(12) UK Patent Application (19) GB (11) 2503932 (13) A

(43) Date of A Publication 15.01.2014

(21) Application No: 1212509.2

(22) Date of Filing: 13.07.2012

(71) Applicant(s):
International Business Machines Corporation (Incorporated in USA - New York)
New Orchard Road, Armonk, N.Y. 10504,
United States of America

(72) Inventor(s):
Ivan Derek Hargreaves
Ian James Mitchell
Julian Charles Horn
Fraser Bohm

(74) Agent and/or Address for Service:
IBM United Kingdom Limited
Intellectual Property Law, Hursley Park,
WINCHESTER, Hampshire, SO21 2JN,
United Kingdom

(51) INT CL:
G06F 9/48 (2006.01) ***G06F 9/46*** (2006.01)
G06F 9/455 (2006.01)

(56) Documents Cited:
EP 1122644 A1 **US 6138169 A**
US 20090210874 A1

(58) Field of Search:
INT CL **G06F**
Other: **Online: WPI, EPODOC, TXTE, INSPEC, XPESP, XPESP2, XPIEE, XPIPCOM, XPI3E, XPLNCS, XPRD, XPSPRNG.**

(54) Title of the Invention: **Hybrid computer thread creation and management**
Abstract Title: **Creating a hybrid processing thread to be executed on multiple application servers**

(57) A computer system 100 is operable to support an application, and includes at least two application server environments 102, 104. The system comprises: a component 106 for intercepting a request relating to the application issued to the first environment before it the requested work is executed; a component 108 for creating a processing thread 110 from the request which is adapted for execution in the first environment; a dispatcher component 112, which attaches a context 114 to the thread to modify the thread into a hybrid thread 116 which is also suitable for execution in the second environment; and a component 118 for returning the hybrid thread to the first environment. This allows the request to access services or components in the second environment without requiring additional layers of processor overhead or modification of either environment. The environments may be incompatible in terms of their programming models, where one may be a procedural environment and the other an object-oriented environment.

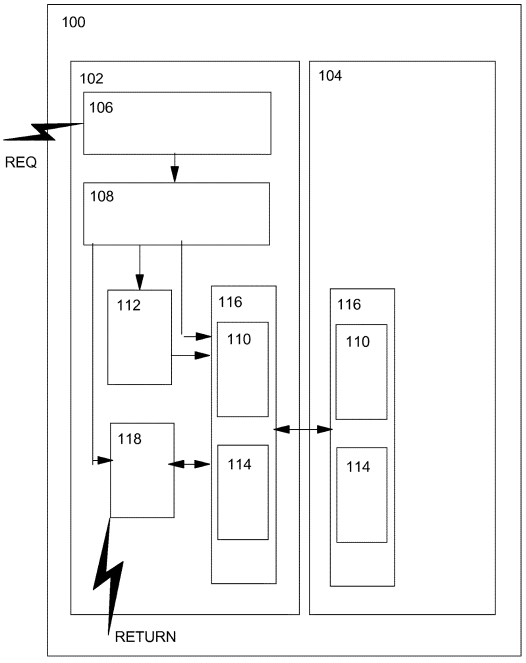

Figure 1

GB 2503932 A

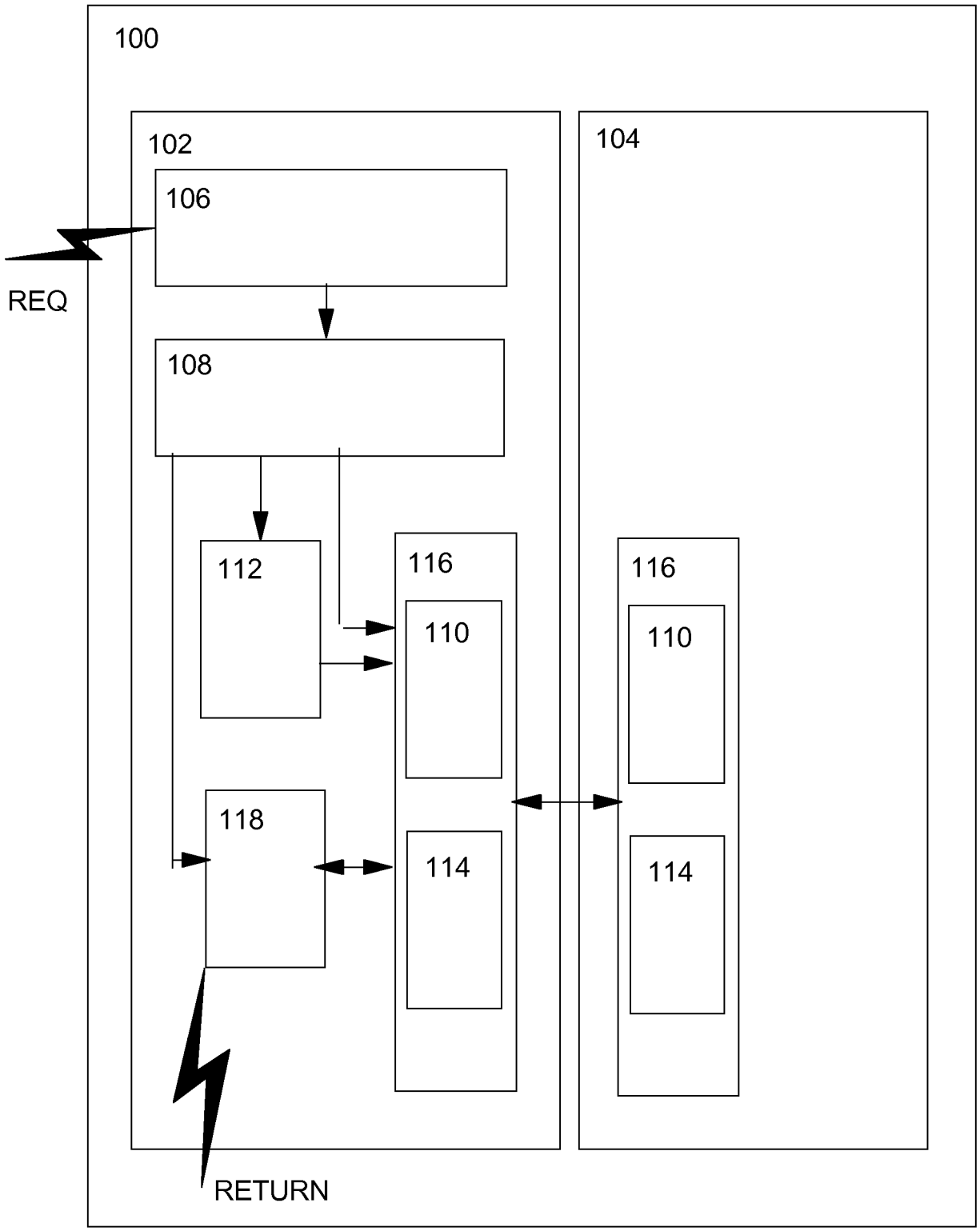

Figure 1

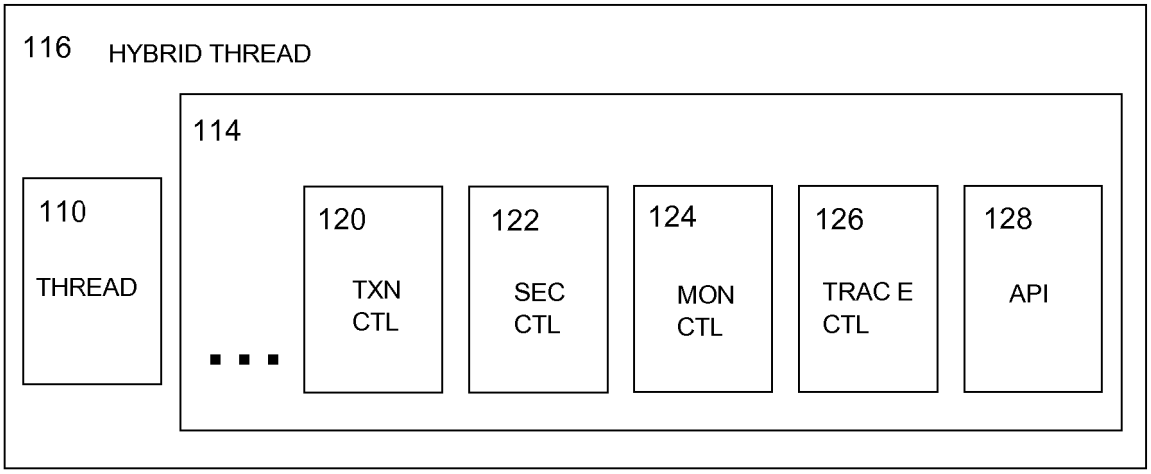
116
HYBRID THREAD
114
110
THREAD
120
TXN
CTL
122
SEC
CTL
124
MON
CTL
126
TRAC E
CTL
128
API

Figure 2

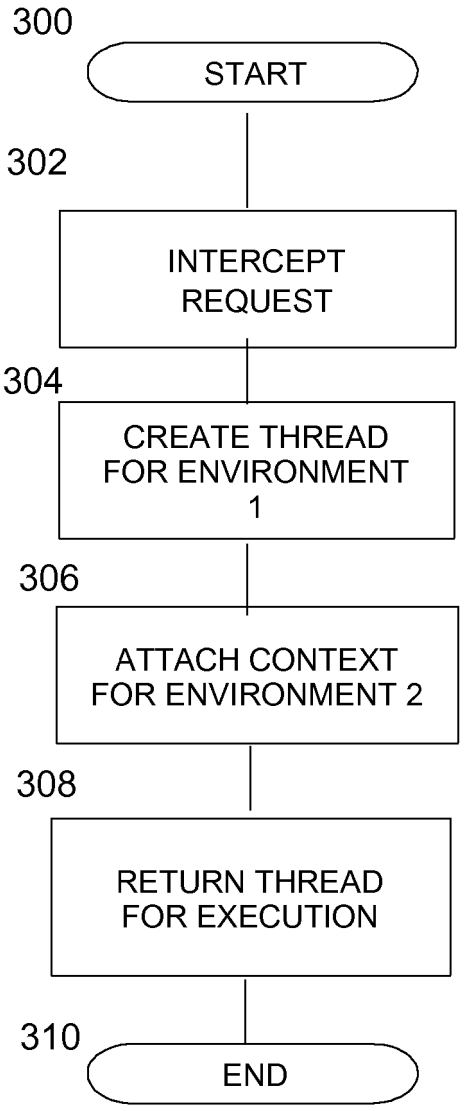
300
START
302
INTERCEPT REQUEST
304
CREATE THREAD FOR ENVIRONMENT 1
306
ATTACH CONTEXT FOR ENVIRONMENT 2
308
RETURN THREAD FOR EXECUTION
310
END

Figure 3

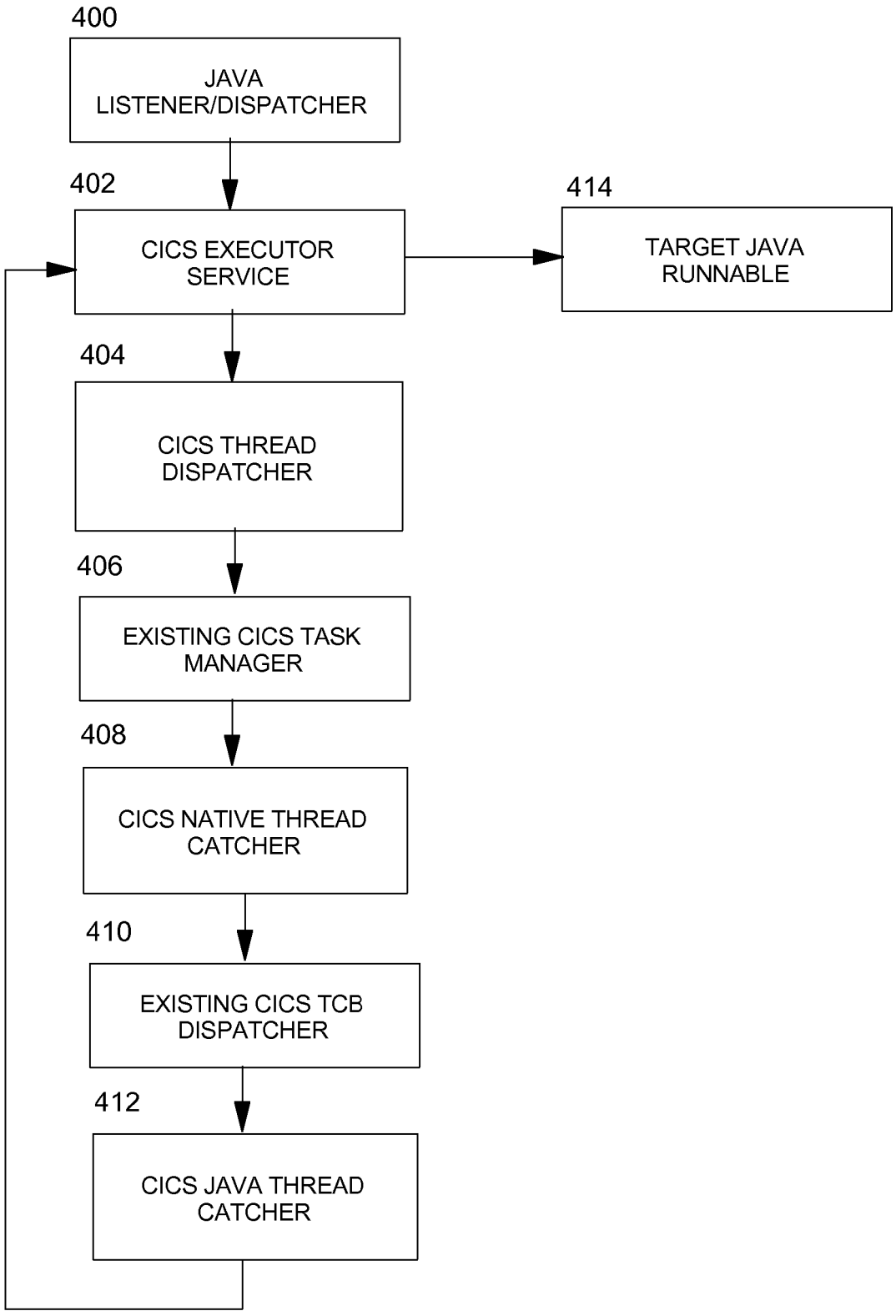
400
JAVA LISTENER/DISPATCHER
402
CICS EXECUTOR SERVICE
414
TARGET JAVA RUNNABLE
404
CICS THREAD DISPATCHER
406
EXISTING CICS TASK MANAGER
408
CICS NATIVE THREAD CATCHER
410
EXISTING CICS TCB DISPATCHER
412
CICS JAVA THREAD CATCHER

Figure 4

HYBRID COMPUTER THREAD CREATION AND MANAGEMENT

BACKGROUND OF THE INVENTION

The present invention relates to a system and method for managing a hybrid thread in heterogeneous application server environments.

As is well known in the computer art, application server environments are supporting environments that mediate between the base computer operating system environment and user applications. Application server environments also provide a number of supporting services to user applications; for example, control of communications between clients and databases, performance monitoring, diagnostic tracing and transaction control services, among others. Many also provide specialised application programming interfaces tailored to enable users to exploit advanced services in a simple, standardised manner.

Examples of application server environments are Oracle Corporation's WebLogic, SAP NetWeaver Application Server, IBM Corporation's WebSphere Application Server, and IBM Corporation's CICS Transaction Server, among others. Application server environments may operate independently, side-by-side within an operating system environment, or in a nested fashion, with one environment embedded within another. (IBM, WebSphere and CICS are trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide. Other product or service names may be trademarks or registered trademarks of others).

As is also well known in the art, application server environments take several forms. Some application server environments follow traditional computing paradigms (for example, using traditional procedural programming approaches), while others follow more recent programming paradigms, such as object-oriented programming and portable programming models like the Java model. (Java and all Java-related trademarks and logos are trademarks or registered trademarks of Oracle Corporation in the US and other countries.)

Where user applications have been created to use the services of, and the programming model provided by, a particular one of these application server environments, serious problems arise in any attempt to move to a different application server environment, or to enable coexistence

of applications between different application server environments. These problems are exacerbated by the need to support "software as a service" and cloud computing models, in which applications need to be made mobile between server platforms and across multiple, potentially heterogeneous environments.

It is known that various attempts to address this problem have been proposed. One example is the provision of wrappers, connectors or adapters, which provide translations of the language and other constructs used by a user application that are specific to a first application server environment into the equivalent language and other constructs that would be used by the user application in a second application server environment. These wrappers, connectors and adapters disadvantageously add layers of processing overhead between the applications and their environments and thus adversely affect the performance, reliability and serviceability of the applications and the application server environments.

It would thus be desirable to alleviate these adverse effects and to provide a means by which applications could take advantage of the services provided by different application server environments without being bound by the limitations of those environments, and without the need to reprogram applications to make them portable between the environments.

It would therefore be desirable to improve upon the known art.

SUMMARY OF THE INVENTION

According to a first aspect of the present invention, there is provided a computer system (100) operable to support an application; at least a first application server environment (102) and a second application server environment (104); and comprising: a request interceptor component (106) for intercepting a work request relating to said application issued to said first application server environment (102) prior to execution of requested work; an executor component (108), responsive to said request interceptor component (106), for creating a thread (110) adapted for execution in said first application server environment (102); a thread dispatcher component (112) , responsive to said executor component (108), for attaching to said thread (110) a context (114) to non-disruptively modify said thread (110) into a hybrid thread (116) additionally suitable for execution in said second application server environment

(104); and a catcher component (118), responsive to said thread dispatcher component (112), for returning said hybrid thread (116) to said first application server environment (102).

Preferably, said hybrid thread (116) is operable to execute in an unmodified said first application server environment (102) and in an unmodified said second application server environment (104). Preferably, said hybrid thread (116) comprises first data visible to a server only in said first application server environment (102) and second data visible to a server only in said second application server environment (104). Preferably, said first application server environment (102) and said second application server environment (104) are incompatible in their program execution models. Preferably, only one of said first application server environment (102) or said second application server environment (104) is a procedural program environment. Preferably, only one of said first application server environment (102) or said second application server environment (104) is an object-oriented program environment. Preferably, said context comprises transactional control data (120) of one of said application server environments (102, 104). Preferably, said context comprises transactional control data (120), security control data (122), monitoring control data (124), diagnostic tracing control data (126), or data (128) enabling access to an application programming interface of one of said application server environments (102, 104).

According to a second aspect of the present invention, there is provided a method of operating a computer system (100) to support an application; at least a first application server environment (102) and a second application server environment (104); and comprising: intercepting, by a request interceptor component (106), a work request relating to said application issued to said first application server environment (102) prior to execution of requested work; responsive to said request interceptor component (106), creating a thread (110) adapted for execution in said first application server environment (102) by an executor component (108); responsive to said executor component (108), attaching to said thread (110), by a thread dispatcher component (112), a context (114) to non-disruptively modify said thread (110) into a hybrid thread (116) additionally suitable for execution in said second application server environment (104); and responsive to said thread dispatcher component (112), returning said hybrid thread (116) to said first application server environment (102) by a catcher component (118).

Preferably, said hybrid thread (116) is operable to execute in an unmodified said first application server environment (102) and in an unmodified said second application server environment (104). Preferably, said hybrid thread (116) comprises first data visible to a server only in said first application server environment (102) and second data visible to a server only in said second application server environment (104). Preferably, said first application server environment (102) and said second application server environment (104) are incompatible in their program execution models. Preferably, only one of said first application server environment (102) or said second application server environment (104) is a procedural program environment. Preferably, only one of said first application server environment (102) or said second application server environment (104) is an object-oriented program environment. Preferably, said context comprises transactional control data (120) of one of said application server environments (102, 104). Preferably, said context comprises transactional control data (120), security control data (122), monitoring control data (124), diagnostic tracing control data (126), or data (128) enabling access to an application programming interface of one of said application server environments (102, 104).

In a third aspect, there is provided a computer program comprising computer program code to, when loaded into a computer system and executed thereon, cause said computer system to perform the steps of the method according to the second aspect.

BRIEF DESCRIPTION OF THE DRAWINGS

Embodiments of the present invention will now be described, by way of example only, with reference to the accompanying drawings, in which:

Figure 1 illustrates the components of a system according to a preferred embodiment of the present invention;

Figure 2 represents a detailed view of a hybrid thread according to a preferred embodiment of the present invention;

Figure 3 represents a flow diagram illustrating the steps of a method of operation according to a preferred embodiment of the present invention; and

Figure 4 represents a flow of control in an exemplary CICS and Java embodiment of the present invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

In a typical commercial data processing system of the present day, there is often a mixture of processing environments, some of which, like IBM's high-performance and high-integrity CICS transaction processing products, follow a conventional, though highly advanced and reliable, procedural programming and operational paradigm, and some of which, like IBM's WebSphere Application Server, offer an inbuilt opportunity to exploit object-oriented and Java processing environments. As will be clear to one of ordinary skill in the art, CICS and WebSphere are merely two examples of application server environments; many other such application server environments are supplied by other vendors.

CICS has for some time provided facilities for permitting a level of interoperability between the normal CICS environment and programs implementing a Java model. However, these facilities have certain limitations – for example, they do not apply to processing threads initiated from inside the Java Virtual Machine (JVM) as a result of a Thread.Start instruction. Such threads are not able to participate in a CICS task and cannot make use of the CICS API and services. This is particularly unfortunate when users have a need to port server side components to run in the JVM under CICS as it is a very common pattern for such components (Jetty, Axis 2, Liberty etc), to attach worker threads from inside the JVM to satisfy external requests. When such components are hosted in CICS it is almost certain that such worker threads will require access to CICS services or API, and this cannot be achieved using the facilities presently provided.

The approaches to improving the situation implemented in various application server environments thus far involve significant reprogramming, both of the application server environments and of the applications themselves, and this is clearly disadvantageous.

The first is to completely remove the listening/dispatching component from one application server environment and to re-implement it using the infrastructure provided by a second application server environment. As noted above, this approach has the disadvantage that it requires a significant rewrite of the ported server side components.

The second approach is to alter the listening/dispatching logic of a first application server environment to issue an additional call to the second application server environment to start another transaction. This transaction is started outside the first application server environment and must then be synchronised with the calling component inside the first application server environment. The approach again requires a significant re-write of the dispatching/listening components to be made workable. It also requires the first application server environment code to know the characteristics of the required task environment of the second application server environment.

Turning to Figure 1, there is illustrated a computer system (100) operable to support an application; at least a first application server environment (102) and a second application server environment (104). In a most preferred embodiment, the first application server environment (102) is a CICS application server environment, and the second application server environment (104) is a Java-enabled environment, such as the WebSphere Application Server.

Computer system (100) comprises a request interceptor component (106) for intercepting a work request relating to the application issued to the first application server environment (102) prior to execution of requested work. In the present embodiment, the work request could be, for example, a Java Thread.Start request. Computer system (100) further comprises an executor component (108), responsive to the request interceptor component (106), for creating a thread (110) adapted for execution in said first application server environment (102).

There is also provided a thread dispatcher component (112), responsive to the executor component (108), for attaching to the thread (110) a context (114) to non-disruptively modify the thread (110) into a hybrid thread (116) additionally suitable for execution in the second application server environment (104). A catcher component (118) is responsive to the thread dispatcher component (112), for returning said hybrid thread (116) to the first application server environment (102).

The computer system (100) as described enables the hybrid thread (116) to execute in an unmodified first application server environment (102) and in an unmodified second application server environment (104). In the computer system (100) as described, the hybrid

thread (116) comprises first data visible to a server only in the first application server environment (102) and second data visible to a server only in the second application server environment (104).

Advantageously, the computer system (100) as described is operable in systems in which the first application server environment (102) and the second application server environment (104) are incompatible in their program execution models. For example, one of the application server environments (102, 104) could be a procedural program environment, while the other application server environment (102, 104) could be an object-oriented program environment.

The context (114) attached to the thread in this way to make a hybrid thread, as is shown in Figure 2, could comprise transactional control data (120), security control data (122), monitoring control data (124), diagnostic tracing control data (126), or data (128) enabling access to an application programming interface of one of the application server environments (102, 104).

Turning to Figure 3, there is shown a method of operating a computer system (100) to support an application; at least a first application server environment (102) and a second application server environment (104); and comprising the step, following START step (300), of intercepting (302), by a request interceptor component (106), a work request relating to the application issued to the first application server environment (102) prior to execution of requested work. The method continues at step (304), responsive to the request interceptor component (106), creating (304) a thread (110) adapted for execution in said first application server environment (102) by an executor component (108). Then, responsive to the executor component (108), the process continues by attaching (306) to the thread (110), by a thread dispatcher component (112), a context (114) to non-disruptively modify the thread (110) into a hybrid thread (116) additionally suitable for execution in the second application server environment (104). Responsive to the thread dispatcher component (112), the catcher component (118) returns, at step (308) the hybrid thread (116) to the first application server environment (102), and the process completes, for this instantiation, at END step (310).

Thus, in the preferred CICS and Java embodiment, there is provided a combined (partially in Java, partially native services) thread dispatching service which can be called from any Java

thread (CICS attached or not) that will provide a Java thread and related CICS task environment such that the thread is also a CICS task and can make full use of CICS API and services.

In a most preferred embodiment, this thread dispatching service is used to implement a Java ThreadExecutor – this allows the advantages of the preferred embodiment of the invention to be achieved with no code changes to the dispatching/listening components and minimal changes to the configuration of the ported component.

Turning to Figure 4, there is shown the flow through a set of components of a preferred CICS and Java embodiment. The Java listener/dispatcher (400) receives the work request and calls the CICS executor service (402). CICS executor service (402) creates the base Java thread and then invokes CICS thread dispatcher (404), which uses the services of the existing CICS task manager (406), the CICS native thread catcher (408) and the existing CICS task control block dispatcher (410) to create and attach the CICS context to the thread, thereby creating a hybrid thread. The thread is then passed back by the CICS Java thread catcher (412) to the CICS executor service (402), which emits the target Java runnable component (414).

As will be appreciated by one skilled in the art, aspects of the present invention may be embodied as a system, method or computer program product. Accordingly, aspects of the present invention may take the form of an entirely hardware embodiment, an entirely software embodiment (including firmware, resident software, micro-code, etc.) or an embodiment combining software and hardware aspects that may all generally be referred to herein as a "circuit," "module" or "system." Furthermore, aspects of the present invention may take the form of a computer program product embodied in one or more computer readable medium(s) having computer readable program code embodied thereon.

Any combination of one or more computer readable medium(s) may be utilized. The computer readable medium may be a computer readable signal medium or a computer readable storage medium. A computer readable storage medium may be, for example, but not limited to, an electronic, magnetic, optical, electromagnetic, infrared, or semiconductor system, apparatus, or device, or any suitable combination of the foregoing. More specific examples (a non-exhaustive list) of the computer readable storage medium would include the following: an electrical connection having one or more wires, a portable computer diskette, a

hard disk, a random access memory (RAM), a read-only memory (ROM), an erasable programmable read-only memory (EPROM or Flash memory), an optical fiber, a portable compact disc read-only memory (CD-ROM), an optical storage device, a magnetic storage device, or any suitable combination of the foregoing. In the context of this document, a computer readable storage medium may be any tangible medium that can contain, or store a program for use by or in connection with an instruction execution system, apparatus, or device.

A computer readable signal medium may include a propagated data signal with computer readable program code embodied therein, for example, in baseband or as part of a carrier wave. Such a propagated signal may take any of a variety of forms, including, but not limited to, electro-magnetic, optical, or any suitable combination thereof. A computer readable signal medium may be any computer readable medium that is not a computer readable storage medium and that can communicate, propagate, or transport a program for use by or in connection with an instruction execution system, apparatus, or device.

Program code embodied on a computer readable medium may be transmitted using any appropriate medium, including but not limited to wireless, wireline, optical fiber cable, RF, etc., or any suitable combination of the foregoing.

Computer program code for carrying out operations for aspects of the present invention may be written in any combination of one or more programming languages, including an object oriented programming language such as Java, Smalltalk, C++ or the like and conventional procedural programming languages, such as the "C" programming language or similar programming languages. The program code may execute entirely on the user's computer, partly on the user's computer, as a stand-alone software package, partly on the user's computer and partly on a remote computer or entirely on the remote computer or server. In the latter scenario, the remote computer may be connected to the user's computer through any type of network, including a local area network (LAN) or a wide area network (WAN), or the connection may be made to an external computer (for example, through the Internet using an Internet Service Provider).

It will be equally clear to one of skill in the art that all or part of a logic arrangement according to the preferred embodiments of the present invention may suitably be embodied in

a logic apparatus comprising logic elements to perform the steps of the method, and that such logic elements may comprise components such as logic gates in, for example a programmable logic array or application-specific integrated circuit. Such a logic arrangement may further be embodied in enabling elements for temporarily or permanently establishing logic structures in such an array or circuit using, for example, a virtual hardware descriptor language, which may be stored and transmitted using fixed or transmittable carrier media.

Aspects of the present invention are described below with reference to flowchart illustrations and/or block diagrams of methods, apparatus (systems) and computer program products according to embodiments of the invention. It will be understood that each block of the flowchart illustrations and/or block diagrams, and combinations of blocks in the flowchart illustrations and/or block diagrams, can be implemented by computer program instructions. These computer program instructions may be provided to a processor of a general purpose computer, special purpose computer, or other programmable data processing apparatus to produce a machine, such that the instructions, which execute via the processor of the computer or other programmable data processing apparatus, create means for implementing the functions/acts specified in the flowchart and/or block diagram block or blocks.

These computer program instructions may also be stored in a computer readable medium that can direct a computer, other programmable data processing apparatus, or other devices to function in a particular manner, such that the instructions stored in the computer readable medium produce an article of manufacture including instructions which implement the function/act specified in the flowchart and/or block diagram block or blocks.

The computer program instructions may also be loaded onto a computer, other programmable data processing apparatus, or other devices to cause a series of operational steps to be performed on the computer, other programmable apparatus or other devices to produce a computer implemented process such that the instructions which execute on the computer or other programmable apparatus provide processes for implementing the functions/acts specified in the flowchart and/or block diagram block or blocks.

The flowchart and block diagrams in the Figures illustrate the architecture, functionality, and operation of possible implementations of systems, methods and computer program products according to various embodiments of the present invention. In this regard, each block in the

flowchart or block diagrams may represent a module, segment, or portion of code, which comprises one or more executable instructions for implementing the specified logical function(s). It should also be noted that, in some alternative implementations, the functions noted in the block may occur out of the order noted in the figures. For example, two blocks shown in succession may, in fact, be executed substantially concurrently, or the blocks may sometimes be executed in the reverse order, depending upon the functionality involved. It will also be noted that each block of the block diagrams and/or flowchart illustration, and combinations of blocks in the block diagrams and/or flowchart illustration, can be implemented by special purpose hardware-based systems that perform the specified functions or acts, or combinations of special purpose hardware and computer instructions.

CLAIMS

1. A computer system (100) operable to support an application; at least a first application server environment (102) and a second application server environment (104); and comprising:

a request interceptor component (106) for intercepting a work request relating to said application issued to said first application server environment (102) prior to execution of requested work;

an executor component (108), responsive to said request interceptor component (106), for creating a thread (110) adapted for execution in said first application server environment (102);

a thread dispatcher component (112) , responsive to said executor component (108), for attaching to said thread (110) a context (114) to non-disruptively modify said thread (110) into a hybrid thread (116) additionally suitable for execution in said second application server environment (104); and

a catcher component (118), responsive to said thread dispatcher component (112), for returning said hybrid thread (116) to said first application server environment (102).

2. The computer system (100) as claimed in claim 1, wherein said hybrid thread (116) is operable to execute in an unmodified said first application server environment (102) and in an unmodified said second application server environment (104).

3. The computer system (100) as claimed in claim 2, wherein said hybrid thread (116) comprises first data visible to a server only in said first application server environment (102) and second data visible to a server only in said second application server environment (104).

4. The computer system (100) as claimed in any preceding claim, wherein said first application server environment (102) and said second application server environment (104) are incompatible in their program execution models.

5. The computer system (100) as claimed in claim 4, wherein only one of said first application server environment (102) or said second application server environment (104) is a procedural program environment.

6. The computer system (100) as claimed in claim 4, wherein only one of said first application server environment (102) or said second application server environment (104) is an object-oriented program environment.

7. The computer system (100) as claimed in any preceding claim, wherein said context (114) comprises transactional control data (120) of one of said application server environments (102, 104).

8. The computer system (100) as claimed in any preceding claim, wherein said context (114) comprises security control data (122) of one of said application server environments (102, 104).

9. The computer system (100) as claimed in any preceding claim, wherein said context (114) comprises monitoring control data (124) of one of said application server environments (102, 104).

10. The computer system (100) as claimed in any preceding claim, wherein said context (114) comprises diagnostic tracing control data (126) of one of said application server environments (102, 104).

11. The computer system (100) as claimed in any preceding claim, wherein said context (114) comprises data (128) enabling access to an application programming interface of one of said application server environments (102, 104).

12. A method of operating a computer system (100) to support an application; at least a first application server environment (102) and a second application server environment (104); and comprising:

intercepting, by a request interceptor component (106), a work request relating to said application issued to said first application server environment (102) prior to execution of requested work;

responsive to said request interceptor component (106), creating a thread (110) adapted for execution in said first application server environment (102) by an executor component (108);

responsive to said executor component (108), attaching to said thread (110), by a thread dispatcher component (112), a context (114) to non-disruptively modify said thread (110) into a hybrid thread (116) additionally suitable for execution in said second application server environment (104); and

responsive to said thread dispatcher component (112), returning said hybrid thread (116) to said first application server environment (102) by a catcher component (118).

13. The method as claimed in claim 12, wherein said hybrid thread (116) is operable to execute in an unmodified said first application server environment (102) and in an unmodified said second application server environment (104).

14. The method as claimed in claim 13, wherein said hybrid thread (116) comprises first data visible to a server only in said first application server environment (102) and second data visible to a server only in said second application server environment (104).

15. The method as claimed in any of claims 12 to 14, wherein said first application server environment (102) and said second application server environment (104) are incompatible in their program execution models.

16. The method as claimed in claim 15, wherein only one of said first application server environment (102) or said second application server environment (104) is a procedural program environment.

17. The method as claimed in claim 15, wherein only one of said first application server environment (102) or said second application server environment (104) is an object-oriented program environment.

18. The method as claimed in any of claims 12 to 17, wherein said context (114) comprises transactional control data (120) of one of said application server environments (102, 104).

19. The method as claimed in any of claims 12 to 18, wherein said context (114) comprises security control data (122) of one of said application server environments (102, 104).

20. The method as claimed in any of claims 12 to 19, wherein said context (114) comprises monitoring control data (124) of one of said application server environments (102, 104).

21. The method as claimed in any of claims 12 to 20, wherein said context (114) comprises diagnostic tracing control data (126) of one of said application server environments (102, 104).

22. The method as claimed in any of claims 12 to 21, wherein said context (114) comprises data (128) enabling access to an application programming interface of one of said application server environments (102, 104).

23. A computer program comprising computer program code to, when loaded into a computer system and executed thereon, cause said computer system to perform the steps of the method as claimed in any of claims 12 to 22.

Application No: GB1212509.2 **Examiner:** Mr Michael Warren

Claims searched: 1-23 **Date of search:** 5 November 2012

Patents Act 1977: Search Report under Section 17

Documents considered to be relevant:

Category	Relevant to claims	Identity of document and passage or figure of particular relevance
A	-	EP 1122644 A1 (SUN MICROSYSTEMS)
A	-	US 2009/0210874 A1 (HARRIS et al)
A	-	US 6138169 A (FREUND et al)

Categories:

X	Document indicating lack of novelty or inventive step	A	Document indicating technological background and/or state of the art.
Y	Document indicating lack of inventive step if combined with one or more other documents of same category.	P	Document published on or after the declared priority date but before the filing date of this invention.
&	Member of the same patent family	E	Patent document published on or after, but with priority date earlier than, the filing date of this application.

Field of Search:

Search of GB, EP, WO & US patent documents classified in the following areas of the UKC^{X} :

Worldwide search of patent documents classified in the following areas of the IPC

G06F

The following online and other databases have been used in the preparation of this search report

WPI, EPODOC, TXTE, INSPEC, XPESP, XPESP2, XPIEE, XPIPCOM, XPI3E, XPLNCS, XPRD, XPSPRNG

International Classification:

Subclass	Subgroup	Valid From
G06F	0009/48	01/01/2006
G06F	0009/46	01/01/2006
G06F	0009/455	01/01/2006