May 31, 2018

A budget-based policy for operating system scheduling of background tasks

Alhaad Gokhale

A budget-based policy for operating system scheduling of background tasks

ABSTRACT

Background applications in an operating system (OS) can be either running or sleeping. If a background application continues running without going to sleep or attempts to wake up with an unnecessarily high frequency, it needlessly consumes system resources, such as power, memory, etc., thus decreasing operational efficiency.

The techniques of this disclosure allocate each background application with a time budget proportional to the number of CPU cycles. The budget decreases with the use of CPU cycles by the background application until the application goes to sleep or runs out of budget. The budget for each background application is replenished over time by the OS, and any sleeping background application with a non-zero budget can be woken up by the OS.

KEYWORDS

- Background applications
- Task scheduling
- CPU allocation
- Time budget
- Application sleep
- Operating system

BACKGROUND

An operating system (OS) of any modern computing device, such as a desktop, laptop, smartphone, etc., is responsible for running applications, many of which are from third parties distinct from the OS provider. Such applications are often not designed explicitly to share resources, such as Central Processing Unit (CPU) time, memory, power, etc., with other applications running concurrently. The framework that manages how the various system resources are shared among applications is provided by the OS. In particular, the OS needs to schedule the allocation of resources to applications running in the background that are not directly interacting with the user. At any given time, a background application can either be running or sleeping while waiting for the OS to wake it up. An application running in the background consumes CPU cycles to perform computations, communicate with other applications, register with the OS to be woken up in response to specific events, or go to sleep.

Background applications are run by the OS either based on user need or in response to non-user-driven events, such as timer alarms, network traffic, etc. In systems connected to a constant power supply, background applications may be scheduled to run whenever CPU time is available. However, such an approach might be suboptimal on battery powered systems since the use of CPU cycles results in draining the battery. As a result, system resources hogged by a misconfigured or malicious background application that uses unnecessarily frequent non-user-driven events puts a strain on battery life.

DESCRIPTION

The techniques of this disclosure describe a strategy that allows an OS to limit the amount of resources used by a background application when responding to non-user-driven events, such as timer alarms, network traffic, etc. The techniques involve allocating each application with a time budget proportional to the number of CPU cycles the application is permitted to consume. The budget decreases with the use of CPU cycles by the background application until the application goes to sleep or runs out of budget. The budget for each background application is replenished over time by the OS, and any sleeping background application with a non-zero budget may be woken up by the OS.

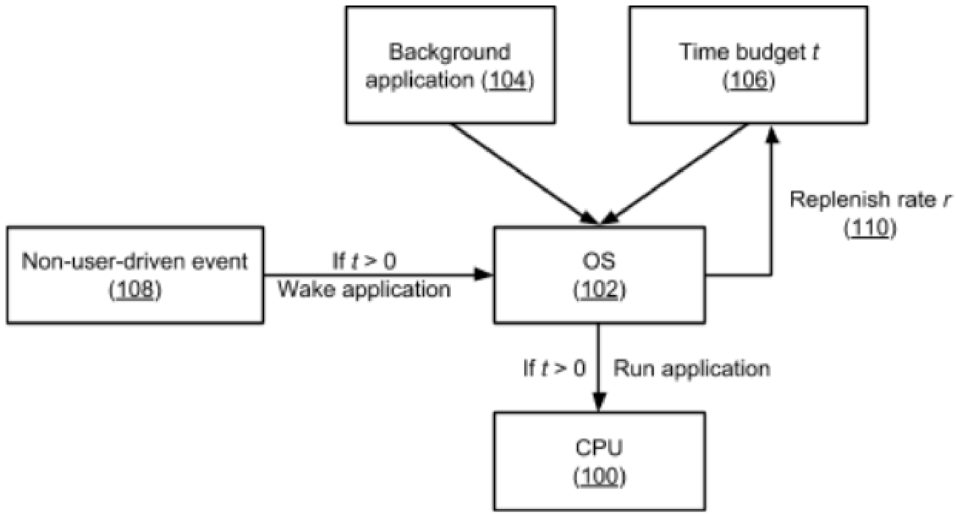

Fig. 1: OS allocation of CPU cycles based on the time budget of a background application

Fig. 1 illustrates an example operationalization of the techniques of this disclosure. The number of cycles of the CPU (100) consumed by an application (104) running in the background is managed by an OS (102). The application is allocated by the OS with a time budget *t* (106), where *t* is a real number proportional to the number of CPU cycles the application is allowed to utilize prior to being terminated by the OS.

The time budget *t* is decremented as CPU cycles are consumed by the application until it goes to sleep, or the time budget *t* reaches 0. If the time budget *t* reaches 0 before the application goes to sleep, the application is terminated by the OS such that it will not be woken up even if has registered to be woken up. On the other hand, if the application goes to sleep while the time budget *t* is greater than 0, it can be woken up by the OS in response to an event (108) for which it has registered to be woken up by the OS.

Upon waking up, the application continues running until it time budget *t* reaches 0 or it goes back to sleep. Over time, the time budget *t* for the application is replenished by the OS at a rate *r* (110), where *r* is a dimensionless number representing the number of CPU cycles added to the time budget *t* for the application per absolute CPU cycle. The rate *r* can be calculated as the principal eigenvector of the normalized adjacency matrix of all background applications running on the OS .

In contrast to approaches that require domain knowledge regarding background applications, such as the purpose or specific tasks performed, the techniques of the disclosure can be applied in a general way without requiring application-specific information. As a result, the techniques can handle any background application an OS may encounter, thus decoupling resource allocation and scheduling policy from specifics of the background application. The time budget for more important background applications gets replenished at a higher rate, thus allowing them relatively larger proportion of CPU cycles compared to less essential background applications.

Further, the techniques of this disclosure provide protection against background applications that make disproportionately high use of CPU cycles, e.g., because of a bug or misconfiguration or due to intentional abuse. In such cases, the background application will end up utilizing CPU cycles at a rate significantly higher than the rate at which its time budget is replenished, eventually running out of time budget allocation and getting terminated by the OS.

In addition to CPU cycles, the techniques of this disclosure can be extended to allocate and manage application access to any OS-controlled resources, such as memory, power, bandwidth, etc.

CONCLUSION

The techniques of this disclosure allocate each background application with a time budget proportional to the number of CPU cycles. The budget decreases with the use of CPU cycles by the background application until the application goes to sleep or runs out of budget. The budget for each background application is replenished over time by the OS, and any sleeping background application with a non-zero budget may be woken up by the OS. In contrast to approaches that require domain knowledge regarding background applications, such as the purpose or specific tasks performed, the techniques of the disclosure can be applied in a general way without requiring application-specific information and can be extended to any OS-controlled resources shared among applications. Further, the techniques provide protection against background applications that make disproportionately high use of resources.