





# A Generic Arrival Process Model for Generating Hybrid Cloud Workload

Chunyan An , Jian-tao Zhou , and Zefeng Mou

Inner Mongolia University, Huhhot 010021, China  
cszhoujiantao@qq.com

**Abstract.** In cloud computing, the arrival process of user requests is becoming more diversiform with the globalization of users and the popularization of mobile technology. Moreover, the workloads in cloud computing are tending towards a hybrid of more applications types. It is hardly for the traditional arrival process models to cover the ever-increasing new arrival processes in reality. For that, we propose a general and flexible arrival process model to describe various arrival processes. At the same time, we present a unified generation algorithm to generate the corresponding workload arrival instance based on the arrival process model automatically. The model defines the arrival process by four steps: firstly defines the number of intervals during the workload lifetime, then defines the length of each time interval, next defines the number of requests arriving during each time interval, lastly defines the arrival time points during each time interval. In the case study, we use the generic arrival process model to describe three arrival process models of typical cloud application types and a custom arrival process model, and present corresponding arrival instances using the generation algorithm. The cases showed the flexibility and extensibility of the model. The model and algorithm are simple and generic and are more approaching to realistic hybrid arrival processes.

**Keywords:** Cloud computing · Cloud workload generation  
Generic arrival process

## 1 Introduction

Cloud applications are reaching more and more industries, such as online social network services, user data processing services, online videos, customer relationship management applications, online mass games, online booking and online banking. Different cloud applications have different workload patterns. At the same time, the global distribution of cloud users and the continuous development of mobile technology make workload patterns more complex. For example, because the users are in different time zones, the new diurnal pattern may become a superposition of multiple traditional diurnal patterns. In most real-world cloud scenarios, the workload population is hybrid with more than one cloud applications, and the workload patterns are more and more diversified. To mimic the cloud workloads closer to reality, a generic workload model is needed. A generic arrival process model plays a key role which should cover a variety of arrival processes and independent of application types.

The limitations of existing arrival process models for cloud workload generation can be broadly classified into two types. One limitation is the arrival process models are too unitary, for example only for specific applications or mimic some specific scenario such as burst. Another limitation is the arrival process models are not general and scalable enough to cover a variety of arrival processes. For instance, most current arrival models are defined as arrival time points that are consistent with an independent and identical distribution like poisson process mostly. These models lack the time-dependent characteristics such as periodicity. Though in recent years, the arrival process models based on MAP (Markov arrival process) were studied to describe some time-dependent characteristics. Unfortunately, the number of states and the generation matrix of MAP models depend on different applications or different scenarios. As a result, the arrival models lack generality and calculations for workload generation will multiply as the number of states increases. Therefore, the MAP model is not suitable for use as a generic arrival process model to generate hybrid workloads.

In this paper, a hierarchical generic arrival process model was developed. The main characteristics of various arrival process models were captured and were independent of different applications types. At the same time, we proposed a unified algorithm to automatically generate arrival process instances that conform to the arrival model definition.

The arrival process model is defined by four steps: (1) the duration of workload which is defined as the number of time intervals, (2) the variation of each time interval, (3) the variation of the requests numbers within each time intervals, (4) each arrival time points within each metering interval in final.

Our generation algorithm supports that each step can be defined as a constant, or a statistical distribution/process, or a self-defined function. Then the algorithm automatically generates an arrival process instance based on the defined arrival process model. The generation of sample arrival time points conforms to the corresponding definition and has random characteristics. The option of self-defined function further enhances the extensibility and flexibility of the generic arrival process model.

In the case study following the model definition, we demonstrated how to use the generic arrival model to define the arrival process models for three representative cloud applications types. A Web application, a batch application and a MapReduce application were picked. Additionally, an arrival process model included a self-defined function was illustrated. Furthermore, the corresponding four arrival process instances were generated based on their arrival process models.

The rest of this article is organized as follows: Sect. 2 analyses the related works involved in arrival process models and generation algorithms. Section 3 details the generic arrival process model and its generation algorithm, as well as four example demonstrations. Section 4 summarizes our work and future directions.

## 2 Related Works

A number of researchers have investigated modeling and simulation of arrival processes. There are three typical representations of arrival process: point process, count process and rate process [1]. The random variable in point process is the interval

between adjacent arrival time points. The random variable in count process is the number of points arrived at each equally spaced interval  $T$ . The random variable in rate process is the normalized count, which equals to the number of points in each time interval divided by time interval  $T$ . The point process is the most accurate because it requires all the arrival time points. Count process and rate process lose the accurate time points with interval  $T$ . However, if the research problem does not care about the accurate arrival time points, the latter two processes could be chosen. For instance, in [2] different arrival rates were defined for different types of jobs. In the existing studies about arrival process model, some studies adopted one representation, others mentioned above combined representations.

Next, the related works about arrival process model are classified into three types: i.i.d. first-order model, temporal dependent model and hierarchical model.

At present, the first-order arrival models mostly adopted point process representation. That is, the intervals sequence conforms to some distribution or some statistical process. The most common assumption was the arrival process followed the Poisson Process [3, 4]. Some researchers argued that this assumption are unrealistic. They built their arrival process models by fitting real data. For example, in [5] the arrival of batch jobs conformed to Weibull distribution. In [6] the arrival intervals were in accordance with Pareto distribution by fitting the traces from the Google internal data center. In [7] the interarrival times in private cloud workloads were well modeled by a 3-phase Hyper-Exponential distribution. The author found the model with 5 parameters was more realistically than the lognormal and Pareto models with 2 parameters. And in [8] the author used a queueing system to model a cloud system with a lot of servers, and interarrival times were denoted as a phase-type distribution. The phase-type distribution is composed of adjustable number of exponential phases. This phase-type arrival process model is a general i.i.d. first-order model because any distributions can be generated closely by a combination of these exponential phases [9].

The i.i.d. first-order arrival process models describe the statistical distribution of interarrival times within a period of time, but not the temporal dependence between different time periods. Namely, they cannot define the temporal features such as periodicities, burstiness and self-similarity. To address these shortcomings, a MAP (Markov Arrival Process) model was proposed in [10] to represent the distribution and correlation of arrival times. Meantime, the author [10] pointed out that the MAP model is a superset of i.i.d. first-order models, and also gave a method to fit the MAP model. MAP models define that arrival process can be in different states. During each state holding period, the arrival process conforms to a certain distribution, which is generally exponential distribution [11], called MMPP (Markov Modulated Poisson Process), or other distribution [12], called semi-Markov process, and the state transition is defined by an intensity matrix. In recent years, some researchers believe MAP models are more realistic than the i.i.d. first-order processes. In [13], the arrival process of a Web application was modeled by MAP. The work in [14] fitted the interarrival times for Grid level jobs through Poisson, Interrupted Poisson Process (IPP), MMPP2, MMPP3, and MMPP4 models. The result was that MMPP2 is closer to the real data in changeability than Poisson and IPP. Although MAP model introduces more realistic temporal correlation, the state definitions and generation matrix are dependent on applications and workload scenarios. Moreover, the complexity of model definition

will be multiplied with the increase of the number of states. Therefore, MAP models are not suitable for generating diverse hybrid cloud workloads in terms of generality and scalability.

In order to more accurately grasp the characteristics of the arrival process, some studies have adopted hierarchical models. This is similar to our modeling approach. In [15] the author built a two-level arrival process model to describe the access to a file system. Firstly the access times were divided into groups by clustering method. Then three features were used to define arrival process: the interval between clusters, the number of accesses within a cluster and the interarrival times within a cluster. Lastly, the arrival process model was proved to synthesize access instances of a distributed replicated file system close to the original data. Another research [1] fitted a LRD (Long Range Dependent) arrival process model in two steps. First, the arrival rate process was fitted by a multifractal wavelet model. Second, Controlled-Variability InF was made to convert rate process to arrival time points. After the two steps a completely determined LRD arrival instance were generated. Strictly speaking, the above two studies only gave the hierarchical method to generate arrival process instances which could be close to real trace logs. Neither of them proposed arrival process model formally. Besides, they cannot be general applicable in other types of applications or scenarios. The hierarchical arrival process model we proposed combines the counting process with point process, captures the main characteristics of the arrival process, and maximizes the flexibility of the model with time-section and hierarchical methods. This model can be used to specify diverse arrival scenarios for different applications.

At present, most cloud workload generation tools are built in the Benchmarks. These workload generators typically enable users to generate the required workloads by defining the distributions or parameter values of the arrival process. Most arrival process models in Benchmarks are i.i.d. first-order model. For example, a popular Web Benchmark: Rubis [16] defined that user session length and think time between sessions conformed to negative exponential distributions. Some researchers [17] adjusted the arrival process model of Rubis, where the request arrival rates can be set. And at a request arrival rate, the requests arrived accord with uniform distribution. A “standard” Benchmark for NOSQL cloud system is: Yahoo! Cloud Serving Benchmark (YCSB) [18]. The total number of operations and different throughputs can be configured before workload generation. Then during each timeframe, the interarrival times are generated conformed to uniform distribution. The generated arrival process based on arrival rate lost the variability of arrival time points within a timeframe. SPEC Cloud™ IaaS 2016 [19] use open source CBTool [20] to generate workload. CBTool allows users to set the duration of a workload, the maximum number of requests, and the distribution of arrival process. Our workload generation tool is more flexible than CBTool. Four features can be configured: the number of intervals during the workload lifetime, the length of each time interval, the number of requests arriving during each time interval and the arrival time points during each time interval.

We noticed there are some recent Benchmarks turned to generate workloads based on MAP models. For example, BURSE [21] was proposed to generate workloads with spikes and self-similarity according to a MMPP model. However, MMPP model is complex and not intuitive for users. And one model is only true usually of one specific application or some specific scenarios. As a consequence, MMPP model is not suitable

for a general workload generation tool. By contrast, our arrival process model and generation algorithm are more simple, general and intuitive. Our work laid a solid foundation for generating hybrid cloud workloads. The work about a general workload model and generation tool in cloud computing can be referred to [22].

### 3 A Generic Arrival Process Model

For simplicity and generalization of the arrival process model, the inner features which are specific to applications are out of the question. For example, for web applications, only the arrival time points of the first request in each session are considered. The other requests in sessions are taken no account because the requests arrived interdependently. Similarly, for MapReduce application, only the arrival time points of each job are included. The start-times of mapper and reducer in each job are excluded. In other words, our model is concerned only the variations in the number and frequency of user requests which are common for different applications. A generic model defined the interdependencies of a cloud application you can refer to our work [22]. Another aspect requires further explanation, the generic arrival model defined one arrival at a time, but it can easily be extended to the batch arrival process by adding a bulk-number parameter.

In this section, the generic arrival process model will be reported in three parts: (1) the mathematical specification of the general arrival process model, (2) the instances generation algorithm based on the arrival process model, (3) case study for three typical cloud applications and one arrival process model with self-defined function.

#### 3.1 Formal Specification of a Generic Arrival Process Model

In this section the arrival process is formally defined by a hierarchical mathematical model. For the convenience of the reader, Table 1 lists a summary of the main annotations in the order introduced in the paper.

**Table 1.** The annotations in the model

Annotation	Data type	Definition
Duration	Integer	The duration of workload
$F(\Delta T_x), \{\Delta T_x, x = 1, 2, \dots, \text{Duration}\}$	Stochastic process	$\Delta T_x$ is the length of xth time interval
$\theta(\cdot)$	Function	$\Delta T_x$ conforms to function $\theta(\cdot)$ , $\Delta T_x \sim \theta(\cdot), x = 1, 2, \dots, \text{Duration}$
$\text{FN}^{R-U}(\Delta T_x), \{\text{N}^{R-U}(\Delta T_x), x = 1, 2, \dots, \text{Duration}\}$	Stochastic process	$\text{N}^{R-U}(\Delta T_x)$ is the quantity of arrived requests in $\Delta T_x$ (the xth interval)
$\delta(\cdot)$	Function	$\text{N}^{R-U}(\Delta T_x)$ conforms to function $\delta(\cdot)$ , $\text{N}^{R-U}(\Delta T_x) \sim \delta(\cdot)$
$F(T(x, k)), \{T(x, k), k = 1, 2, \dots, \text{N}^{R-U}(\Delta T_x)\}$	Stochastic process	$T(x, k)$ is the kth arrival time point within the xth time interval
$\{\text{Func}_x(\cdot), x = 1, 2, \dots, \text{Duration}\}$	A set of functions	$\text{Func}_x(\cdot)$ is the function that the arrival time points within the xth time interval conforms to

A generic arrival process model that can be used to generate hybrid cloud workload requires two conditions as follows

- □ Generality: the model can grasp the essential characteristics of the arrival process for different cloud applications. And more complex and diverse arrival processes can be generated by superposition.
- □ Accuracy: the arrival process model combines counting process and point process to generate time-dependent workloads without losing the accurate arrival time points.

In this model, firstly the total time of the workload is defined in terms of the number of time intervals included. Then, the length of each time interval and the number of requests arrived within each time interval can be constant or variable. And the arrival time points within each time interval can be defined individually. This enables the arrival process model not only to describe the accurate arrival points within a time interval but also to describe the temporal dependence among different time intervals, such as periodicity, bursty, variability and self-similarity. The generic arrival process model is defined in four steps as follows.

1. The total time: *Duration*

An arrival process is divided into several time intervals. The total time of an arrival process is defined as a positive integer *Duration*, which represents the number of time intervals included in an arrival process. The length of each time interval could be unequal.

2. The length of each time interval

A total arrival process is split into *Duration* continuous and disjoint time intervals. The length of each time interval could be equal, noted as a constant  $\Delta T$ . It can also be variable, the variation of the lengths can be represented as a stochastic process  $F(\Delta T_x)\{\Delta T_x, x = 1, 2, \dots, Duration\}$ , where  $\Delta T_x$  is a random variable which represents the length of  $x$  th time interval. Let  $\theta(\cdot)$  be a function used to determine the variation of the interval lengths. The relationship between  $F(\Delta T_x)$  and  $\theta(\cdot)$  is denoted as

$$F(\Delta T_x) \sim \theta(\cdot) \quad (1)$$

3. The quantity of arrivals in each interval

The variation of the arrival numbers in each time interval is denoted as a stochastic process  $FN^{R-U}(\Delta T_x), \{N^{R-U}(\Delta T_x), x = 1, 2, \dots, Duration\}$ , where  $N^{R-U}(\Delta T_x)$  is the number of arrived requests within  $\Delta T_x$  (the  $x$ th time interval). Let  $\delta(\cdot)$  be a function used to determine the variation of the arrival numbers in each time interval. The relationship between  $FN^{R-U}(\Delta T_x)$  and  $\delta(\cdot)$  is denoted as

$$FN^{R-U}(\Delta T_x) \sim \delta(\cdot) \quad (2)$$

A special case is that all time intervals are equal, so we simplified the stochastic process as  $FN^{R-U}(x), \{N^{R-U}(x), x = 1, 2, \dots\}$ , where  $N^{R-U}(x)$  is the quantity of

arrivals in the  $x$ th interval. Let  $\delta(\cdot)$  be a function used to determine the variation of the arrival numbers in each time interval. The relationship between  $\text{FN}^{R-U}(x)$  and  $\delta(\cdot)$  is denoted as

$$\text{FN}^{R-U}(x) \sim \delta(\cdot) \quad (3)$$

4. The arrival time points in each time interval.

The arrival point process for each time interval can be defined individually. For each time interval  $\Delta T_x$ ,  $x = 1, 2, \dots, \text{Duration}$ , an arrival point process composed of  $\text{N}^{R-U}(\Delta T_x)$  time points is denoted as a stochastic process

$$F(T(x, k)), \{T(x, k), k = 1, 2, \dots, \text{N}^{R-U}(\Delta T_x)\} \quad (4)$$

where  $T(x, k)$  is the  $k$ th arrival time point within the  $x$ th time interval.  $T(x, k)$ ,  $k = 1, 2, \dots, \text{N}^{R-U}(\Delta T_x)$  should be satisfied with

$$\sum_{l=1}^{x-1} \Delta T_x < T(x, k) \leq \sum_{l=1}^x \Delta T_x, k = 1, 2, \dots, \text{N}^{R-U}(\Delta T_x) \quad (5)$$

We denote a set of functions  $\{\text{Func}_x(\cdot), x = 1, 2, \dots, \text{Duration}\}$ , where  $\text{Func}_x(\cdot)$  is the function that the arrival time points within the  $x$ th time interval conforms to. Namely,

$$F(T(x, k)) \sim \text{Func}_x(\cdot), x = 1, 2, \dots, \text{Duration}, k = 1, 2, \dots, \text{N}^{R-U}(\Delta T_x) \quad (6)$$

A special case is that each arrival point process is consistent, so we simplified the stochastic process and the function as  $F(T(k))$  and  $\text{Func}(\cdot)$  respectively. And they have

$$F(T(k)) \sim \text{Func}(\cdot), k = 1, 2, \dots, \text{N}^{R-U}(\Delta T_x) \quad (7)$$

### 3.2 A Generation Algorithm Based on Arrival Process Model

The input parameters of the generation algorithm are on basis of the generic arrival process model, include: (1) the number of time intervals *Duration*, (2) the function  $\theta(\cdot)$  determining the variation of the interval lengths, (3) the function  $\delta(\cdot)$  determining the variation of the arrival numbers in each time interval, (4) a set of functions  $\{\text{Func}_x(\cdot), x = 1, 2, \dots, \text{Duration}\}$  determining the arrival time points within each time interval conforms to. Here  $\theta(\cdot)$ ,  $\delta(\cdot)$  and every  $\text{Func}_x(\cdot), x = 1, 2, \dots, \text{Duration}$  can be assigned by three methods: constant, statistical distribution or process and self-defined function. Any statistical distribution can be assigned, such as exponential distribution, Weibull distribution. The steps of the algorithm are consistent with the steps of the generic arrival process model. Firstly, a list of  $\{\Delta T_x, x = 1, 2, \dots, \text{Duration}\}$  are generated randomly according to  $\theta(\cdot)$ . Secondly, a list of  $\{\text{N}^{R-U}(\Delta T_x), x = 1, 2, \dots, \text{Duration}\}$  are generated randomly according to  $\delta(\cdot)$ . Lastly, arrival time points  $\{T(x, k), x = 1, 2, \dots, \text{Duration}, k = 1, 2, \dots, \text{N}^{R-U}(\Delta T_x)\}$  are generated randomly

according to the corresponding  $\text{Func}_x()$ ,  $x = 1, 2, \dots, \text{Duration}$ . Table 2 showed the pseudo-code for generating the arrival process instance based on the arrival process model.

**Table 2.** Pseudo-code of generation algorithm

<p>Input: <math>\text{Duration}</math>, <math>\theta(\cdot)</math>, <math>\delta(\cdot)</math>, <math>\text{Func}_x()</math>, <math>x = 1, 2, \dots, \text{Duration}</math>  Output: <math>\{ T(x, k), x = 1, 2, \dots, \text{Duration}, k = 1, 2, \dots, N^{\text{R.U}}(\Delta T_x) \}</math></p> <hr/> <p>-----</p> <p>Begin  <math>i \leftarrow 1</math>  <math>T(1, 1) \leftarrow 0</math>  while (<math>i \leq \text{Duration}</math>)  {      <math>\Delta T_i \leftarrow \theta(\cdot)</math>  }      if (<math>\delta(\Delta T_x)</math> is not defined)      {          <math>i \leftarrow 1</math>          while (<math>i \leq \text{Duration}</math>)          { <math>k \leftarrow 1</math>              while (<math>T(i, k) &lt; \Delta T_i</math>)              {                  <math>T(i, k + 1) \leftarrow T(i, k) + \text{random number that matches the random process } \text{Func}_x()</math>                  <math>k \leftarrow k + 1</math>              }              <math>i \leftarrow i + 1</math>          }      }      else      {          <math>i \leftarrow 1</math>          while (<math>i \leq \text{Duration}</math>)          {              <math>N^{\text{R.U}}(\Delta T_i) \leftarrow \delta(\Delta T_i)</math>              <math>k \leftarrow 1</math>              while (<math>k \leq N^{\text{R.U}}(\Delta T_i)</math>)              {                  <math>T(i, k + 1) \leftarrow T(i, k) + \text{random number that matches the random process } \text{Func}_x()</math>                  <math>k \leftarrow k + 1</math>              }              <math>i \leftarrow i + 1</math>          }      }  } </p>
--



### 3.3 Case Study

To explain the generality of the arrival process model for different cloud applications in detail, we take three typical cloud applications: Web applications [23], MapReduce applications [24] and batch applications [5] as examples. We explained in detail the definitions of the three arrival processes using our generic model, and gave three generated arrival process instances applying our generation algorithm. At the same time, we presented an extra example on the definition of an arrival process model with self-defined function as well as the generated arrival process instance.

#### Web Application Arrival Process Model Example

In this web application workload, the arrival process was divided into rounds. The time of each round is unequal. Within one round, the number of active users is denoted as *Concurrent\_Users*, and each active user initiated one session. *Ramp\_Up\_Period* specifies the time to initiate all the sessions in one round. If all sessions are created at the same time, then *Ramp\_Up\_Period* = 0. Otherwise, the sessions are created one after another at regular intervals. The interval between two adjacent sessions is *Ramp\_Up\_Period/Concurrent\_Users*. For example, there were 5 active users in a round and 10 s of *Ramp\_Up\_Period*, it will take 2 s between each session creation. The length of each session conforms to the negative exponential distribution  $\text{Exp}(15)$ . Thus, the time of each round is defined as

$$F(\Delta T_x) \sim \theta(\cdot) = \text{Ramp\_Up\_Period} + \text{Exp}(15) \quad (8)$$

Because in one round each active user can only create one session, the number of sessions in one round  $N^{R-U}(\Delta T_x)$  is the number of active users. According to the specification in [23], the variation of the arrival numbers in each round is defined as

$$FN^{R-U}(x) \sim \delta(\cdot) = N(10, 3) \quad (9)$$

The arrival time points in round  $x$  is defined as

$$T(x, k) = (k - 1) * \left( \frac{\text{Ramp\_Up\_Period}}{\text{Concurrent\_Users}} \right), k = 1, 2, \dots, N^{R-U}(\Delta T_x) \quad (10)$$

We use the following steps to generate the arrival process instance.

1. Give *Duration* a value
2. Generate *Duration Concurrent\_Users* which are random sampled by normal distribution  $N(10, 3)$
3. Generate *Concurrent\_Users* session lengths which are random sampled by negative exponential distribution  $\text{Exp}(15)$ . In each round, the length of a round is given by the sum of *Ramp\_Up\_Period* and the maximum session length of the samples.
4. In each round, *Ramp\_Up\_Period* is set as 800 s, then the interarrival time is  $(\text{Ramp\_Up\_Period}/\text{Concurrent\_Users})$

Because of limited space, Fig. 1 showed an arrival process instance with *Duration* = 4, the number of sessions in 4 rounds are 9, 11, 13, 5, the 4 session lengths are 823.6955 s, 849.5503 s, 826.9957 s, 826.9957 s. We take the start time of the round as the first arrival point. And the interarrival time in each round is in turn  $Ramp\_Up\_Period/Concurrent\_Users = 88.8889$  s, 72.7273 s, 61.5385 s, 160 s.

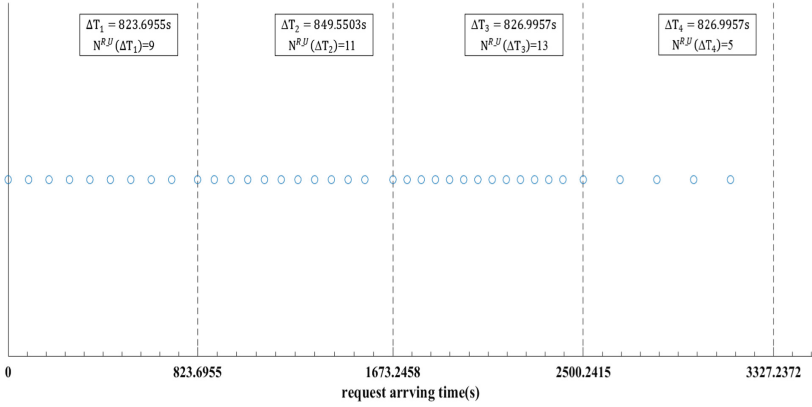


Fig. 1. A web application arrival process instance

**MapReduce Application Arrival Process Model Example**

The arrival process model [24] is first-order. The interarrival time conformed to *Weibull*(20, 0.5). As a result, it is not necessary to divide the arrival process model into time intervals, then *Duration* = 1; And the interarrival time  $\Delta t_k$  conforms to *Weibull*(20, 0.5), that is  $Func(\cdot) \sim Weibull(20, 0.5)$ , thus, the kth arrival time point is equal to

$$T(1, k) = T(1, (k - 1)) + \Delta t_k, k = 1, 2, \dots \tag{11}$$

We use the following steps to generate the arrival process instance.

1. *Duration* = 1,  $\Delta T = 6000$  s
2. Randomly generate sample points conforming to *Weibull*(20, 0.5) as interarrival times until the arrival time point is beyond  $\Delta T$ .

Figure 2 shows an instance of the arrival process, which generates 20 arrivals within 6000 s.

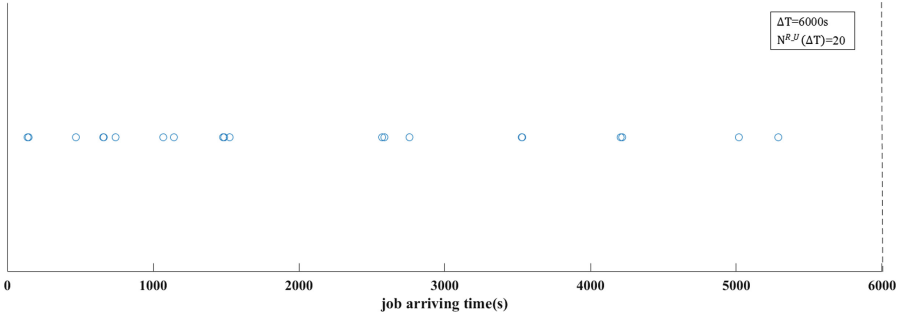


Fig. 2. A MapReduce application arrival process instance

**Batch Application Arrival Process Model Example**

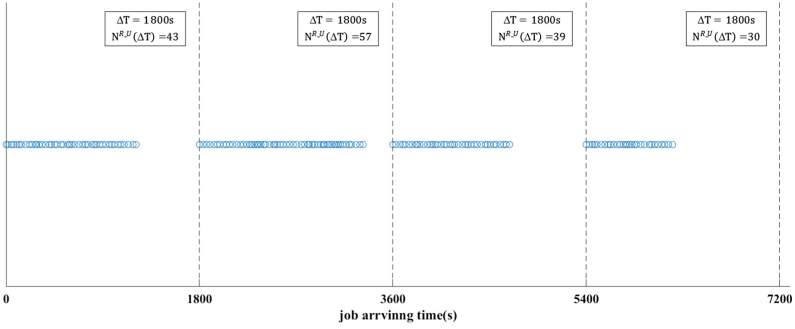
The arrival process model [5] appears Daily-cycle pattern. The day is first divided into 48 intervals by half an hour, i.e.  $\Delta T = 1800$  s. Then the variation of the number of arrivals during each of 48 intervals is defined to conform to Weibull distribution, i.e.  $\delta(\cdot) \sim W(1.79, 24.16)$ . The hours from 8AM to 5PM are called “peak hours”. The variation of the interarrival time  $\Delta t_k$  conforms to also Weibull distribution with different parameters, i.e.  $Func(\cdot) \sim W(4.25, 7.86)$ . Thus, the arrival time points in each interval can be defined as

$$T(x, k) = T(x, k - 1) + \Delta t_k \tag{12}$$

We use the following steps to generate the arrival process instance.

1. *Duration* = 48,  $\Delta T = 1800$  s
2. Randomly generate 48 sample points conforming to  $W(1.79, 24.16)$  as the number of arrivals within each of 48 intervals
3. During the peak hours, the interarrival times in each interval are generated randomly according to the Weibull distribution  $W(4.25, 7.86)$ . The numbers of sample points within each interval are determined by step 2. And in each interval the first arrival time point is the start time of the interval.

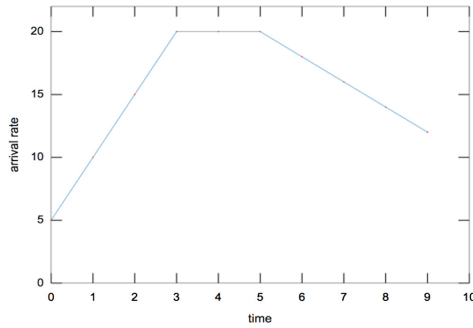
Because the author only gave the interarrival time during the peak hours, also the space is limited, we showed an instance of the arrival process during only the first four time intervals (that is, from 8 to 10) in Fig. 3. A batch arrival process is multiple arrivals at a time. However, we are more concerned with the arrival times in this work, so that the number of jobs in an arrival is not defined.



**Fig. 3.** A batch application arrival process instance

**An Example of Arrival Process Model with Self-defined Functions**

The arrival process model introduces more flexibility by supporting self-defined functions. We presented an example of the arrival process including self-defined functions. Firstly, we gave the specification of the self-define arrival process.



**Fig. 4.** Arrival rate function example

$$\delta(x) = \begin{cases} 5 + 5x & (0 \leq x \leq 3) \\ 20 & (3 \leq x \leq 5) \text{ 且 } \delta(x) = \delta(x - 9) \\ 20 - 2(x - 5) & (5 \leq x \leq 9) \end{cases} \quad (13)$$

As can be seen from Fig. 4, the arrival rate increases linearly from 8 am to 11 am, from average 5 arrivals per hour to 20 arrivals per hour. From 11 pm to 1 pm, the arrival rate remains at average 20 arrivals per hour. From 1 pm to 5 pm, the arrival rate drops linearly until 12 arrivals per hour. After 5 pm it is closed. The time is divided into hours, i.e.  $\Delta T = 1$  h. The arrival time point in an interval conforms to the poisson process, and  $\lambda_i$  of the poisson process is set to the mean number of arrivals in the  $i$ th interval. In Fig. 4, horizontal axis is time with hour as the unit, and vertical axis is

arrival rate. 8 am corresponds to the value 0 on the x-axis. The mathematical function of the variation of arrival rate is defined in Eq. 13.

We specify the access process with our model as follows:

1. The time is divided into hours, i.e.  $\Delta T = 1$  h.
2. The mean number of arrivals in the  $i$ th interval is

$$E(N^{R-U}(i)) = \int_{i-1}^i \delta(x) dx \tag{14}$$

3. The arrival time points in the  $i$ th interval are

$$\{T(i, k), k = 1, 2, \dots, N^{R-U}(i)\}, F_x() \sim \text{Poisson}(E(N^{R-U}(i))) \tag{15}$$

We use the following steps to generate the arrival process instance.

1. *Duration* = 9,  $\Delta T = 1$  h.
2. The number of arrivals in each interval is randomly generated according to the mean number of arrivals in this interval. As shown in Fig. 5, the nine samples are 6, 11, 16, 20, 19, 18, 15, 12.
3. The interarrival times in each interval are generated randomly according to the poisson process with their parameter  $\lambda_i$ .

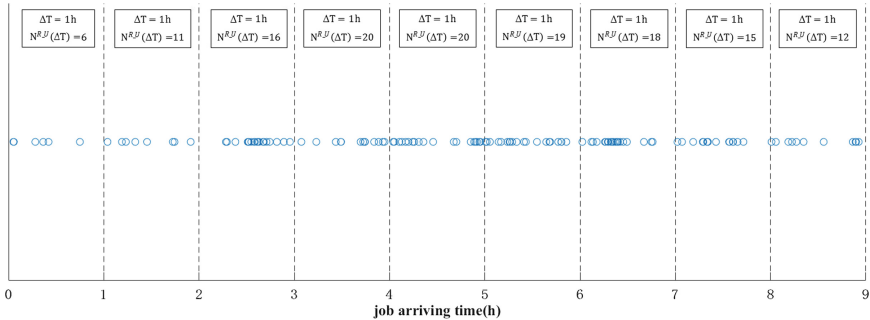


Fig. 5. A self-defined arrival process instance

## 4 Conclusion

This paper presented a general hierarchical arrival process model and an algorithm for generating arrival process instances based on the arrival model.

The general arrival process model was specified in four steps, had two advantages.

- (1) It captured the essential features of arrival process models, was independent of application and workload scenario.
- (2) It combined the advantages of point process and

count process, can not only describe accurate time points in each interval but also describe temporal dependence of intervals.

Our corresponding generation algorithm supports that each of four steps can be defined as a constant, or a statistical distribution/process, or a self-defined function. The option of self-defined function further enhances the extensibility and flexibility of the generic arrival process model. Compared with the existing generation tools, our algorithm is simple and effective, and has stronger scalability.

The case study showed the generality, flexibility and effectiveness of our works. In sum, the generic arrival process model and generation algorithm will provide a solid foundation for generating more realistic hybrid cloud workloads.

In the future, we will further study how to formally define a more complex arrival process model by combined multiple general arrival process models in vertical and horizontal axis of time. In vertical axis of time, a complicated arrival process model can be the superposition of multiple parallel arrival processes model. In horizontal axis of time, a time-dependent arrival process model can be defined by multiple consecutive arrival process models. We believe that the study will make the general arrival process model more comprehensive.

**Acknowledgement.** The authors wish to thank Natural Science Foundation of China under Grant No. 61662054, 61262082, 61562064 and 61462066, Natural Science Foundation of Inner Mongolia under Grand No. 2015MS0608 and 2018MS06029, Inner Mongolia Science and Technology Innovation Team of Cloud Computing and Software Engineering and Inner Mongolia Application Technology Research and Development Funding Project “Mutual Creation Service Platform Research and Development Based on Service Optimizing and Operation Integrating”, Inner Mongolia Engineering Lab of Cloud Computing and Service Software and Inner Mongolia Engineering Lab of Big Data Analysis Technology.

## References

1. Li, H.: Realistic workload modeling and its performance impacts in large-scale science grids. *IEEE Trans. Parallel Distrib. Syst.* **21**(4), 480–493 (2010)
2. Guo, M., Guan, Q., Ke, W.: Optimal Scheduling of VMs in Queueing Cloud Computing Systems with a Heterogeneous Workload, vol. 6 (2018)
3. Vakili, S., Ali, M.M., Qiu, D.: Modeling of the resource allocation in cloud computing centers. *Comput. Netw.* **91**, 453–470 (2015)
4. Lin, A.D., Li, C.S., Liao, W., Franke, H.: Capacity optimization for resource pooling in virtualized data centers with composable systems. *IEEE Trans. Parallel Distrib. Syst.* **29**(2), 324–337 (2018)
5. Iosup, A., Sonmez, O., Anoop, S., Epema, D.: The performance of bags-of-tasks in large-scale distributed systems. In: *Proceedings of the 17th International Symposium on High Performance Distributed Computing—HPDC 2008*, p. 97 (2008)
6. Costa, G.D., Grange, L., Courchelle, I.D., Costa, G.D., Grange, L., Courchelle, I.D.: Modeling and generating large-scale google-like workload (2016)
7. Wolski, R., Brevik, J.: Using parametric models to represent Private cloud workloads. *IEEE Trans. Serv. Comput.* **7**(4), 714–725 (2014)

8. Atmaca, T., Begin, T., Brandwajn, A., Castel-Taleb, H.: Performance evaluation of cloud computing centers with general arrivals and service. *IEEE Trans. Parallel Distrib. Syst.* **27**(8), 2341–2348 (2016)
9. Bolch, et al.: *Queueing Networks and Markov Chains*. Wiley, New York (1998)
10. Casale, G.: Building accurate workload models using Markovian arrival processes. In: *Proceedings of the ACM SIGMETRICS Joint International Conference on Measurement and Modeling of Computer Systems - SIGMETRICS 2011*, p. 357 (2011)
11. Meier-Hellstern, K., Fischer, W.: The Markov-modulated Poisson process (MMPP) cookbook. *Perform. Eval.* **18**(18), 149–171 (1993)
12. Wang, E., Yang, Y., Wu, J., Liu, W., Wang, X.: An efficient prediction-based user recruitment for mobile crowdsensing. *IEEE Trans. Mob. Comput.* **17**(1), 1 (2017)
13. Pacheco-Sanchez, S., Casale, G., Scotney, B., McClean, S., Parr, G., Dawson, S.: Markovian workload characterization for QoS prediction in the cloud. In: *Proceedings—2011 IEEE 4th International Conference on Cloud Computing CLOUD 2011*, pp. 147–154 (2011)
14. Li, H., Muskulus, M., Wolters, L.: Modeling job arrivals in a data-intensive grid. *Job Sched. Strateg. Parallel Process.* **4376**, 210–231 (2007)
15. Ware, P.P., Page, T.W., Nelson, B.L.: Automatic modeling of file system workloads using two-level arrival processes. *ACM Trans. Model. Comput. Simul.* **8**(3), 305–330 (1998)
16. RUBiS. <http://rubis.ow2.org/> (2018)
17. Wilkes, J.: PRESS: PRedictive Elastic ReSource Scaling for cloud systems. In: *2010 International Conference on Network and Service Management*, pp. 9–16 (2010)
18. YCSB. <https://github.com/brianfrankcooper/YCSB/wiki>
19. Cloud, S., et al.: SPEC Cloud™ IaaS 2016 Benchmark Design Overview, pp. 1–37 (2016)
20. CBTOOL. <https://github.com/ibmcb/cbtool/tree/master/scripts>
21. Yin, J., Lu, X., Zhao, X., Chen, H., Liu, X.: BURSE: a bursty and self-similar workload generator for cloud computing. *IEEE Trans. Parallel Distrib. Syst.* **9219** (2014)
22. An, C., Zhou, J., Liu, S., Geihs, K.: A multi-tenant hierarchical modeling for cloud computing workload. *Intell. Autom. Soft Comput.* 1–8 (2016)
23. The Apache Olio Project. <http://incubator.apache.org/olio/>
24. Chen, Y., Ganapathi, A., Griffith, R., Katz, R.: The case for evaluating MapReduce performance using workload suites. In: *2011 IEEE 19th Annual International Symposium on Modelling, Analysis, and Simulation of Computer and Telecommunication Systems*, pp. 390–399 (2011)