

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.

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. At the same time, it can eliminate the challenges of restricted software environments and queue delays in shared high performance computing environments.

Choosing from these diverse resource platforms for a workflow execution poses a challenge for many 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 task scheduling onto different resources, these methods often require fine-scale characterization of the workflow that is onerous for a scientist.

In this position paper, we describe our early exploratory work into using *blackbox* characteristics to do a cost-benefit analysis across of using cloud platforms. We use only very limited high-level information on the workflow *length*, *width*, and *data sizes*. The length and width are indicative of the workflow duration and parallelism. The data size characterizes the IO requirements. We compare the effectiveness of this approach to other resource selection models using two exemplar scientific workflows scheduled on desktops, local clusters, HPC centers, and clouds. 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 cloud computing 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.

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 scheduling workflows is frequently ad hoc, 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 also 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 position 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 existing 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 performance 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?
- What workflow attributes determine their suitability for specific resource platforms?

The rest of this paper is organized as follows. Section 2, provides an overview of common resource platforms and workflow characteristics that guide resource selection; section 3 introduces our blackbox model and describes it in the context of whitebox and graybox models; section 4 presents a comparative evaluation of the blackbox model accuracy for two genomics applications; section 5 describes related work; and section 6 has our conclusions and future work.

2. OVERVIEW

2.1 Resource Platforms

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

2.1.1 Desktop Workstation Resources

A large number of science applications today still run on the desktop. Powerful multi-core machines can now match small clusters in their compute power. However, growth of data and the nature of analysis are far exceeding what is possible even on high-end workstations. The interactive nature of some scientific processes does make it necessary to transfer the final data to a user's desktop for visualization or validation. The workstation allows a user to exercise complete control over the software environment as well as on the privacy of the data.

2.1.2 Local Cluster Resources

Scientists often own and operate mid-sized local clusters (≤ 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.

2.1.3 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. While users can gain access to a larger resource pool, they incur queue wait times that vary with the system load when they want to run their computations. Users in this environment 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.

2.1.4 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. The on-demand access to unlimited cycles

eliminates contention with other users. However, resource virtualization can impact some scientific applications and overheads like Virtual Machine (VM) start time and (comparatively) low network bandwidth from desktop to cloud can impact application runtime. Users need to be aware of resource usage to stay within budgeted funds, and the model of paying for resource usage with a credit card is different from current scientific budget process. Recently, as the demand has spiked, cloud providers like Amazon are offering a tiered model with different access levels.

2.2 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 determine resource selection [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 and networking resources can be specified in terms of input and output file sizes as well as characteristics such as access patterns. Memory requirements may also be key for some data intensive applications.

2.3 Resource Platform Attributes

Resource platforms exhibit characteristics that can be used to evaluate their aptness for running workflows with certain attributes. Availability of both resource and workflow attributes helps perform 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 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. *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 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.

Finally, *cost* of using the resource is a factor. 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.

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.

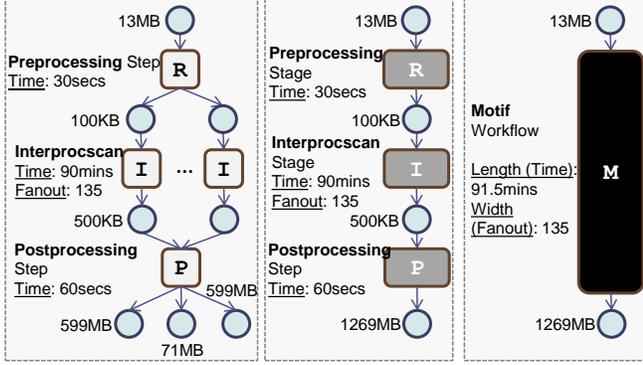


Figure 1. Motif workflow attributes used for whitebox (left), graybox (center), and blackbox (right) workflow scheduling. Each model requires less workflow characterization by the user

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 based on some number of workflow and resource attributes to quantify an optimization factor. In this paper, we limit our optimization factor to minimizing 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 selection models for the Motif workflow introduced in the next section.

- **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 known from the structure. Given this relatively fine-grained detail, each workflow task can be scheduled independently and, potentially, on a different resource platform. Each task incurs a latency time to access one CPU core, but needs the core only for the duration of that task. The time-optimization function for this model gives the total workflow time as:

$$F_{\text{WorkflowTime}} = \sum F^i_{\text{StageTime}}$$

where $F^i_{\text{StageTime}}$ is the time taken by workflow stage i given by:

$$F^i_{\text{StageTime}} = T^i_{\text{Data}} + \text{Ceil}(N^i_{\text{TaskWidth}}/N^i_{\text{Cores}}) \times T^i_{\text{LatencyOne}} + (T^i_{\text{TaskLength}} \times N^i_{\text{TaskWidth}}) / N^i_{\text{Cores}}$$

Where:

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

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

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

$N^i_{\text{TaskWidth}}$: Width or fanout of the number of tasks in i^{th} stage;

N^i_{Cores} : Number of available cores for running the tasks, and $N^i_{\text{Cores}} \leq N^i_{\text{Tasks}}$.

- **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 attributes within each workflow stage are not known. As such, this model needs to acquire the maximum possible number of CPU cores up to its width before it can run. For optimization, the model can look ahead to acquire and retain cores for subsequent stages until the workflow completes. The time-optimization function for this model gives the total workflow runtime as the sum of all i workflow stage runtimes plus the latency time for acquiring the largest number of cores required from among all workflow stages:

$$F_{\text{WorkflowTime}} = T_{\text{LatencyMax}} + \sum F^i_{\text{StageTime}}$$

Where:

$T_{\text{LatencyMax}}$: Latency time, due to queue wait or VM instance start time, to acquire maximum possible cores; and

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

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

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

Where:

T_{DataSum} : 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 Workload

4.1.1 Genome Wide Association Study (GWAS)

Genome Wide Association Studies (GWAS) use 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.

4.1.2 Motif Network

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. A pre-processing task splits a 13MB input file into 135 chunks that are individually operated upon by loosely-coupled, compute intensive *Interproscan* 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. So the workflow *width* is 135, *length* is 90mins, *data input* is 13MB and *data output* is 1269MB [7].

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 Network Weather 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 identical CPU Core to the above workstation. The bandwidth between the HPC center and user’s desktop client holding inputs and outputs is set at 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 20secs per additional VM with a 200sec overall startup time [4] and users are assumed to get between 1 and 2048 VMs at a time. The bandwidth between Azure Cloud and user’s desktop client holding inputs and outputs was measured at 1.2MB/sec.

4.3 Workload Evaluation

We simulated the GWAS and Motif workloads on each of the four resource platforms for each of the three resource selection models. The total workflow runtime is calculated using the time-optimization functions described earlier. We repeated the calculation varying numbers of cores in local cluster, shared HPC and cloud.

Figures 2(a–c) show Log-Log plots of the estimated GWAS workflow runtimes (Y Axis) as a function of the available number of concurrent cores (X Axis). Figures 3(a–c) are Linear plots of the same, zoomed into regions of interest. The pairwise percentage difference in time estimates between blackbox and whitebox models, and blackbox and graybox models are shown in Figures 4(a) and 4(b).

The GWAS 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.

The whitebox model provides a lower time estimate for BigRed HPC than graybox and blackbox, but is higher on SDSC HPC. This is due to the relatively higher queue wait time for shorter duration workflow tasks scheduled independently on SDSC by

whitebox, while the inverse effect is seen on BigRed’s queue. Since both graybox and blackbox use larger granularities for stage and workflow lengths, their estimates are consistent.

The cloud runtime estimate is consistent on all three models. This is because the deterministic VM start time overhead is paid only once in all three models and the intermediate data transfers time hidden in the blackbox is dwarfed by compute/VM start times.

The runtimes estimates for Motif are shown in Figures 5(a – c) for the models. We can draw similar conclusions from them.

5. RELATED WORK

Workflow systems like Pegasus, Swift, and Trident usually 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]. This typifies fine-grained resource selection where detailed workflow structure and resource needs are known. DAG scheduling algorithms [9] for Grids use heuristic models to schedule applications to meet time budgets. Our optimization functions are similar and extend to all three models we evaluate. Deelman, et al. [8] describe resource costs for running a Montage workflow on Amazon EC2. We use similar resource performance measures for Microsoft Azure [4].

6. CONCLUSIONS & FUTURE WORK

From these initial evaluations, we observe 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 platforms with limited workflow knowledge.

As future work, we will study the sources of errors between the different models to help build heuristics that can help improve the blackbox runtime predictions. These will help drive a resource selection service for scientists to plug in high-level characteristics for a workflow and get estimates of the optimal platform for it.

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 also thank Catharine van Ingen and Roger Barga (MSR), and Keith Jackson (LBL) for detailed discussions.

8. REFERENCES

- [1] Correction for Hidden Confounders in the Genetic Analysis of Gene Expression, J. Listgarten, 2010 (*In submission*)
- [2] A Survey of Distributed Workflow Characteristics and Resource Requirements, L. Ramakrishnan and D. Gannon, *Technical Report TR671*, Indiana University, 2008.
- [3] The Eucalyptus Open-Source Cloud-Computing System, D. Nurmi, et al., in *CCGrid*, 2009.
- [4] Bridging the Gap between Desktop and the Cloud for eScience Applications, Y. Simmhan, et al., in *Cloud*, 2010 (*In Submission*)
- [5] <https://portal.teragrid.org/hpc-queue-prediction>
- [6] VGrADS: enabling e-Science workflows on grids and clouds with fault tolerance, L. Ramakrishnan, et al., in *SC* 2009.
- [7] MotifNetwork: Genome-Wide Domain Analysis using Grid-enabled Workflows, J. L. Tilson, et al., in *BIBE*, 2007.
- [8] The Cost of Doing Science on the Cloud, E. Deelman, et al., in *SC* 2008.
- [9] Scheduling Workflows with Budget Constraints, R. Sakellariou, et al., *Integrated Research in GRID Computing*, Springer, 2007.
- [10] Falkon: A fast and light-weight task execution framework, I. Raicu, et al., in *SC* 2007.
- [11] Taxonomies of the Multi-Criteria Grid Workflow Scheduling Problem, M. Wiczczonek, et al., *Grid Middleware and Services*, Springer, 2008.

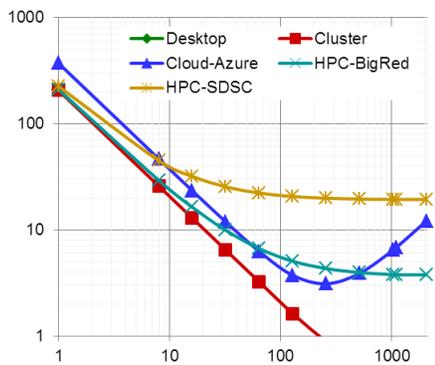


Figure 2(a). GWAS runtime in hours with increasing cores using *whitebox* model. (Log-Log)

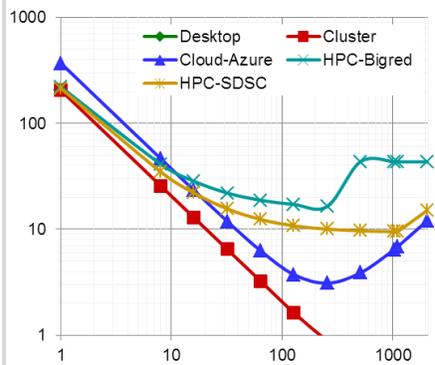


Figure 2(b) GWAS runtime in hours with increasing cores using *graybox* model. (Log-Log plot)

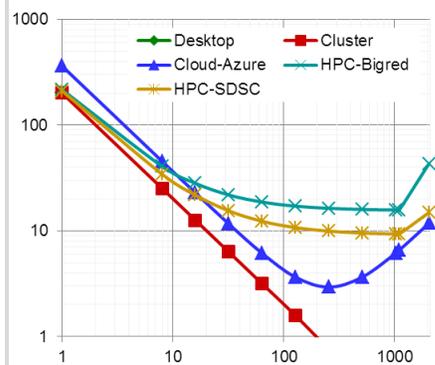


Figure 2(c).GWAS runtime in hours with increasing cores using *blackbox* model. (Log-Log plot)

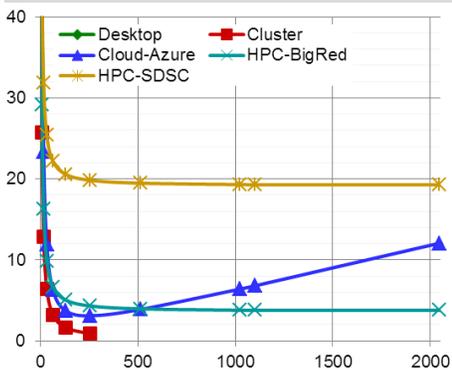


Figure 3(a). GWAS runtime in hours with increasing cores using *whitebox* model. (Linear)

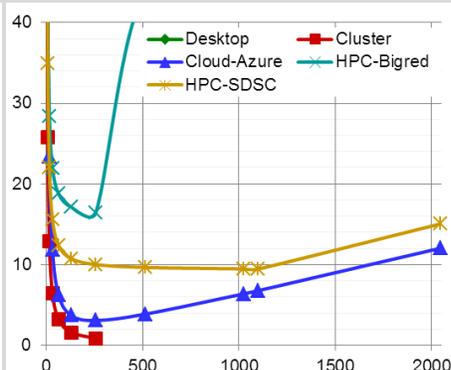


Figure 3(b). GWAS runtime in hours with increasing cores using *graybox* model. (Linear plot)

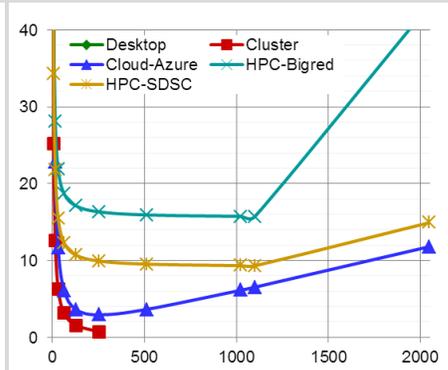


Figure 3(c) GWAS runtime in hours with increasing cores using *blackbox* model. (Linear plot)

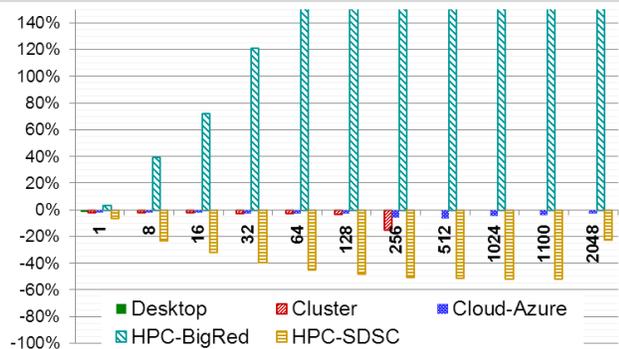


Figure 4(a). GWAS runtime difference % between *blackbox* and *whitebox* resource selection models with increasing number of cores.

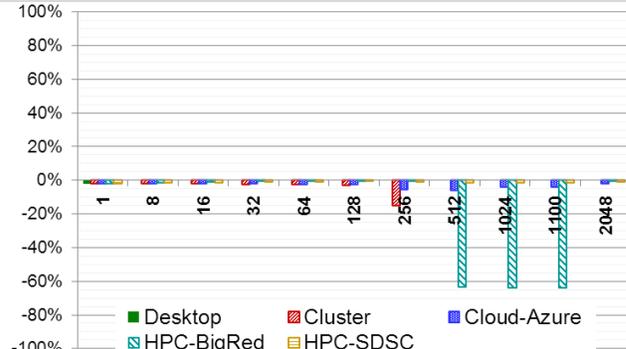


Figure 4(b). GWAS runtime difference % between *blackbox* and *graybox* resource selection models with increasing number of cores.

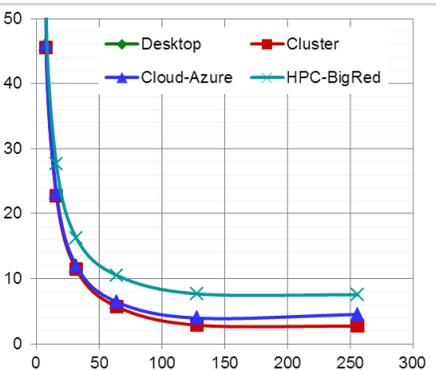


Figure 5(a). Motif runtime in hours with increasing cores using *whitebox* model. (Linear plot)

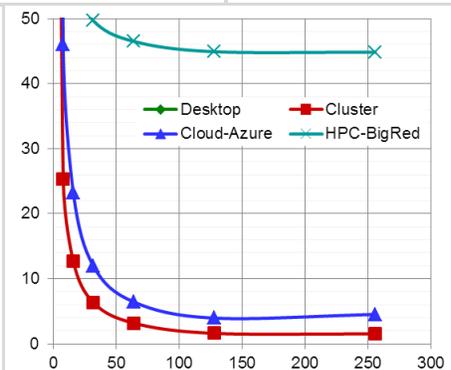


Figure 5(b). Motif runtime in hours with increasing cores using *graybox* model. (Linear plot)

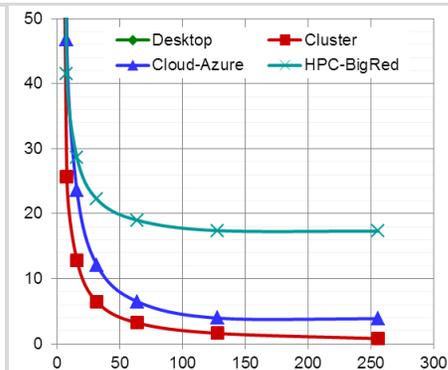


Figure 5(c). Motif runtime in hours with increasing cores using *blackbox* model. (Linear plot)