DESIGN AND IMPLEMENTATION OF A SERVICE FOR PERFORMING HPC COMPUTATIONS IN CLOUD ENVIRONMENT

R.I. Kuchumov [a], V.V. Korkhov [b]

Saint Petersburg State University, 7/9 Universitetskaya nab., St. Petersburg, 199034, Russia

E-mail: [a] kuchumovri@gmail.com, [b] v.korkhov@spbu.ru

Cloud computing became a routine tool for scientists in many domains. In order to speed up achievement of scientific results a cloud service for execution of distributed applications was developed. It obviates users from creating and configuring virtual cluster environment manually or using batch scheduler and allows them only to specify input parameters to perform their computations. One of the key parameters that this service aims to help users with is virtual cluster configuration. For most applications it is difficult to tell the optimal number of cluster nodes, amounts of their threads per node and memory so that application would have minimal execution time. In this work an approach to optimization of cluster configuration is proposed and software system for launching HPC application in a cloud is presented.

Keywords: Cloud Computing, High Performance Computing, Software as a Service.

1. Introduction

In this paper we are focusing on high performance computing (HPC) applications for scientific computations where the most common workflow includes repetitive execution of the same application but with different input parameters and data. When cloud infrastructure is being used as a computation backend, many usability obstacles are introduced that often leads to delays in the achievement of scientific results and unnecessary complications.

To mitigate these problems a cloud service (IdleUtilizer) for helping users to perform HPC computations in a cloud environment was created. It provides users with a web interface that allows them to specify input parameters of their application, submit it for execution and receive its output when it's finished. The details of cluster configuration, its deployment and bootstrapping process are all hidden from the user. Because of service's flexible architecture, users can execute their jobs on different computational backends using the same interface.

2. Idle-Utilizer service design and implementation

Figure 1 shows a high level overview of the Idle-Utilizer service architecture. It consists of three main components. At first, there is a web interface (covered in [1]) that is used by end-users for configuring and submitting jobs and monitoring their statuses. Web interface communicates only with the seconds component, idle-utilizer service. The main responsibilities of this service are to configure computational backend environment, to prepare, execute and monitor user's job, and to send the results back to the user. The third component is the computational backed. In case the user prefers to do computation in a cloud, the cloud provider is accessed for deploying a virtual cluster specifically for executing his job, and in case of the batch scheduler backend, the job would be created and submitted to the scheduler.

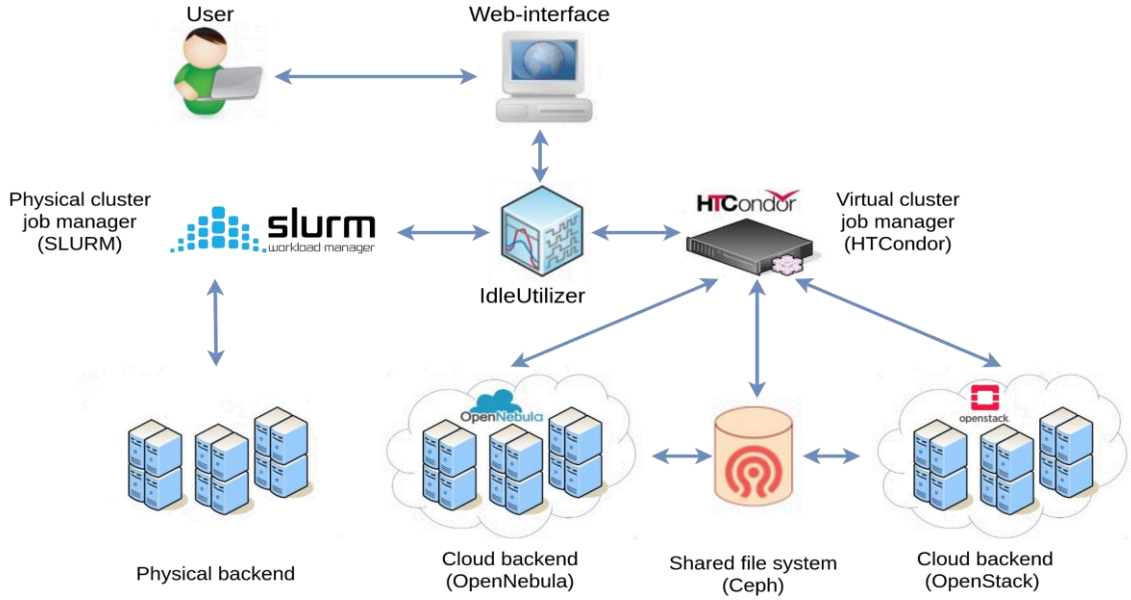

Figure 1. Service high level architecture

3. Experiments with Idle-Utilizer service

In our experiment we have used NAS Parallel Benchmarks (NPB) which consist of several widely used programs that mimic scientific computations and that are designed to evaluate the performance of HPC systems. For a testbed we have used OpenNebula cloud with KVM hypervisor provided by JINR [2].

The goal of our experiments was to justify the need for automatic virtual cluster configuration depending on job input parameters. To achieve that, we have launched benchmarks with throttled network bandwidth to simulate different hardware configurations. Figure 2 shows that in some cases there may be peaks of performance at the certain number of nodes, and when this number increases further, job performance degrades drastically. Moreover, some applications may acquire more resources than they use, or an increase of available resources does not yield proportional increase of performance.

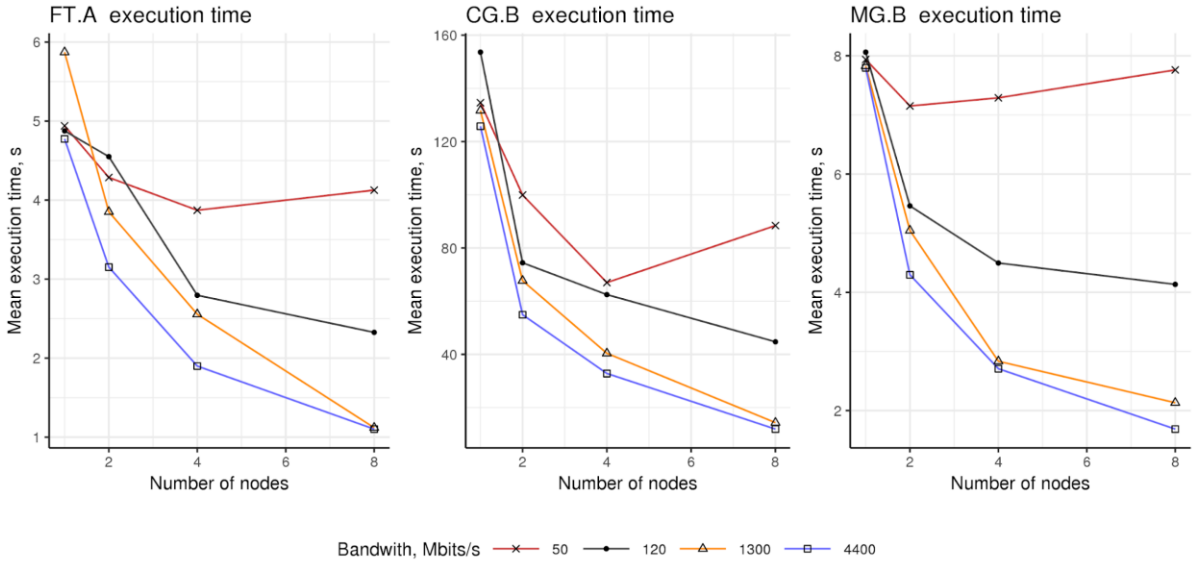

Figure 2. Benchmarks execution time with throttled network bandwidth

4. Cluster configuration optimization problem

One of the parameters that users need to specify to launch their applications is cluster configuration, e.g. number of nodes, threads and memory. Since users would not do the same experiments we did above, it makes sense to automate this process.

Let's denote x as a vector of input parameters of a user task, y as a discrete vector of cluster configuration and $t = t(x,y)$ as task execution time. Formally, this problem can be described as an optimization problem, i.e. the goal is to find optimal cluster configuration that yields minimal execution time: $y^* = y^*(x) : t(x, y^*) \rightarrow min$.

To solve that, we propose to construct the regression model of application execution time as a function of cluster configuration. This model is fitted to the previously executed tasks which are closest by some measure of input parameters (x) to the input parameters of the current task. To construct such a model we used random Fourier series approximation. This model is refined each time new task is executed and thus more data can be used to fit the model. The argument of a minimal value of this model is used as optimal cluster configuration. Since this is optimization search process, to prevent from opting to the local minimum, we also add Gaussian noise at each iteration with its variance decreasing depending as the number of samples increase.

5. Evaluating cluster configuration optimization algorithm

To evaluate our algorithm for automatic cluster configuration, we have created two parallel and distributed applications with synthetic workloads. The first one represents centralized dynamic scheduling strategy where the master process distributes tasks on the requests of slave processes. The second one represents static scheduling strategy, when tasks are distributed evenly among all the processes. Tasks of both applications consist of memory allocations with following random accesses. Both tasks have two input parameters such as batch size and the number of random memory accesses in a single task.

We applied the following approach to our proposed method. For each test application we have calculated its execution time from all possible inputs. Then we've fed random input job parameters, the method would propose cluster configuration and we gave job execution time as a feedback. Results are presented in Figure 3. On the left side there are real values of the application workload, in the middle there are approximations of these data and on the right there are deviations from minimum and maximum value of application execution time. As one can notice, on these data our method finds almost optimal cluster configuration starting from 8-10 iteration.

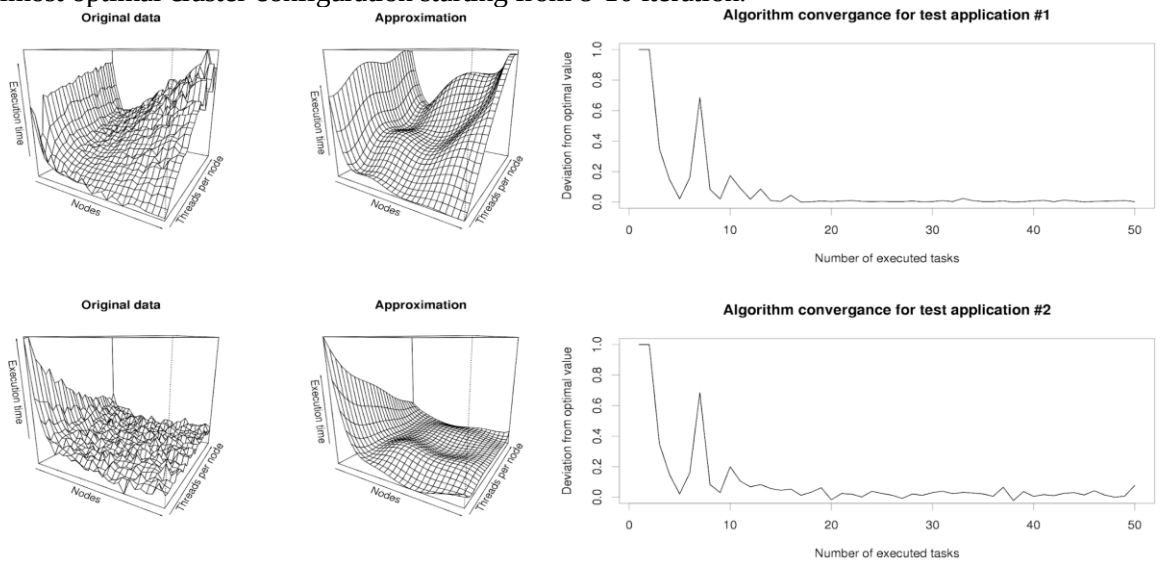

Figure 3. Experiment results with two synthetic workload applications

6. Conclusion

We have created a software-as-a-service system for launching HPC applications. It presents users an interface for submitting computational jobs and makes the process of cluster deployment and job submission to different computation backends transparent. The primary goal of this service is to simplify the workflow of HPC application execution for scientists. IdleUtilizer service has already been deployed in JINR cloud [2] and used with long Josephson junction modelling application [3].

In the current version of the system, users still have to make decisions on a cluster configuration, i.e. he or she needs to know job resource requirements to specify the number of nodes, the amount of memory and CPUs per node. To automate this process we have proposed an optimization method that searches a cluster configuration which yields a minimum execution time for a given job input parameters. The proof of concept version of this algorithm has been tested on synthetic workloads and results show that our implementation finds near-optimal value after 8-10 iterations.

Acknowledgement

Research has been supported by the RFBR grant 16-07-01111.

References

[1] Balashov N.A. et al, JINR CLOUD SERVICE FOR SCIENTIFIC AND ENGINEERING COMPUTATIONS. International scientific journal “Modern Information Technologies and IT-Education”, v. 14, n. 1, p. 61-72, mar. 2018. ISSN 2411-1473.

[2] Baranov A.V. et al. JINR cloud infrastructure evolution. Physics of Particles and Nuclei Letters 13.5 (2016): 672-675.

[3] Bashashin M.V. et al. Numerical approach and parallel implementation for computer simulation of stacked long Josephson Junctions // Computer Research and Modeling, 2016, vol. 8, no. 4, pp. 593-604.