Comparison of Resource Platform Selection Approaches for Scientific Workflows

Yogesh Simmhan
Microsoft Research, Los Angeles CA
yoges@microsoft.com

Lavanya Ramakrishnan
Lawrence Berkeley National Lab, Berkeley CA
LRamakrishnan@lbl.gov

ABSTRACT

Cloud computing is increasingly considered as an additional computational resource platform for scientific workflows. The cloud offers opportunity to scale-out applications from desktops and local cluster resources. Each platform has different properties (e.g., queue wait times in high performance systems, virtual machine startup overhead in clouds) and characteristics (e.g., custom environments in cloud) that makes choosing from these diverse resource platforms for a workflow execution a challenge for scientists. Scientists are often faced with deciding resource platform selection trade-offs with limited information on the actual workflows. While many workflow planning methods have explored resource selection or task scheduling, these methods often require fine-scale characterization of the workflow that is onerous for a scientist. In this paper, we describe our early exploratory work in using *blackbox* characteristics for a cost-benefit analysis of using different resource platforms. In our blackbox method, we use only limited high-level information on the workflow *length, width,* and *data sizes*. The length and width are indicative of the workflow duration and parallelism. We compare the effectiveness of this approach to other resource selection models using two exemplar scientific workflows on desktop, local cluster, HPC center, and cloud platforms. Early results suggest that the blackbox model often makes the same resource selections as a more fine-grained whitebox model. We believe the simplicity of the blackbox model can help inform a scientist on the applicability of a new resource platform, such as cloud resources, even before porting an existing workflow.

Categories and Subject Descriptors

C.2.4 [Distributed Systems]: *Distributed applications*

General Terms

Algorithms, Performance. Experimentation

Keywords

Workflow Patterns, Workflow Structure, Cloud Computing, High Performance Computing, Resource Management, eScience

1. INTRODUCTION

Scientific experiments and applications are often composed of a number of tasks with diverse computation and data needs. The tasks form a dataflow pipeline between logical stages that are represented and executed as workflows or scripts. Each task may have loosely coupled, asynchronous interactions or be a tightly bound MPI application, among others. Many of these workflows are beginning to exceed the available capacity on their preferred resource platform and continuously exploring other options.

Cloud computing has recently gained popularity as a resource platform for on-demand, high-availability, and high-scalability access using a pay-as-you-go model. Web applications have benefited from this paradigm leading to reduction or even elimination of computing and storage infrastructure investments. Scientists are beginning to explore the feasibility of the virtualized cloud resource model for running their computation and data analysis at small and large scales [3].

In addition to commercial cloud offerings, scientists can access several execution resource platforms for running their workflows including local workstations, small clusters owned by research groups, and shared supercomputing resources at national labs. Each platform presents trade-offs in terms of performance, policy, and cost with significant differences among them.

Resource selection for scientific workflows is frequently *ad hoc* and involves offline decisions by users. A user may choose to run applications based on familiarity with an environment or thumb-rules based on earlier experiences. Given the dynamic nature of the resources, both in the short and long terms, such improvised scheduling is often sub-optimal and occasionally punitive.

Several models are in use for *efficient* scheduling of workflows in heterogeneous resource environments [9,10,11]. Efficiency may be defined as any combination of reducing total wallclock time for completing the workflow, improving resource usage, or minimizing monetary cost. These models vary in complexity and accuracy (though the two are not strictly correlated). The scheduling schemes also vary in the degree of *a priori* knowledge about the workflow used for resource selection, ranging from purely structural information of the workflow to knowing fine-grained details of each workflow task. Most workflow scheduling approaches use fine-grained data today.

Detailed knowledge of workflow characteristics may not be available before the workflow is run. Such knowledge entails user overhead for collection and description. Scientists often need an approximate estimate of a resource platform's suitability for their workflow by just providing high-level information about it.

In this paper, we present a blackbox model for resource selection using limited knowledge of workflow characteristics. Our approach is based on the idea that the workflow structure and/or its dominant resource requirement stage are sufficient to evaluate the trade-offs associated with each resource platform. This approach is less studied in literature. We use (a) the workflow's dimensions – *length* and *width* – that signify its potential duration and fanout, and (b) the input and output *data sizes* to the workflow. We describe our early exploratory work into this high-level model that allows users to provide just these three characteristics of their application in order to understand its performance trade-offs on different platforms. We test our hypothesis using previously collected experimental data of two eScience applications and compare the efficiency of our *blackbox* prediction with more fine-grained *whitebox* and *graybox* workflow resource selection models. Our early results show the benefits of using limited information to make a usable estimate of the application throughput on different platforms.

Specifically, we investigate the following questions:

- Is it possible to make intelligent selection of resource platform from several available options using only the workflow dimensions and data input/output sizes?
- What are the trade-offs of running applications on different resource platforms?

The rest of this paper is organized as follows. We provide an overview of common resource platforms and workflow characteristics that guide resource selection in Section 2. We introduce our blackbox model and describe it in the context of whitebox and graybox models in Section 3. We present a comparative evaluation of the blackbox model accuracy for two genomics applications in Section 4. Sections 5 and 6 describe related work and our conclusions and future work.

2. OVERVIEW

Resource decisions need a clear understanding of the workflow characteristics and the characteristics and properties of available resource platforms. Scientific workflows have diverse characteristics and have a range of resource platform choices. In this section, we summarize these characteristics.

2.1 Workflow Characteristics

Workflows exhibit features and have requirements that can be used to determine the resource best suited to run part or all of the workflow stages among those available.

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. Besides the pattern itself, 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 influence resource selection decisions [11].

Resource usage features of a workflow quantify the computational, data storage, and networking resources. The compute usage can be specified as time taken to run the stage on a specific core speed. The data requirement is specified in terms of input and output file sizes and access patterns characteristics. Some data intensive applications may also need large memory.

2.2 Resource Platforms

Common resource platforms available for scientists to run their applications include desktop workstations, local clusters, shared HPC resources and more recently, commercial clouds.

Desktop Workstations: A large number of science applications today still run on the desktop. The workstation allows users complete control over the software environment as well as on data privacy. Multi-core machines can now match small clusters in their compute power. However, growth of data and the nature of analysis are exceeding what is possible even on high-end work-stations. The interactive scientific processes can necessitate transfer of final data to the desktop for visualization or validation.

Local Clusters: Scientists often own and operate mid-sized local clusters ($\leq$ 256 cores) within their research groups. Graduate students and research staff manage these environments in-house. The local cluster is a useful resource platform for groups that can afford the infrastructure and management cost. The captive nature of these resources often makes them under-subscribed and users can get immediate access for their applications. The cluster is often located on a LAN making large data transfers from desktop fast. Nevertheless, these are only suitable for small to mid-range computations that fit within the cluster's core size.

HPC Shared Resources: Scientific workflows also use shared resources at academic and national supercomputing centers. These resources are typically accessible to multiple user groups through one-to-many or peer-to-peer allocations and are often over-subscribed [16]. While users can access a larger resource pool, they incur queue wait times that depend on the system load when they run their computations. HPC users often have less control and are subject to site level policies and software changes. While some users may be on a fast research network to these centers, this is not universal. WAN bandwidth can limit large data transfers.

Cloud Resources: Cloud computing promises a greater degree of freedom to end-users enabling customized and user-controlled software environments while enabling resource scale-out comparable to shared HPC centers. However, virtualization can impact the performance of some scientific applications and overheads like Virtual Machine (VM) start time and (comparatively) low network bandwidth from desktop to cloud can impact application runtime.

The number of *concurrent resources* available on a platform depends on the number of cores available for computation. For desktop and local clusters, users can use all the cores available 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. Throughput can be impacted through *overheads* in batch queues on 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 can decide if intermediate data is moved out of remote platforms to desktop. *Network latency* within the resource platform can affect communication costs of tightly coupled MPI tasks.

3. PLATFORM SELECTION APPROACHES

Workflows are often orchestrated by a workflow engine on a client machine with the actual tasks of the workflow running on local or remote resources. The initial input and final output of the workflow is present in the desktop client. The parallel nature of the workflow may allow multiple tasks to be run concurrently; we term all tasks that can run concurrently as a *stage* in the workflow.

Resource platform selection models help decide which among available resource platforms are best suited to run the workflow in order to optimize for one or more factors. The models use an optimization function or heuristic based on workflow and resource attributes. In this paper, we limit our optimization factor to the makespan (or total runtime) of the workflow. For simplicity, we assume that all tasks of a workflow are run on the same platform.

We classify three workflow selection models based on the degree of detail required to characterize the workflow to choose a suitable resource platform to run it. Figure 1 illustrates the three models for the Motif workflow introduced in the next section.

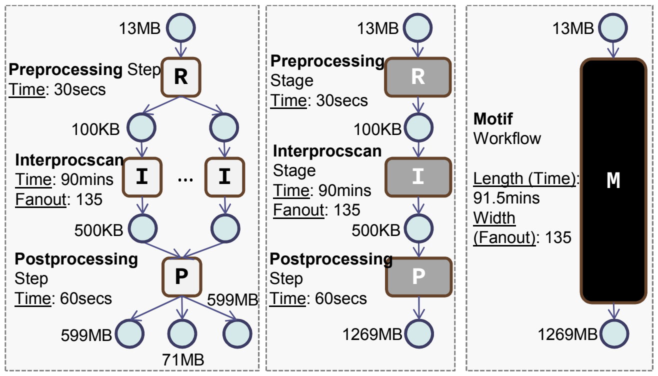

Figure 1. Motif workflow attributes used for whitebox (left), graybox (center), and blackbox (right) workflow scheduling. Each model requires less workflow characterization by the user

- **Whitebox (or Fine-grained) Selection**: This model assumes that the workflow structure and all attributes for each workflow task are available. 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 know from the structure. Given this relatively fine-grained detail, each workflow task is scheduled independently and, potentially, on a different resource platform. Each task incurs an overhead to access one CPU core. The overhead for waiting tasks and execution time for concurrent tasks in a stage overlap, but overhead is incurred for every stage as a task returns its core after use. The time-optimization function for this model estimates the total workflow time as:

$$F_{WorkflowTime} = \Sigma\ F^{i}_{StageTime}$$

where $F^{i}_{StageTime}$ is the time taken by workflow stage *i* given by:

$$F^{i}_{StageTime} = T^{i}_{OverheadOne} + T^{i}_{Data} + (T^{i}_{TaskLength} \times N^{i}_{TaskWidth})/N^{i}_{Cores}$$

Where:

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

$T^{i}_{OverheadOne}$: Overhead time to start executing one task in the *i*th stage on *one* core, due to queue wait- or VM instance start- time;

$T^{i}_{TaskLength}$: Maximum task runtime among those workflow tasks scheduled concurrently in the *i*th stage;

$N^{i}_{TaskWidth}$: Width or fanout of the number of tasks in *i*th stage;

N^{i}_{Cores} : Number of available cores for running the tasks, such that $N^{i}_{Cores} \le N^{i}_{Tasks}$.

- **Graybox (or Hybrid) Selection**: This assumes each stage of the workflow is opaque with only the *stage width*, *stage length*, and *total data* transferred into and out of each stage known. The width is given by the number of parallel tasks within the stage; the *length*, given by the total runtime for the stage when run fully parallel. The tasks within each workflow stage are not known. This model acquires the maximum possible number of CPU cores up to its width before it runs. It can also look ahead to acquire and retain cores for subsequent stages until the workflow completes. The time-optimization function for this model estimates the total workflow runtime as the sum of all *i* workflow stage runtimes plus the overhead time for acquiring the largest number of cores required for concurrent tasks from among all workflow stages:

$$F_{WorkflowTime} = T_{OverheadMax} + \Sigma\ F^{i}_{StageTime}$$

Where:

$T_{OverheadMax}$: Overhead time, due to queue wait or VM instance start time, to acquire maximum possible cores for all stages; and

$$F^{i}_{StageTime} = T^{i}_{Data} + (T^{i}_{TaskLength} \times N^{i}_{TaskWidth})/N^{i}_{Cores}$$

- **Coarse-grained (or Blackbox) Selection**: The blackbox resource selection uses just three commonly known attributes of the workflow: (1) the maximum fanout of the workflow at any point, which we term as the *workflow width*, (2) the total time to run the workflow computation at full parallelism, which we term the *workflow length*, and (3) knowledge of the initial workflow input and final workflow output *data sizes*. Using this approximation, we can reduce the workflow to a blackbox with just one stage and use a function similar to the graybox model above. Its time-optimization function estimates the total workflow runtime as:

$$F_{WorkflowTime} = T_{OverheadMax} + T_{Data} + (T_{Length} \times N_{Width})/N_{Cores}$$

where:

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

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

N_{Width} : Width or maximum fanout of the workflow.

4. EARLY EVALUATION

We compare our whitebox, graybox, and blackbox models for two eScience applications across desktop, local cluster, shared HPC center and cloud resource platforms. We evaluate the relative efficiency of the blackbox model for resource platform selection.

4.1 eScience Workloads

In this paper, we select eScience applications that have loosely coupled Map-Reduce structures since these are well suited to run in cloud environments.

- **MOTIF Networks**: Motif Networks can model gene regulation dependencies that control protein synthesis and behavior [2]. The MotifNetwork project analyses genome-sized networks of sequences that are computationally intensive. A typical Motif workflow has a Map-Reduce stage and two single-task sequential stages (Fig. 1). A pre-processing task splits a 13MB input file into 135 chunks that are individually operated upon by loosely-coupled, compute intensive *Interprocscan* tasks that take 90mins to execute in parallel and produce 500KB of output. A post-processing task gathers the outputs and generates a 71MB file, and two 599MB files. The workflow's *width* is 135, *length* 90mins, *data input* 13MB and *data output* 1269MB [7].
- **Genome Wide Association Studies (GWAS)**: GWAS uses computationally costly statistical algorithms to infer which genetic markers are associated with a particular phenotype or disease of interest [1]. The Linear Mixed Model GWAS workflow consists of two parallelized, compute-intensive Map-Reduce stages and four single-task stages. The *ML stage* calculates the maximum likelihood over input genes with a parallel fanout of 1100 tasks and takes 10mins in parallel. The subsequent *expectation-maximization stage* (EM) with a fanout of 150 improves the ML estimate and takes 9mins to complete in parallel. The input data size to the workflow is ~150MB for 40K genes for 200 subjects. The final EM stage produces a 10MB association matrix as result. So the workflow *width* is 1100, *length* is 19mins, and *data in* and *out* are 150MB and 10Mb respectively.

4.2 Resource Platforms

We use the following resource platform specifications based on earlier measurements for our evaluation:

- **Local Workstation**: We consider a single core workstation with CPU rated at 2.5GHz. All input and output workflow data are on local disk and tasks execute sequentially on local core.
- **Local Cluster**: We consider clusters of sizes between 1 to 256 cores that are connected by Gigabit Ethernet (128MB/s) to the scientist's desktop client. Each node has an identical CPU Core to the above workstation. The desktop contains the inputs to the workflow and all cluster outputs are transferred back to desktop.
- **HPC Cluster**: We consider the TeraGrid shared clusters at Indiana University (BigRed) and at SDSC with queue wait times at the 95% quantile predicted using the Batch Queue Prediction Service [5]. Cores are limited to between 1 and 2048 cores for concurrent use to simulate user quota policy. For simplicity, we assume each node has an identical CPU to the above workstation. The bandwidth between the HPC center and user's desktop client holding inputs and outputs is assumed to be 1.2MB/sec.
- **Cloud**: We consider Microsoft's Azure cloud with small VM CPU cores rated at 1.6GHz. The VM start times are measured at 200sec initial overhead and 20secs per additional VM [4] and users are assumed to get between 1 and 2048 VMs at a time. The bandwidth between Azure Cloud storage and user's desktop client holding inputs and outputs is measured at 1.2MB/sec.

4.3 Results

We calculate the total workflow runtime for GWAS and Motif workloads on each of the four resource platforms using the time-optimization functions for the three resource selection models described earlier. We vary the number of available cores in local cluster, shared HPC, and cloud in our calculations.

Figures 2(a–c) show the estimated Motif workflow runtimes (Y Axis) as a function of the available number of concurrent cores (X Axis). The pairwise percentage difference in time estimates between blackbox and whitebox models, and between blackbox and graybox models are shown in Figures 3(a) and 3(b).

The Motif workflow runtime on the local cluster and the workstation (hidden by 1-core cluster datapoint) are highly similar for all three models due to the predictable nature of their resource attributes. These resource platforms do not have any queue or VM startup overheads and have minimal data transfer time to/from the desktop. Most time is spent running the workflow tasks.

The graybox and blackbox models provide a higher time estimate for BigRed HPC than the whitebox model, but accurately estimate the total time on SDSC HPC. The SDSC queue has a more uniform wait time for different job runtimes while this varies sharply for the BigRed.

The cloud runtime estimate is consistent on all three models. This is because the deterministic VM start time overhead is independent of the task runtimes and paid only once in all three models. The intermediate data transfer times, hidden to the blackbox model, are dwarfed by compute and VM start times.

Figure 3(b) shows that there is negligible variation between the time estimates for blackbox and graybox models for Motif, which means that blackbox specifications are sufficient when compared to graybox. The errors between blackbox and whitebox shown in Fig. 3(a) are small except for BigRed, as explained above.

The runtime estimates for GWAS, which is more complex than Motif with its multiple map-reduce stages, are shown in Figures 4(a – c) for the three models. We can draw similar conclusions from them.

5. RELATED WORK

Workflow systems like Pegasus [18], Swift [10], and Trident [1] incorporate features to schedule tasks onto remote resources, such as Grids or clusters. For example, Swift uses Falkon execution framework to dispatch workflows tasks using multi-level scheduling [10]. Deelman, et al. [8] describe resource costs for running a Montage workflow on Amazon EC2. We use similar resource performance measures for Microsoft Azure Cloud [4].

Mandal et al. [12] propose a heuristic strategy using performance model based in-advance scheduling for optimal load-balancing on grid resources using the GrADS infrastructure [13]. Batch queue prediction has been used to predict queue wait times [14] and used in the performance model for workflow scheduling [16]. Blythe et al. [15] identify and evaluate task-based and workflow-based resource allocation strategies for workflows. The task-based algorithm greedily allocates tasks to resources. Workflow-based algorithms find an optimal allocation for the entire workflow and perform better for data-intensive applications. DAG scheduling algorithms [9] for Grids use heuristic models to schedule applications to meet time budgets. These strategies are similar to our whitebox approach and require detailed knowledge of workflow structure and requirements. Our blackbox and graybox strategies rely on limited information about the workflows and do not require the detailed performance models required for these other DAG scheduling strategies.

6. CONCLUSIONS & FUTURE WORK

Our evaluation demonstrates that the blackbox approach gives runtime estimates that are close to the graybox model for the GWAS and Motif workflows. While the absolute time estimates differ widely between blackbox and whitebox models, the blackbox approach is able to order the resource platform selections similar to the whitebox model in many cases. This fulfills our intended goal of understanding high-level resource platform selection with limited workflow knowledge. Thus, our model can serve as a basis for scientists to plug in high-level characteristics for a workflow and get estimates of a suitable platform for it.

As future work, we will study workflows that are more complex, and vary both in structure and in resource needs in an effort to identify the class of workflows that are suitable for accurate blackbox estimations. Other aspects of resource allocation decisions, such as resource utilization and cost, need to be compared for the different models.

7. ACKNOWLEDGMENTS

This work was supported by the Director, Office of Science, of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231. We thank Catharine van Ingen, Roger Barga and Jennifer Listgarten (MSR), Keith Jackson (LBL) and Emad Soroush (UW) for detailed discussions.

8. REFERENCES

[1] Listgarten, J. 2010. Correction for Hidden Confounders in the Genetic Analysis of Gene Expression. *(In submission)*

[2] Ramakrishnan, L. and Gannon, D. 2008. A Survey of Distributed Workflow Characteristics and Resource Requirements. *Technical Report TR671*, Indiana University.

[3] Nurmi, D., et al. 2009. The Eucalyptus Open-Source Cloud-Computing System. In *CCGrid*.

[4] Simmhan, Y., et al. 2010. Bridging the Gap between Desktop and the Cloud for eScience Applications. *(In Submission)*

[5] https://portal.teragrid.org/hpc-queue-prediction

[6] Ramakrishnan, L., et al. 2009. VGrADS: enabling e-Science workflows on grids and clouds with fault tolerance. In *SC*.
[7] Tilson, J. L., et al. 2007. MotifNetwork: Genome-Wide Domain Analysis using Grid-enabled Workflows, In *BIBE*.
[8] Deelman, E., et al. 2008. The Cost of Doing Science on the Cloud. In *SC*.
[9] Sakellariou, R., et al. 2007. Scheduling Workflows with Budget Constraints. *Integrated Research in GRID Computing*, Springer.
[10] Raicu, I., et al. 2007. Falkon: A fast and light-weight task execution framework. In *SC*.
[11] Wieczorek, M., et al. 2008. Taxonomies of the Multi-Criteria Grid Workflow Scheduling Problem. *Grid Middleware and Services*, Springer.
[12] Mandal, A., et al. 2005. Scheduling Strategies for Mapping Application Workflows onto the Grid. In *HPDC*.
[13] Kennedy, K., et al. 2002. Toward a Framework for Preparing and Executing Adaptive Grid Programs. In *NSF Next Gen. Sys. Prog. Workshop*.
[14] Brevik, J., et al. 2006. Predicting Bounds on Queueing Delay for Batch-scheduled Parallel Machines. In *PPoPP*.
[15] Blythe, J., et al. 2005. Task Scheduling Strategies for Workflow-based Applications in Grids. In *CCGRID*.
[16] Nurmi, D., et al. 2006. Evaluation of a Workflow Scheduler Using Integrated Performance Modelling and Batch Queue Wait Time Prediction. In *SC*.
[17] Simmhan, Y., et al. 2009. Building the Trident Scientific Workflow Workbench for Data Management in the Cloud. In *ADVCOMP*.
[18] Lee, K., et al. 2008. Adaptive Workflow Processing and Execution in Pegasus. In *WaGe*.

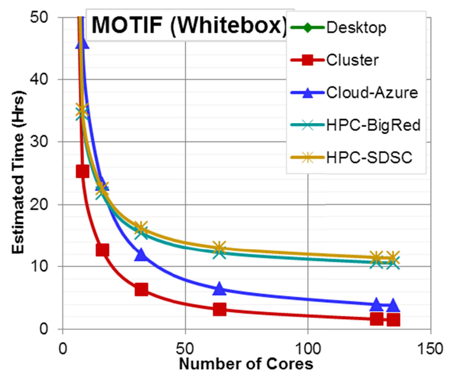

Figure 3(a). MOTIF runtime estimate with increasing cores using *whitebox* model. (Linear plot)

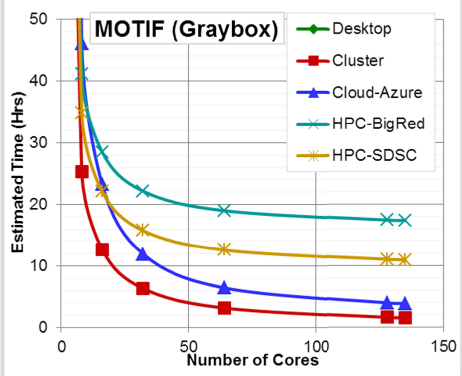

Figure 3(b). MOTIF runtime estimate with increasing cores using *graybox* model. (Linear plot)

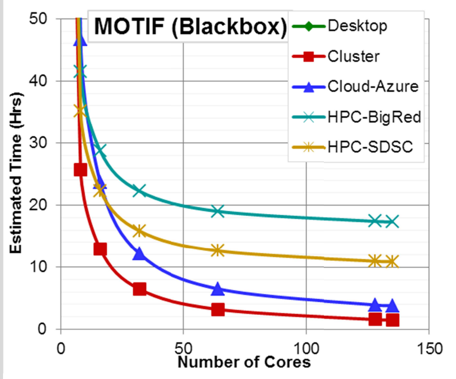

Figure 3(c) MOTIF runtime estimate with increasing cores using *blackbox* model. (Linear plot)

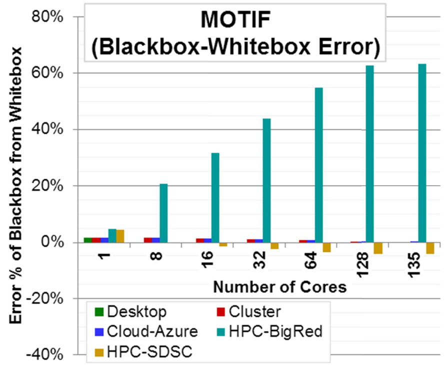

Figure 4(a). MOTIF runtime estimate difference % between *blackbox* and *whitebox* resource selection models with increasing number of cores.

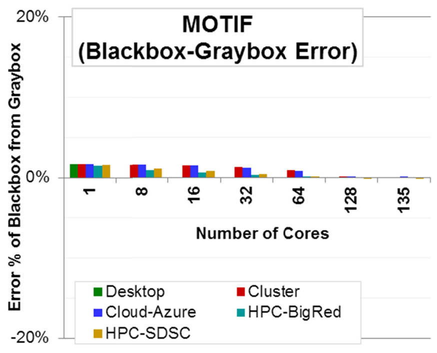

Figure 4(b). MOTIF runtime estimate difference % between *blackbox* and *graybox* resource selection models with increasing number of cores.

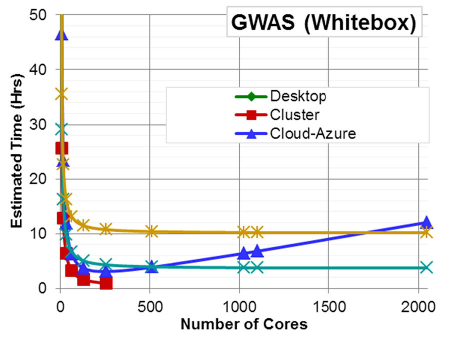

Figure 5(a). GWAS runtime estimate with increasing cores using *whitebox* model. (Linear plot)

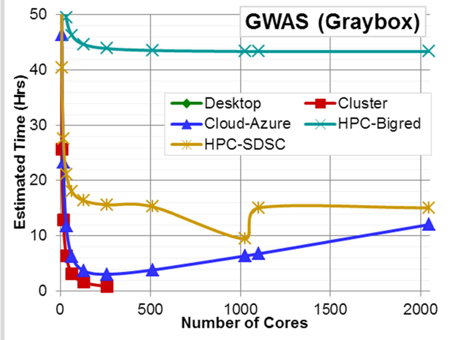

Figure 5(b). GWAS runtime estimate with increasing cores using *graybox* model. (Linear plot)

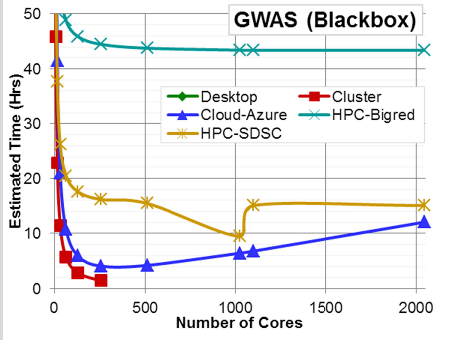

Figure 5(c). GWAS runtime estimate with increasing cores using *blackbox* model. (Linear plot)