Cost and Accuracy Sensitive Dynamic Workflow Composition over Grid Environments

David Chiu[1] Sagar Deshpande[2] Gagan Agrawal[1] Rongxing Li[2]
[1] Department of Computer Science and Engineering
[2] Department of Civil and Environmental Engineering and Geodetic Science
The Ohio State University
Columbus, OH 43210

Abstract

A myriad of recent activities can be seen towards dynamic workflow composition for processing complex and data intensive problems. Meanwhile, the simultaneous emergence of the grid has marked a compelling movement towards making datasets and services available for ubiquitous access. This development provides new challenges for workflow systems, including heterogeneous data repositories and high processing and access times. But beside these problems lie opportunities for exploration: The grid's magnitude offers many paths towards deriving essentially the same information albeit varying execution times and errors. We discuss a framework for incorporating QoS in a dynamic workflow composition system in a geospatial context. Specific contributions include a novel workflow composition algorithm which employs QoS-aware apriori pruning and an accuracy adjustment scheme to flexibly adapt workflows to given time restrictions. A performance evaluation of our system suggests that our pruning mechanism provides significant efficiency towards workflow composition and that our accuracy adjustment scheme adapts gracefully to time and network limitations.

1 Introduction

Technology has paved way for the development of increasingly advanced devices, which have led to a deluge of raw data being accumulated in a number of scientific fields. The lack of a unified and intuitive interface for access, analysis, and manipulation of such large-scaled datasets induce considerable difficulties for scientists and engineers. Obtaining certain information, for instance, may require a nontrivial composition of a series of data retrieval and process executions. To alleviate some of these challenges, scientific workflow management systems [1, 12, 13, 6, 19, 20, 14] emerged and have enabled scientists to compose complex workflows with automatic scheduling and execution.

Recently, grid computing has helped propel a movement towards making datasets and computing services[†] even more widely available. The result is a high number of distributed data repositories storing large volumes of datasets over links with potentially high latency and access costs. In these scenarios a workflow's overall execution time can be impacted by the high access costs of moving grid-based datasets. Often, there are multiple ways of answering a given query, using different combinations of data sources and services. Some combinations are likely to result in higher cost, but better accuracy, whereas other options might lead to quicker results, but lower accuracy. This could be because some data collection methods involve higher resolution than others, or because some datasets are available at servers with lower-access latencies than others. Similarly, data reduction methods, such as sampling, can speed up the time for remote retrieval and computations, but will likely lower the accuracy. Meanwhile, different users interacting with the query framework can have different requirements. Some users may want the answers the fastest, some may demand the most accurate answers, and others might prefer the faster of the methods which can meet certain accuracy constraints. It will be highly desirable if a workflow composition system can incorporate user constraints and preferences. In other words, we want to alleviate the users from the need of understanding the cost and accuracy tradeoffs associated with different datasets and services that could be used to answer a query.

This paper presents a framework for workflow composition that supports user preferences on time and accuracy. Our framework includes models for assessing cost associated with retrieving each dataset or executing any services on a given input. Similarly, we take as input models that predict accuracy associated with the results of a particular service, as a function of the accuracy of the data input to the service. User-specified constraints on accuracy and cost are also taken as input. The system automatically composes workflows while pruning candidates that cannot meet the constraints.

The uses for our proposed framework is three-fold. First, the integration of cost to our workflow composition algorithm allows the system to support user constraints. Secondly, these

[†]Throughout the paper we use *process* and *service* interchangeably as web service deployment of processes has become widely embraced by the grid community.

constraints increase the efficiency of our workflow composition algorithm by enabling the pruning of workflows at an early stage if QoS constraints are not met on the apriori principle. Lastly, we enable opportunities for service developers to expose parameters such that, when tuned, can affect the execution time and accuracy of the composed workflows, e.g., the sampling rate in a data reduction service. Given time restrictions or such physical limitations as low network bandwidth, our system is capable of dynamically suggesting recommended values for the exposed accuracy parameters in order to maximize the user's expectations. To the best of our knowledge, ours is the first automatic grid workflow composition system with these capabilities.

This framework has been incorporated in a system we previously developed [3] which supports high-level information querying of raw datasets from scientific domains. This was done through a combined support of keyword processing, domain-specific metadata indexing, and automatic workflow composition. Although our system is general, this particular implementation has been in the context of geospatial data. Here, data sources ranging from satellites to environmental sensors, which continuously produce readings, generate massive datasets that are typically stored in their original low-level formats on disks across networks. Likewise, access and manipulation of these datasets are provided through grid services that are made, in most cases, publicly available.

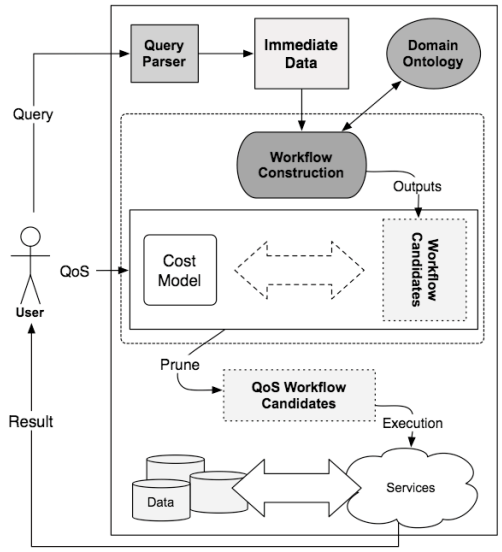

Figure 1. System Overview

Our performance evaluation shows that the additional cost model induces negligible overhead, and in fact, improves the overall execution time of the algorithm through early workflow pruning. We also devised experiments to show the system's efficacy in meeting user constraints and in the provision of robustness under varying network bandwidths, which is often prevalent in grid environments. Our system was evaluated under a real grid environment against user constraints on time as well as network bandwidth limitations. In the worst case in our experiments, we observed an average deviation of 14.3% from the desired time constraints and 12.4% digression from the ideal expectations amidst varying network bandwidths. The remainder of this paper is organized as follows. An overview of our system is presented in Section 2. In Section 3 we discuss technical details of our cost model and workflow composition algorithm. An evaluation of the system is given in Section 4, and a comparison of our work with related research efforts follows in Section 5. Finally, we conclude and discuss future directions in Section 6.

2 System Overview

A conceptual view of our system is shown in Figure 1. We present a quick overview of each component's requirements while detailed descriptions can be found in [3]. First, semantic descriptions of the available data and services including their relationships must be provided to the system. In our case, for geospatial datasets, their metadata is specified in CSDGM (Content Standard for Digital Geospatial Metadata) as recommended by the Federal Geographic Data Committee [7]. CSDGM annotates data files with such descriptions as area coverage, date of creation, coordinate system, etc. Our processes are in forms of web services — their interface is described in WSDL [4].

In addition, our metadata also includes domain information that will aid in supporting autonomous determination of workflow correctness such as dependencies and context suitability. Effective classification of datasets and services along with a description of their relationships will help filter the set of services to those suitable for execution. The standard ontology descriptor, Web Ontology Language (OWL) [5], is utilized denote relationships among domain concepts, datasets, and services. The ontology, shown in Figure 2, is simple as compared to geospatial specific ontologies [15]. However, its generality theoretically allows us to easily port our system to other domains (although, we have not yet attempted this).

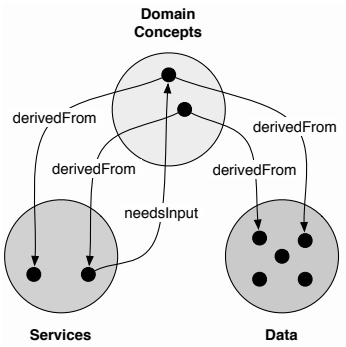

Figure 2. Ontology for Domain Specific Semantic Description

Among our system's goals, one is to provide support for high-level user queries, which implies that a somewhat sophisticated query parser should be included. Specifically, this component parses a query into relevant concepts in our scientific domain, and when available, substantiates the

parsed concepts with user given values. The QoS constraints, if available, are input by the user with the keyword pairs: *QoS:Time=seconds* and *(QoS:ErrConcept=concept, QoS:Err=value)*. The latter pair for applying error restriction is important to note, since workflows may involve multiple error models. For instance consider a service such that, when given an aerial image of a city and corner coordinates, crops and returns a new image with the specified boundaries. Here, errors can include the resolution of the cropped image or measurement discrepancies involving the actual cropping, e.g., the cropping service is accurate up to $\pm$5cm. Multiple error constraints may be coupled together in the same query. In this case, a user might demand:

```
``crop an aerial image of Columbus, Ohio
 with northbound=(x,y), southbound=(x,y), ...
 (QoS:Time=120s AND
 (QoS:ErrConcept=resolution, QoS:Err=90ppi) AND
 (QoS:ErrConcept=perimeter, QoS:Err=+-5cm))''
```

It is worth noting that if only one constraint is given, then system attempts to abide the restriction while optimizing the undefined constraint, and that if neither is provided, the system will execute the workflow containing the lowest error. The user may also request that all workflows meeting the constraints be returned. In this case the user is given time and error predictions of each workflow, and he/she selects which to execute. Given this well-structured query, appropriate services and datasets must be selected for use and their composition is reified dynamically through consultation with the domain ontology. Through this process, the workflow construction engine enumerates a *set* of valid workflow candidates such that when each is executed, returns a suitable response to the query.

From the set of candidates, the workflow construction engine must then examine the cost of each in order to determine a subset that meet user constraints. Additionally, this component can dynamically adjust accuracy parameters in order to meet expected time constraints set by the user. Although shown as a separate entity for clarity, the pruning mechanism is actually pushed deep within the workflow construction engine. The remaining candidate set is sorted top-down according to either the time or accuracy constraint (depending on preference) to form a queue. The execution of workflows is carried out and the presence of faults within a certain execution, caused by such factors as network downtime or data/process unavailability, triggers the next queued workflow to be executed to provide the most optimal possible response.

3 Problem Statement and Technical Details

In practice most scientific workflows can be expressed as directed acyclic graphs where the vertices denote services and data elements and directed edges represent dependencies. Workflows can be also recursively described recursively as follows: Given some arbitrary dataset, D and a set of services S, a workflow w is defined

$$w = \begin{cases} \epsilon \\ d \\ (s, P_s) \end{cases}$$

such that terminals ϵ and $d \in D$ denote a null workflow and a data instance respectively, and nonterminal $(s, P_s) \in S$ is a tuple where s denotes a service with parameter list $P_s = (p_1, \ldots, p_k)$ and each p_i is itself a workflow. Then given a set of workflows $W_q = \{w_1, \ldots, w_n\}$ capable of answering some user query q, our goal is to identify some subset $R_q \subseteq W_q$ such that each workflow $r \in R_q$ either meets or exceeds a user's QoS constraints, namely, processing time and accuracy of results.

3.1 Modeling Workflow Cost

Our previous work [3] showed that it was feasible to exhaustively generate all workflows capable responding to a query in an efficient manner. In fact, we demonstrated that this *enumeration* process was conservatively upper bounded by Depth-First Search. The enumeration algorithm, however, did not take into account pruning mechanisms and thus unable control the growth rate of W_q (the set of candidates) as well as determining each workflow's quality. To this end, we propose that the workflow's time cost is estimated by

$$T(w) = \begin{cases} 0, & \text{if } w = \epsilon \\ t_{net}(d), & \text{if } w \in D \\ t_x(s, P_s) + & \text{if } w \in S \\ \quad t_{net}(s, P_s) + \max\limits_{p_i \in P_s} T(p_i), & \end{cases}$$

If workflow w is a base data element, then $w = d$, and the cost is trivially the network transmission time, t_{net}, taken for this element. When w is a service, then $w = (s, P_s)$, and its time can be summarized as the sum of the service execution time t_x, network transmission time of its product, and, recursively, the maximum time taken by all of its parameters. The second cost function, $E(w)$, represents the error estimation of a given workflow.

$$E(w) = \begin{cases} 0, & \text{if } w = \epsilon \\ \sigma(d), & \text{if } w \in D \\ \sigma(s, P_s) + \max\limits_{p_i \in P_s} E(p_i), & \text{if } w \in S \end{cases}$$

Due to the heterogeneity of datasets, processes, and precision of measuring intruments, it is expected that disparate workflows will yield results with varying measures of accuracy. Again, at the base case lies with the expected error of a particular dataset, $\sigma(d)$. An error value can also be attributed to a service execution, $\sigma(s, P_s)$. For instance, inaccuracies will be introduced if an extrapolation service is used predict an expected value. Moreover, the errors introduced in earlier stages of the execution path must be propagated towards the result. $\sigma(d)$ and $\sigma(s, P_s)$ must thus be provided and evaluated on a per-domain basis since these measures are application sensitive. In both models the value of a NULL workflow, ϵ, is trivially 0.

The obvious goal is providing prudent and reliable measures since cost is the determining factor for pruning workflow candidates. Furthermore, the online computation of cost should only require diminutive overhead. For each service, we are interested four separate models: The $T(w)$ term itself involves the implementation of three distinct models for service execution time (t_x), network transmission time (t_{net}), and, implicitly, an estimation of output size ($size_d$). For t_x, we sampled service runtime by controlling various sized inputs and generating multi-regression models. $size_d$ was computed on a similar basis (note that $size_d$ is known for files). The network transmission time t_{net} was approximated as the ratio of $size_d$ over bandwidth between nodes that host each service or data. Regression, however, cannot be used to reliably capture the capricious nature of an error model. It depends heavily on the application's mechanisms and is largely domain specific. Thus, our model must capture arbitrarily complex equations given by domain experts.

3.2 An Example Error/Accuracy Model

We present an error model through the geospatial application of water level derivation. Envision that a user is interested in water levels pertaining to a point in some body of water. One method to answer this query involves gathering data from multiple gauge stations, which are typically situated along the shorelines. Suppose we choose K different gauge stations for interpolation, the queried water level, Z_{WL} can be obtained with

$$Z_{WL} = \frac{Z_1/d_1^2 + \ldots + Z_K/d_K^2}{1/d_1^2 + \ldots + 1/d_K^2}$$

where Z_i denotes water level information at the ith gauge station and d_i represents the its distance from the interpolated point. The estimated error at the interpolated point, $\sigma_{Z_{WL}}$, would be given by

$$\sigma_{Z_{WL}} = \sqrt{\sum_{i=1}^{K}\left(\frac{\partial Z_{WL}}{\partial Z_i} \times \sigma_{Z_i}\right)^2 + \sum_{i=1}^{K}\left(\frac{\partial Z_{WL}}{\partial d_i} \times \sigma_{d_i}\right)^2}$$

where σ_{Z_i} denotes the vertical reading error at the ith gauge station and σ_{d_i} is its distance to the queried point.

In our framework the former equation for interpolation would be deployed as a service onto the grid, and the latter error model is appended to the service's description. In an effort to keep the error models as lightweight as possible, we maintain some data locally, such as the average σ_{Z_i} values for each gauge station.

3.3 Workflow Enumeration and Pruning

As previously mentioned, our workflow enumeration (Algorithm 1) is a glorification of Depth-First Search[‡]. Starting from the targeted domain concept, we explore each dependent path in the given ontology until it leads to a sink (base data or NULL node). Every concept (intermediate or target) can be realized by various datasets or services. (Line 6) obtains a set of data or service nodes towards the derivation of some domain concept. Each element in this set marks a potential workflow candidate either in the direction of available data or a service product. In the former case (Line 8) w is a base data workflow and immediately considered for inclusion. The latter case (Line 11) considers the service at-hand along with its parameters. Each service parameter is essentially a new subtarget concept in our ontology, and consequently, the algorithm is called recursively to solve for a set of its subworkflows. For instance, consider a service workflow with two parameters of concepts a and b: $(s,(a,b))$. Assuming that target concepts a and b are derived using some set of workflows $W_a = \{w_1^a, w_2^a\}$ and $W_b = \{w_1^b\}$, then the workflows derivable from s includes $W_s = \{(s,(w_1^a,w_1^b)),(s,(w_2^a,w_1^b))\}$. To clarify, if every service parameter can be substantiated with at least one subworkflow, then the full set of workflow compositions is established through a cross product of its derived parameters (Line 23). Each element from the cross product is then coupled with the service and considered for inclusion.

Algorithm 1 enumWF($target$, QoS)

```
1:  W ← ∅
2:  /* static array for memoization */
3:  global subWorkflows[...]
4:
5:  /* B denotes the set of all data/service elements that can
    be used to derive target concept */
6:  B ← derives(target)
7:  for all β ∈ B do
8:    if β ∈ D then
9:      w ← data(β)
10:     W ← QoSMerge(W, w, ∞, ∞, QoS)
11:   else
12:     /* β ∈ S */
13:     /* P_β denotes the set of service's params */
14:     P_β ← getServiceParams(β)
15:     Δ ← ∅
16:     for all p ∈ P_β do
17:       if exists(subWorkflows[p.concept]) then
18:         Δ ← Δ ∪ subWorkflows[p.concept]
19:       else
20:         Δ ← Δ ∪ enumWF(p.concept, QoS)
21:       end if
22:     end for
23:     Params ← crossProduct(Δ)
24:     for all pm ∈ Params do
25:       w ← srvc(β, pm)
26:       W ← QoSMerge(W, w, ∞, ∞, QoS)
27:     end for
28:   end if
29: end for
30: subWorkflows[target] ← W
31: return W
```

When a workflow becomes a candidate for inclusion, QoSMerge (Algorithm 2) is invoked to provide the decision:

[‡]Details on query parsing, the handling of immediate data values, and data identification can be found in [3]

Algorithm 2 QoSMerge(W, w, t', e', QoS)

```
1: /* both QoS constraints are met */
2: if T(w) ≤ QoS.Time ∧ E(w) ≤ QoS.Err then
3:    /* insert w in QoS sorted position (not shown) */
4:    return (W ∪ w)
5: end if
6: /* convergence of model estimations */
7: if |T(w) − t'| ≤ C_T ∧ |E(w) − e'| ≤ C_E then
8:    /* prune w by excluding from W*/
9:    return W
10: else
11:   α ← getAdjustableParam(w)
12:   γ ← suggestParamValue(α, w, QoS)
13:   /* w_γ is param adjusted version of w */
14:   w_γ ← w.setParam(α, γ)
15:   return QoSMerge(W, w_γ, T(w), E(w), QoS)
16: end if
```

prune, include as-is, or modify workflow accuracy. For simplicity, we consider a single error model in this algorithm, and hence, just one adjustment parameter. In practice workflows may involve various error aspects for consideration such as the example given in Section 2.

QoSMerge inputs the following arguments: (1) W, the set of workflow candidates, (2) w, the current workflow under consideration, (3) t' and (4) e' are the predicted time and error values of the workflow from the previous iteration, and (5) QoS is the QoS object from the original query. In summary this algorithm merges the given workflow candidate, w, for with the result set W if it meets the QoS time and error constraints. If either constraint is not met, then the system to provides a suitable value for α, the adjustment parameter of w, given the QoS. Currently, suggestParamValue derives its value by inversing the time model, $T(w)$ to solve for the desired parameter given the constraints.

Taken with the suggested parameter, the procedure is invoked recursively on the adjusted workflow, w_γ. After each iteration, w is adjusted, and if both constraints are met, returned for inclusion. However, when the algorithm detects that the modifications to w provide insignificant contributions to its effects on $T(w)$ and $E(w)$ then w is subsequently pruned. This condition is shown on (Line 7) of Algorithm 2, and convergence thresholds C_T and C_E are predefined per service. The values of t' and e' of the initial QoSMerge call on (Lines 10 and 26) of Algorithm 1 are set to ∞ for dispelling the possibility of premature convergence. When either QoS.Time or QoS.Err are not given by the user, their respective models are actually never invoked, and QoSMerge becomes the trivial procedure of immediate inclusion of workflow candidate w. In Algorithm 2, this is equivalent to assigning the $QoS.*$ constraints to ∞.

3.4 An Example Query

The example query is consistent with the error modeling example presented in Section 3.2.

```
''return water level of at (482593, 4628522) on
01/30/2008 at 00:06''
```

In our system, this query may be solved in two service workflow directions, i.e., the target concept of water level contains two distinct service nodes for derivation. One approach employs services to retrieve Deep Web data from some K nearest water gauge stations to the queried location and interpolates their readings for a more accurate result. Another method consults a water surface simulation model, whose access is made available over the grid.

The following is the actual output from our system given the above query. The values within [...] are the time and error predictions made by our system models. The actual execution times of both workflows are 3.251 sec for w_1 and 1.674 sec for w_2. Note that QoS constraints were not set as to show the comprehensive workflow candidate set with their estimated costs without the effect of pruning. With pruning, if QoS.Time was assigned a value of 2.0, then w_1 would have been discarded at the time of composition of its initial workflow, SRVC.GetGSListGreatLakes(NULL).

```
w_1 =
[t_total=3.501, err=0.004
 --> t_x=1 t_net=0 d_size=0 ]
SRVC.getWL(
  X=482593, Y=4628522, StnID=
    [t_total=2.501, err=0.004
     --> t_x=0.5, t_net=0, d_size=0]
    SRVC.getKNearestStations(
      Long=482593, Lat=4628522, ListOfStations=
        [t_total=2.01, err=0
         --> t_x=2.01 t_net=0 d_size=47889]
        SRVC.GetGSListGreatLakes(NULL)
      RadiusKM=100, K=3
    )
  time=00:06, date=01/30/2008
)

w_2 =
[t_total=2.00, err=2.4997
 --> t_x=2 t_net=0 d_size=0]
SRVC.getWLfromModel(
  X=482593,Y=4628522,time=00:06,date=01/30/2008
)
```

4 Experimental Results

Two main goals are addressed in our experiments: First, to assess the overhead of workflow enumeration and the impact of pruning. The second set of experiments focused on evaluating our system's ability to consistently meet QoS constraints.

For our experiments, we employ three nodes from a real grid environment. The local node runs our workflow system, which is responsible for composition and execution. Another node containing all needed services is located within the Ohio State University campus on a 3MBps line. Finally, a node containing all datasets is located in another campus, Kent State University, about 150 miles away. Here the available bandwidth is also 3MBps.

4.1 Overheads of Workflow Enumeration

In the first experiment, we want to show that the overhead introduced by the cost model is minimal. We generated a syn-

thetic ontology capable of allowing the system to enumerate thousands of workflows for some arbitrary query. Having this many workflow candidates is unlikely in practice, but serves as a way for us to evaluate scalability (shown in Figure 3). This was repeated for an increasing number of candidates (i.e., $|W| = 1000, 2000, \ldots$) on 4 different situations (solid lines) without pruning (i.e., the worst case scenario where all $|W|$ workflows meet any given constraint). The four settings

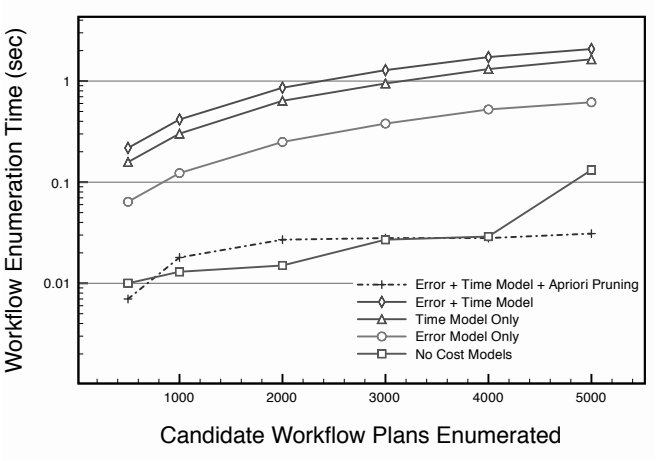

Figure 3. Cost Model Overhead and Pruning

correspond to user queries with (a) no QoS constraints, (b) only error constraints, (c) only time constraints, and (d) both constraints. Naturally, enumeration algorithm runs in proportional time to the number of models supported. It is also expected that the time cost model itself outweighs the error model as it evaluates three distinct predictions, t_x, t_{net}, and $size_d$. To evaluate our algorithm's efficiency, we altered our previous experimental setting to contain exactly one workflow within each candidate set that meets both time and error constraints (all other candidates should be pruned in the early stages of workflow composition, which corresponds to the best case). For each setting of $|W| + 1$, the algorithm now prunes $|W|$ workflows (dashed line). The results show that the pruning algorithm is as efficient as, and later, begins to outperform the *no-cost* model since the amount of subworkflows to consider is minimized.

Table 1. Experimental Queries

$Query_1$	"return surface change around (482593, 4628522) from 07/08/2000 to 07/08/2005"
$Query_2$	"return shoreline extraction at (482593, 4628522) on 07/08/2004 at 06:18"

4.2 Meeting QoS Constraints

Queries 1 and 2 (Table 1) are designed to demonstrate QoS management. Specifically, $Query_1$ must take two digital elevation models (DEM) from the given time periods and location and output a new DEM containing the difference in land elevation. The shoreline extraction in $Query_2$ involves manipulating the water level and a DEM for the targeted area and time. Although less computationally intense than $Query_1$, execution times for both are dominated by data movement.

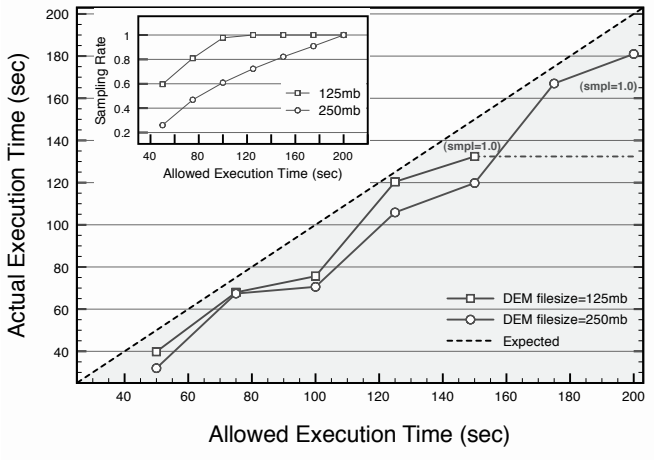

Figure 4. Time Expectations for Query 1

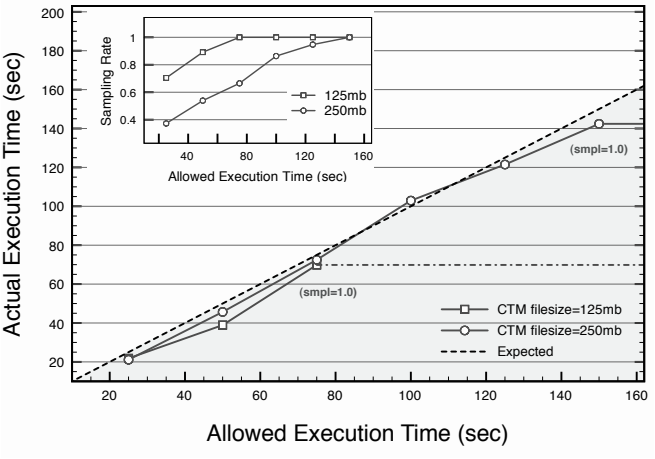

Figure 5. Time Expectations for Query 2

This becomes problematic for low time constraints, but can be mitigated through data reduction, which we implement via sampling along each of the DEM's dimensions. In both queries the sampling rate is the exposed accuracy adjustment parameter, and the goal of our system is to suggest the most appropriate sampling rates such that the actual execution time is nearest to the user allowance. All services involved in these queries have been trained to obtain prediction models for cost estimation.

Figures 4 and 5 show the actual execution times of each query against user-allowed execution times. The dashed line which represents the end-user's expectations is equivalent to the time constraint. The DEM sampling rate, which is em-

bedded in the figures, is inversely proportional to the error of our workflow's payload. A juxtaposition of the outer and embedded figures explains why, in both results, the actual execution time of the workflow pertaining to smaller DEMs flattens out towards the tail-end: at the expected time constraint, it has already determined that the constraint can be met without data reduction.

The gap observed when $AllowedExecutionTime = 100$ in Figure 4 is exposing the fact that the system was somewhat conservative in suggesting the sampling rate for that particular point, and a more accurate workflow could probably have been reached. Situations like these exist due to imprecisions in the time model (we used multi-linear regresion). The implementation of the models, Between the two DEM size configurations, $Query_1$ strays on an average of 15.65 sec (= 14.3%) from the expected line and $Query_2$ by an average of 3.71 sec (= 5.2%). Overall, this experiment shows that our cost model and workflow composition scheme is effective. We obtained consistent results pertaining to error QoS, but these results are not shown due to space constraints.

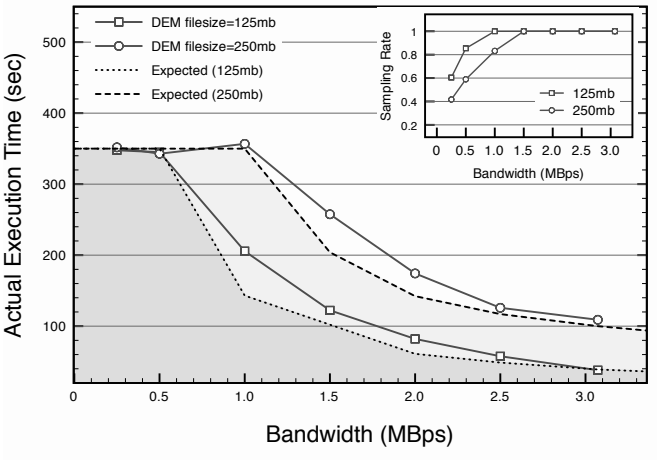

Figure 6. Varying Bandwidth for Query 1

The next experiment shows actual execution times against varying bandwidths of our network links. Ideal expectations in this experiment are much different than the linear trend observed in the previous experiment. When bandwidth is low, sampling is needed to fit the execution within the the given time constraint (we configured this at 350 sec in both experiments). Next, when the bandwidth is increased beyond the point where sampling is necessary, we should observe a steady drop in actual execution time. Finally, this declining trend should theoretically converge to the pure execution time of the services with ideal (zero) communications delay and network overhead.

As seen in Figures 6 and 7, the actual execution times lie consistent with the expected trends. Between the two DEM sizes, $Query_1$ strays on average 13.79 sec (= 6.7%) from the ideal line and $Query_2$ on average 16.05 sec (= 12.4%). It is also within our expectations that the actual execution times generally lie above the ideal lines due to communication overheads and actual network fluctuations. We believe that our experimental results suggest that the system provides and maintains robustness against user defined cost as well as computing costs within dynamic grid environments. Specifically, the accuracy parameter suggestion algorithm was shown to gracefully adapt workflows to restrictions on time and networks.

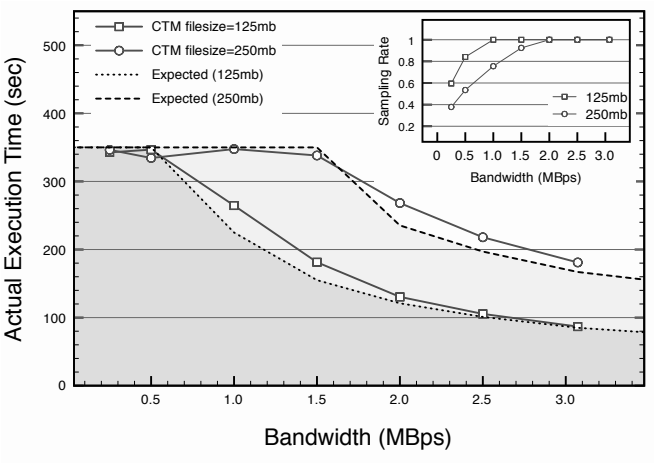

Figure 7. Varying Bandwidth for Query 2

5 Related Works

The Data Grid opened up various opportunities for the scientific community to share resources including, among others, large-scaled datasets and services. This prompted the emergence of scientific workflows for modeling and managing complex applications and processes for data and information derivation. Pioneering works towards this front include Chimera [8], which supports detailed recording of data provenance and enables the reuse of well-designed workflows to generate or recreate derived (*virtual*) data. Other systems including Kepler [1], Taverna [13], and Triana [12], like ours, heavily utilize domain specific metadata. These allow domain experts to define workflows through intuitive interfaces, and the workflow components are then automatically mapped to the grid for execution.

The direction towards automatic discovery, scheduling, and execution of scientific workflows has been reified through a number of efforts in dynamic workflow composition [10, 2, 17, 18, 9], which is a subset of the AI Planning Problem. Our framework again poses no exception, as our domain ontology and enumeration algorithm may also be typified as planning components, leading to the goal of derived data. In particular, Pegasus [6, 11] supports creation and execution of highly complex workflows over the grid. This system automates the management of grid resources and mapping of workflow components for execution.

While our system adopts strongly established features from the above works, it has the following distinguishing characteristics. We envision an *on-demand* domain level querying framework that is applicable to users from naïve

to expert. Data derivation is made available immediately through a high-level keyword interface and abstraction of user-level workflow creation via automatic service composition. The aforementioned workflow managers require some user intervention involving definition of an abstract workflow template (e.g., Pegasus) or recondite rules and goals for the planning algorithms in [14, 20, 16]. Another feature of our work involves the adaptability to QoS constraints. Some workflow managers, such as Askalon [19], attempt to minimize the execution time of workflows by employing performance predictors which factor into scheduling decisions. Our system differs in that the overall goal is not specifically the minimization of execution time. Instead, we focus on an accuracy-oriented task where workflow execution times may be manipulated to fit within the required QoS through error-level adjustment.

6 Conclusion and Future Directions

The work reported herein discusses our approach to bring QoS awareness in the form of time and accuracy constraints to the process of workflow composition. Our framework, which includes methods for error and execution time prediction, employs the apriori principle to prune potential workflow candidates. Our results show that the inclusion of such cost models contributes negligible overhead, and in fact, can reduce the overall workflow enumeration time through pruning unlikely candidates at an early stage. In addition, our dynamic accuracy parameter adjustment offers robustness by allowing workflows to be flexibly accurate for meeting QoS constraints under varying network speeds. Our system was evaluated against actual user constraints on time and the network bandwidth limitations. In its worst case, our system maintained actual execution times that deviate no more than 14.3% from the expected values on average, and no worse than 12.4% from the ideal line when presented with varying network bandwidths.

The dominant bottleneck with any workflow manager is process execution and data transmission. We believe that this problem can be mitigated by caching partial workflow results on intermediate nodes in the grid. Interferences with our cost models arise as we attempt to model a series of implementing mechanisms, including data migration, workflow redirection, and fault handling.

Acknowledgments

This work was supported by NSF grants 0541058 and 0619041. The equipment used for the experiments reported here was purchased under the grant 0403342.

References

[1] I. Altintas, C. Berkley, E. Jaeger, M. Jones, B. Ludscher, and S. Mock. Kepler: An extensible system for design and execution of scientific workflows, 2004.

[2] J. Blythe, E. Deelman, Y. Gil, C. Kesselman, A. Agarwal, G. Mehta, and K. Vahi. The role of planning in grid computing. In *The 13th International Conference on Automated Planning and Scheduling (ICAPS)*, Trento, Italy, 2003. AAAI.

[3] D. Chiu and G. Agrawal. Enabling ad hoc queries over low-level geospatial datasets. Technical report, The Ohio State University, 2008.

[4] E. Christensen, F. Curbera, G. Meredith, and S. Weerawarana. Web services description language (wsdl) 1.1.

[5] M. Dean and G. Schreiber. Owl web ontology language reference. w3c recommendation, 2004.

[6] E. Deelman, G. Singh, M.-H. Su, J. Blythe, Y. Gil, C. Kesselman, G. Mehta, K. Vahi, G. B. Berriman, J. Good, A. C. Laity, J. C. Jacob, and D. S. Katz. Pegasus: A framework for mapping complex scientific workflows onto distributed systems. *Scientific Programming*, 13(3):219–237, 2005.

[7] Metadata ad hoc working group. content standard for digital geospatial metadata, 1998.

[8] I. T. Foster, J.-S. Vöckler, M. Wilde, and Y. Zhao. Chimera: A virtual data system for representing, querying, and automating data derivation. In *SSDBM '02: Proceedings of the 14th International Conference on Scientific and Statistical Database Management*, pages 37–46, Washington, DC, USA, 2002. IEEE Computer Society.

[9] K. Fujii and T. Suda. Semantics-based dynamic service composition. *IEEE Journal on Selected Areas in Communications (JSAC)*, 23(12), 2005.

[10] Y. Gil, E. Deelman, J. Blythe, C. Kesselman, and H. Tangmunarunkit. Artificial intelligence and grids: Workflow planning and beyond. *IEEE Intelligent Systems*, 19(1):26–33, 2004.

[11] Y. Gil, V. Ratnakar, E. Deelman, G. Mehta, and J. Kim. Wings for pegasus: Creating large-scale scientific applications using semantic representations of computational workflows. In Proceedings of the 19th Annual Conference on Innovative Applications of Artificial Intelligence (IAAI), Vancouver, British Columbia, Canada, July 22-26,, 2007.

[12] S. Majithia, M. S. Shields, I. J. Taylor, and I. Wang. Triana: A Graphical Web Service Composition and Execution Toolkit. In *Proceedings of the IEEE International Conference on Web Services (ICWS'04)*, pages 514–524. IEEE Computer Society, 2004.

[13] T. Oinn, M. Addis, J. Ferris, D. Marvin, M. Senger, M. Greenwood, T. Carver, K. Glover, M. R. Pocock, A. Wipat, and P. Li. Taverna: a tool for the composition and enactment of bioinformatics workflows. *Bioinformatics*, 20(17):3045–3054, 2004.

[14] S. R. Ponnekanti and A. Fox. Sword: A developer toolkit for web service composition. In *WWW '02: Proceedings of the 11th international conference on World Wide Web*, 2002.

[15] R. Raskin and M. Pan. Knowledge representation in the semantic web for earth and environmental terminology (sweet). *Computer and Geosciences*, 31(9):1119–1125, 2005.

[16] Q. Sheng, B. Benatallah, M. Dumas, and E. Mak. Self-serv: A platform for rapid composition of web services in a peer-to-peer environment. In *Demo Session of the 28th Intl. Conf. on Very Large Databases*, 2002.

[17] B. Svrivastava and J. Koehler. Planning with workflows - an emerging paradigm for web service composition. In *Workshop on Planning and Scheduling for Web and Grid Services*. ICAPS, 2004.

[18] P. Traverso and M. Pistore. Automated composition of semantic web services into executable processes. In *3rd International Semantic Web Conference*, 2004.

[19] M. Wieczorek, R. Prodan, and T. Fahringer. Scheduling of scientific workflows in the askalon grid environment. *SIGMOD Rec.*, 34(3):56–62, 2005.

[20] D. Wu, E. Sirin, J. Hendler, D. Nau, and B. Parsia. Automatic web services composition using shop2. In *ICAPS'03: International Conference on Automated Planning and Scheduling*, 2003.