SDP Memo 101: Protocol for Downselecting Execution Frameworks during Construction

Document number……SKA-TEL-SDP-0000202
Document Type……MEMO
Revision……01
Author……Verity Allan
Release Date……2019-03-29
Document Classification…… Unrestricted

Lead Author	Designation	Affiliation
Verity Allan	SDP Configuration Manager	University of Cambridge
Signature & Date:	Verity Allan (Mar 29, 2019)	

SDP Memo Disclaimer

The SDP memos are designed to allow the quick recording of investigations and research done by members of the SDP. They are also designed to raise questions about parts of the SDP design or SDP process. The contents of a memo may be the opinion of the author, not the whole of the SDP.

Table of Contents

Introduction

This memo is intended to provide more detailed guidance about applying set-based design and other SAFe (Scaled Agile Framework) techniques for downselecting Execution Frameworks [RD01]. The techniques described here will be applicable for other prototyping areas where a final technology choice has not yet been made. This memo assumes familiarity with the SAFe processes, the Execution Framework component of the Science Data Processor of the SKA, and the SDP Construction Plan [RD01, RD03, RD04] . This memo was written in order to address SDPCDR-100.

References

Reference Number	Reference
RD01	https://www.scaledagileframework.com/
RD02	SKA-TEL-SDP-0000047, Rev 05, SDP Construction and Verification Plan
RD03	SKA-TEL-SDP-0000117, Rev 02, SDP Execution Frameworks Report
RD04	SKA-TEL-SDP-0000013, Rev 07, SDP Architecture

Execution Framework Downselect Context

As noted in RD03, the Execution Framework component of the Science Data Processor remains a relatively high-risk component; the risk will be mitigated by using set-based design. Set-based design here means that multiple technologies can be developed during Bridging and Construction, with a downselect only being made once there is sufficient confidence in the technology.

The Execution Framework module will be built in the SAFe context, using the techniques and milestones described in Construction plan [RD02]. However, there is a stronger than usual need to track new technologies, because of the exceptional performance demands of the SKA, and the rapidly evolving nature of the field. As an example of the latter, five years ago we were contemplating investigating Hadoop, then the leading parallel data processing framework. Three years ago, we started investigating Spark. Now, we are unlikely to continue working with Spark, and our primary focus is on Dask.

SAFe provides a way to investigate any new technologies, via “spikes”, where one can hold a workshop or timebox an interval to research options. I would suggest that at least one such research-oriented spike should be held for every two Program Increments, unless we are completely confident in our technology choice. This allows us to keep exploring the technology landscape. Even if we have made a choice, we may wish to consider continuing with such research spikes on an approximately yearly cadence, as the technology is rapidly evolving, and we may make considerable gains by switching.

The demo session at the end of a sprint is a good chance to expose stakeholders to the results of the research spike, and provides the opportunity to make decisions about whether further research or development is warranted, either by scheduling further spikes or by adding items to the team’s backlog. The Program Increment sync point then allows consideration of the risks and benefits of pursuing particular technologies in the larger SKA context. Similarly, decisions can then be made about adding new items to the program backlog.

The modularity of the domain-specific software of the Science Data Processor has been designed to make it easy to write new pipelines, implement new algorithms, and feasible to implement new Execution Frameworks if the gain in performance, maintainability and/or reliability is sufficient. Therefore, we can relatively easily afford to carry though several technologies or switch technologies relatively late, because the overhead of switching is not excessive. This has been partially validated by the work on the Algorithm Reference Library.

This being said, at AA3, we would anticipate having selected a streaming framework, and one or two batch processing frameworks, which would then be targeted for production readiness for AA4. (This allows us to pick batch Execution Frameworks that give adequate performance for the hardest cases.) We may be able to run with just one or two frameworks at this stage; this would be evaluated in our performance milestones. However, we would still be able to swap or add in another technology should this give us sufficient value.

Process for Downselecting Execution Frameworks

First of all, I am assuming that there is an Agile team, either dedicated to Execution Frameworks, or which contains the expertise to work on Execution Frameworks. This Agile team should be working from a backlog that has been populated from the SDP Critical Design Review panel comments (OARs) and from the Execution Frameworks Prototyping Report [RD03].

Obviously, this team will be working on the usual SAFe cadences, with sprints and program increments, as noted above. The SDP Construction plan [RD02] contains performance milestones, which allow the testing of the Execution Framework (and other critical components) at scale. The results of those performance tests can be used to re-evaluate the set of technologies being developed. Because we are using the SAFe process, additional performance tests can be arranged whenever this is useful to the project, or existing milestones can be brought forward if appropriate.

Via the performance milestones and standard SAFe processes, we will be conducting realistic tests of Execution Frameworks. If, early on, one framework is performing very poorly, we can drop it from the list for consideration. Similarly, if one framework is satisfying our requirements well, we can drop the others. If there are no clear leaders, we may keep more technologies in active development. We should, of course, continue to track developments in previously-dropped frameworks, as they may evolve in such a way as to overcome their previous deficiencies. Again, regular research “spikes” can help us to stay up-to-date with the frameworks used for distributed parallel processing.

Criteria for Execution Framework Evaluation

While the Construction plan and SAFe provide us with many points at which we can evaluate the performance and development of Execution Frameworks, we also need to set out some criteria for doing this evaluation. These criteria go beyond the strict performance metrics, and consider the wider context of SKA development.

Criteria for evaluation:

- Performance on a given pipeline under given conditions
    - Performance across multiple pipelines may be evaluated
    - Reliability:
        - how often does a running pipeline need attention?
        - how often is a pipeline crashing or aborted?
        - How does a pipeline deal with stragglers?
    - How good are the logs? Can we extract crash reports or pipeline heuristics?
- Person hours to develop initial pipeline
- Person hours to tune pipeline
- Programmer evaluation of ease of developing and tuning pipeline

- Programming language of Execution Framework & ease of recruiting staff who can work on it
- Execution Framework developer community - are SKA the only or primary users?
- Execution Framework maturity: how well-supported is the framework?

The SKA Software team may wish to consider formal weighting schemes to balance these competing interests. The performance aspects are easy to capture and measure; the sociological aspects are harder, but may be more important to the organisation in the long run, as they more obviously impact the ease of creating and managing pipelines, and the risks associated with running the SKA Observatory.

Frameworks to Consider During Bridging

There are three primary frameworks to take into construction: Dask, DALiuGE and MPI. These three frameworks are all programmable in Python. MPI is a well-established technology; Dask is emerging as the Python parallel data processing framework of choice; DALiuGE is specifically designed for astronomy data processing at scale. We may wish to continue examining StarPU, but that partly depends on the team composition early in Bridging. If we cannot construct a team with suitable expertise, I would recommend against continuing with StarPU, as it's primarily designed for on-node distribution, and the programming model is somewhat tricky to get to grips with [RD03].

During Bridging, we also need to consider the streaming data case. We can either look at prototyping with Apache Storm, or try prototyping with one of the above-mentioned frameworks we are prototyping for batch processing.

This candidate list can and should be re-evaluated through Bridging and early Construction. For example, I did not include Tensorflow among my initial list of frameworks. However, I would recommend keeping an eye on the development of Tensorflow during Bridging and Early Construction, by means of analysing it during spikes. If we can reduce the science data processing to a Tensorflow problem, then Tensorflow is a compelling Execution Framework candidate.

Conclusion

Whilst I have described a process for downselecting Execution Frameworks, this can be extrapolated to other technologies. The prototyping reports for other modules contain information about areas of higher risk that can be used to populate relevant backlog tasks for Agile teams. SAFe gives us the way to then work on the backlog, and the specific performance milestones described in the Construction plan suggesting the points at which decisions are likely to be needed.

The basic technique of incorporating research spikes can be used to evaluate new technologies or developments in many of the SKA software areas; decisions can then be taken about whether it is fruitful to further pursue them.