Atomic Construct (11A)

- Task
-

Copyright (c) 2024 Young W. Lim.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

Please send corrections (or suggestions) to youngwlim@hotmail.com.

This document was produced by using OpenOffice and Octave.

Atomic operations

Use OpenMP atomic operations
to allow multiple threads to safely update a shared numeric variable,
such as on hardware platforms that support atomic operation use.

An atomic operation applies
only to the single assignment statement
that immediately follows it,
so atomic operations are useful
for code that requires fine-grain synchronization.

https://www.intel.com/content/www/us/en/docs/advisor/user-guide/2024-1/basic-openmp-atomic-operations.html

Atomic operation examples (1)

For example, consider this annotated C/C++ serial code:

```
  int count;
   void Tick() {
     ANNOTATE_LOCK_ACQUIRE(0);
        count = count+1;
     ANNOTATE_LOCK_RELEASE(0);
  }
...
```

https://www.intel.com/content/www/us/en/docs/advisor/user-guide/2024-1/basic-openmp-atomic-operations.html

Atomic operation examples (2)

The parallel C/C++ code after
adding #include <omp.h>
and #pragma omp atomic:

```
#include <omp.h>  //prevents a load-time problem with a .dll not being found
  int count;
   void Tick() {
   // Replace lock annotations
     #pragma omp atomic
        count = count+1;
  }
...
```

https://www.intel.com/content/www/us/en/docs/advisor/user-guide/2024-1/basic-openmp-atomic-operations.html

Atomic operation examples (3)

Consider this annotated Fortran serial code:

```
program ABC
   integer(kind=4) :: count = 0
 . . .
contains
subroutine Tick
  call annotate_lock_acquire(0)
    count = count + 1
  call annotate_lock_release(0)
end subroutine Tick
 . . .
end program ABC
```

https://www.intel.com/content/www/us/en/docs/advisor/user-guide/2024-1/basic-openmp-atomic-operations.html

Atomic operation examples (4)

The parallel Fortran code after adding use omp_lib and the !$omp atomic directive:

```
program ABC
  use omp_lib
  integer(kind=4) :: count = 0
  . . .
  contains
   subroutine Tick
      !$omp atomic
       count = count + 1
   end subroutine Tick
 . . .
end program ABC
```

https://www.intel.com/content/www/us/en/docs/advisor/user-guide/2024-1/basic-openmp-atomic-operations.html

Atomic operation examples (5)

Apart from using critical construct
to synchronize accessing the same variable,
atomic is another choice.

```
#include <stdio.h>
#include <omp.h>

int main(void)
{
    int sum = 0;

    #pragma omp parallel for
    for (int index = 1; index <= 10; index++)
    {
        #pragma omp atomic
        sum += index;
    }

    printf("Sum is %d\n", sum);
    return 0;
}
```

https://nanxiao.gitbooks.io/openmp-little-book/content/posts/atomic-construct.html

Atomic operation examples (6)

the atomic ensures updating sum variable is an atomic operation,
so the program always calculate the correct result.
Besides +, atomic construct also supports
other operators, such as: -, *, /, etc.
Similarly, -=, *=, /= are covered too.

```
# gcc -fopenmp parallel.c
# ./a.out
Sum is 55
# ./a.out
Sum is 55
# ./a.out
Sum is 55
# ./a.out
Sum is 55
```

https://nanxiao.gitbooks.io/openmp-little-book/content/posts/atomic-construct.html

Atomic operation examples (6)

Synchronization: atomic

• atomic provides mutual exclusion

but only applies to the load / update of a memory location.

• This is a lightweight, special form of a critical section.

• It is applied only to the (single) assignment statement that immediately follows a atomic construct

```
{
  …
  #pragma omp parallel
  {
    double tmp, B;
  ….
    #pragma omp atomic
    {
      X+=tmp;
    }
  }
}
```

atomic only protects the update of X

https://nanxiao.gitbooks.io/openmp-little-book/content/posts/atomic-construct.html

Atomic operation examples (6)

```
int ic, i, n;
Ic = 0;

#pragma omp parallel shared(n,ic) private(i)
  for (i=0; i++, i<n)
    {
      #pragma omp atomic
      ic = ic + 1;
    }
```

“ic” is a counter.

The atomic construct ensures
that no updates are lost
when multiple threads are updating a counter value.

https://nanxiao.gitbooks.io/openmp-little-book/content/posts/atomic-construct.html

Atomic operation examples (6)

```
#pragma omp atomic

#pragma omp atomic read|write|update|capture
```

"If the low-level, high performance constructs
for mutual exclusion exist on this hardware,
use them.

Otherwise act like this is a critical section."

Is there any benefit to critical sections in this case?

Perhaps critical sections allow for function calls,
where atomic only refers to a scalar set operation?

just available for simple binary operations to update values.

https://dev.to/winstonpuckett/openmp-notes-1cfa

Atomic operation examples (6)

If, as in the example above, our critical section is a single assignment, OpenMP provides a potentially more efficient way of protecting this.

OpenMP provides an atomic directive which, like critical, specifies the next statement must be done by one thread at a time:

```
#pragma omp atomic
global_data++;
```

Unlike a critical directive:

The statement under the directive can only be a single C assignment statement.

It can be of the form: x++, ++x, x-- or --x.

It can also be of the form x OP= expression where OP is some binary operator.

No other statement is allowed.

https://ianfinlayson.net/class/cpsc425/notes/13-openmp-sync

The motivation for the atomic directive is that some processors provide single instructions for operations such as x++. These are called Fetch-and-add instructions.

References

[1] en.wikipedia.org
[2] M Harris, http://beowulf.lcs.mit.edu/18.337-2008/lectslides/scan.pdf