



US 20040093477A1

(19) **United States**

(12) **Patent Application Publication** (10) **Pub. No.: US 2004/0093477 A1**

**Oberdorfer**

(43) **Pub. Date: May 13, 2004**

(54) **SCALABLE PARALLEL PROCESSING ON SHARED MEMORY COMPUTERS**

**Publication Classification**

(76) **Inventor: Matthias Oberdorfer, Fort Collins, CO (US)**

(51) **Int. Cl.<sup>7</sup> ..... G06F 15/00**

(52) **U.S. Cl. .... 712/13; 712/15**

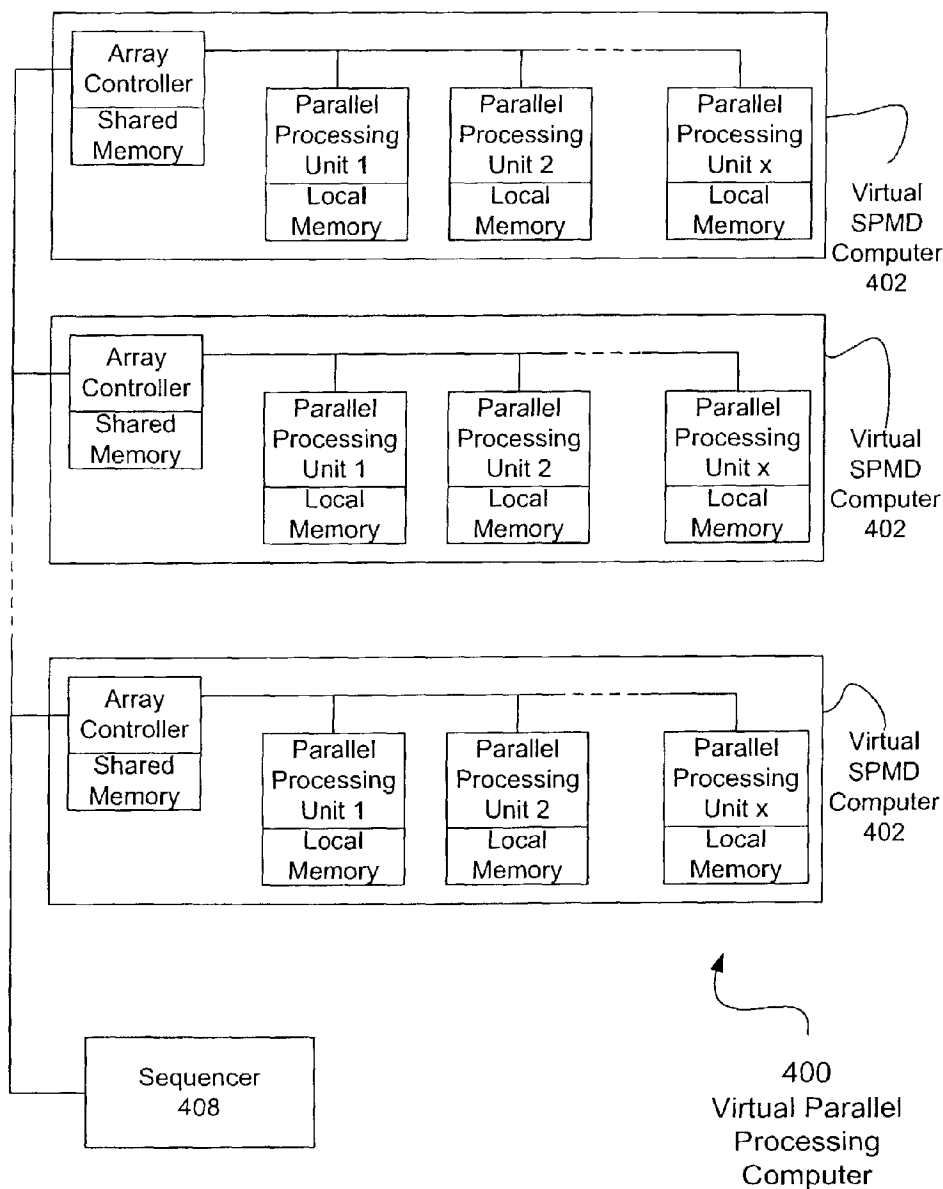
Correspondence Address:  
**COCHRAN FREUND & YOUNG LLC  
3555 STANFORD ROAD  
SUITE 230  
FORT COLLINS, CO 80525 (US)**

(57) **ABSTRACT**

A virtual parallel computer is created within a programming environment comprising both shared memory and distributed memory architectures. At run time, the virtual architecture is mapped to a physical hardware architecture. In this manner, a massively parallel computing program may be developed and tested on a first architecture and run on a second architecture without reprogramming.

(21) **Appl. No.: 10/293,791**

(22) **Filed: Nov. 12, 2002**



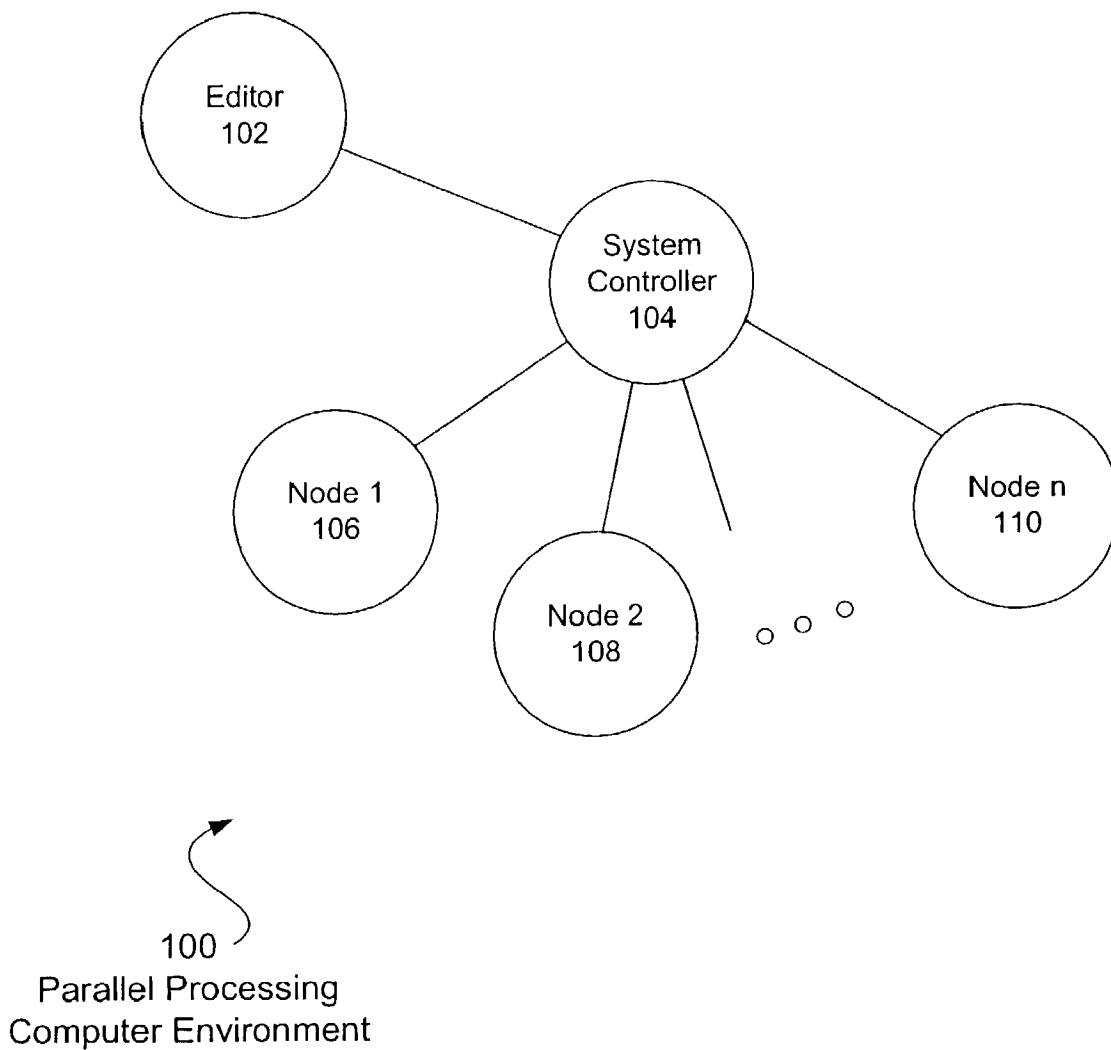


FIGURE 1

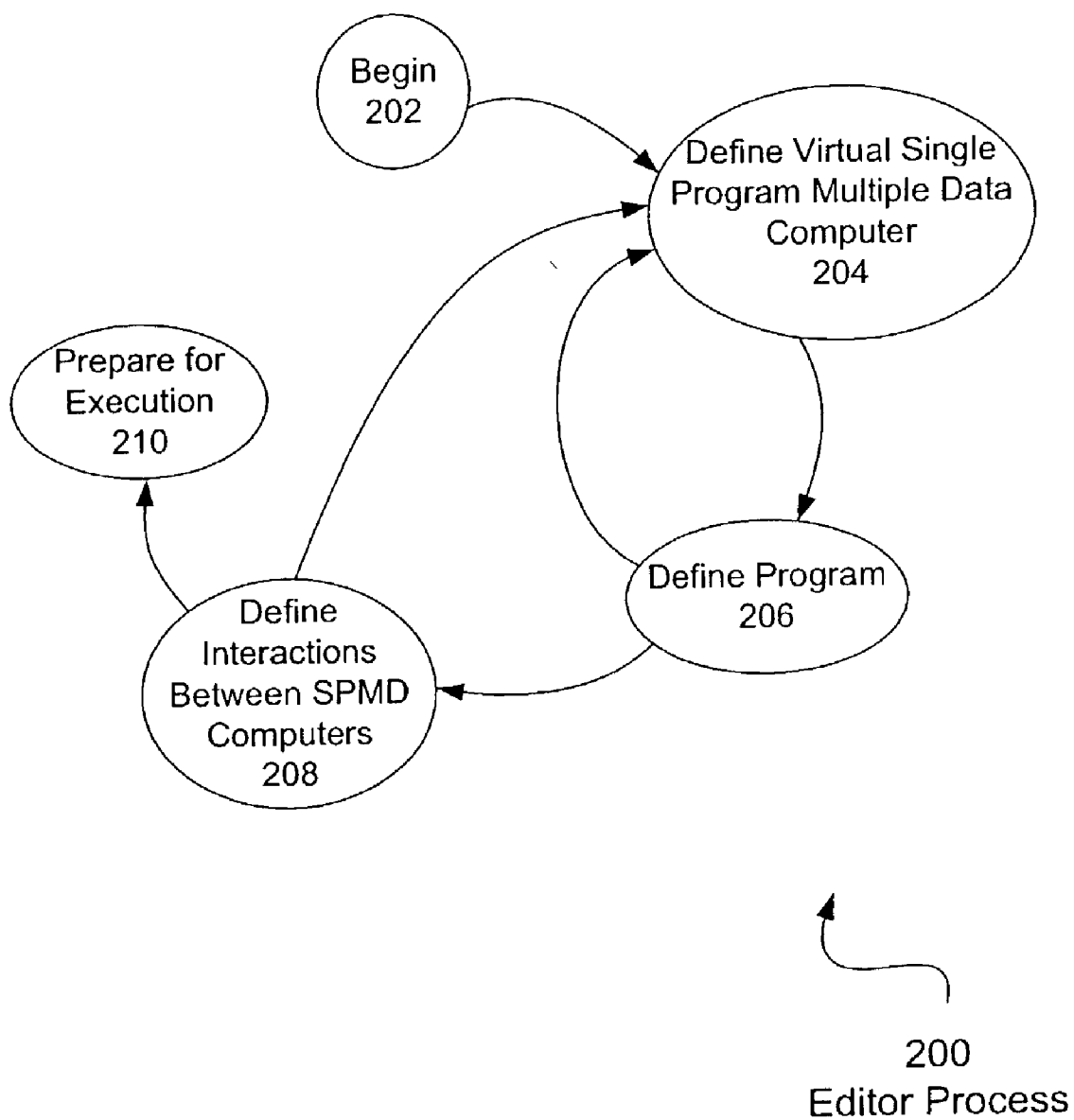


FIGURE 2

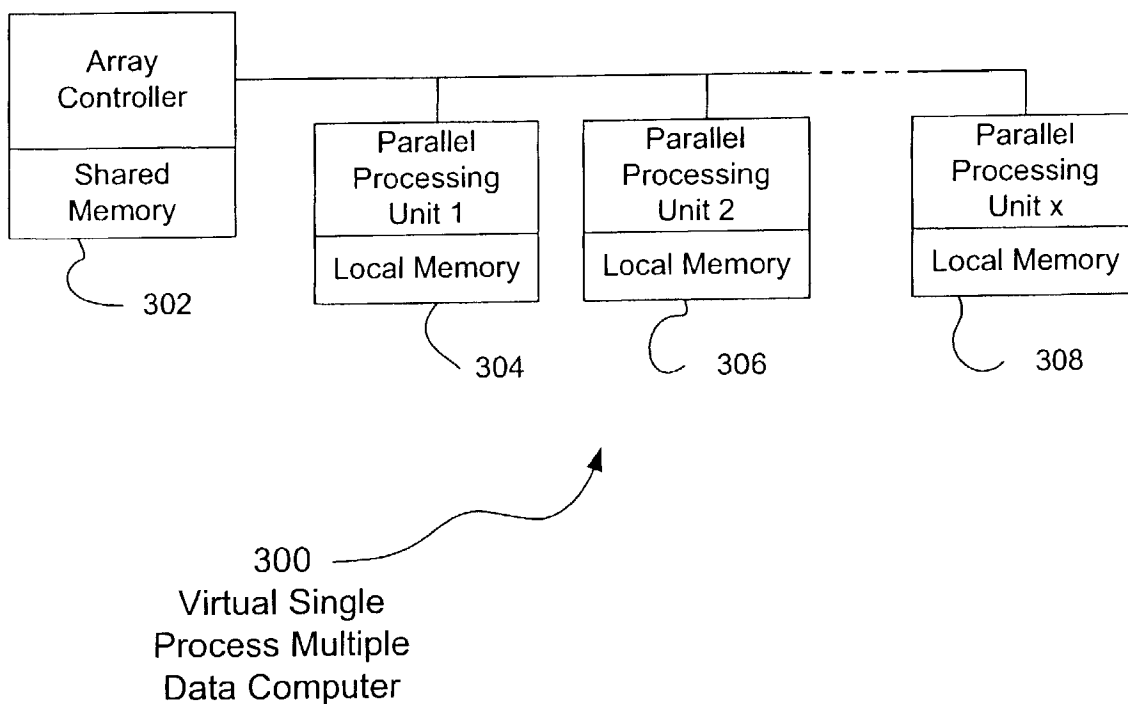


FIGURE 3

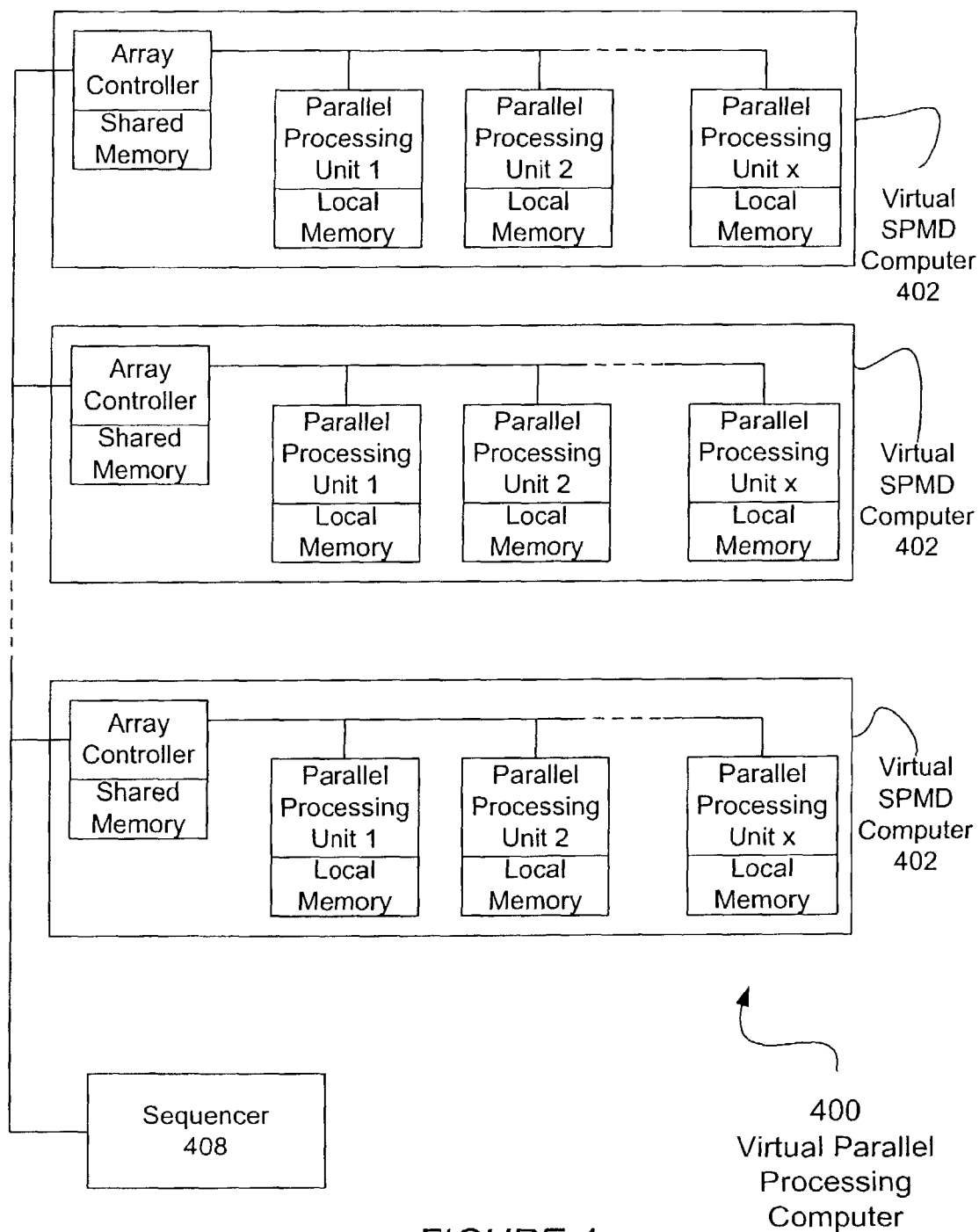


FIGURE 4

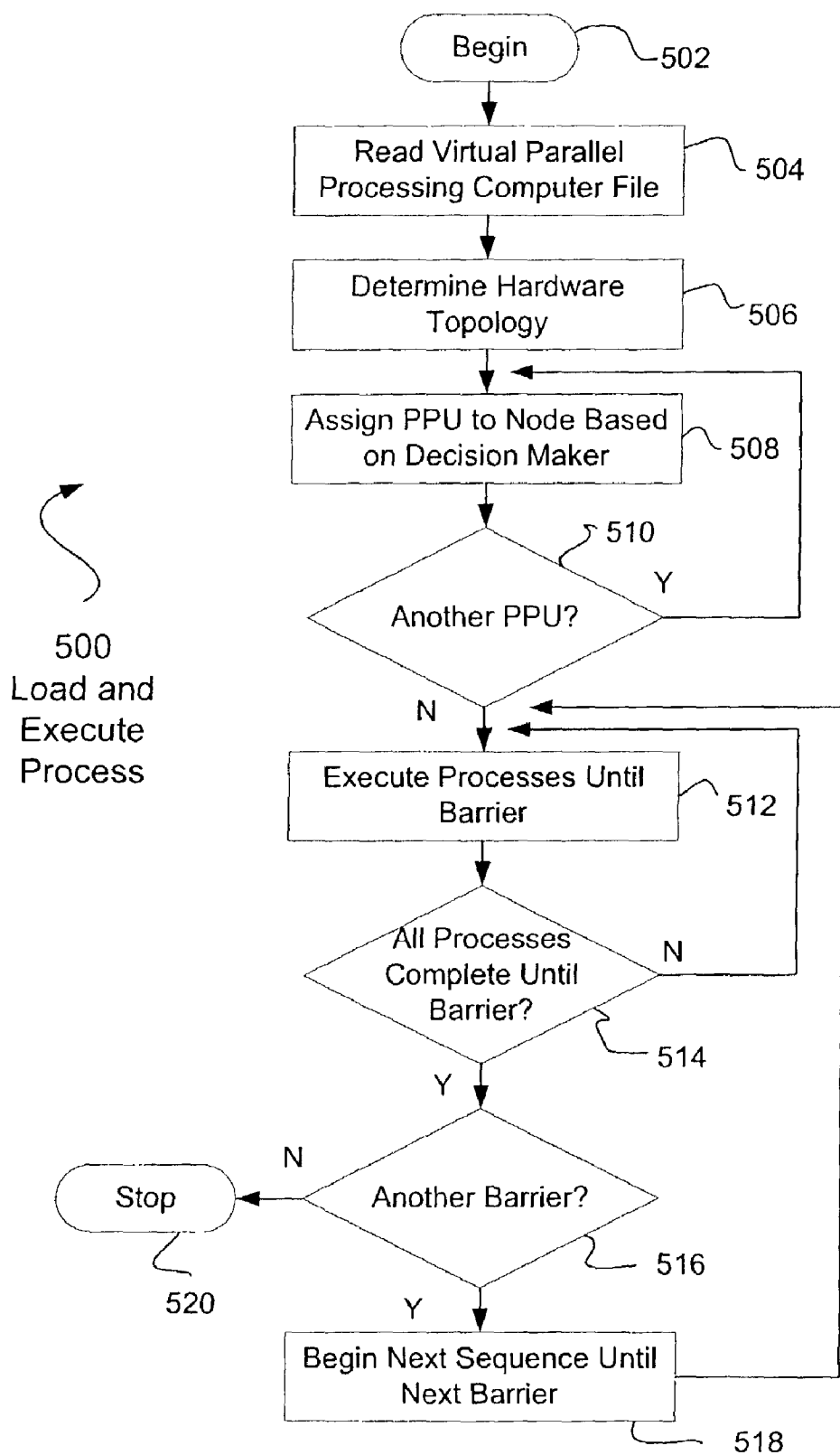


FIGURE 5

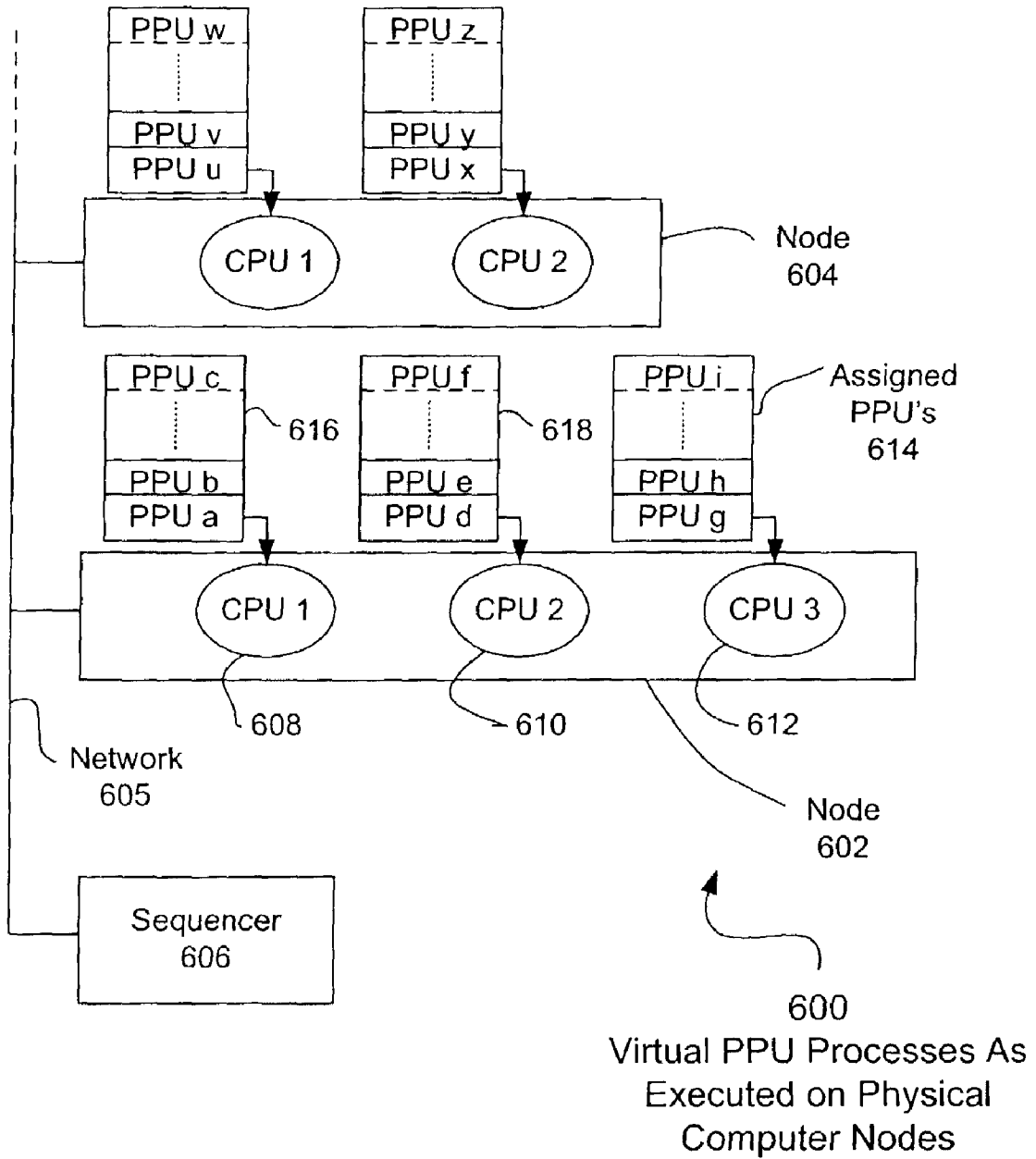


FIGURE 6

## SCALABLE PARALLEL PROCESSING ON SHARED MEMORY COMPUTERS

### CROSS REFERENCE TO RELATED APPLICATIONS

[0001] The present application is filed simultaneously with application Ser. No. \_\_\_\_\_ entitled "Connected Memory Management" by the present inventor, Matthias Oberdorfer, filed 12 Nov. 2002, the full text of which is hereby specifically incorporated by reference for all it discloses and teaches.

### BACKGROUND OF THE INVENTION

[0002] a. Field of the Invention

[0003] The present invention pertains to parallel processing computers and specifically to processing on clusters of computers.

[0004] b. Description of the Background

[0005] Parallel processing on multiple computers has been a challenging and complex task that has always required a high degree of custom software. In many cases, software must be created for a specific physical configuration of computers or processors.

[0006] Many types of parallel computing are known in the art. Shared memory computing has several processors that have memory that is directly accessible by all of the processors. Such computers have the ability to synchronously execute the same or similar instructions on an array of data.

[0007] Parallel computing may be performed by separate computers that are connected over a network, sometimes known as cluster computers. The computers may communicate by passing messages between the computers. In such message passing parallel computing, the individual programs that are operating on the various computers must be synchronized at least in part so that the messages may be transmitted, acknowledged, and replied.

[0008] Other types of multi-processor computing systems are known in the art. However, each of the systems requires that the software is tailored to the specific system. For example, if a program were written for a shared memory computer, the same program would have to be rewritten to operate on a message passing computer system. In some cases, a program for a cluster computer of ten nodes may have to be changed in order to operate on a cluster computer of one thousand nodes.

[0009] It would be advantageous to provide a computing platform wherein the advantages of shared memory systems and message passing systems are both realized. It would be further advantageous if the system were automatically scalable to different computing architectures and thus programs could be developed and tested in a small computing environment prior to running on a large scale cluster computer or the like.

### SUMMARY OF THE INVENTION

[0010] The present invention overcomes the disadvantages and limitations of the prior art by providing a system and method for creating a parallel computer program that is mapped to the available computer architecture at run time.

Further, programming techniques of the shared memory architecture as well as those of the message passing architecture may be utilized where practical without regard to the topology of the hardware.

[0011] The present invention may therefore comprise a method of parallel processing on a computer array comprising the steps of: creating a virtual parallel processing computer system comprising a plurality of parallel processing units; creating programs for each of said parallel processing units wherein each of said programs performs a predetermined function on predetermined data; determining the topology of the physical hardware configuration on which said virtual parallel processing computer system will execute said programs, said topology having at least one computing node; allocating at least one of said parallel processing units to at least one of said computing nodes based upon a distribution algorithm; transferring said programs and said data to said computing nodes for each of said parallel processing units; and executing said programs for said parallel processing units on said computing nodes.

[0012] The present invention may further comprise a computer programming environment for parallel computing comprising: an editor adapted to create a virtual parallel processing computer system comprising a plurality of parallel processing units, said editor further adapted to create programs for each of said parallel processing units wherein each of said programs performs a predetermined function on predetermined data; a virtual computer loader routine adapted to determine the topology of the physical hardware configuration on which said virtual parallel processing computer system will execute said programs, said topology having at least one computing node, said loader routine further adapted to allocate at least one of said parallel processing units to at least one of said computing nodes based upon a distribution algorithm and transfer said programs and said data to said nodes for each of said parallel processing units; and a virtual computer execution routine adapted to execute said programs for said parallel processing units.

[0013] The present invention may further comprise a parallel processing computer system comprising: at least one computing node comprising at least one processor, said computing node connected to a network; an editing system adapted to create a virtual parallel processing computer system comprising a plurality of parallel processing units, said editor further adapted to create programs for each of said parallel processing units wherein each of said programs performs a predetermined function on predetermined data; and a system controller comprising a virtual computer loader routine adapted to determine the topology of the physical hardware configuration on which said virtual parallel processing computer will execute said programs, said topology comprising said at least one computing node, said loader routine further adapted to allocate at least one of said parallel processing units to at least one of said computing nodes based upon a distribution algorithm and transfer said programs and said data to said nodes for each of said parallel processing units, and a virtual computer execution routine adapted to execute at least two of said programs for said parallel processing units substantially simultaneously.

[0014] The advantages of the present invention are that programming tasks that are applicable to the techniques of

shared memory or message passing parallel processing may be combined in a single application. Further, the application may be created and run on any parallel processing architecture without reprogramming.

#### BRIEF DESCRIPTION OF THE DRAWINGS

[0015] In the drawings,

[0016] **FIG. 1** is an illustration of an embodiment of the present invention of a parallel processing computer system.

[0017] **FIG. 2** is an illustration of a workflow diagram of an embodiment of the present invention of an editor process for the creation of a virtual parallel processor computer.

[0018] **FIG. 3** is an illustration of an embodiment of the present invention of a virtual single process multiple data computer.

[0019] **FIG. 4** is an illustration of an embodiment of the present invention of a virtual parallel processing computer wherein multiple single process multiple data computers are connected.

[0020] **FIG. 5** is an illustration of an embodiment of the present invention of a load and execute process that takes a virtual parallel computer and loads the various processes to a physical hardware topology.

[0021] **FIG. 6** is an illustration of an embodiment of the present invention wherein virtual parallel processing units are mapped onto a physical hardware topology.

#### DETAILED DESCRIPTION OF THE INVENTION

[0022] **FIG. 1** illustrates an embodiment of the present invention of a parallel processing computer system **100**. An editor **102** is connected to a system controller **104** that is in turn connected to compute nodes **106**, **108**, and **110**.

[0023] The editor **102**, controller **104**, and nodes **106**, **108**, and **110** may be separate computers or a single computer. In some embodiments, especially those directed at solving large computational tasks, the number of computational nodes may be on the order of hundreds or even thousands of computers. The various computers may be connected by a standard network interface or may be connected through a high-speed proprietary interface. In some embodiments, the nodes may comprise multiple processors. In other embodiments, the editor **102**, controller **104**, and node **106** may be a single processor computer. With such an embodiment, the development of computer code may be performed without requiring a large cluster of computers for development purposes.

[0024] **FIG. 2** illustrates a workflow diagram of an embodiment **200** of an editor process for the creation of a virtual parallel processor computer. The process is begun in block **202** and a virtual single process multiple data (SPMD) computer is defined in block **204**. A program to be executed by the SPMD computer is defined in block **206**. The process may be repeated to define several SPMP computers. The interaction between SPMD computers may be defined in block **208**. When all SPMD computers are defined, the virtual parallel processor computer is prepared for execution in block **210**.

[0025] **FIG. 3** illustrates an embodiment **300** of a virtual single process multiple data (SPMD) computer. An array controller **302** containing shared memory is connected to parallel processing units (PPU) **304**, **306**, and **308**, each having local memory. Such processing schemes are suitable for the simultaneous calculations on large arrays of data among other tasks. In general, each PPU would run an identical program and operate on a different element of an array of data. For example, if an array of 1000 elements were to be analyzed, an embodiment of a virtual SPMD computer may contain one program operating on 1000 PPU's. Each PPU would have in its local memory a single element of the array.

[0026] Tasks that are adaptable and suitable for classical shared memory parallel processing computer systems may be developed for the virtual SPMD computers. In a conventional prior art shared memory computer, one parallel processing unit would correspond to exactly one processor of the multiprocessor computer.

[0027] **FIG. 4** illustrates an embodiment **400** of a virtual parallel processing computer wherein multiple single process multiple data computers are connected. A plurality of SPMD computers **402**, **404**, and **406** are connected to each other and a sequencer **408**. The sequencer **408** may have the ability to coordinate tasks between the various SPMD computers.

[0028] The connection of multiple virtual computers allows a programmer to use known programming techniques suitable to message passing parallel computers. In some cases, a programmer may elect to create multiple SPMD computers each running a single process and handle the synchronization between the various computers using similar programming techniques as message passing parallel computing.

[0029] **FIG. 5** illustrates an embodiment **500** of a load and execute process that takes a virtual parallel computer and loads the various processes to a physical hardware topology. The process is begun in block **502** and the virtual parallel computer file is read in block **504**. The actual hardware topology is determined in block **506**. A parallel processing unit is assigned to a physical node based on a decision maker in block **508**. If there are more unassigned PPU's, block **508** is repeated. When all of the PPU's are assigned to the various nodes, the execution of a process continues until a barrier statement is reached in block **512**. If there are unfinished or unexecuted processes in block **514**, the remaining processes are executed in block **512**. When the processes are complete to the barrier in block **514**, and the barrier statement is not the last barrier statement in block **516**, the processes are started again until the next barrier in block **518**. If the last barrier is reached in block **516**, the process is stopped in block **520**.

[0030] The hardware topology in block **506** may be determined by several different methods. In one embodiment, the hardware topology may be manually entered. In other embodiments, an automated program may analyze the available hardware to determine availability. In still other embodiments, the available hardware may be benchmarked to determine computational speed.

[0031] The embodiment **500** illustrates a method of loading the virtual parallel process computer onto a hardware

topology. Each of the many parallel processing units are assigned to be run on physical hardware based on the decision maker in block **508**. In some cases with high number of parallel processing units, many PPU's may be assigned to a single computer node and executed. In such a case, each PPU may be calculated until a barrier statement is reached, then another PPU may be calculated until the same barrier and so forth until all of the PPU's are brought up to the same stage of processing. In other embodiments containing a large number of nodes, a single PPU may be assigned to a single computational node and the overall process may execute more quickly.

[**0032**] The decision maker used to determine the assignment of a PPU to a hardware node in block **508** may have many different types of algorithms. In a simple example, the PPU's may be assigned to computational nodes in order until all of the PPU's are assigned to nodes. In other cases, PPU's with common data may be assigned to a single node to minimize performance degradation due to transferring data. In still other cases, the computational time requirements of a PPU may be estimated and the PPU's may be dispersed among the available computational nodes to maximize the utility of the nodes. Such estimations may be used with a benchmark program of the available hardware to maximize the efficiency of the entire computer cluster. Those skilled in the arts may develop complex algorithms for determining the optimum placement of PPU's among the available hardware nodes.

[**0033**] The processes are executed until a barrier statement is reached. A barrier statement is a coordination or synchronization step that is traversed when all of the parallel processes have been brought up to the same point. In some cases, a barrier statement may be for the processes within a specific virtual SPMD computer. In other cases, the barrier statement may be global and require that all processes to be brought to a specific point, regardless of the SPMD to which they belong.

[**0034**] **FIG. 6** illustrates an embodiment **600** of the present invention wherein virtual PPU processes are mapped onto a physical hardware topology. Nodes **602** and **604** are connected by a network **605** to a sequencer **606**. The node **602** has three CPU's **608**, **610**, and **612**. The CPU **612** has a stack of PPU's **614** assigned to it.

[**0035**] The CPU **612** may execute one PPU at a time until a barrier statement is reached. If more PPU's within the stack need processing in order to reach the barrier, they may be executed. When all of the PPU's have reached the barrier, the CPU **612** or the node **602** may indicate to the sequencer **606** that the barrier has been reached. When all of the PPU's have been executed to the barrier, the sequencer **606** may send an instruction to begin execution of the PPU's until the next barrier and so forth.

[**0036**] The foregoing description of the invention has been presented for purposes of illustration and description. It is not intended to be exhaustive or to limit the invention to the precise form disclosed, and other modifications and variations may be possible in light of the above teachings. The embodiment was chosen and described in order to best explain the principles of the invention and its practical application to thereby enable others skilled in the art to best utilize the invention in various embodiments and various modifications as are suited to the particular use contem-

plated. It is intended that the appended claims be construed to include other alternative embodiments of the invention except insofar as limited by the prior art.

What is claimed is:

1. A method of parallel processing on a computer array comprising the steps of:

creating a virtual parallel processing computer system comprising a plurality of parallel processing units;

creating programs for each of said parallel processing units wherein each of said programs performs a predetermined function on predetermined data;

determining the topology of the physical hardware configuration on which said virtual parallel processing computer system will execute said programs, said topology having at least one computing node;

allocating at least one of said parallel processing units to at least one of said computing nodes based upon a distribution algorithm;

transferring said programs and said data to said computing nodes for each of said parallel processing units; and

executing said programs for said parallel processing units on said computing nodes.

2. The method of claim 1 wherein said virtual parallel processing computer system comprises at least one single program multiple data virtual computer.

3. The method of claim 2 wherein said virtual parallel processing computer system contains a plurality of single program multiple data virtual computers.

4. The method of claim 1 wherein said distribution algorithm comprises estimating the amount of time required for each of said parallel processing units to process said programs.

5. The method of claim 1 wherein said programs include at least one barrier statement.

6. The method of claim 5 wherein all of said programs are executed until one of said barrier statements is reached before processing continues on any of said programs past said barrier statement.

7. A computer programming environment for parallel computing comprising:

an editor adapted to create a virtual parallel processing computer system comprising a plurality of parallel processing units, said editor further adapted to create programs for each of said parallel processing units wherein each of said programs performs a predetermined function on predetermined data;

a virtual computer loader routine adapted to determine the topology of the physical hardware configuration on which said virtual parallel processing computer system will execute said programs, said topology having at least one computing node, said loader routine further adapted to allocate at least one of said parallel processing units to at least one of said computing nodes based upon a distribution algorithm and transfer said programs and said data to said nodes for each of said parallel processing units; and

a virtual computer execution routine adapted to execute said programs for said parallel processing units.

8. The computer programming environment of claim 7 wherein said virtual parallel processing computer system comprises at least one single program multiple data virtual computer.

9. The computer programming environment of claim 8 wherein said virtual parallel processing computer system contains a plurality of single program multiple data virtual computers.

10. The computer programming environment of claim 7 wherein said distribution algorithm comprises estimating the amount of time required for each of said parallel processing units to process said programs.

11. The computer programming environment of claim 7 wherein said programs include at least one barrier statement.

12. The computer programming environment of claim 11 wherein all of said programs are executed until one of said barrier statements is reached before processing continues on any of said programs past said barrier statement.

13. A parallel processing computer system comprising:

at least one computing node comprising at least one processor, said computing node connected to a network;

an editing system adapted to create a virtual parallel processing computer system comprising a plurality of parallel processing units, said editor further adapted to create programs for each of said parallel processing units wherein each of said programs performs a predetermined function on predetermined data; and

a system controller computer comprising a virtual computer loader routine adapted to determine the topology of the physical hardware configuration on which said virtual parallel processing computer will execute said programs, said topology comprising said at least one computing node, said loader routine further adapted to

allocate at least one of said parallel processing units to at least one of said computing nodes based upon a distribution algorithm and transfer said programs and said data to said nodes for each of said parallel processing units, and a virtual computer execution routine adapted to execute at least two of said programs for said parallel processing units substantially simultaneously.

14. The parallel processing computer system of claim 13 wherein said compute node, said editing system, and said system controller computer are the same computer.

15. The parallel processing computer system of claim 13 further comprising a plurality of compute nodes.

16. The parallel processing computer system of claim 15 wherein at least one of said compute nodes comprises a plurality of processors.

17. The parallel processing computer system of claim 13 wherein said virtual parallel processing computer system comprises at least one single program multiple data virtual computer.

18. The parallel processing computer system of claim 17 wherein said virtual parallel processing computer system contains a plurality of single program multiple data virtual computers.

19. The parallel processing computer system of claim 13 wherein said distribution algorithm comprises estimating the amount of time required for each of said parallel processing units to process said programs.

20. The parallel processing computer system of claim 13 wherein said programs include at least one barrier statement.

21. The parallel processing computer system of claim 20 wherein all of said programs are executed until one of said barrier statements is reached before processing continues on any of said programs past said barrier statement.

\* \* \* \* \*