Pacific Northwest
NATIONAL LABORATORY
Proudly Operated by Battelle Since 1965

An Integrated Security Framework For GOSS Power Grid Analytics Platform

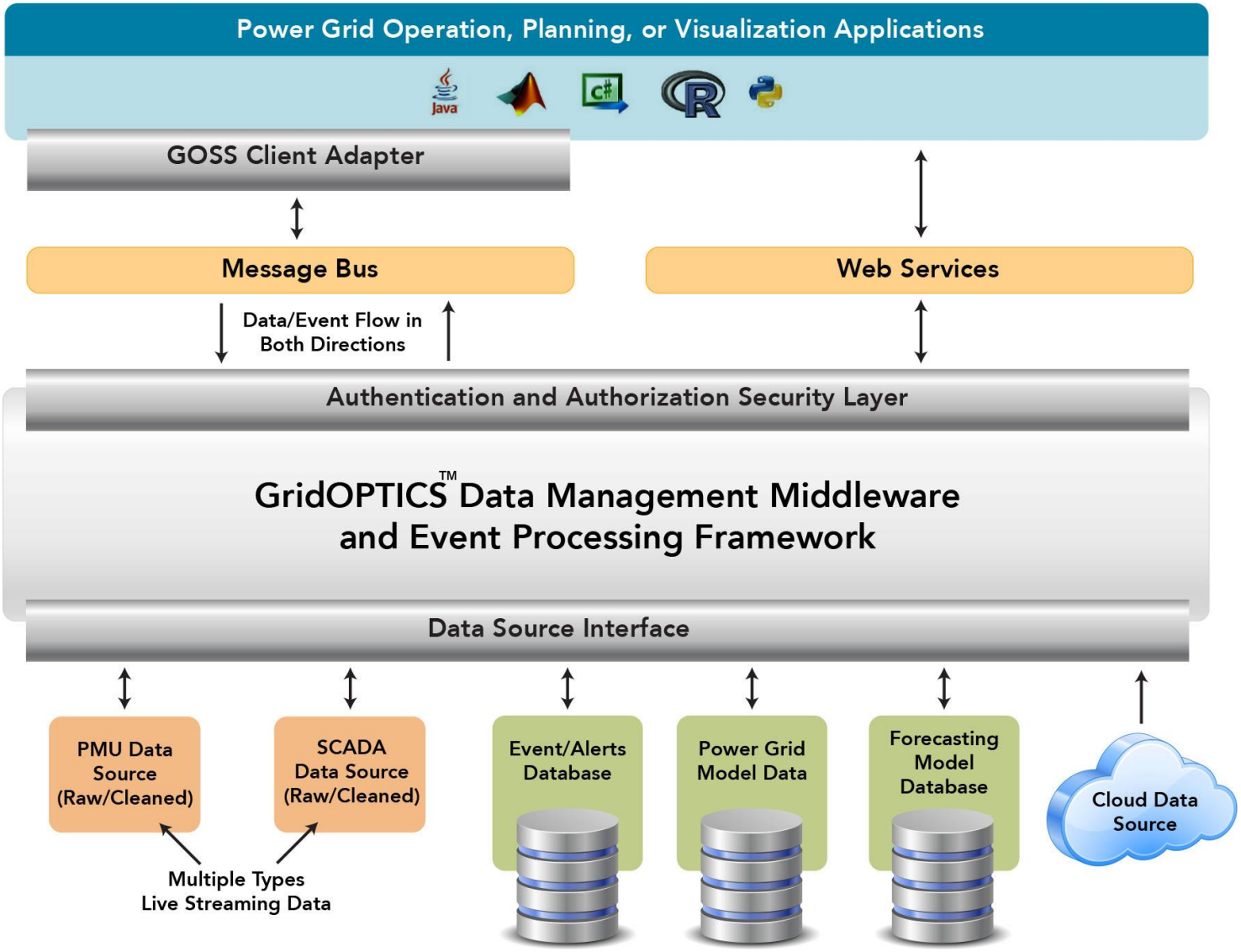
Tara Gibson
PNNL
PNNL-SA-103595

1st International Workshop on Trustworthiness of Smart Grids
June 23, 2014

What is GOSS?

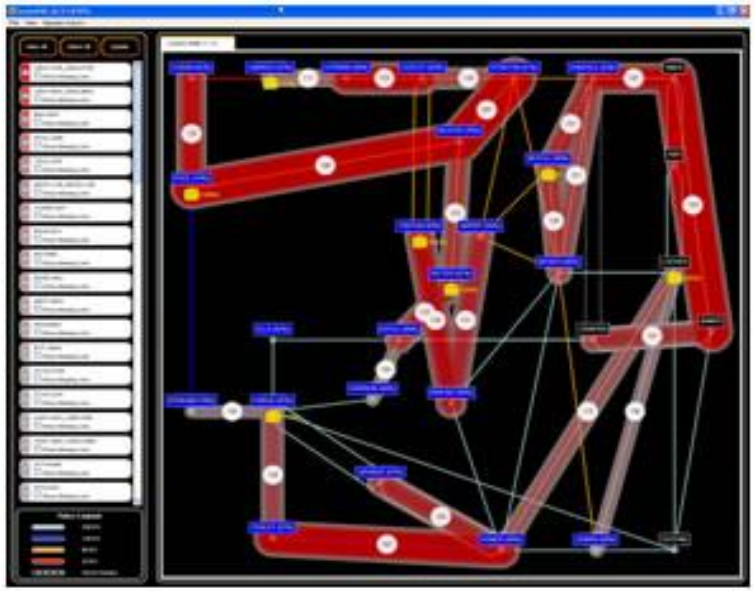
- GOSS is a middleware architecture designed as a research prototype future data analytics and integration platform
- What does that mean?
  - Extensibility – ease of integration of new/existing power grid applications developed in many different languages
  - Separates data sources from applications 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



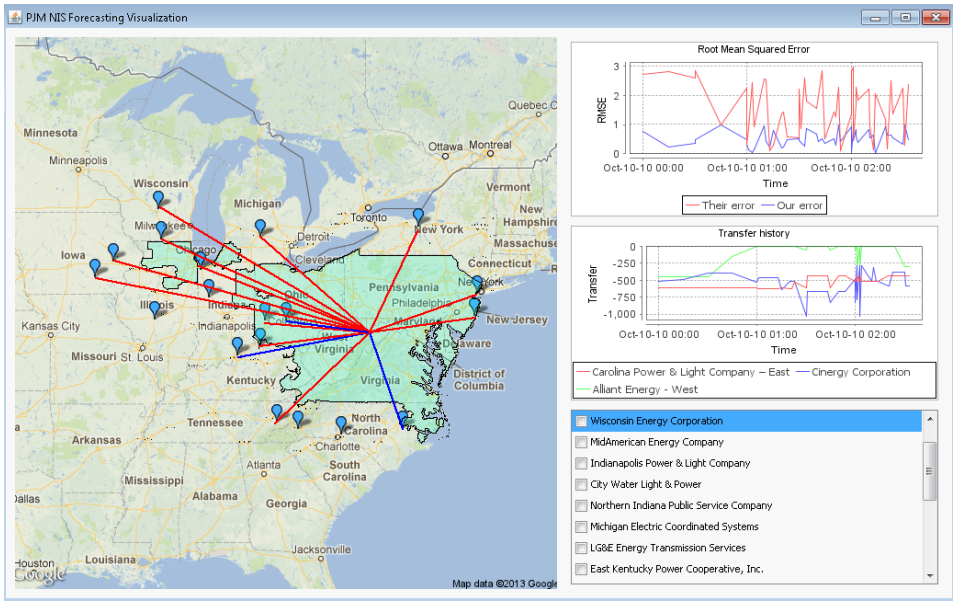
Sample GOSS applications: GCA

- Graphical Contingency Analysis (GCA) - a visual analysis application that aids power grid operators and planners to effectively manage potential network failures (N-1)
- GOSS simplified the application by allowing us to combine all input files (power system model, SCADA, power-flow) into a single data source instead of managing multiple files separately
- Application initiates a request for a topology and allows users to select the model to analyze
- Access is restricted by roles. For each utility, access is granted to a set of roles and the user must be in one of these roles in order to access the data for that utility



Sample GOSS applications: NIS

- Net Interchange Schedule (NIS) displays the sum of the energy import and export transactions between an Independent System Operator (ISO) or a Balancing Authority and neighbors. NIS forecasting (NISF) application was developed to aid the ISOs in economically dispatching the generation resources
- The original application used manually formulated files for the desired time series. With GOSS can use a light-weight client adapter and any time series
- Now able to re-use the algorithm with different data types.
- The input is controlled the same as other PMU data sources, the application will only have access to PMU streams that the user has been granted access to.



Why is security important for powergrid data?

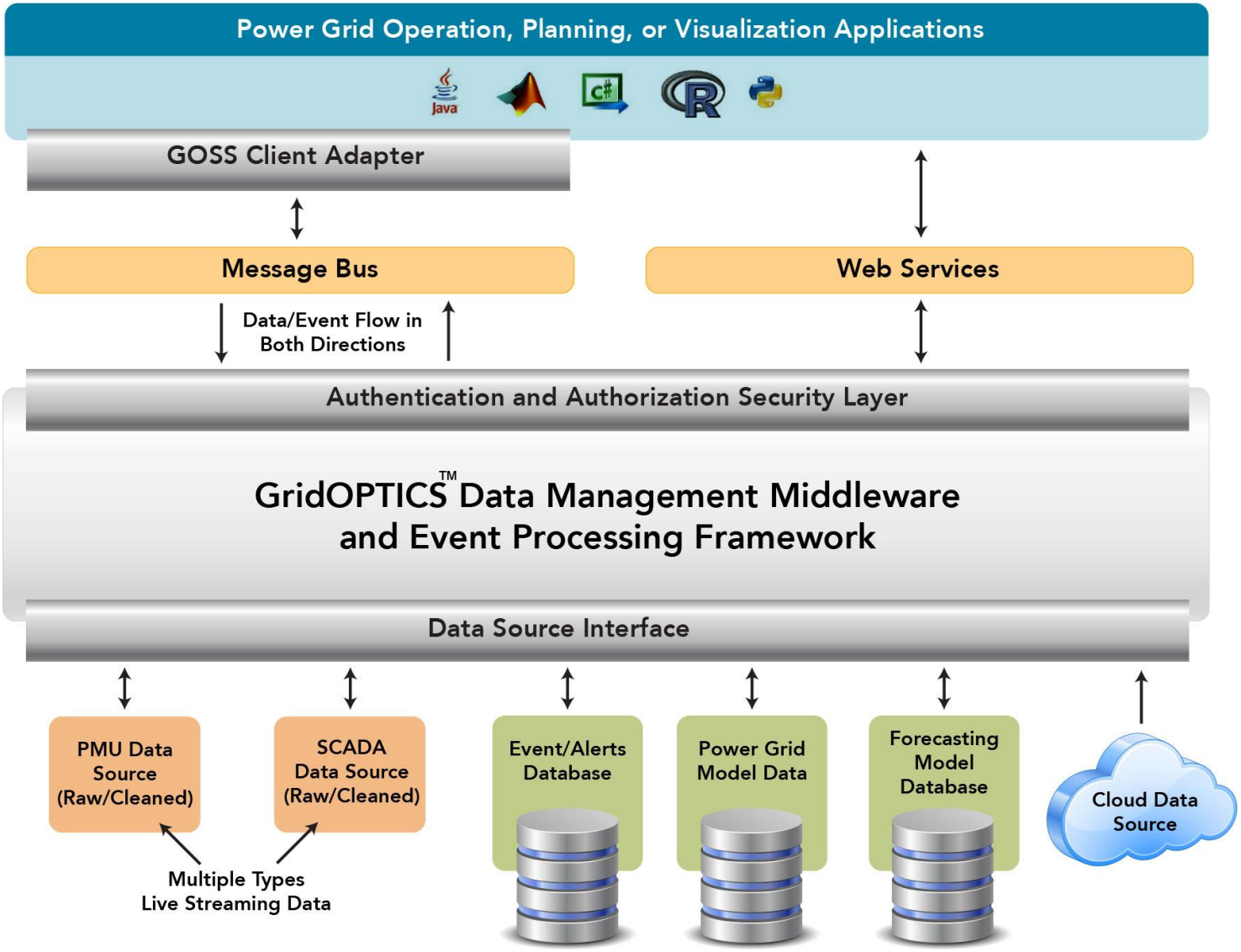
- Data is sensitive, must be protected
- Risk of cyber-attacks
  - Identification of critical points
  - Inject erroneous data
- Critical infrastructure
  - Serious implications for market, stability, and security
- Stability and redundancy are also important
  - Features such as security should not interfere with stability

Security Requirements

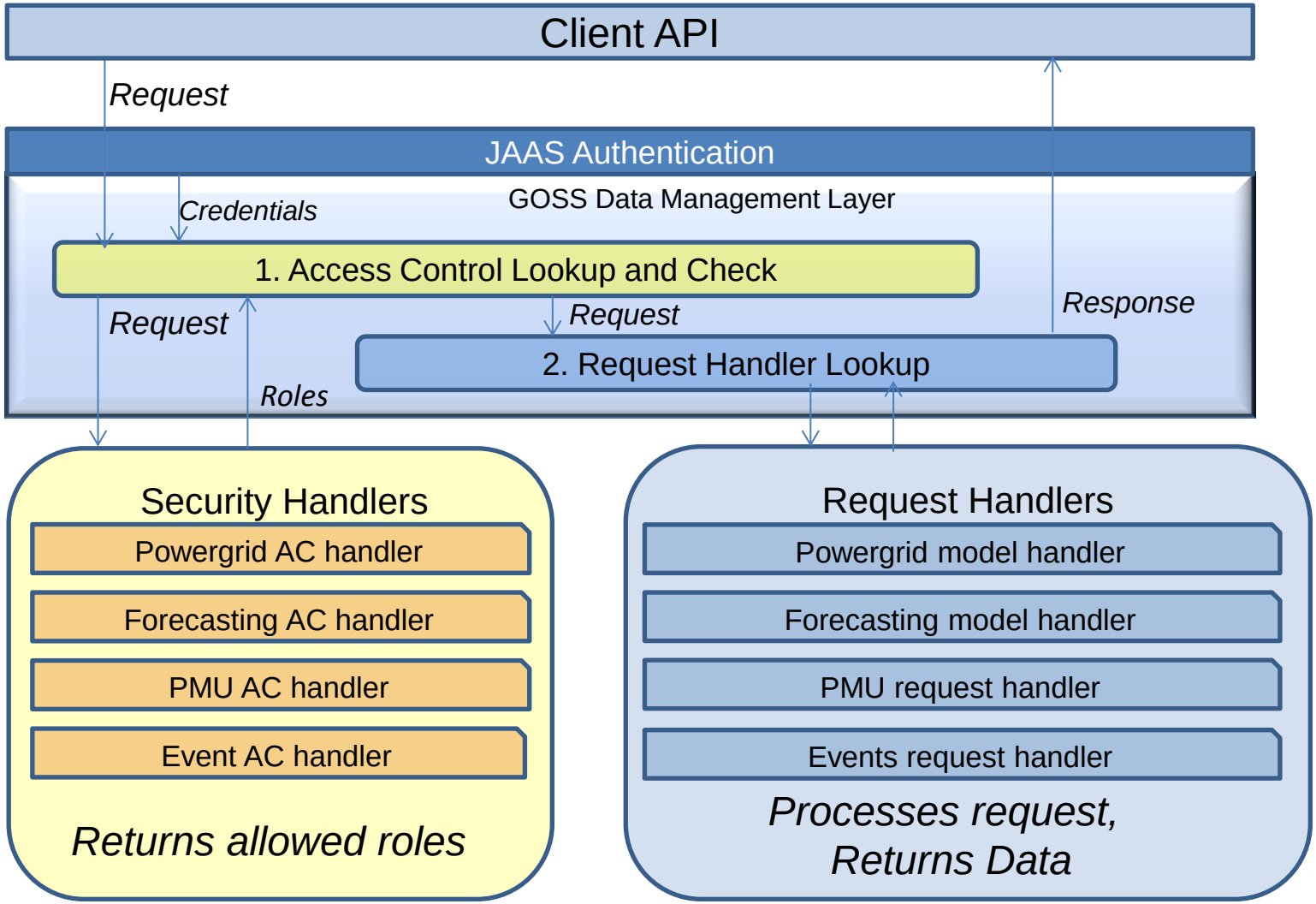
- Authentication should:
  - Provide authentication & identities across multiple domains
  - Prevent intrusions by 3rd party
  - When making a request users/applications should not need to present their credentials many times
- Access Control should:
  - Fine-grained access control - each data source and application can have different security constraints
  - Some users have limited access based on source/age, other users have access to all
  - Access to summarized data may be different than raw data
  - Higher-level organizations can see the data of utilities within their domain
  - Users can’t find out about data/services they don’t have access for

GOSS Conceptual Architecture

Pacific Northwest NATIONAL LABORATORY
Proudly Operated by Battelle Since 1965



GOSS Security & Request Flow



GOSS Authentication

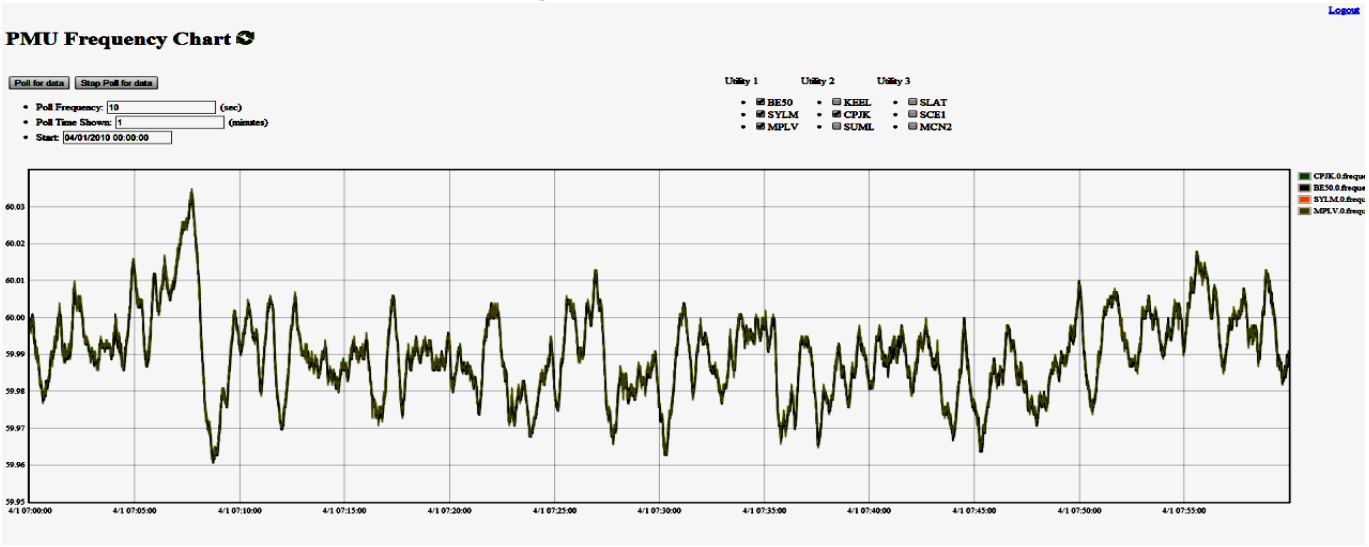
- Authentication – uses widely accepted tools already integrated into communication platform
  - Java Authentication and Authorization Service (JAAS)
    - Easily substitute login modules
  - Lightweight Directory Access Protocol (LDAP)
    - Open, industry standard application protocol for accessing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

GOSS Access Control

- Access Control – customizable for each data source
  - Request Specific Security Handlers
- Security Handlers map request to list of allowed roles
- User verified for correct role access
- Multi-role Access
  - Request combining multiple sources
- Handler implementations for common data types
  - Time series data

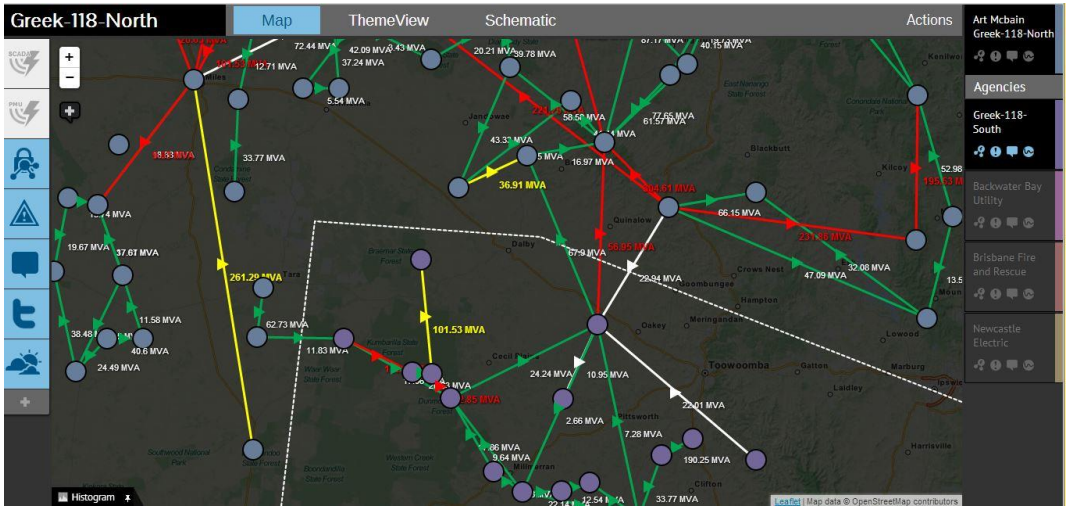
Security Case Studies – Static Access Control

- 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

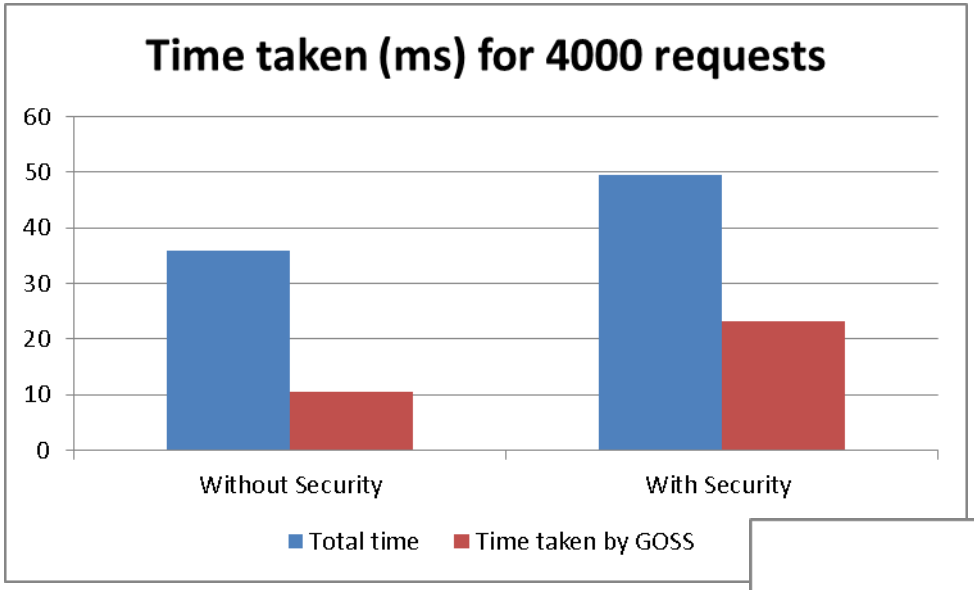


Security Case Studies – Dynamic Access Control

- Shared Perspectives
- Visual application to share live steaming data between users in separate organizations/utilities
- Data is delivered in a CIM compatible format, representing geographic information, sensor data such as angle and voltage, and contingency analysis.
- Two instances representing a ‘north’ and a ‘south’ utility.
- When a user from one utility wants to share data with the other utility the sharing occurs within the Shared Perspective application



Performance Benchmarking Results

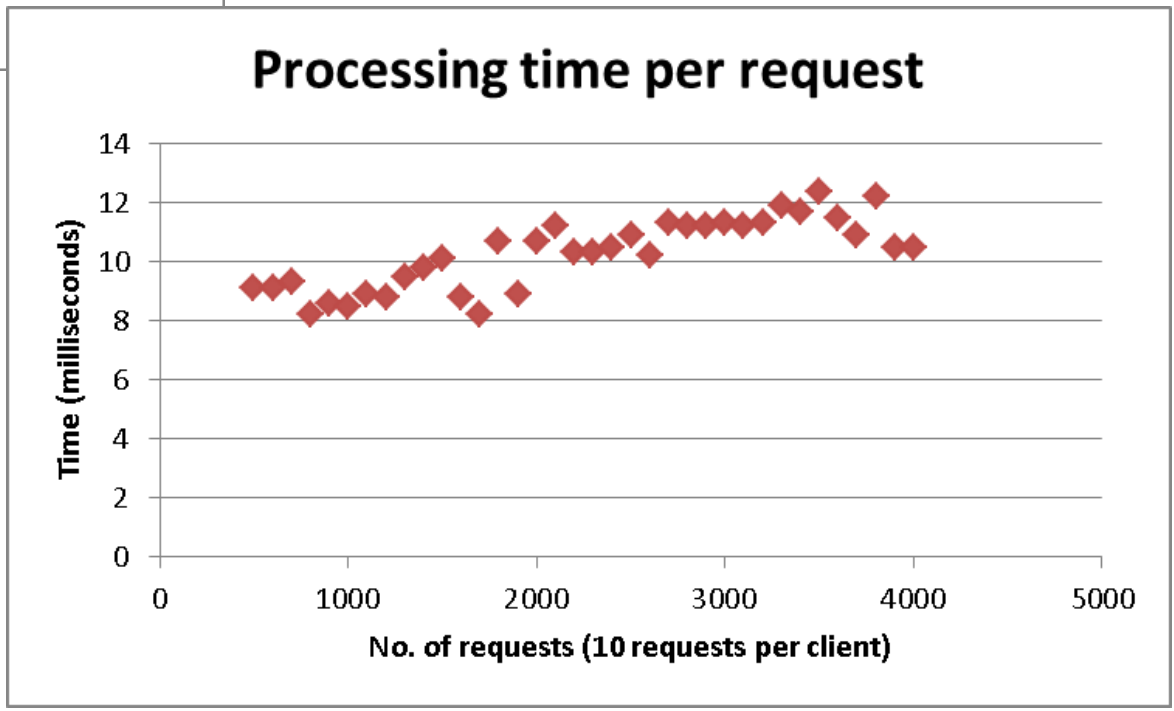


Test 1: Comparison of *average* time taken by data store and GOSS individually in total READ request processing time

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

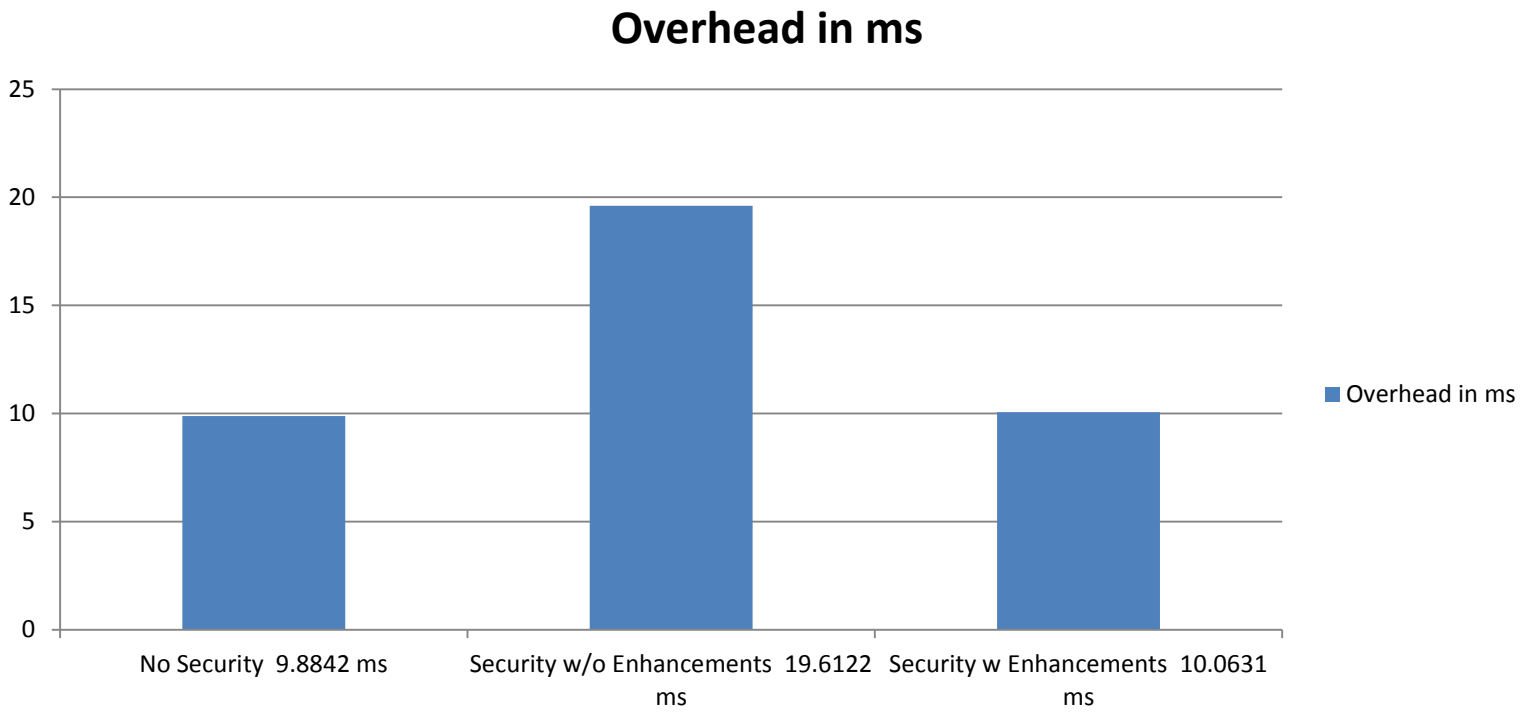
Test 2: Request processing time with increasing number of concurrent READ data requests

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



Synchronous Performance After Enhancements

- GOSS Overheads using same method as previous slide
  - Before enhancements, security adds almost 100% increase
  - After enhancements, reduced to only ~10%



Performance Benchmarking Analysis

- Per Client Request, processing time is stable even with increasing number of clients
- Scales well with increasing load
- Total time spent inside GOSS includes not only data access but also:
  - Data routing between data source and application
  - Query conversion. Generic query format to data store specific query (e.g., SQL)
  - Result conversion. Converting the results to format requested by the application (including object transformation). Eg., JSON, XML, Serialized Object, etc.
  - Security and access control
- Tests show results in “synchronous” access mode. Asynchronous access hides most of these latencies via pipelining.
- Real-time applications likely to use either event-based or asynchronous access.

Performance Issues & Enhancements

- Issues Identified
  - Frequent calls to LDAP to verify authorization
  - Logging adds significant overhead
  - Time spent encrypting/decrypting certificate if used
- Enhancements applied
  - Use Caching for authorization implementation
  - Make sure any additional logging only happens when server is configured in that mode

Other Challenges

- Human element
  - Passwords
- Effective caching
  - Optimization
- Separate Networks
  - Multiple organizations
  - Trust & communication
  - Authentication/Authorization

Future work

- Certificate based authentication
  - Username/password not practical for applications
  - Instead use certificate based authentication
  - Supports multi-organizations
    - Certs from various orgs registered with centralized trust store
  - Roles can be stored within certificates, but may not be as up to date as LDAP store
- Improved Multi-domain support
  - Trust between separate entities
- Redundancy
  - Fault tolerance

Conclusion

- GOSS – open-source freely available data management and application framework
  - https://github.com/GridOPTICS/GOSS
  - Tutorial workshop on July 16 (See reference slides for more details)
- Integration with existing applications
- Security Framework Implementation
  - Adaptable authentication mechanism
  - Allows fine-grained complex access controls
  - Easy integration of new data sources
- Performance evaluation and impact
  - Proper Caching reduced the majority of the performance impact
  - Separate access control checks may still add performance hit and is not something controllable by core GOSS

Thanks to the rest of the GOSS Team!

Pacific Northwest NATIONAL LABORATORY
Proudly Operated by Battelle Since 1965

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

Questions?

- Contact Email: tara@pnnl.gov

Reference slides

Pacific Northwest
NATIONAL LABORATORY
Proudly Operated by Battelle Since 1965

Tutorial Information

- ~July 16, 2014
- Contact Tara Gibson tara@pnnl.gov for more details

Sample GOSS applications: Event Detection

- Detects generator trips from PMU data
- Analyzes PMU data as time ordered segments, originally data stored in files
- Performs data cleaning step to eliminate common errors within the data and reduce false positives
- Saves identified generator trips to a custom data store. These events are viewed through a web application
- Written in R
- GOSS made it very simple to request data for any time period allowing us to process near real-time data. GOSS also enabled us to submit discovered events back to the GOSS framework
- When requesting PMU data to process, we need to verify that the application/user doing the processing has access to all of the raw PMU data that they are requesting.

Example: Time series access control

- Many time series data sources have the same core properties
  - Time
  - Source
- Use Settings in table to map from Request to allowed roles
- Core implementation Time Series Handler allows developers to extend for their own data sources

Time Series Data Access Fields

- dataType (PMU, SCADA, etc)
- source (sensor ID or *)
- age (# of days old, or -1)
- accessLevel (Raw, summary, etc…)
- roles (comma separated list of roles allowed)
- operationsAllowed
- expiration (date this access policy expires)