INTERNATIONAL VIRTUAL OBSERVATORY ALLIANCE
US National Virtual Observatory OPTICON AUI AURA NSF

Applications Framework

D. Tody (NRAO, NVO), P. Grosbol (ESO, OPTICON), et. al.

Applications Framework

- Topics
  - What is it
  - Who is involved
  - Applications framework concept
  - Near term plans

What is the Applications Framework?

- A framework
  - A common desktop environment and computational framework
  - Python, Java are baselined for top level applications, but the architecture is not limited to these
  - Computational components may be in any language
  - Provides scalable computation

- A set of open system standards
  - Specifications for key elements of the framework
  - Reference implementations for the framework elements
  - May be used individually or as an integrated framework

What Is It Not?

- A complete end-user system
  - Framework, app packages, tools are separate products
  - They come together to form a complete system
  - Many system configurations are possible
- A class library for implementing applications
  - O/IR, radio, X-ray, etc. require (and already have) their own specific class libraries and algorithms
  - Legacy software already has its own class libraries built-in
  - We need class libraries to build applications but they are a separate (but potentially integrated) product

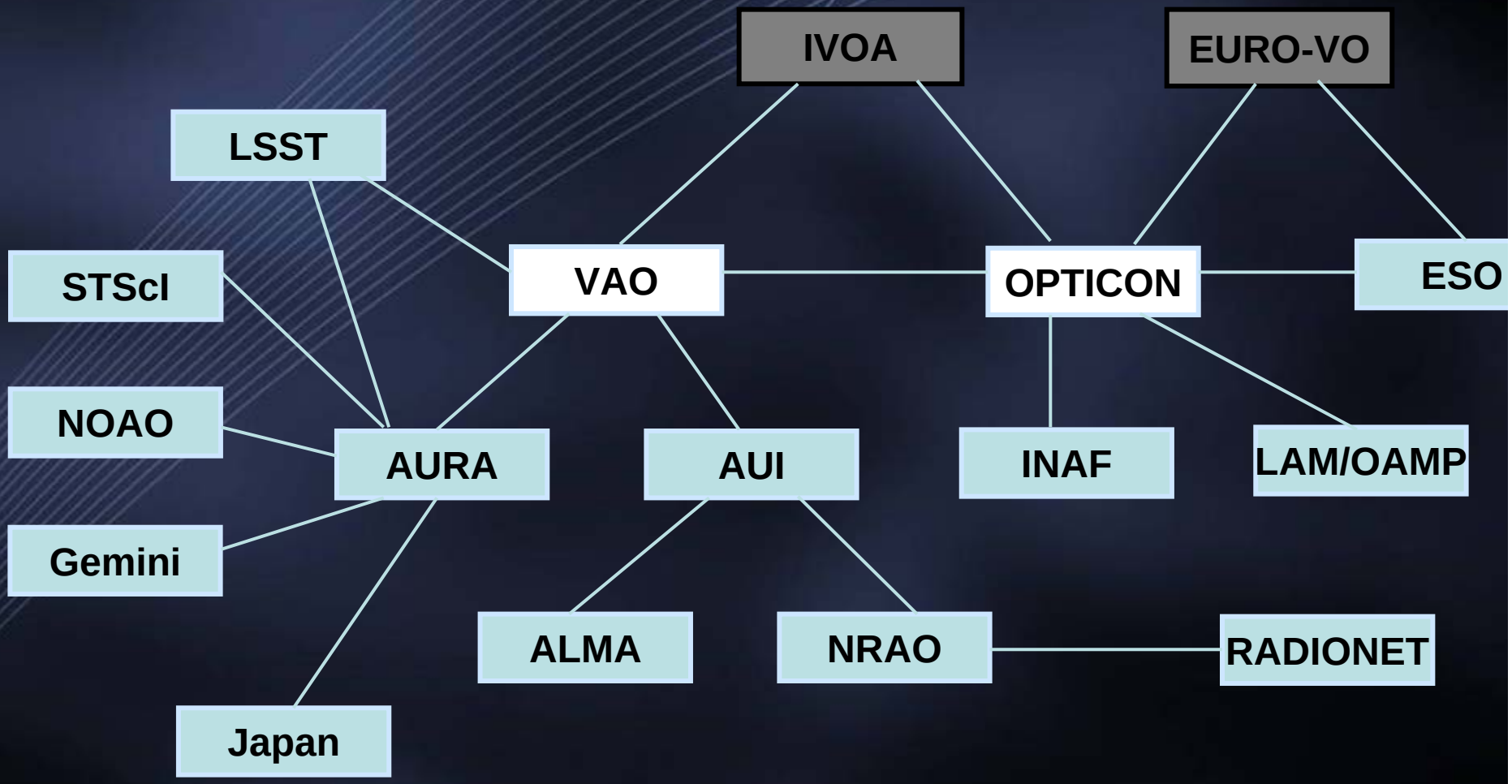
Applications Framework

- Some Goals
  - Integrate software from multiple observatories and projects
    - app packages from many sources install into common environment
  - Integrate observatory software with the VO and with VO technology
    - use VO standards (Grid, data access, DM, etc.) in observatory s/w
    - use observatory software to build VO services and applications
  - Migrate legacy software to a modern new environment
    - AIPS, CASA, GIPSY, IRAF/PyRAF, MIDAS, Starlink, etc.
  - Provide a framework for constructing and deploying new applications
  - Provide common user experience and promote software sharing/reuse
  - Address harder problems
    - open system, scalability, standards development

Typical Use Cases

- Desktop data processing and analysis
  - Python UI/CLI/scripting environment and tools
  - Framework plus applications packages
  - Client integration with VO (VOClient, data access, etc.)
- Pipeline processing
  - Instrumental data processing pipelines
  - Both facility mode and user-reprocessing modes
  - Scalable, remote/distributed execution
- Data processing for VO services
  - Virtual data generation for data services (DAL)
  - Analysis services with Grid front end (CEA etc.)

Who Is Involved?

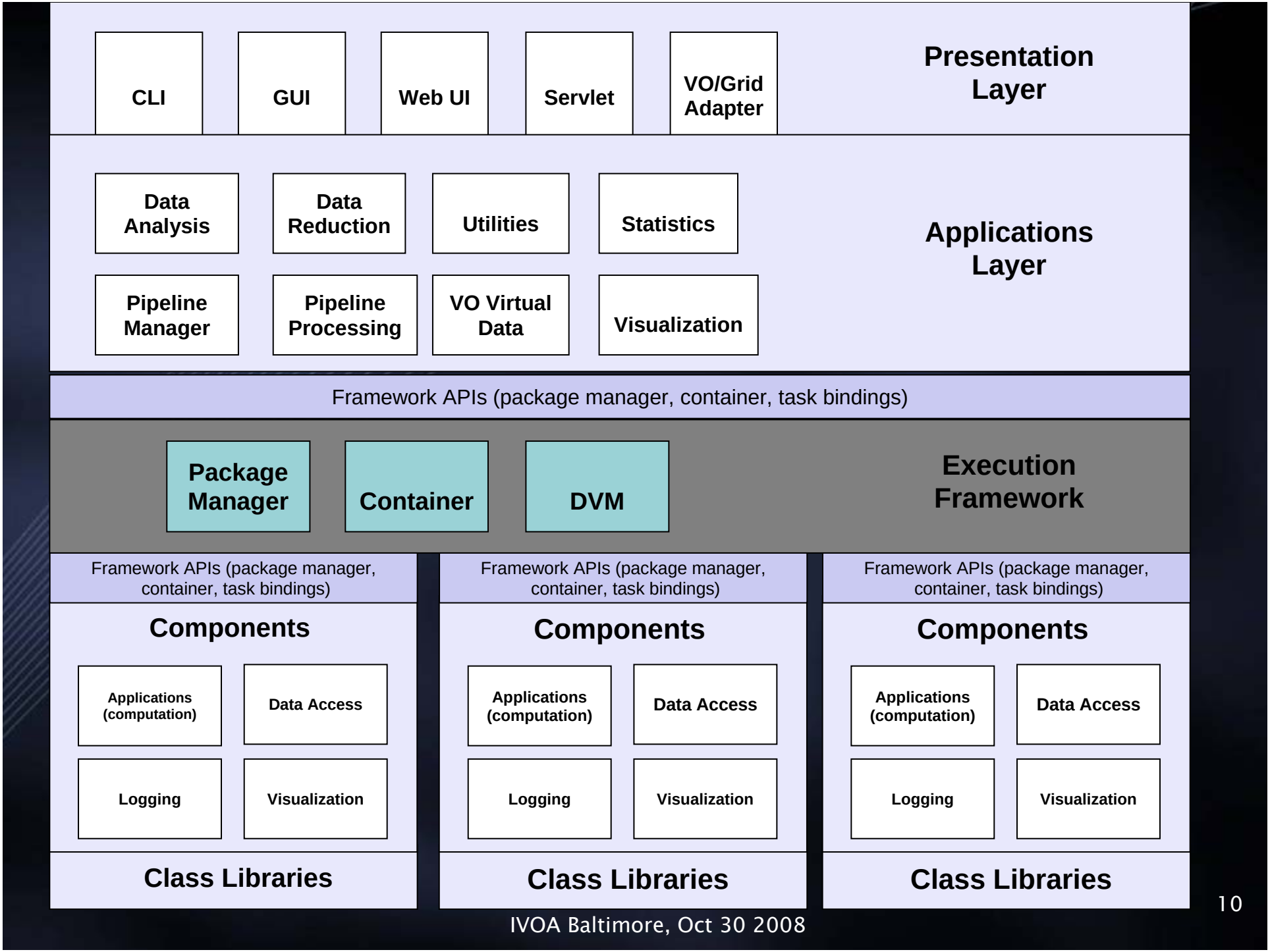


Current Status

- OPTICON/NVO Collaboration
  - Phase I wraps up this year
    - Requirements document
    - Architecture whitepaper
    - Applications framework conceptual design (*new*)
  - Phase II (implementation) funded, begins next year
    - More on this at the end of talk

Current Status

- AURA Software Initiative
  - AURA observatories (NOAO, STScI, Gemini)
  - Also LSST, NVO/VAO, Japan
  - Hilo workshop held June 2008
  - Joint strategic whitepaper in progress
- AUI Software
  - CASA project (also AIPS, ParselTongue/OBIT with RADIONET)
  - AUI and AURA will jointly manage VAO


CLI
GUI
Web UI
Servlet
VO/Grid Adapter
Presentation Layer
Data Analysis
Data Reduction
Utilities
Statistics
Pipeline Manager
Pipeline Processing
VO Virtual Data
Visualization
Applications Layer
Framework APIs (package manager, container, task bindings)
Package Manager
Container
DVM
Execution Framework
Framework APIs (package manager, container, task bindings)
Components
Applications (computation)
Data Access
Logging
Visualization
Class Libraries
Framework APIs (package manager, container, task bindings)
Components
Applications (computation)
Data Access
Logging
Visualization
Class Libraries
Framework APIs (package manager, container, task bindings)
Components
Applications (computation)
Data Access
Logging
Visualization
Class Libraries

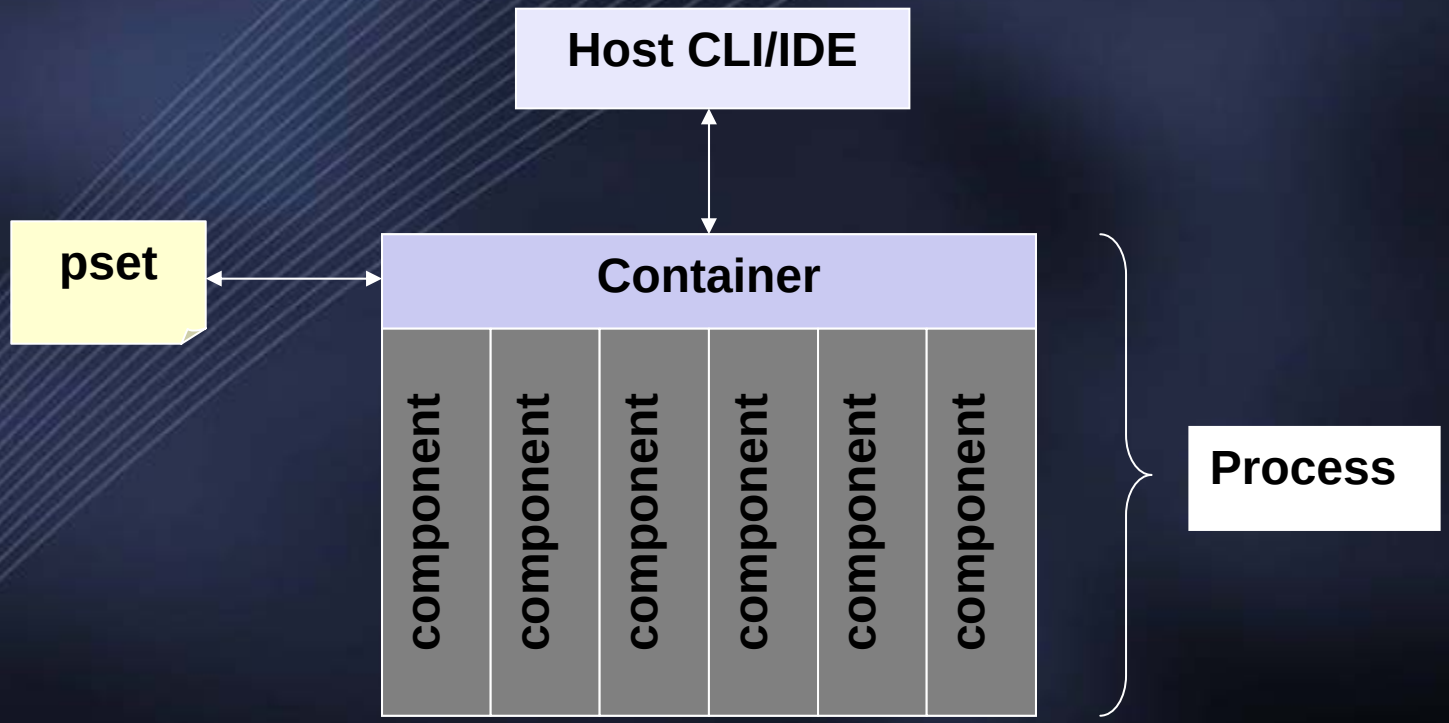
Framework Elements

- Packaging
  - Applications package containing applications, components, parameter sets, package documentation, package metadata, etc.
  - Entire package in one Zip file (like a servlet War file)
  - Legacy software is repackaged in this form
- Component Container
  - Components are "tasks" (one method) and "tools" (DO, stateful)
  - Components have a standard interface, run within the container
- Parameter Mechanism
  - Tasks and tool methods have a parameter set interface
  - Parameter sets used for both input and output, also data entities

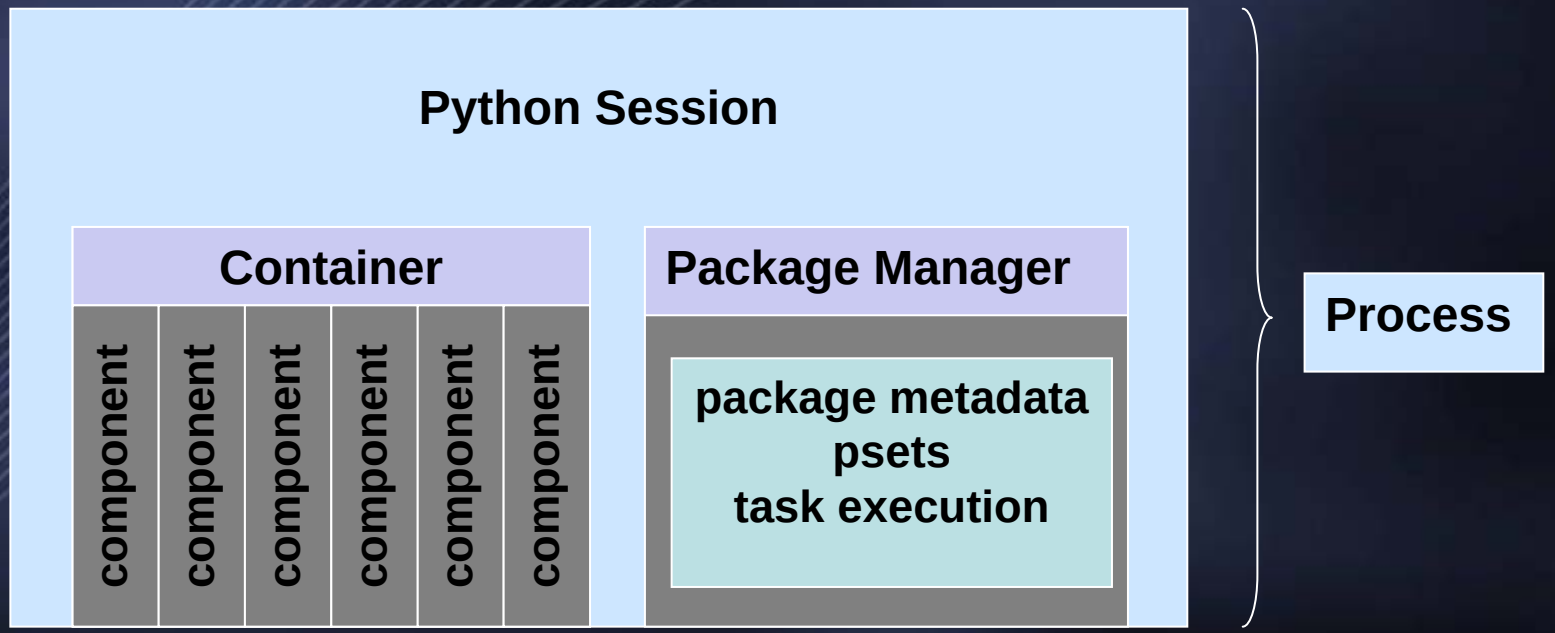
Framework Elements

- Messaging
  - Applications and components can communicate via messaging
  - SAMP (2.0 probably) is part of this but not all there is to messaging
- Package Manager
  - Runtime management of packages, tasks, parameter sets, etc.
  - Manages metadata, instances, pset storage, task execution, etc.
- Distributed Virtual Machine (DVM)
  - Provides basic functionality for distributed, scalable execution
  - MPI integration (e.g. OpenMPI), SPMD parallelization capabilities
  - Low level messaging and process control
- Framework Services
  - Logging, concurrent data access (locking, data integrity)

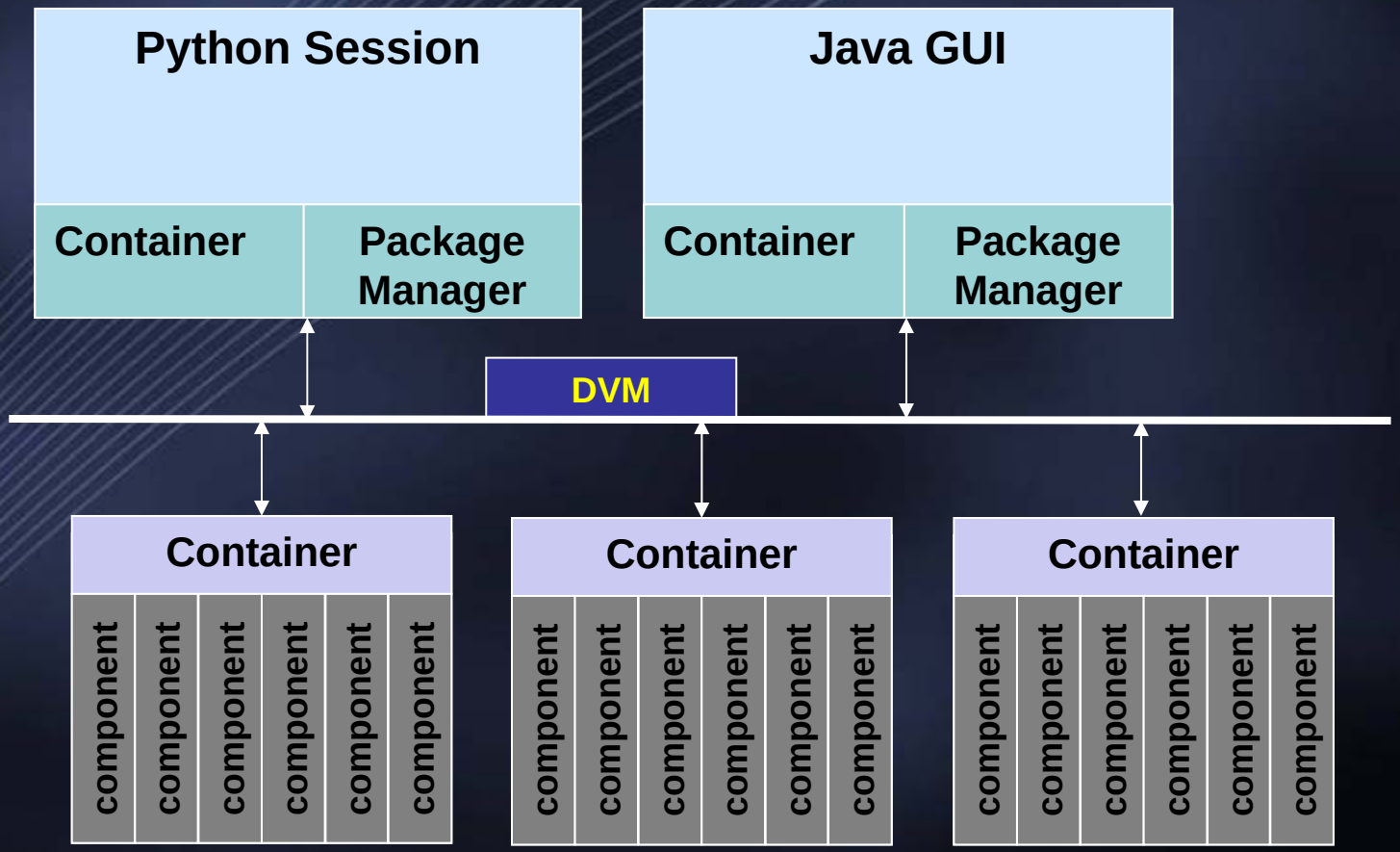
Container (Host Execution)



Container (In-Process)

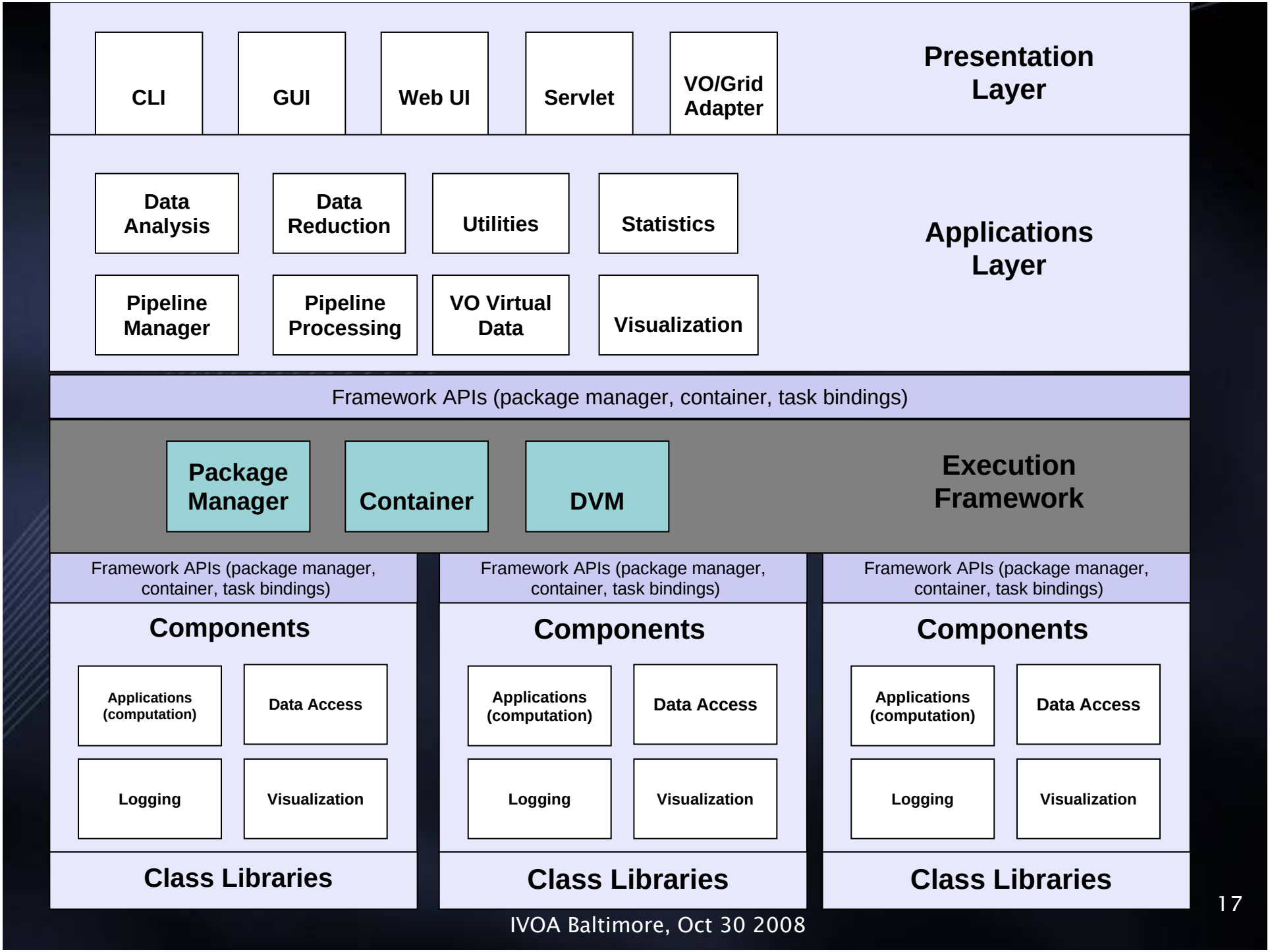


Distributed Execution



Near Term Plans

- VAO/OPTICON effort
  - Runs for 2-3 years
  - Yearly specify/build/test cycle
  - Specify, reference implementations, end-to-end testing
  - Year 1 TBD, e.g., component/container, parameters, Python integration
- Broader Effort
  - Plans still under discussion
  - All projects have short term plans specific to project
  - Longer term planning coordinated with common framework effort
  - Projects do use-case implementations based upon framework


CLI
GUI
Web UI
Servlet
VO/Grid Adapter
Presentation Layer
Data Analysis
Data Reduction
Utilities
Statistics
Pipeline Manager
Pipeline Processing
VO Virtual Data
Visualization
Applications Layer
Framework APIs (package manager, container, task bindings)
Package Manager
Container
DVM
Execution Framework
Framework APIs (package manager, container, task bindings)
Components
Applications (computation)
Data Access
Logging
Visualization
Class Libraries
Framework APIs (package manager, container, task bindings)
Components
Applications (computation)
Data Access
Logging
Visualization
Class Libraries
Framework APIs (package manager, container, task bindings)
Components
Applications (computation)
Data Access
Logging
Visualization
Class Libraries