SPEEDING UP SYSTEMS BIOLOGY SIMULATIONS OF BIOCHEMICAL PATHWAYS USING CONDOR

Xuan Liu[1,2], Simon J E Taylor[1], Navonil Mustafee[3], Jun Wang[1], Qian Gao[1], and David Gilbert[1]

1 Department of Information Systems and Computing, Brunel University, Uxbridge, Middlesex, UB8 3PH, UK. Email: firstname.lastname@brunel.ac.uk

2 Centre for Genomic Research, Institute of Integrative Biology, University of Liverpool, Liverpool, L69 7ZB, UK. Email: xuan.liu@liverpool.ac.uk

3 Centre for Innovation & Service Research, University of Exeter Business School, University of Exeter, Exeter EX4 4ST, UK. Email: n.mustafee@exeter.ac.uk

ABSTRACT

Systems biology is a scientific field that uses computational modelling to study biological and biochemical systems. The simulation and analysis of models of these systems typically explore behaviour over a wide range of parameter values; as such, they are usually characterised by the need for non-trivial amounts of computing power. Grid computing provides access to such computational resources. In previous research we created the grid-enabled *Biochemical Networks Simulation Environment (BioNessieG)* to attempt to speedup system biology simulations over a grid (the UK National Grid Service and ScotGrid). Following on from this work, we have created the *SIMAP Utility*, a standalone simulation tool dedicated to the modelling and analysis of the *EGFR-MAP kinase pathway*. This builds on experiences from BioNessieG by decoupling the simulation modelling elements from the Grid middleware. This new utility enables us to interface with different grid technologies. This paper therefore describes the new SIMAP Utility and an empirical investigation of its performance when deployed over a desktop grid based on the High Throughput Computing middleware *Condor*. We present our results based on a case study with a model of the *mammalian ErbB signaling pathway*, a pathway strongly linked to cancer.

KEYWORDS: Systems biology; biochemical simulations; biomodel engineering desktop grid computing; Condor; job manager.

1. INTRODUCTION

Systems biology concerns the systematic study of biological and biochemical systems in terms of complex interactions rather than their individual molecular components [1,2]. At the core of systems biology is biomodel engineering, which is the science of designing, constructing and analyzing computational models of biological systems [3]. One typical aspect of construction and analysis involves the exploration of the behaviour of the system where the search space is large, requiring a common parameter sweep over a wide range of parameter values using a parameter scan that runs one simulation for each parameter combination. Such a search requires special strategies to deal with the complexity of the problem. A single parameter scan with many steps requires a large number of simulations to execute, which results in a long running time. For example, one

simulation based on the computational model of the *mammalian ErbB signaling pathway* [3], a pathway strongly linked to cancer, requires only 20 seconds to complete on a typical desktop computer, but over 5 hours to run 1000 variations of a single parameter. Such a process is necessary to investigate the relationships between model parameters or to perform sensitivity analysis on a given parameter. However, when several parameters are to be explored in combination, then the time complexity is polynomial in the number of parameters; thus for this example, scanning 2 parameters in combination, each over only 10 values, would take over 11 hours, and 3 parameters would take 3 months. Such computational bottlenecks can adversely affect the progress of areas such as cancer research and drug discovery.

The field of grid computing offers an integrated infrastructure providing geographically distributed sites with secure access to computational, data and instrumentation resources [4]. In our research, we are interested in developing tools for Systems Biology and in the use of grid technologies to speed up simulation and analysis in different areas of science and commerce. Previous work addressed the development of the grid-enabled Biochemical Networks Simulation Environment (BioNessieG) [5]. It used computational resources of the UK National Grid Service (NGS) [6] and ScotGrid [7] to execute large-scale parameter scans. The communication between the grid resources and the client side of BioNessieG was implemented through web services and driven by Apache Axis 2 [8] and the Globus Toolkit 4.0 [9]. 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.

Following on from our experiences with BioNessieG, the European Union Framework Programme 6 funded *Simulation Modelling of the EGFR-MAP Kinase Pathway* (SIMAP) project [10] investigated the decoupling of the simulation modelling elements from the grid-based implementation to develop a comprehensive simulation of a biochemical model of the EGFR-MAP kinase pathway in connection to the clinical phenotype. Specifically, this addressed the modelling of cancer-related pathway behaviour, accompanied by new data mining techniques, modelling tools and clinical data integration. The result was *BioNessie* and it is freely available at: *http://disc.brunel.ac.uk/bionessie*. In this paper we have referred to BioNessie as the SIMAP Utility, since funding from the SIMAP project made possible certain extensions to the tool, in particular, it could now be interfaced with different grid technologies. The specific download link for SIMAP is *http://disc.brunel.ac.uk/bionessie/downloadsimap.html*.

As part of on-going work to investigate the performance of the SIMAP Utility on different grid technologies, this paper reports on experiences with deploying the SIMAP software on a *desktop grid* [11] using Condor [12]. A desktop grid is essentially a network of desktop PCs, as typically found in many organisations, linked together with appropriate grid middleware. Our previous work in implementing a desktop grid in a bank and an automobile manufacturing company met with some success [13]. We were

keen to determine if a desktop grid, implemented in this case with Condor, could be as useful for our SIMAP Utility. To describe this work and its results, we first introduce the background and theory of computational modelling for biochemical pathways in Section 2. Desktop PC-based grid computing is presented in Section 3. The implementation of the SIMAP Utility with Condor is described in Section 4. Section 5 gives experimentation and results. The implications of our results and related work are discussed in Section 6, the concluding section of the paper.

2. MODELLING BIOCHEMICAL PATHWAYS

2.1 Computational Modelling

From a systems biology perspective, a biological system is a set of complex interactions (network structure) rather than many individual molecular components. A biological system comprises large numbers of functionally diverse, and frequently multifunctional sets of elements that interact selectively and nonlinearly to produce coherent behaviours. This can be anything from a simple biological process, such as a biochemical reaction cycle, a gene regulatory network or a signalling pathway in a cell, tissue, an entire organism, or even an ecological web. Part of the core of systems biology is biomodel engineering, which is the science of designing, constructing and analyzing computational models of biological systems [3]. This can involve, for example, the process of simulating an abstract model of a biological system to test hypotheses with *in-silico* experiments or to provide predictions to be tested by *in-vitro* and *in-vivo* studies. In order to achieve the goal of answering biological questions, models have to reliably depict a biological system and be able to predict its behaviour. Based on the schematic representation of its components and their links, the model is a description of the dynamic behaviour of the biological system, which equips the model with predictive power [14] (Figure 1). 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 of the system. Thus we often need to identify key components and processes and to predict biological behaviour such as which processes and proteins are most important for signalling; why certain genes are oncogenes or tumour suppressor genes; or what effects a particular experimental technology (e.g. RNA interference) or drug will have on a biological system. We now review the main modelling techniques and computational tools used in this area.

2.2 Ordinary Differential Equations (ODEs) Modelling Method

Ordinary differential equations (ODEs) can be used to model the behaviour of biochemical pathways, specifically the change of concentrations of species over time [15]. Taking a simple biochemical reaction as an example, the decay of substrate *A* to product *B* at rate *k* can be depicted by a simple mass action equation: $A \xrightarrow{k} B$, which can be translated into differential equations as follows where [A] represents the concentration of A:

$$\frac{d[A]}{dt} = -\frac{d[B]}{dt} = -k \times [A]$$

Furthermore, many biochemical reactions are reversible, for example $A \underset{k2}{\overset{k1}{\rightleftarrows}} B$, which can be described as the following differential equations:

$$\frac{d[A]}{dt} = -k_1[A] + k_2[B]$$
$$\frac{d[B]}{dt} = k_1[A] - k_2[B]$$

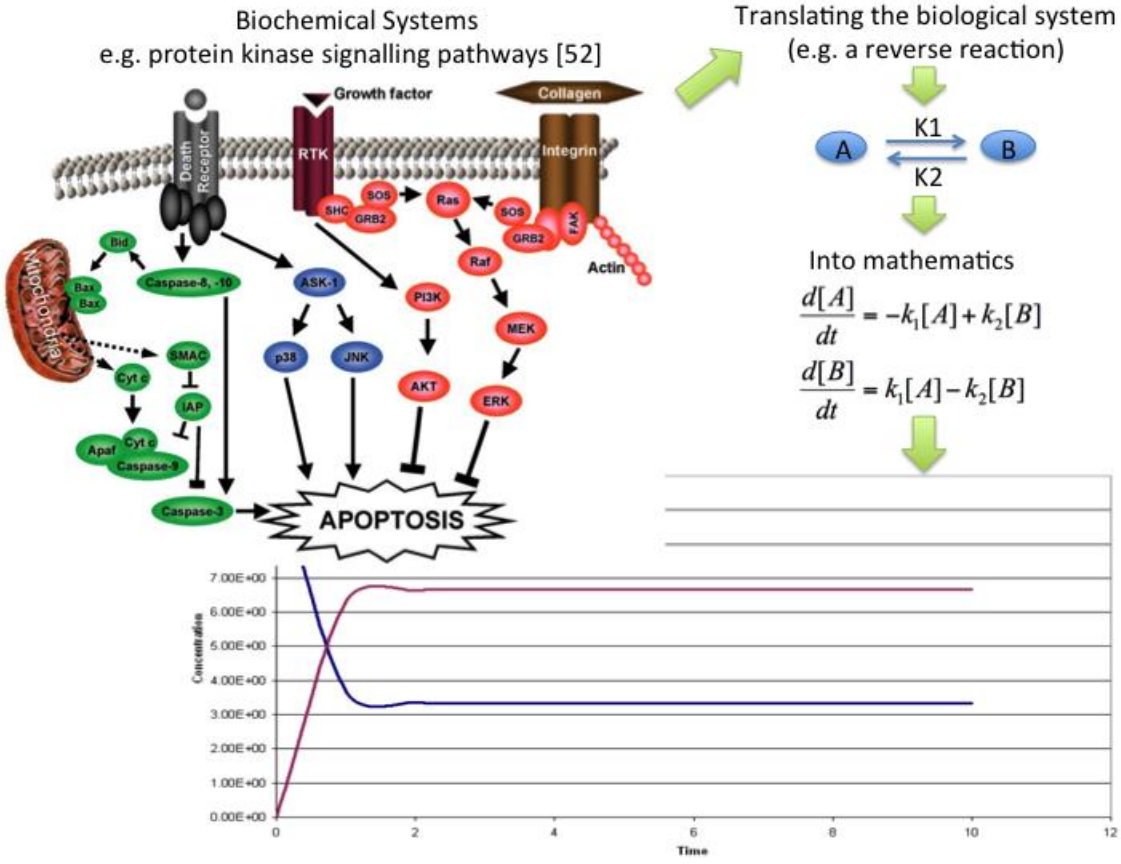

Figure 1: The schematic procedure to show how computational modelling works. A biochemical system will firstly need to be translated into a set of connected reactions, and then particular kinetic laws (e.g. Mass action) will be applied to turn those reactions into mathematics (e.g. ODEs). Subsequently, the mathematics can be solved by various computational modelling tools to depict the dynamic behaviour of the biological system.

Since biochemical reactions are often catalysed by enzymes which are not consumed during the process, we can extend the examples above as the following mass action description and differential equations:

$$A + E \underset{k2}{\overset{k1}{\rightleftarrows}} A|E \xrightarrow{k3} B + E$$

$$\frac{d[A]}{dt} = -k1' [A]' [E] + k2' [A|E]$$
$$\frac{d[A|E]}{dt} = k1' [A]' [E] - k2' [A|E] - k3' [A|E]$$
$$\frac{d[B]}{dt} = k3' [A|E]$$
$$\frac{d[E]}{dt} = -k1' [A]' [E] + k2' [A|E] + k3' [A|E]$$

where E is the enzyme and $A|E$ is the substrate-enzyme complex.

In general, ODE representations of biochemical pathways are highly non-linear in nature, requiring numerical rather than analytical solutions. *d[species]/dt* reflects that this species actually be affected by concentration / time. This, and the fact that such models are often stiff, means that numerical solvers need to employ small time steps in order to maintain accuracy and avoid unstable solutions. This can result in relatively long computational times.

Enzymes serve a wide variety of functions inside living organisms. They are indispensable for signal transduction and cell regulation, often via kinases and phosphatises [16]. Mass action kinetics are often used for modelling reactions within signalling pathways whereas Michaelis-Menten kinetics are often used in modelling 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 [17]. In the following equations $[E_T]$ is the total enzyme concentration, V is the reaction velocity, V_{max} is the maximum reaction velocity, and k_M is the Michaelis constant, which is the substrate concentration required for an enzyme to reach one-half its maximum velocity.

$$V = V_{\max} \times \frac{[A]}{k_M + [A]}$$

With the total enzyme concentration $[E_T]$ and the equation

$$k_{cat} = k3 = \frac{V_{\max}}{[E_T]}$$

we can write the differential equations describing the consumption of the substrate and production of the product as the following:

$$\frac{d[A]}{dt} = -\frac{d[B]}{dt} = -k_{cat} \times [E_T] \times \frac{[A]}{k_M + [A]}$$

Thus, the advantage of using the Michaelis-Menten kinetics is that it enables a single differential equation to describe the enzymatic reaction.

Although ODE-based modelling is one of the most widely used methods, it has one major drawback, which is that it is reliant on high-frequency sampling and parameter data being available, such as kinetic rates and absolute initial concentrations. However, much data generated by biologists are not directly amenable to modelling. Furthermore there is very little standardisation of measurements, data from different laboratories can be compared only in a semi-quantitative or qualitative fashion [17]. Thus, several alternative methods to estimate missing parameter data in an ODE-based model were developed, such as FBA (Flux Balance Analysis) [18] and MCA (Metabolic Control Analysis) [19].

There are alternative methods to ODEs that can be used to model and analyse biological systems. Stochastic modelling approaches are based on representing the individual behaviour of molecules and hence variability in the overall behaviour of a biological system. For example, Resat, et al. [20] developed a probability weighted-dynamic Monte Carlo stochastic simulation, which was an integrated model of both the trafficking and signalling 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. Execution speed of such large scale Monte Carlo simulations (and also large parameter sweep applications) can potentially be increased through Grid and Cloud Computing and the use of a purpose-built distributed computing middleware like Nimrod/G [21]. However, in this paper, we focus on ODE methods.

2.3 Computational Tools

As we have introduced in this paper, modelling efforts with systems biology typically lead to the simulation and analysis of models containing an enormous number of components and associated interactions. Non-linear mathematical models of biological processes and systems require tools that can provide powerful numerical analysis methods to investigate behaviour. A number of software tools are available for computational modelling based on ODEs and use the Systems Biology Markup Language (SBML) [22]. SBML is a computer-readable file format and can be used to represent models of metabolism, cell-signalling, and many other biochemical networks and biological processes. SBML-based software tools [23], include Cell Designer [24], Copasi [25], E-Cell [26], Jarnac combined with JDesigner [27]. In addition, Mathematica [28] and MATLAB® [29] are also widely used modelling tools. However, those tools do have some noticeable drawbacks. For example, E-Cell and JDesigner have been developed for the Windows operating system and are not platform-independent; also E-Cell cannot perform parameter estimation. Most importantly, no published work from these tools specifically addresses large-scale biochemical model simulation and analysis, such as model parameter estimation, parameter scan, gene knockdown *in-silico* and drug intervention *in-silico* using high performance computing. In contrast, the SIMAP Utility has been designed to be platform independent and to enable cross-host coarse-grained parallel execution of simulations in order to facilitate parameter scanning sensitivity analysis and parameter fitting.

3. DESKTOP-BASED GRID COMPUTING

The terms desktop personal computer (PC)-based grid computing and desktop grids in literature have been used to refer to the aggregation of non-dedicated, decentralised, commodity PCs connected through a network and running (mostly) the Microsoft Windows operating system [13]; a system that maximises an enterprise's return on investment for desktop PC by harnessing the CPU cycles for servicing large computations [11]; and a system that is motivated by the exponential growth of global computer ownership, local networks and Internet connectivity, together with the underutilisation of PCs in both organisations and home, and having the objective of harnessing the desktop PCs connected over the Internet [30]. Such grids have the potential of harvesting the commonly available idle computing resources of desktop PCs for processing of parallel, multi-parameter applications which consist of many instances of the same computation with its own input parameters [31]. As discussed in the preceding sections, a parameter scan application in the SIMAP Utility runs one simulation for each parameter combination. It is therefore considered appropriate to investigate the 'grid-enabling' of the SIMAP Utility using middleware for desktop grids to facilitate the distribution of thousands of parameter scanning jobs over the desktop grid resources.

Examples of desktop grid middleware include the Digipede Network [32], Condor [12], and BOINC [33]. The Digipede Network is a commercial desktop grid system that is targeted at enterprises and is marketed as being able to "improve the scalability and speed of (your) most compute-intensive, transaction-intensive, and data-intensive applications". Condor and BOINC middleware are freely available for download and use. However, it can be argued that key differences in the underlying middleware architectures, make Condor more suitable for use in large organisations and SMEs wherein the system can be used for purposes similar to those advertised by Digipede Network and other such providers of commercial desktop grid systems. BOINC, on the other hand, is suitable for a variant of desktop grid computing termed *Public Resource Computing (PRC) or volunteer computing*. PRC utilises of millions of desktop computers primarily to do scientific research [33]. The participants of PRC projects are volunteers who contribute their PCs to science-oriented projects such as SETI@home [34] and Climateprediction.net [35].

In our research we initially chose Condor due to the sophistication of its batch processing system and the flexibility of job allocation schemes. BOINC was also attractive, however, at the time of this work it was felt that the "pull" scheduling method used by BOINC was too inefficient for the number of PCs we had available and that the job registration scheme was too restrictive. This is unsurprising as Condor was effectively developed for use within an enterprise and BOINC for the open Internet. Our decision was also based on our experiences in implementing a BOINC-based desktop grid system [36]. Note that recent developments by the European Desktop Grid Initiative (EDGI) and the SZTAKI desktop grid project [37] have made significant additions to BOINC and we intend to port SIMAP to this architecture for comparison in future work.

Condor is an opportunistic job scheduling system that is designed to maximise the utilisation of workstations through identification of idle resources and scheduling background jobs on them [12]. A collection of such workstations is referred to as a Condor pool. Multi-core PCs are increasingly available in workplaces, and Condor exploits these multiple cores transparently. Condor uses the matchmaking framework and a language called *ClassAds* that matches the "requests for resources" with the "offers of resources" [38]. The Condor scheduling system allows the cross comparison of characteristics and requirements specific to the *jobs* and the *resources*, and this cross comparison is done on a per-host basis. The alternative to this is scheduling on a per-queue basis, as is usually done in a traditional batch/cluster system, wherein all jobs in a queue have the same general properties and they are executed on homogenous systems. The per-host scheduling mechanism allows for heterogeneous systems, as are common in desktop grids, to be accessed through the same grid interface (in this case, Condor).

Condor allows end-users to *submit* jobs and to *query* job status using two alternative mechanisms: (a) through use of a submit description file, and (b) through use of programming APIs that are exposed by Condor as Birdbath Web Services [39]. The latter approach can integrate Condor capabilities into existing software, and this is the approach used by us to integrate the SIMAP Utility with Condor (through a Java-based job manager utilising Condor Web Services).

4. DESKTOP GRID-ENABLED SIMAP UTILITY

In this section, the system architecture of the SIMAP Utility and its grid-enabled version are described.

4.1 Utility architecture

Specifically, the SIMAP Utility is a platform-independent software environment for biomodel engineering [**Error! Bookmark not defined.**], supporting the modelling of biochemical networks, and also the simulation and analysis of the dynamic behaviour of biochemical models. It uses a modular architecture that allows other developers to easily plug-in various components and to update them without reinstalling the whole tool. The system has been developed using Java technology and can be run on many platforms that support JRE (Java Runtime Environment 1.5.x or higher). Netbeans with SWING was used for the User Interface design, and the Utility provides a Graphical User Interface (GUI) which allows the user to import, create, edit and export the biochemical models conforming to the SBML standard [40]. LibSBML [41] has been used to read, write, manipulate, translate, and validate SBML files and data streams. The Concurrent Versions System (CVS) design helps users to keep track of the version history of their SBML models during construction and subsequent modification. The core of the utility comprises the SBML ODE Solver Library (SOSlib) [42].

The tool can compute changes of species concentrations over time with particular parameter values by simulating the SBML model numerically with SOSlib. The simulation results can be presented in two ways: plots and report text files. The system architecture of the SIMAP Utility is shown in Figure 2. The SIMAP Utility consists of several plug-in modules, which include the Data Management module, Simulation and Analysis Tools module, and a grid Access Point. Biochemical models and relevant data are stored in a MySQL-driven database, which is behind a firewall. JDBC has been used to access and communicate with the database from the Data Management module in the utility. The Data Management module allows the user to easily maintain, configure, view and modify the database. The Simulation and Analysis Tools module includes a set of computational modules for simulating and analysing biochemical models. These are an ordinary differential equations-based simulator, a sensitivity analyser, a parameter scanner, a model fitting module, a gene knockdown analyser and a model logic checker. The integration of the utility via the Grid Access Point and the Condor pool is described in the following sections. The client side can either connect to an existing Condor pool and MySQL-driven model database based in Brunel University (subsequent to the necessary permissions been granted by the network administrator), or use its own servers (MySQL DB and Condor pool), which needs to be configured with SIMAP web service.

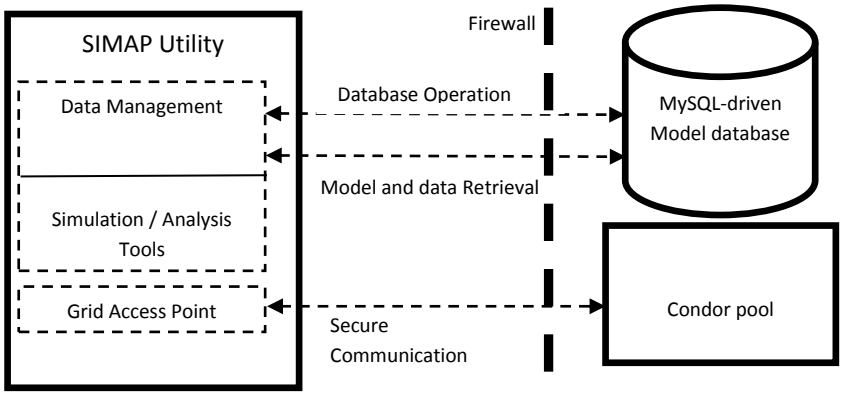

Figure 2: The SIMAP Utility consists of several plug-in modules which include the Data Management module, the Simulation and Analysis Tools module and a Grid Access Point.

4.2 SIMAP Utility Requirements for grid

In order to integrate grid systems with the SIMAP Utility, we took into account the future need for easy migration of the whole software to different user environments. Based on our experience with BioNessieG, the key questions were how to minimise work to port the existing tool to grid technologies and how to make the use of such technologies straightforward for end users.

Ideally, the SIMAP Utility would need basic information, such as the grid server name and an account, so that jobs can be submitted to the grid. When the jobs are completed, the Utility should be able to retrieve output files either through notifications or periodically via queries. Notification means that the Utility would need to open ports on

the machine to "listen" to these notifications. However, this solution is not allowed in many cases because of security policies or firewall settings. Owing to these security considerations, we decided to use a query-based approach and a straightforward request/reply protocol. This has been implemented in the Grid Access Point. The core of the Grid Access Point is a user friendly UI interface integrating with the client side APIs of the Condor based job manager, which is a Java-based high level facade (API packages) for submitting/querying/retrieving jobs between the SIMAP Utility and the Condor pool. This is now described.

4.3 Condor-Based Job Manager

The integration of the SIMAP Utility with the Condor-based job manager through client side APIs is illustrated in Figure 3. The Utility generates the required input file with a set of scanning parameters for each simulation job and wraps the job command (common to all jobs of a simulation) and the corresponding input file into a job request and submits it to the machine running the condor_schedd daemon using Birdbath's multiple transactions. For each transaction, Schedd.requestReschedule() is set in order to trigger schedd scheduling at once and NEGOTIATOR_INTERVAL is set to its default value. This condor_schedd daemon represents resource requests to the Condor pool and any machine that allows users to submit jobs needs to have a condor_schedd running; this is also referred to as the scheduler daemon and tracks all jobs submitted through a given machine [43]. The Utility queries the job status (by connecting to the *condor_schedd* daemon) after submission of job requests, and if any job completes correctly, it will retrieve the relevant output files. During implementation testing of this approach we identified two problems: *job submission* and *output file retrieval*. When the number of job requests is large, for example thousands, the submission procedure itself will take a lot of time (half hour or more in a standard Condor deployment). One option is to aggregate jobs together. However, we decided against this as we want to build a generic job submission system for this application where each parameter scan may contain totally different parameters and ODE formulas. We intend to investigate effective job grouping and scheduling in future work.

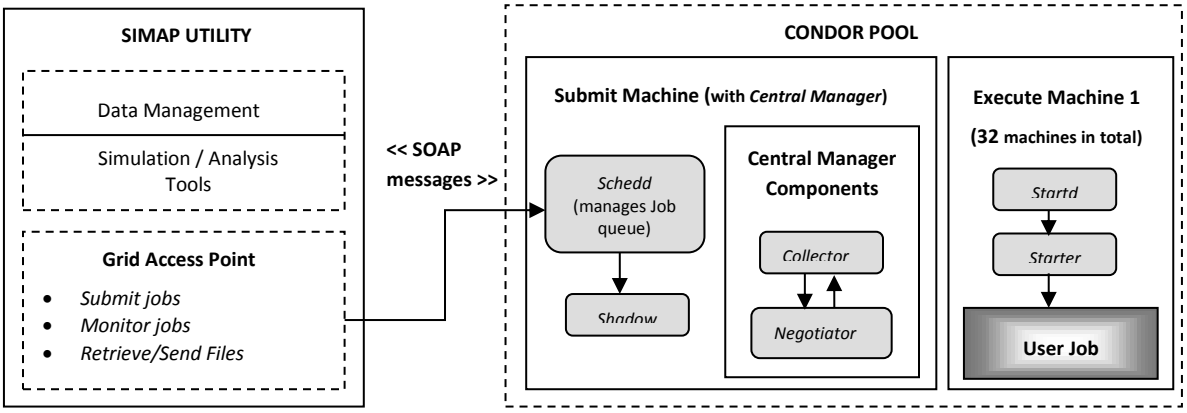

Figure 3: The SIMAP Utility Grid Access Point and CONDOR Pool integration. Condor daemons are indicated as grey oval text boxes.

In order to obtain output files we query the *scheduler* daemon. If the query frequency is not appropriate, useless workload may be put to the daemon by sending too many queries or a bottleneck of many output files waiting for retrieval may accumulate. This is further complicated by the small runtime of a single simulation - in the case study this is 20 seconds, fairly typical of a wide range of SBML models.

With regard to submission time, we investigated how to overlap submission time and job execution time, rather than to reduce submission time itself. In our first version of the job

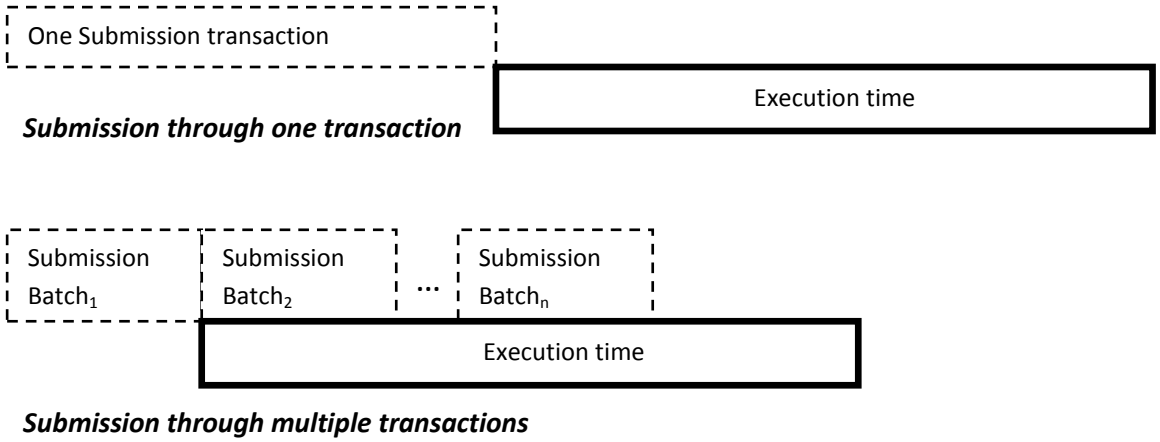

Figure 4: Comparison of two submission strategies

manager, we submitted all job requests in one transaction which means that the machine running the *condor_schedd* daemon could submit jobs to other machines to execute until all job requests are received. We improved this by submitting job requests in batches. This allowed the batched jobs to be distributed and to commence their execution before more jobs were received by the Condor pool. This is shown in Figure 4.

The retrieval of output files is dependent on the decision as to when to query job status (as mentioned earlier, job queues could be viewed by connecting to the *condor_schedd* daemon). In the case of thousands of jobs, we cannot retrieve all files at the time when all jobs complete because the size of thousands of output files can reach several Gigabytes. Therefore, we focus on how to overlap job execution time and file retrieval time. A simple approach is to check the status of each job in a loop and retrieve its output file at once if it completes until all jobs complete. The disadvantage of this approach is that we may make many useless queries regarding status of unfinished jobs. In order to improve the query efficiency, we developed a feedback control-based approach. Here the feedback used for each query is the jobs completion rate *R*. This is the number of completed jobs divided by the number of queried jobs. For example, consider the situation for 100 jobs in total, where a query checks the status of the first submitted 20 jobs. If 15 jobs complete, then the jobs completion rate for this query is 15/20. If *R* is smaller than a threshold *T*, it indicates that the current query frequency is faster compared with the job completion

speed. In this case, we increase the query interval by making the query process sleep longer before launching the next query and decrease the number of queried jobs in the next time. On the other hand, if *R* is larger than *T*, it indicates that the query frequency is slower compared with job completion speed so the query process should increase the rate of querying. Therefore, the interval between two queries relies on the job completion rate. The effect of our feedback control-based approach is to throttle job query frequencies by reacting to the job completion rate. The pseudocode algorithm is presented in Table 1 below.

<< Table 1 about here >>

5. EXPERIMENTS AND RESULTS

In order to investigate the performance of our SIMAP Utility with a Condor pool using the modifications described above, we have used a set of parameter scans on a real example, the *mammalian ErbB signaling pathway* [**Error! Bookmark not defined.**]. Briefly, in this pathway the ERbB1-4 receptor tyrosine kinases (RTKs) and the signalling pathways activate most core cellular processes such as cell division, motility and survival [44] and are strongly linked to cancer when malfunction due to mutations, etc. An ODE-based mass action ErbB model has been constructed and analysed by [**Error! Bookmark not defined.**] in order to determine what roles each protein plays and to ascertain how sets of proteins coordinate with each other to perform distinct physiological functions. The 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.

Our test bed comprised of a Condor pool with 32 desktop PCs. Each machine was configured with dual 2.1GHz cores and 2 Gigabytes RAM. A further PC configured with the same specification hosted the *condor_schedd* daemon. All the machines were connected to the network at 100Mbps. On a single PC of this specification using a single core a single simulation in our case study has an approximate run time of 20 seconds. The job size is around 1 MB as are the output results files. To investigate the performance we ran between 32 to 4096 simulation jobs. The time matrix and corresponding running time, speedup and efficiency graphs are shown below in Table 2 and Figure 5A, 5B and 5C. The time of each test includes jobs submission, execution, and results retrieval time. Each row shows the number of jobs per experiment and each column shows the number of machines used in the desktop grid.

<<Table 2 about here>>

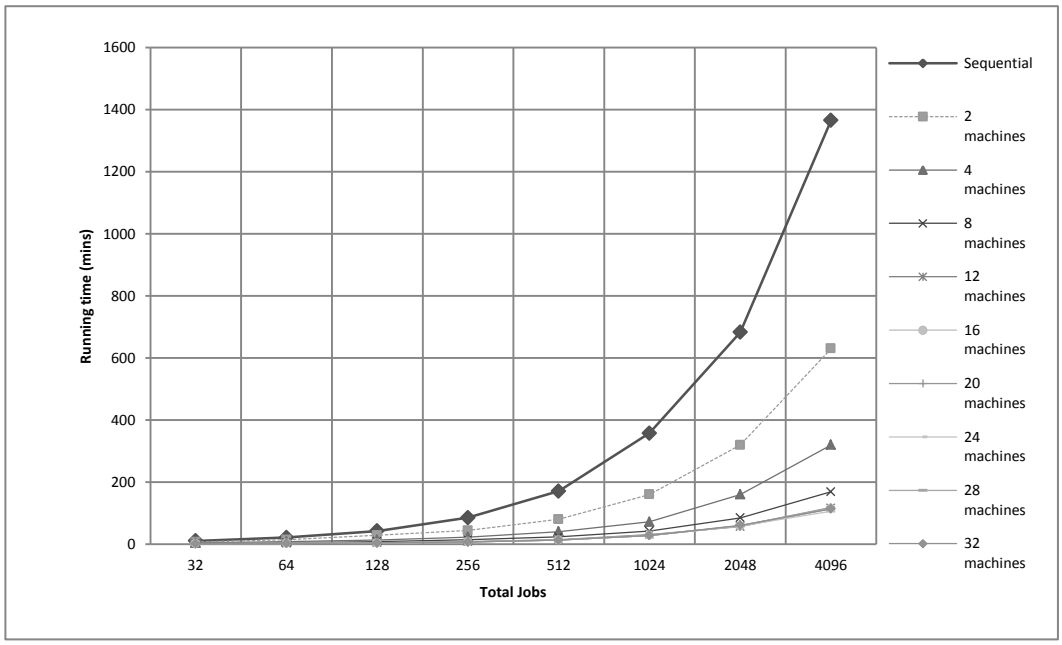

Figure 5A: The running time for different number of jobs on different numbers of machines (12, 16, 20, 24, 28 and 32)

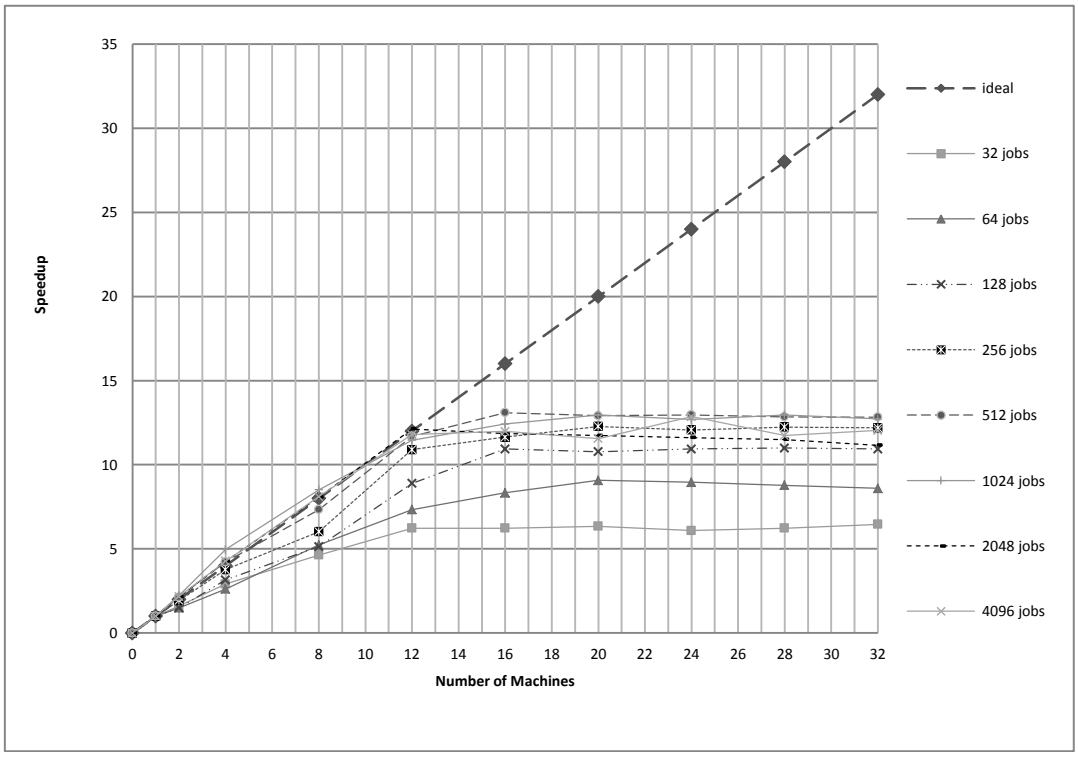

Figure 5B: Speedup of jobs on different numbers of machines

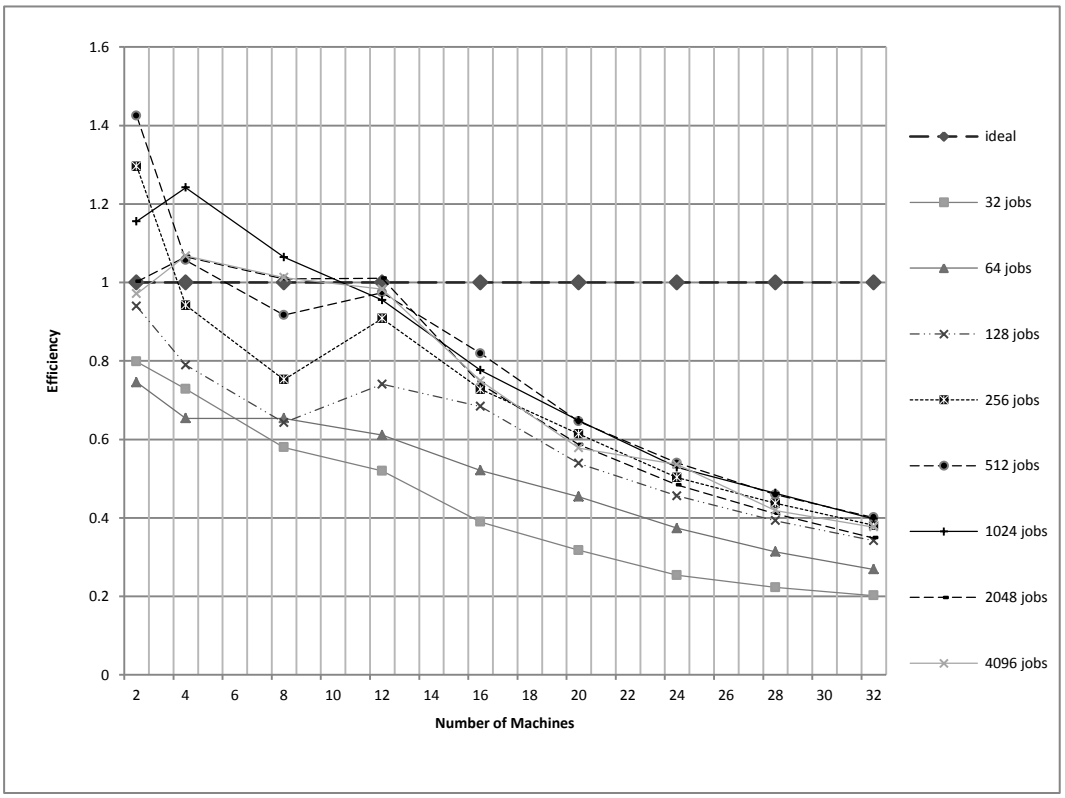

Figure 5C: Speedup of jobs on different numbers of machines

The test results in Figure 5B illustrate that in the case of 32 jobs and 64 jobs, the job submission overhead and having less than 4 jobs executing on each machine resulted in a lower overall speedup. The results also show that our Condor implementation achieved a linear speedup when less than 12 machines (<24 cores) were used; the peak speedup was around 12 with 12 machines (24 cores); for greater than 12 machines we notice a levelling out of speedup. Another conclusion from the results is that beyond 16 machines the system was unable to achieve further speedup, and thus additional machines were not utilised or not utilised efficiently. We believe reasons for this included the fact that we staged in an almost identical 1 MB file with each job and the job granularity used the default, so the Condor pool could not schedule these short jobs fast enough. In order to confirm our conclusion, we performed another three sets of tests to investigate, (a) the impact of job granularity, (b) the impact of an additional machine executing the *scheduler* daemon, and (c) the impact of job granularity against the additional *scheduler* daemon.

In the first test, to investigate the impact of job granularity, we ran 512 jobs on 16 machines and 32 machines respectively and varied the computation time of jobs with the same sizes of input/output files as before. To summarise the results from these further experiments, we define the relative speedup of a simulation between 16 and 32 machines as:

Speedupr1 = (Completion time with 16 machines) / (Completion time with 32 machines).

The graph in Figure 6 shows that when the job granularity increases, a better larger speedup is obtained by using 32 machines compared with 16 machines. This arguably

expected result implies that for smaller jobs the single machine executing the *condor_schedd* daemon might be the bottleneck despite our attempts to improve work distribution. In order to confirm this assumption, we performed a second test which investigated the effect of an additional *scheduler* running on a separate machine. We first submitted 512 original jobs (20s computation time) over 32 machines through one machine with one scheduler and then through two machines, each with one scheduler. The performance improvement by using two machines/two schedulers is shown in Figure 8. This demonstrates that using two *condor_schedd* daemons achieves a better speedup than just one.

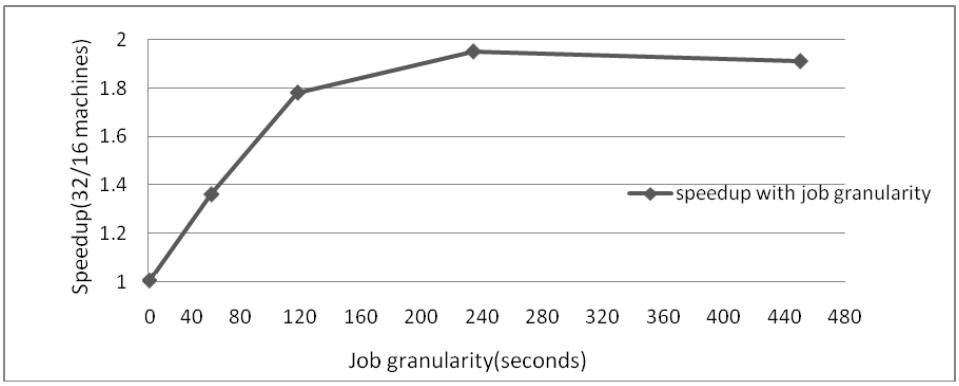

***Figure 6: Relative speedup (32/16 machines) with increasing job granularity (*using one scheduler daemon)**

In the third and final experiment, to test the impact of job granularity against the number of executing instances of the *condor_schedd* daemon, the number of machines was fixed at 32 and the job granularity was varied. The relative speedup of n machines with different agents is defined as

Speedupr2 = (Completion time with *n* machines using one *machine with one scheduler daemon*) / (Completion time with *n* machines using two *instances of scheduler daemon running on two different machines*).

The graphs in Figure 7 and 8 show that when the job granularity is relatively small the completion time can be decreased by using two scheduler daemons (compared with the use of only one scheduler executing on one machine). This indicates that when the computation time of jobs is short, and the input/output files are relatively large, the scheduler will be a bottleneck when the size of a Condor pool increases. However, as seen in Figure 9 below, when the job granularity gets larger, this effect is negated as we have shown that two *condor_schedd* daemons running on two separate machines do not give us more benefit than a single daemon in the case of 32 machines because one *condor_schedd* daemon, executing on one machine, can already deal with the jobs smoothly.

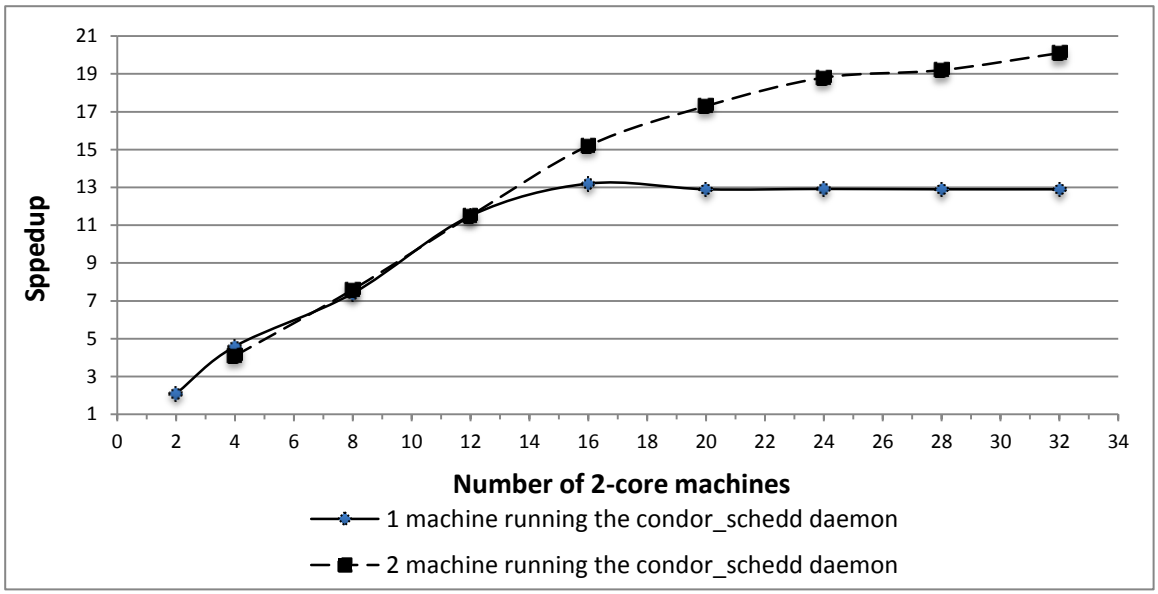

Figure 7: Performance comparison between one and two machines (each running the condor_schedd daemon)

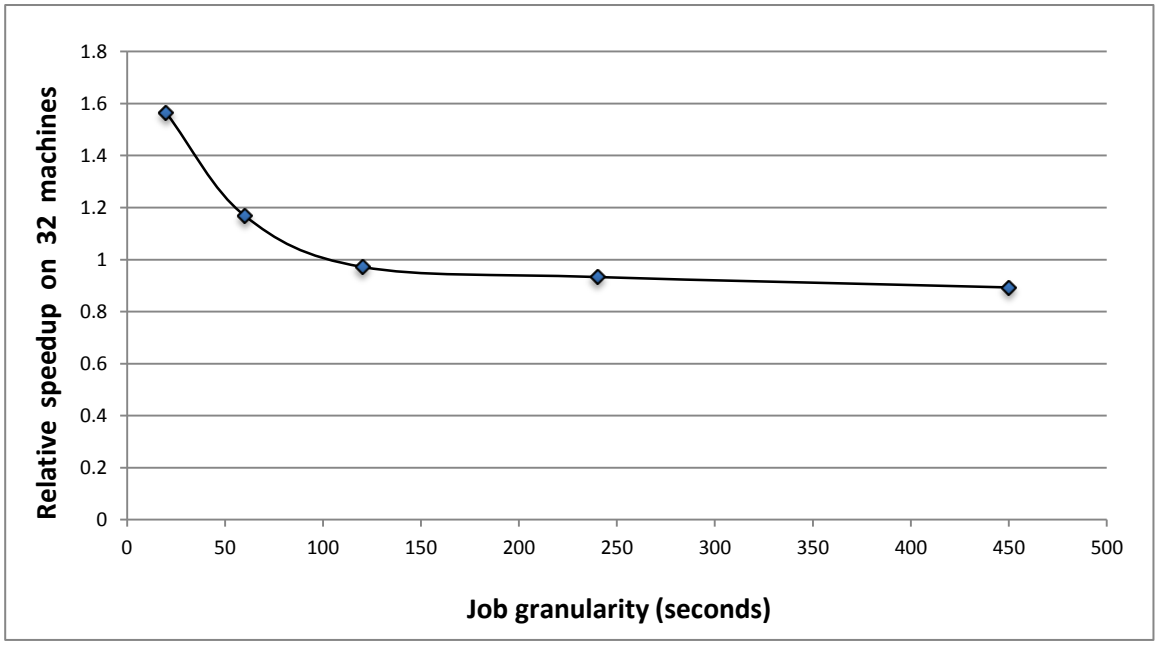

Figure 8: The graph shows the performance comparison between one and two instances of condor_schedd daemon (each executing on a separate machine) with job granularity.

6. DISCUSSION AND CONCLUSIONS

In this paper we have presented the problem of biochemical pathway analysis, our SIMAP Utility and its grid-based implementation using a Condor pool of desktop PCs. The results have shown reasonable performance with small job size and the implications

of using two scheduler daemons executing on two separate machines. Given that many biochemical models are of similar size these results will be useful for researchers who do not have access to High Performance Computing facilities but do have desktop PCs available.

More specifically, the test results show good speedups with 4096 biochemical simulation jobs (1MB input/output file each, 20s computation time) completed in 1.9 hours using 12 dual-core PCs (i.e. 24 cores) in a Condor pool compared with about 23 hours sequential running time. This gave an approximate speedup of 12 and did not increase when more machines were used. Further analysis demonstrated an expected result that speedup increased with job size as more machines were used. The investigation into the effect of increasing the number of condor_schedd daemons with small job size yielded an increased speedup as again more machines were used. This effect was negated with increased job size.

The key problem in this work is how to speed up the modelling and analysis of biochemical pathways such as the *mammalian ErbB signaling pathway*. In computing terms this becomes the problem of achieving an acceptable speedup with relatively small job size. Our approach to job management avoids bottlenecks created with continuous job submission and results collection by using a batched input and throttled query output. However, despite this the job size means that even though there are many Condor pool machines available the speeds at which jobs are sent and results returned limits the practical number of machines that can be used. In our case study an acceptable speedup is achieved up to 12 machines with two cores. Furthermore, we have shown that by adapting our implementation to use two scheduler daemons, better speedup can be achieved through the utilisation of more machines (demonstrated with 512 jobs increasing to 20 times speedup with 32 machines with two cores each). Having two or more schedulers also opens up the opportunity to experiment with multiple instances of the SIMAP Utility (client) whereby they concurrently submit jobs to the Condor pool through multiple job submissions machines (each executing a *condor_schedd* daemon). It will be interesting to compare this scenario with (a) job submission using one machine which supports multiple SIMAP Utility clients and (b) comparing these results with the performance results already presented in this paper (i.e., to the use of one/two scheduler daemons to support one SIMAP Utility client.) This work is planned for the future.

In terms of grid, future work in this area will address several interesting themes. We intend to investigate the effect on performance of using more scheduler daemons and also the effect of varying system settings such as the Condor negotiation_interval with job size. The outcome of this is to give a basis for profiling SBML job size to adapt a Condor pool to make use of the most machines to increase speedup. We are also developing a Web-based Portal that will allow models to be submitted remotely for analysis to support the rapidly growing Systems Biology community. Finally, we are also exploring other grid technologies and e-Infrastructures to use with the Utility.

In conclusion, the work presented in this paper adds to the growing corpus of literature focussing on the performance improvement aspects of e-Science applications on grid

infrastructures. The contribution of this paper is the empirical investigation of the performance of a compute-intensive application (the SIMAP Utility) over a Condor-based desktop grid solution using several scenarios. Some of our performance results corroborate those obtained from an existing study that used Condor through TeraGrid for grid computing of spatial statistics [45]. Their experimental results show some similar performance issues in which the speedup does not increase despite having available machines (however, we acknowledge that our Condor pool cannot be easily compared with the TeraGrid infrastructure, and therefore the inference to results presented in [45] may not be simplistically compared with the results of our experiments). There are several publications [46,47,48,49,50] related to scientific applications on Condor-based grids, but none deal with this problem of small jobs and thus it is expected that our analysis of relationships among job granularity, the number of Condor machines, and the number of Condor scheduler daemons should benefit applications requiring high throughput processing of relatively small job sizes.

ACKNOWLEDGEMENTS

This work has been partially supported by the *Simulation Modelling of the EGFR-MAP Kinase Pathway* (SIMAP) project which was funded by the European Commission under the FP6 - Information Society Technology Program. Elements of this work have appeared in a significantly shorter conference paper presented by the authors at the *2009 UK e-Science All Hands Meeting* [51].

REFERENCES

1 Sreenath SN, Cho KH, Wellstead P. Modelling the dynamics of signalling pathways. *Essays in Biochemistry: Systems Biology* 2008; **45**:1-28.

2 Cho CR, Labow M, Reinhardt M, van Ooostrum J, Peitsch MC. The application of systems biology to drug discovery. *Current Opinion in Chemical Biology* 2006; **10**(4): 294-302.

3 Breitling R. Donaldson RA, Gilbert DR, Heiner M. Biomodel Engineering – from structure to behaviour. *Trans on Computational Systems Biology XII*, Priami C et al (eds). Springer LNBI, 2010; 5945:1-12.

4 Foster I, Kesselman C. *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann: Los Altos, CA, 1998.

5 Liu X, Jiang J, Ajayi O, Gu X, and Gilbert D. Bionessie(G)- a grid enabled biochemical networks simulation environment. *Studies in Health Technology and Informatics*, 2008; **138**:147-157.

6 NGS. National Grid Service (project home). *http://www.ngs.ac.uk/* [Last accessed 03 September 2012].

7 ScotGrid. ScotGrid (project home). *http://www.scotgrid.ac.uk/* [Last accessed 03 September 2012].

8 Perera S, Herath C, Ekanayake J, Chinthaka E, Ranabahu A, Jayasinghe D, Weerawarana S, Daniels G. Axis2, Middleware for Next Generation Web Services. *Proceedings of the IEEE International Conference on Web Services*, Chicago, IL, Sept 2006; 833-840.

9 Foster I. Globus Toolkit Version 4: Software for Service-Oriented Systems. *Proceedings of the IFIP International Conference on Network and Parallel Computing*, Beijing, China, Nov-Dec 2005. Springer-Verlag LNCS, 2005; 3779: 2-13.

10 EC-funded FP6 Project. SIMAP: Simulation modelling of the MAP kinase pathway. *http://ec.europa.eu/information_society/activities/health/docs/publications/fp6upd2007/simap2007.pdf* [Last accessed 13 September 2012].

11 Kondo D, Chien A, Casanova H. Resource management for rapid application turnaround on enterprise desktop grids. *Proceedings of the 2004 Conference on Supercomputing (SC'04)*, Pittsburgh, Pennsylvania, Nov 2004; Paper 17.

12 Litzkow M, Livny M, Mutka M. Condor - a hunter of idle workstations. *Proceedings of the 8th International Conference of Distributed Computing Systems*, San Jose, CA, June 1988; 104-111.

13 Mustafee N, Taylor SJE. Speeding Up Simulation Applications Using WinGrid. *Concurrency and Computation: Practice and Experience* 2009; **21**(11): 1504-1523.

14 Kolch W. Meaningful relationships: the regulation of the Ras/Raf/MEK/ERK pathway by protein interactions. *The Biochemical journal* 2000; **351**(Pt. 2): 289-305.

15 Orton RJ, Sturm OE, Vyshemirsky V, Calder M, Gilbert DR, Kolch W. Computational modelling of the receptor-tyrosine-kinase-activated MAPK pathway. *The Biochemical journal*2005; **392**(Pt. 2): 249-261.

16 Hunter T. Protein kinases and phosphatases: the yin and yang of protein phosphorylation and signaling. *Cell* 1995; **80**(2): 225–36.

17 Gilbert DR, Breitling R, Heiner M, Donaldson RA. An Introduction to BioModel Engineering, Illustrated for Signal Transduction Pathways. *Membrane Computing*, Corne D, Frisco P, Păun G, Rozenberg G, Salomaa A (eds.). Springer LNCS, 2009; 5391: 13-28.

18 Bonarius HPJ, Schmid G, Tramper J. Flux analysis of underdetermined metabolic networks: The quest for the missing constraints. *Trends in Biotechnology* 1997; **15**(8):308-314.

19 Kacser H, Burns JA. The control of flux. *Symposia of the Society for Experimental Biology* 1973; **27**:65-104.

20 Resat H, Ewald JA, Dixon DA, Wiley HS. An integrated model of epidermal growth factor receptor trafficking and signal transduction. *Biophysical journal* 2003; **85**(2):730-743.

21 Abramson D, Giddy J and Kotler L. High Performance Parametric Modeling with Nimrod/G: Killer Application for the Global Grid? *International Parallel and Distributed Processing Symposium (IPDPS),* pp 520-528, Cancun, Mexico, May 2000.

22 Hucka M, Finney A, Sauro HM, Bolouri H, Doyle JC, Kitano H, et al. The systems biology markup language (SBML): a medium for representation and exchange of biochemical network models. *Bioinformatics* 2003; **19**(4):524-531

23 Pettinen A, Aho T, Smolander O, Manninen T, Saarimem A, Taattola K, Yli-Harja O, Linne M. Simulation tools for biochemical networks: evaluation of performance and usability. *Bioinformatics* 2005; **21**(3):357-363.

24 The Systems Biology Institute. CellDesigner™: A modeling tool of biochemical networks (project home). *www.celldesigner.org* [Last accessed 03 September 2012].

25 COPASI project. COPASI: biochemical network simulator (project home). *www.copasi.org* [Last accessed 03 September 2012].

26 E-Cell. E-Cell Project (project home). *www.e-cell.org/ecell* [Last accessed 03 September 2012].

27 Systems Biology Workbench. JDesigner (project home). *http://sbw-app.org/?page_id=56* [Last accessed 03 September 2012].

28 Wolfram Research, Inc. Wolfram Mathematica. *http://www.wolfram.com/mathematica/* [Last accessed 03 September 2012].

29 MathWorks, Inc. MATLAB®. *http://www.mathworks.co.uk/products/matlab/* [Last accessed 03 September 2012].

30 Luther A, Buyya R, Ranjan R, Venugopal, S. Alchemi: a .NET-based enterprise grid computing system. *Proceedings of the 6th International Conference on Internet Computing (ICOMP'05)*, Las Vegas, USA, June 2005; 269-278.

31 Choi S, Baik M, Hwang C, Gil J, Yu H. Volunteer availability based fault tolerant scheduling mechanism in desktop grid computing environment. *Proceedings of the 3rd IEEE International Symposium on Network Computing and Applications*, Boston, Massachusetts, Aug-Sept 2004; 366-371.

32 Digipede Technologies. The Digipede Network. *http://www.digipede.net/products/digipede-network.html* [Last accessed 03 September 2012].

33 Anderson D. BOINC: A System for Public-Resource Computing and Storage. *Proceedings of the 5th IEEE/ACM International Workshop on Grid Computing*, Pittsburgh, USA, Nov 2004; 4-10 .

34 Anderson D. P, Cobb J, Korpela E, Lebofsky M, Werthimer D. SETI@home: an experiment in public-resource computing. *Communications of the ACM*, 2002, **45**(11):56-61.

35 Christensen C, Aina T, Stainforth D. The challenge of volunteer computing with lengthy climate model simulations. *Proceedings of the First International Conference on e-Science and Grid Computing (e-SCIENCE '05)*, Melbourne, Australia, Dec 2005; 8-15.

36 Zhang J, Mustafee N, Saville J, Taylor, S.J.E. Integrating BOINC with Microsoft Excel: a case study. *Proceedings of the 29th Information Technology Interfaces Conference*, Dubrovnik, Croatia, June 2007; 733 – 738.

37 Balaton Z, Gombas G, Kacsuk P, Kornafeld A, Kovacs J, Marosi A.C, Vida G, Podhorszki N, Kiss T. SZTAKI Desktop Grid: a Modular and Scalable Way of Building Large Computing Grids. *Proceedings of the 2007 IEEE International Parallel and Distributed Processing Symposium*, Long Beach, California, March 2007; 1 - 8.

38 Thain D, Tannenbaum T, Livny M. Distributed computing in practice: the Condor experience. *Concurrency and Computation: Practice and Experience* 2004; **17**(2–4):323–356.

39 Chapman C, Goonatilake C, Emmerich W, Farrellee M, Tannenbaum T, Livny M, Calleja M, Dove M. Condor Birdbath-Web Service interfaces to Condor. *Proceedings of the UK e-Science All Hands Meeting*, Nottingham, UK, Sept 2005; 737-744.

40 Hucka M, Finney A, Bornstein B, Keating S, Shapiro B, Matthews J, Kovitz B, Schilstra M, Funahashi A, Doyle J, Kitano H. Evolving a lingua franca and associated software infrastructure for computational systems biology: the systems biology markup language (sbml) project. *Systems Biology* 2004; **1**(1): 41-53.

41 Bornstein BJ, Keating SM, Jouraku A,Hucka M. LibSBML: An API Library for SBML. *Bioinformatics* 2008; **24**(6):880–881.

42 Machn R, Finney A, Mller S, Lu J, Widder S, Flamm C. The SBML ODE solver library: a native API for symbolic and fast numerical analysis of reaction networks. *Bioinformatics* 2006; **22**(11):1406-1407.

43 Yi, S-Y, Livny, M. Extending the Condor distributed systems for mobile clients. *ACM SIGMOBILE Mobile Computing and Communications Review* 1999, **3**(1):38-46.

44 Citri A, Yarden Y. EGF-ERBB signalling: towards the systems level. *Nature Reviews Molecular Cell Biology* 2006, **7**(7):505-516.

45 Wang S, Cowles MK, Armstrong MP. Grid computing of spatial statistics: using the TeraGrid for Gi*(d) analysis. *Concurrency and Computation: Practice and Experience* 2008; **20**(14): 1697-1720.

46 He L, Dove M, Hayes M, Murray-Rust P, Calleja M, Yang X, Milman V. Developing Lightweight Application Execution Mechanisms in Grids. *Proceedings of UK e-Science All Hands Meeting*, Nottingham, UK, Sept 2006; 201-208.

47 Caton SJ, Rana OF, Batchelor BG. Distributed image processing over an adaptive Campus Grid. *Concurrency and Computation: Practice and Experience* 2009; **21**(3):321-336.

48 McCaulay DS, Sheppard R. PFASC – A Parallel Framework for Audio Similarity Clustering. *Proceedings of the 3rd TeraGrid 2008 Conference*, Las Vegas, NV, June 2008.

49 Downes P, Yaikhom G, Giddy JP, Walker DW, Spezi E, Lewis DG. High-performance computing for Monte Carlo radiotherapy calculations. *Philosophical Transactions of the Royal Society A* 2009; **367**(1897):2607-2617.

50 Price AR, Myerscough RJ, Voutchkov II, Marsh R, Cox SJ. Multi-objective optimization of GENIE Earth system models. *Philosophical Transactions of the Royal Society A* 2009; **367**(1898):2623-2633.

51 Wang J, Liu X, Mustafee N, Gao Q, Taylor SJE, Gilbert DR. Grid-enabled SIMAP Utility: Motivation, Integration Technology and Performance Results. *Proceedings of UK e-Science All Hands Meeting*, Oxford, UK, Dec 2009.

52 Gober MD, Wales SQ, and Aurelian L. Herpes simplex virus type 2 encodes a heat shock protein homologue with apoptosis regulatory functions. *Frontiers in Bioscience*, 10, 2788-2803, September 1, 2005

TABLES

Table 1: Pseudocode of job submission, query and execution

```
numberOfJobsPerQuery = default value;
numberOfCompletedJobs = 0;
threshold = default value;
While (unfinished job list is not empty) {
  for the first numberOfJobsPerQuery jobs in the list {
    if they complete then remove them from the list and
    increase the value of numberOfCompletedJobs.
  } if (numberOfCompletedJobs/numberOfJobsPerQuery < threshold) {
    the query execution frequency is faster compared with
    completed jobs so the query process should slow down by
    sleeping longer and the numberOfJobsPerQuery should decrease;

  } else {
    the query execution frequency is fine or probably
    slower compared with completed jobs so the query
    process should speed up by sleeping shorter
    and the numberOfJobsPerQuery should increase;
  }
}
```

Table 2: Running time (minutes) of different number of machines and different sizes of jobs (the running time includes job submission time)

		Machines									
		1	**2**	**4**	**8**	**12**	**16**	**20**	**24**	**28**	**32**
Total Jobs	**32**	10.66	6.68	3.66	2.3	1.71	1.71	1.68	1.75	1.71	1.65
	64	21.33	14.31	8.16	4.08	2.91	2.56	2.35	2.38	2.43	2.48
	128	42.66	28.46	13.51	8.3	4.8	3.9	3.96	3.9	3.88	3.9
	256	85.33	44.18	22.65	14.18	7.83	7.33	6.95	7.07	6.97	7
	512	170.67	79.95	40.38	23.27	14.6	13.03	13.22	13.17	13.28	13.3
	1024	357.6	160.2	72	42	31.2	28.8	27.6	28.19	27.6	28.1
	2048	682.8	319.2	160.2	84.6	56.3	57.6	58.2	58.8	59.4	61.2
	4096	1365.6	630	319.8	168.6	115.8	114	118.2	106.2	116.4	113.4