DOI: http://dx.doi.org/10.7551/978-0-262-31709-2-ch012

The ALife Zoo: cross-browser, platform-agnostic hosting of Artificial Life simulations

Simon Hickinbotham, Michael Weeks and James Austin

University of York, Computer Science Department, York, UK
simon.hickinbotham@york.ac.uk

Abstract

We describe a new approach to sharing software simulations that is of great potential benefit to Artificial Life researchers. youShare is an online collaborative facility that allows users to upload data, and software in the form of services. An attached execution environment allows services to be run over a heterogeneous cluster of compute nodes, where the service infrastructure guarantees that the service will be executed in the correct environment, and provide consistent results. It allows software to be made available as a service regardless of the operating system they run upon. This allows software to be maintained more easily, and to be available to all researchers with internet access. We demonstrate this by making three Artificial Life simulations available over the web: Tierra, Avida and Stringmol. These services form the foundation of an ALife "Zoo", in which visitors can interact with ALife simulations for research and education. In addition, youShare offers a workflow facility whereby multiple services can be connected to create more complex tasks. We demonstrate the utility of this system in Artificial Life research via a workflow which calculates evolutionary activity for runs of Tierra and Stringmol.

Introduction

It is common practice for researchers and scientists that develop their own algorithms and programs, to make them available to other researchers as source code and/or binaries. Often these are distributed to the community via personal, community or commercial websites. With a limited range of operating system (OS) configurations available to the originators of the software, it is very common that third-party developers who attempt to compile and execute these applications experience compile-time errors, dependency errors and run-time errors.

One solution is to develop and maintain source code or binaries that run on a range of operating systems. There are several problems with this approach:

- it is time consuming for the developer, who is often only experienced in writing software for personal use whilst pursuing their own research;
- considerable expertise is required to compile binaries or install software, particularly when it has not been written by an expert in writing production-quality software;
- the code base becomes increasingly unwieldy, making development by the original authors or the community more difficult.

It should also be noted that if researchers do not make their code available, it can promote the independent development of code bases that may not conform to the original algorithm specification. The lack of availability of source code, or difficulties in compiling code if it is available, have the potential to reduce the impact of the research. The onus is therefore upon the researcher to make software available, which places further burden on a finite resource, namely the researcher's time.

Researchers in Artificial Life (ALife) will be particularly familiar with these issues, not least due to the interdisciplinary nature of the field. A cross-disciplinary research team will have fewer skilled programmers than a pure computer science project, and fewer domain experts than a pure biological project. There are myriad ALife computer simulations available (for a list, see `http://en.wikipedia.org/wiki/Artificial_life#Notable_simulators`), many of which share common themes and research applications. However, research that explores ALife issues using more than one of these simulators is rare.

In this contribution, we demonstrate a system that is capable of overcoming these issues, based upon the YouShare Virtual Laboratory (`https://portal.youshare.ac.uk`)(Austin et al., 2011). YouShare is an online collaborative facility that allows users to upload data, and software in the form of services. An attached execution environment allows services to be run over a heterogeneous cluster of compute nodes, where the service infrastructure guarantees that the service will be executed in the correct environment, and provide consistent results. Of particular interest to the ALife community may be the workflow facility that allows multiple services to be connected together to create more complex scenarios. To demonstrate this we

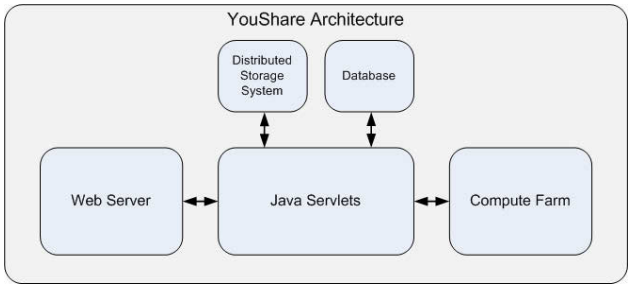

Figure 1: Overview of the YouShare Platform Architecture

have wrapped instances of a small set of representative simulators and made them publicly accessible to all YouShare users. In addition, we have created two auxiliary data analysis services that can be used on comma-delimited data files that are easily produced from the simulators.

YouShare Platform

YouShare is an internet-based Virtual Laboratory Environment (VLE) that provides a collaborative repository for data, executable software services and workflows. The YouShare platform is based upon work achieved on the CARMEN Neuroscience platform (http://www.carmen.org.uk) (Watson et al., 2007) to provide a VLE for neuroscientists. YouShare extends CARMEN to be a generic, cross-domain platform for UK academics and researchers.

The YouShare platform is based upon a standard three-tier web architecture (Eckerson, 1995), as shown in figure 1. The first-tier, also known as the presentation layer, consists of a web portal, providing the user with access to the system via a web browser. The portal is built using the Google Web Toolkit (GWT), which accommodates inconsistencies in the implementations of Javascript between all the web browsers in common use today. The application layer (tier 2) consists of a group of Java Servlets, as defined by Oracle (Oracle, 2013). These provide an API onto the Data Storage Tier, and provide access to the compute servers, which are typically 8-core zeon virtual machines with 16GB RAM. The data storage layer (tier 3) comprises an SQL type database and a storage system. The database contains user accounts, metadata, system states, and the relationships between users and artefacts (files, services, metadata, workflows, etc). The storage system is used to store user's files, service implementations, and workflows.

YouShare is accessed via the portal (https://portal.youshare.ac.uk); a login facility provides secure access for users. Data can be uploaded to the YouShare platform, along with fully descriptive metadata (Jessop et al., 2010). Once uploaded, sharing permissions can be applied to both the data and the metadata. Several levels of sharing rights are available: private to the user; shared with specific people or groups of people; or public to all registered users of YouShare.

Software applications can be deployed to the platform in the form of executable services. These services can be used to analyse private, shared or public data from the data repository. A service consists of the software implementation and a metadata description of the service. The services are executed on a heterogeneous array of compute servers attached to the YouShare platform. Services can be executed from the YouShare portal individually, or combined into a workflow to create more complex processing scenarios. The owners of YouShare services and workflows can set sharing permissions in a similar way to data. To aid discovery of suitable data, services and workflow, an Apache Lucene based search engine is provided to search each of the repositories.

YouShare Services

A YouShare Service (Weeks et al. (2013)) is a combination of software that can be executed on the YouShare platform, and a metadata document to describe it's operation and implementation. The software can be based upon a users proprietary code, or a common software package these can be converted into a service via a wrapping process using the YouShare Service Builder application, then deployed on the YouShare platform. We can currently create services from programs written in Python, Java, R, C/C++, Matlab, Perl, Bash/Bat scripts, or indeed most executables or scripts that can be run via a command line. YouShare Services also support multiple OS platforms, such as Linux and Windows.

The Service metadata document is a key component in the service. It provides the name and description of the service so that user's can select suitable services for their requirements. A description of the input parameters provides an interface on the portal when a service execution request is made. This interface provides a description for each input, and a suitable means to set the inputs, i.e. file browser, text box, or drop-down menu. Platform and environment specific information for the service execution can be set, to ensure that service is deployed onto the correct type of compute server, along with any data that needs to be staged. This ensures correct execution, and completely removes the need for user's to know the implementation details. Upon successful execution of the service, the metadata is used to stage the results back into the data repository and to provide a portal interface to display the output results.

Creating Services

One of the principles behind YouShare is that users can create their own services. To this end, the service creation process needs to be quick and easy to achieve. We also need to support multiple operating systems and application programming types, and need a common method for passing data into and out of the software. These requirements can be achieved by converting the software into a command line application. Input parameters are passed into the application

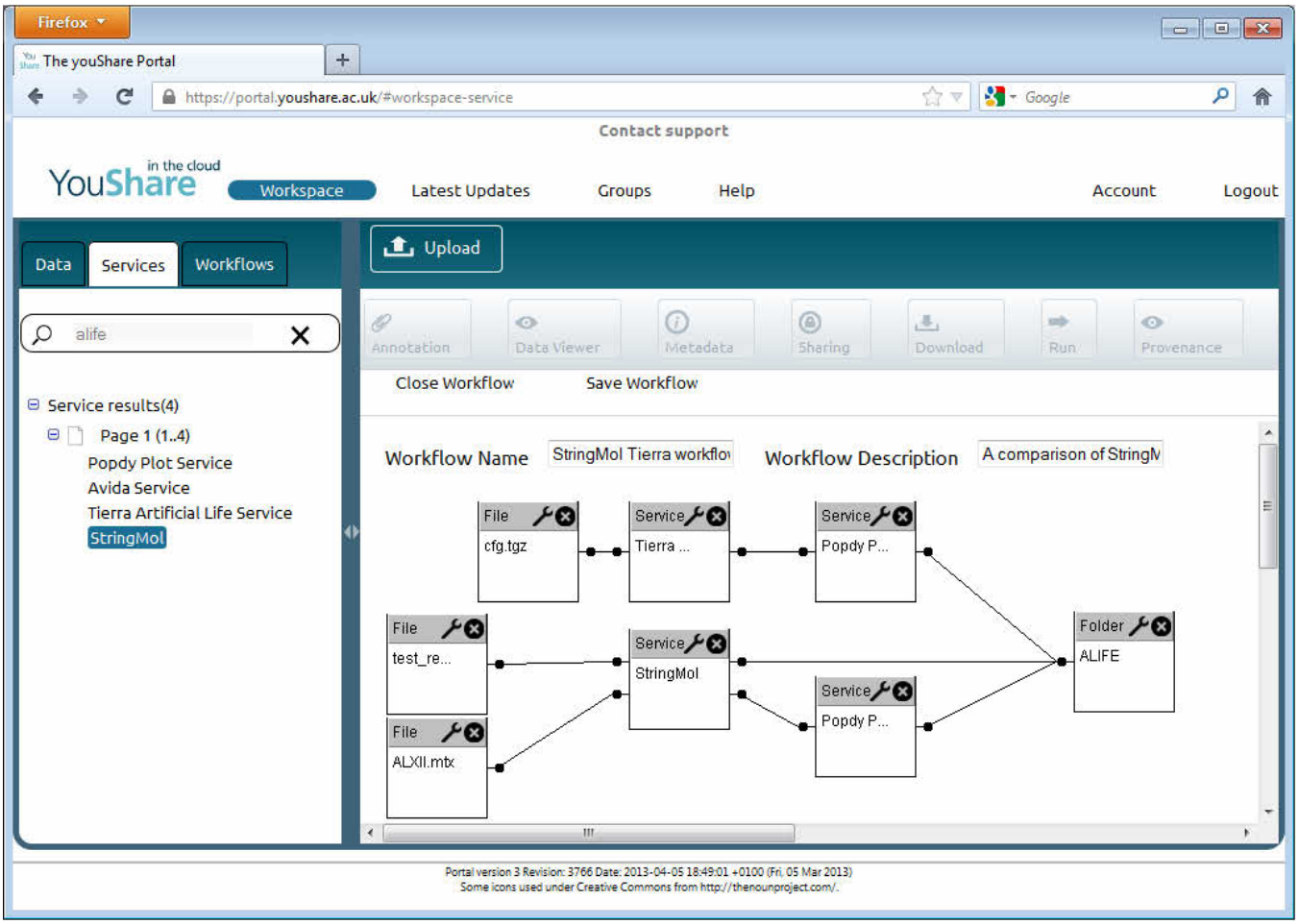

Figure 2: Example YouShare workflow consisting of Stringmol and (a modified) Tierra services being analysed by the Stringmol popdy plotting service. The results from both services are placed in the same output folder

via the command line parameter list. On completion of the application, the service needs to register its outputs in the database. This is achieved by getting the application to print any output values and/or file names to the screen (stdout) in a comma-separated list, surrounded by xml <output> tags, which can be achieved either by simple changes to the source code of the application, or by calling the application from a script which prints the output data once the application has finished.

To enable the command-line application to be interfaced with YouShare's Java Servlets (application layer), the application is encapsulated within a small Java class wrapper. A metadata document is also necessary to describe the application/service from a user and system standpoint. We have developed a tool, called the YouShare Service Builder, to automatically generate the wrapping and metadata. The Service Builder is a Java standalone application which uses a wizard-based approach using simple forms. The Service Builder generates a jar (Java archive) file containing the service implementation, and an XML file containing the metadata document. These files are uploaded to an admin panel on the portal in order to deploy the service.

Running Services

To run a service, the user must first discover it. A search facility allows users to search the service repository, and displays a list of matching services that are accessible to that user depending on the service sharing settings. The user can browse the list of returned services and select one that is suitable. The service description can be viewed, and the service bookmarked for later use, or executed by selecting the "run" button which displays the service execution panel. This panel displays the input parameters to the service as described by the service metadata, and allows the user to select suitable data inputs. Once the inputs are set, the user can select a folder in their data repository for the results to be placed, and the service execution is then initiated. Whilst running the service execution progress is displayed

in the service log, where, upon completion the results are displayed. The results can also be viewed by browsing the specified results folder in the data repository.

Workflows

Services can also be executed within the YouShare workflow facility that is available on the portal. Workflows allow more complex analyses to be performed, using a collection of suitable services, connected together in serial and/or parallel. The YouShare workflow tool is based on two components; a graphical workflow editor embedded into the portal, and a workflow enactment engine in the application layer servlets. The workflow editor allows users to place and connect input files, services, and output folders on the workflow editor panel. Once created a workflow can be saved and reloaded for later use. Modifying a workflow creates a newer version, though previous versions can be reloaded very simply. A workflow can be shared with other users, or groups, in the same way as files and services, however each component of the workflow must have a suitable sharing permission for a user to use a shared workflow. An example YouShare workflow is shown in figure 2. Once a workflow is ready for execution, it must be saved and the workflow editor closed. Selecting the "run" button, passes the workflow script to the workflow enactment engine. The enactment engine builds a model of the workflow, and orchestrates the service execution, and the flow of data between the service processes. Since a service implementation details are handled via the service API via the service metadata, the workflow can be constructed from a combination of services that each require different platforms and environments.

The ALife Zoo

Having described the youShare system, we now turn to its utilisation in ALife. Our principle aim is to create an ALife "Zoo" consisting of simulators and associated data that are available to run on a single, free resource, and to foster the development of common methods of analysis of their outputs. The first step towards this goal is to make simulators available as services. In this section, we detail how this has been achieved for Tierra, Avida and Stringmol.

Tierra as a service

The first paper on Tierra appeared in 1991 (Ray, 1991) - Tierra is close to a quarter of a century old, yet it remains highly influential. The first paper has 931 citations on google scholar. Research on Tierra is still active (Shao and Ray, 2010). Anecdotally, during the panel discussion at ECAL 2011, one of the topics was 'most influential work' – the panel unanimously credited Tierra.

Building the core service We downloaded Tierra version 6.02, with the patch from Matthias Rav (`http://tierra.lolwh.at/`), and compiled it on one of our CentOS5 64-bit Linux compute servers. Rather than modify the C source code to suit our requirements, we created a bash script to wrap the Tierra command line application and provide the XML <output> tags. This ensures that the service runs as the authors of the original software intended.

One of the limitations of our service framework is that a service must have a fixed number of inputs and outputs. This can be a problem where the software that the service is built from can be configured with varying numbers of arguments. Tierra takes a group of configuration files as it's input data, and produces a number of files in an output folder. To solve this, the bash script takes in a single zip or tgz archive of input configuration files, and un-archives them to the working folder on the execution server. Similarly, the output folder is archived by the bash script to produce a single output tgz file.

When Tierra fails it does so in an interactive way, asking the user for input. As this will be executing on a remote server, Tierra would not get a user response and so the service would hang, eventually timing out. Ideally, we want the Tierra service to terminate on error, so we modified the Tierra source code by commenting out the interactive code. To provide additional robustness, the bash script performs some degree of pre-checking to see if the configuration files are complete. The Tierra command line and it's calling bash script were then wrapped using the Service Builder tool, and the resulting service was deployed on YouShare with public sharing access.

Avida as a service

Unlike Tierra, Avida (Adami and Brown, 1994; Fortuna et al., 2013) assigns every digital organism its own protected region of memory, and executes it with a separate virtual CPU. By default, other digital organisms cannot access this memory space, neither for reading nor for writing, and cannot execute code that is not in their own memory space. A second major difference is that the virtual CPUs of different organisms can run at different speeds, such that one organism executes, for example, twice as many instructions in the same time interval as another organism. The speed at which a virtual CPU runs is determined by a number of factors, but most importantly, by the tasks that the organism performs: logical computations that the organisms can carry out to reap extra CPU speed as bonus.

Building the Avida service Avida is very similar in operation to Tierra, in that there are multiple input configuration files and multiple output files. The Avida C++ source code was compiled for our CentOS5 Linux environment, and a similar bash script was used to encapsulate the Avida command line executable. This script is used to pass the configuration files into the service as a single input archive file, and the output files as a single archive file. The bash script also performed some degree of checking for correctness on

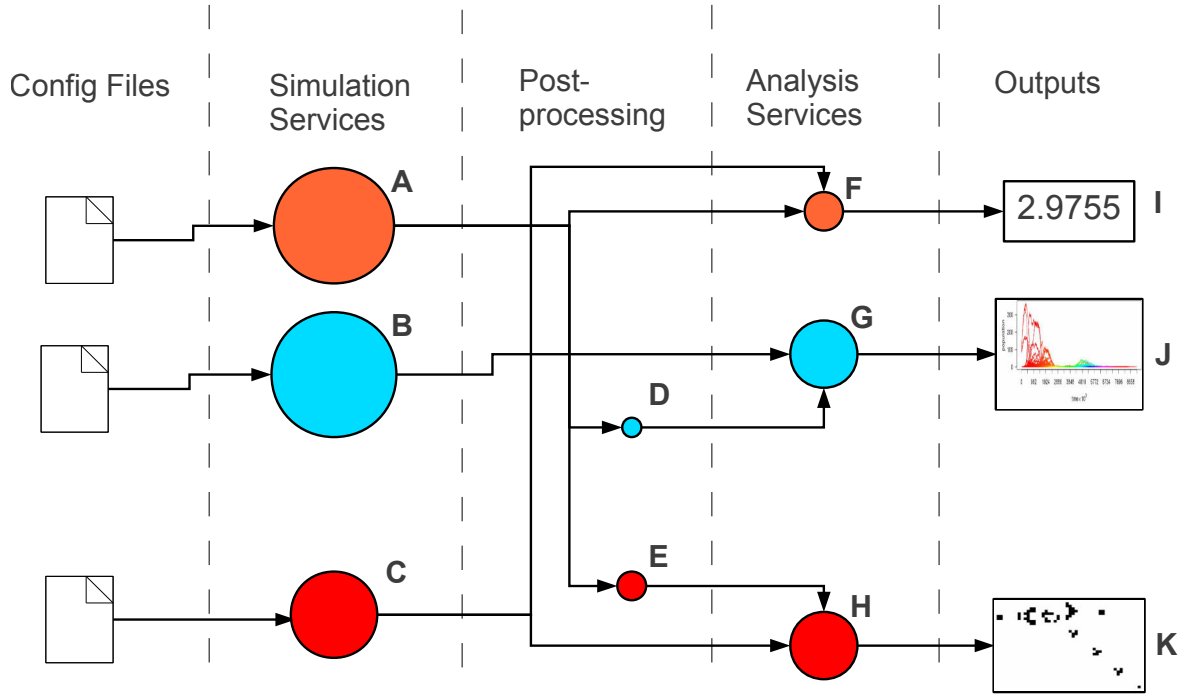

Figure 3: Hypothetical illustration of the set of possible workflows using services developed by three research groups, shown in Orange, Blue and Red. Each service is represented by a circle. The size of the circle indicates the effort involved to develop the original software (effort to wrap the software into a service is negligible). An individual workflow is constructed by connecting services together following arrows from left to right. Services developed by different teams can be linked together provided they are compatible. For example, service A can only be linked to service G if it is passed through post processing service D.

the input configuration files. We wrapped the bash script and Avida version 2.12.4 (Anon., 2012a) into a service and deployed them on YouShare with public sharing rights.

Stringmol as a service

Stringmol is a more recent simulator which uses a much simpler model of the individual (Hickinbotham et al., 2010, 2012b). In Stringmol, there is no virtual CPU and all operations are carried out on the sequence of the individual, rather than storing values in a stack. Despite the simplicity of the implementation, Stringmol is capable of generating suprisingly innovative programs.

Service implementation Stingmol has not had the extensive software development that Tierra and Avida have undergone and so was much more straightforward to implement. Stringmol is written in the C++ programming language, and we compiled version 0.2.1 for our CentOS5 Linux servers, and wrapped it as a service. Two input files must be specified, plus a numerical selection of the type of simulation that is to be executed (bi-molecular interaction, simulation of a single container of molecules, or simulation of a population of containers of molecules). However, the number of output files that are created depends on the configuration of the simulation. A script was used to collect these output files into a single zipped file that forms the output of the service.

Analysis services

Although the three simulators described above run as standalone programs, each of them have auxiliary programs that are used to analyse the outputs of the simulation in order to carry out research on their behaviour. The analysis program need not be written in the same programming language as the simulator, nor need it run on the same operating system.

To illustrate how these analyses can be coupled to simulators in a workflow, we have deployed two analysis services, written in the R language. These services were originally written to analyse the outputs of the Stringmol simulator, but we have adapted Tierra to produce output that can also be analysed by these services. This approach allows analysis tools to be developed which can be used to evaluate and compare the different simulators. The first analysis service produces graphs of population dynamics of different species

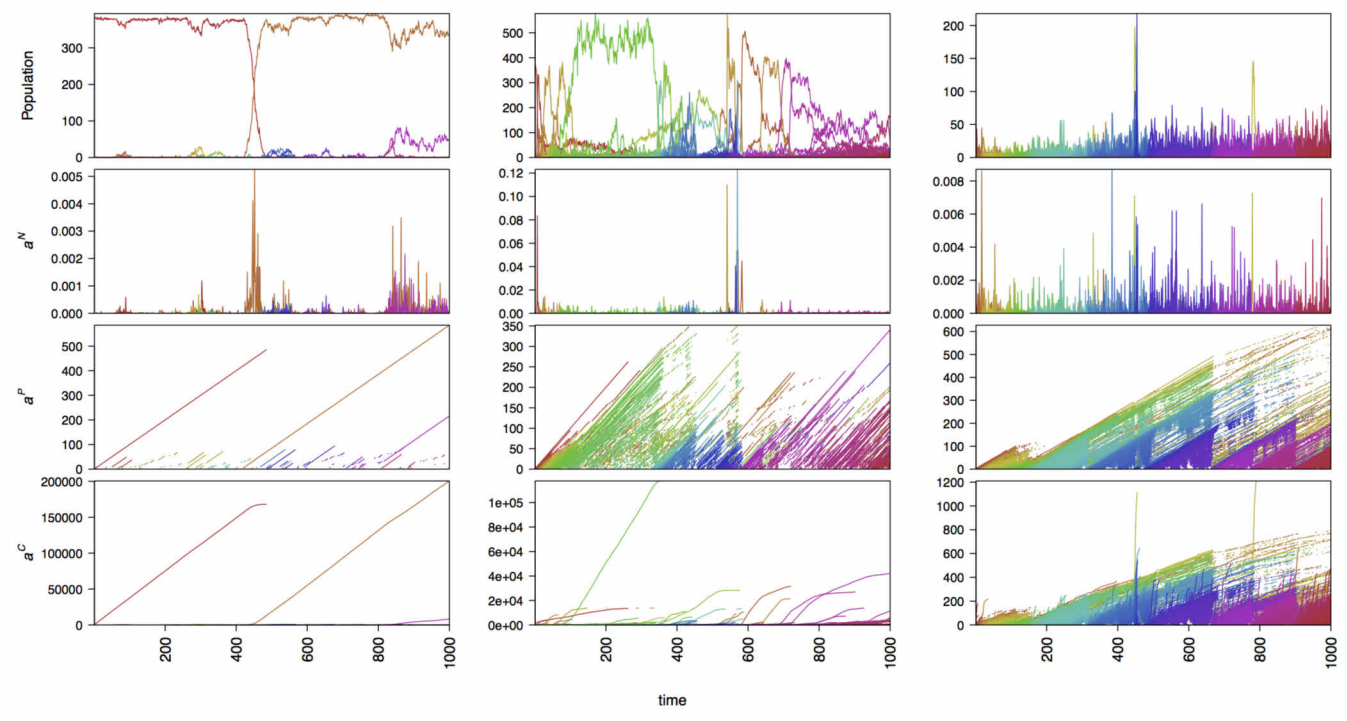

Figure 4: A plot of population dynamics (top) and non-neutral, population- and count-based measures of evolutionary activity, as executed in the workflow in figure 2. See Droop and Hickinbotham (2012) for further details

in the simulator, and the second produces a measure of evolutionary activity (Droop and Hickinbotham, 2012).

ALife Workflows

Although the provision of software services is a useful aim in itself, the real power of the YouShare system arises when services can be connected together to form workflows. This allows research groups to collaborate on developing common research themes with relatively little effort. Figure 3 illustrates a hypothetical example of a set of workflows that use services originating from three different research teams, shown in different colours in the figure. Each team has developed an ALife simulation and an analysis tool, that have been deployed on the portal. In this example, the analyses take three forms: Numeric data (I), Graphs (J) and animations (K), demonstrating that workflows offer a powerful and efficient way of re-using code, providing a common analysis base between research groups.

Example - Analysis of StringMol and Tierra data

As an illustration of the workflow capabilities, we created a scenario comparing evolutionary activity in StringMol and Tierra (see figure 2).The resulting workflow automates the experiment published in Droop and Hickinbotham (2012), in which population dynamics and three measures of evolutionary activity we calculated for configurations of Tierra and StringMol, configured with a range of mutation rates.

To achieve this, we created a modified Tierra service so that it created output in StringMol format. The ”popdy plotting service” was then created, based on existing R code that produced visualisation of the StringMol output data. The workflow loads data from a publicly shared location, pushes the data into both StringMol and Tierra, the outputs of which are separately analysed and written to a directory in the current user space.

Upon completion of the workflow execution, the output folder contains two PDF files from the Tierra and StringMol analyses respectively. See figure 4 for the analysis of Tierra data. The top row shows the population dynamics of different component types in Tierra during the run. The second row shows the non-neutral evolutionary activity, and the third and fourth rows show population- and count- based evolutionary activities of Bedau et al. (1997).

The output folder also contains an optional log file that was generated by the StringMol service. This workflow has been shared publicly for all users of YouShare.

Further Work

For the current contribution, we have implemented the ALife Zoo in the generic YouShare system. This allows a test of concept to be created quickly, and range of other generic services that have been implemented for other research domains to be available. Like many web applications, YouShare undergoes constant improvements as the user and

code base expands. Whilst the ALife Zoo could be hosted by the "vanilla plain" YouShare platform on an ongoing basis, we recognise that there may be special demands for specific features from the ALife research community. It is possible to branch the first tier of the service to deliver a bespoke user experience. We have successfully done this for several projects (e.g. Hickinbotham et al. (2012a)). Below we discuss potential improvements to the core YouShare system that could be implemented in future.

Visualisation

Where simulations produce large amounts of output data, it is desirable to create a visualisation of the data before deciding to download the data and proceeding to more detailed analysis. This problem has been partially addressed in the CMAC project, where interactive visualisation of the service output is made available on-line using the Raphael Javascript library (Anon., 2012b). The challenge in ALife Zoo is to make these visualisations as informative as possible, whilst keeping them sufficiently generic to be applicable to a range of simulations.

Linking to Publications

YouShare was developed to function as a Virtual Laboratory, and since publications are a permanent record of the work in Laboratories, it makes sense to link them to their data, services and workflows in YouShare. A facility to make such links is currently under development.

Linking to External HPC

Although the compute resource in YouShare is considerable, it is not infinite. Certain services may require high-memory or many-core processing to be delivered in a timely manner. YouShare could be extended to carry out the interfacing and staging of jobs on HPC clusters with relative ease, and so combine the advantages of HPC with the advantages of a browser-based user interface. A key issue here is handling big data, but ALife simulations tend to be relatively small in terms of their data footprint.

Linking to External Tier 1 Server

Finally, it is possible that particular specimens in the ALife Zoo may require special handling in their presentation to the public. In other words, the GWT-based youShare portal may not be sufficiently flexible to demonstrate particular simulations. For example, it would be extremely difficult to implement Avida-Ed, the educational version of Avida, in YouShare, due to the highly graphical and interactive nature of the interface. However, it is possible to develop other browser based front ends (for example, a LifeRay portal) that could access the YouShare back end whilst delivering a highly specialised user interface.

Exposing services to other web applications

Since the services in youShare are RESTful, it is straightforward to make them available to other web applications, rather than using the generic portal. We are in the process of developing an API to make these services available in this way. The main issue to be addressed is the maintenance and handling of user access rights to youShare.

Conclusion

YouShare provides a method for collaborative sharing of data and software. Software is deployed as "ready to go" services that are guaranteed to be deployed on the correct platforms of a heterogeneous compute facility. However, service implementation details are hidden from the user, allowing them to concentrate on service functionality alone. Mixed-platform services can be combined in a workflow environment to create more powerful tools. The YouShare compute facility can be extended to the Cloud, and with Virtual Machine (VM) technology we provide a sustainable software model. YouShare also encourages cross-domain research; there are currently 83 services on YouShare, including the new ALife services we describe here. These are a mix of Windows, Scientific Linux 4, and CentOS Linux 5 services, and cover domains such as neuroscience, text mining, image/3D processing, and neural networks, as well as generic services.

Two of the ALife services and workflows we report in this contribution have been highly influential in the field over the past two decades. With relatively little effort, and thanks to their open-source licenses, we have been able to make them available as web services for the first time. We are also able to show how analysis services developed more recently can be used to analyse the outputs of Tierra. This demonstrates that platforms such as YouShare are able to create flows of configuration, simulation and analysis between ALife researchers in a novel and efficient manner.

Acknowledgements

The youShare project funding is provided by the UK HEFCE University Modernisation Fund. CARMEN was developed under funding provided by the UK EPSRC on contract number EP/E002331/1, and currently supported by the UK BBSRC under contract number BB/1000984/1. CMAC project funding was funded by the UK Technology Strategy Board.

References

Adami, C. and Brown, C. (1994). Evolutionary learning in the 2d artificial life systems avida. In Brooks, R. and Maes, P., editors, *ALife IV*, pages 377–381. MIT Press, Cambridge, MA.

Anon. (2012a). Avida software: `http://sourceforge.net/projects/avida/files/avida-stable/2.12.4/` - retrieved 30th june 2013.

Anon. (2012b). Raphaelgwt: http://code.google.com/p/raphaelgwt/ - retrieved 3rd may 2013.

Austin, J., Fletcher, M., Jackson, T., Jessop, M., Turner, A., and Weeks, M. (2011). Youshare, an online collaboration research environment for sharing data and services. In *UK e-Science All Hands Meeting 26th-29th September 2011, York, United Kingdom.*

Bedau, M., Snyder, E., Brown, T., and Packard, N. (1997). A comparison of evolutionary activity in artificial evolving systems and in the biosphere. In *ALife IV*, pages 125–134. MIT Press.

Droop, A. and Hickinbotham, S. (2012). A quantitative measure of non-neutral evolutionary activity for systems that exhibit intrinsic fitness. In *ALife XIII*. MIT Press.

Eckerson, W. (1995). Three Tier Client/Server Architecture: Achieving Scalability, Performance, and Efficiency in Client Server Applications. In *Open Information Systems 10*.

Fortuna, M., Zaman, L., Wagner, A., and Ofria, C. (2013). Evolving digital ecological networks. *PLoS Comput Biol*, 9(3).

Hickinbotham, S., Austin, J., and McAvoy, J. (2012a). Interactive graphics on large datasets drives remote condition monitoring on a cloud. *J. Phys.: Conf. Ser.*, 364.

Hickinbotham, S., Clark, E., Stepney, S., Clarke, T., Nellis, A., Pay, M., and Young, P. (2010). Diversity from a monoculture: Effects of mutation-on-copy in a string-based artificial chemistry. In *ALife XII*. MIT Press.

Hickinbotham, S., Clark, E., Stepney, S., Clarke, T., Nellis, A., Pay, M., and Young, P. (2012b). Specification of the stringmol chemical programming language version 0.2. Technical Report YCS-2010-458, Univ. of York.

Jessop, M., Weeks, M., and Austin, J. (2010). CARMEN: a practical approach to metadata management. *Philosophical transactions. Series A, Mathematical, physical, and engineering sciences*, 368(1926):4147–4159.

Oracle (2013). Java servlet technology overview : http://www.oracle.com/technetwork/java/javaee/servlet/index.html - retrieved 3rd may 2013. Oracle.

Ray, T. (1991). An approach to the synthesis of life. In *ALife II*, pages 371–408.

Shao, J. and Ray, T. (2010). Maintenance of species diversity by predation in the tierra system. In *ALife XII*, pages 533–540. MIT Press, Cambridge, MA.

Watson, P., Jackson, T., Pitsilis, G., Gibson, F., Austin, J., Fletcher, M., Liang, B., and Lord, P. (2007). The CARMEN Neuroscience Server. In *Proceedings of the UK e-Science All hands Meeting*, pages 135–141.

Weeks, M., Jessop, M., Fletcher, M., Hodge, V., Jackson, T., and Austin, J. (2013). The carmen software as a service infrastructure. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 371(1983).