(54) Title: CLOUD SYSTEM AND METHOD WITH MOBILE SUPERCLOUD COMPUTING, DATA LINK REDIRECTION AND LAYOUT EDITING

(57) Abstract: A mobile supercloud system and method are provided that can improve upon the complexity regarding synchronization and sharing of data with others over the cloud. In another aspect, a high-performance web-based cloud services system and method by data link redirection are provided. The system and method provide ways to build, deploy, and scale online cloud-based web applications without necessarily requiring large investments in hardware infrastructure and network bandwidth. In another aspect, a system and method for page layouts using dynamic reflow are disclosed. Embodiments of the system and method provide features that can allow the user of a small mobile device to be able to quickly and easily create and/or edit page layouts for digital scrapbooks.
Published:

- with international search report (Art. 21(3))

- before the expiration of the time limit for amending the claims and to be republished in the event of receipt of amendments (Rule 48.2(h))
CLOUD SYSTEM AND METHOD WITH MOBILE SUPERCLOUD COMPUTING. DATA LINK REDIRECTION AND LAYOUT EDITING

Priority Claims/Related Applications

This application claims priority under 35 USC 120 and the benefit under 35 USC 119(e) to U.S. Provisional Patent Application Serial Nos. 62/016,602, 62/016,598 and 62/016,590 all filed on June 24, 2014, the entirety of which are incorporated herein by reference.

Field

The disclosure relates generally to cloud based systems.

Background

Today's cloud-based web applications tend to share a common architecture. That architecture is driven by the desire, on the part of the application developer, to allow the user of the web browser to interact with data objects within web pages, by tapping into the resources of powerful networks of remote computational services coupled to large data storage facilities.

Typically, a browser with an AJAX engine talks to an API interface that passes commands and data through to remote server-based code modules. Such modules routinely then access server-side databases and perform queries and computations on the database before sending computation results back for display to the user in the web page.

In these situations, the user's device acts simply as a client to the application code and centralized databases on the server side. Some systems use individual databases for each user. Such systems, however, simply provide each user access to a small individualized database, stored on the remote server as with other larger database systems.

This existing approach has been especially useful in the mobile device arena, where small handheld devices are often computationally- and resource-constrained. Existing systems provide users of such mobile devices with access to powerful remote computational and data storage resources.

In some cases, mobile users use these cloud-based applications to share files created on the local device with other users, by manipulating the files on the local device, and then uploading the files to various online sharing systems. To do this, the user typically employs an upload feature in a mobile app, to explicitly direct the app to upload the desired files, which then
become available for others to access. If updates would be needed to be made to the shared files, the user would again edit the file on the local device, and then direct the app to upload a new copy of each changed file, thereby replacing the previously-uploaded files.

The current architecture of these prior cloud systems imposes a number of limitations on what can be accomplished using these services, some of these limitations include: 1) the labor-intensive manual-upload approach utilized results in limiting the currency of data that has been shared online, due to eventual resistance by the user to provide frequent updates; 2) the cumbersome data interchange mechanisms result in inefficient synchronization between client-side and server-side of cloud-based applications; 3) current systems limit each mobile app to being able to take advantage of only one cloud service at a time; 4) even though many existing cloud-based storage resources are quite large, they are inherently limited in that they make little use of the enormous collective storage capacity resident on billions of end users' mobile devices; 5) these prior systems follow a predominantly uni-directional data flow model; and, as a result, 6) server crash-and-restore systems use server-side backup files, which don't reflect changes made to client-side data during the crash-induced server downtime period.

In summary, due to their architectural limitations, existing cloud-based web and mobile-app application architectures can deliver a less-than-optimal user experience with respect to ease of use, responsiveness, seamlessness, and reliability.

In addition, the current plethora of commercial cloud services which are available to developers of web-based and mobile-app-based applications has been a boon to the technology startup community. Whereas in previous decades a web-based software startup needed to raise considerable investment capital to merely get started, today a startup can use a cloud resource such as Amazon's® AWS to begin with relatively meager funding, yet create a system that can auto-scale to be able to handle millions of users.

The cloud service provider provides storage, virtual machines, and a set of software services to facilitate the development of large systems. This allows both startups and existing companies to quickly build and deploy large cloud services that aren't technically resource constrained, systems that can activate additional storage, CPUs, services, and bandwidth.

If such systems are properly designed, the applications can be scaled up by adding on CPUs and storage, as well as bandwidth either manually or via automatic algorithms.
If the developer has designed an appropriate economic model for the application, and if it is properly implemented, then the company may be able afford to add these expanded resources as demand increases. Unfortunately, however, businesses often do not run according to plan. When such a situation exists in the context of web applications that are expanding rapidly upon fee-based cloud services platforms, economic disaster can quickly ensue.

As a result, many companies have been driven into bankruptcy by the very popularity which they were created to seek, by implementing auto-scaling systems that grew very large very quickly due to rapidly increasing demand, but which had no way to pay for these resources due to the currently-popular trend of attempting to attract end user "eyeballs" without worrying about making money along the way. As these cloud-based applications grow, costs mount, not merely as a result of the direct costs of the services and resources themselves, but also because of the often-larger indirect cost of system development, internal infrastructure, personnel, and other consequences of implementation, deployment and support of large complex information systems.

These companies are often found to be operating under the belief that, if only they could attract enough users, then investors would follow. However, this can create a foolish race between trying to raise investment before accruing costs swamp available capital. This dynamic can create a treacherous ground upon which to plant the seeds of a new Internet-based startup.

Furthermore, an increasingly-growing number of applications exist for managing and sharing photos and other media. Many of these applications are directed towards helping users create book-like layouts of images and text passages, typically for printing. The task of manipulating the designs for layout of such pages often involves difficult multi-step processes, complicated tools, and large displays.

Because of these factors, such existing applications can be ill-suited for use on small mobile devices. Further, when implemented on touch-based devices, their complicated interfaces require concerted effort, typically using both hands, for the user to perform the various editing functions to create the desired result.

The existing systems may not be very useful for enabling the user to quickly create and edit digital scrapbook page layouts on small mobile devices, such as phones or small tablets, where one-handed operation is often desirable, and where easy-to-use interfaces are needed.
Brief Description of the Drawings

A further understanding of the nature and the advantages of particular embodiments disclosed herein may be realized by reference of the remaining portions of the specification and the attached drawings.

Figure 1 illustrates an example of a cloud services system that incorporates a mobile supercloud component;

Figure 2 illustrates data and asset flow in an example embodiment of the cloud services system when the cloud services system is a scrapbooking system;

Figure 3 illustrates an example mobile supercloud arrangement for the scrapbooking system example shown in Figure 2.

Figure 4 illustrates an example of an implementation of a web-based cloud services system that incorporates data link redirection;

Figure 5 illustrates a method for providing cloud services using data link redirection;

Figure 6 illustrates an example of a content delivery system implementation of a web-based cloud services system that incorporates data link redirection.

Figure 7 illustrates an example of content system that may utilize layout editing with dynamic reflow;

Figure 8 illustrates an example of a layout editing method with dynamic reflow;

Figure 9 shows a first image layout pattern;

Figure 10 shows a second image layout pattern;

Figure 11 shows a third image layout pattern;

Figure 12 shows a fourth image layout pattern;

Figure 13 shows a fifth image layout pattern; and

Figure 14 shows a sixth image layout pattern.

Detailed Description of One or More Embodiments

The disclosure is particularly applicable to a scrapbooking cloud services system and it is in this context that the disclosure will be described in detail. It will be appreciated, however, that the system and method has greater utility since the system may be implemented in other manners that those shown in the figures and described below that are within the scope of the disclosure and the mobile supercomputing system and method may be used with any cloud services system
such as the one shown in Figure 1. Thus, the scrapbooking cloud services system is only a representative example of the type of cloud services system that may utilize the mobile supercomputing system and method described below.

Features and embodiments presented herein can improve upon the complexity regarding synchronization and sharing of data with others over the cloud. Embodiments can provide systems that can be quickly responsive to changes made to data on the client device, and which can automatically propagate those changes wherever needed. The system can combine the resources of multiple cloud services together with the computational and storage resources resident on the mobile device. In the system, the user's device can participate in completing the restoration of state on the server after catastrophic failure on the server-side.

One embodiment provides a method comprising the following acts performed by one or more processors: identifying a first plurality of cloud computing resources in a first cloud; associating a meta-cloud with the first cloud; and receiving a signal from a client device computing device to access at least one of the plurality of computing resources by using the meta-cloud.

Another embodiment provides an apparatus comprising: one or more processors; a non-transitory storage medium including instructions executable by the one or more processors for: identifying a first plurality of cloud computing resources in a first cloud; associating a meta-cloud with the first cloud; and receiving a signal from a client device computing device to access at least one of the plurality of computing resources by using the meta-cloud.

Another embodiment provides a non-transitory storage medium including instructions executable by one or more processors for: identifying a first plurality of cloud computing resources in a first cloud; associating a meta-cloud with the first cloud; and receiving a signal from a client device computing device to access at least one of the plurality of computing resources by using the meta-cloud.

Figure 1 illustrates an example of a cloud services system 100 that incorporates a mobile supercloud component. The mobile supercloud component may allow any cloud services systems to improve synchronization and sharing of data with others over the cloud and to automatically propagate those changes wherever needed, to combine the resources of multiple cloud services together with the computational and storage resources resident on the mobile
device and to permit the user's device to participate in completing the restoration of state on the server after catastrophic failure on the server-side.

The system 100 may include one or more computing devices 102, such as 102A, 102B, ..., 102N as shown in Figure 1, that connect to and exchange data over a communications path 104 with a backend component 106 that may provide one or more cloud services. The backend component 106 may incorporate a mobile supercloud component 112 that provides the capabilities described above which are also described below in more detail.

Each computing device 102 may be a client that communicates with and uses the one or more cloud services. Each computing device 102 may be a processor based device with at least one processor, memory, persistent storage, a display and connectivity circuits that can be used by a user to connect to and interact with the backend and the one or more cloud services. For example, each computing device may be a smartphone device, such as an Apple® iPhone® or an Android® operating system based device, a tablet computer device, a personal computer device, a terminal device, a mobile device, a laptop computer device and the like. In some embodiments, each computing device 102 may store (or download) and use an application 103 that is a plurality of lines of computer code that may be executed by the processor of the computing device 102 to communicate with and interact with the one or more cloud services. The application may be a browser application, a mobile application or any other application. In one example of the system when used for scrapbooking, the application may be a Keepsayk™ app 100 or a web client 103 that is described in more detail below. The browser application may have a well-known Uniform Resource Locator (URL) address bar (address bar) that displays a URL address from which the browser application is retrieving data.

Each computing device 102 also may have one or more resources 105 that may be accessed and/or used by the system as described below. For example, each computing device may have a management resource such as component that can manage certain actions, a computational resource such as a processor, a graphics processor, a co-processor or a digital signal processor housed in the computing device and a data storage resource such as a memory or persistent storage in the computing device or a database or data store resident on the computing device. The one or more resources may further include a data file resource, network storage
resource, a virtual filesystem resource, a virtual processor/CPU resource and a stored data resource.

The communications path 104 may be a wired communications path or wireless communications path or a combination of the two. For example, the communication path 104 may be Ethernet, the Internet, a wireless data network, a wireless computer data network, a wireless cellular data network, a computer data network and the like. The communication path may use various communication protocols, such as HTTP or HTTPS as well as various data transfer protocols, such as HTML, JSON to provide the communication path between the one or more computing devices 102 and the one or more cloud services hosted by and provided by the backend 106.

The backend component 106 may include one or more cloud services components 108 (only one is shown in Figure 1 for clarity) and each cloud service component 108 may further have a web server component and an API server component 110 wherein the components of the cloud services component 108 communicate with each computing device 102 and provide the cloud service to each computing device 102. The backend component and its components may be implemented using one or more computing resources such as server computers, processors, memory, persistent storage components, blade computers and the like. Each of the components 108, 110 and 112 may be implemented in hardware or software. When a component is implemented in software, the component may be a plurality of lines of computer code that may be executed by a processor so that the processor is configured to perform the operations and functions of those components as described below. When a component is implemented in hardware, the component is a circuit, such as a programmable logic device, microcontroller, application specific integrated circuit and the like, that performs the operations and functions of those components as described below.

The system 100 may also have one or more other clouds 114, 116 that are also capable of being coupled to the communication path 104. Each cloud 114, 116 may further comprise one or more cloud resources 114A, ..., 114N and 116A, ... 116N that may be accessed by the system 100 as described below. The one or more cloud resources may include a management resource such as component that can manage certain actions, a computational resource such as a processor, a graphics processor, a co-processor or a digital signal processor and/or a data storage
resource such as a memory or persistent storage in a computer or a database or data store capable of being coupled to a computer. The one or more cloud resources may further include a data file resource, network storage resource, a virtual filesystem resource, a virtual processor/CPU resource and a stored data resource.

The web server component and API server component 110 may receive data requests for each computing device (and more specifically each application 103 of each computing device) and exchange data with each computing device as described below with reference to Figure 2 and respond to the data requests and provide code and data identifiers for the requested data. The mobile supercloud component 112, that may be resident on the backend component 106 and resident in each computing device 102 may implement the mobile supercloud within the cloud service system 100 in Figure 1.

In one embodiment of a Mobile Supercloud (MSC), the system can extend the "cloud" to a client's device (computing devices 102) so that the computing device's computational and storage resources can act as an integrated extension to the cloud-based application environment of the system 100. Thus, it can treat the computing device 102 as a fully-functional computational and storage peer in a distributed application system. The system may use a network API to extend the cloud. The listing of the resources for each element of the extended cloud, including resources on the computing device, may be part of the network API. The system may also use a unique identifier and a shared secret for each resource as a technique for access control to each resource. The MSC 112 puts an abstracted layer over cloud resources so a particular Supercloud system is not restricted to any individual third-party cloud provider in any way or fashion. Interfacing with each cloud services provider requires just one interface module that may include an API that permits the MSC to access each cloud resource. For example, the MSC may have an integrated handler for each cloud service and a valid set of credentials from the current user (provided to the handler) to enable the MSC resource access for the user.

The MSC can require very little interface complexity on the part of the cloud services provider. As long as there's an API to the cloud resource that lets MSC check the file attributes (time and date) and that the MSC system can write to (and can get a callback) the MSC can sit on top of anyone's existing cloud service. This means there can potentially be an unlimited number of cloud services/resources that can be added to the system over time. When more than one
cloud services system is integrated in this way, the MSC merges them into a single collective resource that the app 103 can utilize in providing the various facilities which it provides to the user. The multi-cloud integration capabilities of MSC create a meta-cloud, what might be described as a "cloud of clouds." Additionally, the meta-cloud can be combined with the ability of the MSC to also include the computational, data management and data storage capacity of the end user's computing device 102. Using the API and the interface, the MSC 112 may then associate a plurality of cloud resources with the meta-cloud and thus permit the app 103 to access/utilize any of the cloud resources that are part of the meta-cloud using the interface/API. In a particular example of the system shown in Figure 3, the third party cloud service/resource may be a storage resource provided by Dropbox® that is part of the MSC. A Dropbox Sync API provides a virtual drive and a few callback functions to the app developer.

The MSC 112 can make use of individual end-user databases, with eventual consistency between an SQLite database resident on each computing device 102 and a corresponding eventually-mirrored database 114A on the server-side since both are resources that may be part of the meta-cloud. This approach can make the resident database on the computing device 102 an integral part of the distributed MSC application system. Thus, a user uses the app 103 and accesses the local SQLite database on the computing device 102 repeatedly and changes the content in the local SQLite database. When this occurs, for example, when the user makes a change to a piece of content, the change may be reflected back to the storage resources of the system and to the user's personal copy of the database on the backend 106 or coupled to the backend. An advantage of this approach is that, if the primary MSC storage resource, such as a database server external to the computing device 102, goes down, the local database, that may be within the app 103, can restore the primary MSC storage resource and bring the primary MSC storage resource up to date, insuring that the user can continue to work without interruption and have all of his or her changes immediately reflected in the system. Furthermore, the restore capability can be bi-directional, as well so that, if the computing device 102 is ever lost, stolen, or damaged, the MSC architecture allows for the easy automatic restoration of the device-resident database to its previous state. The above restoring is made possible since each app 103 may communicate with the backend 106 (using APIs and interfaces). This approach provides the user with the functionality of a personal database in the cloud, with seamless and instant
reflection of data changes from the mobile device to the server, while also providing the redundancy, backup, and fail-over advantages more typical of enterprise-class database systems. As a result of this design, the computing device can become a single point of reference to what the data state of the master copy of the database should be.

Now, an example of the scrapbooking cloud system that incorporates the MSC and the data and asset flow in the scrapbooking cloud system is described in more detail. In the scrapbooking cloud system, the app 103 may be a "The Keepsayk™ app". In this example, the app 103 may display the "preparing to upload" message to the user. When that occurs, the MSC steps through each of the user's scrapbook media asset files, determines if the file needs to be written to Dropbox, uploads the file to Dropbox if the determination is positive, and initiates the eventual synchronization of the client database with its counterpart on the MSC database server. For example, the Keepsayk app 103 on each computing device 102 may access its local SQLite database repeatedly during the user's operation of the mobile app. When this occurs, for example, when the user makes a change to a publicly-shared scrapbook, the change is reflected back to the server to the user's personal copy of the database on the server.

Figure 2 illustrates data and asset flow in an example embodiment of the cloud services system when the cloud services system is a scrapbooking system. The top diagram shows the app-to-cloud interface, while the bottom diagram shows the webview-to-cloud interface. In the (top) app-to-cloud interface diagram of Figure 2, it is shown that the following processes occur as the Keepsayk mobile app 103 communicates with a storage resource 114N, such as a database server.

1) The Keepsayk app 103 on the computing device 102 sends pieces of data, such as layout data, caption and text post data, and asset URLs for each piece of data to the node/web server 110. The asset URLs identify the location of each piece of data since those pieces of data may reside on any of the storage resources, such as DropBox as shown in Figure 2, that are part of the meta-cloud.

2) The app 103 also uploads photo and video asset files to the end-user's account on the storage resource 114N, such as Dropbox.

3) the node/web server 110 may update the user's database tables on the storage
resource of the system 114N, such as a database server, via eventual consistency synchronization as described above.

In the bottom diagram of Figure 2, the data flow in the webview-to-cloud interface is illustrated:

1) the web client 103 sends a web page request to the node/web server 110 for a web view of the scrapbook;
2) the node/web server 110 makes an sql call via the webapi module to the storage resource 114N, such as a database server as shown in Figure 2, which then returns the sql query results to the node/web server 110;
3) the node/web server 110 uses the returned data to build json code for the scrapbook layout;
4) the node/web server 110 returns the scrapbook layout as json code and Keepsayk asset ID data URLs to the web browser;
5) the web browser 103 loads images by requesting Keepsayk asset IDs via URL request to the node/web server 110;
6) the node/web server 110 responds for each asset ID request with a 301 redirect pointing directly to the Dropbox location of the asset; and
7) The web browser 103 loads each image or video asset directly from Dropbox.

The use of 301 redirects shown in step 6 allows for the deployment of efficient web services for a vast number of services without the performance impact and infrastructure requirements that other typical media-rich web-based cloud applications have needed. When this approach is then put together with the layering of multiple third-party cloud services a synergistic combinations results. The data-link-redirection design technology is described in more detail in co-pending, commonly owned patent application serial number XX/XXX,XXX filed on the same day herewith which is incorporated herein by reference. The data-link-redirection allows the MSC to scale the amount of accessible storage without the consequential complexity of having to manage the storage and bandwidth directly, system growth without friction.
The mapping of each resource by the MSC may be performed by a mapping component of the MSC, that may be a plug-in, that handles one particular cloud vendors’ API and protocol. Thus, the MSC may have a plurality of mapping components that handle each API/protocol. These mapping components of the MSC may be executed on the backend component 106.

Figure 3 illustrates an example mobile supercloud arrangement for the scrapbooking system example shown in Figure 2. In Figure 3, a long dashed line (-----) shows the database-related communications traffic, a dotted line (- - -) shows web-based communications traffic and a shorter dashed line (-- -- --) shows image and/or video asset file transfer traffic. In this example, the system may have a plurality of nodes 110 as shown.

In the example, the app 103, such as a Keepsayk (KS) app in this example, on each computing device 102 communicate with the nodes 110, that may be node servers in this example. These nodes 110 are extremely lightweight hybrid servers, which act as either database/API servers or as very lightweight web servers, depending upon the context of the client making a request to the server, as described in the discussion regarding Figure 2 above. The node servers 110, in turn send sql commands to the storage resource 114N, such as a database server in this example, which respond with table data resulting from the execution of the sql commands. The storage resource 114N is backed up via ZFS replication over the local LAN through to a secondary local server/router 300, and is also replicated via ZFS replication over cryptographically-secure ssh channels to a tertiary offsite remote server. If the primary storage resource 114N ever goes down, the secondary server takes over, and is brought up to date by the device-resident personal database, for each Keepsayk user. The Keepsayk app 103 also communicates directly to the third-party-cloud service, in this example Dropbox, to upload photo and video media assets.

For shared scrapbooks, the web client 103 may make a scrapbook web page request to the node server 110, which then sends an sql command to the database server 114N. The database server 114N responds with table data which is used to feed a json generator module within the node server 110. This json data is then sent back to the web client 103, and is interpreted into Javascript-accessible data on the web client 103. The web client 103 then issues URL requests to the node server 110 for the photo and video image assets required for the desired scrapbook, the node server 110 then responds by sending the web client 301 redirects pointing the web clients
directly to the third-party cloud service URLs corresponding to the locations on, for example, Dropbox, for those media assets. Load among the node servers is balanced through the use of a round-robin DNS facility. Further, web caching services, such as Cloudflare, are also utilized to improve the large-scale performance of the web communications.

In one embodiment, nothing in the central core of the MSC need communicate directly with the third-party cloud service. Only the mobile apps and the web clients ever need to communicate with those services. The Keepsayk MSC can extend the concept of cloud-based web applications to include the user's mobile device as a seamless part of the distributed cloud-based application and datastore. Further, it can allow a plurality of third-party cloud services to be multiplexed into extremely large media storage and transmission capability which can be transparently utilized by both the Keepsayk apps and web clients to provide high-performance rich interactive experiences to a virtually unlimited number of users.

It also can extend these third-party resources by integrating the user's own local device resources into the cloud. In this way, it allows the creation of a new generation of cloud-based applications that can tap into the enormous collective resources of billions of end-users' mobile devices to provide application functionality that gives a seamless continuity between user edits and updates to data and representations of data that have been shared online for widespread dissemination.

Data Link Redirection

In another aspect of the system, the disclosure is particularly applicable to a web-based cloud service scrap-booking system that uses data link redirection and it is in this context that the disclosure will be described. It will be appreciated, however, that the system and method has greater utility since it may be used with various different cloud services and may be used to provide ways to build, deploy, and scale online cloud-based web applications or cloud services without necessarily requiring large investments in hardware infrastructure and network bandwidth. For example, the system may be used with any cloud service that requires a significant amount of data storage such as various content delivery systems and the like.

In one embodiment, a method is provided that causes execution of an application on a client device, detection of a request from the client device to access a resource and using the application to redirect a link to provide the resource to the client device. Another embodiment
provides an apparatus that has one or more processors and a non-transitory medium including
instructions executable by the one or more processors for: causing execution of an application on
a client device; detecting a request from the client device to access a resource; and using the
application to redirect a link to provide the resource to the client device. In another embodiment,
a non-transitory medium is provided that includes instructions for execution by one or more
processors for: causing execution of an application on a client device; detecting a request from
the client device to access a resource; and using the application to redirect a link to provide the
resource to the client device.

Figure 4 illustrates an example of an implementation of a web-based cloud services
system 4100 that incorporates data link redirection. A data link redirection component provides
ways to build, deploy, and scale online cloud-based web applications/cloud services without
necessarily requiring large investments in hardware infrastructure and network bandwidth. The
system 4100 may include one or more computing devices 4102, such as 4102A, 4102B, ...,
4102N as shown in Figure 1, that connect to and exchange data over a communications path
4104 with a backend component 4106 that may provide one or more cloud services. The
backend component 4106 may incorporate a data link redirection component 4113 that provide
ways to build, deploy, and scale online cloud-based web applications or cloud services without
necessarily requiring large investments in hardware infrastructure and network bandwidth as
described below in more detail.

Each computing device 4102 may be a client that communicates with and uses the one or
more cloud services. Each computing device 4102 may be a processor based device with at least
one processor, memory, persistent storage, a display and connectivity circuits that can be used by
a user to connect to and interact with the backend and the one or more cloud services. For
example, each computing device may be a smartphone device, such as an Apple® iPhone® or an
Android® operating system based device, a tablet computer device, a personal computer device,
a terminal device, a laptop computer device and the like. In some embodiments, each computing
device 4102 may store (or download) and use an application 4103 that is a plurality of lines of
computer code that may be executed by the processor of the computing device 4102 to
communicate with and interact with the one or more cloud services. The application may be a
browser application, a mobile application or any other application. In one example of the system
when used for scrapbooking, the application may be a Keepsayk™ app that is described in more detail below. The browser application may have a well-known Uniform Resource Locator (URL) address bar (address bar) that displays a URL address from which the browser application is retrieving data.

The communications path 4104 may be a wired communications path or wireless communications path or a combination of the two. For example, the communication path 4104 may be Ethernet, a wireless data network, a wireless computer data network, a wireless cellular data network, a computer data network and the like. The communication path may use various communication protocols, such as HTTP or HTTPS as well as various data transfer protocols, such as HTML, JSON to provide the communication path between the one or more computing devices 4102 and the one or more cloud services hosted by and provided by the backend 4106.

The backend component 4106 may include one or more cloud services components 4108 (only one is shown in Figure 1 for clarity) and each cloud service component 4108 may further have a web server component 4110 and an API server component 4112 wherein the components of the cloud services component 4108 communicate with each computing device 4102 and provide the cloud service to each computing device 4102. The cloud services component 4108 or each app 4103 on each computing device may further comprise a data link redirection component 4113. The backend component and its components may be implemented using one or more computing resources such as server computers, processors, memory, persistent storage components, blade computers and the like. Each of the components 4108, 4110, 4112, 4113 may be implemented in hardware or software. When a component is implemented in software, the component may be a plurality of lines of computer code that may be executed by a processor so that the processor is configured to perform the operations and functions of those components as described below. When a component is implemented in hardware, the component is a circuit, such as a programmable logic device, microcontroller, application specific integrated circuit and the like, that performs the operations and functions of those components as described below.

The backend component 4106 may be capable of connecting with and exchanging data with one or more other systems. For example, the backend component may be coupled to a data storage system 4114 that may be a database server and a third party data storage system 4116 as shown. The third party data storage system 4116 and the data link redirection component allows
the system to build, deploy, and scale online cloud-based web applications/cloud services
without necessarily requiring large investments in hardware infrastructure and network
bandwidth.

The web server component 4110 may receive data requests for each computing device
(and more specifically each application 4103 of each computing device) and exchange data with
each computing device as described below with reference to Figure 5. The API server
component 4112 may respond to the data requests and provide code and data identifiers for the
requested data. The data link redirection component 4113 receives the data identifiers for the
requested data and then generates a redirect as described below so that the data request is
redirected to the third party data storage system 4116 as described below with reference to Figure
5.

Figure 5 illustrates a method 4200 for providing cloud services using data link
redirection. The method and processes in Figure 5 may be implemented by the components of
the cloud services component 4108 in Figure 1 for example, but may also be implemented using
other hardware and software components. In the method, a client (implemented using the
application 4103 on one of the computing device 4102 in Figure 1) may request an app or code
(4202) in order to interact with a particular cloud service and the app or code may be downloaded
to the particular computing device. The app or code may then be used to request additional code
and one or more data asset identifiers (4204) for one or more data assets requested by the client
from the cloud services component. Each data asset may be a type of digital content such as an
image, a video, text, audio and the like. Each data asset identifiers may identify a location for
each requested data asset so that the data asset may be retrieved and downloaded by the client.

In the method, data link redirections may be generated for each data asset based on the
data asset identifier (4206) for the data asset. In one embodiment, the generation of the data link
redirections may be performed by the data link redirection component in Figure 4. In one
implementation, the method may use an HTTP 301 redirect. In a typical redirect to a remote
resource, there may be a web client request and a web server response as follows:

GET /asset/201.lrg.jpg
Host: scrapbook.keepsayk.com
And the server would respond with:

HTTP/ 1.1 301 Moved Permanently
Location: https://dl.dropboxusercontent.com/s/kvvky4a5v68s20o/5. 141.jpg

As shown, in the prior systems, the "Status" response is typically "301 Moved Permanently". This results in the browser not only being redirected to the new location, but the web client browser address bar will reflect the new address. In order to make the redirected asset appear as if it is from the originating domain, such as scrapbook.keepsayk.com, the method substitutes the original asset URL in place of the "Moved Permanently" response.

Thus, the server response in the method of Figure 5, for example, looks like:

HTTP/1.1 301 /asset/201.lrg.jpg
Location: https://dl.dropboxusercontent.com/s/kvvky4a5v68s20o/5.141.jpg

With this response, the web browser's URL address bar continues to indicate the originating domain, such as keepsayk.com URL, so the data served in response to the redirected image asset request appears to have been served by the keepsayk.com server, while it actually has been served by the end-user's third-party cloud hosting server.

Returning to Figure 5, the client may then load the one or more data assets from the remote system (4208) using the data link redirection.

Now, an example of a scrapbooking cloud service implementation of a web-based cloud services system that incorporates data link redirection as shown in Figure 6 will be described. In the implementation shown in Figure 6, each computing device may execute a "Keepsayk™ app" that may be a Keepsayk mobile iOS® app for the creation, editing, viewing and sharing of instant digital scrapbooks that combine photos, videos, text posts and captions into automatically-formatted beautiful presentations.

The backend component 4106 shown in Figure 4 may be implemented as the Keepsayk Mobile Supercloud (MSC), which expands the capabilities of the system beyond anything that was possible before in the mobile arena. The MSC allows the combination of multiple third-party cloud services into a single pool of resources available to the user for storing and sharing scrapbooks containing rich digital media. To do this, the MSC puts an abstracted layer over cloud storage, so Keepsayk's Supercloud system is not restricted to any individual third-party cloud provider in any way or fashion. Interfacing with each cloud services provider requires just
one interface module. This means there can potentially be an unlimited number of cloud services that can be added to the system over time. The multi-cloud integration capabilities of MSC create a meta-cloud, what might be described as a "cloud of clouds." That capability, when combined with the ability of the MSC to also include the computational, data management and data storage capacity of the end user's mobile device, is what puts the "super" into the Mobile Supercloud approach.

Unlike previous approaches to cloud services scalability, where system growth often results in at least geometric increases in both direct and indirect costs relating to storage, CPUs, bandwidth, system development, internal infrastructure, personnel, and other consequences of implementation, deployment and support, a Keepsayk Pointe System (KPS) has been developed that is a novel approach to internal system design and integration with external systems. The KPS facilitates the creation of systems which can be expanded rapidly without the friction and complexity of previous approaches.

A good example of the user of the KPS system is within the Keepsayk MSC, which incorporates a web-based viewer for instant scrapbooks, where the HTML5 and CSS code for the scrapbook layout are located on the Keepsayk system servers (an implementation of the cloud services component 4108 in Figure 1), together with image caption and text asset information, but where the media data assets, photo images and video files, are located on a commercial cloud service (an implementation of the third party data storage system 4116 in Figure 1), such as Dropbox®. In order to create a unified experience for the end user, the user's web browser needs to appear to be pulling all of the data necessary to render each scrapbook from the Keepsayk server, however this would typically require that the Keepsayk server, at the very least, act as a proxy to Dropbox, first downloading images to Keepsayk before passing them on to the user. KPS avoids this requirement.

Rather than downloading images to the Keepsayk server, and rather than incorporating explicit links to media assets on the Dropbox server, KPS makes use of the web's existing mechanism for handling a dereferenced Uniform Resource Locator (URL) in order to provide a masked indirect link to the asset residing on the Dropbox servers, while the user only sees links that appear local to the Keepsayk system.

This provides high-performance access to those remotely-stored media assets, while also
providing referential integrity to the Keepsayk system designer, and therefore the ability to create
stable web interface designs, together with link reassignment and maintenance capabilities, while
simultaneously maintaining the ability to provide Keepsayk and its users' link and asset privacy.
In one embodiment, the KPS performs this feat by taking the approach of choreographing data by
reference.

In dance, a fully extended vertical foot is said to be en pointe when its tip is touching the
floor, even when not bearing weight. Similarly, the Keepsayk Pointe System allows web-based
cloud applications to be fully extended to extraordinary scales, without having to support
anything even approaching the weight of those scaled-up systems. Only the tip, the interface of
the system, needs to be manipulated to virtually choreograph the data by indirect reference.

By the elegant manipulation of the web's 301-redirect mechanism, the Keepsayk
node/web servers offload the heavy lifting of serving large volumes of rich media files to third-
party cloud services, rather than having to directly communicate and handle that data within the
core of the MSC. An illustration of the surprising results from taking this KPS approach can be
seen in the exclamation of the Keepsayk developer who created the Pointe System upon first
witnessing the efficiencies of this approach in action: "OMG, it freaking works."

KPS-based systems can still take advantage of Web-based caching services, to even
further optimize response under heavy usage. Cloudflare (shown in Figure 6), for example will
cache the redirects just as they cache other web traffic, so repeated accesses of a given media
asset don't have to reach the Keepsayk servers. Even without services such as Cloudflare, in
many cases, the third-party cloud service becomes a cache analogue, since the redirects result in
time-limited URLs that do not immediately expire.

By de-referencing the original URL requests from the users' browsers, and keeping those
from the visibility of the end users, those end users are never able to see the original URL. This
provides the ability to maintain link privacy for the end user's media asset data files, resident on
the third-party cloud server, while sharing scrapbooks containing those media files. The KPS
hides the URL, issues the 301 redirect, and then points the end user at the time-limited URL.
The example for link privacy has been described above.

This approach also provides an advantage in the eventuality that Keepsayk ever receives a
takedown request for a particular scrapbook media element. Using KPS, Keepsayk can easily
respond to that request, because the original URL is hidden. All Keepsayk needs to do in such a situation is to remove (take down) the internal KPS representation of the URL link, and then invalidate the cache, thereby destroying the relevant redirect. Keepsayk can thereby remove that media element from the logical scrapbook structure without ever having to touch the original data asset file on the user's third-party cloud storage account.

In the example implementation shown in Figure 6, the data flow process may include:

1) the web client sends a web page request to the node/web server for a web view of the scrapbook;
2) the node/web server makes an sql call via the webapi module to the database server, which then returns the sql query results to the node/web server;
3) the node/web server uses the returned data to build json code for the scrapbook layout;
4) the node/web server returns the scrapbook layout as json code and Keepsayk asset ID data URLs to the web browser;
5) the web browser loads images by requesting Keepsayk asset IDs via URL request to the node/web server;
6) the node/web server responds for each asset ID request with a 301 redirect pointing directly to the Dropbox location of the asset; and
7) The web browser loads each image or video asset directly from Dropbox.

The use of 301 redirects shown in step 6 allows for the deployment of efficient web services for a vast number of services without the performance impact and infrastructure requirements that other typical media-rich web-based cloud applications have needed. When this approach is then put together with the layering of multiple third-party cloud services, a synergistic combination results. The data-link-redirection design technology allows Keepsayk's MSC to scale the amount of accessible storage without the consequential complexity of having to manage the storage and bandwidth directly, resulting in the potential for virtually unlimited system growth without friction.

The Keepsayk Pointe System provides an extremely efficient way to build, deploy, and scale online cloud-based web applications without requiring large investments in hardware
infrastructure and network bandwidth, thereby making it possible for small entities with very little in the way of capital resources to nevertheless be able to design, build, deploy and support web-based cloud applications of a large enough scale to support many millions of users at very low cost.

5 Layout Editing

In another aspect of the system, one embodiment allows the user generate layout designs on small portable electronic devices such as Apple® iPhone® and iPod® Touch. Although specific devices and user input mechanisms may be described, it should be apparent that embodiments can be implemented on other computing devices and with alternate input mechanisms. For example, the computing device may be mobile phones, tablets, laptops, phablets, music players, portable game devices, etc., may be used. Similarly, touch screens, gesture recognition, voice commands, and other forms of user input may be employed. In general, any type of suitable platform can be used using any suitable hardware or software.

The system and method provide features that can allow the user of a computing device, such as a small mobile device, to be able to quickly and easily create and/or edit page layouts for digital books of content having a plurality of pages, such as digital scrapbooks. The system may also allow the books to be shared directly from the computing device after creation.

One embodiment provides a method for manipulating images, the method comprising the following acts performed by one or more processors: accepting a signal from a user input device to select a page layout; displaying a first group of images using the selected page layout; and displaying a second group of images using a different page layout in response to the selection of the selected page layout.

Another embodiment provides an apparatus for manipulating images, the apparatus comprising: one or more processors coupled to a non-transitory medium including instructions executable by the processors for: accepting a signal from a user input device to select a page layout; displaying a first group of images using the selected page layout; and displaying a second group of images using a different page layout in response to the selection of the selected page layout.

Another embodiment provides a non-transitory medium including instructions executable by one or more processors for: accepting a signal from a user input device to select a page layout;
displaying a first group of images using the selected page layout; and displaying a second group of images using a different page layout in response to the selection of the selected page layout. Now, an example of an implementation of a system that provide layout editing with dynamic reflow is described.

Figure 7 illustrates an example of content system 7100 that may utilize layout editing with dynamic reflow. The system 7100 may include one or more computing devices 7102, such as 7102A, 7102B, ..., 7102N as shown in Figure 7, that connect to and exchange data over a communications path 7104 with a backend component 7106 that may provide one or more cloud services including a content service. Each computing device 7102 may have an application 7103 resident on the computing device that is executed by a processor of the computing device to interact and exchange data with the backend component 7106. The application 7103 may be a browser application, a mobile application or an application downloaded from the backend component 7106. The application 7103 may include a dynamic reflow component 7103A that allow the user of a computing device, such as a small mobile device, to be able to quickly and easily create and/or edit page layouts for books of content having a plurality of pages, such as digital scrapbooks as described below in more detail.

Each computing device 7102 may be a client that communicates with and uses the one or more cloud services, such as a content service 7108. Each computing device 7102 may be a processor based device with at least one processor, memory, persistent storage, a display and connectivity circuits that can be used by a user to connect to and interact with the backend and the one or more cloud services. For example, each computing device may be a smartphone device, such as an Apple® iPhone®, or an Android® operating system based device, a mobile device, a tablet computer device, a personal computer device, a terminal device, a laptop computer device and the like. The application 7103 may be a plurality of lines of computer code that may be executed by the processor of the computing device 7102. In one example of the system when used for a digital scrapbooking cloud service, the application may be a Keepsayk™ app that is described in more detail below.

The communications path 7104 may be a wired communications path or wireless communications path or a combination of the two. For example, the communication path 7104 may be Ethernet, a wireless data network, a wireless computer data network, a wireless cellular
data network, a computer data network and the like. The communication path may use various communication protocols, such as HTTP or HTTPS as well as various data transfer protocols, such as HTML, JSON to provide the communication path between the one or more computing devices 7102 and the one or more cloud services hosted by and provided by the backend 7106.

The backend component 7106 may include a digital book cloud services component 7108 and the digital book cloud service component 7108 may further have a web server/API server component 7110 wherein the components of the content cloud services component 7108 communicate with each computing device 7102 and provide the book cloud service to each computing device 7102. For example, the digital book cloud services component 7108 may electronically distribute digital books generated by a user, manage the storage of the digital content for the digital books as well as the assembled digital books, and store the digital content for the digital books as well as the assembled digital books. The backend component and its components may be implemented using one or more computing resources such as server computers, processors, memory, persistent storage components, blade computers and the like.

Each of the components 7108, 7110 may be implemented in hardware or software. When a component is implemented in software, the component may be a plurality of lines of computer code that may be executed by a processor so that the processor is configured to perform the operations and functions of those components as described below. When a component is implemented in hardware, the component is a circuit, such as a programmable logic device, microcontroller, application specific integrated circuit and the like, that performs the operations and functions of those components as described below.

The backend component 7106 may be capable of connecting with and exchanging data with one or more other systems. For example, the backend component may be coupled to a data storage system 7114 that may be a database server and a third party data storage system 7116 as shown. The third party data storage system 7116 allows the system to build, deploy, and scale online cloud-based web applications/cloud services without necessarily requiring large investments in hardware infrastructure and network bandwidth.

The web/API server component 7110 may receive data requests for each computing device (and more specifically each application 7103 of each computing device) and exchange data with each computing device as described below with reference to Figure 8. The web/API
server component 7110 also may respond to the data requests and provide code and data identifiers for the requested data.

Figure 8 illustrates an example of a layout editing method 7200 with dynamic reflow. In one embodiment, the method shown in Figure 8 may be implemented by the dynamic reflow component 7103A and the application 7103 on each computing device. Specifically, the processor of each computing device may execute the plurality of instructions of the dynamic reflow component 7103A so that the processor is configured to perform the processes shown in Figure 8. The method may also be performed in other manners that are within the scope of the disclosure.

In the method, a page layout is displayed (7202) on the computing device. Examples of the page layout display are shown in Figures 9-14. The page layout may be for a book, such as a digital scrapbook. Although the examples of the page layouts show images, the system and method more generally may be used for any type of media or digital content, such as video, text, document, slide, etc. As shown in Figure 9, the page layout may include a command portion 7300 that allows the user to select between different functions of the application 7103 including the page layout. The page layout further includes a layout display portion 7302 that shows the current page layout for the current page and one or more subsequent pages and a layout schematic portion 7304 that displays the types of different layouts that may be selected by the user.

In the application 7103, when a user taps on content in the layout display portion 7302 or the layout schematic portion 7304, the layout for the current page may be changed by the application and the computing device receives an input indication of the change in layout (7204). In the method, the page layout may move through a sequence of layouts, one for each tap, for that page. Thus, for example, when the first tap (and input indication to the computing device) the page layout may transition from a first layout (shown in Figure 9) to a second layout (shown in Figure 10.)

In one example, the page layouts may be internally numbered 1 to 5. Layout One is a single landscape image (Figure 9). Layout Two contains two side-by-side portrait images (Figure 10). Layout Three positions one portrait image on the left and two quarter-page-size landscape images on the right (Figure 11). Layout Four positions two landscape images on the left and one
portrait image on the right (Figure 12). And Layout Five positions four quarter-page-sized
landscape images on the page, in a 2x2 grid pattern (Figure 13). Tapping again causes the
sequence to cycle back to Layout One again as shown in Figure 14. However, the page layout
editing system and method is not limited to the different layouts in Figures 9-14 and the page
layout editing system and method may be implemented using less or more different page layouts.

Thus, in the example, Figures 9-14 shows an example of a progression of layout incrementing
and propagation that takes place as the user taps repeatedly on the selected page.

The layout change may be automatically and dynamically propagated to the subsequent
pages of the book (7206) in the background which is part of the dynamic reflow process. The
application 7103 may also display the changed layout (7208). Assuming that the book has 10
pages and the user is currently viewing the layout of page 4, pages 1-3 may be referred to as
upstream page(s) (one or more pages in the book before the selected page) and pages 5-10 may
be referred to a downstream page(s) (one or more pages in the book after the selected page.)
Thus, the process 7206 reflows the content into the new layout positions downstream from the
selected page, but upstream pages are unaffected. This is done by incrementing a sequence
number (the Layout number described above) for the current page, and then incrementing the
sequence number for each subsequent page, so that each of these subsequent pages has one added
to its sequence number as well. This results in a default layout scheme where each page has a
different layout pattern than the preceding page. The changing of the layouts may be animated to
show the movement of the content on each page into the new positions so that the user has good
visual feedback that a reflow event has occurred.

In order to optimize the processor load of the computing device for very large books, such
as very large scrapbooks, the reflow animations described above may be performed first for only
the two affected pages visible on the screen (the current selected page and the immediately
subsequent page), then the remainder of the book is reflowed in the background. With this
approach, even books with 1000 pieces of content or more can be reflowed instantly with a single
tap. In other embodiments, more or less pages and layout patterns can be displayed on the screen
at any given time. A user may use standard types of user interface features such as zooming,
panning/scrolling, etc. Different methods of rendering pages and performing background
processing can be used.
Once the user has settled on the desired layout design for the selected page, then the page images can be scrolled to the next page, and the process shown in Figure 8 can be repeated. The user can repeat this process very quickly and easily as each page is stepped through, all via the action of a single finger or thumb. When the user is satisfied with all the changes to the scrapbook, the Save button in the command portion 300 can be tapped, to save the changes. If the user has manually reflowed the layouts from a position within the book, using the mechanism described above, then moves back to an upstream page in the book, such as page 2 in the example above, the method handles this situation. Specifically, when the user taps on the page layout to reflow from that point, the reflow propagation will only be performed up to the downstream page where the manual reflow was performed. When this occurs, the application 7103 may alert the user with a vibrate signal. This reflow behavior allows the user to move back and forth within the book as he or she progressively refines the overall design of the book.

The dynamic reflow system allows the user of a small mobile device to be able to quickly and easily create and edit page layouts for books, such as digital scrapbooks, so that those books can be instantly shared directly from the phone immediately after creation. The application 103 allows users to capture, format, and share important events in their lives, all through the use of a small mobile device. Using this innovative system, rich experiences can be shared with the world in the instant, within the flow of and without disruption to end users' increasingly busy lives.

Although the description has been described with respect to particular embodiments thereof, these particular embodiments are merely illustrative, and not restrictive. For example, although details of particular application specific interfaces, languages and other resources and/or components may be mentioned, it should be apparent that embodiments may be adaptable for use with other types of resources and components.

Any suitable programming language can be used to implement the routines of particular embodiments including C, C++, Java, assembly language, etc. Different programming techniques can be employed such as procedural or object oriented. The routines can execute on a single processing device or multiple processors. Although the steps, operations, or computations may be presented in a specific order, this order may be changed in different particular embodiments. In some particular embodiments, multiple steps shown as sequential in this
specification can be performed at the same time.

Particular embodiments may be implemented in a computer-readable storage medium for use by or in connection with the instruction execution system, apparatus, system, or device. Particular embodiments can be implemented in the form of control logic in software or hardware or a combination of both. The control logic, when executed by one or more processors, may be operable to perform that which is described in particular embodiments. For example, a tangible, or non-transitory, medium such as a hardware storage device can be used to store the control logic, which can include executable instructions.

Particular embodiments may be implemented by using a programmed general purpose digital computer, by using application specific integrated circuits, programmable logic devices, field programmable gate arrays, optical, chemical, biological, quantum or nanoengineered systems, components and mechanisms may be used. In general, the functions of particular embodiments can be achieved by any means as is known in the art. Distributed, networked systems, components, and/or circuits can be used. Communication, or transfer, of data may be wired, wireless, or by any other means.

It will also be appreciated that one or more of the elements depicted in the drawings/figures can also be implemented in a more separated or integrated manner, or even removed or rendered as inoperable in certain cases, as is useful in accordance with a particular application. It is also within the spirit and scope to implement a program or code that can be stored in a machine-readable medium to permit a computer to perform any of the methods described above.

A "processor" includes any suitable hardware and/or software system, mechanism or component that processes data, signals or other information. A processor can include a system with a general-purpose central processing unit, multiple processing units, dedicated circuitry for achieving functionality, or other systems. Processing need not be limited to a geographic location, or have temporal limitations. For example, a processor can perform its functions in "real time," "offline," in a "batch mode," etc. Portions of processing can be performed at different times and at different locations, by different (or the same) processing systems. A computer may be any processor in communication with a memory. The memory may be any suitable processor-readable storage medium, such as random-access memory (RAM), read-only
memory (ROM), magnetic or optical disk, or other tangible media suitable for storing instructions for execution by the processor.

As used in the description herein and throughout the claims that follow, "a", "an", and "the" includes plural references unless the context clearly dictates otherwise. Also, as used in the description herein and throughout the claims that follow, the meaning of "in" includes "in" and "on" unless the context clearly dictates otherwise.

Thus, while particular embodiments have been described herein, latitudes of modification, various changes, and substitutions are intended in the foregoing disclosures, and it will be appreciated that in some instances some features of particular embodiments will be employed without a corresponding use of other features without departing from the scope and spirit as set forth. Therefore, many modifications may be made to adapt a particular situation or material to the essential scope and spirit.

As disclosed herein, features consistent with the disclosure may be implemented via computer-hardware, software and/or firmware. For example, the systems and methods disclosed herein may be embodied in various forms including, for example, a data processor, such as a computer that also includes a database, digital electronic circuitry, firmware, software, or in combinations of them. Further, while some of the disclosed implementations describe specific hardware components, systems and methods consistent with the innovations herein may be implemented with any combination of hardware, software and/or firmware. Moreover, the above-noted features and other aspects and principles of the innovations herein may be implemented in various environments. Such environments and related applications may be specially constructed for performing the various routines, processes and/or operations according to the invention or they may include a general-purpose computer or computing platform selectively activated or reconfigured by code to provide the necessary functionality. The processes disclosed herein are not inherently related to any particular computer, network, architecture, environment, or other apparatus, and may be implemented by a suitable combination of hardware, software, and/or firmware. For example, various general-purpose machines may be used with programs written in accordance with teachings of the invention, or it may be more convenient to construct a specialized apparatus or system to perform the required methods and techniques.
It should also be noted that the various logic and/or functions disclosed herein may be enabled using any number of combinations of hardware, firmware, and/or as data and/or instructions embodied in various machine-readable or computer-readable media, in terms of their behavioral, register transfer, logic component, and/or other characteristics. Computer-readable media in which such formatted data and/or instructions may be embodied include, but are not limited to, non-volatile storage media in various forms (e.g., optical, magnetic or semiconductor storage media) though again does not include transitory media. Unless the context clearly requires otherwise, throughout the description, the words "comprise," "comprising," and the like are to be construed in an inclusive sense as opposed to an exclusive or exhaustive sense; that is to say, in a sense of "including, but not limited to." Words using the singular or plural number also include the plural or singular number respectively. Additionally, the words "herein," "hereunder," "above," "below," and words of similar import refer to this application as a whole and not to any particular portions of this application. When the word "or" is used in reference to a list of two or more items, that word covers all of the following interpretations of the word: any of the items in the list, all of the items in the list and any combination of the items in the list.

While the foregoing has been with reference to a particular embodiment of the disclosure, it will be appreciated by those skilled in the art that changes in this embodiment may be made without departing from the principles and spirit of the disclosure, the scope of which is defined by the appended claims.
Claims:

1. A cloud resource system, comprising:
   one or more computing devices;
   a processor and a memory;
   a mobile supercloud component coupled to the one or more computing devices and
   executed by the processor; and
   the mobile supercloud component being configured to identify a plurality of cloud
   computing resources in a cloud connected to the Internet, associate the plurality of cloud
   computing resources with a meta-cloud, map at least one of the plurality of cloud computing
   resources to at least one meta-cloud resource, receive a request via a communication path, from
   one of the one or more computing devices to access the at least one said resource and causing a
   transfer of information from the at least one said resource in response to the request to the client.

2. The system of claim 1, wherein the transferred information is a location of the
   abstracted resource to enable the client to access said at least one of the abstracted resources.

3. The system of claim 1, wherein the mobile supercloud component associates at
   least one resource of the computing device with the meta-cloud and maps the at least one
   resource of the computing device to at least one meta-cloud resource.

4. The system of claim 3, wherein the resource of the computing device is a database
   resident on the computing device.

5. The system of claim 4 further comprising a client on one of the one or more
   computing devices that changes a piece of information in the resident database.

6. The system of claim 5, wherein the mobile supercloud component synchronizes
   the changed piece of information in the resident database with a database server.

7. The system of claim 1, wherein the mobile supercloud component synchronizes a
   changed piece of data in a database server with a database resident on the computing device.

8. The system of claim 4, wherein the mobile supercloud component restores a
   database server using the database resident on the computing device.

9. The system of claim 4, wherein the mobile supercloud component synchronizes
   restores the database resident on the computing device with a database server.

10. A method for accessing a cloud resource for a client, comprising:
identifying a plurality of cloud computing resources in a cloud connected to the Internet; associating the plurality of cloud computing resources with a meta-cloud; mapping at least one of the plurality of cloud computing resources to at least one meta-cloud resource;

receive a request via a communication path, from a client on a computing device to access the at least one said resource;

causing a transfer of information from the at least one said resource in response to the request to the client.

11. The method of claim 10, wherein the transferred information is a location of the abstracted resource to enable the client to access said at least one of the abstracted resources.

12. The method of claim 10 further comprising associating at least one resource of the computing device with the meta-cloud and mapping the at least one resource of the computing device to at least one meta-cloud resource.

13. The method of claim 12, wherein the resource of the computing device is a database resident on the computing device.

14. The method of claim 13 further comprising changing a piece of information in the resident database.

15. The method of claim 14 further comprising synchronizing the changed piece of information in the resident database with a database server.

16. The method of claim 10 further comprising synchronizing a changed piece of data in a database server with a database resident on the computing device.

17. The method of claim 13 further comprising restoring a database server using the database resident on the computing device.

18. The method of claim 13 further comprising restoring the database resident on the computing device with a database server.

19. A method, comprising:
requesting, by a client, a data asset using a browser having an address bar;
receiving a data asset identifier for the requested data asset, the data asset identifier identifying a first location of the data asset;
generating a masked data link redirection for the requested data asset based on data asset identifier, wherein the masked data link redirection redirects the browser to a different location for the retrieval of the data asset and the address bar of the browser displays the first location of the data asset specified by the data asset identifier; and

retrieving, by the client, the data asset from the different location.

20. The method of claim 19, wherein generating the masked data link redirection further comprises using a dereferenced Uniform Resource Locator.

21. The method of claim 19, wherein generating the masked data link redirection further comprises using a 301 redirect.

22. The method of claim 19 further comprising requesting, by the client, information from a cloud service wherein the information includes the requested data asset.

23. The method of claim 19, wherein the data asset identifier is a uniform resource locator for the data asset.

24. The method of claim 19, wherein the different location is a storage location on a third party system.

25. The method of claim 22, wherein the client is a scrapbooking application and requesting information from a cloud service further comprises a piece of code for a layout of a scrapbook and the requested data asset is a data asset being laid out in the scrapbook.

26. The method of claim 27, wherein the piece of code is JSON code.

27. A system, comprising:

   a computing device having a client, wherein the client requests a data asset using a browser having an address bar and receives a data asset identifier for the requested data asset, the data asset identifier identifying a location of the data asset;

   a processor and memory that hosts a cloud services component;

   the processor configured to generate a masked data link redirection for the requested data asset based on data asset identifier, wherein the masked data link redirection redirects the browser to a different location for the retrieval of the data asset and the address bar of the browser displays the location of the data asset specified by the data asset identifier; and

   wherein the client retrieves the data asset from the different location.
28. The system of claim 27, wherein the processor uses a dereferenced Uniform Resource Locator to generate a masked data link redirection.

29. The system of claim 27, wherein the processes uses a 301 redirect to generate a masked data link redirection.

30. The system of claim 27, wherein the client requests information from a cloud service wherein the information includes the requested data asset.

31. The system of claim 27, wherein the data asset identifier is a uniform resource locator for the data asset.

32. The system of claim 27, wherein the different location is a storage location on a third party system.

33. The system of claim 30, wherein the client is a scrapbooking application and requesting information from a cloud service further comprises a piece of code for a layout of a scrapbook and the requested data asset is a data asset being laid out in the scrapbook.

34. The system of claim 33, wherein the piece of code is JSON code.

35. An apparatus, comprising:
   a processor and a memory;
   a cloud services component executed by the processor;
   wherein the processor is configured to receive, by a client, a request for a data asset, generate a masked data link redirection for the requested data asset based on a data asset identifier, wherein the masked data link redirection redirects a browser of the client to a different location for the retrieval of the data asset and the address bar of the browser displays the location of the data asset specified by the data asset identifier so that the client is capable of retrieving the data asset from the different location.

36. The apparatus of claim 35, wherein the processor uses a dereferenced Uniform Resource Locator to generate the masked data link redirection.

37. The apparatus of claim 35, wherein the processor uses a 301 redirect.

38. The apparatus of claim 35, wherein the data asset identifier is a uniform resource locator for the data asset.

39. The apparatus of claim 35, wherein the different location is a storage location on a third party system.
40. A method for editing a layout of a plurality of pages in a digital book, each page
in the book being sequentially numbered and the book having a plurality of layouts and each
layout for a page being sequentially numbered, the method comprising:

displaying, on a computing device, two or more pages of the digital book, the two or
more pages each having a layout selected from the plurality of layouts;

receiving, by a computer device, an input indication of a layout change for a selected page
of the digital book;

displaying, on the computer device, a new layout for the selected page in response to the
input indication, wherein the new layout is a next page layout in the sequentially numbered
plurality of page layouts; and

propagating, by the computer device in response to the input indication, a new layout for
a next page of the digital book that is a downstream page from the selected page, wherein the
downstream page is a higher numbered page of the digital book and the new layout is a next
layout in the sequentially numbered plurality of page layouts.

41. The method of claim 40, wherein displaying the new layout further comprises
displaying a first layout of the sequentially numbered plurality of page layouts when a current
layout of the page is a last layout of the sequentially numbered plurality of page layouts.

42. The method of claim 40, wherein propagating the new layout further comprises
propagating, by the computer device, a new layout for a next page of the digital book that is a
downstream page from the selected page, the new layout being a first layout of the sequentially
numbered plurality of page layouts when a current layout of the page is a last layout of the
sequentially numbered plurality of page layouts.

43. The method of claim 40, wherein propagating the new layout further comprises
incrementing a layout sequence number for the downstream page.

44. The method of claim 40 further comprising animating, by the computing device,
the changes in the layout of the pages of the digital book.

45. The method of claim 40 further comprising receiving, by the computer device, an
input indication of a layout change for a second selected page of the digital book, wherein the
second selected page is a lower numbered page of the digital book and propagating, by the
computer device, a new layout for a next page of the digital book that is a downstream page from the second selected page.

46. The method of claim 40, wherein the input indication is a user tap.

47. The method of claim 40, wherein the digital book is a digital scrapbook.

48. An apparatus for editing a layout of a plurality of pages in a digital book, each page in the book being sequentially numbered and the book having a plurality of layouts and each layout for a page being sequentially numbered, the apparatus comprising:

- a computing device having a processor and a memory;
- an application executed by the processor of the computing device that includes a dynamic reflow component;
- the processor configured to display two or more pages of the digital book, the two or more pages each having a layout selected from the plurality of layouts;
- the processor configured to receive an input indication of a layout change for a selected page of the digital book;
- the processor configured to display a new layout for the selected page in response to the input indication, wherein the new layout is a next page layout in the sequentially numbered plurality of page layouts; and
- the processor configured to propagate, in response to the input indication, a new layout for a next page of the digital book that is a downstream page from the selected page, wherein the downstream page is a higher numbered page of the digital book and the new layout is a next layout in the sequentially numbered plurality of page layouts.

49. The apparatus of claim 48, wherein the processor is configured to display a first layout of the sequentially numbered plurality of page layouts when a current layout of the page is a last layout of the sequentially numbered plurality of page layouts.

50. The apparatus of claim 48, wherein the processor is configured to propagate a new layout for a next page of the digital book that is a downstream page from the selected page, the new layout being a first layout of the sequentially numbered plurality of page layouts when a current layout of the page is a last layout of the sequentially numbered plurality of page layouts.
51. The apparatus of claim 48, wherein the processor is configured to increment a layout sequence number for the downstream page to propagate a layout change in the pages of the book.

52. The apparatus of claim 48, wherein the processor is configured to animate the changes in the layout of the pages of the digital book.

53. The apparatus of claim 48, wherein the processor is configured to receive an input indication of a layout change for a second selected page of the digital book, wherein the second selected page is a lower numbered page of the digital book and to propagate a new layout for a next page of the digital book that is a downstream page from the second selected page.

54. The apparatus of claim 48, wherein the input indication is a user tap.

55. The apparatus of claim 48, wherein the digital book is a digital scrapbook.
FIGURE 2
START

CLIENT REQUESTS APP OR CODE

DATA ASSET IDENTIFIERS REQUESTED

GENERATE DATA LINK REDIRECTIONS BASED ON DATA ASSET IDENTIFIERS

LOAD DATA ASSET FROM REMOTE SYSTEM

END

FIGURE 5
req #1: web client requests webview html

req #2: webview calls webapi that returns the json layout and ks asset ID data urls

req #3: webview loads images by requesting ks asset ID. The response is a 301 redirect to the real dropbox asset

req #4: webview loads image/video direct from dropbox

FIGURE 6
START

PAGE LAYOUT DISPLAYED

START

COMPUTING DEVICE RECEIVES AN INPUT INDICATION OF A CHANGE IN THE LAYOUT OF THE PAGE

START

PROPAGATE THE LAYOUT CHANGE TO SUBSEQUENT PAGE(S) IN THE BACKGROUND

START

DISPLAY CHANGED LAYOUT

END

FIGURE 8
FIGURE 10
FIGURE 12
FIGURE 13
FIGURE 14
INTERNATIONAL SEARCH REPORT

International application No.
PCT/US2015/037528

A. CLASSIFICATION OF SUBJECT MATTER
IPC(8) - G06F 9/44 (2015.01)
CPC - G06F 9/44 (2015.04)
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)
IPC(8) - G06F 9/44, 15/173, 17/30; H04W 72/04 (2015.01)
CPC - G06F 9/44, 15/173, 17/30; H04W 72/04 (2015.04) (keyword delimited)

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched
CPC - G06F 9/44, 15/173, 17/30; H04W 72/04 (2015.04) (keyword delimited)

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
Orbit, Google Patents, Google Scholar, Google.
Search terms used: supercloud, identifies, resources, phone, location, database, synchronization

C. DOCUMENTS CONSIDERED TO BE RELEVANT

<table>
<thead>
<tr>
<th>Category</th>
<th>Citation of document, with indication, where appropriate, of the relevant passages</th>
<th>Relevant to claim No.</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>US 2013/0013767 A1 (STOBER et al) 10 January 2013 (10.01.2013) entire document</td>
<td>1,10</td>
</tr>
<tr>
<td>Y</td>
<td>US 2014/0101 102 A1 (APPLE, INC) 10 April 2014 (10.04.2014) entire document</td>
<td>2,1 1</td>
</tr>
<tr>
<td>A</td>
<td>US 7,120,462 B2 (KUMAR) 10 October 2006 (10.10.2006) entire document</td>
<td>4,6,8,9,13-15,17,18</td>
</tr>
</tbody>
</table>

Further documents are listed in the continuation of Box C. 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 application or patent 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

"G" - document member of the same patent family

Date of the actual completion of the international search
06 October 2015

Date of mailing of the international search report
04 NOV 2015

Name and mailing address of the ISA:
Mail Stop PCT, Attn: ISA-US, Commissioner for Patents
P.O. Box 1450, Alexandria, Virginia 22313-1450
Facsimile No. 571-273-8300

Authorized officer
Blaine Copenheaver
PCT Helpdesk: 571-272-4300
PCT OSP: 571-272-7774

Form PCT/ISA/2 10 (second sheet) (January 2015)
### Observations where certain claims were found unsearchable

This international search report has not been established in respect of certain claims under Article 17(2)(a) for the following reasons:

1. □ Claims Nos.:
   - because they relate to subject matter not required to be searched by this Authority, namely:

2. □ Claims Nos.:
   - because they relate to parts of the international application that do not comply with the prescribed requirements to such an extent that no meaningful international search can be carried out, specifically:

3. □ Claims Nos.:
   - because they are dependent claims and are not drafted in accordance with the second and third sentences of Rule 6.4(a).

### Observations where unity of invention is lacking

This International Searching Authority found multiple inventions in this international application, as follows:

**See Extra Sheet**

1. □ As all required additional search fees were timely paid by the applicant, this international search report covers all searchable claims.
2. □ As all searchable claims could be searched without effort justifying additional fees, this Authority did not invite payment of additional fees.
3. □ As only some of the required additional search fees were timely paid by the applicant, this international search report covers only those claims for which fees were paid, specifically claims Nos.:

4. ☒ No required additional search fees were timely paid by the applicant. Consequently, this international search report is restricted to the invention first mentioned in the claims; it is covered by claims Nos. 1-18

**Remark on Protest**

- □ The additional search fees were accompanied by the applicant's protest and, where applicable, the payment of a protest fee.
- □ The additional search fees were accompanied by the applicant's protest but the applicable protest fee was not paid within the time limit specified in the invitation.
- □ No protest accompanied the payment of additional search fees.

Form PCT/ISA/2 to (continuation of first sheet (2)) (January 2015)
This application contains the following inventions or groups of inventions which are not so linked as to form a single general inventive concept under PCT Rule 13.1. In order for all inventions to be examined, the appropriate additional examination fees must be paid.

Group I, claims 1-18, drawn to a cloud resource system.
Group II, claims 19-39, drawn to requesting, by a client, a data asset using a browser having an address bar
Group III, claims 40-55, drawn to a method for editing a layout of a plurality of pages in a digital book.

The inventions listed as Groups I, II and III do not relate to a single general inventive concept under PCT Rule 13.1 because, under PCT Rule 13.2, they lack the same or corresponding special technical features for the following reasons: the special technical feature of the Group I invention: map at least one of the plurality of cloud computing resources to at least one meta-cloud resource, receive a request via a communication path, from one of the one or more computing devices to access the at least one said resource and causing a transfer of information from the at least one said resource in response to the request to the client as claimed therein is not present in the invention of Groups II and III. The special technical feature of the Group II invention: generate a masked data link redirection for the requested data asset based on data asset identifier, wherein the masked data link redirection redirects the browser to a different location for the retrieval of the data asset and the address bar of the browser displays the location of the data asset specified by the data asset identifier as claimed therein is not present in the invention of Groups I or III. The special technical feature of the Group III invention: the processor configured to display two or more pages of the digital book, the two or more pages each having a layout selected from the plurality of layouts; the processor configured to receive an input indication of a layout change for a selected page of the digital book; the processor configured to display a new layout for the selected page in response to the input indication, wherein the new layout is a next page layout in the sequentially numbered plurality of page layouts as claimed therein is not present in the invention of Groups I or II.

Groups I, II and III lack unity of invention because even though the inventions of these groups require the technical feature of one or more computing devices; a processor and a memory, this technical feature is not a special technical feature as it does not make a contribution over the prior art.

Specifically, US 7,120,462 B2 (KUMAR) 10 October 2006 (10.10.2006) teaches one or more computing devices and a processor and a memory (abstract).

Since none of the special technical features of the Group I, II or III inventions are found in more than one of the inventions, unity of invention is lacking.