(Yet) Another Attempt at Online Failure Prediction

Nuno Antunes
nmsa@dei.uc.pt
Department of Informatics Engineering
University of Coimbra - Portugal
**Failure Prediction**

Uses prediction models trained with data of failure events
- Data can be numerical (e.g., free memory) or categorical (e.g., events)
- Models can be built using machine learning, statistics, etc.

- Salfner & Malek’s model:
  - Predictors trained using data from $\Delta t_d$
  - Prediction performed at time $t$ for failures occurring in the interval $\Delta t_l \pm \Delta t_p$
    - $\Delta t_w$ is the minimal time below which (even) a predicted failure cannot be avoided
  - Output: 0/1, failure probability
NOT A NEW CONCERN

HotDep
Fault Injection for Failure Prediction Methods Validation

Marco Vieira\textsuperscript{1}, Henrique Madeira\textsuperscript{1}, Ivano Irrera\textsuperscript{1}, Miroslaw Malek\textsuperscript{2}
\textsuperscript{1} University of Coimbra – Portugal
\textsuperscript{2} Humboldt-Universität zu Berlin – Germany
mvieira@dei.uc.pt, henrique@dei.uc.pt, ivano@dei.uc.pt, malek@informatik.hu-berlin.de
Problems with failure prediction…

- Obtaining training data is hard
  - Fortunately, failures are rare events!
- Identifying the relevant data for training is difficult
- Selecting the most adequate algorithm(s) is complex
- Furthermore: systems change over time!!!
Generating failure-related data using realistic software fault injection + virtualization
Assessment and comparison of failure prediction systems
PROBLEMS...

Systems change

- Virtual machines are not the ideal solution
- Hard to implement in complex systems
- Boundaries of the system are unclear

Practical Applicability
WHAT ABOUT CONTAINERS?

Containerized applications based on microservices are highly flexible and scalable

- Widely spread, e.g. in cloud environments
- Isolation
- Stability in the context surrounding the application
- Boundaries
- Easy to replicate and manage

This may be what we need to make O.F.P work!
Use **data** that are **only about the container**

- OS data cannot be considered
- Docker API, cAdvisor
Use **data** that are **only about the container**

- OS data cannot be considered
- Docker API, cAdvisor

- Consider each container individually
  - Well defined boundaries!
  - Each may contribute to higher level models
Use **data** that are only about the container

- OS data cannot be considered
- Docker API, cAdvisor

- Consider each container individually
  - Well defined boundaries!
  - Each may contribute to higher level models

- Automate replication and fault injection to handle the “Online” part
  - Easier to do in containers
Are the container-dependent variables enough to make state of the art approaches work?
– We do no plan to develop new ones

- Are the monitored variables really consistent across containers running the same workloads?
- Fault injection and representativeness thereof
SUMMARY

We believe that practical applicability is the current key issue

- Containers make a set of assumptions valid that may help us to solve the problem
- We are just starting...

- Not a Silver Bullet
  - Obviously, this cannot be applied to every application
  - Application that fit the containerized model are suited
    - e.g. Microservices
Questions?

FOR A FAIR SELECTION EVERYBODY HAS TO TAKE THE SAME EXAM: PLEASE CLIMB THAT TREE

Nuno Antunes
Department of Informatics Engineering
University of Coimbra
nmsa@dei.uc.pt | http://eden.dei.uc.pt/~nmsa