OnCall: Defeating Spikes with Dynamic Application Clusters

Keith Coleman and James Norris
Stanford University

June 3, 2003

Problem & Motivation

- Today’s static clusters can’t respond to flash crowds
- Adding new machines to handle spikes is slow and difficult
- Serving all spikes quickly requires huge excess capacity => Expensive
- Excess capacity can’t be shared between applications
The OnCall Solution

• **Hypothesis:**
  A dynamic application cluster can handle spikes more effectively than a static cluster (and at lower cost, to boot)
  - Dynamic application cluster based on VMs for generality and legacy support
    - Current version is application generic
  - Cluster runs many applications, sharing capacity among them

Platform: Overview

Physical Machines running VMware (plus Daemons and Applications)

Scheduler Node

L7 Load Balancers

Network-Attached Storage
# Goals

- Allow same or better performance as same size static cluster
- Allow higher max capacity per app (to handle spikes)
- Maintain performability isolation
  - Spiking app shouldn’t hurt others

Key Innovation: Scheduling

- VM Scheduling
  - Don’t want thrashing with 4 GB VMs (hard to avoid)
  - Solution: Double-Hysteresis
    - Based on OS concepts in ESX Server
      - But clusters are different than OSes => forced innovations
    - Keep 3 performance averages
      - Slow, Fast averages + Future (linear) prediction
      - Pick max of these 3
    - Market-based Idle-tax approach
      - Tax idle resources 75% => not proportional (gives running apps benefit of the doubt)
      - “Shares” allow prioritization (and business)
  - Plus, guaranteed computers
    - At least $g_{-min}$ computers at all time
    - Up to $g_{-max}$ computers guaranteed if needed
    - Beyond that, up to the market
Measuring App Load

• For scheduling to work, need a good measurement of app load

• 3 Options:
  – OS statistics. Base on disk/ram/cpu measurements from OS
    • Definitely inaccurate
    • Pipeline problem
  – End-to-End. Measure end-to-end response rate against typical rate
    • Doesn’t demonstrate usage on cluster node (could be on backend DB)
    • Very application specific (and OnCall is app generic)
  – Meter. Time standard benchmark on each host
    • Maybe not entirely accurate
    • But best because measuring resource usage but not reliant on OS reporting

Copying vs. Preboot

• Originally were going to test both in production system, but copying overhead in prototypes was too high (~ 3-4 mins)

• Preboot or copy-on-demand much better
The End

Questions?

EXTRA SLIDES

Extra Slides
From
Computer Forum Poster Session
**Platform: Scheduler**

- *Scheduler* activates VMs/apps to handle traffic
- *Scheduler* runs as “just another application” in the cluster
  - Improves fault tolerance
- Gathers resource usage data from node *Daemons*

**Platform: Daemon**

- One *Daemon* per physical machine
- Launches and retires VM/apps
- Monitors resource usage for reporting to *Scheduler*
- *Daemons* and *Scheduler* run on a lease-based system
  - If contact with *Scheduler* is lost (e.g. leases not renewed), *Scheduler* kills self, *Daemons* start new instance of scheduler
Platform: “Collective” Base

• *OnCall* system based on Stanford’s “Collective” project
• Collective provides primitives for treating VMs as first class objects
• Key features include:
  – *Demand Paging* – block-grain transfer of VM images => instant startup
  – *Prefetching* – sends blocks ahead of time, based on experienced demand

Spike Response Strategies

• **Copy VM/Application On-Demand**
  Full VM/application images copied to new physical boxes, replacing other apps running on those boxes, in direct response to traffic changes
  **Upside:** consumes no extra resources
  **Downside:** potentially slow

• **Pre-boot VMs**
  VMs are pre-booted and simply signaled when traffic loads demanded their use.
  **Upside:** fast
  **Downside:** consumes resources

• **Prepare based on historic data**
  When a spike begins, resource manager begins copying and/or pre-booting based on historic spike data
  **Upside:** may increase response speed
  **Downside:** potential complexity increase
Platform: Network

- NAT and virtual routers used to hide physical network topology from VMs
- Also provides:
  - Isolation between applications (they are on separate virtual networks)
  - Support for legacy applications

Metrics: Spike Handling

- To test hypothesis, will measure:
  - Maximum Single Application Throughput (as compared with static cluster)
  - Spike Response Delay
  - Inter-application Performability Isolation
**Metrics: Specific Measures**

- The above general metrics will be obtained through low-level measurements:
  - End-to-End Throughput (as measured by application-level requests)
  - Request Latency
  - Dropped Requests

**Metrics: Test Scenarios**

- *Base Setup:* Cluster shared between two or more applications

- *Simulation:* Load spike on one application
- *Simulation:* Time-varying application demand
- *Simulation:* Induced node failures during spike