http://www.archive.ece.cmu.edu/~ece548/tools/dinero/man/dinero.htm

In support of 18-742.

DINERO(1) DINERO(1)

NAME
dinero - cache simulator, version III (Enhanced Version)

SYNOPSIS
dinero -b block_size -u unified_cache_size -i instruction_cache_size -d data_cache_size [other_options]

DESCRIPTION
dinero is a trace-driven cache simulator that supports sub-block placement. Simulation results are determined by the input trace and the cache parameters. A trace is a finite sequence of memory references usually obtained by the interpretive execution of a program or set of programs. Trace input is read by the simulator in din format (described later). Cache parameters, e.g. block size and associativity, are set with command line options (also described later). dinero uses the priority stack method of memory hierarchy simulation to increase flexibility and improve simulator performance in highly associative caches. One can simulate either a unified cache (mixed, data and instructions cached together) or separate instruction and data caches. This version of dinero does not permit the simultaneous simulation of multiple alternative caches.

dinero differs from most other cache simulators because it supports sub-block placement (also known as sector placement) in which address tags are still associated with cache blocks but data is transferred to and from the cache in smaller sub-blocks. This organization is especially useful for on-chip microprocessor caches which have to load data on cache misses over a limited number of pins. In traditional cache design, this constraint leads to small blocks. Unfortunately, a cache with small block devotes much more on-chip RAM to address tags than does one with large blocks. Sub-block placement allows a cache to have small sub-blocks for fast data transfer and large blocks to associate with address tags for efficient use of on-chip RAM.

The enhancements to dinero version III are: (1) dinero is not limited to 32 bit addresses anymore. It is possible to use addresses that are as large as ULONG_MAX on the machine dinero is being run. (2) The word size (one datum referenced by an address) is no longer 4 bytes but is user defined, with 4 bytes being the default. The same word size is used for the instruction and data caches. (3) The bus width is no longer the same as a (sub-)block size but is user defined. The bus width is a proper multiply of the a word size. The bus width cannot be wider than a sub-block when sub-blocks are used, or wider than a block when there are no sub-blocks. (4) The report now includes more information on the bus traffic: how many transfers were initialized (bursts) and how many transfer cycles were required to handle all the traffic. These numbers can be different, for instance: a bus that is half the size of a block. On a read, the entire block is read from memory (1 burst) and takes 2 trasfer cycles (each carrying data the size of half block).

Trace-driven simulation is frequently used to evaluating memory hierarchy performance. These simulations are repeatable and allow cache design

parameters to be varied so that effects can be isolated. They are cheaper than hardware monitoring and do not require access to or the existence of the machine being studied. Simulation results can be obtained in many situations where analytic model solutions are intractable without questionable simplifying assumptions. Further, there does not currently exist any generally accepted model for program behavior, let alone one that is suitable for cache evaluation; workloads in trace-driven simulation are represented by samples of real workloads and contain complex embedded correlations that synthetic workloads often lack. Lastly, a trace-driven simulation is guaranteed to be representative of at least one program in execution.

dinero reads trace input in din format from stdin. A din record is two-tuple label address. Each line of the trace file must contain one din record. The rest of the line is ignored so that comments can be included in the trace file.

The label gives the access type of a reference.

0 read data.
1 write data.
2 instruction fetch.
3 escape record (treated as unknown access type).
4 escape record (causes cache flush).

The address is a hexadecimal byte-address between 0 and ULONG_MAX inclusively.

Cache parameters are set by command line options. Parameters block_size and either unified_cache_size or both data_cache_size and instruction_cache_size must be specified. Other parameters are optional. The suffixes K, M and G multiply numbers by 1024, 1024^2 and 1024^3, respectively.

The following command line options are available:

-b block_size
sets the cache block size in bytes. Must be explicitly set (e.g. -b16).

-u unified_cache_size
sets the unified cache size in bytes (e.g., -u16K). A unified cache, also called a mixed cache, caches both data and instructions. If unified_cache_size is positive, both instruction_cache_size and data_cache_size must be zero. If zero, implying separate instruction and data caches will be simulated, both instruction_cache_size and data_cache_size must be set to positive values. Defaults to 0.

-i instruction_cache_size
sets the instruction cache size in bytes (e.g. -i16384). Defaults to 0 indicating a unified cache simulation. If positive, the data_cache_size must be positive as well.

-d data_cache_size
sets the data cache size in bytes (e.g. -d1M). Defaults to 0 indicating a unified cache simulation. If positive, the instruction_cache_size must be positive as well.

-S subblock_size
sets the cache sub-block size in bytes. Defaults to 0 indicating that sub-block placement is not being used (i.e. -S0).

-a associativity
sets the cache associativity. A direct-mapped cache has associativity 1. A two-way set-associative cache has associativity 2. A fully associative cache has associativity data_cache_size/block_size. Defaults to direct-mapped placement (i.e. -a1).

-r replacement_policy
sets the cache replacement policy. Valid replacement policies are l (LRU), f (FIFO), and r (RANDOM). Defaults to LRU (i.e. -rl).

-f fetch_policy
sets the cache fetch policy. Demand-fetch (d), which fetches blocks that are needed to service a cache reference, is the most common fetch policy. All other fetch policies are methods of prefetching. Prefetching is never done after writes. The prefetch target is determined by the -p option and whether sub-block placement is enabled.

d demand-fetch which never prefetches.
a always-prefetch which prefetches after every demand reference.
m miss-prefetch which prefetches after every demand miss.
t tagged-prefetch which prefetches after the first demand miss to a (sub)-block. The next two prefetch options work only with sub-block placement.
l load-forward-prefetch (sub-block placement only) works like prefetch-always within a block, but it will not attempt to prefetch sub-blocks in other blocks.
S sub-block-prefetch (sub-block placement only) works like prefetch-always within a block except when references near the end of a block. At this point sub-block-prefetches references will wrap around within the current block.

Defaults to demand-fetch (i.e. -fd).

-p prefetch_distance
sets the prefetch distance in sub-blocks if sub-block placement is enabled or in blocks if it is not. A prefetch_distance of 1 means that the next sequential (sub)-block is the potential target of a prefetch. Defaults to 1 (i.e. -p1).

-P abort_prefetch_percent
sets the percentage of prefetches that are aborted. This can be used to examine the effects of data references blocking prefetch references from reaching a shared cache. Defaults to no prefetches aborted (i.e. -P0).

-w write_policy
selects one of two the cache write policies. Write-through (w) updates main memory on all writes. Copy-back (c) updates main memory only when a dirty block is replaced or the cache is flushed. Defaults to copy-back (i.e. -wc)

-A write_allocation_policy

selects whether a (sub)-block is loaded on a write reference. Write-allocate (w) causes (sub)-blocks to be loaded on all references that miss. Non-write-allocate (n) causes (sub)-blocks to be loaded only on non-write references that miss. Defaults to write-allocate (i.e. -Aw).

-D debug_flag
used by implementor to debug simulator. A debug_flag of 0 disables debugging; 1 prints the priority stacks after every reference; and 2 prints the priority stacks and performance metrics after every reference. Debugging information may be useful to the user to understand the precise meaning of all cache parameter settings. Defaults to no-debug (i.e. -D0).

-o output_style
sets the output style. Terse-output (0) prints results only at the end of the simulation run. Verbose-output (1) prints results at half-million reference increments and at the end of the simulation run. Bus-output (2) prints an output record for every memory bus transfer. Bus_and_snoop-output (3) prints an output record for every memory bus transfer and clean sub-block that is replaced. Defaults to terse-output (i.e. -o0). For bus-output, each bus record is a six-tuple:

BUS2 are four literal characters to start bus record
access is the access type (r for a bus-read, w for a bus-write, p for a bus-prefetch, s for snoop activity (output style 3 only).
size is the transfer size in bytes
address is a hexadecimal byte-address between 0 and ULONG_MAX inclusively
reference_count is the number of demand references since the last bus transfer
instruction_count is the number of demand instruction fetches since the last bus transfer

-Z skip_count
sets the number of trace references to be skipped before beginning cache simulation. Defaults to none (i.e. -Z0).

-z maximum_count
sets the maximum number of trace references to be processed after skipping the trace references specified by skip_count . Note, references generated by the simulator not read from the trace (e.g. prefetch references) are not included in this count. Defaults to 10 million (i.e. -z10000000).

-Q flush_count
sets the number of references between cache flushes. Can be used to crudely simulate multiprogramming. Defaults to no flushing (i.e. -Q0).

-W word_size
sets the data word size in bytes. The default is 4 bytes, 32bits, (i.e. -W4). A word size is the number of bytes that are referenced by a single address. When a word is smaller than the bus width, several words are read on a fetch to fill up a (sub-)block in the cache. The same word size is used in both instruction and data cahces.

-B bus_width
sets the bus width size in bytes. The width should be a proper multiply of the word size. When sub-blocks are used, the bus width cannot be wider than a sub-block size. When no sub-blocks are defined, the bus width cannot be wider than the block size. The default value for is a sub-block size when sub-blocks are defined or a block size when no sub-blocks are defined.

FILES

doc.h contains additional programmer documentation.

SEE ALSO

Mark D. Hill and Alan Jay Smith, Experimental Evaluation of On-Chip Microprocessor Cache Memories, Proc. Eleventh International Symposium on Computer Architecture, June 1984, Ann Arbor, MI.

Alan Jay Smith, Cache Memories, Computing Surveys, 14-3, September 1982.

BUGS

Not all combination of options have been thoroughly tested.

AUTHOR

Mark D. Hill
Computer Sciences Dept.
1210 West Dayton St.
Univ. of Wisconsin
Madison, WI 53706

markhill@cs.wisc.edu

REVISED

Nitzan Weinberg
Electrical and Computer Engineering Dept.
Carnegie Mellon University
5000 Forbes Ave.
Pittsburgh, PA 15213

nitzan@andrew.cmu.edu