

C.3.2.5 Workload Characterisation Framework

Summary

JIRA ticket	<div>ARCHIVE-1516 - C</div> <div>.3.2.5 Workload Characterisation Framework</div> <div>TO DO</div>
Owner	Rahim Lakhoo
Developers	Chen Wu
TRL Summary	

Links to Documents

(Insert links to any relevant Google Docs, SysML models, items in Configuration management tools etc.)

Table of Contents

- [Summary](#)
- [Links to Documents](#)
- [Table of Contents](#)
- [Page Tree](#)
- [Tickets](#)
- [1.0 Introduction](#)
 - [1.1 Background and Strategic Fit](#)
 - [1.2 Context](#)
 - [1.3 Behaviour](#)
 - [1.4 Interfaces](#)
- [2.0 Requirements](#)
 - [Profiling Requirements](#)
 - [Related Requirements](#)
 - [2.2 Functional Requirements](#)
 - [Workload Capture](#)
 - [Workload Replayer](#)
 - [Workload Configurator](#)
 - [Benchmark Execution](#)
 - [Time Series DB](#)
 - [Query, Visualisation and Modelling](#)
 - [Workload Characterisation Engine](#)
 - [Workload Performance Analysis](#)
 - [Related Functional Requirements](#)
 - [2.3 Verification Requirements](#)
 - [Related Requirements](#)
 - [2.4 Cost](#)
 - [2.5 Schedule](#)
- [3.0 Select Candidate Solution](#)
 - [3.1 Architectural Drivers](#)
 - [3.2 Candidate Solutions](#)
 - [Workload Capture](#)
 - [Tracing Format](#)
 - [Query, Visualisation and Modelling](#)
 - [3.3 Concept Selection Table](#)
 - [3.4 Risk Assessment Table](#)
 - [3.5 Select Preferred Option\(s\)](#)
- [4.0 Critical Technology Element Selection](#)
- [5.0 Technology Readiness Level Assessment](#)
- [6.0 List of TBDs](#)
- [7.0 Prioritised Prototyping Test plan](#)
- (please link to JIRA tickets where possible)
- [8.0 Not Doing/Not Considered](#)

Page Tree

Tickets

(insert JIRA filter here)

These headings below are a guide. See [Prototype Selection Process Template](#), the [Reference Guide: Prototype Selection Process](#), and the worked example: [C.1.1.1.9 Fast Buffer](#). There is also a Google docs version of the template in SDP Informal Document Sharing.

1.0 Introduction

1.1 Background and Strategic Fit

The goal of Workload Characterisation Framework (WCF) is three-fold.

First, it aims to provide the scheduler (or the “mapper”) with the essential resource usage estimates (aka workload characteristics) for each *Processing Component* (i.e. Application DROP in the logical graph) in order to make optimal (cost-efficient) scheduling decision. It aims to facilitate the mapping from logical graph to the physical graph, thus determining which processing components are allocated to which compute island. The workload characteristics are directly related to the *Resource Usage* (including compute, I/O, and power) for each processing component, and the estimates will be provided in the form of a set of performance metrics with some basic statistics. For example, if an processing component exhibits a strong random-writes-dominated I/O access pattern, it is desirable that the scheduler allocate to this processing component adequate SDD-backed storage I/O buffer.

Second, it aims to assist SDP developers / deployers of Pipeline *Processing Components* to obtain a quantitative understanding of the compute and data access behaviours exhibited by various radio astronomy data processing pipelines and algorithms.

Third, it aims to enable SDP platform vendors to benchmark their platforms and derive standardised performance statistics, which can be used to either optimise the platforms or compare with one another.

1.2 Context

Figure 1. The box "Component Profiling Details" are the output of the Workload Characterisation framework, which constitute the essential inputs for scheduler to produce the physical graph.

? Unknown Attachment

Figure 2. In addition to profiling details, the Scheduler also needs "Observation parameters" to produce the physical graph. The observation parameters provide concrete information on datasets such as data type, volumes, deadlines, sub-array information, which is essential for the scheduler to determine the data distribution / partition before the mapping from logical graph to physical graph can take place. Therefore, the scheduler needs both WCF for each *Processing Component* and Observation parameters to complete the mapping procedure from logical graph to physical graphs.

1.3 Behaviour

The LMC Data Flow Model describes the workload characteristics of Processing Components. The information is needed to allocate compute resources and is manifested in the Physical Deployment Graph at runtime. See [Figure 3] for the profiling and benchmarking framework that performs the relevant measurements on the target platform. It is implemented during the pre-construction phase and helps with the ongoing definition of the LMC Data Flow Model itself. During the operational phase Processing Components will be characterised once for a given hardware configuration and whenever the setup changes. The framework derives standardised data sets which are also useful for optimisation and comparative analysis (including procurement, if desired). Recognising the complexity of the processing system and domain specific workloads it is designed to collect data on many levels (PDR.02.02 DATA SUB-ELEMENT DESIGN REPORT).

Figure 3. Workload Characterisation and Profiling Framework.

1.4 Interfaces

2.0 Requirements

Profiling Requirements

- WCF_REQ-1: O/S level application profiling - ability to profile applications without specific instrumentation. Profiling should cover at least:
 - CPU Usage - load (%), FLOPS, FLOPS/Watt, Cache information, instructions, power, cycles.
 - I/O subsystem usage - load (%), Disk IOPs and throughput, Disk capacity, Network throughput, traffic profile and packet loss, PCIe transfer statistics.
 - Memory usage - bandwidth and usage statistics, locality and hardware counter statistics.
- WCF_REQ-2: User/application level profiling - The aim is to provide non-intrusive profiling with a low over head (<10%), with options to provide further detailed analysis when needed. A more detailed breakdown if needed will be likely at the cost of performance of the application being profiled.
- WCF_REQ-3: O/S or profiler hooks should be available to allow applications to provide additional explicit profiling data in addition to the implicit data.
- WCF_REQ-4: Power consumption profiling should ideally be available, both at a component, system and application levels.
- WCF_REQ-5: To provide information that can be used for quality assurance and verification functionality.
- WCF_REQ-6: To facilitate the metrics suitable for aiding DFM functionality, graphing and resource estimation/allocation.
- WCF_REQ-7: Ability to monitor standard O/S metrics - e.g. disk use, memory use, CPU utilisation, network utilisation etc.
- WCF_REQ-8: Logging of all monitoring data to enable identification of trends as well as provide historical data for comparison when making changes to systems and applications.
- WCF_REQ-9: Hooks to allow applications to expose application-specific monitoring information to the common monitoring system as appropriate.
- WCF_REQ-10: Ability to expose monitoring data via standardised interface - e.g. SNMP
- WCF_REQ-11: Ability to define thresholds for monitoring metrics which will trigger certain actions, or developer notifications, eg alarms, shutdowns etc.
- WCF_REQ-12: Be able to capture and replay characterised workloads/system interactions in a repeatable manner.
- WCF_REQ-13: Allow for workload characterisations to be retrieved and compared.
- WCF_REQ-14: Provide developers/vendors/telescope operators with an interface for visualising and viewing profiling information.
- WCF_REQ-15: Provide the ability to run benchmark applications for specific micro-benchmarking of system aspects, in a comparable and repeatable manner.

Related Requirements

A list of related requirements for this product tree item.

Requirement	Description	Related	Comments
SDP_REQ-4 Capability availability	The SDP shall on request from the TM provide availability information for the requested capability	SDP_REQ-14 SDP_REQ-15 SDP_REQ-16 SKA1-SYS_REQ-2133	WCF_REQ-1, WCF_REQ-2, WCF_REQ-3, WCF_REQ-4, WCF_REQ-5, WCF_REQ-6, WCF_REQ-7, WCF_REQ-8, WCF_REQ-11, WCF_REQ-14
SDP_REQ-14 Availability - Compute	The Master Controller shall be able to query the Scheduler in order to determine the availability of compute resources (hardware and software) for a specific capability.	SDP_REQ-4	WCF_REQ-1, WCF_REQ-2, WCF_REQ-3, WCF_REQ-4, WCF_REQ-5, WCF_REQ-6, WCF_REQ-7, WCF_REQ-8, WCF_REQ-11

SDP_REQ-15 Availability - Data Layer	The Master Controller shall be able to query the Data Layer, via the Data Flow Manager, in order to determine the availability of data layer resources (transport, buffer and archive) for a specific capability.	SDP_REQ-4	WCF_REQ-1, WCF_REQ-2, WCF_REQ-3, WCF_REQ-4, WCF_REQ-5, WCF_REQ-6, WCF_REQ-7, WCF_REQ-8, WCF_REQ-11
SDP_REQ-16 Availability - Delivery	The Master Controller shall be able to query the Data Delivery subsystem in order to determine the availability of delivery resources for a specific capability.	SDP_REQ-4	WCF_REQ-1, WCF_REQ-2, WCF_REQ-3, WCF_REQ-4, WCF_REQ-5, WCF_REQ-6, WCF_REQ-7, WCF_REQ-8, WCF_REQ-11
SDP_REQ-18 Scheduling - Data Layer	The Master Controller shall be able to instruct the Data Layer, via the Data Flow Manager, to schedule on observation to commence at a particular time with a particular capability.		WCF_REQ-1, WCF_REQ-2, WCF_REQ-3, WCF_REQ-4, WCF_REQ-5, WCF_REQ-6, WCF_REQ-7, WCF_REQ-8, WCF_REQ-11
SKA1-SYS_REQ-2133 Mode transition	The switching time between telescope operating modes shall take less than 30 seconds (not including antenna slewing time).	SDP_REQ-4	WCF_REQ-1, WCF_REQ-2, WCF_REQ-3, WCF_REQ-4, WCF_REQ-5, WCF_REQ-6, WCF_REQ-7, WCF_REQ-8, WCF_REQ-11 Resource estimations must occur in less than 30 seconds.
SKA1-SYS_REQ-2280 System status	The system shall extract information about the current condition of the system from the science and calibration data streams, and log this information along with other relevant system and environmental status information. Based on this information, it shall be possible to monitor, save, and analyse the technical performance of the system	SDP_REQ-593 Status Logging SDP_REQ-592 Status Monitoring SDP_REQ-594 Status Reporting	WCF_REQ-1, WCF_REQ-3, WCF_REQ-4, WCF_REQ-7, WCF_REQ-8, WCF_REQ-9
SDP_REQ-593 Status Logging	The SDP shall log received status information.	SKA1-SYS_REQ-2280	WCF_REQ-8
SDP_REQ-592 Status Monitoring	The SDP shall query status of science and calibration data streams.	SKA1-SYS_REQ-2280	WCF_REQ-7, WCF_REQ-8
SDP_REQ-594 Status Reporting	The SDP shall use received status information to report on technical performance.	SKA1-SYS_REQ-2280	WCF_REQ-5
SDP_REQ-377 System health monitoring	The SDP compute system platform management function shall collect and expose system health information to be made available to the LMC.	SDP_REQ-34 SDP_REQ-597	WCF_REQ-1, WCF_REQ-3, WCF_REQ-4, WCF_REQ-7, WCF_REQ-8, WCF_REQ-9
SDP_REQ-378 Deployment system	The SDP compute system platform management function shall contain a deployment service.		WCF_REQ-1, WCF_REQ-2, WCF_REQ-7, WCF_REQ-8
SDP_REQ-379 Scheduler	The SDP compute system platform management function shall provide a scheduler for the compute system resources.	SDP_REQ-378	WCF_REQ-6
SDP_REQ-382 Component system consistency	The SDP compute system deployment system shall ensure all component systems are in a consistent, verifiable, documented, and reproducible state at all time.		WCF_REQ-5, W CF_REQ-12, WCF_REQ-13, WCF_REQ-15

SDP_REQ-34 Self testing	The SDP shall provide an automated self-test function that will generate a report on the health of the SDP system as a whole, including malfunction and out of tolerance operation.	SDP_REQ-377	WCF_REQ-5, WCF_REQ-12, WCF_REQ-13, WCF_REQ-14, WCF_REQ-15, WCF_REQ-1, WCF_REQ-2, WCF_REQ-4, WCF_REQ-7, WCF_REQ-8
SDP_REQ-597 Component system state information	Component system state information of the SDP compute system shall be made available as part of the observation meta data.	SDP_REQ-377	WCF_REQ-1, WCF_REQ-2, WCF_REQ-3, WCF_REQ-5, WCF_REQ-6, WCF_REQ-7, WCF_REQ-8, WCF_REQ-14
SYS_REQ-2546 Continuous performance monitoring	Where possible, the system shall be designed to provide continuous performance monitoring.	SKA1-SYS_REQ-2816 SDP_REQ-265 SDP_REQ-287	WCF_REQ-1, WCF_REQ-2, WCF_REQ-3, WCF_REQ-4, WCF_REQ-6, WCF_REQ-7, WCF_REQ-8
SDP_REQ-265 Continuous Performance Monitoring Metrics	Performance monitoring shall be compliant with the capabilities specified in SKA-TEL.SDP.SE-TEL.TM.SE-ICD-001. SDP_REQ-266 Status Information	SYS_REQ-2546	WCF_REQ-1, WCF_REQ-2, WCF_REQ-3, WCF_REQ-6, WCF_REQ-7
SDP_REQ-287 Continuous performance monitoring	Performance monitoring shall be compliant with the capabilities specified in SKA-TEL.SDP.SE-TEL.TM.SE-ICD-001. SKA1-SYS_REQ-2546 Continuous performance monitoring	SYS_REQ-2546	WCF_REQ-1, WCF_REQ-2, WCF_REQ-3, WCF_REQ-6, WCF_REQ-7, WCF_REQ-8, WCF_REQ-11
SKA1-SYS_REQ-2816 Design for testability	Designs shall include an assessment of testability in accordance with BS EN IEC 60706-5	SYS_REQ-2546	WCF_REQ-5, WCF_REQ-12, WCF_REQ-13, WCF_REQ-14, WCF_REQ-15
SYS_REQ-2552 Malfunction detection	All equipment malfunction shall be detected at the system level.	SKA1-SYS_REQ-2816	WCF_REQ-8, WCF_REQ-11
SYS_REQ-2744 Quality assessment	Quality assessment shall be based on the comparison of a Performance Assessment and a Performance Goal.	ConOps Section 2.1	WCF_REQ-5, WCF_REQ-12, WCF_REQ-13, WCF_REQ-14, WCF_REQ-15
SDP_REQ-26 Resource sharing	The Master Controller shall ensure that any shared resources between SDP instances on different telescopes do not compromise the ability of the instance to act concurrently and independently.	SKA1-SYS_REQ-2126 SDP_REQ-25 SDP_REQ-26	WCF_REQ-1, WCF_REQ-2, WCF_REQ-3, WCF_REQ-5, WCF_REQ-6, WCF_REQ-7, WCF_REQ-14
SKA1-SYS_REQ-2126 Simultaneous operation of telescopes	Simultaneous operation of telescopes. All three telescopes shall be capable of operating concurrently and independently.	SDP_REQ-26	WCF_REQ-1, WCF_REQ-2, WCF_REQ-3, WCF_REQ-5, WCF_REQ-6, WCF_REQ-7, WCF_REQ-14

SDP_REQ-577 Decoupling real-time Ops	Access methods to the science data archive shall not impact the performance of telescope operations such as science data product ingest.		WCF_REQ-1, WCF_REQ-2, WCF_REQ-3, WCF_REQ-5, WCF_REQ-6, WCF_REQ-7, WCF_REQ-14
SDP_REQ-25 Independent Operations	LMC functions should be able to operate concurrently and with performance which is independent of other LMC instances	SDP_REQ-26	WCF_REQ-1, WCF_REQ-2, WCF_REQ-3, WCF_REQ-5, WCF_REQ-6, WCF_REQ-7, WCF_REQ-14
SDP_REQ-272 Performance Feed	From PDR document for DATA (SKA-TEL-SDP-0000023, pg 30) - can't be found in innoslate.	LMC and DATA Layer.	TBD
SDP_REQ-239 Power Monitoring	The SDP shall have the ability to report low latency power consumption information on a per component (i.e. server, network switch, storage pod, etc.) basis.		WCF_REQ-4
SDP_REQ-245 Power Monitoring	There shall be a mechanism to make available low latency power consumption information on a per component (i.e. server, network switch, storage pod, etc.) to SDP LMC.		WCF_REQ-4
SDP_REQ-247 Data Layer Management	There shall be a system of software systems that manages the data flow from the reception of raw data from CSP to the delivery of science products.		WCF_REQ-1, WCF_REQ-7, WCF_REQ-8,
SDP_REQ-376 Platform management interface to LMC	The SDP compute system platform management function shall have an interface to the SDP LMC system.		WCF_REQ-1, WCF_REQ-3, WCF_REQ-4, WCF_REQ-6, WCF_REQ-7, WCF_REQ-8, WCF_REQ-9, WCF_REQ-11
SDP_REQ-627 Hardware availability	The SDP shall determine the availability of hardware resources needed for the construction of data flow graphs.		WCF_REQ-1, WCF_REQ-2, WCF_REQ-3, WCF_REQ-4, WCF_REQ-5, WCF_REQ-6, WCF_REQ-7, WCF_REQ-8, WCF_REQ-11
SDP_REQ-267 Availability Metric	The data layer manager's subsystems shall have a defined metric for deriving the overall availability of a given runtime configuration.		WCF_REQ-1, WCF_REQ-2, WCF_REQ-3, WCF_REQ-4, WCF_REQ-7, WCF_REQ-8, WCF_REQ-9, WCF_REQ-11
SDP_REQ-269 Storage Technology Transition	A strategy for phasing in new storage technology, e.g. new media types, and phasing out obsolete storage technology shall be part of the data migration plan.		
SDP_REQ-25 Independent Operations	LMC functions should be able to operate concurrently and with performance which is independent of other LMC instances		
SDP_REQ-251 Concurrency	The data layer manager of each SKA telescope and all sub-arrays shall be capable of operating concurrently and independently. It shall be possible to run additional instances of the data layer manager for other purposes such as: commissioning, maintenance and simulation in parallel.		
SDP_REQ-252 Concurrent Workflows	The data layer manager shall concurrently generate data products for multiple observing programs. It shall also support a single observing program concurrently generating multiple data products as well as a mix of both scenarios.		
SDP_REQ-253 Workload Balance	The data layer shall provide a load balancing optimisation of the data locality based on the specific performance and I/O work load at runtime.		

SDP_REQ-255 Tracing Data	It shall be possible to trace each data product in the archive back to a scheduling block and an observing program. Reversely, it shall be possible to either search by observing program or scheduling block and to subsequently retrieve all or part of the associated archived data. This includes relevant logging and monitoring information as well as quality assessment data collected during the observations and the standard processing.		
SDP_REQ-375 SDP platform management	The SDP shall provide a dedicated platform management function.		
SDP_REQ-386 Continuum imaging pipeline	The SDP shall provide a Continuum Imaging pipeline that constructs wide-band images. Polarisation shall be available if requested or necessary for calibration or quality assurance.		
SDP_REQ-387 Spectral line emission pipeline	The SDP shall provide a Spectral Line Emission pipeline that constructs channel cubes of spectral line emission either with continuum emission remaining or with continuum emission removed.		
SDP_REQ-388 Spectral line absorption pipeline	The SDP shall provide a Spectral Line Absorption pipeline that constructs channel cubes of spectral line absorption with continuum sources removed.		
SDP_REQ-389 Slow transient pipeline	The SDP shall provide a Slow Transient imaging pipeline that shall be capable of constructing a continuum image after a GSM has been subtracted for every correlator integration time or slower, searching for transient sources, and producing a time-ordered catalog.		
SDP_REQ-41 Performance assessment	The User Interface shall allow the configuration and evaluation of a performance assessment based on data provided by internal SDP metrics.		
SDP_REQ-417 Calibration pipeline.	SDP shall have a Calibration pipeline that derives current telescope parameters using a recent observation and either a known or the most recent Global Sky Model.		
SDP_REQ-42 Performance goals	The User Interface shall allow the configuration of performance goals that describe the desired outcome of a particular performance assessment.		
SDP_REQ-43 Quality assessment calculation	The User Interface shall allow the calculation of a quality assessment based on a comparison of a particular performance assessment and its associated performance goal.		
SDP_REQ-470 Receive Data	The SDP shall receive the data packets from CSP in compliance with the CSP-SDP ICD.		
SDP_REQ-527 Pulsar Search Data Input	SDP shall be capable of receiving pulsar periodicity search data in accordance with the SKA-TEL.SDP.SE-TEL.CSP.SE-ICD-001 Interface Control Document.		
SDP_REQ-539 Non-imaging Transient Input	SDP shall be capable of receiving non-imaging transient search data in accordance with the SKA-TEL.SDP.SE-TEL.CSP.SE-ICD-001 Interface Control Document.		
SDP_REQ-599 Data Service Layer	The Data Service Layer ensures that data access shall be independent of the physical schema of the underlying compute infrastructure.		
SDP_REQ-624 Data Flow Graphs	The SDP shall be responsible for the identification, creation and dissemination of data flow graphs		WCF_REQ-1, WCF_REQ-2, WCF_REQ-3, WCF_REQ-4, WCF_REQ-5, WCF_REQ-6, WCF_REQ-7, WCF_REQ-8, WCF_REQ-11
SDP_REQ-626 Component Profiles	The SDP shall maintain profiles of pipeline components needed for the construction of data flow graphs.		WCF_REQ-1, WCF_REQ-2, WCF_REQ-3, WCF_REQ-4, WCF_REQ-5, WCF_REQ-6, WCF_REQ-7, WCF_REQ-8, WCF_REQ-11, WCF_REQ-12

2.1 Performance Requirements

In line with SKA1-SYS_REQ-2133 Mode transition, all gathering, reporting, or calculations must be completed within a suitable time frame to

allow mode transition.

To perform SYS_REQ-2546 Continuous Profiling with minimum impact <10%, ideally 1-5%, and basic workload characterisation with minimum impact <1-5%, ideally less.

Continuous Profiling must keep in line with such performance requirements - SDP_REQ-301 SDP ingest data rate, The SDP system shall ingest CSP data at TBD Gbps, while losing less than TBD% (0.01%?) of the data.

The profiling should be able to cope with concurrent activity, with items running in parallel - SDP_REQ-26, SDP_REQ-251 and SDP_REQ-252.

WCF for profiling and analysing metrics should be able to cope with processing pipeline performance - SDP_REQ-386, SDP_REQ-387, SDP_REQ-388, SDP_REQ-389, and SDP_REQ-417.

The performance for Profiling SDP_REQ-470, SDP_REQ-527 and SDP_REQ-539 should have minimal impact on the data rates.

2.2 Functional Requirements

Workload Capture

INPUT - 1. Pipeline components
2. Workload Configurator (C.3.6.1.3)
3. Workload metrics required in logical graph

OUTPUT - 1. Workload traces collected from executing the processing components deployed by the Workload Configurator
2. Application logs collected from AS ABOVE

Workload Replayer

INPUT - 1. Workload traces
2. Replay type (e.g. I/O, compute, or mixed, etc.)
3. Target hardware platforms
4. Replay nobs (optional, e.g. intensity, etc.)

OUTPUT - Replaying of workload traces on the target platform based on replay type and nobs

Workload Configurator

INPUT - 1. Profiling/monitoring hooks provided in COMP
2. Platform environment specification
3. Profiling metrics used in C.3.6.1.1

OUTPUT - 1. Established channels for Workload Capture (C.3.6.1.1) to obtain measurements
2. Deployment of Processing components and Workload Replayer (C.3.6.1.2) on target platform / environment

Benchmark Execution

INPUT - 1. Microbenchmark (e.g. FFT, IOZone, etc.) and parameters OR
2. Workload traces OR
3. Workload characterisation high-level profiles / models, viz the output of C.3.6.2.3 (e.g. [state transition models](#)) OR
4. Manual script (e.g. FIO, [Workload Model Language](#))

OUTPUT - Execution of Workload / Microbenchmark, and Performance Analysis (C.3.6.2.4)

Time Series DB

INPUT - 1. Workload traces
2. Application logs

OUTPUT - 1. Trace storage and retrieval
2. Viewing traces with basic graphing (e.g. zoom-in/out, panning)
3. Correlation between Workload traces and Application logs (optional)

Query, Visualisation and Modelling

INPUT - 1. Descriptive query (basic statistics such as max, min, mean, distribution)
 2. "What-if" query
 (e.g. How power consumption and latency might change if SSD capacity is doubled but CPU frequency is reduced for Workload ABC?)
 3. Visualisation requests (e.g. trace comparison, trace overlay, and trace correlation)
 4. Model development requests (LP)

OUTPUT - 1. Basic statistics
 2. A range of possible scenarios
 3. Trace compared / overlaid / correlated along time axis
 4. Workload characterisation model template defined (LP)

Workload Characterisation Engine

INPUT - Workload traces

OUTPUT - Workload profiles and characterisation models (e.g. metrics filled with values) for LMC / DFM (resource allocation and scheduling)

Workload Performance Analysis

INPUT - 1. Workload traces
 2. Workload metrics required in logical graph

OUTPUT - 1. Report summarised performance against metrics
 2. Performance comparisons (e.g. spider diagrams)

Related Functional Requirements

A list of related functional requirements for this product tree item.

Function	Description	Related
F.1 Continuum Imaging	To make science-ready images cubes with spectral dimension representing a Taylor expansion of continuum spectra, and any alternative representations which are found to be appropriate, or spectral cubes at low spectral resolution	
F.2 Spectral line Imaging	To make science-ready image cubes with high or moderate spectral resolution.	
F.3 Ingest Data	To receive, store or forward data from CSP (and other sources) and prepares data for further processing. (This will satisfy requirements for doing EoR).	
F.4 Real-time Calibration	To calculate calibration parameters in real-time for other elements of the SKA.	
F.6 Science Analysis		
F.7 Imaging Transient Search		
F.8 Non-Imaging Transient Post Processing		
F.9 Pulsar Timing Post Processing		
F.10 Pulsar Search Post Processing	To sieve pulsar candidates to find confirmed pulsars.	
F.11 Update Global Sky Model	To calculate updates to the global sky model.	
F.12.1 Data Layer Management	Manages and executes the logical data flow from CSP to the final delivery	

F.12.1.4.2 Status Reporting	returns processing status	
F15.2 monitoring	Provides monitoring, logging, alarm, and event handling services for exposure upwards to the Telescope Manager, and for internal use within the SDP.	
F.15.1.2.2 Component Profiling	Tools to profile pipeline components to assist with decision making in the graph building process.	
F.15.1.2.3 Hardware Resource Availability	Services to interrogate and report on the current availability of hardware resources needed to construct a physical data flow graph.	
F.15.2.4 Health State	Produces health indications suitable for consumption by the Telescope Manager. Based on information collected by the monitor, logger and event frameworks.	
F.15.1.5.2 Observation Resource Estimation	Used to estimate the resources required to execute a particular observation. Used by TM in the observation planning phase.	
15.2.2 Central Logging	A central (but likely physically distributed) logging capability that allows fine-grained control of logging at a variety of levels within each SDP sub-element.	
F.15.2.1 Monitoring Data Collection	Services to collect, aggregate and store monitoring data from SDP sub-elements	
F.15.2.3.2 Event Notification	Function to provide notification services for all event types.	F.15.2.3.1 Handle Alarms
F.12.1.4.1 Load Measurement	gather application performance data	
F.12.1.4 Monitoring	Access to system monitoring and application performance data	
F.15.1.2 Manage Data Flow Graphs	Manages the identification, creation and deployment of data flow graphs.	
F.15.1.2.1 Manage Data Flow Models	Curation and provision of the underlying data flow models from the data flow model store to the data flow manager as they are needed.	
F.20.1 Platform Management		
F.20.1.2 Deployment		
F.15.3.1 Pipeline Metric Collection	Collects, sanitises and aggregates metrics produced by the pipelines.	
F.15.3.1.1 Ingest Metric Collection	Collects, sanitises and aggregates metrics produced by the ingest pipeline. This is a special case of the general metric collection function, and will require significant differences in terms of volume of data.	
NF.2 Equipment & Component Specifications	SDP equipment and component requirements.	
NF.7 Maintenance, Test & Support	SDP maintenance, test and support requirements.	

2.3 Verification Requirements

Related Requirements

A listed of related versification requirements.

Requirement	Description	Related
VR-3 Verification by Demonstration	An element of verification that involves the actual operation of an item to provide evidence that the required functions are accomplished under specific scenarios. The items may be instrumented and performance monitored.	TC-1 SDP Test Case 1

2.4 Cost

2.5 Schedule

3.0 Select Candidate Solution

3.1 Architectural Drivers

3.2 Candidate Solutions

Workload Capture

Figure 4, shows the current mature *Workload Capture* technologies and the Linux stack layers from which performance metrics (resource utilisation estimates) will be measured. We have tested several of the above tools including KTap, SystemTap, iostat, pidstat, strace, perf, dstat, sar and blktrace. For example, we used both pidstat/iostat and Strace to collect performance metrics for both compute (e.g. CPU and memory usage) and I/O (e.g. I/O operations, throughput, inter-arrival time, etc.). In pidstat/iostat, one can periodically measure a list of process-specific kernel counters available in the Linux proc file system while the processing tasks were running. The sample interval is currently one measurement per second. While this method provides useful measurements on CPU and memory usage, some detailed I/O metrics cannot be directly derived from the proc file system.

.

Figure 4, Current availability for monitoring and profiling the Kernel.

Therefore, one can use the Strace linux system tool — to access more advanced I/O performance indicators, such as whether the I/O requests were sequential or random and the size of each I/O requests issued to the underlying file system. Strace is able to capture all system calls and signals. However, if one is only interested in I/O requests made by the processing components as system calls, one can instruct Strace to only measure four types of system calls — file descriptor related, process management- related (in order to track sub-processes), socket-related, and those which take a file name as an argument such as open, close, read, write, etc. One issue of strace is the tracing overhead associated with frequent context switching, the other is multithread applications and the traceability of block devices. We have observed overheads in excess of 300% (for tens of thousands of system calls per second), other Kernel tracing authors have noted similar issues [Gregg, B. Strace](#)). The overhead substantially prolongs the applications runtime, but for simple I/O access patterns (random and sequential) analysis this is not always an issue. In addition, multithreaded applications are serialised in the trace output gathered and is not a true representation of activities that may have happened concurrently. Many of the filesystem and transport optimisations rely on device-level interactions, such as RDMA and GPUDirect, and Strace cannot access many of these areas for examination. The overheads and loss of detail in Strace's capture has a knock on effect for high-speed/throughput, or high-performance applications, were certain issues or patterns of behaviour only occur at near full-speed operation; other solutions better suited to continuous profiling and monitoring need to be accessing information from the Kernel will less overhead.

For continuous profiling other solutions that provide access to the same, or similar set of systemically or metrics are available. Previous Data Challenge prototyping activities included SystemTap, KTap Blktrace and Perf, using various Kernel probes have shown that the same or similar trace calls can be profiled with a much lower overhead, typically between 2-25% depending on the complexity of the information gathering from within the Kernel. For example, CPU FLOPS calculations and power/energy readings (micro-Joules) from CPU and Memory, have been typically gathered with only a few percent overhead, and without any code changes. Blktrace can show access patterns from the device-level, per CPU core, meaning it can represent concurrent I/O activity and with typically no more than a 5% overhead. Tap, SystemTap and Perf are able to trace the some of the same sys calls as Strace, but also provide FLOPS, Cache, filesystem, power and network information simultaneously, providing a much better capture of activities. There are other Kernel developments with KTap and Perf being able to use the same profiling points, along with FTrace. The Linux community is active in the space and we are following developments as they happen. We standardise on Kernel-level interfaces that will keep us in-line with current developments and provide future sustainability.

Tracing Format

The following candidates are identified:

- The Open Trace Format has been developed by the Technical University of Dresden in collaboration with the Jülich supercomputer centre and the University of Oregon and Lawrence Livermore National Laboratory (LLNL) of University of California. Each trace is split up in one or more streams that are written to one or more files each, using an ASCII encoding.
- The Common Trace Format has been developed by EciOS. In this format, a trace consists of several streams that each are stored in a binary format. The format is stored in an ASCII using a language called the Trace Stream Description Language (TSDL).

The use of the Common Trace Format allows the reuse of what was originally discovered as LTTNG's Eclipse plugin, namely Trace Compass ([Trace Compass, 2015](#)). The use of the CFT for trace capture has given access to a GUI for analysis.

Figure 5, Trace Compass plugin for visualising trace files locally ([Trace Compass, 2015](#)).

Query, Visualisation and Modelling

In essence workload trace data are time series based, thus the storage and visualisation of such data, is best suited using systems designed to handle time series data. In this section we identified three time series database and visualisation systems, namely OpenTSDB, Cube, Influx DB, and Grafana.

- The Open Time Series Database (OpenTSDB) has been developed by StumbleUpon. The database behind it is HBase, which makes it highly scalable to large numbers of data points. OpenTSDB is written in Java. The main disadvantage of OpenTSDB is that the standard plotting functionality is very limited to standard gnuplot plots. However this could be circumvented by using an alternate frontend.
- Cube is a system that is build for collecting timestamped events and deriving metrics. The database backend behind it is MongoDB. The visualisation of the data is based on node.js and makes live dashboard creation with zoom and shift functionality available. The authors do not mention anything on the typical numbers that the system can handle. The main issue with Cube is however that it is not under active development or maintainment anymore.
- InfluxDB is an efficient time-series database that supports SQL-like query. A main advantage of InfluxDB over OpenTSDB lies in its easiness of setup and configuration. To install OpenTSDB in a production environment, Hadoop stacks are often required, which can take substantial know-how to configure it correctly. However, InfluxDB is not as mature as OpenTSDB and its cluster features are recently added in the latest version (1.0).
- Grafana is an open source project forked out from another open source Kibana, which is also part of the open source suite --- Elastic Search. However, compared to Kibana, Grafana is dedicated to supporting time series datasets. More importantly, it supports an array of major time series databases including OpenTSDB, Elastic Search, InfluxDB, etc.

3.3 Concept Selection Table

3.4 Risk Assessment Table

3.5 Select Preferred Option(s)


4.0 Critical Technology Element Selection


5.0 Technology Readiness Level Assessment


6.0 List of TBDs

7.0 Prioritised Prototyping Test plan


(please link to JIRA tickets where possible)


 [PRO-149](#) - JIRA issue doesn't exist or you don't have permission to view it.


 [PRO-150](#) - JIRA issue doesn't exist or you don't have permission to view it.

 [PRO-151](#) - JIRA issue doesn't exist or you don't have permission to view it.

 [PRO-152](#) - JIRA issue doesn't exist or you don't have permission to view it.

 [PRO-153](#) - JIRA issue doesn't exist or you don't have permission to view it.

 [PRO-220](#) - JIRA issue doesn't exist or you don't have permission to view it.

 [PRO-226](#) - JIRA issue doesn't exist or you don't have permission to view it.

8.0 Not Doing/Not Considered