Web Based Notebooks for Teaching, an Experience at Universidad de Zaragoza

Miguel Angel Marco Buzunariz(✉)

Universidad de Zaragoza-IUMA, 50003 Zaragoza, Spain
mmarco@unizar.es

Abstract. Since 2012, web based notebooks have been used as interface for computer algebra systems in teaching mathematics courses at Universidad de Zaragoza. We present an overview of the experience, detailing the advantages and problems that have been noticed during this time.

1 Introduction

Computer algebra systems, both proprietary and free, have been used as part of some mathematics related courses since several decades ago. In particular, the Universidad de Zaragoza used to have campus licenses for Mathematica, Maple and Matlab. Some professors were not happy with the use of proprietary software, and opted for free systems such as CoCoA [1] or GAP [2].

However, the user interfaces of these systems used to be less friendly to the user than the ones of the mentioned proprietary systems, which constituted a problem for the adoption. Moreover, many of the free systems were oriented to a very specific area of mathematics (e.g. CoCoA is focused specifically in computations in polynomial rings, GAP is focused in group theory and so on).

By 2010, SageMath [6] had reached a level of maturity that made its use for these tasks viable. It was feature rich enough for the courses, and it had a user interface that was much more user friendly than the ones of other free systems. An aspect that was very unusual at that time was that this user interface was web-based (that is pretty common nowadays, but at that point SageMath was a pioneer). But the main drawback was that it couldn't be installed over Windows operating systems, which is what most students used in their personal computers.

We then decided to take advantage of the fact that the user interface was web based to use a server-centered approach, setting up a server that run SageMath's web interface, students could log in and use SageMath without installing any software in their computers.

This approach has been used for nine years. In the 2019–2020 academic year we started a migration process from the classic SageMath notebook to a Jupyter [3] based one. In the rest of the paper we will describe the technical details of the setups we have used, and analyze their differences. We will also mention the advantages and drawbacks that we have experienced in each case.

Partially supported by MTM2016-76868-C2-2-P and Grupo "Investigación en Educación Matemática" of Gobierno de Aragón/Fondo Social Europeo.

A. M. Bigatti et al. (Eds.): ICMS 2020, LNCS 12097, pp. 386–392, 2020.
https://doi.org/10.1007/978-3-030-52200-1_38

2 The SageNB Notebook

SageMath included from the very beginning a web based notebook GUI. It included some options that were useful for our use case:

- It allowed using user accounts, so each instructor and student could log in, and keep his/her worksheets in the server.
- It allowed publishing worksheets, so everybody could see them, and logged in users could make a copy to work in them.
- It allowed to share a worksheet with other specific users, so they could both see and edit it.

Our usual workflow was the following:

1. The instructor of each course creates a worksheet with explanations, examples, and exercises; and published it.
2. The students create a copy of it, work on their copy solving the exercises, and share it with the instructor.
3. The instructor sees the work done by each student, and modifies the student's worksheet to grade it or add further comments or explanations. In some cases, they can include further work or corrections for the students to do.
4. The student does the extra corrections requested by the instructor.

This approach worked reasonably well, although there were some problems that were solved on the way. We will now mention some of them, and how they were dealt with.

The server we used couldn't handle the peaks of work. It was solved by adding another machine and using the ability of SageNB to create sessions in remote machines through ssh.

The number of published worksheets, and the list of worksheets shared to each user kept growing to the point of making it hard for the user to find a specific one. Sadly, SageNB never included a proper method to organize them with folders or tags. We partially mitigated this by restarting the database of worksheets each academic year. Besides, we kept different instances of the notebook server for different degrees. All of them run in the same server, with a proxy server that filtered and redirected connections by domain name. That is, for example, requests to `sage-mtm.unizar.es` were redirected to the port were one of the notebook servers was listening; whereas `sage-inf.unizar.es` was redirected to another port.

The problems of lack of CPU and RAM were specially usual during exams. We suspect that it was due to some students trying to sabotage the exams by overloading the server. We tried limiting the amount of RAM and CPU that each session could use, but this remained a problem.

Also, users were created manually (theoretically, SageNB allowed LDAP identification, but we were never able to make it work with our university's directory). So we had to create scripts to create the users each academic year, from a list of students registered in each course.

The tasks of installing, updating and managing the server were done mainly by one person, with another one helping in minor issues. Both admins were professors that dedicated part of their spare time for this task. Most of the time this wasn't a problem, but at some specific times they couldn't handle the issues that appeared as quickly as it would be desirable.

Overall, it was a viable option, although far from perfect.

3 JupyterHub and JupyterLab

The SageNB notebook was deprecated in SageMath version 9 (although its development was virtually abandoned long ago). Instead, SageMath had been moving since several years ago towards the Jupyter notebook. In these circumstances, we had to start migrating our infrastructure to this new model. However, we kept the old SageNB server running for one more year to allow a grace period for the instructors that had a hard time making the switch.

When we started considering the options to switch to, our desired features were the following:

1. It should allow different users to log in, ideally using the university's directory.
2. It should allow users to organize their worksheets in folders or similar.
3. It should allow a persistent storage of the work of each user.
4. It should allow users to share worksheets with other users, at least in a similar fashion than the workflow we had in SageNB.
5. It should be able to scale to many users.
6. It should be able to isolate the computations of each users, in such a way that one user cannot exhaust the available resources.

The obvious answer for several of the requirement was to use the Jupyter ecosystem.

The Jupyter notebook [3] is a web based application that allows to combine text, executable code and graphics in the same document. The code can be executed interactively. It uses a specific protocol to communicate with the different kernels, which are programs that execute the corresponding code and return the result. That way, one can run sessions of different programming languages (the name was chosen as a combination of Julia, Python and R, but many more kernels have been added since then). JupyterLab is a redesign of Jupyter, including a desktop-like environment, with a file browser, an embedded tiling window (and tabs) manager for notebooks, interactive consoles and file editors.

Both Jupyter and JupyterLab are single-user applications, but there is a front-end that can handle Jupyter sessions for several users called JupyterHub [4]. It has different modules for authenticating users and spawning sessions. In the authentication side, we used the CAS authenticator, that worked seamlessly with our university's Single Sign On system.

As for the spawner choice, one of the most popular solutions to accomplish the isolation requirement is to use Docker. However, the Docker approach is hard to mix with the requirement of having persistent storage for each user. So

we opted for the Systemd spawner that runs each session as the corresponding system's user, under a Systemd container that allows to limit the CPU and RAM available.

To better handle the requirement of organizing the files in folders, we opted for JupyterLab instead of plain Jupyter (since it allows, for example, moving files to folders by just dragging them). Moreover, the Jupyter project has already stated that their long term plan is to move to JupyterLab and deprecate plain Jupyter.

As for the requirement to share worksheets with other users, we looked into nbgrader [5], which is a tool designed for this specific purpose, but it wasn't a good fit since it assumes that the user logs in a session that is dedicated to each course; whereas we envisioned a global system for all the university, where each student could log in and have access to his/her files related to all courses. We started working on a JupyterLab extension that would allow instructors to send files to the students enrolled in a certain course, and students to send them back to instructors. However, that work is no ready yet, so for the moment we are using external channels (Moodle and email) to send files back and forth. Luckily, the JupyterLab interface makes uploading and downloading files easier than the SageNB one.

To achieve the scalability, we contacted a research institute in our university that provides cloud services. They were kind to provide us with some virtual machines to make a test deployment. In those machines, we deployed the following design:

- A HAProxy server acts as a web frontal and https terminator. It redirects each http session to one of the computation nodes.
- A database node provides a persistent NFS volume to the computing nodes, and also a user database to make sure that usernames and uid's are kept in sync in all computing nodes. We also planned to keep here the database of teachers/students/courses for the extension that would allow to share files between users, but it is not ready yet.
- In each computing node, JupyterHub and JupyterLab are installed. Each request is authenticated by the CASauthenticator against the university's SSO. Then the system user is created (if it doesn't exist already) and the session is started under a Systemd container.

As a backup option in case of failures, a single instance was also installed in a regular computer (so it is not scalable).

The work of making the systems design, installing and configuring the software was done by a student as part of his degree thesis, together with the professor that managed the older server. This professor has also been the one that has worked in the JupyterLab extension to send files back and forth between students and instructors. Both that and the task of administering the server and deal with the problems that have arisen have proven to be too much load for the time he can dedicate to this task, which is the main reason why the extension is still in early development phase.

This approach has worked during the current academic year, where we have had troubles with the peak capacity. A limit of 1 GB RAM per user seemed insufficient for some tasks (partly due to the fact that most of them didn't properly shut down the worksheets that they don't use anymore, the UI allows to do so, but it is not evident, and most assume that just closing the tab is enough). However, if we raised that limit to 1.5 GB or 2 GB, we encountered that the whole computing node got its RAM exhausted, and didn't respond until the service was restarted. This contrasts with the peaks that the SageNB server was able to handle, which, we suspect, shows that the Jupyter server introduces a non-negligible overhead.

4 Impact on Teaching and Learning

We started with a test experience with only three instructors on the 2011–2012 academic year. After that, it was offered to all instructors that wanted to use it in their courses. The adoption was modest the first year, but then it growed quickly and has been stable since then. At the current academic year, the JupyterLab servers were used by 24 instructors and 442 students. Most of them used SageMath notebooks, but some used other ones, such as Python and R.

The typical way to use it was during problem/exercises sessions in the computer lab: the students got a notebook document that combined the theoretical explanations, code examples, and the questions they should answer by running the corresponding computations (either by using the CAS as a calculator or writing some actual code to solve the problems). Maybe the instructor could combine it with a general explanation, and/or give specific hints to the students that got stuck at some point.

As we can see, adoption increased quickly when the first notebook was introduced; and then it has remained stable. The introduction of the Jupyter environment resulted in some instructors migrating to it, while others preferred to keep using the legacy one (under the warning that it would be eventually deprecated). An undesired result of this way of switching is that some students are forced to use the legacy system for some courses, and the new one for others.

Although we didn't make a general survey about the satisfaction of students, we did receive some comments (that should only be taken as anecdotal). Some of those comments were:

- Some engineering students felt overwhelmed by having to learn one programming language (C++) for their programming courses, and a different one (Sage/Python) for their mathematics courses.
- Several complained about the punctual availability and stability problems of the server.
- Several students of science degrees without a big mathematical content (Optics, Biotech...) had a very hard time grasping the general ideas of a programming language.

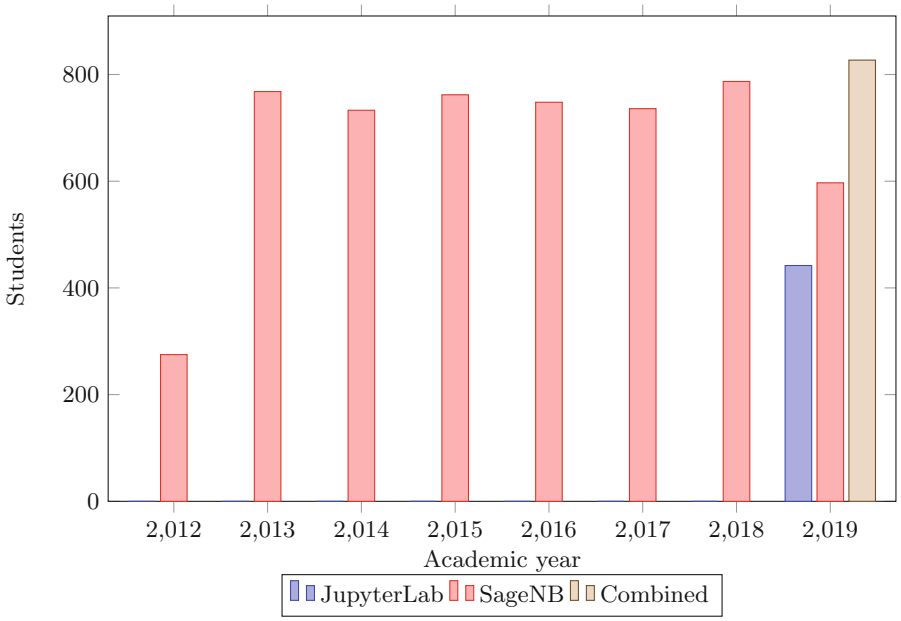

Fig. 1. Number of students that used the notebooks each year.

– In general, students of mathematics and physics degree showed a more positive attitude towards these tools. Some of them used these tools also as an assistance for doing exercises in courses where it was not formally used (Fig. 1).

During the confinement declared in the COVID-19 pandemic, the university switched to an on-line model with virtually no time to redesign the courses and workflows. It was very hard in general, and some aspects could not really be adapted. However, in particular, the computer lab lessons were one of the aspects that was more easily adapted, since it didn't require the students to install any software on their computers, and the instructors could use the same exercises and documents that were planned.

5 Conclusions

The combination of JupyterHub/JupyterLab has the potential to be a very useful tool in teaching, and maybe some day it might be considered as a standard service provided by universities (just like email or virtual campus). Its modularity and flexibility makes it adaptable to each institution's needs. However, most of the development, modules and documentation seems to be focused to different use cases, which makes it non obvious how to proceed for the university wide approach.

To find a setup that fits our requirements is still a challenge. Significant computing resources would be necessary to provide this kind of services at a whole university level. It would also require dedicated personnel to install and maintain it. An estimate of this requirements would be:

- A minimum of 2 GB of RAM per expected simultaneous user at peak time. A conservative security threshold should be added to that to prevent problems in case of an unexpectedly high peak.
- CPU doesn't seem to be a big limitation, unless the students are expected to do very CPU intensive computations. And even in that case, the only expected problem is that they would take longer.
- About personnel requirements, the initial installation and setup could be done by a small group (one or two persons) with the appropriate skills and documentation, within a few days. Proper maintenance would require a sysadmin with knowledge of the system (although, as many sysadmin work, the difference in workload between normal days and peak times would be very big).

References

1. Abbott, J., Bigatti, A.M.: CoCoA: a system for doing Computations in Commutative Algebra. http://cocoa.dima.unige.it
2. The GAP Group, GAP - Groups, Algorithms, and Programming, Version 4.11.0; 2020. https://www.gap-system.org
3. Project Jupyter. https://Jupyter.org/
4. JupyterHub. https://JupyterHub.readthedocs.io/en/stable/
5. nbgrader. https://nbgrader.readthedocs.io
6. The Sage Developers, SageMath, the Sage Mathematics Software System (Version 9.0) (2020). http://www.sagemath.org