The original architecture of PhEDEx

PhEDEx was written as a set of agents, each performing a highly specific task. Some agents run only at a single, central site, others run at each site, configured for the specific conditions at that site.

Agents do not communicate with each other directly, only through changes they make in the central Oracle database.

This architecture corresponds to a toolkit of ‘power tools’. The tools are not general purpose, and can only be used for the single task they are designed for.

This keeps the agents clearly separated, which makes it easier to code and maintain them. It also simplifies tuning a PhEDEx agent for performance with over growing volumes of data.

Problems with the original architecture

The number of PhEDEx users is increasing. Operators want more detailed views of PhEDEx, physicists want to know about their own data, and experiment managers want summaries and higher level overviews of the behaviour of the whole system.

Many people write their own tools, often with significant functional overlap but little common code. This is a poor use of manpower.

There was minimal code sharing between agents, neither of the Perl code nor the SQL statements they use. Duplicate code leads to maintenance and behavioural inconsistencies. Database load can increase if several variants of the same SQL statement are used.

When PhEDEx was originally developed, there was no freely available event-driven framework which could be used to build the agents. This made the agents less flexible than they could have been. Multi-threading, for example, was all but impossible to achieve.

Developing a new agent is a specialised task. For new developers, the learning curve can be significant, which makes it hard to use people who can only contribute a fraction of their time.

The solution

We are re-engineering PhEDEx as a collection of open frameworks. People should be able to develop code that is supported by PhEDEx, so they can accomplish their task easily, efficiently, and in a way that can be shared by others.

PhEDEx now has several lightweight frameworks for developers. They are the data service, the next-gen website, the AgentLite framework, the Namespace framework, and the LifeCycle agent.

The data service

PhEDEx provides a web-based data-service* for access to information. Data is available in Perl Data: Dumper, JSON and XML formats. A large number of API calls provide fine-grained selection of data.

Some APIs are generic, such as those for exploring replicas of data at sites, others are more task-specific, such as the one that reports stuck transfer queues at sites, based on a complex algorithm.

New APIs are being added constantly to address new needs of the user community.

AgentLite and the core agent framework

AgentLite agents were re-engineered some years ago to use POE*, the Perl Object Environment. This is an open source event-driven framework for cooperative multi-tasking. As a result, agents, or specialised tasks like monitoring, can be coded independently as separate POE programs, then combined in the same process for ease of management.

At the same time, SQL statements were factored out into libraries of functions which could be shared. Use of caching in the library functions further reduces the load on the database while keeping agent code simple.

Now we are breaking the agent behaviour into separate modules, for database interaction, droplet queue management, and others.

Specialised agents, like the PhEDEx watchdog, can be written using the AgentLite framework, dropping the parts they do not need. It can also be used for non-PhEDEx work, or outside CMS.

References

1. CMS Data Transfer operations after the first year of LHC collisions, CERN 2012
2. Data transfer infrastructure for CMS data taking, ACAT 2008
3. The PhEDEx data service, CERN 2010
4. POE*
5. Data storage accounting and verification in LHC experiments (L3 224)
6. HEP4: Establishing a new generation of service for distributed analysis at CMS (L3 240)
7. Data bookkeeping service 3 - A new event data catalog for CMS (L3 187)
8. Front-End: A tool for the LHC-HEP waits-queue (L3 147)

Conclusion

This has been a work in progress for several years, as we opportunistically refactor parts of PhEDEx.

We expect this work to continue, with the move from a dedicated toolkit to a set of open frameworks being a guiding principle of the way PhEDEx is developed in the future.