Risk-aware Checkpoint Selection in Cloud-based Scientific Workflow

Mingzhong Wang*, Liehuang Zhu* and Jinjun Chen†
*Beijing Lab of Intelligent Information Technology
School of Computer Science
Beijing Institute of Technology
Beijing 10081, China
Email: {wangmz, liehuangz}@bit.edu.cn
†Faculty of Engineering and Information Technology
University of Technology, Sydney
PO Box 123, Broadway NSW 2007, Australia
Email: Jinjun.Chen@uts.edu.au

Abstract

Scientific workflows are generally computing- and data- intensive with large volume of data generated during their execution. Therefore, some of the data should be saved to avoid the expensive re-execution of tasks in case of exceptions. However, cloud-based data storage services come at some expense. In this paper, we extend the risk evaluation model, which assigns different weights to tasks based on their ordering relationship, to decide the occasion to perform backup or checkpoint service after the completion of a task. The proposed method computes and compares the potential loss with and without data backup to achieve the tradeoff between overhead of checkpointing and re-execution after exceptions. We also design the utility function with the model and apply a genetic algorithm to find the optimized schedule. The results show that the robustness of the schedule is increased while the possible risk of failure is minimized, especially when the generated data is not large.

Index Terms

Scientific workflow; Risk; Robustness; Exception handling; Checkpoint

1. Introduction

Scientific workflow is an important unifying mechanism to support modelling, management and execution of large-scale scientific computing in many complex e-science applications such as climate modelling, astrophysics, medical surgery and disaster recovery [1]. Technically, a scientific workflow is composed by a set of computing or data intensive tasks, as well as the dependencies between them. At runtime, each task needs to be instantiated for execution. Due to the cost-efficiency and flexibility of service-oriented computing and cloud computing, the execution of a task in scientific workflows is usually delegated to a certain web service. Flooded with large numbers of service candidates with different capabilities, costs, reputations and reliability, the performance and reliability of workflow execution are highly dependent on the service selection for each task [2].

Some research work proposed to use the concept of trustworthiness or reputation to model and measure the probability that each service will fulfill its commitments and promises from its historical behaviors, and incorporate trustworthiness into the service selection process to improve the reliability and robustness of the execution. However, most existing work [3], [4] has not taken the workflow structure into consideration, thus losing the opportunity to find better solutions. For example, a task at the end of a workflow should get more attention or priority than a task at the beginning, since the failure of the ending one will ruin all previous effort. Correspondingly, the workflow manager should allocate more resources (time or money) for the ending task to assure its successful completion.

In [5], we have proposed a risk evaluation model by assigning different impact factors or weights to tasks based on their ordering relationship. However, the model is designed for general workflow appli-

978-0-7695-4864-7/12 $26.00 © 2012 IEEE
DOI 10.1109/CGC.2012.46

cation without considering the computing- and data-intensive features of scientific workflows. Specifically, the execution of a task in scientific workflow takes long time and high cost to complete. Failure of any service may lead to a complete rerun of the whole workflow, which is not acceptable. Therefore, some results of tasks should be stored as checkpoint for future recovery in case of service failure. Consequently, the workflow manager should not only select the appropriate service to execute, but also need to decide whether to conduct a checkpoint with respect to the selected service. Therefore, in this paper we extend the risk model to measure and achieve the tradeoff between the overhead of backup storage and the cost of data transfer/regeneration in failure, making the service selection and execution more efficient and robust.

The remainder of this paper is organized as follows. Section 2 gives a concrete example to illustrate the problem. Section 3 provides an introduction to the risk evaluation model, as well as its extension to address the computing- and data-intensive features of scientific workflows to improved the efficiency and reliability of execution. The design of experiment and results are demonstrated and discussed in Section 4. Section 5 provides an overview of the state of the art. Finally, in Section 6 we present our conclusions and perspective for future work.

2. Motivation

Figure 1 illustrates a general example of bioinformatics workflow adapted from [6]. In the workflow, each task is represented as a rectangle, while its output data are represented in ovals. In the flow, there are a set of functional eligible service candidates to complete each task, with the input from its parent tasks and the output to its children tasks. In the example, MAFFT (Multiple Alignment with Fast Fourier Transform) is a multiple sequence alignment program for amino acid or nucleotide sequences, PHYLIP (PHYLogeny Inference Package) is a package of programs for inferring phylogenies (evolutionary trees), and UNIPROT (Universal Protein Resource) stands for a comprehensive and freely accessible database of protein sequence and functional information.

As merchants in the market, services are usually with different execution duration, price, and reliability for the same task, making the selection of optimized service critical for the running of a workflow. We have proposed to take the task dependency into consideration for improved workflow reliability in [5]. For example, for a simplified workflow consisting of two tasks MAFFT and PHYLIP, the reliability of PHYLIP has higher priority than MAFFT with respect to minimizing the side effect of service failure.

Since MAFFT and PHYLIP are expensive and time-consuming to execute, if the second one has high probability to fail, it would be wise to save the output of MAFFT for re-execute of the flow. Unfortunately, the call of a backup service incurs delay and expense in cloud-based scientific workflow for the consumption of bandwidth and storage. Specifically, to make the decision of backup cost-effective and reliable, the risk of failure for each task needs to be considered, and the tradeoff between overhead of backup and recovery needs to be achieved. Therefore, optimally deciding whether to call a backup service after a task completion is the focus of this paper.

3. Risk-aware service selection for scientific workflow

In this section, an extended risk evaluation model is proposed to make the service selection and execution more efficient and robust. The section first provides the preliminary knowledge about workflows and the basic risk evaluation model. Then, an extended model with backup service is introduced to achieve the tradeoff between overhead of backup and re-execution after exceptions. Finally, a genetic algorithm incorporating the model is designed and applied.

3.1. Workflow model

A workflow process can be specified as a directed acyclic graph (DAG). In case of a workflow with iterations, its corresponding cyclic graph can be converted into a DAG by loop unrolling. For the purpose of this paper, all loops are assumed to be terminable. [7] provides more discussions on the loop unrolling.

Definition 1 (Workflow graph definition): A workflow is abstracted as a DAG, which is a quadruple $G =< V, E, t_0, t_n >$ consisting of following elements:

- V is a finite set of nodes. Each vertex is associate with an executable atomic task $t_i, 0 \leq i \leq n$. To simplify the discussion and without loss of generality, we assume that tasks associated with different nodes are different.
- $E \subset V \times V$ is a set of edges connecting nodes in the graph. An edge, which is a pair $e =< t_i, t_j >\in E$, defines a precedent relation between two tasks. It means task t_j will start if and only if t_i completes successfully.
- $t_0 \in V$ is the starting node of the graph.
- $t_n \in V$ is the ending node, signifying the completion or termination of the workflow.

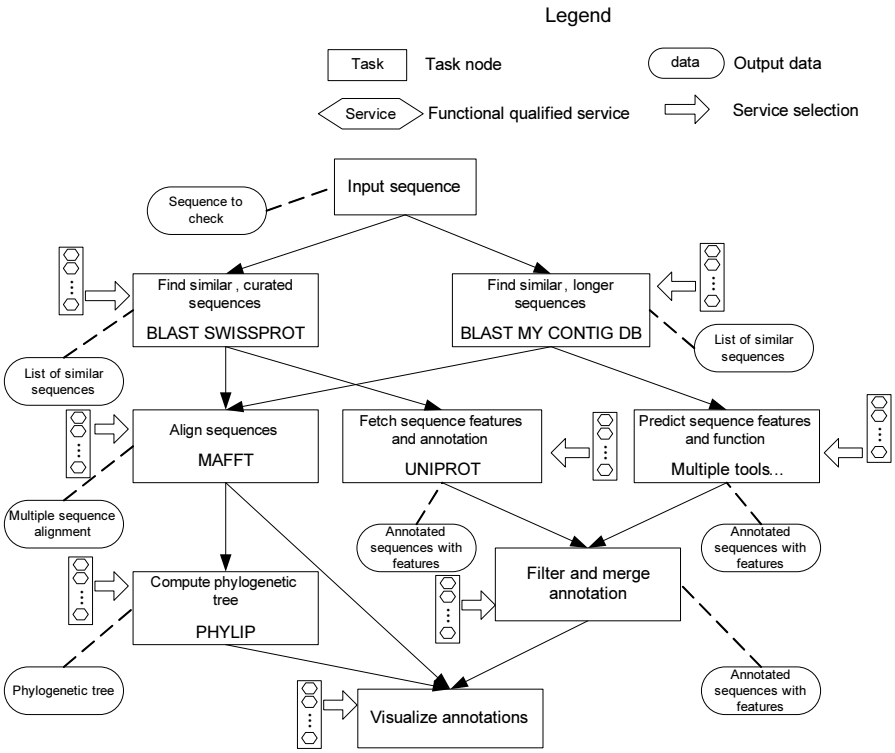


Figure 1. Example of bioinformatics workflow

It is required that each workflow has only one entry point, and completes at a single end node. The DAG definition for a workflow in fact defines a partial order among tasks.

To simplify the discussion, we assume workflow graphs are well-structured [8] in the sense that it consists of matching pairs of a node that splits the flow and a node that joins the flow. [8] proves with corresponding transforming algorithms that all workflow graphs can be converted to be well-structured.

Definition 2 (Precedence relation): The edge $e =< t_i, t_j >\in E$ in a DAG defines a precedence relation between tasks, denoted as $t_i \rightarrow t_j$. Task t_j is the successor of t_i, and t_i is the predecessor of t_j.

$\rightarrow^*$ is the reflexive and transitive closure of $\rightarrow$ and that this closure defines a partial order.

Definition 3 (Execution path): In a DAG, a set of tasks with the relation of $(t_i \rightarrow t_{i+1}) \wedge (t_{i+1} \rightarrow t_{i+2}) \wedge ... \wedge (t_{j-1} \rightarrow t_j)$ forms an execution path $\pi = (t_i, t_{i+1}, t_{i+2}, ..., t_j)$. If the path between t_i and t_j is unique, it is denoted as $\pi_{i...j}$.

3.2. Basic risk evaluation model

Risk assessment, which evaluates quantitative or qualitative value of risk related to a concrete situation and a recognized threat, is a common practice in business and social management. Our previous paper [5] has explained the concept of the cost of failure to customize the usage of risk assessment for general workflow structure. This section provides a brief introduction to the model. Refer to [5] for the details of the definitions in this section.

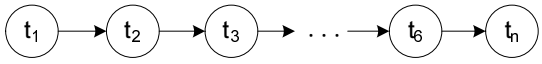


Figure 2. Sequential execution of tasks

Definition 4 (Cost of failure): The risk for performing a certain activity or decision is evaluated as the cost of failure, which is the multiplication of the potential loss L, and the probability p that the loss will occur. Formally, $cost_f = Lp$.

For a given execution path as shown in Figure 2, failure of any task on the path will cause the whole sequence to fail. Definition 5 and 6 shows the formula to compute the cost of failure for the sequence, which helps developers to evaluate the risk and consequences of task failure.

Definition 5 (Failure cost of a task): In execution path $\pi_{1...n} = (t_1, t_2, \ldots, t_n)$, if each $t_i (1 \leq i \leq n)$ has the probability $p(i)$ to be completed for the price

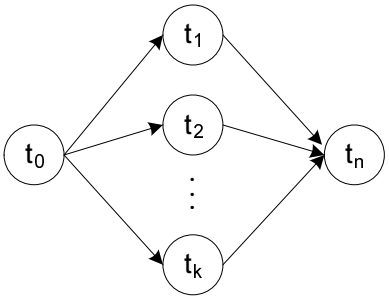


Figure 3. Concurrent execution of tasks

of $cost(i)$, then the failure of the k^{th} task will cost

$$cost_f^{t_k} = \prod_{i=1}^{k-1} p(i)(1-p(k)) \sum_{j=1}^{k} cost(j)$$

Definition 6 (Failure cost of a path): In execution path $\pi_{1...n} = (t_1, t_2, \ldots, t_n)$, if each $t_i (1 \leq i \leq n)$ has the probability $p(i)$ to be completed for the price of $cost(i)$, then the cost of failure for the path is

$$\begin{aligned} cost_f^{\pi_{1...n}} &= \sum_{k=1}^{n} cost_f^{t_k} \\ &= \sum_{k=1}^{n} \Big(\prod_{i=1}^{k-1} p(i)(1-p(k)) \sum_{j=1}^{k} cost(j) \Big) \end{aligned}$$

Compared with sequential task execution, concurrent tasks, as shown in Figure 3, are triggered at the same time. Definition 7 provides the method to compute the cost of failure for parallel running tasks.

Definition 7 (Failure cost of concurrency): $\|_{t_1,...,t_k}$ represents that $t_1, t_2, \ldots, t_k$ is a group of tasks running concurrently, satisfying $(t_0 \rightarrow t_1) \wedge (t_0 \rightarrow t_2) \wedge ... \wedge (t_0 \rightarrow t_k)$ and $(t_1 \rightarrow t_n) \wedge (t_2 \rightarrow t_n) \wedge ... \wedge (t_k \rightarrow n)$. If each task $t_i (1 \leq i \leq k)$ has the probability $p(i)$ to be completed for the price of $cost(i)$, then the cost of failure for their concurrent execution is

$$cost_f^{\|_{t_1,\ldots,t_k}} = \sum_{i=1}^{k} cost(i) \big(1 - \prod_{i=1}^{k} p(i)\big)$$

Applying the concept of graph reduction and sub-workflow substitution [9], the cost of failure for the workflow, denoted as $cost_f^G$, can be computed recursively with Definition 6 and 7. The details and algorithms to compute $cost_f^G$ are provided in [5].

3.3. Risk evaluation with backup service

In this section, we first introduce a data backup service $DBS(i)$, which is in charge of saving the output of task t_i for future usage. According to most cloud storage service providers, such as Amazon and Google, the service is charged for both the storage space and the outwards network usage. Therefore, let $vol(i)$ be the size of output by t_i, $price_s$ be the unit price for the storage, $price_{out}$ be the unit price for outwards network transfer, then

- For the backup process, the cost is $cost(backup_i) = vol * price_s$;
- For the retrieving process, the cost is $cost(retrieve_i) = vol * price_{out}$.

If $DBS(i)$ is activated, the backup process will be carried out accordingly. However, the retrieving process will only be triggered in case of failure of t_i's descendants.

The basic risk evaluation model in Section 3.2 does not consider the use of data backup service, therefore, the cost of failure needs to be calculated from the start node each time. In comparison, with the backup service, the retry of the workflow can retrieve the saved intermediate results and continue thereafter.

Definition 8 (Retry sphere): Retry sphere, denoted as RS_j, is a set of tasks which need to be re-executed in case of retrying a failed task t_j. Let $saved(t_x)$ stand for that the result of t_x has been saved with backup service $DBS(x)$. Formally, $RS_j = \{t_j\} \bigcup \{t_i | [(t_i \rightarrow^* t_j) \wedge \neg saved(t_i) \wedge [\neg \exists t_k (t_i \rightarrow^* t_k) \wedge (t_k \rightarrow^* t_j) \wedge saved(t_k)]]\}$.

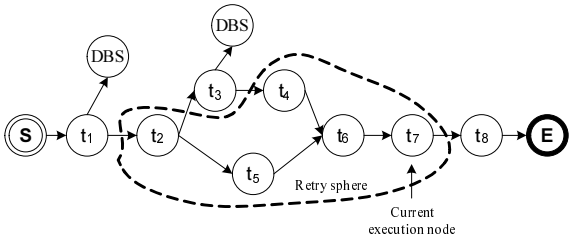


Figure 4. Example for retry sphere

Figure 4 shows an example for retry sphere RS_7. In the example, there are two paths to t_7, $t_2 \rightarrow t_3 \rightarrow t_4 \rightarrow t_6 \rightarrow t_7$ and $t_2 \rightarrow t_5 \rightarrow t_6 \rightarrow t_7$ respectively. According to Definition 8, the retry sphere for the current running task t_7, marked inside the dotted line, can be specified.

All tasks in the retry sphere are required for re-execution since their outputs are not stored. However, tasks outside the sphere have no impact on the retry of the failed task since their outputs can be retrieved from the storage service directly without re-execution. Although backup services can avoid the re-execution of the corresponding tasks, they come with the extra overhead of data storage and network transfer. In some

cases, the overhead may outweigh the re-execution cost, especially when each service has a high probability of success. Therefore, we apply the concept of cost of failure to find the optimal decision on whether to perform backup for the current running task or not.

Theorem 1: Let t_i be the completed task and t_j be its child. If the output of t_i is not saved, the cost of failure for the retry of t_j is $cost_f^{subG(i)}$. $subG(i)$, consisting of RS_i and t_j, is the subgraph of the original workflow G.

Proof: After the completion of t_i, since $DBS(i)$ is not performed, all tasks inside its retry sphere RS_i need to be re-executed if t_j fails. Therefore, these tasks are part of the re-execution subflow.

If t_j fails, itself will be retried. Therefore, t_j is also included in the re-execution subflow.

In summary, the cost of failure for retrying t_j is that of the subgraph including $RS(i)$ and t_j. □

Theorem 1 shows that after the completion of the current task t_i, if the output of t_i is not saved, the failure of its child will lead to the re-execution of all tasks inside the retry sphere of t_i as well as itself. As shown in Figure 4, if the output of t_7 is not saved, the retry of t_8 contains a task set of $\{t_2, t_4, t_5, t_6, t_7, t_8\}$. A dummy task can be inserted as a splitting point for t_2 and t_4 to convert the subgraph to be well-structured DAG. Therefore, the cost of failure in this situation comes from the cost of failure for the subgraph contains the retry set. The details computation process can be referred to [5].

Theorem 2: Let t_i be the completed task and t_j be its child. If the output of t_i is saved, the cost of failure for the retry of t_j is $cost_f^{DBS(i)} = cost(backup_i) + (1 - p(j))cost(retrieve_i)$.

Proof: After the completion of t_i, since $DBS(i)$ is performed, all tasks before t_i are no longer need to be re-executed. However, the overhead of running service $DBS(i)$, which is $cost(backup_i)$, is required no matter t_j is retried or not. Moreover, since the cost for retrieving saved data is $cost(retrieve_i)$ and the probability to retrieve is $1 - p(j)$, the cost of failure for retrying t_j equals to $cost(backup_i) + (1 - p(j))cost(retrieve_i)$. □

According to Theorem 2, if $DBS(i)$ is performed, all tasks prior to t_i are no longer needed and the risk are diverted into $DBS(i)$. As shown in Figure 4, if the output of t_7 is saved, the cost of failure to retry t_8 is $cost(backup_7) + (1 - p(8))cost(retrieve_7)$, which is highly dependent on the volume of output by t_7.

With Theorem 1 and 2, the algorithm in Figure 5 can be applied to decide whether to perform backup service after the completion of the current task.

Require: G - DAG definition
Require: t_i - Current completed task
▷ Make optimal decision on performing backup or not
1: **function** DECISION_BACKUP(G, t_i)
2: do_backup = false
▷ Construct the subflow for possible retry
3: G_{sub} = GET_RS(G, t_i)
4: treat all children of t_i as parallel and substitute them as t_j
5: add t_j into G_{sub}
6: add dummy task into G_{sub} to make it DAG
7: compute $cost_f^{G_{sub}}$
8: compute $cost_f^{DBS} = cost(backup_i) + (1 - p_j)cost(retrieve_i)$
9: **if** $cost_f^{G_{sub}} >= cost_f^{DBS}$ **then**
10: do_backup = true
11: **end if**
12: return do_backup
13: **end function**
▷ Find the retry sphere of t_i
14: **function** GET_RS(G, t_i)
15: $RS = UnVisited = \{t_i\}$
▷ Graph search
16: **for all** $t_j \in UnVisited$ **do**
17: remove t_j from $UnVisited$
18: get $Parents = \{t_x | t_x \rightarrow t_j\}$
19: **for all** $t_k \in Parents$ **do**
20: **if** $saved(t_k) == false$ **then**
21: put t_k into RS and $UnVisited$
22: **end if**
23: **end for**
24: **end for**
25: return RS
26: **end function**

Figure 5. Functions for task substitution

3.4. Risk-aware service selection

In service-oriented environment, there are generally several services with different processing capacities eligible to complete each task in the workflow graph G. The selection of service s_i to execute task t_i results in a pair $< cost(i), p(i) >$, representing its execution cost and probability to succeed. Moreover, users may set the constraint of budget B as part of their business logic in the workflow definition.

After service assignments to each task in the workflow, the algorithm in Figure 5 can be applied to find the set of tasks that need backup service, denoted as TS_{dbs}. Thereafter, the original workflow graph are partitioned into a set of sub-graphs, denoted as GS_{sub}, by the elements in TS_{dbs}. Finally, multi-objective optimization parameters can be defined to judge the suitability of the service selection.

$$Cost(G) = \sum_{t_i \in G} cost(i) + \sum_{t_j \in TS_{dbs}} cost_f^{DBS(j)}$$
$$P(Suc(G)) = \prod_{t_i \in G} p(i)$$
$$cost_f^G = \sum_{g \in GS_{sub}} cost_f^g + \sum_{t_j \in TS_{dbs}} cost_f^{DBS(j)}$$

Maximize $P(Suc(G))$ and minimize $cost_f^G$ subject to:

$$Cost(G) < B$$

$Cost(G)$ is the total cost of the overall schedule, and must be lower than the budget. $P(Suc(G))$ indicates the probability that the solution will finally succeed. It is a product of $p(i)$, which indicates the probability that each task will succeed, because the failure of any task will cause the whole schedule to fail. We assume that each service's probability of success is independent. $cost_f^G$ is the cost of failure for the workflow. The value represents the risk of a schedule with respect to the possible loss.

Therefore, a cost-effective and robust scheduler should try to maximize $P(Suc(G))$ and minimize $cost_f^G$. To find an optimized schedule for this multi-objective scheduling problem, we adopt genetic algorithm (GA) to design and implement the scheduling process.

Typically, a GA contains two main problem-dependent components: the encoding schema and the evaluation function. The encoding scheme breaks a potential solution into discrete parts that can vary independently. In GA, the discrete parts are called "genes" and the solution is called a chromosome. For scheduling purpose, we treat each task in the workflow as a gene, and the vector of scheduled tasks as a chromosome.

The evaluation function, denoted as Ψ, measures the quality of a particular solution according to the given optimization objectives. Ψ contains two parts: the fitness function, denoted as $F(G)$, to encourage the higher robustness of the schedule, and the penalty function, denoted as $P(G)$, to enforce the constraints of budget B. In our definition, the higher the value of Ψ, the better the solution.

$$\begin{aligned} F_{cost}(G) &= 1 - Cost(G)/B \\ F_{risk}(G) &= cost_f^G/Cost(G) \\ F(G) &= (1 - F_{risk}(G)) \cdot P(Suc(G)) \cdot F_{cost}(G) \\ P(G) &= \begin{cases} F_{cost}(G) & \text{if } Cost(G) >= B \\ 0 & otherwise \end{cases} \\ \Psi &= F(G) + P(G) \end{aligned}$$

In $F(G)$, the multiplication of $P(Suc(G))$ and $F_{cost}(G)$ indicates that the saving on execution cost is only valuable if the schedule succeeds. The result is further multiplied with $(1 - F_{risk}(G))$ to ensure the portion of saving on execution time or cost is in the safety zone. $P(G)$ is designed to enforce the bias against ineligible schedules which exceed the budget. After the chromosome and the evaluation function are defined, standard GA operators can be applied to find the optimized schedule.

4. Experiments and evaluations

In this section, experiments were designed to evaluate reliable scheduling strategies with and without precise risk assessment. The environment for the experiments was created by adopting the DAG dataset of the Resource-Constrained Project Scheduling Problem [10].

In simulation, each task was supported with a set of different services with various cost and success probability levels (mean values). Assume the size of data for task t_i to process is VOL_i, which was randomly selected between 100 and 2000 (GB). To make the simulation more realistic, we assume that the cost of services for t_i is in direct proportion to its VOL_i. That is, $mean\_cost(t_i) = r_i \cdot VOL_i$, where r_i is a random number between $[0.1, 0.5]$. Moreover, the probability level was set to 1.0.

The exact value of cost and probability for each service were obtained by randomly fluctuating about 20% around the cost and probability level of its corresponding task. To make the simulated environment more practical, the fluctuation followed the observation that in commerce the success probability of a service is in direct proportion to the cost.

In addition, the size of output data for t_i is set to $vol_i = r_i \cdot VOL_i$ where output portion r_i is a random number between $[0, 1]$. To investigate the impact of output volume on system performance, we classify the experiments into three categories with different r_i groups, including $[0, 0.1]$, $[0, 0.2]$, $[0, 0.5]$. Referring to

Amazon and Google, the prices for data storage and outbound network usage are both set as $\$0.10/GB$.

To simulate different levels of user constraints, the budget B was evaluated at a wide range of possible values between the maximum and minimum allowed cost for the workflow. k was used for the constraint levels of B, which ranged from 0 to 1. The higher the value, the tighter the constraints.

$$B = maxCost - k(maxCost - minCost)$$

In the experiments, $maxCost$ and $minCost$ were obtained by always selecting the service with the most or least processing cost for each task respectively.

The implementation of the GA was based on [11]. The population for each generation was 50 and the total generation was 2000 for each experiment. All other parameters, such as crossover and mutation probability, were the system default.

The result was evaluated with two parameters. One was $P(Suc(G))$ for the probability to succeed, and the other was the risk ratio $cost_f^G/Cost(G)$ for the percentage of cost of failure in the total cost. $cost_f^G$ for both service selection strategies, with and without backup services, are computed and compared. The higher, the better for $P(Suc(G))$, but the lower, the better for risk ratio.

The experiments were carried out by increasing k from 0 to 1 by 0.1, representing constraint of B from the relaxed to the tight. Figure 6 shows the results about the correlation between B, $P(Suc(G))$, and risk ratio of schedules, in a randomly chosen DAG with 30 nodes. Altering the experiments by changing resource settings and DAG definitions yields similar results.

The results show that $P(Suc(G))$ and risk ratio get worse as the constraint levels become tighter for both strategies. However, service selection with backup services performs better, with respect to the probability of overall success and possible failure loss, especially when the output portion is small. The improvement relies on the running of backup service when it outperforms the re-execution overhead. As shown in the algorithm in Figure 5, when the output portion increases, the cost of failure for applying backup service becomes larger, making it less likely to happen.

5. Related work

Risk management is a common business practice, and should be part of QoS (Quality of Service) management. However, the common strategy in QoS for risk management is rescheduling or retrying alternative services [12], [13], neglecting the precise calculation of cost of failure for each decision. Our previous work [5] has proposed such a risk evaluation model by assigning different impact factors or weights to tasks based on their ordering relationship. However, the model is designed for general workflow application without considering the computing- and data-intensive features of scientific workflows. In comparison, this paper studies the impact of data backup service on system reliability and exception handling.

Data backup service works like checkpoint for workflows [14], which makes a snapshot of the execution state for future recovery. However, the location of the checkpoint usually needs to be decided manually. In comparison, this paper applies the concept of risk to measure the benefit of performing the backup, and make decisions automatically at runtime to achieve tradeoff between the overhead of backup itself and the potential loss.

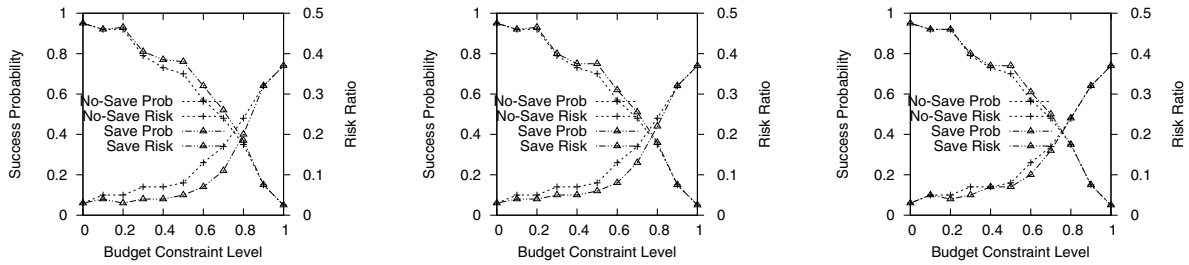
Our work can be categorized as a subset of data provenance, which aims to support reproducibility of data sets and processes. Most work in this area [15] focuses on the approach to capture, filter, query, and trace provenance data on demand. In contrast, we investigate the automatic decision mechanism for data backup of internal tasks.

6. Conclusion and future Work

In this paper, we extend the risk evaluation model [5] with the consideration of data backup service to balance the tradeoff between overhead of backup and re-execution after exceptions. The proposed method computes and compares the potential loss with and without data backup to optimally decide the occasion to perform backup service after the completion of a task. Thereafter, we design the utility function with the model and apply a genetic algorithm to find the optimized schedule, thereby maximizing the robustness of the schedule while minimizing the possible cost of failure.

The experiments show that application of backup service can generally improve system robustness and reliability without exceeding the customer constraints, such as cost and reliability requirements. Specifically, the proposed method performs better when the data generated by each task is not large.

For future work, we intend to evaluate and study the impact of various cost models from cloud providers on the application of the proposed method. We also plan to take more QoS parameters, such as deadline and responsiveness, into the service selection process. Moreover, we will design some dynamic scheduling algorithms to select the checkpoint in the ever-changing



(a) Output portion < 0.1 (b) Output portion < 0.2 (c) Output portion < 0.5

Figure 6. Cost of failure at variant output portion and constraint levels

service environment.

Acknowledgment

The research work reported in this paper is supported by National Science Foundation of China under Grant No. 61100172 and No. 61003262.

References

[1] X. Liu, W. Dou, J. Chen, S. Fan, S. C. Cheung, and S. Cai, "On design, verification, and dynamic modification of the problem-based scientific workflow model," *Simulation Modeling Practice and Theory*, vol. 15, no. 9, pp. 1068–1088, 2007.

[2] J. Yu, R. Buyya, and K. Ramamohanarao, "Workflow scheduling algorithms for grid computing," in *Metaheuristics for Scheduling in Distributed Computing Environments*, F. Xhafa and A. Abraham, Eds. Springer Berlin Heidelberg, 2008, vol. 146, pp. 173–214.

[3] M. Rahman, R. Ranjan, and R. Buyya, "Reputation-based dependable scheduling of workflow applications in peer-to-peer grids," *Comput. Netw.*, vol. 54, pp. 3341–3359, December 2010.

[4] M. Wang, K. Ramamohanarao, and J. Chen, "Trust-based robust scheduling and runtime adaptation of scientific workflow," *Concurrency and Computation: Practice and Experience*, vol. 21, no. 16, pp. 1982–1998, 2009.

[5] ——, "Dependency-based risk evaluation for robust workflow scheduling," in *2012 Workshop on Large Scale Distributed Service-oriented Systems (In conjunction with 26th IEEE International Parallel & Distributed Processing Symposium)*, May 2012.

[6] S. Pettifer, J. Ison, M. Kalas, D. Thorne, P. McDermott, I. Jonassen, A. Liaquat, J. M. Fernandez, J. M. Rodriguez, I. Partners, D. G. Pisano, C. Blanchet, M. Uludag, P. Rice, E. Bartaseviciute, K. Rapacki, M. Hekkelman, O. Sand, H. Stockinger, A. B. Clegg, E. Bongcam-Rudloff, J. Salzemann, V. Breton, T. K. Attwood, G. Cameron, and G. Vriend, "The embrace web service collection," *Nucleic Acids Research*, 2010.

[7] M. Weißbach and W. Zimmermann, "Termination analysis of business process workflows," in *Proceedings of the 5th International Workshop on Enhanced Web Service Technologies*. New York, NY, USA: ACM, 2010, pp. 18–25.

[8] J. Vanhatalo, H. Völzer, F. Leymann, and S. Moser, "Automatic workflow graph refactoring and completion," in *Proceedings of the 6th International Conference on Service-Oriented Computing*. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 100–115.

[9] J. Cardoso, A. Sheth, J. Miller, J. Arnold, and K. Kochut, "Quality of service for workflows and web service processes," *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 1, no. 3, pp. 281 – 308, 2004.

[10] Project Scheduling Problem Library - PSPLIB, accessed on 28-May-2012. [Online]. Available: http://129.187.106.231/psplib/

[11] K. Meffert, N. Rotstan, C. Knowles, and U. B. Sangiorgi, JGAP - Java Genetic Algorithms and Genetic Programming Package, accessed on 28-May-2012. [Online]. Available: http://jgap.sourceforge.net/

[12] E. Deelman, D. Gannon, M. Shields, and I. Taylor, "Workflows and e-science: An overview of workflow system features and capabilities," *Future Generation Computer Systems*, vol. 25, no. 5, pp. 528 – 540, 2009.

[13] L. Qi, W. Lin, W. Dou, J. Jiang, and J. Chen, "A QoS-aware exception handling method in scientific workflow execution," *Concurrency and Computation: Practice and Experience*, vol. 23, no. 16, pp. 1951–1968, 2011.

[14] R. Duan, R. Prodan, and T. Fahringer, "DEE: A distributed fault tolerant workflow enactment engine for grid computing," ser. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2005, vol. 3726, pp. 704–716.

[15] T. Dalman, M. Weitzel, W. Wiechert, B. Freisleben, and K. Noh, "An online provenance service for distributed metabolic flux analysis workflows," in *Ninth IEEE European Conference on Web Services (ECOWS)*, 2011, pp. 91 –98.