RGMA as an example of a generic framework for information exchange

HPDC 2007 Workshop on Grid Monitoring, Monterey, USA, 25 June, 2007

Steve Fisher/RAL on behalf of JRA1-UK

www.eu-egee.org
www.glite.org

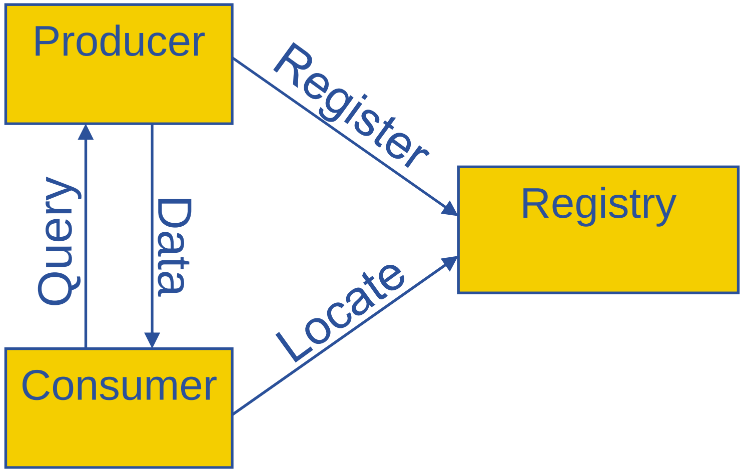
Information Society and Media

Overview

- GMA and R-GMA
- Generic framework – pros and cons
- Custom solution – pros and cons
- Fixing the problems …

GMA

- **Defined by the GGF**
  - Now OGF
- **3 Components**
  - Producer
  - Consumer
  - Registry

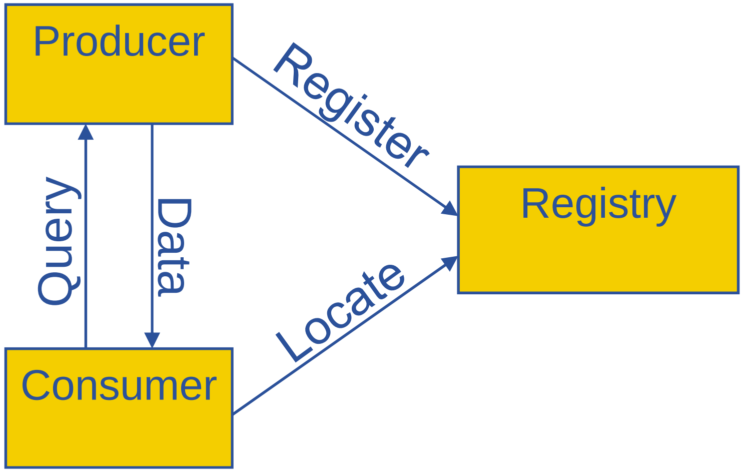


- **Real system needs to tie down message formats**
  - This has been done by R-GMA

- **The INFOD-WG at GGF**
  - IBM, Oracle and others have defined a GMA compliant specification

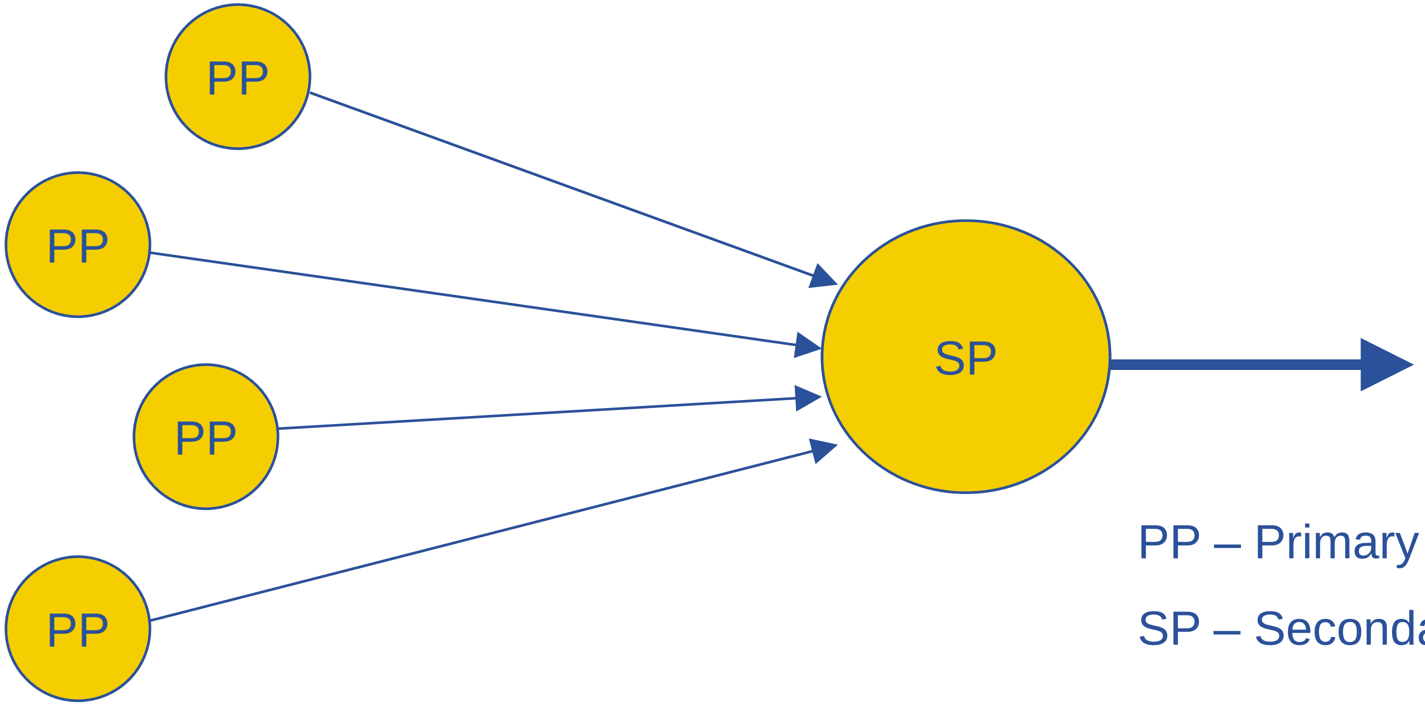
R-GMA

- **Relational implementation of the GGF’s GMA**
- **Provides a uniform method to publish and access both information and monitoring data**
- **Registry is hidden**
- **It is intended for use by:**
  - Other middleware components
  - End users
- **Easy for individuals to define, publish and retrieve data**
- **All data has a timestamp, enabling its use for monitoring**



R-GMA Producers

- **Primary – source of data**
- **Secondary – republish data**
  - Co-locate information to speed up queries
  - Reduce network traffic



PP – Primary Producer

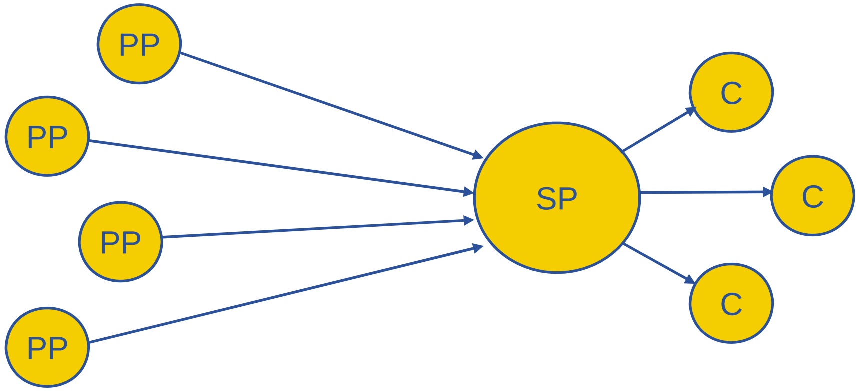
SP – Secondary Producer

R-GMA: Some Applications

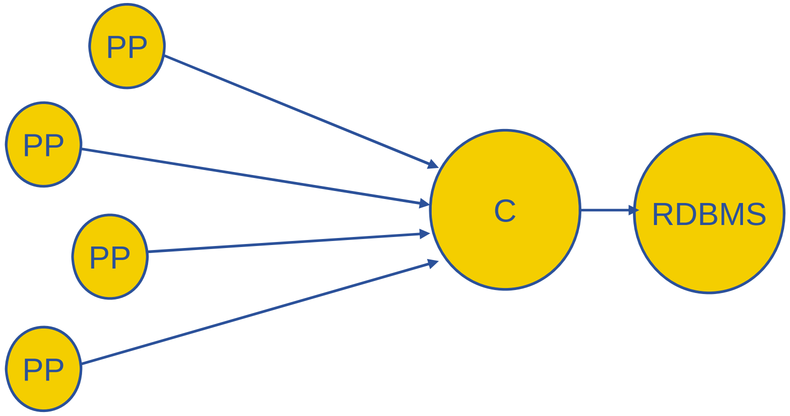
- gridFTP monitoring (GridView)
- Job monitoring (GridView)
- Network performance monitoring
- Intrusion detection
- Application monitoring
- APEL

R-GMA Patterns

- Secondary Producer



- Proprietary RDBMS



Benefits of the generic framework

- **R-GMA will have been configured for you at each site**
  - No need to worry about firewalls
- **Very easy to get started**
  - Define table(s)
  - Publish from job or infrastructure
    - No knowledge of consumers
  - Consume from job or infrastructure
    - No knowledge of producers
- **Can set up one or more secondary producers to hold latest or historical information**

Problems with the generic framework

- **You may not like the chosen data model**
  - Relational in the case of R-GMA
- **Applications can interfere with each other**
  - Registry can be overloaded by unsuitable code running on a large number of worker nodes

Benefits of custom solution

- **For any one application you can always do better with something you write yourself**
  - It may even be possible to do this quite quickly
  - Many applications have simple star topology
    - e.g. APEL sends data to GOC
    - Can replace knowledge of registry location by knowledge of central repository and just push data there

Problems with the custom solution

- Pushing data is easy to configure but what do you do when data are blocked by firewall?
- Your single central node may not be able to cope; how do you then partition the data?
- How do you combine results from custom solution A with custom solution B?
- Beware of the difference between a prototype and deployment on 200 sites

R-GMA has addressed all above

Fixing the problems …

- **Murphy’s law applies very well to distributed system**
  - One or more sites will be misconfigured or dead
- **Must avoid single points of failure**
  - Obvious
- **Must avoid evil interactions**
  - Less obvious

- **Ultimately it is a plumbing problem**
- **All blockages are a problem**

Avoiding single points of failure

- **Multiple Virtual Databases**
  - Separate registries for different communities
  - Rogue user will only affect his own community
- **Registry replication**
  - Registry failure is not then a problem

Avoiding evil interactions

- **Avoid queues of messages waiting to be sent**
  - Registry does not notify
- **Prefer pull to push**
  - Slaves poll master
- **Design system to not require responses to internal messages**

CHEP talk being prepared to go into many of the details

Summary

- **Generic solutions are hard to develop**
- **Generic distributed systems are very hard to develop**
- **Simple ad-hoc solutions seem attractive – but they must not be allowed to grow complex**
- **Simple solutions can turn out to be too restrictive**
  - Might solve by making it more complex
  - Admit that problem is harder than anticipated
- **If you feel that a custom solution is right for you then keep it very simple – and avoid feature creep**