Architecting Dependable Systems Using Virtualization

HariGovind Ramasamy
hvr@zurich.ibm.com
IBM Zurich Research Laboratory

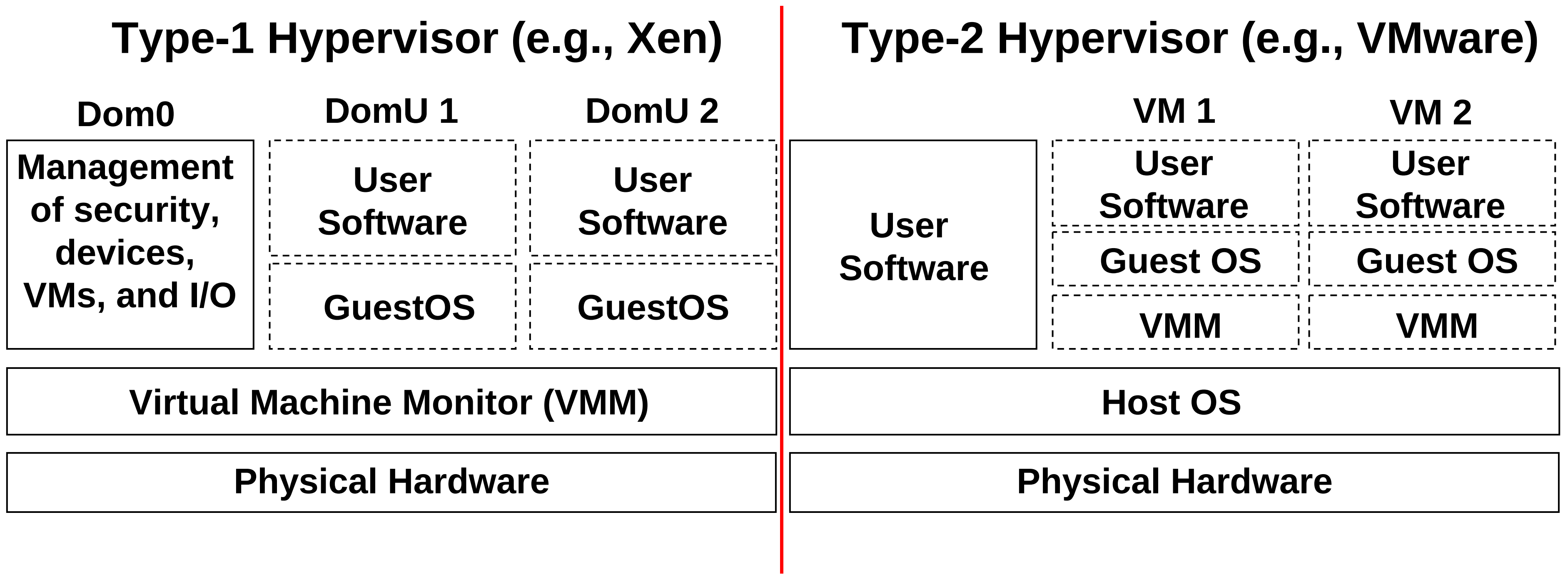
Joint work with
Matthias Schunter
mts@zurich.ibm.com
IBM Zurich Research Laboratory

DSN Workshop on Architecting Dependable Systems

June 27, 2007

Background: Virtualization

- Abstracts away the real hardware configuration
- Allows hosting of multiple virtual machines (VMs) on a physical machine



Contributions

- **How can virtualization improve system dependability?**
  - leverage VM flexibility characteristics to build around OS problems
- **When does virtualization really help?**
  - Quantifying the impact of virtualization on system reliability

Related Work

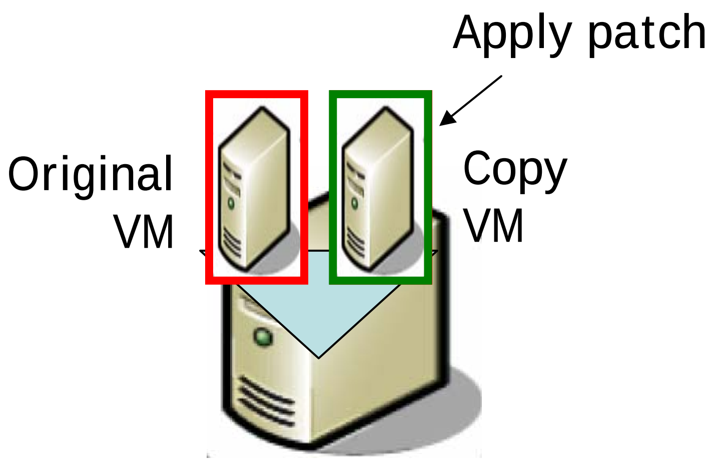
- Introduce enhancements at the VMM level transparent to OS/apps
  - e.g., checkpointing-recovery at the granularity of VMs,
    ensuring determinism at the VM level [Bressoud-Schneider’96],
    VM logging-replay [Dunlap et al. ‘02]

- Instrument OS/middleware/apps with them being aware of running on VMs as opposed to physical machines
  - e.g., checkpointing a Java application state at the VM-level or byte-code level (as opposed to native code) [Agbaria-Friedman’02]

Patch Application for High-Availability Services

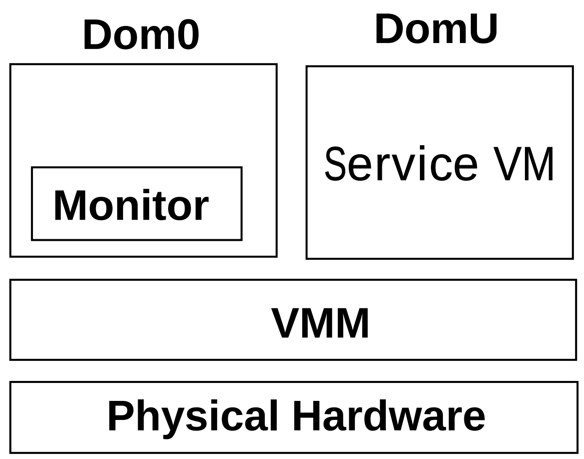
- **Motivation**
  - patch application typically involves system restart; negatively affecting service availability

- **Mechanism**
  - service is hosted on a VM instead of a physical machine
  - instantiate copy of VM, apply patch on copy instead of original VM
  - restart copy VM, while original VM continues to run
  - original VM gracefully shut down
  - copy VM takes over
  - Stateful service?
    - VM checkpointing + VM live migration [Clark et al. '05]



Enforcing Fail-Safe Behavior

- **Motivation**
  - Latency between publicizing vulnerability exploit & patch availability
    - avg. of 4.5 months for Windows security problems [2005]
  - Can’t shut down many services until patch becomes available!
  - *Compromise:* run service as long as possible



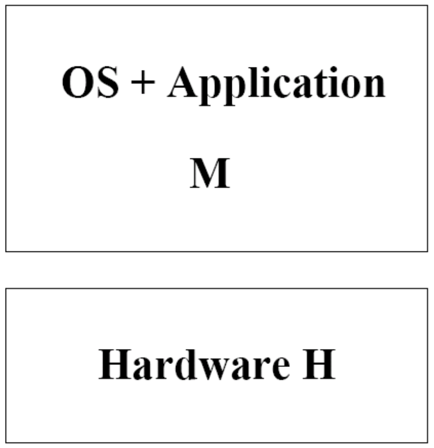
- **Observation:** Publicizing a flaw is accompanied by
  - details of attack signature
  - symptoms of exploited flaw

- **Mechanism**
  - service is hosted on a VM instead of a physical machine
  - develop a monitor external to *service VM* to detect symptoms of exploited flaw on *service VM*
  - monitor signals VMM to crash *service VM* upon flaw detection
  - e.g., in Xen, monitor can be in Dom0 and service VM can be DomU

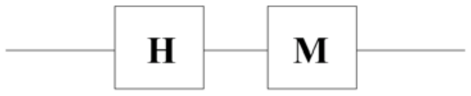
Boundary Conditions for Virtualization to Yield Reliability Benefits on a Single Physical Node

Non-Virtualized Service

Architecture



Combinatorial Model

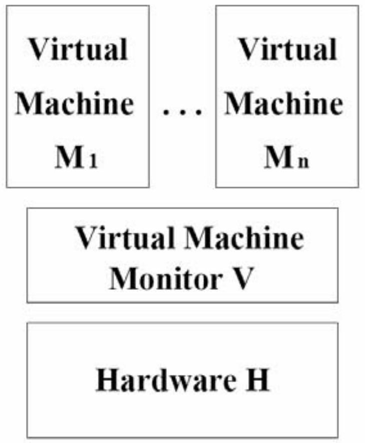


System Reliability

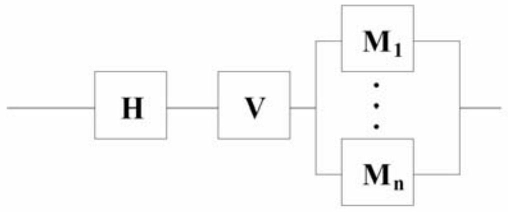
$$R_{sys}^{NV} = R_H \cdot R_M$$

n-Replicated Service

Architecture



Combinatorial Model

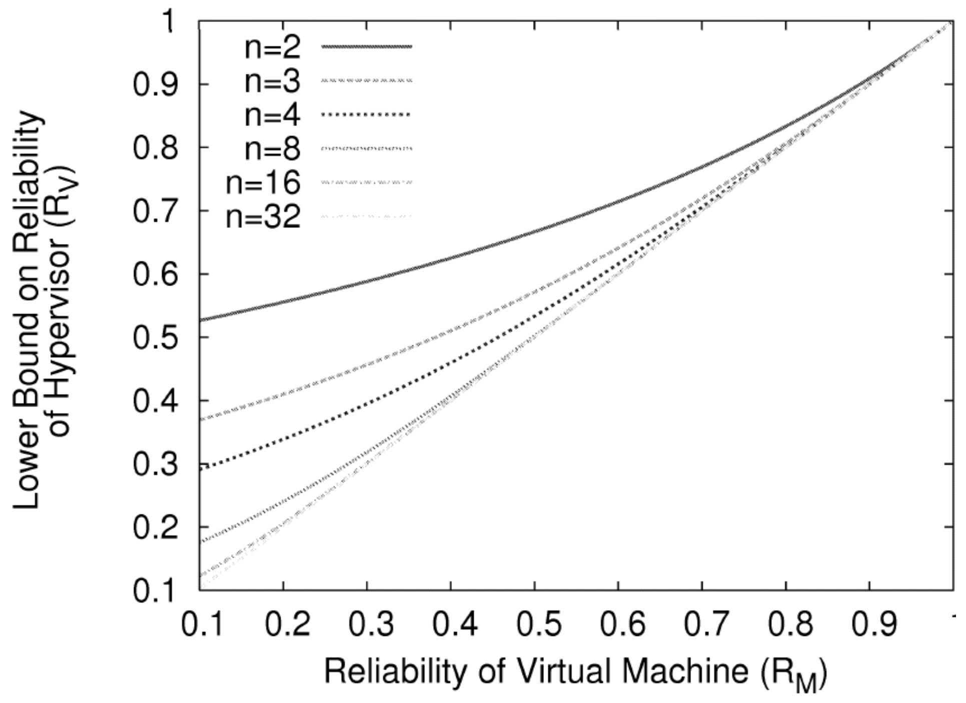


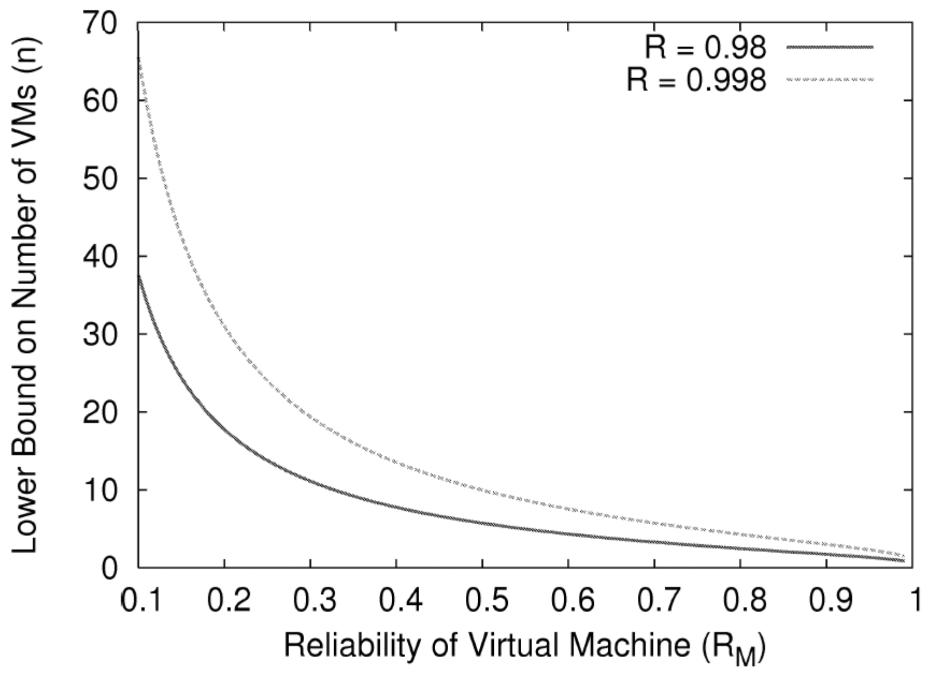
System Reliability

$$R_{sys}^{n} = R_H \cdot R_V \cdot [1 - \prod_{i=0}^{n}(1 - R_{M_i})]$$

Boundary Conditions for Virtualized Node to have Better Reliability

$$R_V \text{ ¤}[1 \text{ ¡ } (1 \text{ ¡ } R_M)^n] > R_M \longrightarrow \text{A}$$

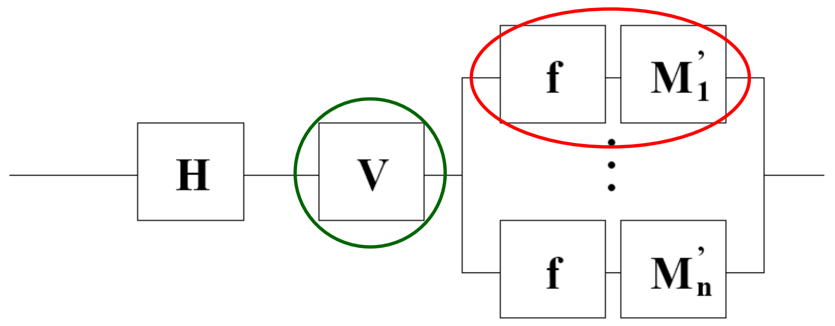




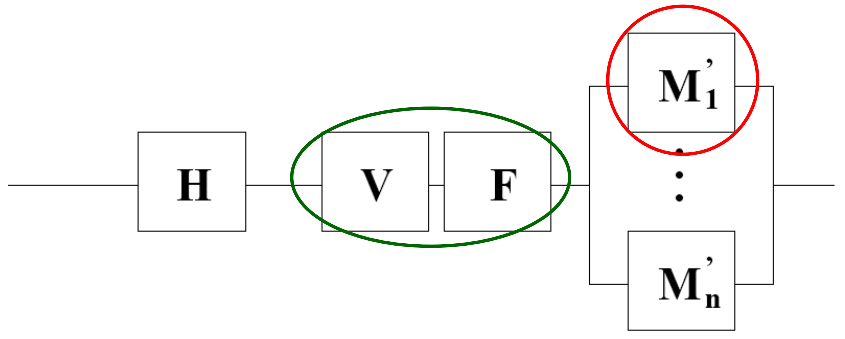
- For n=1, inequality (A) doesn't hold.
- Hypervisor has to be more reliable than VM.
- Hypervisor has to be more reliable when deploying fewer VMs (fixed R_M).
- There exists a min. *n* value below which (A) doesn't hold (fixed R_V and R_M).

Boundary Conditions:
Moving Functionality out of the VMs into Hypervisor

Distributed configuration

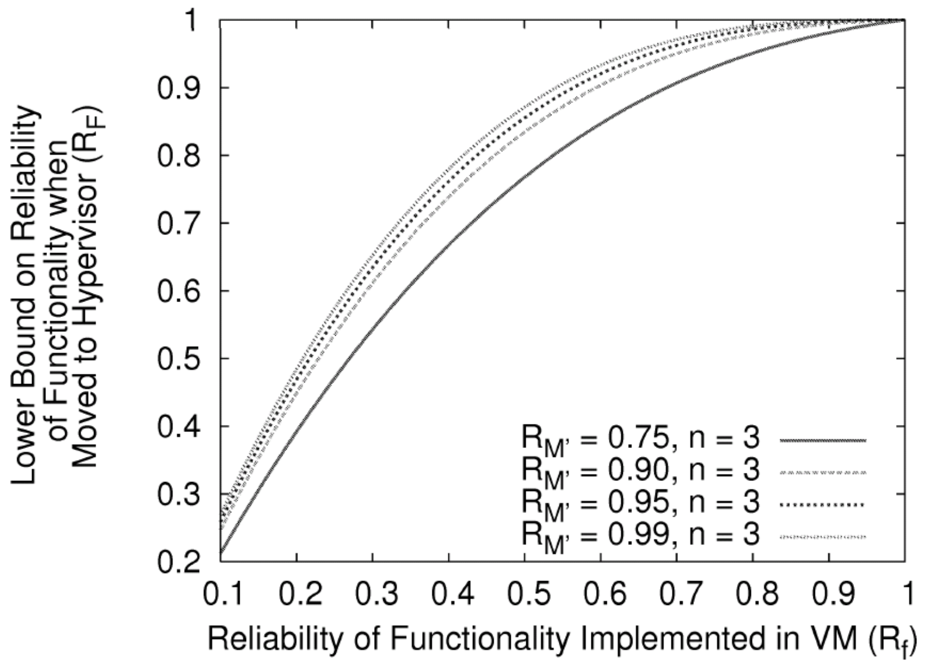


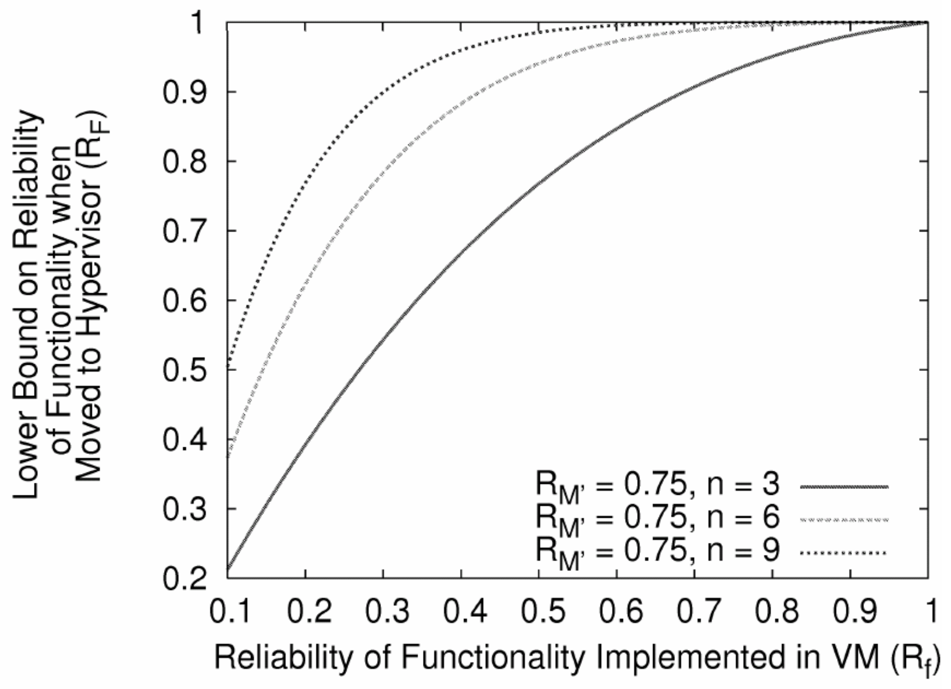
Consolidated configuration



Boundary Conditions: Moving Functionality out of the VMs into Hypervisor

$$R_F \geq \frac{[1-(1-R_f R_{M'})^n]}{[1-(1-R_{M'})^n]} \longrightarrow \text{B}$$





- Retaining a poorly reliable *f* in the VM is better than moving it into hypervisor.

Conclusion

- Ample opportunities for leveraging virtualization for dependability
- General trend to move services out of guest OS into VMM should be treated with caution
  - our results show that unless some boundary conditions are met, virtualization may, in fact, lower system reliability
- Rigorous modeling, analysis of dependability attributes in the context of virtualization is important

Proactive Software Rejuvenation

- Proactively rejuvenate guest OS and services inside a guest VM
  - by hooks introduced into the VMM layer
  - in a performance- and availability-preserving way

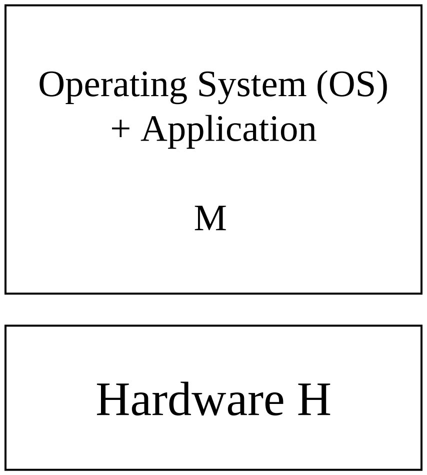
- Mechanism
  - *Reincarnation VM* booted from a clean VM image, while service is operational in another VM
  - original VM gracefully shut down
  - reincarnation VM takes over

- Stateful service?
  - VM checkpointing + VM live migration
  - possible to tune the amount of resources devoted to booting/initializing the reincarnation VM by adjusting time for reboot

Reliability Analysis

- Redundant FT designs involving virtualization on a single node
  - Model: n-replicated service
    - multiple VMs run concurrently on the node
    - VMs offer identical service
- Baseline for comparison: non-virtualized, single-OS node

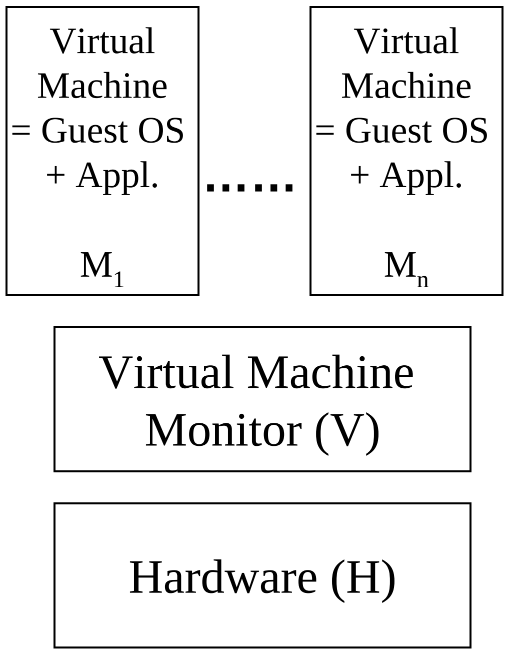
Non-Virtualized Service, Single Physical Node



$$R_{sys}^{NV} = R_H ¤ R_M$$

- Assumptions
  - M, H fail independently
- General Observation
  - Since assumption is unlikely to hold in practice, R_{sys} gives upper bound on system reliability

n-Replicated Service, Single Physical Node


Virtual Machine = Guest OS + Appl.
M_1
……
Virtual Machine = Guest OS + Appl.
M_n
Virtual Machine Monitor (V)
Hardware (H)

- Assumptions
  - $M_1,..M_n$, V, H fail independently
  - $M_1,\ldots M_n$ operate concurrently and provide service
  - No need for synchronization between $M_1,\ldots M_n$

$$R_{sys}^{Vn} = R_H \cdot R_V \cdot \left[1 - \prod_{i=0}^{n}(1 - R_{M_i})\right]$$