Publication Year	2018
Acceptance in OA@INAF	2020-11-11T17:21:41Z
Title	Very large scale high performance computing and instrument management for high availability systems through the use of virtualization at the Square Kilometre Array (SKA) telescope
Authors	Morgado, J. B.; Maia, D.; Barbosa, D.; Barraca, J. P.; Bergano, J.; et al.
DOI	10.1117/12.2313559
Handle	http://hdl.handle.net/20.500.12386/28266
Series	PROCEEDINGS OF SPIE
Number	10707

PROCEEDINGS OF SPIE

SPIEDigitalLibrary.org/conference-proceedings-of-spie

Very large scale high performance computing and instrument management for high availability systems through the use of virtualization at the Square Kilometre Array (SKA) telescope

Morgado, J. B., Maia, D., Barbosa, D., Barraca, J. P., Bergano, J., et al.

J. B. Morgado, D. Maia, D. Barbosa, J. P. Barraca, J. Bergano, D. Bartashevich, M. Di Carlo, M. Canzari, M. Dolci, R. Smareglia, "Very large scale high performance computing and instrument management for high availability systems through the use of virtualization at the Square Kilometre Array (SKA) telescope," Proc. SPIE 10707, Software and Cyberinfrastructure for Astronomy V, 107070I (6 July 2018); doi: 10.1117/12.2313559

SPIE.

Event: SPIE Astronomical Telescopes + Instrumentation, 2018, Austin, Texas, United States

Downloaded From: https://www.spiedigitallibrary.org/conference-proceedings-of-spie on 13 Oct 2020 Terms of Use: https://www.spiedigitallibrary.org/terms-of-use

Very large scale high performance computing and instrument management for high availability systems through the use of virtualization at the Square Kilometre Array (SKA) telescope

J.B. Morgado[a*], D. Maia[a], D. Barbosa[b], J.P. Barraca[b], J. Bergano[b], D. Bartashevich[b], M. Di Carlo[c], M. Canzari[c], M. Dolci[c], R. Smareglia[d]
[a]CICGE, Faculdade de Ciências da Universidade do Porto, 4430-146 V. N. Gaia; [b]Instituto de Telecomunicações, Campus Universitario de Santiago, 3810-193 Aveiro; [c]INAF Osservatorio Astronomico d'Abruzzo, Teramo, Italy; [d]INAF Osservatorio Astronomico di Trieste, Trieste, Italy

ABSTRACT

The Square Kilometer Array (SKA) Telescope, is an ongoing project set to start its building phase in 2019 and achieve first light in 2022 The first part of the project, the SKA1 will be comprised of 130.000 low frequency antennas (50 MHz to 350 MHz) and about 200 mid frequency antennas (350 MHz to 15.5 GHz). The SKA1 will produce a raw data rate of ~10 Tb/s, requiring a computing power of 100 Pflop/s and an archiving capacity of hundreds of PB/year. The next phase of the project, the SKA2, is expected to increase the number of both low and mid antennas by a factor of 10 and increase the computing requirements accordingly. The key requirements for the project are a very demanding availability of 99.9% for its operations, computing scalability and scientific outcome reproducibility. Focusing on the SKA Telescope Manager requirements and architecture, we propose an approach to enforce these requirements - with an optimal use of resources - by using highly distributed computing and virtualization technologies.

Keywords: SKA, Telescope Manager, HPC, Virtualization, Radio Astronomy, Reproducibility

INTRODUCTION

The Square Kilometer Array (SKA) Telescope, is an ongoing project set to start its building phase in 2019 and achieve first light by 2022, with an expected SKA1 system fully operational by 2025. The SKA will gather radio signal observations from the all sky in a high range of frequencies from 50 MHz to 14 GHz, expected to be operational by 2018 and collecting data for at least 20 years. The first phase of the project, the SKA1, will consist of 130.000 low frequency radio antennas - operating in the ~50 MHz to ~350 MHz band - and about 200 mid frequency radio antennas - operating in the 350 MHz to 15.5 GHz band, incorporating the MeerKAT precursor 64 antennas. For Phase, ie SKA2,. the project will expand in frequency and spread stations in Africa and Australia over an area of 3000 square kilometers in two continents - Africa and Oceania.

The SKA1 will produce a very high raw data rate of approximately 10 Tb/s, require a computing power of 100 Pflop/s and an archiving capacity for science products - for the already reduced data - of up to 300 PB/year, just for acquiring, processing the signal, transfer the data products to data centers and store them. The computing requirements for doing science with these data products is expected to be several orders of magnitude higher. For comparison, in 2016 the worldwide total internet traffic was approximately 212.8 Tb/s [1], the present worlds' fastest supercomputer has a computing capacity of 93 Pflop/s [2] and at the date of its Initial Public Offering (IPO), Facebook stored approximately 100 PB of photos and videos [3] (although from 2012, this is the most recent storage metrics for any of Facebook, Google, Amazon or Apple). Due to the nature of data gathered by the SKA project and due to its operating time frame - extending for several decades - it is also of major importance to provision for future uses of data showing no interest for scientific production that could, on first approach, be discarded in order to keep the data storage, processing and transfer

Software and Cyberinfrastructure for Astronomy V, edited by Juan C. Guzman, Jorge Ibsen, Proc. of SPIE
Vol. 10707, 107070I · © 2018 SPIE · CCC code: 0277-786X/18/$18 · doi: 10.1117/12.2313559

Downloaded From: https://www.spiedigitallibrary.org/conference-proceedings-of-spie on 13 Oct 2020
Terms of Use: https://www.spiedigitallibrary.org/terms-of-use

costs down. For a practical example on the analysis of the use of data previously discarded as noise, that ended up having scientific interest going beyond the original purpose of the instrument, see Morgado et al. [10].

The next phase of the project, the SKA2, is going to increase the number of both low and mid antennas by a factor of 10 and, as such, increase all the computing, data transfer and storage requirements accordingly. However, we will benchmark activities in this paper for SKA1.

Observation operations and of course data transfer, computing and storage requirements have a key requirement of very high availability of 99.9% of the time, therefore requiring redundant systems in order to ensure that this specification is met. A final requirement, not only for SKA, but one that we expect to be transversal to most of scientific research in the near future, is that of *reproducibility*. Reproducibility is starting to be seen as a necessity for present scientific endeavours, mostly due to the greater availability of scientific data. As now, when more and more researchers have access to the data used in refereed publications, scientists are supposed to be able to reproduce the results from those publications independently, but in many cases we are failing at it in what is being aptly named a "*Reproducibility Crisis*" [8]. In the case of SKA, the issue may go even beyond that. The scientific outcomes will rely on such a massive amount of data and involve such a demanding amount of computation resources, that we may not afford the necessity to re-run specific computations or algorithms, a data reduction pipeline or data collection, in order to check if the results were readily achieved using any specific parameter configuration, an algorithm or a even specific workflows that connect information from well known astronomical databases or suites [11, 12]. As such, we argue, that for the SKA, reproducibility must be an enforced feature throughout the all SKA system.

The SKA Telescope Manager (ie TM) is an element responsible for the architecture design of the Observations Planning and Operations and the Monitoring and Control system of the SKA. This architecture includes the TM interaction through Internal Infaces with the compute intensive elements like the Science Data Processor (SDP) and the Central Signal Processing (CSP). Within SKA.TM, its Local Infrastructure Architecture (TM LINFRA) and the TM Services (TM SER) are solving part of these technological hurdles with the help of virtualization technologies. We propose an approach to enforce these requirements with an optimal use of resources, by using highly distributed computing and virtualization technologies. We also propose that by enforcing certain programing paradigms based on finite state machines, we can optimally use the computing resources available therefore reducing the hardware, maintenance and running costs of the project.

Virtualization technologies have been under heavy development during the last decade accompanying the change in computing paradigm from increasingly fast single processing units to an increasingly bigger number of processing units that show little gains in terms individual computing velocity but where the overall velocity of the system - when fully used - can show increases between less than 100% in the case of CPUs and several orders of magnitude in the case of GPUs. This change in computing paradigm meant a convergence between High Performance Computing (HPC) and Cloud based environments enabling promising data mining scenarios that paved the way to a renewed effort in parallel computing algorithms, and associated with it, an interest in virtualization and the ability to run several isolated threads of the same or different programs distributed across various computing units, including geographically distributed units.

Looking at the current landscape of virtualization technologies we chose to base the design of our system in the OpenStack platform [4] taking into account its modularity and the open source development that is being accomplished. Still, all the rationale presented in our work is platform agnostic and can be adapted and used with other virtualization technologies. We propose a system philosophy that can take into account the requirements for scalability, availability and reproducibility needed in SKA.

REQUIREMENTS

Assuming that the SKA will have several levels of interest and interaction we started building a system based on virtualization that will provide different tools to different key-holders while keeping the duplication of efforts to a minimum. Our main work so far, has been done inside TM, which purpose and scope are presented at great length in [5], but of special importance for the basis of our work in virtualization, is the nature of the deliverables inside TM. Identified as a TM App, these deliverables may be stand alone or exist in conjunction with other TM Apps and be long or

Downloaded From: https://www.spiedigitallibrary.org/conference-proceedings-of-spie on 13 Oct 2020
Terms of Use: https://www.spiedigitallibrary.org/terms-of-use

short term running services. TM components constitute one of the main critical system elements in the telescope. Aiming for a 99.9% availability, this aligns TM with a Tier 3 system as defined by the TIA-942 [9] standard. Therefore, SKA TM LINFRA is designed according to the ruling principles of such systems, as they reflect years of industry best practices for operating computational infrastructures.

Although we focus on TM, similar principles apply to the SKA Science Data Processor (SDP) element [6] - with the role of processing science data from the correlator or non-imaging processor into science data products - and to the future storage and provision of data through regional digital infrastructures that are currently in planning phase. For the later, and as an example, in Europe, the H2020 project Advanced European Network of E-infrastructures for Astronomy with the SKA (AENEAS) [7] - is architecting the design of the SKA data mining infrastructure federating the underlying large-scale e-infrastructures (compute, connectivity), to enable the scientific community at large to future access and exploitation of the collected data products.

The key requirements for a TM App, from the point of view of SKA1 TM LINFRA and SKA1 TM SER, and extendable to the Apps running in a virtualization layer in SDP or any future AENEAS infrastructure federations is then three fold:

- Modularization - An App should be defined in smaller, independent parts:

  This means an App should not be monolithic but instead rely on smaller modules. For instance, an App that uses an internal database, should run all pertaining functionality to that database in a module outside the main module and have the main module waiting for the availability of the database module. The rationale for this requirement, is that it will allow the underlying infrastructure of TM LINFRA to migrate these modules in real time between hardware resources, while the App is running. This capability, will at the same time, allow for failsafe implementations and the optimal use of computing resources.

- State based - An App should depend only on the state of the system as input in order to function:

  This is the principle behind a logical state machine. Although it adds complexity to the development process of an App, it will allow the App to run independent of its starting state. For example, if the App computation relies only on a set of configuration files, databases and temporary state description variables stored in a file, then it can be easily migrated in the middle of a computation and be able to pick up those computations exactly where it left them. This operating principle allied to a redundancy of the underlying computation layer, will ensure the high availability of the system and its scalability.

- Parallelization - An App should, as much as possible, be able to carry out its function while running in parallel with other instances of the same App:

  This is now a quite normal requirement in computing due to the nature of the change in computing paradigm from the last decade. We are steadily moving from increases in the velocity of processing in single threaded computing to a massive parallelization of computing resources. If an App is then capable of performing in parallel - and this requirement will be especially important for systems like the SKA SDP element and for future federation of e-infrastructures efforts like AENEAS - the underlying orchestration system is then capable of launching several threads of the same App if the available computing resources at a given time permit it, thus greatly speeding up the computation in some cases.

METHODS

In order to allow for the previously exposed requirements to integrate with the different systems - namely with at the TM level as a showcase - we propose that the access and the configuration of the various parts of the all system, built upon the Virtualization layer, relly on a three layered approach. At the base level, the configuration of the systems would be defined by SKA1 TM LINFRA, then a middle layer with access and computing resources agreed upon the various key players, and finally a top layer where the Apps would be able to perform the computation and require access between them and beyond them - as defined by the middle layer.

Downloaded From: https://www.spiedigitallibrary.org/conference-proceedings-of-spie on 13 Oct 2020
Terms of Use: https://www.spiedigitallibrary.org/terms-of-use

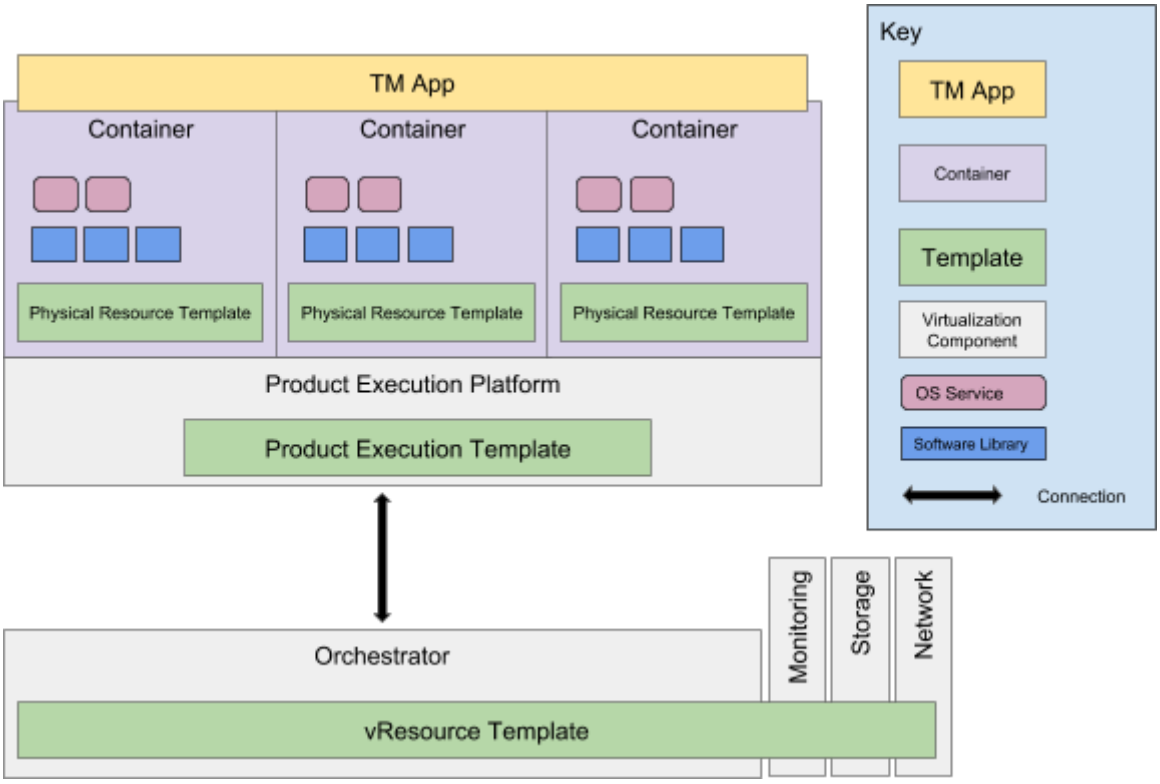

Fig 1: *Template and Instance presentation.*

As such, we predefined these three levels of complexity within the system and divided the needed configurations for each one. Figure 1 presents a view of this approach and it is specified as following:

- A Physical Resource Layer that will provide a uniform hardware view for all software:

  As the SKA1 will include tens of thousands of devices just for controlling the telescopes and acquire data, we want to provide all the different key-holders of the project with the means to build tools - deliverables - that will work independently of the hardware. At the same time we want to ensure that advances in computing hardware or a change in the computing resources providers can be met easily by the project. This absolutely avoids any adaptation process outside of the work done by TM LINFRA and TM SER.

- A Product Execution Layer that will consist of a distributed environment operating towards the provisioning of highly available products:

  SKA1 services will be composed of a multitude of software code components that needs to observe certain key points. One of those key points is to ensure that the adequate computing resources get allocated to each of these services and TM LINFRA will be able to automatically balance the hardware resources available to each service according to their immediate specific needs and the hardware resources available at the moment. Another key point is to ensure a high availability of the services, and in order to do so SKA TM LINFRA will detach the running hardware from a specific machine meaning that in case of hardware failure the service will continue to run on other available machines.

- A Virtualized Resource Layer, consisting of the virtual machines, containers and other hardware that has a logical representation to the Product Execution Layer and is part of a template or is available to be used by future templates:

  SKA services need a high degree of interoperability and communication, this part of the software abstraction will provide a deliverable product, modular in its nature, that will already have defined its hardware

Downloaded From: https://www.spiedigitallibrary.org/conference-proceedings-of-spie on 13 Oct 2020
Terms of Use: https://www.spiedigitallibrary.org/terms-of-use

access, resource usage and communication capabilities with other parts of SKA. By ensuring that it resides in a layer above the other two layers, it means that the specifics of the SKA product can be changed at a lower level and later deployed and put into production without disruptions to the service being provided.

What this all architecture achieves, is that the interface for the virtualization system - that will be the basis upon which all the SKA Telescope Manager software will run - will be constrained by exposing template actions externally according to the aforementioned three layers. The end objective is then to provide high availability, abstracting the underlying hardware infrastructure, and allowing software defined failover and horizontal scalability.

CONTEXT

The TM Virtualization Service is implemented at the lower level, below the domain logic and the infrastructure as presented in Figure 2. It follows the following layered structure:

- Domain/Business Layer: functional monitoring and controlling of business logic performed by each application;
- Services Layer: Monitors and controls processes on a generic level (non-functional) like web services, database servers, custom applications;
- Infrastructure Layer: Monitors and controls virtualization, servers, OS, network, storage.

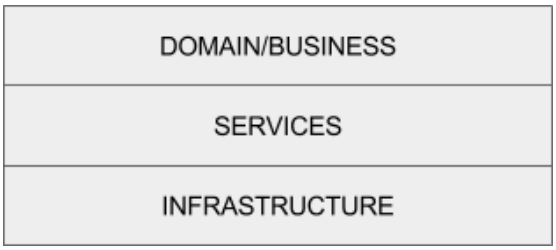

Fig 2: *TM architecture context.*

QUALITY ATTRIBUTES

The main quality that drove the development of the present architecture was the maintainability intended as availability (reliability and recovery), scalability and reproducibility.

This design then focuses on the following quality attributes:

- Maintainability: By creating an uniformization layer above the hardware and by providing a set of descriptive creation of the TM Apps running times, it becomes significantly simpler to maintain the service.
- Reusability: The three tier system for the description of the Virtualization Service allows for each building block to be reused in creating more complex ones.
- Availability: By defining the priority of the processes in the interface therefore allowing the Virtual Machine Service to manage the allocation of resources, we are able to ensure high hardware fault tolerance and high concurrent availability of resources.
- Performance: The Virtual Machine Service automatic allocation or resources according to their priority will ensure that the available resources are maximized.
- Reliability: Having a fault tolerant system by containerizing all its components makes it capable of redirecting computing resources upon their failure to the available ones automatically greatly increases reliability.

Downloaded From: https://www.spiedigitallibrary.org/conference-proceedings-of-spie on 13 Oct 2020
Terms of Use: https://www.spiedigitallibrary.org/terms-of-use

- Scalability: The system is built to be scalable by design and from the ground up. Increasing or decreasing the available computing resources will automatically be managed by the virtual Machine Service and immediately put to use as soon as they are added to the existing computing resources pool.
- Reproducibility: Using Virtualized containers fully defined from the ground up and having a system, with all its components defined from the hardware level up to the product execution level, permits to know exactly the conditions (software versions, algorithms) on which any of the TM Apps had run in order to access the final result.

RUNTIME VIEW

SKA1 TM LINFRA will function as a block managing the available hardware through the use of virtualization resources (vResources), namely the orchestrator; networking modules; storage; and monitoring modules, assigned to a specific configuration of an hardware entity. Every entity has associated a template that is a description of the set of instances (servers, VMs or containers) with an SLA (Service Level Agreement), user ACLs (Access Control Lists) and network ACLs needed by the entity.

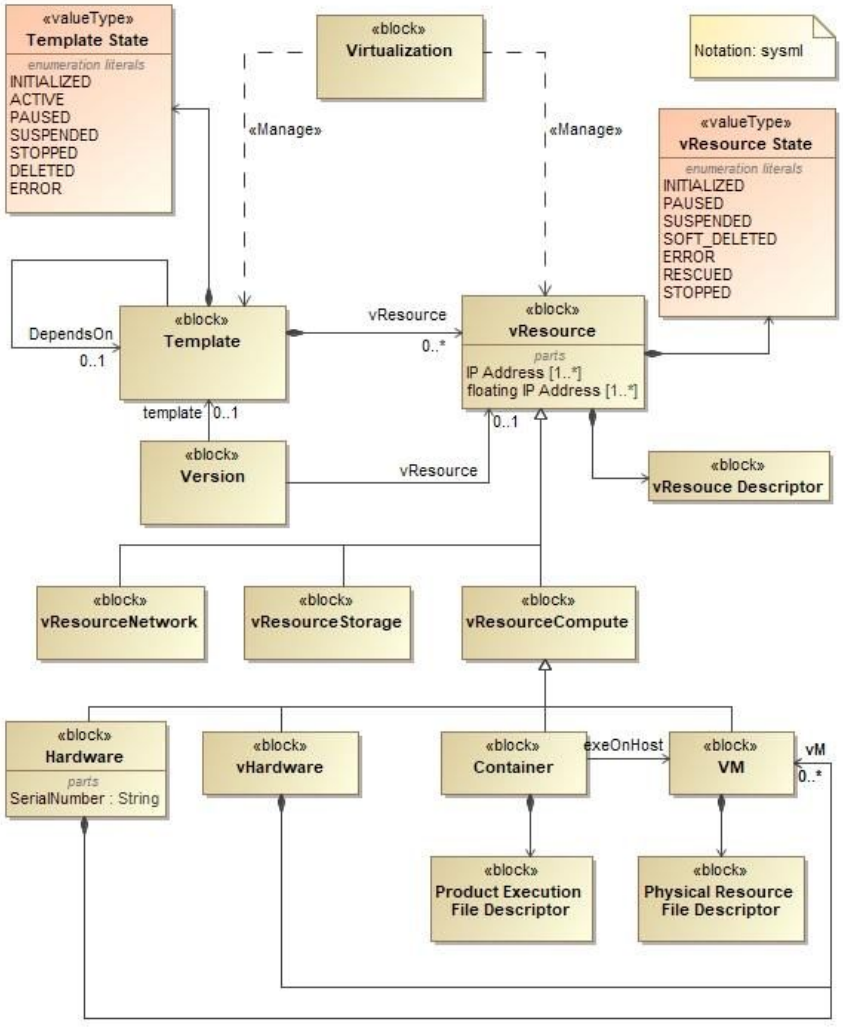

Fig 3: *Virtualization data model within the context of TM.*

Downloaded From: https://www.spiedigitallibrary.org/conference-proceedings-of-spie on 13 Oct 2020
Terms of Use: https://www.spiedigitallibrary.org/terms-of-use

A vResource is composed by:

- vResourceCompute: Virtualized computational resources (Hardware, vHardware, Container or Virtual Machine);
- vResourceNetwork: Virtual network for communication between modules of an App or between different Apps;
- vResourceStorage: Virtual access to storage for an App and its modules.

The Template and the vResource blocks have a state associated that is collected and managed by the Software System Monitor (SSM). Figure 3 shows the data model for the use of the virtualization system made for TM.

CONCLUSION

The SKA, due to its massive scale, has extremely demanding requirements set in place at various levels in its infrastructure. These requirements mainly focus on three key points: availability, scalability and reproducibility.

Implementing from the start of the project an architecture built upon virtualization technologies and expressing a set of requirements for the Apps enforcing: modularization, state based logic and parallelization; it enables to build a virtualization implementation, relying upon a three level access and control system, that will be capable of optimally managing the computing resources available in order to deliver the final products for the TM. It also sets a reliable and similar scenario for other SKA elements like SDP and future post-SDP e-infrastructures federations like AENEAS. More so, this same architecture when implemented in a system with redundant hardware resources, will ensure an high availability of the services while at the same time fully utilizing whatever redundant and available computing resources do exist at a given time.

Finally, since all the various components of the system will need, at any given time, to be described through a set of configuration files that define their building blocks, resource access and computing steps needed to obtain any product, the system will be, by design, reproducible at any given time in the future. With this attribute, any data or scientific deliverable created by the system, could, in principle, always have associated the full history - from source data gathering to the final result - on how that deliverable came to exist.

ACKNOWLEDGEMENTS

This research was supported by the project Enabling Green E-science for the SKA Research Infrastructure (ENGAGE SKA), reference POCI-01-0145-FEDER-022217, funded by COMPETE 2020 and FCT, Portugal.

This work has been made possible thanks to the financial support by the Italian Government (MEF - Ministero dell'Economia e delle Finanze, MIUR - Ministero dell'Istruzione, dell'Università e della Ricerca). BM, DB and JPB acknowledge support from the European Commision through H2020 AENEAS project, Grant 731016.
We thank fruitful discussions on reproducibility with Lourdes Verdes-Montenegro and Suzana Sanchez from IAA-CSIC, Spain.

REFERENCES

[1] The Zettabyte Era: Trends and Analysis, Cisco, Document ID:1465272001812119 (June 2017)
[2] TOP500 project, https://top500.org
[3] Registration Statement, Facebook Inc., https://www.sec.gov/Archives/edgar/data/1326801/000119312512034517/d287954ds1.htm (February 2012)
[4] OpenStack, https://www.openstack.org

Downloaded From: https://www.spiedigitallibrary.org/conference-proceedings-of-spie on 13 Oct 2020
Terms of Use: https://www.spiedigitallibrary.org/terms-of-use

[5] Swaminathan Natarajan, Domingos Barbosa, Joao Paulo Barraca, Alan Bridger, Subhrojyoti Roy Choudhury, Matteo Di Carlo, Mauro Dolci, Yashwant Gupta, Juan Guzman, Lize Van den Heever, Gerhard LeRoux, Mark Nicol, Mangesh Patil, Riccardo Smareglia, Paul Swart, Roger Thompson, Sonja Vrcic, Stewart Williams, "SKA Telescope Manager (TM): Status and Architecture Overview," Proc. of SPIE 9913, 991302 (2016)
[6] SKA Data Processor Architecture, P. Alexander, V. Allan, R. Bolton, P. C. Broekema, G. van Diepen, S. Gounden, Á. Mika, R. Nijboer, B. Nikolic, S. Ratcliffe, A. Scaife, R. Simmonds, J. Taylor, A. Wicenec, SKA-TEL-SDP-0000013,rev 2016-07-29
[7] AENEAS, https://www.aeneas2020.eu/
[8] Baker, Monya, "1,500 scientists lift the lid on reproducibility" Nature 533, 452–454 (2016)
[9] TIA STANDARD, Telecommunications Infrastructure Standard for Data Centers, https://manuais.iessanclemente.net/images/9/9f/Tia942.pdf
[10] J.B. Morgado, D. Maia, L. Lanzerotti, P. Gonçalves and J. Douglas Patterson, "The low energy magnetic spectrometer on Ulysses and ACE response to near relativistic protons," Astronomy & Astrophysics 577, 61 (2015)
[11] J. E. Ruiz, J. Garrido, J.D. Santander-Vela, S. Sánchez-Expósito, L.Verdes-Montenegro, “AstroTaverna - Building workflows with Virtual Observatory services”, Astronomy and Computing, Vols 7-8, pp 3-11 (2014). DOI: 10.1016/j.ascom.2014.09.002
[12] Hettne, K., M, Dharuri, H., Zhao, J.; Wolstencroft, K., Belhajjame, K.; Soiland-Reyes, S., Mina, E.; Thompson, M.; Cruickshank, D.; Verdes-Montenegro, L., Garrido, J., de Roure, D., Corcho, O., Klyne, G., van Schouwen, R., 't Hoen, P. A. C., Bechhofer, S., Goble, C., Roos, M., “Structuring research methods and data with the Research Object model: genomics workflows as a case study”, Journal of Biomedical Semantics 2014, 5:41. DOI: 10.1186/2041-1480-5-41

Downloaded From: https://www.spiedigitallibrary.org/conference-proceedings-of-spie on 13 Oct 2020
Terms of Use: https://www.spiedigitallibrary.org/terms-of-use