THE UNIVERSITY OF ALABAMA ENGINEERING

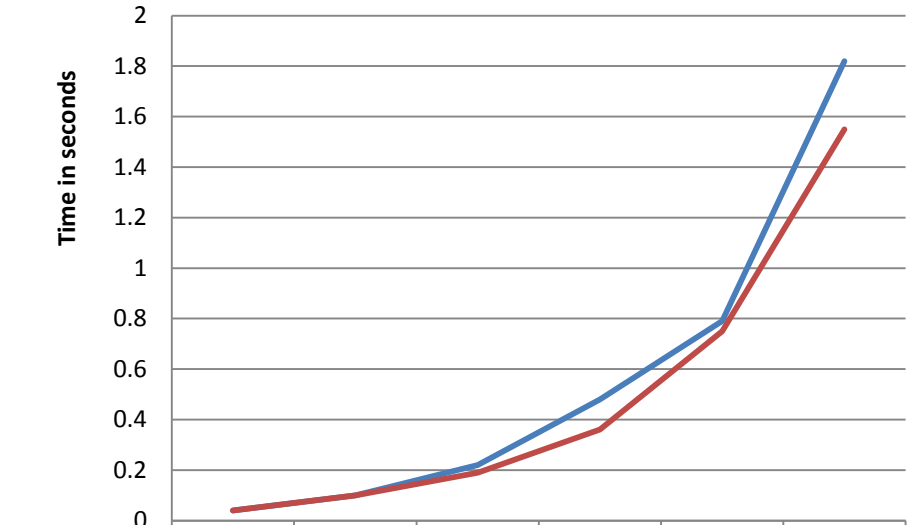
UAB

Refining High Performance FORTRAN Code from Programming Model Dependencies

Ferosh Jacob[1], Jeff Gray[1], Purushotham Bangalore[2], Marjan Memik[32]

[1]Department of Computer Science, University of Alabama
[2]Department of Computer and Information Sciences, University of Alabama at Birmingham
[3]Faculty of Electrical Engineering and Computer Science, University of Maribor
Contact: *fjacob@crimson.ua.edu*

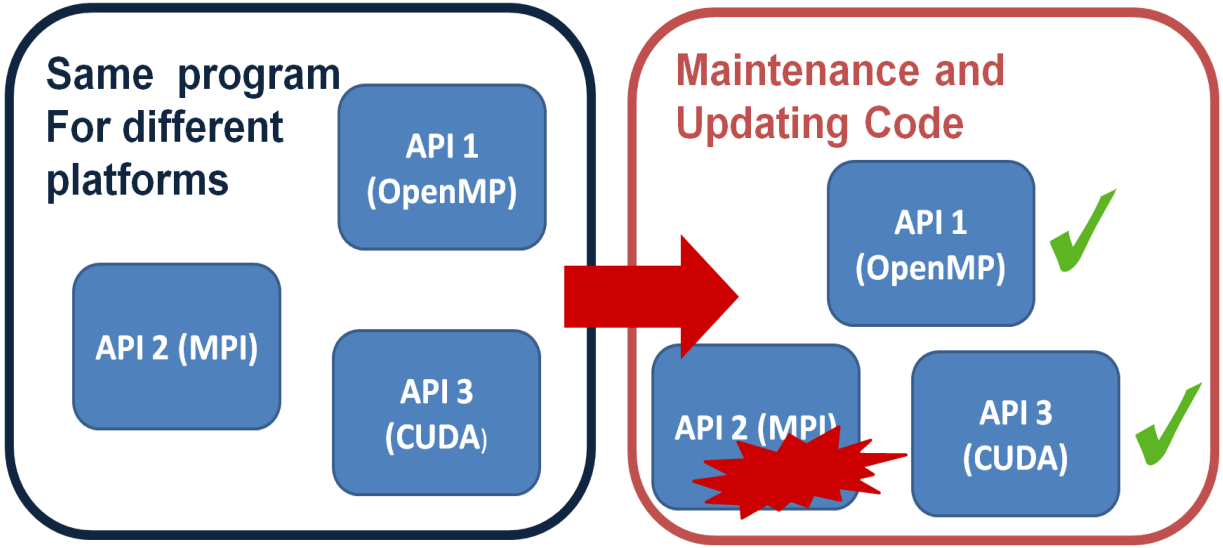
Challenges in Parallel Programming



	19	20	21	22	23	24
OpenMP	0.04	0.1	0.22	0.48	0.79	1.82
MPI	0.04	0.1	0.19	0.36	0.75	1.55

The execution plot of the satisfiability problem in Figure 1 shows that even though the performance of OpenMP and MPI are comparable, for small problems the OpenMP version is faster than an MPI solution. In cases where the size of the data varies, different versions of the same program might be required if a single HPC library is used.

Architecture Dependencies in Parallel Programming



Manually maintaining such variations induces unnecessary redundant effort that is also very prone to human errors in maintaining and updating the core algorithms. Therefore, the development of an HPC program is often limited to a specific parallel library. Otherwise, the programmer pays the price of developing and maintaining several versions of the same program.

Program Analysis of FORTRAN OpenMP Programs

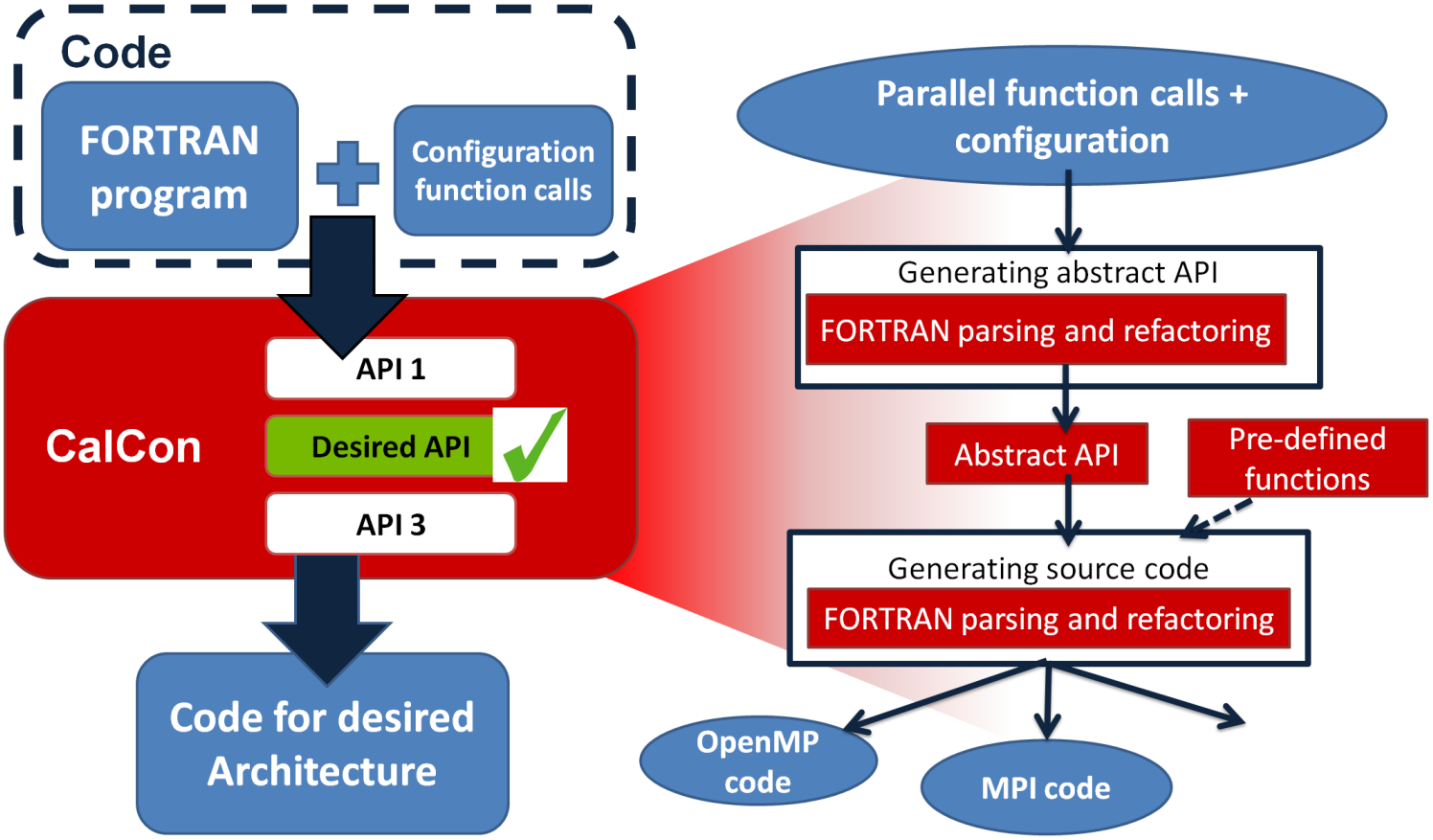
No	*Program Name*	*Total LOC*	*Parallel LOC*	*No. of blocks*	*R*	*W*
1	2D Integral with Quadrature rule	601	11 (2%)	1	√	
2	Linear algebra routine	557	28 (5%)	4		√
3	Random number generator	80	9 (11%)	1		
4	Logical circuit satisfiability	157	37 (18%)	1	√	
5	Dijkstra's shortest path	201	37 (18%)	1		
6	Fast Fourier Transform	278	51 (18%)	3		
7	Integral with Quadrature rule	41	8 (19%)	1	√	
8	Molecular dynamics	215	48 (22%)	4	√	√
9	Prime numbers	65	17 (26%)	1	√	
10	Steady state heat equation	98	56 (57%)	3	√ √	

Shared memory features	*Parallel features*
Variable modifiers, Critical and Singular blocks, Number of threads	Parallel blocks, Reduction and Barrier blocks, Number of instances, Workshare

Analysis Conclusion

A DSL that uses only the parallel features can express parallel problems in a platform-independent manner. Most of the programs involve an initialization segment that initializes the execution of the parallel part, and a code segment that is used to collect data from the parallel instances

Proposed Approach to Express Parallel Programs in FORTRAN



MPI Case Study

```
!Part 1: Master process setting up the data
if ( my_id == 0 ) then    do p = 1, p_num - 1

      my_a = ( real ( p_num - p,     kind = 8 ) * a    &
             + real (         p - 1, kind = 8 ) * b ) &
             / real ( p_num     - 1, kind = 8 )

      target = p
      tag = 1
      call MPI_Send ( my_a, 1, MPI_DOUBLE_PRECISION, &
                      target, tag, &MPI_COMM_WORLD, &
                      error_flag )
.....
    end do

!Part 2: Parallel execution

  else

    source = master
    tag = 1
    call MPI_Recv ( my_a, 1, MPI_DOUBLE_PRECISION, source,  tag,
&
      MPI_COMM_WORLD, status, error_flag )

    my_total = 0.0D+00
    do i = 1, my_n
      x = ( real ( my_n - i,     kind = 8 ) * my_a    &
          + real (         i - 1, kind = 8 ) * my_b ) &
          / real ( my_n     - 1, kind = 8 )
      my_total = my_total + f ( x )
    end do
    my_total = ( my_b - my_a ) * my_total / real
                                       ( my_n, kind = 8 )

  end if

!Part 3: Results from different processes are collected to
! calculate the final result

  call MPI_Reduce ( my_total, total, 1,
                    MPI_DOUBLE_PRECISION, & MPI_SUM,
                    master, MPI_COMM_WORLD, error_flag)
```

```
!Work share part
    do p = 1, instance_num - 1
      my_a = ( real ( instance_num - p,     kind = 8 ) * a    &
             + real (         p - 1, kind = 8 ) * b ) &
             / real ( instance_num     - 1, kind = 8 )
      call distribute (my_a)
    end do
!Declaring parallel block
 call parallel(num,'quadrature')
    my_total = 0.0D+00
    do i = 1, my_n
      x = ( real ( my_n - i,     kind = 8 ) * my_a    &
          + real (         i - 1, kind = 8 ) * my_b ) &
          / real ( my_n     - 1, kind = 8 )
      my_total = my_total + f ( x )
    end do
    my_total = ( my_b - my_a ) * my_total / real
                                       ( my_n, kind = 8 )
 call endparallel('quadrature');
!
! Configuration file for FORTRAN program above
!
block 'quadrature'
init:
    source = master
    tag = 1
    call MPI_Recv ( my_a, 1, MPI_DOUBLE_PRECISION, source,  tag,
&
      MPI_COMM_WORLD, status, error_flag ).
final:
call MPI_Reduce ( my_total, total, 1,
                  MPI_DOUBLE_PRECISION, & MPI_SUM,
                  master, MPI_COMM_WORLD, error_flag).
distribute param:
      call MPI_Send ( param, 1, MPI_DOUBLE_PRECISION, &
                      target, tag, &MPI_COMM_WORLD, &
                      error_flag ).
```

Conclusion and Future Work

Conclusion:

For the evolution of high performance FORTRAN code, it is necessary to separate the code of **the core computation** from the **machine or architecture dependencies** that may come from usage of a specific API.

We **analyzed** ten FORTRAN programs from diverse domains to understand the usage of OpenMP in scientific code. The analysis revealed that programs often share a common structure such that platform and machine details could be specified in a different file.

A case study is included to show that the approach can be extended to other architectures.

Future work:

Includes **refactoring the legacy code** to the approach specified in this paper with minimum input from the user. Another direction will be focused on executing the **parallel programs to a GPU**. Conducting a **user study** to explore the advantages and disadvantages from a human factors perspective is another direction of work.

Thank You

- Questions?

Ferosh Jacob[1], Jeff Gray[1], Purushotham Bangalore[2], Marjan Memik[32]

[1]Department of Computer Science, University of Alabama
[2]Department of Computer and Information Sciences, University of Alabama at Birmingham
[3]Faculty of Electrical Engineering and Computer Science, University of Maribor

Contact: *fjacob@crimson.ua.edu*