DEBS Grand Challenge -- Non-functional requirements

Sebastian Frischbier, Ruben Mayer, Christoph Doblander, Jawad Tahir

23.02.2021

Purpose

Pushing the boundaries of event-based and distributed systems with novel approaches and their practical validation is at the heart of the DEBS conference series.

By their nature these novel solutions tend to be very much ahead of the status quo of systems' engineering and custom-tailored to solve a very specific use case. This hampers reuse and adoption by others than the original authors - both in research and in practise. Furthermore, research prototypes for novel solutions usually require a lot of engineering adaptation until they can be adopted in practise as they need to address additional non-functional requirements such as configurability, scalability, and operational resilience.

The DEBS Grand Challenge 2021 aims at fostering a more widespread reuse and adoption of novel solutions inside and outside of the DEBS community by rewarding solutions that can be easily adopted by others. For this purpose we will mark out solutions that are implemented in a way to be easily reused, adapted, and operated by other researchers and by practitioners; in particular solutions that are built on top of widely-used industry-strengths open-source platforms like those curated by recognised Open Source foundations (https://opensource.com/resources/organizations). We promote these open-source platforms not only because they already have a diverse user base and ecosystem but because most of them already solve important non-functional requirements that are paramount for a use in practise.

Criteria

While all submitted solutions strive for maximum performance we want to encourage members of the community to push beyond solving a problem correctly in the functional sense to providing a prototypical implementation that allows others to build upon with ease.

Thus, every submission will also be evaluated for how it addresses certain non-functional criteria in its design and implementation. For the audience award we will reveal to the audience for each submission, which features are indeed implemented and to what level of sophistication.

In this regard we distinguish between **hard** and **soft** non-functional requirements.

- **Hard** criteria are must-haves that have to be described and implemented; while
- **Soft** criteria do not necessarily have to be implemented but must be covered in the submission's description

Generally, we encourage distributed solutions to address the impact of contemporary operational models, infrastructures (i.e., private/public/hybrid cloud, edge computing), and legal requirements (e.g., data locality due to GDPR or domain-specific legislation like the Swiss banking secrecy law). Furthermore, performant distributed solutions have to specifically excel at challenges like:

- State management (distributing stateless applications is a no-brainer)
- Privacy (like in edge-computing; you are not allowed to centrally compute all data but need to pre-process them in different locations → see data locality requirements of GDPR or Swiss Bankengeheimnis law)
- Bandwidth restrictions (e.g., edge-computing)
- Trusted computing / Checks-and-balance (as there is no central authority)

Hard criteria / requirements

Hard criteria have to be addressed in the submitted paper, even if they are not addressed by the design or have not been implemented.

Configurability

How can your solution be configured? Do you have a way to configure your system or is it hard-wired? Do you have a query language or do you rely on queries to be implemented in a high-level language?

Please elaborate at least on the (a) spectrum of expressiveness: hard-wired vs expressive configuration, and (b) the richness of the mechanism (e.g., query language, config parameters).

Scalability

How does your solution scale to keep up with an increasing workload?

Describe how your solution scales logically and how your implementation can scale practically if confronted with an increasing volume of input data (e.g., by adding computation nodes, by increasing the resources available to existing processes etc).

Remark: Generally, horizontal scalability is preferred over vertical scalability due to contemporary operational models (i.e., deployments on cloud infrastructure).

Operational reliability / resilience

In practise, your solution must be operated on virtual or physical infrastructure. Failures on that level can have a severe impact on the availability and performance of your solution. They might not be your fault but they will be your problem if you do not prepare your design and implementation for it.

Provide a statement on how you address topics like

- Infrastructure outages (e.g., network failure, virtual machine state lost etc.)
- Recoverability (e.g., how do you recover the state of your solution?)
- Maintainability (i.e., how to deal with maintenance work on infrastructure such as upgrades of middleware etc that are powering your solution)?

Accessibility of the solution's sources

Provide a statement on whether your code is freely available. Either provide a link to where the code can be found or provide a statement on why you are not giving access to your code (e.g., patent pending etc).

Integration with standard tools

Please share whether you use or rely on standard tools (open source or commercial).

Documentation

Please provide input on whether additional technical documentation is available apart from the submitted paper and the source code.

Soft criteria / requirements

Soft criteria can be addressed in the paper but do not necessarily need to be covered by the implementation. With this we want to give those authors a stage that have considered any of the following soft criteria in their solution’s design.

Examples for soft criteria are:

- Security (encryption, authentication, authorisation)

- Deployment
    - Packaging (e.g., containers / serverless)
    - Build pipeline (ready-to-use / packaged)

- Utilization of special hardware
    - Examples
        - FPGAs
        - SDNs
        - GPUs