International Journal of Grid Distribution Computing
Vol. 8, No. 3, (2015), pp.227-244
http://dx.doi.org/10.14257/ijgdc.2015.8.3.23

# A Kriging Based Forecasting and Scheduling System for Scientific Computing Cloud Applications

Zhaojun Li[1,2], Xinyu Wang[2], Zheng Li[2,*], Xicheng Wang[2] and Keqiu Li[1]

[1]*School of Computer Science and Technology, Dalian University of Technology, Dalian City, Liaoning Province, P.R.China*
[2]*State Key Laboratory of Structural Analysis for Industrial Equipment, Dalian University of Technology, Dalian City, Liaoning Province, P.R.China*
*lizhaojun@mail.dlut.edu.cn, lizheng@dlut.edu.cn*

***Abstract***

*Regarding to the theories and techniques of cloud computing having been developed and applied in scientific computing field, tasks can be conveniently managed by the cloud platform on the basis of standardized scheduling system with cost (resources consumed) recorded. However, there are two issues which drag the customers' attention: 1) When will the tasks expect for termination (response time) under a specific resource scheduling; 2) What is the best scheduling solution by considering cost. In order to reply these two questions, a Kriging based forecasting and scheduling system has been proposed in this paper. With the cooperation between the scientific designer and the cloud designer, the design variables for evaluating the cloud applications can be achieved; Kriging surrogate model is then introduced to simulate the approximate functional relationship between the design variables and the response time of the tasks; Sequential quadratic programming optimization algorithm then provides the best scheduling solution for the tasks if cost constraints are to be met. Two real scientific computing cloud applications have been testified on an OpenStack cloud platform, with consequences described in details. The work in this paper has put forward a novel way for the designers and the customers on predictable and reasonable scheduling strategies for the various resource-intensive scientific computing cloud applications with surrogate models and optimization algorithms.*

***Keywords:*** *Resource-intensive scientific computing, Cloud computing, task scheduling, Kriging, Sequential quadratic programming, OpenStack*

## 1. Introduction

The various scientific computing applications always demand a lot of resources to accomplish from hardware to software, which lead to resource-intensive modes. Meanwhile, the sophisticated characters of the problems also indicate uncertain time of periods for doing their tasks, with great quantity of which to be the long term repeatable computing tasks. An ordinary easy way to perform these applications would be simply writing programs to control the process and related parameter files, with the existing hardware and software resources under the normal desktop or laptop computing environments. The users or designers could not do any further stuff but wait for the termination, with the response time acquired through experience [1-3]. After the high performance computing technologies having been worked up such as parallel computing, cluster computing and grid computing, the implementation time for the resource-intensive scientific computing applications may be reduced due to high performances. But still,

ISSN: 2005-4262 IJGDC

what not having been changed so far is that the time of executions for these applications could not be given reasonably, and the resources scheduled are always dependent on the experience of the users too[4, 5].

With cloud computing, the computing resources could be measured by volume consumed for these applications through the virtualization techniques and methods from different implementation levels. There are a wide variety of generic or dedicated cloud platforms for dealing with the real scientific computing problems privately or publicly based on the open source cloud platforms, aiming to address these issues much more efficiently and easily. However, the strategies on instructing the customers about how to manipulate the computing resources that the applications relied on and how to obtain the trustful response time of them have not been widely discussed [6].

In this paper, we have proposed a Kriging based forecasting and scheduling system for the resource-intensive scientific computing cloud applications: 1) To analyze the applications in a rational mode, the design variables have to be found out specifically from two aspects including scientific designers and cloud designers; 2) Kriging surrogate model is built upon the design variables and their responses (response time) and reasonably answers of responses would be given by this approximate model using the predictor function in Kriging for all kinds of resource arrangements; 3) If the cost (always charged based on the resources consumed in clouds) limits have been set up, the optimal solutions would be received with necessary calculations made by the optimization algorithm ( SQP) so as to tell whether or not scheduling exists and what will it be if the answer is positive. 4) And to test and verify the system, an OpenStack driven cloud platform with two real computing applications have been framed, with data gained showing that effective and valid for these kinds of applications.

The rest of this article is organized as: 2. Related work discusses some background theories and knowledge corresponding to cloud computing and resource-intensive scientific computing applications according to the literature; 3. Research background makes some analysis for the methods and algorithms applied in this paper; 4. System design detailed investigates the ideas and implementation of the system, along with the platform constructed for testing; 5. Testing cases are the two applications performed as for verifying the methods and system proposed; 6. Conclusion gives a summary for this paper.

## 2. Related work

During the past decade, the cloud theories and technologies have been developed rapidly: both the techniques to build practical public and private cloud platforms and the algorithms suitable for cloud based scheduling have got great achievements in this area. This section discusses the current research status about the cloud computing and task scheduling methods applied for them; meanwhile the application-level analysis for the resource-intensive scientific computing applications and the methods for their scheduling have also been discussed.

### 2.1. Cloud Computing and Task Scheduling

Cloud computing is a novel outcome of business ideas based on traditional computer science theories and technologies such as parallel computing, cluster computing, grid computing, distributed computing, utility computing, virtualization, load balance, etc.[7]. Designers have made achievements through different virtualization levels while offering the cloud services for the various needs: infrastructure as a service (**IaaS**), platform as a service (**PaaS**) and software as a service (**SaaS**), and so on. The services established on the high performance hardware deployed in datacenters automatically manage the resources, transform them into the forms for what the customers requested and recycle them when they are no longer needed[8, 9].

 

Platforms e.g. Amazon EC2, Google App Engine, Microsoft Azure are some worldwide famous commercial clouds founded by the big IT crocodiles, which could be utilized for developing applications such as net disk applications, mail service applications, web-based applications, standard Windows-based applications and so on[10]; moreover, the open source cloud platforms e.g. Eucalyptus, CloudStack, OpenStack, and the cloud computing tools e.g. HTCondor, CloudSim, Gridsim are getting more and more attentions due to the ever-growing participators including IT companies, cloud service vendors, engineering researchers etc.[11-13].

The task scheduling problem is one of the resource scheduling problems in cloud computing. The task scheduling algorithms are responsible for mapping the cloud tasks defined by users onto the virtual machines (**VMs)** currently available in the cloud datacenter normally based on a matrix called estimated time of completion (**ETC**). The length and width of the ETC matrix stand for the number of cloud tasks and the VM numbers respectively; and each value devotes to the estimated time of the task to be scheduled on the corresponding VM for completion based upon the resources requested by the task and owned by the VM, such as CPU performance, memory space, input & output data file sizes[14, 15]. The objectives for the scheduling include response time, load, service price, etc.

The task scheduling problem in cloud computing can be also regarded as an optimization issue with its own characteristics: the design variables are discrete and bounded, the objective function is the maximum/minimum value of the response time, load, price and so on; the constrains may be defined as the relevant sequence of the tasks if some are relied on the others (a directed acyclic graph) [16]. The algorithm such as: FCFS, Min-min and Round-robin are the earliest ones to address this problem originated from the resource scheduling algorithms investigated in operating systems[16, 17]; Huang proposes a CASA algorithm with one meta scheduler on each node[18]; Lee discusses a novel model for calculation the earning contains both the cloud services and the other services[19]; Garg defines a multi-datacenter scheduling algorithm based on carbon/energy with the purpose of obtaining the minimum carbon emissions and the maximum economic profits[20].

Meanwhile, the heuristic optimization algorithms such as genetic algorithm (**GA**), particle swarm optimization (**PSO**), ant colony optimization (**ACO**) are also introduced into the task scheduling problem in the recent cloud designs. For example, Mezmaz investigates a GA based mixed scheduling algorithm for achieving the minimum energy cost and the minimum response time[21]; Li presents a load balancing ant colony optimization (**LBACO**) algorithm to balance the entire system load while trying to minimizing the response time of a given tasks set[22]; Pandey puts forward a PSO based heuristic to schedule applications to cloud resources that takes into account both the computation cost and the data transmission cost[23].

### 2.2. Application-Level and Resource-Intensive Scheduling Methods

It is a tough errand to dynamically schedule the resources possessed by applications as one should grasp the diversification of these applications according to the resources as clear as possible. Based on this purpose, Berman proposes an application-centric scheduling which is used by virtually all meta-computer programmers to achieve performance on meta-computing systems[24]; Li presents a portable middleware layer component designed for implementation over POSIX-compliant operating system for deal with the real-time scheduling[25]; Toporkov develops a combination of job-flow and application-level techniques of scheduling within virtual organizations within a grid computing environment[26]; Zhu introduces a scheduling optimization for a Web server cluster with a master/slave architecture which separates static and dynamic content processing[27]; Aziz investigates a framework for resource allocation and task scheduling, management architecture for resource services (**MARS**), which efficiently

allocates resources for resource-intensive applications which can be represented as task interaction graph (**TIG**)[28]; Verboven discusses a method using performance models based on the runtime characteristics of virtual workloads to address the black-box scheduling for resource-intensive virtual machine workloads[29].

## 3. Research Background

The research proposed in this article is based on several widely accepted notations and theories, including the optimization algorithm SQP, the surrogate model tool Kriging and the open source cloud platform OpenStack.

### 3.1. Sequential Quadratic Programming

Consider a nonlinear programming problem with the form as described below[30]:

$$\begin{aligned} &\min\ f(x), \\ &s.t.\left\{\begin{array}{l} b(x) \geq 0 \\ c(x) = 0 \end{array}\right\} \end{aligned} \tag{1}$$

The Lagrangian form for this problem is:

$$L(x, \lambda, \sigma) = f(x) - \lambda^T b(x) - \sigma^T c(x) \tag{2}$$

in which $\lambda$ and $\sigma$ are the Lagrange multipliers. At an iterate $x_k$, a basic sequential quadratic programming algorithm defines an appropriate search direction $d_k$ as a solution to the quadratic programming sub-problem[30, 31]:

$$\begin{aligned} &\min\ f(x_k) + \nabla f(x_k)^T d + \frac{1}{2} d^T \nabla^2_{xx} L(x_k, \lambda_k, \sigma_k) d, \\ &s.t.\left\{\begin{array}{l} b(x_k) + \nabla b(x_k)^T d \geq 0 \\ c(x_k) + \nabla c(x_k)^T d = 0 \end{array}\right\} \end{aligned} \tag{3}$$

Note that the term $f(x_k)$ in the expression above may be left out for the minimization problem, since it is constant. SQP is a historic gradient-based optimization algorithm, which has still being proved to be effective in many engineering computing applications. Matlab software has an optimization toolbox which consists of many different optimization algorithms including SQP, normally could be called from the function named "fmincon"[32]. One should provide the lower and upper boundaries of the design variables, along with the objective function and the constrain function to be optimized, and the optimal solution would be achieved after a certain number of computing iterations shortly.

### 3.2. Kriging Surrogate Model

The Kriging surrogate model has been widely accepted by engineering researchers because this method could predict a "surrogate" mathematical function between samples and their responses which could be used for optimizations. With samples $x = [x_1, x_2, \mathrm{K}, x_n]$ and their responses $y = [y_1, y_2, \mathrm{K}, y_n]$, Kriging model can be established as[33, 34]:

$$\hat{y}(x) = f^T(x)\beta + z(x) \tag{4}$$

it contains two parts: regression part and random part. Coefficient $\beta$ is the regression ratio. Deterministic drift $f(x)$, providing the global approximation of simulation in the design domain, is often described as a polynomial of $x$.

 

There is an open-source Kriging surrogate model toolbox written in Matlab script called DACE with generic regression models and correlation functions which are relied by Kriging[34, 35]. Normally, researchers only need to provide the necessary guesses on $\theta$ , the correlation function parameters along with the design variables and their responses to use the toolbox for Kriging model construction. With the surrogate function $\hat{y}(x)$ obtained, optimization can be processed using standard gradient-based optimization algorithms *e.g.*, SQP.

### 3.3. Openstack Open-Source IaaS

OpenStack IaaS is an open source cloud platform which could control large number of pools of computing, storage, and networking resources throughout a datacenter. Founded by Rackspace Hosting and NASA, OpenStack has grown to be a scalable open-source cloud operating system. The latest version of OpenStack is Icehouse [36, 37].

A standard OpenStack platform should include compute, networking, storage, identity and image services, which could normally implemented by nova, quantum, cinder, keystone and glance modules. Keystone and glance are shared services. Messages are passed around by using Rabbit MQ software. OpenStack has also provided powerful RESTful accessing and programming interfaces that could be used for managing low level physical resources and building user customized cloud platforms, in order to provide infrastructure, platform and software services themselves, as shown in Figure 1 [38, 39].

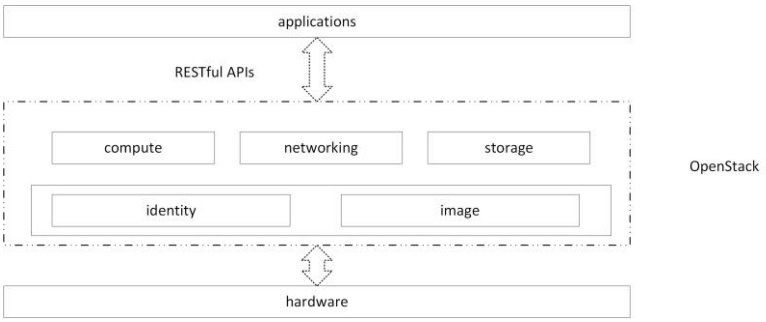


**Figure 1. OpenStack IaaS**

## 4. System Design

As to grasp the operating characteristics of the resource-intensive scientific computing cloud applications from a rational way instead of ad-hoc, we have set up a Kriging surrogate model based system for the analysis and forecasting. The design variables must be selected before a Kriging model could be acquired. When there are resource quantitative restrictions, a SQP optimization process would give the optimal solutions according to the surrogate model.

### 4.1. Design Variables

No matter what kinds of the applications, the computing resources are the indispensable stuff they must be relied on for fulfilling their specific tasks from hardware to software. To a given application, the software will be decided and installed on specific virtual machine images and the hardware resources are scheduled by the cloud platform for running and recycled after the termination. Cost of the cloud tasks is mainly valued by whatever resources have been consumed, usually a linear expression of the hardware (software could be an additional constant cost, sometimes). To the important scheduling objective know as response time (the duration from the submission to the termination of the task), we use Eq.(5) for modeling it.

$$f(x_D, x_R) = t \tag{5}$$

where $x_D$ is the vector of the design variables form the scientific designer aspect and $x_R$ is the vector of the design variables from the cloud designer aspect regarding to the response time. For examples, if one want to do a Moldflow analysis application, $x_D$ could contain the number of the FEM elements; if an Ansys based structural optimization design is considered, $x_D$ could involve both the dimensions of the structure and the optimization iterations defined, *etc*. And $x_R$ may always include the hardware resources *e.g.*, virtual CPU number, virtual MEM amount, virtual disk capacity and so on. Meanwhile, to another key scheduling target cost, we use Eq.(6) for calculating it.

$$c = g(R, t) = t\sum_j w_j R_j,\ j = 1, 2, \mathrm{K}\ , k \tag{6}$$

k is the type of the computing resources needed by the application (only hardware related is considered in this article), $w_j$ is the weight of resource type j and $R_j$ is defined as the resource amount scheduled to the given task.

### 4.2. Application-Based "Sampling"

Sampling is generally known as the procedure to generate samples in applications for analysis, which is especially significant for doing the all kinds of engineering optimization problems. With samples given as input data, the statistical data (which is usually called as "response" in optimization fields) will be calculated by standard software and programs. The quotation mark means the samples mentioned in this paper are not actually generated using sampling algorithms but collected from the applications' historical running statistical data. More specifically, every single application instance would be recorded and logged carefully, the main factors interested in this article include all of the values of design variables and the scheduling objectives. The samples group is surely the values of the design variables, and the responses group is covered both the response time and the cost.

### 4.3. Kriging Surrogate Model Creation

Apparently, the exact formula *f* standing for in Eq.(5) is unable to easily achieved, so we decided to build a surrogate relationship instead, as described in Eq. (7).

$$samples = x(x_D, x_R) \xrightarrow[Kriging]{\hat{f}} t = responses \tag{7}$$

where a Kriging surrogate model is worked up on the samples and their responses (response time specifically, and cost will be obtained from the linear expression Eq.(6) easily). In this paper, we use the DACE toolbox for getting a Kriging model. And the response (response time) for a new sample (design variables) would be forecasting from this Kriging model easily with the predictor function in the DACE toolbox, as could be expressed as $\hat{f}(x) = \%$. Beware that $\%$ is only the predicted response time based on the model, which could further be one part of the samples and responses after the task is computed over.

A surrogate model is never going to be the genuine function, expect for approaching it gradually with samples growing. As to our experience, samples must be at least 5 times as the dimension of the design variables as to build an acceptable surrogate Kriging model.

### 4.4. Scheduling Optimization with Cost Limits

When the cost limits are set by whoever the end users or the cloud owner, the scheduling problem becomes an optimization issue with constrains to be met. Basically,

 

customer concerns how many resources to be allocated for a given application task with budget limitations. The problem could be pictured in the following Eq.(8)

$$\begin{cases} \min f(x) = \min f(x_D, x_R) \rightarrow \min \hat{f}(x_D, x_R) \\ c \rightarrow \hat{c} \leq C \\ x_D = X_D \end{cases} \tag{8}$$

where C and XD are the maximum cost allowed for this task and the values of the scientific designer related design variables. We use the SQP optimization algorithm to cipher this constrained optimization problem and finally get the values of the cloud designer related design variables XR, which is the trustful optimal scheduling solution based on the Kriging surrogate model constructed.

### 4.5. Virtual Computing Machine Image and Virtual Computing Application

To be able to present a generic scientific computing cloud platform, we have developed two key modules： virtual computing machine image (**VCMI**) and virtual computing application (**VCA**) based on OpenStack platform, as shown in Figure 2 and Figure 3.

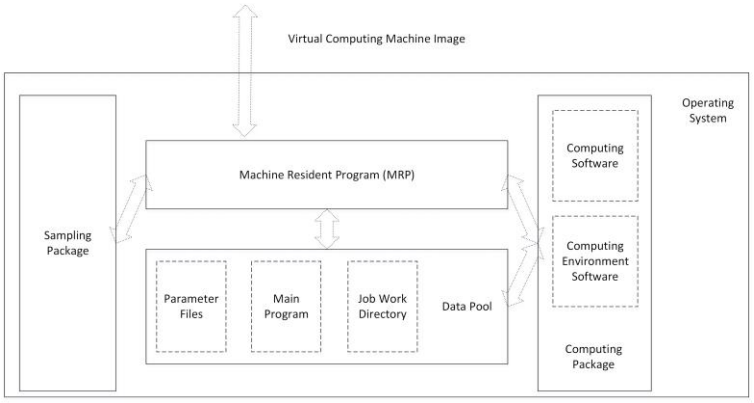


**Figure 2. VCMI**

A VCMI is an image (in our platform, KVM virtual machine image) created for general scientific computing applications: the computing package consists of the software needed for the computing and the software for building up the computing environment (MPICH2 software for parallel computing, Condor software for grid computing *etc*.); the sampling package collects the values of all samples and their responses through the computing logs of tasks, only activated on the head host; the data pool is the local place of storage for the tasks' parameter files, main programs and work directories, which is invisible to the users; and the machine resident program (**MRP**) is the gate of the host (running image) for communication with the cloud backend or the other running hosts. The image is generated on standard operating system, Windows 7 Professional, CentOS 64 bit, and so on.

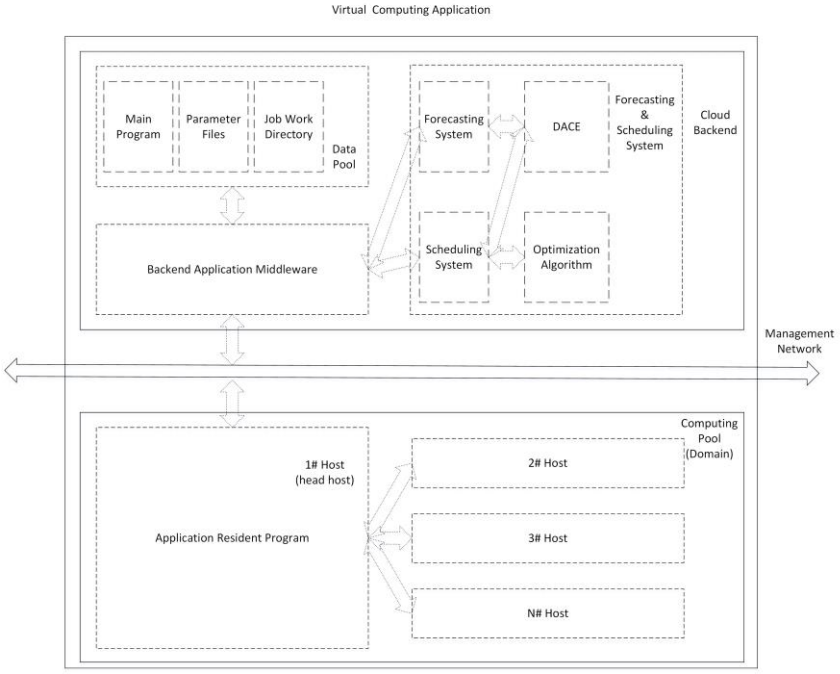


**Figure 3. VCA**

A VCA is a logic module what we created for modeling generic scientific computing applications: the data pool saves the data submitted by the users and also the results of terminated tasks; the DACE includes the Kriging surrogate program and the data of the samples and responses passed by the backend application middleware, providing the forecasting function called by the forecasting system and the scheduling function called by the scheduling system with the help of standard optimization algorithm; the backend application middleware is the interface for the cloud backend packages and the computing pool; a computing pool which formed with a few running hosts is the carrier for all of the computing tasks, which is also called Domain in our platform; the first set up host in a domain will be assigned for being the head host with the

### 4.6. Platform Mechanism

The mechanism of the testing platform can be described using the following chart Figure4, with activities marked in rectangles grouped by different initiators. Some necessary explanations are listed as:

1) User prepares all the parameter files ready belonging to the task which need to run for a specific application, along with the values of the design variables $x_D$ which he would be aware of;

2) User makes a request for setting up a computing pool (domain) for tasks, the scheduling option for the cloud resources will be either defined manually (short as manual in the following) or computed automatically (short as automatic in the following), and the cost maximum amount should also be provided if the latter one is chosen;

3) Cloud backend will do the optimization calculation in Section 4.4, the optimal scheduling of the resource combination is achieved when automatic is selected;

4) User submits the task to the pool he founded;

5) Cloud backend starts to launch a proper number of VCMIs as running hosts to make up the domain, with total computing resources same as the user claimed to be (manual) or the optimal solution (automatic);

 

6) All of the VCMIs get themselves ready for computing after initializing such as host name setting, ip address setting etc. and environment construction with their pre-installed computing environment packages, with commands passed from the head host;

7) The head host downloads the parameter files from the cloud backend, transfers them to the right computing host;

8) All of the computing hosts do the computing as needed, send the result files to the head host when finished;

9) The head host uploads all the result files to the cloud backend for all the tasks which have been done for computing;

10) User will submit as many as tasks for computing with specific domain options;

11) The head host will log all tasks' status and draw out all the related data as samples and responses, and share them with the cloud backend for usage;

12) User will get the forecasting response time for the tasks he concerned, as predicted by the cloud backend though Kriging surrogate model;

13) Cloud backend will shut down all the hosts and recycle the computing resources when all the tasks have successfully terminated and no more tasks is submitted by the user, after a certain period of time (to wait for further commands or tasks, in this paper, 120s) or a domain "STOP" command has been passed from the user.

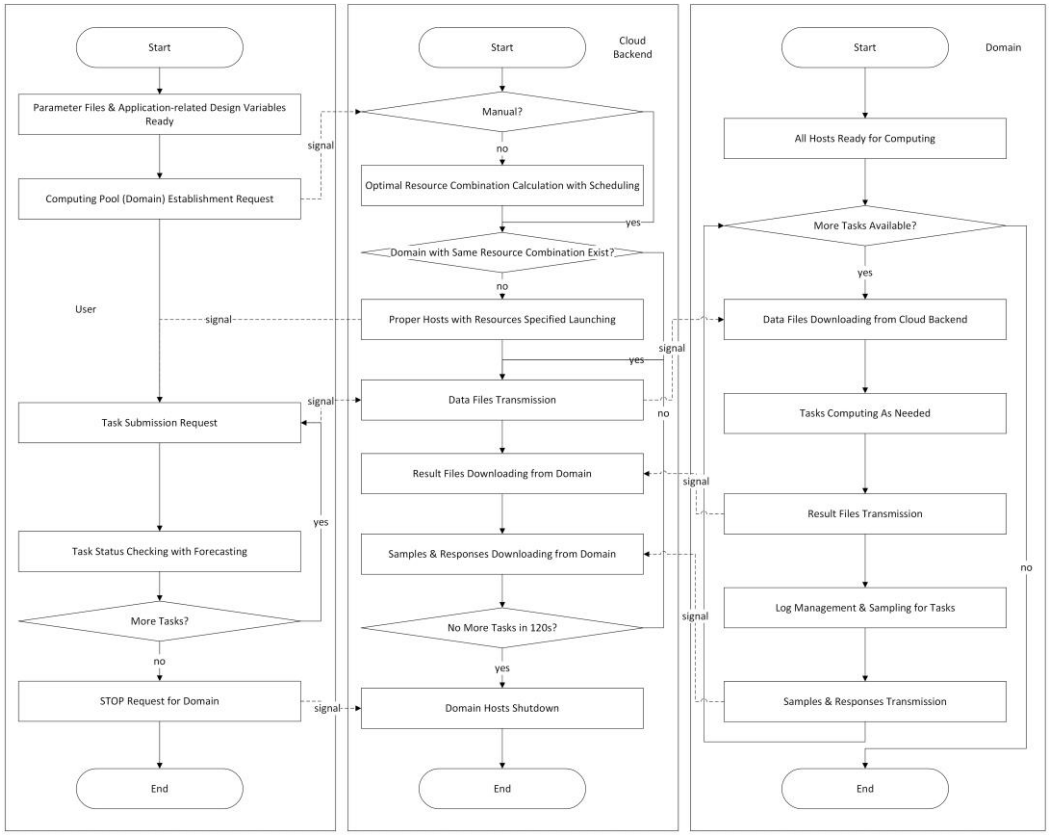


**Figure 4. Mechanism**

## 5. Testing Cases

Two Practical Applications have been worked up on the cloud platform we established, separately named simulation of the formation of a droplet with dissipative particle dynamics and warpage optimization with dynamic injection molding technology. For both applications, the vector of the cloud designer related design variables is defined as $x_R = [x_{VCPU}, x_{VMEM}]$, with their limits set to [1,6] and [3,12], respectively. And the

weighted parameter $w_j$ has been set to 1.0 for all. To be able to build a trustful Kriging model, a few samples are to collected for both of the applications which are randomly generated according to the ranges for all design variables. And to verify the system proposed, 5 tasks are presented for both of the applications to be the test samples. Note that it is the forecasting and scheduling performances we care about in this article other than the final computing results, which have been ignored for detail discussion. The testing data has been carefully sorted out from the tests' computations and described using figures and tables within the following subsections.

### 5.1. Testing Case One

Dissipative particle dynamics (**DPD**) is one mesoscale simulation method to investigate many physical behavior of fluids, like water, polymer solution etc. This application will be simulating the formation of a droplet by employing the form of potential energy (see Figure 5). Some parameters, for instances the time of steps, total number of particles, potential parameters etc., will induce different degrees of the time consumption of this simulation.

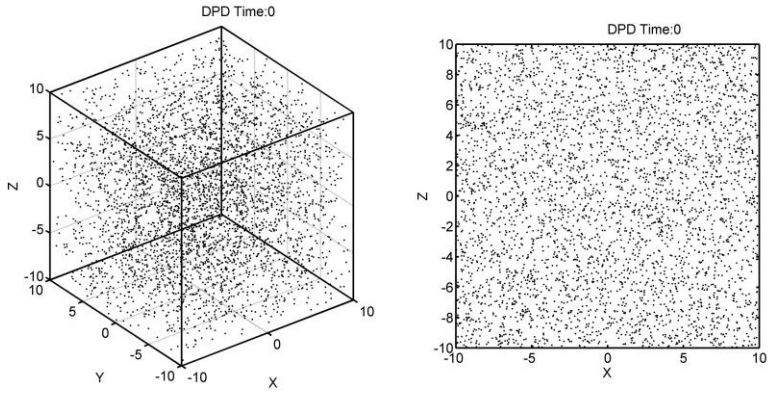


**Figure 5. App I**

The vector of the scientific designer related design variables in this case is defined as $x_D = [x_S]$ with values ranges in [3000,20000], which means the time of steps for simulating the procedure. The 20 initial samples of this case are shown in Table.1, and the forecasting testing consequences have been shown in Table.2. The cost limits of this case are defined as 10000, 20000 and 30000 for three sub-tests, with results shown in Tables. 3-5. Figure 6 has picturing the performances of all the tests.

**Table 1. Samples of Testing Case One**

| $\mathbf{x}_D$ | $\mathbf{x}_R$ | | Scheduling Objectives | |
|---|---|---|---|---|
| $x_S$ (Steps) | $x_{VCPU}$ | $x_{VMEM}$ | t(s) | c |
| 15900 | 1 | 10 | 2911 | 32021 |
| 10400 | 6 | 6 | 1948 | 23376 |
| 18000 | 1 | 4 | 3589 | 17945 |
| 6400 | 3 | 10 | 1108 | 14404 |
| 2000 | 6 | 11 | 350 | 5950 |
| 16900 | 1 | 12 | 3326 | 43238 |
| 3900 | 5 | 9 | 684 | 9576 |
| 20000 | 2 | 3 | 3747 | 18735 |
| 3500 | 4 | 8 | 608 | 7296 |

 

| | | | | |
|---|---|---|---|---|
| 8800 | 6 | 6 | 1608 | 19296 |
| 19100 | 2 | 3 | 3877 | 19385 |
| 2800 | 5 | 8 | 494 | 6422 |
| 6500 | 3 | 7 | 846 | 8460 |
| 1200 | 6 | 6 | 216 | 2592 |
| 2500 | 6 | 5 | 465 | 5115 |
| 7100 | 5 | 6 | 1286 | 14146 |
| 19900 | 4 | 9 | 3396 | 44148 |
| 7000 | 3 | 7 | 1284 | 12840 |
| 11000 | 2 | 10 | 1920 | 23040 |
| 3700 | 3 | 9 | 639 | 7668 |

**Table 2. Forecasting Tests of Testing Case One**

| $\mathbf{x}_D$ | $\mathbf{x}_R$ | | Scheduling Objectives | | | |
|---|---|---|---|---|---|---|
| $x_S$ (Steps) | $x_{VCPU}$ | $x_{VMEM}$ | t(s) | c | ℉(s) | ℉ |
| 18800 | 2 | 3 | 3880 | 19400 | 3809.877 | 19049.38 |
| 17900 | 1 | 4 | 3321 | 16605 | 3541.091 | 17705.46 |
| 11600 | 2 | 10 | 2014 | 24168 | 1919.952 | 23039.42 |
| 3600 | 5 | 9 | 669 | 9366 | 652.9559 | 9141.382 |
| 17300 | 4 | 11 | 2790 | 41850 | 3024.499 | 45367.48 |

**Table 3. Scheduling Tests of Testing Case One--I(MAXC=10000)**

| $\mathbf{x}_D$ | $X_R$ (℉) | | Scheduling Objectives | | | |
|---|---|---|---|---|---|---|
| $x_S$ (Steps) | $x_{VCPU}$ | $x_{VMEM}$ | t(s) | c | ℉ | ℉ |
| 18800 | 1 | 3 | 3969 | 15876 | 3783.39 | 15133.6 |
| 17900 | 1 | 3 | 3838 | 15352 | 3454.61 | 13818.4 |
| 11600 | 1 | 4 | 2124 | 10620 | 1949.96 | 9749.82 |
| 3600 | 4 | 8 | 700 | 8400 | 609.365 | 7312.38 |
| 17300 | 1 | 3 | 3105 | 12420 | 3074.32 | 12297.3 |

**Table 4. Scheduling Tests of Testing Case One--II(MAXC=20000)**

| $\mathbf{x}_D$ | $X_R$ (℉) | | Scheduling Objectives | | | |
|---|---|---|---|---|---|---|
| $x_S$ (Steps) | $x_{VCPU}$ | $x_{VMEM}$ | t(s) | c | ℉ | ℉ |
| 18800 | 1 | 3 | 3969 | 15876 | 3783.39 | 15133.6 |
| 17900 | 2 | 3 | 3357 | 16785 | 3436.83 | 17184.2 |
| 11600 | 1 | 9 | 2089 | 20890 | 1919.98 | 19199.8 |
| 3600 | 4 | 8 | 700 | 8400 | 609.365 | 7312.38 |
| 17300 | 4 | 3 | 3070 | 21490 | 2831.24 | 19818.7 |

**Table 5. Scheduling Tests of Testing Case One--III(MAXC=30000)**

| $\mathbf{x}_D$ | $X_R$ (℉) | | Scheduling Objectives | | | |
|---|---|---|---|---|---|---|
| $x_S$ (Steps) | $x_{VCPU}$ | $x_{VMEM}$ | t(s) | c | ℉ | ℉ |
| 18800 | 6 | 3 | 2972 | 26748 | 3064.83 | 27583.4 |
| 17900 | 6 | 3 | 2706 | 24354 | 2740.76 | 24666.9 |
| 11600 | 1 | 12 | 1887 | 24531 | 1908.95 | 24816.3 |
| 3600 | 4 | 8 | 700 | 8400 | 609.365 | 7312.38 |
| 17300 | 6 | 3 | 2522 | 22698 | 2505.54 | 22549.8 |

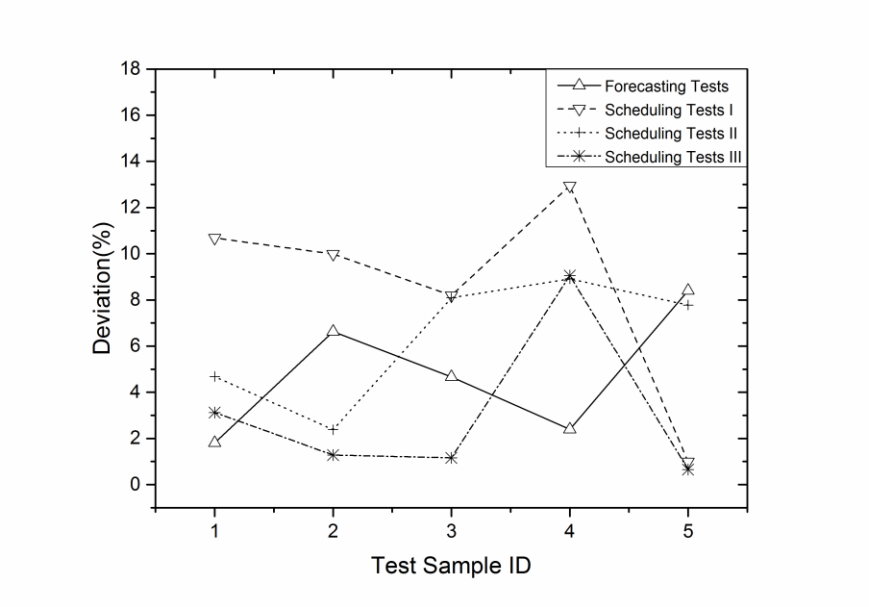


**Figure 6. Forecasting & Scheduling Performances of Test Case One**

Basically, the Kriging surrogate model based forecasting is reliable: the predicted objective values are quite close to the real ones, with possibilities larger or smaller compared to them. With cost limits clearly stated, the optimal scheduling options could be given in most of the time and a few exceptions may be occurred when the limits are too strict for the test samples (sub-tests I and II, test sample 1 for example) or the limits are too loose for the test samples (e.g. sub-tests I, II and III, test sample 4).

### 5.2. Testing Case Two

Warpage is an important quality index in injection molding field. This application aims at reducing the warpage of the thin-walled plastic injection molding products (see Figure 7). EI-based sequential optimization method is employed for this engineering optimization problem. Therefore, the original optimization objective is substituted by the EI function. In order to search the maximum EI value, GA is taken into consideration.

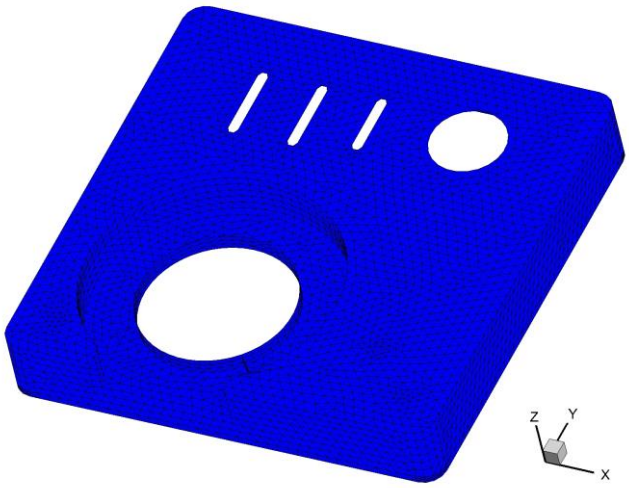


**Figure 7. App II**

In this case, the vector of $x_D$ is defined as $x_D = [x_S, x_G, x_P]$ with ranges in [2,20], [20,100] and [20,100] separately. $x_S$ stands for the total iterations for this optimization problem, $x_G$ means the generations in each iteration for GA, $x_P$ represents the value of the population size of GA. The 30 samples of this testing case is available in Table. 6. Tables 7-10 respectively shows the forecasting tests, and the three sub-tests (cost limits set to 50000, 100000 and 150000) of the scheduling tests. Meanwhile, Figure 8 gives all of the performances belonging to this testing case.

**Table 6. Samples of Testing Case Two**

| $\mathbf{x}_D$ | | | $\mathbf{x}_R$ | | Scheduling Objectives | |
|---|---|---|---|---|---|---|
| $x_I$ (Steps) | $x_G$ | $x_P$ | $x_{VCPU}$ | $x_{VMEM}$ | t(s) | c |
| 17 | 75 | 95 | 2 | 6 | 13761 | 110088 |
| 8 | 20 | 30 | 2 | 10 | 7458 | 89496 |
| 12 | 65 | 90 | 3 | 5 | 12749 | 101992 |
| 7 | 25 | 55 | 1 | 8 | 8260 | 74340 |
| 17 | 85 | 70 | 1 | 6 | 18260 | 127820 |
| 7 | 60 | 35 | 5 | 8 | 6078.5 | 79020.5 |
| 5 | 20 | 55 | 6 | 4 | 4773.5 | 47735 |
| 8 | 50 | 35 | 6 | 3 | 8844 | 79596 |
| 16 | 70 | 75 | 4 | 5 | 14354 | 129186 |
| 3 | 25 | 40 | 3 | 11 | 3512.5 | 49175 |
| 2 | 30 | 20 | 1 | 8 | 2760 | 24840 |
| 5 | 25 | 25 | 6 | 10 | 5261 | 84176 |
| 8 | 30 | 45 | 3 | 6 | 7488.5 | 67396.5 |
| 12 | 75 | 85 | 6 | 10 | 11768.5 | 188296 |
| 19 | 80 | 100 | 1 | 9 | 20832 | 208320 |
| 6 | 40 | 45 | 3 | 5 | 5931.5 | 47452 |
| 2 | 25 | 20 | 6 | 12 | 3029.5 | 54531 |
| 2 | 55 | 20 | 4 | 3 | 3041.5 | 21290.5 |
| 12 | 70 | 95 | 6 | 11 | 11356 | 193052 |
| 5 | 55 | 25 | 4 | 11 | 5471 | 82065 |

**Table 7. Forecasting Tests of Testing Case Two**

| $\mathbf{x}_D$ | | | $\mathbf{x}_R$ | | Scheduling Objectives | | | |
|---|---|---|---|---|---|---|---|---|
| $x_I$ (Steps) | $x_G$ | $x_P$ | $x_{VCPU}$ | $x_{VMEM}$ | t(s) | c | % | % |
| 13 | 65 | 95 | 2 | 4 | 12253 | 73518 | 13736.8 | 82421 |
| 8 | 55 | 20 | 3 | 12 | 7763 | 116445 | 6942.91 | 104144 |
| 10 | 55 | 45 | 6 | 8 | 9692 | 135688 | 10305.8 | 144281 |
| 13 | 85 | 75 | 3 | 4 | 12317.5 | 86222 | 12951.7 | 90661.6 |
| 14 | 80 | 75 | 4 | 5 | 13465 | 121185 | 13958.3 | 125625 |

**Table 8. Scheduling Tests of Testing Case Two--I(MAXC=50000)**

| $\mathbf{x}_D$ | | | $X_R$ (Fixed) | | Scheduling Objectives | | | |
|---|---|---|---|---|---|---|---|---|
| $x_I$ (Steps) | $x_G$ | $x_P$ | $x_{VCPU}$ | $x_{VMEM}$ | t(s) | c | ρ% | θ% |
| 13 | 65 | 95 | 1 | 3 | 16013 | 64052 | 16408.7 | 65634.9 |
| 8 | 55 | 20 | 2 | 3 | 10581 | 52905 | 9861.66 | 49308.3 |
| 10 | 55 | 45 | 1 | 3 | 13787 | 55148 | 14361.8 | 57447.2 |
| 13 | 85 | 75 | 1 | 3 | 15412 | 61648 | 16429.9 | 65719.5 |
| 14 | 80 | 75 | 1 | 3 | 16720 | 66880 | 16393.4 | 65573.6 |

**Table 9. Scheduling Tests of Testing Case Two--II(MAXC=100000)**

| $\mathbf{x}_D$ | | | $X_R$ (Fixed) | | Scheduling Objectives | | | |
|---|---|---|---|---|---|---|---|---|
| $x_I$ (Steps) | $x_G$ | $x_P$ | $x_{VCPU}$ | $x_{VMEM}$ | t(s) | c | ρ% | θ% |
| 13 | 85 | 75 | 3 | 4 | 12317.5 | 86222 | 12951.7 | 90661.6 |
| 8 | 55 | 20 | 4 | 10 | 6979 | 97706 | 6756.95 | 94597.3 |
| 10 | 55 | 45 | 4 | 6 | 10710 | 107100 | 9834.86 | 98348.6 |
| 13 | 85 | 75 | 3 | 4 | 12269 | 85883 | 12951.7 | 90661.6 |
| 14 | 80 | 75 | 3 | 5 | 13519 | 108152 | 12296.4 | 98371 |

**Table 10. Scheduling Tests of Testing Case Two--III(MAXC=150000)**

| $\mathbf{x}_D$ | | | $X_R$ (Fixed) | | Scheduling Objectives | | | |
|---|---|---|---|---|---|---|---|---|
| $x_I$ (Steps) | $x_G$ | $x_P$ | $x_{VCPU}$ | $x_{VMEM}$ | t(s) | c | ρ% | θ% |
| 13 | 65 | 95 | 3 | 9 | 11481 | 137772 | 11814.1 | 141769 |
| 8 | 55 | 20 | 4 | 11 | 6188 | 92820 | 6752.4 | 101286 |
| 10 | 55 | 45 | 5 | 12 | 8480 | 144160 | 8523.29 | 144896 |
| 13 | 85 | 75 | 3 | 9 | 11843 | 142116 | 11855.2 | 142262 |
| 14 | 80 | 75 | 3 | 10 | 12141 | 157833 | 11363.9 | 147731 |

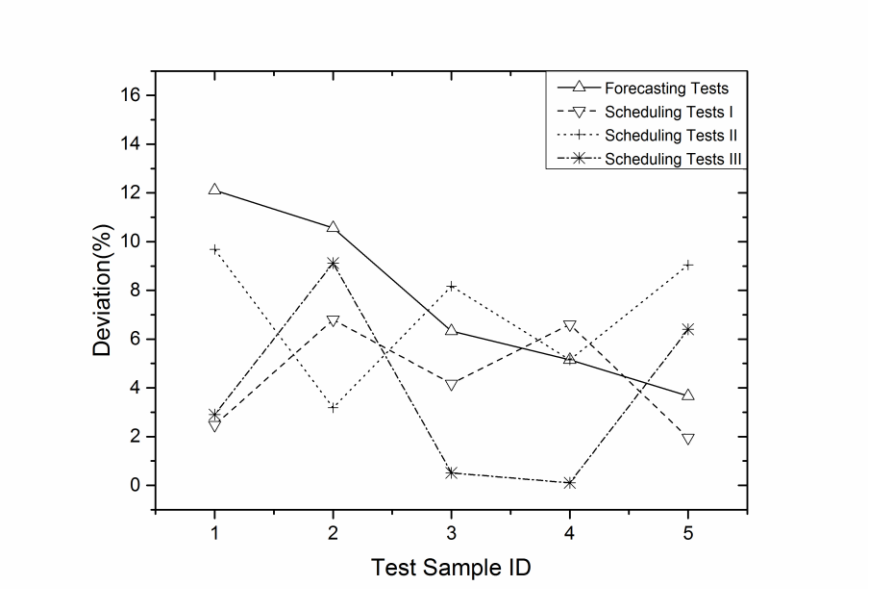


**Figure 8. Forecasting & Scheduling Performances of Test Case Two**

Similarly, the forecasting tests have made uneven performances for all, with not perfectly matches but acceptable predictions based on surrogate model. To this testing case, the first cost limit (50000) seems to be really hard for being met with almost all the testing samples, the latter two larger cost limits are much more better. Even giving a quite large cost limit, the resources may still be impossible for getting larger any more as they are almost the larger boundaries of the resources (sub-test III, testing sample 4), which is a common phenomenon in constrained optimization problems.

### 5.3. Results Analysis and Discussion

As a surrogate model, Kriging may give the predicted conclusion hardly to become the true value, which is why normally uneven deviations existing in the above test cases. however, the Kriging based forecasting strategy is quite useful for the users as there would be no necessary for them to waste the valuable time on checking the status of the tasks with the estimated termination time provided, not to mention the fact that the forecasting will be more and more accurate with tasks accumulated day after day and become the samples for the surrogate model.

As for another key scheduling objective in the cloud platforms, cost or the resources consumed by the cloud tasks have been measured through the virtual types. With testing data shown in the figures and tables, one can tell that the solutions calculated from the optimization algorithm could be trustful. The optimization on the resources to all of the cloud tasks may make a great effort for the customers: they will get their tasks done in proper computing pool (domain) as soon as they could be under the acceptable resource ranges defined, the resources and cost will be spent on the right time and the right place. Besides, this strategy would give an advice on customers when the budget (cost) should be improved or cut from an reasonable surrogate model.

Meanwhile, the Kriging surrogate model based on the samples and the responses is also played an important role for the scientific designers to work out more sufficient applications in the future, which is very meaningful on the application level. The designers will be aware of what kinds of resources the applications mainly rely on and how to use them wisely with the improvements on the programs they have conducted.

The paper has developed a model of $x = (x_D, x_R)$ for the design variables to the resource-intensive scientific computing cloud applications, which has been further

divided into scientific designer part and cloud designer part. If the forecasting or the optimal scheduling is not accurate as expected, more design variables should be summarized from whichever the part related. For instance, the testing cases only brought out two kinds of resources corresponding to the clouds based on the applications we developed, while in many cases it could be expanded for considering the virtual disks at the same time. Moreover, other optimization algorithms could also be applied instead, not only the gradient-based as shown in this paper. In particular, the exhaustive algorithm may also be extremely effective on cases that small possible ranges defined according to the discrete property of the optimization issue itself.

## 6. Conclusion

For the resource-intensive scientific computing applications, a traditional implementation approach will run them in the local PCs or submitted onto standard computer servers to accomplish , and wait for the termination by checking the status of the tasks time after time. Besides the high performance hardware which accelerate the progress of the computing, cloud computing also provides a "pay-as-you-go" kind of services for executing these applications in virtual types. The virtualization for both the resources and the applications makes the resource-intensive scientific computing problems into services that cloud be measured using resource quantities of consumed as the cost for the tasks, with them recycled and rescheduled for the other applications and tasks afterwards. A cloud platform for general scientific computing applications is not hard to establish with the help of the so many open-source cloud infrastructures in nowadays, and the platform developed in this article has built on an OpenStack one.

To help the customers getting the estimated termination time for their submitted tasks, we have proposed a Kriging surrogate model forecasting system between the samples (values of both scientific designer related design variables and cloud designer related design variables) and the responses (termination time for the tasks). Normally, these samples and responses data could be obtained from the historical statistical logs, and more samples could improve the accuracy of the prediction. Meanwhile, as for supporting the customers to choose the proper resource combinations for the computing tasks, we have developed a resource scheduling system based on Kriging surrogate model and a constrained optimization scheduling problem. Through the two testing cases presented, we can see that the forecasting and scheduling system could be trusted for real resource-intensive scientific computing cloud applications. We highly hope that, with the virtualization and cloud techniques getting practical continuously, more and more scientific computing cloud applications will be investigated and the system proposed in this paper could be used more widely in the near future.

## Acknowledgements

The work proposed in this paper has sponsored by the following funds: the National Basic Research Program of China (No. 2012CB025905), the Fundamental Research Funds for the Central Universities of China (No. DUT14RC(3)060), the National Natural Science Funds of China (No. 11202049) and the 111 Project (No. B14013).

## References

[1] H. W. Coleman and W. G.Steele, "Engineering application of experimental uncertainty analysis", AIAA Journal, vol. 33, no. 10, (**1995**).
[2] S. S. Rao, "Engineering optimization: theory and practice", John Wiley and Sons, (**2009**), New York.
[3] J. H. Ferziger, "Numerical methods for engineering application", Wiley, (**1981**), New York.
[4] F. Berman, G. Fox and A. J. Hey, "Grid computing", making the global infrastructure a reality, John Wiley and sons, (**2003**), New York.

 

[5] F. T. Leighton, "Introduction to parallel algorithms and architectures", Morgan Kaufmann, (**1992**), San Francisco.
[6] Q. Chen and Q. Deng, "Cloud computing and its key techniques. Journal of Computer Applications, vol. 29, no. 9, (**2009**).
[7] B. T. Ograph and Y. R. Morgens, "Cloud computing", Communications of the ACM, vol. 51, no. 7, (**2008**).
[8] S. Marston, Z. Li, S. Bandyopadhyay, J. Zhang and A. Ghalsasi, "Cloud computing", The business perspective, Decision Support Systems, vol. 51, no. 1, (**2011**).
[9] Z. Shen, L. Li, F. Yan and X. Wu, "Cloud computing system based on trusted computing platform", (**2010**).
[10] N. Sadashiv and S. M. D. Kumar, "Cluster, grid and cloud computing", A detailed comparison, Computer Science & Education (ICCSE), 2011 6th International Conference on, (**2011**) August 3-5, Singapore.
[11] G. von Laszewski, J. Diaz, W. Fugang and G. C. Fox, "Comparison of Multiple Cloud Frameworks", Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on, (**2012**) June 24-29 , Honolulu, HI.
[12] T. Cordeiro, D. Damalio, N. Pereira, P. Endo, A. Palhares, G. Gon C C Alves, D. Sadok, J. Kelner, B. Melander, V. Souza and Others, "Open Source Cloud Computing Platforms", (**2010**).
[13] Z. Shen, L. Li, F. Yan and X. Wu, "Cloud computing system based on trusted computing platform", (**2010**).
[14] L. Wang, J. Tao, M. Kunze and A. C. Castellanos, "David Kramer and Wolfgang Karl". Scientific Cloud Computing, Early Definition and Experience, (**2008**).
[15] S. Ostermann, A. Iosup, N. Yigitbasi, R. Prodan, T. Fahringer and D. Epema, "A performance analysis of EC2 cloud computing services for scientific computing", Cloud Computing, (**2010**).
[16] M. Ambrust, A. Fox, R. Griffith, A. D. Joseh, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica and Others, "A View of Cloud Computing", Communications of the ACM, vol. 53, no. 4, (**2010**).
[17] B. T. Ograph and Y. R. Morgens, "Cloud computing", Communications of the ACM, vol. 51, no. 7, (**2008**).
[18] Y. Huang, N. Bessis, P. Norrington, P. Kuonen and B. Hirsbrunner, "Exploring decentralized dynamic scheduling for grids and clouds using the community-aware scheduling algorithm", Future Generation Computer Systems, vol. 29, no. 1, (**2013**).
[19] Y. Choon Lee, C. Wang, A. Y. Zomaya and B. B. Zhou, "Profit-driven scheduling for cloud services with data access awareness", Journal of Parallel and Distributed Computing, vol. 72, no. 4, (**2012**).
[20] S. K. Garg, C. S. Yeo, A. Anandasivam and R. Buyya, "Environment-conscious scheduling of HPC applications on distributed cloud-oriented data centers", Journal of Parallel and Distributed Computing, vol. 71, no. 6, (**2011**).
[21] M. Mezmaz, N. Melab, Y. Kessaci, Y. C. Lee, E. G. Talbi, A. Y. Zomaya and D. Tuyttens, "A parallel bi-objective hybrid metaheuristic for energy-aware scheduling for cloud computing systems", Journal of Parallel and Distributed Computing, vol. 71, no. 11, (**2011**).
[22] K. Li, G. Xu, G. Zhao, Y. Dong and D. Wang, "Cloud task scheduling based on load balancing ant colony optimization", Chinagrid Conference (ChinaGrid), 2011 Sixth Annual, (**2011**).
[23] S Pandey, L. Wu, S. M. Guru and R. Buyya, "A particle swarm optimization-based heuristic for scheduling workflow applications in cloud computing environments", Advanced Information Networking and Applications (AINA), 2010 24th IEEE International Conference on, (**2010**).
[24] F. Berman, R. Wolski, S. Figueira, J. Schopf and G. Shao, "Application-level scheduling on distributed heterogeneous networks" Supercomputing, 1996, Proceedings of the 1996 ACM/IEEE Conference on, (**1996**).
[25] P. Li, B. Ravindran, S. Suhaib and S. Feizabadi, "A formally verified application-level framework for real-time scheduling on posix real-time operating systems" Software Engineering, IEEE Transactions on, vol. 30, no. 9, (**2004**).
[26] V. Toporkov, "Application-level and job-flow scheduling: an approach for achieving quality of service in distributed computing", (**2009**), pp. 350-359, Springer.
[27] H. Zhu, B. Smith and T. Yang, "Scheduling optimization for resource-intensive web requests on server clusters", Proceedings of the eleventh annual ACM symposium on Parallel algorithms and architectures, (**1999**).
[28] A. Aziz and H. El-Rewini, "Grid resource allocation and task scheduling for resource intensive applications", Parallel Processing Workshops, 2006, ICPP 2006 Workshops, 2006 International Conference on, (**2006**).
[29] S. Verboven, K. Vanmechelen and J. Broeckhove, "Black box scheduling for resource intensive virtual machine workloads with interference models", Future Generation Computer Systems, vol. 29, no. 8, (**2013**).
[30] J Lee, S Leyffer, P. E. Gill and E. Wong, "Sequential Quadratic Programming Methods", Jon Lee and Sven Leyffer, (**2012**), pp. 147-224, Springer New York.
[31] M. B. Biggs, "Sequential Quadratic Programming", (**2008**), pp. 1-14, Springer US.
[32] M. Björkman and K. Holmström, "Global optimization using DIRECT algorithm in matlab", (**1999**).

 

[33] J. S. Ryu, M. S. Kim, K. J. Cha, T. H. Lee and D. H. Choi, "Kriging interpolation methods in geostatistics and DACE model", KSME International Journal, vol. 16, no. 5, (**2002**).
[34] T. W. Simpson, T. M. Mauery, J. J. Korte and F. Mistree, "Kriging models for global approximation in simulation-based multidisciplinary design optimization", AIAA Journal, vol. 39, no. 12, (**2001**).
[35] S. N. Lophaven, H. B. Nielsen and J. Søndergaard, "DACE-A Matlab Kriging toolbox", version 2.0. **(2002)**.
[36] S. Zhao, L. Li, J. Yang, C. Xu, X. Ling and S. Huang, "Deployment and Performance Evaluation of Virtual Network based on OpenStack". (**2013**).
[37] I. Campos, E. Fernández-del-Castillo, S. Heinemeyer, A. Lopez-Garcia, F. Pahlen and G. Borges, "Phenomenology tools on cloud infrastructures using OpenStack", The European Physical Journal C, (**2013**).
[38] T. Cordeiro, D. Damalio, N. Pereira, P. Endo, A. Palhares, G. Gon C C Alves, D. Sadok, J. Kelner, B. Melander, V. Souza and Others, "Open Source Cloud Computing Platforms", (**2010**).
[39] P. T. Endo, G. E. Gon, C C Alves, J. Kelner and D. Sadok, "A survey on open-source cloud computing solutions", (**2010**).

## Authors

**Zhaojun Li (1986)**, Ph. D. candidate at Dalian University of Technology, majoring in computer application technology. Research topics are high performance computing technologies including parallel, grid and cloud computing, etc. He is also a sponsored research associate in Cardiff Engineering School (2013-2014), head designer for optimization cloud.

**Xinyu Wang (1986)**, Ph.D candidate at Dalian University of Technology, majoring in computational mechanics. His research interests incorporate mold design, optimization algorithms and so on.

**Zheng Li (1982)**, Lecturer at Department of Engineering Mechanics, Dalian University of Technology. Research topics are systematic optimization, mold design and high performance computing. He is head designer for an 863 Project about mold design.

**Xicheng Wang (1946-2013)**, Professor at Department of Engineering Mechanics, Dalian University of Technology. He has interests in optimization algorithms, high performance computing techniques and drug design methods. He is also a multidisciplinary supervisor for graduate students with majors of engineering mechanics, computer application technology and molecular biology, etc.

**Keqiu Li (1972)**, Professor at School of Computer Science and Technology, Dalian University of Technology. The main research areas include networking, cloud computing and big data. He is also the associate editor for IEEE Trans. on Computers and IEEE Trans. on Parallel and Distributed Systems. He had earned a National Outstanding Youth Science Foundation fund of China in 2013.

 