KRELL
institute

Open | SpeedShop™

Open Source Performance Analysis for Large Scale Systems

Generalizing Components from Open|SpeedShop

Workshop on Performance Tools for Petascale Computing

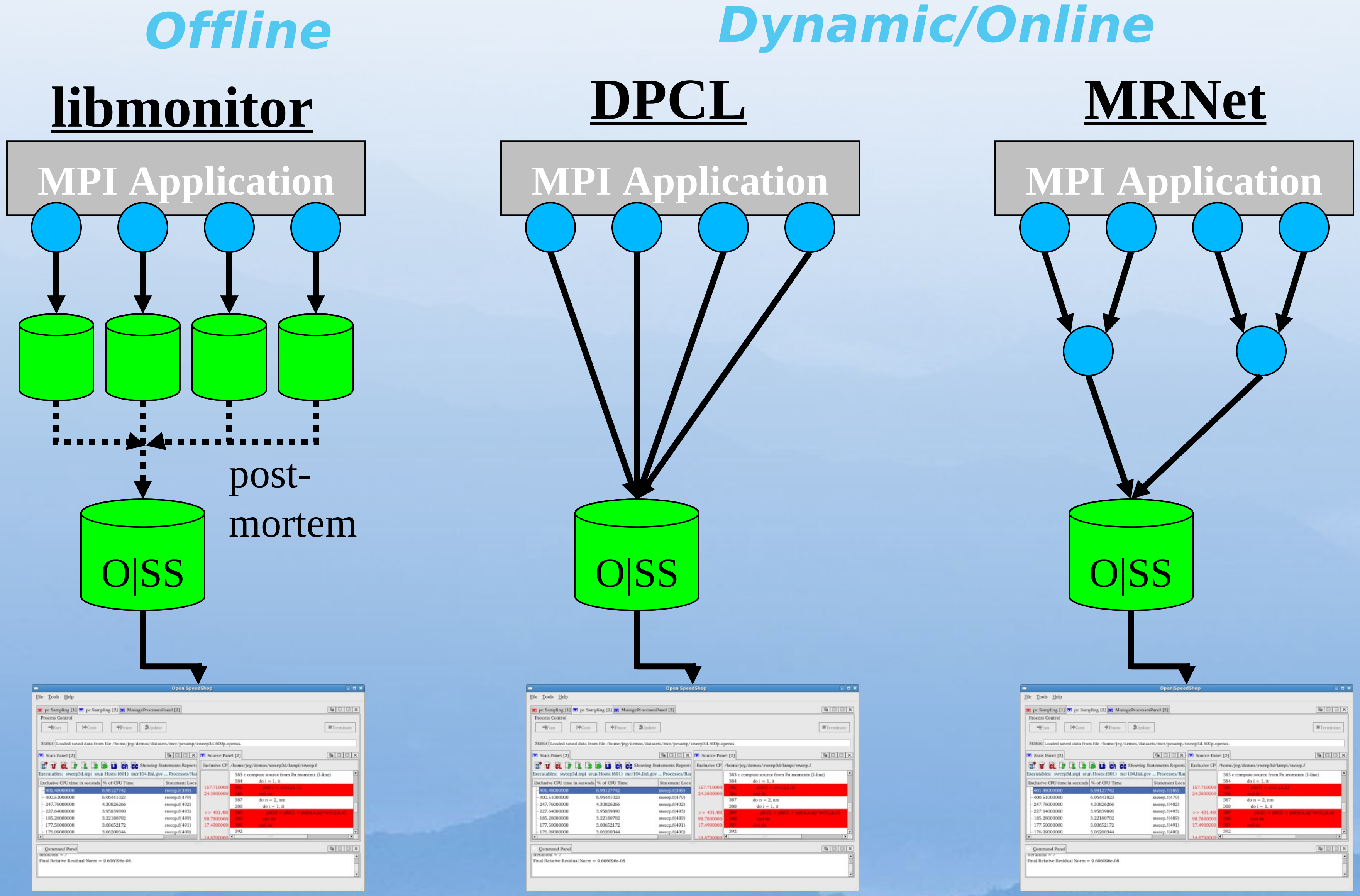
July 21, 2008

Jim Galarowicz, Krell Institute

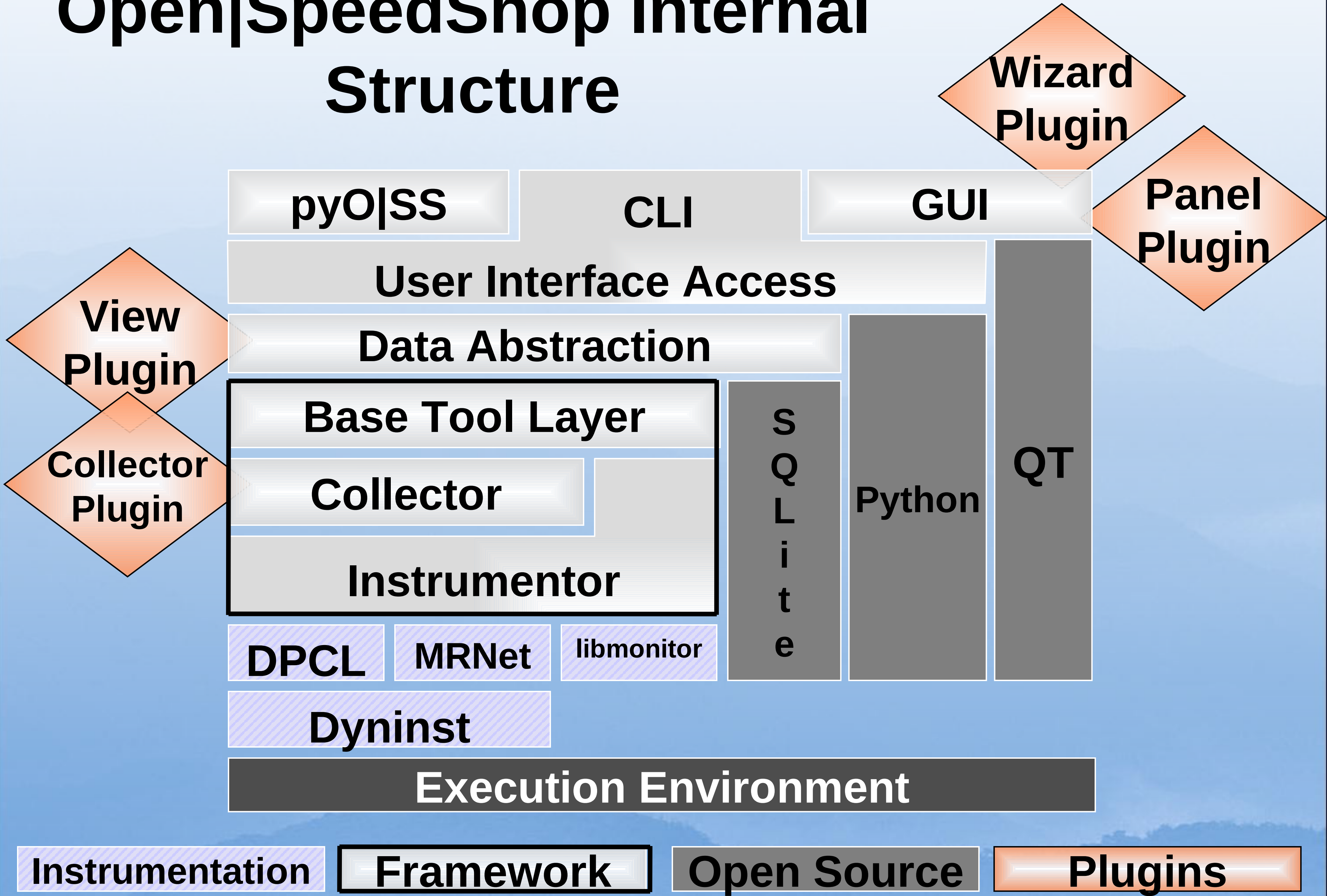
Talk Outline

- O|SS Internal Structure
- External components used by O|SS
- Current components provided by O|SS
- Future components and O|SS structure
- External components wanted
- Experience integrating vampirtrace
- Questions

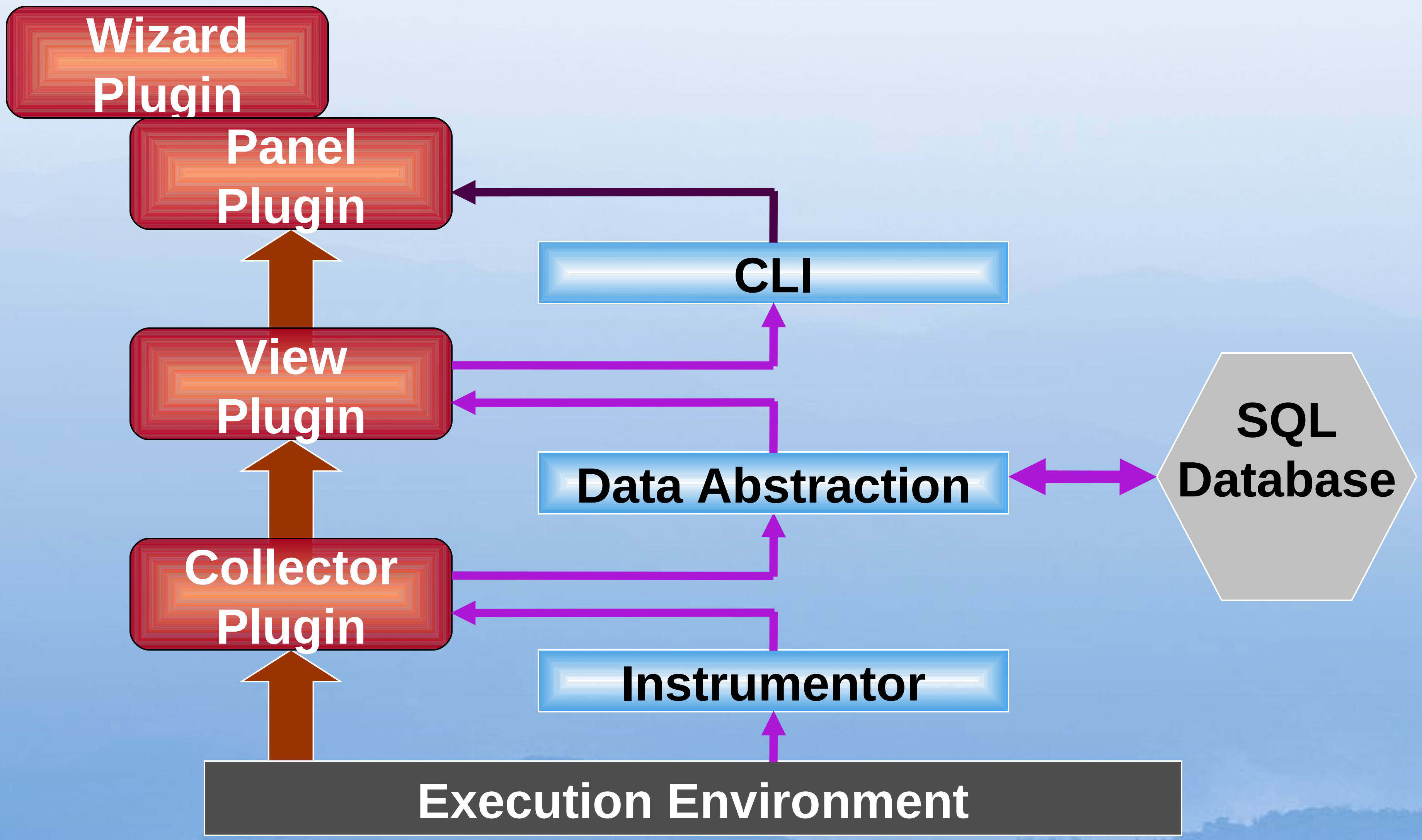
Open|SpeedShop Overview



Open|SpeedShop Internal Structure



Plugin Data Flow

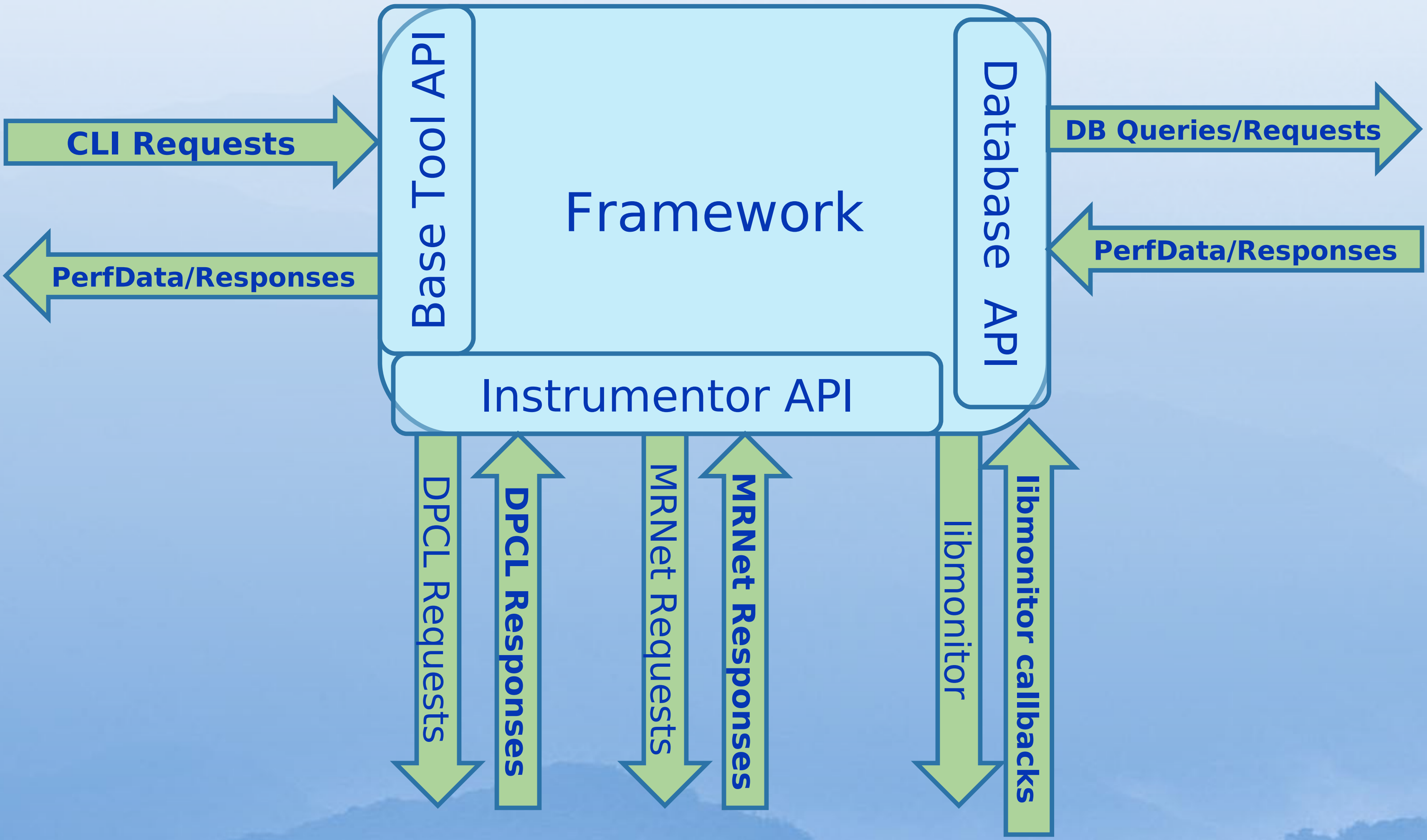


External Components

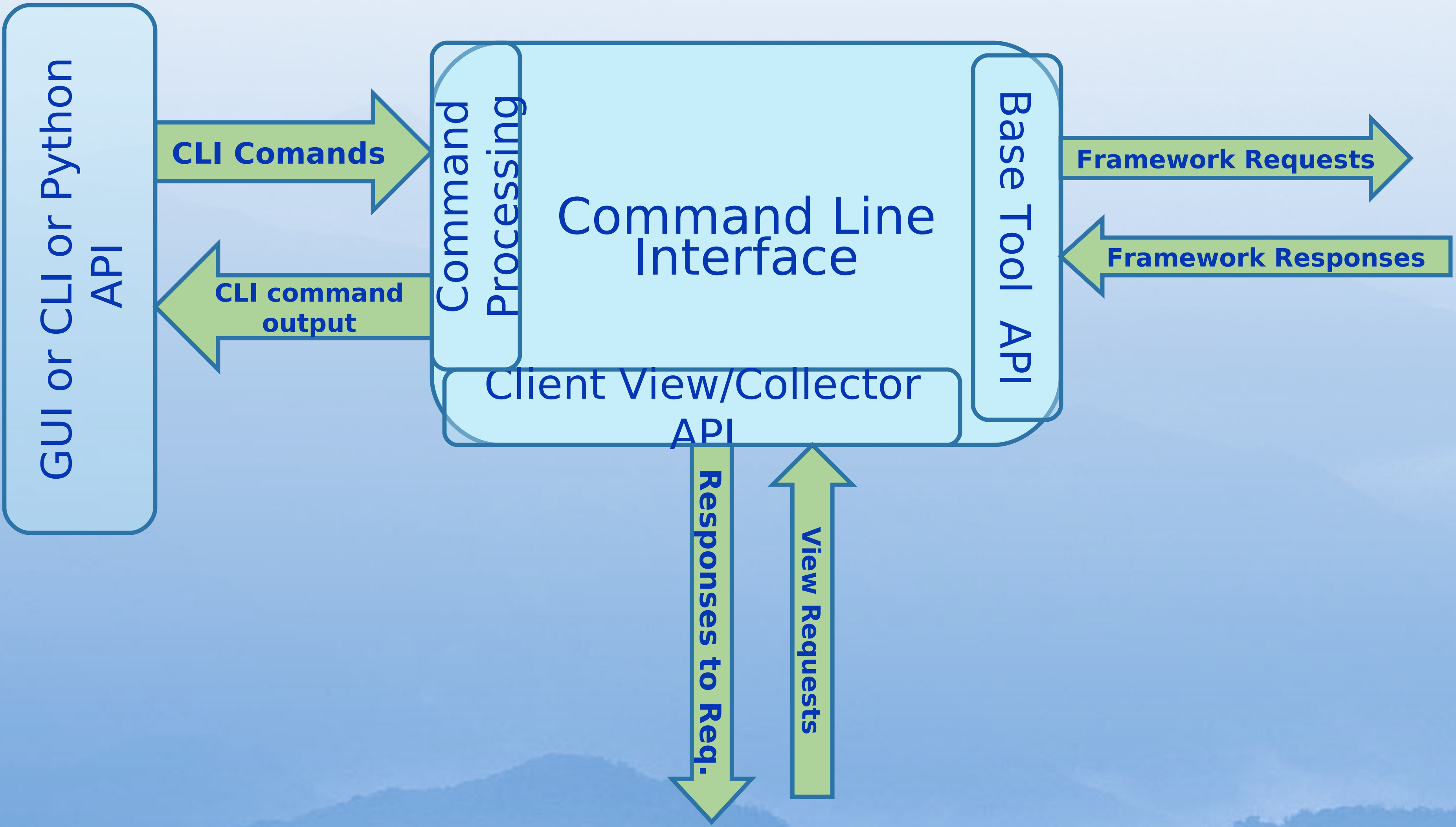
Current external components used by O|SS:

- **Dyninst, symtabAPI, MRNet, [DPCL]**
- **SQLite, Python, QT, PAPI**
- **libelf, libdwarf, libmonitor, libunwind**
- **xdr, libbfd, libopcodes, binutils**
- **vampirtrace**
- **In process: mpiP, Javelina**
- **Future: perfmon2, LaunchMON, stackwalkerAPI**

Framework Component



CLI Component



Current OSS provided components

- **Framework component Interfaces:**
  - **Instrumentors: MRNet, DPCL, Offline (libmonitor: LD_PRELOAD, static relinking)**
  - **Database (SQL based interface, SQLite implemented)**
  - **Base tool API (Process state, Access Data)**
- **Command Line component Interfaces:**
  - **Python scripting, CLI interactive , GUI interface**
  - **Process CLI commands that drive component/tool**
  - **Interface with framework component**
  - **Interface with view/collector client plugin**

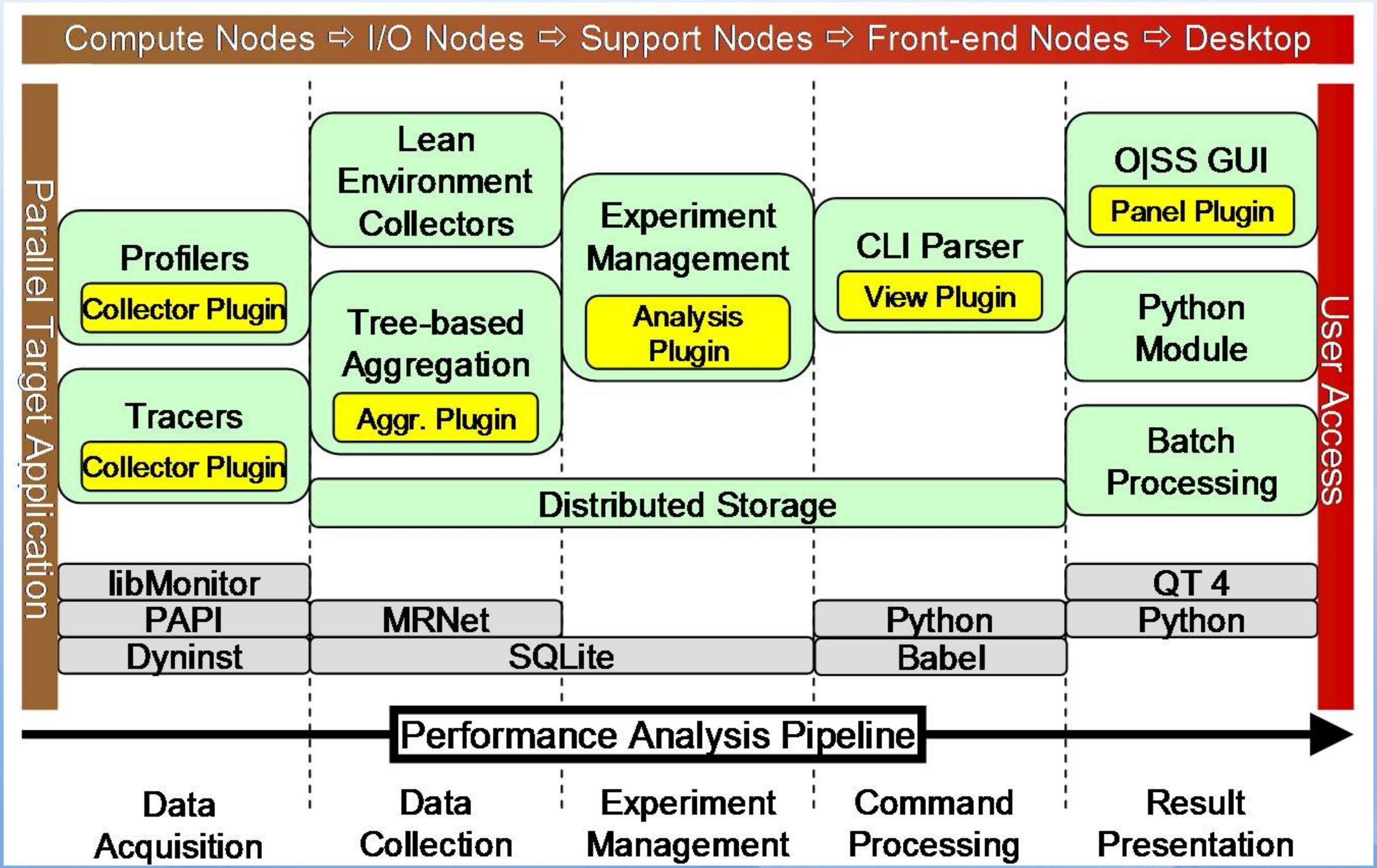
Current OSS components available

- **Runtime support component Interfaces:**
  - **API for runtime collector components/plugins**
- **Plugin components (Collector, View, GUI) Interfaces:**
  - **Collector (pcsamp, usertime, hwc, hwctime, io, iot, fpe, mpi, mpit, mpiotf)**
    - **Runtime support API**
    - **MRNet/DPCL daemon API**
  - **Views**
    - **Database API & CLI View API**
  - **GUI (Panels/Wizards)**
    - **CLI command interface (Commands)**

Future O|SS Structure

- Goal
  - Highly scalable individual components
  - Generalized API for each component
  - Reassemble the components into new O|SS
  - Create other tools by assembling components
- Path to the Goal
  - Re-engineer O|SS-centric components
    - Take out O|SS specific hooks
    - Decompose components to be free standing
  - Generalize APIs

Future OSS components and structure



External components wanted

Components we would like to use:

- **Binary rewriter**
- **Highly scalable distributed data transport/storage**
- **Graphical view with well defined API to specify the data**

Issues using external components

The good:

- You don’t have to reinvent the wheel!
- Big win: usually!
- Example: Integrating vampirtrace into O|SS to get OTF capability

The not so good:

- Components are constantly changing (APIs, library interfaces)
- Most likely don't have control over the changes

Experience integrating vampirtrace

- Integration into offline version
  - Mainly configuration issues building multiple MPI implementation versions
- Integration into dynamic online version
  - Move MPI dependent routines in vt to collector
    - Compile into each MPI Implementation dependent collector
  - Complicated due to fact we stop in MPI_Init to attach to all MPI ranked processes
    - vampirtrace initialization is done in MPI_Init
    - Separated out the init routine from vt code and executed it as one time code snippet.

Questions?

Jim Galarowicz

jeg@krellinst.org

Krell Institute

http://www.krellinst.org