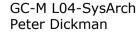
System Architecture

A Distributed Systems Perspective

A brief survey of the topic, exploring some architectural ideas and principles of value in Distributed and Grid computing

Emphasising Thought Experiments as a way of gaining insight





Hardware Perspectives - basics

- Internal machine architectures
 - Single machines now becoming multi-core, network-on-a-chip approach
 - Super-computers: e.g. vector processors, SIMD & MIMD multiprocessors
- Desk area networks
 - Connecting peripherals and computational elements
- Small-scale Local area networks
 - Classics: Ethernet, token rings etc
- Composing networks
 - Using bridges and routers: consider reach of multicast/broadcast
- Metropolitan networks
 - Access points connect sites to a regional network facility
 - Multiple institutions: need to provide safeguards e.g. firewalls
 - Physical security and staff screening no longer enough

Hardware perspectives - autonomy

- Multi-site single organisation networks
 - Use VPN and similar technologies
 - Communicate across the network core from one boundary point to another
 - Can encrypt/decrypt at organisation boundaries
- Multi-site multi-organisation networks
 - Negotiation at the boundary points
 - Problem:
 - Large organisations don't collaborate in their entirety
 - Sections within them collaborate for specific projects
- Internet-scale networking
 - Millions of organisations
 - Ever-changing dance of collaborative activity
- Focus today:
 - architectural issues within a single project in a single administration

Thought Experiment 1 - Hardware

- Layout of a bungalow:
 - Entrance/hallway
 - Main living room
 - Kitchen
 - Dining Room
 - Bathroom
 - Two Bedrooms
 - Extensive gardens

- Make it fit for purpose:
 - Owner, enjoys cooking
 - Frequent dinner parties

• Problem: designing for the future

What if the purpose changes?

Software Architecture

- Architecture Description Languages
 - Components, connectors
 - syntax for capturing the design, usually graphical
- Architectural Styles (Shaw & Garlan)
 - Dataflow systems
 - Batch sequential; pipes & filters
 - Call-and-return Systems
 - Program & subroutine; OO; Hierarchical layers
 - Independent Components
 - Communicating processes; event systems
 - Virtual Machines
 - Interpreters; rule-based systems
 - Data-centred Systems
 - Databases; hypertext systems; blackboards

Architectural Heterogeneity

- Fitness for purpose is the key...
 - Real systems are hybrids
- Layering in the network
- Independent Components protected by firewalls as organisations
- Application systems built from Cooperating components, with reconfigurable connectivity, as a layer over the OS/File System
- Shared data repository (internally distributed with replication)
- Single component is actually a composite, often in a different design style, so the overall effect is somewhat hierarchical

Example Architectural Approaches

- Client-Server
 - Pure
 - exclusive-or
 - Semi-pure
 - exclusive-or at given layer, servers at layer N+1 are clients at layer N
 - General
 - Client-server perspective for a single interaction only
- Three-tier systems
- Multi-tier systems
- Information Dissemination alternatives:
 - Publish-Subscribe
 - Event-based approaches
- Implications of Push vs Pull and information hiding

Thought Experiment 2 – LHC data

- Large Hadron Collider
 - CERN device, multiple experiments in caverns around the ring
 - Interests physicists at CERN and around the world
 - Vast data output
- Sketch the data repositories and data flows

Performance and Flexibility

- Any given design will offer a certain level of performance
 - Depending on the features of its components parts
- A design will also have some degree of flexibility/adaptability
 - Ease of change may vary considerably
- System evolution is one aspect of long-term maintenance
- Change/evolution at run-time, using dual systems, or off-line
- Distributed applications are built for purpose
 - If short-lived purpose, unlikely to be intended to be flexible
 - If built from components, reconfigurability may be almost free
 - If long-lived, some element of future-proofing included, e.g.:
 - Clean interfaces to allow component replacement and interceptors/adapters
 - High-level specs to allow re-use

Building Distributed Systems – Guidance 1

- Maintain separation of concerns
 - Even if it seems expensive
- Keep it clean
- Keep it simple
- Decide on some design principles early
 - Prototype, evaluate and discard
- Security cannot be retrofitted
- When performance matters, ensure you keep the performant parts central and simple
- Think carefully about synchronisation, consistency, global knowledge etc

Building Distributed Systems – Guidance 2

- Think carefully about the overall architecture of your system
- Component placement, data flows, predicted loads etc
- Build the system out of tested components
- Honour components interfaces and avoid stressing them
- Use your desired behaviour (functional spec) to specify the tests, and then really do conduct the tests
- Hope you don't have to seriously debug it

Thought experiment 3 – a file system

- Design a distributed file system
- What features do you want it to have?
- Consider basic technologies you will use
- Consider data placement, cacheing, consistency etc