Investigating a new monitoring solution for FTS

a Summer Student Project
by Antoni Marti Albons
supervised by Steven Murray
Summer of 2025

Table of contents:

Introduction

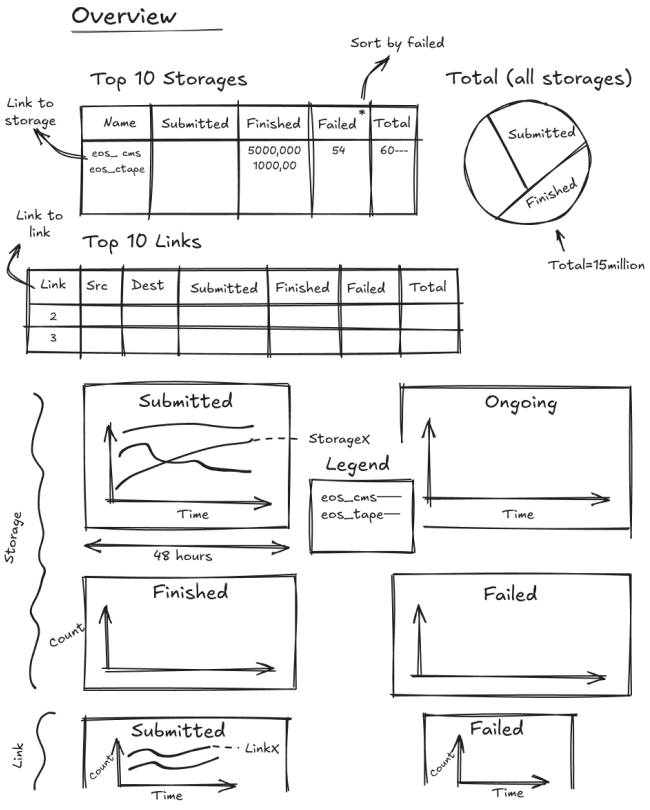
The File Transfer System team is dedicated to maintaining a software used by CERN and collaborators to automatize the transfers of big volumes of files between servers.

I joined them to work on a new website to show monitoring metrics of their system. They already had one, but it was not intuitive to use. Navigating it felt like you were fighting against the page. It also had performance issues, some pages would take up to one minute to load or simply timeout. In theory the existing solution could have been improved to address the performance issues but it also has the problem of using two redundant technologies not used elsewhere within the project. It uses Django to handle the endpoints and Angular to create the interface. Flask is used to provide the REST interface of the project and with some extra effort and care can be used to replace these two big dependencies.

So it was decided to create a new monitoring solution that solved the above problems. The solution must be as frictionless as possible, not overload the system and must not introduce yet another new framework.

First Steps

When I arrived, I received a picture of a whiteboard draft that suggested how the new Overview page should look like. It looked something like the image on the right. So I started making a page that showed tables, time series plots and pie charts.



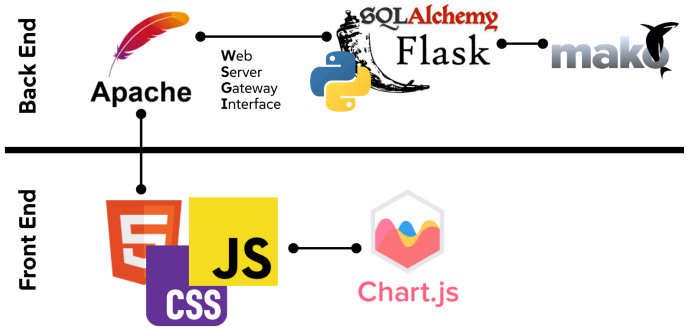
When I was doing this, the exact concepts that the tables and graphs were going to represent changed. For example the per-storage and per-link graphs were later combined into one summarizing the whole instance. The failed and finished graphs were also combined with an a summary pie chart located next to the right. Archiving and retrieving graphs were added. Multiple tables were placed at the bottom of the

page for various reasons and later reduced to one.

The backend did not initially provide all the data that I needed to plot. I started my work on my own box, which was running its own FTS instance. I could share my progress with anyone connected to CERN's internal network.

After a bit more than one month I had the overview page done, with a search bar that could be used to go to the Storage, Link, Job or Transfer's pages, but they had nothing in them. By this point the page was shared with the experiments that use FTS, this is Atlas, CMS and LHCb, and we asked some members of their teams if I could interview them, to see what some users of the system would like to have added. The interviews gave us new ideas, like a page summarizing the transfer errors. You can view them on the old system, but they are only grouped if they are exactly the same, so you often get multiple results for the same repeated message, just the fine name changing. Some form of smart grouping that ignored file names was suggested. We also learned that one of the main reasons the experiments use our monitoring is because they can get live information from the system. Some of them already had a monitoring team in charged to store and provide historical information about the services they use, including FTS, so they weren't that excited about the new graphs, because they already had a similar thing, but their in-house monitoring didn't provide live information. We still kept the graphs because not all FTS users have a specialized monitoring team. We also discovered that the concept of a "good link" is a bit ambiguous. A link can be considered good because it has a low failure count or because it has high trughtput, since both can provide useful information we decided to get rid of the top 10 tables and provide a single table of all links that can be sorted byt these different metrics.

Stack



The technologies behind the new solution are the following:

On the server side:

- **Apache**: Provides access to the pages and endpoints used by them, the behavior of those is defined by Flask.
- **Flask**: A Python framework to create WSGI applications, those allow python to respond to http requests, here they return pages or JSON. To modify the pages in the server, it uses Mako.
- **Mako**: A templating language that allows you to embed python in a HTML document.

On the client side:

- **HTML**, **CSS** and **JS**: To display and update the pages. For the graphs, JS uses Chart.js.
- **Chart.js**: Library that allows you to create plots and modify their behavior.

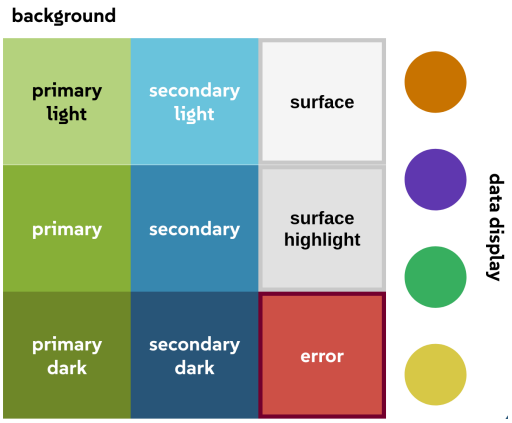
The only technology that wasn't already in the project is Chart.js. With this, Django and Angular are no longer necessary, which makes the pages easier to maintain for a team that doesn't have a dedicated web developer.

Char.js wasn't the first charting library I tried Before I started with C3 which is a simplification layer of the much more popular D3. But after using it for some time, there were too many things we needed that weren't easy to do or supported with C3, so the switch was made.

Design

Since no framework for the styling was used, if I wanted the UI to look consistent, I had to come up with something to enforce that. To use the same HTML in multiple pages I've used two approaches, if it could be generated in the backend, I used Mako, creating the section I wanted to reuse in a different document and importing it. If it was content that needed to be updated, I generated it with JS, and it can be used anywhere importing the module that implement's it. This covers the structure of the page, but not the styling, which is the most important to make everything look like it belongs to the same place.

The same header and footer templates were included in every page. A common.css stylesheet is linked in the header template. Some generic CSS variables are defined in this stylesheet for things like color, spacing, transition duration... Whenever an element needs to be separated by a small amount from another I could use the *--small-spacing* variable, if I needed something to stand out from the background I could change the background color to --surface. This ensures consistency and, as a plus, allows you to easily change things like the color pallet. The image on the right is the color pallet that I designed for the website, based on the colors of the FTS logo.



Challenges

Some of the concepts proposed for the new interface didn't have the corresponding data on the system, the graphs were such a case. My supervisor Steve had to create a new table in the database and a way to collect the record to be inserted into it. Other concepts like throughput had to be put aside for the moment due to the same reason.

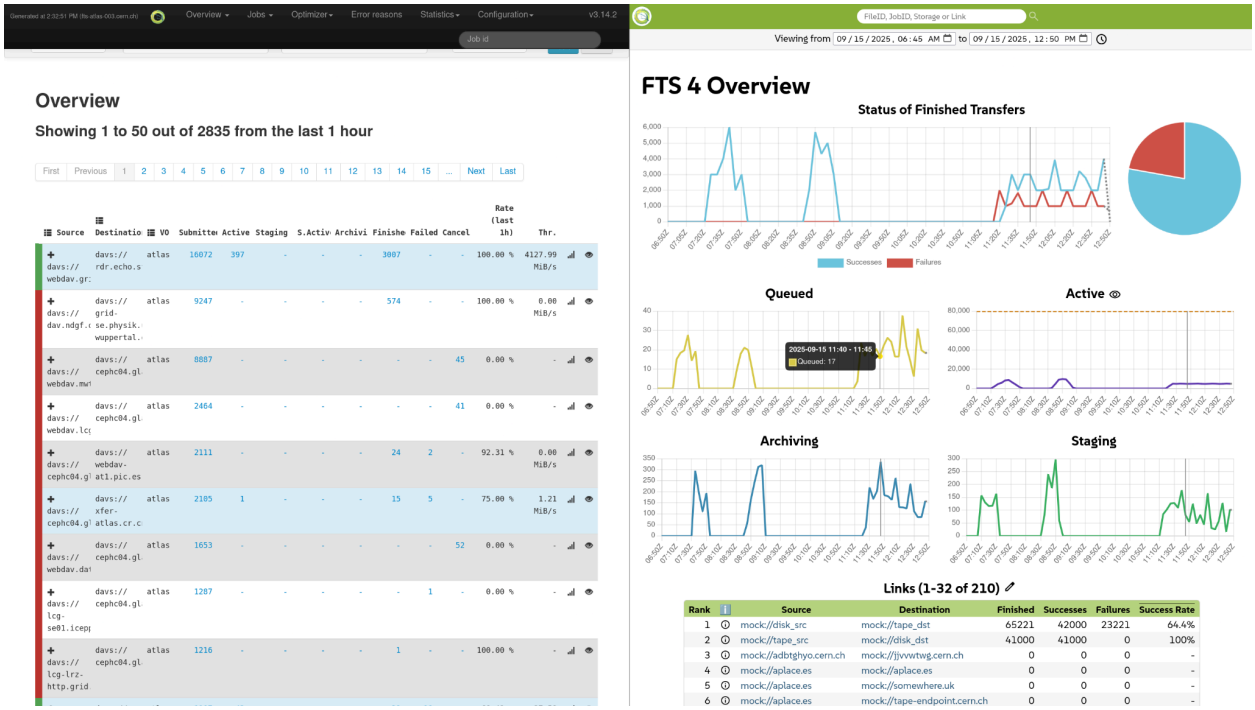
Deployment

Once the end of the summer approached, and the website was more stable, my development work was merged into the main project and deployed to our team's instance. That allowed to see the system with some real data. We then realized that some things didn't look as intended with it. The active and queued values were several orders of magnitude apart, so we had to separate the plots.

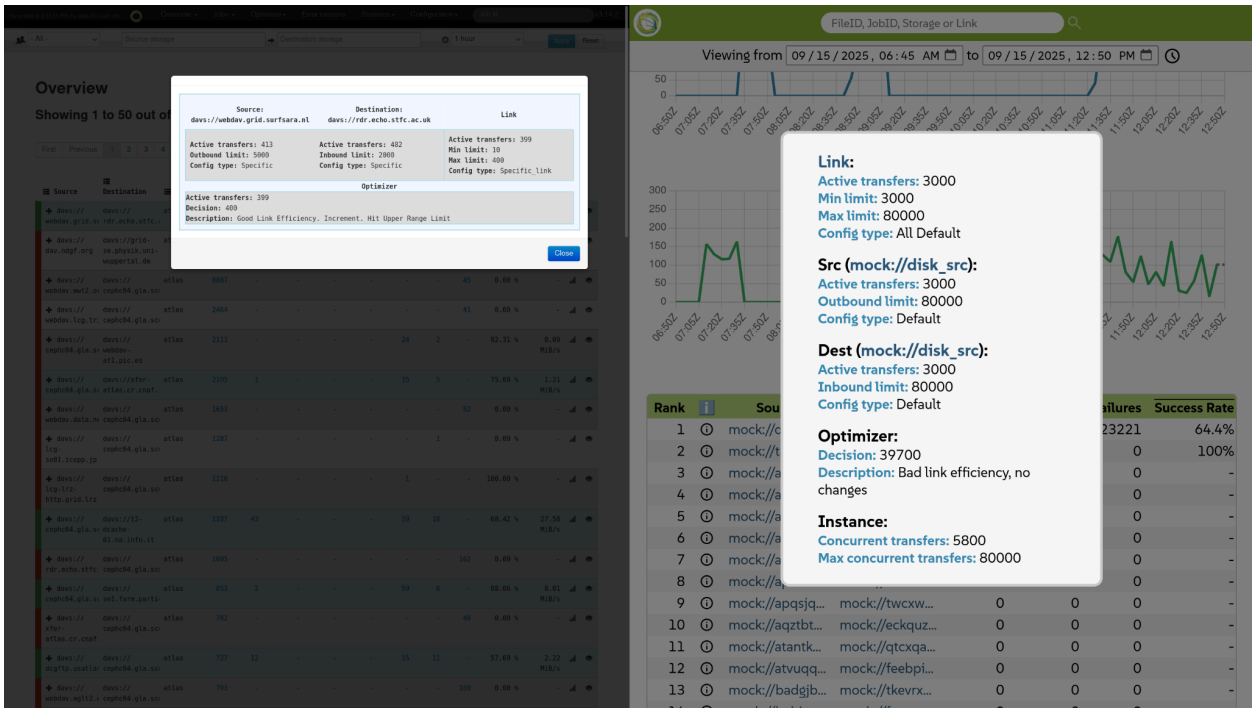
After the first deployment, it was easy to introduce new changes as they were being made.

Results

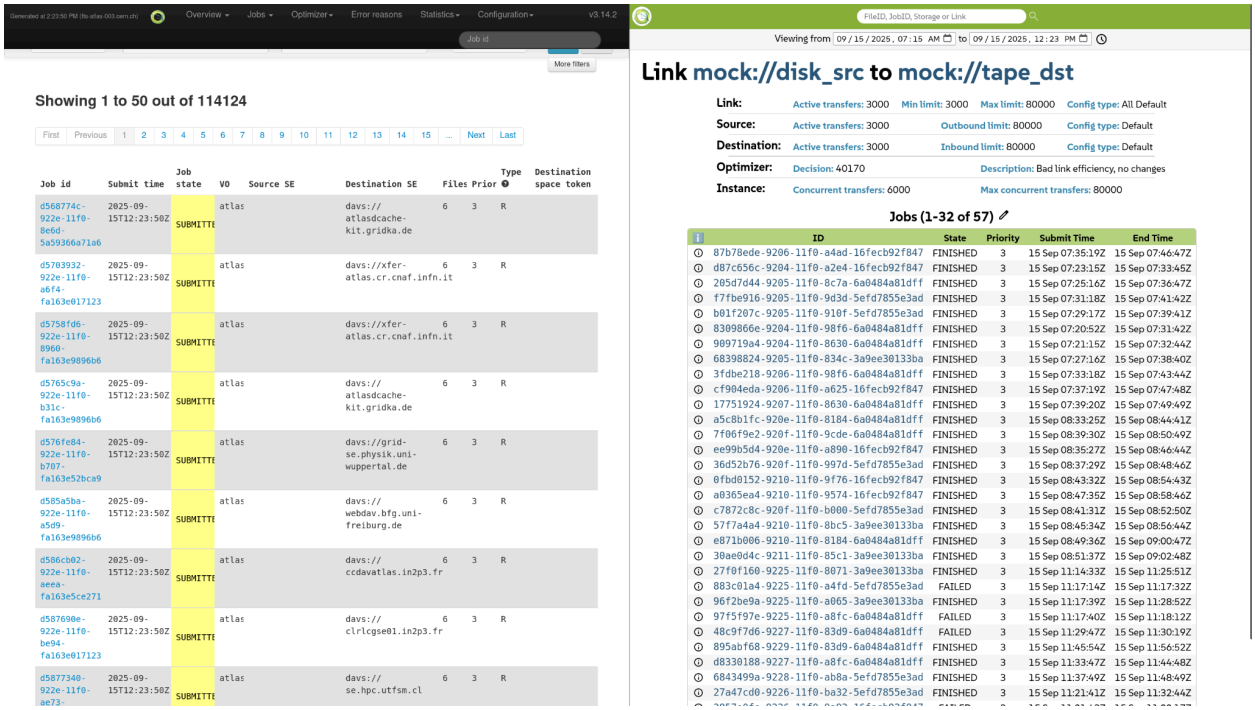
Next are some screenshots of the same pages comparing the previous solution with the new one.



Note that the old system hasn't adapted well to the new aspect ratio, the topbar is covering some of the tools.



This is what you see when you click on the details button of a link.



This is the dedicated link's page. Now it is much more information dense than before.

Final thoughts

This project has been a great opportunity to learn how websites can be made with no JS frameworks. It was the first time I did things in this way, and it has improved my understanding of these technologies in a way that couldn't have happened if I kept using technologies that hide this details.

Since FTS doesn't have anyone specializing in web development, I had to take a lot of the design decisions myself. Although the new monitoring system does not implement all of the features of the old one, I think I have managed to provide a good foundation for the future of FTS's monitoring.