User Guide of GloudSim v1.0

Aug. 2013

Contact: disheng222@gmail.com

Introduction

Google released a large-scale real production trace (http://code.google.com/p/googleclusterdata/) in Nov. 2011. Since then, many researchers have studied and characterized the trace carefully. Many characterization papers can be found on Internet. However, it's still uneasy to make use of the Google trace for simulation, because the Google trace involves too much information (hundreds of G bytes in disk space size) and a lot of useful information is actually hidden for confidentiality. The hidden information includes the real memory usage, the real CPU rate consumed per task, etc. Instead, through the trace, we can know their "relative" values (a.k.a., scaled or normalized values) in that each observed value is got by dividing the true value by the maximum value appearing in the system. More details can be found in Google trace guide (https://docs.google.com/file/d/0B5g07T_gRDg9NjZnSjZTZzRfbmM/edit).

Based on the Google trace, we designed and implemented a simulation system to reproduce the Google jobs, tasks, events, and resource utilization. Our objective is to support and ease the further research on cloud computing based on the Google trace. This simulation system is called Gloudsim (Google trace based Cloud Simulation System). The whole system is coded in pure Java for high portability. We tried best to make the whole simulation as close as possible to the real situation. For example, we designed a method by executing a Java program with loaded file size to simulate the "REAL" Google task execution with real memory size. We use BLCR to checkpoint and restart tasks at run time. XEN serves as the hypervisor to manage the VM instances, which run millions of Google tasks.

HINT: Using GloudSim, you don't need to download and study Google's original trace any more (unless you want to do more what GloudSim can help you). You can find the sample job object files in the simFailureTrace directory.

What can GloudSim do for you?

On a **REAL** cluster or a data center (with multiple VM instances you already deployed), GloudSim can help:

- simulating Jobs, Tasks, Task Failure Events, Task memory utilization
- queuing tasks when necessary
- calling BLCR to perform the real checkpoint/restart behaviors,
- optimizing checkpoint interval (with implemented Young's formula)

- generating a set of log files in course of the simulation, including the true workload to process, the wall-clock time of each task execution, job execution time, queuing length, number of jobs finished over time, etc.
- analyzing the log files by math tools like computing distribution.

There are a few steps to simulate a Google job in the system.

- First, we summarize and construct Google sample jobs based on Google traces. Each sample job is a real job appearing in Google trace, and it contains one or more batchtasks. Different batchtasks in one job run in parallel (i.e., Bag-of-Tasks mode). In one batchtask, there is one chain of tasks connected in series, and each task means a uninterrupted execution duration. For example, suppose a job has one batchtask, and this batchtask has 5 tasks, then, there are 4 failure events during the job execution.
- Then, one or more sample jobs will be selected from among all sample jobs. The selected sample jobs will serve as the “REAL” jobs in simulation. Sample job’s properties and behaviors (e.g., failure events) are stored in a .obj file, which can be retrieved easily (see prepare.TestLoadJobTrace.java).
- Finally, based on simTraceFailure/jobArrivalTrace.txt, the simulated “real” jobs will be submitted to the system over time. Our simulation system will schedule them one by one and run them on VM instances. During their execution, the failure events will be simulated based on trace by using BLCR toolkit.

For details, you are highly recommended to read the paper published in SC’13 (Optimization of Cloud Task Processing with Checkpoint_restart Mechanism). You can find many details about how to optimize checkpoint intervals there. The SC’13 paper is the first successful case of using ***GloudSim***.

Getting Started (Quick Start Steps)

1. Download the package.

```
svn checkout http://gloudsim.googlecode.com/svn/trunk/
```

2. Prerequisites:
   a) You need to install BLCR (Berkey Lab Checkpoint/Restart Toolkit). BLCR is used to perform checkpoint and restart for any running/failed program. If you encounter any problems, you can post the questions in BLCR discussion group, and the developers will reply you very soon with very great patience!
   b) Install and set NFS (Network File System), e.g., called /cloudNFS
   c) Install XEN3 or XEN4 on each physical machine, and make sure the

XEN commands like “xm create xxx.cfg” are normal.

d) On each physical machine, you should bootup the VM instances before hand. In my testbed, for example, there are 8 physical machines, on each deploying 7 VM instances. Each physical machine has 8 cores, so it is necessary to reserve one core for XEN, which means there are only 7 cores to use if you hope to let each VM instance correspond to one core. At the end of this step, vm1,vm2,….,vm56 will be started on the 8 machines.

e) The VM version I used in my testbed is centos 5.3. You should configure the network (i.e., /etc/sysconfig/network-scripts/[NIC-card-config-file]) among the VM instances such that: (1) they can communicate each other; (2) they can communicate with physical machines (because physical machines are NFS servers). On VM instances, mount necessary directories, e.g., the VM image directory, the shared-disk (NFS) for storing checkpoint files, the local directory for storing checkpoint files.

HINT: You need to install BLCR on VM images. In fact, for simplicity, you could use NFS devise to install BLCR, and mount it onto any VM instance, then, every VM instance has BLCR installed.

3. You can use the java programs under *fr.imag.mescal.gloudsim.prepare* to prepare the necessary files used for simulation.

   a) You need sample job files: For simplicity, I already generated the files for you. You could find them in the *simFailureTrace* directory. In this directory, jobArrivalTrace.txt records the submission dates of jobs in the Google trace. There are three further sub-directories, single, batch, mix. They just contain corresponding types of jobs. For example, single means in the job, there is only one batchtask. In batch directory, each job has multiple parallel batchtasks (like mapreduce). “mix” means the mixture of the two types of jobs.

   (BatchTask actually refers to one TASK mentioned in the paper. A batchtask contains a chain of subtasks, each of which refers to an uninterrupted execution duration.)

   b) You need to modify prop.config file for your environment. For example, how many physical machines to use.

4. Start the simulation using JobEmulator.java class, which is the entry point.
5. Some useful scripts can be found in the package.

Other Hints:

When using 8 machines on my VM instance setting, for example, the NFS mounting looks as follows: /localfs/contextNFS is DM-NFS (see SC'13 paper for details)

```
[root@vm1 ~]# df -h
Filesystem                 Size  Used Avail Use% Mounted on
/dev/sda1                  986M  880M   56M  95% /
se017e:/localfs/contextNFS/0
                           211G   65G  136G  33% /localfs/contextNFS/0
se018e:/localfs/contextNFS/1
                           211G   63G  138G  32% /localfs/contextNFS/1
se019e:/localfs/contextNFS/2
                           211G  8.4G  192G   5% /localfs/contextNFS/2
se020e:/localfs/contextNFS/3
                           211G  8.8G  192G   5% /localfs/contextNFS/3
se021e:/localfs/contextNFS/4
                           211G  9.0G  192G   5% /localfs/contextNFS/4
se022e:/localfs/contextNFS/5
                           211G  8.8G  192G   5% /localfs/contextNFS/5
se023e:/localfs/contextNFS/6
                           211G  8.5G  192G   5% /localfs/contextNFS/6
se024e:/localfs/contextNFS/7
                           211G   72G  129G  36% /localfs/contextNFS/7
se024e:/cloudNFS           211G   72G  129G  36% /cloudNFS
tmpfs                     1000M     0 1000M   0% /ramfs
```

Reference

1. Sheng Di, Yves Robert, Frédéric Vivien, Derrick Kondo, Cho-Li Wang, Franck Cappello, "**Optimization of Cloud Task Processing with Checkpoint-Restart Mechanism**", to appear in IEEE/ACM Proc. of 25th International Conference of SuperComputing (*IEEE/ACM SC2013*), Denver, CO, US, 2013.