1 List of Assignments Cloud Computing: CS G527

Name: SANTONU SARKAR (PI)
Date: August 27, 2016

Instructions

Students can choose either a research project (1.9 and 1.10) or assignments. There are total 17 students in the class. A student has an option of choosing ONE research project and complete it by himself OR he should form a group with another student to do assignments and a seminar. The following rules need to be maintained:

1. If a student chooses a research project, he needs to submit a research report and the solution/tool- followed by a presentation.

2. If a student chooses a class project, she needs to form a group with another classmate and choose two projects and ONE seminar topic. The seminar topic needs to be jointly prepared and presented.

3. There are 8 assignments and 8 seminar topics. Each group should choose a unique seminar topic from the list. Each group should pick up two assignments in such a manner that not more than TWO groups do the same assignment.

4. For research projects, the rule is simple. One student picks one research project which involves literature survey as well. The literature survey becomes her seminar topic.

5. The seminar report will be checked against copying from internet sources. If the degree of plagiarism exceeds 20% you will be awarded zero marks in the seminar. The same rule applies for research project or assignments.

Marks

40% for the assignment/implementation and 10% for the seminar (or background study)

1.1 Compute Nth Prime number in a load-balanced manner

Implement a service that calculates N'th prime number. You can restrict the range of prime numbers to a large-enough value (e.g., 10^{16}). Use the algorithm given in `https://primes.utm.edu/nthprime/index.php`, and use a large enough range so that a single-user query takes around 1 second. Use naive algorithms so that the problem becomes compute/IO-intensive. Here the objective is to implement the smartness of the load-balancing and allocation scheme to make the problem scalable and efficient. Try to use Amazon EC2 VMs for load-balancing.

1.2 Amazon VM benchmarking

For this assignment, various Amazon EC2 instances (*micro, small, medium, and large*) needs to be benchmarked. Login to Amazon EC2 using your credential. Once logged in to the Amazon Management console, you will have access to all of the Amazon Web Services. Use the EC2 dashboard to create launch virtual machine instances as required. For uniformity, choose Ubuntu LTS OS and go through `http://www.youtube.com/watch?v=JPFoDnjR8e8` for a video guide on how to launch an instance and log into it.

Perform some benchmarking on *micro, small, medium, and large* instances. For each of these instances you need to perform one of each of the following types of benchmarks:

1. CPU: Use any one of the benchmarks like `Dhrystone, Whetstone, SuperPi, Linpack`
2. IO: Use any one of the benchmarks like `bonnie++, iozone, hardinfo`
3. CPU/IO mixed: You can choose either i) Build Linux OS, OR ii) untar a large archive file

For each instance type, you should bring up 5 instances of that type and execute your selected benchmarks at least 5 times. Hence, for each instance type, you will need to run $5 \times 5 = 25$ samples.

You should write a script (Perl/Python/shell) to run and collect the testing results. Once you are satisfied with your script, you should then make an AMI (Amazon Machine Image) of your virtual machine as described in `http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/creating-an-ami.html` Once you have your own custom AMI preloaded with your benchmarking scripts, you can then just create different sized instances of this image to execute and collect your benchmark data.

In your report, answer the following questions:

1. Describe your overall approach and method to benchmarking the Amazon EC2 instances.
2. What benchmark applications did you choose and why? Specifically, what do these tests do and what do their results tell us about the system?
3. What issues if any did you have in performing these benchmarks? How did you overcome these challenges?
4. What are the CPU, memory, and disk characteristics of each of the VM instances? Include a table that summarizes these properties.
5. What were the results of your benchmarks? For each benchmark show a graph or plot of the results and describe what happened. Were any of the results surprising, why or why not?
6. Given the results of your benchmarking and what you know of virtualization, what sort of applications are well-suited for Amazon EC2? What applications would not be a good fit?

1.3 Inside Virtualization: Understanding binary translation

Install a small OS (PintOS `http://web.stanford.edu/class/cs140/projects/pintos/pintos.html`) on QEMU virtualized infrastructure. For this project, the students are expected to read:

1. Read QEMU, a Fast and Portable Dynamic Translator, Fabrice Bellard.
2. Read QEMU Emulator User Documentation
3. Read Intel Reference Manuals
   (a) Basic Architecture
   (b) Software Developer's Manual Vol. 1 and 2
4. Download QEMU and build it
5. Download pintos.tar.gz and type 'make' in the threads/ directory to build kernel.bin
6. Use the command: `'pintos/src/utils/pintos --make-disk=guest.dsk --hardware --kernel pintos/src/threads/build/kernel.bin --loader=pintos/src/threads/build/loader.bin'` to build `'os.dsk'`
7. Run the pintos kernel as a guest OS on QEMU

The assignment is about counting the number of instructions executed by the VMM:

- Read the QEMU paper and understand how the dynamic translator inside QEMU works
- Change the dynamic translator to count the number of instructions executed in one execution of pintos, and print this value at the end of the execution.
- Change the dynamic translator to count the number of increment ('inc') instructions executed in one execution of pintos, and print this value at the end of the execution.
- Count the number of 'call' instructions and the number of 'ret' instructions.
- Report how many call instructions you see, and how many ret instructions you see. Why is there a mismatch between the two counts?

#Header file	PageRank	List of file locations
`stdlib.h`	1.0	`stdio.h main.c`
`string.h`	0.8	`main.c`

1.4 MR based Twitter Analysis

Use Twitter APIs to download 50K tweets. Copy the tweets file in a Hadoop HDFS. Find all sets of sentences that are 90% similar to each other, i.e. 90% of the words match. Formulate using MapReduce and implement in parallel. Once you implement, run your program in

1. Hadoop cluster in A200 Lab and

2. MR cluster from Amazon using your credential (select a cluster with whatever configuration you feel appropriate)

In your report, specifically describe i) explain why you chose the configuration in Amazon and ii) give a comparative study of the difference in performance using bar graph/charts.

1.5 MR Based Inverted Index Creation

Implment a MapReduce based application in Java that constructs the inverted index of the Linux 3.8 source. Construct the inverted index for functions (i.e. any string that starts with "function name(". The end result of your Inverted Index MapReduce application should be a mapping from word to a list of files that contains that word. Here is an example of what your output should look like:

```
# Function    # List of file locations
printf       stdio.c main.c
sbrk        stdlib.c string.h flipdemtables.c
```

Run the application on a Hadoop cluster. Increase the number of mappers - 10, 50, 100, 200, 500, 1000.... Plot a graph showing the time taken to perform the application against the number of mappers. Discuss the anomalies of the graph if you find any.

1.6 MR Based PageRank Calculation

You are to compute the PageRank of the header files in the Linux 3.8 source code. Scan both the headers and source code files for `#include` and use these imports as dependency links to generate the PageRank score for each header file. You can read the original PageRank paper by L.Page et. al "The PageRank Citation Ranking: Bringing Order to the Web. Technical Report 1999-66, Stanford InfoLab, November 1999. Previous number = SIDL-WP- 1999-0120". A MapReduce implementation in Java is given in: `http://hadooptutorial.info/mapreduce-use-case-to-calculate-pagerank/`.

Your output should look like:

You need to run your program in the Hadoop cluster in the A200 Lab and then run the same on Amazon Elastic MapReduce (`http://docs.aws.amazon.com/ElasticMapReduce/latest/DeveloperGuide/emr-get-started.html`). Compare the performance of the program by spawning the identical number of mappers/reducer tasks in both the cases. The comparison is meaningful only when your code is producing an identical result in both the cases.

1.7 Extending a Checkpointing System

DMTCP (Distributed MultiThreaded Checkpointing) transparently checkpoints a single-host or distributed computation in user-space – with no modifications to user code or to the O/S. It works on most Linux applications, including Python, Matlab, R, GUI desktops, MPI, etc. It is robust and widely used (on Sourceforge since 2007). Visit `http://dmtcp.sourceforge.net/index.html` for details.

One of the major problem with the checkpointing package is the "Closed World Assumption"- which assumes that the application does not interact with any other external system or service. When it does, for instance, interact with an UI, send packets through network, open and read/write files, access database and so on, a checkpoint system does not know how to handle this situation.

Write an extension to DMTCP that implements a heuristics to deal with an application accessing a database. Implement the plugin where the application accessing a database is able to disconnect from the server before checkpoint and reconnect after resume/restart. Remember that the plugin works on top of DMTCP, the application code is not aware of such a plugin. So ideally you shouldn't be changing any application code to test your approach. To write a plugin read the tutorial `https://github.com/dmtcp/dmtcp/blob/master/doc/plugin-tutorial.pdf`

1.8 Writing a fault-tolerant program

Netflix's Hystrix (`https://github.com/Netflix/Hystrix/wiki/How-it-Works` and many other associated links) is a framework that implements various fault-tolerant design patterns taught in the class. Use this framework to implement a sample application and deploy the application in the cloud (Amazon EC2 VM). Load this application using an open source load generator and observe the behavior of the application. Collect the statistics of the throughput of the application and present it in the report. Specifically, report how the framework can achieve the fault-tolerance. To compare the fault-tolerance, you need to run this application with and without Hystrix.

1.9 Research Project1

1.9.1 Prior Art Analysis and Presentation

There are various research papers related to provisioning, bidding, allocation and deallocation of spot VMs in Amazon. Specifically look at: (i) Dynamic

Models that predict spot VM lifetimes statistically and empirically (ii) How Specific Applications leverage Spot instances while still building some reliability.

Summarize the research contributions under three to four buckets. Specifically articulate your own point of view on the defining prediction model for spot bid-price that will give maximum uptime by analyzing the empirical distribution of expected spot instance lifetimes. The report should have Introduction, Spot-Instance (advantages and problems), Survey of the existing work (i and ii), your point of view, references.

1.9.2 Implementation of a Predictive Model

Create new algorithm that will analyze the spot instance history data for various availability zones made available by AWS and also experimentally provision spot instances via APIs and report the following:

1. Observe if the fluctuations (intrinsically supply-demand witnessed by the availability zone in question) follows any statistical distribution.

2. Observe the mean/median/mode times (in hours) that a spot instance survives in a chosen set of availability zone.

3. Articulate a point-of-view on predicting the minimum spot bid price that will result in the maximum uptime (see the section on prior art that looks at empirical distribution of expected spot instance lifetimes.)

4. Compare the movement of spot cost relative to reserved cost assuming various allocation rates for the latter.

The 90-day spot pricing history on a per-availability-zone basis is available from `http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-spot-instances-history.html`.

1.10 Research Project2

In this project you are supposed to analyze a running as-is instance of the application-to-be-migrated and determine:

1. Deployment strategy: Whether it is suitable for this application to run on a Virtual machine or it is better to run the application in a container- such as running on Google app-engine or oracle PAAS

2. Candidate deployment architecure specification- in terms of VM size (in terms of vCPU, vMEM and storage), network bandwidth (OR container specifications), with concomitant load-balancers and generation of auto-scaling rules (including determination of the minimum pre-scaled size of the compute environment)f or the provisioned compute- w/o violating the original applications response time and throughput.

3. The feasible deployment with least cost

1.10.1 Prior Art Analysis and Presentation

As a part of the prior art analysis, you should study existing approaches in this regard, their strengths and weaknesses.

After your experiment, based on the operational data analysis (using splunk tool, for instance) you should also write your point of view on the following:

1. Is there a way to indicate that an application is suited for containerization (deployment on Docker containers) vis-à-vis virtualization (deployment on VMs).

2. Is there a way to conclude if a monolithic application can be advantageously re-factored into a System-of-Engagement (SoE) front-end and a System-of-Record (SoR) backend, along with application-specific guidance? SoE is that part of the application where end-users interact with IT; SoR is usually the storing and secure processing of back-end data. SoE tends to be hosted off-premise on public clouds due to its unpredictable nature - SoE needs to grow or shrink on demand. SoR, on the other hand, has an affinity to run on-premise behind firewalls.

1.10.2 Implementation of a Predictive Model

1. Emulate a production grade enterprise application deployment by deploying the opensource e-commerce application Opencart (`www.opencart.com`). Opencart is a 3-tier application – MySQL constitutes the DB tier; Apache Tomcat has a HTTP server (web tier) and an application server (app tier) that hosts the opencart e-commerce service written in PHP. There are other enterprise applications as well such as `http://blog.capterra.com/best-free-open-source-ecommerce-software-solutions/`.

2. Deploy the Web/App tier in one physical server and the DB tier in another physical server. Install the load generator in a third server, virtual or physical. Make sure that the systems are not under-provisioned.

3. Write down a set of assumed non-functional requirements- called NFRs- (# of simultaneous users, # of simultaneous transactions, throughput, all-the-above as a function of time, for example, max users on Sundays, least users on Mondays). Use LoadStorm (loadstorm.com) or Blazemeter (blazemeter.com) to load-test the opencart deployment conforming to the recorded NFRs for a period of 2-4 weeks.

4. Access available log files from the physical server, the Apache application server, and the opencart software over a period of 2-4 weeks.

   1. CPU and Memory Utilizations imposed by the application
   2. Storage requirements in terms of space and IOPs
   3. In-bound and out-bound network throughput required by the application

4. MTBF (mean-time-between-failure) of the application, and
5. Transaction traffic as a function of time over a period of 2-4 weeks.

Use an opensource log analytics tool such as splunk (a free trial version is available at splunk.com) to convert both the above data into the following information/knowledge:

   (a) The minimum and maximum per-server CPU power that the application needed in terms of a processor-independent benchmark such as RPE2 (example, 2000 RPE2 at a particular time; 500 RPE2 at another time, and in-between at all other times).

   (b) Associated memory requirements.

   (c) A determination on the variables that cause the compute/memory demand to grow or shrink (for example, CPU load is maximum on Sundays and minimum on Mondays; or the app-tier load increases as a function of network throughput on the web-tier; and such).

   (d) The MTBF (mean-time-between-failure) of the application. This would translate to the uptime expectation on the application.

5. Now you perform the analysis to determine the three objectives stated at the beginning of this project.

1.11 Research Project3

In this project you are supposed to analyze the same running as-is instance of Research Project2 but with a different objective.

1.11.1 Prior Art Analysis

Read papers related to hybrid and the notion of cloud brokerage. One recommended paper is "Dynamic Cloud Instance Acquisition via IaaS Cloud Brokerage"- IEEE TPDS-2014. Read similar papers related to Cloud brokerage. Identify the research gaps.

1.11.2 Implementation

Here you need to do two experiments.

Impact of Security on VM distribution

1. Modify opencart (if necessary in the source code) so that the MySQL database (DB tier) can be an external database. Do the following deployments:

   (a) Install the Web and App tier of opencart on the AWS public cloud, and the data (DB) tier on an on-premise server.

(b) Install Web and App tier of opencart on AWS and the database as Oracle instance running on Oracle cloud.

2. Use LoadStorm (loadstorm.com) or Blazemeter (blazemeter.com) to load-test the Web tier of the opencart deployment on AWS. Increase the load until the throughput demand between the Web/App and DB tier cannot be met due to network latencies (max-throughput).

3. Encrypt the traffic between the App/Web tier and the DB tier using the algorithms listed in `http://www.cryptographyworld.com/algo.htm`. For each encryption algorithm and for each level of throughput (0 to max-throughput), measure the throughput overhead due to encryption.

4. Use Oracle cloud to put opencart in Java containers and user Oracle database instead of MySQL.

Write a report with insights on which encryption algorithm should be used on data-in-motion when; and the trade-offs (degree of security offered by the encryption algorithm vis-à-vis the associated throughput overhead). In your report do a comparative study of the two deployments.

Optimal VM Allocation Here you need to simply recreate the experiments of the paper published in IEEE-TPDS 2014. The dataset is already available.

2 List of Seminar topics

Students who have chosen the research projects need not select any seminar topic.

1. Green and Efficient Cloud Computing, - from Sp. Issue of IEEE TCC 2016. Read papers like: "eCope: Workload-Aware Elastic Customization for Power Efficiency of High-End Servers", and "Energy-Efficient Virtual Machines Scheduling in Multi-Tenant Data Centers"

2. Cloud Inter-operability and Federated Clouds

3. Latest advances in VM migration: Start with the paper "Sandpiper: Black-box and Gray-box Strategies for Virtual Machine Migration". Timothy Wood, Prashant Shenoy, Arun Venkataramani, and Mazin Yousif, Usenix 2007 and create report based on at least 5 to 6 recent papers (2012-2016)

4. Cloud Security: Start with the paper- Sarkar et al. ACM Computing Survey Vol 47(3) 2015

5. Capacity Planning and Performance issues in Cloud Computing. Discuss latest advancements on auto-scaling.

6. Virtualization technology to enable Internet of things: Read the book "Internet of Things, Principles and Paradigms", chapter 6 and 7

7. Robustness and security issues in Internet of things: Read the book "Internet of Things, Principles and Paradigms", chapter 10 and 11

8. Recent comparative study of VMWare ESX with other similar products. The comparative study should first describe and explain what VMWare ESX is, followed by other similar products in the market, and a feature-by-feature comparison of VMWare with other products.

Quality of the seminar will be evaluated based on

- Quality of survey (number of papers and their venues)
- Articulation of the problem (so that everyone understands)
- Existing solutions and their drawbacks- your own viewpoint- not a copy from the paper.