CCRG OpenMP : Experiments and Improvements

Huang Chun & Yang Xuejun

National Laboratory for Parallel and Distributed Processing, P.R. China

Outline

- Motivations
- CCRG OpenMP Compiler
- Optimized STATIC Schedule Implementation
- Inter-Procedural Optimization
- Conclusion and Future Work

Motivations

- Provide an open source OpenMP compiler infrastructure
  - Portable
  - Productive
  - Robust
- Provide a platform for building
  - Performance analysis system and debug tool for OpenMP applications
  - Static analyzer to help user to correct OpenMP applications

COMP
CCRG Open MP

Main Contributions

- CCRG OpenMP Fortran95 Compiler
- Performance evaluation & analysis
- Source-level optimization
  - Static schedule implementation
  - Inter-procedural optimization

CCRG OpenMP Compiler

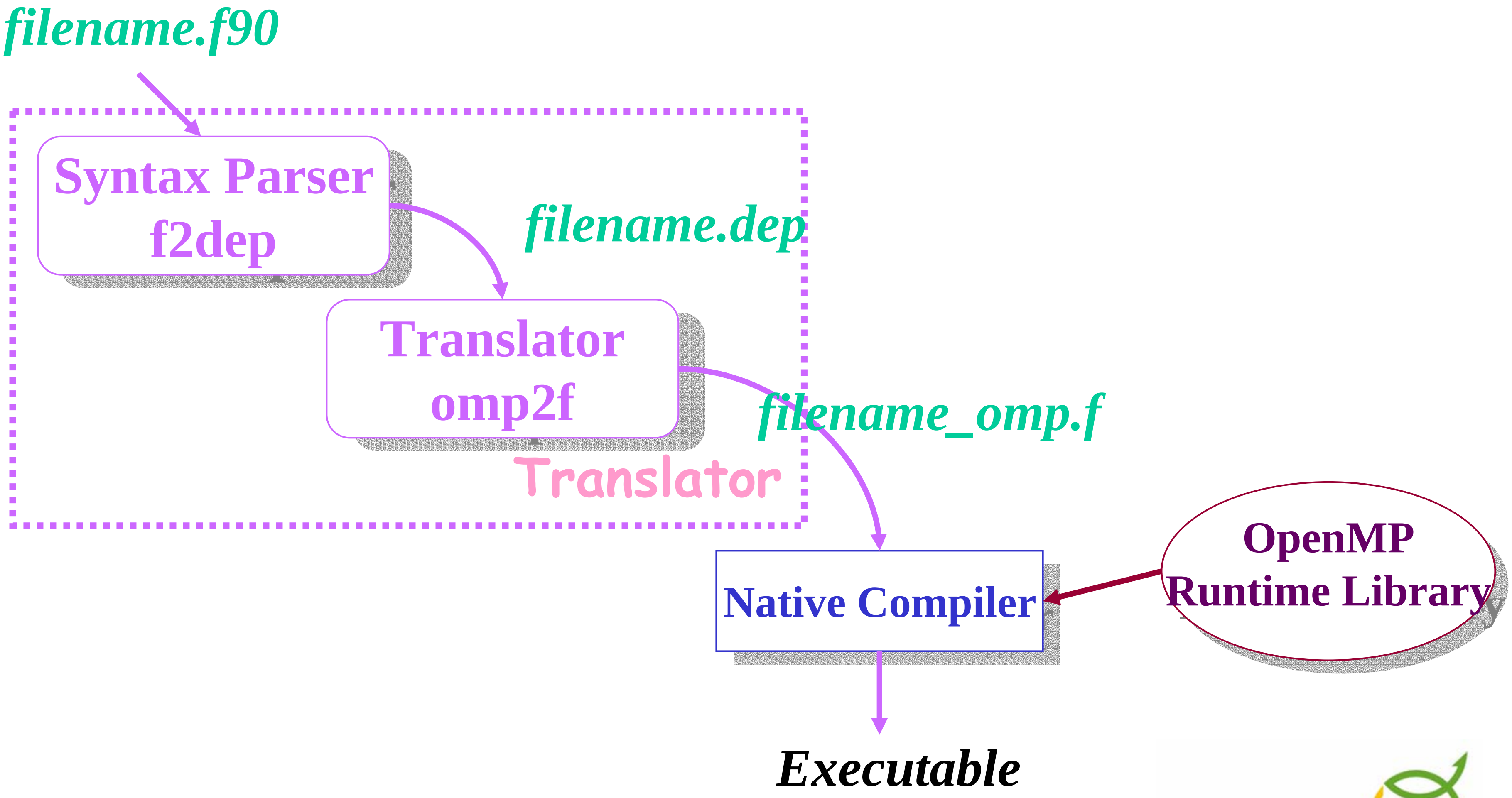
- Source-to-Source Compiler
- CCRG OpenMP Compiler is consist of
  - Translator
    - Transform OpenMP program to equivalent Fortran code
  - Runtime library
  - Native Compiler
    - GNU GCC
    - Commercial Compilers, such as Intel Compiler

- Features
  - Support Fortran95 languages
  - Use **ENTRY** statement to reduce the size of the code
  - Portable implementation of OpenMP for SMPs and SDSM

COMP

CCRG Open MP

CCRG OpenMP Compiler



COMP
CCRG Open MP

Translator

- Based on Sage++*
  - Fortran OpenMP syntax parser – f2dep
    - Add syntax description for OpenMP directives

```
omp_directive:
    omp_parallel
  | omp_paralleldo
  | omp_parallelworkshare
  |  ……;
 omp_parallel:
    PARALLEL end_spec needkeyword omp_clause_opt keywordoff
    {
      omp_binding_rules (OMP_PARALLEL_NODE);
      $$ = get_bfnd (fi, OMP_PARALLEL_NODE, SMNULL,
                            $4, LLNULL, LLNULL);
    }
```

  - Translator – omp2f

* See www.extreme.indiana.edu/sage for more information

Translator

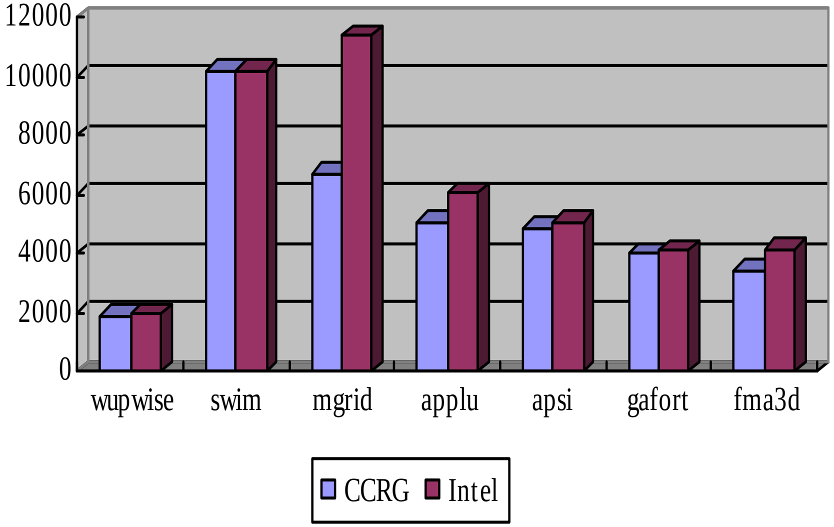
```
SUBROUTINE test()
  DIMENSION a(100)
!$OMP PARALLEL DO NUM_THREADS(4)
  DO 100 k = 1, 100
100 a(k) = 0.9
!$OMP PARALLEL …
!$OMP END PARALLEL
END
```

number of dummy
of test_$1 generated
by the translator

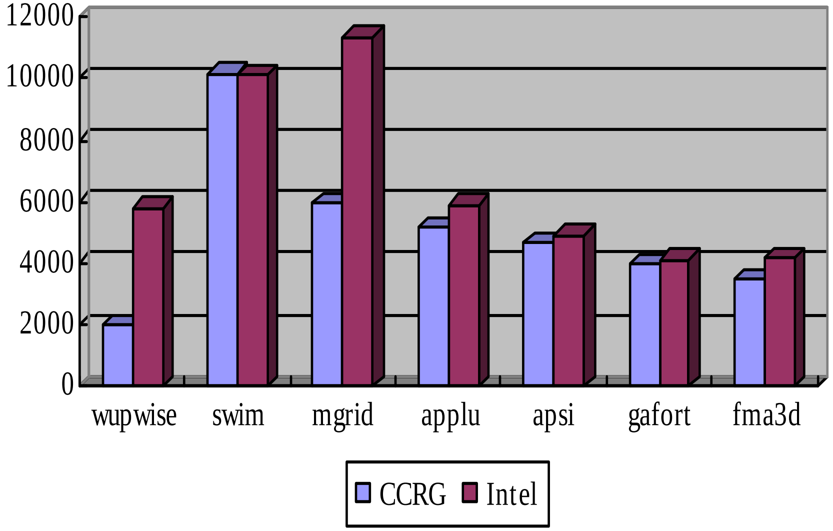
```
SUBROUTINE test()
  DIMENSION a(100)
  EXTERNAL test_$1, test_$2
  CALL comp_runtime_init()
  CALL comp_parallel(test_$1,4, 1, a)
  CALL comp_parallel(test_$2,……)
  CALL comp_exit()
END
```

```
      SUBROUTINE test_$0(a)
        DIMENSION a(100)
        INTEGER lc_k
        INTEGER _omp_dolo, _omp_dohi, comp_static_more
        ENTRY test_$1(a)
          CALL comp_static_setdo (1, 100, 1, 0)
          DO WHILE (comp_static_more(_omp_dolo,
     &                    _omp_dohi, 1).eq.1)
          DO 100 lc_k =_omp_dolo,_omp_dohi, 1
100         a(lc_k) = 0.9
          END DO
          CALL comp_barrier()
        RETURN
        ENTRY test_$2(a)
          ………
        RETURN
     END
```

Performance Results



Base Ratios of CCRG and Intel without IPO*



Base Ratios of CCRG and Intel without IPO*

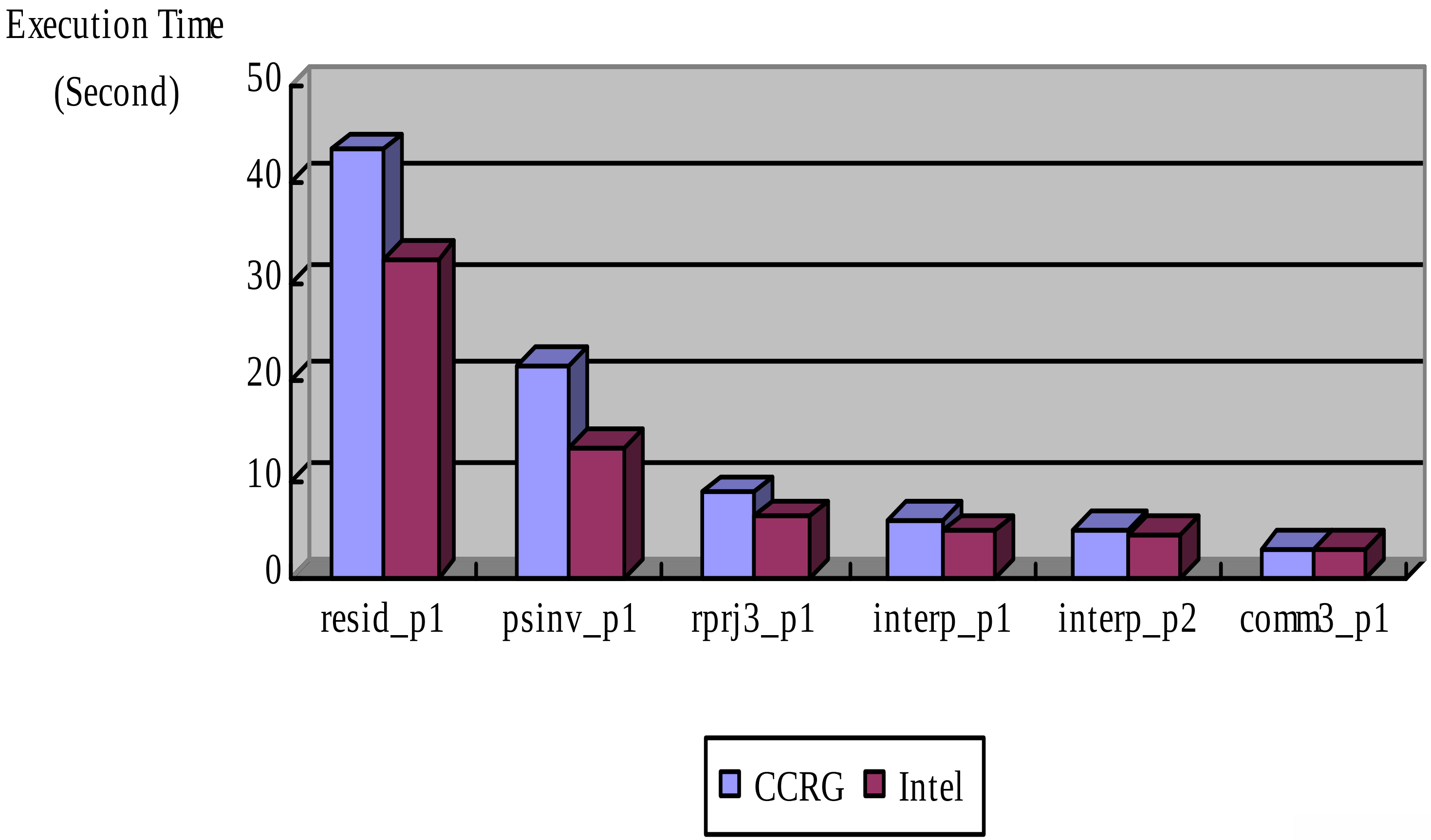
Intel Fortran Compiler 8.0 is used as the naïve compile of CCRG OpenMP Compiler
* “-ipo” option enables inter-procedural optimization(IPO) across files.

COMP
CCRG Open MP

Performance Results

- Most of SPEC OMP Fortran Programs show good performance as Intel Compiler
- Why **mgrid** & **wupwise** perform poorly?
  - mgrid
  - wupwise

Profile of mgrid



- “-O3”
- “TRAIN” input set is used

Implementation of DO Directive

```
        CALL comp_type_setdo (lo, hi, in, chunk)
         DO WHILE (comp_type_more(_omp_dolo,   _omp_dohi, in) .eq. 1)
         DO 100 lc_k =_omp_dolo,_omp_dohi, in
100          a(lc_k) = 0.9
         END DO
```

- DO WHILE loop is introduced to implement the schedule clause of OpenMP
- All of the schedule types of OpenMP use the same approach
- *type* is one of the following:
  - static
  - dynamic
  - guided
  - runtime

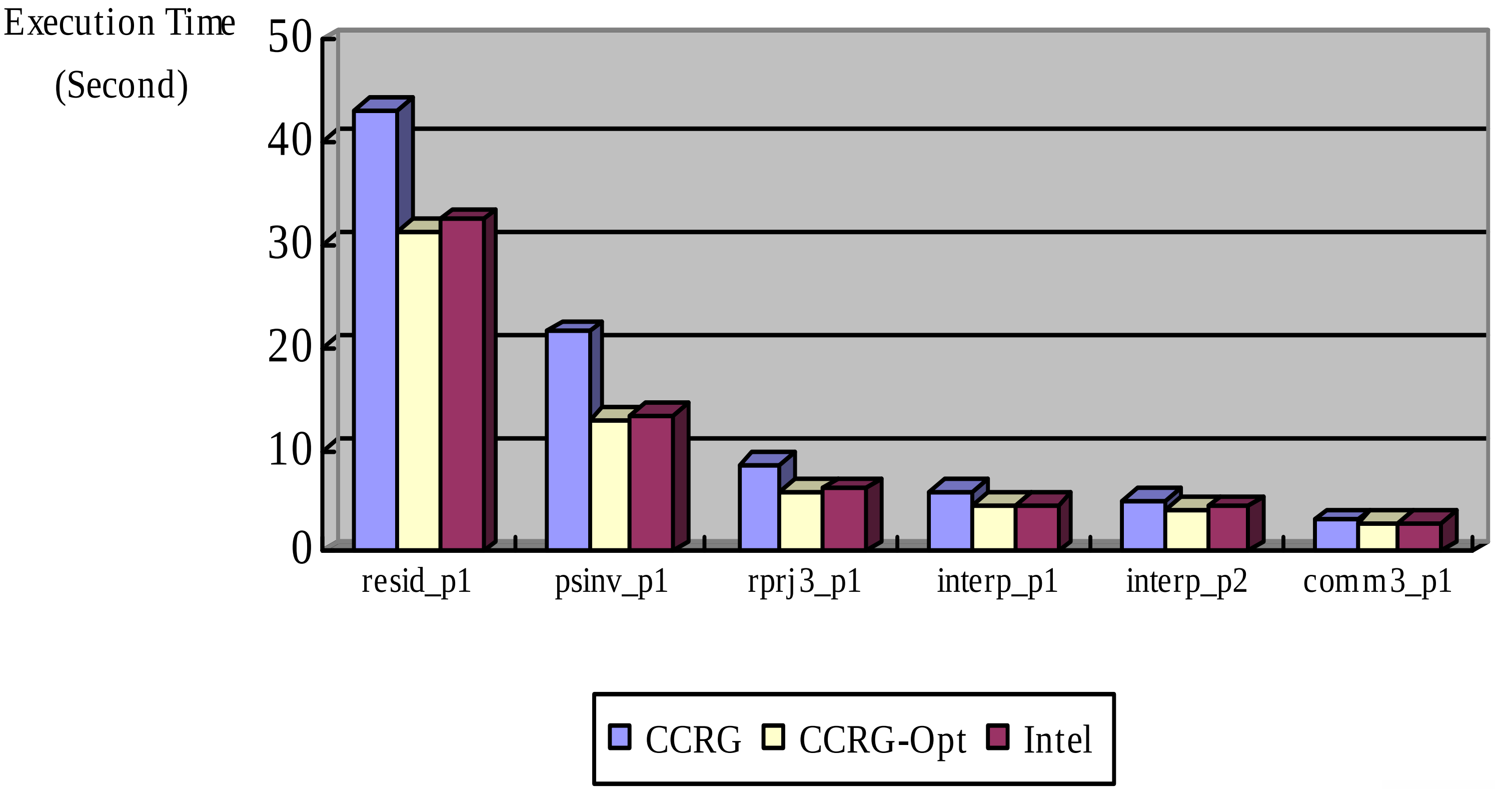
Optimized STATIC Schedule

- `DO WHILE` loop can be omitted if
  - No `SCHEDULE` clause.
  - `SCHEDULE(STATIC)` is specified.
  - Static schedule
    - Both chunk size, number of iteration and number of threads are known during compile time
    - (chunk size × number of threads) ≤ number of iteration.

```
    CALL comp_static_setdo(1,100,1,0)
    CALL comp_static_once(_omp_dolo, _omp_dohi, 1)
    DO 100 lc k = _omp_dolo, _omp_dohi, 1
100   a(lc_k) = 0.9
```

- `SCHEDULE` clause is not specified in most of OpenMP programs

Profile of mgrid after Optimization



wupwise

- Inter-Procedural Optimization (IPO)
- Source-to-source transformation **can not** keep the information about the **caller-callee** relationship between the original procedures.

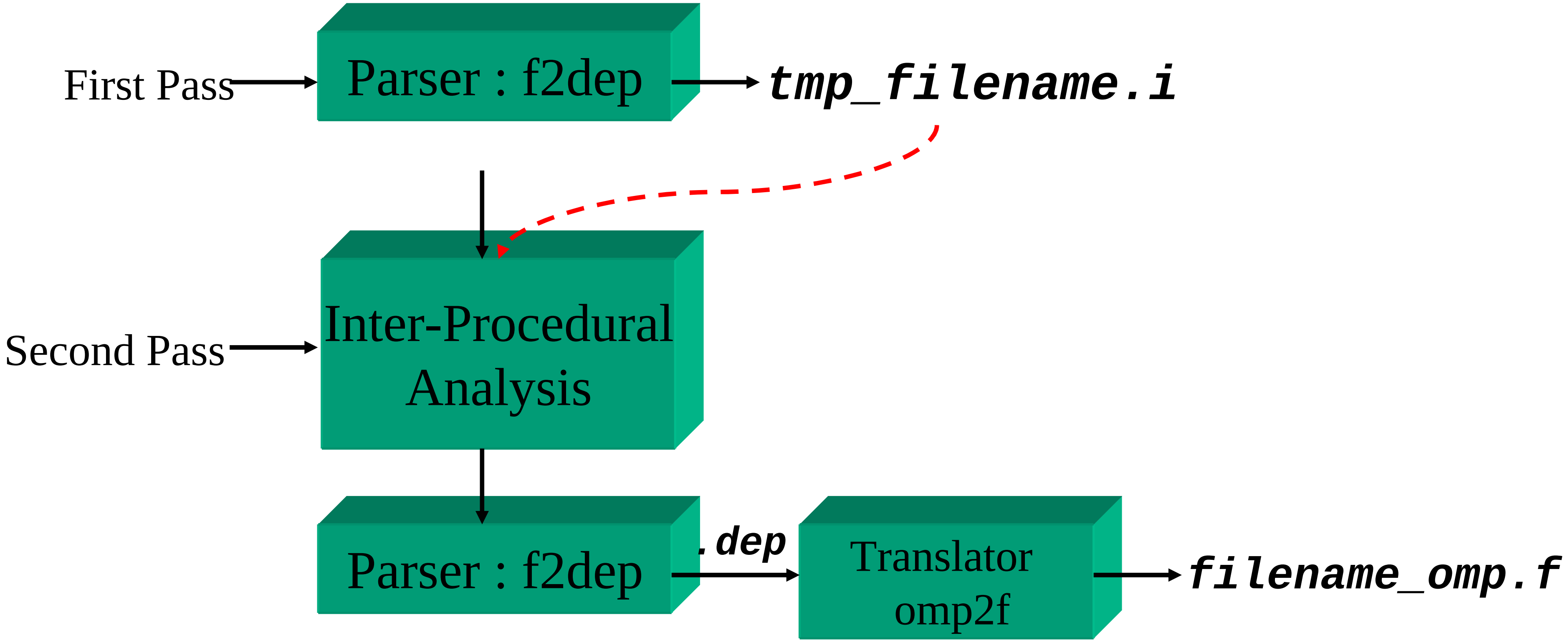
	CCRG		Intel	
	subroutine	execution time	subroutine	execution time
1	**`zgemm`**	82.10	`dlaran`	9.77
2	`gammul`	10.77	`zaxpy`	8.57
3	`zaxpy`	7.74	**`zgemm`**	**7.91**
4	`dlaran`	7.35	`lsame`	1.87

SU3MUL & ZGEMM

```
 SUBROUTINE SU3MUL(U,TRANSU,X,RESULT)
    …………
    CALL ZGEMM(TRANSU, 'NO TRANSPOSE',3,4,3,
&              ONE,U,3,X,3,ZERO,RESULT,3)
    RETURN
   END

 SUBROUTINE ZGEMM (TRANSA,TRANSB,M,N,K,
&                  ALPHA,A,LDA,B,LDB,BETA,C,LDC )
  ………
 END
```

- `M,N,K` are loop control variables in ZGEMM
- `M,N,K` are used in the logical expression of `IF` statement
- The values of **M,N,K** have not been propagated to zgemm when using CCRG

Inter-Procedural Optimization



COMP
CCRG Open MP

Intermediate file in IPO

tmp_su3mul.i :

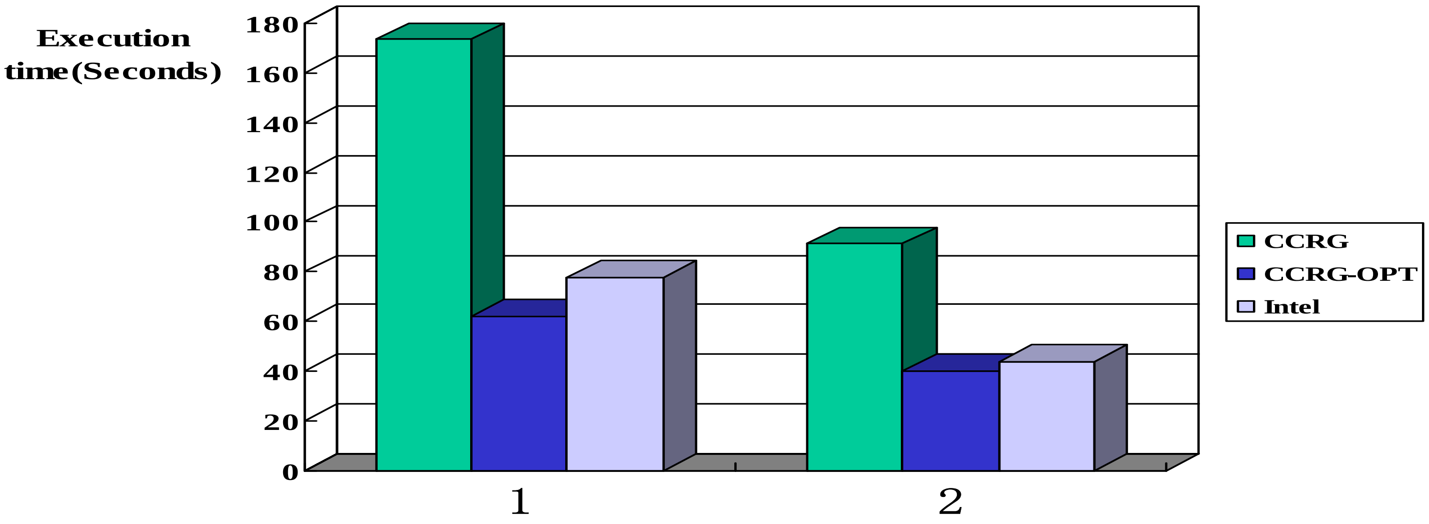
```
{ SUBROUTINE "SU3MUL"
  (FORMAL ("U" COMPLEX*16 DIMENSION(2 3 *))
              ("TRANSU" CHARACTER*1 SCALAR)
              ("X" COMPLEX*16 DIMENSION(1 *))
              ("RESULT" COMPLEX*16 DIMENSION(1 *)))

  (SUBROUTINE "ZGEMM"
     (ACTUAL (TRANSU, ’NO TRANSPOSE’,3,4,3,
                ONE,U,3,X,3,ZERO,RESULT,3)
}
```

ZGEMM after IPO

```
     SUBROUTINE ZGEMM (TRANSA, TRANSB, M, N, K, ALPHA,
&                     A, LDA, B, LDB,BETA, C, LDC )
     ! Variables Declaration Statements.........
     ! Assignment to Formal parameters
       M = 3
       N = 4
       K = 3
     !Other Executable Statements
     END
```

COMP
CCRG Open MP

Performance of wupwise after IPO



Conclusions

- With CCRG OpenMP compiler, all of SPEC OMP programs can be compiled and executed on SMP machines efficiently.
- To improve the performance, it is necessary and feasible for OpenMP compilers to optimize programs at the source level.

COMP
CCRG Open MP

Future Work

- Fortran 95 Syntax
  - **KIND**

```
integer, parameter:: b8 =  selected_real_kind(14)
real(b8) a
```

The value of b8 should be calculated by the translator.

- Source-level Optimization
  - Data Privatization
    - **e.g, FIRSTPRIVATE**
  - More Classic Optimization