METHOD AND SYSTEM FOR DYNAMICALLY PARALLELIZING APPLICATION PROGRAM

Inventors: Seung Won Lee, Hwaseong-si (KR); Shi Hwa Lee, Seoul (KR); Dong-In Kang, San Jose, CA (US); Minkyung Kang, Jeju Special Self-Governing Province (KR)

Correspondence Address: STAAS & HALSEY LLP SUITE 700, 1201 NEW YORK AVENUE, N.W. WASHINGTON, DC 20005 (US)

Assignee: SAMSUNG ELECTRONICS CO., LTD, Suwon-si (KR)

Appl. No.: 12/662,657

Filed: Apr. 27, 2010

ABSTRACT

Provided is a method and system for dynamically parallelizing an application program. Specifically, provided is a method and system having multi-core control that may verify a number of available threads according to an application program and dynamically parallelize data based on the verified number of available threads. The method and system for dynamically parallelizing the application program may divide a data block to be processed according to the application program based on a relevant data characteristic and dynamically map the threads to division blocks, and thereby enhance a system performance.
FIG. 2A

1
1
1
2
2

BLOCK ALLOCATION SCHEME

FIG. 2B

1
2
1
2
1

CYCLING SCHEME
FIG. 4

START

1. VERIFY DATA AREA
2. GENERATE DATA BLOCK
3. VERIFY NUMBER OF AVAILABLE THREADS
4. DIVIDE DATA BLOCK

Determine Thread Allocation Scheme

Generate Threads & Allocate Threads to Division Blocks

Provide Data

Perform Unique Function

Store Operation Result

Combine Threads into Single Thread

COMPLETED?

CHANGE EVENT?

YES

VERIFY NUMBER OF AVAILABLE THREADS

CHANGE DIVISION TYPE OF DATA BLOCK

NO

END
METHOD AND SYSTEM FOR DYNAMICALLY PARALLELIZING APPLICATION PROGRAM

CROSS-REFERENCE TO RELATED APPLICATIONS

[0001] This application claims the benefit of Korean Patent Application No. 10-2009-0037726, filed on Apr. 29, 2009, in the Korean Intellectual Property Office, the disclosure of which is incorporated herein by reference.

BACKGROUND

[0002] 1. Field

[0003] One or more embodiments relate to a method and system for dynamically parallelizing an application program, and more particularly, a method and system having multi-core control that may dynamically divide a data block to be processed according to an application program, and may dynamically allocate threads to division blocks, respectively, and thereby perform a dynamic parallelization.

[0004] 2. Description of the Related Art

[0005] Currently, processors including a plurality of cores to maximize silicon efficiency are popular. Because of an increase in processor density, a parallel scheme of controlling the plurality of cores, that is, multi-core control is proposed.

[0006] The parallel scheme is widely used for cluster super computers and general personal computers (PCs), due to a dissemination of a multi-core technology.

[0007] A multi-core processor having a few to tens of cores is currently produced. In the near future, a multi-core processor having hundreds or more cores will be produced.

[0008] In a parallel system having the above multi-core processor, many application programs may share increased resources. In this case, the resource demand of the application programs needs to be changed depending on an amount of available resources.

[0009] However, even when the amount of available resources is changed, a conventional parallel system may not dynamically utilize the available resources.

[0010] In a situation where a parallel application program dynamically operates in the parallel system, a number of threads determined when the parallel application program is generated may be used as resources without considering the number of available threads corresponding to the available resources. Accordingly, when more resources are desired to be used in the parallel application program, a source code may need to be changed and re-compiled.

SUMMARY

[0011] According to one aspect of the present disclosure, there may be provided a system for dynamically parallelizing an application program, the system including a multi-core controller to assign a unique operation to each of plural threads that are generated while running the parallel application program, a block generator to generate a data block having the size of a data area to be processed by the parallel application program, using a block function, a monitoring unit to monitor the multi-core controller to determine a number of available threads, and a decision unit to divide the data block using a map function, and to dynamically determine a thread allocation scheme.

[0012] The decision unit may determine a division type of the data block and the thread allocation scheme based on a characteristic of data stored in the data block and the determined number of available threads.

[0013] The system may further include a scatter unit to generate threads of performing a same operation using a scatter function, and to allocate the threads to division blocks, respectively, according to the determined thread allocation scheme, and a gather unit to combine the threads using a gather function.

[0014] The system may further include a view unit to read data from a corresponding division block in response to a request of each of the threads, and to provide the read data to a corresponding thread, or to store a result of the thread in the corresponding division block using a view function.

[0015] According to another aspect of the present disclosure, there may be provided a method for dynamically parallelizing an application program in a system, the method including generating a data block having the size of a data area to be processed by the application program, using a block function, determining a number of available threads, dividing the data block using a map function, and dynamically determining a thread allocation scheme, and generating threads according to the thread allocation scheme to allocate the threads to division blocks, respectively.

[0016] The dividing of the data block and the dynamically determining of the thread allocation scheme further may include determining a division type of the data block and the thread allocation scheme based on a characteristic of data stored in the data block and the determined number of available threads.

[0017] The generating of the threads may include generating threads of performing a same operation using a scatter function, to allocate the threads to division blocks, respectively, according to the dynamically determined thread allocation scheme, reading data from a corresponding division block in response to a request of each of the threads, to provide the read data to a corresponding thread using a view function, performing a unique operation in each of the threads, storing a result of the thread in a corresponding division block using the view function, and combining the threads using a gather function.

[0018] Additional aspects, features, and/or advantages of embodiments will be set forth in part in the description which follows and, in part, will be apparent from the description, or may be learned by practice of the disclosure.

BRIEF DESCRIPTION OF THE DRAWINGS

[0019] These and/or other aspects and advantages will become apparent and more readily appreciated from the following description of the exemplary embodiments, taken in conjunction with the accompanying drawings of which:

[0020] FIG. 1 illustrates a configuration of a system for dynamically parallelizing an application program, according to one or more embodiments;

[0021] FIGS. 2A and 2B illustrate allocating of threads in a system, according to one or more embodiments;

[0022] FIGS. 3A through 3D illustrate parallelizing of an application program of a system, according to one or more embodiments;

[0023] FIG. 4 illustrates a flowchart of a method for dynamically parallelizing an application program in a system, according to one or more embodiments; and
FIG. 5 illustrates an allocating of threads for a JPEG codec, according to one or more embodiments.

DETAILED DESCRIPTION

Reference will now be made in detail to embodiments, examples of which are illustrated in the accompanying drawings, wherein like reference numerals refer to like elements throughout. In this regard, embodiments of the present invention may be embodied in many different forms and should not be construed as being limited to embodiments set forth herein. Accordingly, embodiments are merely described below, by referring to the figures, to explain aspects of the present invention.

According to one or more embodiments, there is provided a system and method having multi-core control that may dynamically divide a data block to be processed according to an application program and dynamically allocate threads to division blocks, respectively, and thereby perform a dynamic parallelization.

FIG. 1 illustrates a configuration of a system for dynamically parallelizing a parallel application program, according to one or more embodiments.

The system may include a multi-core controller 110, a block generator 111, a monitoring unit 112, a performance measurement unit 113, a decision unit 114, a scatter unit 115, a view unit 116, a gather unit 117, a change unit 118, a non-volatile memory 120, and a data memory 130, for example.

The non-volatile memory 120 may store a system parameter and storage data. The non-volatile memory 120 may store an application program, and may store input data to be processed by the application program.

The data memory 130 may temporarily store a data block to be processed by the application program. The data memory 130 may use a Random Access Memory (RAM), for example.

The multi-core controller 110 may have at least two cores, and may function to drive the application program, and to assign a unique function or operation to each of threads generated according to the application program. A number of threads may be generated to be within the number of available cores.

The block generator 111 may verify a data area to be processed by the application program, generate a data block having the size of the data area using a block function, and store the generated data block in the data memory 130.

The monitoring unit 112 may monitor the multi-core controller 110 to verify a number of available threads. The number of available threads indicates a total number of threads that may be simultaneously processed based on a number of available cores that are in an idle state.

The performance measurement unit 113 may measure or predict a performance of the thread allocation system according to the application program.

When a decision request is received according to a progress sequence of the application program, the decision unit 114 may divide the generated data block using a map function, and determine a thread allocation scheme. The data block may be divided according to an input of a programmer, or may be divided based on a characteristic of data stored in the data block.

To consider the number of available threads and the characteristic of data stored in the data block, the decision unit 114 may utilize the thread allocation scheme and a division of the data block having a determined optimum performance based on the measurement or prediction of the thread allocation system according to the parallel application program.

FIGS. 2A and 2B illustrate an allocating of threads in a system, according to one or more embodiments.

FIGS. 2A and 2B show a scheme of allocating two threads {1, 2}. FIG. 2A shows a block allocation scheme, as an available thread allocation scheme. The block allocation scheme may divide a data block into division blocks having approximately equal sizes, and allocate the same thread to adjacent division blocks. The block allocation scheme may be appropriate when data correlation between the adjacent division blocks is relatively high according to an aspect of the relevant data characteristic.

FIG. 2B shows a cycling scheme, as another available thread allocation scheme. The cycling scheme may alternately allocate a thread to division blocks based on a round-robin scheme. The cycling scheme may be appropriate when a data correlation between the adjacent division blocks is relatively low according to an aspect of the relevant data characteristic.

In addition to the aforementioned two schemes, various types of thread allocation schemes may alternatively be applicable.

The decision unit 114 may divide the data block based on the number of available threads and the corresponding data characteristic since the number of available threads and the data characteristic may greatly affect the performance of the application program resulting from the number of divisions and division type. The performance of the application program may vary depending on which thread allocation scheme is used.

When a distribution request is received, according to the progress sequence of the application program, the scatter unit 115 may generate threads for performing the same operation using a scatter function, and may allocate the threads to the division blocks, respectively, according to the determined thread allocation scheme.

The view unit 116 may read data from each of the division blocks allocated with the threads, and provide the data to each of the threads using a view function. The view unit 116 may receive an operation result from each of the threads, and may store the operation result in a division block corresponding to each of the threads using the view function.

Specifically, in response to a request from a thread, the view unit 116 may apply the view function to the thread, and may approach a corresponding data block of the thread to thereby read and write data.

When a combination request is received according to the progress sequence of the parallel application program, the gather unit 117 may combine distributed threads into a single thread using a gather function.

When a request of a programmer, for example, is received, or when a change event according to a change in the number of available threads is detected via the monitoring unit 112, the change unit 118 may increase or decrease the number of threads using a change function. When a change request is received according to the progress sequence of the application program, the change unit 118 may increase or decrease the number of threads using the change function.

The multi-core controller 110 may control the application program and the threads, and may also control the block generator 111, the monitoring unit 112, the perfor-
The application program of FIG. 4 may have a thread structure of a basic parallel form, for example. In operation 410, the application program may verify a data area to process. In operation 412, the application program may generate a data block having a size corresponding to the data area.

In operation 414, the application program may verify a number of available threads. In operation 416, the application program may divide a data block based on the number of available threads and a determined characteristic of the data stored in the data block. In operation 418, the application program may determine a thread allocation scheme based on the number of available threads and the data characteristic.

In operation 420, the application program may generate threads to perform the same operation using a scatter function and allocate the threads to division blocks, respectively, according to the determined thread allocation scheme.

In operation 422, the application program may provide data of a division block corresponding to each of the threads using a view function. In operation 424, the application program may perform a unique operation in each of the threads. In operation 426, the parallel application program may store an operation result of each of the threads in a corresponding division block using the view function.

In operation 428, the application program may combine the distributed threads into a single thread using a gather function.

In operation 430, the application program may verify whether the dynamic parallelization of the application program is completed.

When the dynamic parallelization is completed, a dynamic parallelization algorithm may be terminated. Conversely, when the dynamic parallelization is not completed, the application program may determine whether a change event is detected in operation 432. The change event may occur according to a request of a programmer, for example, a change in the number of available threads, or a progress sequence that is preset when the application program is generated.

When the change event is detected, the application program may verify the number of available threads in operation 434, and change a division type of the data block based on the number of available threads and the characteristic of the data stored in the data block in operation 436. The application program may then return to operation 418.

Conversely, when the change event is not detected, the application program may just return to operation 418.

FIG. 5 illustrates an allocating of threads in a J2K codec, according to one or more embodiments. Referring to FIG. 5, the illustrated left image to be encoded may have a 2048x2048 pixel size. As illustrated, when the image is divided into 256x256 pixel sizes, the image may include 64 tile images.

Thus, the illustrated left image side of FIG. 5 shows the case where the J2K codec performs encoding without dynamic parallelization, i.e., a single thread may be needed to encode each of the 64 tile images. Accordingly, the total encoding time may be 64 times the amount of time needed to encode a single tile.

The illustrated right image of FIG. 5 shows the case where the J2K codec performs encoding according to a dynamic parallelization. Here, four available threads A1, A2, A3, and A4 may exist. In this case, it is possible to decrease
the amount of time by a factor of four, compared to the non-dynamic parallelization encoding, by parallelizing the four threads.

[0071] Specifically, as shown in the illustrated right image of FIG. 5, when the threads are paralleled, it is possible to enhance a system performance by maximally using the available threads.

[0072] According to one or more embodiments, there is provided a system and method having multi-core control that may dynamically divide a data block to be processed by an application program and dynamically allocate distributed threads to division blocks, respectively, and thereby may perform a dynamic parallelization. Through this, it is possible to verify a number of available threads and to dynamically parallelized data. The system performance may be enhanced by dividing a data block and dynamically allocating the threads using a most efficient scheme based on a characteristic of the data stored in the data block and the number of available threads when dividing a data block and dynamically allocating the threads.

[0073] The above-described exemplary embodiments may be implemented by non-transient computer-readable media including computer readable code, such as program instructions, to control at least one processing device, such as a processor or computer. The media may also include, in combination with the example program instructions, data files, data structures, and the like. Examples of computer-readable media include: magnetic media such as hard disks, floppy disks, and magnetic tape; optical media such as CD ROM disks and DVDs; magneto-optical media such as optical disks; and hardware devices that are specially configured to store and perform program instructions, such as read-only memory (ROM), random access memory (RAM), flash memory, and the like. Examples of computer readable code include both machine code, such as produced by a compiler, and files containing higher level code that may be executed by the at least one processing device using an interpreter.

[0074] While aspects of the present invention has been particularly shown and described with reference to differing embodiments thereof, it should be understood that these embodiments should be considered in a descriptive sense only and not for purposes of limitation. Descriptions of features or aspects within each embodiment should typically be considered as available for other similar features or aspects in the remaining embodiments.

[0075] Thus, although a few embodiments have been shown and described, with additional embodiments being equally available, it would be appreciated by those skilled in the art that changes may be made in these embodiments without departing from the principles and spirit of the invention, the scope of which is defined in the claims and their equivalents.

What is claimed is:

1. A system for dynamically parallelizing an application program, the system comprising:
   a multi-core controller to assign a unique operation of each of plural threads that are generated while running the application program;
   a block generator to generate a data block having a size corresponding to a data area to be processed by the application program, using a block function;
   a monitoring unit to monitor the multi-core controller to determine a number of available threads; and
   a decision unit to divide the data block using a map function, and to dynamically determine a thread allocation scheme.

2. The system of claim 1, wherein the decision unit determines a division type of the data block and the thread allocation scheme based on a characteristic of data stored in the data block and the determined number of available threads.

3. The system of claim 1, wherein the thread allocation scheme corresponds to any one of a block allocation scheme of allocating a same thread to an adjacent division block, and a cycling scheme of alternately allocating a thread according to a round-robin scheme.

4. The system of claim 1, further comprising:
   a performance measurement unit to measure or predict a performance of the thread allocation system according to a division type of the data block and the thread allocation scheme,
   wherein the decision unit dynamically determines the thread allocation scheme and the division type of the data block having a determined optimum performance based on the measurement or prediction of the performance measurement unit.

5. The system of claim 1, further comprising:
   a scatter unit to generate threads of performing a same operation using a distribution, and to allocate the threads to division blocks, respectively, according to the determined thread allocation scheme; and
   a gather unit to combine the threads using a combination function.

6. The system of claim 5, further comprising:
   a view unit to read data from a corresponding division block in response to a request of each of the threads, and to provide the read data to a corresponding thread, or to store a result of the thread in the corresponding division block using a view function.

7. The system of claim 5, further comprising:
   a change unit to selectively increase or decrease the number of threads using a change function when a change in the number of available threads is detected by the monitoring unit.

8. The system of claim 7, wherein, when a change in the number of available threads is detected by the monitoring unit, the change unit requests the decision unit to change a division type of the data block and the thread allocation scheme.

9. A method for dynamically parallelizing an application program in a system, the method comprising:
   generating a data block having a size corresponding to a data area to be processed by the parallel application program, using a block function;
   determining a number of available threads;
   dividing the data block using a map function, and dynamically determining a thread allocation scheme; and
   generating threads according to the thread allocation scheme to allocate the threads to division blocks, respectively.

10. The method of claim 9, wherein the dividing of the data block and the dynamically determining of the thread allocation scheme further comprises determining a division form of the data block and the thread allocation scheme based on a characteristic of data stored in the data block and the determined number of available threads.

11. The method of claim 9, wherein the thread allocation scheme corresponds to any one of a block allocation scheme
of allocating a same thread to an adjacent division block, and
a cycling scheme of alternately allocating a thread according
to a round-robin scheme.

12. The method of claim 9, wherein the dividing of the data
block and the dynamically determining of the thread allocation
scheme further comprises:
measuring or predicting a performance of the thread allo-
cation system according to a division type of the data
block and the thread allocation scheme; and
determining the thread allocation scheme and the division
form of the data block having a determined optimum
performance based on the measured or predicted perform-
ance.

13. The method of claim 9, wherein the generating of the
threads further comprises:
generating threads of performing a same operation using a
scatter function, to allocate the threads to division
blocks, respectively, according to the dynamically deter-
mined thread allocation scheme;
reading data from a corresponding division block in
response to a request of each of the threads, to provide
the read data to a corresponding thread using a view
function;
performing a unique operation in each of the threads;
storing a result of the thread in a corresponding division
block using the view function; and
combining the threads using a gather function.

14. The method of claim 13, wherein the generating of the
threads further comprises:
selectively increasing or decreasing the number of threads
using a change function when a change in the deter-
mined number of available threads is detected.

15. The method of claim 14, wherein the generating of the
threads further comprises:
changing a division type of the data block and the thread
allocation scheme using the map function when the
change in the determined number of available threads is
detected.