Grid infrastructure analysis with a simple flow model.

Lev Shamardin[*†]
Scobeltsyn Institue of Nuclear Physics, Moscow State University
E-mail: shamardin@theory.sinp.msu.ru

Andrey Demichev
Scobeltsyn Institue of Nuclear Physics, Moscow State University
E-mail: demichev@theory.sinp.msu.ru

Alexander Kryukov
Scobeltsyn Institue of Nuclear Physics, Moscow State University
E-mail: kryukov@theory.sinp.msu.ru

Grigory Shpiz
Scobeltsyn Institue of Nuclear Physics, Moscow State University
E-mail: shpiz@theory.sinp.msu.ru

The main purpose of this work was to analyse the behaviour of a grid system under a high workload. A simple grid model based on job and information service request flows was created. In this model the translations of the job flows are studied instead of simulating the execution of each single grid job. The structure of the grid system used for analysis is automatically generated from the data published in the LHC Computing Grid information systems using a set of rules. The performance analysis of the LCG shows some possible locations of the bottlenecks in the system, and several ways of elimination of these bottlenecks are modeled.

XI International Workshop on Advanced Computing and Analysis Techniques in Physics Research
April 23-27 2007
Amsterdam, the Netherlands

*Speaker.
†The research was partially supported by INTAS-CERN Grant 2005-7509 and RFBR Grant 06-07-89199

1. Introduction

Grid simulation is a very useful tool for grid architecture and performance analysis. It allows to carry out the experiments that will never be possible on a real system. Simulation allows to alter the grid structure, to study its behavior under stress conditions, to find the bottlenecks in the system. It is also possible to simulate different working conditions like mass job execution failures allowing to determine the critical parameters of the system.

There are several different approaches to the grid simulation. The most popular method is direct modeling of the grid nodes and links between them. In this case the simulator emits virtual jobs and after that the whole trace of the execution is followed in the simulator from the initial job submission to the execution and job completion or failure. This approach is implemented in many existing grid simulators like simgrid [1], gridsim [2] and beosim [3]. Such approach can be quite resource hungry in general case.

Another method is to study the different informational flows in the system caused by the job flows, such as informational system request flows, job error flows, etc. In this case the model usually consists of a system of nodes taking a number of flows, possibly of different types, as input and producing a number of flows as output. This approach is quite effective because in this case the system parameters we are interested in are produced in the simulation directly.

In this research we develop a simulator of the informational flows caused by the job submissions in a grid (based on the EGEE/LCG grid [4]). The simulator is capable of generating the model of the system based on the information from the informational subsystems of the EGEE/LCG and from the information gathered by the special system analysis jobs. After the generation the model can be edited by hand which allows easy modifications for parameter analysis depending on the changes in the structure.

The purpose of this modeling is finding the defects (bottle-necks) in the system and possibilities to remove them by changing the system layout. To achieve the main goal, that is to process all input jobs completely, the system must have enough resources of any kinds, therefore there should be no defects at all. If there are any defects in the system it is not really important whether the system will handle the load after it is reduced by the defects because the initial goal is not achieved anyway. Considering this in all simulations we only mark the defective nodes and do not reduce the resulting request streams.

2. The Model

The grid structure is modeled after the EGEE/LCG grid [4]. We define the following node types: User Interfaces (UI), Resource Brokers (RB), Computing Elements (CE) and the informational system nodes (BDII). A number of requests of any kind from one node to another will be called a flow. The job execution starts with job submission request from a UI to a RB, which generates a request to an informational system and a job execution request to the CE.

2.1 Node types

User Interface is a source of the job submission requests in the system. Each UI can be connected to one or several RBs. UI is described by a flow j of the job submission requests.

Resource Broker accepts incoming job submission requests and generates a flow of informational requests to the BDII and a flow of job execution requests to the CE. The incoming job submissions requests flow is a sum of all job submission request flows J from all connected UIs. Maximum job submission requests flow which can be processed by a RB is equal to J^{max}. The flow of requests to the BDII from the RB is

$$Q = kI \tag{2.1}$$

where k is the average number of requests sent to the BDII per one job submission request. It may be possible that $0 < k < 1$ if the RB does cache the information from the BDII and sends only one request per a number of jobs. The Resource Broker also generates a job execution requests flow R to the CE which cannot exceed R^{max}. The J^{max} and R^{max} are defined essentially by the performance of the nodes with the RB software.

The informational system BDII handles the informational requests flow from the RB and it is described by a maximum flow of requests Q^{max} it can handle.

Computing Element "executes" the jobs. This node is described by the maximum job execution requests flow S^{max} it can handle. If the whole grid is overloaded in terms of its computational resources it is

$$\sum_{n\in UI} j_n > \sum_{m\in CE} S_m^{max} \tag{2.2}$$

In this case the actual overload will cycle between different CEs in the system and does not make sense to distinguish a separate overloaded CEs from others. This overload "rotation" will be caused by the fact that in case of a CE job execution rejection the RB will try to execute the job on another CE where again it will be rejected due to the system overload. Consequently we can join all CEs into one equivalent CE with

$$S_{eq}^{max} = \sum_{n\in CE} S_n^{max} \tag{2.3}$$

2.2 Node connections

Every User Interface is connected to one or more Resource Brokers. For each such link it weight W^{UI} is defined. This means that the matrix W_{nm}^{UI} is defined, in which the number of rows is equal to the number of User Interfaces in the system and the number of columns is equal to the number of Resource Brokers. Thus the job submission request flow from all User Interfaces to the Resource Broker m is equal to

$$J_m = \sum_{n\in UI} W_{nm}^{UI} j_n. \tag{2.4}$$

Every RB is connected to one or more BDII servers. Similar to the case with UI–RB links its weight W^{BDII} is defined. And therefore

$$Q_k = \sum_{m\in BDII} W_{mk}^{BDII} k_m I_m. \tag{2.5}$$

All Computing Elements in the model are combined into one equivalent Computing Element, and all Resource Brokers are connected to this Computing Element. The job submission request flow from the Resource Broker β to the equivalent Computing Element is

$$S_m = J_m. \tag{2.6}$$

3. Grid analysis tools

The global grid infrastructure of the EGEE is such big and evolves so fast changing dynamically that there is no one precise description of all connections between all components in the system. Therefore for getting the model parameters two software tools were developed.

The *tool for analysis of connections between RB and BDII* obtains the list of the Resource Brokers from the BDII database and tries to find the particular BDIIs used by the RBs. This information is never published in any public informational systems, so the tool exploits the fact that one of the RB configuration files may be accessed by any authenticated to the RB user via gridftp protocol. The tool downloads the RB configuration and tries to find the address of the used BDII server from that file.

The *BDII performance evaluation program* is used to measure the average reply times for the different kinds of typical queries [5]. It uses the queries similar to LFC location query (usually sent by a job running at the CE) and typical queries from the RBs (usually sent from a RB during the matchmaking process [6]). The program allows to send a specified number of queries of different types in several parallel processes. The queries in each process are shuffled in its own random order. Reply times are collected and average reply times are calculated.

4. Automatic grid model structure generation

The model structure is generated from the information from SAM [7] and BDII, and from the information about RB and BDII connections gathered by the corresponding tool. The grid model structure generation program should be executed on any User Interface by a user with as broad access to the grid resources as possible (e.g. an ops VO user, or dteam VO user) because the authenticated access to as much RBs as possible is required.

The links between different grid nodes are generated based on geographical subdivision. The information about node's country/region is taken from the SAM database. The grid model structure is generated using the following rules:

1. It is assumed that every registered site has only one UI.
2. If there is one or several RBs on the site then each UI is considered to be equally connected to each of them. Otherwise the UI is considered to be equally connected to each of the RBs in the country. If there are no RBs in the country regional RBs are used.
3. The links between RBs and BDIIs are generated according to the information gathered by the analysis tool. If this information is unavailable the links are generated similar to the links between UIs and RBs.
4. Information about performance of all CEs in all regions is added to the performance of the equivalent CE.
5. The properties of the nodes are reset to the default values.

An example of the generated structure is shown on figure 1. This example shows the part of RDIG [8] structure generated by the program. The numbers correspond to the RDIG nodes (see

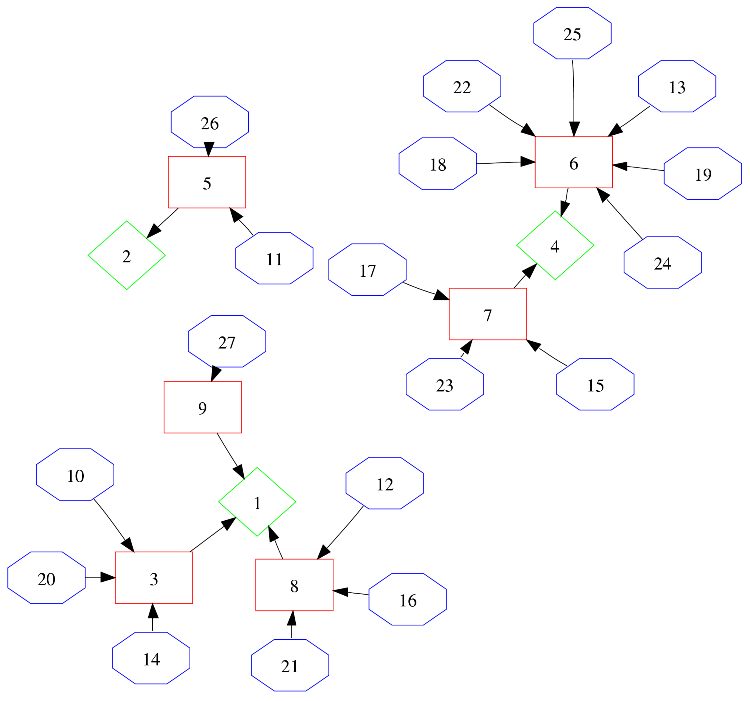

Figure 1: Automatically generated RDIG structure. Octagon is a UI, rectangles is a RB and diamond is a BDII. Only one UI–RB connection is shown for each UI.

table 1). Only one UI–RB connection is shown for each UI. The automatically generated model is adequate to the real RDIG structure.

5. Flow calculations

The order of the flow calculation and defect marking is the following:

1. Job submission request flows for each UI are read from the input files.
2. The model parameters are read from the model configuration files.
3. The flows between UIs and RBs are calculated.
4. On each RB all input job submission flows are summed up. If the resulting flow exceeds the I^{max} for a particular RB the node is marked as defected.
5. The informational system requests flows are calculated.
6. Input flows on the BDIIs are summed up. If the resulting flow exceeds the Q^{max} for a particular BDII the node is marked as defected.
7. The job execution requests flows are calculated and summed up. This sum is checked against S_{eq}^{max}.

As the output of the simulation we have a file with all defective nodes marked.

Table 1: Number – node mapping for figure 1

N	Node	N	Node	N	Node
1	lcg15.sinp.msu.ru	10	UI ru-IMPB-LCG2	19	UI RU-SPbSU
2	bdii.itep.ru	11	UI RRC-KI	20	UI ru-Moscow-FIAN-LCG2
3	lcg16.sinp.msu.ru	12	UI ru-Moscow-GCRAS-LCG2	21	UI RU-Phys-SPbSU
4	lcgbdii.jinr.ru	13	UI RU-Moscow-KIAM-PPS	22	UI JINR-LCG2
5	wmslb.itep.ru	14	UI ru-Moscow-SINP-LCG2	23	UI RU-Protvino-IHEP
6	lcgrb01.jinr.ru	15	UI ru-Chernogolovka-IPCP-LCG2	24	UI ru-Moscow-MEPHI-LCG2
7	lcgrb02.jinr.ru	16	UI ru-PNPI-LCG2	25	UI RU-Moscow-KIAM-LCG2
8	cluster.pnpi.nw.ru	17	UI ru-Novgorod-NOVSU-LCG2	26	UI ITEP
9	lcg36.sinp.msu.ru	18	UI Ru-Troitsk-INR-LCG2	27	UI SU-Protvino-IHEP

6. Results

In this work a simple grid simulator based on a flow model and EGEE/LCG modeling system were created. The model is generated automatically from the information published in the different EGEE/LCG services and from the information gathered by the special analyser jobs. The performance parameters of the different elements in the model is based on the real performance of the EGEE/LCG nodes. Initial investigations show that if the grid is loaded with the short-time jobs (with typical run time of 30 minutes) even the 50% CPU load, i.e. when only 50% of theoretical maximum number of jobs per day is handled about 25% of control nodes (RBs and BDIIs) starts to fail. BDIIs start to fail first and RBs fail second.

The authors would like to thank Andrey Kiryanov, Victor Abramovsky, Mirco Mazzukato and Jürgen Knobloch for their help with the organization of this research.

References

[1] *SimGrid*. http://gcl.ucsd.edu/simgrid.

[2] *GridSim*. http://www.gridbus.org/gridsim/.

[3] *BeoSim*. http://www.parl.clemson.edu/ wjones/research.

[4] M. Lamanna, *The LHC computing grid project at CERN*, *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* **543** (2004), no. 1-2 1–6. Proceedings of the IXth International Workshop on Advanced Computing and Analysis Techniques in Physics Research.

[5] L. Shamardin, *LCG/gLite BDII performance measurements*, . in these proceedings.

[6] *gLite: Lightweight Middleware for Grid Computing*. http://glite.web.cern.ch/glite/.

[7] G. Debreczeni, *Service Availability Monitor*, 2007. Available online, https://twiki.cern.ch/twiki/bin/view/LCG/SAMOverview.

[8] *Russian Data Intensive Grid*. http://egee-rdig.ru/.