



TEXAS ADVANCED COMPUTING CENTER

WWW.TACC.UTEXAS.EDU

UMassAmherst  
The Commonwealth's Flagship Campus



TEXAS  
The University of Texas at Austin

# Semi-Automatic Code Modernization for Optimal Parallel I/O

SCEC 2018

December 14, 2018

**PRESENTED BY:**

Trung Nguyen Ba:

[tnguyenba@cs.umass.edu](mailto:tnguyenba@cs.umass.edu)

Ritu Arora: [rauta@tacc.utexas.edu](mailto:rauta@tacc.utexas.edu)

# Interactive Parallelization Tool (IPT)

```
INTERACTIVE PARALLELIZATION TOOL

Terminal Compile Run Job History Help

Terminal

Your IPT terminal is ready.

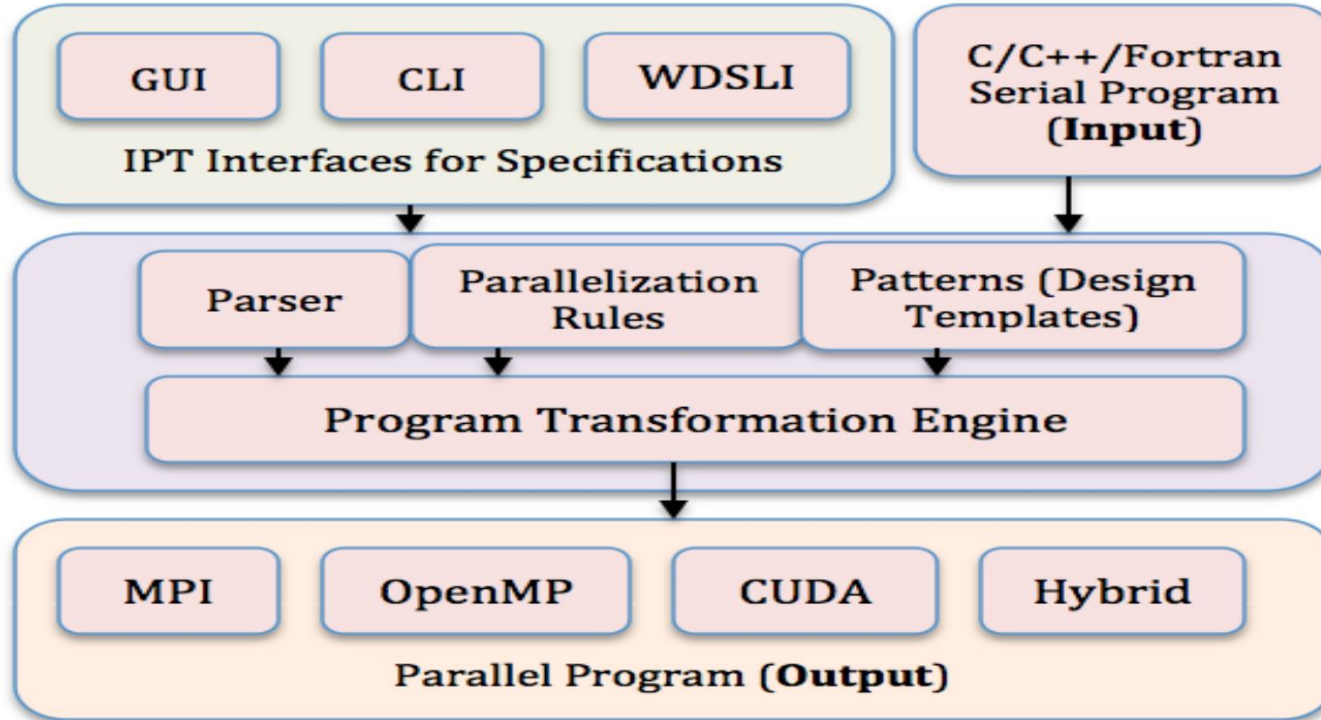
ipt@4d6b1fcaa34d:~$ IPT md.c
NOTE: We currently support only C and C++ programs.

Please select a parallel programming model from the following available options:
1. MPI
2. OpenMP
3. CUDA
1

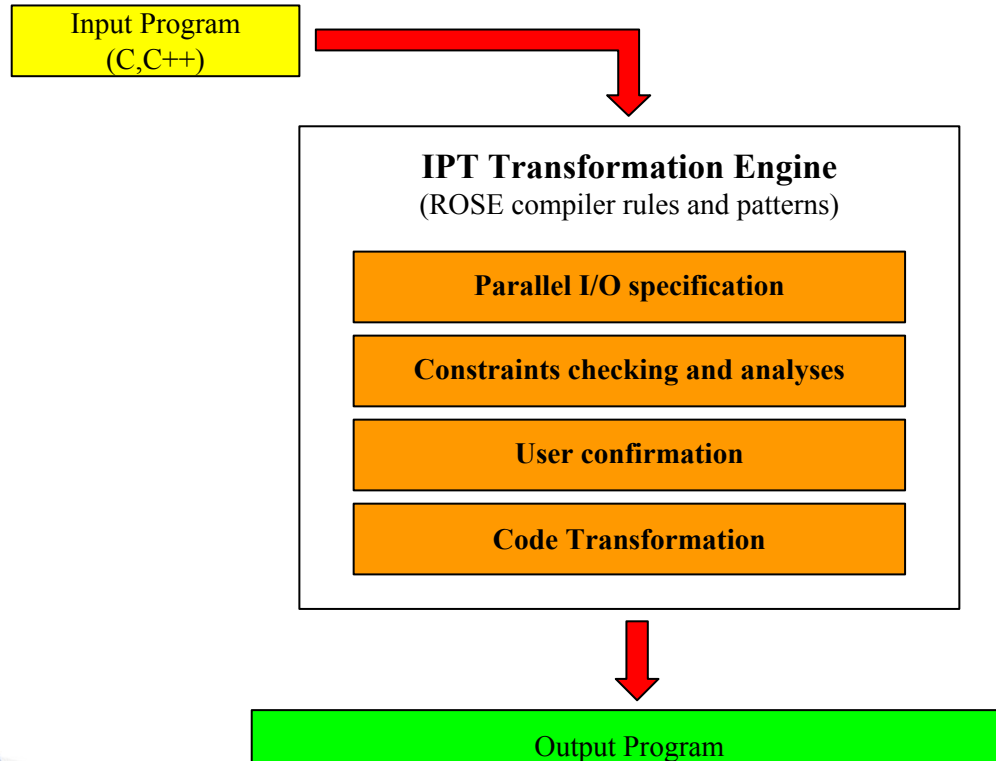
Please note that by default, the MPI Environment Initialization functions will be set in the main
function.

Please choose the function that you want to parallelize from the list below
1 : main
2 : compute
3 : cpu_time
4 : dist
5 : initialize
6 : r8mat_uniform_ab
7 : timestamp
8 : update
```

# IPT Design Overview

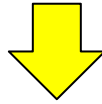


# Parallel MPI I/O with IPT

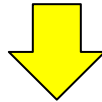


# Writing/Reading ASCII Files

User chosen the block of I/O code



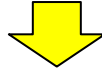
IPT inserts code calculating file offset  
and buffering file write/read statements



IPT inserts the MPI I/O calls

# Writing/Reading 1-D, 2-D arrays in Binary Files

User chosen the block of I/O code



IPT detects important writing/reading information



IPT inserts MPI I/O and remove the serial I/O code



IPT inserts the MPI I/O calls

# Example of Optimizable I/O Patterns

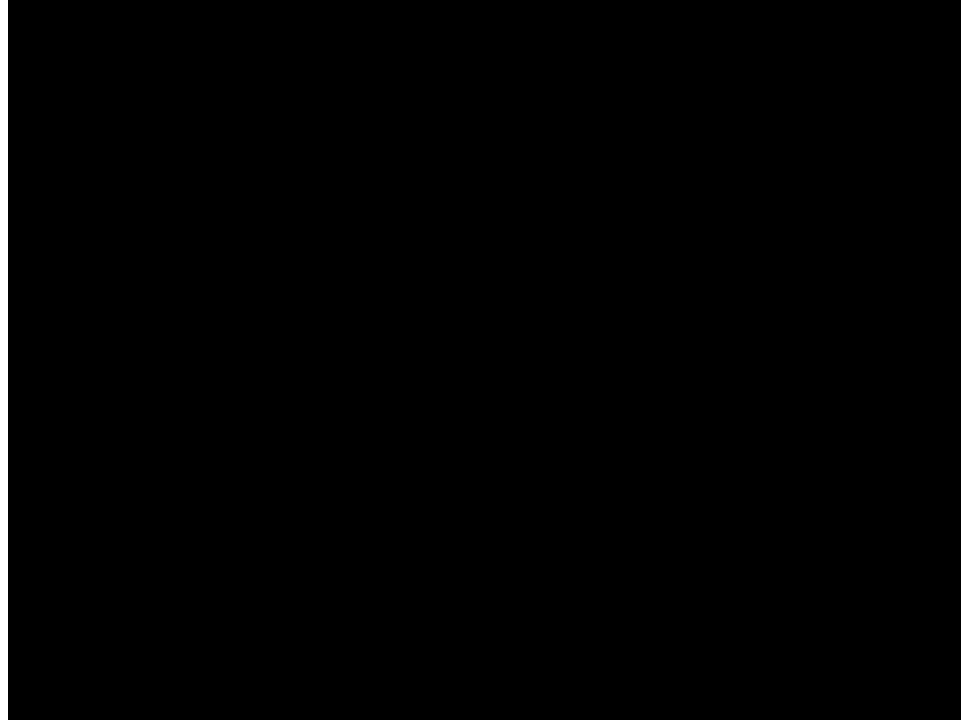
Optimizable 1-D array I/O	Optimizable 2-D array I/O
<pre>int a[100]; for (int i =0; i &lt; 100;i++) {     fprintf(f, "%d",a[i]); }</pre>	<pre>int a[100][100]; for (int i =0; i &lt; 100;i++) {     for (int j =0; j &lt; 100;j++) {         fprintf(f, "%d",a[i]);     } }</pre>

# Lustre filesystem

- File stripping to increase I/O bandwidth
  - Inserting stripe size
  - Inserting stripe count



# Demo



# Results and Evaluations

<b>Examples</b>	<b>Serial Time</b> Taken in Seconds	<b>IPT Parallel</b> Time Taken in Seconds <i>4 MPI processes used</i>	<b>Manual Parallel</b> Time Taken in Seconds <i>4 MPI processes used</i>
1-D Array - reading	42	0.55	0.39
1-D Array - writing	54	1.7	1.66
2-D Array - reading	36	0.53	0.55
2-D Array - writing	40	1.71	1.74

***1-D integer array with 100,000,000 elements***

***2-D integer array with 10,000x10,000 elements***

<b>Examples</b>	<b>Serial</b> <i>Total #LoC</i>	<b>IPT Parallel</b> <i>(#LoC Inserted-or-Deleted) / (#LoC)</i>	<b>Manual Parallel</b> <i>(#LoC Inserted-or-Deleted) / (Total #LoC)</i>
1-D Array - reading	11	Lines deleted: 3 Lines added: 32 Total number of lines: 40 %age of code change: 87.5	Lines deleted: 5 Lines added: 16 Total number of lines: 22 %age of code change: 95.5
1-D Array - writing	13	Lines deleted: 3 Lines added: 36 Total number of lines: 46 %age of code change: 84.7	Lines deleted: 6 Lines added: 15 Total number of lines: 22 %age of code change: 95.5
2-D Array - reading	13	Lines deleted: 5 Lines added: 30 Total number of lines: 38 %age of code change: 92.1	Lines deleted: 6 Lines added: 20 Total number of lines: 27 %age of code change: 96.3
2-D Array - writing	18	Lines deleted: 5 Lines added: 38 Total number of lines: 51 %age of code change: 84.3	Lines deleted: 7 Lines added: 24 Total number of lines: 35 %age of code change: 85.6

*LoC = Lines of Code*

# Conclusion

- Overview of parallelizing I/O code with IPT
- IPT supports both ASCII and Binary read and write
  - It also supports file stripping on Luster filesystem
- Performance:
  - IPT-parallel version has almost the same performance as the manual parallel version
  - Reducing the manual effort for parallelizing code for more than 80%

# Acknowledgement

The work presented in this paper was made possible through the National Science Foundation (NSF) award number 1642396.