Integrating prediction, provenance, and optimization into high energy workflows

M Schram, V Bansal, R D Friese, N R Tallent, J Yin
K J Barker, E Stephan, M Halappanavar and D J Kerbyson
Pacific Northwest National Laboratory, Richland, WA, USA
E-mail: malachi.schram@pnnl.gov

Abstract. We propose a novel approach for efficient execution of workflows on distributed resources. The key components of this framework include: performance modeling to quantitatively predict workflow component behavior; optimization-based scheduling such as choosing an optimal subset of resources to meet demand and assignment of tasks to resources; distributed I/O optimizations such as prefetching; and provenance methods for collecting performance data. In preliminary results, these techniques improve throughput on a small Belle II workflow by 20%.

1. Introduction
Motivated by the complex workflows within Belle II, we propose an approach for efficient execution of workflows on distributed resources that integrates provenance, performance modeling, and optimization-based scheduling. The key components of this framework include: modeling and simulation methods to quantitatively predict workflow component behavior; optimization-based scheduling such as choosing an optimal subset of resources to meet demand and assignment of tasks to resources; distributed I/O optimizations such as prefetching; and provenance methods for collecting appropriate performance data.

The Belle II experiments deal with massive amounts of data [1]. Designed to probe the interactions of the fundamental constituents of our universe, the Belle II experiments will generate 15 petabytes of raw data per year. During the course of the experiments, the necessary storage is expected to reach over 200 petabytes. Data is generated by the Belle II detector, Monte Carlo simulations, and user analysis. The detector’s experimental data is processed and re-processed through a complex set of operations, which are followed by analysis in a collaborative manner. Users, data, storage and compute resources are geographically distributed across the world creating a complex data intensive workflow.

To select a good schedule for Belle II workflows, we decompose the problem into two parts. Given an estimated demand for compute and storage resources for a period of time, the first (top) level of decision making involves identifying an optimal (sub)set of resources that can meet the predicted demand. We use the analogy of “unit commitment” problem in electric power grids to solve this problem. Once a cost-efficient set of resources are chosen, the next step is to assign individual tasks from the workflow to specific resources. For Belle II, we consider the situation of Monte Carlo campaigns that involves a set of independent tasks (bag-of-tasks) that need to be assigned on distributed resources.
To support accurate scheduling, we use predictive performance modeling to rapidly predict task performance across available hardware platforms. The goals of this performance modeling work are to gain insight into the relationship between workload parameters, system characteristics, and performance metrics of interest (e.g., task throughput or scheduling latency); to characterize observed performance; and to guide future and run-time optimizations (including task/module scheduling). Of particular interest, these quantitative and predictive models provide the cost estimates to the higher-level task scheduling algorithms, allowing the scheduler to make informed decisions concerning the optimal resources to utilize for task execution.

Figure 1 illustrates our approach. To enable reasoning about different schedules, we develop modeling tools to generate accurate predictions of workflow task execution times (Sec. 2). To maximize throughput, we develop a scheduler that intelligently assigns tasks to resources, staggering schedules when necessary to minimize I/O contention (Sec. 3). To hide distributed data transfer latency, we use selective prefetching (which hides file system latency) and localized data compression (which hides network latency) (Sec. 4). We use a provenance system to collect performance data (Sec. 5) and validate models and expectations. We present preliminary results showing that the combination of these techniques can improve throughput on a small BelleII workflow by 20% (Sec. 6).

2. Performance Modeling
To reason effectively about workflow schedules, we model the performance of each constituent workflow task. Given a task name and its run-time arguments, a model is a small executable program that rapidly evaluates a run-time prediction for a given platform. A common approach for generating models is to employ statistical or machine learning techniques. Although these techniques can capture complex patterns, a disadvantage is that the relationship between model inputs and outputs is opaque, and hence ‘black-box’. Most of our efforts have focused on a ‘white-box’ approach that generates predictions using algebraic formulas of model parameters. The formulas are based on a static representation of a task’s dynamic behavior. To help create such models, we use the Palm tool [2,3] (Performance and Architecture Lab Modeling tool). The modeler uses task source code annotations to identify logical task parameters; Palm uses the annotations to parameterize its static and dynamic program analysis. Although annotations require human effort, ‘white-box’ models almost always require some human insight.

The model generation process is a one-time and offline event. Each models is parameterized not only by task inputs, but also by hardware platform parameters. Thus, given a task, its parameters and target platform, the models generate a series of predictions. These predictions enable the scheduler to reason about task completion within the larger distributed environment.

We created and validated a series of white-box and black-box models for common Belle II tasks. The black-box models, generated using standard regression techniques, are sufficient when
a task’s execution time is a linear function of an input parameter, such as the number of events to process. The white-box models can capture more complex behavior and abstract hardware parameters. For example, whereas one white-box model can be used across different machine types, black-box models must be trained for each task and machine combination.

3. Workflow Scheduling

Belle II uses DIRAC as the workflow management tool. In this section, we provide a brief outline of DIRAC and our approach to augment DIRAC with intelligent scheduling through cost-efficient selection of resources and energy-efficient assignment of jobs (tasks) to resources.

3.1. The DIRAC System

The DIRAC framework has served as a stable workflow management system for the Belle II experiments [4]. Two fundamental aspects of DIRAC that are of relevance to Belle II are: (a) the Workload Management System (WMS), and (b) the Data Management System (DMS). DIRAC continuously monitors resources using light-weight (computationally non intensive) pilot agent jobs. Based on the results of sanity checks performed by the pilot jobs, actual jobs (tasks) are submitted to resources using a ‘pull’ model, a model where the resources pull jobs from a central queue. DIRAC centrally coordinates tasks and manages their execution in an environment with differing compute and storage capabilities. Matching of tasks to resources in DIRAC is straightforward using the ClassAd based approach, where job requirements are matched against the resource specifications. All these activities are part of the Workload Management System.

Scheduling in DIRAC is currently driven by the availability of compute nodes. Therefore, DIRAC does not optimize for usage cost and energy consumption, or considers the effects of network connectivity between resources. Since modern computer architectures have evolved towards heterogeneity in several domains, the ability to effectively use different types of resources through job scheduling can lead to substantial gains in time-to-completion and task throughput. Heterogeneity in resources can arise from different types of resources such as dedicated and shared HPC platforms and cloud computing platforms, including heterogeneity of architectures. We therefore attempt to address these limitations in our approach through intelligent scheduling that we briefly describe next.

3.2. Assigning Jobs to Resources

The complex workflows from Belle II need an efficient system for scheduling of resources. We therefore developed a hierarchical framework for scheduling [5]. The first step in our framework performs a cost-efficient selection of resources to meet the predicted demand. Given a set of geographically distributed resources with different cost structures, the objective is to select a subset of resources that can meet the predicted demand at a minimum cost. We formulate this problem as a mathematical programming problem and solve it using Linear Programming and Generic Algorithm based methods [5,6]. The next step is to compute an efficient assignment of tasks to resources that have been selected from the first step. Different objectives for assignment can be chosen. For example, minimization of the total compute time, or the total energy consumption. Further, the assignment problem can also be formulated as a multi-objective optimization problem that can simultaneously optimize for multiple functions.

In our current implementation targeting resources within the institution, we have implemented a scalable assignment algorithm using semi-matching. Given a bipartite graph $G = (S \cup T, E)$ with weight function on the edges, a matching $M$ is a subset of edges such that no two edges in $M$ are incident on the same vertex. While vertex set $S$ represents tasks, $T$ represents resources, an edge $e(s, t) \in E$ represents if a task $s \in S$ can be executed on a resource $t \in T$, and the
weight of the edge can represent the compute time or energy consumption. Thus, matching can be seen as a unique pairing of vertices in $S$ (tasks) to vertices in $T$ (resources). A semi-matching relaxes the constraint and allows vertices in one set ($S$ or $T$) to be matched to multiple edges. Therefore, semi-matching enables multiple tasks to be mapped to a single resource. The goal of semi-matching can be to minimize the maximum load on any resource (sum of the compute tasks assigned to a resource), or to balance the average load on all the resources. Semi-matching can be computed efficiently for large problems [7].

In addition to semi-matching, we also implement a genetic algorithm (GA) based task scheduler. Genetic algorithms are iterative optimization techniques inspired by nature, where a population of possible solutions (chromosomes) ‘evolve’ over time to produce more efficient solutions. During each iteration, genetic operators are applied to the population to find new solutions, for example, swapping genetic material between two chromosomes or introducing new genetic material through random mutations. In our formulation, individual chromosomes are represented as a vector of tasks, where each task is assigned a specific machine. Chromosomes are evaluated based on the completion time of all tasks (i.e., makespan), thus chromosomes with smaller completion times are better solutions.

We currently modify the resource assignment directly in the job description and process them through DIRAC. Note that we have altered the traditional ‘pull’ model of DIRAC to a ‘push’ model where jobs are directed to specific set of resources. Once an optimal assignment is obtained for a given set of tasks, we can further optimize for data placement to minimize the overall cost of data movement. We plan to implement these strategies when we scale the Belle II workflows on geographically distributed resources.

4. I/O Optimization

Many extreme-scale scientific workflows must process or generate a large amount of data. Even though the performance of compute memory is improving, it is improving in a slower pace than that of processors. Moreover, existing I/O systems mainly target commercial systems. As a result, the I/O software stack from device driver to page caches are not optimized for extreme scale scientific applications. Thus, customizing I/O systems for extreme scale-scientific workflows can improve I/O performance and end-to-end application performance.

Many Belle II workflows access data stored remotely. Currently, whole files are copied from remote locations before tasks can begin processing, increasing total completion time. We have introduced the following optimizations. First, based on intercepting I/O calls, we only retrieve the parts of files that are needed instead of whole files. Second, we use a lightweight compression specially customized for scientific data to further reduce I/O overhead and latency. In the traditional approach of using compression, only data storage volume is reduced and end-to-end performance is not improved because the compressed data needs to first be decompressed. Furthermore, traditional compression only supports decompression from the beginning of an object. In the scenario where only a part of an object is needed, all the data before this part must be retrieved from disk and the I/O overhead can be increased instead of being reduced. In contrast, our compression techniques allow decompression to start at multiple points within a compressed object. Hence, if we only need a part of file, only that part is retrieved from I/O devices and transferred over the network, which can be a fraction of the cost of transferring everything from the beginning. Since our algorithm can decompress data faster than the data can be retrieved from I/O devices, I/O performance can be improved as much as compression ratios.

5. Provenance

To track tasks’ performance history and validate our models, we record provenance data such as workflow inputs, machine configurations, task processing relationships, data usage and transformations, as well as temporal and task performance specific information. The resulting
provenance database can be used for performance optimization and prediction, dynamic steering, experiment reproducibility and anomaly detection. We utilize the Provenance Environment [8] (ProvEn), a scalable provenance platform. ProvEn is an extended provenance capture platform, meaning it allows for not only the capture of traditional provenance data represented in semantic graph form, but also observed host system-level metrics (e.g., CPU’s, storage, networking, I/O, etc.) which are correlated with the traditional semantic provenance to impart an enhanced description of a workflow in relation to its run-time experience. ProvEn refers to this fusion of semantic and metric data as hybridization.

To track provenance corresponding to optimization experiments, the DIRAC job scheduler was selected to record job submission history. To do this, job submission description language (JSDL) script tasks were configured to report job launching requests and data snapshots of the target compute resource’s system metrics (e.g., load average and free memory at the time of submission) using ProvEn’s client API, known as the DescribeAnything Provenance Interface (DAPI). DAPI sends this DIRAC JSDL provenance and metrics information to the ProvEn data store where the descriptive information (i.e. semantic provenance) is saved in semantic graph form and the system-level metrics are saved as time-series records in the metric store. Although provenance was a core component in our initial experiments, it did lend insights into how provenance and metrics can be correlated to optimization tests to explain architecture, the state of the environment, and the nature of the experiment being conducted.

6. Experimental Setup and Results
Our goal is to show improved throughput on several real-world BelleII workflows. In this section, we present preliminary results on a small-scale workflow representing very common BelleII tasks. We are currently working on applying our techniques to a broader set of tasks.

6.1. Experimental Setup
The experiments where performed on six nodes of a heterogeneous cluster at PNNL. (Full scale experiments are expected to be thousands of nodes, distributed across the globe.) Each of the six nodes belonged to one of four node types, detailed in Table 1. The workload consisted of 25 Monte Carlo tasks and 25 ITop Recovery tasks. In future experiments, we will increase the scale with respect to both the workload and computing environment, while also accounting for the distributed nature of real Belle II workflows.

To ensure accurate measurements, the six nodes were reserved to prevent any other system users from utilizing them. Only one task is allowed to execute on a node at a time. For both task types, the individual tasks differed by the number of (physics) events processed. For each task, the number of events processed was sampled uniformly in the range 1-100. Task models were generated using task type, number of events, and node type to produce a prediction of execution time. Performance is measured as the makespan, i.e., the finishing time of the last task.

| CPU                                      | Memory | Nodes in Partition |
|------------------------------------------|--------|--------------------|
| 2 × Intel Xeon E5-2680 v2 @ 2.8GHz (10 core), 25MB L3 cache | 128 GB | 2                  |
| 2 × Intel Xeon E5-2650 v2 @ 2.60GHz (8 core), 20MB L3 cache  | 128 GB | 1                  |
| 2 × Intel Xeon E5-2698 v3 @ 2.30GHz (16 core), 40MB L3 cache | 128 GB | 1                  |
| 2 × AMD A10-7850K Radeon R7 @ 3.7GHz (4 core), 2 × 2MB L2 cache | 32 GB  | 2                  |

Table 1. Compute node types

All tasks begin with any input files stored on a separate server (with a Maxtor Atlas 8J147S0 SAS disk, 10K rpm, maximum read of 89 MB/s). We tested data transfers using two methods. The first (base case) is a standard blocking copy (scp) that transfers all the data to the node
before execution begins. The second uses our optimized I/O layer (Sec. 4) to pre-fetch the data and amortized the cost while the task is executing.

We tested three different scheduling algorithms. In all cases, SLURM is the underlying workload management system on the cluster. For a base case, jobs are allocated using the default SLURM scheduling algorithm. The second and third algorithms are the Semi-matching algorithm and Task-Based Scheduling Genetic Algorithm, respectively, described in Sec. 3.2.

For the semi-matching and GA based schedulers, we test two different allocation modes. The first mode allocates tasks to node partitions (nodes consisting of the same type), then allows SLURM to determine the specific node a task will execute on within the partition. The second mode allows the schedulers to allocate tasks specifically to the nodes themselves.

We test each scheduling and allocation mode with each of the two data transfer methods, for a total of ten different experiment combinations. Each combination executed the same set of 50 tasks. We perform 10 trials for each experiment and calculate the 95% confidence interval. A number of measurements are captured for each experiment. For experiments using the blocking copy, we explicitly measure the transfer times and task execution times separately, while for the optimized I/O, the cost of the data transfer is included in the task execution time. Additionally, we measure the cumulative idle time (no task being executed) and any overhead due to the scheduler accrued by each node.

6.2. Experimental Results

Experimental results are shown in Figure 2. The relative makespan, with respect to default SLURM and I/O (including 95% confidence intervals), is shown for each of the ten experimental combinations discussed above. Furthermore, the makespan is divided up to display the contributions of three main components: SLURM overhead/queue idle time, blocking I/O time, and task computation time. The first bar on the left represents the default configuration.
In all experiments, using the optimized prefetching I/O technique results in a decrease of the overall makespan of the system. Recall, our optimized I/O can overlap the transmission of data with computation, helping to hide some of the cost of the data transfer. Therefore, an explicit blocking I/O transfer before execution does not occur in the optimized cases, thus the disappearance of the green bars. Instead, any time spent waiting for data is captured in the computation time (blue bars).

Using either Semi-matching or the GA to allocate tasks to a partition (SLURM still assigns tasks to specific nodes) with blocking I/O results in increased makespan. Further analysis shows that while the actual computation time of the tasks decreases, both the scheduler overhead and blocking I/O time increase.

Allocating tasks directly to nodes results in decreased makespan for the Semi-matching and GA schedulers. Performance increases regardless of the I/O mode used. This fine-grained scheduling significantly reduces overheads introduced by SLURM, as well as reduces the amount of time nodes sit idle. When using Semi-matching and the optimized I/O we are able to reduce makespan by approximately 20%.

These initial experiments and results indicate many opportunities exist to improve throughput of large scale workflows. We show it is possible to combine multiple optimization techniques (dealing with different portions of the workflow) to improve overall performance. Future studies will examine distributed environments and increase scale with respect to cluster size, task types, and data volume.

7. Related Work
The current largest High Energy Physics (HEP) enterprise is the Large Hadron Collider (LHC) project [9]. Its four major experiments use distinct workflow management systems (WMS): ALICE (with AliEn [10]), ATLAS (with PanDA [11] and BigPanDA [12]), CMS (with glideinWMS [13] and Condor [14]), and LHCb [9] (with DIRAC [15]). Each WMS uses a greedy scheduler based on the ‘pull’ model. The ‘pull’ model works well in distributed environments and can easily account for heterogeneous resources. Because scheduling is greedy, tasks are scheduled on a first-come first serve basis to the first appropriate resource. Our contribution is to schedule by solving an optimization problem. The scheduling is rapid by using an efficient approximation method. To enable scheduling, we use quantitative predictive models for workflow tasks.

Current WMS techniques for improving distributed I/O are largely based on XRootD [16], e.g., ALICE [17], ATLAS [18] and CMS [19]. XRootD provides both a prefetching and data compression mechanism. However, both operate at a much coarser data granularity; and the compression algorithm is slower. As a result, when prefetching data with on-the-fly compression, our techniques both increase bandwidth and decrease latency. Further, our system also overlaps data transfer with task computation.

8. Conclusions
We have presented an integrated framework targeting the improved performance of complex distributed scientific workflows. This framework incorporates provenance data capture, predictive performance modeling, optimization-based scheduling, and I/O optimization. Using a simplified workflow capturing many of the salient characteristics of the Belle II high-energy physics workflow and executing on a small-scale heterogeneous distributed system, we have demonstrated the efficacy of our approach. Specifically, we have demonstrated the benefit of I/O prefetching which allows the movement of large data files to be overlapped with ongoing computation, intelligence incorporated into the task scheduling framework enabling optimized task location, and fine-grained control over ultimate task placement. Taken in aggregate, these techniques account for a 20% reduction in the overall execution time for our benchmark problem configuration. Future plans include increasing the complexity of both the target workflow and distributed hardware platform,
targeting more realistic large-scale workflows and systems. In addition, we plan to more deeply incorporate collected provenance information to continuously evaluate model prediction accuracy, and to use predicted task performance prediction to avoid I/O contention at shared resources, a major factor in overall workflow performance.

Acknowledgments
Todd Elsethagen (PNNL) and Bibi Raju (PNNL) are part of the ProvEn team. We are grateful for funding support from the U.S. Department of Energy’s (DOE) Office of Advanced Scientific Computing Research as part of “Integrated End-to-end Performance Prediction and Diagnosis.” Pacific Northwest National Laboratory is operated by Battelle for the DOE under Contract DE-AC05-76RL01830.

References
[1] Hara T and Belle II computing group 2015 J. Phys. Conf. Ser. 664 012002 doi:10.1088/1742-6596/664/1/012002
[2] Tallent N R and Hoisie A 2014 Palm: Easing the burden of analytical performance modeling Proc. of the 28th ACM Intl. Conf. on Supercomputing (New York, NY: ACM) pp 221–230 doi:10.1145/2597652.2597683
[3] Friese R D, Tallent N R, Vishnu A, Kerbyson D J and Hoisie A 2017 Generating performance models for irregular applications Proc. of the 31st IEEE Intl. Parallel and Distributed Processing Symp. (New York, NY: IEEE) To appear.
[4] Tsaregorodtsev A, Brook N, Ramo A C, Charpentier P, Cloiser J, Cowan G, Diaz R G, Lanciotti E, Mathe Z, Nandakumar R, Paterson S, Romanovsky V, Santinelli R, Sapunov M, Smith A C, Miguelez M S and Zhelezov A 2010 J. Phys. Conf. Ser. 219 062029 doi:10.1088/1742-6596/219/6/062029
[5] Halappanavar M, Schram M, de La Torre L, Barker K, Tallent N R and Kerbyson D 2015 Towards efficient scheduling of data intensive high energy physics workflows Proc. of the 10th Workshop on Workflows in Support of Large-Scale Science (New York, NY, USA: ACM) pp 3:1–3:9 doi:10.1145/2822332.2822335
[6] Friese R D 2016 Efficient genetic algorithm encoding for large-scale multi-objective resource allocation Proc. of the 30th Intl. Parallel and Distributed Processing Symp. Workshops (New York, NY: IEEE) doi:10.1109/IPDPSW.2016.36
[7] Harvey N J A, Ladner R E, Lovász L and Tamir T 2006 J. Algorithms 59 53–78 doi:10.1007/978-3-540-45078-8_26
[8] Elsethagen T, Stephan E, Raju B, Schram M, MacDuff M, Kerbyson D, van Dam K K, Singh A and Altintas I 2016 Data provenance hybridization supporting extreme-scale scientific workflow applications New York Scientific Data Summit (New York, NY: IEEE) pp 1–10 doi:10.1109/NYSDDS.2016.7747819
[9] The large hadron collider http://home.cern/topics/large-hadron-collider
[10] Bagnasco S, Betev L, Buncic P, Carminati F, Cirstoiu C, Grigoras C, Hayrapetyan A, Harutyunyan A, Peters A J and Saiz P 2008 J. Phys. Conf. Ser. 119 062012 doi:10.1088/1742-6596/119/6/062012
[11] Nilsson P, Caballero J, De K, Maeno T, Potekhin M and Wenaus T 2008 The PanDA system in the ATLAS experiment Proc. of XII Advanced Computing and Analysis Techniques in Physics Research URL https://pos.sissa.it/cgi-bin/reader/contribution.cgi?id=070/027
[12] The BigPanDA project http://news.pandawms.org/bigpanda.html
[13] Strigoi I, Bradley D C, Holzman B, Mhashilkar P, Padhi S and Wurthwein F 2009 The pilot way to grid resources using glideinWMS WRI World Congress on Computer Science and Information Engineering vol 2 (New York, NY: IEEE) pp 428–432 doi:10.1109/CISIE.2009.950
[14] HTCondor http://www.cs.wisc.edu/condor/
[15] Paterson S, Cloiser J and the LHCb Dirac Team 2010 J. Phys. Conf. Ser. 219 072015 doi:10.1088/1742-6596/219/7/072015
[16] XRootD http://xrootd.org/
[17] Colamaria F, Colella D, Donvito G, Elia D, Franco A, Luparello G, Maggi G, Miniello G, Vallero S and Vino G 2015 J. Phys. Conf. Ser. 664 042013 doi:10.1088/1742-6596/664/4/042013
[18] Gardner R, Campana S, Dukeek G, Elmshaeuser J, Hanushevsky A, Hönig F G, Iven J, Legger F, Vukotic I, Yang W and the Atlas Collaboration 2014 J. Phys. Conf. Ser. 513 042049 doi:10.1088/1742-6596/513/4/042049
[19] Bloom K, Boccali T, Bockelman B, Bradley D, Daou S, Dost J, Fanzago F, Strigoi I, Tadel A M, Tadel M, Vuosalo C, Würthwein F, Yagil A and Zvada M 2015 Any data, any time, anywhere: Global data access for science Proc. of the IEEE/ACM 2nd Intl. Symp. on Big Data Computing (New York, NY: IEEE) pp 85–91 doi:10.1109/BDC.2015.33