 500 million nodes
- 1 billion nodes

For each of these sizes, the following test results are provided:

- In-Place File Ingestion

  Ingesting files is a typical action taken by customers which consists of ingesting large volumes of data into an Alfresco Content Services environment. This can be a daily ingestion task or possibly a legacy ingestion task. This test focuses on number of ingested batches and throughput to ingest those batches.
- File Upload

  Uploading files is a common task for users and it different than an in-place file ingestion. This test focuses on the uploads per minute with concurrent requests.
- Metadata Update

  Updating the properties associate with a file – which can include changing a date, a customer name, or a status – is something that is done often within daily business. It is important to capture performance information about this task.

[1] https://aws.amazon.com/s3/

- File Download

  Downloading files from the Alfresco Content Services environment is a typical and common user task. This test shows throughput of file downloads per minute with concurrent requests.
- Search

  Searching is one of the fundamental operations within an Alfresco Content Services system. Searches can be by date, property, full text, or combinations of these. For the search performance tests, the following were conducted:
  - Full Text Search within a Site
  - Search by Date Range within a Site
  - Search by Text Property within a Site

For each test, we have provided two graphs – one for throughput and one for response time results. These graphs show increasing parallel operations on the x-axis and operations per minute (for throughput) and response time in milliseconds (for response time) graphs. Each graph shows the results for all four environments (small, medium, large, and extra-large).

Please note that “parallel operations” are not the same “concurrent users.” If you wish to make this extrapolation, remember that one should expect the system to be able to handle between six and twelve times the amount of the parallel operations.

Max concurrent users ~= 12x max parallel operations

As previously mentioned, each operation was invoked with just one concurrent request, and this concurrency was gradually increased until no further increase in throughput was observed. This is referred to as the "saturation point".

6.1. Interpretation

The data in this document can be overwhelming when viewed out of context, so we have provided some examples on how to use the data in this document to assist you in applying the findings to your own requirements (Interpretation beginning on page 33). Reading the results for each test as reported in this document is only part of the challenge – there is also some need to interpret these results to answer specific questions you might have about your own configuration.

6.2. Results for 80 Million Nodes

This section contains the results for small, medium, large, and extra-large for an ACS environment with ***80 million*** nodes.

6.2.1. In-place File Ingestion

Test Results

The first results show the ingestions per minute for 100-file batches. For example, a total of 66 100-file batch ingestions were completed in the ***small*** environment before reaching the saturation point where no additional increases were observed.

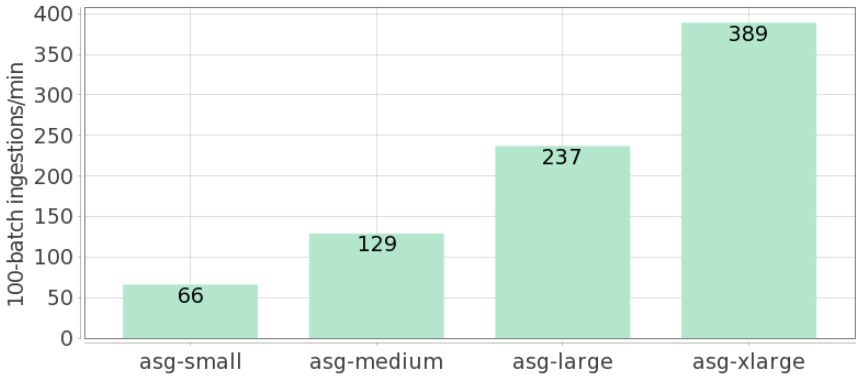

Figure 6. In-Place File Ingestion (100-file batches) per Minute – 80 million nodes

In all environments, it takes between 4 and 5 seconds to ingest each batch of 100 files.

6.2.2. File Upload

Test Results

The figure below shows the file uploads per minute for concurrent requests to the ReST API on each of the environments.

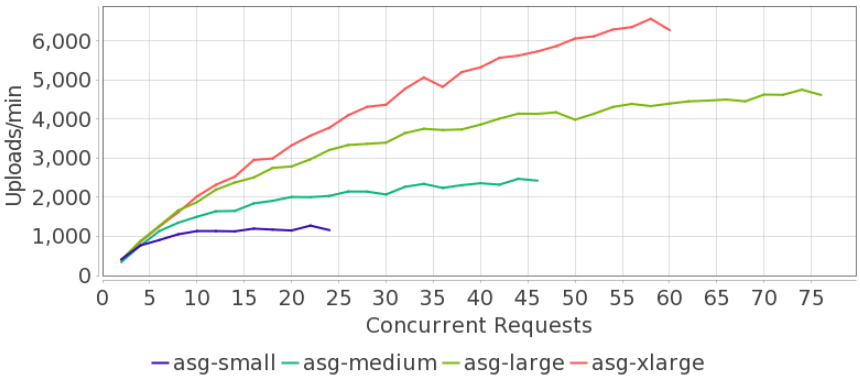

Figure 7. File Upload Operations per Minute – 80 million nodes

Another important metric when reviewing file upload is the response time for requests. The response time is provided in the next figure. These values represent the response time achieved for 95% of the tests.

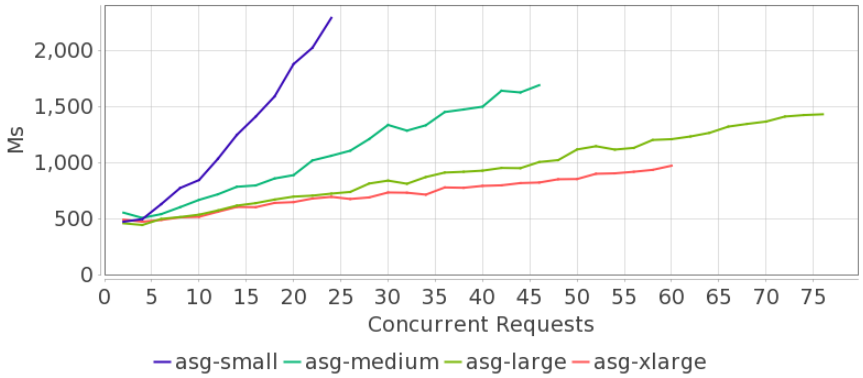

Figure 8. Response Times (95 Percentile) File Upload – 80 million nodes

6.2.3. Metadata Update

Test Results

The graph below shows the results for updating metadata (properties) of nodes in the repository.

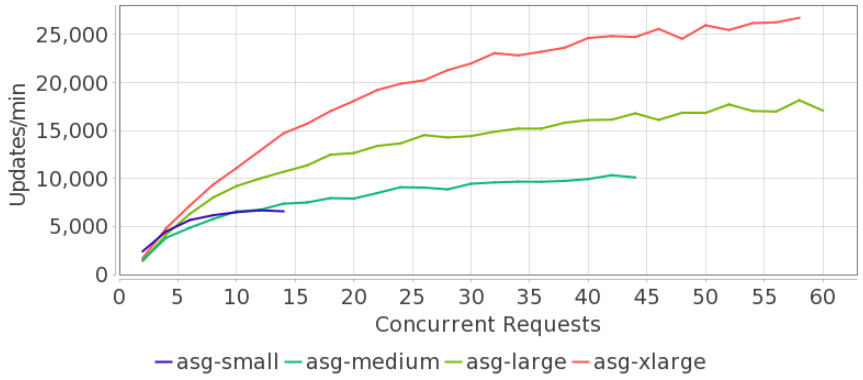

Figure 9. Metadata Update Operations per Minute – 80 million nodes

The response time is also provided in the next figure. These values represent the response time achieved for 95% of the tests.

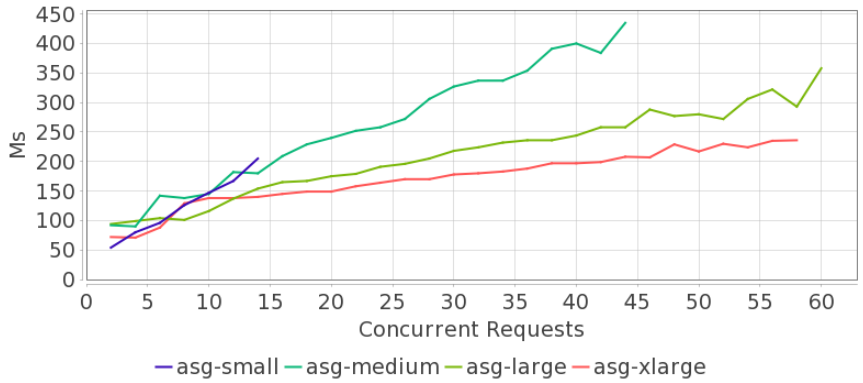

Figure 10. Response Times (95 Percentile Metadata) Update – 80 million nodes

6.2.4. File Download

Test Results

The graph below shows the results for downloading files from the repository.

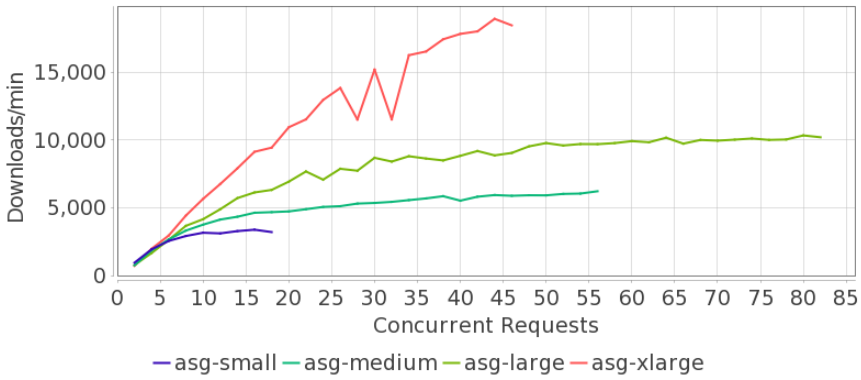

Figure 11. File Download Operations per Minute – 80 million nodes

As can be seen in the figure above, the number of downloads per minute seems to peak for the extra-large environment at approximately 47 concurrent requests with about 18,000 downloads per minute while the large environment levelled out and continued to perform just under 10,000 downloads per minute from 45 concurrent requests to 85 concurrent requests. The automated test system used by Alfresco attempts to detect when saturation has been reached to determine when to stop ramping up the load. In this case the algorithm was a little premature, and there is no reason to think the extra-large environment would not continue following its trajectory to reach about 20,000 requests per minute and achieve at least the same level of concurrency as the large environment in the real world.

The response time is also provided in the next figure. These values represent the response time achieved for 95% of the tests.

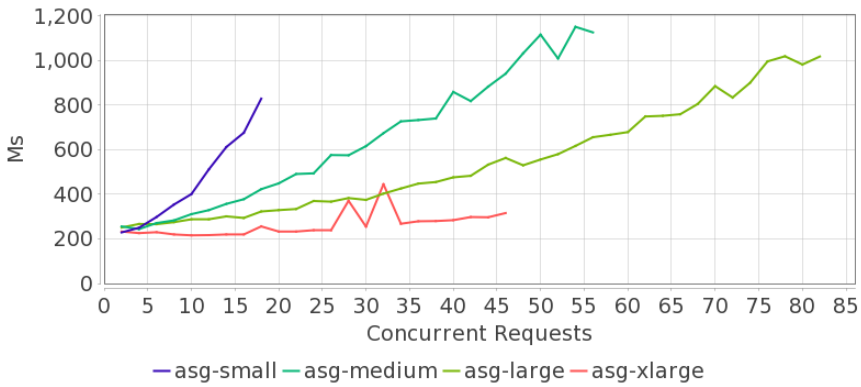

Figure 12. Response Times (95 Percentile) File Download – 80 million nodes

6.2.5. Search

Test conditions

- 80 million files are in the Solr index.
- 40 million files are in each Solr shard.
- The number of documents per site is ~5-100k distributed randomly.
- Each individual search request is executed against a single site. Each test agent is allocated a site at random at the start of the test and uses that site for the duration of the test.

The following queries are used for search scenarios:

- Full Text Search within a Site:
  `"query": "SITE:\"${site}\" AND TEXT:${term}"`
- Search by Date Range within a Site:
  `"query": "SITE:\"${site}\" AND cm:modified:[NOW/DAY-${dateFrom} TO NOW/DAY+1DAY] AND TYPE:\"cm:content\""`
- Search by Text Property within a Site:
  `"query": "(SITE:\"${site}\" AND TYPE:\"cm:content\" AND cm:name:${term})"`

Full Text Search within a Site

Test Results

The graph below shows the results for the number of concurrent requests for full text searches executed within a site.

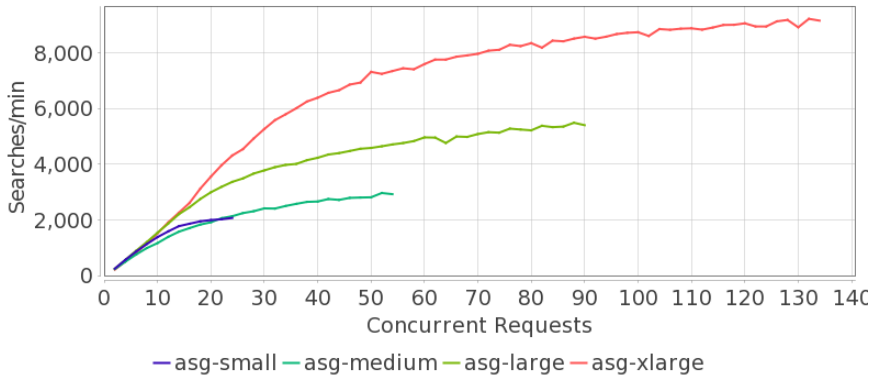

Figure 13. Full Text Search within a Site Operations per Minute – 80 million nodes

The response time is also provided in the next figure. These values represent the response time achieved for 95% of the tests.

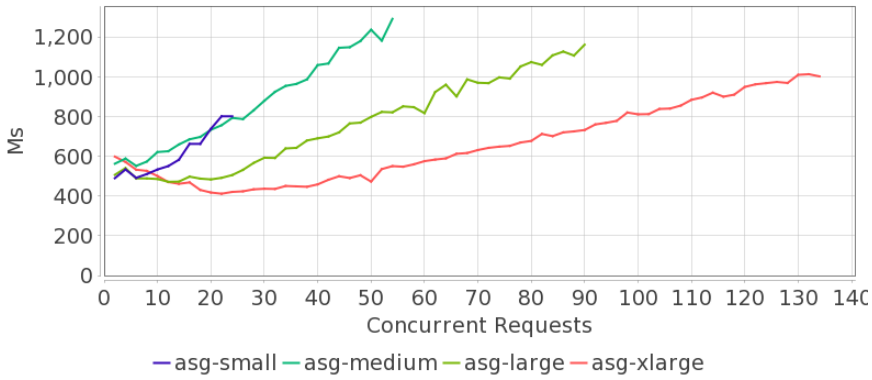

Figure 14. Response Times (95 Percentile) Full Text Search within a Site – 80 million nodes

Search by Date Range Within a Site

Test Results

The graph below shows the results for the number of concurrent requests for searches within a date range (such as all files created between 2019-01-02 and 2019-12-31) within a site.

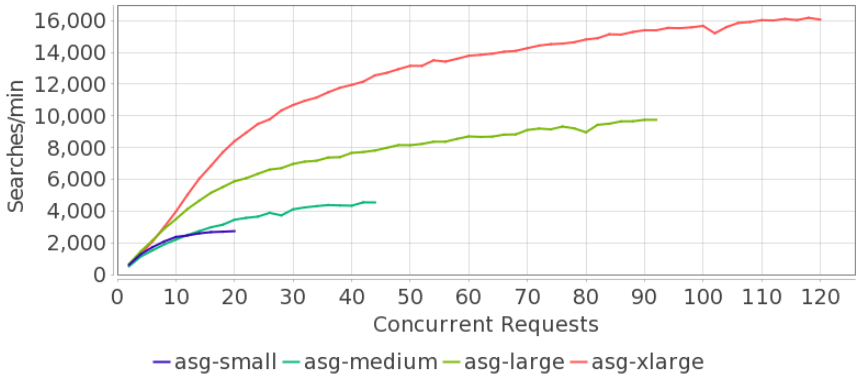

Figure 15. Search by Date Range within a Site Operations per Minute – 80 million nodes

The response time is also provided in the next figure. These values represent the response time achieved for 95% of the tests.

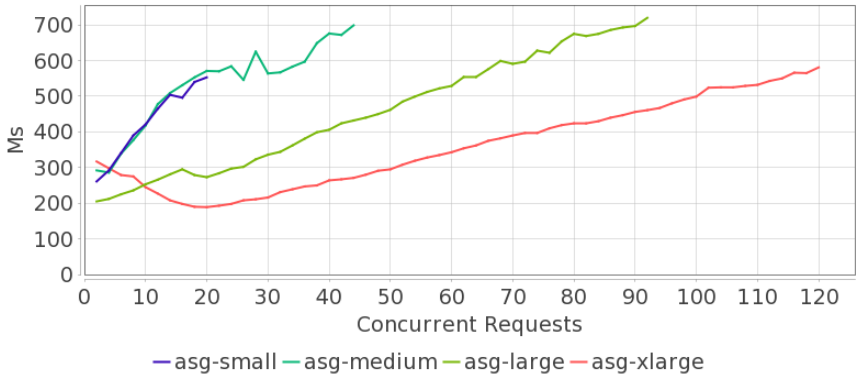

Figure 16. Response Times (95 Percentile) Search by Date Range within a Site – 80 million nodes

Search by Text Property within a Site

Test Results

The graph below shows the results for searches by a text property per minute within a site.

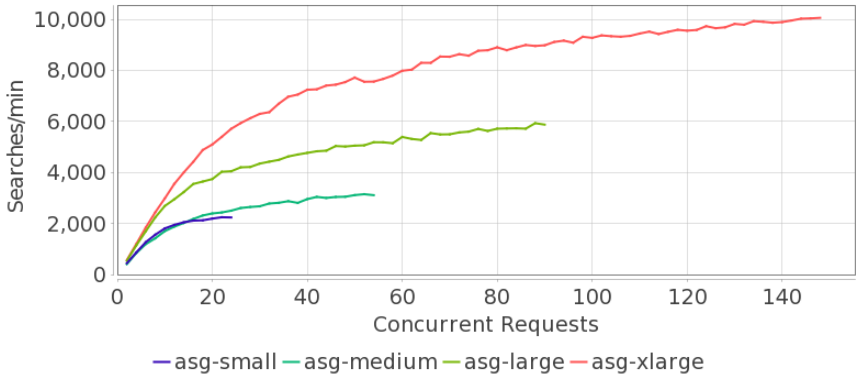

Figure 17. Search by Text Property within a Site Operations per Minute – 80 million nodes

The response time is also provided in the next figure. These values represent the response time achieved for 95% of the tests.

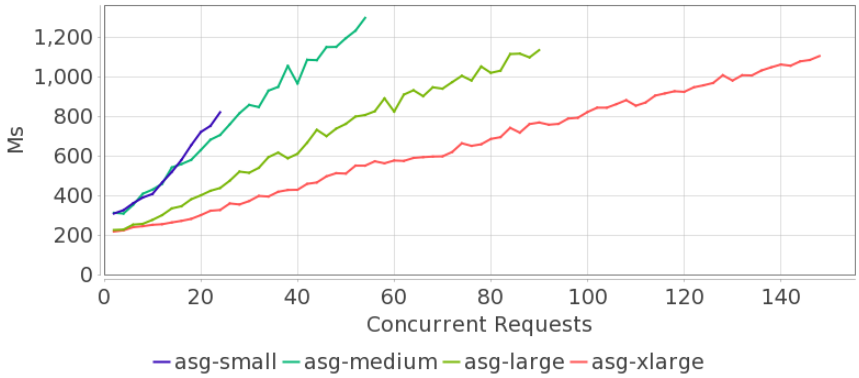

Figure 18. Response Times (95 Percentile) Search by Text Property within a Site – 80 million nodes

6.2.6. Summary

For ease of reviewing the information provided in this section, the table below summarizes all information and also includes CPU utilization by component:

- Repository (Repo)
- Solr
- Transformation Server (TS)
- Database (DB)

Test	Environment	Throughput per minute	Concurrent requests	Error rate, %	Response time, ms	CPU % Repository	CPU % Solr (coordin)	CPU % Solr (others)	CPU % TS	CPU % DB	Reason of saturation
In-place File Ingestion (100-docs batch)	***Small***	66	14	0.0	4,090	95	9	8	99	39	CPU REPO
	Medium	129	31	0.0	3,954	87	12	12	98	33	CPU REPO
	Large	237	59	0.0	7,175	89	10	9	93	21	CPU REPO
	XLarge	389	82	0.02	5,012	81	5	5	75	23	CPU REPO
File Upload	***Small***	1,275	22	0.0	2,024	99	7	6	78	25	CPU REPO
	Medium	2,471	44	0.0	1,625	98	8	7	98	29	CPU REPO
	Large	4,750	74	0.01	1,423	96	6	6	91	44	CPU REPO
	XLarge	7,916	84	0.01	1,066	90	4	4	64	50	CPU REPO
Metadata Update	***Small***	6,690	12	0.0	167	98	38	15	0	60	CPU REPO
	Medium	10,346	42	0.0	384	91	39	9	1	45	CPU REPO
	Large	18,157	58	0.0	293	91	20	4	0	33	CPU REPO
	XLarge	26,274	56	0.0	235	83	9	2	0	29	CPU REPO
File Download	***Small***	3,384	16	0.0	676	99	1	0	1	2	CPU REPO
	Medium	6,054	54	0.0	1,151	98	23	2	1	5	CPU REPO
	Large	10,350	80	0.0	982	97	18	3	0	5	CPU REPO
	XLarge	18,964	44	0.0	297	86	1	0	0	2	CPU REPO
Full Text Search within a site	***Small***	2,024	22	0.03	802	98	23	21	0	31	CPU REPO
	Medium	2,969	52	0.04	1,183	87	29	28	1	22	CPU REPO
	Large	5,497	88	0.05	1,108	85	27	26	0	9	CPU REPO
	XLarge	9,238	132	0.09	1,014	80	23	23	0	10	CPU REPO
Search by Date Range	***Small***	2,697	18	0.0	540	99	36	19	1	11	CPU REPO
	Medium	4,548	42	0.02	672	96	61	36	1	5	CPU REPO
	Large	9,751	90	0.05	697	94	61	36	0	4	CPU REPO

Test	Environment	Throughput per minute	Concurrent requests	Error rate, %	Response time, ms	CPU % Repository	CPU % Solr (coordin)	CPU % Solr (others)	CPU % TS	CPU % DB	Reason of saturation
within a site	*XLarge*	16,180	118	0.1	565	85	57	24	0	6	CPU REPO
Search by Text Property within a site	*Small*	2,244	22	0.2	751	98	39	34	0	27	CPU REPO
	Medium	3,147	52	0.22	1,232	88	52	50	2	19	CPU REPO
	Large	5,929	88	0.24	1,097	84	49	41	0	9	CPU REPO
	XLarge	10,038	146	0.27	1,084	81	41	35	0	10	CPU REPO

Table 19. Summary Table – 80 million nodes

The results show a couple of key points.

The first is that the system scales well – doubling the amount of compute resource available broadly results in a doubling of the throughput. The Metadata Update operation, which triggers a reindex of the affected node, doesn't quite achieve this perfect scaling profile, and this will be important to consider for use cases that are heavy on update operations.

The second pattern observed is that, on these test environments, it is always the CPU available on the repository machines that constrains the performance – even for the search operations. This means that when choosing to scale an ACS system, it is likely most helpful to scale the CPU on the repository instances first. Naturally, this is only true if the starting system is sized similarly to one of the test systems described here.

6.3. Performance Results for 500 Million Nodes

This section contains the results for small, medium, large, and extra-large for an ACS environment with ***500 million*** nodes.

6.3.1. In-place File Ingestion

Test Results

The first results show the ingestions per minute for 100-file batches. For example, a total of 375 100-file batch ingestions were completed in the ***extra-large*** environment before reaching the saturation point where no additional increases were observed.

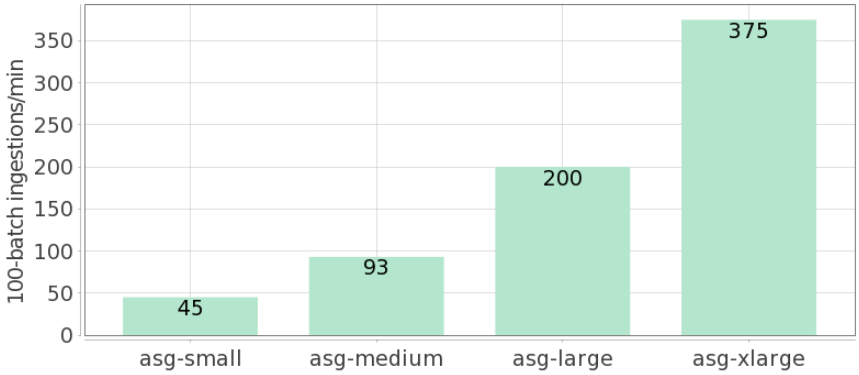

Figure 20. In-Place File Ingestion (100-file batches) per Minute – 500 million nodes

It is also important to understand the length of time required for batch ingestion. The following figure shows the average time in milliseconds for the ingestion of a 100-file batch for each environment tested.

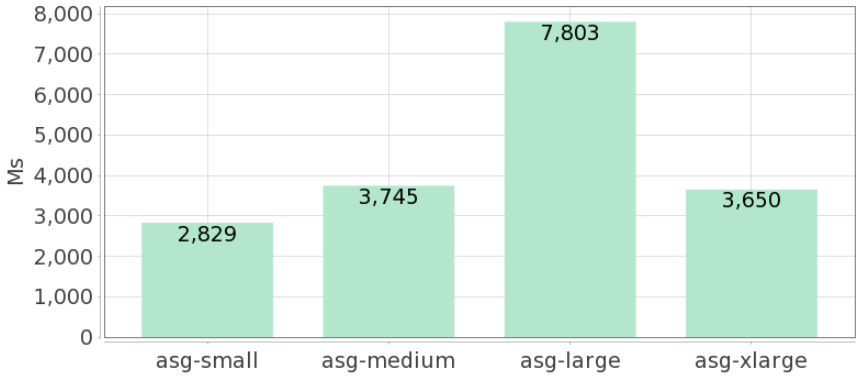

Figure 21. . Average Time for Ingesting a Batch of 100-files – 500 million nodes

6.3.2. File Upload

Test Results

The figure below shows the file uploads per minute for concurrent requests for each of the environments shown.

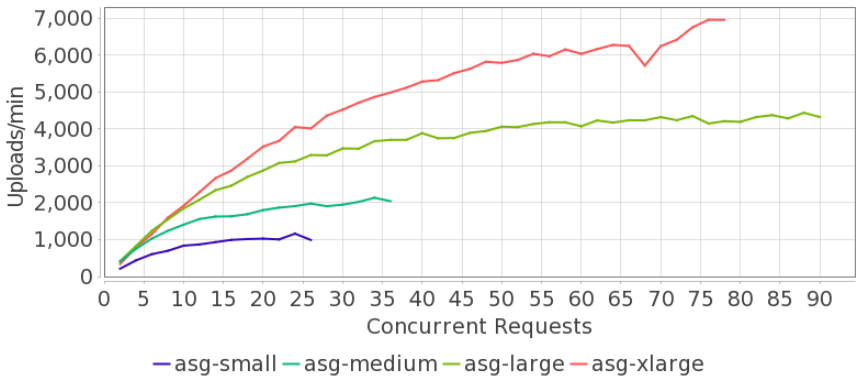

Figure 22. File Upload Operations per Minute – 500 million nodes

Another important metric when reviewing file upload is the response time for requests. The response time is provided in the next figure. These values represent the response time achieved for 95% of the tests.

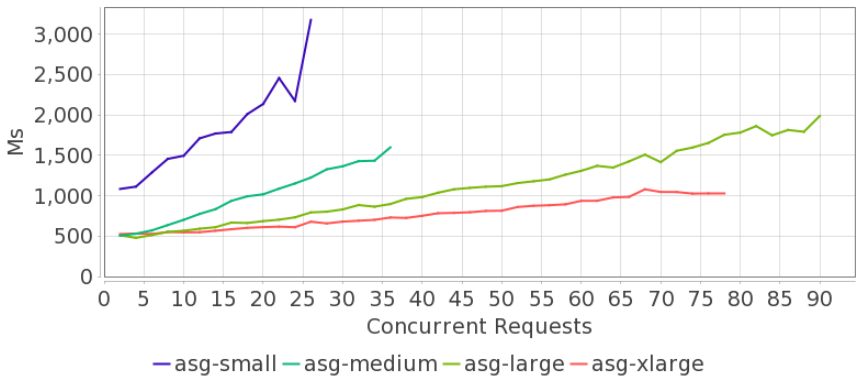

Figure 23. Response Times (95 Percentile) File Upload Operations – 500 million nodes

6.3.3. Metadata Update

Test Results

The graph below shows the results for updating metadata (properties) of nodes in the repository.

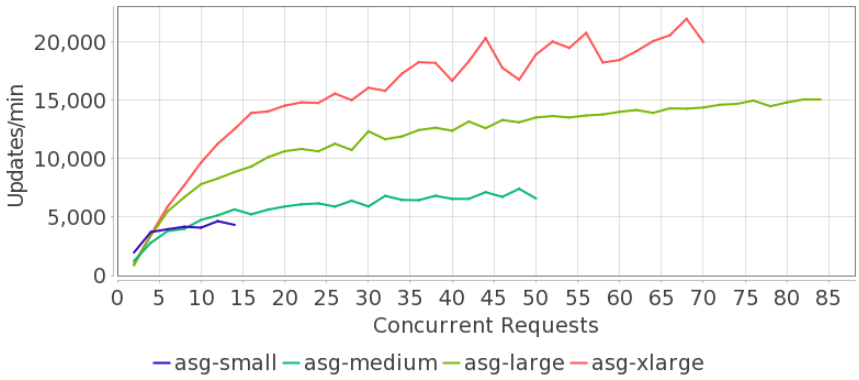

Figure 24. Metadata Update Operations per Minute – 500 million nodes

The response time is also provided in the next figure. These values represent the response time achieved for 95% of the tests.

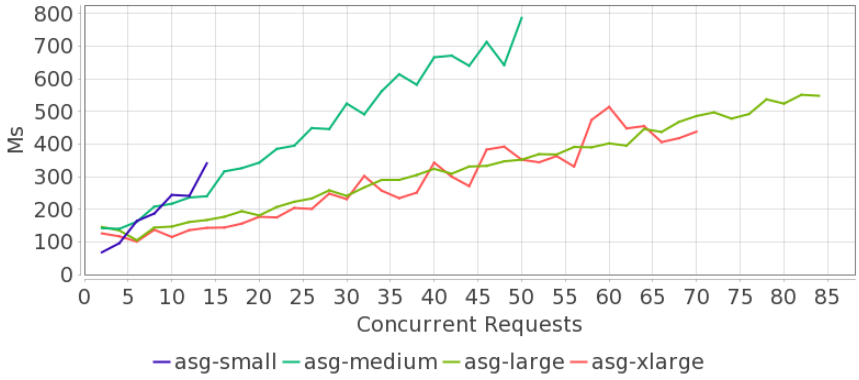

Figure 25. Response Times (95 Percentile) Metadata Update Operations – 500 million nodes

6.3.4. File Download

Test Results

The graph below shows the results for downloading files from the repository.

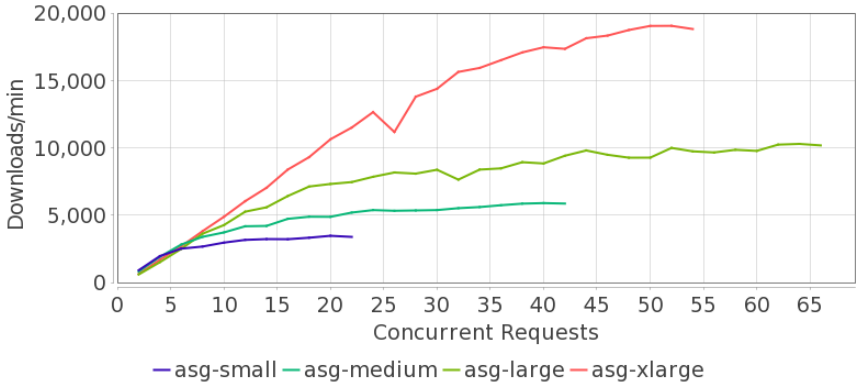

Figure 26. File Download Operations per Minute – 500 million nodes

The response time is also provided in the next figure. These values represent the response time achieved for 95% of the tests.

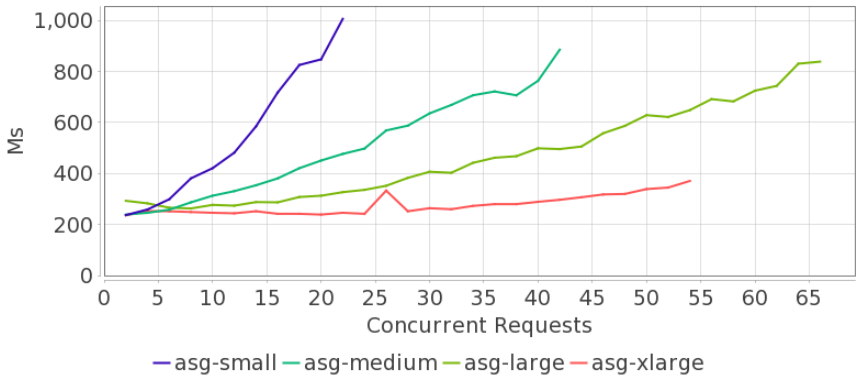

Figure 27. Response Times (95 Percentile) File Download Operations – 500 million nodes

6.3.5. Search

Test conditions

- 500 million files are in the Solr index.
- 55 million files are in each Solr shard.
- The number of documents per site is ~5-100k distributed randomly.
- Each individual search request is executed against a single site. Each test agent is allocated a site at random at the start of the test and uses that site for the duration of the test.

The following queries are used for search scenarios:

- Full Text Search within a Site:
  `"query": "SITE:\"${site}\" AND TEXT:${term}"`
- Search by Date Range within a Site:
  `"query": "SITE:\"${site}\" AND cm:modified:[NOW/DAY-${dateFrom} TO NOW/DAY+1DAY] AND TYPE:\"cm:content\""`
- Search by Text Property within a Site:
  `"query": "(SITE:\"${site}\" AND TYPE:\"cm:content\" AND cm:name:${term})"`

Full Text Search within a Site

Test Results

The graph below shows the results for the number of concurrent requests for full text searches executed within a site.

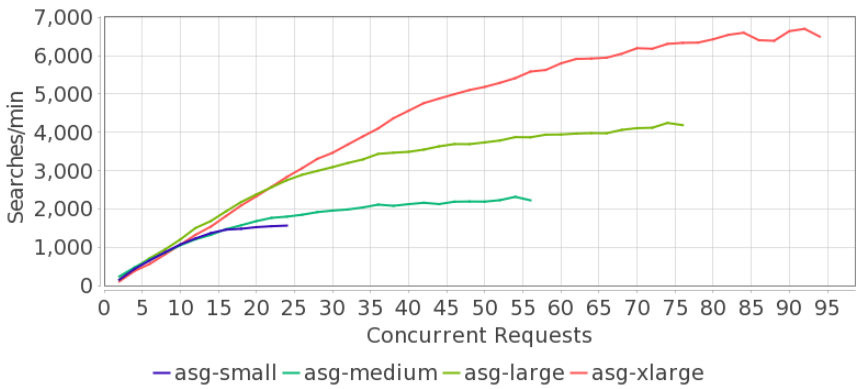

Figure 28. Full Text Search within a Site Operations per Minute – 500 million nodes

The response time is also provided in the next figure. These values represent the response time achieved for 95% of the tests.

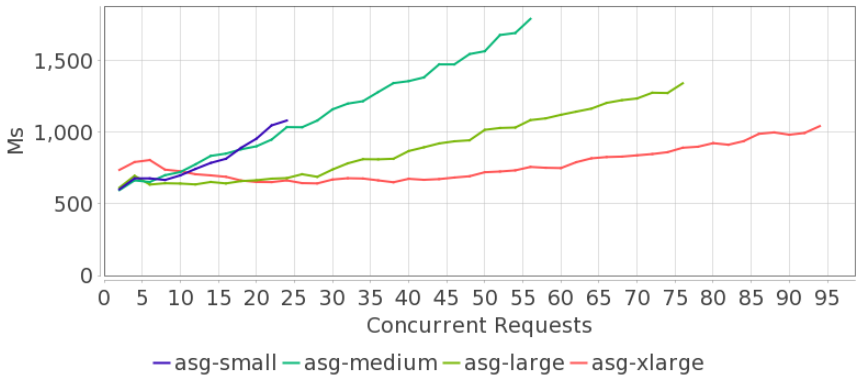

Figure 29. Response Times (95 Percentile) Full Text Search within a Site – 500 million nodes

Search by Date Range Within a Site

Test Results

The graph below shows the results for the number of concurrent requests for searches within a date range (such as all files created between 2019-01-02 and 2019-12-31) within a site.

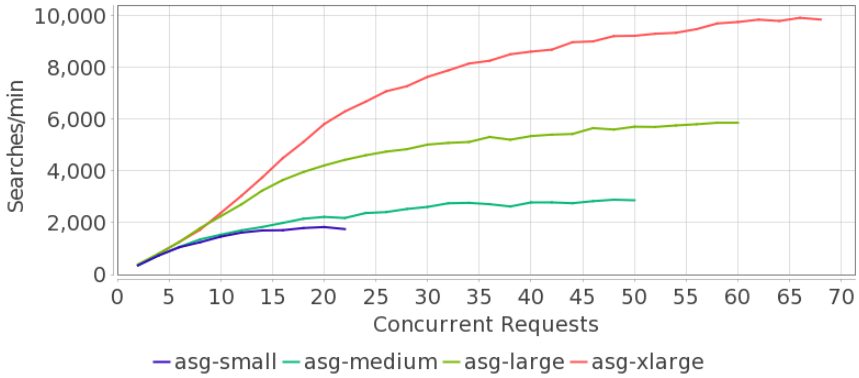

Figure 30. Search by Date Range within a Site Operations per Minute – 500 million nodes

The response time is also provided in the next figure. These values represent the response time achieved for 95% of the tests.

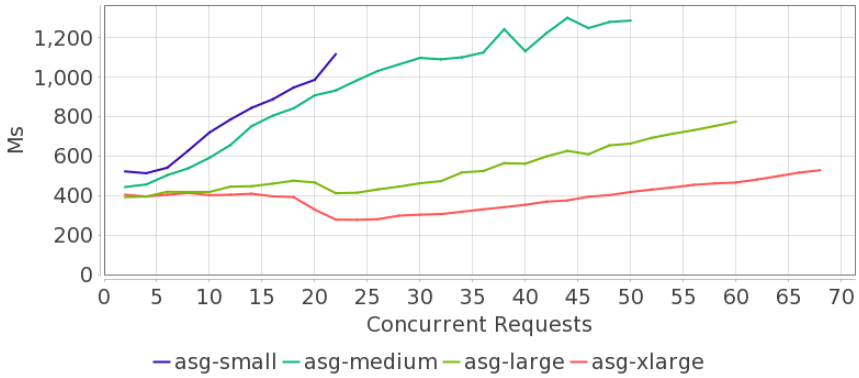

Figure 31. Response Times (95 Percentile) Search by Date Range within a Site – 500 million nodes

Search by Text Property within a Site

Test Results

The graph below shows the results for searches by a text property per minute within a site.

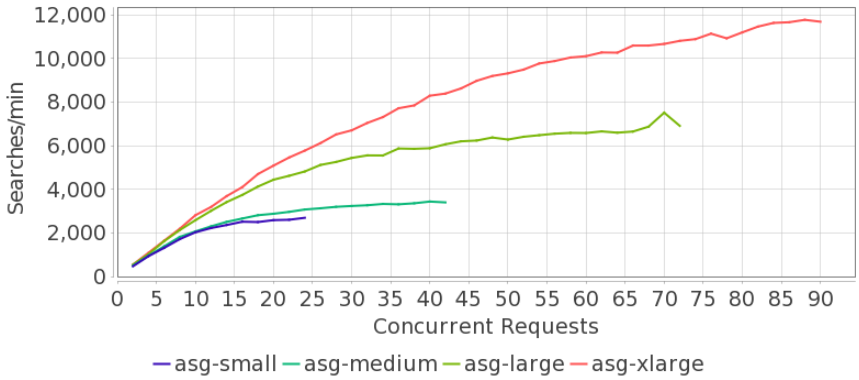

Figure 32. Search by Text Property within a Site Operations per Minute – 500 million nodes

The response time is also provided in the next figure. These values represent the response time achieved for 95% of the tests.

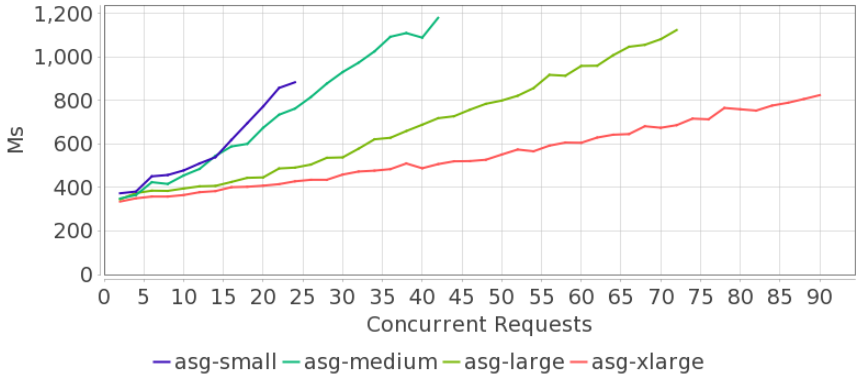

Figure 33. Response Times (95 Percentile) Search by Text Property within a Site – 500 million nodes

6.3.6. Summary

For ease of reviewing the information provided in this section, the table below summarizes all information and includes CPU utilization by component:

- Repository (Repo)
- Solr
- Transformation Server (TS)
- Database (DB)

Test	Environment	Throughput per minute	Concurrent requests	Error rate, %	Response time, ms	CPU % Repository	CPU % Solr (coordin)	CPU % Solr (others)	CPU % TS	CPU % DB	Reason of saturation
In-place File Ingestion (100-docs batch)	***Small***	45	15	0.0	2,829	92	8	4	98	45	CPU REPO
	Medium	93	27	0.0	3,745	91	7	6	98	57	CPU REPO
	Large	200	56	0.0	7,803	92	4	3	95	41	CPU REPO
	XLarge	375	76	0.01	3,650	87	3	2	75	41	CPU REPO
File Upload	***Small***	1,161	24	0.0	2,172	99	5	4	77	42	CPU REPO
	Medium	2,134	34	0.0	1,433	98	7	5	98	42	CPU REPO
	Large	4,438	88	0.01	1,791	96	4	3	94	36	CPU REPO
	XLarge	6,958	76	0.02	1,028	87	6	3	77	29	CPU REPO
Metadata Update	***Small***	4,654	12	0.0	241	99	18	7	0	85	CPU REPO
	Medium	7,429	48	0.0	642	94	26	7	1	78	CPU REPO
	Large	15,086	82	0.0	551	92	20	4	0	65	CPU REPO
	XLarge	21,990	68	0.0	418	81	10	2	0	53	CPU REPO
File Download	***Small***	3,471	20	0.0	848	99	0	0	0	2	CPU REPO
	Medium	5,901	40	0.0	764	98	5	1	1	17	CPU REPO
	Large	10,300	64	0.0	831	97	10	2	0	25	CPU REPO
	XLarge	19,075	52	0.0	345	86	0	0	0	2	CPU REPO
Full Text Search within a site	***Small***	1,548	22	0.04	1,048	98	18	11	1	36	CPU REPO
	Medium	2,315	54	0.09	1,693	84	29	15	1	23	CPU REPO
	Large	4,245	74	0.23	1,274	82	22	13	1	25	CPU REPO
	XLarge	6,702	92	0.42	995	76	17	9	0	50	CPU REPO
Search by Date Range	***Small***	1,833	20	0.0	986	99	25	15	0	20	CPU REPO
	Medium	2,888	48	0.12	1,279	91	67	22	1	9	CPU REPO
	Large	5,859	58	0.27	752	87	57	18	1	4	CPU REPO

Test	Environment	Throughput per minute	Concurrent requests	Error rate, %	Response time, ms	CPU % Repository	CPU % Solr (coordin)	CPU % Solr (others)	CPU % TS	CPU % DB	Reason of saturation
within a site	*XLarge*	9,911	66	0.51	516	81	42	13	0	5	CPU REPO
Search by Text Property within a site	*Small*	2,601	22	0.2	857	98	58	45	0	49	CPU REPO
	Medium	3,435	40	0.33	1,088	81	96	58	1	16	CPU SOLR (Coordinator)
	Large	7,509	70	0.51	1,081	83	69	50	0	19	CPU REPO
	XLarge	11,766	88	0.82	806	76	61	38	0	35	CPU REPO

Table 34. Summary Table – 500 million nodes

The general patterns observed with the 500-million repository are the same as for the 80-million repository, namely that the scaling profile is very good and that the CPU on the repository instances is normally the limiting factor (with one exception in this case).

The other information that can be gathered when comparing these results with those from the 80-million repository is that there is a small reduction in performance when the repository grows to 500-million documents. In general, the throughput for this size of repository is around 10-20% lower than for the 80-million repository. The performance is still good – supporting the ingestion of 37500 documents per minute and updating 22000 documents per minute on the XLarge environment, for example – and this information can be used to plan system enhancements as a repository grows.

6.4. Performance Results for 1 Billion Nodes

This section contains the results for small, medium, large, and extra-large for an ACS environment with ***1 billion*** nodes.

6.4.1. In-place File Ingestion

Test Results

The first results show the ingestions per minute for 100-file batches. For example, a total of 45 100-file batch ingestions were completed in the ***large*** environment before reaching the saturation point where no additional increases were observed.

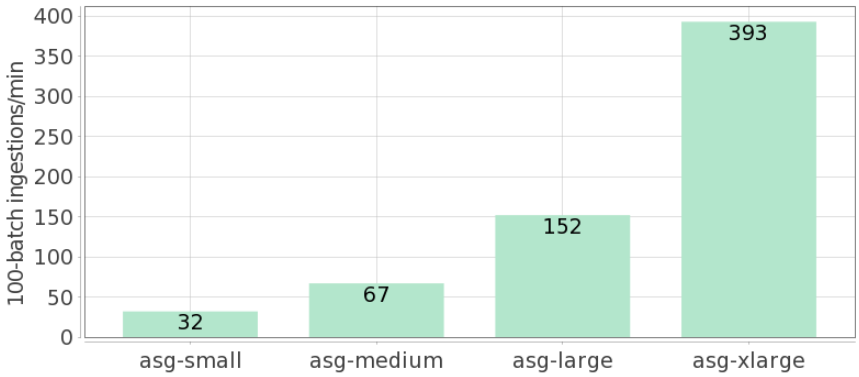

Figure 35. In-Place File Ingestion (100-file batches) per Minute – 1 billion nodes

It is also important to understand the length of time required for batch ingestion. The following figure shows the average time in milliseconds for the ingestion of a 100-file batch for each environment tested.

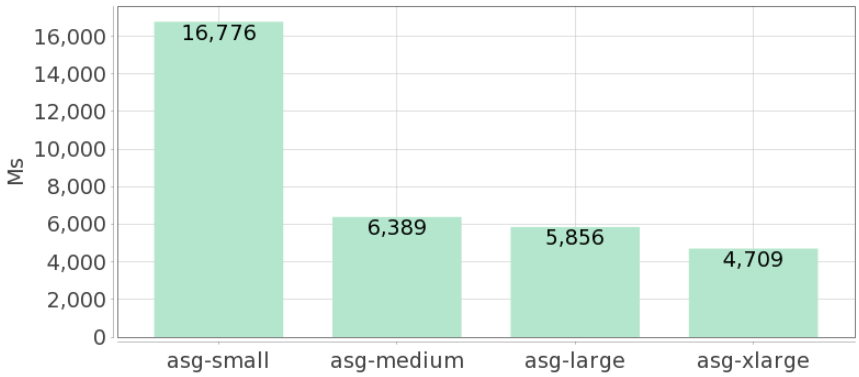

Figure 36. Average Time for Ingesting a Batch of 100-files – 1 billion nodes

6.4.2. File Upload

Test Results

The figure below shows the file uploads per minute for concurrent requests for each of the environments shown.

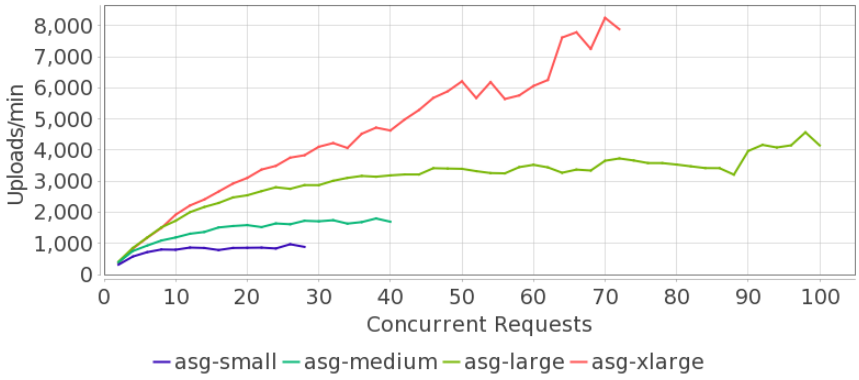

Figure 37. File Upload Operations per Minute -1 billion nodes

Another important metric when reviewing file upload is the response time for requests. The response time is provided in the next figure. These values represent the response time achieved for 95% of the tests.

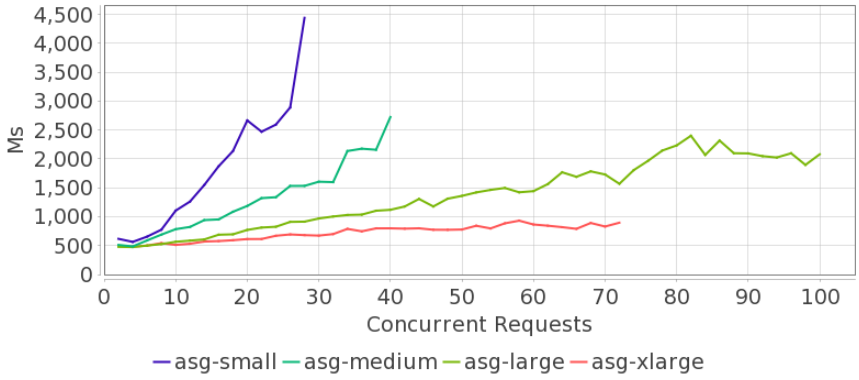

Figure 38. Response Times (95 Percentile) File Upload Operations – 1 billion nodes

6.4.3. Metadata Update

Test Results

The graph below shows the results for updating metadata (properties) of nodes in the repository.

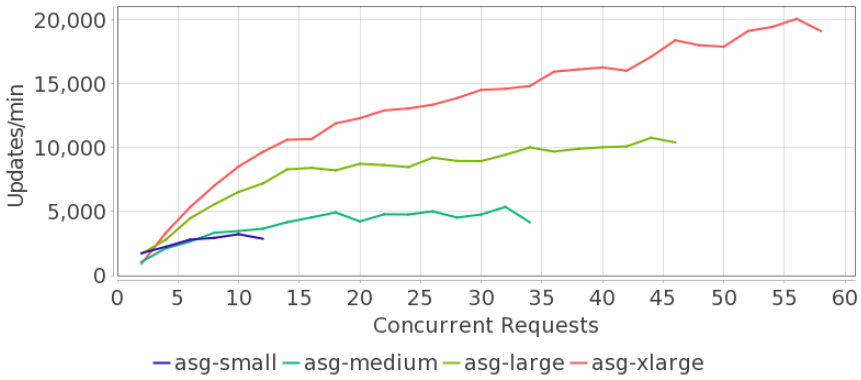

Figure 39. Metadata Update Operations per Minute – 1 billion nodes

The response time is also provided in the next figure. These values represent the response time achieved for 95% of the tests.

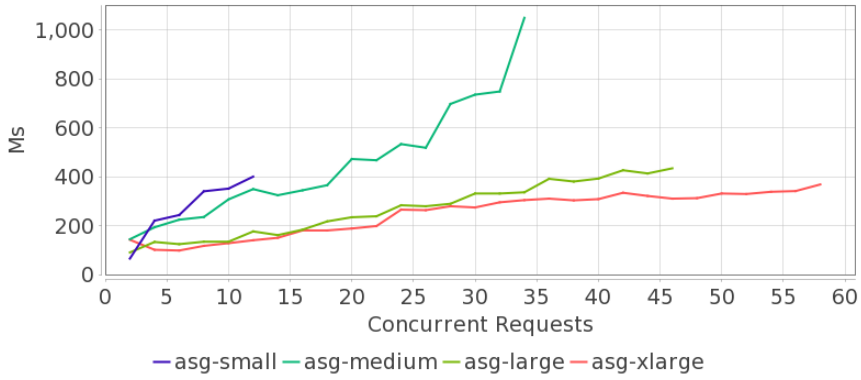

Figure 40. Response Times (95 Percentile) Metadata Update Operations – 1 billion nodes

6.4.4. File Download

Test Results

The graph below shows the results for downloading files from the repository.

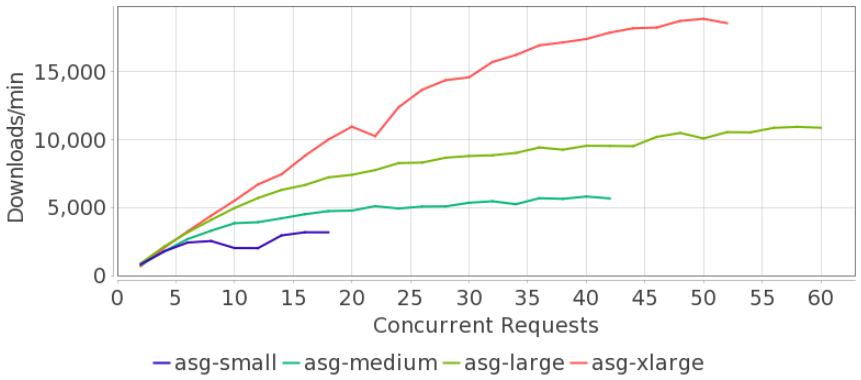

Figure 41. File Download Operations per Minute – 1 billion nodes

The response time is also provided in the next figure. These values represent the response time achieved for 95% of the tests.

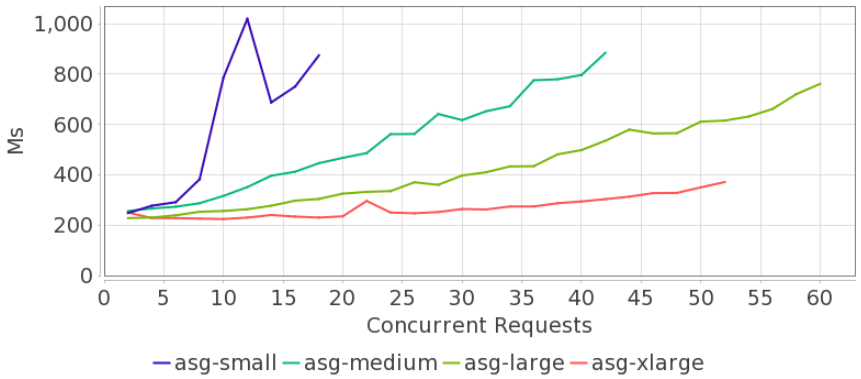

Figure 42. Response Times (95 Percentile) File Download Operations – 1 billion nodes

6.4.5. Search

Test conditions

- 1 billion files are in the Solr index.
- 58 million files are in each Solr shard.
- The number of documents per site is ~5-100k distributed randomly.
- Each individual search request is executed against a single site. Each test agent is allocated a site at random at the start of the test and uses that site for the duration of the test.

The following queries are used for search scenarios:

- Full Text Search within a Site:
  `"query": "SITE:\"${site}\" AND TEXT:${term}"`
- Search by Date Range within a Site:
  `"query": "SITE:\"${site}\" AND cm:modified:[NOW/DAY-${dateFrom} TO NOW/DAY+1DAY] AND TYPE:\"cm:content\""`
- Search by Text Property within a Site:
  `"query": "(SITE:\"${site}\" AND TYPE:\"cm:content\" AND cm:name:${term})"`

Full Text Search within a Site

Test Results

The graph below shows the results for the number of concurrent requests for full text searches executed within a site.

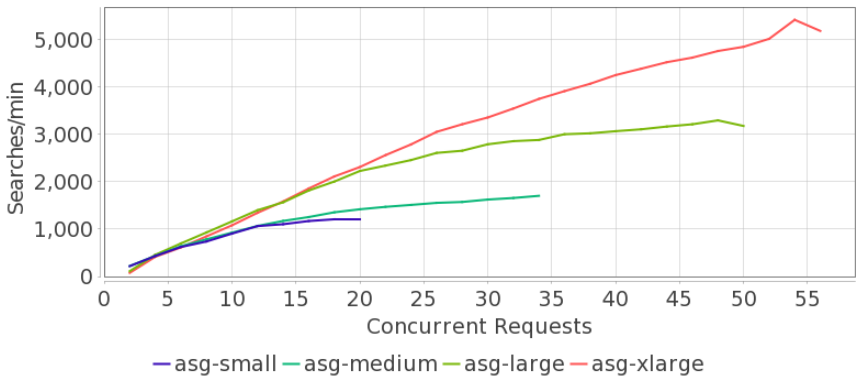

Figure 43. Full Text Search within a Site Operations per Minute – 1 billion nodes

The response time is also provided in the next figure. These values represent the response time achieved for 95% of the tests.

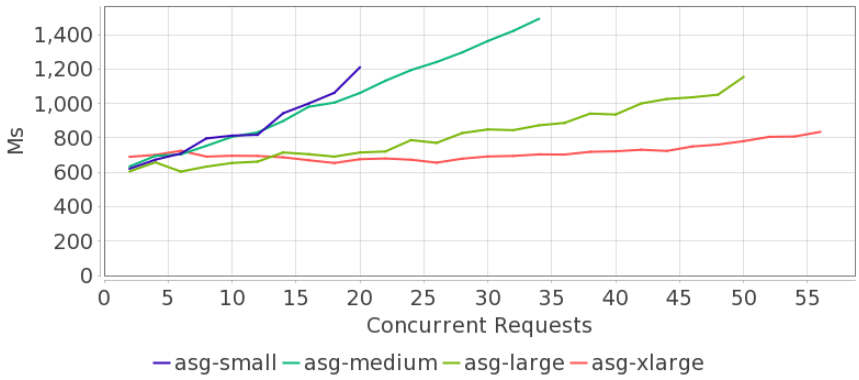

Figure 44. Response Times (95 Percentile) Full Text Search within a Site – 1 billion nodes

Search by Date Range Within a Site

Test Results

The graph below shows the results for the number of concurrent requests for searches within a date range (such as all files created between 2019-01-02 and 2019-12-31) within a site.

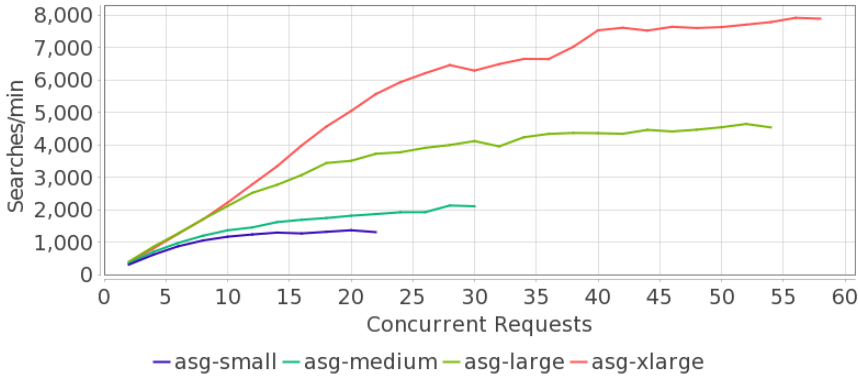

Figure 45. Search by Date Range within a Site Operations per Minute – 1 billion nodes

The response time is also provided in the next figure. These values represent the response time achieved for 95% of the tests.

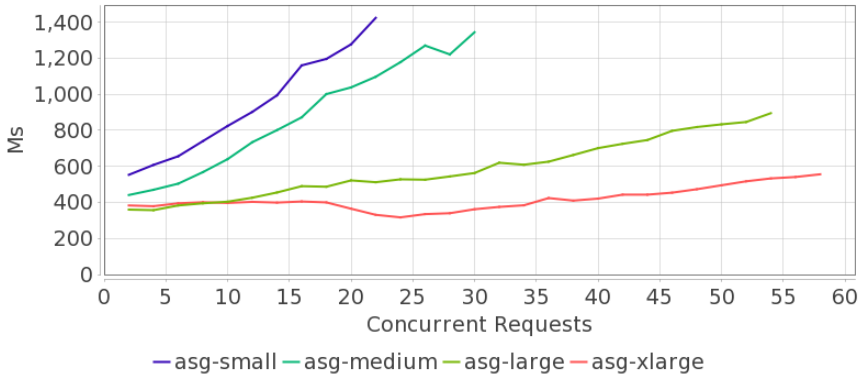

Figure 46. Response Times (95 Percentile) Search by Date Range within a Site – 1 billion nodes

Search by Text Property within a Site

Test Results

The graph below shows the results for searches by a text property per minute within a site.

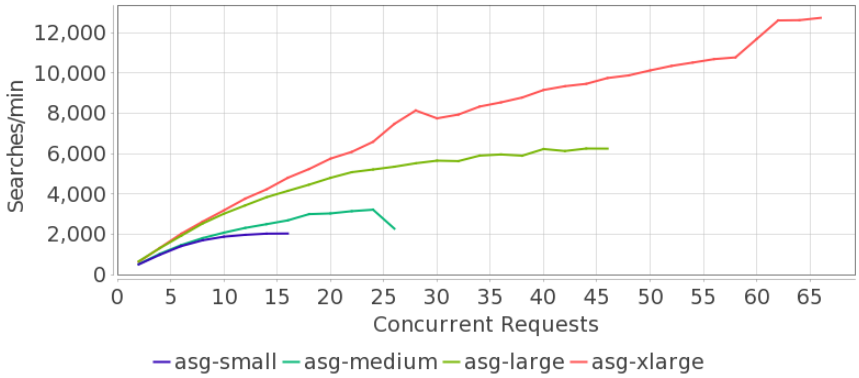

Figure 47. Search by Text Property within a Site Operations per Minute – 1 billion nodes

The response time is also provided in the next figure. These values represent the response time achieved for 95% of the tests.

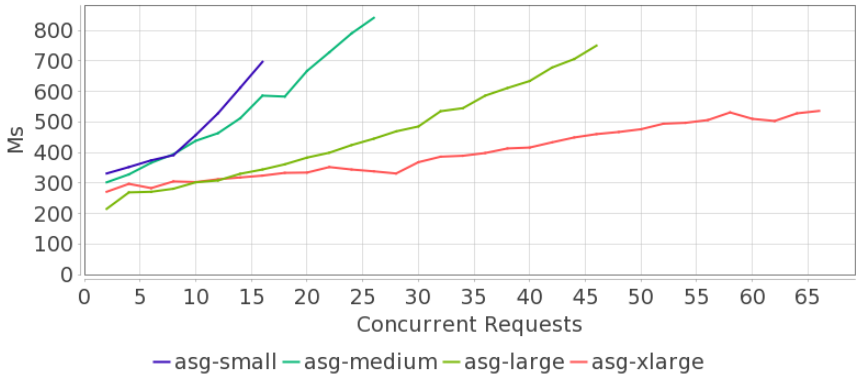

Figure 48. Response Times (95 Percentile) Search by Text Property within a Site – 1 billion nodes

6.4.6. Summary

For ease of reviewing the information provided in this section, the table below summarizes all information and also includes CPU utilization by component:

- Repository (Repo)
- Solr
- Transformation Server (TS)
- Database (DB)

Test	Environment	Throughput per minute	Concurrent requests	Error rate, %	Response time, ms	CPU % Repository	CPU % Solr (coordin)	CPU % Solr (others)	CPU % TS	CPU % DB	Reason of saturation
In-place File Ingestion (100-docs batch)	***Small***	32	16	0.0	16,776	98	14	4	97	52	CPU REPO
	Medium	67	30	0.0	6,389	97	8	4	98	62	CPU REPO
	Large	152	45	0.0	5,856	97	3	2	95	49	CPU REPO
	XLarge	393	81	0.0	4,709	87	2	1	76	42	CPU REPO
File Upload	***Small***	973	26	0.0	2,890	99	32	18	69	48	CPU REPO
	Medium	1,802	38	0.0	2,158	99	6	4	98	62	CPU REPO
	Large	4,573	98	0.0	1,896	97	5	3	94	62	CPU REPO
	XLarge	8,256	70	0.0	830	78	2	1	70	43	CPU REPO
Metadata Update	***Small***	3,233	10	0.01	351	99	12	7	1	84	CPU REPO
	Medium	5,372	32	0.0	748	98	17	7	1	95	CPU REPO
	Large	10,786	44	0.0	413	87	10	3	0	82	CPU REPO
	XLarge	20,106	56	15.52	341	80	6	3	0	76	CPU REPO
File Download	***Small***	3,184	16	0.0	750	98	1	0	0	3	CPU REPO
	Medium	5,818	40	0.0	796	98	2	1	1	23	CPU REPO
	Large	10,949	58	0.0	720	96	0	0	0	1	CPU REPO
	XLarge	18,909	50	0.0	350	85	1	0	0	2	CPU REPO
Full Text Search within a site	***Small***	1,207	18	0.03	1,062	98	12	8	0	41	CPU REPO
	Medium	1,656	32	0.17	1,422	92	22	11	1	25	CPU REPO
	Large	3,293	48	0.44	1,051	84	19	9	1	27	CPU REPO
	XLarge	5,414	54	0.84	808	73	15	7	0	43	CPU REPO
Search by Date Range	***Small***	1,365	20	0.0	1,277	98	16	11	0	20	CPU REPO
	Medium	2,128	28	0.26	1,221	94	24	16	1	9	CPU REPO
	Large	4,644	52	0.49	846	88	24	14	0	5	CPU REPO

Test	Environment	Throughput per minute	Concurrent requests	Error rate, %	Response time, ms	CPU % Repository	CPU % Solr (coordin)	CPU % Solr (others)	CPU % TS	CPU % DB	Reason of saturation
within a site	*XLarge*	7,919	56	1.04	541	79	19	11	0	6	CPU REPO
Search by Text Property within a site	*Small*	2,025	14	0.21	612	99	44	32	1	13	CPU REPO
	Medium	3,213	24	0.61	790	91	78	54	1	9	CPU REPO
	Large	6,246	44	1.04	706	84	68	42	0	9	CPU REPO
	XLarge	12,613	64	1.8	528	80	62	37	0	13	CPU REPO

Table 49. Summary Table – 1 billion nodes

The key takeaways for these results are very similar to those for the 80-million and 500-million repositories. Scaling is very good and the CPU on the repository instances is the bottleneck.

There is a small degradation of performance on some of the operations when compared to the 500-million repository – typically around 5-10% - but the performance remains good.

7. Summary

7.1. Interpretation

As previously mentioned, all this data can be overwhelming when viewed out of context, so we have provided some examples on how to use the data in this document to apply the findings to your own requirements. Reading the results found in this document is only part of the challenge – there is also some need to interpret these results to answer specific questions you might have about your own configuration. This section shows some examples in how to read certain graphs and results to answer some typical questions.

7.1.1. Use Case 1

Let's assume that a customer expects to have 400 concurrent users and anticipates approximately 1,500 searches per minute at their peak. What size system would this customer need based on the following performance results provided in the graph below?

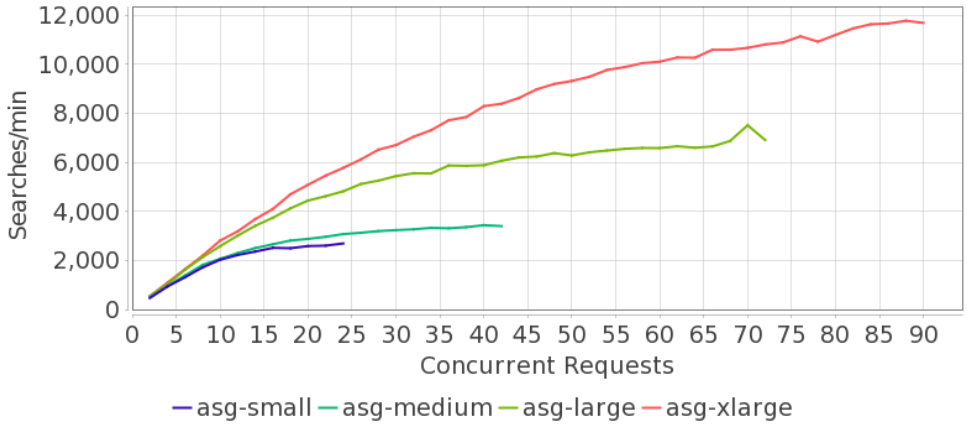

Figure 50. Use Case 1 – Searches per Minute

First, we must remember that concurrent users are not the same as concurrent requests. We must use the multiplier suggested of approximately 6 to 12 times to obtain the concurrent requests for 400 concurrent users. As shown below, this would be between 33 and 66 concurrent requests.

400 concurrent users / 12 = 33 concurrent requests

400 concurrent users / 6 = 66 concurrent requests

So, we will block off the area between 33 and 66 concurrent requests and put a line at 1,500 searches per minute (as shown in the figure below with the light blue lines and highlighted region).

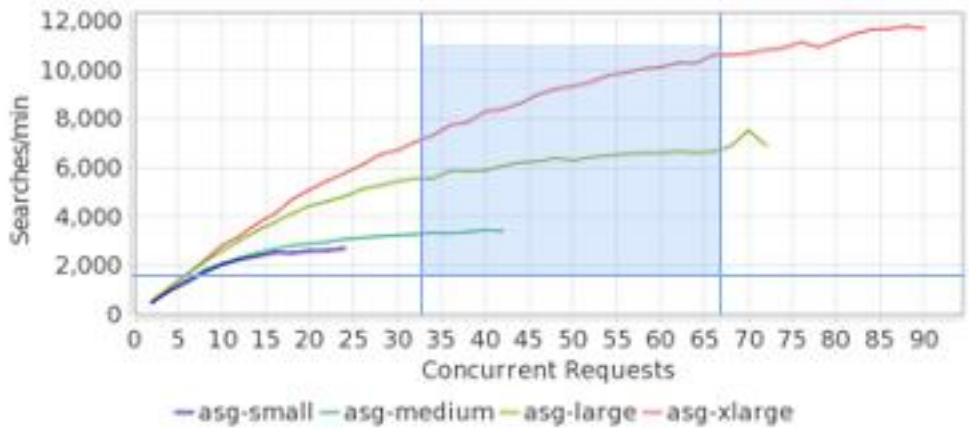

Figure 51. Use Case 1 (with Acceptable Region) – Searches per Minute

As shown, neither the small nor the medium configuration would not be sufficient to fulfil this request. Notice that the medium configuration reaches saturation just under 4,000 searches/minute which would only meet the requirements at the low end of the concurrent requests to concurrent user ration. However, both the large and the extra-large would meet this requirement.

7.1.2. Use Case 2

For this use case, we will be using the same requirements at the first case (400 concurrent users with 1,500 searches/minute) and add in a required response time of under a second.

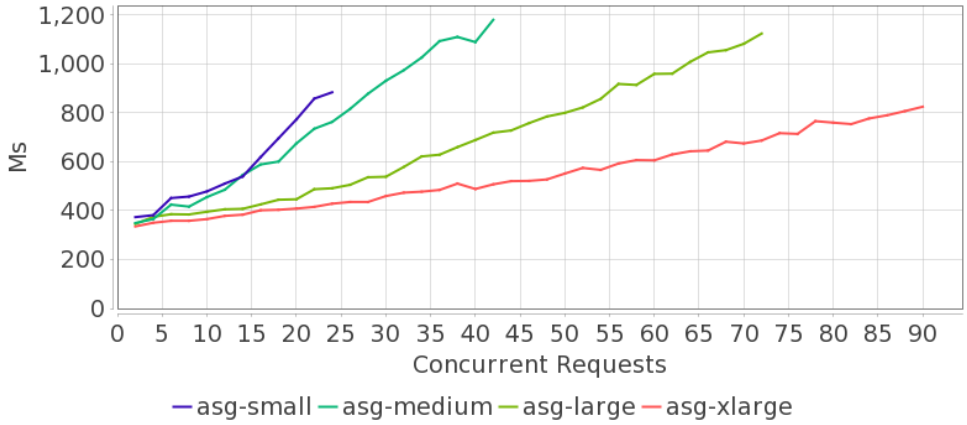

Figure 52. Use Case 2 – Search Response Time

Keeping the 33 to 66 concurrent requests (for the 400 concurrent users from case 1), we have drawn the gold line at the 1,000 millisecond (1 second) response time and then vertical lines at 33 and 66 concurrent requests.

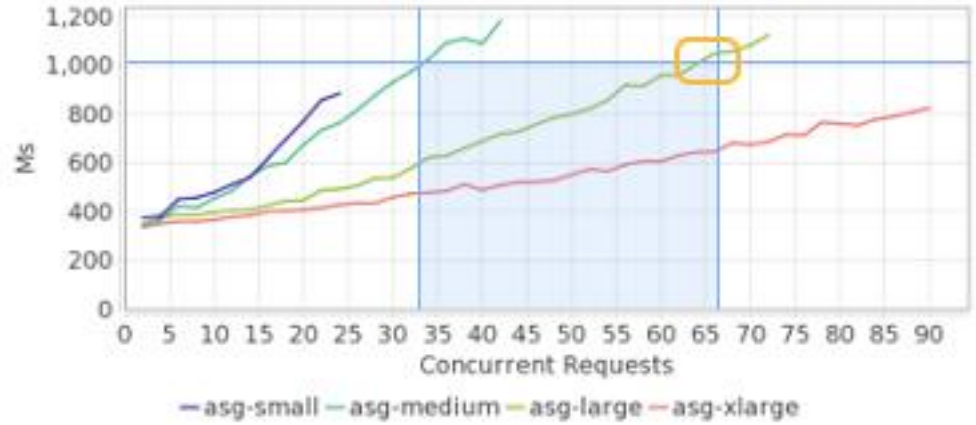

Figure 53. Use Case 2 (with Acceptable Region) – Search Response Time

If you review the graph, neither the small nor medium environments would provide the required response time as neither appears in the blue shaded area. If you look at the gold box, you see that even the large environment may not meet the requirement at the high end of the operations. In this case, the required environment would be an extra-large environment.

So, looking at the requirements for Use Case 1 and combining that with Use Case 2 – the required environment would be an extra-large to meet both requirements. This makes it clear that you need to review peak throughput, concurrency, and the required response time.

7.1.3. Use Case 3

Our final use case provides us with a chance to extrapolate the results found to determine what might happen in the core environment is changed. In this case, we will use a new graph to show the extrapolation. Reviewing the information about each environment found in the Environment Configuration table on page 41 – how many concurrent requests can be achieved if the customer went from 16 to 32 cores in his/her repository environment while maintaining less than one second response time?

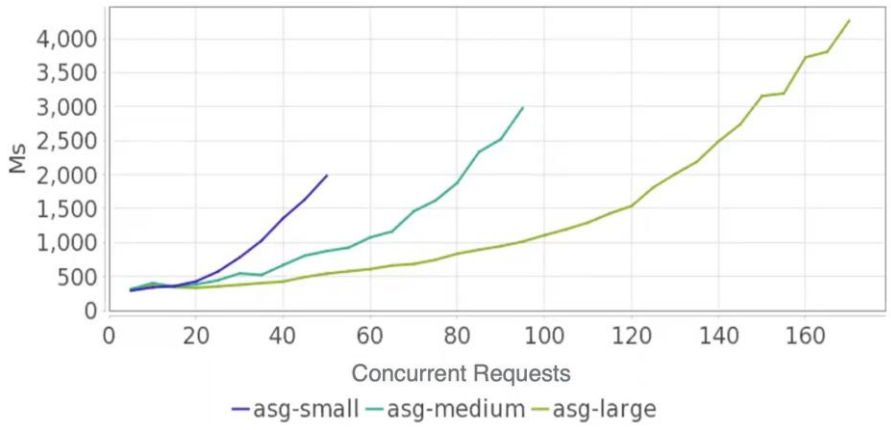

Figure 54. Use Case 3 – Search Response Time

In the graph below, we have indicated where a 4-core, 8-core and 16-core environments will meet the one second response time. Based on this information, we can assume that the 32-core environment falls in the position noted on the far right (red line) in the graph.

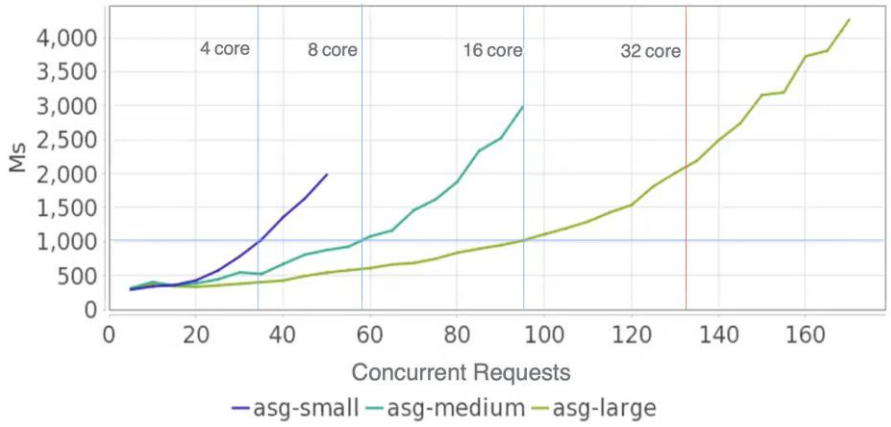

Figure 55. Use Case 3 (Core Delineation) – Search Response Time

7.2. Recommendations

As can be seen in this document, there is no true "formula" that can be applied that will provide the exact sizing figure that are required for a properly configured architecture. As we have documented, sizing and capacity planning depends on many factors that are different from project to project. These include:

- Number of customizations
- Types of customizations
- External systems integrations
- Network topology
- Architecture design
- Operating system specifics
- Database vendor
- User behaviour
- Other project specific factors

The only sure way to properly size your environment is to have dedicated benchmarks that reproduce all production conditions as accurately as possible. However, we hope this document was able to provide some additional insight into the different layers of the Alfresco application of a normal deployment that organizations can plan accordingly.

7.3. Summary

As is apparent in the graphs and performance output, sizing a system depends greatly on your specific requirement and details on not only concurrency of operations, but user concurrency, peak throughput as well as maximum response time is necessary to determine the optimal size of a system. In addition, you must take future volumes into consideration when making these recommendations.

For example, as these performance tests were conducted, it was found that from a pure performance point of view, scaling up the repository until you get to 16 cores and then scaling out provides the best options for increased performance.

There are many variables for which we can test performance. For example, how is performance impacted by the type of access control list (ACL) structure – simple or complex? What about the

complexity of the registered users and groups? Does performance change as you work with only media files versus only text files and what is impacted? The diagram below shows the variables that have been considered.

We have conducted performance testing using a range for some variables, and a fixed value for others. The red lines below show where testing has been conducted across a range of values, and the black lines with red bars show where a variable has been fixed for all performance tests and the red lines in the figure below are tests that we have already conducted. The red tick marks on the lines show the level of complexity in that line of testing.

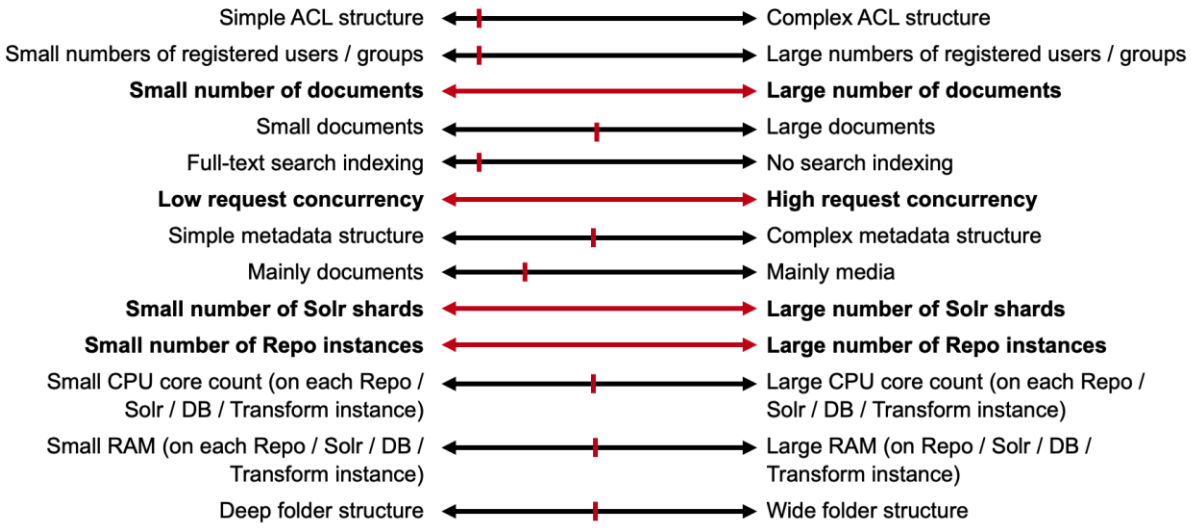

Figure 56. Axes of Variability

As part of our testing, there are many things that we added to the tests and some items where we have only begun to test and investigate. This type of variability can affect performance and should be considered when reviewing your requirements and performance expectation. Testing on these additional items will continue to be added over time.

8. Appendix 1 - Environment Information

The following information provides additional data on the configuration for each environment used for the testing reported in this document.

8.1. Product Deployment information

Service	Components
Database	Amazon Aurora PostgreSQL 11.8
Repository	Alfresco Share v 7.0.0.1 Alfresco Content Services v 7.0.0.1 Alfresco Digital Workspace v 7.0.0.1 Alfresco Repo v 7.0.0.1
Search	Alfresco Search Services v 1.4.3.3
Transformation Server	Alfresco Transformation Services v 1.3.2

Table 57. Version Deployment Information

8.2. Deployment Structure

As mentioned, the Alfresco environment was deployed using Amazon Web Services (AWS) using the following high-level structure.

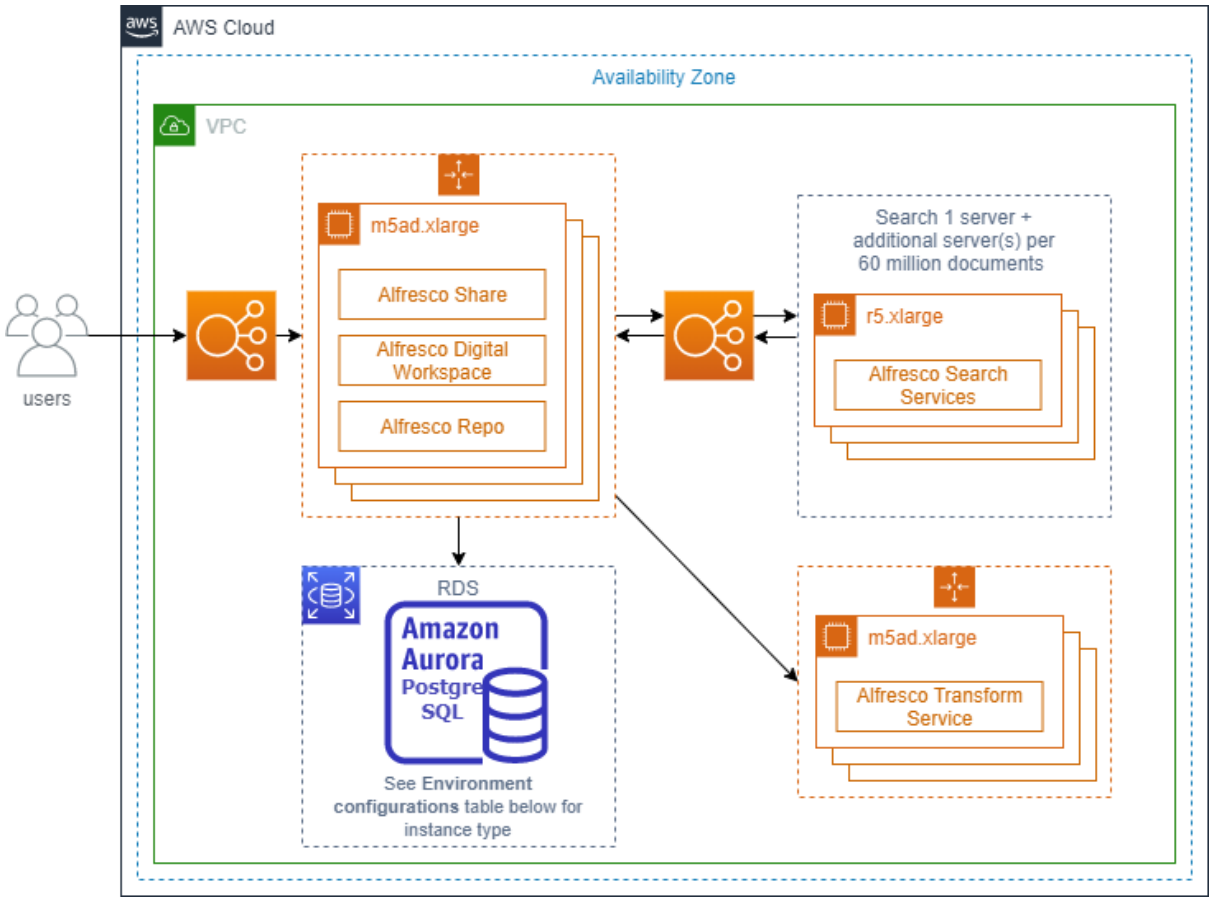

Figure 58. Alfresco in AWS Deployment Structure

8.2.1. Instances in Each Environment

Each environment was configured to scale by adding AWS instances to the environment. The number of instances for each service is listed in the following table.

Service	Small	Medium	Large	Extra Large
Repo	1	2	2	2
Search	2	2	2	2
Transformation	1	1	1	1

Table 59. Number of AWS Instances for Each Service

8.2.2. Versions of Supported Software

The table below shows the versions of software used in the test environments

Package Name	Package Version
aims_version	1.2.0
alfresco_distribution_version	7.0.0.1
alfresco_enterprise_repository_version	8.344
alfresco_imagemagick_version	2.1.0
alfresco_libreoffice_version	2.1.0
alfresco_pdfrenderer_version	2.1.0
alfresco_shared_file_store_controler_version	0.8.0
alfresco_tika_version	2.1.0
alfresco_transform_misc_version	2.1.0
aos_version	1.3.1
api_explorer_version	6.2.0
digital_workspace_version	1.4
imagemagick_version	7.0.7-27
insight_engine_version	1.4.3.3
insight_zeppelin_version	1.4.3.3
jre_version	1.8.0_25
libreoffice_path_version	6.3
libreoffice_version	6.3.5
mysql_version	8.0.19
openjdk_version	11.0.2
pdfrenderer_version	1.1
pg_version	42.2.6
saml_version	1.2.0
tomcat_version	9.0.41
transform_server_router_version	1.3.2
transform_service_libreoffice_version	6.3.5.2

Table 60. Package Versions for ACS 7.0.0.1 Test Environments

8.3. Environment configurations

Service	Configuration	Small	Medium	Large	Extra-Large
Bastion	***Instances***	1	1	1	1
	Image	ami-0c447cab0e5538287	ami-0c447cab0e5538287	ami-0c447cab0e5538287	ami-0c447cab0e5538287
	Instance type	t2.micro	t2.micro	t2.micro	t2.micro
	Max Instances	1	1	1	1
	Root Disk IOPS	1200	1200	1200	1200
	Root Disk Size	400	400	400	400
	Root Disk Type	gp2	gp2	gp2	gp2
RDS	***Engine***	aurora-postgresql	aurora-postgresql	aurora-postgresql	aurora-postgresql
	Engine version	11.8	11.8	11.8	11.8
	Members / Instances	1	1	1	1
	Availability zone	eu-west-2b	eu-west-2b	eu-west-2b	eu-west-2b
	Instance size	db.r5.2xlarge	db.r5.4xlarge	db.r5.8xlarge	db.r5.12xlarge
	Encrypted storage	false	false	false	false
	Writer	true	true	true	true
Repo	***Instances***	1	2	2	2
	Image	ami-05b6818a652cc9351	ami-05b6818a652cc9351	ami-05b6818a652cc9351	ami-05b6818a652cc9351
	Instance type	m5a.xlarge	m5a.xlarge	m5a.2xlarge	m5a.4xlarge
	Max Instances	1	2	2	2
	Root Disk IOPS	0	0	0	0
	Root Disk Size	600	600	600	600
	Root Disk Type	gp2	gp2	gp2	gp2
S3 bucket	***Bucket***	env-acs-large80-repo-bucket	env-acs-large80-repo-bucket	env-acs-large80-repo-bucket	env-acs-large80-repo-bucket
Search	***Instances***	2	2	2	2
	EBS Disk IOPS	9000	9000	9000	9000
	EBS Disk Size	3000	3000	3000	3000
	EBS Disk Type	gp2	gp2	gp2	gp2
	Image	ami-0cc05bd04fcfd1673	ami-0cc05bd04fcfd1673	ami-0cc05bd04fcfd1673	ami-0cc05bd04fcfd1673
	Instance type	r5.xlarge	r5.xlarge	r5.2xlarge	r5.4xlarge
	Max Instances	2	2	2	2
	Root Disk IOPS	0	0	0	0
	Root Disk Size	150	150	150	150
	Root Disk Type	gp2	gp2	gp2	gp2

Service	Configuration	Small	Medium	Large	Extra-Large
Transformation	***Instances***	1	1	1	1
	Image	ami-090f52f2f186bfaab	ami-090f52f2f186bfaab	ami-090f52f2f186bfaab	ami-090f52f2f186bfaab
	Instance type	m5a.xlarge	m5a.xlarge	m5a.2xlarge	m5a.xlarge
	Max Instances	1	1	1	1
	Root Disk IOPS	0	0	0	0
	Root Disk Size	150	150	150	150
	Root Disk Type	gp2	gp2	gp2	gp2

Table 61. Environment Configuration

For more detailed information on each of the AWS Elastic Compute Cloud (EC2) instance types mentioned in the above table, refer to the Amazon website: https://aws.amazon.com/ec2/instance-types/. For the Relational Database Service (RDS) instance types, more information can be found here: https://aws.amazon.com/rds/instance-types/.

A note about virtual CPU (vCPU) and cores from the Amazon EC2 Documentation:

> *Amazon EC2 instances support multithreading, which enables multiple threads to run concurrently on a single CPU core. Each thread is represented as a virtual CPU (vCPU) on the instance. An instance has a default number of CPU cores, which varies according to instance type. For example, an m5.xlarge instance type has two CPU cores and two threads per core by default—four vCPUs in total.*

You can review the default vCPU and default CPU cores by instance type here: https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/instance-optimize-cpu.html#cpu-options-gen-purpose.

8.4. Server Configuration

Repository

```
JAVA_MEMORY_OPTS = -Xms8192M -Xmx8192M
JAVA_OPTS =  -XX:+DisableExplicitGC -XX:+UseG1GC -XX:ReservedCodeCacheSize=128m -
Djava.net.preferIPv4Stack=true -Djava.awt.headless=true -Dalfresco.home=/opt/alfresco-
content-services -XX:ReservedCodeCacheSize=128m -Dcom.sun.management.jmxremote
```

Search

```
JAVA_OPTS = -Xms4096M -Xmx4096M -XX:+DisableExplicitGC -XX:+UseG1GC -
XX:ReservedCodeCacheSize=128m -Djava.net.preferIPv4Stack=true -Djava.awt.headless=true -
Dalfresco.home=/opt/alfresco-content-services -XX:ReservedCodeCacheSize=128m -
Dcom.sun.management.jmxremote
SOLR_JAVA_MEM = -Xms20480M -Xmx20480M
```

Transformation

```
IMAGEMAGICK_JAVA_OPTS = -Dserver.port= -Xms512m -Xmx512m
LIBREOFFICE_JAVA_OPTS = -Dserver.port= -Xms1G -Xmx2G
PDF_RENDERER_JAVA_OPTS = -Dserver.port= -Xms1G -Xmx1G
SHARED_FILE_STORE_JAVA_OPTS = -Dserver.port= -Xms512m -Xmx512m -DfileStorePath=/opt/efs
TIKA_JAVA_OPTS = -Dserver.port= -Xms2G -Xmx3G
TRANSFORM_ROUTER_JAVA_OPTS = -Dserver.port= -Xms2G -Xmx4G
```

Table 62. Server Configuration

8.5. Property Files Changes

Below is a listing of the property file modifications made for each environment. Other properties not listed here were not modified.

Service	File	Environment	Changes
Repo	alfresco-global.properties	Small	db.pool.max = 500 solr.http.connection.timeout = 300000 solr.http.socket.timeout = 300000 system.content.caching.cacheOnInbound = false system.content.caching.maxUsageMB = 0
		Medium	db.pool.max = 500 solr.http.connection.timeout = 300000 solr.http.socket.timeout = 300000 system.content.caching.cacheOnInbound = false system.content.caching.maxUsageMB = 0
		Large	db.pool.max = 500 solr.http.connection.timeout = 300000 solr.http.socket.timeout = 300000 system.content.caching.cacheOnInbound = false system.content.caching.maxUsageMB = 0
		Xlarge	db.pool.max = 500 solr.http.connection.timeout = 300000 solr.http.socket.timeout = 300000 system.content.caching.cacheOnInbound = false system.content.caching.maxUsageMB = 0
	server.xml	Small	ajp.number_of_threads = 500 http.number_of_threads = 500 http11.number_of_threads = 500
		Medium	ajp.number_of_threads = 500 http.number_of_threads = 500 http11.number_of_threads = 500
		Large	ajp.number_of_threads = 500 http.number_of_threads = 500 http11.number_of_threads = 500
		Xlarge	ajp.number_of_threads = 500 http.number_of_threads = 500 http11.number_of_threads = 500
Search	alfresco-global.properties	Small	db.pool.max = 275 transform.service.enabled = true

Service	File	Environment	Changes
		Medium	db.pool.max = 275 transform.service.enabled = true
		Large	db.pool.max = 275 transform.service.enabled = true
		Xlarge	db.pool.max = 275 transform.service.enabled = true
	server.xml	Small	ajp.number_of_threads = 1000 http.number_of_threads = 1000 http11.number_of_threads = 1000
		Medium	ajp.number_of_threads = 1000 http.number_of_threads = 1000 http11.number_of_threads = 1000
		Large	ajp.number_of_threads = 1000 http.number_of_threads = 1000 http11.number_of_threads = 1000
		Xlarge	ajp.number_of_threads = 1000 http.number_of_threads = 1000 http11.number of threads = 1000
	solrcore.properties	Small	shard.count = 2 shard.method = DB_ID shard.size = 60000000
		Medium	shard.count = 2 shard.method = DB_ID shard.size = 60000000
		Large	shard.count = 2 shard.method = DB_ID shard.size = 60000000
		Xlarge	shard.count = 2 shard.method = DB_ID shard.size = 60000000

Table 63. Changes made to the Property Files

9. Appendix 2 - Test Content Profile

This appendix provides details on the files and metadata used for the performance testing.

9.1. File Structure

At the start of the tests the existing files in the repository are spread out across sites. Each site contains between 5,000 and 100,000 files in its document library. A folder structure is created within each document library. This folder structure is no more than three levels deep, and no folder contains more than 100 files.

9.2. Content Files

The repository used for the tests described in this guide was populated with number of nodes noted (80 million, 500 million or 1 billion) files.

The mix of file mime types was as follows:

- Microsoft Word documents (25%)
- JPEG images (25%)
- PDF documents (25%)
- Microsoft PowerPoint documents (25%)

For each kind of file, the file sizes were evenly distributed across these ranges:

- 0-500KB
- 500KB-2MB
- 2MB-4MB
- 4MB-6MB
- 6MB-8MB
- 8MB-10MB

For the text-based documents, English words were used to generate them. Google's analysis of the 10,000 most-used English words was used to select the words to generate the documents, and the distribution of words selected followed this distribution:

- 35% between word 0 and word 499
- 25% between word 500 and word 999
- 20% between word 1000 and word 2999
- 10% between word 3000 and word 5999
- 7% between word 6000 and word 7999
- 3% between word 8000 and word 9999

Based on this dictionary, the following terms were defined:

- Word is any single word from this dictionary.
- A Phrase is a sequence of between three and eight Words, each separated by a space character.
- A Sentence is any sequence of between six and fifteen Words, each separated by a space character. Every Sentence is terminated by a full stop character followed by a space character.
- A Paragraph is any sequence of between three and seven Sentences. Every Paragraph is terminated by two new line characters.

Based on this set of terms, the rules used to guide the generation of the files were:

- DOCX files: Between 10 and 60 Paragraphs and between 0 and 4 embedded images
- PDF files: Between 10 and 60 Paragraphs and between 0 and 4 embedded images
- PPTX files: Between 8 and 32 slides, each slide having between 4 and 8 Sentences and between 0 and 3 images

In both the term definitions and document-generation rules, where a range is specified a normal distribution across that range is assumed.

9.3. Metadata

For each content file, the associated metadata properties are taken from one of three libraries: Simple, Moderate, and Complex. The property definitions used for the "Moderate" metadata are a superset of those used for the "Simple" metadata, and those used for the "Complex" metadata are a superset of the "Moderate" metadata.

The general structure of these metadata properties is:

Library Type	Total properties	Text properties	Time properties
Simple	8	5	3
Moderate	13	9	4
Complex	15	10	5

Table 64. Metadata Property Structure

A total of 400,000 instances of the Complex metadata set were generated and stored in a library. From this library, instances of Simple and Moderate metadata could be derived. When selecting the metadata set to pair with each file, this distribution was used:

- 29% Simple
- 36 Moderate
- 35% Complex

The details of the metadata models, remembering that Moderate builds on Simple and Complex builds on Moderate, are as follows:

Model	Property	Type	Indexing Attributes	Value Rules
Simple	name	text	Atomic; Not stored; Not facetable; Tokenised & Untokenised	Regex: (.*[\"*\\\>\<\?\/\:|]+.*)|(.*[\.]?.*[\.]+$)\|(.*[ ]+$) Between one and four Words taken from the 10000-word Google dictionary, each separated by a space. Terminated with an appropriate file extension for the file type.
	created	datetime	Atomic; Not stored; Facetable; Tokenised & Untokenised	1 year ago – 9 months ago: 25% 8 months ago – 6 months ago: 25% 5 months ago – 3 months ago: 25% 3 months ago – 0 months ago: 25%
	creator	text	Atomic; Not stored; Facetable; Tokenised	One of 200 usernames
	modified	datetime	Atomic; Not stored; Facetable; Tokenised & Untokenised	Randomly chosen time between "created" and now
	modifier	text	Atomic; Not stored; Facetable; Tokenised	Randomly generated name with format "<first_name> <last_name>"

Model	Property	Type	Indexing Attributes	Value Rules
	accessed	datetime	Atomic; Not stored; Not facetable; Tokenised & Untokenised	Same values as "modified"
	title	text	Atomic; Not stored; Not facetable; Tokenised & Untokenised	Between 4 and 9 Words, each separated by a space
	description	mltext	Atomic; Not stored; Not facetable; Tokenised & Untokenised	Between one and two Sentences
Moderate	publisher	text	Atomic; Not stored; Facetable; Tokenised	From list: ["Hachette Livre","Random House","Penguin Books","HarperCollins","Pan Macmillan","Pearson Education",Bloomsbury","Oxford University Press","Simon & Schuster","John Wiley & Sons","Egmont","Elsevier"]
	contributor	text	Atomic; Not stored; Facetable; Tokenised	Randomly generated name with format “<first_name> <last_name>”
	type	text	Atomic; Not stored; Facetable; Tokenised	From list: ["Action and Adventure","Anthology","Classic","Comic and Graphic Novel","Crime and Detective","Drama","Fable","Fairy Tale","Fan-Fiction","Fantasy","Historical Fiction","Horror","Humor","Legend","Magical Realism","Mystery","Mythology","Realistic Fiction","Romance","Satire","Science Fiction","Short Story","Suspense/Thriller","Biography/Autobiography","Essay","Memoir","Narrative Nonfiction","Periodicals","Reference Books","Self-help Book","Speech","Textbook","Poetry"]
	from	datetime	Atomic; Not stored; Facetable; Tokenised & Untokenised	Date/time between 1st Jan 1910 and 31st Dec 2019
	hits	int	Not Indexed	Random integer between 1 and 2,000,000,000
Complex	to	datetime	Atomic; Not stored; Facetable; Tokenised & Untokenised	Date/time later than “from” value and between 31st Dec 1970 and 31st Dec 2110
	identifier	text	Not Indexed	UUID

Table 65. Details of Simple and Complex Metadata Models

10. Appendix 3 – List of Tables and Figures