Newcastle University

ePrints

Hiden HG, Woodman SJ, Watson P.

Prediction of workflow execution time using provenance traces: practical applications in medical data processing.

***In: 2016 IEEE 12th International Conference on e-Science*.**

23-26 October 2016, Baltimore, MA: IEEE.

Copyright:

© 2016 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

Date deposited:

14/09/2016

Newcastle University ePrints - eprint.ncl.ac.uk

Prediction of workflow execution time using provenance traces: practical applications in medical data processing

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

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

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

***Abstract*—The use of cloud resources for processing and analysing medical data has the potential to revolutionise the treatment of a number of chronic conditions. For example, it has been shown that it is possible to manage conditions such as diabetes, obesity and cardiovascular disease by increasing the right forms of physical activity for the patient. Typically, movement data is collected for a patient over a period of several weeks using a wrist worn accelerometer. This data, however, is large and its analysis can require significant computational resources. Cloud computing offers a convenient solution as it can be paid for as needed and is capable of scaling to store and process large numbers of data sets simultaneously.**

However, because the charging model for the cloud represents, to some extent, an unknown cost and therefore risk to project managers, it is important to have an estimate of the likely data processing and storage costs that will be required to analyse a set of data. This could take the form of data collected from a patient in clinic or of entire cohorts of data collected from large studies. If, however, an accurate model was available that could predict the compute and storage requirements associated with a piece of analysis code, decisions could be made as to the scale of resources required in order to obtain results within a known timescale.

This paper makes use of provenance and performance data collected as part of routine e-Science Central workflow executions to examine the feasibility of automatically generating predictive models for workflow execution times based solely on observed characteristics such as data volumes processed, algorithm settings and execution durations. The utility of this approach will be demonstrated via a set of benchmarking examples before being used to model workflow executions performed as part of two large medical movement analysis studies.

I. Introduction

Historical and ongoing research projects are investigating the links between levels of physical activity and chronic conditions such as Type II Diabetes and Myotonic Dystrophy Type I[1][2]. The most common method for monitoring the physical activity of a subject is via the use of wearable devices such as accelerometers. Typically these take the form of a wristwatch that captures data at 100Hz over a period of several weeks. Devices such as the GENEActiv[3] and Actiwatch[4] are commonly deployed and a wide body of research has been published using data captured in this way [1].

Movement data takes the basic form of a long timeseries containing three distinct channels of data captured from three perpendicular axes (X, Y & Z). An analysis of the patterns within this data can give an insight into a number of underlying conditions: Activity level [2], sleep quality [3] and, gait [4] which can be used to indicate quality of life, aid in assessments and measure rehabilitation of some medical conditions.

In conjunction with this movement data capture, in many studies, patients attend clinics periodically in order to discuss progress with a clinician. They will often have other measurements, such as weight, blood glucose etc. taken and their physical activity inspected.

The issue, however, is that as the wrist worn accelerometers measure movement data over three axes at approximately 100Hz for many days, the quantity of data to be analysed is large - a typical data file is approximately 800MB in size and comprises some 100 million rows of data. Once collected, the analysis procedure [5] for this data involves categorizing the acceleration signals into one of several categories for instance: Sedentary, Light Activity, Walking and Running. This is then used to make recommendations as to suitable exercise plans for that specific patient. Further analysis may also be conducted [6] which can answer different research questions such as whether the patients sleep is affected by a change in medication or investigate exercise patterns across a large population.

Although the task of processing data for a single patient is tractable, clinical studies have collected movement data from large numbers of participants and analysing this data requires larger scale facilities. The sizes of three studies are shown in Table I

Clearly, given an hour of typical processing time for a simple analysis, the task of processing data from a complete

[1]http://www.directclinicaltrial.org.uk/

[2]http://optimistic-dm.eu/

[3]http://www.geneactiv.org

[4]http://www.philips.co.uk/healthcare/product/HC1046964/actiwatch-spectrum-activity-monitor

TABLE I
MOVEMENT STUDY SIZES

Study	Participants	Data Size
Newcastle 85+	1000	300GB
Whitehall	4300	4TB
UK Biobank	100k	24TB

study is not a trivial one. For example, using the information shown in Table I, an analysis of the Whitehall study using an algorithm such as PAC1 [5] would take approximately six months of continuous computation on a single PC ignoring any data preparation and transformation required. This raises issues regarding the both provision of appropriate resources to complete an analysis in an acceptable time frame and also keeping track of the analysis process and software versions used.

This tracking requirement is particularly important as movement analysis algorithms are the subject of active research with improvements and fixes published frequently. For example, one popular algorithm [6] has seen thirteen releases in a two year period in order to fix bugs and increase functionality.

The use of workflows to coordinate the analysis of data generated in these projects is increasingly common. The developers of algorithms such as PAC1 and GGIR are experts in classifying movement based on accelerometery. However, in the context of a long running study there are usually study management issues such as participant tracking and reporting, and other associated data prepraration tasks. Workflow engines mitigate some of these issues by providing suites of services used in traditional Extract, Translate, Load pipelines [7] which are convenient in this context too. The availability of these services allows the algorithm developers to focus on the medical aspects of their work and use in-built tools for the other tasks. The framework that the workflow will execute within usually also deals with concurrent executions and dependency resolution, again relieving the algorithm developer from having to consider these issues.

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 [8]. It provides data storage and sharing facilities along with a workflow engine designed to operate at large scale within a hosted cloud environment.

Within clinical and pre-clinical trials, usually a data analyst within the project team will determine the algorithmic needs of the study and develop the workflows which will generate reports. During the study the staff collecting the data, often in a clinic, will upload that data into e-Science Central where the workflow developed by the analyst will process it. 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 running a trial. 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 is particularly pertinent given the uneven processing requirements in these studies - devices are usually recovered sporadically when clinics run and lead to higher processing requirements at these times. At other times there will be little or no requirement for processing resources.

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.

This paper is structured as follows: The following Section provides context and compares the work in this paper to previously published research. Section III gives an overview of the architecture of e-Science Central and how the performance data is captured. The process for generating and managing predictive models is described in Section IV and these are evaluated against medical workloads in Section V. Finally, conclusions are drawn in Section VII.

II. 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 [9]–[11] whilst others have attempted to decompose these into components as we do [12], [13]. The work presented in by Duan [12] 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 [14] 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 [15] 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 [16] 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 [17] 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. 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 [5], [6], [18].

The work around the Prophesy project to develop a general purpose performance analysis system is most similar to our approach [19]. 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 [20]. 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 [21].

III. 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. 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 or create online. A range of data analysis and programming languages are supported, including Java, Javascript, R and Octave.

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 [22]. The benefit of supporting pure data flow, as we will see in Section IV, 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.

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 automatically versioned when they are updated by the user. 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 IV-B). 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 server. Each workflow engine 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.

Each time a block in a workflow 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 [23]). 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-Science Central, 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.

The data within the Performance server is consumed asynchronously from a JMS queue which prevents it from affecting the performance of the system that it is monitoring and allows multiple sources of performance metrics to send data concurrently. When received, the data is persisted into a Postgres database. In order to generate the models we make use of the Apache Commons Maths 3 library 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.

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,

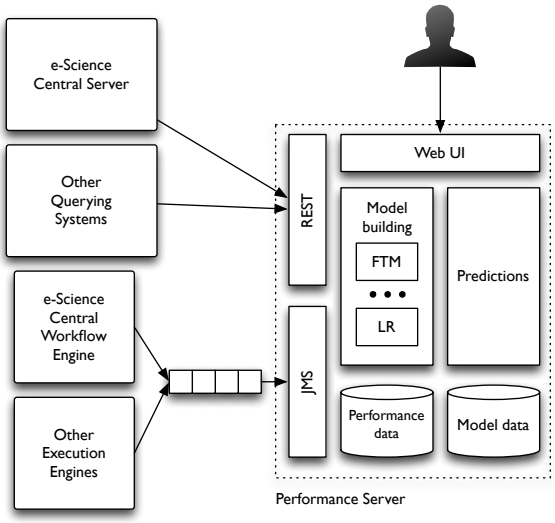

Fig. 1. Architecture of the Performance Server

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.

IV. 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.

The basic structure of an e-Science Central workflow is illustrated in figure 2 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

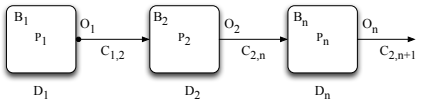

Fig. 2. e-Science Central workflow structure

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:

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

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:

$$\hat{D_{wf}} = \sum_{i=1}^{n} \hat{D_i} \tag{2}$$

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 III). The estimated duration of any block within the workflow can therefore be calculated using:

$$\hat{D_i} = f_{Di}(P_i, O_{i-1}) \tag{3}$$

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:

$$\hat{D_{wf}} = \sum_{i=1}^{n} (f_{Di}(P_i, O_{i-1})) \tag{4}$$

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:

$$\hat{O_i} = f_{Oi}(P_i, \hat{O_{i-1}}) \quad (5)$$

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 V will investigate whether, for the workflows examined in this paper, this is indeed the case.

A. 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. 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. 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.
3) The block duration exhibits no correlation to any of the observed execution data. 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.

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 IV-C) 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 IV-B).

B. 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.

C. 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 [14] 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 V, 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.

V. 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 performed in Amazon EC2.

A. 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 3 and contains the following basic blocks:

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.

1) Experiment 1: For the first experiment, the workflow shown in figure 3 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.
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 4 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 5). However, an examination of the spread of execution times 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.

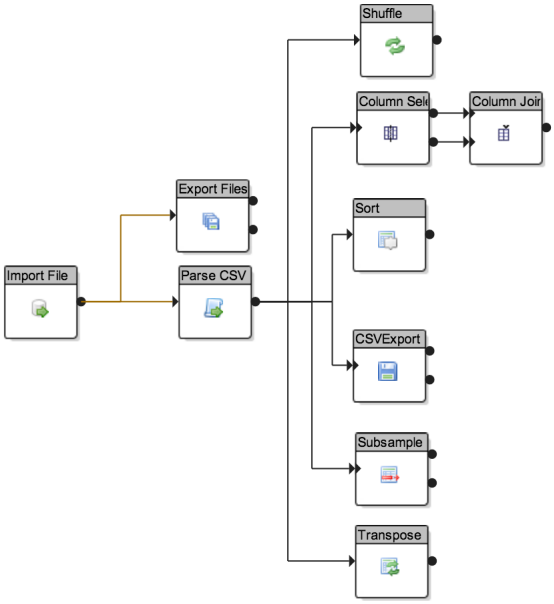

Fig. 3. Simple data generation workflow

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 3), a different workflow was constructed which again processed a set of randomly generated data (figure 6).

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 7. Figure 7 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 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.

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

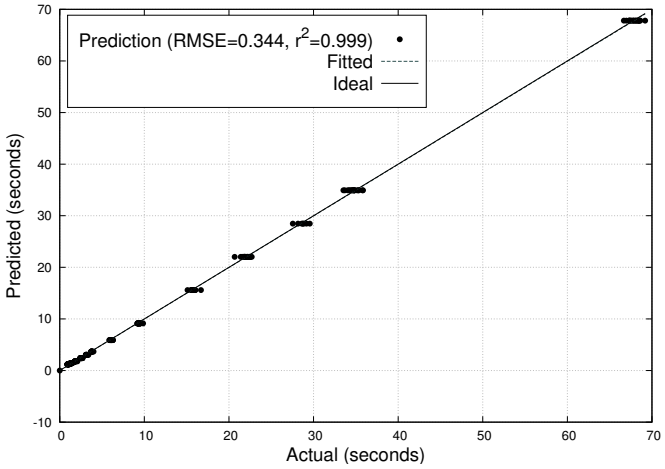

Fig. 4. Prediction of execution duration of the Shuffle block on local server

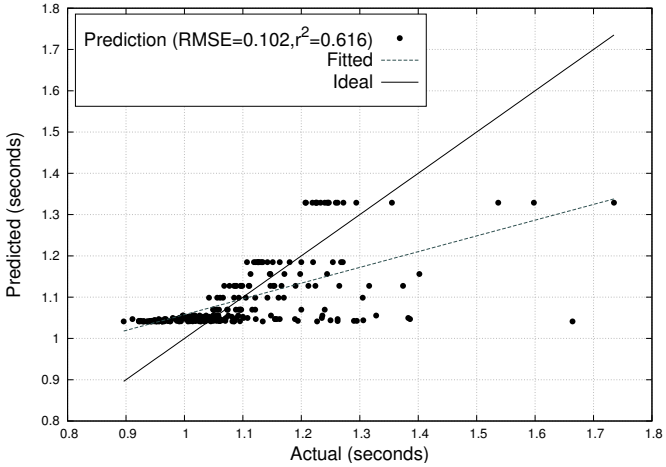

Fig. 5. Prediction of execution duration of the Import File block on local server

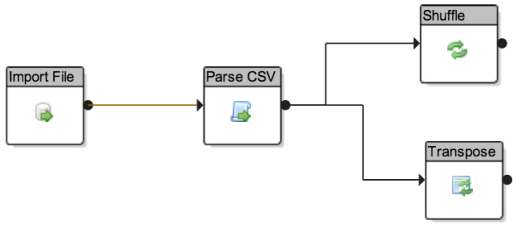

Fig. 6. Testing workflow

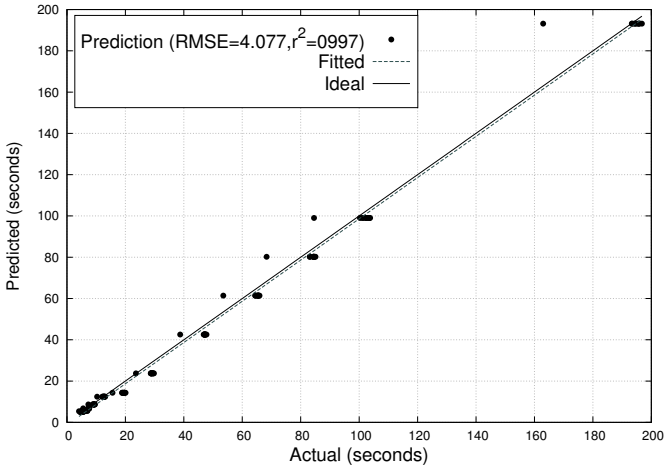

Fig. 7. Execution duration prediction for new workflow

CPUs and 15GB RAM. Once again, 250 executions of the simple calibration workflow shown in figure 3 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 8, which illustrates the model performance on the Shuffle block), albeit with a slightly larger spread in predicted execution times when processing larger data files.

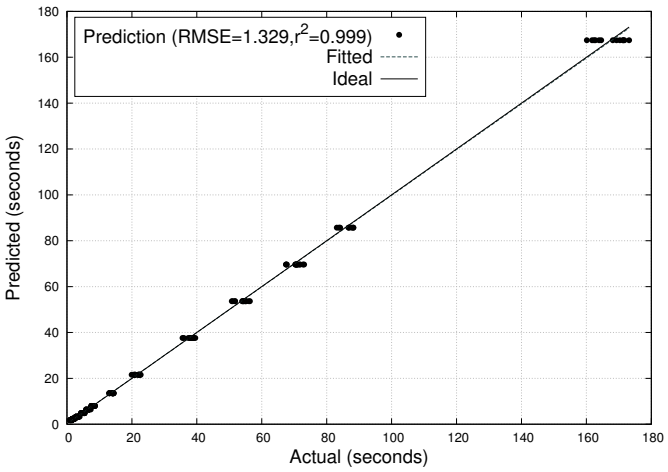

Fig. 8. Prediction of execution duration of the Shuffle block on Amazon EC2

VI. Movement Predictions

Having determined the feasibility of generating models via the use of the ambient provenance and performance data capture architecture show in Figure 1, the next experiment produced predictive models of a set of movement analytics routines operating within the e-Science Central platform. Although the plaform has been deployed in multiple studies, the models built in this section focus on a standard library provided in R (GGIR) [6] that process raw accelerometry data collected from wearable devices. This library was selected

because it has seen the widest adoption of the various algorithms contained in the system and has the largest collection of performance data available. The algorithm itself comprises two operational phases:

- **Part 1**: Is a computationally expensive process that is performed once for each data file. Its purpose is to generate a set of metadata that can be used to facilitate a range of different analyses and produce various summary reports.
- **Part 2-n**: Once the summary metadata has been generated a set of second phase analyses can be carried out to generate reports.

The workflows modelled therefore follow a standard pattern: Data import and parsing, GGIR part 1 metadata generation then GGIR part 2 report generation. Data was collected from each of the two GGIR phases over a set of studies with the aim of generating duration and output size models.

Models for execution duration were genarated using data gathered from the Optimistic (Observational Prolonged Trial In myotonic dystrophy type 1 to Improve Quality of Life Standards, a Target Identification Collaboration) study, which was operated by Newcastle University with data collection distributed between Newcastle, Munich, Nijmegen and Paris. This study focussed around approximately 300 people with a genetic diagnosis of myotonic dystrophy type 1, 18 years old and able to walk independently [24]. As part of this study, participants were asked to visit an assesement centre at least five times over a seventeen month period and were required to use a wrist worn accelerometer.

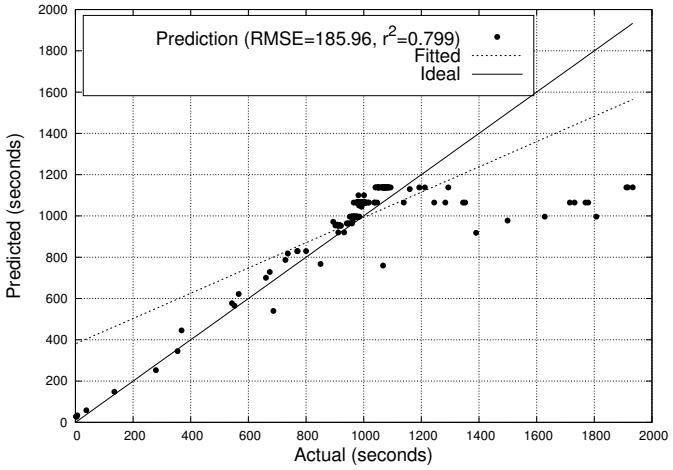

Fig. 9. GGIR Part 1 Duration Model

Data gathered from the three centres was uploaded to a central e-Science Central installation where workflows automatically performed a GGIR Part 1 processing step, followed by the generation of a customised report (a GGIR Part 2 operation). Interesting, the GGIR Part 1 model demonstrated a roughly linear performance up until the execution time reached about 1200 seconds. Subsequently, the code took substatially longer than predicted to execute. After an investigation, it became apparent that the data files were uploaded by to the e-SC server by the various study centres in batches. This had the effect of a number of files competing for resources on the system, with the effect that some executions took longer than predicted. The output size model (i.e. the size of the generated report) demonstrated a broadly linear response with the actual metadata size being in the range 7-10MB for each participant collection exercise. It is also apparent from the graphs that the majority of durations and output size predictions are clustered in a fairly tight area. The most likely cause of this observation is the fact that the study protocol produced very similar data file sizes as the participants all wore the devices for broadly similar periods of time.

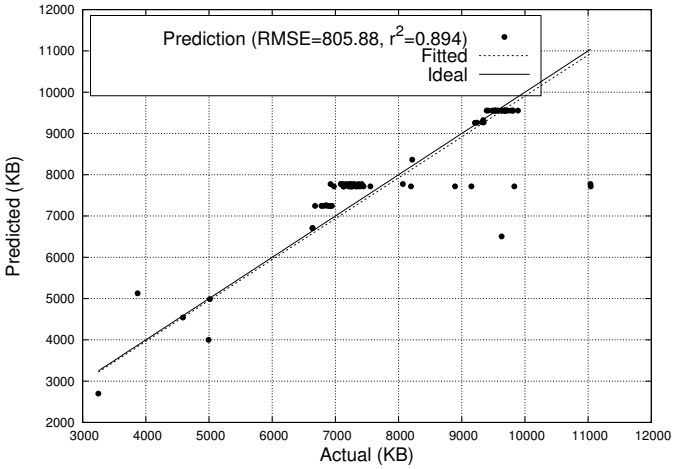

Fig. 10. GGIR Part 1 Generated Data Size Model

An updated version of the GGIR algorithm, which combines both Part 1 and Part 2 demonstrates this clustering behaviour more clearly, with an upper predicted execution time of approximately 1400 seconds. (Figure 11).

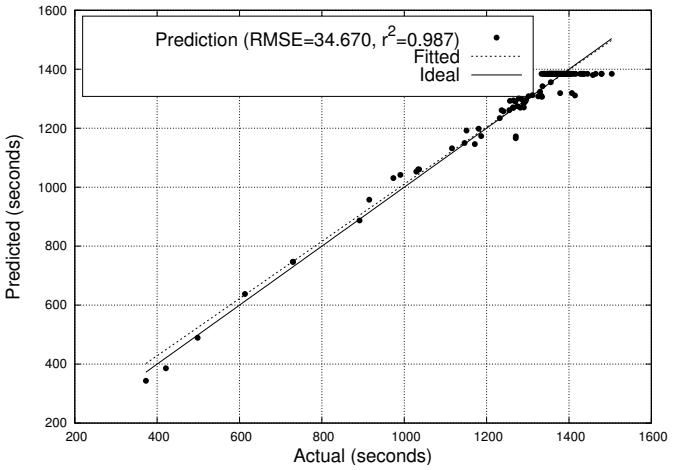

Fig. 11. GGIR Version 2 Duration Model

VII. Conclusions

This paper has demonstrated that, for the set of workflow components studied it is feasible to build reliable predictions of the execution time and volume of data produced from past executions and configuration parameters. These models have been shown to be robust in both controlled, local, environments and using shared virtualised environments provisioned in a commercial cloud provider. The components modelled include internal e-Science Central blocks and also open source algorithms used in the analysis of data in clinical trials. Using these models it is possible to predict the performance of large workflows, a capability which has been implemented in the e-Science Central Performance Server. Further work will include assessing the reliability of these predictions during subsequent studies which are scheduled to begin mid 2016.

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 a similar format.

Acknowledgements

This work has been funded by the Digital Institute at Newcastle University, UK.

References

[1] S. Sabia, P. Cogranne, V. T. van Hees, J. A. Bell, A. Elbaz, M. Kivimaki, and A. Singh-Manoux, "Physical activity and adiposity markers at older ages: Accelerometer vs questionnaire data," *Journal of the American Medical Directors Association*, vol. 16, no. 5, pp. 438.e7 – 438.e13, 2015.

[2] S. E. Crouter, J. R. Churilla, and D. R. Bassett, "Estimating energy expenditure using accelerometers," *European Journal of Applied Physiology*, vol. 98, no. 6, pp. 601–612, 2006. [Online]. Available: http://dx.doi.org/10.1007/s00421-006-0307-5

[3] V. T. van Hees, S. Sabia, K. N. Anderson, S. J. Denton, J. Oliver, M. Catt, J. G. Abell, M. Kivimki, M. I. Trenell, and A. Singh-Manoux, "A novel, open access method to assess sleep duration using a wrist-worn accelerometer," *PLoS ONE*, vol. 10, 11 2015.

[4] A. Godfrey, R. Conway, D. Meagher, and G. ÓLaighin, "Direct measurement of human movement by accelerometry," *Medical Engineering and Physics*, vol. 30, no. 10, pp. 1364 – 1386, 2008, special issue to commemorate the 30th anniversary of Medical Engineering and Physics30th Anniversary Issue.

[5] 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*, vol. 44, no. 4, pp. 742–748, April 2012.

[6] 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*, vol. 8, no. 4, p. e61691, 04 2013. [Online]. Available: http://dx.doi.org/10.1371%2Fjournal.pone.0061691

[7] U. Dayal, M. Castellanos, A. Simitsis, and K. Wilkinson, "Data integration flows for business intelligence," in *Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology*, ser. EDBT '09, 2009, pp. 1–11.

[8] 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*, vol. 371, no. 1983, 2013. [Online]. Available: http://rsta.royalsocietypublishing.org/content/371/1983/20120085.abstract

[9] 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*, 2006, pp. 359–360.

[10] 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*, ser. SC '09. New York, NY, USA: ACM, 2009, pp. 33:1–33:12. [Online]. Available: http://doi.acm.org/10.1145/1654059.1654093

[11] F. Nadeem and T. Fahringer, "Using templates to predict execution time of scientific workflow applications in the grid," in *Proceedings of the 2009 9th IEEE ACM Internation Symposium on Cluster Computing and the Grid*. IEEE Computer Society, 2009.

[12] 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*, 2009, pp. 339–347.

[13] 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*, vol. 84, no. 3, pp. 354 – 376, 2011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0164121210003262

[14] 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*, ser. CloudDB '11. New York, NY, USA: ACM, 2011, pp. 23–24. [Online]. Available: http://doi.acm.org/10.1145/2064085.2064091

[15] 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*, ser. MGC '11. New York, NY, USA: ACM, 2011, pp. 1:1–1:6. [Online]. Available: http://doi.acm.org/10.1145/2089002.2089003

[16] 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*, 2011, pp. 500–507.

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

[18] 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*, ser. SERVICES '11. Washington, DC, USA: IEEE Computer Society, 2011, pp. 161–168. [Online]. Available: http://dx.doi.org/10.1109/SERVICES.2011.37

[19] V. Taylor, X. Wu, and R. Stevens, "Prophesy: an infrastructure for performance analysis and modeling of parallel and grid applications," *SIGMETRICS Perform. Eval. Rev.*, vol. 30, no. 4, pp. 13–18, Mar. 2003. [Online]. Available: http://doi.acm.org/10.1145/773056.773060

[20] 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*, 2002, pp. 125–134.

[21] 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*, 2008, pp. 1–8.

[22] J. Cala, H. Hiden, S. Woodman, and P. Watson, "Cloud computing for fast prediction of chemical activity," *Future Generation Computer Systems*, vol. 29, no. 7, pp. 1860 – 1869, 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167739X13000253

[23] 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*, ser. WORKS '11. New York, NY, USA: ACM, 2011, pp. 127–136. [Online]. Available: http://doi.acm.org/10.1145/2110497.2110512

[24] B. van Engelen, "Cognitive behaviour therapy plus aerobic exercise training to increase activity in patients with myotonic dystrophy type 1 (dm1) compared to usual care (optimistic): study protocol for randomised controlled trial," *Trials*, vol. 16, no. 1, pp. 1–19, 2015. [Online]. Available: http://dx.doi.org/10.1186/s13063-015-0737-7