Adaptive Resource Allocation for Clouds under Bursty Workloads

J. Tai, J. Zhang, W. Meleis and N. Mi

Electrical and Computer Engineering Dept.
Northeastern University
{jtai, jzhang, meleis, ningfang}@ece.neu.edu
Cloud Computing

**Two-level load balancing**

- **Level 1**: balance the load across a set of instances of the same application
- **Level 2**: balance the load of multiple applications among physical computers
Existing Problems (1)

- **Burstiness in computer systems**
  - Dramatically degrade the performance

- **Burstiness in Clouds**
  - Multi-remote-users
  - Not single-program-single-execution
  - Application variety increases
Existing Problems (2)

- Information query delay

  - Network delay
  - Inter-server communication delay
  - System load self-update delay
Simulation Environment

- Burst/Non-burst
- Round Robin
- Random
- Qlen, Est. QT, Act. QT...
- Submission Delay

Diagram:
- User Demands
- Decentralized Load Balancer
- Host RMS
- Job Queue
- CPU Pool
- Computing Site 0
- Site 1
- Site n
Arrival Traces

- Non-Bursty: Even distribution of arrivals over time.
- Weak Bursty: Slight bursts of arrivals.
- Strong Bursty: Significant bursts of arrivals.

All traces have the same mean arrival rate.
### Burstiness & Delay Effect

<table>
<thead>
<tr>
<th>Workload</th>
<th>Rand</th>
<th>Qlen</th>
<th>Est. QT</th>
<th>Act. QT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Non-bursty</td>
<td>80.5</td>
<td>7.6</td>
<td>7.6</td>
<td>7.4</td>
</tr>
<tr>
<td>Weak bursty</td>
<td>168.5</td>
<td>466.5</td>
<td>466.5</td>
<td>466.2</td>
</tr>
<tr>
<td>Strong bursty</td>
<td>1520.9</td>
<td>6541.5</td>
<td>6540.8</td>
<td>6541.0</td>
</tr>
</tbody>
</table>

We need a new load balancer to counteract the impact of burstiness and submission delay.
Project Goal

- Adaptive resource allocation for clouds
  - Forecasts changes in user demands and system loads
  - Develop effective two-level load balancers

- Our expectations
  - Allow cloud users to experience higher quality of service
  - Allow cloud systems to make better use of their infrastructure
Our Initial Step: A new load balancer

- Balance the load **within an application**

![Diagram showing load balancing](image)

Arriving tasks $\lambda$

Front-end Dispatcher

$\text{uniform}(1, k)$

Back-end Nodes

$N = 6$ sites

10 7 13 2 5 10
Our Initial Step: A new load balancer

- Balance the load within an application
- **Online adjust** $k$ candidates

Index of dispersion based prediction

$$I = SCV \left( 1 + 2 \sum_{k=1}^{\infty} \rho_k \right)$$

![Graph showing Arrival Rate vs. Time (x100K) seconds]
Our Initial Step: A new load balancer

- Balance the load within an application
- Online adjust $k$ candidates
  - Idle phase: small $k$
  - Busy phase: large $k$
Some Preliminary Results

Response times

Policy

Greedy
New_Pred
New_Opt
Random
Next Steps …

Global views of user demands and system loads
Coherency and dependency of arrival request
Diversity of applications and client behaviors
Failure and security issues
Implementation in real systems (scalability)
Thanks

Q & A