LARGE LAB EXERCISE A

DISTRIBUTED COMPUTING SYSTEMS (400130)

The Virtual Grid System:
Distributed Simulation of Distributed Systems

A. Iosup, A. Uta, L. Versluis

v.1.0, November 3, 2017

A Learning Objectives

1. Implement complex operations of cloud computing in realistic scenarios.
2. Analyze the tradeoffs inherent in the design of cloud-computing-based applications.

This exercise combines elements of Chapter 4 (communication), Chapter 6 (synchronization), Chapter 7 (consistency and replication), and Chapter 8 (fault tolerance) of the course text book (Andrew S. Tanenbaum and Maarten van Steen, *Distributed Systems: Principles and Paradigms*, third edition, Prentice Hall, 2017).

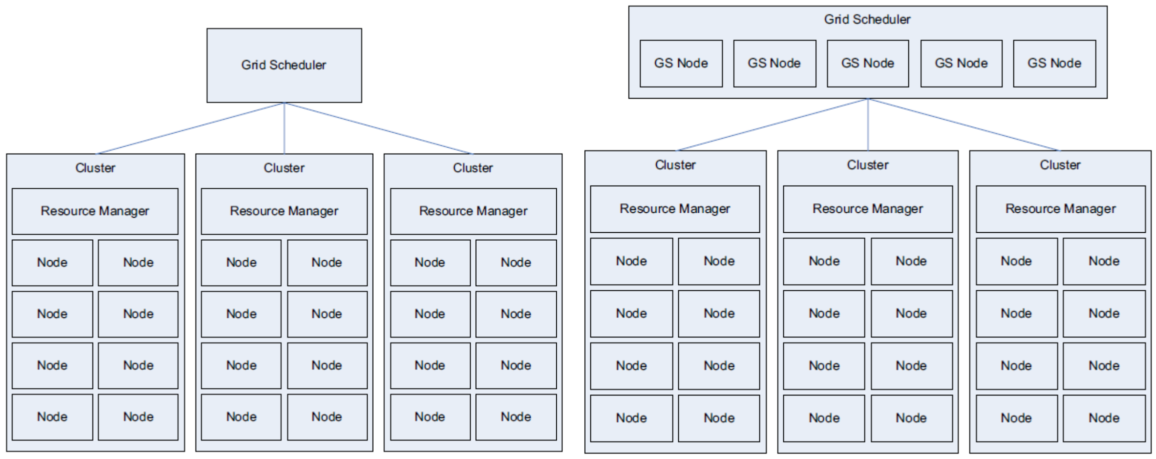

Figure 1. The structure of the VGS with (left) a single-node grid scheduler, and (right) a multi-node, distributed grid scheduler.

B Assignment

The vision of a seamlessly shared infrastructure comprising heterogeneous resources to be used by multiple organizations and independent users alike, which was formulated in the beginning of the 1990s in the context of grids [4], has also been adopted by many of today's commercial Infrastructure-as-a-Service clouds. Today, grid and cloud system designers often rely on multi-cluster distributed systems as infrastructure. Although many grids and clouds exist, key features of multi-cluster systems, such as sophisticated resource planning strategies and the adaptation of existing applications to the distributed conditions of (wide-area) multi-cluster systems, are still active research topics. Many of these features require new or more in-depth understanding of the behavior of multi-cluster systems, but, because few such systems are available for exclusive access for research experiments, simulation remains an important research tool. Thus, companies and research labs may invest in in-house or market-provided products for multi-cluster system simulation.

Your team ("you") is hired by WantDS BV to develop their new multi-cluster simulator. Because the computational complexity of these simulators is high, and because the CTO of WantDS BV believes distributed systems offer an excellent performance-cost trade-off when running many simulations for many users, WantDS BV has decided to invest in the design and implementation of a distributed simulator of multi-cluster systems, the virtual grid system simulator (the VGS). **You must design, develop, experiment with realistic scenarios, and report back on the feasibility of a distributed VGS.**

Many grids are created from sets of independent clusters, which each contain resources of two types: processors and a Resource Manager (RM). Such an RM receives requests (supposed to be sequential jobs that use a processor exclusively in this exercise) from its users, dispatches them on the cluster when a processor becomes idle, and maintains a job queue of waiting jobs. For grid operation, such a set of clusters needs a grid scheduler (GS), which enables load sharing among the clusters, i.e., the transfer of jobs from heavily loaded clusters to lightly loaded ones. A non-distributed version of a VGS (see Figure 1) has been implemented by a previous consultant of WantDS BV and is available for study.

(Mandatory requirements) You must design and implement a distributed (i.e., with multiple server nodes), replicated, and fault-tolerant version of the VGS system, and assess its properties. The VGS system should meet the following requirements:

1. *System operation requirements:* When a request arrives in a cluster, it is put in the local RM's job queue or sent to the GS. The GS also maintains a job queue, from which it sends jobs to the RMs. When an RM receives a job from the GS, it has to accept it, although possibly by adding it to a waiting queue. Each cluster has an unique identifier and a fixed number of nodes (its size, e.g., 32, 64, 128). Each job has an unique identifier and a fixed duration (its runtime on a single node), and records the identifier of the cluster where it originally arrives,. The GS may also load-balance across the simulated multi-cluster system, in which case each job will record all additional clusters to which it is sent to run.

2. *Fault tolerance:* Consider a simple failure model, in which the grid scheduler nodes may crash and restart. The VGS must by design be resilient against resource manager and grid scheduler node crashes. In addition, all grid and system events (e.g., job arrivals, job starts and completions, RM and GS node restarts) must be logged in the order they occur, on at least two grid scheduler nodes.

3. *Scalability requirements:* The properties of the VGS must be demonstrated when it contains 20 clusters with at least 1,000 nodes, 5 grid scheduler nodes, and when the workload consists of at least 10,000 jobs in total. To enable the study of realistic imbalance situations, the ratio of the numbers of jobs arriving at the most and least loaded cluster should be, for at least a significant part of the experiments, at least 5.

To ascertain the correct completion of the exercise you have to write a report of 4-6 pages, to be filled as described in the following, and to be assessed in three stages.

(Optional, for bonus points, see Section F) Additional features that you may want to consider for your system (pick at most two):

1. *Realistic experiments*: by using workloads or parts of workloads taken from the Grid Workloads Archive [5].
2. *Multiple consistency models*: analysis of the impact of the consistency model on performance.
3. *Advanced fault-tolerance:* through tolerance for and analysis of the impact of multiple failures or other failure models than fail-stop on the VGS system.
4. *Multi-tenancy*: support multiple users running simulations for the VGS system, through a queue of simulation jobs and adequate scheduling policies.
5. *Repeatability*: making sure that the outcome of the simulations is always the same for the same input, through the use of priority numbers for concurrent simulated events [1, Section 3.9.2].
6. *Benchmarking*: by running service workloads designed to stress particular elements of the system.
7. *Portfolio scheduling*: by adapting the concept of portfolio scheduling [6] to the VGS system.

Good guidelines for this work can be found in previous publications of the At-large group [2,3,6].

C Deadlines

1. **(mandatory) November 6, 2017 (week 45): Enroll in Canvas. Students who have not enrolled will not be graded and cannot receive credits for this course.**
2. (recommended) Week 45: Discuss with TA your group's system requirements and proposed approach for fault-tolerance and scalability, for the lab exercise.
3. (recommended) Week 46: Discuss with TA your group's system requirements and proposed approach for replication and consistency, for the lab exercise.
4. **(mandatory) Anytime before December 6, 2017 (week 49): Demonstrate to TA the system implemented for the lab exercise.**
5. **(mandatory) December 15, 2017 (week 50): Turn in to TA the lab-exercise report, get results (and/or recommendations for improvement).**
6. (recommended) December 20, 2017 (week 51): Demo Day for the selected groups.

To achieve the above-mentioned deadlines, your assignment can be divided into three parts:

Weeks 2—3:
- Analyze the system functionality and summarize its requirements.
- Analyze the potential system implementation and summarize your design's key features. Discuss with your team-mate the alternatives for the design choices you have made, and summarize them in your report. Try to have the resulting requirements and design document (maximum 2 pages) approved by the assistants before you start the actual implementation.
- Implement *replication* and a suitable *consistency* mechanism.

Weeks 4—5:
- Implement *fault-tolerance*.
- Test your system's implementation for *scalability* (see the scalability requirements, point 3 in the mandatory requirements).
- Consider appropriate failure rates and recovery durations. Make **a test plan for scenarios with failures (maximum 0.5 pages)**, and try to have it approved by the assistants.

Weeks 6—7:
- Implement and execute the test plan with failures.
- Demonstrate your system to the assistants.
- Finish your report by including the achieved results and your conclusions, and have it approved by the assistants.

Note: the estimated time required for the completion of WantDS's assignment (the large exercise) is 112 hours, equally divided by a team of six.

Notes:

- Try not to exceed, for any experiment, 10 hours of work.
- You can leverage the same development and setup for several experiments. In this case, report the time spent for the shared part only for the first experiment and explain the large amount of time spent for it in the report.

D Report Structure

The report should have the following structure:

1. *Report title, authors, and support cast* (Lab assistant and course instructors). For each person in your team, give name and contact information (email).
2. *Abstract*: a description of the problem, system description, analysis overview, and one main result. Size: one paragraph with at most 150 words.
3. *Introduction (recommended size, including points 1 and 2: 1 page)*: describe the problem, the existing systems and/or tools about which you know (related work), the system you are about to implement, and the structure of the remainder of the article. Use one short paragraph for each.
4. *Background on Application (recommended size: 0.5 pages)*: describe the VGS application (1 paragraph) and its requirements (1 paragraph per each of consistency, scalability, fault-tolerance, and performance).
5. *System Design (recommended size: 1.5 pages)*
    a. *System overview*: describe the design of your system, including the system operation, fault tolerance, and scalability components (which correspond to the homonym features required by the WantDS CTO).
    b. **(Optional, for bonus points, see Section F)** *Additional System Features*: describe each additional feature of your system, one sub-section per feature.
6. *Experimental Results (recommended size: 1.5 pages)*
    a. *Experimental setup*: describe the working environments (DAS, Amazon EC2, etc.), the general workload and monitoring tools and libraries, other tools and libraries you have used to implement and deploy your system, other tools and libraries used to conduct your experiments.
    b. *Experiments*: describe the experiments you have conducted to analyze each system feature, such as consistency, scalability, fault-tolerance, and performance. Analyze the results obtained for each system feature. Use one sub-section per experiment (or feature). In the analysis, also report:
        i. *Service metrics of the experiment*, such as runtime and response time of the service, etc.
        ii. *(optional) Usage metrics and costs of the experiment*.
7. *Discussion (recommended size: 1 page)*: summarize the main findings of your work and discuss the tradeoffs inherent in the design of the VGS system. Should WantDS use a distributed system to implement the VGS system? Try to extrapolate from the results reported in Section 6.b for system workloads that are orders of magnitude higher than what you have tried in real-world experiments.
8. *Conclusion*
9. *Appendix A: Time sheets (see Section E)*

E Document the Time You Spend

Because you conduct a consultancy job for WantDS, you should report on the time it takes to conduct each major part of the assignment. Specifically, report:

- the `total-time` = total amount of time spent in completing the assignment (the large exercise).
- the `think-time` = total amount of time spent in thinking about how to solve the assignment (the large exercise).
- the `dev-time` = total amount of time spent in developing the code needed to solve the assignment (the large exercise).
- the `xp-time` = total amount of time spent in experiments for the assignment (the large exercise).
- the `analysis-time` = total amount of time spent in analyzing the results of the experiments for the assignment (the large exercise).
- the `write-time` = total amount of time spent in writing this report
- the `wasted-time` = total amount of time spent in activities related to the assignment (the large exercise), but which cannot be charged as think-time, dev-time, xp-time, analysis-time, or write-time.

F Scoring

1. **Quality of the report [8,9]** (presentation, style of writing, graphing, requirements analysis, design, analysis of results): **+2,000 points**.
2. **Technical quality of the systems work** (basic system features are supported)**: +2,000 points**.

3. *Bonuses*
    a. Making your source code open-source and your report public: +100 points.
    b. Additional system features (feature and report): +250 points/feature.
    c. Excellent report (writing, graphing, description of system): at most +500 points.
    d. Excellent analysis (design of experiments, analysis of results): at most +500 points.

4. *Limits on bonuses*
    a. The total additional points for this exercise can amount to at most +2,000 points, representing additional features, excellent report and/or analysis bonuses, etc.
    b. The additional features (see page 3) can amount to at most +1000 points. In other words, you can receive bonuses for at most 4 well-supported additional features.

Note: writing a good technical report is scored *equally* to writing code for a system that works! This means that you should schedule time to write a good report, and for one revision to respond to our feedback.

G Use of Existing Technology

You are not limited for this exercise to any technology, but:

- **The non-distributed version of the VGS is available, as a Java program, in the file** `400130-VGS.zip`. **This version is provided as-is, for study purposes only, without even implied guarantees that it will compile on your system.**
- You CAN use the machines provided by our system DAS-4, or by Amazon EC2 or another IaaS cloud. For Amazon EC2, you can use the free resources provided through the Free Usage Tier [http://aws.amazon.com/free/]; if you use the `m1.small` (paid) instances of Amazon EC2, the estimated cost for this exercise does not exceed 8 EUR[1].
- You CAN use libraries that already provide the functionality required in Section B as long as you implement a bonus feature listed in Section B in exchange AND after approval of the TAs. If you only implement the mandatory funcitonalities, you CAN NOT use such libraries. For example,
  - **you CAN use Java RMI.**
  - **you CAN use Python's default communication libraries (sockets).**
  - **You CAN use Java Akka.**

When using an external library, give due credit in your code and report (see also Section H).

When in doubt about technology you intend to use, consult with the lab assistant.

H Anti-Fraud Policy (Zero-Tolerance)

Our anti-fraud policy is simple: zero-tolerance, within the limits set by VU Amsterdam. We will pursue each case of potential fraud, and will use to the maximum extent the means provided by VU Amsterdam to prevent, then to discover and punish (attempts to) fraud.

You can learn more about how to prevent that you commit fraud via a discussion with the "studieadviseur", from Appendix 2 in the Regulations and Guidelines regarding examinations FEWEB [7], or even from international sources such as Harvard's guidelines on avoiding plagiarism [8].

1 At 8 Euro-cents per hour, assuming a total workload of 100 hours. Mileage may vary.

References

[1] Richard Fujimoto, Parallel and distributed simulation systems, John Wiley & Sons, 2000.

[2] Alexandru Iosup, Omer Ozan Sonmez, Dick H. J. Epema: DGSim: Comparing Grid Resource Management Architectures through Trace-Based Simulation. Euro-Par 2008: 13-25. [Online] Available: http://dx.doi.org/10.1007/978-3-540-85451-7_3 or http://www.st.ewi.tudelft.nl/~iosup/dgsim08europar.pdf.

[3] Alexandru Iosup, Omer Ozan Sonmez, Shanny Anoep, Dick H. J. Epema: The performance of bags-of-tasks in large-scale distributed systems. HPDC 2008: 97-108. [Online] Available: http://doi.acm.org/10.1145/1383422.1383435 or http://www.st.ewi.tudelft.nl/~iosup/perf-bots-lsdcs08hpdc.pdf.

[4] Ian Foster, Carl Kesselman, and Steve Tuecke, ”The Anatomy of the Grid: Enabling Scalable Virtual Organizations”, International Journal of Supercomputing Applications, 15(3), 2002. Available from the Lab site on Blackboard.

[5] Alexandru Iosup, Hui Li, Mathieu Jan, Shanny Anoep, Catalin Dumitrescu, Lex Wolters, Dick H. J. Epema: The Grid Workloads Archive. Future Generation Comp. Syst. 24(7): 672-686 (2008). [Online] Available: http://dx.doi.org/10.1016/j.future.2008.02.003 and http://www.pds.ewi.tudelft.nl/~iosup/GWA_FGCS.pdf.

[6] Kefeng Deng, Junqiang Song, Kaijun Ren, Alexandru Iosup: Exploring portfolio scheduling for long-term execution of scientific workloads in IaaS clouds. SC 2013: 55 [Online] Available: http://doi.acm.org/10.1145/2503210.2503244 and http://www.pds.ewi.tudelft.nl/~iosup/portfolio-scheduling13sc.pdf .

References for Anti-Fraud Policy

[7] Vrije Universiteit anti-fraud policy, section 10.3. in the Student Charter. [Online] Available: https://www.vu.nl/en/Images/Studentcharter_2016_2017_tcm270-793620.pdf

[8] Tips to Avoid Plagiarism, Harvard web site document. [Online, accessed October 2017] Available: https://www.extension.harvard.edu/resources-policies/resources/tips-avoid-plagiarism

References for Quality of Writing

[9] Strunk and White, The Elements of Style. Longman, Fourth Edition, 1999.

[10] Simon Peyton Jones, How to write a good research paper. [Online] Available: http://research.microsoft.com/en-us/um/people/simonpj/papers/giving-a-talk/giving-a-talk.htm . Last retrieved: 10-Feb-2014.