FIRE – A Dual Archive Service Provided By EMBL-EBI Since 2008

Joan Marc Riera Duocastella
Technical Services Cluster
European Molecular Biology Laboratory - European Bioinformatics Institute (EMBL-EBI)
Hinxton Cambridge UK
marc.riera@ebi.ac.uk 0000-0002-0609-0137

Stuart Meacham
Technical Services Cluster
European Molecular Biology Laboratory - European Bioinformatics Institute (EMBL-EBI)
Hinxton Cambridge UK
smeacham@ebi.ac.uk 0000-0002-3667-6059

Fahri Cihan Demirci
Technical Services Cluster
European Molecular Biology Laboratory - European Bioinformatics Institute (EMBL-EBI)
Hinxton Cambridge UK
fcdemirci@ebi.ac.uk 0000-0002-6229-8709

Guilherme Formaggio De Mello
Technical Services Cluster
European Molecular Biology Laboratory - European Bioinformatics Institute (EMBL-EBI)
Hinxton Cambridge UK
gfmello@ebi.ac.uk 0000-0002-9829-091X

Steven Newhouse
Technical Services Cluster
European Molecular Biology Laboratory - European Bioinformatics Institute (EMBL-EBI)
Hinxton Cambridge UK
steven.newhouse@ebi.ac.uk 0000-0003-1531-5198

Sarah Butcher
Technical Services Cluster
European Molecular Biology Laboratory - European Bioinformatics Institute (EMBL-EBI)
Hinxton Cambridge UK
sarahb@ebi.ac.uk 0000-0002-4494-5124

Abstract— **One of the European Bioinformatics Institute's core objectives is to provide long-lasting and freely available life science data resources. In 2008, several of the fastest growing data resources had reached 80TB and to support their continued and accelerating growth, a software-defined File Replication system (FIRE) was created. The system capacity has grown from archiving 20TB a year in 2008 to 10PB a year in 2019, while also providing access to almost all that data, to a constantly growing research community, through permanent URLs. Whilst the initial system relied on NFS storage to replicate its data, it has transitioned in several stages to using tape systems and geo-dispersed object storage, to support the continuous data growth, which at current projections is expected to reach the Exabyte milestone before 2030. The challenges faced by such a system are introduced and the implementation changes during different phases of the project are shared, together with the lessons learned over the past 12 years while providing a continually evolving system to the present day.**

Keywords—archive, tape, API, software-defined storage

I. Introduction

EMBL's European Bioinformatics Institute (EMBL-EBI) is the world's leading sources of biological and biomolecular data. Its core mission is to enable life science research and its translation to medicine, agriculture, industry and society by providing biological data, tools and knowledge. EMBL-EBI is Part of the European Molecular Biology Laboratory (EMBL), an open science intergovernmental organisation [1]. Since the EMBL-EBI opened in 1994, data volumes have increased year on year and technology changes have driven a huge increase in biological data diversity. By 2018, EMBL-EBI was hosting over 40 different biomolecular databases, with data content spanning genomic and metagenomic data (DNA, RNA and protein) and structures as well as proteomics, metabolomics, bioactive molecules, macromolecular complexes, systems, pathways, ontologies, scientific literature and light and electron microscopy imaging data [2].

Data access and demand from the world's researchers also continues to increase significantly. In 2018 for example, the average number of daily requests to EMBL-EBI websites almost doubled from the previous year, to 64 million [3]. To underpin these requirements, EMBL-EBI manages an infrastructure of extensive, high-performance and high-throughput compute infrastructure along with large data-storage capacity [4]. The hardware is dispersed across three discrete data centres in different geographical locations to assure long-term security. The internal network has a 100 Gigabit backbone within its data centres and multiple redundant 100 Gigabit connections between data centres and to Janet, Internet2 and Geant (the UK, pan-American and pan-European research networks, respectively).

A. Long term archives with specific needs

The world's first public nucleotide sequence database: the EMBL Nucleotide Sequence Data Library, now EMBL Bank, part of the European Nucleotide Archive, was established in 1980 to be a central database of DNA sequences abstracted from scientific literature as a resource for research. Demand quickly grew and researchers began submitting data directly, with the project evolving into a major database activity with highly skilled informaticians managing the activity. To ensure continued capacity for rapid growth and secure storage for some of the most rapidly growing deposition databases at EMBL-EBI a new storage solution FIRE - the File Replication Service was conceived in 2008 and is still in use today. It stores data for two of the largest archives at EMBL-EBI, ENA and EGA, together with 3 smaller DNA-based archives and a rapidly increasing new bio-image archive.

1) European Nucleotide Archive (ENA)

ENA provides a comprehensive record of the world's nucleotide sequencing information, covering raw sequencing data, sequence assembly information and functional annotation. Data arrive at ENA from a variety of world-wide sources. Access to ENA data in standard formats is provided via a browser, through search tools, large scale file download

We gratefully acknowledge EMBL, UKRI LFCF and SPF and Chan Zuckerberg Initiative DAF, an advised fund of Silicon Valley Community Foundation

and via the API and embodies the FAIR Data Principles which promotes published data to be Findable, Accessible, Interoperable and Reusable [5]. Most data are fully public and available for anonymous download, while a subset of new datasets will be embargoed for a short time - only available to selected authenticated people.

2) European Genome-phenome Archive (EGA)

The EGA is a service for permanent archiving and sharing of all types of personally identifiable genetic and phenotypic data resulting from biomedical research projects and data access requires prior ethical agreement to be in place [6]. Data are stored at rest encrypted and while access to EGA data is provided using similar methods to ENA, but all data leaves EMBL-EBI re-encrypted and requiring unique decryption keys that only EGA can provide to the authorised individual.

B. Archive types and requirements

Traditional computational storage archives may categorise as follows:

- Cold archive - usually low-cost tape archive. Access is not direct and requires some mechanical actions before being online. Usually for Disaster Recovery (DR) purposes.
- Active archive - also called storage tiering. When data value or user interest is known, data are placed on the most appropriate storage to maximize cost-benefit between ease of user access and storage cost.
- Governance archive - driven by regulatory compliance and/or a desire to be prepared for discovery in legal proceedings, with stringent requirements and SLAs. Vendor contracts tend to be complex and the user experience is often an afterthought.

However, the long term archival requirements of ENA and EGA, (as well as other EMBL-EBI archives), fall somewhere in between these traditional archive categories, creating an aggregate requirement to have two archives with one central point of access; one archive being actively accessed; the other providing a disaster recovery facility to guard our mandate to store open data long term on behalf of worldwide communities. Good practices require that there is no vendor lock-in or single technology dependency on the solution used to save and serve the data, mitigating the scenario in which data archivers lose control and ownership of the data as a result of changes to restrictive licensing and commercial interests. Physical infrastructure risks are also taken into account, requiring data to be replicated in distant locations, to survive a disaster affecting any one of those locations. In the case of ENA, this extends beyond EMBL-EBI holdings to other International Nucleotide Sequence Database Collaboration (INSDC) [7] partners. Thus, the chosen solution needs to maintain two internal archives, while appearing as a single unified interface to the users. Those same users, (e.g. ENA and EGA), also require the presentation layer of the archive to be POSIX [8] compliant, available from the EMBL-EBI data centre compute clusters as well as usable to present the data to the outside world through standard FTP and other transfer protocols. As the archive is in constant use, high availability is also a key requirement. In addition to being resilient to hardware failure, the system also has to enable the seamless addition of new back-end storage systems.

C. The challenge of predicting growth rates

At the end of 2008 the total size of ENA was in the region of 100TB [9] and was growing at a rate of 200% each year. As data are not generated locally and come from a huge range of resources, forecasting growth rates has proven to be inexact. At that time EGA was experiencing similar growth. As of the end of 2019 the average growth in the past 10 years has been around 90% (see Fig 1), with a 42% growth in 2019. With other public archives starting to use FIRE the growth expected for 2020, in terms of archived data, is around 55%, including the organic growth of the existing archives.

D. Scaling egress

EMBL-EBI is a global hub for life science data, which means that once data have been received, processed, and value added (e.g. by addition of metadata, cross-references etc.), data are made available for redistribution to the global life science community. Since some institutions still download copies of all the data published, and many thousands of distinct individual data downloads occur every single day, it can be expected that for every petabyte of data stored in FIRE, many more petabytes of data download capacity will need to be supported (i.e. every item of data will be downloaded more than once during its lifetime).

E. Cost of ownership and predicting the long-term re-use value of data

The nature of EMBL-EBI's grant-based infrastructure funding requires it to put out tenders for data centre space every few years, potentially leading to regular physical migration between locations. Reducing the downtime during these transitions requires data to be accessible in more than one place at once. Having data available (cold archive data are not considered as available in this context) in at least two data centres simultaneously forces the infrastructure to be replicated with an associated increase in cost.

An alternative storage model for 'active tiering' – the Hierarchical Storage Management (HSM) system is aimed at reducing costs for an active archive, by transparently tiering the data over several storage layers, each with different performance and cost characteristics, moving the less accessed data away from its original location into less performant cheaper internal storage tiers, depending on usage policies. However, implementing an HSM system efficiently requires the ability to establish data access patterns. over a long tail distribution, where data in the tail of the graph would be moved to slower and cheaper storage systems. However, in the case of life sciences data, 'the long tail' may in fact represent the mainstream research [10].

The 'value' (and likelihood of access) of a particular dataset cannot purely be predicted by its age in the archive (e.g. last in first out) as value depends on scientific context [11] and this can be changed by multiple factors. Some data may gain interest at any point in time, with triggers as diverse as newly developed tools, a virus outbreak, a publication, a need to compare with a newly generated dataset. Datasets used as reference data for particular analyses may continue to hold interest for extended periods (e.g. the 1000Genomes dataset [9]), leading to repeated downloads. Depending on the size of the data and the number of downloaders, those triggers could potentially pose a risk to the infrastructure. With new data continually reaching the public domain and old data being reused unpredictably an HSM solution was deemed impractical and the approach chosen was to store data on a

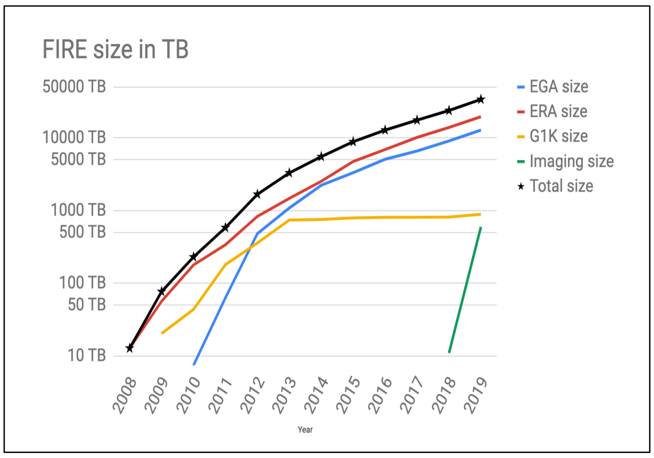


Fig. 1. Growth of Data Projects in FIRE

spinning disk copy and ensure its continual availability via a variety of file transfer protocols.

F. *Storage lifecycle*

Due to the long-term nature of the archive, many publications reference datasets and files directly or through specific archive proxies. To allow those links to remain valid over the years, even as underlying storage changed, the solution had to provide a lasting way to reach the archived objects, while ensuring immutability of those objects.

II. FIRE, A DUAL ARCHIVE SERVICE PROVIDED BY EMBL-EBI SINCE 2008

To summarise, EMBL-EBI needed a data archive solution that would scale continuously over many years in line with the data size; have high availability and resiliency; be abstracted from the underlying hardware to prevent vendor lock-in and disruption at the end of hardware lifecycles; be geo-dispersed in at least two data centres, use a mix of storage types to provide optimum user access, different technologies and cost-benefit, and allow POSIX-compliant file presentation. To fulfil these essential criteria, the File Replication Service (FIRE) was designed and brought into production.

A. *Phase 1 implementation- 2008 to 2018, when nothing fits your needs*

The FIRE project was started in 2008. At that time no mature solution was identified that could ensure an affordable Total Cost of Ownership (TCO) in the long run, so the EMBL-EBI Systems and Networking Teams developed an automated file replication mechanism that could be used by the biggest EMBL-EBI public archives which were already struggling with their growth and storage requirements. This solution is now referred to as the phase 1 implementation.

1) *Data access*

To fulfil the requirement of a storage system usable by the already existing pipelines and workflows, the team of that time decided to ensure the solution could be presented with a POSIX-compliant filesystem implementation. This FUSE implementation required root access to the servers accessing the data. This allowed a granular control on what data was presented inside the EMBL-EBI, and by using that internal presentation of data through already existing communication protocols (i.e. FTP, Aspera, HTTP), leveraged the authentication and authorization of those services obviating the need to implement another layer of authentication on the FIRE solution itself.

2) *Archival process and shared ownership*

Incoming data were uploaded to the EMBL-EBI infrastructure using communication protocols such as FTP or FASP (Aspera) and then enter a data-specific workflow of validation, tagging and curation to ensure contents meet the defined requirements. By the end of the workflow the archiver issued archival actions in an database table inside a project-specific database shared with the FIRE infrastructure.

Each data project also required the following FIRE components:

- Python-based daemon scheduled to pull new archival actions from the project-specific database. This daemon inserted new FIRE actions in the internal FIRE database. The daemon also created action specific jobs in the FIRE owned batch processing cluster based in LSF [12].
- Project-specific staging area isolating project specific IO operations. The resulting absence of shared IOPS among projects reduced the need for quality of service mechanisms. The staging could be co-located depending on each project workflow location and mounted only by the FIRE LSF hosts in that data centre.
- Although a specific LSF host was not required for each project, an internal batch queue was, ensuring that LSF actions would only be consumed by the hosts mounting the staging areas from the same data centre, preventing unnecessary cross-data centre traffic wherever possible.

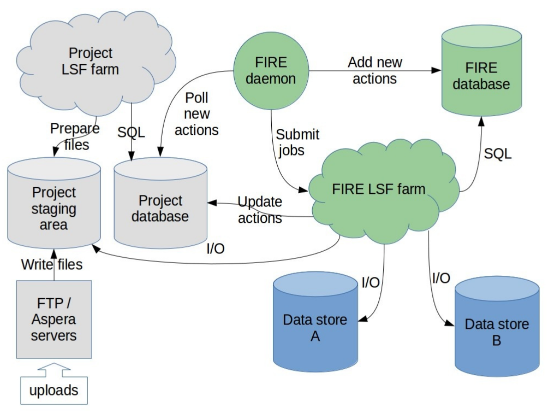
The LSF jobs processed the FIRE actions, represented by a series of IO and metadata events, tracking each action with different states. Once the FIRE actions finished the last steps updated the internal FIRE database and the project database and removed the archived file from the staging area when necessary. In the event of a failure specific exit codes ensured LSF resubmission.

3) *Disaster recovery and data migration*

As seen in Fig 2 below, the phase 1 FIRE implementation uses two different datastores (A and B) to store the same data. To mitigate the risk of data being corrupted because of a hardware or firmware failure, this data replication is done using storage platforms from different vendors. To mitigate the risk of data loss due major disasters the datastores are located in different data centres. With this replication in place, recovery from a disaster preventing data access to one of the datastores requires replication of all data from the online datastore to a newly installed datastore, thus restoring the replication level to pre-disaster levels.

1) *Breaking away from the hardware lifecycle*

FIRE has been designed with the ability to copy data from one datastore to another, for disaster recovery; however, this ability is more often used to migrate data inside a datastore. When a storage element needs to be retired data are moved into the newly installed storage, and this transition is transparent to users. To enable full control of the data and its replica locations, FIRE provides an external file object Identifier (OID) to the user, while managing the replicas of that file individually with as many internal OIDs are needed at any point in time. The external OID is fixed for the whole live of the file, while the internal OIDs, usually one for each datastore, change every time the file is moved to a new storage element.



1) User uploads data to 'Project staging area' through FTP or Aspera
2) Each project validates/prepares the uploaded files before they are listed to be archived in the 'Project database'
3) 'FIRE daemon' detects new actions in the 'Project database' and will create all necessary jobs and internal actions for each of them
4) 'FIRE LSF farm' will pull files from 'Project staging areas' and write to both Data stores, A & B, and update the 'Project database' on success
5) 'FIRE database' will contain and track all the actions and files

Fig. 2. Diagrammatic representation of FIRE – Phase 1 implementation showing data flow

During the migration process, each file being migrated gains one extra internal OID, the file will be copied and validated from one of the existing replicas. The last step of the process removes the old replica and its references to the external OID thus ensuring no existing data consumer tracks that data have physically moved.

B. *Phase 2 - Cost reduction by Introduction of Tape-based archive*

Over time, the biggest data projects like ENA and EGA showed location-specific access patterns (i.e. predominantly one datacentre), allowing the non-accessed replica to be moved into its own cold archive. An in-house Object Tape Archive mechanism, known as OTA, was built, operating as an object store onto which existing data for those specific projects was migrated. This object store added some complexity to the FIRE solution but reduced the total cost of ownership for the overall solution due to the lower running and maintenance costs of tape compared to spinning disks.

Tape requires data to be sent to it as a stream, and the streaming rate needs to be sufficient to meet optimal performance for the write operations. As it would be difficult for the archives to control such a data stream, a buffer was put in place. This comprised a 250TB lustre filesystem, where data were written by the OTA object store endpoints and internal mechanisms then bundled it before being sent to tape using a single tar data stream. This move to tape brought one important limitation. While the archival process, from FIRE perspective, considered the data sent to OTA as replicated, it would not be written to tape until the expected file resulting from the tar operation would be big enough to fill at least 90% of a tape.

Years after the initial implementation of OTA, all the existing data projects were migrated, allowing the retirement of all spinning disk on one of the replicas by replacing it by OTA.

C. *Phase 3 - Third data centre and geo-dispersed storage*

During 2015, EMBL-EBI acquired a third data centre, and FIRE was extended to deploy a 5PB Cleversafe geo-dispersed object store. This had the effect that any data stored on such object storage was locally fetchable by the bespoke FUSE implementation, meaning that the POSIX compliant file system was now available from all data centres, although only two of them were publicly accessible. This geo-dispersed storage was used from that point onwards as the back-end for what from that point onwards we would call the primary (or active) replica, with the secondary replica moved to the OTA (tape) system which *de facto* is a cold archive inside the FIRE active archive.

Geo-dispersed storage like Cleversafe or similar implementations, have internal algorithms responsible for distributing stored data over multiple data centres providing resilience, and allowing data to be readable even if one of the data centres is not available. This high availability mechanism allows upgrades and scheduled maintenance execution without disruption for the end user, while providing a consistent presentation layer on all data centres. Object storage also has the added benefit of using HTTP to read and write the data, while the previous storage used required mounting by the root user over NFS. The benefits of not requiring a kernel-based module like NFS, are many and will be discussed in the next sections.

Positive cost benefit analysis of the new IBM Cleversafe geo-dispersed solution, led to the purchase of new geo-dispersed storage to extend, and later replace, the first one. This has become a regular procedure and today FIRE operates on three Western Digital Activescale X100 systems (also known as HGST) aggregating 40PB of usable space, and the original Cleversafe has already been retired.

D. *Phase 4 - 2017 to 2021 - Transition to standard REST*

After the risk of loss of a data centre was mitigated by using geo-dispersed object storage as the primary replica of all existing archives, a decision was made to increase the serviceability of the FIRE solution. Until this point, any new user project for FIRE replication methodology required its own staging area and project databases, which also required bespoke development and infrastructure integration with their pipelines. As the primary replica was already compatible with the Amazon S3 REST application programming interface (API) it was a logical step to offer similar REST API access to users. With some users already requesting REST access to FIRE, the impetus provided by two new funded large data projects drove work on the creation of a new REST API.

1) *Maintaining two infrastructures*

With the expectation that transitioning research pipelines from the existing database-based API to the new REST API was likely to take more than one year, a new monitoring tool was deployed for basic monitoring on all servers related to FIRE. The Open Monitoring Distribution (OMD [13]) was chosen as it leverages Check_mk [14] as the server agent and monitors over 30 items by default, including CPU load, mount points, and network interfaces. OMD provided a bird's eye view on the existing complex infrastructure, highlighting several bottlenecks, and allowed the team to invest time in the new infrastructure, while focusing only on relevant maintenance of the existing infrastructure.

The original infrastructure included the following components:

- 10 LSF servers in data centre A and 6 in data centre B.
- 2 servers providing access to the tape libraries (Spectra T950 and IBM T4500)
- 14 Gbps Brocade SAN switch to connect the tape libraries
- 1 Oracle database for FIRE and a standby in a second data centre for DR
- 1 Oracle database for OTA and a standby in a second data centre for DR
- 5 staging areas, one per data project, mounted by the LSF servers of the same data centre
- 13 PB of data replicated in Object Store and tape
- 2 metadata servers for FUSE filesystem access on each data centre

The new REST API particularly benefitted new users by lowering the entrance barrier. While existing data project users were able to archive all the data they were receiving with this infrastructure, they could also benefit from the changes if these were adopted by unlinking existing staging areas and the project databases from the FIRE LSF servers, thus ensuring all IOPS of those storage were only used by their internal processes.

2) *Load balancers and DNS alias*

Where possible, IP addresses and hostnames were replaced by the use of DNS aliases, which allowed services to be provided via a HAProxy load balancer. After monitoring the resultant traffic, the most used services (such as the metadata servers) were provisioned on virtual machines – leading to a total of 8 metadata servers on each data centre. The existing FUSE implementation already accepted multiple metadata servers, adding the load balancer ensured the stability of an already resilient capable solution.

3) *Containerized infrastructure set up*

With the experience acquired from the initial implementation of FIRE using Python, a decision was made to rewrite from scratch a new solution based on Java and Spring boot. Spring boot facilitates the containerization of Java applications, it is relatively simple to learn and forces a more rigid code structure than Python.

4) *CI/CD from Spring Boot to Kubernetes*

As the initial REST endpoints were being developed with Spring boot the first Kubernetes cluster was installed as a development environment, using 10 Virtual Machines (VM) with 6 CPU each and 16 GBs of RAM and minimal local disk to prevent its use for caching. Using an internal GitLab repository manager helped the FIRE team develop Continuous Integration pipelines which connected the local repositories of each developer with the development Kubernetes environment. Each developer was made responsible for creating unit and functional tests for every endpoint created, and mandatory code review ensured alignment of naming conventions and software practices. To ensure development continued while the users tested the first implementation of the REST API, a second cluster was created, this time in a different data centre, and configured as a Continuous Deployment step in the GitLab pipeline, ensuring that only code that had gone through the development environment ran on the test environment. The test cluster was provisioned on 10 VMs with the same sizing as the development cluster, ensuring pod (groups of containers inside Kubernetes) sizing could be reused from the development cluster.

Code coverage checks, and unit tests were automated, so that the initial container image was created and saved into the GitLab repository only if all those tests were passed. The resulting image will then be used, without being recreated, on each environment, forcing codebase alignment on all environments. This image reuse implies that the profiles used to run the container must handle the different configurations for each environment, since different databases and backend storage were used, but this also ensured that no configuration was embedded inside the codebase.

While users were functionally testing the test environment and starting to integrate their own pipelines with it, the focus switched to deploying two production clusters, one on each publicly accessible data centre. These production clusters used 6 physical servers each, with 64 CPU and 128G of RAM.

A round robin (RR) DNS was configured over 6 IP addresses assigned to MetalLB [15] layer 2 configuration, with the aim to distribute traffic among the 6 bare-metal servers and applied to all environments for consistency, including development and testing environments. Since the users use the RR DNS 6 Kubernetes Traefik [16] services are run to ensure the 6 IPs (virtual IPs) are properly configured. With this configuration an unscheduled reboot of one of the servers triggers its virtual IP relocation to another server.

5) *Capacity planning and GDPR*

Inside EMBL-EBI one single team is responsible for providing service metrics to the end-user, ensuring GDPR regulations are followed. FIRE infrastructure uses a data centre-aware fluentd [17] configuration to send all logs to an existing syslog setup in each data centre. These are anonymized but provide a quantitative overview of data being uploaded to the system as well as data being downloaded. To ensure all logs are sent to the syslog servers an internal policy was defined for all containers to send their logs to the local output as Kubernetes allows fluentd to collect them on every node where they are generated and send them to syslog without need of controlling logs from the application layer.

6) *Scaling containerized services*

In the long-term FIRE should be maintainable by a small team and so the system needs to avoid unnecessary complexity. The original system scaled effectively by solving the problem of replicating data using only an LSF batch queue and that design has been translated to run using Kubernetes as the batch engine, aiming for the same simplicity. However, a relevant improvement has been the addition of a second subset of containers focused solely on reading data (i.e. no write actions), dividing the infrastructure into two services with the same codebase but with different database access. These two services are hidden from the users by the Traefik edge-router which publishes them and sends the specific requests to one or another service depending on the need of read or write on data. Both services span the same image inside Kubernetes, but are provisioned with different database configurations, relieving the central database of any read-related work since the read service pods can use the read-only disaster recovery databases. As previously discussed, EMBL-EBI's general aim is to share its data with the world, and to support this, the connectivity with the outside world has recently been increased to 100Gbps Since this is expected to remove a current bottleneck for large-scale data transfers, the risk of greatly increased consumption of data having a negative impact on the ability to continue ingesting new data from inside the EMBL-EBI requires such mitigation measures.

7) 'On-time replication' or 'eventually replicated'

With the addition of the geo-dispersed storage as the primary replica the requirement of data centre failure resilience was already being fulfilled with one single copy of the data. Following agreement with all the existing users, data replication into OTA was reimplemented as an 'eventually consistent' mechanism. When using the FIRE API, the user will receive its archived object OID when successfully copied to the primary replica, and the secondary replica will be queued internally for FIRE to execute when possible.

This 'eventually consistent' property has several high impact benefits; the first of which is being able to do maintenance on the secondary replica without ingress service disruption. The second benefit is that the user receives the OID of the uploaded file faster than with the first implementation, since the first version required both copies to be validated for FIRE to provide the OID to the user, thus making that response as slow as the slowest of the two replicas – the tape. This also allows FIRE to ingress higher data volumes than OTA can handle for short periods. Finally, data only needs to be pulled from the staging area once, so half of the original read IOPs load on such staging areas is removed.

8) Leveraging community standards and open source tools

Since first presented in 2006, Amazon S3 has become a de-facto standard in the life sciences community, and so read-only S3 support was built into the new FIRE API. This S3 implementation adds an authentication layer to the previous fuse development. Leveraging already existing S3 clients (such as Goofys [18]) ensured the team was able to focus solely on the REST API, based on Java, instead of adding the requirement of learning another language, like C or Go for the API client. REST API development has been focused on industry standard software and infrastructure implementations for sustainability reasons. Open source tools now used include GitLab [19] (repositories contain application and infrastructure configuration), Foreman and Puppet [20] (to deploy and configure basic services), SaltStack [21] (to configure load balancers and Grafana dashboards) and Rundeck [22] (to provide audit trail for cron jobs).

III. Design lessons after 12 years of intense usage and continual growth

From 2008 to 2012 the total data volume archived in FIRE was 1.4PB, which is approximately the volume archived during 2013 alone. Planning for such growth rate is remarkable, and while the yearly growth rate has seen some reduction the archived volume still grew by 42% during 2019, with an average of 750TB per month. In contrast, the average total data download from EMBL-EBI was in the region of 3PB per month.

A. Bottlenecks do not disappear, they just move

It is a fact that bottlenecks will appear when a service is being used with enough intensity and being able to identify such bottlenecks is key to being able to support continuous growth. Any time, or effort invested on any other part of the system may not show any immediate benefit.

B. Cache reaching cost-benefit limits

Data provision via FUSE in the phase 1 implementation relied on an NFS cache to optimise file retrieval. Requested data was copied from slower, more resilient object storage to the FIRE cache before being returned to the user. This caching mechanism enabled FIRE to satisfy a high percentage of requests with extremely low latency as every subsequent request for the same data was retrieved from the cache. Over time both total data and user base increased resulting in the majority of cached data being expunged before a second request was received. The overhead of copying, monitoring and clearing the cache no longer provided any performance gain so was excluded from subsequent implementations of FIRE.

C. Tape lifecycle

When tape was introduced in 2015 as the secondary replica for some projects, the available tape generation was LTO-6; since then LTO-7 and LTO-8 have been made available with a new generation likely every two to three years, each time doubling in capacity. By 2018 all projects finished their secondary replica migration to tape, and by 2019 the used tape bank consisted of around 10PB in each LTO generation, with a further 10PB stored in 3592-JD tapes.

Read-write capacity is currently provided by 12 LTO drives in a Spectra Logic T950 with a further 3 IBM drives (3592-JD) installed in an IBM TS4500 library. The number of installed drives at any time was defined as that required to meet the throughput of FIRE when running optimally and this was only achievable by ensuring a constant input data stream. As previously noted, TAR (rather than the less performant LTFS) was used on a lustre buffer file system, to bundle enough files to into one single uncompressed file to fill a tape. Such a solution was felt to have better performance than any other but has the following drawbacks:

- Only FIRE/OTA can use such drives and tapes
- Migrating away from such a solution isn't possible by just scanning the tapes
- Recovering one single file requires recovering the full tape
- It is not possible to migrate from LTO-N to LTO-N+1 without recovering all tapes
- The current setup provides few idle drives on average, that could be used for disaster recovery

D. Sharing infrastructure with your clients

The original database-API relied on infrastructure not under the full control of the FIRE team – the mounted filesystems from where the data were pulled and the databases where the user's workflow stored the archiving instructions. The consequent requirement to liaise with each user team before making even minor changes convoluted maintenance and migration procedures.

The following are lessons learned during the transition from the database API-based system to the REST API implementation.

E. Early user involvement

After several consultations with the technical leaderships of the biggest users, the first general design of the new REST API was shared with all the stakeholders for feedback. From these initial conversations a test environment was created and with it the feedback loop with our users started to help us rank our development priorities. Having at least one new team willing to archive data to FIRE via a new API, and without experience or infrastructure built around the existing database-based API has had a very positive impact on the speed of development and feedback. As well as sometimes daily direct communications with user teams, a FIRE Users Group was used with all stakeholders for a quarterly validation of the new REST API project evolution.

F. Measure, deploy, automate, test, repeat

To ensure proper development speed any developer requires enough time to be focused on the task at hand, and before the move to extensive monitoring, code review and unit testing, any even apparently trivial code changes triggered sufficient numbers of user-reported errors to disturb development efforts. Reducing the reactive interruptions caused in this way required a proactive approach where all existing systems were redeployed from scratch and CI/CD was used to deploy the application on the LSF cluster, OTA servers, Kubernetes and all other servers. While gradually replacing all previous installations, daemon monitoring and logging housekeeping was introduced by default. To guard against inaccurate capacity planning the infrastructure was extended over Virtual Machines to ensure available capacity. After all of this, with fully automated deployments and configurations, extending the infrastructure became mundane work.

G. Simplicity tends to be harder than expected, but is rewarded

Not all developers understand the 'legacy mindset' which means that the next developer should have it easier than you. Making things simpler is not always simple. Ensuring that no niche knowledge was needed to maintain the infrastructure was important to ensure future maintainability.

H. Naming conventions

While it may appear a limited investment to name servers with a descriptive naming convention, it reduces the time invested in identifying servers and reduces human error. The same applies inside of the code by properly naming variables and using a review process to check your co-workers do the same. In the same way DNS aliases that help users to identify services and environments while allowing the system administrator to balance or move load at will have been a great benefit when preparing maintenance works, since moving a DNS alias allows maintenance work to be transparent to end users.

IV. Work in Progress

A. Security needed to allow external access

During 2020 and 2021 the project will implement a Role-Based access control (RBAC) in order to provide users full control of who can access each file. The security implementation should allow users with certain roles to change the ownership and permissions of files inside the project they belong to. This activity will be done in agreement with the existing pipeline owners to ensure the minimum amount of disruption reaches the external users and data depositors. Part of this security effort is in response to the need for basic internal authentication and authorization for depositors to be allowed to push data directly into the FIRE archive. However, the ability to present our S3 data presentation directly to the outside world is also taken into account.

B. Data centre-specific cache inside the REST API

Work has started to gather statistics to inform sizing of an object store-based system for each data centre, aiming to reduce the traffic generated by the geo-dispersed nature of the current primary storage.

C. Cloud and associated costs

Legal and technical resources have been invested in finding a suitable method to provide access to data from public cloud tenancies. So far, all approaches to uploading public data to cloud providers have been solved by providing access from EMBL-EBI cloud infrastructures based on OpenStack or internal Kubernetes services. Our archivers require that any access to private and public data needs to be registered, gathering use statistics is necessary for reporting to funders, as well as identifying and understanding community interests over time.

D. Opening tape access to other services

The current tape libraries can only be used by FIRE, and its object storage presentation (written in Python2) has reached the point where it requires a refactoring from scratch. The OTA solution was created in-house because there were no reasonably priced appropriate products in the market at that time, and for the past years it has provided a great service. Having said that we are aiming to fully replace or refactor OTA. We have identified several potential object storage interfaces for our tape libraries and we are considering options.

While the OTA performance (on write) may be higher than any LTFS based solution, the current solution is less efficient during read operations, which has implications for disaster recovery actions. In the long term, one has to consider the Recovery Time Objectives (RTO) in case of such a disaster recovery. Tape use has increased linearly with the amount of data being archived, increasing from 200TB to 1000TB each month, and this is pushing our OTA solution beyond its current disaster recovery capacity. There is a plan to extend tape access by adding more drives, to allow faster restore actions.

If the expected yearly data growth continues, our project will reach the Exabyte milestone before 2030, and by then our solution will likely be very different from the current one.
With the experience gathered over the last 12 years and the subsequent evolution of the FIRE software-defined storage implementation, it is clear that abstracting the service from the physical infrastructure where possible, assists in simplifying the implementation of future improvements while being transparent to users.

Acknowledgment

We gratefully acknowledge previous members of the FIRE team for their work over many years; particularly Petteri

Jokinen, the original architect of FIRE and OTA, and also the systems infrastructure and database teams for their help in maintaining the FIRE networking, physical storage and databases respectively. Finally, we would like to thank our funders

REFERENCES

[1] EMBL-EBI Annual Report 2018 https://www.ebi.ac.uk/about/digital-bookshelf/publications/EMBL-EBI_Scientific_Report-2018.pdf

[2] Vamathevan,J., Apweiler,R. and Birney,E. "Biomolecular data resources: Bioinformatics infrastructure for biomedical data science." Annu. Rev. Biomed. Data Sci., 2, 2019, 199–222.

[3] Cook CE, Lopez R, Stroe O, et al. "The European Bioinformatics Institute in 2018: tools, infrastructure and training." Nucleic Acids Research. 2019 Jan;47(D1):D15-D22. DOI: 10.1093/nar/gky1124K.

[4] Cook CE, Stroe O, Cochrane G, Birney E, Apweiler R. "The European Bioinformatics Institute in 2020: building a global infrastructure of interconnected data resources for the life sciences." Nucleic Acids Research. 2020 Jan;48(D1):D17-D23. DOI: 10.1093/nar/gkz1033R.

[5] Wilkinson M.D., Dumontier M., Aalbersberg I.J., Appleton G., Axton M., Baak A, et al. The FAIR Guiding Principles for scientific data management and stewardship. Scientific Data. 2016; 3:160018.

[6] Lappalainen I., Almeida-King J., Kumanduri V., Senf A., Spalding J.D., Ur-Rehman S., et al. "The European Genome-phenome Archive of human data consented for biomedical research." Nat. Genet. 2015; 47:692–695.

[7] Cochrane G., Karsch-Mizrachi I., Takagi T. "The International Nucleotide Sequence Database Collaboration". Nucleic Acids Res. 2016; 44:D48–D50.

[8] IEEE and The Open Group https://pubs.opengroup.org/onlinepubs/9699919799/ retrieved 10/02/2020.

[9] Cochrane, G., Akhtar, R., Bonfield, J., Bower, L., Demiralp, F., Faruque, N., et al. "Petabyte-scale innovations at the European Nucleotide Archive." *Nucleic acids research*, *37* (Database issue), (2009). D19–D25. https://doi.org/10.1093/nar/gkn765

[10] Ferguson, A., Nielson, J., Cragin, M., Bandrowski, A. E & Martone, M.E. "Big data from small data: data-sharing in the 'long tail' of neuroscience." Nat Neurosci 17, 1442–1447 (2014). https://doi.org/10.1038/nn.3838.

[11] Pryor, G. "Multi-scale data sharing in the life sciences: Some lessons for policy makers." International Journal of Digital Curation, 4(3), (2009). 71-82

[12] LSF Platform Load Sharing Facility [software] IBM https://www.ibm.com/us-en/marketplace/hpc-workload-management

[13] Open Monitoring (OMD) [software] https://omdistro.org/

[14] Check_MK [software] https://checkmk.com/

[15] MetalLB [software] https://github.com/metallb/metallb

[16] Traefik The Cloud Native Edge Router [software] https://traefik.io

[17] Fluentd [software] https://www.fluentd.org

[18] Goofys. [software] https://github.com/kahing/goofys

[19] Gitlab [software] https://about.gitlab.com/

[20] Puppet [software] https://puppet.com/

[21] SaltStack [software] https://www.saltstack.com/

[22] RunDeck [software] https://www.rundeck.com/open-source