November 07, 2018

SUSPENSION POLICY FOR CLOUD-BASED VIRTUAL MACHINES

Ramdoot Kumar Pydipaty

Jithendra Balakrishnan

Amit Kumar Saha

Debo Dutta

SUSPENSION POLICY FOR CLOUD-BASED VIRTUAL MACHINES

AUTHORS:
Ramdoot Kumar Pydipaty
Jithendra Balakrishnan
Amit Kumar Saha
Debo Dutta

ABSTRACT

Techniques are described herein for obtaining usage metrics of Virtual Machines (VMs) over a number of days. This data may be used to automatically identify periodic (e.g., daily) blocks of time when groups of VMs can be suspended. Unlike typical scenarios in which VMs in a cloud-based environment stay up irrespective of whether the VM is being used, these techniques avoid unnecessary cost to the client.

DETAILED DESCRIPTION

Typically, a compute instance's utilization spikes when the user signs in (or some process starts) and falls back to a low level when the user signs off (or the process ends). However, a Virtual Machine (VM) typically stays powered on for the entire duration even though it is not used for the entire period. Over a large collection of VMs there is scope for identifying suspension windows such that each suspension window corresponds to set of VMs that can be suspended during that window.

The overall solution has three major steps. In the first step, the on/off utilization levels for each VM are estimated and the most likely suspension period/candidate for each VM is calculated. In the second step, the suspension candidates for all VMs (or a configurable list of VMs) are combined to come up with a list of VMs, for a specified suspension period/candidate. In the third step, the user specifies how many suspension periods are needed and the top candidates are returned to maximize the total suspension time.

Figure 1 below illustrates one approach to solve the aforementioned problem by collecting utilization data of a VM over some number of days (e.g., 15-20 days).

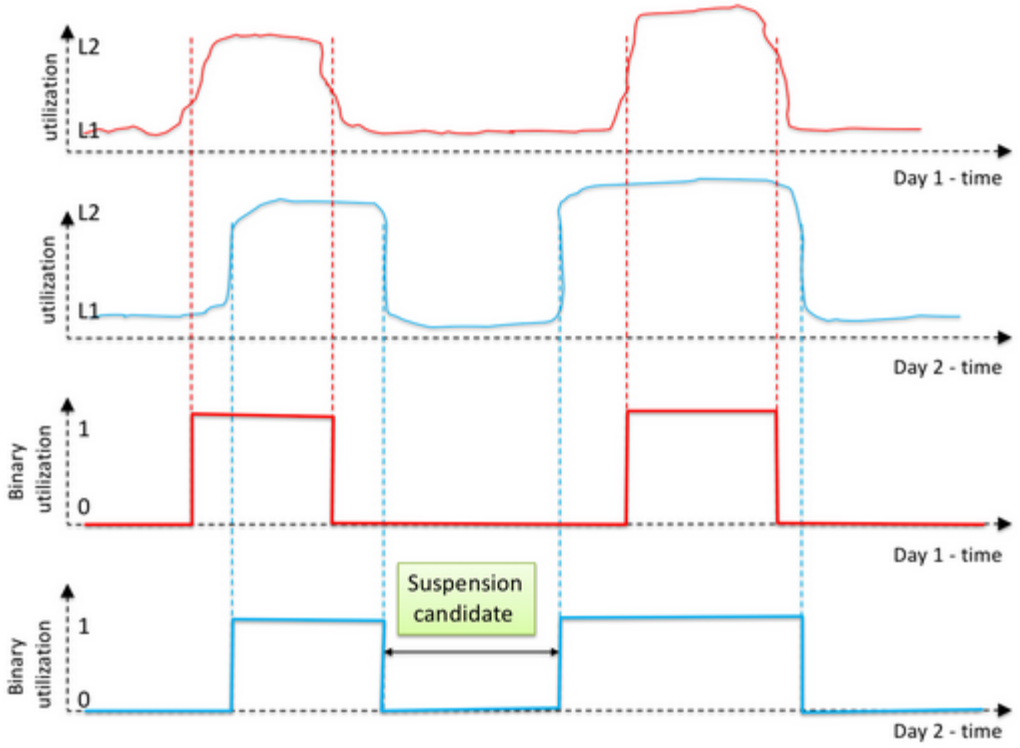

Figure 1

The first step involves estimating on/off utilization and calculating a suspension candidate for a single VM. This step operates on a single VM. The utilization data collected for a single VM is used to calculate the suspension candidates for that VM.

Levels L1 and L2 are estimated for a single day's data. L1 is a low level utilization and L2 is a high level utilization. L1 and L2 may be possibly represent multiple levels (e.g., L11, L12, … for L1 and L21, L22, … for L2 over different days. Essentially, it is a binary representation where L1 corresponds to 0 and L2 corresponds to 1. K-means clustering of the moving/rolling average of the data may provide the cluster centers. A moving average (calculated using a rolling window) may be used to smooth out jitters in the measurement.

A minimum of the calculated cluster centers is considered as L1, subject to a configurable maximum. Other cluster centers which are sufficiently far from L1 (e.g., multiples of L1 or a few standard deviations far from L1) may be logically considered as L2. The data points which are closer to L1 may be mapped to L1, else to L2. Hence, the input time series is transformed to a L1, L2 binary function.

Since values of L1 and L2 are not fixed, it is important to use an automated way to estimate values of L1 and L2. In the absence of an automated way, it is necessary to manually determine the values of L1 and L2, which is clearly not scalable.

Timestamps (e.g., T1, T2, T3, T4, …) may be derived when a shift occurs from L1 to L2 or vice versa, for each day. Here, T1 corresponds to L1 → L2, T2 corresponds to L2 → L1, T3 corresponds to L1 → L2, and so on. Thus, T3 - T2 is a period of under-utilization of the instance.

All overlapping periods of under-utilization (e.g., T3-T2, T5-T4, etc.) may be selected across all days whose lengths are greater than a user-defined/configurable predefined duration threshold D_t. Boundary conditions of days should be accounted for while calculating overlapping periods. Underutilized duration may be aggregated over day boundaries.

These periods are candidates for suspension over all days for the specific VM. One example suspension candidate is shown in green in Figure 1 above. Instead of a deterministic calculation of the suspension period, a predictive approach is taken (e.g., using some standard Machine Learning (ML) approaches). At the end, the suspension periods are candidates (but may also be considered prediction) since future behavior can never be guaranteed to exactly mirror past behavior (which is represented by the collected data).

The second step involves combining the suspension candidates of all VMs. In this step, the suspension candidates for all VMs (calculated in the first step) are combined to produce a set of policies. This may still be too many policies to practically implement. Accordingly, this step allows the user/administrator to specify the number of policies and the policies that are expected to maximize the total suspension time.

Figure 2 below illustrates the creation of hash buckets that span 24 hours. At each hour of the day, there are multiple buckets specifying a duration. For example, at hour zero, the bucket represents a suspension period that starts at hour zero and has a duration of four hours. Since these durations are supposed to represent suspension periods, Figure 2 shows a case where the minimum duration of suspension is two hours with a maximum duration of 24 hours – 2 hours = 22 hours.

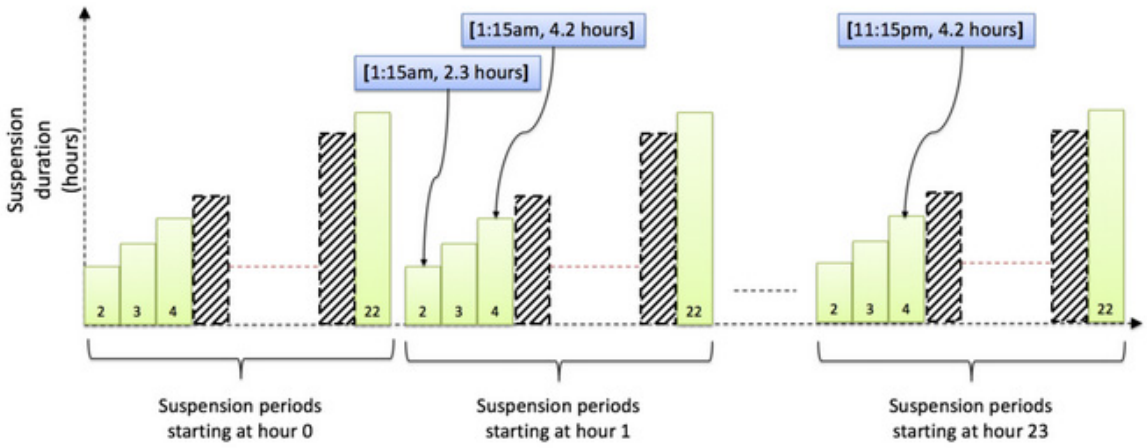

Figure 2

Assume that for VM v_i its suspension candidate is specified as C_i [v_i, s_i, d_i]. This means that (1) it is a suspension for VM v_i, (2) starts at time s_i, and (3) has a duration of d_i hours.

C_i may be hashed into the appropriate bucket. For example, if C_i were [v_i, 1:15am, 4.2 hours] it would get hashed into the bin marked "4" in the bins for suspension periods starting at hour one.

This process may be repeated for all VMs. At the end, all suspension candidates are mapped to the respective hash buckets. For each hash bucket that has more than one entry, the entries are combined so that the final/merge window is an intersection of all the candidate windows in that bucket.

Thus, a list of suspension candidate is provided where each candidate points to a list of VMs that can be suspended during that candidate window. In the worst case, this can lead to as many windows as there are hash buckets (i.e., each suspension window is unique). This is a non-ideal scenario that can lead to an impractically large number of suspension windows. This situation is now handled by the third step.

The third step involves selecting a user defined number of policies. A practical system should not have a large number of suspension windows, and in this step, the user is allowed to specify a maximum number of suspension windows that is expected. Suppose N suspension candidates are expected. The hash buckets are sorted according to a key where the key for each hash bucket is calculated as: calculated_suspension_duration x number_of_vms, where calculated_suspension_duration is the intersection of the candidate

windows in the bucket and number_of_vms is the number of candidate windows in that hash bucket.

In other words, each hash bucket is weighted according to the total duration of suspension for that bucket. For example, a bucket that has a calculated_suspension_duration of 1 hour and affects just 11 VMs (total savings = 11 hours) may be weighed more than a bucket that has a calculated_suspension_duration of 10 hours but affects 1 VM only (total savings = 10 hours). The top candidate suspension windows corresponding to the top N buckets are then returned.

Once the policy is enforced, the suspension policy becomes a fixed policy. The data for the suspended duration is no longer present, so the suspended duration can only increase and cannot decrease. As such, it is suggested to have periods (e.g., of multiple days) when the suspension policies are not effective so that new measurements can be taken and new suspension policies can be calculated.

Depending on the underlying workload, it is quite possible that the candidates for suspension will vary between weekdays and weekends, with weekends presumably having longer periods of low utilization levels. Hence, when using the aforementioned steps, it is suggested to do a higher level filtering to weed out weekdays and weekends.

Since prior history is not a guarantee that a VM will behave exactly the same way in the future, the implementation of suspension policies has to be performed carefully. In particular, the following may be considered when implementing a policy. First, the suspension policy should only be applicable to non-critical VMs. Second, when attempting to suspend a VM, the current utilization of the VM should be monitored to make sure that the VM is indeed in a low utilization state. If the VM is not in a low utilization state that warrants suspension, that particular VM may be removed from the suspension policy so that a fresh measurement can be taken and the new data can be used to come up with suspension candidates with a better fit.

A VM that has a cron job that triggers at a specific time would be active at that time. Hence the monitoring data may sample that time as an active time. Additionally, VMs having cron jobs are not usual practice, especially ones that are launched as part of a coordinated set of VMs implementing a service.

These techniques may be particularly useful where VMs are used in a "pay as you use" model. Moreover, scaling up/down may be performed instead of suspension, for example in scenarios where there are multiple instances of the same VM (with a load balance in front). These techniques may apply to generic collection of VMs or for a set of VMs providing the same functionality that can be scaled down (in a number of instances). In one example, a VM may be suspended so that the user does not incur any cost for simply keeping the VM up.

In summary, techniques are described herein for obtaining usage metrics of VMs over a number of days. This data may be used to automatically identify periodic (e.g., daily) blocks of time when groups of VMs can be suspended. Unlike typical scenarios in which VMs in a cloud-based environment stay up irrespective of whether the VM is being used, these techniques avoid unnecessary cost to the client.