Building a Community Infrastructure for Scalable On-Line Performance Analysis Tools

Jim Galarowicz, Krell Institute

Dave Montoya, LANL

Wouldn't if be advantageous to our users if we could more easily create tools from all the great work/research being done in our community?

If we could create components
that could be shared
we could tailor the solution
to the user problem and
the environment they are running in.

Agenda:

- ***Team***
- ***Project overview***
- ***Enabling an Open Community Infrastructure***
  - *Open Interfaces acceptable across community*
  - *Mechanisms for Constraint and Dependency Recognition and Tool Creation*

Project Team

- **Project Team**
  - Krell Institute
  - University of Maryland
  - University of Wisconsin
  - Oak Ridge National Laboratory
  - Lawrence Livermore National Laboratory
  - Los Alamos National Laboratory
  - Sandia National Laboratories
  - Carnegie Mellon University
  - Others welcome……

Agenda for OASCR/NNSA Project Overview

- *Objectives and Rationale*
- *Research Challenges*
- *Flexible Performance Tools Pipeline*
- *Creating a Performance Tools Pipeline*
- *Target Challenges*

Objectives and Rationale

- **Objectives:**
  - *Create a toolbox of components for building high-level end user tools and/or quickly build tool prototypes.*
  - *Tools should be easily configurable/adjustable w/o rebuilding.*
  - *Able to mix components from several groups and/or vendors. Everyone should be able to contribute and use the new components.*
  - *We would like contributors to define the interfaces with us so that we can share components later in both directions.*

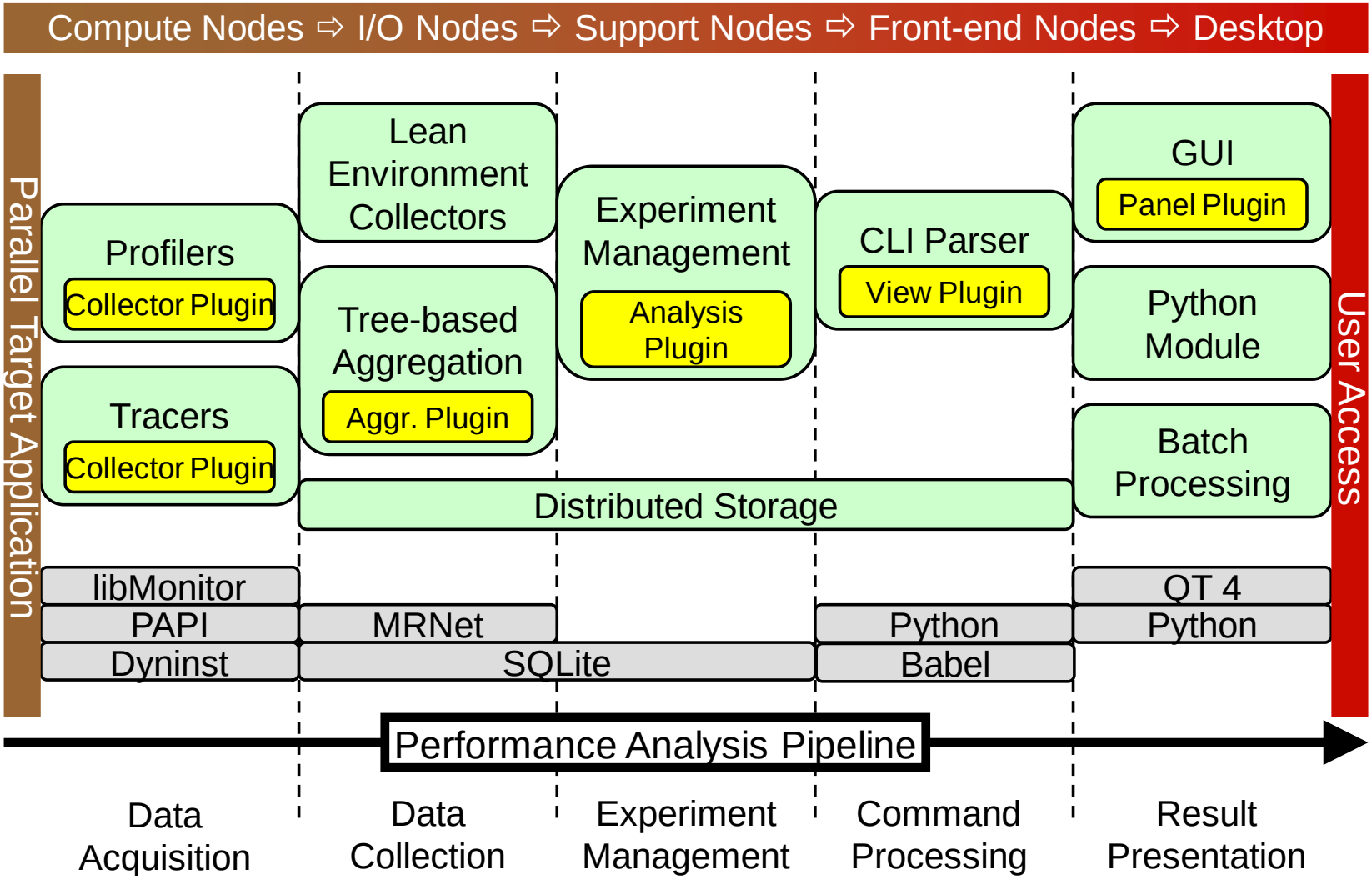
Objectives and Rationale

- ***Rationale:***
  - *Petascale environments need tool sets that are flexible*
  - *Need to quickly create new and specialized tools*
  - *Better availability of tools across more platforms*
  - *Avoid creating stove pipe tools*
  - *Better support model because more groups are involved with their own components?*

Research Challenges Project Requirements

- **Research Challenges/Project Requirements:**
  - *In general components must be designed for scale but also have a need for generality.*
  - *Must also support specialized tool components intended for serial or small scale usage.*
  - *Infrastructure must support online data aggregation because of potentially high data volume at scale.*
  - *Petascale machines are likely to have limited OS capabilities requiring new and light-weight data acquisition techniques.*
  - *Must be able to efficiently store the performance data.*
  - *Must be able to map any combination of tool components to the target architecture.*

Performance Analysis Pipeline



Performance Tools Pipeline

- **Performance Tools Pipeline**
  - *Work closely with tools community and application teams sharing ideas and feedback, as well as sharing components.*
  - *Specify the components interactions with other components.*
    - *Allows others to add or replace components.*
    - *Integrate components into their tools*
    - *Customize existing solutions.*

- *Note: The presented option is one that we think makes sense, but we are willing to change it to match other tools.*

Performance Tools Pipeline

- **Creating a first Performance Tools Pipeline prototype**
  - *Start with Open|SpeedShop components as one set of examples for such an infrastructure.*
  - *Decompose core components into general building blocks.*
  - *Arrange building blocks into a logical performance analysis pipeline.*
    - *Allows users and tool builders to select individual components for each pipeline stage.*
    - *Supports a flexible mapping onto the target architecture which provides efficient execution and visualization (incl. remote operation) environments.*

Target Challenges

- Scalability
- Data Collection and Aggregation
- Novel Data Acquisition Techniques
- Distributed Performance Data Storage
- Mapping the Application to the Architecture
- Open Interfaces

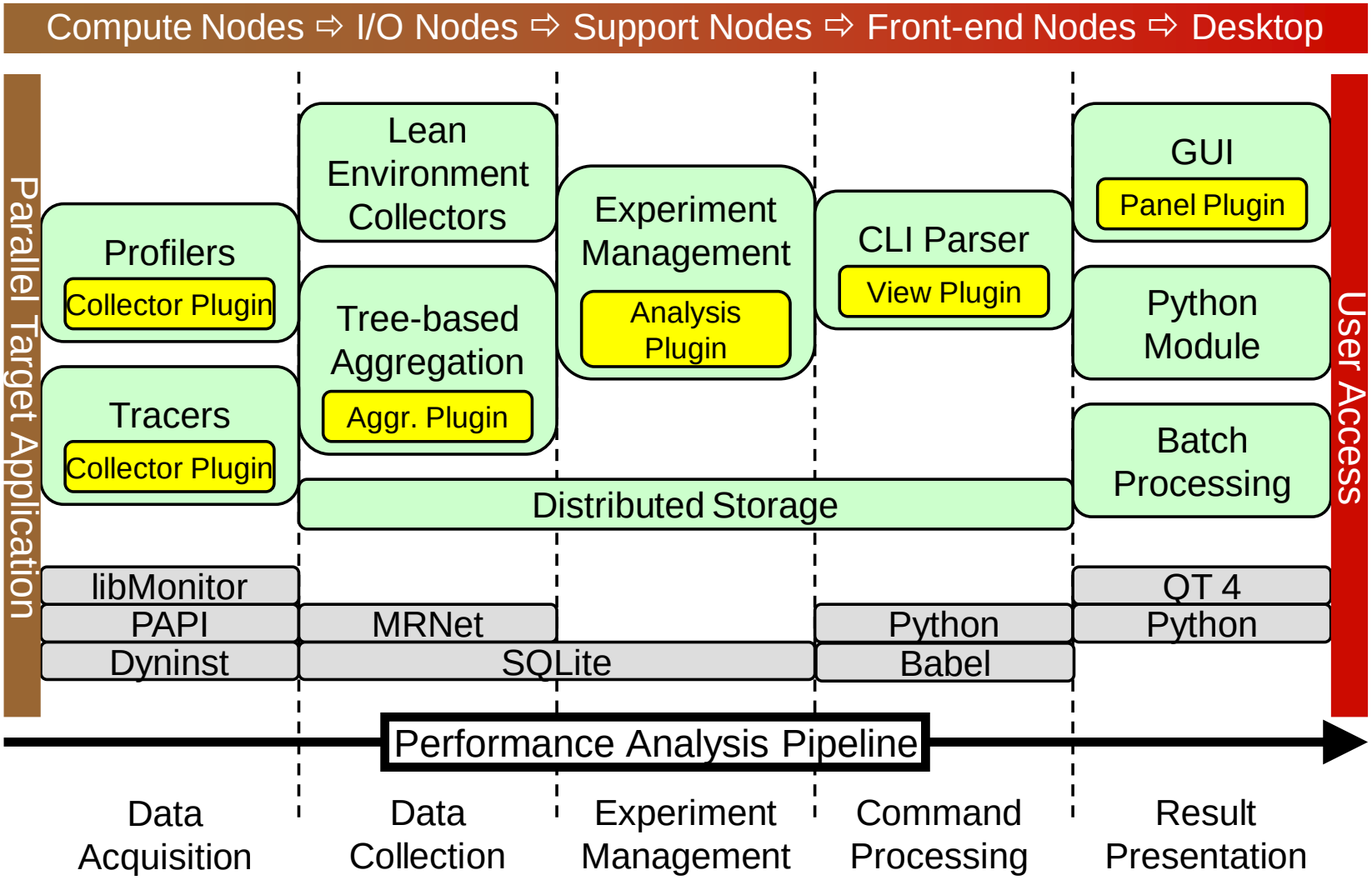
Agenda for Component Discussion

- ***Creating Open Community Components***
  - *Design Approaches*
  - *Subsystems plug-in structure*
    - *Initial analysis diagram*
  - *Would welcome your input on the interface design*

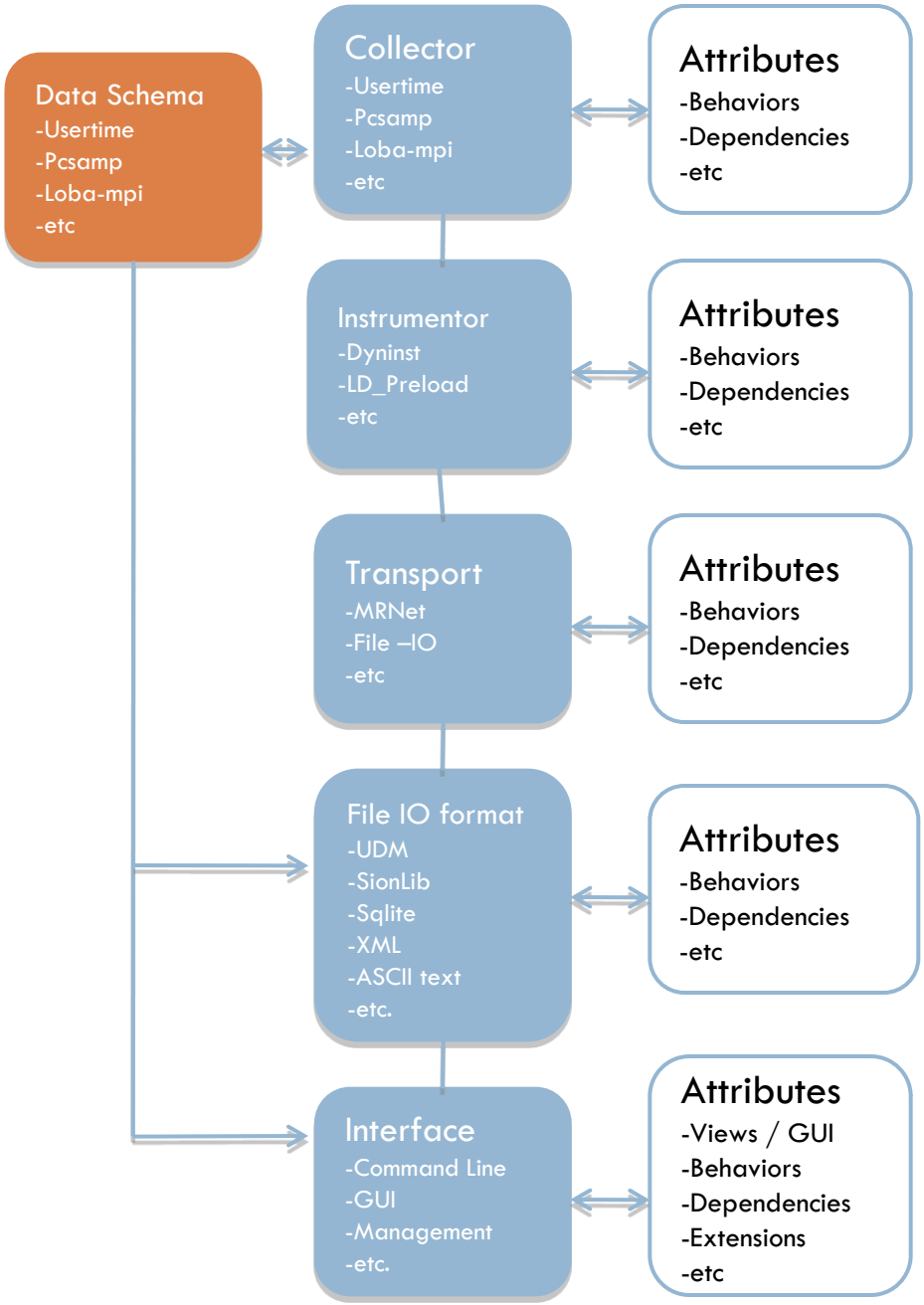
Design Approaches

- ***Identify appropriate interface points***
  - *Define interface point behavior / constraints*
- ***Assess usage scenarios that address***
  - *Challenges*
  - *Tool integration*
- ***Focus on flexibility and sustainability***
- ***Design support for frameworks - CMU team***

Performance Analysis Pipeline



For Discussion - Analysis of interface points



Example
Subsystems plug-in structure
Initial analysis diagram

Behaviors:

Address commonality. All plug-ins for a given system must have common behaviors for them to work in unison. Do we have different strands of behavior through the system?
-Dynamic vs. pre-set
-Light-weight OS

Dependencies:

Address expectations. plug-ins for a given sub-systems are expected to adhere to a defined interface of what is provided by compatible upstream sub-systems and what they provide to downstream subsystems. There are also specific constraints that are defined by plug-ins that serve as service providers at the various levels.

QUESTIONS and DISCUSSION

- *Are the components in the pipeline at the correct granularity?*
- *Any thoughts on approaches to defining interfaces?*
- *Which components would fit into this scenario?*
- *Which components would not fit and why?*
- *What can we do to help ensure more components will be included?*