SCALE and heterogeneity

Grid Computing (M)

Lecture 15



Section Outline

- What is scale? Why should I care?
- Four principal scale dimensions of interest:
  - Number of nodes
  - Size of distributed database
  - Size of objects
  - Communication delays
- General approaches to issues of scale
  - Move functionality into “leaves”
  - Partition problem into domains and introduce gateways
  - caching
  - Asymptotic convergence
  - Smooth out control response
  - Paradigm shift
- Case studies
  - DNS
  - FAST TCP
  - Delay-tolerant networking

Copyright © 2006 University of Glasgow

Wikipedia on scale ...

- **Scale** in the computing field is used as a verb.
- An algorithm, design, networking protocol, program, or other system is said to **scale** if it is suitably efficient and practical when applied to large situations (e.g. a large input data set or large number of participating nodes in the case of a distributed system). For example, the distributed nature of the Domain Name System allows it to work efficiently even when all hosts on the worldwide Internet are served, so it is said to "scale well".
- If the design fails when the quantity increases then it **does not scale**. For example, some early P2P applications, such as Gnutella, had scaling issues as each node broadcast its requests to all peers. This means that the demand on each peer would in the worst case increase in proportion with the total number of peers, potentially quickly overrunning the peers' limited capacity. Other P2P systems like BitTorrent scale well because demand on each peer is independent of the total number of peers.

Copyright © 2006 University of Glasgow

Move functionality into "leaves"

- Distributed applications are usually organized into hierarchical structures, in which an entity at each level "controls" some number of lower-level entities
- If higher-level entities must be involved in interactions between lower-level entities, the scalability of the application is drastically affected, as these higher-level entities represent concentration points for multiple lower-level interactions
- Moving the responsibility/functionality into lower-levels improves the ability for the system to scale
- e.g. delegation of naming authority to subdomains in DNS

Copyright © 2006 University of Glasgow

Partition into domains and interact via gateways

- Each algorithm is characterized by a particular complexity in terms of messages transmitted, bandwidth consumed, etc.
- As such, there is usually some maximal scale to which the algorithm can be applied before these overheads negatively affect the chosen application
- If the problem to be addressed exceeds this maximal scale, a possible solution is to break the problem into domains that do NOT exceed the maximal size, and to construct some other structure for interactions between the domains.
- e.g. flooding routing protocols, like OSPF or IS-IS, permit the establishment of sub-regions in which the flooding protocols are used, but use a non-flooding regime for interactions between the sub-regions

Copyright © 2006 University of Glasgow

Cacheing

- Many types of interactions in distributed applications exhibit locality of reference – i.e. if a particular question has been asked once, it is highly likely that it will be asked again in the near future
- Caches are the mechanisms we use for enhancing the performance of applications that exhibit such locality of reference
- Of course, there is no free lunch; we have simply moved the complexity into cache coherency issues
- As long as the time scale over which the data (upon which the caches are based) changes are significantly shorter than the time scale over which repeat queries will be posed, the cache hit rate will justify the cache coherency overhead
- e.g. the many and varied caches used for WWW content

Copyright © 2006 University of Glasgow

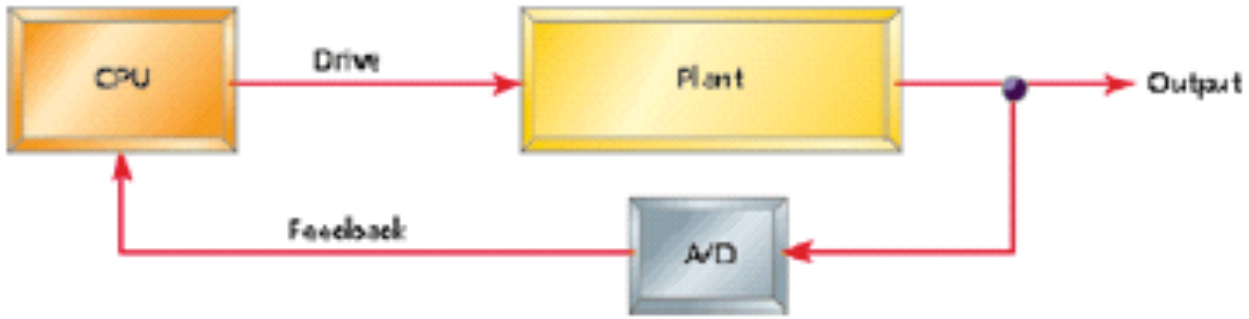
Asymptotic convergence

- 2-phase commit protocols for distributed database management exhibits n^2 complexity
- This has extremely poor scalability for large distributed databases
- Is it essential that a distributed database conform to ACID properties?
- As with caching, if the time scale over which the data changes is long relative to the time scale over which queries are made, the data just has to be asymptotically convergent/consistent
- e.g. DNS's zone update protocol

Copyright © 2006 University of Glasgow

Smooth out control response

- Many distributed applications involve response to external events
- Often times, these responses are part of a closed control loop, in which the system is attempting to maintain operation within a certain performance regime.
- The algorithm that interprets the feedback must avoid making rapid changes to the system, as these may lead the system to instability; some form of hysteresis or other smoothing function must be applied
- e.g. equation-based flow control of TCP Vegas and FAST TCP



Copyright © 2006 University of Glasgow

Paradigm shift

- We are used to changing the complexity of an algorithm by looking at the problem in a completely different way
  - e.g. Heapsort vs Selection sort; paradigm shift involves doing comparisons and exhanges over much larger distances on the dataset, thus yielding $n\log_2 n$ behaviour instead of n^2
- Sometimes, scale issues simply require a complete rethink of how to address the problem, as existing solutions at smaller scales simply do not work
- e.g. delay-tolerant networking

Copyright © 2006 University of Glasgow

Case Study 1

Domain Name System

Copyright © 2006 University of Glasgow

DNS Data Model[1]

- DNS is a global, loosely consistent, delegated database
  - delegated → contents are under local control
  - loosely consistent → shared information (within constraints)
    - does not need to match or be up-to-date
    - operation is global with owners of "names" responsible for serviing up their own data
  - data on the wire is binary
  - domain names are case insensitive
  - hostnames are restricted to a small character set (although IDN provides standard encoding for names in non-US ASCII

Copyright © 2006 University of Glasgow

[1]Much of this material taken from the DNS Tutorial at IETF-63 by Ólafur Gudmundsson and Peter Koch.

DNS Terms

- Domain name: any name represented in the DNS format
  - a.b.example
- DNS label:
  - each string between two "." unless the dot is prefixed by \
  - i.e. foo.bar is two labels, while foo\.bar is one label
- DNS zone:
  - a set of names that are under the same authority
  - e.g. example.com and ftp.example.com
  - zones can be deepter than one label – e.g. gla.ac.uk
- Delegation:
  - Transfer of authority for a domain
    - gla.ac.uk is a delegation from ac.uk
    - the former is typically termed the child and the latter is typically termed the parent

Copyright © 2006 University of Glasgow

More DNS Terms

- RR: a single Resource Record
- RRset: all RR's of the same type at a name
  - this constitutes the minimum transmission unit for the protocol
- TTL: the time a RRset can be cached/reused by non-authoritative servers

Copyright © 2006 University of Glasgow

DNS Elements

- Resolver
  - stub: simple, only asks questions
  - recursive: takes simple query and takes all necessary steps to obtain the full answer
- Server
  - authoritative: the servers that contain the zone file for a zone, one Primary and one or more Secondaries
  - caching: a recursive resolver that stores prior results and reuses them
  - some perform both roles at the same time

Copyright © 2006 University of Glasgow

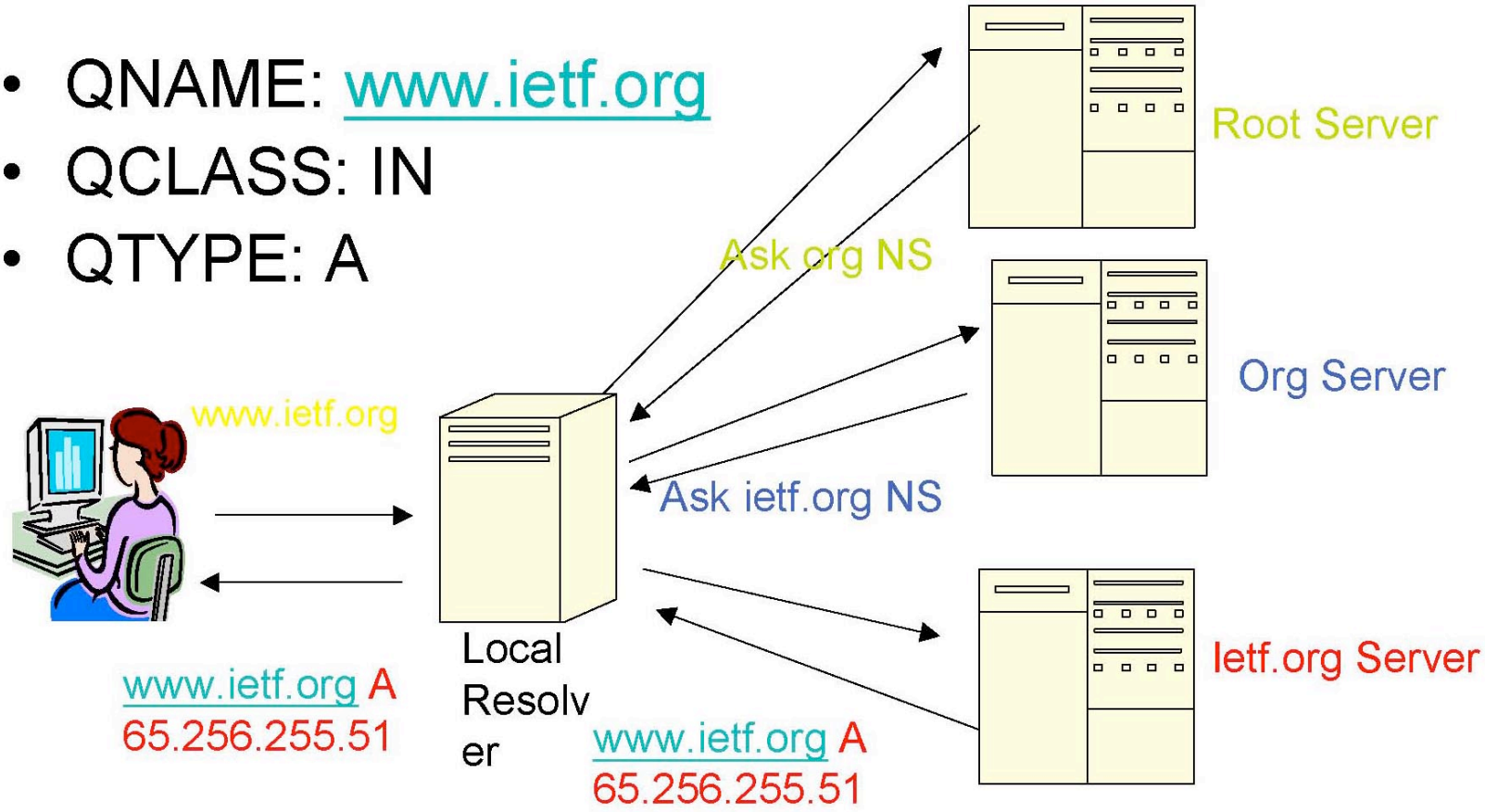
DNS Communication Protocol

- Normal request response uses a UDP-based datagram protocol with retransmissions
- Retry timer is configurable, typically 4 or 8 seconds
- Often, retries are extremely persistent
- Use a transaction ID field to disambiguate responses
- Key point: Application using DNS is typically decoupled from the DNS resolver making recursive queries
- Zone transfers use TCP (bulk data, rather than RPC-style communication)

Copyright © 2006 University of Glasgow

DNS query

- QNAME: www.ietf.org
- QCLASS: IN
- QTYPE: A



Copyright © 200

DNS Operation

- DNS is a “lookup service”
  - simple queries → simple answers
  - no search!
  - no “best fit” answers
  - limited data expansion capability
- DNS zone is loaded on authoritative servers
  - servers keep in sync using zone update protocol – asymptotic consistency
- DNS caches only store data for a “short” time
  - defined by TTL on RRset
- a DNS resolver starts at the longest match for the QName in its cache, and follows delegations until an answer or negative answer is received
  - DNS packets are small
  - DNS transactions are fast if servers are reachable

Copyright © 2006 University of Glasgow

How well does it scale[2]?

- The DNS has, with a few minor tweaks, worked well as the Internet has grown over SIX orders of magnitude.
- DNS scales because of good NS-record caching, which partitions the database
  - alleviates load on root/TLD servers
- Hierarchy in the names is NOT the reason for DNS scalability
  - the namespace is essentially flat in practice
- A-record (mapping domain name to IP address) caching is, to first order, a non-contributor to scalability
- The zone update protocol enables the system to perform well without having to suffer through ACID consistency overheads

Copyright © 2006 University of Glasgow

[2]Some of this material taken from the presentation entitled "The Internet Domain Name System" by H. Balakrishnan.

What have we learned about scale?

- Reasons for success of DNS
  - Simple – queries consist of name, class, and type
  - Clean – explicit transfer of authority
    - parent is authoritative for existence of delegation
    - child is authoritative for contents
  - Lightweight – UDP used to transport queries and responses
  - Relaxed consistency semantics via zone update protocol
- Scalability bottom line
  - most important aspect contributing to scalability of DNS is the partitioning of the database by delegation
  - caching of NS records is critical for reducing traffic to root and TLD servers, and as a result, the communication load on the WAN
  - its relaxed consistency semantics enables multiple authoritative servers to respond to queries in parallel

Copyright © 2006 University of Glasgow