JABAWS Documentation

Release 2.2

Peter Troshin **Alexander Sherstnev** **Jim Procter**
Daniel Barton **Fábio Madeira** **Alexey Drozdetskiy**
Suzanne Duce **Geoffrey J. Barton**

Aug 18, 2017

The Barton Group
Division of Computational Biology
School of Life Sciences
University of Dundee
Dow Street
Dundee DD1 5EH
Scotland, UK

CONTENTS:

Note: Some of the hyper-links found in this pdf might not work properly.

CHAPTER

ONE

GETTING STARTED

JABAWS is a collection of web services for bioinformatics, and currently provides services that make it easy to access well-known multiple sequence alignment and protein disorder prediction programs (see the list of currently supported programs). Future versions of JABAWS will incorporate other tools.

JABAWS consists of a server and a client, but unlike most bioinformatics web-service systems, you can download and run both parts on your own computer! If you want a server just for yourself, then download and install the JABAWS Virtual Appliance (VA). It requires no configuration and is simple to install. If you want to install JABAWS for your lab or institution then download the JABAWS Web Application aRchive (WAR). It is slightly more complicated to configure but is very straightforward too. Finally, if you want to script against any version of JABAWS or are interested in writing your own client, the JABAWS Command Line Interface (CLI) client is what you need.

The public server based on JABAWS 2.1 at the University of Dundee has been in production since October 2013 and serviced over 442,000 jobs for users worldwide.

New: You can now run JABAWS with Docker. Click here to learn more.

1.1 JABAWS Benefits

- Can be deployed on most operating systems, as a VMware or compatible Virtual Appliance, as well as a Tomcat Java Web Application.
- Comes complete with sources and binaries for all the bioinformatics programs that it runs.
- Can operate as a stand alone server or one that submits jobs to a cluster via *DRMAA*.
- Easy to access from Jalview using its graphical client, or using the JABAWS command line client.
- Clients can submit jobs to any JABAWS servers that they might want to access, such as the one running on your local computer, your lab's server, or the publicly available services at the University of Dundee.
- Local or intranet installation eliminates any security concerns you might have about sending sensitive data over the internet.
- Wide range of configuration options to control size of jobs accepted by a server, and the command line options available for the program run by a service.

1.2 JABAWS Distributions

Tip: To help you choose the JABAWS distribution that better suits your needs read on the quickstart guides below.

I want to use JABAWS for...

- *Jalview and the JABAWS Public Server* - Running JABAWS services through Jalview on the JABAWS *public* server
- *Command Line Client (CLI)* - Accessing a *public* or *private* JABAWS server using the JABAWS client
- *Web Application aRchive (WAR)* - Running JABAWS for my group, lab, or organization on the *local* infrastructure
- *Virtual Appliance (VA)* - Running JABAWS services through Jalview or the CLI client on a *private* virtual machine server

1.2.1 Jalview and the JABAWS Public Server

Jalview, a multiple sequence alignment and analysis application, is a good example of a graphical JABAWS client. This client uses the same functionality as the JABAWS Command Line Interface (CLI) client, but instead allows JABAWS services to be accessed in a more user-friendly manner, through a graphical user interface. In this way, this is the easiest way to run JABAWS web services. Simply launch Jalview and run any of the methods provided under the 'Web Service' menu. Jalview uses the public JABAWS server by default. If you are concerned about privacy or want to run sensitive analysis on your own hardware, you can either setup a local JABAWS Virtual Appliance (VA) or configure the JABAWS Web Application aRchive (WAR) in your infrastructure.

1.2.2 Command Line Client (CLI)

This is a single Java archive which contains the JABAWS command line interface (CLI) client. It allows anyone who wants to connect to the JABAWS web-services running at the University of Dundee's Public Server, or to run a local private JABAWS server from their own software. You can read more about how to use JABAWS command line (CLI) client given in the CLI documentation pages, but a brief instructions are given below:

1. Download the Client Jar file
2. Download and install Java (version 1.7)
3. Provided that you have the Java ready to run, you can get command line help by changing to the directory where you downloaded the client jar, and typing:

```
java -jar jabaws-full-client-2.2.0.jar
```

The JABA Web Services are WS-I compliant. This means that you can access them from any language that has libraries or functions for consuming interoperable SOAP web services. More information on how to develop software that access JABAWS services is provided in the documentation pages.

1.2.3 Web Application aRchive (WAR)

The JABAWS Web Application aRchive (WAR) is for anyone who wants to run JABAWS for their group, lab or organization, or wants to enable their local JABAWS server to use the cluster or perform very large tasks. Complete documentation is provided in the WAR documentation pages, but brief instructions are given below:

1. Download the JABAWS WAR file
2. Download and install Apache-Tomcat

   You will need at least Tomcat version 5.5 of (we would recommend version 8.5) and at least *Java* 1.7 (i.e. JAVA 7).
3. Drop the JABAWS WAR file into `tomcat/webapps` directory.
4. (Re)start the Tomcat.
5. Once the tomcat has started, it should automatically unpack the WAR into the webapps directory (if it doesn't, simply unpack the WAR archive).
6. If you are on Mac or other unix-like architecture with GNU compilers available or you'd like to get a maximum performance

   `cd` to `webapps/jabaws/binaries/src/` and execute `./compilebin.sh` script to compile all binaries JABAWS depends on.

Testing

You can test that your JABAWS server is working in several ways.

1. Visit Services Status page available from the JABAWS main page using your web browser.
2. If you are working on the command line, then use the command line client shipped with the JABAWS war to test it by running:

```
java -jar <Path to tomcat WebApp directory>/jabaws/WEB-INF/lib/
↪jabaws-client.jar -h=http://localhost:8080/jabaws
```

   In this example we assumed that your JABAWS server URL is `http://localhost:8080` and JABAWS context path is *jabaws*

3. Alternately, you can point Jalview at your new server:
   (a) Launch the desktop version of Jalview
   (b) Open the Jalview desktop's preferences panel (from the Tools->Preferences menu option), elect the Web-services panel and press the New Service URL button.
   (c) Enter the URL for the tomcat server, including the context path for the JABAWS web app (e.g. http://localhost:8080/jabaws).

1.2.4 Virtual Appliance (VA)

The Virtual Appliance (VA) package allows you to run a JABAWS server installed on TurnKey Linux as a virtual machine on your laptop or desktop computer. A complete guide to the JABAWS VA is given in the VA documentation pages, but for the impatient, brief instructions are given below:

If you work on Windows, Linux or Unix:

1. Download JABAWS Virtual Appliance
2. Download and install VMWare Player
3. Unpack the JABAWS virtual appliance and open it with VMware Player

If you work on Mac do the same using VMware Fusion.

Testing

To check that your JABAWS virtual appliance is working visit the Services Status page available from the main JABAWS menu. For this enter the JABAWS URL for your new server into a web browser. This is shown once the appliance is booted up.

Alternatively you can use Jalview to complete the testing.

1. Launch the desktop version of Jalview
2. Open the Jalview desktop's preferences panel (from the Tools->Preferences menu option), select the `Webservices` panel and press the `New Service URL` button.
3. Enter the JABAWS URL for your new server. This is shown once the appliance is booted up.

CHAPTER

TWO

INCLUDED TOOLS

2.1 Multiple Sequence Alignment

- Clustal Omega (version 1.2.4)
- ClustalW (version 2.1)
- Mafft (version 7.310)
- Muscle (version 3.8.31)
- T-coffee (version 11.00.8cbe486)
- Probcons (version 1.12)
- MSAProbs (version 0.9.7)
- GLProbs (version 0.9.7)

2.2 Protein Disorder Prediction

- DisEMBL (version 1.5)
- IUPred (version 1.0)
- Jronn - Java implementation of Ronn (version 3.1)
- GlobPlot (version 2.3)

2.3 Amino Acid Conservation

- AACon (version 1.1)

2.4 RNA Secondary Structure

- RNAalifold from ViennaRNA (version 2.0)

CHAPTER

THREE

COMMAND LINE CLIENT (CLI)

The JABAWS client is a Java application that lets you run the programs for which a JABAWS server provides web services. This command line application this is able to call any of the JABAWS web services on any instance of JABAWS Server available over the web. The basic client is useful if you would like to test or execute the programs provided by the JABAWS server in your own scripts, but you do not want to handle any web service specific details. The client is an open source software, so you can also use the source code to as an example how to manipulate with JABAWS web services in your own code. The JABA Web Services are WS-I compliant. This means that you can access them from any language that has libraries or functions for consuming interoperable SOAP web services. More information on how to develop software that access JABAWS services is provided in the documentation pages.

The command line client comes as a part of client package which you are welcome to download. The command line client can be used to align sequences using any of JABAWS supported web services. The client is OS independent and supports most of the functions which can be accessed programmatically via JABAWS API. Using this client you could align sequences using presets or custom parameters, please see examples of this below. Here is the list of options supported by the command line client.

3.1 Installing

Tip: Check if you are running the recommended version of Java.

You need Java 7 or higher installed in your machine to be able to run the JABAWS CLI client. Please see the *Java* web site for up to date instructions and downloads.

3.2 Usage

```
java -jar jabaws-full-client-2.2.0.jar
```

```
Usage:
java -jar <path_to_jar_file> -h=host_and_context -s=serviceName ACTION [OPTIONS]
-h=<host_and_context> - a full URL to the JABAWS web server including context path e.
↪g. http://10.31.10.159:8080/ws
-s=<ServiceName> - one of [MafftWS, MuscleWS, ClustalWS, ClustalOWS, TcoffeeWS,␣
↪ProbconsWS, AAConWS, JronnWS, DisemblWS, GlobPlotWS, IUPredWS]
```

```
ACTIONS:
-i=<inputFile> - full path to fasta formatted sequence file, from which to align␣
↪sequences
-parameters - lists parameters supported by web service
-presets - lists presets supported by web service
-limits - lists web services limits
Please note that if input file is specified other actions are ignored


 OPTIONS: (only for use with -i action):
-r=<presetName> - name of the preset to use
-o=<outputFile> - full path to the file where to write an alignment
-f=<parameterInputFile> - the name of the file with the list of parameters to use.

Please note that -r and -f options cannot be used together. Alignment is done with␣
↪either preset or aparameters from the file, but not both!
```

3.3 Example Usage

Align sequences from input.fasta file using Mafft web service with default settings, print alignment in Clustal format to console.

```
java -jar jabaws-full-client-2.2.0.jar -h=http://myhost.compbio.ac.uk:8080/jabaws -
↪s=MafftWS -i=d:\input.fasta
```

Content of input.fasta file is show below (please note sequences has been trimmed for clarity)

```
>Foobar
MTADGPRELLQLRAAVRHRPQDFVAWL
>Bar
MGDTTAGEMAVQRGLALHQ
>Foofriend
MTADGPRELLQLRAAV
```

Align as in above example, but write output alignment in a file out.clustal, using parameters defined in prm.in file

```
java -jar jabaws-full-client-2.2.0.jar -h=http://myhost.compbio.ac.uk:8080/jabaws  -
↪s=MafftWS -i=d:\input.fasta -o=d:\out.clustal -f=prm.in
```

The content of the prm.in file is shown below

```
--nofft
--noscore
--fastaparttree
--retree=10
--op=2.2
```

The format of the file is the same for all JABAWS web services. Parameters are specified in exactly the same way as for native executables - alignment programs like Mafft etc. So parameters which you can use with command line version of an alignment program can be used with JABAWS. Most of the settings controlling alignment process are supported, but because any output has to be handled by JABAWS, settings controlling output are not allowed to be changed. For a list of parameters supported by a web service see the next example. In prm.in parameters are separated

by the new line, and name of the parameter is separated from its value with an equals sign. This format is constant no matter which JABAWS web service is used.

```
java -jar jabaws-full-client-2.2.0.jar -h=http://myhost.compbio.ac.uk:8080/jabaws -
↪s=MafftWS -parameters
```

The same client can be used to access JABAWS on different hosts. Just point the client to the host you want to use by changing the value of -h key.

For example you used `-h=http://myhost.compbio.ac.uk:8080/jabaws` server, now you want to use another server to `-h=http://mylabserver.myuni.edu`. This comes handy if your favorite server is off and you need to do the job yesterday.

CHAPTER

FOUR

WEB APPLICATION ARCHIVE (WAR)

JABAWS Web Application aRchive can run on any host operating system that supports Java and Apache-Tomcat. JABAWS requires a Java web application server compliant with version 2.4 of the Java Servlet specification, and a Java 7 runtime environment. We recommend using an official Oracle Java 7 runtime environment, and Apache-Tomcat web application server version 8.5, but older Tomcat versions above 5.5 will work too.

Danger: The JABAWS WAR is not generally compatible with older Mac systems based on the PowerPC architecture, since Java 1.7 is not available to run JABAWS.

However JABAWS depends on a number of third party programs which are not available for all operating systems. In particular, not all web services are currently available for MS Windows platform. JABAWS comes with pre-compiled MS Windows and Linux x86 binaries, as well as the source code and build scripts necessary to recompile them.

To run JABAWS on the cluster you must have shared disk space accessible from all cluster nodes.

4.1 Installing

Tip: Check if you are running the recommended versions of Java and Apache-Tomcat.

JABAWS is distributed as a web application archive (WAR). To deploy JABAWS in Apache-Tomcat - simply drop the war file into the webapps directory of a running Tomcat, and it will do the rest. If you used this deployment procedure, do not remove the Jabaws WAR file, otherwise Tomcat will undeploy your application! The context path for your deployed application will be the same as the name of the war file. For example, assuming the Tomcat server is running on the `localhost:8080` and *jaba.war* file is put into the `<tomcat server root>/webapps` directory, the deployed application from the jabaws.war file then can be accessed by this URL `http://localhost:8080/jabaws`.

For any other web application server, please follow your server's specific deployment procedure for 'WAR' files. If you install JABAWS on a MS Windows machine, then at this point your JABAWS installation will already be up and running, and you can try its services out as described here in the documentation. If you install JABAWS on Linux you will need to compile the binaries for your system and set an executable flag for binaries (more details here and here).

4.2 Usage

Running many JABAWS instances on the same server

JABAWS is supplied as a Web Application aRchive which can be dealt with as any other web applications. So it is perfectly possible to run two JABAWS instances from the same server. Just make two different contexts on your application server and unpack JABAWS in both of them. For example if your server name is http://www.align.ac.uk, and the context names are public and private. Than one group of users could be given a URL http://www.align.ac.uk/public and another http://www.align.ac.uk/private. These contexts will be served by two independent JABAWS instances, and could be configured differently. If you keep local engine enabled, make sure you reduce the number of threads local engine is allowed to use to avoid overloading the server. Alternatively two completely separate web application server instances (e.g. Apache-Tomcat) could be used. This will give you a better resilience and more flexibility in memory settings.

JABAWS on a single server

You can run JABAWS on a single server. Obviously the capacity will be limited, but it may be sufficient for a small lab. Installed on a single server, JABAWS executes tasks in parallel, so the more cores the server has the more requests it will be able to handle.

JABAWS supported cluster batch management systems

JABAWS uses DRMAA v1.0 library to send and manage jobs on the cluster. DRMAA supports many different cluster job management systems. Namely Sun Grid Engine, Condor, PBS, GridWay, Globus 2/4, PBSPro, LSF. For up to date information please consult DRMAA web site. We found that DRMAA implementation differ from platform to platform and were trying to use only the basic functions. We have only tested JABAWS on Sun Grid Engine v 6.2. Please let use know if you have any experience of running JABAWS on other platforms.

4.3 Troubleshooting

If Apache-Tomcat fails to deploy jabaws.war file:

- Make sure Tomcat has sufficient access rights to read your war file.
- Restart the Tomcat, sometimes it will not restart after the new war file is added without restart
- If Tomcat still refuses to unpack the war file, unpack it manually into web application folder (the war file is just a zip archive). Restart the Tomcat.

If Tomcat undeploys your application:

If the war file is automatically removed by Tomcat use an explicit application descriptor. Add a context descriptor file into `<tomcatRoot>conf/Catalina/localhost` directory. Name your context file the same as your application folder e.g. if your JABAWS resides in `webapps/jabaws` folder, then call the context file jabaws.xml. Below is an example of content this file might have.

```
<?xml version="1.0" encoding="UTF-8"?>
<Context antiResourceLocking="false" privileged="true" />
```

This should be sufficient to prevent Tomcat from removing your JABAWS from WEBAPPS. For more information about the Tomcat deployer read this documentation on the Apache-Tomcat web site.

CHAPTER

FIVE

VIRTUAL APPLIANCE (VA)

The JABAWS Virtual Appliance is a way to run a JABAWS server locally, without the need to connect to the internet or configure JABAWS. What the appliance provides is a 'virtual server machine' (or more simply - *virtual machine* or *VM*), running an installation of the JABAWS Web Application Archive (WAR) on TurnKey Linux 12.1 (Standlone Tomcat). Once this has started up, it displays a message indicating the IP address of the JABAWS server, allowing any JABAWS client (such as Jalview or the JABAWS command line client) to connect to it.

You can run the appliance with freely available program such as VMware Player, but you will need to install it first. We have tested the JABAWS appliance with VMware Player v 3.1.2 on Windows and Linux, and VMware Fusion on Mac. However, you are not limited to these virtualization systems and can use the JABAWS Appliance with any other virtualization platform. You can use VMware OVF tool to prepare JABAWS image for a different virtualization platform e.g. VirtualBox. VirtualBox can also use the `jabaws*.vmdk` directly as an *existing preconfigured virtual hard disk file*, when creating a new Linux 2.4 / 3.x / 4.x 64-bit VM.

Note: The appliance best suits users who would like to use the JABAWS web-services locally. This might be because they do not want to access systems over an internet, or just want to keep their data private. It is also the recommended option for users who want to install JABAWS on Windows, which does not support all the bioinformatics programs that JABAWS can run.

For servers that will be used heavily, we recommend that a JABAWS Server WAR distribution is deployed, rather than the Virtual Appliance version of JABAWS. This is because the JABAWS appliance is pre-configured to use only 1 CPU and 512M of memory (where the minimum amount of memory required for a JABAWS server is about 378M), which is unlikely to be sufficient for heavy computation. It is possible to reconfigure the virtual appliance so it uses more computation resources, but for most production environments, the JABAWS WAR distribution will be easier to deploy and fine tune to take advantage of the available resources.

5.1 Installing

Tip: Check if you are running the recommended version of VWMare.

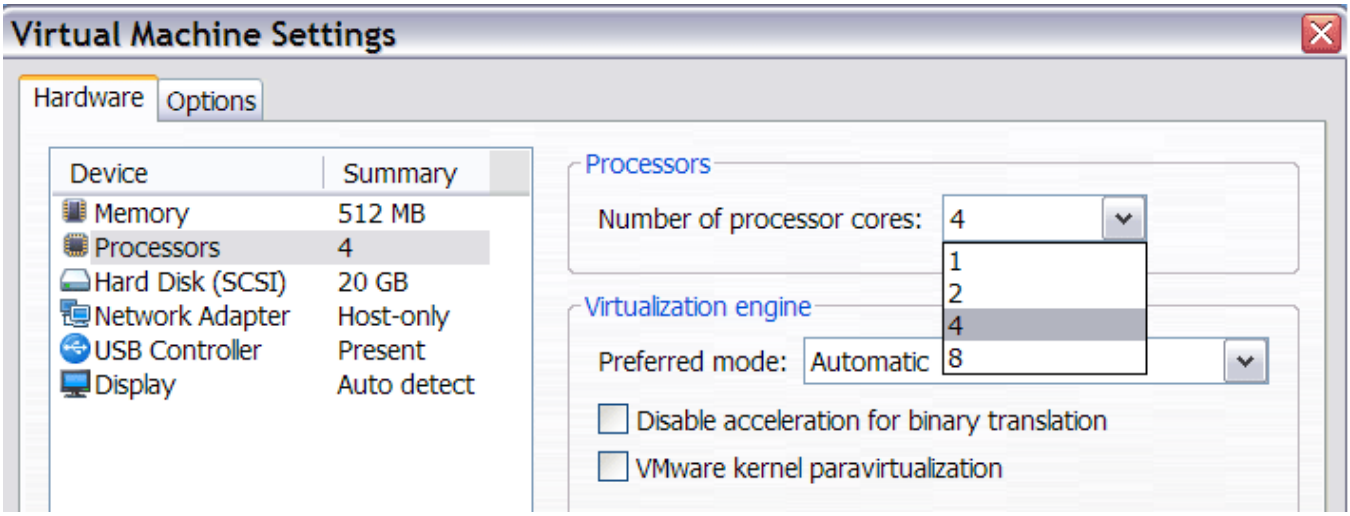
The free VMware Player can be used to run the JABAWS services from the Windows and Linux host operating systems. VMware Fusion, a commercial VMware product, offers virtual machine support for Mac.

To run the JABAWS server on VMware player, unpack the JABAWS VM into one of the folders on your local hard drive. Open VMware Player, click "Open Virtual Machine" and point the Player to the location of the JABAWS, then choose the `jabaws*.vmx` file to open an appliance.

When you play the machine for the first time the Player might ask you whether "This virtual machine may have been moved or copied.", say that you have copied it. That is all.

5.2 Usage

By default, the JABAWS virtual appliance is configured with 512M of memory and 1 CPU, but you are free to change these settings. If you have more than one CPU or CPU core on your computer you can make them available for the JABAWS virtual machine by editing virtual machine settings. Please bear in mind that more CPU power will not make a single calculation go faster, but it will enable the VM to do calculations in parallel. Similarly, you can add more memory to the virtual machine. More memory lets your VM deal with larger tasks, e.g. work with large alignments.



The VMware Player screen shot above displays JABAWS VM CPU settings.

5.3 Configuration

5.3.1 VM configuration

VMware info

- CPUs : 1
- RAM : 512 MB
- Networking : Host only (the VM has no access to the outside network, nothing from the outside network can access the VM)
- Hard disk : 20 GB (expanding)
- VMware tools : Installed

OS information

- OS : TurnKey Linux (v. 14.1, Standalone Tomcat) based on Debian GNU/Linux 8 (Jessie)
- Installation : Oracle Java 7, Tomcat 7, JABAWS v. 2.2

- IPv4 address : dhcp
- IPv6 address : auto
- DNS name : none
- Name server : dhcp
- Route : dhcp
- Keyboard : US_intl

Login credentials

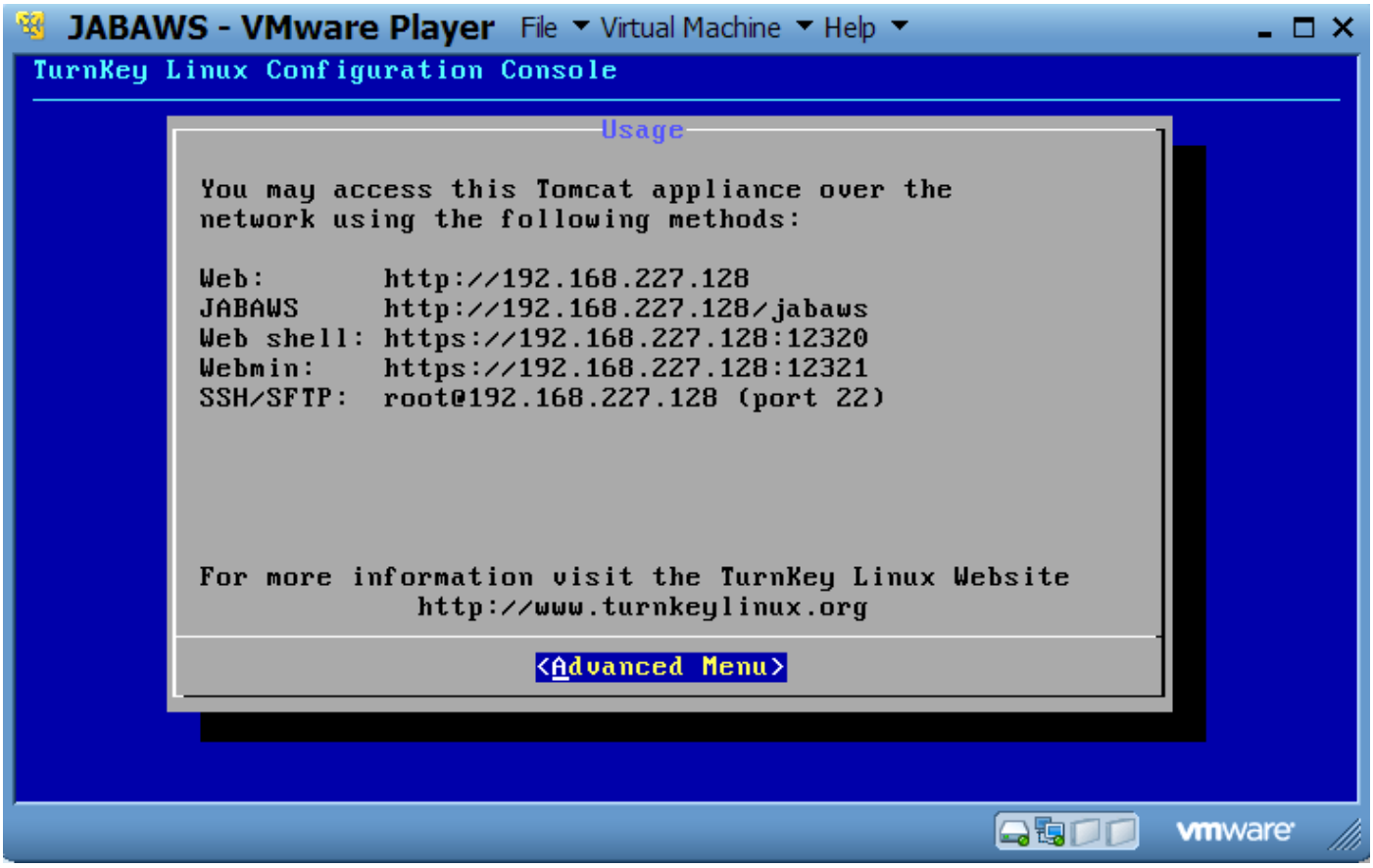
- Root password: JabawsAdmin1
- MySQL password: JabawsAdmin1
- Tomcat admin password: JabawsAdmin2

Services available at the virtial machine IP (e.g. VM_IP = 172.16.232.149)

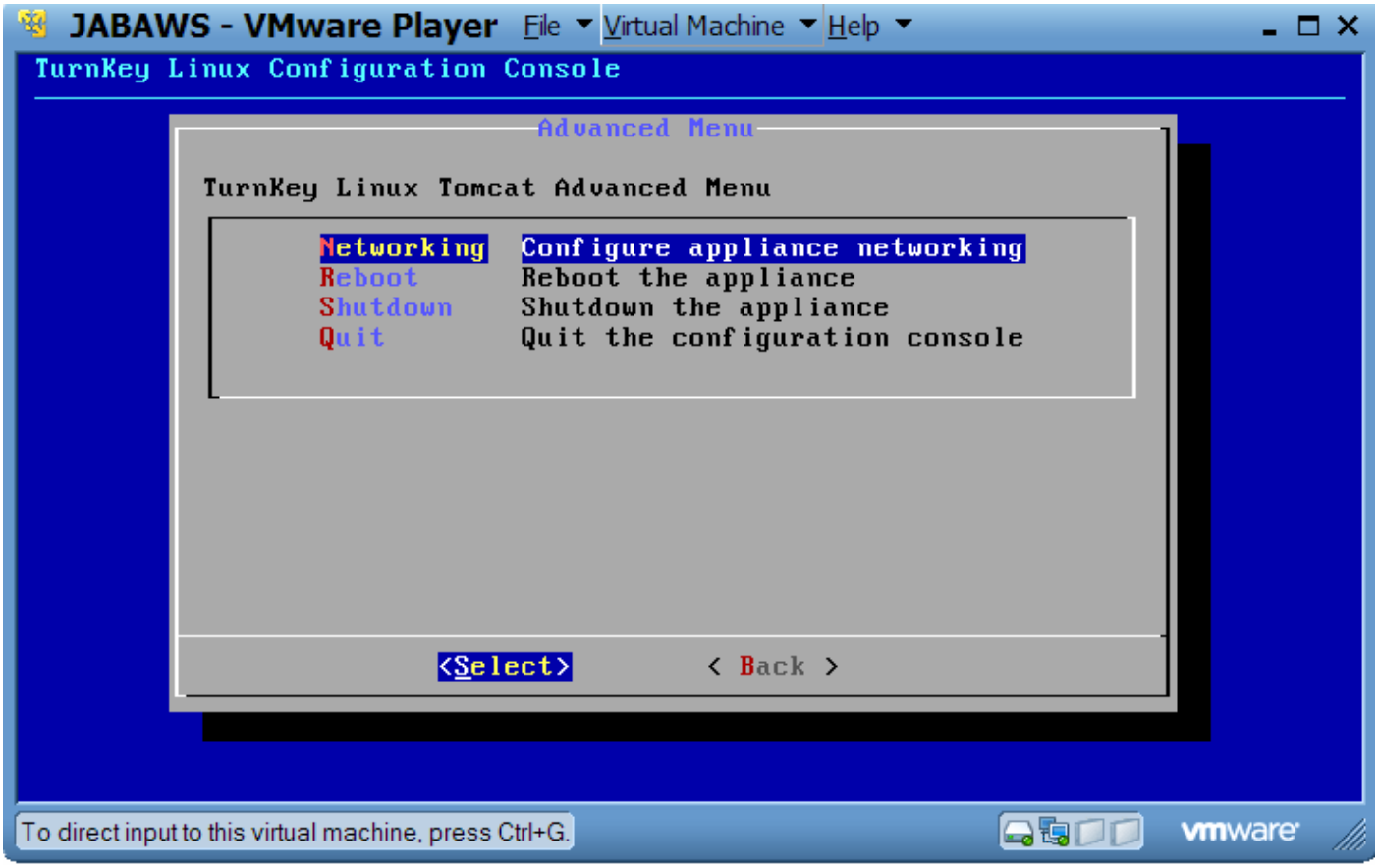
- Tomcat Web Server: http://VM_IP (e.g. http://172.16.232.149)
- Jabaws URL: http://VM_IP/jabaws (e.g. http://172.16.232.149/jabaws)
- Web Shell: https://VM_IP:12320/ (e.g. https://172.16.232.149:12320)
- Webmean: https://VM_IP:12321/ (e.g. https://172.16.232.149:12321)
- SSH/SFTP: root@VM_IP (e.g. ssh root@172.16.232.149)

5.3.2 JABAWS configuration

After booting the JABAWS VM, you should see similar screen, however, the IP address of your VM may be different. To enable Jalview to work with your JABAWS appliance you need to go to Jalview->Tools->Preferences->Web Services -> New Service URL, and add JABAWS URL into the box provided. For more information please refer to Jalview help pages.



If you click on Advanced Menu, you will see the configuration console, similar to the one below.



By default the JABAWS VM is configured to use host-only networking. This means that the host can communicate with the VM via a network, but no other machines can. Similarly, the VM cannot communicate with any other computers apart from the host. If you want to connect to the Internet from the VM, configure your VM to use NAT network. However, you will not be able to connect to the VM from the host in such case. If you want to be able to connect to your VM and let VM connect to the internet at the same time you would have to use a Bridged network. In such a case you would have to configure the VM IP address manually (unless of course your network has a DHCP server to do that).

CHAPTER

SIX

DOCKER CONTAINER

An alternative to running the JABAWS Virtual Appliance (VA) or the JABAWS Web Application aRchive (WAR) in your local machine, is to use Docker to handle all the requirements to deploy JABAWS in Tomcat. We provide a Dockerfile, which Docker uses to build an exact image of the required system, (i.e. Apache Tomcat server with the JABAWS WAR deployed).

6.1 Installing Docker

In order to run JABAWS using Docker, you need docker installed and running in your system. Installation instructions are provided in the Docker documentation pages.

6.2 Running JABAWS with Docker

Once you have docker installed and running in your machine:

```
docker build -t jabaws:2.2 http://www.compbio.dundee.ac.uk/jabaws22/archive/docker/
↪Dockerfile
```

This will download the required Docker contexts (Tomcat, etc.) and setup JABAWS. Once the setup is finished you can run the JABAWS container with:

```
docker run --rm -it -p 8080:8080 jabaws:2.2
```

This will launch Tomcat and deploy JABAWS. By accessing `localhost:8080/jabaws/` you can verify whether the JABAWS is running properly. You can then use your Docker JABAWS container to power your analysis using Jalview or the JABAWS CLI.

CHAPTER

SEVEN

ADVANCED USAGE

JABAWS web services are WS-I basic profile compliant, which means they can be accessed using any programming language or system that can utilize standard SOAP web services. The Web Service Definition Language (WSDL) for each service is published on the JABAWS home page, and you can use this to automatically generate service bindings for your program. If you use Java you may wish to use our client package to access JABAWS. This package is based on the autogenerated source code produced by wsimport, which is the Java tool for creating web service bindings. In addition, this offers some additional methods that simplify working with JABAWS. For more information please refer to the data model javadoc.

7.1 Valid WSDL

Multiple sequence alignment services

- ClustalOWS - http://www.compbio.dundee.ac.uk/jabaws/ClustalOWS?wsdl
- ClustalWS - http://www.compbio.dundee.ac.uk/jabaws/ClustalWS?wsdl
- MuscleWS - http://www.compbio.dundee.ac.uk/jabaws/MuscleWS?wsdl
- MafftWS - http://www.compbio.dundee.ac.uk/jabaws/MafftWS?wsdl
- TcoffeeWS - http://www.compbio.dundee.ac.uk/jabaws/TcoffeeWS?wsdl
- ProbconsWS - http://www.compbio.dundee.ac.uk/jabaws/ProbconsWS?wsdl
- MSAprobsWS - http://www.compbio.dundee.ac.uk/jabaws/MSAprobsWS?wsdl
- GLprobsWS - http://www.compbio.dundee.ac.uk/jabaws/GLprobsWS?wsdl

Protein disorder prediction services

- IUPredWS - http://www.compbio.dundee.ac.uk/jabaws/IUPredWS?wsdl
- GlobPlotWS - http://www.compbio.dundee.ac.uk/jabaws/GlobPlotWS?wsdl
- DisemblWS - http://www.compbio.dundee.ac.uk/jabaws/DisemblWS?wsdl
- JronnWS - http://www.compbio.dundee.ac.uk/jabaws/JronnWS?wsdl

Amino acid conservation service

- AAConWS - http://www.compbio.dundee.ac.uk/jabaws/AAConWS?wsdl

RNA Secondary Structure Prediction

- RNAalifoldWS - http://www.compbio.dundee.ac.uk/jabaws/RNAalifoldWS?wsdl

Please replace http://www.compbio.dundee.ac.uk/ with your JABAWS instance host name, and jabaws with your JABAWS context name to access your local version of JABAWS web services. For example http://localhost:8080/jabaws would be a valid URL for the default Apache-Tomcat installation and jabaws.war file deployment.

7.2 JABAWS Configuration

There are three parts of the system you can configure. The local and the cluster engines, and the paths to the individual executables for each engine. These settings are stored in configuration files within the web application directory (for an overview, then take a look at the war file content table).

Initially, JABAWS is configured with only the local engine enabled, with job output written to directory called "jobsout" within the web application itself. This means that JABAWS will work out of the box, but may not be suitable for serving a whole lab or a university.

7.2.1 Local Engine Configuration

The Local execution engine configuration is defined in the properties file `conf/Engine.local.properties`. The supported configuration settings are:

`engine.local.enable=true` - enable or disable local engine, valid values true | false

`local.tmp.directory=D:\\clusterengine\\testoutput` - a directory to use for temporary files storage, optional, defaults to java temporary directory

`engine.local.thread.number=4` - Number of threads for tasks execution (valid values between 1 and 2x cpu. Where x is a number of cores available in the system). Optional defaults to the number of cores for core number <=4 and number of cores-1 for greater core numbers.

If the local engine going to be heavily loaded (which is often the case if you do not have a cluster) it is a good idea to increase the amount of memory available for the web application server. If you are using Apache-Tomcat, then you can define its memory settings in the JAVA_OPTS environment variable. To specify which JVM to use for Apache-Tomcat, put the full path to the JRE installation in the JAVA_HOME environment variable. (We would recommend using Sun Java Virtual Machine (JVM) in preference to Open JDK). Below is an example of code which can be added to `<tomcat_dir>/bin/setenv.sh` script to define which JVM to use and a memory settings for Tomcat server. Tomcat server startup script (`catalina.sh`) will execute `setenv.sh` on each server start automatically.

```
export JAVA_HOME=/homes/ws-dev2/jdk1.6.0_17/
export JAVA_OPTS="-server -Xincgc -Xms512m -Xmx1024m"
```

7.2.2 Cluster Engine Configuration

Supported configuration settings:

`engine.cluster.enable=true` - enable or disable local engine true | false, defaults to false

`cluster.tmp.directory=/homes/clustengine/testoutput` - a directory to use for temporary files storage. The value must be an absolute path to the temporary directory. This is required. The value must be different from what is defined for local engine. This directory must be accessible from all cluster nodes.

For the cluster engine to work, the SGE_ROOT and LD_LIBRARY_PATH environment variables have to be defined. They tell the cluster engine where to find DRMAA libraries. These variables should be defined when the web application server starts up, e.g.

```
SGE_ROOT=/gridware/sge
LD_LIBRARY_PATH=/gridware/sge/lib/lx24-amd64
```

Finally, do not forget to configure executables for the cluster execution, they may be the same as for the local execution but may be different. Please refer to the executable configuration section for further details.

7.2.3 Executable Configuration

All the executable programs are configured in conf/Executable.properties file. Each executable is configured with a number of options. They are:

```
local.X.bin.windows=<path to executable under windows system, optional>
local.X.bin=<path to the executable under non-windows system, optional>
cluster.X.bin=<path to the executable on the cluster, all cluster nodes must see it,␣
↪optional>
X.bin.env=<semicolon separated list of environment variables for executable, use hash␣
↪symbol as name value separator, optional>
X.--aamatrix.path=<path to the directory containing substitution matrices, optional>
X.presets.file=<path to the preset configuration file, optional>
X.parameters.file=<path to the parameters configuration file, optional>
X.limits.file=<path to the limits configuration file, optional>
X.cluster.settings=<list of the cluster specific options, optional>
```

Where X any of the bioinformatics tools available (e.g. clustalw, muscle, mafft, probcons, t-coffee, etc.).

Default JABAWS configuration includes path to local executables to be run by the local engine only, all cluster related settings are commented out, but they are there for you as examples. Cluster engine is disabled by default. To configure executable for cluster execution uncomment the X.cluster settings and change them appropriately.

By default limits are set well in excess of what you may want to offer to the users outside your lab, to make sure that the tasks are never rejected. The default limit is 100000 sequences of 100000 letters on average for all of the JABA web services. You can adjust the limits according to your needs by editing `conf/settings/<X>Limit.xml` files. After you have completed the editing your configuration may look like this:

```
local.mafft.bin=binaries/mafft
cluster.mafft.bin=/homes/cengine/mafft
mafft.bin.env=MAFFT_BINARIES#/homes/cengine/mafft;FASTA_4_MAFFT#/bin/fasta34;
mafft.--aamatrix.path=binaries/matrices
mafft.presets.file=conf/settings/MafftPresets.xml
mafft.parameters.file=conf/settings/MafftParameters.xml
mafft.limits.file=conf/settings/MafftLimits.xml
mafft.cluster.settings=-q bigmem.q -l h_cpu=24:00:00 -l h_vmem=6000M -l ram=6000M
```

Please not that relative paths must only be specified for the files that reside inside web application directory, all other paths must be supplied as absolute!

Furthermore, you should avoid using environment variables within the paths or options - since these will not be evaluated correctly. Instead, please explicitly specify the absolute path to anything normally evaluated from an environment variable at execution time.

If you are using JABAWS to submit jobs to the cluster (with cluster engine enabled), executables must be available from all cluster nodes the task can be sent to, also paths to the executables on the cluster e.g. `cluster.<exec_name>.bin` must be absolute.

Executables can be located anywhere in your system, they do not have to reside on the server as long as the web application server can access and execute them.

Cluster settings are treated as a black box, the system will just pass whatever is specified in this line directly to the cluster submission library. This is how DRMAA itself treats this settings. More exactly DRMAA `JobTemplate.setNativeSpecification()` function will be called.

For further details and examples of configuration please refer to the `Executable.properties` file supplied with JABAWS.

7.2.4 Defining Environment Variables for Executables

Environment variables can be defined in property

```
x.bin.env
```

Where x is one of thw executables supported by JABAWS. Several environment variables can be specified in the same line. For example.

```
mafft.bin.env=MAFFT_BINARIES#/homes/cengine/mafft;FASTA_4_MAFFT#/bin/fasta34;
```

The example above defines two environment variables with names `MAFFT-BINARIES` and `FASTA_4_MAFFT` and values `/homes/cengine/mafft and /bin/fasta34` respectively. Semicolon is used as a separator between different environment variables whereas hash is used as a separator for name and value of the variable.

7.2.5 Configure JABAWS to Work with Mafft

If you use default configuration you do not need to read any further. The default configuration will work for you without any changes, however, if you want to install Mafft yourself then there is a couple of more steps to do.

Mafft executable needs to know the location of other files supplied with Mafft. In addition some Mafft functions depends on the fasta executable, which is not supplied with Mafft, but is a separate package. Mafft needs to know the location of fasta34 executable.

To let Mafft know where the other files from its package are, change the value of MAFFT-BINARIES environment variables. To let Mafft know where is the fasta34 executable set the value of FASTA_4_MAFFT environment variable to point to a location of fasta34 program. The latter can be added to the PATH variable instead. If you are using executables supplied with JABAWS, the path to Mafft binaries would be like `<relative path to web application directory>/binaries/src/mafft/binaries` and the path to fasta34 binary would be `<relative path to web application directory>/binaries/src/fasta34/fasta34`. You can specify the location of Mafft binaries as well as fasta34 program elsewhere by providing an absolute path to them. All these settings are defined in `conf/Executable.properties` file.

7.2.6 Limiting the size of the job accepted by JABAWS

JABAWS can be configured to reject excessively large tasks. This is useful if you operate JABAWS service for many users. By defining a maximum allowed task size you can provide an even service for all users and prevents waste of resources on the tasks too large to complete successfully. You can define the maximum number of sequences and the maximum average sequence length that JABAWS accepts for each JABA Web Service independently. Furthermore, you can define different limits for different presets of the same web service.

By default limits are disabled. You can enable them by editing `conf/Executable.properties` file. You can adjust the limits according to your needs by editing `conf/settings/<X>Limit.xml` files.

7.2.7 Pre-compiled binaries

JABAWS comes with pre-compiled x86 Linux binaries, thus on such systems JABAWS should work straight out of the box. If you are in any doubts or experience problems you may want to make sure that the binaries supplied work under your OS. The source code for these programs is also provided so you can recompile the binaries for your own architecture and exploit any optimizations that your system can provide.

To check if the bundled binaries are working in your system execute each binary, without any command line options or input files. If you see an error message complaining about missing libraries or other problems, then you probably need to recompile the binaries.

Alternately, if you have already got binaries on your system, then you can simply reuse the existing binaries by updating the paths in JABAWS's configuration files so these are used instead. If you have a different version of an executable (e.g. an alignment program) which you prefer, you could use it as long as it supports all the functions JABAWS executable require. This could be the case with more recent executable. If the options supported by your chosen executable is different from the standard JABAWS executable, then you need to edit the `ExecutableNameParamaters.xml` configuration file.

You can try all the JABAWS functionality with the JABAWS test client or have a look at deploying on Tomcat tips if you experience any problems.

Note: You may want to enable logging for testing different executables as described in section 'JABAWS Internal Logging'

7.2.8 Recompiling binaries for your system

If you have a fully equipped build environment on your (POSIX-like) system, then you should be able to recompile the programs from the source distributions which are included in the JABAWS war file. A script called 'compilebin.sh' is provided to automate this task.

1. In a terminal window, change the working directory to `binaries/src`
2. Execute the compilebin.sh script:

```
chmod +x compilebin.sh; compilebin.sh > compilebin.out;
```

3. Then run:

```
chmod +x setexecflag.sh; sh setexecflag.sh
```

If any of the binaries was not recompiled, then a 'file not found' error will be raised.

4. Finally, restart your Tomcat server (or JABAWS application only), and test JABAWS to check that it can use the new binaries.

If you couldn't compile everything, then it may be that your system does not have all the tools required for compiling the programs. At the very least check that you have gcc, g++ and make installed in your system. If not install these packages and repeat the compilation steps again. You should also review the compilebin.sh output - which was redirected to compilebin.out, and any errors output to the terminal.

7.2.9 Obtaining or reusing binaries

You could search for pre-packaged compiled executable in your system package repository or alternately, download pre-compiled binaries from each alignment program's home page. Then, either replace the executables supplied with the downloaded ones, or modify the paths defined in `executable.properties` as described below. .. Below are some suggestions on where you may be able to get the binaries for your system.

If you would like to use the binaries you already have, then you just need to let JABAWS know where they are. To do this, edit: `conf/Executable.properties`

When specifying paths to executables that already exist on your system, make sure you provide an absolute path, or one relative to the JABAWS directory inside webapps. For example, the default path for clustalw is defined as `local.clustalw.bin=binaries/src/clustalw/src/clustalw2` Alternatively, instead of changing `Executable.properties` you could also replace the executables bundled with JABAWS with the ones that you have, or make symbolic links to them. Then the default configuration will work for you. More information about the Executable.properties file is given in the JABAWS Configuration page.

7.3 Load balancing

If your cluster is busy and has significant waiting times, you can achieve a faster response by allowing the server machine to calculate small tasks and then reserve the cluster for bigger jobs. This works especially well if your server is a powerful machine with many CPUs. To do this you need to enable and configure both the cluster and the local engines. Once this is done decide on the maximum size of a task to be run on the server locally. Then, edit "# LocalEngineExecutionLimit #" preset in `<ServiceName>Limits.xml` file accordingly. JABAWS server then will balance the load according to the following rule: If the task size is smaller than the maximum task size for local engine, and the local engine has idle threads, then it calculates task locally otherwise it submit the task to the cluster.

7.4 Testing the JABAWS Server

Access `<your_JABAWS_server_URL>/ServiceStatus` to test all web services. Each time you access this URL, all services are tested. For production configuration we recommend prohibiting requests to this URL for non authenticated users to prevent excessive load on the server.

Alternatively, you can use a command line client (part of the client only package) to test your JABAWS installation as described here. If you downloaded a JABAWS server package, you can use `<your_jaba_context_name>/WEB-INF/lib/jaba-client.jar` to test JABAWS installation as described here. If you downloaded the source code, then you could run a number of test suites defined in the build.xml Apache Ant file.

First of all make sure that Tomcat server is started successfully. If this was the case, then you should see JABAWS home page when you navigate to your Tomcat JABAWS context path in your browser (e.g. at `http://myhost.compbio.ac.uk:8080/jabaws` => `<jabaws_server>`)

If you see it, then it is time to make sure that web services are working too. The easiest way to do this is to access Services Status page available from the main JABAWS web page menu.

If you need to monitor web service health automatically when the best option is to use service checker that responds with the standard HTTP status code. To access this checker use the following URL:

Using JABAWS service status checker

If you see it, then it is time to make sure that web services are working too. The easiest way to do this is to access Services Status page available from the main JABAWS web page menu.

If you need to monitor web service health automatically when the best option is to use service checker that responds with the standard HTTP status code. To access this checker use the following URL: `<jabaws-server>/HttpCodeResponseServiceStatus` or alternatively `<jabaws-server>/man_serverwar.jsp`

This page returns code 200, and no page context if all services are operational, 503 if one of the services have problems. You can also check each web service individually by providing the name of the web service to check at the end of the service checker URL like this: `<jabaws_server>/HttpCodeResponseServiceStatus/ClustalWS`

Upon request, the service status checker will examine the health of the ClustalWS web service only. If the service name is not valid, then the service checker will return code 400.

Using command line client

Alternatively, you should be able to use the test program which can be found in `<webapplicationpath>/WEB-INF/lib/jabaws-client.jar` file. To run the tests type:

```
java -jar jabaws-client.jar -h=<Your web application server host name, port and␣
↪JABAWS context path>
```

For example to test all JABAWS web services on host myhost.compbio.ac.uk type:

```
java -jar jabaws-client.jar -h=http://myhost.compbio.ac.uk:8080/jabaws
```

You can choose a particular web server using -s option like this java -jar jabaws-client.jar -h=http://myhost.compbio.ac.uk:8080/jabaws -s=ClustalWS This command line assumes that java executable is in your path and jabaws-client.jar is located in the current directory.

An example of the report testing tool produces for operating web service looks like this:

```
Connecting to service MuscleWS on http://myhost.compbio.ac.uk:8080/jabaws ... OK
Testing alignment with default parameters:
Queering job status...OK
Retrieving results...OK
Testing alignment with presets:
Aligning with preset 'Protein alignment(Fastest speed)'... OK
Aligning with preset 'Nucleotide alignment(Fastest speed)'... OK
Aligning with preset 'Huge alignments (speed-oriented)'... OK
Queering presets...OK
Queering Parameters...OK
Queering Limits...OK
Queering Local Engine Limits...OK
Check is completed service MuscleWS IS WORKING
```

An example of the response of a web service which is deployed but is not operating is below:

```
Connecting to service ProbconsWS on http://localhost:8080/ws ... OK
Testing alignment with default parameters:FAILED
Service ProbconsWS IS NOT FUNCTIONAL
```

If the web server did not respond the message looks like following:

```
Connecting to service TcoffeeWS on http://localhost:8080/ws ... FAILED
```

7.5 JABAWS internal logging

JABAWS can be configured to log what it is doing. This comes in handy if you would like to see who is using your web services or need to chase some problems. JABAWS uses log4j to do the logging, the example of log4j configuration is bundled with JABAWS war file. You will find it in the `/WEB-INF/classes/log4j.properties` file. All the lines in this file are commented out. The reason why the logging is disabled by default it simple, log4j has to know the exact location of where the log files are stored. This is not known up until the deployment time. To enable the logging you need to define logDir property in the log4j.properties and uncomment section of the file which corresponds to your need. More information is given in the log4j.properties file itself. Restart the Tomcat or the JABAWS web application to apply the settings.

After you have done this, assuming that you did not change the log4j.properties file yourself, you should see the application log file called activity.log. The file called activity.log. The amount of information logged can be adjusted using different logging levels, it is reduced in the following order of log levels TRACE, DEBUG, INFO, WARN, ERROR, FATAL.

If you would like to know who is using your services, you might want to enable Tomcat request logging.

7.5.1 JABAWS requests logging

Enable Tomcat log valve. To do this uncomment the following section of <tomcat_root>/conf/server.xml configuration file.

```
<Valve className="org.apache.catalina.valves.AccessLogValve" directory="logs"
         prefix="localhost_access_log." suffix=".txt" pattern="common" resolveHosts=
↪"false"/>
```

The following information will be logged:

Remote IP	Date | Method server_URL protocol		HTTP status	Response size in bytes
10.31.11.159	[10/Feb/2010:16:51:32 +0000]	“POST /jws2/MafftWS HTTP/1.1”	200	2067

Which can be processed in various programs for log analysis, such as WebAlizer, Analog, AWStats.

7.6 JABAWS and Google Analytics

JABAWS reports web services usage to our group Google Analytics (GA) account. JABAWS usage statistics are collected for funding and reporting purposes, and no private information is collected. The data sent by JABAWS is as follows:

1. The IP address of the JABAWS server machine (the server IP can anonymized see `conf/GA.properties` config file)
2. The name of the web service that was called.
3. A few details of the system such as JABAWS version, java version, user language, color depth, screen resolution and character encoding.

Google Analytics can be disabled or adjusted by removing/editing `conf/GA.properties` Google Analytics (GA) settings file. We would appreciate it greatly if you could leave it on!

All calls to GA are very lightweight, completed asynchronously, create very little overhead and do not influence the server response time or performance.

7.7 JABAWS War File Content

Directory	Content description
conf/ contains	configuration files such as Executable.properties, Engine.local.properties, Engine.cluster.properties
conf/settings	Contains individual executable description files. In particular XXXParameters.xml, XXXPresets.xml, XXXLimits.xml where XXX is the name of the executable
Execution-Statistics	The database for storing the execution statistics
statpages	Web pages for usage statistics visialization and webservices status queries
jobsout/	Contains directories generated when running an individual executable. E.g. input and output files and some other task related data (optional)
binaries/	Directory contains native executables - programs, windows binaries (optional)
binaries/src	Contains source of native executables and Linux i386 binaries
bina-ries/windows	Contains binaries for MS Windows operating system
bina-ries/matrices	Substitution matrices
WEB-INF	Web application descriptor
WEB-INF/lib	Web application libraries
WEB-INF/classes	log4j.properties - log configuration file (optional)
static	Static content such as CSS, JavaScript and Image files

Help Pages	
/	help pages, index.html is the starting page
dm_javadoc	JavaDoc for the JABAWS Data Model
full_javadoc	JavaDoc for the complete JABAWS
prog_docs	Documentation for programs that are included in JABAWS

CHAPTER

EIGHT

FOR DEVELOPERS

8.1 Source Code

Note: This is an open source project. If you want to contribute or report an issue have a look at our Git-tracker.

Publicly available Git repository: http://source.jalview.org/gitweb/?p=jabaws.git

```
git clone http://source.jalview.org/git/jabaws.git
```

8.2 The API

Data Model JavaDoc - read this if your are coding against JABA Web Services

Complete JavaDoc - for developers who want to use JABAWS framework and use Engines and Executables directly

8.3 Structure of the project

- *binaries* contains native executables e.g. clustalw
  - *src* contains sources of native executables
  - *windows* contains pre-compiled Windows binaries
  - *compilebin.sh* the script to complile binaries
  - *setexecflag.sh* the script to set executable flag for the binaries
- *conf* contains JABAWS configuration files
- *ExecutionStatistics* the database for storing collected execution statistics
- *jobsout* a default folder for temporary job directories
- *statpages* the web pages for execution statistics display
- *WEB-INF* default
- *docs* contains the reStructuredText documentation files
- *website* contains the JABAWS web pages
  - *archive* contains JABAWS packages, the WAR and JAR files

datamodel contains the JABAWS datamodel

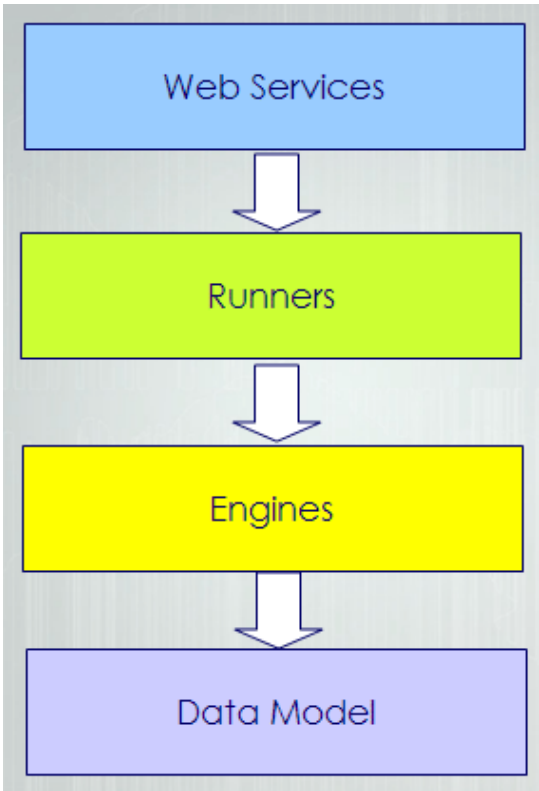
engine contains the JABAWS engine - the code that abstract the execution environment and executes native binaries

runner contains the JABAWS runners - thin wrappers for native binaries

webservices contains the JABAWS SOAP web services

testsrc contains the JABAWS unit tests

8.4 The code structure



Each source folder depends on the upper folders for compilation. For example, the datamodel is the top level folder so it has no other dependencies on other JABAWS code. The Engine level depends on the datamodel to compile etc. The web services folder is the bottom layer and depends on all the other source code.

So the JABAWS project is split into 4 layers. From bottom-up the first layer consists from the value classes used by all other layers of the hierarchy, in particular web services. So, to be able to use JABAWS one needs to have these classes. At the same time classes on this layer does not have any dependencies on the layers above.

The second layer contains code for execution of the wrappers, which are the abstraction describing native executables. JABAWS can execute tasks locally that is on the same machine as JVM and on the cluster. Thus currently code on this layer contain two engines. This layer depends on the layer underneath, the data model layer, but is completely independent from the code above.

The third layer consists of the wrappers for the native executables and classes to handle their configuration. It depends on the engines and the data model, but know nothing about the web services.

Finally, the upper layer contains the web services, that depend on all the layers below.

The layer isolation is archived though specially designed compilation task which is executed sequentially in several stages so that the first layer compiles before any other layers, second layer compiles after that and process continies before all the code is compiled. Any violation of the layer boundaries results in the compilation failure. Use Ant "Compile" or "Complile_with_debug" tasks to perform the staged compilation.

A client package contains only classes from data model layer and a simple web services client. Framework package is for anyone who want to use JABAWS framework for controlling native executables in local or cluster environments. Framework exclude the web services layer. Server package contains all the code.

8.5 Unit Testing

JABAWS uses TestNG framework for testing. The test results for the JABAWS package offered for download can be found at: Test Results JABAWS uses TestNG for testing. There is a TestNG plugin available for Eclipse which has functionality similar to JUnit. However, no plugins are necessary to run the test cases, as testng jar is supplied with JABAWS together with an ant tasks to run the test cases.

Several testing groups are supported:

- All tests ('Test')
- Cluster tests ('Run_cluster_dependent_test')
- Cluster independent tests ('All_cluster_independent_tests')
- Windows only tests ('All_cluster_independent_windows_only_tests')
- Performance and stability tests ('Long_tests')
- Re-run failed tests ('Rerun_failed_tests')
- Run custom test ('CustomTest')

To run the tests you need to download all sources from repository. Once you have done that, enter into the command line mode, change directory to the project directory and type:

```
ant -f build.xml <test group name>
```

Make sure you have Apache Ant installed and path to ant executable is defined in your path environmental variable. Replace test group name with the one of the names given in the list above to run required group of tests e.g for running cluster only tests use the following command:

```
ant -f build.xml Run_cluster_dependent_test
```

If you work under Linux you could use a simple script from the root folder of repository called `runtests.sh`. This script simply contains a collection of the test commands described above and paths to java home directory and an ant executable, which you can define once for your system and then reuse.

A handy feature of TestNG is its ability to re-run failed tests. Failed test ant file is stored in `test-output/testng-failed.xml`. and is used in the ant task called `Rerun_failed_tests`. So re-running failed tests requires no more work than running any other test group and could be accomplished with the command:

```
ant -f build.xml Rerun_failed_tests
```

CustomTest runs the test defined in the project root directory file called `temp-testng-customsuite.xml`. This file is generated by TestNG plugin every time you run the test from Eclipse. Thus an easy way to run a test in a different environment is to run it from Eclipse first and then from ant using a custom test procedure.

For cluster execution make sure that the property `LD_LIBRARY_PATH` defined in build.xml points to cluster engine LD libraries directory in your local system.

8.6 Accessing JABAWS from your program

8.6.1 Web services functions overview

All JABAWS multiple sequence alignment web services comply to the same interface, thus the function described below are available from all the services.

Functions for initiating the alignment

```
String id = align(List<FastaSequence> list)
String id = customAlign(List<FastaSequence> sequenceList, List<Option> optionList)
String id = presetAlign(List<FastaSequence> sequenceList, Preset preset)
```

Functions pertaining to job monitoring and control

```
JobStatus status = getJobStatus(String id)
Alignment al = getResult(String id)
boolean cancelled = cancelJob(String id)
ChunkHolder chunk = pullExecStatistics(String id, long marker)
```

Functions relating to service features discovery

```
RunnerConfig rc = getRunnerOptions()
Limit limit = getLimit(String name)
LimitsManager lm = getLimits()
PresetManager pm = getPresets()
```

Please refer to a Data Model JavaDoc for a detailed description of each methods.

8.6.2 Structure of the template command line client

Packages	Classes and Interfaces
compbio.data.msa	MsaWS the interface for all multiple sequence alignment web services
compbio.data.sequence	JABAWS data types
compbio.metadata	JABAWS meta data types
compbio.ws.client	JABAWS command line client

Additional utility libraries that this client depend upon is the compbio-util-1.3.jar and compbio-annotation-1.0.jar.

Please refer to a Data Model JavaDoc for a detailed description of each class and its methods.

8.6.3 Connecting to JABAWS

For a complete working example of JABAWS command line client please see compbio.ws.client.Jws2Client class. JABAWS command line client source code is available from the download page. Please note that for now all the examples are in Java, other languages will follow if there is sufficient demand.

Download a binary JABAWS client. Add the client to the class path. The following code excerpt will connect your program to Clustal web service deployed in the University of Dundee.

```
import java.net.URL;
import javax.xml.namespace.QName;
import javax.xml.ws.Service;
// (...)
String qualifiedName = "http://msa.data.compbio/01/01/2010/";
URL url = new URL("http://www.compbio.dundee.ac.uk/jabaws/ClustalWS?wsdl");
QName qname = new QName(, "ClustalWS");
Service serv = Service.create(url, qname);
MsaWS msaws = serv.getPort(new QName(qualifiedName, "ClustalWSPort"),
MsaWS.class);
```

Line 1 makes a qualified name for JABA web services.

Line 2 constructs the URL to the web services WSDL.

Line 3 makes a qualified name instance for Clustal JABA web service.

Line 4 creates a service instance.

Line 5 makes a connection to the server.

A more generic connection method would look like this

```
import java.net.URL;
import javax.xml.namespace.QName;
import javax.xml.ws.Service;
import compbio.ws.client.Services
// (...)
String qualifiedServiceName = "http://msa.data.compbio/01/01/2010/";
String host = "http://www.compbio.dundee.ac.uk/jabaws";
// In real life the service name can come from args
Services clustal = Services.ClustalWS;
URL url = new URL(host + "/" + clustal.toString() + "?wsdl");
QName qname = new QName(qualifiedServiceName, clustal.toString());
Service serv = Service.create(url, qname);
MsaWS msaws = serv.getPort(new QName(qualifiedServiceName, clustal + "Port"), MsaWS.
↪class);
```

Where Services is enumeration of JABAWS web services. All JABAWS multiple sequence alignment methods confirm to MsaWS specification, thus from the caller point of view all JABAWS web services can be represented by MsaWS interface. The full documentation of MsaWS functions is available from the JavaDoc.

8.6.4 Aligning Sequences

Given that *msaws* is web service proxy, created as described in "Connecting to JABAWS" section, the actual alignment can be obtained as follows:

```
List<FastaSequence> fastalist = SequenceUtil.readFasta(new FileInputStream(file));
String jobId = msaws.align(fastalist);
Alignment alignment = msaws.getResult(jobId);
Line one loads FASTA sequence from the file.
```

Line two submits them to web service represented by msaws proxy.

Line three retrieves the alignment from a web service. This line will block the execution until the result is available. Use this with caution. In general, you should make sure that the calculation has been completed before attempting retrieving results. This is to avoid keeping the connection to the server on hold for a prolonged periods of time. While this may be ok with your local server, our public server (www.compbio.dundee.ac.uk/jabaws) will not let you hold the connection for longer than 10 minutes. This is done to prevent excessive load on the server. The next section describes how to check the status of the calculation. Methods and classes mentioned in the excerpt are available from the JABAWS client library.

8.6.5 Checking the status of the calculation

You may have noticed that there was no pause between submitting the job and retrieving of the results. This is because getResult(jobId) method block the processing until the calculation is completed. However, taking into account that the connection holds server resources, our public server (www.compbio.dundee.ac.uk/jabaws) is configured to reset the connection after 10 minutes of waiting. To work around the connection reset you are encouraged to check whether the calculation has been completed before accessing the results. You can do it like this:

```
while (msaws.getJobStatus(jobId) != JobStatus.FINISHED) {
    Thread.sleep(2000); // wait two  seconds, then recheck the status
}
```

8.6.6 Aligning with presets

```
PresetManager presetman = msaws.getPresets();
Preset preset = presetman.getPresetByName(presetName);
List<FastaSequence> fastalist = SequenceUtil.readFasta(new FileInputStream(file));
String jobId = msaws.presetAlign(fastalist, preset);
Alignment alignment = msaws.getResult(jobId);
```

Line one obtains the lists of presets supported by a web service.

Line two return a particular Preset by its name.

Lines three to five are doing the same job as in the first aligning sequences example.

8.6.7 Aligning with custom parameters

```
RunnerConfig options = msaws.getRunnerOptions();
Argument matrix = options.getArgument("MATRIX");
matrix.setValue("PAM300");
Argument gapopenpenalty = options.getArgument("GAPOPEN");
```

```
gapopenpenalty.setValue("20");
List<Argument> arguments = new ArrayList<Argument>();
arguments.add(matrix); arguments.add(gapopenpenalty);
List<FastaSequence> fastalist = SequenceUtil.readFasta(new FileInputStream(file));
String jobId = msaws.customAlign(fastalist, arguments);
Alignment alignment = msaws.getResult(jobId);
```

Line one obtains the `RunnerConfig` object that holds information on supported parameters and their values

Line two retrieve a particular parameter from the holder by its name.

Lines three sets a value to this parameter which will be used in the calculation.

Line four and five do the same but for another parameter.

Line six makes a List to hold the parameters.

Line seven puts the parameters into that list.

Line eight and ten is the same as in previous examples.

Line nine submit an alignment request with the sequences and the parameters.

The names of all the parameters supported by a web service e.g. "PAM300" can be obtained using `options.getArguments()` method. Further details on the methods available from `RunnerConfig` object are available from the JavaDoc.

8.6.8 Writing alignments to a file

There is a utility method in the client library that does exactly that.

```
Alignment alignment = align(...)
FileOutputStream outStream = new FileOutputStream(file);
ClustalAlignmentUtil.writeClustalAlignment(outStream, align);
```

8.6.9 A complete client example

Finally, a complete example of the program that connects to JABAWS Clustal service and aligns sequences using one of the Clustal web service presets. All you need for this to work is a JABAWS CLI client. Please make sure that the client is in the Java class path before running this example.

```
import java.io.ByteArrayInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.net.URL;
import java.util.List;

import javax.xml.namespace.QName;
import javax.xml.ws.Service;

import compbio.data.msa.MsaWS;
import compbio.data.sequence.Alignment;
import compbio.data.sequence.FastaSequence;
import compbio.data.sequence.SequenceUtil;
```

```
import compbio.metadata.JobSubmissionException;
import compbio.metadata.LimitExceededException;
import compbio.metadata.Preset;
import compbio.metadata.PresetManager;
import compbio.metadata.ResultNotAvailableException;
import compbio.metadata.UnsupportedRuntimeException;
import compbio.metadata.WrongParameterException;

public class Example {

  /*
   * Input sequences for alignment
   */
  static final String input = ">Foo\r\n"
            + "MTADGPRELLQLRAAVRHRPQDFVAWLMLADAELGMGDTTAGEMAVQRGLALHPGHPEAVARLGR"
            + "VRWTQQRHAEAAVLLQQASDAAPEHPGIALWLGHALEDAGQAEAAAAAYTRAHQLLPEEPYITAQ"
            + "LLNWRRRLCDWRALDVLSAQVRAAVAQGVGAVEPFAFLSEDASAAEQLACARTRAQAIAASVRPL"
            + "APTRVRSKGPLRVGFVSNGFGAHPTGLLTVALFEALQRRQPDLQMHLFATSGDDGSTLRTRLAQA"
            + "STLHDVTALGHLATAKHIRHHGIDLLFDLRGWGGGGRPEVFALRPAPVQVNWLAYPGTSGAPWMD"
            + "YVLGDAFALPPALEPFYSEHVLRLQGAFQPSDTSRVVAEPPSRTQCGLPEQGVVLCCFNNSYKLN"
            + "PQSMARMLAVLREVPDSVLWLLSGPGEADARLRAFAHAQGVDAQRLVFMPKLPHPQYLARYRHAD"
            + "LFLDTHPYNAHTTASDALWTGCPVLTTPGETFAARVAGSLNHHLGLDEMNVADDAAFVAKAVALAS"
            + "DPAALTALHARVDVLRRESGVFEMDGFADDFGALLQALARRHGWLGI\r\n"
            + "\r\n"
            + ">Bar\r\n"
            + "MGDTTAGEMAVQRGLALHQQRHAEAAVLLQQASDAAPEHPGIALWLHALEDAGQAEAAAAAYTRAH"
            + "QLLPEEPYITAQLLNAVAQGVGAVEPFAFLSEDASAAESVRPLAPTRVRSKGPLRVGFVSNGFGA"
            + "HPTGLLTVALFEALQRRQPDLQMHLFATSGDDGSTLRTRLAQASTLHDVTALGHLATAKHIRHHG"
            + "IDLLFDLRGWGGGGRPEVFALRPAPVQVNWLAYPGTSGAPWMDYVLGDAFALPPALEPFYSEHVL"
            + "RLQGAFQPSDTSRVVAEPPSRTQCGLPEQGVVLCCFNNSYKLNPQSMARMLAVLREVPDSVLWLL"
            + "SGPGEADARLRAFAHAQGVDAQRLVFMPKLPHPQYLARYRHADLFLDTHPYNAHTTASDALWTGC"
            + "PVLTTPGETFAARVAGSLNHHLGLDEMNVADDAAFVAKAVALASDPAALTALHARVDVLRRESGV"
            + "FEMDGFADDFGALLQALARRHGWLGI\r\n"
            + "\r\n"
            + ">Friends\r\n"
            + "MTADGPRELLQLRAAVRHRPQDVAWLMLADAELGMGDTTAGEMAVQRGLALHPGHPEAVARLGRV"
            + "RWTQQRHAEAAVLLQQASDAAPEHPGIALWLGHALEDHQLLPEEPYITAQLDVLSAQVRAAVAQG"
            + "VGAVEPFAFLSEDASAAEQLACARTRAQAIAASVRPLAPTRVRSKGPLRVGFVSNGFGAHPTGLL"
            + "TVALFEALQRRQPDLQMHLFATSGDDGSTLRTRLAQASTLHDVTALGHLATAKHIRHHGIDLLFD"
            + "LRGWGGGGRPEVFALRPAPVQVNWLAYPGTSGAPWMDYVLGDAFALPPALEPFYSEHVLRLQGAF"
            + "QPSDTSRVVAEPPSRTQCGLPEQGVVLCCFNNSYKLNPQSMARMLAVLREVPDSVLWLLSGPGEA"
            + "DARLRAFAHAQGVDAQRLVFMPKLPHPQYLARYRHADLFLDTHPYNAHTTASDALWTGCPVLTTP"
            + "GETFAARVAGSLNHHLGLDEMNVADDAAFVAKAVALASDPAALTALHARVDVLRRESI";

  public static void main(String[] args) throws UnsupportedRuntimeException,
            LimitExceededException, JobSubmissionException,
            WrongParameterException, FileNotFoundException, IOException,
            ResultNotAvailableException, InterruptedException {

            String qualifiedServiceName = "http://msa.data.compbio/01/01/2010/";

            /* Make a URL pointing to web service WSDL */
            URL url = new URL("http://www.compbio.dundee.ac.uk/jabaws/ClustalWS?wsdl
↪");

            /*
             * If you are making a client that connects to different web services
             * you can use something like this:
```

```
             */
            // URL url = new URL(host + "/" + Services.ClustalWS.toString() +
            // "?wsdl");

    QName qname = new QName(qualifiedServiceName, "ClustalWS");
    Service serv = Service.create(url, qname);
    /*
     * Multiple sequence alignment interface for Clustal web service
     * instance
     */
    MsaWS msaws = serv.getPort(new QName(qualifiedServiceName, "ClustalWS"
                    + "Port"), MsaWS.class);

    /* Get the list of available presets */
    PresetManager presetman = msaws.getPresets();

    /* Get the Preset object by preset name */
    Preset preset = presetman
                    .getPresetByName("Disable gap weighting (Speed-oriented)");

    /*
     * Load sequences in FASTA format from the file You can use something
     * like new FileInputStream(<filename>) to load sequence from the file
     */
    List<FastaSequence> fastalist = SequenceUtil
                    .readFasta(new ByteArrayInputStream(input.getBytes()));

    /*
     * Submit loaded sequences for an alignment using preset. The job
     * identifier is returned by this method, you can retrieve the results
     * with it sometime later.
     */
    String jobId = msaws.presetAlign(fastalist, preset);

    /* This method will block for the duration of the calculation */
    Alignment alignment = msaws.getResult(jobId);

    /*
     * This is a better way of obtaining results, it does not involve
     * holding the connection open for the duration of the calculation,
     * Besides, as the University of Dundee public server will reset the
     * connection after 10 minutes of idling, this is the only way to obtain
     * the results of long running task from our public server.
     */
    // while (msaws.getJobStatus(jobId) != JobStatus.FINISHED) {
    // Thread.sleep(1000); // wait a second, then recheck the status
    // }

    /* Output the alignment to standard out */
    System.out.println(alignment);

    // Alternatively, you can record retrieved alignment into the file in
    // ClustalW format

    // ClustalAlignmentUtil.writeClustalAlignment(new FileOutputStream(
    // "output.al"), alignment);

}
```

```
}
```

For a more detailed description of all available types and their functions please refer to the Data Model JavaDoc.

8.7 Adding new web-services

8.7.1 Brief Guide

1. Add a new executable which you'd like to wrap as a JABAWS web service to the binaries folder. If it has the source code and can be recompiled for different platforms include it under `binaries/src`. Edit `setexecutableflag.sh` and `compilebin.sh` scripts in `binaries/src` accordingly.
2. Make sure that all the dependencies of the software being installed are satisfied. If there are other binaries they should be included as well. Keep the dependent binaries in a subfolder for the main executable. Update `compilebin.sh` and `setexecflag.sh` scripts accordingly.
3. Make sure that the new executable does not have any hard links to its dependencies, e.g. is able to run from any installation folder and does not contain any hard coded paths.
4. Describe executable in `conf/Exectuable.properties` file. The lowercase name of the wrapper should be included in the name of the property for example Clustal properties all include clustal as a part of the name e.g. `local.clustalw.bin`. The same property for MAFFT will be called `local.mafft.bin`. For more help please refer to the Executable.properties file.
5. Describe the executable supported parameters in the `<ExecutableName>Parameters.xml`, presets in the `<ExecutableName>Presets.xml` and the execution limits in the `<ExecutableName>Limit.xml`. By convention these files are stored in `conf/settings`. All of these are optional. If the executable does not support parameters you do not have to mention the `XXXParameter.xml` file in the `Executable.properties` file at all. The same is true for Presets and Limits.
6. Create a Java wrapper class for your executable. Create it within runner source directory. Examples of other wrappers can be found in `compbio.runner.msa` or in other `compbio.runner.*` packages. Wrapper should extend `SkeletalExecutable<T>` and implement `PipedExecutable<T>` if you need to pass the input or collect the results from the standard in/out. Please see Mafft code as example. Wrapper should expend `SkeletalExecutable<T>` if input/output can be set as a parameter for an executable. Please see the ClustalW code as example.
7. Create a testcase suit for your wrapper in `testsrc` and run the test cases.
8. Create parser for the output files of your executable. Suggested location `compbio.data.sequence.SequenceUtil`.
9. Test the parser.
10. Decide which web services interfaces your executable is going to match. For example if the executable output can be represented as SequenceAnnotation then SequenceAnnotation interface might be appropriate. For multiple sequence alignment an Msa interface should be used.
11. If you find a web interface that matches your returning data type, then implement a web service which confirms to it within a webservices source folder.
12. Register web service in `WEB-INF/web.xml` and `WEB-INF/sun-jaxws.xml`.
13. Add generated wsdl to wsbuild.xml ant script to generate the stubs.
14. Run build-server task in wsbuild file. Watch for errors. If the task fails that means that JAXB cannot serialize some of your new data structures. Add appropriate annotations to your data types. Also check that:

- you do not have interfaces to serialize, since JAXB cannot serialize them
- you have a default no args constructor (can be private if you do not need it)
- JAXB cannot serialize Java Map class, use a custom data structure instead
- Enum cannot be serialized as its abstract class (do not confuse with enum which is fine)
- Fields serialization leaves a little more space for manoeuvre. If you do this then you may accept and return interfaces, e.g. List, Map; abstract classes etc, from your methods

If you have the data on the server side, but nothing is coming through to the client, this is a JAXB serialization problem. They tend to be very silent and thus hard to debug. Check your data structure can be serialized!

15. Modify the client to work with your new web service. Update Services enumeration to include new service and ensure that all the methods of this enumeration take into account the new service. Update the client help text (`client_help.txt`) and insert it into the Constraints class.
16. Test the web service with the client.
17. Test on the cluster.

8.7.2 Building web services artifacts

JABAWS are the standard JAX-WS SOAP web services, which are WS-I basic profile compatible. This means that you could use whatever tool your language has to work with web services. Below is how you can generate portable artifacts to work with JABAWS from Java. However if programming in Java, we recommend using our client library as it provides a handful of useful methods in addition to plain data types.

```
wsimport -keep http://www.compbio.dundee.ac.uk/jabaws/ClustalWS?wsdl
```

Server side artifacts should be rebuild whenever the data model, meta model or MSA interface were changed. To do that run build-server task from wsbuild.xml ant build file. WSDL files will be generated in `webservices/compbio/ws/server/resource` directory. It is not necessary to edit them if any of the JABAWS clients are used. JABAWS are the standard JAX-WS web services, which are WS-I basic profile compatible.

8.7.3 Preparing Distributives

There are a number of ant tasks aimed for preparing distributives for download. Currently a few types of JABAWS packages are offered:

1. Client only (contains classes required to access JABA Web Services)
2. Platform specific JABAWS (windows and other)
3. JABAWS with and without binaries
4. JABAWS framework and complete project

The easiest way to build all distributives is to call `build-all` ant task. There are more tasks defined in build.xml than described here. They are mostly self explanatory.

If you made any changes to the data model and would like to generate a complete JABAWS distro make sure you have rebuilt jaxws artifact as described below.

CHAPTER

NINE

USAGE STATISTICS

JABAWS comes with a web application for visualizing usage statistics. The screenshot below shows the main page of this application. The individual month is linked to detailed usage statistics (described later). Please note, that the links to the detailed monthly statistics are only available for authenticated users in the role admin. There is a link at the bottom of the page that lets you login, if you have not done so.

If you are using JABAWS VA (Virtual Appliance) then the username is *jabaws* and password is not defined, i.e. empty.

If you have deployed a JABAWS WAR file, then please see the configuring privileged access for Tomcat web application server section for further details.

The table contains the number of jobs processed by JABAWS per month, for the whole period when the statistics was collected.

For each month the table contains the following information.

- Month - the period of time for which statistics is displayed. For example Jan 2011 means period of time from the first of January to the first of February
- Total - the total number of jobs accepted by JABAWS
- Incomplete - the number of jobs for which the result file was not found or was empty excluding cancelled
- Cancelled - the number of jobs cancelled by the user
- Abandoned - the number of jobs which result(s) were not collected

The summary for each column is displayed in the last row of the table.

9.1 Detailed View

Detailed execution statistics for each month is available for authenticated users only.

Each table contains the number of jobs processed by JABAWS during the period of time specified in the title:

- The “All Jobs” table contains the summary of all jobs
- “Local Jobs” table - contains the summary of the jobs calculated by the local engine
- “Cluster Jobs” table - contains the summary of the jobs calculated by the cluster

Each table contains the following information for each web service:

- Total - the total number of jobs accepted by a particular JABA service

- Incomplete - the number of jobs for which the result file was not found or was empty excluding cancelled
- Cancelled - the number of jobs cancelled by the user
- Abandoned - the number of jobs which result(s) were not collected

---

9.2 Job List

Please note that if you deployed JABAWS WAR, in order to be able to navigate to the job directory from this view, the application server may need to be configured. Please see the Configuring JABAWS execution statistics section for further details.

Columns:

- JobID - the JABAWS job id, unique for every job
- Cluster JobID - cluster job id
- InputSize - input size in bytes
- ResultSize - result size in bytes
- Runtime (s) - job's runtime in seconds
- Start time (s)- job's start time and date
- Finish time (s)- job's finish time and date
- isCancelled - whether the job was cancelled
- isCollected - whether the job was collected. False for the jobs that has been initiated but which results has never been retrieved
- isFinished - whether the job has finished. This does not necessarily mean that the job has produced the result. The job can sometime finish in failure

---

9.3 Job Directory Contents

STARTED and FINISHED files contain Unix timestamp - when the job was started and completed respectively. STARTED is replaced by SUBMITTED if the job has been submitted to the cluster, as opposed to executed locally, on the server.

COLLECTED file is empty and indicates that the job results were collected by the user. Due to asynchronous nature of the job it is possible that the job was started and finished, but the results has never been requested.

RunnerConfig.xml file contains a complete description of the job and JABAWS can restart the job based on this description.

procError.txt and procOutput.txt files contains the content of the standard out and standard error streams of the process.

result.txt file contains the results.

input.txt file contains input into the process.

There are maybe other files depending on the nature of the job, but the one described above will be present in most cases. In this example, stat.log file stories the execution statistics generated by (clustal executable in this example) process.

If you have deployed JABAWS WAR file or made changes to JABAWS configuration you may need to make a few changes to the Tomcat configuration to be able to see the content of the job directory. Please see the Configuring JABAWS execution statistics section for further details.

9.4 Configuring JABAWS execution statistics

JABAWS execution statistics is a multi-component system. First is a crawler whose job is to collect and preprocess the statistics from the job temporary directories and record the collected statistics into the database. The second part of the system is a web application whose job is to visualise the statistics from the database.

It is possible to enable/disable the statistics collector by changing the following properties in the `conf/Cluster.engine.properties` and `conf/Local.engine.properties` files.

```
# Enable/disable cluster statistics collector true = enable, false = disable
cluster.stat.collector.enable=false
# Maximum amount of time the job is considered be running in hours. Optional defaults␣
↪to 7 days (168h)
cluster.stat.maxruntime=24
```

```
# Enable/disable cluster statistics collector true = enable, false = disable
local.stat.collector.enable=true
# Maximum amount of time the job is considered to be running in hours. Optional␣
↪defaults to 24 hours
local.stat.maxruntime=6
```

If the statistics collector is enabled then the crawler starts automatically soon after (10 minutes for local engine, and 60 minutes for cluster engine) the JABAWS web application and will be collecting the execution statistics every 24 hours after the start.

The details of the job are only available if the job temporary directory is located within a JABAWS web application. If not, the system administrator can create a symbolic link pointing to the temporary job directories outside of a web application and configure the application server to allow navigation to the links. For the Tomcat application server the context configuration file should be created and copied to the `<TOMCAT_ROOT>/conf/Catalina/localhost` directory. The name of the file should be the same as the web application context name, for example *jabaws.xml* for jabaws. Where the `TOMCAT_ROOT` is the location of the Tomcat web application server. Here is an example of such a file:

```
<?xml version="1.0" encoding="UTF-8"?>
<Context antiResourceLocking="false" privileged="true" allowLinking="true"/>
```

The key option here is this: `allowLinking="true"`. Please also make sure that you have defined the user in role *admin* as described below.

9.5 Configuring a privileged access for Tomcat web application server

Access to configuration files, detailed job execution statistics and job directories are allowed only for authenticated users in role *admin*.

If you use Tomcat, then the simplest way to set up privileged access is to use a plain text configuration file `conf/tomcat-user.xml`. Here is an example of such configuration file defining user "peter" in role *admin*.

```
<tomcat-users>
<role rolename="admin"/>
<user username="peter" password="your password here " roles="admin"/>
</tomcat-users>
```

For more information on users and roles please consult Apache-Tomcat help pages.

CHAPTER

TEN

CITATIONS

Note: It is important that you cite JABAWS when you use it. Citing us helps us funding the work we do and allow us to continue to improve the project further.

Peter V. Troshin, James B. Procter and Geoffrey J. Barton - **Java Bioinformatics Analysis Web Services for Multiple Sequence Alignment - JABAWS:MS** Bioinformatics 2011. 27 (14): 2001-2002. doi: 10.1093/bioinformatics/btr304

Peter V. Troshin, Alexander Sherstnev, James B. Procter, Daniel L. Barton, Fábio Madeira and Geoffrey J. Barton (2017) **JABAWS 2.2 Distributed Web Services for Bioinformatics: Protein Disorder, Conservation and RNA Secondary Structure.** *Paper in preparation.*

CHAPTER

ELEVEN

CHANGELOG

11.1 Version 2.2 (Released 18 August 2017)

The website and documentation were improved:

- Sphinx is now used to generate our documentation pages.
- Documentation was updated to reflect the latest changes introduced in the project.
- Usage Statistics are now cached and refreshed every hour for improved website loading times.
- Service Status are now similarly cached and monitored every 10 minutes.
- We now provide a Docker image which allows for JABAWS to be run in Docker containers.
- Downloading the JABAWS distributions no longer require 'sign in' or 'sign up' to a user account.
- The pre-configured JABAWS Amazon Machine Image (AMI), which allowed for JABAWS to be run in the Amazon EC2 cloud, is no longer provided due to very limited use by the scientific community peers.

The versions of several application programs provided by JABAWS were bumped to the latest available.

- Clustal Omega was updated to version 1.2.4
- ClustalW was updated to version 2.1
- Mafft was updated to version 7.3.10
- T-Coffee was updated to version 11.00.8cbe486
- AACon was updated to version 1.1
- Protein secondary structure prediction with Jpred (version 3.0.3) was dropped from the list of provided services, as the use of the dedicated Jpred REST API (Jpred 4) is encouraged and recommended. This is the version that is currently provided within Jalview 2.9 or later.

Note: JABAWS version 2.2 is fully backward compatible with JABAWS v1.0 and v2.0. This means all JABAWS 1.0, 2.0, 2.0.1 and 2.1 clients should also be able to use JABAWS 2.2 services.

11.2 Version 2.1 (Released 1st Oct 2013)

Several new web services are available in this version of JABAWS:

- Two multiple sequence aligners (MSAprobs and GLprobs), both services return the standard Alignment object

- RNAalifoldWS returns RNAStructScoreManager, which is the standard ScoreManager objects with several additional methods
- JpredWS returns the JpredAligment object, which is the standard alignment with additional methods for extracting Jpred predictions. These predictions are supplied as additional sequences in the aligment

Some bugs have been fixed and several improvements have been done:

- WS status servlet returns version and some additional information on each web service
- a bug with path to help in the client
- Fix two bug with the Google Analytics library: no-stop due to running thread
- GoogleAnalytics gets proper JABAWS version

11.3 Version 2.0.1 (Released 2nd Jul 2013)

JABAWS 2.0.1 includes several bug fixes and minor updates for JABAWS Version 2.0. These are listed below:

- Disembl returned swapped strings for HOTLOOPS and REM465
- Jronn failed to process jobs with more than 3 sequences
- JABAWS could not deal with FASTA records with '>' symbols in the record identificator
- Change of parameter description for AACon: parameters have been replaced with options for calculation methods. This allows a user to get several AACon's conservation scores in one call
- JABAWS never cleaned up job directories. Now JABAWS deletes the job directory if it exist longer than a period defined in Engine.properties
- Default web security has been incompatible with Tomcat 7.0.31 and newer
- Documentation has been updated

11.4 Version 2 (Released 16th Dec 2011)

Compared to JABAWS 1, JABAWS 2 offers a greater number and diversity of web services, Amazon EC2 integration and improved ease of use.

It now contains:

- Updates for all multiple sequence alignment services
- Four new protein disorder prediction services
- Clustal Omega multiple sequence alignment web service
- Amino acid conservation service
- Web services execution statistics visualization
- Web services status check from a web page
- VirtualBox support was dropped in favour of VMware
- New WAR package for Mac users

- Amazon Machine Image (AMI) distributive to enable users to use JABAWS on the EC2 cloud
- Improved web services client API
- Simplified WAR package installation

Warning: To access the analysis web services introduced in JABAWS 2.0, clients that were designed for JABAWS v1.0 must be updated.

Note: This is an open source project. If you want to contribute or report an issue have a look at our Git-tracker.