

Chapter 8

Notifications



Communicating with a process

For a long complex job – you may need to know what point in its operation it has reached – and sometimes how it got there

For a job on a single machine a traditional way of obtaining the information is via a log file.

A log file is a set of text messages

Skypes log file from my machine

Date/Time: 2016-08-12 15:51:43 +0100
OS Version: Mac OS X 10.11.6 (Build 15G31)
Architecture: x86_64
Report Version: 19

Command: Skype
Path: /Applications/Skype.app/Contents/MacOS/Skype
Version: 7.31 (7.31.0.304)
Parent: launchd [1]
PID: 9880
Event: wakeups (microstackshots only)
Wakeups: 209 wakeups per second for 216

Communicating with a process(es) on remote machines

You can look at a log file at any time with little
processing or network overhead.

Some small overhead in terms of disk space and
processing.

What do we do if the machines are at the other end of
the network?

Monitor the progress of a job

You wish to know

Other processes in a common calculation need to
know where the job has reached.

There are essentially three possibilities.

Query-response

Broadcast

Publish-subscribe

Query-response

The requesting process interrogates the process.

What point have you reached?

Respond with the last significant point passed

Have you reached this point yet?

Respond yes or no

Problems

- Where is the process? Which data centre?
part of farm and it may not even have an
external IP address. Many jobs of interest.
- Process must have a sub-process which
listens for requests and responds.
- How often to “poll”
 - Wasted bandwidth and processing
power for answer No.
 - Gap between reaching the point and
the requesting process proceeding

Broadcast

Running process simply reports when it has reached a certain point.

Problems

- May mean a large amount of data on the network which is of no interest.
- Broadcasting unwanted data is a waste of resources in the issuing program.
- Any programme interested in the output has to be listening all the time.
- Allowing broadcasting (or connection to an arbitrary address) has security implications – possible to launch a DOS attack

Advantages

- Latest data always available

Publish-subscribe

A process (or person) who wishes to be informed about a particular event (or class of events) registers with the process producing the events.

When an event occurs the process sends out messages to all the clients which have registered to be informed of this event.

Problems

- DOS attack – still possible.
- More complicated

Advantages

- Latest data always available.
- No unnecessary data on the network
- The listeners only get interrupted when there is data of interest.

Registration itself can be achieved in a number of ways.

Stock-market

Lots of information ... much of interest to only a few people. Number of bits of information very large (number of stocks). Update rate potentially very large (changes on the millisecond scale potentially important).

Large data volumes for the network

Large data volumes for the client to sift through, much of it irrelevant.

Publish-subscribe the obvious solution.

Relational Grid Monitoring Architecture

Instead describe a system RGMA, design and implementation part of a project which Prof Hobson and I worked on a few years ago.

The problem was getting monitoring/logging information from potentially thousands (or tens of thousands) of jobs running in hundreds of computer centres round the world.

A client wishing to monitor the state of part of the job running on a CPU, would have no easy way to determine where the process was running.

Since the job was distributed, in order to get an accurate idea of its progress information had to be collated throughout the world.

Jobs run on massive farms of processors and for security reasons, these processors are not allowed to make calls out onto the wider network.

Analysis for
CERN
experiments
looking for the
Higgs

Notifications

Relational Grid Monitoring Architecture

The Global Grid Forum (**GGF**) model

Producers who provide information

Consumers who request information

Registry (single) mediates between the two

R-GMA implements this via a standard query language
a subset of **SQL**.

Producers publish data in rows (tuple) sql *insert*.

Consumers query using sql *select*.

All entries carry a **time stamp** for monitoring

It is not a database – it re-uses sql the RDBMS
language.

- No new language needs to be developed
- Users understand the language without training
- Data is presented as a database – a model which users are likely to understand

How does the user – or job the job doing the monitoring know where to find the running jobs?

How does a running job know where to send logging information?

Cluster/Farm

They don't

The cluster on which they are running has an interface which knows about the registry.

They talk to the interface and the interface forwards the requests to the registry.

Producer

Interface

Registry



Cluster/Farm

Consumer

Interface



Notifications

A producer will send a message to the registry saying that it will produce certain sorts of message.

It “creates” a table.

An consumer will tell the registry that it is interested in receiving entries from a particular “table”

The registry returns a list of suitable producers, the consumer then contacts the producer to negotiate the transfer of information.

When a new piece of information is available it is formatted as a sql insert command and goes via the interface of the producer direct to the interface of the consumer(s), where it is forwarded to a target machine.

Cluster/Farm

Producer

Interface



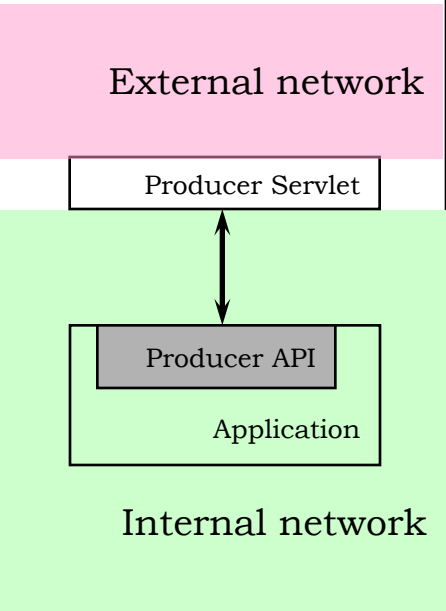
Cluster/Farm

Consumer

Interface



Notifications



Processor

Somewhere on the site a machine is running a producer servlet. It must be on the same network as the target processor

An API exists for a programme to make calls to the servlet.

The programme defines a schema for the information it wishes to publish via a

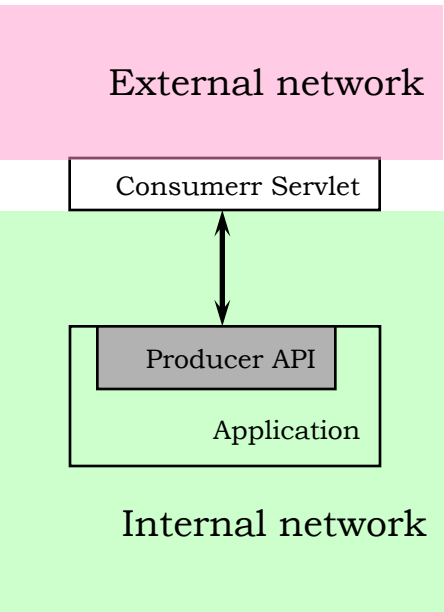
create table command

As it runs it outputs the monitoring data via a series of

Insert table ... commands

Programmer has no need to know about the details of R-GMA implementation, including location of registry etc.

Processor can communicate with the outside world with no (fewer) security issues. It has no direct access to the external network – and the producer servlet will talk to the registry and other servlets



Monitor

An application to monitor the performance of a job can run on any machine which can see a Consumer servlet.

Again the servlet hides from the application (monitor) the details of the R-GMA infrastructure

The monitor makes a request for information via a
select a,b,c from tablename.

Information flows in from the consumer servlet.

All producers insert into the same table.

All monitors (there can be more than one) select from the same table.

It is as if there is a single database. All implementation details are hidden by the servlets.

Servlets

The servlets need to know where the registry is that the should talk to.

This is set up by the sysadmin during the installation of R-GMA.

registry

External network

Consumer Servlet

So the user makes a call to “R-GMA” and connects to the servlet machine.

The machine involved is transparent to the user. Part of site configuration (both producer and server).

Servlets talk to correct registry.

User knows nothing about the underlying infrastructure.

When a job has finished (producer or consumer) it should contact the registry and ask to be removed.

Because failures are possible – either crashes or careless programming, there is a timeout. This is called soft state registration and helps to stop the registry becoming full up.

Handling of the timeouts and “keep alive” signals is handled by the infrastucture

registry

External network

Producer Servlet

Registry needs to keep list of logging and monitoring machines, so if a new producer comes on line it can be told where to send the messages

Notifications

15 User view

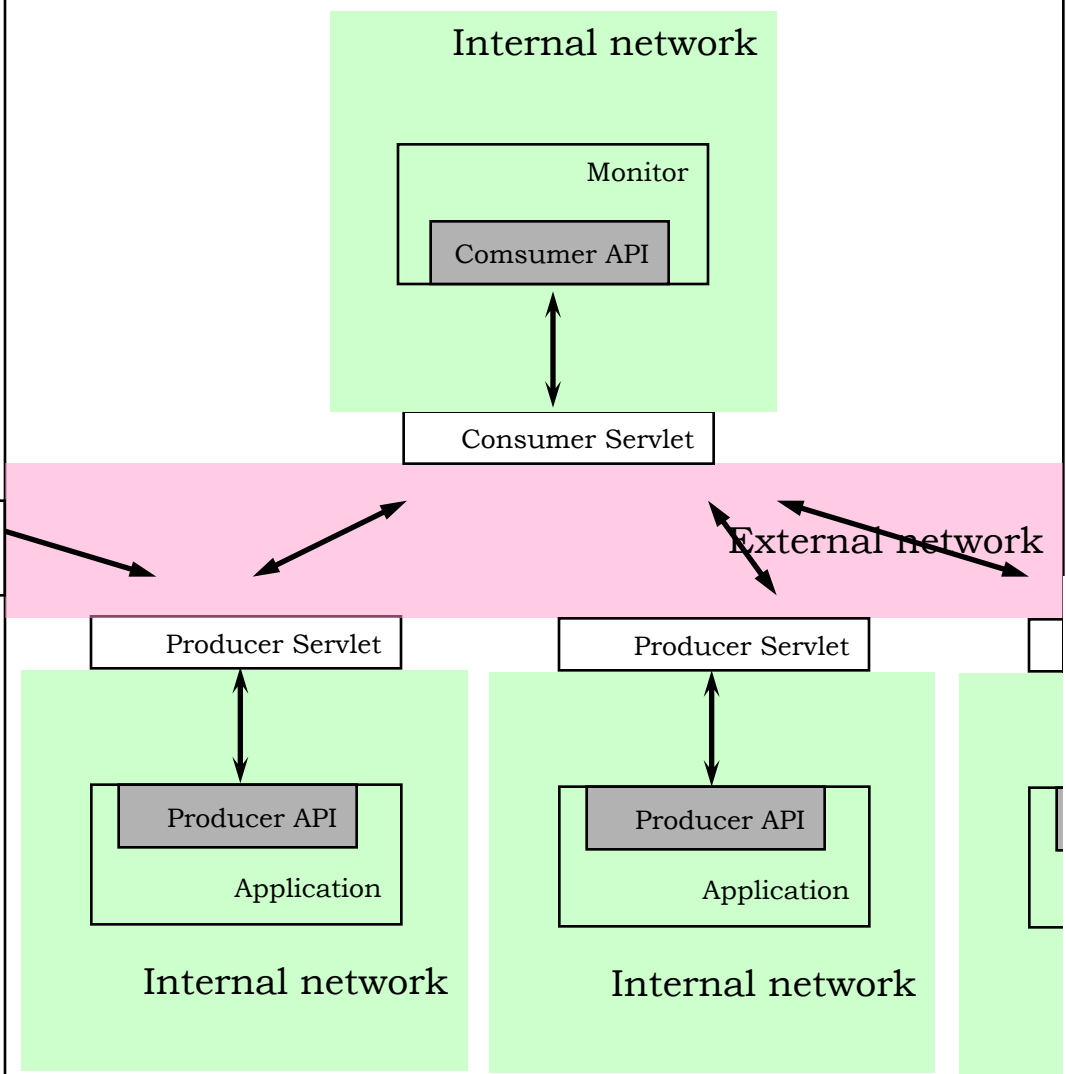
Producer can publish to more than one consumer

Consumer can consume from more than one source

registry

Servlet communication established via registry

Should make registry resilient. High quality machine, possible hot backup



Notifications

Summary

The knowledge of the infrastructure lies in the infrastructure, the users have no need to understand it.

They talk to the infrastructure via a well defined interface (API). Virtualisation.

The machines themselves need no access to the outside world – communication is done through a well defined interface.

The interfaces talk to each other and the registry, they have no need to access any other sites and this means that they can be made much more secure