Sample EE699 Project Presentation

Hank Dietz
Professor and James F. Hardymon Chair in Networking
Electrical and Computer Engineering Department
University of Kentucky
Lexington, KY 40506-0046
http://aggregate.org/hankd/

My Project: `bzip2`

- Compression is important...
- `http://sources.redhat.com/bzip2/`
- `bzip2` is patent free and very efficient
- `bzip2` is highly tuned code, still too darned slow... nearly 100% compute (not I/O) bound!

```
bzip2-1.0.2.tar     1771520
bzip2-1.0.2.tar.gz   665198  0.246s  0.220s
bzip2-1.0.2.tar.bz2  624837  3.119s  3.040s
```

What's Slow?

- Changed `Makefile` to use `-pg` (and not `omit-frame-pointer`)
- Ran over `bzip2` source TAR file
- Gprof results (Athlon MP 1800+):

```
 %   cumulative   self              self     total
time   seconds   seconds    calls   s/call   s/call  name
41.25      1.25     1.25        1     1.25     1.70  fallbackSort
14.85      1.70     0.45  2156817     0.00     0.00  mainGtU
12.21      2.07     0.37   225210     0.00     0.00  fallbackQSor
11.22      2.41     0.34        2     0.17     0.49  mainSort
 7.92      2.65     0.24        2     0.12     0.12  generateMTFV
 4.95      2.80     0.15    40386     0.00     0.00  mainQSort3
```

The inner loop in `fallbackSort()`

```
/*-- find the next non-singleton bucket --*/
k = r + 1;
while (ISSET_BH(k) && UNALIGNED_BH(k)) k++;
if (ISSET_BH(k)) {
  while (WORD_BH(k) == 0xffffffff) k += 32;
  while (ISSET_BH(k)) k++;
}
l = k - 1;
if (l >= nblock) break;
while (!ISSET_BH(k) && UNALIGNED_BH(k)) k++;
if (!ISSET_BH(k)) {
  while (WORD_BH(k) == 0x00000000) k += 32;
  while (!ISSET_BH(k)) k++;
}
r = k - 1;
```

Conversion of the inner loop

```
/* How many bits are 1 from here? */
RUInt32 w = WORD_BH(k);
w >>= (k & 31);
k += trailing1s(w);
if ((k & 31) == 0) {
  while ((w = WORD_BH(k)) == 0xffffffff) k += 32;
  k += trailing1s(w);
}
```

Branchless `trailing1s()`

```
inline static UInt32
trailing1s(RUInt32 x)
{
  RUInt32 t, c = 0;
  t = ((x & 0xffff) - 0xffff); t = (((Int32) t) >> 31);
  t = ((~t) & 16); c += t; x >>= t;
  t = ((x & 0xff) - 0xff); t = (((Int32) t) >> 31);
  t = ((~t) & 8); c += t; x >>= t;
  t = ((x & 0xf) - 0xf); t = (((Int32) t) >> 31);
  t = ((~t) & 4); c += t; x >>= t;
  t = ((x & 0x3) - 0x3); t = (((Int32) t) >> 31);
  t = ((~t) & 2); c += t; x >>= t;
  return(c + (x & 1));
}
```

Other Improvements

- Insertion of (more) `register` where appropriate
- Minor loop rewrites:
  - Use of `memset()` for clearing arrays
    (tried and removed; it slowed-down the code!)
  - Prefix addition using a register across iterations
  - Reversed a few loops to walk in memory order

Did it all work?

% time	cumulative seconds	self seconds	calls	self s/call	total s/call	name
40.14	1.18	1.18	1	1.18	1.65	fallbackSort
13.95	1.59	0.41	224388	0.00	0.00	fallbackQSort3
13.27	1.98	0.39	2156817	0.00	0.00	mainGtU
11.90	2.33	0.35	2	0.17	0.47	mainSort
7.48	2.55	0.22	2	0.11	0.11	generateMTFVal
5.44	2.71	0.16	40386	0.00	0.00	mainQSort3
2.38	2.78	0.07	2	0.04	0.04	sendMTFValues
2.04	2.84	0.06	342979	0.00	0.00	fallbackSimple
1.36	2.88	0.04	111832	0.00	0.00	mainSimpleSort
1.36	2.92	0.04	357	0.00	0.00	copy_input_unt

The Final Results

- Removed `-pg` from CFLAGS, added athlon flags
- Test case speedup just over 1.05, about 5%:

```
bzip2-1.0.2.tar.bz2  624837  3.119s  3.040s  original
bzip2-1.0.2.tar.bz2  624837  2.930s  2.890s  optimized
```

- A huge test case (my email archive)...
  from 91.95s to 87.80s;
  speedup also about 5%
- Major improvement requires algorithm change...