(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
20 September 2012 (20.09.2012)

WIPO | PCT

(10) International Publication Number
WO 2012/123544 A1

(51) International Patent Classification:
G06Q 30/06 (2012.01) *H04L 29/08* (2006.01)
G06Q 50/14 (2012.01)

(21) International Application Number:
PCT/EP2012/054563

(22) International Filing Date:
15 March 2012 (15.03.2012)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
11305281.5 15 March 2011 (15.03.2011) EP
13/065,312 18 March 2011 (18.03.2011) US

(71) Applicant *(for all designated States except US)*: **AMADEUS S.A.S.** [FR/FR]; 485 route du Pin Montard - Sophia Antipolis, F-06410 Biot (FR).

(72) Inventors; and
(75) Inventors/Applicants *(for US only)*: **DOR, Pierre** [FR/FR]; 1019 Route de Grasse, F-06140 Vence (FR). **FAUSER, Dietmar** [DE/FR]; 102 Avenue Jean XXIII, F-06130 Grasse (FR). **DANIEL, Jérôme** [FR/FR]; Lotissement Rivage Azur, Villa 1, Avenue Honoré Lions, F-06130 Grasse (FR). **MONBEL, Stéphane** [FR/FR]; Résidence Les Jardins de la Corniche, 127 Avenue de la Corniche Fleurie, F-06200 Nice (FR). **DEGUET, Cyril** [FR/FR]; 9 Rue du Bosquet, F-06160 Juan Les Pins (FR).

(74) Agent: **HAUTIER, Nicolas**; Cabinet Hautier, 20 Rue De La Liberte, F-06000 Nice (FR).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

[Continued on next page]

(54) Title: METHOD AND SYSTEM FOR PROVIDING A SESSION INVOLVING A PLURALITY OF SOFTWARE APPLICATIONS

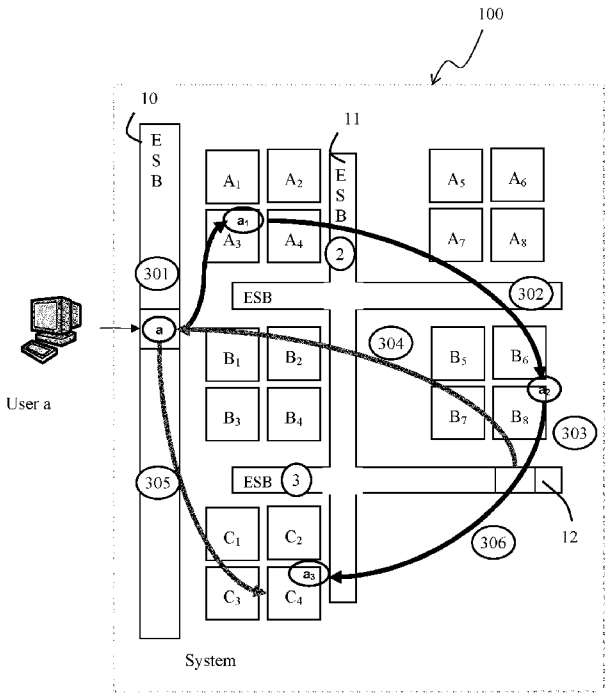

FIGURE 3

(57) Abstract: A computer-implemented method is disclosed for providing a user with a consistent view of user session in a distributed environment. The method includes providing application servers (A1, A2,...) with data storage means for storing part of the user context for that user session, defining thereby for each user session a set of application servers (A3, B8, C4) having each an affinity with the user session. Each application server is configured to process a software application that is required for that user session. At a routing means (10, 11, 12), performing the following steps with at least one data processor: o receiving request from the user and routing transactions of the user session toward the application servers (A1, A2,...) having an affinity with the user session in order to fulfill the request, o assigning to the user session a correlation record (DCX) arranged to comprise Affinity Keys, each Affinity Key indicating an application server that has an affinity with the user session for a given software application, o propagating the correlation record with transactions, allowing thereby the routing means (10, 11, 12) to target the application servers (A3, B8, C4) that are linked to the user context of that user session and that process the software application relevant to process the transaction.

WO 2012/123544 A1

(84) **Designated States** *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:

— *with international search report (Art. 21(3))*

"Method and System for providing a session involving a plurality of software applications"

TECHNICAL FIELD

The present invention relates generally to methods and systems that provide a user of software applications with a consistent view of each open session in a distributed computing environment. More particularly, the invention relates to methods and systems wherein a very high number of user sessions are possibly opened simultaneously and wherein each user session may require the cooperation of various software applications to complete.

BACKGROUND

As the number of people using e-services in their every day life grows everyday, systems providing these e-services are required to process an always increasing number of user sessions. In addition, users are now expecting that information and services delivered to be more and more efficient and sophisticated. Then, to achieve this goal, the transactions for each user session often need to be processed by distinct software applications.

For instance, in the travel and tourism industry, a reservation system has to process tenths of thousands of user sessions opened at the same time and that request checking availability of flights. Information provided by the user to such a user session typically includes the date of departure, the origin and destination of the flights. They form part of the user context for that session which also comprises data gathered from the various transactions processed by the session such as availability information, prices, flight references etc.

In order to maintain a sufficient throughput, some solutions are based on centralized computing systems having a single very powerful processing unit. These centralized systems are however hardly scalable both in term of deployment of new additional applications and in term of increasing of processing capabilities. In addition, these systems are not reliable as a failure of the centralized processing unit may lead to a general outage.

Other solutions rely on distributed systems wherein the total processing capability is spread over a large number of distributed machines. In these existing distributed systems all transactions dedicated to a given user session are still managed by a single machine though.

Scalability is then simply achieved by increasing the number of machines so that the overall processing capability of the system can be easily adapted to fulfill the throughput needs.

However, with these distributed systems, it remains difficult to manage the integration of new services, especially when they are based on new software applications intended to interact with the software applications already loaded while always maintaining consistency from the user side.

Hence, to be able to maintain a consistent and unified view from the user side, the new software applications must be loaded on every single machine. On top of being a time consuming operation this does not however allow consistency between all users to be actually achieved until loading step has completed over the entire distributed system.

Therefore, a general objective of the invention is to describe a method and a system capable of providing a user with a unified view of a session, while allowing scalability of processing capability and while easing the integration of new services.

SUMMARY

The foregoing and other objectives are fulfilled at least partially, and other advantages are realized, in accordance with the embodiments of this invention.

The invention relates to a method of providing a user with a user unified view of a computerized user session, wherein the user session requires the operation of a plurality of software applications processed by a system, the system operating with the user in a client/server mode, a plurality of software applications being each linked to at least a part of a user context related to the user session, characterized in that it comprises the following steps performed with at least one processor:

- for each user session, each software application that is linked to at least a part of a user context is processed by a dedicated application server among a group of application servers processing each independently that software application, defining thereby for each session a set of application servers having each an affinity with the user session,
- providing each application server having an affinity with the user session with data storage means configured to store the part of the user context linked to the application software that is processed by said each application server,
- at one or more routing means, one among the one or more routing means being a main routing means in charge of the user session: receiving at least a request from the user and routing transactions of the user session toward at least the application servers having an affinity with the user session in order to fulfill the request, the step of routing transactions comprising:
  - assigning to the user session a correlation record (DCX) arranged to comprise Affinity Keys, each Affinity Key indicating an application server that has an affinity with the user session for a given software application,
  - propagating the correlation record with each transaction, allowing thereby the routing means to read the correlation record and to target the application servers that have an affinity with the user session and that process the software applications relevant to process the transaction.

Another aspect of the present invention relates to a method of providing a user with a user session, wherein the user session requires the operation of a plurality of software applications processed by a system operating with the user in a client/server mode, at least one of the software applications being linked to at least a part of a user context related to the user session, characterized in that for each user session, each

software application is processed by one application server among a group of application servers that each processes independently said each software application, in that each application server that processes a software application linked to a part of the user context is thereby linked to the user context i.e. has an affinity with the session and is provided with data storage means configured to store the part of the user context to which it is linked. The method comprises a step, performed with at least one processor, of assigning to the user session a correlation record (DCX) configured to comprise Affinity Keys, each Affinity Key indicating the application server that must be targeted for a given software application and for said user session. The method also comprises the following steps that are performed at one among one or more routing means through a processor, one routing means being a main routing means in charge of the user session: receiving a transaction; determining a software application that is called by the transaction, determining if the transaction requires to be routed to an application server linked to the user context:

- if the transaction requires to be routed to an application server linked to the user context then:
  - if the correlation record contains any Affinity Key for the called software application, then routing the transaction according to the Affinity Key;
  - if the correlation record does not contain an Affinity Key for the called software application, one of the routing means selects an application server among the group of application servers that process the called software application, enriches the correlation record with an Affinity Key indicating the selected application server, routes the transaction to the selected application server and the main routing means stores the correlation record.

Optionally and preferably, if the transaction does not require to be routed to an application server linked to the user context, then the routing means selects an application server among the group of application servers that process the called software application, and routes the transaction to the selected application server.

Optionally and preferably, the step of receiving a transaction comprises receiving a transaction from the user. Optionally, the step of receiving a transaction comprises receiving a transaction from a software application.

Another aspect of the invention relates to a non-transitory computer-readable medium that contains software program instructions, where execution of the software program instructions by at least one data processor results in performance of

operations that comprise execution of the method as in any one of the preceding claims.

According to another aspect, the invention relates to a system comprising means configured to execute the method as in any one of the preceding features and steps.

Another aspect of the present invention relates to a system for providing a user with a user session, comprising a plurality of software applications, the operation of which being required for providing the user session, at least one of the software applications being linked to at least a part of a user context related to the user session,

characterized in that it comprises:

- a plurality of machines comprising each at least a processor,
- a plurality of application servers running each on a machine, each application server being arranged so that for that user session each software application is processed by one application server among a group of application servers dedicated to that software application, defining thereby for each user session a set of application servers having each an affinity with the user session,
- data storage means associated to each application server that processes a software application linked to a part of the user context, each storage means being arranged to store the part of the user context that is linked to the software application that is processed by the application server associated to said storage means,
- one or more routing means, among which a main routing means is arranged to be in charge of the user session, the one or more routing means being configured to:

receive at least a request from the user and route transactions of the user session toward the application servers having an affinity with the user session in order to fulfill the request, the routing means being configured so that the transaction step comprises:

- assigning to the user session a correlation record (DCX) arranged to comprise Affinity Keys, each Affinity Key indicating an application server that has an affinity with the user session for a given software application,
- propagating the correlation record with each transaction, allowing thereby the routing means to read the correlation record for targeting the application servers that have an affinity with the user session and that process the software applications relevant to handle the transaction.

BRIEF DESCRIPTION OF THE DRAWINGS

The foregoing and other aspects of the embodiments of this invention are made more evident in the following Detailed Description, when read in conjunction with the attached Drawing Figures, wherein:

Figure 1 shows an example of a system according to an embodiment of the invention.

Figure 2 shows an example of a system according to another embodiment of the invention. An exemplary use case is depicted.

Figure 3 shows the system of Figure 2 and another exemplary use case illustrating the steps wherein the life duration of a part of the user context needs to be linked to the life duration of the user session.

Figure 4 is a simplified high level block diagram of some components of the system that illustrates the affinity initialization steps.

Figure 5 is a simplified high level block diagram of some components of the system that illustrates an exemplary use case.

DETAILED DESCRIPTION

Some advantageous features and steps will be described below. Then some exemplary embodiments and use cases will be further detailed in regard with the drawings.

As indicated above, the invention relates to a method of providing a user with a user unified view of a computerized user session, wherein the user session requires the operation of a plurality of software applications processed by a system, the system operating with the user in a client/server mode, a plurality of software applications being each linked to at least a part of a user context related to the user session. The method comprises the following steps performed with at least one processor:

- for each user session, each software application that is linked to at least a part of a user context is processed by a dedicated application server among a group of application servers processing each independently that software application, defining

thereby for each session a set of application servers having each an affinity with the user session,

- providing each application server having an affinity with the user session with data storage means configured to store the part of the user context linked to the application software that is processed by said each application server,
- at one or more routing means, one among the one or more routing means being a main routing means in charge of the user session: receiving at least a request from the user and routing transactions of the user session toward at least the application servers having an affinity with the user session in order to fulfill the request, the step of routing transactions comprising:
  - assigning to the user session a correlation record (DCX) arranged to comprise Affinity Keys, each Affinity Key indicating an application server that has an affinity with the user session for a given software application,
  - propagating the correlation record with each transaction, allowing thereby the routing means to read the correlation record and to target the application servers that have an affinity with the user session and that process the software applications relevant to process the transaction.

Thus, each group of services is processed on an independent set of application servers, each application server being possibly run by an independent machine, each part of the user context related to a group of services is stored locally on one of the application servers in charge of that group of services and each user interacts with the system as if it was only one application server. The DCX is added on each transaction coming into the system to correlate the parts of the user context that are spread through a possibly high number of servers. The Affinity Key is specific information to identify which application server (and machine) is holding a part of the user context for a given set of services. Then the Affinity Key is used by the ESBs to route the transactions to the specific application server managing the context of a user.

Therefore the invention discloses a solution to provide a user with a consistent and unified view of a user context, keeping modularity of the system to ease the increase of processing capacity and allowing easy integration of new services/products.

The invention also provides significant advantages in term of operability as it allows an easier control of the resources allocated to each sub part of a global product comprising various software applications. It also permits to minimize the risk of global

outage since an outage on a given machine or set of machines does not impact significantly the other machines.

All application servers are independent of each other. They do not know each other. Only the correlation record that is read by the routing means allows establishing a conversation between two application servers processing different software applications. Thus, the Affinity Key is a reference added to the correlation record and that allows the routing means to target the application server that holds the part of the user context linked to the user session and that is relevant to process the transaction.

Optionally, the invention may comprise at least one of the following non limitative features and steps:

The one or more routing means is configured to know how to reach every single application servers.

Thus, the one or more routing means knows the presence of each application server and where each application server is located. The one or more routing means also knows on which application servers a given software application is processed.

Preferably, the correlation record is stored in data storage means associated to the main routing means.

The main routing means is the routing means that receives the user request. The Affinity Keys are set by the routing means.

The routing means is an enterprise service bus (ESB).

A routing means receives a transaction from an application server processing a software application and the correlation record assigned to the session and routes the transaction to another application server dedicated to process another software application. The step of routing the transaction comprises a step wherein the routing means reads the correlation record.

Upon reception of the correlation record at said routing means, if the correlation record does not contain the Affinity Key indicating said other application server where the transaction must be routed to, then the step wherein said routing means routes the transaction to another application server comprises the additional following step: the

routing means determines if a life duration of the part of the user context linked to said other software application must be linked to the life duration of the session.

Upon reception of the correlation record at said routing means, if the correlation record does not contain the Affinity Key indicating said other application server where the transaction must be routed to and if a life duration of the part of the user context linked to said other software application must be linked to a life duration of the user session, then the step wherein said routing means routes the transaction to another application server comprises the additional following step:

- said routing means sends to the main routing means a request for affinity initialization,
- then the main routing means:
  - selects said other application server among the group of application servers that processes said other software application,
  - sends to said other application server a message of affinity initialization,
  - receives from said other application a confirmation of affinity initialization,
  - enriches the correlation record with an Affinity Key indicating said other application server,
  - sends the correlation record as enriched to said routing means,
- then said routing means routes the transaction to said other application server according to the Affinity Key of the correlation record as enriched.

Upon reception of the correlation record at said routing means, if the correlation record does not contain the Affinity Key indicating said other application server where the transaction must be routed to and if the life duration of the part of the user context linked to said other software application does not have to be linked to the life duration of the session, then the step wherein the routing means routes the transaction to said other application server comprises the additional following step: said routing means selects said other application server among the group of application servers that processes said other software application that must be processed to fulfill the transaction and routes the transaction to said other application server.

Preferably, the correlation record is enriched with the Affinity Key indicating said other application server.

The correlation record is enriched by said routing means and the correlation record is returned via the reply path to the main routing means and stored in data storage means associated to the main routing means. The routing means that enriches the correlation record and the main routing means are is distinct. According to an alternative embodiment, the routing means that enriches the correlation record and the main routing means are the same.

Upon reception of the correlation record at said routing means, if the correlation record already contains the Affinity Key indicating said other application server where the transaction must be routed to, then said routing means routes the transaction according to the Affinity Key to said application server.

An application server receives a transaction from a routing means, generates a part of the user context through processing the transaction and stores the part of the user context in data storage means provided with said application server.

The application server enriches the correlation record with an Applicative Context Key which allows retrieving from said application server said part of the user context.

Thus, the Applicative Context Key is a reference that indicates where said part of the user context linked to the software application is located on said application server.

The correlation record as enriched with the Applicative Context Key is sent back via the reply path to the main routing means and is stored in data storage means associated to the main routing means.

Upon reception at said application server of a further transaction related to said user session, said application server locates said part of the user context in said storage means based on the Applicative Context Key. Said part of the user context can then be further retrieved and processed. This allows an easy retrieval of all parts of the user context wherever they are located.

The correlation record is propagated with each transaction. The correlation record is enriched by the routing means with Affinity Keys to allow the routing means to target the application servers that hold the part of the context of the user session. The correlation record is also enriched by the application servers with Applicative Context Keys to allow each application server to retrieve the part of the user context that is

stored in its storage means. Each time the correlation record is enriched with a new Affinity Key or with a new Applicative Context Key, the correlation record is returned to the main routing means.

If a life duration of a part of the user context linked to a software application must be linked to a life duration of the user session, then a stateful conversation is open between the main routing means and the application server processing said software application as long as the user session is active, the stateful conversation being automatically closed upon ending of the user session.

Advantageously, as long as the user session is active, the main routing means regularly indicates to said application server that said part of the user context must be kept stored.

The main routing means indicates to said application server that said part of the user context linked to said other software application can be deleted.

The part of the user context linked to a software application may be generated by the user and/or by the software application linked to the application server comprising data storage means that stores said part of user context. The part of user context linked to a software application may also be required or generated by that software application.

Each data storage means of a given application server is distinct from the data storage means of the other application servers.

The application servers run on machines comprising processor and data storage means. According to an embodiment, at least a plurality of application servers runs on a same machine. According to another embodiment one single application server runs on per machine.

Typically, the session allows at least one of: checking the availability of a travel product, booking a travel product or paying a travel product. For instance, the part of the user context may relate to a travel product availability. Typically, the travel product is a ticket for a flight, railway or boat transportation. The part of the user context may also relate to a travel a passenger name record (PNR).

In the context of the invention, a user session can be considered as a session wherein a user is provided with at least a service, a service requiring one or more software applications to be processed.

For instance, during a user session the user accesses a service related to train ticket availability information and to a service related to flight ticket availability information. Each of these services require the operation of software applications such as for instance: a software application for receiving inputs from the user regarding a departure date and/or an origin and/or a destination; a software application for transforming the protocol of a user request into another protocol; a software application for retrieving availability data regarding train from an railway inventory database, a software application for retrieving availability from an airline inventory database etc.

As illustrated on Figure 1, the system comprises at least a routing means that receives a request from a user. Preferably, the routing means is an enterprise service bus. Alternatively, the routing means can also be a router or any other means able to route a transaction to an appropriate application server. In the following, the routing means will be referred to as ESBS. Whatever is their nature: enterprise service bus, router etc.

For each user session, one ESB is in charge of the session. This ESB is referred to as the main ESB. Preferably, the main ESB is the ESB that receives the request from the user.

Preferably, the system comprises a plurality of machines. Each machine is a hardware device that comprises at least a processing means running at least an application server. At least some of the machines also comprise data storage means. At least an application server runs on a machine. The application server uses the processing means. At least some of the application servers also use the data storage means. Thus an application server is linked to processing means and eventually to data storage means.

According to a particular embodiment, a plurality of application servers runs on a single machine. Each application server may use its own processor means and eventually its own eventual data storage means. Alternatively, a plurality of application servers may share the same processor and eventually the same data storage means.

According to another embodiment, only one application server runs for a given machine. Thus, according to this other embodiment, routing a transaction to a machine will also mean routing a transaction to an application server in the description below.

Each application server processes a software application.

Advantageously, the same software application is independently processed by a plurality of application servers, these application servers running on the same machine or on different machines.

The system can also be referred to as a platform.

Application servers are organized into groups 20, 30, 40, 50.

All application servers of the same group of application servers process the same software application. For instance, each application server A1, A2, ..., A12 of group 20 processes the software application A. Each application server B1, B2, ..., B8 of group 30 processes the software application B. Each application server C1, C2, C3 of group 40, processes the software application C.

For a given session there is only one application server that operates per software application. Thus, among a group of application servers processing the same software application, one application server is dedicated to that given session.

As indicated above, each application server that needs to store data is provided with data storage means. The application server dedicated to a session stores in its storage means the part of the user context that is relevant for the software application that is processed by that dedicated application server.

In the present invention, the user context also referred to as context is a context related to a user and relevant to process a session. It represents all the functional and technical information that are used by the system for this specific user to perform the requested functionalities, e.g. in the travel reservation system, the reservation (shopping) session context which is linked to an active end session. The part of the user context may be data provided by the user such as user personal references, a departure date, an origin or a destination. This part of the user context may be required by the software application linked to the storage means. The part of the user context may also be data generated by the software application. For instance, the part of the user context may relate to flight availability retrieved by the software application or to prices computed or retrieved by the software application. The part of the user context may also relate to a passenger name record (PNR) or a part of a PNR.

Thus, the context of the user is distributed over a possible very large number of application servers, each part of the user context being stored locally. As depicted on

Figure 1, the context of user Ua comprises the parts a0, a1, a2, a3 and is distributed through the application servers A2, B3, C2. The context of user Ub comprises the parts b0, b1, b2, b3 and is distributed through the application servers A5, B5, C4. More precisely, the part a1 of the context of the user Ua is stored in the data storage means of the application server A2; the part a2 of the context of the user Ua is stored in the data storage means of the application server B3; the part c3 of the context of the user Uc is stored in the data storage means of the application server C3 etc.

Therefore, all application servers that are dedicated to a given user session form a set of application servers having each an affinity with that session.

For processing a session and to provide the user with a unified view of the session, the system has to target all application servers having an affinity with that session.

Providing the user with a unified view of the session means that the user does not notice that the various software applications are processed by various and independent application servers. Thus, the number of software applications and the application servers involved in the session is transparent to the user. The user interacts with the system as if it is one unique application server and machine.

The solution for targeting the relevant application servers is detailed below with reference to Figures 2 to 5.

Figure 1 clearly illustrates that additional application server can be added to the system, increasing thereby the processing capability of the entire system. For instance, the application servers A9, A10, A11, A12 are added to the group 20 comprising the application servers A1, A2,..., A8. Thus, more transactions can be processed by group 20 and a higher number of sessions can be handled simultaneously without decreasing the processing time. Additionally, the integration of additional application servers is easy and is totally transparent to the user.

Advantageously, the invention also allows integrating to the system new services or new software applications. The group 50 of application servers D1, D2, D3, D4 illustrates the integration of application servers processing a new software application that does not process the previously integrated application servers. Thus, additional services can be offered to the all users without requiring loading the new software application on every single application server already integrated. Moreover, integrating a new software application in the system is quite simple as only the ESB(s) must know the presence of these new software application and application servers. More precisely, when integrating each new application server, the application server and the

software application that runs on it are declared to the ESB(s). In case there are a plurality of ESBs, all ESBs know the topology of the system i.e., each ESB knows where to locate each application server and what is the software application that is processed on each application server.

In a particular embodiment, more than one application server runs on a single hardware machine. However, the ESBs know all application servers dedicated to each software application.

The groups of application servers already in the system do not have to know that additional application servers or additional software applications are integrated. Thus, integrating additional processing capacity or additional services does not impact the rest of the system and is transparent to the use.

Thanks to the invention, having the new software application available for all users at the same time is also easy and is not time consuming.

In the illustrated use case, every application server holds a part of the user context. In some embodiments, an application server does not store a part of the user context. For instance, such application server is in charge of retrieving some static data such as HTML resources in charge if propagating sell transactions to external providers.

Affinity Keys

The solution for providing a user with a unified view of the user session will be now explained with further details.

During a session, a user enters the system through the same ESB. Preferably, the ESB that forms the entry point is the main ESB and is in charge of the session until the end of that session.

For each transaction coming into the system, the main ESB creates a record referred to as the Distributed Context Correlator (DCX).

The DCX is dedicated to a unique user session. It is stored at the main ESB. The DCX comprises references. A reference identifies which application server is holding the user context for a given software application or set of software application. This reference is referred to as the Affinity Key. As each ESB knows the topology of the system, by reading the Affinity Keys of the DCX, the ESB is able to target the application server having an affinity with the session i.e. the application server using data storage means that stores the part of the user context. Thus, the entire context of

the user can be accessed, even if it is spread through a high number of independent machines.

The DCX is added to each transaction. It is cascaded to all transactions between two application servers. When an application server calls another application server, the DCX is transmitted to that other application server. Such transactions wherein an application server calls another application server are referred to as collateral calls or collateral transactions.

Preferably, each application server that receives the DCX can enrich it with additional information that allows said application server to retrieve the part of the user context that is stored in its storage means. Thus, next time the application server will receive a transaction for that session; it will be able to easily retrieve the part of the user context for that session by simply reading that information of the DCX and wherever from which other application server the transaction is coming. In the present invention, this piece of information stored in the DCX is referred to as the Applicative Context Key.

Thus, the DCX comprises a reference of the session it is attached to, Affinity Keys for targeting application servers having an affinity with that session and Applicative Context Keys to allow each application server to retrieve its part of the user context.

Applicative Context Keys

The DCX also contains a unique identifier. According to a particular embodiment, this unique identifier can be used to reference the sub applicative contexts instead of using a specific reference. Thus this unique identifier can be used by the application server as an implicit applicative context key instead of creating and storing their own applicative context key. In such a case, the application context is indexed with the unique identifier in the storage means. Thus, the DCX contains less information while still allowing the application servers to retrieve their part of the user context.

Advantageously, the DCX is available on any transaction, independently of the communication protocol that is used in the transaction.

Therefore at any time when a software application is called, the ESB that receives a transaction determines if that transaction requires to be routed to an application server having an affinity with that session.

If an affinity is required, i.e., the transaction needs to be routed to an application server that holds a part of the user context associated to that software application, then the ESB has to target that application server. To this end, the ESB reads the Affinity Key of the DCX and identifies the relevant application server.

Then, said relevant application server receives the transaction and the DCX. Then, it can read the Application Context Key to retrieve the part of the user context that is required to process the transaction.

If no affinity is required by the transaction that reaches the ESB, said transaction does not need to be routed to a specific application server of the group of application servers that processes the called software application.

For instance, this occurs for a software application that generates its own part of the user context and when the application server processing said software application receives for the first time a transaction related to that session. Before that first transaction, no part of the user context has been created by said application server. A typical example relates to a use case wherein the application server aims at retrieving flights that are available for given parameters: date/origin/destination. Before receiving these parameters, the application server does not hold any context for that session. However after a first processing, said application server stores the retrieved flights in its dedicated storage means, creating thereby a part of the user context.

Then an affinity is created between that application server and the session. The application server enriches the DCX with an Affinity Key that will further allow every ESB to target it. Preferably, the application server also enriches the DCX with Applicative Context Key that will allow it to retrieve the available flights next time it receives a transaction for that session.

Once the DCX is enriched, it is sent back using the reply path to the main ESB where it is stored.

Another example wherein no affinity is required relates to software application that does not necessitate storing any data related to the session. For instance such a software application allows modifying the format of the message to suit another protocol. Once transformed, the message is then propagated but does not need to be stored at the application server that has performed the format transformation. Then, the DCX is not enriched with any Affinity Key for that software application.

When an ESB receives a transaction and determines that this transaction does not require affinity, the ESB decides the application server where to route the

transaction. Typically, the ESB takes into account load balancing or proximity parameters.

The invention will be now explained with reference to **Figure 2** which depicts an exemplary use case.

For sake of clarity, only user Ua is represented on Figure 2. At the beginning of the session, there is no user context. In this embodiment, the system comprises a plurality of ESBs 10, 11, 12. ESBs are configured so that a transaction reaches an ESB after each processing on an application server.

User Ua starts the session and enters the system through ESB 10, which is therefore the main ESB for that session. User Ua provides main ESB 10 with information forming the part a_0 of the user context. Main ESB 10 holds the part a_0 of the user context. Typically, a_0 comprises the DCX or is the DCX as stored in the main ESB. ESB 10 determines that the software application A is called. ESB 10 determines whether an affinity is requested. As affinity is requested at this stage and as the DCX does not contain any Affinity Key for software application A, then ESB selects any one of the application server of the group of application servers that processes software application A. In this example, application server A3 is chosen. Main ESB 10 sends the transaction (transaction 1) to application server A3 (step 201). Application server A3 processes the transaction, generates the part a1 of user context and stores a_1 in the data storage means of the machine where A3 runs. Main ESB also enriches the DCX with an Affinity Key that will allow any ESB to further target application server A3 when software application A will be called for that session. The request of affinity is part of the ESB configuration. Based on a transaction parameters, such as the source, the destination and the software application of the transaction reaching an ESB, that ESB, thanks to its configuration is able to determine whether the transaction must be processed taking into account an affinity. Thus, static information (configuration of the ESB) is taken into account in order to determine whether an affinity is requested while. The content of the DCX is dynamic information.

Preferably, application server A3 adds in the DCX the Application Context Key that will further allows application server A3 to easily retrieve part a_1 of the context next time it will receive a transaction for that session.

The DCX is then returned back to the main ESB 10 and is stored there. ESB 11 receives the transaction from application server A3 and determines that software application B is called by software application A (step 202). Such call is a collateral call

or collateral transaction. As an affinity is requested for software application B but no affinity is stored in the DCX for application B, then ESB11 can select any one of the application servers of the group of application servers running software application B. ESB 11 selects application server B8, enriches the DCX with an affinity key corresponding to B8 and collateral transaction required to process transaction 1 reaches application server B8. Application server B8 processes the transaction and creates a part a_2 of the context for user Ua. Parts of the user context that are created through collateral calls are also referred to as sub-context.

The DCX is enriched with the Applicative Context Keys that relates to application server B8.

Similarly than for calls 201, 202 received by software application A and software application B, software application C is called without request for affinity (step 203). Application server C4 processing software application C is selected to process transaction 1. Application server C4 creates the part a_3 of the user context and stores it in its data storage means. The DCX is enriched with an Affinity Key that allows targeting application server C4 to retrieve the part a_3 of the user context. The DCX is returned back with the reply to the main ESB 10.

The main ESB 10 receives a new transaction from user Ua (transaction 2). Main ESB 10 determines that software application B is called to process that new transaction, and that an affinity is requested. Thus, the application server of group of service B that holds the part of the user context associated to that service must be targeted. Through reading the DCX, main ESB 10 reads the reference of the relevant application server of group of service B for that session and therefore targets application server B8.

Then main ESB 10 then routes the new transaction to application server B8 (step 204).

Then application server B8 will receive the new transaction will retrieve its part a_3 of the user context and will process the new transaction.

Therefore, through allowing the various servers and ESB to interact, the invention provides an efficient solution to provide a unified view of a session while its context is distributed and stored locally.

Context life duration of a part of a user context

Non limitative but particularly advantageous features of the invention will be now described. These features allow maintaining available during all the session some parts

of the user context that could eventually be required for processing a transaction during that session. When the session ends, these parts of the context are removed.

Indeed, the user context is preferably valid only during the life duration of the session. Thus, the context of the session is not persistent, and upon ending of the session, the context is not stored any more and is removed.

For instance, all the parts of a user context used in the user context of a user session to manage a booking are kept until the end of the user session.

The Distributed Context Correlator (DCX) is available during the whole life duration of a user session. Any application server holding part of the user context can request to link the life duration of its context to the life duration of the user session.

User context and user session are linked in term of life duration by opening a stateful conversation between the ESB holding the DCX and the application server holding the user context.

The request for opening a stateful conversation is statically defined in the ESB configuration as part of the service called. More precisely, when an ESB receives a transaction, it analyses the parameters of the transaction such as its source, destination and the called software application. Then the ESB, thanks to its configuration is able to determine whether a stateful conversation must be opened. The same mechanism applies for determining whether a transaction must takes into account an affinity.

If an application server calls a service or software application processed on another application server requiring to have the life duration of its user context linked to the one of the DCX, the ESB that is in charge of routing the collateral transaction then calls the main ESB and asks the latter to open a stateful conversation with said other application server. Then the main ESB opens a stateful conversation with said other application server. The ESB in charge of routing the collateral transaction then routes the transaction to said other application server on which the stateful conversation has been opened with the main ESB.

A stateful conversation between the main ESB and a given application server has to be open prior to any transaction going to that given application server. This is for this reason that if a transaction has to be routed to a given application server by collateral traffic and before having a stateful conversation opened, the ESB routing the collateral transaction will request the main ESB to open the stateful conversation before routing the transaction to the application server. This functionality is referred to as auto-init of conversation.

Once such a stateful conversation is opened, the life duration of the DCX is linked to the one of the user context stored in the machine where the application server that is in stateful conversation with the main ESB runs. Said user context will be deleted only when the DCX will disappear.

Thanks to this solution, the invention allows maintaining all parts of the context that must be available during the session while limiting the data stored at the end of the session. Additionally, this solution limits the traffic and the volume of data transmitted between application servers.

According to a preferred embodiment, when a stateful conversation is open, the ESB regularly sends messages to the application server sharing its stateful conversation, said messages indicating that the user context must be maintained. These messages are referred to as “keep alive” messages. Thus, “keep alive” messages are sent from the main ESB to all application servers in stateful conversation with that main ESB. When the session ends, the DCX can be deleted and the main ESB sends to all application servers in stateful conversation a message indicating that the user context can be deleted. These messages are referred to as “terminate” messages.

There is only one stateful conversation from the main ESB owner of the DCX per application server or machine and user context. All the collateral transactions are using stateless conversations, even if the global transaction is stateful.

Figure 3 depicts a first example of transaction wherein a stateful conversation must be open.

User Ua sends a request to the system though main EBS 10. Steps 301 and 302 are similar to steps 201 and 202 of Figure 2. When application server B8 has processed the transaction, ESB 12 receives the transaction (step 303) and is in charge of routing it. ESB 12 determines that the software application C that is called by software application B requires having the life duration of the part of its user context linked to the one of the session.

If no part of user context already exists for customer Ua on any application server of service group C, then ESB 12 in charge of routing the transaction to service group C calls main ESB 10 holding the DCX and requests it to open a stateful conversation with one application server of the service group C (step 304). Then, main ESB 10 opens a stateful conversation with one application server of the application servers of service group C (step 305). Then the DCX is enriched with the reference of the chosen

application server C4. Then DCX is returned back to ESB 12 in charge of routing the collateral transaction. Then ESB 12 targets the application server C4 through reading the DCX and routes the transaction to application server C4 that is in stateful conversation with main ESB (step 306).

Figure 4 depicts another example of transaction wherein a stateful conversation must be open.

Through a graphical user interface (GUI), the main ESB receives at step 401 a request from a user. For instance, that request is a creation of a passenger file. Its protocol is HTML. In order to be processed by the system, the request must be transformed into another protocol such as an EDIFACT protocol. Main ESB determines that for transforming the request, an application server of the group of application server processing software application A must be targeted. No affinity is required for such transformation. Then main ESB selects an application server of group of service A and routes the transaction to that application server (step 402). That application server is referred to as A1.

A1 processes the transaction by transforming the protocol. No data has to be stored by A1. Therefore, no part of the user context is created at A1. Conversation between main ESB and A1 is stateless.

Then, an ESB, referred to as collateral ESB (ESB Col) receives the transaction (step 403). Collateral ESB determines that the transaction must be routed to an application server that processes software application B. Collateral ESB also determines that an affinity is requested and that the life duration of the part of the context associated to the application server in charge of the next processing will have to be linked to the life duration of the session. Collateral ESB reads the DXC and determines that no affinity is set in the DCX for software application B (step 404). Then collateral ESB calls the main ESB and requests him to open a stateful conversation with an application server processing software application B (step 405). Main ESB selects an application server, referred to as B2, dedicated to process software application B. Main ESB enriches the DCX with affinity key corresponding to B2 (step 406) and forwards to that application server the request for affinity initialization (step 407). Thus, B2 is informed that the life duration of its part of user context must be linked to the life duration of the session (step 405). B2 allocates a portion of its data storage means to the part of the user context (step 408). B2 enriches the DCX with the

Applicative Context Key and sends back the DCX to the main ESB (step 409). DCX is stored at the main ESB and is sent back to the collateral ESB (step 410).

Collateral ESB retrieves the original message, updates it with new DCX and targets application server B2 in accordance with the Affinity Key of the DCX (step 411). The transaction is thus routed to B2 (step 411). B2 then receives a request in an EDIFACT protocol and processes it. For instance, the request may consist in creating a passenger file.

As the life duration of the part of the user context of B2 is linked to the one of the session, the passenger file will be available at B2 during the entire time life of the session and will be deleted afterwards.

Figure 5 depicts another use case of the invention. In this exemplary use case, ESB 10 comprises a plurality of multiplexer (MUX1, MUX2, MUX3) and at least a server (SRV2) in connection with multiplexer MUX3.

At step 501, a user sends a request to server SRV2. Server SRV2 creates a DCX record for that session.

As the session has just started, no context is available yet and the application server A1 processing software application A is called and receives the transaction along with the DCX. DCX is enriched which an Affinity Key for that application server A1. Affinity is then set for the software application A (step 502). A1 processes the transaction, stores its part of user context and enriches the DCX with it Applicative Context Key. For instance, user context relates to flight availabilities and A1 is in charge of retrieving references of available flights. DCX is then sent back (step 503) at main ESB 10 and stored (step 504). At step 505, the user is provided with results of the processing of A1 (references of flights that are available). This ends the first transactions for that session.

The user then initiates another transaction. For instance the user wants to purchase one segment of the flights that were retrieved through transaction 1.

SRV2 receives the purchase requests and determines that software application B must be called (step 506). An affinity is then created with an application server B1of group of service B. DCX is sent to that application server B1 (step 507). An Applicative Context Key is also created for B1 (508). The DCX is then cascaded through multiplexer MUX1 in charge of handling all transactions coming from application servers processing software application B (step 509). MUX1 determines that software application A is called with a request of affinity. The DCX allows MUX1 to target A1

(step 510). Thanks to its Applicative Context Key, A1 retrieves its part of the context stored during the previous transaction (references of available flights). Then A1 sends back the transaction along with the DCX to MUX1 (step 511). The reply from A1 is return by ESB to B1 (step 512). B1 processes the transaction thanks to the data retrieve from A1. For instance, an available segment is priced. DCX is cascaded (step 513) and sent with the results of the processing to the main ESB (step 514). DCX is stored. Results of processing are provided to the user (step 515).

As detailed in the above description, the invention provides a solution to provide a consistent and unified view of a user context, keeping modularity of the system to ease the increase of processing capacity and allowing easy integration of new services/products.

The invention also provides significant advantages in term of operability as it allows an easier control of the resources allocated to each sub part of a global product comprising various software applications. It also enables to minimize the risk of global outage since an outage on a given machine or set of machines does not impact significantly the other machines.

The foregoing description has provided by way of exemplary and non-limiting examples a full and informative description of various method, apparatus and computer program software for implementing the exemplary embodiments of this invention. However, various modifications and adaptations may become apparent to those skilled in the relevant arts in view of the foregoing description, when read in conjunction with the accompanying drawings and the appended claims. As but some examples, the use of other similar or equivalent data structures and logic flows may be attempted by those skilled in the art. However, all such and similar modifications of the teachings of this invention will still fall within the scope of the embodiments of this invention.

Furthermore, some of the features of the exemplary embodiments of this invention may be used to advantage without the corresponding use of other features. As such, the foregoing description should be considered as merely illustrative of the principles, teachings and embodiments of this invention, and not in limitation thereof.

CLAIMS

1. Method of providing a user (1, 2, 3) with a user session, wherein the user session requires the operation of a plurality of software applications processed by a system (100), the system operating with the user in a client/server mode, a plurality of software applications being linked to at least a part of a user context related to the user session, characterized in that it comprises the following steps performed with at least one data processor:

- for each user session, each software application that is linked to at least a part of a user context is processed by a dedicated application server (A1, A2,...) among a group (20, 30, 40, 50) of application servers (A1, A2,...) processing each independently that software application, defining thereby for each session a set of application servers (A3, B8, C4) having each an affinity with the user session,
- providing each application server having an affinity with the user session with data storage means configured to store the part of the user context linked to the application software that is processed by said each application server,
- at one or more routing means (10, 11, 12), one among the one or more routing means being a main routing means (10) in charge of the user session: receiving at least a request from the user and routing transactions of the user session toward at least the application servers (A3, B8, C4) having an affinity with the user session in order to fulfill the request, the step of routing transactions comprising:
  - assigning to the user session a correlation record (DCX) arranged to comprise Affinity Keys, each Affinity Key indicating an application server that has an affinity with the user session for a given software application,
  - propagating the correlation record with each transaction, allowing thereby the routing means (10, 11, 12) to read the correlation record for targeting the application servers (A3, B8, C4) that have an affinity with the user session and that process the software applications relevant to handle the transaction.

2. The method according to the preceding claim, wherein the one or more routing means (10, 11, 12) is configured to know how to reach every single application servers (A3, B8, C4).

3. The method according to any one of the preceding claims, wherein the correlation record is stored in data storage means associated to the main routing means (10).

4. The method according to any one of the preceding claims, wherein the main routing means (10) is the routing means (10, 11, 12) that receives the request from the user.

5. The method according to any one of the preceding claims, wherein the Affinity Keys are set by the routing means (10, 11, 12).

6. The method according to any one of the preceding claims, wherein the routing means (10, 11, 12) is an enterprise service bus (ESB).

7. The method according to any one of the preceding claims, wherein a routing means (10, 11, 12) receives, from an application server processing a software application, a transaction along with the correlation record assigned to the user session; then routes the transaction to another application server processing another software application, the step of routing the transaction comprises a step wherein the routing means (10, 11, 12) reads the correlation record.

8. The method according to the preceding claim, wherein upon reception of the correlation record at said routing means (10, 11, 12), if the correlation record does not contain an Affinity Key indicating said other application server, then the step wherein said routing means (10, 11, 12) routes the transaction to another application server comprises the following step: the routing means (10, 11, 12) determines if a life duration of the part of the user context linked to said other software application must be linked to the life duration of the user session.

9. The method according to any one of the two preceding claims, wherein upon reception of the correlation record at said routing means (10, 11, 12), if the correlation record does not contain an Affinity Key indicating said other application server and if a life duration of the part of the user context linked to said other software application must be linked to a life duration of the user session, then the step wherein

said routing means (10, 11, 12) routes the transaction to another application server comprises the additional following steps:

- said routing means (10, 11, 12) sends to the main routing means (10) a request for affinity initialization,
- then the main routing means (10):
  - selects said other application server among the group of application servers (B1, B2,...) that processes said other software application,
  - enriches the correlation record with an Affinity Key indicating said other application server,
  - sends to said other application server a message of affinity initialization,
  - receives from said other application a confirmation of affinity initialization,
  - sends the correlation record as enriched to said routing means (10, 11, 12),
- then said routing means (10, 11, 12) routes the transaction to said other application server according to the Affinity Key of the correlation record as enriched.

10. The method according to claim 8, wherein upon reception of the correlation record at said routing means (10, 11, 12), if the correlation record does not contain the Affinity Key indicating said other application server and if the life duration of the part of the user context linked to said other software application does not have to be linked to the life duration of the user session, then the step wherein the routing means (10, 11, 12) routes the transaction to said other application server comprises the additional following step: said routing means (10, 11, 12) selects said other application server among the group of application servers (B1, B2,...) that processes said other software application and routes the transaction to said other application server.

11. The method according to the preceding claim, wherein the correlation record is enriched with the Affinity Key indicating said other application server.

12. The method according to the preceding claim, wherein the correlation record is enriched by said routing means (10, 11, 12) and wherein the correlation record is

returned to the main routing means (10) and stored in data storage means associated to the main routing means (10).

13. The method according to claim 7, wherein upon reception of the correlation record at said routing means (10, 11, 12), if the correlation record already contains the Affinity Key indicating said other application server, then said routing means (10, 11, 12) routes the transaction according to the Affinity Key.

14. The method according to any one of the preceding claims, wherein an application server receives a transaction from a routing means (10, 11, 12), generates a part of the user context through processing the transaction and stores the part of the user context in data storage means provided with said application server.

15. The method according to the preceding claim, wherein the application server enriches the correlation record with an Applicative Context Key which allows retrieving from said application server said part of the user context.

16. The method according to the preceding claim, wherein the correlation record as enriched with the Applicative Context Key is sent to the main routing means (10) and is stored in data storage means associated to the main routing means (10).

17. The method according to any one of the two preceding claims, wherein upon reception at said application server of a further transaction related to said user session, said application server locates said part of the user context in said storage means based on the Applicative Context Key.

18. The method according to any one of claims 1 to 9, wherein if a life duration of a part of the user context linked to a software application must be linked to a life duration of the user session, then a stateful conversation is open between the main routing means (10) and the application server processing said software application as long as the user session is active, the stateful conversation being automatically closed upon ending of the user session.

19. A non-transitory computer-readable medium that contains software program instructions, where execution of the software program instructions by at least

one data processor results in performance of operations that comprise execution of the method as in any one of the preceding claims.

20. System (100) for providing a user (1, 2, 3) with a user session, comprising a plurality of software applications, the operation of which being required for providing the user session, at least one of the software applications being linked to at least a part of a user context related to the user session,

characterized in that it comprises:

- a plurality of machines comprising each at least a processor,
- a plurality of application servers (A1, A2,...) running each on a machine, each application server being arranged so that for that user session each software application is processed by one application server among a group (20, 30, 40, 50) of application servers (A1, A2,...) dedicated to that software application, defining thereby for each user session a set of application servers (A3, B8, C4) having each an affinity with the user session,
- data storage means associated to each application server that processes a software application linked to a part of the user context, each storage means being arranged to store the part of the user context that is linked to the software application that is processed by the application server associated to said storage means,
- one or more routing means (10, 11, 12), among which a main routing means (10) (10) is arranged to be in charge of the user session, the one or more routing means (10, 11, 12) being configured to:

receive at least a request from the user and route transactions of the user session toward the application servers (A3, B8, C4) having an affinity with the user session in order to fulfill the request, the routing means (10, 11, 12) being configured so that the transaction step comprises:

- assigning to the user session a correlation record (DCX) arranged to comprise Affinity Keys, each Affinity Key indicating an application server that has an affinity with the user session for a given software application,
- propagating the correlation record with each transaction, allowing thereby the routing means (10, 11, 12) to read the correlation record for targeting the application servers (A2, B3, C2) that have an affinity with the user session and that process the software applications relevant to handle the transaction.

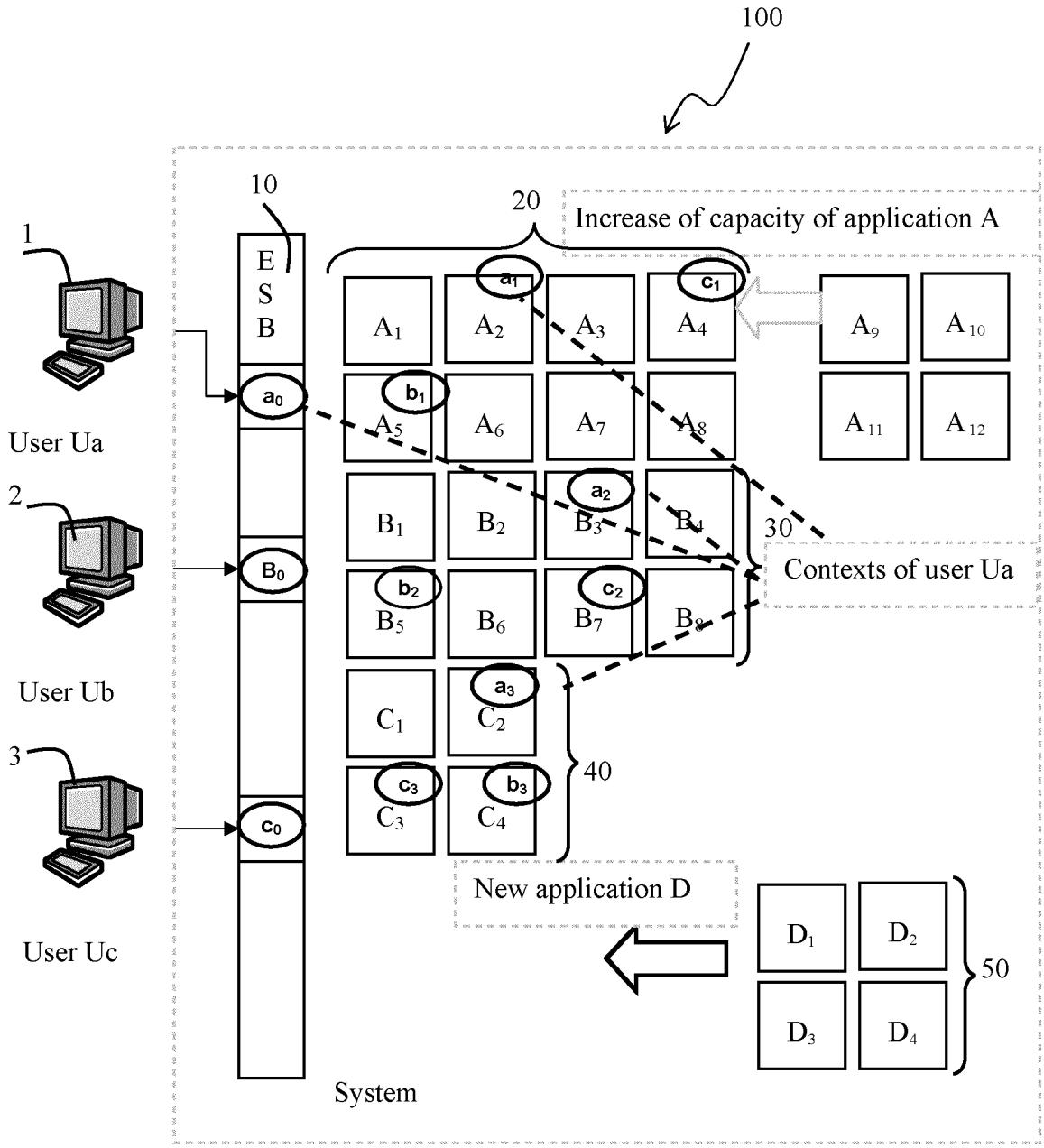
100
10
20
Increase of capacity of application A
1
User Ua
2
User Ub
3
User Uc
E
S
B
a0
B0
c0
a1
b1
c1
a2
b2
c2
a3
c3
b3
A1
A2
A3
A4
A5
A6
A7
A8
A9
A10
A11
A12
B1
B2
B3
B4
B5
B6
B7
B8
C1
C2
C3
C4
30
Contexts of user Ua
40
New application D
D1
D2
D3
D4
50
System

FIGURE 1

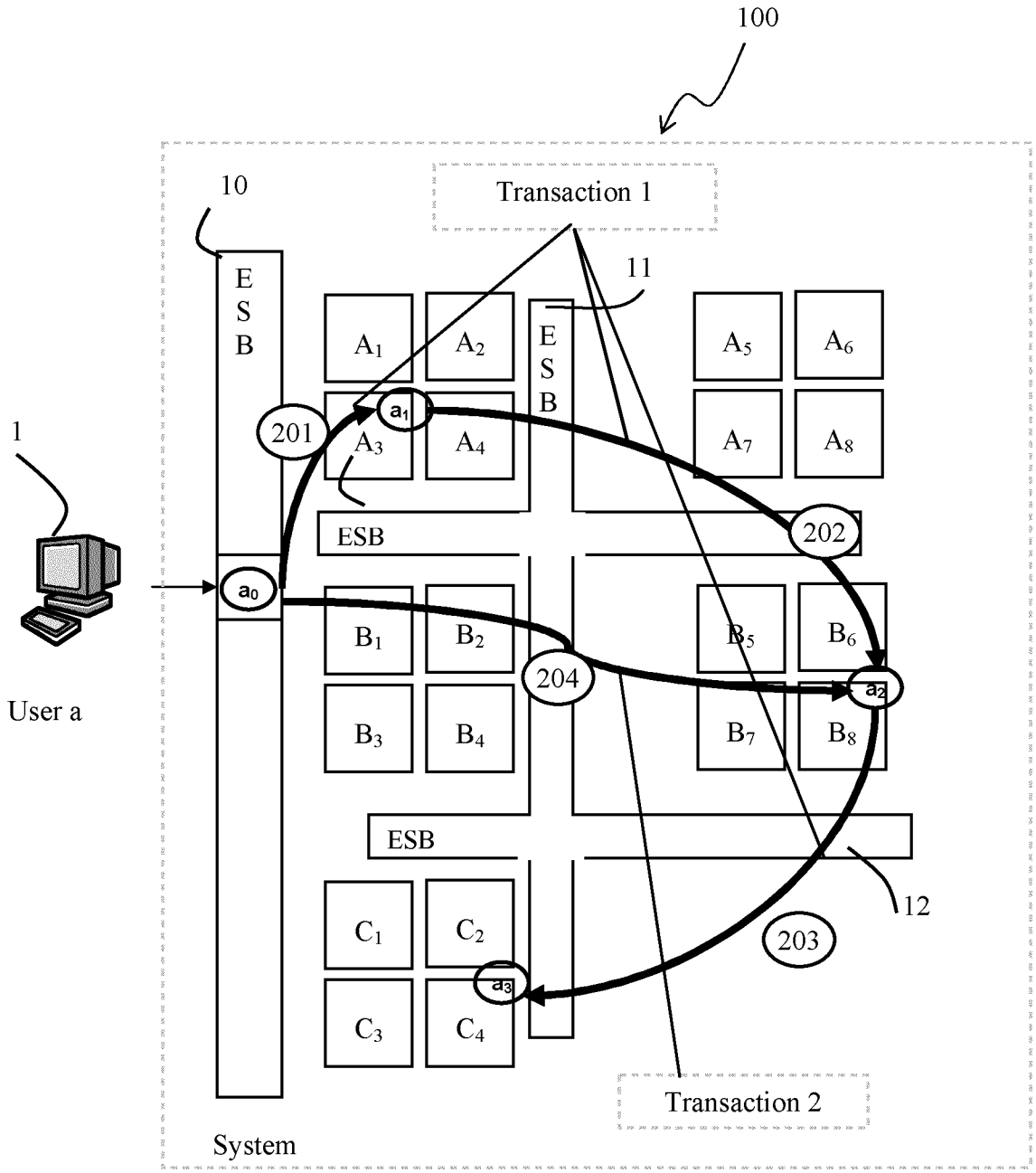
100
10
Transaction 1
11
E
S
B
A1
A2
E
S
B
A5
A6
a1
201
A3
A4
A7
A8
1
ESB
202
a0
B1
B2
B5
B6
204
a2
B3
B4
B7
B8
User a
ESB
12
C1
C2
203
a3
C3
C4
Transaction 2
System

FIGURE 2

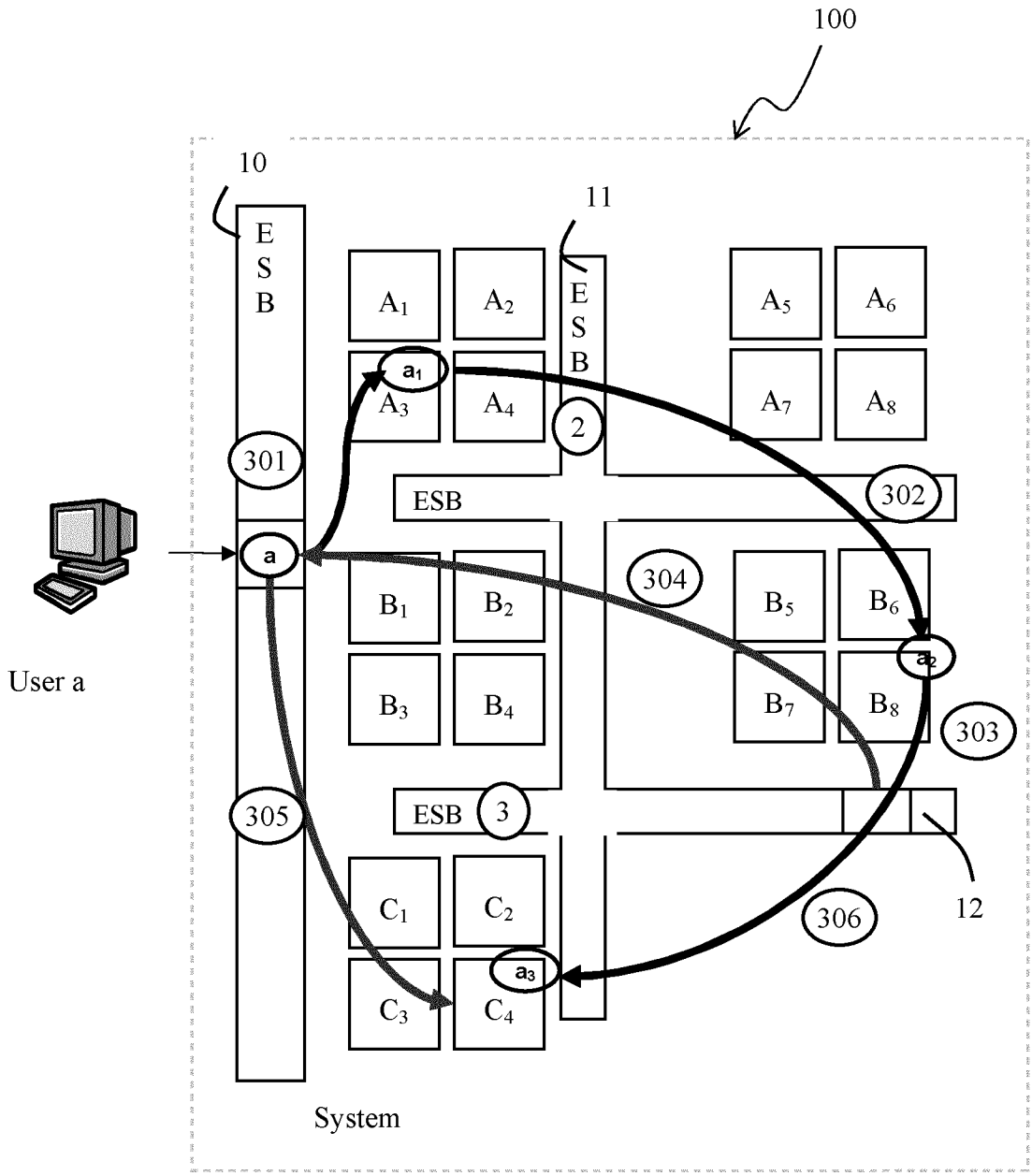
100
10
11
E S B
A1
A2
E S B
A5
A6
a1
A3
A4
2
A7
A8
301
ESB
302
a
304
B1
B2
B5
B6
a2
B3
B4
B7
B8
303
305
ESB
3
C1
C2
306
12
a3
C3
C4
User a
System

FIGURE 3

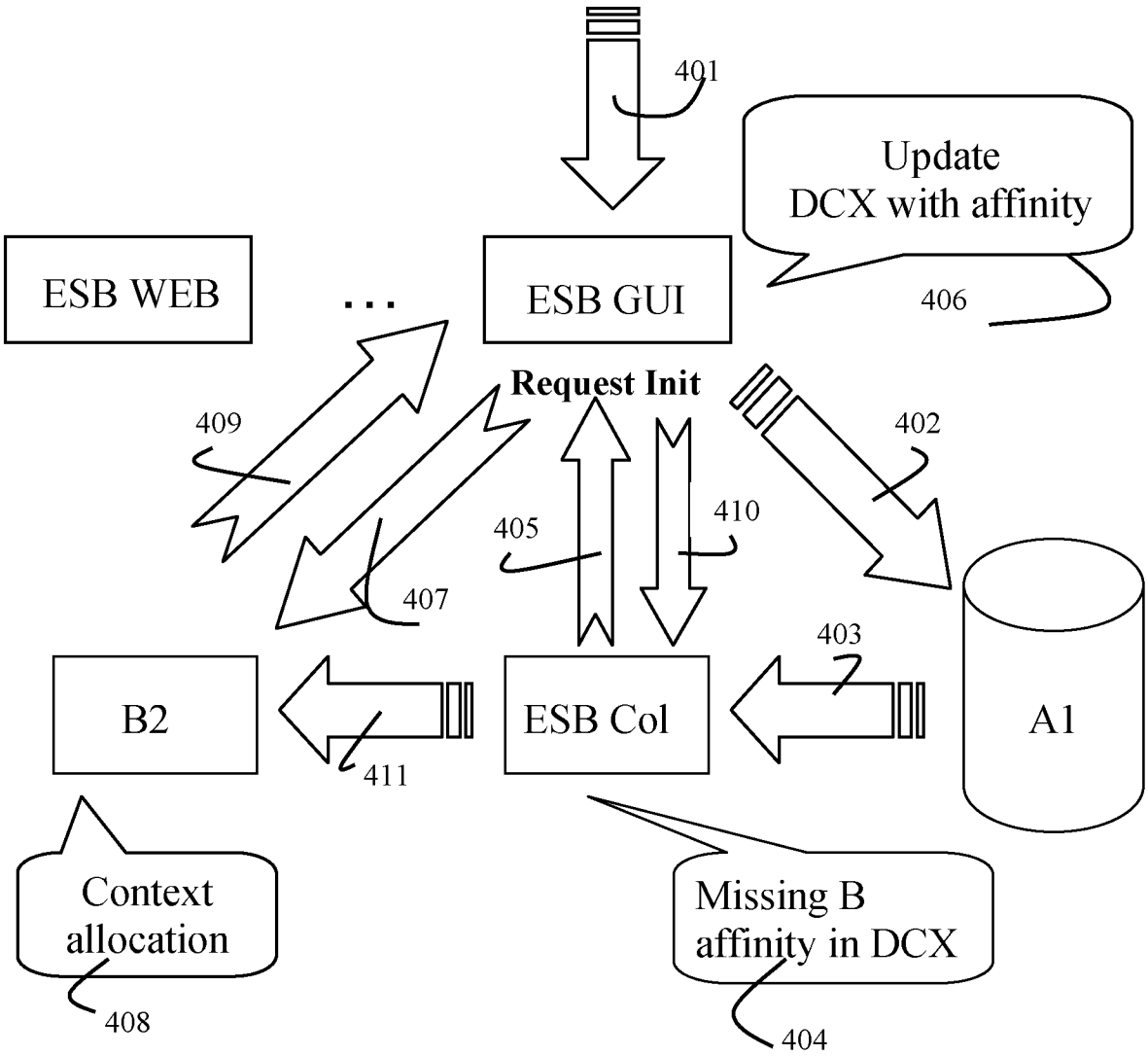
401
Update
DCX with affinity
406
ESB WEB
...
ESB GUI
Request Init
409
402
405
410
407
403
B2
ESB Col
A1
411
Context
allocation
408
Missing B
affinity in DCX
404

FIGURE 4

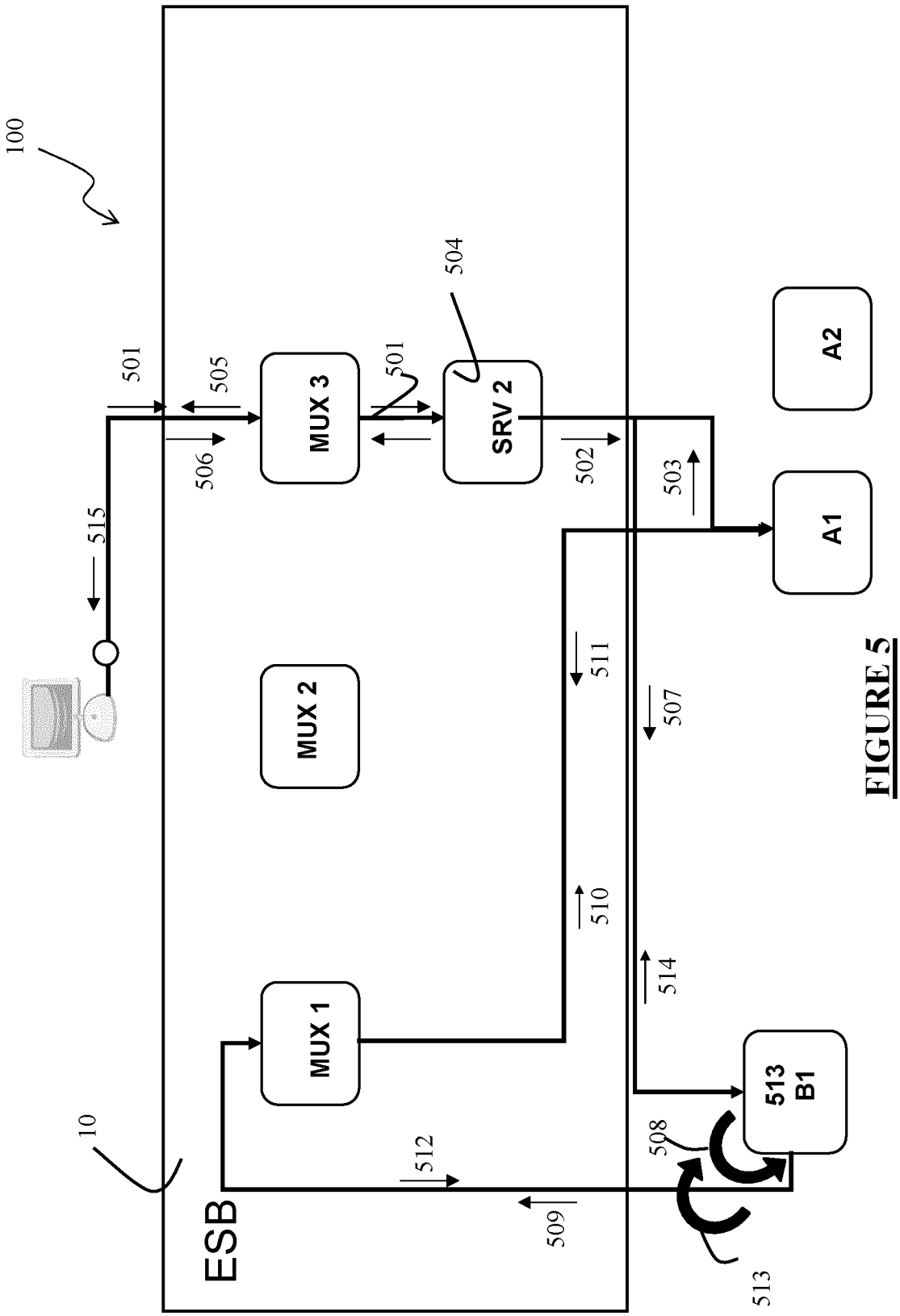
100
10
ESB
MUX 1
MUX 2
MUX 3
SRV 2
A1
A2
513
B1
501
505
506
501
504
502
503
515
511
507
510
514
508
512
509
513

FIGURE 5

INTERNATIONAL SEARCH REPORT

International application No
PCT/EP2012/054563

A. CLASSIFICATION OF SUBJECT MATTER
INV. G06Q30/06 G06Q50/14 H04L29/08
ADD.

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
G06Q H04L

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)
EPO-Internal, WPI Data

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 2006/155857 A1 (FEENAN JAMES J JR [US] ET AL) 13 July 2006 (2006-07-13) abstract figures 1-6 paragraph [0011] - paragraph [0018] paragraph [0032] - paragraph [0034] paragraph [0041] - paragraph [0059] claims 1-18 -----	1-20
X	US 2007/192492 A1 (OKAZAKI KOTARO [JP]) 16 August 2007 (2007-08-16) abstract figures 1,4-7 paragraph [0037] paragraph [0042] paragraph [0056] paragraph [0068] - paragraph [0080] ----- -/--	1-20

[X] Further documents are listed in the continuation of Box C.

[X] See patent family annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
29 March 2012	05/04/2012

Name and mailing address of the ISA/	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Fax: (+31-70) 340-3016	Berlea, Alexandru

2

Form PCT/ISA/210 (second sheet) (April 2005)

C(Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	"WebSphere Edge Server for Multiplatforms Getting Started Version 2.0", , 1 December 2001 (2001-12-01), pages 1-119, XP055004698, Retrieved from the Internet: URL:maben.homeip.net/static/cvdocs/ND20GetStart.pdf [retrieved on 2011-08-11] page 24 - page 27 page 43 - page 45	1-20
X	"WebSphere Application Server V6 Scalability and Performance Handbook", , 1 May 2005 (2005-05-01), pages 1-370, XP055004700, Retrieved from the Internet: URL:http://www.redbooks.ibm.com/redbooks/pdfs/sg246392.pdf [retrieved on 2011-08-11] page 19 - page 31 page 82 - page 88 page 110 - page 119 page 264 - page 267 page 279 - page 285	1-20

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No
PCT/EP2012/054563

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2006155857 A1	13-07-2006	NONE	
US 2007192492 A1	16-08-2007	JP 2007219608 A	30-08-2007
		US 2007192492 A1	16-08-2007