Practicality and Feasibility of Improving Linux Container Utilization with Task Rebalancing Strategy

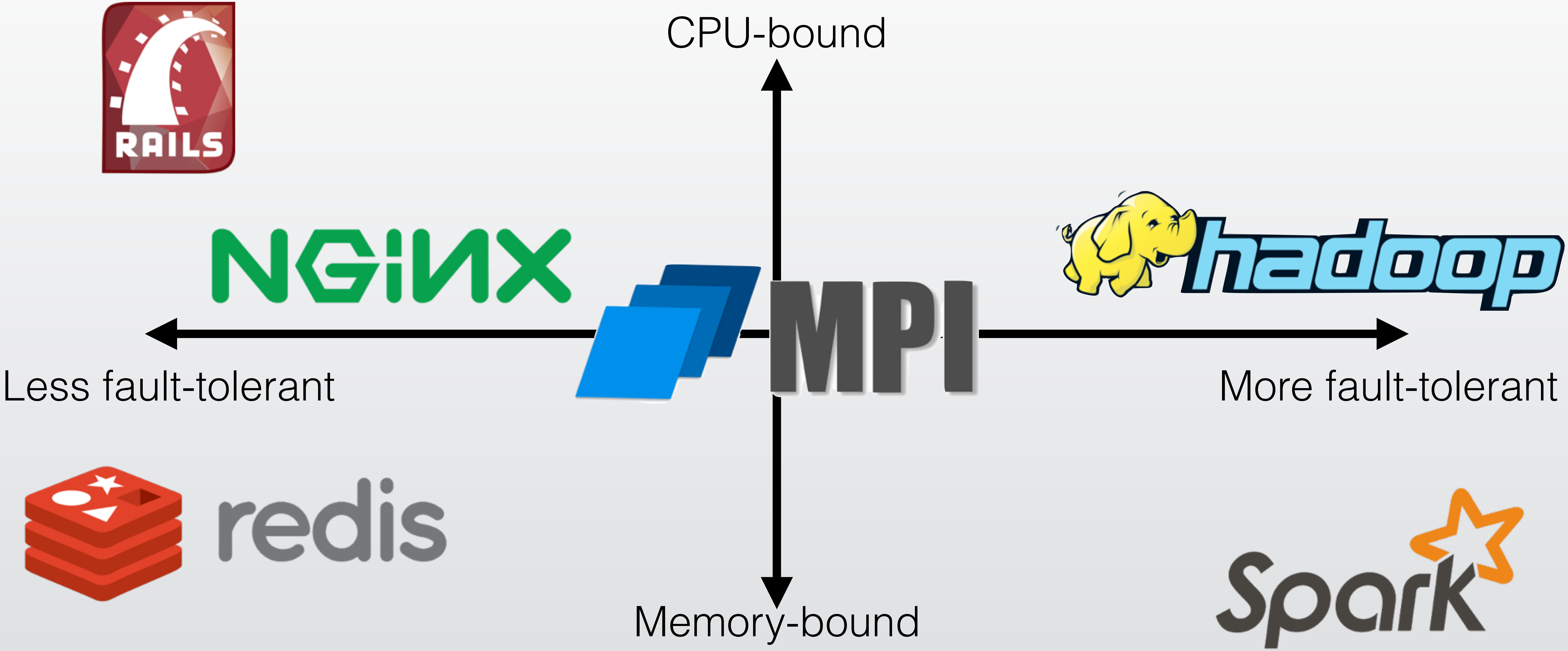
Pongsakorn U-chupala
Nara Institute of Science and Technology

PRAGMA30 | Manila Philippines

HPC is everywhere

NGINX
Spark
hadoop
redis
RAILS
MPI

HPC workloads are diverse



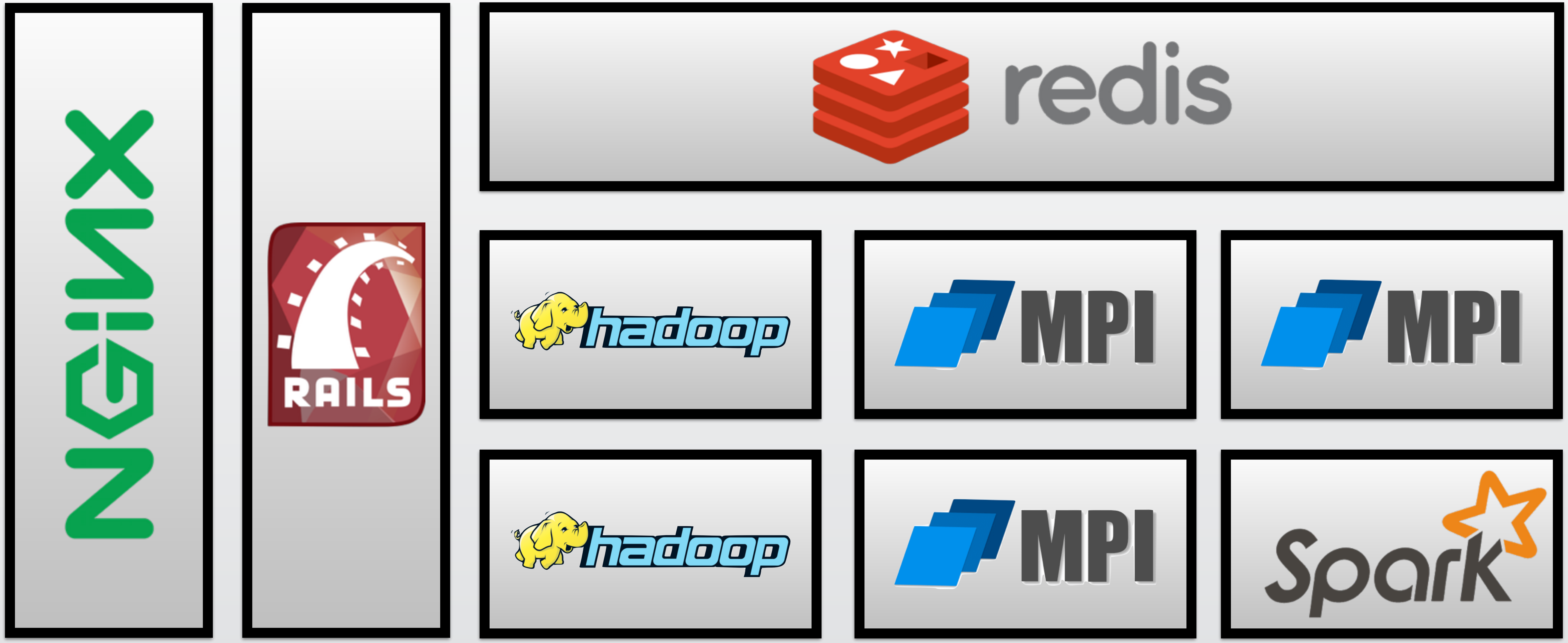
CPU-bound

Memory-bound

Less fault-tolerant

More fault-tolerant

Resource sharing with VM is cost effective



PROBLEM: Scheduling is inefficient

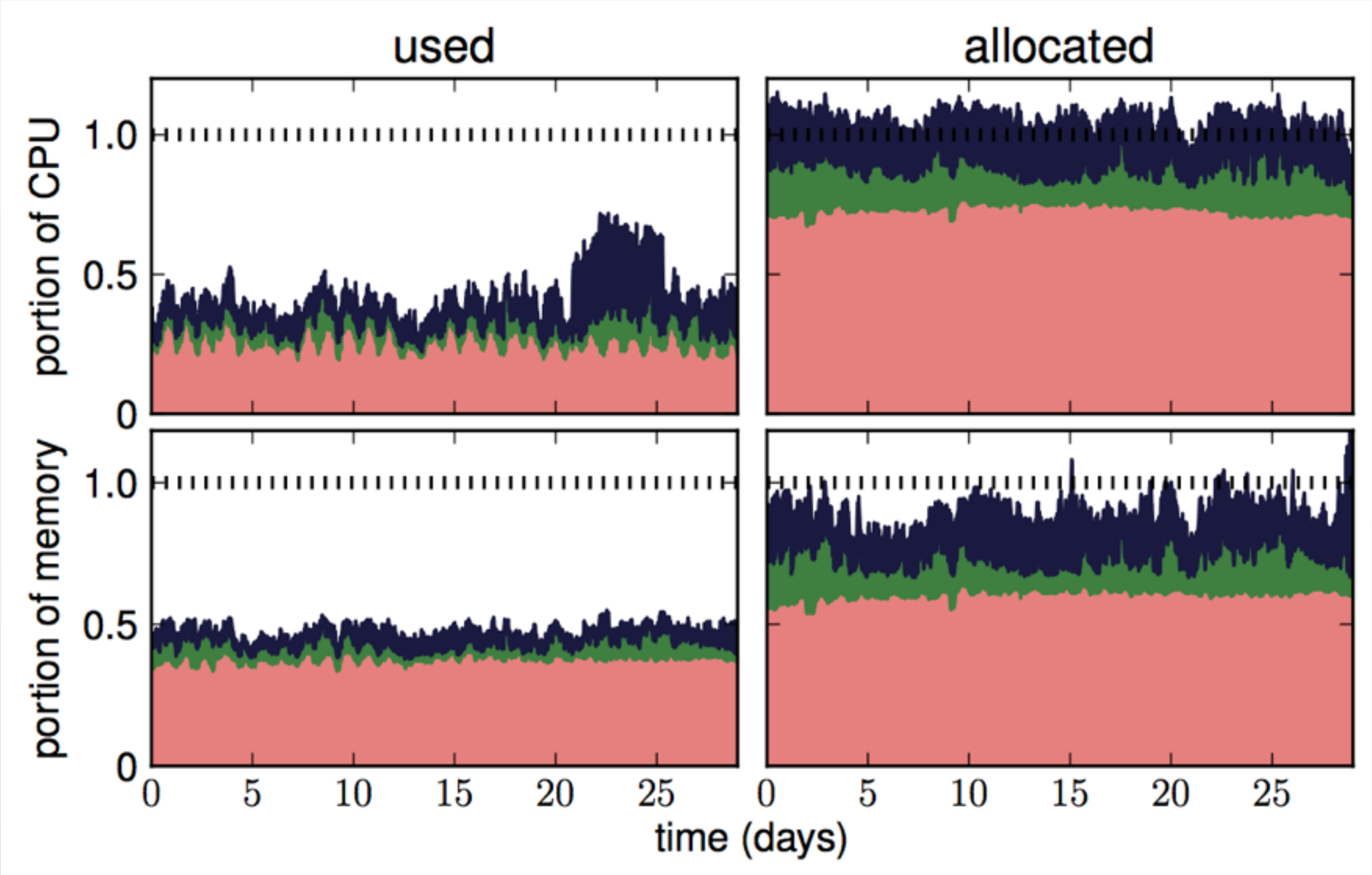
Moving Hourly Average
from Google Cluster Data
(color indicates priority)

Utilization

Resource Requests

CPU

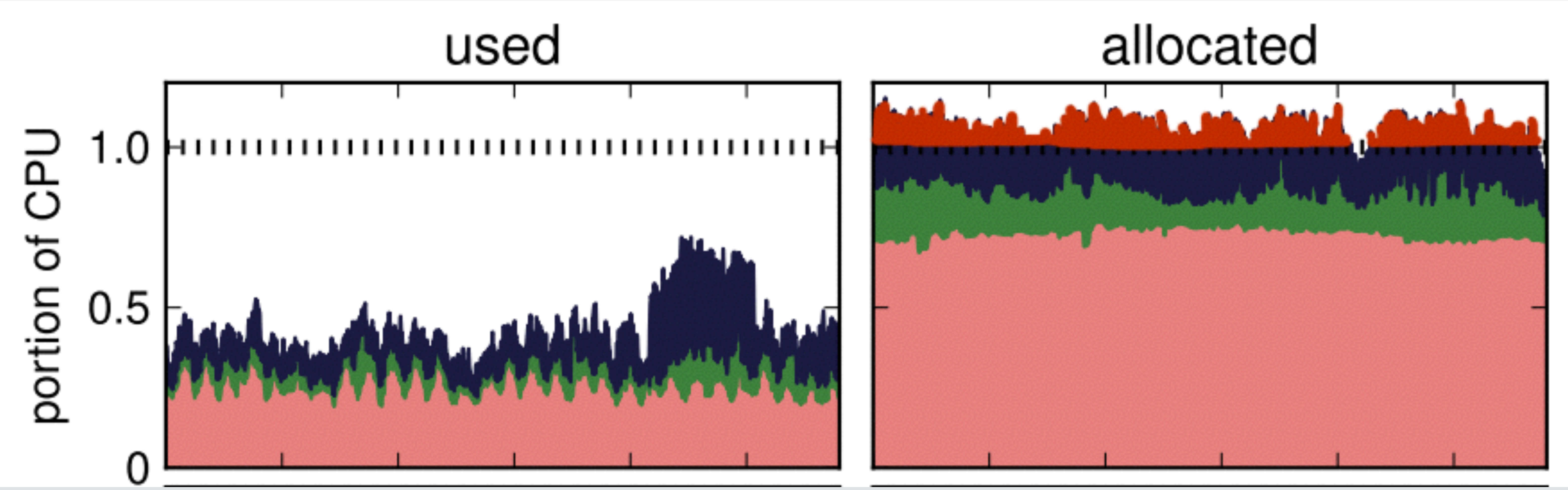
Memory



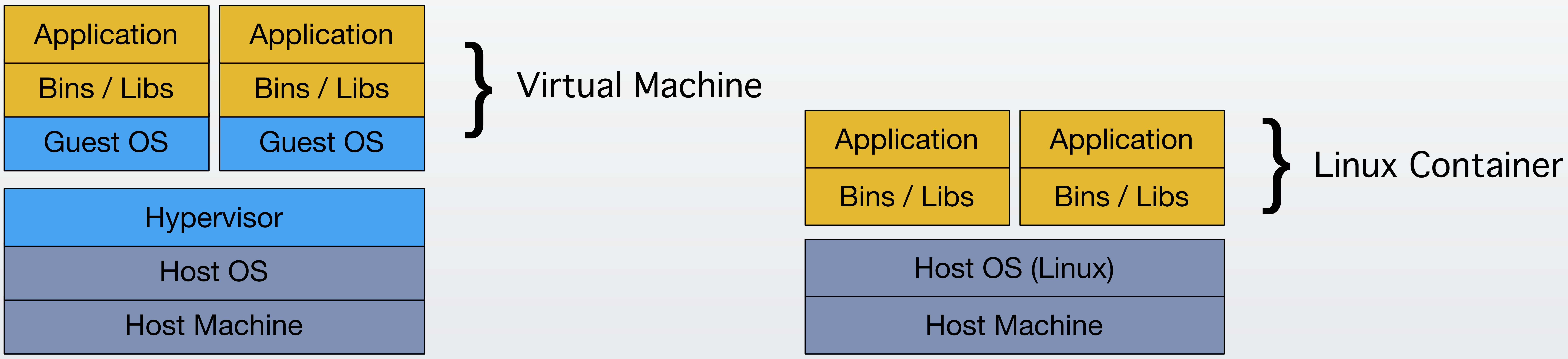
C. Reiss, A. Tumanov, G. R. Ganger, R. H. Katz, and M. a. Kozuch, “Heterogeneity and dynamicity of clouds at scale,” Proc. Third ACM Symp. Cloud Comput. - SoCC ’12, pp. 1–13, 2012.

Overcommitting

- Allocating more resource than available => Increase chance of task failure
- **Overcommit factor:** How much over-commit is allowed? (example: 1.2x of available resources)
  - Too high => Increase task failure rate
  - Too low => Resources are underutilized



Advent of Linux Container



Application | Application
Bins / Libs | Bins / Libs
Guest OS | Guest OS
Hypervisor
Host OS
Host Machine

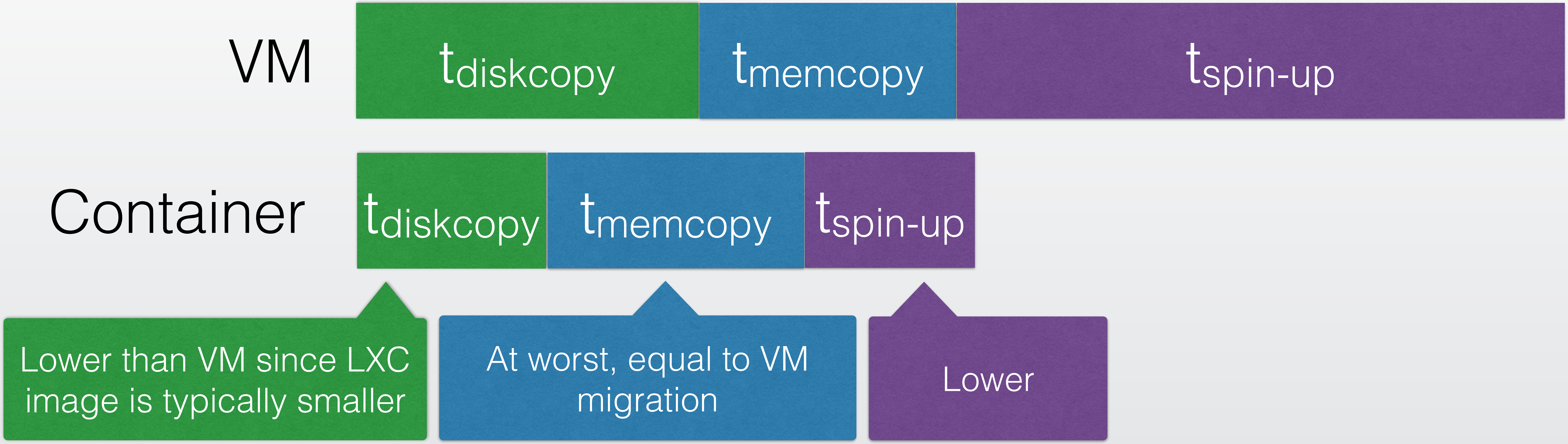
} Virtual Machine

Application | Application
Bins / Libs | Bins / Libs
Host OS (Linux)
Host Machine

} Linux Container

Less overhead → Lower spin-up time

Migration Time



VM: $t_{diskcopy}$ $t_{memcopy}$ $t_{spin\text{-}up}$

Container: $t_{diskcopy}$ $t_{memcopy}$ $t_{spin\text{-}up}$

Lower than VM since LXC image is typically smaller

At worst, equal to VM migration

Lower

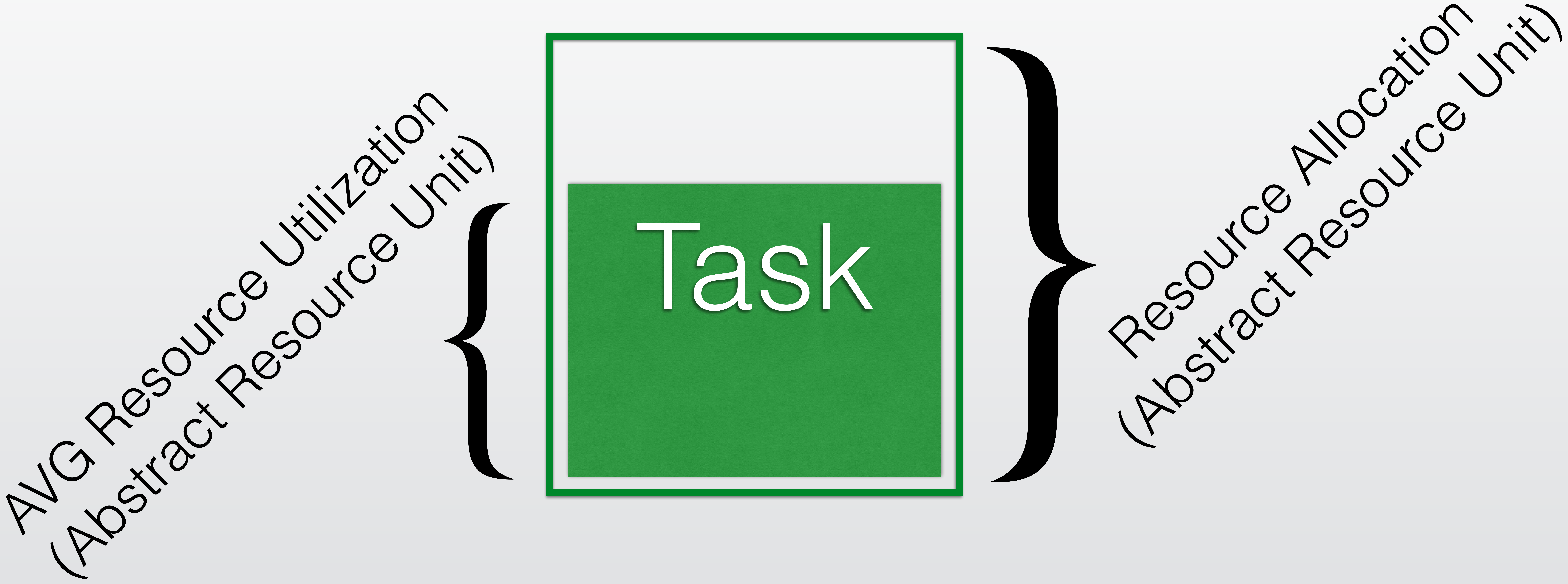
Faster migration → Enabling task micromanagement

PROPOSAL: Task Rebalancing

- Real-time host load-balancing
  - Increase optimal overcommit factor => Increase utilization
- Minimal interference to the scheduler and scheduled tasks
- Easier to explain with example

Example

Task

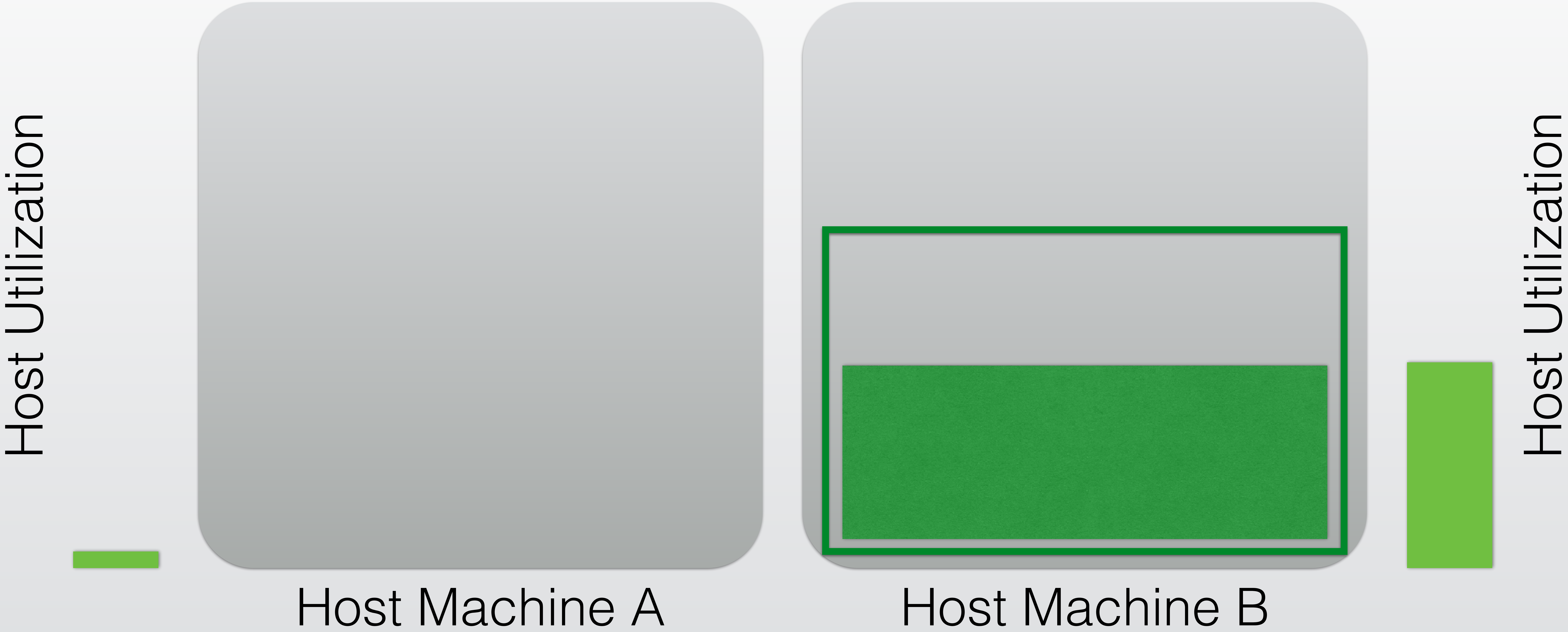


AVG Resource Utilization
(Abstract Resource Unit)

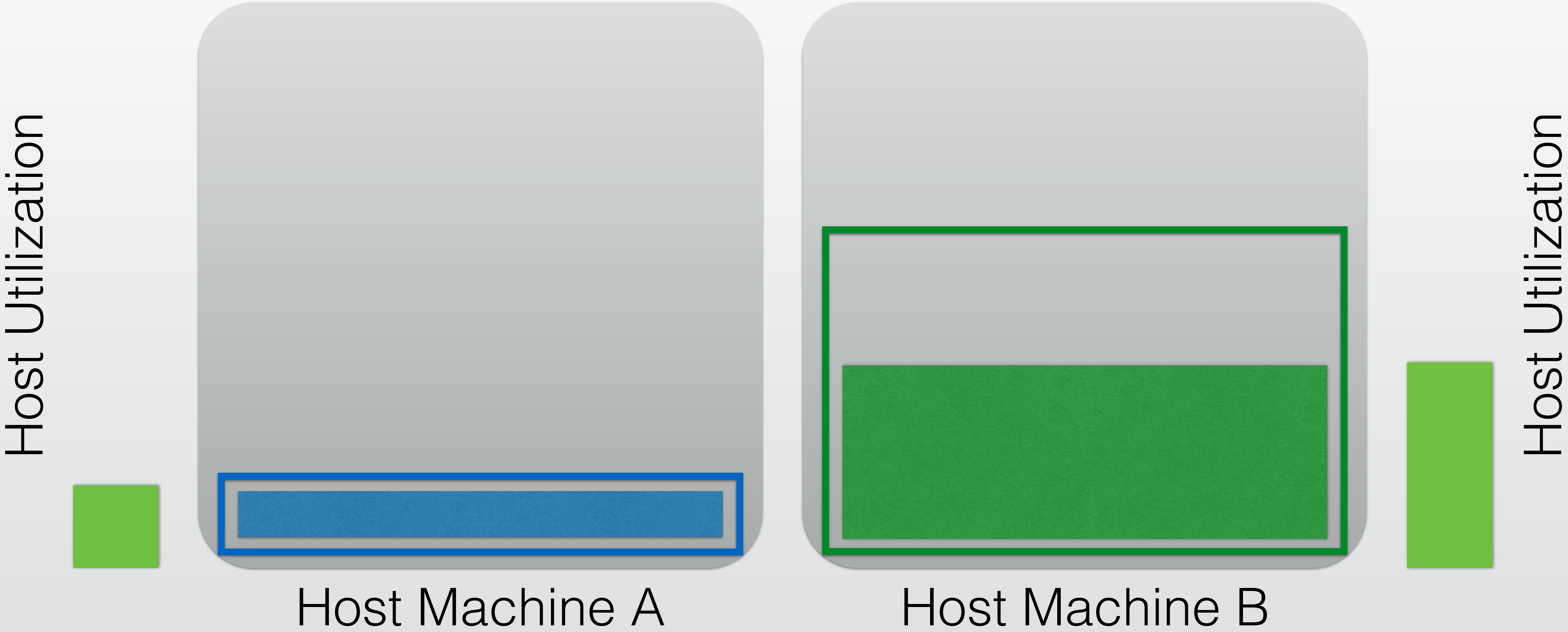
Task

Resource Allocation
(Abstract Resource Unit)

Scheduling


Host Utilization
Host Machine A
Host Machine B
Host Utilization

Scheduling



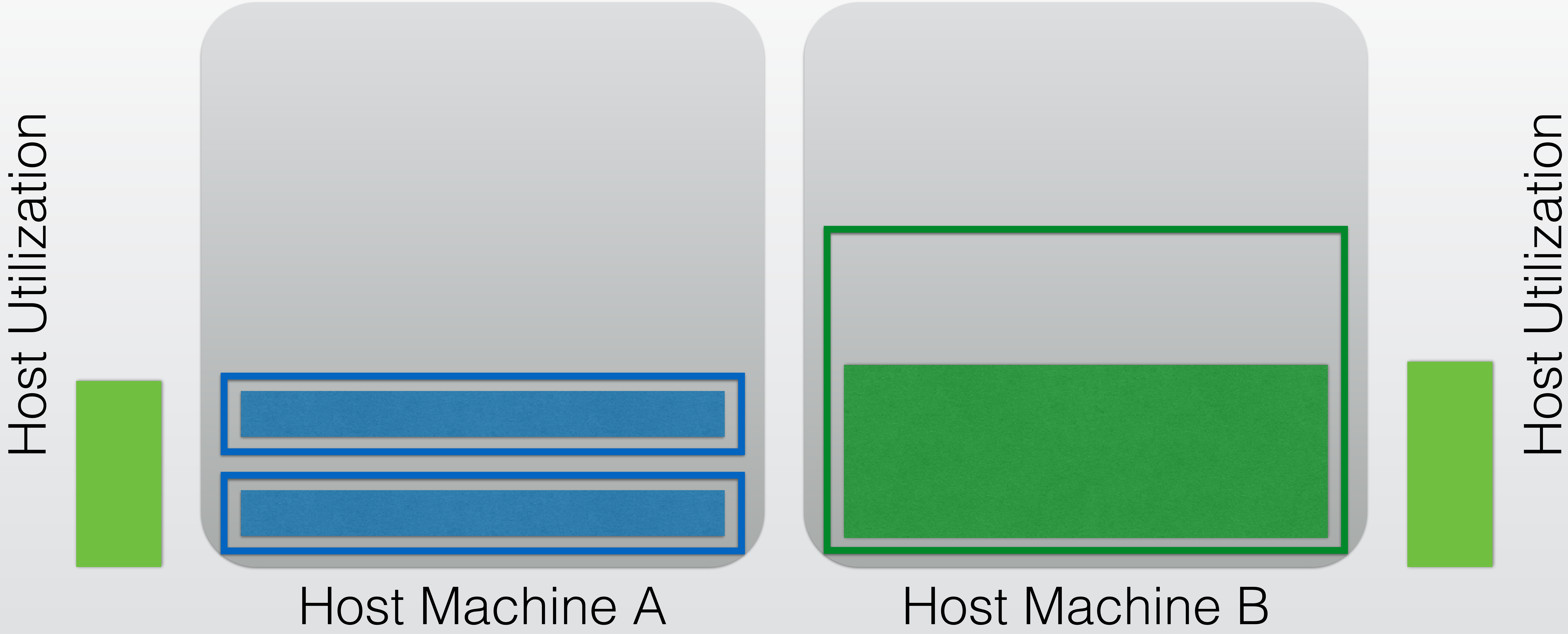
Host Utilization

Host Machine A

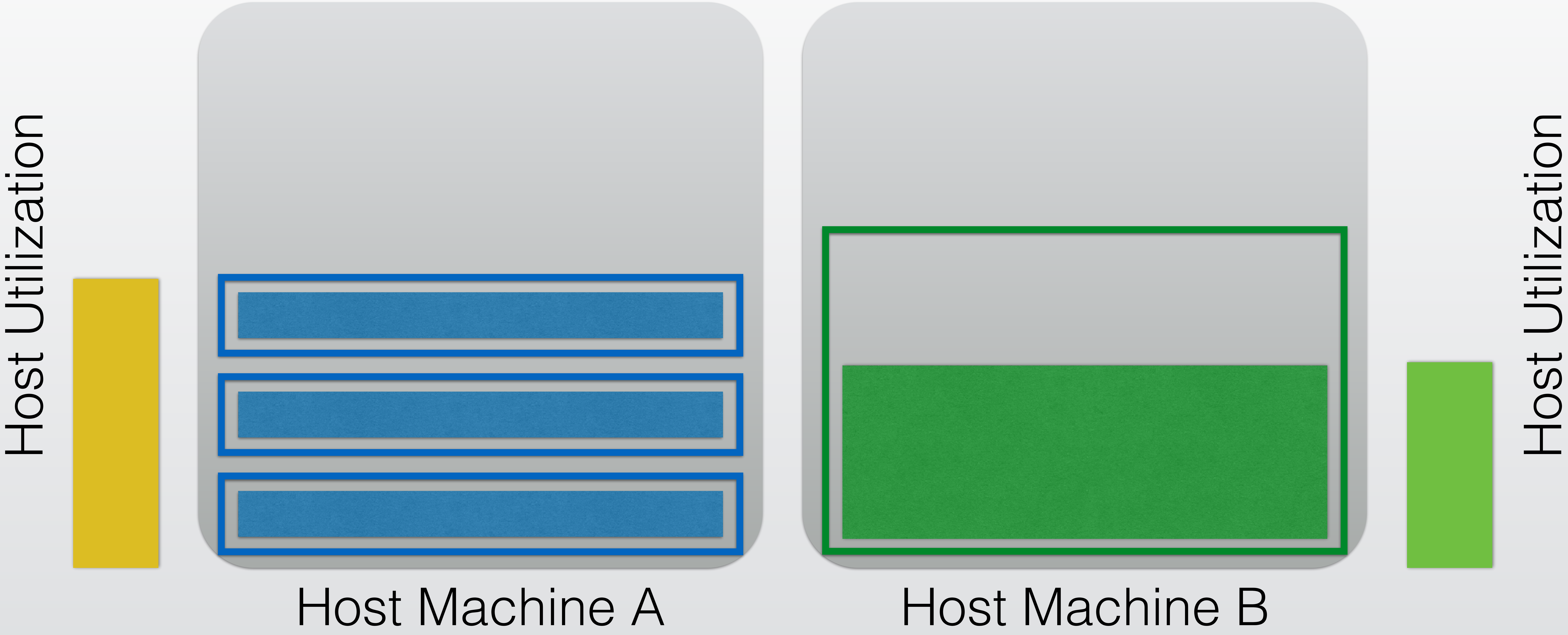
Host Machine B

Host Utilization

Scheduling

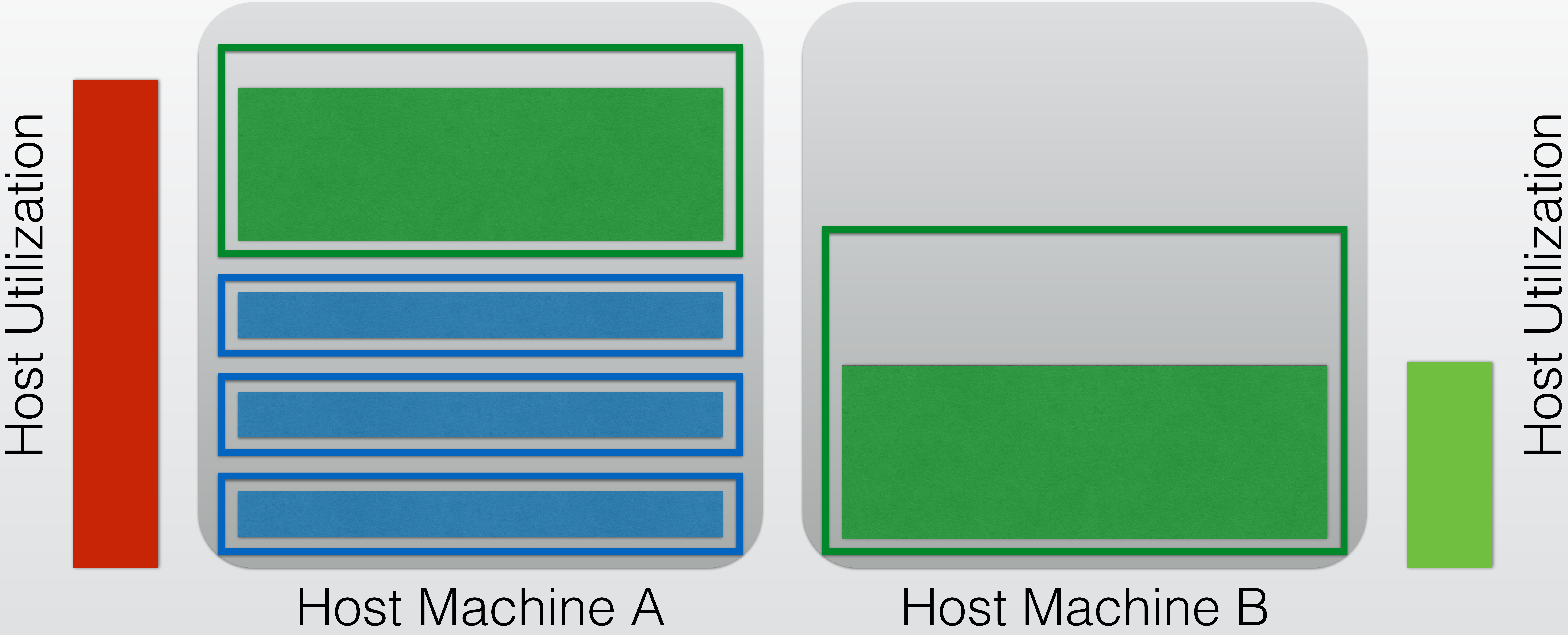

Host Utilization
Host Machine A
Host Machine B
Host Utilization

Scheduling

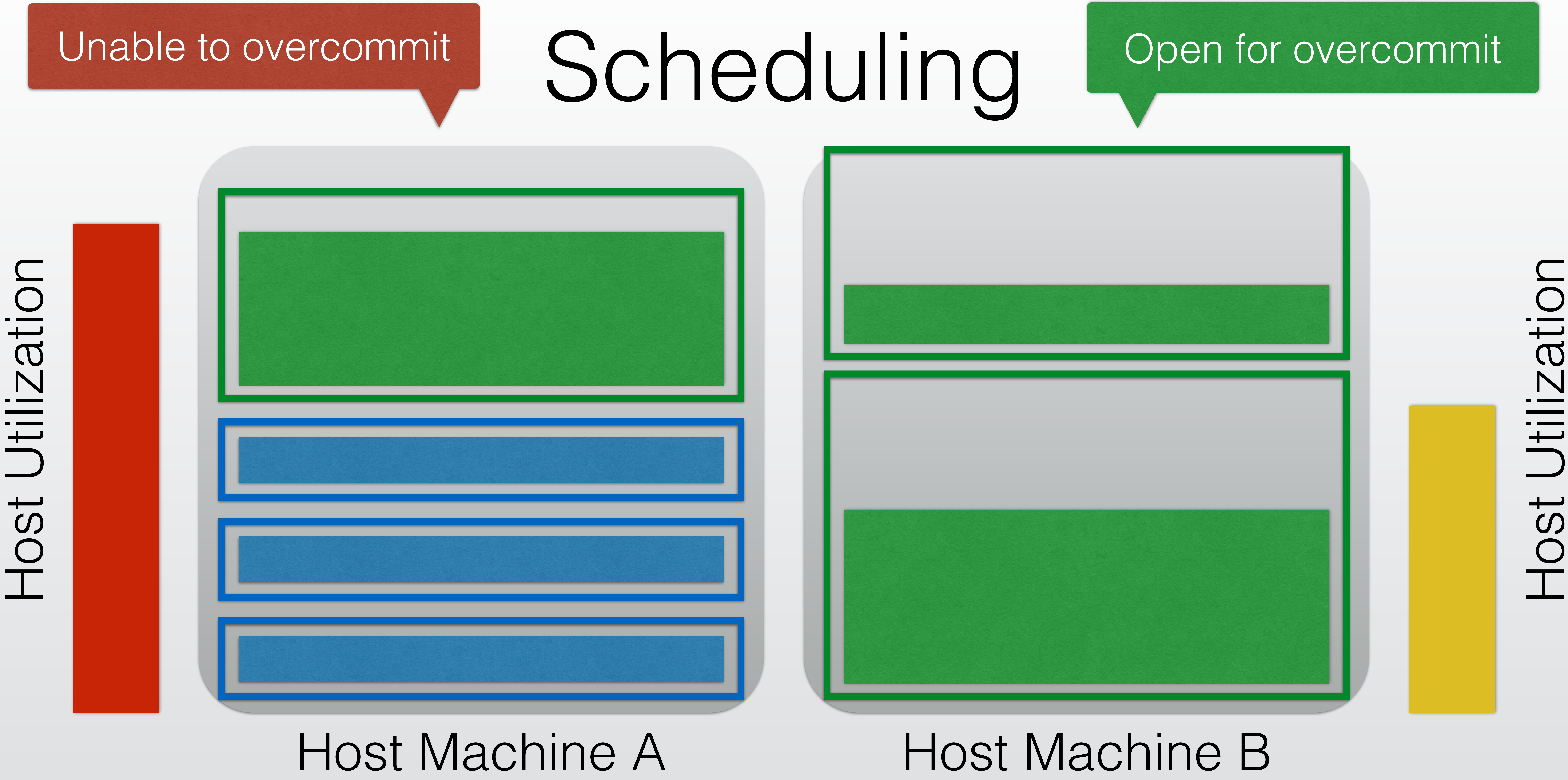


Host Utilization
Host Machine A
Host Machine B
Host Utilization

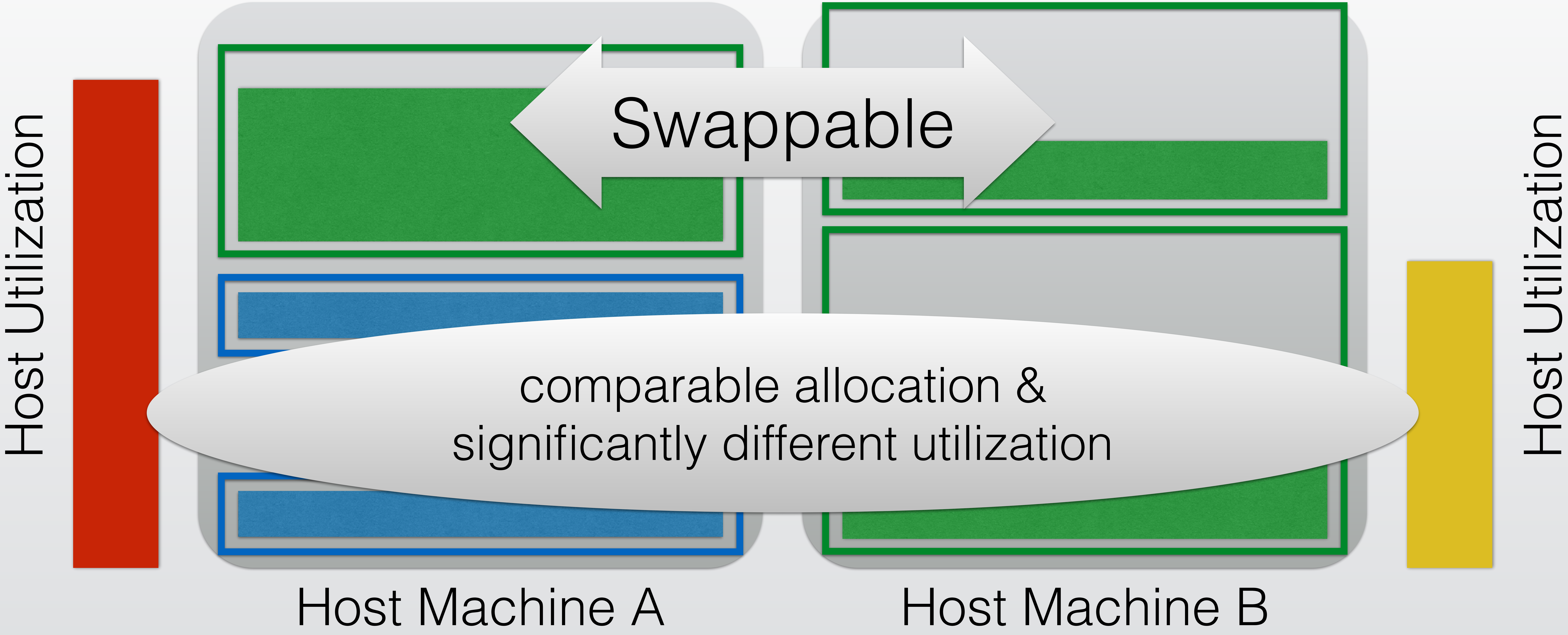
Scheduling


Host Utilization
Host Machine A
Host Machine B
Host Utilization

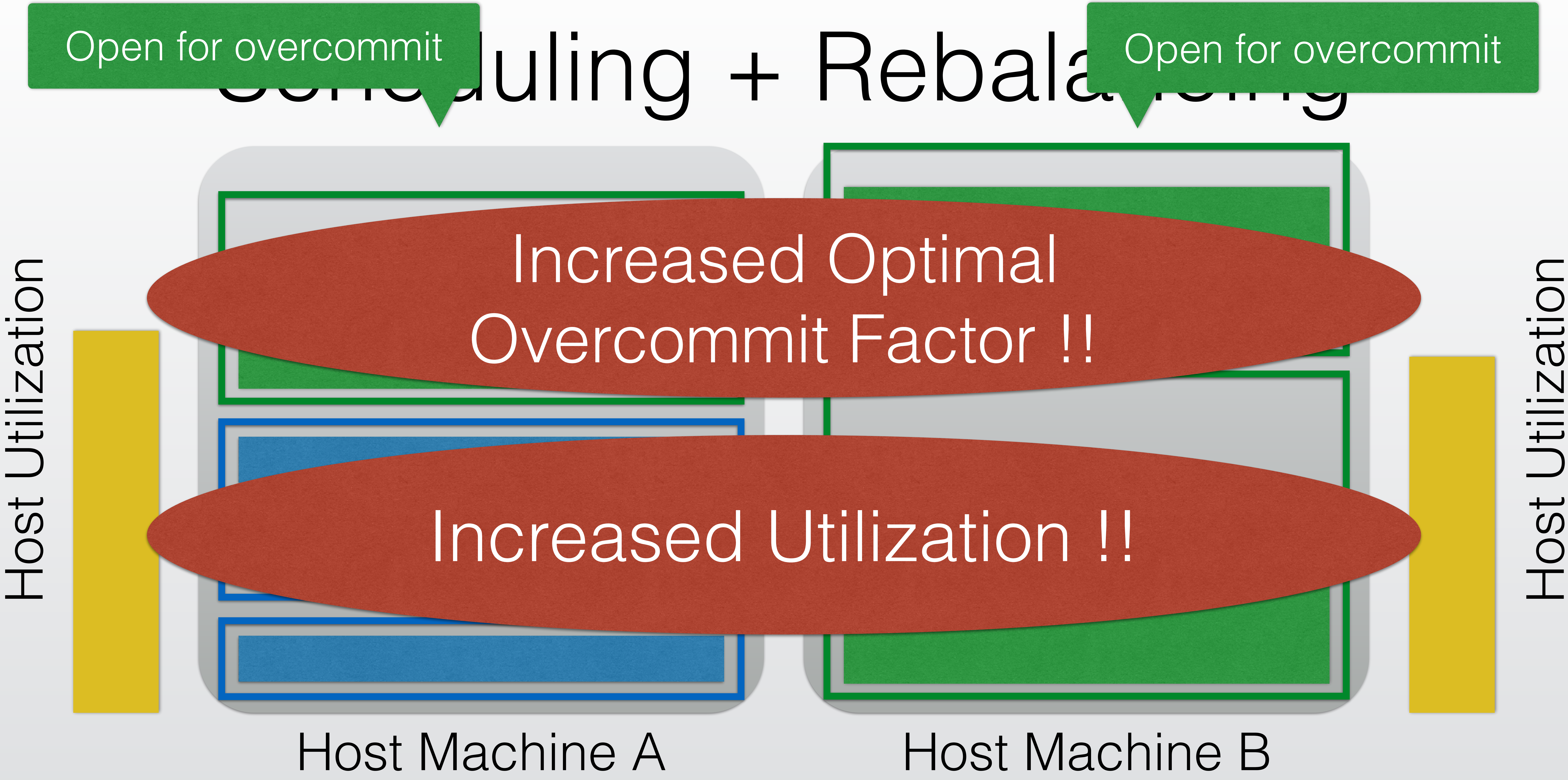
Scheduling


Unable to overcommit
Open for overcommit
Host Utilization
Host Machine A
Host Machine B
Host Utilization

Scheduling + Rebalancing



Scheduling + Rebalancing



Increased Optimal Overcommit Factor !!

Increased Utilization !!

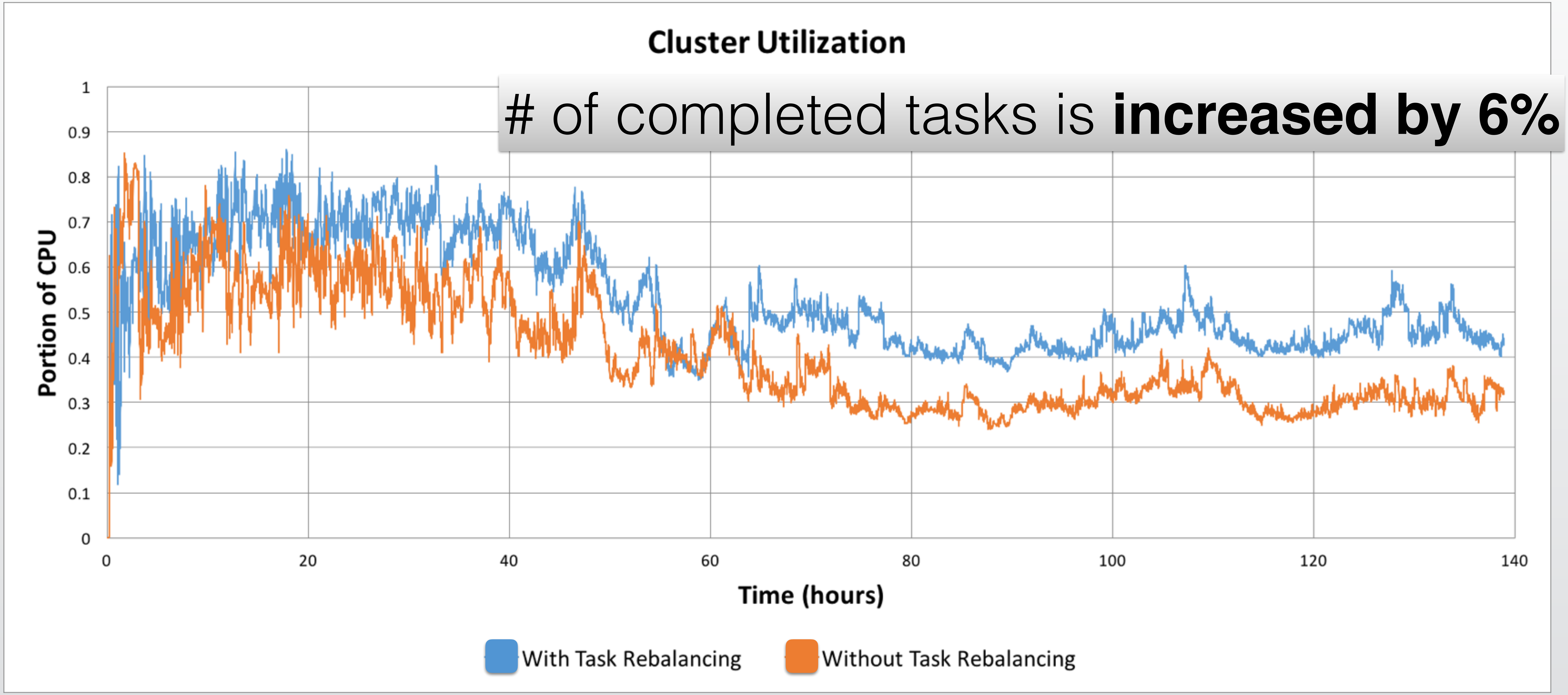
Evaluation

- Simulate cluster with/without task rebalancing
- Driven by Google's cluster data traces *,**
- Measure performances
  - Metrics: cpu utilization, # of completed tasks, etc.

* J. Wilkes, *"More Google cluster data."* Nov-2011.

** C. Reiss, J. Wilkes, and J. L. Hellerstein, *"Google cluster-usage traces: format + schema,"* Mountain View, CA, USA, Nov. 2011.

Preliminary Results



# of completed tasks is **increased by 6%**

Conclusion

- PROBLEM: Scheduling is inefficient
- INSIGHT: Linux container enables task micromanagement
- PROPOSAL: Task rebalancing
  - Find and swap *swappable task pair* to load-balance hosts
  - Increase optimal overcommit factor => Increase cluster utilization
- EVALUATION: Simulation driven by Google’s cluster data