Low-Level Grid Computing Benchmarks

Vudutala China Venkata Rao, B.Ranga Swami
{vcvrao, rangab}@cdac.in

National PARAM Super Computing Facility, Centre for Development of Advanced Computing (C-DAC), Pune University Campus, Ganesh Khind, Pune 410007, India

Abstract

We propose a set of low-level grid benchmarks to estimate the grid middleware overheads such as inter-site communication for different message sizes, data transfer, basic rates of networks, and grid information service. The low-level grid benchmarks for a replica management system, which manage the copying, and placement of files have been proposed in order to optimize the performance of the date analysis process in a typical grid application. We analyze the overheads incurred by Grid Resource Broker (GRB) for job submission, and the efficiency of scheduling algorithms for different scenarios. The benchmarks have been designed for ease of deployment and make simple estimates for overheads to understand intrinsic issues of grid middleware. The benchmarks can demonstrate the usability and robustness of the probes by testing in an automated fashion for many repetitions and investigate the extent to which our probes can provide insight into the grid stability, robustness, and performance.

1. Grid Communication Overhead Measurement Benchmarks

The objective of grid probes is to assess the quantification of overheads associated with grid middleware in which the communication overheads among the grid sites (nodes) can be estimated. The quantification of overheads is tightly coupled with grid resource broker, grid scheduler, and the grid middleware layer, which provides the authentication, job submission, and resource discovery mechanisms when a job is submitted. We assume that the grid nodes are connected through an external wide area network (WAN) or through dedicated network connections. In a dynamic grid computing environment, where the grid nodes are federated and shared by other organizations, the grid resource brokering which is on top of local resource managers play an important role to measure these overheads in our benchmarks.

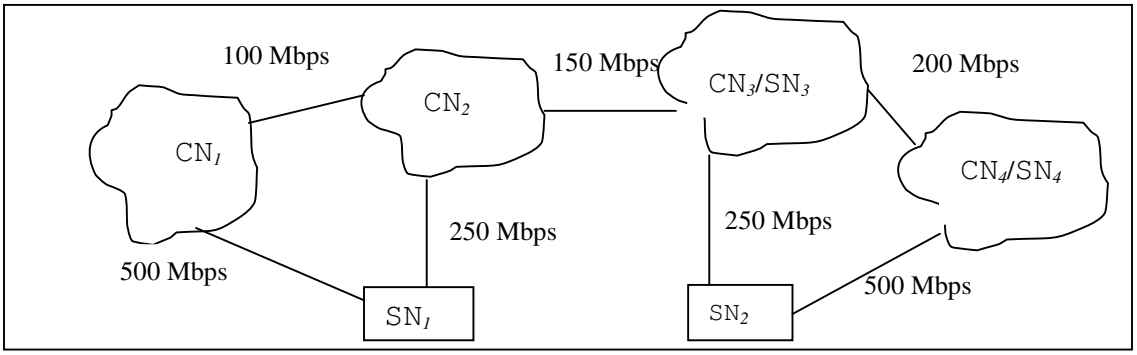

Figure 1. Sample grid configuration in Distributed Tera-scale grid facility

Figure 1 shows a simple configuration of a grid consisting of two job-submits nodes (SN_1, SN_2) and four compute grid nodes (CN_1, CN_2, CN_3, CN_4), connected through different networks with different speeds. The estimated communication time can be expressed as

$$\theta(t) = \sum_{i=1}^{n} (t_s + t_{ng} + t_w + t_o + t_{nd})$$

where t_s is startup time, t_{ng} is negotiation time, t_w is time taken to transfer w words of a job, t_{nd} represents overhead time due to occupancy and the network delay and n is the number of grid nodes (sites) involved. The negotiation time (t_{ng}) affects the overall performance, from the CPUs to the data transfer layer, from authentication system to the job scheduler and the variation of workload with respect to grid resources.

To measure the communication time for job submission of grid application, and to capture the important characteristics of compute intensity and data intensity, the concept of *grid normalized bandwidth matrix* and *grid communication time matrix* have been defined. The measured overhead time value, between the submit node SN_i and the compute node CN_j in the *grid communication time matrix* describes the measured communication time for each pair of grid nodes. The *grid normalized bandwidth matrix values* can be considered as a function of other network parameters such as number of hops, network delays at the physical layer, CPUs to the data transfer layer, from the authentication system to the job scheduler in more complex grid environments. The overheads can be attributed to the start-up cost, overheads of job submission, information query, and authentication in the grid middleware layer. These benchmarks consists of measuring communication time taken for job submission and execution which involve travel of small message sizes between grid nodes, and different types of data transfer from one node to another.

1.1. *Ping* , *Ping-Pong* and *Data Transfer* Benchmark

The *Ping* benchmark measures the communication overheads for a small message size between two grid nodes as shown in Figure 1 in which the sites $\mathbf{S}_1$, $\mathbf{S}_2$, $\mathbf{S}_3$,…..,$\mathbf{S}_N$ are involved in grid infrastructure. The aim is to accurately calculate the overheads of startup, negotiation and job submission utilities in the grid environment. Each grid node is connected to every other grid node either directly or through multiple paths, as defined in typical grid architecture. The benchmark proceeds with calculation of communication time by submitting a small job like "*sleep 0*" or "exit" to know actual time taken for the startup, negotiation of the services across various layers of the grid architecture, and submission of the job. The execution of *ping* benchmark periodically can be used as troubleshooting and tool and monitors the health of the grid system.

The *ping-pong* benchmark involving all sites gives an overview of health of the gird infrastructure and the tool and monitors the health of the grid system. The job submission refers to spawning of jobs on two node sites each of which will transfer a large message from one to another grid node in a *ping-pong* scheme. The benchmark implements a roundtrip scheme of communication between two sites. There is a high degree of inter-dependency among grid nodes for inter task communication and the resource scheduler play an important role to negotiate the services as well as to identify the cost of resources utilized.

The data transfer benchmark estimates the overhead time involved in data transfer from one storage node to another grid compute node. The benchmark consists of encryption of the data and sending data (file) from one grid node to another grid node using the data transfer utility. Decryption of the data on the second grid node is performed and the decrypted data is sent back to the first node. The data is checked on *sender* and *receiver* grid nodes for accuracy. The experiment will be repeated with varying data sizes ranging from 100 MB to 10 GB for pair of grid nodes to understand the cost of communication overheads and network traffic.

All probe benchmarks check the initial operations such as valid grid proxy, validate the configuration, grid RSL specification, basic authentication mechanism, and check directory size to ensure that target directories on remote sites can accommodate files to be transformed, different query to MDS to find the available information to all nodes involved in the probe. This experimental data gives an insight into the issues involved for assessment of the grid middleware software and performance of connectivity layer of ideal grid architecture.

2. Benchmarks for Assessing Replica Management

The purpose of replica is to provide mappings between logical names for files or collections and one or more copies of these objects on physical storage systems and to improve access speeds and reducing network loads by replicating frequently accesses datasets at locations chosen to be "near" the eventual users. The grid middleware play an important role to improve access speeds and reduce network loads by replicating frequently accesses data sets at locations chosen to be "near" the eventual users. The benchmarks suites are aimed to know difference in performance characteristics, depending upon location of replicas and algorithms used in the replica management. The goal in designing these benchmarks is to provide a set of suites to estimate overheads involved in managing the replication process.

Whenever two grid nodes access the same data, then data needs to be communicated between the two sites or replicated at the nearest sites so that each site can access required copy. Various scheduling scenarios within a data grid on data replication have been analyzed, which recommends decoupling of data replication from computation, while scheduling the jobs. Also, different jobs on computational nodes may require to access from different locations with different access control policies. The following two benchmarks analyze the overheads incurred in replica management and tests the data replication used in grid resource brokers of data grids.

2.1. FrequentAccess Benchmark

The benchmark is designed in reflecting a replica of the data chunks of the requested file is made close to the requesting node after certain number of accesses within specific period of time. This enables reduction in the access and communication time for successive accesses of the file. The benchmark frequently accesses the same file from a remote site with a certain time delay between successive accesses. The benchmark is executed for various data sizes repeatedly for 10 to 15 times for each time delay to measure the overheads accurately. Figure 2 explains the `FrequentAccess` benchmark on a grid with three sites, showing the main components in the

replica management. The benchmark can be used to estimate the overheads in a complex scenario of grid in which replica catalogs and hierarchy of replica managers exists.

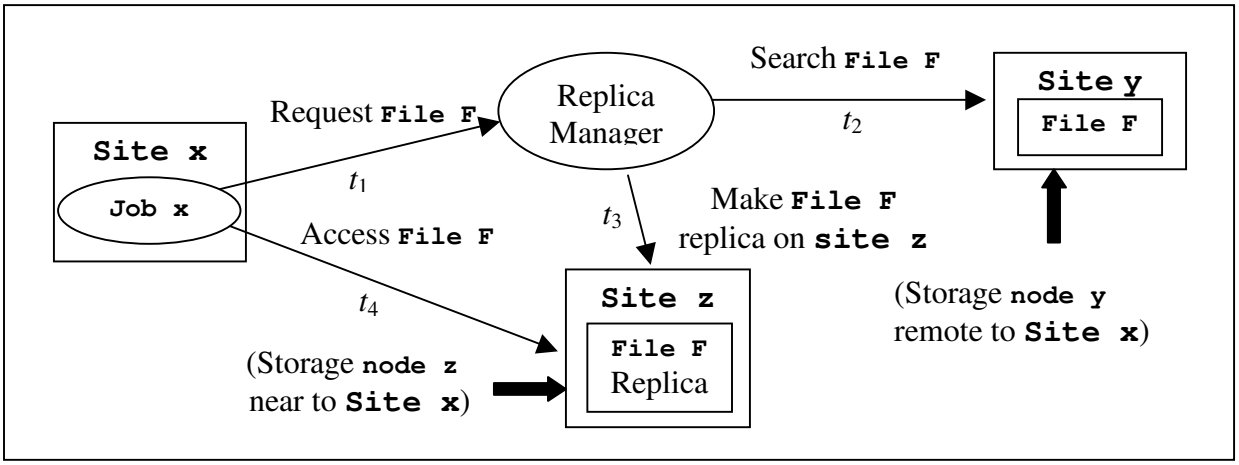

Figure 2. Typical Scenario of Frequent Access Benchmark of file on three Grid sites

The estimated time taken to access **File F** by **Job x** on first request is sum of t_1, t_2, t_3, t_4. After replication of **File F** by **Job x** on n^{th} request, the time taken on n^{th} request is t_4, which is relatively less. Our experiments indicate that **Site z** is nearer to **Site x** as compared to the proximity of **Site x** from **Site y** due to network distance and grid middleware overheads. The benchmark assesses the characteristics such as (1) time period for which a replica is maintained and kept near to a requesting node, (2) number of accesses within a specified time requires replica manager to create replicas of the replica management system, and (3) algorithm efficiency with distributed data chunks of the same file.

2.2. ParallelJobs Benchmark

The benchmark investigate the identification of overheads on the efficiency of replica management service in the context of accessing large data sets when requested by multiple jobs at the same instance. This benchmark involves spawning two jobs on two different sites requesting the same file on a remote site frequently with a certain time delay for each request. The benchmark is repeated for varying number of jobs requesting the same file from a remote site. Experiments focus on different scenarios has been conducted in which replicas are made on the computational nodes and the jobs are assigned to resources near to the storage grid nodes. Different possibility of scenarios exists and the performance of replica management services can be estimated.

The experiments test the efficiency of scheduling algorithms, which replicate the data as required, in some of data grids. Also, the efficiency of the replica management algorithms when the sites are far away from each other can be assessed. This benchmark acts as an efficient tool to assess the efficiency of the algorithms used in grid resource brokering and can be used to compare the overhead time for different scenarios of replica management systems. The benchmark is repeated for varying number of jobs requesting the same file from a remote site. The experiments test the efficiency of scheduling algorithms, which replicate the data as required, in some of in data grids. This benchmark acts as an efficient tool to assess the

efficiency of the algorithms used in GRB and can be used to compare the performance for different scenarios in grid environment.

3. Efficiency of Grid Scheduler

Grid resource brokering involves making scheduling decisions over multiple local resource managers, inorder to schedule jobs for metajobs, decision have to be made for jobs. Some jobs request resources, which can be provided, by more than one local resource manager, as well as the metajobs that are to be mapped to multiple resources some of which may be provided by different local resource managers. The benchmark aim is to detect inefficient schedule in a distributed grid environment and to bring out the overheads of the resource broker and information services when a resource failure is occurred in a grid environment. The benchmarks measure overheads incurred by resource broker for resource selection and resource allocation. We believe that these benchmarks provide insight into stability, robustness and scalability of grid. If there is a grid resource failure, then the grid information services will get resource-specific information through different ways and the grid resource brokering takes enough time to process next job, and consequently increasing the overhead time in identifying the required resources for multiple jobs. The analysis of our experiments indicate the efficient design of grid resource brokering, handling resource failures, and job migration implications and the updating of the information services can be tested, indicating scalability, usability and robustness of grid. This can be extended to test the scheduling decisions for all jobs in the system and efficiency of central *metascheduler* to produce efficient solutions.

The benchmark is focused on efficiency of grid schedulers (central metascheduler) and assesses the grid resource broker. The scheduler has all information about all the jobs and scalability with respect to either the number of resources managed or the number of jobs it can handle. The coordination between local resource managers, which accepts local jobs in addition to jobs, received from the metascheduler. The benchmark addresses the issues such as locating and allocating resources, authentication, job launching and management to avoid *data races.* The benchmark also provides the insight into the overheads of submitting a job from one site to another and the effect of various negotiations of services between GRB and local resource manager for completion of the *meta* job.

4. Implementation Framework

We use cluster of workstations as a grid node at each site and internal connections provide very high bandwidth in the range of several hundred MB per second with low latencies, the external connection hosts of much lower quality. The `Globus toolkit 2.4/4.0` with `Nimrod-G`, and `Condor-G`, resource brokers will be used for implementation of grid probe benchmarks. The `Globus Toolkit` components GRAM (Globus Resource Allocation Manager), `MDS` (Meta computing Directory Service), `GSI` (Global Security Infrastructure), GASS (Global Access to Secondary Storage) and `GrADS` (Grid Application Development Software) have been used. The software tools used are `C` and `python` language, `MPICH-G2`, `perl, scripts` languages such as `XML` based on web services. We have been implementing as `Perl` scripts, and use configuration file to contain all information. Simple C programs perform computations and generate data files.

The ping benchmark can be implemented with job submission utility such as globusrun or globus-job-submit. Experiments with various resource brokers like Nimrod/G, Condor-G can be included to measure the GRB overheads. The pingpong benchmark and generalized pingpong benchmarks involving message transfer can be implemented at the MPI layer (such as MPICH-G2 over Globus) or with the APIs provided by the grid environment. We would experiment using C language API and Java CoG provided by Globus to perform message transfer between two *meta* jobs and analyze the performance. The message transfer using XML will be done in our experiments. The data transfer benchmark can be implemented using the toolkits or the API provided by the grid environment. The benchmarks based on replica management and efficiency of grid scheduler can be implemented in various ways as per the requirement and we use the Globus API for replica management, and grid resource failure. The benchmarks can be implemented as web services based on OGSA (Open Grid Services Architecture) and OGSI (Open Grid Services Infrastructure) using Globus 4.0. To execute file transfers we use GridFTP via the Globus program and globus-job-run to launch executables. We implement the benchmark using Data Access APIs provided for access and functioning of MDS, GASS components.

5. Acknowledgements

The Centre for Development of Advanced Computing (C-DAC) supported this work under mission grants. We would like to thank many researchers across globe for their valuable contribution in this work. Their clarifications of grid benchmarks and excellent contribution helped us translate into practice to design and develop low-level grid benchmarks at C-DAC.