Database Systems Implementation (w4112)

Project 2 (Parts II & III)

Shubham Bansal, sb3766

Mark Rampton, mcr2176

April 28, 2016

ENVIRONMENT AND IMPLEMENTATION

Machines

The following clic machine was used for testing and analysis.

beijing.clic.cs.columbia.edu

```
DISTRIB_ID=Ubuntu
DISTRIB_RELEASE=12.04
DISTRIB_CODENAME=precise
DISTRIB_DESCRIPTION="Ubuntu 12.04.5 LTS"
NAME="Ubuntu"
VERSION="12.04.5 LTS, Precise Pangolin"
ID=ubuntu
ID_LIKE=debian
PRETTY_NAME="Ubuntu precise (12.04.5 LTS)"
VERSION_ID="12.04"
```

Machine Characteristics

Architecture	x86_64
CPU op-mode(s)	32-bit, 64-bit
Byte Order	Little Endian
CPU(s)	4
On-line CPU(s) list	0-3
Thread(s) per core	1
Core(s) per socket	1
Socket(s)	4
NUMA node(s)	1
Vendor ID	GenuineIntel
CPU family	6
Model	44
Stepping	2
CPU MHz	2666.761
BogoMIPS	5333.52
Hypervisor vendor	VMware
Virtualization type	full
L1d cache	32K
L1i cache	32K
L2 cache	256K
L3 cache	12288K
NUMA node0 CPU(s)	0-3

Machine Usage

A number of other users were also logged into ***beijing*** while our testing and analysis was underway. However, these additional users did not place significant demands on either the machines memory or cpu resources. Below is a representative snapshot of the system usage during the time of our testing.

$ vmstat

```
procs -----------memory---------- ---swap-- -----io---- -system-- ----cpu----
 r  b   swpd   free   buff  cache   si   so    bi    bo   in   cs us sy id wa
 0  1 7701556 3527368 133992 1244452    1    5     6    33    1    1 22 12 66  0
```

$ top

```
top - 21:33:57 up 42 days,  4:57,  9 users,  load average: 0.05, 0.14, 0.20
Tasks: 501 total,   1 running, 497 sleeping,   3 stopped,   0 zombie
Cpu(s):  0.9%us,  1.3%sy,  0.0%ni, 97.8%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Mem:   6114092k total,  2574532k used,  3539560k free,   133576k buffers
Swap:  8385532k total,  7701556k used,   683976k free,  1244052k cached
```

Compiling Process

Compiling was carried out using gcc. Make was used for with the following build instruction.
build:

```
gcc -std=gnu99 -pedantic -msse4.2 -msse4a -O2 -o build main.c p2random.c tree.c -lrt
```

Information on Provided flags:

-lrt is needed for *clock_gettime* which we depend on for our timekeeping and analysis.
-std=gnu99 is required to enable POSIX instructions which we require *struct timespec*

Timing Instrumentation

The program was implemented in a manner so as to make singling out and timing the probing portion of the code simple. In general, the timekeeping was carried out through code such as the following.

```
clock_gettime(CLOCK_MONOTONIC, &start);
// all probing occurs here
clock_gettime(CLOCK_MONOTONIC, &finish);
```

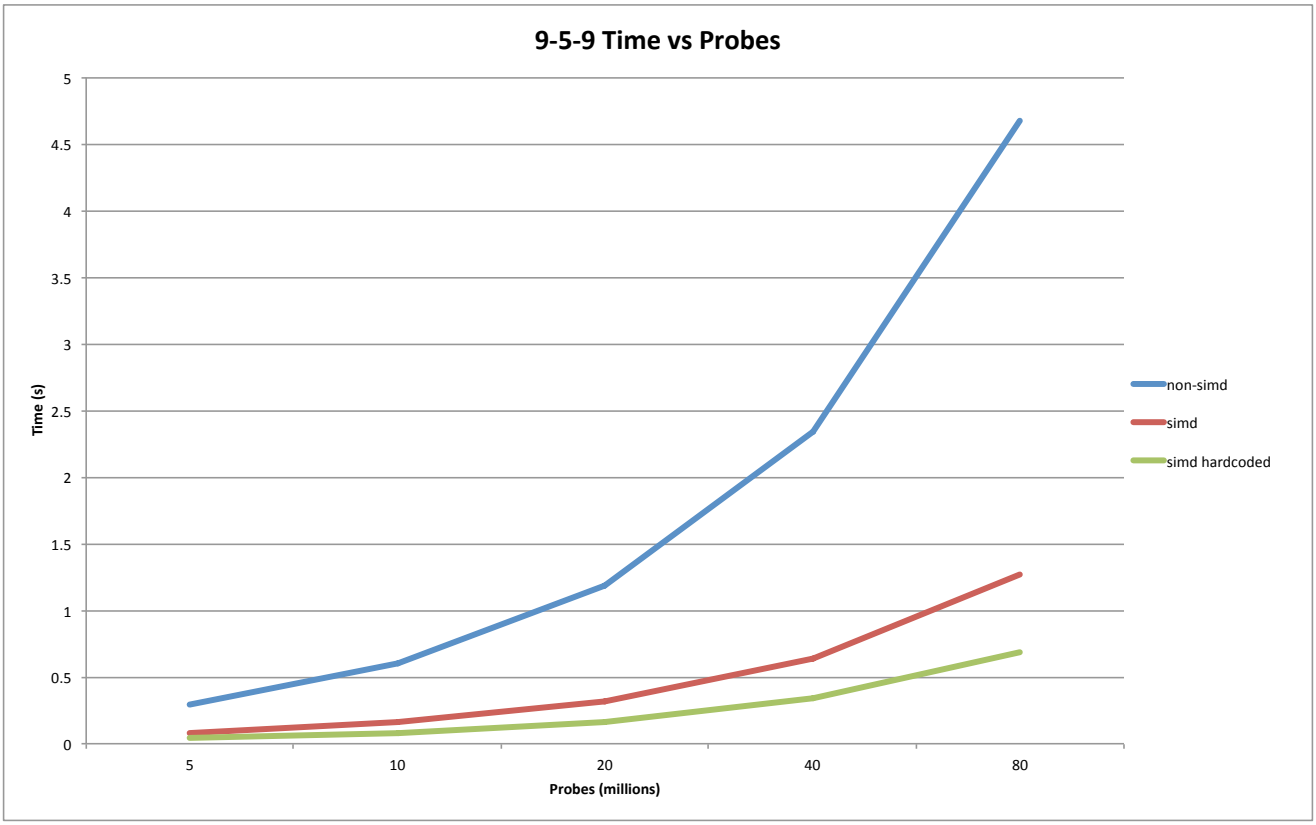
Timekeeping was kept when probing using both the general tree structure probing (completed in Part I of this project), as well as for the simd probing implementation. Additionally, timing was tracked for simd probing against a hardcoded tree of structure **9-5-9**. We have used the data collected from probing the above mentioned trees in order to compare their performance.

RESULTS

9-5-9 Tree Structure

We built our 9-5-9 tree structure with 300 keys and probed it with probe counts ranging from 5 million to 80 million. As expected the simd probing outperforms the non-simd probing method by significant margins. Furthermore, the simd probing of the hardcoded tree is even more performant — performing 80 million probes in roughly the same time the simd (non-hardcoded) takes to perform 40 million. The different methods all display characteristics of nlog(n) growth, but this is most evident in the non-simd probing; the simd methods both display more subdued growth patters, even if they are not quite linear.

Keys/Probes	non-simd probe time (s)	simd probe time (s)	simd hardcoded probe time (s)
300 / 5 million	0.293245	0.079824	0.043322
300 / 10 million	0.597671	0.160804	0.084937
300 / 20 million	1.184453	0.318369	0.165007
300 / 40 million	2.340498	0.633958	0.338249
300 / 80 million	4.676538	1.266153	0.691492

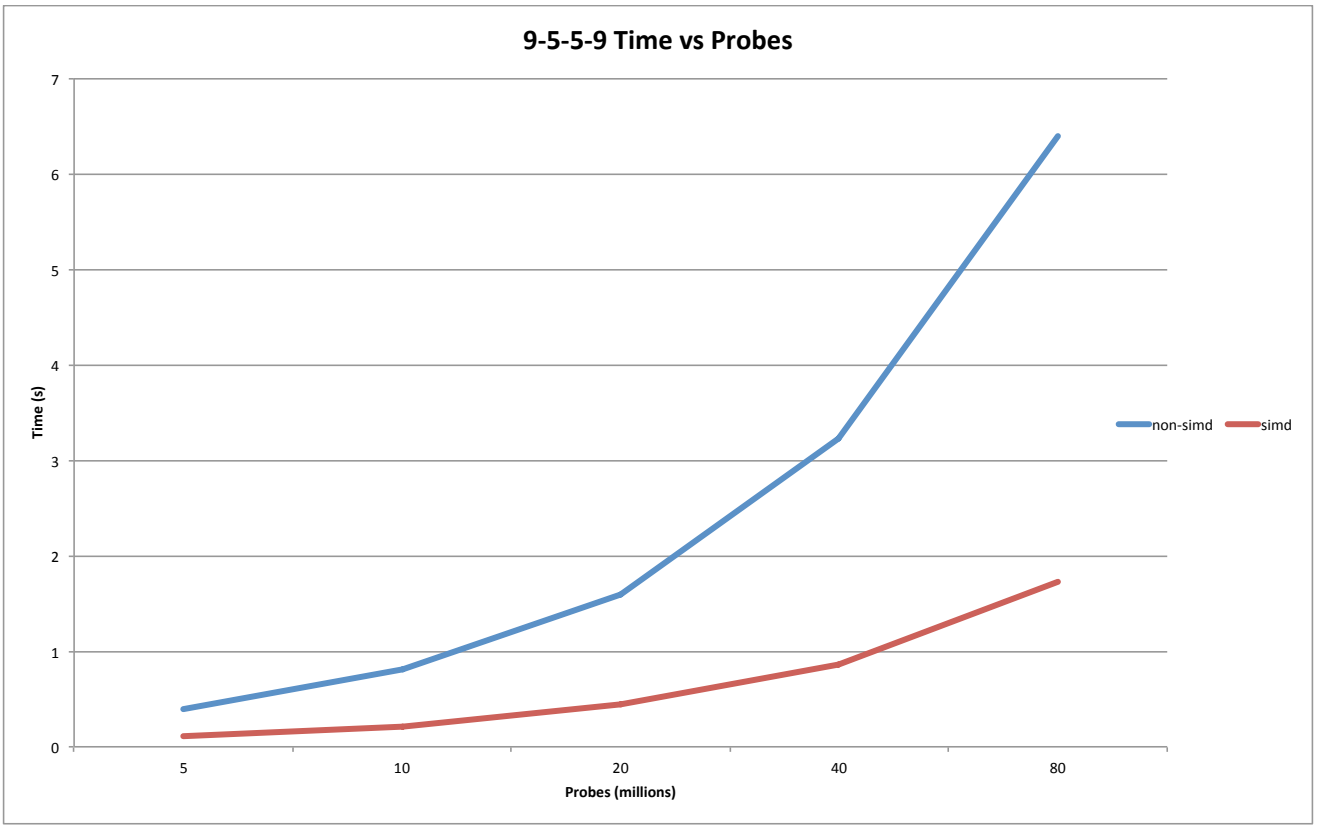


9-5-5-9 Tree Structure

We built our 9-5-5-9 tree structure with 1000 keys and probed it with probe counts ranging from 5 million to 80 million. As in the 9-5-9 tree probing, the simd probing outperforms the non-simd probing method by significant margins. Because the 9-5-5-9 tree is of one greater depth than the 9-5-9 tree, we expect times required for probing to be greater. In fact, it would take nearly as much time to perform 60 million probes against the 9-5-5-9 tree as 80 million probes against the 9-5-9 tree. The nlog(n) characteristic of the non-simd method, as described above, is also evident here — while the simd method continues to appear more closely to linearly defined.

Keys/Probes	non-simd probe time (s)	simd probe time (s)
1000 / 5 million	0.393741	0.106696
1000 / 10 million	0.811659	0.213779
1000 / 20 million	1.594233	0.441981
1000 / 40 million	3.226902	0.859649

Keys/Probes	non-simd probe time (s)	simd probe time (s)
1000 / 80 million	6.397579	1.722480



17-17 Tree Structure

We built our 17-17 tree structure with 250 keys and probed it with probe counts ranging from 5 million to 80 million. As in the 9-5-9 tree probing, the simd probing outperforms the non-simd probing method by significant margins. Because the 17-17 tree is of one lesser depth than the 9-5-9 tree, we expect times required for probing to also be less — despite the fanout of each level being significantly larger. This is shown to play out as expected, and both probing methods probe perform 80 million probes in less time than that required for the 9-5-9 tree probing.

Keys/Probes	non-simd probe time (s)	simd probe time (s)
250 / 5 million	0.277840	0.072865
250 / 10 million	0.556014	0.146342
250 / 20 million	1.106042	0.283265
250 / 40 million	2.194447	0.567704
250 / 80 million	4.379297	1.136160