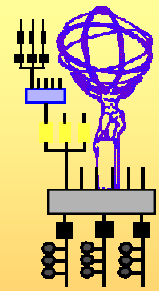
EXPERIENCE WITH CORBA COMMUNICATION MIDDLEWARE IN THE ATLAS DAQ



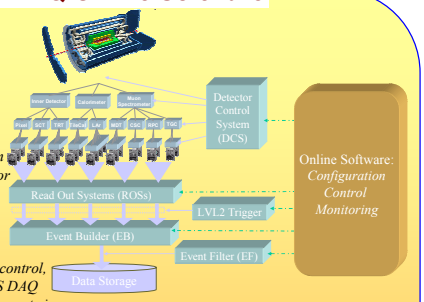
S.Kolos, University of California, Irvine, USA
D.Liko, D.Burckhart-Chromek, J.Flammer, M.Dobson, R.Jones, L.Mapelli, CERN, Geneva, Switzerland
I.Alexandrov, V.Kotov, M.Mineev , S.Korobov, Joint Institute for Nuclear Research, Dubna, Russia
A.Amorim, N. Fiuza de Barros, D.Klose, L.Pedro, Universidade de Lisboa, Faculdade de Ciencias, Lisbon, Portugal
E.Badescu, M.Caprini, National Institute for Nuclear Physics and Engineering, Bucharest, Romania
A.Kazarov, Y.Ryabov, I.Soloviev, Petersburg Nuclear Physics Institute, Gatchina, Russia

1. ATLAS DAQ Online Software

Atlas
ATLAS is one of the four experiments in the Large Hadron Collider (LHC) accelerator at CERN.

DAQ System
The ATLAS Data Acquisition (DAQ) system transports event data from the 1600 detector read-out links to mass storage.

Online Software
The Online Software is responsible for the control, configuration and monitoring of the ATLAS DAQ system. Online Software has to be able to operate in highly-distributed multi-partition computing environment.



In order to meet these requirements it has been decided to use CORBA based communication middleware as a basis for the DAQ Online Software implementation.

2. CORBA

What is CORBA
CORBA is an open standard for distributed object computing, which has been proposed in 1991 by the Object Management Group (OMG). It standardizes many common network programming tasks such as object registration, location and activation; parameter marshalling and demarshalling; operation dispatching and so on.

Advantages
- *CORBA supports most of the widely used programming languages, like for example C, C++, Java; as well as a number of scripting languages like for example Python.*
- *Different implementations of the CORBA standard are able to communicate with each other. This feature is called interoperability and it is one of the most important advantages of CORBA.*
- *All CORBA compliant brokers are compatible at the source code level. This means that any CORBA based user's program can be compiled with almost no changes using different CORBA brokers.*
- *CORBA provides a high-level object-oriented paradigm for the distributed programming hiding low level aspects of the communication implementation.*

Drawback
The API and communication model, which are provided by CORBA, are considerably complex. This complexity is a result of the flexibility, which CORBA offers for the distributed application development.

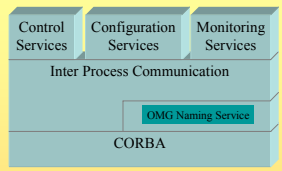
In order to overcome the CORBA complexity, a light-weight software wrapper, called Inter Process Communication (IPC), has been implemented on top of CORBA in both C++ and Java.

3. Inter Process Communication

IPC
IPC is a thin wrapper above the CORBA and the OMG Naming Service .

Online Software services
The Online Software provides a number of software services, which are implemented on top of IPC.

Groups of Services
All the services are separated out into three groups:
- *Control*
- *Configuration*
- *Monitoring*



IPC provides the following features for the Online Software services:
- ***Simple API for the core CORBA facilities***
- ***Simple API for the OMG Naming Service***
- ***Transparent cache for the remote object references***
- ***Notion of the DAQ Partition***

4. IPC API

The Run Control IDL interface

```
module rc {
  interface controller : ipc::servant {
    void command( in string cmd );
  }
}
```

A Run Control Controller implementation

```
class RCController :
   public IPCNamedObject<POA_rc::controller>
{
   RCController( const IPCPartition & p,
                 const std::string & name )
   : IPCNamedObject<POA_rc::controller>( p, name )
   { publish(); }

  ~RCController()
   { withdraw(); }

   void command( const char * cmd )
   { ; }
};

int main( int argc, char ** argv )
{
  IPCCore::init( argc, argv );

  IPCPartiton partition( "MyPartition" );

  RCController * rc;
  rc = new RCController( p, "MyCtrl" );
  ...
}
```

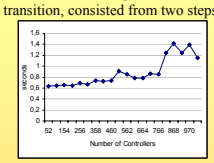
Sending command to the Run Control Controller

```
int main( int argc, char ** argv )
{
  IPCCore::init( argc, argv );

  IPCPartiton partition( "MyPartition" );

  rc::controller_var rc;
  rc = partition.lookup<rc::controller>( "MyCtrl" );

  try {
      rc->command( "MyCmd" );
  }
  catch( CORBA::SystsemException & e )
  { ; }
}
```

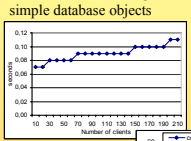
A beginning software developer can easily learn the IPC API in several hours.

5. Performance measurements

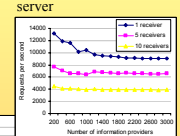
Mean time for a single Run Control transition, consisted from two steps



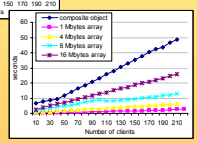
Mean time for Reading simple database objects

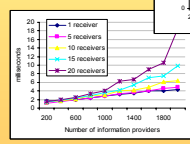


Number of messages per second, which can be handled by a single Information Service server



Mean time for reading complex and large database objects





Mean time for a single information update

These measurements show that CORBA provides the necessary performance and scalability for the distributed ATLAS DAQ.

4. Conclusion

- **The control, monitoring and configuration facilities for the ATLAS DAQ system are implemented on top of the CORBA standard.**
- **The only drawback of CORBA is a complicated API and communication model, which implies a significant learning curve.**
- **In order to overcome this issue a special software wrapper, called IPC, has been provided on top of CORBA.**
- **IPC dramatically simplifies the usage of CORBA in the scope of the ATLAS DAQ.**
- **A beginning software developer can easily learn the IPC API in several hours.**
- **This makes the software development process faster and efficient and results in the production of the highly effective and reliable software.**
- **Performance and scalability studies show that CORBA provides the necessary performance and scalability for the distributed ATLAS DAQ.**