UNIVERSITY OF OREGON

Scalable Observation System for Scientific Workflows

Overview, Usage, Resiliency, & Security

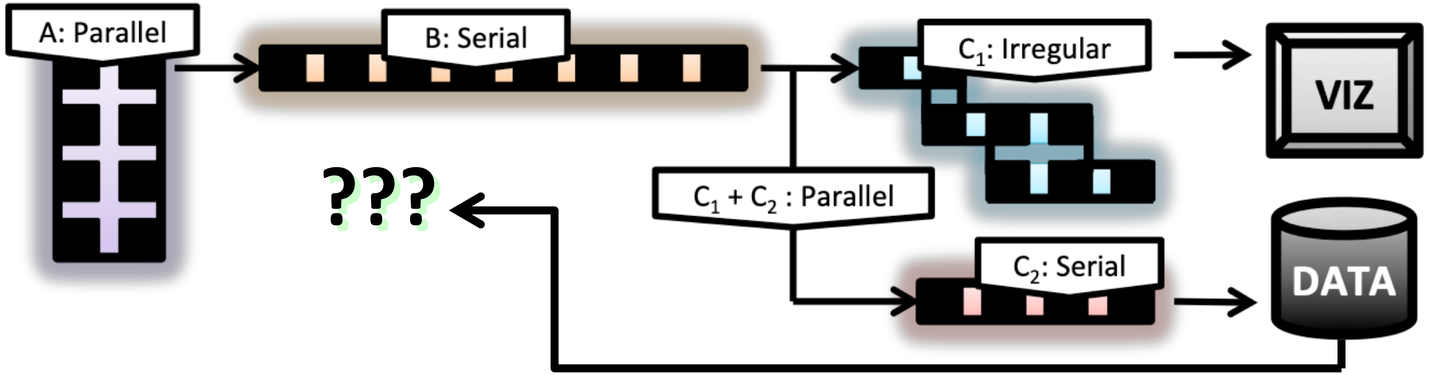
Chad Wood, Kevin Huck

SC18 BoF: Enabling Data Services for HPC, November 13th 2018

Purpose of the SOS Project

- It is reasonable to want to **use** “information” during application execution, and this is uniquely challenging
- **More than just performance data, an integrated perspective**
- Information comes from the application as well as from the environment in which the application is executing
- *Application*: Performance, problem-specific data and metadata, ...
- *Environment*: System state, resource usage, runtime properties, ...
- *Workflow*: Several components running together, sharing resources
- *Campaign*: Access to data within and across allocations

Where SOS is Needed

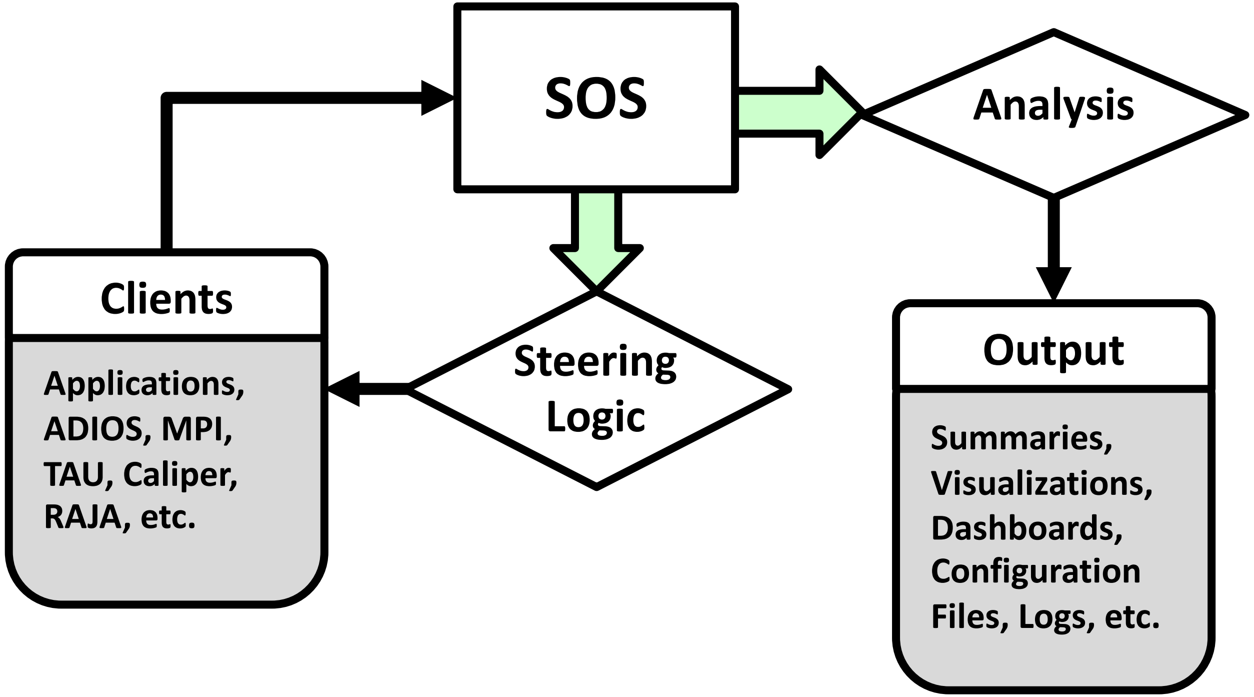


- Multiple components with ***data flow***
- Applications using ***MPI*** and ***filesystem*** for coordination are sensitive to interference
- Complex interactions with ***dynamic behavior***
- Wherever ***variability is evident*** in application behavior, hardware, or shared resource utilization during a run
- Situations where ***acting on information is possible***

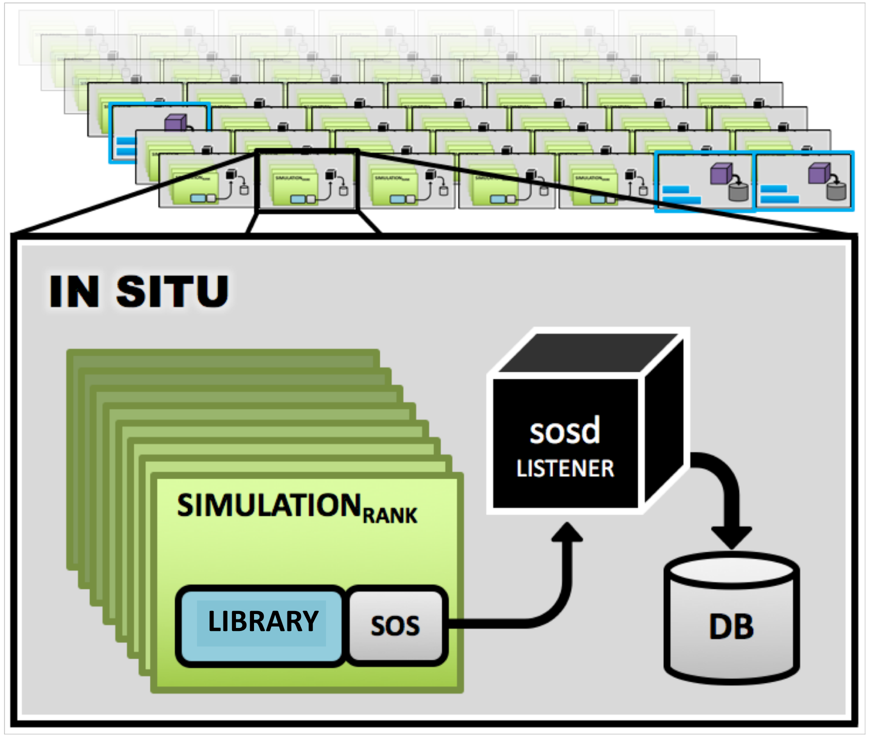
SOS Topology : Logical

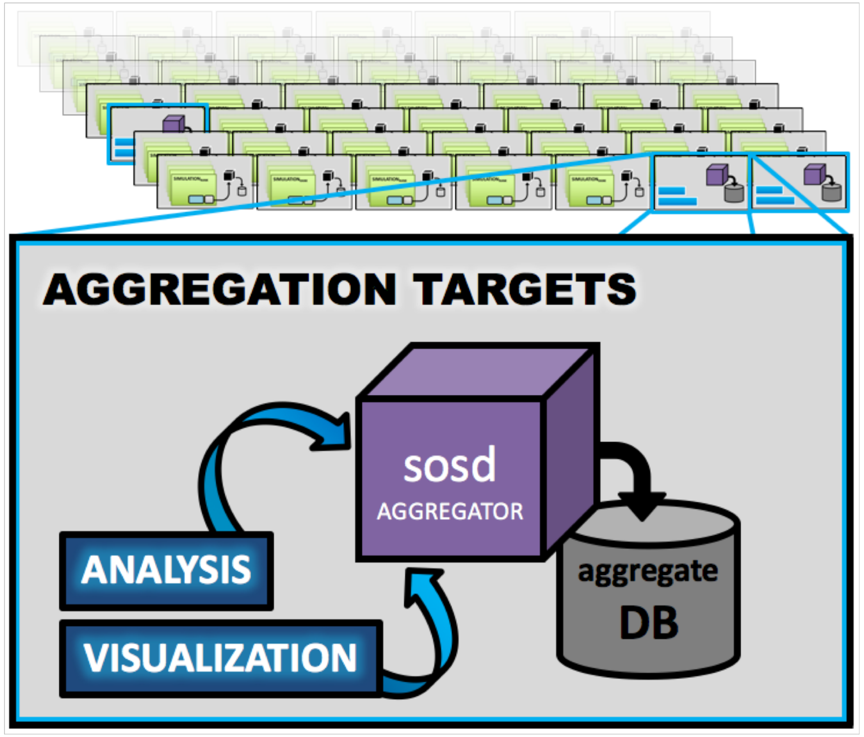
SOSflow functions as a hub for collecting, aggregating, and acting on a variety of information at runtime.

SOSflow's in situ (online) services work together to provide global views and online data analytics in an HPC environment.


SOS
Analysis
Clients
Applications, ADIOS, MPI, TAU, Caliper, RAJA, etc.
Steering Logic
Output
Summaries, Visualizations, Dashboards, Configuration Files, Logs, etc.

SOS Topology : Deployment





SOS as a Service

Integration

- Runs entirely in user space
- Direct use or via tools:
  - **TAU**: Asynchronous periodic output to SOS
  - **Caliper**: Enable SOS service to receive event-driven output
  - **Alpine**: Geometry extraction, projection
  - **Apollo** & **RAJA**: ML for portable performance
  - **Python, C++, C** APIs
- **Challenges**:
  - Coordinating start-up and shutdown w/apps

Resiliency

- SOS is highly configurable to many scenarios, with trade/offs in data persistence, analysis support, and shared resource use facilitated by:
  - File-based SQL DB
  - Memory-only SQL DB
  - Circular Cache
  - Flexible aggregation and analysis topology
- **Challenges**:
  - Drowning by firehose
  - Client-library response to service failure

Security

- How much is needed?
  - Edge security covers user space (for now)
  - Dashboards, Jupyter
  - Inter-machine coordination
  - Long-term storage of collected data
  - What resources are available
- **Current solution**: **Munge**
  - Used by DoE for MPI
  - Practical isolation
  - Offers many more features than we use