Developing Node.Js - MPI Bridge for Cluster Computational Environment

Mikayel Gyurjyan
Institute for Informatics and Automation Problems of NAS RA
e-mail:
mikayelg@gmail.com

Zaven Naghashyan
Institute for Informatics and Automation Problems of NAS RA
e-mail:
znaghash@gmail.com

ABSTRACT

Cluster job management system was designed for organizing parallel access to the cluster by different users. The management and efficient exploitation of clusters among users, applications and data continue to be a time-consuming and challenging task. In many Linux-based clusters the jobs should be designed as programs that utilize MPI (message passing interface). MPI became de facto industry standard for the parallel computing environments. The most common programming languages that use MPI are C and C++.

One of the main limitations of the system is the fact, that programs should be implemented using C++ programming language. The number of engineers with deep knowledge of C++ during the last decade stopped to grow in a way it was before.

Modern programming trends made JavaScript as one of the most popular languages for majority of programmers. Initially JavaScript was designed to be used within web browsers. Node.Js is an environment that allows the programs written in JavaScript to run in the server environment. It supports special APIs that allow JavaScript programs to communicate with the operating system and external resources. Those APIs are mostly designed as asynchronous functions.

MPI-Node.js Bridge is designed and implemented to allow Node.Js programs to be executed within Arm-cluster computational environment and utilize MPI functionality that allows the simplicity and efficient use of the available computing resources, as well as, making the system usable for vast majority of new users that have JavaScript programming skills.

Keywords

Cluster, job management system, MPI, Node.Js.

1. INTRODUCTION

Cluster jobs management system was designed and implemented for organizing parallel access to the cluster by different users. It is used for executing special tasks (hereafter called Jobs) provided by the users. The main 2 goals of the system were to simplify access of users to the cluster and optimal usage of the cluster resources by the user jobs that can run at the same time. These two fields are controlled by two different parts of the system: Web-based User Interface and the Job Scheduler. A special Administrative Panel was designed and developed to configure and control the system main functions.

The system provides a comfortable interface to users, for reservations and scheduling shared computational resources. It also keeps history of job executions per user.

Using the web-based interface users can see detailed information on a particular Job. The system also allows scheduling jobs by performing the following actions:

1. Select the required number of cores. There are limitations on number of cores that can be reserved depending on the role the users have in the system.
2. Indicate the start-end dates to run the Job
3. Select a free period of time to run the Job according to the availability of the cores.

The scheduler module of the system controls the jobs according to preset schedule. This allows balancing the load and minimizing the impact of jobs on each other. It also gathers statistics about how jobs are running. It is designed to be as portable and diverse as possible, and, therefore, uses many common applications that are available on a variety of computing platforms. It is easily configured with the well-known PBS (Portable Batch System) [1-2], Condor, Torque queuing systems [3].

One of the main limitations of the system is the fact, that programs should be implemented using C++ programming language. The number of engineers with deep knowledge of C++ during the last decade stopped to grow in a way it was before. Nowadays vast majority of researchers and software developers use modern technologies for the programming needs. JavaScript is one of the most used programming languages in the world [4, 5, 6].

Possibility to have JavaScript-based programs run within the cluster environment could dramatically increase the number of potential users and tasks.

In the current article we consider the possibility to use JavaScript and Node.js server side environment to implement programs that can be used in the job management system.

2. ADMINISTRATIVE WEB-BASED INTERFACE

The Web-based interface supports a hierarchical structure of users that belong to groups. Every group is associated with a project. These associations are important for regulating cluster accounting. Projects can have more than one user performing work towards the project, and have their hours be counted under the same project. Administrator assigns a cost unit for each project, which allows sharing the given cost unit between the users of the project or group.

There are various cost units calculation variants depending on the project purposes and the roles of users in projects. For

example, a university student will have a lower cost unit than a commercial project.

A few administration interfaces have been developed:

- User Administration View. The administrator of the system can add or change the parameters of the system users. Strict checking is done at job-submission time, and if the user does not have an entry made in the system the job will not be permitted to enter the queue. The passwords do not have to match, and it is actually preferable to have different passwords. The system user account is used only to protect access to the user's statistics on the web statistical engine. It has no correlation to the authentication mechanism on the cluster itself.
- Cost Administration View. The administrator can edit or create different types of cost structures for computational projects. It consists of the following components: Description (A plain-text identifier for the administrator's purpose), Normal Cost Cluster Unit (the price per cost unit used for a normal, prepaid or preallocated cost unit balance), Overrun Cost Cluster Unit (the price per cost unit when a particular project's job runs over the prepaid balance, or is already in a negative balance).
- Scheduling Administration View. It supports basic scheduling capability. It allows administrators to customize the existing policies and define new scheduling policies for the cluster. It can be used to extend the existing scheduling policies or implement custom scheduling policies.
- Project Administration View. Here you can edit or create your projects. This is the project name that users should enter on the command line, when submitting the job. The project owner is a user that can alter some aspects of the project profile as well as see detailed statistical information regarding the overall project. Since the project owner only has to be a user with a username and password, they do not have to be a user on the computational system. This feature is useful for facilities that project management team members have. The administrator must assign a particular cost group to the particular project. They can enable or disable the project's ability to run in overrun mode. The administrator or project owner can provide access to a particular project.

The package consists of complex tools and scripts that can be grouped into two major components: the statistical monitoring and web interface package, and the plug-ins for PBS and Condor queuing systems. The C++ wrappers use instead of common queuing commands. The original binaries are required to perform the actual job submission to the query system. The PHP and C++ are used as the main programming languages for developing the interface.

3. MPI-BASED MODEL FOR PARALLEL COMPUTING

MPI interface is designed to provide an abstraction layer for organizing synchronization and communication functionality between processes running on different computational nodes. As a typical way of configuration each CPU (or core in a multi-core machine) is assigned to a single process. This assignment happens at runtime through the agent that starts the MPI program, normally called mpirun or mpiexec. [7]

The interfaces are designed to be thread safe. It is relatively easy to write a multithreaded point-to-point MPI code, and some implementations support such code.

MPI functions include point-to-point send/receive operations. It is possible to choose between a Cartesian or graph-like logical process topology.

Send/receive operations between processes are designed for exchanging data pairs, including intermediate results of computations (gather and reduce operations), nodes synchronization (barrier operation) and obtaining network-related information (e.g., the number of processes in the computing session, current processor identity, neighboring processes accessible in a logical topology, and so on). There are four types of point-to-point operations:

- synchronous,
- asynchronous,
- buffered,
- ready forms.

4. NODE.JS AS A COMPUTATIONAL ENVIRONMENT

Node.js is a technology and software environment that allows users to create a web servers and networking tools using JavaScript as a programming language. It uses a group of “modules”, which handle different core functions to handle server related functionality. The modules are used to serve file system I/O, cryptography functions, networking, such as DNS, HTTP, TCP, TLS/SSL, UDP, data streams, binary data, etc. To reduce complication of writing server applications, the modules of node.js use API design.

One of the following servers are supported by the node.js applications:

- Linux
- macOS
- Unix
- Microsoft Windows.

The alternative writing language for JavaScript is either CoffeeScript, Dart or Microsoft TypeScript. Actually, JavaScript may be replaced with any language, which can compile it.

Node.js is mainly used for building web server network programs. The main aspect, which differs PHP from node.js, is that functions of PHP block before completion (commands may not be executed until previous commands are completed), while the functions of node.js are non-blocking (commands may be executed in parallel. To show any failure or completion they use callbacks).

5. NODE JS – MPI INTERFACE

5.1. Node.js C++ Add-ons

Node.js Add-ons are shared objects, which are linked dynamically. C or C++ languages are used during writing Node.js Add-ons. To load it into node.js the required function should be used in the same way as standard Node.js module. The main usage is providing an interface between C/C++ libraries and JavaScript, which runs in Node.js[10].

The method for implementing Add-ons involves knowledge of several components and APIs:

- V8. A JavaScript implementation by C++ library. V8 has mechanisms, which allows creating objects, calling functions, etc.
- Libuy: The event loop(with its worker threads and the entire asynchronous platform behaviors) of Node.js is implemented by the C library. It provides a cross-platform abstraction library, which allows easy, POSIX-like access across all target operating systems to various ordinary system tasks (timers, file system interaction, sockets, and system events). Libuy serves as a pthreads-like threading abstraction. The usage is to power more sophisticated asynchronous Add-ons, which should be moved beyond the common event loop.
- Internal Node.js Libraries. Node.js exports a C/C++ APIs, which may be used in Add-ons.
- Collection of other statically linked libraries that are located within the deps/directory of the Node.js.

5.2. Open MPI

The Open MPI is an open source implementation of Message Passing Interface standard. It is developed and maintained by a group of academic, research, and industry partners [9]. The following features are offered by Open MPI software package:

- Full MPI-3.1 standards conformance
- Thread safety and concurrency
- Dynamic process spawning
- Network and process fault tolerance
- Support network heterogeneity
- Single library supports all networks
- Run-time instrumentation
- Many job schedulers supported
- Many OS's supported (32 and 64 bit)
- Production quality software
- High performance on all platforms
- Portable and maintainable
- Tunable by installers and end-users
- Component-based design, documented APIs
- Active, responsive mailing list
- Open source license based on the BSD license

Taking into account the above-mentioned advantages we have chosen the Open MPI as MPI implementation for the Cluster job management system.

5.3. Open MPI integration with Node.js as Add-on

To enable MPI features to Node.Js based jobs we implemented a Node.Js add-on that wraps the Open MPI APIs. Using JavaScript functions it is possible to delegate the functional calls to Open MPI.

The Node.JS event-scheduling mechanism supports call backs from Open MPI to JavaScript.

As the next step of the project we are consider evaluation of the system performance and its scalability over the number of processors. A benchmarking script will designed to compare Node.Js performance with C/C++ on the same tasks. The benchmarks would be focused on two metrics:

- **Runner Total Execution Time (RTET):** which is actual time taken by the job, form the moment the runner is executed with the job to the moment it exits.
- **Max Process CPU Time (MPCT):** which is the maximum of the CPU time of all the participating processes in the job.

As possible tasks we consider PI number approximation and Counting Prime numbers.

6. CONCLUSION

The main direction of the research was the investigation of modern technologies that are used in concurrent computing, organization of the high performance computational system, distributed databases and web crawlers. The goal was to ensure the fairness, simplicity and efficient cluster management system of IIAP compatible clusters with the modern technologies used in modern projects. The web interface provides different allocation methods, credit-based economy and control over delinquent members. Theoretical research was conducted on the following topics:

- Distributed computing in high performance computing systems using MPI model
- Possibility to use modern programming languages and environments in MPI-based distributed systems (JaveScript, Node js).

A software package was developed to enable MPI features to Node.Js-based jobs. The package represents a Node.js add-on that wraps the Open MPI APIs. Using the package it is possible to create jobs that are suitable to run within the Cluster job management system environment using the popular Node.js JavaScript framework.

Evaluation of the system performance and its scalability is considered as the next steps of the project.

REFERENCES

[1]. R. Henderson, D. Tweten, Portable Batch System: Requirement Specification, NAS Technical Report, NASA Ames Research Center, April 1995

[2]. R. Henderson, Job scheduling under the portable batch system, Job Scheduling Strategies for Parallel Processing, pages 279- 294. Springer-Verlag, 1995. Lecture Notes in Computer Science volume 949

[3]. Torque Resource Manager.http://www.adaptivecomputing.com/products/open-source/torque/

[4]. Developer Survey Results 2016. https://insights.stackoverflow.com/survey/2016

[5]. Language Trends on GitHub · GitHub. http://githut.info/

[6]. State of the Computer Book Market, part 4: The Languages. http://radar.oreilly.com/2012/04/computer-book-market-2011-part4.html

[7]. Y. Zhang, T. K. Sarkar; John Wiley & Sons, Parallel Solution of Integral Equation-Based EM Problems in the Frequency Domain;

[8]. Node.JS official site. https://nodejs.org

[9]. Open MPI official site. https://www.open-mpi.org/

[10]. Node.Js Add-ons.http://nodejs.org/api/addons.html