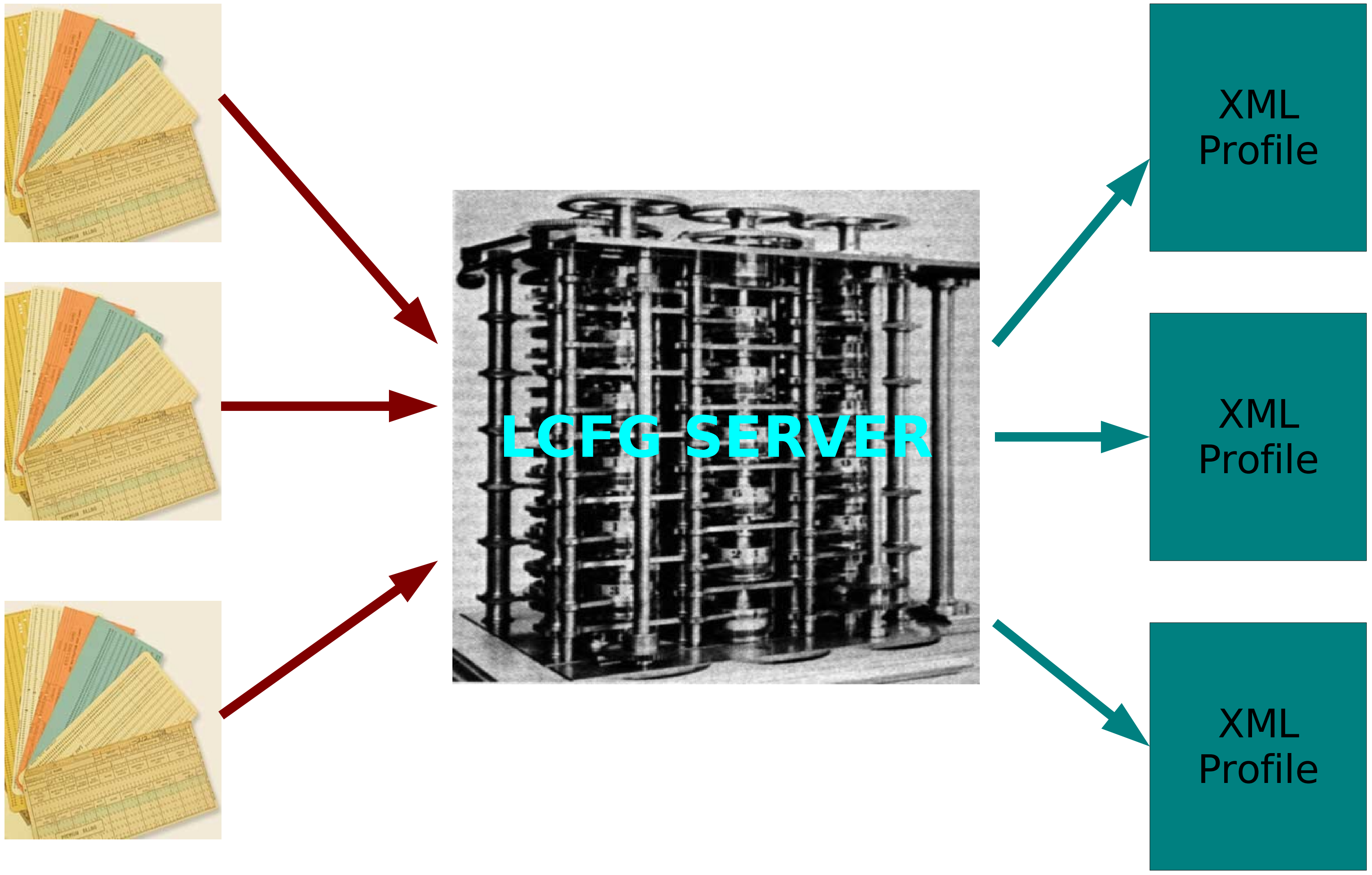
System Configuration with LCFG

Paul Anderson <dcspaul@inf.ed.ac.uk>
Stephen Quinney <squinney@inf.ed.ac.uk>

How it Works



The Profile

- Describes the required state of a client machine.
- Consists of a set of *components* and, optionally, a list of packages.
- Built by the LCFG server.
- Shipped to the client as an XML file.

A Component

- Consists of:
  - Set of *resources* – basically key/value pairs.
  - Optional *templates* for config files
  - Optional control code, to manage daemons, for example, stop/start/configure, based on a framework provided by LCFG – shell or perl.

Example 1a: MOTD

- Manage a single file on a single machine

```
#include <local/site.h>
#include <lcfg/os/minimal.h>

!file.files         mADD(motd)
file.file_motd      /etc/motd
file.tmpl_motd      Welcome to <%profile.node%>.<%profile.domain%>
file.type_motd      literal
```

Template variables

School of informatics

Example 1b: MOTD

- Configuration re-use for multiple machines

```
#include <local/site.h>
#include <lcfg/os/minimal.h>
#include <local/motd.h>
```

Just put it into a file that can be included

Example 1c: MOTD

- Configuration re-use and modification

```
/* local/staff-computer.h */

#include <local/motd.h>

!file.tmpl_motd mCONCAT(This is for staff usage only)
```

```
/* source profile */

#include <local/site.h>
#include <lcfg/os/minimal.h>
#include <local/staff-computer.h>
```

Minimal Platform

- Able to manage contents of individual files
- Might be able to manage daemons
- Problems:
  - We do not control the entire state.
  - Multiple system admin approaches will almost certainly end up with conflicts.

Managed Platform

- The aim is to describe the characteristics of the machine in the source profile:
  - Operating System
  - Hardware type
  - Staff machine? In a public lab?
  - Available for condor pool?
  - Special software required?

Example 2: Authorization

```
/* local/computer.h */

!auth.users  mSET(@sysmans)
```

```
/* local/staff-computer.h */

#include <local/computer.h>

!auth.users mADD(@staff)
```

```
/* local/lab-computer.h */

#include <local/computer.h>

!auth.users mADD(@staff)
!auth.users mADD(@students)
```

Example 2: Authorization

```
/* source profile */

#include <local/managed-site.h>
#include <lcfg/os/fc6.h>
#include <local/staff-computer.h>

/* allow a non-staff user access */

!auth.users mADD(fred)
```

Example 3: Managing a Server

```
/* local/web-server.h */

!tcpwrappers.allow   mADD(apache)
tcpwrappers.allow_apache httpd : 192.168.

!ipfilter.export    mADD(http)
```

```
/* local/rsync-server.h */

!tcpwrappers.allow   mADD(rsyncd)
tcpwrappers.allow_rsyncd rsyncd : 192.168.1.1

!ipfilter.export     mADD(rsync)
```

Example 3: Managing a Server

```
/* source profile */

#include <local/managed-site.h>
#include <lcfg/os/fc6.h>
#include <local/web-server.h>
#include <local/rsync-server.h>
```

Spanning Maps

- A component in one profile can *publish* resources to which a component in the profile for another machine *subscribes.*
- Usage includes:
  - dhcp
  - ipfilter
  - inventory

Conclusion

- Usability – common config language
- Can describe required state completely.
- Devolved management.
- Easy to manage relationships within the network.
- Autonomics – machine configures itself.

Further Information

- http://www.lcfg.org/
- lcfg-discuss@inf.ed.ac.uk