COMPUTING SCIENCE

A Framework for Dynamically Generating Predictive Models of Workflow Execution

Hugo Hiden, Simon Woodman and Paul Watson

TECHNICAL REPORT SERIES

No. CS-TR-1396 October 2013

Bibliographical details

HIDEN, H., WOODMAN, S., WATSON, P.

A Framework for Dynamically Generating Predictive Models of Workflow Execution
[By] H. Hiden, S. Woodman and P. Watson
Newcastle upon Tyne: Newcastle University: Computing Science, 2013.

(Newcastle University, Computing Science, Technical Report Series, No. CS-TR-1396)

Added entries

NEWCASTLE UNIVERSITY
Computing Science. Technical Report Series. CS-TR-1396

Abstract

The ability to accurately predict the performance of software components executing within a Cloud environment is an area of intense interest to many researchers. The availability of an accurate prediction of the time taken for a piece of code to execute would be beneficial for both planning and cost optimisation purposes. To that end, this paper proposes a performance data capture and modelling architecture that can be used to generate models of code execution time that are dynamically updated as additional performance data is collected. To demonstrate the utility of this approach, the workflow engine within the e-Science Central Cloud platform has been instrumented to capture execution data with a view to generating predictive models of workflow performance. Models have been generated for both simple and more complex workflow components operating on local hardware and within a virtualised Cloud environment and the ability to generate accurate performance predictions given a number of caveats is demonstrated.

About the authors

Hugo Hiden is the software development lead on the SIDE project. His main interests are the development of the e-Science Central system, cloud based data analysis systems and workflows. Before this, he was the Technical Director of the UK North Eastern Regional e-Science Centre (NEReSC) at Newcastle University where his primary responsibility was the technical co-ordination of the various research projects within the centre. Prior to this, he spent 6 years in industry developing advanced data integration, analysis and modelling tools at Avantium Technologies and GSE Systems Inc. He holds a PhD in the application of Genetic Programming to chemical process data analysis and has published numerous papers on the subjects of process monitoring, computer security and collaborative R&D.

Simon Woodman is a Research Associate in the Scalable Information Management group. His interests include workflow enactment, especially those systems that allow distributed coordination; service description languages, data provenance, social networking and collective intelligence. He is currently developing an e-Science system to allow secure data sharing and workflow enactment for the neuroscience domain. Simon was awarded his PhD in 2008 for his thesis on "A Programming System for Process Coordination in Virtual Organisations", supervised by Prof. Santosh Shrivastava. During his undergraduate studies, Simon worked at IBM Hursley for a year as part of a team developing the CICS Information Centre. Simon received his BSc (hons) from Newcastle in 2002.

Paul Watson is Professor of Computer Science and Director of the Digital Institute. He also directs the £12M RCUK-funded Digital Economy Hub on Social Inclusion through the Digital Economy. He graduated in 1983 with a BSc in Computer Engineering from Manchester University, followed by a PhD on parallel graph reduction in 1986. In the 80s, as a Lecturer at Manchester University, he was a designer of the Alvey Flagship and Esprit EDS systems. From 1990-5 he worked for ICL as a system designer of the Goldrush MegaServer parallel database server, which was released as a product in 1994. In August 1995 he moved to Newcastle University, where he has been an investigator on research projects worth over £30M. His research interest is in scalable information management with a current focus on Cloud Computing. Professor Watson is a Chartered Engineer, a Fellow of the British Computer Society, and a member of the UK Computing Research Committee.

Suggested keywords

CLOUD COMPUTING
PREDICTIVE MODELLING
PERFORMANCE ANALYSIS

TECHNICAL REPORT SERIES

No. CS-TR-1396 October, 2013

A Framework for Dynamically Generating Predictive Models of Workflow Execution

H. Hiden, S. Woodman and P. Watson

Abstract

The ability to accurately predict the performance of software components executing within a Cloud environment is an area of intense interest to many researchers. The availability of an accurate prediction of the time taken for a piece of code to execute would be beneficial for both planning and cost optimisation purposes. To that end, this paper proposes a performance data capture and modelling architecture that can be used to generate models of code execution time that are dynamically updated as additional performance data is collected. To demonstrate the utility of this approach, the workflow engine within the e-Science Central Cloud platform has been instrumented to capture execution data with a view to generating predictive models of workflow performance. Models have been generated for both simple and more complex workflow components operating on local hardware and within a virtualised Cloud environment and the ability to generate accurate performance predictions given a number of caveats is demonstrated.

Printed and published by Newcastle University,
Computing Science, Claremont Tower, Claremont Road,
Newcastle upon Tyne, NE1 7RU, England.

A framework for dynamically generating predictive models of workflow execution

Hugo Hiden
School of Computing Science
Newcastle University
Newcastle upon Tyne, UK
hugo.hiden@ncl.ac.uk

Simon Woodman
School of Computing Science
Newcastle University
Newcastle upon Tyne, UK
simon.woodman@ncl.ac.uk

Paul Watson
School of Computing Science
Newcastle University
Newcastle upon Tyne, UK
paul.watson@ncl.ac.uk

ABSTRACT

The ability to accurately predict the performance of software components executing within a Cloud environment is an area of intense interest to many researchers. The availability of an accurate prediction of the time taken for a piece of code to execute would be beneficial for both planning and cost optimisation purposes. To that end, this paper proposes a performance data capture and modelling architecture that can be used to generate models of code execution time that are dynamically updated as additional performance data is collected. To demonstrate the utility of this approach, the workflow engine within the e-Science Central Cloud platform has been instrumented to capture execution data with a view to generating predictive models of workflow performance. Models have been generated for both simple and more complex workflow components operating on local hardware and within a virtualised Cloud environment and the ability to generate accurate performance predictions given a number of caveats is demonstrated.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous; D.2.8 [**Software Engineering**]: Metrics—*complexity measures, performance measures*

General Terms

Performance, Measurement

Keywords

Cloud Computing, Predictive Modelling, Performance Analysis

1. INTRODUCTION

The e-Science Central cloud data analytics platform is an Open Source multi-user system for the storage and analysis of a wide range of data sets [5]. It provides data storage and sharing facilities along with a workflow engine designed to operate at large scale within a hosted cloud environment.

Users upload their data and then process it either using a workflow they have constructed or by selecting a workflow that another user has shared. When these workflows are executed, they are queued for execution by one of several servers dedicated to the task of running workflows (Workflow Engines). In order to maintain a fair access to these Workflow Engines, it is important to balance the calculation time between all of the users of the system and to schedule workflow execution over a range of resources. Predicting the runtime of any given workflow, therefore, is a vital first step towards achieving this and is also critical for the provision of the estimated cost and storage requirements associated with executing specific workflows. Additionally, because workloads on the system can require the addition of significant numbers of workflow engines, a prediction of the likely end time for these workloads would enable workflow engines to be brought online proactively and shut down in a more timely manner.

This paper describes a system which captures live performance data and uses it to build a suite of models that can be used to predict various characteristics of workflows. These models can be updated on demand or in response to the collection of a sufficient quantity of additional logging data.

Although the performance modelling work presented in this paper was carried out using the e-Science Central platform, the methodology and indeed the code developed is applicable to any system that can be instrumented to produce the correct form of performance logging data. The contributions of this paper fall largely into three areas:

1. Estimating the future performance of software components based on predictive models trained on historical data.

2. The ability to combine a number of models of individual unit operations in the same order as they would be invoked in arbitrary workflows and the ability to predict the likely performance of these workflows.

3. A system to capture and store historical performance data and generate suites of predictive models from it.

2. RELATED WORK

A significant quantity of research has been performed in an attempt to predict the execution time of software in order to improve scheduling, particularly within Grid and HPC

environments. Some researchers have considered complete applications [3, 8, 9] whilst others have attempted to decompose these into components as we do [4, 6]. The work presented in by Duan [4] is of particular interest as it closely resembles ours but focuses on Grid deployment scenarios. One of the key differences is our use of the 'Panel of Experts' pattern [15] to generate multiple predictive models of each service rather than their use of a Radial Basis Function neural network. We have found that it is imperative to include multiple modelling techniques as some components will model significantly better or worse depending on the technique used.

The work by Cushing [2] discusses how to scale Map-Reduce style problems based on the expected execution time. The aim here is to reduce the overall computation time by dedicating more resources to components which are expected to take a longer duration. In addition they aim to prevent starvation of future components due to a previous one having not completed execution. We do not restrict our execution pattern to Map-Reduce, although we are able to construct such a pattern using e-Science Central workflows.

Within the context of cloud computing, Roy [10] use autoregressive moving averages to predict the current workload of a system. However, they are concerned with scaling cloud architecture to minimise response time in a web application rather than scientific workflow applications which exhibit different characteristics.

Work by Miu [7] considers other features of the input data other than the size when generating predictive models for algorithms such as those found in the Weka toolkit[1]. This work is, in some respects, more ambitious than ours but also more expensive in terms of computing power required and knowledge of the algorithm being modelled. We consider each service as a black box and make no attempt to include features other than the size of the input data and the code configuration parameters within our models. In future we would like to encompass other features of the input data but this would increase the complexity of the system. In addition we have found that many of the services used within scientific workflow applications deployed within e-Science Central to date are amenable to simple analysis based on the size of the input data including physical activity analysis and image processing algorithms [17, 14, 18].

The work around the Prophesy project to develop a general purpose performance analysis system is most similar to our approach [13]. However, they focus on lower level instrumentation of the source code than we do where our 'building blocks' are workflow services. Further, they require the source code to be available in order to insert the performance monitoring hooks during the compilation phase. Instead, our work has focused on instrumenting the execution environment to allow any code deployed into the environment to benefit from performance capture.

The literature around the Prophesy system also details some approaches to leveraging multiple predictive models to generate a prediction for a larger unit of work [12]. As their work is principally aimed at lower level functions with more complex inter-relationships they generate what can be considered to be a cross-product of model relationships between each 'kernel' of computation. Our approach differs in that we only consider the effects of the data transferred from one component to another and, given that we are dealing with higher level components without such inter-relationships, we do not need to compute the cross-product of all components. We also show that it is feasible to use the output of one predictive model as the input to another whereas other systems simply consider the summation of the predictions from each model [11].

3. ARCHITECTURE

e-Science Central is a portable 'platform-as-a-service' that can be deployed on a variety of hardware platforms ranging from a Raspberry Pi to public/private clouds and supercomputing infrastructures. Cloud computing has the potential to give scientists the computational resources they need, when they need them. However, cloud computing does not make it easier to build the often complex, scalable secure applications needed to support scientists. e-Science Central was designed to overcome these obstacles by providing a platform on which users can carry out their research, and build high-level applications. The architecture of e-Science Central is described in [5] but at the core it combines three technologies – Software as a Service (so users only need a web browser to do their research), Social Networking (to support sharing and community interaction) and Cloud Computing (to provide storage and computational power). Using a web browser, users can upload data, share it in a controlled way with colleagues, and analyse the data using either a set of pre-defined blocks, or their own, which they can upload for execution and sharing. A range of data analysis and programming languages are supported, including Java, Javascript, R and Octave. From the point of view of users, this gives them the power of cloud computing without them actually having to manage the complexity of developing cloud-specific software – they can create blocks in a variety of languages, upload them into e-Science Central, and have them run transparently on the cloud.

The blocks which are hosted within e-Science Central can be combined into larger units of computational work, workflows, which compose multiple re-usable components to perform data analysis. Workflows in e-Science Central only include data flow – control flow, outside of each block, is not provided. However, as workflows are able to execute other workflows a simple recursive structure can be described but the termination condition must be expressed within a specialised block. We have found through extensive work with scientists in varying application areas that data flow alone is sufficient to suit their needs [1]. The benefit of supporting pure data flow, as we will see in Section 4, is that it greatly simplifies the process of generating predictive models of the workflow execution time. Workflows are created using an online editor which supports drag and drop workflow creation from a palette of both common and user supplied blocks. Blocks themselves can be created using another online editor or common software development tools such as Maven[2].

Versioning is an integral storage feature in e-Science Central, allowing users to work with old versions of data, blocks and workflows. All objects (data, files and workflows) are stored in files through a virtual filestore driver than can be mapped onto a range of actual storage systems including standard local and distributed filesystems, and Amazon S3. When a file is stored, if a previous version exists then a new

[1] http://www.cs.waikato.ac.nz/ml/weka/

[2] `http://maven.apache.org`

one is automatically created. From the perspective of modelling blocks, this allows us to directly compare the execution time of different versions of the same block and use the previous version when insufficient data is available for the current version of the block (see Section 4.2). In addition to each block being versioned, they may be parameterised with different runtime configurations. Such parameters include the source of the data to import, initialisation parameters for algorithms and other runtime settings. These parameters are included in the data collected for model construction wherever possible (specifically when the parameter has, or can be represented as, a numerical, non-categorical value).

Workflows are enacted by a set of workflow engines which typically run on separate machines from the main e-Science Central server. A workflow within e-Science Central differs from the more traditional workflow model in that the data flow it represents is executed entirely on the workflow engine and does not typically make calls to external services. The individual blocks within the workflow are separate code libraries that are downloaded to the workflow engine (or potentially installed using the operating system package manager) where they then operate upon the data generated by the workflow execution. Because the workflow engines themselves perform all of the computational tasks required in order to execute a workflow, adding more workflow engines increases the processing power of the system. Depending on the characteristics of the workload, we have been able to support over 200 workflow engines using a single main server. Each workflow server executes a workflow by analysing the directed acyclic graph which represents it. From this, a sequence of block executions is constructed which allows the engine to execute them in an order which ensures that the input data is available for each block within the workflow when required.

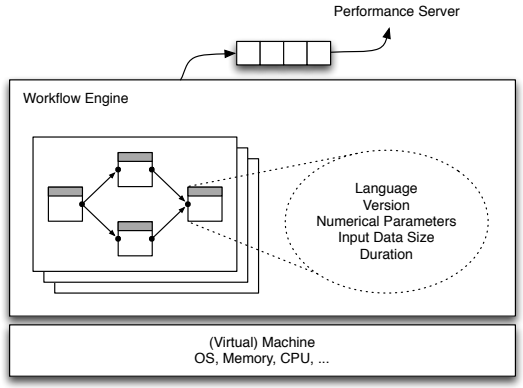

Figure 1: Data Capture

In Figure 1, each time a workflow block is executed, a provenance message is sent to a queue for persistence in the provenance database. Currently, the provenance message for a workflow block includes details of the code used within the block, the data sets the block operated on, the configuration of the block and the user running the workflow. This information was selected because it contains sufficient data to recreate the actions performed on a given piece of data within the system [16]). During the development of the performance modelling system described in this paper, the provenance capture platform was extended to include performance data which was stored in a separate provenance database. The approach adopted was to log all of the available parameters of the execution of blocks within workflows that could possibly be used to predict performance. Specifically, the following performance attributes were logged in the performance database:

Execution time This is the total time taken for a block within the workflow to execute. The time is measured from the time that the process executing the block is started to the time it terminates. This time measurement does not include the time taken by the workflow engine to deploy any code that the block depends on and is therefore a direct measurement of code execution time and not a combination of code execution time and workflow management overhead.

Block settings Each block within the workflow can be configured with a number of settings. These settings are block specific and can have a significant influence on the time taken for a block to run. For example, a modelling block might include a parameter for specifying the model complexity. This would have a dramatic effect on the block execution time. The performance capture system logs any numerical block property in the performance database. References to data stored within the e-Science Central file system are treated slightly differently in that the size of the document is logged as a parameter in the performance database. The data capture was limited to numerical properties in this case because the modelling tools selected do not operate on non-numerical data. If, however, classification algorithms that consider categorical data were found to yield useful predictions, additional block settings could be captured trivially.

Data volumes The volume of data consumed by each block is a critical parameter for modelling execution time and is captured in the performance database where it is linked to individual block inputs and outputs. Model building algorithms can then access the information about the total volumes of data passing through workflow blocks.

Machine characteristics The type of the machine executing workflow blocks is logged because it enables block executions to be grouped by machine type when building models. The actual machine data (CPU speed, memory type etc) is not used for modelling as it is impossible, with the current version of e-SC, to know in advance which engine within the execution pool will execute a given workflow. It can, however, be used to select the appropriate model to use to estimate workflow termination time once it has started and the exact characteristics of the selected workflow engine are known.

At the core of the architecture of the Performance Server, shown in Figure 2 is a Postgres database[3] which stores the raw data captured by the workflow engines during their operation. The data is consumed via a Java Messaging Service[4]

[3] `http://www.postgresql.org`
[4] `http://en.wikipedia.org/wiki/Java_Message_Service`

(JMS) queue which provides a queue to which multiple workflow engines can submit performance data, thus decoupling the Performance Server from the system (the e-Science Central Workflow Engine in this case) which is generating the performance metrics. In order to generate the models we make use of the Apache Commons Maths 3 library[5] and a JavaEE application server to host the code. This simplifies access to the performance database and the provision of the various APIs which allow the data to be consumed by other systems. The models, once constructed, are stored in the database and allow predictions to be generated and comparisons between different models of same block to be made.

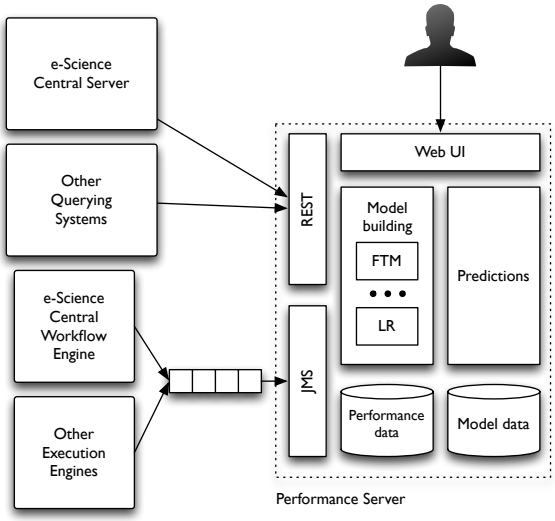

Figure 2: Architecture of the Performance Server

The actual selection of the parameters used in any model is delegated to the specific model building algorithms deployed within the system. This approach was adopted because the modelling system has been designed to build multiple predictive models for each workflow block, each of which is likely to include a different subset of the performance data contained in the database. The capture of this comprehensive set of parameters also allows models to be built of properties other than execution time. For example, the data captured could allow models to be built relating the physical RAM consumed by a block to the quantity of data processed.

There are two ways that the models and predictions can be consumed by other interested parties: a simple web application is provided to allow users to view performance data and generate predictions whilst a REST based API allows integration with the core e-Science Central server and other external systems.

Although currently we are only concerned with modelling performance data collected from e-Science Central workflows, the architecture is generic and can interface with other systems in order to generate and consume performance data. For instance, workflow enactors such as Taverna or Galaxy could be instrumented to submit performance data into the system. The only modification required would be to the logic for combining the predictive models generated for each block or action within the workflow. Indeed it is not limited to capturing workflow based execution data: any system which is able to log the performance characteristics of a task could submit data. In addition, we have integrated the prediction code into the e-Science Central workflow editor to allow real time feedback to users as they are creating their workflows.

[5] `http://commons.apache.org/math/`

4. MODELLING PERFORMANCE

Within e-Science Central, workflows can be considered as a linked set of individual software components (blocks) which act sequentially upon items of data. Execution proceeds in the following manner:

1. The workflow is analysed to discover all of the blocks that contain no input connections. These are defined as data source blocks that act to bring data to the server hosting the workflow engine.

2. Once the data source blocks have been identified, an execution thread is started which starts from the first data source block and propagates data through all of downstream blocks, which are executed in an order which ensures that all of the required input data items for each block have been produced by any linked upstream blocks.

3. Execution terminates once all of the possible execution paths from the data source blocks have been traversed.

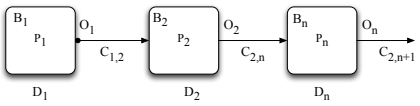

Figure 3: e-Science Central workflow structure

The basic structure of an e-Science Central workflow is illustrated in figure 3 which shows a number of connected blocks ($B_1 ... B_n$). Each of these blocks can contain a property set that defines its behaviour ($P_1 ... P_n$). The analysis pass of the workflow execution process will identify B_1 as the single data source block. The execution thread will first execute B_1 using the property set, P_1. This will take a period of time, D_1, and produce a piece of output data, O_1. This data item will be propagated to the second block in the workflow, D_2, along the connection, $C_{1,2}$. The second block, B_2, will then be executed using it's property set, P_2, and input data set, O_1. This process will take D_2 seconds.

From this it can be seen that the total actual execution duration for the workflow, D_{wf} can be expressed as:

$D_{wf} = \sum_{i=1}^{n} D_i$

To generate an estimated execution duration for the workflow, $\hat{D_{wf}}$, a summation of duration estimates for the individual workflow blocks is therefore required:

$\hat{D_{wf}} = \sum_{i=1}^{n} \hat{D_i}$

This approach is applicable to the e-Science Central workflow engine because it does not attempt to execute any blocks in parallel, so the total execution time can easily be calculated. For cases where workflow paths can be operated in parallel, the longest duration for each parallel path must be summed in order to predict the total execution time. In

order to generate a prediction of the execution duration for a particular block, $\hat{D_i}$, a relationship needs to be defined that relates execution duration to the various attributes of the block that can influence performance. In general the execution duration for a block will be a function of the input data size to the block, O_{i-1}, the actual code within the block and the block parameter settings, P_i (see Section 3). The estimated duration of any block within the workflow can therefore be calculated using:

$\hat{D_i} = f_{Di}(P_i, O_{i-1})$

where f_{Di} represents the predictive duration model for the i^{th} workflow block. From this, it follows that the total workflow duration can be predicted by:

$\hat{D_{wf}} = \sum_{i=1}^{n}(f_{Di}(P_i, O_{i-1}))$

Because the duration estimate for each block within the workflow is dependent upon the size of the data flowing into it, the process of estimating the duration of a multi-block workflow is complicated by the fact that, for non data source blocks (i.e. most blocks within the workflow) a value for the input data size must also be estimated. If the output data size for a block is assumed, like the duration estimate, to be a function of the input size (O_{i-1}) and the block settings (P_i), the output data size for a given block within a workflow can be modelled using:

$\hat{O_i} = f_{Oi}(P_i, \hat{O_{i-1}})$

Where f_{Oi} represents the predictive output size model for the i^{th} workflow block. During the process of producing a duration estimate for an entire workflow, this size estimate is propagated throughout the workflow in place of the actual data sizes. It follows, therefore that as the size of the workflow increases, the model prediction will be degraded by both the errors in predicting the duration of each block and also the errors accumulated by propagating size estimates to each duration prediction. The availability of accurate models which can predict the quantity of data produced by executing individual blocks is therefore central to accurately estimating total workflow execution time. Section 5 will investigate whether, for the workflows examined in this paper, this is indeed the case.

4.1 Model Types

The relationship between block duration and observed execution data for blocks within an e-Science Central workflow can fall into one of three broad categories:

1. The block duration can be estimated using a linear combination of the execution data contained within the performance database (figure 4). In this case a simple linear model of the form $y = mx + c$ can be used to estimate the execution duration.

2. The block duration follows a non-linear relationship between execution time and the captured performance data (figure 5). In this case one of a number of non-linear models (for example, polynomial regression or a neural network) can be used to estimate execution duration. During our experiments, none of the deployed blocks exhibited a non-linear relationship so these model types were not considered. It should, however be noted that some of the blocks studied in this paper could eventually demonstrate a non-linear relationship as larger data volumes are processed. In this situation it would be fairly straightforward to add additional model types to the performance modelling system allowing non-linear duration models to be constructed.

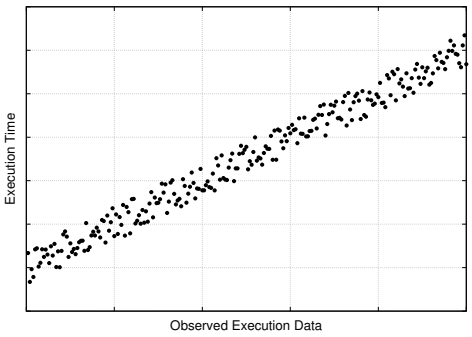

Figure 4: Linear duration with respect to block parameters

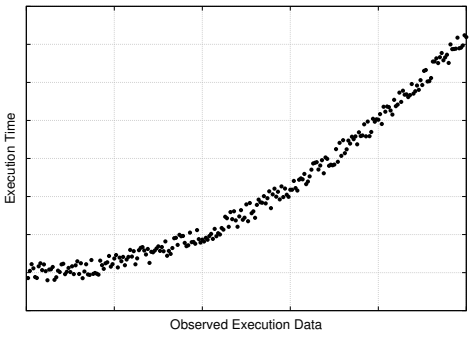

Figure 5: Non-linear duration with respect to block parameters

3. The block duration exhibits no correlation to any of the observed execution data (figure 6). In this situation, the duration prediction for the block is modelled as the average execution time for all observed executions of the block contained within the performance database.

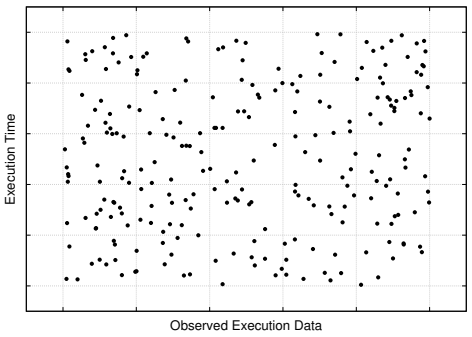

Figure 6: Duration uncorrelated with block parameters

The performance modelling system can maintain models for each version of each block observed during workflow executions and generate duration predictions using the most appropriate model on demand. This requires models to be managed (Section 4.3) and also the facility to generate some

sort of prediction even in situations where the quantity of observed data is insufficient to create one of the models described above (Section 4.2).

4.2 Model Fallbacks

One of the key requirements for the performance modelling application is to provide a robust prediction of performance properties that are refined as more data becomes available. Therefore, a number of fallback predictions are provided to cater for situations where models are unavailable for a block. This could be because a block has never been executed or that the data collected at the current time is unsuited to generating predictions. The following fallback logic has been implemented:

1. If there is no model available for a specific version of a block a version agnostic model is used. This model is constructed from all of the executions of a block and covers data collected from all versions.
2. If there are no models of any sort for a block, but there is at least one observation for a block, average values for execution duration and output size will be used.
3. If there is no data of any sort for a block, the average duration for all blocks will be used and the average output data size will be used for predicting output sizes.
4. If the system has just been initialised and no data of any type is present, no prediction will be returned.

The reasoning behind the above logic is to return a prediction wherever possible and to always return the best prediction that the system can provide at a given point in time. Predictions returned are marked with a quality flag which indicates whether the prediction has been generated using data collected for the correct version of a block or whether any fallback predictions have been used.

4.3 Model management

Because the nature of a block cannot generally be determined *a priori*, the performance modelling system must be able to determine automatically whether the execution duration of a block is linear, non-linear or uncorrelated with respect to the observed data. In order to achieve this, the system builds every type of model contained within its library for each block. In the experiments presented in this paper, this involved building linear and mean predictor models at each model update step. This pattern has been adopted for some earlier chemical modelling work [15] and has been referred to as the "panel of experts" approach. Once built, the model demonstrating the best performance on a set of test data is used to generate duration predictions until the next model update step. During the generation of the results shown in Section 5, the models were updated only once, after the initial data sets had been collected. In an actual deployment, an automatic model updating strategy would be required. This could be triggered, for example by the availability of a significant quantity of new observations, an increase in model prediction error or the age of the models within the library passing some threshold. Regardless of the strategy adopted, the process of rebuilding the models is identical: a model update message is sent to the performance server which then initiates a number of model update threads. These threads rebuild the models without requiring an interruption to the data collection process.

5. EXPERIMENTAL RESULTS

In order to demonstrate the suggested approach to modelling workflow performance, a number of experiments were performed. Initially, these focused on running simple workflows containing a set of trivial blocks under ideal conditions to investigate whether it was indeed possible to generate reliable performance models. Experiments were then performed using the same simple workflows in a more challenging Cloud environment (Amazon EC2) to establish the level of inaccuracies introduced be operating within a multi-tenanted environment. The next set of experiments focused on running a workflow containing blocks performing more complex work. These were again performed on a local server and then within Amazon EC2.

5.1 Simple workflow tests

The workflow studied in the initial experiments contained ten simple Java blocks that are provided with every installation of e-Science Central. These blocks perform basic data manipulation tasks and are therefore more IO than CPU intensive. As such, the execution of these blocks is likely to be very highly correlated to the data volumes being passed through them. The workflow used is shown in figure 7 and contains the following basic blocks:

Name	Description
Import File	Loads a file from storage into the workflow engine
Export Files	Writes files back to e-Science Central storage
Parse CSV	Interprets a data file as comma separated values
Shuffle	Randomises the order of the input data
Column Select	Picks a column from the input data
Column Join	Joins two inputs into a single set of data
Sort	Sorts the input data into ascending or descending order
CSV Export	Writes the input data as a CSV file
Subsample	Takes a subset of the rows from the input data
Transpose	Transposes the input data swapping rows for columns

All models built during these experiments were compared using the Root Mean Squared Error measurement (RMSE), and the correlation (r^2) between the predicted and observed execution durations.

5.1.1 *Experiment 1*

For the first experiment, the workflow shown in figure 7 was executed 250 times with a set of randomly generated input files ranging in size from 7KB to 14MB. These files were pre-generated and one was selected at random for each of the 250 executions. The same sets of data were transferred to Amazon EC2 for the second experiment. The results of this experiment demonstrated that:

1. It was possible to generate an accurate prediction of the volumes of data produced by each of the blocks studied. For example, the volume of data produced by the Import File block is predicted with almost 100% accuracy using a linear model with the size of the imported file captured as a block property as it's single

input variable (See figure 8).

2. For the majority of blocks, it was possible to generate an accurate prediction of execution time based upon the size of the input data and the captured block configuration parameters. For example, figure 9 shows the duration prediction model for the Shuffle block (RMSE=0.344,r^2=0.999).

3. For some blocks, the duration model, at the data sizes tested, did not generate a good prediction. For example the model for the Import File block demonstrated a poor fit to the observed data (see figure 10). However, an examination of the spread of execution times (figure 13) shows a fairly small spread when compared with other data intensive blocks. It is likely that, at the scales tested (a maximum data size of 14MB), the imported file sizes do not take a significant time to copy to the workflow engine meaning that there is an insufficiently rich set of data to build any meaningful predictive models.

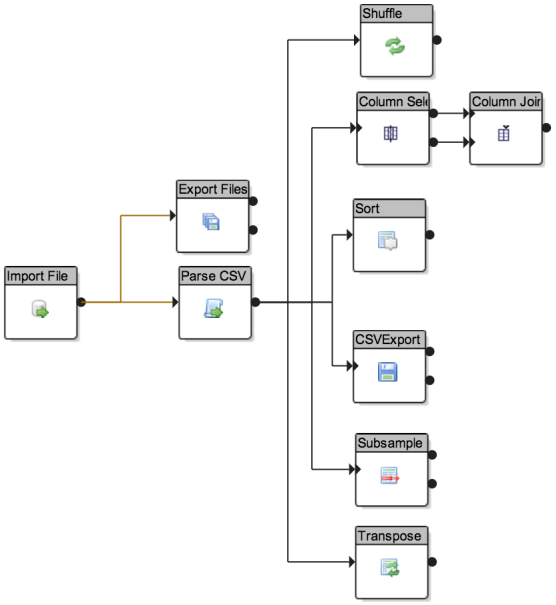

Figure 7: Simple data generation workflow

This set of 250 workflow executions yielded 2500 observations within the performance database which were then used to build the suite of predictive models. In order to assess the performance of these models when applied to different workflows (containing a subset of the blocks shown in figure 7), a different workflow was constructed which again processed a set of randomly generated data (figure 11).

This workflow was executed multiple times and for each execution, the actual duration was recorded along with a duration estimate generated by the performance monitoring system (RMSE=4.077,r^2=0.997). The results of this exercise are shown in figure 12. Figure 12 shows a number of runs executing significantly faster than predicted. An examination of the results shows that these are the final executions of the experiment. Because the workflow engine executes multiple workflows concurrently, towards the end

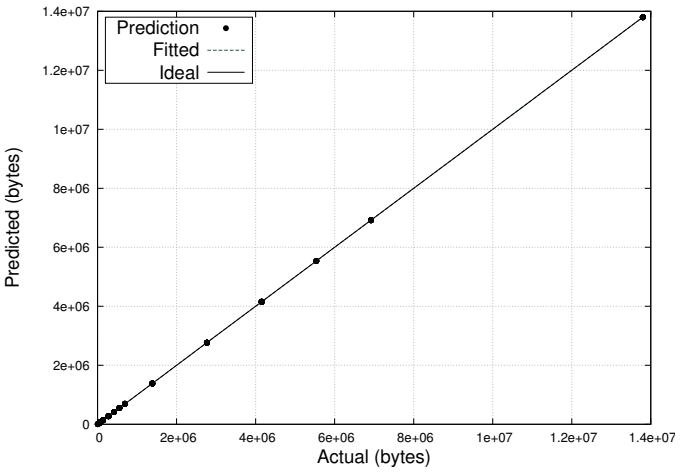

Figure 8: Prediction of output size from Import File block on local server

of any run, in the current setup, there is a strong chance that one workflow will be left executing alone. This workflow will not be competing for resources (filesystem, CPU etc) with other workflows and, as such, executes faster than predicted by the model.

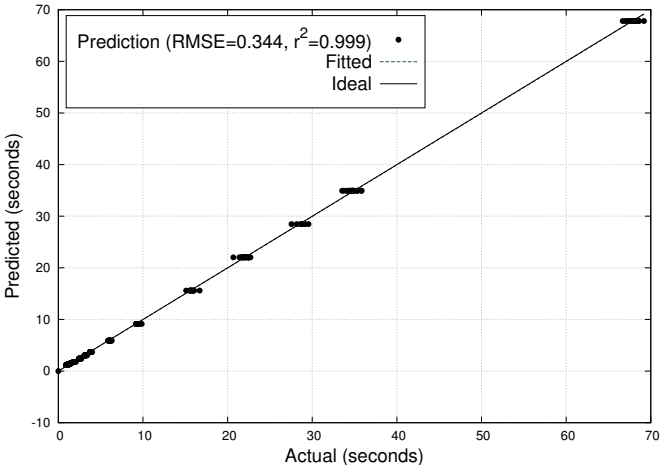

Figure 9: Prediction of execution duration of the Shuffle block on local server

5.1.2 Experiment 2

The second experiment carried out was a direct repeat of the first, but performed on a public cloud (Amazon EC2). The installation was configured in a manner typical of many e-Science Central installations and comprised a single server machine containing the JBoss application server and Postgres database with two additional machines, each containing a workflow engine. All machines used were m1.xlarge instances configured with 4x 2GHz Intel Xeon CPUs and 15GB RAM. Once again, 250 executions of the simple calibration workflow shown in figure 7 were started and the results captured using the performance monitoring system. The results of this experiment also indicated a good model performance (see, for example figure 14, which illustrates

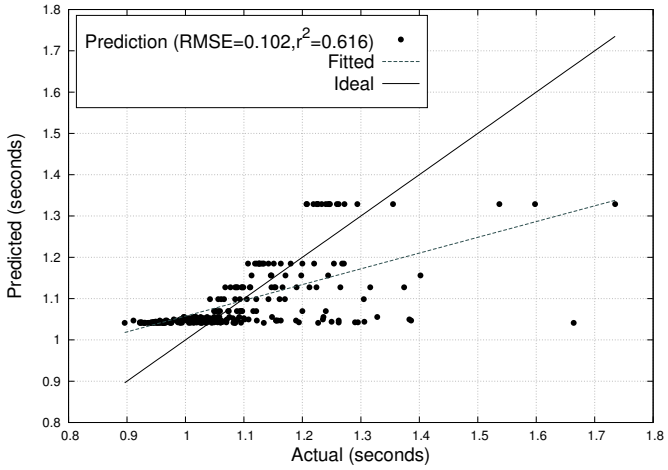

Figure 10: Prediction of execution duration of the Import File block on local server

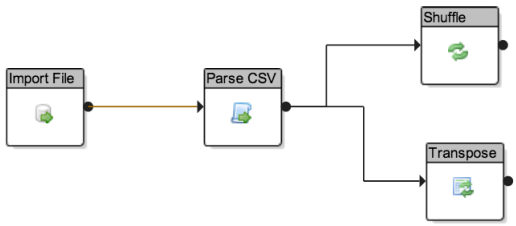

Figure 11: Testing workflow

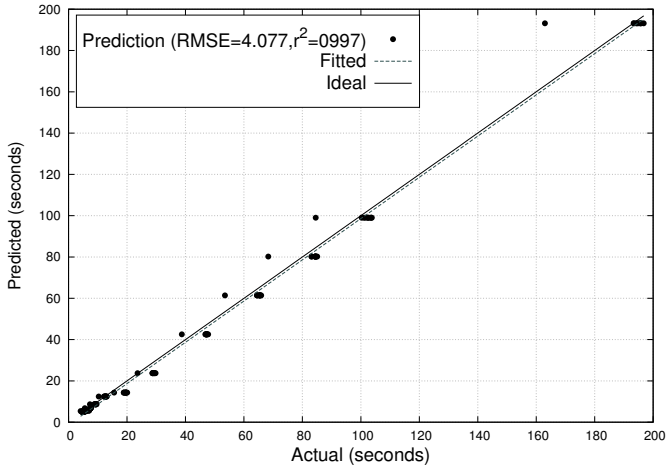

Figure 12: Execution duration prediction for new workflow

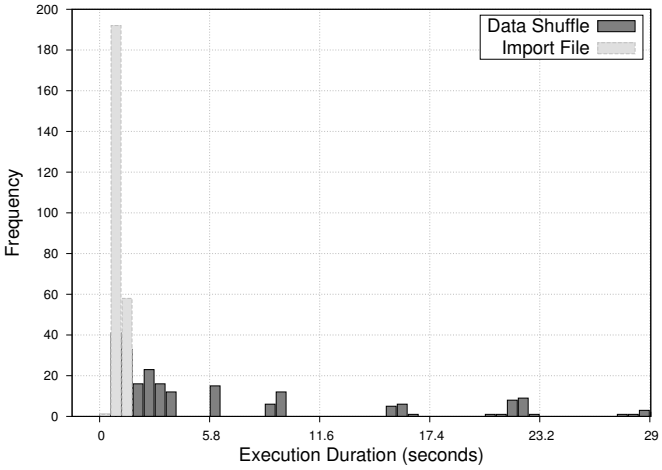

Figure 13: Comparison of Import File and Shuffle execution times on local server

the model performance on the Shuffle block), albeit with a slightly larger spread in predicted execution times when processing larger data files.

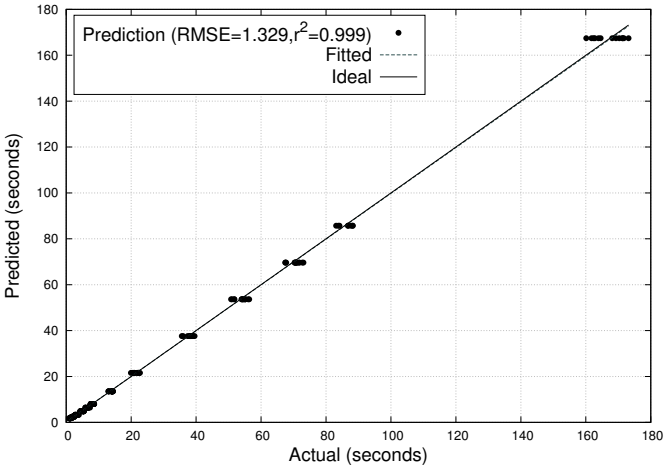

Figure 14: Prediction of execution duration of the Shuffle block on Amazon EC2

5.1.3 Experiment 3

The aim of the third experiment performed was to investigate the predictability of more complex workflow blocks than the simple data manipulation blocks examined in the first two experiments. For this experiment we used the Chemical Development Kit[6] library to calculate a number of molecular descriptors for a set of chemical structures. Descriptors are parameters that are derived from chemical structure data and can be used in the development of predictive models of complex chemical properties such as toxicity [15]. This problem was selected because the time taken to calculate these descriptors is highly dependent on the structures contained in the input files. The code is also far more CPU than IO intensive and, as such, allowed us to test our

[6] `http://sourceforge.net/projects/cdk/`

model building code in a different scenario. The calculations were performed using the workflow shown in figure 16 which imported a library of 330 structures and calculated the descriptors for a random sized randomly selected subset of these individual structures. The performance models were constructed using 50% of the structure library with the remaining 50% being reserved for model testing as a set of unseen validation structures. As with the first two experiments, 250 executions of this workflow were initiated in order to build a reasonably rich set of performance data. The results of this exercise are illustrated in figure 15, which shows a comparison of the predicted execution time for the workflow shown in figure 16. This prediction was first generated over the training data and then a new set of unseen structures were passed through the same workflow and the predicted execution time recorded. The results demonstrate an extremely good fit on the training (RMSE=5.008, r^2=0.980). It is notable that the prediction quality on the testing data is almost identical to that on the training data (RMSE=4.698,r^2=0.981). This is atypical of many modelling exercises and the likely cause is the similarity of the two input data sets – even though the testing data contained structures unseen during the model building phase, both sets contained a large sampling of a single data set (ChemBL[7] biological activity database) which contains many structures related to a fairly narrow field of biological activity measurements. It is probable that a data set containing different categories of structures would not exhibit the same accuracy during this model validation phase.

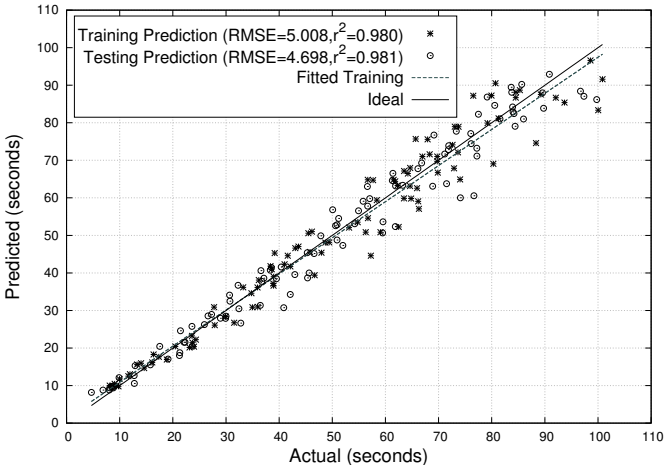

Figure 15: Workflow prediction comparison for descriptor calculator workflow on local server

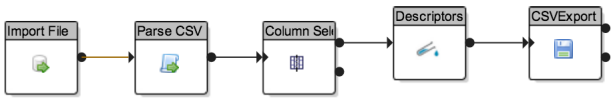

Figure 16: Chemical descriptor calculation workflow

[7]https://www.ebi.ac.uk/chembl/

5.1.4 *Experiment 4*

The fourth experiment performed was a direct repeat of the previous experiment, but again performed on a public Amazon EC2 cloud. The configuration was identical to that adopted in 5.1.3 and the results are shown in figure 17. Interestingly, the duration prediction for the descriptor calculation is more accurate (RMSE=2.591,r^2=0.995 on the training data set and RMSE=2.799,r^2=0.995 on the testing data set) on the public EC2 setup than on the local server (figure 18). Whilst reports of large execution time variations have been reported for public clouds, this effect did not manifest itself in these experiments. There are two likely explanations for the improved model performance: the experiments were performed over a fairly short period of time (approximately 15 minutes) over a small number of machines so this reported variability simply did not appear during this interval; also as the installation was split over several servers on EC2 with the application server and database residing on one instance with two workflow engines on separate instances the qsar calculations and database operations were not performed using a single set of resources whereas, in the local experiments everything was performed on a single server with all the associated problems of resource contention.

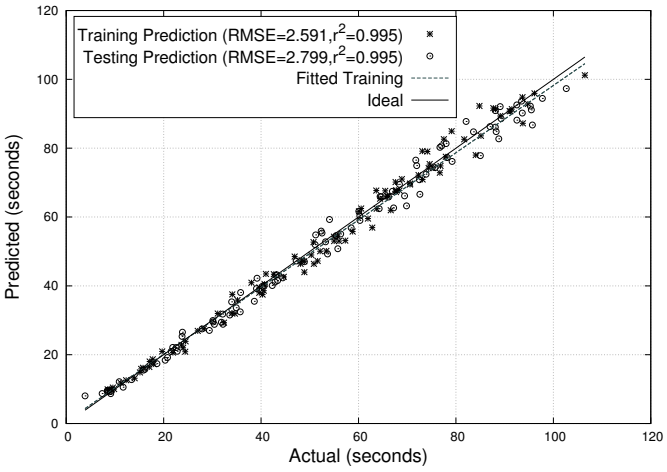

Figure 17: Workflow prediction comparison for descriptor calculator workflow executed on Amazon EC2

6. FUTURE WORK

This paper has presented an architecture for capturing and modelling workflow performance data within the e-Science Central cloud platform and performed some initial tests on a number of simple workflow blocks as well as a more complex chemical property modelling block. Whilst we have managed to produce some useful models, much more work is needed in order to fully quantify the prediction quality of the models generated. In particular, the robustness of models and the most applicable model regeneration strategies in the face of new data need to be investigated. This requirement also highlights the more general need for improved model performance metrics, particularly as predictions are propagated through larger and larger workflows.

As the system described in this paper is a repository for all of the performance data generated during workflow oper-

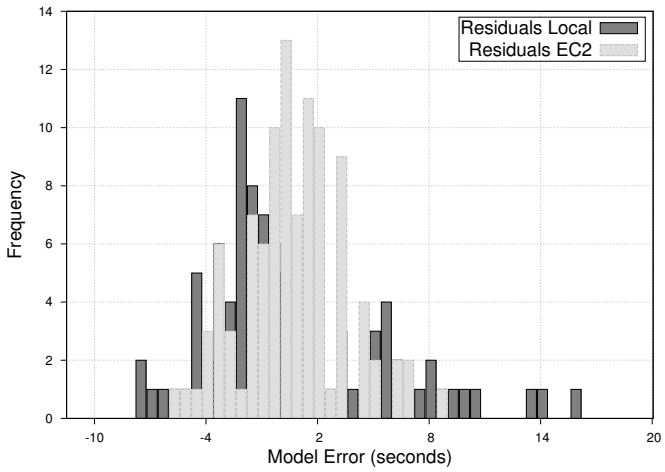

Figure 18: Workflow prediction comparison for EC2 and Local Server

ations, the potential exists to create a fully featured monitoring platform that can be used to analyse the performance of workflow runs in realtime in order to identify poorly performing cloud instances or to detect performance issues with specific workflow components. Such a system would be beneficial not only to large scale e-Science Central deployments but also to any software that has the capability to generate a live stream of performance log events.

7. CONCLUSIONS

This paper has demonstrated that, for the limited range of workflows and blocks studied, we can build reliable duration predictions that operate on both local servers and public Cloud infrastructures. Although these models performed well, we have identified the need for a greater variety of experiments to be performed to evaluate a) the long term reliability of these models as more data is collected and b) the rate of increase in prediction error as more complex workflows are modelled.

The success of any workflow duration predictions generated using this approach is is dependent upon being able to generate good relationships between the amount of data consumed by a block and the amount of data produced. For code that does not have an input-output size relationship that is amenable to modelling, the duration predictions for larger workflows will degrade rapidly as model errors are propagated through the various modelling algorithms. For code that does model well, we can propagate size estimates through a number of consecutive blocks (five for the chemical structure analysis workflow) and still generate a useful prediction of expected workflow execution duration.

Although we have only integrated the performance capture and modelling system with the e-Science Central platform, both the concepts and code should be applicable to any system that can generate performance logging messages in the correct format. All of the code used in the development of both e-Science Central and the performance modelling platform are available on the e-Science Central Sourceforge[8] project site.

[8]`http://sourceforge.net/projects/esciencecentral/`

8. ACKNOWLEDGEMENTS

This work has been funded as part of the SiDE project, the RCUK Digital Economy Research Hub on Social Inclusion through the Digital Economy, EP/G066019/1 and by the Digital Institute at Newcastle University.

9. REFERENCES

[1] J. Cala, H. Hiden, S. Woodman, and P. Watson. Cloud computing for fast prediction of chemical activity. *Future Generation Computer Systems*, 29(7):1860 – 1869, 2013.

[2] R. Cushing, S. Koulouzis, A. S. Z. Belloum, and M. Bubak. Prediction-based auto-scaling of scientific workflows. In *Proceedings of the 9th International Workshop on Middleware for Grids, Clouds and e-Science*, MGC '11, pages 1:1–1:6, New York, NY, USA, 2011. ACM.

[3] M. Dobber, R. van der Mei, and G. Koole. Effective prediction of job processing times in a large-scale grid environment. In *High Performance Distributed Computing, 2006 15th IEEE International Symposium on*, pages 359–360, 2006.

[4] R. Duan, F. Nadeem, J. Wang, Y. Zhang, R. Prodan, and T. Fahringer. A hybrid intelligent method for performance modeling and prediction of workflow activities in grids. In *Cluster Computing and the Grid, 2009. CCGRID '09. 9th IEEE/ACM International Symposium on*, pages 339–347, 2009.

[5] H. Hiden, S. Woodman, P. Watson, and J. Cala. Developing cloud applications using the e-science central platform. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 371(1983), 2013.

[6] X. Liu, Z. Ni, D. Yuan, Y. Jiang, Z. Wu, J. Chen, and Y. Yang. A novel statistical time-series pattern based interval forecasting strategy for activity durations in workflow systems. *Journal of Systems and Software*, 84(3):354 – 376, 2011.

[7] T. Miu and P. Missier. Predicting the Execution Time of Workflow Activities Based on Their Input Features. In I. Taylor and J. Montagnat, editors, *Procs. WORKS 2012*, Salt Lake City, US, 2012. ACM.

[8] F. Nadeem and T. Fahringer. Predicting the execution time of grid workflow applications through local learning. In *Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis*, SC '09, pages 33:1–33:12, New York, NY, USA, 2009. ACM.

[9] F. Nadeem and T. Fahringer. Using templates to predict execution time of scientific workflow applications in the grid. In *Cluster Computing and the Grid, 2009. CCGRID '09. 9th IEEE/ACM International Symposium on*, pages 316–323, 2009.

[10] N. Roy, A. Dubey, and A. Gokhale. Efficient autoscaling in the cloud using predictive models for workload forecasting. In *Cloud Computing (CLOUD), 2011 IEEE International Conference on*, pages 500–507, 2011.

[11] S. Sadjadi, S. Shimizu, J. Figueroa, R. Rangaswami, J. Delgado, H. Duran, and X. Collazo-Mojica. A modeling approach for estimating execution time of long-running scientific applications. In *Parallel and*

Distributed Processing, 2008. IPDPS 2008. IEEE International Symposium on, pages 1–8, 2008.

[12] V. Taylor, X. Wu, J. Geisler, and R. Stevens. Using kernel couplings to predict parallel application performance. In *High Performance Distributed Computing, 2002. HPDC-11 2002. Proceedings. 11th IEEE International Symposium on*, pages 125–134, 2002.

[13] V. Taylor, X. Wu, and R. Stevens. Prophesy: an infrastructure for performance analysis and modeling of parallel and grid applications. *SIGMETRICS Perform. Eval. Rev.*, 30(4):13–18, Mar. 2003.

[14] V. T. van Hees, L. Gorzelniak, E. C. Dean Leon, M. Eder, M. Pias, S. Taherian, U. Ekelund, F. Renstrom, P. W. Franks, A. Horsch, and S. Brage. Separating movement and gravity components in an acceleration signal and implications for the assessment of human daily physical activity. *PLoS ONE*, 8(4):e61691, 04 2013.

[15] P. Watson, H. G. Hiden, S. J. Woodman, D. E. Leahy, J. Cala, and P. Missier. The panel of experts cloud pattern. In *Proceedings of the third international workshop on Cloud data management*, CloudDB '11, pages 23–24, New York, NY, USA, 2011. ACM.

[16] S. Woodman, H. Hiden, P. Watson, and P. Missier. Achieving reproducibility by combining provenance with service and workflow versioning. In *Proceedings of the 6th workshop on Workflows in support of large-scale science*, WORKS '11, pages 127–136, New York, NY, USA, 2011. ACM.

[17] Q. Wu and V. V. Datla. On performance modeling and prediction in support of scientific workflow optimization. In *Proceedings of the 2011 IEEE World Congress on Services*, SERVICES '11, pages 161–168, Washington, DC, USA, 2011. IEEE Computer Society.

[18] S. Zhang, A. Rowlands, P. Murray, and T. Hurst. Physical activity classification using the genea wrist-worn accelerometer. *Medicine and Science in Sports and Exercise*, 44(4):742–748, April 2012.