Int J Adv Manuf Technol (2016) 84:119–131
DOI 10.1007/s00170-015-7804-9

ORIGINAL ARTICLE

A scientific workflow management system architecture and its scheduling based on cloud service platform for manufacturing big data analytics

Xiu Li[1] · **Jingdong Song**[1] · **Biqing Huang**[2]

Received: 27 May 2015 / Accepted: 2 September 2015 / Published online: 11 September 2015

Abstract To improve the efficiency and productivity of modern manufacturing, the requirements for enterprises are discussed. The new emerged technologies such as cloud computing and internet of things are analyzed and the bottlenecks faced by enterprises in manufacturing big data analytics are investigated. Scientific workflow technology as a method to solve the problems is introduced and an architecture of scientific workflow management system based on cloud manufacturing service platform is proposed. The functions of each layer in the architecture are described in detail and implemented with an existing workflow system as a case study. The workflow scheduling algorithm is the key issue of management system, and the related work is reviewed. This paper takes the general problems of existing algorithms as the motivation to propose a novel scheduling algorithm called MP (max percentages) algorithm. The simulation results indicate that the proposed algorithm has performed better than the other five classic algorithms with respect to both the total completion time and load balancing level.

Keywords Manufacturing big data · Scientific workflow · Scheduling algorithm · Cloud service platform

✉ Jingdong Song
songjingdong.thu@gmail.com

[1] Division of Information Science & Technology, Graduate School at Shenzhen, Tsinghua University, Shenzhen 518055, China

[2] Department of Automation, Tsinghua University, Beijing 100084, China

1 Introduction

Improving the efficiency and productivity is a great challenge to manufacturing industry in the current economic globalization. The optimization of manufacturing processes plays an important role in enhancing the competitiveness of enterprises. However, that requires deeper analysis of various data and a large number of computing experiments in the entire life-cycle of a product. Fortunately, we are embracing the big data environment that provides us the foundation of data analysis and the possibility to discover the optimal strategy. Under the "Internet of Things" (IoT) ideology, we adopt smart sensors technologies such as RFID technologies and wireless sensor network technologies to collect useful data ubiquitously [1]. Cloud computing which aims at providing the computer storage and computation as services to help the enterprises solve the problem of limited local resources during the process of big data analytics [2]. But many enterprises are still suffering the deficiency of the capabilities of efficient storage, management and utilization of big data, and the techniques of advanced computing and experiment management. They cannot excavate useful information from big data and take full advantage of existing tools or resources to improve the efficiency of scientific discoveries. Therefore, a tool or platform that intends to exploit the cloud or other distributed service recourses, handle massive amounts of data, and manage the complex procedures of different data analytics is required.

The analysis and utilization of manufacturing big data mentioned above requires a lot of complex experiments and a data processing application may consist of multiple computation steps and dependencies within them. In one application, it may need the same one or more computation steps in another application. Scientific workflow is very suitable and convenient to model such process and express the entire data processing steps and dependencies for the users [3]. A

scientific workflow is similar to the general workflow, which organizes a set of tasks in an order according to their dependent relations, but it is data-oriented not control-oriented which means it pays more attention to the intensive operations on data sets, and the dependencies emphatically describe the flow of data streams [4]. The technology of scientific workflow has been successfully introduced in the scientific fields such as bioinformatics, genetics, astrophysics, and geophysics which provide an environment to aid the scientific discovery process through the combination of available tools for scientific data management, analysis, simulation, and visualization [5]. Obviously Scientific Workflow Management System (SWfMS) is a tool to execute workflows and manage data sets. In modern scientific environment, scientific workflow becomes data-intensive, which can enable the parallel execution to process large-scale data onto distributed resources [3]. Besides, cloud computing which easily provides the distribute hardware and software resources as visual services for scientific researches and engineering applications can also be combined with the scientific workflows [2]. Within cloud environments, we can encapsulate resources and complex collecting, filtering, computing, visualizations, and other processing steps into different services to be orchestrated as a model of scientific workflows to be managed to process the manufacturing big data [6].

Our previous work [7] has studied the manufacturing information modes and technologies and proposed a cloud manufacturing service platform which integrates the technologies such as cloud computing and internet of things to arrange the manufacturing resources and provides user services. That paper focuses on solving the uncovered problems when the manufacturing industry transfers from the traditional product-oriented type to service-oriented type but not care much about the data processing management. In this paper, a scientific workflow management system based on our previous cloud platform is introduced as an efficient tool to formalize and structure complex and distributed scientific processes to enable and accelerate scientific discoveries for the future manufacturing [8]. The scientific workflow management system can invoke the cloud resource services and combine the new and distributed programming paradigms. Then we also propose a prototype system architecture of it.

Workflow scheduling algorithms are crucial to the efficiency of workflow management systems. Many scientific workflow management systems which have been employed in some research projects own their specific scheduling algorithms [9, 10]. Although each algorithm can present a good performance, it just appears in its own suitable situation. And some algorithms can evidently shorten the completion time of scientific workflows, but cannot balance the load well. Some algorithms perform well both in the completion time and load balance, however the average scheduling time of those algorithms are intolerable. It is necessary to find a scheduling algorithm to overcome the shortcomings above and have a good comprehensive performance. Given this motivation, a better scheduling algorithm called MP algorithm is designed, which can meet various requirements in most of the types of scientific workflows. And the MP algorithm can far outperform other existing ones particularly in some task unbalance situations. This algorithm takes all the information about workflows and heterogeneous resources into consideration to ensure the load balance, and uses Min-Min and Sufferage algorithms for reference to make the total completion time short.

The rest of this paper is structured as follows. In the next section, we introduce the cloud manufacturing service platform. In the section 3, the related works of existing workflow management system are outlined and we propose a prototype system architecture. Section 4 gives an overview of scheduling algorithms, then establishes the problem description models and makes some necessary definitions. In Section 5, our MP scientific workflow algorithm is discussed. Experimental details and simulation results are described in Section 6. Finally, Section 7 closes the paper with directions for further work.

2 Cloud manufacturing service platform

Cloud manufacturing is a new computing and service-oriented manufacturing mode based on the existing advanced manufacturing models such as networked manufacturing (NM) [11], application service provider (ASP) [12]. With the help of new information technologies of cloud computing, cloud security, and IoT, various manufacturing resources and abilities can be automatically sensed, managed, and encapsulated into different cloud services for sharing. And users can search and invoke the specific cloud services to meet their needs. Xun [13] discussed some of the essential features of cloud computing and pointed out that it will be one of the major enablers for the manufacturing industry that can transform its business models, help it align product innovation with business strategy, and create intelligent factory networks that encourage effective collaboration. Tao et al. [14] summarized the applications of IoT and cloud computing in manufacturing field among three usage scenarios: in the workshop, enterprise; and enterprises and proposed a cloud computing- and IoT-based cloud manufacturing (CMfg) service system. And they also studied the method of optimal allocation of computing resources in cloud manufacturing systems [15]. In [7], a cloud manufacturing service platform composed of 12 layers is proposed. It has the characteristics of the cloud manufacturing mode and the paper gives a full function description of each layer. The platform focuses on realizing the intelligent embedded access of underlying physical manufacturing resources as well as service encapsulation of manufacturing

resources and abilities. Layers of implementing the above functions are discussed in detail while the layers about the big data processing and resources scheduling are not involved much. Although the concept of workflow management is mentioned in the cloud manufacturing system, it still falls within the field of business workflow. The platform suffers from the deficiency of using scientific workflow to analyze the manufacturing big data.

3 Scientific workflow management systems

Scientific workflow which is concerned with the automation of procedures usually makes a collection of scientific tasks streamlined to enable and accelerate scientific discoveries [8].The most significant advantage of using scientific workflow is that it can provide easy access to many complicated computing technologies without knowing the underlying knowledge. It presents every analytical step in a visual way, and we can reuse part or all of a workflow with little effort. Scientific workflow management systems define, manage, and execute the workflows in various computing environments. With the development of sensor technologies, communicating between machines and collecting data has become easy in manufacturing [16], so how to use and manage the big data to support better decisions is an enormous challenge for the manufacturing information system. In this section, some related work and a typical architecture of scientific workflow management system are discussed.

3.1 Various SWfMSs

Scientific workflow management systems have been studied for many years and several SWfMSs are now used intensively by various research communities [3]. Kepler [17] is an open-source SWfMS which is developed by UC Berkeley and San Diego Supercomputer Center (SDSC) and has been widely used in many scientific domains such as astronomy and biology. Kepler as shown in Fig. 1 provides an intuitive graphical user interface to design and present workflows in the work area called canvas, and scientists can just simply drag and drop the specific modules into the canvas to create their own executable scientific workflows which make scientists or others with little background in computer science can understand the execution of a workflow easily[18]. Taverna is also an open-source SWfMS that is mainly used in biology and bioinformatics. It adopts a textual language SCUFL (Simple Conceptual Unified Flow Language) [19] to define and present the workflows which can be easily used locally or remotely. Galaxy is different with Kepler and Taverna, which is a web-based SWfWS and especially for genomic research. Besides the above, Pesasus [20], ASKALON [21], Java Cog Kit [22], P-GRADE [23], Grid Flow [24], K-WF [25] are all fairly mature SWfMSs and they all have their own suitable situations and workflow management methods.

3.2 The architecture of scientific workflow management systems

An excellent SWfMS must meet the following requirements: friendly user interaction, software reuse, supporting parallel computing, heterogeneous and distributed resources management, fault tolerance, collaborative sharing, and other necessary features. Liu et al. [3] gave a five-layer architecture of SWfMSs, e.g., presentation layer, user services layer, WEP (Workflow Execution Plan) generation layer, WEP execution layer, and infrastructure layer. Lin et al. [8] proposed a reference architecture of SWfMSs which composes four layers, namely, presentation layer, workflow management layer, task management layer, and operational layer. Zhaoet al. [6] devised a cloud scientific workflow platform for big data, and the framework includes client layer, service layer, middleware layer, and infrastructure layer. We reference the related work discussed above, and propose a prototype architecture of scientific workflow management system based on the cloud manufacturing service platform (as shown in Fig. 2), but it can also be extended to other research areas. We also divide the system into four layers which are application layer, service layer, management layer, and infrastructure layer. In the picture, we list only some of the main features of each layer. The first three layers can be seen as the integration and improvement of the last five layers of the previous cloud manufacturing service platform, and the fourth layer includes a brief expression of cloud computing services of the previous platform.

Our workflow management system is oriented to scientific institution projects and personal researches. In the application layer, they can construct their workflows and submit their requirements to the system by textual or graphic user interfaces. Modular programming provides the flexibility and reusability of local resource components and cloud service components. Then the workflows will be translated into predefined mathematical models that can be easily dealt with in the next layer. In the future manufacturing, we can customize several workflows such as production health assessment or other signal processing applications. Besides, the application layer also provides many presentation and visualization functions for the computing results.

The execution of scientific workflows is done in the service layer and it is also responsible for the monitoring and fault tolerance of scientific workflows to guarantee the workflow management system to function well. The service layer receives the requirements from the upper layer and scheduling strategies form the lower layer. The separation of workflow execution from the task acquisition and scheduling can improve the stability and scalability of the management system.

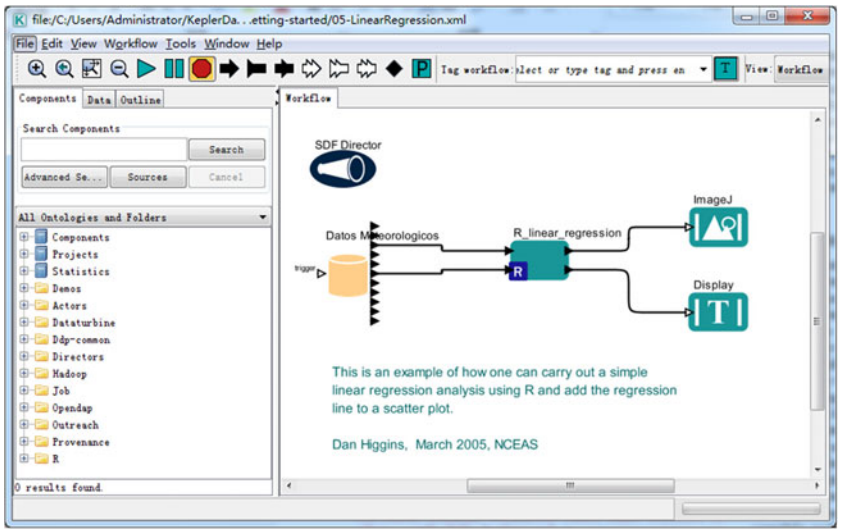

Fig. 1 The workbench of Kepler workflow management system

The management layer is the bridge of physical resources and workflow execution. The key part of the management layer is the scientific workflow scheduling module that decides which scheduling algorithm to be taken and provides the specific scheduling strategies to perform. It plays a major role in paralleling data processing and optimizing the task procedures. Before it makes a scheduling plan, it should obtain the applications and resources information about the system information services as the basis. During the scheduling there also may involve data movements which need an efficient data management module. Besides, provenance management can record the derivation history of data productions to track the experiment processes and validate the experiment results.

The infrastructure layer expands the forgoing services platform which was designed totally based on the cloud

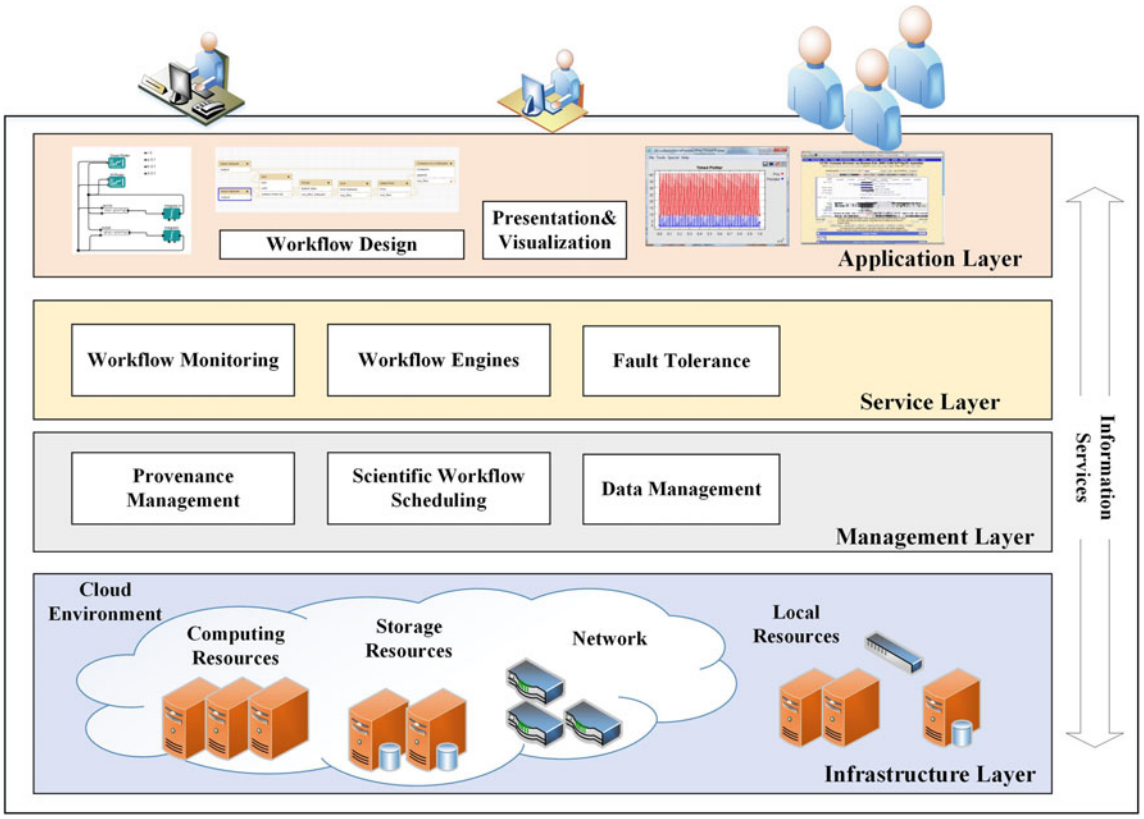

Fig. 2 The functional architecture of scientific workflow management system

environment by adding the local resources. The utilization of local resources can provide more service selections and save the service costs. Each computing environment includes computing, storage, network, and other resources. Smart sensors installed in the manufacture instruments extract the various signals such as temperature and pressure, and transform the signals by communication protocols, such as MTConnect [26] and OPC (Object Linking and Embedding (OLE) for Process Control). Then manufacturing big data is collected and stored in this layer. Before we use those computing or storage resources, we virtualize all the heterogeneous resources and encapsulate them into separate services, then provide interfaces to link the physical resources with the executor module in the service layer. All information about virtual resources we collect is to help formulate the scheduling strategies and the interfaces realize the interoperation between programs and physical resources.

3.3 Architecture implement

To implement a scientific workflow management designed as the architecture talked above, we can construct it all by ourselves from the bottom to the top, or we can make a secondary development on an existing system. Kepler as shown above is a promising workflow system to meet all requirements to enable the manufacturing big data analytics. Kepler adopts the actor-oriented programming, and each actor can provide an independent function, such as remote data access, Web services invocation. An actor can be atomic or composite which are collections or sets of atomic and/or composite actors bundled together to perform more complex operations. Kepler has a powerful function library which is composed of more than 530 actors. Besides, we can design the custom actor to achieve the desired functions by composing exist actors or Java programming language. Also, Kepler provides a special type of entities, called a director, to manage the execution of linked actors [27]. It can be synchronous, parallel, or other ways. Nowadays, many studies focus on the combination of Kepler and data parallel computing in big data environment, and Kepler has added the Mapreduce actor to integrate the Hadoop framework to facilitate data-intensive applications [28]. Spark [29], as the next generation of big data processing and analysis tool, has carried out some attempts on the Kepler platform. What' more, it is very convenient to share the workflows which designed by one or several scientists with others around the world.

For Kepler has a friendly GUI and the abilities of big data processing, Web services invocation, and remote resource access, we use the Kepler as the basis of our system, and what we need do to make it be suitable for the manufacturing analysis environment is to develop the specific Kepler workflow suit which includes the actors to deal with the professional manufacturing big data analysis issues by ourselves. And we also need to present and implement the professional analysis algorithms in the workflow way. The Kepler[1] project has told us the methods of building the custom actors. As a case study, the product performance degradation assessment and prediction are crucial to the management of production costs, lead time, and optimal machine utilization [30], and Fig. 3 illustrates the requirements for Kepler development. The Feature Extraction_A and Feature Extraction_P are two actors responsible for the performance degradation assessment and prediction respectively, and can be parallelized. They all receive the same sensor data, but extract the different features. The specific methods used in the actors should be chosen or developed by ourselves.

4 Scheduling problem description

Scheduling of scientific workflows decides the executive arrangement of tasks which will be mapped on heterogeneous and distributed resources. And the perfect strategy can make tasks allocated to the suitable resources and check the best execution sequence of parallel tasks. Generally we just research and consider the deterministic abstract workflows, for which the dependencies of tasks, the sizes of each task, the sizes of the output files of each task, the computing capabilities and communication rates of resources are known in advance, but we do not care about the physical locations of resources [2]. What we input to the algorithm is only the abstract mathematic model. During the process of scheduling, we represent a workflow as a Directed Acyclic Graph (DAG) [31], whose nodes represent tasks and edges represent the dependent relations among the tasks [32]. As the problem of DAG scheduling belongs to a class of NP-complete problem, scientists have proposed various heuristics and meta-heuristics algorithms without strict mathematical justification. Some schedules [9, 33–35] also include the aspects of the execution cost and QoS of resources in utility cloud environments where users should pay for what they obtain. In this section, we will present some of the mature best-effort based algorithms which are usually used on enterprise private clouds which are built for companies themselves to make the best to deal with data protection and services or public clouds which focus on sharing the resources, promoting scientific researches, and freely opening to the public not commercial interests.

The core idea of Min-Min heuristics algorithm is that it prefers to execute the minimum one of parallel tasks on the resource which cost the minimum time [36, 37]. It can usually minimize the total completion time of workflow, but it cannot

[1] Kepler project: https://kepler-project.org/

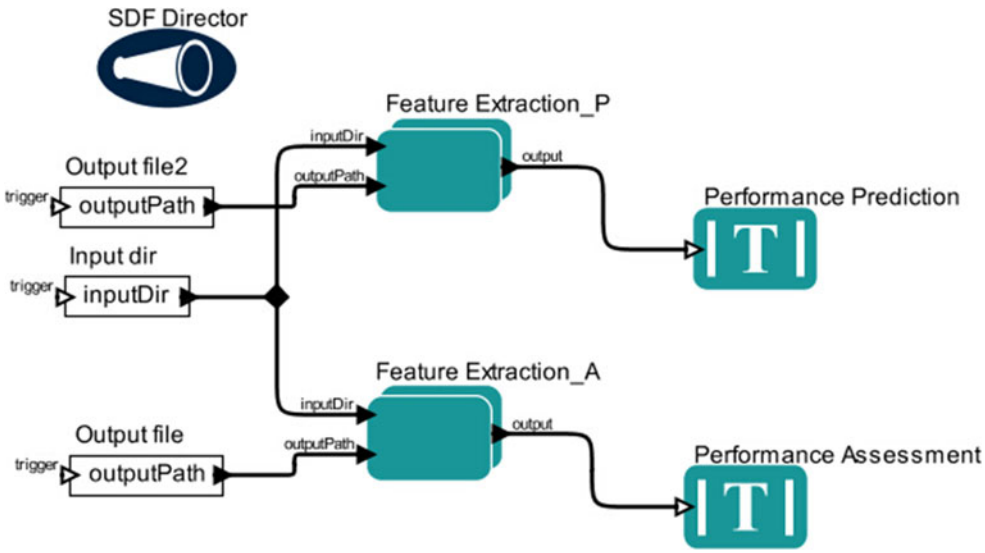

Fig. 3 A workflow for product performance degradation assessment and prediction in Kepler

guarantee the load balance when some of the resources have a considerable advantage in computing capability over other ones. The Max-Min [38] workflow scheduling algorithm is similar to the Min-Min algorithm, but it schedules large tasks first. Compared with the Min-Min algorithm, the Max-Min can show a well load balancing level and minimize the total completion time in most types of scientific workflows. Nevertheless when the number of long tasks is much more than that of the short tasks, the Min-Min algorithm is the better choice to generate the scheduling strategy. The Sufferage [39] heuristic algorithm focuses on the max time lose on heterogeneous resources. It defines the difference between the expected completion time on the best resource and the second best resource as the sufferage value, and gives the task with the biggest sufferage value the highest priority to execute among all available tasks. It can outperform Min-Min and Max-Min when there are large variations between resources, but [40] has proven that the sufferage is not suitable for data-intensive applications where many files may be reused many times. The DCP (Dynamic Critical Path) algorithm defines the critical path and priority tasks in the path. The algorithm will cost more time than other algorithms above when they have scheduled all the tasks. And when the algorithm is firstly introduced in [32], it is employed to dealing with homogeneous processors not for heterogeneous resources like cloud environment. GA (Genetic Algorithm) [41, 42] belongs to meta-heuristic algorithms, and it uses a larger space search to find the optimal solution. In most cases, GA can give us a high-quality solution and outperform than other algorithms. However, it pays for the advantage with a large time cost. What is worse, sometimes the algorithm may fall into the local optimal solution when the fitness function in the phase of selection is not well chosen.

4.1 Task abstract description

As the description above, scientific workflows are represented as DAGs, and the nodes of DAGs compose the set of tasks, $T = \{t_i | 1 \le i \le N\ (T)\}$ where $N\ (T)$ is the number of tasks. The directions of edges decide the dependencies between the tasks. The tasks at the beginning of edges are the parent tasks, and the end tasks of edges are called child tasks. So there is a natural constraint that the child tasks cannot start its execution until the system finishes its parent tasks. If a task does not have any predecessors, we call it an entry task. Correspondingly, we call a task without successors an exit task [43, 44]. The values of each node and edge are the length and size of the output of each task respectively. These are two important parameters to affect the scheduling results. An example of DAG is shown in Fig. 4, the specific units of these parameters will be discussed in section 6.

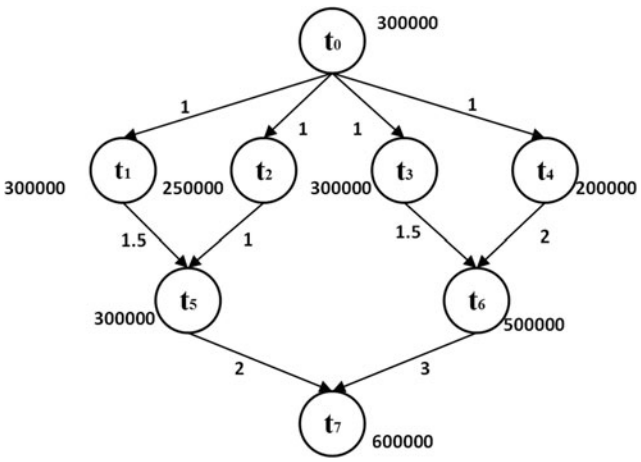

Fig. 4 A typical workflow presented in DAG

4.2 Resource abstract description

Compared with the practical cloud environments, we make several simplifying assumptions. We just consider the computing resources and assume other works before scheduling have been done. The main target of our scheduling algorithm is to map tasks on their suitable computing resources to obtain superb execution performance. Let $R=\{r_j|1\le j\le N(R)\}$ represents the set of computing resources, where $N(R)$ is the number of resources. For each resource, we introduce two attributes: processing capability (PC_j) and bandwidth (BW_j) which means the transfer capability. And we constrain that a resource can only execute one task at a time.

4.3 Definition

Before the introduction of the proposed algorithm, we define some variables to help understand as follows:

Definition 1 the execution time EXT (t_i, r_j)

EXT (t_i, r_j) is the total cost time of task t_i when it is executed on resource r_j, which can be computed by the following equation:

$$\mathrm{EXT}(t_i, r_j) = \frac{\text{The size of } t_i}{\mathrm{PC}_j} \tag{1}$$

Definition 2 the available time of resource ART (t_i, r_j)

ART (t_i, r_j) is defined as the earliest available time when if task t_i has been mapped on resource r_j, the resource r_j is available for task t_i without executing other tasks which are mapped on this resource prior to task t_i or transferring output files of the previous task.

Definition 3 the available starting time of task ATT (t_i, r_j)

ATT (t_i, r_j) is similar to the ART (t_i, r_j), but it means the earliest available time of task t_i when the parent tasks of task t_i are all executed.

Definition 4 the ready time RT (t_i, r_j)

RT (t_i, r_j) represents the time when the task t_i and resource r_j are all available and task t_i can be executed on resource r_j right now. And it can be computed through the following equation:

$$\mathrm{RT}(t_i, r_j) = \max\{\mathrm{ART}(t_i, r_j), \mathrm{ATT}(t_i, r_j)\} \tag{2}$$

Definition 5 the completion time CT (t_i, r_j)

CT (t_i, r_j) is defined as the finish time when task t_i has been executed on resource r_j. And it can be computed through following equation:

$$\mathrm{CT}(t_i, r_j) = \mathrm{EXT}(t_i, r_j) + \mathrm{RT}(t_i, r_j) \tag{3}$$

Table 1 Key code of MP algorithm

```
1. Traverse all the resources and find out the resource which has the best processing capability. Then record the value of the best processing capability as the PC;
2. While ∃t ∈ T is not scheduled do
3. availT ← get a set of unscheduled ready tasks whose parent tasks have been completed
4. Schedule ( availT)
5. Delete the tasks which have been scheduled
6. end while
7. PROCEDURE: Schedule ( availT )
8. While ∃t ∈ availT is not scheduled do
9. for all t ∈ availT do
10. availR ← get available resources for t
11. for all r ∈ availR do
12. calculate EXT (t, r), RT (t, r), CT (t, r), and p (t, r)
13. end for
14. end for
15. Ts, Rs ← arg max p (t, r) // get the task with maximum p (t, r) and find out the task and the resource associated
16. schedule Ts on Rs
17. remove Ts from availT
18. end while
```

Definition 6 the percentage of task p (t_i, r_j)

p (t_i, r_j) is the evaluation criteria to select which resource the task t_i should be mapped on. It can be obtained using the follow equation:

$$p(t_i, r_j) = \frac{\mathrm{EXT}(t_i, r_j)}{\mathrm{CT}(t_i, r_j)} \times \frac{\mathrm{PC}_j}{\mathrm{PC}} \tag{4}$$

where the PC is the max one of all PC_j, $1\le j\le N(R)$.

5 Proposed algorithm

5.1 Algorithm description

Reviewing all the algorithms described above, each of them has their own suitable situations. The MP algorithm we proposed has a bright future of combining all advantage of all

Table 2 The initial CT (t, r) matrix

	r_0	r_1
t_1	3	6
t_2	4	8
t_3	15	30

Table 3 The initial p (t, r) matrix

	r_0	r_1
t_1	1	0.5
t_2	1	0.5
t_3	1	0.5

algorithms. We consider the information of all tasks and resources dynamically to obtain the better load balancing level and scheduling strategy. The core idea of our algorithm is to search the resource which is affected most on its service time if one task was mapped on it, and then we allocate this task to it. We use the percentages of the completion time of each available task occupying the total used time of each resource to measure the effects on resources, and the task which has the max percentage will be scheduled to be executed. The definition of available tasks is the same as that of other algorithms whose parent tasks have been executed and are mutually independent. The detail of the MP algorithm is given in Table 1.

In step 15, there may be one or several tasks have the same p (t, r) on different resources, we schedule the task on the resource which couple has the minimum EXT (t, r). If different tasks have the same p (t, r) on one resource, we prior schedule the task which has the maximum scheduling loss. The definition of scheduling loss is the same with that in the Sufferage algorithm which means the completion time difference on the best resource and the second best resource.

Nowm we will use a specific example to illustrate the steps of our algorithm. Table 2 shows three available tasks with their CT (t, r) on two heterogeneous resources. We suppose all other external environments are idle, and all RT (t, r)s are zero. The PC_1 is half of the PC_0.

For we have known the CT (t, r), we can calculate the p (t, r)s of these tasks on the two resources. The result is shown in Table 3.

The task t_1, t_2, and t_3 have the same p (t, r) on resource r_0, but t_3 has the maximum scheduling loss (30–15=15). Then we schedule task t_3 on resource r_0, and update the CT (t, r) Matrix shown in Table 4 and p (t, r) Matrix shown in Table 5.

In the same way, the task t_1 and t_3 have the same p (t, r) on resource r_1, but t_1 has the maximum scheduling loss (18−6=12). So we schedule task t_1 on resource r_1. Then update the CT (t, r) Matrix shown in Table 6 and p (t, r) Matrix shown in Table 7.

For task t_2 has the bigger p (t, r) on resource r_1, we schedule t_2 on resource r_1 and end the algorithm. Figure 5 shows the scheduling result.

Table 4 The CT (t, r) matrix after scheduling task t_1

	r_0	r_1
t_1	18	6
t_2	19	8

Table 5 The p (t, r) matrix after scheduling task t_1

	r_0	r_1
t_1	3/18	0.5
t_2	4/19	0.5

5.2 Algorithm analysis

The motivation of Min-Min algorithm is to preferentially execute the fastest task on its most suitable resource to reduce the overall completion time. The Sufferage algorithm sets up a new concept sufferage value to make the scheduling decision whose rationale is to cut down the time loss and balance the use of resources. Form the description above, we can see our algorithm takes the intrinsic correlation of resources and tasks into consideration by using the percentages of the completion time of each available task occupying the total used time of each resource. Especially we consider the performance of different resources when we calculate the percentages in Eq. (4).Through this way we can obtain both short overall completion time and excellent load balance level in most workflow situations.

And the advantage of MP algorithm is more prominent when the tasks in parallel are much unbalanced but the numbers of short tasks and long tasks are approximately equal. Other algorithms have serious flaws in these situations. The Min-Min algorithm prefers the situation that a few short tasks along with many long tasks in parallel branches, but Max-Min algorithm prefers the situation that a few long tasks along with many short tasks in parallel branches. As for Sufferage algorithm, when the size of tasks is much long compared to the computing capabilities of available resources, the sufferage values would be small and the algorithm may obtain a dramatic result about the overall performance [40].

Considering the time complexity of algorithms, we suppose there are m resources and n tasks in a DAG. The computing time of MP algorithm can be on the whole divided into two parts: the time $t_1$of searching the available tasks and the time $t_2$of calculating the percentages that each available tasks on all the resources. The first part is just a simple task sequential access problem, and we suppose there are $k(0 \le k \le n)$ tasks that have not been scheduled. Its estimation is like:

$$t_1 = O(k) \tag{5}$$

For time t_2, it is related to the specific structure of the DAG and the resources number m. In every iteration, we

Table 6 The CT (t, r) matrix after scheduling task t_3

	r_0	r_1
t_2	19	14

Table 7 The p (t, r) matrix after scheduling task t_3

	r_0	r_1
t_2	4/19	2/7

suppose the available task number is c ($0 \le c \le n$) and its estimation can be:

$$t_2 = o\left(c^2 \cdot m\right) \tag{6}$$

Suppose that the number of iteration times is I, then the total processing time t of MP algorithm can be calculated as follows:

$$t = I \cdot (t_1 + t_2) = I \cdot \left[o(k) + o\left(c^2 \cdot m\right)\right] \tag{7}$$

Through the analysis above, we can see the time complexity presentation of MP algorithm is the same with the Min-Min algorithm, Max-Min algorithm, and Sufferage algorithm. In this paper, the resource is abstract concept that it may be a local Hadoop cluster or a Spark cluster supported by one cloud computing provider. And the tasks usually are coarse grained. In a practical DAG task graph, the time complexity is acceptable.

6 Experimental results and discussion

6.1 Experimental environment

We use Java to complete simulation experiments, and firstly design a random workflow generator to generate various types of workflows and guarantee the accuracy and credibility of the experiment results. Before we use it, we are required to specify (1) number of tasks in the workflow; (2) range of the lengths of tasks; (3) range of the sizes of output files of tasks; (4) max out-degree of every task; (5) number of computing resources in Grid; (6) range of computing capabilities of resources; (7) the range of bandwidth of resources. In the simulation, we utilize the Million Instructions (MI) to measure the lengths of tasks and the GigaBytes (GB) to measure the sizes of output files. The length of tasks is set by the range [10,000, 6,000,000], and the sizes of output files are within the range [2, 34]. And the max out-degree of a task is set for 4. The computing capabilities and Bandwidth are measured in million instructions per second (MIPS) and million bits per second (Mbps) with respective ranges [400, 3,000] and [100, 1,024].

Table 8 Parameters of GA

Parameters	Value
Population size	Number of task×2
Crossover probability	0.7
Replacing mutation probability	0.5
Fitness	Makespan of workflow
Selection scheme	Elitism-Roulette Wheel
Stopping condition	100 iterations
Initial individuals	Randomly generated

6.2 Performance metrics

At present, there are numerous basic performance metrics to evaluate the application effects of scheduling algorithms. We

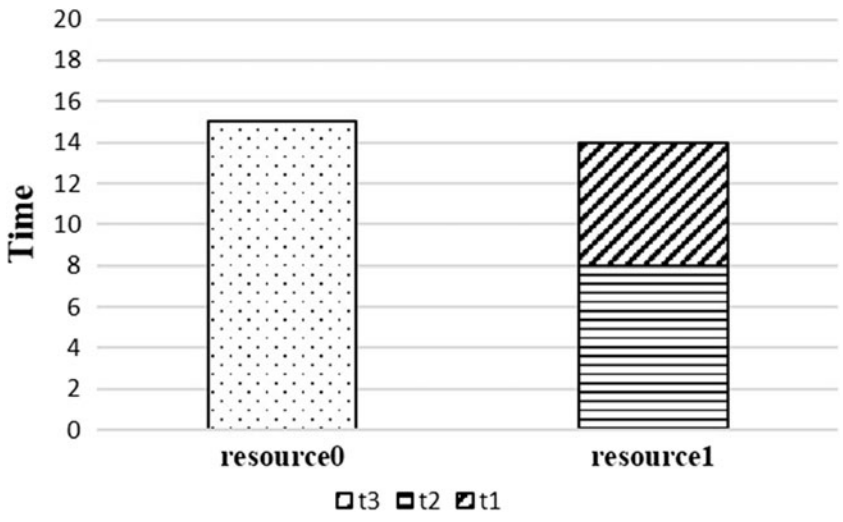

Fig. 5 The scheduling result of the case in Section 5.1

reference many papers about this area and choose the following performance metrics [45].

- Makespan is the main and common measure in the field of scientific workflow scheduling, and it is defined by

$$\text{Makespan} = \max\left\{\text{CT}\left(t_i, r_j\right)\right\}, 1 \le i \le N(T), 1 \le j \le N(R) \tag{8}$$

where the CT(t_i, r_j) is the final result of task t_i after completing scheduling.

- Average resource utilization rate (ARUR) is computed by dividing the total number of resources by the sum of resource utilization rates (RUR) of every resource.

$$\text{ARUR} = \frac{\sum_{j=1}^{N(R)} \text{RUR}_j}{N(R)} \tag{9}$$

Here, RUR_j can be calculated using Eq. (10):

$$\text{RUR}_j = \frac{\text{time}_j}{\text{Makespan}} \tag{10}$$

Here, time_j represents the cost time on resource r_j including executing the tasks mapped on it and transferring the output files to the next resources.

- Load balancing level (LBL) is an important measure of a scheduling algorithm on workflows and before we give its definition, we must calculate the mean square deviation of resource utilization rate as Eq. (11).

$$d = \sqrt{\frac{\sum_{j=1}^{N} \left(\text{ARUR} - \text{RUR}_j\right)^2}{N(R)}} \tag{11}$$

Then the LBL can be given by Eq. (12):

$$\text{LBL} = \left(1 - \frac{d}{\text{ARUR}}\right) \times 100\% \tag{12}$$

- Comprehensive Performance (CP) is used to measure the combination property of algorithms about Makespan and LBL. It can be calculated by Eq. (13). From (13), we can see that the value is smaller, its comprehensive performance is better.

$$\text{CP} = \text{Makespan} \times \frac{1}{\text{LBL}} \tag{13}$$

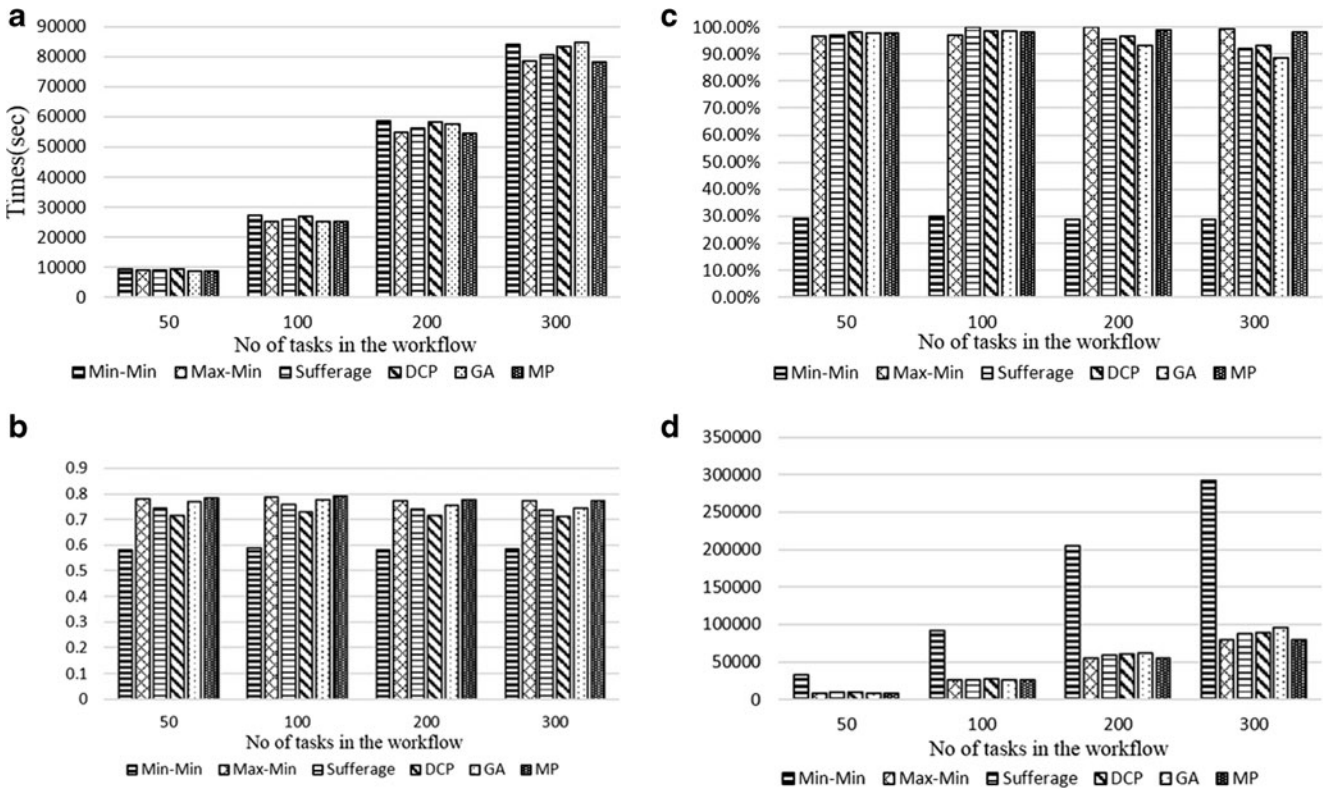

Fig. 6 Diagrams of experiment for performance comparison of six scheduling algorithms. **a** Makespan for tasks with different number. **b** ARUR for tasks with different number. **c** LBL for tasks with different number. **d** CP for tasks with different number

Table 9 Execution time of all algorithms

Algorithm	50 tasks(ms)	100 tasks (ms)	200 tasks (ms)	300 tasks (ms)
Min-Min	2	7	11	15
Max-Min	2	6	11	15
Sufferage	3	6	16	18
DCP	3	6	12	24
GA	803	6,265	32,458	118,872
MP	3	6	14	19

6.3 Experimental results analysis

In experiment 1, we compare Min-Min algorithm, Max-Min algorithm, DCP algorithm, Sufferage algorithm, GA and MP algorithm. The specific experimental environment has been discussed in part 1 of this section. We randomly generate scientific workflows composed of 50, 100, 200, and 300 tasks, and apply those algorithms on them to get the comparative results of Makespan, ARUR, LBL, and CP [46]. Table 8 shows the values of different parameters of GA. We choose 2 random resources for simulation which makes the resource competition fiercer and the comparative experimental results more evident. Then we use the average value of three repeated experiments. Figure 6 shows that the MP algorithm performs best of all algorithms in random experiments. Through the experimental contrast analysis, the average Makespan value of MP algorithm is preferable to the DCP algorithm, Min-Min algorithm, Max-Min algorithm, Sufferage algorithm, and GA by 7.1, 7.8, 0.7, 3.3, and 3.7 %, respectively. And the average ARUR value outperforms by 8.0, 25.1, 0.2, 4.6, and 2.5 %, respectively. As to the LBL, the load balancing level of Min-Min algorithm is terrible which may be explained by the too large difference of random resources in the experiments, but Max-Min algorithm as well as the MP algorithm performs excellently. The LBL values of DCP algorithm, Sufferage algorithm, and GA are slightly lower than our proposed algorithm. On average, the MP algorithm gets the best comprehensive performance, which means our algorithm can go for a better schedule not only on total completion time but resource utilization.

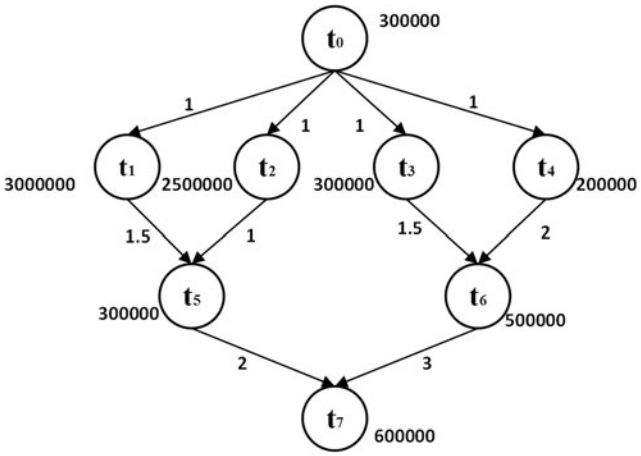

Fig. 7 Revision of . 4

From the experiment above, we also can see GA can generate good schedules because it explores the solution in the global space, but it may fall into the local minima. What is worse, the scheduling time is growing exponentially with the solution space. In our simulation, for a workflow of 300 tasks, the scheduling time of GA is hundreds and even thousands of times than others. The execution times of all algorithms for different workflows are shown in Table 9.

In experiment 2, we focus on the situations we talked above that the tasks in some parallel branches are much heavier than that of others, but long tasks and short tasks are approximately evenly distributed. As we have shown in Fig. 3, the distribution of tasks in this DAG is quite balanced. Table 10 is the experimental results that applying all algorithms on the workflow in Fig. 3, and we assume the sizes of output files of all tasks are 1 GB to simplify the workflow. The computing capabilities of resources used in this experiment are 400 and 800 MIPS, respectively, and Bandwidths are the same, both 100 Mbps. The results (shown in Table 10) suggest that the performance of our proposed algorithm is the same with the Max-Min and GA algorithms, and better than Min-Min, Sufferage, and DCP algorithm in Makespan. Although its load balance level is acceptable, it is not better than Sufferage algorithm. Now, we make task t_1 and t_2 bigger by expanding 10 times which means the left branches are much heavier than the

Table 10 Performance comparison on the balanced workflow of Fig. 4

Algorithm	Makespan(sec)	ARUR	LBL	CP
Min-Min	3,144.42	0.63	41.84 %	7,514.71
Max-Min	2,831.92	0.77	66.67 %	4,247.88
Sufferage	3,120.76	0.73	80.38 %	3,882.60
DCP	3,144.42	0.63	41.84 %	7,514.71
GA	2,831.92	0.77	66.67 %	4,247.88
MP	2,831.92	0.77	66.67 %	4,247.88

Table 11 Performance comparison on the unbalanced workflow of Fig. 7

Algorithm	Makespan(sec)	ARUR	LBL	CP
Min-Min	9,081.92	0.57	25.58 %	35,504.26
Max-Min	8,538.84	0.76	98.06 %	8,707.41
Sufferage	8,456.92	0.65	46.78 %	18,078.52
DCP	8,995.76	0.74	92.84 %	9,689.20
GA	8,370.76	0.80	94.07 %	8,898.57
MP	7,913.84	0.82	98.10 %	8,066.94

right ones as shown in Fig. 7. The number of long tasks is equal to the number of short tasks, and compared to the computing capabilities of available resources, the t_1 and t_2 are long enough that the differences between the complete times on the two resources are small. The MP algorithm generates up to average 8.8 % better Makespan, 17.4 % better ARUR, 80.6 % better LBL, and 33.2 % better CP than other algorithms in Table 11. The GA algorithm falls into the local optima, and it cost the longest time to reach its strategy. Combine with the experiment 1, we can obtain that our algorithm can solve the serious flaw in scheduling performance of other algorithms when branches in workflows are heavily unbalanced like above.

7 Conclusions

In this paper, we summarize the research hotspots nowadays in modern manufacturing and analyze the requirements for the processing of manufacturing big data. Then we introduce the scientific workflow management system which is an efficient tool to execute and manage big data, and beneficial to the scientific discoveries. With reference to the related work, a prototype architecture of scientific workflow management system applied to the future manufacturing is proposed.

This paper also points that an efficient workflow scheduling algorithm is important for using heterogeneous resources within systems. And a better workflow scheduling algorithm that can shorten the total completion time and have a good load balancing level is needed. Then the MP algorithm is proposed which considers the intrinsic correlation of information about resources and tasks and utilizes the percentages of resource service times to emphasize the overall performance. Compared with the classic DCP, Min-Min, Max-Min, Sufferage, and GA algorithms based on the common evaluation criteria, MP algorithm performs best and satisfies the requirements discussed above.

The research of scientific workflow management system is still at the initial stage. Future works in this research can be performed in the following directions. First, we do not have considered the security problem of the system, which is a broad direction to follow up in future research. Second, we just use simple heterogeneous environments which are assumed to be static. However, the actual computing environments are more complex and variable. In the next step, we should carry out more simulation experiments in complicated and dynamic environments to evaluate our algorithm.

Acknowledgment This research was supported by National Natural Science Foundation of China (NSFC, project no. 71171121).

References

1. Chui M, Loffler M, Robert R (2010) The internet of things. Mckinsey Q 2:1–9
2. Barolli L, Chen X, Xhafa F (2014) Advances on cloud services and cloud computing. Concurr Comput-Pract Exp 27(8):1985–1987
3. Liu J, Pacitti E, Valduriez P, Mattoso M (2015) A survey of DataIntensive scientific workflow management. J Grid Comput 3: 1–37
4. Holl S, Zimmermann O, Palmblad M, Mohammed Y, Hofmann-Apitius M (2014) A new optimization phase for scientific workflow management systems. Futur Gener Comp Syst 36:352–362
5. Gan Z, Wang J, Salomonis N, Stowe JC, Haddad GG, McCulloch AD, Zambon AC (2014) MAAMD: a workflow to standardize meta-analyses and comparison of affymetrix microarray data. BMC Bioinformatics 15(1):69
6. Zhao Y, Li Y, Lu S, Lin C (2014) Devising a cloud scientific workflow platform for big data. IEEE World Congress on Services IEEE: 393-401
7. Huang B, Li C, Yin C, Zhao X (2013) Cloud manufacturing service platform for small-and medium-sized enterprises. Int J Adv Manuf Technol 65(9-12):1261–1272
8. Lin C, Lu S, Fei X, Chebotko A, Pai D, Lai Z, Hua J (2009) A reference architecture for scientific workflow management systems and the VIEW SOA solution. IEEE Trans Serv Comput 2(1):79–92
9. Malawski M, Juve G, Deelman E, Nabrzyski J (2015) Algorithms for cost-and deadline-constrained provisioning for scientific workflow ensembles in IaaS clouds. Futur Gener Comp Syst 48: 1–18
10. Fard HM, Radu P, Thomas F (2014) Multi-objective list scheduling of workflow applications in distributed computing infrastructures. J Parallel Distr Com 74(3):2152–2165
11. Hongzhao D, Dongxu L, Yanwei Z, Ying C (2005) A novel approach of networked manufacturing collaboration: fractal web-based extended enterprise. Int J Adv Manuf Technol 26(11-12): 1436–1442
12. Flammia G (2001) Application service providers: challenges and opportunities. IEEE Intell Syst 16(1):22–23
13. Xu X (2012) From cloud computing to cloud manufacturing. Robot Cim-Int Manuf 28(1):75–86
14. Tao F, Cheng Y, Da Xu L, Zhang L, Li BH (2014) CCIoT-CMfg: cloud computing and Internet of things-based cloud manufacturing service system. IEEE T Ind Inform 10(2):1435–1442
15. Laili Y, Tao F, Zhang L, Sarker BR (2012) A study of optimal allocation of computing resources in cloud manufacturing systems. Int J Adv Manuf Technol 63(5-8):671–690
16. Lee J, Lapira E, Bagheri B, Kao HA (2013) Recent advances and trends in predictive manufacturing systems in big data environment. Manuf Lett 1(1):38–41
17. Barseghian D, Altintas I, Jones MB et al (2010) Workflows and extensions to the Kepler scientific workflow system to support environmental sensor data access and analysis. Ecol Inf 5(1):42–50
18. Li X, Song J, Huang R (2014) A Kepler scientific workflow to facilitate and standardize marine monitoring sensor parsing and dynamic adaption. 5th IEEE Int Conf Softw Engine Serv Sci IEEE: 1023-1026
19. Paterson T, Law A (2009) An XML transfer schema for exchange of genomic and genetic mapping data: implementation as a web service in a Taverna workflow. BMC Bioinformatics 10(1):252
20. Deelman E, Singh G, Su MH, Blythe J, Gil Y, Kesselman C, Katz DS (2005) Pegasus: a framework for mapping complex scientific workflows onto distributed systems. Sci Programming-Neth 13(3): 219–237
21. Fahringer T, Prodan R, Duan R, Hofer J, Nadeem F, Nerieri F, Wieczorek M (2007) Askalon: A development and grid computing

environment for scientific workflows. Workflows for e-Science. Springer, London, pp 450–471
22. Von Laszewski G, Gawor J, Lane P, Rehn N, Russell M (2002) Features of the java commodity grid kit. Concurr Comput-Pract Exp 14(13-15):1045–1055
23. Kacsuk P, Sipos G (2005) Multi-grid, multi-user workflows in the P-GRADE grid portal. J Grid Comput 3(3-4):221–238
24. Guan Z, Hernandez F, Bangalore P, Gray J, Skjellum A, Velusamy V, Liu Y (2006) Grid-flow: a grid-enabled scientific workflow system with a petri-net-based interface. Concurr Comput-Pract Exp 18(10):1115–1140
25. Bubak M, Nowakowski P, Unger S (2006) K-WfGrid—knowledge-based workflow system for grid applications. Proc CGW 6: 1–12
26. MTConnect Institute (2013) MTConnect Institute. http://www.mtconnect.org/. Accessed on May 27 2015
27. Wang J, Crawl D, Altintas I, Li W (2014) Big data applications using workflows for data parallel computing. Comput Sci Eng 16(4):11–21
28. Wang J, Crawl D, Altintas I (2009) Kepler + Hadoop: a general architecture facilitating data-intensive applications in scientific workflow systems. Proceedings of the 4th Workshop on Workflows in Support of Large-Scale Science ACM : 12
29. Zaharia M, Chowdhury M, Franklin MJ, Shenker S, Stoica I (2010) Spark: cluster computing with working sets. Proceedings of the 2nd USENIX conference on Hot topics in cloud computing: 10-10
30. Djurdjanovic D, Lee J, Ni J (2003) Watchdog Agent—an infotronics-based prognostics approach for product performance degradation assessment and prediction. Adv Eng Inform 17(3): 109–125
31. Wang L (2013) Directed acyclic graph. Encyclopedia of Systems Biology: 574-574
32. Yu-Kwong K, Ahmad I (1996) Dynamic critical-path scheduling: an effective technique for allocating task graphs to multiprocessors. IEEE Trans on Par and Dist Systems 5(7):506–521
33. Rodriguez Sossa M, Buyya R (2014) Deadline based resource provisioning and scheduling algorithmfor scientific workflows on clouds, cloud computing. IEEE Trans 2(2):222–235
34. Chunlin L, Layuan L (2006) QoS based resource scheduling by computational economy in computational grid. Inf Process Lett 98(3):119–126
35. Singh S, Chana I (2015) QRSF: QoS-aware resource scheduling framework in cloud computing. J Supercomput 71(1):241–292
36. Ezzatti P, Pedemonte M, Martin A (2013) An efficient implementation of the Min-Min heuristic. Comput Oper Res 40(11):2670–2676
37. He XS, Sun XH, Von Laszewski G (2003) QoS guided min-min heuristic for grid task scheduling. J Comput Sci Technol 18(4):442–451
38. Singh M, Suri PK (2008) QPS Max-Min Min-Min: a QoS based predictive Max-Min, Min-Min switcher algorithm for Job scheduling in a grid. Inform Technol J 8:1176–1181
39. Kartal Tabak E, Barla Cambazoglu B, Aykanat C (2014) Improving the performance of independent task assignment heuristics minmin, maxmin and sufferage. IEEE Trans Parallel Distrib Syst 25(5): 1244–1256
40. Casanova H, Legrand A, Zagorodnov D, Berman, F (2000) Heuristics for scheduling parameter sweep applications in grid environments. Proceedings 9th Heterogeneous Computing Workshop. IEEE: 349-363
41. Yu J, Buyya R (2006) Scheduling scientific workflow applications with deadline and budget constraints using genetic algorithms. Sci Programming-Neth 14(3-4):217–230
42. Wieczorek M, Prodan R, Fahringer T (2005) Scheduling of scientific workflows in the ASKALON grid environment. ACM Sigmod Rec 34(3):56–62
43. Guo F, Yu L, Tian S, Tian S (2014) A workflow task scheduling algorithm based on the resources' fuzzy clustering in cloud computing environment. Int J Commun Syst 28(6):1053–1067
44. Bittencourt LF, Madeira ERM (2013) Using time discretization to schedule scientific workflows in multiple cloud providers. Sixth IEEE Int Conf Cloud Comput IEEE: 123-130
45. Chauhan SS, Joshi RC (2010) A weighted mean time min-min max-min selective scheduling strategy for independent tasks on grid. 2rd IEEE Int Adv Comput Conf IEEE:4-9
46. Ahmad I, Kwok YK, Wu MY (1996) Analysis, evaluation, and comparison of algorithms for scheduling task graphs on parallel processors. Proceedings of Second International Symposium on Parallel Architectures, Algorithms, and Networks IEEE: 207-213