017 - Deploying eScriptorium online: notes on CREMMA's server specifications

Alix Chagué, Thibault Clérice

Published on December 22, 2023 on the Blog "A research (b)log" at
https://alix-tz.github.io/phd/017/.

017 - DEPLOYING ESCRIPTORIUM ONLINE: NOTES ON CREMMA'S SERVER SPECIFICATIONS

Alix Chagué¤ and Thibault Clérice¤

¤*ALMAnaCH, Inria, Paris.*

eScriptorium is a web application designed to perform automatic text recognition campaigns, by default powered by the OCR/HTR engine Kraken. It comes in a decentralized form, meaning that the application is not distributed by a single organization but can, on the contrary, be deployed by several actors on many different servers. In fact, you can also deploy eScriptorium on your personal machine, simulating a local server.[1]

As eScriptorium is gaining attention, more institutions are interested in building their own server to host the application and offer it to their associates. At Inria, we deployed eScriptorium for the first time in 2020, specifically for the project called LECTAUREP which we ran with the French national archives between 2018 and 2021. While the initial server was hosted on a virtual machine, without any GPU, and open to a relatively small amount of users, our current eScriptorium application already counts nearly 500 users and will soon be hosted on a much different server infrastructure, funded by the CREMMA project. Between the original LECTAUREP-eScriptorium server and the CREMMA server, we moved to a dedicated server (`Traces-6`) for which we invested about 20K€.

Since I have been regularly in touch with people from different institutions who were looking into buying the hardware to create their own server for eScriptorium, I thought it was largely time to put all the deets in writing!

To write today's post, I'm very happy to welcome a second pair of hands: Thibault Clérice's. His expertise and involvement in designing CREMMA server are crucial here!

Let's first discuss some technical requirements, then we'll describe how the CREMMA server was designed. We finish with some very important remarks on the necessity (or not) to build a server and on useful alternatives for the community!

Should you buy GPUs?

GPUs (or Graphics Processing Units) are not mandatory at all when you use eScriptorium. This is the reason why it is perfectly acceptable to run eScriptorium locally, on your own computer. Actually GPUs are not even mandatory to train Kraken models: training can be done on CPUs (your computer's processor), they will simply go much much much slower.

That, however, is true for personal or light use of the training features. If on the contrary you create a server open to dozens of users or more, then connecting eScriptorium to GPUs is very much a good idea: since training a model on a CPU alone can take 2-3 days (or much more), you don't really want 10 users to start a training task at the same time. In the absence of shared GPUs, their training will be queued for days or even weeks and the overload might degrade the

experience of other users on the rest of the application. As long as we are building an infrastructure (and hopefully sharing costs), we may as well enhance the experience of everyone, no?

This being said, you shouldn't rush and go buy a GPU right away. Instead, you should first look at options to *optimize* its usage or at infrastructures that are already available to you. For example, the FONDuE infrastructure, at the University of Geneva, doesn't use the GPUs only for eScriptorium: they connect their application to a cluster which is used by researchers for intense computation tasks outside of eScriptorium (it's an HPC with a university-wide queue controlled by SLURM). This is a very good solution for optimization, because training Kraken models is not a constant activity: if the GPU is dedicated to eScriptorium only, then it will be used for a few hours here and there, not even at 100% of its capacity. Think of it: users of the application will usually need to train a model at the beginning of their transcription campaign, therefore once they have an accurate model, they will focus on using the model for prediction, which doesn't rely on the GPUs (and Kraken isn't really optimized for GPU usage at prediction time anyway).

Other possibilities include connecting the server to a completely physically separate cluster where training jobs are submitted. This is a possibility that several people told me they were exploring, but I don't know if anyone has set it already. Why would you opt for a solution with an external cluster? To replace some huge investment costs (original funding) with some smaller (but much more regular) functioning costs: for example, for CREMMA, nearly half of our 40K€ budget was spent, in 2022, on buying two A100 graphic cards from Nvidia. When using someone else's GPUs, not only you save the money you would spend on the hardware, but on top of that, you contribute to optimizing the use of other GPUs already in place. Another reason is because you might not have the human resources to administer the system and the GPUs. There are multiple calculation clusters created for Academia (of the top of our head: Jean Zay or Calcul Québec), and you could even consider using commercial solutions as well (like AWS, Google Cloud and the like). Then, your money is spent on the actual computation and not on making the computation possible in the first place.

Fair enough, plugging eScriptorium's task manager to an external server might not be that simple. However, for smaller groups of users, it is also worth taking into account that it is perfectly possible to train Kraken models using Kraken directly (through an SSH connection to a (super-)cluster, for example) before uploading them into the application. In such a case, eScriptorium is only used for its ergonomics, not as a simplified interface to train models.

Let's summarize the point here: GPUs are not always a must-have for eScriptorium or Kraken, so you should definitely consider first and foremost your future usage. They currently represent the biggest share in the hardware expenses to build a calculation server. There are options out there where you don't spend 10K€ to buy a GPU but rather connect to an external, ready-to-use service. Or, if you do decide to spend the money, you should consider ways to maximize its usage for other training tasks, possibly outside of eScriptorium.

Some considerations on storage

Normally, eScriptorium is used as an (assisted) annotation environment to obtain the transcription of documents. You would use eScriptorium:

1. In a preparatory phase:

    - (1a) to produce training data, and
    - (1b) to elaborate (aka train) performant segmentation or transcription models;
2. In a production phase, but only for relatively small corpora, to apply segmentation and transcription models and manually correct the results (in which case the size of the corpora must be compatible with the scale of what an individual or your assembled team can process);
3. In a post-production phase, including for samples of a very large corpus, to easily visualize and control the result of the (large-scale) automatic prediction and potentially correct it (cf. n°2).

On the other hand, large scale transcription campaigns should probably be led with Kraken in the command line directly (so only n°1 and n°3 necessitate eScriptorium). Thibault has even produced a small python library to design such campaigns (RTK, for Release the Krakens) which was recently used in a paper[2] where a 38.5M token corpus was produced. In some cases, n°1b even benefits from being performed outside of eScriptorium, since the application offers a very limited control over Kraken's training parameters.

This has several consequences on the way you should consider storage on a server dedicated to eScriptorium. Duplicates of images are created on the server while they are being processed in the application, but they should always be considered as such: temporary duplicates while phase 1, 2 or 3 are under progress. They shouldn't be considered as if eScriptorium was 1) an archiving solution for transcription projects, 2) a querying interface to explore a corpus or even 3) a publication environment for a minimalistic digital edition. eScriptorium is only one brick --an early one even-- in the corresponding pipelines. Instead, the original image files should be stored somewhere else, in an adapted data warehouse (like Zenodo, Nakala, etc.), or published in digital libraries under the responsibility of their owner (like Internet Archive, Gallica, etc.).

What this means when designing a server to host eScriptorium is that its storage capacity should of course be big enough to store the temporary image files,[3] while users are working on their annotation, aka the active projects. However, this storage doesn't need to be expended all the time and it should also be ok to flush the terminated projects: at that point the images and their annotations should have been archived on more appropriate data warehouses by their creators, and it should be their responsibility.

Don't forget the RAM!

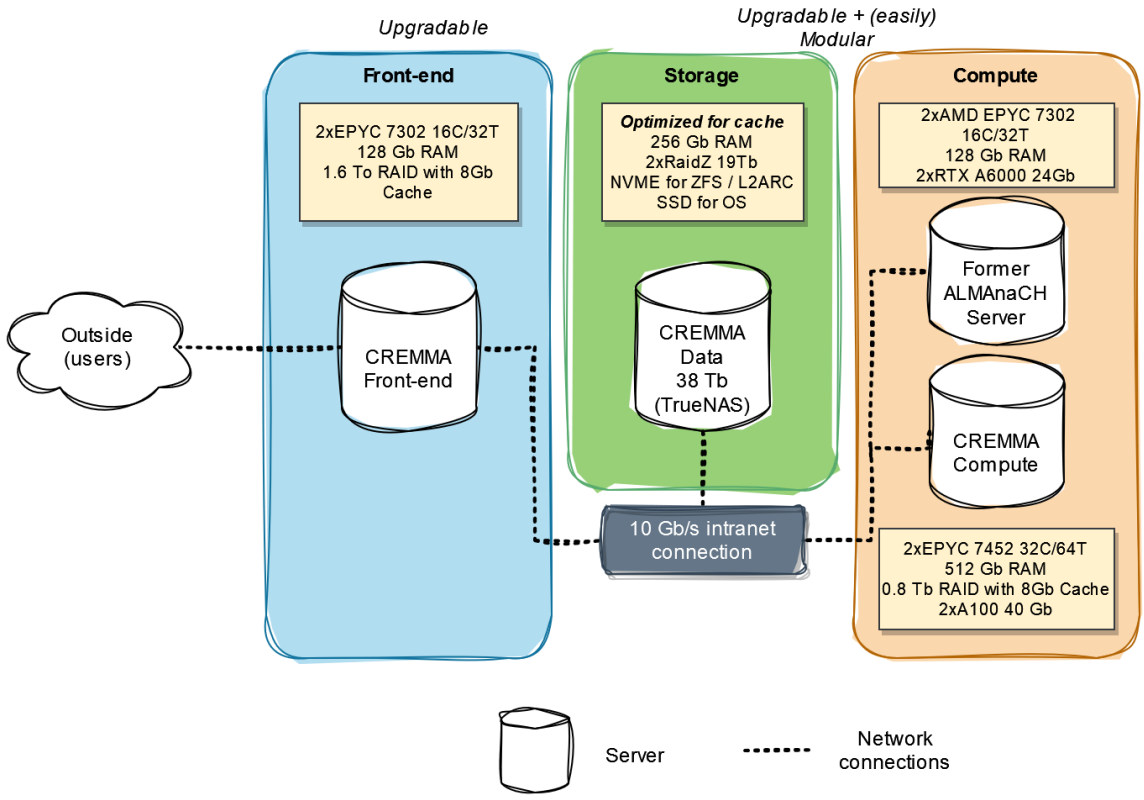
Not overlooking the RAM is very important when designing your server! But what is it used for? It's used for cache by the web application: it means that frequently accessed data, like web pages and images but also the content of the database, are temporarily loaded in live memory. Cache thus ensures that the requests sent by the users are served quickly. For example, if you don't have enough RAM (or enough cache), pages will load slowly, and if you have used eScriptorium before reading this post, you know how important it is to be able to load images fast enough.

RAM is also essential for inference and training because images and annotations are loaded in memory before being passed to the CPU or the GPU. If the RAM is not powerful enough, it will be detrimental to computation and will cause a bottleneck situation. Thus having invested in GPUs and/or CPUs but not in enough RAM would be like having a horse to pull a Ferrari: even if prediction and training could go fast on the processing units, it will be restrained by the available live memory.

Modularity for the CREMMA infrastructure

The CREMMA infrastructure was originally designed by Thibault with a simple but essential principle in mind: modularity. Instead of thinking of an eScriptorium server as a monolithic block of hardware designed for front-end service, storage and intense computation, he suggested to break each of these blocks into individual servers connected together. CREMMA[4] is thus made of at least three servers, as shown in the schema below:

- `CREMMA_FRONTEND`, for the front-end, where the application is deployed and where the database is stored.
- `CREMMA_STORAGE`, for storage, where all the images and models, as well as the backup of the database are stored on the long term. Currently, `CREMMA_STORAGE` has a storage capacity of 38Tb[5] but we could easily add more disks if we find that it is necessary.
- `CREMMA_COMPUTE`, where the two A100 GPUs I mentioned earlier are plugged and where the application task manager "sends" all the jobs, whether they are to be run on CPU (these tasks include segmentation and transcription prediction for example), or on GPU (training for the most part).



As you can see on the schema, there will actually be a fourth server involved in the infrastructure: `Traces-6`, the server we currently use to deploy eScriptorium at Inria. Like `CREMMA_COMPUTE`, `Traces-6` can be called by `CREMMA_FRONTEND` for computation tasks. In fact, this is where the modularity of the system is interesting: with such a set-up, it is possible to add more computation servers to the pool of GPUs reachable by `CREMMA_FRONTEND` without having to redesign the whole infrastructure. On their side, `CREMMA_FRONTEND` and `CREMMA_STORAGE` can be upgraded (to add more RAM or more storage) very easily.

This modularity also means that the GPUs remain free for other uses: for example if we were to have to run maintenances on `CREMMA_COMPUTE`, we can simply cut it from the infrastructure, and let `CREMMA_FRONTEND` interact with `Traces-6` only while we work on `CREMMA_COMPUTE`.

`CREMMA_COMPUTE` is equipped with two A100 graphic cards, and `Traces-6` with two RTX 6000. Actually, it doesn't mean that only 4 training can be happening at once. Each of these GPUs offer between 24 and 40 Gb of RAM for intense computation. It's a lot. It's so much actually that training a Kraken model at max speed would rarely use more than 40% of this processing power. Virtualization is a nice trick to "break" the GPU down into smaller virtual GPUs (or vGPUs). What is broken down is the RAM capacity. We opted for the following virtualization set up:

- Each of the A100 graphic cards and their 40Gb of RAM are turned into 1 10Gb vGPU + 5 5Gb vGPUs (since 10+5x5=35, note that we must leave 5Gb out of the equation for the virtualization).
- No virtualization is applied to Traces-6's RTX6000s.

How did we decide on these numbers? Thibault ran a series of small tests executing either `segtrain` or `train` and playing with two different parameters: the batch size[6] and the single point precision[7]. He found that for training a recognition model with a batch size of 8 and either 32 or 16 of precision, less than 5 Gb of RAM on the GPU is enough. With a batch size of 1 and a precision of 32, it's even less than 1 Gb. To train a segmentation model, less than 10Gb is enough, and this type of training is more rare. Since our goal for the infrastructure is not to maximize the speed of the training but to maximize the amount of possible parallel training jobs at decent speed, we decided that 10 vGPUs with 5Gb of RAM and 2 vGPUs with 10Gb of RAM were a good compromise. If we find that more GPU RAM is occasionally needed, we still have two times 24Gb with the RTX6000!

Should you build your own server?

We have spent all this time writing about how to build, how to spec out your server or your infrastructure, but let's talk about the elephant in the room: should you do it?

Well, it's all a matter of perspectives. We'd say it probably makes sense if:

1. You are a very big organization, you have a lot of money available to you, a super-cluster (and possibly a well staffed IT services department), and you have a high demand;
2. You are working on very sensitive data that can't be shared with the outside (*e.g.* medical reports);
3. You are geographically far away from any other existing server, and face latency issues when you connect to potential welcoming servers;
4. Servers that exist around you are reluctant to onboard you and the teams behind the request for a server of your own.

These four points are definitely valid. But we'd say that, if you are in another situation, sharing infrastructural costs probably makes way more sense. In our experience, building a server is long, tedious, require special (and rare) skills[8] and costly (in terms of human resources as well!). Setting up a working server can take a really long time. For CREMMA, we ended up outsourcing part of the installation of the new infrastructure because we realized that we did

not have the time nor the skills to set everything up ourselves. The cost of this installation by a third-party? Between 8 and 12K€, and again, a little time and bandwidth on our end.

Next you have the maintenance fees. You can outsource them, for a little bill from a company which would make sure that everything is installed on time, that updates work well, etc. Or you can do the maintenance yourself. But again, this comes with a cost: human time. A worker on the server goes down? You are in for a few hours. Some people crashed a third-party server by uploading too much IIIF images on your instance of eScriptorium? Well, then you will not only receive emails from these third parties (and this is completely normal), but also have to deal with your user base doing things that eScriptorium allows and that you may not (yet) be able to control/limit.

In the end, we would definitely recommend that, when this is possible, you first consider joining existing servers, including by offering *quid pro quo* by:

1. Participating in covering the salary of people maintaining the server (through some kind of yearly fees for example);
2. Providing some money to expand the existing infrastructure (to increase storage or computation, etc);
3. In general, helping eScriptorium grow, discussing with the owners of the server you are joining and/or the eScriptorium team about what kind of new functionality should be added, and if you can contribute to fund these updates.

This final point is super important: sure, owning your own server sounds appealing, even if it is costly to put in place. However, developing eScriptorium also comes with expenses. Thus, participating in eScriptorium directly -- we think -- is also very beneficial and welcome by the developing team. Open-source is free to use, free of charge but is not appearing out of thin air: developing costs money. And the more people participate in infrastructural costs (servers or software), the better the experience will be.

---

1. If you don't know anything about local servers and are curious to learn more, you can check this page: https://www.freecodecamp.org/news/what-is-localhost/. Or you can also take a look at the corresponding entry in Wikipedia! ↩
2. The full reference is: Jean-Baptiste Camps, Nicolas Baumard, Pierre-Carl Langlais, Olivier Morin, Thibault Clérice, et al.. Make Love or War? Monitoring the Thematic Evolution of Medieval French Narratives. Computational Humanities Research (CHR 2023), Dec 2023, Paris, France. ⟨hal-04250657⟩ ↩
3. By temporary, we don't mean that the image file are stored for a few hours only, on the contrary, they can stay on the disk for many years. We mean that it should be ok to consider that they can be erased whenever a user is done working on a corpus and has moved away from the transcription phase. ↩
4. From now on, "CREMMA" means the server created through the CREMMA project. ↩
5. Safety first! We have 38 Tb available, but there is actually a little more physically because we have redundancy and spare. We have 2 series of disks working with redundancy (RaidZ). In each series two disks are entirely dedicated to redundancy only, and one more is completely unused until something fails (it is used as a safety spare disk). While `CREMMA_STORAGE`, as we said before, is not used as a permanent storage solution, it needs to be a little bit safe for the user base. ↩

6. To understand what the batch size corresponds to and why it is important, you can check this entry in the Stack Exchange forum: https://stats.stackexchange.com/questions/153531/what-is-batch-size-in-neural-network. ↩
7. To quote Kraken's documentation: "When using an Nvidia GPU, set the --precision option to 16 to use automatic mixed precision (AMP). This can provide significant speedup without any loss in accuracy." Kraken's default value for precision is 32. ↩
8. It can be difficult to justify hiring a full-time or even part-time system administrator for a team because it is a very specialized and highly demanded type of profile. For example, public organizations can rarely offer competitive salaries compared to the private sector. In addition, the workload for administrating a web server can be irregular, and it can be difficult to make the skills for system administration meet with other needs faced by a team, complicating even more offering a meaningful full-time job. ↩