Visual Analysis on the Resilience of HPC Applications

Hailong Jiang [1] Shaolun Ruan [2] Yong Wang [2] Bo Fang [3] Kevin Barker [3] Ang Li [3] Qiang Guan [1]

[1]Kent State University [2]Singapore Management University [3]Pacific Northwest National Laboratory

Motivation

- The resulting resilience characteristics of a series of program states can be scattered, lacking a holistic view for the users;
- To classify and summarize the unstructured resilience-related data generated by those approaches requires enormous efforts for large-scale HPC applications;
- There is no platform to post-analyze the results from different resilience frameworks.

VISILIENCE Overview

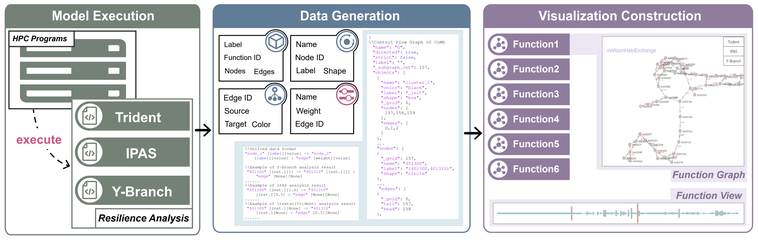

Figure 1. An overall overview of VISILIENCE

(A) Three resilience analysis models in Resilience Analysis part: TRIDENT [2], IPAS [1] and Y-Branch [3];

(B) Data Generation part generate CFG data , and encodes the resilience analysis results into a unified format;

(C) Visualization Engine takes the formatted data and CFG data as input and outputs an interactive visual interface of the resilience analysis results.

Data Generation

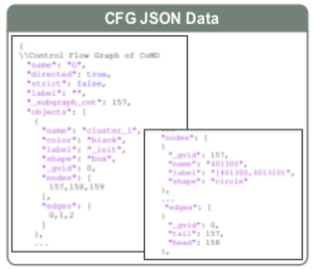

Figure 2. CFG json data

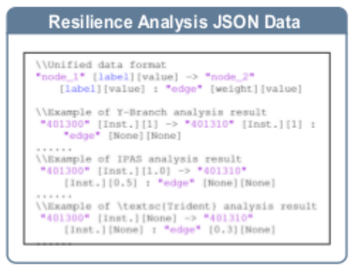

Figure 3. Resilience analysis json

Data Generation (Cont.)

The Three analysis models above analyze resilience on different levels and output three data formats. The Data Transformer part encodes the resilience analysis result to a unified format shown in Figure 2 and passes it to the Visualization Engine.

The first line in resilience analysis json data in Figure 3 shows the unified format of data. The *"node_number"* and *"edges"* are the same as those in Figure 4 (a). The *"label"* and *"value"* of each elements are the Data Interface between Data Transformer and Visualization Engine.

Visual Encoding

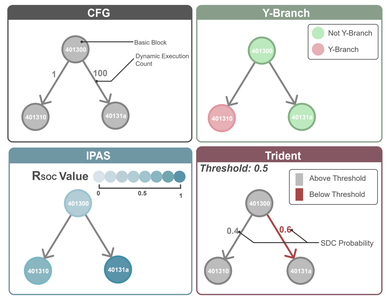

Figure 4. Visual encoding diagrams

- CFG
  - Nodes: Basic Blocks
  - Edges: Control Flows
- Y-Branch
  - Nodes: Green/Red -> Y-Branch/not
  - Edges: Control Flow
- IPAS
  - Nodes: Darkness -> Rates of SOC inst.
  - Edges: Control Flow
- Trident
  - Nodes: Basic Blocks
  - Edges: Control Flow
  - Weights: SDC probability

Interface and Web link

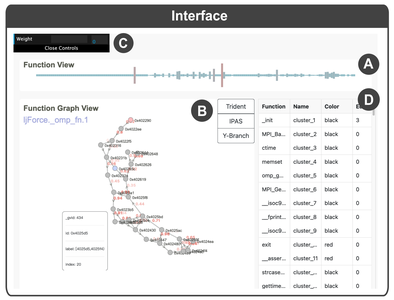

Figure 5. The interface of VISILIENCE

- A Function view is a series of dots at top represent the functions;
- B The graph is shown in the Graph view and the nodes are basic blocks;
- C Weight threshold is used to set the weight threshold;
- D The functions with specific names are listed in Function List.

Figure 6. The web link of VISILIENCE

Case Study

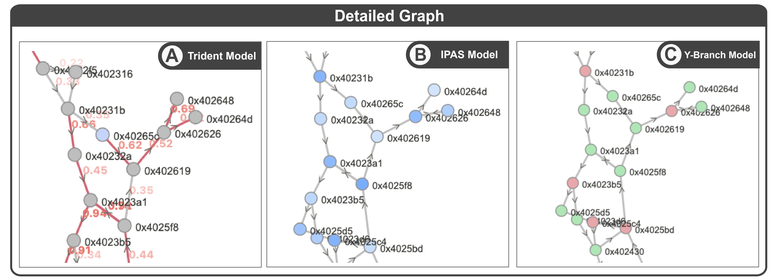

Figure 7. The snapshots of the partial view of CoMD benchmark under three models

Trident

The weights on the edges are the SDC propagation possibilities between basic blocks. The weight threshold bar on the very left top can be slid from 0 to 1.

IPS

The darkness of the nodes represents the SOC-generating-instruction rate: the darker the colour, the higher the rate.

Y-Branch

Basic blocks in Y-Branch node are green or red, representing Y-branch and non-Y-Branch.

Conclusion

In this poster, we present VISILIENCE, a visual resilience analysis framework to show the resilience analysis results to programmers in an intuitive way. VISILIENCE takes the Control Flow Graph as a layout and maps the resilience analysis data on it. VISILIENCE conducts three resilience analysis models and encodes these data into a unified data format, and visualizes the data into an interactive interface. The Visualization Engine provides several human-computer interactions, which help the users understand the data better. Multiple case studies have been conducted to demonstrate the effectiveness of VISILIENCE.

References

[1] Ignacio Laguna, Martin Schulz, David F. Richards, Jon Calhoun, and Luke Olson. Ipas: Intelligent protection against silent output corruption in scientific applications. CGO 2016, 2016.

[2] G. Li, K. Pattabiraman, S. K. S. Hari, M. Sullivan, and T. Tsai. Modeling soft-error propagation in programs. In *2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pages 27–38, June 2018.

[3] Nicholas Wang, Michael Fertig, and Sanjay Patel. Y-branches: when you come to a fork in the road, take it. *Parallel Architectures and Compilation Techniques, 2003. PACT 2003. Proceedings. 12th International Conference on*, 2003.