Lawrence Berkeley National Laboratory

Lawrence Berkeley National Laboratory

Title

BReW: Blackbox Resource Selection for e-Science Workflows

Permalink

<https://escholarship.org/uc/item/1zp0v4cf>

Author

Simmhan, Yogesh

Publication Date

2010-11-09

DISCLAIMER

This document was prepared as an account of work sponsored by the United States Government. While this document is believed to contain correct information, neither the United States Government nor any agency thereof, nor the Regents of the University of California, nor any of their employees, makes any warranty, express or implied, or assumes any legal responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by its trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof, or the Regents of the University of California. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof or the Regents of the University of California.

BReW: Blackbox Resource Selection for e-Science Workflows

Yogesh Simmhan¹, Emad Soroush², Catharine van Ingen³, Deb Agarwal⁴ and Lavanya Ramakrishnan⁴

¹Computer Engineering Division, University of Southern California, Los Angeles CA

²Computer Science Department, University of Washington, Seattle WA

³Science Group, Microsoft Research, San Francisco CA

⁴Advanced Computing for Science Department, Lawrence Berkeley National Lab, Berkeley CA

simmhan@usc.edu, soroush@cs.washington.edu, vaningen@microsoft.com, {daagarwal, lramakrishnan}@lbl.gov

Abstract—Workflows are commonly used to model data intensive scientific analysis. As computational resource needs increase for eScience, emerging platforms like clouds present additional resource choices for scientists and policy makers. We introduce BReW, a tool enables users to make rapid, high-level platform selection for their workflows using limited workflow knowledge. This helps make informed decisions on whether to port a workflow to a new platform. Our analysis of synthetic and real eScience workflows shows that using just total runtime length, maximum task fanout, and total data used and produced by the workflow, BReW can provide platform predictions comparable to whitebox models with detailed workflow knowledge.

Keywords—*workflow; resource selection; planning; workflow migration; resource platforms; cloud; HPC*

I. INTRODUCTION

The need to manage and analyze the deluge of digital data from sensors and scientific instruments has resulted in the popularity of eScience workflows [1,2]. The tasks in these workflows are scientific applications or data processing scripts that pass the output of their computation to the next application in the logical dataflow. eScience workflows run on a spectrum of resource platforms today, from desktops, clusters, and high performance computing (HPC) resources to, more recently, cloud platforms. With the increasing scale of experiments, scientific user groups are interested in exploring large capacity platforms like HPC and cloud computing [3,13].

Porting an existing or evolving workflow onto a new resource platform and evaluating it is often a tedious exercise that requires interdisciplinary teams of domain scientists, computer researchers, and programmers. As a result, eScience workflow developers and users are interested in rapid, high-level analysis of workflows on novel platforms. Low overhead tools that let them compare and contrast various resource platforms prior to embarking on the costly exercise of workflow and application migration are opportune.

Current workflow planning and resource analysis tools enable careful planning of user workflows on diverse resource platforms. However, they are unsuitable for making high-level resource platform decisions for two reasons. First, these tools require a detailed knowledge of a workflow's structure, resource needs, and performance characteristics.

This information is often hard to predict, measure and maintain for a large number of constantly evolving eScience workflows. Second, these tools are often embedded into execution frameworks such as workflow engines and require porting some or all of the workflow into these environments.

In this paper, we introduce and describe **BReW** – **Blackbox Resource selection for e-Science Workflows**, which facilitates resource platform decisions with limited workflow information. BReW uses *blackbox* or coarse-grained knowledge of a workflow, such as workflow fanout width and runtime length dimensions, that are easily available, to estimate the workflow *makespan* on resource platforms with different resource availability [4]. Our current implementation of BReW is focused on comparing existing platform choices – desktop, cluster, HPC and cloud resources. However, our approach, methodology, and implementation are extensible to support other new and emerging hybrid resource platforms.

Specifically, we describe BReW and make the following contributions in this paper:

1. We develop the BReW blackbox platform selection model and contrast it with existing resource scheduling methods.
2. We evaluate and analyze our model for synthetic workflows with varying characteristics to understand the scope and limitations of our model.
3. We apply the blackbox model to four exemplar eScience workflows to gauge its practical effectiveness.

Our analysis of eScience workflows and synthetic workloads show that the BReW approach is feasible on a variety of platforms, including HPC and cloud, for diverse workflow sizes, and is limited by the quality of resource information available.

The rest of this paper is organized as follows. We discuss related work in Section II; provide an overview of eScience workflows and resource platforms in Section III; detail the BReW blackbox model in Section IV; describe our evaluation approach in Section V; present our analysis of BReW for synthetic workflows in Section VI; evaluate it for four eScience workflows in Section VII; and present our conclusions in Section VIII.

II. RELATED WORK

Science workflows are run today on desktops and grid environments. There have been a number of tools that manage execution in these varied environments and manage resource

Figure 1 & 2. Motif workflow attributes used for (1) whitebox (left) and (2) blackbox (right) approach to considering workflow structure.

selection decisions [15]. Workflow systems like Pegasus [5], Swift [6], and Trident [7] usually incorporate features to schedule tasks onto remote resources, such as Grids or clusters. For example, Swift uses Falcon execution framework to dispatch workflows tasks using multi-level scheduling [8]. This typifies fine-grained resource selection where detailed workflow structure and resource needs are known. DAG scheduling algorithms for Grids use heuristic models to schedule applications to meet time budgets [9]. The blackbox methodology is similar to these methods but finding an optimal schedule is not our goal. A number of groups have evaluated clouds for running scientific workflows [8,10,11].

However, these tools do not enable users to make high-level resource policy decisions. Our work considers various aspects of cloud computing and the feasibility of running in that environment given workflow characteristics and resource platform characteristics.

III. OVERVIEW

In current day eScience workflow scheduling techniques, detailed knowledge of each workflow task is needed, such as the execution time, input and output data in the Motif genomics workflow [12] shown in Figure 1. This workflow consists of a pre-processing step followed by a number of parallel runs of *interproscan* followed by a post-processing step that generates over one gigabyte of output data. However, due to the constantly evolving nature of scientific processes and uncertainties that often get resolved at runtime for dynamic workflows [21], this level of detail is typically not available for all scientific workflows. The overhead for instrumenting, collecting and maintaining fine-grained application profile information on diverse platforms is prohibitive for most but advanced users.

In contrast, we use a blackbox approach where coarse-grained, approximate dimensions of the workflow as shown in Figure 2, viz., length (runtime duration), width (number of CPU cores) and total data input and output, to make resource platform decisions. In our model, we forego knowledge of the detailed execution time of the tasks and the intermediate data products that might enable certain optimizations. Despite that, we are able to use just the critical pieces of

workflow information we have identified to generalize the resource needs and make informed resource platform selection choices.

A. Resource Platforms and their Selection

Common resource platforms available for scientists to run their applications include desktop workstations, local clusters, shared HPC resources and more recently, commercial clouds. In this paper, we use these as our candidate resource platforms for eScience workflows. However, our methodology is not bound to any specific platform and can be applied in general to other resources that might be available.

1) *Resource Platforms*: A large number of science applications today still run on *desktop workstations* whose multiple cores are suited for compute intensive and interactive visualizations. However, growth of data and the nature of analyses are far exceeding what is possible even on high-end workstations. Scientists often own and operate mid-sized *local clusters* (≤ 512 cores) within their research groups. These dedicated resources are often under-subscribed with low queue latency. Nevertheless, these are only suitable for small to mid-range computations that fit within the cluster's core capacity.

Scientific workflows also use *shared HPC resources* at academic and national supercomputing centers. These are typically accessible to multiple user groups through allocations and are often over-subscribed causing significant queue wait times. Users in this environment often have less control and are subject to site level policies and software changes. *Cloud computing* promises a greater degree of freedom to end-users while enabling resource scale-out comparable to shared HPC centers. On-demand access to massive cycles eliminates queue contention but virtualization can impact some applications.

2) *Platform Selection*: We use the term *platform selection* to denote the policy and process required to select a particular resource platform for an eScience workflow. Unlike workflow, task, or resource scheduling, our goal is not the optimization of the workflow on a particular or a range of resources. We consider platform selection as a high-level decision process that will be used by decision makers such as funding agencies and principal investigators to decide the choice of platforms for their scientific explorations. This information is also crucial when deciding whether to port an existing application from one platform to another, and analyze the cost-benefit trade-off. Thus, while there is an overlap of concepts, our platform selection approach is complementary to traditional scheduling methods. Once a platform is selected, users can use scheduling tools for actual workflow execution on the platform.

Platform suitability also depends on several qualitative factors such as availability of dependent operating system, packages, access policies, and so on. These issues are orthogonal to our approach but will still need to be considered.

In our evaluation, we use resource information from typical cloud and HPC resources available today. For

simplicity, we assume that all tasks of a workflow are run on the same platform. These resources were selected as a representative set of diverse platform characteristics. We demonstrate the selection of platforms for eScience and synthetic workflows in our comparative evaluation. Our goal is to show the effectiveness of our blackbox approach in comparison to traditional whitebox approach. A particular choice of platform is an artifact of the workflow and resource characteristics we use. Thus, the results should not be interpreted as one platform is better than the others for a certain or for all workflows.

B. Resource Platform and Workflow Attributes

Resource platforms have attributes that can be used to evaluate their suitability for running workflows with certain characteristics. Availability of both resource and workflow attributes enables matchmaking.

The *degree of parallelism* offered by a resource platform depends on the number of cores available for computation. For desktop and local clusters, this may be all available cores while for cloud and shared HPC, the bounds may be set by policy. The *core speed* can also impact the computation since cloud resources may be rated at a lower speed or run slower due to virtualization. *Computation latency* can be introduced through batch queues controlling access to shared HPC clusters or by VM startup times. *Network bandwidth* in and out of the resource platform from desktop determines data transfer time between client and remote compute resources. *Persistence* and *size* of available local storage determine if intermediate data must be moved between remote platforms and desktop. *Network latency* within the resource platform can affect communication costs of tightly coupled MPI tasks. We consider these resource attributes in our blackbox model to characterize resource behavior.

Workflows exhibit features and have requirements that can be used to determine the resource best suited to run part or all of the workflow. *Structural* features of a workflow characterize the data and control flow pattern. Common patterns are sequential pipeline, map-reduce (or fork-join) pattern, and iterations of these. The *width* of the structure (i.e. fanout of tasks), the *length* in terms of number of stages and their runtime, and the number of iterations determine resource selection [11]. *Resource usage* features of a workflow quantify the computational, data storage, and networking resources needed. The compute usage can be specified as time taken to run the stage on a specific core speed. The data and networking resources can be given in terms of input and output file sizes as well as characteristics such as access patterns. Memory size may also be key for some applications.

C. Resource Cost

Finally, *cost* of using the resource is a factor in resource choice. The costs of a desktop, local cluster or shared HPC resources are often partially hidden or amortized. Cloud costs are very visible. Shared HPC resources may also have quotas that limit user access. Users might need to tradeoff system performance for cost in their decisions. Cost models

for resource platforms is a separate issue and is outside the scope of this paper.

IV. THE BREW BLACKBOX MODEL

BReW has been developed to enable users to perform high-level platform selection. BReW works with the limited knowledge of the workload that the user provides and information it has about resource platforms. It is not a replacement for existing workflow technologies but a tool that scientists and decision makers will use to determine suitability of a resource platform for specific scientific workflows. The blackbox model used by BReW is discussed below.

A. Model Attributes

The blackbox resource selection uses four commonly known attributes for the entire workflow:

1) *Workflow Width*: The maximum fanout of the workflow tasks at any point in the workflow. This captures the maximum parallelism of the workflow and would be equal to one for a serial workflow.

2) *Workflow Length*: The total time to run the workflow computation at full parallelism assuming required resources are available (i.e. no latency). This is the execution time of the workflow and depicts the duration of resource needs.

3) *Data Sizes*: The initial workflow input and final workflow output *data sizes* that are transferred prior to and after a workflow executes on a particular resource platform.

4) *Concurrent Minimum Cores (mincore)*: Minimum number of concurrent cores needed for workflow execution. Captures constraints where part of the workflow, such as MPI tasks, needs at least a certain number of CPU cores. For purely loosely coupled tasks, this equals one core.

B. Blackbox Model

We use total workflow turnaround time as a metric to compare resource platforms. The workflow turnaround time denotes the duration between when the user submits the workflow for execution and the time when the last task of the workflow completes.

The blackbox model provides us an approximation of the resource needs that we use to calculate the workflow turnaround time as follows:

$$F_{\text{WorkflowTime}} = T_{\text{LatencyMax}} + T_{\text{DataSum}} + \frac{T_{\text{Length}} \times N_{\text{Width}}}{N_{\text{Cores}}}$$

Where

$T_{\text{LatencyMax}}$: Time to acquire resources for the workflow. This corresponds to batch queue wait times in HPC centers and virtual machine startup overheads on cloud machines.

T_{DataSum} : Time to transfer initial input and final output data between client desktop and the workflow execution platform;

T_{Length} : Workflow length time as defined above;

N_{Width} : Width/maximum fanout of the workflow;

N_{Cores} : # of available cores to run workflow.

C. Comparison with Traditional Whitebox Model

Whitebox model assumes that the workflow structure and all attributes for each workflow task are available. This is similar to workflow scheduling algorithms that is used to manage execution of a workflow on a diverse distributed set of resources. This means the data input, data output and the CPU time for each task is known before workflow's launch. Also, the fanout of each stage of the workflow is known from the structure. Given this relatively fine-grained detail, each workflow task can be scheduled independently. Each task incurs a latency time to access one CPU core, but needs the core only for the duration of that task. The workflow turnaround time for a workflow given the whitebox model is

$$F_{\text{WorkflowTime}} = \sum F_{\text{StageTime}}^i$$

Where

$F_{\text{StageTime}}^i$ is the time taken by workflow stage i , and is:

$$F_{\text{StageTime}}^i = T_{\text{Data}}^i + \text{Ceil}(N_{\text{TaskWidth}}^i / N_{\text{Cores}}^i) \times T_{\text{LatencyOne}}^i + (T_{\text{TaskLength}}^i \times N_{\text{TaskWidth}}^i) / N_{\text{Cores}}^i$$

Where:

T_{Data}^i : Time to transfer input, output data between desktop and the execution platform for the i^{th} stage;

$T_{\text{LatencyOne}}^i$: Latency time to start executing one task in the i^{th} stage on *one* core, due to queue wait or VM start-time;

$T_{\text{TaskLength}}^i$: Maximum task runtime among those workflow tasks scheduled concurrently in the i^{th} stage;

$N_{\text{TaskWidth}}^i$: Width/fanout of the number of tasks in i^{th} stage;

N_{Cores}^i : # of available cores to run tasks; $N_{\text{Cores}}^i \leq N_{\text{Tasks}}^i$.

V. ANALYSIS FRAMEWORK AND ASSUMPTIONS

The goal of our analysis is to evaluate the effectiveness and limitations of BReW for workflow resource selection when compared to the whitebox model. Effectiveness is measured by the ability to accurately discriminate between or rank the resource platforms based on the makespan estimated for a given workflow, rather than the precision of the absolute values estimated for the makespan.

BReW is implemented as a prototype decision making framework that takes the specification of a workflow and considers it against resource platforms it is configured for to provide the makespan estimate for the workflow on each platform. Our framework implements both the proposed blackbox model and the whitebox model. This allows us to compare the makespan estimates from both the models for the same workflow, with the blackbox model using only the workflow dimensions as input while the whitebox uses the complete workflow DAG details. The details of workflow specification used, resource platform configurations, and assumptions of our experiments are described here.

A. Resource Platform Configuration

We perform the analysis for three types of platforms, and four platform instances: Local Cluster, Azure Cloud, IU BigRed HPC cluster, and SDSC TeraGrid HPC Cluster. The local cluster is a hypothetical cluster configured with between 1—512 CPU cores rated at 2.5GHz connected on a 1Gbps LAN to the desktop client. While not explicitly considered, a 1-core local cluster is the equivalent of a workstation. For the shared HPC clusters and Azure, we use prior micro-benchmarks and public data to define their configuration. The Azure cloud has up to 2048 cores available rated at 1.6GHz with a 10Mbps network connection to the desktop client. While the Azure cloud may be much larger, this is representative of policy limits that may be enforced. The median VM startup latency is measured to be a linear function of number of cores required at $(200+20 \times \text{cores})$. The shared HPC clusters are modeled on the TeraGrid clusters at Indiana University (BigRed) and San Diego Supercomputing Center. They have similar configurations with 2048 available cores rated at 2.5GHz, a 100Mbps network bandwidth to the desktop client, and the queue latency time for job submission is provided by the TeraGrid Batch Queue Prediction Service (QBETS) [14,16]. We use a 50% quantile for these predictions. These platform configurations are provided as a text file input to BReW and can be easily updated.

B. BReW Implementation

The BReW tool provides blackbox makespan runtimes used to rank resource platforms. It uses the blackbox model by default but can also provide whitebox model predictions if provided the complete workflow DAG. Workflow inputs are provided as a DAG in case of whitebox estimates (Figure 1) or as the workflow dimensions in case of blackbox (Figure 2). The nodes in the whitebox workflow DAG are tasks that have task runtime, data input and output specified. In our implementation, tasks run on a single core by default, and tasks that can run concurrently are grouped into stages. Stages additionally have the minimum required cores (*mincore*) specified that captures tasks that might require simultaneous access to more than one core. BReW supports estimates for arbitrary whitebox DAGs, but we limit our analysis to workflows that are a linear pipeline of stages. For blackbox estimates, the workflow inputs are the task fanout width, runtime length, total data input and output, and optionally, the *mincore* (defaulting to workflow width), as shown in the bounding box in Figure 2. For convenience, the simulator is able to reduce a given whitebox DAG specification to the equivalent blackbox dimensions for the sake of comparison of the models. Both DAGs and blackbox dimensions are represented using a simple XML schema input.

In case of whitebox estimates, the provided DAG is traversed and the *makespan* function applied for the tasks and stages in the workflow for each resource platform. Queue latency for the HPC clusters is dynamically queried from the QBETS web service while static configuration values are used for the other platforms. The cumulative runtime for the stages is computed and emitted as the

makespan for the workflow on each platform. For blackbox, we apply the *makespan* function on the workflow dimensions for each platform to provide the *makespan* estimate. It too uses the QBETS service for HPC queue latency. In both models, we do not actually execute the workflow but just apply the *makespan* functions. The tool is implemented in C# and expected to be publicly available.

C. Assumptions

We make the following assumptions in BReW when estimating the makespan using the models, to make the problem tractable. In both models, we request for the smaller of available cores on the resource platform and the minimum number required cores for the workflow or stage for calculating the queue wait and VM startup latency times. The latency time is a function of the number of cores in Azure cloud. In HPC clusters, both number of cores and the duration they are requested for affect queue latency times.

In the BReW blackbox model, acquired cores are retained for the duration of the workflow, so the latency time appears only once per workflow in the blackbox model. In the whitebox model, the cores are retained for the duration of the task. The latency times appear once per task since they are independently scheduled. For concurrent tasks in a stage, their latency times also run concurrently, but across stages, the latency times accumulate. When more concurrent tasks are present than available cores, the latency times of later tasks interleave with runtime of prior tasks, thus causing the latency times to appear only once per stage even in this case.

For simplicity, all workflow tasks are assumed to run on the same platform. While in the blackbox model, we estimate data transfer between desktop workflow client and the resource platform only at the start and end of workflow execution, in whitebox model, input and output data are transferred to and from the desktop client for each workflow task.

VI. MODEL ANALYSIS ON WORKFLOW PARAMETER SPACE

In these analyses, we measure the sensitivity of BReW to different workflow characteristics to determine the scope of workflows for which the blackbox model shows a high degree of consistency in platform selection with the whitebox model. We use synthetic workflows using a workflow generator tool that produces whitebox DAGs and their blackbox dimensions based on provided parameter ranges that are fed to the BReW simulator. The parameter space that we cover for generating the synthetic workflows is shown in Table I.

TABLE I. SPACE OF SYNTHETIC WORKFLOWS CONSIDERED

WF Length (L)	20mins–60hrs
Stage Length	30secs – 6hrs
# of Stages	4, 10, 50, 100
WF Width (W)	1–1600
Mincores per Stage	1, 0.25×W, 0.50×W, 0.75×W, W

Figure 3. Heatmap showing consistency of platform prediction using blackbox estimates with the whitebox estimates with varying workflow length (X Axis) and width (Y Axis). The inconsistency in discriminating between the two HPC clusters (bright red) is due to different queue latency times estimated by the models. A large region at the upper extreme of width and length (maroon) has not data available from QBETS.

A. Effect of Length

In the first set of experiments, we evaluate the effect of increasing workflow length on the consistency of our blackbox model's platform prediction to the whitebox model. We estimate *makespan* using whitebox and blackbox models as we increase the workflow length from 20mins to 60hrs, and fanout from 1 to 1600 tasks. We use 10-stage workflows with uniform fanout, and *mincore* equal to fanout for both workflow and stages. The platforms are assumed to have sufficient cores available. The data transfer sizes are 10MB.

1) *Effect of Workflow Total Length*: Figure 3 shows a heatmap of the consistency of the BReW blackbox platform selection to the whitebox as total workflow length varies on the X Axis, and workflow width on Y axis. Each cell is one workflow, and its color indicates if the blackbox and whitebox make consistent platform choices. Given that this workflow has 10 stages, the length per stage at each cell is the workflow length/10. Blue indicates blackbox and whitebox are consistent, with specific shades referring to specific platforms selected by both models. Bright red is correct selection of HPC over Local Cluster and Azure, but incorrect discrimination between SDSC and IU BigRed. Maroon is absence of estimates due to lack of information from the QBETS service.

QBETS relies on historical data to make queue predictions. For workflow that are 200mins or longer with a

Figure 4. Plot of queue latency times predicted by QBETS for increasing number of requested core-minutes for SDSC and IU BigRed clusters. Plot shows crossover between SDSC and BigRed, where latter initially has lower latency but increases beyond 32 core-hours.

high fanout of 64 to 1600, it does not return any results for the blackbox model (maroon color) due to the extreme resource needs. Given this constraint of QBETS, the BReW blackbox model cannot be used for very long and wide workflows. At the very extreme corner of 60hrs and 1024/1600 width, we see that Azure is selected (dark blue) due to lack of information from QBETS for both blackbox and whitebox models.

At the lower end of the spectrum, for very short workflow lengths of 20 to 300mins using fanout of 8 tasks or less, we see that both blackbox and whitebox select HPC platforms over Azure, but the blackbox is unable to discriminate between SDSC and IU BigRed. This is because of two reasons: (1) the difference in SDSC and BigRed queue times for small jobs below 32 Core-Minutes, where BigRed initially has smaller queue times than SDSC, but returns larger queue times beyond that (Figure 4), and (2) the difference in scheduling approaches for blackbox and whitebox models –blackbox requests HPC resources to fit the entire workflow while whitebox requests resources for a stage at a time, since $mincore=fanout$. For workflows larger than 32 core-mins but the core-time per stage smaller than 32 core-minutes, the blackbox chooses SDSC while whitebox chooses BigRed. When the core-time per stage is greater than 32 core-mins, both models choose SDSC (Figure 3, light blue). At the lower extreme, the workflow size itself is less than 32 core-mins, so both models choose BigRed accurately.

For the rest of the region in light blue, we see that the BReW blackbox model is able to accurately track the whitebox selection for the workflows considered. In the space of synthetic workflows considered for this experiment, none of the estimations were incorrect across HPC and Azure.

2) *Effect of Length per Workflow Stage:* Blackbox and whitebox models use different granularity (workflow vs. stage/task) for acquiring resources, and hence the length of each stage has an impact on the resource prediction.

We compare the effect of the length per workflow stage on the degree of consistency between blackbox and whitebox models, measured as the % of times the blackbox model makes the same prediction as the whitebox for a set of

Figure 5. Prediction consistency % between blackbox and whitebox models for 162 workflows as length per workflow stage increases. Inconsistency of 20% is introduced in small stage lengths <20mins due to incorrect queue latency estimation for the two HPCs. For stage lengths > 30mins, QBETS does not provide sufficient data for making estimates.

workflows. We consider 162 workflows with length of each stage varying from 2mins to 6hrs, uniform fanouts ranging from 1 – 1600 tasks each, and with constant 10 stages, all using $mincore=fanout$ for the workflow and stages. The effect of varying stage length on the consistency % between blackbox and whitebox models is plotted in Figure.

Figure 5 shows that for small workflow stage lengths less than 20mins, we are able to give a Blackbox prediction consistent with whitebox at least 80% of the time (green). The remaining 20% of inconsistency (red) is due to inability to discriminate between the two HPC clusters as discussed before. For 20min stages and longer, we see the introduction of errors due to lack of data from QBETS for the BReW blackbox. This region grows larger as the length per stage increases from 20mins to 6hrs, with consistent predictions decreasing from 80% to 55%.

When we vary the number of stages above and below the 10 stages considered here (not shown due to lack of space), we observe the shape of the plot and the green/red/gray regions to remain the same but their areas change. The peak of the consistent % (green) is observed at workflow lengths of 100mins: 95% for 4 stages/25min per stage, 90% for 10 stage/10min per stage, and 78% for 50 stages/2mins per stage.

3) *Effect of Number of Stages in Workflow:* Since BReW blackbox and whitebox models schedule at different granularity (workflow vs. stage/task), the number of stages has an impact on the prediction.

This study is similar to the previous one, except that we vary the number of stages from 1 to 100 while keeping their overall length constant, for workflows with lengths and fanouts that range from 20mins – 60hrs and 1 – 1600 tasks. We plot the % of consistent blackbox and whitebox resource predictions grouped by stages for a set of 162×5 workflows with varying workflow length, fanout, and stages.

Figure 6 shows that as the number of stages increases, the ability of the blackbox model to discriminate between the two HPC clusters reduces (red). The percentage of HPC discrimination inconsistency increases linearly with number

Figure 6. Prediction consistency % between blackbox and whitebox models for 810 workflows. HPC discrimination inconsistencies increase with number of stages as it accumulates at each stage. QBETS inconsistencies remain static since they were observed for blackbox model which is unaffected by number of stages.

of stages. This is the result of the earlier described queue latency difference between the HPC clusters for small stage lengths, with its effect being compounded at each stage. As the number of stages increase for a constant length workflow, the length of the stage decreases, thus pushing them into this error space, and these errors also accumulate at each workflow stage.

The inconsistency due to lack of QBETS information is almost constant across workflow stages. Since it was mainly observed for queue latencies requested by the blackbox model, it has no impact in these experiments since the blackbox model does not use the number of stages for its estimates. For a workflow with just one stage, there is no error since the blackbox and whitebox models have identical information about the workflow available to them.

B. Effect of Workflow Width

The effect of total workflow width was discussed as part of discussion on workflow length (Sec VI.A.1). To recap, as the workflow width goes beyond 64 tasks for large values of workflow lengths of 200mins to 60hrs, we are unable to get results from QBETS (maroon in Figure 3). For small values of fanout below 32 core-hours, the Blackbox model is unable to discriminate between the two HPCs.

1) *Varying width of stages according to a probability distribution:* We use a normal distribution to generate the width of each stage in the workflow. All stages have the same mean width (μ), but we increase the standard deviations (σ) as a fraction of the mean width. As the σ increases, the width of stages within a workflow shows more variability.

In a normal distribution, 99% of randomly drawn values lie within 3σ of the mean. For example, with a mean workflow stage fanout of $\mu=20$ and using $\sigma=0.25\mu$, the width of a stage in the workflow lies between 5 and 35 with 99% probability.

In this experiment, for workflows with lengths and fanouts that range from 20mins – 60hrs, 1 – 1600 tasks, we increase the fanout σ from 0 to μ , and plot the consistency

Figure 7. Prediction consistency % between blackbox and whitebox models (primary Y Axis) with increasing workflow stage with variation. Minimal impact is seen on consistency % with increasing width variation within a workflow. However, resource utilization % (secondary Y axis) using blackbox model reduces with width variation since cores remain idle for narrower stages in the workflow compared to its widest.

between blackbox and whitebox model resource predictions on the primary Y Axis. Since the BReW blackbox model overestimates the core-hours required by the workflow, we also plot the actual utilization % of core-hours by blackbox against the acquired core-hours in the secondary Y Axis.

We observe that with increasing workflow stage width variability, the impact on the platform prediction consistency between the two models is small. We see that the consistency % changes from 75% for no width variation to 65% for $\sigma=\mu$. When using a large static variability ($\sigma=64$, plot not shown) we see more errors due to missing HPC prediction values even at smaller fanouts for 50 or more stages.

Despite consistent platform prediction with variability in width, the utilization of resources in the blackbox model decreases as the variability increases (Figure 7, secondary Y axis). Resource utilization is ratio of resources (core-hours) actually used as against the resources that blackbox model estimates it will use. However, the blackbox model is just used for selection of resource platform and not for resource scheduling at runtime. So the *actual* resource utilization is only be affected by the scheduling model at execution.

C. Effect of Min-Cores

We vary the *mincores* required by each stage in the whitebox model between 1 (pure loosely coupled), a fraction of fanout in a stage, and the max-fanout for a stage (pure tightly coupled). We predict the resource platform consistency % for synthetic workflows with workflow lengths from 20mins – 60hrs, workflow widths from 1 – 1600 tasks, and with 4, 10 and 50 stages. For the BReW blackbox model, we use a constant *mincore* of workflow width (tightly coupled workflow), shown in Figures 8.

In Figure 8, we observe that when whitebox uses *mincore*=1 and blackbox uses *mincore*=workflow width, the fraction of times that they are consistent is small at 30% (green). 50% of the inconsistency is due to lack of discrimination between the HPC clusters (red) and 20% due

Figure 8. Prediction consistency between blackbox and whitebox models as the mincores for each workflow stage in whitebox DAG varies from 1 core to fractions of stage width (W). Blackbox uses constant mincore=workflow width.

to lack of QBETS data. This is because with $mincore=1$ in whitebox, the HPC clusters are requested one core at a time causing the core-hours per task to be small – within the 32 core-hour region of error seen in Figure 4. As the whitebox $mincore$ increases as a fraction of stage width, the region of HPC disagreement monotonically decreases with a concomitant increase in consistency to reach 75% consistency at $mincore$ =stage width. The fraction of time that we do not get a result from QBETS is uniformly constant (gray), caused by the extreme values of workflow width and length used by the BReW blackbox model with $mincore$ =workflow width.

When the same experiment is done using $mincore=1$ (not shown for brevity) for the blackbox model, we see a similar HPC discrimination inconsistency with whitebox $mincore=1$, but this diminishes sharply for any higher fractions of whitebox $mincore$. As all tasks are loosely coupled in both models, we do not hit extremities where QBETS has no data.

VII. MODEL ANALYSIS FOR eSCIENCE WORKFLOWS

In this section, we use BReW to make resource platform selections for five different eScience workflows with features described in Table II.

TABLE II. CHARACTERISTICS OF eSCIENCE WORKFLOWS ANALYZED

Name	Domain	Length (h:m:s)	Width	Stages	Data In/Out
Montage	Astronomy	0:06:10	662	9	700M/1.5M
MODIS	Environ Sci.	0:29:40	60,000	4	400G/1G
Motif	Genomics	1:31:30	135	3	13M/1.2G
GWAS	Comp. Bio.	0:19:00	1100	7	150M/10M

We compare the blackbox platform prediction against whitebox using the four platforms used before: Local cluster, SDSC HPC, IU BigRed HPC and Azure Cloud, and in addition impose constraints on the number of cores available in each platform to mimic real world policy decisions that BReW is intended for. In all cases, we use the blackbox

Figure 9. Estimated runtime for MOTIF with whitebox & blackbox models with increasing number of available cores on the 4 platforms. Platform selection is consistent across all core availabilities.

$mincore$ as the workflow width while the $mincore$ for whitebox is 1 in all workflows unless noted otherwise. In the evaluation we present below, the intended goal is for BReW to make the same platform choice as the whitebox model with full workflow DAG knowledge does.

A. Simple reference: Motif Network

The Motif workflow is a long running workflow (shown in Figure 1) with just three stages and is not very wide at 135 tasks [12, 17]. Figure 9 plots the *makespan* estimates of this workflow using the whitebox (Fig.9a) and BReW blackbox (Fig.9b) models on the Y axis for 1 to 512 available cores. The platform ordering by the models is examined.

We see that the ordering of resource platforms by BReW is consistent with the whitebox for different core availabilities considered of the four platforms. The absolute values predicted in both cases are also similar with the most absolute error exhibited by IU BigRed at 50%. However, even this variation is small enough that we can compare the platforms with not just the same number of cores but even different cores across platforms. For e.g. Azure cloud with 128 cores has a shorter makespan than cluster with 64 cores in both cases.

B. Short Tasks: Montage Astronomy Image Processing

Montage is a popular image processing workflow for astronomy datasets [18]. It is characterized by seven short stages between 11 – 100 seconds long, three of which have

Figure 10. Estimated runtime for Montage with whitebox & blackbox models with increasing number of available cores on the 4 platforms. Platform selection is consistent across Local cluster, HPC and Cloud but not within the 2 HPC platforms.

fanouts between 166 – 662 tasks. Whitebox and BReW blackbox *makespan* estimates are shown in Figure 10 for 1 to 2048 available cores on each platform.

We see that the ordering across Cloud, HPC and Cluster platforms are consistent across both models across different core availabilities. Beyond 128 cores, we see the Azure Cloud runtime increasing for BReW since the sequential VM startup latency time outstrips the application runtime gains. The whitebox model starts VMs independently in parallel since the application is loosely coupled ($mincore=1$), giving it constant latency times.

However, BReW fails to discriminate between IU BigRed and SDSC HPC platforms. The cause for this is the difference in latency times of the HPC systems when the workflows have short stages, as discussed in Section VI.A.2. Given that this is a loosely coupled workflow ($mincore=1$) with short tasks (<2mins), it falls in the far left of the X axis in Figure 4 at ~2 core-mins in the whitebox model, where SDSC latencies are longer than BigRed. However, our blackbox model treats this as a workflow requiring 4000 core-mins (662 width, 6:10mins length) and it falls at the far right of the X Axis in Figure 4, where IU BigRed latencies are longer than SDSC.

C. Data intensive: MODIS Environmental Data Subsetting

The MODIS satellite data reduction pipeline is a 4-stage, highly data parallel and data intensive workflow with stage widths of 1000 to 60,000 loosely coupled tasks used for

Figure 11. Estimated runtime for MODIS with whitebox & BReW models with increasing number of available cores.

processing 1-years' worth of data [19]. Figure 11 shows its whitebox and BReW runtime estimates for 1—2048 cores.

While the data transfers were marginal in earlier workflows, the input and intermediate data are large in this workflow. We see that BReW predicts the platform selection ordering consistent with the whitebox model. Our blackbox underestimates the data transfer times since it is not aware of the intermediate data transfers, but overestimates the computation compensating for each other.

D. Compute Intensive: GWAS

The Genome Wide Association Study (GWAS) workflow [20] implements computational and statistical techniques to analyze relationships between gene markers and features seen in subjects. The compute intensive GWAS workflow consists of two fork-join stages of fanouts 1100 and 150 that take about 10mins each. Figure 12 shows the whitebox and blackbox estimates for GWAS on the four platforms.

The behavior of GWAS is similar to Montage: the blackbox model tracks the whitebox model for selection across cluster, cloud and HPC but fails to order SDSC and HPC due to the short runtimes of the stages. In addition, we see that BigRed estimates for Blackbox hits another crossover point at 1024 cores that causes its latency times to increase further and obviate application runtime gains from the additional cores. Azure hits a similar transition point at 256 cores.

Figure 12. Estimated Runtime for GWAS from whitebox & blackbox models with increasing number of available cores

VIII. CONCLUSION

BReW introduces a novel, model for making high-level resource platform selection decisions that is crucial for domain scientists and policy makers as new resource platforms like clouds emerge. BReW is distinct from traditional workflow execution scheduling, and it uses limited workflow knowledge to make platform selections with low user overhead. Our detailed analysis using synthetic workflows has shown that BReW is as effective as the more complex, whitebox model for a large class of workflows. The limited inconsistencies seen were more due to inability to discriminate between the HPC clusters and lack of QBETS data. Our evaluation with real eScience workflows from diverse domains also shows that the blackbox method makes accurate platform selections.

IX. ACKNOWLEDGEMENTS

This work was supported by the Director, Office of Science, of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231.

X. REFERENCES

- [1] The Fourth Paradigm: Data-Intensive Scientific Discovery, Tony Hey, Stewart Tansley, Kristin Tolle (Eds.), Microsoft Research, 2009.
- [2] Workflows for e-Science: Scientific Workflows for Grids, Taylor, I.J.; Deelman, E.; Gannon, D.B.; Shields, M. (Eds.), Springer, 2007.
- [3] Scientific workflows and clouds, Gideon Juve and Ewa Deelman, *ACM Crossroads*, 16(3), 2010.
- [4] Comparison of Resource Platform Selection Approaches for Scientific Workflows, Yogesh Simmhan and Lavanya Ramakrishnan, in *Workshop on Scientific Cloud Computing (ScienceCloud)*, 2010.
- [5] Pegasus: Mapping Scientific Workflows onto the Grid, E Deelman, J Blythe, Y Gil, C Kesselman, G Mehta, S Patil, M Su, K Vahi and M Livny, in *Grid Computing*, 2004.
- [6] Swift: Fast, Reliable, Loosely Coupled Parallel Computation, Y. Zhao, M. Hategan, B. Clifford, I. Foster, G. von Laszewski, V. Nefedova, I. Raicu, T. Stef-Praun, and M. Wilde, in *SERVICES*, 2007.
- [7] Building the Trident Scientific Workflow Workbench for Data Management in the Cloud, Yogesh Simmhan, Roger Barga, Catharine van Ingen, Ed Lazowska, and Alex Szalay, in *Advanced Engineering Computing and Applications in Sciences (ADVCOMP)*, 2009.
- [8] Falcon: A fast and light-weight task execution framework, Ioan Raicu, Yong Zhao, Catalin Dumitrescu, Ian Foster, and Mike Wilde in *SC* 2007.
- [9] Scheduling Workflows with Budget Constraints, R. Sakellariou, H. Zhao, E. Tsiakkouri and M. Dikaiakos, in *Integrated Research in GRID Computing*, Springer, 2007.[10] Bioclouds, *Nature Methods*, vol. 7, 2010.
- [11] Seeking Supernovae in the Clouds: A Performance Study, K. Jackson, L. Ramakrishnan, R. Thomas and K. Runge, in *ScienceCloud*, 2010.
- [12] MotifNetwork: Genome-Wide Domain Analysis using Grid-enabled Workflows, J. L. Tilson, et al., in *BIBE*, 2007.
- [13] An Autonomic Approach to Integrated HPC Grid and Cloud Usage, H. Kim, Y. el-Khamra, S. Jha, M. Parashar, in *IEEE eScience* 2009.
- [14] QBETS: queue bounds estimation from time series, Nurmi, D. C., Brevik, J., and Wolski, R., *SIGMETRICS Perform. Eval. Rev.*, 35(1), 2007.
- [15] Forecasting Duration Intervals of Scientific Workflow Activities Based on Time-Series Patterns, Liu, X., Chen, J., Liu, K., and Yang, Y., in *IEEE eScience* 2008.
- [16] <https://portal.teragrid.org/hpc-queue-prediction>
- [17] A Survey of Distributed Workflow Characteristics and Resource Requirements, L. Ramakrishnan and D. Gannon, *Technical Report TR671*, Indiana University, 2008.
- [18] Characterization of Scientific Workflows, S. Bharathi, A. Chervenak, E. Deelman, G. Mehta and M. Su, in *Workflows in Support of Large Scale Science (WORKS)*, 2008.
- [19] eScience in the Cloud: A MODIS Satellite Data Reprojection and Reduction Pipeline in the Windows Azure Platform, Jie Li, Deb Agarwal, Marty Humphrey, Catharine van Ingen, Keith Jackson, and Youngryel Ryu, in *IPDPS*, 2010.
- [20] Correction for Hidden Confounders in the Genetic Analysis of Gene Expression, J. Listgarten, C. Kadie, E. Schadt and D. Heckerman, *PNAS* (In press).
- [21] Chapter: Dynamic, Adaptive Workflows for Mesoscale Meteorology, Dennis Gannon, Beth Plale, Suresh Marru, Gopi Kandaswamy, Yogesh Simmhan and Satoshi Shirasuna, *Workflows for e-Science: Scientific Workflows for Grids*, Taylor, I.J.; Deelman, E.; Gannon, D.B.; Shields, M. (Eds.), Springer, 2007.