

A decorative graphic on the left side of the slide featuring three balloons in green, light blue, and purple, each with yellow streamers and triangular flags.

Configuration Database

David Forrest
University of Glasgow



Celebration

▶ We have a configuration database system

This talk presents the state of the system as it is, my thoughts on work going forward and answer questions about integrating the database with other systems as well as how to actually use it.

The configuration database (CDb) replaces the spreadsheet and provides extra functionality for data managing cabling, calibration, geometry and alarm handler information in addition to a superset of the set values by shifters which were contained in the spreadsheet

For extra pointers on how to develop apps using the CDb please see my talk at CM26 or mice note x

This Talk

- I want to discuss:
 - Status
 - Performance
 - Backups, security
 - Handover
 - Mitigating future issues
-

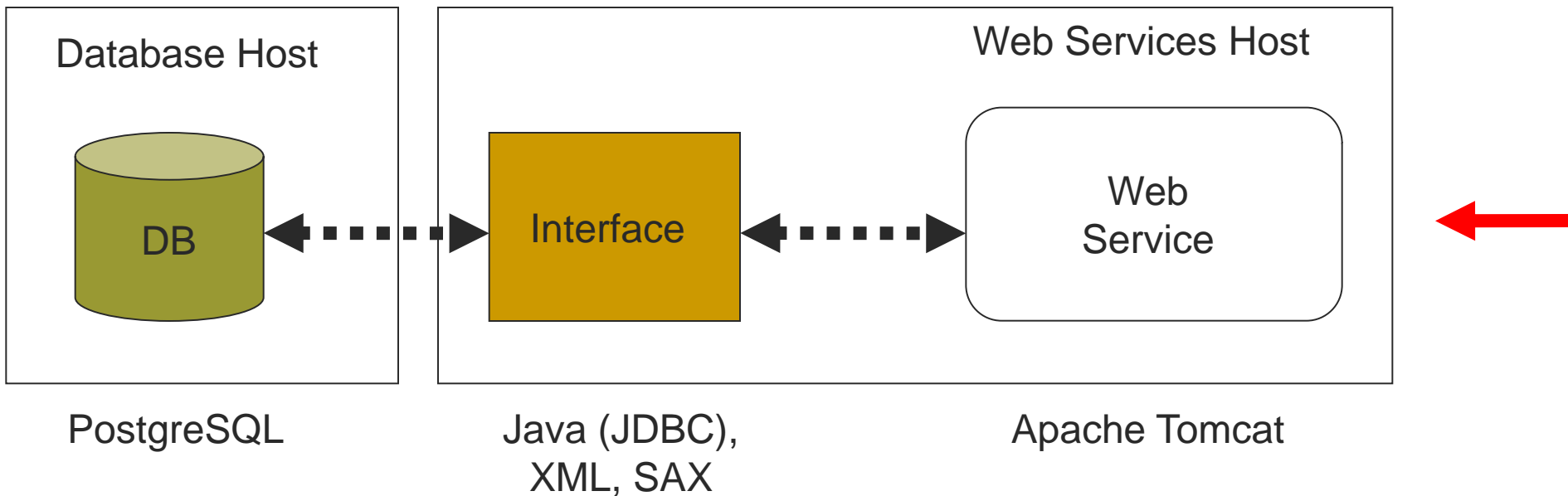
The Other Talk

- In the other talk I will discuss how to use the configuration database web guis
 - I will talk about development of G4MICE applications which is a priority
 - This is the 'what it is' talk and the next one is the 'how do I use it' talk
-

Status

- The config database itself has been ready for some time, certainly before the current diet of runs(!)
- The limiting factor has been installation of hardware and interfaces between systems at RAL – external to the database
- That is now complete!
- However the time taken to get the hardware in place and achieve necessary network changes and delay in links with other systems has severely shortened the window between installation and end of PhD in a manner which was not unforeseen – see talks from previous CMs

Diagram

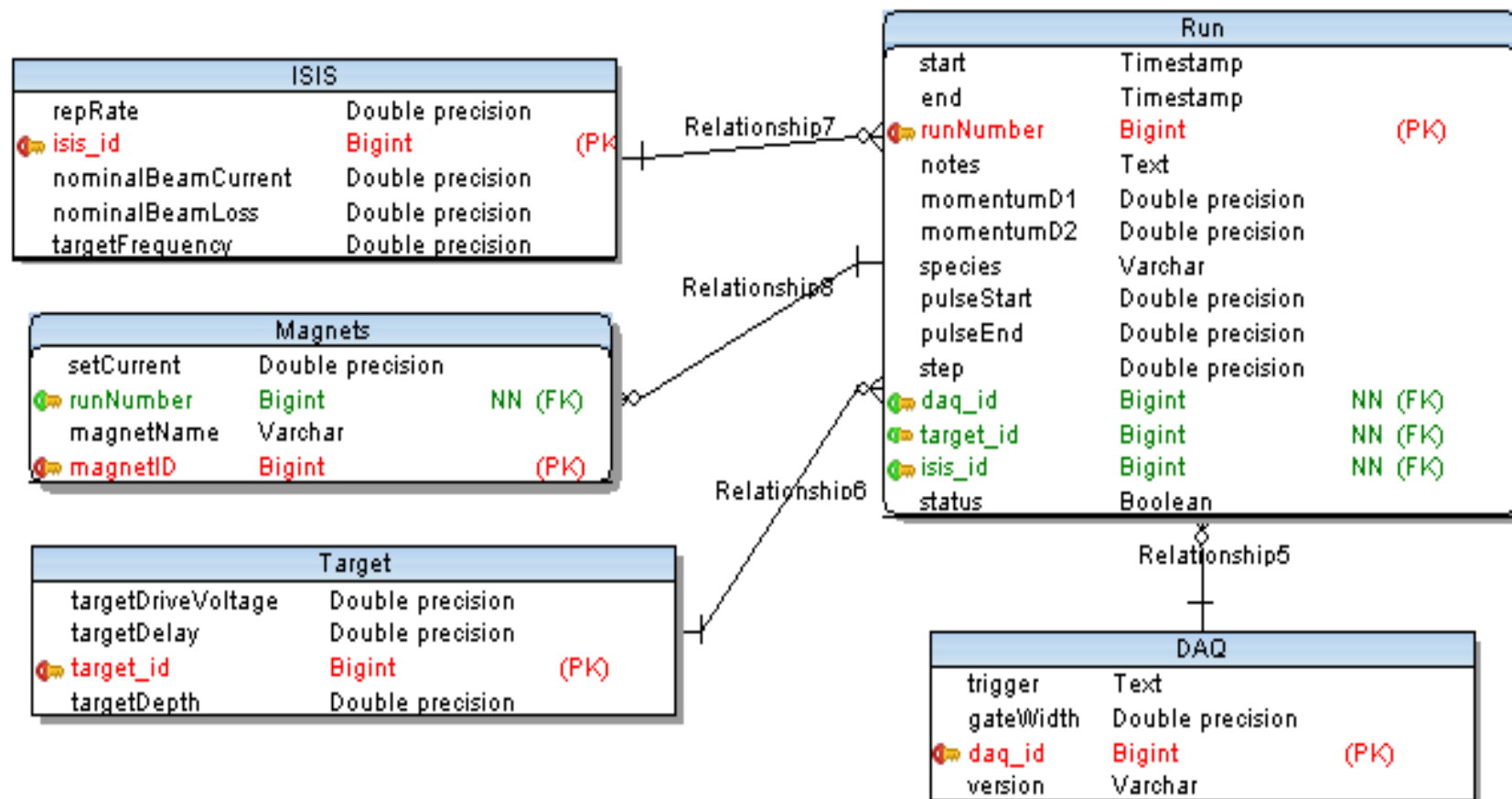


Clients are independent of database and vice-versa

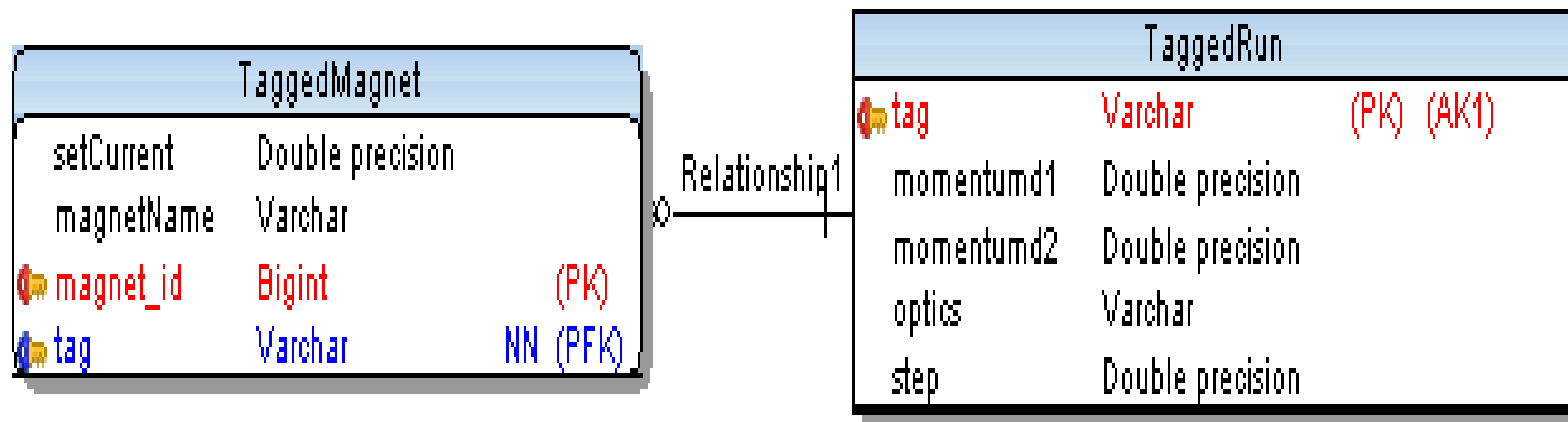
People don't fiddle with the DB directly but call safe pre-defined functions in the Interface

Able to communicate via Tomcat over HTTP from laptop, cluster, grid node, phone, anything

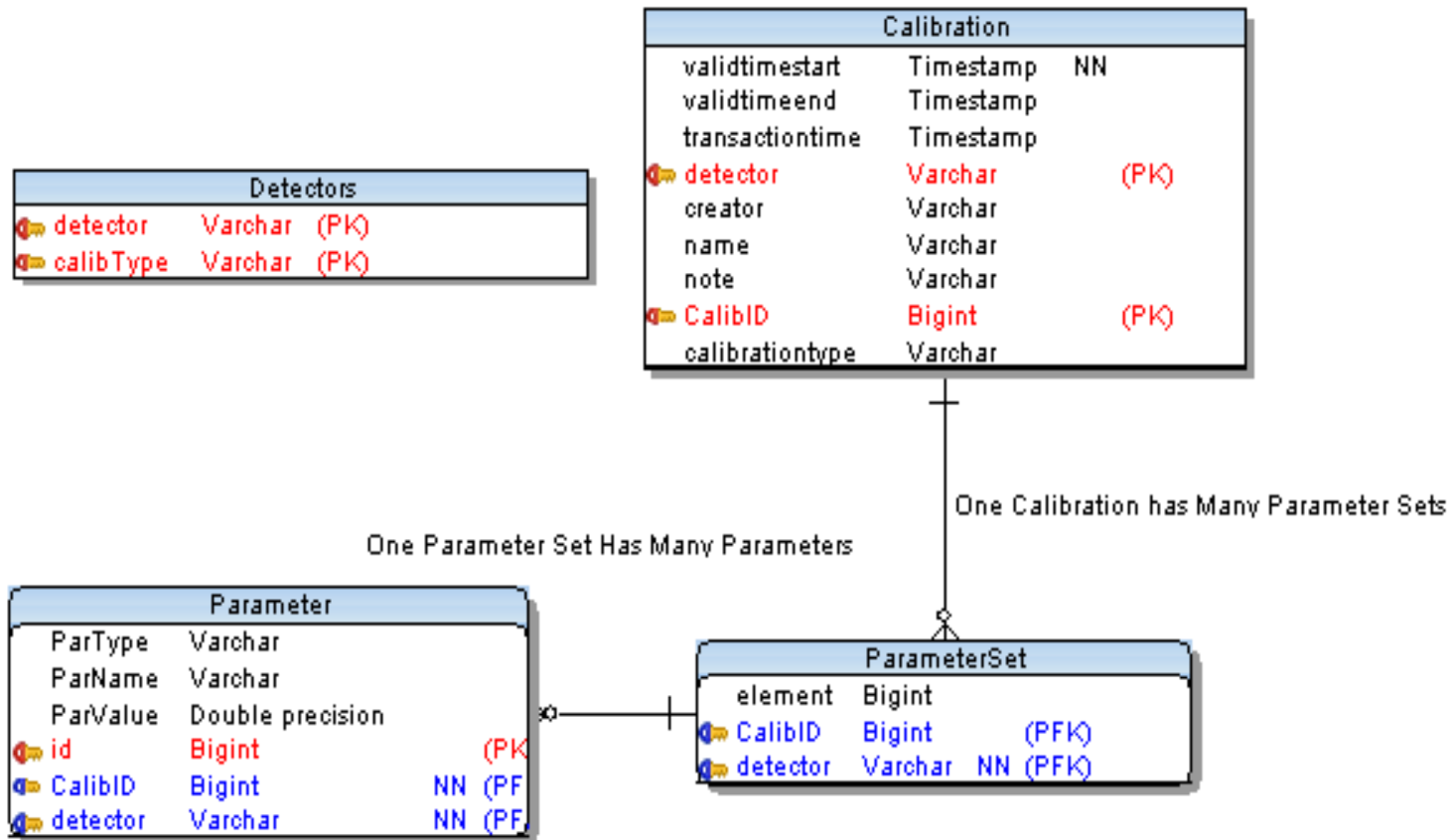
Runs



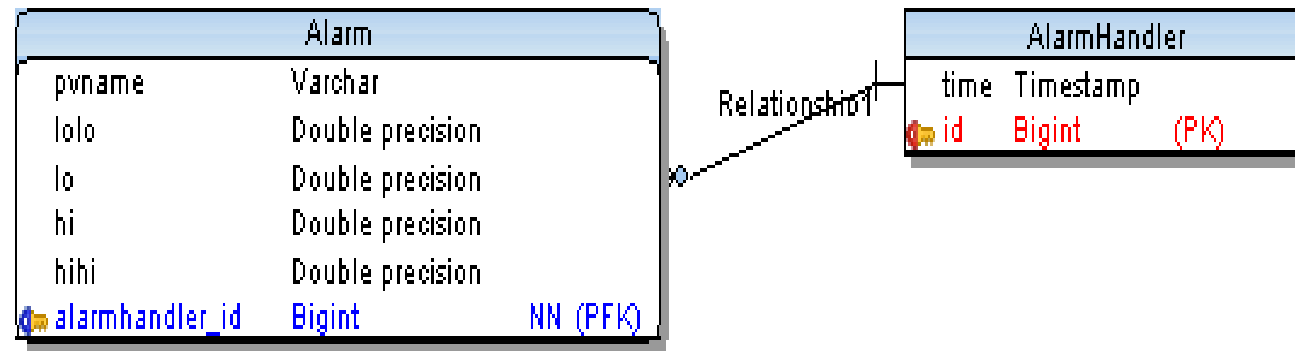
Tagged Runs




Calibrations



Alarm Handler



Geometry

MICEModules		
 micemoduleID	Bigint	NN (PK) (AK2)
configurationID	Bigint	
name	Varchar	
timestamp	Timestamp	
volType	Varchar	
dim1	Double precision	
dim2	Double precision	
dim3	Double precision	
positionx	Double precision	
positiony	Double precision	
positionz	Double precision	
rotationx	Double precision	
rotationy	Double precision	
rotationz	Double precision	
scaleFactor	Double precision	

And so on

- I haven't shown you cabling, which changed, and I have to put some finishing touches on. It'll be in the note.
 - I have had to skip a lot. If you want more detail on the design please consult the set of previous talks at collaboration meetings.
-

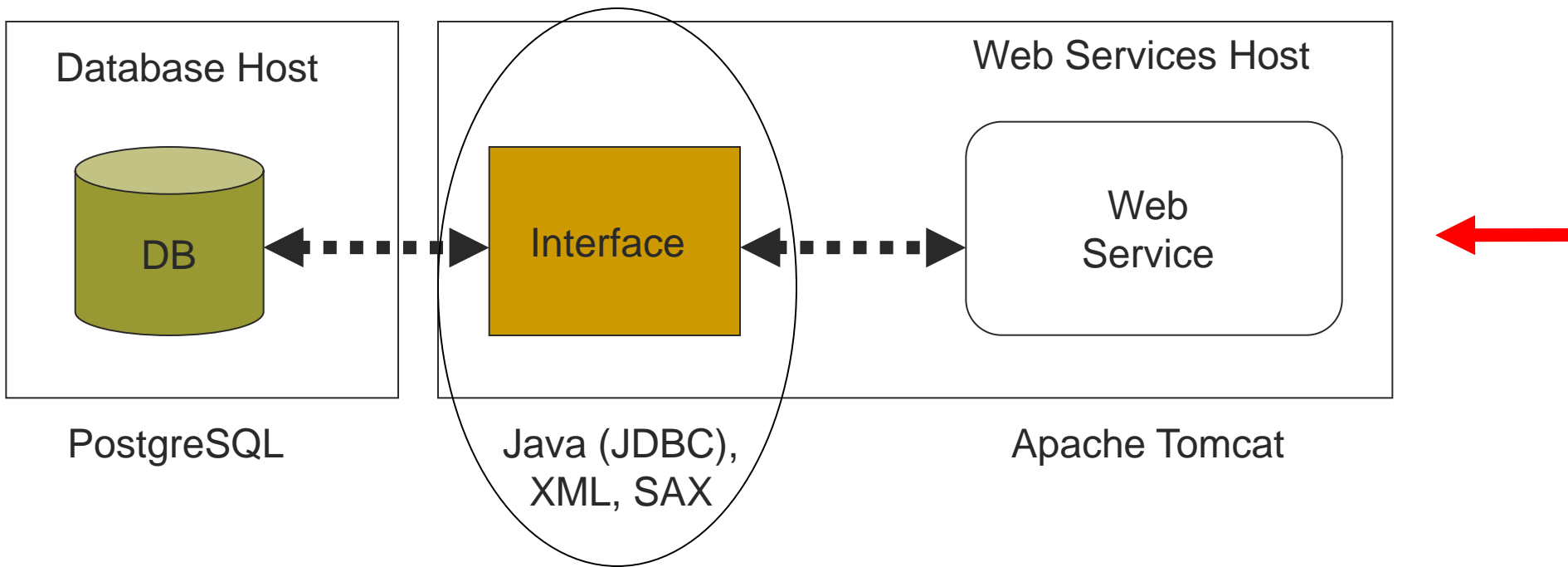
How to Use It

- Control Room
 - G4MICE
 - Web Interface
 -
 - Future Development
-

The implementation

- The Configuration Database System comprises:
 - Postgresql Database Management System
 - Apache Tomcat web server with Java based interface for database functionality
 - Web gui frontend
-

Diagram



API

- The API consists of lots of functions...
- eg:
- `getSetValues(run_number)`
- `getSetValues(time)`
- `SetSetValues(...)`, `updateSetValues()` etc
- And so on for the different domains of the database

API

■ You can call these functions by sending xml eg

- `<?xml version='1.0' encoding='UTF-8'?>`
- `<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance" xmlns:xsd="http://www.w3.org/1999/XMLSchema">`
- `<SOAP-ENV:Body>`
- `<ns1:getSetValues xmlns:ns1="urn:miceapi6" SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">`
- `<ns1:input xsi:type="xsd:int">1999</ns1:input>`
- `</ns1:getSetValues>`
- `</SOAP-ENV:Body>`
- `</SOAP-ENV:Envelope>`
- Of course you don't need to know anything about XML...this is just whats under the hood of the client program which takes your request for run 1999.

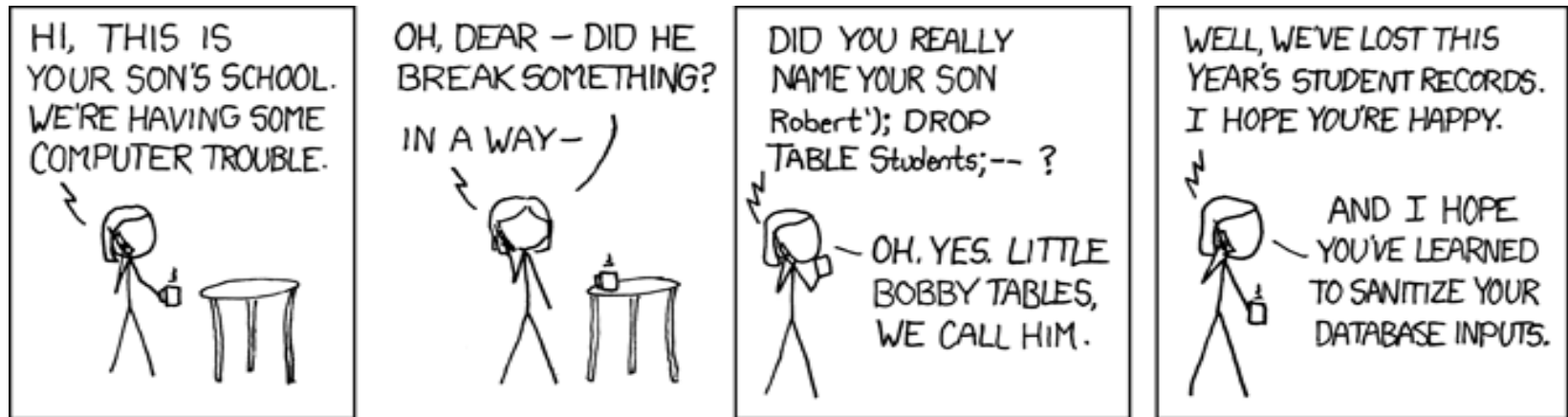
Web Guis

- Some simple use cases, for straight forward referencing, will be provided using web guis.
 - A link will be requested from the MICE website (MICO+Software pages?)
 - More on using these in my talk in the demonstration session where you can see it all working
-

Meta Data Catalogue

- Decided with David Colling that we needed to change manage integration of meta data catalogue (MDC) with CDB
 - Work on MDC progressed with CDB integration in mind (eg Postgres platform)
 - Integration should be straightforward in principle, requirements for how we access it should be carefully considered
-

Security



Credit: xkcd.com

- The only writing to the database will be done from the control room
- Other applications can only read (eg web user interface)
- Database inputs are sanitised

Backups

- A cron job runs on the web server and the DB server
- Currently db critical files sent to a backups folder on the same machine every five minutes
- These are picked up by the backup bot once an hour – this can be increased to match the db periodicity, but I have left the DB backup frequency at 5 mins in the meantime. This is not a technical limit.
- This also forms the perfect basis for a read-only mirror of the DB itself, which I don't believe is immediate to our needs but you can pursue
- API in a state which is easy to set up on another machine, can use portable java files
- I don't want to talk about a mirror on this slide, leave it til the questions at the end :)

Testing

- EPICS writes to the database at the end of every run
- It may or may not request information just prior to the start of a run (eg if a previous configuration is to be repeated)
- Similarly, a G4MICE application would at maximum contact the db once at the beginning of a simulation or analysis task
- *****When running many simulations on the Grid, accesses are not synchronous or sustained*****
- I expect a low traffic rate $O(10)$ Hz, but I have not constrained the performance of the DB which is tunable
- I have tested using performance metrics which I will now present
- I have artificially forced accesses to be synchronous. Not natural circumstances. Had to learn some new tricks to do that 😊

Caveats

- The situation I am testing is not normal running, it is far beyond it.
- I am trying to address the concern – can a large number of reads delay writing from the control room
- I do not believe this is a showstopper risk for many reasons including but not limited to the possibility of giving control room a reserved connection
- I am ignoring that and exaggerating the circumstances then providing a test to allay fears....not to stir them :)
- Anyway we can have a dedicated connection for the control room

Analysis of Whole System Performance

- I wanted to look at synchronous remote access from many locations
 - So I decided to use the Grid (if there is a simpler way of doing this please don't tell me)
 - The tests which I present here reach significantly beyond our present need or capacity(!)
-

Grid analysis

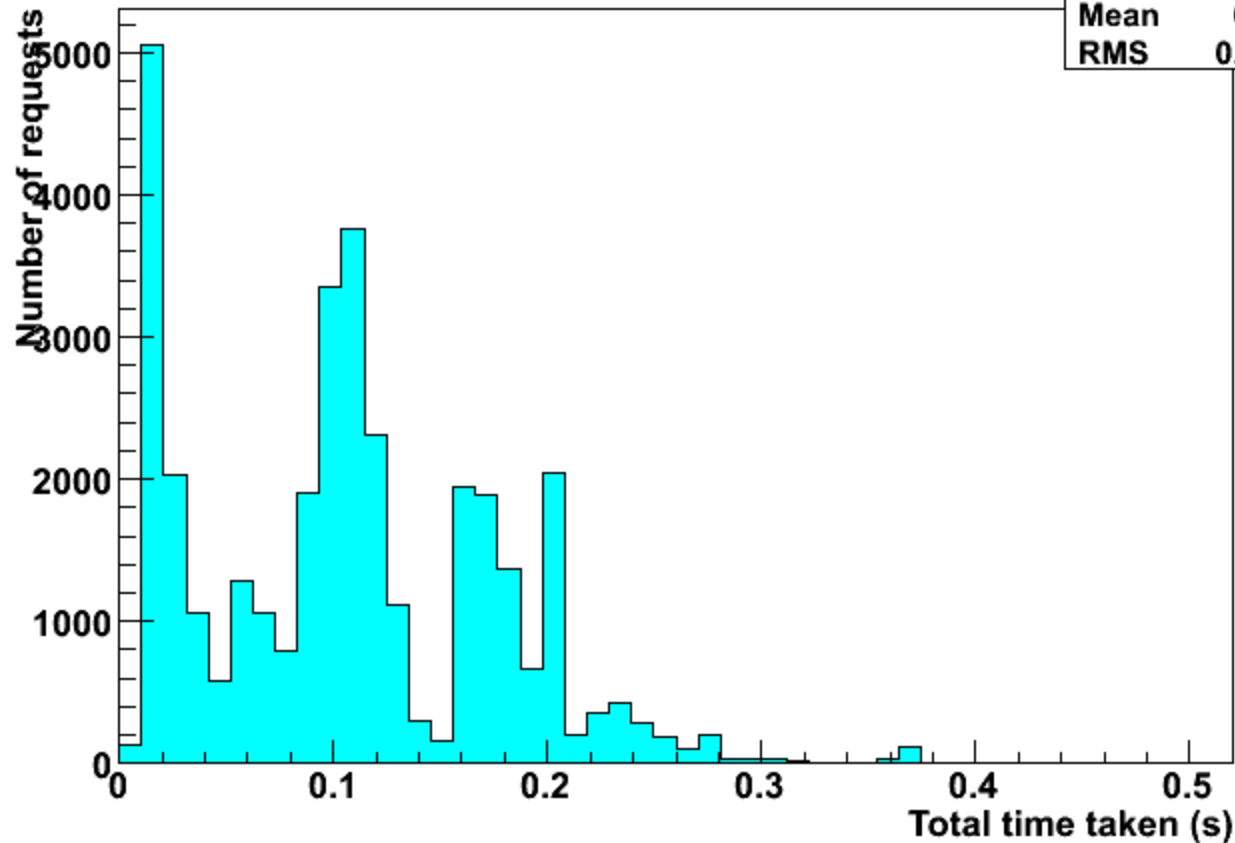
- I launched a job with $N=50$ scripts communicating with the database over a sustained period of time
 - I measure the time taken for connection, transfer of request, result acquisition, close of connection, operation on result – a realistic scenario.
-

$N=0$

- We want to compare the amount of time taken to make a “write” of set values (control room operations) with no traffic, and with significant traffic
 - So our starting point is no traffic, Number of client nodes= $N=0$
 - Time taken : 0m0.109s (obviously this number will fluctuate depending on client node)
-

Results N=50

Time taken for N=50



Time Taken	
Entries	34962
Mean	0.1077
RMS	0.06943

Time for write: 0.110s (compared with 0.109s)

Traffic: ~600 accesses per second

Results $N > 50$

- Maximum traffic not determined, probably around 3000 per second. As the traffic approaches this, total time only drops by 0.06s
- Maximum traffic is not a hard limit, as the database is tunable; presently configured for ~600 concurrent clients globally (Postgres defaults to 100)
- We can calculate what the performance metric should be, identify the magic number and tune for it
- If traffic went beyond this it would not crash any system, it would just return an error FOR READ OPERATIONS
- **However, even when traffic is beyond a maximum you can still keep a reserved connection FOR WRITE OPERATIONS (control room processes)**
- The system is designed to cause as little disruption to running as possible.

So...

- High traffic perhaps not very realistic
 - But DB robust to high traffic anyway
 - Total time taken for connection, request, result, close of connection and parsing of result ~ .1 sec
 - Should not delay start of next run (cf DAQ set up ~ 20 secs, can be done in parallel)
-

Data taking

- Its at the forefront of my mind throughout this that a failure in the database, no matter how unlikely, should not impair data taking
 - You can mitigate this on the client side
 - What happens if the DB goes down? EPICs fails to write. It sets an alarm. It writes later. Nothing should be lost. The run plan can continue.
 - It should take seconds for an expert to bring the CDB back up again.
 - Aside from networking issues I am not sure exactly how to bring it down...but that should not be interpreted as a challenge-!
-

Maintenance



- I am at an advanced stage in providing documentation for the database for users, maintainers and developers
- I have tried to cover things I think can come up eg 'what if the server goes down' and 'how do I add new functionality'
- See forthcoming MICENote. Acme Bear Wax sold separately.

Handover

- Paul Kyberd is well placed to oversee future work on the database, by himself and others
 - A couple of currently unnecessary but easily added pieces of functionality have deliberately been left out so that PaulK and co can try to implement them now
 - If problems arise its best that they arise when I'm at least contactable by phone for advice
 - I have offered to run a workshop where we can do real development of such features – early august? BOOK NOW with Malcolm Ellis to avoid disappointment.
-

Training

- You will need:
 - 2x DB Experts
 - N x DB Users
 - 1 x DB developer
 - 2 x Client Developers
 - There may be some overlap between some of these
-

The Pitfalls of the CDB

- It is difficult to organise a system which sits on the interface of many systems.
 - It is not just a technical challenge (easy?) but a human one
 - You are everyones 'second priority', if lucky
 - I have found that I cannot force people to make their system interact with the database, or to use it
 - But if we don't get this right, we will create our own problems
 - So the system actually needs to be championed as well as implemented, requires support at all levels to be driven pushed and shoved forward, due to many interfaces with other systems
-

“Technical” pitfalls and tips

- Don't write `System.exit` in an exception handler for a web application. Use the `<error>` tag or SOAP fault string (see template functions in `MICEapi.java`)
 - Don't do dev work on a live copy of any of the database systems (eg coding/compiling/stress testing), make a local copy on your laptop
 - Always sanitise database input to mitigate SQL injection, especially for web guis. You can use parameterised queries for this.
 - Excellent tomcat/postgres support available in docs, google and `#postgres`, `#tomcat` on `irc.freenode.net`
 - Stick to the scope of configuration applications – resist mission creep...should not be a data store or a personal organiser!
-

Who's Who?

- Who to ask questions to ...
 - CDB requirements & development up until now
+ web guis: David Forrest
 - CDB going forward: Paul Kyberd (but it would be nice if you leave the hard questions until after the documentation and workshop!)
 - EPICS client: James Leaver
 - G4MICE client: Vassil Verguilov & Malcolm Ellis
-

My last things to do

- Give this presentation (I hope to get this done sometime today)
 - Finish documentation
 - Finish cabling (client required!)
 - Provide a hands on workshop
 - Fix a couple of bugs found in demos (may get done today)
-

Other things to do

- Client code!!!!!!
- Include diffuser thickness, proton absorber thickness and run meta data like emittance etc
- Training
- Verification

Thanks

- Thanks to those who have developed client code so far (James Leaver, Malcolm...)
 - Thanks to Malcolm Ellis for enormous support throughout this project
 - Thanks for trusting a PhD student to do this
 - Thanks to everyone who attended the launch party in the rack room - really, really sorry about the mess.
-

Conclusions

- Overview
- Performance
- Testing
- Operation
- Future development

This is probably my last collaboration meeting, so please take the opportunity to ask questions about the system. I'm here just today.