Project details

Summary: Porting CRIT functionalities to Go
Mentor: Radostin Stoyanov

Abstract

CRIU provides a Python-based tool called CRIT in order to explore and manipulate checkpoint images. As CRIU is adopted into more and more Go projects, it is necessary to be able to provide Go bindings for perfoming read/write/encode/decode operations on the image files. This project aims to extend the go-criu library to perform all image-based operations that CRIT is currently used for, both as an import-and-use dependency as well as a standalone CLI application like `crit`.

My interest in applying to this project is due to my prior experience with Go and containerization technologies. This solution is expected to provide native bindings in Go for all CRIU image manipulations, allowing Go projects that use CRIU to perform better analysis and testing on checkpoint images without setting up any additional infrastructure to create an RPC connection with CRIU. It also attempts to provide a more extensible alternative to the current CLI implementation of CRIT.

Implementation

Approach

The basic library will consist of standalone files for each of the operations provided by CRIT. By leveraging the concept of interfaces in Go, a common worker agent will be able to call all operation functions. As more functionality is required, this interface can simply be extended and the respective functions added.

```
// crit.go
type Crit struct {
    inputFilePath string
    outputFilePath string
    inputDirPath string
    exploreType string
    pretty bool
    nopl bool
}

type CritService interface {
    Show()
    Encode()
    Decode()
    X()
    Info()
}
```

```
func New(inputFilePath, outputFilePath, inputDirPath, exploreType string, pretty, nopl bool)
    return &Crit{
        inputFilePath: inputFilePath,
        outputFilePath: outputFilePath,
        inputDirPath: inputDirPath,
        exploreType: exploreType,
        pretty: pretty,
        nopl: nopl,
    }
}

func (c *Crit) Show() {
    ...
}
```

Partial work on the operations has already been done in this PR, which can be re-used to an extent in this implementation. All interface functions will be exportable so that they can be called through imports. A `cli.go` file will provide a standalone binary that uses this worker agent to run the CRIT commands as a CLI application, built using cobra. Every command will create a `CritService` instance with the necessary struct variables and call the respective function through this service.

Timeline

- **Before May 20**: Understand the Python implementation of CRIT in detail and go through any other relevant code necessary to implement this solution.
- **May 20 - June 12 (Community Bonding Period)**: Discuss the finer aspects of the solution with my mentor and other community members. This time can also be used to attempt to investigate any other potential features that the CRIU community would want and accordingly accomodate them into the timeline.
- **June 13 - July 24**: Implement the `crit show` and `crit decode` commands, along with appropriate unit tests.
- **July 25 - July 29**: Phase 1 evaluation. Discuss progress with my mentor and any potential change of plan going forward.
- **July 25 - Sep 4**: Implement the `crit encode`, `crit x`, and `crit info` commands, along with appropriate unit tests. Add necessary changes to the Makefile, test suite, and build ecosystem of go-criu in order to completely integrate the new code into the library.
- **Sep 5 - Sep 12**: Add documentation and examples to the project README and the CRIU website.
- **Sep 12 - Sep 19**: Final evaluation. This also serves as a one week buffer in order to accomodate unexpected delays or emergencies.
- **After Sep 20**: Discuss outcome of the project with my mentor and chart out plan of action for future contribution. Engage with community members to get feedback on project implementation and discuss add-on features.

Personal information

Name: Prajwal S N
Email: prajwalnadig21@gmail.com
Mobile:

GitHub: snprajwal
LinkedIn: snprajwal
Location: Bengaluru, India
Timezone: GMT +0530

About me

I am a second year student pursuing my engineering in computer science from Dayananda Sagar College of Engineering, Bengaluru, India. My areas of interest include distributed systems, networking, and scalability. I am actively involved in open-source projects related to Kubernetes.

I'm a generalist, and have spent considerable time poring over articles, research papers, and repositories to try and understand today's technological ecosystem better. I have also interned as a back-end developer at a leading startup in India, which has made me proficient in Go and its related tooling ecosystem. My experience with writing production-grade code and operating with multiple Go projects at scale makes me a suitable candidate for this project.

Open source activity

I have been a regular contributor to the Brigade project, mostly working on the Brigade CLI. The relevant PRs are given below:

- cli: add version command
- Print server version with brig version
- cli: test connection to api server before returning to caller

Commitments during GSoC 2022

I will be dedicating 40 hours a week on average towards this project. My 4th semester classes will be happening during the program, but I am negotiating with the university to obtain permission for full-time work. If granted, I will be able to put in additional hours and implement features beyond the discussed scope. Any absence of mine shall be informed to my mentor well in advance and those hours will be compensated in the following weeks.