



Pacific Northwest
NATIONAL LABORATORY

Proudly Operated by Battelle Since 1965

GOSS: A middleware solution for flexible, interoperable and secure power grid applications

Mark Rice
PNNL

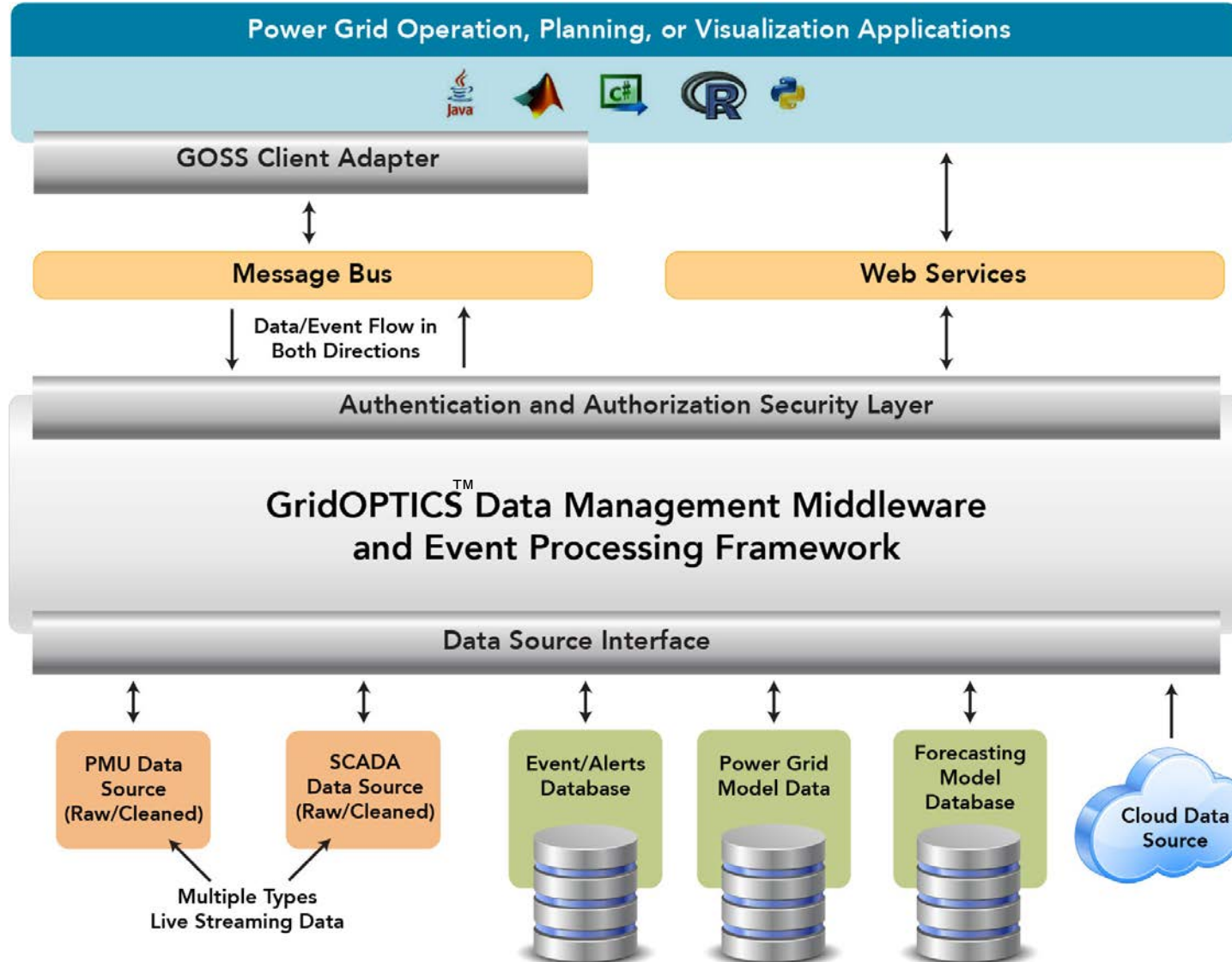
North American SynchroPhasor Initiative
Working Group Meeting
March 24, 2015

PNNL-SA-108927

What is GOSS?

- ▶ GOSS is a middleware architecture designed as a research prototype future data analytics and integration platform
- ▶ What does that mean?
 - Extensibility – ease of integration of new/existing power grid applications developed in many different languages
 - Separates data sources from applications and provides a unified application programming interface (API) for access
 - Quickly make new data available to the many applications already integrated with GOSS
 - Provide redundant data access for improved reliability
 - Real-time – subscription to streaming data and events
 - Scalability & Performance

GOSS Conceptual Architecture

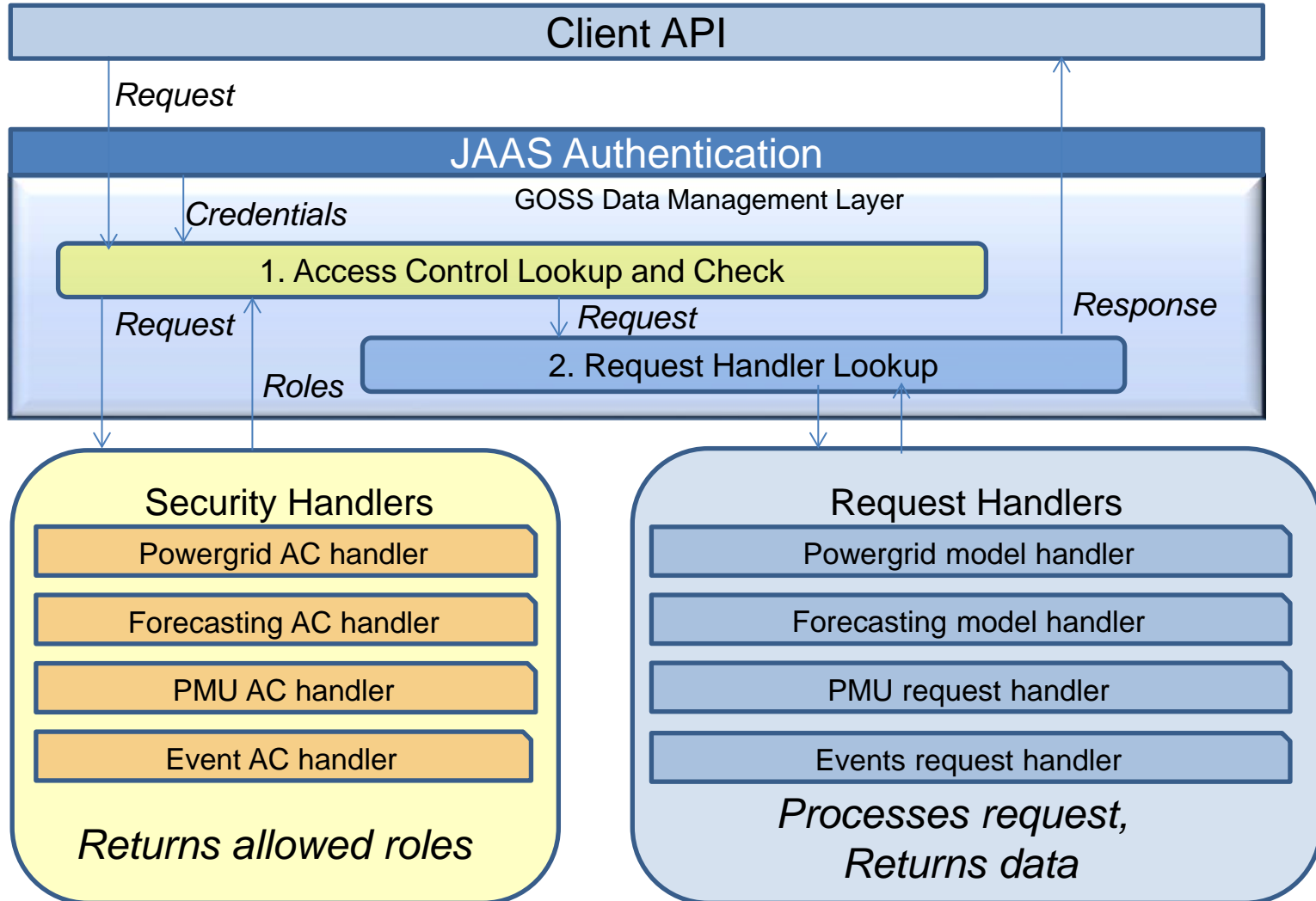


Data Sharing and Export

- ▶ Authorized users can share data and events via:
 - Applications integrated using Client API
 - Web services
 - Web socket API.
- ▶ Data can be shared/accessed synchronously, asynchronous or based on events.
- ▶ Access restrictions can be applied based on
 - Requesting user
 - Data source
 - Data age
 - Data status (raw, processed etc.)
- ▶ Domain/Utility specific access can be provided for higher-level organization viewing

- ▶ Authentication – uses widely accepted tools already integrated into communication platform
 - Java Authentication and Authorization Service (JAAS)
 - Easily substitute login modules
 - Lightweight Directory Access Protocol (LDAP)
 - Open, industry standard application protocol for accessing and maintaining distributed directory information services
 - Transport Layer Security/Secure Sockets Layer (SSL)
 - Cryptographic protocols to provide communication security

GOSS Security & Request Flow



Initial Performance Benchmarking

Chart Area **Time taken (ms) for 4000 requests**



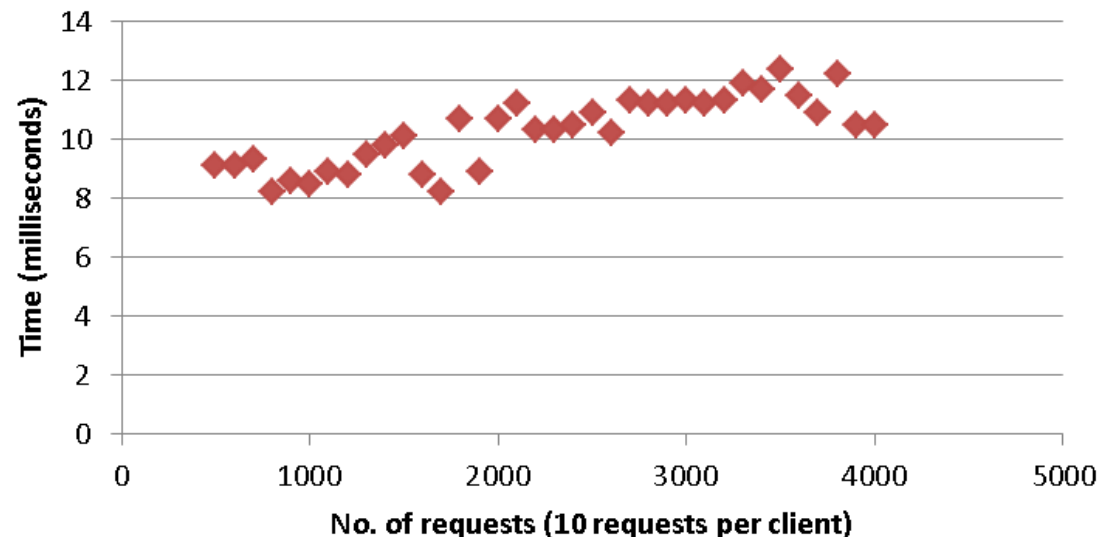
Test 1: Comparison of *average* time taken by data store and GOSS individually in total READ request processing time

- Data size ~700 KB
- Number of requests = 4,000
- Number of Clients = 1
- Each client executed in separate thread

Test 2: Request processing time with increasing number of concurrent READ data requests

- Each client sends 10 requests
- Data size ~700 KB
- Each client executed in separate thread

Processing time per request



Conclusion

- ▶ GOSS – open-source, freely available grid analytic framework
 - <https://github.com/GridOPTICS/GOSS>
- ▶ Integration with existing applications
- ▶ Security
 - Adaptable authentication mechanism
 - Allows fine-grained complex access controls
 - Easy integration of new data sources
- ▶ Performance
 - Per Client Request, processing time is stable even with increasing number of clients
 - Scales well with increasing load

The GOSS Team!



Pacific Northwest
NATIONAL LABORATORY

Proudly Operated by **Battelle** *Since 1965*

- ▶ Bora Akyol bora@pnnl.gov
- ▶ Poorva Sharma poorva.sharma@pnnl.gov
- ▶ Mark Rice mark.rice@pnnl.gov
- ▶ Tara Gibson: tara@pnnl.gov
- ▶ Craig Allwardt craig.allwardt@pnnl.gov

Security Case Studies – Static Access Control

- ▶ Shows PMU data access via a UI
- ▶ Developed to test and demonstrate fine grained access control
 - Configured to use 2 user roles, 3 users
 - Access per PMU is granted to one of these roles
 - Web UI to choose which PMUs to display in a graph
 - Fails and notifies user if access denied for any of the selected PMUs
- ▶ Can view data for multiple roles/utilities

