Cluster-Based Scalable Network Services

Armando Fox, Steven D. Gribble, Yatin Chawathe, Eric A. Brewer and Paul Gauthier

Presented by Hari Sivaramakrishnan

Advantages of Clusters

- **Scalability**
  - Linear increase in hardware to handle load
  - Adding resources easy for clusters

- **Availability**
  - 24 x 7 service, despite transient hardware or software errors
  - Nodes are independent in a cluster. Failures masked by software

- **Cost Effectiveness**
  - Economical to maintain and expand
  - Commodity hardware
Challenges to using Clusters

- Administration
  - Software available

- Component vs System Replication
  - Can support part of a service, not all of it
  - Handled in the architecture design
    - Functions are well described, and interchangable

- Partial Failures

- Shared State
  - None in a cluster
  - Can be emulated, but performance can be improved if need for shared state is minimized

Architectural Features

- Exploits strengths of cluster computing

- Separation of content from services

- Programming model based on composition of worker models

- BASE semantics
  - Basically Available, Soft State, Eventual Consistency

- Measurements and monitoring
Architecture of a SNS

Layered Architecture

Service: Service-specific code
- Workers that present human interface to what TACC modules do, including device-specific presentation
- User interface to control the service

TACC: Transformation, Aggregation, Caching, Customization
- API for composition of stateless data transformation and content aggregation modules
- Uniform caching of original, post-aggregation and post-transformation data
- Transparent access to Customization database

SNS: Scalable Network Service support
- Incremental and absolute scalability
- Worker load balancing and overflow management
- Front-end availability, fault tolerance mechanisms
- System monitoring and logging

Figure 1: Architecture of a generic SNS. Components include front ends (FE), a pool of workers (W) some of which may be caches (S), a user profile database, a graphical monitor, and a fault-tolerant load manager, whose functionality logically extends into the manager stubs (MS) and worker stubs (WS).

Figure 2: Scalable Network Service Layered Model
SNS Layer

- **Scalability**
  - Use incrementally added nodes to spawn new components
  - Workers are simple and stateless

- **Centralized load balancing**
  - Policy implemented in manager, can be changed easily
  - Trace information collected from workers, decisions sent to FEs
  - Fault tolerant

- **Prolonged Bursts, Incremental growth**
  - Overflow pool
  - Workers spawned by manager

- **API**
  - Provided by manager and FE to allow for new services
  - Worker stub handles load balancing, fault tolerance etc.
  - Worker code focuses on service implementation

TACC : Programming model

- **Transformation**
  - Operation on a single data object
  - Example: encryption, encoding, compression

- **Aggregation**
  - Collating data from various objects

- **Customization**
  - User specific data automatically fed to workers
  - Same worker can be used with different parameter sets

- **Caching**
  - ISPs observed 40 – 50 % savings...critical
  - Can cache original and transformed data
TansSend

- Front Ends
  - SPARCstation machine cluster
  - HTTP interface
  - Request served from cache if available or computed
  - 400 threads

- Load balancer
  - MS contacts manager to locate a distiller
  - WS accepts requests and reports load info
  - Manager spawns distiller if load increases

TansSend contd.

- Fault Tolerance
  - Registration system used to locate distillers
  - Timeouts detect dead nodes
  - All state is soft
  - Watcher process needs to know if peer is alive by periodic monitoring
  - Peers start one another
    - Manager starts FE
    - FE starts a manager
    - Manager reports distiller failures to MS which updates its cache
  - Programmed in the manager stubs
TransSend contd.

- User profile database
  - Normal ACID database

- Caching
  - Harvest object cache workers

- Distillers
  - Image processing
  - Off the shelf code
  - Did not have to remove all the bugs because if a node crashes, it will be restarted by a peer

- Graphical Monitor
  - Detect system state and resource usage

TransSend’s use of BASE

- Load balancing data
  - MS don’t have most recent information
  - Errors are corrected by using timeouts
  - Perf improvements outweigh problems

- Soft state
  - Transformed content is cached

- Approximate answers
  - If system is overloaded, can return a slightly different version of data from cache
  - User can get accurate answer by resubmitting a request
Input Characteristics

- Average cache hit takes 27ms to serve
- 95% of hits take less than 100ms
- Miss penalty anywhere from 100ms to 100s

- Cache perf related to number of users and size
  - Hit rate increases monotonically with size
  - When sum of users exceeds cache size, hit rate falls
Load balancing

- Metric – queue length at distillers
- New distillers spawned when load is very high
- Delay D to allow for new distillers to stabilize the system before adding more distillers

![Graph of load balancing](image)

Scalability

- Limited by shared or centralized components – SAN, manager, user profile DB
- DB
  - Was never near saturation in their tests
- Manager
  - Has capability to handle three orders of magnitude more traffic than the peak load
  - Even commodity hardware can get the job done

![Graph of scalability](image)
Scalability of SAN

- Close to saturation, unreliable multicast traffic dropped
  - This information is needed by manager to load balance

- Workarounds
  - Separate network for data and control traffic
  - High performance interconnect

Economic Feasibility

- Caching saves an ISP a lot of money

- A server can pay for itself in 2 months

- Administration costs not considered
  - Do not expect it to be very significant
Conclusion

- Architecture works around deficiencies of using clusters

- Defined a new programming model which makes adding new services extremely easy

- BASE (weaker than ACID) semantics enhances performance