Reduction of Workflow Resource Consumption Using a Density-based Clustering Model

Qimin Zhang
Key Laboratory of Space Utilization
Technology and Engineering Center for Space Utilization
Chinese Academy of Sciences
Beijing, China
zhangqimin16@csu.ac.cn

Nathaniel Kremer-Herman
Dept. of Computer Science and Engineering
University of Notre Dame
Notre Dame, USA
nkremerh@nd.edu

Benjamin Tovar
Dept. of Computer Science and Engineering
University of Notre Dame
Notre Dame, USA
btovar@nd.edu

Douglas Thain
Dept. of Computer Science and Engineering
University of Notre Dame
Notre Dame, USA
dthain@nd.edu

***Abstract*—An end user running a scientific workflow will often ask for orders of magnitude too few or too many resources to run their workflow. If the resource requisition is too small, the job may fail due to resource exhaustion; if it is too large, resources will be wasted though job may succeed. It would be ideal to achieve a near-optimal number of resources the workflow runs to ensure all jobs succeed and minimize resource waste. We present a strategy for addressing this resource allocation problem: (1) resources consumed by each job are recorded by a resource monitor tool; (2) a density-based clustering model is proposed for discovering clusters in all jobs; (3) a maximal resource requisition is calculated as the ideal number of each cluster. We ran experiments with a synthetic workflow of homogeneous tasks as well as the bioinformatics tools Lifemapper, SHRIMP, BWA and BWA-GATK to capture the inherent nature of resource consumption of a workflow, the clustering allowed by the model, and its usefulness in real workflows. In Lifemapper, the least time, cores, memory, and disk savings are 13.82%, 16.62%, 49.15%, and 93.89%, respectively. In SHRIMP, BWA, and BWA-GATK, the least cores, memory, and disk savings are 50%, 90.14%, and 51.82%, respectively. Compared with fixed resource allocation strategy, our approach provide a noticeable reduction of workflow resource consumption.**

***Index Terms*—high throughput computing (HTC), density-based clustering, automatic resource allocation, resource consumption optimization.**

I. Introduction

High throughput computing (HTC) is an essential component of the scientific enterprise in fields as diverse as computer science, physics, biology, economics, and aerospace. In HTC, researchers will often run a very large number of independent jobs across a large number of independent nodes (machines) [1]. Workflow management systems are a widely-used technology for organizing large HTC runs, organizing a body of work into a directed graph of jobs and the files consumed and produced by each job.

In these workflows, a failed job caused by resource exhaustion will waste time and resources as it is returned and re-submitted. Thus, the user tends to submit jobs with a large resource requisition to ensure all jobs succeed [2]. However, the goal of HTC is to run a workflow with a maximal number of jobs completed over a long period of time and minimal resource consumption of jobs. Therefore, researchers will often ask for how many resources (such as cores, memory, and disk) should be requested for each job. In practice, a researcher running a scientific workflow will have very little detailed information about the resources required by each job; also, jobs may vary widely in the required resources, due to differences in their functions and input data. We are then confronted with the resource allocation problem: how should the user make resource allocation strategies without detailed information of per job so as to minimize the waste of their resource consumption?

Data mining-based algorithms have been widely applied to scientific workflows to facilitate resource allocation and management. In [3], Menache et al. presented an online learning resource allocation to balance the computation cost and performance. It is based on the learning from the executing performance of prior job while collaborating history of spot prices and workload characteristics. In [4], Chen et al. presented a statistical model checker UPPAAL-SMC for optimizing and evaluating for cloud workflow resource allocation strategies. In [5], Xiong et al. proposed a cost-aware resource management system SmartLA using machine learning techniques for achieving the optimum profits. However, whether the prediction algorithm is nonlinear regression, a statistical model, or neutral network, a prediction model is never 100% accurate. Therefore, at least a small number of jobs will fail due to the resource requisition is too small, which means their methods are generally not able to ensure all tasks succeed according to their resource allocation strategies.

Considering that sometimes users need to run the same workflow multiple times or run a small workflow for testing

before running the whole workflow we present a comprehensive solution to the resource allocation problem for minimizing the waste of resource consumption while running all jobs successfully. We extract features from data collected by a resource monitor tool, then we use a density-based clustering model for discovering clusters in a large number of jobs, calculating optimal resource requisition for each cluster to optimize resource management. Section 2 gives some background, including Makeflow [6], the resource monitor tool [8], scientific workflows, HTCondor [7] and an overview of our method. Section 3 formalizes our ideas, presenting the algorithm and structure of our density-based clustering model, and our parameters' value evaluation strategies. Section 4 evaluates our approach using resource consumption data collected from production workflows in domains of bioinformatics. We demonstrate that our resource allocation strategy leads to an overall decrease in resource consumption compared to fixed allocations under the richness of resource variations. We run our evaluation on workflows drawn from the Makeflow Examples [16] archive, with the following result highlights: In Lifemapper, the least time, cores, memory, and disk savings are 13.82%, 16.62%, 49.15%, and 93.89%, respectively. In SHRIMP, BWA, and BWA-GATK, the least cores, memory, and disk savings are 50%, 90.14%, and 51.82%, respectively. Thus, our approach is able to provide an efficient way for workflow users. Also the clustering model presented in this paper is able to discover inherent relationship between hundreds of or thousands of jobs.

II. Implementation

A. Background

Scientific workflows allow users to easily express multi-step computational tasks, for example the Lifemapper workflow consists of three steps: pre-processing, maxent, and projection. A scientific workflow often contains hundreds of thousands of tasks, and tasks in a scientific workflow can be everything from short serial tasks to very large parallel tasks. The structure of workflows can vary from simple to complex. In our experiments, Lifemapper, BWA, and SHRIMP are simple workflow with three steps. BWA-GATK has a more complex structure.

We use Makeflow [6] to test our approach, which is a workflow system designed to allow users to express scientific workflows using the classic "Make" syntax. Makeflow can run jobs locally on a large multiple core machine, or interface with batch systems such as HTCondor, Torque, and SLURM; cluster manages such as Mesos and Kubernetes; and cloud services such as Amazon EC2 and Lambda. To schedule each job in the target systems, Makeflow allow users to set various resource requests, cores, memory (in MB), and disk (in MB) ahead of each job. We run scientific workflows both locally and the HTCondor to evaluate the behavior of our approach.

HTCondor [7] is a distributed batch computing system that creates HTC environment and allows many users to share their computing resources around the world. Researchers working with scientific workflows often submit lots of jobs to HTCondor and take advantage of all the idle cycles from under-used machines. We also use the HTCondor to run SHRIMP, BWA, and BWA-GATK to effectively utilizes the computing power of workstations.

The resource monitor tool [8] generates report files: a summary file with the maximum values of resource and time each job used, and monitors the maximum resources limit specified in the workflow file. If one of the resources goes over the limit, then the monitor terminates the task, including a report of the resource that was above the limit.

We assume the user provides a workflow description which indicates a set of jobs to be run, and the dependencies between them. A resource monitor tool to monitor the computational resources of a process. Then the density-based clustering model do data clustering with all report files and the ideal resource allocation strategy is calculated, in terms of a quantity of cores, memory, and disk.

B. Our approach

The structure of our solution is shown in Fig. 1. When we run a scientific workflow, we use a resource monitor tool to observe, record, and enforce resource limits on jobs. These observations are used as data source for the next step.

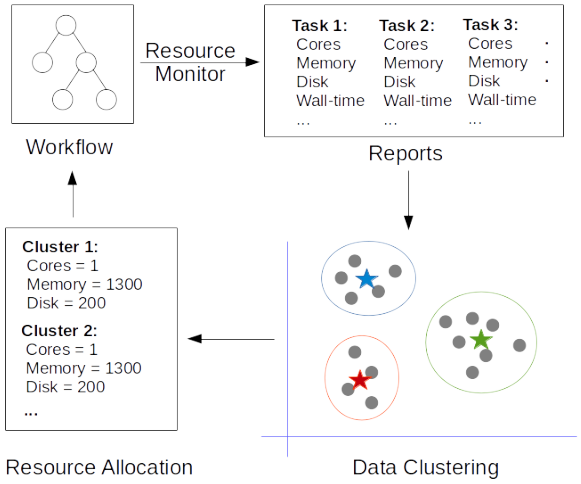


Fig. 1. System Components. The system consists of three main steps: (1) generate resource consumption reports, (2) data clustering with data collected by the resource monitor tool, (3) calculate ideal resource allocation strategy, then we re-run the workflow to measure the efficiency of our approach.

A density-based clustering model is presented to do data clustering. We extract cores, disk, memory and wall-time data from report files as four-dimensional features of each job and specify weights for each dimension to represent the importance of each feature. Then we cluster data according to the density aggregation of each point (job). After clustering, we label each task with its specified cluster number and the maximum resources of each cluster is calculated in order to ensure no task would fail. In this way, we obtain the ideal number of

cores, memory and disk of each cluster with their specified cluster label.

We evaluate these techniques on a synthetic workflow of homogeneous tasks as well as the bioinformatiocs tools Lifemapper [14], SHRIMP [13], BWA [15] and BWA-GATK [11]. The experiments were designed to capture the inherent nature of resources consumption of a scientific workflow, the clustering allowed by the model, and its usefulness in real workflows.

III. DENSITY-BASED CLUSTERING MODEL

We assume that there are some latent clusters in a workflow, and if we could set optimal resource request to each cluster rather than to global jobs, we may be able to use less resources to run. A scientific workflow may have hundreds of or thousands of jobs, it is unrealistic for users to have detailed information about each job, which means, users do not have prior knowledge about job clusters. So if they want to label jobs or discover latent clusters, they need to do classification without prior knowledge.

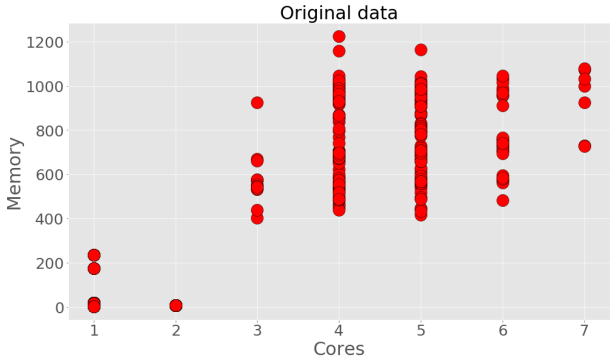
Clustering is the proper method we use when we want to label or categorize objects without prior knowledge, it finds common characteristics and latent internal relations between jobs, and then it provides predicted clusters. So clustering is adopted in our work.

We present a density-based clustering model for discovering the inherent nature of jobs in scientific workflows. In this section, we discuss the advantages of a density-based algorithm, model structure, crucial definitions, and strategy for selecting parameters in our clustering model.

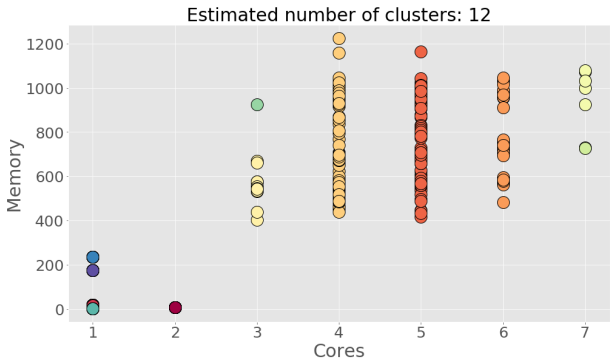
A. Clustering Algorithm

Clustering is used to discover inherent groupings, or relationships within all related jobs, and provide an ideal resource allocation strategy for users. Usually, users do clustering with very little or no detailed information of each task. We assume that users are not always able to determine how many clusters should be in the scientific workflow. Moreover, the distribution of hundreds or thousands of jobs are not always able to have regular shapes like circles or squares. Actually, the distribution of all tasks in a scientific workflow often have various shape and size. Some classic clustering algorithms, for example, K-means [9] which clusters data by trying to separate samples in n groups of equal variance, minimizing a criterion known as the inertia or within-cluster sum-of-squares, is limited by requiring the number of clusters to be specified in advance and its poor performance when clusters have irregular shapes.

The density-based clustering algorithm [10] was selected in our strategy due to two main advantages: (1) clusters found by density can be any shape; (2) the number of clusters is not required to be specified a priori. The density-based clustering algorithm views clusters as areas of high density separated by areas of low density, so the shape of clusters depends on density differences between points, which means clusters are not required to be convex shaped that is required in K-means and other clustering algorithms. Also the number of clusters is determined by parameters in our algorithm, so the number of clusters is not required to be specified beforehand.



(a) Before Clustering



(b) After Clustering

Fig. 2. Clustering result of BWA-GATK. The density-based clustering model used in our method is four dimensional, in terms of weighted cores, memory, disk and wall-time. Four dimensional distribution is hard to express in a figure. So we present the clustering result in two dimension: cores, and memory.

B. Feature Extraction

We consider the clustering results related to a job in four stages: (1) how many cores the job used, (2) how much memory the job used, (3) how much disk the job used, and (4) how much wall-time the job took. Cores, memory, and disk are three main parameters for resource allocation, so they are selected as parts of features. Wall-time is also selected as a feature for measuring the job size or function type.

The resource monitor provides summary files about computational resources of per task, so all these summary files are used to be data source of our clustering model. In our method, we extract features from reports in terms of the quantity of cores, memory, disk, and wall-time consumed by each task.

C. Model Parameters

Using data clustering, we are able to obtain several clusters, label all points (tasks) with cluster numbers and calculate the maximum resources (cores, memory, disk) of each cluster. As show in Fig. 2, we provide the clustering result of BWA-GATK-workflow [11] as an example. This workflow is a bioinformatics example using Makeflow to parallelize the

Burroughs-Wheeler Alignment and Genome Analysis Toolkit (BWA-GATK) tool.

The structure and parameters of the density-based clustering model have a large influence on the clustering results. Clustering model with different features or parameters will have various results. With data from the resource monitor tool, cores, memory, disk and wall-time are extracted as features of each single task, and we set weights to these four features. Then we provide a clustering model based on density for discovering clusters in large spatial databases. The central component is the concept of core samples, which are samples that are in areas of high density. A cluster is therefore a set of core samples, each close to each other (measured by some distance measure) and a set of non-core samples that are close to a core sample (but are not themselves core samples).

The following are several crucial definitions in the density-based clustering model:

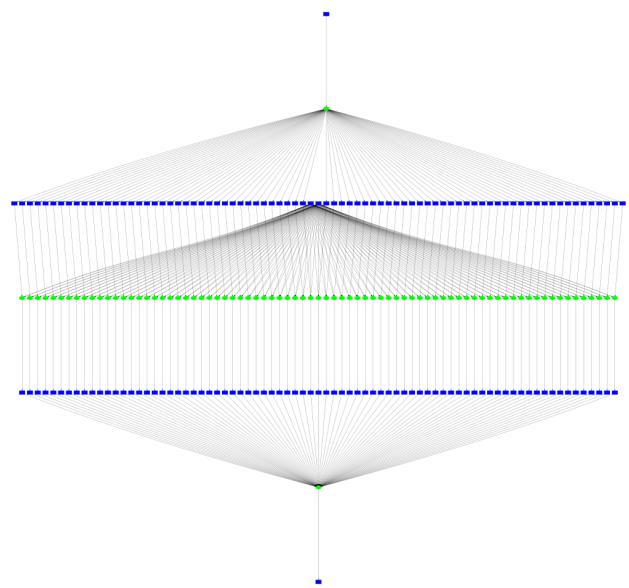
- ***distance***. The definition of distance in our method is weighted-Minkowski. The distance $D(X,Y)$ between two points $X = (x_1, x_2, x_3, x_4)$ and $Y = (y_1, y_2, y_3, y_4)$ is defined as:

$$D(X,Y) = (\sum_{i=1}^{4} w \times |x_i - y_i|^p)^{(1/p)} \quad (1)$$

  w represents the weights, we use the familiar Euclidean distance (p = 2), and the distance $D(X,Y)$ also represents inherent relationship or similarity between two points.
- ***weights***. Weights represent the importance of features(or axes). In our method, memory was considered to be the most representative and important feature of task attributes, so memory was given the largest weight. Weights of cores, memory, disk and wall-time were set to be 0.1, 0.7, 0.1, 0.1 respectively.
- ***eps***. The maximum distance between two samples for them to be considered as in the same cluster(or neighborhood). If the *eps* is too small, we will get too many clusters; on the other hand, if the *eps* is too large, we will get too few clusters due to points would be considered in the same cluster when their distance is smaller than *eps*. The value of *eps* should be modified in different workflows to achieve better performance.
- ***minimal samples***. The minimal number of sample in a cluster. If *minimal samples* was set to be more than 1, which means each cluster has two points at least, the point isolated with any other points would be seen as noise. To be noted, each job should have its cluster label to calculate its ideal resource requisition. Thus, *minimal samples* is set to be 1 in our method.

As seen in Fig 2 (b), rvery point represents a task, and different colors represent different clusters. As also shown in Fig. 2, all tasks have their own cluster labels, which means they have a specified number of resources according to their labels. In each cluster, the maximal resources consumed by tasks, in terms the quantity of cores, memory, and disk, are selected to be the ideal value of resources limitation. Then we are able to do an optimal resource allocation of all jobs in the scientific workflow.

(a) Clustering Result



(b) Workflow Structure

Fig. 3. Match Performance for SHRIMP. SHRIMP is a bioinformatics workflow and the workflow graph clearly shows the general structure. The corresponding cluster labels: (cluster 1, cluster 2, cluster 3), show the high matching performance of our clustering model and the real structure.

D. Parameters Selection Strategy

We have discussed the algorithm and structure of the density-based clustering model, also the value of *weights*, *minimal samples*, and *distance* have been discussed above. However, the value of *eps* is required to be modified in individual workflow in order to performance better.

How do we select the value of parameters in clustering model, and how do we evaluate the performance of clustering model with different value? We present two ways to evaluate the result of clustering: (1) compare with workflow structure, the higher the similarity, the better we think of the performance, (2) use Calinski-Harabaz index [12], the higher the score, the better we think of the performance.

1) Compare with Workflow Structure: Comparing the command line with the clustering result is a relatively simple method and this method is suitable for workflows with simple structure, such as SHRIMP [13]. In this method, the structure of workflow or the command line is used to be the ground truth of clustering results. If the clustering result is similar with the workflow structure, we think the clustering model have a good performance. When the value of *eps* makes the clusters match best with the workflow structure, this value will be selected as the optimal one. As shown in Fig. 3 for the SHRIMP workflow, of a best match between clustering result and the visualization of workflow structure. We present the figure in two axes, memory and disk, due to the value of cores consumption of all tasks in this workflow are 1.

As shown in Fig. 3, this workflow has three layers, and the clustering model also found three clusters. Meanwhile, corresponding clusters between Fig. 2(a) and (b) include the same tasks. We conclude the clusters obtained by clustering model match perfectly with the workflow structure, and the value of *eps* in this model is the optimal number.

We also select parameter value by comparing the workflow structure and clustering results in Lifemapper and BWA worflows. When the structure is simple, we could have an ideal clusters using this method. However, sometimes scientific workflow might be complex and hard to do the comparison. Then we need to use an index to evaluate the performance of clustering model.

2) Index evaluation: When the workflow structure is too complex or users do not have detailed information about the tasks in the workflow, the Calinski Harabaz index [12] can be used to evaluate the model, where a higher Calinski Harabaz score (C-H score) relates to a model with better defined clusters. The definition of C-H score is: for k clusters, the C-H score s is given as the ratio of the between-clusters dispersion mean and the within-cluster dispersion:

$$s(k) = \frac{Tr(B_k)}{Tr(W_k)} \times \frac{N-k}{k-1} \tag{2}$$

where $Tr(A)$ represents the matrix trace of matrix A, B_K is the between group dispersion matrix to measure the dispersion degree between different clusters, and W_K is the within-cluster dispersion matrix to measure the similarity of points within the same cluster, and they are defined by:

$$W_K = \sum_{q=1}^{k} \sum_{x \in C_q} (x - c_q)(x - c_q)^T \tag{3}$$

$$B_K = \sum_{q} n_q (c_q - c)(c_q - c)^T \tag{4}$$

with N being the number of points in our data, C_q being the set of points in cluster q, c_q being the center of cluster q, c be the center of E, n_q be the number of points in cluster q.

As shown in Fig. 4, BWA-GATK has a relatively complex structure with many layers, which means it is unrealistic for users compare the clustering result with workflow structure. Thus, we use C-H score to select an ideal value of *eps*. In this method, which the value provides the highest C-H score is the optimal number. Using BWA-GATK as an example, we present the relationship between different value of *eps* and its C-H score.

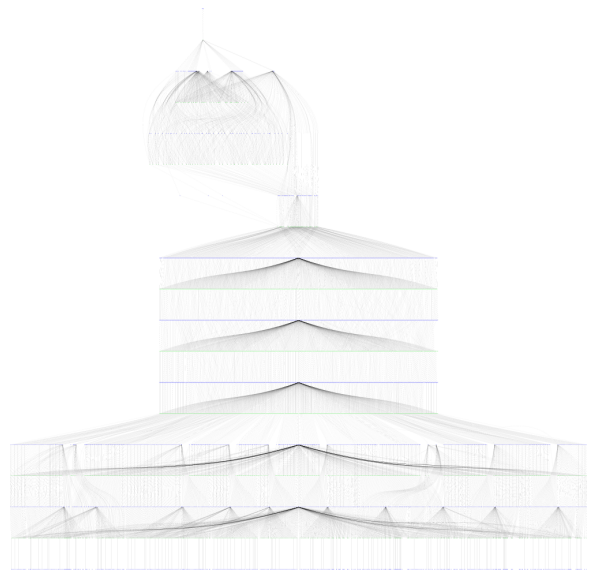


Fig. 4. Structure of BWA-GATK-workflow. BWA-GATK-workflow is a bioinformatics workflow that performs genotyping of sequences related to the oak tree.

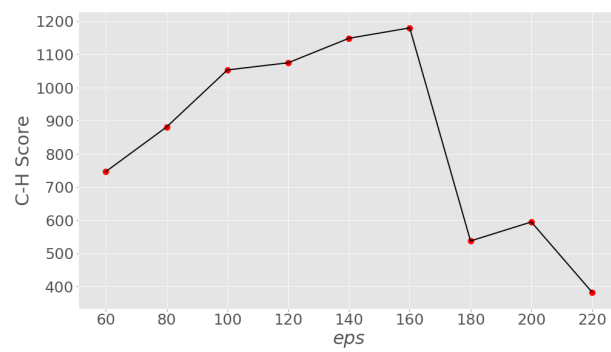


Fig. 5. *eps* and C-H Score. We measure the C-H score of various *eps* from 60 to 220 with each 20 intervals. Each red scatter point has its related C-H score.

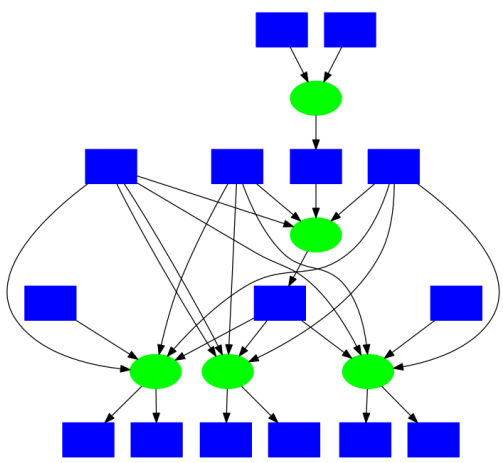
As shown in Fig. 5, C-H score fluctuates with different value of *eps*. When the *eps* is changed from 60 to 220, the C-H score first increases, then decreases, and the highest score is 1180 when *eps* is 160. So in this workflow, we select 160 as the optimal value of *eps*. When *eps* is selected, the number of clusters, k, will be determined in the clustering model. In this workflow, the number of clusters is twelve, and the clustering result is shown in Fig. 2.

In BWA-GATK workflow, we use C-H score to find the optimal parameter value for the clustering model. However, it should be noted that the optimal value we find is a local optimal number rather than a global optimal number since it is impossible for users to test all values from zero to infinity. Thus, we give a relatively sufficient number of values for

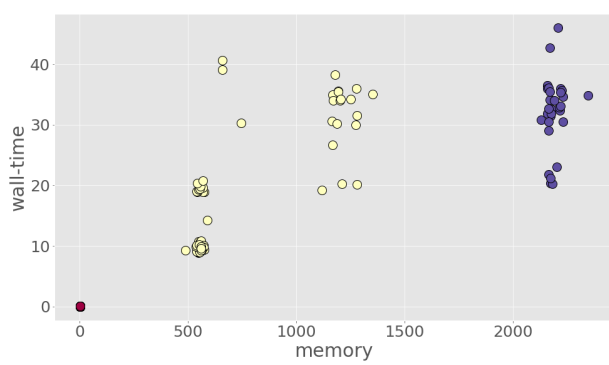
parameters, and then we use C-H index for finding the optimal one.

IV. EVALUATION

To evaluate our method, we apply it to data collected from production workflows runs on a local machine and the HTCondor batch system. For each job in a workflow, we use the resource monitor tool to produce a summary file about how many resources ware consumed by each job. Using this data, we use the density-based clustering model for labeling each task with its cluster number, calculating the maximal resources consumed by tasks in each cluster. Thus, we obtain an ideal resource allocation strategy. Using Makeflow, we convey the resources information about what resources each job needs by resetting the variables cores, memory, and disk ahead of each job. Then we re-run the workflow with this new resource allocation strategy, and capture the cores, memory, disk, and wall-time consumed by each job. Finally, we compare two resource consumption reports described above and analyze cores, memory, disk, and time savings of each workflow.



(a) Workflow Structure



(b) Clustering Result

Fig. 6. Clustering result for Lifemapper. In (a), we use a simplified model to shown the workflow structure clearly. In (b), We use four features to cluster data. Considering four dimensional distribution is hard to express in a graph, we present the clustering result in two dimension: memory and wall-time.

We evaluate these results on four different workflows:

Lifemapper [14] consists of three steps: pre-processing, maxent, and projection. This workflow produce the most interesting results due to the real data it used, which leads to the various resource consumption reports when we run the same Lifemapper- workflow multiple times. Thus, we run it for three times, and use all reports captured in these three times to get a more adaptive resource allocation strategy.

SHRIMP consists of 763 jobs, divided in three steps: split, analysis, and join. The structure of this workflow is relatively simple, which has discribed in Section 3.4.

BWA [15] is a bioinformatics workflow build on the Burrows Wheeler Alignment (BWA) tool, and it consists of 102 jobs, divided three steps: split, analysis, and join. Thus, the structure of this workflow is relatively simple, we analyze this workflow and do clustering in the same way as SHRIMP-workflow described in Section 3.4.

BWA-GATK is a bioinformatics workflow that parallelize the Burroughs-Wheeler Alignment and Genome Analysis Toolkit (BWA-GATK) tool. The structure of this workflow is the most complex one in these five workflows.

A. Improvement Index

We analyze the reduction of resource consumption in four stages: time, cores, memory, and disk savings. Their definitions are described in the following:

- ***time savings*** represents the decrease of running time. t is the number of cores consumed for running the workflow, then cores savings t_s are define as::

$$t_s = (t_b - t_a)/t_b \times 100\% \tag{5}$$

  t_b is time consumed before clustering, t_a is time consumed after clustering.
- ***cores savings*** represents the decrease of cores consumption. c is the number of cores consumed for running the workflow, then cores savings c_s are define as:

$$c_s = (c_b - c_a)/c_b \times 100\% \tag{6}$$

  c_b is the number of cores consumed before clustering, c_a is the number of cores consumed after clustering.
- ***memory savings*** represents the decrease of memory consumption. m is the number memory (GB.min) consumed for running the workflow, we find the m to be:

$$m = \sum_{i=1,2,3...} t_i \times m_i$$

  i is the number of task, so the memory savings m_s are define as:

$$m_s = (m_a - m_b)/c_1 \times 100\% \tag{7}$$

  m_a is memory consumed before clustering, m_b is memory consumed after clustering.
- ***disk saving*** represents the decrease of disk consumption. d is the number disk (GB.min) consumed for running the workflow, we find the s to be:

$$d = \sum_{i=1,2,3...} t_i \times m_i$$

Lifemapper	Job	resource allocation cores	memory (GB)	disk (GB)	Resource Consumption time (s)	cores	memory (GB.min)	disk (GB.min)
First run	153	8	8000	50000	1394	1224	539	3370
Second run	153	4	8000	50000	542	612	406	2540
Third run	153	2	8000	50000	246	306	293	1833
After clustering	153	cluster 1: Preprocessing			212	255	149	112
		1	100	3000				
		cluster 2: Maxent						
		2	5000	3000				
		cluster 3: Projection						
		2	1500	3000				

Fig. 7. Resource consumption for Lifemapper-workflow. We run the same Lifemapper three times with different resource allocation. Considering users often have little knowledge about detailed job, we use a fixed allocation strategy before clustering and record the resource consumption as shown in table. Comparing with our clustering resource allocation strategy, the resource consumption is noticeable as shown in the table.

Lifemapper	Resources Savings time	cores	memory	disk
1st run	84.79%	79.17%	72.36%	96.68%
2nd run	60.89%	58.33%	63.30%	95.59%
3rd run	13.82%	16.67%	49.15%	93.89%

Fig. 8. Resources savings for Lifemapper-workflow. The results of time, cores, memory, and disk savings are calculated as Function (5), (6), (7), and (8), respectively. For instance, the memory consumption of the first run is 539 GB.min, and the memory consumption of our approach is 149 GB.min. Thus, with function (7), we are able to measure memory savings between our approach and the first run is 72.36%.

i is the number of task, so the disk savings m_s are defined as:

$$d_s = (d_a - d_b)/c_1 \times 100\% \tag{8}$$

d_a is disk consumed before clustering, d_b is disk consumed after clustering.

B. Offline Analysis

We applied the density-based clustering model to all four workflows and compare the resource consumptions before clustering and after clustering. The results are used to demonstrate the efficiency improvements with our method. We show the results in Fig.s 6, 7, 8 for Lifemapper, and show the results in Fig.9, 10, 11 for BWA, SHRIMP, and GWA-GATK, respectively.

Lifemapper provides an interesting result. If users run the same workflow multiple times, they may get different output due to the special real data this workflow used. As shown in Fig. 6 (a), it consists of three steps, and each step contains 51 jobs. We ran this workflow three times, and we found that tasks in the second step Maxent consume various resources between each run. For example, task 68 consumes 2166 GB memory in the first run, but it consumes 1357 GB memory in the second run. Thus we use reports from all three times and use the maximal reports of each task. Then, we do the clustering for finding ideal clusters. We use the strategy described in Section 3.4.1 to find the ideal parameters' value and the clustering result is shown in Fig. 6 (b).

As shown in Fig. 6 (b), we obtain three clusters and each cluster contains the same task with their corresponding steps. We use this clustering model to calculate the max number of resources of each cluster in order to gain the ideal resource allocation. In practical workflow, we add a little more to the number we get to make sure all task succeed.

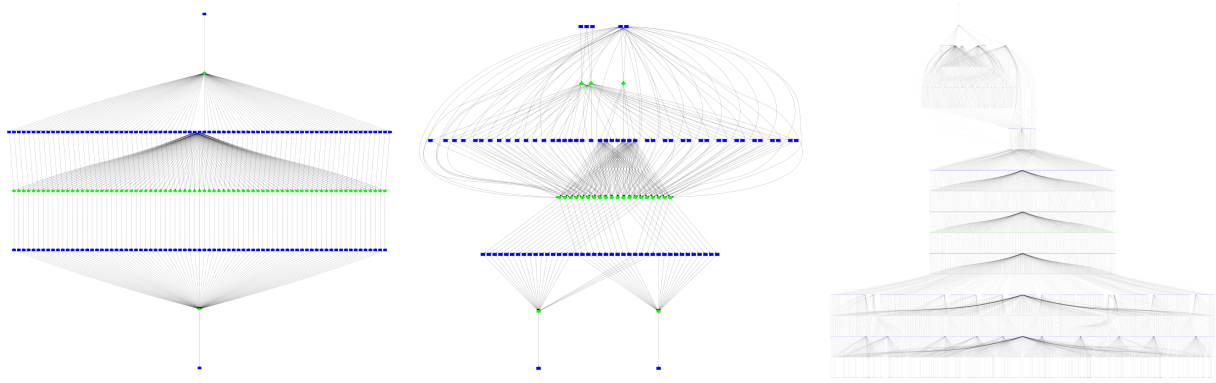
Then we re-run Lifemapper and the results of resources savings are shown in Fig. 8. We ran Lifemapper locally, so time savings are the reduction of running time for the whole workflow. We can note that the time savings corresponds closely to the cores savings. The reason is that a task would not run faster with more cores, but requesting too many cores will increase the waiting time of the following tasks. So when we use new resource allocation, in which we set cores requisition to 1 rather than 2, we decrease the waiting time for tasks. Thus, the time and cores savings are similar and all range from about 80% to about 15%. The results also show obvious memory and disk savings, the least memory savings are 49.15% and the least disk savings are 93.89%. We observe a very noticeable resources savings with our method.

We also applied our method to SHRIMP, BWA, and BWA-GATK. We show the visualization of their structure, the clustering results, and the resources savings results in Fig. 9, 10, 11, respectively. In all these four workflow experiments, we compare the resource consumption with a fixed resource management, which is often adapted by users, and the ideal resource allocation strategy provided by our approach.

Comparing Fig. 9 and Fig. 10, we are able to find that clustering result of our density-based clustering model is related to the layers, or steps, in the workflow structure, which indicates that our clustering model can find inherent relationship within hundreds or thousands of jobs. The clustering approach is not only able to calculate ideal resource allocation strategies, but also help users better understand their workflow.

We note that SHRIMP was run on top of the HTCondor batch system, and the number of workers we used fluctuated during the running. Often times, a workflow will run faster with more workers. Thus, time savings are not discussed in these three workflows.

In Fig. 12, the table highlights the advantages of using



(a) SHRIMP-workflow (b) BWA-workflow (c) BWA-GATK-workflow

Fig. 9. Visilization for scientific workflows

(a) SHRIMP-workflow (b) BWA-workflow (c) BWA-GATK-workflow

Fig. 10. Clustering results for scientific workflows. We use four features to cluster data in SHRIMP, BWA, BWA-GATK workflows. To show the clustering results clearly, we present the clustering results in two dimension: memory, and disk. Comparing corresponding workflows in Fig. 9 and Fig. 10, we are able to visualize the relation of real workflow structure and clustering result.

workflow	category	job	resource allocation				resource consumption		
			cluster	cores	memory (GB)	disk (GB)	cores	memory (GB.min)	disk (GB.min)
SHRIMP	w/o clustering	763	1	2	8000	50000	1526	18703	116892
	with clustering	763	1	1	100	1300	763	1845	9238
			2	1	1000	5000			
			3	1	100	10000			
BWA	w/o clustering	102	1	2	2048	2048	204	8.78	8.78
	with clustering	102	1	1	10	800	102	0.40	4.23
			2	1	200	1000			
			3	1	10	1400			
BWA-GATK	w/o clustering	530	1	8	8000	10000	4240	4287	5359
	with clustering	530	1	2	100	300	1734	215	278
			2	1	100	300			
			3	5	1500	100			
			4	6	1500	100			
			5	4	1500	100			
			6	3	1000	100			
			7	7	1500	100			
			8	7	1000	100			
			9	3	1200	100			
			10	1	100	3500			
			11	1	500	3500			
			12	1	300	2000			

Fig. 11. Clustering results for scientific workflows. We run SHRIMP, BWA, and BWA-GATK for evaluation. Considering users tend to set a relatively large resource requisition to run each job succeed, we use a relatively large resource requisition before clustering.

workflow	resources saved		
	cores	memory	disk
SHRIMP	50.00%	90.14%	92.10%
BWA	50.00%	95.44%	51.82%
BWA-GATK	59.10%	94.98%	94.81%

Fig. 12. Clustering results for scientific workflows. The results of time, cores, memory, and disk savings are calculated as Function (5), (6), (7), and (8), respectively. For instance, the memory consumption of the SHRIMP is 18703 GB.min, and the memory consumption of our approach is 1845 GB.min. Thus, with function (7), we are able to measure the memory savings between our approach and the first run is 90.14%.

memory allocations and disk allocations with our clustering model. The cores savings are 50.00%, the memory savings are 90.14%, and the disk savings are 92.10%. These results demonstrate that we are able to have a large percentage of memory and disk saved when we have clustering information rather than setting a relatively high resource requisition to ensure no task failed. We are capable running all tasks successfully and have a noticeable memory and disk savings with our method.

Resources savings results for BWA is also noticeable. We ran the BWA on the HTCondor batch system. We show the results of cores, memory, and disk savings. The cores savings are 50.00%, the memory savings are 95.44% (noticeable high), and the disk savings are 51.82%.

We composed small, medium, and large instances of BWA. We ran a medium one, and the clustering result of this medium size can predict the resource request for the large one. If users want to run a large workflow, they could first run a small batch of the workflows, and get new resource allocation with our clustering model. Then, users can run the large size with ideal resource allocation.

We do the same analysis to BWA-GATK. In Fig. 12, the table shows the results for BWA-GATK. We can highlight the following numbers: cores savings are 59.10%, and memory, and disk savings are 94.98%, and 94.81%, respectively. These results demonstrate that our method can produce large improvements without detailed information, even the workflow structure is complex.

Often times the workflows or scientific workflows are complex and users do not have detailed information about how many resources should be given to each job. Thus, comparing with fixed resource allocation strategies, our method provides an efficient way for determining resource requisition for jobs in complex workflows, which is useful for scientists.

V. CONCLUSIONS

This paper introduced a density-based clustering model for efficient use of computational resources for scientific workflows. We use reports collected by a resource monitor tool as a data source, use a clustering model for discovering inherent clusters within hundreds of or thousands of jobs, and then provide ideal resource allocation for future runs.

Our method is able to reduce resource consumption when users need to run the same workflow multiple times or run a small workflow to obtain optimal resources assignments before running the whole scientific workflow. Using our method, users can both run scientific workflows in a more efficient way, and discover some interesting inherent relationship between jobs.

VI. ACKNOWLEDGEMENTS

This work was supported by an iSURE summer fellowship at Notre Dame. We thank Tim Shaffer and Kyle Sweeney, researchers at the University of Notre Dame, for assistance with Makeflow and the HTCondor distributed batch computing system.

REFERENCES

[1] Jackson K R, Ramakrishnan L, Muriki K, et al. Performance Analysis of High Performance Computing applications on the Amazon Web Services Cloud. IEEE Second International Conference on Cloud Computing Technology and Science. IEEE, 2011:159-168.

[2] Tovar B, Silva R F D, Juve G, et al. A Job Sizing Strategy for High-Throughput Scientific Workflows. IEEE Transactions on Parallel & Distributed Systems, 2018

[3] I. Menache, O. Shamir and N. Jain. On-Demand, Spot, or Both: Dynamic Resource Allocation for Executing Batch Jobs in the Cloud, International Conference on Autonomic Computing (ICAC), 2014, pp. 177187.

[4] Chen M, Huang S, Fu X, et al. Statistical Model Checking-Based Evaluation and Optimization for Cloud Workflow Resource Allocation. IEEE Transactions on Cloud Computing, 2016, PP(99):1-1.

[5] P. Xiong, Y. Chi, S. Zhu and H. J. Moon. Intelligent Management of Virtualized Resources for Databased Systems in Cloud Environment, Int. Conference on Data Engineering (ICDE), 2011, pp. 8798.

[6] . Albrecht, P. Donnelly, P. Bui, and D. Thain, Makeflow: A portable abstraction for data intensive computing on clusters, clouds, and grids, in Proceedings of the 1st ACM SIGMOD Workshop on Scalable Workflow Execution Engines and Technologies, ser. SWEET 12. New York, NY, USA: ACM, 2012, pp. 1:11:13. [Online]. Available: http://doi.acm.org/10.1145/2443416.2443417

[7] Yuan D, Yang Y, Liu X, et al. A cost-effective strategy for intermediate data storage in scientific cloud workflow systems. IEEE International Symposium on Parallel & Distributed Processing. IEEE, 2010:1-12.

[8] Juve G, Tovar B, Silva R F D, et al. Practical Resource Monitoring for Robust High Throughput Computing. IEEE International Conference on CLUSTER Computing. IEEE, 2015:650-657.

[9] Arthur D, Vassilvitskii S. k-means++:the advantages of careful seeding. Eighteenth Acm-Siam Symposium on Discrete Algorithms, New Orleans, Louisiana. Society for Industrial and Applied Mathematics, 2007:1027-1035.

[10] Ester M, Kriegel H P, Xu X. A density-based algorithm for discovering clusters a density-based algorithm for discovering clusters in large spatial databases with noise. International Conference on Knowledge Discovery and Data Mining. AAAI Press, 1996:226-231.

[11] McKenna A, Hanna M, Banks E, Sivachenko A, Cibulskis K, Kernytsky A, Garimella K, Altshuler D, Gabriel S, Daly M, DePristo MA: The Genome Analysis Toolkit: a MapReduce framework for analyzing next-generation DNA sequencing data. Genome Res 2010, 20:12971303.

[12] T. Caliski, J Harabasz. A dendrite method for cluster analysis. Communications in Statistics, 1974, 3(1):1-27.

[13] Robinson C, Thain D. Automated packaging of bioinformatics workflows for portability and durability using makeflow. ACM, 2013:98-105.

[14] Beach, J.H., A.M. Stewart, C.J. Grady and J.A. Cavner, 2015. Lifemapper. [Computational services and software for species distribution modeling and biodiversity pattern analysis]. Web site: http://www.lifemapper.org

[15] Peters D, Luo X, Qiu K, et al. Speeding Up Large-Scale Next Generation Sequencing Data Analysis with pBWA. Journal of Applied Bioinformatics & Computational Biology, 2012.

[16] Makeflow Examples Archive. Available: http://github.com/makeflow-examples