Why and how to Scale?

why
- Server systems must be able to start small
  - Small-size company (garage-scale) v.s. international company (kingdom-scale)
- System should be able to grow as demand grows e.g.
  - eCommerce made system growth more rapid & dynamic
  - ASP also need dynamic growth

how
- Scale up - expanding a system by incrementally adding more devices to an existing node – CPUs, discs, NICS, etc.
  - inherently limited
- Scale Out – expanding the system by adding more nodes – convenient (computing capacity can be purchased incrementally), no theoretical scalability limit

Farm/Geoplex

Farm - the collection of servers, applications and data at a particular site
- features:
  - functionally specialized services (email, WWW, directory, database, etc.)
  - administered as a unit (common staff, management policies, facilities, networking)
- Geoplex – a replicated (duplicated?) farm at two or more sites
  - disaster protection
  - may be
    - active-active: all farms carry some of the load;
    - active-passive: one or more are hot-standbys (waiting for fail-over of corresponding active farms)
Clone

- A replica of a server or a service
- Allows load balancing
- External to the clones
  - IP sprayer (like Cisco LocalDirector™) dispatches (sprays) requests to different nodes in the clone to achieve load-balancing
- Internal to the clones
  - IP sieve like Network Load Balancing in Windows 2000
  - Every request arrives at every node in the clone, each node intelligently accepts a part of these requests;
  - Distributed coordination among nodes

RACS

- RACS (Reliable Array of Cloned Services) – collection of clones for a particular service
- Two types
  - Shared-nothing RACS – each node duplicates all the storage locally
  - Shared-disk RACS – all the nodes (clones) share a common storage manager. Stateless servers at different nodes access a common backend storage server

RACS advantages

- Scalable – good way to add processing power, network bandwidth, and storage bandwidth to a farm;
- Available
  - Nodes can act as backup for one another: one node fail, other nodes continue to offer service (probably with degraded performance)
  - Failures could be masked if node- and application- failure detection mechanisms are integrated with the load-balancing system or with client applications
- Easy to manage – administrative operations on one service instance at one node could be replicated to all others.

Problems with RACS

- Shared-nothing RACS
  - Not a good way to grow storage capacity; updates at one node’s must be applied to all other nodes’ storage
  - Problematic for write-intensive services: all clones must perform all writes (no throughput improvement) and need subtle coordination
- Shared-disk RACS
  - Storage server should be fault-tolerant for availability (only one copy of data)
  - Still require subtle algorithms to manage updates (such as cache validation, lock managers, transaction logs, etc.)

Partitions and Packs

- Partition – service is grown by dividing data among nodes
  - Only one copy of data in each partition – availability is not improved
- Pack – each partition is implemented by a set of servers
  - Shared disk
  - Shared nothing
    - Active/active – all members service partition
    - Active/passive – one member services partitions, the others - standby

How to partition, RAPS

- Typically, the application middleware partitions the data and workload by object:
  - Mail servers partition by mailboxes
  - Sales systems partition by customer accounts or product lines
- Challenges
  - When a partition (node) is added, the data should be automatically repartitioned among the nodes to balance the storage and computational load.
  - The partitioning should automatically adapt as new data is added and as the load changes.
- RAPS (Reliable Array of Partitioned Services) – a collection of nodes that support a packed-partitioned service
  - Provides both scalability and availability;
  - Better performance than RACS for write-intensive services.

Figure 2: Two clone design styles: shared-nothing clones and shared-disk clones. Shared-nothing is simpler to implement but scales I/O bandwidth as the site grows. But for large or update-intensive databases a shared-disk design is more economical.

Figure 3: Partitions and Packs: Data objects (mailboxes, database records, business objects,..) are partitioned among storage and server nodes. For availability, the storage elements may be served by a pack of servers.
RACS vs. RAPS

- Clones and RACS
  - for read-mostly applications with low consistency and modest storage requirements (<= 100 GB)
  - Web/file/security/directory servers
- Partitions and RAPS
  - For update-intensive and large database applications (routing requests to specific partitions)
    - Email/instant messaging/enterprise resource planning (ERP)/record keeping

Multi-tier application example

- Functional separation
  - front-tier: web and firewall services (read mostly)
  - middle-tier: file servers (read mostly)
  - data-tier: SQL (database) servers (update intensive)

Summary

- Scalability technique
  - Replicate a service at many nodes
- Simpler forms of replication
  - Duplicate both programs and data: RACS
- For large databases or update-intensive services
  - Data partitioned: RAPS
  - Packs make partitions highly available
- Against disaster
  - The entire farm is replicated to form a geoplex

Figure 5: A scaled website showing cloned front ends doing web and firewall service, then shared-disk cloned file servers and packed and partitioned SQL servers.