UMassAmherst

An Empirical Study of Memory Sharing in Virtual Machines

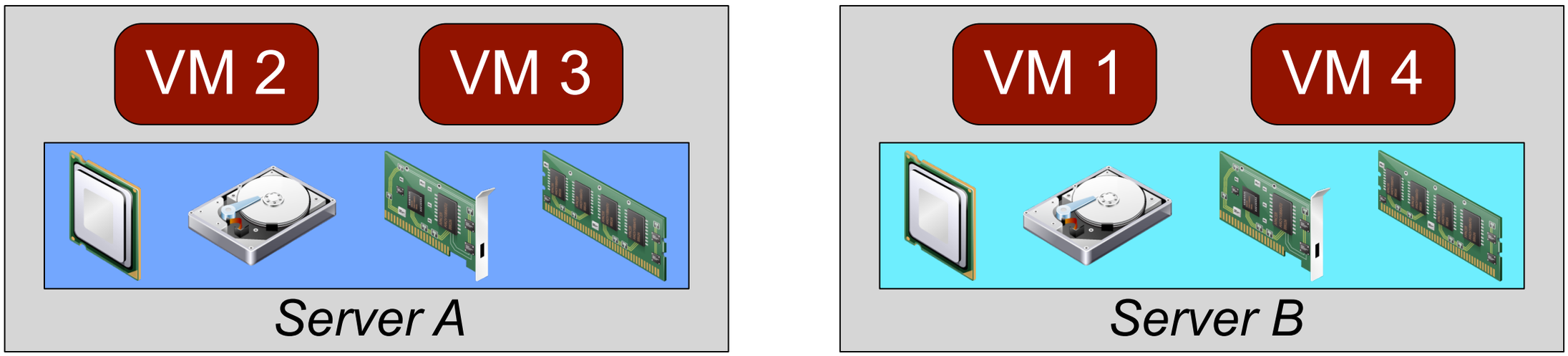
Sean Barker, Timothy Wood†,
Prashant Shenoy, and Ramesh Sitaraman

University of Massachusetts Amherst
The George Washington University†

Virtualization in Data Centers

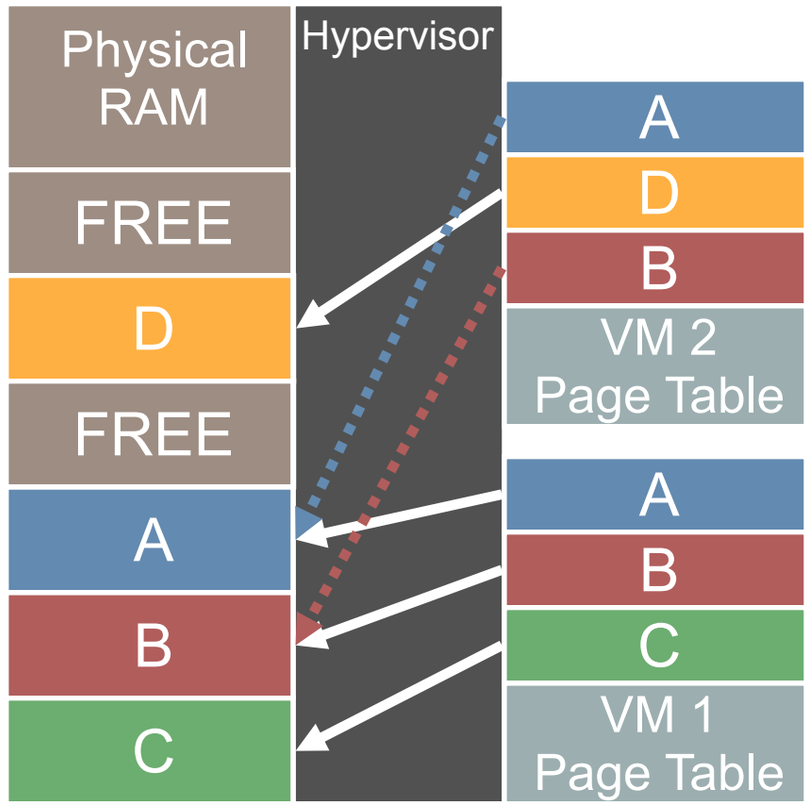
- Data centers use **virtualization** to improve resource utilization
  - Flexible mapping of resources to users
  - More servers and applications
  - Smaller hardware footprint

- Maximizing benefits
  - Efficient resource sharing
  - Virtual machine placement



Content Based Page Sharing

- Eliminate **identical pages** of memory across multiple VMs
- Virtual VM pages mapped to physical pages
- Hypervisor detects duplicates
- Replaced with copy-on-write references



Page Sharing Systems

- Extensive prior work in exploiting page sharing

- VMware ESX Server [SIGOPS 02]
  - Periodic memory **scanning** to detect duplicates
  - **>30%** memory savings

- Difference Engine [OSDI 08]
  - **Sub-page** sharing and patching
  - **>60%** memory savings

- Satori [USENIX 09]
  - Sharing of **short-lived** pages
  - **>90%** of possible sharing captured

Open Questions on Sharing

- What levels of sharing are possible in typical **real-world machines**?

- What are the **factors** that impact sharing potential?
  - OS family? Versions? Applications?

- How will **emerging technologies** impact sharing?
  - New OS technologies?
  - VDI farms? LAMP clusters?

- **Our goal:** Provide practical insights into these questions through a careful study of memory data

Outline

- Background and motivation
- Data collection and types of sharing
- Study of real-world sharing potential
- Study of the factors impacting sharing
- Conclusions

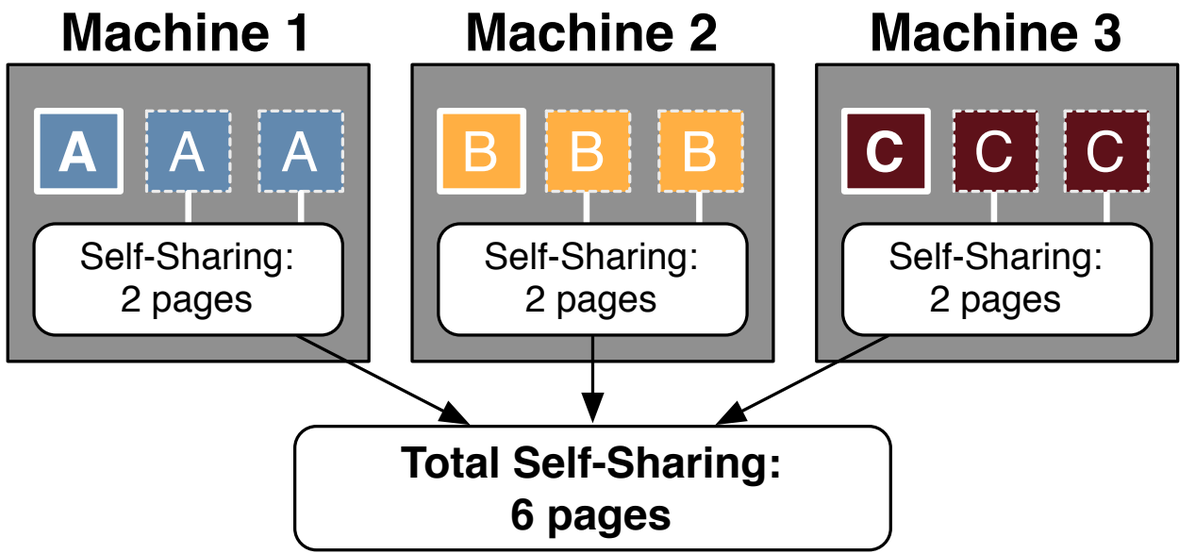
Data Collection

- Real-world memory traces
  - ~50 real machines (server/desktop mix)
  - Uncontrolled user workloads
  - Memory snapshots every 30 minutes

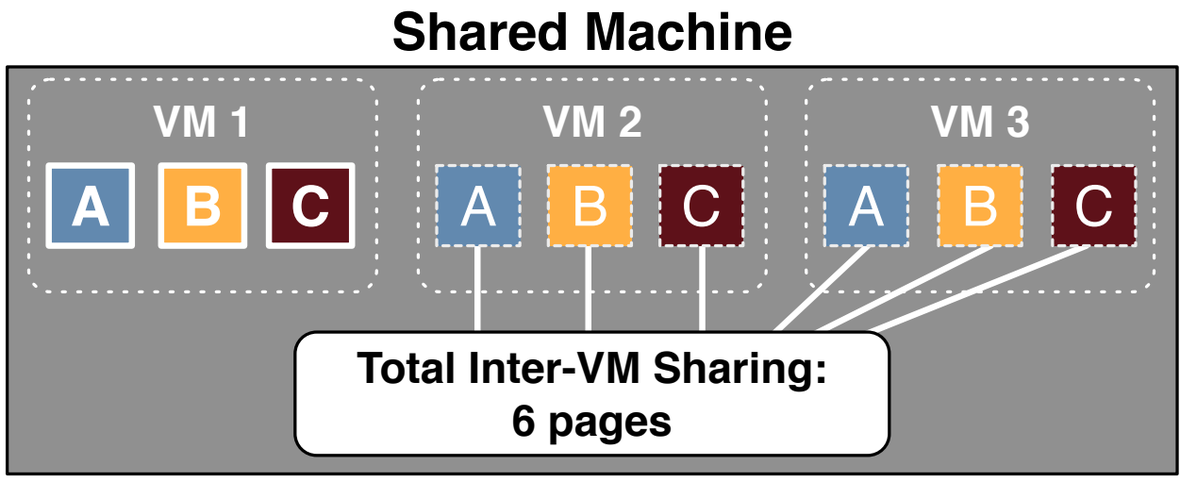
- Supplementary traces from controlled VMs
  - Mac/Win/Linux, mixed versions, 32/64 bit
  - 3 application setups per VM:
    - No workload (freshly booted)
    - Server apps (LAMP stack)
    - Desktop apps (office, browser, media player)

Types of Sharing

- **Self-sharing**: sharing within individual VMs
  - E.g., multiple zero pages



- **Inter-VM sharing**: sharing across multiple VMs
  - E.g., shared OS state

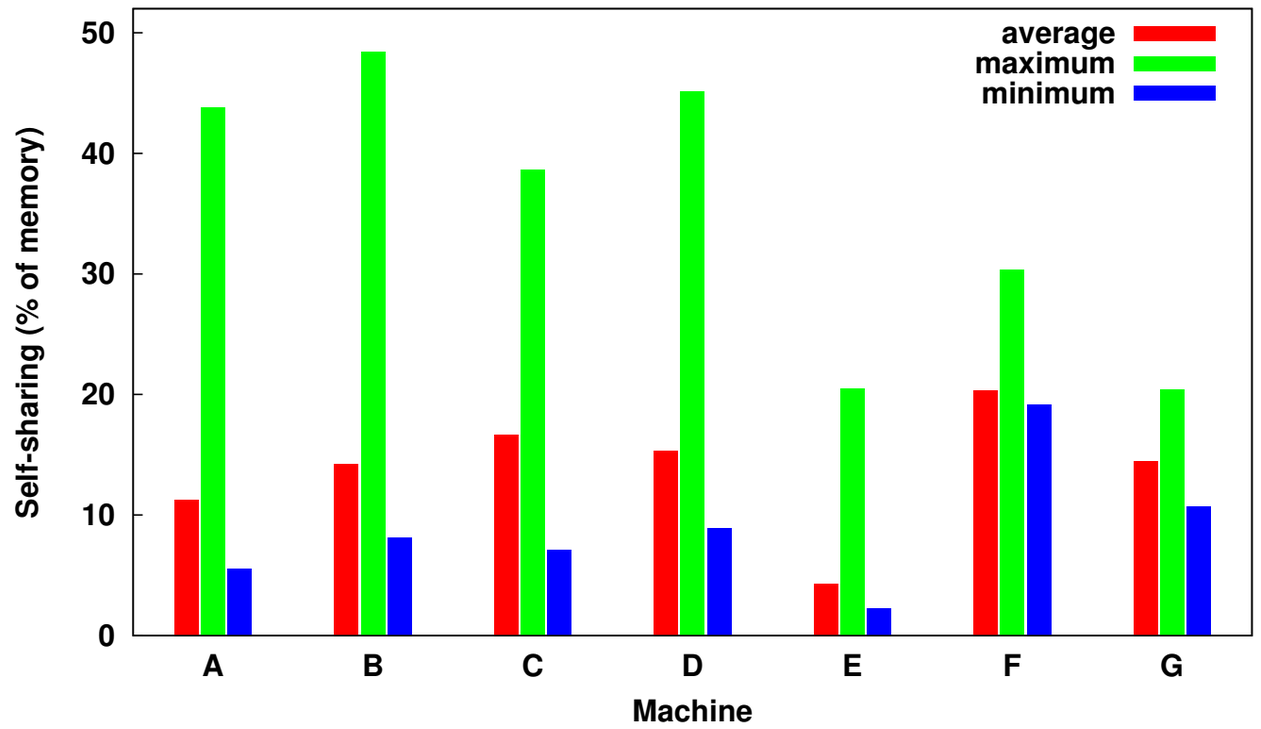


Outline

- Background and motivation
- Data collection and types of sharing
- Study of real-world sharing potential
- Study of the factors impacting sharing
- Conclusions

Self-Sharing in Real-World Traces

- Average sharing of **14%**
  - Excluding zero pages
- Peak sharing up to **50%**

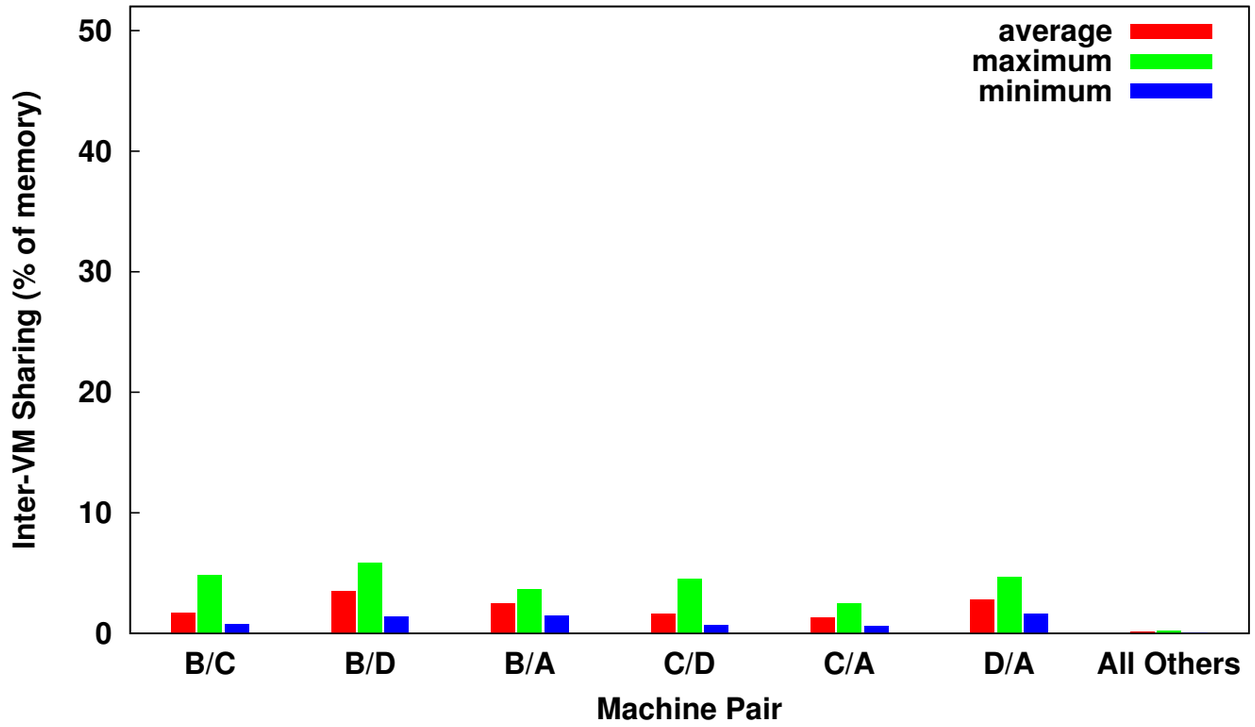


- Stable ‘baseline’ sharing of **8%**

- Significant (**~15%**) self-sharing potential observed

Inter-VM Sharing in Real-World Traces

- 'High' average sharing of just **2%**
- **<0.1%** sharing in 15 of 21 pairings



- In our traces, inter-VM sharing never above **6%**

- Observed minimal (**<2%**) inter-VM sharing potential

Real-World Trace Observations

- Typical **15%** possible sharing observed
  - Significant, but less than expected from synthetic workloads
- Most (**85+%**) sharing derived from self-sharing
  - What about collocating many VMs?
  - All 7 machines…still **80+%** from self-sharing
- Self-sharing doesn't require virtualization!
  - Could capture it within a VM or nonvirtualized host

- Self-sharing is significant, but what causes it?

Self-Sharing Case Study

- What causes self-sharing in a Linux desktop?
  - Looking at nonzero sharing

- Expanded version of Linux memory tracer
  - Track page contents and processes

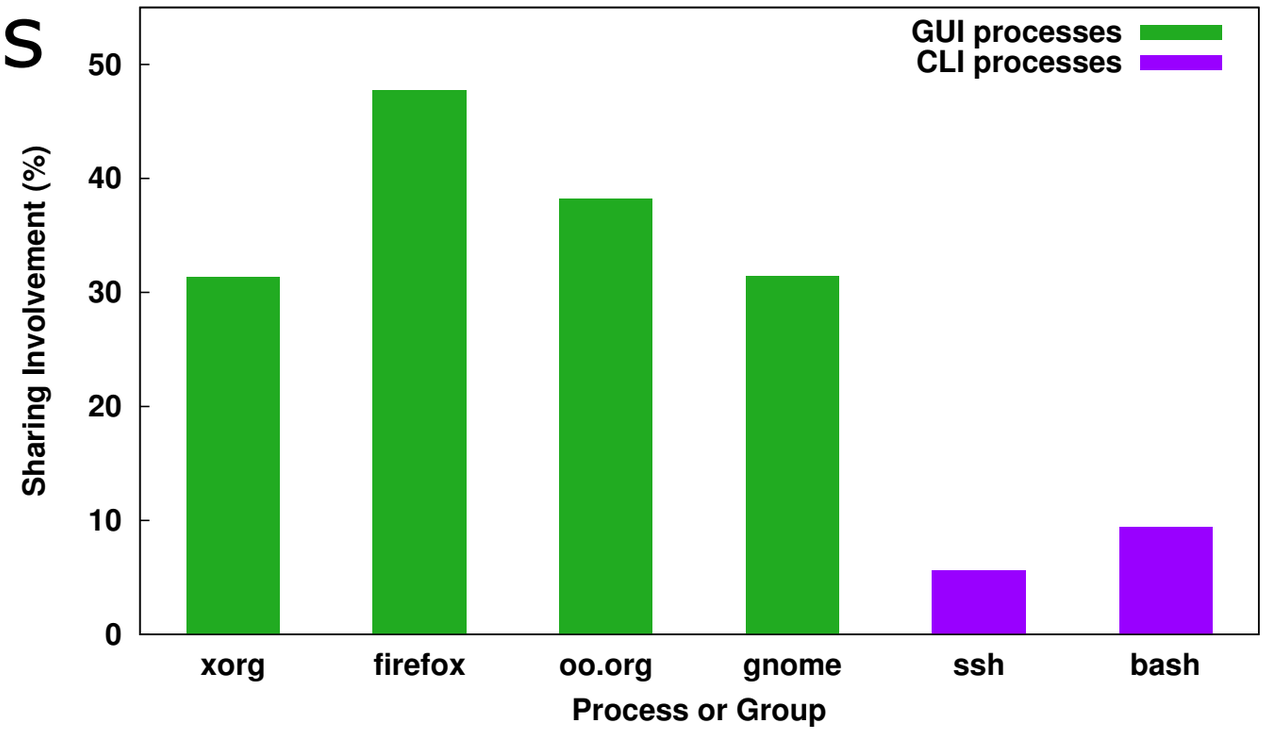
```
[libc-2.12.so 000b6000 r-xp]: sshd apache2
```

- Group sharing involvement (% of self-sharing) by content and process

Self-Sharing by Process

- **>30%** sharing processes GUI apps/libraries

- **<20%** sharing from other system processes

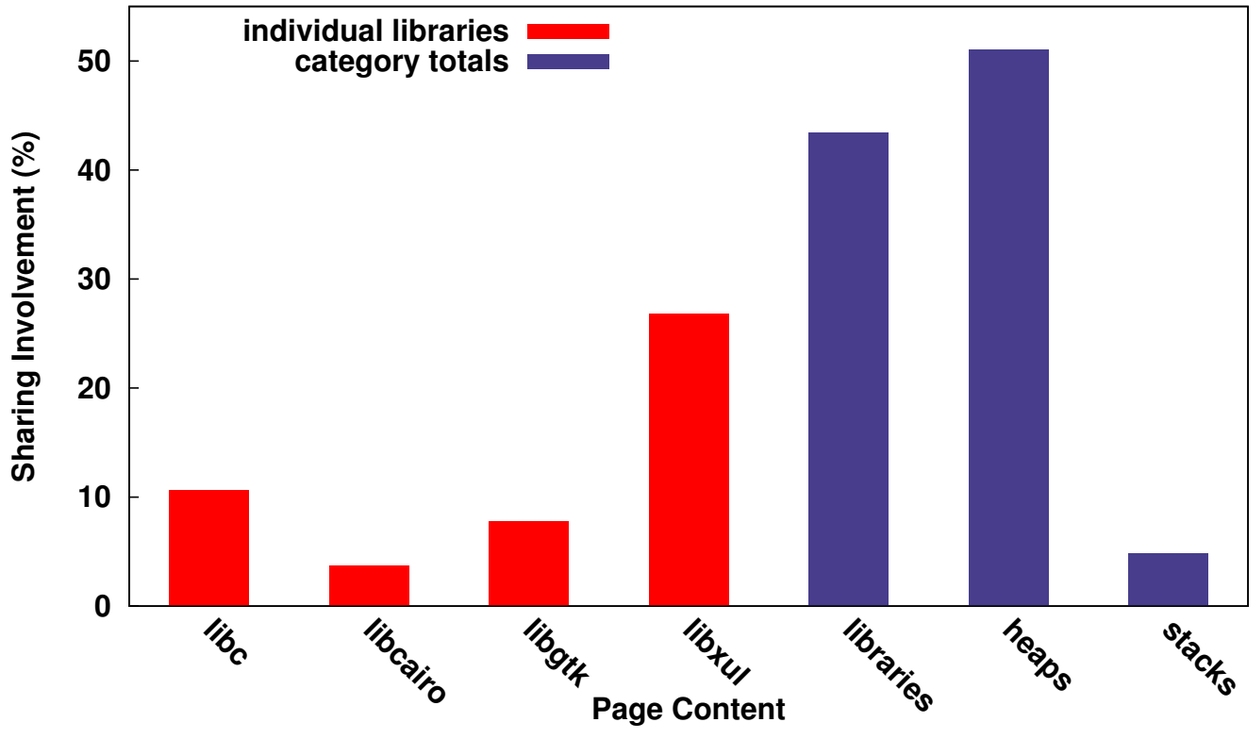


- Memory footprint likely dominated by GUI

- Process self-sharing resulting from user workload

Self-Sharing by Content

- **94%** sharing from libraries and heaps
- Possibly from recreated data structures



- **2.3 MB** sharing from single Xorg heap page (~600 copies)

- Duplicate data allocations evident in processes

Outline

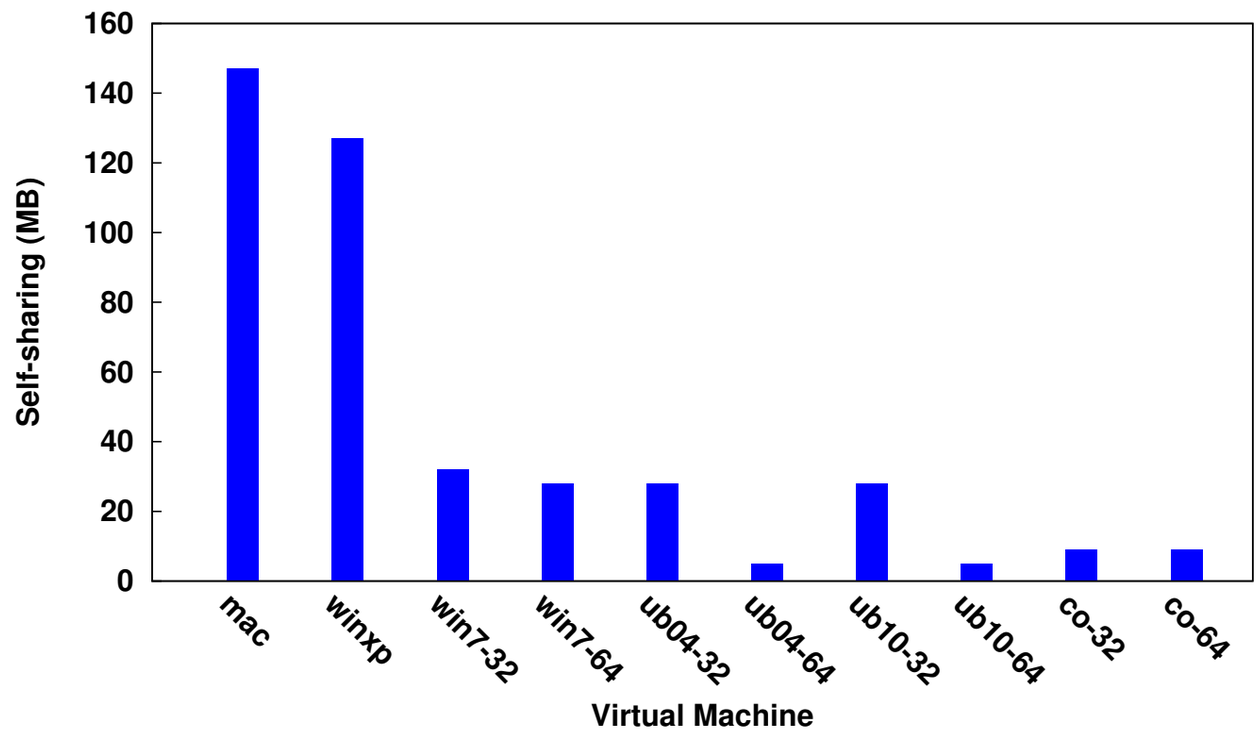
- Background and motivation
- Data collection and types of sharing
- Study of real-world sharing potential
- Study of the factors impacting sharing
- Conclusions

Factors Impacting Sharing

- How do various properties influence sharing?
- Operating system characteristics
  - Family (e.g., Linux or Windows)
  - Version (e.g., Windows XP/7, Ubuntu 10.04/10.10)
  - Architecture (x86 or x64)
- Application setup (LAMP and VDI setups)
- Sharing granularity (number of pages per chunk)
- New OS technologies (e.g., ASLR)

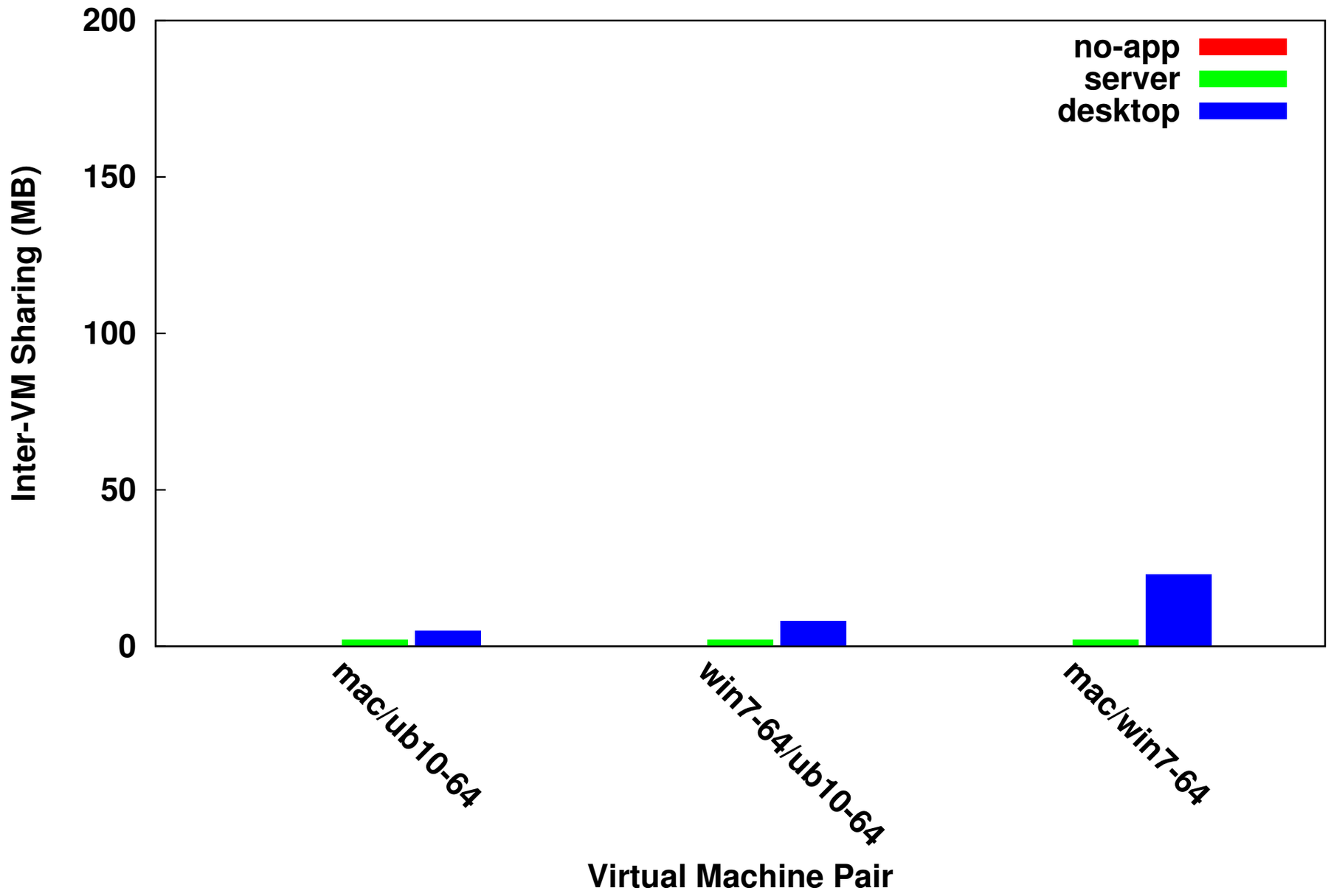
Self-Sharing Across VMs

- **~100 MB** differences between OS families, major versions (XP/7)
- **<20 MB** differences between minor versions, architectures

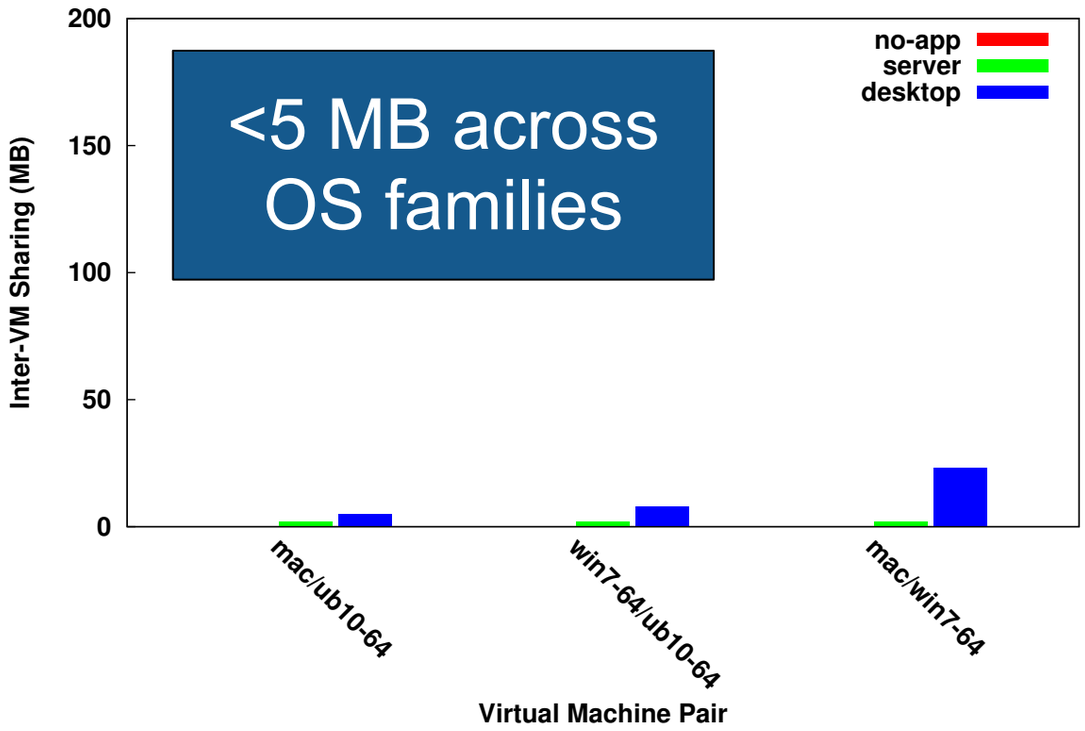


- Large self-sharing variations between 'base' OSes

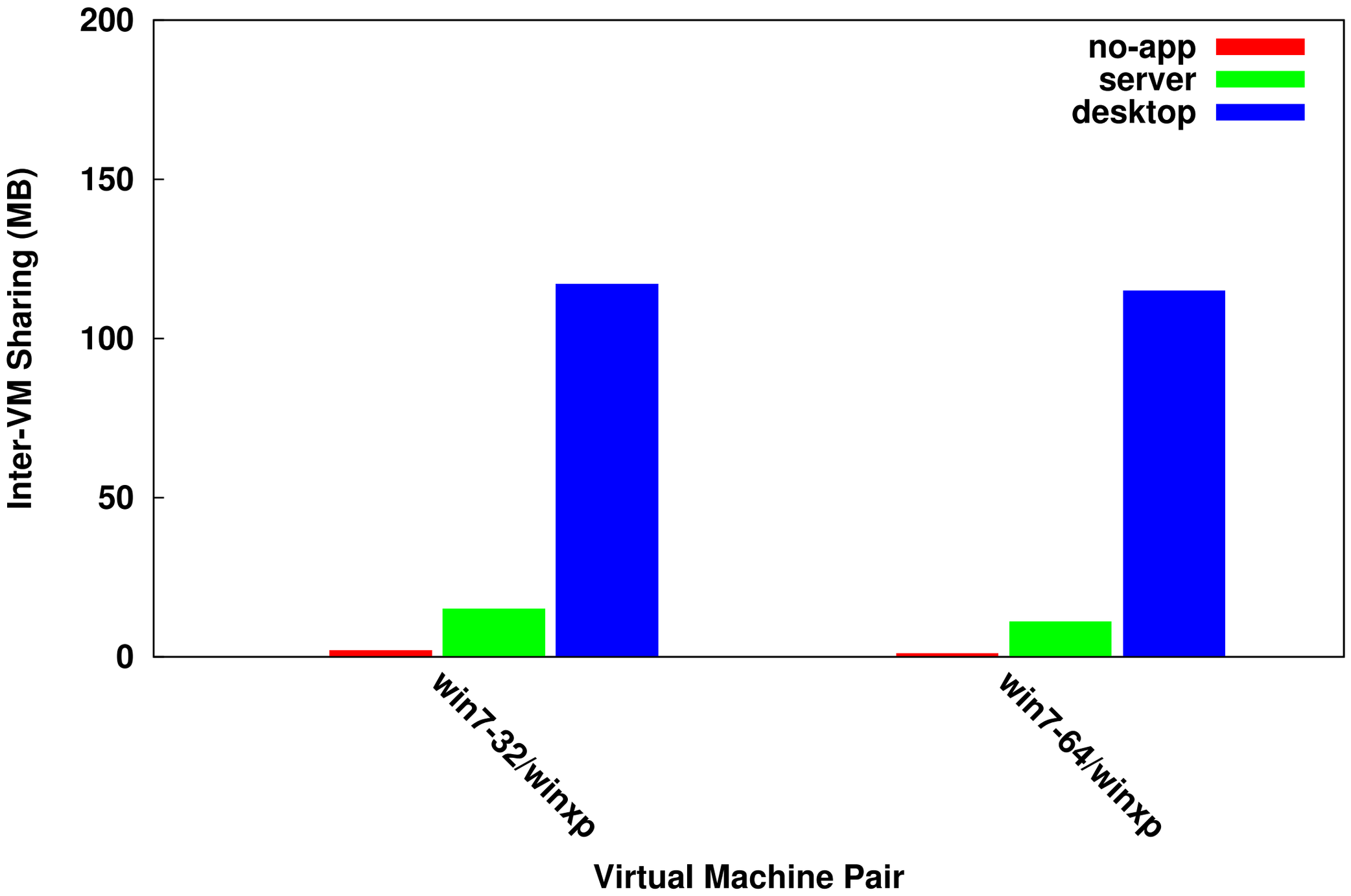
Sharing Across VMs



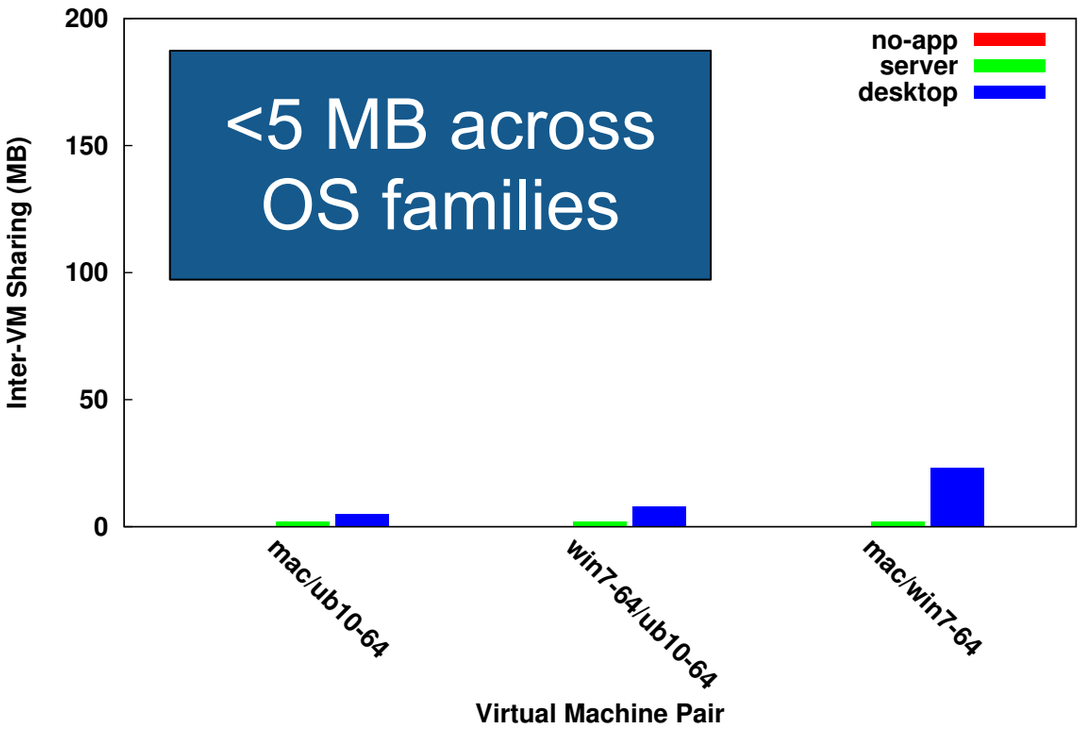
Sharing Across VMs

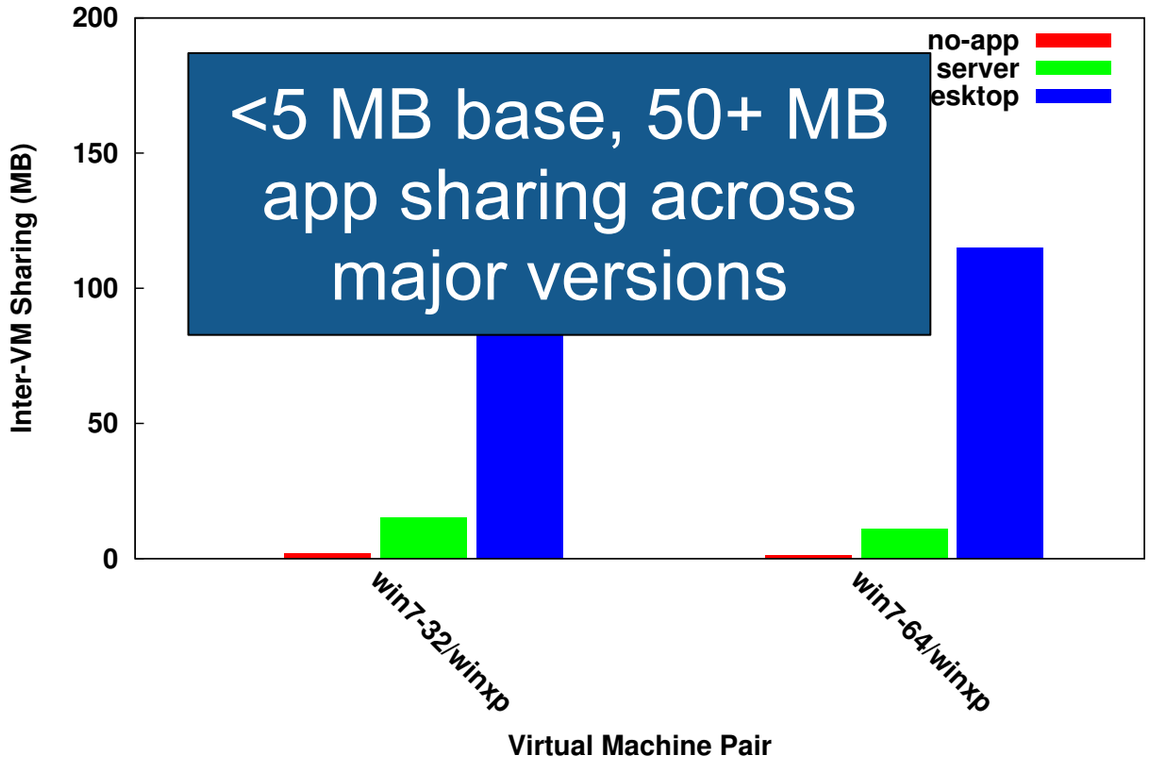


Sharing Across VMs

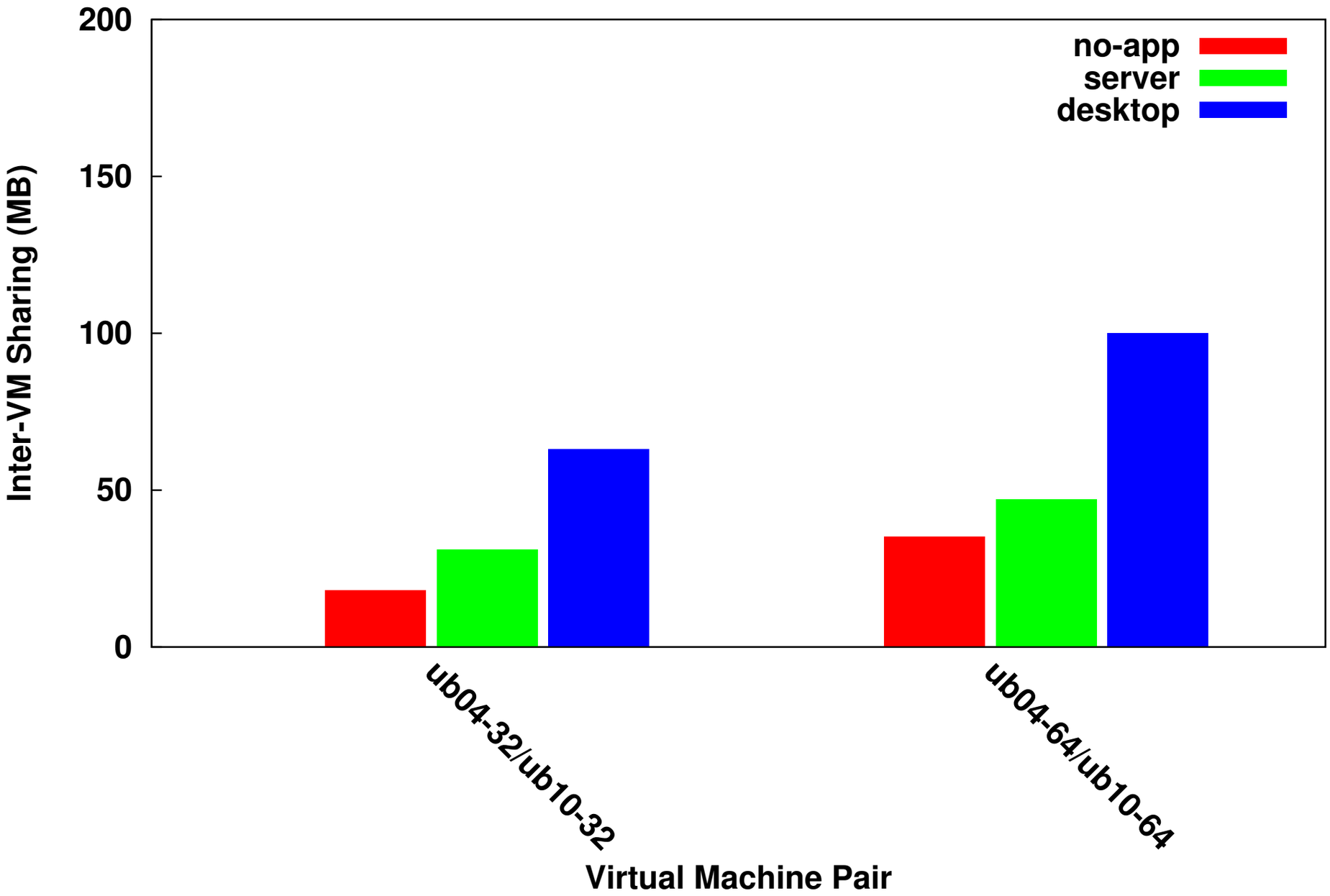


Sharing Across VMs

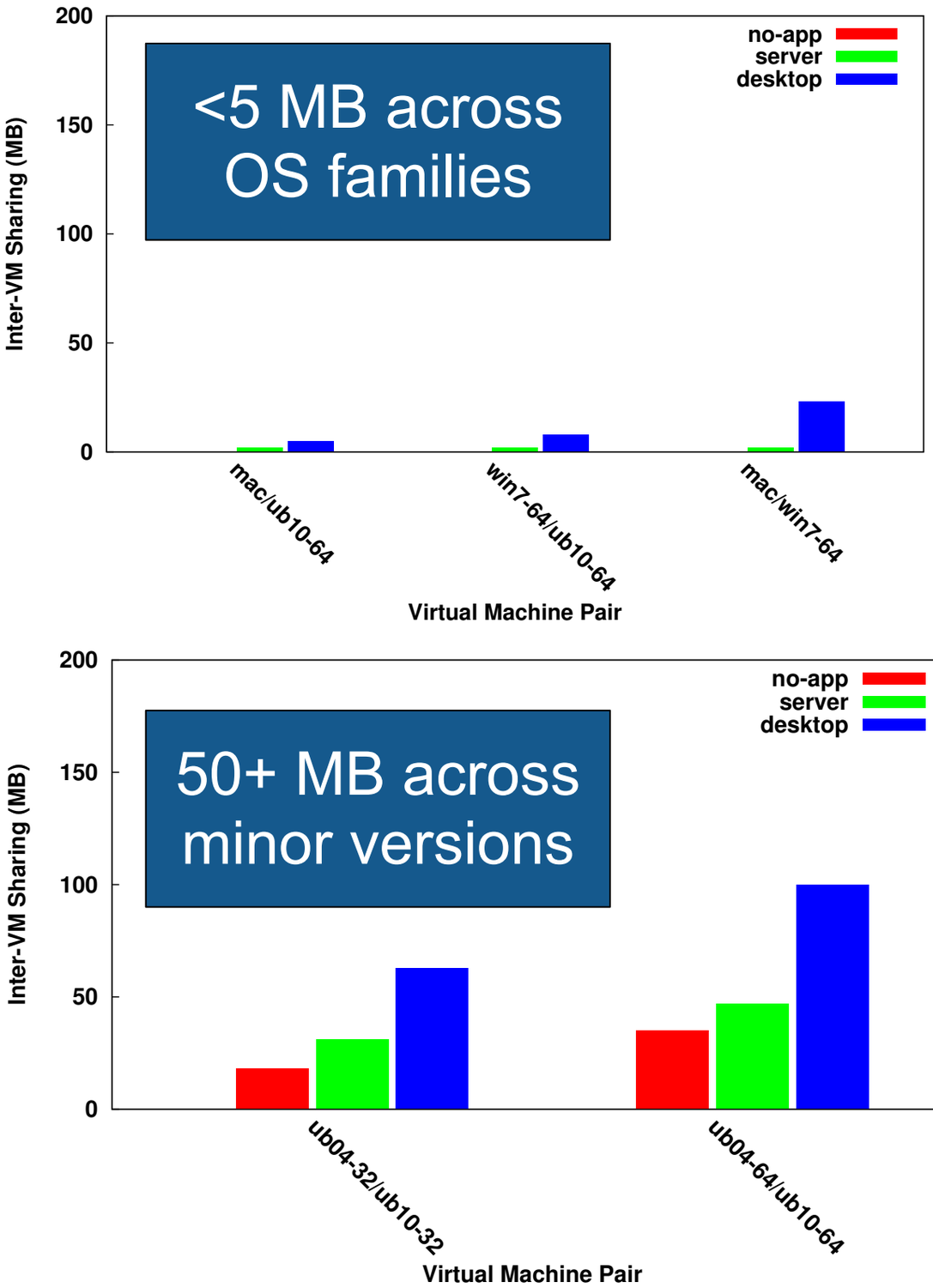


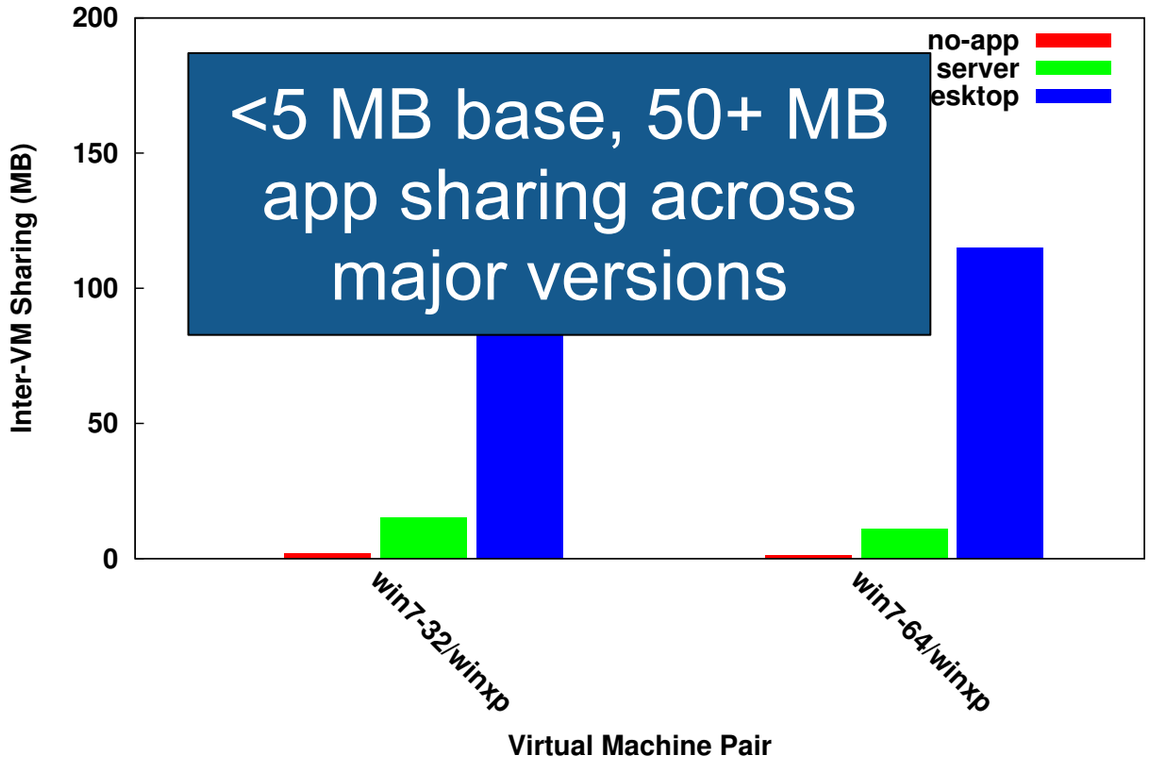


Sharing Across VMs

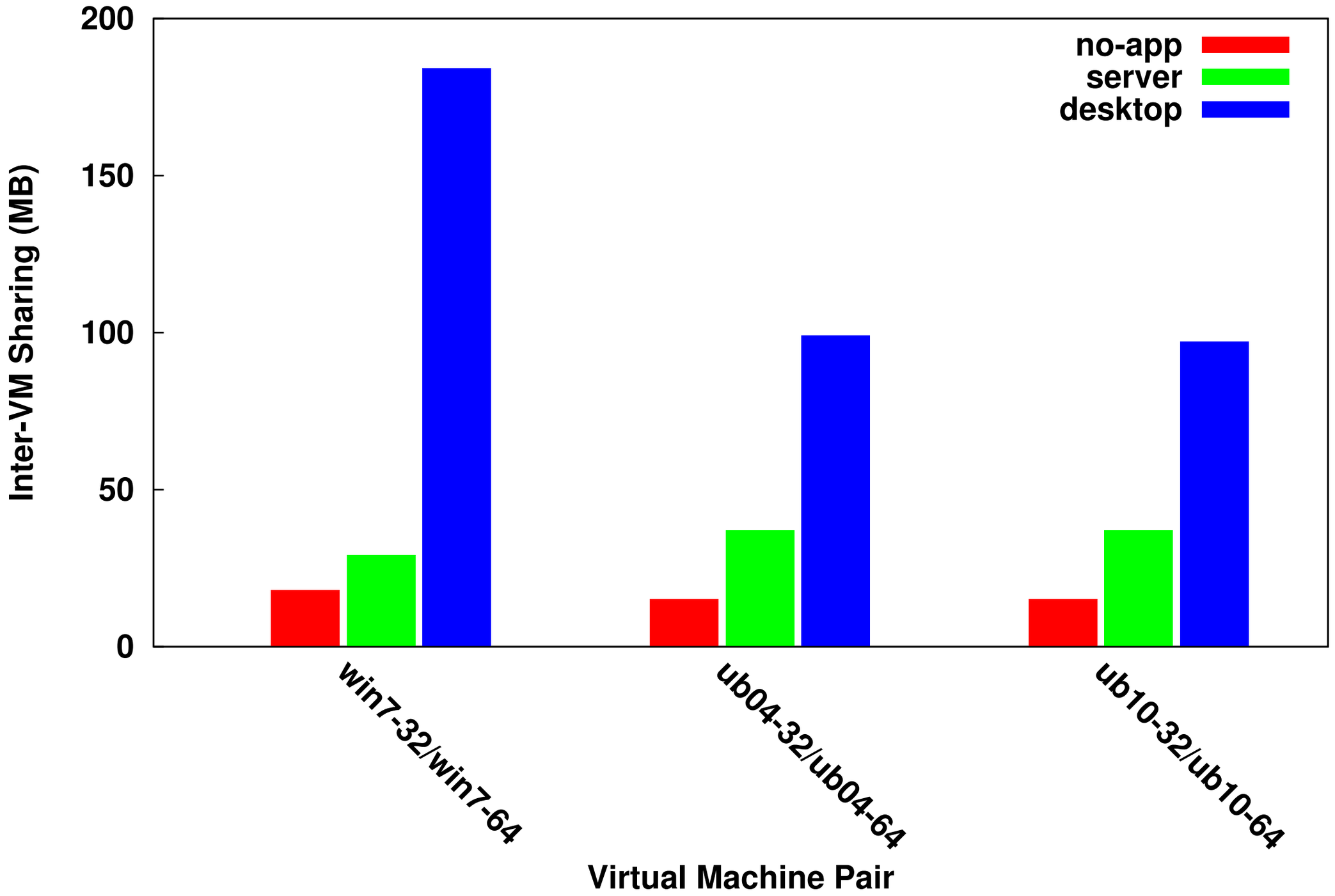


Sharing Across VMs

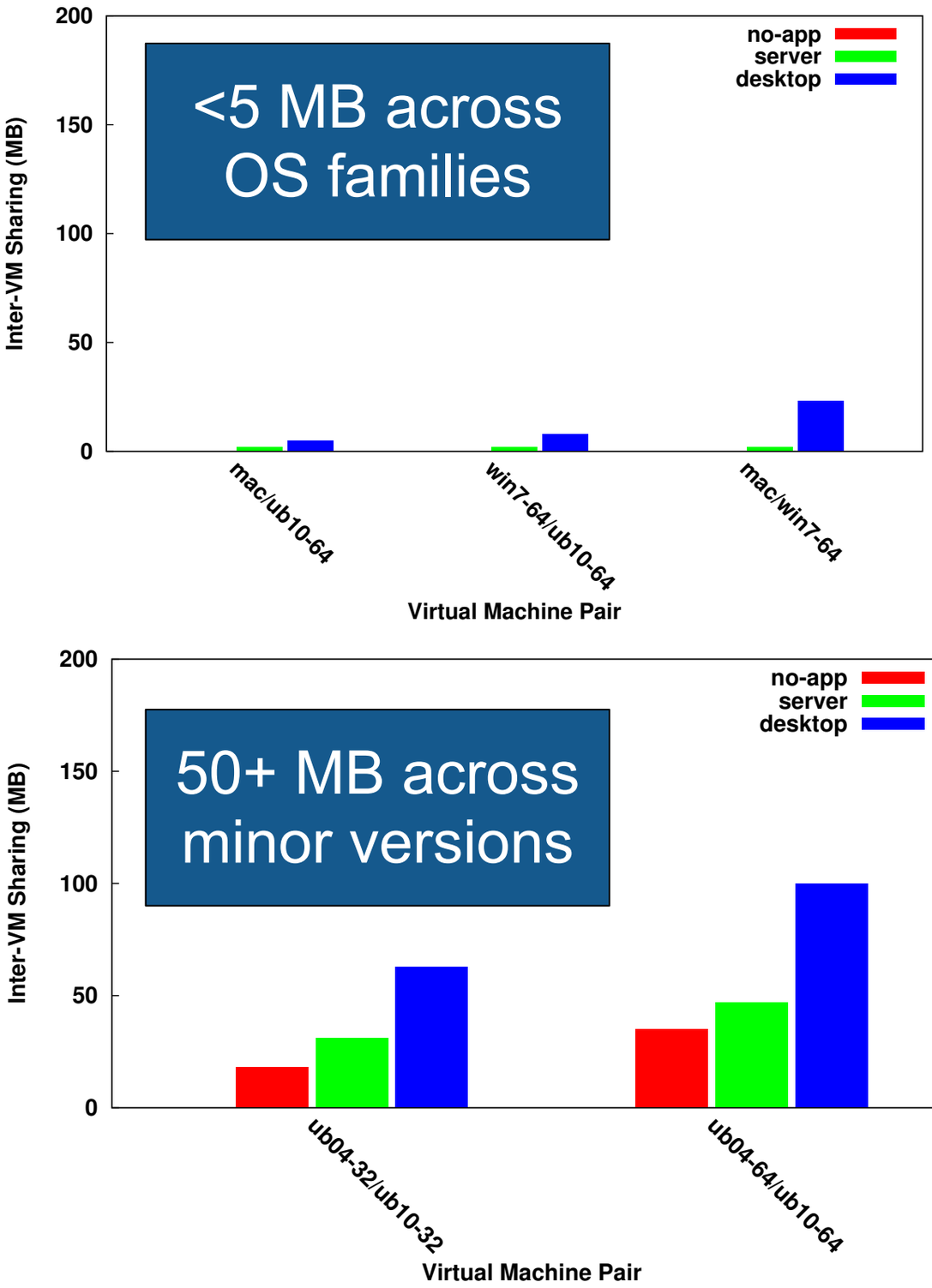


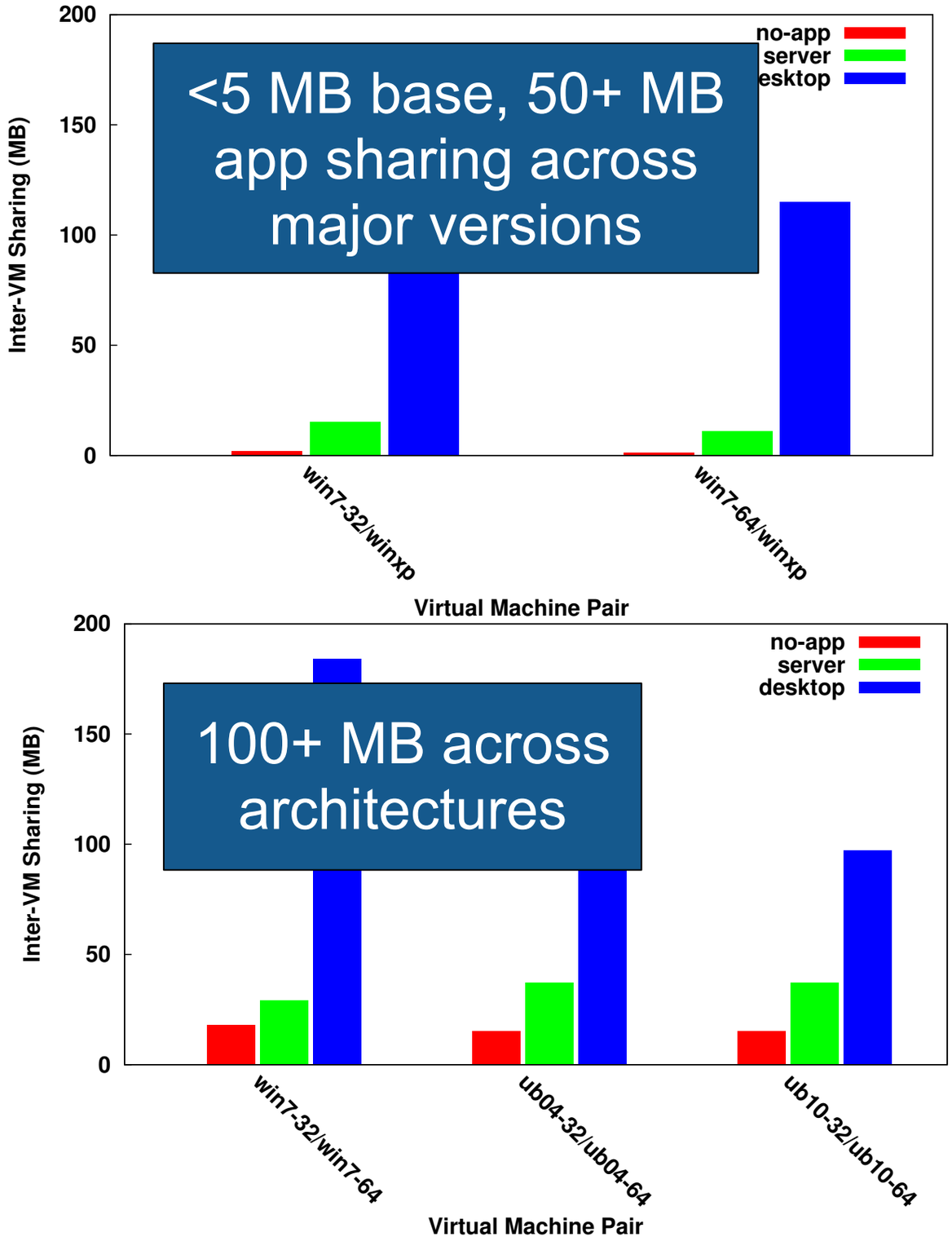


Sharing Across VMs

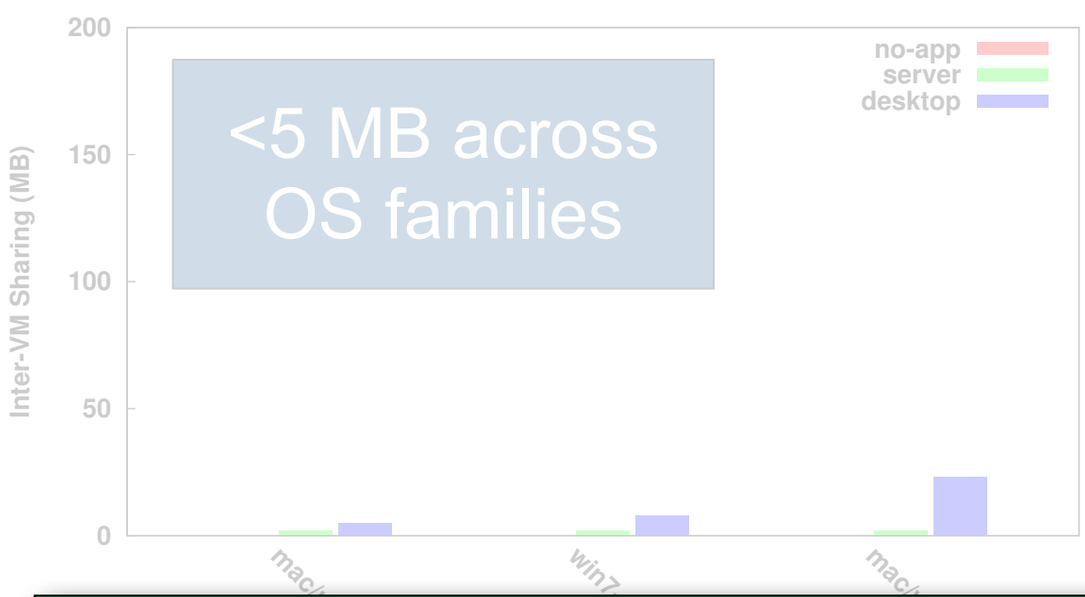


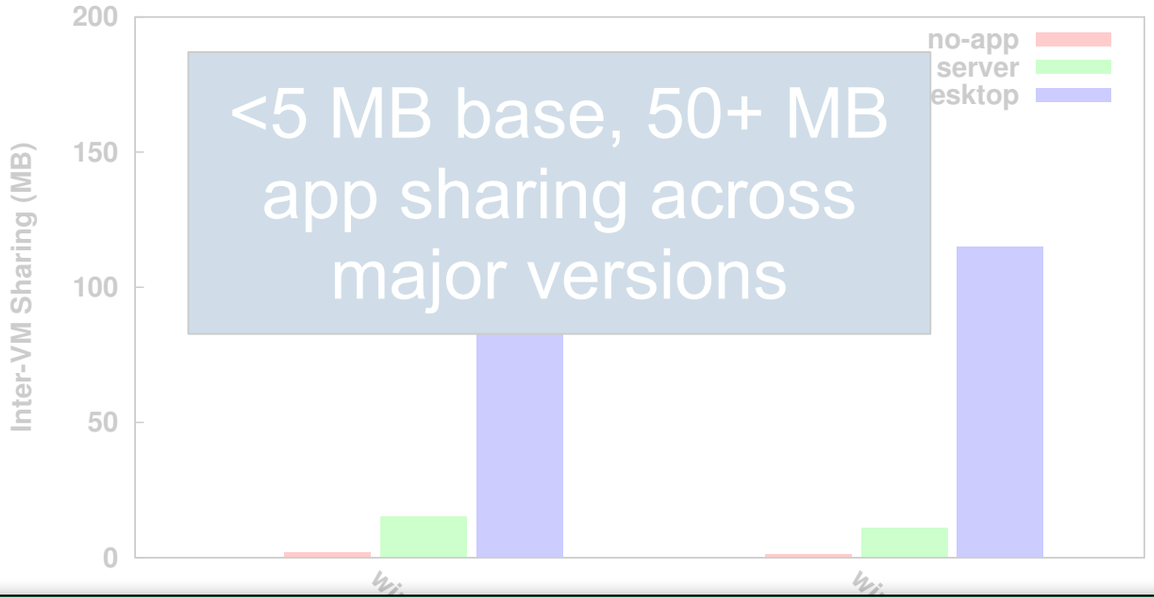
Sharing Across VMs



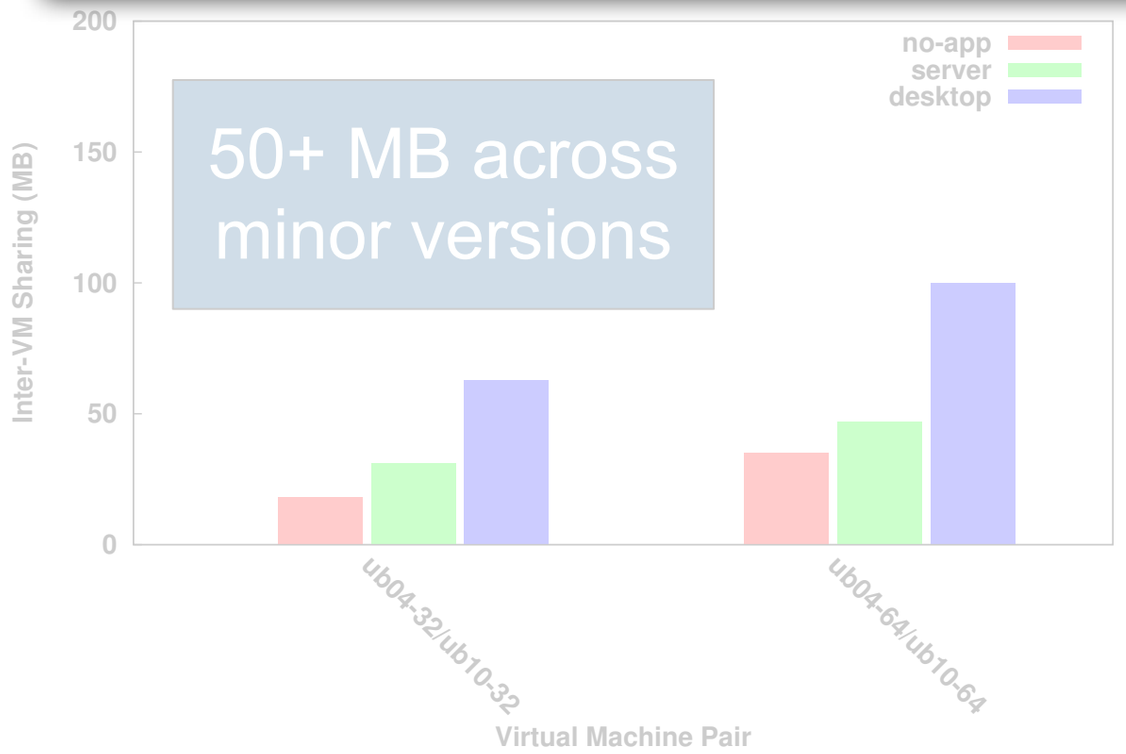


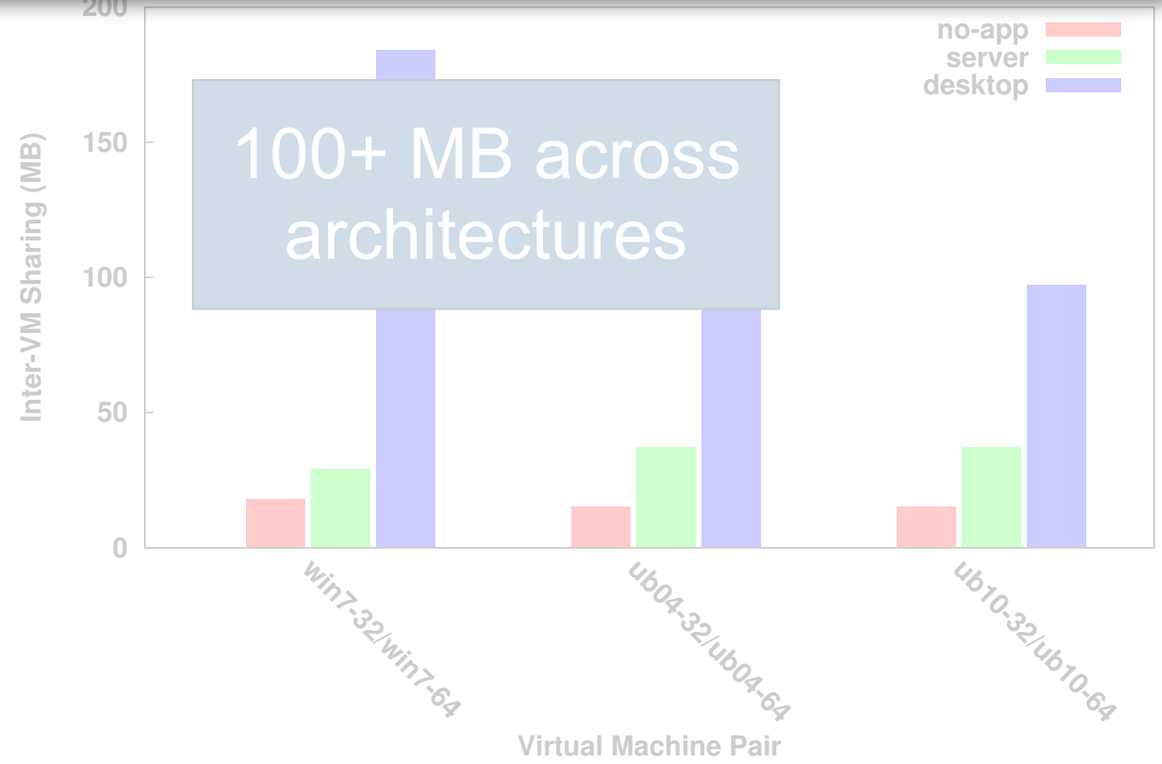
Sharing Across VMs





- Hierarchy: family, applications, version, architecture





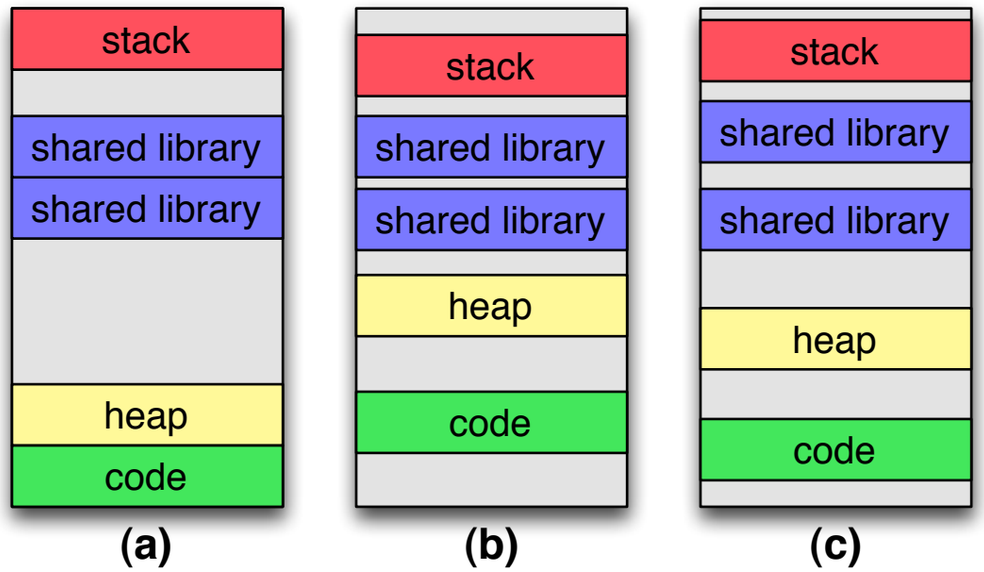
Sharing Granularity

- Share memory chunks of size k (≠1) pages
- Only even page divisions provide decent returns
- Diminishing benefits from smaller chunk sizes

- Tradeoff between overhead and sharing potential

Address Space Layout Randomization

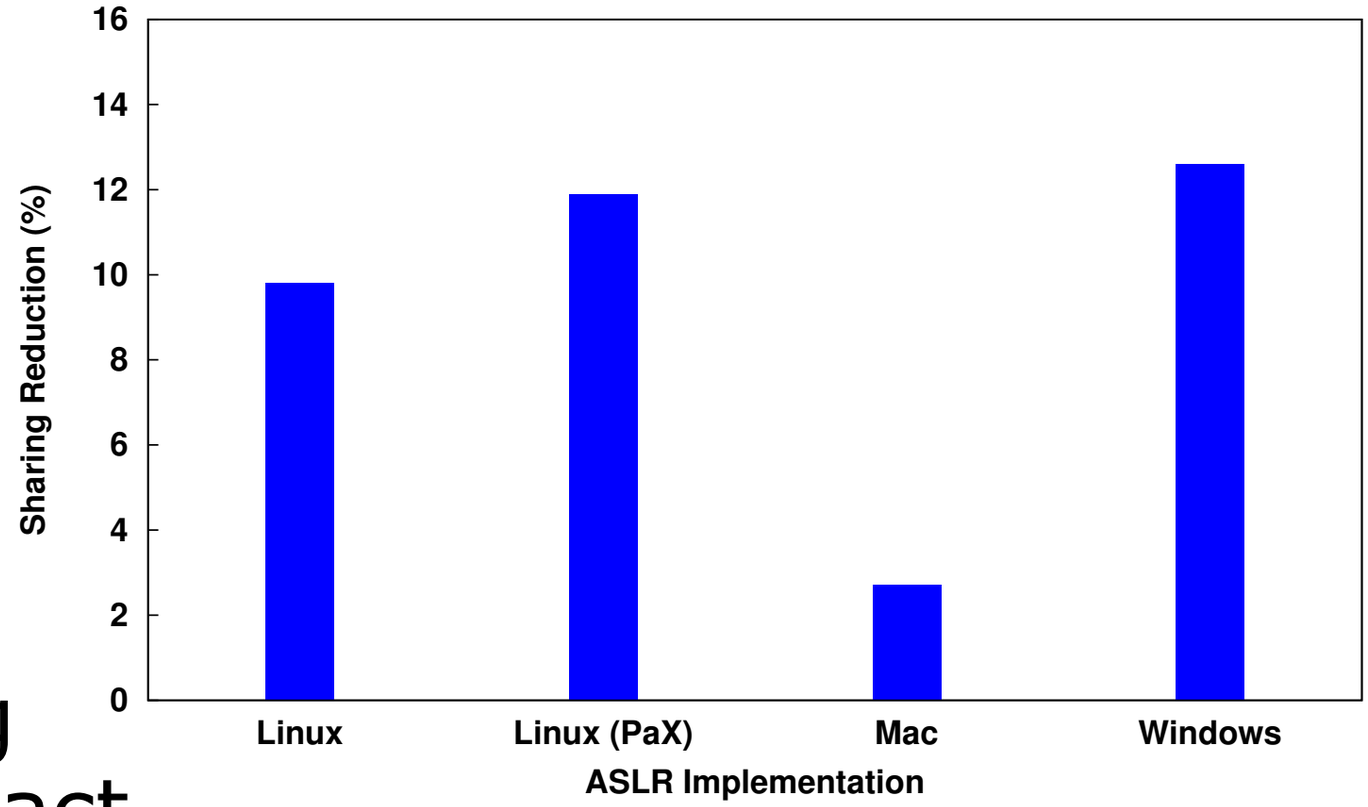
- ASLR scrambles memory to improve system security
  - libraries, code, stack, heap, ...
- Does ASLR have a negative impact on memory sharing?



- Impact of 4 ASLR implementations:
  - Linux: mainline (2.6.32) and PaX
  - Windows 7 (SP1)
  - Mac OS X (Lion)
- Desktop applications with and without ASLR

Sharing Impact of ASLR

- **>10%** reduction in three of four cases



- ‘Better’ (PaX) sharing in Linux worsens impact

- ASLR doesn’t prevent sharing but does reduce it

Sharing Factor Observations

- Hierarchy with respect to sharing potential
  - OS family, application setup, OS version, OS architecture
- Platform homogeneity
  - Minimal sharing across heterogeneous systems
  - Significant gains in homogeneous deployments (but still modest absolute levels)
- Finer-grained sharing may be leveraged to improve sharing potential
- OS improvements like ASLR may reduce sharing

Conclusions

- Study into practical issues of page sharing
  - Examined real-world machines and specific sharing scenarios

- Observed real-world sharing around **15%**
  - Significant, but less than expected
  - Largely self-sharing, for which **no virtualization needed**

- Studied a variety of factors impacting sharing
  - Key role of platform homogeneity
  - Varying impact of modifying OS characteristics and applications
  - New technologies may change the impact of sharing

Questions?
sbarker@cs.umass.edu