Performance Evaluation of BaSyx based Asset Administration Shells for Industry 4.0 Applications[*]

Christian Sauer, Holger Eichelberger
{sauer, eichelberger}@sse.uni-hildesheim.de
University of Hildesheim, Hildesheim, Germany

Abstract

The Asset Administration Shell (AAS) is an upcoming information model standard, which aims at interoperable modeling of "assets", i.e., products, machines, services or digital twins in IIoT/Industry 4.0. Currently, a number of IIoT-platforms use proprietary information models similar to AAS, but not a common standard, which affects interoperability.

A key question for a broad uptake is if AAS can be applied in a performant and scalable manner. In this paper, we examine this question for the open source Eclipse BaSyx middleware. To explore capabilities and possible performance limitations, we present four experiments measuring the performance of experimental AAS in BaSyx and, within the context set by our experiments, i.e., 10-1000 AAS instances, can conclude good scalability.

1 Introduction

In IIoT/Industry 4.0 information access and information sharing is essential for the efficient coordination and planning of a production. Especially in IIoT-platforms and the numerous Industry 4.0 applications within them there is the need to represent any relevant entity in an easily accessible way. Furthermore, it is important to allow to interact with these entities, e.g., control a machine in a standardized manner or let a product proactively determine its production flow. One approach that can be used for these purposes is the upcoming Asset Administration Shell (AAS) [6] standard, which aims at interoperable modeling of Industry 4.0 "assets", i.e., products or machines. An AAS is a virtual representation of Industry 4.0 components, employing a standardized information model. In our research we identified a number of commercial IIoT-platforms that already use information models similar to AAS but do not rely on a common standard [4], for example Amazon AWS IoT, Bosch IoT Suite and SAP Leonardo. However, these approaches are proprietary for each platform and hence are hindering if not outright preventing interoperability outside of the respective platform and/or between different platforms.

An AAS consists of information on an asset being structured into it's properties, and possible operations it can perform. This information is stored in submodels, which can be grouped in collections. A detailed review of the information model of an AAS is given in [6]. The information stated in an AAS can further be linked to external catalogs, such as ECLASS[1], using so-called semanticIDs. AAS can be passive or active. A passive AAS only provides the information on the asset it describes. An active ASS can additionally be used to control an asset. Further, AAS can be used locally or over a network as network-connected AAS [1, 6]. Offering access to information and to interfaces to control an asset, AAS can, from a software perspective, be seen as functional interfaces allowing for remote access in a transparent way.

One initiative in the sector of Industry 4.0 in Germany is the "Basissystem Industrie 4.0" (BaSys4) defining a reference architecture for production systems[2]. One realization of the BaSys4 architecture is the open-source middleware Eclipse BaSyx[3], which enables creating, using and managing AAS [3]. The structure of various types of submodels is currently in standardization by the IDTA[4], where BaSyx is considered as a reference implementation and hence of interest for us. Further AAS realizations do exist, which are out of scope for this paper. A possible use of BaSyx-based AAS raises questions of performance and scalability, especially with the background of the scale of Industry 4.0 installations and the sometimes limited capabilities of industrial devices, on which AAS would operate. Therefore the aim of our work was to analyze the performance of Basyx-based AAS.

A further motivation is that in the context of the project IIP-Ecosphere[5], we are currently involved in the development of an open source AI-enabled Industry 4.0 platform. In this platform all resources and (software) services provide a self-description in terms of an AAS. We therefore face the question whether it is beneficial to integrate BaSyx into the platform.

[*]IIP-Ecosphere is partially supported by the German Federal Ministry of Economic Affairs and Climate Action (BMWK) under grant 01MK20006D.

[1]`https://eclass.eu/`
[2]`https://www.basys40.de/`
[3]`https://www.eclipse.org/basyx/`
[4]`https://industrialdigitaltwin.org/`
[5]`https://www.iip-ecosphere.eu/`

The rest of this paper is structured as follows: In Section 2, we present the approach that we used to evaluate the performance of the BaSyx-based AAS. Then we discuss the experiments we conducted and their results in Section 3. In Section 4, we conclude the paper and give an outlook on future work.

2 Approach

The approach we used to evaluate the performance of the creation, use and management of BaSyx-based AAS was as follows: We created 4 experiments, 2 to evaluate "local" AAS, which are directly accessed, discerning the use of 1 local AAS versus the use of 100 local AAS and 2 experiments to evaluate "network-connected" AAS, which are accessed via a locale network, to avoid fluctuations, using an AAS-Registry, discerning the use of 1 network-connected AAS versus the use of 100 network-connected AAS. The experiments aimed to measure the time being used to create BaSyx-based AAS' with a varying number of submodels, each holding a property (a temperature value from a "sensor"). Further, we measured the time to access the sensor-submodels of the network-connected AAS. This setup is based on an initial BaSyx example of a local and a network-connected AAS[6], which we modified to perform our measurements. One particular modification is the number of AAS/submodels to be created in order to analyze scalability. In related work on the use of BaSyx-based AAS, a conceptual view on the use of BaSyx-based AAS in Industry 4.0 is given in [5]. An alternative approach of measuring the performance of BaSyx-based AAS is demonstrated in [2].

3 Experiments and Results

We present now the experiment setup and the results[7].

Experimental environment: We set up a technical environment on a standard PC with an Intel(R) Core(TM) i7-8665U CPU 1.90GHz and 32 GB RAM. The OS used was Windows 10 Professional. To accommodate for possible fluctuations,which usually were very minor but occasionally significant, each measurement was performed 10 times and the results were averaged. The Software environment that we used was as follows: We used the Eclipse IDE for Enterprise Java and Web Developers, version 2022-06 (4.24.0), Java Development Kit (JDK) version 15, Apache-Tomcat Server, version 9.0 and the Eclipse BaSyx version available on January the 3rd 2022.

Subject and variants: Our experiments aimed to measure the response time necessary to create first 1 and then 100 AAS and their subsequent 10 to 1000 sensor-submodels, each holding one property that stores a temperature as a single value, to investigate the effects of scaling up the AAS use to expected industrial levels. The four variants of the experiment were as follows: All four experiments consisted of 11 iterations, starting with 10 sensor-submodels and then increasing the number of submodels per AAS in ten iterations to 1000 sensor-submodels. The execution time for each iteration was measured, based on system time at start and end of the experiment, 10 times and the averaged time for each iteration was used.

- Experiment 1 created 1 local AAS with 10 sensor-submodels in the first iteration, and then increased the number of submodels to 1000.
- Experiment 2 created 100 local AAS with 10 sensor-submodels each in the first iteration, and then increased the number of sensor-submodels to 1000 submodels for each of the 100 AAS.
- Experiment 3 created 1 network-connected AAS with initially 10 sensor-submodels, and then increased the number of sensor-submodels to 1000.
- Experiment 4 created 100 network-connected AAS with 10 sensor-submodels each in the first iteration, and then increased the number of sensor-submodels to 1000 submodels for each of the 100 network-connected AAS.

Experimental procedure: The generation of the AAS' and their submodels and the accessing of the information within the submodels was measured using milliseconds in system time, each experiments' 11 iterations were all repeated 10 times and the measured times of the iterations were averaged. After each experiment iteration with network-connected AAS the Apache Tomcat Server was cleaned and restarted.

Results: The following figures show the results of our experiments with the numbers of submodels in each iteration as blue bars and the averaged times for each iteration (in ms) as an orange graph.

- For Experiment 1 (Figure 1) the amount of time used to generate the one local AAS and subsequently increasing numbers of sensor-submodels almost stayed level, which clearly indicates scaling the number of sensor-submodels for a single AAS, does not significantly impact performance.

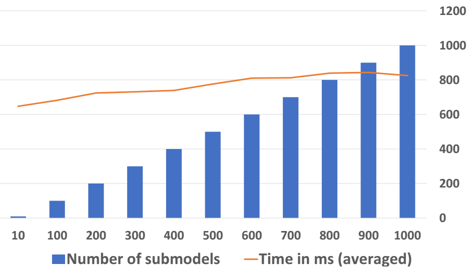

Figure 1: Experiment 1, 1 local AAS

- For Experiment 2 (Figure 2) we found that with 100 locale AAS the effort of creating the AAS

[6]https://wiki.eclipse.org/BaSyx_/_Introductory_Examples

[7]Code and results: https://zenodo.org/deposit/7180716

and their subsequently increasing numbers of sensor-submodels (up to 1000 sensor-submodels per AAS, so 100 AAS with 100000 submodels in total) increased, as expected. The increase for each iteration is almost linear.

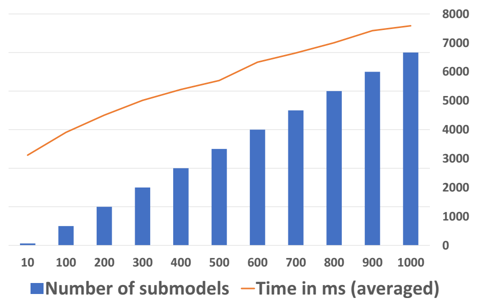

Figure 2: Experiment 2, 100 local AAS

- Experiment 3 (Figure 3) showed a significant increase in effort when we used network-connected AAS, compared to the locale AAS we used in Experiment 1. Interestingly, the increase in effort was more prominent in the lower range of the number of sensor-submodels being created and accessed. The increase of effort for more than 400 sensor-submodels became more linear, which already indicated the linear increase in effort we observed then in Experiment 4.

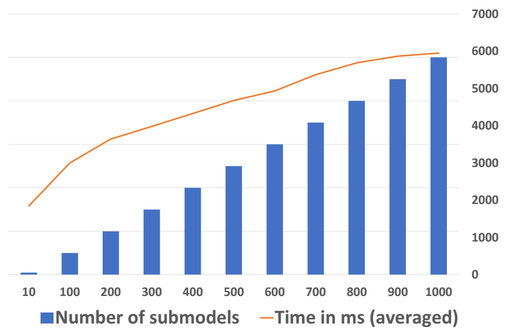

Figure 3: Experiment 3, 1 network-connected AAS

- For Experiment 4 (Figure 4) we found that with the 100-fold increase in scale, 100 network-connected AAS, in comparison to the one network-connected AAS in Experiment 3 the scaling of the sensor-submodels in the networked environment and the reading of data from them showed a very clear linear increase in effort.

4 Conclusion

We presented a performance and scalability evaluation of BaSyx-based AAS to estimate potential impacts on Industry 4.0 applications using them on a larger scale.

Although we ran a limited set of experiments, based on our results we conclude that the creation and use of BaSyx-based AAS, passive, as well as active AAS,

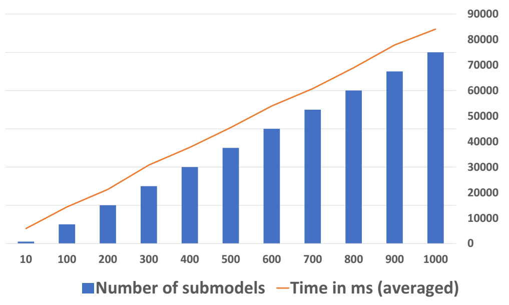

Figure 4: Experiment 4, 100 network-connected AAS

is efficient enough to scale well in even larger numbers, as shown in Experiment 4, with 100 network-connected AAS with 1000 submodels each being generated and the resulting 100000 submodels being read. We therefore can say that in the context of our experiments the BaSyx-based AAS scaled well with the AAS-operations we tested. However, we would like to advice that our experiments were limited. We didn't delete AAS and/or submodels, we always performed the same operations with the AAS' on a local computer and we didn't use all of the possible elements in an AAS, such as using semanticIDs or AAS references.

In future, we plan to further explore the possibilities to evaluate the functionalities and elements of BaSyx-based AAS, such as the scalability of real-world AAS implemented with BaSyx, for example the implementation of AAS currently used in the IIP-Ecosphere platform, instead of artificial example AAS.

References

[1] N. Chilwant and M. S. Kulkarni. "Open Asset Administration Shell for Industrial Systems". In: *Manufacturing Letters* 20 (2019), pp. 15–21.

[2] M. G. Casado and H. Eichelberger. "Industry 4.0 Resource Monitoring - Experiences with Micrometer and Asset Administration Shells". In: *Symposium on Software Performance*. 2021.

[3] S. Kannoth et al. "Enabling SMEs to Industry 4.0 Using the BaSyx Middleware: A Case Study". In: *European Conference on Software Architecture*. Springer. 2021, pp. 277–294.

[4] C. Sauer et al. *Current Industry 4.0 Platforms - An Overview*. doi:10.5281/zenodo.4485756. 2021.

[5] P. O. Antonino et al. "Continuous engineering for Industry 4.0 architectures and systems". In: *Software: Practice and Experience* 52.10 (2022), pp. 2241–2262.

[6] S. Bader, E. Barnstedt, and H. B. et al. *Details of the Asset Administration Shell*. `https://www.plattform-i40.de/PI40/Redaktion/DE/Downloads/Publikation/Details_of_the_Asset_Administration_Shell_Part1_V3.html`. (Date accessed: 24.08.2022).