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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
4 December 2003 (04.12.2003)

PCT

(10) International Publication Number
WO 03/100630 A1

(51) International Patent Classification[7]: G06F 15/00

(21) International Application Number: PCT/US03/16990

(22) International Filing Date: 28 May 2003 (28.05.2003)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
10/156,443 28 May 2002 (28.05.2002) US
10/156,390 28 May 2002 (28.05.2002) US

(71) Applicants: **TOSHIBA CORPORATION** [JP/JP]; 6-78, Minami-cho, Mishima-shi, Shizuoka 411-8520 (JP). **TOSHIBA TEC KABUSHIKI KAISHA** [JP/JP]; 6-78, Minami-cho, Mishima-shi, Shizuoka 411-8520 (JP).

(71) Applicant and
(72) Inventor: **SEIFI, Mustafa** [IN/US]; 7 Carpenterra, Irvine, CA 92602 (US).

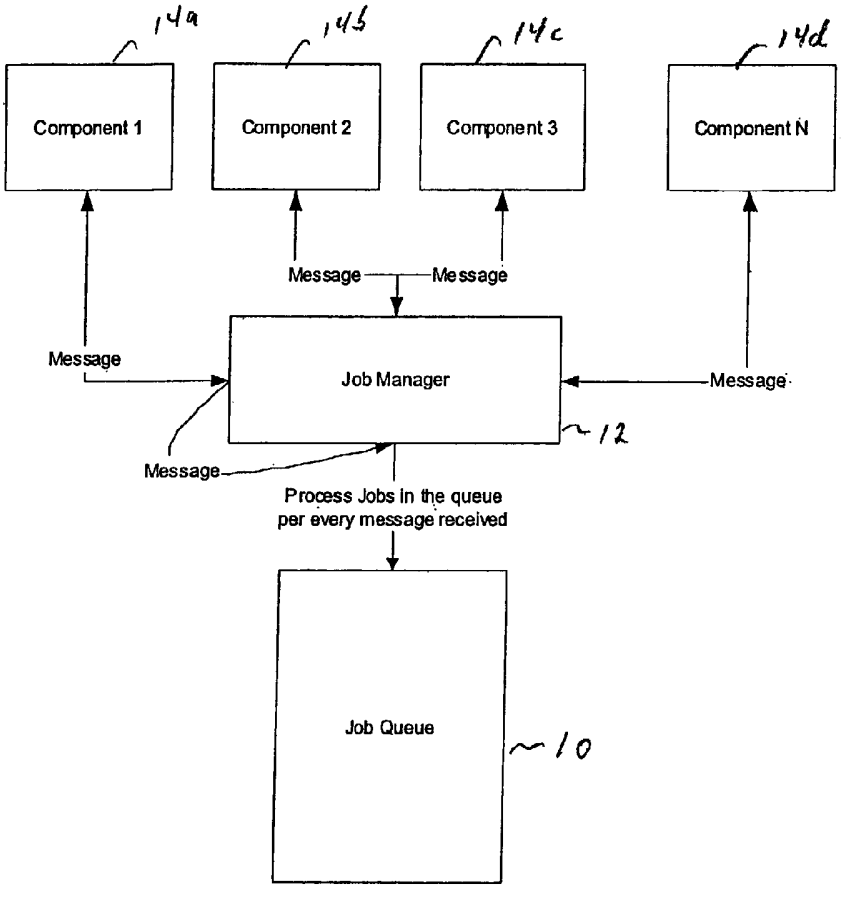
(74) Agent: **MIZER, Susan**; Arter & Hadden LLP, 1100 Huntington Building, 925 Euclid Avenue, Cleveland, OH 44115-1475 (US).

(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NI, NO, NZ, OM, PH, PL, PT, RO, RU, SC, SD, SE, SG, SK, SL, TJ, TM, TN, TR, TT, TZ, UA, UG, UZ, VC, VN, YU, ZA, ZM, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IT, LU, MC, NL, PT, SE, SI, SK, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

[Continued on next page]

(54) Title: METHOD AND IMPLEMENTATION FOR MESSAGE-DRIVEN JOB PROCESSING



(57) Abstract: A method and implementation for job managing which enables the creating of a job in a client system and sending a job creation instruction message to a job manager (12) to notify it that a new job has entered a processing queue (10). At least one processing message is sent from the job manager (12) to at least one respective component (14a, 14b, 14c, or 14d). Another aspect is a relational job request queue (318) preferably for digital imaging devices (310). At least one parent queue data store is created for storing a plurality of parent queue records (106) and at least one child queue data store is created for storing a plurality of child queue records wherein each child queue records (108)wherein each child queue record comprises a parent record identifier data store.

WO 03/100630 A1

Published:
— *with international search report*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

METHOD AND IMPLEMENTATION FOR MESSAGE-DRIVEN JOB PROCESSING

Background of the Invention

This invention pertains generally to job request queues, and more specifically to the implementation of a relational hierarchical job request queue.

In a network, it is common to process jobs, e.g. printing jobs, in a queue. Jobs are assigned a position in the queue and a network device, e.g. a network printer, processes each of the jobs in a predetermined order. Fort example, the jobs can be processed in the order they have been received. Also, the jobs can be processed in accordance with priority, with high priority jobs being given preference.

In a network, the operation of the queue can be implemented in a number of different ways. In one common previous-type method of processing jobs in a queue, the system constantly loops through all the jobs in the queue to insure that the jobs are being processed quickly. However, this method requires that the queue be monitored at all time, even when no jobs are awaiting processing. This approach therefore uses up lots of CPU time by monitoring the queue at all times, and therefore inefficiently manages system resources.

In another previous-type method of job processing, a "daemon" process can be implemented that "wakes up" in predefined time intervals and processes any jobs that are waiting in the queue. While this method puts less demand on the CPU, it suffers from the drawback that jobs may sit idle for an undesirable period of time until the daemon process wakes up to service the waiting jobs. In this way, this approach is inefficient at managing the performance of the system.

In addition to the aforementioned problems, there is a fundamental need to provide service for a variety of types of job requests in client-server systems. Modern digital imaging devices ("DID") are often capable of performing a variety of functions. DIDs suitably include devices such as printers, fax machines, scanners, copiers, multi-function peripherals ("MFPs"), and other like peripheral devices. Furthermore, a single computer can often access a plurality of different types of DIDs, which means that a single computer is capable of generating various types of job requests with various destinations. There exists a need to organize the various job requests sent from a user to any one of a plurality of types of DIDs connected to a network.

There also exists a need to provide both clients and administrators in a client-server environment with the ability to perform management functions on job request queues. It is

preferable if clients have the ability to specify how a job should be processed. It is also preferable if clients have the ability to monitor jobs in a job request queue.

It is also preferable if administrators have the ability to create queues, assign priorities to queues, relate parent queues to child queues, move jobs between queues, etc.

Summary of the Invention

The difficulties and drawbacks of the previous-type approaches are resolved by, in one aspect disclosed herein, the method and implementation for job managing of the present invention which enables the creation of a job in a client system and sending a job creation instruction message to a job manager to notify that a new job has entered a processing queue. At least one processing message is sent from the job manager to at least one respective component. A first job state change message is sent from the respective component to the job manager to notify the job manager when the component begins to process a given job. A second job state change message is sent to the job manager from the component to notify the job manager when the component is done processing the given job.

Another aspect of the present invention, comprises a method for storing received job requests. The method comprises the steps of creating at least one parent queue data store for storing a plurality of parent queue records, wherein each record comprises a parent queue record identifier data store that is configured for storing a parent queue record identifier that is unique at least within the parent queue data store. The method further comprises the step of creating at least one child queue data store for storing a plurality of child queue records. Each of the child queue records suitably comprises a child queue record identifier data store and a parent queue record identifier data store. The child queue record identifier is preferably configured for storing a child queue record identifier that is unique at least within the child queue data store and. The method further comprises the step of creating a job request data store for storing a plurality of job request records, wherein each record comprises a job request record identifier data store that is configured for storing a job request record identifier that is unique at least within the job request data store. Also, the method comprises the step of creating for each received job request a record in the job request data store and a related record in at least one of the parent queue data store or child queue data store.

Also according to the present invention, there is provided a relational queue system for storing job requests a client-server network environment. The system comprises a data transport network, at least one client communicatively coupled to the data transport network, and at least one digital imaging device communicatively coupled to the data transport network. Also communicatively coupled to the network is a server comprising a job request table for storing information relative to job requests received from the at least one client, a parent queue table for organizing and servicing job requests, and at least one child queue table for organizing and servicing job requests, wherein the job request table, parent queue table, and the at least one child queue table for a relational database.

As will be realized, the invention is capable of other and different embodiments and its several details are capable of modifications in various respects, all without departing from the invention. Accordingly, the drawing and description are to be regarded as illustrative and not restrictive.

Brief Description of the Drawings

Fig. 1a is a block diagram showing the process flow of the job manager in accordance with the present invention.

Figure 1b illustrates a diagram of the organizational structure of a relational queue system according to the present invention;

Figure 2 illustrates a flow chart generally representing a method for creating a relational queue and storing received job requests according to the present invention; and

Figure 3 illustrates the present invention in a network environment.

Detailed Description of the Invention

The present invention includes a job manager that ensures continuous and timely attendance to the jobs in the queue. In the present method and implementation, as particularly shown in Fig. 1a, a job (e.g. a printing job) is created by a client system (not shown), and a job

creation instruction message (not shown) is sent to the job manager 12 to notify that a new job has entered the queue by the Job Queue 10. The message can be sent and received using any type of conventional messaging system. The job manager 12 then sends processing messages to one or more components 14a, 14b, 14c 14d that are assigned to process the jobs in the queue 10 for each specific client. A job state change message is sent from a component 14a, 14b, 14c, 14d to the job manager 12 to notify the job manager 12 when the component 14a, 14b, 14c, 14d begins to process a given job. Another job state change message is sent to the job manager from the component to notify the job manager when the component is done processing the given job.

Any "job create" and "job state change" message to the job manager 12 will cause the job manager 12 to process all the jobs in the queue 10. As long as components 14a, 14b, 14c, 14d continue to process of their respective jobs and notify job manager 12 of their progress the jobs in the queue will be processed. If the processing of a job is interrupted for any reason, e.g. if a printer runs out of paper, an "interrupt" message is sent from the component 14a, 14b, 14c, 14d to the job manager 12, who then reports the interruption to a user or system administrator. All other jobs in the queue will be processed normally.

To ensure communication continuity between the job manager 12 and other components 14a, 14b, 14c, 14d, the job manager 12 will receive a message from a thread within its process based on a configurable amount of time (T). This message will cause the job manager 12 to process all the jobs in the queue 10 if the current time (CT) minus T is greater than the last time (LT) when the job manager received a "job create" or a "state change" message. The above condition can also be written as: job processing if ((CT – T) > LT).

Another aspect of the present invention provides a method and system for implementing a relational queue that is preferably compatible with a plurality of operating systems.

Turning to **Figure 1**, a diagram of the organizational structure of a relational queue system according to the present invention is provided. The system comprises at least three data stores, a job data store **102**, a parent queue data store **106** and a child queue data store **108**. The child queue data store is preferably related to at least the parent queue data store and the data stores preferably form a relational structure. In the presently preferred embodiment, the data stores are embodied as tables in a relational database, although it will be appreciated by those skilled in the art that flat files, markup based files such as XML, or other embodiments are also suitably used. For clarity purposes, the system will be described in terms of a relational database.

The job data store **102** preferably exists as a table in a relational database. It suitably comprises a plurality of records and each record suitably comprises a plurality of data stores. In database terms, the job table comprises a plurality of records having a plurality of fields. The records of the job data store **102** preferably comprises an identifier **110** that is unique at least within the job data store. This identifier **110** is suitably created as an auto-increment type field in a database.

Records stored in the job data store **102** also preferably comprise a job type data store **114**. The job type data store **114** preferably stores information related to the type of job request that is received and stored as a record in the job data store **102**. The job type data store **114** preferably stores data representing job types selected from the group consisting of: print job, private print job, fax job, scan job, document management job, etc.

Records stored in the job data store **102** also preferably comprise a job name data store **112**. The job name data store **114** preferably stores descriptive information identifying a job request record. The descriptive information is suitably generated by automation procedures or manually.

Records stored in the job data store **102** also preferably comprise a job destination data store **138**. The job destination data store **138** preferably stores descriptive information identifying a specific destination for a received job request. This information suitably identifies which component(s) will service a job request. As shown, the destination data store **138** is located in a job ticket data store **104**. Preferably, the job ticket data store **104** is related to the job data store **102** in a one-to-one fashion. As such the information stored in the job ticket data store **104** is also suitably stored in the job data store **102**. The job ticket data store **104** suitably comprises information relative to the manner in which the received jobs are to be serviced.

Records stored in the job data store **102** also preferably comprise a job priority data store **120**. The job priority data store **120** preferably stores information related to the job priority. The job priority data store **120** preferably stores data selected from the group consisting of integers ranging from 0 to 10.

In addition, records stored in the job data store **102** also preferably comprise a job lifespan data store **122**. The job lifespan data store **122** preferably stores date and time information. The stored date and time information suitably identifies a span during which a job must be serviced.

In addition, records stored in the job data store **102** preferably comprise a job status data store **124**. The job status data store **124** preferably stores status information. The stored status information suitably is preferably data selected from the group consisting of: spooling, printing, scanning, faxing, private printing, ripping, creating jobs, etc.

Furthermore, the records stored in the job data store **102** preferably comprise a job creation time data store **126** and job completion time data store **128**. These data stores preferably store both time and date information, which is suitably generated through automated means.

Each of the records of the parent queue data store **106** and child queue data store **108** preferably comprise an identifier **140** and **156** respectively that is unique at least within the respective queue data store. Furthermore, because the parent queue data store **106** and child queue data store **108** are preferably related, the child queue data store records preferably comprise a parent queue record identifier data store **160**. It should be noted that the present invention is suitably embodied in a single queue data store wherein the records in the data sgtore are related to one another through a parent queue data store. Also, the records stored in the parent queue data store **106** and child queue data store **108** preferably comprise a type data store **146** and **162**, respectively.

It should be noted that while the system illustrated in **Figure 1** details a plurality of data stores, each configured to store a plurality of data fields or data stores, additional information is also suitably included in the data stores and additional data stores are suitably included without departing from the scope of the present invention.

Turning now to **Figure 2**, a flow chart for creating a relational queue and storing received job requests is provided. The basic flow commences at start block **200**, from which progress is made to process block **202** wherein a job request data store is created. Flow continues to process block **204** wherein a parent queue data store is created. Progression then flows to process block **206** wherein a child queue data store is created. The newly created child queue data store is preferably related to the parent queue data store created at process block **204**.

Progression continues to process block **208** wherein an incoming job request is received, after which progression flows to process block **210**. At process block **210**, a new record is suitably created in the job request data store created at process block **202**. Progression then continues to process block **212** wherein a new record is created in at least one of the parent and child queues created at process blocks **204** and **206** respectively. Flow then proceeds to

termination block **214** where the process terminates.

Turning now to **Figure 3**, a system diagram illustrating a relational queue in a network environment in accordance with the present invention is provided. The system comprises a data transport network **300** illustrative of a LAN or WAN environment in which a preferred embodiment is provided. The network **300** is suitably any network and is suitably comprised of physical and transport layers such as illustrated by a myriad of conventional data transport mechanisms such as Ethernet, Token-Ring™, 802.11(b), or other wire-based or wireless data communication mechanisms as will be apparent to one of ordinary skill in the art. Connected to a data transport network **300** are a Client **302** and a Server **310**. In the presently preferred embodiment, the Server **310** is a DID, although it will be appreciated by those skilled in the art that any server for accommodating selective query support, selective data access, data archiving, and the like.

One or more Clients, such as representative Client **302**, is also placed, or selectively placed, in data communication with the data transport system **300**. The Client **302** is preferably configured to interact with Server **310** as will be appreciated by one who is skilled in the art. It should be noted that the Client **302** is suitably a Thick Client or a Thin Client, additional server(s), personal digital assistant ("PDA"), or any equipment capable of interacting with Server **310** to send and receive data. Thus, a data path between Server **310** and the one or more Clients, such as representative Client **302**, are in shared data communication. The Client **302** is preferably in data communication with a data transport network **300** through a network interface **308**. The Client **302** preferably comprises storage, which is preferably a hard disk and random access memory ("RAM"), as will be appreciated by those skilled in the art. Stored on the storage is preferably an operating system **306** for controlling the Client **302**. The operating system **306** is suitably any operating system, such as Windows (XP, 2000, NT, 9x, etc.), Macintosh, Unix, Linux, etc.

The server **310** is suitably any Server for accommodating selective query support, selective data access, data archiving, and the like, as will be appreciated to one of ordinary skill in the art, but is preferably a DID. It should be noted that in a case where the server **310** is not a DID, there is preferably a DID also connected to data transport network **300**. The DID **310** is preferably in data communication with a data transport system **300** through a network interface **314**. Thus, a data path between one or more DIDs, such as that illustrated by DID **310**, is in shared data communication with at least one computer, such as that illustrated by Client **302**.

The DID **310** is suitably any networked DID as will be appreciated to one of ordinary skill in the art. The DID **310** preferably has an internal device controller **312** suitably acting as a fully functional server with the necessary hardware and software that ensure proper operation of the DID **310** as will be appreciated by those skilled in the art. In addition, the DID **310** preferably comprises internal storage **316**, which is preferably a hard disk and random access memory ("RAM") and is also suitably optical storage, removable memory, flash memory, or any other rewritable storage as will be appreciated by those skilled in the art. Preferably stored on storage **316** are the relational queue **318** and at least one software module or software component **320**.

As will be appreciated by those skilled in the art, the relational queue is preferably a relational database as illustrated in **Figure 1**, but is also suitably embodied as a markup language based document, such as XML, or as flat files, or any other searchable files. The software component ("SC") **320** is suitably computer-readable code written in any language and stored on a computer readable medium such as storage **316**. The SC **320** is preferably compiled, pre-written code that defines interfaces and is callable to provide the functionality that the SC **320** encapsulates. SCs are typically packaged in "industry standard" ways such that they are callable from multiple languages, or from multiple environments. The computer-readable code, in the case of SCs is suitably a unit of independent deployment that has no persistent state. As such, it provides seamless integration with existing development tools, such Forte for Java or Microsoft Visual Studio. SCs are suitably used out-of-the-box, or extended and specialized by developers as desired. It should be noted that the SCs of the present invention are suitably designed for any language binding, such as Common Object Request Broker Architecture ("CORBA"), .NET, COM, DCOM, C++, ActiveX, etc., as will be appreciated by those skilled in the art. In the presently preferred embodiment, the SC **320** is a C++ SC that is callable from multiple languages, or from multiple environments, or operating systems.

In the presently preferred embodiment of the invention, job requests are sent preferably from the client **302** to the DID **310** across the data transport network **300**. For example, a user of client **302** sends a print job request through the use of a DID driver, such as a printer driver, across data transport network **300** via network interface **308**. In addition, the print job request is preferably sent by a software program **304**, such as a word processing program or automation program, or by any other computer or user connected to the data transport network **300**.

The print job request is preferably received by a SC **320** that is preferably stored on storage **316** of the DID **310**. At least one SC **320** suitably performs the functionality described in **Figure 2**. The SCs **320** preferably provide at least the following system functionality:

Create, delete, pause and resume jobs;

Move jobs between queues;

Promote and demote jobs in a queue;

Assign priorities to jobs;

Process jobs based on their priority;

Search for all jobs with specific criteria;

Create, delete, clone, pause and resume queues;

Assign priorities to queues;

Process queues based on their priority; and

Create configurable execution path (job ticket) for different job types.

In addition, at least one SC **320** preferably also provides functionality for selecting predefined parent and child queues of varying types. Preferably such functionality is limited to system administrators. In particular, SCs **320** preferably provide system administrators with at least the following functionality:

Create custom queues, wherein each queue has a separate queue data store;

Assign priorities to queues;

Associate parent queues with child queues. In the presently preferred embodiment, the child queues inherit all the characteristics of their parents unless an administrator chooses to overwrite an inherited behavior with a new behavior;

Move jobs between queues;

Promote and demote jobs in queues;

Delete jobs from queues;

Pause and resume queues; and

Clone predefined or custom queues.

Similarly, SCs **320** preferably provide users (clients **302**) with at least the following functionality:

View the available queues to which a job can be submitted;

Create job ticket for a job describing how his job should be processed. In the presently preferred embodiment, the system provides a default job ticket if a client does not provide one.

Monitor the progress of a job;

Delete jobs that the client created from a queue; and

Pause and resume jobs in a queue that the client created.

Although the preferred embodiment has been described in detail, it should be understood that various changes, substitutions, and alterations can be made therein without departing from the spirit and scope of the invention as defined by the appended claims. It will be appreciated that various changes in the details, materials and arrangements of parts, which have been herein described and illustrated in order to explain the nature of the invention, may be made by those skilled in the area within the principle and scope of the invention as will be expressed in the appended claims.

I claim:

1. A method for job managing comprising:

creating a job in a client system,

sending a job creation instruction message to a job manager to notify that a new job has entered a processing queue;

sending at least one processing message from the job manager to at least one respective component;

sending a first job state change message from the respective component to the job manager to notify the job manager when the component begins to process a given job;

sending a second job state change message to the job manager from the component to notify the job manager when the component is done processing the given job.

2. The method of claim 1 wherein receipt of one of a job creation message and a job state change message by the job manager will cause the job manager to process all the jobs in the queue.

3. The method of claim 1 wherein if processing of the job is interrupted, an "interrupt" message is sent from the component to the job manager, who then reports the interruption to one of a user and a system administrator.

4. The method of claim 1 wherein, upon receipt of a message with a configurable interval of time (T), the job manager will process any jobs in the queue if the current time (CT) minus T is greater than the last time (LT) when the job manager received a "job create" or a "state change" message such that ((CT – T) > LT).

5. The method of claim 1 wherein the job is a printing job and the respective component for completing the job is a network printer.

6. The method of claim 1 wherein a respective component is assigned to process jobs in the queue for each respective client.

7. The method of claim 1 wherein the messages are sent and received using any type of conventional messaging system.

8. A job manager comprising:

an implementation for creating a job in a client system,

an implementation for sending a job creation instruction message to a job manager to notify that a new job has entered a processing queue;

an implementation for sending at least one processing message from the job manager to at least one respective component;

an implementation for sending a first job state change message from the respective component to the job manager to notify the job manager when the component begins to process a given job;

an implementation for sending a second job state change message to the job manager from the component to notify the job manager when the component is done processing the given job.

9. The job manager of claim 8 further comprising an implementation wherein receipt of one of a job creation message and a job state change message by the job manager will cause the job manager to process all the jobs in the queue.

10. The job manager of claim 8 further comprising an implementation wherein if processing of the job is interrupted, an "interrupt" message is sent from the component to the job manager, who then reports the interruption to one of a user and a system administrator.

11. The job manager of claim 8 further comprising an implementation wherein, upon receipt of a message with a configurable interval of time (T), the job manager will process any jobs in the queue if the current time (CT) minus T is greater than the last time (LT) when the job manager received a "job create" or a "state change" message such that ((CT – T) > LT).

12. The job manager of claim 8 wherein the job is a printing job and the respective component for completing the job is a network printer.

13. The job manager of claim 8 wherein a respective component is assigned to process jobs in the queue for each respective client.

14. The job manager of claim 8 wherein any type of conventional messaging system is employed for sending and receiving messages. method for storing received job requests comprising the steps of:

creating at least one parent queue data store for storing a plurality of parent queue records, each record comprising a parent queue record identifier data store that is configured for storing a parent queue record identifier that is unique at least within the parent queue data store;

creating at least one child queue data store for storing a plurality of child queue records, each record comprising:

a child queue record identifier data store that is configured for storing a child queue record identifier that is unique at least within the child queue data store, and

a parent queue record identifier data store; and

creating a job request data store for storing a plurality of job request records, each record comprising a job request record identifier data store that is configured for storing a job request record identifier that is unique at least within the job request data store; and

creating for each received job request a record in the job request data store and a related record in at least one of the parent queue data store and the child queue data store.

15. A method for storing received job requests comprising the steps of:

creating at least one parent queue data store for storing a plurality of parent queue records, each record comprising a parent queue record identifier data store that is configured for storing a parent queue record identifier that is unique at least within the parent queue data store;

creating at least one child queue data store for storing a plurality of child queue records, each record comprising:

a child queue record identifier data store that is configured for storing a child queue record identifier that is unique at least within the child queue data store, and

a parent queue record identifier data store; and

creating a job request data store for storing a plurality of job request records, each record comprising a job request record identifier data store that is configured for storing a job request record identifier that is unique at least within the job request data store; and

creating for each received job request a record in the job request data store and a related record in at least one of the parent queue data store and the child queue data store.

16. The method of claim 15 wherein the parent queue data store and child queue data store are the same data store and wherein each record of the parent queue data store.

17. The method of claim 15 wherein each job request record further comprises a job type data store.

18. The method of claim 17 wherein the step of creating a record in the job request data store further comprises the step of storing in the job type data store data representing job types selected from the group consisting of: print job, private print job, fax job, scan job, and combinations thereof.

19. The method of claim 15 wherein each job request record further comprises a job name data store.

20. The method of claim 15 wherein each job request record further comprises a job destination data store.

21. The method of claim 15 wherein each job request record further comprises information relative to the manner in which the job is to be serviced.

22. The method of claim 15 wherein each job request record further comprises a job priority data store.

23. The method of claim 22 wherein the step of creating a record in the job request data store further comprises the step of storing in the job priority data store data selected from the group consisting of: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, and 10.

24. The method of claim 15 wherein each job request record further comprises a job lifespan data store.

25. The method of claim 15 wherein each job request record further comprises a job status data store.

26. The method of claim 25 wherein the step of creating a record in the job request data store further comprises the step of storing in the job status data store data selected from the group consisting of: spooling, printing, scanning, faxing, private printing, ripping, creating jobs, and combinations thereof.

27. The method of claim 26 wherein each job request record further comprises a creation time data store and a completion time data store.

28. The method of claim 27 wherein the creation time data store and the completion time data store are configured for storing both date and time data.

29. The method of claim 28 wherein each parent request record further comprises a status data store.

30. A system for storing received job requests comprising:

computer readable code on a computer readable medium for creating at least one parent queue data store for storing a plurality of parent queue records, each record comprising a parent queue record identifier data store that is configured for storing a parent queue record identifier that is unique at least within the parent queue data store;

computer readable code on a computer readable medium for creating at least one child queue data store for storing a plurality of child queue records, each record comprising:

a child queue record identifier data store that is configured for storing a child queue record identifier that is unique at least within the child queue data store, and

a parent queue record identifier data store; and

computer readable code on a computer readable medium for creating a job request data store for storing a plurality of job request records, each record comprising a job request record identifier data store that is configured for storing a job request record identifier that is unique at least within the job request data store; and

computer readable code on a computer readable medium for creating for each received job request a record in the job request data store and a related record in at least one of the parent queue data store and the child queue data store.

31. The system of claim 30 wherein the parent queue data store and the child queue data store are the same data store.

32. The system of claim 30 wherein the job request data store, the parent queue data store, and the child queue data store form a relational database.

33. The system of claim 30 wherein each job request record further comprises a job type data store.

34. The system of claim 30 wherein each job request record further comprises a job name data store.

35. The system of claim 30 wherein each job request record further comprises a job destination data store.

36. The system of claim 30 wherein each job request record further comprises information relative to the manner in which the job is to be serviced.

37. The system of claim 30 wherein each job request record further comprises a job priority data store.

38. The system of claim 30 wherein each job request record further comprises a job lifespan data store.

39. The system of claim 30 wherein each job request record further comprises a job status data store.

40. The system of claim 30 wherein each job request record further comprises a creation time data store and a completion time data store.

41. The system of claim 30 wherein the creation time data store and the completion time data store are configured for storing both date and time data.

42. A system for storing job requests in a client-server network environment comprising:

a data transport network;

at least one client communicatively coupled to the data transport network;

at least one digital imaging device communicatively coupled to the data transport network; and

a server communicatively coupled to the data transport network comprising:

a job request table for storing information relative to job requests received from the at least one client,

a parent queue table for organizing and servicing job requests, and

at least one child queue table for organizing and servicing job requests;

wherein the job request table, parent queue table, and the at least one child queue table form a relational database.

43. The system of claim 42 wherein the digital imaging device and the server are the same device.

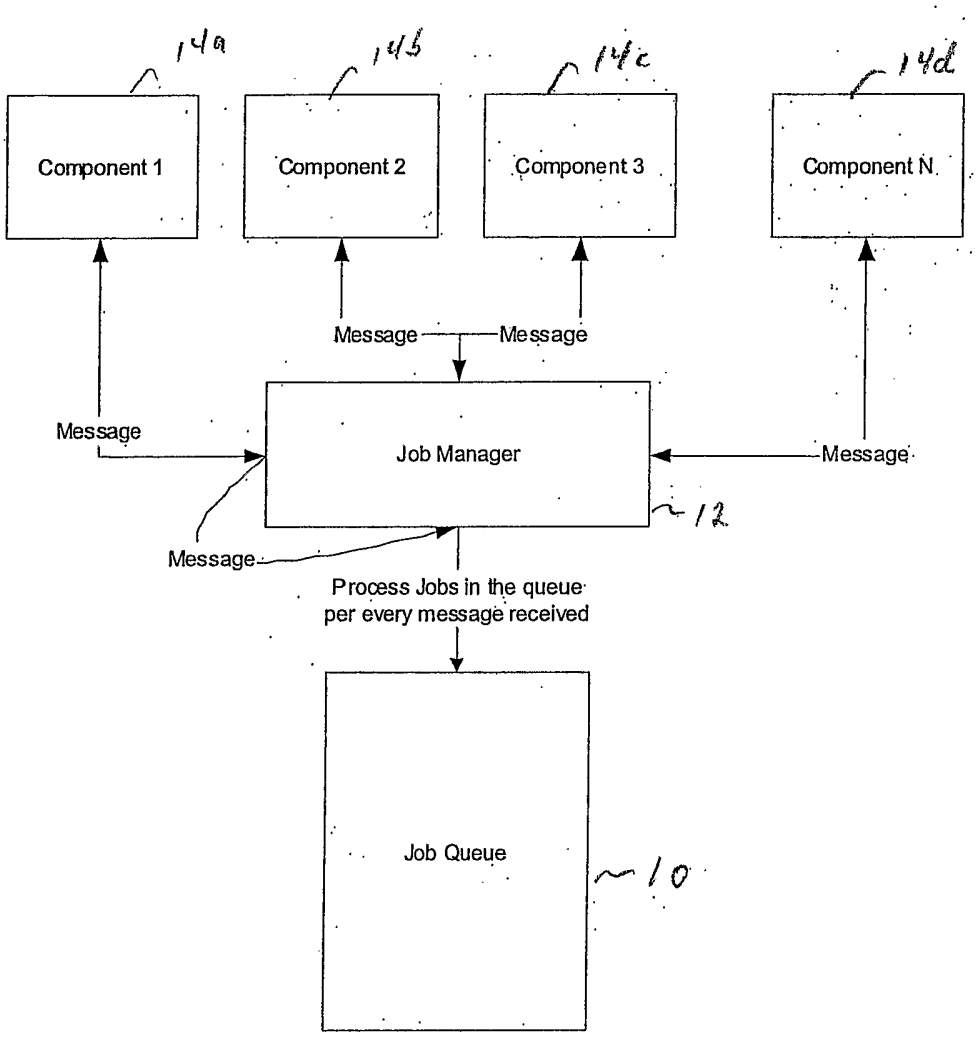

14a
14b
14c
14d
Component 1
Component 2
Component 3
Component N
Message
Message
Message
Job Manager
Message
12
Message
Process Jobs in the queue per every message received
Job Queue
10

FIG. 1a

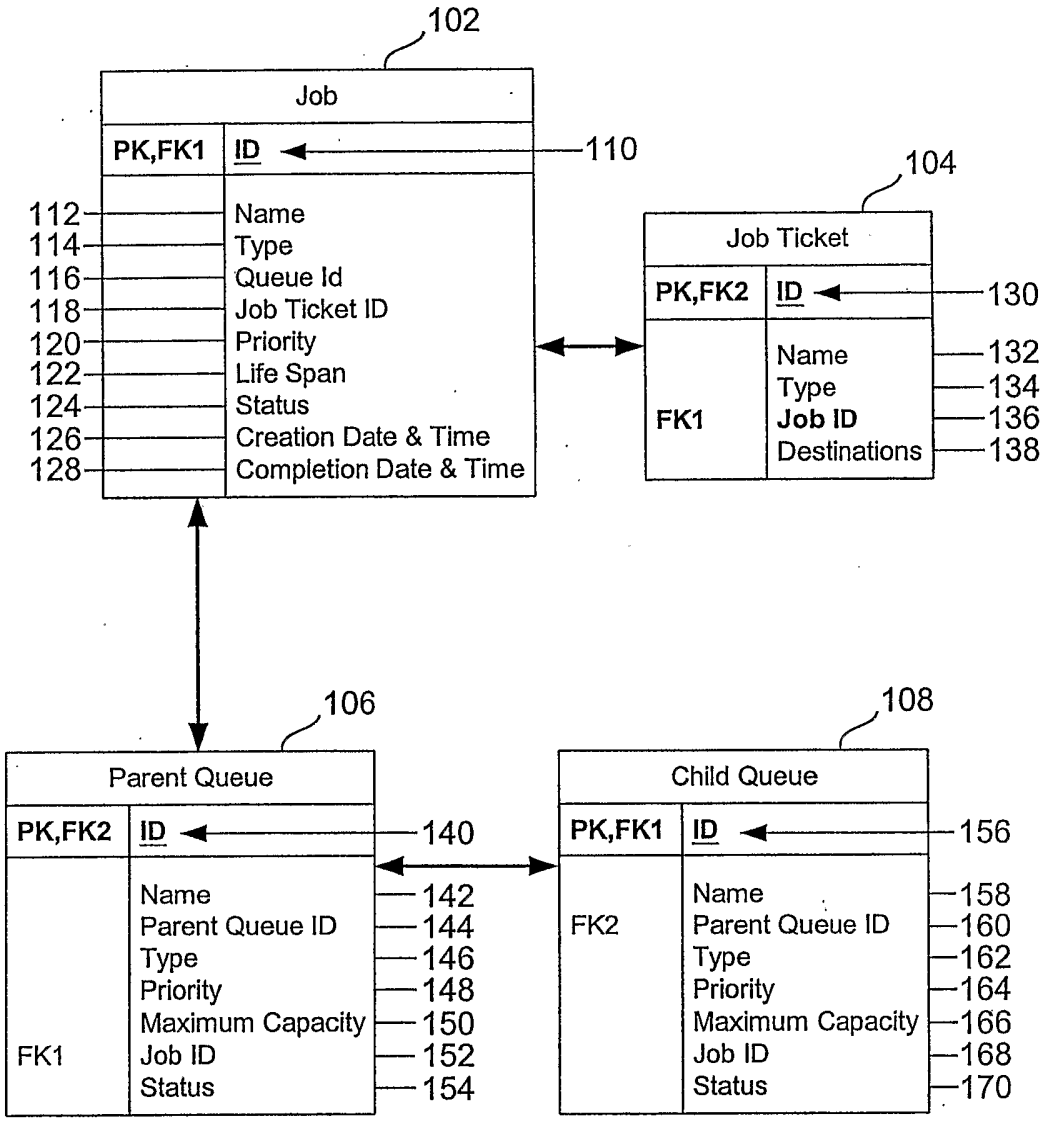

102
Job
PK,FK1
ID
110
112 Name
114 Type
116 Queue Id
118 Job Ticket ID
120 Priority
122 Life Span
124 Status
126 Creation Date & Time
128 Completion Date & Time
104
Job Ticket
PK,FK2
ID
130
FK1
Name
Type
Job ID
Destinations
132
134
136
138
106
Parent Queue
PK,FK2
ID
140
FK1
Name
Parent Queue ID
Type
Priority
Maximum Capacity
Job ID
Status
142
144
146
148
150
152
154
108
Child Queue
PK,FK1
ID
156
FK2
Name
Parent Queue ID
Type
Priority
Maximum Capacity
Job ID
Status
158
160
162
164
166
168
170

Figure 1b

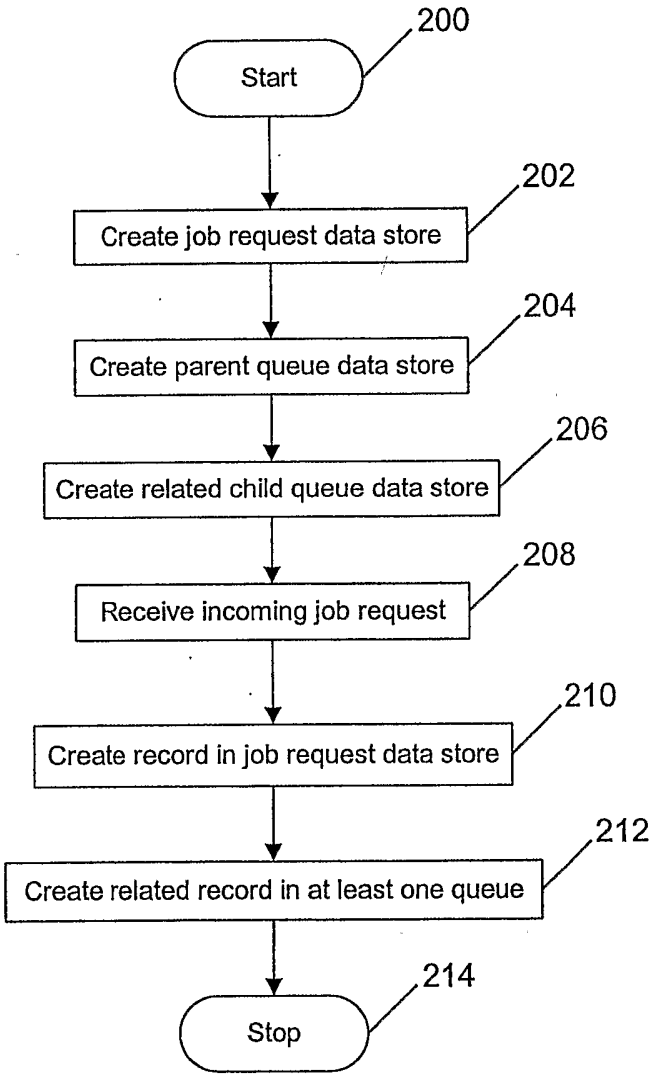

200
Start
202
Create job request data store
204
Create parent queue data store
206
Create related child queue data store
208
Receive incoming job request
210
Create record in job request data store
212
Create related record in at least one queue
214
Stop

Figure 2

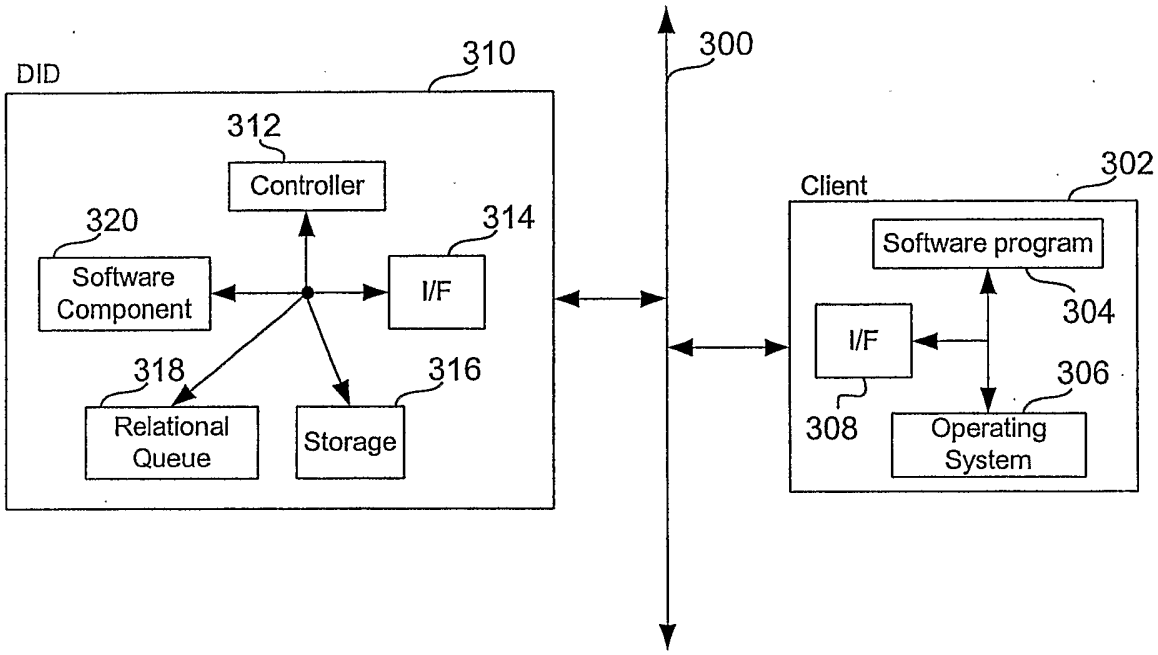

DID
310
312
Controller
320
Software Component
314
I/F
318
Relational Queue
316
Storage
300
Client
302
Software program
304
I/F
306
308
Operating System

Figure 3

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US03/16990

A. CLASSIFICATION OF SUBJECT MATTER

IPC(7) : G06F 15/00
US CL : 358/1.15

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)
U.S. : 358/1.15, 1.13, 1.14

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched
NONE

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
EAST

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X,E	US 6,587,735 B1 (YAGUCHI) 01 July 2003 (1.07.2003), the entire document.	1-43
X,P	US 6,570,670 B1 (SALGADO et al) 27 May 2003 (27.05.2003), the entire document	1-43
X,P	US 6,474,881 B1 (WANDA) 05 November 2002 (05.11.2002), the entire document.	1-43

☐ 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 published on or after the international filing date

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

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

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

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

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

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

"&" document member of the same patent family

Date of the actual completion of the international search
11 August 2003 (11.08.2003)

Date of mailing of the international search report
16 SEP 2003

Name and mailing address of the ISA/US
Mail Stop PCT, Attn: ISA/US
Commissioner for Patents
P.O. Box 1450
Alexandria, Virginia 22313-1450
Facsimile No. (703)305-3230

Authorized officer
Mark E. Wallerson
Telephone No. (703) 305-4700

Form PCT/ISA/210 (second sheet) (July 1998)