

FIG. 9 is a flow chart illustrating operations involved in a write operation to a memory element by speculative thread **203** in accordance with an embodiment of the present invention. If a space-time dimensioned version **510** does not exist, the system creates a space-time dimensioned version **510** in speculative heap **406** (step **902**). The system also updates status word **504** to indicate that speculative thread **203** has written to the object if such updating is necessary (step **903**). The system next writes to space-time dimensioned version **510** (step **904**). Such updating is necessary if head thread **202** must subsequently choose between writing to both primary version **500** and space-time dimensioned version **510** or writing only to primary version **500** as is described above with reference to FIG. 7.

FIG. 10 is a flow chart illustrating operations involved in a join operation between head thread **202** and a speculative thread **203** in accordance with an embodiment of the present invention. A join operation occurs for example when head thread **202** reaches a point in the program where speculative thread **203** began executing. The join operation causes state associated with the speculative thread **203** to be merged with state associated with the head thread **202**. This involves copying and/or merging the stack of speculative thread **203** into the stack of head thread **202** (step **1002**). It also involves merging space-time dimension and primary versions of objects (step **1004**) as well as possibly garbage collecting speculative heap **406** (step **1006**). In one embodiment of the present invention, one of threads **202** or **203** performs steps **1002** and **1006**, while the other thread performs step **1004**.

FIG. 11 is a flow chart illustrating operations involved in a join operation between head thread **202** and a speculative thread **203** in accordance with another embodiment of the present invention. In this embodiment, speculative thread **203** carries on as a pseudo-head thread. As a pseudo-head thread, speculative thread **203** uses indirection to reference space-time dimensioned versions of objects, but does not mark objects or create versions. While speculative thread **203** is acting as a pseudo-head thread, head thread **202** updates primary versions of objects.

Extension to Additional Speculative Threads

Although the present invention has been described for the case of a single speculative thread, the present invention can be extended to provide multiple speculative threads operating on multiple space-time dimensioned versions of a data object in parallel.

Process of Setting Marking Bits

FIG. 12 is a flow chart illustrating the process of setting a marking bit associated with a referenced field within an object in accordance with an embodiment of the present invention. First, the system receives a reference to the field within the object (step **1202**). This reference may be a read or a write operation. (Note that in the case of an array object the field is an array element.)

Next, the system identifies a marking bit associated with the field (step **1204**). In one embodiment of the present invention, the system maintains a separate set of read marking bits **604** for the object to indicate that a read operation has occurred to the field, and a separate set of write marking bits **606** to indicate that a write operation has occurred to the field. In this embodiment, if the operation is a read operation, one of the read marking bits **604** is selected. Otherwise, one of the write marking bits **606** is selected.

In one embodiment of the present invention, the marking bit is identified by performing modulo operation. For example, if the object includes N marking bits numbered **0**, **1**, **2**, . . . , N-1 and M fields numbered **0**, **1**, **2**, . . . , M-1,

the system starts with a field number for the field, and applies a modulo N operation to the field number to produce a number for the associated marking bit. This modulo operation can be efficiently performed if N is a power of two because the modulo operation reduces to a simple bit mask operation that isolates the lower order bits of the field number.

The selection a value of N for an object involves a tradeoff. If N is too small, there tends to be lot of aliasing and system performance suffers due to false rollbacks. If N is too large, a great deal of memory is used for marking bits which can cause cache performance to suffer. In one embodiment of the present invention, N=8. In another embodiment, N=16.

In the case of an array object, the system applies a division operation to the array element number (field number) to identify the associated array element. For example, if the array object has N marking bits numbered **0**, **1**, **2**, . . . , N-1 and M array elements numbered **0**, **1**, **2**, . . . , M-1, the step of identifying the marking bit includes dividing the array element number by the ceiling of M/N to produce a number for the associated marking bit. If the ceiling of M/N is a power of two, the division operation can be accomplished by shifting the array index so that the most significant bits of the array index become the number for the associated marking bits.

The above-described mapping between array elements and marking bits for array objects associates consecutive array locations with a single marking bit. This ensures that not all of the marking bits are set by common block copy operations involving only a portion of the array.

After the marking bit is identified, the marking bit is set (step **1206**) and the reference is performed to the field (or array element) within the object (step **1208**).

In general the marking mechanism according to the present invention can be used in any application that must keep track of accesses to fields within an object. However, in one embodiment of the present invention, marking is performed for read operations by speculative thread **203**. In another embodiment, marking is performed to write bits **606** during a write operation by speculative thread **203** and to read bits **604** during a read operation by speculative thread **203**.

After the marking bits have been set, if a head thread **202** subsequently performs a write operation to a field in the object, head thread **202** can identify the associated marking bit using the above-described modulo or division operations. Next, the marking bit is extracted for examination purposes using a special bit extract operation that is part of the instruction set of the underlying computer system.

FIG. 13 illustrates how a marking bit number can be determined from a field number or an array element number in accordance with an embodiment of the present invention. The system starts with a field number or an array element number **1302**. In the case of a field number, the system performs a modulo operation by masking off all but the lower order three bits of field number **1302** to produce a three bit index (1,0,0) that specifies a marking bit. In the case of an array index, the system performs a division operation by shifting array element number **1302** until only the three most significant bits (0,1,1) remain.

FIG. 14 illustrates how a block transfer operation sets multiple marking bits in accordance with an embodiment of the present invention. The example illustrated in FIG. 14 includes an array of data elements **1402**. These data elements are numbered **0**, **1**, **2**, . . . , **31**. FIG. 14 also includes an array of read bits **604**. These read bits are numbered **0**, **1**,