Coursework COMSM0072 CCBD

Version: 24.11.2020 v1.1

Changes:

24.11.2020 v1.1 – removed requirement to submit source code in appendix

Summary

In this coursework you are being asked to write a report documenting the construction of a fault tolerant cloud system that performs an embarrassingly parallelisable task. Specifically, we would like you to construct a distributed application, introduce failures and demonstrate with measurements that the system continues to operate. Then write a report in the LNCS format.

Weighting: This assessment is worth 100% of your total unit 20 credits.

Set: 13:00. Friday 20rd Nov 2020.
Due: 13:00. Friday 11th Dec 2020.

Submission:

Via blackboard submit one pdf containing:

- an 8-page report in LNCS format (10% penalty for every partial page over) including in the appendix a link to the source code in github repository(s) in the github organisation ccdb-uob including deployment instructions of your application and a tag cw_ccbd_2020.
- A zip/tarball archive of all source code required to run your application and perform the analysis.

In this document we provide a detailed explanation of the task, and the different ways you could go about solving it and the approach to marking.

A Fault Tolerant Cloud Application

In the first lectures we spoke about embarrassingly parallelisable tasks that can leverage cloud capabilities of scalability and on-demand.

In this coursework we ask you construct a cloud application that concurrently works on an embarrassingly parallelisable task of your choice. We would then like to demonstrate that the application provides another important property of cloud systems – fault tolerance. This system then needs to be empirically evaluated in its base performance (when no faults occur) and when you introduce faults in various system parts. The description of the system and the evaluation of its performance will need to be documented in a report in the LNCS format.

This project has a variety of extension points. Below we provide illustrate a basic solution. Together, this coursework offers every student the ability to pass the coursework by applying the concepts they have

been taught plus some self-study. Higher achieving students can demonstrate their abilities with more advanced concepts that require a greater degree of self-study.

Task

You are free to choose the task that your application carries out yourself. Importantly, it needs to demonstrably benefit and exercise scalability of your cloud application.

One of the most easily implemented tasks is brute force password cracking. Kenny Muli in an article on medium states that a "8-GPU rig can crack an 8-character, MD5 hash password within 4 hours by just random guessing." An simple task would be to build a brute force password cracker. The performance of the running system could be analysed in terms of cost per average password cracked, for example.

However, there are many other (more ethical) embarrassingly tasks out there. You are free to chose, but do provide a motivation for your choice in the report.

Choice of Cloud

Your application needs to run in your own application cluster in the cloud.

You can freely chose what cloud to implement the solution on. AWS Educate is a solid choice, given that you have $150 of credits available. If it looks like you happen to exhaust those, please get in touch with us early enough that we can get a top-up arranged.

Something you need to be aware of resource limits with the educational programs. AWS Educate has these as well. Check out these limits so you don't get surprised.

Application Architecture

You are free to compose your application as most appropriate for the task. We expect that you will build a system that from decoupled components in some way. The greater the degree of components that are outsourced to the cloud provider the easier it might be, but at the expense that you have less opportunity to demonstrate your ability to built scalable applications yourself. So there is a trade-off.

One of the easiest solutions might be a system using a task queue with worker nodes on top of Elastic Beanstalk. If you choose a PaaS system such as Beanstalk the management of the cluster is outsourced to the cloud provider making it much easier for you to focus on the compute task. For instance, you would scale your cluster up and down as a group without addressing specific instances. Because of the delegation of management tasks, if you do decide to use PaaS you'll miss the opportunity to demonstrate your ability to apply many of the concepts that we discovered in this unit, from use of VMs all the way to synchronisation in distributed systems. Thus, you will miss out on some marks awared for this higher challenge.

One other decision is where your application controller is located – either on the cloud or on your own computer. At a minimum you can use tools such fabric (see activity to week 5) to deploy your cluster from your laptop and have the controller monitor the state of your cluster from your local computer. If you do run your application from your local computer with AWS Educate you will have to take into account that your auth keys are only valid for a 2h session at a time.

Once your application is running you should take performance measures. This can include aspects such as the degree of variability of throughput or network latency between nodes ("are all nodes equally

fast”). As you monitor the performance of the cluster your controller has to bring failed instances back online. This will take some time. You can develop a regression model of progress degradation over instance failure rates (you can introduce your own delays to make this more pronounced).

Finally, could decide to develop their own task queue that provides failover. This could include the use of Zookeeper. A natural extension point would then be to include zookeeper in the list of chaos tested instances, or the queue.

Fault Injection

In order to demonstrate fault tolerance you will have to disrupt some components of your architecture. Netflix has coined the term “chaos testing” to refer to randomly turning off nodes in an application.

How you go about doing this is up to you. As with the other aspects there are levels of complexity to this as well. In the most basic approach you might ask your controller to randomly terminate worker nodes (that then need to be rescheduled after some delay, maybe).

Github

Your entire source code must be included in one or more repository in the ccdb-uob organisation. You need to provide specific deployment instructions in the README.md file in the repository.

The repository must be private.

You also must create a tag of the repository with a timestamp of or before 13:00. Friday 11th Dec 2020 labeled cw_ccbd_2020.

You also need to include the source code of your cloud application, any local controller code and code used in the analysis of the results.

Report

Your paper should be formatted according to the Springer Lecture Notes in Computer Science (LNCS) conference-proceedings format, details of which are available here: http://www.springer.com/computer/lncs?SGWID=0-164-6-793341-0

Your paper should be no longer than eight pages in LNCS format, including all figures, references, and any segments of code you include to explain your application. You can write 8 pages of text and graphs and tables explaining your work. The appendix is not included in the page count. It should include the link to your github repository on github.com/ccbd-uob.

Every partial page over the 8 page limit will incur a 10% mark penalty.

Writing Tips

Here we list miscellaneous resources that may be useful across the degree; they're offered in no particular order.

- Dave Cliff's Accessibility score: Low Click to improve TipsOnWriting is a sequence of tips on how to avoid some common problems when writing
- The classic William Strunk's *The Elements of Style* in UoB Library: https://bris.on.worldcat.org/oclc/854969873.
- Steven Pinker's lecture on writing, at the Royal Institution.

- Larry McEnerney from the University of Chicago gives very good advice on postgraduate-level technical writing in this lecture.
- How to Speak. In this video MIT's Patrick Winston gives a one-hour tutorial in how to speak. It is well worth watching.
- Professional proof-readers and copy-editors can help improve your writing. Examples of specialists in academic reading/editing are: Typewright, Margaret Towson, and TheFilthyComma, although lots of other providers are available.

Marking

Group Work

You have already told us what groups you would like to work in. If this changes in any way you must tell us immediately. No changes to group composition can be accepted after Monday 23.11. Noon.

If any group member experiences external difficulties that affect his ability to contribute to the project, the group must inform us immediately in writing.

If one of your group members is subject to circumstances that qualify as EC, the regular UoB EC process will apply. Other group members will be assessed, taking into account the reduced group size, discount in proportion to the time available to the assessment submission deadline.

Marking Bands

Below marking bands with maximum possible mark range achievable given approximate scope of work.

+80% Outstanding report. Extensive exploration and analysis demonstrating deep understanding and reading outside of the lectures.

70 - 80% Excellent report. Wide set of distributed system components used. Some novelty in design choices.

60 - 70% Report of correct length detailing motivation, extensive performance analysis, IaaS or other with full automation.

50 - 60% Report of correct length detailing motivation, PaaS setup with manual failure injection on workers, example task, basic statistical analysis and results.

<50% Report is not at an appropriate standard. Objectives of the assignment have not been (fail) demonstrated.

Marking Rubric

	Good	Average	Poor

Quality of experiment, writing and presentation, including results. [30%]	Writing is clear and economical: - clear structure – intro, main, summary present and effective - absence of spelling errors - phrasing of high quality. Results are clearly presented. The clearly summarises the essence of the project and indicates the results. The project is well motivated. Spelling errors are absent. Key terms are defined and accessible to a reader with a broad general knowledge of cloud computing. Specific technologies are introduced with explanation. The text is realistic in its description of the system and its evaluation. It provides discussion of technological choices made. It finishes with an interpretation and closing summary.	The writing is in many aspects that of a good exept for minor omissions and lack of polish in phrasing.	The report may be difficult to comprehend due to a lack of clarity in structure or poor phrasing. The report may suffer from incorrect spelling. Poor phrasing my obscure the meaning. Terms may lack introduction and the structure might be sufficiently cohesive.

	All diagrams are clearly readable based on appropriate font size. The captions are complete and provide interpretation. All references are consistent. The diagrams add value in the form of evidence or summary. Color choices and hashes are appropriate for black and white printing.	Most of the attributes of good diagrams.	Significant shortcomings wrt to size of illustrations or fonts. Diagrams might be poorly referenced or do not support to what was said in the text.

Scalability (30)	Provides clear description of the architecture and how it supports scalability. Includes appropriate empirical evaluation of the scalability behaviour. - good application of scalability - manages scaling compute herself (IaaS) or - when scaling of compute is arranged by framework (Kubernetes, PaaS) then competency in scaling is elsewhere, e.g. use of scalable storage, application architecture (Load balancers, queues, service discovery)	The system architecture is scalable through appropriate interfaces between components. There scalability performance of the application was adequately evaluated.	The architecture of core components is monolithic. No evaluation of the scalability of the system is present.
Chaos/Failover (20)	The system's ability to withstand failures has been robustly and comprehensively demonstrated.	Component failure was introduced in a small subset of the application or the observed data was incomplete.	No demonstration of failure resilience was given, or the system failed on component failure.
Performance (15)	The system as well as infrastructure performance was observed and	Some performance metrics were provided but these were incomplete	No or minimal performance data is provided and interpreted.

	analysed comprehensively. Appropriate statistical analysis chosen and conducted correctly. Correct conclusions clearly drawn from the analysis and explained.	(ignore task or infrastructure) or the interpretation was superficial.	
Parallel Task (5)	The chosen task is embarrassingly parallel and interesting/useful.	The task parallelisability is constrained in ways that negatively affect it's use in this application.	The system design is not parallelisable or it vastly overambitious in scope.

Deadline

The deadline for submission of all optional unit assignments is 13:00 on Friday 11th of December. Students should submit all required materials to the "Assessment, submission and feedback" section of Blackboard - it is essential that this is done on the Blackboard page related to the "With Coursework" variant of the unit.

Time commitment

The expectation is that students will spend 3 full working weeks on their two assignments. The effort spent on the assignment for each unit should be approximately equal, being roughly equivalent to 1.5 working weeks each.

Academic Offences

Academic offences (including submission of work that is not your own, falsification of data/evidence or the use of materials without appropriate referencing) are all taken very seriously by the University. Suspected offences will be dealt with in accordance with the University's policies and procedures. If an academic offence is suspected in your work, you will be asked to attend an interview with senior members of the school, where you will be given the opportunity to defend your work. The plagiarism panel are able to apply a range of penalties, depending the severity of the offence. These include: requirement to resubmit work, capping of grades and the award of no mark for an element of assessment.

Extenuating circumstances

If the completion of your assignment has been significantly disrupted by serious health conditions, personal problems, periods of quarantine, or other similar issues, you may be able to apply for consideration of extenuating circumstances (in accordance with the normal university policy and processes). Students should apply for consideration of extenuating circumstances as soon as possible when the problem occurs, using the following online form:

https://apps.powerapps.com/play/3172b943-0956-4b88-bf3d-3f37871d1170?tenantId=b2e47f30-cd7d-4a4e-a5da-b18cf1a4151b

You should note however that extensions are not possible for optional unit assignments. If your application for extenuating circumstances is successful, it is most likely that you will be required to retake the assessment of the unit at the next available opportunity.

Implications of UK "travel window"

The UK Government will be instigating a "travel window" to allow students return home from University once UK national restrictions have been lifted. This will take place between the 3rd and 9th of December and as such, will occur during the optional unit assignment period. Students whose work is significantly impacted by extended periods of travel and quarantine during this period should apply for extenuating circumstances. As discussed previously, extensions are not possible for optional unit assignments. If your application for extenuating circumstances is successful, it is most likely that you will be required to retake the assessment of the unit at the next available opportunity.