Success with OpenMP in R package data.table

JSM Vancouver, 2 Aug 2018

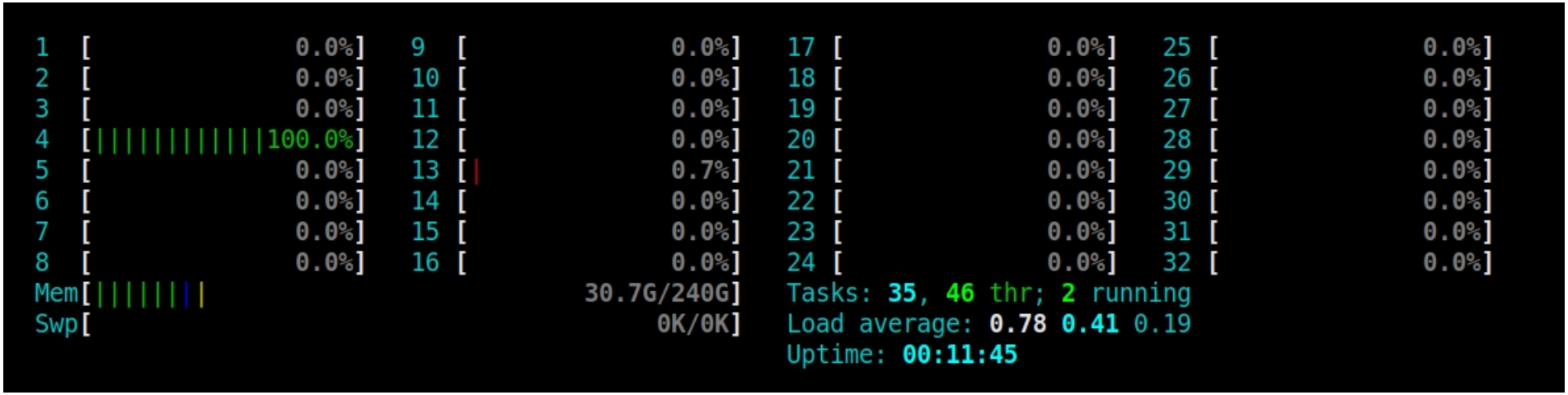
Matt Dowle

H_2O.ai
Machine Intelligence

Abstract

Matt will share his **positive** experience of parallelizing C code using OpenMP in the R package data.table. He will cover several tasks that are complete and released to CRAN: **fwrite, fread, sort and shuffle**. The focus will be on general techniques used (e.g. OpenMP's ordered clause) that may be applied by a wider audience to their fields. **The examples will be from R but the same principles apply in Python, Julia and any environment where OpenMP can be used at C level.** Problems overcome will include: how to halt with error (not thread safe) from a thread, the ability to reorder a character column in parallel even though the R API function SET_STRING_ELT() is not thread safe, how to reason with and tackle the fact that even on a server with 32 CPUs we still typically only have 32K of L1D, a mere 16 cache lines per thread. The talk will contain **OpenMP example code** and one or two references to Ulrich Drepper's 2007 paper: "what every programmer should know about memory".

Motivation

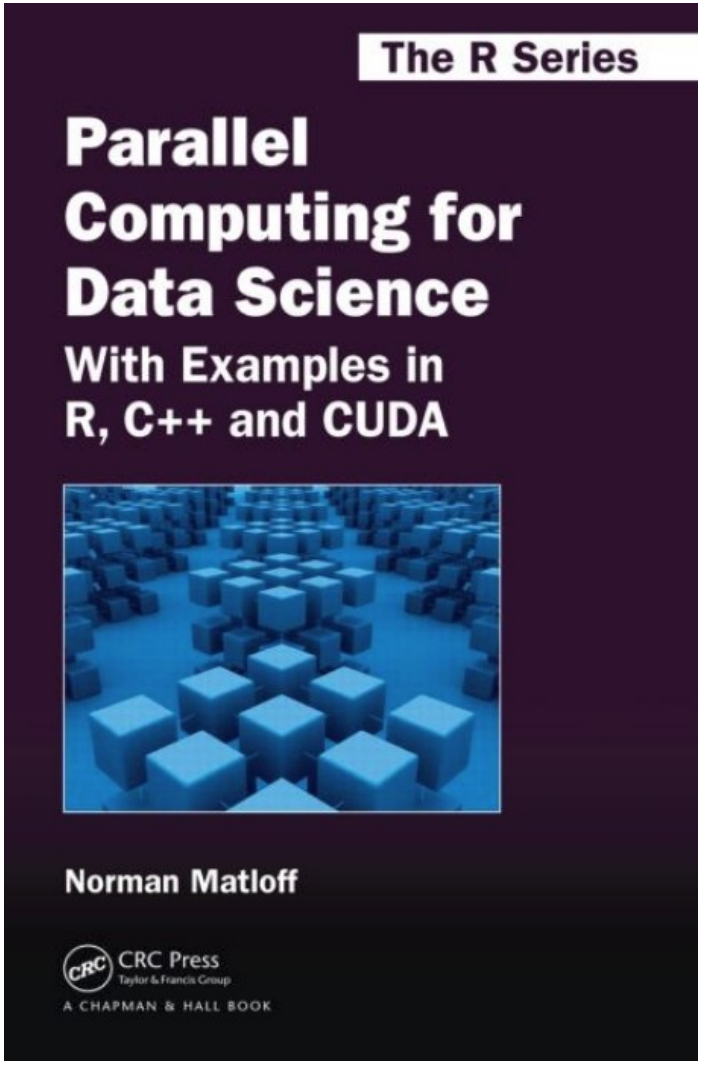

1 [0.0%] 9 [0.0%] 17 [0.0%] 25 [0.0%]
2 [0.0%] 10 [0.0%] 18 [0.0%] 26 [0.0%]
3 [0.0%] 11 [0.0%] 19 [0.0%] 27 [0.0%]
4 [||||||||||100.0%] 12 [0.0%] 20 [0.0%] 28 [0.0%]
5 [0.0%] 13 [| 0.7%] 21 [0.0%] 29 [0.0%]
6 [0.0%] 14 [0.0%] 22 [0.0%] 30 [0.0%]
7 [0.0%] 15 [0.0%] 23 [0.0%] 31 [0.0%]
8 [0.0%] 16 [0.0%] 24 [0.0%] 32 [0.0%]
Mem[||||||| 30.7G/240G] Tasks: 35, 46 thr; 2 running
Swp[0K/0K] Load average: 0.78 0.41 0.19
Uptime: 00:11:45

R data.table	Python datatable
OpenMP multithreaded single process	
No attempt to distribute across multiple machines, and no plan to	
Automatically multithreaded No user-code changes needed	
in-memory	mmap’d disk

Why OpenMP?

- 2014 useR! Conference
- Met Norman Matloff
- He suggested it

- Chapters 4&5: OpenMP



Why not OpenMP?

1. Consensus: “R’s C API is not thread-safe”
2. Consensus: “R’s C API is not thread-safe!”
3. Consensus: “R’s C API is not thread-safe!!!!!!!!”

Matloff 2014: “maybe some things are possible, Matt”

He encouraged me to try.

(1) Multithreaded csv write: fwrite()

- Initial contribution by Otto Seiskari in March 2016
- I parallelized it using OpenMP
- Can **read** R memory multithreaded (*)
- v1.9.8 on CRAN Nov 2016
- https://blog.h2o.ai/2016/04/fast-csv-writing-for-r/

(*) recent caveat altrep

		Laptop SSD 4core/16gb 10m rows		Server 32core/256gb 100m rows		
		Time Sec	Size GB	RamDisk Time	HDD Time	Size GB
`fwrite(DT,"fwrite.csv")`	csv	2	0.8	9	61	7.5
`write_feather(DT, "feather.bin")`	bin	5	1.0	27	75	9.1
`save(DT,file="save1.Rdata",compress=F)`	bin	11	1.2	90	137	12.0
`save(DT,file="save2.Rdata",compress=T)`	bin	70	0.4	647	679	2.8
`write.csv(DT,"write.csv.csv",**)`	csv	63	0.8	749	824	7.3
`readr::write_csv(DT,"write_csv.csv")`	csv	132	0.8	1997	1571	7.3

[**] row.names=F,quote=F

```
#pragma omp parallel num_threads(nth)
{
  char *myBuff = malloc(1MB);
  #pragma omp for ordered schedule(dynamic)
  for(int64_t start=0; start<nrow; start+=rowsPerBatch) {
    if (failed) continue;
    char *ch = myBuff;
    for (int64_t row=start; row<(start+rowsPerBatch); row++) {
      for (int col=0; col<args.ncol; col++) {
        (funs[whichFun[col]])(columns[col], row, &ch);
        *ch++ = sep;
      }
      ch--;
      *ch++ = '\n';
    }
    #pragma omp ordered
    write(f, myBuff, (int)(ch-myBuff));
  }
}
```

(2) multithreaded sort: fsort()

- data.table's radix sort is based on work by :

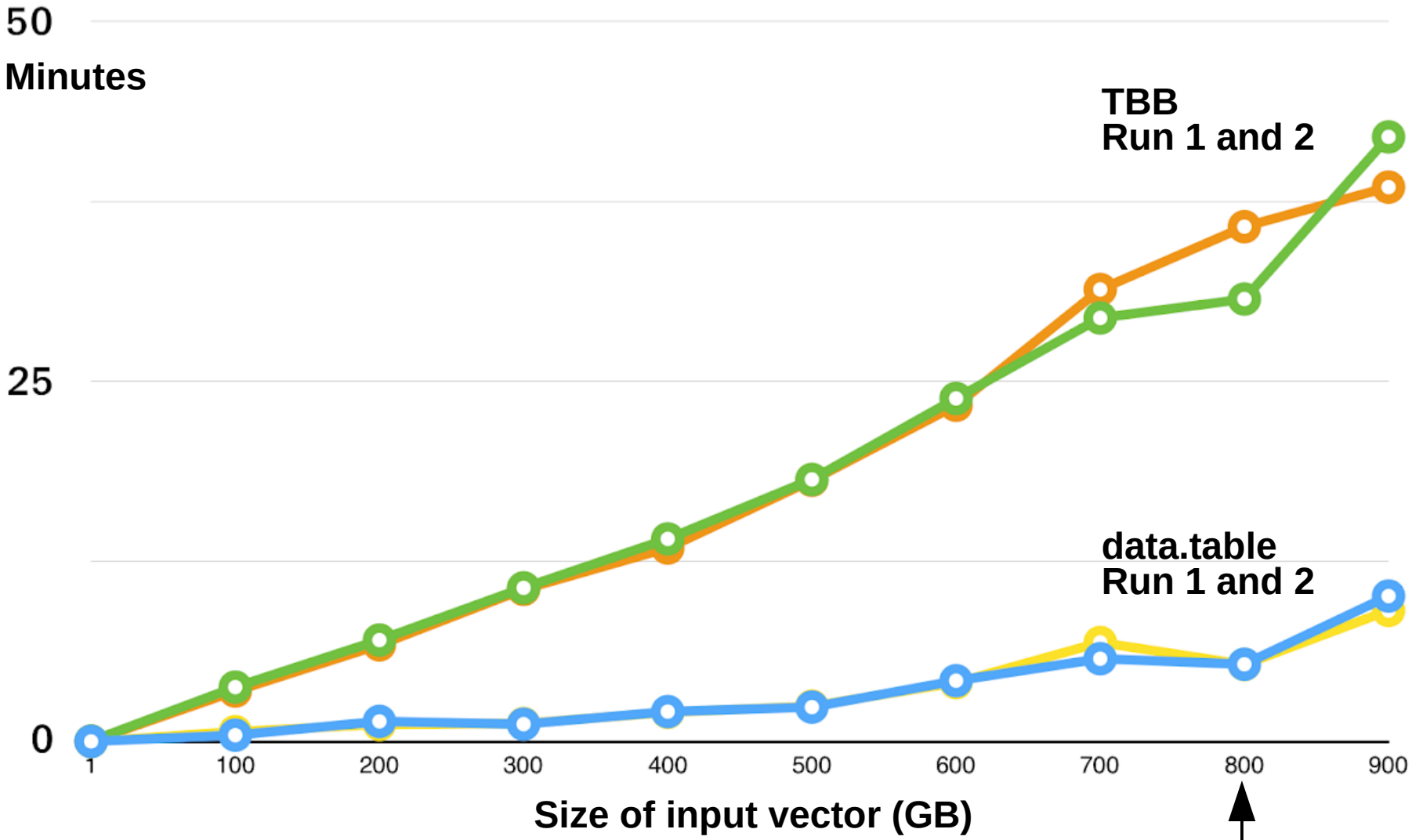
  Pierre Terdiman, 2000

  Michael Herf, 2001

  e.g. IEEE754 bit twiddle and more

- Forwards radix for parallelism

Sort random doubles on EC2 X1 (2TB RAM)
Intel Thread Building Blocks and data.table::fsort()



```
// find minULL and maxULL in parallel

int maxBit = log(maxULL-minULL) / log(2);
int shift = maxBit - 16;  // 2 byte MSB

#pragma omp parallel for num_threads(nth)
for (int batch=0; batch<nBatch; batch++) {
  for (int j=0; j<batchSize; j++) {
    countMatrix[batch, (*(*ULL)x - minULL) >> shift]++;
  }
}

// cumulate countMatrix columnwise
// gather by MSB then descend in cache
// do biggest bins first to minimize waiting for last bin
```

(3) multithreaded shuffle: setkey()

- find the order, then reorder all the columns by that order
- each thread does one column
- int and double are just memory moves ...
- ... and so are CHARSXP in this special case (!)

Conclusion

- OpenMP is great fun
- Speedups exceeded expectations
- Work in progress
- Demo on EC2

github.com/Rdatatable/data.table (R)

github.com/h2oai/datatable (Python)