Introduction To SeaScape

2020 R3 Release

Guidelines For Training

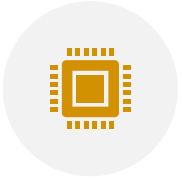
- This event will be for 2 hours. There will be 15 min Q&A at the end of this session for questions.
- To ask questions during training, use the Q&A window and select option "All panelists" to ask question
- For audio, join using Audio Broadcast option
- Slides and recording links will be sent in 1-2 days after training
- Users successfully completing the training will get a digital certificate.
  - Need minimum 70 % attendance for this (10/14 training sessions)

Agenda

Introduction To Big Data

SeaScape Platform

User Interface



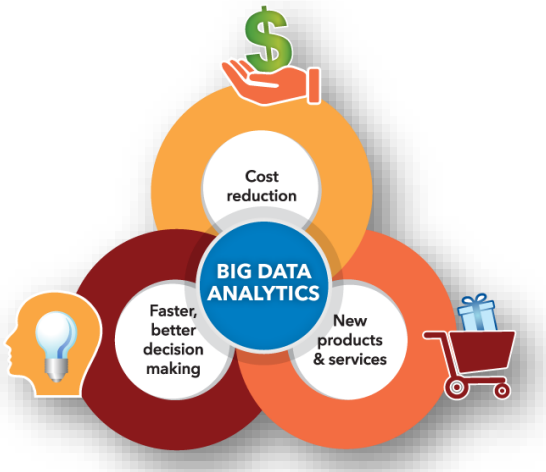
SeaScape Help System & API's

Mapreduce Programming

Using SeaScape Platform

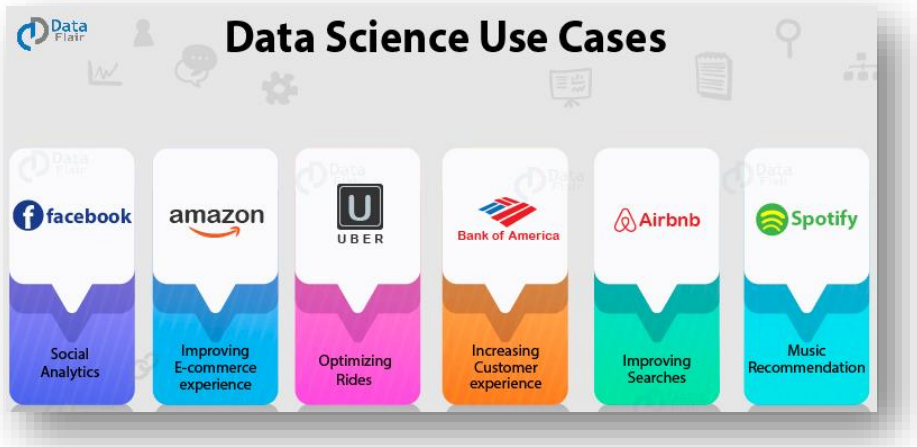
Introduction To Big Data

Introduction to Big Data Analytics

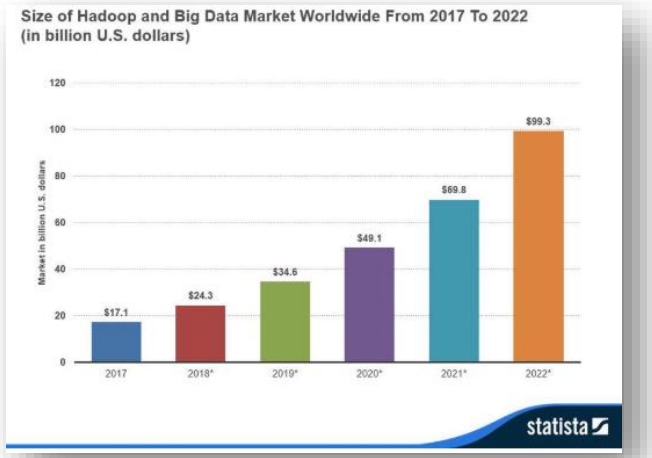


How big data analytics help ?
Image courtesy : https://sas.com

Challenges in Big Data
Image courtesy : https://edureka.com



Top Applications of Big Data
Image courtesy : https://data-flair.training



Growing market of big data science – 100 B dollar by 2022
Image courtesy : forbes.com

Big Data Techniques

Google search says :

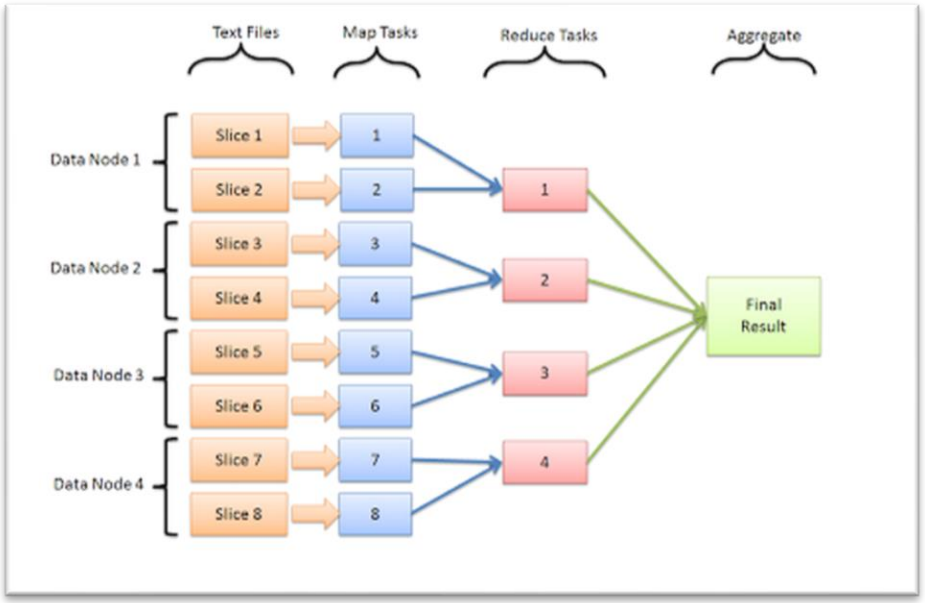
MapReduce is the heart of **Apache® Hadoop®.** It is this programming paradigm that allows for massive scalability across hundreds or thousands of servers in a **Hadoop** cluster

Ansys says :

Scheduling algorithms similar to MapReduce are the heart of **SeaScape®.** It is this programming paradigm that allows for massive scalability across hundreds or thousands of servers in **customer** cluster

Big Data Techniques

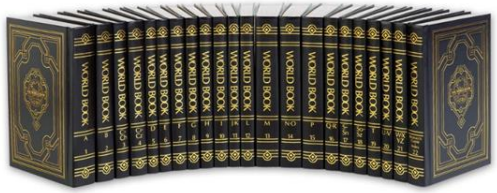
- **MapReduce** is a simplified model for processing large data sets with a parallel, distributed algorithm on a cluster in an automated, easy to write, fashion
- All big-data engines use MapReduce or its derived/augmented techniques
- Some popular big data engines below:

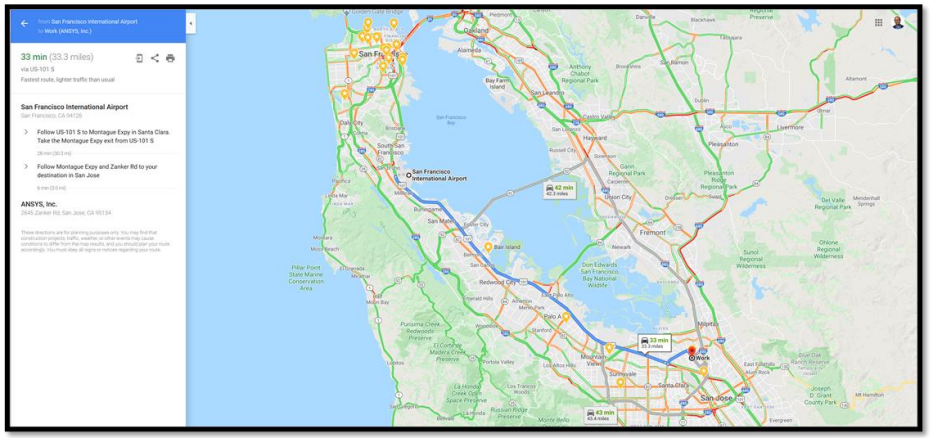


SeaScape Platform

Ansys

Foundational Innovation was Necessary





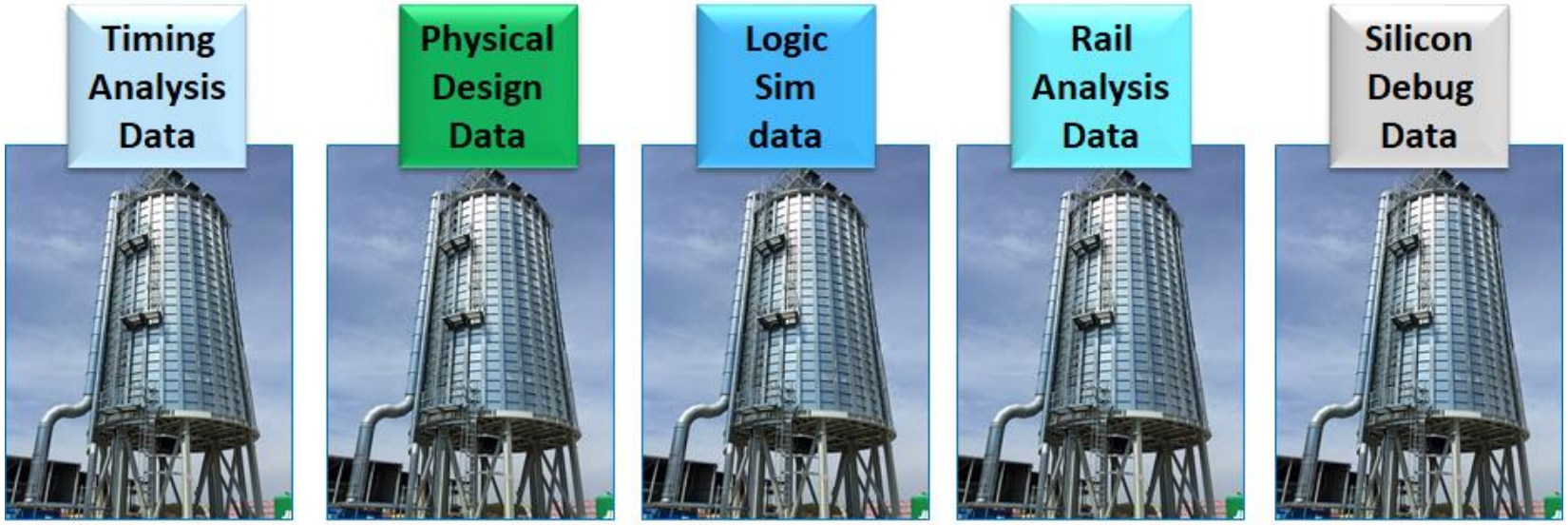
“Pre-Google™”

The data was there, but you had to work hard to make it useful.

Today

Data is instantly available and useful; freeing you to create more

Traditional EDA Big Data Exists in Silos



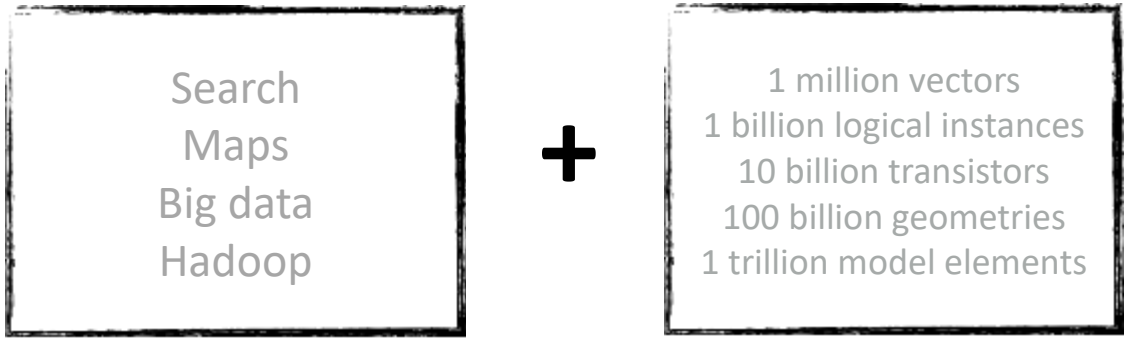
Multi-Domain Chip-Level Analysis is not well supported by existing EDA tools and databases

Problem with Silos

Design margins too large

Coverage is poor

Results are not actionable

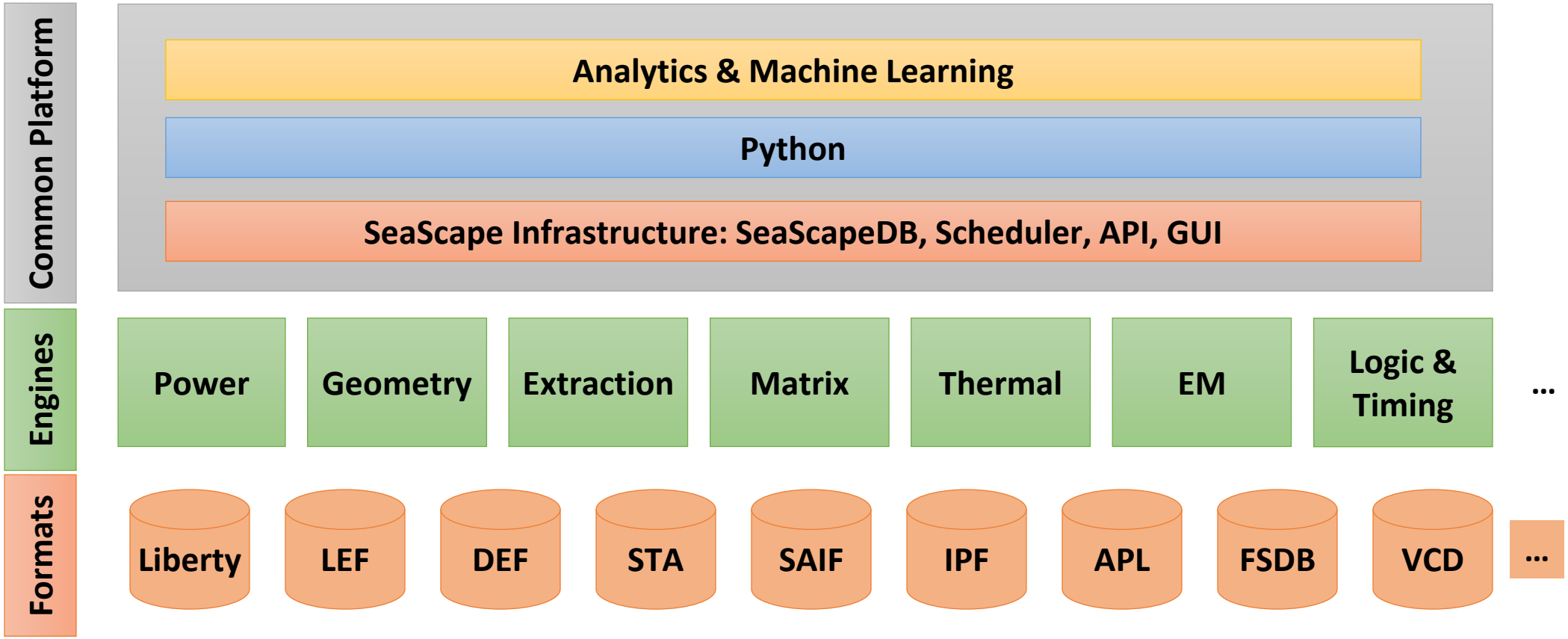


Big data for chip design?

The SeaScape Platform

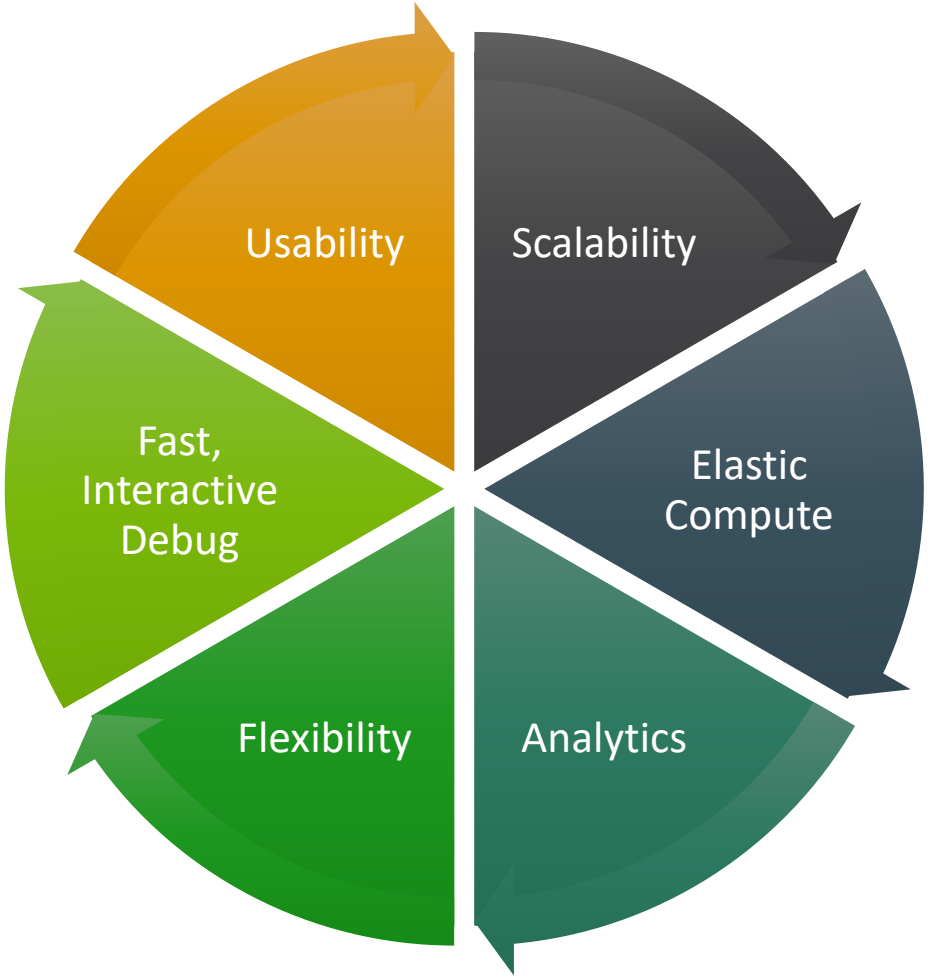
EDA-specific platform for distributed data and compute

1. SeaScape is the industry's first big data platform for Actionable analytics, using Big data and machine-learning apps
2. All products of ANSYS will work with the SeaScape platform.



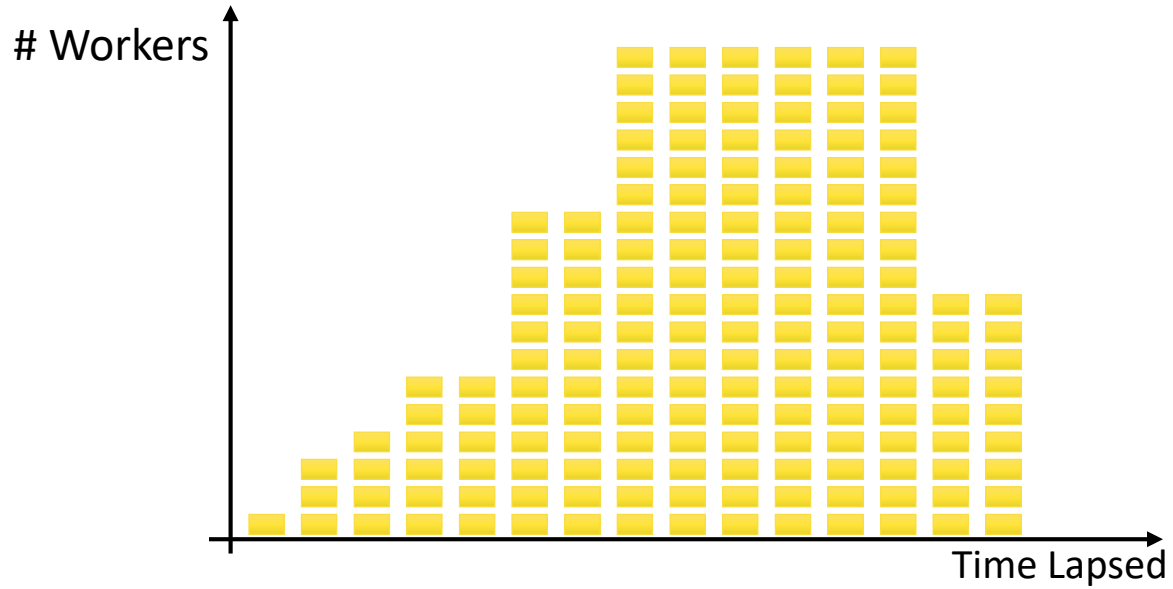
Advantages of using SeaScape Platform

- Scalability: can handle massive full-chip designs with billions of instances
- Elastic Compute: efficient use of hardware resources
- Powerful Multi-Variable Analytics: User can apply MapReduce to query data and compute custom results, generate flexible heatmaps
- Flexibility: one run can open multiple dbs, each with multiple views
- Fast Interactive Debug: load huge design layout and results in multiple GUI in seconds
- Usability: MapReduce Wizard, Console, Online Help



SeaScape Platform

Integrated ML Stack for Scientific Computing



Launch runs immediately
- No waiting for full machines
- Instantly view results on single cores

Auto checkpoint
- Reuse data from terminated runs
- Build multiple flows & reuse data

Resiliency
- Auto-recover from bad cores
- Or network issues

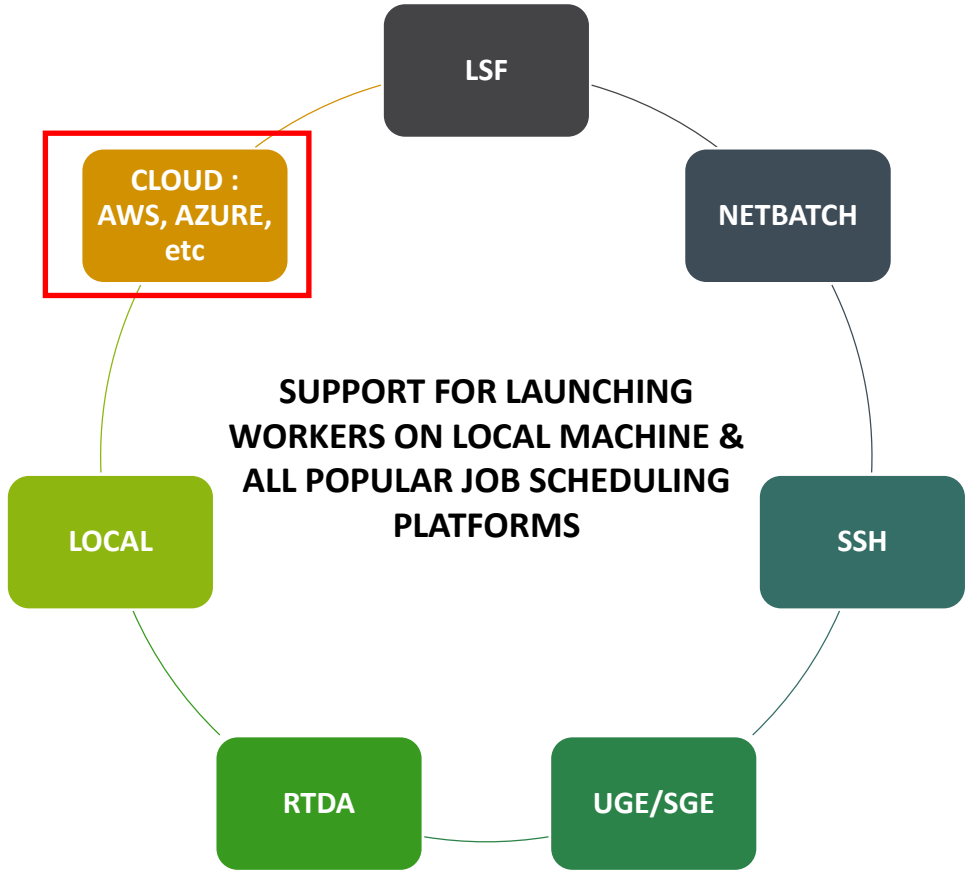
Parallelization
- All stages are scalable across cores
- Matrix solve > 1,000 cores

SeaScape Big Data Platform

Actionable Analytics | Machine Learning / AI | **Elastic Compute** | Visualization & Debug

Launching Workers

- **Definition of worker :**
  - SC is a distributed platform, runs on a set of CPU cores
  - CPU cores used by RH-SC are called workers
  - SC can launch hundreds of thousands of workers to process jobs efficiently
  - 1 worker == 1 CPU core with certain amount of memory
- **Where are the workers located ?**
  - Workers can be on the same machine or different machines in a network
- **How are workers launched ?**
  - SC supports all popular job scheduling platforms like LSF , UGE , RTDA etc
  - SC can also run on cloud based services like AWS, Azure etc



More On Workers

- **Launcher commands syntax :**

```
create_grid_launcher(name, submit_command, num_workers_per_launch=None, initial_exec_function=None,
max_num_threads_per_worker=None, memory_gb_per_worker=None)
```

  - name : Launcher identifier name (type=str, required=True)
  - submit_command : grid submit launch command (type=str, required=True)
  - num_workers_per_launch : specify number of workers per launch (auto detect if not specified) (type=int, default_value=None)
  - initial_exec_function : function pointer (type=object, default_value=None)
  - max_num_threads_per_worker : maximum number of threads per worker (type=object, default_value=None, constraint="int > 0 or string 'auto'")
  - memory_gb_per_worker : requested memory in GB (type=int, default_value=None, constraint="int > 0")

- **Example of UGE launcher command :**

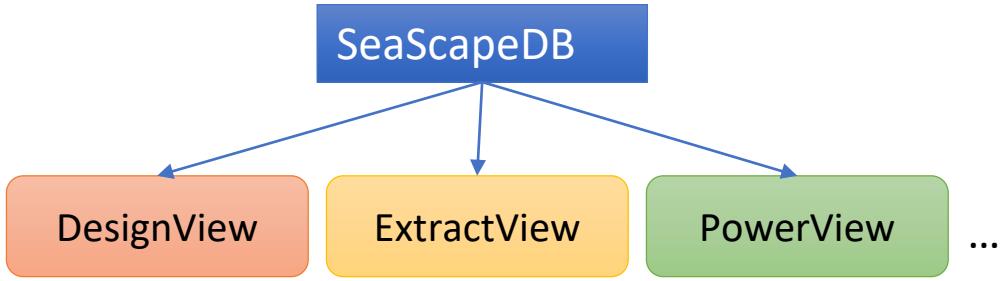
```
ll = create_grid_launcher('uuu',"qsub -V -b y -q sjo_all_hosts -l mfree=10G -l h='sjoq*' -l platform='linux64e7' -o
uge.log". num_workers_per_launch=1, max_num_threads_per_worker=2)
```

- **Launch workers:**

```
ll.launch(64) # launch 64 workers
```

- **Making The Launcher Automatic (optional):**
  - use 'register_default_launcher' function for SeaScape to automatically launch as many workers as needed [as run progresses]:
  - `register_default_launcher(launcher=ll , min_num_workers=5, max_num_workers=50)`

The SeaScape Database



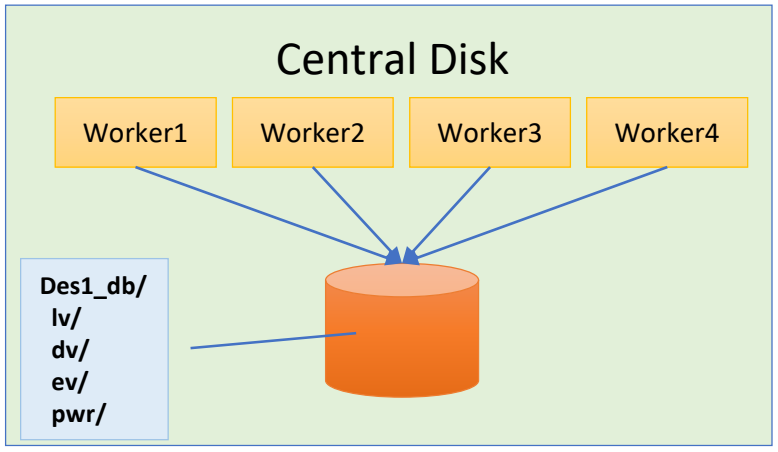
- SeaScapeDB holds all data used within SeaScape
- Data from external formats like DEF or FSDB
- Results of processing within the tool such as instance power

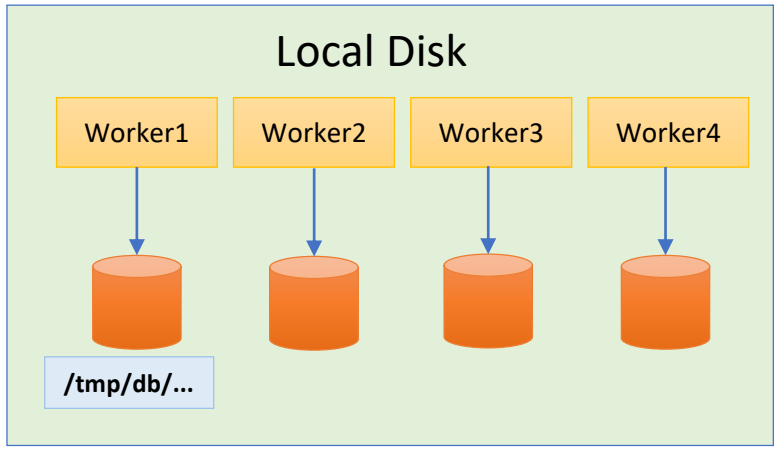
- One db contains multiple views
- Each view holds related data
- All data is partitioned into chunks for efficient processing

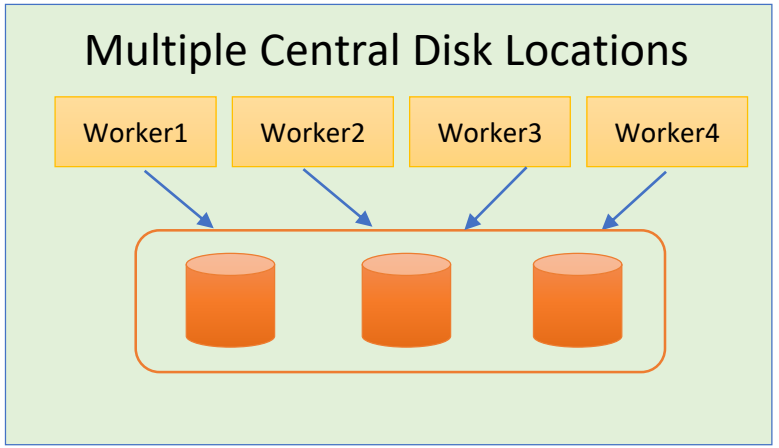
- Data is initially stored in each worker's memory, then later written to disk in the background

- Can read and/or write to multiple dbs in one tool session

DB Storage Options

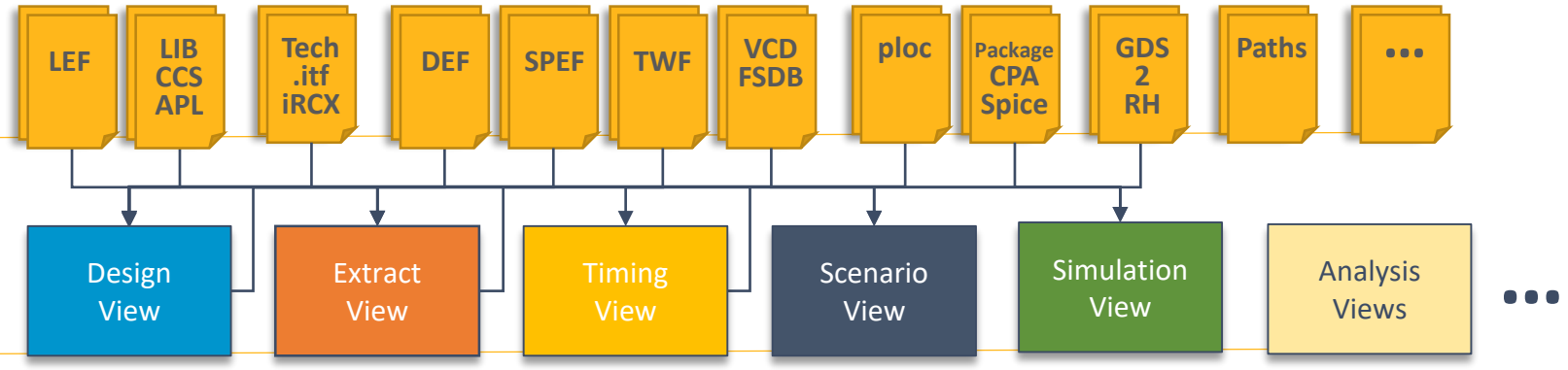




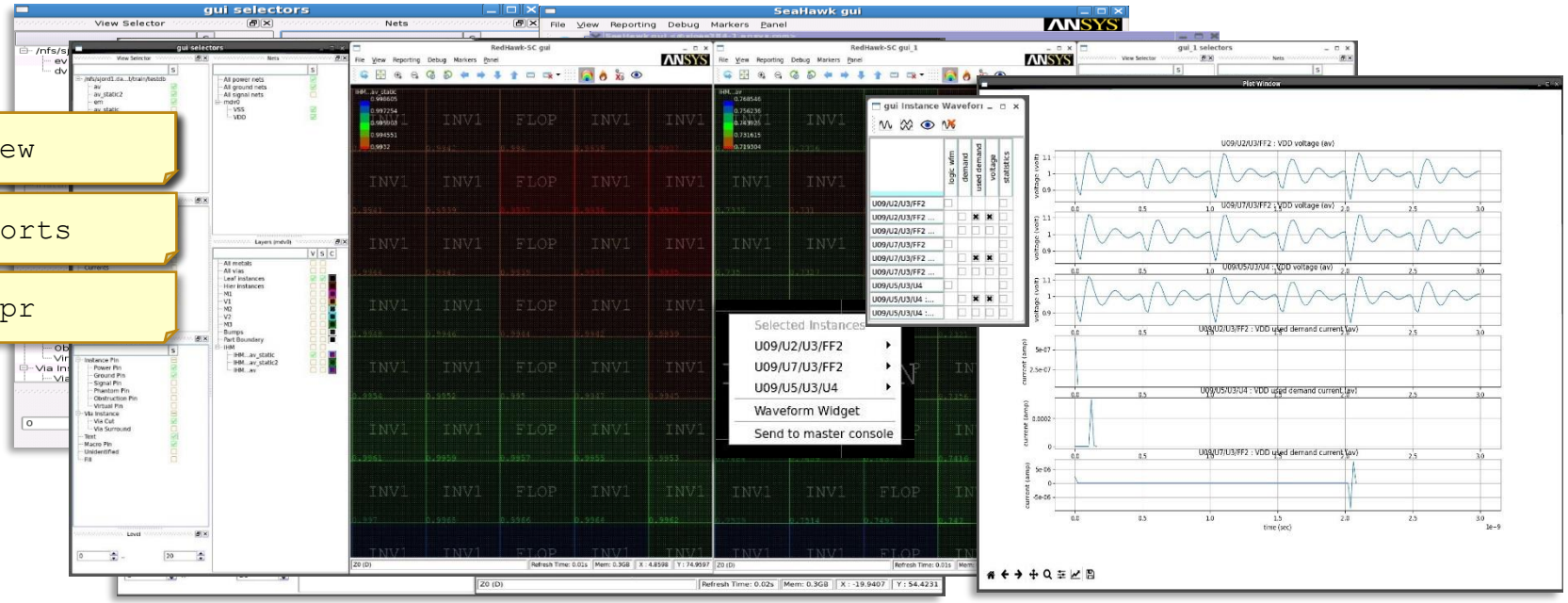


SeaScape

EDA-Project Data into SC "Views"




```
<python> create_analysis_view
<python> report_detailed_shorts
<python> get_instance_pin_spr
```



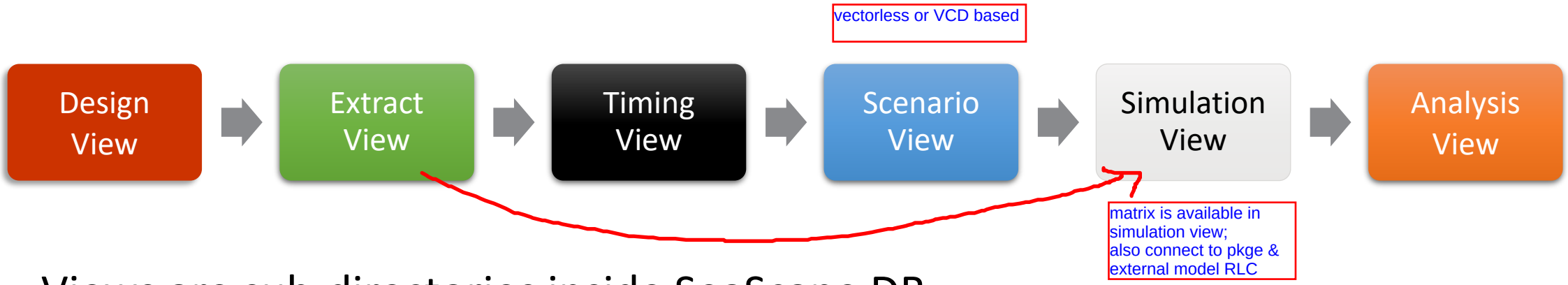
RedHawk-SC User Case

SeaScapeDB – A View Oriented Data System

SeaScape DB Overview

- A SeaScapeDB is simply a linux directory containing views
- The db is created through *"open_db"* command
- Directory name for the DB is set in the "*open_db"* command
- If db already exists, the db will be opened
- DB contains several views, each view is a sub-directory in the DB
- Views are created through "*create_<type>_view"* command
- Directory name for the view can be specified while creating the view
- Multiple SeaScapeDBs can be opened simultaneously
- A DB can be accessed simultaneously from multiple RH-SC sessions

Views in SeaScape DB



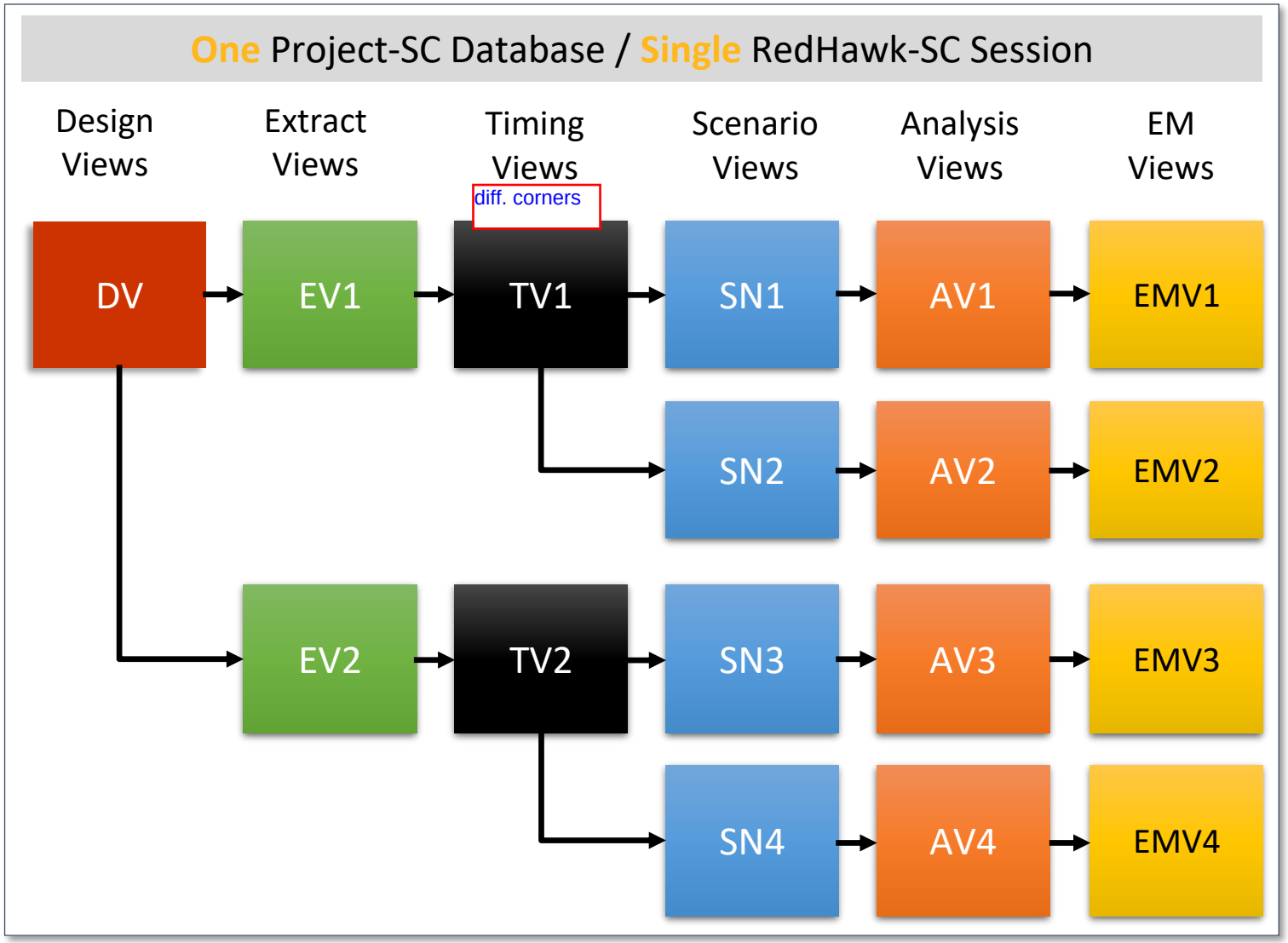
- Views are sub-directories inside SeaScape DB
- They are representations of a group of data
- Each view stores data from different stages of analysis
  - Write once, read many times
  - Automatic versioning is maintained when a view is over-written
  - Tag/directory name will have additional #v1, #v2… appended
  - Example tag='tv', on disk will be: ./<SeaScapeDB>/tv#v1)
- It is possible to restart the run from any view for faster TAT
- Views are python objects with query APIs to access their data

a view is written only Once, but can read many times. Once written, is is locked.

Views - Example

- **Single session can have views for:**
  - Different corners (Liberty and Extraction)
  - Different conditions (voltage, temperature)
  - Different modes of operation
  - Different vector sets
  - Different vector-less settings
  - Different conditions for IR drop versus EM
- **Example**
  - SN1, SN2 represent VCD & V-less scenarios
  - EV1 and EV2 represent 2 different extraction corners
- **Related views can be traced-back**
  - EMV4 ◄ AV4 ◄ SN4 ◄ TV2 ◄ EV2 ◄ DV

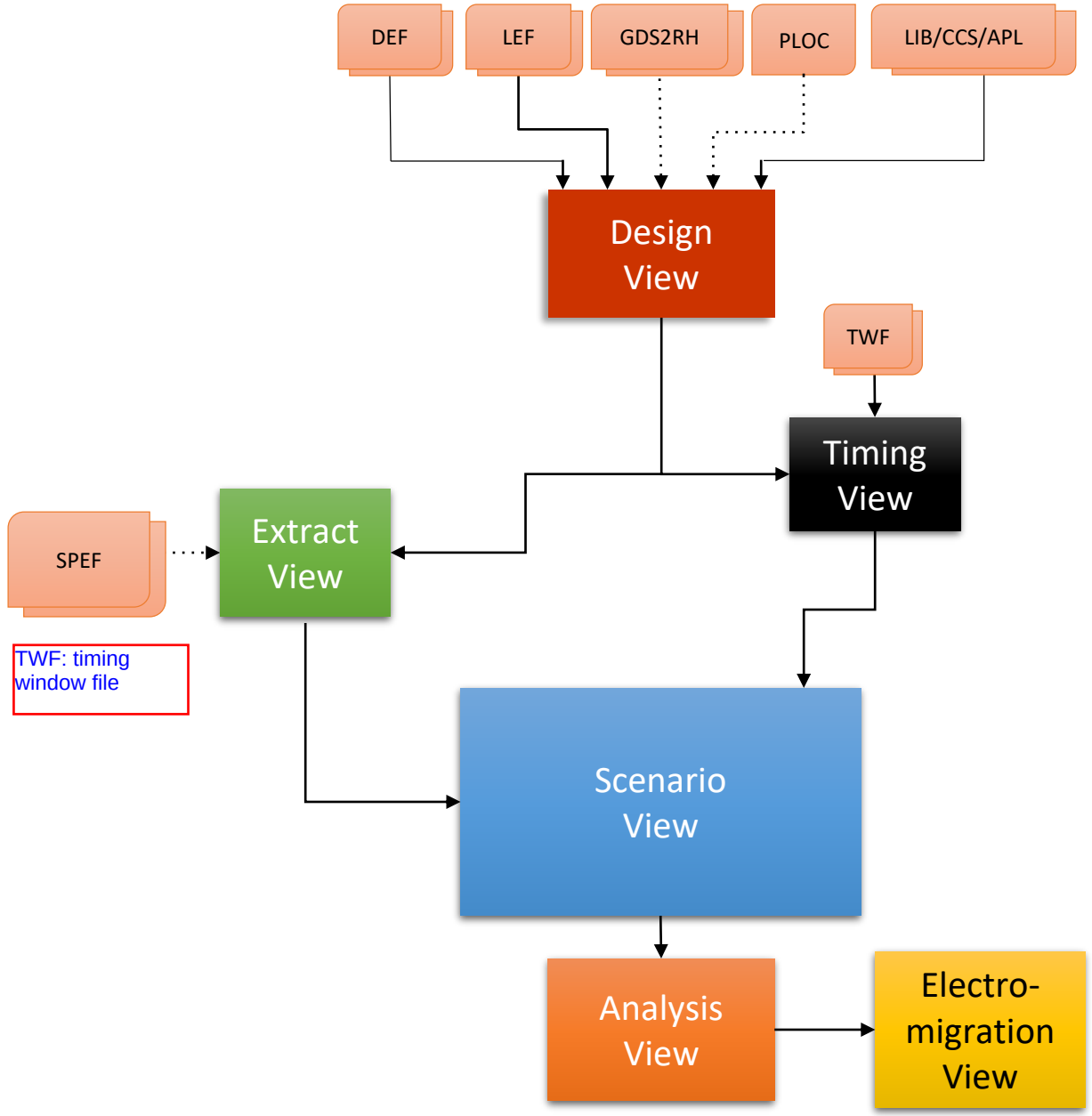
through attribute commands



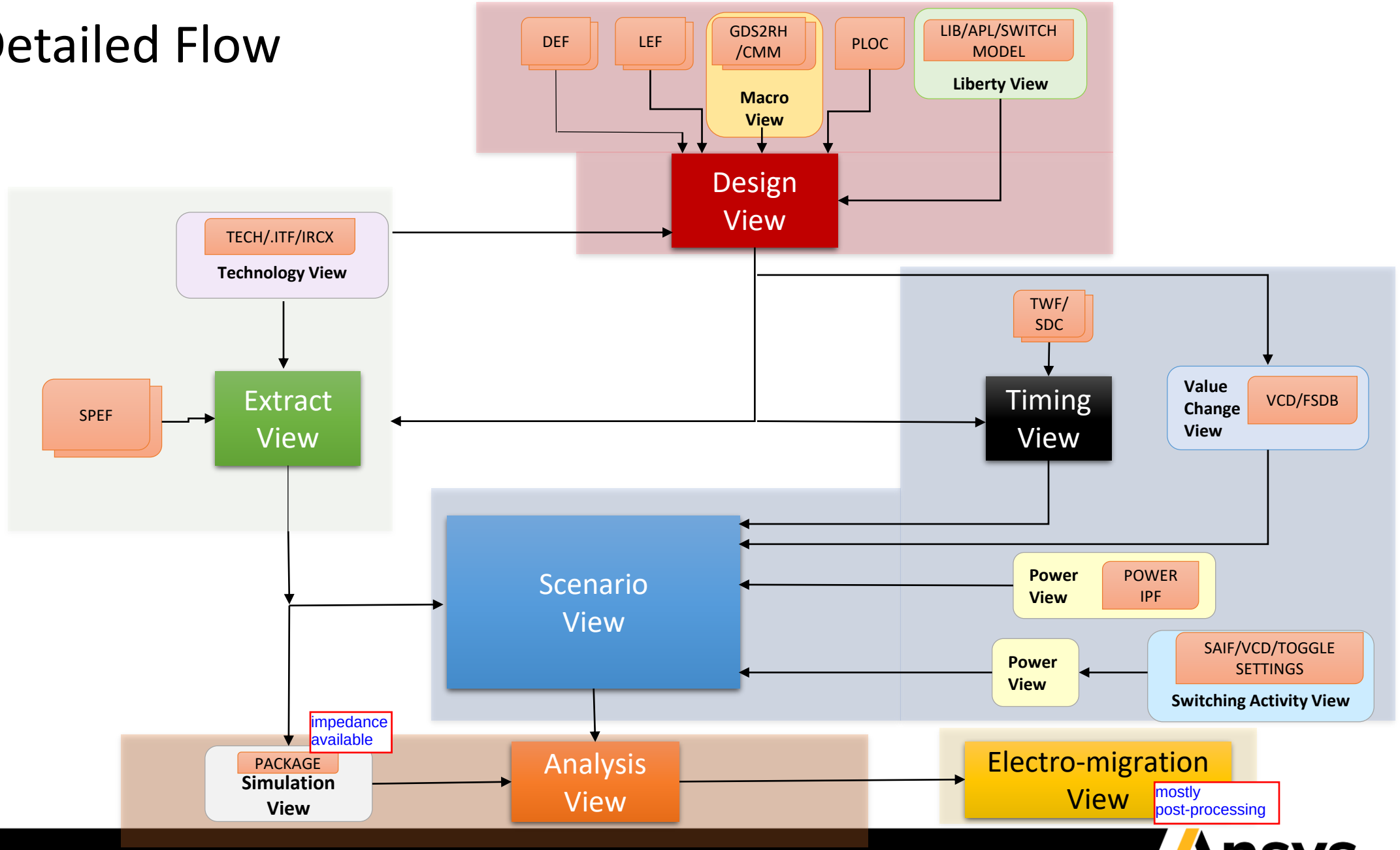
Flow Overview

Main SeaScape Views

View	Function
DesignView	• Stores netlist and layout information • Library information (through LibertyView)
ExtractView	• Stores RC extraction & SPR data • Imports and stores SPEF
TimingView	• Stores imported TWF and SDC constraints • Stores clock associations and slews. • Computes timing graph.
ScenarioView	• Calculates DC instance currents for static scenario • Stores logic events and transient instance currents for dynamic scenario
AnalysisView	• Stores result of simulation IR sim.
ElectromigrationView	• Reads in EM rules and produces EM results on every wire/via segment



More Detailed Flow



Supplementary SeaScape Views

View	Function
LibertyView	• Imports and stores Liberty, CCS models and APL models
MacroView	• Read in macro models like gds2rh , Totem CMM models , etc
TechView	• Stores extraction technology information from Ansys tech file or derived from ITF or iRCX
ValueChangeView	• Stores imported FSDB/VCD.
SwitchingActivityView	• Stores toggles – propagated/Unpropagated • Reads in toggle rate settings ,SAIF, VCD/FSDB etc
PowerView	• Stores imported instance power file • Processes power and toggle scaling • Computes power based on toggles – propagated or unpropagated from SwitchingActivityView
SimulationView	• Sets up for the analysis and reads in package models

Launching RedHawk-SC

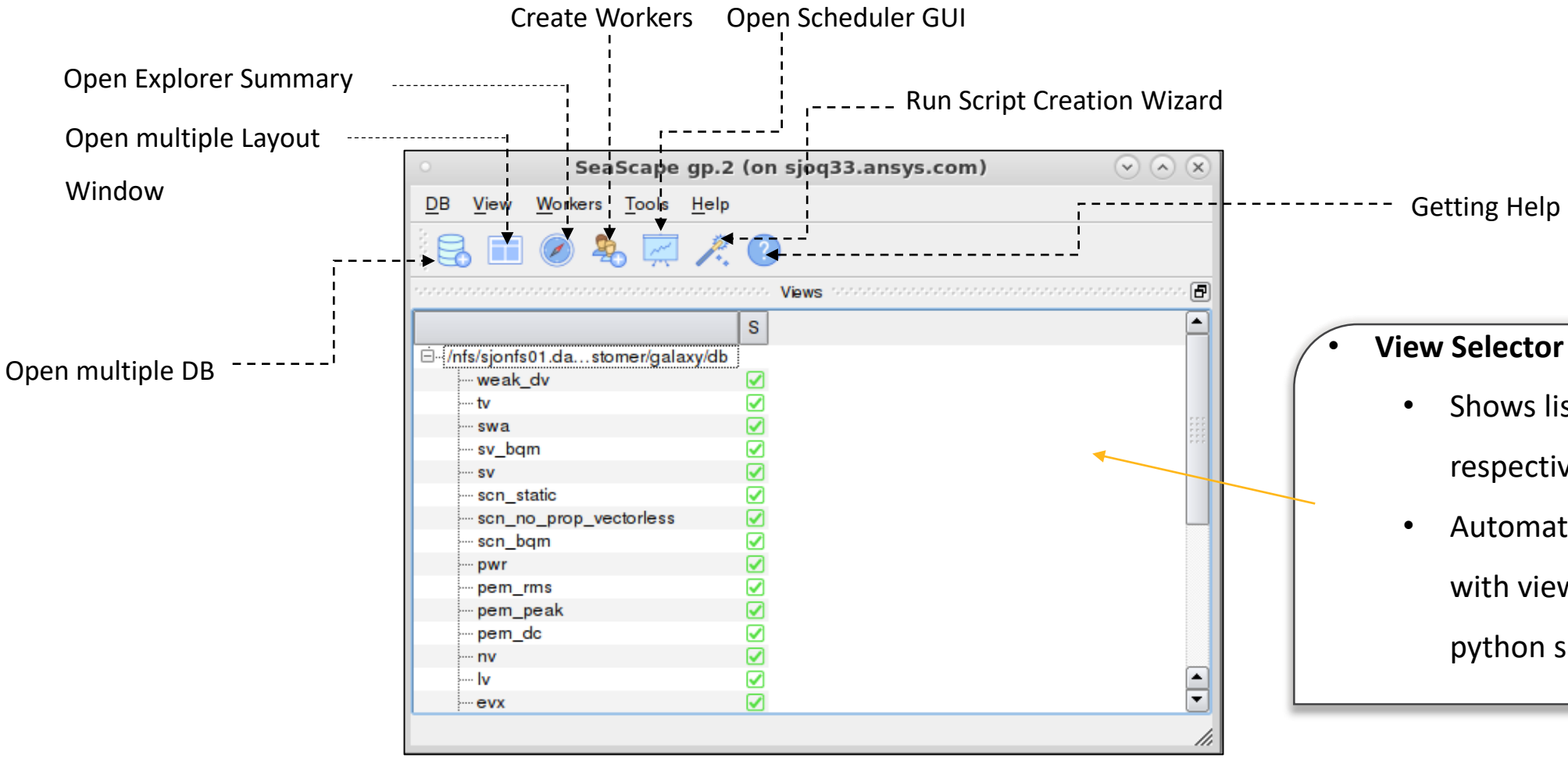
- **Batch mode execution example:**
  - `<path_to_rhsc_installation>/bin/redhawk_sc <python_command_file>`
- **Interactive mode execution example:**
  - `<path_to_rhsc_installation>/bin/redhawk_sc -i <python_command_file>`
  - It needs an exit() command in script or entered manually to exit the Python shell
- **Connecting to a live RedHawk-SC run:**
  - RedHawk-SC allows querying of data/results from an active session, by remotely attaching to the session
  - Multiple users can attach to the same session from multiple machines for querying/viewing results
  - `<path_to_rhsc_installation>/bin/redhawk_sc -r <gp_dir>`
- **Execution Log Files:**
  - All RHSC log files reside by default in gp<> folder
    - If the run is fired in the same directory, tool will create gp.1, gp2 incrementally. ‘latest.gp’ link will point to the most recent gp directory
  - Main log file for RedHawk-SC would be <gp_directory>/run.log file.
  - Other important log files are run_v.log (more detailed log file for master), Worker*.log files for individuals workers.

SeaScape GUI Overview

Launching RedHawk-SC Console

Run command : `<path_to_rhsc_installation>/bin/redhawk_sc <python_script> --console`

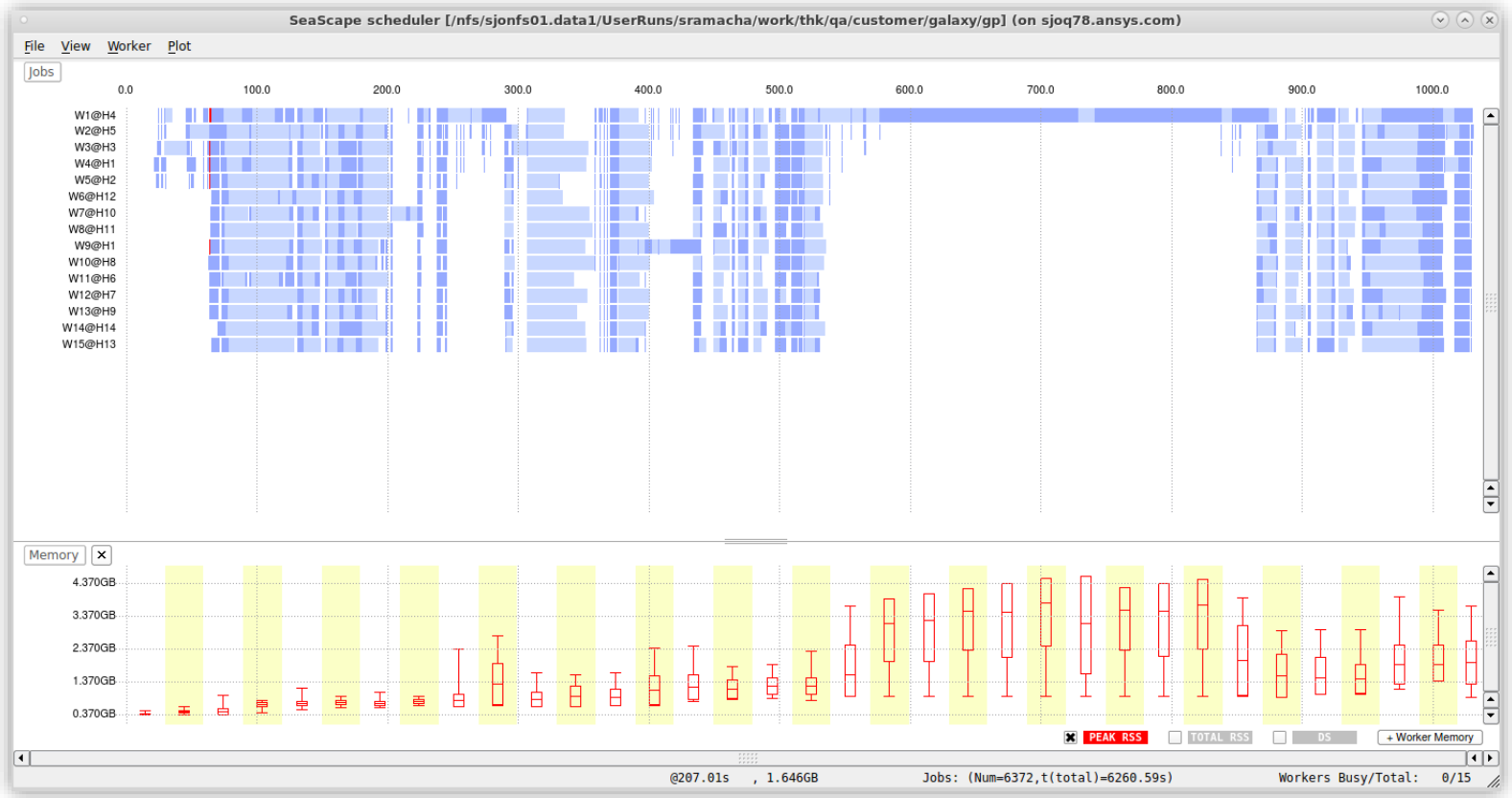
To invoke from RHSC shell , type : `open_console_window()`



- **View Selector**
  - Shows list of open DBs and their respective views
  - Automatically create variables with view names in the RH-SC python shell

Launching Scheduler

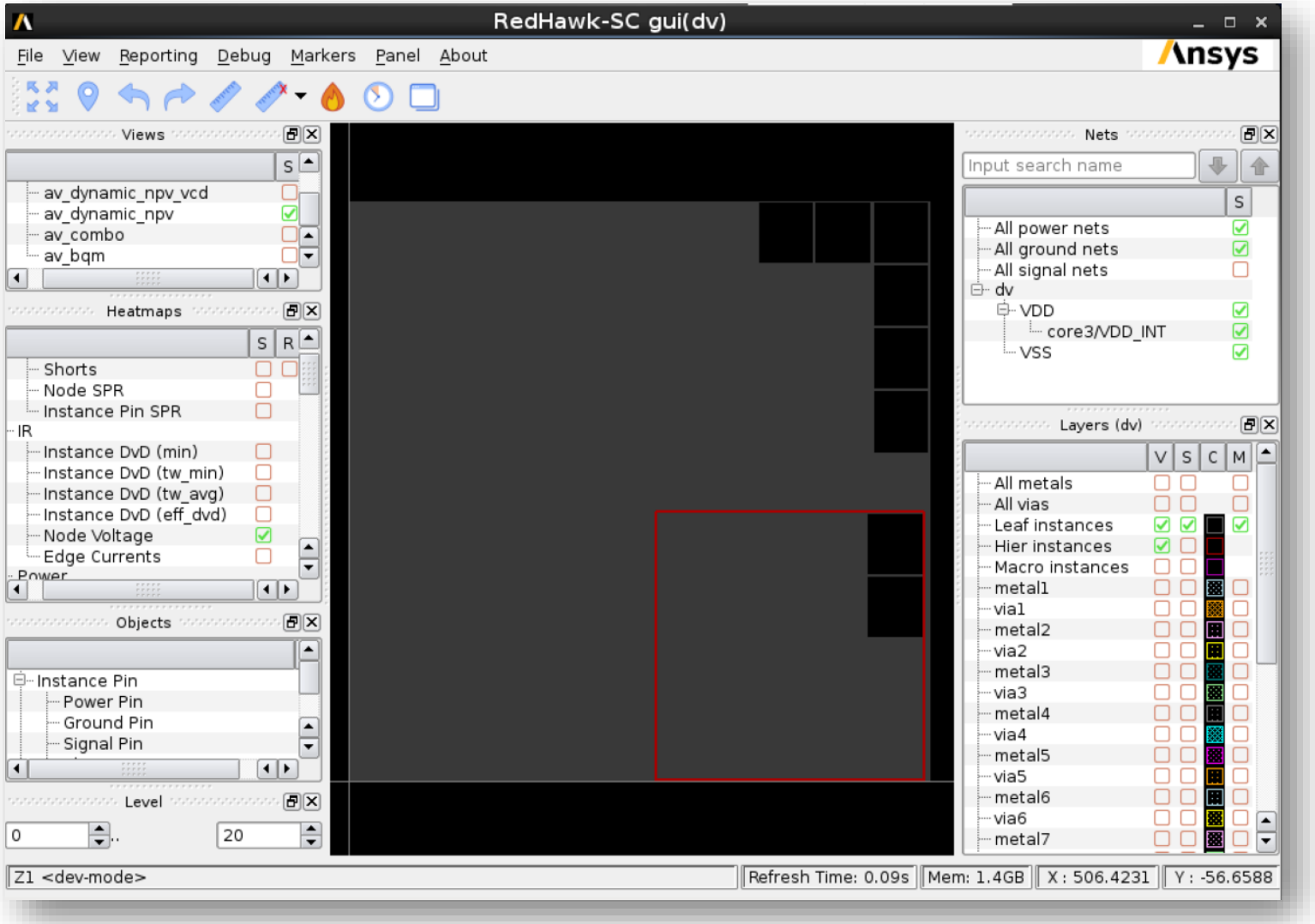
- Provides real time visualization of workers and their jobs
- Displays the jobs executed per worker (Y - axis) on the time axis (X - axis)
- Can be invoked from console or using the command '**open_scheduler window()**'
- For an ongoing or completed run , from the command line , we can invoke the scheduler gui by the command : **redhawk_sc –g –l –r <path_to_gp_directory>**



FEATURES OF SCHEDULER GUI

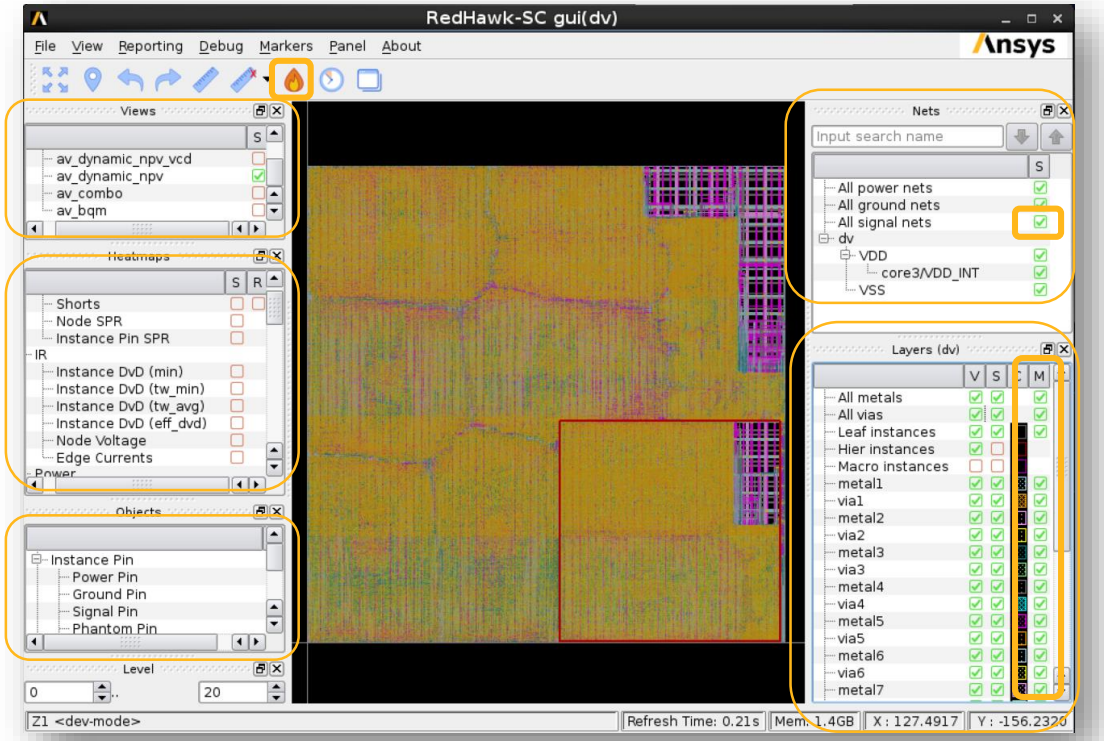
- **Worker info**
- **Host info**
- **Highlight Jobs using**
  - run-time
  - memory
- **Total number of workers, jobs**
  - name
- **Worker Stats Plots**
  - Worker memory
  - Machine load
  - Read ds rate
  - Total ds read
- **Identify network slowness/issues**
  - red boxes
  - using plots
- **Identify performance sticks ***

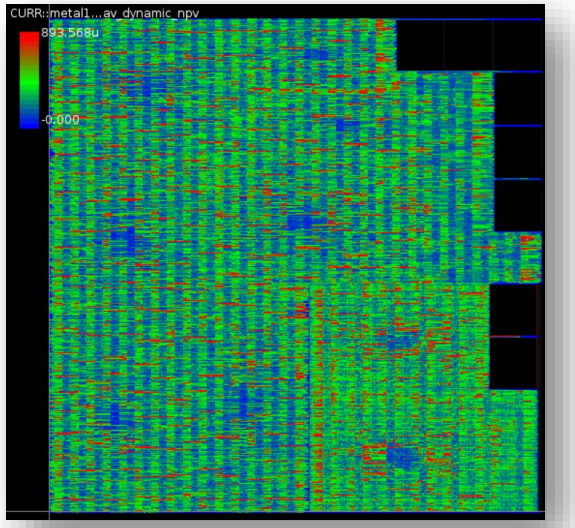
RedHawk-SC™ GUI - Thin Client

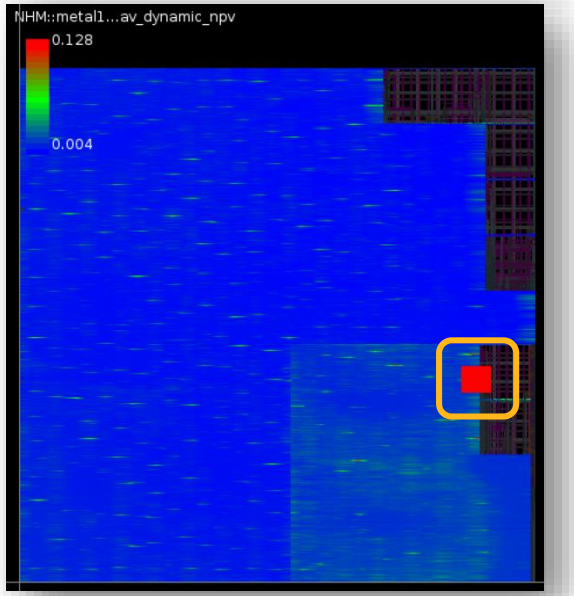


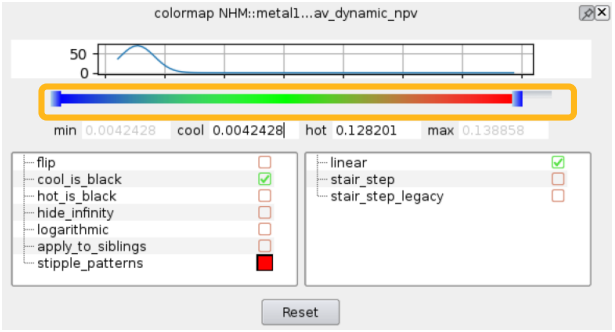
- GUI is a separate process that connects to master
- Multiple GUIs can be opened at same time with locked zoom
- It can run on a different host, or different network
- All data reads are on demand
- Very low memory consumption for viewing largest designs (~ 16 GB)

RedHawk-SC™ GUI - Thin Client

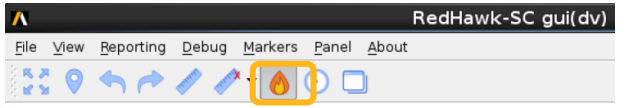








- ✓ **SLEEK SLIDER** to instantly change map thresholds



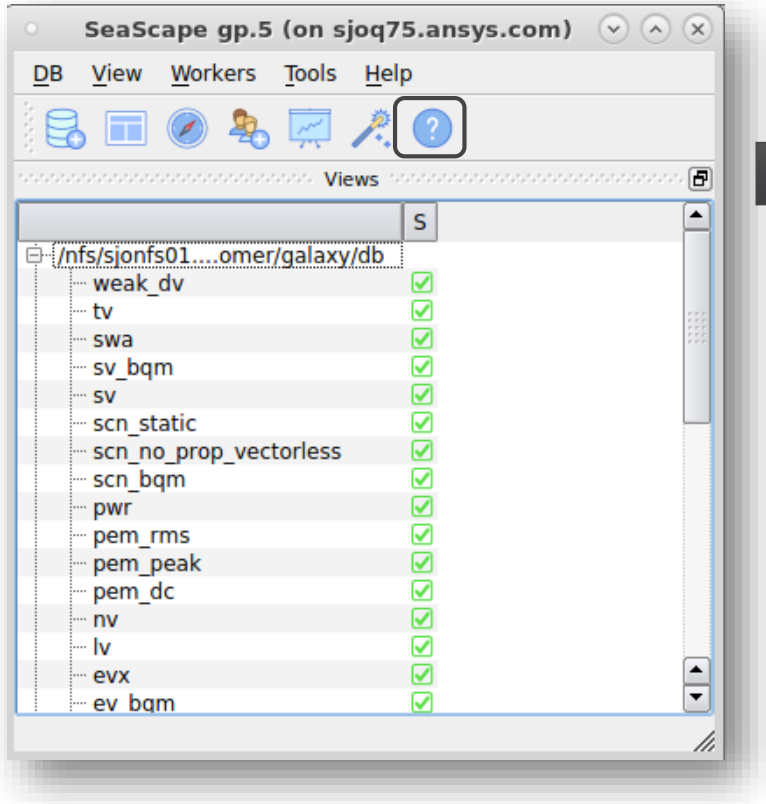
- ✓ **FLAME BUTTON** to quickly find hotspots above a threshold

- ✓ **VIEW SELECTOR :** viewing results from multiple views in same GUI
- ✓ **HEATMAP SELECTOR :** enabling multiple heatmaps together
- ✓ **OBJECT SELECTOR** for easy access to all PnR world objects – pins , vias , OBS etc
- ✓ **NET SELECTOR :** show PG and signal nets in single GUI
- ✓ **LAYER SELECTOR :** quick enabling/disabling layers
  - **MAP SELECTOR** (M button) for easy map access for instance based/layer based heatmaps

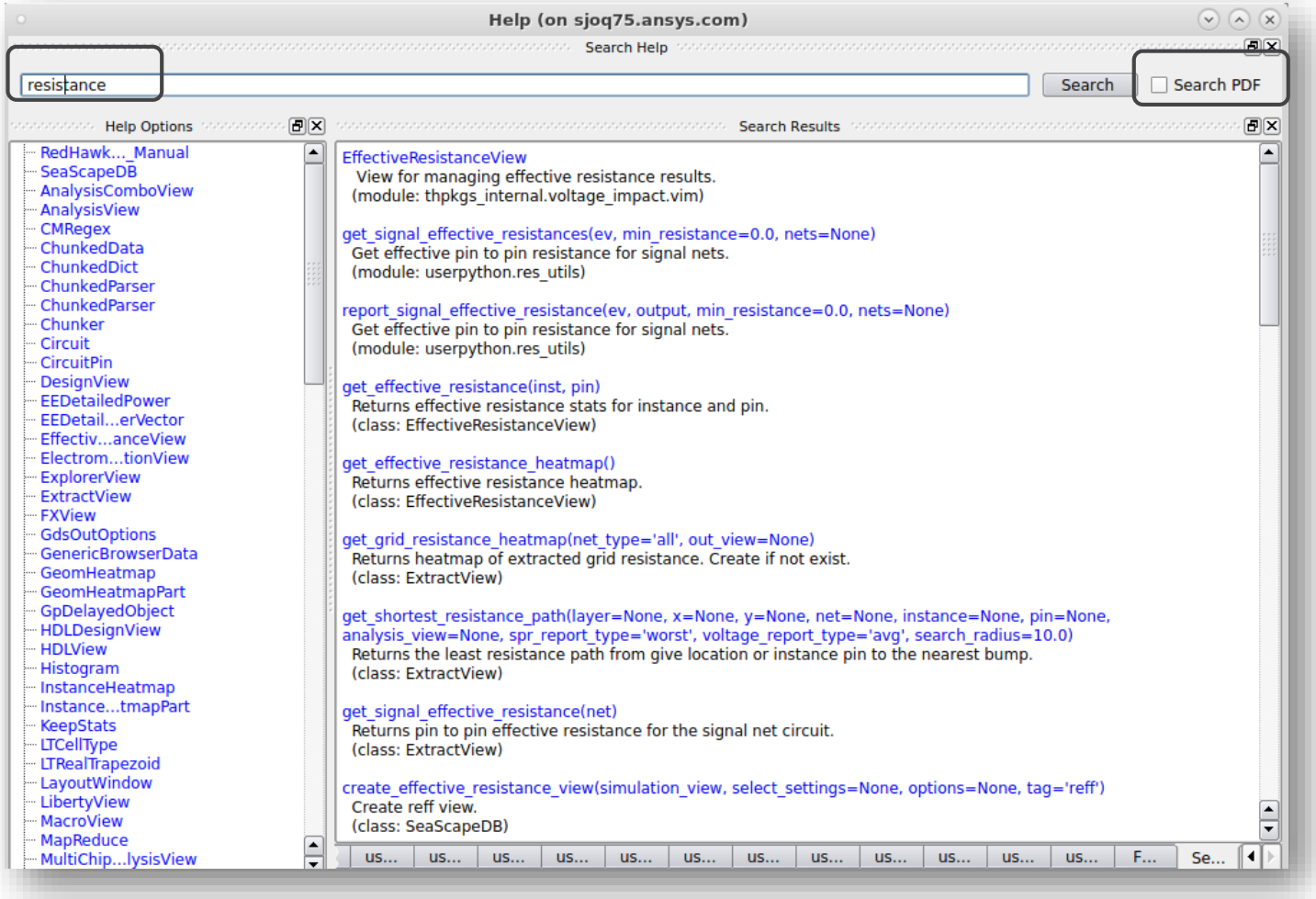
Introduction to SC Help System & APIs

Ansys

Accessing Help From RedHawk-SC Console

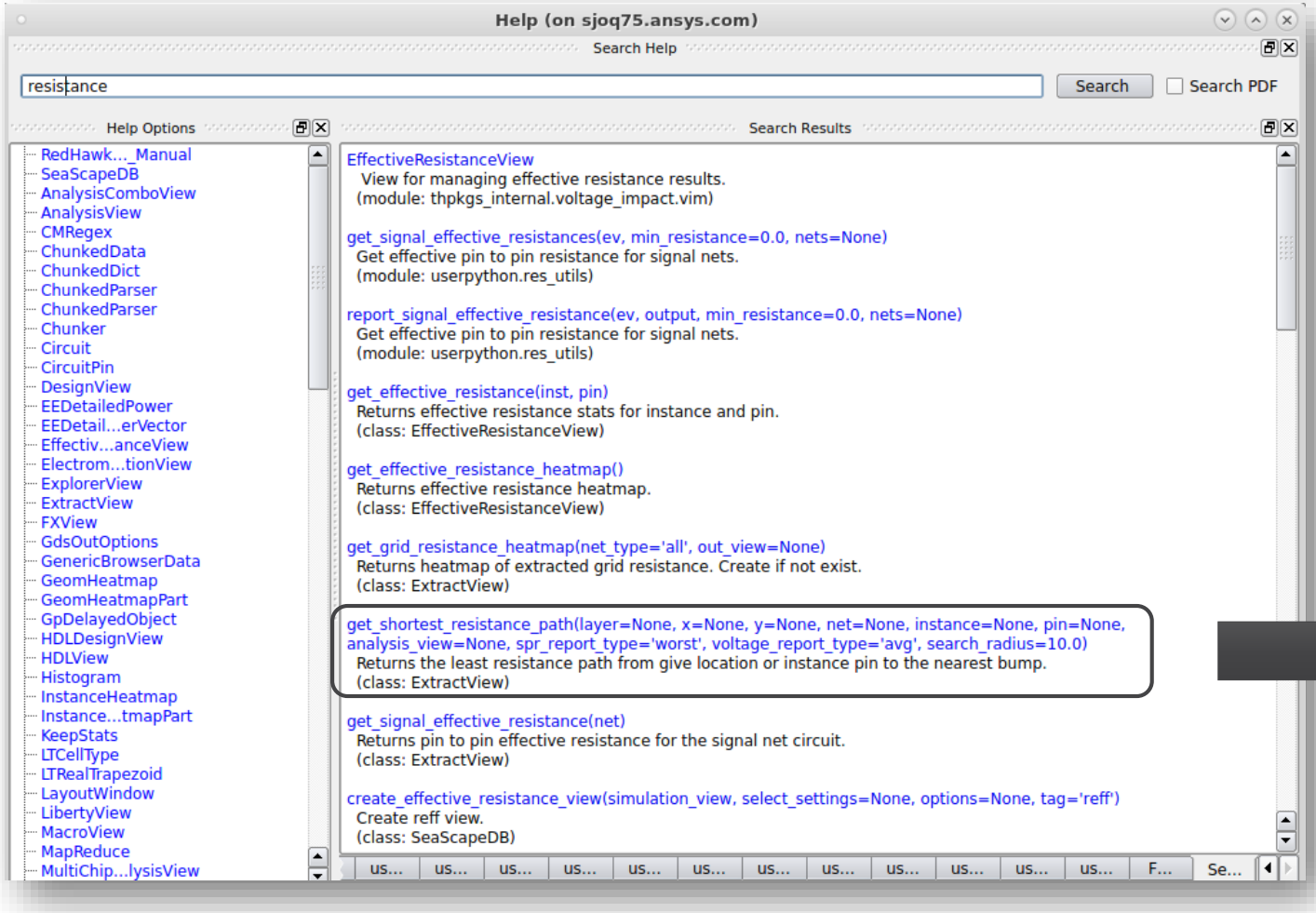


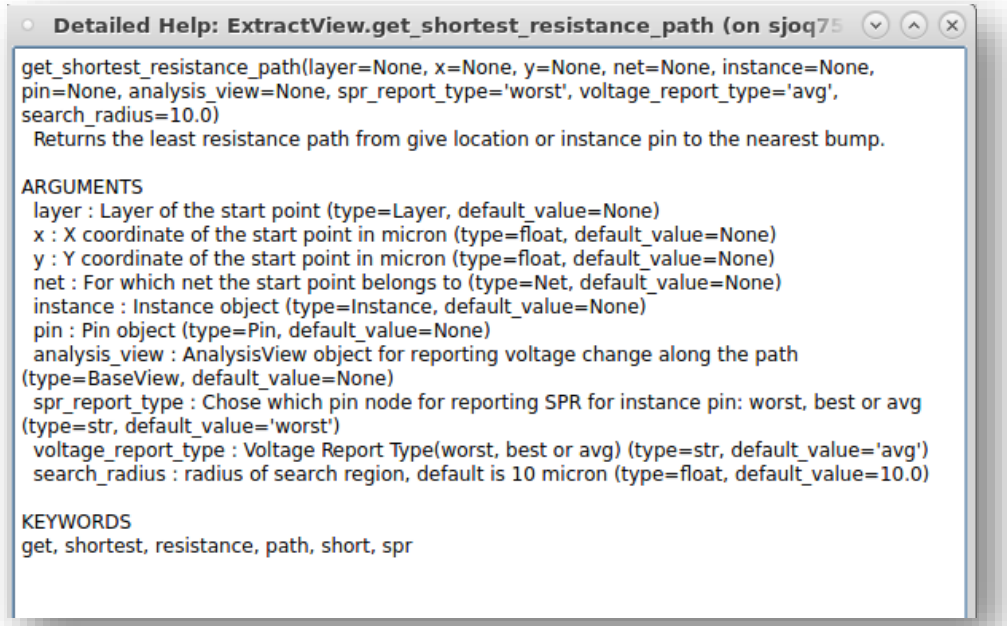
- Open console and click on help button



- Enter the item , for e.g resistance in the search box and get all results
- Click on ‘Search PDF’ if you want to search inside RedHawk-SC documentation

Accessing Help From RedHawk-SC Console

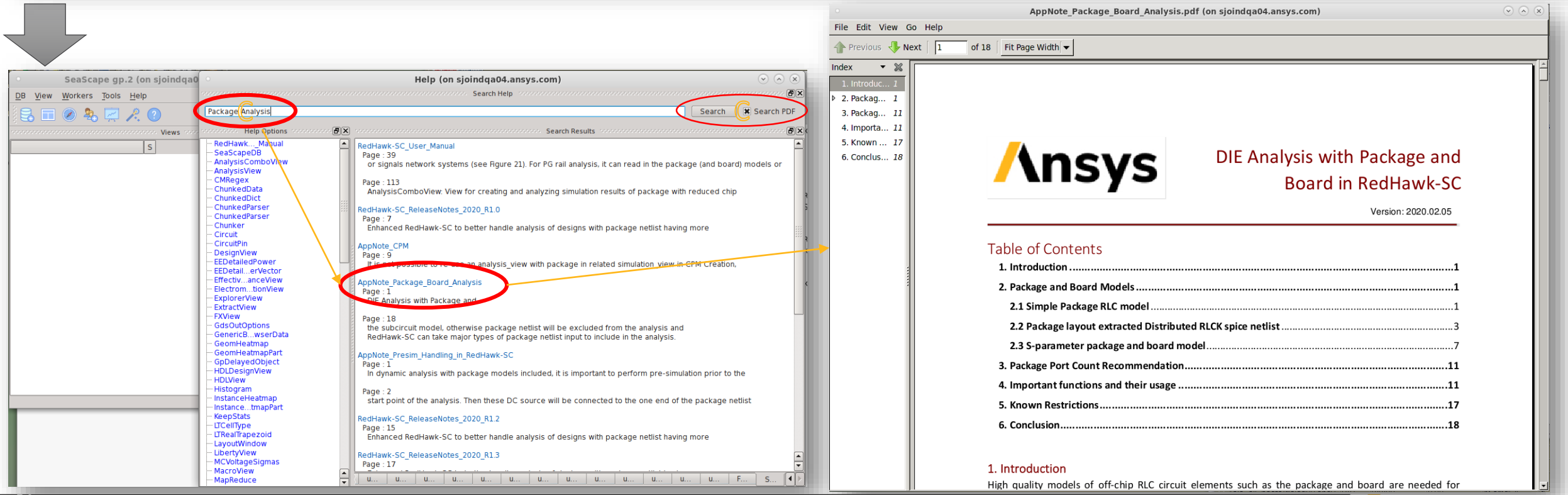




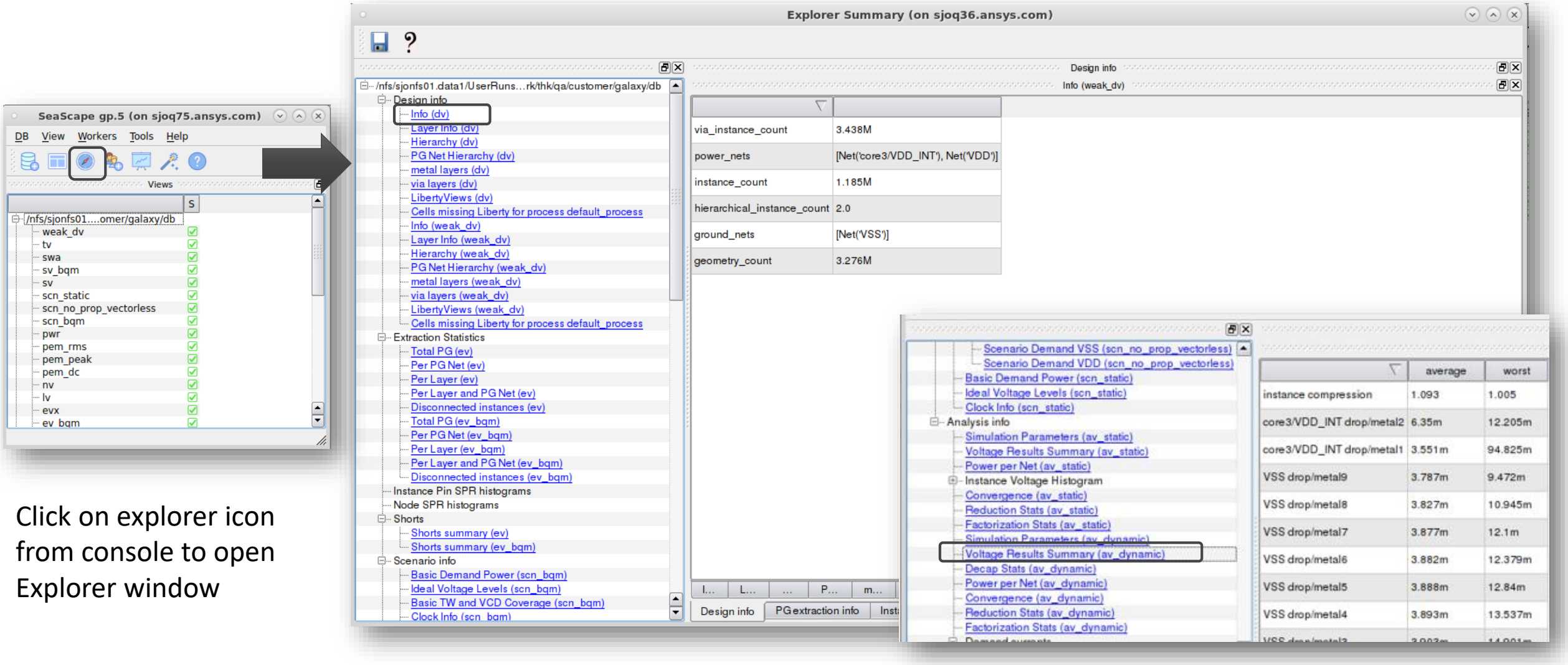
- Click on any search result to get more details

Application Notes

```
indqa@sjoindqa04.ansys.com - /projs00/tomahawk/build/r20_R2/20.05.23_09.01.04/linux_x86_64_rhel6/Optimized/seascape_release/2020_R2.0.ENG/linux_x86_64_rhel6/doc/training
 - 96>
ls AppNote_*
AppNote_Built-in_Reporting.pdf                          AppNote_Macro_Handling_in_RedHawk-SC.pdf       AppNote_Scheduler_GUI.pdf
AppNote_CPM.pdf                                         AppNote_Multi-Chip_Analysis_in_RedHawk-SC.pdf  AppNote_SelfHeat_Analysis.pdf
AppNote_CTM_Creation_in_RedHawk-SC.pdf                  AppNote_NPV.pdf                                AppNote_Signal_EM.pdf
AppNote_ClockFX_RedHawk-SC.pdf                          AppNote_Package_Board_Analysis.pdf             AppNote_Time-step_Determination_in_RHSC.pdf
AppNote_Controlling_Macro_Switching_for_Dynamic_Analysis.pdf  AppNote_PowerProfileView.pdf             AppNote_Using_Mapreduce_in_Redhawk-SC.pdf
AppNote_Decap_Handling_in_RedHawk-SC.pdf                AppNote_Presim_Handling_in_RedHawk-SC.pdf      AppNote_VCD_Handling.pdf
AppNote_Diskspace_Management_in_Seascape.pdf            AppNote_RedHawk-SC_License_Management.pdf      AppNote_Vectorless_Scan.pdf
AppNote_Early_Grid_Analysis_and_Robustness_Checks.pdf   AppNote_Resiliency_in_RedHawk-SC.pdf          AppNote_Voltage_Stats_Reporting_in_RedHawk-SC.pdf
AppNote_FSDB_Replay_in_Fullchip_Dynamic_Simulation.pdf  AppNote_Rollup_Analysis.pdf                   AppNote_Worker_Management_in_Redhawk-SC.pdf
AppNote_In-Rush_Analysis.pdf                            AppNote_SEB.pdf                                AppNote_ecos_in_RedHawk-SC.pdf
AppNote_Introduction_to_Heatmaps.pdf                    AppNote_Scaling_Power_in_PowerView.pdf
```



Looking at Explorer Results



Click on explorer icon from console to open Explorer window

Accessing Help From RedHawk-SC Python Shell

```
>>> help(DesignView)
CLASS
DesignView
   View for design data including layout and connectivity.

SEE ALSO
SeaScapeDB.create_design_view

FUNCTIONS
convert_to_id(obj)
  Converts name-based object or iterable to id-based object or list.

convert_to_name(obj)
  Converts id-based object or iterable to name-based object or list.

get_attributes(obj, include_pin_geoms=None, include_phantoms=None, include_clock_gate_pins=None,
include_di_checks_stats=None, only_data_types=None, include_data_types=None)
  Returns a dict of DesignView attributes for the specified object.

get_cell_data_coverage()
  Returns a dict of DesignView DI Summary.
.. .. ..
```

Understanding the API – SeaScapeDB Objects

- SeaScape organizes data by SeaScapeDB objects, the most common of which are:
  - **Layer, Cell, Instance, Pin, Net** , etc
- Each of these objects has both a string and integer representation
- In distributed processing and MapReduce, strings are highly inefficient
- Conversion between string and integer can be done through :
  - <DesignView>.convert_to_name(<object>)
  - <DesignView>.convert_to_id(<object>)

```
>>> dv.convert_to_id(Instance('core0.regfile_data_memory.OAI22_X1_821'))
>>> Instance(89172)
>>> dv.convert_to_name(Instance(89172))
>>> Instance('core0.regfile_data_memory.OAI22_X1_821’)
>>> dv.convert_to_id(Pin('VSS'))
>>> Pin(5)
```

Object Attribute Query Examples

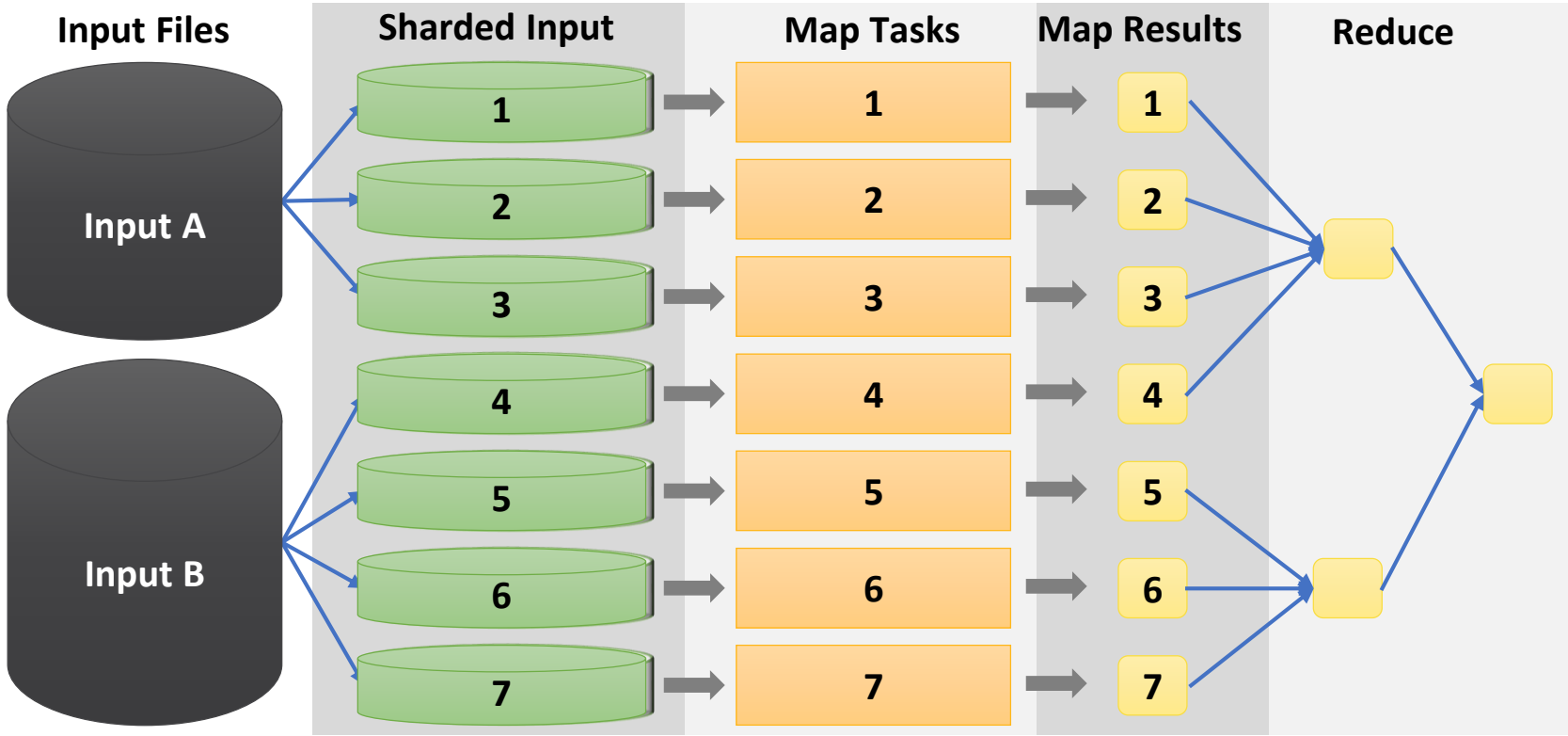
```
>>> dv.get_attributes(Instance("core1.regfile_program_memory.OAI21_X1_1777"))
>>>  {'cell': Cell('OAI21_X2'),
     'cell_id': 135,
     'coord': RealCoord(145.92,659.4),
     'instance_id': 202470,
     'is_leaf': True,
     'rotation': Rotation(RotationType.FS)}
```

```
>>> scn_no_prop_vectorless.get_attributes(Instance("core1.regfile_program_memory.OAI21_X1_1777"))
{'logic': {'clock_instance': False,
          'clock_tree_level': None,
          'clocks_reached': ['default_clock'],
          'frequency': 125000000.0,
          'load_cap': 2.8008853713451e-14,
          'logic_level': -1,
          …… }
```

MapReduce Programming

Introduction to MapReduce (MR)

MapReduce (MR) is a simplified model for processing and generating large data sets with a parallel, distributed algorithm on a cluster in an automated, easy to write, fashion.



Disclaimer: This section is going to cover only basics of MapReduce usage in RedHawk-SC.
For detailed training, please refer to RedHawk-SC Modular Training on “Python APIs & MapReduce”

Hands On MapReduce In SeaScape

- **What type of data am I going to analyze in MapReduce?**
  - Most common data which can be analyzed using MapReduce are sets of **Instances**
  - Also available for MapReduce are:
    - **Circuits** :: the actual extracted circuit
    - **Shapes** :: the geometries in the design
- **What type of data am I going to return?**
  - Most common data set returned from MapReduce is **list** and **dict**
  - Reduce operations are built-in within SeaScape
  - You can write your own custom reduce function also
  - Support for statistics, Waveforms, and others are available
- **What type of reduction will I need for my return?**
  - Any Python object that has '+' operator uses **reduce_add**
  - For reducing dict, **reduce_add_dict** is available
  - Reduce_add_dict is the default reducer for MapReduce (when not specified)

MapReduce Example: Find My Clock Instances In Design

```
def get_clk_inst_select(instances, scn):
    dv = scn.get_related_views(DesignView)[0]
    out = list()
    for inst in instances:
        is_clock = scn.get_attributes(inst)['logic'].get('clock_instance')
        if is_clock:
            out.append(inst)
    return out
```

Serial Execution :

```
data = get_clk_inst_select(dv.get_instances('*'), scn=scn_no_prop_vectorless)
```

Parallel Execution using MapReduce :

```
mm = MapReduce(dv)
mm.map_reduce(dv.get_mr_instances(), partial(get_clk_inst_select, scn=scn_no_prop_vectorless))
data = mm.get()
gp_print("Found ",len(data),' clock instances')
clki_selection = InstanceSelection(dv,data)
gui.add_layer(clki_selection,'clock_instance_map’)
```

Anatomy of a SeaScape MapReduce function

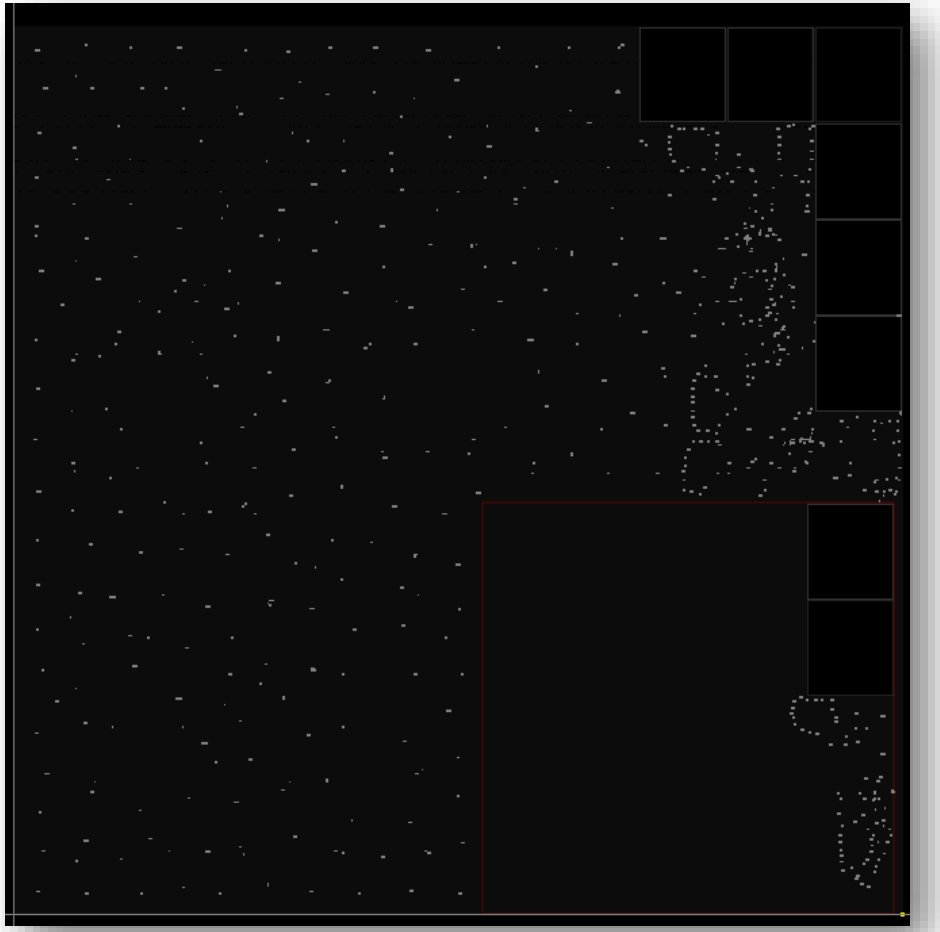
```
def map_func(<mapped data>, <arguments>):
    return <something>
    # map_func is the python job that will run in every worker
    # map_func will do some operations with a return data object (list, dict, etc.)
instances = dv.get_mr_instances()
# Get the sharded or mapped data , here gets sharded instances
# Sharded data can be Instances, Nets, Circuits etc
mm = MapReduce(dv)
# Informs the system what sharded data ( here Instances ) will be used in MapReduce.
# SeaScape internally keeps the instance data in sharded way for MapReduce operations
mm.map_reduce(<mapped data>,partial(<map_func>,<arguments by name>))
# Start the map and reduce operations .
data = mm.get()
# Do Something with the returned data from MapReduce
```

MapReduce Output

```
>>> INFO<USER.000>  Found  697  clock instances
```

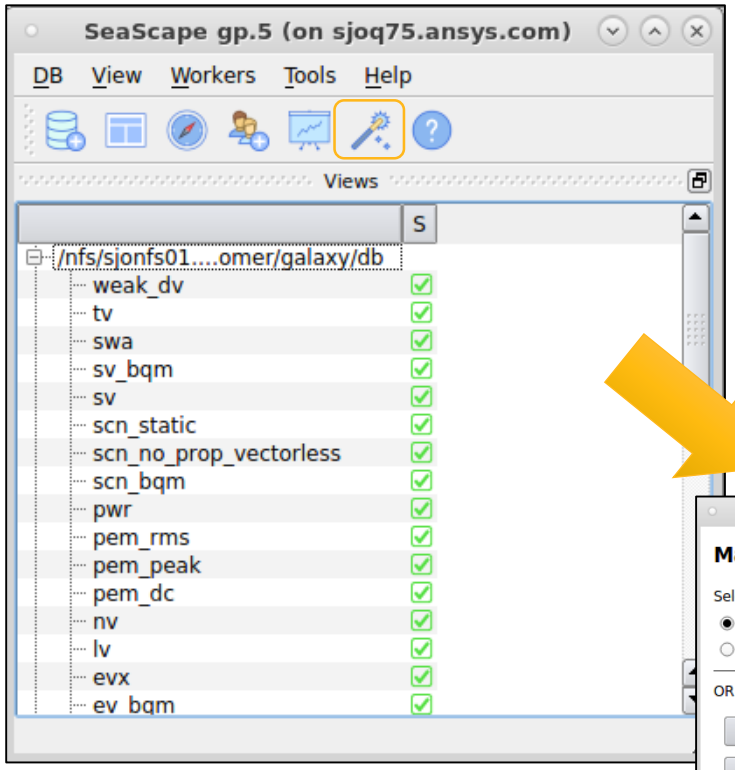
The parallel execution using MapReduce operations is much faster than serial execution on a large design

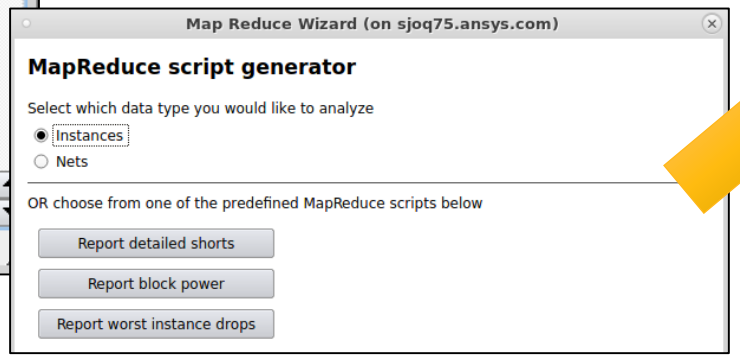
The clock instances from MapReduce are displayed in the GUI

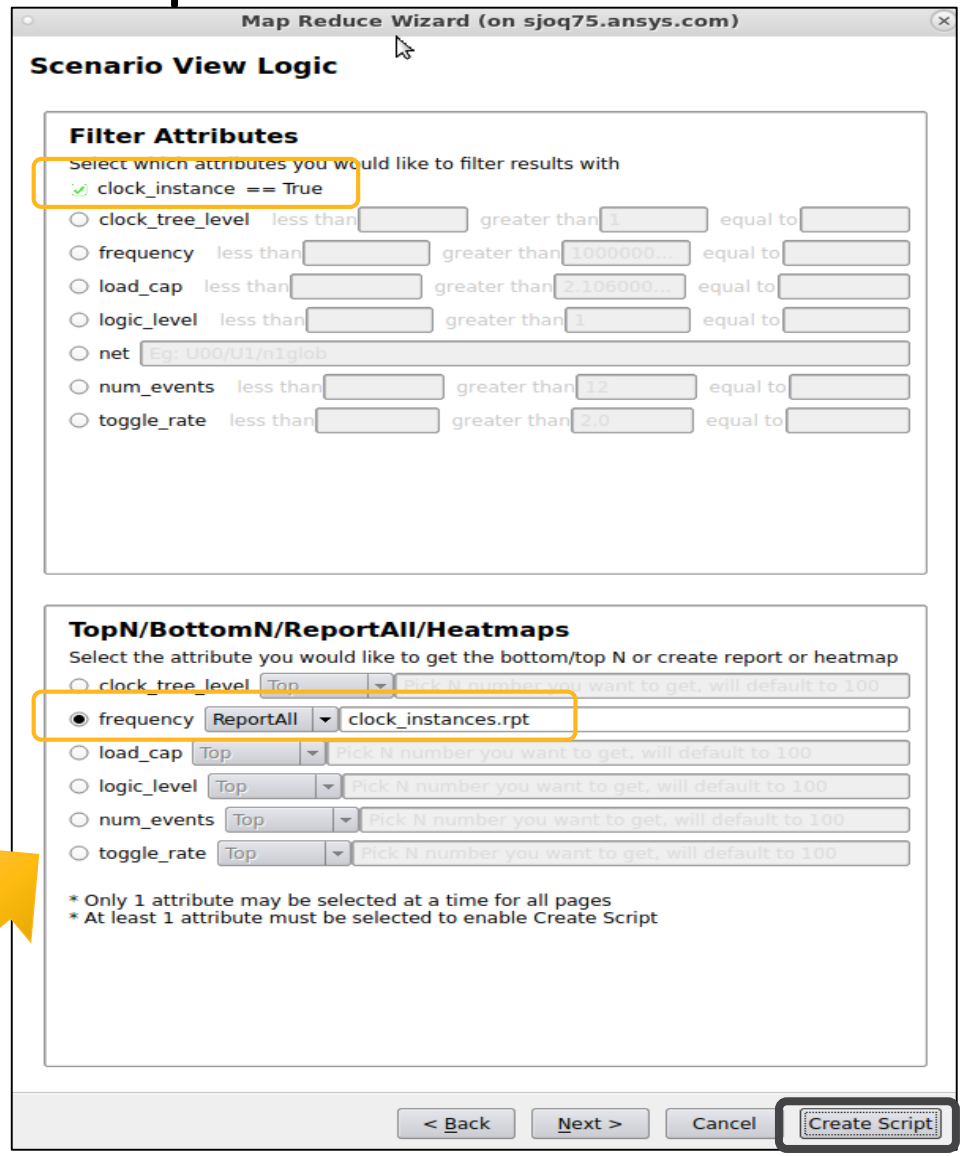


Using MapReduce Wizard To Run Python Scripts

- You don’t need to be a Python expert to run MapReduce script in SC
- Use the MapReduce wizard to generate your Python script for most common use cases







Using MapReduce Wizard To Run Python Scripts

Run script (on sjoq75.ansys.com)

File

Click Here to run the script

```
from gp import *
open_scheduler_window(show_job_text=True)


def dv_filter(ii,dv_list):
    for dv_id, dv in enumerate(dv_list):
        dv_attributes = dv.get_attributes(ii)
        if dv_attributes.get('is_leaf') != True:
            return False
    return True


def scn_filter(ii,scn_list):
    for scn_id, scn in enumerate(scn_list):
        scn_attributes = scn.get_attributes(ii)
        if scn_attributes['logic'].get('clock_instance') != True:
            return False
    return True

def get_ii_frequency(ii,scn_list):
    stats = StdStatsDouble()
    for scn in scn_list:
        scn_attributes  = scn.get_attributes(ii)
        frequency = scn_attributes['logic'].get('frequency')
        if frequency is not None:
            stats.add_sample(frequency)
    if stats.get_count() > 0:
        return (ii,stats.get('avg'))
    else:
        return None
```

Python script is automatically generated by MR Wizard

clock_instances.rpt (/nfs/sjonf

File Edit Tools Syntax Buffers Window H

```
cts_inv_583961390 125000000.0
cts_inv_584361394 125000000.0
cts_inv_584461395 125000000.0
cts_inv_584961400 125000000.0
cts_inv_585261403 125000000.0
cts_inv_585361404 125000000.0
cts_inv_585461405 125000000.0
cts_inv_586361414 125000000.0
cts_inv_587161422 125000000.0
cts_inv_587261423 125000000.0
cts_inv_587761428 125000000.0
cts_inv_587861429 125000000.0
cts_inv_589861449 125000000.0
cts_inv_600161552 125000000.0
cts_inv_600461555 125000000.0
```

Using SeaScape Platform

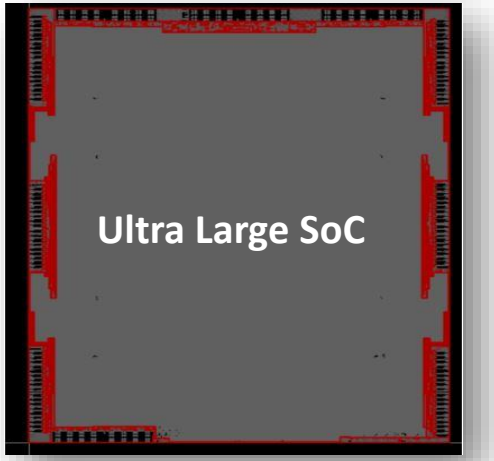
Performance of Tools Using SeaScape Platform

N7
TSMC

775 mm²
Design

54B
Transistors

66B
Nodes



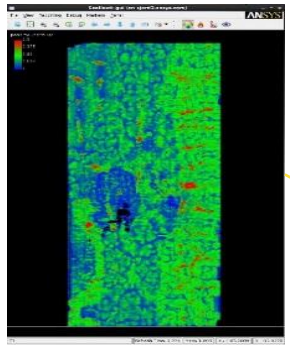
Analysis Type	Static + Dynamic
Total Run-time	**41hrs**
#Workers	**2400**
Peak worker memory	**22GB**

RedHawk-SC delivers reliable big data compute over 100,000+ core-hours

An Example Of Big Data Analytics Application Using SeaScape Platform

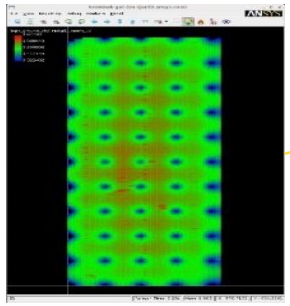
RAIL DATA

CURRENT CROWDING



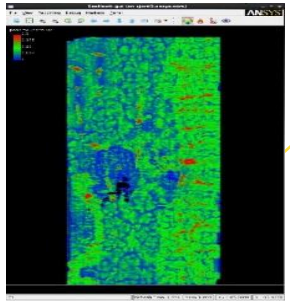
RAIL DATA

RESISTANCE



TIMING DATA

SLACKS



Example Analytics

highlight instances which are timing critical, with high PG resistance and placed in high current crowding area

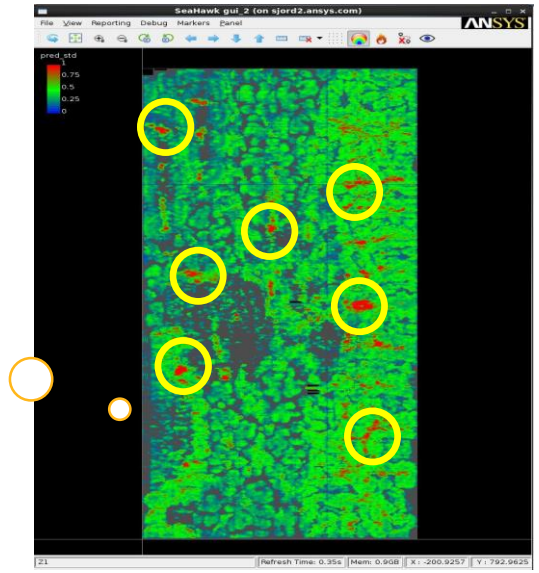
PHYSICAL DESIGN DATA

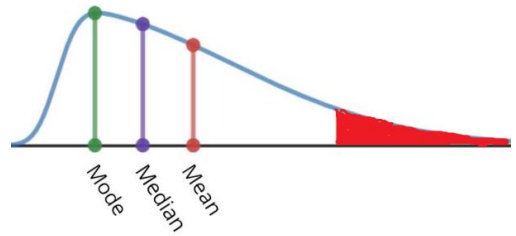
PER LAYER CONGESTION

LIBRARY DATA

CELL VOLTAGE SENSITIVITY

Multi-metric Analytics

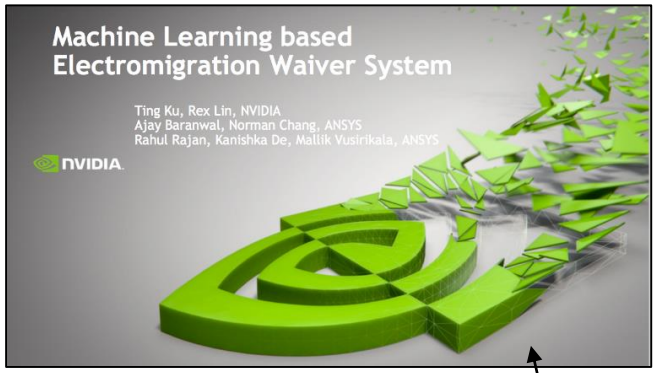


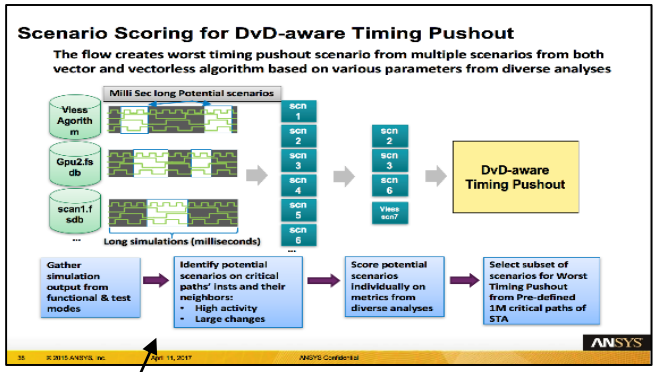


Statistical distribution of outliers from SC data-analytics can help in finding design weakness

SeaScape Platform

Integrated ML Stack for Scientific Computing





EM Waiver System | **Timing Assistant**

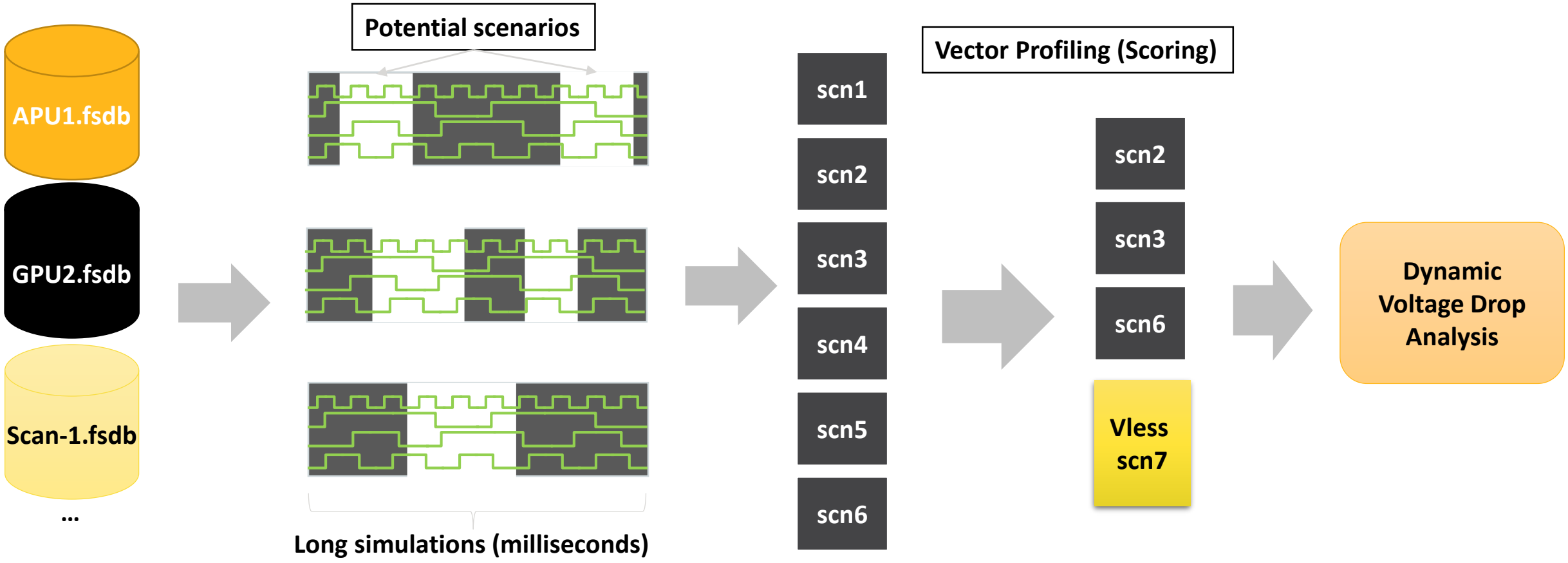
Neural Network / Deep Learning

SeaScape Big Data Platform

Actionable Analytics | **Machine Learning / AI*** | Elastic Compute | Visualization & Debug

** Google TensorFlow Numpy K-Means Clustering, …*

Scenario Scoring – Mix Of Vector + Vectorless Approaches



Gather simulation output from functional & test modes

Identify potential scenarios (High activity, Large changes)

Score potential scenarios individually on metrics

Select subset of scenarios for best overall coverage + Add vectorless


Ansys