Towards Predictable + Resilient Multi-Tenant Data Centers

Presenter: Ali Musa Iftikhar (Tufts University)

in joint collaboration with:

Fahad Dogar (Tufts), {Ihsan Qazi, Zartash Uzmi, Saad Ismail, Gohar Irfan} (LUMS),
Mohsin Ali (USC), Ruwaifa Anwar (SUNY SB)
Multi-Tenant Data Centers

• Flexible pay as you go model is attractive to tenants
• Meets variability in tenant demands
Multi-Tenant Data Centers

• Flexible pay as you go model is attractive to tenants

• Meets variability in tenant demands

• Yet, there are challenges to deal with
Why is Predictability Important?

• Data center is a shared resource

• Leads to high variability in the network
  
  • Potentially results in tenant’s cost variability
Why is Predictability Important?

• Data center is a shared resource

• Leads to high variability in the network
  • Potentially results in tenant’s cost variability

Thus we need to provide some sort of Predictability
Virtual Abstractions for Predictable Performance

Virtual abstractions:

- Expose a virtual network to the tenants
- Tenants can then demand for guaranteed bandwidth

Examples of such abstractions include:

{Oktopus, FairCloud, CloudMirror} (Sigcomm ’11 ’12 ’14), Hadrian (NSDI ’13)
Virtual Abstractions for Predictable Performance

Virtual abstractions:

• Expose a virtual network to the tenants

• Tenants can then demand for guaranteed bandwidth

Examples of such abstractions include:

{Oktopus, FairCloud, CloudMirror} (Sigcomm ’11 ’12 ’14), Hadrian (NSDI ’13)

But, they tend to ignore a crucial factor!
A stark Reality – Failures!

Datacenter **Network Failures** are common:

- Studies have shown: *(Understanding network failures in data centers, Sigcomm '11)*
  - 30% of the components show **less than** four 9s of availability
  - Time between successive failures could be as short as **5 minutes**
  - Time for recovery could even go beyond **1 week**
- These failures result in significant service **downtimes** hurting the tenants!
A stark Reality – Failures!

Datacenter **Network Failures** are common:

- Studies have shown: *(Understanding network failures in data centers, Sigcomm ‘11)*
  - 30% of the components show **less than** four 9s of availability
  - Time between successive failures could be as short as **5 minutes**
  - Time for recovery could even go beyond **1 week**

- These failures result in significant service **downtimes** hurting the tenants!

Thus we need to provide **Reliability** + **Predictability**
### “Predictability + Resilience”: Requirements

<table>
<thead>
<tr>
<th>Goal</th>
<th>Requirement(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Predictability</td>
<td>Bandwidth Reservation</td>
</tr>
</tbody>
</table>
“Predictability + Resilience”: Requirements

<table>
<thead>
<tr>
<th>Goal</th>
<th>Requirement(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Predictability</td>
<td>Bandwidth Reservation</td>
</tr>
</tbody>
</table>
| Resilience    | • Firstly: Provide **Backup Resources** to enable recovery  
|               | • Secondly: Ensure **speedy recovery** *(Aspen Trees CoNEXT ‘13, F10 NSDI ‘13)* |
“Predictability + Resilience”: Requirements

<table>
<thead>
<tr>
<th>Goal</th>
<th>Requirement(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Predictability</td>
<td>Bandwidth Reservation</td>
</tr>
<tr>
<td>Resilience</td>
<td>• Firstly: Provide <strong>Backup Resources</strong> to enable recovery</td>
</tr>
<tr>
<td></td>
<td>• Secondly: Ensure speedy recovery <em>(Aspen Trees CoNEXT ‘13, F10 NSDI ‘13)</em></td>
</tr>
</tbody>
</table>
Providing Backup Resources for Resilience

One approach:

• **Reserve Backup Bandwidth** to tolerate failures along with tenant reservations

  We simulate this approach on a typical fat-tree topology to test our hypothesis.
Reserving Backup Bandwidth on Fat-Tree: Simulation

Simulation details:

- **48-ary fat-tree**: *A Scalable, Commodity Data Center Network Architecture (Sigcomm ’08)*
- **Induce failure model**: *Understanding network failures in data centers (Sigcomm ’11)*
- **Virtual cluster abstraction**: *Oktopus (Sigcomm ’11)*
- **Metric**:

  \[
  \text{Percentage Availability} = \frac{\text{Total uptime experienced by tenants}}{\text{Total duration}} \times 100\% 
  \]
Reserving Backup Bandwidth on Fat-Tree: Simulation

Simulation details:

- 48-ary fat-tree: *A Scalable, Commodity Dc*
- Induce failure model: *Understanding net*
- Virtual cluster abstraction: *Oktopus (Sig)*
- Metric:

\[
\text{Percentage Availability} = \frac{\text{Total uptime experienced by tenants}}{\text{Total duration}} \times 100\%
\]
Reserving Backup Bandwidth on Fat-Tree: Simulation

Simulation details:

- 48-ary fat-tree: *A Scalable, Commodity Dc*
- Induce failure model: *Understanding net*
- Virtual cluster abstraction: *Oktopus (Sig)*
- Metric:

\[
\text{Percentage Availability} = \frac{\text{Total uptime experienced by tenants}}{\text{Total duration}} \times 100\%
\]

Levels off before three 9s of Availability
Reserving Backup Bandwidth on Fat-Tree: Simulation

Simulation details:

- 48-ary fat-tree: *A Scalable, Commodity Dc*
- Induce failure model: *Understanding net*
- Virtual cluster abstraction: *Oktopus (Sig)*
- Metric:

\[
\text{Percentage Availability} = \frac{\text{Total uptime experienced by tenants}}{\text{Total duration}} \times 100\%
\]

So what did we overlook?
Single Point of Failure – ToRs

Inherent to the fat-tree topology

- No alternate path to reroute ToR traffic!
Single Point of Failure – ToRs

Inherent to the fat-tree topology

- No alternate path to reroute ToR traffic!

Potential solutions:

- VM migration
  - Has its own set of challenges
- Modify topology
Fat-Resilient-Trees: High Level Idea

Key idea: Multi-home the end hosts
Fat-Resilient-Trees: High Level Idea

Key idea: Multi-home the end hosts

Goals we target:

• Must have the same cost as its fat-tree counterpart
  
  Which requires having the same number and size of switches
Fat-Resilient-Trees: High Level Idea

Key idea: Multi-home the end hosts

Goals we target:

• Must have the same cost as its fat-tree counterpart
  Which requires having the same number and size of switches

So we simply **Rearrange** the existing redundancy

• Introducing redundancy at ToR level by stripping it from overly redundant levels.
Fat-Resilient-Trees: High Level Idea

• Uniformly remove the overly redundant links
Fat-Resilient-Trees: High Level Idea

- Uniformly remove the overly redundant links
- Reconnect them in a way which ensures that every end-host is connected to every other end-host
Fat-Resilient-Trees: High Level Idea

Works because of Locality in Traffic:

- Collocation motivates that full bisection BW is perhaps at every level an overkill
Fat-Resilient-Trees: High Level Idea

Works because of Locality in Traffic:

- Collocation motivates that full bisection BW is perhaps at every level an overkill

Preliminary simulation results show *five 9s of Availability*
Ongoing Work

• Understand and evaluate the implications Fat-Resilient-Trees

• Extensively compare against existing topologies

• Build a fast recovery mechanism
Questions & Feedback?

Thank you for your time 😊
References


## Backup Slides

### VM Migration

<table>
<thead>
<tr>
<th>Configuration</th>
<th>AVAIL</th>
<th>Efficiency</th>
</tr>
</thead>
<tbody>
<tr>
<td>oktopus + nothing</td>
<td>99.683</td>
<td>1</td>
</tr>
<tr>
<td>oktopus + t2t backup</td>
<td>99.809</td>
<td>0.8225308642</td>
</tr>
<tr>
<td>oktopus + t2t + 2 backups</td>
<td>99.83</td>
<td>0.7685185185</td>
</tr>
<tr>
<td>oktopus + e2e + 1 backup</td>
<td>99.9998</td>
<td>0.4907407407</td>
</tr>
<tr>
<td>oktopus + e2e + 2 backups</td>
<td>99.9999</td>
<td>0.3364197531</td>
</tr>
<tr>
<td>oktopus + sharing + 1 pod</td>
<td>99.9998</td>
<td>0.9768518519</td>
</tr>
<tr>
<td>oktopus + sharing + 5 pods</td>
<td>99.9999</td>
<td>0.8796296296</td>
</tr>
<tr>
<td>oktopus + new topology + backups</td>
<td>99.997</td>
<td>0.8641975309</td>
</tr>
</tbody>
</table>