Scalability Evaluation of an Energy-Aware Resource Management System for Clusters of Web Servers

2015-07-27
SPECTS15

Simon Kiertscher, Bettina Schnor
University of Potsdam
Before we start ...
Outline

• Motivation
• Energy Saving Daemon (CHERUB)
• Scalability: Measurements
• Scalability: Simulation (ClusterSim)
• Conclusion & Future Work
Cluster Computing Basics

- **High-Performance-Computing (HPC)**
  - Few computationally intensive jobs which run for a long time (e.g. climate simulations, weather forecasting)

- **Web Server / Server-Load-Balancing (SLB)**
  - Thousands of small requests
  - Facebook as example:
    - 18,000 new comments per second
    - > 500 million user upload 100 million photos per day
Components of a SLB Cluster

- Internet
- VIP [public]
- Gateway/Dispatcher
- Switch
- Server 1
- Server 2
- Server n
Outline

• Motivation
• Energy Saving Daemon (CHERUB)
• Scalability: Measurements
• Scalability: Simulation (ClusterSim)
• Conclusion & Future Work
Motivation

- Energy has become a **critical resource** in cluster designs
- Demand of energy is still permanently rising
- Strategies for saving energy:
  1. **Switch off unused resources**
  2. Virtualization
  3. Effective cooling
     (e.g. build your cluster in north Sweden like Facebook did)
Motivation

- 30% of servers world-wide are comatose
- Corresponds to 4GW
  The most power full nuclear power plant block on earth generates 1.5GW
Outline

• Motivation
• Energy Saving Daemon (CHERUB)
• Scalability: Measurements
• Scalability: Simulation (ClusterSim)
• Conclusion & Future Work
Cherubs functionality

- Centralized approach - no clients on back-ends
- Daemon located at master node polls the system in fixed time intervals to analyze its state
  - Status of every node
  - Load situation
- Depending on the state and saved attributes and the load prediction, actions are performed for every node
- **Online system** - we don’t need any information about future load
- Cherub Publications: [3,4]
Outline

• Motivation
• Energy Saving Daemon (CHERUB)
• Scalability: Measurements
• Scalability: Simulation (ClusterSim)
• Conclusion & Future Work
Scalability: Measurements

• Test with 2 back-ends are not sufficient
• Aim: prove scalability up to 100+ nodes in terms of performance and strategy
• Methodology:
  • Measure key functions
  • Simulation
Key Functions

Key functions are either:

- Invocation rate depends on number of nodes
- Runtime depends directly on number of nodes

Two different types of key functions:

- State changing functions
- Information gathering functions
State Changing Functions

- Boot/Shutdown/Register/Sign Off
- All very equal in structure and invocation rate

![Graph showing time in seconds vs. number of nodes for different state changing functions]
Information Gathering Functions

• Status function:
  determines status of every node

• Load function:
  determines the load of the system
Information Gathering Functions

• **Status function:**
  determines status of every node

• **Load function:**
  determines the load of the system
Prototype:
Sequentially for every node:

• Query RMS for every node if registered
  Yes: Node is Online or Busy (load dependent)
  No: Test if physically on (via ping, http req., etc.)
    • Reachable: Node is Offline
    • Not reachable (1 sec timeout): Node is Down

• Worst Case $\rightarrow$ all $N$-nodes Down
  $\rightarrow T_{statusfun}(N) = N$ sec
2 different approaches:

- Simple: Prototype function for all nodes in a separate thread
- Complex: Non-blocking sockets and RMS query done for all nodes at once
Status Function - Results
Information Gathering Functions

• Status function: determines status of every node
• Load function: determines the load of the system
Load Function

Prototype:

• Every node is checked if the load forecast (2 minutes history) violates the overload threshold

→ Linear regression computation for each node is far to expansive

→ Drawback: No knowledge of the overall demand
Load Function

Re-Implementation:

• Checks load of the whole system
• Computes linear regression only once

→ Benefit: knowledge about how many nodes must be booted

→ Drawback: we now rely on a good schedule
Load Function - Results

The graph illustrates the performance of different load functions over the number of nodes. The x-axis represents the number of nodes, while the y-axis shows the time in seconds on a logarithmic scale.

- **Per node version (prototype)**: This line represents the performance of the prototype version of the load function, showing a significant increase in time as the number of nodes grows.
- **Global version (re-implementation)**: This line depicts the performance of the re-implemented global version, which maintains a relatively flat curve, indicating a more efficient handling of the load as the number of nodes increases.

The graph highlights how the re-implementation of the load function achieves better scalability compared to the prototype version.
Outline

• Motivation
• Energy Saving Daemon (CHERUB)
• Scalability: Measurements
• Scalability: Simulation (ClusterSim)
• Conclusion & Future Work
Simulation - Normal Setup

Internet

Cluster-system

Commands:
- Boot
- Shutdown
- Sign_off
- Register

Info: Node Status / Load

Cherub

Cherub_config.py: What nodes to manage, etc

Ipvsadm: Registered or not?

user
Simulation - Simulation Setup

- **ClusterSim**
  - Info: Node Status / Load
  - Commands: Boot, Shutdown, Sign_off, Register

- **Cherub**
  - Cherub_config.py: What nodes to manage, etc

- **User**
  - Http Logfile
Simulation - ClusterSim Architecture

Virtual Cluster

- VNode
- VNode
- VNode

Load-balancer

Request Library

Trace

State Server

Cherub
ClusterSim - Limitations

• No reimplementation of the *Completely Fair Scheduler*

• No typical discrete event driven simulation  
  → Bulk arrivals and Backlog Queue (BLQ) checks

• No modeling of system noise

• No concurrent resource access
ClusterSim - Validation - Metrics of Interest

• Service Level Agreement (SLA) in % violated if a 5 sec timeout is hit

• Median duration in ms of all successfully served requests
ClusterSim - Validation - Bordercase

Measurement details:
• 1 node, 4 cores, 4 workers, BLQ 20
• 10 minutes steady load of 4 req/sec
• Border case scenarios:
  • Low load (req duration 0.8 msec)
  • Overload (req duration 3.6 sec)
## ClusterSim - Validation - Bordercase Results

<table>
<thead>
<tr>
<th>Request type</th>
<th>Metric</th>
<th>REAL</th>
<th>SIM</th>
</tr>
</thead>
<tbody>
<tr>
<td>low load</td>
<td>SLA</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td></td>
<td>median duration</td>
<td>0.92 msec</td>
<td>0.802 msec</td>
</tr>
<tr>
<td>overload</td>
<td>SLA</td>
<td>0.166%</td>
<td>0.166%</td>
</tr>
<tr>
<td></td>
<td>median duration</td>
<td>3.582 sec</td>
<td>3.578 sec</td>
</tr>
</tbody>
</table>
Measurement details:

- 1 node, 4 cores
- 4/8 workers
- BLQ 20/40/60/80
- 10 minutes steady load of 4/8/12/16/20 req/sec
- Req duration 0.36 sec
8 Workers

SLA in %

Req/Sec

BLQ

REAL

SIM

0 100

20 40

60 80

100

4

8

12

16

20

20 40 60 80
First Results

• Cherub + ClusterSim with 100 vnodes configured
• 30 minutes Trace with load peak
• 180 sec boottime
• Initial number of started nodes 10/50
• Results:
  95.6% / 99.45% SLA
  20.8% / 13.8% energy savings
• 42.5% theoretical optimum
100 Nodes Simulation With 50 Initial Started
Outline

• Motivation
• Energy Saving Daemon (CHERUB)
• Scalability: Measurements
• Scalability: Simulation (ClusterSim)
• Conclusion & Future Work
Conclusion & Future Work

• All key functions are fast enough to handle bigger clusters, proved with measurements
• ClusterSim mimics our real setup in a convincing way, proved with a border case study
• CHERUB scales up to 100+ nodes

• Deeper investigations on CHERUB + ClusterSim situations, tuning CHERUB parameters!
Thank you for your attention!
Any Questions?

Contact:
kiertscher@cs.uni-potsdam.de
www.cs.uni-potsdam.de
Sources

[1] “New data supports finding that 30 percent of servers are ‘Comatose’, indicating that nearly a third of capital in enterprise data centers is wasted” by Jonathan Koomey and Jon Taylor, 2015

