September 07, 2018

Machine-Learning-Assisted Scheduler for Filesystem Input and Output

Mark Schott

Follow this and additional works at: https://www.tdcommons.org/dpubs_series

Recommended Citation
Schott, Mark, "Machine-Learning-Assisted Scheduler for Filesystem Input and Output", Technical Disclosure Commons, (September 07, 2018)
https://www.tdcommons.org/dpubs_series/1492

This work is licensed under a Creative Commons Attribution 4.0 License.
This Article is brought to you for free and open access by Technical Disclosure Commons. It has been accepted for inclusion in Defensive Publications Series by an authorized administrator of Technical Disclosure Commons.
Machine-Learning-Assisted Scheduler for Filesystem Input and Output

Abstract:

Traditional computing systems use simple, fast-to-compute heuristics to inform various decisions, such as allocating fractions of input/output bandwidth when switching among various tasks. Although fast and simple to execute, heuristics alone fail to account for the context of the computing system and the system’s behavior as a whole and cannot make a more-optimal decision that accounts for hardware components, energy resources, or the environment in which the computing device operates. A machine-learning-assisted or neural-network-based scheduler can make inferences or predictions based on the system information, conditions, and dynamics and make a more-optimal decision in allocating system resources during input and output operations. However, a machine-learning-assisted or neural-network-based scheduler can be slow to run, which may interfere with or disrupt normal computing system operations without an appreciable benefit. A hybrid, machine-learning-assisted scheduler can combine insights from a machine-learning model into traditional heuristic rules in such a way that the computing system can make more-accurate predictions with respect to input and output operation scheduling while still enjoying the fast response time of traditional heuristics.

Keywords:

Heuristic, energy, energy consumption, machine-learning, optimize, optimization, neural network, decision, scheduling, input, output, I/O, kernel.

Background:

A traditional computing system includes input and output components that allow a user to both receive information from an outside component or device and transfer information to an outside component or device, such as a portable hard drive, network device, or a USB-connected
camera. The traditional computing system will include a heuristic or rule-based scheduler that
determines which files or data to transfer between the system and the input/output device, a queue
or order of file transfer, how much of a processor to use to perform the transfer, and other logistical
necessities of the input/output operations. Often, the heuristic rules can be used to make a decision
on the order of microseconds, which can efficiently navigate through many input/output operations
and improve user experience. However, the heuristic rules may not account for hardware
components, energy resources, or the operating environment of the computing device in decision
making, which could result in suboptimal input/output operations.

Description:

Traditional computing systems use simple, fast-to-compute heuristics to inform various
decisions, such as allocating fractions of input/output bandwidth when switching among various
tasks. Although fast and simple to execute, heuristics alone fail to account for the context of the
computing system and the system’s behavior as a whole and cannot make a more-optimal decision
that accounts for hardware components, energy resources, or the environment in which the
computing device operates. A machine-learning-assisted or neural-network-based scheduler can
make inferences or predictions based on the system information, conditions, and dynamics and
make a more-optimal decision in allocating system resources during input and output operations.
However, a machine-learning-assisted or neural-network-based scheduler can be slow to run,
which may interfere with or disrupt normal computing system operations without an appreciable
benefit. A hybrid, machine-learning-assisted scheduler can combine insights from a machine-
learning model into traditional heuristic rules in such a way that the computing system can make
more-accurate predictions with respect to input and output operation scheduling while still
enjoying the fast response time of traditional heuristics.
Consider the simple, common computing environment of Figure 1 below. Figure 1 illustrates hardware or processing components of a computing device connected to both the input/output components, such as a portable hard drive, network device, or a USB-connected camera, and the remaining components of the computing device, such as memory, storage, user input, and display devices. The processing components include a heuristic scheduler, which makes decisions regarding how to process information and data flow between the input/output components and the remaining components of the computing device. Normally, heuristic-based schedulers run in a kernel and can make decisions on the order of microseconds in determining how to allocate bandwidth, dynamic priority, or whether to idle on a queue, which are just a few examples of the operations coordinated by the scheduler.

![Figure 1](image)

**Figure 1**

Although practical, useful, and fast, the heuristic scheduler fails to account for operations and context of the system in part or as a whole. Consider the nearly identical environment of Figure 2, which includes the same hardware or processing components, input/output components, and remaining components of the computing device. However, the computing device of Figure 2 uses a machine-learning-assisted or neural-network-based scheduler to manage the input/output operations. Here, the machine-learning-assisted scheduler accounts for energy resource
availability, energy consumption, hardware capability, task priority, or task necessity to make inferences and predictions and optimize overall system performance.

![Diagram](https://www.tdcommons.org/dpubs_series/1492)

**Figure 2**

Many of these additional system characteristics are difficult to incorporate into or account for using traditional heuristics. Although more comprehensive, the machine-learning-assisted scheduler requires significantly more time to schedule tasks, on the order of tens of milliseconds instead of the microseconds of traditional heuristics, and must be calibrated and trained, which can be even slower and more time consuming. The decisions of the machine-learning-assisted scheduler are better but come at a cost.

Figure 3 illustrates a combination or hybrid computing system including a machine-learning-assisted heuristic scheduler that combines insights from a machine-learning model such that the processing hardware or kernel can make more-optimal predictions with respect to input/output scheduling while enjoying the faster response time of traditional heuristics.
Figure 3

Figure 4 illustrates a modified input/output scheduler that incorporates insights from the machine-learning model. The modified scheduler comprises three main components: a userspace daemon handling the machine learning operations, a modifiable heuristic, and a combination module to blend the other two components.
The userspace daemon generates machine-learning inferences based on process data and system-wide input/output statistics, including user-defined priority levels such as a real-time priority and nice values, dynamic input/output scheduler priorities, runtime behaviors such as bytes read or written, percent of allocated bandwidth used, how often the process blocks, and the load on the system. In some circumstances, system performance may be improved by using a recurrent neural network that performs past predictions for the processes or the system as a whole.
Generally, the userspace daemon is trained beforehand and only runs inference calculations in real time. The input to this model can be collected over a set time on the order of hundreds of milliseconds to several seconds. An on-device graphics processing unit or digital signal processor could be helpful in making inferences faster and more energy efficient.

The modifiable heuristic combines portions of the traditional heuristic with portions of the machine-learning model such that neither dictates the entire output. For example, consider the following sample linear combination formula:

First Modifiable Heuristic = \( C_0 + C_1 \times X_1 + C_2 \times X_2 \)

Here, \( C_0, C_1, \) and \( C_2 \) are constants that could be set at compile time or run time, \( X_1 \) is the output of the traditional heuristic, and \( X_2 \) is the output of the machine-learning model. Another variation of a combination formula could be:

Second Modifiable Heuristic = \( M_0 + M_1 \times H_1 \)

Here, \( M_0 \) and \( M_1 \) are the outputs of the machine-learning model and \( H_1 \) is the output of the traditional heuristic. In this second example, the machine-learning model can nudge the output of the desired function in a particular direction over a longer period of time, such as hundreds to thousands of milliseconds, while the traditional heuristic can still vary over much shorter time intervals to react quickly to changes in the system load. Some heuristics that could be modified in such a manner include central processing unit frequency scaling, load balancing, processor affinity, dynamic priority, time slice, and many others. The modifiable heuristic composed of several different parts would also allow the system, under certain circumstances, to disable various components of the machine-learning model, which would allow the system to execute the traditional heuristic without the machine-learning additions.
The combination module or interface allows the userspace daemon and the modifiable heuristic to run without jeopardizing the traditional heuristic operations. For example, the userspace daemon running the machine-learning model can read the data it needs to run an inference from the traditional system files and write back output data to a different set of system files. Advantageously, the original or traditional system files retain their values and the kernel is never waiting for the machine-learning model to run an inference in order to obtain the needed files to manage input/output operations. Indeed, if the kernel were waiting for the machine-learning model, the kernel or other system processing component could be making decisions on slightly outdated data. Here, when the machine-learning model completes its inferences during its normal system monitoring and analysis operations, the kernel or other system processing component can use the data written in the different set of system files as it becomes available to improve overall system performance without being forced to wait. Thus, the system overall can iteratively improve its performance over time but can still enjoy the quick response of the traditional heuristic by obtaining data from the traditional set of system files.

The modified input/output scheduler features several advantages over traditional systems. First, machine-learning components of the system can be removed or decoupled from kernels and take advantage of userspace libraries, which also decreases the amount of privileged code running and may decrease security vulnerabilities. Second, asynchronous updates from the machine-learning model stored in different files prevents the kernel or other system processing component from unnecessarily waiting. Third, the hybrid modified input/output scheduler combines the best of both the traditional heuristic and modifiable heuristic, especially in terms of latency and quality of predictions or inferences. Fourth, the modified input/output scheduler and the modifiable
heuristic can be constructed in any number of different manners to achieve particular system objectives while retaining the speed and benefit of the traditional heuristic.

A hybrid, machine-learning-assisted scheduler can combine insights from a machine-learning model with traditional heuristic rules in such a way that the computing system can make more-accurate predictions with respect to input and output operation scheduling while still enjoying the fast response time of traditional heuristics.