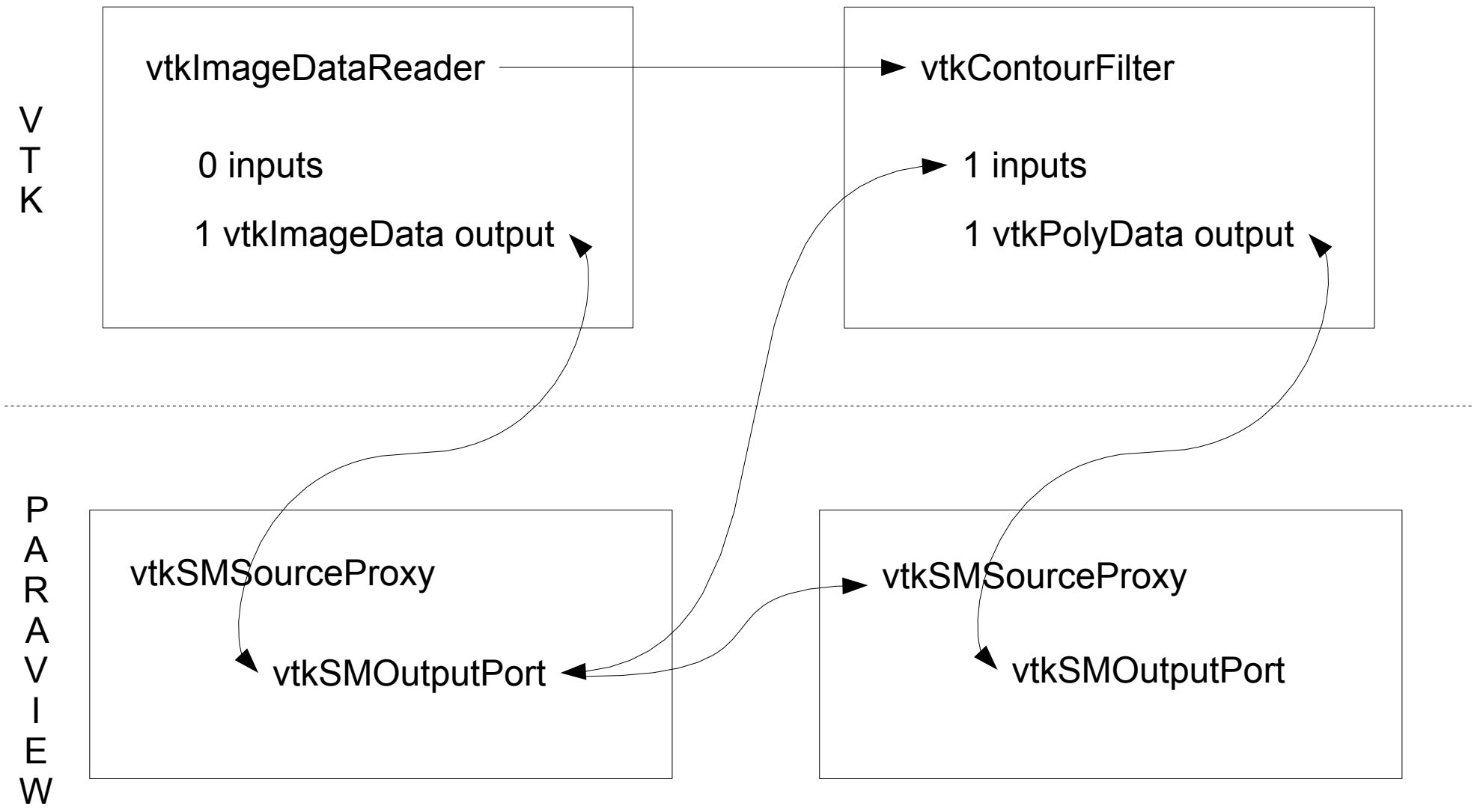
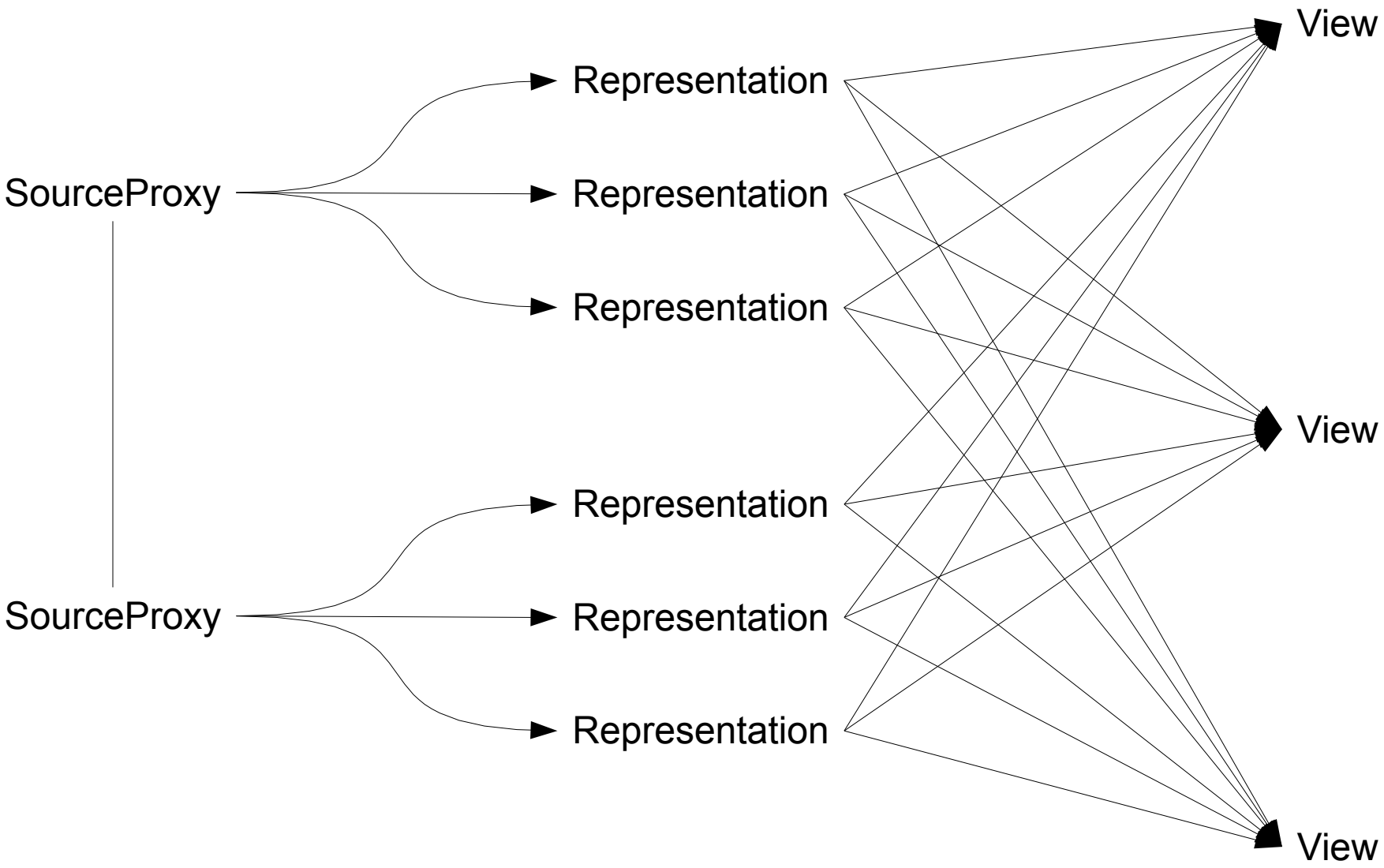


The ParaView client controls vtkAlgorithms that live on the server through vtlSMSourceProxies. For each output of an algorithm, ParaView adds a vtkSMOutputPort to the SourceProxy for the algorithm. ParaView uses SMOutputPorts to represent connections between algorithms, and can use them to update the pipline. For example, to gather meta-information.

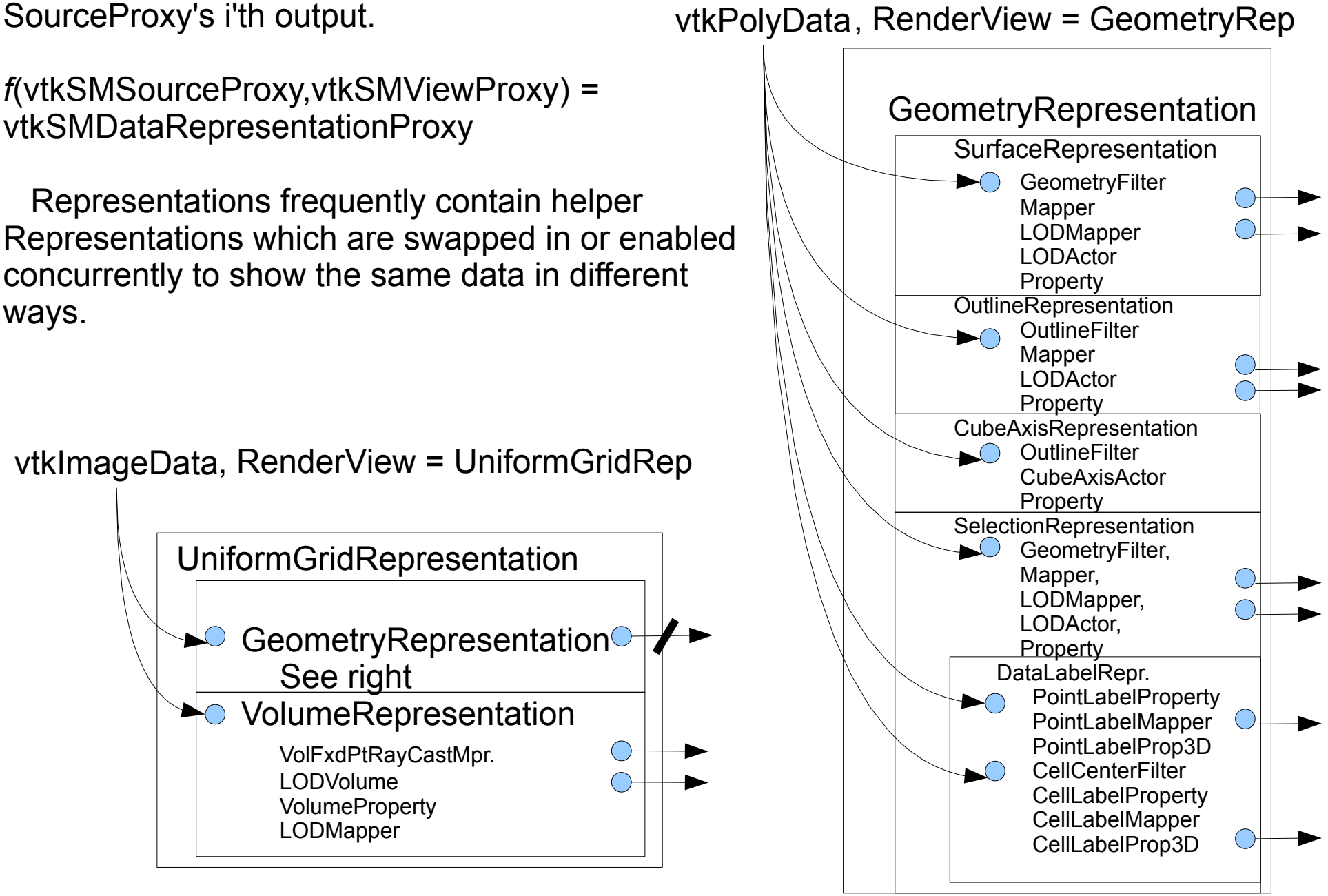


ParaView creates many rendering pipelines to show every SourceProxy in every View. The rendering pipeline is managed by a SMRepresentationProxy, which controls VTK objects such as Mappers, Actors and Properties. ViewProxies controls VTK objects such as Cameras, Renderers, and RenderWindows

The exact type of Representation is a function of the View and the Data Type produced by the SourceProxy's i'th output.

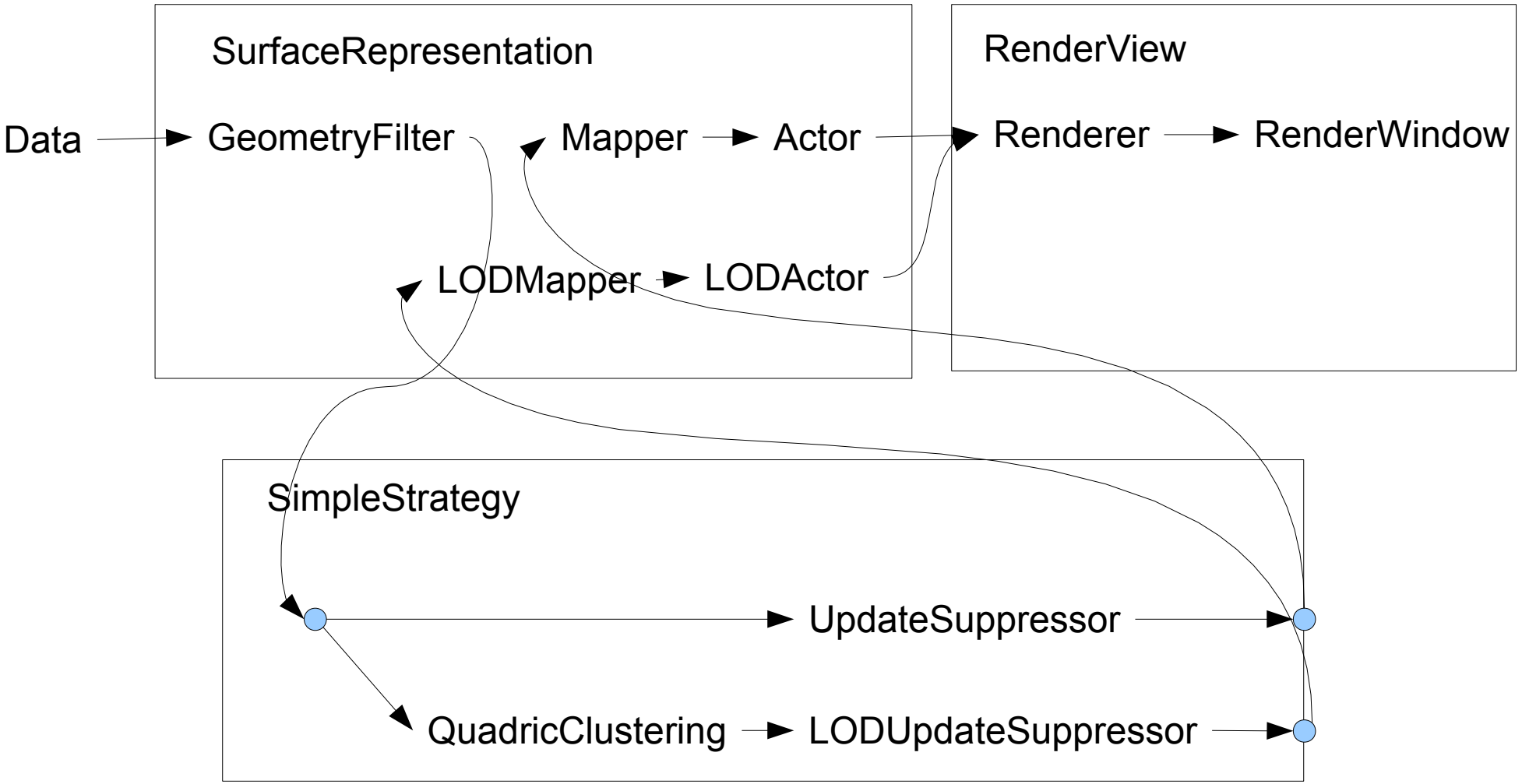
f(vtkSMSourceProxy,vtkSMViewProxy) = vtkSMDataRepresentationProxy

Representations frequently contain helper Representations which are swapped in or enabled concurrently to show the same data in different ways.



Representations contain Strategies. Strategies encapsulate the complexity of transporting data from the producing data server pipeline to the rendering pipeline(s). Depending on the data type and parallel configuration, the View will create different types of Strategy for any given Representation.

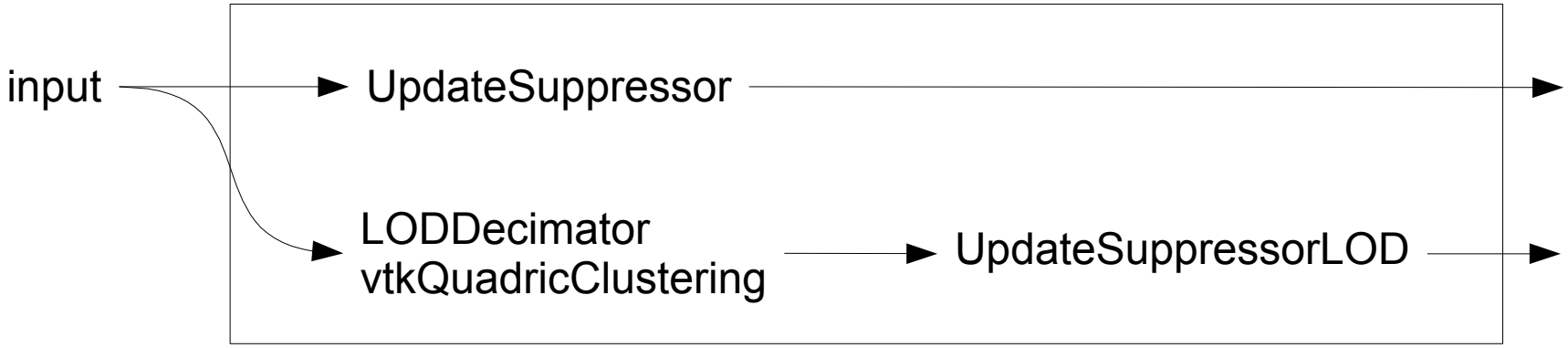
Different portions of the strategy will be active on the client, render server, and data server. Generally UpdateSuppressor::ForceUpdate is called to manually make the segmented pipeline flow because the normal vtk pipeline update model does not run across processors.



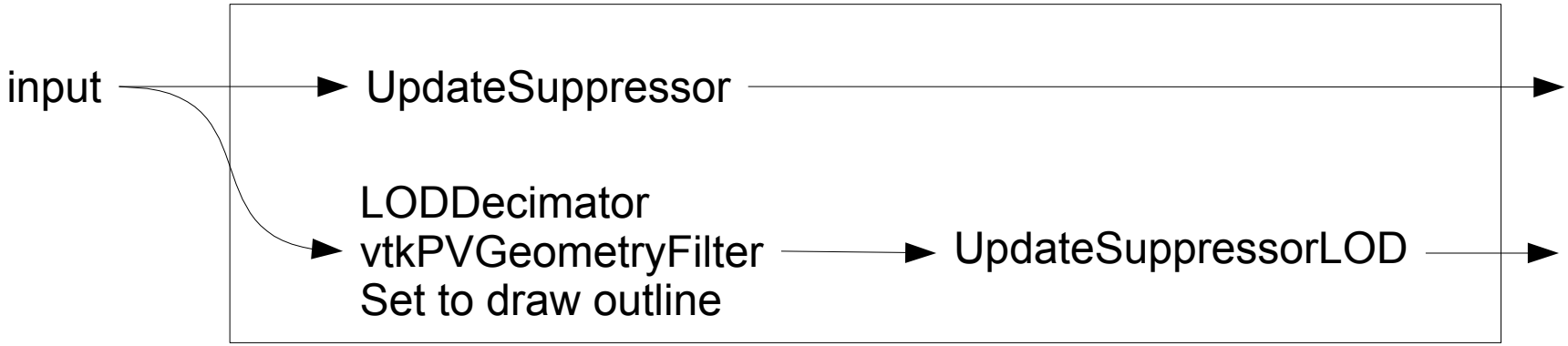
The following diagrams document the pipelines that exist within strategies. Use the table to lookup the diagram that corresponds to a given parallel configuration, process within that configuration, view type, display mode, and data type.

	SURFACE PolyData, UnstructuredGrid	VOLUME UniformGrid	VOLUME UnstructuredGrid	SLICE ImageData
Builtin Client	1	1	2	3
Parallel DataServer	4a	5	To be announced (haven't finished diagramming it)	TBA
Parallel RenderServer	4b	5	TBA	TBA
Parallel Client	4c	5	TBA	TBA

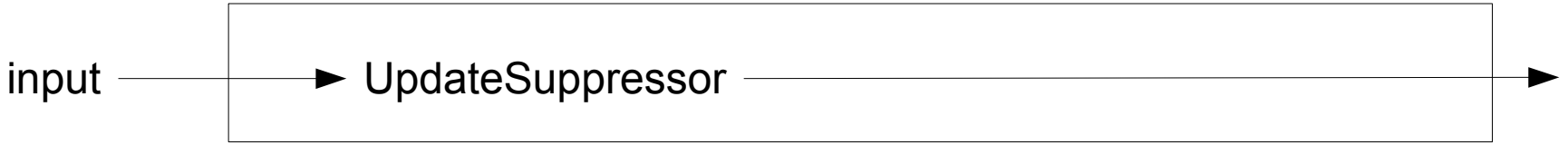
1 SimpleStrategy – used when connected to builtin server.
As configured for surface rendering and uniform grid volume rendering



2 SimpleStrategy – As configured for unstructured grid volume rendering

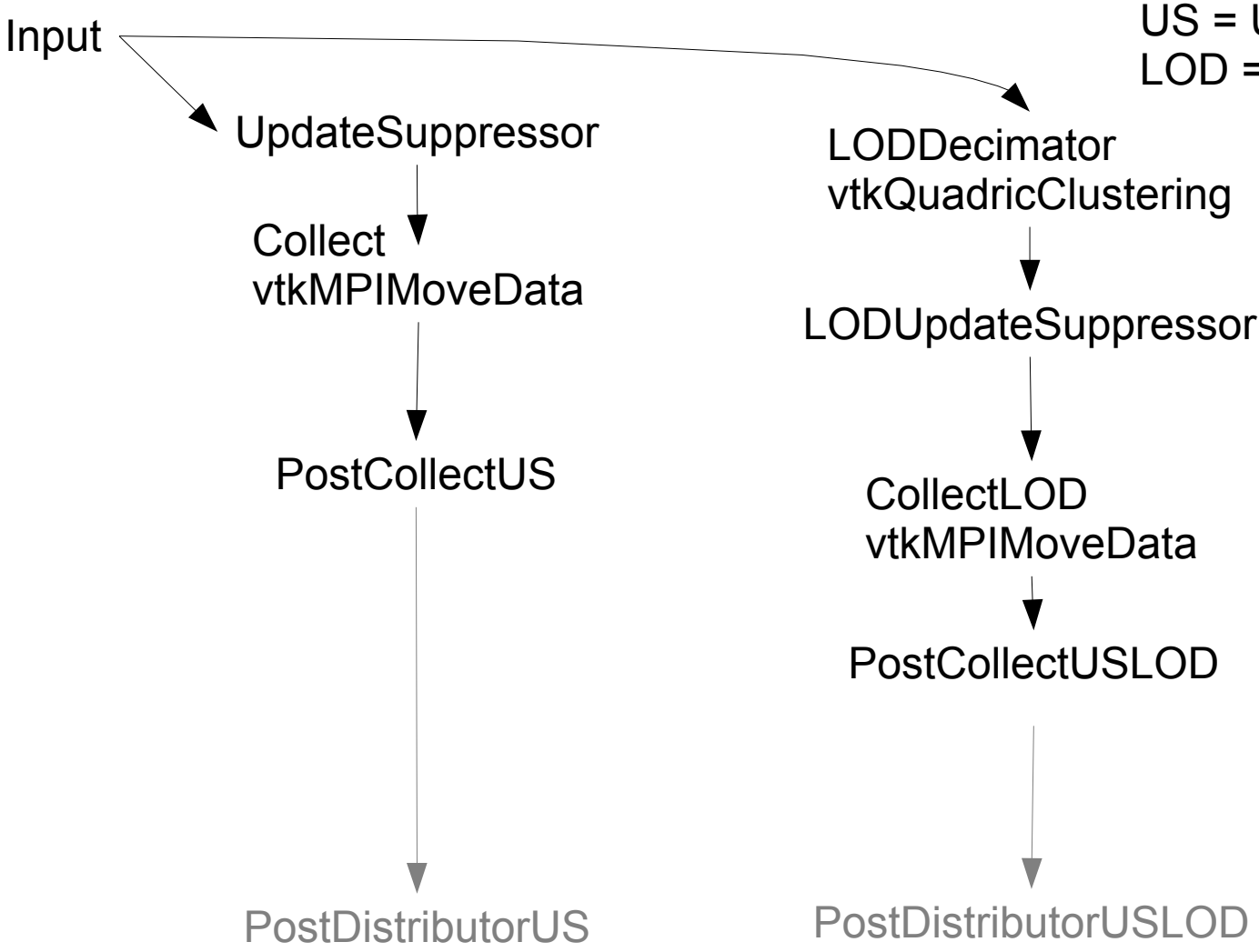


3 SimpleStrategy – As configured for slice rendering, which lacks LOD



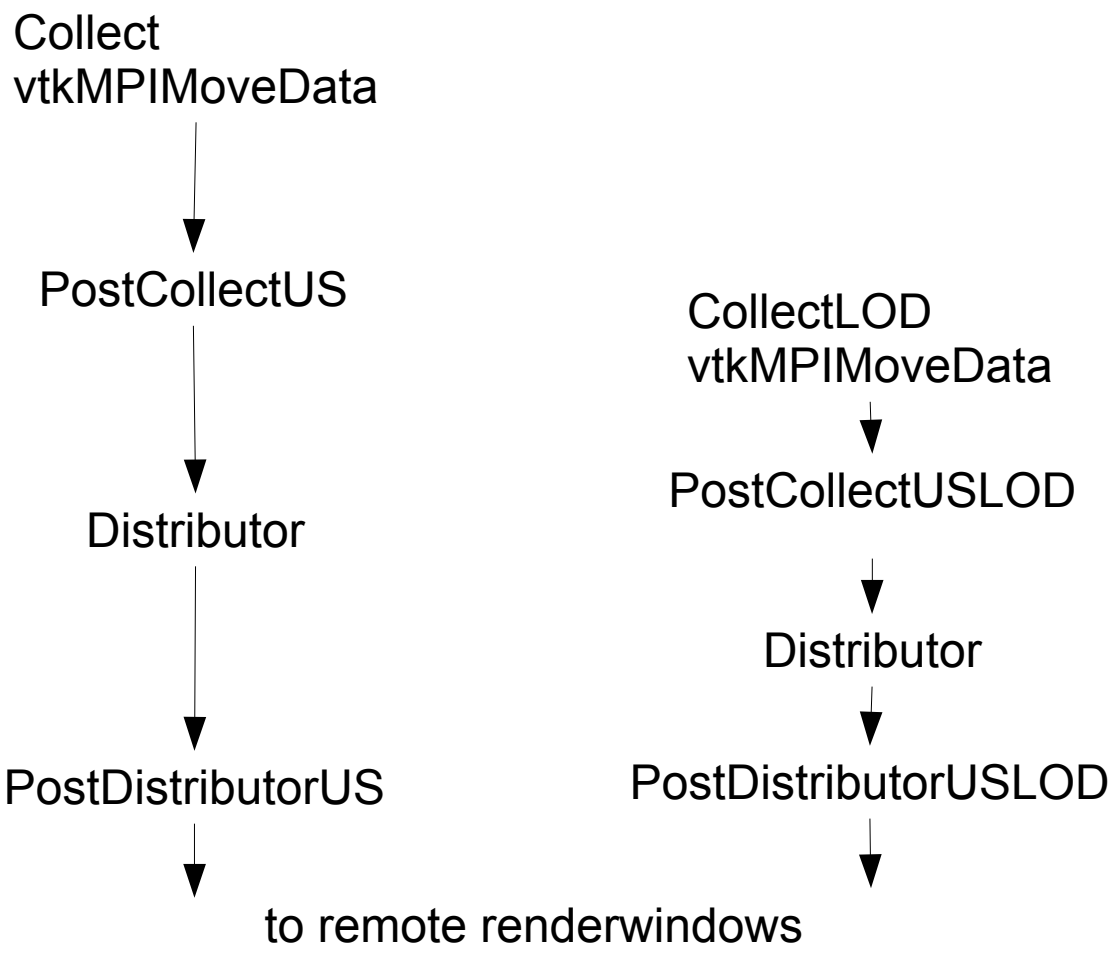
4a SimpleParallelStrategy – used when connected to a remote server
As configured for surface rendering of poly data input.
This portion runs on DataServer

Collects send data over TCP
to RS and/or Client.
US = UpdateSupressor
LOD = Level of Detail



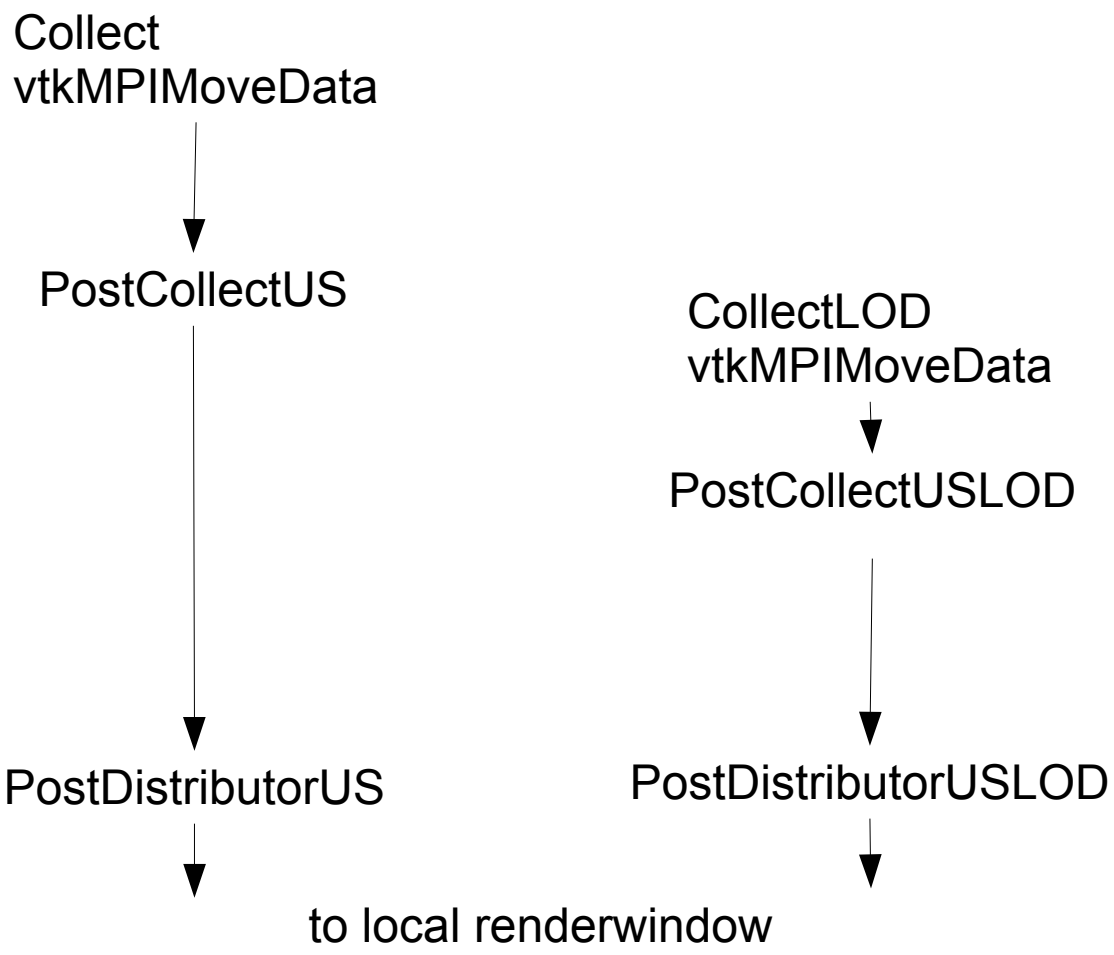
4b SimpleParallelStrategy – This portion that runs on DataServer

Collects get data from data server over TCP.
Distributors do front to back ordering for opacity function.
They share a common KdTree.



4c SimpleParallelStrategy – This portion runs on Client

Collects get data from data server over TCP.



5 SimpleParallelStrategy – As configured for volume rendering of unstructured grid data in parallel configurations

The actual pipelines are same a 4a,b,c, except full resolution collect filters output type are all configured to prodyuce unstrutured grid instead of polydata