Reduction Clause (5A)

- Loop
-

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

This document was produced by using OpenOffice and Octave.

Clauses (4)

reduction (operator: list)

Performs a **reduction** on all **scalar variables** in list
using the specified **operator**.
Reduction variables in list are separated by commas.

A **private copy** of each variable in list is created for each thread.

At the end of the statement block,
the **final** values of all **private copies** of the reduction variable
are combined in a manner appropriate to the **operator**,
and the **result** is placed back in the **original value**
of the **shared reduction variable**.

https://www.ibm.com/docs/en/xl-c-aix/13.1.2?topic=processing-pragma-omp-section-pragma-omp-sections

Reduction (1)

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

int main(void)
{
    int sum = 100;

    #pragma omp parallel for
    for (int i = 1; i <= 4; i++)
    {
        sum += i;
    }

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

https://nanxiao.gitbooks.io/openmp-little-book/content/posts/collapse-clause.html?q=

Reduction (2)

Because sum is a shared variable in threads, so we need to use synchronization to protect accessing it:

```
#pragma omp parallel for
for (int i = 1; i <= 4; i++)
{
   #pragma omp critical
   sum += i;
}
```

But this will cause losing the advantage of using parallelism. The other method is using reduction clause:

reduction(reduction-identifier : list)

https://nanxiao.gitbooks.io/openmp-little-book/content/posts/collapse-clause.html?q=

Reduction (3)

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

int main(void)
{
    int sum = 100;

    printf("Before parallelism, sum's address is %p\n", &sum);

    #pragma omp parallel for reduction(+ : sum)
    for (int i = 1; i <= 4; i++)
    {
        printf("sum's address in thread %d is %p, value is %d\n", omp_get_thread_num(), &sum, sum);
        sum += i;
    }

    printf("After parallelism, sum's address is %p, and value is %d\n", &sum, sum);
    return 0;
}
```

https://nanxiao.gitbooks.io/openmp-little-book/content/posts/collapse-clause.html?q=

Reduction (4)

```
# gcc -fopenmp parallel.c
# ./a.out
Before parallelism, sum's address is 0x7ffcc880baf0
sum's address in thread 3 is 0x7f6baea5ee20, value is 0
sum's address in thread 2 is 0x7f6baf25fe20, value is 0
sum's address in thread 0 is 0x7ffcc880ba90, value is 0
sum's address in thread 1 is 0x7f6bafa60e20, value is 0
After parallelism, sum's address is 0x7ffcc880baf0, and value is 110
```

https://nanxiao.gitbooks.io/openmp-little-book/content/posts/collapse-clause.html?q=

Clauses (5)

reduction (operator: list)

For example, when the **max operator** is specified,
the original reduction variable value combines
with the final **values** of the **private copies**
by using the following expression:

```
original_reduction_variable =
    original_reduction_variable  < private_copy
                                 ? private_copy
                                 : original_reduction_variable;
```

https://www.ibm.com/docs/en/xl-c-aix/13.1.2?topic=processing-pragma-omp-section-pragma-omp-sections

References

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