

EXHIBIT A
Part 2 of 2

15

In prior art systems, application providers wishing to provide applications had to buy a server, then they must buy or develop the applications that are going to be loaded and run on that server, load the server, and activate the server to provide access to that application. The server is a fully dedicated resource, so that 100% of the time an application is dedicated to a specific server. The present on-demand application system 140 reverses or switches this paradigm and instead of applications being dedicated to a server, the on-demand system 140 provides computing resources on-demand, when demand for an application is received, and additionally frees up resources when demand falls off for the restoring of completely different applications. Further, application providers no longer need to purchase the servers. Application providers simply take advantage of the on-demand application processing system 140 already deployed by loading their applications onto the distributed on-demand system 140. The on-demand system 140 allows an application provider to allow substantially an unlimited number of users to access substantially the same application at substantially the same time without over loading the system 140 or the application, all without the need to incur the extremely expensive capital expense of providing their own system. Instead, the application provider pays for the amount of resources utilized to provide their users access to the applications. As demand increase, the on-demand system 140 increases the number of applications running, increasing the amount of compute resources and capacity, thus the application provider is charged more; as demand falls, the on-demand system 140 decreases the number of application instances, reducing the amount of computational capacity, thus the application provider is charged less. Thus, the application provider is charged for the amount of resources used.

In one embodiment, the present invention provides for a distributed on-demand system 140 such that potentially thousands of servers 354 in hundreds of edgepoints 350 are globally deployed and linked to create a virtual single server view throughout the world. This virtual single server view provides an application provider with access and control over their own applications in the system 140.

FIG. 14A-C show one implementation of one embodiment of the novel method and system 140 which allows the application providers to dictate the distribution of their applications, the potential compute resources to be available, upgrade or alter their applications, replace applications, monitor applications, and monitor the amount of resources utilized by entities accessing their distributed applications.

Prior art application processing systems require an application provider to route a user to a single central site to allow access to the applications. Every user attempting to access the application is directed to the single central site. Thus, resulting in the bottle neck as discussed above. In the prior art single server or single central site, the application provider, however, does maintain access to and control over the application. In some systems where the application provider outsources their server capacity, the application provider must select from a preselected, limited number of applications. Further, the application provider no longer has direct control over the application. Any changes desired by the application provider are submitted by request to the server provider. Then the server provider must schedule a time at low demands to take the server down to make the changes. This process results in large lag times between the decision to make changes and the implementation of those changes.

FIG. 14A shows a simplified block diagram of one embodiment of the present on-demand application processing system 140 in cooperation with the preexisting internet infrastructure

16

126. The present distributed on-demand application processing method and system 140 provides for distributed processing capabilities, with on-demand capacity, as well as providing the application provider with direct access to the on-demand system 140 and thus direct access and control over their applications. The application provider has control to distribute new applications or change already distributed applications throughout the on-demand system 140. In one embodiment, the on-demand system 140 is configured to provide an application provider with a virtual single server view of the on-demand system 140. This virtual single server view allows an application provider complete access and control over the applications which the application provider decides to distribute over the system 140. In one embodiment, the system 140 provides an application provider with the ability to deploy applications throughout the distributed on-demand system 140, change and update deployed applications, and monitor any one or all of the applications deployed throughout the internet 126 from a single terminal or site. In one embodiment, the application provider can deploy or access their applications through any one of a plurality of terminals or locations, including computers located at their own facilities. The present on-demand system 140 provides the application provider with the ability to deploy applications and access those applications once distributed through the system 140. As referred to herein, conduit 360 is a staging facility that enables the application provider to deploy applications across a computer network. Through the conduit 360 application provider deploys applications, retains control over their deployed applications, monitors the operation of their applications on the system 140, checks billing information, checks metering information, checks performance information, and so forth.

By structuring the on-demand system 140 as a single distributed system, and allowing the application provider with access to the on-demand system 140 through a single point, the on-demand system 140 appears to the application provider as a single server. Thus, when the application provider wishes to load and implement a new application onto the system 140, the application provider simply accesses the on-demand system 140 through conduit 360. Still referring to FIG. 14A, in one embodiment, application provider is capable of loading an application as an appshot 220 onto the on-demand system 140 from a single origin site 362 through the conduit 360. The application provider loads the appshot 220 onto the system 140. The application provider is then capable of designating specific locations (for example, areas of the world wide system 140, such as, edgepoints 350a and 350b representing Asia, Western Europe, the United States, Germany, a north-eastern portion of the United States, or London) or the entire system to allow users to access the deployed applications. Once the areas of distribution are designated, the on-demand system 140 distributes the appshot 220 through a hub 370 to the edgepoints 350 in the areas designated by the application provider. FIG. 14B shows a simplified block diagram of one embodiment of the on-demand system 140 with the appshot 220 distributed to edgepoints 350a and 350b. Once the appshot 222 is distributed, user 124 can then be routed 378 to the most optimal edgepoint 350a having the specified application, instead of being routed to the central site 362. Because the on-demand system 140 is configured to appear as a single virtual server, the application provider loads the appshot 222 once on to the system 140. The application provider does not need to load the application onto each edgepoint to distribute the application to specific areas of the on-demand system 140 or throughout the system 140.

Further, the virtual single server mechanism also allows the application provider access to the appshot 220 through conduit 360 from a single point in the on-demand system 140. Referring to FIG. 14C, the application provider is capable of making changes or up-grading 374 the appshot 220 through conduit 360 and/or replacing the appshot. In one embodiment, the change or up-grade 374 is made once. This change or up-grade 374 is then distributed throughout the system to those edgepoints 350a-b providing the desired application without additional input from the application provider. Thus, the application providers maintain control over their applications and how the applications are distributed. The application providers are further capable of directly implementing changes and updates and replacements to their applications.

FIGS. 15A-C show a simplified block diagram of one implementation of one embodiment of the optimal user and entity routing provided by the present invention. FIG. 15A shows a block diagram of one embodiment of the on-demand application processing system 140. One of the advantages of the on-demand system 140 and virtual single server mechanism is the ability to route a user 124 to an optimal edgepoint 350 which will provide the user 124 with the least amount of latency delays while avoiding overloading a single edgepoint 350. For example, referring to FIG. 15A, to reduce the amount of latency delay, it would be preferable to route the user 124 to the geographically closest edgepoint 350a, as designated by dashed line 1. However, because of an overriding condition, for example edgepoint 350 is overloaded by other users, the user 124 is routed to a second edgepoint 350b, as designated by arrow 2, adding a minimal amount of latency delay but providing enhanced response time because the second edgepoint 350b is not as heavily loaded, thus providing an overall superior interaction.

Referring to FIG. 15B, another advantage of the on-demand system 140 is that the applications 356 can be interactive. Thus allowing the user 124 to make changes or additions 380 to the information provided through the application 356. Further, once changes are made to the application 356, the on-demand system 140 will continue to route the user 124 to that same edgepoint 350b, as designated by arrow 3, for future connections of that user 124 to the on-demand system 140 for that application 356. This affinity for that user 124 ensure that the user 124 continues to interact with the most current version of the application 356 according to the user's information and changes. The on-demand system 140 also provides the capability to synchronize or update the system 140 to reflect the changes 380 made by the user 124. The updates are made at anytime as dictated by the on-demand system 140, such as periodically, randomly or by schedule. As shown in FIG. 15B, the changes 380 made by the user 124 are forwarded to the origin site 362 through conduit 360, as shown with reference to arrows 4 and 5 updating the origin site. In one embodiment, the on-demand system 140 is further capable of distributing the changes 380 to the other edgepoints 350a which provide access to the application 356, shown by arrow 6. Thus, the on-demand system 140 synchronizes data and user's changes 380 throughout the system, maintaining an active and current system, without further interaction by the application provider. Examples of user changes include changes to a user profile, items added to a shopping cart, and other such changes. One example of data changes from the application provider would be an online catalog update from the application provider to the various edgepoints. Referring to FIG. 15C, once the on-demand system 140 is updated and the user's changes 380 are distributed throughout the on-demand system 140, the on-demand system 140 is again free to re-route or route the user 124 for

future connections to any optimal edgepoint 350 in the system 140 providing the needed application 356, as noted by arrow 7. Thus, the present on-demand method and system 140 provides affinity in the routing scheme along with the ability to synchronize the on-demand system 140.

Some of the additional features and benefits provided by the novel on-demand application processing method and system 140 include edge staging and load testing of applications and sites. The on-demand system 140 allows application providers to directly install new versions of a website or application onto the system 140. The system 140 allows the application provider to limit the access to the new application or website. Thus, application providers are able to access the new website or application and functionally test the distributed site or application. Further, the on-demand system 140 allows the application provider to load test the application or site prior to allowing public access and use. For example, utilizing the on-demand system resources, numerous synthetic simultaneous sessions are able to be activated at a single time to load test the application or site. The application provider is able to perform these load tests without the capital expenditure of having to purchase additional equipment to perform load testing. Further, because of the pricing scheme of the present on-demand method, the application provider then pays for the amount of capacity utilized during this testing. Which is significantly less expensive than purchasing additional equipment.

FIG. 16 shows a simplified flow diagram of one implementation of one embodiment of the process or method 600 and system providing on-demand computing resources. In step 604, an application request is received from an entity. Once a request is received, the process 600 enters step 606 where it is determined if the request from the entity is bound to a specific compute resource, such as a specific edgepoint. If the request is not bound, step 610 is entered where the optimal edgepoint is determine for providing the compute resources for application processing. In one embodiment, the optimal edgepoint is determined based on a plurality of criteria, including minimal latency delay, capacity of the edgepoint, the distribution of the desired application across the network, and other such parameters. Step 612 is entered if, in step 606, the request is bound or once the optimal edgepoint is determined in step 610. In step 612 it is determined if the bound or optimal edgepoint is operating at capacity. If the edgepoint is operating at capacity, step 614 is entered where it is determined whether the edgepoint can free up sufficient resources by snapshotting one or more application instances. If resources cannot be freed up, the process 600 returns to step 610 to reevaluate and determine the optimal edgepoint. Step 616 is entered, if in step 612, it is determined that the edgepoint is not at capacity, or in step 614 it is determined that capacity can be freed up on the edgepoint. In step 616, the user is routed to the edgepoint where the edgepoint determines optimal routing of the user to an instance of the desired application.

FIG. 17 shows a simplified block diagram of one implementation of one embodiment of the on-demand apparatus 140 including a plurality of edgepoints 350a-b. The network 140 can include substantially any number of edgepoints. FIG. 17 depicts two edgepoints for simplicity, however, it will be apparent to one skilled in the art that the network 140 can include substantially an unlimited number of edgepoints. Network 140 is configured to at least provide application processing for remote users 124 over the internet 126. Network 140 can include any number of edgepoints allowing the computational capacity of network 140 to be scaled. Network 140 provides user 124 with differential computational capacity through either of the edgepoints 380a-b. In one embodi-

ment, network **140** includes a global dispatcher (GD) **430** which at least provides routing of application requests from user **124** to an edgepoint of the network. Based on network parameters including, but not limited to, edgepoint load and which applications are currently provisioned on each edgepoint **380a-b**, GD **430** routes the user to the optimal edgepoint, for example first edgepoint **380a**. Once the first edgepoint **380a** receives the user request, the request is accepted and dispatched by a local dispatcher **434a**. A resource manager **432a** determines which of a plurality of compute resources or modules **436a₁-436a₃** would be the optimal compute module in which to route the application request. In determining the optimal compute module, the resource manager **432a** determines if a compute module is currently running the desired application or whether the application needs to be restored from a snapshot state. The resource manager further determines if compute resources need to be freed up. If resources need to be freed, the resource manager signals a snapd module **440** of the optimal compute module, where the snapd module snapshots one or more application instances, thus freeing up the resources which were associated with the snapshot applications. Once the optimal compute module has the available resources, the resource manager **432a** signals the optimal compute module, for example first compute module **436a₁**, where the restored module **442** restores the desired application from memory **352a** if necessary, and initializes the application to allow the user to interact with or operate the desired application.

In one embodiment, restored is a daemon that restores the state of all processes that belong to a single snapid. Snaplist identifies the list of processes that need to be restored. In one embodiment, the restore program “morphs” into the target process. The restore program creates processes in the order determined by, for example, a parent/child relationship. Each restore program is designated a process-id (pid) that it morphs to and each will do so by reading the appropriate snapshot file.

In one embodiment, the resource manager **432** further monitors the compute resources utilized by each application instance and records the compute resources utilized. The on-demand system **140** uses the recorded resource utilization for determining the amount an application provider is charged for the use of the compute resources. In one embodiment, the system includes a monitoring module **464**, which monitors the compute resources and provides the monitored results to the resource manager and other components of the system. In one embodiment, the resource manager (RM) utilizes the perfd module to collect at least some of the data to determine the amount of resources utilized per application instance. The resource manager **432** monitors such things as CPU utilization, network bandwidth, disk utilization, license usage, response time latencies, and other such parameters for determining resource utilization.

In one embodiment, a perfd module comprises an agent or other entity running on the computer node (CN) that computes or otherwise determines the performance of each of the CNs. A call to this perfd agent is initiated by the resource manager (RM). This perfd agent makes system calls to collect the statistics and sends it to resource manager. Viewed somewhat differently, perfd is or includes a daemon or other mechanism that collects performance information, hence the abbreviation perfd for performance daemon. In a more general way, a perfd module is any performance determining module.

In one embodiment, a compute node is a component within an embodiment of an edgepoint that runs the components of an Application Instance. Typically, these are application tiers such as a webserver connected to a middle-tier and a database.

In one embodiment, an edgepoint is a machine or collection of machines that run the customers site. An administrative node (AN) is the component within an edgepoint that runs the administrative components of an edgepoint. For example, a configuration database, deployer components, data synchronization components, and monitoring components are run in the administrative or admin node.

In one embodiment, the network **140** further includes at least one deployment center **444**, deployment database (DDB) **446**, conduit **360**, and dashboard (i.e., a network operating center (NOC) dashboard **454** and/or a customer dashboard **456**). In one embodiment, the edgepoints further include a global dispatch module **460**, a data synchronization module **462** and the metering module **464**. The plurality of edgepoints communicate such that snapshot instances of applications can be transferred and users rerouted.

The global dispatch module **460** provides communication with and access to the network **140**, and aids in network routing to allow entities, such as users **124**, access to the applications and compute resources. The global dispatch module **460** further receives information from the network and other edgepoints regarding the amount of compute resources available on other edgepoints **350** and throughout the network **140** to aid in optimizing the resources of the network. The data synchronization module **462** communicates with the network to receive data and information from the network to update the edgepoint **350** with the data. The data synchronization module **462** allows new or changed data distributed across the network to be forwarded to the compute modules **436** and/or the memory **352** of the edgepoint **350**. In one embodiment, the data synchronization module **462** allows data added or changed by a user **124** to be distributed across the network **140**. The metering module **464** monitors the edgepoint and the compute resources utilized by an application, user, group of applications, and any combination thereof. The metering module **464** acts in cooperation with the resource manager **432** to monitor the compute modules **436** and collects data from the compute modules regarding usage. In one embodiment, the metering module is further configured to determine an amount to charge the application providers based on the amount of compute resources utilized by each application provider for application processing.

In one embodiment, the deployment center **444** acts as the hub that collects data, policies and applications and distributes them to the edgepoints **350a-b**. Deployment center **444** maintains application and data versions and distributes updates, revisions and replacements which are forwarded to the deploy modules **433a-b** of the edgepoints **350a-b**. In one embodiment, the deployment through the deployment center **444** includes capturing application states (initial and updates), policies and testing methods as released to the network **140** from application providers and network administrators and moving it to the edgepoints **350a-b**. The policies include deployment and execution policies. Application states include the actual application data/binaries and the method to create the snapshots.

In one embodiment, the DDB **446** is the repository for the NOC **452** and serves also as the repository for deployment by the deployment center **444**. Conduit **360** provides an application provider with access to the network **140** to distribute, update and monitor their applications distributed throughout the edgepoints **350a-b**. In one embodiment, the conduit **360** abstracts or virtualizes the distributed nature of the network **140** and allows the application provider to update, manage and view their data and applications without being burdened by the location and load of the actual edgepoints **350a-b**.

In one embodiment, the dashboards provide at least two forms of data viewing, including immediate and aggregated. Immediate viewing allows a current, up-to-date view of an entity of the network **140**, such as edgepoints **350a-b**, global dispatcher **430**, deployment center **444** and conduit **360**. In one embodiment, immediate viewing is updated on a pre-
 5 defined schedule, periodically when a predefined change occurs or upon request. Aggregated viewing provides a cumulative temporal view of the entities of the network **140**, such as application instance usage, user patterns, edgepoint usage, etc. In one embodiment, immediate views are obtained by polling the edgepoints **350a-b** and conduits **360**. The aggregate view is accumulated at the deployment center **444**. In one embodiment, dashboards receive resource utilization information and determine an amount to charge each application
 10 provider based on the amount of resources utilized.

The NOC dashboard **454** allows network operators and controllers of network **140** to gain access to and view information and data relating to components on the network **140**. In one embodiment, NOC dashboard **454** allows access to
 20 components on the network at machine levels and application instance levels.

Customer dashboards **456** allow application providers to view the state of their outsourced applications, such as response time, data arrival rates, comp-utilities used, amount
 25 of compute resources utilized, cost per application, and other such information. In one embodiment, the network **140** prevents customer dashboards to gain access to the actual edgepoints **350a-b** and the applications stored and operated by the edgepoints **350a-b**.

In one embodiment, network **140** includes additional dashboards which allow other operators and users of the network access to information on and about the network. One example
 30 of an additional dashboard is an independent software vendor (ISV) dashboard which allows ISV's to view the usage patterns of applications on a per application provider or per application basis. This is an audit for the ISV's to understand how their applications are behaving in a real environment.

FIG. **18** depicts a simplified flow diagram of one implementation of one embodiment of a process **702** for an application provider to access and distributing applications onto
 40 the distributed system **140**. In step **704**, the application provider gains access to the system **140**. In one embodiment, the application provider gains access through conduit **360**. In step **706** the application provider dictates to the system the distribution of the application being deployed. As discussed above, the application provider is capable of limiting the distribution to specific geographic locations, to specific markets, to populated areas (such as only to resources located near metropolitan areas) and other such deployment distribution. The application
 45 provider can also allow an application to be distributed across the entire system **140**. In step **710**, the application provider specifies limits on the amount of compute resources to be utilized. The limit may be based on a monetary limit or other limits. In step **712**, the application provider dictates a maximum response time, such that when demand is such that the response time is exceeded, the system **140** will activate additional instances of a desired application. In step **714**, the application is distributed. In step **716**, entities, such as users, servers and computers are allowed to access the distributed
 50 applications.

FIG. **19** depicts a simplified flow diagram of one embodiment of a process **730** for an application provider to monitor and update applications distributed onto the system **140**. In
 55 step **732**, the application provider monitors the applications distributed. In one embodiment, the application provider is capable of monitoring the system through the dashboards **456**

and conduit **360**. In step **734**, it is determined whether the application provider wishes to update or change the distributed application or applications. If updates are to be made,
 60 step **736** is entered where the application provider submits the updates and the updates are distributed. If, in step **734**, updates are not to be made, or following updates, it is determined in step **740** whether the application provider wishes to replace a distributed application. If yes, then step **742** is entered where the application provider submits the replacement application and the replacement application is distributed. If, in step **740**, no replacement is needed, step **744** is entered where it is determined whether the application provider wishes to adjust the distribution of one or more application. If the adjustments are received and the adjustments to
 65 distribution are made in step **746**. If adjustments are not needed, step **750** is entered where it is determined if the application provider wishes to adjust resource limits, such as response time limits, monetary limits, capacity limits and other such limits. If yes the adjustments are received and implemented in step **752**.

FIG. **20** depicts a flow diagram of one embodiment of a process **770** for monitoring demand and determining an amount to bill an application provider. In step **772**, the on-demand system **140** monitors the demand for each application distributed on the system. In step **774**, it is determined whether the response time for an application exceeds limits. In one embodiment, the limits are default limits. In one
 70 embodiment the limits are previously specified by the application provider. If the response time limits are exceeded, the system determines if the system is at capacity, where capacity is dictated by several factors including, but not limited to compute resource availability, resource limits specified by the application provider, and other such factors. If the system is not at capacity, step **780** is entered where a new instance is restored where an instance of the application is restored and activated to allow entities to access the application. In step
 75 **782**, it is determined if demand exceeds a first threshold, where the threshold is defined as a number of users per instance or other such parameters. The threshold is a default threshold or defined by the application provider. If the demand exceed the first threshold, then step **784** is entered where it is determined if the system is at capacity (as described above). If the system is not at capacity then step **786** is entered where an instance of a desired application is activated to allow entities to continue to access the application without exceeding the threshold. In step **790** it is determined whether the demand is below a second threshold. If the demand is below the second threshold, at least one instance of an application will be snapshotted in step **792** to free up
 80 resources and reduce the compute resource cost to the application provider.

FIG. **21** depicts a simplified flow diagram of one implementation of one embodiment of a process **808** for determining an amount of resources utilized for an application and the amount to be charged to the application provider based on the amount of resources utilized. In step **810**, the compute resources providing application processing is monitored. In step **812**, the amount of resources utilized by each distributed application is determined. In step **814**, a total amount of resources utilized by each individual application provider all applications distributed by each application provider is determined. In step **816**, an amount to charge each application provider is determined based on the amount of compute
 85 resources utilized in application processing of all application distributed by each application provider. In step **818**, the

metered data is presented via a portal to application providers and partners (such as resellers and independent software vendors).

The capability to meter resource usage creates the further ability to develop pricing schemes which reflect the economic value or cost of providing service, including, for example, the opportunity cost of using compute resources for a given application. As an example, demand for computing resources during the business day may be higher than demand for processing during the night. The system provides a mechanism for pricing for peak usage versus off-peak usage.

The system facilitates a transfer pricing mechanism for computing resources. By metering resources used on an application level, the system enables a compute resource provider to determine how much resource usage is related to each specific application or customer. In one embodiment, this enables a corporation to allocate the cost of a centralized computing resource between different departments according to usage.

In one embodiment, more than one party may own the compute resources within the distributed network. In this case, the present invention enables the trading of compute resources between any number of compute resource suppliers and users. For example, a party may be an owner of compute resources that in the normal course of events provide sufficient capacity for its processing needs. This party can, by deploying the present invention and interconnecting its computing network with those of others, sell underutilized computing resources, and buy compute resources from other parties at times of peak demand. This enables parties to significantly improve the utilization of their computing infrastructure.

In the more general case, the present invention enables the development of an efficient market for trading computing resources, facilitating economic pricing. By way of example, a spot market could develop, better reflecting supply and demand of computing resources. Instruments for managing the financial risk of fluctuations in price on the spot market could then develop, for example forward contracts, options and derivatives.

In one embodiment, the novel system **140** is configured to provide at least six data paths including: a data path that connects an entity **124** to an application instance at an edgepoint **380**; a data path that sets up application instances for application providers; a data path which implements the snapshot/restore framework; a data path which provides a centralized view of edgepoints to application providers, the network provider and other such entities for monitoring the edgepoints and the compute resources utilized; a data path which provides database and file updates; and a path which prepares an application or plurality of applications for deployment and data synchronization.

As discussed above, the edgepoint **350** is capable of performing a plurality of actions or functions based on parameter and compute resource utilization information collected, such as performing snapshots of active application instances; restoring applications based on demand, response time and other parameters; effecting a move of an application instance; identifying optimal compute resources, such as an optimal compute module for routing a request; monitoring the performance and available capacity of the edgepoint to optimize performance and to signal the global dispatcher **430** when the edgepoint is operating at or near capacity; and monitoring the compute resources utilized per application instance such that the application provider is charged for the resources utilized in operating the application provider's distributed applications.

In one embodiment, the edgepoint **350** effects moves of application instances based on the results of an overload of a compute module, under-utilization of another compute module, and/or prioritization of one application instance over another (based for example, on distribution and prioritization specified by the system provider and the application providers).

The edgepoint **350** determines if the edgepoint is overloaded and hence notifies the global dispatcher **430** such that bind requests are re-routed back to the global dispatcher or initially routed by the global dispatcher to alternative edgepoints. In one embodiment, the resource manager **432** sends periodic load or resource utilization messages to the global dispatcher, such that the global dispatcher can accommodate the server weighting in the databases and memory.

The edgepoint further monitors, meters, and/or collects resource consumption information from application instances. This information is used by the dashboards **454**, **456** and for billing. In one embodiment, the information that is collected is logged into the databases or memory. The information collected by the edgepoint includes, but is not limited to, CPU usage, memory usage, disk usage, network bandwidth usage on a per application instance basis. In the case of CPU usage, information is collected at the software component level, providing a greater level of granulating than prior art systems. This information may be used to identify and allocate resources, manage partnerships and for usage based billing purposes.

The edgepoint additionally collects performance information such as application response time. In one embodiment, for each application instance, the resource manager **432** performs a periodic response time check. This information is also used to initiate the snapshot or move actions.

In one embodiment, the conduit **360** allows an application provider to create and distribute one or more application onto the network **140** producing the outsourced applications. The conduit performs a plurality of function including, but not limited to: determining cleave points; capturing distributed applications and associated data; capturing distribution policy, response time policy and other such policies as designated by the application provider; and test captures.

Cleaving includes a process of dividing one or more applications into applications that are distributed across the on-demand network **140**, and applications that are not to be distributed, and instead, for example, maintained by the application provider. One example is a website, where the site is cleaved to allow some of the application processing of the site to be handled by the distributed on-demand network, and some of the application processing of the site to be handled by the central site of the application provider. Thus, cleaving separates the applications that are outsourced by the application provider to be distributed over the present invention to take advantage of the on-demand compute resources.

The novel on-demand network **140** acquires and captures the applications and data associated with those applications through a capture process. The capture process analyzes how to bring up the necessary applications associated with a request, such as all the applications in a website operated from the on-demand system. The capture process further collects files and database data files for the application instances to satisfy a request. The capture process maps applications to application instances and documents the process of capturing the process of creating an application instance, and maps data to an application instance for data synchronization and captures.

25

In one embodiment, application and data capture is a process for determining how an outsourced application is constructed or produced. Some of the steps for application and data capture include:

- a) analyzing how to bring up applications in the network, such as bringing up applications configured to produce a web site;
- b) collecting the files and database data files for the operation of application instances;
- c) mapping applications to application instances and documenting the process of creating an application instance. In one embodiment, this is predominantly the data used by a packager (not shown) for initial deployment, and the instructions to start and stop application instances; and
- d) mapping data to an application instance, data synchronization and capturing the data synchronization components.

In one embodiment, the application provider dictates the distribution of the applications onto the distributed, on-demand network **140**. The application provider is capable of designating specific geographic areas for distribution, high traffic areas, such as specific metropolitan areas, and other such distribution. The application provider is also capable of designating the quantity of distribution, which will allow the application provider to limit the cost by limiting the compute resources utilized based on the distribution designation.

Policy capturing includes collecting deployment and execution policies. Deployment policies determine coverage information and are used by the deployment center **444** to aid in selecting the edgepoints **350** that will hold the outsourced applications. Execution policies relate to user-level SLAs and priorities for execution. Policy capture allows the application provider to limit and determine the potential cost spent by the application provider in utilizing the on-demand network.

In one embodiment, the on-demand network **140** is further capable of providing capacity testing of applications to aid the application provider to determine the accurate and operational capacity of application. One example is the testing of the capacity of a web site before allowing users to access the web site. Test capture includes a method, data and frequency to run tests on edgepoints before enabling the applications.

In one embodiment, the conduit **360** includes a studio module (not shown) which is a module used to perform application/data, policy and test captures in the conduit. The studio module includes at least six functional modules, including: a catalog module (not shown) which creates an inventory of deployed application; a camera module (not shown) which is the portion of the studio used to capture the process of bringing up and/or restoring an application, and establishing an initial snapshot; a packager module (not shown) configured to assemble an installed application and snapshots into a format (package) suitable for deployment; a publisher module (not shown) capable of transferring the packaged application to a deployment center; a cleaver module (not shown) which identifies the portions that are handled by the out-sourced applications, and will initiate data synchronization; and a policy editor module (not shown) configured to specify deployment policies. Application strength, deployment coverage and level of service are specified through the policy editor module. In one embodiment, the coverage is coarsely defined as a number of application instances, a number of edgepoints and/or which geographic location or locations. A level of service is also coarsely defined as a response time of an application.

The on-demand method and system **140** further provides remote control capabilities. The remote control feature pro-

26

vides: policy-based server management with the alignment to business objectives; deployment policies with application provider's selection of coverage, along with deployment and initiation timing; resource use policy to aid in obtaining the desired response time within application provider's budget constraints; and web-based policy editor.

Another advantage of the novel on-demand method and system **140** is providing application providers with direct access to the system **140** and allowing the application provider to make immediate changes or version updates to existing applications or site. Further, application providers are able to immediately load completely new applications or sites onto the system **140** without the need to bring the application or site down.

Another advantage of the present method and system **140** is the ability to provide fast rollback. Because of the design of the system **140** and ability to maintain applications in an inactive or snapshot state as an appshot **220**, prior versions of applications can be maintained on the edgepoints **350** while new versions are loaded onto the edgepoints **350**. If there is a glitch or error in the new version, the system **140** is able to quickly redirect the system **140** to reinstate the old version. Thus, avoiding catastrophic errors and glitches.

Another advantage provided by the novel on-demand method and system **140** is the ability to distribute users **124** to applications **356** throughout the system **140** thus spreading the load of the users. This provides the application provider with additional benefits which were not available through the prior art without enormous capital expenditures. One benefit of the on-demand system **140** is the ability to handle extremely large numbers of entities **124** at a single time because entities can be directed to application instances distributed throughout the system **140**. As an example, application and web providers have the ability to announce and broadcast an event which may attract abnormally large numbers of users without overloading the system. Because the users can be routed to edgepoints all over the system **140**, the maximum load on any given edgepoint will not exceed the capacity of the resources of the edgepoints. Thus allowing abnormally large numbers of entities to utilize or view the application or website. This is all achieved without the need for the application provider to purchase large numbers of servers and accompanying hardware, as well as the need to configure, load and maintain these large numbers of machines for such a limited surge in user load.

An additional benefit is that application providers are able to interact with abnormally large numbers of users instead of just broadcast to those users. Because the applications are distributed, the capacity to operate those applications is also distributed. Thus, allowing each instance of an application to utilize a larger amount of resources without exhausting the resources of the system. Thus, more interactive applications and sites are capable without the need for additional capital expenditures by the application provider. These are significant advantages provided by embodiments of the invention that are not available in conventional content delivery systems, networks, or methods.

As an example, big media companies have the capability through the present on-demand method and system **140** to now start getting a one to one opportunity with users accessing their applications and sites. As a comparison, television allows the distribution of a single program without any interaction. However, with the present method and system **140**, a plurality of different programs can be distributed while allowing direct interaction with the user without overloading a system and without cost prohibitive capital expenditures. As a further example, if a prior art application or site were to get

a million and a half simultaneous “views” of an event, there is no way, with prior art systems, to turn those “views” immediately into a million and a half purchases or registrations or any other interaction because a computer system large enough to handle that amount of a load at a single site would be too cost prohibitive. But with the present on-demand method and system **140**, the million and a half load is distributed throughout the world wide system of data centers, each housing a plurality edgepoints. Thus, instead of a single server or machine handling a million and a half simultaneous hits, the load is distributed across hundreds of edgepoints and/or servers, which results in thousands or less simultaneous hits per edgepoint **350**. A load of tens of thousands of simultaneous hits is manageable for a single server or machine. Thus, the benefits of distributing loads becomes apparent through the scalable, on-demand capacity provided by the present system **140**.

A further advantage of the present on-demand method and system is that the user maintains control over their own applications and websites which are deployed over the on-demand system. In prior art systems, the application provider owns their own servers, allowing the application provider with complete control over the application or site. The application provider knows exactly what is being provided. Further, the application provider has direct access to the single server allowing the application provider the ability to monitor the application, the load of the application or site, and the types of interaction occurring with the application or site. However, the prior art systems require the large up-front capital expenditure to initiate and maintain the servers. Further, the prior art systems have either too much capacity and thus wasted capital expenditure, or too little capacity and thus unsatisfied users.

In one embodiment, the present on-demand method and system **140** is designed to allow the application provider with direct control and access to their applications. With the added benefit of being able to monitor specific regions, the application provider has the ability to adjust their applications or websites according to feedback received from a specified region to more accurately address the needs and desires of users in that region by simply adjusting the instances of the appshot housed in edgepoints in those regions of interest. Further, the application provider is able to fully monitor the application. In one embodiment, this is achieved by allowing application providers to create different applications to be deployed geographically as desired. In one embodiment, the on-demand method and system includes: a web-based performance portal to allow the application provider comprehensive statistics on the virtual single server with the additional benefit of obtaining further web response time metrics; alerts based on set bands of acceptable performance; and expense monitoring based on the amount of resources used by the application provider including daily web-based bill review, and alerting of faster-than-expected spending.

Some of the safety precautions or security architecture provided by the novel on-demand method and system **140** are discussed below. The security architecture ensures that edgepoints run registered appshots **220** to prevent hackers from starting other applications; appshots **220** do not allow login, SetUserID, and other such conditions to prevent hackers from breaking out of the appshot control; appshots **220** access limited disk storage, memory, sockets and other such resources to protect user data; the conduit **360** and hub **324** authenticate each other before transfers of data or appshots; appshots **220** are compressed and encrypted when transferred from the conduit **360** to the edgepoints **350**; administration is authenticated and changes are audited. The system **140** also

prevents denial-of-service attacks because of the size of the distributed on-demand system **140** and the number of edgepoints **350**.

In one embodiment, the present on-demand method and system **140** utilizes links from other applications, such as application provider’s home or central web site, to route the user **124** to the desired application **356** stored and maintained on an edgepoint. The system allows an application provider to outsource only a portion of their applications to the on-demand system **140**, while still maintaining some of their application processing. For example, an application provider may outsource some of the applications to operate a web site, but, the application provider’s central site is still maintained by the application provider. In an alternative embodiment, an application provider outsources their entire application suite and sites, including their central site, to the on-demand system **140**. In one embodiment, the link or pointer from the central site points to a site maintained and controlled by the application provider, but is stored and operated from the resources of the on-demand system **140**. When the link or pointer is activated, the on-demand system **140** is accessed and the user **124** is routed to the most optimal edgepoint providing the application desired. In one embodiment, the optimal edgepoint is determined based on network latency and edgepoint load. If the loading on a first edgepoint **350a** is too great, the system will route the user **124** to a second edgepoint **350b** even though the second edgepoint **350b** maybe a further distance way from the user **124** than the first edgepoint **350a**. This rerouting is performed because it is worth taking additional latency delays along the routed path to get to the second edgepoint **350b** because the second edgepoint **350b** is under less load or stress and will provide a superior response, resulting in a superior response even with the added latency delay.

The present on-demand method and system **140** not only provides on-demand, distributed application processing, the on-demand method and system **140** also provides shared resources throughout the distributed on-demand system **140**. In one embodiment, because of the unique ability to store an application in a snapshot state, the present invention is also capable of removing an application from resources when the application is not being actively used, thus freeing up the resources. This allows an alternative application to be activated on those resources. Thus providing on-demand, distributed application processing through shared resources which reduces the cost of the resources because a plurality of application providers are utilizing the same resources. In one embodiment, the present invention provides for the ability to return an application not being used into a snapshot state to free up resources for other applications to utilize. Further, in one embodiment, when an active application is returned to a snapshot state freeing up resources, the application provider is no longer charged for the resources that the application was utilizing. The application provider pays for the amount of resources which are actually used by applications distributed by the application provider. The amount of consumed resources are measured in a variety of different ways including: the amount of processor usage; the amount of memory usage; the number of processors operating; the amount of network bandwidth usage; the number of appshots deployed; the density of appshot deployment; and any combination thereof.

Prior art or conventional systems and methods have been developed which distribute content to allow local routing of users. However, these conventional systems and methods do not provide for a method of communicating or returning processed information back to the main site. Prior art out-

sourced content providers do not provide for processing capabilities of the application providers specific applications. The present invention provides for the separation of applications, the outsourcing of those applications to distribute the load utilizing distributed resources allowing superior performance, without limiting the functions or processing of the applications.

In one embodiment, the present on-demand method and system **140** provides for the scheduling of website or applications to servers and/or resources. The inventive method and system are dynamic and real-time or near-real time. This scheduling of resources is an inversion of the prior-art paradigm of requiring an application to be dedicated to a single server. Typical prior-art systems are configured such that applications are implemented to run on fixed machines or servers. When a request to access an application comes in to a prior art system, the request gets routed to a waiting server. Therefore, the applications on such systems must be active at all times because requests cannot be predicted. Because these applications are fixed or tied to the machine or server, the prior-art server must also be kept running all the time.

The present method and system **140** provides for the dynamic scheduling of a website or application to be processed on demand by restoring an appshot to its running state. Thus, an application can be shut down or removed from server resources until a request for that application is issued. A snapshotted application **220** can be loaded from a shared memory into a server and accompanying resources in less than approximately five seconds and more usually in less than about three seconds, activating what was an idle server. These times are guidelines and not limitations. Thus, the present method and system **140** allows for instant activation of the application on substantially any chosen compute resource. Further, when the application is no longer being used, the present method and system **140** provides the capability to halt the application to free up the resources for another application. Therefore, the system **140** is able to provide economies of scale and favorable pricing to application providers. Attempting to try and achieve this capability through prior art systems is completely impractical because of the amount of time needed to take down an entire application to free up a server and resources along with the amount of time needed to install and activate a new application is completely prohibitive. Thus the present method and system **140** allows for the dynamic scheduling of applications to be processed on demand on optimal compute resources by restoring an appshot to its running state which reverses the paradigm of dedicating servers to applications. Batch processing is therefore well supported.

The on-demand network **140** provides the capability of handling more applications per CPU, computer, microprocessor and/or processor than is available through prior art computers, systems or networks by over provisioning the number of applications that are provided by the edgepoint and time multiplexing the use of these applications. In part, this is a result of the "bursting" nature of demand. This is achieved through the unique snapshot/restore ability of the network and edgepoint. Prior art systems cannot provide this capability because of the amount of time needed to take down an application and activate a new application, as well as the loss of data and current state information associated with an application at the time it is taken down. The system **140** provides for the unique ability to quickly halt an application, and store the application and associated states. The halting of the application is achieved without adversely affecting the application or adversely affecting the operation of the application when the application is reactivated or restored for operation. By

snapshotting an application, the edgepoint frees up the set of resources for an alternative application. Thus, the edgepoint can multiplex the access and operation of applications without adversely affecting the operation of the application, and without the application and user's knowledge.

In one embodiment, the on-demand method and system **140** is application oriented as apposed to process oriented. The on-demand method and system **140** provides for a virtualization layer in the operating system, such that an application is considered the object of interest, instead of considering the processes as the object of interest. Thus allowing the freezing or halting of an application, and the storage of the application stack, including the different processes, their interprocess communication and its state.

By enabling an application oriented processing network, the method and system enables a higher level of utilization of computing resources. This is a result of increasing the number of applications than can be handled per CPU coupled with the inherently variable nature of demand for computing resources. Prior art systems are characterized by very low average levels of utilization for computing resources. Because it is not possible to quickly take down an application and activate a new application, a single processing resource must necessarily lie idle when demand for the application tied to that resource diminishes.

Application demand varies according to application type, type and geographic location, among other variables. By way of example, demand for enterprise applications usually peaks during business hours whereas demand for consumer-centric web sites may peak during evening hours. Peak demand times will be different in different time zones. Further, demand for processing for certain applications is less time-dependent than others. The present invention enables less time-critical applications (such as batch processing) to be run when demand for more time-critical applications (such as web applications) is low. Because the technology enables the sharing of processors between different applications, this leads to improvements in utilization levels.

FIG. **22** depicts typical exemplary demand situation for two different applications (or customers) across or over a twenty-four hour time period. Demand for Application 1 peaks at **1400** whereas demand for Application 2 peaks at **1900**. With prior art systems, the amount of processing capacity that would be required to satisfy the total demand for both customers or applications is the sum of the total amount of processing capacity that would be required for each of the applications individually (in this case **200** units). With the present invention, the total amount of processing capacity required is equal to the maximum aggregated demand in any time period, in this case **160** units. This is a direct result of the inventive system and method's ability to quickly take down and set up applications resulting from the snapshot/restore technology.

The novel on-demand application processing method and system **140** creates a completely new economic model. The present invention further provides a new technology and method to share compute resources. Still further, the present invention provides the technology and method to bring compute capacity on demand very quickly. The present method and system **140** also provides for dynamic server allocation and resource sharing. Thus, providing on-demand resources at significantly reduced cost to the application provider.

It will be appreciated in light of the description provided herein that the inventive system, method, business model, and operating service moves the provisioning of utility-based or computing utility services to the next level of services, that of a true utility service. Aspects of the invention provide cus-

31

tomers the ability to buy just what they need without being trapped into having to pay for capacity they don't use. Essentially computing services are moved from buying fixed capacity, the traditional outsourcing model, to buying variable capacity.

The inventive system and method provides on-demand computing solutions that enable enterprises to improve the server efficiency, application performance and financial return of their information technology (IT) environments. Leveraging an embedded software platform, the inventive system and method enables server infrastructure to be shared securely across applications to offer a range of computing infrastructure management, provisioning and operations solutions to enterprises and service providers with significant application investments. By dynamically allocating, retrieving and tracking computing resources, the inventive system and method enables the first true computing utility. In one embodiment, the system and method provide a service platform offering on-demand web application processing using a utility-based pricing model.

While various aspects of the system and method of the invention have already been described, FIG. 23 provides a diagrammatic illustration showing an exemplary embodiment of a system according to the invention. The diagram illustrates relationships between the Internet (or other network) and routers, distribution nodes (DN), Ethernet Hub, Administrative Nodes (AN), computer nodes (CN) and SAN Hub with associated disk array. Also illustrated are the global dispatcher (GP), deployment center, and NOC. Users and a main site are also coupled to other elements of the system over the Internet or other network connection. An NOC dashboard and a customer dashboard are also provided in this particular system configuration.

The foregoing description of specific embodiments and examples of the invention have been presented for the purpose of illustration and description, and although the invention has been illustrated by certain of the preceding examples, it is not to be construed as being limited thereby. They are not intended to be exhaustive or to limit the invention to the precise forms disclosed, and obviously many modifications, embodiments, and variations are possible in light of the above teaching. It is intended that the scope of the invention encompass the generic area as herein disclosed, and by the claims appended hereto and their equivalents.

Having disclosed exemplary embodiments and the best mode, modifications and variations may be made to the disclosed embodiments while remaining within the scope of the present invention as defined by the following claims.

What is claimed is:

1. A method for providing distributed, on-demand application processing, comprising:

allowing a first application provider to deploy at least a first application onto a network, wherein the network includes distributed compute resources configured to provide application execution;

receiving a request for execution of the first application from the first application provider; and utilizing the distributed compute resources for execution of the first application in response to receiving the request for execution of the first application from the first application provider;

metering and monitoring an amount of the compute resources utilized in execution of the first application; charging the first application provider based on the amount of compute resources utilized in execution of the first application;

32

increasing the amount of compute resources utilized in execution of the first application by a first set of compute resources; and increasing the amount charged to the first application provider based on the first set of compute resources.

2. The method as claimed in claim 1, further comprising: allowing the first application provider to update the first application.

3. The method as claimed in claim 2, further comprising: allowing the first application provider to monitor the first application.

4. The method as claimed in claim 3, further comprising: allowing the first application provider to replace the first application with a second application.

5. The method as claimed in claim 1, further comprising: allowing an entity to access the compute resources over the network to interact with the first application.

6. The method as claimed in claim 1, wherein increasing the amount of computer resources includes increasing the amount of compute resources due to an increased demand for the first application.

7. The method as claimed in claim 1, further comprising: reducing the amount of compute resources utilized in execution of the first application by a second set of compute resources; and reducing the amount charged to the first application provider based on the second set of compute resources.

8. The method as claimed in claim 1, further comprising: providing a second application provider with access to the network;

allowing the second application provider to distribute at least a third application onto the network;

allowing an entity to access the compute resources over the network to interact with the third application;

monitoring the amount of the compute resources utilized for providing application processing of the third application; and

charging the second application provider based on the amount of compute resources utilized in processing the third application.

9. A method of providing on-demand computational resources over a distributed network, the method comprising: providing an application provider with access to a distributed network;

through the network the application provider dictating at least a first portion of the distributed network to receive at least one application; and

distributing the application onto computational resources within the first portion of the distributed network dictated to receive the application.

10. The method as claimed in claim 9, further comprising: the application provider dictating a limit on an amount of the computational resources to be utilized in providing at least one entity with access to the application for processing of the application.

11. The method as claimed in claim 10, further comprising: through the network, the application provider updating the first application.

12. The method as claimed in claim 10, further comprising: metering and monitoring the amount of computational resources utilized in providing the at least one entity with access to application; and

charging the application provider based on the amount of computational resources utilized in providing the at least one entity with access to the application.

13. The method as claimed in claim 12, further comprising: metering and monitoring demand for the application; and

33

adjusting the amount of computational resources utilized in providing the at least one entity with access to the application based on the demand.

14. The method as claimed in claim 13, wherein adjusting the amount of computational resources utilized comprises not exceeding the limit of the computational resources dictated by the application provider.

15. The method as claimed in claim 13, wherein adjusting the amount of computational resources utilized further comprises:

initiating an additional instance of the application if the demand for the application exceeds a first threshold; and halting an instance of the application if the demand for the application falls below a second threshold.

16. The method as claimed in claim 15, wherein initiating an instance of the application comprises restoring a snapshotted application; and wherein halting an instance of the application comprises snapshotting an active application.

17. The method as claimed in claim 10, further comprising: the application provider monitoring the amount of computational resources utilized in providing the at least one entity with access to the application; and the application provider adjusting the limit on the amount of computational resources available to provide the at least one entity with access to the application.

18. The method as claimed in claim 17, wherein: the first threshold is based on response time of the application.

19. An apparatus for providing on-demand compute resources, comprising:

a plurality of compute resources including at least one memory, wherein the plurality of compute resources are distributed across a network, wherein the plurality of compute resources are coupled to allow communication between at least first compute resources and second compute resources; a conduit coupled to the network, and

configured to provide an application provider access to the network to distribute at least a first application onto the network and request execution of the first application using the plurality of compute resources;

a first resource manager coupled with at least the first compute resources, wherein the first resource manager is configured to activate at least a first set of the first compute resources for processing of the first application when demand for the first application exceeds a first threshold;

and a metering module coupled with the first compute resources, wherein the metering module is configured to monitor the first compute resources including the first set of the first computer resource being activated for processing of the first application, and wherein the metering module is configured to determine an amount to bill the first application provider based on the first set of the first compute resources utilized;

wherein the first resource manager is configured to activate a second set of the first compute resources for processing

34

of the first application when demand for the first application exceeds a second threshold;

and wherein the metering module is configured to monitor the second set of the first computer resource being activated for processing of the first application, and wherein the metering module is configured to determine an increased amount to bill the first application provider based on the second set of the first compute resources utilized.

20. The apparatus as claimed in claim 19, wherein: the conduit is further configured to provide the application provider with access to the network to update the first application.

21. The apparatus as claimed in claim 20, wherein: the conduit is further configured to provide the application provider with access to the network to replace the first application with a second application.

22. The apparatus as claimed in claim 20, wherein: the first compute resources includes at least a first and second set of compute resources, where the first and second set of the first compute resources are configured to be activated and deactivated based on a demand and to provide application processing for at least the first application.

23. The apparatus as claimed in claim 22, further comprising wherein:

the first resource manager coupled with the first compute resources, and is configured to monitor demand for at least the first application such that the resource manager activates and deactivates at least one of the first and second sets of the first computer resources based on demand.

24. The apparatus as claimed in claim 23, wherein the first resource manager is further configured to monitor the amount of the first compute resources utilized in processing the first application; and the apparatus further comprising:

wherein the metering module coupled with the first resource manager, and is configured to receive the amount of the first compute resources utilized and to determine an amount to bill the first application provider based on the amount of the first compute resources utilized.

25. The apparatus as claimed in claim 19, wherein: the first resource manager is configured to deactivate one of the first and second sets of the first compute resources processing the first application when the demand for the first application falls below a third threshold; and

the metering module registers a decreased amount of compute resources utilized for application processing of the first application based on the deactivation of one of the first and second sets of the first computer resources being deactivated, and the metering module is configured to determine a reduce amount to billed the first application provider based on the decreased amount of the first compute resources utilized.

* * * * *

UNITED STATES PATENT AND TRADEMARK OFFICE
CERTIFICATE OF CORRECTION

PATENT NO. : 7,596,784 B2
APPLICATION NO. : 09/950559
DATED : September 29, 2009
INVENTOR(S) : Abrams et al.

Page 1 of 1

It is certified that error appears in the above-identified patent and that said Letters Patent is hereby corrected as shown below:

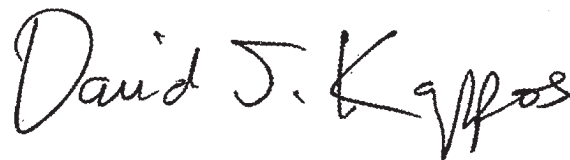
On the Title Page:

The first or sole Notice should read --

Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b)
by 2232 days.

Signed and Sealed this

Twenty-eighth Day of September, 2010

A handwritten signature in black ink that reads "David J. Kappos". The signature is written in a cursive style with a large, stylized 'D' and 'K'.

David J. Kappos
Director of the United States Patent and Trademark Office