# Progress Bar web service, for distributed computing

Many data reduction tasks involve software running in several processes on one computer, or on several processes on each of many computers. This web service allows the end user to keep track of the progress of each 'worker', by displaying a bar chart showing the live status of each worker. The workers send their status to the server with a single HTTP request - that could be a single line calling 'curl http://....' in a shell script, or a requests.get('http://...') funct ion call in a Python script. The web service collates the updates from all of the workers, and allows the users to view the status of all workers, in real time, in a web browser.

Eg:  http://ws.mwatelescope.org/progress/show?jobid=demo1

To use this service, whatever starts all the worker processes needs to use the 'create' service to define a new job, with a job ID that's unique. The job ID is an arbitrary string or number - possibilities include a random 32-bit hash or integer, a stirng made up of the observation ID plus the current Unix timestamp, etc. The job ID distinguishes this multi-process job from any other job, run by you or someone else. Job data is stored on the server for at least a week (by default).

After the progress job has been created by calling a /progress/create URL, the startup code than has to pass the job ID to all of the worker processes (along with the data that they need to process), and send the user the job ID that it chose, or the 'view plot' URL with that job ID.

Each of the worker processes then calls a '/progress/update' URL at regular intervals to send its current state to the web service. It sends the job ID that it's been given, its own 'worker ID' string (unique inside that job), the total number of 'things' that it's been told to process, and the number it's done so far. This is simply plotted as (100 * current / total), as a percentage complete. Workers can also include an arbitrary string (shown in the plot inside or next to their progress bar), and an 'ok' flag, which if equal to zero, results in their bar being plotted in red instead of green.

## The web services themselves:

The URLs for these services all have a similar format, starting with the base URL, e.g.

**Base Webservice URL**

http://ws.mwatelescope.org/

Parameters are passed in the URL - a '?' character separates the path from the first 'name=value' parameter, and '&' characters separate any subsequent 'name=value' parameters.

### create - make a new progress job.

`BASEURL/progress/create` - eg http://ws.mwatelescope.org/progress/create?jobid=myjob123&desc=Testing

Parameters are:

- **jobid**: Arbitrary string, integer or float, must be unique.
- **desc**: Description - human readable text to display as the plot title.
- **maxage**: Time in seconds after which this job might be deleted by the web server. Defaults to one week.

The create service is called once, before any workers are started. The job ID used then needs to be passed on to the worker processes, as they are started, and sent to the end user, so they can view the progress bar in a web browser. If you call the create service with a job ID that already exists in the system, the existing job will be overwritten, and any existing progress data from workers will be lost (but if those workers are still running, it will get any subsequent status updates).

### update - set the current state of a worker process.

`BASEURL/progress/update` - eg https://ws.mwatelescope.org/progress/update?jobid=myjob123&workid=1&current=5&total=22, https://ws.mwatelescope.org/progress/update?jobid=myjob123&workid=2&current=10&total=22, https://ws.mwatelescope.org/progress/update?jobid=myjob123&workid=3&current=21&total=22,

Parameters are:

- **jobid**: The job ID passed in when this worker was created.
- **workid**: Arbitrary string, integer or float defining one particular worker process - eg 'worker7', 'node32', 'flagger', etc. Must be unique within this individual job. Displayed as the plot's Y tick labels, so keep it short.
- **current**: Integer or float representing the progress of this worker so far.
- **total**: The 'end point' when this worker will have finished. The plot simply shows current/total as a percentage.
- **ok**: If this parameter is supplied, and equal to zero, then the bar plot for this worker is shown in red, to indicate a problem. The worker could send this status if it exits without finishing cleanly.
- **desc**: Arbitrary status string to display in the plot, inside or next to the bar for this worker. Keep it short, or the plot will look ugly.

The update service should be called at regular intervals by each worker, to send their current state to the web service.

### show - display an animated progress bar plot for this job.

`BASEURL/progress/show` - eg https://ws.mwatelescope.org/progress/show?jobid=myjob123

Parameters are:

- **jobid**: The job ID chosen by the worker startup code when the job was created
- **interval**: Plot update interval in seconds (defaults to 30)

This service returns a web page showing a dynamic (automatically refreshes) plot of the current state of each of the worker processes for that job.

### render - return a PNG file with the progress bar plot for this job.

`BASEURL/progress/render` - eg https://ws.mwatelescope.org/progress/render?jobid=myjob123

Parameters are:

- **jobid**: The job ID chosen by the worker startup code when the job was created

This service returns a single PNG file with the plot for a given. Useful if you want to embed the progress bar plot in another web page.

## Calling these services in your code

In a shell script, it's probably easiest to use 'curl' to call the URL - for example:

```
$ JOBID=${LOGNAME}:`date -Ins`

$ curl "http://ws.mwatelescope.org/progress/create?jobid=${JOBID}&desc=My+Job"

$ echo "View progress of this job at https://ws.mwatelescope.org/progress/show?jobid=${JOBID}"
```

That sets the JOBID variable to something unique (login name, followed by an ISO timestamp), displays the viewing URL to the user, and calls the create service. You would then pass the JOBID variable to the worker processes.

Note that if you have an spaces in your description, you need to turn them into '+' characters in the URL passed to curl, or use the --data-urlencode option in the curl command, eg:

```
$ curl -G --data-urlencode jobid="${JOBID}" --data-urlencode desc="This is my job" http://ws.mwatelescope.org/progress/create
```

If you're calling these services from Python, the easiest way is to use the requests library - for example:

```
>>> import requests

>>> requests.get('https://ws.mwatelescope.org/progress/update', params={'jobid':sys.argv[1], 'workid':'flagger1', 'current':7, 'total':50, 'desc':'lots of RFI'})
```