Precimonious

Tuning Assistant for Floating-Point Precision

Ignacio Laguna, Harshitha Menon, Tristan Vanderbruggen
Lawrence Livermore National Laboratory

Michael Bentley, Ian Briggs, Ganesh Gopalakrishnan
University of Utah

Cindy Rubio-González
University of California at Davis

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

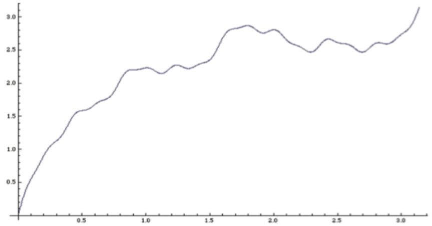
- Common practice: use highest available precision
  - Disadvantage: more expensive!

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

Example: Arc Length

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

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



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

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

	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;
}
```

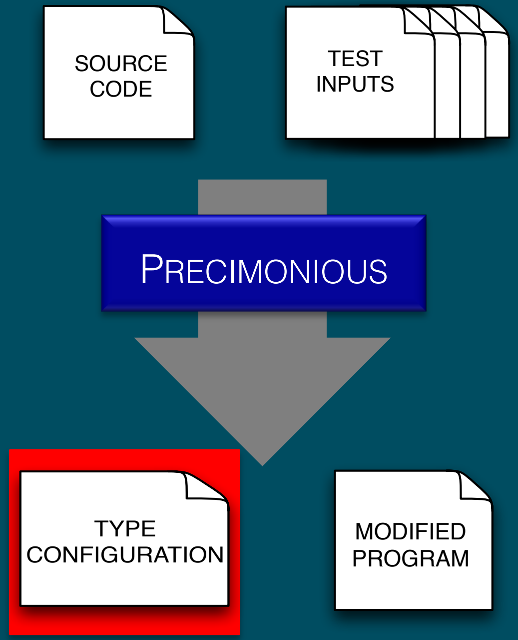
Mixed Precision Program

Precimonious

“Parsimonious or Frugal with Precision”

Dynamic Analysis for Floating-Point Precision Tuning

Annotated with error threshold



Less Precision

Speedup

Modified program in executable format

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?

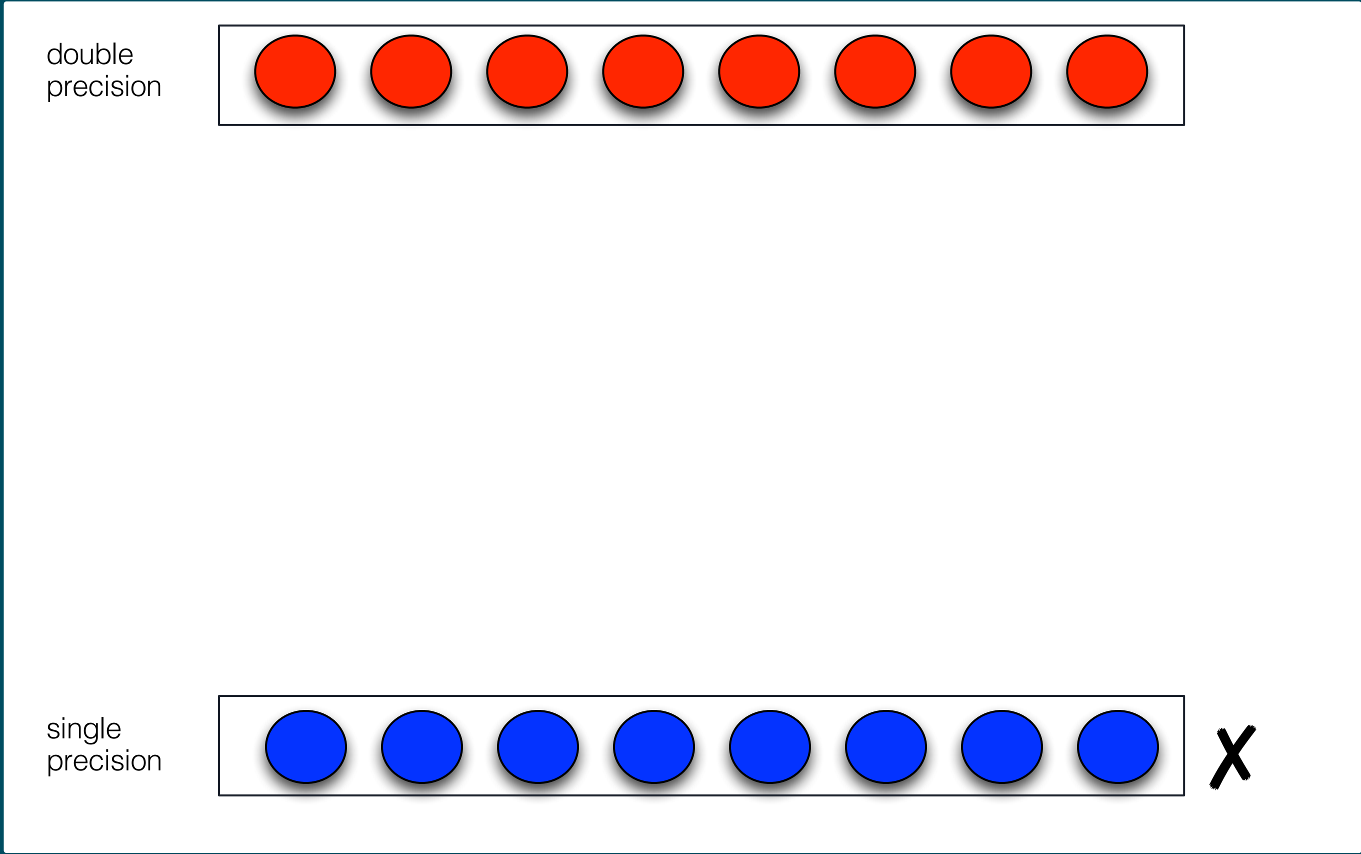
Automated

Specified by the user

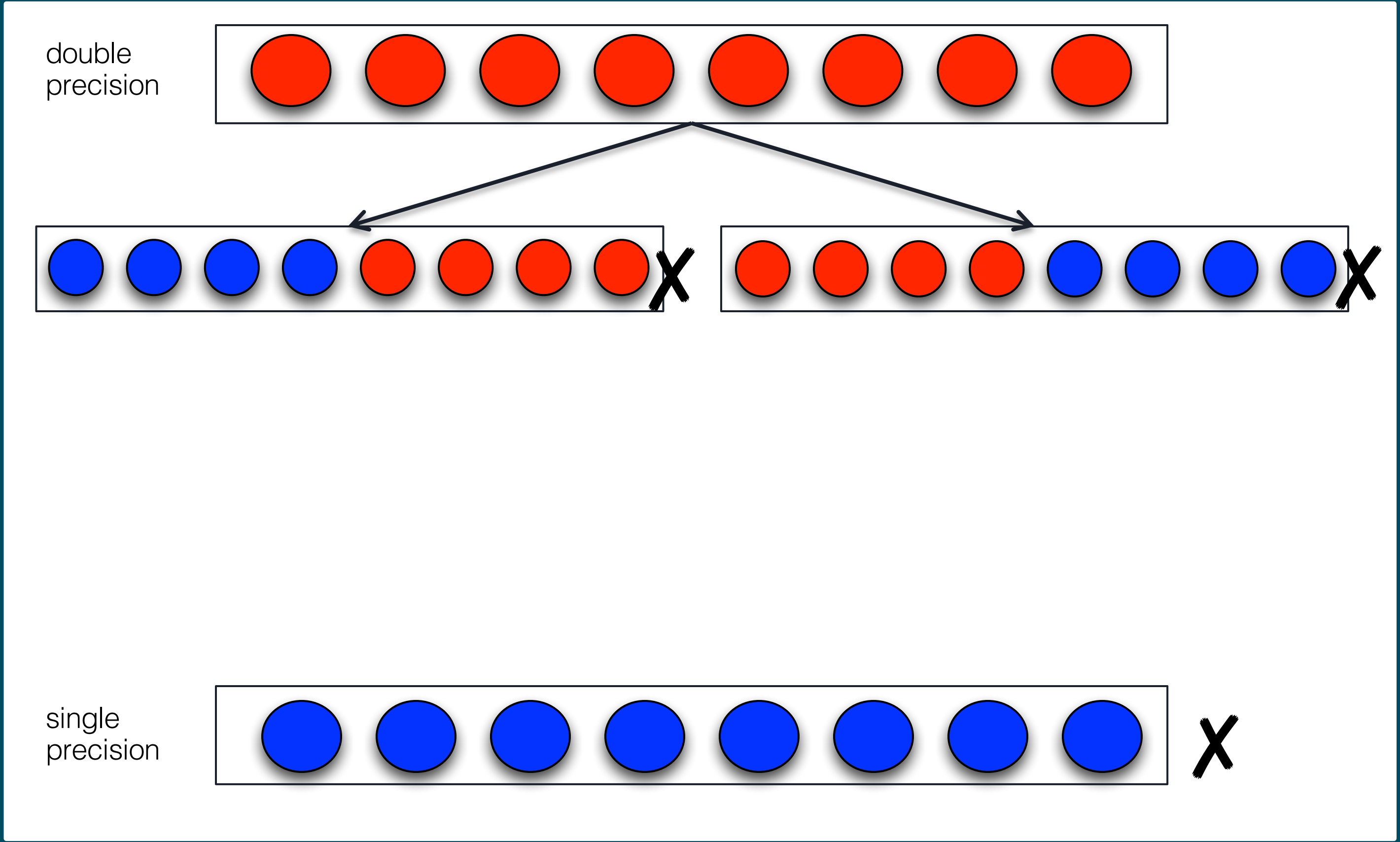
Precimonious Search Algorithm

- Based on Delta Debugging Algorithm (TSE'02)
- Our definition of a change
  - Lowering the precision of a floating-point variable in the program
    - Example: double x -> float x
- Main idea
  - We can do better than making a change at the time
  - Start by dividing the change set into two equally sized subsets
  - Narrow the search to the subset that satisfies the success criteria
  - Otherwise, increase the number of subsets
- Our success criteria
  - Resulting program produces an answer within the given error threshold
  - Resulting program is faster than original program
- Find local minimum
  - Lowering the precision of any one more variable violates the success criteria

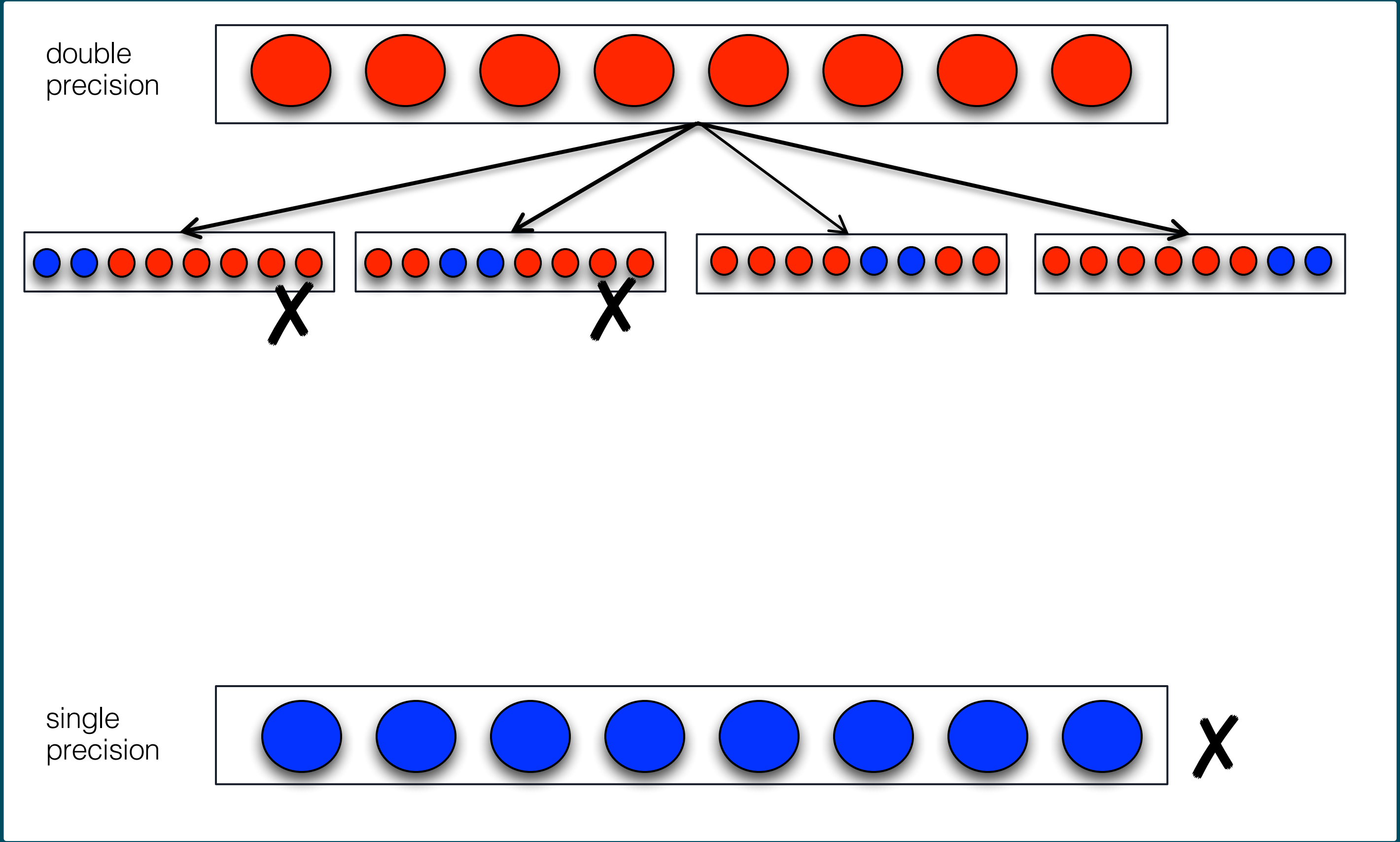
Searching for Type Configuration



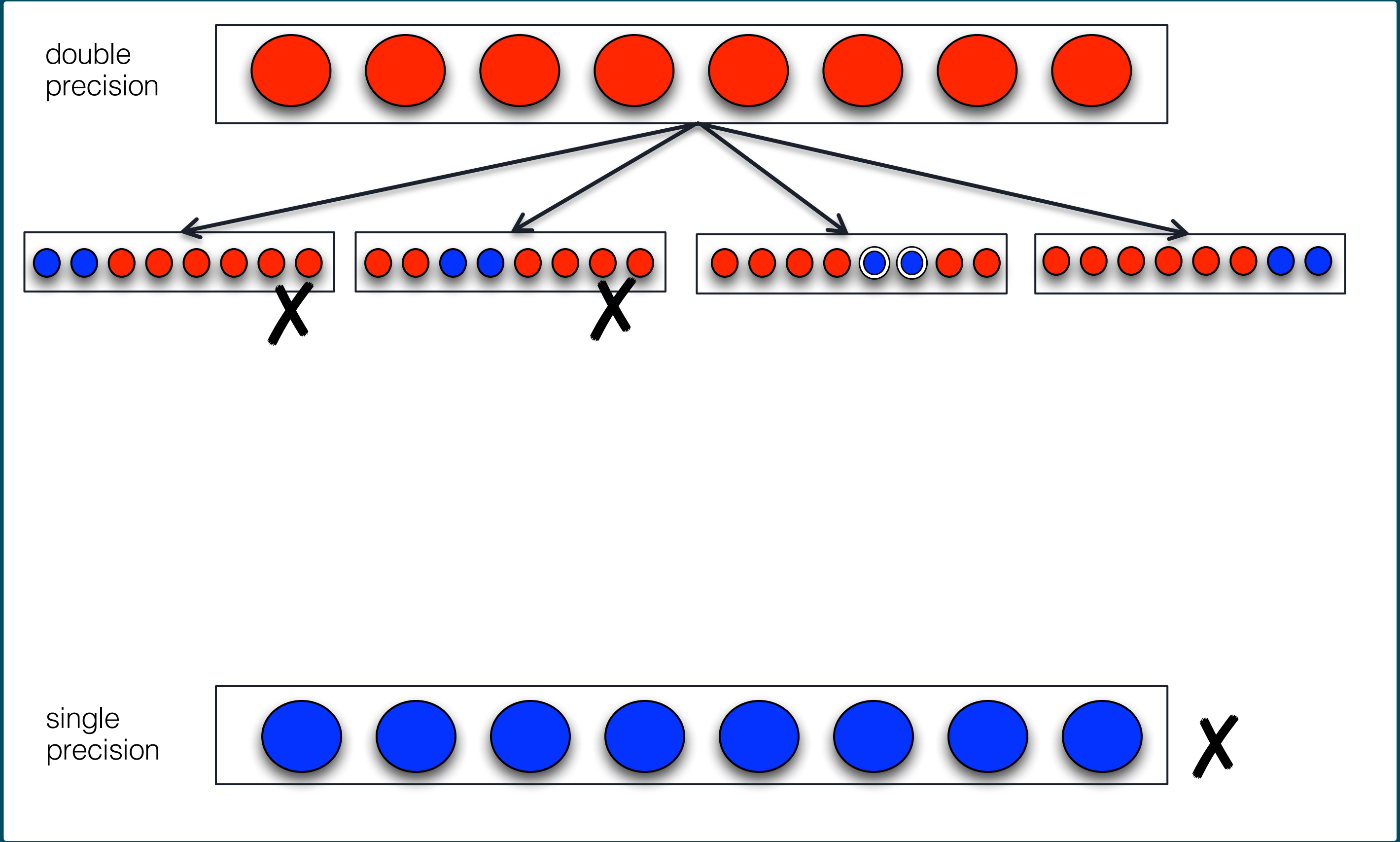
Searching for Type Configuration



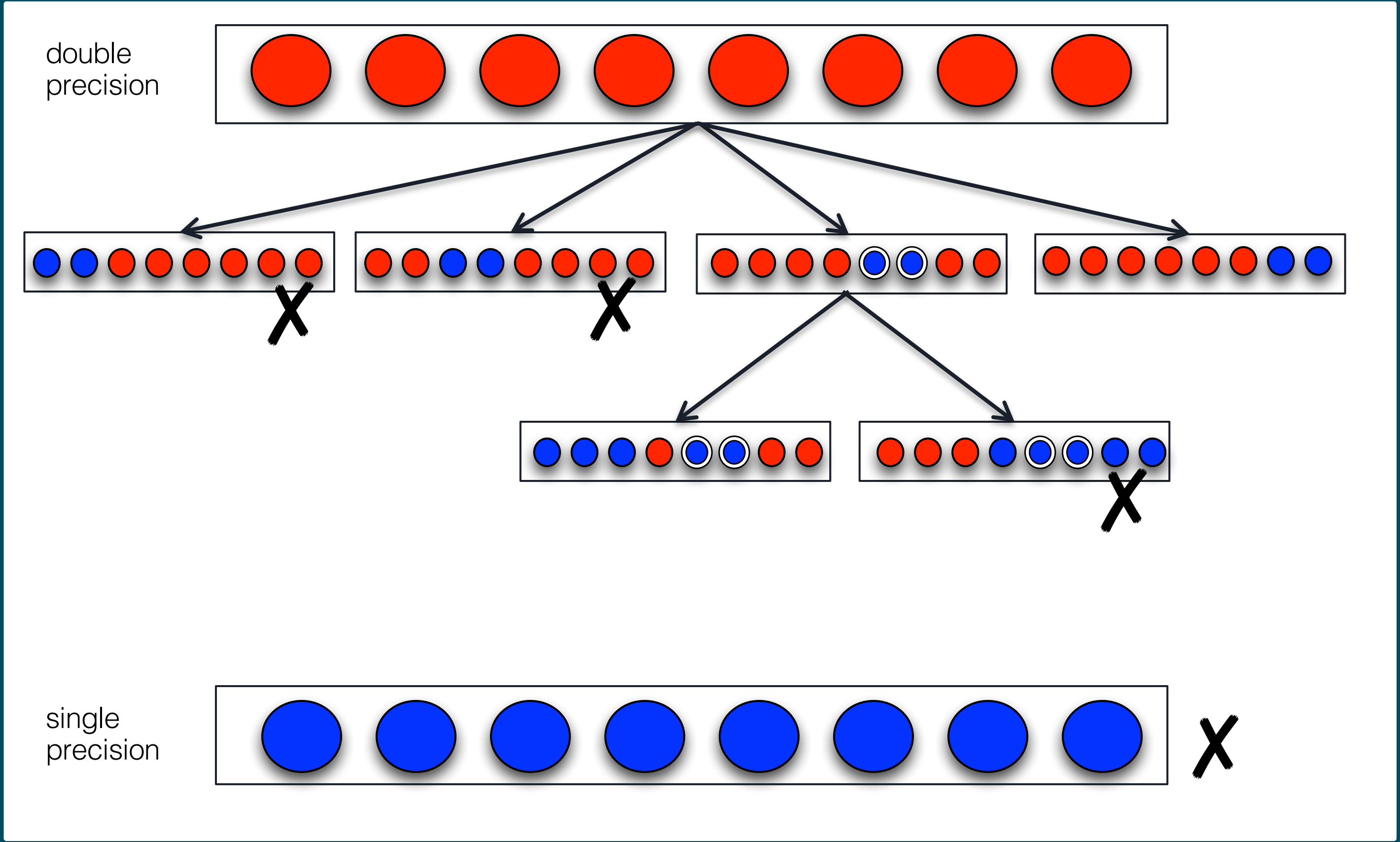
Searching for Type Configuration



Searching for Type Configuration

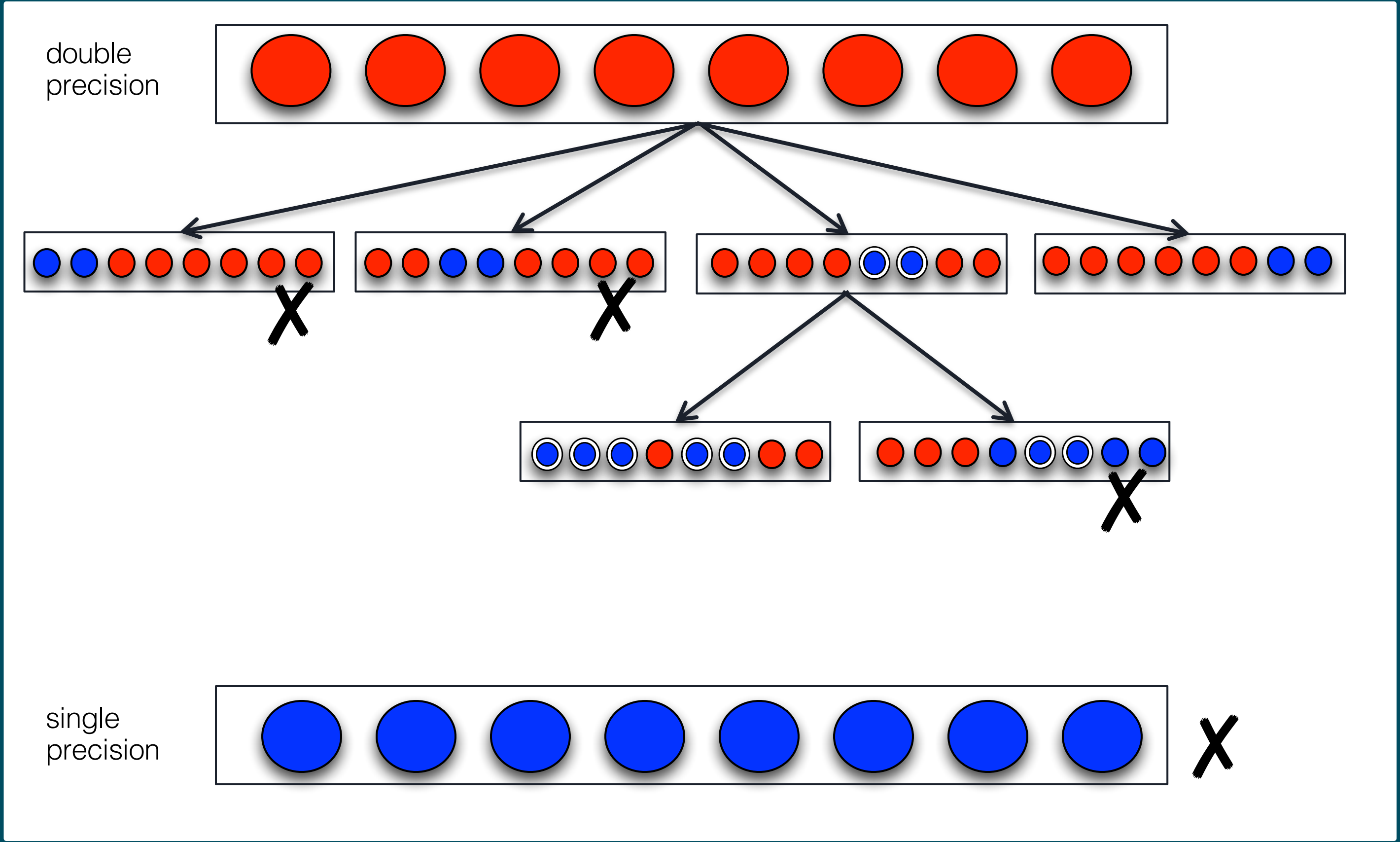


Searching for Type Configuration

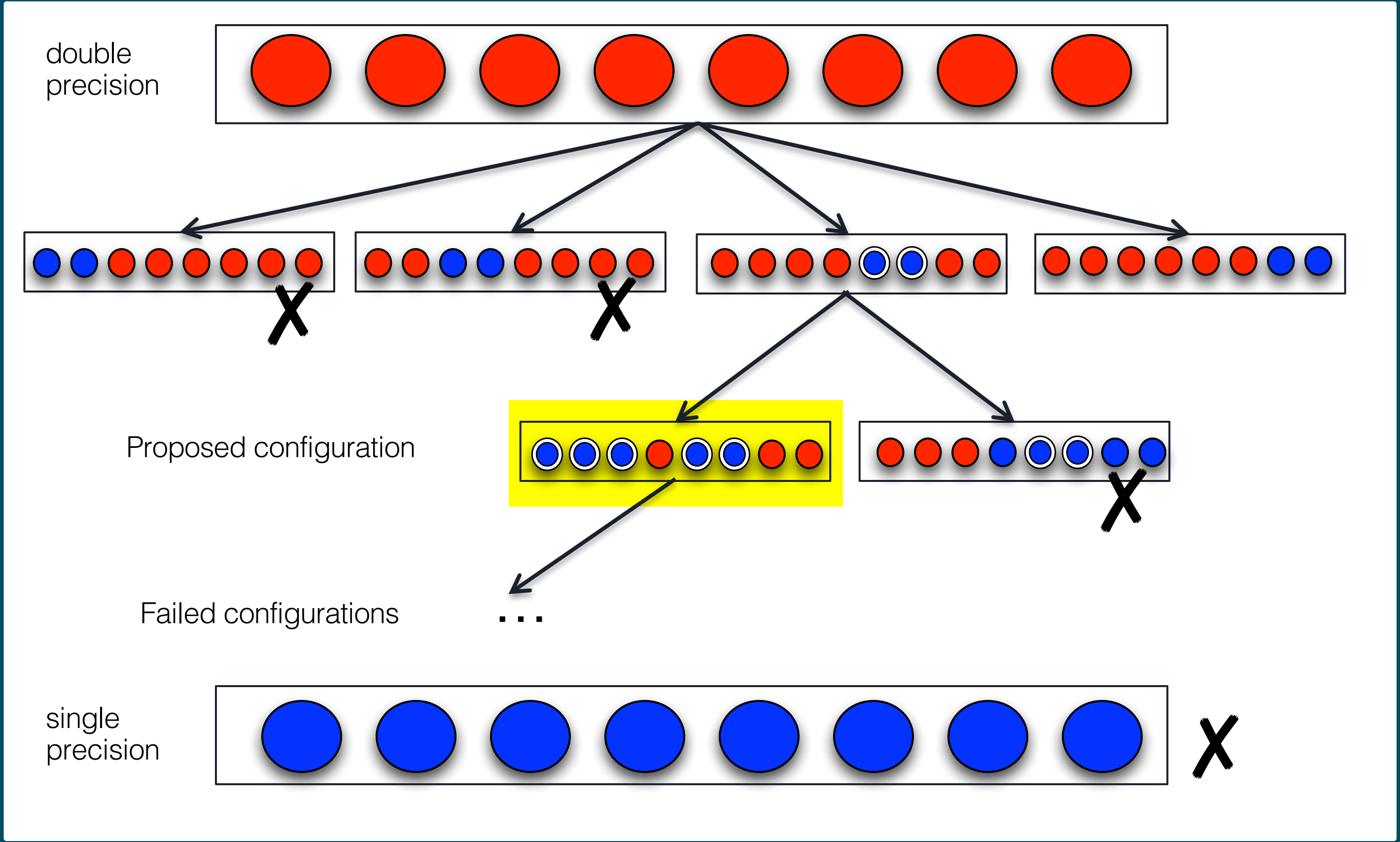


http://fpanalysistools.org/

Searching for Type Configuration



Searching for Type Configuration



http://fpanalysistools.org/

Applying Type Configuration

- Automatically generate program variants
  - Reflect type configurations produced by the algorithm
- Intermediate representation
  - LLVM IR
- Transformation rules for each LLVM instruction
  - alloca, load, store, fadd, fsub, fpext, fptrunc, etc.
  - Changes equivalent to modifying the program at the source level
  - Clang plugin to provide modified source code (not discussed today)
- Able to run resulting modified program
  - Evaluate type configuration: accuracy & performance

Limitations

- 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)

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

Questions?

Exercises

Exercises with Precimonious

1. Run Precimonious on sample program `funarc`
2. Run Precimonious on sample program `simpsons`

Directory Structure

```
/Module-Precimonious
     |---/exercise-1
     |---/exercise-2
```

Exercise 1

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-1
  - $ ls

The program we will tune:

```
root@2b744b834ee7:~/Module-Precimonious/exercise-1# ls
Makefile           funarc.c           reference          run-dependencies.sh
exclude.txt        include.txt        run-analysis.sh    spec.cov
exclude_local.txt  include_global.txt run-config.sh
```

- Open funarc.c file

Accuracy logging & checking

Performance logging

```
/***** BEGIN PRECIMONIOUS ACCURACY and PERFORMANCE LOGGING ******/
threshold = result*pow(10, epsilon);

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

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

Step 3: Compile Program with Clang

- Execute :
  - $ make clean
  - $ make

```
root@2b744b834ee7:~/Module-Precimonious/exercise-1# make
/root/llvm-3.0/bin/clang -emit-llvm -c -I/root/Module-Precimonious/precimonious/logging/ -Wno-unused-value funarc.c -o temp_funarc.bc
/root/llvm-3.0/bin/clang -emit-llvm -c /root/Module-Precimonious/precimonious/logging//cov_checker.c -o cov_checker.bc
/root/llvm-3.0/bin/clang -emit-llvm -c /root/Module-Precimonious/precimonious/logging//timers.c -o timers.bc
/root/llvm-3.0/bin/clang -emit-llvm -c /root/Module-Precimonious/precimonious/logging//cov_serializer.c -o cov_serializer.bc
/root/llvm-3.0/bin/clang -emit-llvm -c /root/Module-Precimonious/precimonious/logging//cov_log.c -o cov_log.bc
/root/llvm-3.0/bin/clang -emit-llvm -c /root/Module-Precimonious/precimonious/logging//cov_rand.c -o cov_rand.bc
/root/llvm-3.0/bin/llvm-link -o funarc.bc temp_funarc.bc cov_checker.bc cov_serializer.bc cov_log.bc cov_rand.bc timers.bc
/root/llvm-3.0/bin/opt -O2 funarc.bc -o original_funarc.bc
/root/llvm-3.0/bin/llc original_funarc.bc -o original_funarc.s
/root/llvm-3.0/bin/clang original_funarc.s -lm -o original_funarc.out
root@2b744b834ee7:~/Module-Precimonious/exercise-1#
```

- Creates LLVM bitcode file and optimized executable for later use

```
root@2b744b834ee7:~/Module-Precimonious/exercise-1# ls
Makefile            exclude_local.txt    original_funarc.out   spec.cov
cov_checker.bc      funarc.bc            original_funarc.s     temp_funarc.bc
cov_log.bc          funarc.c             reference             timers.bc
cov_rand.bc         include.txt          run-analysis.sh
cov_serializer.bc   include_global.txt   run-config.sh
exclude.txt         original_funarc.bc   run-dependencies.sh
root@2b744b834ee7:~/Module-Precimonious/exercise-1#
```

Step 4: Run Analysis on Program

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

Sample output:

```
** Exploring configuration #108
** Changing precision of variables
        Variable t1: x86_fp80 -> double
        Variable d1: x86_fp80 -> float
        Variable s1: x86_fp80 -> double
        Variable t1: x86_fp80 -> double
        Variable t2: x86_fp80 -> double
        Variable h: x86_fp80 -> float
        Variable dppi: x86_fp80 -> float
** Replacing function calls
        Function call: acos -> acosf
        Function call: sqrt -> sqrtf
** Result is NOT within error threshold

** Exploring configuration #109
** Changing precision of variables
        Variable t1: x86_fp80 -> double
        Variable d1: x86_fp80 -> float
        Variable s1: x86_fp80 -> double
        Variable t1: x86_fp80 -> double
        Variable t2: x86_fp80 -> double
        Variable h: x86_fp80 -> double
        Variable dppi: x86_fp80 -> float
** Replacing function calls
        Function call: acos -> acosf
        Function call: sqrt -> sqrtf
** Result is within error threshold

Check dd2_valid_funarc.bc.json for the valid configuration file
root@2b744b834ee7:~/Module-Precimonious/exercise-1#
```

Type changes are listed for each explored configuration

Suggested type configuration

Step 4: Run Analysis – Configuration File

- Open config_funarc.json
- Original type configuration

```
{"config": [
        {"localVar": {
                "function": "fun",
                "name": "t1",
                "type": "longdouble"
        }},
        {"localVar": {
                "function": "fun",
                "name": "d1",
                "type": "longdouble"
        }},
        {"localVar": {
                "function": "fun",
                "name": "x",
                "type": "longdouble"
        }},
        {"call": {
                "id": "24",
                "function": "fun",
                "name": "sin",
                "switch": "sin",
                "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": "t1",
                "type": ["float", "double", "longdouble"]
        }},
        {"localVar": {
                "function": "fun",
                "name": "d1",
                "type": ["float", "double", "longdouble"]
        }},
        {"localVar": {
                "function": "fun",
                "name": "x",
                "type": ["float", "double", "longdouble"]
        }},
        {"call": {
                "id": "24",
                "function": "fun",
                "name": "sin",
                "switch": ["sinf","sin"],
                "type": [["float","float"], ["double","double"]]
        }},
```

Step 4: Run Analysis – Output Files

- Execute :
  - $ cd results
  - $ ls

```
root@2b744b834ee7:~/Module-Precimonious/exercise-1/results# ls
FAIL1_config_funarc.bc_3.json      INVALID_config_funarc.bc_38.json  INVALID_config_funarc.bc_72.json   VALID_config_funarc.bc_17.json
INVALID_config_funarc.bc_1.json    INVALID_config_funarc.bc_39.json  INVALID_config_funarc.bc_73.json   VALID_config_funarc.bc_19.json
INVALID_config_funarc.bc_10.json   INVALID_config_funarc.bc_4.json   INVALID_config_funarc.bc_74.json   VALID_config_funarc.bc_2.json
INVALID_config_funarc.bc_100.json  INVALID_config_funarc.bc_40.json  INVALID_config_funarc.bc_75.json   VALID_config_funarc.bc_25.json
INVALID_config_funarc.bc_101.json  INVALID_config_funarc.bc_41.json  INVALID_config_funarc.bc_76.json   VALID_config_funarc.bc_27.json
INVALID_config_funarc.bc_102.json  INVALID_config_funarc.bc_43.json  INVALID_config_funarc.bc_77.json   VALID_config_funarc.bc_42.json
INVALID_config_funarc.bc_103.json  INVALID_config_funarc.bc_44.json  INVALID_config_funarc.bc_78.json   VALID_config_funarc.bc_54.json
INVALID_config_funarc.bc_104.json  INVALID_config_funarc.bc_45.json  INVALID_config_funarc.bc_79.json   VALID_config_funarc.bc_60.json
INVALID_config_funarc.bc_105.json  INVALID_config_funarc.bc_46.json  INVALID_config_funarc.bc_8.json    VALID_config_funarc.bc_7.json
INVALID_config_funarc.bc_106.json  INVALID_config_funarc.bc_47.json  INVALID_config_funarc.bc_80.json   VALID_config_funarc.bc_70.json
INVALID_config_funarc.bc_107.json  INVALID_config_funarc.bc_48.json  INVALID_config_funarc.bc_81.json   VALID_config_funarc.bc_82.json
INVALID_config_funarc.bc_108.json  INVALID_config_funarc.bc_49.json  INVALID_config_funarc.bc_83.json   config_temp.json
INVALID_config_funarc.bc_12.json   INVALID_config_funarc.bc_5.json   INVALID_config_funarc.bc_84.json   dd2_diff_funarc.bc.json
INVALID_config_funarc.bc_13.json   INVALID_config_funarc.bc_50.json  INVALID_config_funarc.bc_85.json   dd2_diff_funarc.bc_0.json
INVALID_config_funarc.bc_14.json   INVALID_config_funarc.bc_51.json  INVALID_config_funarc.bc_86.json   dd2_diff_funarc.bc_109.json
INVALID_config_funarc.bc_16.json   INVALID_config_funarc.bc_52.json  INVALID_config_funarc.bc_87.json   dd2_diff_funarc.bc_11.json
INVALID_config_funarc.bc_18.json   INVALID_config_funarc.bc_53.json  INVALID_config_funarc.bc_88.json   dd2_diff_funarc.bc_15.json
INVALID_config_funarc.bc_20.json   INVALID_config_funarc.bc_55.json  INVALID_config_funarc.bc_89.json   dd2_diff_funarc.bc_17.json
INVALID_config_funarc.bc_21.json   INVALID_config_funarc.bc_56.json  INVALID_config_funarc.bc_9.json    dd2_diff_funarc.bc_19.json
INVALID_config_funarc.bc_22.json   INVALID_config_funarc.bc_57.json  INVALID_config_funarc.bc_90.json   dd2_diff_funarc.bc_2.json
INVALID_config_funarc.bc_23.json   INVALID_config_funarc.bc_58.json  INVALID_config_funarc.bc_91.json   dd2_diff_funarc.bc_25.json
INVALID_config_funarc.bc_24.json   INVALID_config_funarc.bc_59.json  INVALID_config_funarc.bc_92.json   dd2_diff_funarc.bc_27.json
INVALID_config_funarc.bc_26.json   INVALID_config_funarc.bc_6.json   INVALID_config_funarc.bc_93.json   dd2_diff_funarc.bc_42.json
INVALID_config_funarc.bc_28.json   INVALID_config_funarc.bc_61.json  INVALID_config_funarc.bc_94.json   dd2_diff_funarc.bc_54.json
INVALID_config_funarc.bc_29.json   INVALID_config_funarc.bc_62.json  INVALID_config_funarc.bc_95.json   dd2_diff_funarc.bc_60.json
INVALID_config_funarc.bc_30.json   INVALID_config_funarc.bc_63.json  INVALID_config_funarc.bc_96.json   dd2_diff_funarc.bc_7.json
INVALID_config_funarc.bc_31.json   INVALID_config_funarc.bc_64.json  INVALID_config_funarc.bc_97.json   dd2_diff_funarc.bc_70.json
INVALID_config_funarc.bc_32.json   INVALID_config_funarc.bc_65.json  INVALID_config_funarc.bc_98.json   dd2_diff_funarc.bc_82.json
INVALID_config_funarc.bc_33.json   INVALID_config_funarc.bc_66.json  INVALID_config_funarc.bc_99.json   dd2_valid_funarc.bc.json
INVALID_config_funarc.bc_34.json   INVALID_config_funarc.bc_67.json  VALID_config_funarc.bc_0.json      log.csv
INVALID_config_funarc.bc_35.json   INVALID_config_funarc.bc_68.json  VALID_config_funarc.bc_109.json    log.dd
INVALID_config_funarc.bc_36.json   INVALID_config_funarc.bc_69.json  VALID_config_funarc.bc_11.json     output.txt
INVALID_config_funarc.bc_37.json   INVALID_config_funarc.bc_71.json  VALID_config_funarc.bc_15.json
root@2b744b834ee7:~/Module-Precimonious/exercise-1/results#
```

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: d1  at fun longdouble -> float
localVar: s1  at main longdouble -> double
localVar: t1  at main longdouble -> double
localVar: t2  at main longdouble -> double
localVar: h  at main longdouble -> double
localVar: dppi  at main longdouble -> float
call: acos at mainacos -> acosf
call: sqrt at mainsqrt -> sqrtf
```

Step 5: Apply Result Configuration & Compare Performance

- Execute :
  - $ ./run-config.sh funarc

- Execute :
  - $ time ./original_funarc.out
  - $ time ./tuned_funarc.out

```
root@2b744b834ee7:~/Module-Precimonious/exercise-1# ./run-config.sh funarc
** Applying precimonious configuration
** Changing precision of variables
        Variable t1: x86_fp80 -> double
        Variable d1: x86_fp80 -> float
        Variable s1: x86_fp80 -> double
        Variable t1: x86_fp80 -> double
        Variable t2: x86_fp80 -> double
        Variable h: x86_fp80 -> double
        Variable dppi: x86_fp80 -> float
** Replacing function calls
        Function call: acos -> acosf
        Function call: sqrt -> sqrtf
** Result is within error threshold

Run the following to compare performance:
time ./original_funarc.out
time ./tuned_funarc.out
root@2b744b834ee7:~/Module-Precimonious/exercise-1#
```

Exercise 2

Exercise 2: Run Precimonious on simpsons program

- Open exercise-2/simpsons.c to see annotated program
- Execute :
  - cd ../exercise-2
  - make clean
  - make
  - ./run-analysis.sh simpsons
  - ./run-config.sh simpsons
- Open results/dd2_valid_simpsons.bc.json to see configuration in JSON format
- Open results/dd2_diff_simpsons.bc.json to see difference between original program and proposed configuration

Collaborators

University of California, Berkeley


Cuong Nguyen


Diep Nguyen


Ben Mehne


James Demmel

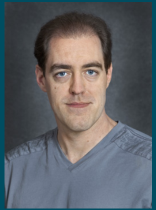

William Kahan


Koushik Sen

Lawrence Berkeley National Lab


Costin Iancu


David Bailey


Wim Lavrijsen

Oracle


David Hough

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

Questions?