ServiceX user experience

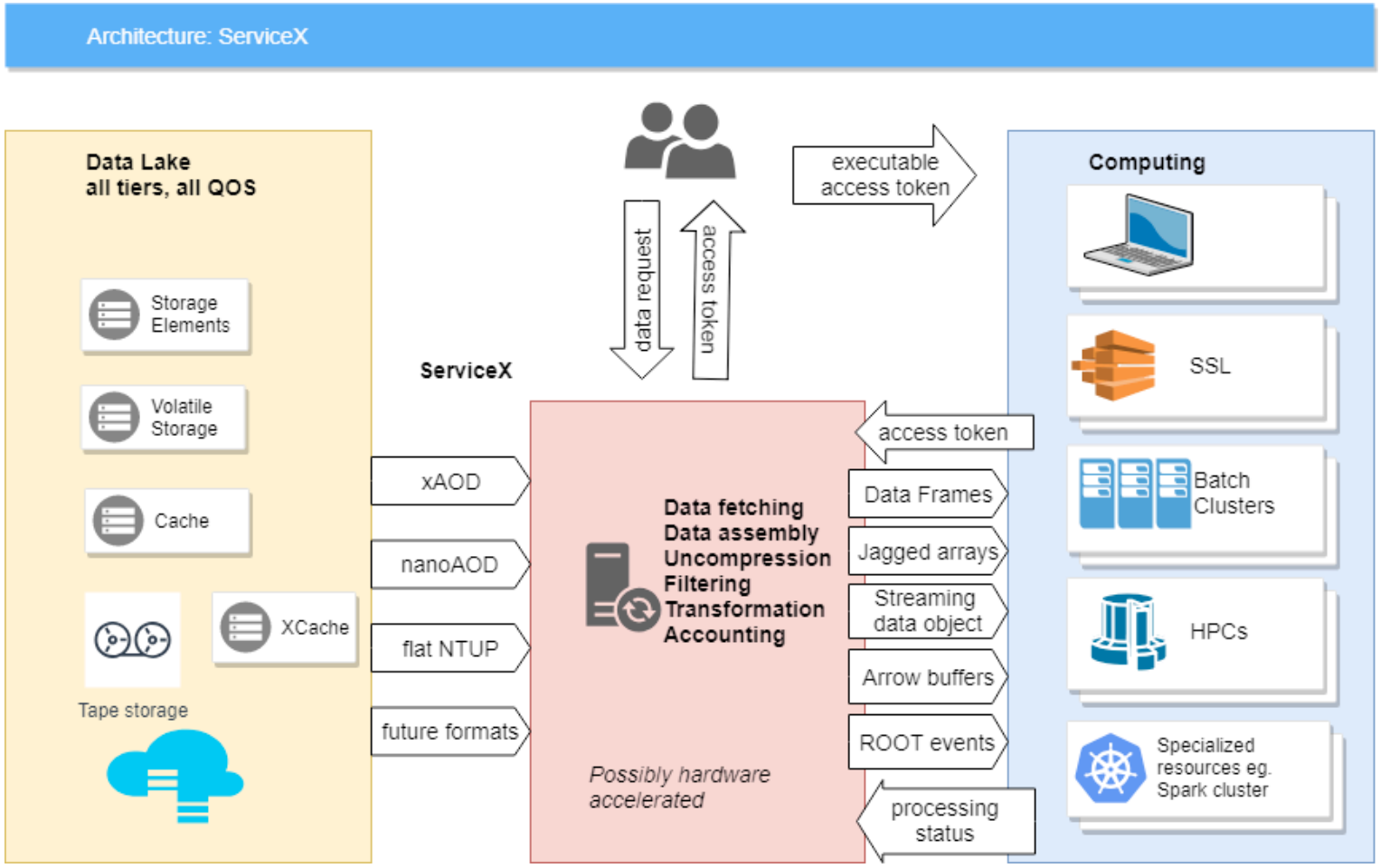
Tal van Daalen

AGC Workshop

May 3rd 2023

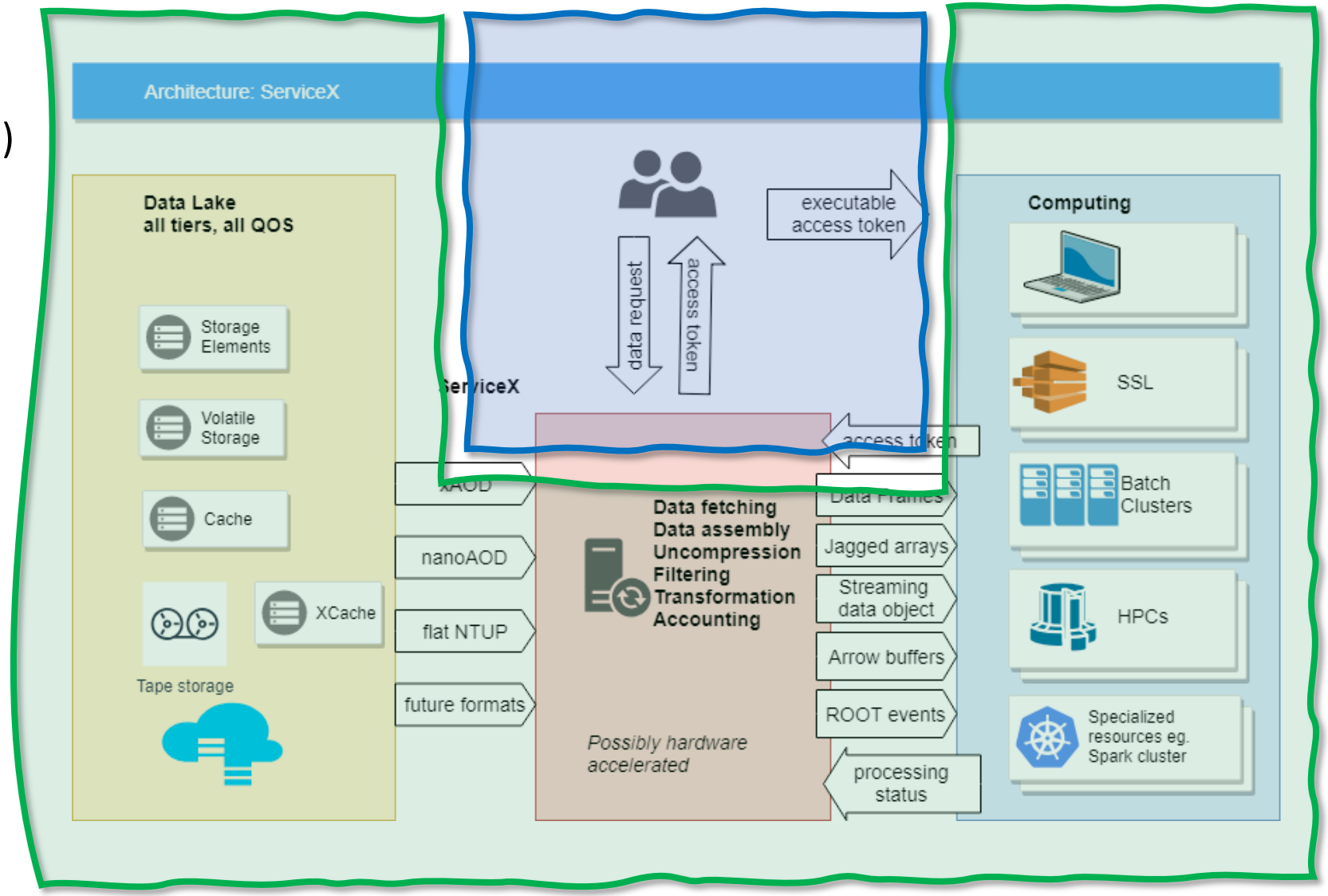
Columnar data delivery with ServiceX

- Efficient delivery of columnar data for range of physics analyses
- Allows user to access data – located anywhere – perform on-the-fly transformations into multiformat files
- Use declarative analysis language func_adl to filter, select, compute new variables, to only run analysis on the necessary events

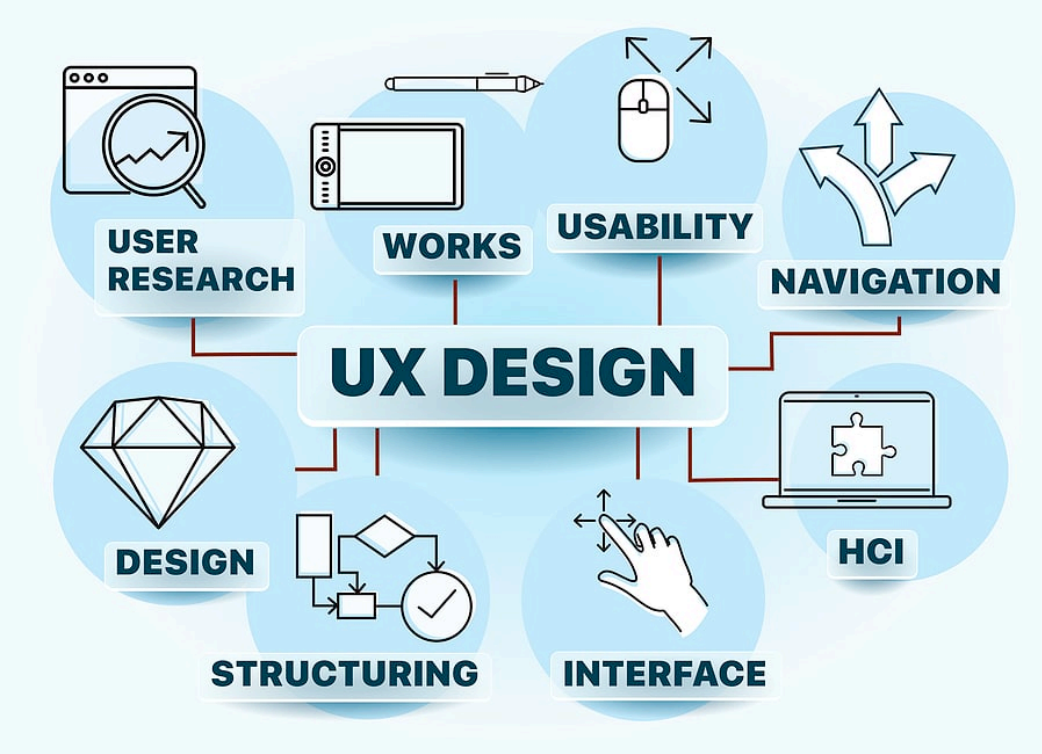


Columnar data delivery with ServiceX

- Clear divide between **user-accessible surface** (**frontend**) and **underlying processes** (**backend**)
- From UX perspective: ideal amount of insight user should have into what is going on?



User experience



- **Main goal of this talk:**
- Identify short + long-term goals towards making ServiceX (+adjacent packages) as **user-friendly** as possible
  - Crucial for establishing as viable alternative to conventional toolkit in coming years
  - Important for maintaining active user base afterwards
- Compile at end of workshop – towards an AGC showcase event
- This is an **interactive session**. Participation is encouraged! (especially from fellow physicists!)

Demo time

- Dummy analysis on ATLAS open data – **rediscovering the Higgs boson!**

- Feel free to note down points you think are unintuitive, overly complex (or overly simplified) in google doc

Post-demo discussion

- Notes?

- Some pointers borne from experience

- **ServiceX-specific:**
  - Unintuitiveness of some methods (dummy dataset, .value(), etc.)
  - Reporting of total number of files to be processed can take long
- Debugging and error reporting
  - Frontend-specific errors are not very descriptive
  - Errors relating to new backends are stored in Flask database (for how long?)
  - Running 1000s of jobs? Good luck finding the log for that one failed job…
  - Example: ATLAS BigPanda stores all stdout etc. output for grid jobs practically indefinitely – can be tedious, but reliable

Post-demo discussion

- **General AGC pipeline**
- Everything that is possible now (ROOT-based) should remain possible
- Big outstanding holes:
  - Systematics
  - Anything (semi) data-driven?
    - Data-driven background? Reweighted MC?
  - Usual analysis checks (Cutflow? Duplicate events?)
    - Probably possible, but risk of creating big gap in complexity between streamlined and user-designed methods
    - Can tank user-friendliness if analyzer has previously done similar checks with e.g. ROOT and now has to take completely different approach
  - What is your least-favorite part of an analysis? We need to make this as painless as possible while not sacrificing transparency
- Long-term user support:
  - Examples: ATLAS DAST, ROOT forum
  - Similar setup not worth it currently, but should move from private help messages to e.g. mailing list-based support

Thanks for listening!