SICSA Benchmarks in C

SICSA Multi-core challenge Workshop Dec 2010

Paul Cockshott
Glasgow

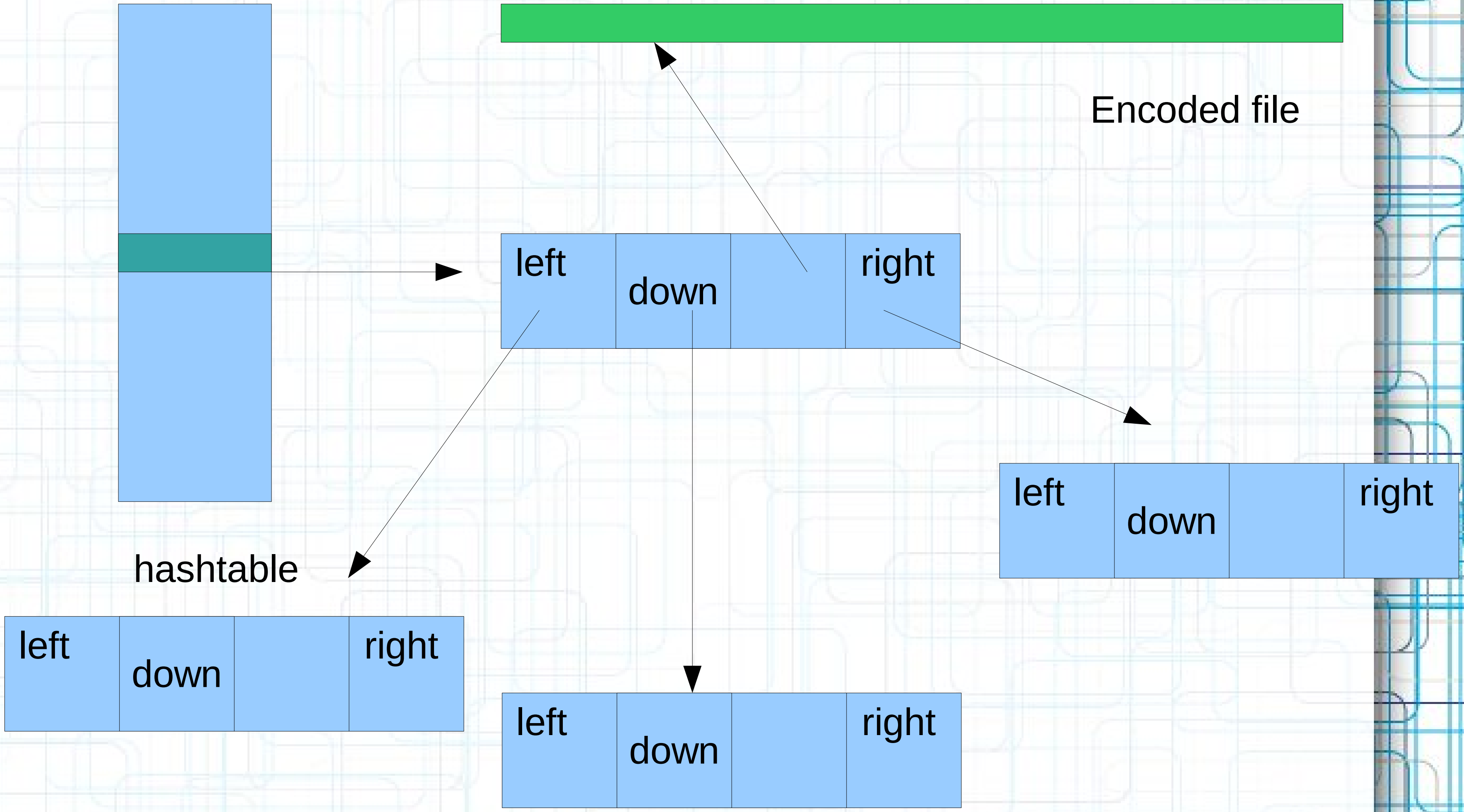
•First Serial Experiments

- Prior to doing any parallelisation it is advisable to initially set up a good sequential version. I was initially doubtfull that this challenge would provide an effective basis for parallelisation because it seemed such a simple problem. Intutively it seems like a problem that is likely to be of either linear or at worst log linear complexity, and for such problems, especially ones involving text files, the time taken to read in the file and print out the results can easily come to dominate the total time taken. If a problem is disk bound, then there is little advantage in expending effort to run it on multiple cores.

Algorithm

The algorithm is four pass.

1) Read the file into a buffer
2) Produce a tokenised version of the buffer
3) Build the hash table and prefix trees.
4) If the concordance is to be printed out, perform a traversal of the trees printing out the word sequences in the format suggested by the Haskell version.
5) If we want it sorted, pipe through Linux sort


Encoded file
left
down
right
hashtable
left
down
right
left
down
right
left
down
right

Serial results

Machine [illegible]

language	Filesize	OS	Time
Haskell	3580	windows	0.82
C	3580	windows	0.03
Haskell	4792092	windows	timeout>2hrs
C	4792092	windows	3.67
C	4792092	Linux	2.68

Conclusion from serial test

- As the summary above shows the C version is
- Significantly faster than the Haskell version. This is not surprising as one would expect C to be more efficient.
- Appears to have a lower complexity order than the Haskell version. This would indicate that the Haskell version is not a good starting point.
- Its run time is dominated by the time to ouput the results to file.
- The test files provided in the initial proposal were too short to get a realistic estimate of performance
- Linux implementations run substantially faster than Windows on the same processor + gcc.

Parallel Experiments

-
- As a first parallel experiment a dual core version of the C programme was produced using the pthreads library and it was tested on the same dual processor machine as the original serial version of the algorithm.
- A second parallel version used a simple shell script

```
./l1concordance WEB.txt 4 P 1 0 >WEB0.con&
./l1concordance WEB.txt 4 P 1 1 >WEB1.con
wait
cat WEB1.con >>WEB0.con
```

-

Parallel timings

Mechanism	OS	Threads used	Time	sorted output	opt level
serial	windows	1	3.67	no	0
pthreads	windows	2	5.63	no	0
serial	linux	1	2.68	no	0
pthreads	linux	2	2.26	no	0
pthreads	linux	2	2.45	yes	3
Shell &	linux	2	1.93	yes	3

Conclusions for 2 core machine

- There was no gain using multithreading on windows. It looks as if the pthreads library under windows simply multi threads operations on a single core rather than using both cores.
- On Linux there was a small gain in performance due to multithreading - about 17% faster in elapsed time using 2 cores.
- Since a large part of the program execution is spent writing the results out, this proves a challenge to multicore. Parallel version adopted the strategy of allowing each thread to write its part of the results to a different file.
- The best performance used the oldest approach, classic Unix shell scripts along with C
-

SCC 48 core machine

	elapsed time in seconds
1 core doing full concordance	26.17
1 core doing half concordance	13.48
1 core doing 1/8 concordance	5.59
2 cores doing half each	49.0
8 cores doing 1/8 each	36.0
32 cores doing 1/32 each	34 .0
host processor doing it all	1.03
host processor using both cores	0.685

```
# shell script to run on host to run concordance on 32 scc cores
rm /shared/stdout/*
pssh -t 800 -h hosts32 -o /shared/stdout
/shared/sccConcordance32
cat /shared/stdout/* |sort > WEB.con
```

Why did it not work on SCC

- Too much i/o
- Coms path to disk for the individual cores is poor
- Bandwidth of the PCI link to the chip is slow
- All io has to go through the host anyway

Proposed New Benchmarks

- Nbody problem 1024 bodies under gravitational attraction.
- Image Bluring 1024 by 1024 pixel 24bit colour image
- Mandelbrot set for image siz 2048 8 bit colour resolution.