CIVAIS: Campus Infrastructures Visual Analytics and Informatics Services

Rob Gardner, David Champion, Marco Mambelli

University of Chicago

Vision

We describe a visual analytics and informatics service for science throughput, targeted for the Campus Infrastructures Community (CIC) of the OSG. The service provides powerful viewing services and repositories of information principally for campus users, faculty and college deans, campus CI resource directors, provisioners and systems administrators. The services are meant to augment those systems already in place but are tailored to the distributed high throughput computing (dHTC) components of a collective campus computing framework and, if appropriate, off-campus resources from the shared OSG infrastructure, commercial and private clouds, and HPC resources.

We also envision multi-campus, hierarchical views in order to provide aggregated sets of rolled-up analytics for resource informatics by scientific discipline, user/group/VO/campus unit, task/project/campaign, resource type, funding sources, services used, access modes employed, and by institutional contribution and consumption with configurable filtering mechanisms to allow for desired privacy levels for reporting from campuses. We emphasize both quantitative and visual aspects of analytics with the ability to derive clear, evocative metrics and to associate (post-hoc) published results to resources consumed and services delivered.

The system should be able to compose vivid, personalized descriptions profiling the end-user experience in both real-time and post-execution, including, for example, information and knowledge about job/task execution times, efficiencies and latencies, time-to-completion estimates, success and failures rates by category, and associated costs in terms of consumed CPU-time, storage, cache, IO bandwidth (at the cluster fabric, the campus, and across wide area research networks) and, where appropriate, HPC service units or dollar costs where commercial cloud resources are used. This naturally leads to knowledge about resource delivery that faculty, executive steering committees and college deans observe and evaluate, CI resource directors and administrators monitor, and that the OSG Executive Team assesses in terms of the core OSG missions of facilitating high throughput science for the community, enabling resource sharing, and promoting resource brokering and exchange among campuses for the overall benefit of science in the U.S. cyberinfrastructure. Such information should be a basis for reporting at all levels including agency program managers, campus newsletters, and importantly the general public.

For developers and systems integrators, the services and repositories will adhere to open principles of collaboration, open standards and non-proprietary implementations and dependencies wherever practical, standard formats and protocols to encourage extended use of the services, re-use of components, and exportability of the infrastructure and informatics artifacts. A developers and users kit with easy-to-use programming interfaces to the core services would allow for "third party" consumers of the data, allowing for higher level analytics beyond those built into the baseline system. A standard API would allow controlled access to all the data to encourage new creative uses.

Survey of Related Software, Services and Selected Ideas

A number of services already exist which have desirable features and components that might be either reused or adapted to meet the broader objectives outlined above, and there are several sources of good ideas for data analytics, visualization and informatics. We review in outline their main purposes, user base, architectural features, features and ideas.

Gratia, the OSG accounting service

- Project delivered to OSG, no longer in development
- Accounting data
- Used mostly by VO, OSG and WLCG project management teams
- Gratia Service (Collector): information repository database and information display
- Information providers (Probes) report information to the collector in a standard way
- OSG deploys probes in all Sites, provides a central Collector and allows optional site-level Collectors
- Documentation: http://www.opensciencegrid.org/bin/view/Accounting/WebHome https://twiki.grid.iu.edu/bin/view/Accounting/GratiaDataCollectionNotes
- Download (Service): http://twiki.grid.iu.edu/bin/view/Documentation/Release3/InstallGratiaService
- Download (probes): http://twiki.grid.iu.edu/bin/view/Accounting/ProbeInstallation
- Example: OSG Web Display: http://display.grid.iu.edu/
- Example: Gratia Web: http://gratiaweb.grid.iu.edu/gratia/
- Contact: OSG, Tanya Levshina, grid-accounting@fnal.gov

Project developed by OSG. It accumulated and stored in an elaborated relational database a lot of records about job execution and data transfers in OSG. There is a wide array of probes collecting information from OSG Compute and Storage Elements. Any solution would have to be able to ingest data from a Gratia server and possibly also from the probes directly. There have been episodes where the server was not scaling up and the data collected had to be reduced (e.g. records about individual transfers from CMS sites).

Experiment Dashboard (aka ARDA Dashboard)

- Continuous development by WLCG for the last few years, on-going
- Aims to provide a single entry point to the monitoring data collected from the distributed computing systems of the LHC virtual organization.
- Custom Web framework based on Python
- Message bus technology (ActiveMQ)
- Variety of clients and tertiary databases and services (e.g. popularity service)
- Useful operational tool
- Documentation: http://dashboard.cern.ch/ http://dashb-build.cern.ch/build/stable/doc/guides/common/html/dev/index.html
- Example view: http://dashb-atlas-job.cern.ch/dashboard/templates/web-job2/
- Contact: dashboard-support@cern.ch

It is a nice Python Web Framework (with ORM, MQ and MVC), has nice views and drill-down reports, it is already known and familiar to the HEP community and developers may be known and involved in other CERN projects. On the other side it has a lot of custom elements not well documented and it may be safer to rely on mainstream frameworks like Django and TurboGears that are better documented and supported.

SAM (Service and Availability Monitor)

- Continuous development for last few years, on-going (probably)

- Monitors the status of services and evaluates and reports availability using various displays
- Uses Open Source technologies like Nagios, ActiveMQ (message queue), Django, JasperReports
- Documentation: https://twiki.cern.ch/twiki/bin/view/LCG/SamCern https://tomtools.cern.ch/confluence/display/SAMDOC/SAM+Overview http://openlab.web.cern.ch/sites/openlab.web.cern.ch/files/technical_documents/Robert-Veznaver_report.pdf
- Example view (MyWLCG): http://grid-monitoring.cern.ch/mywlcg/
- Contacts:

Shares some technologies with the Experiment Dashboard (comes from the same community), seems to use more standard technologies. It monitors and stores availability information for resources used by LHC experiments.

APEL (Accounting Processor for Event Logs)

- Continuous development since 2004, used in WLCG and EMI [17]
- Gathers CPU accounting information by parsing batch system and BLAH accounting log files and inserting the data into a local database. APEL then publishes the data into a centralized repository)
- APEL Server is written in Python and has the following components: SSM (Secure Stomp Messenger), Record loader, Database (MySQL), Summarizer, Record publisher, Authentication, Monitoring
- APEL Client: Parsers and Publisher (MySQL DB and ActiveMQ to the server)
- Standard records (OGF): CAR (CPU accounting) and StAR (Storage accounting)
- Collects from multiple sources: APEL, DGAS, ARC/SGAS, OSG/Gratia, Unicore accounting, GridSafe
- Documentation: https://wiki.egi.eu/wiki/Accounting_Portal https://accounting.egi.eu/links/acct_ibergrid06_final14.pdf https://twiki.cern.ch/twiki/pub/EMI/APELClient/APEL_Client_User_Guide.pdf
- Interesting report on messaging: https://wiki.egi.eu/w/images/a/ab/APEL-Messaging-v2.2.pdf
- Example view: https://accounting.egi.eu/egi.php
- Contact: egee-admin@cesga.es
- Useful operational tool

APEL records very specific information but the messaging component (SSM) is interesting and could be useful for CIVAIS

XDMoD (XSEDE Metrics on Demand)

- It is probably a couple of years old and under active development
- Provides detailed information on resource utilization and performance across all resource providers
- XDMoD Data Warehouse, which ingests data daily from the XSEDE Central Database, the XDMoD REST API that provides hooks to external applications, and the XDMoD Portal
- The display is in HTML5 and JavaScript and uses the ExtJS library
- Demo: http://www.youtube.com/watch?v=R8_O1-t778k
- Example view *https://xdmod.ccr.buffalo.edu/#main_tab_panel2:tg_summary*

Very little documentation is available but the authors probably could provide more (there are several collaborations between OSG and XSEDE). The role-based views are an interesting interface. The display uses standard technologies (HTML5, JavaScript), is responsive, offers many customization capabilities and the interface is not too cluttered.

Brazos CMS Data Analysis Site Monitoring Utility

- Site Monitoring developed by Joel Walker for CMS Tier3 Sites
- Open Source. Under current active development
- Flexible dashboard showing jobs status and data transfer using local and CMS (centralized) information
- No privileged account, uses HTML, SSI, cgi-bin with Perl and Shell scripts
- Download: *http://www.joelwalker.net/code/brazos/brazos.tgz*
- Example view: *http://brazos.tamu.edu/~ext-jww004/mon/*
- Contact: Joel Walker *JWW004@SHSU.EDU*

It is a good example of a non-invasive tool (no privileges required) and easy to install.

CycleServer (monitoring) by Cycle Computing Inc

- CycleServer provides valuable insights through visualization, analysis, and reporting, and saves time by providing an easy interface to managing and using your HPC clusters (HTCondor. SGE, Torque and Hadoop coming soon)
- Commercial product under active development. Free under 2000 nodes
- Java (1.6) application querying and administering the job manager
- Download: http://www.cyclecomputing.com/cycleserver/overview
- Documentation: http://www.cyclecomputing.com/wiki/index.php?title=CycleServer
- Example view: http://uc3-mgt.uchicago.edu:8000/
- Contact: http://www.cyclecomputing.com/contact

The zoomable graph showing the number of jobs that ran on the system is the main graph currently used to show the activity on UC3.

HTCondorView

- HTCondorView Client is a contrib module distributed with HTCondor
- It consists of few scripts querying the HTCondorView Server to generate statistics and a Java applet for customizable views
- HTCondorView Server is a HTCondor collector collecting ClassAds and storing historical data
- Open Source. Client development stopped in 1998 (the Server is a HTCondor daemon)
- Documentation: http://research.cs.wisc.edu/htcondor/manual/v7.9/8_4HTCondorView_Client.html
- Original project: http://www.st.ewi.tudelft.nl/condor/condorview60.html

The graphic interface is obsolete but provides a good example on how to collect information about the execution of HTCondor jobs.

Highly Interactive Gateways for Understanding NSF Investments

- Developed by Purdue and Virginia Tech, Krishna Madhavan, lead; status?
- Geographical visualization of awards, by congressional district
- Relationships between investigators.

City of Chicago Urban Metrics Project

- Open data archive of various city statistics. Several datasets are made available.
- Some information is already processed and presented on the Web site
- A RESTful interface and clear access procedures allow access to the data and to build on top of it.
- http://opencityapps.org/
- Open311 ticket system: http://www.cityofchicago.org/content/dam/city/depts/311/general/NewOpen311InformationChart.pdf
- Socrata (http://www.socrata.com/) is a private company offering an Open Data platform (GovStat) and procedures [11] for government agencies including the City of Chicago
- Allows access to data via SODA (Socata Open Data API): http://opendata.socrata.com/api/docs/

It is a good example of Open Data were the City of Chicago improved its image (projecting transparency the willingness of being close to the people) and benefits from the creativity of users interested in using or providing information [14][15].

ESPN Analytics

Given the explosion in saber metrics and all manner of sports metrics, which are useful in betting and fantasy league contests, the teams a ESPN The Magazine produces highly visual analytics and measures based on their sports databases. ESPN recently visited CERN to understand what LHC does with big data. Can we open a collaboration with them?

Misc

- http://www.informationisbeautiful.net/
- Storing spreadsheet-type data in Google Docs, e.g. could we make an OSG CIC Data Room, a repository of 'fun' (and important) spreadsheets, e.g. https://docs.google.com/spreadsheet/ccc?key=0AvZNxPpzAwyIdEZoR0dRY2tZbm1xSUdTWlJTUmVnU3c#gid=1
- SandDance web visualization by Microsoft Research http://www.computer.org/portal/web/computingnow/high%20performance/content?g=53319&type=article&urlTitle=video%3A-sanddance

Google Data API allows interacting with Google documents, both retrieving and storing data and this is definitely a useful feature for CIVAIS.

FiveThirtyEight.com

- What can be learned from Nate Silver's blogs on predictive elections and sports, http://fivethirtyeight.blogs.nytimes.com/

TED Lectures on Data Visualization and Design

- Excellent lectures on data visualization
- On design aspects relating to user interfaces

Sencha Ext JS

- JavaScript Framework for Rich Desktop Apps
- Allows highly interactive displays with drill-down graphs, etc.
- All HTML5 and JavaScript, vectorial graphic (no Flash, faster and more scalable than bitmap generation)
- Solves compatibility problems with different devices
- Used by XDMoD
- Visual App builder to help fast prototyping (http://www.sencha.com/products/architect/)
- Website: http://www.sencha.com/products/extjs/

Interesting platform, used by XDMoD. Sencha Architect, a visual development product by the same company, would simplify rapid prototyping. It should at least be evaluated for CIVAIS.

Message Queue

- A Message Queue allows a more robust and flexible routing of messages
- Online there are descriptions and comparisons of message queues
- From a comparison of RabbitMQ, 0MQ, ActiveMQ on StackOverflow [5] and a couple of other online tests and reviews [6][7]:
- RabbitMQ is one of the leading implementation of the AMQP protocol (along with Apache Qpid). Therefore, it implements a broker architecture, meaning that messages are queued on a central node before being sent to clients. This approach makes RabbitMQ very easy to use and deploy, because advanced scenarios like routing, load balancing or persistent message queuing are supported in just a few lines of code. However, it also makes it less scalable and "slower" because the central node adds latency and message envelopes are quite big. Server written in Erlang.
- ZeroMQ is a very lightweight messaging system specially designed for high throughput/low latency scenarios like the one you can find in the financial world. Zmq supports many advanced messaging scenarios but contrary to RabbitMQ, you'll have to implement most of them yourself by combining various pieces of the framework (e.g. : sockets and devices). Zmq is very flexible but you'll have to study the 80 pages or so of the guide (recommended reading for anybody writing distributed system [8]) before being able to do anything more complicated that sending messages between 2 peers. Much faster than the other but has no central broker (central point of control, persistency and reliability)
- ActiveMQ is in the middle ground. Like Zmq, it can be deployed with both broker and P2P topologies. Like RabbitMQ, it's easier to implement advanced scenarios but usually

at the cost of raw performance. Considered the Swiss army knife of messaging by a reviewer. Used by ARDA Dashboard. Server running in JVM (Tomcat)

- All 3 RabbitMQ, ZeroMQ and ActiveMQ: support AMQP (Advanced Message Queuing Protocol), have client APIs for the most common languages (C++, Java, .Net, Python, PHP, Ruby, …); have strong documentation, are actively supported
- Kestrel is a very simple message queue that runs on the JVM. It supports multiple protocols including memcache (no AMQP). Optimized for big flows, allows out of order. Used by Twitter.
- ZeroMQ seems the best choice performance-wise but limited data loss must be acceptable or avoided with custom solutions. RabbitMQ seems to outperform ActiveMQ and would not require JVM

A message queue would allow decoupling data acquisition from data consumption within CIVAIS data warehouse. Solutions based on a central broker, like RabbitMQ and ActiveMQ, would provide also added reliability.

Database backend

- Gratia and most of the Web frameworks (Django, TurboGears, ARDA Dashboard) use, directly or via an ORB, SQL databases, like MySQL or Postgres.
- noSQL DB like MongoDB, used by some of the City o Chicago repositories and other Big Data projects, would allow better scalability
- An abstraction level could allow to postpone the choice but could limit performance and possible operations

NoSQL (or newSQL) storages allow scaling by simply adding hardware (more machines instead of new high performance machine) and may improve reliability and complex elaborations. Beside the values and provenance information all the data collected should include from the beginning information about the life cycle (expiration, expected replacement), the ownership and who is authorized to access it.

RESTful API

- Allows a standard way to access data using HTTP methods (PUT, GET, POST, DELETE)
- Some framework help deploying the interface

The Representational State Transfer (REST) Web API has emerged as a predominant Web API design model and is common practice to provide Open Data and allow interaction. Other Web API and Web Services (like SOAP) provide a standard interface as well, allowing sometime more complex interactions, but are less immediate and less successful. Clear access and retention policies have to be in place for all data since collection in order not to cause conflicts at time of publishing.

Potential First Level Requirements

We gather here ideas for potential requirements based on queries that most likely will arise based on consumer role, beginning first with campus infrastructure users. The below are based on literature surveys and practical experience. Later we plan to conduct formal questionnaire surveys and interviews.

Users, researchers using OSG or the Campus Grid to run jobs or transfer data, are interested in getting work done. Ideally would like not to worry about jobs. CIVAIS information becomes important only if their resources are limited and they want to know how much of their quota they used and how quickly they will get work done. It may come handy also to troubleshoot problems (failed jobs or data transfers)

Administrators, PIs, directors are interested in aggregated information showing the use of a resource or all the resources used by a group.

Technicians like network managers are interested in information that can help troubleshooting, dimensioning of structures or improve operation by understanding usage patterns.

What do users most want to know?

- What resources are available to me and what are their characteristics and policies
- What is my priority and/or allocation
- What is the average wait time for a job to begin
- How many of my jobs are running
- How many of my jobs are waiting
- How many of my jobs have completed successfully/failed over the past (hour, 12 hours, day, week, month, year)
- How much CPU-time have I consumed for a given task or project
- How much CPU-time do I need
- Will I be able to get enough, in time to meet a publication/presentation deadline
- What is the estimated completion time (by job, by job set/task)
- How does my performance and usage compare to the average user

What do faculty sponsors and research directors most want to know?

- Are my students and postdocs successfully accessing and consuming resources
- Do they have enough for our current set of projects

What do campus research computing center directors most want to know?

- How satisfied are faculty investors
- How efficiently are the acquired resources being used
- What is the pattern of use by expert, intermediate and novice users
- What are the weak points in the infrastructure
- Which applications and services are most popular with users from an ease-of-use perspective
- Which services are most stressed, and what are the causes of those stresses
- How can I make the infrastructure more reliable

- How do operational costs (power, space, cooling) compare against CPU-time and storage delivered
- What are the bottlenecks in the system (networking) and which applications / users / projects are driving them
- Are contributions my center has made to external groups, even if having used up cycles that otherwise would have gone to waste, accounted for, recognized, and if so, where?
- Is my campus community taking advantage of over provisioned resources on other campuses, if so, how much.

What do computing center executive/steering committees most want to know?

- How are resources being used
- Are they serving investing stakeholders fairly, as well as the broader university community
- Is capacity meeting demand
- Which technologies (processing, storage, network, visualization) are most likely to yield the most benefit to the most users
- How do we judge the effectiveness of resource usage for advancing the scientific goals of our stake-holding partners

What do college deans most want to know?

- Are faculty and academic units being served fairly
- How much are the resources being used for research versus education versus training
- How are university-wide investments being used
- How have academic units driven acquisitions of resources on campus
- How well the shared campus cluster/condominium model meet faculty needs and is it cost effective

What do campus CIO/CTO officials most want to know?

- What are the cost/benefits associated with investments made to date
- Which capabilities should the next major acquisition or campus investments focus on
- How can current usage and demand inform decisions about future campus-wide technology investments
- What benefit has my campus received as a result of plugging into the OSG shared high throughput service
- Have campus resources been more efficiently used as a result of enabling a campus-wide high throughput infrastructure
- What are the costs / trade-offs for university-owned resources versus commercially provided resources

What do campus network engineers and managers most want to know?

- How is the network being used in terms of users and data flows
- Are current requirements being satisfactorily met
- What are the future sources of demand

What do research and education network organizations most want to know?

- How is the network being used in terms of users and data flows
- Are current requirements being satisfactorily met
- What are the future sources of demand
- What impacts do specific user communities have on network technology choices in terms of reliability and advanced capabilities
- What are the overall trends by discipline

What does the OSG Executive Team most want to know?

- How effective is the campus program in engaging new users and research communities on campuses
- Which disciplines, outside of high energy physics, have received benefit from OSG
- What are the impacts of OSG services and technologies on accelerating the scientific mission of our stake-holding organizations as well as the national

What do agency program managers most want to know?

- What are the societal impacts on funding investments made in software, data and computing infrastructure.
- How successful have funded programs been in delivering promised resources and science results

What do OSG stakeholders most want to know?

- How effective have campus-based services been in expanding the pool of accessible resources to my virtual organization.
- Has any effort my organization invested in the campus program yielded additional resources.
- How beneficial have these additional resources been, compared to the effort expended (cost/benefit analysis)

Summary of Prototype Features

- Collect job metrics from the submit host
- Collect metrics from OSG's Gratia
- Collect data from an online feed, e.g. Google spreadsheet
- Display the data
- At least two distinct roles
- At least three distinct interactive views showcasing different plots and scaling and drill-down capabilities
- Deployed prototype and separate source repository
- Internal documentation for the developers

Summary of Alpha Features

- Collect job metrics from the submit host
- Collect metrics from OSG's Gratia
- Collect data from an online feed, e.g. Google spreadsheet
- Display the data
- At least one authentication system (username/password, or x509 certificates, or CILogon)
- At least two distinct roles (both in the authentication and the views)
- At least four distinct interactive views showcasing different plots and scaling and drill-down capabilities
- At least rudimentary packaging for at least one platform
- Some documentation on how to use the viewer (it should be mostly self documenting)
- Documentation on how to install and configure the server

Additional Features to include before release

These are in addition to the features for the alpha release and include some sets that have not been defined yet (e.g. set of important roles)

- Customizable views (ability to add custom interactive views)
- Support all three authentications: username/password, x509 certificate and CILogon or LDAP connection
- Support all the important roles (4 or 5 to be defined)
- Support all the important views (6-8 to be defined)
- Possibility to generate bitmaps from the views (export pictures)
- Possibility to retrieve formatted data from the views (the numbers behind the plots)
- REST API to retrieve data (e.g. 3rd party consumer)
- Simple deployment for both the probe (data collection components) and the server in a Campus Infrastructure
- Ability to organize the servers in a hierarchical structure
- Some documentation on how to use the viewer (it should be mostly self documenting)
- Complete documentation on how to install and configure the server and the probes

- Documentation on how to retrieve data and an example of a 3rd party consumer (e.g. e-mail report)

Prototype Architecture

The design should start from the user experience: the display presenting the data, the API to access the data.

There are currently some possible alternatives for the architecture that will be defined during the development of the service mock-up and the prototype.

Multiple data sources, including Gratia, provide data that is processed and stored in a Data Warehouse. Multiple consumers access data via a RESTful API. Other CIVAIS Data Warehouses can act as consumers allowing the connection in a hierarchical structure. An important consumer developed within the project is a display allowing the visualization and the exploration of data on multiple devices (e.g. all devices supporting a Web browser).

Here some additional ideas about CIVAIS components and architecture:

- The Web Display allows to modify interactively the views (apply cuts, change scales, …). Possibly it should use HTML5 and vector graphic.
- A message bus should be used to decouple data collection and elaboration and to be flexible in the routing of the messages. In the architecture diagram in Figure 1 the arrows with double lines represent connections brokered by a message bus.
- A data warehouse will include a database to collect the information and provide persistent storage. Alternative technologies to evaluate include traditional SQL databases, noSQL and newSQL databases, Round-Robin databases.
- The data warehouse allows connections to other data warehouses to build a hierarchical structure with multiple CIVAIS instances
- The display should be decoupled by the data (MVC pattern).
- Data coming form different sources is filtered and aggregated before being stored

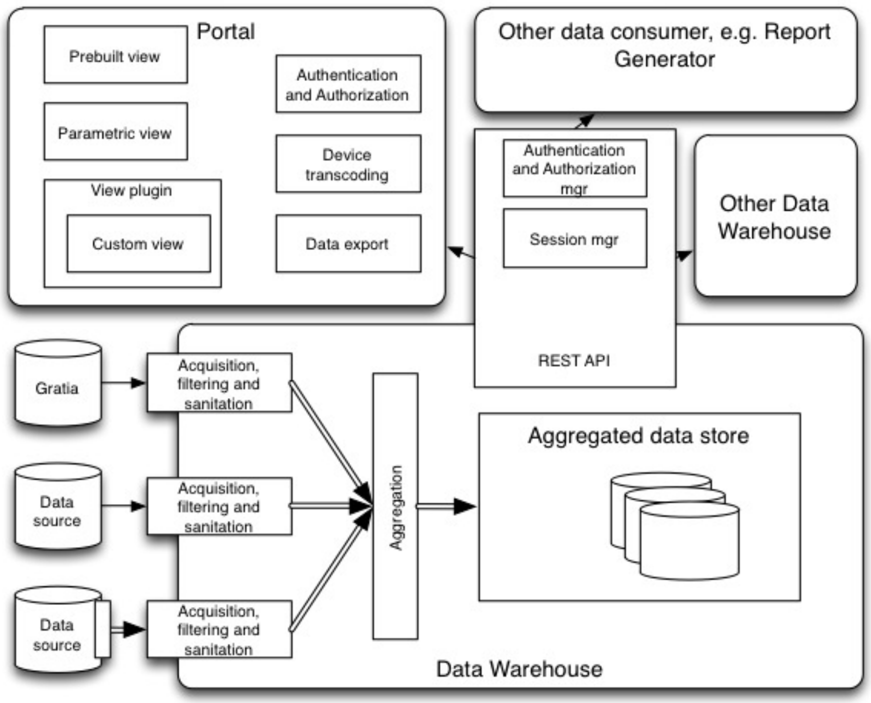
Portal
Prebuilt view
Parametric view
View plugin
Custom view
Authentication and Authorization
Device transcoding
Data export
Other data consumer, e.g. Report Generator
Authentication and Authorization mgr
Session mgr
REST API
Other Data Warehouse
Gratia
Data source
Data source
Acquisition, filtering and sanitation
Acquisition, filtering and sanitation
Acquisition, filtering and sanitation
Aggregation
Aggregated data store
Data Warehouse

Figure 1 - CIVAIS architecture diagram

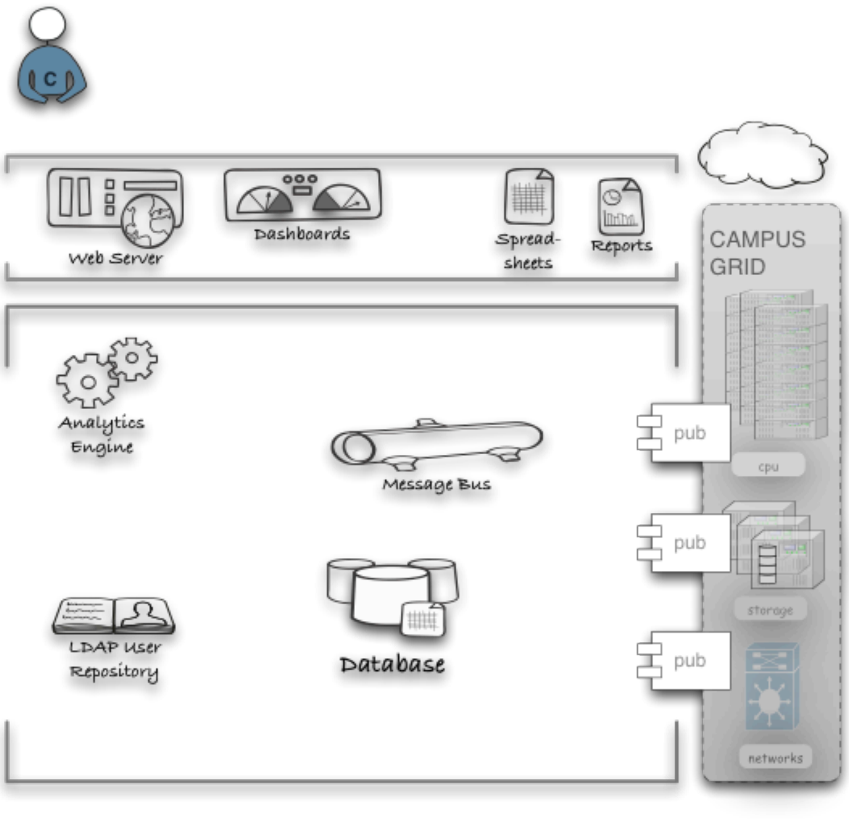

Figure 2 - Components in CIVAIS

Milestones

- Create version 1 of a whitepaper with broad outline of use cases, requirements and reference architecture (**this document**). (April 8)
- Create a presentation to highlight motivations (April 8)
- Create and conduct surveys for
- Users
- Faculty sponsors, research advisors and college deans
- CI directors and resource providers
- Mock-up of the service ("what would it look like") (July 1)
- Bi-weekly progress checkpoint starting July 12 until the beta release (7/12, 7/26, 8/2, 8/16, 8/30, 9/13)
- Prototype for feasibility running on UC3 (July 26)
- Alpha release (August 30)
- Deployment of the alpha release in a campus grid environment (UC3) (July 26)
- Deployment at a second campus (August 16)
- OSG CIC Webinar (September)
- OSG CIC iFORUM (September)
- Feature set for production release (mid of September)
- Production 1.0 release (end of September)

References

[1] Gratia, http://www.opensciencegrid.org/bin/view/Accounting/WebHome

[2] XD Metrics on Demand portal, https://xdmod.ccr.buffalo.edu/

[3] WLCG Dashboard, http://dashboard.cern.ch/

[4] Cycle Computing, http://www.cyclecomputing.com/

[5] Message Queue comparison on StackOverflow: http://stackoverflow.com/questions/731233/activemq-or-rabbitmq-or-zeromq-or

[6] MQ comparison and test: http://mikehadlow.blogspot.com/2011/04/message-queue-shootout.html

[7] Reasons to use a MQ: http://www.bitmechanic.com/2011/12/30/reasons-to-use-message-queue.html

[8] ZeroMQ guide: http://zguide.zeromq.org/page%3aall

[9] Article "Opening government, the Chicago way" http://radar.oreilly.com/2011/08/chicago-data-apps-open-government.html

[10] GovStat, an Open Data platform by Socrata (http://www.socrata.com/)

[11] Socrata Open Data Field Guide: http://www.socrata.com/open-data-field-guide/

[12] Socrata Open Data API (SODA) http://opendata.socrata.com/api/docs/ and Python API https://github.com/socrata/socrata-python

[13] Getting started with SODA 2.0, a guide for developers accessing or publishing Open Data: http://dev.socrata.com/consumers/getting-started

[14] Opening data the Chicago way, O'Reilly radar: http://radar.oreilly.com/2011/08/chicago-data-apps-open-government.html

[15] "Pootholepalooza", a "Weekend Festival of Pothole Reporting", where Chicagoans are encouraged to report potholes to CDOT http://www.cityofchicago.org/city/en/depts/cdot/provdrs/street/news/2013/apr/_potholepalooza_hitschicagothisweekend.html

[16] Byte cop, Brett Goldstein maps out Chicago's crime-ridden spots with his predictive-analysis system: http://magazine.uchicago.edu/1102/arts_sciences/byte-cop.shtml

[17] Global Accounting in the Grid and Cloud, John Gordon, HEPiX 2012