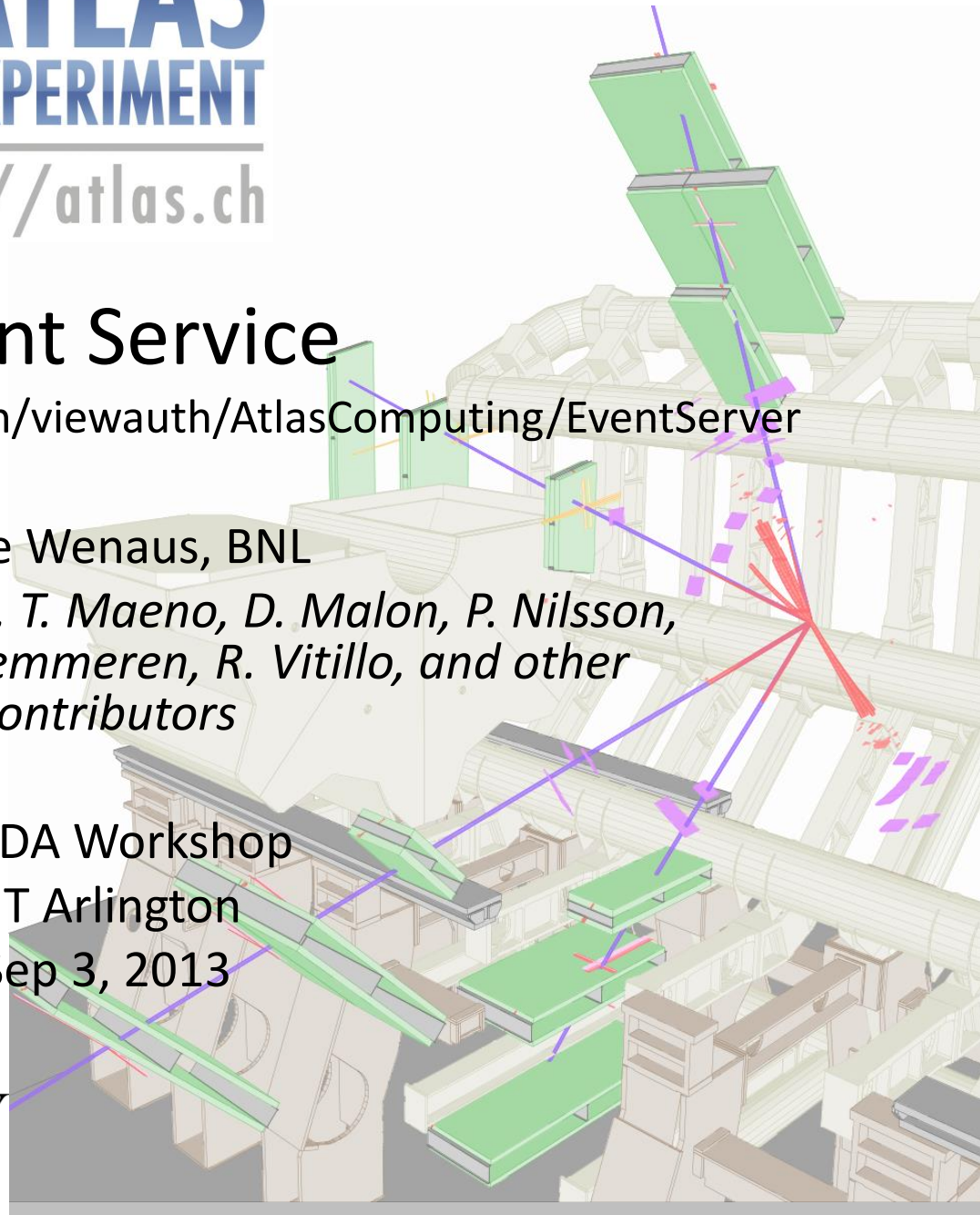


Event Service

https://twiki.cern.ch/twiki/bin/viewauth/AtlasComputing/EventServer

Torre Wenaus, BNL

For P. Calafiura, K. De, T. Maeno, D. Malon, P. Nilsson, V. Tsulaia, P. Van Gemmeren, R. Vitillo, and other contributors

PanDA Workshop

UT Arlington

Sep 3, 2013

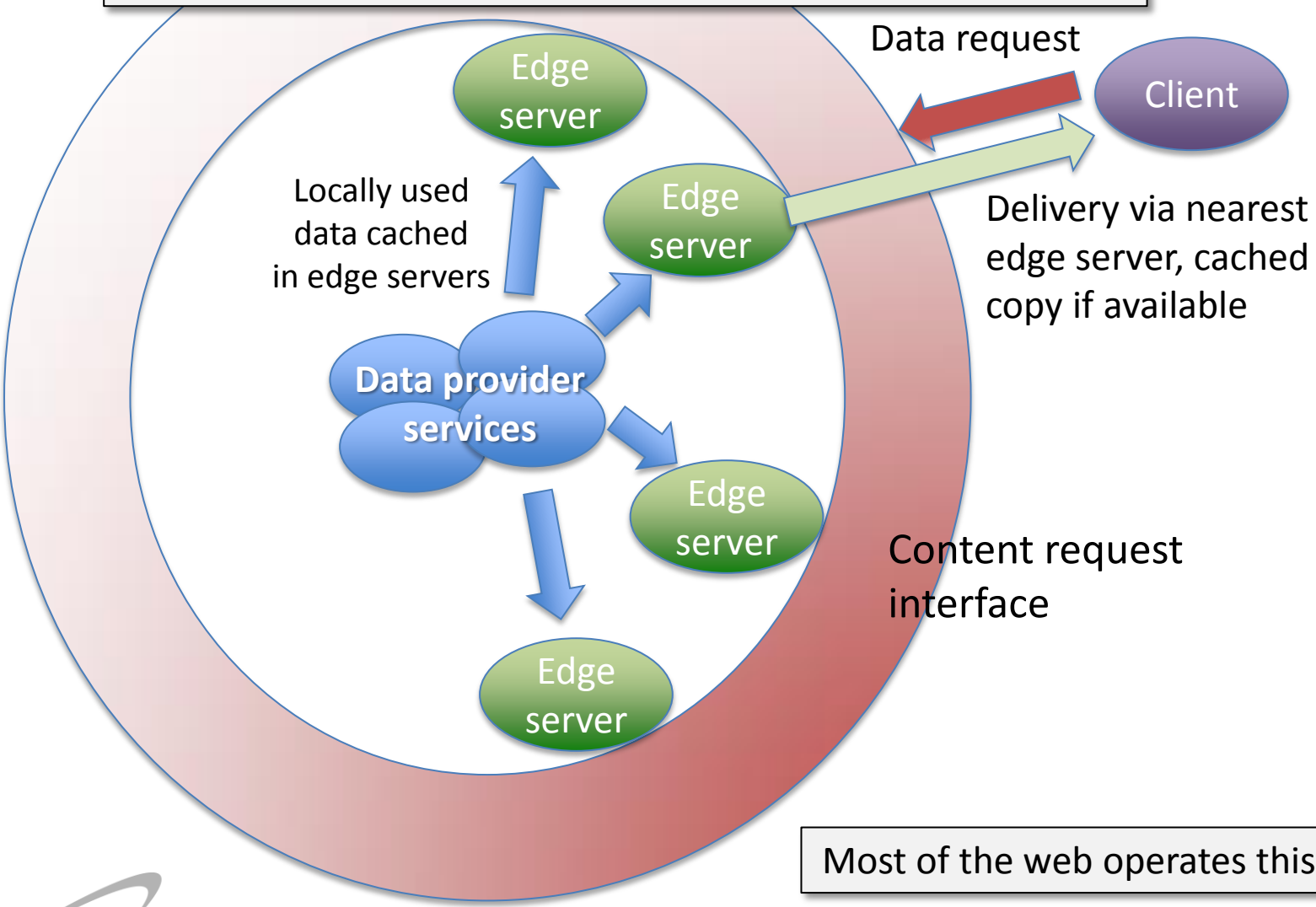
BROOKHAVEN NATIONAL LABORATORY

What's an event service?

- On the model of… ask for exactly what you need, have it delivered to you by a service that knows how to get it to you efficiently
- Why ask for files when what you really want are (particular) events
- (Perhaps the events you're asking for don't even exist in the required form, and they'll be transparently created for you – that's another discussion… virtual data… but on the same path)

Service Based Content Delivery

Content delivery network: deliver data quickly and efficiently by placing data of interest close to its clients



Most of the web operates this way

Service Based Content Delivery

Many services operate broadly on the CDN model, event service is one more

Service	Implementation	In production
Frontier conditions DB	Central DB + web service cached by http proxies	~10 years (CDF, CMS, ATLAS, …)
CERNVM File System (CVMFS)	Central file repo + web service cached by http proxies and accessible as local file system	Few years (LHC expts, OSG, …)
Xrootd based federated distributed storage	Global namespace with local xrootd acting as edge service for the federated store	Xrootd 10+ years Federations ~now (CMS AAA, ATLAS FAX, …)
Event service	Requested events delivered to a client agnostic as to event origin (cache, remote file, on-demand generation)	First ATLAS implementation coming in next 6-12 months
Virtual data service	The ultimate event service backed by data provenance, regeneration infrastructure	Few years?

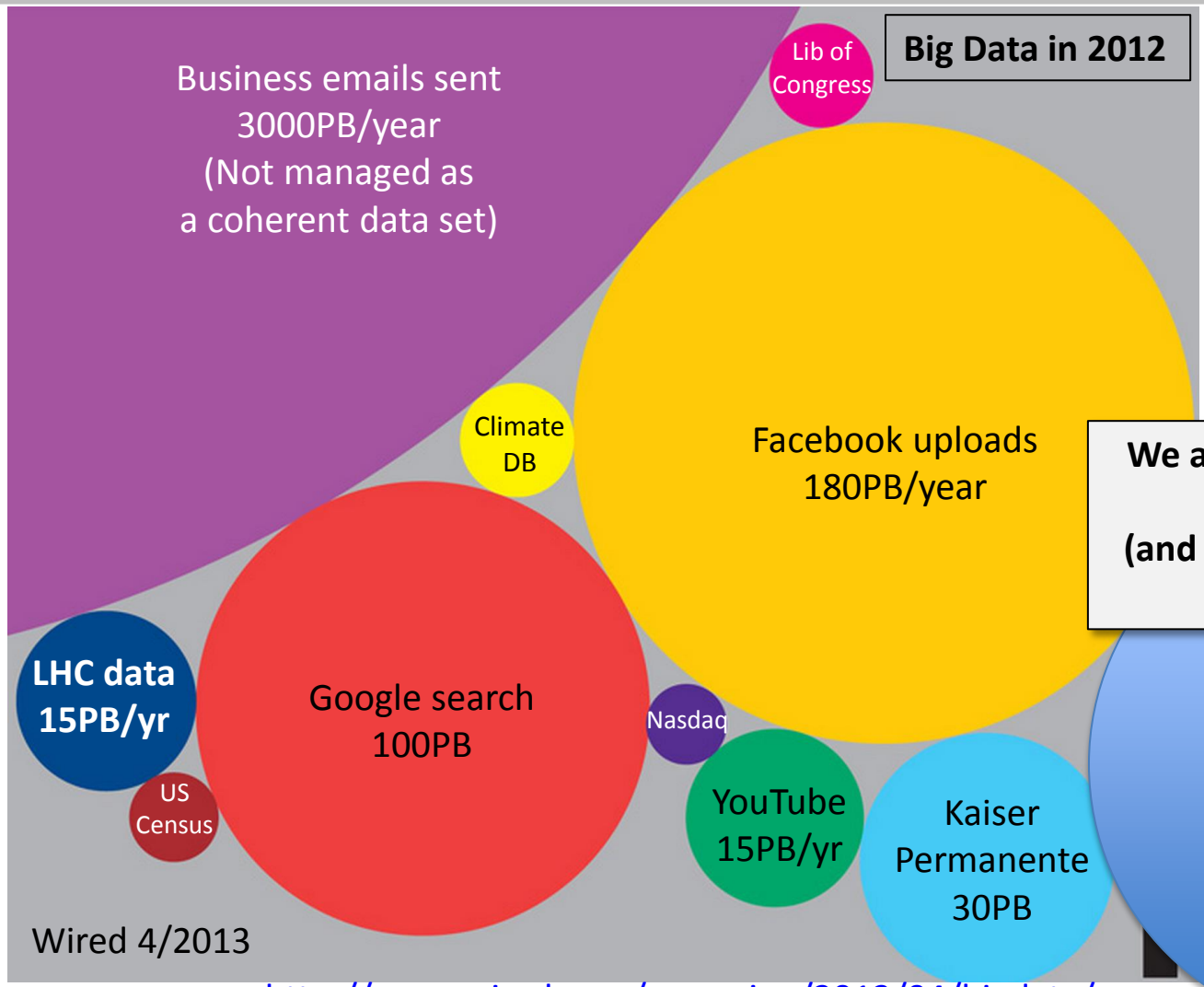
Why an event service?
Do for data what PanDA did for processing

- PanDA shows that intelligent automated dispatch of jobs to intelligent clients works very well
  - Makes processing self-correcting – you use the resources that work
- The biggest operational load is in data management
  - We are hit hard by storage resources that don't work, or are full
  - DDM is complex
- PD2P showed that dynamically distributing only the data that is needed can improve resource usage efficiency – use the network dynamically rather than using storage statically
  - **In general it's much cheaper to transport data than to store it**
- The event service is a further step
  - In making the client agnostic to where the data is – transparently use the data sources that work – to reduce the operational burden and user impact of storage problems
  - In making full use of the network to deliver just the data needed from an optimized (and minimal) set of sources
- We have a distributed processing infrastructure very well suited to give it a try
  - PanDA brokers and delivers jobs in an (increasingly) intelligent way
  - Let's use the same infrastructure to deliver events
  - Make resilient data access an integral, automated part of the processing; no DDM in the production workflow and no precondition of data pre-placement
- We have a software infrastructure ready to support it

Why an event service? Opportunistic resources to expand computing throughput

- Opportunistic use of HLT after LS1
- Opportunistically soaking up cycles on supercomputers
- Free research clouds, commercial clouds
  - More/cheaper resources available if we can be highly opportunistic
    - e.g. Amazon spot market – cheap, even free under 1 hour
- 'ATLAS@Home' via BOINC on tens of thousands of computers in China?
- Common characteristic to using opportunistic resources: we have to be agile
  - Quick start (rapid setup of software, data and workloads) when they appear
  - Quick exit when they're about to disappear
  - Robust against their disappearing with no notice : minimize losses
  - Use them until they disappear – soak up unused cycles
  - Fill them with fine grained workloads
    - Send a steady stream of events, and return outputs in a steady stream
      - No heavy data prestaging, and a hasty exit loses very little
- **Event service is a means of enabling agile, efficient use of opportunistic resources**

Why an event service?
Efficiency of storage usage is essential for us



In 2012: 2800 exabytes created or replicated
1 Exabyte = 1000 PB

Reputed capacity of NSA's new Utah center: 5000 ExaB (50-100 MW)

We are big… not NSA-big, but big
(and more cost efficient… and poorer)

http://www.wired.com/magazine/2013/04/bigdata/

Other event service advantages

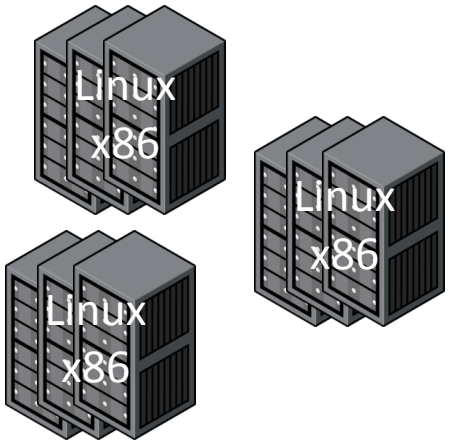
- Avoid idle cores in athenaMP processing – if parallel threads process events in large chunks (files), cores can sit idle while the slowest one completes
- DDM simplification – no DDM involvement on input or output
- Output merging flexible and simple: file size is tunable, aggregation of outputs to merge site proceeds concurrently with processing, JEDI knows when to trigger merge
- A crash doesn't lose the job, only a few events which will be re-dispatched

Required for an event service

- Managing workflows at the event level the way PanDA has been managing them at the file level, with the necessary event-level bookkeeping
  - Check – JEDI now provides this
- Excellent networking; check
- Workloads with high CPU/IO; check – simu
- Network-efficient event data access; check – the event I/O optimizations of recent years makes remote event streaming possible
  - *Preferably buffered by asynchronous caching*
  - Just as JEDI is an enabler for the event server scheme on the grid side, athenaMP and associated I/O developments and optimizations are enablers on the core sw side
    - Queueing and streaming events to be consumed by workers
    - I/O optimizations making event reading over WAN practical
    - Asynchronous pre-fetch to remove network latencies from processing workflow

Distributed Computing Evolution

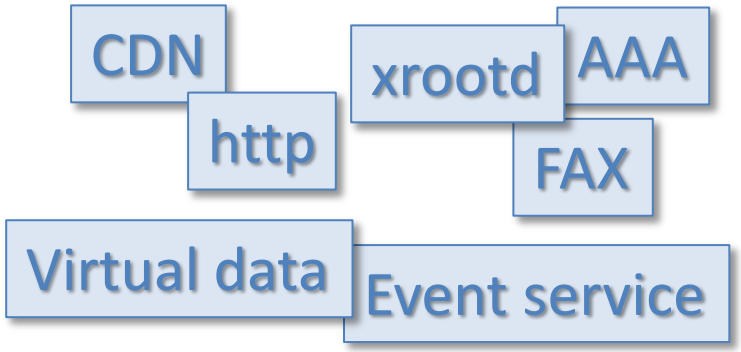
1980s: Plethora of architectures & OSes



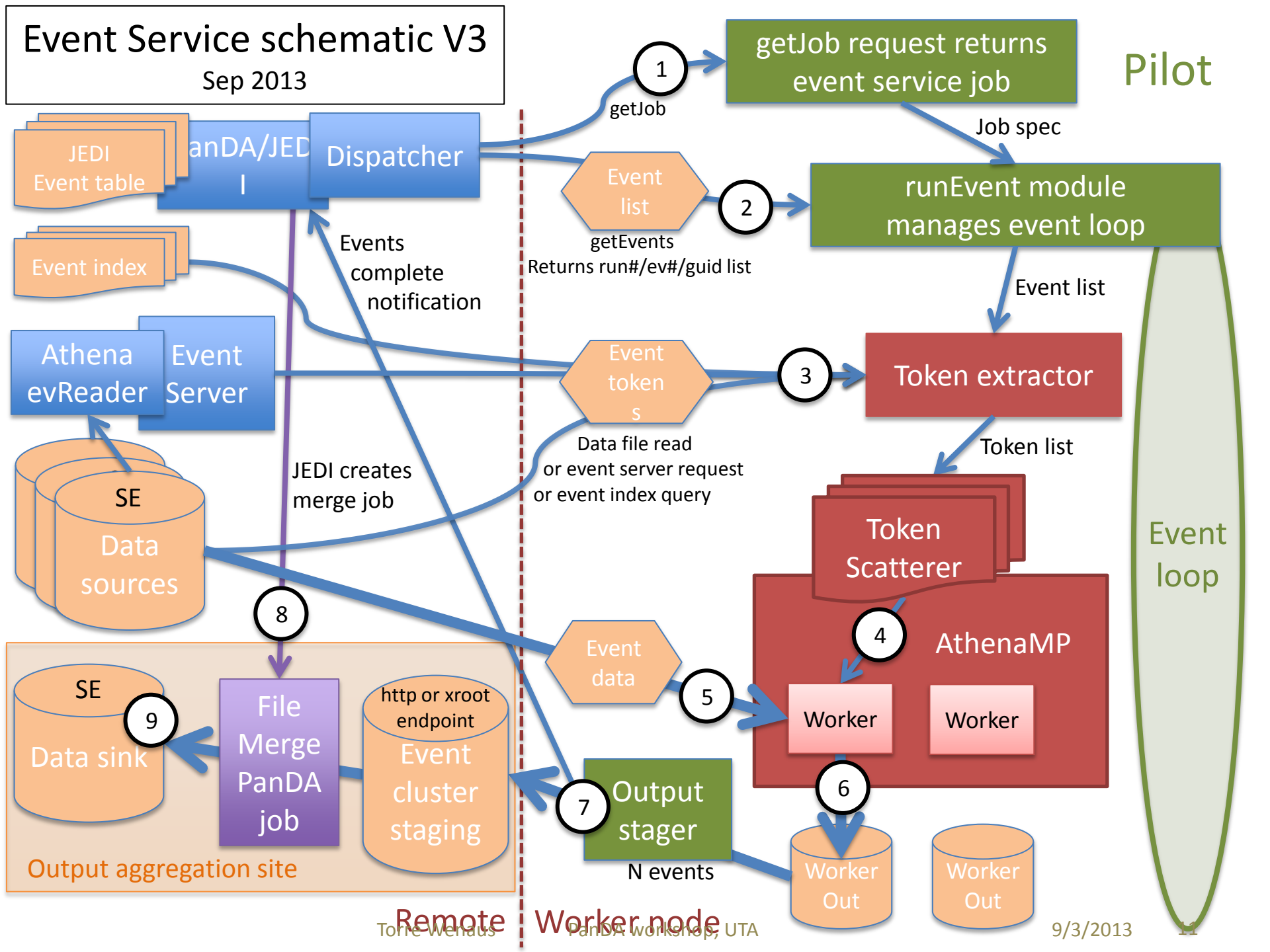
1990s: **Uniform OS/architecture**
Linux/x86 standard for commodity cluster computing

2000s: **Uniform fabric and access**
Globally federated resources enabled by network and grid

2010s: **Uniform environment**
VMs and clouds put the user in control of the environment – take it with you anywhere and everywhere



2010s: **Uniform data access**
Working towards transparent distributed data access enabled by the network


Event Service schematic V3
Sep 2013
Pilot
getJob request returns event service job
1
getJob
Job spec
JEDI Event table
anDA/JEDI
Dispatcher
Event list
2
getEvents
Returns run#/ev#/guid list
runEvent module manages event loop
Event index
Events complete notification
Event list
Athena evReader
Event Server
Event tokens
3
Token extractor
Data file read
or event server request
or event index query
Token list
SE
Data sources
JEDI creates merge job
Token Scatterer
Event loop
8
4
AthenaMP
Event data
5
Worker
Worker
SE
9
Data sink
File Merge PanDA job
http or xroot endpoint
Event cluster staging
7
Output stager
6
N events
Worker Out
Worker Out
Output aggregation site
Remote
Worker node
Torre Wenaus
PanDA workshop, UTA
9/3/2013
11

Workflow

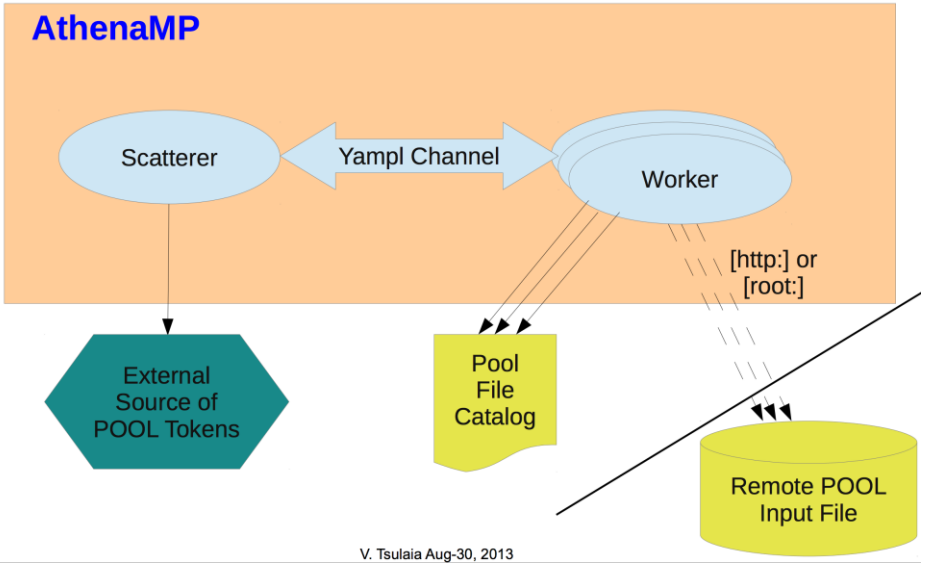
- Pilot makes a getJob request to PanDA
- PanDA determines event service type job is appropriate, selects and dispatches one
- Pilot's runEvent module invoked to manage event processing
  - Determines input files, makes POOL catalog file
  - Triggers athenaMP launch, configuration
  - Enters event processing loop
    - Requests event list from PanDA dispatcher
    - Receives list, GUID, retry number; PanDA marks events as in process
    - Event list passed to token extractor
      - Obtains tokens via direct data file read, event server or event index
    - Token list sent to athenaMP TokenScatterer for distribution to workers
    - Workers read event data from source and process events
    - Each worker writes its own output file
      - Single-event files initially; appropriate for simu with long event processing time

Workflow (2)

- Outputs managed in a near real time, granular way, just as inputs are
  - Output aggregation and monitoring concurrent with task processing
  - Minimal losses in case of sudden eviction from or disappearance of resource
- Output stager monitors output directory, detects completed output files
  - Transfers them to aggregation point using lightweight service (http, xroot) or just a copy if local
  - Informs PanDA of event completion; PanDA updates event table
  - Cleans up output files
- PanDA monitors event completion in the event table
- When event processing for a job is complete, PanDA merge job is generated to merge output files, store merged outputs on SE and register in DDM
- If processing of an event cluster by a consumer times out, PanDA invalidates that consumer by incrementing the retry number for those events and reassigning them to another consumer
  - If original consumer does finally return results, they will be ignored; their retry number is no longer the latest

Event reading and processing status

- Event reader and event server web service prototypes exist
- First reader implementation was with bytestream events, now extended to POOL files
  - Targeting G4 simulation as first use case
    - Simulation is well suited to the opportunistic resources that are a principal target
- AthenaMP workers receive POOL event tokens from the TokenScatterer, ROOT retrieves event data
  - First version exists, relies on POOL file catalog for file/protocol info
- Successfully tested with sample AtlasG4_tf job
  - Currently needs 1-event input file for configuration
  - Remote http read works, outputs are identical to local read



V. Tsulaia, R. Vitillo, P. Van Gemmeren

Workplan

- Initial implementation target is Geant4 simulation
  - The prime candidate for the opportunistic resources that have the most to gain from the event service
- Settle the dispatcher communication protocols for specifying a job as event type and sending an event cluster in response to a request from runEvent (Tadashi, Paul)
- Set up POOL catalog file creation for event type jobs (Paul)
- Implement token extractor and translation of event list received from pilot into token list delivered to Scatterer. Tools are mostly there to create the token extractor, in the form of tag collection utilities to extract tokens out of files. (LBNL, Peter)
- Set up the invocation and execution of the athenaMP payload initialization step (LBNL, Paul)
- Set up the athenaMP workflow in the event loop from receipt of token list, to WN event processing using the pilot-generated POOL file catalog, to output generation in dedicated output directory (LBNL, Peter)
- Check WAN performance on POOL event read. Can we use TTreeCache asynchronous pre-fetch (? Ilija?)
- Look into how to handle IOV metadata. (Peter)

Workplan (2)

- Implement the communication from pilot to PanDA/JEDI to inform PanDA of completed events: status of completion and transmission to aggregation point, retry number (Tadashi, Paul)
- Implement the output monitor to detect completed output files, send them to aggregation point, inform PanDA/JEDI, and clean them up (?)
- Implement prototype output aggregation point (local and/or remote/webdav/xrootd implementation), and PanDA job that can build cluster files from aggregation area into full output file for registration with DDM (?)
- Implement PanDA/JEDI service that detects event job completion (from accumulation of event completion metadata) and triggers PanDA merge job, and times out/reassigns events held by non-responding consumers (Tadashi)
- Establish testing setup. NB runEvent can be used outside the context of the full pilot, for testing purposes
- Establish Hammercloud based testing setup for full infrastructure. (all)

Conclusion

- Prodsys2 is designed to enable efficient & flexible resource usage across a diverse range of resources – multi/many-core, HPC, clouds, opportunistic, …
- Together with concurrency-directed core software work, it enables event servers for efficient use of these resources
- Leverages existing work in athenaMP, event I/O optimization, and FAX
- Effective WAN direct access to data is an enabler and a prerequisite
- Prototyping and development under way, subject to higher priorities but progressing quite rapidly
- Most of the who and how questions for an initial implementation answered, but not all
  - In particular output aggregation and merge management needs attention
- Help is welcome!