SCALABLE AND SUSTAINABLE LONG TERM DIGITAL PRESERVATION OF SCIENTIFIC DATASETS

Matthew Addis

Arkivum
UK
matthew.addis@arkivum.com
https://orcid.org/0000-0002-3837-2526

Arkivum contributors to the work presented in this paper include: Callum Barnes, Simon Bostock, Suja Chandra, Sharanya Dhanaraaj, Rob Fowler, David Kirkhope, Alex Nell, Jack Silk, Tom Vass and Dave Ward.

Abstract: The European Commission supported ARCHIVER project (Archiving and Preservation for Research Environments) aims to "introduce significant improvements in the area of archiving and digital preservation services, supporting the IT requirements of European scientists and providing end-to-end archival and preservation services, cost-effective for data generated in the petabyte range with high, sustained ingest rates, in the context of scientific research projects". This paper presents a software solution developed by Arkivum to meet the needs of long-term digital preservation of scientific datasets in ARCHIVER. We present and discuss how this solution is scalable (able to process and store very large volumes of research data) and sustainable (both economically and environmentally). This is achieved through a combination of serverless computing, deployment on hyperscale infrastructure, and implementation of configurable 'Minimum Effort Ingest' workflows. In particular, we show how high-performance and scalable Long Term Digital Preservation (LTDP) of very-large datasets can be done in a way that is entirely compatible with high levels of cost-efficiency and minimized environmental impact.

Keywords: Scalability, Sustainability, Environment, Cost, Research Data

Conference Topics: Building the Capacity & Capability; Scanning the New Development.

I. BACKGROUND AND CHALLENGES

The ARCHIVER digital preservation project [1], is part of the European Open Science Cloud initiative (EOSC). EOSC aims "to federate existing research data infrastructures in Europe and realise a web of FAIR data and related services for science, making research data interoperable and machine actionable following the FAIR guiding principles". FAIR means that data should be Findable, Accessible, Interoperable and Reusable. EOSC targets 1.7 million European researchers and 70 million professionals in science, technology, the humanities and social sciences. There is significant value in making research data open and accessible [3] according to FAIR principles [4]. Benefits include science that is higher quality and more productive, faster development of new products and services, and increased impact for research when addressing societal challenges. These benefits only fully accrue if FAIR data remains accessible and usable for the long-term. As described in the TRUST article in Nature [5], "to make data FAIR whilst preserving them over time requires trustworthy digital repositories (TDRs) with sustainable governance and organizational frameworks, reliable infrastructure, and comprehensive policies supporting community-agreed practices". As identified in the ARCHIVER D2.1 report on the state of the art into LTDP for large scale scientific data, there is much work still to be done to achieve this vision, including the need for better LTDP infrastructures and software [6]. This is backed up by the Digital Preservation Coalition (DPC) report 'FAIR Forever' [7] which makes a series of

urgent recommendations for digital preservation within EOSC.

ARCHIVER is currently the only EOSC project that is addressing the need for new software and services for LTDP of large scientific datasets. ARCHIVER has contracted several organisations, including Arkivum, to develop new digital preservation solutions. The aim is to provide new services in EOSC that will help research organisations to deliver long-term FAIR data through TDRs. ARCHIVER is driving this through a set of use cases [8] from scientific organisations including CERN, EMBL-EBI, PIC and DESY. All of the use cases involve very large datasets (Petabyte scale), decade or longer retention timescales, and fast ingest and access (at the time of writing the ARCHIVER project is in its Prototype phase with a target ingest rate of 100TB per day per organization). These datasets need to be archived cost-effectively with long-term preservation and access that is economically sustainable. As described in the ARCHIVER D2.1 report, this is not achievable using current digital preservation products and services, which is why ARCHIVER is funding new Research and Development in this area.

In addition to the need for economic sustainability of LTDP, the need for environmental sustainability should not be overlooked. This topic is covered in detail in the Pendergrass report [9] which offers both stopgap measures for reducing digital preservation's negative environmental impacts and proposed shifts in digital preservation practice in the areas of appraisal, permanence, and availability. This is important given the scale of the ICT infrastructure needed for PB scale LTDP and decade-level storage and access to huge datasets.

Whilst the core use cases for ARCHIVER center on very large-scale research datasets, EOSC has a much broader remit and also aims to support the Long Tail of Science (LToS). LToS refers to the large number of individual researchers and small laboratories who do not have access to dedicated computational resources and online services to manage and analyse large amount of data [10]. In science terms, the long tail is made up of scientific/research projects handled by individual laboratories or small groups of researchers. Only an estimated 16% of users of cloud services for research have a need for huge usage [11]. Therefore, the LTDP services provided as part of EOSC need to support usage scenarios that range from the very small scale (long-tail) through to the very large scale (ARCHIVER core use cases).

In summary, there are a series of significant challenges that need to be addressed in order to support LTDP of FAIR data in EOSC. ARCHIVER is addressing these challenges through the R&D of a new set of LTDP solutions and services. This includes the need for:

- Scalable and high performance LTDP systems that can process, store and provide access to multi-petabyte datasets and support ingest rates of 100TB+ per day.
- Cost effective LTDP that helps minimizes the Total Cost of Ownership (TCO) of FAIR data over decade level timescales and longer.
- Environmentally sustainable LTDP that minimizes energy consumption and use of ICT infrastructure.
- LTDP services in EOSC that can be made available to all EOSC users from the long-tail of science through to very large-scale data-intensive institutions and research infrastructures.

II. APPROACH

LTDP systems will often perform a series of data processing steps. Typical processing activities include checksum generation and validation, file format identification, virus scanning, file format conformance checks, metadata extraction and format characterisation, content extraction, file format conversions, e.g. as part of migrations or normalisation, compression and decompression, encryption and decryption, package generation, replication of files/packages to multiple storage locations, and initial and ongoing fixity checks. These correspond to preservation activities and events such as those defined in the PREMIS event vocabulary [12] and described by PAR [13][34]. In some cases, further specialist processing may need to be done such as forensic analysis, redaction, and using legacy software applications for rendering and using content e.g. as part of emulation based approaches to digital preservation. LTDP systems typically execute these steps using a range of tools and according to preservation policies and workflows. To do this at scale for large datasets can require significant computing resource. The conventional approach to processing large volumes of data in LTDP systems is to use large servers with significant memory, CPU cores and fast disks (scale-up). Multiple servers allows large-scale parallel processing, e.g. by using a scale-out compute cluster.

However, this server-based model can be inefficient and hard to scale. Provisioning sufficient resources to meet peak load (e.g. 100TB per day ingests) risks that the system is significantly underutilized at other times – potentially with periods where the whole system is idle. This wastes money and energy. It also requires more hardware than actually necessary so is not economically or environmentally sustainable.

Moving servers to the cloud can partially address this problem by making use of the elastic nature of Infrastructure as a Service (IaaS) offerings from hyperscale cloud providers such as Amazon (AWS) and Google (GCP). However, it is a common misconception that virtual machines in the cloud are provided on a 'pay per use' basis. VMs in the cloud will typically incur costs irrespective of whether they are being used or not. Furthermore, scaling up or scaling down a set of VM servers used for LTDP in the cloud (or anywhere else for that matter) will typically involve manual processes for adding/removing servers in order to match the total computing capacity to the data load that needs to be handled at any given time. Whilst cloud platforms do provide 'autoscaling' facilities that can dynamically increase/reduce the number of servers in response to load, this is not something that LTDP systems have traditionally been architected and designed to take advantage of. This leads to further inefficiencies and increased costs.

Serverless computing [14] is a relatively recent cloud computing innovation that removes the need for an application to run on one or more servers (virtual or physical). Instead, an application consists of services (code) to which compute resources are allocated on-demand. If no processing is required at a given point, no services are executed, and no compute infrastructure is either consumed or paid for. Furthermore, splitting an application (LTDP system in this case), into a set of small services that are stateless and run in a serverless environment also affords a great deal of scalability. If there is a large dataset to process that consists of many files and where there a large number of processing steps to be done for each file, then this can be done massively in parallel in a serverless environment. Hundreds or even thousands of files can be processed concurrently. This approach has already been used to good effect for analysing very large-scale scientific datasets in the cloud [15]. Our approach in the ARCHIVER project uses serverless computing for LTDP. The system we have developed consists almost entirely of small services that run in a serverless environment (using Kubernetes and knative in our case). This allows the LTDP solution to scale-up to meet big workloads, but just as importantly, to also scale-down (scale-zero) so that resources are not consumed if the system is idle or under minimal load.

LTDP requires data to be processed in order to be preserved and accessed, but it also needs storage. Data needs to be physically stored, typically in multiple locations and with ongoing integrity management and hardware/software migrations to guard against technical obsolescence. Good practice in this area is described by the NDSA levels of preservation [16], DPC RAM [17] and the Digital Preservation Storage Criteria [18]. The conventional approach for archiving and storing large-scale scientific datasets is to use Hierarchical Storage Management (HSM) systems that combine storage technologies such as disk servers and data tape robots. There is a move towards data archiving in the cloud, e.g. using cloud storage services such as AWS Glacier and Google archive storage, but this is not yet mainstream for long-term storage of scientific datasets. Large-scale on-premise archival storage systems can be difficult to scale, difficult to provision so that they are not underutilized yet also have capacity to cope with big ingests, and difficult to build in a way that meets a wide range of data usage scenarios that go from very high frequency access to datasets over the Internet through to low-cost deep-archiving of very infrequently accessed raw datasets.

Cloud infrastructure providers have evolved to provide multiple tiers/classes of cloud storage where each class will typically have a different profile for cost, frequency of access, retention period and data safety. Google cloud storage is an example with four tiers: standard, nearline, coldline and archive [19]. Individual files (objects) can be quickly and easily moved or replicated across tiers, either through APIs or through policies that form part of automated Object Lifecycle Management (OLM). The ability to optimize the storage of data at a fine level of granularity allows for cost optimization. For example, original raw data files in a scientific research dataset might be stored using low-cost deep-archive tiers. Derived data on the other hand might be held on fast access tiers for easy download and reuse by a scientific community. Cloud storage has a pricing model that is primarily based on the volume of data that is being stored (often with discounts for large

volumes) and how often it is being accessed. This is a pay-per-use model and means that storage costs are not incurred when data is not being stored or not being accessed. This helps minimize the TCO of storage across a wide range of use cases from frequent public access through to deep archiving for Disaster Recovery (DR).

The combination of cloud serverless computing and multi-tiered cloud storage goes a long way to addressing both the scalability/performance and cost-efficiency requirements of LTDP in EOSC. The need to support LTDP from the long-tail through to the large-scale is accommodated by using multi-tenanting techniques, both at the application level and also at the cloud infrastructure level. We do this in our solution so that small organisations (long-tail) can use and take advantage of the same LTDP services and cost-efficiencies that are available for large organisations who have very big datasets.

Further efficiencies and cost reduction can be achieved by adopting a Minimal Effort Ingest (MEI) [24] approach, which in our case we term Minimal Viable Preservation (MVP) [25]. This reduces the application of unnecessary processing in the early stages of preservation. Instead, the focus is on doing the bare minimum in the context of the specific data that is being preserved and how it may need to be used in the future. In our case, this will typically involve an ingest process that does basic fixity checks, identifies and records file formats, extracts basic metadata, organizes files into searchable datasets, and then puts these files into archival storage as fast as possible. More advanced operations, for example file format normalisations or detailed content analysis, can be deferred to a later stage. We allow the minimal set of steps to be defined using metadata and at a very granular level. This includes storage requirements as well as processing requirements. For example, this allows a user to specify that files X,Y,Z in their ingest are raw data files that should be check-summed, replicated and deep-archived but otherwise need no further processing. However, files A,B,C in the same ingest might be frequently accessed so need to go on a fast access storage tier, they should be fully indexed so will need to undergo metadata extraction, they need file format conformance checks so need to undergo format identification and validation, and they need to be virus scanned because they will be put online for use by others. This minimizes processing so that specific operations are only applied by the system to the files that it makes sense to apply them to. Furthermore, it allows the same core LTDP solution to be used in a wide range of scenarios, e.g. for long-tail science, because users are able to define what happens to their data in order to meet their needs without this being 'hard coded' into preset workflows within the system.

One of the challenges enumerated in the first section of this paper was the need for environmental sustainability. As described in the Pendergrass report and further discussed by this author [20], the ICT aspects of environmental impact can be broken down into the impact of the 'production' of ICT (the embodied footprint) and the 'use' of that ICT (e.g. energy consumption and cooling). The environmental costs should not be underestimated of extracting and processing the raw materials needed to make up computer hardware along with its eventual disposal and, ideally, recycling – which along with manufacturing and transport constitutes the embodied footprint. LTDP requires ICT hardware for processing, storing and providing access to data and this applies both to the cloud and on-premise deployments. In recent years, cloud hyperscalers have made significant advances and commitments towards the environmental sustainability of their facilities and services – for example as described in the Greenpeace click-clean report [21]. The move towards renewable energy sources, natural cooling, and higher-efficiency infrastructure, including ICT, all means that the 'use' dimension of environmental impact is not as 'dirty' or 'spiraling out of control' [22] as some would have us believe. For example, Google, which provides the IaaS that Arkivum uses in ARCHIVER, was carbon neutral in 2007, purchases 100% of its energy from renewable sources, and is committed to be carbon free by 2030 [23]. This still leaves the 'production' phase of ICT and the position of cloud IaaS providers is less clear in this respect. However, given their position and status, they are all increasingly incentivized to both encourage and adopt environmentally sustainable ICT equipment in their facilities. Their scale and leverage means this is far easier for them to achieve at their scale of operations than it is for organisations who instead procure their own ICT equipment and run it on-premise or in private data centres.

The combination of serverless computing, optimized placement of data onto appropriate storage classes, and hyperscale infrastructure all go hand-in-hand to reducing environmental impact.

For example, the adoption of serverless architectures at the application level (our LTDP software system) affords the IaaS provider (Google in our case) more flexibility for resourcing the application and minimizing the physical ICT systems they need to have in place. Serverless computing allows the IaaS provider to optimize their hardware to achieve higher efficiencies and utilization levels. This not only lowers energy consumption (use phase) but also reduces the amount of hardware needed (production phase). The result is a lower total cost both economically and environmentally. Achieving these efficiencies is much harder when using dedicated servers or in-house infrastructures. Whilst not a panacea, and with clearly more work to be done by cloud IaaS providers on environmnental sustainability, the use of hyperscale platforms such as Google to underpin LTDP does offer many positive environmental benefits compared with alternative approaches.

In summary, our approach is to:

- Use hyperscale cloud IaaS (Google in our case) in order to resource the LTDP system so that it can achieve very high ingest and storage rates (up to 100TB per day per organization)
- Use serverless computing to achieve very high levels of parallelism and performance whilst at the same time ensuring resources are only consumed and paid for when actually needed.
- Use metadata attributes on datasets to provide users with fine-grained control over the specific processing and storage applied to their content. This helps users adopt MEI/MVP and minimize costs.
- Use multi-tenanting and a single core platform to deliver a LTDP service for all sizes of organization from the long-tail of science through to very big organisations such as CERN and EMBL-EBI.
- Use a combination of all of the above to minimize the need for ICT resources (in both the production and use phases) and hence minimize the environmental impact of LTDP.

III. IMPLEMENTATION AND RESULTS

In order to take advantage of serverless computing, the Arkivum solution is architected as a set of microservices. The services that form the solution are shown in the diagram below (Figure 1). This includes services provided by Google as part of GCP, services provided by the Arkivum software, and services provided by external applications such as keycloak for authentication. In addition, (not shown) there are further services that provide supporting infrastructure for messaging (Kafka), metadata storage (MongoDB) and indexing/search (ElasticSearch).

The services are grouped in the diagram according to the requirements of the ARCHIVER end-users [26]. L1 (level 1) is large scale archival storage, L2 (level 2) is Long Term Digital Preservation, L3 (level 3) is added value services for example dataset management, and L4 (level 4) is services for analysing archived data, for example being able to run scientific analysis codes.

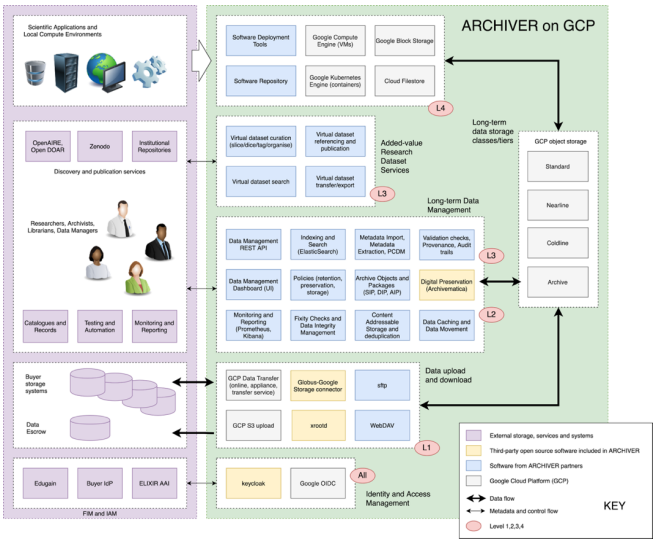

Figure 1 High-level services based architecture of the ARCHIVER solution.

The services are deployed as containers that run in Kubernetes (k8s) [32] clusters. The initial deployment of the infrastructure is automated using Ansible and Terraform. Knative [27] is used to manage the k8s clusters and, in particular, to autoscale the number of pods and nodes in response to workload. This includes both scaling up and scaling down. Rancher [31], Prometheus [29] and Grafana [28] are used for monitoring the k8s clusters and have proven useful in optimising the workflow of the system and identifying bottlenecks.

Several datasets have been used to test the scalability, performance and efficiency of the solution. These include scientific datasets from ARCHIVER end-users and datasets from the heritage domain, for example images from the British Library's Digitised Books collection [28].

A. *Large scale ingest (50TB)*

50TB of data consisting of 100,000 files was ingested into the system. Files represent scientific experiments done at the DESY synchrotron in Germany (DESY is one of the ARCHIVER end-users). The ingest and storage process completed in 8 hours. This corresponds to an ingest rate of 150TB/24hrs. The ingest process included: checksum generation (four algorithms were computed: Adler32, MD5, Sha256 and Sha512); file format identification (Apache Tika with most of the files in the dataset being identified as application/x-hdf, which is HDF5 format [33]); caching into internal storage; chopping up of large files into 100MB size pieces that could be processed and stored in parallel; and then replicating the content into two separate long-term GCP storage locations (buckets), which included MD5 checks to confirm storage was successful. Data was first uploaded to a GCP bucket (upload bucket) and was ingested into the system from there. The end of the process resulted in data being replicate so that there are two complete and separate copies of the data held in long-term storage buckets (the exact class of bucket depends on user requirements – see the discussion on costs below).

Figure 2 shows the number of CPUs used by the system during the ingest. The number climbs sharply when the ingest starts (just before 9:00am), stays relatively high (240 – 290 vCPUs during the ingest), and then falls back down when the ingest completes (just before 5:00pm). This shows (a) how compute resources have been dynamically allocated in response to the ingest workload and (b) how resources are only consumed when there is work to be done. The much lower background level before and after the ingest corresponds to CPUs that are permanently allocated to persistent services (e.g. Mongo, ElasticSearch, UI, system scheduler).

Figure 2. CPU count (the total number of virtual CPUs in use across all k8s clusters)

The dynamic provisioning of nodes in the cluster can be seen in Figure 3. This shows a node being used for part of the ingest (blue line) and then another node taking its place later in the ingest to do further work (green line). This is the result of using pre-emptible nodes as a cost saving strategy (see further discussion later in this paper). During the ingest, the average level of CPU utilisation across all nodes in the k8s clusters was just over 50%. This includes the infrastructure nodes running Mongo, ElasticSearch which were only lightly loaded. For the nodes running microservices that are directly involved in data processing, e.g. checksum generation or file format identification, the CPU utilisation reached near 100%.

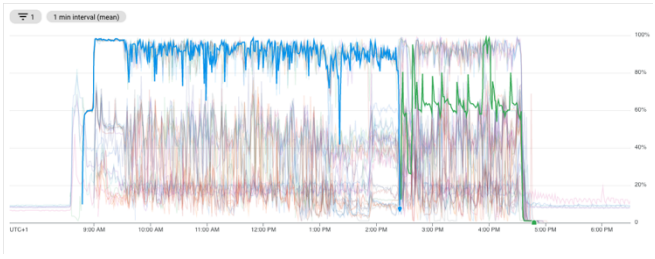

Figure 3. CPU utilisation for individual compute nodes. Two nodes are highlighted (blue, green) showing how different nodes are spun-up/spun-down during the ingest process.

During the ingest, data was read from storage, which includes reading from the GCP upload bucket containing the data to be ingested plus reading data from the internal cache. Read operations achieved a sustained rate of approx. 4GB/sec (32Gbit/sec) across the ingest and cache buckets. This is shown in Figure 4. Likewise, data was written to storage during the ingest. This includes putting data into the internal cache and then creating two copies in the long-term storage locations used for archival storage. The write operations achieved a sustained rate of approximately 2GB/sec (16Gbit/sec) as shown in Figure 5.

The high and sustained data rates show the capability of object storage to support I/O intensive digital preservation processes such as replication and fixity checks. The overlapping reads and writes that happen throughout the duration of the ingest is a result of the high level of parallelism in the system. For example, one file might be being replicated to long-term storage (write) whilst another file is being check-summed (read) and another file is being copied into the system cache from the source bucket (read and write).

The read bandwidth is higher than the write bandwidth because files are read multiple times, for example when computing checksums and performing file format identification.

Figure 4. Read data rate (GiBytes/sec)

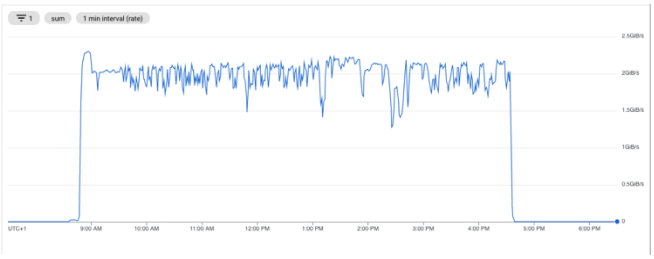

Figure 5. Write data rate (GiByte/sec)

During the ingest, nodes (compute resources) and pods (groups of containers that run the microservices) were added on demand by knative. A snapshot of this process can be seen below. This shows the Rancher Dashboard that is used to monitor the system and track the number of pods and cores being used. In this example, some events are shown that correspond to the creation of a new container that provides integrity services (checksum generation in this case).

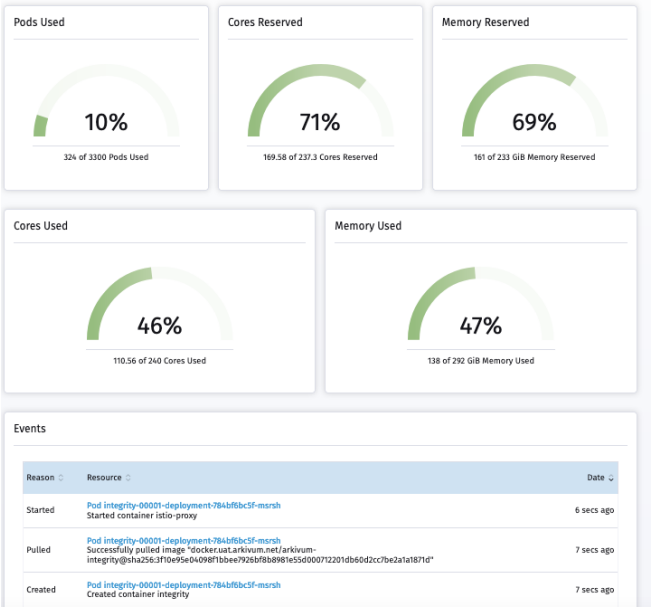

Figure 6. Rancher dashboard showing a snapshot of pod and node usage along with container provisioning events.

B. Large scale ingest (216,000 files)

Images from the British Library's Digitised Books collection [28] were ingested to investigate how the system responded to large numbers of files. The ingest consisted of approximately 216,000 files in jpeg format with a total data volume of 88GB.

The ingest process was the same as described above for the 50TB ingest of scientific data. However, in addition, metadata extraction was also done for each of the files (using Apache Tika). This added an extra microservice to the workflow. The results of metadata extraction were stored in the system's Mongo database, indexed using ElasticSearch so that the metadata is searchable, and also serialised and replicated to the long-term storage buckets along with the jpeg files.

The number of virtual CPUs used to process the ingest is shown in Figure 7. Ingest started at 12:45am and completed by 6:45pm. The number of CPUs in use is higher in the early part of the ingest because of the processing intensive steps done at the beginning of the ingest process, which include file format identification, metadata extraction and checksum generation. The number of CPUs allocated to the k8s clusters then drops and in this stage of the ingest the system is completing the replication and fixity checking part of the process.

Figure 7. CPU count (the total number of virtual CPUs in use across all k8s clusters)

The different phases of the ingest process can also be seen in the data read rate (Figure 8). Data read operations are heavily loaded towards the first part of the ingest because each file is read multiple times at this stage. Read rate then drops and corresponds to files being read as part of the copying process where they are replicated to long-term storage buckets.

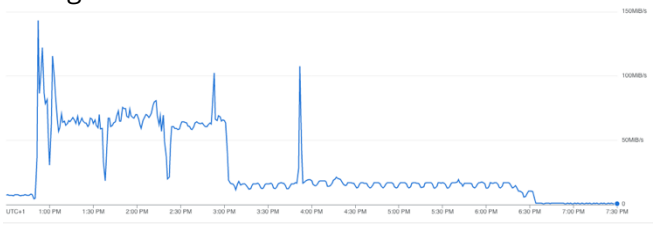

Figure 8 Figure 9. Read data rate during ingest (MiBytes/sec)

The ingest completed in just under 6 hours, including replication and checking of the files in the

long-term storage locations. This corresponds to a file processing rate of approximately 860,000 files per 24hrs.

From a user point of view, progress of the ingest can be tracked through the system's Dashboard and reports are also available through a REST API. The user can see what the system is currently doing, including the throughput for the last seven days, and how many files have been ingested and replicated to each long-term storage location (Figure 10). Note that in the example shown, more files had been subsequently ingested into the system in addition to the British Library dataset (1.84M files in total).

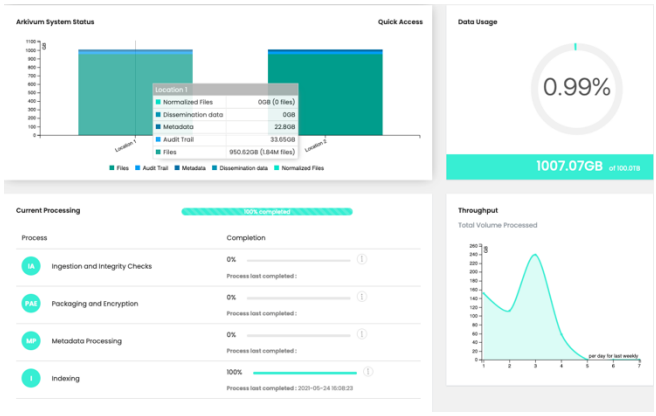

Figure 10. System Dashboard showing overall status of the system.

Reporting is available for the individual steps in the ingest process for each file. For example as shown in Figure 11 for one of the files in the British Library dataset. The processing steps are on the right with details shown for the metadata extraction microservice.

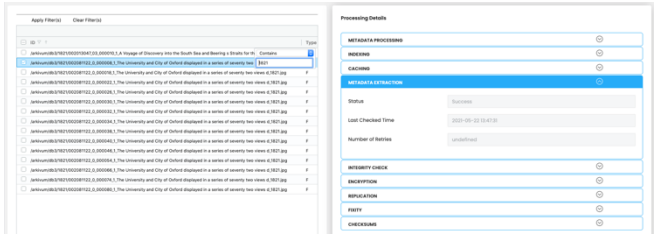

Figure 11. Steps of the ingest process for a specific file.

The user can see the results of each step, for example the metadata that has been extracted from the image. An example is shown in Figure 12 for one of the jpeg images in the British Library dataset.

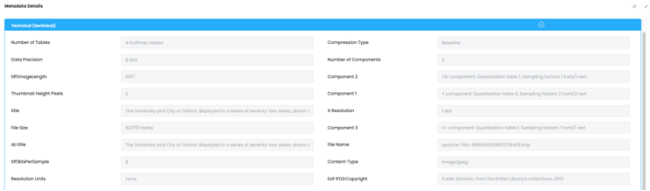

Figure 12. Subset of the metadata fields extracted from one of the images in the dataset.

C. *Cost optimization*

As described above, the system is able to scale-up and scale-down the resources it consumes in response to workload. The use of k8s and the serverless model allows for costs only to be incurred when resources are being consumed. This contrasts with more conventional server-based computing approaches where Virtual Machines (VMs) used to run server applications. These VMs will incur costs even if those applications are not being used. It is a common misconception that cloud compute servers (VMs) are 'pay per use'. This is typically not the case and the charging model used by cloud providers involves customers paying for capacity (e.g. based on the number of cores and memory for a VM and the duration it is provision for) irrespective of whether the VM is used or not. For example, the same per-minute cost will apply for a VM irrespective of whether it is utilized at 1% or 100%. In the case of Google Kubernetes Environment (GKE), costs are only incurred for actual resources used because nodes are only added and used in the cluster when there is work to be done. If there is no work to be done on a node then it is automatically removed and costs are no longer incurred. Furthermore, our system uses pre-emptible nodes in the cluster when processing data. In GCP, pre-emptible nodes have a lifetime of less than 24hrs and GCP can reclaim them for other purposes at any time (e.g. for other customers who pay more for guaranteed availability and performance). However, pre-emptible nodes are available at up to 70% lower cost for each CPU-hour consumed. This is a large cost reduction. In many archiving scenarios, processing of data is not time-critical and it doesn't matter if available resources fluctuate. Therefore, we designed our system to be robust so that ingest workflows will continue even if nodes in the k8s clusters are pre-empted. This can be seen in Figure 3 where 7 nodes were reclaimed by GCP at one point in the workflow (one such pre-emption is shown in the diagram). These nodes were automatically replaced with others that were available and processing continued with very little disruption and no need for any manual intervention.

Pre-emptible nodes are not used for the persistent part of the system such as the infrastructure services that run the Mongo database, the ElasticSearch index, and the UI service. These

components incur costs irrespective of the system load. However, these components are also multi-tenanted so that costs can be distributed across multiple organisations and efficiencies of scale are achieved. The infrastructure services are deployed within k8s so that they can be scaled up or down if needed.

In our tests, the net result of (a) serverless computing, (b) pay-per-use billing, and (c) cost-reduction through pre-emptible nodes was very cost-effective ingests. The 50TB ingest incurred compute costs of approx. $100, which represents a per TB cost of approx. $2. The 216k file ingest incurred compute costs of approx. $40, which represents a cost of approx. $2 per 10,000 files. Mixed workloads would incur a mix of these costs. There are of course additional costs for long-term storage of the data. There will also be costs for accessing and retrieving the data. These additional costs depend on the type of storage being used and the frequency of access. For example, GCP provides four tiers of object storage (standard, nearline, coldline and archive). Each provides immediate access to data, but the cost per TB falls as the frequency of access decreases. At the time of writing, if data is accessed less than once a year, i.e. deep archiving, then typical archive tier storage list prices are $0.0025 per GB per month, which equates to $30 per TB per year. Costs vary depending on which Google storage location is used (e.g. US, Europe, Asia) and whether the data is stored in multiple regions. For example, archive storage in GCP Finland as a single location is currently $14 per TB per year whereas multi-region storage using multiple geographic locations in Europe is $48 per TB per year. If data is accessed frequently, then other GCP storage tiers would become more appropriate and these have a higher per-TB costs. Depending on the network used between the end-user and GCP, there can be additional egress charges too, for example when accessing data over the Internet.

The Arkivum system design allows users to specify what classes of storage they would prefer to use for their data (e.g. hot, medium, cold) based on their expected access profiles along with the level of data safety that is needed. This is done through metadata attributes that users can apply to some or all of their ingests. The attributes can be applied to single files, subsets of a dataset, or to an ingest as a whole. This gives users very fine-grained control over how their content is stored. For example, a user might set attributes on the raw scientific data files in a dataset to indicate that these files must be stored with a high level of safety but can otherwise be deep-archived because they will be accessed infrequently (cold). Whereas other files in the same ingest might have attributes set to indicate that these files are derived data, which requires a lower level of data safety, but these files will likely be accessed frequently by end users (hot). This gives the system the flexibility and information to store copies of the user's data on appropriate tiers to achieve the required mix of accessibility, data safety and cost for each file or group of files. For example, this might be done by storing three copies of the data – one on GCP standard storage as the 'access copy' used to serve access requests, one on GCP archive storage as a low-cost replica in a different GCP region, and one using an entirely independent cloud provider (Arkivum typically uses Azure) as an additional copy for DR and exit-strategy purposes.

A detailed cost model that includes compute, storage and networking is not in the scope of this paper. However, further work in ARCHIVER includes Arkivum creating a detailed cost model for LTDP using our system. This is a contractual deliverable of the project. The results of this will be included in the presentation of this paper if accepted for the conference.

IV. SUMMARY AND CONCLUSIONS

In this paper we have presented an approach to scalable and cost-effective long-term data archiving and digital preservation in the cloud. Hyperscale cloud IaaS (GCP) is used to resource the LTDP system so that it can achieve very high ingest and storage rates (>100TB per day). Serverless computing is used to achieve very high levels of parallelism and performance whilst at the same time ensuring resources are only consumed and paid for when actually needed. Further cost optimizations include multi-tenanting so that a single core platform can deliver a LTDP service for all sizes of organization from the long-tail of science through to very big organisations such as CERN and EMBL-EBI. The combination of the above all help to minimize the need for ICT resources for LTDP (in both the production and use phases) and hence helps to minimize the environmental impact of processing and storing data. Tests show that the system scales well under various workloads, including large scientific datasets from ARCHIVER, but also for other

types of data such as image collections that are more common in memory institutions.

ACKNOWLEDGMENTS

This paper presents the work of the Arkivum Team who have collectively designed, developed, deployed, tested and supported the Arkivum ARCHIVER software solution. The author would like to thank and acknowledge the contribution of all members of this team, including: Callum Barnes, Simon Bostock, Suja Chandra, Sharanya Dhanaraaj, Rob Fowler, David Kirkhope, Alex Nell, Jack Silk, Tom Vass and Dave Ward.

The Arkivum ARCHIVER solution has been deployed on Google Cloud Platform (GCP). The author would like to acknowledge Google's significant support for the ARCHIVER project.

The work presented in this paper is being done as part of the ARCHIVER (Archiving and Preservation for Research Environments) project that has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 824516.

REFERENCES

[1] ARCHIVER project. https://archiver-project.eu/

[2] European Open Science Cloud (EOSC). https://eosc-portal.eu/about/eosc

[3] J. Tennison. "The Economic Impact of Open Data: What Do We Already Know?", *The Huffington Post.* Oct 2016 https://www.huffingtonpost.co.uk/jeni-tennison/economic-impact-of-open-data_b_8434234.html?guccounter=1

[4] Findable Accessible Interoperable Reusable (FAIR). https://www.go-fair.org/fair-principles/

[5] D Lin et al. "The TRUST Principles for digital repositories". *Scientific Data.* 7, Article 144. May 2020. https://www.nature.com/articles/s41597-020-0486-7

[6] J. Fernandes et al. "ARCHIVER D2.1- State of the Art, Community Requirements and OMC Results ". Jan 2020. https://zenodo.org/record/3618215

[7] A. Curry, W. Killbride. "FAIR Forever? Long Term Data Preservation Roles and Responsibilities, Final Report". Feb 2021. https://zenodo.org/record/4574234

[8] ARCHIVER deployment scenarios. https://www.archiver-project.eu/deployment-scenarios

[9] K.L.Pendergrass et al. "Toward Environmentally Sustainable Digital Preservation ". *The American Archivist* (2019) 82 (1): 165–206. https://dash.harvard.edu/handle/1/40741399

[10] "Serving the long tail". *Digital Infrastructures For Research*. Sept 2016. https://www.digitalinfrastructures.eu/content/serving-long-tail

[11] M. Devouassoux, B. Jones, J. Fernandes. "Long-Tail-of-Science's Requirements for Commodity Cloud Services in Europe" Oct 2019. https://zenodo.org/record/3564668

[12] "Preservation Events Controlled Vocabulary Revision 1" Aug 2017. https://www.loc.gov/standards/premis/v3/preservation-events-revision1.pdf

[13] Preservation Action Registries (PAR). https://parcore.org/

[14] Serverless Computing. https://en.wikipedia.org/wiki/Serverless_computing

[15] L. Heinrich, R. Rocha. "Reperforming a Nobel Prize discovery on Kubernetes". *24th International Conference on High Energy and Nuclear Physics.* Nov 2019. https://indico.cern.ch/event/773049/contributions/3581373/attachments/1939661/3215578/chephiggs.pdf

[16] NDSA Levels of digital Preservation. https://ndsa.org/publications/levels-of-digital-preservation/

[17] DPC Rapid Assessment Model (RAM). https://www.dpconline.org/digipres/dpc-ram

[18] S. Schaefer et al. "Digital Preservation Storage Criteria". Jan 2018. https://osf.io/sjc6u/

[19] Google Cloud Storage.https://cloud.google.com/storage

[20] M. Addis. "Is digital preservation bad for the environment? Reflections on environmentally sustainable digital preservation in the cloud". DPC blog. June 2020. https://www.dpconline.org/blog/is-digital-preservation-bad-for-the-environment

[21] G. Cook et al. "CLICKING CLEAN: WHO IS WINNING THE RACE TO BUILD A GREEN INTERNET?" 2017. http://www.clickclean.org/uk/en/

[22] https://datacenters.lbl.gov/sites/default/files/Masanet_et_al_Science_2020.full_.pdf

[23] E. Masanet et al. "Recalibrating global data center energy-use estimates". *Science Magazine*. Mar 2020. https://sustainability.google/commitments/

[24] B. A. Jurik et al. "Minimal Effort Ingest". iPRES 2015. Poster. https://digitalbevaring.dk/wp-content/uploads/2015/12/151208_Minimal-Effort-Ingest_iPRES_2015_poster.pdf

[25] M. Addis. "Minimum Viable Preservation". DPC blog. Nov 2018. https://www.dpconline.org/blog/minimum-viable-preservation

[26] J. Fernandes et al. "ARCHIVER D2.1- State of the Art, Community Requirements and OMC Results". Jan 2020. https://zenodo.org/record/3618215#.YK0c6-vTWL5

[27] Knative. https://knative.dev/

[28] "Digitised Books - Images identified as Medium Sized Images. c. 1567 - c. 1900." British Library. https://data.bl.uk/digbks/db18.html

[29] Prometheus. https://prometheus.io/

[30] Grafana. https://grafana.com/

[31] Rancher. https://rancher.com/

[32] Kubernetes. https://kubernetes.io/

[33] Hierarchical Data Format. https://en.wikipedia.org/wiki/Hierarchical_Data_Format

[34] M. Addis et al. "Digital Preservation Interoperability through Preservation Actions Registries". iPRES 2018. https://doi.org/10.6084/m9.figshare.6628418