

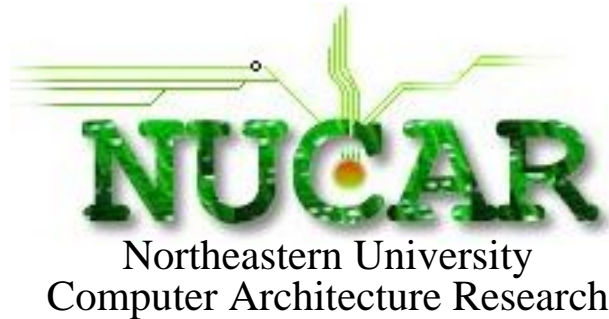
program tracing generation, manipulation, and analysis

Alireza Khalafi

David Morano

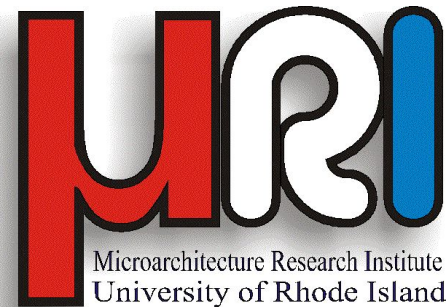
David Kaeli

Northeastern University



Augustus Uht

University of Rhode Island



June 27, 2003

DAM 1

outline



- **introduction**
- **generation**
- **manipulation**
 - filtering
 - conversion
- **analysis**
 - comparison
 - program characterization
 - microarchitectural simulation
 - secondary analysis
- **modularity**
 - code similarity and program pipe fitting
 - static and dynamic modules
- **summary**

introduction



- **process overview**
 - run a program somewhere
 - record various information from the executing program creating a trace of its execution history
 - store the trace for later use
 - transform the trace for input to other programs
 - perform various manipulations on the trace
 - analyze the trace (characterization or simulation)
- **alternatives to tracing**
 - analyze during execution
 - execution based simulation
- **advantages to tracing**
 - same trace analysed multiple times from a single stored copy
- **disadvantages**
 - trace needs to be stored (could be huge)

trace generators (1)



- **Pixie (SGI)**
 - provides
 - ⇒instruction addresses
 - ⇒provides partial memory addresses (lower 24 bits)
 - programs needed to convert to ET format (in order)
 - ⇒pixie2levo
 - ⇒tracepixie
 - runs fairly quickly
 - fairly loer error rate
- **ET (Execution Trace):** is a common trace format that provides plug-in-play among various trace tools

trace generators (2)



- **DBX (SGI)**
 - provides
 - ⇒instruction addresses
 - ⇒instruction destination register values
 - ⇒memory source addresses for loads
 - ⇒memory destination addresses and values for stores
 - programs to convert to ET format (in order)
 - ⇒dbx
 - ⇒dbxvout
 - ⇒stripopgarb
 - ⇒tracedbx
 - runs very (very) slowly -- not really generally useful as a result
 - fairly high error rate -- not generally useful as a result

trace generators (3)



- **SimpleSim**
 - provides anything about program execution that one might want
 - ⇒ instruction addresses
 - ⇒ source register addresses and values
 - ⇒ source memory addresses and values
 - ⇒ destination register addresses and values
 - ⇒ destination memory addresses and values
 - ⇒ system calls and affected memory addresses and values
 - programs to convert to ET format (in order)
 - ⇒ *none* -- produces ET format directly
 - runs fairly quickly
 - zero error rate on some programs
 - some errors on other programs may accumulate and lead to catastrophic program failure

trace generators (4)



- **DamMint**
 - a modified version of MINT that can run more programs without crashing
 - can provide
 - ⇒ instruction addresses
 - ⇒ source memory addresses
 - ⇒ destination register addresses and values
 - ⇒ destination memory addresses and values
 - programs to convert to ET format (in order)
 - ⇒ *none* -- produces ET format directly
 - runs fairly quickly
 - very high error rate but can generally runs many programs

trace manipulation (1)



- **copying and filtering**
 - can filter out some information from an input trace and create a new output trace
 - ⇒system calls (from point of view of programmer)
 - ⇒source or destination register or memory addresses or values
 - primary tool -- TRACECOPY -- ET to ET
 - for system-call filtering
 - ⇒a list of system calls to be filtered must also be supplied
 - ⇒the original program executable (machine object file) is needed for symbol lookup in its symbol table
 - ⇒useful for generating traces that can be compared from generators that do not provide system call information or that provide OS dependent system-call information

trace manipulation (2)



- **conversion to other formats**
 - since there are so many different trace formats, normally conversion would present an $O(n^2)$ number of different conversions
 - but using a common trace format (ET) there only needs to be $2 * n$ conversions to cover all possibilities
 - other formats of interest
 - ⇒ various formats for input to FastLevo
 - conversion to ET has already been seen
 - conversion to other formats (from ET) using TRACEDUMP

trace analysis (1)



- **trace comparison**
 - compare two traces for equality
 - compare two traces for common subsequences
 - useful for verifying program execution or simulation integrity
 - primary tool: TRACECMP
- **program characterization**
 - instruction execution statistics
 - ⇒ counts for loads-stores, ALU, branches
 - values
 - ⇒ register and memory
 - branch behavior
 - subroutine function coverage
 - example tools: TRACESTAT, ICOUNT, FCOUNT

trace analysis (2)



- **microarchitectural simulation**
 - branch predictor behavior
 - cache behavior
 - machine units
 - you name it !
 - example tool: TRACESTAT
- **secondary analysis**
 - primary characterization or simulatorion analyzer creates output data that needs to be further analyzed between different trace-analysis runs
 - example: collating branch predictor results from various predictors or predictor configurations
 - example tool: BPSORT

other trace manipulation



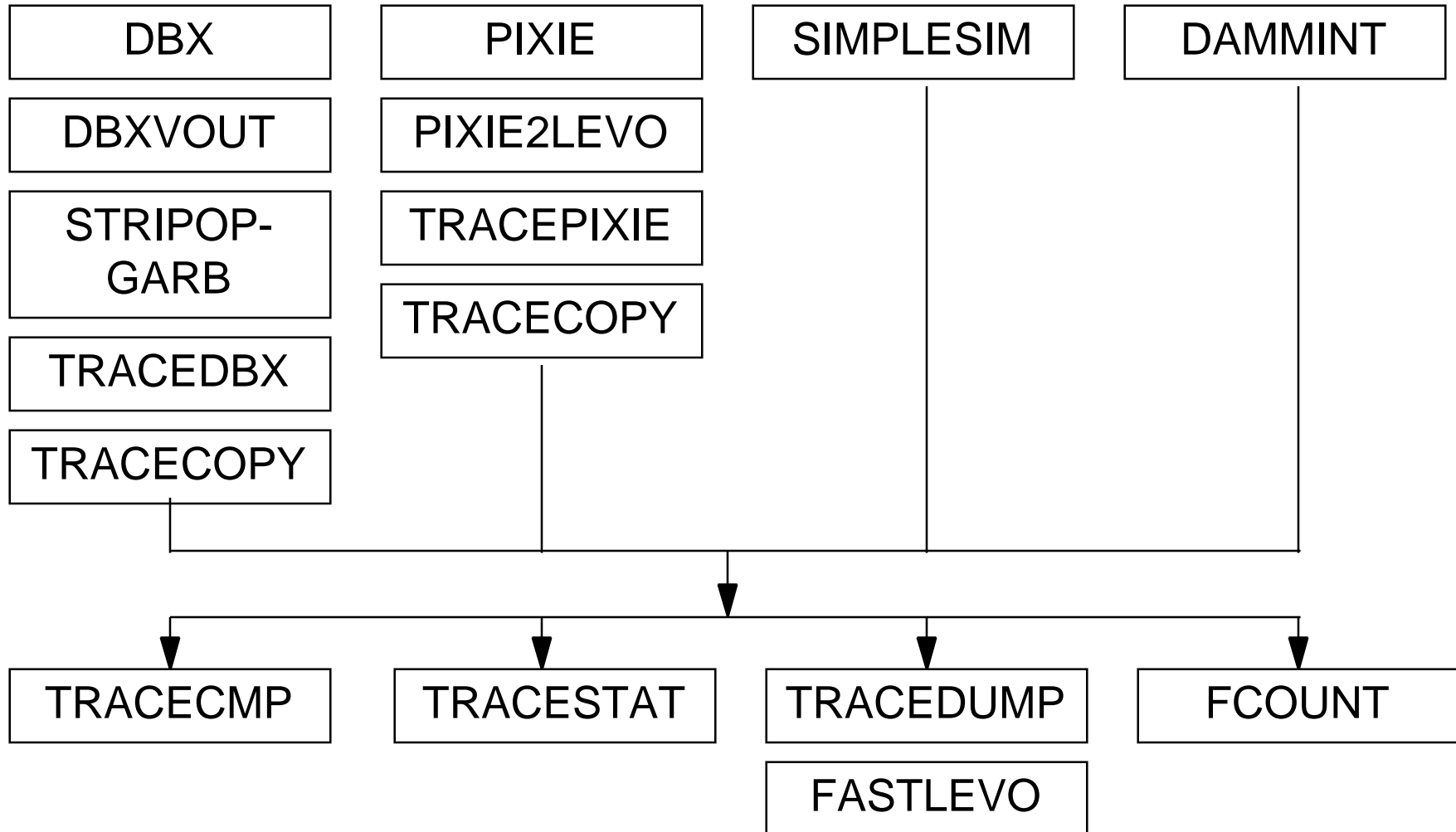
- **creating source operands from destination operands**
 - Why ? -- FastLevo wanted it !
 - can create source register operands from destinations by maintaining a committed copy of the register file and the source operand addresses
 - can create source memory operands from
 - ⇒source operand addresses
 - ⇒destination memory operands
 - ⇒the original program executable object file
 - ⇒maintaining the entire program committed memory state in trace-reader programs (available as a module !)
- **with source operand creation**
 - no need to store source operand values (only source addresses)
 - smaller trace files ! (could be important due to space)

modularity



- **trace generation and reading**
 - using a common trace format (ET) provides opportunities for code modularity and program modularity
 - code for generating and reading traces is similar
 - programs can be fitted together to provide more complex analysis without duplicated specialized code
- **some analysis tools provide for the inclusion of**
 - statically linked code modules
 - ⇒TRACESTAT, LevoSim, SimpleSim, FastLevo
 - dynamically (run-time) linked code modules
 - ⇒TRACESTAT
 - example modules
 - ⇒SS Hammock branch detection and characterization
 - ⇒branch predictor simulation

modularity (typical pipelines)



summary (tool programs)



- **SIMPLESIM** -- simulator and trace generator
- **DAMMINT** -- trace generator
- **DBXVOUT** -- DBX post-processor
- **STRIPOGARB** -- DBXVOUT post-processor
- **TRACEDBX** -- STRIPOGARB post-processor
- **PIXIE2LEVO** -- Pixie post-processor
- **TRACEPIXIE** -- PIXIE2LEVO post-processor
- **TRACECOPY** -- trace filtering and copying
- **TRACECMP** -- trace comparator
- **ICOUNT** -- instruction characterization
- **FCOUNT** -- subroutine function coverage analyzer (also module)
- **TRACESTAT** -- characterization and simulation (also uses dynamic plugins)

summary



- **when developing simulators, verifying program execution integrity is often both difficult and important -- good trace handling tools helps a lot**
- **a common trace format (ET) and modular trace tools allow for more complex trace generations, manipulations, and analyses**
- **I spent about a year handling traces ! :-)**