(19) **United States**

(12) **Patent Application Publication** (10) Pub. No.: **US 2018/0267806 A1**

Chirayath Kuttan (43) **Pub. Date:** **Sep. 20, 2018**

(54) **CALCULATING WAIT TIME FOR BATCH SCHEDULER JOBS**

(71) Applicant: **Flexera Software LLC**, Itasca, IL (US)

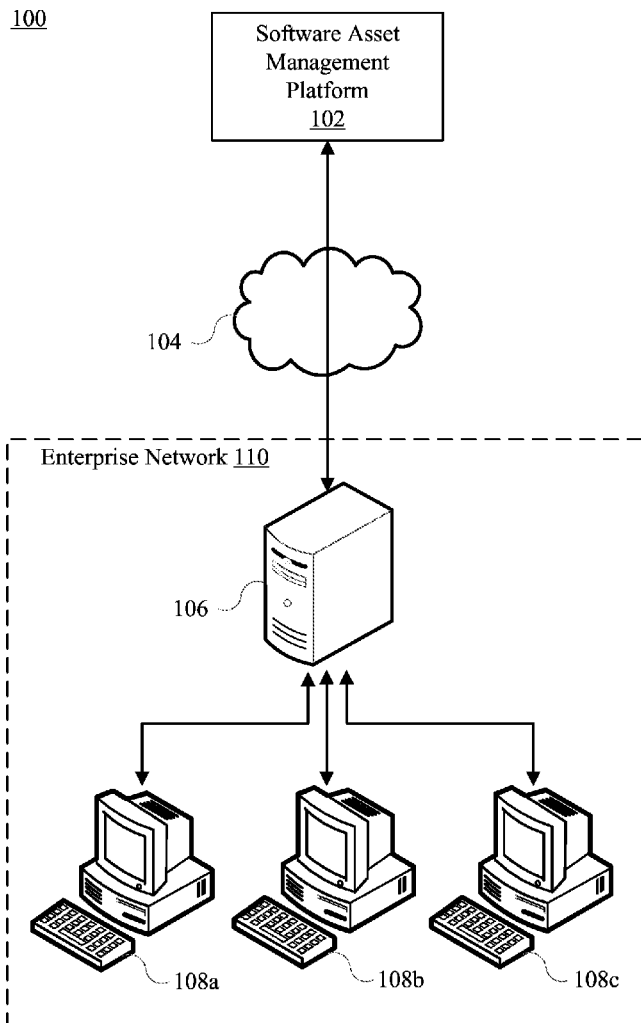(72) Inventor: **Rajeesh Chirayath Kuttan**, Victoria (AU)

(57) **ABSTRACT**

Computer programs and computer-implemented techniques are described here for predicting when jobs in the queue of a batch scheduler will be completed. More specifically, various embodiments are described herein that relate to mechanisms for predicting the wait time and/or the estimated time to completion for jobs that are to be executed by a software asset management platform. For example, heuristics and algorithms could be used to discover when execution of a job is likely to begin and/or end. The estimated time to completion for a given job can be estimated by summing the expected execution time of the given job and the expected execution times of any jobs to be executed prior to the given job, while the wait time for a given job can be estimated by summing the expected execution times of any jobs to be executed prior to the given job.
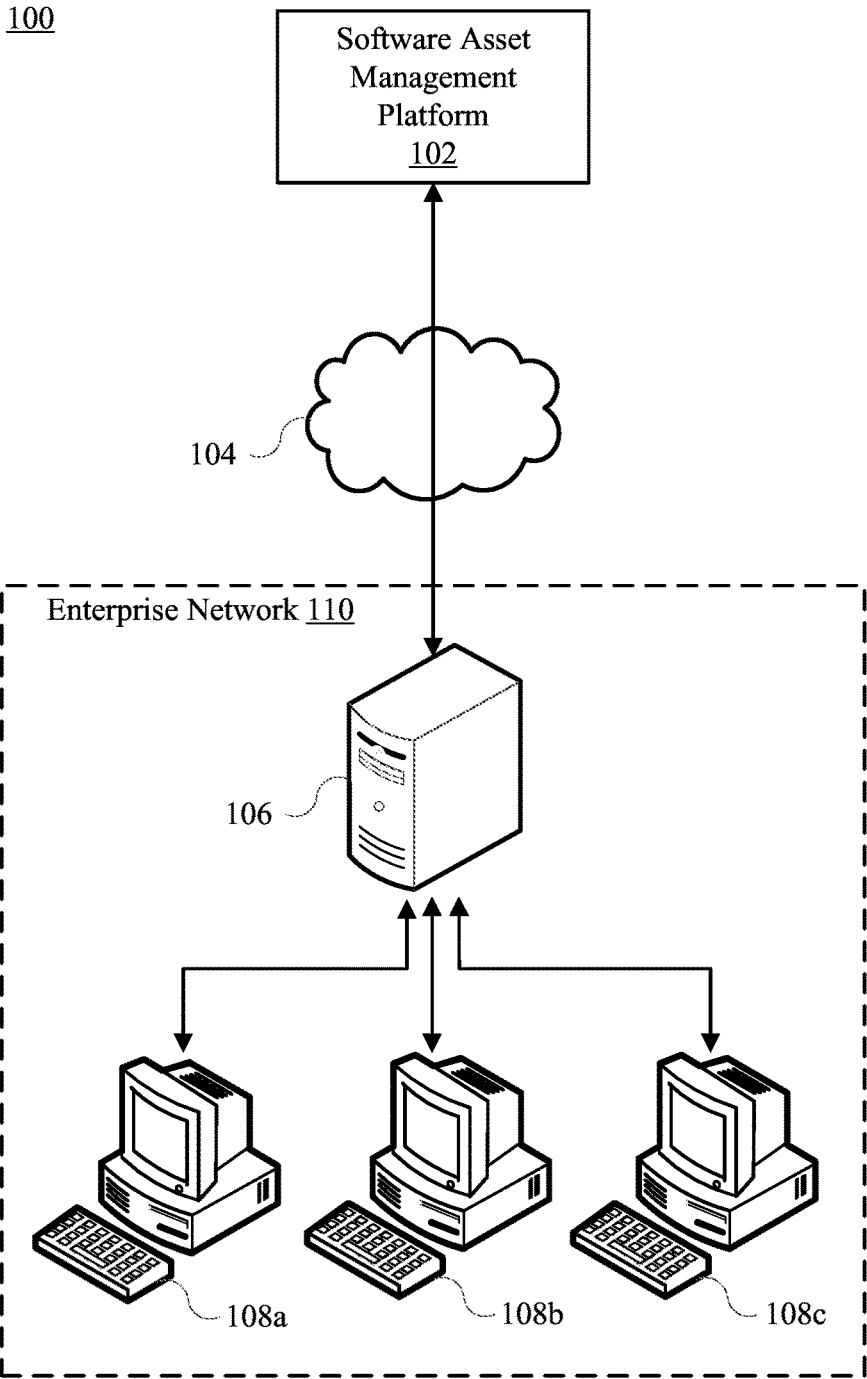
100

100

Software Asset
Management
Platform
102

104

Enterprise Network 110

106

108a          108b          108c

**FIG. 1**

Software Asset Management
Platform 202

Batch Scheduler 204

Job Execution Module 206

Calculator Module 208

GUI Module 210

Storage Module(s) 212

Network
214a

216

Network
214b

Storage
Medium
218

# FIG. 2

**FIG. 3**

400

Execute jobs of different types
401

Create an entry for each executed job by recording the execution time
402

Build a log of historical execution times from the entries
403

Store the log of historical execution times in a data store
404

# FIG. 4

500

Receive input at a user interface indicative of a request to initiate a particular
job
501

Place the particular job in a queue of a batch scheduler
502

Determine that one or more jobs are ahead of the particular job in the queue
503

Identify a corresponding job type for each job of the one or more jobs
504

Compute an expected time to completion for the particular job
505

Post the expected time to completion for the particular job to the user interface
506

# FIG. 5

600
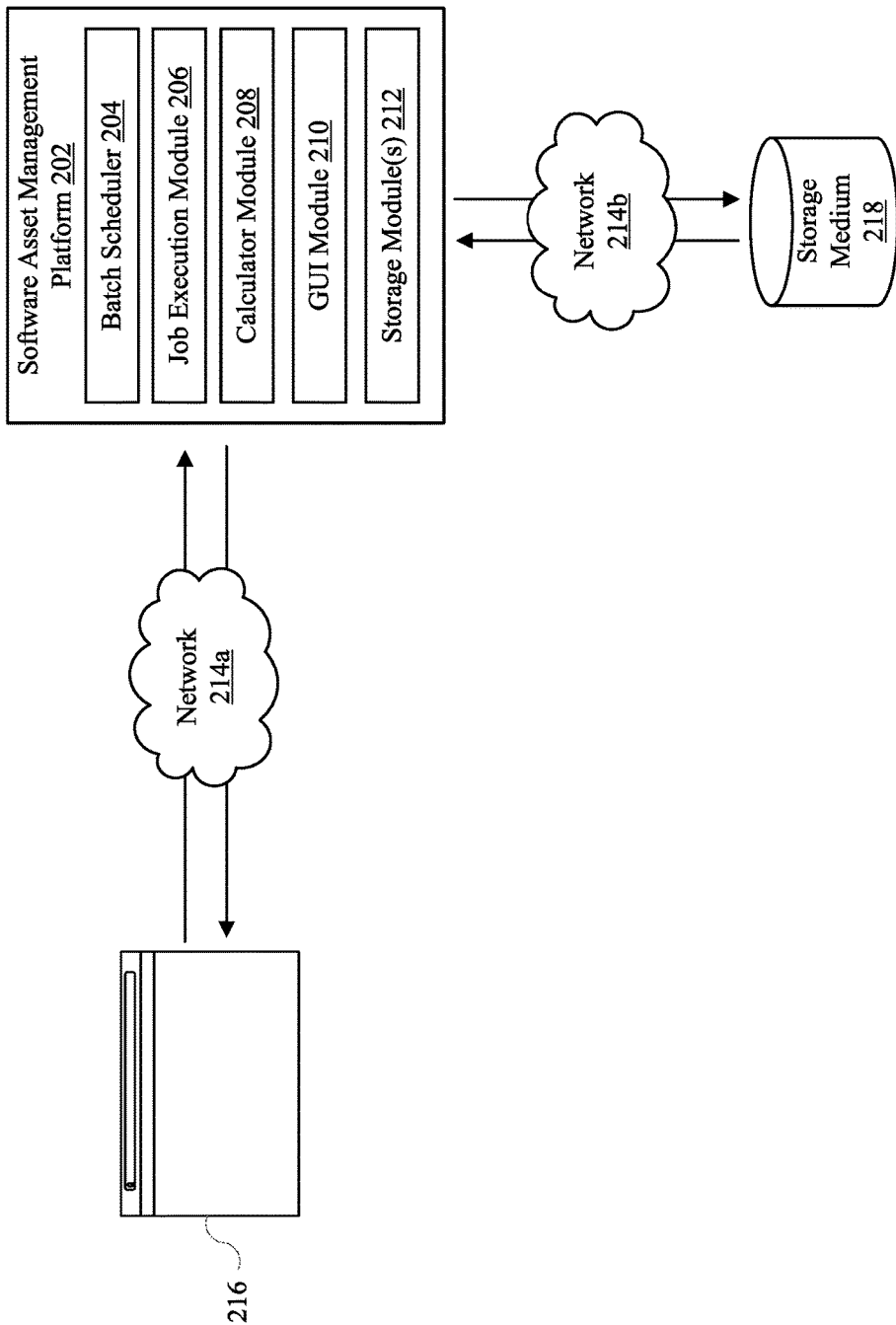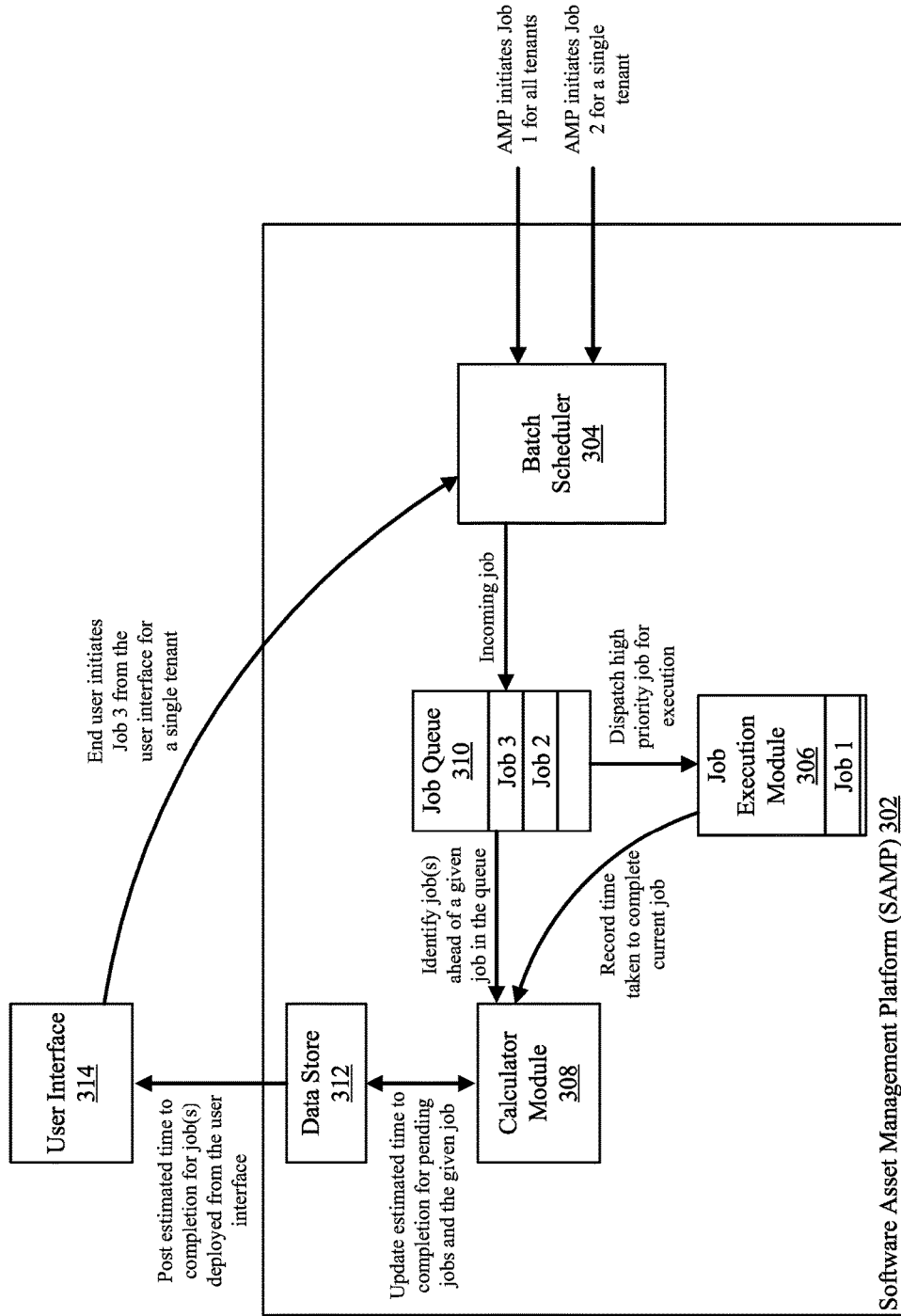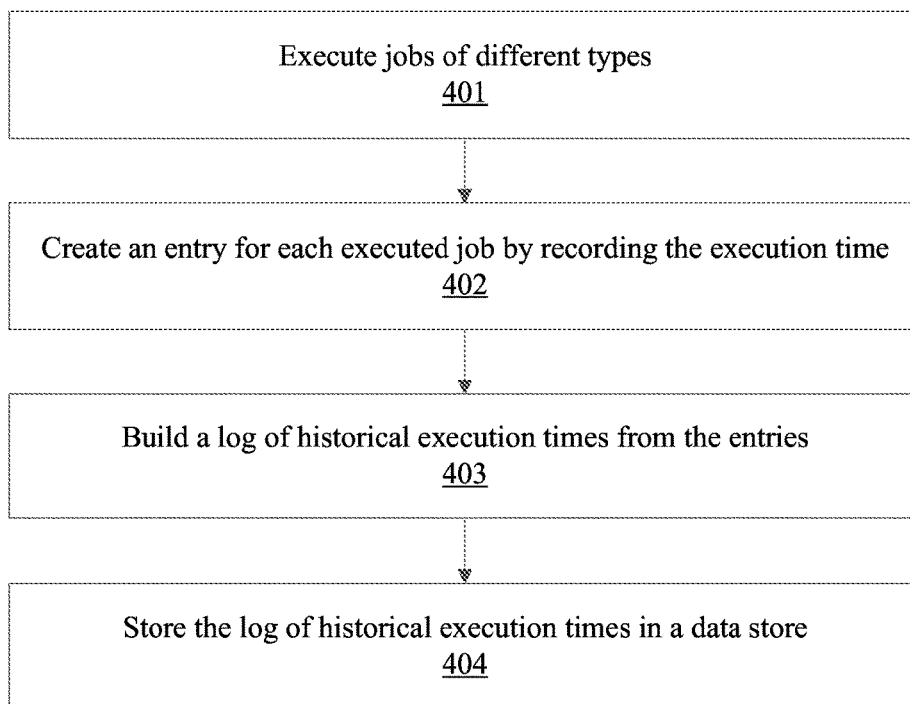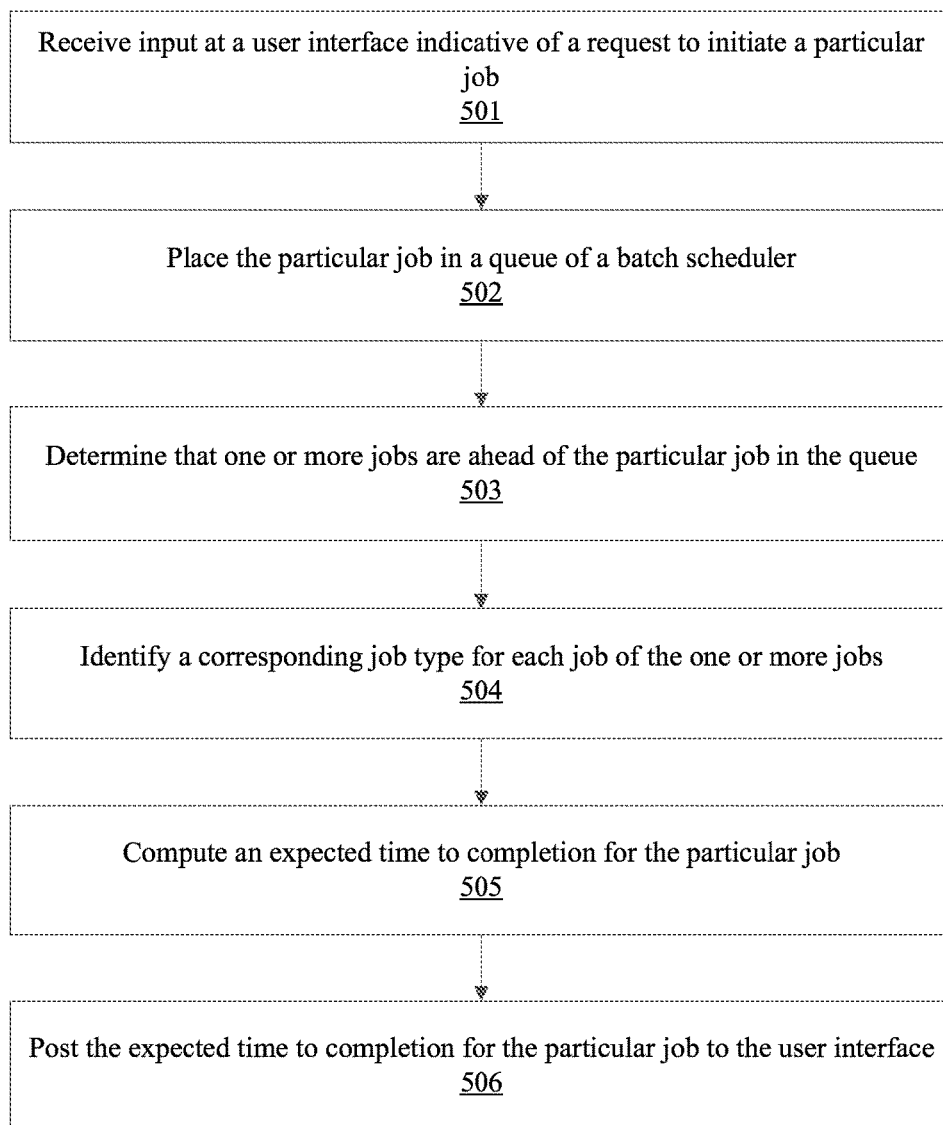


**FIG. 6**

# CALCULATING WAIT TIME FOR BATCH SCHEDULER JOBS

## RELATED FIELD

[0001]   Various embodiments relate generally to providing feedback on when jobs are expected to be finished. More specifically, various embodiments relate to computer programs and computer-implemented techniques for predicting when jobs in a queue of a batch scheduler will be completed.
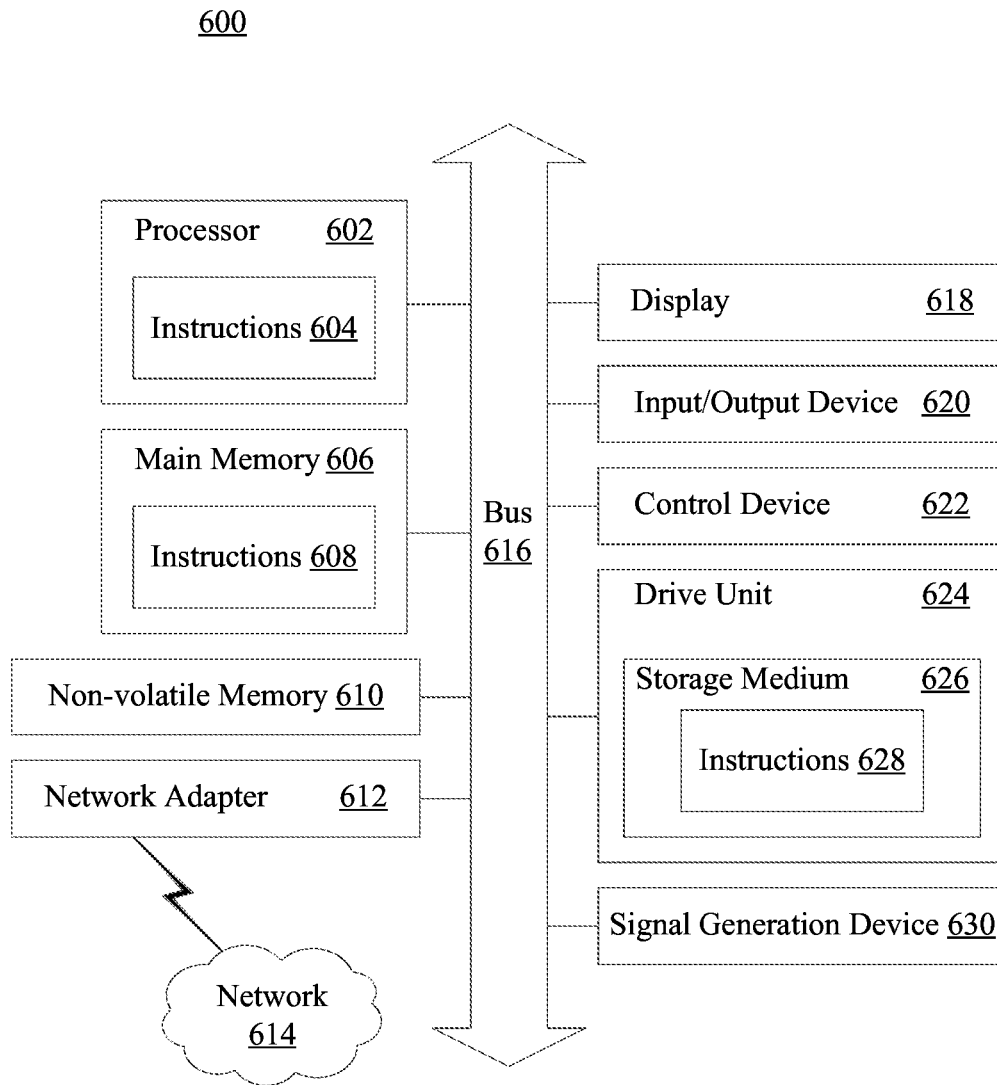
## BACKGROUND

[0002]   Software asset management platforms provide extensive capabilities for managing software from various vendors and desktop, laptop, and server hardware. Software asset management platforms may also provide automated, entitlement-based license optimization for software from vendors such as Adobe®, Microsoft®, and Symantec®. Accordingly, software asset management platforms are often used by end users (e.g., individuals and enterprises) to reduce ongoing software licensing costs, maintain continuous license compliance, and gain better control over software spend while reducing the time and effort required to manage software assets.

[0003]   Graphical user interfaces are often used by the end users to interact with the software asset management platforms. For example, an end user may access a graphical user interface and submit a request to a software asset management platform to execute a particular job (also referred to as a "task"), such as a license reconciliation task, an import of software purchase order details, etc.

[0004]   Software asset management platforms may include a computer application that controls unattended background program execution of jobs. More specifically, the computer application can represent a single point of control for defining and monitoring background executions in a distributed network of computing device in a streamlined manner. Such a computer application is often referred to as a "batch scheduler" or "job scheduler." A batch scheduler can be implemented as a Microsoft Windows® service running on a computing device (e.g., a server).

[0005]   When an end user initiated a job (e.g., by submitting a request through the graphical user interface), there is no indication as to when execution of the job will begin or when the job will be finished. The end user must simply wait until execution has finished and the outcome of the job is available for review, though such a requirement may lead to a negative end user experience in interacting with the software asset management platform.

## SUMMARY

[0006]   Software asset management platforms may include a computer application referred to as a "batch scheduler" or "job scheduler." Said another way, a batch scheduler is a computer application that controls unattended background program execution of jobs. This is commonly called batch scheduling, while execution of jobs is commonly called batch processing.

[0007]   Conventional batch schedulers typically provide a graphical user interface that may represent a single point of control for defining and monitoring background executions in a distributed network of computing devices. For example, an end user (e.g., an individual or enterprise) may use the graphical user interface to submit a request to execute a particular job. Batch schedulers prioritize jobs to be run (i.e., executed) using a data structure known as a job queue. However, when the end user initiates the particular job, there is no indication as to when execution of the job will begin or when the job will be finished.

[0008]   Introduced here, therefore, are computer programs and computer-implemented techniques for predicting when jobs in the queue of a batch scheduler will be completed. More specifically, various embodiments are described herein that relate to mechanisms for predicting the estimated time to completion for jobs that are to be executed by a software asset management platform. The estimated time to completion for a given job can be estimated by summing the expected execution time of the given job and the expected execution times of any jobs to be executed prior to the given job.

[0009]   Implementing the computer programs and computer-implemented techniques described herein can improve the processing capabilities of the computing device(s) that host the software asset management platform and the processing capabilities of the computing device(s) used to interact with the software asset management platform. Implementation of such computer programs and computer-implemented techniques also improves the end user experience when interacting with the software asset management platform. For example, rather than simply wait for an unknown amount of time until execution of a requested job has finished, the software asset management platform can provide an indication as to how long the end user must wait.

## BRIEF DESCRIPTION OF THE DRAWINGS

[0010]   One or more embodiments of the present invention are illustrated by way of example and not limitation in the figures of the accompanying drawings, in which like references indicate similar elements.

[0011]   FIG. 1 is a system-level diagram of a software licensing and distribution model.

[0012]   FIG. 2 depicts a system-level diagram that illustrates how an end user may interact with a software asset management platform through a graphical user interface according to some embodiments.

[0013]   FIG. 3 is a diagrammatic illustration of a process for calculating the execution times for some or all of the tasks that are executed by a software asset management platform, and then using the execution time(s) to predict the expected time to completion for a given job.

[0014]   FIG. 4 depicts a process for building a log of historical execution times that can be used by a software asset management platform to predict the estimated time to completion for jobs in a job queue.

[0015]   FIG. 5 depicts a process for predicting the estimated time to completion for jobs that reside within a job queue and are to be executed by a software asset management platform.

[0016]   FIG. 6 is a block diagram illustrating an example of a processing system in which at least some of the operations described herein can be implemented.

## DETAILED DESCRIPTION

[0017]   Introduced here are computer programs and computer-implemented techniques for predicting when jobs in the queue of a batch scheduler will be completed. More specifically, various embodiments are described herein that

2

relate to mechanisms for predicting the wait time and/or the estimated time to completion for jobs that are to be executed by a software asset management platform. The estimated time to completion for a given job can be estimated by summing the expected execution time of the given job and the expected execution times of any jobs to be executed prior to the given job, while the wait time for a given job can be estimated by summing the expected execution times of any jobs to be executed prior to the given job.

[0018] Implementing the computer programs and computer-implemented techniques described herein can improve the processing capabilities of the computing device(s) that host the software asset management platform and the processing capabilities of the computing device(s) used to interact with the software asset management platform. Implementation of such computer programs and computer-implemented techniques also improves the end user experience when interacting with the software asset management platform. For example, rather than simply wait for an unknown amount of time until execution of a requested job has finished, the software asset management platform can provide an indication as to how long the end user must wait.

[0019] Certain software asset management platforms (e.g., Flexera FlexNet Manager®) may be mentioned herein for the purposes of illustration. Other network-accessible platforms could also be used to run computer programs that utilize a batch scheduler or host resources required by the batch scheduler. Software asset management platforms may be accessible via one or more of a web browser, mobile application, software program, or over-the-top (OTT) application. Accordingly, a software asset management platform may be accessed using any appropriate network-accessible electronic device, such as a mobile phone, tablet, personal computer, game console (e.g., Sony PlayStation® or Microsoft Xbox®), music player (e.g., Apple iPod Touch®), wearable electronic device (e.g., a watch or fitness band), network-connected ("smart") device (e.g., television), virtual/augmented reality system (e.g., Oculus Rift® or Microsoft Hololens®), or some other electronic device.

General System Overview

[0020] FIG. 1 is a system-level diagram of a software licensing and distribution model 100. The hallmark of software licenses is that a licensor (e.g., the software developer, publisher, or distributor) grants a licensee (e.g., an individual or enterprise) the right to use copies of software. The license agreement includes terms that define the authorized uses of the software, such as the number of installations allowed and the terms of distribution.

[0021] Software is often licensed to the licensee based on one or more criteria, such as the total number of computing devices/installations, the total (expected) usage, the desired feature(s), etc. For example, Software as a Service (SaaS) applications are often licensed to licensees on a subscription basis. When the licensee is an enterprise, the licenses are typically made accessible to one or more computing devices 108a-c associated with the enterprise. The computing device(s) 108a-c could include, for example, mobile phones, tablets, or personal computers (e.g., laptop computers or desktop computers) that are owned or operated by employees of the enterprise.

[0022] As shown in FIG. 1, the licenses may be locally hosted on a license server 106 residing within an enterprise network 110. Additionally or alternatively, the licenses may

be remotely hosted by the licensor (or some other third party) on a remote license server accessible to the computing devices 108a-c across an external network 104, such as the Internet, a local area network (LAN), a wide area network (WAN), a point-to-point dial-up connection, cellular network, etc. When software is needed by a computing device (e.g., computing device 108a), the computing device can submit a request to the license server 106 for a license for the software.

[0023] However, software applications (and thus licenses) can be difficult to strategically manage, particularly as the software licensing and distribution model becomes more complex. For example, the software applications required by each computing device may vary based on the corresponding end user's role within the enterprise.

[0024] Accordingly, in some embodiments a software asset management platform 102 is responsible for managing software from various vendors and desktop, laptop, and server hardware. The software asset management platform 102 may also provide automated, entitlement-based license optimization for software applications from vendors such as Adobe®, Microsoft®, and Symantec®. Thus, the software asset management platform 102 can be used by an end user (e.g., an individual or an enterprise) to reduce ongoing software licensing costs, maintain continuous license compliance, and gain better control over software spend while reducing the time and effort required to manage software assets. The software asset management platform 102 (which resides on a computing device such as a server) may be communicatively coupled to the license server 106 and/or the computing devices 108a-c.

[0025] FIG. 2 depicts a system-level diagram that illustrates how an end user may interact with a software asset management platform 202 through a graphical user interface 216 according to some embodiments. As noted above, the software asset management platforms described herein allow for more precise management of software assets and hardware assets owned or leased by the end user.

[0026] Typically, the end user interacts with the software asset management platform 202 by logging in through the graphical user interface 216, which may be accessible via a web browser, mobile application, desktop software, or over-the-top (OTT) application. In short, the end user is able to remotely access and utilize the software asset management platform 202 by forming a connection between the graphical user interface 216 and the computing device on which the software asset management platform 202 resides (e.g., a server) over a network 214a.

[0027] The end user can access the graphical user interface 216 and request that the software asset management platform 202 perform a job (also referred to as a "task"). In some embodiments, the graphical user interface 216 prompts the end user to enter login credentials (e.g., preexisting credentials used to log into other software applications accessible to the end user) that are analyzed and validated by the software asset management platform 202 prior to allowing the end user to submit the request.

[0028] The software asset management platform 202 includes a batch scheduler 204 that may be implemented, for example, as a Microsoft Windows® service. The batch scheduler 204 is responsible for streamlining the execution of jobs by controlling the unattended background execution of the jobs. However, software asset management platforms (and, more specifically, batch schedulers) conventionally do

3

not provide any indication as to when a job will be completed when an end user initiates the job (e.g., by submitting a request through the graphical user interface **216**). This causes some end users to report negative user experiences because they do not know how much longer until a given task will be completed and must simply wait until execution has finished. In fact, such issues may lessen the value provided to the end user by the software asset management platform because strategic decisions may be dependent on knowing when certain job(s) will finish. For example, an end user planning to generate a report on a compliant license position may need to wait until a license reconciliation job has been completed by the software asset management platform. Such issues are more evident in cloud-based environments, where multiple tenants may compete for system resources on a single shared software asset management platform.

[0029] Accordingly, computer programs and computer-implemented techniques are introduced here for predicting the time to completion for jobs that are to be executed by the software asset management platform **202**. As further described below, a log of historical execution times is initially constructed by recording the execution times for some or all of the jobs that are executed by the software asset management platform **202**. For example, a job execution module **206** may be responsible for executing job and then creating an entry for each executed job within the log of historical execution times.

[0030] Each entry may include a job identifier, an execution time, a job type, one or more operating characteristics (e.g., input/output (I/O) subsystem speed, central processing unit (CPU) speed, network connectivity status, network connection bandwidth, etc.), or some combination thereof. Accordingly, the log of historical execution times is embodied in an appropriate data structure, such as a table. The log of historical execution times can then be stored within a data store that is accessible to the software asset management platform **202**. More specifically, the data store may reside within one or more storage modules **212** of the software asset management platform **202**, a remote storage medium **218** that is accessible to the software asset management platform **202**, or both.

[0031] The batch scheduler **204** prioritizes jobs to be run (i.e., executed) using a data structure known as a job queue. Therefore, when the end user initiates a particular job, the particular job is placed in the job queue maintained by the batch scheduler **204**. Jobs may be automatically initiated by the software asset management platform **202** or manually initiated by the end user. For example, a job may be initiated responsive to a graphical user interface (GUI) module **210** receiving input indicative of a request to initiate the particular job at the graphical user interface **216**.

[0032] After placing the particular job in the job queue of the batch scheduler **204**, the software asset management platform **202** can predict the expected time to completion for the particular job based on the log of historical execution times maintained in the data store. More specifically, a calculator module **208** can apply heuristics and algorithms for predicting the estimated time to completion for pending jobs. For example, the calculator module **208** can initially determine whether any jobs are ahead of the particular job in the job queue of the batch scheduler **204**. If one or more jobs are discovered, the calculator module **208** can identify a job type for each of the one or more jobs, and then compute the expected execution time for each of the one or more jobs. The expected execution time for a given job is generally derived by averaging the historical execution times maintained in the data store that are associated with the same job type as the given job. Such a technique allows two different metrics to be readily produced:

[0033] Expected Wait Time—This metric provides an indication as to when execution of the particular job will begin, and is produced by summing the expected execution times corresponding to each of the one or more jobs ahead of the particular job in the job queue.

[0034] Expected Time to Completion—This metric provides an indication as to when execution of the particular job will be completed, and is produced by summing the expected execution time corresponding to the particular job and the expected execution times corresponding to each of the one or more jobs ahead of the particular job in the job queue.

[0035] The calculator module **208** can then forward one or both of these metrics to the GUI module **210**, which posts the metric(s) for review by the end user responsible for submitting the request to perform the particular job.

[0036] In some embodiments, the graphical user interface **216**, software asset management platform **202**, and remote storage medium **218** communicate with one another over one or more networks **214***a-b*, such as the Internet, a local area network (LAN), a wide area network (WAN), a point-to-point dial-up connection, a cellular network, etc. Although FIG. **2** illustrates a remote storage medium **218** that is distinct from the software asset management platform **202**, some embodiments are entirely self-contained. That is, the data store could be hosted locally on the software asset management platform **202** (e.g., within one or more storage modules **212**).

[0037] Note that the term "module" as used herein refers broadly to software, hardware, and/or firmware components. Modules are typically functional components that generate useful data or some other output using specified input(s). Moreover, a module may or may not be self-contained. For example, a computer program may include one or more modules, or a module can include one or more computer programs.

[0038] FIG. **3** is a diagrammatic illustration of a process for calculating the execution times for some or all of the tasks that are executed by a software asset management platform **302**, and then using the execution time(s) to predict the expected time to completion for a given job. At a high level, the process involves three steps: (1) jobs are executed by the software asset management platform **302**; (2) the execution times for those jobs are recorded by the software asset management platform **302**; and (3) the software asset management platform **302** approximates when a future job will be completed based on the execution times of previously-executed jobs. The accuracy of such an approximation can be improved by learning about the actual execution time every time a job is executed by the software asset management platform **302**, and then using those actual execution times to approximate the expected execution time for jobs in a job queue **310**.

[0039] For example, information may be stored in a batch scheduler **304** based on a previous run (also referred to as a "historic run") of the following tasks:

[0040] A full import run for Tenant A was completed in four (4) hours; and

[0041] A System Center Configuration Manager (SCCM) import run for Tenant A was completed in one (1) hour.

[0042] At a subsequent point in time, a BMC Atrium Discover & Dependency Mapping (ADDM) import (also referred to as a "BMC Discovery import") import may be initiated by an end user and placed in the job queue **310** of the batch scheduler **304** behind a single instance of a full import and a single instance of an SCCM import. Note that the end user may or may not be Tenant A. Assuming that the ADDM import will be executed following the full import and the SCCM import (i.e., jobs are executed by a job execution module **306** in chronological order), then the wait time for the ADDM import is five (5) hours. Meanwhile, the expected time to completion will be five (5) hours plus the expected execution time for the ADDM import. Either or both of these metrics can be shown to the end user on a user interface **314** through which the end user submitted a request for the ADDM import to be executed.

[0043] As noted above, the expected execution time for a given job will generally be derived by averaging the execution times maintained in a data store **312** that are associated with the same job type as the given job. While the example provided above includes a single value for different types of imports, multiple values often exist for a single job type. For example, the data store **312** may include separate entries noting that a first full import run was completed in four (4) hours, a second full import run was completed in five (5) hours, and a third import run was completed in four and a half (4.5) hours. In such a scenario, the expected execution time for a full import run can be calculated as follows:

$$\text{Expected Execution Time} = \frac{4 \text{ Hours} + 4.5 \text{ Hours} + 5 \text{ Hours}}{3 \text{ Instances}} = 4.5 \text{ Hours}$$

[0044] Note, however, that the expected execution time may not always be in the form of the arithmetic mean. Other mathematical and statistical models may be used. For example, an arithmetic mean may be appropriate when all external factors (also referred to as "operating characteristics") remain substantially the same because the execution time is expected to be roughly the same. However, in some embodiments weights are assigned to certain execution times so that corresponding entries in the data store **312** are weighted more heavily than others. For example, the software asset management platform **302** may assign a weight to each of the full import runs listed above based on how many operating characteristics are shared between each of those jobs and the job currently in the job queue **310**.

[0045] One skilled in the art will recognize that such an approach requires that each job in the job queue **310** (or a similar job of the same type) has been previously executed at least one time so that an actual execution time is available to determine the expected execution time. Similar prediction techniques may simply substitute a value for those jobs that have not previously been executed by the job execution module **306** of the software asset management platform **302**. For example, a calculator module **308** may substitute a time corresponding to a similar job or a placeholder value that is specified for each job type or a certain set of operating characteristics.

[0046] Here, for example, the software asset management platform **302** automatically initiates two jobs (i.e., Job **1** and

Job **2**). Each job may be performed on behalf of a single tenant of the software asset management platform **302**, all tenants, or a subset of tenants. After these jobs have been completed, the task execution module **306** records an execution time for each job by creating or updating corresponding entries within a log maintained in the data store **312**. The software asset management platform **302** also initiates another job (i.e., Job **3**) upon receiving input indicative of a request to perform the other job at the user interface **314**. After this job has been completed, the task execution module **306** records an execution time for the job by creating or updating a corresponding entry within the log. The software asset management platform **302** (and, more specifically, the task execution module **306**) may record an execution time for all executed jobs or some subset thereof.

[0047] At some later point in time, the software asset management platform **302** initiates the same set of jobs (i.e., Job **1**, Job **2**, and Job **3**). However, the software asset management platform **302** (and, more specifically, the calculator module **308**) can now predict how long until each job is finished based on the actual execution times previously recorded in the data store **312**.

[0048] For example, assuming the jobs are placed in, and dispatched from, the job queue **310** in the following order: Job **1**-Job **2**-Job **3**, the calculator module **308** can calculate the expected wait time (EWT) for each job using expected execution times (EET) as follows:

EWT(Job 1)=None

EWT(Job 2)=EET (Job 1)

EWT(Job 3)=EET(Job 1)+EET(Job 2)

[0049] The expected wait time for a given job represents a prediction of how long it will take until the given job is started. Moreover, the calculator module **308** can predict the estimated time to completion (ETC) as follows:

ETC(Job 1)=EET(Job 1)

ETC(Job 2)=EET(Job 1)+EET(Job 2)

[0050] ETC(Job 3)=EET(Job 1)+EET(Job 2)+EET(Job 3)

[0051] The expected time to completion for a given job represents a prediction of how long it will take until the given job is completed.

[0052] FIG. **4** depicts a process **400** for building a log of historical execution times that can be used by a software asset management platform to predict the estimated time to completion for jobs in a job queue. Jobs of different types are initially executed by the software asset management platform (step **401**). The jobs may be manually initiated by an end user or automatically initiated by the software asset management platform. Moreover, the jobs may be initiated on behalf of one or more tenants of the software asset management platform. For example, a first job may be executed on behalf of a single tenant, while a second job may be executed on behalf of all tenants.

[0053] An entry can then be created for some or all of the jobs executed by the software asset management platform by recording the execution time (step **402**). Each entry may also specify a job identifier, job type, one or more operating characteristics, or both. Operating characteristics may include, for example, the input/output (I/O) subsystem

5

speed, central processing unit (CPU) speed, network connectivity status, network connection bandwidth, or some combination thereof.

[0054] A log of historical execution times can be built from the entries (step **403**). The log can be constructed in several different ways. For example, in some embodiments the entries corresponding to jobs of similar job types are arranged together, while in other embodiments the entries are arranged in chronological order of execution. The log of historical execution times is then stored in a data store that is accessible to the software asset management platform (step **404**). For example, the data store may be maintained locally on the software asset management platform or remotely on another network-accessible computing device.

[0055] FIG. **5** depicts a process **500** for predicting the estimated time to completion for jobs that reside within a job queue and are to be executed by a software asset management platform. The software asset management platform may initially receive input indicative of a request to initiate a particular job (step **501**). The request may be submitted by an end user through a user interface that is accessible via a web browser, mobile application, desktop software, or over-the-top (OTT) application. In some embodiments, the particular job is instead automatically initiated by the software asset management platform. For example, the software asset management platform may include configuration instructions that specify certain job(s) are to be performed in accordance with a predetermined schedule.

[0056] The software asset management platform then places the particular job in a job queue of a batch scheduler (step **502**). Generally, the batch scheduler dispatches jobs to a job execution module for execution in chronological order. However, in some embodiments the batch scheduler dispatches jobs to the job execution module for execution based on a priority assigned to each job. For example, time-sensitive tasks may be assigned a higher priority than non-time-sensitive tasks, and multi-tenant tasks are assigned a higher priority than single tenant tasks. As another example, the end user may be able to specify a priority ordering among jobs using a user interface.

[0057] The software asset management platform then determines whether any jobs exist in the job queue of the batch scheduler ahead of the particular job. For instance, the software asset management platform may determine that one or more jobs are ahead of the particular job in the job queue (step **503**). In such scenarios, the software asset management platform identifies a corresponding job type for each job of the one or more jobs (step **504**), and then derives an expected execution time for each job of the one or more jobs. The expected execution time for a given job is generally derived by averaging the execution times maintained in a log of historical execution times that are associated with the same job type as the given job. For example, the expected execution time for a full import can be predicted by averaging the actual execution time(s) corresponding to previous full imports executed by the software asset management platform.

[0058] The expected execution time can be determined in several different manners. For example, in some embodiments the calculator module of the software asset management platform may use the arithmetic mean, while in other embodiments different mathematical and/or statistical models are used. For example, the software asset management platform may assign a weight to each actual execution time

used to determine the expected execution time based on how many operating characteristics are shared between each of those jobs and the job currently in the job queue.

[0059] The software asset management platform can then compute the expected time to completion for the particular job (step **505**). The expected time to completion can be predicted by summing the expected execution times corresponding to the one or more jobs and the expected execution time corresponding to the particular job. The software asset management platform then posts the expected time to completion for the particular job to a user interface for review (step **506**). For example, the user interface may be accessible to an end user who was responsible for submitting the request to initiate the particular job.

[0060] In some embodiments the expected time to completion is updated upon completion of each job ahead of the particular job in the job queue of the batch scheduler, while in other embodiments the expected time to completion is continually updated in accordance with a refresh rate specified by the end user or set by the software asset management platform (e.g., every 15 minutes or 30 minutes).

[0061] One skilled in the art will recognize that in some instances the software asset management platform may determine that no jobs are ahead of the particular job in the job queue. In such scenarios, the expected time to completion is simply the expected execution time for the particular job.

[0062] Unless contrary to physical possibility, it is envisioned that the steps described above may be performed in various sequences and combinations. For example, the expected execution time for each job type may be maintained and continually updated by the software asset management platform upon completing a job.

[0063] Additional steps could also be included in some embodiments. For example, in some embodiments the software asset management platform may alternatively or additionally compute an optimal expected time to completion and/or a sub-optimal expected time to completion. The optimal expected time to completion may be computed by multiplying the expected time to completion by a first factor (e.g., 0.7 or 0.8), while the sub-optimal expected time to completion may be computed by multiplying the expected time to completion by a second factor (e.g., 1.2 or 1.3). These values can be determined by comparing the current operating characteristics of the underlying computer system (e.g., I/O subsystem speed, CPU speed, network connectivity status, network connection bandwidth, etc.) with the operating characteristics of previous executions. Moreover, these values (which indicate how quickly or slowly tasks may be executed assuming a change in operating characteristics) may be posted to the user interface for review by the end user.

[0064] As another example, in some embodiments the software asset management platform may apply machine learning technique(s) to the job history of the asset management system to improve the accuracy of predicting the expected time to completion and/or the wait time for a given job. For instance, the software asset management platform may apply a Naïve Bayes Classifier algorithm, a K Means Clustering algorithm, a Support Vector Machine algorithm, linear regression, logic regression, artificial neural networks, etc. The software asset management platform can then adjust a parameter for computing the expected time to completion

and/or the wait time for the given job based on the result(s) of applying the machine learning technique(s).

Processing System

[0065] FIG. 6 is a block diagram illustrating an example of a processing system 600 in which at least some operations described herein can be implemented. For example, the processing system 600 may be responsible for monitoring compliance of a licensee or managing software assets. The computing system may include one or more central processing units ("processors") 602, main memory 606, non-volatile memory 610, network adapter 612 (e.g., network interfaces), display 618, input/output devices 620, control device 622 (e.g., keyboard and pointing devices), drive unit 624 including a storage medium 626, and signal generation device 630 that are communicatively connected to a bus 616. The bus 616 is illustrated as an abstraction that represents any one or more separate physical buses, point to point connections, or both connected by appropriate bridges, adapters, or controllers. The bus 616, therefore, can include, for example, a system bus, a Peripheral Component Interconnect (PCI) bus or PCI-Express bus, a HyperTransport or industry standard architecture (ISA) bus, a small computer system interface (SCSI) bus, a universal serial bus (USB), IIC (I2C) bus, or an Institute of Electrical and Electronics Engineers (IEEE) standard 1394 bus, also called "Firewire."

[0066] In various embodiments, the processing system 600 operates as a standalone device, although the processing system 600 may be connected (e.g., wired or wirelessly) to other machines. For example, the processing system 600 may include a terminal that is coupled directly to licensing server of a licensee. As another example, the processing system 600 may be wirelessly coupled to the licensing server.

[0067] In various embodiments, the processing system 600 may be a server computer, a client computer, a personal computer (PC), a user device, a tablet PC, a laptop computer, a personal digital assistant (PDA), a cellular telephone, an iPhone, an iPad, a Blackberry, a processor, a telephone, a web appliance, a network router, switch or bridge, a console, a hand-held console, a (hand-held) gaming device, a music player, any portable, mobile, hand-held device, or any machine capable of executing a set of instructions (sequential or otherwise) that specify actions to be taken by the computing system.

[0068] While the main memory 606, non-volatile memory 610, and storage medium 626 (also called a "machine-readable medium) are shown to be a single medium, the term "machine-readable medium" and "storage medium" should be taken to include a single medium or multiple media (e.g., a centralized or distributed database, and/or associated caches and servers) that store one or more sets of instructions 628. The term "machine-readable medium" and "storage medium" shall also be taken to include any medium that is capable of storing, encoding, or carrying a set of instructions for execution by the computing system and that cause the computing system to perform any one or more of the methodologies of the presently disclosed embodiments.

[0069] In general, the routines executed to implement the embodiments of the disclosure, may be implemented as part of an operating system or a specific application, component, program, object, module, or sequence of instructions referred to as "computer programs." The computer programs typically comprise one or more instructions (e.g., instructions 604, 608, 628) set at various times in various memory and storage devices in a computer, and that, when read and executed by one or more processing units or processors 602, cause the processing system 600 to perform operations to execute elements involving the various aspects of the disclosure.

[0070] Moreover, while embodiments have been described in the context of fully functioning computers and computer systems, those skilled in the art will appreciate that the various embodiments are capable of being distributed as a program product in a variety of forms, and that the disclosure applies equally regardless of the particular type of machine or computer-readable media used to actually effect the distribution.

[0071] Further examples of machine-readable storage media, machine-readable media, or computer-readable (storage) media include recordable type media such as volatile and non-volatile memory devices 610, floppy and other removable disks, hard disk drives, optical disks (e.g., Compact Disk Read-Only Memory (CD ROMS), Digital Versatile Disks (DVDs)), and transmission type media such as digital and analog communication links.

[0072] The network adapter 612 enables the processing system 600 to mediate data in a network 614 with an entity that is external to the processing system 600 through any known and/or convenient communications protocol supported by the processing system 600 and the external entity. The network adapter 612 can include one or more of a network adaptor card, a wireless network interface card, a router, an access point, a wireless router, a switch, a multi-layer switch, a protocol converter, a gateway, a bridge, bridge router, a hub, a digital media receiver, and/or a repeater.

[0073] The network adapter 612 can include a firewall which can, in some embodiments, govern and/or manage permission to access/proxy data in a computer network, and track varying levels of trust between different machines and/or applications. The firewall can be any number of modules having any combination of hardware and/or software components able to enforce a predetermined set of access rights between a particular set of machines and applications, machines and machines, and/or applications and applications, for example, to regulate the flow of traffic and resource sharing between these varying entities. The firewall may additionally manage and/or have access to an access control list which details permissions including for example, the access and operation rights of an object by an individual, a machine, and/or an application, and the circumstances under which the permission rights stand.

[0074] As indicated above, the techniques introduced here implemented by, for example, programmable circuitry (e.g., one or more microprocessors), programmed with software and/or firmware, entirely in special-purpose hardwired (i.e., non-programmable) circuitry, or in a combination or such forms. Special-purpose circuitry can be in the form of, for example, one or more application-specific integrated circuits (ASICs), programmable logic devices (PLDs), field-programmable gate arrays (FPGAs), etc.

[0075] Note that any of the embodiments described above can be combined with another embodiment, except to the extent that it may be stated otherwise above or to the extent that any such embodiments might be mutually exclusive in function and/or structure.

[0076] Although the present invention has been described with reference to specific exemplary embodiments, it will be recognized that the invention is not limited to the embodiments described, but can be practiced with modification and alteration within the spirit and scope of the appended claims. Accordingly, the specification and drawings are to be regarded in an illustrative sense rather than a restrictive sense.

1. A computer-implemented method comprising:

building a log of historical execution times by recording an execution time for each job that is executed by a software asset management platform;

storing the log of historical execution times in a data store that is accessible to the software asset management platform;

receiving input at a user interface indicative of a request to initiate a particular job;

placing the particular job in a queue of a batch scheduler computer application that is executed by the software asset management platform;

identifying multiple execution times in the log of historical execution times corresponding to multiple past jobs that are of a same type as the particular job;

assigning a weight to each of the multiple execution times based on how many operating characteristics are shared between each of the multiple past jobs and the particular job;

predicting an expected execution time for the particular job based on the multiple weighted execution times; and

predicting an expected time to completion for the particular job based on the log expected execution time.

2. The computer-implemented method of claim 1, wherein building the log of historical execution times comprises:

executing jobs of different types; and

creating an entry for each job in the data store that includes an execution time, a job type, and one or more operating characteristics.

3. The computer-implemented method of claim 2, wherein predicting the expected time to completion for the particular job comprises:

determining that one or more jobs are ahead of the particular job in the queue of the batch scheduler computer application;

identifying a corresponding job type for each job of the one or more jobs; and

computing the expected time to completion for the particular job by summing expected execution times corresponding to the one or more jobs and the expected execution time corresponding to the particular job,

wherein expected execution time for a given job is derived by averaging execution times maintained in the log of historical execution times that are associated with a same job type as the given job.

4. The computer-implemented method of claim 3, further comprising:

posting the expected time to completion for the particular job to the user interface for review by an end user responsible for submitting the request.

5. The computer-implemented method of claim 4, wherein the posted expected time to completion for the particular job is updated upon completion of each job ahead of the particular job in the queue of the batch scheduler computer application.

6. The computer-implemented method of claim 1, wherein the user interface is accessible via a web browser, mobile application, software program, or over-the-top (OTT) application.

7. The computer-implemented method of claim 1, wherein jobs executed by the software asset management platform are manually initiated by an end user or automatically initiated by the software asset management platform.

8. A computer-implemented method comprising:

receiving input at a user interface indicative of a request to a software asset management platform to initiate a particular job;

placing the particular job in a queue of a batch scheduler computer application that is executed by the software asset management platform;

determining that one or more jobs are ahead of the particular job in the queue of the batch scheduler computer application;

identifying a corresponding job type for each job of the one or more jobs;

computing an expected time to completion for the particular job by summing expected execution times corresponding to the one or more jobs and an expected execution time corresponding to the particular job,

wherein expected execution time for a given job is derived by averaging execution times maintained in a log of historical execution times that are associated with a same job type as the given job,

wherein the log of historical execution times includes an entry for each job executed by the software asset management platform, each entry including an execution time, a job type, and one or more operating characteristics,

wherein the expected execution times corresponding to the one or more jobs and the expected execution time corresponding to the particular job are weighted based on how many operating characteristics are shared between the particular job and each of the one or more jobs, and

wherein those expected execution times corresponding to jobs that share at least one operating parameter in common with the particular job are weighted more heavily; and

posting the expected time to completion for the particular job to the user interface for review by an end user responsible for submitting the request.

9. The computer-implemented method of claim 8, wherein the log of historical execution times is maintained in a data store that is accessible to the software asset management platform.

10. The computer-implemented method of claim 9, wherein the operating characteristics include input/output (I/O) subsystem speed, central processing unit (CPU) speed, network connectivity status, network connection bandwidth, or some combination thereof.

11. The computer-implemented method of claim 8, further comprising:

updating the posted expected time to completion for the particular job upon completion of each job ahead of the particular job in the queue of the batch scheduler computer application.

**12**. The computer-implemented method of claim **8**, further comprising:

continually updating the posted expected time to completion for the particular job in accordance with a refresh rate specified by the end user or the software asset management platform.

**13**. An asset management system comprising:

a processor operable to execute instructions stored in a memory; and

the memory that includes specific instructions for predicting estimated times to completion for jobs that are to be executed by the asset management platform, wherein execution of the specific instructions causes the processor to:

receive input indicative of a request to initiate a particular job that is submitted at a user interface;

place the particular job in a queue of a batch scheduler computer application;

determine that one or more jobs are ahead of the particular job in the queue of the batch scheduler computer application;

identify a corresponding job type for each job of the one or more jobs;

assign a weight to expected execution times corresponding to the one or more jobs and an expected execution time corresponding to the particular job based on how many operating characteristics are shared between the particular job and each of the one or more lobs;

compute an expected time to completion for the particular job based on the expected execution times corresponding to the one or more jobs and the expected execution time corresponding to the particular job; and

cause the expected time to completion for the particular job to be posted to the user interface for review by an end user responsible for submitting the request.

**14**. The asset management system of claim **13**, wherein computing the expected time to completion for the particular job comprises:

summing the expected execution times corresponding to the one or more jobs and the expected execution time corresponding to the particular job.

**15**. The asset management system of claim **14**, wherein expected execution time for a given job is derived by averaging execution times maintained in a log of historical execution times that are associated with a same job type as the given job.

**16**. (canceled)

**17**. The asset management system of claim **13**, wherein the one or more jobs ahead of the particular job in the queue are dispatched for execution based on a priority assigned to each job.

**18**. The asset management system of claim **13**, wherein execution of the specific instructions causes the processor to:

compute an optimal expected time to completion for the particular job by multiplying the expected time to completion by a first factor;

compute a sub-optimal expected time to completion for the particular job by multiplying the expected time to completion by a second factor; and

cause the optimal expected time to completion, the sub-optimal expected time to completion, or both to be posted to the user interface for review by the end user.

**19**. The asset management system of claim **13**, wherein execution of the specific instructions causes the processor to:

apply one or more machine learning techniques to a job history of the asset management system; and

adjusting a parameter for computing the expected time to completion for the particular job based on a result of applying the one or more machine learning techniques, wherein said adjusting improves causes accuracy of said computing to be improved.

* * * * *