US 20120079486A1

(19) United States

(12) Patent Application Publication

(10) Pub. No.: US 2012/0079486 A1

(43) Pub. Date: Mar. 29, 2012

Brandt et al.

(54) **INTEGRATION OF DISSIMILAR JOB TYPES INTO AN EARLIEST DEADLINE FIRST (EDF) SCHEDULE**

(75) Inventors: **Scott A. Brandt**, San Jose, CA (US); **Richard A. Golding**, San Francisco, CA (US); **Theodore M. Wong**, Mountain View, CA (US)

(73) Assignee: **INTERNATIONAL BUSINESS MACHINES CORPORATION**, Armonk, NY (US)

(21) Appl. No.: **12/889,263**

(22) Filed: **Sep. 23, 2010**

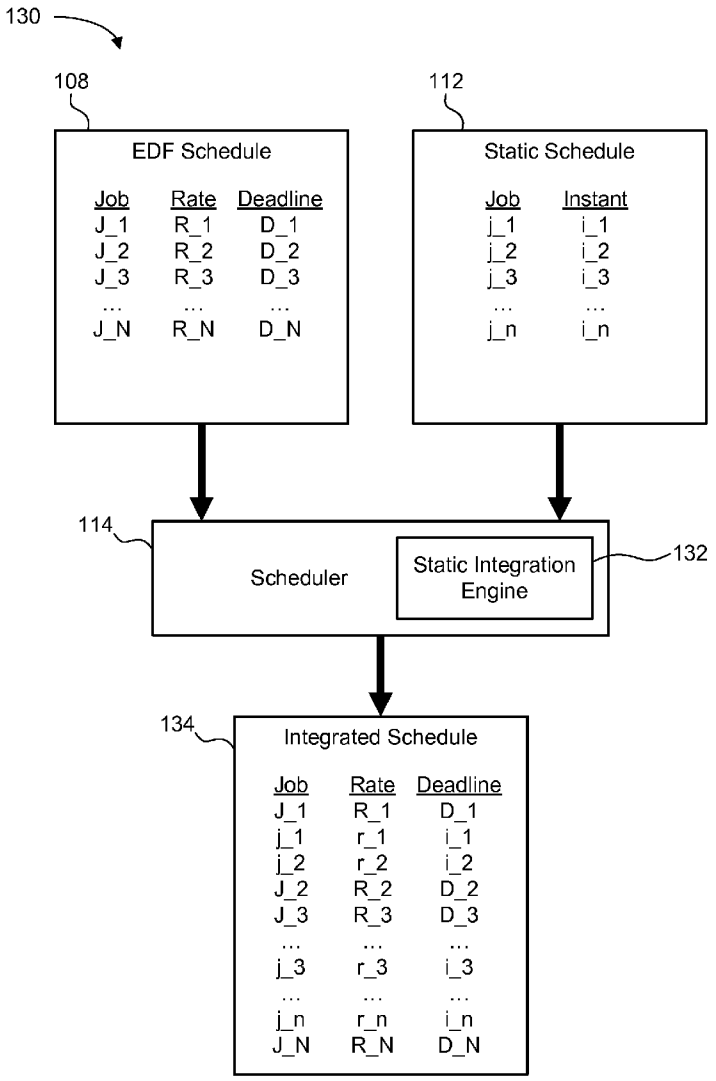
Publication Classification

(51) **Int. Cl.**
G06F 9/46 (2006.01)

(52) **U.S. Cl.** **718/102**

(57) **ABSTRACT**

A system for inserting jobs into a scheduler of a processor includes the processor and the scheduler. The processor executes instructions related to a plurality of jobs. The scheduler implements an earliest deadline first (EDF) scheduling model. The scheduler also receives a plurality of jobs from an EDF schedule. The scheduler also receives a separate job from a source other than the EDF schedule. The separate job has a fixed scheduling requirement. The separate job also may be a short duration sporadic job. The scheduler also inserts the separate job into an execution plan of the processor in response to a determination that an available utilization capacity of the processor is sufficient to execute the separate job according to the fixed scheduling requirement associated with the separate job.



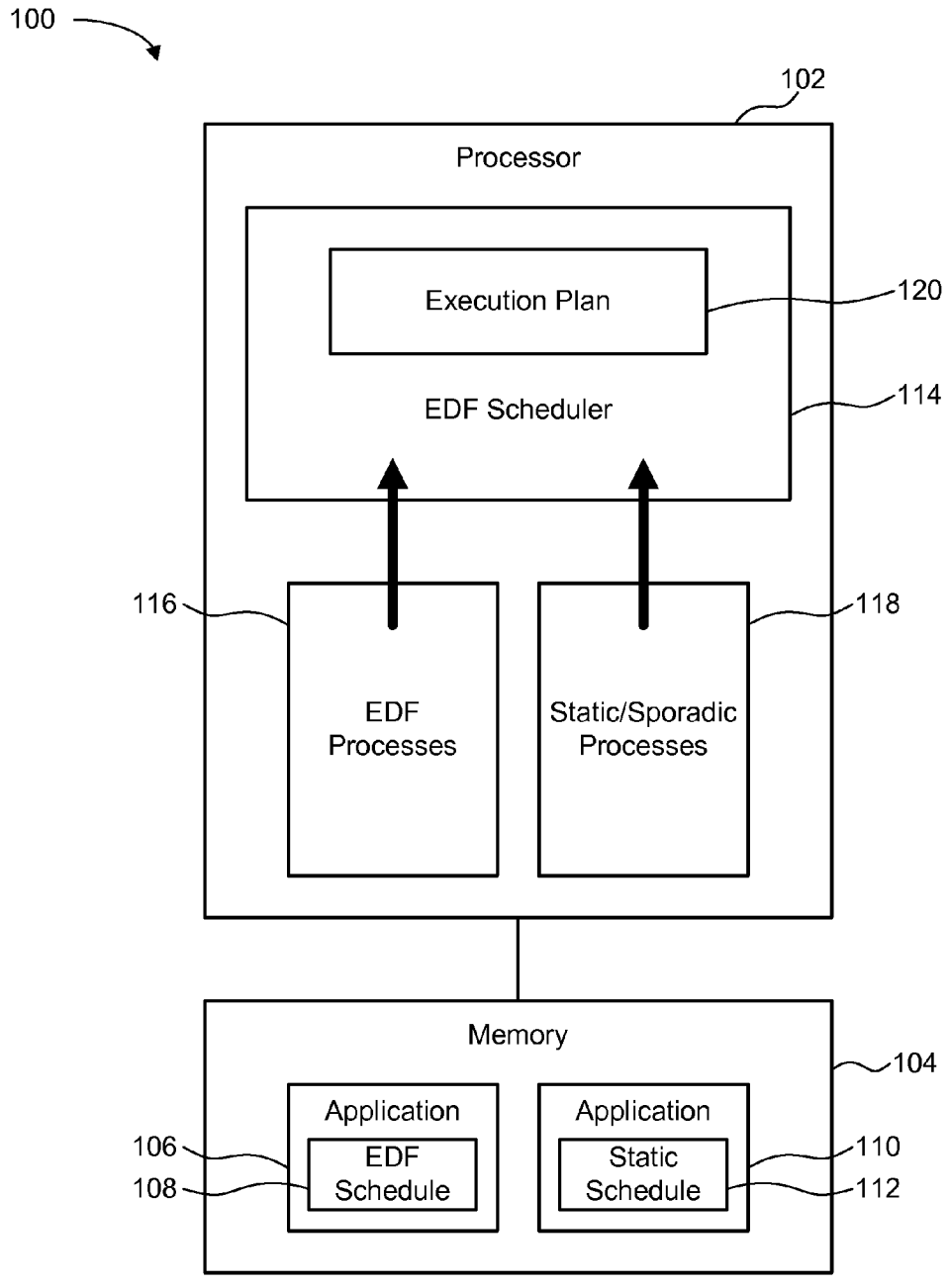

100
102
Processor
Execution Plan
120
EDF Scheduler
114
116
118
EDF Processes
Static/Sporadic Processes
Memory
104
Application
EDF Schedule
Application
Static Schedule
106
108
110
112

FIG. 1

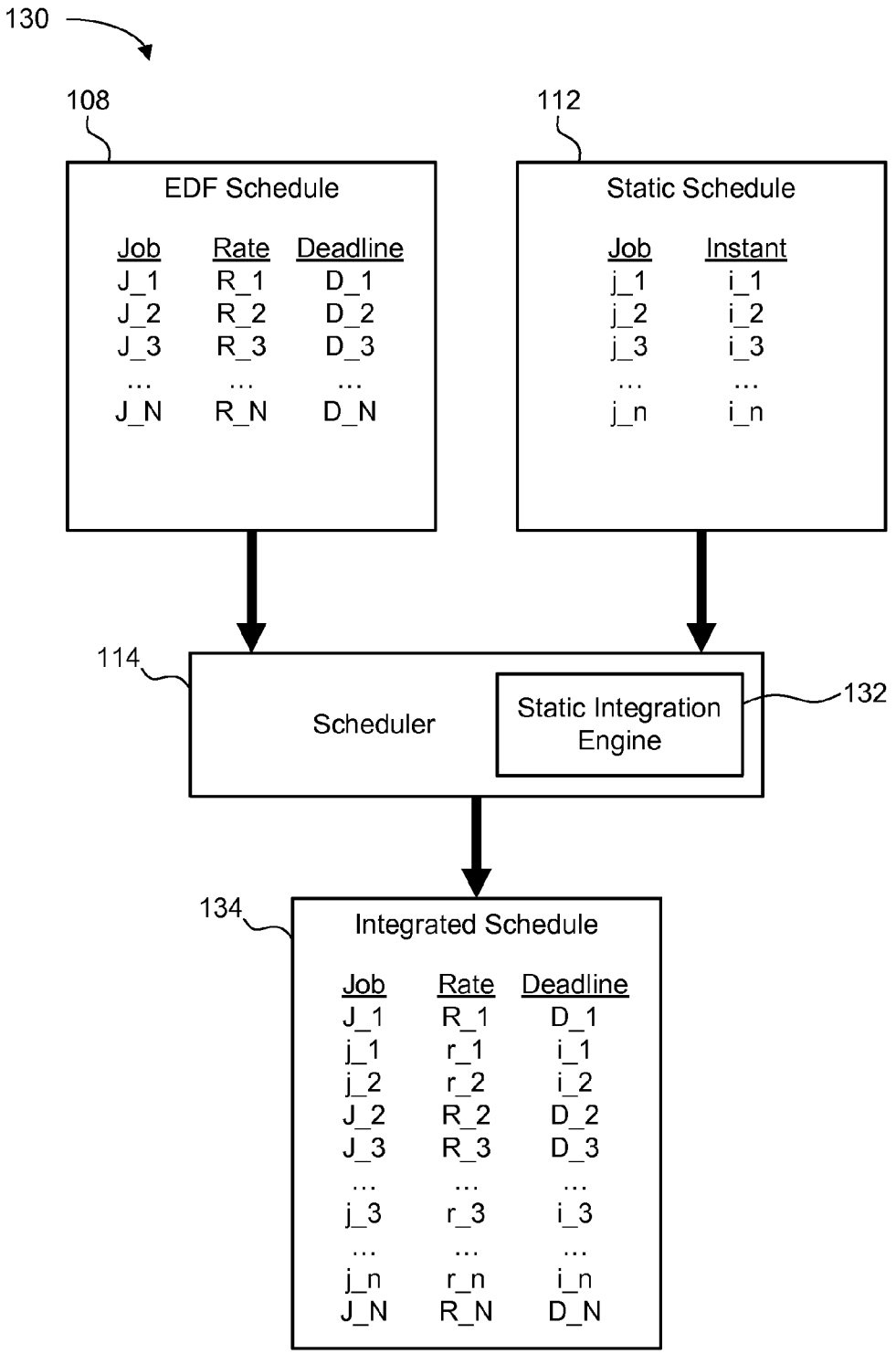

130
108
EDF Schedule
Job Rate Deadline
J_1 R_1 D_1
J_2 R_2 D_2
J_3 R_3 D_3
... ...
J_N R_N D_N
112
Static Schedule
Job Instant
j_1 i_1
j_2 i_2
j_3 i_3
... ...
j_n i_n
114
Scheduler
Static Integration Engine
132
134
Integrated Schedule
Job Rate Deadline
J_1 R_1 D_1
j_1 r_1 i_1
j_2 r_2 i_2
J_2 R_2 D_2
J_3 R_3 D_3
... ...
j_3 r_3 i_3
... ...
j_n r_n i_n
J_N R_N D_N

FIG. 2

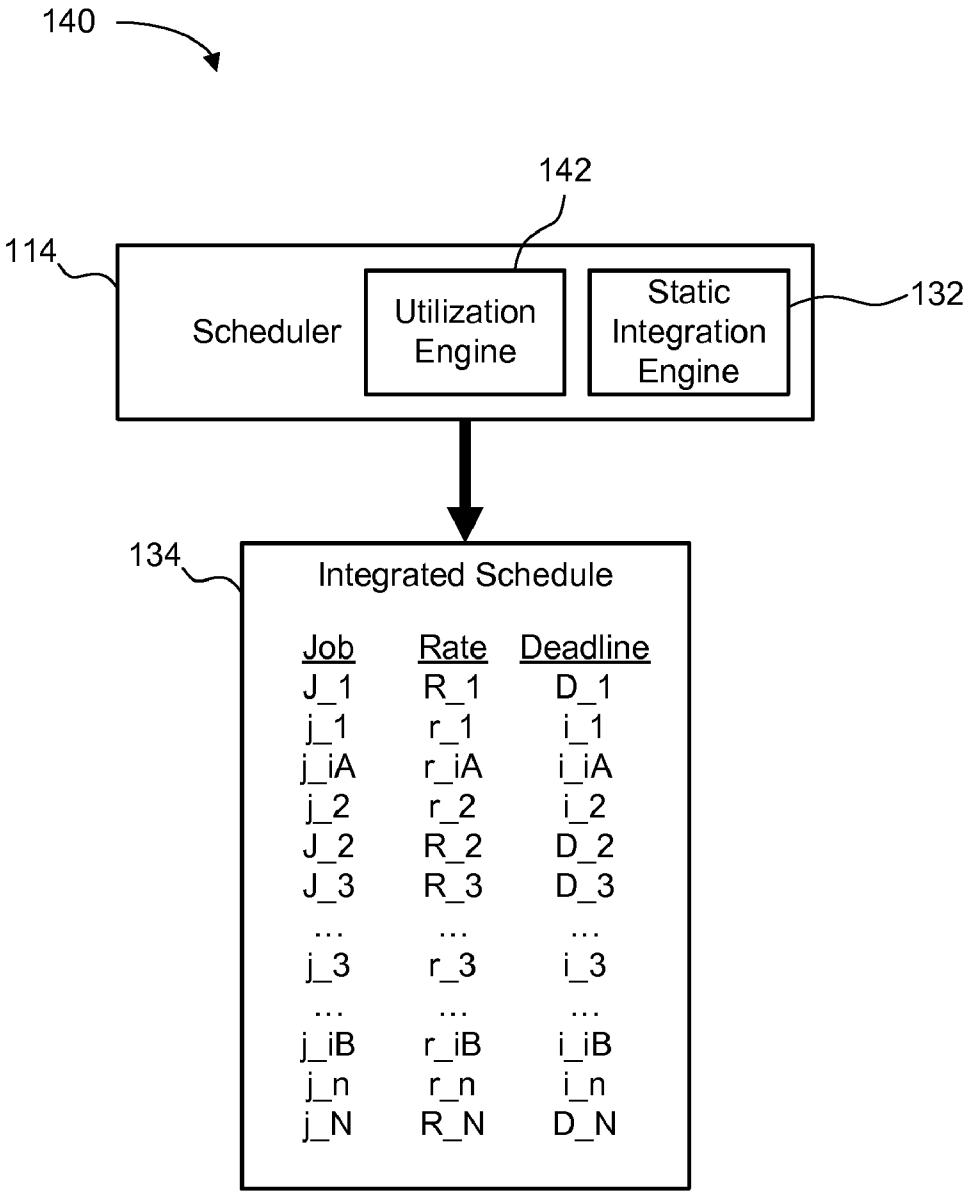

140
142
114
Scheduler
Utilization Engine
Static Integration Engine
132
134
Integrated Schedule
Job Rate Deadline
J_1 R_1 D_1
j_1 r_1 i_1
j_iA r_iA i_iA
j_2 r_2 i_2
J_2 R_2 D_2
J_3 R_3 D_3
... ...
j_3 r_3 i_3
... ...
j_iB r_iB i_iB
j_n r_n i_n
j_N R_N D_N

FIG. 3

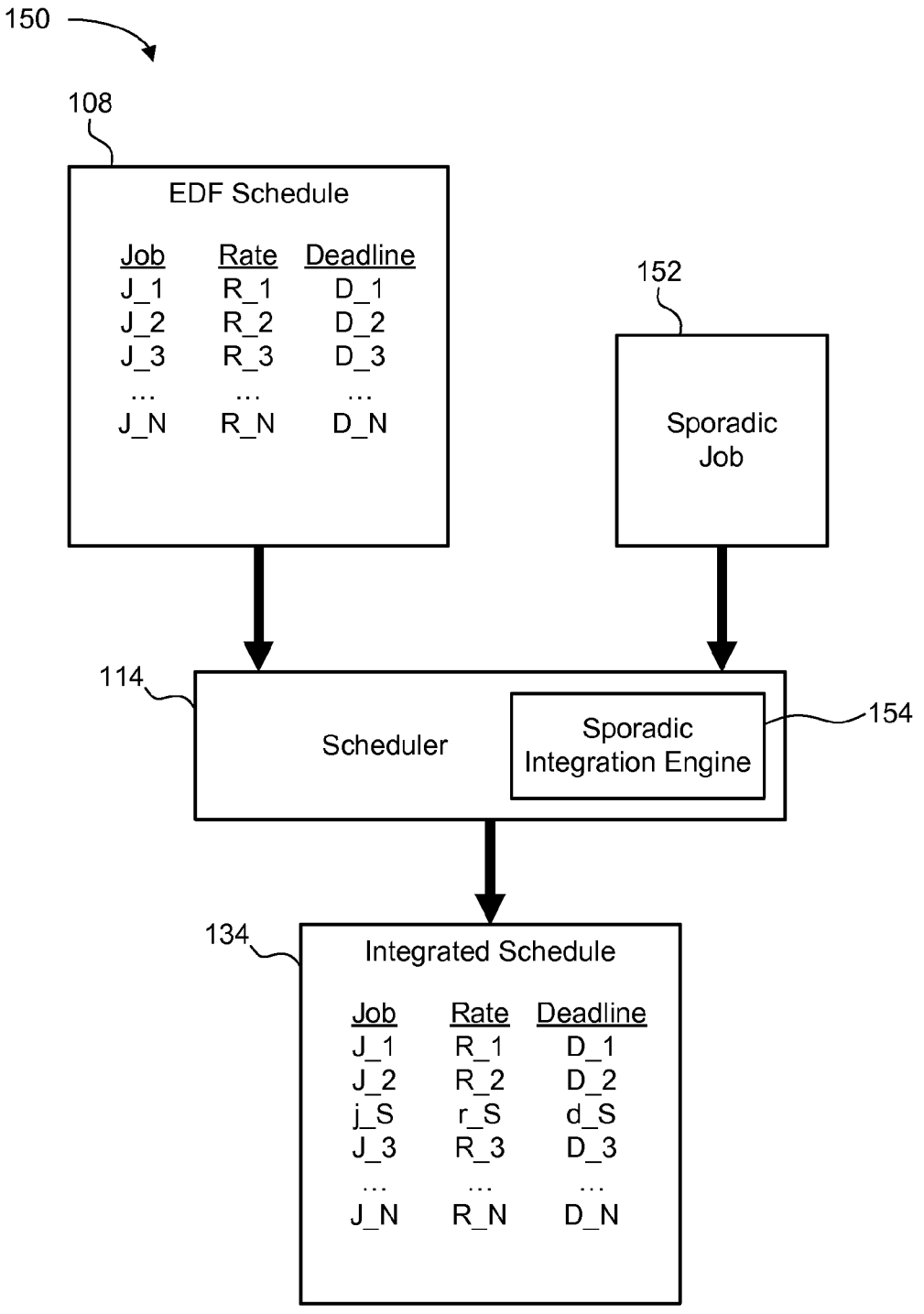

150
108
EDF Schedule
Job Rate Deadline
J_1 R_1 D_1
J_2 R_2 D_2
J_3 R_3 D_3
... ...
J_N R_N D_N
152
Sporadic Job
114
Scheduler
Sporadic Integration Engine
154
134
Integrated Schedule
Job Rate Deadline
J_1 R_1 D_1
J_2 R_2 D_2
j_S r_S d_S
J_3 R_3 D_3
... ...
J_N R_N D_N

FIG. 4

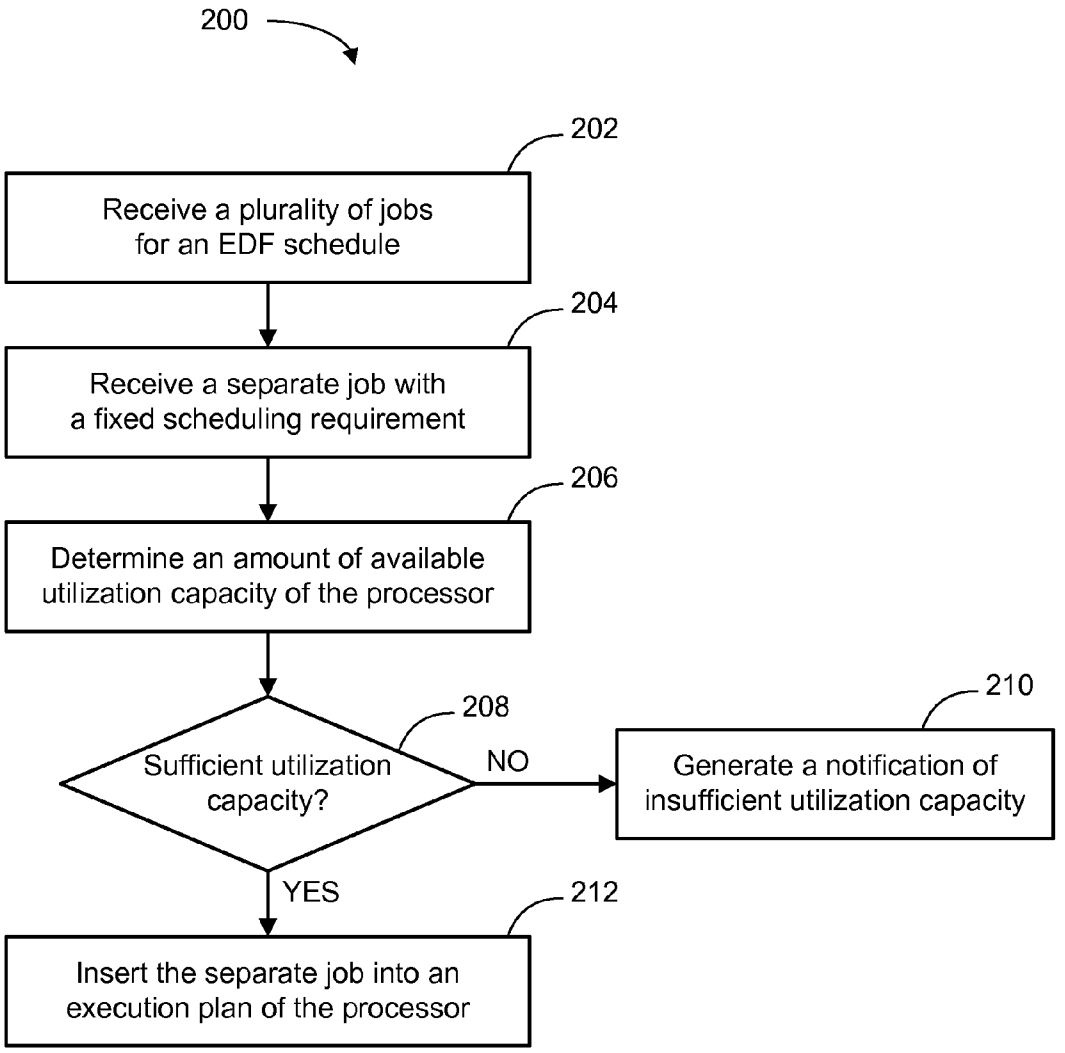

200
202
Receive a plurality of jobs for an EDF schedule
204
Receive a separate job with a fixed scheduling requirement
206
Determine an amount of available utilization capacity of the processor
208
Sufficient utilization capacity?
NO
210
Generate a notification of insufficient utilization capacity
YES
212
Insert the separate job into an execution plan of the processor

FIG. 5

INTEGRATION OF DISSIMILAR JOB TYPES INTO AN EARLIEST DEADLINE FIRST (EDF) SCHEDULE

STATEMENT OF FEDERALLY SPONSORED RESEARCH

[0001] This invention was made with Government support under Contract No. HR0011-08-C-0031 awarded by Defense Advanced Research Projects Agency (DARPA). The Government has certain rights in the invention.

BACKGROUND

[0002] Many systems, including embedded systems, have real-time timeliness constraints on their processing. Such systems often interact with the real world and/or users and have specific constraints on when they must respond to events, produce results, etc. Some examples of such systems include safety-critical systems such as anti-lock brakes, control systems such as flight control systems, multimedia systems such as video or audio players, and so forth. Systems intended to address such real-time situations must be carefully designed to ensure the timing requirements of the applications they are built to execute.

[0003] The CPU scheduler is a core operating system component responsible for managing the execution scheduling of different programs. The CPU scheduler is responsible for determining which program to run at any given time. Where programs have implicit or explicit timeliness constraints or deadlines, the CPU scheduler is primarily responsible for ensuring that the applications execute in time to satisfy the timeliness constraints or deadlines. Many different operating systems and CPU schedulers have been developed to address various sorts of timing requirements.

[0004] One type of scheduler that is frequently used in legacy systems (e.g., military and aerospace systems) is based on static schedules. Static schedules are very basic table-driven schedules that include an ordered list of tasks along with a set of times at which those tasks should occur. Upon completion of all of the tasks in the list, the list typically repeats with a fixed frequency, or period. Static schedules are common in legacy systems due in part to their extreme simplicity. However, static schedules are limited in their use because each static schedule is essentially hand-designed to ensure that all work can be completed in the time available. This is feasible only for small task sets.

[0005] In contrast to static schedules, other schedulers use dynamic schedules. One type of dynamic schedule is a dynamic priority-based schedule. Priority scheduling (e.g., rate monotonic (RM)) assigns a static priority to each concurrently executing task. Tasks are executed according to their priority, with the highest priority task that is currently executable always selected for execution. A limitation of static priority-based schedules is that in general only a relatively low fraction of the CPU utilization may be guaranteed. Another type of dynamic schedule is a dynamic deadline-based schedule. Deadline scheduling assigns a set of deadlines to each task. In one deadline-based scheduling algorithm known as earliest deadline first (EDF), the system always runs the task with the earliest deadline. An advantage of EDF is that 100% of the CPU utilization may be guaranteed. However, EDF is less frequently used due to concerns about its overhead. Recent research indicates that deadline based schemes are effective in supporting a broad range of types of processes ranging from best-effort to hard real-time.

[0006] An advantage of RM and EDF dynamic scheduling schemes is that they allow the concurrent execution of different applications with different timeliness requirements with less hand-integration than a static schedule. In this way, each application can be admitted to the system based on knowledge of the resource availability, without detailed knowledge of the other applications. A potential disadvantage of RM and EDF dynamic scheduling is that they are incapable of supporting legacy processes designed to operate in isolation with static schedulers. This prevents the migration of legacy applications and systems to modern multi-programmed processing platforms. Since legacy applications that use static scheduling are common in aerospace, automotive, and other industries, there are many legacy applications that currently cannot be integrated into modern multi-programmed processing platforms that use dynamic scheduling.

SUMMARY

[0007] Embodiments of a computer program product are described. The computer program product includes a computer readable storage medium to store a computer readable program that, when executed on a computer, causes the computer to perform operations for scheduling jobs on a processor. The scheduler implements an EDF scheduling model. In one embodiment, the operations include receiving a plurality of jobs for an earliest deadline first (EDF) schedule. The operations also include receiving one or more separate jobs with a fixed scheduling requirement. In some embodiments, the separate job additionally or alternatively may be a short duration sporadic job. The operations also include determining an amount of available utilization capacity of the processor. The operations also include inserting the separate job into an execution plan of the processor in response to a determination that the available utilization capacity of the processor is sufficient to execute the separate job according to the fixed scheduling requirement associated with the separate job. Other embodiments of the computer program product and corresponding operations are also described.

[0008] Embodiments of a system are also described. In one embodiment, the system includes a processor and a scheduler. The processor executes instructions related to a plurality of jobs. The scheduler implements an earliest deadline first (EDF) scheduling model. The scheduler also receives a plurality of jobs for an EDF schedule. The scheduler also receives a separate job with a fixed scheduling requirement. The scheduler also inserts the separate job into an execution plan of the processor in response to a determination that an available utilization capacity of the processor is sufficient to execute the separate job according to the fixed scheduling requirement associated with the separate job. Other embodiments of the system are also described.

[0009] Embodiments of a method are also described. In one embodiment, the method is implemented within a scheduler for a processor. The scheduler implements an earliest deadline first (EDF) scheduling model. An embodiment of the method includes receiving a plurality of jobs for an EDF schedule. The method also includes receiving a separate job with a fixed scheduling requirement with a specific execution time. The method also includes determining an amount of available utilization capacity of the processor. The method also includes inserting the separate job into an execution plan of the processor with the plurality of jobs from the EDF

schedule in response to a determination that the available utilization capacity of the processor is sufficient to execute the separate job according to the fixed scheduling requirement associated with the separate job. Other embodiments of the method are also described.

[0010] Other aspects and advantages of embodiments of the present invention will become apparent from the following detailed description, taken in conjunction with the accompanying drawings, illustrated by way of example of the principles of the invention.

BRIEF DESCRIPTION OF THE DRAWINGS

[0011] FIG. **1** depicts a schematic block diagram of one embodiment of a computer system for integrating dissimilar job types into an integrated schedule for dynamic scheduling.

[0012] FIG. **2** depicts a schematic diagram of one embodiment of a process for integrating static jobs into an EDF schedule.

[0013] FIG. **3** depicts a schematic diagram of one embodiment of a process for inserting idle jobs into the integrated schedule of FIG. **2**.

[0014] FIG. **4** depicts a schematic diagram of one embodiment of a process for integrating a sporadic job into an EDF schedule.

[0015] FIG. **5** depicts a flow chart diagram of one embodiment of a method for operating the EDF scheduler for the processor within the computer system of FIG. **1**.

[0016] Throughout the description, similar reference numbers may be used to identify similar elements.

DETAILED DESCRIPTION

[0017] In the following description, specific details of various embodiments are provided. However, some embodiments may be practiced with less than all of these specific details. In other instances, certain methods, procedures, components, structures, and/or functions are described in no more detail than to enable the various embodiments of the invention, for the sake of brevity and clarity.

[0018] It will be readily understood that the components of the embodiments as generally described herein and illustrated in the appended figures could be arranged and designed in a wide variety of different configurations. Thus, the following more detailed description of various embodiments, as represented in the figures, is not intended to limit the scope of the present disclosure, but is merely representative of various embodiments. While the various aspects of the embodiments are presented in drawings, the drawings are not necessarily drawn to scale unless specifically indicated.

[0019] The present invention may be embodied in other specific forms without departing from its spirit or essential characteristics. The described embodiments are to be considered in all respects only as illustrative and not restrictive. The scope of the invention is, therefore, indicated by the appended claims rather than by this detailed description. All changes which come within the meaning and range of equivalency of the claims are to be embraced within their scope.

[0020] Reference throughout this specification to features, advantages, or similar language does not imply that all of the features and advantages that may be realized with the present invention should be or are in any single embodiment of the invention. Rather, language referring to the features and advantages is understood to mean that a specific feature, advantage, or characteristic described in connection with an embodiment is included in at least one embodiment of the present invention. Thus, discussions of the features and advantages, and similar language, throughout this specification may, but do not necessarily, refer to the same embodiment.

[0021] Furthermore, the described features, advantages, and characteristics of the invention may be combined in any suitable manner in one or more embodiments. One skilled in the relevant art will recognize, in light of the description herein, that the invention can be practiced without one or more of the specific features or advantages of a particular embodiment. In other instances, additional features and advantages may be recognized in certain embodiments that may not be present in all embodiments of the invention.

[0022] Reference throughout this specification to "one embodiment," "an embodiment," or similar language means that a particular feature, structure, or characteristic described in connection with the indicated embodiment is included in at least one embodiment of the present invention. Thus, the phrases "in one embodiment," "in an embodiment," and similar language throughout this specification may, but do not necessarily, all refer to the same embodiment.

[0023] While many embodiments are described herein, at least some of the described embodiments facilitate integrating non-EDF jobs with EDF jobs within an EDF scheduling model. In some embodiments, statically scheduled tasks are executed concurrently with other tasks in a multiprogrammed real-time environment. In other embodiments, short-duration sporadic jobs (which may or may not be part of a static job schedule) are safely executed in a multiprogrammed real-time environment. More specifically, short-duration periodic or aperiodic pieces of code may be executed in a system scheduled with EDF.

[0024] As a result of the embodiments described herein, multiprogrammed real-time systems may safely execute legacy code, or other high-priority code such as interrupt service routines, which are designed to execute at specific times. Additionally, the execution of such legacy code can be implemented without causing other real-time and non-real-time processes to violate their timing requirements. This includes jobs implemented via legacy code designed for execution on isolated processors via static schedules. This enables legacy code to execute in modern real-time environments and facilitate graceful transitions from legacy code to modern real-time code designed for multiprogrammed environments. In other embodiments, the executed code is not necessarily legacy code.

[0025] As a matter of convenience, the terms "task" and "job" are used interchangeably within this description. These terms are not intended to have specific meanings herein, other than to reference a general amount of executable instructions or code. There is not necessarily any correlation between a task or job and a specific amount of code or typical types of scheduled units such as processes or threads (although some examples may use implementations with processes and/or threads). Additionally, there is no restriction on the type of multiprogramming environment in which tasks or jobs may be executed.

[0026] FIG. **1** depicts a schematic block diagram of one embodiment of a computer system **100** for integrating dissimilar job types into an integrated schedule for dynamic scheduling. Although the computer system **100** is illustrated with specific component parts which are described with certain functionality, other embodiments of the computer system

100 may include fewer or more component parts capable of implementing more or less functionality.

[0027] In the illustrated embodiment, the computer system **100** includes a processor **102** and a memory **104**. The processor **102** is representative of any type of processor which executes instructions or program code. Some examples of such processors include, but are not limited to, a central processing unit (CPU), a graphics processing unit (GPU), a microprocessor, a coprocessor, or another similar type of processor. In some embodiments, the illustrated processor **102** is representative of a multi-processor environment in which multiple processors cooperatively execute tasks or jobs in a multi-threaded or other parallel processing approach.

[0028] The memory **104** is representative of any type of memory that can store instructions or program code for execution by the processor **102**. In some embodiments, the memory **104** is separate from the processor **102**, as in the case of main memory or lower levels of cache. In other embodiments, the memory **104** is integrated with the processor **102**, as in the case of higher levels of cache that are implemented on the same die as the processor **102**.

[0029] The depicted memory **104** stores a program application **106** that is designed for execution on the processor **102**. The application **106** may be designed to be implemented according to an EDF schedule **108**. In this way, the application **106** is designed for implementation so that the task with the earliest deadline is implemented first, at any given time during the execution of the application **106**.

[0030] The memory **104** also stores another program application **110** that is not natively designed for execution according to an EDF schedule. Rather, this application **110** is designed for execution according to a static schedule **112**. In some embodiments, this type of situation occurs when a legacy application **110** from a legacy system (not shown) is loaded into the memory **104** of the computer system **100**. The legacy system may have been specially designed to implement the static schedule **112** of the legacy application **110** and, thus, is not inherently compatible with the computer system **100** onto which it is loaded.

[0031] The processor **102** includes an EDF scheduler **114** that schedules tasks for processing by the processor **102**. Specifically, the EDF scheduler **114** schedules the tasks according to an EDF scheduling model. In a specific embodiment, the EDF scheduler **114** implements a rate-based earliest deadline (RBED) scheduling model. For implementation of the EDF scheduling model, each task has an associated deadline, and the scheduler **114** schedules the tasks so that the tasks with the earliest deadlines are processed before the tasks with later deadlines. Assuming the processor **102** has sufficient resources, the scheduler **114** can satisfy all of the associated deadlines by allocating processor time for execution of specific tasks prior to their corresponding deadlines.

[0032] In the illustrated embodiment, the EDF scheduler **114** schedules tasks for both of the applications **106**, **110** stored in the memory. Hence, the scheduler **114** schedules EDF processes **116** for the application **106** designed for implementation with the EDF schedule **108**. Also, the scheduler **114** schedules static/sporadic processes **118** for the application **110** designed for implementation with the static schedule **112**.

[0033] In one embodiment, the scheduler **114** takes the tasks (e.g., processes **116**, **118**) for the applications **106**, **108** and creates an execution plan **120** for use by the processor **102**. In one embodiment, the scheduler **114** may create and store an actual execution plan **120** prior to execution of the tasks by the processor **102**. If the execution plan **120** is stored for later execution, the execution plan **120** may be stored in the memory **104** or on another memory device (e.g., on-board cache) coupled to the scheduler **114**. In another embodiment, the scheduler **114** dynamically creates the execution plan **120** so that the execution plan is created by the scheduler **114** at approximately the same time that the tasks are executed by the processor **102**.

[0034] FIG. **2** depicts a schematic diagram of one embodiment of a process **130** for integrating static jobs into an EDF schedule. For reference, certain component parts of the computer system **100** are shown in FIG. **2**. However, the flow of data and processing operations depicted in FIG. **2** are not necessarily indicative of a specific physical layout or data coupling of such component parts within the computer system **100**.

[0035] A periodic real-time task commonly includes a unit of work called a job that is repeatedly executed with a frequency equal to 1/p, where p is the period of the task. Thus, a task with a period of 1/10 of a second would execute the job as many as ten times per second. Jobs become ready to run at their release time and must complete by their deadline. A common practice is to make the deadline of the previous job the release time of the next job, and to set the deadline of each job equal to the period of the task. In that scenario, the execution of the task includes a sequence of jobs—one job per period, and each job is ready to run as soon as the previous job's deadline has been reached. Other embodiments may be implemented with other possible configurations that use different release times and/or deadlines.

[0036] In the illustrated embodiment, the scheduler **114** includes a static integration engine **132** which uses the EDF schedule **108** and the static schedule **112** to generate an integrated schedule **134**. An example of the EDF schedule **108** is shown in table format, including a list of jobs (designated as J_1 through J_N) with associated deadlines (D_1-D_N). The deadline indicates the time at which the job must be completed.

[0037] In one embodiment, tasks also may have associated rates (R_1-R_N) that indicate the fraction of the total processor time they will receive between each of their deadlines. To enforce these rates, a timer may be set to interrupt the processing of any job to prevent it from using more than its assigned rate by its assigned deadline. Each task initially has a budget. In one embodiment, the budget is calculated as B_i=R_i*P_i, where R_i is the task's rate, and P_i is the time between consecutive deadlines of the same task. Other embodiments may use other mathematical operations or algorithms to calculate the budget for a specific task. For example, in some embodiments, that budget may be calculated according to B_i=R_i/P_i. Each time a task executes, B_i is reduced by the amount of time the task executed. When the task is executed, the timer is set to go off B_i time units in the future. Upon receipt of such an interrupt, the scheduler may set the corresponding task to a state that is not currently executable and proceed to execute the executable task with the highest priority (earliest deadline). If the task will execute again with a new deadline, B_i may be replenished (e.g., again set to R_i*P_i) when the previous deadline has been reached.

[0038] An example of the static schedule **112** is also shown. The depicted static schedule **112** includes a list of jobs (designated as j_1 through j_n) and corresponding instants (i_1-i_n), which are also referred to as critical instants. In a legacy

system, the static schedule **112** would be executed on a dedicated processor that is not running any other jobs or tasks. The critical instants indicate the times by which the corresponding jobs must be completed and, in some embodiments, at which the next job must be started.

[0039] Typically, this type of static schedule would be designed for a specific processing environment to ensure that the tasks are executed exactly as indicated in the schedule to satisfy the critical instants. However, in embodiments described herein, the EDF scheduler **114** is capable of integrating the static jobs into a dynamic schedule for implementation on a processor other than the specific processor for which the static schedule was designed.

[0040] One example of an integrated schedule **134** is shown in FIG. **2**. The integrated schedule **134** includes the jobs from both the EDF schedule **108** and the static schedule **112**. In one embodiment, the entire static schedule **112** is treated as a non-periodic real-time task, and the critical instants of the static jobs are used as the deadlines of the jobs in the task in the integrated schedule **134**. When all of the jobs in the static schedule **112** have completed, the entire set of jobs may be repeated, similar to the way that individual jobs in a periodic task are repeated.

[0041] In one embodiment, the scheduler **114** also generates a new rate (designated as r_S) corresponding to the task containing the static jobs. In one embodiment, some or all of the static jobs are combined into a single task with a rate r_S that is a summation of individual rates (r__1 through r_n) that would otherwise be associated with the individual static jobs. An example of how to generate this rate is described in more detail below. However, embodiments of the scheduler **114** may use various methods for generating and assigning rates to the static jobs within the integrated schedule **134**.

[0042] In one embodiment, the scheduler **114** is an RBED integrated real-time scheduler which supports a variety of different types of processes. The scheduler **114** supports different processes by reserving a rate for each task. The rate may be expressed as any type of resource allocation for the processor **102**. In one embodiment, the rate is expressed as a percentage of CPU processing cycles. In some embodiments, the deadlines, or times at which the indicated resource allocation is expected to equal the corresponding reservation, are also represented. By implementing the scheduler **114** in this manner, all tasks and jobs, including static tasks and jobs, are scheduled according to EDF, or more specifically according to RBED.

[0043] In one embodiment, the scheduler **114** implements RBED to support statically scheduled tasks in the following manner. In this example, a static schedule (S) is designed to operate on a CPU (C__1) with a known frequency (F__1). The scheduler (S) schedules a set of jobs (J={j__1, j__2, . . . , j_n}) that have critical instants (I={i__2, i__3, . . . , i_n}) corresponding to the start times and finish times of the tasks in the static schedule. In other words, job j__1 must finish before time i__1, and job j__2 must start at time i__1 and finish before time i__2.

[0044] In order to run the jobs (J) that are originally in the static schedule (S) on a second CPU (C__2) with a frequency (F__2; where F__2 is greater than F__1) that also may be running other tasks scheduled with EDF, having assigned rates and deadlines, additional parameters are generated for the jobs (J). In particular, the jobs (J) can run on the second CPU (C__2) concurrently with other tasks by assigning the task containing jobs (J) the following processing parameters:

$$\text{Rate} = \frac{F_1}{F_2}; \text{ and Deadlines} = \{i_1, i_2, \ldots, i_n\}.$$

[0045] In one embodiment, the RBED implementation ensures that each task starts and finishes at the appropriate times. This is accomplished by ensuring that the expected resources, exactly equivalent to the resources that would be provided by the first CPU (C__1), are available between any two critical instants La and i_b. In other words, the scheduler **114** makes sure that the same or similar processing resources are available for the static jobs as the processing resources that would be available on another processor for which the static schedule is designed.

[0046] It should also be noted that although embodiments are described herein with specific reference to RBED, other embodiments may be implemented with another scheduling model based upon EDF, for example cluster based scheduling (CBS) or variable rate execution (VRE).

[0047] FIG. **3** depicts a schematic diagram of one embodiment of a process **140** for inserting idle jobs into the integrated schedule of FIG. **2**. In some embodiments, it may be useful to obscure the actual execution times of the jobs (J) from the static schedule (S) within the integrated schedule **134**. For example, there may be security reasons to prevent or impair detection of the execution times via covert channels based on CPU usage. In one embodiment, idle times within the static schedule (S) can be allocated to idle tasks that consume any unused CPU time.

[0048] In the illustrated embodiment, the scheduler **114** includes a utilization engine **142** to identify idle times within the static schedule (S) and to generate idle jobs (designated as j_iA A and j_iB) that consume some or all of the unused CPU time. Corresponding deadlines (i_iA and i_iB) (and optionally rates (r_iA and r_iB)) are also generated. Alternatively, some or all of the idle time may be available for allocation to other processes.

[0049] FIG. **4** depicts a schematic diagram of one embodiment of a process **150** for integrating a sporadic job **152** into an EDF schedule. For reference, certain component parts of the computer system **100** are shown in FIG. **4**. However, the flow of data and processing operations depicted in FIG. **4** are not necessarily indicative of a specific physical layout or data coupling of such component parts within the computer system **100**.

[0050] In some embodiments the sporadic jobs include interrupt service routines or jobs related to other externally generated events.

[0051] In the illustrated embodiment, the scheduler **114** includes a sporadic integration engine **154** which integrates the sporadic job **152** with the EDF schedule **108** to generate the integrated schedule **134**.

[0052] In an embodiment using the RBED scheduler, sporadic jobs may be handled by assigning to the job (designated s__1) a corresponding rate (r__1) and deadline (d__1). The job then may be scheduled along with all other jobs handled by the RBED scheduler.

[0053] In the illustrated example of the integrated schedule **134**, the sporadic job (designated as j_S) is scheduled between dynamic jobs J__2 and J__3. Also, the scheduler **114** generates a corresponding rate (r_S) and deadline (d_S) for the sporadic job (j_S).

[0054] In an RBED embodiment, the rate assigned to a sporadic job must be chosen so that the sum of the assigned rates is less than or equal to 100% of the CPU. In particular, if R is the sum of all of the assigned rates in the system prior to the arrival of the sporadic job, then the sporadic job may be assigned any rate r_S up to r'=100%−R. If the sporadic job should complete as quickly as possible, then it should be assigned the rate r_S=r'.

[0055] If a sporadic job has a worst case execution time e_S, then its deadline d_S may be set to its arrival time+e_S*r_S. The rate r_S is considered to be allocated and unavailable until either the deadline or until the arrival time e'_S*r_S, whichever is later, where e'_S is the actual execution time of the job.

[0056] If the resulting deadline of the sporadic job is the earliest deadline in the system, the job may begin execution immediately and, as long as no other job enters the system with an earlier deadline, it may safely execute to completion without interfering with the processing of any other job or task. This is useful in the case of interrupt service routines and other urgent but not necessarily compute-intensive sporadic processing.

[0057] This functionality can be used in at least two ways. First this type of functionality can be used for determining the feasibility of specific jobs or tasks with fixed scheduling requirements in a system of tasks scheduled with EDF. Second, this type of functionality can be used for determining at run-time when it is safe to allow interrupt service routines (or other asynchronous events) to be executed. As described above, the scheduler **114** may implement RBED or another type of EDF scheduling model.

[0058] FIG. **5** depicts a flow chart diagram of one embodiment of a method for operating the EDF scheduler **114** for the processor **102** within the computer system **100** of FIG. **1**. Although the method **200** for determining integration parameters is described in conjunction with the computer system **100** of FIG. **1**, other embodiments of the method **200** may be implemented with other types of computer systems.

[0059] In the illustrated embodiment, the depicted method **200** includes receiving **202** a plurality of jobs for an earliest deadline first (EDF) schedule. In one embodiment, the EDF scheduler **114** implements an EDF scheduling model using the jobs. The EDF scheduler **114** also receives **204** a separate job that is, in some embodiments, from a source other than the EDF schedule. The separate job has a fixed scheduling requirement with a specific execution time. In some embodiments, the separate job is a short duration sporadic job, and the fixed scheduling requirement is associated with the sporadic job. The EDF scheduler **114** then determines **206** an amount of available utilization capacity of the processor **102**. If the EDF scheduler **114** determines **208** that there is insufficient utilization capacity, then the EDF scheduler **114** may generate **210** a notification of the insufficient utilization capacity. Alternatively, the EDF scheduler inserts **212** the separate job into an execution plan of the processor **102** with the plurality of jobs from the EDF schedule **108** in response to a determination **208** that the available utilization capacity of the processor **102** is sufficient to execute the separate job according to the fixed scheduling requirement associated with the separate job. The depicted method **200** then ends.

[0060] It should also be noted that at least some of the operations for the methods may be implemented using software instructions stored on a computer useable storage medium for execution by a computer. As an example, an embodiment of a computer program product for inserting jobs into a scheduler, which implements an EDF scheduling protocol, of a processor includes a computer useable storage medium to store a computer readable program that, when executed on a computer, causes the computer to perform operations. In one embodiment, the operations include receiving a plurality of jobs from an earliest deadline first (EDF) schedule. The operations also include receiving a separate job from a source other than the EDF schedule. The separate job has a fixed scheduling requirement. The operations also include determining an amount of available utilization capacity of the processor. The operations also include inserting the separate job into an execution plan of the processor in response to a determination that the available utilization capacity of the processor is sufficient to execute the separate job according to the fixed scheduling requirement associated with the separate job.

[0061] Embodiments of the invention can take the form of an entirely hardware embodiment, an entirely software embodiment, or an embodiment containing both hardware and software elements. In one embodiment, the invention is implemented in software, which includes but is not limited to firmware, resident software, microcode, etc.

[0062] Furthermore, embodiments of the invention can take the form of a computer program product accessible from a computer-usable or computer-readable storage medium providing program code for use by or in connection with a computer or any instruction execution system. For the purposes of this description, a computer-usable or computer readable storage medium can be any apparatus that can store the program for use by or in connection with the instruction execution system, apparatus, or device.

[0063] The computer-useable or computer-readable storage medium can be an electronic, magnetic, optical, electromagnetic, infrared, or semiconductor system (or apparatus or device), or a propagation medium. Examples of a computer-readable storage medium include a semiconductor or solid state memory, magnetic tape, a removable computer diskette, a random access memory (RAM), a read-only memory (ROM), a rigid magnetic disk, and an optical disk. Current examples of optical disks include a compact disk with read only memory (CD-ROM), a compact disk with read/write (CD-R/W), and a digital video disk (DVD).

[0064] An embodiment of a data processing system suitable for storing and/or executing program code includes at least one processor coupled directly or indirectly to memory elements through a system bus such as a data, address, and/or control bus. The memory elements can include local memory employed during actual execution of the program code, bulk storage, and cache memories which provide temporary storage of at least some program code in order to reduce the number of times code must be retrieved from bulk storage during execution.

[0065] Input/output (I/O) devices (including but not limited to keyboards, displays, pointing devices, etc.) can be coupled to the system either directly or through intervening I/O controllers. Additionally, network adapters also may be coupled to the system to enable the data processing system to become coupled to other data processing systems or remote printers or storage devices through intervening private or public networks. Modems, cable modems, and Ethernet cards are just a few of the currently available types of network adapters.

[0066] Although the operations of the method(s) herein are shown and described in a particular order, the order of the

operations of each method may be altered so that certain operations may be performed in an inverse order or so that certain operations may be performed, at least in part, concurrently with other operations. In another embodiment, instructions or sub-operations of distinct operations may be implemented in an intermittent and/or alternating manner.

[0067] Although specific embodiments of the invention have been described and illustrated, the invention is not to be limited to the specific forms or arrangements of parts so described and illustrated. The scope of the invention is to be defined by the claims appended hereto and their equivalents.

What is claimed is:

1. A computer program product comprising:

a computer readable storage medium to store a computer readable program, wherein the computer readable program, when executed on a computer, causes the computer to perform operations for inserting jobs into a scheduler of a processor, the operations comprising:

receiving a plurality of jobs from an earliest deadline first (EDF) schedule, wherein the scheduler implements an EDF scheduling model;

receiving a separate job from a source other than the EDF schedule, wherein the separate job has a fixed scheduling requirement;

determining an amount of available utilization capacity of the processor; and

inserting the separate job into an execution plan of the processor in response to a determination that the available utilization capacity of the processor is sufficient to execute the separate job according to the fixed scheduling requirement associated with the separate job.

2. The computer program product of claim **1**, wherein the fixed scheduling requirement comprises a specific execution time for the separate job.

3. The computer program product of claim **1**, wherein the separate job comprises a static job from a static schedule, wherein the static job is natively scheduled for implementation according to a static scheduling model.

4. The computer program product of claim **3**, wherein the operations further comprise:

obtaining a critical instant for the static job from the static schedule, wherein the critical instant indicates a specific execution time requirement of the static job; and

assigning a deadline to the static job within the execution plan, wherein the deadline is based on the critical instant of the static job.

5. The computer program product of claim **4**, wherein the operations further comprise reserving a rate for the static job, wherein the rate comprises a utilization factor indicative of an amount of the utilization capacity of the processor that is allocated to the static job.

6. The computer program product of claim **5**, wherein the operations further comprise calculating the rate for the static job according to the following ratio:

$$\text{Rate} = \frac{F_1}{F_2},$$

in which F_2 represents an operating frequency of the processor on which the static job executes, and F_1 represents an operating frequency of another processor on which the static job is designed to execute.

7. The computer program product of claim **6**, wherein the operations further comprise reserving resources on the processor that are substantially equivalent to resources designed to be available for execution of the static job on the other processor.

8. The computer program product of claim **5**, wherein the operations further comprise scheduling the static job for execution on the processor concurrently with other jobs from the EDF schedule.

9. The computer program product of claim **3**, wherein the operations further comprise scheduling an idle task at a time corresponding to an unscheduled time within the static schedule, wherein the idle task is configured to consume an unused processor resource.

10. The computer program product of claim **1**, wherein the separate job comprises a short duration sporadic job, wherein the sporadic job is assigned an earliest deadline, is allowed to execute to completion, and affects the EDF schedule.

11. The computer program product of claim **1**, wherein the sporadic job comprises an interrupt service routine.

12. The computer program product of claim **1**, wherein the sporadic job comprises a specific execution time requirement.

13. A system comprising:

a processor configured to execute instructions related to a plurality of jobs; and

a scheduler coupled to the processor, wherein the scheduler is configured to implement an earliest deadline first (EDF) scheduling model, and the scheduler is further configured to:

receive a plurality of jobs from an EDF schedule;

receive a separate job from a source other than the EDF schedule, wherein the separate job has a fixed scheduling requirement; and

insert the separate job into an execution plan of the processor in response to a determination that an available utilization capacity of the processor is sufficient to execute the separate job according to the fixed scheduling requirement associated with the separate job.

14. The system of claim **13**, wherein the separate job comprises a static job from a static schedule, wherein the static job is natively scheduled for implementation at a specific execution time according to a static scheduling model, and the scheduler further comprises a static integration engine to integrate the static job into the execution plan with the plurality of jobs from the EDF schedule.

15. The system of claim **14**, wherein the scheduler is further configured to:

obtain a critical instant for the static job from the static schedule, wherein the critical instant indicates the specific execution time requirement of the static job;

assign a deadline to the static job within the execution plan, wherein the deadline is based on the critical instant of the static job; and

reserve a rate for the static job, wherein the rate comprises a utilization factor indicative of an amount of the utilization capacity of the processor that is allocated to the static job.

16. The system of claim **14**, wherein the scheduler further comprises a utilization engine to schedule an idle task at a time corresponding to an unscheduled time within the static schedule, wherein the idle task is configured to consume an unused processor resource.

17. The system of claim **13**, wherein the separate job comprises a short duration sporadic job, wherein the sporadic job is assigned an earliest deadline, is allowed to execute to completion, and affects the EDF schedule, and the scheduler further comprises a sporadic integration engine to integrate the sporadic job into the execution plan with the plurality of jobs from the EDF schedule.

18. A method for implementation within a scheduler for a processor, the method comprising:

receiving a plurality of jobs from an earliest deadline first (EDF) schedule, wherein the scheduler implements an EDF scheduling model;

receiving a separate job from a source other than the EDF schedule, wherein the separate job has a fixed scheduling requirement with a specific execution time;

determining an amount of available utilization capacity of the processor; and

inserting the separate job into an execution plan of the processor with the plurality of jobs from the EDF schedule in response to a determination that the available utilization capacity of the processor is sufficient to execute the separate job according to the fixed scheduling requirement associated with the separate job.

19. The method of claim **18**, wherein the separate job comprises a static job from a static schedule, and the static job is natively scheduled for implementation according to a static scheduling model, and the method further comprises:

obtaining a critical instant for the static job from the static schedule, wherein the critical instant indicates the specific execution time requirement of the static job;

assigning a deadline to the static job within the execution plan, wherein the deadline is based on the critical instant of the static job; and

reserving a rate for the static job, wherein the rate comprises a utilization factor indicative of an amount of the utilization capacity of the processor that is allocated to the static job, and the rate is calculated according to the following ratio:

$$\text{Rate} = \frac{F_1}{F_2},$$

in which F__2 represents an operating frequency of the processor on which the static job executes, and F__1 represents an operating frequency of another processor on which the static job is designed to execute.

20. The method of claim **18**, wherein the separate job comprises a short duration sporadic job, wherein the sporadic job is assigned an earliest deadline, is allowed to execute to completion, and affects the EDF schedule, and the method further comprises integrating the sporadic job into the execution plan with the plurality of jobs from the EDF schedule.

* * * * *