Fast Exploration of the QSAR Model Space with e-Science Central and Windows Azure

Jacek Cała, Hugo Hiden, Simon Woodman, Paul Watson
School of Computing Science
University of Newcastle upon Tyne
United Kingdom

Extended Abstract

Quantitative Structure Activity Relationship (QSAR) is a method to create mathematical models that can predict biological activity of compounds from their chemical structure [1]. Chemists use QSAR models to focus synthesis of new compounds, to design better, safer drugs, as well as more environmentally benign products.

In the search for highly predictive QSAR models a number of approaches has been applied from straightforward techniques, like using multiple linear regression, to using specialized modelling environments offering users variety of data mining tools [2][3], to automated systems that can apply various modelling techniques for the same input and select the most promising models [4][5]. The automated QSAR is an attractive approach to accelerate building predictive models because it can quickly explore and assess thousands of models with no need for human intervention. It is especially important as new, large databases of chemical compounds have recently become available, covering many different types of biological targets and activities.

Overall, QSAR is a specialized machine learning process (Figure 1). As input it receives a set of pairs relating molecular structure of a compound with its activity against a particular biological target. The input set is split into a training and testing subset. Then, a set of molecular descriptors that numerically represent certain properties of the compound (e.g. molecular weight) are calculated. Next, the relevant descriptors are selected and used by model-building algorithms (e.g. multiple linear regression, neural networks) to create mathematical models. Finally, the produced models are tested and those with high prediction capabilities are stored for later use.

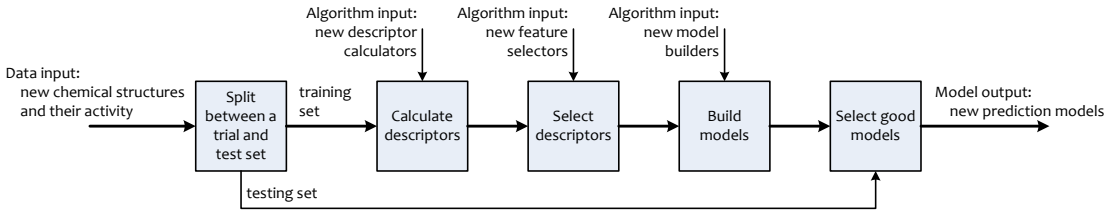

Figure 1. A general view on the QSAR modelling process.

Technical Challenge. Building useful QSAR models requires substantial computational resources. The most computing intensive stages are: (1) calculating molecular descriptors, which can produce hundreds or even thousands of parameters for each compound, (2) filtering descriptors to remove redundant or irrelevant features so as to speed up learning and increase the generalisability of the results, and (3) using certain model building algorithms like neural networks which can take over an hour to produce an output.

With the availability of large databases of chemical compounds, such as ChEMBLdb[1] that provides over one million of small molecule structures with corresponding activities, exploring QSAR model space quickly becomes workload of CPU-year size. And even if the process can build thousands of models, only a small percentage of them is valid and useful for further QSAR prediction. Therefore, finding an efficient way to perform the modelling will enable us to experiment with different descriptor calculation, feature selection and model building algorithms in the future.

Method. Fortunately, the cloud computing approach very well fits the presented problem. The modelling process is a combination of the task- and data-based parallelism and can be effectively run on a cluster of machines. Moreover, after initial processing of the whole ChEMBLdb database with available model building algorithms, further efforts with QSAR analysis will require much less resources.

We based the design of our QSAR modelling tool on the Discovery Bus — a system that implements an auto-QSAR best practice modelling workflow [4]. However, our previous work showed that the use of the Discovery Bus to build QSAR models was of limited scalability [6]. For this reason we decided to model the original drug discovery workflows using our e-Science Central platform (e-SC) and process them with e-SC workflow engines running in Azure. e-Science Central is a cloud-based workflow enactment system [7]. It uniquely combines the Software-as-a-Service approach with social networking and cloud-computing, all to support scientists in designing, running and sharing their analyses on large scale while being freed from most of the burden of software maintenance.

For evaluation purposes we used Windows Azure to run a complete copy of the publicly available e-SC system.[2] Figure 2 shows the architecture and deployment of the system in the cloud. The central server executes in two Azure VM instances — one hosts e-SC frontend on top of a JEE application server while the other runs the database engine. Machines have been started using a single deployment package. This allowed us to connect them directly via JDBC/TCP and to avoid the

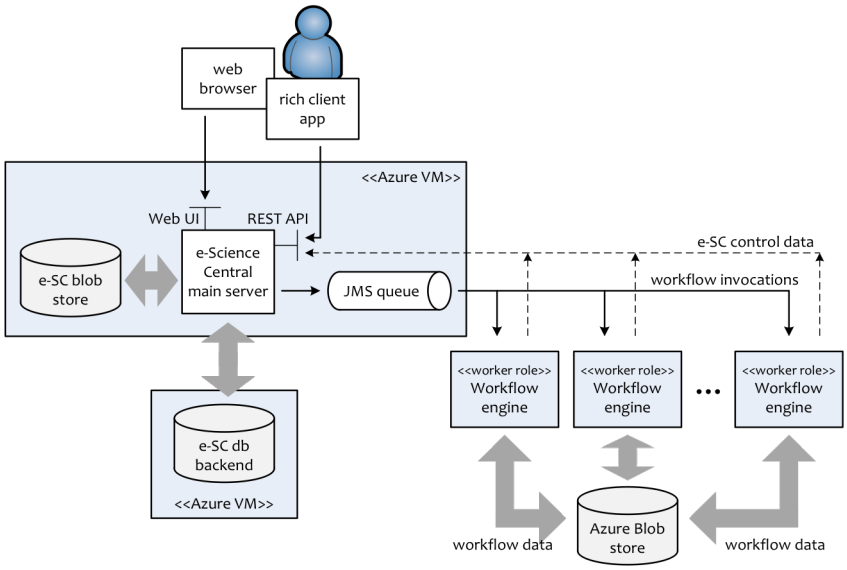

Figure 2. The architecture of e-Science Central deployed in Windows Azure.

[1] https://www.ebi.ac.uk/chembl
[2] http://www.esciencecentral.co.uk

Azure load balancer between the server and database, which is redundant in our case.

Separately, a number of e-SC workflow engines is running, each in its own Azure worker role node. The engines are connected with the server by a JMS message queue and the REST-based API. Despite Azure offers its own queuing service, to preserve as much consistency with the original e-SC implementation as possible we did not decide to switch the queue services. There is no obvious benefit for such a change. Conversely, as shown by Hill et al. [8], reading rates for the Azure queue service when accessed by a large number of clients can drop to as little as 2 messages per second per client. Their observation is in line with the scalability target of 500 messages per second for a single queue as indicated by Calder [9].

To implement our QSAR scenario we built 12 workflows corresponding to the process presented in Figure 1. Users submit their workflows via a web browser or dedicated desktop application. The submission is accepted by the server which creates for it a workflow invocation. The invocation comprises a sequence of service (block) calls. These are either core services available within e-Science Central or custom blocks that the scientist has uploaded. e-SC supports execution of various service kinds such as Java, R and Octave. They can be as simple as downloading data from blob storage or as complex as building a QSAR model which can consume over one CPU-hour. System also offers control blocks that can initiate subsequent workflow invocations, and so create invocation chains, trees or even loops. Importantly, workflow invocations are completely independent of each other and may be processed by any of the workflow engines.

All created workflow invocations are sent to a single message queue from which they are acquired by the engines. Adoption of the work-stealing approach rather than explicit task scheduling, better fits the Windows Azure platform for at least two reasons. First, worker nodes may be restarted or taken offline anytime during their operation. This may happen either because of node failure or due to service healing or automatic upgrade of the OS. Second, the global invocation queue facilitates adding nodes to and removing them from the resource pool. There is no need for rescheduling tasks when the pool size changes.

When a workflow invocation is executed, the engine runs the included blocks one by one according to the structure of the flow of data. The definition of a block contains not only the declaration of input ports which the block requires to run but also software dependencies that must be met to start it. For example, a number of blocks in our QSAR scenario need the R runtime environment, and so this requirement is expressed in the block descriptor as a library dependency. Before running a service, any unavailable libraries are downloaded from the server on demand. Once all software dependencies are met, the engine starts executing a service. To improve security and reliability every block execution involves creation of a dedicated process in the operating system. In the case of Java blocks it is a JVM process, while for R blocks R runtime environment is started.

The overall result of a workflow invocation is sent back to the server as a simple status message (success or failure). Additionally, the server creates for each invocation a dedicated folder where all invocation specific data may be stored; to transfer them e-SC offers a number of I/O blocks.

Evaluation. The evaluation of the presented system was run in the Windows Azure platform located in the Western Europe data centre. The server was hosted in two extra large Azure VM instances.

Workflow engines were deployed in 1–200 small instance worker role nodes. Input data for the evaluation purposes were selected from ChEMBLdb — a database of bioactive drug-like molecules.

Figure 3 presents the observed speed up in data processing in relation to the number of workers. As shown, our QSAR scenario scales nearly linearly up to 200 worker nodes. The observed speed up for 200 workers was 88.2% of the ideal linear speed up when compared to 20 workers.

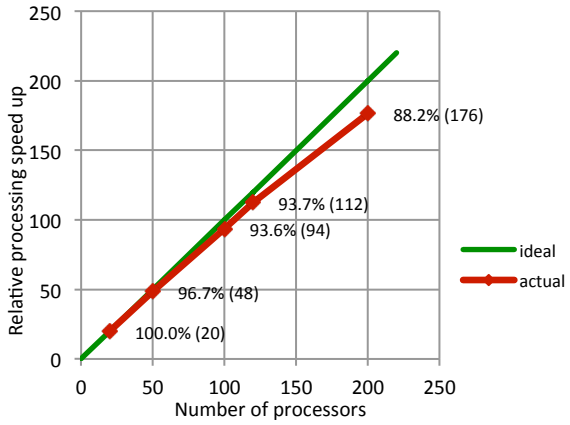

Figure 3. Speed-up in processing QSAR workflows in relation to the number of worker nodes.

Discussion. We presented a fast and scalable way to perform the exploration of the QSAR model space. The acceleration achieved is much beyond what existing solutions can offer. Overall, the cloud computing model is a very good fit for the presented scenario. After initial processing of the whole ChEMBLdb database with all available model building algorithms, further efforts with QSAR analysis will require much less resources. The database is regularly updated, thus we can extract several hundred new input datasets every three months. This is less than 10% of the current input size, and so it will need just a fraction of the infrastructure for less than 24 hours to be processed. Also, the development of new QSAR algorithms can be tested on a relatively small part of the input sets and for only the most promising ones the whole input data shall be applied.

When moving the QSAR analysis to the cloud, our main concern was on improving performance while increasing the number of running workflow engines. Several aspects were important in this respect. Firstly, crucial for effective operation of e-SC workflow engines was reducing the amount of data transferred between the server and the engines. For example, ability to share common software dependencies between workflow blocks was a simple yet effective way to reduce the amount of data transferred from the e-SC data repository to the engines. And with the increasing number of engines the saving are significant and can easily reach tens of gigabytes of data even if the dependencies are relatively small such as R runtime environment (less than 20 MB).

Secondly, a substantial gain in the number of processors working in parallel and overall processing effectiveness was achieved after changing the storage for intermediate data produced by workflows. We used the Azure blob storage that proved to be scalable enough to overcome a bottleneck related to communication with the central e-SC data repository. Switching to the blob storage was as simple as adding to the palette of existing e-SC blocks a few new I/O services (100–150 lines of Java code each) and changing existing I/O blocks in all related workflows.

Definitely, a valuable feature of our system is that the basic unit of work it relies on is a workflow rather than task invocation. Not only does it increase the run time of an invocation, which improves effectiveness, but also it allows for fast data transfer between the subsequent services. Unlike many other solutions based on task scheduling (e.g. Falcon, Condor and Pegasus; see [10][11] for an overview), blocks in our system communicate using local disk rather than shared file system; an important property for cloud-based solutions in which users also pay for network communication.

The current design of the system reaches scalability limitation at about 200 worker nodes. Running more workers causes overload to the data store VM, results in execution failures and lowers overall system performance. Whilst, for our QSAR use case 200 nodes gave more than satisfactory results, in the future we would like to remove this limitation.

References

[1] C. Hansch, A. Leo, and D. H. Hoekman, Exploring QSAR: Fundamentals and applications in chemistry and biology. American Chemical Society, 1995.

[2] C. Luscombe, "QSAR Workbench: Guided QSAR Model Building for nonExperts." The UKQSAR and ChemoInformatics Group, Cambridge, UK, 2011.

[3] J. C. Stålring, L. A. Carlsson, P. Almeida, and S. Boyer, "AZOrange - High performance open source machine learning for QSAR modeling in a graphical programming environment.," Journal of cheminformatics, vol. 3, no. 1, p. 28, Jan. 2011.

[4] J. Cartmell, S. Enoch, D. Krstajic, and D. E. Leahy, "Automated QSPR through Competitive Workflow.," Journal of computer-aided molecular design, vol. 19, no. 11, pp. 821–833, Nov. 2005.

[5] D. J. Wood, D. Buttar, J. G. Cumming, A. M. Davis, U. Norinder, and S. L. Rodgers, "Automated QSAR with a Hierarchy of Global and Local Models," Molecular Informatics, vol. 30, pp. 960–972, Nov. 2011.

[6] P. Watson et al., "Accelerating Chemical Property Prediction with Cloud Computing," in Microsoft Research eScience Workshop, 2010.

[7] P. Watson, H. Hiden, and S. Woodman, "e-Science Central for CARMEN: science as a service," Concurrency and Computation: Practice and Experience, vol. 22, no. 17, pp. 2369–2380, Dec. 2010

[8] Z. Hill, J. Li, M. Mao, A. Ruiz-Alvarez, and M. Humphrey, "Early Observations on the Performance of Windows Azure," in High Performance Distributed Computing, 2010, pp. 367–376.

[9] B. Calder, "Windows Azure Storage Abstractions and their Scalability Targets," Blog post: http://blogs.msdn.com/b/windowsazurestorage/archive/2010/05/10/windows-azure-storage-abstractions-and-their-scalability-targets.aspx, accessed 29/Feb/2012.

[10] E. Deelman, D. Gannon, M. Shields, and I. Taylor, "Workflows and e-Science: An overview of workflow system features and capabilities," Future Generation Computer Systems, vol. 25, no. 5, pp. 528–540, May 2008.

[11] I. Raicu et al., "Middleware support for many-task computing," Cluster Computing, vol. 13, no. 3, pp. 291–314, Apr. 2010.

Jacek Cała is a Research Associate in the School of Computing Science, Newcastle University, UK. His main interests are focused around adaptive, distributed and component-based systems. For the last three years he has been involved in projects closely related to cloud computing. While developing a system for automated QSAR in the cloud, he has been searching for scalable and efficient application architectures that can well fit the cloud approach.

Paul Watson is Professor of Computer Science and Director of the North East Regional e-Science Centre at Newcastle University, UK. In the 80's, as a Lecturer at Manchester University, he was a designer of the Alvey Flagship and Esprit EDS systems. From 1990-5 he worked for ICL as a system

designer of the Goldrush MegaServer parallel database server, which was released as a product in 1994. In August 1995 he moved to Newcastle University, where he has been an investigator on research projects worth over $40M. His research interests are in scalable information management. This includes data-intensive e-science, grid and cloud computing.