*Proceedings of the 2014 Winter Simulation Conference*
*A. Tolk, S. Y. Diallo, I. O. Ryzhov, L. Yilmaz, S. Buckley, and J. A. Miller, eds.*

# INVESTIGATING THE SPEEDUP OF SYSTEMS BIOLOGY SIMULATION USING THE SZTAKI DESKTOP GRID

Simon J. E. Taylor
Mohammadmersad Ghorbani

Modelling & Simulation Group
Brunel University
Uxbridge, UB8 3PH, UK

Navonil Mustafee

Centre for Innovation and Service Research
University of Exeter Business School
Exeter, EX4 4PU, UK

Tamas Kiss
Peter Borsody

Centre for Parallel Computing
University of Westminster
London, W1W 6UW, UK

Annette Payne
David Gilbert

Centre for Systems and Synthetic Biology
Brunel University
Uxbridge, UB8 3PH, UK

## ABSTRACT

Systems biology studies the complex interactions of biological and biochemical systems rather than their individual molecular components. System biology simulations can be embarrassingly parallel jobs that have no dependency among individual simulation instances, and thus lend themselves to parallel execution over distributed resources to reduce their overall execution time. One example of such distributed resources is a Desktop Grid for Volunteer Computing that aims to use vast numbers of computers to support scientific applications. The *SZTAKI Desktop Grid* (SZDG) uses a modified form of the volunteer computing software BOINC to implement an institution-wide Desktop Grid. This paper reports on experiences of porting the *SIMAP systems biology ODE simulator* to SZDG. A case study using a simulation of the *mammalian ErbB signaling pathway* reports on significant speedup.

## 1 INTRODUCTION

Systems biology studies the complex interactions of biological and biochemical systems rather than their individual molecular components (Sreenath, Cho, and Wellstead 2004; Cho et al. 2006). Simulation can be used to test hypotheses with in-silico experiments or to provide predictions to be tested by in-vitro and in-vivo studies. Ordinary Differential Equation (ODE)-based modeling is one of the most widely used simulation methods in systems biology. One form of system study investigates behavior by performing a parameter scan on a model with one or more parameters. A simulation is run for each combination of parameters. This can lead to excessively long run times. For example, a simulation of the mammalian ErbB signaling pathway only takes 20 seconds to run (Chen et al. 2009), a scan over one parameter in 1000 steps would take 5 hours, 2 parameters with 10 values each would take 11 hours and 3 parameters with 10 values each would take 3 months. Parameter scans are an example of embarrassingly parallel jobs, where there exists no dependency between those parallel jobs. As such, they lend themselves to parallel execution over distributed resources with the objective of speeding-up execution. These distributed resources may be wholly owned by individual organizations or may be shared between organizations; they may consist of supercomputers with High Performance Computing capabilities or may be composed of a network of hundreds of commodity PCs for High Throughput Computing; the distributed resources may be interconnected using private high speed networks or they may use the Internet or the corporate Intranet to share data and programs; the resources may be dedicated for solving a particular problem or they may utilize their computing cycles among various applications; further, the distributed resources may comprise of a varying pool of volunteer computing devices or may consist of a static collection of end-user devices.

These characteristics of distributed computing resources can arguably be organized in three general terms: Grid Computing, Desktop Grid and Volunteer Computing. A critical aspect of distributing computing resources is the manner by which users create Grid programs and workflows, access databases, submit jobs and receive results. These are discussed in the subsequent paragraphs.

A 'Grid' aims to share resources between organizations and user communities (Foster, Kesselman, and Tuecke 2001). Distributed Grid computing services (such as TeraGrid in the US and European Grid Infrastructure (EGI) in Europe) share processing power (ranging from supercomputers to desktop PCs), sensors and data storage capacity over these networks and the Internet across international communities of scientists organized into Virtual Organizations (VOs) (e-Infrastructures Roadmap 2010). These are typically linked to networks of specialized high performance computers. Common, standards-based middleware underpins these infrastructures, such as gLite and Globus (Laure et al. 2006), and supports application software developed to serve different scientific communities.

Many organizations have commodity PCs that often remain idle or run low CPU-intensive applications. It is the aim of Desktop Grid research to harness these idle CPU resources to support compute-intensive applications. There are two themes to this: Institutional Desktop Grid Computing (IDGC) and Volunteer Computing (VC) (Kacsuk et al. 2009). IDGC collectively uses office/lab PCs across local area networks within an institution to run applications. Simplistically, any PC user can submit applications to the IDGC by specifying the application and input parameters. CONDOR is one of the most widely used systems (Thain and Livny 2004). Others exist including Entropia (Chien et al. 2003) and OurGrid (Cirne 2006). Alternatively, VC aims to use vast numbers of home computers to support scientific applications such as searching for signs of extraterrestrial intellegence (SETI@Home) (Anderson et al. 2002) and protein folding (Folding@Home) (Beberg et al. 2009). Anyone can volunteer their computer by setting up client (worker) software which then connects to a server and downloads jobs to process. Arguably, the most well-known VC system is the Berkley Open Infrastructure for Network Computing (BOINC) (Anderson 2004). However, unlike CONDOR job submission the users of BOINC cannot submit applications unless they do so via a dedicated server (using pre-registered signed application binaries due to security considerations). Both approaches are attractive in that installation and maintenance of software is relatively simple and the processing power comes from already available commodity PCs rather than expensive to maintain computing clusters. However, this means these systems come with absolutely no guarantees on their quality of service.

A critical aspect of Grid systems is the manner by which users create Grid programs, develop workflows, submit jobs, monitor job execution and receive results. A Grid portal supports these functional aspects and, for example, allows users to access resources such as local clusters, supercomputers, desktop grids and cloud resources; to access application repositories containing scientific applications that can run on these resources; and to run and parameterize these applications transparently. Several portals exist (Russell et al. 2008; Barbera et al. 2007; Christie and Marru 2007; Lin et al. 2007). In this paper we use the WS-PGRADE Portal (Kacsuk 2011).

The focus of this paper is on the use of Volunteer Computing and a web-based Grid portal for speeding-up the execution of embarrassingly parallel system biology simulations. We report on efforts to speedup systems biology simulation using the SZTAKI Desktop Grid system. This combines a modified version of BOINC with the WS-PGRADE portal to produce a sophisticated IDGC system that has been successfully implemented in several institutions and has reported good speedups on several applications. Following this introduction section, the remainder of the paper is structured as follows. Section 2 gives the context of this work and introduces systems biology simulation and its computational requirements. Section 3 introduces the SZTAKI Desktop Grid system and the WS-PGRADE Portal. Section 4 describes the case study. Section 5 reports on performance results and the implications of these. Section 6 summarizes related work and Section 7 concludes the paper.

## 2 ODE MODELING IN SYSTEMS BIOLOGY

Systems biology studies the complex interactions of biological and biochemical systems rather than their individual molecular components and attempts to formulate descriptive models of such systems and expected responses to stimuli (Sreenath, Cho, and Wellstead 2004; Cho et al. 2006). A biological system consists of large numbers of functionally diverse and frequently multifunctional sets of elements that interact selectively and nonlinearly to produce coherent behaviors. For example, this can be a simple biological process, such as a biochemical reaction cycle, a gene regulatory network or a signaling pathway in a cell, tissue, an entire organism, or even an ecological web. Simulation can be used to test hypotheses with in-silico experiments or to provide predictions to be tested by in-vitro and in-vivo studies. However, a model is not a real or exact portrait of the biological system. It is rather a simplified description to assist in the analysis and understanding of the system. Thus we often need to identify key components and processes and attempt to predict biological behavior: such as which processes and proteins are most important for signaling, why certain genes are oncogenes or tumor suppressor genes, or what effects a particular experimental technology (e.g. RNA interference) or drug will have on a biological system.

Ordinary differential equations (ODEs) can be used to model the behavior of biochemical pathways, specifically the change of concentrations of species over time (Orton et al. 2005). In general, ODE representations of biochemical pathways are highly non-linear in nature, requiring numerical rather than analytical solutions. For example, enzymes serve a wide variety of functions inside living organisms. They are indispensable for signal transduction and cell regulation, often via kinases and phosphatises (Hunter 1995). Mass action kinetics are often used for modeling reactions within signaling pathways whereas Michaelis-Menten kinetics are often used in modeling the metabolic pathways. This latter kinetic model is relevant to situations where very simple kinetics can be assumed, holding at the initial stage of a reaction before the concentration of the product is appreciable, and makes the assumptions that the concentration of product is close to zero, no product reverts to the initial substrate and the concentration of the enzyme is much less than the concentration of the substrate (Gilbert et al. 2009). Thus the advantage of using Michaelis-Menten kinetics is that it enables a single differential equation to describe the enzymatic reaction.

ODE methods can have two major drawbacks. They are reliant on high-frequency sampling and parameter data being available, such as kinetic rates and absolute initial concentrations, and therefore can suffer from a lack of data. Also, numerical solvers need to employ small time steps in order to maintain accuracy and avoid unstable solutions, resulting in relatively long computational times. There are alternative methods to ODEs that can be used to model and analyze biological systems. Stochastic modeling approaches are based on representing the individual behavior of molecules and hence variability in the overall behavior of a biological systems. For example, Resat, et al. (2003) developed a probability weighted-dynamic Monte Carlo stochastic simulation, which was an integrated model of both the trafficking and signaling components of the EGFR system that comprises of hundreds of distinct endocytic compartments and about 13,000 reactions that occur over a broad spatio-temporal range. In this paper we focus on ODE methods to study the speedup that Volunteer Computing can offer for systems biology simulations.

## 3 THE SZTAKI DESKTOP GRID AND THE WS-PGRADE PORTAL

The SZTAKI Desktop Grid (SZDG) (Kacsuk et al. 2009; Urbah et al. 2009) (Figure 1) is an extension of BOINC in order to make suitable for institutional non-volunteer Desktop Grids and for the execution of parameter scan/sweep applications from a generic, high level user interface without the intervention of the BOINC project administrator. It was further developed during the European Desktop Grid Initiative project (www.edgi-project.eu). Porting legacy (existing) applications to a BOINC-based grid is facilitated by using

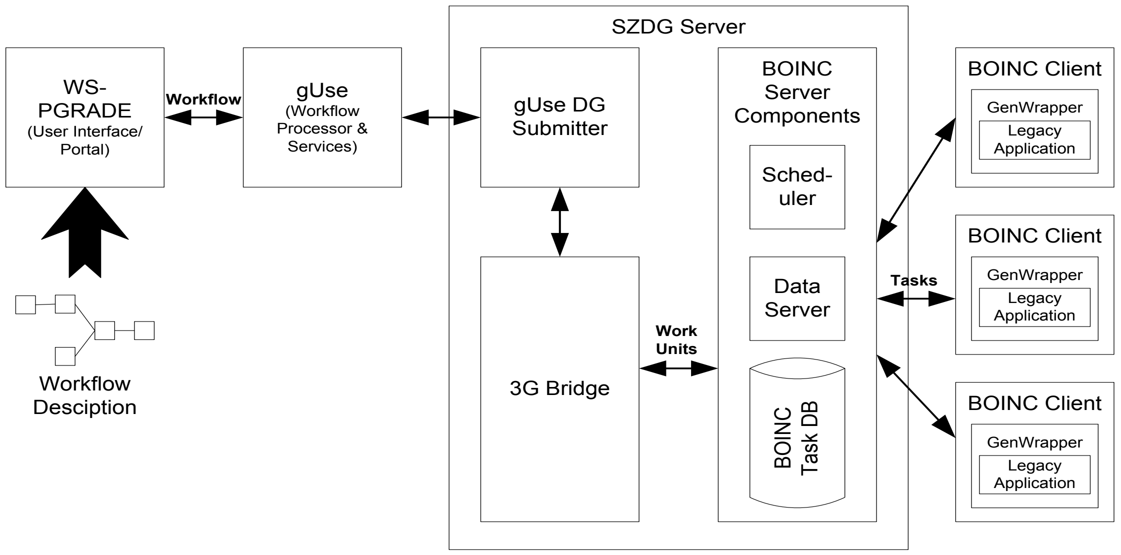

Figure 1: The SZTAKI Desktop Grid.

a BOINC Wrapper, essentially a batch/script file used by a BOINC client to execute the application. This has several limitations. The SZDG provides a more generic solution by using GenWrapper (Marosi, Balaton, and Kacsuk 2009). A BOINC-based DG system can limit the implementation of an application. For existing (legacy) applications, implementation consists of placing an application in a BOINC Wrapper that allows the application to be run as a sub-process while providing communication with the Grid Client. It is configurable (weight, checkpointing, etc.) and has a limited XML-based script language. This language is limited in functionality. GenWrapper was developed to provide a fully featured script language for this environment. It uses a POSIX-like shell scripting environment that describes how the application runs and how its work units should be processed. It consists of an extended version of BusyBox (www.busybox.net) to provide a single binary implementation of common UNIX commands (such as sed, grep, unzip, tar, awk, etc.), a POSIX shell interpreter (based on ash) and BOINC API functions. It runs on Windows, Mac OS X and Linux. The following describes the key components of GenWrapper and how a GenWrapper application runs; paraphrased from Marosi, Balaton, and Kacsuk (2009).

GenWrapper uses the GitBox and Launcher applications. GitBox for GenWrapper shares roots with GitBox (www.git-scm.org) which is a small footprint open source distributed version control system. However, this version was extended to include functionality from BusyBox and BOINC specific shell commands (and due to historical reasons is still called GitBox). Launcher is required to start the application running. A Genwrapper application therefore typically consists of a zip file containing files required by the legacy application to execute, GitBox, Launcher and an optional profile script to perform platform-specific preparations. This runs as follows. The client downloads these components with a work unit (input files and a work unit shell script). Launcher is started by BOINC and acts as a BOINC application and handles all communication with the client. Launcher then looks for a zip file and extracts all files. After unzipping the application archive, the Launcher generates a starter script which first sources the profile script if exists and then executes the work unit shell script. Launcher then calls the built in POSIX shell interpreter (ash) of GitBox which starts to execute this generated script. The Launcher remains running while GitBox executes the script and handles communication with the Core Client and performs similar tasks as the BOINC Wrapper.

Jobs are submitted to the SZDG via the WS-PGRADE portal. The WS-PGRADE portal, is a widely used general purpose Grid portal based on the P-GRADE ‘family’ of portals (Kacsuk 2011) and is based on Liferay (www.liferay.com). The WS-PGRADE portal supports the development of grid applications through a high-level, workflow-oriented programming approach using Directed Acyclic Graphs. The nodes of the workflow represent jobs (computations) and connections between nodes represent file transfers. The portal uses lower level services of the gUSE (Grid User Support Environment) framework. gUSE has several functions. It is a repository to store and access all workflow objects by providing a scalable set of high-level Grid services for data management and control (effectively a collaborative Grid application development environment). gUSE also manages the workflow created in the portal. Previous versions used the Condor DAGMan workflow engine. The current engine is Zen which has been created to manage huge numbers of simultaneous jobs (tested with one million). A file storage service handles input and output files. Jobs are submitted to different Grid systems via dedicated Submitter services. A Meta-broker service is used to send jobs to specific resources. The portal also allows local and remote grid resources to be mapped. This is supported by the 3G Bridge (Generic Grid-Grid Bridge) and the DC-API (essentially a Grid to BOINC adaptor) which provides Desktop Grid access to other Grid Infrastructures.

Many Grid applications are simulations that are run repeatedly with different parameters. To reflect this form of experimentation, or *parameter scanning*, one workflow object supported by WS-PGRADE is specifically designed for parameter scanning and is called the *computing* object. This works with generator objects and collector objects. Generator objects create the files to be processed. Collector objects collate the results returned from these jobs. The computing object turns the files created by the generator into jobs. In terms of workflow this means that parameter scanning applications can be created quickly and simply (i.e. a generator connected to a computing object connected to a collector). The computing object is quite expressive in that, for example, it has two input ports with input file numbers *N* and *M* respectively then the computing object will create *NxM* jobs (Kacsuk, Farkas, and Fedak 2008).

## 4 CASE STUDY

Our case study was carried out on the University of Westminster Local Desktop Grid (WLDG), an implementation of the SZDG. WLDG connects laboratory PCs of the University of Westminster (London, UK) into an IDGC infrastructure. The university is set over four main campuses and some additional smaller locations in Central and North-West London, each of them offering a variable number of Windows-based dual core PCs for teaching purposes. Over 1600 of these machines are connected to the WLDG. It is also connected to the European Grid Infrastructure (EGI) by the 3G Bridge allowing EGI users to run validated applications on the WLDG. The Westminster Grid Application Support Service (W-GRASS) offers application porting services and runtime support for perspective users. Currently nine different applications are supported by the WLDG from diverse disciplines, including biomolecular simulations, 3-D video rendering, x-ray profile analysis and digital signal processing. W-GRASS implements applications on the Grid by following an application development methodology developed by the EDGeS project (the precursor to the EDGI project) (www.edges-project.eu) - the EDGeS Application Development Methodology (EADM). This approach follows a series of familiar software development steps and concentrates only on application specific aspects needed for porting to a service grid/desktop grid infrastructure. Its goal is to address necessary questions of application porting within existing technical constraints and limitations. The steps of EADM are: analysis of current application, requirements analysis, systems design, detailed design, implementation, testing, validation, deployment and user support, maintenance & feedback.

The SIMAP Utility, developed at Brunel University, is a platform-independent software environment for systems biology (Wang et al. 2009) and follows previous research on BIONESSIE-G (Liu et al. 2008). This supports the modeling of biochemical networks, and also the simulation and analysis of the dynamic behavior of biochemical models. The tool can compute changes of species concentrations over time with particular parameter values by simulating a Systems Biology Markup Language (SBML) model numerically with the SBML ODE Solver Library (SOSLib) (http://www.tbi.univie.ac.at/~raim/odeSolver/). The SIMAP Utility includes a set of computational modules for simulating and analyzing biochemical models. These are an Ordinary Differential Equations-based simulator, a sensitivity analyzer, a parameter scanner, a model fitting module, a gene knockdown analyzer, and a model logic checker.

The EADM was followed to develop a version of SIMAP that runs on WLDG. The core of SIMAP is SOSLib. SIMAP has a front end that allows a user to specify and run a biological model specified in the Systems Biology Markup Language (SBML). Alternatively a user can use a command line interface with arguments that specify the SBML model and associated data. A single experiment is run (the simulation of the model) and results obtained. A parameter sweep uses SIMAP to run (simulate) a SBML model potentially many thousands of times. A typical usage scenario of the application is the following.

- The user prepares the SBML model files that he/she wants to execute.
- The user specifies/uploads the files (or a single archive with the input files) through the user interface (preferably a portal interface).
- The solution creates work units from the input files, and sends them for execution to the compute nodes of the grid.
- The compute nodes run the simulation and send back the results to the server.
- The front-end downloads the results.
- The user uses the downloaded results and for further analysis and processing.

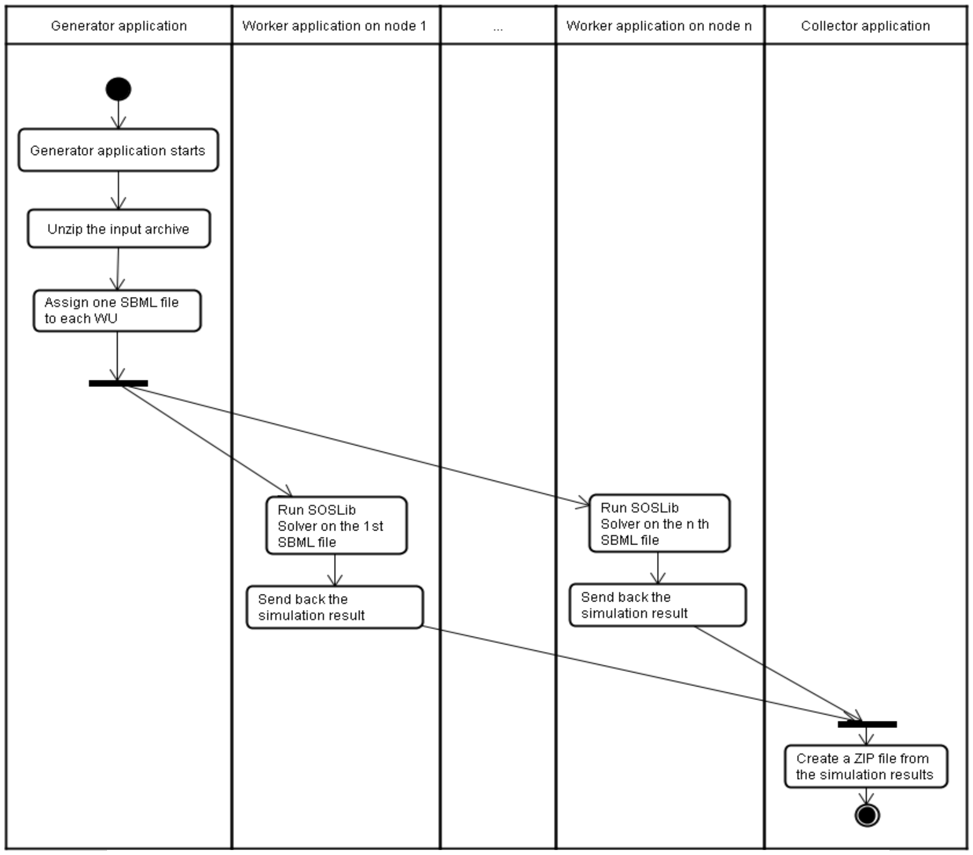
Generator application
Worker application on node 1
...
Worker application on node n
Collector application
Generator application starts
Unzip the input archive
Assign one SBML file to each WU
Run SOSLib Solver on the 1st SBML file
Send back the simulation result
Run SOSLib Solver on the n th SBML file
Send back the simulation result
Create a ZIP file from the simulation results

Figure 2: Control Flow of the Ported SIMAP Application.

Each worker runs t [illegible] cation is run using GenWrapper (and allows SOSlib to process (simulate) multiple SBML files if needed). This is made significantly easier as SOSlib can be run from the command line. Without modifying the original application at all, the application can be simply distributed with required SBML input files to the worker nodes. The application will therefore contain three parts: a generator, a set of workers and a collector application. The generator gets input from the user and creates a work unit for each SIMAP input file. The worker runs the simulation and sends back the results to the server, where the collector creates an archive which will hold the result files. Master-worker type parallelization is possible as the simulations are independent from each other. The control flow is shown in Figure 2. The key here to a quick Grid implementation is that the SIMAP application does not require any modification as it can be accessed by a command line interface. Only the GenWrapper scriptfile needed to implement control flow. The application is computation but not data intensive. Therefore, there are no special requirements on data access. The normal BOINC data distribution mechanism is sufficient. The following list summarizes the minimum set of files required by the implementation.

```
# Logical names of the input/output files
INPUT_FILE_NAME="input.xml"
OUTPUT_FILE_NAME="results.txt"
SOLVER_EXECUTABLE="odeSolver.exe"
STDOUT_FILE_NAME="stdout.log"
STDERR_FILE_NAME="stderr.log"
GRIDNFO_FILE_NAME="gridnfo.log"
# SOSlib parameters
TIME=100
PRINTSTEP=200
# Error codes
GENERAL_ERROR=1
RESOLVE_ERROR=2
INPUT_FILE_MISSING_ERROR=3

#Initialise files
echo "Initialising files..." 1>&2
RESOLVED=`boinc resolve_filename "${INPUT_FILE_NAME}"`
INPUT_FILE=${RESOLVED}
RESOLVED=`boinc resolve_filename "${OUTPUT_FILE_NAME}"`
OUTPUT_FILE=${RESOLVED}
RESOLVED=`boinc resolve_filename "${STDOUT_FILE_NAME}"`
STDOUT_FILE=${RESOLVED}
RESOLVED=`boinc resolve_filename "${STDERR_FILE_NAME}"`
STDERR_FILE=${RESOLVED}

# Execute SOSlib
echo "Executing the solver..." 1>&2
chmod +x ${SOLVER_EXECUTABLE}
./${SOLVER_EXECUTABLE} ${INPUT_FILE} --time ${TIME} --printstep
${PRINTSTEP}  1> ${OUTPUT_FILE}
if [ $? -ne 0 ]; then
     echo "Solver execution failed. Error code: $?" 1>&2
     exit $?
fi
echo "finished" >> ${STDERR_FILE}
echo "finished" >> ${STDOUT_FILE}
echo "Solver finished." 1>&2

# Tidy up
RESOLVED=`boinc resolve_filename "${GRIDNFO_FILE_NAME}"`
GRIDNFO_FILE=${RESOLVED}
```

Figure 3: GenWrapper scriptfile for SOSlib Application.

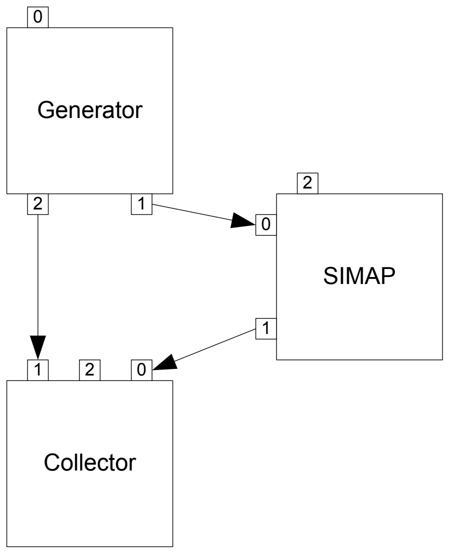

Figure 4: Workflow of the Ported SIMAP Application.

1. **Input ZIP file:** The archive file that will be uploaded by the user on the portal. This archive holds all of the SBML files. The size of this file can be large and will have some effect on performance. However, this should be mitigated by overall runtime.
2. **SBML model file:** the SOSlib application that runs on the worker nodes executes a simulation according to the parameters described in this file.
3. **Result files:** the SOSlib application on the client nodes generates a text file that is the output of the simulation. These output files should be sent back to the server to be processed by the collector application.
4. **Output file:** the final output of the workflow is again a ZIP file that holds all of the result files.
5. **SOSlib files:** all of the files that are needed to execute the SBML ODE Solver Library application on a client computer. These are sent to the worker with the model files.
6. **Log files:** Log files generated by the worker application.
7. **Work unit script** file: The script file that runs the the SOSlib simulations (see Figure 3).

Figure 4 shows the workflow created using the WS-PGRADE portal. This was made relatively straightforward by using the generator, computing and collector objects. As the figure shows, port 0 of the generator object is used to initialize the computation. The SBML models are sent from the generator to the SIMAP computing object port 0. Port 2 of the object is used to specify the computation granularity (simulations per job). The portal uses this workflow object to generate jobs and distribute them to available worker PCs. Port 0 of the collector object is used to retrieve and collate result files. Additionally, the WS-PGRADE portal allows users to upload the required files and specify the different parameters on an easy to use web based form (a simple HTML form) before submission.

## 5 RESULTS

To investigate the performance we used the *mammalian ErbB signaling pathway* model as mentioned in the introduction. Briefly, this model represents the ERbB1-4 receptor tyrosine kinases (RTKs) and the signaling pathways they activate which govern most core cellular processes such as cell division, motility and survival and are strongly linked to cancer when they malfunction due to mutations, etc. The ODE-based mass action ErbB model comprises 499 species (molecules), 201 parameters and 828 reactions. The model implements compartments for plasma, endosomal membranes, cytosol, nucleoplasm and lysosomal lumen, as well as clathrin-mediated endocytosis. The model takes 20 seconds to run per simulation and while some systems biology models take longer to run, the runtime is largely representative of many ODE-based systems biology simulations. 6400 simulations were performed in total for each experiment. Experiments were performed using different granularities (numbers of simulations) per job (i.e. the model/solver were only sent once and the job performed *N* simulations). These are: 4, 8, 16, 32 and 64 (giving 1600, 800, 400, 200 and 100 jobs respectively). 5 runs were performed for each experiment at different times of the day.

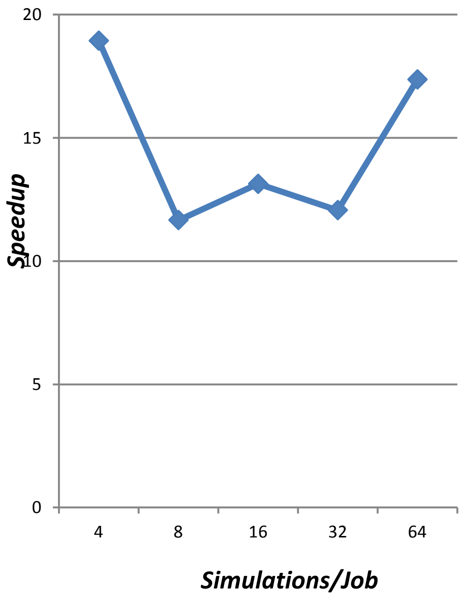

Figure 5: Average Speedup.

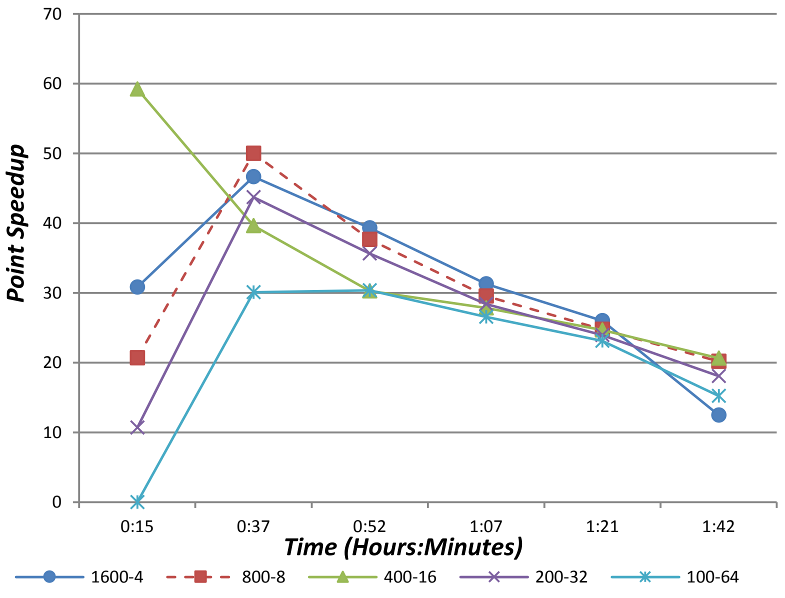

Figure 6: Point Speedup.

Table 1 shows the performance data and Figure 5 shows the average speedups. As can be seen the average speedup does not follow the 'typical' speedup expected in distributed computing (i.e. depending on parallelism (number of parallel jobs), the larger the granularity, the better the speedup). The maximum speedup is 18.9 (4 simulations/job) and the minimum is 12.1 (32 simulations/job). Within each experiment the speedups vary considerably. For example, the minimum/maximum speedup of 4 simulations/job is 13.6

and 24.5. The number of PCs used is not known as this statistic cannot be collected from the WLDG. Also, the WLDG was in full use by other Grid users.

Table 1: Speedup (6400 Simulations).

| | | Run | | | | | |
|---|---|---|---|---|---|---|---|
| **Sim /job** | **No. of Jobs** | **1** | **2** | **3** | **4** | **5** | **Average** |
| 4 | 1600 | 18.9 | 21.8 | 24.5 | 15.9 | 13.6 | 18.9 |
| 8 | 800 | 9.1 | 15.0 | 12.9 | 11.1 | 10.1 | 11.7 |
| 16 | 400 | 8.8 | 14.8 | 14.2 | 12.5 | 15.3 | 13.1 |
| 32 | 200 | 9.4 | 11.7 | 11.2 | 14.8 | 13.3 | 12.1 |
| 64 | 100 | 8.6 | 23.7 | 17.9 | 20.7 | 15.9 | 17.4 |

By a comparison, results from using a granularity of 1 simulation/job over a CONDOR Grid of 32 Dual Core PCs generated a speedup of around 12 (Taylor et al. 2010). However, when similar granularities were used this speedup rose to around 40. To further investigate this we studied what we term the 'point' speedup. This is the speedup taken at regular intervals during the computation. Figure 6 shows the average speedups for our experiments over time. This presents a very different picture. The results clearly show that after around half an hour the computations had reached peak speedups of between 30 and 50. The exception to this is that experiments for 4 simulations/job show a speedup of almost 60 after 15 minutes. All experiments then follow the same trend, i.e. speedup gradually reduces over time. This is a result of failed computations. All jobs on the WLDG are automatically cancelled when a regular user (typically a student) logs in and starts using the computer for learning. This results in a failed computation. The WLDG will repeat all failed/cancelled jobs until the job is registered as finished. Repeated jobs represent additional work that result in the reducing speedup.

## 6 RELATED WORK

Previous work addressed the development of the grid-enabled Biochemical Networks Simulation Environment (BioNessieG) (Liu et al. 2008). It used the High Performance Computing cluster facilities of the UK National Grid Service (NGS) and ScotGrid to execute large-scale parameter scans. The communication between the grid resources and the client side of BioNessieG was implemented through web services and the Globus Toolkit (Foster 2006) . A job scheduler was also developed to evaluate the suitability and availability of grid resources prior to job submission. However, poor speedup was achieved by this approach due to in part to the small job size (relative to communication overheads) and unpredictable job queuing on the NGS and ScotGrid when the experiments were carried out.

There are several examples of where systems biology related simulations have been implemented on parallel and distributed computing systems. Not all related work report on performance. An implementation of MCell (a computational biology simulation framework) using Globus obtained a 50 times speedup using 113 dual Supercomputer CPU nodes using a test model (Casanova et al. 2004). Mosca, *et al*. (2009) investigate parameter sweep experimentation for complex biochemical systems on the EGEE framework via gLite and CONDOR-G using the stochastic simulator $\tau$-DPP (Cazzaniga et al. 2006) (an alternative to ODE-based simulation). Experiments were run on the BIOMED Virtual Organization, which shares on average at the time 2000 compute cluster CPUs, with the Challenge Control System portal (Milanesi et al. 2012). Work closely related to grid systems include Burrage, *et al*. (2009) who discuss parallel computing techniques used to speed up the simulation of a plasma membrane model using cluster computing. The research has delivered some very promising speedup results (47 and 15 times over 32 nodes using different architectures) that could lead to implementation on a grid system but is limited to this class of model. Chang, *et al*. (2008) have developed the metabolic simulation code HiPer SBTK. Initial performance studies using a master-slave architecture have been performed for a 64-parameter sensitivity minimization on our prototype model of *C. reinhardtii*. Results appear to show that a speedup of less than 10 for up to 100 processors.

## 7 CONCLUSIONS

This paper has reported on experiences of using the SZTAKI Desktop Grid system as an institutional Desktop Grid. Reported speedups were initially disappointing (despite promising point speedup) and are a consequence of using a shared Grid resource in a general computing environment. However, as shown in related work, these are comparable unless using dedicated hardware and/or exploiting specific algorithmic features. The use of the WS-PGRADE portal is significant as it resulted in the quick development and deployment of our systems biology application in a production environment. This is unlike our previous experience of using BOINC to speed-up Monte Carlo simulations (Zhang et al. 2007), which required server side installation and was time consuming. Two main threads of research are now being followed. The first will investigate the use of a dedicated server that can be used when the point speedup drops beyond an acceptable speedup. The second follows the evolution of the SZTAKI Desktop Grid as researchers investigate the deployment of the technology onto a Cloud.

## REFERENCES

Anderson, D. 2004. "BOINC: A System for Public-resource Computing and Storage." In *Proceedings of the 5th IEEE/ACM GRID Workshop* (Pittsburgh, Pennsylvania, Nov 8, 2004), 4-10. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Anderson, D., J. Cobb, E. Korpela, M. Lebofsky, and D. Werthimer. 2002. "Seti@home: An Experiment in Public-resource Computing." *Communication of the ACM* 45:56-61.

Barbera, R., A. Falzone, V. Ardizzone, and D. Scardaci. 2007. "The GENIUS Grid Portal." In *Proceedings of the 16th IEEE Workshops on Enabling Technologies*: *Infrastructure for Collaborative Enterprises* (Paris, France, June18-20, 2007), 279-283. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Beberg, A. L., D. L. Ensign, G. Jayachandran, S. Khaliq, and V. S. Pande. 2009. "Folding@home." In *Proceedings of the 8th IEEE Workshop on HPC Biology* (Italy, May 23-29), 1-8. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Burrage, P. M., K. Burrage, K. Kurowski, M. Lorenc, D. V. Nicolau, M. Swain, and M. A. Ragan. 2009. "A Parallel Plasma Membrane Simulation." In *Proceedings of the 2009 Workshop on High Performance Computational Systems Biology* (Trento, Italy, Oct 14-16), 105-112. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Casanova,H., F. Berman, T. Bartol, E. Gokcay, T. Sejnowski, A. Birnbaum, J. Dongarra, M. Miller, M. Ellisman, M. Faerman, G. Obertelli, R. Wolski, S. Pomerantz, and J. Stiles. 2004. "The Virtual Instrument: Support for Grid-enabled MCell Simulations." *International Journal of HPC Applications* 18:3-17.

Cazzaniga, P., D. Pescini, D. Besozzi, and G. Mauri. 2006. "Tau Leaping Stochastic Simulation Method in P Systems Membrane Computing." *Lecture Notes in Computer Science*, 4361, 298–313.

Chang, C. H., P. Graf, D. M. Alber, K. Kim, G. Murray, M. Posewitz, and M. Seibert. 2008. "Photons, Photosynthesis, and High-performance Computing: Challenges, Progress, and Promise of Modeling Metabolism in Green Algae." *Journal of Physics: Conference Series,* 125:1-13.

Chen, W. W., B. Schoeberl, P. J. Jasper, M. Niepel, U. B. Nielsen, D. A. Lauffenburger, and P. K. Sorger. 2009. "Input–output Behavior of ErbB Signaling Pathways as Revealed by a Mass Action Model Trained Against Dynamic Data." *Molecular Systems Biology* 5, 239.

Chien, A., B. Calder, S. Elbert, and B. Karan. 2003. "Entropia: Architecture and Performance of an Enterprise Desktop Grid System." *Journal of Parallel and Distributed Computing* 63:597-610.

Cho, C. R., M. Labow, M. Reinhardt, J. van Ooostrum, and M. C. Peitsch. 2006. "The Application of Systems Biology to Drug Discovery." *Current Opinion in Chemical Biology* 10:294-302.

Christie, M., and S. Marru. 2007. "The LEAD Portal: A TeraGrid Gateway and Application Service Architecture: Research Articles." *Concurrency & Computation*: *Practice & Experience* 19**:**767–781.

Cirne, W., F. Brasileiro, N. Andrade, L. B. Costa, and A. Andrade. 2006. "Labs of the World, Unite!!!" *Journal of Grid Computing* 4:225-246.

e-Infrastructures Roadmap. 2010. Accessed August 2011. *www.e-irg.org/images/stories/publ/eirg-roadmap.pdf*.

Foster, I. 2006. "Globus Toolkit Version 4: Software for Service-Oriented Systems." *Journal of Computer Science and Technology* 21:513–520.

Foster, I., C. Kesselman, and S. Tuecke. 2001. "The Anatomy of the Grid: Enabling Scalable Virtual Organizations." *International Journal of High Performance Computing Applications* 15:200-222.

Gilbert, D. R., R. Breitling, M. Heiner, and R. A. Donaldson. 2009. "An Introduction to BioModel Engineering, Illustrated for STP." *Membrane Computing*. Springer LNCS, 5391, 13-28.

Hunter, T. 1995. "Protein Kinases and Phosphatases: The Yin and Yang of Protein Phosphorylation and Signaling." *Cell* 80:225–36.

Kacsuk, P. 2011. "P-GRADE Portal Family for Grid Infrastructures." *Concurrency and Computation: Practice and Experience* 23:235-245.

Kacsuk, P., Z. Farkas, and G. Fedak. 2008. "Towards making BOINC and EGEE Interoperable." In *Proceedings of the Fourth IEEE International Conference on eScience* (Indianapolis, Indiana, Dec 7 - 12, 2008), 478-484. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Kacsuk, P., J. Kovács, Z. Farkas, A. Marosi, G. Gombás, and Z. Balaton. 2009. "SZTAKI Desktop Grid (SZDG): A Flexible and Scalable Desktop Grid System." *Journal of Grid Computing* 7:439–461.

Laure, E., S.M. Fisher, A. Frohner, C. Grandi, P. Kunszt, A. Krenek, O. Mulmo, F. Pacini, F. Prelz, J. White, M. Barroso, P. Buncic, F. Hemmer, A. Di Meglio, and A. Edlund. 2006. "Programming the Grid with gLite." *Computational Methods in Science and Technology* 12: 33-45.

Lin, A. W., S. T. Peltier, J. S. Grethe, and M. H. Ellisman. 2007. "Case Studies on the Use of Workflow Technologies for Scientific Analysis: The Biomedical Informatics Research Network and the Telescience Project." In *Workflows for e-science*, edited by I. J. Taylor, E. Deelman, D. B. Gannon, and M. Shields, 109–125, Berlin: Springer.

Liu, X., J. Jiang, O. Ajayi, X. Gu, D. R. Gilbert, and R. Sinnott. 2008. "BioNessie(G)-A Grid Enabled Biochemical Networks Simulation Environ." *Studies in Health Technology & Informatic*s 138:147-157.

Marosi, A. C., Z. Balaton, and P. Kacsuk. 2009. "GenWrapper: A Generic Wrapper for Running Legacy Applications on Desktop Grids." In *Proceedings of the IEEE International Symposium on Parallel & Distributed Processing* (Rome, Italy, May 25-29, 2009), 1-6. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Milanesi, L., I. Merelli, G. T. P. Cozzi, and A. Orro. 2012. "Functional Genomics Applications in Grid." In *Grid and Cloud Computing: Concepts, Methodologies, Tools and Applications*, edited by M. Cannataro, 149-167. Hershey, PA: Medical Information Science Reference.

Mosca, E., I. Merelli, L. Milanesi, P. Cazzaniga, D. Pescini, and G. Mauri. 2009. "Stochastic Simulations on a Grid Framework for Parameter Sweep Applications in Biological Models." In *Proceedings of the 2009 International Workshop on High Performance Computational Systems Biology* (Trento, Italy, October 14 -16, 2009), 33-42. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Orton, R. J., O. E. Sturm, V. Vyshemirsky, M. Calder, D. R. Gilbert, and W. Kolch. 2005. "Computational Modelling of the Receptor-tyrosine-kinase-activated MAPK Pathway." *Biochemical J* 392:249-261.

Resat, H., J. A. Ewald, D. A. Dixon, and H. S. Wiley. 2003. "An Integrated Model of Epidermal Growth Factor Receptor Trafficking and Signal Transduction." *Biophysical Journal* 85:730-743.

Russell, M., P. Dziubecki, P. Grabowski, M. Krysiński, T. Kuczyński, D. Szjenfeld, D. Tarnawczyk, G. Wolniewicz, and J. Nabrzyski. 2008. "The Vine Toolkit: A Java Framework for Developing Grid

Applications." *Parallel Processing and Applied Mathematics*, *Lecture Notes in Computer Science* 4967:331–340.

Sreenath, S. N., K. H. Cho, and P. Wellstead. 2008. "Modelling the Dynamics of Signalling Pathways." *Essays in Biochemistry: Systems Biology* 45:1-28.

Taylor, S. J. E., N. Mustafee, S. Kite, C. Wood, S. J. Turner, and S. Strassburger. 2010. "Improving Simulation through Advanced Computing Techniques: Grid Computing and Simulation Interoperability." In *Proceedings of the 2010 Winter Simulation Conference* (Baltimore, Maryland, December 5 – 8, 2010), 216-230. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Thain, D., and M. Livny. 2004. "Building Reliable Clients and Services." In *The Grid: Blueprint for a New Computing Infrastructure*, edited by I. Foster and C. Kesselman, 285–318. San Francisco: Morgan Kaufman.

Urbah, E., P. Kacsuk, Z. Farkas, G. Fedak, G. Kecskemeti, O. Lodygensky, A. Marosi, Z. Balaton, G. Caillat, G. Gombas, A. Kornafeld, J. Kovacs, H. He, and R. Lovas. 2009. "EDGeS: Bridging EGEE to BOINC and XtremWeb." *Journal of Grid Computing* 7:335–354.

Wang, J., X. Liu, N. Mustafee, Q. Gao, S. J. E. Taylor, and D. R. Gilbert. 2009. "Grid-enabled SIMAP Utility: Motivation, Integration Technology and Performance Results." In *Proceedings of UK e-Science All Hands Meeting* (Oxford, UK, Dec 7 - 9, 2009).

Zhang, J., N. Mustafee, J. Saville, and S. J. E. Taylor. 2007. "Integrating BOINC with Microsoft Excel: A Case Study." In *Proceedings of the 29th Information Technology Interfaces Conference* (Dubrovnik, Croatia. June 25-28, 2007), 733 – 738. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

## AUTHOR BIOGRAPHIES

**SIMON J. E. TAYLOR** is a Reader in the Department of Computer Science, Brunel University. He is the leader of the Modelling & Simulation Group. His email address is Simon.Taylor@brunel.ac.uk.

**MOHAMMADMERSAD GHORBANI** was a PhD student in the Department of Computer Science, Brunel University and graduated in 2014. His email address is Mohammadmersad.Ghorbani@brunel.ac.uk.

**NAVONIL MUSTAFEE** is a Senior Lecturer in the Centre for Innovation and Service Research, University of Exeter Business School. His email address is n.mustafee@exeter.ac.uk. His webpage can be accessed at http://sites.google.com/site/navonilmustafee/.

**TAMAS KISS** is a Reader at the Department of Business Information Systems, Faculty of Science and Technology, University of Westminster. His email address is T.Kiss@westminster.ac.uk.

**PETER BORSODY** is a Software Engineering in the Centre for Parallel Computing at the University of Westminster. His email address is P.Borsody@westminster.ac.uk.

**ANNETTE PAYNE** is a Lecturer in the Department of Computer Science, Brunel University. His email address is Annette.Payne@brunel.ac.uk.

**PROFESSOR DAVID GILBERT** is the co-director of the Brunel Centre for Systems and Synthetic Biology. His email address is David.Gilbert@brunel.ac.uk.