Pin

Alex Jackson
Jack Ivy

GOAL

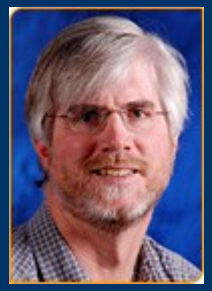
- What is Pin?
  - “A tool for dynamic instrumentation of programs”
  - Instrumentation:
    - Monitoring and measuring a program's performance
- Pin allows us to monitor performance related information as a program is executing

Faces of Pin

Chi-Kung Luk

Harish Patil

Steven Wallace



Geoff Lowney

Robert Cohn

Kim Hazelwood

Vijay Janapa Reddi

Robert Muth

Artur Klauser

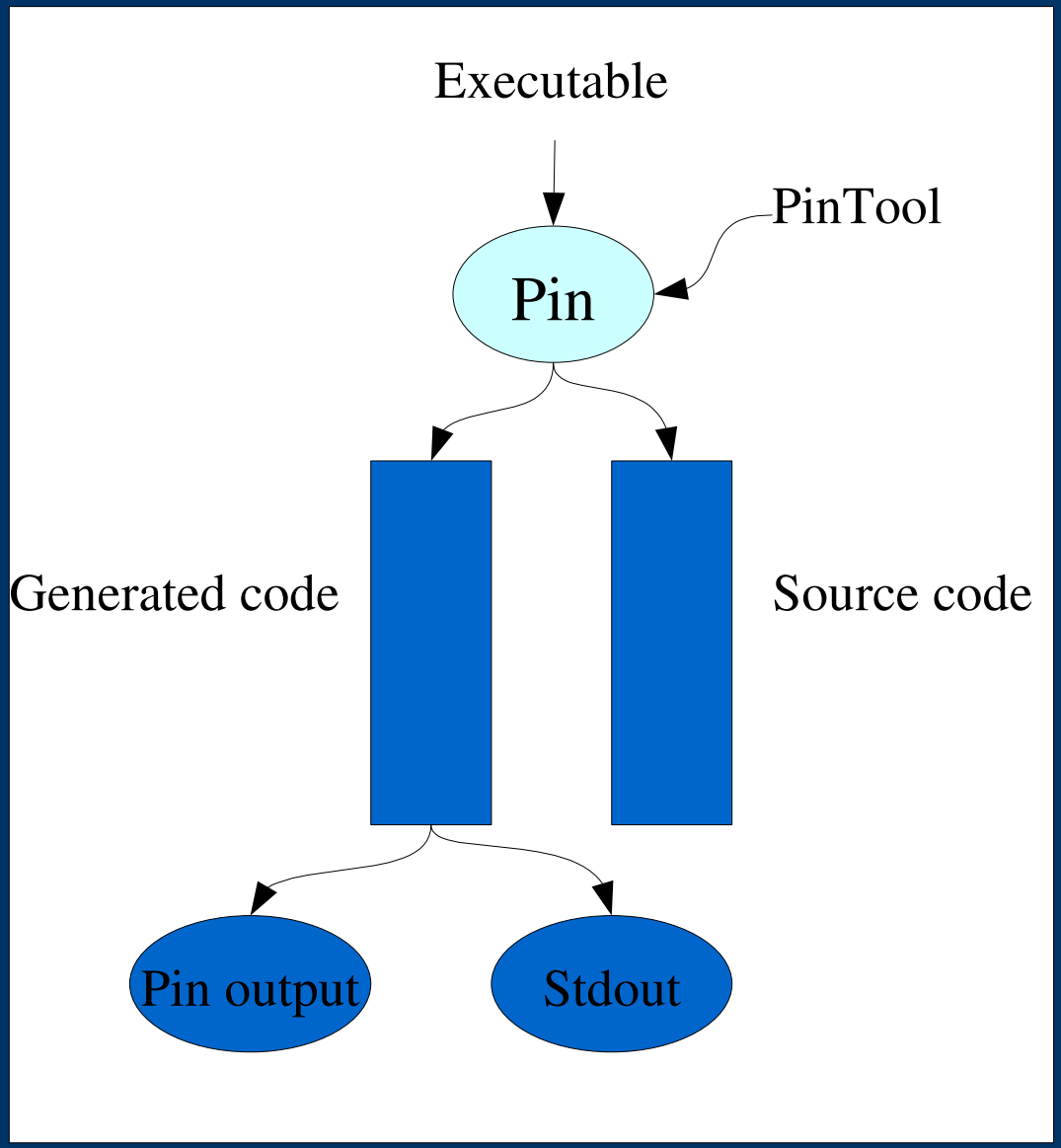
Intended Uses

- We can run Pin with *PinTools*
  - *Specify what information we get back*
- Example PinTools
  - inscount.so
    - Counts instructions
  - memtrace.so
    - Traces memory registers

How Pin works

- Takes an executable as an argument
  - Does not run the executable
    - Generates nearly identical code
      - New code inserted
    - Generated code is run
- Also takes a pintool file
  - Can find one for download or write your own

Pin Flowchart



Pin in Practice

- Pin has been used to analyze performance
  - Pin team found a problem in a compiler
  - Help to get rid of unnecessary or inefficient code
  - Show difference between 64 and 32 bit applications
- Workload Characterization
  - Collect statistical data on programs with large data sets
- Loop-centric profiling
  - Identify opportunities for parallelism

DEMO

Installing Pin

- Navigate to www.pintool.org
- Click on Downloads
- In the Linux listing, click on gcc 4.0 under:
  “IA32 and intel64...”
- Save it to disk

Let Me Help You

- Navigate to Desktop (where your file should have been downloaded)
- Run: **bash /home/courses/cs297/shared/extract.sh**
- ...wait a minute
- OK, now we're ready to rock and roll

Make It Easier on Yourself

- Create an environment variable named EXAMPLES to the folder:
  - **~/pin/source/tools/ManualExamples/obj-ia32**
- Don't forget to run export afterward!
- Create a directory in ~ titled **pinexamples**
  - This will house our output
- Now, let's run some pintools!

Pin Syntax

- **pin -t <pintool (.so file)> -- <program>**
- All of the pintools are in the $EXAMPLES folder we've just created a variable to

Running our first pintool...

- Simple Instruction Count
  - This tool counts the number of instructions executed for a given program
- Output will be placed in our current folder
  - At this time, cd into **~/pinexamples**
- Run: **~/pin/pin -t $EXAMPLES/inscount0.so -- /bin/ls**

Check Results

- Run **cat inscount.out**
- Results are shown!
- Sadly, we were unable to figure out a quick way to redirect output to a different file name
  - i.e. your count will be overwritten every time :(

Very Cool Pintools...

- Run the pintool called **pinatrace.so**
- Then run **more** on the out file
- ...how long do you think it is?
- Run **grep ... | wc** to find out how many reads vs. writes there are

Even Cooler Pintools...

- Run the pintool called **proccount.so**
  - This is a Procedure Instruction Count
- **more** the out file

Try On Non-Unix Command

- Run on **spherecalc**
  - Located in the cs297/shared folder
- Extra credit: run one of the pintools we haven't run yet! :)