

# Online Data Collection: better way

Dmitry Arkhipkin

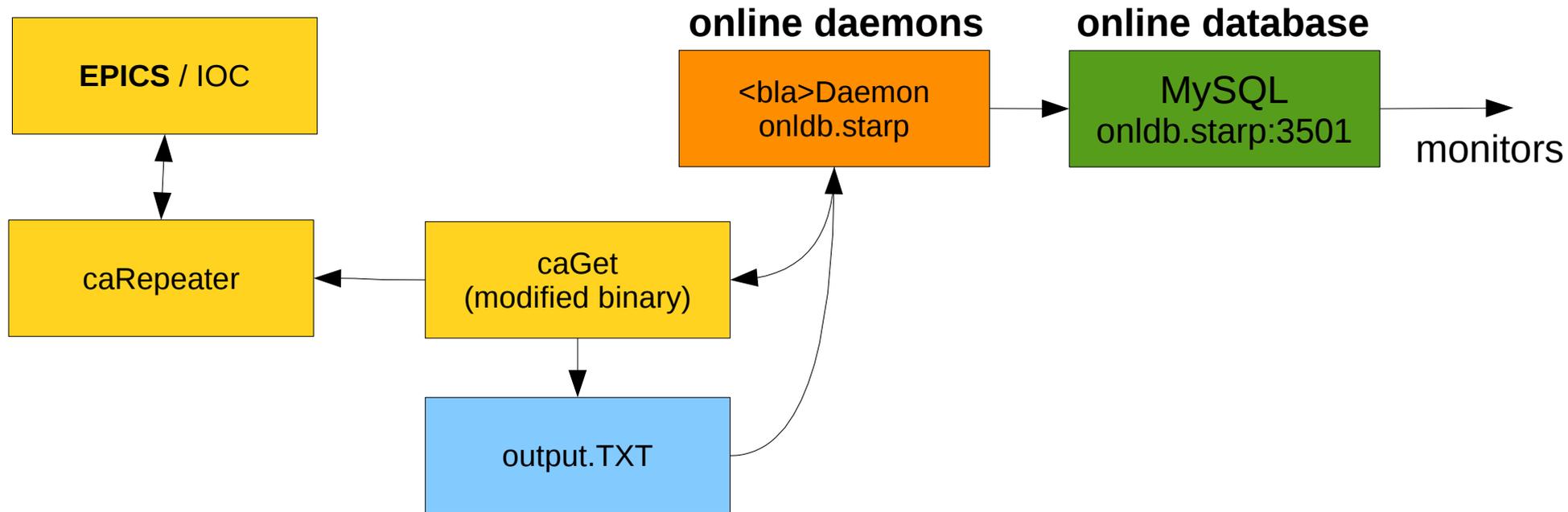
STAR S&C Group  
Brookhaven National Laboratory

# Overview

- “Old vs New” Online data collection setup;
- MQ-based setup design details;
- MQ-based setup implementation details;
- Prototype of MQ-based setup:
  - Deployment
  - Performance
- Summary & ToDo;

# Online data collection and monitoring : existing setup

**Current setup:** onldb.starp node has ~10 <bla>Daemon services running, which execute external binary (caget) periodically, parse resulting txt file, store data to MySQL;



## Notes:

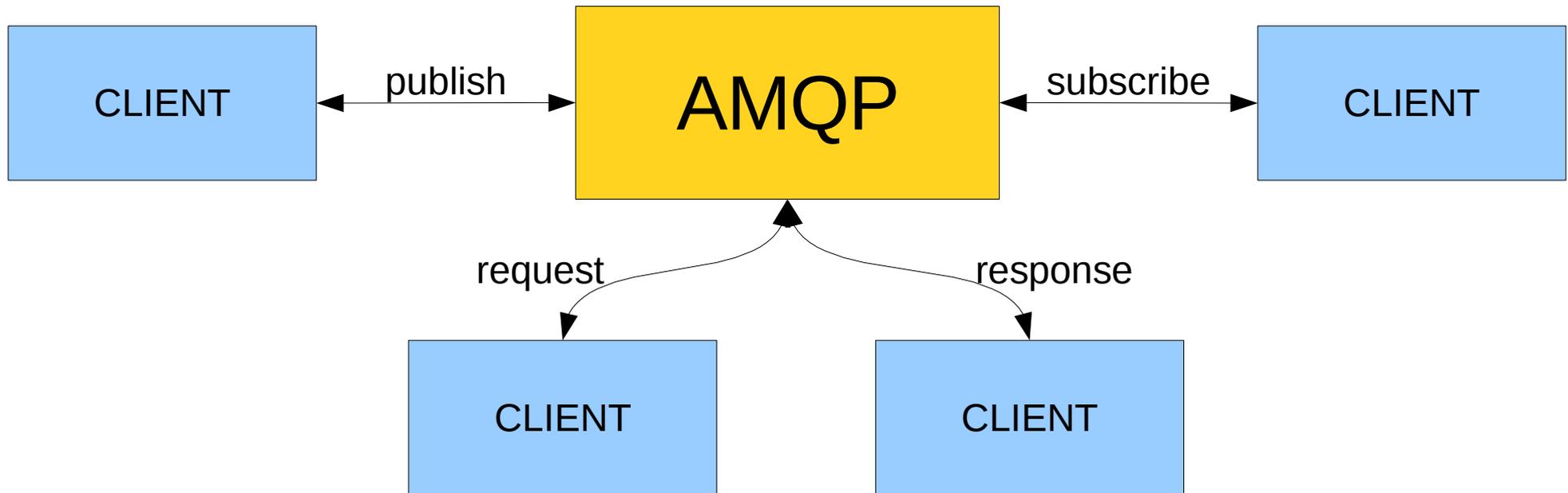
- a) caget binary is a modified sample code from EPICS CA package, unfortunately, there's no source code left, only binary..
- b) <bla>Daemons contain detector-specific logic, and know how to work with MySQL only. It is somewhat non-trivial to add new subsystem or switch to some other db backend. Data providers are tightly coupled to data storage and data consumers (private storage format);
- c) data monitoring codes are polling very same MySQL instance for data;

# What do we expect from “New”:

- Stable data collection flow;
- Better data stream **and** service monitoring;
- Better cross-system communications;
- More options for real-time data processing;
- Clean codebase & easy installation and maintenance;

# Online data collection and monitoring : new approach

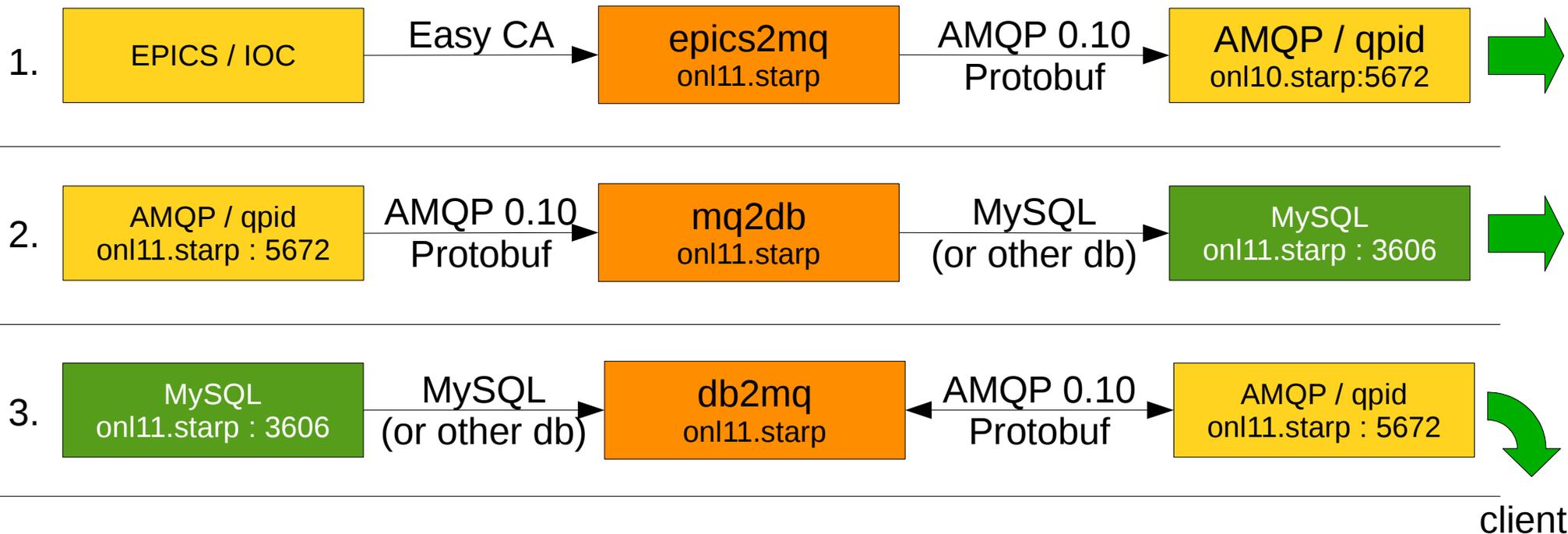
## Let's use Message Queuing System!



MQ allows to convert existing complex, monolithic system into set of simple, loosely coupled services. Greatly enhances scalability of a project..

# Online data collection and monitoring : new approach

**Proposed setup:** use message bus to decouple data providers from data storage, data consumers and data monitoring; Let's introduce these three simple components: epics2mq, mq2db, db2mq (see below).

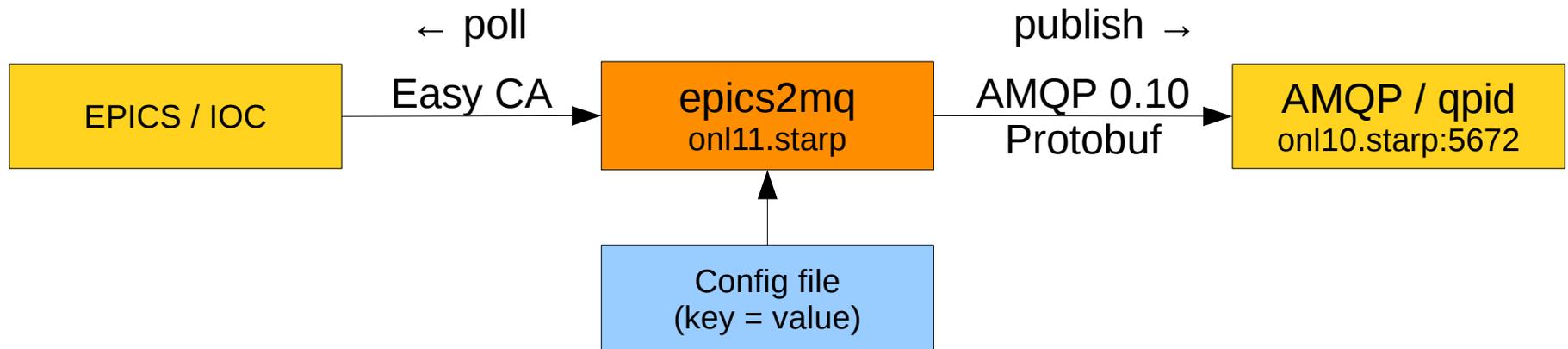


## What we get:

- loose coupling: only Protobuf structure descriptions are shared between all services;
- easy extension/expansion of any component;
- no more constant database polling by monitoring code;
- messages could be routed to external facilities (Remote Control Rooms or monitors);
- ability to use any storage engine we like (speed, maintenance, other factors)

# epics2mq ( publish )

**epics2mq** : daemon, which polls EPICS periodically, serializes results using Protobuf, and sends message to AMQP server. EPICS & AMQP server parameters are read from config file, as well as epics channel names to poll...



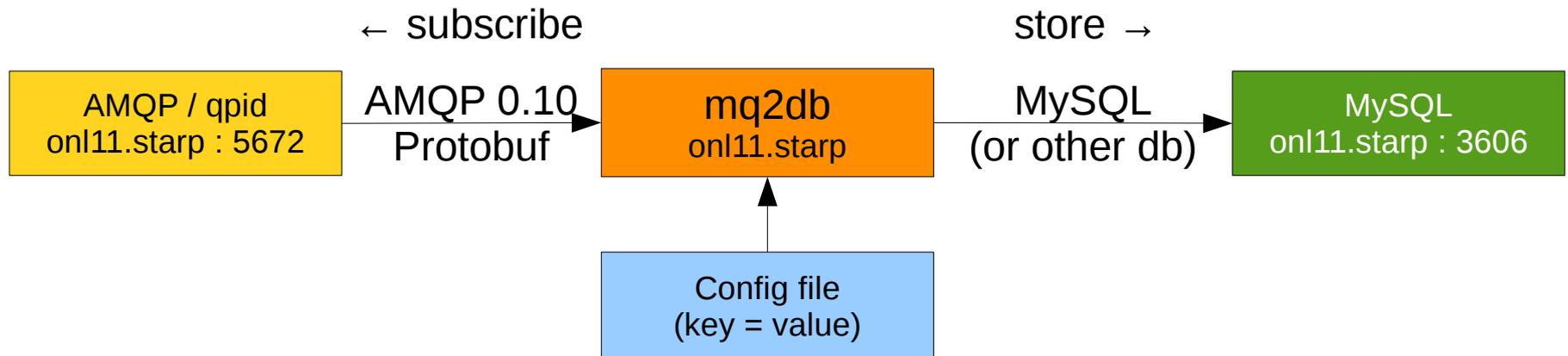
## Notes:

- there is no need to write a single line of code to add more channels to the data collector system, just run another epics2mq instance with separate config file listing desired epics channels!
- epics2mq code is simple, less chance to introduce bugs;

# mq2db

( publish/subscribe )

**mq2db** : daemon, which receives messages published by epics2mq, deserializes results (Protobuf), and stores data to backend database via abstract interface. All AMQP and database parameters are read from config file.



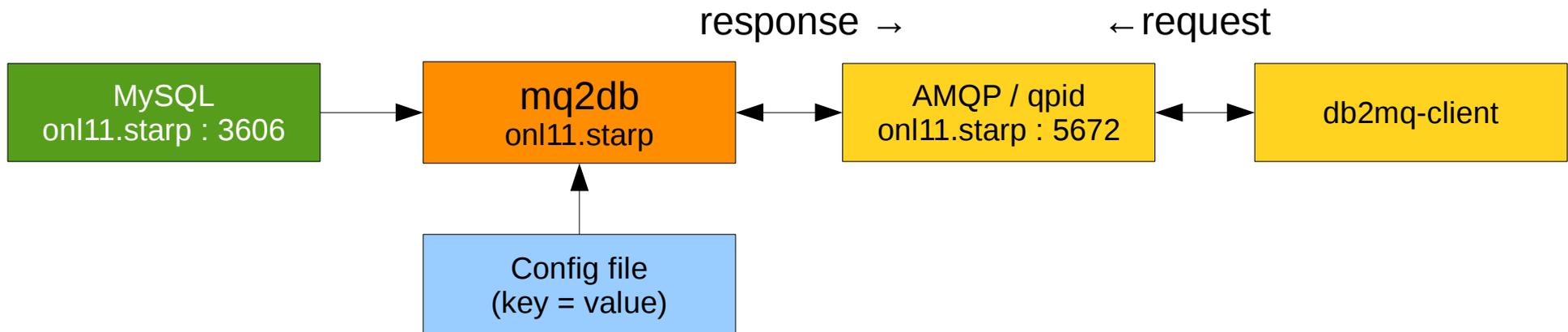
## Notes:

a) there's no need to modify mq2db daemon to store extra data channels: it automatically converts "path" and data into database / table / record form for all messages arriving over AMQP line (it will create new databases and tables automatically)!

b) you can have two or more completely independent, parallellized data archivers, using various databases as storage backends: just run another instance of mq2db in parallel to your primary archiver!

# db2mq ( request/response )

**db2mq** : daemon, which awaits for client data requests over AMQP, fetches data from database backend, sends data to client via AMQP;

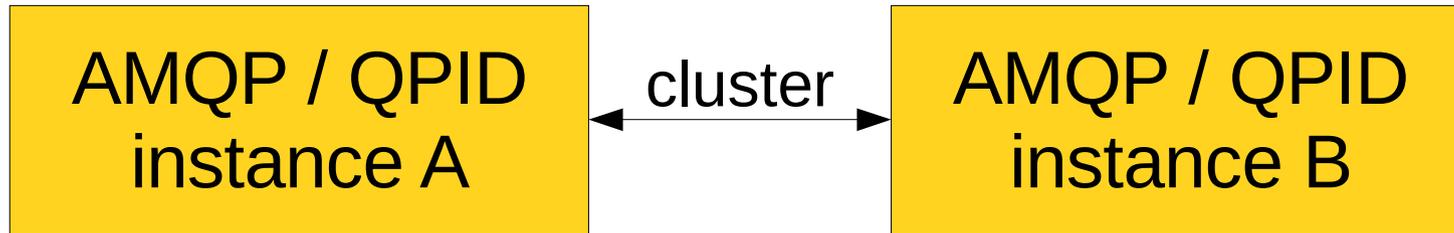


## Notes:

a) fairly standard request/response implementation, nothing special here..

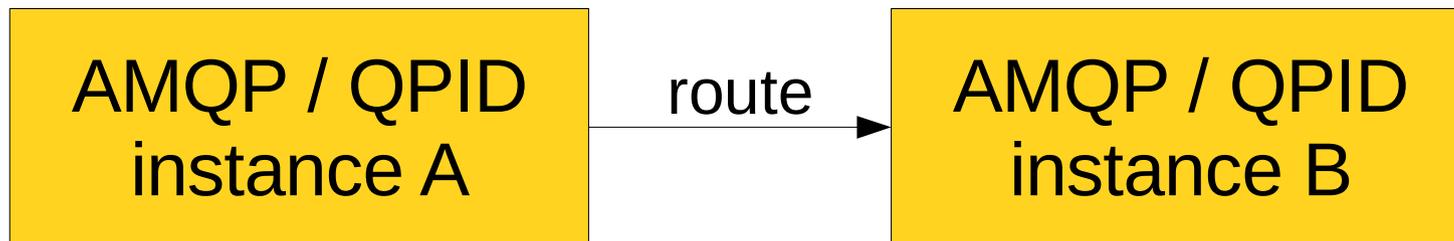
# AMQP server : clustering/failover

So, OK, AMQP could be a heart of the system, but what about Single Point of Failure?



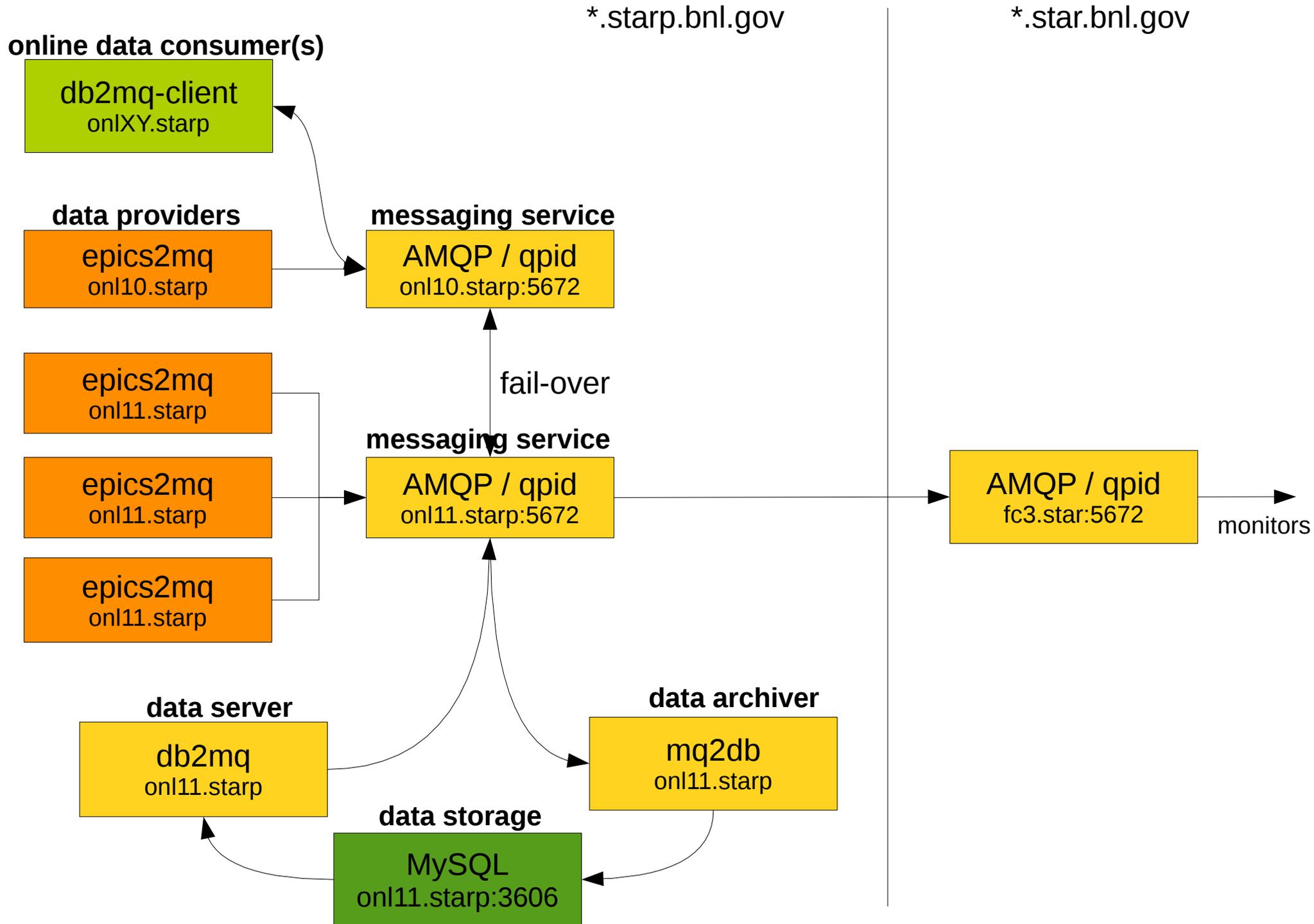
**qpid supports clustering through OpenAIS**

Well, I'd like to export only some of my queues to external facility, can I do that?

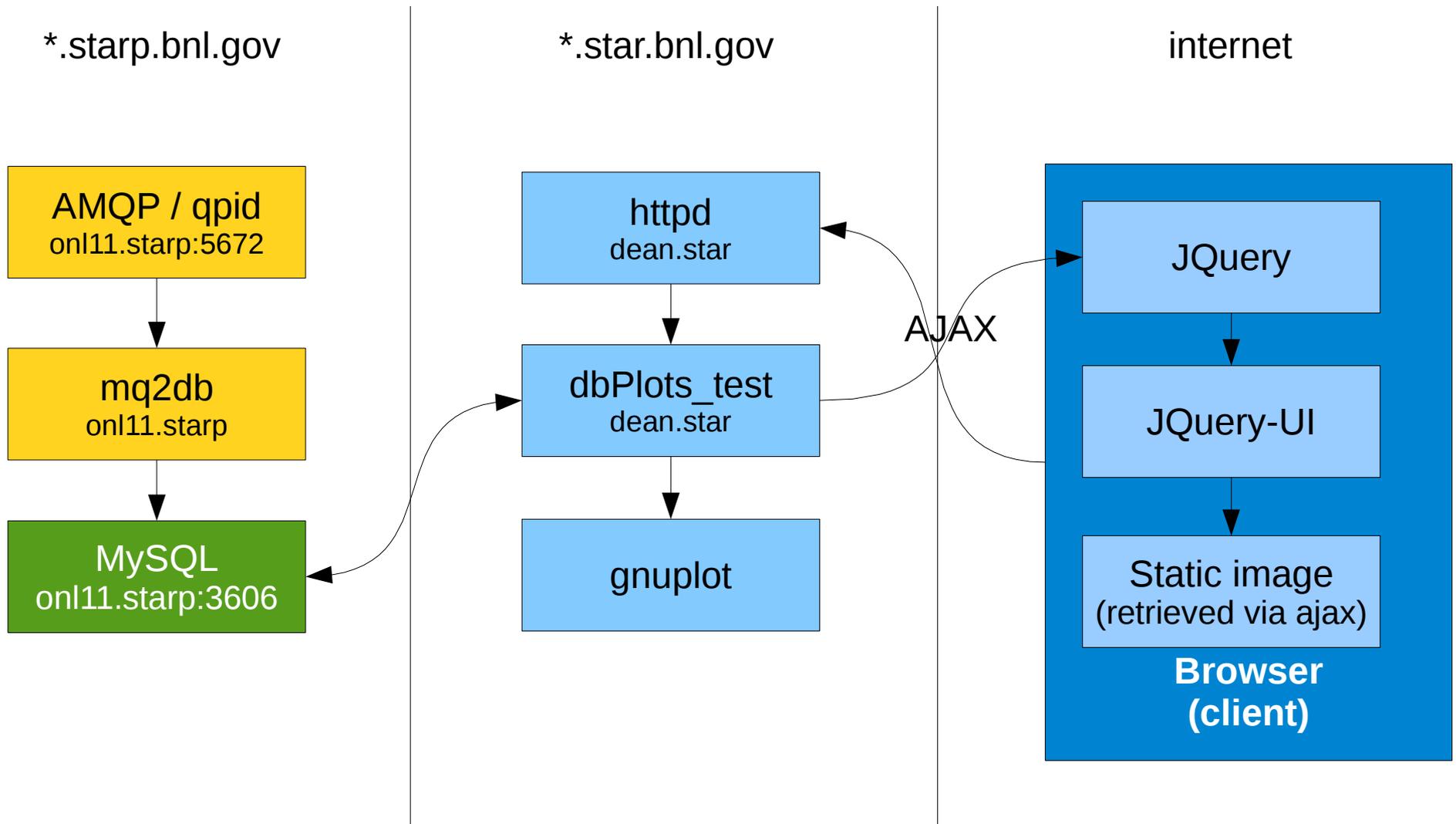


**qpid supports message routing to external qpid servers**

# Online data collection and monitoring : new approach

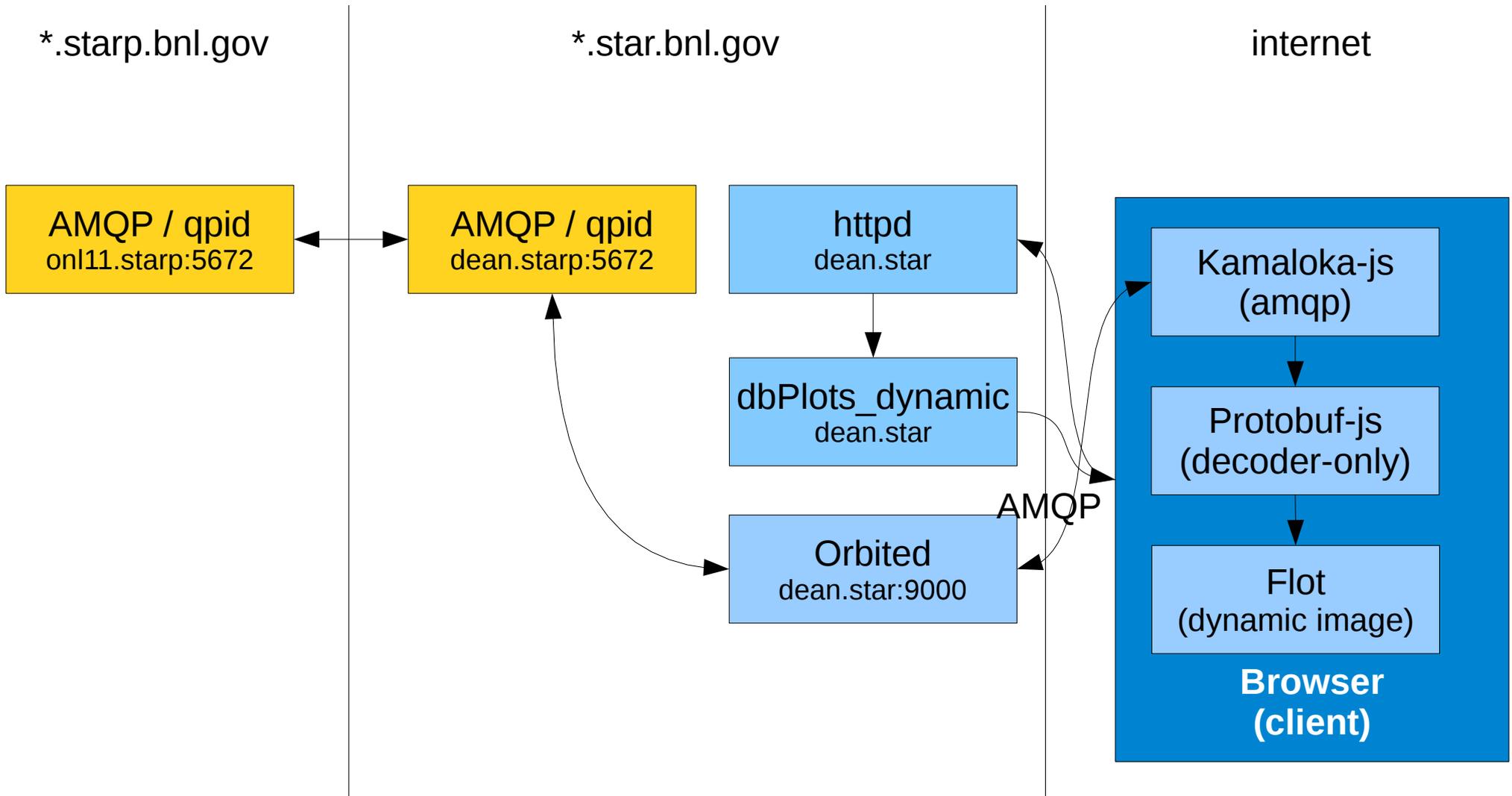


# Current **data monitoring**, static images



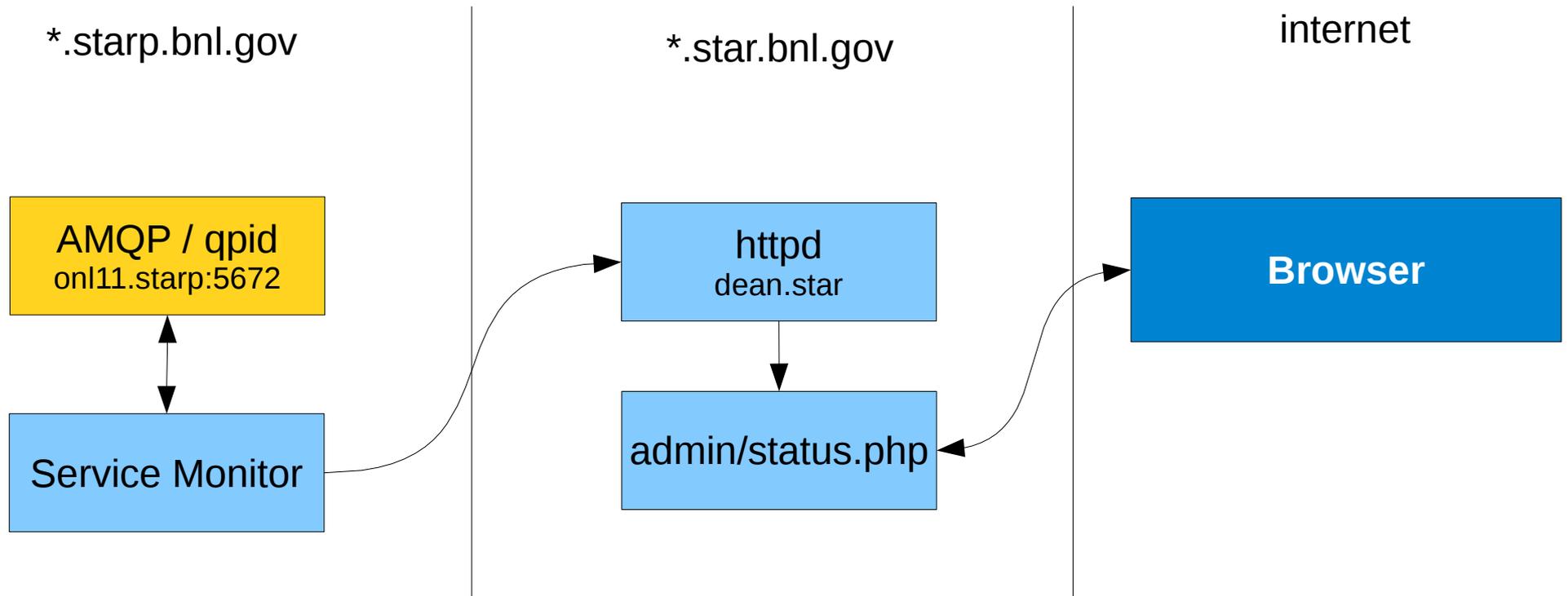
Basic monitoring, plot is updated by user request

# Enhanced **data monitoring**, dynamic images



Fully dynamic monitoring, plot updates as soon as message arrives via MQ

# Enhanced **service monitoring**, last update/status



Service monitor subscribes to ALL epics channels at once : gov.bnl.star.online.epics.#  
+ for a special heartbeat channel : gov.bnl.star.online.services.# to get data + status  
information! No need for complex and unreliable checks of service pids, log files and  
such..

# Installation and Maintenance

- Installation prerequisites:

- **qpid** is available as rpm from Redhat MRG:  
<http://ftp.redhat.com/pub/redhat/linux/enterprise/5Server/en/RHEMRG/SRPMS/>
- **protobuf** is available as rpm from Fedora (RHEL-test):  
<ftp://rpmfind.net/linux/fedora/updates/11/SRPMS/protobuf-2.2.0-2.fc11.src.rpm>
- **log4cxx** is available as rpm from Fedora (RHEL-test):  
<ftp://download.fedora.redhat.com/pub/fedora/linux/releases/11/Everything/source/>
- **EPICS** is available as rpm from PSI:  
<http://linux.web.psi.ch/dist/scientific/5/gfa/current/>

- Installation:

- **epics2mq, mq2db, db2mq** are in STAR CVS :  
[http://www.star.bnl.gov/cgi-bin/protected/cvsweb.cgi/online/DataBase/online\\_api/](http://www.star.bnl.gov/cgi-bin/protected/cvsweb.cgi/online/DataBase/online_api/)

- Documentation:

- <http://drupal.star.bnl.gov/STAR/comp/db/onlinedb/onlinemq>

# Performance test : basics

**4 “slow” epics2mq** services running for 12 days :

- 11.2k x 4 messages published;
- 0.4 mb of RAM used (stable over the whole period)

**1 “fast” epics2mq** service running for 9 days :

- 77.9k messages published;
- 0.5 mb of RAM used (stable over the whole period)

**1 mq2db** service running for 12 days :

- 81.7k messages stored to db;
- 0.4 mb of RAM used (stable over the whole period)

**Total data** transfer: 9.77MB in, 13.7MB out (12 days)

**MQ Broker info (qpid):**

- uptime : 101 day (mq service started well before the test);
- 44 queues created;
- 9 unique connections registered;
- 0.2 mb of RAM used to serve aforementioned clients (0.2 on average, 0.3 at peak);

**Crash test:**

[X] **epics2mq** was sending malformed data (crafted) to mq, **mq2db** rejected it successfully;

[X] **epics2mq** got “unstable” EPICS channels, correctly handled it;

**BASIC TEST: SUCCESSFUL!**

# Performance test: high load

Initial idea of stress testing was rejected because 100+ EPICS channels I tried to use last week belong to various STAR power supplies – not quite suitable for stress test :)

## What to expect in ~2-3 days :

Yuri created a bunch of **simulated EPICS channels** specifically for my testing yesterday :

- **80** channels generating values of “**double**” type;
- **20** channels generating values of “**string**” type;

Note: all aforementioned channels appear as real live channels to my client – no difference in access compared to real IOC access;

## Stress testing:

### I. Message publishing rates :

- 20 epics2mq instances, requesting 80 “double” channels every 1-5 seconds and publishing it to mq (groups by 1-10 channels per epics2mq service);
- 5 epics2mq instances, requesting 20 “string” channels every 1-30 seconds;
- memory and cpu usage monitoring;

### II. Data storage backend performance : mq2db (MySQL-based)

- local queue test (how many messages can it hold before going down);
- “data push” rate (X messages processed and stored to db in Y hours;
- peak memory and cpu usage monitoring;

**STRESS TESTING RESULTS: IN 2-3 DAYS**

# Summary & ToDo List

- Design of MQ-based system is complete;
- Prototypes of MQ-based system components are implemented in C++:  
[http://www.star.bnl.gov/cgi-bin/protected/cvsweb.cgi/online/DataBase/online\\_api/](http://www.star.bnl.gov/cgi-bin/protected/cvsweb.cgi/online/DataBase/online_api/)
- Test setup for “EPICS to MySQL” is deployed – complete Online Data Collection system for STAR (no data migration to Offline for now):  
[http://online.star.bnl.gov/dbPlots\\_test/](http://online.star.bnl.gov/dbPlots_test/)
- Real-time data flow monitoring is underway – need firewall exceptions installed (amqp routing from star to starp domains);
- Documentation, client examples & performance testing :  
<http://drupal.star.bnl.gov/STAR/comp/db/onlinedb/onlinemq>
- ToDo: RTS/DAQ is now looking into this setup : Jeff notified;
- ToDo: Field testing: Run 11, in parallel to “old” system; Event processing to be added during the upcoming Run.