November 2022

A SYSTEM AND METHOD FOR API BASED FILE PROCESSING

Prakhar Gangwar
visa

Shivam Mohan
Visa

Alok Roy
Visa

Neeraj Neeraj
Visa

Satyam Raj
Visa

See next page for additional authors

Inventor(s)
Prakhar Gangwar, Shivam Mohan, Alok Roy, Neeraj Neeraj, Satyam Raj, Ravi Shanker Kumar Sinha, Shankar RC, Kandikatla Varshini, Prajna Shetty, and Prithwish Banerjee

A SYSTEM AND METHOD FOR API BASED FILE PROCESSING

VISA

Prakhar Gangwar
Shivam Mohan
Alok Roy
Neeraj Surana
Satyam Raj
Ravi Shanker Kumar Sinha
Shankar RC
Kandikatla Varshini
Prajna Shetty
Prithwish Banerjee

FIELD OF INVENTION:

The present invention generally relates to the field of networking and API based file processing systems. More particularly, but not specifically, the present invention relates to a system and method for processing Application Programming Interface (API) transitions.

BACKGROUND:

These days systems are moving towards Application Programming Interface (API) based solutions since API enables software services and products to communicate with each other for large areas and/or longer time periods.

Generally, APIs exist as an intermediary layer between an application and the webserver, that processes data transfer between system. Further, APIs extract data from multiple APIs and / or multiple backend systems, in which the resulting activities can link several microservices and other APIs. Consequently, an API call can invoke one or more serial and parallel activities. So, the linking process of several microservices and other APIs grows eventually as each API call further prompts other API calls. Therefore, the complexity of handling each API transition multiplies exponentially and poses many challenges.

To handle these challenges efficiently, there were introduced batch processing to schedule groups of loads (jobs) to be processed at the same time with or without human intervention. Batch processing is typically used to process large amounts of data that consolidates all the information from other batch processing systems and or with new sources of data. Each job from a batch involves a precise volume or capacity of the processing machine for a precise time. Further, the batch processing time is generally the longest processing times of all the jobs in the corresponding batch. To increase the efficiency of batch processing the makespan must be minimized. Makespan is the length of time that elapses from start of the job to the end.

Therefore, there is need for an API based solution to minimize the makespan of batch processing for API transitions which involves a large amount of data.

SUMMARY:

Various embodiments of the present invention provide a system and method for defining, implementing, and / or executing batch processing of API transaction services and products.

Such API transitions access applications and services via the proposed system which processes the API traffic in batches to solve the technical problems associated with the existing API file processing environment. Thus, it is possible to envisage a tool which sits between API solutions and client applications therefore reduces the long lag time to process the API traffic by incorporating the present invention. Furthermore, the system provides capability to call API for every line and / or bunch of lines and store output of all the results received in the output file.

According to a further aspect of the present invention, there is provided a computer implemented system for processing the input files with large amount of data associated with API traffic in batches. The system comprising, one or more processors and one or more memory communicatively coupled with the one or more processors. The one or more processors of the said system may be configured to receive a plurality of file processing requests associated with API traffic from one or more clients. The system further configured to batch, one or more jobs associated with the plurality of file processing request for the API traffic. Further, the processor within the system is configured to pick a request from one or more jobs from the batch associated with the API traffic.

In an embodiment the processor is configured to initiate the process of executing the API call associated with the API transitions. After picking a job from the batch, the processor is configured to read the input file from the input storage associated with the job and further splits the input file into plurality of chunks and invoke the API call associated with the corresponding chunk from the plurality of APIs. The responses received from API transitions invoked by the chunks related to API services and products being stored in chunks. The processor within the system is configured to consolidate the plurality of responses stored in the chunks and write the response to an output file.

Accordingly, in an embodiment, the present invention discloses a method for processing the input files with large amount of data associated with API traffic in batches comprising, receiving receive a plurality of file processing requests associated with API traffic from one or more clients by a processor. The method further comprises the technique of queueing / batching the one or more jobs associated with the plurality of file processing request for the API traffic to form batches. Further, the method comprises picking a request from one or more jobs from the batch associated with the API traffic.

In an embodiment the method further initiates the process of executing the API call associated with the API transitions. After picking a job from the batch, the method performed by the processor reads the input file from the input storage associated with the job and further splits the input file into plurality of chunks based on the configuration parameters. Thus, the method invokes the API call associated with the corresponding chunk from the plurality of APIs. The responses received from API transitions invoked by the chunks related to API services and products being stored in chunks. Further, the method comprises the process of consolidating the plurality of responses stored in the chunks and writing the response to an output file.

The foregoing summary is illustrative only and is not intended to be in any way limiting. In addition to the illustrative aspects, embodiments, and features described above, further aspects, embodiments, and features will become apparent by reference to the drawings and the following detailed description. For a better understanding of exemplary embodiments of the present invention, together with other and further features and advantages thereof, reference is made to the following description, taken in conjunction with the accompanying drawings.

BRIEF DESCRIPTION OF THE ACCOMPANYING DRAWINGS

The accompanying drawings, which are incorporated in and constitute a part of this disclosure, illustrate exemplary embodiments and together with the description, serve to explain the disclosed principles. In the figures, the left-most digit(s) of a reference number identifies the figure in which the reference number first appears. The embodiments of the disclosure itself, however, as well as a preferred mode of use, further objectives, and advantages thereof, will best be understood by reference to the following detailed description of an illustrative embodiment. Some embodiments of system and/or method in accordance with embodiments of the present subject matter are now described below, by way of example, and with reference to the accompanying figures.

Figure 1 is an illustration of one of the existing system schematic of an exemplary batch file processing system.

Figure 2 depicts a schematic diagram of an exemplary for defining, implementing, and / or executing batch processing of API transaction services and products in accordance with the present invention.

Figure 3 depicts a schematic block diagram of an exemplary API based file processing system for defining, implementing, and / or executing batch processing of API transaction services and products in accordance with the present invention.

Figure 4 depicts a schematic block diagram of an exemplary transition and hybrid support by migration from legacy file-based communication to hybrid model communication in accordance with the present invention.

Figure 5 depicts a flow chart of an exemplary API based file processing system for defining, implementing, and / or executing batch processing of API transaction services and products in accordance with the present invention.

Figure 6 depicts a schematic block diagram of an exemplary computer system for implementing various embodiments consistent with the present disclosure.

The figures depict embodiments of the disclosure for purpose of illustration only. One skilled in the art will readily recognize from the following description that alternative embodiments of the structures and methods illustrated herein may be employed without departing from the principles of the disclosure herein.

It should be appreciated by those skilled in the art that any block diagrams herein represent conceptual views of illustrative systems embodying the principles of the present subject matter. Similarly, it will be appreciated that any flow charts, flow diagrams, state transition diagrams, pseudo code, and the like represent various processes which may be substantially represented in computer readable medium and executed by a computer or processor, whether or not such computer or processor is explicitly shown.

DESCRIPTION

In the present document, the word "exemplary" is used herein to mean "serving as an example, instance, or illustration." Any embodiment or implementation of the present subject matter described herein as "exemplary" is not necessarily to be construed as preferred or advantageous over other embodiments.

While the disclosure is susceptible to various modifications and alternative forms, specific embodiment thereof has been shown by way of example in the drawings and will be described in detail below. It should be understood, however that it is not intended to limit the

disclosure to the particular forms disclosed, but on the contrary, the disclosure is to cover all modifications, equivalents, and alternative falling within the scope of the disclosure.

The terms “comprises”, “comprising”, or any other variations thereof, are intended to cover a non-exclusive inclusion, such that a setup, device, or method that comprises a list of components or steps does not include only those components or steps but may include other components or steps not expressly listed or inherent to such setup or device or method. In other words, one or more elements in a system or apparatus proceeded by “comprises… a” does not, without more constraints, preclude the existence of other elements or additional elements in the system or apparatus.

In the following detailed description of the embodiments of the disclosure, reference is made to the accompanying drawings that form a part hereof, and in which are shown by way of illustration specific embodiments in which the description may be practiced. These embodiments are described in sufficient detail to enable those skilled in the art to practice the disclosure, and it is to be understood that other embodiments may be utilized and that changes may be made without departing from the scope of the present disclosure. The following description is, therefore, not to be taken in a limiting sense.

Figure 1 depicts an exemplary illustration of one of the existing system schematics of an exemplary batch file processing system. The environment comprises one or more input file storage 101 (101.a, 101.b, 101.c), a one or more output file storage 102 (102.a, 102.b, 102.c), a one or more process flow associated with API call request 103 (103.a, 104, 103.c) and one or more process implementation and job invocation method and different I/O method 105 (105.a, 105.b, 105.c). The one or more Input / Output file storage system are associated with one or more client applications which are demanding access to its corresponding APIs.

In the existing batch file processing systems, for each process flow being requested by one or more client applications, there is a subsequent defined implementation and job invocation method. So, these process flows are limited to each process requirement associated with the Process request being invoked. Therefore, there exists a long lag time in transitioning these API solutions from batch solution. Thus, in the existing network architecture each API request /message received from a one or more client select an appropriate process implementation and job invocation method and different I/O method which is limited to that particular API. Therefore there is a need for a system and method for processing these API requests received from one or more client applications with minimized lag time or makespan.

Figure 2 depicts a schematic diagram of an exemplary for defining, implementing, and / or executing batch processing of API transaction services and products in accordance with the present invention. The environment comprises a computer system (202) configured to define, implement, and / or execute batch file processing of API transaction services and products in accordance with the present invention. The exemplary environment comprises one or more client applications 201 (201.a, 201.b, 201.c) with one or more request to access one or more API based services and products (203.a, 203.b, 203.c, 203.d) and the system 202 disposed as network intermediate between one or more client applications 201 (201.a, 201.b, 201.c) and one or more API based services and products (203.a, 203.b, 203.c, 203.d). Based on the information received API requests from one or more clients, the system 202 processes the API traffic in batches thus minimizes the time taken to complete the API transition invoked.

The system 202 is not a replacement of API solution but it will simplify the transitions from batch to API solution. The system 202 is further configured to identify the corresponding API based services and products (203.a, 203.b, 203.c, 203.d) based on one or more parameters regarding client applications and API based solutions then invokes the corresponding API call. Then the system processes the API based file in chunks and stores the responses received from API based solutions. Furthermore, the system consolidates all the responses received for the input file in chunks and writes the file with the consolidated output.

Figure 3 depicts a schematic block diagram (300) of an exemplary API based file processing system for defining, implementing, and / or executing batch processing of API transaction services and products in accordance with the present invention. As illustrated in Fig.3, the API based file processing system includes one or more client applications 301 (301.a, 301.b, 301.c) and I/P file storage (305), O/P file storage (306), Requested job queue (302), Processing request module (304), and API based solutions.

The system configured to initiate the process (304) for forwarding the received client Application API request by picking a job from the batch (302), reading the input file from the input file storage (305) and upon the input specification chunking and parsing the file to identify corresponding API solution / API call. By invoking the corresponding API call, the system is configured to receive the response from the API module and store the results in the corresponding chunk. After processing all the chunks made from the input file, consolidate the responses to write in the O/P file storage.

However, various versions of the modules involved for processing the API calls may be equally configured or adapted to implement embodiments for various other types of file processing systems. Therefore, the following examples are not intended to be limited as to various types or formats of API based services and products or and or client applications.

In yet another embodiment the system configuration parameters comprise Input specifications, API specifications, Output specifications, and Processing specifications. The input specifications may include, but not limited to file name, location, file format, file parser type, and fields to be extracted from input for API call. Similarly, the API specifications may include but not limited to endpoint details, and arguments to be passed for each API call. Likewise, the output specifications may include parameters such as, but not limited to output fields specifications, such as but not limited to chunking logic, and processing parameters.

Figure 4 depicts a schematic block diagram of an exemplary transition and hybrid support by migration from legacy file-based communication to hybrid model communication in accordance with the present invention. As illustrated the system enables legacy file-based models (402) to communicate with hybrid model (401) where both file and API communication is supported. Therefore, the present invention provides a benefit of utilizing API based communications and retire file-based communication for applications requiring to migrate.

In hybrid mode, the API-engine will not only act as an intermediary for applications that use file-based communication to consume products that use API based communication (404.a), but also allows the entire ecosystem to work in synergy even if few of the components are still using file-based communication (404.b).

Figure 5 depicts a flow chart of an exemplary API based file processing system for defining, implementing, and / or executing batch processing of API transaction services and products in accordance with the present invention. The file processor enables the client application to call various APIs based on the configuration provided. The system in accordance with the present disclosure, takes all the configuration inputs to process the one or more API request to access the API based solutions.

The configuration parameters may include but not limited to Input specifications, API specifications, Output specifications, and Processing specifications. The input specifications may include, but not limited to file name, location, file format, file parser type, and fields to be extracted from input for API call. Similarly, the API specifications may include but not limited to endpoint details, certificate details / credentials for accessing the APIs and arguments to be

passed for each API call. The processing specifications many include, but not limited to chunking logic to be used, chunk size / micro batch size for calling API, sequential processing / parallel processing, and number of processors / threads to be used for processing. Likewise, the output specifications may include parameters such as, but not limited to output fields specifications, such as but not limited to output file format, output parameters extraction, and errored records output file.

At step 501, the method receives a plurality of file processing requests associated with API traffic from one or more clients by a processor. One or more files can be passed in one processing flow. Further, the files can be processed in sequence or in parallel as required and specified in processing specifications. Furthermore, the system, in accordance with the present disclosure, supports multiple formats and can be extended to support new input files as required as well as can be in different formats. The file formats may include but are not limited to text, JSON, YAML, XML, etc.

At step 503, the method further does queueing / batching the one or more jobs associated with the plurality of file processing request for the API traffic to form batches. Further, the method performs picking a request from one or more jobs from the batch associated with the API traffic, at method step 505.

In an embodiment the method further initiates the process of executing the API call associated with the API transitions. After picking a job from the batch, the method performed by the processor reads the input file from the input storage associated with the job and further splits the input file into plurality of chunks based on the configuration parameters at step 507. Thus, the method invokes the API call associated with the corresponding chunk from the plurality of APIs. The responses received from API transitions invoked by the chunks related to API services and products being stored in chunks at step 509. Further, the method performs the process of consolidating the plurality of responses stored in the chunks and writing the response to an output file at step 511.

Computing System

Figure 6 illustrates a block diagram of an exemplary computer system 600 for implementing embodiments consistent with the present disclosure. In an embodiment, the computer system 600 is used to implement the evaluating system for evaluating development of the at least one plant. The computer system 600 may include a central processing unit (“CPU” or “processor”)

502. The processor 602 may include at least one data processor for executing processes in Virtual Storage Area Network. The processor 602 may include specialized processing units such as, integrated system (bus) controllers, memory management control units, floating point units, graphics processing units, digital signal processing units, etc.

The processor 602 may be disposed in communication with one or more input/output (I/O) devices 609 and 610 via I/O interface 601. The I/O interface 601 may employ communication protocols/methods such as, without limitation, audio, analog, digital, monaural, RCA, stereo, IEEE-1394, serial bus, universal serial bus (USB), infrared, PS/2, BNC, coaxial, component, composite, digital visual interface (DVI), high-definition multimedia interface (HDMI), radio frequency (RF) antennas, S-Video, VGA, IEEE 802.n /b/g/n/x, Bluetooth, cellular (e.g., code-division multiple access (CDMA), high-speed packet access (HSPA+), global system for mobile communications (GSM), long-term evolution (LTE), WiMax, or the like), etc.

Using the I/O interface 601, the computer system 600 may communicate with one or more I/O devices 609 and 610. For example, the input devices 609 may be an antenna, keyboard, mouse, joystick, (infrared) remote control, camera, card reader, fax machine, dongle, biometric reader, microphone, touch screen, touchpad, trackball, stylus, scanner, storage device, transceiver, video device/source, etc. The output devices 610 may be a printer, fax machine, video display (e.g., cathode ray tube (CRT), liquid crystal display (LCD), light-emitting diode (LED), plasma, Plasma Display Panel (PDP), Organic light-emitting diode display (OLED) or the like), audio speaker, etc.

The processor 602 may be disposed in communication with a communication network 611 via a network interface 603. The network interface 603 may communicate with the communication network 611. The network interface 603 may employ connection protocols including, without limitation, direct connect, Ethernet (e.g., twisted pair 10/100/1000 Base T), transmission control protocol/internet protocol (TCP/IP), token ring, IEEE 802.11a/b/g/n/x, etc. The communication network 611 may include, without limitation, a direct interconnection, local area network (LAN), wide area network (WAN), wireless network (e.g., using Wireless Application Protocol), the Internet, etc. Using the network interface 503 and the communication network 611, the computer system 600 may communicate with at least one user device 612 via communication network 611 to provide preference-based campaign page. The network interface 603 may employ connection protocols include, but not limited to, direct

connect, Ethernet (e.g., twisted pair 10/100/1000 Base T), transmission control protocol/internet protocol (TCP/IP), token ring, IEEE 802.11a/b/g/n/x, etc.

In an embodiment, the computer system (600) may receive plurality captured of images related to the at least one plant and user first input via application installed in the at least one user device (612) through the communication network (611).

The communication network 611 includes, but is not limited to, a direct interconnection, an e-commerce network, a peer to peer (P2P) network, local area network (LAN), wide area network (WAN), wireless network (e.g., using Wireless Application Protocol), the Internet, Wi-Fi, and such. The first network and the second network may either be a dedicated network or a shared network, which represents an association of the different types of networks that use a variety of protocols, for example, Hypertext Transfer Protocol (HTTP), Transmission Control Protocol/Internet Protocol (TCP/IP), Wireless Application Protocol (WAP), etc., to communicate with each other. Further, the first network and the second network may include a variety of network devices, including routers, bridges, servers, computing devices, storage devices, etc.

In some embodiments, the processor 602 may be disposed in communication with a memory 605 (e.g., RAM, ROM, etc. not shown in **Figure 6**) via a storage interface 604. The storage interface 604 may connect to memory 605 including, without limitation, memory drives, removable disc drives, etc., employing connection protocols such as, serial advanced technology attachment (SATA), Integrated Drive Electronics (IDE), IEEE-1394, Universal Serial Bus (USB), fiber channel, Small Computer Systems Interface (SCSI), etc. The memory drives may further include a drum, magnetic disc drive, magneto-optical drive, optical drive, Redundant Array of Independent Discs (RAID), solid-state memory devices, solid-state drives, etc.

The memory 605 may store a collection of program or database components, including, without limitation, user interface 606, an operating system 607, web browser 608 etc. In some embodiments, computer system 600 may store user/application data, such as, the data, variables, records, etc., as described in this disclosure. Such databases may be implemented as fault-tolerant, relational, scalable, secure databases such as Oracle ® or Sybase®.

The operating system 607 may facilitate resource management and operation of the computer system 600. Examples of operating systems include, without limitation, APPLE

MACINTOSH® OS X, UNIX®, UNIX-like system distributions (E.G., BERKELEY SOFTWARE DISTRIBUTIONTM (BSD), FREEBSDTM, NETBSDTM, OPENBSDTM, etc.), LINUX DISTRIBUTIONSTM (E.G., RED HATTM, UBUNTUTM, KUBUNTUTM, etc.), IBMTM OS/2, MICROSOFTTM WINDOWSTM (XPTM, VISTATM/7/8, 10 etc.), APPLE® IOSTM, GOOGLE® ANDROIDTM, BLACKBERRY® OS, or the like.

In some embodiments, the computer system 600 may implement a web browser 608 stored program component. The web browser 608 may be a hypertext viewing application, such as Microsoft Internet Explorer, Google Chrome, Mozilla Firefox, Apple Safari, etc. Secure web browsing may be provided using Hypertext Transport Protocol Secure (HTTPS), Secure Sockets Layer (SSL), Transport Layer Security (TLS), etc. Web browsers 508 may utilize facilities such as AJAX, DHTML, Adobe Flash, JavaScript, Java, Application Programming Interfaces (APIs), etc. In some embodiments, the computer system 600 may implement a mail server stored program component. The mail server may be an Internet mail server such as Microsoft Exchange, or the like. The mail server may utilize facilities such as ASP, ActiveX, ANSI C++/C#, Microsoft .NET, Common Gateway Interface (CGI) scripts, Java, JavaScript, PERL, PHP, Python, WebObjects, etc. The mail server may utilize communication protocols such as Internet Message Access Protocol (IMAP), Messaging Application Programming Interface (MAPI), Microsoft Exchange, Post Office Protocol (POP), Simple Mail Transfer Protocol (SMTP), or the like. In some embodiments, the computer system 600 may implement a mail client stored program component. The mail client may be a mail viewing application, such as Apple Mail, Microsoft Entourage, Microsoft Outlook, Mozilla Thunderbird, etc.

Furthermore, one or more computer-readable storage media may be utilized in implementing embodiments consistent with the present disclosure. A computer-readable storage medium refers to any type of physical memory on which information or data readable by a processor 602 may be stored. Thus, a computer-readable storage medium may store instructions for execution by one or more processors, including instructions for causing the processor(s) 602 to perform steps or stages consistent with the embodiments described herein. The term “computer-readable medium” should be understood to include tangible items and exclude carrier waves and transient signals, i.e., be non-transitory. Examples include Random Access Memory (RAM), Read-Only Memory (ROM), volatile memory, non-volatile memory, hard drives, Compact Disc (CD) ROMs, DVDs, flash drives, disks, and any other known physical storage media.

The described operations may be implemented as a method, system or article of manufacture using standard programming and/or engineering techniques to produce software, firmware, hardware, or any combination thereof. The described operations may be implemented as code maintained in a “non-transitory computer readable medium”, where a processor 602 may read and execute the code from the computer readable medium. The processor 602 is at least one of a microprocessor and a processor capable of processing and executing the queries. A non-transitory computer readable medium may include media such as magnetic storage medium (e.g., hard disk drives, floppy disks, tape, etc.), optical storage (CD-ROMs, DVDs, optical disks, etc.), volatile and non-volatile memory devices (e.g., EEPROMs, ROMs, PROMs, RAMs, DRAMs, SRAMs, Flash Memory, firmware, programmable logic, etc.), etc. Further, non-transitory computer-readable media may include all computer-readable media except for transitory. The code implementing the described operations may further be implemented in hardware logic (e.g., an integrated circuit chip, Programmable Gate Array (PGA), Application Specific Integrated Circuit (ASIC), etc.).

The illustrated steps are set out to explain the exemplary embodiments shown, and it should be anticipated that ongoing technological development will change the manner in which particular functions are performed. These examples are presented herein for purposes of illustration, and not limitation. Further, the boundaries of the functional building blocks have been arbitrarily defined herein for the convenience of the description. Alternative boundaries can be defined so long as the specified functions and relationships thereof are appropriately performed. Alternatives (including equivalents, extensions, variations, deviations, etc., of those described herein) will be apparent to persons skilled in the relevant art(s) based on the teachings contained herein. Such alternatives fall within the scope and spirit of the disclosed embodiments. Also, the words "comprising," "having," "containing," and "including," and other similar forms are intended to be equivalent in meaning and be open ended in that an item or items following any one of these words is not meant to be an exhaustive listing of such item or items or meant to be limited to only the listed item or items. It must also be noted that as used herein and in the appended claims, the singular forms “a,” “an,” and “the” include plural references unless the context clearly dictates otherwise.

Furthermore, one or more computer-readable storage media may be utilized in implementing embodiments consistent with the present disclosure. A computer readable storage medium refers to any type of physical memory on which information or data readable

by a processor may be stored. Thus, a computer readable storage medium may store instructions for execution by one or more processors, including instructions for causing the processor(s) to perform steps or stages consistent with the embodiments described herein. The term “computer readable medium” should be understood to include tangible items and exclude carrier waves and transient signals, i.e., are non-transitory. Examples include random access memory (RAM), read-only memory (ROM), volatile memory, non-volatile memory, hard drives, CD ROMs, DVDs, flash drives, disks, and any other known physical storage media.

With respect to the use of substantially any plural and/or singular terms herein, those having skill in the art can translate from the plural to the singular and/or from the singular to the plural as is appropriate to the context and/or application. The various singular/plural permutations may be expressly set forth herein for sake of clarity.

The enumerated listing of items does not imply that any or all of the items are mutually exclusive, unless expressly specified otherwise. The terms "a", "an" and "the" mean "one or more", unless expressly specified otherwise.

A description of an embodiment with several components in communication with each other does not imply that all such components are required. On the contrary a variety of optional components are described to illustrate the wide variety of possible embodiments of the invention.

When a single device or article is described herein, it will be readily apparent that more than one device/article (whether or not they cooperate) may be used in place of a single device/article. Similarly, where more than one device or article is described herein (whether or not they cooperate), it will be readily apparent that a single device/article may be used in place of the more than one device or article, or a different number of devices/articles may be used instead of the shown number of devices or programs. The functionality and/or the features of a device may be alternatively embodied by one or more other devices which are not explicitly described as having such functionality/features. Thus, other embodiments of the invention need not include the device itself.

Finally, the language used in the specification has been principally selected for readability and instructional purposes, and it may not have been selected to delineate or circumscribe the inventive subject matter. It is therefore intended that the scope of the

invention be limited not by this detailed description, but rather by any claims that issue on an application based here on.

While various aspects and embodiments have been disclosed herein, other aspects and embodiments will be apparent to those skilled in the art. The various aspects and embodiments disclosed herein are for purposes of illustration and are not intended to be limiting, with the true scope being indicated by the following claims.

Reference Number	Description
101	Input File Storage
102	Output File Storage
103	Process Flow
104	Process Request Flow
105	API based Process Methods
201	Client Applications
202	System (API based file processing system)
302	Job queue
304	Processing request Module
401	Hybrid model client
402	Legacy model client
403	API based file processing system
404.a	Products & Services using APIs
404.b	Products & Services using batch model
600	Computer system
602	Plurality of processors
601	I/O interface
605	Memory
603	Network interface
604	Storage interface
606	User interface
607	Operating System

608	Web server
611	Communication network
612	API based services and products
613, 614	Client Applications

A SYSTEM AND METHOD FOR API BASED FILE PROCESSING

ABSTRACT

Various embodiments of the present invention provide a system and method for defining, implementing, and / or executing batch processing of API transaction services and products. The system is configured to receive a plurality of file processing requests associated with API traffic from one or more clients and batch, one or more jobs associated with the plurality of file processing requests for the API traffic. Further, the system is configured to pick and initiate the process of executing the API call associated with the API transitions. Furthermore, it splits the input file into plurality of chunks and invokes the API call associated with the corresponding chunk and receives responses from API transitions as well as store the same in chunks. The processor within the system is configured to consolidate the plurality of responses stored in the chunks and write the response to an output file.

[Figure 3]

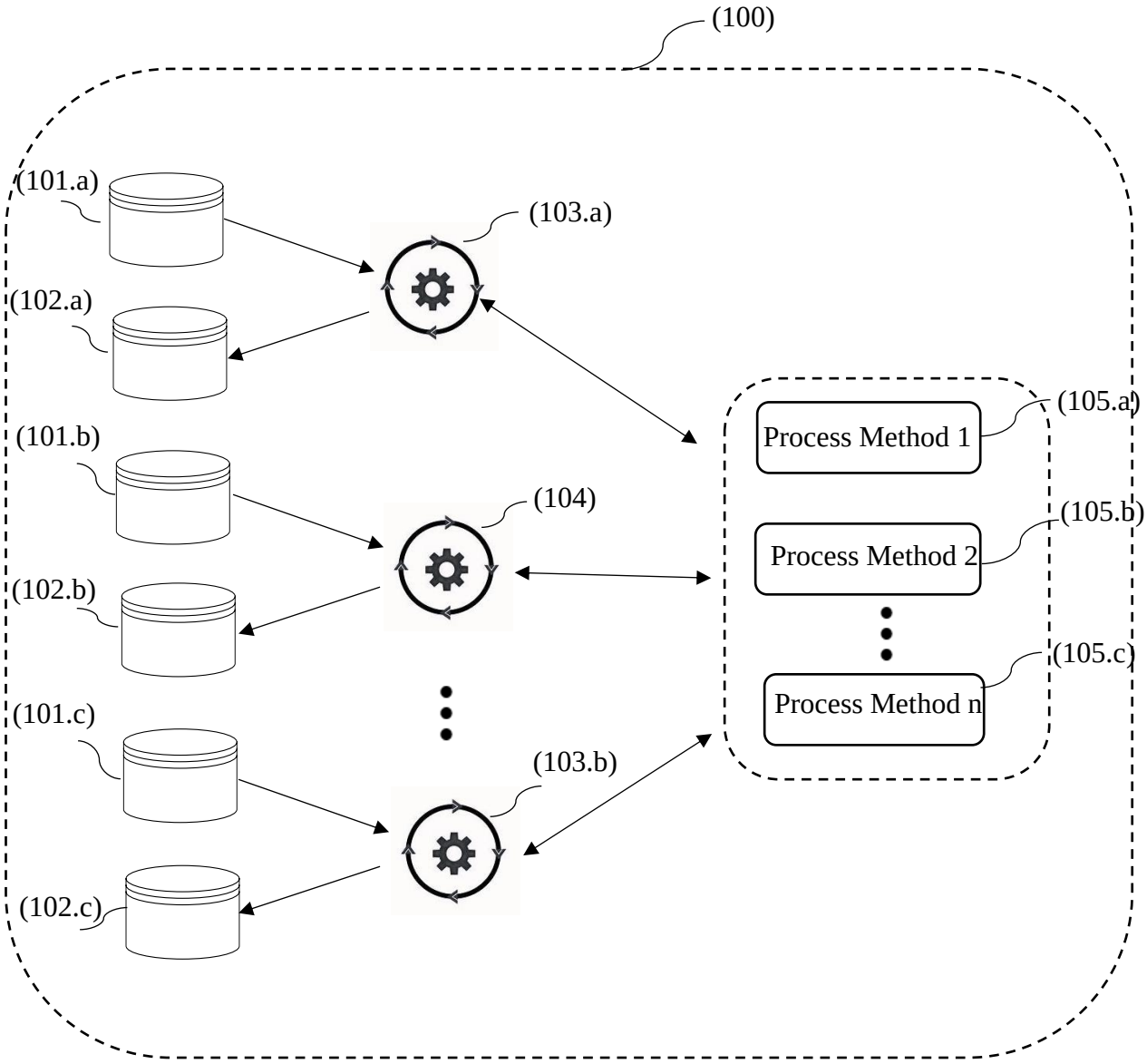
(100)
(101.a)
(103.a)
(102.a)
(101.b)
(104)
(105.a)
Process Method 1
(105.b)
Process Method 2
(102.b)
(105.c)
Process Method n
(101.c)
(103.b)
(102.c)

Figure 1

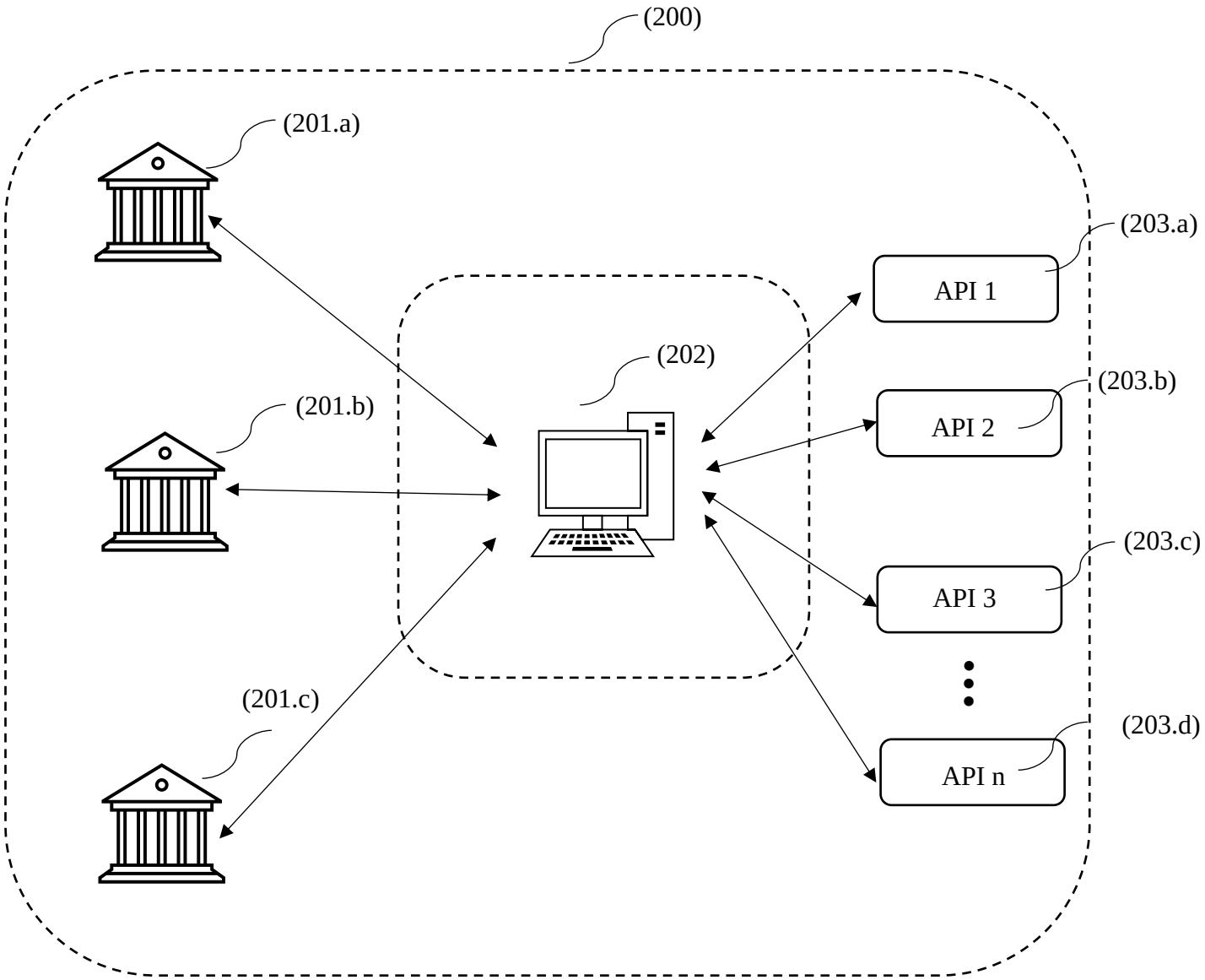
(200)
(201.a)
(201.b)
(201.c)
(202)
(203.a)
API 1
(203.b)
API 2
(203.c)
API 3
(203.d)
API n

Figure 2

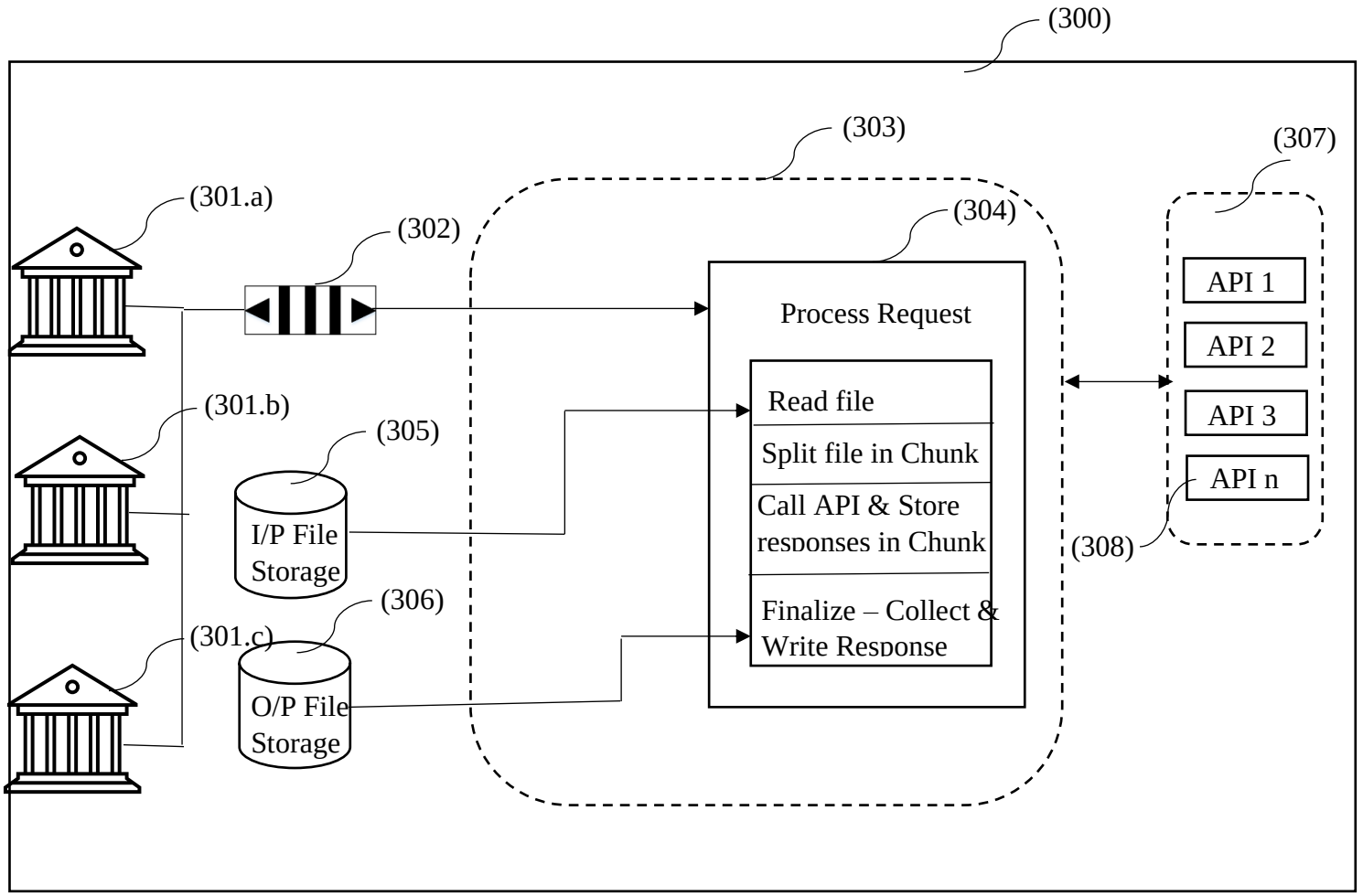
(300)
(303)
(307)
(301.a)
(302)
(304)
Process Request
API 1
API 2
API 3
API n
(301.b)
(305)
Read file
Split file in Chunk
Call API & Store responses in Chunk
I/P File Storage
(306)
(308)
(301.c)
Finalize – Collect & Write Response
O/P File Storage

Figure 3

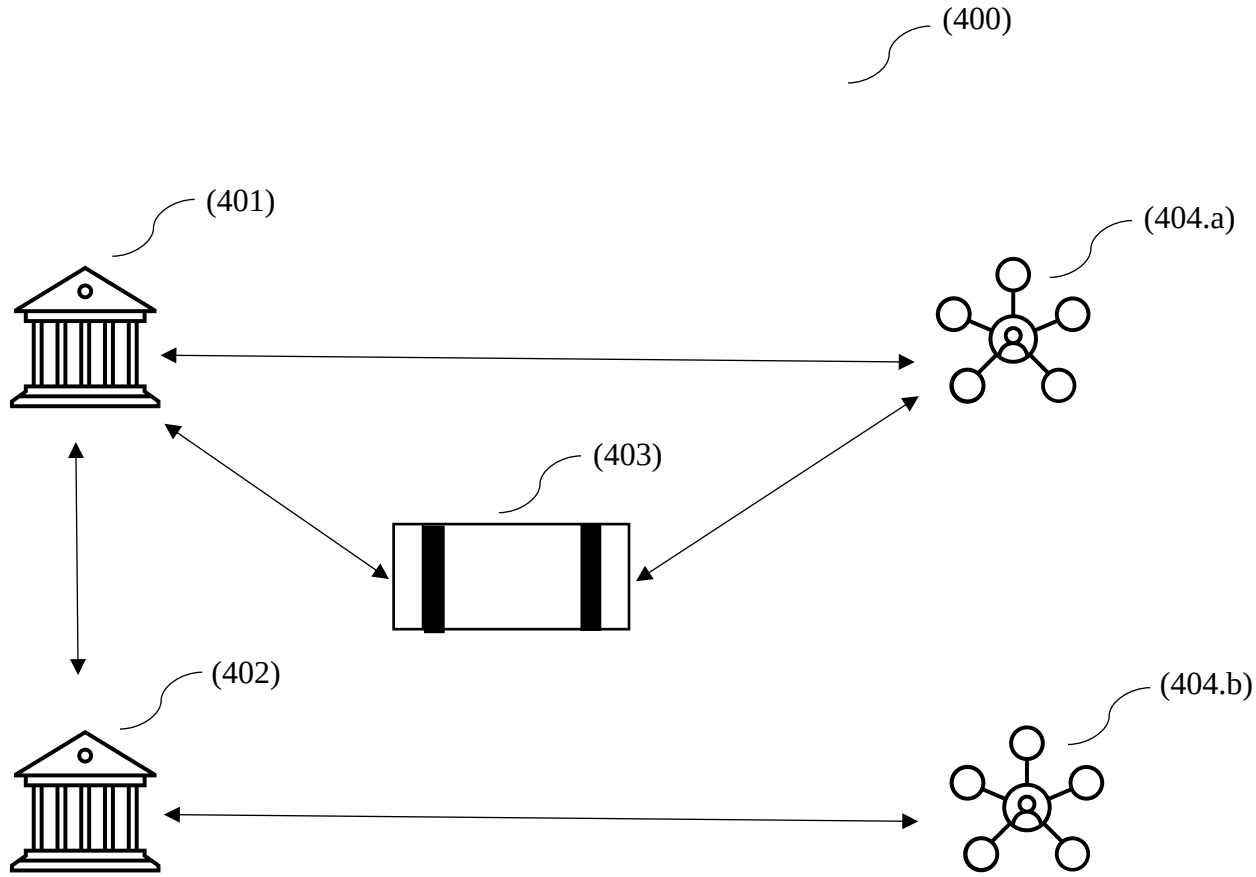

Figure 4

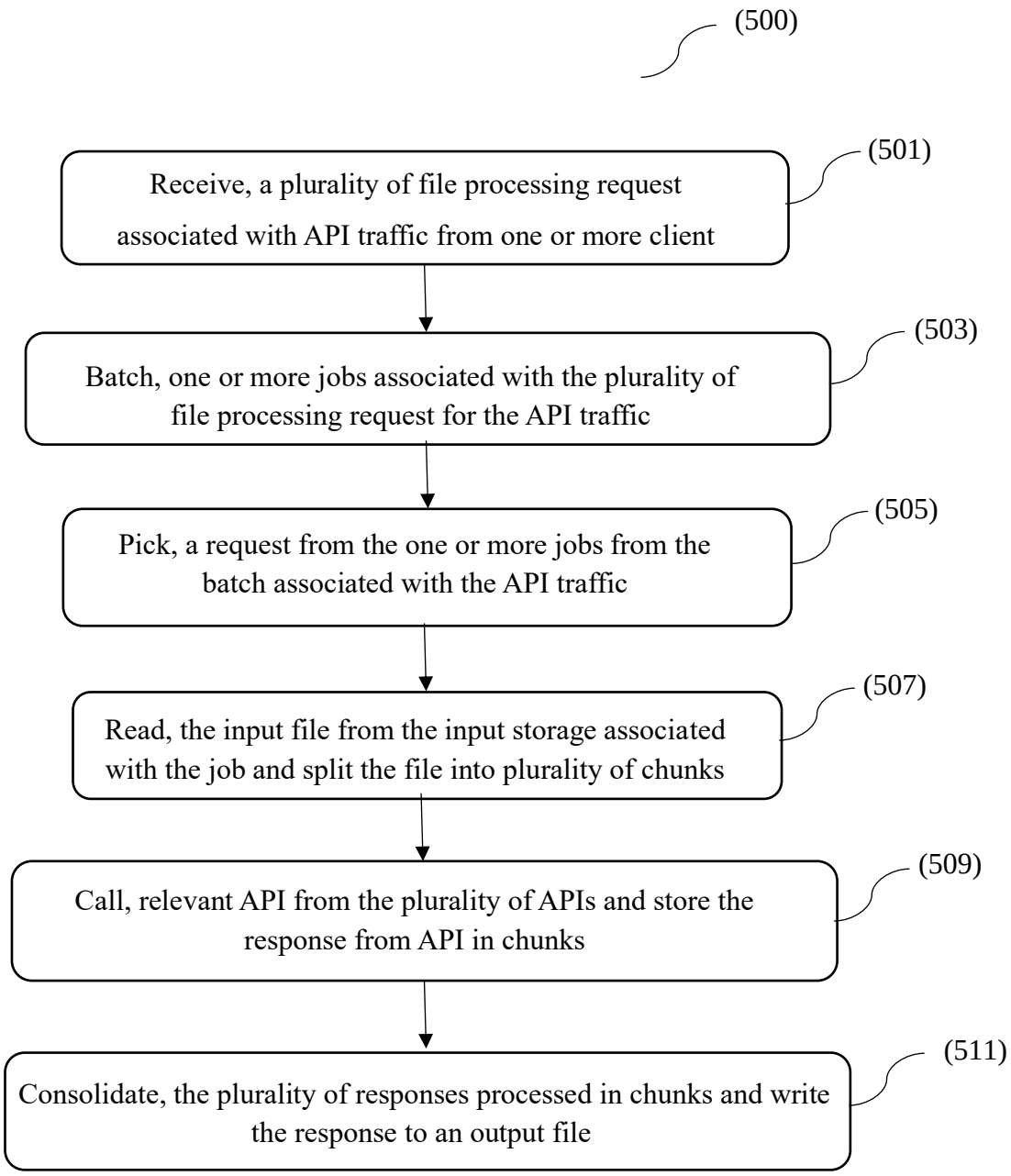
(500)
Receive, a plurality of file processing request associated with API traffic from one or more client
(501)
Batch, one or more jobs associated with the plurality of file processing request for the API traffic
(503)
Pick, a request from the one or more jobs from the batch associated with the API traffic
(505)
Read, the input file from the input storage associated with the job and split the file into plurality of chunks
(507)
Call, relevant API from the plurality of APIs and store the response from API in chunks
(509)
Consolidate, the plurality of responses processed in chunks and write the response to an output file
(511)

Figure 5

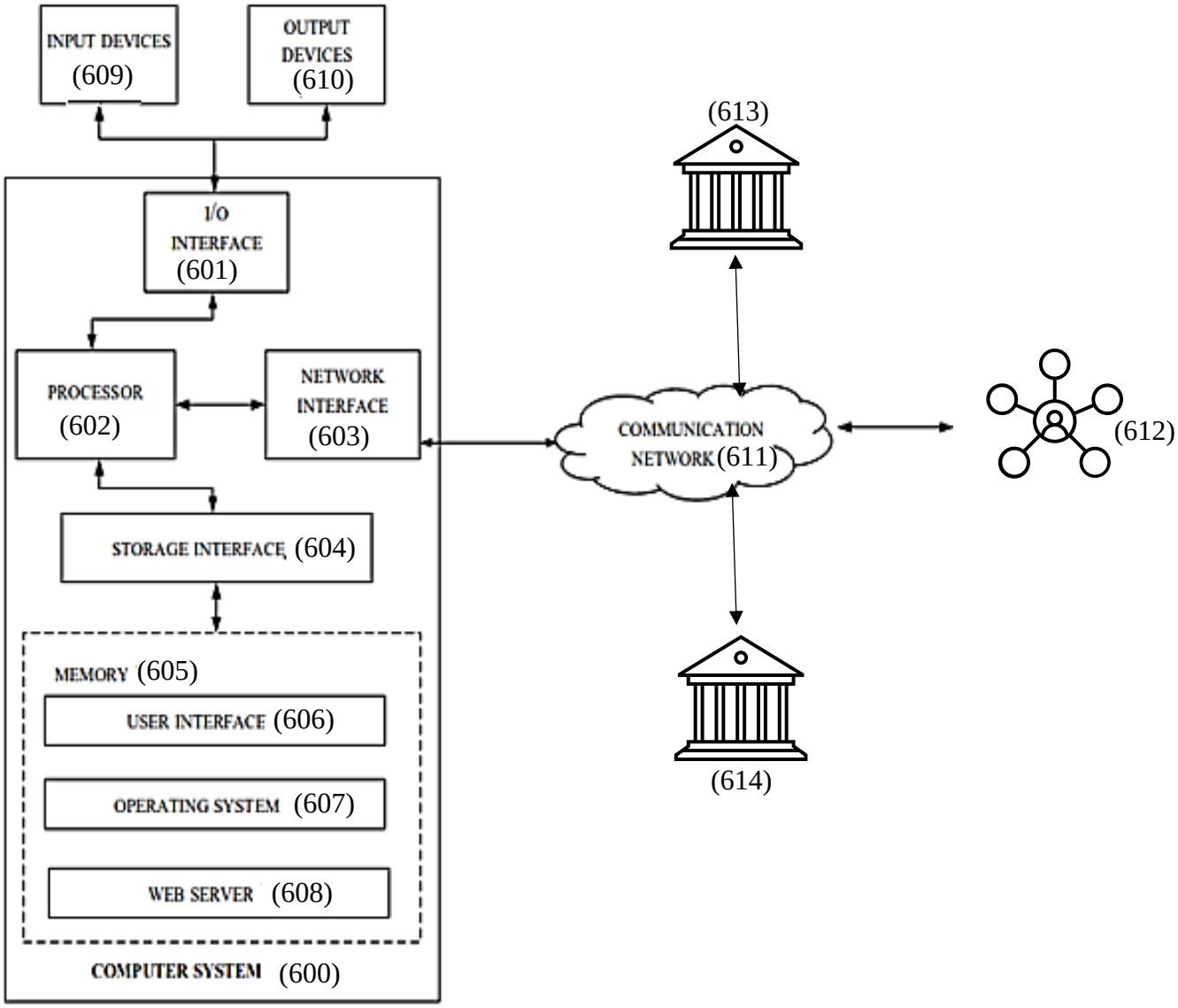
INPUT DEVICES
(609)
OUTPUT
DEVICES
(610)
I/O
INTERFACE
(601)
PROCESSOR
(602)
NETWORK
INTERFACE
(603)
STORAGE INTERFACE (604)
MEMORY (605)
USER INTERFACE (606)
OPERATING SYSTEM (607)
WEB SERVER (608)
COMPUTER SYSTEM (600)
(613)
COMMUNICATION
NETWORK (611)
(612)
(614)

Figure 6