Technische Universität Braunschweig

Suresoft
Sustainable Research Software

Institut für Betriebssysteme und Rechnerverbund
Verteilte Systeme

Experience Report on the Sustainable development of a Byzantine Fault-Tolerant Framework

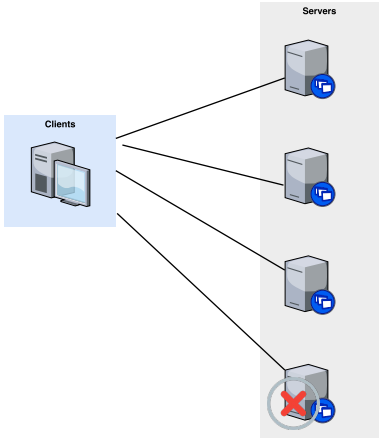
Themis Framework

Institut für Betriebssysteme und Rechnerverbund, May 5, 2022

IBR Use-case: BFT frameworks

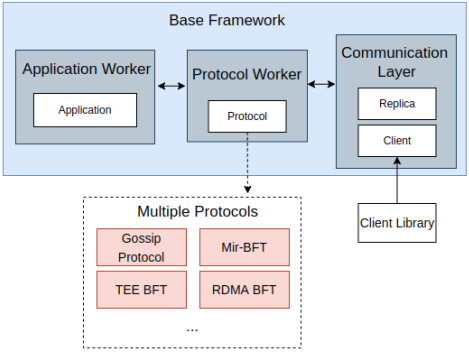
Our Motivation: Byzantine Fault Tolerant frameworks

- Fault-tolerant distributed framework
  - → Needs automated deployment on multiple machines
- Bugs mean invalidation of the protocol
  - More vulnerabilities and attacks
  - → Need integration and unit testing
- Complex and fragile implementation
  - → Need to be continuously maintained
- Different versions and optimization
  - → Keep track of the environment for reproducibility



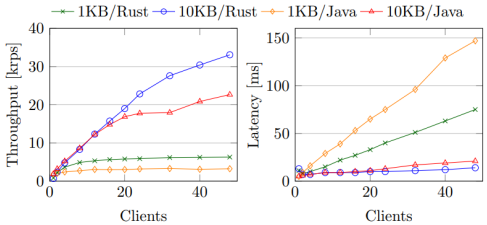
About Themis Framework

- An Efficient and Memory-Safe BFT Framework in Rust [Rüsch et al., BCRB'19]
- Various BFT versions are based on Themis (e.g., different signing methods)
- Base framework is ≈ 14k SLOC
- Continuously adding features and improvement



Themis Main Challenges

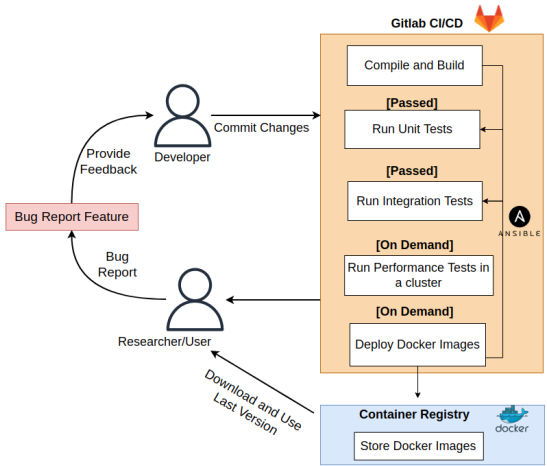
- Effect of new updates on the performance
  - New features should not slow down the system
- Correctness of the newly written features
  - Continuous good code quality



→ **Preserving good performance and code quality is key!**

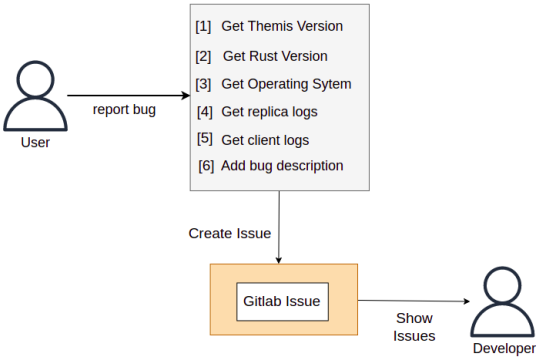
Applying Suresoft to Themis Framework

- **Code quality challenge**
  - Automated unit and integration tests
  - Bug reporting
  - Static code analysis using *Miri* for rust
- **Performance challenge**
  - Distributed monitoring tests
  - Throughput, Latency and Memory
  - Comparing versions and changes (e.g., increase 7%)
  - Automated deployment using *Ansible*

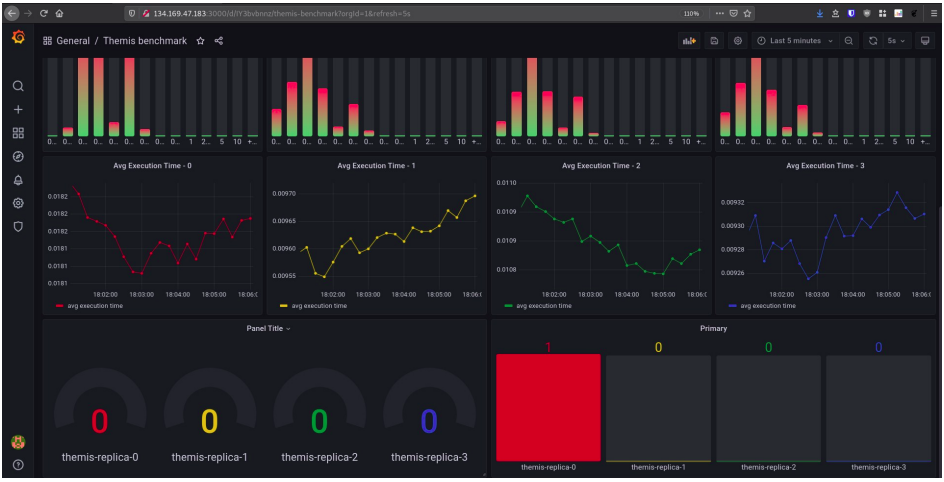


Reporting Issues

- Users can provide feedback
- Automatically gets important information
- Creates issues in Gitlab
- Included in the deployed version

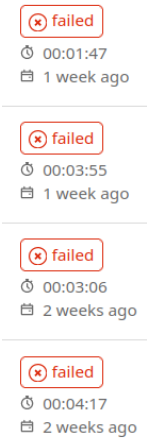


Performance Monitoring



More Resulting Challenges

- Multiple authors with varying experiences
  - Each member needs to test, document and follow the architecture
- Active development means changing interface
- Deployment to custom server is not always guaranteed
- Some distributed tests are discontinued
  - Reliance on specific machines and account



Lessons learned

- Remove tests that are not valid and deprecated due to code updates
- Continuously maintain your test suite and CI pipeline
- Emphasize the importance of documentation and testing for each member
- Use code-coverage to highlight chunks of code you have missed
  - Aim for functions coverage
  - Our test coverage decreased from 64.89% to 62.28%

↪ Documentation is key

Sustainability is a spectrum that requires constant engagement

Work In Progress and Future Plans

- Distributed Tests
  - Detect free unreserved machines to run measurements
  - Use a bot account for connections
  - Use *ansible* for the distributed tests
- Show a full report between versions, running it with various configuration and display it in a dashboard
- Add archiving support provided by the GITZ
- Enable reproducibility of the archived results and logging newly updated ones