Chaos Engineering Day Stockholm edition, 2017

Organization: Martin Monperrus, KTH

http://chaos.conf.kth.se/

Chaos Engineering Day?

- Goals:
  - Meet: Know each other
  - Learn: High quality technical presentations
  - Plan: Next collaborations in research, industry and open-source
- Worldwide
  - San Francisco (Nov 4 2015 & 2017), Seattle (Aug 26 2016)
  - Paris (Nov 24 2017), London (Dec 12 17)

Presentation of each participant

Key Statistics

- 1 keynote, 5 presentations
- 43 participants (and counting...)
  - from industry, from academia
- From many countries: Sweden, Norway, Spain, France, Germany, Denmark

Program (morning)

- Presentations are allocated 20 minutes (incl. questions).
- Agenda slack and interactions is builtin

9:15-9:45 Workshop introduction (Martin Monperrus, KTH)

9:45-10:45 Keynote: "Let it crash" (Joe Armstrong, father of Erlang)

10:45-11:10 Coffee Break

11:10-11:30 "Lineages as a first-class construct for fault-tolerant distributed programming" (Philipp Haller, KTH),

11:30-11:50 "Configuration testing for better DevOps" (Anatoly Vasilevskiy, SINTEF)

12:00-13:15: Lunch (free for registered participants)

13:15-14:00: Parallel sessions (email, walk)

Lunch

- 12:00 – 13:15 (10 minutes walk)
- 13:15 – 14:00 Email session and walk session

Program (afternoon)

14:00-14:20 "Continuous Diversification in a DevOps pipeline" (Nicolas Harrand, Univ Rennes)

14:20-14:40 "High Frequency Chaos Engineering" (Mats Jonsson, SAAB)

14:40-15:00 "Correctness Attraction: Runtime Perturbation for Full Correctness" (B. Danglot, Inria)

15:00-15:30 Coffee break

15:30-16:15 Breakout Group Discussion

16:15-16:30 Presentation of group results

16:30-16:45 Closing

Wifi

- Network eduroam: your institutional login

- Guest Logins, see sheet

Acknowledgements

- Presenters
- Participants
- KTH CASTOR Center for Software Research for funding
- Sandhya Hagelin (KTH Service) for the organization

Introduction to Chaos Engineering

Chaos Engineering Examples

- Chaos monkey:
  - Automatically and randomly shutdown servers
  - Verifies that the system withstand crashes
  - Abstract over a wide range of problems (HW, OS, SW)
- Gameday exercise
  - Simulates a network partition isolating a whole data center
  - Planned and monitored

Chaos Engineering Definitions

- “Chaos Engineering is the discipline of experimenting on a distributed system in order to build confidence in the system's capability to withstand turbulent conditions in production” (principlesofchaos.org)

- “Chaos Engineering is the discipline of experimenting on a **software system** in production in order **to verify a property**”

Chaos Engineering Definition

"Chaos Engineering is the discipline of perturbing a software system in production **for fun and profit**"

(working definition for today)

Chaos Engineering Related Work

- The scientific method
- Popper’s falsifiability
- Ghost planes (1975!)

Chaos Engineering Related Work

- Randomization & software diversity
- Testing;
  - In-the-field testing
  - Stress testing
- Devops:
  - Canari testing / Rolling deployment
  - A/B testing
  - Disaster recovery

Chaos Engineering Methodology

- Invariant: measurable output that indicates normal behavior.
- Failure model: reflect real world events like crash.
- Hypothesis: control group and experimental group.
- Try to falsify the hypothesis by looking for a difference in steady state between the control group and the experimental group.

Chaos Engineering Research

- Perturbation models
  - Coarse-grain: crash
  - Fine-grain: nullify a single variable
  - Human based
- Perturbation gains & costs
  - Chaos monkey: zero cost
- Perturbation controller
  - Targeted perturbations
  - Use of undo? Use of isolation?
  - Maximize the gained knowledge

Chaos engineering and open-source

- Netflix: Simianarmy (Java) and chaosmonkey (Go)
- jepsen-io/jepsen
- Kube-monkey (chaos for Kubernetes)
- Pumba (chaos for Docker)
- os-faults & destroystack (openstack)
- faulterl: Erlang library-level fault injection
- See also list at https://www.oreilly.com/ideas/chaos-engineering