Kevin Chen
Edinburgh

HPC Mapping

A Nuffield Research Placements & Experience Project

Introduction

In the world of high performance computing and parallel computing there are very few public resources on details of HPCs (High Performance Computers) available to researchers. My placement project was to create a resource to map and help researchers find more information on HPCs in the UK. This involved creating and designing a website to effectively convey information a researcher might want to know about HPCs, where it's located, and narrow results down with filters. A database for the data must also be designed and filled with sufficient data.

I knew a lot about computer hardware prior to the placement however very little about specifics relating to HPCs. Such as the structure of nodes and interconnects between nodes for communication, as this is only required by large scale computers.

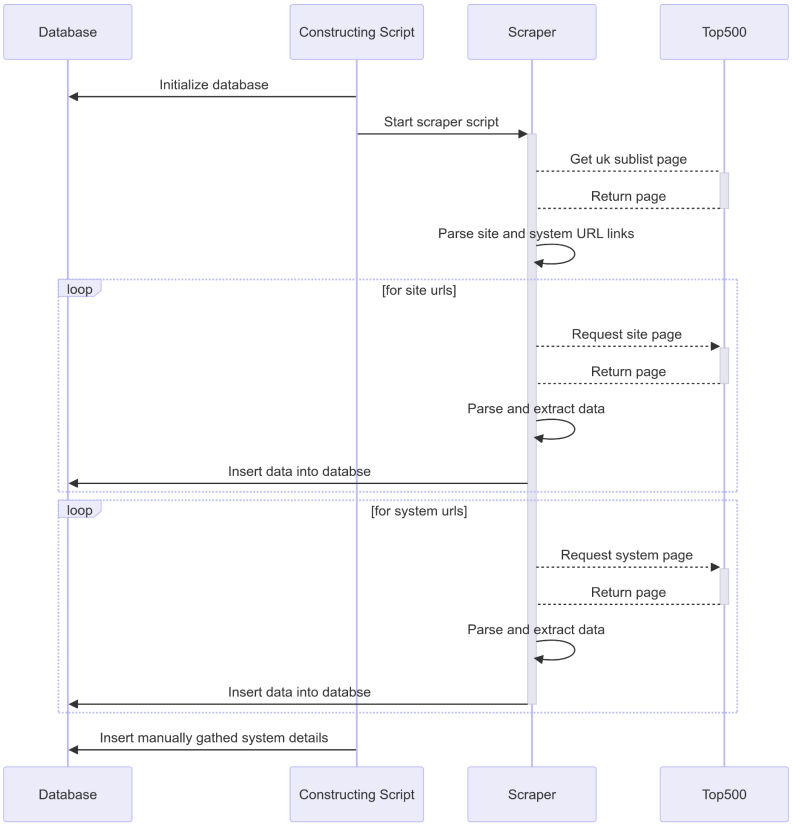
In the first few days of my placement I focused on learning more about HPCs and parallel computing in order to understand what a researcher might need to know and search for. I followed a course supplied by EPCC on both topics and also read a variety of different online resources. I also went to the Advanced Computing Facility which hosts all of the University of Edinburgh's HPCs, such as ARCHER2 and DiRAC Tursa. There I had the opportunity to talk to people that work there and had a look at the hardware of the computers. It was quite easy to find information on parallel computing due to the abundance of learning resources but very difficult to find any information on specifics on hardware.

Methodology

Backend

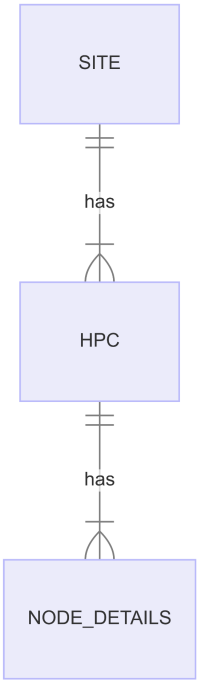
The main source of data was Top500, as it was the only list of HPCs that I could find, with information on its hardware and computing capabilities. This means I would need to get a large amount of data from the website, however a lot of this data existed on different web pages. Conveniently, Top500 provides a 'Sublist generator' to filter for UK HPCs and provides basic information on the HPCs and the sites they were on. Instead of requesting the page and submitting the filters programmatically, I manually filtered for the UK and saved the results into a file. I did this because the website required a 'keep-alive' connection to request for its filter submission and I was new to the libraries that I used, so I chose the easier route. Further information about the computer and the sites were gathered from scraping the sites that linked from the initial sublist page.

Due to the constant changes I make to the database during the development I created a script to rebuild the database and insert the data back in. The figure below shows how the constructing script and the scraper works by fetching data from top500, parsing it then inserting it into the database.



The scraper uses the python ‘requests’ package to fetch the HTML from the Top500 servers and ‘pyquery’ to parse the HTML and be able to pull out data from the website. The initial page with the list of UK HPCs was first scraped and a list of number pairs were extracted. These number pairs are the IDs that Top500 use for their systems and its respective site. I ended up using these as the primary identifiers for my own database for ease. The list of site and system ids were then used to find their specific page with all their details to be parsed and extracted.

To store all of the data gathered I chose SQLite 3 because it is simple and it is included in the python standard library, although it may struggle with high user counts and as the database grows.



HPC	
string	name
int	system_id
int	site_id
int	top500_rank
string	manufacturer
int	total_cores
int	total_nodes
string	processor_name
string	interconnect
int	installation_year
string	os
float	r_max
float	r_peak
int	n_max
str	website
int	system_tier
str	system_status
str	system_type
str	additional_info

SITE	
int	site_id
str	name
str	website
str	segment
str	city
str	country
float	longitude
float	latitude

NODE_DETAILS	
int	system_id
int	number
str	processor_name
int	processor_cores
str	accelerator
int	memory

Planning the structure of the database was one of the first things I did as it helps me understand the scopes and requirements of the project. In the design shown above, the fields of all the tables are listed with their types and the relationship between tables. The relationship between Site-HPC and HPC-Node details are both one to many. The table contains primary and foreign keys as it is a relational database, which is the structure that best fits my use case.

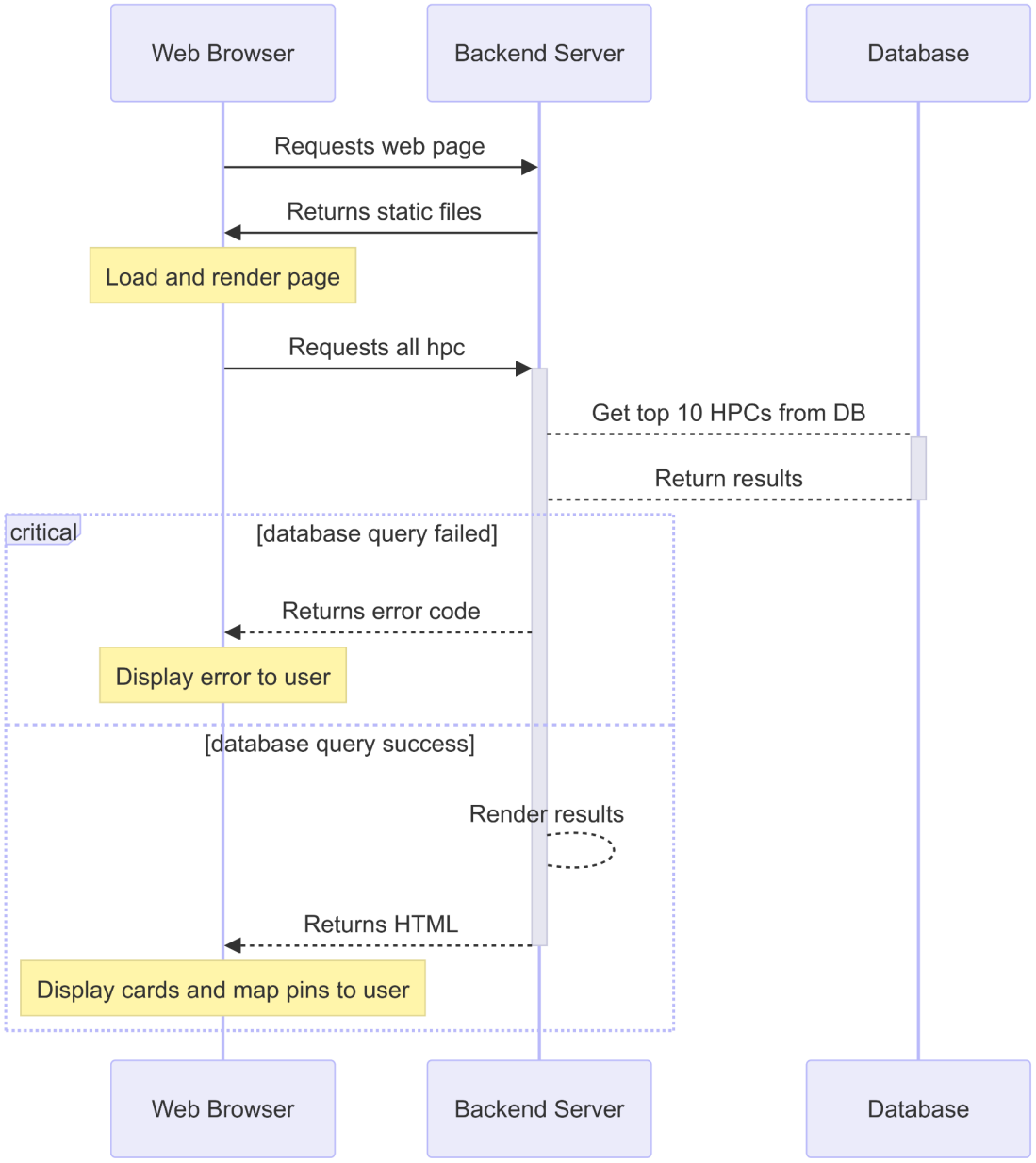
Frontend

The front end is just a static HTML website and the backend server is in python. This is due to the limitations of my laptop at the start of the placement. It was extremely slow and could barely manage to run chrome and a text editor at the same time. I managed to get a different laptop for the rest of the placement which sped up the development but I stuck with the framework choices as plain HTML and JS are still very good options. They are very fast to start development as they don't require any external tooling which means iterations are extremely fast. The backend being written in python was a good idea in hindsight, as it is a very easy language to learn and is popular, so the development was intuitive and solutions to errors were easily found online.

The website consists of a grid of cards containing information about the filtered HPCs and a map that has pins that show where each HPC is located. This is an efficient way of communicating the computer's details but also where it is located. A modal that containers further information can also open with a button on the card and contains links to find further information on the HPC. I also chose the website to be relatively simple and minimalist which meant it was easy to navigate and intuitive to use.

The frontend uses AJAX to fetch and update information on the page from an API endpoint. This is done instead of having all information on the client side, for example in a JSON file. The advantages of querying a database over reading a JSON file is that for every user, the server will need to send a large file of all the HPCs to the user which requires a lot of bandwidth, where as if the filter is applied the large data set can remain on the server and smaller paginated data could be sent to the user.

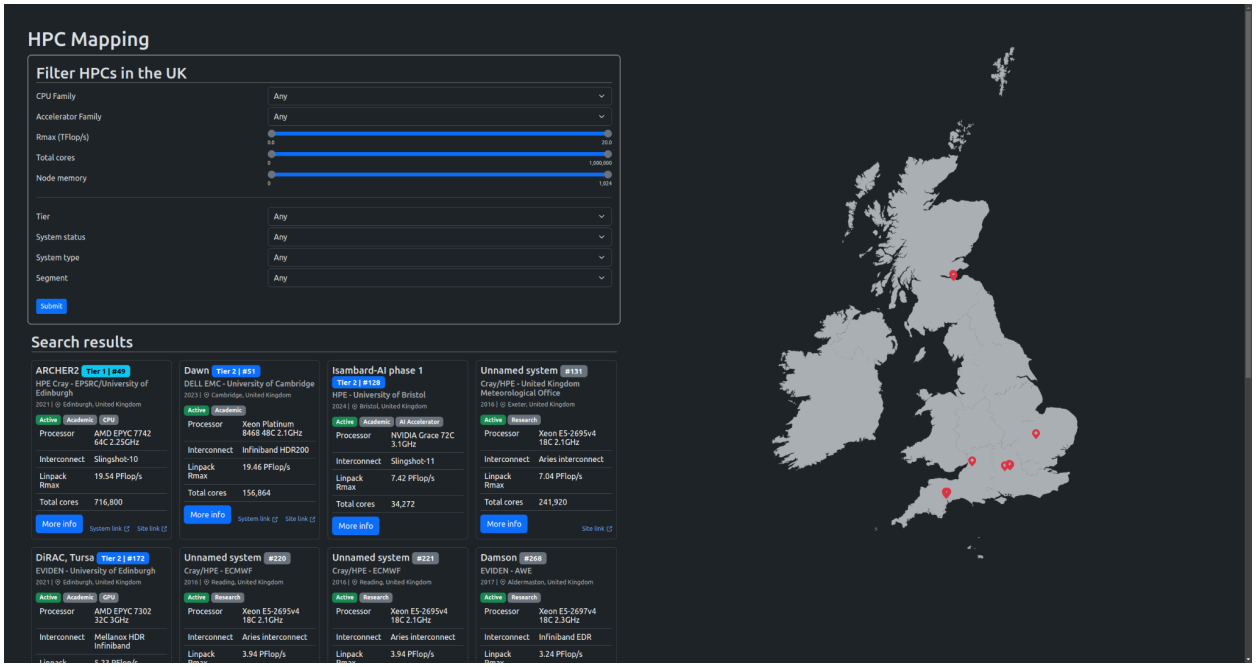
When the filter form has been submitted, the data is sent to the backend server to query the database, the information returned from the database is then rendered into HTML on the backend before being sent to the client. This isn't ideal but it is the easiest solution given the libraries being used.

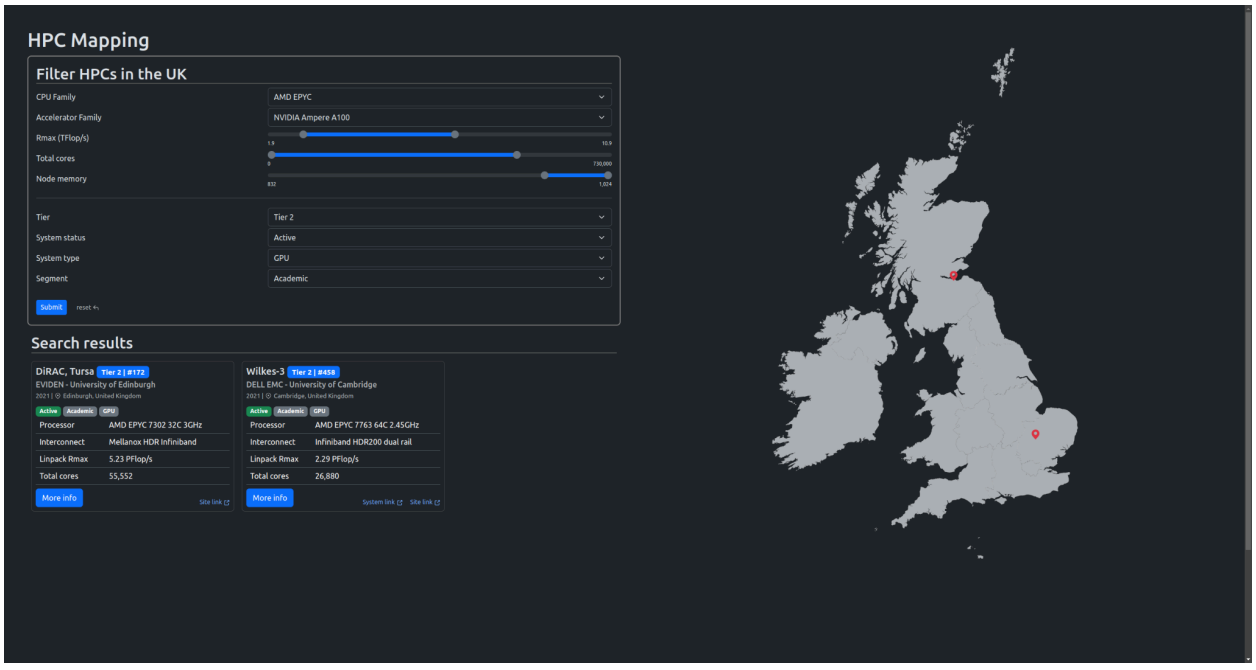


Above is the sequence diagram of the process of loading the webpage and fetching the HPCs from the backend+database. The diagram also shows the 2 possible results from the backend server to the web page. If the database fails then the error is shown to the user who could debug the issue, or the query is successful, then turns the data from the database into HTML to show to the user.

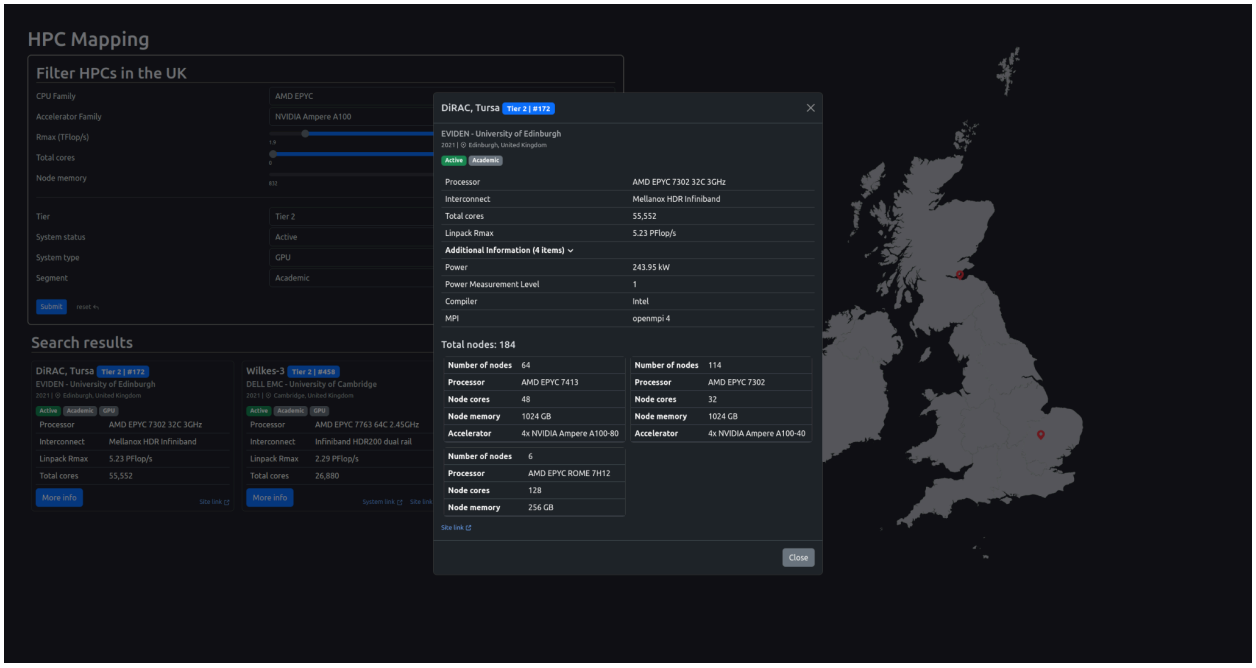
Results

Web interface





In the photos you can see the filters being used to narrow down results from the database. This allows researchers to be able to find the HPC or site most fitting to what they need.



Above is the modal which shows additional information scraped from Top500 and additional information about the nodes such as memory and accelerators.

Data

The database resulted in 16 HPCs across the UK with some of them with additional information about their nodes and accelerators, it was some work getting the data for this project but still nowhere near enough for a proper resource for researchers, more on this will be discussed in the evaluation section.

The source code can be found on my GitHub: link

Evaluation

The produced results meet all of the goals set at the start of the project, however there are a few drawbacks to the produced results.

- There are currently only 16 HPCs in the database, this is due to the fact the only source of HPCs is from the top500 website. Limiting the HPCs to only the top 500 in the world will leave out lots of smaller but still very relevant HPCs.
- The project is limited to UK HPCs by design but the UK makes up a small percentage of the world's HPCs (3.4% on Top500). There are many resources outside the UK.
- There are not many filters on the website which might mean a researcher won't be able to find what they are looking for, this was due to the limited time I had during the placement, I have implemented most of the common/important filters.
- The website is not being publicly hosted anywhere, which means in order to use the project, some computing knowledge is required to host it yourself. Although I have made it as easy as possible for people to use by including a guide and simple scripts, this still increases the entry barrier to the resource and is less accessible to the general public.
- The code quality isn't as good as I would have liked, most of this is due to the fact I learnt how to use the frameworks and language during the placement and was not familiar with them.

If I had a better system while starting the development I would have used more capable frameworks and libraries for my project, such as django instead of flask and using react instead of vanilla HTML, CSS and JS.

Acknowledgements

Thank you to James Richings and Chris Wood at EPCC for making this placement project possible, and everyone else at EPCC for making this an enjoyable placement and creating a nice environment to work in. Also big thanks to the Research Placements & Experiences programme for the placement.

The placement has helped me learn more about computing hardware and introduced me into the world of HPCs. It has also taught me to manage my time and how to work efficiently and set achievable goals.

Data Sources

- Top500 - HPC Data
- Google Maps - SIte coordinates

Hardware details

- ARCHER 2
- Dawn
- DiRAC, Tursa
- Isambard-AI phase 1
- Baskerville
- Wilkes-3