TELEPORTING VIRTUAL MACHINES

Research Project 1

Harm Dermois
harm.dermois@os3.nl
Carlo Rengo
carlo.rengo@os3.nl

Supervisors:
Oskar van Deventer (TNO)
Jan Sipke van der Veen (TNO)

Wednesday 4th February, 2015

Master System and Network Engineering
University of Amsterdam

What is VM “teleportation”?

- A classic **VM** copy across the Internet moves an unnecessary quantity of bits
- Instead of copying the whole **VM**, a description of the source is used to recreate it on the destination Hypervisor
- Like a “teleport”, the **VM** is broken down into logical parts (software, configuration and user data) and reconstructed somewhere else
- The new **VM** is not an exact replica, but it’s still a functional copy

VIRTUAL MACHINE TELEPORTATION (2)

Why would you need VM “teleportation”?

- Slow network speed between source and destination (i.e. endpoints very distant from each other) makes fetching data from other sources desirable
- Might be used as a baseline for a **VMDN** (Virtual Machine Delivery Network)
- As in a **CDN** (Content Delivery Network) an object might be moved “next” to the end users for faster services responsiveness
- Might save bandwidth
- Might be faster

RESEARCH QUESTIONS

- Is it possible to implement a *teleporting system* in a real world scenario?
- Is the data transferred less for a teleported VM than for a conventionally migrated one?
- Is a teleported VM indeed quicker up-and-running than a conventionally migrated one?

CONTEXT & SCOPE

- Focus on data transferred between source and destination
- Focus on time spent teleporting a VM (CPU and memory consumption were not measured)
- Full control (root access) of source and destination servers
- Source **VM** is powered off for the sake of simplicity
- Every **VM** has only one virtual disk

We wrote a Proof of Concept that:

- Analyzes local and remote **VM**s
- Can create a new **VM** from scratch
- Uses the most similar **VM**s (if any) to recreate the source
- Automatically installs any needed software on the destination **VM**
- Synchronizes any difference from the source to the destination **VM**

We wanted our **PoC** to:

- Be easy to install
- Use only common libraries (`libvirt`, `libguestfs`)
- Make no changes to the source **VM**
- Work (with some modifications) with hypervisors such as **KVM**, **Xen** and **VMware®**

Note: *At the moment, only **CentOS** and **Ubuntu** guests are supported*

SMART-MIGRATE ALGORITHM

ALGORITHM DESCRIPTION - STEPS

1. **generateDescription()** and **fetchDescription()** - Contact the source Hypervisor and ask it to create a description of the VM (OS version and packages installed)
2. **listImages()** and **pickCandidate()** - Look for a local VM with the same distribution and version, clone the one with the least amount of differences from the source VM. If there is no candidate, create a VM from scratch.
3. **swPrepare()** and **swInstall()** - Install any missing distribution package on the cloned/new VM and remove any extra package.
4. **smartSync()** - Copy user data (files, databases, etc...) and software configurations

ALGORITHM DESCRIPTION (2)

pickCandidate() details:

- Only virtual disks/snapshots with same distribution and version are taken into account
- Dry runs (no real transfer, only an estimation - very fast!) of `rsync` to find the best candidate
- If no candidate is found, create a new **VM**

smartSync() details:

- Two runs of `rsync`
- The first one syncs everything but the installation folders
- The second one syncs all the files in the installation folders that do not exist on the destination (software/libraries not installed by a package manager).

SETUP

Local Hypervisor (Delft Brasserskade):

Model:	Dell System XPS L702X
CPU:	Intel®Core™i7-2620M CPU @ 2.70GHz (Dual Core)
Memory:	8GiB RAM SODIMM DDR3 Synchronous 1333
Disk:	Seagate ST9500420AS - 500GB (non SSD)
OS:	Ubuntu 14.04 64-bit with KVM

Remote Hypervisor (Amsterdam Science Park):

Model:	Dell PowerEdge R210 II
CPU:	Intel®Xeon®CPU E3-1220L V2 @ 2.30GHz (Dual Core)
Memory:	8GiB RAM DIMM DDR3 Synchronous 1333 MHz
Disk:	Seagate ST1000NM0011 - 1TB (non SSD)
OS:	Ubuntu 14.04 64-bit with KVM

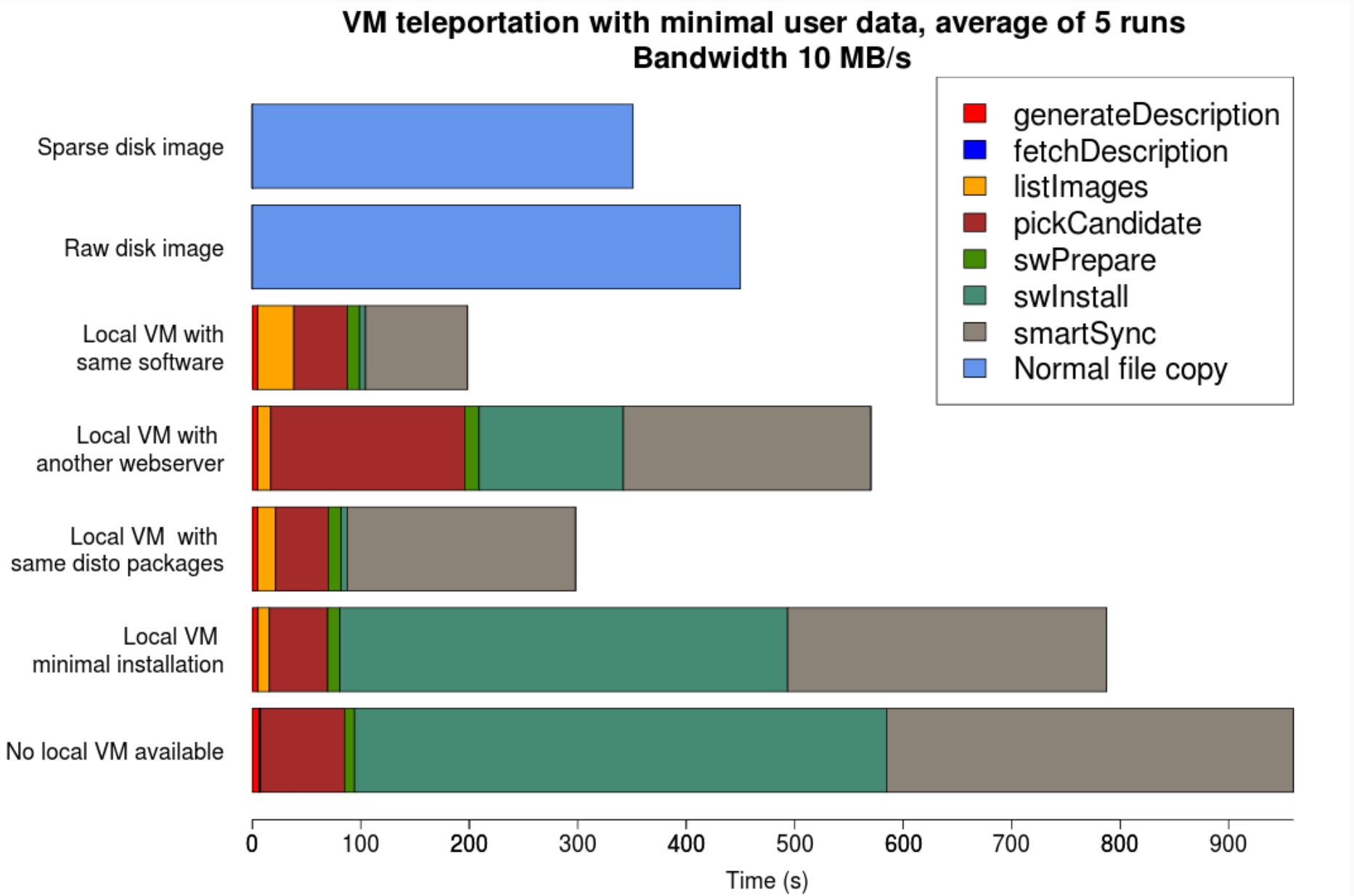
SOURCE VMS DESCRIPTION

Two VMs with the following characteristics:

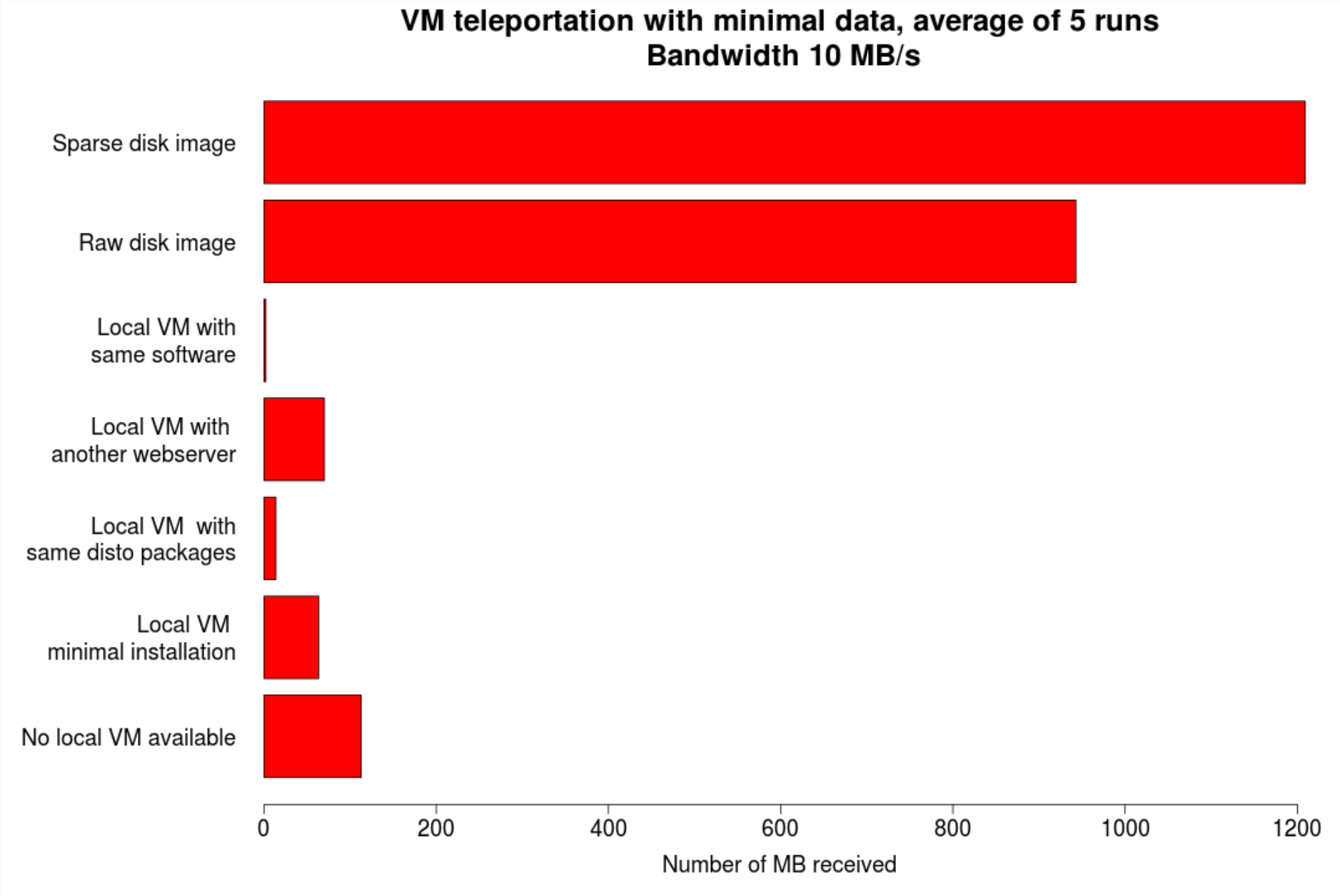
OS:	CentOS 7.0 64-bit
Software:	ISPConfig hosting panel (mostly distribution packages plus some compiled software)
Data:	A couple of website (and their databases)
Data(2):	Same as above, but with 9GiB of random data divided in small and big files.
Network:	DHCP
Disk usage:	2GiB out of 30GiB
Disk usage(2):	11GiB out of 30GiB

Teleportation results - 2GiB VM

TELEPORTATION - TIME

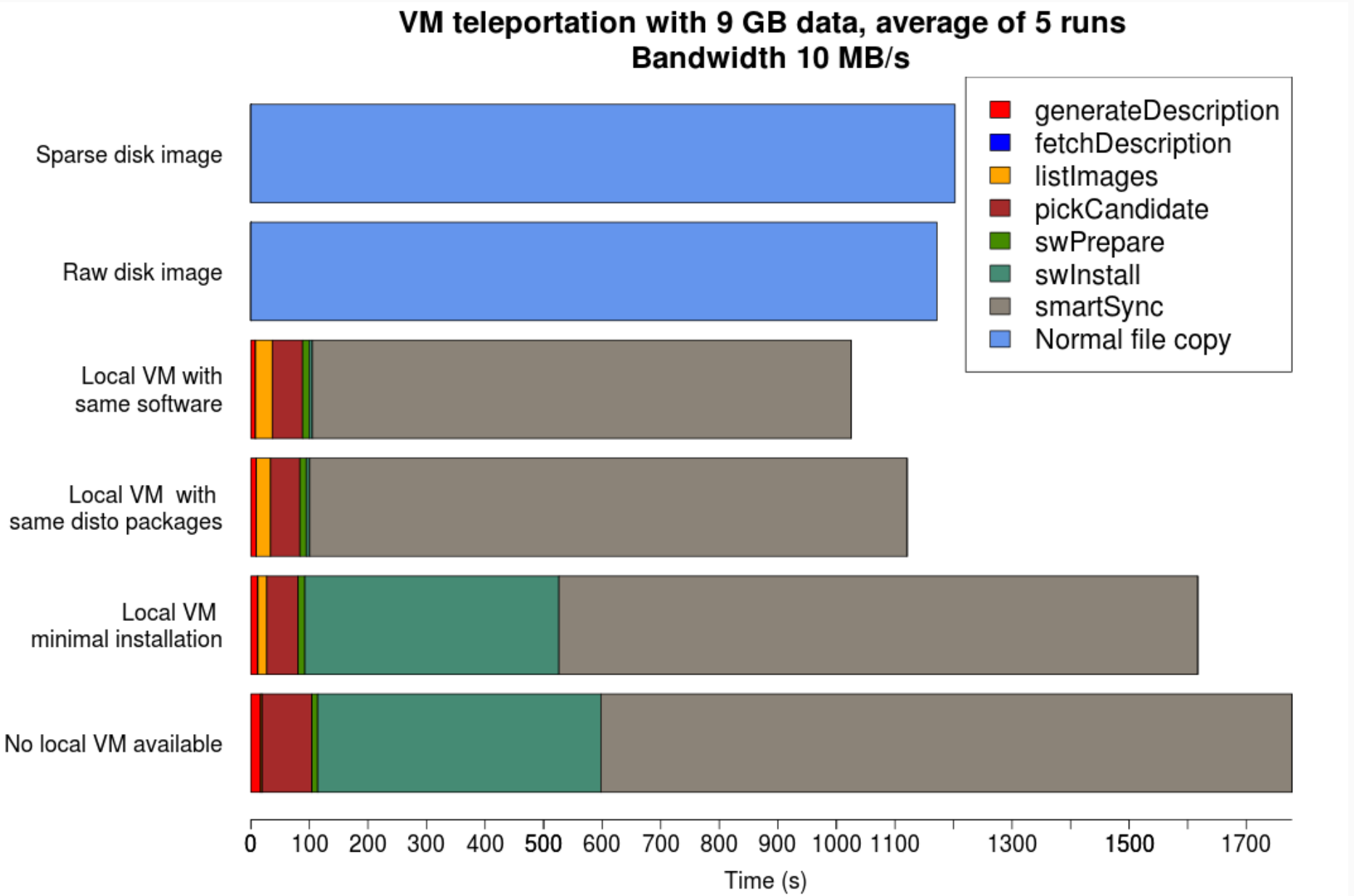


TELEPORTATION - BANDWIDTH CONSUMPTION BETWEEN HYPERVISORS

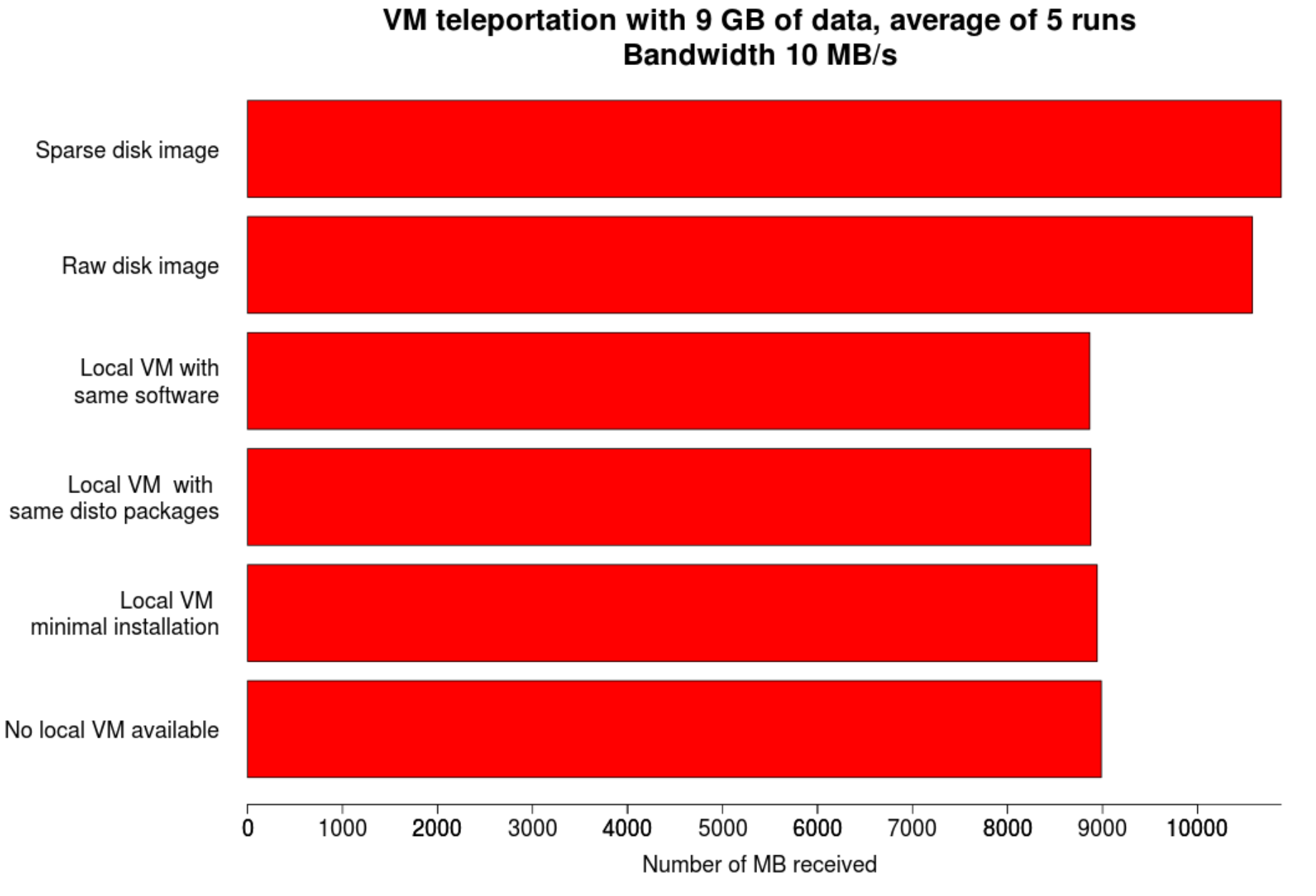


Teleportation results - 11GiB VM

TELEPORTATION - TIME(2)



TELEPORTATION - BANDWIDTH CONSUMPTION BETWEEN HYPERVISORS(2)



CONCLUSIONS AND FUTURE WORK

CONCLUSIONS

- It works, although more scenarios should be tested
- Very little metadata was sent between source and destination
- **VM** Teleportation can save bandwidth...
- **VM** Teleportation can be faster...
- but having similar local copies and snapshots is crucial to achieve such results!

Optimizations (Software can be definitively improved):

- Smarten the algorithm (i.e. sometimes a plain copy is just better)
- Implement parallelization
- Dump & restore database instead of copy
- Some functions in the algorithms can be precomputed asynchronously
- Might also use tools such Puppet and Docker, or wrappers like Vagrant

Thank you for listening!

Questions?

Problems occured:

- Finding the required packages in a powered off **VM**
- Installing packages on a powered off **VM**

`rsync` commands in **smartSync()**:

```
rsync -azAX --delete --stats --exclude={"/dev","/tmp","/proc",\
"/sys","/var/tmp","/run","/mnt","/media","/lost+found","/usr",\
"/lib","/etc/fstab","/lib32","/lib64","/boot"} \
$HYPERVISOR:$R_MNT_PATH/ $L_MNT_PATH

rsync -azAX --ignore-existing --stats --exclude={"/dev","/tmp",\
"/proc","/sys","/var/tmp","/run","/mnt","/media","/lost+found",\
"/boot"} $HYPERVISOR:$R_MNT_PATH/ $L_MNT_PATH
```