Macintosh HD:Users:ecstv:Documents:Matt Gussin:DOCUMENTS:Open Day Posters and banners etc:2016:5275_UoS_OpenDayTimetable_PSE Folder:Links:black.epsMacintosh HD:Users:ecstv:Documents:Matt Gussin:Media peices/logos/patterns:Logos:University Logos:university_southampton_black.eps

**Distributed Data Processing**

This is a hands-on activity that allows students to consider the optimal way to solve a computational problem involving the counting of word occurrences across multiple documents.

Time needed: 10-20 mins

**Learning outcomes**

Having completed this activity, students will be able to:

* Demonstrate understanding some of the challenges associated with big data and distributed data processing.
* Appreciate the need to effectively distribute tasks to solve a problem in a timely manner.
* Appreciate the ways in which a simple computational task can be decomposed and (re)composed to maximise utilisation of resources.

**Syllabus mappings**

AQA: 4.4.1.9 Decomposition; 4.4.1.10 Composition; 4.11.1 Big Data (volume; distributed processing)

OCR: 2.1.5 Thinking concurrently; 2.2.2 Computational methods (problem decomposition; divide and conquer)

**Equipment needed**

* Packs of word cards; one pack per group of ~6 students.
* Pen and paper for each student



The aim of this activity is to demonstrate some of the challenges associated with “big” data processing in a distributed environment. The aim of the task is for the students to explore how to efficiently count words across documents, and to then aggregate and sort those counts.

The cards provided with this exercise are designed such that when the counts are produced a hidden message can be read by reading the words in order of decreasing occurrence. Our chosen message is a quote by Alan Turing.

If you want to make your own cards with a different message or formatting, a python script to generate them can be found at:

<https://github.com/jonhare/cs-outreach/blob/master/WordCountingCards/generate_cards.py>

A pdf of the default cards that can be printed on standard business card paper can be found at:

<https://github.com/jonhare/cs-outreach/blob/master/WordCountingCards/cards.pdf>

**Steps:**

1. Split the class into groups of ~6 students and issue a pack of cards to each group.
2. Ask the students to count the number of occurrences of each word across all the cards and to sort the words by decreasing number of occurrences. You might set a time limit, and perhaps encourage competition between the groups to solve the problem in the shortest time.
3. Ask each team to describe the process they took to solving the problem, and discuss the pros and cons of the approaches as a class.

**Typical approaches that might be taken:**

The following list sums up some of the approaches that students might use to successfully solve the task (although not necessarily within any given time limit):

* Single-worker: a single student performs the task and produces the result
  + This doesn’t scale well, or efficiently utilise resource.
* Two-worker: one student reads the cards and another tallies up the counts & performs the sort.
  + This is better than the single worker, but still isn’t an effective use of resource as most of the group is idle.
* Distributed reader workers: cards are split between all but one member of the group. Each person with cards goes through them and reads out the words for the group member without cards to tally up and subsequently sort.
  + The bottleneck here is with the student doing the tallying, and their ability to keep up with the workers. The process still has the overall problem of being serial in nature; even though the workers can read in parallel each word has to be communicated from each worker in series so as not to confuse the student doing the tallying.
* Fully distributed counting: cards are distributed amongst all team members. Each team member must compute the tally for all their cards and sort it. The first member to finish then becomes responsible for aggregating the counts for the other students as they finish.
  + This process is entirely distributed and can run in parallel, with the exception of the final sort. Worker utilisation is close to optimal.
  + Variants of this approach could go even further – aggregation and sorting could be arranged between pairs of students (i.e. the first and second to finish counting could merge and sort their tallies; the third and fourth would do the same and so on. The paired and sorted aggregates are then combined (the first pair of students to finish would combine their results with the second pair, and so on), and this process could repeat until the final result is computed. The advantage of this approach is that the sorting itself becomes a fully distributed merge sort.

In Electronics and Computer Science (ECS), we have created an environment that has inspired an extraordinary level of achievement. ECS is one of the best places in the world to study electronics and electrical engineering, computer science and Software Engineering.

