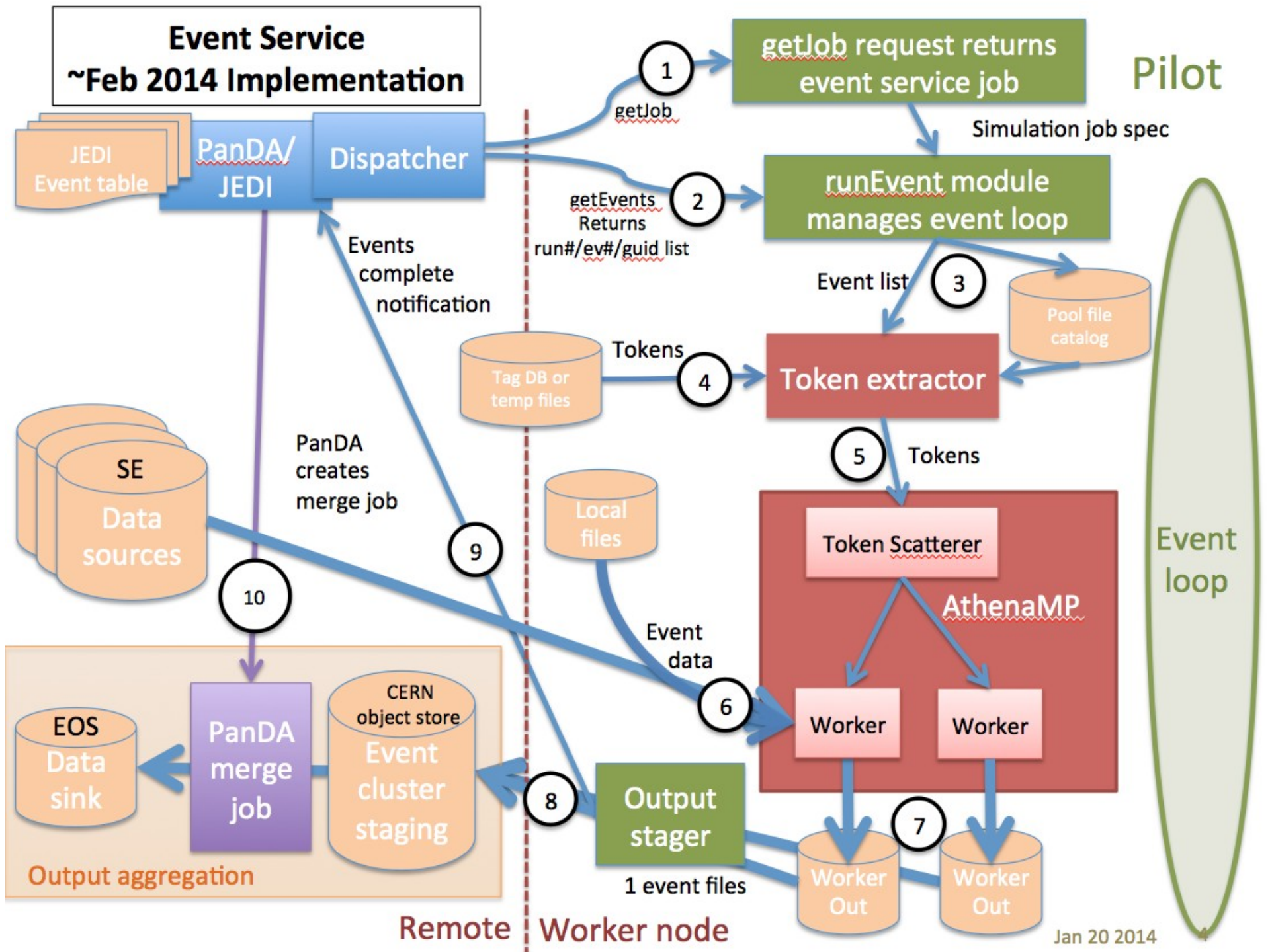
Event Processing for the Event Service

Status
Issues/Concerns
Ideas for Future Development

Vakho Tsulaia
LBL

US ATLAS S&C Meeting
Berkeley, August 21, 2014


Event Service
~Feb 2014 Implementation
Pilot
JEDI
Event table
PanDA/
JEDI
Dispatcher
1
getJob
getJob request returns
event service job
Simulation job spec
getEvents
Returns
run#/ev#/guid list
2
runEvent module
manages event loop
Events
complete
notification
Event list
3
Pool file
catalog
Tag DB or
temp files
Tokens
4
Token extractor
PanDA
creates
merge job
5
Tokens
SE
Data
sources
Local
files
Token Scatterer
AthenaMP
Event
loop
9
10
Event
data
6
Worker
Worker
CERN
object store
EOS
Data
sink
PanDA
merge
job
Event
cluster
staging
8
Output
stager
7
Worker
Out
Worker
Out
Output aggregation
1 event files
Remote
Worker node
Jan 20 2014

Status

- End-to-end prototype tests are currently running with nightly software releases (19.X.0)
  - **One known issue:** the Output File Sequencer keeps all output files open until the end of the job, and closes them all at finalization
    - Peter working on the fix

- One new development recently:
  - Support for reading event tokens from the **Event Index** has been added to the **Token Extractor**
    - Token Extractor is a “standalone” utility within Event Service, which converts positional event numbers from **Event Ranges** into corresponding POOL tokens

Connection with the Event Index

- POOL tokens for few EVGEN events have been uploaded on the **prototype Event Index @ CERN**
  - The prototype is not very stable
- Reading of these tokens from the Token Extractor is OK
- For the pre-production test:
  - POOL tokens for **all events** used in the test need to be **available in the Event Index**
  - Event Index needs to be **accessible** for jobs running **outside CERN**
    - Current limitation with the prototype Event Index
- Reading POOL tokens from **TAG files as a backup solution** for the pre-production test (not desirable)

Meta-data

- ERROR from the `CutFlowSvc`, which cannot find `EventBookkeeperCollection` in the output meta-data store
  - Jack working on the fix
- **In general, we need to have an infrastructure, which will properly handle meta-data in the event-based workloads!**
  - Not a simple task. Requires time and manpower
- Perhaps not that difficult to handle meta-data for G4 simulation. **Getting much more complicated for other types of jobs**
- *Meta-data produced by the end-to-end ES test is yet to be validated*

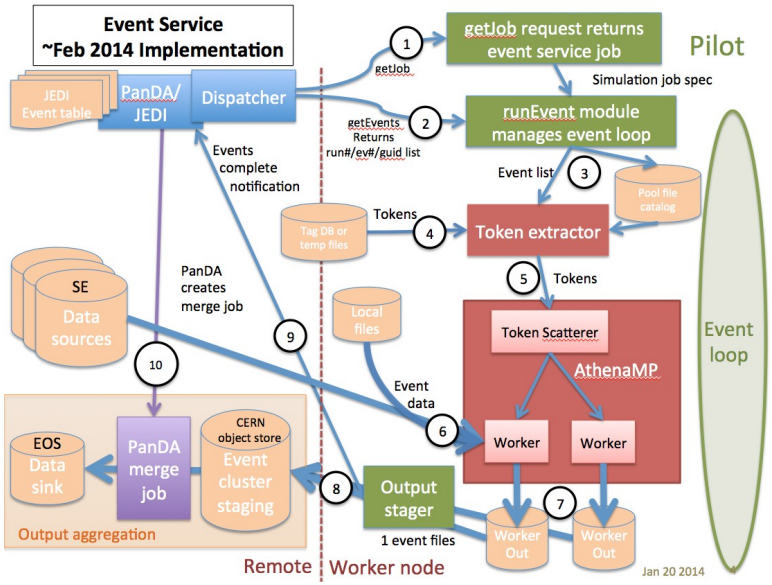
Validation

- HITS produced by the Event Service G4 simulation jobs need to be validated
  - **NO validation done so far**
  - **Debugging:** A **quick comparison** of HITS produced by identically configured ES and Serial jobs (**very few events**). Results are encouraging ☺

- We should plan to have validation runs once we are able to produce large samples with the Event Service
  - **Stability and scalability** aspects of the Event Service need to be addressed

Error Handling

- The end-to-end test has already helped us a lot in identifying and fixing various problems
- But there are still quite few places where we yet need to come up with error handling strategy
  - Event Service is a **complex** system



- Just one example scenario:
  - Token Extractor tries to read event tokens for given GUID from the Event Index
  - But event tokens for that GUID have not been uploaded to the Event Index
  - Retrying such Event Ranges is a waste of time. No matter how many times you retry, the tokens still won't be in the Event Index
  - How to proceed?

Performance

- We have not yet looked into performance aspects of the AthenaMP payload inside Event Service
  - **Example:** how much CPU-time do the workers waste either waiting for the next event range, or reading event data from the remote file?

- One possible area for performance improvement: make the **Token Scatterer** process **multi-threaded**
  - In order to **parallelize** the tasks of **token retrieval** from the Token Extractor and **token delivery** to the worker processes
  - Requires development

Longer-term development: **Shared Reader**

- **Current model:**
  - A specialized AthenaMP sub-process – **Token Scatterer** – sends Event POOL Tokens to the worker processes
  - Each worker **reads event data from the remote file independently**
- In particular, for the G4 Simulation jobs this means
  - Each worker reads **entire basket with many events** in EVGEN format
  - Only one event is used. **The rest of the basket is thrown away**
- With **Shared Reader (aka Event Source)**
  - **Just one process reads event data (entire baskets). Remote file access for reading is contained withing single process.**
  - **The reader extracts individual events and sends them over to worker processes**
  - Can possibly eliminate the need in Token Extractor
- It's **relatively simple** to implement Shared Reader for **EVGEN** events (still a major development). **Much more complicated would be to implement the same for other event formats**

Longer-term development: **Shared Writer**

- With **Shared Writer (aka Event Sink)** AthenaMP workers don't make output files by themselves
- Instead they send event data over to a specialized process – either local or remote – which writes them out to the disk
- This will **not eliminate the merging altogether**, but has **a potential to reduce the number of output files**
  - Small one-event HITS files created by ES in its current implementation
- The implementation of such **Shared Writers** will arguably be **more complicated** than the implementation of **Shared Readers**