REMARKS

Claims 46-67 were pending. Claim 64 has been cancelled. New claim 68 has been added.

Claim 53 has been rejected under 35 USC 112, second paragraph. Claim 53 has been amended to remove the term "at arbitrary time intervals", and thus is believed to conform with 35 USC 112. No new matter has been added.

Claims 46 and 55 have been rejected under 35 USC 103(a) as being unpatentable over Daoud et al. (US 7,984,147), in view of Zuniga (US 2005/0245243), in view of Hunter et al. (US 7,242,809), and further in view of Griffin (US 2007/0124331). This rejection is respectfully traversed.

In particular, it is respectfully submitted that Daoud et al. do not teach computer readable code, which when executed by a computer, causes said computer to *send a request to* a network-based server, the request including a unique identifier for identifying an audio stream, and to load a list of library servers *received from the* network-based server, the list of library servers determined in dependence upon the unique identifier, as defined in claim 46.

Daoud et al. teach program code for selecting a requested level of service for a transaction 200 and program code for assigning the requested level of service. Referring to column 8, lines 31-50, Daoud et al. teach that the transaction 200 is received at a load balancer 300, which reads the requested level of service and selects a server (e.g., 512) based on a server index 600. Referring to column 6, lines 22-50, the server index is a multi-dimensional array stored in memory accessible by the load balancer, and is used to determine the server in the server pool that can best provide the requested level of service. The server pool is managed by the load balancer 300.

Daoud et al. do not teach computer readable code, which when executed by a computer, causes said computer to *send a request to a network-based server* and to *load a list of library servers received from the network-based server*. In fact, since the server index taught by Daoud et al. is stored in memory accessible by the load balancer 300, and appears to be managed by the load balancer 300, it is respectfully submitted that the program code of the load balancer 300 already has access to the server index and thus does not *send a request to a network-based server* and *load a list of library servers received from the network-based server,* as defined in claim 46 of the instant application. For example, it is respectfully submitted that the load balancer software cannot load a list of library servers *received* from a network-based server because the load balancer manages/creates the server pool/server index. In addition, it is respectfully submitted that one of ordinary skill in the art would never interpret the memory accessible by the load balancer 300 to be a network-based server, since at column 7, lines 22-30 Daoud et al. define a server as any computer or device that manages resources (e.g., which memory alone cannot do). It is further submitted that the one of ordinary skill in the art would never interpret the program code for selecting a requested level of service for a transaction 200 and program code for assigning the requested level of service to be computer readable code, which when executed by a computer, causes said computer to *send a request to* a network-based server, and to load a list of library servers *received from the* network-based server. For example, referring to column 6, lines 45-55, only the load balancer accesses the server index (i.e., the program code for assigning the requested level of service never interacts with the server index).

Notably, providing computer readable code, which when executed by a computer, causes said computer to *send a request to a network-based server* and to *load a list of library servers received from the network-based server*, provides client-based performance management. As discussed in paragraphs [65]-[67] of the instant application, client-based performance management is an important factor in ensuring the integrity of the audio stream available to the user. Advantageously, since the server statistics are created and maintained in the client only, the client software selects the server using performance data that is specific to the client. For example, as discussed in paragraph [67], this client-based performance management allows the time of operations such as logging in, getting the file, and/or getting the file size to be used to select the server. More specifically, the entire time for the transaction (e.g., from the original

request to a response from the request) is used to select the server. Accordingly, this client-based performance management is able to balance network and server loads on the basis of performance defined by the client. In contrast, Daoud et al. only teaches resource managing and does not provide client-based performance management wherein the transaction is evaluated/monitored from the client side (i.e., the speed of the transaction between the load balancer and the origin of the transaction is ignored).

Since Daoud et al. do not teach computer readable code, which when executed by a computer, causes said computer to send a request to a network-based server, the request including a unique identifier for identifying an audio stream, and to load a list of library servers received from the network-based server, the list of library servers determined in dependence upon the unique identifier, as defined in claim 46, it is respectfully submitted that the combination provided by the cited references does not teach the combination of elements found in claim 46 and that a *prima facie* case of obviousness has not been established. Accordingly, claim 46 and claims 47-57, which depend therefrom, are believed to be patentable.

In addition, it is respectfully submitted that modifying the combination of Daoud, Zuniga, and Hunter in view of Griffin does not provide the invention defined in claim 46, for the following reasons.

First, as discussed *supra,* Daoud et al. teach a first program code for selecting a requested level of service for a transaction for assigning the requested level of service and a second program code used by the load balancer. It is respectfully submitted that the first and second program codes are different codes and are provided on separate non-transitory computer readable storage media. For example, as is well know to those of ordinary skill in the art, load balancing is typically provided by dedicated software or hardware (e.g., a load balancing engine) that is disposed far from the origin of the transaction (e.g., is often coupled to a port where external clients connect to access services). Further support that the first and second program codes are provided on separate non-transitory computer readable storage media is found in Daoud at column 6, lines 45-50, wherein it is stated that the transaction is received by the load balancer, and that the service tag is read using suitable program code. Since the first and second program

codes are provided on separate non-transitory computer readable storage media, it is respectfully submitted that the cited combination cannot teach *a non-transitory computer readable storage medium* including computer readable code, which when executed by a computer, causes said computer to: *send a request to a network-based server*, *load a list of library servers received from the network-based server*, and *download a first digital audio file from the plurality of digital audio files for playback with a media player*, as defined in claim 46. For example, if the first program code for selecting a requested level of service for the transaction is modified in view of Griffin to download a first digital audio file from the plurality of digital audio files for playback with a media player, the combination will not provide program code used to load a list of library servers (i.e., as discussed *supra*, the server index taught by Daoud is only accessible by the load balancer). In contrast, if the second program code used by the load balancer is modified in view of Griffin to download a first digital audio file, the combination will not provide computer readable code that causes said computer to *send a request to* a network-based server and to load a list of library servers *received from the* network-based server (e.g., as discussed *supra* the load balancer does not receive the list of library servers from a network-based server). Furthermore, it is respectfully submitted that one of ordinary skill in the art would never modify the second program code used by the load balancer to download a first digital audio file because the load balancer is used solely for managing the internet load (e.g., in a server farm) in a transparent manner and because it would be pointless for a load balancer to download an audio file for playback with a media player. In fact, modifying the program code of a load balancer to download an audio file for playback with a media player would slow down the transaction, which is in direct contrast to the focus of Daoud et al. (e.g., to improve service using level of service assigned).

Second, referring to paragraph [0020], Griffin teaches downloading and storing content files, each of which has a bookmark associated therewith. It is respectfully, submitted that one of ordinary still in the art would understand that each of these content files represents a single, complete unit (i.e., that the content files stored on the content server taught by Griffin are not digital audio files including different segments of an audio stream). Accordingly, it is respectfully submitted that one of ordinary skill in the art would never interpret Griffin to teach "download a first digital audio file from the plurality of digital audio files for playback with a

media player, each digital audio file in the plurality of digital audio files including a different segment of the audio stream" as defined in claim 46 of the instant invention. It is further submitted that one of ordinary skill in the art would not find it obvious to modify the teachings of Daoud, Zuniga and Hunter in view of Griffin, because Griffin teaches away from providing "a plurality of digital audio files for playback with a media player, each digital audio file in the plurality of digital audio files including a different segment of the audio stream", by specifying that a separate bookmark file is created for each content file (e.g., see paragraph [0020]). Accordingly, claim 46 and claims 47-57, which depend therefrom, are believed to be patentable.

In addition, with specific regard to claim 55, it is respectfully submitted that the cited combination does not teach "download a first digital audio file from the plurality of digital audio files for playback with a media player, each digital audio file in the plurality of digital audio files including a different segment of the audio stream" and "download a second other digital audio file from a second library server for playback with the media player." More specifically, it is respectfully submitted that no combination of the cited references teaches downloading different segments of an audio stream from different libraries.

With specific regard to claim 57, it is respectfully submitted that Arons does not teach small digital audio files and thus cannot teach "wherein the computer determines the first digital audio file for playback using a time offset external to the descriptor file and the at least one of the start time, end time, and play time of each digital audio file in the plurality of digital audio files." For example, referring to section 3.9 on page 20 of the SpeechSkimmer document, Arons teaches that a *single file* is created that contains all of the segmentation data. In the same section, on page 21, it is specifically stated that audio data are read from *the* sound file.

Applicant would like to thank the Examiner for indicating that claim 64 would be allowable if rewritten in independent form, including all of the limitations of the base claim and any intervening claims. The subject matter of claim 64 has been written in independent form as amended claim 58. In addition, amended claim 58 corrects the phrase "resident with the computer" to --resident within the computer--. Claim 64 has been cancelled. No new matter

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.