Using Templates to Predict Execution Time of Scientific Workflow Applications in the Grid

Farrukh Nadeem, Thomas Fahringer
Institute of Computer Science, University of Innsbruck, Austria
farrukh@dps.uibk.ac.at

Abstract

Workflow execution time predictions for Grid infrastructures is of critical importance for optimized workflow executions, advance reservations of resources, and overhead analysis. Predicting workflow execution time is complex due to multeity of workflow structures, involvement of several Grid resources in workflow execution, complex dependencies of workflow activities and dynamic behavior of the Grid. In this paper we present an online workflow execution time prediction system exploiting similarity templates. The workflows are characterized considering the attributes describing their performance at different Grid infrastructural levels. A "supervised exhaustive search" is employed to find suitable templates. We also make a provision of including expert user knowledge about the workflow performance in the procession of our methods. Results for three real world applications are presented to show the effectiveness of our approach.

1 Introduction

Workflow applications from scientific and business domains typically consist of several different tasks (activities) with (complex) execution dependencies among them. The execution of such workflows in an automatic fashion, usually accomplished through Grid application development and runtime environment like ASKALON [3], requires to make several dynamic decisions (like resource selection, scheduling workflow activities etc.) regarding their optimized executions. While the performance can be gained by intelligently planning the execution of workflow activities in the Grid, there are also several overheads associated with them [12], which degrades the performance. An online workflow performance prediction service provides a decisive base for such optimization decisions.

Predicting the execution time of a workflow application is a complex and has been ignored so far due to involvement of several Grid resources and their inherent architectural and functional heterogeneity. In addition, a large variety of structures of the workflows and complex control flow and dataflow dependencies among their activities [13], external load and dynamic behavior of the Grid make the problem even more challenging, and debar the classical prediction methods at outset.

In this paper, we present an online workflow application execution time prediction system exploiting similarity templates. A *Template* refers to a set of selected workflow attributes. Workflow attributes correspond to the features describing workflow execution. We parameterize workflow application execution in terms of its attributes defining its structure and execution (Section 3), and use similarity templates to deliver the execution time predictions (Section 4). A comprehensive search method *Supervised Exhaustive Search* (Section 4.1) has been employed to define and search suitable templates. Supervised exhaustive search iteratively generates templates by selecting different workflow attributes and selects the best of them. The selection of workflow attributes in the templates is supported by the supervision of expert user through providing probabilities for the selection of different attributes. We define relationships among different workflow attributes and exploit these relationships during the search for suitable templates. Results are presented (Section 5) from a prototype development of our system, which has been integrated as a Grid service in ASKALON. Please note that in the course of this paper, by *workflow performance prediction* we mean execution time prediction of the entire workflow.

2 Scientific Workflow Applications

The workflow is a common way to model compound applications as a graph of activities, dependencies among them. Dependencies among the activities define their execution order and the dataflow from one activity to another. An activity in a workflow may be an executable to be executed or a service to be invoked.Usually, a Directed Acyclic Graph (DAG) is used to represent a simple workflow, however, additional constructs are used to show loops and con-

978-0-7695-3622-4/09 $25.00 © 2009 IEEE
DOI 10.1109/CCGRID.2009.77

ditional branching in complex scientific workflow applications [3]. Different Grid environments, like [9, 3, 10, 8], provide support to develop scientific workflows. Please note that in this paper, the terms workflow and workflow application bear the same meanings.

Execution of such workflows through a high level Grid-middleware system like ASKALON (as in our case) involves several high level services like resource broker, activity deployment service, execution enactment engine, scheduler, performance predictor etc. After workflow composition, workflow is submitted to an enactment engine, which then obtains optimized schedule of the workflow activities from a meta scheduling service. The meta scheduling service contacts a resource management service for available resources, and required software deployments, and obtains execution time predictions of individual activities on available Grid sites from a prediction service, to deliver the activity mappings and schedules. Enactment engine executes workflow activities according to this schedule. In case of more activities encountered at run time, new schedules are obtained. Execution of individual activities on different Grid sites is mainly influenced by local scheduling policies, queue wait time, external load at the time of execution and site computation power. The distributed execution of workflow activities may require data transfers among the sites. The control flow (critical execution path) of workflow activities, and enumeration of loops in the workflow are usually driven by the input problem size. All of these factors shape the overall performance of the workflow and collectively make performance prediction a complex problem.

3 Workflow Execution Similarity

It is well known that similar applications are more likely to have similar execution time [18, 7, 5] than the application which have a little or nothing in common. The authors in [18, 7, 5] have demonstrated the effectiveness of using historical run time information to identify *similar* jobs. However, they have restricted themselves to only a simple definition of similarity of single activities, and do not consider full workflows which is the target of this paper. In this work, we present a more sophisticated definition of similarity of *workflow application execution* and argue that carefully selected properties of similarity can lead to significant improvements in prediction accuracy. In addition, our contributions in this part is a flexible definition of similarity at a workflow level, employing workflow structural properties as well as the run time information at each phase of workflow execution.

In our definition of similarity, we characterize the workflow application execution at the levels of its composition and different phases of its execution. At coarse grain level,

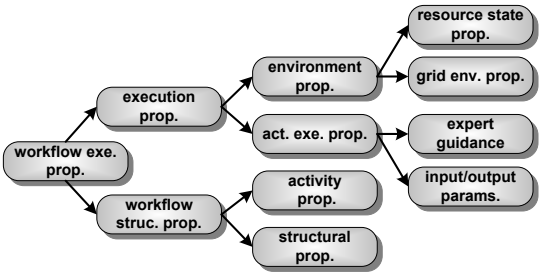


Figure 1. Workflow properties composition

we define workflow properties in terms of static (workflow structural) and dynamic (execution) properties. Workflow structural properties can be considered in terms of structure properties (like activity dependency (control and data flow dependency) etc.) and activity properties (like problem-size, executables, versions etc.). The workflow structural properties need to be defined and considered in workflow similarity definitions [19] to incorporate the effects of structural differences in different executions of the workflows.

The execution properties are defined in terms of activity execution properties and execution environment properties. Furthermore, activity execution properties are defined in terms of application problem size and scheduling policy/algorithm used by meta scheduler etc. The external load at the time of activity execution is considered in terms of number of jobs running in parallel with the workflow activity and number of CPUs occupied by the jobs running in parallel. Likewise, the execution environment properties include Grid sites selected for execution, resource state properties (like number of jobs in the queue, number of jobs running, free memory etc.) and Grid environment properties (like Grid-middleware). This composition of properties is shown in Figure 1. The workflow execution attributes at these levels are summarized in Table 1.

4 Similarity Templates

One of the core issues for the effectiveness of the predictions generated from the templates is to define a suitable set of templates and evaluate them by using historical traces. On the one hand, putting fewer or unrelated attributes in the templates will lead to generation of the classes which will classify the unrelated data instances together. A *class* or *template class* represents a set containing one specific value of each of the attributes in a template. The attribute values are specific to a workflow execution and are generated from the templates. On the other hand, putting too many attributes in a template will result in too many classes with lesser execution data classified in each. None of these cases will truly be a candidate for better predictions for a wide range of queries, and thus predictions with poor accuracy

Table 1. Workflow execution attributes in the Grid.

	Attributes	Rel. status*
Workflow Level	workflow name	A,D
	set of activities	D
	activity position	D
	control flow relation	D
	dataflow relation	D
	workflow start time	D
Application Level	App. name, desc.	A
	problem Size	D
	executables	A,D
	versions	D
	file Sizes	D
Execution Level	set of Grid sites	A
	time of submission	D
	Grid middleware	B
	scheduling strategy	D
Resource Level	jobs already in queue	D
	CPUs req. in queue	D
	jobs already running	D
	occupied CPUs	D
	parallel running jobs	D
	para. occupied CPUs	D
Policy Level	user-name	A,D
	group (VO)	A
	queues	D
Network Level	bandwidth	B
	latency	B
	no. of para. transfers	B

*A:antecedent, B:barron, D:descendant.

will be generated. Another contribution of this work is that we introduce the notion of relations between workflow attributes, and use these relations while including more than one attributes together in a template. The templates generated this way are more meaningful and result in predictions with higher accuracy.

We define and present the relations between different workflow attributes (wherever applicable) in Figure 2. Two attributes may be related to each other as *antecedent*(A) or *descendants*(D). A descendant is an attribute that additionally describes its antecedent. A descendant can be of type *physical* or *logical*. The logical descendants alone (as well as with their antecedents) can classify the related jobs together, however the physical descendants without their antecedents lack this capability. The attributes which do not have any antecedent or decedent are referred as *barren*(B) (indicated in Table 1). We set a policy for physical descendant to be included in the templates only with their antecedents.

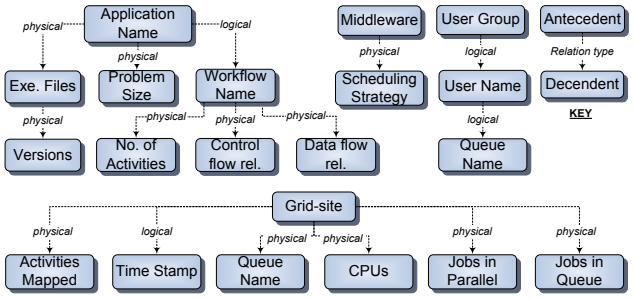


Figure 2. Workflow attributes relationships

While the number of attributes for workflow executions included in our traces are relatively large, an exhaustive search is needed to find the templates set which can better classify the similar workflow executions together. We introduce a search technique called *Supervised Exhaustive Search* that exploits the attribute relationships to construct a suitable and meaningful set of templates τ. τ contains a set of workflow attributes α_j (described in Table 1), and may be any subset of $\mathbb{P}\{(\alpha_1^1), (\alpha_1^2, \alpha_2^2), ..., (\alpha_1^j, \alpha_2^j, ..., \alpha_j^j)\}$, where α_i^j represents ith attribute in jth template, and $\mathbb{P}(\alpha_i^j)$ represents the power set of α_i^j.

4.1 Supervised Exhaustive Search

The Supervised Exhaustive Search (*SES*) proceeds iteratively over the entire workflow attributes set to construct the set of templates τ. We introduce the notion of supervision by embedding knowledge from the *expert user* to emphasize more on some attributes in the definition of templates than the others. The expert knowledge will help in selecting of attributes to better classify the similar executions together, and is done by assigning probabilities to the attributes. The attributes with high probability are more likely to be included in the templates. Without any supervision, the algorithm assigns equal probabilities to all the attributes. The process of supervised exhaustive search comprises two phases, and is presented in Algorithm 1.

In the first phase the SES algorithm iteratively generates templates from the antecedent and barren attributes according to attributes' probabilities (line 3 to line 5 and line 19 to line 24). In each generation of templates, a total of $size/n$ templates (line 20) containing n attributes is generated, where $size$ is the number of attributes in the $AttribSet$ and n is the generation (loop) counter. The probabilistic selection of each attribute is made based on their probabilities from $ProbSet$ provided by the expert user. In our current study we set equal probabilities for all attributes. In case of equal probabilities of attributes, to ensure fairness in selecting descendant attributes, the strength of each attribute is measured in terms of its number of descendants,

Algorithm 1 Supervised exhaustive search algorithm

Input: Set of all workflow attributes: $U = \{\alpha_1, \alpha_2, ..., \alpha_n\}$
R: Descriptions of attribute relationships
Set of ordered pair of attributes and their probabilities: $\rho = \bigcup_{i=1}^{n}\{(\alpha_i, pr_i)\}$
Output: τ: Set of templates

1: $\beta \Leftarrow getAntecendentsAndBarren(U, R)$
2: $m = |\beta|$
3: **for** $j = 1$ to m **do**
4: $t \Leftarrow getTemplates(j, m, \rho, \beta)$
5: **end for**
6: **for all** $t_i \in t$ **do**
7: $\tau \Leftarrow \tau \cup t_i$
8: $\zeta \Leftarrow getDescendents(t_i, U, R)$
9: $z \Leftarrow |\zeta|$
10: **for** $k = 1$ to z **do**
11: $x \Leftarrow getTemplates(k, z, \rho, \zeta)$
12: **for all** $x_i \in x$ **do**
13: $x_i \Leftarrow x_i \cup t_i$
14: **end for**
15: $\tau \Leftarrow \tau \cup x$
16: **end for**
17: **end for**
18: **return** τ

$getTemplates(n, size, ProbSet, AttribSet)$
19: $\chi = \{()\}$
20: **for** $i = 1$ to $\frac{size}{n}$ **do**
21: $\chi \Leftarrow$ template of probabilistically selected n $attributes$ from $AttribSet$ according to probabilities in $ProbSet$.
22: $probDec(selectAttrib)$
23: **end for**
24: **return** χ

and used for its selection. Attributes with more descendants are more likely to be selected. At each generation of templates the probability of selected attributes is decreased (line 22) to ensure *fairness* in selecting other attributes. The second phase generates new templates from those generated in the first phase by probabilistically including the descendant attributes of those already in the templates (line 6 to line 17), in the same manner as in the first phase.

4.2 Generating Predictions

The template generation phase is followed by generating classes from these templates by assigning different values (from traces) to the attributes in the template. The trace data is assigned to these classes according to attribute values and the classes are then used to generate predictions. While assigning data to the classes, we evaluate classes for different amount of historical data. This allows us to evaluate the effect of amount of historical data over the predictions, and also serves as a control for *fine tuning* our predictions. A lesser value of history size means that only recent values of historical data are used for predictions. We select the class containing number of data instances greater than a given threshold and with minimum standard deviation for generating predictions. In the current study, the threshold for minimum number of data instances in a template class is set to 50, because we found from our experiments that the classes with fewer data instances yield higher prediction errors. The predictions are generated by taking average of execution times of the all the data instances present in the selected class.

5 Empirical Evaluation

5.1 Scientific Workflows

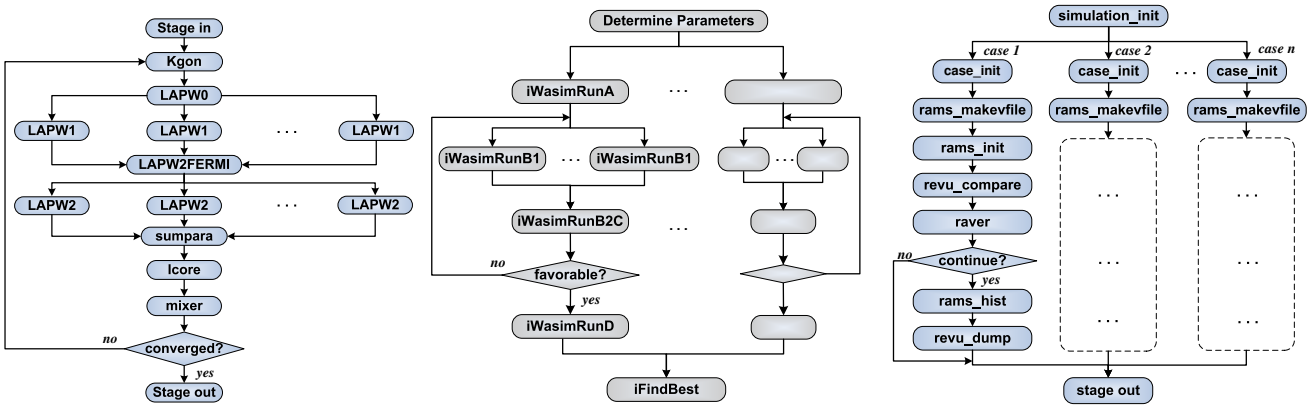
In our present study we consider three scientific workflows: Wien2k [11], Invmod [2] and MeteoAG [14]. Wien2k (Figure 3(a)) is a program package for performing electronic structure calculations of solids using density functional theory based on the full-potential augmented plane-wave and local orbital methods. Invmod (Figure 3(b)) aims at *The Water Flow and Balance Simulation Model*, to help in studying the effects of climatic changes on the water balance in order to obtain improved discharge estimates for extreme floods. The MeteoAG (Figure 3(c)) is a workflow for meteorological simulations, which produce atmospheric fields of heavy precipitation cases over the western part of Austria to forecast alpine watersheds and thunderstorms. We executed these workflows using ASKALON on the Austrian Grid infrastructure [15], which is a real Grid consisting of 28 Grid sites connected through high speed internet over various cities of Austria.

5.2 Experiments

Two kinds of metrics are used to evaluate the effectiveness of our prediction approach. The prediction accuracy is measured as *normalized absolute error* in predictions, defined as $|t_{real} - t_{pred}|/t_{real}$. The t_{real} and t_{pred} denote the real and predicted execution times respectively. Furthermore, to compare our results with similar approaches like Lee et al. in [6], the *relative error* is formulated as $\epsilon_{rel} = (t_{pred} - t_{real})/(t_{pred} + t_{real})$. The *absolute relative error* is measured by taking modulus of ϵ_{rel}, i.e. $|\epsilon_{rel}|$.

In all of the experiments the prediction time was measured in milli seconds and workload data set (the traces) was divided into training data set (the data instances assigned to the classes) and test data set (the data instances used to evaluate the predictions).

The accuracy of the predictions is evaluated across different problem-sizes of a workflow and also along different number of Grid sites on the Austrian Grid used to execute the workflow. Figure 4 depicts the accuracy of predictions in terms of absolute relative error for different problem-sizes of Wien2k, Invmod and MeteoAG workflows, where the number of Grid sites was varied from one to nine using different combinations of Grid sites on the Austrian



(a) Wien2K Workflow. (b) Invmod Workflow. (c) MeteoAG Workflow.

Figure 3. Workflow structures of three real-world scientific application.

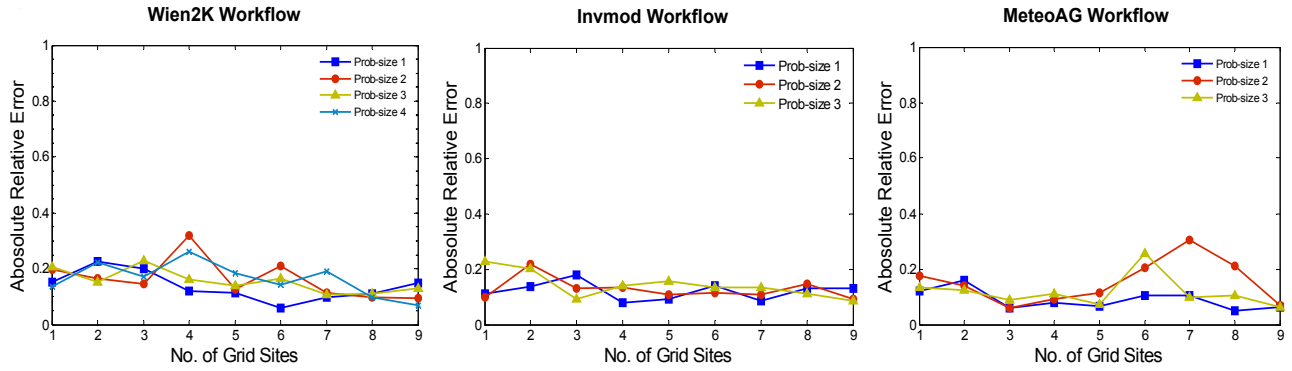


Figure 4. Absolute relative error across different problem sizes and sets of Grid sites.

Grid. For Wien2k the absolute relative error for the four problem-sizes (average per problem-size) ranged between 13%-16%, whereas it ranged between 10%-21% across different Grid-sizes (number of Grid sites) as average per Grid-size. We observed the best(worst) prediction (w.r.t accuracy) against 6%(31%) relative error for problem-size=1, Grid-size=6 (problem-size=2, Grid-size=4). The overall prediction accuracy of all the experiments for Wien2K was 15%. For Invmod the absolute relative error ranged between 12%-14% for the three problem-sizes (average per problem-size), and remained between 10%-19% for different Grid-sizes (average per Grid-size). The best(worst) prediction was observed against 8%(22%) of relative error for problem-size=1, Grid-size=4 (problem-size=3, Grid-size=1). The overall prediction accuracy of all the experiments for Invmod was 13%. The absolute relative error for MeteoAG ranged between 9%-15% and 6%-19% for the three problem-sizes and for different Grid-sizes, respectively. We observed the best(worst) prediction against 5%(30%) of relative error for problem-size=1, Grid-size=8 (problem-size=2, Grid-size=7). The MeteoAG workflow showed the best (of the three workflows) prediction accuracy (in terms of least absolute relative error) of all the experiments, only 11% absolute relative error (on average).

To reflect the distribution of relative errors, the histograms of relative errors of our experiments for the three workflows are shown in Figures 5, 6 and 7 respectively. It should be noted that the histograms show the spread of errors for several different runs of predictions for different problem-sizes and Grid-sizes, whereas Figure 4 shows absolute relative errors as an average of several runs for individual problem-sizes and Grid-sizes. Concluding the spread of relative error, the highest percentage mostly lies on and around center (zero) and decreases moving away from it, while a few of the predictions appear on the ends.

Now we present our prediction accuracy results as average *normalized absolute error*. The normalized absolute error for four problem-sizes of Wien2k and three problem-sizes each of Invmod and MeteoAG, and for different combinations of Grid sites (from one to nine) is shown in Figure 8. The average normalized absolute errors in our predictions for Wien2k workflow for four problem-sizes were

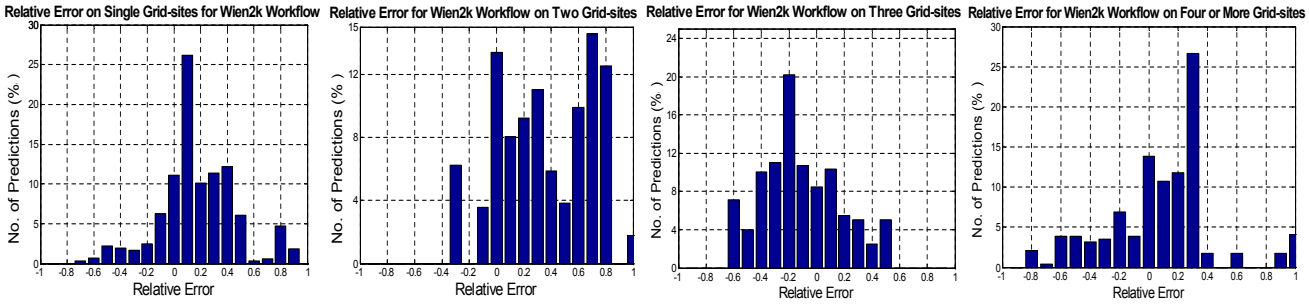


Figure 5. Histograms of relative error for Wien2k workflow on different Grid sites combinations.

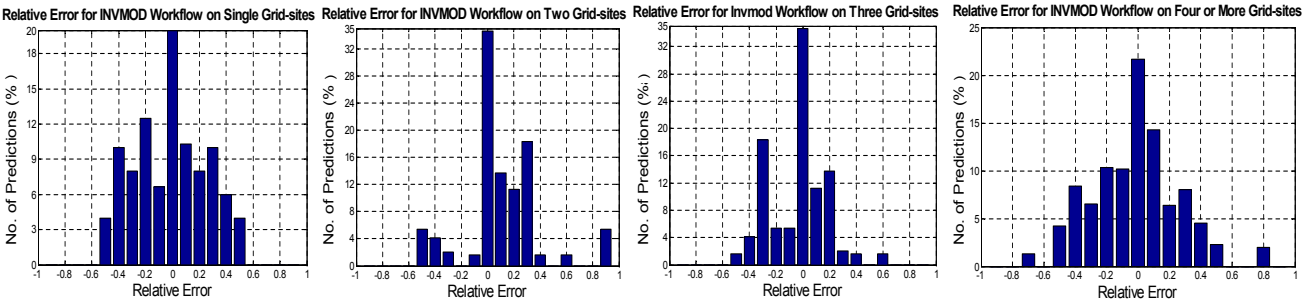


Figure 6. Histograms of relative error for Invmod workflow on different Grid sites combinations.

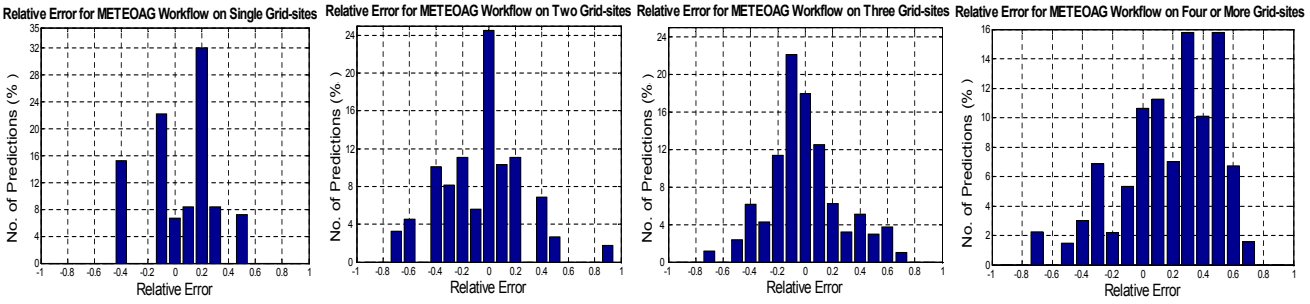


Figure 7. Histograms of relative error for MeteoAG workflow on different Grid sites combinations.

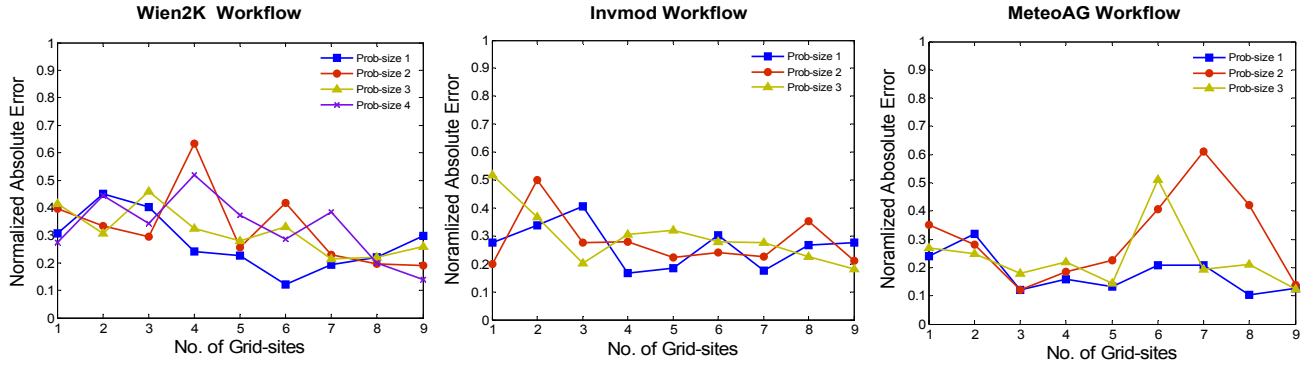


Figure 8. Normalized absolute error for different problem sizes and sets of Grid sites.

respectively 27%, 33%, 31% and 33%. The same for different number of Grid sites was 31%. We observed the maximum(minimum) normalized error across the four problem-sizes at two(six), four(nine), three(seven) and four(nine) Grid sites respectively. The overall best(worst) accuracy of predictions (minimum(maximum) error) was found against normalized error of 12% on six Grid sites (63% on four Grid sites). Similarly, the average normalized errors in our predictions for Invmod workflow for the three problem-sizes were 26%, 29% and 30% respectively. The same over different number of Grid sites was observed to be 28%. The respective maximum(minimum) normalized errors were against three(four), two(one) and one(nine) Grid sites. The overall best(worst) accuracy of predictions was found against normalized error of 17% on four Grid sites (52% on one Grid site). In sequel, the average normalized absolute errors in our predictions for MeteoAG workflow for the three problem-sizes over different combinations of Grid sites were respectively 18%, 30% and 23%. The same across different number of Grid sites was 24%. For the three problem-sizes the maximum(minimum) normalized error remained at different number of Grid sites; two(eight), seven(three) and six(five) respectively. The overall maximum error (61%) was found against problem-size=3 when executed on different combinations of seven Grid sites, and the minimum error (10%) was found for problem-size=1 when executed on different combinations of eight Grid sites.

We did not notice any specific increasing or decreasing patterns in relative or normalized errors w.r.t problem-size and Grid-size in the three workflows. Moreover, the best and worst predictions in the three workflows remained for different problem-sizes and Grid-sizes. Furthermore, increasing and decreasing trends of relative error for the respective problem-sizes appeared to be different for different number of Grid sites too. These observations indicate our approach's strong capability of encompassing the performance related effects (like parallelism, network transfers etc.) incurred from variations in problem-sizes and Grid-sizes. At the same time the errors in predictions are due to inherent dynamic nature of the Grid, which is very difficult to control.

Despite the fact that our problem domain (full workflow performance prediction) appears to be much more complex than those in [5, 18, 7] (targeting single activity performance predictions), quantitatively our prediction accuracy in terms of absolute relative error is 24% better than work in [5], 19% better than approach in [18], and 13% better than work in [7]. We believe that our higher accuracy is due to our more sophisticated definition of similarity, and exploiting attribute relationships. At the same time, we believe that accuracy of the predictions will improve when more historical data is available, because with respect to large dimensional space of workflow attributes the trace data is very less.

6 Conclusion and Future Work

In this paper we present a workflow performance prediction system using similarity templates. We define workflow execution in terms of different attributes reflecting workflow performance at different Grid infrastructure levels. The templates are generated from different workflow execution attributes. We also define workflow attribute relationships in terms of antecedent and descendants and set a policy to include an attribute in a template only if its antecedent (if any) is included. An exhaustive search method is introduced to generate and evaluate suitable templates. Our templates exploit a sophisticated and comprehensive definition of workflow execution similarity, considering different relationships among the attributes describing workflow execution. We evaluated our approach through a series of experiments for three real-world workflows for several different setups of application problem-sizes, and Grid sites (with different local setups). Our evaluations show reasonably better results than the related work employing similar techniques. We believe that our higher accuracy of predictions is due to comprehensive definition of workflow execution similarity including workflow structural attributes, attribute relationships and an exhaustive search for suitable templates. Furthermore, we also consider workflow problem size, which has not been accounted for by related work employing similar techniques.

In future, we intend to evaluate the effectiveness of our approach for other real world workflows. We also plan to find the effects of user provided information like probabilities set for different attributes describing the workflows, user estimated execution time etc. on the prediction accuracy.

7 Related Work

There is a series of related work for single activity execution time predictions using application profiles [17], analytical methods [1], soft benchmarks [4], historical data [7, 18] etc. But to the best of our knowledge, no effort has been made to predict workflow application performance with different variations of problem-sizes, Grid sites and high level middleware services.

Authors in [5, 18, 6] have used different attributes to define similarity of execution of single activities. Lee et al. [6] have also described attributes to define similarity of resource states and policies. In contrast to these approches which are for single activities execution time predictions, our work is for the complete workflow execution time prediction. We consider (almost) all major workflow attributes

describing its execution at different Grid infrastructural levels (like Grids-site, network etc.), in particular the workflow structure attributes to consider workflow structure similarities defined by Wombacher et al. [19]. Another major difference in our approach from the existing approaches (for activity execution time prediction) is the inclusion of problem-size attribute to describe a workflow execution.

Gibbons et al. [5] and Lee et al. [7] use a fixed set of templates, and Smith et al. [18] employ greedy search and genetic search to find effective templates. Comparing to these efforts, we employ a dynamic method of supervised exhaustive search to find suitable template sets. Supervised exhaustive search is better than the greedy search in way that it use a notion of external supervision, by assigning probabilities of selection to different attributes and thus guides the selection process. Moreover this is much more efficient than the genetic search used by Smith et al. in [18]. In contrast to these approaches, we also introduce a notion of attribute relationships to decide their inclusion while selecting attributes for suitable templates. Our better definition of similarity considering all major attributes, attributes relationships and external supervision by the user enable us to achieve higher accuracies (than the those in related work) in our predictions.

A closer effort to our focus is by Glatard et al. in [16]. The authors use probabilistic models to analyze workflow performance in the Grid, by considering execution times of individual activities, and data transfers between the activities and modeling the rest of time (taken in different execution phases) as random variable. This work assumes simple workflow applications, ignoring complex control flows between activities, and loops over different (sets of) activities. Moreover, variations in execution time due to input data set are not taken into account. In contradiction, we consider the workflow structure attributes as well as the problem-size used for workflow application execution. We also consider external load at the time of execution of the workflow, which allows us to gain higher accuracies in our predictions.

Acknowledgments

This work is partially supported by the European Union through IST-034601 edutain@grid project, Austrian Grid project (funded by the BMWF (Federal Ministry of Science and Research) through contract: GZ BMWF-10.220/0002-II/10/2007), and by Higher Education Commission of Pakistan.

References

[1] David A. Bacigalupo et al. An investigation into the application of different performance prediction methods to distributed enterprise applications. *J. Supercomput.*, 34(2):93–111, 2005.

[2] Dieter Theiner et al. Reduction of calibration time of distributed hydrological models by use of grid computing and nonlinear optimisation algorithms. In *Proceedings of the 7th International Conference on Hydroinformatics*, Sep. 2006.

[3] T. Fahringer, R. Prodan, R. Duan, J. Hofer, F. Nadeem, F. Nerieri, J. Q. Stefan Podlipnig, M. Siddiqui, H.-L. Truong, A. Villazon, , and M. Wieczorek. *Scientific Workflows for Grids*, chapter ASKALON: A Development and Grid Computing Environment for Scientific Workflows, page 530. Workflows for e-Science. Springer, Springer Verlag, 2007.

[4] Farrukh Nadeem et al. Soft benchmarks-based application performance prediction using a minimum training set. In *E-SCIENCE '06: Proceedings of the Second IEEE International Conference on e-Science and Grid Computing*, page 71. IEEE Computer Society, 2006.

[5] R. Gibbons. A historical application profiler for use by parallel schedulers. In *Job Scheduling Strategies for Parallel Processing*. Springer Verlag, 1997.

[6] Hui Li et al. Improving a local learning technique for queue-wait time predictions. In *CCGRID '06*.

[7] Hui Li et al. Predicting job start times on clusters. In *CCGRID '04*.

[8] Ian Taylor et al. Distributed computing with triana on the grid: Research articles. *Concurr. Comput. : Pract. Exper.*, 17(9), 2005.

[9] Ludäscher, Bertram et al. Scientific workflow management and the kepler system. *Concurrency and Computation: Practice & Experience*, 18(10):1039–1065, 2006.

[10] Oinn, T. et. al. Taverna: a tool for the composition and enactment of bioinformatics workflows. *Bioinformatics*, 20(17):3045–3054, Nov 2004.

[11] P. Blaha et al. *WIEN2k: An Augmented Plane Wave plus Local Orbitals Program for Calculating Crystal Properties*. Institute of Physical and Theoretical Chemistry, TU Vienna, 2001.

[12] R. Prodan and T. Fahringer. Overhead analysis of scientific workflows in grid environments. *IEEE Trans. Parallel Distrib. Syst.*, 19(3):378–393, 2008.

[13] J. Qin and T. Fahringer. Advanced data flow support for scientific grid workflow applications. In *SC '07*.

[14] S. Felix. et al. Performance, Scalability and Quality of the Meteorological Grid Workflow MeteoAG. In *2nd Austrian Grid Symposium, Innsbruck, Austria*, Sep. 2006.

[15] The Austrian Grid Consortium. http://www.austriangrid.at.

[16] Tristan Glatard et al. A probabilistic model to analyse workflow performance on production grids. In *CCGRID*, pages 510–517, 2008.

[17] Valerie Taylor et al. Using kernel couplings to predict parallel application performance. In *HPDC '02*.

[18] Warren Smith et al. Predicting application run times with historical information. *J. Parallel and Distrib. Comput.*, 2004.

[19] A. Wombacher and M. Rozie. Piloting an empirical study on measures forworkflow similarity. In *IEEE SCC*, pages 94–102, 2006.