ATLAS TAG Services

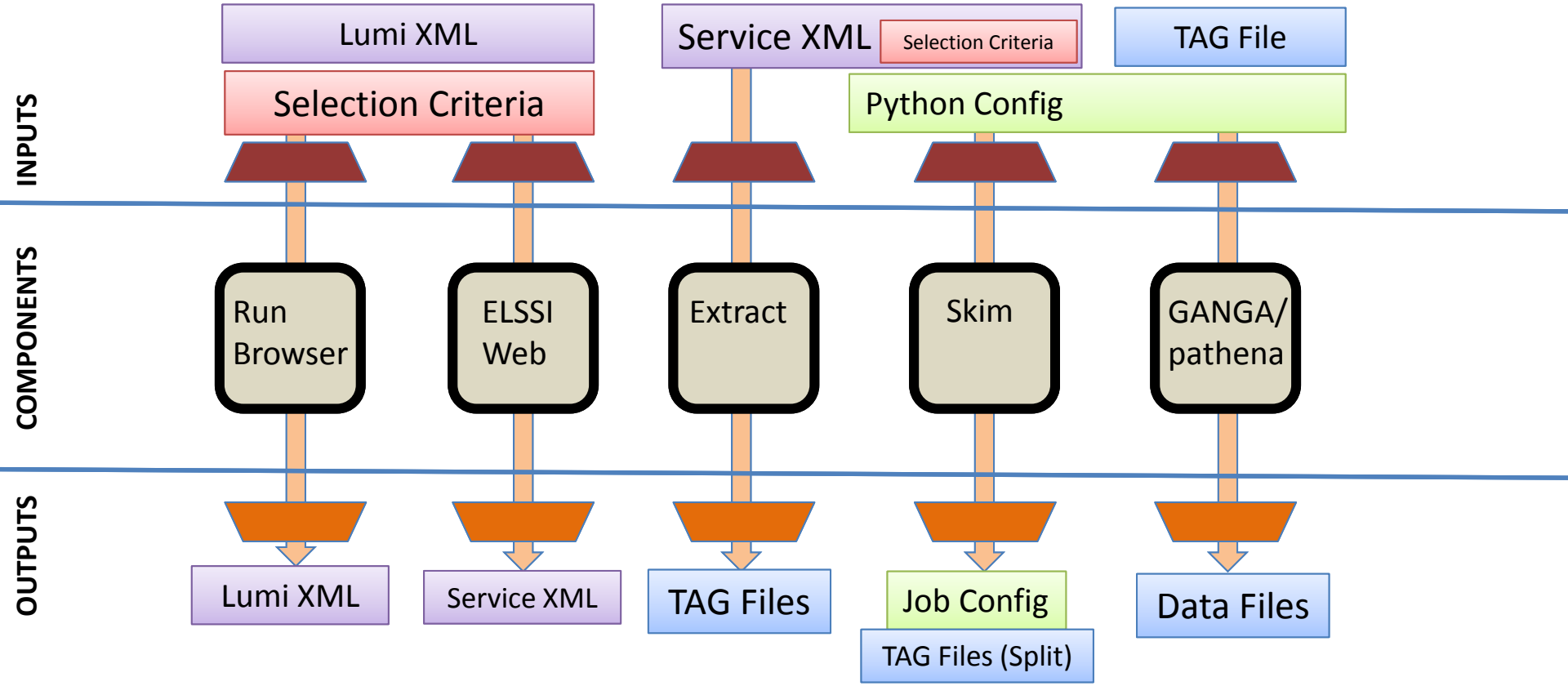
Jack Cranshaw

with support from

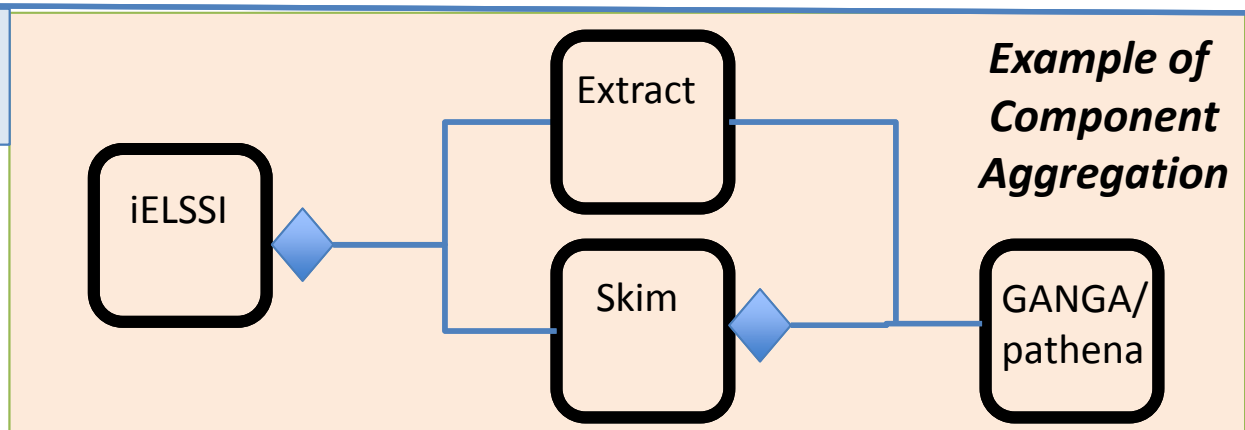
Thomas Doherty, Julius Hrivnac, Marcin Nowak

The ELSSI Suite

- Interfaces
  - ELSSI (Navigator) (QZ)
  - iELSSI (QZ)
  - RunBrowser (RB)
- Services
  - Lookup (MN) depends POOL, TASK
  - Extract (JH) depends POOL/Atlas
  - Skim (TD) depends GANGA/Atlas
  - Web/Helper services (QZ) depends COMA, TASK
  - Histo Service (JH)



All components are available if appropriate inputs are provided.



Exchange Formats

- Services
  - Services need to exchange information
  - Services use various programming environments.
  - Use a non-language specific format with many third-party tools. XML is a natural choice.
- POOL Utilities
  - Utilities share many common CLI arguments.
  - There are cases where a user may want to keep 90% of the arguments the same and modify one, or even use the same arguments with a different utility.
  - A generalized C++ CLI argument architecture (Args2Container, CmdLineArgs2) was developed by ATLAS and migrated into the POOL code base.
  - The state of the Args2Container was made persistent by writing it to an XML file.

Analysis Model Context

- TAG content and multiple use cases presented by Tulay on Monday in PAT session.
- Two general use cases presented by Mark on Tuesday in Distributed Analysis session.
  - Coarse selection, touch all/most payload files
  - Fine selection, touch only a few payload files

Lookup Service

- (Event) Lookup Service (alias Event Picking)
    - Accessible from command line (python) and Athenaeum web site
    - Currently installed only at CERN (but available worldwide and remote access is not much slower). Can be replicated to other sites hosting a TAG database replica
  - Current main user is Panda
- Functionality: for a list of event numbers, the service returns a list of file GUIDs where the events are located.
    - TAGs currently support RAW, ESD, AOD and TAG file GUIDs.
    - All physics streams are searched, unless a subset is specified (much faster)
  - Example usage:

```
runEventLookup.py --runs_events "152713 9, 152168 11961"
--stream "express_express„
840C0BBD-523C-DF11-9878-0030487C6B94 StreamAOD_ref
...
```

- Essentially a packaged call to CollListFileGUID using xml input option.
- SVN
  - Server: atlasoff/Database/TAGPoolServices
  - Python client: atlasoff/Database/AthenaPOOL/AtlasCollectionTools
- Many times uses local POOL patches or POOL version ahead of ATLAS release version

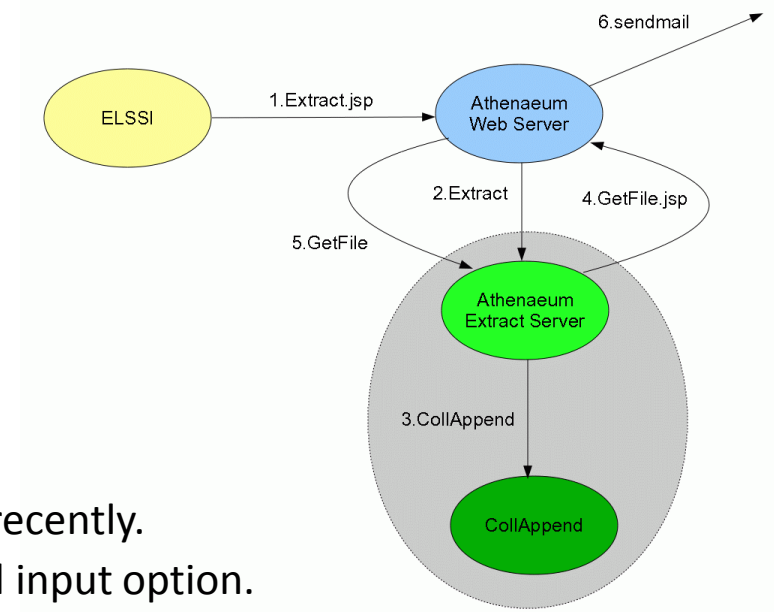
Lookup Service

- Performance and Latency
  - Lookup rate is roughly 100 Hz
- Scalability
  - Run-based lookup not expected to scale past 1000s of events., but period usage may alleviated this.
  - Web server performance problems may require output splitting.

Extract Service

- Extract service provides input TAG files for athena jobs based on TAG queries
  - Most of the logic is in java/xlst which is built into a web archive and deployed into a j2ee container.
  - Jobs run using a python and shell scripts on atlddm10 for production or in private accounts on other machines for development.
- Accessible from web (iELSSI) or command line
- Input is a config file in xml format

```
<athenaeum>
  <email>
  <extract extract_id=x>
  <atts>
  <collections>
    <collection name="LLL">
    <table>oracle:/conn/table</table>
    <query>
    <metadata name qual iov ver>
    <LumiRangeCollection>
  <profile>
```

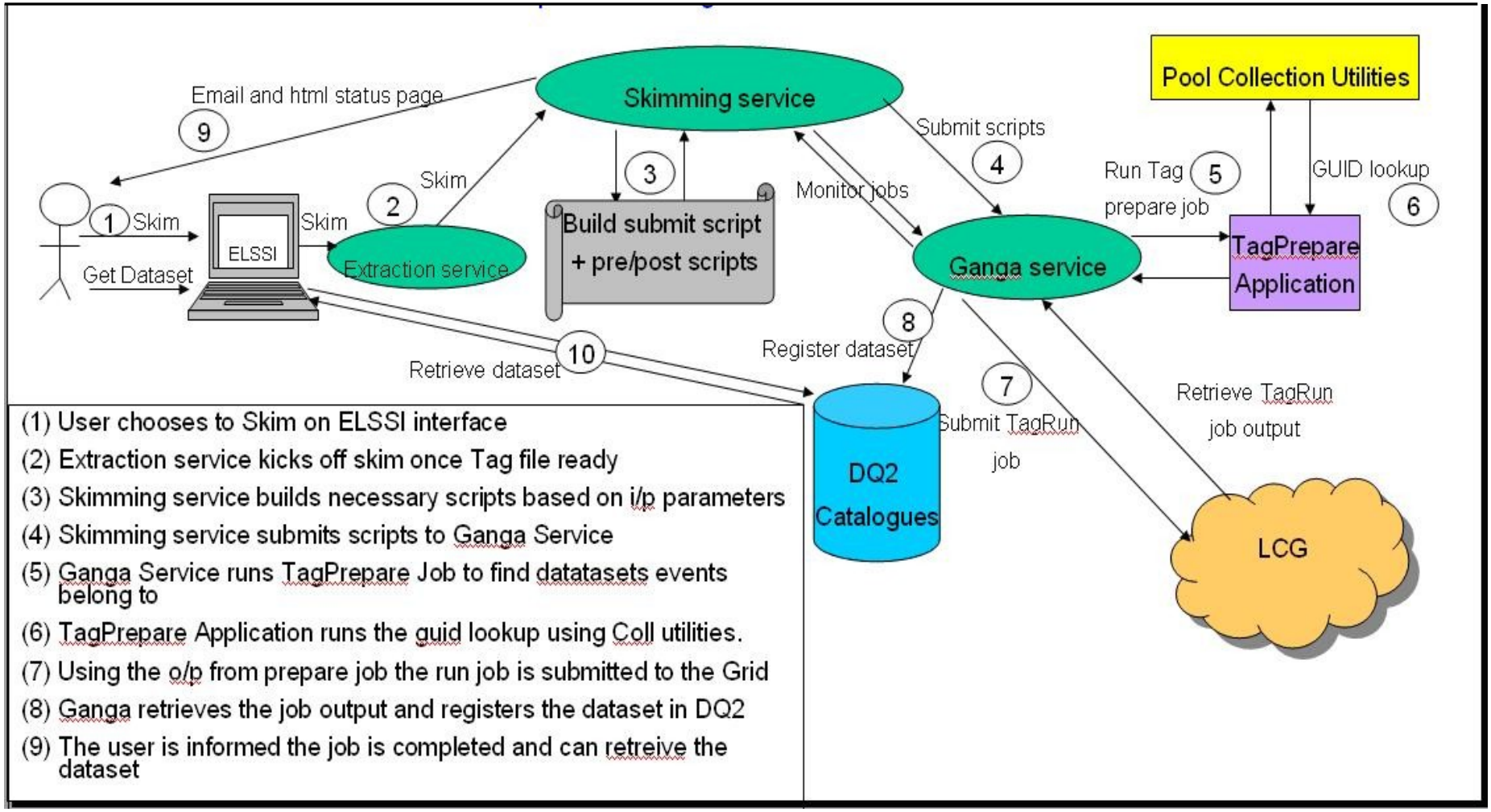


- Lots of development on configuration and output recently.
- Essentially a packaged call to CollAppend using xml input option.
- Output available through web server: wget.
- SVN:
  - Manager: atlusr/hrivnac/Athenaeum
  - Server: atlasusr/Database/TAGPoolServices
- Uses Atlas release or local patches to an Atlas release

Extract Service

- Performance and Latency
  - Extraction rate is roughly 1-10 kHz.
  - Large selections could be several million events, so extraction times are minutes.
  - Extract should be called asynchronously.
- Scalability
  - Scalability theoretically handled by submitting multiple CollAppend jobs in parallel, which should scale as well as Oracle query planning scales.
- Issues
  - Extract not fully functional without ELSSI computations
    - Luminosity
    - Overlap removals
    - “Job” splitting based on TASK
  - Distributed queries only available in ELSSI

Skim Service



Skim Service

- Takes the output of Extract and runs a grid job to generate one of several standard outputs.
  - Most of the logic is in Skimming .py
- Accessible from web (iELSSI) or command line
- Input is a config file in xml format

```
<skim>
  <inputs>
    <extract_id>
    <tag_url>
    <tag_dataset>
  <inputref>
  <tool>ganga</tool>
  <release>
  <options>
```

- SVN:
  - Manager: atlusr/hrivnac/Athenaeum
  - Server: atlasusr/Database/TAGPoolServices

Skim Service

- Performance and Latency
  - Production of submission scripts dominated by TagPrepare times. Various optimizations being considered. Otherwise, performance dominated by DA tools.
- Issues
  - More options on what flavor of DA tool to use.
  - Better support for user optimizations
    - User options
    - User code patches
  - Improve job resubmission logic.
  - Use transforms instead of job options.

Releases and Testing

- All code resident in SVN.
- Tagged versions deployed to production
- Dependencies tracked in developers' heads
  - Plan to put this into the TASK database
- Change control
  - Development done in user test areas
  - Integration and production done in tagops@atlddm10
- Testing
  - Component testing in various (un)documented forms available for all services.
  - Integration testing is currently through running the tutorial.

Resources / Deployment

- CERN
  - Production server maintained by IT: atlddm10.cern.ch
  - Development server available to individual developers: voatlas69.cern.ch
- Tier 1's
  - Only ELSSI at the moment
  - Deployment model being developed
  - Each is effectively just another site service in a VO box.