Curso de

01. Introducción a ParaView

Michael Heredia Pérez
mherediap@unal.edu.co

Universidad Nacional de Colombia
Sede Manizales

Toda la información aquí presentada es obtenida de la documentación oficial de ParaView https://www.paraview.org/ y páginas y foros en línea

ParaView

- Software Libre con licencia BSD de análisis y visualización de datos científicos multiplataforma, capaz de procesar grandes cantidades de información.
- Es de uso General o *End-User.*
- Contamos con diferentes herramientas:

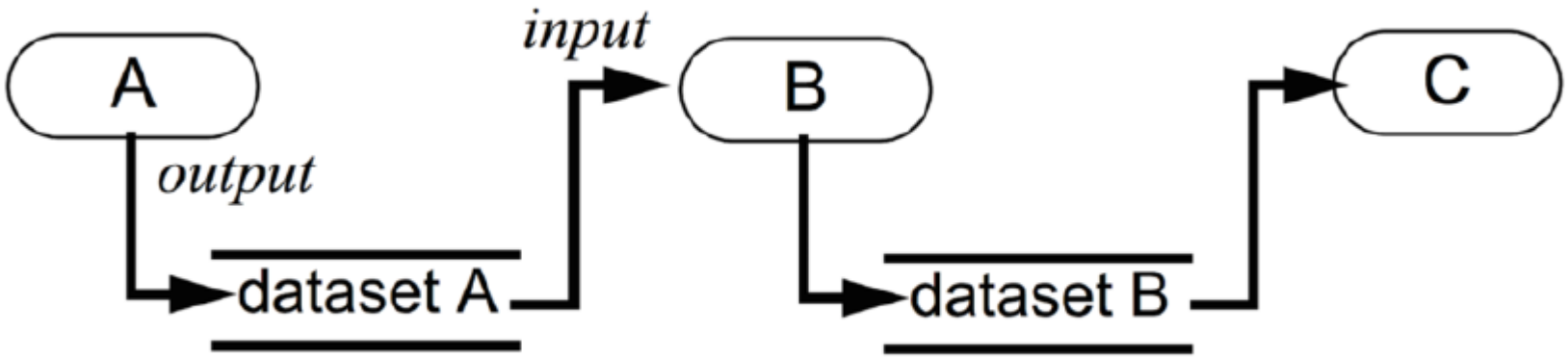
Visualización en la Web	Paraview Web
Análisis In-Situ	Catalyst
Scripting y batch processing	Python
Visualización en grandes pantallas e inmersión 3D	Immersive

Definamos dos conceptos fundamentales:

Visualizar → Convertir datos en imágenes.

Renderizar → Obtener un mejor entendimiento de los datos.

Un flujo de información viaja por un sistema, en el cual es transformada por diferentes algoritmos conocidos como módulos.



Los algoritmos dentro de *ParaView*

Algoritmos → Puertos que reciben información (*input ports*) y puertos que devuelven (*output ports*) la información procesada si la hay.

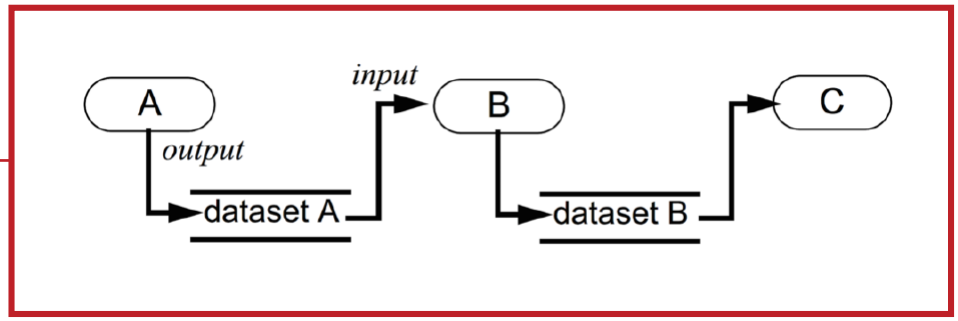
Sources — **Fuentes** → Sin entradas pero una o más salidas. — Readers

Sinks — **Sumideros** → Una o más entradas pero ninguna salida. — Writers

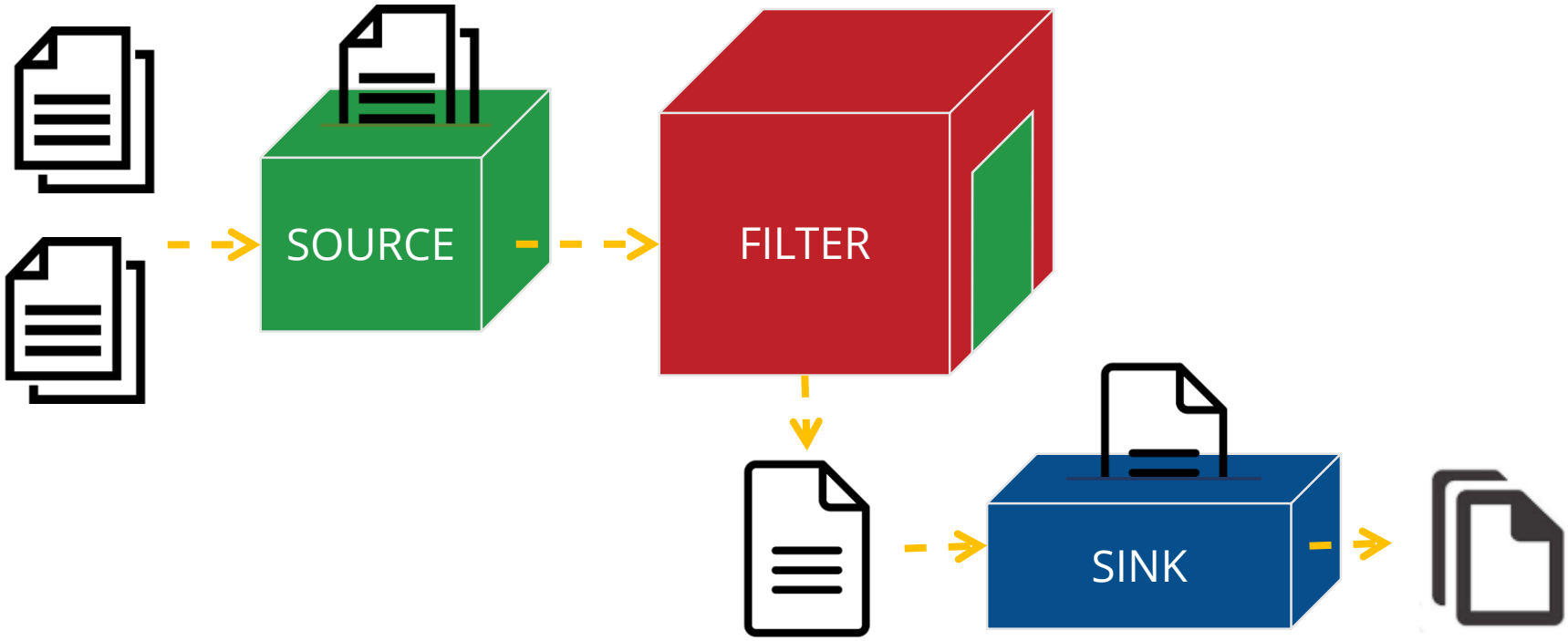
Filters — **Filtros** → Muy variables, con propiedades entre *Sources* y *Sinks.*

El Pipeline

Es la agrupación de algoritmos, el canal por donde fluye la información



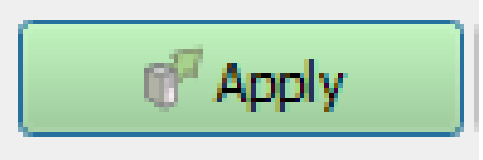
Los algoritmos dentro de *ParaView*



Los ejecutables de *ParaView*

`paraview`	Interfaz gráfica GUI a la que nos referimos cuando decimos ParaView.
`pvpython`	Intérprete de Python que corre los scripts de ParaView hechos en Python de forma interactiva.
`pvbatch`	Igual que `pvpython` pero internamente, encargado del batch processing y con capacidad de correr en paralelo.
`pvserver`	Análisis, procesamiento y renderizado remoto desde el computador en un servidor de recursos HPC.
`pvdataserver` y `pvrenderserver`	Configuración en la cual el `pvserver` se separa para cumplir dos funciones específicas por separado.

El botón Apply



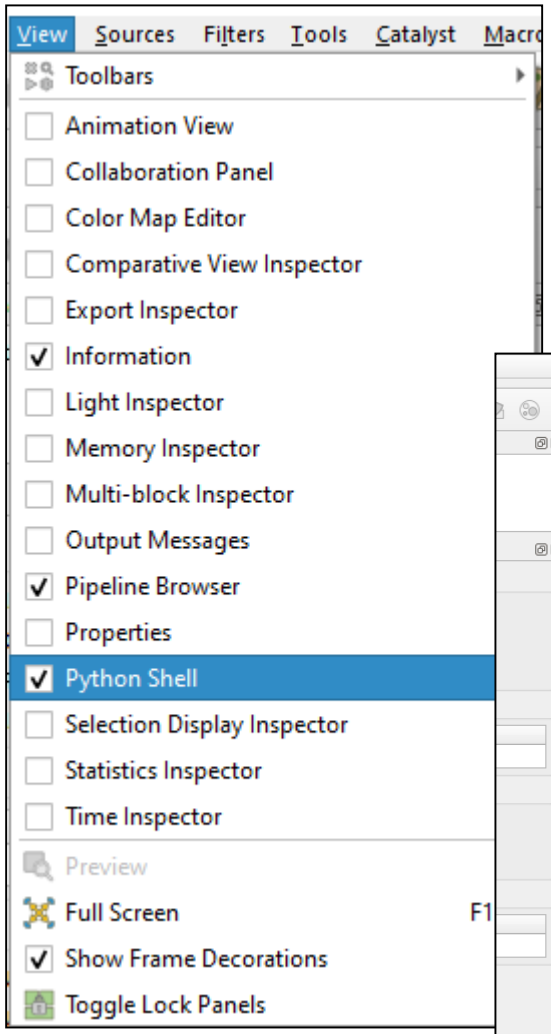
Cada fuente cuenta con determinadas propiedades las cuales pueden ser modificadas; al modificarlas se están haciendo cálculos internos y reajustes, los cuales representan un gasto computacional que ParaView optimiza al permitir hacer todas las modificaciones posibles antes de ejecutarlas.

Apply envía la orden de ejecutar las modificaciones en paraview; sin embargo, al modificar propiedades de visualización o de renderizado no es necesario, ya que estas no consumen tanta máquina.

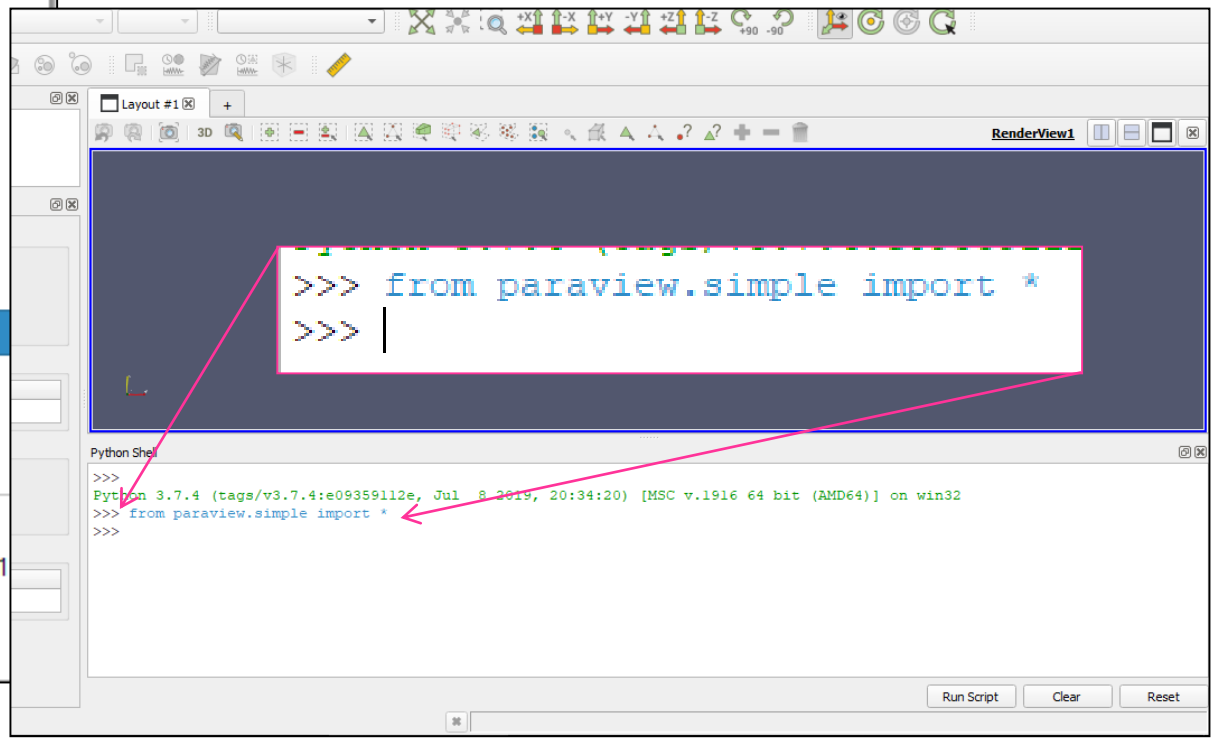
Se puede activar una función, desde la configuración, en la cual el Apply es automático.

Autorizar las modificaciones es necesario también desde pvpython mediante los comandos Show(), Render() y UpdatePipeline(), aunque la ocurrencia de actualizar es diferente a paraview.

Acceder al Python Shell



Para entrar al *Python Shell* hay que dirigirse al menú *View*. Automáticamente se cargará la librería `paraview.simple`



Error de código en el *ParaViewGuide*

La rutina para obtener la instancia de la entrada de un filtro en la guía está mal planteada, se obtiene un error:

```
>>> Hide(shrinkInstance)
Traceback (most recent call last):
  File "<console>", line 1, in <module>
  File "C:\Program Files\ParaView 5.8.1-Windows-Python3.7-msvc2015-64bit\bin\Lib\site-packages\paraview\simple.py", line
542, in Hide
    controller.Hide(proxy, proxy.Port, view)
  File "C:\Program Files\ParaView 5.8.1-Windows-Python3.7-msvc2015-64bit\bin\Lib\site-packages\paraview\servermanager.py",
line 1322, in __getattr__
    return getattr(self.SMProperty, name)
AttributeError: 'paraview.modules.vtkRemotingServerManager.vtkSMInp' object has no attribute 'Port'
```

Según el foro de ParaView (link) la forma correcta de hacerlo es como se muestra a continuación:

```python
# Definir el filtro como un objeto.
filtro = Shrink()

Show()      # Se vuelve activo.
Render()    # Se renderiza.

# Se busca la instancia de la entrada de datos del filtro con el método
# .Input
entrada_filtro = filtro.Input

# Se oculta la fuente de la esfera.
Hide(entrada_filtro)

Render()    # Se renderiza.
```