

## 1. Summary

The Styx Grid Service (SGS) software allows existing binary executables to be wrapped and exposed as remote services. The major advantages are:

- Remote services can be used *exactly* as if they were local executables
- Workflows can be created using simple shell scripts
- Data can be streamed directly between remote services
- Very easy to install and use (**5 minutes to install from scratch**)
- Firewall friendly: requires only **one incoming port to be open on the server** and **no incoming ports to be open on the client**
- Supports interactive use such as computational steering

## 4. Workflows are simply shell scripts

The ability to execute a Styx Grid Service just like a local command is very powerful. Because of this feature, **workflows can be created using simple shell scripts** (or batch files in Windows).

For example, let's say we have two SGSs in an environmental science application. The first (called `daily_means`) calculates the mean cloud cover for the day, given a set of files that represent snapshots of the cloud cover at several times during the day. The second service is the `makegif` service (see section 3) that turns an input file into a GIF image. See figure 1 below.

A shell script to execute this simple workflow would look like this:

```
daily_means means.nc snapshot*.nc
makegif means.nc means.gif          (1)
```

This is **exactly the same** script that would be used to perform the same task with local programs.

It is more efficient for the intermediate file (`means.nc`) to be **passed directly from one service to the other**. This can be achieved very simply with a minor change to the script:

```
daily_means means.nc snapshot*.nc --output-refs
makegif means.nc means.gif          (2)
```

The `--output-refs` argument is a signal to the client to download a *reference* (i.e. a URL) to the output file, rather than the file itself. This reference is then passed to the `makegif` service, which obtains the `means.nc` file directly from the `daily_means` service.

## 2. Why use this software?

There are many reasons for exposing programs as services:

- To run programs on powerful compute resources (e.g. a cluster)
- To run a program close to a large data store
- To run a program that is tied to a particular platform
- To allow others to use the program

## 3. Services are run just like local programs

Once deployed as Styx Grid Services, programs can be run from anywhere on the Internet from the command line.

Let's say we have exposed a program called `makegif` as a Styx Grid Service. This program takes an input data file and outputs a picture of the data in GIF format.

The `makegif` Styx Grid Service can be invoked with the command:

```
makegif <infile> <outfile>
```

This is **exactly the same command** that would be used to run the program locally. The SGS software performs the following tasks:

- Creates a new instance of the `makegif` service on the SGS server
- Uploads the input file to the server
- Runs the `makegif` service instance
- Downloads the output file that was produced

All of this is performed automatically, **removing the need for the user to manually upload and download files**.

## 5. Computational steering and visualization

Styx Grid Services can be used in a **computational steering** environment. Computational steering is when the parameters of a running simulation (e.g. the driving pressure of a fluid) are varied while the program is running. The screenshot shows a Lattice Boltzmann simulation of fluid flow (which has been exposed as an SGS) being steered from a client application.

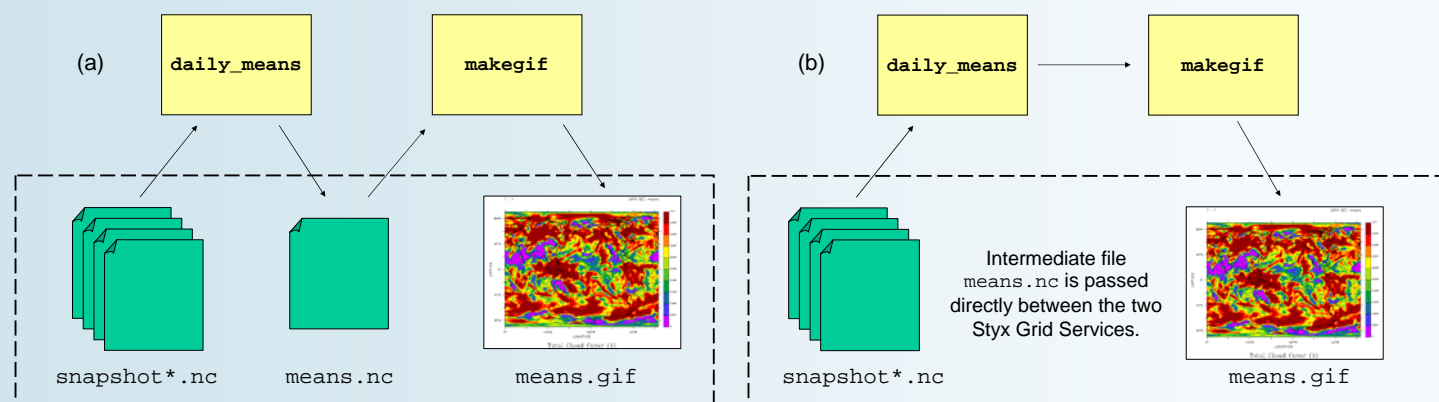
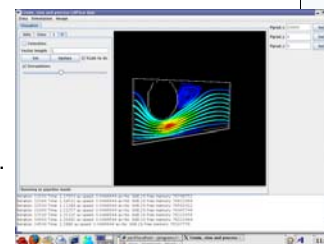


Figure 1: (a) Simple workflow involving two Styx Grid Services (yellow boxes). This workflow is executed using the script (1) in section 4 above. The dashed box represents files that exist on the client's machine.

(b) The same workflow, but this time the intermediate file (`means.nc`) is passed directly between the two Styx Grid Services, without being downloaded to the client. This workflow is executed using only a minor modification to the script: see the script marked (2) in section 4 above.