Add support for running worker on non-python platforms

March 20, 2020

About Me

Name : Samartha S M

Email : samarthamahesh@gmail.com, samartha.s@students.iiit.ac.in

Github : https://github.com/samarthamahesh

Mobile : +919535246725

University :

- Name : International Institute of Information Technology, Hyderabad
- Major : Computer Science and Engineering (CSE)
- Joining Year : 2018
- Expected Graduation Year : 2022
- Degree : Bachelor of Technology (B. Tech)

Location/Timezone : Hyderabad, India UTC+5:30

Project Abstract

Buildbot is an open-source framework for software build, test and release processes. Basically, Buildbot is a job scheduling system, executes jobs when resources are available and reports the results to users through different notifiers like Email, Web Status, IRC, Status Client.

Buildbot installation will have one or more masters and a collection of workers. Master monitors for any changes in source code repositories, coordinates activities of workers and reports results to users and developers. Workers can run on a variety of operating systems.

The current version of Buildbot automated build, test, release software is implemented in python and twisted and is part of python packages and hence the platforms that support Buildbot are limited to those that support python and twisted. But Buildbot users would want to run tests on a wide variety of platforms.

Hence, this project aims at designing a new language-independent protocol and new worker implementation in C++ (language with fewer runtime dependencies). C++ is chosen due to its ubiquity and mentoring resources.

Project Goals

1. **Designing / Investigating a language-independent protocol that supports the current master-worker communication functionality**
2. **Creating a protocol specification**
3. **Adapting both worker and master to this new protocol**
4. **Implementing a new worker with less dependencies for the protocol in C++**
5. **Documenting and open-sourcing the development process**
6. **The new worker is built and tested on weird platforms (stretch goal)**

Specifications

1. **Investigating a language-independent protocol that supports the current master-worker communication functionality**
    - Investigating a language-independent protocol will help us verify complete support for communication between master and worker. Choosing any protocol, it should support all functionalities of current communication between master and worker.
2. **Creating a protocol specification**
    - Creating protocol specification to study what kind of messages go between master and worker.
    - Studying how the protocol formats and transfers data in the network.
3. **Adapting both worker and master to this new protocol**
    - Applying this protocol to Buildbot, so that master and worker communicate in this protocol.
    - This is required so that there would be no need to depend on C++ to run tests.
4. **Implementing a new worker with less dependencies for the protocol in C++**
    - As the project aims to make workers runnable on a variety of platforms, including those with no support for python libraries, new worker will be implemented in C++.
    - C++ is chosen because of its wide support over a variety of platforms including embedded platforms.
5. **Documenting and open-sourcing the development process**
    - During the entire process, each and every step (no matter how small) will be documented for developers as well as for the users.
6. **The new worker is built and tested on weird platforms (stretch goal)**
    - After the worker is built on the new protocol chosen, it will be tested on a weird platform to check its working.
    - Continuous Integration will be set up to do testing of workers in weird platforms.

Timeline

Date	Activity
April 1 - April 26	Getting comfortable with the Buildbot's source code + Getting familiar with working of master and workers + Issue solving
April 27 - May 18 Community Bonding Period	Community Bonding + Getting familiar with organization's workflow
May 19 - May 26 Coding Period Starts	Investigating new protocol which is language-independent + Documentation
May 27 - June 5	Creating protocol specification + Documentation
June 6 - June 15	Adapting master and worker to this protocol + Documentation
June 16 - June 19	**Phase 1 Evaluation**
June 20 - June 25	Continue work on adapting master and worker to this protocol + Documentation
June 26 - July 7	Start implementing worker in Cpp + Documentation
July 8 - July 12	Buffer period (If any work is pending, this period will be utilized to complete the pending work)
July 13 - July 17	**Phase 2 Evaluation**
July 18 - July 27	Refactoring Code. Testing and Removing bugs (if any)
July 28 - August 5	Test worker on weird platforms (stretch goal)
August 6 - August 10	Final documentation + Bug fixing (if any)
August 10 - August 17	**Final Submission and Evaluation**

Benefits to community and users

If this project is successfully completed, it will enable workers to run on a wide variety of platforms with no support to python libraries which will enable users to easily automate their build, test and release processes. Community will be able to review the implementation due to proper documentation.

Deliverables

- Master and worker adapted to new language-independent protocol
- Worker implemented in C++
- Detailed Documentation for the Dev Process

Future Developments

- Contribute to BuildBot Community
- Contribute to MacPorts Community

I intend to be as open, friendly and communicative as possible, and will develop a bond with the community which will most definitely extend even after GSOC ends. I would inform the community with new updates regarding my project using mailing lists and my GSoC blog. I also request the mentors to test my work on a timely basis, so that we will be on the same page.