Precimonious & HiFPTuner

Tuning Assistant for Floating-Point Precision

Ignacio Laguna, Harshitha Menon
Lawrence Livermore National Laboratory

Michael Bentley, Ian Briggs, Pavel Panchekha, Ganesh Gopalakrishnan
University of Utah

Hui Guo, Cindy Rubio González
University of California at Davis

Michael O. Lam
James Madison University

http://fpanalysistools.org/

This work was supported by through the X-Stack program funded by the U.S. Department of Energy, Office of Science, Advanced Scientific Computing Research under collaborative agreement SC0008699, NSF grant 1750983, and a gift from Oracle.

Floating-Point Precision Tuning

- Floating-point (FP) arithmetic used in variety of domains

- Reasoning about FP programs is difficult
  - Large variety of numerical problems
  - Most programmers are not experts in FP

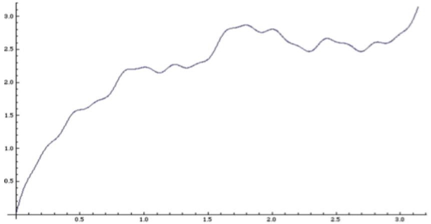
- Common practice: use highest available precision
  - Disadvantage: more expensive!

- Goal: automated techniques to assist in tuning floating-point precision

Example: Arc Length

- Consider the problem of finding the arc length of the function

$$g(x) = x + \sum_{0 \le k \le 5} 2^{-k} \sin(2^k x)$$



- Summing for $x_k \in (0, \pi)$ into n subintervals

$$\sum_{k=0}^{n-1} \sqrt{h^2 + (g(x_{k+1}) - g(x_k))^2}$$ with $h = \pi / n$ and $x_k = kh$

	Precision	Slowdown	Result	
1	double-double	20X	5.795776322412856	✓
2	double	1X	5.795776322413031	✗
3	mixed precision	< 2X	5.795776322412856	✓

Example: Arc Length

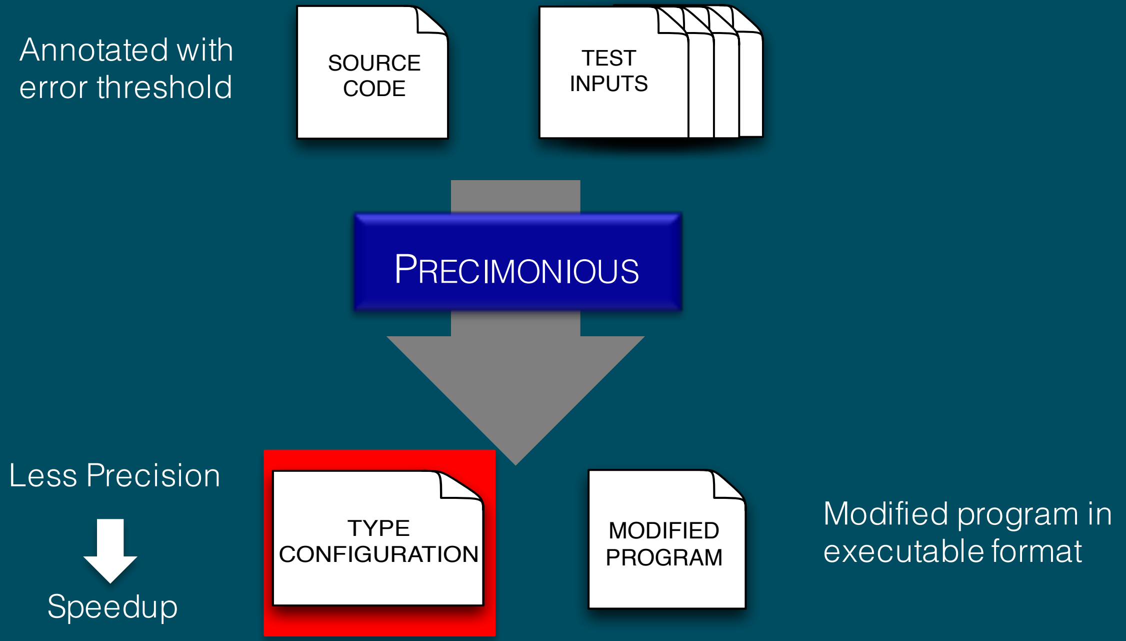
```
long double g(long double x) {
  int k, n = 5;
  long double t1 = x;
  long double d1 = 1.0L;

  for(k = 1; k <= n; k++) {
     ...
  }
  return t1;
}

int main() {
  int i, n = 1000000;
  long double h, t1, t2, dppi;
  long double s1;
  ...
  for(i = 1; i <= n; i++) {
    t2 = g(i * h);
    s1 = s1 + sqrt(h*h + (t2 - t1)*(t2 - t1));
    t1 = t2;
  }
  // final answer stored in variable s1
  return 0;
}
```

Mixed Precision Program

Precimonious

"Parsimonious or Frugal with Precision"

Dynamic Analysis for Floating-Point Precision Tuning



Annotated with error threshold

SOURCE CODE

TEST INPUTS

PRECIMONIOUS

Less Precision

Speedup

TYPE CONFIGURATION

MODIFIED PROGRAM

Modified program in executable format

http://fpanalysistools.org/

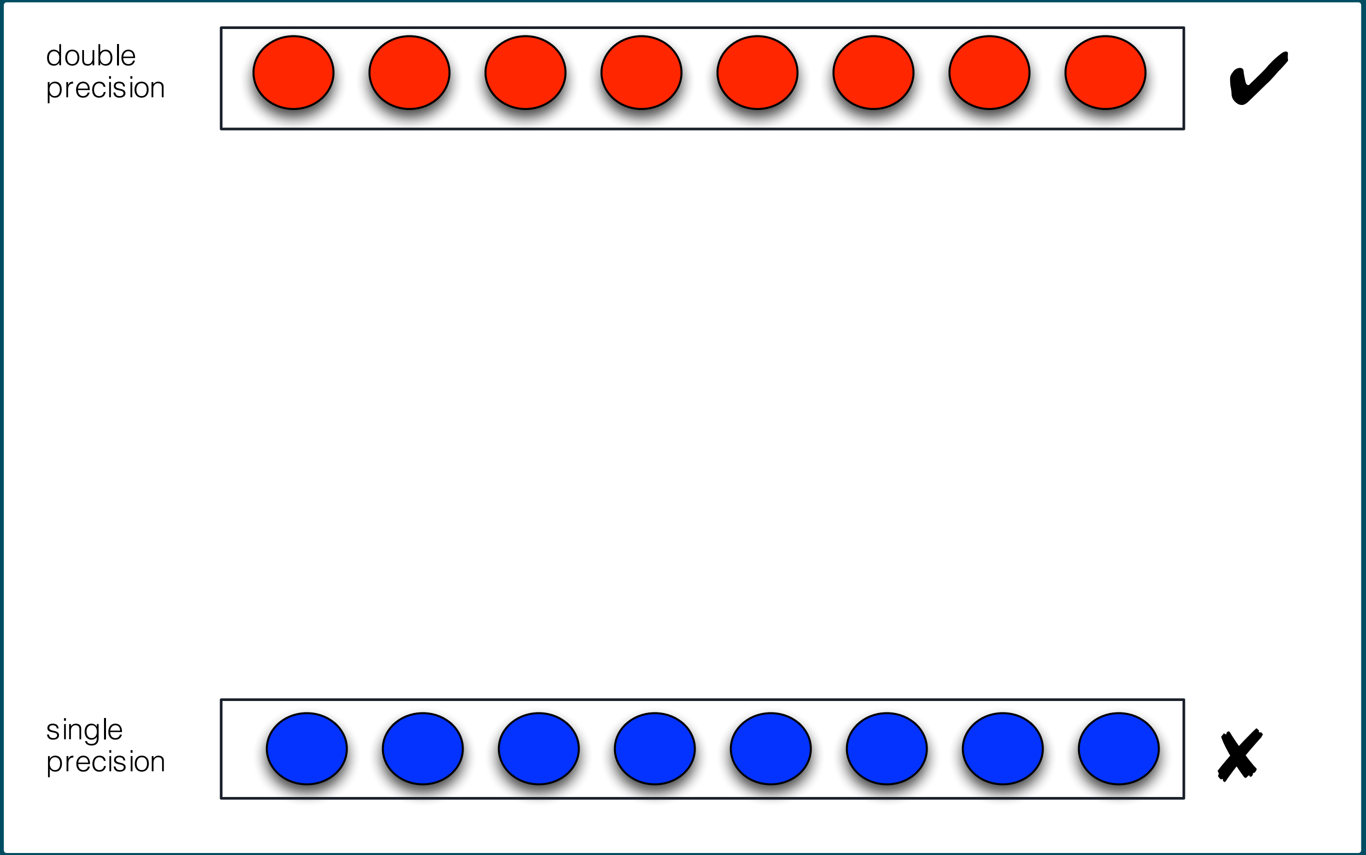
Challenges for Precision Tuning

- Searching efficiently over variable types and function implementations
  - Naïve approach -> exponential time
  - 19,683 configurations for arclength program (3^9)
  - 11 hours 5 minutes
  - Global minimum vs. Local minimum
- Evaluating type configurations
  - Less precision not necessarily faster
  - Based on runtime, energy consumption, etc.
- Determining accuracy constraints
  - How accurate must the final result be?
  - What error threshold to use?

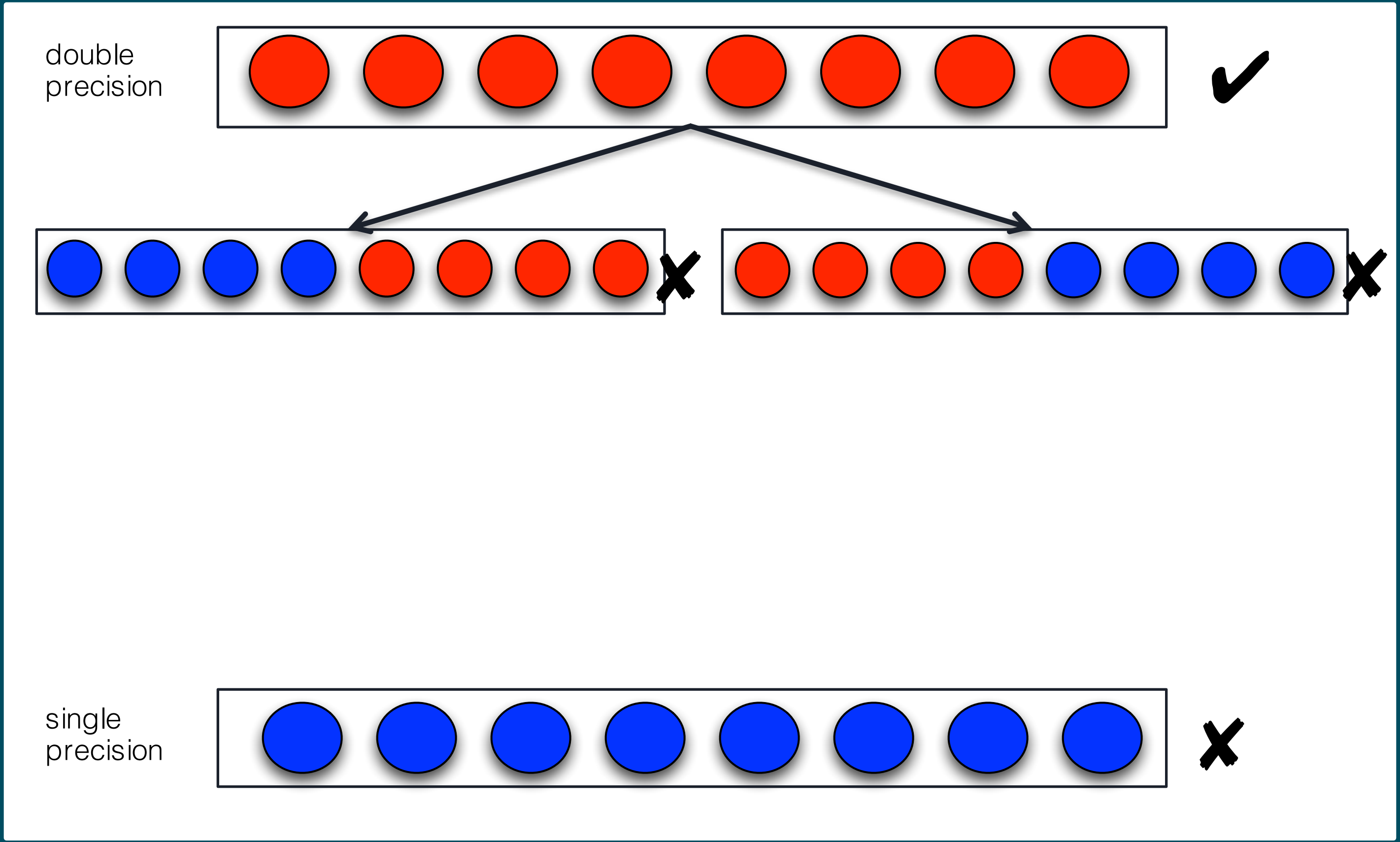
Automated

Specified by the user

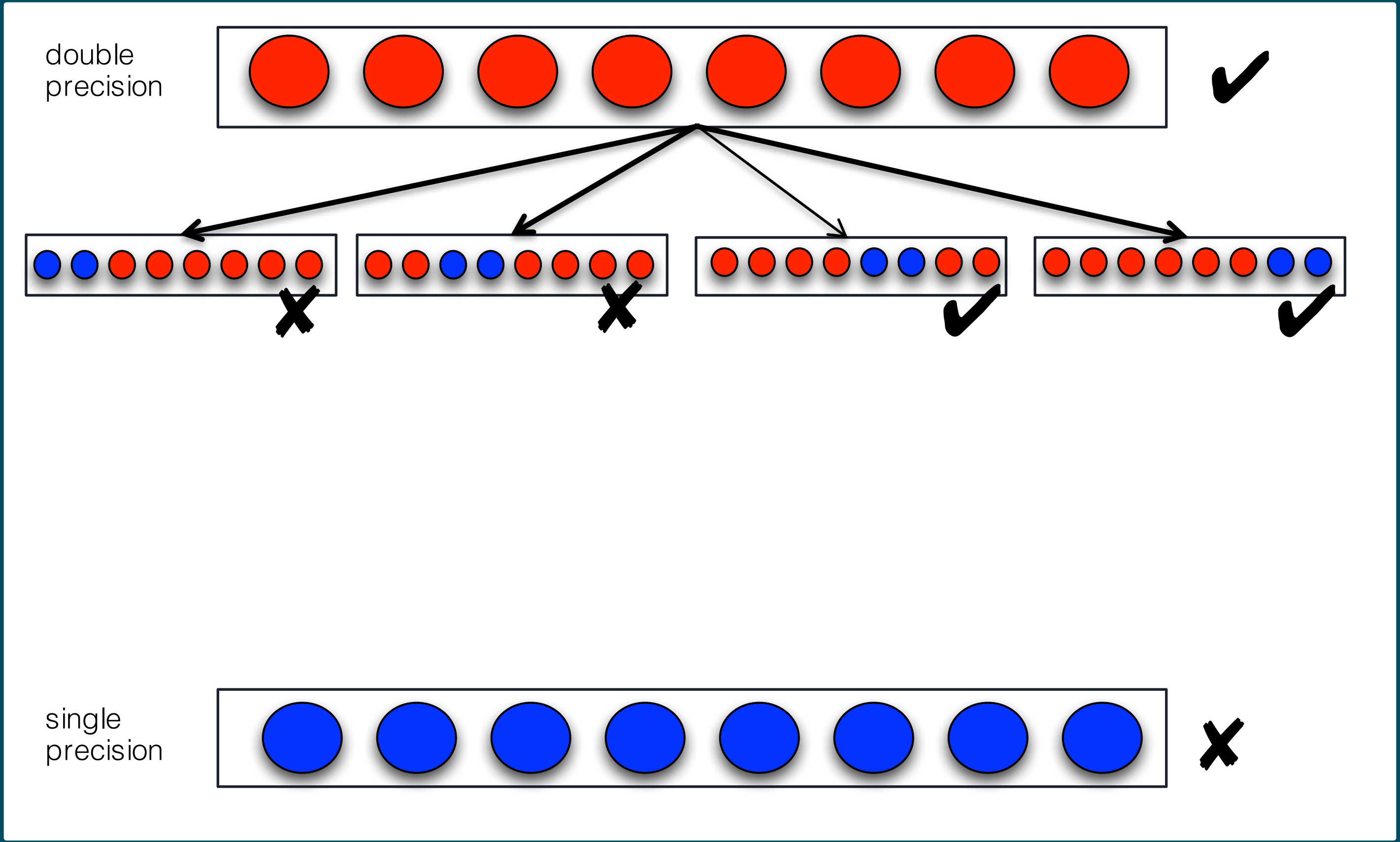
Searching for Type Configuration



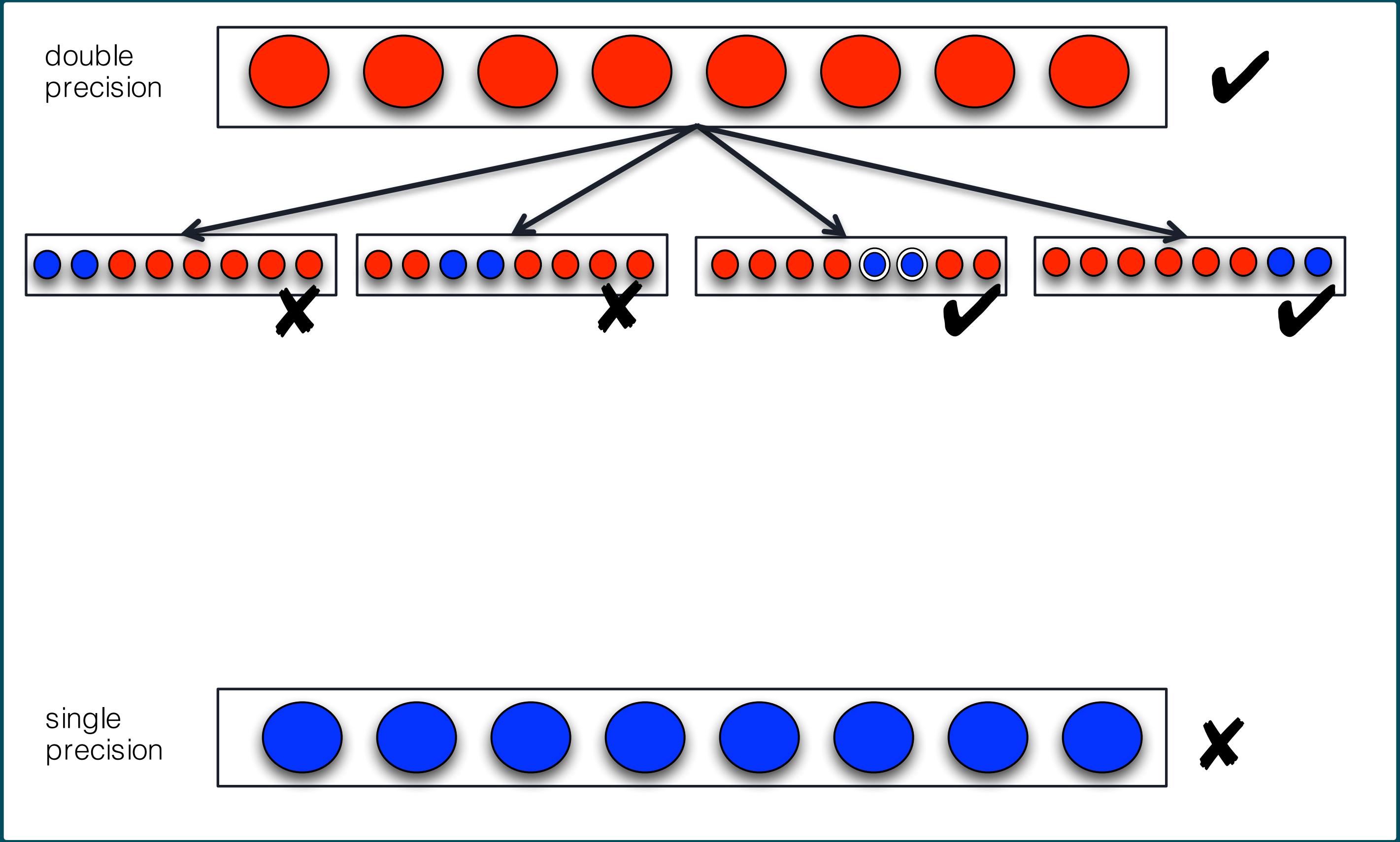
Searching for Type Configuration



Searching for Type Configuration

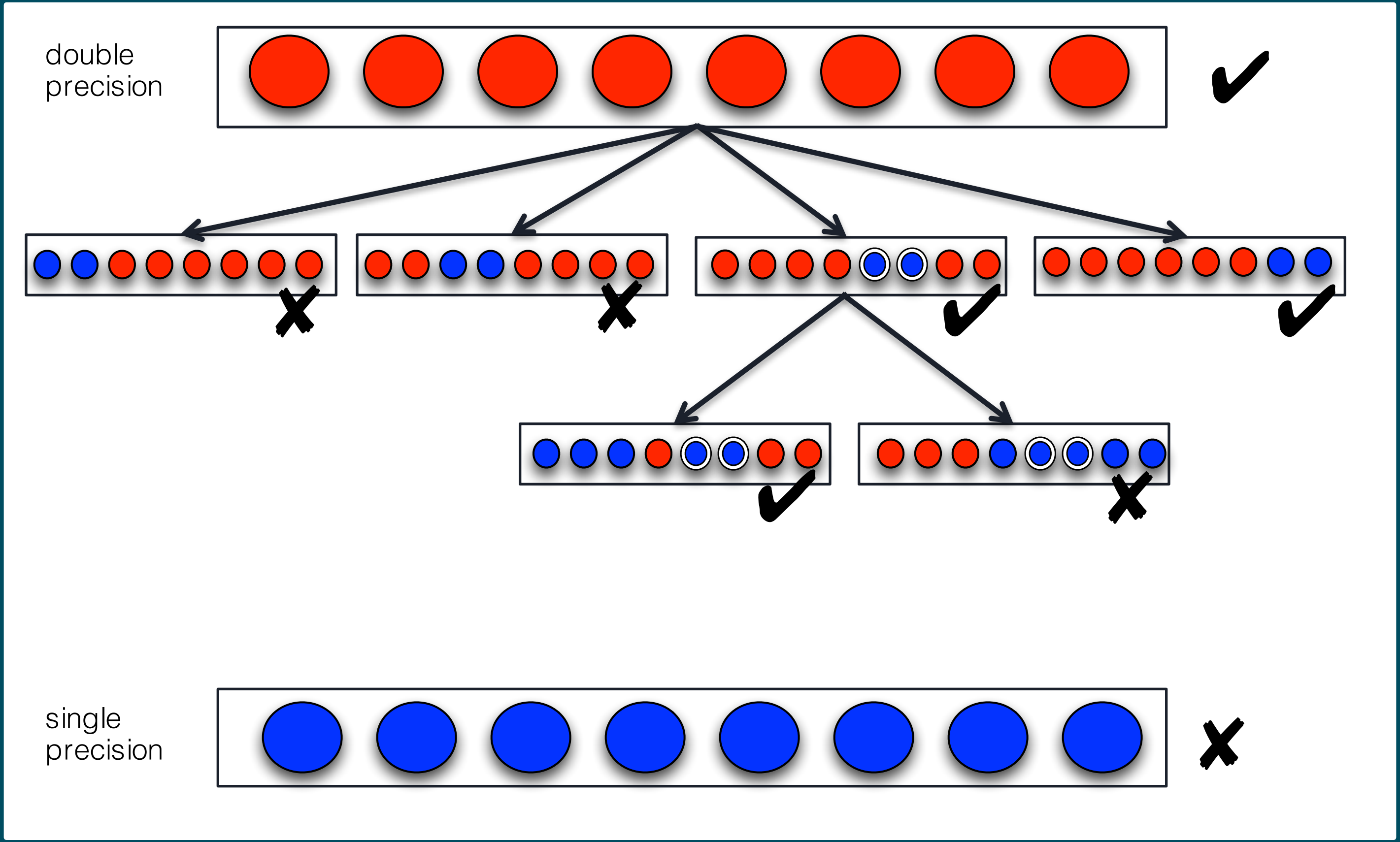


Searching for Type Configuration



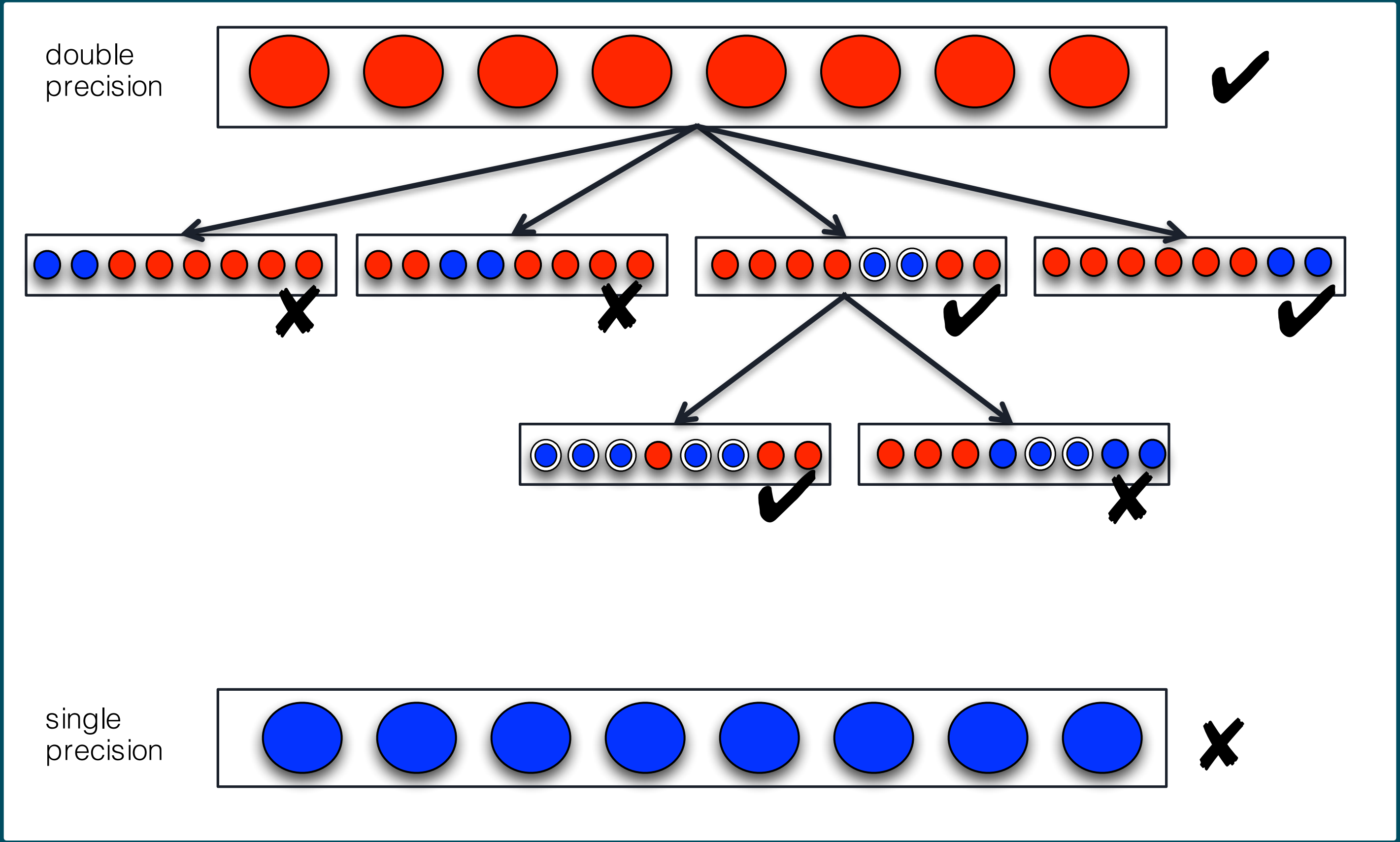
http://fpanalysistools.org/

Searching for Type Configuration

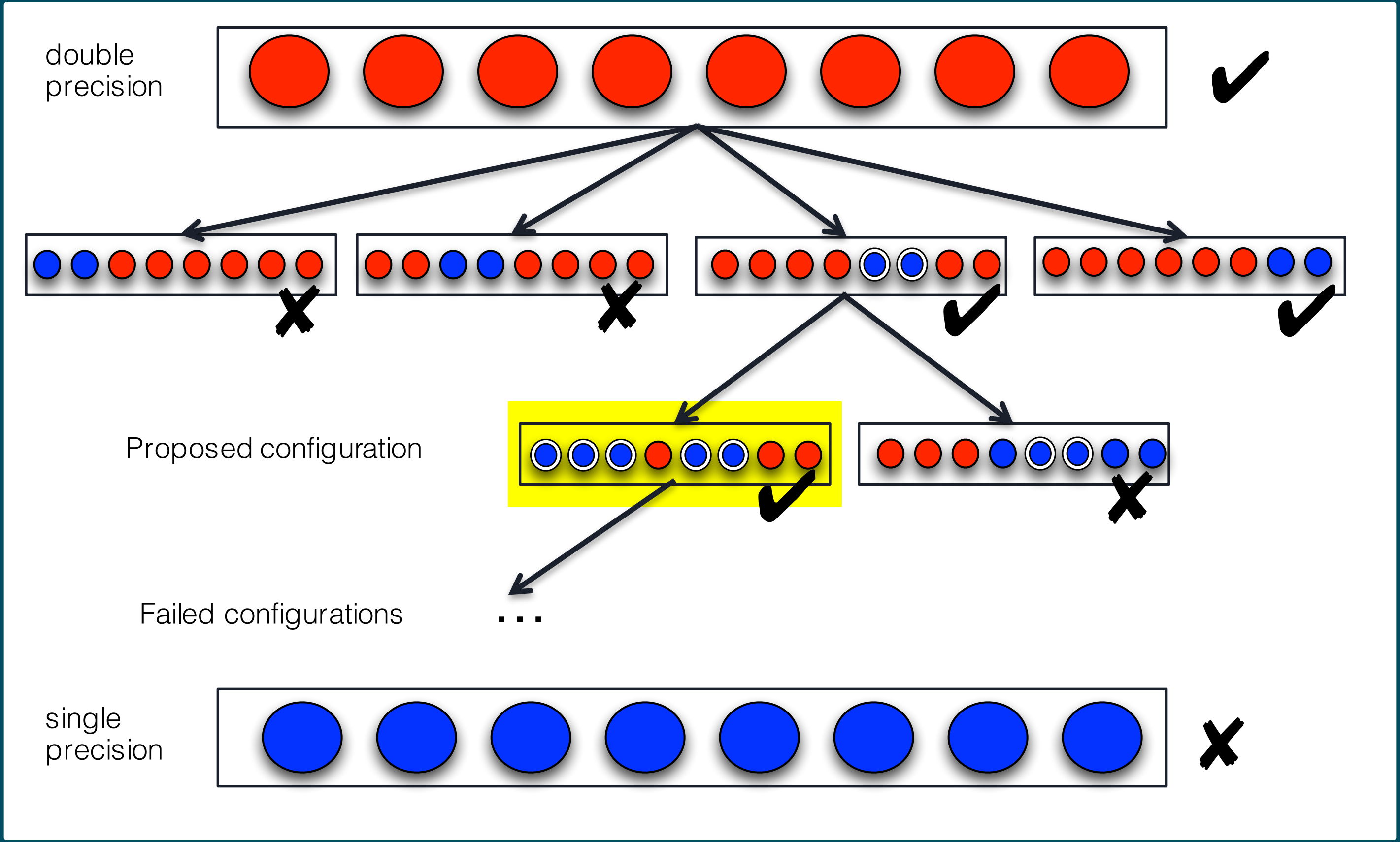


http://fpanalysistools.org/

Searching for Type Configuration



Searching for Type Configuration



double precision

Proposed configuration

Failed configurations

...

single precision

Source code available:
https://github.com/plse/precimonious

Questions?

Directory Structure

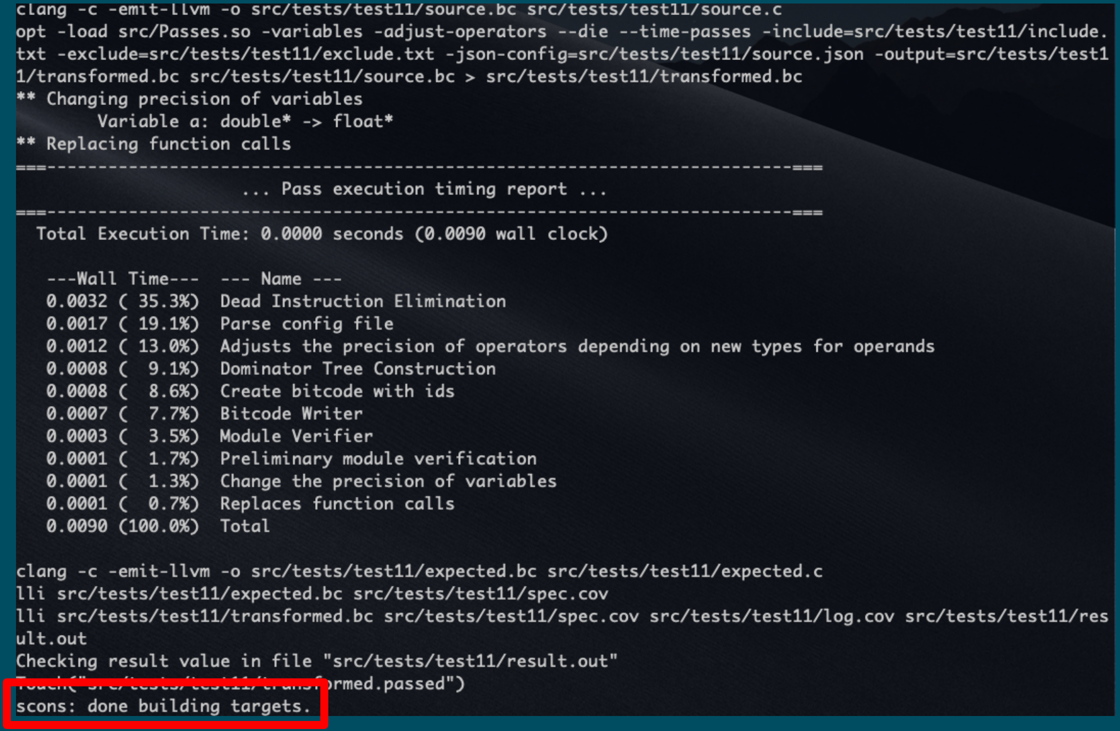
```
/$HOME

  |--/Module-Precimonious
       |---/exercise
       |---/exercise-2

  |--/Module-HiFPTuner
       |---/exercise
       |---/exercise-2
```

Exercise

```
$ cd Module-Precimonious
```

Step 1: Build Precimonious

- Open setup.sh file
- Precimonious uses LLVM and is built using scons
- Execute :
  - $ ./setup.sh



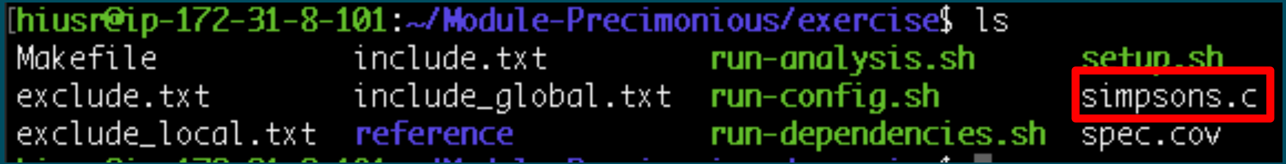
```
clang -c -emit-llvm -o src/tests/test11/source.bc src/tests/test11/source.c
opt -load src/Passes.so -variables -adjust-operators --die --time-passes -include=src/tests/test11/include.
txt -exclude=src/tests/test11/exclude.txt -json-config=src/tests/test11/source.json -output=src/tests/test1
1/transformed.bc src/tests/test11/source.bc > src/tests/test11/transformed.bc
** Changing precision of variables
        Variable a: double* -> float*
** Replacing function calls
===-------------------------------------------------------------------------===
                      ... Pass execution timing report ...
===-------------------------------------------------------------------------===
  Total Execution Time: 0.0000 seconds (0.0090 wall clock)

   ---Wall Time---  --- Name ---
   0.0032 ( 35.3%)  Dead Instruction Elimination
   0.0017 ( 19.1%)  Parse config file
   0.0012 ( 13.0%)  Adjusts the precision of operators depending on new types for operands
   0.0008 (  9.1%)  Dominator Tree Construction
   0.0008 (  8.6%)  Create bitcode with ids
   0.0007 (  7.7%)  Bitcode Writer
   0.0003 (  3.5%)  Module Verifier
   0.0001 (  1.7%)  Preliminary module verification
   0.0001 (  1.3%)  Change the precision of variables
   0.0001 (  0.7%)  Replaces function calls
   0.0090 (100.0%)  Total

clang -c -emit-llvm -o src/tests/test11/expected.bc src/tests/test11/expected.c
lli src/tests/test11/expected.bc src/tests/test11/spec.cov
lli src/tests/test11/transformed.bc src/tests/test11/spec.cov src/tests/test11/log.cov src/tests/test11/res
ult.out
Checking result value in file "src/tests/test11/result.out"
Touch("src/tests/test11/transformed.passed")
scons: done building targets.
```

Success building and running tests

Step 2: Annotate Program (already done)

- Execute :
  - $ cd exercise
  - $ ls

The program we will tune:

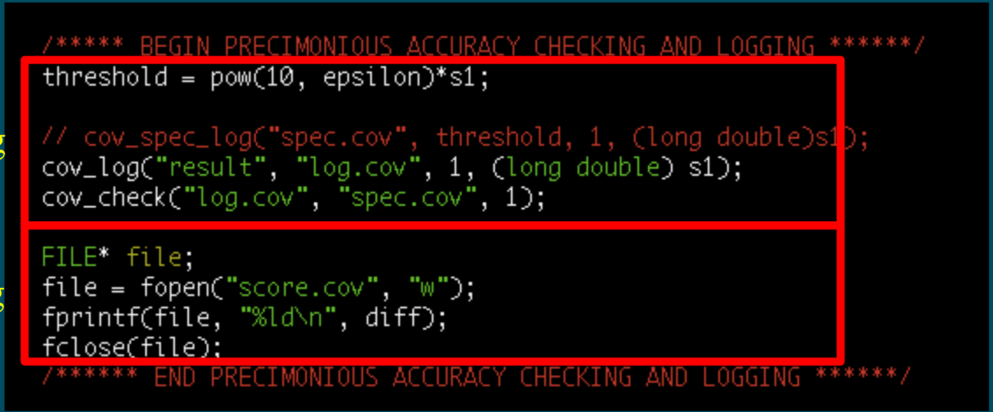


```
[hiusr@ip-172-31-8-101:~/Module-Precimonious/exercise$ ls
Makefile           include.txt          run-analysis.sh      setup.sh
exclude.txt        include_global.txt   run-config.sh        simpsons.c
exclude_local.txt  reference            run-dependencies.sh  spec.cov
```

- Open *simpsons.c* file

Accuracy logging & checking

Performance logging

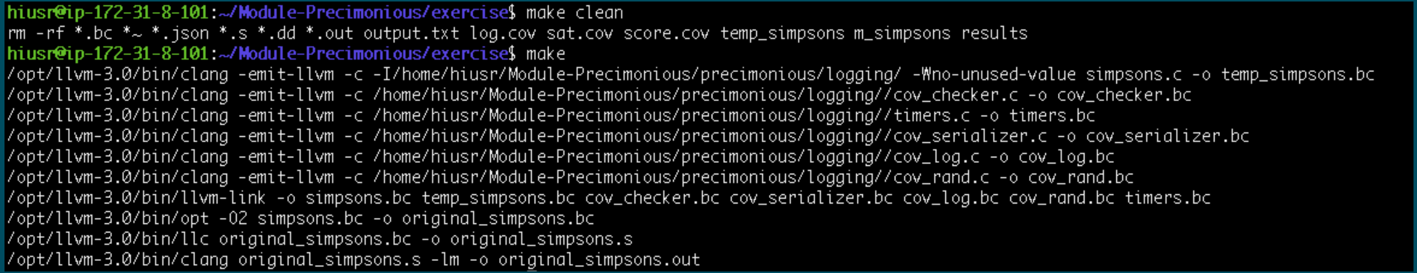


```
/***** BEGIN PRECIMONIOUS ACCURACY CHECKING AND LOGGING ******/
threshold = pow(10, epsilon)*s1;

// cov_spec_log("spec.cov", threshold, 1, (long double)s1);
cov_log("result", "log.cov", 1, (long double) s1);
cov_check("log.cov", "spec.cov", 1);

FILE* file;
file = fopen("score.cov", "w");
fprintf(file, "%ld\n", diff);
fclose(file);
/****** END PRECIMONIOUS ACCURACY CHECKING AND LOGGING ******/
```

http://fpanalysistools.org/

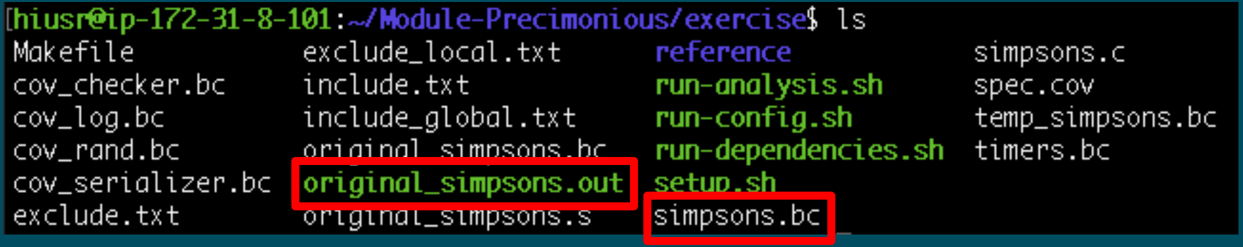
Step 3: Compile Program with Clang

- Execute :
  - $ make clean
  - $ make



```
hiusr@ip-172-31-8-101:~/Module-Precimonious/exercise$ make clean
rm -rf *.bc *~ *.json *.s *.dd *.out output.txt log.cov sat.cov score.cov temp_simpsons m_simpsons results
hiusr@ip-172-31-8-101:~/Module-Precimonious/exercise$ make
/opt/llvm-3.0/bin/clang -emit-llvm -c -I/home/hiusr/Module-Precimonious/precimonious/logging/ -Wno-unused-value simpsons.c -o temp_simpsons.bc
/opt/llvm-3.0/bin/clang -emit-llvm -c /home/hiusr/Module-Precimonious/precimonious/logging//cov_checker.c -o cov_checker.bc
/opt/llvm-3.0/bin/clang -emit-llvm -c /home/hiusr/Module-Precimonious/precimonious/logging//timers.c -o timers.bc
/opt/llvm-3.0/bin/clang -emit-llvm -c /home/hiusr/Module-Precimonious/precimonious/logging//cov_serializer.c -o cov_serializer.bc
/opt/llvm-3.0/bin/clang -emit-llvm -c /home/hiusr/Module-Precimonious/precimonious/logging//cov_log.c -o cov_log.bc
/opt/llvm-3.0/bin/clang -emit-llvm -c /home/hiusr/Module-Precimonious/precimonious/logging//cov_rand.c -o cov_rand.bc
/opt/llvm-3.0/bin/llvm-link -o simpsons.bc temp_simpsons.bc cov_checker.bc cov_serializer.bc cov_log.bc cov_rand.bc timers.bc
/opt/llvm-3.0/bin/opt -O2 simpsons.bc -o original_simpsons.bc
/opt/llvm-3.0/bin/llc original_simpsons.bc -o original_simpsons.s
/opt/llvm-3.0/bin/clang original_simpsons.s -lm -o original_simpsons.out
```

- Creates LLVM bitcode file and optimized executable for later use

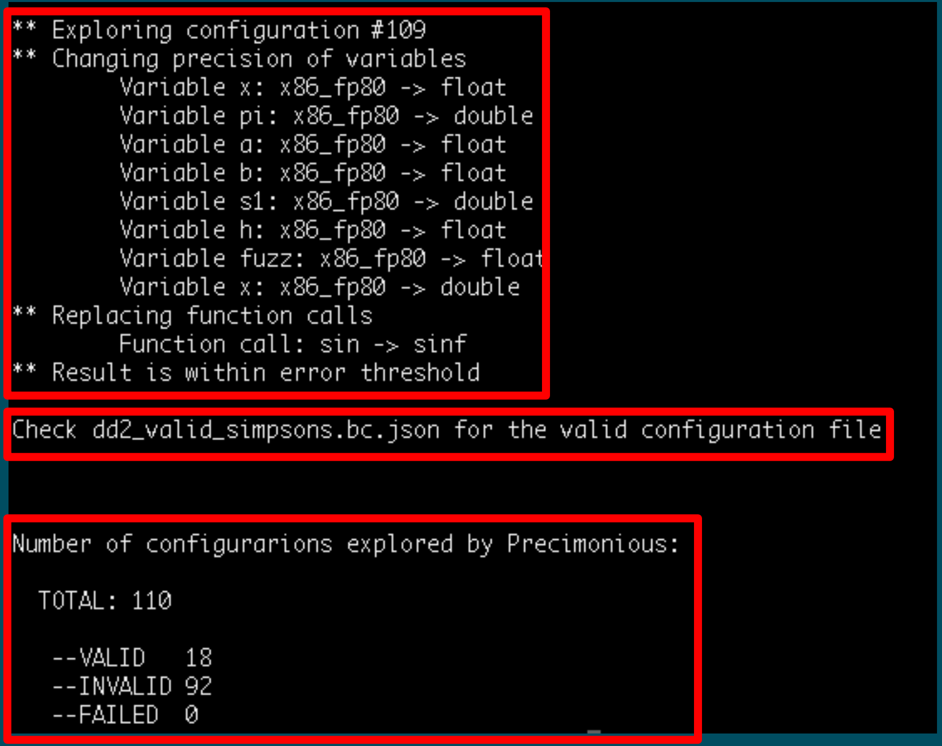


```
[hiusr@ip-172-31-8-101:~/Module-Precimonious/exercise$ ls
Makefile           exclude_local.txt      reference            simpsons.c
cov_checker.bc     include.txt            run-analysis.sh      spec.cov
cov_log.bc         include_global.txt     run-config.sh        temp_simpsons.bc
cov_rand.bc        original_simpsons.bc   run-dependencies.sh  timers.bc
cov_serializer.bc  original_simpsons.out  setup.sh
exclude.txt        original_simpsons.s    simpsons.bc
```

Step 4: Run Analysis on Program

Sample output:

- Execute :
  - $ ./run-analysis.sh simpsons



```
** Exploring configuration #109
** Changing precision of variables
        Variable x: x86_fp80 -> float
        Variable pi: x86_fp80 -> double
        Variable a: x86_fp80 -> float
        Variable b: x86_fp80 -> float
        Variable s1: x86_fp80 -> double
        Variable h: x86_fp80 -> float
        Variable fuzz: x86_fp80 -> float
        Variable x: x86_fp80 -> double
** Replacing function calls
        Function call: sin -> sinf
** Result is within error threshold
```

```
Check dd2_valid_simpsons.bc.json for the valid configuration file
```

```
Number of configurarions explored by Precimonious:

  TOTAL: 110

   --VALID   18
   --INVALID 92
   --FAILED  0
```

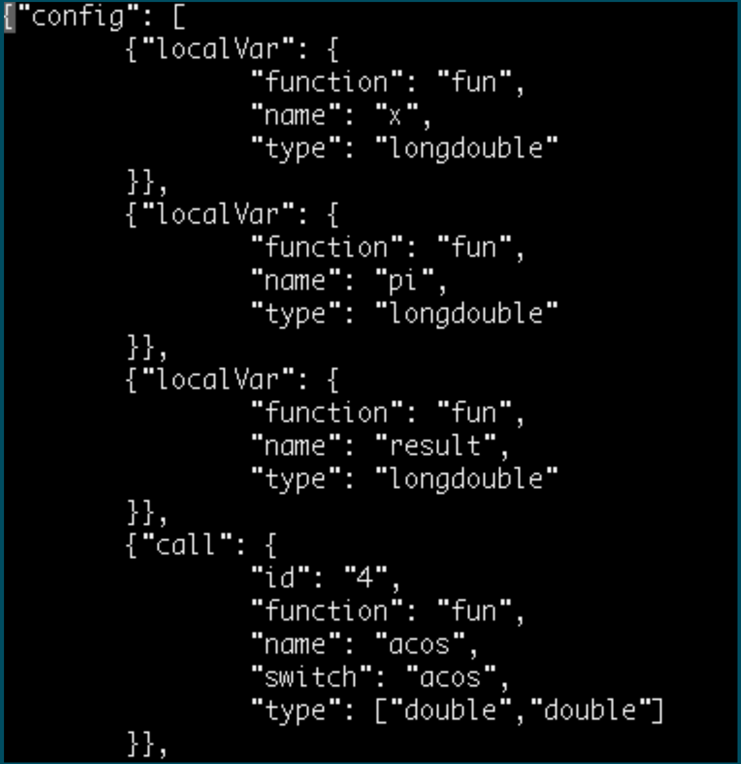
Type changes are listed for each explored configuration

Suggested type configuration

Number of explored configurations

Step 4: Run Analysis – Configuration File

- Open *config_simpsons.json*
- Original type configuration

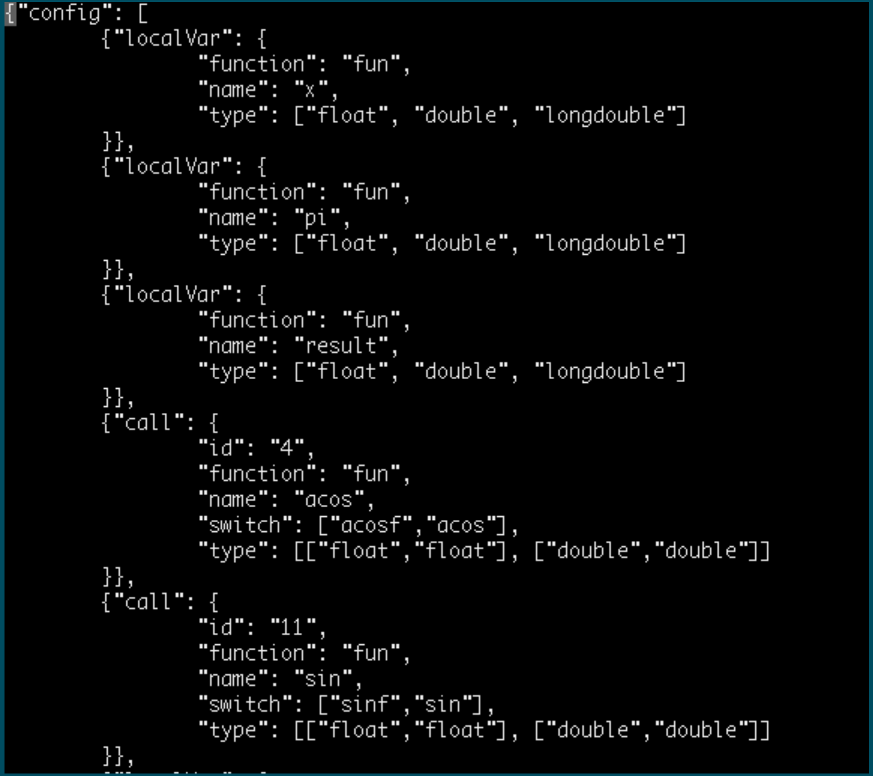


```
{"config": [
        {"localVar": {
                "function": "fun",
                "name": "x",
                "type": "longdouble"
        }},
        {"localVar": {
                "function": "fun",
                "name": "pi",
                "type": "longdouble"
        }},
        {"localVar": {
                "function": "fun",
                "name": "result",
                "type": "longdouble"
        }},
        {"call": {
                "id": "4",
                "function": "fun",
                "name": "acos",
                "switch": "acos",
                "type": ["double","double"]
        }},
```

Step 4: Run Analysis – Search File

- Open search_funarc.json
- Search space file

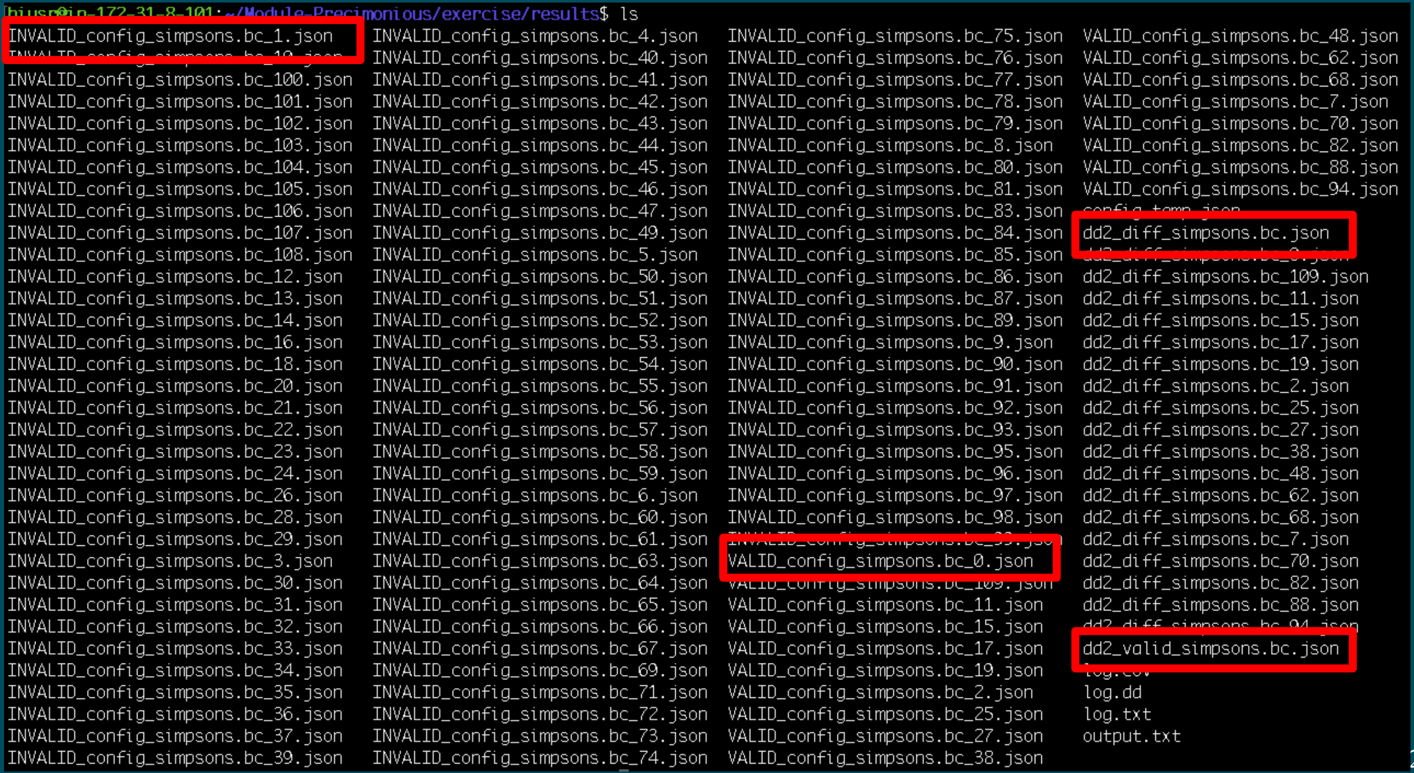
- To exclude functions edit exclude.txt
- To exclude variables edit exclude_local.txt
- Or you can directly edit search file prior to analysis



```
{"config": [
        {"localVar": {
                "function": "fun",
                "name": "x",
                "type": ["float", "double", "longdouble"]
        }},
        {"localVar": {
                "function": "fun",
                "name": "pi",
                "type": ["float", "double", "longdouble"]
        }},
        {"localVar": {
                "function": "fun",
                "name": "result",
                "type": ["float", "double", "longdouble"]
        }},
        {"call": {
                "id": "4",
                "function": "fun",
                "name": "acos",
                "switch": ["acosf","acos"],
                "type": [["float","float"], ["double","double"]]
        }},
        {"call": {
                "id": "11",
                "function": "fun",
                "name": "sin",
                "switch": ["sinf","sin"],
                "type": [["float","float"], ["double","double"]]
        }},
```

Step 4: Run Analysis – Output Files

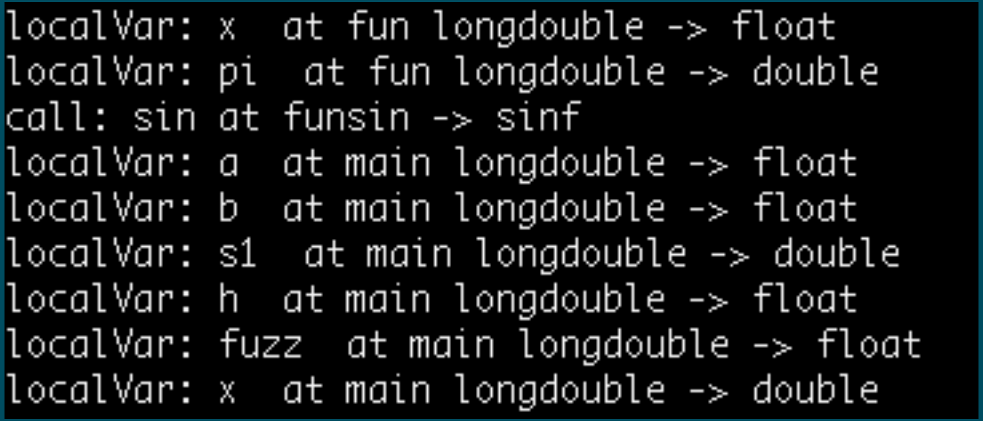
- Execute :
  - $ cd results
  - $ ls



```
hiusr@ip-172-31-8-101:~/Module-Precimonious/exercise/results$ ls
INVALID_config_simpsons.bc_1.json    INVALID_config_simpsons.bc_4.json    INVALID_config_simpsons.bc_75.json   VALID_config_simpsons.bc_48.json
INVALID_config_simpsons.bc_10.json   INVALID_config_simpsons.bc_40.json   INVALID_config_simpsons.bc_76.json   VALID_config_simpsons.bc_62.json
INVALID_config_simpsons.bc_100.json  INVALID_config_simpsons.bc_41.json   INVALID_config_simpsons.bc_77.json   VALID_config_simpsons.bc_68.json
INVALID_config_simpsons.bc_101.json  INVALID_config_simpsons.bc_42.json   INVALID_config_simpsons.bc_78.json   VALID_config_simpsons.bc_7.json
INVALID_config_simpsons.bc_102.json  INVALID_config_simpsons.bc_43.json   INVALID_config_simpsons.bc_79.json   VALID_config_simpsons.bc_70.json
INVALID_config_simpsons.bc_103.json  INVALID_config_simpsons.bc_44.json   INVALID_config_simpsons.bc_8.json    VALID_config_simpsons.bc_82.json
INVALID_config_simpsons.bc_104.json  INVALID_config_simpsons.bc_45.json   INVALID_config_simpsons.bc_80.json   VALID_config_simpsons.bc_88.json
INVALID_config_simpsons.bc_105.json  INVALID_config_simpsons.bc_46.json   INVALID_config_simpsons.bc_81.json   VALID_config_simpsons.bc_94.json
INVALID_config_simpsons.bc_106.json  INVALID_config_simpsons.bc_47.json   INVALID_config_simpsons.bc_83.json   config_temp.json
INVALID_config_simpsons.bc_107.json  INVALID_config_simpsons.bc_49.json   INVALID_config_simpsons.bc_84.json   dd2_diff_simpsons.bc.json
INVALID_config_simpsons.bc_108.json  INVALID_config_simpsons.bc_5.json    INVALID_config_simpsons.bc_85.json   dd2_diff_simpsons.bc_0.json
INVALID_config_simpsons.bc_12.json   INVALID_config_simpsons.bc_50.json   INVALID_config_simpsons.bc_86.json   dd2_diff_simpsons.bc_109.json
INVALID_config_simpsons.bc_13.json   INVALID_config_simpsons.bc_51.json   INVALID_config_simpsons.bc_87.json   dd2_diff_simpsons.bc_11.json
INVALID_config_simpsons.bc_14.json   INVALID_config_simpsons.bc_52.json   INVALID_config_simpsons.bc_89.json   dd2_diff_simpsons.bc_15.json
INVALID_config_simpsons.bc_16.json   INVALID_config_simpsons.bc_53.json   INVALID_config_simpsons.bc_9.json    dd2_diff_simpsons.bc_17.json
INVALID_config_simpsons.bc_18.json   INVALID_config_simpsons.bc_54.json   INVALID_config_simpsons.bc_90.json   dd2_diff_simpsons.bc_19.json
INVALID_config_simpsons.bc_20.json   INVALID_config_simpsons.bc_55.json   INVALID_config_simpsons.bc_91.json   dd2_diff_simpsons.bc_2.json
INVALID_config_simpsons.bc_21.json   INVALID_config_simpsons.bc_56.json   INVALID_config_simpsons.bc_92.json   dd2_diff_simpsons.bc_25.json
INVALID_config_simpsons.bc_22.json   INVALID_config_simpsons.bc_57.json   INVALID_config_simpsons.bc_93.json   dd2_diff_simpsons.bc_27.json
INVALID_config_simpsons.bc_23.json   INVALID_config_simpsons.bc_58.json   INVALID_config_simpsons.bc_95.json   dd2_diff_simpsons.bc_38.json
INVALID_config_simpsons.bc_24.json   INVALID_config_simpsons.bc_59.json   INVALID_config_simpsons.bc_96.json   dd2_diff_simpsons.bc_48.json
INVALID_config_simpsons.bc_26.json   INVALID_config_simpsons.bc_6.json    INVALID_config_simpsons.bc_97.json   dd2_diff_simpsons.bc_62.json
INVALID_config_simpsons.bc_28.json   INVALID_config_simpsons.bc_60.json   INVALID_config_simpsons.bc_98.json   dd2_diff_simpsons.bc_68.json
INVALID_config_simpsons.bc_29.json   INVALID_config_simpsons.bc_61.json   INVALID_config_simpsons.bc_99.json   dd2_diff_simpsons.bc_7.json
INVALID_config_simpsons.bc_3.json    INVALID_config_simpsons.bc_63.json   VALID_config_simpsons.bc_0.json      dd2_diff_simpsons.bc_70.json
INVALID_config_simpsons.bc_30.json   INVALID_config_simpsons.bc_64.json   VALID_config_simpsons.bc_109.json    dd2_diff_simpsons.bc_82.json
INVALID_config_simpsons.bc_31.json   INVALID_config_simpsons.bc_65.json   VALID_config_simpsons.bc_11.json     dd2_diff_simpsons.bc_88.json
INVALID_config_simpsons.bc_32.json   INVALID_config_simpsons.bc_66.json   VALID_config_simpsons.bc_15.json     dd2_diff_simpsons.bc_94.json
INVALID_config_simpsons.bc_33.json   INVALID_config_simpsons.bc_67.json   VALID_config_simpsons.bc_17.json     dd2_valid_simpsons.bc.json
INVALID_config_simpsons.bc_34.json   INVALID_config_simpsons.bc_69.json   VALID_config_simpsons.bc_19.json     log.cov
INVALID_config_simpsons.bc_35.json   INVALID_config_simpsons.bc_71.json   VALID_config_simpsons.bc_2.json      log.dd
INVALID_config_simpsons.bc_36.json   INVALID_config_simpsons.bc_72.json   VALID_config_simpsons.bc_25.json     log.txt
INVALID_config_simpsons.bc_37.json   INVALID_config_simpsons.bc_73.json   VALID_config_simpsons.bc_27.json     output.txt
INVALID_config_simpsons.bc_39.json   INVALID_config_simpsons.bc_74.json   VALID_config_simpsons.bc_38.json
```

Step 4: Run Analysis – Output Files

- Open dd2_valid_funarc.bc.json: suggested configuration file in JSON format
- Open dd2_diff_funarc.bc.json: summary of type changes

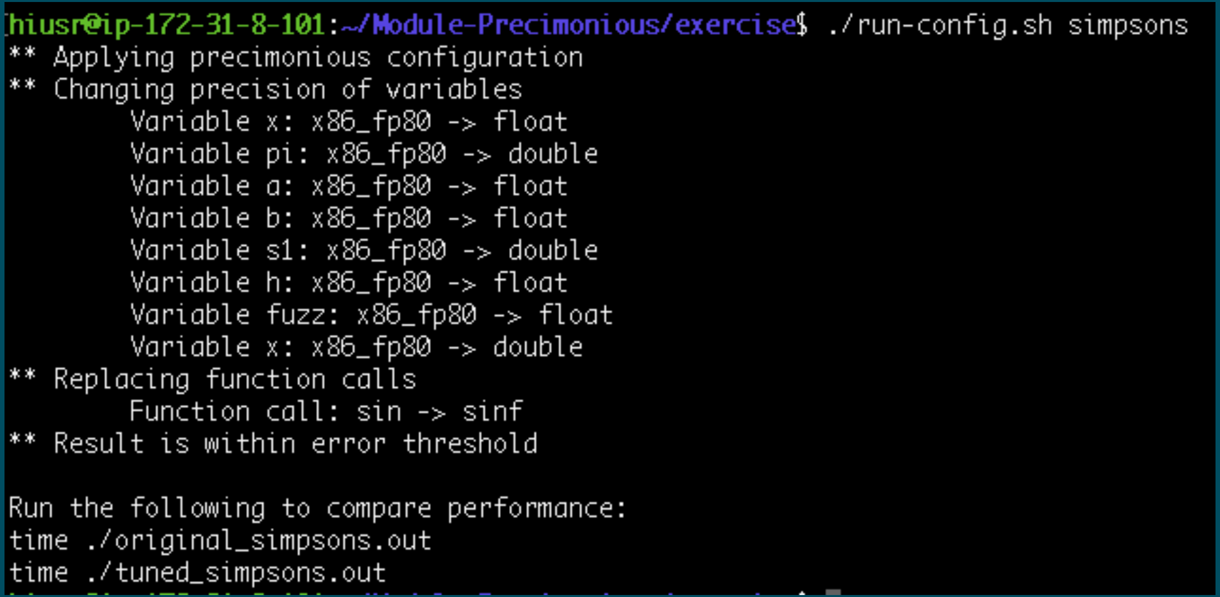


```
localVar: x  at fun longdouble -> float
localVar: pi  at fun longdouble -> double
call: sin at funsin -> sinf
localVar: a  at main longdouble -> float
localVar: b  at main longdouble -> float
localVar: s1  at main longdouble -> double
localVar: h  at main longdouble -> float
localVar: fuzz  at main longdouble -> float
localVar: x  at main longdouble -> double
```

Step 5: Apply Result Configuration & Compare Performance

- Execute :
  - $ cd ..
  - $ ./run-config.sh simpsons

- Execute :
  - $ time ./original_simpsons.out
  - $ time ./tuned_simpsons.out



```
hiusr@ip-172-31-8-101:~/Module-Precimonious/exercise$ ./run-config.sh simpsons
** Applying precimonious configuration
** Changing precision of variables
        Variable x: x86_fp80 -> float
        Variable pi: x86_fp80 -> double
        Variable a: x86_fp80 -> float
        Variable b: x86_fp80 -> float
        Variable s1: x86_fp80 -> double
        Variable h: x86_fp80 -> float
        Variable fuzz: x86_fp80 -> float
        Variable x: x86_fp80 -> double
** Replacing function calls
        Function call: sin -> sinf
** Result is within error threshold

Run the following to compare performance:
time ./original_simpsons.out
time ./tuned_simpsons.out
```

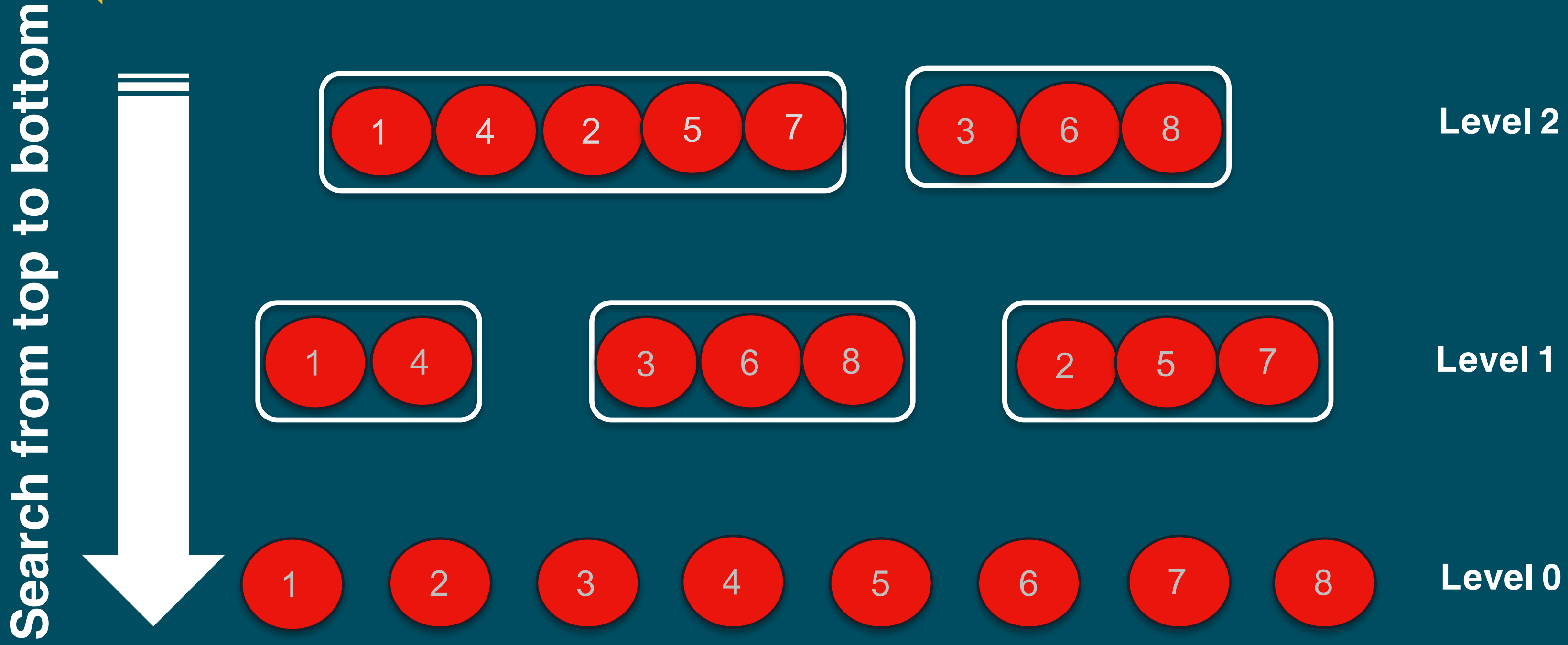
Exercise 2: Run Precimonious on funarc program

- Open exercise-2/funarc.c to see annotated program
- Execute :
  - cd ../exercise-2
  - make clean
  - make
  - ./run-analysis.sh funarc
  - ./run-config.sh funarc
- Open results/dd2_valid_funarc.bc.json to see configuration in JSON format
- Open results/dd2_diff_funarc.bc.json to see difference between original program and proposed configuration

Limitations of Precimonious

- Type configurations rely on inputs tested
  - No guarantees if worse conditioned input
  - Could be combined with input generation tools (e.g., S3FP)
- Getting trapped in local minimum
- Analysis scalability
  - Approach does not scale well for long-running applications
  - Need to reduce search space and reduce number of runs
  - Check out our follow up work on Blame Analysis (ICSE'16)
- Analysis effectiveness
  - Approach does not exploit relationship among variables
  - Check out our follow up work on HiFPTuner (ISSTA'18)

HiFPTuner: exploiting the community structure of the variables in precision tuning



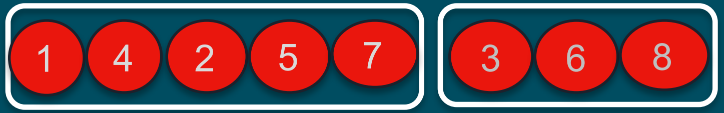
Search from top to bottom

Level 2

Level 1

Level 0

Search Faster and Reach Better Configurations



Same type for variables in one community

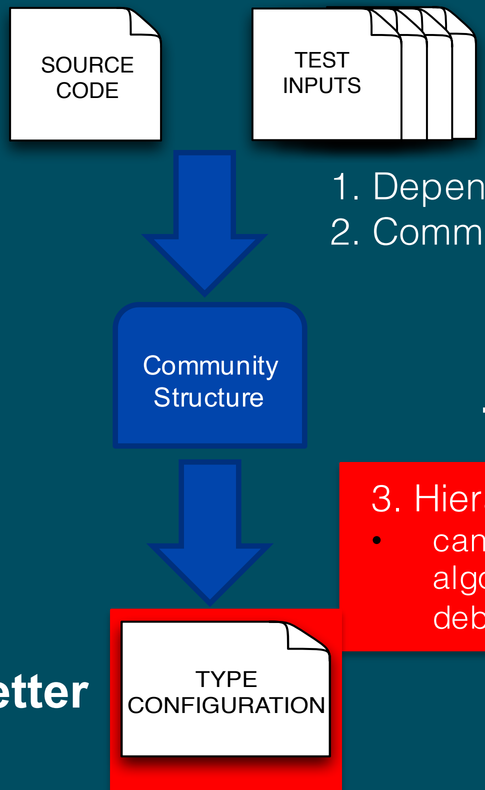
- Decreased search space - only exploring the configurations which satisfy the community structure of the variables
- Better configurations for speed-up - dependent variables are assigned with the same type which avoids type casts

One type per variable

- Exponential number of type configurations with regard to the number of variables – large search space
- Trapped in local optimum introducing many type casts

HiFPTuner

Hierarchical Floating-Point Precision Tuning



SOURCE CODE

TEST INPUTS

1. Dependence analysis
2. Community detection

Community Structure

faster

3. Hierarchical Search
   - can be combined with any base search algorithm such as binary search or delta-debugging algorithm

better

TYPE CONFIGURATION

Source : https://github.com/ucd-plse/HiFPTuner

Exercise

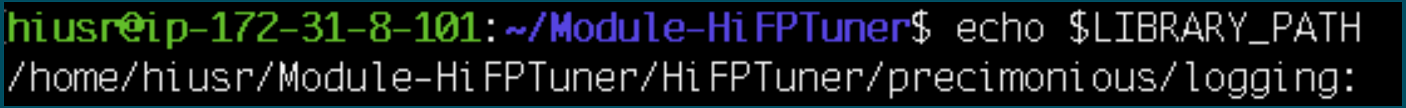
```
$ cd Module-HiFPTuner
```

Build HiFPTuner

```
$ source ./setup.sh
```

Check the environment variable

```
$ echo $LIBRARY_PATH
```



```
hiusr@ip-172-31-8-101:~/Module-HiFPTuner$ echo $LIBRARY_PATH
/home/hiusr/Module-HiFPTuner/HiFPTuner/precimonious/logging:
```

Step 1: Annotate Program and Compile it to *bitcode* File

```
$ cd exercise
$ ls
```

Source: *simpsons.c* (annotated with accuracy logging/checking functions and timing code shown before)

Compile *simpsons.c* to LLVM *bitcode* file

```
$ make clean; make
```

It generates *simpson.bc* and the executable *original_simpsons.out*

Note: *original_simpsons.out* will be used later for performance comparison

Step 2: Run HiFPTuner

Run HiFPTuner on *simpsons.bc*

```
$ ./run-hifptuner.sh simpsons
```

Output files:

./results-hifptuner

result file

- dd2_valid_simpsons.bc.json : the precision configuration file

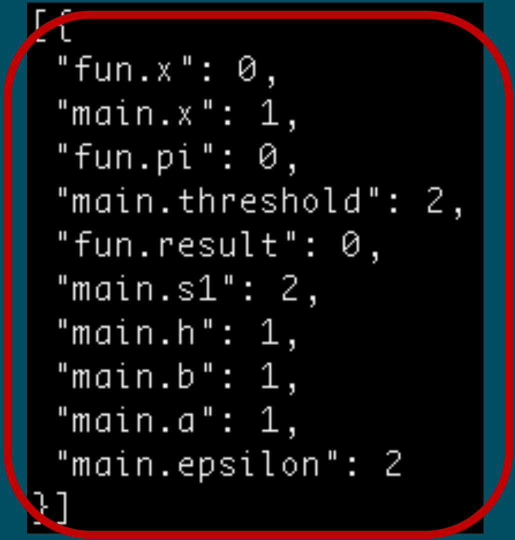
log files

- log.txt, log.dd : search log of HiFPTuner
- sorted_partition.json : the community structure of floating-point variables
- auto-tuning_analyze_time.txt : dependence analysis time
- auto-tuning_config_time.txt : community detecton time

Step 2: Run HiFPTuner – community detection

Input : *varDepPairs_pro.json, edgeProfilingOut.json*

Output : *sorted_partition.json*



```
[{
 "fun.x": 0,
 "main.x": 1,
 "fun.pi": 0,
 "main.threshold": 2,
 "fun.result": 0,
 "main.s1": 2,
 "main.h": 1,
 "main.b": 1,
 "main.a": 1,
 "main.epsilon": 2
}]
```

Hierarchy height : 1

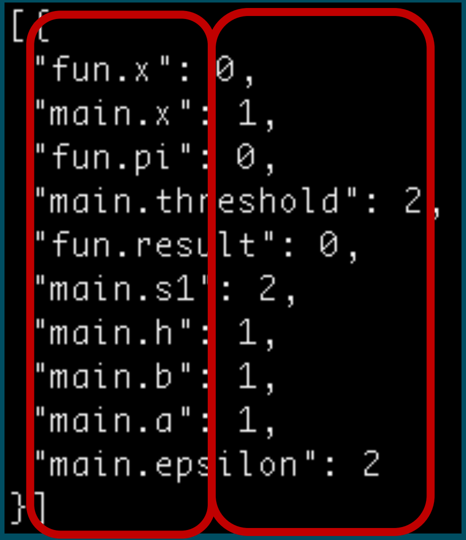
Step 2: Run HiFPTuner – community detection

Input : *varDepPairs_pro.json, edgeProfilingOut.json*

Output : *sorted_partition.json*

Hierarchy height : 1

Floating-point variable



```
[{
 "fun.x": 0,
 "main.x": 1,
 "fun.pi": 0,
 "main.threshold": 2,
 "fun.result": 0,
 "main.s1": 2,
 "main.h": 1,
 "main.b": 1,
 "main.a": 1,
 "main.epsilon": 2
}]
```

Community number

(sorted in the topological order of dependence)

Step 2: Run HiFPTuner – hierarchical search

Input : *simpsons.bc,*
search_simpsons.json, config_simpsons.json,
sorted_partition.json
Output : *dd2_valid_simpsons.bc.json*

Step 3: Tuned Program VS Original Program

Generate tuned executable *hifptuner_tuned_simpsons.out*

```
$ cd ..
$ ./run-config.sh simpsons
```

Time the execution of the original program and the tuned program and compare the execution time

```
$ time ./original_simpsons.out
$ time ./hifptuner_tuned_simpsons.out
```

0m1.710s VS 0m0.951s

Step 4: HiFPTuner VS Precimonious

- Tuned program: which is faster?
- Search time: which explored less configurations?

Step 4: HiFPTuner VS Precimonious : tuned program

Time the execution of the tuned programs of Precimonious and HiFPTuner, and compare the execution time

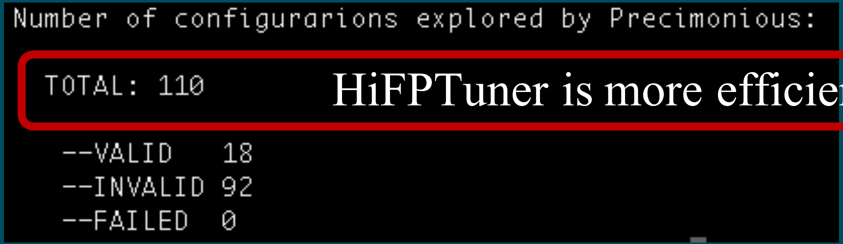
```
$ time ../../Module-Precimonious/exercise/tuned_simpsons.out
$ time ./hifptuner_tuned_simpsons.out
```

0m1.191s VS 0m0.951s: HiFPTuner found better configuration.

Step 4: HiFPTuner VS Precimonious : search time

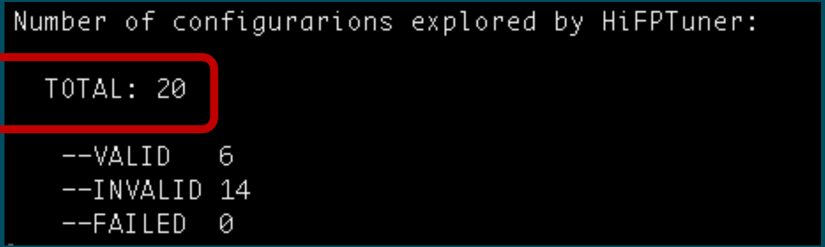
Compare the search effort of Precimonious and HiFPTuner:

```
$ cat ../../Module-Precimonious/exercise/results/log.txt
```



```
Number of configurarions explored by Precimonious:

 TOTAL: 110

   --VALID   18
   --INVALID 92
   --FAILED  0
```

```
$ cat results-hifptuner/log.txt
```



```
Number of configurarions explored by HiFPTuner:

 TOTAL: 20

   --VALID   6
   --INVALID 14
   --FAILED  0
```

HiFPTuner is more efficient.

VALID configuration : accuracy check ✔
INVALID configuration : accuracy check X
FAILED configuration : it crashes

Exercise 2: Run HiFPTuner on funarc program

- Open exercise-2/funarc.c to see annotated program
- Execute :
  - cd ../exercise-2
  - make clean
  - make
  - ./run-hifptuner.sh funarc
  - ./run-config.sh funarc
- Open results-hifptuner/dd2_valid_funarc.bc.json to see configuration in JSON format
- Open results-hifptuner/dd2_diff_funarc.bc.json to see difference between original program and proposed configuration

Run Precimonious or HiFPTuner on Your Program

- Annotate your program with our utility functions

Accuracy log and check

- cov_spec_log: log the accurate result yielded by original precision to file “spec.cov”
- cov_log: log the result of the program in each execution to file “log.cov”
- cov_check: check whether the result in current execution satisfies the accuracy criterion

Performance log

- log the execution time of the code of interest to the file: “score.cov”

- Compile your program to LLVM *bitcode*

WLLVM, https://github.com/travitch/whole-program-llvm - for large projects

- Download the precision tuning docker image

“$ docker pull ucdavisplse/precision-tuning”, https://github.com/ucd-plse/tutorial-precision-tuning

Collaborators

University of California, Berkeley

Cuong Nguyen

Diep Nguyen

Ben Mehne

James Demmel

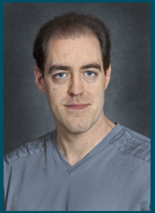
William Kahan

Koushik Sen

Lawrence Berkeley National Lab

Costin Iancu

David Bailey


Wim Lavrijsen

Oracle

David Hough

Precimonious:
https://github.com/ucd-plse/precimonious

HiFPTuner:
https://github.com/ucd-plse/HiFPTuner

Hui Guo <higuo@ucdavis.edu>

Cindy Rubio-Gonzalez <crubio@ucdavis.edu>

Questions?