

LinuxDNA Benchmarks for the 2.6.33 Kernel

Conducted and documented by Luyi Cheng and C. Tyler McAdams of LinuxDNA
02/07/2010

Conditions:

Tests were run using LMBench 3.0 against both an ICC and GCC kernel with ~similar flags on an Intel Atom 330 64bit dual core cpu on the 2.6.33-rc6 kernel with Glibc-2.11, & Binutils-2.20.51.20091120

Kernel configuration file can be downloaded at:

<http://www.linuxdna.com/config-dna>

GCC 4.5 Flags:

-O3 -march=atom -mtune=atom

(Please note -O3 in GCC 4 now turns on some advanced optimizations by default like automatic vectorization)

ICC 11.1.059 Flags:

-O3 -xSSE3_ATOM -ip -fp-model fast=2 -unroll-aggressive -vec-guard-write

Benchmark results for LMBench 3.0:

Basic system parameters

Host	OS Description	Mhz	tlb pages line bytes	cache par load	mem	scal
atom-gcc	Linux 2.6.33-	x86_64-linux-gnu	1600	64	1.0000	1
atom-icc	Linux 2.6.33-	x86_64-linux-gnu	1600	64	1.0000	1

Processor, Processes - times in microseconds - smaller is better

Host	OS	Mhz	null	null	open	slct	sig	sig	fork	exec	sh
		call	I/O	stat	clos	TCP	inst	hdl	proc	proc	
atom-gcc	Linux 2.6.33-	1600	0.20	0.36	1.97	5.81	7.44	0.49	2.69	461.	1417 4756
atom-icc	Linux 2.6.33-	1600	0.21	0.36	2.07	5.85	7.18	0.49	2.90	332.	1320 4565

Basic integer operations - times in nanoseconds - smaller is better

Host	OS	intgr bit	intgr add	intgr mul	intgr div	intgr mod		
atom-gcc	Linux 2.6.33-	0.6400	0.4100	0.2500	40.4	40.5		
atom-icc	Linux 2.6.33-	0.6300	0.4100	0.2800	40.1	40.2		

Basic uint64 operations - times in nanoseconds - smaller is better

```
-----
Host          OS int64 int64 int64 int64 int64
              bit add  mul  div  mod
-----
atom-gcc Linux 2.6.33- 0.630    0.7600 96.2 96.5
atom-icc Linux 2.6.33- 0.630    0.7800 95.3 96.0
```

Basic float operations - times in nanoseconds - smaller is better

```
-----
Host          OS float float float float
              add  mul  div  bogo
-----
atom-gcc Linux 2.6.33- 3.1000 2.4900 20.8 28.0
atom-icc Linux 2.6.33- 3.1100 2.4900 20.6 27.8
```

Basic double operations - times in nanoseconds - smaller is better

```
-----
Host          OS double double double double
              add  mul  div  bogo
-----
atom-gcc Linux 2.6.33- 3.1000 3.1300 39.1 47.0
atom-icc Linux 2.6.33- 3.1100 3.1200 38.8 46.7
```

Context switching - times in microseconds - smaller is better

```
-----
Host          OS 2p/0K 2p/16K 2p/64K 8p/16K 8p/64K 16p/16K 16p/64K
              ctxsw ctxsw  ctxsw  ctxsw  ctxsw  ctxsw  ctxsw
-----
atom-gcc Linux 2.6.33- 15.0 16.6 13.5 17.9 24.2 22.9 27.0
atom-icc Linux 2.6.33- 4.0300 7.2000 4.3400 9.0700 14.2 12.3 17.8
```

Local Communication latencies in microseconds - smaller is better

```
-----
Host          OS 2p/0K Pipe AF  UDP RPC/ TCP RPC/ TCP
              ctxsw  UNIX  UDP  UDP  TCP conn
-----
atom-gcc Linux 2.6.33- 15.0 34.6 34.2 71.1 90.9 158.
atom-icc Linux 2.6.33- 4.030 11.5 24.1 49.5 73.2 216.
```

Remote Communication latencies in microseconds - smaller is better

```
-----
Host          OS UDP RPC/ TCP RPC/ TCP
              UDP  TCP conn
-----
atom-gcc Linux 2.6.33-
atom-icc Linux 2.6.33-
```

File & VM system latencies in microseconds - smaller is better

Host	OS	OK File	10K File	Mmap	Prot	Page	100fd
		Create	Delete	Create	Delete	Latency	Fault
						Fault	selct
atom-gcc	Linux 2.6.33-	40.2	30.3	96.7	44.0	46.8K	0.896
atom-icc	Linux 2.6.33-	46.4	34.6	101.1	45.3	45.2K	1.017

Local Communication bandwidths in MB/s - bigger is better

Host	OS	Pipe	AF	TCP	File	Mmap	Bcopy	Bcopy	Mem	Mem
		UNIX	reread	reread	(libc)	(hand)	read	write		
atom-gcc	Linux 2.6.33-	522.	1106	384.	1189.1	2970.8	963.9	968.3	2401	1130.
atom-icc	Linux 2.6.33-	767.	620.	350.	1196.4	2993.0	986.9	978.8	2430	1145.

Memory latencies in nanoseconds - smaller is better
(WARNING - may not be correct, check graphs)

Host	OS	Mhz	L1 \$	L2 \$	Main mem	Rand mem	Guesses
atom-gcc	Linux 2.6.33-	1600	1.9090	9.6120	39.7	286.8	
atom-icc	Linux 2.6.33-	1600	1.9010	9.5580	39.2	284.3	

make[1]: Leaving directory `/root/lmbench-3.0-a9/lmbench-3.0-a9/results'