JETSCAPE Framework

Software

Joern Putschke and Ebrahim Khalaj
Software Design Working Group
January 09, 2019

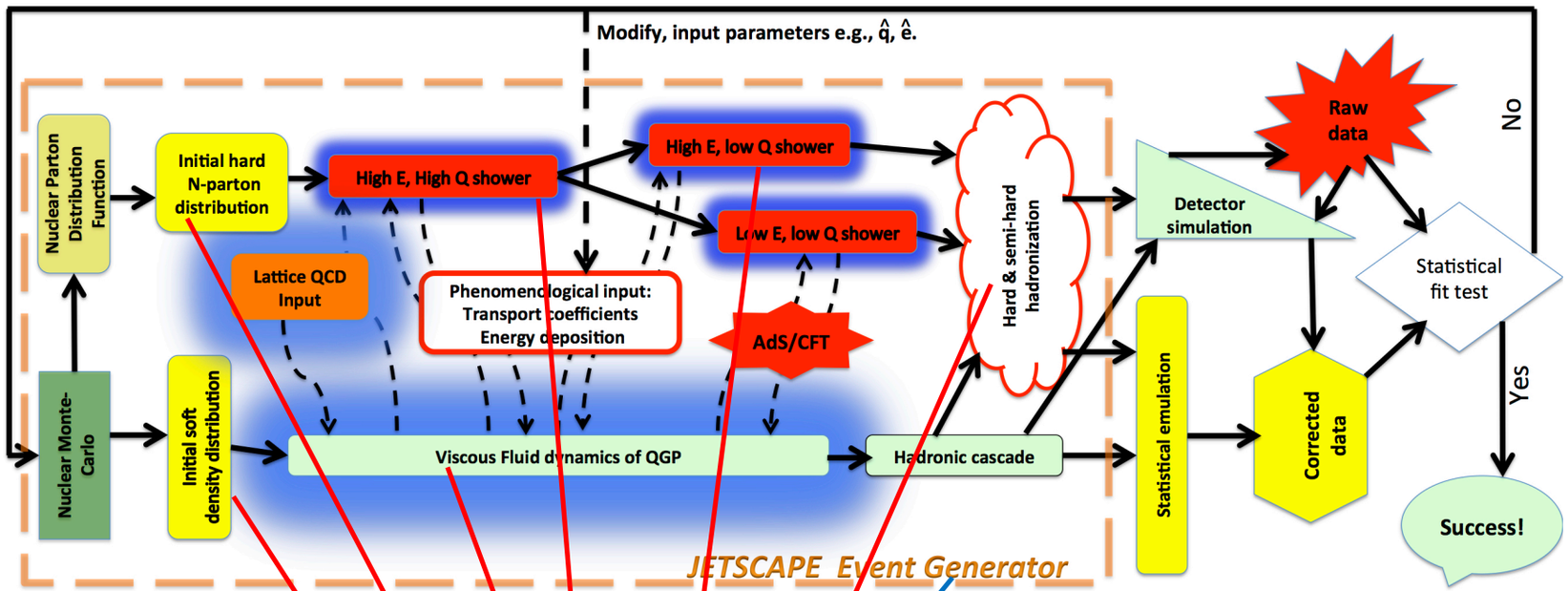
WAYNE STATE UNIVERSITY

Our Code Is Public!

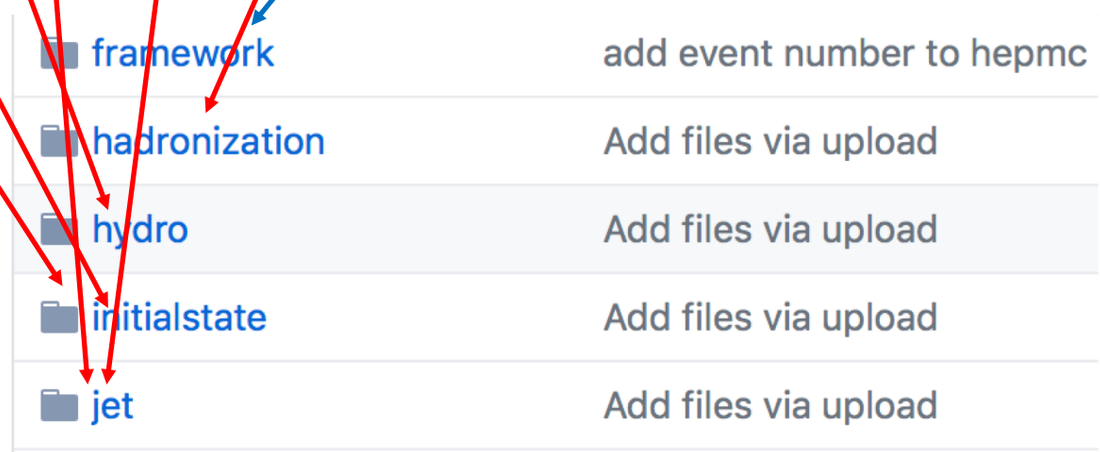
- On GitHub: https://github.com/JETSCAPE/JETSCAPE
  - More than 17KLOC of C++
  - 1000+ commits and counting (development repository)
  - 200+ views every two weeks
  - 20+ clones every two weeks
  - 11 forks so far
- GitHub associated website: https://jetscape.github.io/JETSCAPE/

Directory Structure

- One to one mapping to initial design



- Under “src” directory
  - Framework code → place holders
  - Physics modules → fill in
  - Framework is decoupled from Physics modules



External Packages

- Initial State: **Trento** [by Jonah E. Bernhard, J. Scott Moreland, Steffen A. Bass]
    - Well established code package from Duke University
- Initial Hard Scattering: **Pythia**
- Free Streaming module [by Derek Everett]
    - From Ohio State University
- Hadronization: **Pythia**
- Hydrodynamics: **MUSIC** [by Gabriel Denicol, Charles Gale, Sangyong Jeon, Matthew Luzum, Jean-François Paquet, Björn Schenke, Chun Shen]
- MUSIC and Free Streaming modules are not part of JETSCAPE code base, need to be cloned

Why a Task-Based Framework?

- To simulate all aspects of the collision of the ions
  - From the initial overlap
  - To the explosive expansion and evaporation to conventional matter
- Our solution: Modular, task-based framework
  - Experts on one aspect just reuse modules from other aspects
  - Users provide code of their expertise

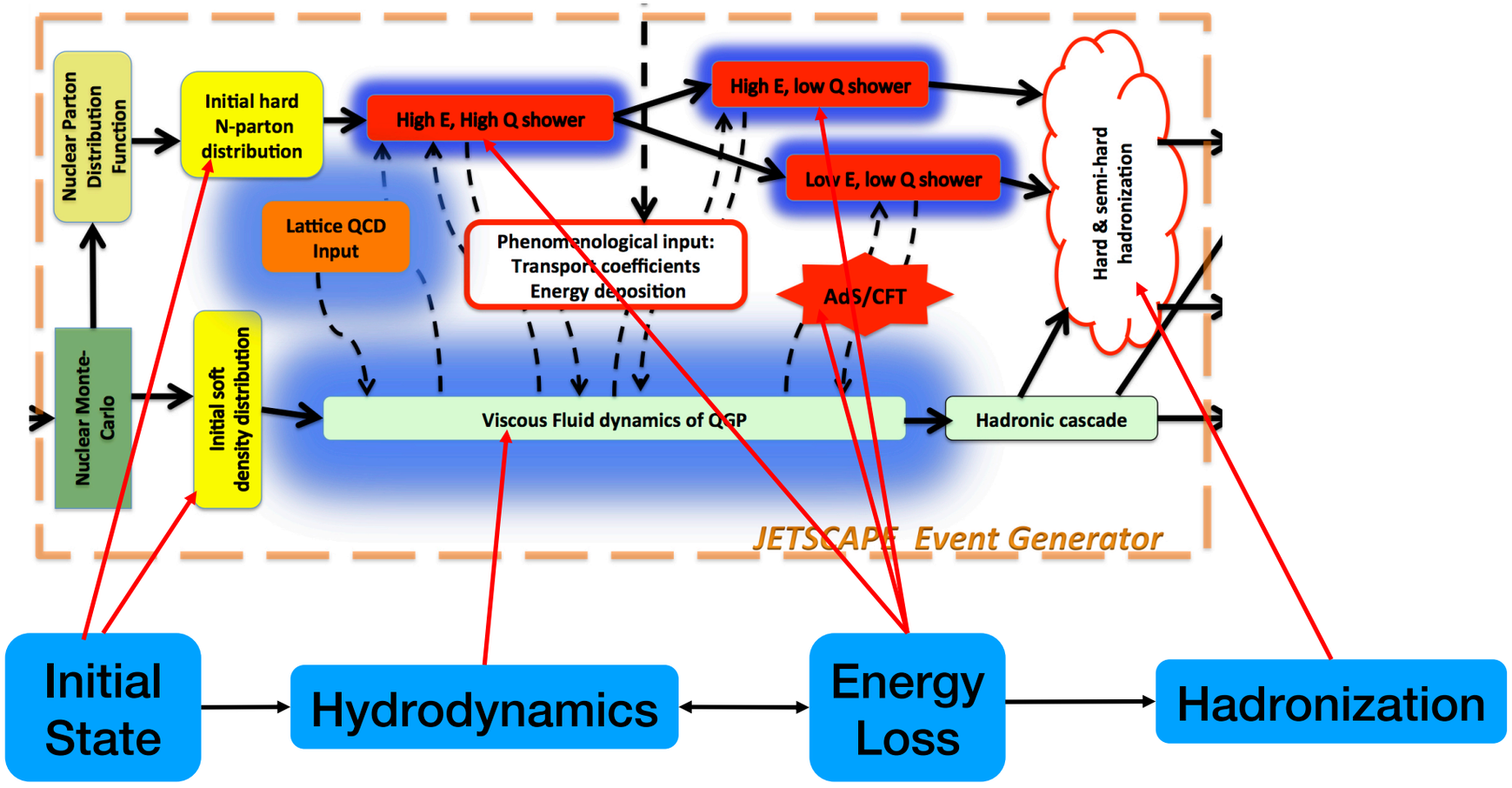
User's responsibility

  - Framework knows how to run users code

Framework's responsibility

Framework Workflow

- Works in a natural way, based on the physics
- Flow of the simulation matches the mental picture of what is happening in the collider



Framework Responsibility

- Defines data structures

  JetScapeParticleBase PartonShower

- Registers modules corresponding to different aspects of the collision as tasks

  JetScapeTask

- Knows how to initialize, execute and finish tasks

  JetScapeModuleBase

- Knows the order of execution and data exchanges

  JetScapeSignalManager

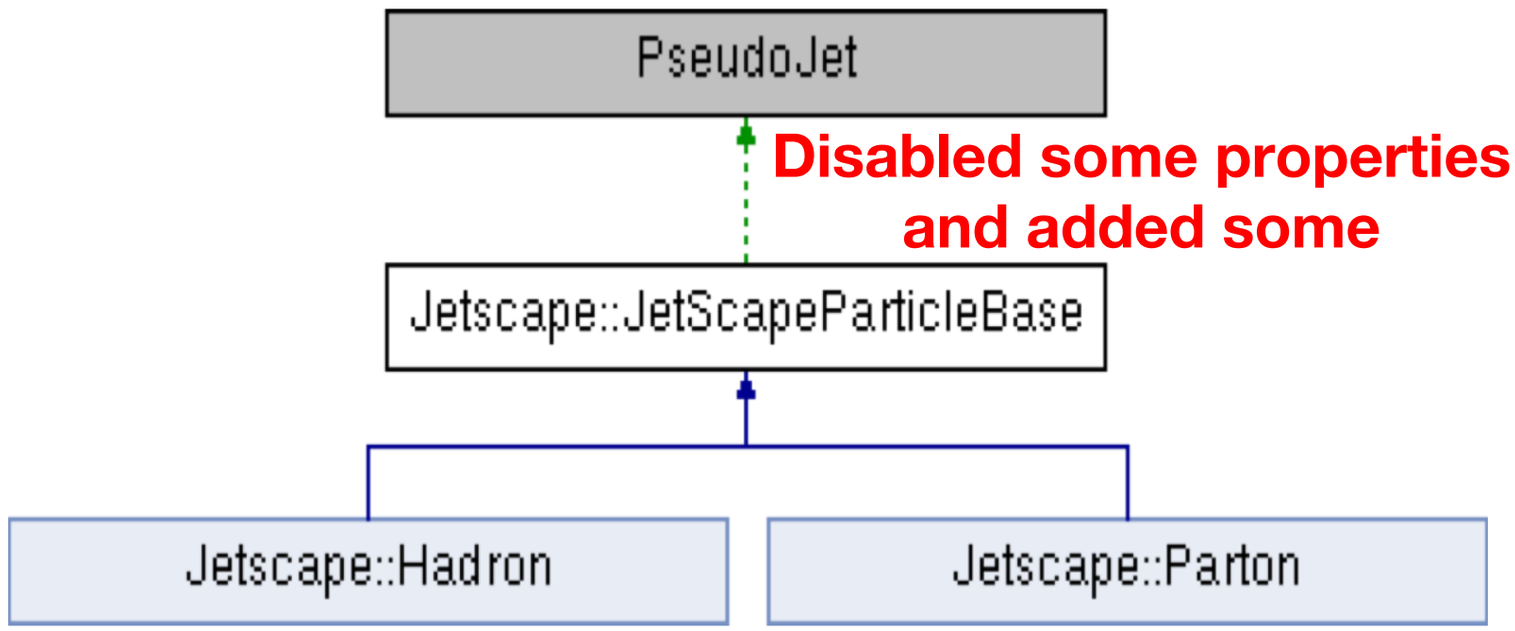
- Provides different kinds of output for analysis

  JetScapeWriter

  JetScapeReader

Data Structures

- Class **JetScapeParticleBase**
  - The base class for all the JETSCAPE particles
  - Privately inherits from FastJet PseudoJet and has
    - PID and rest mass
    - A location (4-vector)
    - Label and status
- Derived classes so far:
  - Parton and Hadron



Parton Class

- Properties are:

We define getter and setter Methods for the properties

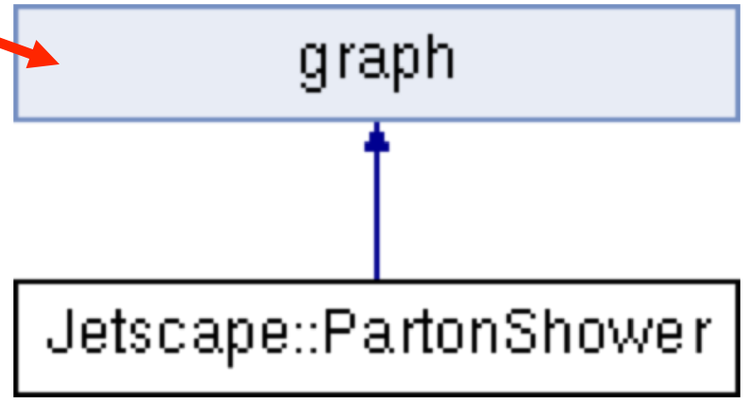
```
protected :
  double mean_form_time_  ; ///< Mean formation time
  double form_time_       ; ///< event by event formation time
  unsigned int Color_      ; ///< Large Nc color of parton
  unsigned int antiColor_  ;///< Large Nc anti-color of parton
  unsigned int MaxColor_    ; ///< the running maximum color
  unsigned int MinColor_    ; ///< color of the parent
  unsigned int MinAntiColor_; ///< anti-color of the parent
```

- Can be created:

```
Parton (int label, int id, int stat, const FourVector& p, const FourVector& x);
Parton (int label, int id, int stat, double pt, double eta, double phi, double e, double* x=0);
Parton (const Parton& srp);
```

Data Structures

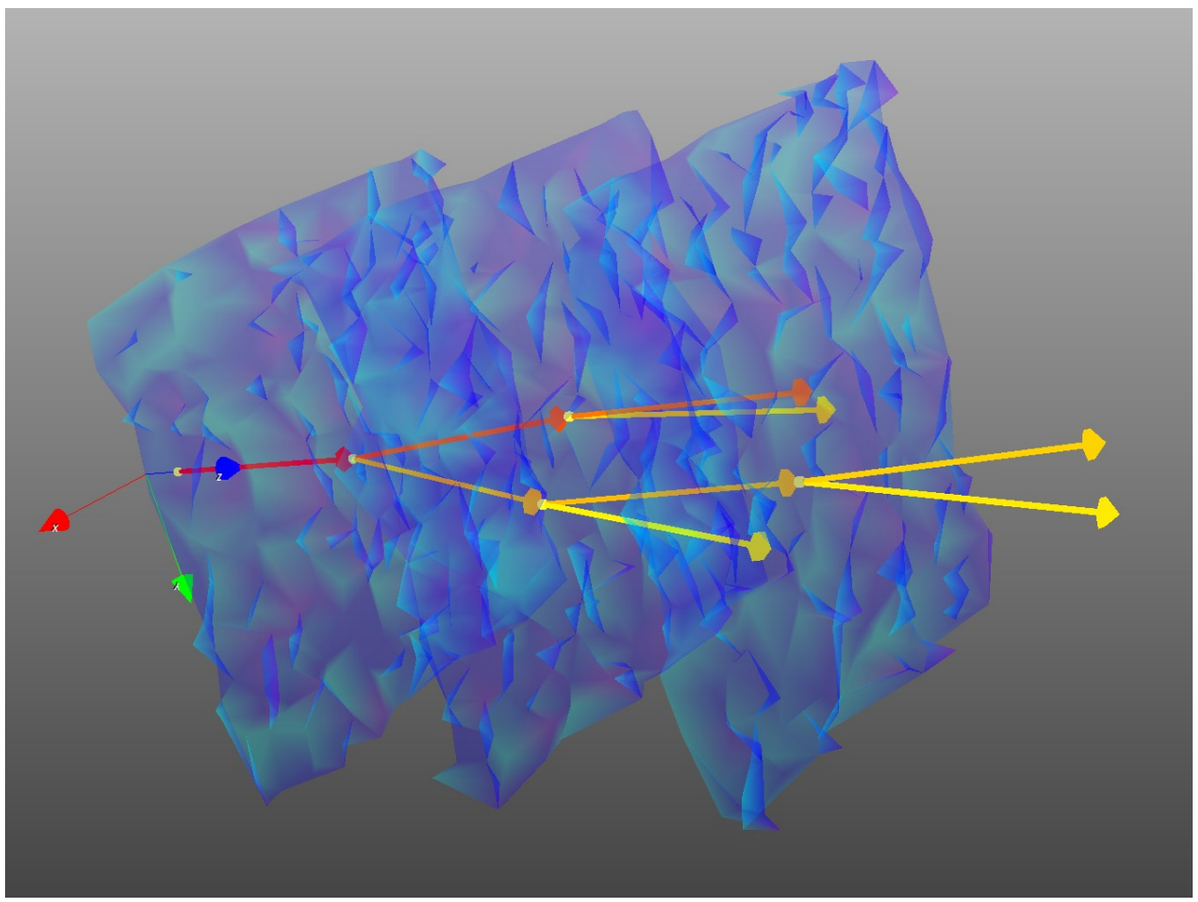
GTL Graph Template Library
[https://github.com/rdmpage/graph-template-library]



- Class **PartonShower**
- Models parton showering as a **Graph**
  - Partons are the **edges**
  - Partons split at **vertices**

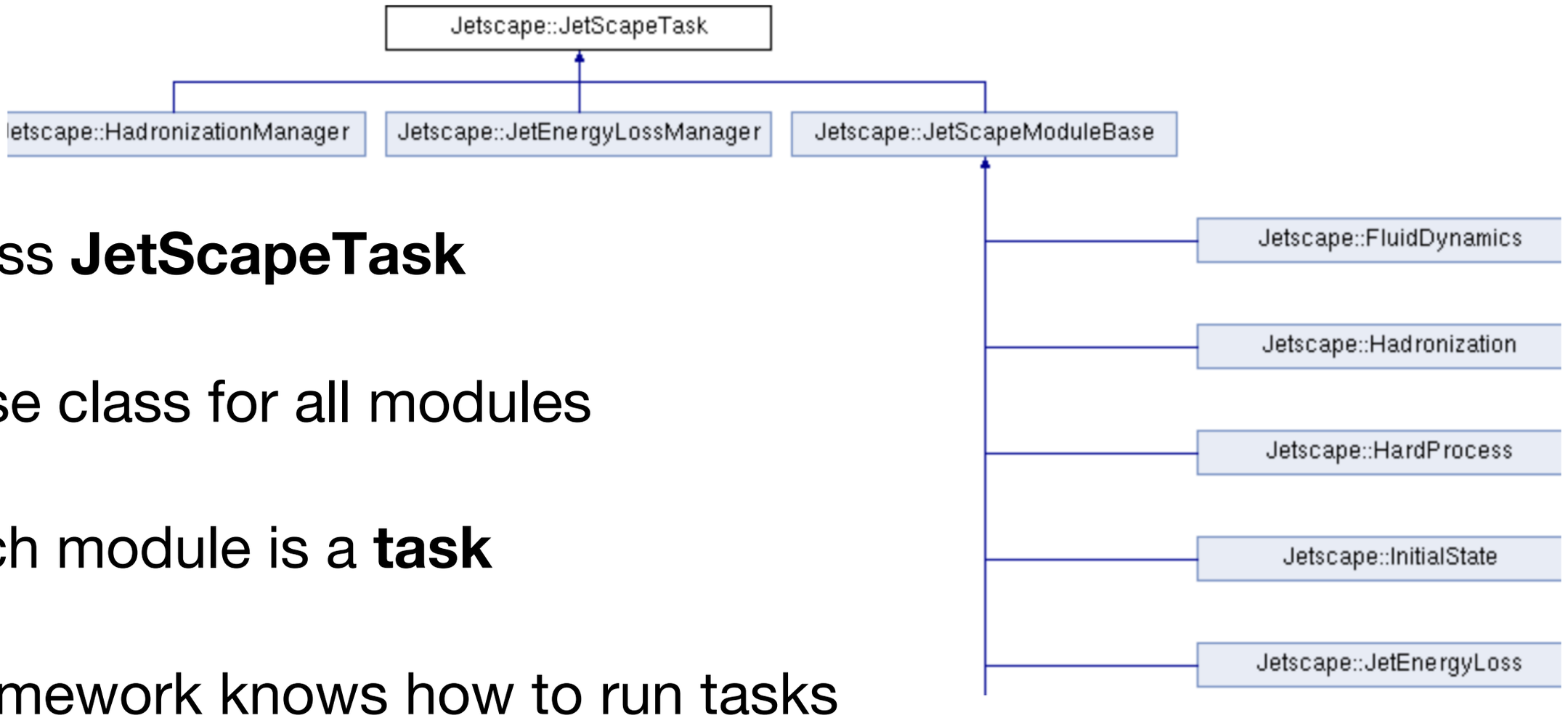
int	**GetNumberOfParents** (int n)
int	**GetNumberOfChilds** (int n)
ared_ptr< **Parton** > >	**GetFinalPartons** ()
< fjcore::PseudoJet >	**GetFinalPartonsForFastJet** ()
int	**GetNumberOfPartons** () const
int	**GetNumberOfVertices** () const

Provides functionalities to query shower



One Framework for All Aspects of the Collision

- Need to have a unified interface for all the aspects



- Class **JetScapeTask**
- Base class for all modules
- Each module is a **task**
- Framework knows how to run tasks

How Does JETSCAPE Framework Run Tasks?

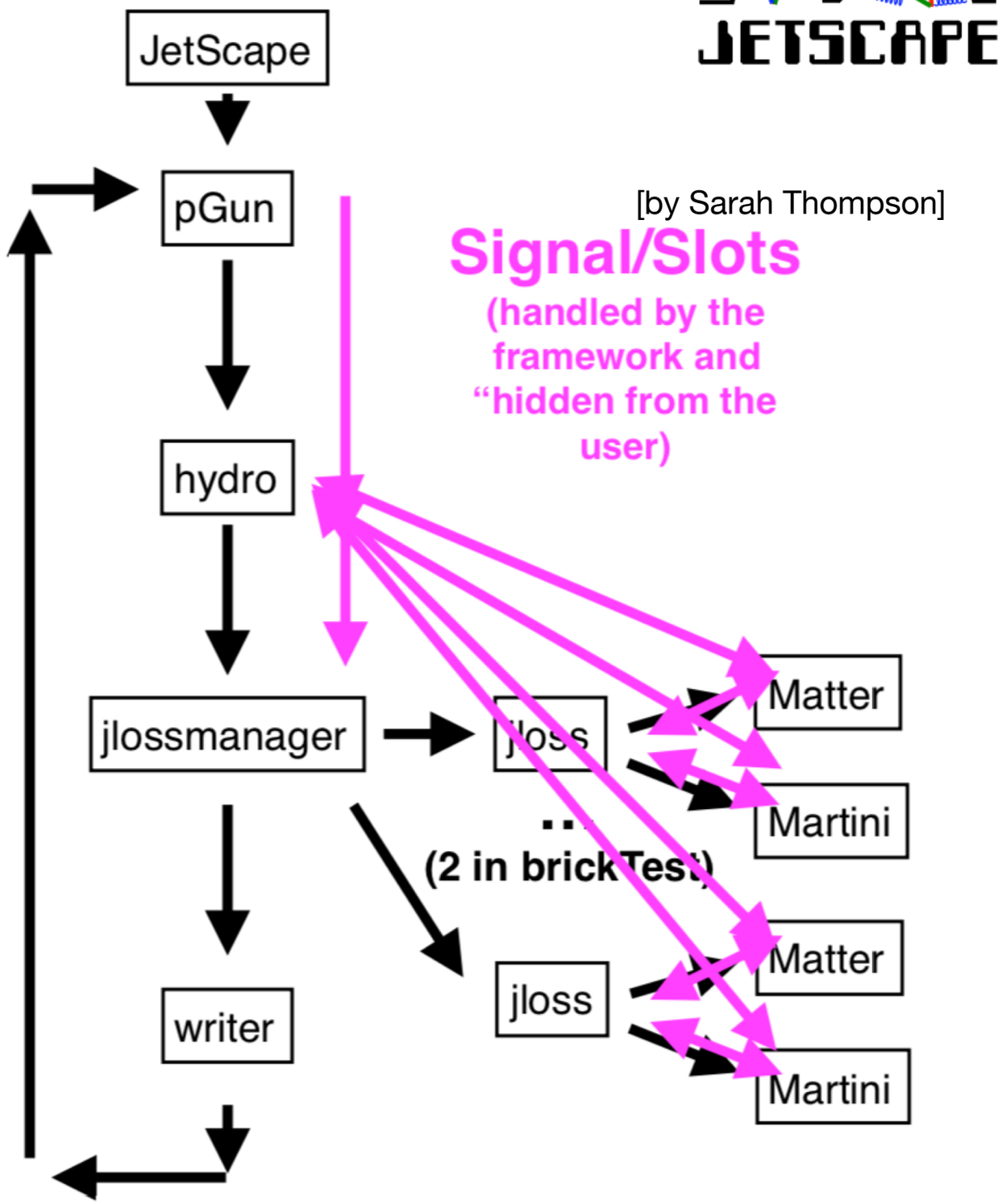
- **JetScapeModuleBase**
  - Ensures unified interface via JetScapeTask
  - Knows how to connect modules via signal/slots
  - Ensures task/thread safe random number seed
  - Each module will be
    - Initialized, executed
    - and cleared

	JetScapeModuleBase (string m_name)
virtual	**~JetScapeModuleBase** ()
virtual void	**Init** ()
virtual void	**Exec** ()
virtual void	**Clear** ()
void	**SetXMLFileName** (string m_name)
string	**GetXMLFileName** ()

Functions will be called recursively

Framework Internals

- **JetScapeSignalManager**
  - Handles connections and data exchanges
  - Implement Observer design pattern
  - Send a signal when you're done



How Does JetscapeSignalManager work?

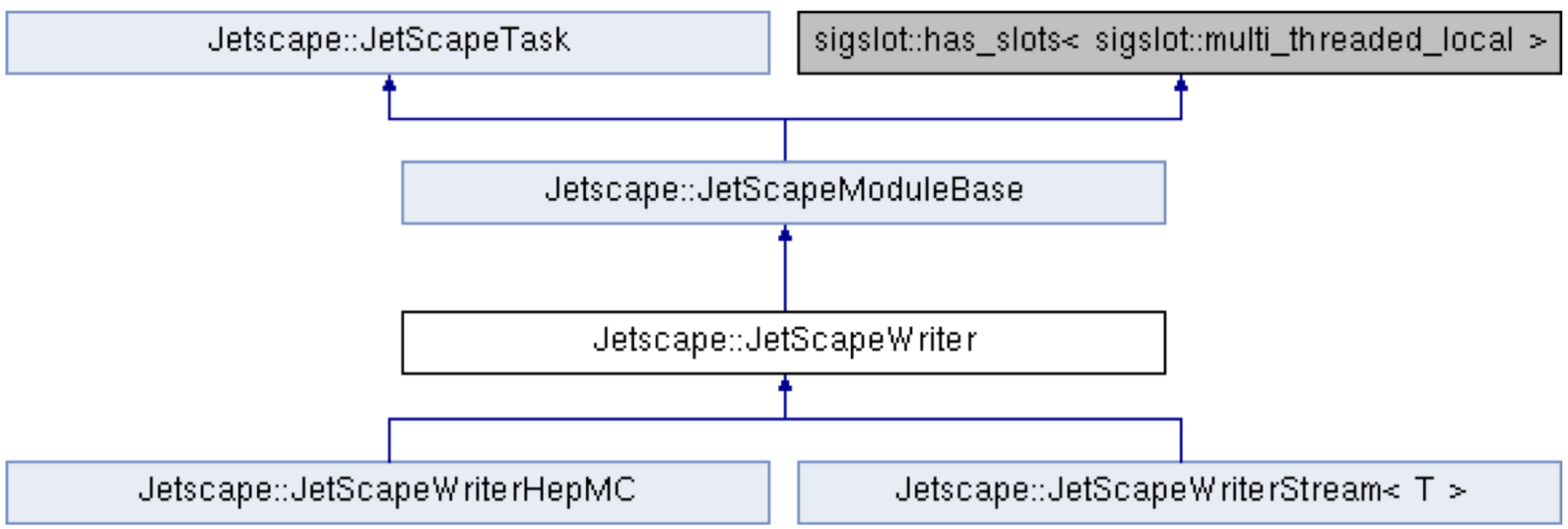

```
void JetScapeSignalManager::ConnectGetHardPartonListSignal(shared_ptr<JetEnergyLossManager> jm)
{
   if (!jm->GetGetHardPartonListConnected())
   {
      auto hpp = GetHardProcessPointer().lock();
      if ( hpp )
      {
         jm->GetHardPartonList.connect(hpp.get(),&HardProcess::GetHardPartonList);
         jm->SetGetHardPartonListConnected(true);
      }
   }
}
```

Signal

Slot

Framework Output

- Class **JetScapeWriter**



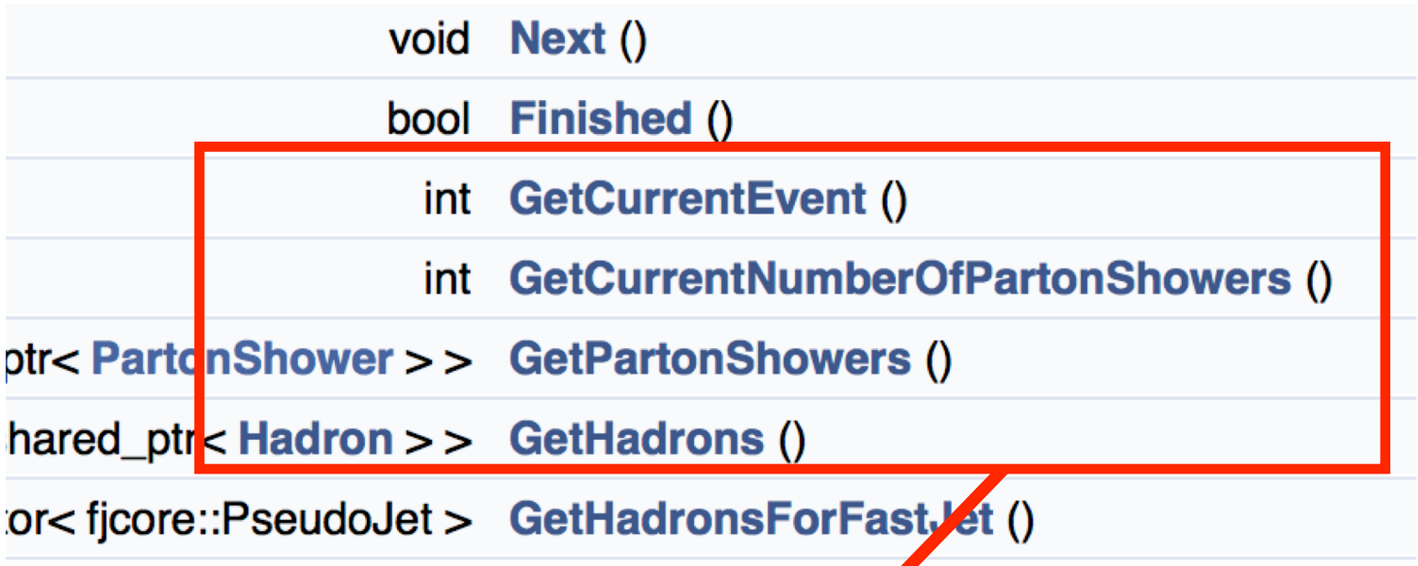
- Subclass of JetScapeModuleBase
  - Can be attached as a task
- Derived classes so far:
  - HepMC writer
  - ASCII writer

Each task overrides its own write method

virtual void	**FinishTask** ()
virtual void	**FinishTasks** ()
virtual void	**WriteTasks** (weak_ptr< **JetScapeWriter** > w)
virtual void	**WriteTask** (weak_ptr< **JetScapeWriter** > w)
virtual void	**CollectHeader** (weak_ptr< **JetScapeWriter** > w)
virtual void	**CollectHeaders** (weak_ptr< **JetScapeWriter** > w)

Reading Output

- Class **JetScapeReader**
  - Base class for reading JETSCAPE output files
- Not a JETSCAPE task
  - To be used after producing output
- Derived classes so far:
  - JetScapeReaderAscii

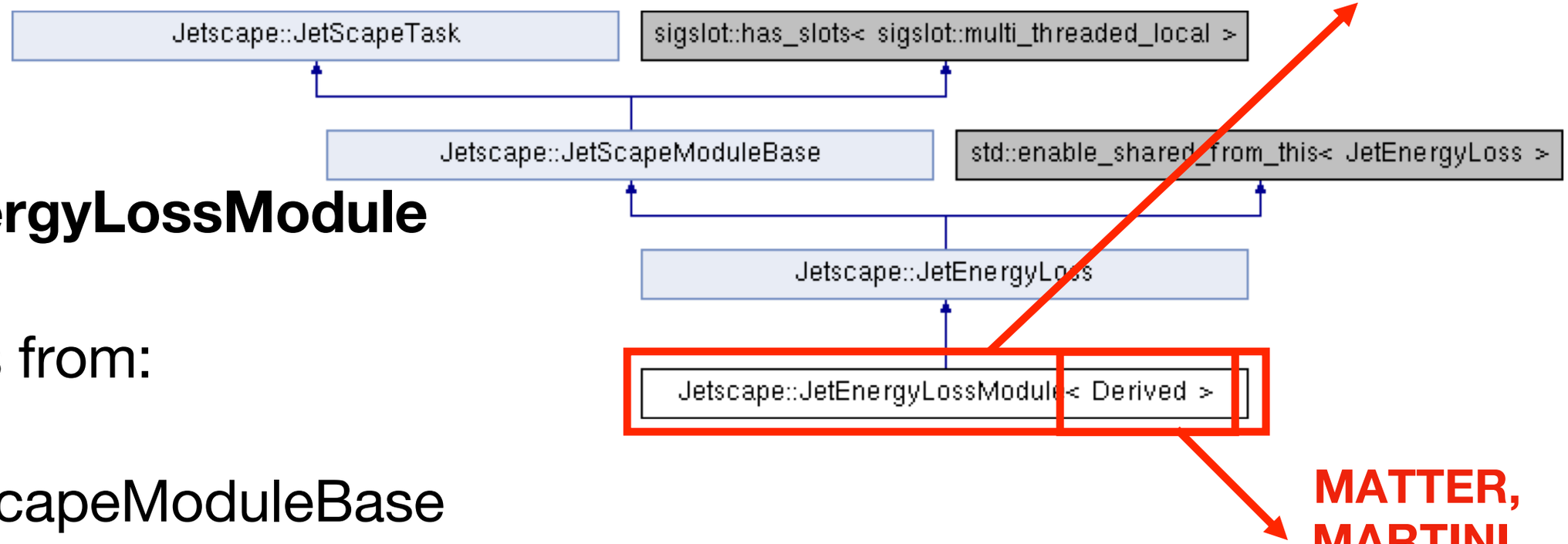


Users Responsibility: Add Physics Modules

- Users can ignore the internals of JETSCAPE framework and how it works

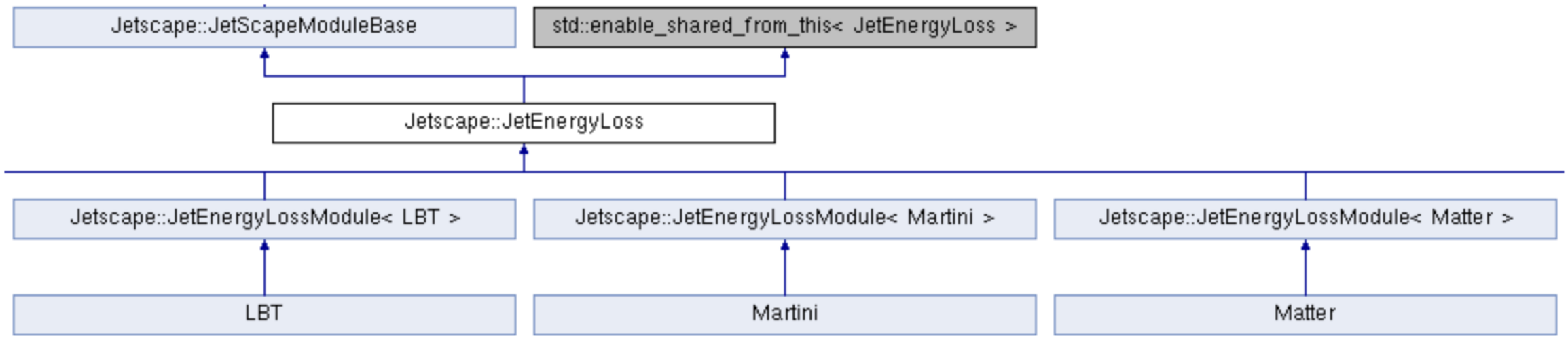
- Users take an existing code
  - Change it to respect JETSCAPE framework interface
  - Framework takes care of the rest
- Users contribute code to any aspect of heavy-ion collision
  - From initial state to hadronization

Base Class For Adding an Energy-Loss Module



- **JetEnergyLossModule**
- Inherits from:
  - JetScapeModuleBase
    - Framework knows how to run the code
  - JetEnergyLoss
    - Enables doing Energy-Loss calculations

An Example To Add an Energy Loss Module



- User's code must be a subclass of **JetEnergyLossModule**
- User's code must override **init()** method for initializations
- User's code must override **DoEnergyLoss()** method
  - The actual energy loss calculations happen in this method

Simple Examples

- Users can use example codes to construct their simulations
  - Simple p-p collision: PythiaBrickTest
  - Simple smooth hydro: HydroJetTest
  - Full hydrodynamics evolution: MUSICTest

Conclusion

- JETSCAPE is one code package for all aspects of Heavy-Ion collisions
- Using JETSCAPE, experts on one aspect can just reuse modules for other aspects
- By respecting the interface, users can integrate their code into JETSCAPE framework
  - Users only need to know input from framework and how to use output
- We require feedback from additional theorists to see how we can integrate their physics