Cache Performance Analysis

Urvashi Jouhari
Department of Computer Engineering and Computer Science, California State University Long Beach
{urvashi.jouhri@student.csulb.edu}

ABSTRACT

The performance of the cache is very important in evaluating the overall performance of the processor since access times increase as the processor moves to lower levels of memory in the hierarchy. Its performance in turn affects performance of the processor. In this paper, the various factors that effect the performance of the cache are analyzed and the results verified via simulation experiments using the SimpleScalar tool sets and SPEC2000 benchmark suite. The experiments are performed based on changing the L1 block size, L1 cache capacity with a set L2 cache capacity, L2 cache capacity with a set L1 cache capacity and replacement policies respectively. The experimental results are analyzed in detail. The results match the principles of cache architecture. The paper concludes by drawing optimal cache parameters.

Keywords **– Cache, Memory, Experimentation, Performance.**

1. INTRODUCTION

Memory hierarchy provides an efficient solution to the growing need for faster and larger memories. The memory hierarchy is organized into multiple levels. The memory placed closest to the processor is the smallest, fastest memory. As we move away from the CPU, the memory gets slower and larger. Thus, memory hierarchy introduces the concept of cache. Cache is the level of memory that is placed between the processor and main memory. It is the smaller, faster memory used to reduce the time taken by processor to find data. Figure 1 represents a basic memory hierarchy design with two cache levels. The cache achieves better performance gains by taking advantage of the principle of locality. Principle of locality states that programs have the tendency to reuse data and instructions that they have used recently, which implies that with the knowledge of recently accessed data or items, one can predict with reasonable accuracy the instructions and data that a program will use in the near future based on its accesses in the recent past. There are two types of locality, temporal and spatial. The former implies that recently accessed items are likely to be accessed in the near future and the latter states that items that have addresses close to each other have the tendency to be referenced close together in time.

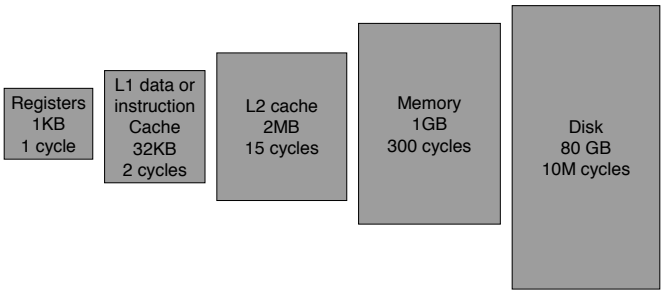

Figure 1 – Memory hierarchy design

2. CACHE BASICS

The basic idea of cache is to retain local copies of regularly used data so that most accesses are local. The cache design is such that it can transparently fetch the required data from the lower levels of memory hierarchy. Cache levels were introduced for the idea that a better hit ratio can be achieved by maintaining spatial data in higher levels [3]. Presence of cache levels helps the performance of the processors by reducing the number of accesses to the main memory on a miss. This is importance because of the 'performance distance' between the processor and the main memory. Present day CPU designs contain multiple cache levels. For the

purpose of this paper however, we assume that the cache has two levels. The first is the L1-cache and the other the L2-cache. If the data required is found in the cache, it signifies a hit and if not a miss. Upon encountering a miss, data is searched for in the next lower level. One of the most important parameters in determining cache performance is the miss rate. Miss rate is defined as the number of memory accesses not finding the data in the upper level with respect to the total number of memory accesses. This paper mainly analyses the effects of different parameters on the miss rate of both L1-cache and L2-cache.

3. FACTORS AFFECTING CACHE PERFORMANCE

A number of factors affect the performance of the cache. The main factors that affect the performance are analyzed in detail below. For the purpose of this paper, it is assumed that the cache has only two levels, L1-cache and L2-cache.

3.1. Cache Capacity

The accepted convention is that larger the size of the cache, lower the miss rate. However larger cache capacity come at higher costs and higher overhead. L1 cache sizes are typically smaller and range between 8KB-256KB and L2 cache sizes are larger than L1 cache sizes and range between 128KB-4MB. L1 and L2 cache capacities should be chosen suitably in order to achieve optimal performance.

3.2. Cache block size

The min. chunk of data that can be copied in the cache is the block or cache line [3]. The block size is required to be a multiple of the word size in order to be able to exploit the principle of locality. The general principle states that as the block size is increased, the miss penalties will increase and as a result the miss rate will decrease. However once cache block size increases to a certain degree, the miss rates begin to increase. This is due to two reasons: one is that the number of cache blocks is so small that it leads to completion of memory blocks and the other is that if the cache block is too large, spatial locality of block decreases. [1]

3.3. Replacement methods

Once the cache is full, the replacement method in the cache determines which block gets replaced from the cache. The most commonly used replacement methods are least recently used (LRU) that replaces the block that was used least recently, first-in first-out (FIFO) where the block that was first entered is replaced and random, which replaces blocks randomly.

3.4. Associativity

Associativity is number of memory block mirrors to cache.[1] Increasing the associativity of the cache usually lowers the miss rate. However, increasing the associativity also leads to an increase in hit time. Hence, suitable associativity needs to be implemented to achieve better performance.

3.5. Application Program Characteristics

The characteristic of the application program that is being run also affects the performance of the cache. In cases that cache capacity is fixed, the miss-rate is low when data set of program operation is small. [1] Similarly, if the data sets are large, miss-rates are higher. Additionally, cache performance has great differences for programs that emphasize on different aspects. [1]

4. SIMULATION TOOLS

The cache performance was evaluated using the SimpleScalar 3.0 tool sets with SPEC 2000 benchmark suite on a Ubuntu 12.10 Linux Platform. The SimpleScalar tool set is a system software infrastructure used to build modeling applications for program performance analysis, detailed microarchitectural modeling, and hardware-software co-verification. [1] Todd Austin initially developed the SimpleScalar tool set with the assistance of Doug Burger and Guri Sohi. SimpleScalar tool sets are now developed and supported by the SimpleScalar LLC. SimpleScalar tools permit users to setup

experimental simulations on processors. This includes modeling of the cache in terms of block size, L1-cache size, L2-cache size, associativity and replacement policies. SimpleScalar also contains tools for debugging, verifying infrastructure and resources for statistical analysis. SimpleScalar simulators can emulate the Alpha, PISA, ARM, and x86 instruction sets. [4] SimpleScalar tools can be built on either UNIX or Windows NT-based Operating Systems. Most 32-bit and 64-bit systems support it. However the most commonly used operating system is the Linux (64-bit) operating system. SPEC CPU2000, commonly known as SPEC 2000 is a standardized CPU intensive benchmark suite. It was designed by Standard Performance Evaluation Corporation (SPEC) to provide a comparative measure of compute intensive performance across the widest practical range of hardware. [5] Source code benchmarks in the SPEC 2000 are developed via real user applications. These benchmarks prove as a base that permit the testing and analysis of processors, memories and compilers based on defined criteria. Four SPEC 2000 benchmarks were chosen for the simulation experiments at random. The chosen benchmark suites 'gcc', 'bzip2', 'applu' and 'swim'. 'gcc' is a C complier under Linux. [1][4]. 'bzip2' is used in compressions, 'swim' is used for shallow water modeling and 'applu' is employed in the fields of computational fluid dynamics and computational physics.

5. SIMULATIONS AND RESULTS

The experiments performed aim to identify the effects of changing the L1 cache block size, L1 cache capacity, L2 cache capacity and least recently used (LRU) and random replacement policies on the miss rates of both the L1-cache and L2-cache. The experiments use the SimpleScalar tool sets and the SPEC 2000 benchmark suite. Most experiments are carried out on different benchmarks of the SPEC 2000 to verify the effects of different test programs on the cache performance. The various factors and parameters considered are explained in detail below:

5.1. L1 Block size - Using 'sim-cache'

5.1.1. Parameters and program illustration

The L1 cache size was defined as 8KB. The following program was run under tool 'sim-cache' with L1 block sizes ranging between 8 bytes to 64 bytes:

```
#define SIZE_OF_ARRAY 16*1024

#define LOOPS 100

int main()
{
  char array[SIZE_OF_ARRAY];
  register int out_loop;
  register int in_loop;
  register int solution = 0;

  for (out_loop = 0; out_loop < NUM_LOOPS; out_loop++)
    {
     for (in_loop = 0; in_loop < ARRAY_SIZE; in_loop++)
       {
         solution *= array[in_loop];
       }
    }
  return solution;
}
```

5.1.2. Experimental observations and analysis

Upon running the above program with the SimpleScalar tool sim-cache, L1-cache and L2-cache miss rates are observed. Table 1 indicates the miss rates of L1-cache and L2-cache respectively as the L1 block size ranges between 8bytes-64bytes. Chart 1 represents the comparison of the L1 and L2 miss rates as block sizes of L1-cache are changed. From the Chart, it can be observed that L1 cache miss rate is highest when the block size is 8 bytes. Consequently the L1 miss rate drops as the size of the block increases. However, the L2 cache miss rates increase as the block size of L1 is increased.

L1 Block Size	8 bytes	16 bytes	32 bytes	64 bytes
L1 Miss Rate	0.1257	0.0629	0.0314	0.0158
L2 Miss Rate	0.0031	0.0062	0.0123	0.0243

Table 1- Cache miss rates when L1 block size is changed.

5.2. L1 Cache and Block-size – Under 'gcc' benchmark

5.2.1. Parameters and program illustration

This experiment compares the effect of changing the L1 cache size and L1 block size on the miss rates of the L1-cache and L2-cache under SPEC 2000 benchmark suite. The test sets were based under the 'gcc' benchmark. Gcc is an integer benchmark, usually used for C-program compilations under Linux platforms.

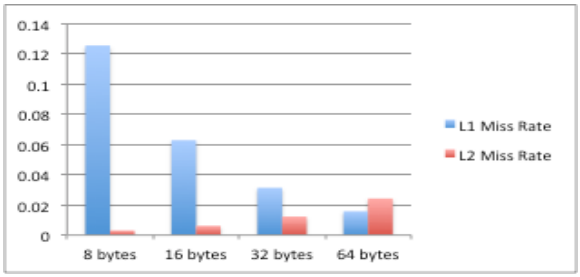

Chart 1- Comparison of L1 and L2 miss rates as the block size of L1-cache is increased.

5.2.2. Experimental observations and analysis

Table 2 represents the miss rates of L1-cache and L2-cache for L1 block sizes ranging between 8 bytes to 64 bytes and L1 cache sizes ranging from 8KB-256KB. From Chart 2, it is obvious that L1 miss rates decrease as the block size of L1 increases and L2 miss rates increase as the block size of L1 increases. Additionally, at L1-block size of 32 bytes, minimal miss rates are encountered. All further experiments assume block size of 32 bytes for L1 cache.

	8bytes		16bytes		32 bytes		64 bytes	
L1 Cache size	L1	L2	L1	L2	L1	L2	L1	L2
8KB	0.0641	0.0962	0.0521	0.1067	0.0479	0.1120	0.0507	0.1122
16KB	0.0430	0.1111	0.0341	0.1200	0.0307	0.1245	0.0314	0.1253
32KB	0.0268	0.1233	0.0202	0.1303	0.0171	0.1344	0.0161	0.1361
64KB	0.0149	0.1353	0.0106	0.1374	0.0085	0.1387	0.0075	0.1397
128KB	0.0106	0.1362	0.0072	0.1358	0.0054	0.1360	0.0045	0.1365
256KB	0.0067	0.1383	0.0042	0.1342	0.0029	0.1318	0.0023	0.1311

Table 2 – L1 and L2 miss rates for different L1 cache sizes with variable block sizes

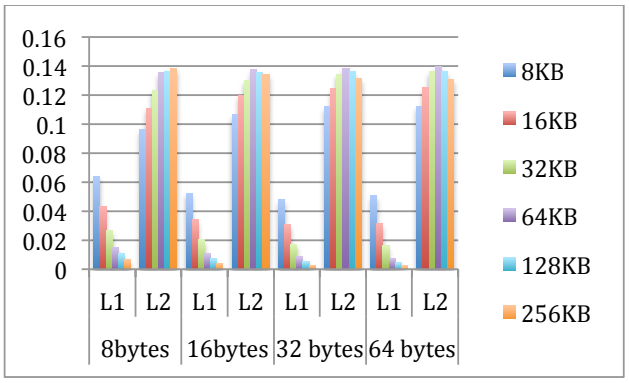

Chart 2 - Comparison of L1 and L2 miss rates as the block size of L1-cache is increased for different L1 cache sizes.

5.3. L1-cache size- Under various SPEC 2000 benchmarks

5.3.1. Parameters and program illustration

This experiment compares the effect of changing the L1 cache size on the miss rates of the L1-cache and L2-cache under SPEC 2000 benchmark suite. At random, two integer and two floating-point benchmarks of the SPEC 2000 benchmark suite are chosen. The two integer programs chosen are gcc and bzip2. The two floating-point programs chosen are applu and swim.

5.3.2. Experimental observations and analysis

Table 3 represents the L1 miss rates for the benchmark suites 'gcc', 'bzip2', 'applu' and 'swim' when L1 cache size is varied from 8KB-256KB. Table 4 represents the L2 miss rates for the benchmark suites 'gcc', 'bzip2', 'applu' and 'swim' when L1 cache size is varied from 8KB-256KB. Charts 3 and 4 provide comparative analysis of L1

miss rates for different cache sizes and L2 miss rates for different cache sizes respectively. The common observations for all the benchmarks are thatL1 miss rates are high for L1 capacity 8K. Additionally, with increase in the L1 capacity, corresponding miss rates decrease. Moreover, the miss rate drops in the L1 are significant initially, however the decreasing speed of the miss rates slow down as the L1 size grows and L2 miss rates increase gradually as L1 capacity increases.

	L1 miss rates			
L1 cache size	gcc	bzip2	applu	swim
8k	0.0023	0.0151	0.1823	0.1139
16k	0.0005	0.0135	0.1163	0.1039
32k	0.0002	0.0121	0.1017	0.0931
64k	0.0001	0.0106	0.0964	0.0928
128k	0.0001	0.0088	0.0888	0.0897
256k	0.0001	0.0063	0.0866	0.0893

Table 3- L1 miss rates for variable L1 cache sizes

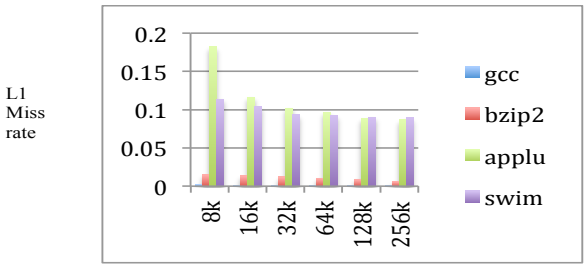

Chart 3 –L1 miss rates for different L1 capacities

	L2 miss rates			
L1 cache size	gcc	bzip2	applu	swim
8k	0.0013	0.2851	0.1576	0.2639
16k	0.0011	0.3306	0.2085	0.2732
32k	0.0011	0.3776	0.2406	0.3035
64k	0.0011	0.4337	0.2638	0.3045
128k	0.0011	0.5231	0.3004	0.5442
256k	0.0011	0.7077	0.4803	0.6909

Table 4- L2 miss rates for variable L1 cache sizes

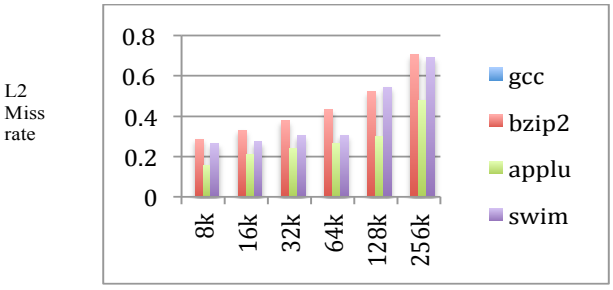

Chart 4 –L2 miss rates for different L1 capacities

5.4. L2-cache size- Under various SPEC 2000 benchmarks

5.4.1. Parameters and program illustration

This experiment compares the effect of changing the L2 cache size on the miss rates of the L1-cache and L2-cache under SPEC 2000 benchmark suite. The tests are again run on 'gcc', 'bzip2', 'applu' and 'swim'.

5.4.2. Experimental observations and analysis

Table 5 represents the L1 miss rates for the benchmark suites 'gcc', 'bzip2', 'applu' and 'swim' when L2 cache size is varied from 128KB-4MB. Table 6 represents the L2 miss rates for the benchmark suites 'gcc', 'bzip2', 'applu' and 'swim' when L1 cache size is varied from 128KB-4MB. Charts 5 and 6 provide comparative analysis of L1 miss rates for different cache sizes and L2 miss rates for different cache sizes respectively. The common observations for all the benchmarks are that L1 miss rates stay unchanged with changes in the L2 capacity. In L2 miss rates gradually decrease as size of L2 cache increases. Also, The miss rate reduction of L2 is high initially, however the decreasing speed of miss rates slow down after a certain capacity.

	L1 miss rates			
L2 cache size	gcc	bzip2	applu	swim
128K	0.0261	0.0931	0.1017	0.0931
256K	0.0261	0.0931	0.1017	0.0931
512K	0.0261	0.0931	0.1017	0.0931
IM	0.0261	0.0931	0.1017	0.0931
2M	0.0261	0.0931	0.1017	0.0931
4M	0.0261	0.0931	0.1017	0.0931

Table 5 - L1 miss rates for variable L1 cache sizes

5.5. Replacement policies- Under various SPEC 2000 benchmark

5.5.1. Parameters and program illustration

This experiment compares the effect of changing the replacement policies on the miss rates of the L1-cache and L2-cache under SPEC 2000 benchmark

suite. The tests are again run on 'gcc', 'bzip2', 'applu' and 'swim'. The two policies tested are Least recently used and Random.

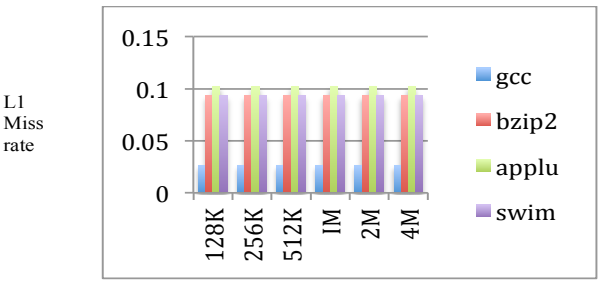

Chart 5- L1 miss rates for different L2 capacities

	L2 miss rates			
L2 cache size	gcc	bzip2	applu	swim
128K	0.1464	0.5676	0.2704	0.3049
256K	0.0913	0.3776	0.2406	0.3035
512K	0.0520	0.2190	0.2200	0.3028
IM	0.0186	0.1171	0.2144	0.3013
2M	0.0102	0.0834	0.2095	0.3001
4M	0.0076	0.0795	0.2060	0.2997

Table 6 – L2 miss rates for variable L1 cache sizes

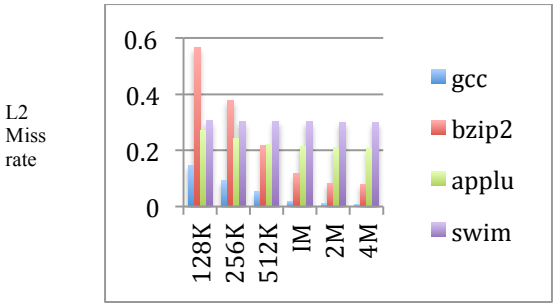

Chart 6- L2 miss rates for different L2 capacities

5.5.2. Experimental observations and analysis

Table 7 represents the L1 miss rates for the benchmark suites 'gcc', 'bzip2', 'applu' and 'swim' for the different combinations of replacement policies. Table 8 represents the L2 miss rates for the benchmark suites 'gcc', 'bzip2', 'applu' and 'swim' for the different combinations of replacement policies. Charts 7 and 8 provide comparative analysis of L1 miss rates and L2 miss rates for the different combinations of replacement policies. The common observations for all the benchmarks are that if the replacement policy of the L1 cache is left unchanged, the miss rate of the L1 cache remains unchanged, irrespective of the replacement policy of L2 cache. Changing replacement policy of L1 from LRU to Random, increases the miss rate of L1cache, however the miss rate of L2 cache is lowered. It can be inferred that the LRU replacement policy is better than the Random replacement

	L1 miss rates			
Replacement Policy	gcc	bzip2	applu	swim
L-L	0.0261	0.0121	0.1017	0.0931
L-R	0.0261	0.0121	0.1017	0.0931
R-R	0.0261	0.0126	0.1017	0.0935
R-L	0.0261	0.0133	0.1017	0.0938

Table 7- L1 miss rates for different replacement policies

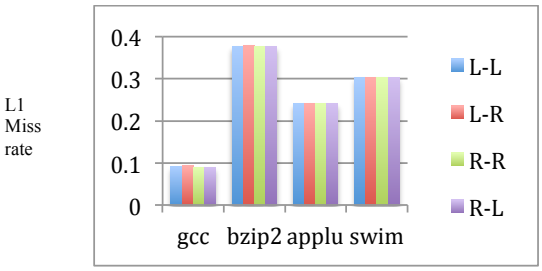

Chart 7- L1 miss rates for different replacement policies

	L2 miss rates			
Replacement Policy	gcc	bzip2	applu	swim
L-L	0.0913	0.3776	0.2406	0.3035
L-R	0.0936	0.3791	0.2409	0.3035
R-R	0.0894	0.3769	0.2406	0.3026
R-L	0.0894	0.3767	0.2406	0.3021

Table 8- L2 miss rates for different replacement policies

Chart 8 – L2 miss rates for different replacement policies

6. INFERENCES

The basic inferences that can be made from the simulation experiments performed are the following:

- The experiments conformed with the principle of processor architectures.
- As the block size of the L1 cache is increased, the miss rate of the L1 cache decreases. This is because, once block size of L1cache increases, the L1 cache can better utilize the spatial locality property and reduce miss rates. However, once block size is increased from 32bytes to 64bytes, the miss rate of L1 cache increased. This could be due to completion between memory blocks. An optimal block size therefore would be that of 32 bytes.

- As the L1 capacity is increased, the miss rate of the L1 cache decreases. This happens because once the L1 capacity increases, the probability of a hit becomes higher as a result decreasing the miss rate. It could also be noted that the drop in miss rates of L1 were significantly large as size of L1 increased initially, however the drop slowed down once the L1 cache capacity reached between 32KB-64KB. This is because once L1 capacity is sufficient for usage; its size does not have a great effect on miss rates.

- As the L1 cache capacity is increased, the L2 cache miss rate increases. This is due to the fact that once L1 cache capacity increases, the probability of it containing more all-round data increases. This implies that by the principle of locality if the data is not found in the L1 cache, the probability of finding it in the L2 cache is also low.

- As the L2 cache capacity is increased, the L1 cache capacity remains unchanged. Since the L1 cache is a subset of the L2 cache and the memory accesses to L1 cache happen before the memory accesses to L2 cache, the probability of finding data in the L1 cache remains unchanged.

- With increase in the L2 cache capacity, the L2 cache miss rate decreases. The drop in miss rates of L1 were significantly large as size of L1 was increased initially, however the drop slowed down once the L2 cache capacity reached between 512KB-1MB. This is because once L1 capacity is sufficient for usage; its size does not have a great effect on miss rates.

- The changes in replacement policies affected the L2 cache miss rates more than the L1 cache miss rates. The general inference however was that the least recently used policy (LRU) gave comparatively lower miss rates for both the L1 and L2 caches than the Random policy.

7. CONCLUSION

The paper aimed at analyzing the various factors that affected performance of the cache in detail. Optimal cache design parameters based on the lowest miss rates were found to be a L1 cache with a 32 byte block size, a 32KB-64KB L1 cache capacity and a 512-1MB L2 cache capacity using the LRU replacement policy. The simulation experiments conformed to the principles of computer architecture.

8. REFERENCES

1. "*Cache Performance Simulation and Analysis under SimpleScalar Platform*", MA Hai-feng, YAO Nian-min, FAN Hong-bo, *2009* . IEEE Publication

2. "*Miss Rate Prediction across All Program Inputs*"**,** Yutao Zhong, Steven G. Dropsho, and Chen DingThe.

3. "*Computer Architecture – A Quantitative Approach*", 5th Edition, by John L. Hennessy and David A. Patterson

4. http://www.simplescalar.com/

5. http://www.spec.org/cpu2000/

6. *"SimpleScalar Tool Set, Version 2.0",* Doug Burger, Todd M. Austin.
7. *"An Analytical Model for Cache Replacement Policy Performance",* Fei Guo and Yan Solihin, *2006.* IEEE Publication.

8. *"SimpleScalar Tool Set, Version 2.0",* Doug Burger, Todd M. Austin.

9. "*SimpleScalar Hacker's guide"*, Todd Austin.

10. *"SimpleScalar Tutorial"*, Todd Austin, Dan Ernst, Eric Larson, Chris Weaver, RajDesikan, Ramadass Nagarajan, Jaehyuk Huh, Bill Yoder, Doug Burger, Steve Keckler,

11. http://harryscode.blogspot.com/2008/10/installing-simplescalar.html