Scaling Peer-to-Peer Testing with Linux Containers

Mircea Bardac, Razvan Deaconescu, Adina Magda Florea

POLITEHNICA University of Bucharest

9th RoEduNet International Conference
June 24-26, 2010, Sibiu, Romania

Contents

- Introduction & Context (P2P and LXC)
- Experiment design & discovered limitations
- Experimental evaluation of LXC scaling
- Scaling challenges
- Future work & Conclusions

Introduction

- **Studying the peer performance and behaviour** inside a swarm
  - Known and unknown protocols
  - Real-life client implementations
- Approaches for studying P2P systems
  - Simulators
  - Real-deployments
  - **Virtualization**
- BitTorrent as a P2P implementation

Virtualization

- Previously tested virtualized applications:
  - tracking application interaction (Huang et. all)
  - simulating BitTorrent swarms (Deaconescu et. all)
- BitTorrent on OpenVZ infrastructure (Deaconescu et. all)
  - Limited number of virtualized peers (max. 5 peers/node)
  - OpenVZ - No support in the Linux kernel mainline
- Known results: *hrktorrent* determined as the fastest client
- Solution: Linux Containers (LXC)

Linux Containers (LXC)

- Lightweight virtualization solution (Operating System level)
  - isolated resources
  - processes (~ process groups), memory, file system (~ chroot)
- Virtualization solution implemented in kernel mainline
  - starting with kernel version 2.6.29 (March 2009)
- Node = Container
- Real BitTorrent clients running each node (= peers)

Linux Containers (2)

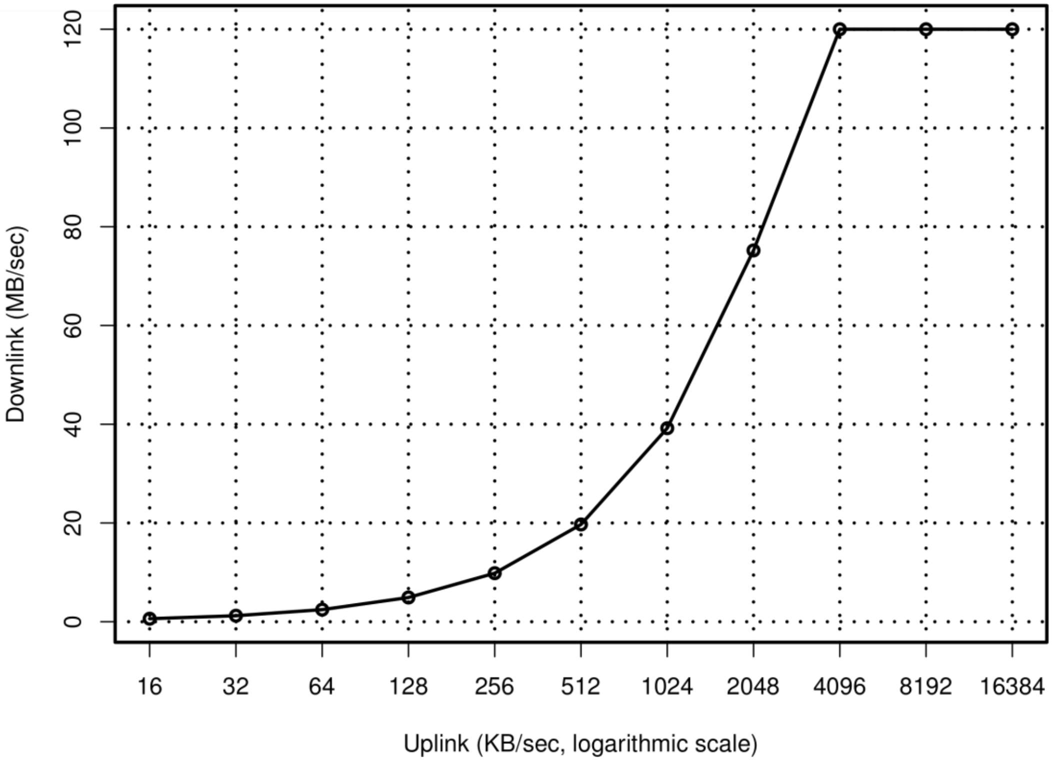
- Control Groups (cgroups) for:
  - Process environment isolation
  - Managing restrictions
- Created framework for managing the entire life-time of the Peer-to-Peer swarm on top of LXC
  - Description of the node topology and their attributes
  - Description of the P2P clients
  - Starting/Stopping/Destroying the nodes

Experiment overview

- Host system
  - Intel(R) Core(TM)2 Duo CPU T7500 @ 2.20GHz
  - 1.5 GB RAM
  - 5.7 GB HDD partition
- Debian testing ("squeeze") with stock 2.6.32 Linux kernel
- Bandwidth limitation: **tc**

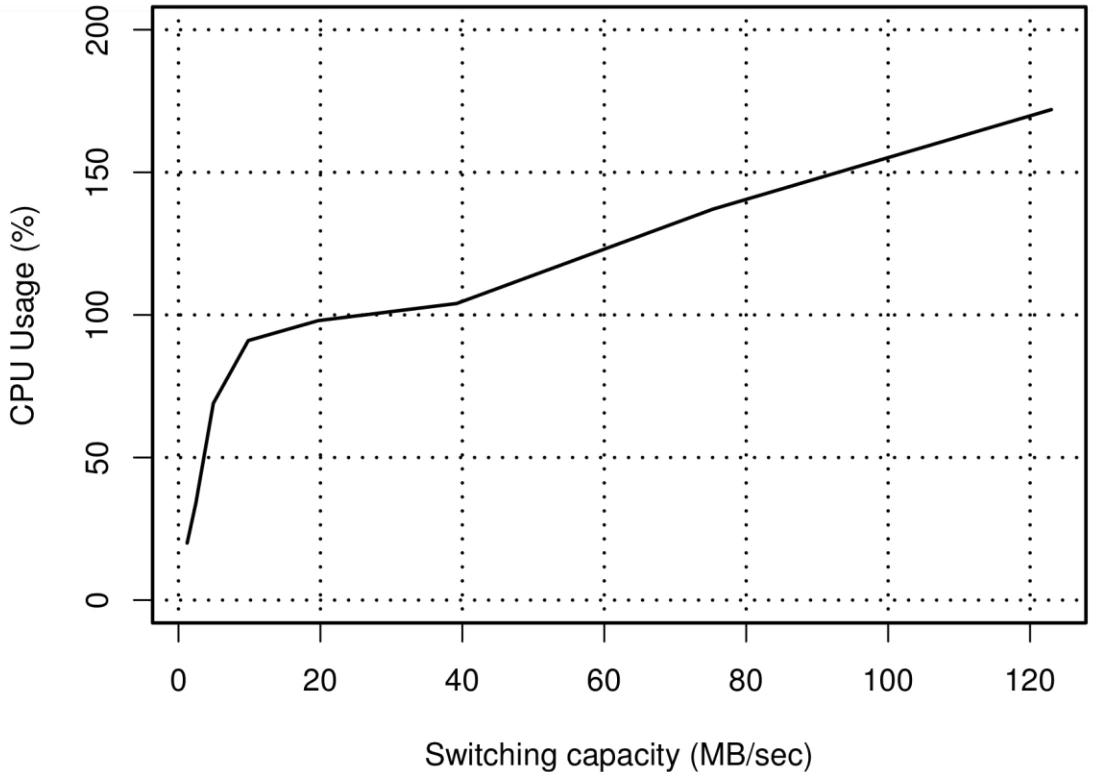
Virtualization Limitations

- **Uplink limitation** can affect the **downlink capacity**



Virtualization Limitations (2)

- Switching capacity - CPU usage (peak 123 MB/sec – 172 % CPU)



Experiment details

- 6 scenarios
- Containers:
  - 1 tracker
  - 0, 20, 40, 60, 80, 100 peers (10 % seeders, 90 % leechers)
- Node bandwidth limitations:
  - Uplink: 32 KB/s
  - Downlink: 128 KB/s

Experiment details – File system

- Baseline file system for all peers
- **Read-only bind mount points** for common file system directories (/bin, /usr, /lib etc.)
- Per container directories:
  - /root – container specific files, torrent data
  - /var – logs, temporary files

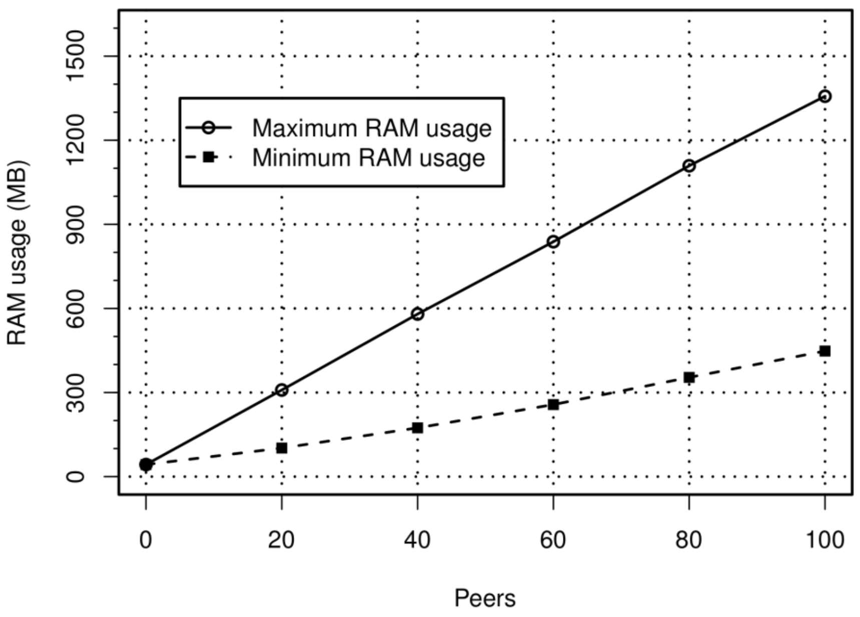
Host analysis

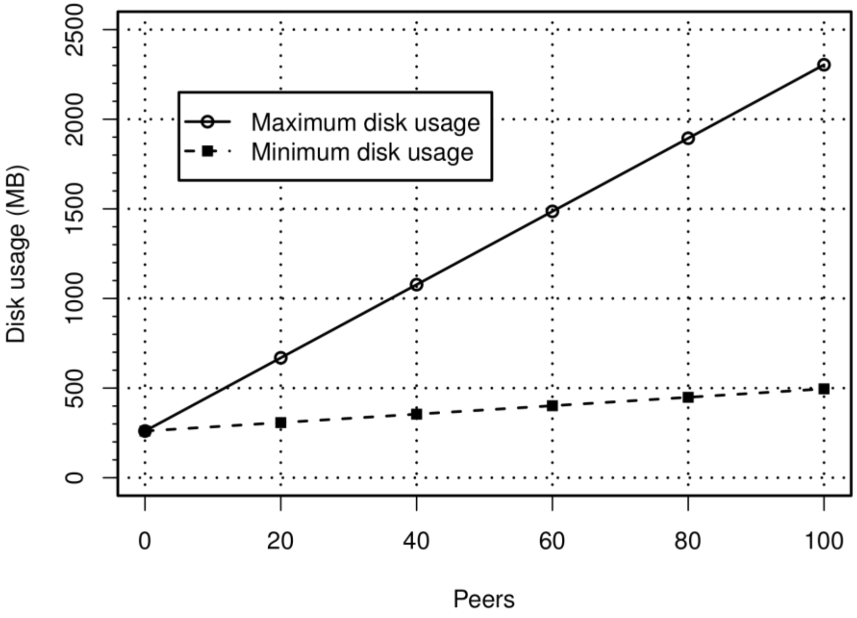
- File system usage
  - Baseline file system: 260.45 MB
  - Node file system:
    min 1 MB, max 1 MB + 20 MB torrent data + logs 100 KB
  - Peak (100 peers): 2043.2 MB
- RAM
  - Host RAM: 43 MB
  - Container RAM: min 4 MB, max 13 MB
  - Peak (Host + 100 peers): 1357 MB

Host analysis (2)

- CPU usage:
  - Processes/node: lxc-start, init, gettty, sshd, hrktorrent/bttrack
  - Peak node process count (100 peers): 500 processes
  - Peak CPU usage: 90%
- Linear growth of resource usage
  - File system (peak ~2 GB)
  - CPU (peak 90%)
  - RAM (peak 1357 MB - close to experiment limit)

Host analysis (3)

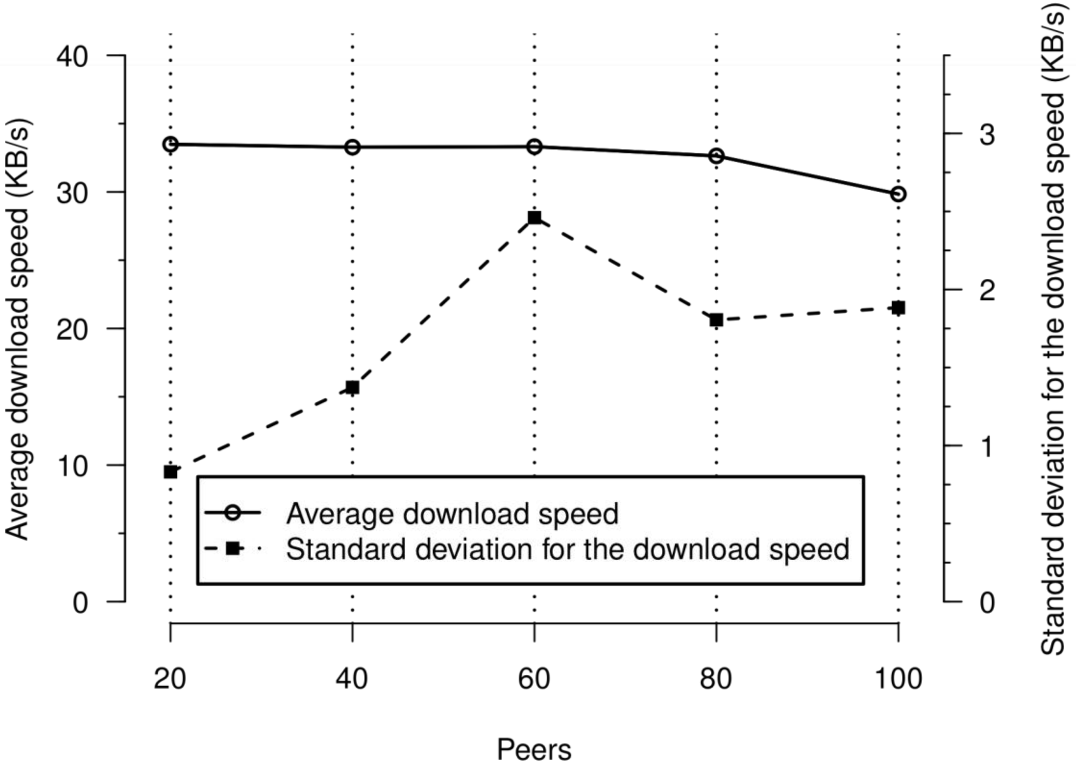




Swarm analysis

- Tracker logs vs. **Peer logs**
- Measuring the impact on performance for each peer:
  - ~~Average download speed (time)~~
  - **Standard deviation** for the average download speed (time)
- Slowly increasing trend for the standard deviation

Swarm analysis (2)



Scaling challenges

- Switching leads to increased CPU usage
  - Prevent CPU contention by traffic shaping
- Uplink limitations can affect downlink traffic
- Peer implementation details
  - Example: *hrktorrent* favors local network peers

Scaling challenges (2)

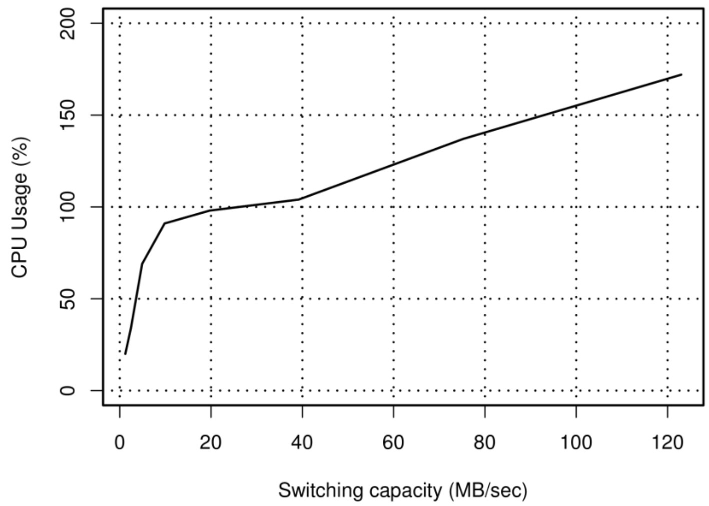
- Host components might not scale properly - **Network ARP cache**
- *ping* command error - ‘connect: No buffer space available’
- Logs: ‘Network table overflow’ errors
- A normal host vs. 100 containers with less than 100 neighbours
- Solution: increase network caches by a factor of 256

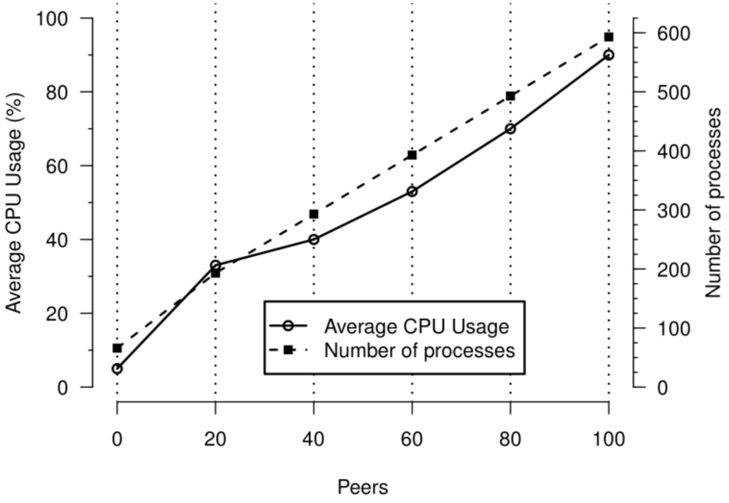
Future work

- Linux Containers benefits from using *cgroups*
- CPU-set support – pinning containers to specific CPU cores
- CPU accounting
- Memory Resource Controller (adds overhead)
- Block I/O controller

Future work (2)

- Correlations between **swarm resource consumption** and **swarm performance**
- initial switching capacity testing: **9.83 MB/sec** uses **90% CPU**
- 100 peers swarm: **2.62 MB/sec** uses **90 % CPU**





Future work (3)

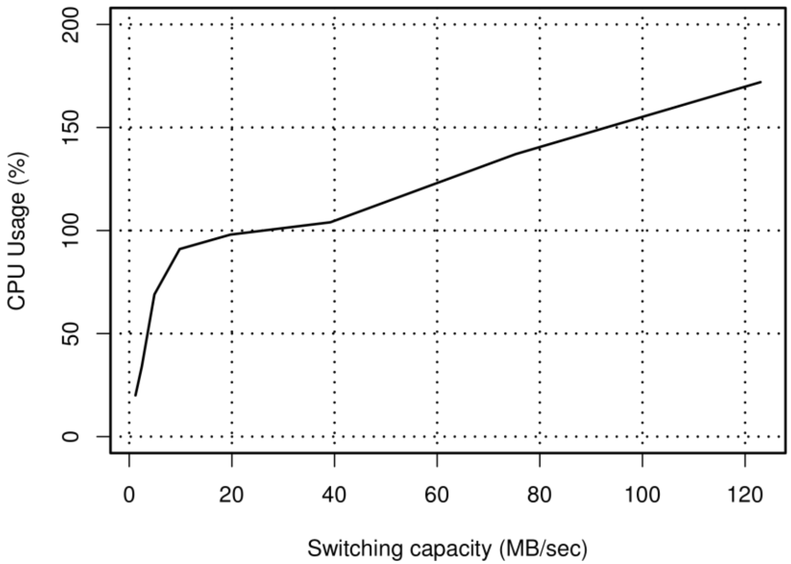
- Resource usage expectations:
- 100 peers swarm: **2.62 MB/sec** uses **90 % CPU**
- 2.62 MB/sec **should use ~60 % CPU**

- Overhead of 500 processes running at the same time: context switches, I/O etc.

- Maximum switching capacity with the same overhead?
- 172 % CPU usage has a maximum 123 MB/sec
- With the same overhead (~ 30%), switching should be the equivalent of ~120 % CPU usage

Future work (3)

- Maximum switching capacity with the same overhead
  - 172 % CPU usage has a maximum 123 MB/sec
  - ~120 % CPU usage has **a maximum of 40-50 MB/sec?**



Conclusions

- LXC – virtualization platform
- Advantages
  - Real P2P clients (BitTorrent)
  - Read-only bind mount points (extremely low disk footprint)
  - Available in the kernel mainline
- Disadvantages
  - Still in development (example: cgroups)
  - Scarce documentation

Conclusions (2)

- Impact of scaling an LXC-based testing platform on
  - Host resource utilization (File system, RAM, CPU)
  - Swarm performance
- Virtualization in multiple-container scenarios
- Multiple scaling challenges
- Platform for testing **real-life P2P applications**

Scaling Peer-to-Peer Testing with Linux Containers

Mircea Bardac, Razvan Deaconescu, Adina Magda Florea

POLITEHNICA University of Bucharest

9th RoEduNet International Conference
June 24-26, 2010, Sibiu, Romania