Coursework COMSM0099 CCBD

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 15 credits.

Set: 13:00. Thursday 17th Nov 2022.
Due: 13:00. Thursday 8th Dec 2022.

Submission:

Via blackboard submit one pdf containing:

- an 8-page report in LNCS format (pages after the 8 page limit will be ignored) 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_2022.
- 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 choose, but do provide a motivation for your choice in the report.

When it comes to implementing the task please note that the purpose of this coursework is to allow you to demonstrate learning of Cloud Computing and Big Data methods. That means that sophistication of the jobs that the worker nodes carry out that falls outside of Cloud Computing and Big Data; and using AI or ML will not be rewarded.

Choice of Cloud

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

You can freely choose what cloud to implement the solution on. AWS Academy is a solid choice, given that you have $100 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 Academy has these as well. Check out these limits so you don't get surprised.

Avoid using services that may require you to pay. We will not be able to reimburse you for any money spent.

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.

Equally, if you try to create an entire system from scratch using only VMs, you are demonstrating a good ability to create cloud applications, but you may find that you struggle to implement scaling and fault tolerance in the time available.

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 awarded 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 Academy you will have to take into account that your auth keys are only valid for a 4h 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 a 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. Thursday 8th Dec 2022 labeled cw_ccbd_2022.

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.

If you have not been given access to a repository in the ccdb-uob organization, please contact jed.preist@bristol.ac.uk as soon as possible.

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.

Everything over the 8 page limit will be ignored.

Report Format

The **Abstract, Introduction** and **Conclusion** (in bold) are compulsory sections. The rest can be adapted to suit your preferences when you are writing the report.

Abstract

A short summary of your report, covering your chosen task, why cloud computing is an appropriate solution for it, a summary of your architecture and the results and conclusion of your testing.

Should not be more than about half a page long.

Introduction

A clear description of your chosen task, why your task is important and why cloud computing is an appropriate solution. Should also include a summary of the general principles used in cloud computing.

Application Architecture

A description of your application architecture, including justification for design decisions and appropriate diagrams.

Testing

A description of the methodology and results of the performance and reliability testing carried out.

Analysis

An interpretation of the results produced during testing, making use of appropriate charts and diagrams.

Conclusion

A summary of the content of the report, and a description of potential future improvements to the application.

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 28th of November 12:00.

Only work in a group with people you trust. All group members will get the same mark. We aren't able to deal with disagreements between group members.

If you work in a group we will be looking for an appropriate increase in the complexity of the project. A group of two people is expected to produce a system twice as complex as someone working individually, for example. This obviously won't be perfectly equivalent in difficulty to a person working individually, but we will do our best to avoid any significant differences.

If any group member experiences external difficulties that affects their 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.

Band	Criteria
70+%	**Code** • A reasonably complex parallelised application has been produced. • The system is capable of scaling to meet both normal and unusual/malicious changes in demand. • If scaling is managed by a framework, then it is well supported by the application architecture. • The system is capable of handling severe failure of its components. (e.g. queue failure, multiple partitions) **Report** • Key terminology is used well throughout, with clear definitions. • A good description of the application architecture is given. • Well thought out justification for design decisions is given, and potential alternatives are considered. • Good performance and reliability analysis is present, highlighting shortcomings and potential improvements. • Analysis methodology is clearly explained and justified.

60 – 70%

Code

- A parallelised application of reasonable scope has been produced.
- The system is capable of scaling to meet normal changes in demand.
- If scaling is managed by a framework, then some attempt is made to support it with the application architecture. (e.g. using load balancers/queues.)
- The system is capable of handling reasonable failures of its components.

Report

- The report makes appropriate use of key terminology, including giving clear definitions.
- A good description of the application architecture is given.
- Good justification is given for design decisions.
- Decent performance and reliability analysis.
- Analysis methodology is clearly explained.

50 – 60%

Code

- A simple parallelised application has been produced.
- The system is capable of some scaling **OR** has scaling entirely managed by the cloud frameworks.
- The system has simple fault tolerance **OR** has fault tolerance entirely managed by the cloud frameworks.

Report

- Key terminology is used.
- A simple but clear description of the application architecture is given.
- Basic justification is given for design decisions.
- Some appropriate performance and reliability analysis is present.
- Some attempt is made to explain analysis methodology.

<50%	**Code** • The chosen task is inappropriate for use in the cloud **OR** the chosen task is either trivially simple or far too overambitious. • Little or no attempt at parallelising the task has been made. • Few or no cloud frameworks are used. • The system does not scale **OR** little or no attempt is made to demonstrate scaling. • The system is not fault tolerant **OR** little or no attempt is made to demonstrate fault tolerance. **Report** • Little or no understanding of key terminology is demonstrated. • Description of the application architecture is unclear. • Little or no justification is given for design decisions. • Analysis is inappropriate or not present.

Deadline

The deadline for submission of all optional unit assignments is 13:00 on Thursday 8th 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 of any significant length are not possible for optional unit assignments. If your application for extenuating circumstances is successful, you may be required to retake the assessment “You should note however that extensions of any significant length are not possible for optional unit assignments. If your application for extenuating circumstances is successful, you may be required to retake the assessment of the unit at the next available opportunity (e.g. during the summer reassessment period)of the unit at the next available opportunity (e.g. during the summer reassessment period).

Available Support

We will be answering questions on the COMSM0099 Teams channel. Please do not post your code on Teams, you might end up part of a plagiarism case if someone copies it.

There will also be a two hour in-person drop-in session each week. This will be in MVB 1.15 from 1-3pm on Tuesdays.

Q&A

- **Level of sophistication of a certain feature**: Polishing a feature will provide you with increasingly diminishing returns in terms of marks. By investing time into making a feature perfect and compromising on other areas (see marking rubric) you might lose marks compared with a more well-rounded piece of work.
- **Can we use S3, Kubernetes, XYZ?** You can use any technology that is reasonable for your system. You will make a choice of technology/solution based on some literature review (includes blog/tech articles on the web) and your own conceptual understanding. What matters is that in your report you include this reasoning and argumentation.
- **Is my task appropriate?** Your task should have some utility to someone that you should illustrate in the introduction/background. More importantly, the task should genuinely benefit from increasing the number of nodes in your cluster. If your task can be solved by a small number of nodes already, then you are potentially depriving yourself of the ability to evaluate your systems performance under failure.