

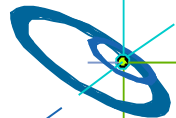


An OpenMP Validation Suite

EWOMP 2003

Matthias Müller, Pavel Neytchev
mueller@hls.de

HLRS
High Performance Computing Center Stuttgart
Allmandring 30
D-70550 Stuttgart
<http://www.hls.de>



09.10.2003

Matthias Müller
Höchstleistungsrechenzentrum Stuttgart



Motivation

Matthias Müller

09.10.2003

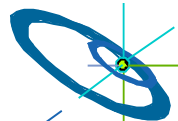
Höchstleistungsrechenzentrum Stuttgart

H L R I S 



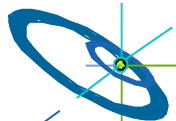
Current status of OpenMP

- OpenMP 1.0
- OpenMP 2.0
 - 106 pages (C/C++) released March 2002
 - 124 pages Fortran, released November 2000
- C/C++:
 - 10 different constructs
 - 2 directives
 - 11 clauses
 - 22 runtime library calls



Do we need a validation suite?

- No:
 - OpenMP 2.0 is mature
 - OpenMP 1.x is rock solid
 - The compiler people already have them in house
- Yes:
 - Many open source projects could make use of it
 - It should be independent from the compiler
 - It can also be used to simply check whether 2.0 is supported
 - Testing is always good
 - OpenMP is large and complex enough to make it difficult to implement it efficiently

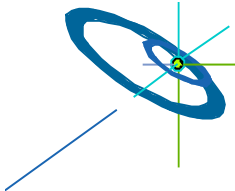


OpenMP directives and valid clauses

parallel	if, private, firstprivate, default, shared, copyin, reduction, num_threads
for	private, firstprivate, lastprivate, reduction, ordered, schedule, nowait
sections	private, firstprivate, lastprivate, reduction, nowait
single	private, firstprivate, copyprivate, nowait
parallel for	if, private, firstprivate, default, shared, copyin, reduction, num_threads, reduction, ordered, schedule
parallel sections	if, private, firstprivate, default, shared, copyin, reduction, num_threads

6+42 combinations

Design



09.10.2003

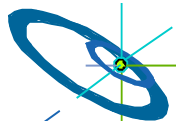
Höchstleistungsrechenzentrum Stuttgart

Matthias Müller

H L R I S 

Design - Principles

- User approach: validate OpenMP by using different constructs to perform a number of calculations, verify the results
- Portability:
 - Only test properties guaranteed by the standard
 - Stay independent of a single compiler
- Proof the efficiency of the validation suite:
 - By finding compiler bugs
 - With a coverage analysis



Design - Main Loop

- Each subroutine `check_foo()` validates a construct `foo`
- `crosscheck_foo()` verifies that the check is effective
- The tests are repeated `N` times, to allow race conditions to occur

```
for (i=0; i<N; i++){  
    if (!check_foo())  
        failed=1;  
    if (!crosscheck_foo())  
        nf++;  
}
```

Example for simple test

```
int check_omp_reduction(){
    int i;
    int sum=0;
    int known_sum=N*(N-1)/2;
#pragma omp parallel for reduction(+:sum)
    for(i=1; i<N ; i++){
        sum=sum+i;
    }
    return (sum==known_sum);
}
```

Example for simple test: cross check

```
int check_omp_reduction(){
    int i;
    int sum=0;
    int known_sum=N*(N-1)/2;
#pragma omp parallel for
    for(i=1; i<N ; i++){
        sum=sum+i;
    }
    return (sum==known_sum);
}
```

OpenMP directives and clauses and their substitutes

Clause	Substitute
if	
private	shared
firstprivate	private
lastprivate	private
ordered	
single	
copyprivate	private
reduction	
num_threads	

Probability to pass the test without a correct implementation

- Total number of tests: M
Number of failed cross tests: f
- Estimated probability of the real test to fail:

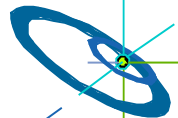
$$p=f/M$$

- Probability to pass the test by accident:

$$p_a=(1-p)^M$$

- Certainty of the test:

$$p_c=1-p_a$$



Tests for OpenMP 2.0

- num_threads clause: verify the number of threads
- copyprivate: check broadcast of private variable modified inside single directive
- omp_get_wtime: compare with value returned by gettimeofday
- omp_get_wtick: existence, value is positive



Results

Matthias Müller

09.10.2003

Höchstleistungsrechenzentrum Stuttgart

H L R I S 

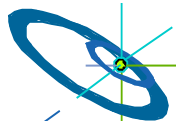


Result of coverage analysis

File	gcc	OpenMP	gcc+OpenMP
C-compile-decl.c	72%	45%	72%
C-compile-expr.c	83%	34%	83%
C-expr-mem.c	96%	81%	96%
C-lex.c	73%	46%	74%
C-main.c	62%	41%	62%
C-mem.c	100%	84%	100%
C-omp-pragma.c	0%	56%	56%
C-opt-expv.c	0%	0%	0%
C-pragma.c	5%	24%	24%
C-tea-pragma.c	0%	0%	0%
X-decompile.c	0%	0%	0%
X-io.c	32%	28%	32%
machine-dep.c	89%	89%	89%

Reason for uncovered areas

- Compiler specific:
 - Unused compiler flags
 - Internal error handling
 - Unused language extensions
 - Syntax errors in OpenMP pragmas
- Real issues:
 - **default** clause (shared, none)
 - Different schedule clauses (**guided, runtime**)
 - **nowait** clause
 - Reduction operators (*,-,&^,|,&&,||)



Results

Verified with 100%

100

Not verified

0

Verified with races

60

Failed

-1

Platform	1	2	3	4	5	6	7	8	9	10
check has openmp	100	100	100	100	100	100	100	100	100	100
check omp get num threads	100	100	100	100	100	100	100	100	100	100
check omp in parallel	100	100	100	100	100	100	100	100	100	100
for ORDERED	100	100	100	100	100	100	100	-1	100	100
for REDUCTION	100	0	100	100	100	100	100	100	100	100
for PRIVATE	100	0	100	100	100	100	-1	100	0	100
for FIRSTPRIVATE	0	100	100	100	40	100	0	100	0	100
for LASTPRIVATE	100	100	60	20	0	0	100	40	100	20
section REDUCTION	0	0	100	100	100	100	100	100	100	100
section PRIVATE	80	0	100	100	100	100	0	100	100	100
section FIRSTPRIVATE	100	100	100	100	100	100	100	100	100	100
section LASTPRIVATE	100	100	100	100	100	100	100	100	100	100
SINGLE	100	100	100	100	100	100	100	100	100	100
single PRIVATE	0	60	60	0	0	0	60	0	0	0
SINGLE NOWAIT	100	100	100	100	100	100	100	100	100	100
parallel for ORDERED	100	100	100	100	100	100	100	-1	100	100
parallel for REDUCTION	0	100	100	100	0	100	100	0	100	100
parallel for PRIVATE	0	0	0	0	0	0	0	0	0	0
parallel for FIRSTPRIVATE	100	100	100	100	100	100	100	100	100	100
parallel for LASTPRIVATE	100	100	100	100	100	100	100	100	100	100
parallel section REDUCTION	20	100	100	100	100	100	100	100	100	100
parallel section PRIVATE	0	100	100	100	100	100	0	100	100	100
parallel section FIRSTPRIVATE	100	100	100	100	100	100	100	100	100	100
parallel section LASTPRIVATE	100	100	100	100	100	100	100	100	100	100
omp MASTER	100	100	100	100	100	100	100	100	100	100
omp CRITICAL	0	100	100	100	100	100	60	100	100	100
omp ATOMIC	0	100	100	100	100	100	-1	100	100	100
omp BARRIER	100	100	100	100	100	100	100	100	100	100
omp FLUSH	0	0	0	0	0	0	0	0	0	-1
omp THREADPRIVATE	100	100	100	-1	100	100	100	-1	-1	-1
omp COPYIN	100	100	100	100	80	100	100	100	-1	100
omp LOCK	100	100	100	100	100	100	-1	100	100	100
omp TESTLOCK	100	100	100	100	100	100	-1	100	100	100
omp NEST LOCK	100	100	100	100	100	100	-1	100	100	100
omp NEST TESTLOCK	100	100	100	100	100	100	-1	100	100	100
RESULT	0	0	0	-1	0	0	-1	-1	-1	-1

Results

Platform	1	2	3	4	5	6	7	8	9	10
for ORDERED	100	100	100	100	100	100	100	-1	100	100
for REDUCTION	100	0	100	100	100	100	100	100	100	100
for PRIVATE	100	0	100	100	100	100	-1	100	0	100
for FIRSTPRIVATE	0	100	100	100	40	100	0	100	0	100
for LASTPRIVATE	100	100	60	20	0	0	100	40	100	20
section REDUCTION	0	0	100	100	100	100	100	100	100	100
section PRIVATE	0	0	100	100	100	100	0	100	100	100

With one exception all tests only fail with one compiler

Repetitions are important to allow race conditions to

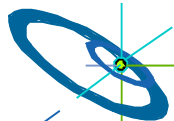
5 out of 10 compilers pass all tests

Difficult to verify without independent synchronization constructs

omp CRITICAL	100	100
omp ATOMIC	100	100
omp FLUSH	0	-1
omp THREADPRIVATE	-1	-1
omp COPYIN	100	100
omp LOCK	100	100
omp TESTLOCK	100	100
omp NEST LOCK	100	100
omp NEST TESTLOCK	100	100

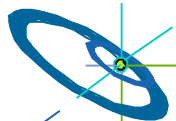
RESULT

-1



Summary and Outlook

- The suite currently contains
 - 35 tests for OpenMP 1.x
 - 4 tests for OpenMP 2.0
- The coverage analysis with the Omni OpenMP compiler showed that it tests a substantial portion of the OpenMP relevant part of the compiler
- The suite showed that it is already capable to discover bugs
- Future work:
 - Fortran version?
 - Extension of the suite
 - Integration into external framework (compiler options, # threads)



Results + Conclusion

- Results:
 - 5 out of 10 compilers fail the test
 - 22 out of 35 cross tests show different behavior with different compilers
- Conclusion:
 - Either OpenMP compilers are not mature enough
 - or OpenMP is
 - **not portable**
 - **too difficult to use**

