ERICSSON

P2P OPENSTACK

SCALABILITY AND THE EDGE USE-CASE

openstack™

João Monteiro Soares
Ericsson Research

DISTRIBUTED CLOUD
AND THE EDGE USE-CASE

Hierarchical vs P2P

› Hierarchical

+ Scalable
+ Simpler logic
+ Effective/deterministic
+ Efficient

- Single entry point (root)
- Failure handling difficult
- Can't handle churn

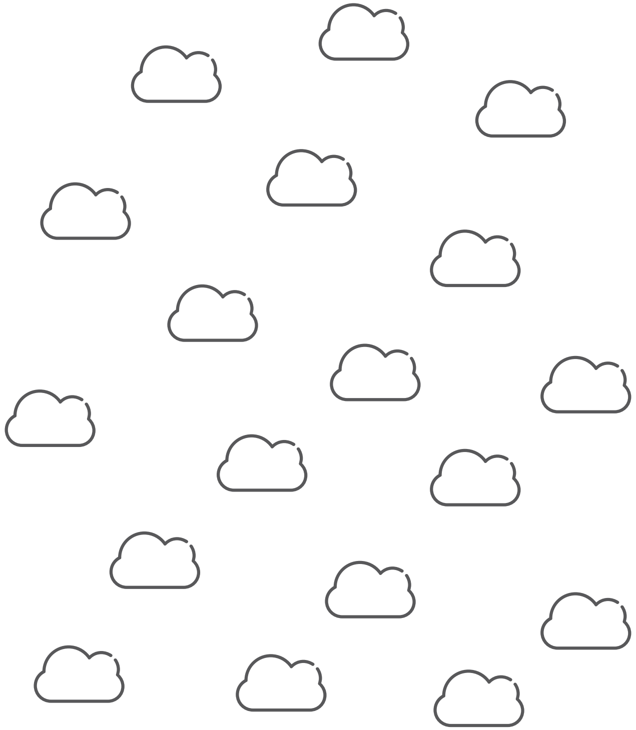
› (Unstructured) peer-to-peer

+ Extremely scalable
+ No single point (of failure or bottleneck)
+ Can handle churn well

- Probabilistic/approximate
- More complex logic
- More overhead

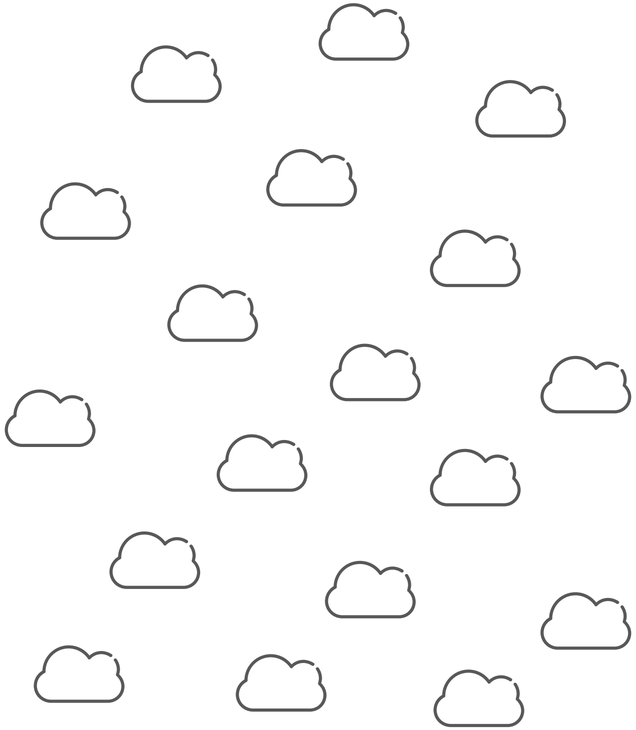
DISTRIBUTED CLOUD
AND THE EDGE USE-CASE

- How to manage a cloud infrastructure with such scale?

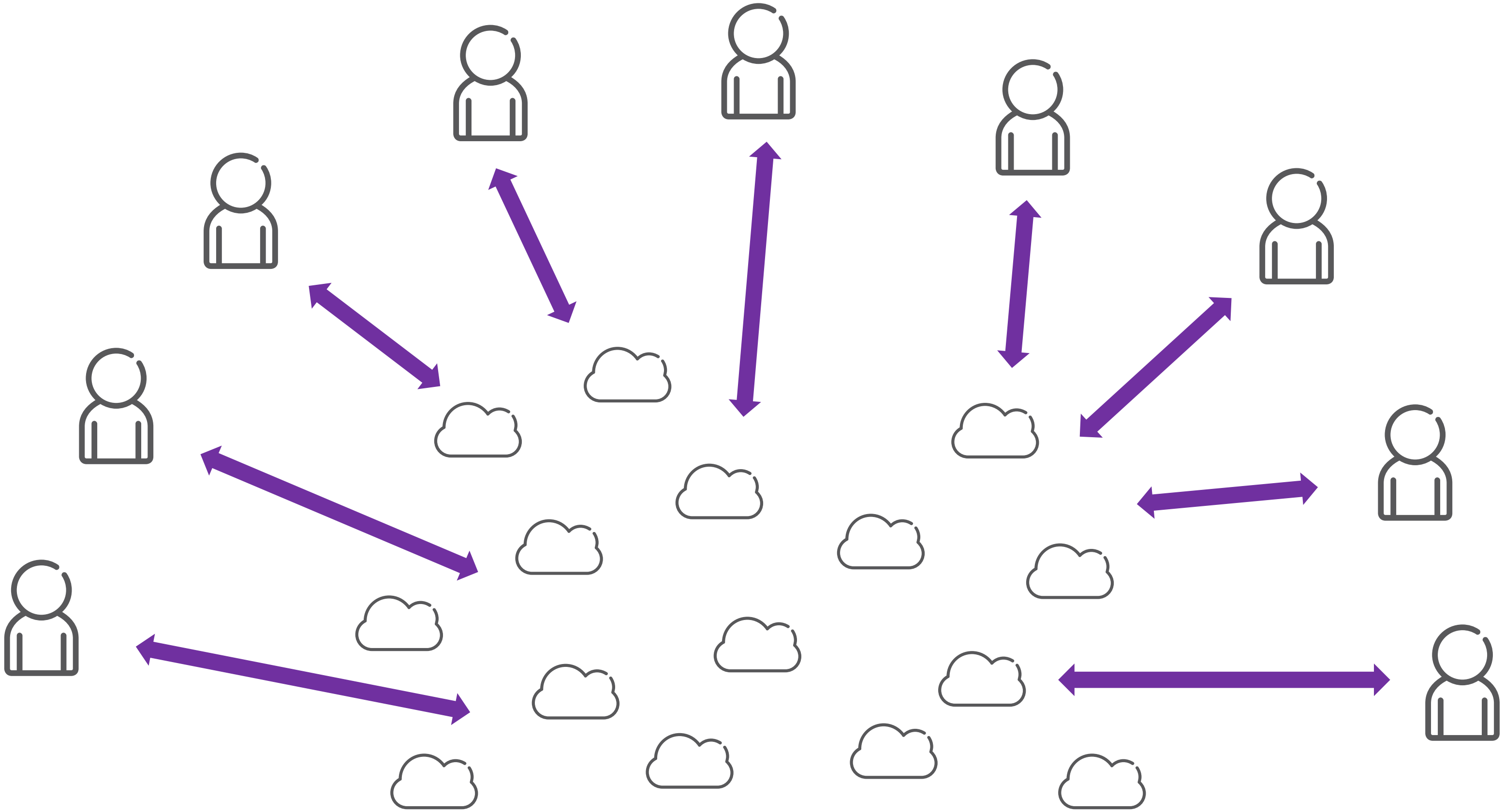


DISTRIBUTED CLOUD
AND THE EDGE USE-CASE

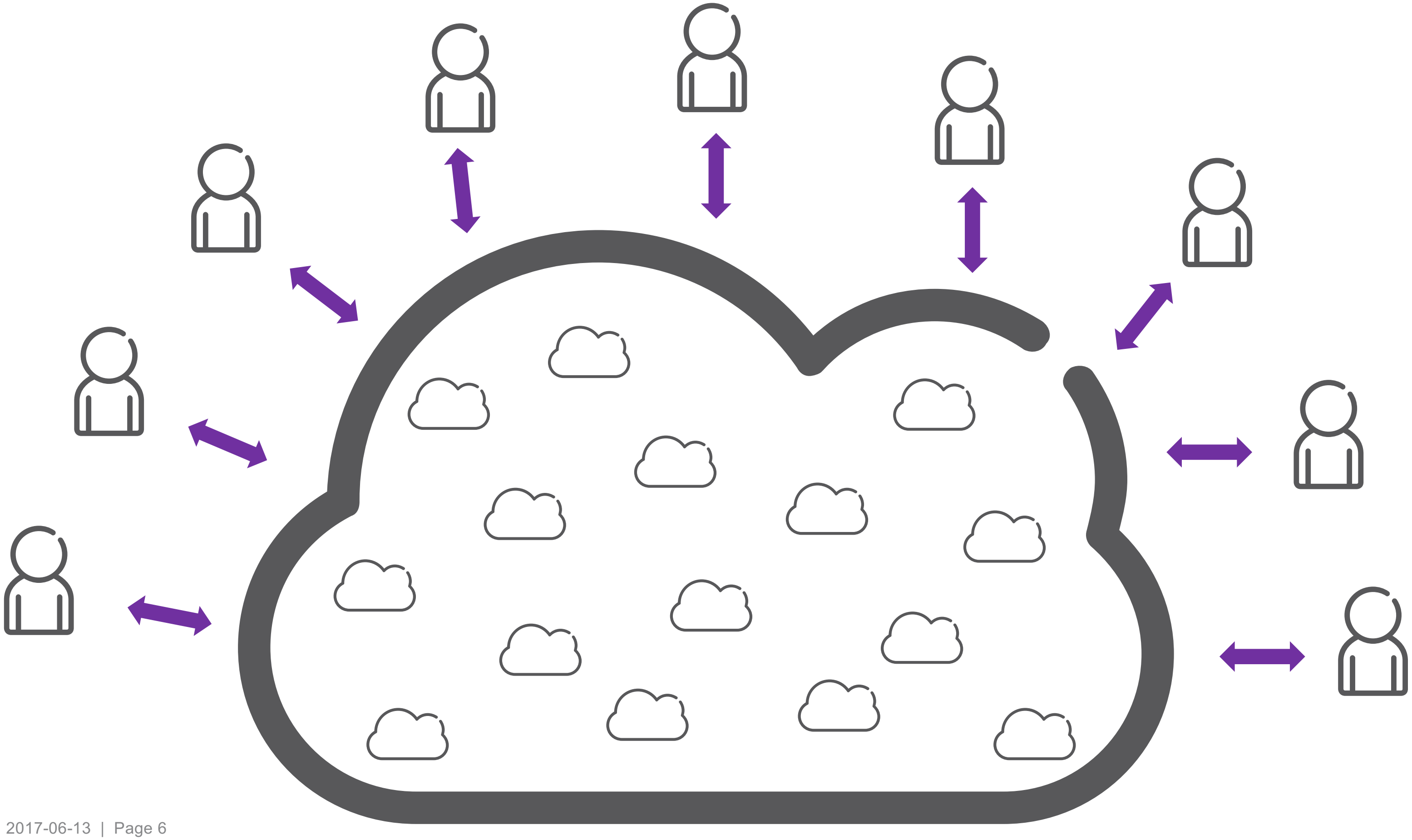
- How to manage a cloud infrastructure with such scale?
  - Seeking key design principles:
    - Avoid centralized components (as much possible)
    - Avoid (Minimize) any changes to the existing software
  - Proposal:
    - Use P2P techniques to horizontally scale OpenStack
    - Provide a single-cloud abstraction



SINGLE CLOUD ABSTRACTION



SINGLE CLOUD ABSTRACTION



A P2P APPROACH

- Divide an OpenStack deployment into a (possibly large) number of smaller OpenStack ‘cloudlets’, each with its own controller and compute nodes

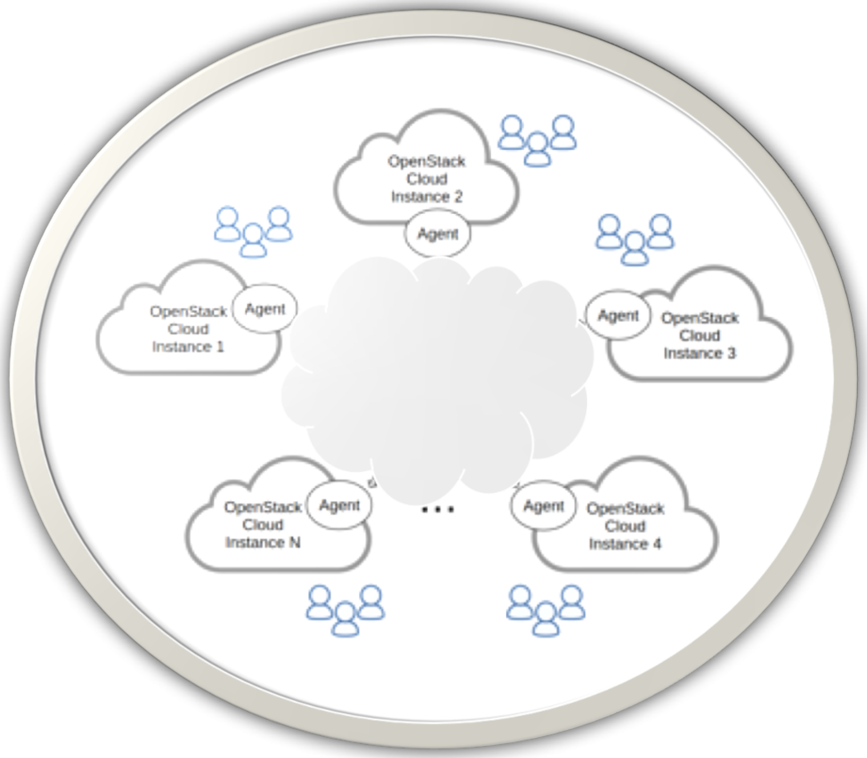
- Associate an agent with each cloudlet

  (Note: agent functionalities could be directly embedded in OpenStack)

- Agents forward requests to other cloudlets (reverse proxy)

- Implement the single-cloud abstraction

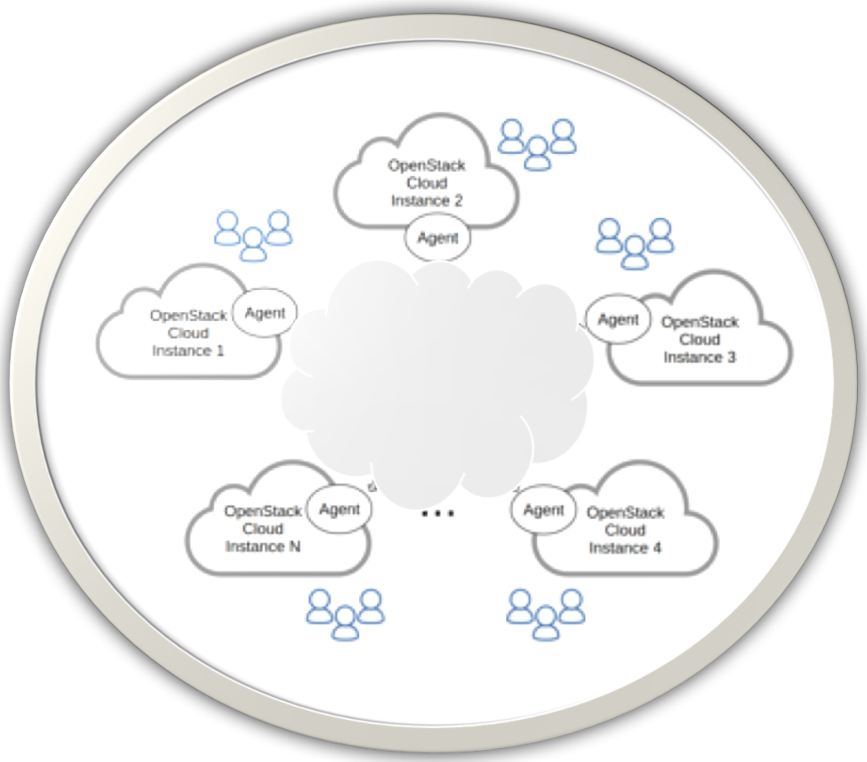
- OpenStack APIs ‘untouched’ (remain the same)


OpenStack Cloud Instance 2
Agent
OpenStack Cloud Instance 1
Agent
Agent
OpenStack Cloud Instance 3
OpenStack Cloud Instance N
Agent
...
Agent
OpenStack Cloud Instance 4

A P2P APPROACH

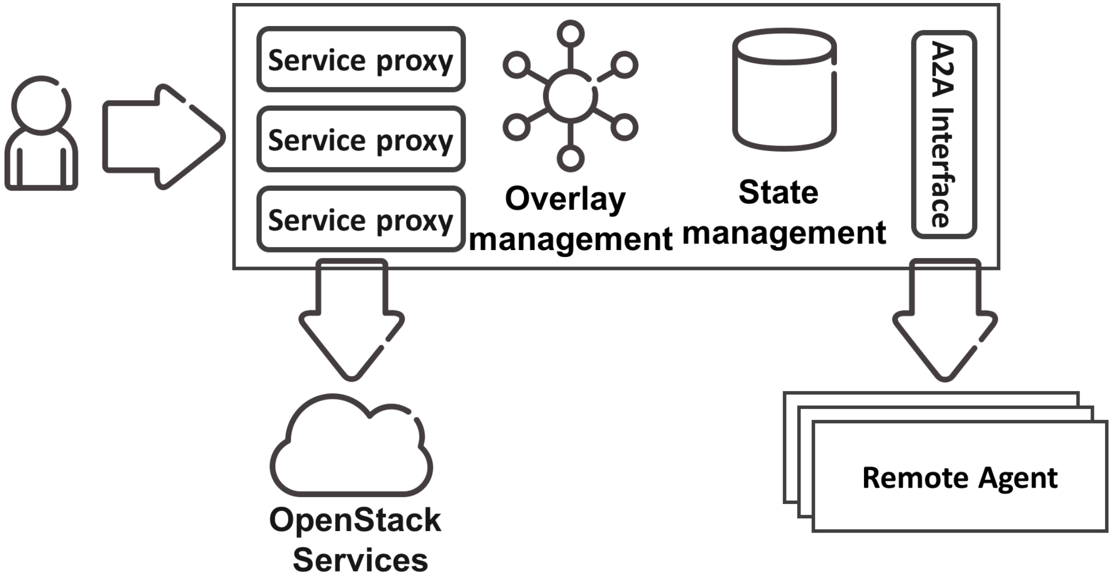
- Projects/tenants are mapped to agents

- When an agent receives a request it
  - first chooses the cloudlet responsible for that request
  - forwards the request to the controller of the chosen cloudlet
  - updates a local mapping table upon receipt of the response
  - returns the reply to the user


OpenStack Cloud Instance 2
Agent
OpenStack Cloud Instance 1
Agent
Agent
OpenStack Cloud Instance 3
OpenStack Cloud Instance N
Agent
. . .
Agent
OpenStack Cloud Instance 4

A P2P APPROACH

- An agent implements three key functions
  - Service Proxy (Keystone, Nova, Glance, Neutron, …)
  - Overlay Management
  - State Management



- Agent-to-Agent Interface

A P2P APPROACH

- AA(A) - Keystone Federation
  - An Identity Provider (IDP) outside of the cloud
  - Keystone in each cloudlet configured as a Service Provider (SP)
- Images:
  - external storage, only add location to Glance
- Block storage (tbd)
- Networks (tbd):
  - e.g. OpenStack Tricircle

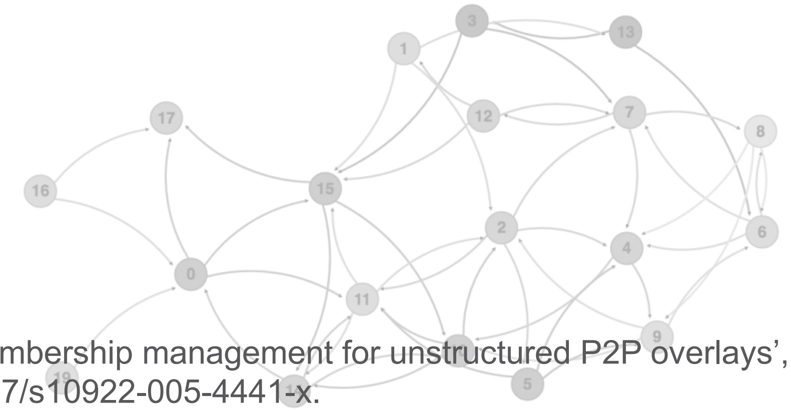
A P2P APPROACH: SCHEDULING

- Randomized algorithms for:
  - Peer sampling: choose random peers
    - Peer Sampling Protocol, **Cyclon**, Newscast, SCAMP
  - Distributed search: determine which node takes the workload
    - **“The Power of Two Choices”**

CYCLON

- › Each node maintains a random ‘cache’ of nodes
- › Periodically this cache is exchanged with the oldest node in the cache
- › Protocol attempts to keep the size of the cache ‘constant’

➢After the protocol stabilizes (in O(log n) steps), each node’s cache holds a random subset of the other nodes in the system



Voulgaris, S., Gavidia, D. and van Steen, M. (2005) ‘CYCLON: Inexpensive membership management for unstructured P2P overlays’, Journal of Network and Systems Management, 13(2), pp. 197–217. doi: 10.1007/s10922-005-4441-x.

THE POWER OF TWO CHOICES

- Objective is to balance load

- Very simple algorithm:
  - Select two nodes at random
  - Send the workload to the node with smallest load

- Compared to randomly choosing a node, the maximum load decreases from O(log n) to O(log log n)

Mitzenmacher, M. (2001) ‘The power of two choices in randomized load balancing’,
IEEE Transactions on Parallel and Distributed Systems, 12(10), pp. 1094–1104. doi: 10.1109/71.963420.

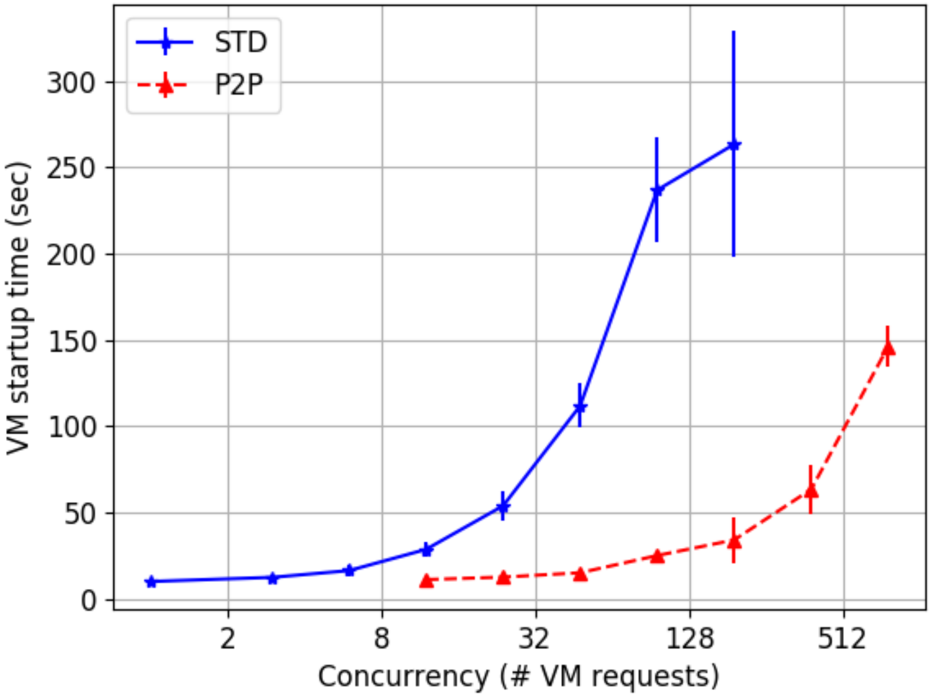
P2P VS. STANDARD
SYSTEM PERFORMANCE

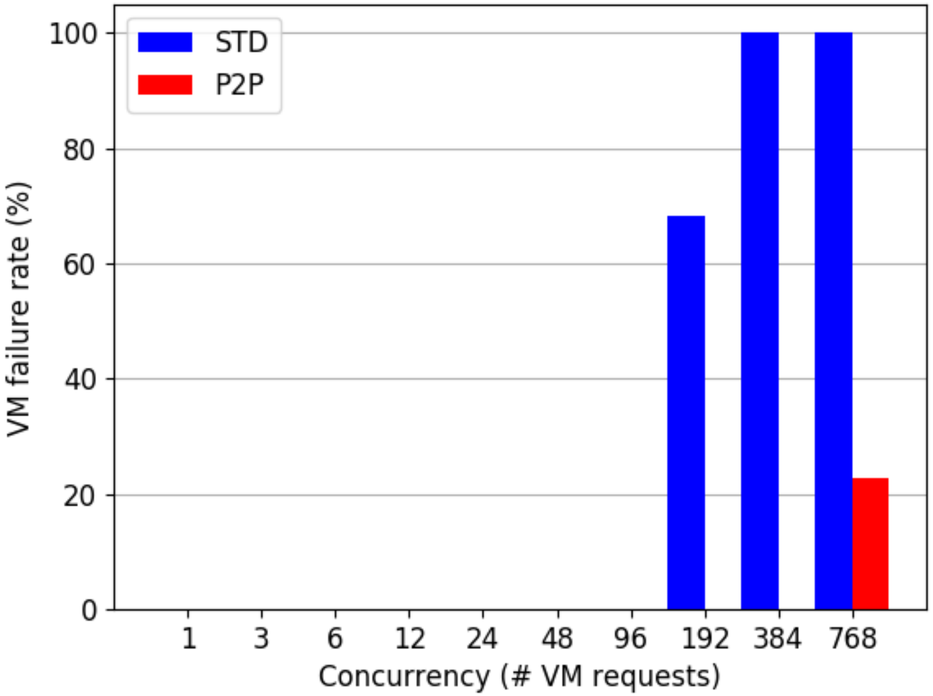
- Standard:
  - 1 Controller (8 cores/16GB RAM)
  - 64 Compute nodes (1 core/ 2GB RAM)

- P2P :
  - 64 Cloudlets
  - 1 Controller (2 Cores/ 4GB RAM+1GB SWAP)
  - 1 Compute (1 core/ 2GB RAM)
  - 1 Services (2 cores/4GB RAM)
    - IDP, file server, Cyclon Introducer

P2P VS. STANDARD

SYSTEM PERFORMANCE





MOVING FORWARD

- Current approaches use hierarchical techniques to achieve OpenStack scalability
  - Distributed cloud and the edge case bring another dimension to the problem

- What about P2P techniques to horizontally scale OpenStack ?
  - A novel approach to address cloud scalability
  - Reference implementation proves feasibility of the approach
  - Several challenges still remain

- Next steps:
  - Support location requirements
  - Open Source reference implementation

ERICSSON