US 20190342380A1

(19) United States

(12) Patent Application Publication (10) Pub. No.: US 2019/0342380 A1

THOTA et al. (43) Pub. Date: Nov. 7, 2019

(54) **ADAPTIVE RESOURCE-GOVERNED SERVICES FOR PERFORMANCE-COMPLIANT DISTRIBUTED WORKLOADS**

(71) Applicant: **Microsoft Technology Licensing, LLC**, Redmond, WA (US)

(72) Inventors: **Shireesh Kumar THOTA**, Redmond, WA (US); **Momin Mahmoud AL-GHOSIEN**, Sammamish, WA (US); **Rajeev Sudhakar BHOPI**, Mercer Island, WA (US); **Samer BOSHRA**, Woodinville, WA (US); **Madhan GAJENDRAN**, Bengaluru (IN); **Atul KATIYAR**, Sammamish, WA (US); **Abhijit Padmanabh PAI**, Bangalore (IN); **Karthik RAMAN**, Sammanish, WA (US); **Ankur Savailal SHAH**, Redmond, WA (US); **Pankaj SHARMA**, Kirkland, WA (US); **Dharma SHUKLA**, Bellevue, WA (US); **Shreshth SINGHAL**, Seattle, WA (US); **Hari Sudan SUNDAR**, Redmond, WA (US); **Lalitha Manjapara VISWANATHAN**, Redmond, WA (US)

(21) Appl. No.: **15/991,953**

(22) Filed: **May 29, 2018**

Related U.S. Application Data

(60) Provisional application No. 62/668,226, filed on May 7, 2018.

Publication Classification

(51) **Int. Cl.**

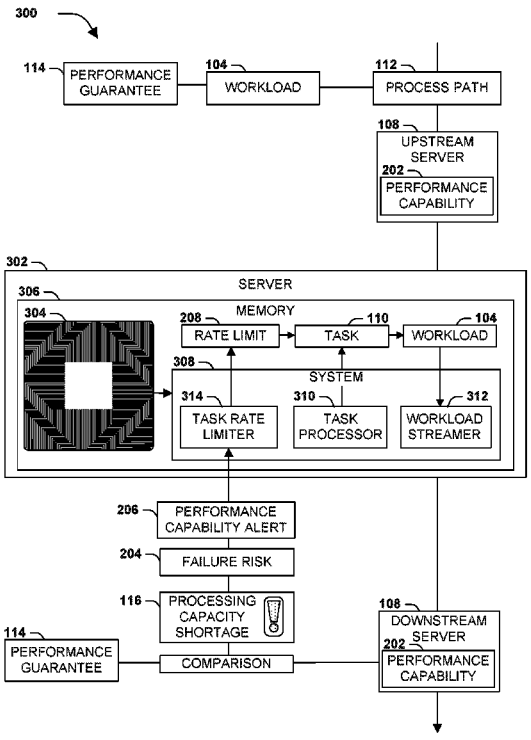
H04L 29/08	(2006.01)
H04L 12/24	(2006.01)
H04L 29/06	(2006.01)

(52) **U.S. Cl.**

CPC ***H04L 67/1012*** (2013.01); ***H04L 67/1008*** (2013.01); ***H04L 67/1029*** (2013.01); ***H04L 69/24*** (2013.01); ***H04L 41/5009*** (2013.01); ***H04L 41/5022*** (2013.01); ***H04L 67/1034*** (2013.01)

(57) **ABSTRACT**

Processing services are often provisioned by defining and adjusting the performance capabilities of individual servers, and in multitenancy scenarios, servers may allocate computational resources to ensure that a first client workload does not impact a second client workload. However, a reduced performance capability of a server may create a processing jam with respect to an upstream server of the process path of the workload, where the processing rate mismatch creates a risk of failing to fulfill the performance guarantee for the workload. Instead, the downstream server may monitor and compare its performance capability with the performance guarantee. If a performance guarantee failure risk arises, the server may transmit a performance capability alert to the upstream server, which may rate-limit the processing of the workload. Rate-limiting by the first server in the server path may limit workload intake to a volume for which the process path can fulfill the performance guarantee.



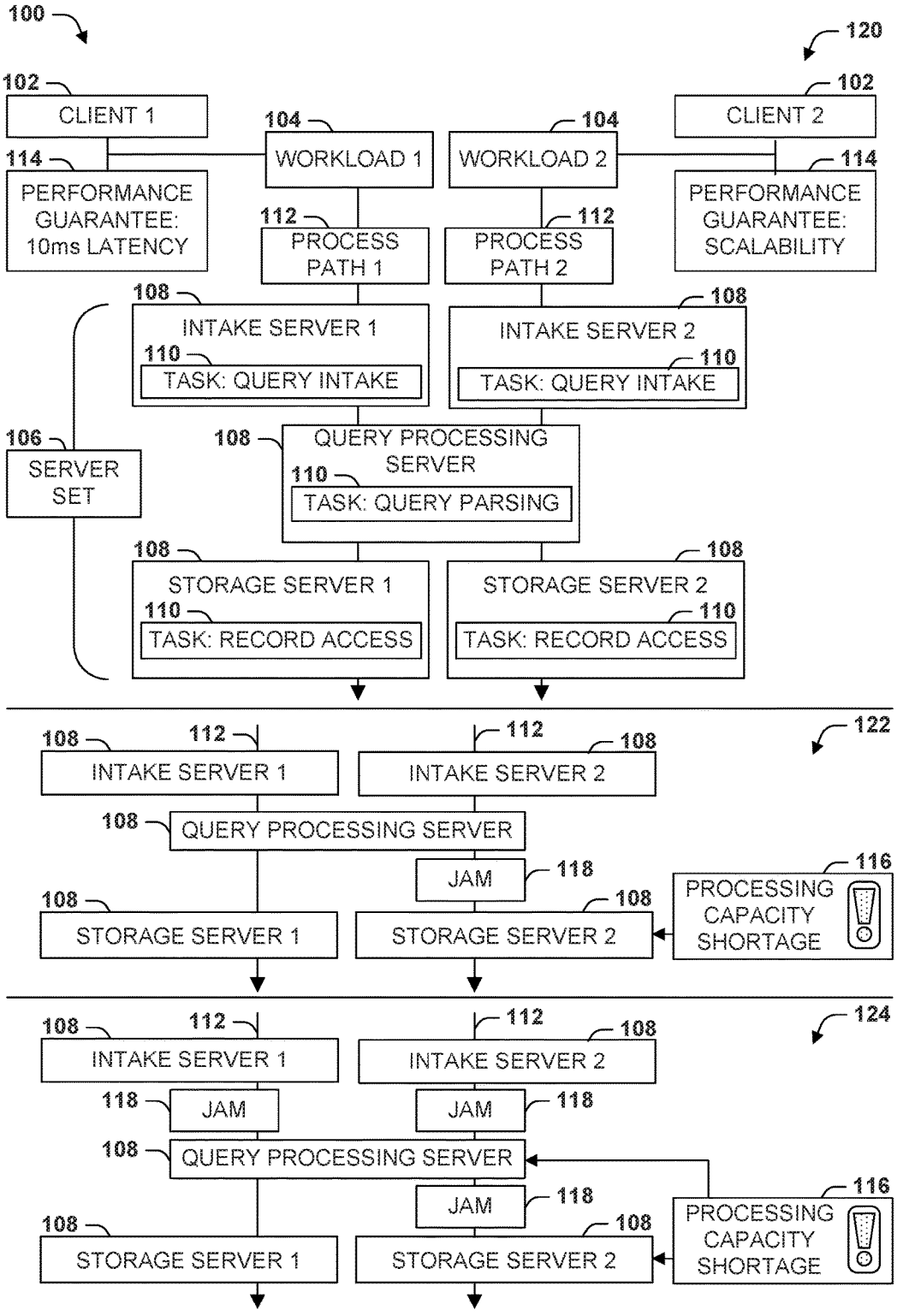

100
120
102
CLIENT 1
104
WORKLOAD 1
114
PERFORMANCE GUARANTEE: 10ms LATENCY
112
PROCESS PATH 1
108
INTAKE SERVER 1
110
TASK: QUERY INTAKE
106
SERVER SET
108
QUERY PROCESSING SERVER
110
TASK: QUERY PARSING
108
STORAGE SERVER 1
110
TASK: RECORD ACCESS
102
CLIENT 2
104
WORKLOAD 2
114
PERFORMANCE GUARANTEE: SCALABILITY
112
PROCESS PATH 2
108
INTAKE SERVER 2
110
TASK: QUERY INTAKE
108
STORAGE SERVER 2
110
TASK: RECORD ACCESS
122
108
INTAKE SERVER 1
112
INTAKE SERVER 2
108
QUERY PROCESSING SERVER
JAM
118
116
PROCESSING CAPACITY SHORTAGE
108
STORAGE SERVER 1
108
STORAGE SERVER 2
124
108
INTAKE SERVER 1
112
INTAKE SERVER 2
108
118
JAM
JAM
118
108
QUERY PROCESSING SERVER
JAM
118
116
PROCESSING CAPACITY SHORTAGE
108
STORAGE SERVER 1
108
STORAGE SERVER 2

FIG. 1

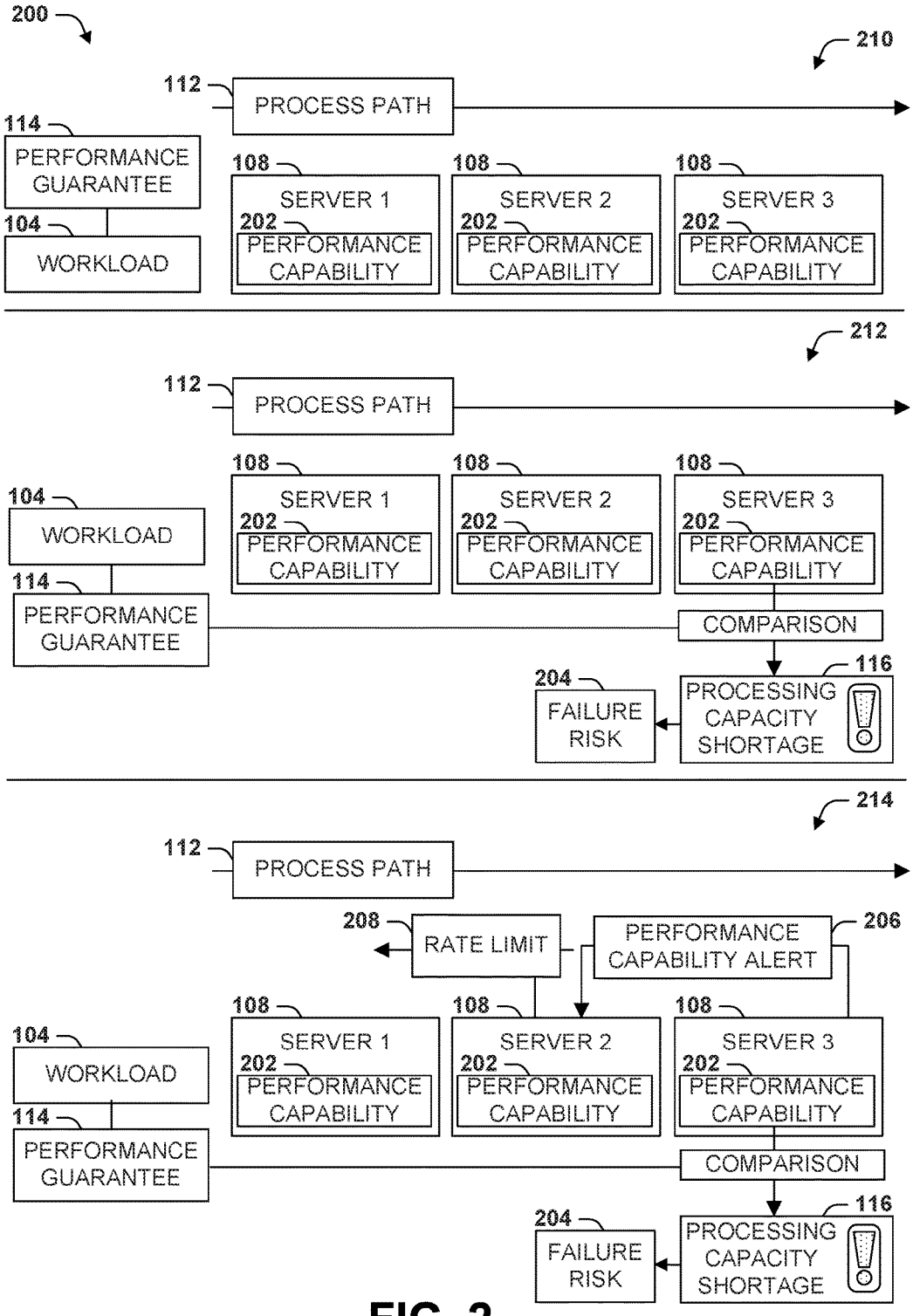

200
210
112
PROCESS PATH
114
PERFORMANCE GUARANTEE
104
WORKLOAD
108
SERVER 1
202
PERFORMANCE CAPABILITY
108
SERVER 2
202
PERFORMANCE CAPABILITY
108
SERVER 3
202
PERFORMANCE CAPABILITY
212
112
PROCESS PATH
108
SERVER 1
202
PERFORMANCE CAPABILITY
108
SERVER 2
202
PERFORMANCE CAPABILITY
108
SERVER 3
202
PERFORMANCE CAPABILITY
104
WORKLOAD
114
PERFORMANCE GUARANTEE
COMPARISON
116
PROCESSING CAPACITY SHORTAGE
204
FAILURE RISK
214
112
PROCESS PATH
208
RATE LIMIT
PERFORMANCE CAPABILITY ALERT
206
108
SERVER 1
202
PERFORMANCE CAPABILITY
108
SERVER 2
202
PERFORMANCE CAPABILITY
108
SERVER 3
202
PERFORMANCE CAPABILITY
104
WORKLOAD
114
PERFORMANCE GUARANTEE
COMPARISON
116
PROCESSING CAPACITY SHORTAGE
204
FAILURE RISK

FIG. 2

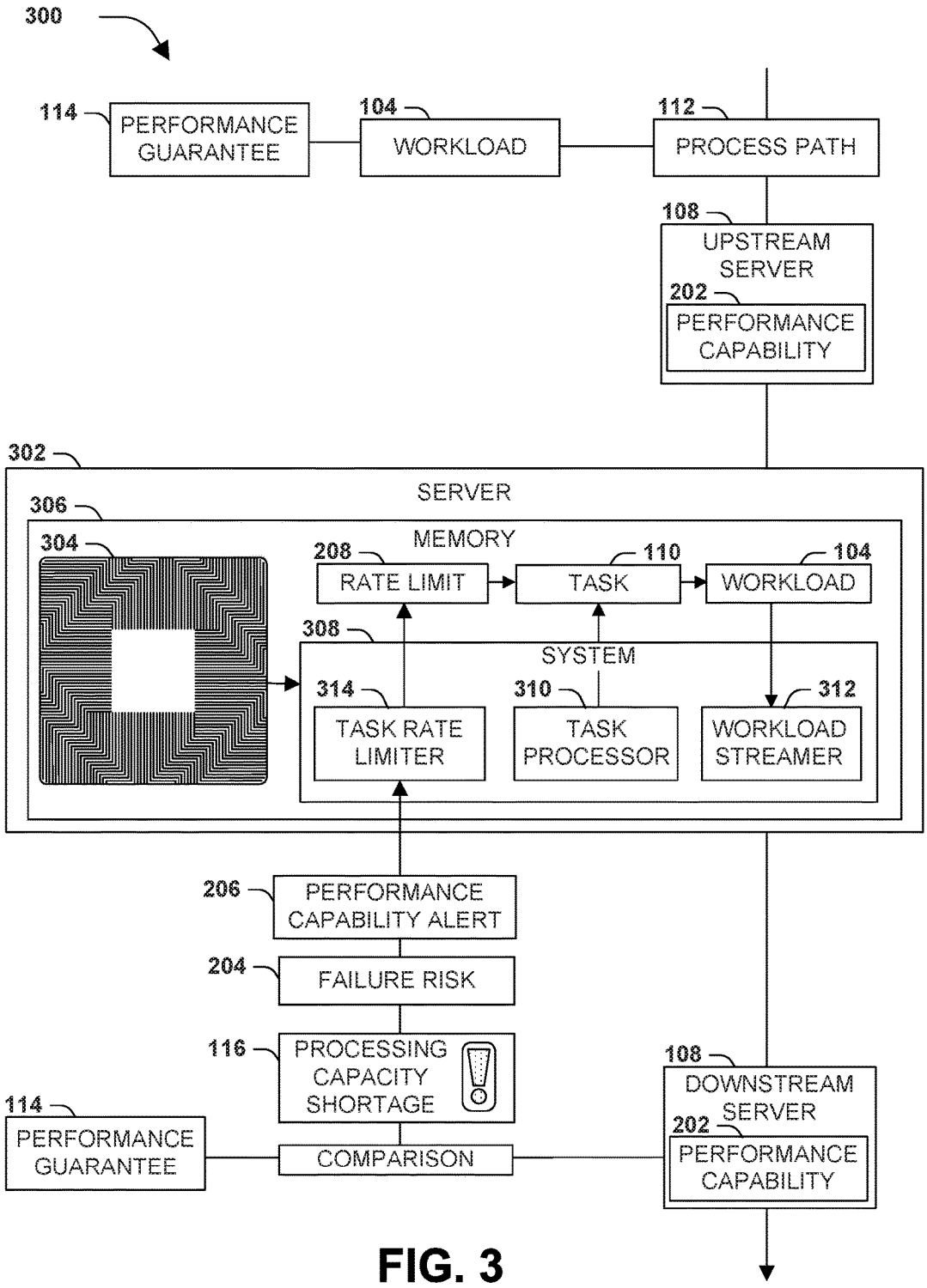

300
114
PERFORMANCE GUARANTEE
104
WORKLOAD
112
PROCESS PATH
108
UPSTREAM SERVER
202
PERFORMANCE CAPABILITY
302
SERVER
306
304
MEMORY
208
RATE LIMIT
110
TASK
104
WORKLOAD
308
SYSTEM
314
TASK RATE LIMITER
310
TASK PROCESSOR
312
WORKLOAD STREAMER
206
PERFORMANCE CAPABILITY ALERT
204
FAILURE RISK
116
PROCESSING CAPACITY SHORTAGE
108
DOWNSTREAM SERVER
202
PERFORMANCE CAPABILITY
114
PERFORMANCE GUARANTEE
COMPARISON

FIG. 3

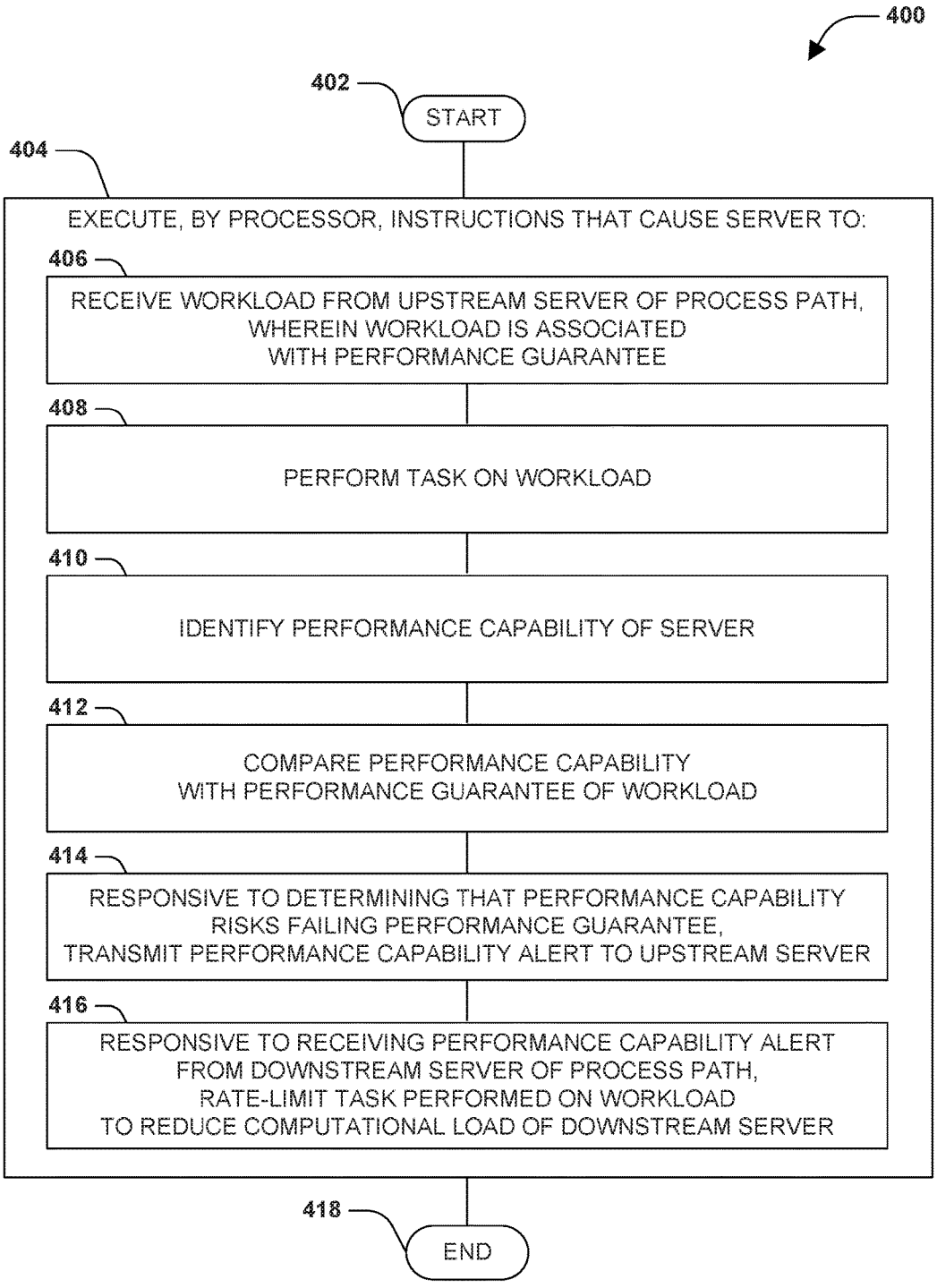

400
402
START
404
EXECUTE, BY PROCESSOR, INSTRUCTIONS THAT CAUSE SERVER TO:
406
RECEIVE WORKLOAD FROM UPSTREAM SERVER OF PROCESS PATH, WHEREIN WORKLOAD IS ASSOCIATED WITH PERFORMANCE GUARANTEE
408
PERFORM TASK ON WORKLOAD
410
IDENTIFY PERFORMANCE CAPABILITY OF SERVER
412
COMPARE PERFORMANCE CAPABILITY WITH PERFORMANCE GUARANTEE OF WORKLOAD
414
RESPONSIVE TO DETERMINING THAT PERFORMANCE CAPABILITY RISKS FAILING PERFORMANCE GUARANTEE, TRANSMIT PERFORMANCE CAPABILITY ALERT TO UPSTREAM SERVER
416
RESPONSIVE TO RECEIVING PERFORMANCE CAPABILITY ALERT FROM DOWNSTREAM SERVER OF PROCESS PATH, RATE-LIMIT TASK PERFORMED ON WORKLOAD TO REDUCE COMPUTATIONAL LOAD OF DOWNSTREAM SERVER
418
END

FIG. 4

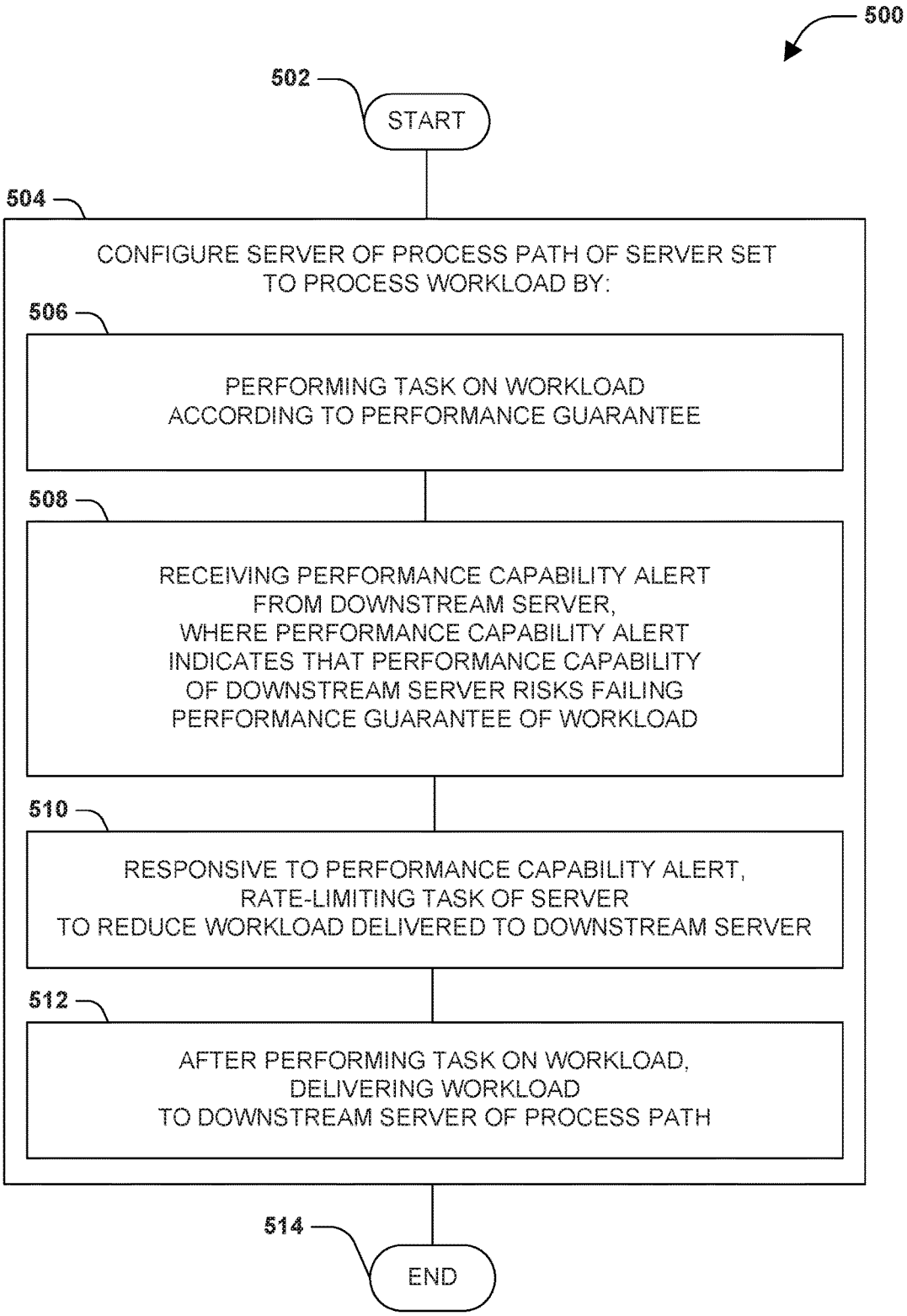

500
502
START
504
CONFIGURE SERVER OF PROCESS PATH OF SERVER SET TO PROCESS WORKLOAD BY:
506
PERFORMING TASK ON WORKLOAD ACCORDING TO PERFORMANCE GUARANTEE
508
RECEIVING PERFORMANCE CAPABILITY ALERT FROM DOWNSTREAM SERVER, WHERE PERFORMANCE CAPABILITY ALERT INDICATES THAT PERFORMANCE CAPABILITY OF DOWNSTREAM SERVER RISKS FAILING PERFORMANCE GUARANTEE OF WORKLOAD
510
RESPONSIVE TO PERFORMANCE CAPABILITY ALERT, RATE-LIMITING TASK OF SERVER TO REDUCE WORKLOAD DELIVERED TO DOWNSTREAM SERVER
512
AFTER PERFORMING TASK ON WORKLOAD, DELIVERING WORKLOAD TO DOWNSTREAM SERVER OF PROCESS PATH
514
END

FIG. 5

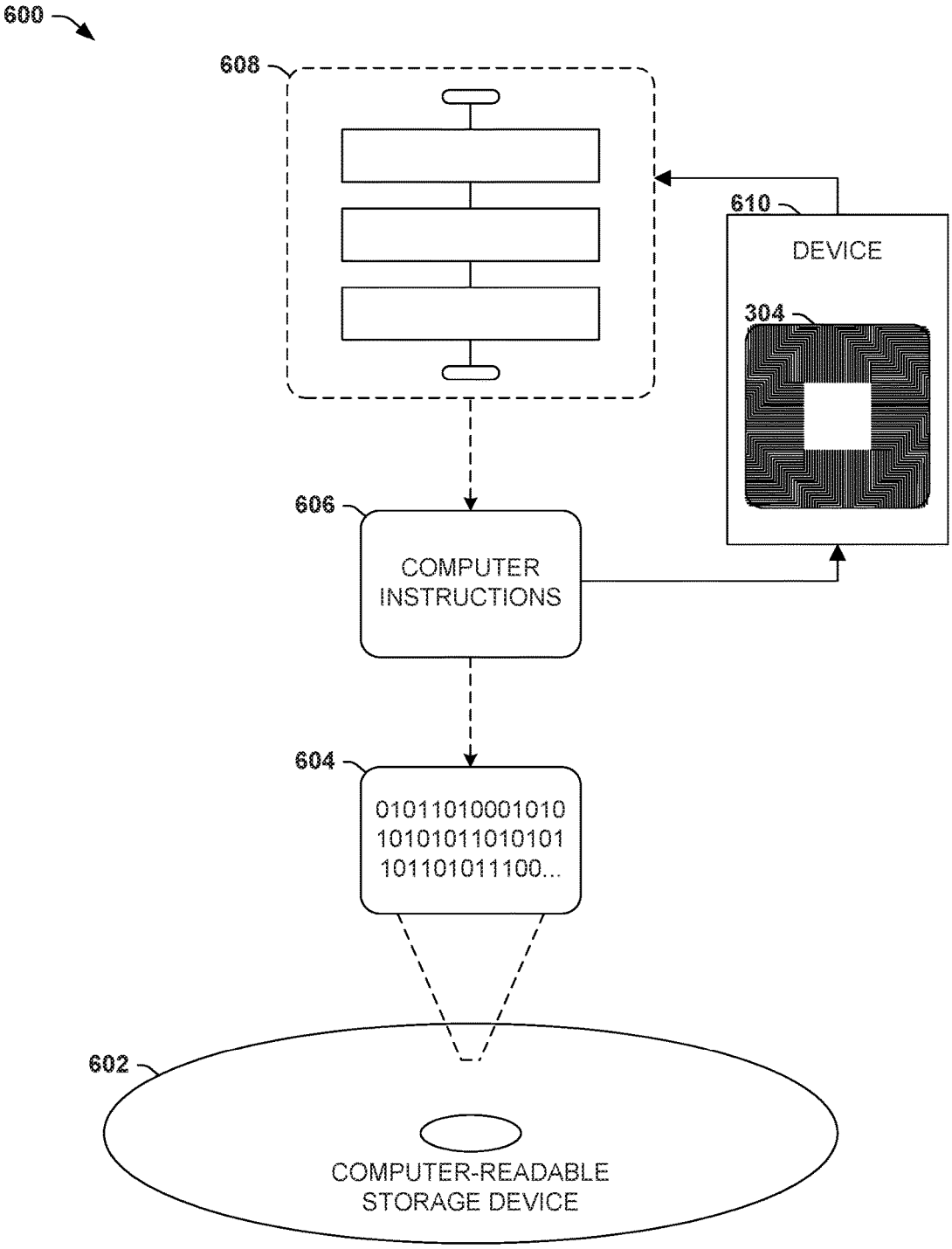

600
608
610
DEVICE
304
606
COMPUTER INSTRUCTIONS
604
01011010001010
10101011010101
101101011100...
602
COMPUTER-READABLE STORAGE DEVICE

FIG. 6

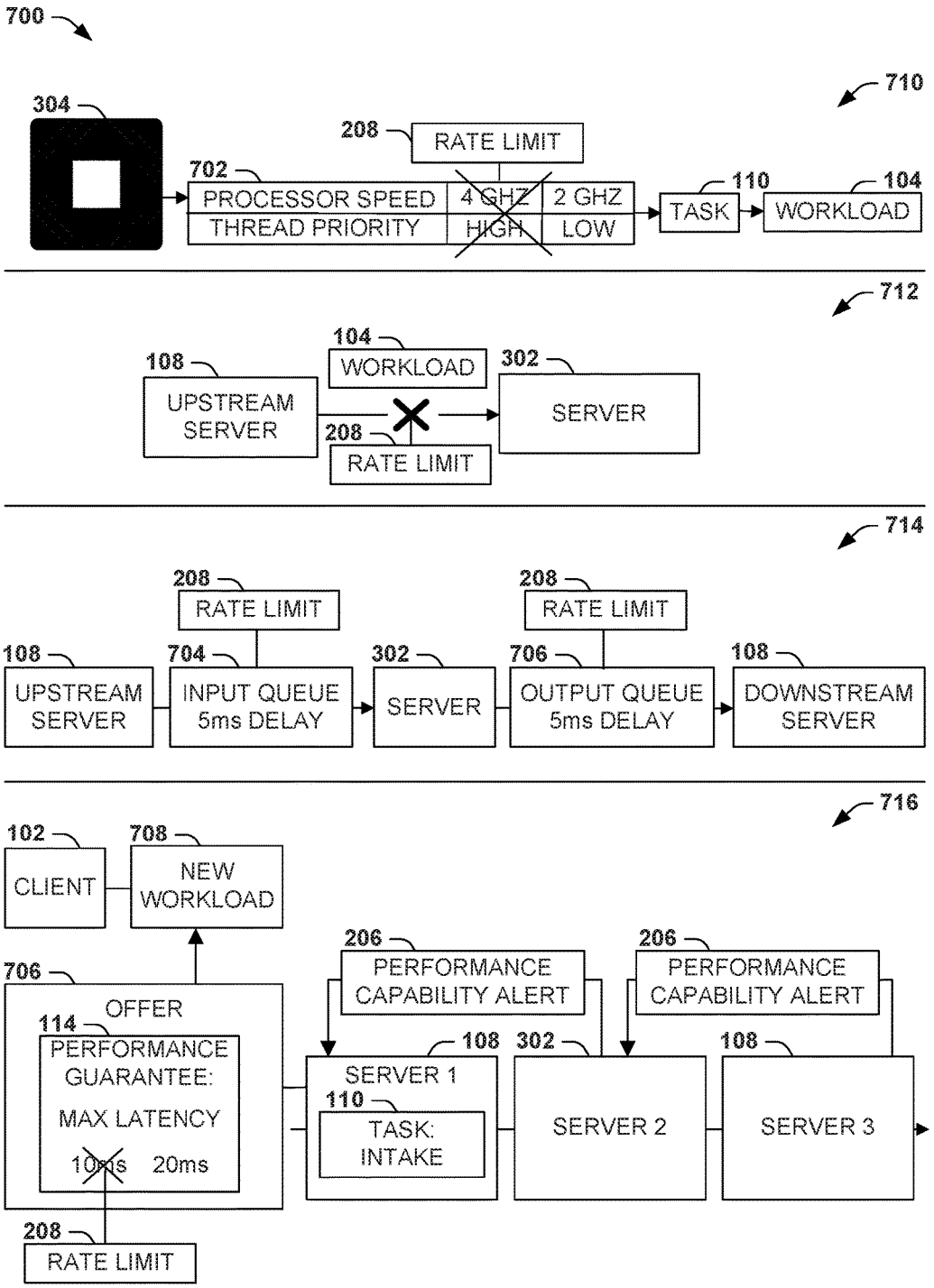

700
710
304
208
RATE LIMIT
702
PROCESSOR SPEED
4 GHZ
2 GHZ
THREAD PRIORITY
HIGH
LOW
110
TASK
104
WORKLOAD
712
104
WORKLOAD
108
UPSTREAM SERVER
302
SERVER
208
RATE LIMIT
714
208
RATE LIMIT
208
RATE LIMIT
108
UPSTREAM SERVER
704
INPUT QUEUE 5ms DELAY
302
SERVER
706
OUTPUT QUEUE 5ms DELAY
108
DOWNSTREAM SERVER
716
102
CLIENT
708
NEW WORKLOAD
206
PERFORMANCE CAPABILITY ALERT
206
PERFORMANCE CAPABILITY ALERT
706
OFFER
114
PERFORMANCE GUARANTEE:
MAX LATENCY
10ms
20ms
108
SERVER 1
110
TASK: INTAKE
302
SERVER 2
108
SERVER 3
208
RATE LIMIT

FIG. 7

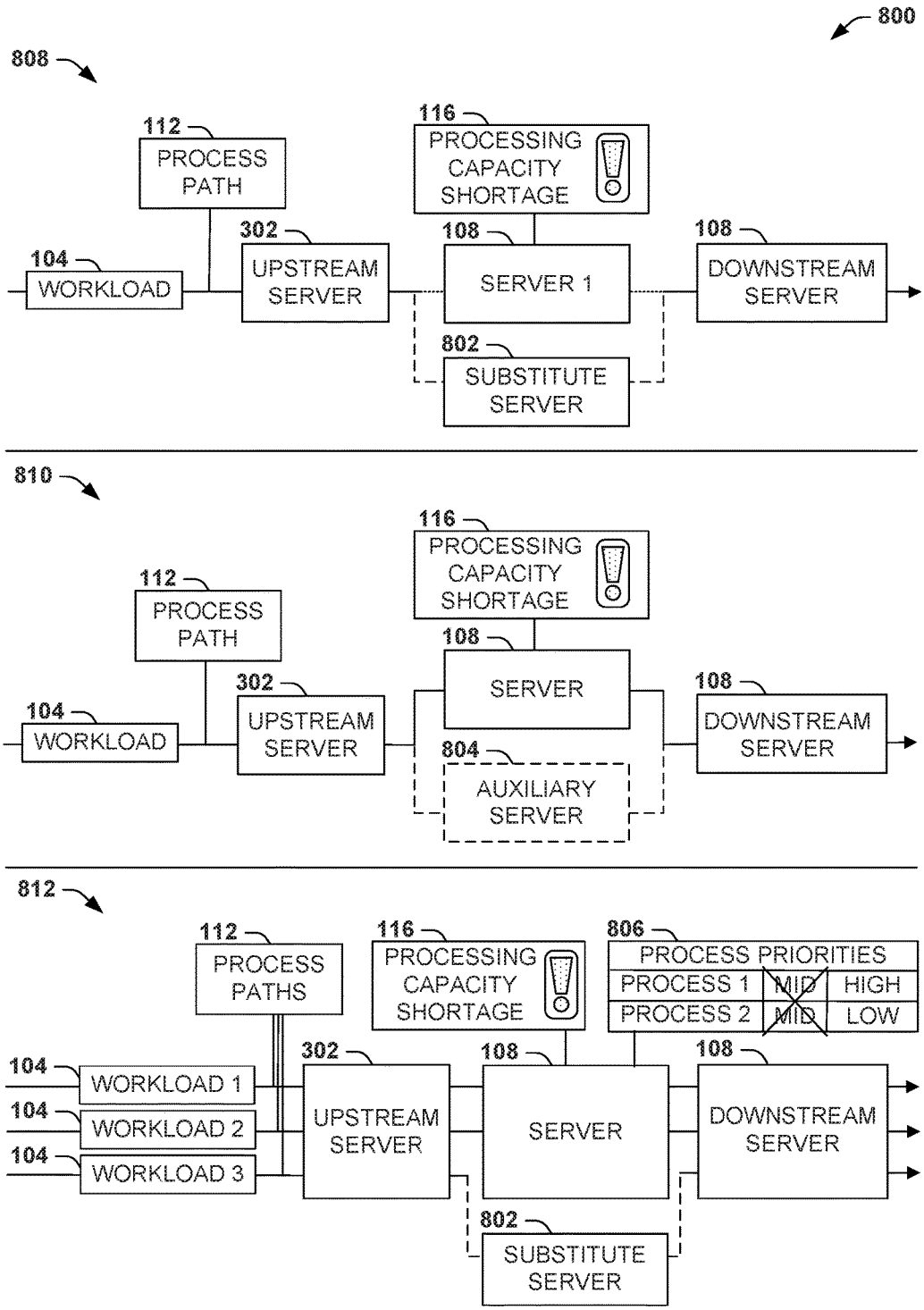

800
808
112
PROCESS PATH
116
PROCESSING CAPACITY SHORTAGE
302
UPSTREAM SERVER
108
SERVER 1
108
DOWNSTREAM SERVER
104
WORKLOAD
802
SUBSTITUTE SERVER
810
116
PROCESSING CAPACITY SHORTAGE
112
PROCESS PATH
108
SERVER
302
UPSTREAM SERVER
108
DOWNSTREAM SERVER
104
WORKLOAD
804
AUXILIARY SERVER
812
112
PROCESS PATHS
116
PROCESSING CAPACITY SHORTAGE
806
PROCESS PRIORITIES
PROCESS 1 MID HIGH
PROCESS 2 MID LOW
302
108
108
104
WORKLOAD 1
104
WORKLOAD 2
104
WORKLOAD 3
UPSTREAM SERVER
SERVER
DOWNSTREAM SERVER
802
SUBSTITUTE SERVER

FIG. 8

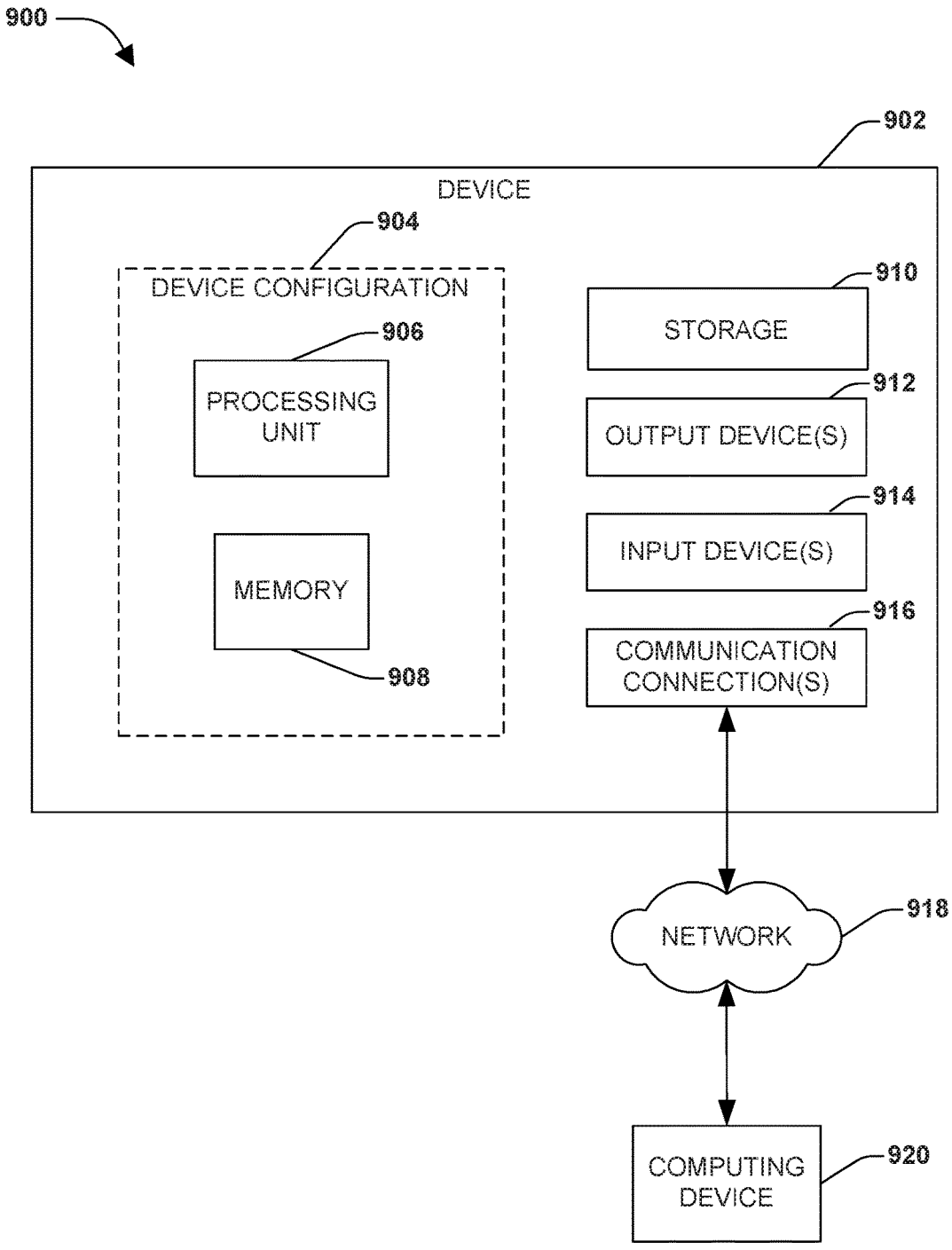

900
902
DEVICE
904
DEVICE CONFIGURATION
906
PROCESSING UNIT
MEMORY
908
910
STORAGE
912
OUTPUT DEVICE(S)
914
INPUT DEVICE(S)
916
COMMUNICATION CONNECTION(S)
NETWORK
918
COMPUTING DEVICE
920

FIG. 9

ADAPTIVE RESOURCE-GOVERNED SERVICES FOR PERFORMANCE-COMPLIANT DISTRIBUTED WORKLOADS

CROSS-REFERENCE TO RELATED APPLICATIONS

[0001] This application is a continuation of, and claims priority under 35 U.S.C. §§ 119-120 to, U.S. Patent Application No. 62/668,226, entitled "DISTRIBUTED DATABASES," filed on May 7, 2018, the entirety of which is hereby incorporated by reference as if fully rewritten herein.

BACKGROUND

[0002] Within the field of computing, many scenarios involve a processing service that performs a set of workloads using a server set. For example, a database service may provide a distributed set of servers with various capabilities, such as a query intake server that receives a query; a query processing server that parses the query; and a storage server that applies the logical operations of the parsed query over a data set.

[0003] A large-scale, distributed server set may involve a significant number of servers that perform a large number of distinct workloads for a variety of applications and/or clients. Moreover, the workloads of various applications and clients may utilize different process paths through the server set. For example, a process path for a first workload may involve a first sequence of tasks to be performed by a corresponding first sequence of servers, such as a first intake server in a first region; a query processing server; and a first storage server that stores records involved in the first workload. A process path for a second workload may involve a different sequence of tasks to be performed by a corresponding second sequence of servers, such as a second intake server in a second region; the same query processing server; a second storage server that stores records involved in the second workload.

[0004] In such scenarios, workloads may be subject to various forms of performance sensitivities. As a first such example, a workload may be sensitive to latency (e.g., a realtime application in which users or devices have to receive a result of the workload within a limited time, and in which delays may be perceptible and/or problematic). As a second such example, a workload may be sensitive to scalability and throughput (e.g., demand for the workload may fluctuate over time, and the inability of the server set to scale up to handle an influx of volume may be problematic). As a third such example, a workload may be sensitive to consistency and/or concurrency issues (e.g., a strictly deterministic workload may have to receive the same result across multiple instances, where inconsistent results may be problematic). As a fourth such example, a workload may be sensitive to replication and/or resiliency (e.g., downtime, data loss, or the failure of the workload may be problematic). In view of such sensitivities, it may be desirable to enable the server set to provide a performance guarantee for a workload, e.g., a guarantee that the server set is capable of handling a surge of volume up to a particular amount while maintaining latency below a particular level.

[0005] In multitenant scenarios, workloads for different applications and/or clients may share a process path. Other workloads may take different process paths through the server set, but may both include one or more servers of the server set that are to be shared by the workloads. Servers may be shared among workloads that are associated with different clients and/or applications; in some scenarios, a server may concurrently handle workloads on behalf of hundreds or even thousands of different applications or clients.

[0006] A variety of multi-tenancy techniques may be utilized to ensure that a first workload on behalf of a first client or application does not interfere with a second workload on behalf of a second client or application. As a first such example, the server may utilize process and data isolation techniques to ensure that a first workload on behalf of a first client cannot achieve unauthorized access to a second workload on behalf of a second client, including accessing data owned by the second workload or even identifying the presence of the second workload, including the second client or application for which the second workload is processed.

[0007] As a second such example, the server may protect against resource overutilization. For instance, if a first workload through the server begins exhibiting a surge of volume that exceeds the share of computing resources allocated to the first workload, the use of a resource-sharing technique may enable the server to confine the consequences of the excessive volume to the first workload and to avoid impacting the processing of the second workload, such that a performance guarantee extended to the second workload remains fulfilled. In this manner, servers may allocate and regulate computing resource utilization to promote fulfillment of performance guarantees and allocate computational resources fairly over all of the workloads handled by the server.

SUMMARY

[0008] This Summary is provided to introduce a selection of concepts in a simplified form that are further described below in the Detailed Description. This Summary is not intended to identify key factors or essential features of the claimed subject matter, nor is it intended to be used to limit the scope of the claimed subject matter.

[0009] The processing of workloads through a large-scale server set, in view of performance guarantees, may encounter difficulties due to the sequential nature of the workloads and interactions of servers along various process paths. For example, a workload may involve a process path that involves a sequence of tasks to be performed by an intake server, a query processing server, and a storage server, and to fulfill a performance guarantee. While the respective servers may utilize computational resource allocation to handle the individual tasks of the workload, a problem may arise if the storage server begins to experience an excessive computational load. Such excessive computational load may arise, e.g., due to an over-allocation of tasks onto the storage server; a shortage of computational resources, such as a reduction of network bandwidth; or an unanticipated surge of processing, such as a failure of a process on the storage server that necessitates the invocation of a recovery process. In addition to slowing the processing of the workload through the storage server, the excessive computational load of the storage server may create a processing jam between the storage server and the query processing server that is upstream of the storage server in the process path. For example, the query processing server may continue to

handle the query processing task of the workload at the same rate, but the rate at which the query processing server is able to pass the workload to the storage server may be diminished. The query processing server may address the discrepancy in various ways, such as utilizing an outbound queue for the completed workload of processed queries; however, if the reduced processing rate of the storage server persists, the outbound queue may overflow. Moreover, the additional processing burden placed upon the query processing server may propagate the processing jam upward, even to the point of potentially affecting other workloads that have a processing path through the query processing server that do not utilize the storage server. The resulting gridlock may cause a widespread failure of performance guarantees for a variety of workloads due the processing jam between the storage server and the upstream query processing server.

[0010] In view of such problems, it may be desirable to configure the server set to evaluate the processing paths of the various workloads, and to provide techniques for mitigating a processing jam that may arise between a particular server and a downstream server. In particular, a server of a process path may estimate, measure, and/or monitor its processing capabilities, and compare such processing capabilities with the performance guarantees of the workloads utilizing a process path through the server. If the server detects a risk of failing the performance guarantee (e.g., due to an overprovisioning of the server or a computational resource shortage), the server may transmit a performance capability alert to an upstream server of the server path, as an indication that the workload being passed to the server may be too large to ensure that the performance guarantees are met. The upstream server that receives the performance capability alert may respond by rate-limiting its processing of the workload, within the performance guarantee, thereby downscaling the processing rate of the upstream server upon the workload to match the diminished processing rate of the downstream server. In some scenarios, the upstream server may propagate the performance capability alert further upstream. If the performance capability alert reaches a first server of the server path, the first server may refuse or slow a workload acceptance rate. In this manner, the server set may adapt the acceptance of the workload for which the process path is capable of fulfilling the performance guarantee, and in response to fluctuating processing capabilities (including an unexpected loss of processing capability) of the servers of the server path.

[0011] A first embodiment of the presented techniques involves a server of a server set that performs workloads according to a performance guarantee. The server comprises a processor and a memory storing instructions that, when executed by the processor, cause the server to operate in accordance with the techniques presented herein. In particular, the server performs a task of the workload according to a performance guarantee, wherein the workload is processed through the server set according to a process path. On condition of receiving, from a downstream server of the process path, a performance capability alert indicating that a computational load of the downstream server risks failing the performance guarantee for the workload, the server may rate limit the task of the workload to reduce the computational load of the downstream server. After completing the task (to which the rate-limit may have been applied), the server delivers the workload to the downstream server.

[0012] A second embodiment of the presented techniques involves a method of configuring a server of a server set to participate in workloads. The method involves, executing, by a processor of the server, instructions that cause the server to operate in accordance with the techniques presented herein. In particular, the server receives a workload from an upstream server of a process path of the workload, wherein the workload is associated with a performance guarantee. The server performs a task on the workload, and further identifies a performance capability of the server and compares the performance capability with the performance guarantee of the workload. Responsive to determining that the performance capability risks failing the performance guarantee, the server transmits a performance capability alert to the upstream server. Additionally, responsive to receiving a performance capability alert from a downstream server of the process path, the server rate limits a receipt of additional workloads from the upstream server.

[0013] A third embodiment of the presented techniques involves a method of configuring a server set to perform a workload according to a performance guarantee. The method involves configuring a server within a process path of the workload through the server set to operate in accordance with the techniques presented herein. The method further involves configuring the server to perform a task on the workload according to the performance guarantee. The method further involves configuring the server to receive a performance capability alert from the downstream server, wherein the performance capability alert indicates that a computational load of the downstream server risks failing the performance guarantee for the workload. The method further involves configuring the server to rate-limit the task of the server to reduce the workload delivered to the downstream server. The method further involves configuring the server to, after performing the task on the workload, deliver the workload to a downstream server of the process path.

[0014] To the accomplishment of the foregoing and related ends, the following description and annexed drawings set forth certain illustrative aspects and implementations. These are indicative of but a few of the various ways in which one or more aspects may be employed. Other aspects, advantages, and novel features of the disclosure will become apparent from the following detailed description when considered in conjunction with the annexed drawings.

DESCRIPTION OF THE DRAWINGS

[0015] FIG. **1** is an illustration of an example scenario featuring a processing of workloads through a server set.

[0016] FIG. **2** is an illustration of an example scenario featuring a processing of a workload through a server set in accordance with the techniques presented herein.

[0017] FIG. **3** is a component block diagram illustrating an example server featuring an example system for configuring a server set to process a workload in accordance with the techniques presented herein.

[0018] FIG. **4** is a flow diagram illustrating an exemplary method of configuring a server to process a workload through a process path of a server set in accordance with the techniques presented herein.

[0019] FIG. **5** is a flow diagram illustrating an exemplary method of configuring a server set to process a workload through a process path in accordance with the techniques presented herein.

[0020] FIG. **6** is an illustration of an example computer-readable medium storing instructions that provide an embodiment of the techniques presented herein.

[0021] FIG. **7** is an illustration of a set of example scenarios featuring a variety of rate-limiting mechanisms for a task in accordance with the techniques presented herein.

[0022] FIG. **8** is an illustration of a set of example scenarios featuring a variety of process path modifications for a workflow in accordance with the techniques presented herein.

[0023] FIG. **9** illustrates an exemplary computing environment wherein one or more of the provisions set forth herein may be implemented.

DETAILED DESCRIPTION

[0024] The claimed subject matter is now described with reference to the drawings, wherein like reference numerals are used to refer to like elements throughout. In the following description, for purposes of explanation, numerous specific details are set forth in order to provide a thorough understanding of the claimed subject matter. It may be evident, however, that the claimed subject matter may be practiced without these specific details. In other instances, well-known structures and devices are shown in block diagram form in order to facilitate describing the claimed subject matter.

A. Introduction

[0025] FIG. **1** is an illustration of an example scenario **100** featuring a server set **106** that processes workloads **104** on behalf of clients **102**. In this example scenario **100**, the server set **106** comprises a distributed query processing system that accepts queries from clients **102** and processes the queries over a data set. However, the example scenario **100** may similarly apply to a variety of workloads **104**, such as content generation, media rendering and presentation, communication exchange, simulation, supercomputing, etc.

[0026] In this example scenario **100**, the server set **106** comprises a set of server **108** that respectively perform a task **110**. For example, a pair of intake servers **108** may serve as a front-end, client-facing interface that accepts queries to be processed on behalf of the clients **102**; a query processing server **108** that parses queries, such as translating the query from a query language into a sequence of logical relationships to be applied over the data set; and a pair of storage servers **108** that store a replica or a portion of the data set over which the queries are to be applied. The servers **108** may be arranged such that the workloads **104** of the clients **102** are processed according to a process path **112**, e.g., a sequence of servers **108** that respectively apply a task **110** to the workload **104** and deliver the partially processed workload **104** to the next, downstream server **108** in the process path **112**. The process path **112** may enable a pipelined evaluation of workloads **104** that enable the servers **108** to apply the tasks **110** in the manner of an assembly line, thereby reducing idle processing capacity and promoting the scalability of the server set **106** to handle a significant volume of workloads **104** in a concurrent manner.

[0027] As further shown in the example scenario **100** of FIG. **1**, different process paths **112** may be utilized for different workloads **104**; e.g., the first intake server **108** may receive workloads **104** from a first set of clients **102** and/or geographic regions while the second intake server **108** receives workloads **104** from a second set of clients **102** and/or geographic regions. Similarly, the first workload **104** may present a first query over the portion of the database stored by the first storage server **108**, while the second workload **104** may present a second query over the portion the database stored by the second storage server **108**. Conversely, the process paths **112** of different workloads **102** may coincide through one or more servers **108**; e.g., both workloads **104** utilize the same query processing server **108**.

[0028] As further shown in the example scenario **100** of FIG. **1**, the workloads **104** may be associated with various kinds of performance constraints. As a first such example, the first workload **104** may be particularly sensitive to latency; e.g., the first workload **104** may comprise time-sensitive data that the client **102** seeks to process in an expedited manner, and delays in the completion of the workload **104** may be highly visible, and may reduce or negate the value of the completed workload **104**. As a second such example, the second workload **104** may involve fluctuating volume, such as a data-driven service that is sometimes heavily patronized by users and other times is used by only a few users. The second workload **104** may therefore be sensitive to scalability, and may depend upon comparatively consistent processing behavior of the server set **106** to handle the second workload **104** even as demand scales upward. The performance dependencies may be driven, e.g., by the computational constraints of the workload **104**; the intended uses of the results of the processing; the circumstances of an application for which the workload **104** is performed; and/or the preferences of the client **102** submitting the workload **104**. Moreover, the sensitivities and/or tolerances of different clients **102** and workloads **104** may vary; e.g., the first workload **104** may present a highly consistent and regular volume such that which scalability is not a concern, while the second client **102** is able to tolerate reasonable variations in latency **114**, such as marginally delayed completion of the workloads **104**, as long as processing is completed correctly at peak volume.

[0029] Due to the performance dependencies of the workloads **104**, the server set **106** may extend to each client **102** a performance guarantee **114** of a performance capability of the server set **106** to handle the workload **104**. For example, the server set **106** may extend to the first client **102** a performance guarantee **114** that processing of the majority of workloads **104** (e.g., 95% of workloads **104**) will complete within 10 milliseconds. For the second client **102**, the server set **106** may offer a performance guarantee **114** of correct processing of the second workload **104** up to a defined volume, such as 1,000 requests per second. The server set **106** may be arranged to fulfill the performance guarantees **114** of the workloads **104**, e.g., by supplementing, adapting, and/or optimizing the configuration of servers **108** comprising the data set.

[0030] When a server set **106** is arranged such that a particular server **108** processes different workloads **104**, particularly for different clients **102**, problems may arise due to the concurrent and/or consecutive sharing of the server **108**. As a first example, the server **108** may have to secure and isolate the workloads **104** of the first client **102** from the workloads **104** of the second client **102**, e.g., to prevent the second load **104** from accessing proprietary information of the first client **102** and tampering with the operation of the first workload **104**. The significance of isolation may grow with the scalability of the server set **106**; e.g., a particular

server **108** may process hundreds or even thousands of workloads **104** of various clients **102**, and safeguarding the information of each client **102** may be a high priority. As a second example, the server **108** may comprise a limited set of computational resources, such as processor capacity, storage capacity, and network bandwidth. An overconsumption of the computational resources of the query processing server **108** by the first workload **104** may create a shortage of computational resources of the server **108** for the second workload **104**, such as limited processing capacity; an exhaustion of available storage; and/or constrained network bandwidth. Such overconsumption by the first workload **104** may lead to delays or even a failure in the processing of the second workload **104**, including a failure of the performance guarantee **114** of the second workload **104**, such as an inability of the query processing server **108** to scale up in order to handle peak volume of the second client **102**. In view of such concerns, techniques may be utilized to allocate and compartmentalize the computational resources of the server **108** for each workload **104**, such as processor time-slicing and per-workload quotas or caps on storage and network capacity. Using such techniques, a server **108** may limit the adverse effects of an overconsumption of resources to the workload **104** responsible for the overconsumption; e.g., increased processing demand by the first workload **104** may result in delayed completion of the first workload **104** without impacting the processing of the second workload **104**. However, resource limitations may also consume computational resources (e.g., processor time-slicing among a set of workloads **104** may present overhead due to context-switching). Such inefficiency may scale with the scalability of the server set **106**, such as using a particular server **108** to process hundreds or even thousands of workloads **104**, so the isolation and allocation techniques may have to be implemented with careful attention to efficiency.

[0031] Other developments may also present a potential source of failure of a performance guarantee **114** of a particular workload **104**. For example, computational resources may be limited by systemic factors, such as a shortage of storage capacity due to a failure of a hard disk drive in a storage array, or a surge in a computational process, such as a background maintenance process. An introduction of line noise into a network connection, such as due to electromagnetic interference or a faulty cable, may lead to diminished throughput and increased latency. The server set **106** may experience a failure of a particular server **108** and may have to re-route the workload **104** of the failed server to other servers **108** of the server set **106**, thereby increasing the computational load of the individual servers **108**. Any such change in the performance of the server set **106** may interfere with the process path **112** of a workload **104**, which may risk failing the performance guarantee **114**.

[0032] In view of such risks, load-balancing techniques are often utilized to detect and mitigate computational overload of a particular server **108**. For example, the respective servers **108** of the server set **106** may include a monitoring process of a performance capability, such as available processor capacity, storage, network throughput, and latency. The performance capabilities may also include considerations such as resiliency to data loss, e.g., the volume of data stored by the server **108** that has not yet been transmitted to a replica, as a measure of the risk of data loss in the event of a failure of the server **108**. The server **108** may track the performance capabilities and, if detecting a potential shortage that risks failing a performance guarantee **114**, may invoke a variety of "self-help" measurements to alleviate the shortage. For example, in the event of a shortage of processing capacity, the server **108** may place the processor into a "boost" mode; awaken and utilize dormant processing capacity, such as additional processing cores; and/or reduce or suspend some deferrable processing, such as maintenance tasks. In the event of a shortage of storage capacity, the server **108** may delete or compress data that is not currently in use, including significant data that may later be restored from a replica. In the event of a shortage of network capacity, the server **108** may suspend processes that are consuming network capacity, or shift network bandwidth allocation from processes that are tolerant of reduced network bandwidth and latency to processes that are sensitive to constrained network bandwidth or latency.

[0033] Alternatively or additionally to such "self-help" techniques, the server **108** may report the performance capability shortage to a network administrator or network monitoring process, which may intercede to reconfigure the server set **106**. For example, the computational load of a storage server **108** may be alleviated by provisioning a new server **108**, replicating the data set onto the new server **108**, and altering process paths **112** to utilize the new server **108**. However, reconfiguration of the architecture of the server set **106** may be a comparatively expensive step, and/or may involve a delay to implement, during which time the performance guarantee **114** of a workload **104** may fail.

[0034] However, these and other techniques may be inadequate to address a particular source of interference with the processing of the server set **106** that may jeopardize performance guarantees **114**.

[0035] As further illustrated in the example scenario **100** of FIG. **1**, at a second time **122**, a storage server **108** may encounter a processing capacity shortage **116** that delays the processing of a workload **104** through the storage server **108**. Such delay by the storage server **108** may lead to a lag in the acceptance by the storage server **108** of the workload **104** delivered by the upstream query processing server **108**. That is, the query processing server **108** may complete the task **110** of parsing a number of queries that are to be applied by the second storage server **108**, but the second storage server **108** may not be ready to accept the parsed queries. In some cases, the acceptance rate of the second storage server **108** may be diminished; in other cases, the acceptance rate of the second storage server **108** may be reduced to zero, such as an overflow of an input queue that the query processing server **108** uses to record parsed queries for processing by the second storage server **108**. The interface between the query processing server **108** and the storage server **108** may therefore experience a processing jam **118** that interferes with the delivery of the partially processed workload **104** from the query processing server **108** to the storage server **108**.

[0036] The query processing server **108** may respond to the processing jam **118** in numerous ways. For example, the query processing server **108** may retry the delivery of the workload **104** for a period of time, in case the processing jam **118** is ephemeral and is momentarily alleviated. The query processing server **108** may utilize an outbound queue for the workload **104** that the storage server **108** may be able to work through and empty when the processing capacity shortage **116** is alleviated, or that may be transferred to a

replica of the storage server **108** following a reconfiguration of the server set **106**. However, these techniques may also fail if the processing jam **118** is prolonged and a substitute for the storage server **108** is unavailable. The outbound queue of the query processing server **108** may also overflow, or the workloads **104** allocated to the query processing server **108** may begin to starve, inducing a failure of the performance guarantee **114**. In some cases, the source of the fault may be misattributed to the query processing server **108**, since the performance guarantees **114** failed while the query processing server **108** retained the workloads **104** for a prolonged period. For example, an automated diagnostic process may identify the query processing server **108** as a processing bottleneck, and may initiate a failover of the query processing server **108** that fails to resolve the actual limitation of the performance of the server set **106**.

[0037] As further illustrated in the example scenario **100** of FIG. **1**, at a third time **124**, even more significant problems may arise when the processing capacity shortage **116** of the storage server **108** spills over to create a processing capacity shortage **116** of the upstream server. For example, the volume of completed workloads **104** that the query processing server **108** that are pending delivery to the storage server **108** may cause delays in the handling of other workloads **104** by the query processing server **108**. This backward propagation of the processing capacity shortage **116** may create processing jam **118** in the interfaces of the query processing server **108** not with the second intake server **108** along the same process path **112** of the second workload **104**. Moreover, the processing capacity shortage **116** may create a processing jam in the interface with the first intake server **108**, leading to delayed processing and completion of the first workload **104**, even though the process path **112** of the first workload **104** does not include the second storage server **108**. In this manner, the processing capacity shortage **116** of the second storage server **108** may induce delays in other servers **108** and process paths **112** of the server set **106**, and the failure of performance guarantees **114** even of workloads **104** that do not utilize the second storage server **108**.

B. Presented Techniques

[0038] In view of the problems depicted in the example scenario **100** of FIG. **1**, a server set **106** that handles a variety of workloads **104** and process paths **112**, such as multitenant distributed server sets **106**, may benefit from the use of techniques to detect and alleviate processing jams **118** that occur between servers **108**, wherein a processing capacity shortage **116** of a downstream server **108** impacts the performance capabilities of an upstream server **108**.

[0039] Additionally, because such incidents may occur suddenly, may quickly present risks to the failure of a performance guarantee **114**, and may often be only transient, it may be advantageous to utilize techniques that may be applied rapidly and without involving a significant and potentially expensive allocation of resources, such as inducing failover of the afflicted server **108** to a substitute server **108**. It may also be advantageous to utilize techniques that may be applied automatically in the locality of the afflicted server **108**, without necessarily resorting to a centralized manager that holistically evaluates the server set **106** to identify potential solutions, and/or without involving a human administrator who may not be able to respond to the processing capacity shortage **116** in due time.

[0040] Additionally, because the effects of a processing jam **118** may spill over onto other servers **108** in a process path **112**, it may be advantageous to provide techniques that may be easily propagated to a broader neighborhood of the afflicted server **108**, and therefore expand to incorporate the other servers **108** in the resolution of the processing capacity shortage **116**.

[0041] FIG. **2** is an illustration of an example scenario **200** featuring a server set **200** that operates in accordance with the techniques presented herein. In this example scenario **200**, a server set **106** processes a workload **104** as a sequence of servers **108** that apply respective tasks **110** as a process path **112**. The workload **104** is associated with a performance guarantee **114**, such as a maximum total processing duration of the workload **104**; a scalability guarantee that the process path **112** will remain capable of handling the workload **104** at a higher volume; and/or a resiliency of the server set **108** to data loss, such as a maximum volume of data of the workload **104** that is not replicated over at least two replicas and that is therefore subject to data loss.

[0042] At a first time **210**, the servers **108** of the server set may apply the tasks **110** to the workload **104**, where each server **108** completes the task **110** on a portion of the workload **104** and delivers the partially completed workload **104** to the next downstream server **108** of the process path **112**. Additionally, the servers **108** may individually monitor the performance capabilities **202**, and compare the performance capabilities **202** with the performance guarantee **114**. For example, if the performance guarantee **114** comprises a maximum latency, such as 10 milliseconds, the respective servers **108** may monitor the duration of completing the task **110** over a selected portion of the workload **104** to ensure that the task **110** is completed within 2.5 milliseconds on each server **108**. If the performance guarantee **114** comprises a maximum volume of unreplicated data that is subject to data loss, the server **108** may monitor and manage a queue of unreplicated data that is awaiting synchronization with a replica. In this manner, the respective servers **108** may ensure that the performance capabilities **202** of the individual servers **108** are sufficient to satisfy the performance guarantee **114**; such that maintaining adequate individual performance capabilities **202** of all servers **108** in the server path **112** results in a satisfaction of the performance guarantee.

[0043] However, at a second time **212**, the third server **108** in the process path **112** may detect a diminished performance capability **202**, such as limited processing capacity, storage capacity, or network bandwidth. Comparison of the diminished performance capability **202** with the performance guarantee **114** may reveal a processing capacity shortage **116** that introduces a risk **204** of failing the performance guarantee **114** for the workload **114**.

[0044] In some circumstances, the third server **108** may be capable of utilizing "self-help" measures to restore the performance capability **202**. In other circumstances, the processing capacity shortage **116** may rapidly be identified as severe and unresolvable, such as a complete failure of a storage device that necessitates substitution of the third server **108**. However, in some circumstances, the diminished performance capability **202** may be resolved by temporarily reducing the workload **104** handled by the third server **108**. Such reduction of the workload **104** may be achieved by reducing the delivery of the workload **104** to the third server **108** by the upstream servers **108** of the process path **112**.

Such reduction may provide a window of opportunity in which the third server **108** may apply the available performance capabilities **202** to a workload **104** of reduced volume, which may enable the third server **108** to catch up with the volume of the workload **104**. For instance, the third server **108** may utilize an input buffer of workloads **104** delivered by the upstream server **108**. If the rate at which the workload **104** is delivered into the input buffer exceeds the rate at which the third server **108** removes and completes the workload **104** from the input buffer, the input buffer may steadily grow to reflect a deepening processing queue with a growing latency. Reducing the input rate of delivery of the workload **104** into the input buffer below the rate at which the third server **108** takes the workload **104** out of the input buffer may shrink the input buffer and enable the third server **108**. When the input buffer is depleted or at least reduced to an acceptable latency, and/or the cause of the diminished performance capability **202** and processing capacity shortage is resolved, the input rate to the input buffer may be restored.

[0045] As further shown in the example scenario **200** of FIG. **2**, the reduction of the delivery rate of the workload to the third server **108** may be achieved through coordination with the upstream servers **108**. At a third time point **214**, responsive to detecting the processing capacity shortage **116** and identifying the risk **204** of failing the performance guarantee **114**, the third server **108** may transmit a performance capability alert **206** to the upstream server **108**. The second server may receive the performance capability alert **206** and respond by applying a rate limit **208** the task **110** performed on the workload **104** by the second server **108**. The rate limit **208** may comprise, e.g., slowing the rate at which the task **110** is performed on the workload **104**, such as by reducing the processing priority of the task **110**, a processor rate or core count of a processor that handles the task **110**, or an allocation of network bandwidth used by the task **110**. The rate limit **208** may also comprise slowing the acceptance rate of the workload **104** by the second server **108** from the upstream first server **108** and thereby reducing the rate of the completed workload **104** delivered to the third server **108**. The rate limit **208** may also comprise enqueuing the workload **104** received from the upstream first server **108** for a delay period; and/or enqueuing the workload **104** over which the task **110** has been completed for a delay period before attempting delivery to the third server **108**.

[0046] The second server **108** may continue to apply the rate limit **208** to task **110** for the duration of the processing capacity shortage **116** of the third server **108**. For example, the third server **108** may eventually report an abatement of the processing capacity shortage **116**, or the second server **108** may detect such abatement, e.g., by detecting an emptying of the outbound queue to the third server **108**, at which point the second server **108** may remove the rate limit **208** and resume unrate limited processing of the task **110** for the workload **104**. Alternatively, if the processing capacity shortage **116** is prolonged or indefinite, or if the second server **108** identifies that applying the rate limit **208** to the task **110** may impose a new risk **204** of failing the performance guarantee **114**, the second server **108** may propagate the performance capability alert **206** to the next upstream server **108** of the process path **112**, i.e., the first server **108**. The first server **108** may similarly respond to the performance capability alert **206** by applying a rate limit **208** to the task **110** of the first server **108** over the workload **104**. The first server **108**, as the intake point of the workload **104**, may reduce the commitment of the entire process path **112** to a smaller workload volume over which the performance capability **202** may be guaranteed even while afflicted with the processing capacity shortage **116**. In this manner, the backward propagation of performance capability alerts **206** and the application of a rate limit **208** the task **110** of the second server **108** operate as a form of "backpressure" on the upstream servers **108** of the process path **112**, which reduces the computational overload of the third server **108** and promotes the satisfaction of the performance guarantee **114** of the server set **106** over the workload **104** in accordance with the techniques presented herein.

C. Technical Effects

[0047] A first technical effect that may arise from the techniques presented herein involves the resolution of the processing capacity shortage **116** of a downstream server **108** and the risk **204** of failing the performance guarantee **114** of the workload **104** through the application of backpressure on upstream servers **108** of the process path **112**.

[0048] The use of a rate limit **208** by the second server **108** in accordance with the techniques presented herein may effectively address the process capacity shortage **116** and the risk **204** of failure of the performance guarantee **114** of the workload **104** in numerous ways. As a first such example, it may be feasible for the second server **108** to apply the rate limit **208** to the task **110** may be feasible by the second server **108** without the introduction of the rate limit **208** exacerbating the risk **204** of failing the performance guarantee **114**. For example, the second server **108** may have a surplus performance capability **202**, and may be capable of working through the workload **104** significantly faster than required by the performance guarantee **114** (e.g., the second server **108** may have a maximum allocation of 2.5 milliseconds to perform the task **110** over the workload **104** within the performance guarantee **114**, but may be capable of completing the task **110** in only 0.7 milliseconds). That is, the application of the rate limit **208** to the task **110** may offload some of the delay caused by the processing capacity shortage **116** from the third server **108** to the second server **108**, thus enabling the third server **108** to work through a backlog of the workload **104** and restore the performance capability **202**.

[0049] As a second such example, the second server **108** and third server **108** may share two workloads **104**, wherein the processing capacity shortage **116** may introduce a risk **204** of failing the performance guarantee **114** of the first workload **104**, but may pose no risk **204** of failing the performance guarantee **114** of the second workload **104** (e.g., the first workload **104** may be sensitive to latency, while the second workload **104** may be relatively tolerant of latency). The application of the rate limit **208** to the task **110** of the second server **108** may reduce the rate of delivery to the third server **108** of both the first workload **104** and the second workload **104**. The reduced volume of the second workload **104** may enable the third server **108** to apply the performance capability **202** to work through a backlog of the first workload **104** and therefore alleviate the processing capacity shortage **116**, without introducing a risk **204** of failing a performance guarantee **114** for the second workload **104** that is not significantly affected by increased latency.

[0050] As a third such example, the use of a performance capability alert **206** and rate limit **208** may be applied to

further upstream servers **108**. For example, in the example scenario **200** of FIG. **2**, the second server **108** may be unable to apply a rate limit **208** to the task **110** without creating a further risk **204** of failing the performance guarantee **114** (e.g., the margin between the performance capability **202** of the second server **108** and the performance guarantee **114** may already be thin). Alternatively, the rate limit **208** may initially be applied to the task **110** by the second server **108**, but a protracted and/or unresolvable processing capacity shortage **116** by the third server **108** may eventually render the rate limit **208** insufficient, such as an overflow of the outbound queue of the second server **108**, or where the application of the rate limit **208** to the task **110** introduces a risk **204** of failing a performance guarantee **114** of another workload **104** over which the server **108** applies the task **110**. In these and other ways, the "backpressure" induced by the backward propagation of the performance capability alert **206** and the application of the rate limit **208** to a task **110** of an upstream server **108** may effectively alleviate the processing capacity shortage **116** of the downstream server **108**.

[0051] A second technical effect that may arise from the techniques presented herein involves the capability of the server set **106** to respond to performance capacity shortages in an efficient, rapid, and automated manner. As a first such example, the techniques presented herein may be applied without conducting a holistic, extensive analysis of the capacity of the server set **106**, such as may be performed by a network monitoring or network administrator, to determine the root cause of the processing capacity shortage **116** and to assess the available options. Rather, the server **108** afflicted by diminished performance capability **202** may simply detect the processing capacity shortage **116** and transmit the performance capability alert **206** to the upstream server **108**. As a second such example, the techniques presented herein do not involve a significant and potentially expensive reconfiguration of the server set **106** or a commitment of resources, such as provisioning a substitute server for the afflicted server **108**, which may involve remapping associations within the server set **106** and/or introduce a delay in the recovery process. In some cases, the delay involved in applying the recovery may outlast the duration of the processing capacity shortage **116**. In other cases, the performance guarantee **114** for the workload **104** may fail during the delay involved in applying such heavy recovery techniques. In some circumstances, such recovery may impose additional computational load on the afflicted server **108**, thus hastening the failure of the performance guarantee **114**. By contrast, the comparatively simple techniques presented herein are applicable merely by transmitting the performance capability alert **206** to the upstream server **108** and causing the second server **108** to apply the rate limit **208** to the task **110** may be applied rapidly and with a negligible expenditure of resources, and may therefore be effective at resolving some processing capacity shortages **116**, particularly serious but ephemeral shortages, that other techniques may not adequately address. Moreover, the transmission of the performance capability alert **206** and the application of the rate limit **208** to the task **110** utilize currently existing and available resources and capabilities of the downstream and upstream servers (e.g., processor clock rate adjustment, adjustment of thread and process priorities, and/or the use of queues), and do not depend upon the introduction of complex new process management machinery or protocols.

[0052] A third technical effect that may arise from the techniques presented herein involves the extension of the process to reduce or avoid the risk **204** of failing the performance guarantee **114** altogether. In the example scenario **200** of FIG. **2**, the first server **108** is positioned at the top of the process path **112** and serves as an intake point for the workload **104**. If the server set **106** propagates the performance capability alert **206** all the way to the first server **108** at the top of the process path **112**, the first server **108** may respond by reducing the acceptance rate of the workload **104** into the process path **112**. That is, rather than imposing a risk **204** of failing the performance guarantee **114** of a previously accepted workload **104**, reducing the acceptance rate of the workload **104** into the process path **112** may alleviate the risk **204** altogether by reducing the volume of the workload **104** over which the performance guarantee **114** is offered. In more significant cases such as a protracted or indefinite processing capacity shortage **116**, the first server **108** may reduce the performance guarantee **114** that is offered for the workload **104** (e.g., raising a latency performance guarantee **114** of the workload **104** from 10 milliseconds to 50 milliseconds), and/or may altogether refrain from offering a performance guarantee **114** or accepting new workloads **104** until the processing capacity shortage **116** is alleviated. In this manner, the "backpressure" techniques presented herein may enable the process path **112** to respond to processing capacity shortages **116** by reducing the initial commitment of the server set **108** to the workload, thus avoiding problems of overcommitment of the server set **108** by only offering performance guarantees **114** that the process path **112** is capable of fulfilling. Many such technical effects may arise from the processing of the workload **104** by the server set **106** in accordance with the techniques presented herein.

D. Example Embodiments

[0053] FIG. **3** is an illustration of an example scenario **300** featuring some example embodiments of the techniques presented herein, including an example server **302** that processes a workload **104** as part of a server set **106**. The example server **302** comprises a processor **304** and a memory **306** (e.g., a memory circuit, a platter of a hard disk drive, a solid-state storage device, or a magnetic or optical disc) encoding instructions that, when executed by the processor **304** of the example server **302**, cause the example server **302** to process the workload **104** in accordance with the techniques presented herein. More particularly, in this example scenario **300**, the instructions encode components of example system **308** that perform various portions of the presented techniques. The interoperation of the components of the example system **308** enables the example server **302** to process the workload **104** in accordance with the techniques presented herein.

[0054] The example system **308** comprises a task processor **310**, which performs a task **110** of the workload according to a performance guarantee, wherein the workload **104** is processed through the server set **106** according to a process path **112** that includes an upstream server **108** and a downstream server **108** relative to the example server **302**. The example system **308** also includes a task rate limit **314**, which receives a performance capability alert **206** from a downstream server **108** of the process path **112**, e.g., in response to a comparison of the performance capability **202** of the downstream server **108** with the performance guar-

antee **114** of the workload **104**, which indicates a processing capacity shortage **116** and a risk **204** of failing the performance guarantee **114** of the workload **104**. Responsive to the performance capability alert **206**, the task rate limit **314** applies a rate limit **208** to the task **110** performed on the workload **104** to reduce the computational load of the downstream server **108**. The example system **308** also includes a workload streamer **312**, which, after completion of the task **110** on the workload **104**, delivers the workload **104** to the downstream server **108** of the process path **112**. In this manner, the example system **308** enables the example server **302** to apply the task **110** to the workload **104** as part of the process path **112** in accordance with the techniques presented herein.

[0055] FIG. **4** is an illustration of an example scenario featuring a third example embodiment of the techniques presented herein, wherein the example embodiment comprises an example method **400** of configuring a server **108** to process a workload **104** in accordance with techniques presented herein. The example method **400** involves a server **108** comprising a processor **304**, and may be implemented, e.g., as a set of instructions stored in a memory **306** of the server **108**, such as firmware, system memory, a hard disk drive, a solid-state storage component, or a magnetic or optical medium, wherein the execution of the instructions by the processor **304** causes the server **108** to operate in accordance with the techniques presented herein.

[0056] The example method **400** begins at **402** and involves executing **404**, by the server, instructions that cause the server to perform in the following manner. The execution of the instructions causes the server **108** to receive **406** a workload **104** from an upstream server **108** of a process path **112**, wherein the workload **104** is associated with a performance guarantee **114**. The execution of the instructions also causes the server **108** to perform **408** a task **110** on the workload **104**. The execution of the instructions also causes the server **108** to identify **410** a performance capability **202** of the server **108**. The execution of the instructions also causes the server **108** to compare **412** the performance capability **202** with the performance guarantee **114** of the workload **104**. The execution of the instructions also causes the server **108** to respond to determining that the performance capability **202** risks failing the performance guarantee **114** by transmit **414** a performance capability alert **206** to the upstream server **108**. The execution of the instructions also causes the server **108** to respond to a performance capability alert **206** received from a downstream server **108** of the process path **112** by rate-limiting **416** the task **110** performed on the workload **104** to reduce the computational load of the downstream server **108**. In this manner, the example method **400** may enable the server **108** to process the workload **104** as part of the process path **112** in accordance with the techniques presented herein, and so ends at **418**.

[0057] FIG. **5** is an illustration of an example scenario featuring a third example embodiment of the techniques presented herein, wherein the example embodiment comprises an example method **500** of configuring a server set **106** to process a workload **104** that is associated with a performance guarantee **114** in accordance with techniques presented herein. The example method **500** involves a server set **106** comprising a collection of servers **108** respectively comprising a processor **304**, and may be implemented, e.g., as a set of instructions stored in a memory **306** of the server **108**, such as firmware, system memory, a hard disk drive, a solid-state storage component, or a magnetic or optical medium, wherein the execution of the instructions by the processor **404** causes the server **108** to operate as a member of the server set **106** in accordance with the techniques presented herein.

[0058] The first example method **500** begins at **502** and involves configuring a server **108** of the server set **106** that is within the process path **112** to process the workload **104** in the following manner. The server **108** performs **506** a task **110** on the workload **104** according to the performance guarantee **114**. The server **108** further receives **508** a performance capability alert **206** from the downstream server **108**, wherein the performance capability alert **206** indicates that a computational load of the downstream server **108** risks failing the performance guarantee **114** for the workload **104**. Responsive to the performance capability alert **206**, the server **108** further rate-limits **510** the task **110** of the server **108** to reduce the workload delivered to the downstream server **108**. After performing the task **110** on the workload **104**, the server **108** further delivers **512** the workload **104** to a downstream server **108** of the process path **112**. In this manner, the example method **500** may enable the server **108** to operate as part of a server set **106** to participate in the processing of the workload **104** in accordance with the techniques presented herein, and so ends at **514**.

[0059] Still another embodiment involves a computer-readable medium comprising processor-executable instructions configured to apply the techniques presented herein. Such computer-readable media may include various types of communications media, such as a signal that may be propagated through various physical phenomena (e.g., an electromagnetic signal, a sound wave signal, or an optical signal) and in various wired scenarios (e.g., via an Ethernet or fiber optic cable) and/or wireless scenarios (e.g., a wireless local area network (WLAN) such as WiFi, a personal area network (PAN) such as Bluetooth, or a cellular or radio network), and which encodes a set of computer-readable instructions that, when executed by a processor of a device, cause the device to implement the techniques presented herein. Such computer-readable media may also include (as a class of technologies that excludes communications media) computer-computer-readable memory devices, such as a memory semiconductor (e.g., a semiconductor utilizing static random access memory (SRAM), dynamic random access memory (DRAM), and/or synchronous dynamic random access memory (SDRAM) technologies), a platter of a hard disk drive, a flash memory device, or a magnetic or optical disc (such as a CD-R, DVD-R, or floppy disc), encoding a set of computer-readable instructions that, when executed by a processor of a device, cause the device to implement the techniques presented herein.

[0060] An example computer-readable medium that may be devised in these ways is illustrated in FIG. **6**, wherein the implementation **600** comprises a computer-readable memory device **602** (e.g., a CD-R, DVD-R, or a platter of a hard disk drive), on which is encoded computer-readable data **604**. This computer-readable data **604** in turn comprises a set of computer instructions **606** that, when executed on a processor **304** of a server, cause the server to operate according to the principles set forth herein. For example, the processor-executable instructions **606** may encode a system that processes a workload **104** as part of a server set **106**, such as the example system **308** of FIG. **3**. As another

example, the processor-executable instructions **606** may encode a method of configuring a server **108** to process a workload **104** as part of a server set **106**, such as the example method **400** of FIG. **4**. As yet another example, the processor-executable instructions **606** may encode a method of configuring a server set **106** to process a workload **104**, such as the example method **500** of FIG. **5**. Many such computer-readable media may be devised by those of ordinary skill in the art that are configured to operate in accordance with the techniques presented herein.

E. Variations

[0061] The techniques discussed herein may be devised with variations in many aspects, and some variations may present additional advantages and/or reduce disadvantages with respect to other variations of these and other techniques. Moreover, some variations may be implemented in combination, and some combinations may feature additional advantages and/or reduced disadvantages through synergistic cooperation. The variations may be incorporated in various embodiments (e.g., the first example server **302** and/or the example system **308** of FIG. **3**; the example method **400** of FIG. **4**; the example method **500** of FIG. **5**; and the example device **602** and/or example method **608** of FIG. **6**) to confer individual and/or synergistic advantages upon such embodiments.

[0062] E1. Scenarios

[0063] A first aspect that may vary among implementations of these techniques relates to scenarios in which the presented techniques may be utilized.

[0064] As a first variation of this first aspect, the presented techniques may be utilized with a variety of servers **108** and server sets **106**, such as workstations, laptops, consoles, tablets, phones, portable media and/or game players, embedded systems, appliances, vehicles, and wearable devices. The server may also comprise a collection of server units, such as a collection of server processes executing on a device; a personal group of interoperating devices of a user; a local collection of server units comprising a computing cluster; and/or a geographically distributed collection of server units that span a region, including a global-scale distributed database. Such servers **108** may be interconnected in a variety of ways, such as locally wired connections (e.g., a bus architecture such as Universal Serial Bus (USB) or a locally wired network such as Ethernet); locally wireless connections (e.g., Bluetooth connections or a WiFi network); remote wired connections (e.g., long-distance fiber optic connections comprising Internet); and/or remote wireless connections (e.g., cellular communication). Additionally, such servers **108** may serve a variety of clients **102**, such as a client process on one or more of the servers **108**; other servers **108** within a different server set **106**; and/or various client devices that utilize the server **108** and/or server group on behalf of one or more clients **102** and/or other devices.

[0065] As a second variation of this first aspect, the server set **106** may present a variety of services that involve applying tasks **110** to workloads **108**. As a first such example, the service may comprise a distributed database or data storage system, involving tasks **110** such as receiving the data; storing the data; replicating and/or auditing the data; evaluating queries over the data; and/or running reports or user-defined functions over the data. As a second such example, the service may comprise a content presentation system, such as a news service, a social network service, or social media service, which may involve tasks **110** such as retrieving and storing content items; generating new content items; aggregating content items into a digest or collage; and transmitting or communicating the content items to clients **102**. As a third such example, the service may comprise a media presentation system, which may involve tasks **110** such as acquiring, storing, cataloging, and archiving the media objects; rendering and presenting media objects to clients **102**; and/or tracking engagement of the clients **102** with the media objects. As a fourth such example, the service may comprise a software repository, which may involve tasks **110** such as storing and cataloging software; deploying software to various clients **102**; and receiving and applying updates such as patches and upgrades to the software deployed of the clients **102**. As a fifth such example, the service may comprise a gaming system, which may involve tasks **110** such as initiating game sessions; running game sessions; and compiling the results of game sessions among various clients **102**. As a sixth such example, the service may comprise an enterprise operational service that provides operational computing for an enterprise, which may involve tasks **110** such as providing a directory of entities such as individuals and operating units; exchanging communication among the entities; controlling and managing various processes; monitoring and logging various processes, such as machine sensors; and generating alerts. Those of ordinary skill in the art may devise a range of scenarios in which a server set **106** configured in accordance with the techniques presented herein may be utilized.

[0066] E2. Performance Capabilities and Performance Guarantees

[0067] A second aspect that may vary among embodiments of the techniques presented herein involves the performance capabilities **202** monitored by the servers **108** and the comparison with performance guarantees **114** over the workload **104** to identify a processing capacity shortage **116** and a risk **204** of failing the performance guarantee **114**.

[0068] As a first variation of this second aspect, the performance capabilities **202** may include, e.g., processor capacity; storage capacity; network bandwidth; availability of the server set **106**; scalability to handle fluctuations in the volume of a workload **104**; resiliency to address faults such as the failure of a server **108**; latency of processing the workload **104** through the server set **106**; and/or adaptability to handle new types of workloads **104**.

[0069] As a second variation of this second aspect, the performance guarantees **114** of the workloads **104** may involve, e.g., a processing latency, such as a maximum end-to-end processing duration for processing the workload **104** to completion; a processing throughput of the workload **104**, such as a sustainable rate of completed items; a processing consistency of the workload **104**, such as a guarantee of consistency among portions of the workload **104** processed at different times and/or by different servers **108**; scalability to handle a peak volume of the workload **104** to a defined level; a processing replication of the workload **104**, such as a maximum volume of unreplicated data that may be subject to data loss; and/or a minimum availability of the server set **106**, such as a "sigma" level.

[0070] As a third variation of this second aspect, a server **108** may identify the performance capabilities **202** in various ways. As a first such example, a server **108** may predict the performance capability of the server **202** over the workload

104, such as an estimate of the amount of time involved in applying the task 110 to the workload 104 or a realistically achievable throughput of the server 108. Such predictions may be based, e.g., upon an analysis of the workload 104, a set of typical performance characteristics or heuristics of the server 108, or previous assessments of processing the task 110 over similar workloads 104. Alternatively or additionally, the server 108 may measure the performance capability 202 of the server while performing the workload 104. Such measurement may occur with various granularity and/or periodicity, and may involve techniques such as low-level hardware monitors (e.g., hardware timers or rate meters) and/or high-level software monitors (e.g., a timer placed upon a thread executing the task 110). As a third such example, a server 108 may not actively monitor the performance capability 202 but may receive an alert if an apparent processing capacity shortage 116 arises (e.g., a message from a downstream server 108 of a reduced delivery of the completed workload 104).

[0071] As a fourth variation of this second aspect, a server 108 may and compare such performance capabilities 202 with the performance guarantees 114 of the workload 104 in various ways. As a first such example, the server 108 may compare an instantaneous measurement of the performance capability 202 with an instantaneous performance guarantee 114, such as a current data transfer rate compared with a minimum acceptable data transfer rate, and/or periodic measurements, such as a number of completed tasks 110 over a workload 104 in a given period vs. a quota of completed tasks 110. As a second such example, the server 108 may compare a trend in the performance capability 202, e.g., detecting a gradual reduction of processing capacity over time that, while currently satisfying the performance guarantee 114, may indicate an imminent or eventual risk 204 of failing the performance guarantee 114, such as a gradually diminishing rate of completed tasks 110. As a third such example, a workload 104 may be associated with a set of at least two performance guarantees 114 for at least two performance capabilities 202 and a priority order of the performance guarantees 114 (e.g., a first priority of a maximum latency of processing individual tasks 110 over the workload 104 at a typical rate of 10 milliseconds, but in the event of an ephemeral failure of the first performance guarantee 114, a second priority of a maximum throughput of processing tasks 110 of the workload 104 within a given period, such as at least 100 tasks completed per second). Such prioritization may enable the performance guarantees 114 to be specified in a layered or more nuanced manner. The server 108 may compare the respective performance capabilities 202 of the server 202 according to the priority order to evaluate the risk 204 of failing the collection of performance guarantees 114 for the workload 114.

[0072] As a fifth variation of this third aspect, a performance capability alert 206 may be relayed from a downstream server 108 to an upstream server 108 in a variety of ways. As a first such example, the performance capability alert 206 may comprise a message initiated by the downstream server 208 and transmitted to the upstream server 108 in response to the identification of a risk 204 of failing the performance guarantee 114. The message may be delivered in-band (e.g., as part of an ordinary communication stream) or out-of-band (e.g., using a separate and dedicated communication channel). As a second such example, the performance capability alert 206 may comprise a performance metric that is continuously and/or periodically reported by the downstream server 108 to the upstream server 108 (e.g., an instantaneous measurement of processing capacity), where the upstream server 108 may construe a fluctuation of the metric as a performance capability alert 206 (e.g., the downstream server 108 may periodically report its latency in completing the task 110 over the workload 104, and the metric may reveal an excessive latency that is approaching a maximum latency specified by the performance guarantee 114). As a third such example, the performance capability alert 206 may comprise part of a data structure shared by the downstream server 108 and the upstream server 108, such as a flag of a status field or a queue count of a workload queue provided at the interface between the downstream server 108 and the upstream server 108. As a fourth such example, the performance capability alert 206 may comprise a function of the upstream server 108 that is invoked by the downstream server 108, such as an API call, a remote procedure call, a delegate function, or an interrupt that the downstream server 108 initiates on the upstream server 108. Many such techniques may be utilized to compare the performance capability 202 to the performance guarantee 14 to identify a risk 204 of failing the performance guarantee 114 of the workload 104 in accordance with the techniques presented herein.

[0073] E3. Task Rate-Limiting

[0074] A third aspect that may vary among embodiments of the presented techniques involves the manner of applying a rate limit 208 to a task 110 over the workload 104 in accordance with the techniques presented herein. FIG. 7 is an illustration of a set 700 of example scenarios featuring various techniques for rate-limiting a task 110 applied to a workload 104 of a server 302.

[0075] As a first variation of this third aspect, illustrated in a first example scenario 710, a server 302 may rate limit a task 110, responsive to receiving a performance capability alert 206 from a downstream server 108, by reducing the performance capabilities 202 of the server 108. As a first example, the server 208 may reduce a processor speed 702 of a processor 304, such as reducing the clock speed or core count that is applied to perform the task 110 over the workload 104. As a second example, the server 302 may reduce a thread priority of the task 110, such that a multiprocessing processor 304 performs increments of the task 110 less frequently, or even suspends the task 110 temporarily if other tasks 110 are of higher priority. Other types of performance capabilities 202 that may be reduced for the workload 104 include volatile or nonvolatile memory allocation; network bandwidth; and/or access to a peripheral device such as a rendering pipeline. In some scenarios, the server 302 may rate limit the task 110, relative to a severity of the performance capability alert 206, such as the degree of constraint on the network capacity of the downstream server 108 or the pending volume of unprocessed work that the downstream server 108 has to work through to alleviate the performance capability alert 206.

[0076] As a second variation of this third aspect, illustrated in a second example scenario 712, a server 302 may rate limit a task 110, responsive to receiving a performance capability alert 206 from a downstream server 108, by temporarily refusing to accept the workload 104 from an upstream server 108, e.g., by initiating a processing jam 118. The processing jam 118 may be initiated in increments, such that the upstream server 108 is only capable of sending batches of the workload 104 to the server 302 in intervals

that are interspersed by a cessation of the workload **104** arriving at the server **302**. Alternatively or additionally, the server **302** may reduce an acceptance rate of the workload **104** from the upstream server **108**; e.g., the upstream server **108** may utilize an output queue of workload **104** to deliver to the server **302**, and the server **302** may only check the output queue at an interval, or at a reduced interval, thereby slowing the rate at which the server **302** accepts workload **104** from the upstream server **108** and delivers the workload **104** to the downstream server **108**.

[0077] As a third variation of this third aspect, illustrated in a third example scenario **714**, a server **108** may rate limit a task **110**, responsive to receiving a performance capability alert **206** from a downstream server **108**, by utilizing one or more queues that slow the intake and/or delivery of the workload **104** to the downstream server **108**. As a first such example, the server **302** may implement an input queue **704** that enqueues the task **110** for the workload **104** for a delay period, and withdraw the task **110** from the input queue to perform the task **110** on the workload **104** only after the delay period. As a second such example, the server **302** may implement an output queue **706** with a delivery delay that slows the rate at which processed work is delivered to the downstream server **108**.

[0078] As a fourth variation of this third aspect, a server **302** may rate limit the task **110** over the workload **104** only within the performance guarantee **114** of the workload **104**. For example, the performance guarantee **114** may comprise a maximum 10-millisecond latency of processing the workload **104** through the process path **112**, and a particular server **302** may be permitted to expend up to 2.5 milliseconds per task **110** while the task **110** remains in conformity with the performance guarantee **114**. If the server **302** typically performs the task **110** in 0.7 milliseconds, the server **302** may rate limit the task **110** for up to or close to an additional 1.8 milliseconds to reduce the rate at which the workload **104** is delivered to the downstream server **108**. If further rate-limiting is to be applied, instead of introducing a new risk **204** of failing the performance guarantee **114**, the server **302** may refrain from further rate-limiting the task **110**. Instead, as shown in the fourth example scenario **716** of FIG. **4**, the server **302** may propagate the performance capability alert **206** to an upstream server **108**. Additionally, if the server **302** comprises a first server in the process path **112** that is assigned the task **110** of intake of new workload **708** from one or more clients **102**, the server **302** may rate limit the workload by reducing an intake rate of the new workload **708** to the entire process path **112**. That is, the server **302** may only agree to accept a diminished volume of the new workload **708** for which the performance guarantee **114** is assigned. Alternatively or additionally, the first server **302** may apply the rate limit **208** to the performance guarantee **114** in an offer **706** provided to the client **102** to extend the new workload **708**, such as extending a maximum latency of the performance guarantee from 10 milliseconds to 20 milliseconds. In this manner, the server **302** may adapt the commitment offered by the server set **302** toward a performance guarantee **114** that the process path **12**, including the server **108** afflicted by a processing capacity shortage **116**, is currently able to guarantee. Many such techniques may be utilized to rate limit the task **110** of a server **108** in response to a performance capability alert **206** in accordance with the techniques presented herein.

[0079] E4. Process Path Adaptation

[0080] A fourth aspect that may vary among embodiments of the techniques presented herein involves adjustment of the process path **112** to adapt to a processing capacity shortage **116**. In some scenarios, rate-limiting the tasks **110** of upstream servers **108** may be adequate to resolve a processing capacity shortage **116**. However, in other scenarios, the processing capacity shortage **116** may be severe, prolonged, and/or of indefinite duration, such that in addition to rate-limiting a task **110** of an upstream server **108**, the server set **106** may implement more significant steps to maintain the satisfaction of the performance guarantee **114**. FIG. **8** is an illustration of a set **800** of example scenarios that illustrate some of the variations of this fifth aspect.

[0081] As a first variation of this fourth aspect, illustrated in a first example scenario **808**, a server **302** may respond to a performance capability alert **206** of a downstream server **108** by redirecting the process path **112** through the server set **102** to provide a substitute server **802** in lieu of the server **108** exhibiting the performance capacity shortage **116**. The substitute server **802** may be previously allocated and allocated and ready for designation as a substitute server **802**, or may be newly provisioned for **802** and inserted into the process path **112**. Alternatively, the substitute server **802** may already exist in the process path **112** of the workload **104** or in another process path **112** of the server set **104**, and the task **110** performed by the server **108** may be transferred to the substitute server **802** along with the workload **104**.

[0082] As a second variation of this fourth aspect, a server **302** may respond to a performance capability alert **206** by expanding a computational resource set of the server **108** exhibiting the processing capacity shortage **116**. As a first such example, the server **108** may comprise a virtual machine, and the processing resources allocated to the virtual machine may be increased (e.g., raising a thread priority and/or processing core usage of the virtual machine). As a second such example, illustrated in a second example scenario **810**, the server **108** exhibiting the processing capacity shortage **116** may be supplemented by the addition of an auxiliary server **804** that expands the processing capacity of the server **108**. For example, the workload **104** may be shared between the server **108** and the auxiliary server **804** until the processing capacity shortage **116** of the server **108** is alleviated.

[0083] As a third variation of this fourth aspect, a server **108** exhibiting a processing capacity shortage **116** may experience a processing capacity shortage **116** that risks failing a performance guarantee **114** of a first workload **104**, but that presents lower or no risk of failing performance guarantees **114** of other workloads **104** of the server **108**. The server **108** may therefor prioritize the processing of the first workload **104** over the other workloads **104** alleviate the processing capacity shortage **106**. As a first such example, illustrated in a third example scenario **812**, the server **108** may adjust by reducing a process priority **806** of another workload **104** that the server **108** processes, e.g., a workload **104** that involves no performance guarantee **114**, or may involve a second performance guarantee **114** that is amply satisfied (e.g., a dependency upon a different type of performance capability of the server **108**, such as a CPU-bound workload as compared with a network-bound workload). The relative adjustment of the process priorities **806** may enable the server **108** to work through a backlog and resolve the processing capacity shortage **116**. As a second such

example, where the server **108** processes a third task **110** for a third workload **104** according to a third process path **112**, the server **108** may redirect the second process path for the third workload **110** through a substitute server **802**. The server **108** may therefore reserve a greater proportion of computational resources to address the processing capacity shortage **116**.

[0084] As a fourth variation of this fourth aspect, a server **108** that implements rate-limiting of a task **110** in order to alleviate a processing capacity shortage **116** of a downstream server **108** may curtail or end the rate-limiting of the task **110** based upon an alleviation of the performance capability shortage **116** of the downstream sever **108**. As a first such example, a downstream server **108** initiating the performance capability alert **206** may send a notification to an upstream server **302** applying the rate-limiting to indicate an abatement of the processing capacity shortage **116**. As a second such example, an upstream server **302** applying rate-limiting to a task **110** may detect an abatement of the processing capacity shortage **116**, e.g., as a depletion of an output queue of workloads **104** to deliver to the downstream server **108**. As a third such example, the upstream server **302** may apply the rate-limiting only for a set interval, such as one second, and may then remove the rate-limiting, such that a persistence of the processing capacity shortage **116** at the downstream server **108** may result in a second performance capability alert **206** and a reapplication of the rate limit to the process **110**. In some scenarios, the reapplication may occur at an increasing interval (e.g., first one second, then two seconds, etc.) to reduce an inefficiency of the transmission and receipt of multiple performance capability alerts **206**, which may reduce the ability of the downstream server **108** to alleviate the processing capacity shortage **116**.

[0085] As a fifth variation of this fourth aspect, the adjustments of the process paths **112** may be requested and/or implemented by the server **108** experiencing the processing capacity shortage **116**. As another example, the adjustments of the processing paths **112** may be requested and/or implemented by the upstream server **302**, e.g., upon determining that the rate-limiting of the task **110** by the upstream server **302** is insufficient to resolve a processing capacity shortage **116** that is prolonged, indefinite, overly frequent, and/or unresolvable by rate-limiting. As yet another example, the adjustments of the processing paths **112** may be implemented at the request of an automated network monitor or network administrator. Many such techniques may be utilized to provide further adaptations of the server set **106**, in conjunction with the rate-limiting of the task **110** by the upstream server **302**, in accordance with the techniques presented herein.

F. Computing Environment

[0086] FIG. **9** and the following discussion provide a brief, general description of a suitable computing environment to implement embodiments of one or more of the provisions set forth herein. The operating environment of FIG. **9** is only one example of a suitable operating environment and is not intended to suggest any limitation as to the scope of use or functionality of the operating environment. Example computing devices include, but are not limited to, personal computers, server computers, hand-held or laptop devices, mobile devices (such as mobile phones, Personal Digital Assistants (PDAs), media players, and the like), multiprocessor systems, consumer electronics, mini computers, mainframe computers, distributed computing environments that include any of the above systems or devices, and the like.

[0087] Although not required, embodiments are described in the general context of "computer readable instructions" being executed by one or more computing devices. Computer readable instructions may be distributed via computer readable media (discussed below). Computer readable instructions may be implemented as program modules, such as functions, objects, Application Programming Interfaces (APIs), data structures, and the like, that perform particular tasks or implement particular abstract data types. Typically, the functionality of the computer readable instructions may be combined or distributed as desired in various environments.

[0088] FIG. **9** illustrates an example of a system comprising a computing device **902** configured to implement one or more embodiments provided herein. In one configuration, computing device **902** includes at least one processing unit **906** and memory **908**. Depending on the exact configuration and type of computing device, memory **908** may be volatile (such as RAM, for example), non-volatile (such as ROM, flash memory, etc., for example) or some combination of the two. This configuration is illustrated in FIG. **9** by dashed line **904**.

[0089] In other embodiments, device **902** may include additional features and/or functionality. For example, device **902** may also include additional storage (e.g., removable and/or non-removable) including, but not limited to, magnetic storage, optical storage, and the like. Such additional storage is illustrated in FIG. **9** by storage **910**. In one embodiment, computer readable instructions to implement one or more embodiments provided herein may be in storage **910**. Storage **910** may also store other computer readable instructions to implement an operating system, an application program, and the like. Computer readable instructions may be loaded in memory **908** for execution by processing unit **906**, for example.

[0090] The term "computer readable media" as used herein includes computer storage media. Computer storage media includes volatile and nonvolatile, removable and non-removable media implemented in any method or technology for storage of information such as computer readable instructions or other data. Memory **908** and storage **910** are examples of computer storage media. Computer storage media includes, but is not limited to, RAM, ROM, EEPROM, flash memory or other memory technology, CD-ROM, Digital Versatile Disks (DVDs) or other optical storage, magnetic cassettes, magnetic tape, magnetic disk storage or other magnetic storage devices, or any other medium which can be used to store the desired information and which can be accessed by device **902**. Any such computer storage media may be part of device **902**.

[0091] Device **902** may also include communication connection(s) **916** that allows device **902** to communicate with other devices. Communication connection(s) **916** may include, but is not limited to, a modem, a Network Interface Card (NIC), an integrated network interface, a radio frequency transmitter/receiver, an infrared port, a USB connection, or other interfaces for connecting computing device **902** to other computing devices. Communication connection (s) **916** may include a wired connection or a wireless connection. Communication connection(s) **916** may transmit and/or receive communication media.

[0092] The term "computer readable media" may include communication media. Communication media typically embodies computer readable instructions or other data in a "modulated data signal" such as a carrier wave or other transport mechanism and includes any information delivery media. The term "modulated data signal" may include a signal that has one or more of its characteristics set or changed in such a manner as to encode information in the signal.

[0093] Device **902** may include input device(s) **914** such as keyboard, mouse, pen, voice input device, touch input device, infrared cameras, video input devices, and/or any other input device. Output device(s) **912** such as one or more displays, speakers, printers, and/or any other output device may also be included in device **902**. Input device(s) **914** and output device(s) **912** may be connected to device **902** via a wired connection, wireless connection, or any combination thereof. In one embodiment, an input device or an output device from another computing device may be used as input device(s) **914** or output device(s) **912** for computing device **902**.

[0094] Components of computing device **902** may be connected by various interconnects, such as a bus. Such interconnects may include a Peripheral Component Interconnect (PCI), such as PCI Express, a Universal Serial Bus (USB), Firewire (IEEE 1394), an optical bus structure, and the like. In another embodiment, components of computing device **902** may be interconnected by a network. For example, memory **908** may be comprised of multiple physical memory units located in different physical locations interconnected by a network.

[0095] Those skilled in the art will realize that storage devices utilized to store computer readable instructions may be distributed across a network. For example, a computing device **920** accessible via network **918** may store computer readable instructions to implement one or more embodiments provided herein. Computing device **902** may access computing device **920** and download a part or all of the computer readable instructions for execution. Alternatively, computing device **902** may download pieces of the computer readable instructions, as needed, or some instructions may be executed at computing device **902** and some at computing device **920**.

G. Usage of Terms

[0096] Although the subject matter has been described in language specific to structural features and/or methodological acts, it is to be understood that the subject matter defined in the appended claims is not necessarily limited to the specific features or acts described above. Rather, the specific features and acts described above are disclosed as example forms of implementing the claims.

[0097] As used in this application, the terms "component," "module," "system", "interface", and the like are generally intended to refer to a computer-related entity, either hardware, a combination of hardware and software, software, or software in execution. One or more components may be localized on one computer and/or distributed between two or more computers.

[0098] Furthermore, the claimed subject matter may be implemented as a method, apparatus, or article of manufacture using standard programming and/or engineering techniques to produce software, firmware, hardware, or any combination thereof to control a computer to implement the disclosed subject matter. The term "article of manufacture" as used herein is intended to encompass a computer program accessible from any computer-readable device, carrier, or media. Of course, those skilled in the art will recognize many modifications may be made to this configuration without departing from the scope or spirit of the claimed subject matter.

[0099] Various operations of embodiments are provided herein. In one embodiment, one or more of the operations described may constitute computer readable instructions stored on one or more computer readable media, which if executed by a computing device, will cause the computing device to perform the operations described. The order in which some or all of the operations are described should not be construed as to imply that these operations are necessarily order dependent. Alternative ordering will be appreciated by one skilled in the art having the benefit of this description. Further, it will be understood that not all operations are necessarily present in each embodiment provided herein.

[0100] Any aspect or design described herein as an "example" is not necessarily to be construed as advantageous over other aspects or designs. Rather, use of the word "example" is intended to present one possible aspect and/or implementation that may pertain to the techniques presented herein. Such examples are not necessary for such techniques or intended to be limiting. Various embodiments of such techniques may include such an example, alone or in combination with other features, and/or may vary and/or omit the illustrated example.

[0101] As used in this application, the term "or" is intended to mean an inclusive "or" rather than an exclusive "or". That is, unless specified otherwise, or clear from context, "X employs A or B" is intended to mean any of the natural inclusive permutations. That is, if X employs A; X employs B; or X employs both A and B, then "X employs A or B" is satisfied under any of the foregoing instances. In addition, the articles "a" and "an" as used in this application and the appended claims may generally be construed to mean "one or more" unless specified otherwise or clear from context to be directed to a singular form.

[0102] Also, although the disclosure has been shown and described with respect to one or more implementations, equivalent alterations and modifications will occur to others skilled in the art based upon a reading and understanding of this specification and the annexed drawings. The disclosure includes all such modifications and alterations and is limited only by the scope of the following claims. In particular regard to the various functions performed by the above described components (e.g., elements, resources, etc.), the terms used to describe such components are intended to correspond, unless otherwise indicated, to any component which performs the specified function of the described component (e.g., that is functionally equivalent), even though not structurally equivalent to the disclosed structure which performs the function in the herein illustrated example implementations of the disclosure. In addition, while a particular feature of the disclosure may have been disclosed with respect to only one of several implementations, such feature may be combined with one or more other features of the other implementations as may be desired and advantageous for any given or particular application. Furthermore, to the extent that the terms "includes", "having", "has", "with", or variants thereof are used in either the

detailed description or the claims, such terms are intended to be inclusive in a manner similar to the term "comprising."

What is claimed is:

1. A server of a server set that performs workloads according to a performance guarantee, the server comprising:

a processor; and

a memory storing instructions that, when executed by the processor, cause the server to:

perform a task of the workload according to a performance guarantee, wherein the workload is processed through the server set according to a process path;

receive a performance capability alert from the downstream server, wherein the performance capability alert indicates that a computational load of a downstream server risks failing the performance guarantee for the workload;

rate-limit the task performed on the workload to reduce the computational load of the downstream server; and

after completing the task, deliver the workload to the downstream server of the process path.

2. The server of claim **1**, wherein rate-limiting the task for the workload further comprises: refusing to accept the workload from an upstream server.

3. The server of claim **1**, wherein rate-limiting the task for the workload further comprises: slowing an acceptance rate of the workload from an upstream server.

4. The server of claim **1**, wherein rate-limiting the task for the workload further comprises:

enqueuing the task for the workload in an input queue for a delay period; and

withdrawing the task from the input queue to perform the task of the workload after the delay period.

5. The server of claim **1**, wherein rate-limiting the task for the workload further comprises: rate-limiting a processing rate of the task within the performance guarantee.

6. The server of claim **1**, further comprising: responsive to receiving the performance capability alert from the downstream server, propagating the performance capability alert to an upstream server.

7. The server of claim **1**, wherein:

the server further comprises an intake server that accepts the workload from a client into the process path; and

rate-limiting the task for the workload further comprises: refusing to accept the workload into the process path.

8. The server of claim **1**, wherein the performance guarantee involves a performance capability selected from a performance capability set comprising:

a processing latency of the process path for the workload;

a processing throughput of the process path for the workload;

a processing consistency of the process path for the workload; and

a processing replication of the workload within the server set.

9. A method of configuring a server of a server set to participate in workloads, the method comprising:

executing, by a processor of the server, instructions that cause the server to:

receive a workload from an upstream server of a process path, wherein the workload is associated with a performance guarantee;

perform a task on the workload;

identify a performance capability of the server;

compare the performance capability with the performance guarantee of the workload;

responsive to determining that the performance capability risks failing the performance guarantee, transmit a performance capability alert to the upstream server; and

responsive to receiving a performance capability alert from a downstream server of the process path, rate-limit the task performed on the workload to reduce the computational load of the downstream server.

10. The method of claim **9**, wherein identifying the performance capability of the server further comprises: predicting the performance capability of the server performing the workload.

11. The method of claim **9**, wherein identifying the performance capability of the server further comprises: measuring the performance capability of the server while performing the workload.

12. The method of claim **9**, wherein:

the performance guarantee further comprises a set of performance guarantees for at least two performance capabilities and a priority order; and

comparing the performance capability with the performance guarantee further comprises: comparing the performance capabilities according to the priority order.

13. The method of claim **9**, wherein executing the instructions further causes the server to, responsive to an alleviation of the performance capability alert, reduce the rate-limiting of the task for the workload.

14. A method of configuring a server set to perform a workload according to a performance guarantee, the method comprising:

configuring a server of a process path of the server set to process a workload by:

performing a task on the workload according to the performance guarantee;

receiving a performance capability alert from a downstream server of the process path, wherein the performance capability alert indicates that a computational load of the downstream server risks failing the performance guarantee for the workload;

rate-limiting the task of the server to reduce the workload delivered to the downstream server; and

after performing the task, delivering the workload to the downstream server.

15. The method of claim **14**, further comprising: responsive to the performance capability alert, redirecting the process path through the server set to provide a substitute server for the downstream server.

16. The method of claim **14**, wherein:

the downstream server is further processing a second task for a second workload according to a second process path; and

the method further comprises: redirecting a second process path for the second workload to provide a substitute server for the downstream server.

17. The method of claim **14**, further comprising, responsive to the performance capability alert:

identifying a second workload of the process path for which the server set is satisfying a second performance guarantee; and

increasing a processing priority of the workload relative to the second workload.

18. The method of claim **14**, wherein:

the downstream server is processing the workload using a computational resource set; and

the method further comprises: responsive to the performance capability alert, expanding the computational resource set of the downstream server.

19. The method of claim **18**, wherein expanding the computational resource set of the downstream server further comprises:

selecting an auxiliary server to supplement the downstream server; and

sharing the workload between the downstream server and the auxiliary server.

20. The method of claim **14**, further comprising: responsive to the performance capability alert, transferring a computational task from the downstream server to a substitute server.

* * * * *