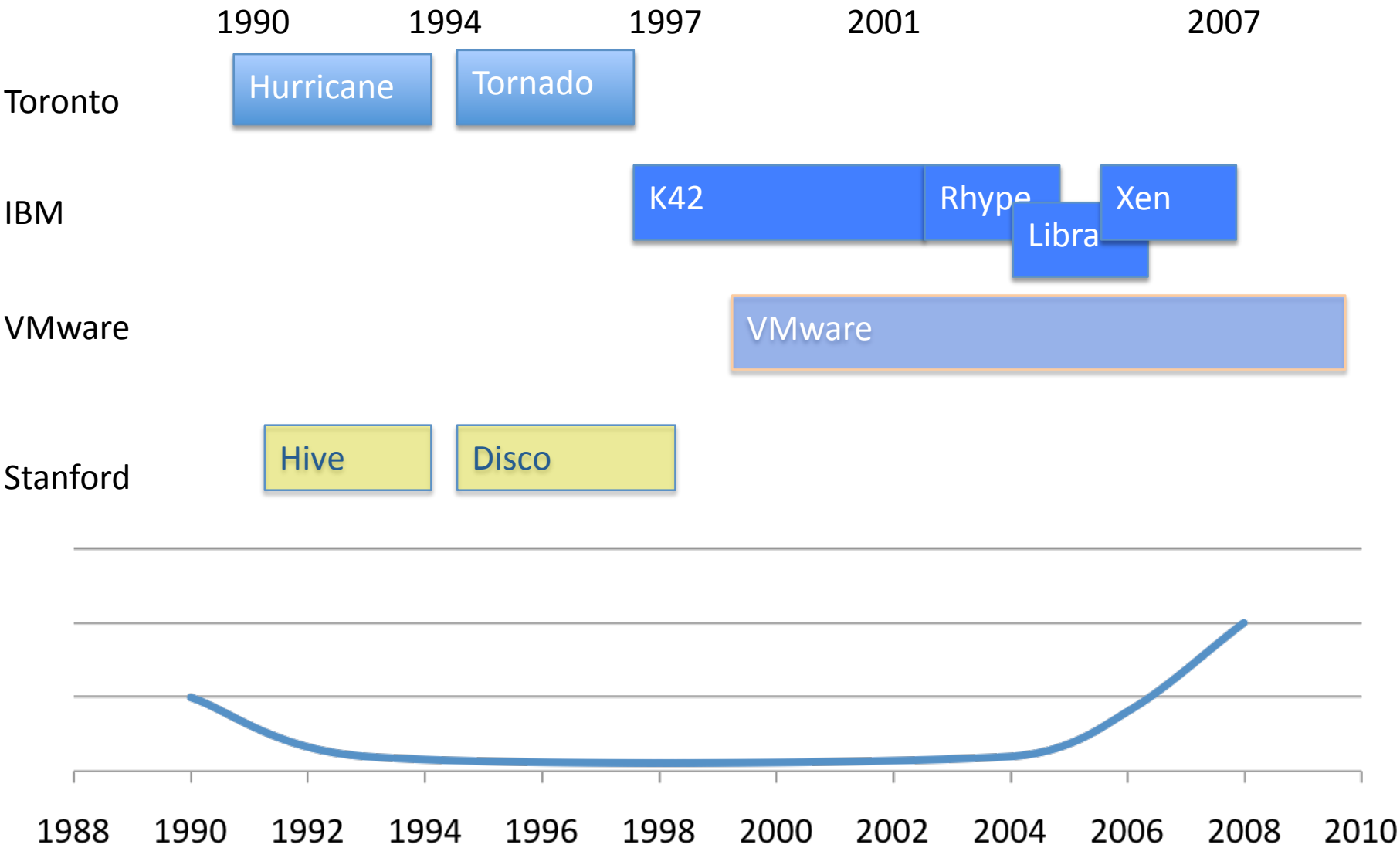
Musings about virtualization and the future of OS research.

Orran Krieger

A bit of history



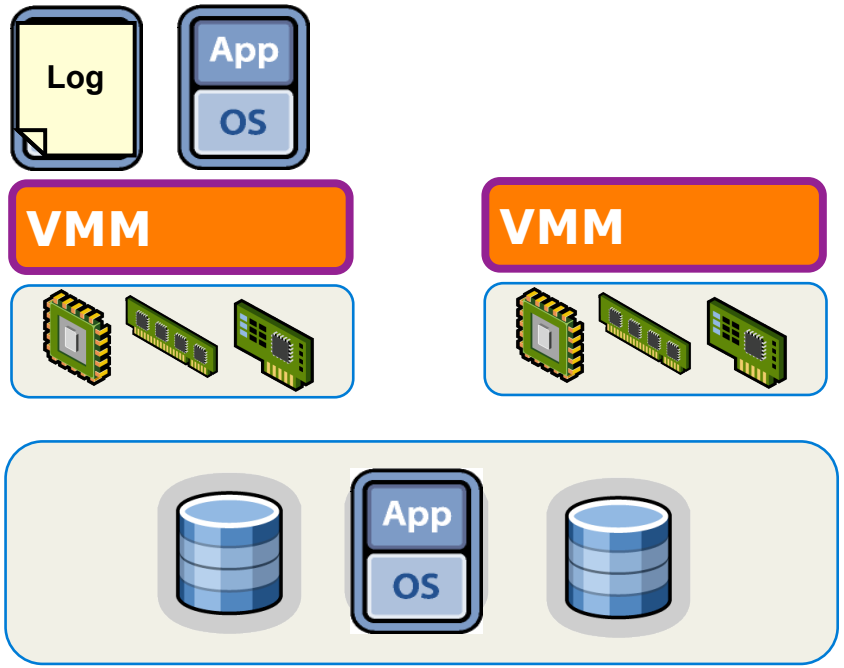
Outline

- Virtualization will be pervasive
- Why does a 1960s technology make sense?
- New uses of virtualization:
  - Application distribution
  - Utility computing
  - The OS of the future

Outline

- Virtualization will be pervasive
- Why does a 1960s technology make sense?
- New uses of virtualization:
  - Application distribution
  - Utility computing
  - The OS of the future

Virtual is better than physical

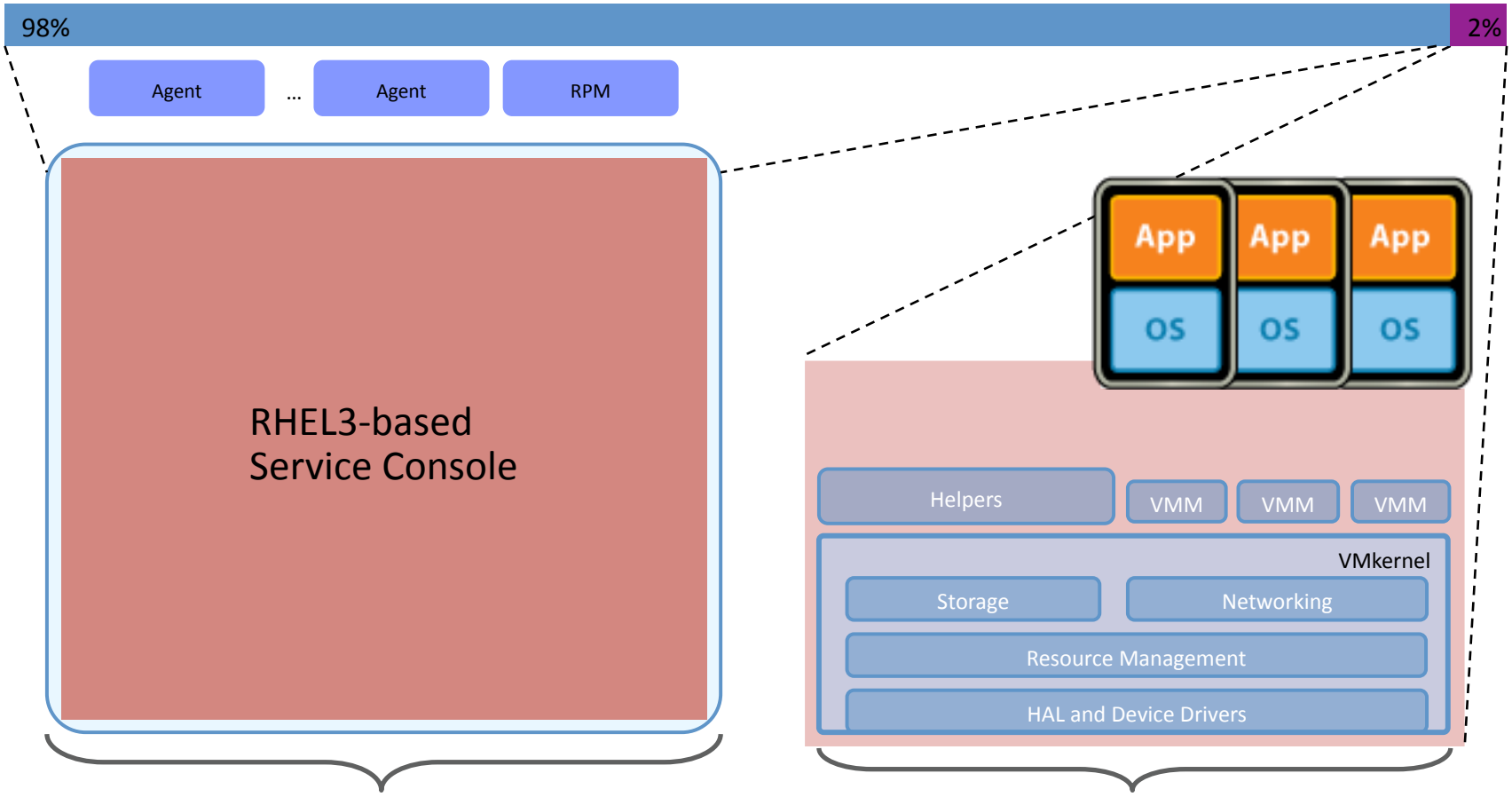


- Suspend
- Resume
- Snapshot
- Clone
- Migrate
- Record
- Replay
- *etc.*

Virtualization will be pervasive

- As of this year IBM, HP, Dell, FSC, NEC support new embedded hypervisor on x86 systems, although not yet pervasive
- IBM has shipped on z/p/I for years, Sun & HP shipping on non x86 HW.
  - Gives HW vendor place to deploy function
  - Gives much simpler out-of-the box experience
  - Long term… virtualization will be pervasive on all platforms.
- Requires fundamental changes.

Its got to get on a diet

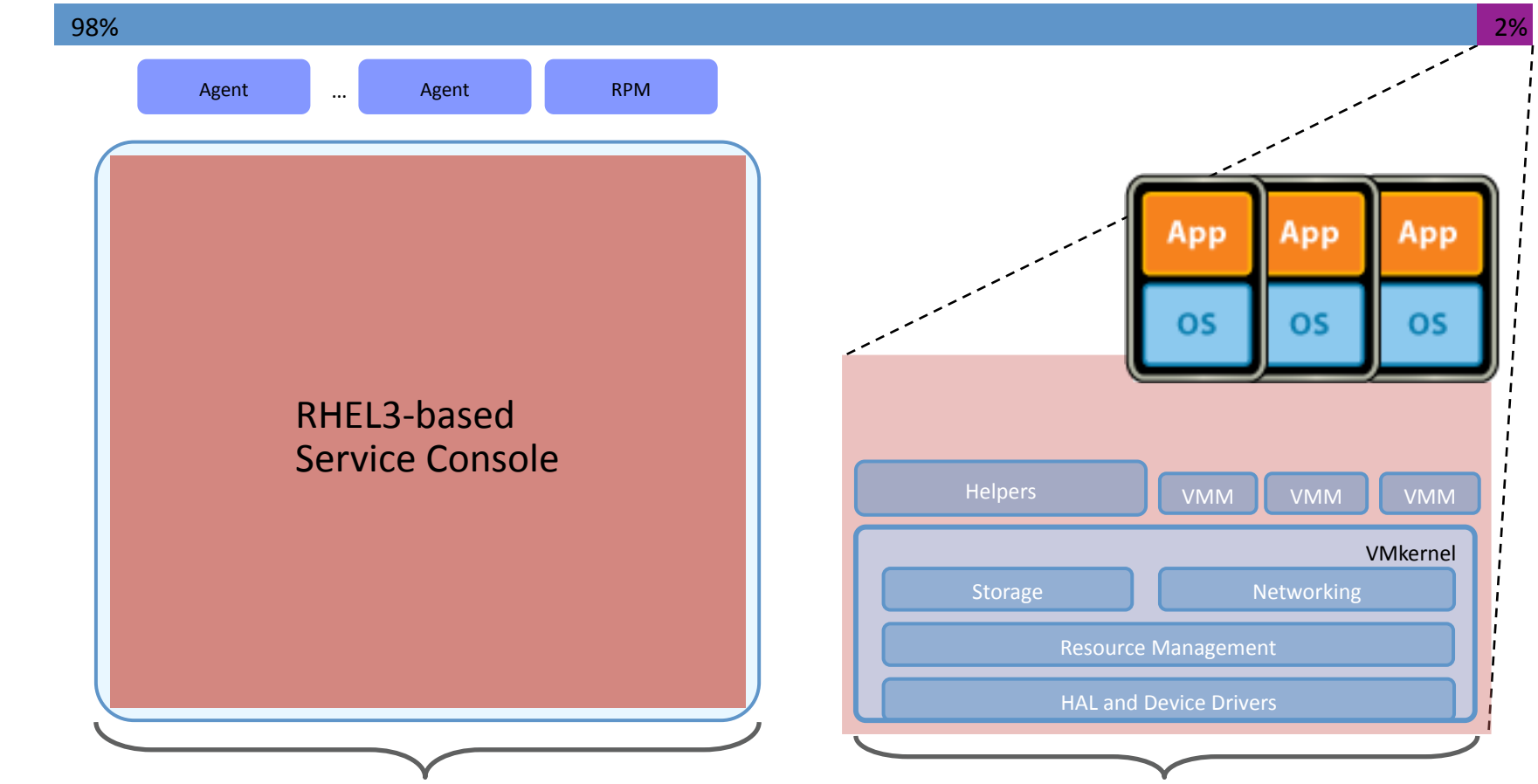


Disk Footprint: 2 GB

Percent of Patches >50%

Disk Footprint: 32 MB

Its got to get on a diet



Disk Footprint: 2 GB
Percent of Patches >50%

Disk Footprint: 32 MB

Other requirements

- Support all requirements that HW might be used for:
  - Deterministic real time
  - High assurance
  - Embedded, Client & server
- We will need to move to much more customizable design, where platform depends on the modules installed.

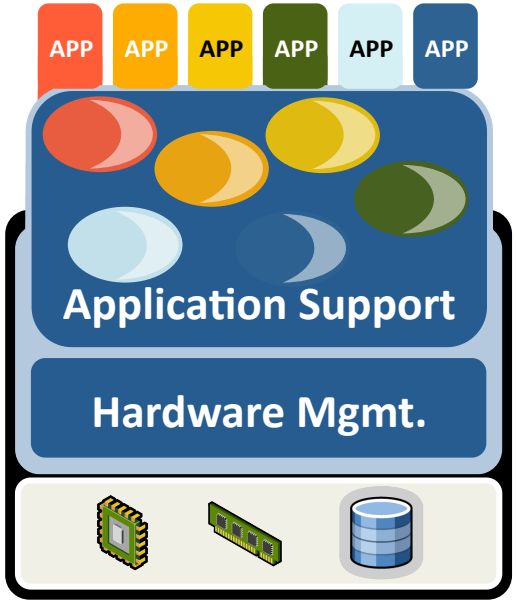
Outline

- Virtualization will be pervasive
- Why does a 1960s technology make sense?
- New uses of virtualization:
  - Application distribution
  - Utility computing
  - The OS of the future

How can virtualization be important?

- Virtualization invented to allow OS development on massive mainframes.
  - Allowed HW and OS developers to work independently: everyone else had tight coupling, new OS for each HW platform
  - Allowed new stacks to be deployed & tested by customer.
- Why is this technology important in a world of cheap scale-out commodity computers?
  - Much of the original value for HW virtualization subsumed by virtualization levels in and above OS.

Modern OS Evolution



- Commodity OSes had enormous pressure to add function to support new applications.
- Security and reliability secondary concern in the broad market, -> feature creep…
- The rich function of general purpose OS became necessary for servers…

- Problems — Too complex
  - Security
  - Reliability
  - Manageability
  - Performance
  - Innovation

The research community failed

- Research community developed alternatives:
  - Microkernels
  - Exokernels
  - Customizable OSes …
- No innovative technology could get sufficient function to build up massive user base:
  - User had to dedicate HW to the special purpose OS: new innovation needs to be free to get community
  - Had to support large complex HW
- Existing OSes too complex to allow incremental innovation

What virtualization gives us

- A stable interface that the rich function of commodity OSes runs above.
- Allows new technology to be deployed alongside or underneath commodity OSes.
  - Zero cost to deploy innovation, if it solves a problem, great!
  - HW complexity disappears, way easier to develop new OS.
- "Any problem in computer science can be solved with another level of indirection" (David Wheeler)
- Virtualization, as the level of indirection, allows us to have massive adoption and innovation.

Outline

- Virtualization will be pervasive
- Why does a 1960s technology make sense?
- New uses of virtualization:
  - Application distribution
  - Utility computing
  - The OS of the future

Problems with SW distribution model

- Not so bad with MacOS, System Z… vendor controls it all, especially if application specific to platform.
- In the high volume MS and Linux space, lots of vendors, working at different schedules…:
  - ISV must support many operating systems from a correctness and performance optimization perspective.
  - ISV must support for all patches (e.g., app, websphere, DB2, OS), applying patch may break any level, DLL hell...
  - Installation: user must install full OS, administer all the levels.
    - 50% of support calls during install/configure cycle
    - Installation different from what ISV expected, performance/function problems
  - Support hell: where does the problem come from
  - Backup, HA… tools different from what ISV tested
  - Patching hell: patch of any level may break others
  - Management and performance optimization complexity for user…
  - Multi-tiered application has multiple components that need to be configured and integrated to operate together.
  - Huge barrier to entry for new SW:
    - SW must have compelling value, solving many problems
    - SW gets more complicated, and we are stuck in a vicious circle.

Hardware Appliance-Based Solutions: PROS

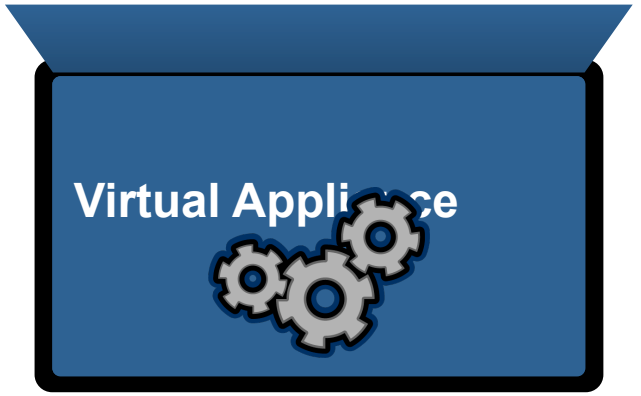
- Comes “pre-assembled” with all of the required components for the solution (hardware, OS, application bits)
- Simple plug-and-play installation
- Consistent stack makes support easier for vendor
- Underlying lower levels (OS) can be “hidden” from user

Hardware Appliance-Based Solutions: CONS

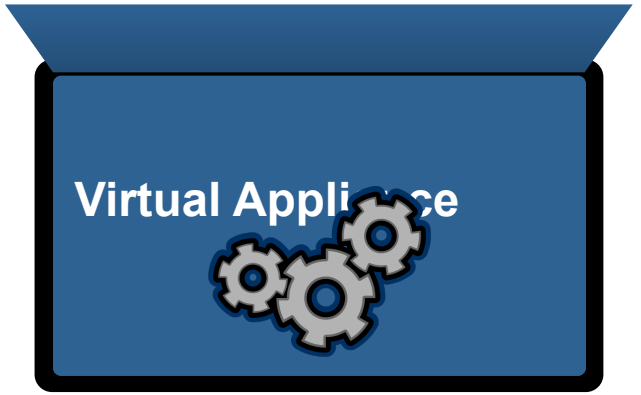
- Proliferation of non-standard hardware
- Hardware support provided by non-standard vendor (possibly multiple vendors: one for hardware, one for overall solution)
- Operations team has to learn how to support both the hardware and the solution
- Hardware used for a single purpose may be under utilized
- Hardware takes up rack-space, consumes additional power, and requires additional HVAC

What is a Virtual Appliance

- Pre-built, pre-configured and ready-to-run software application packaged with the OS inside a Virtual Machine.



- Or packaged inside multiple Virtual Machines



VA distribution model

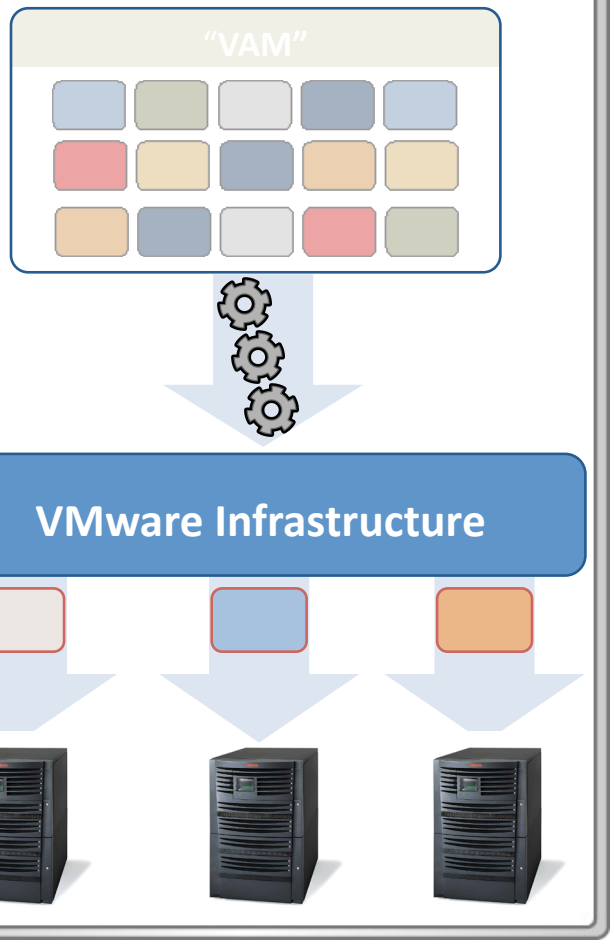
- ISV supports a single OS, and can full optimize
- ISV controls the full environment, patches…
- No installation effort
- No finger pointing
- Backup, HA… either integrated in appliance, or at the VA level
- Multi-tiered application configured by ISV
- New SW can be easily deployed, tested, discarded, does not need to solve all the worlds problems
- Customer picks the HW

Where we are going long term

Ready-to-Go Music

iPod

Ready-to-Go Apps



What is needed

- Tools for developing virtual appliances
- Tools to allow ISV ongoing role in administering solution.
- Standards in infancy: OVF
- Meta data to describe expected role of components to VI
- How do we make download cheap (streaming, de-duplication…)
- Evolution of SW to separate installation from customization.
- VI that can introspect on appliance, what level of trust in ISV required?

- Utility infrastructure
- New OS model

Outline

- Virtualization will be pervasive
- Why does a 1960s technology make sense?
- New uses of virtualization:
  - Application distribution
  - Utility computing
  - The OS of the future

Grid and grid-like technologies are the future

- Web applications being developed with scale-out frameworks
- MapReduce and related frameworks…, Data Intensive Supercomputing
- Utilities inside companies
- Hosting companies like Rackspace
- Shared utilities like Amazon's EC2
- The rise of SAAS like salesforce.com
- HPC capacity and capability systems

Grid and Virtualization are complementary:

- Grid: motivated by analogy of power grid
- Virtualization: converts computation into a fungible commodity; hugely simplifies realization of grid

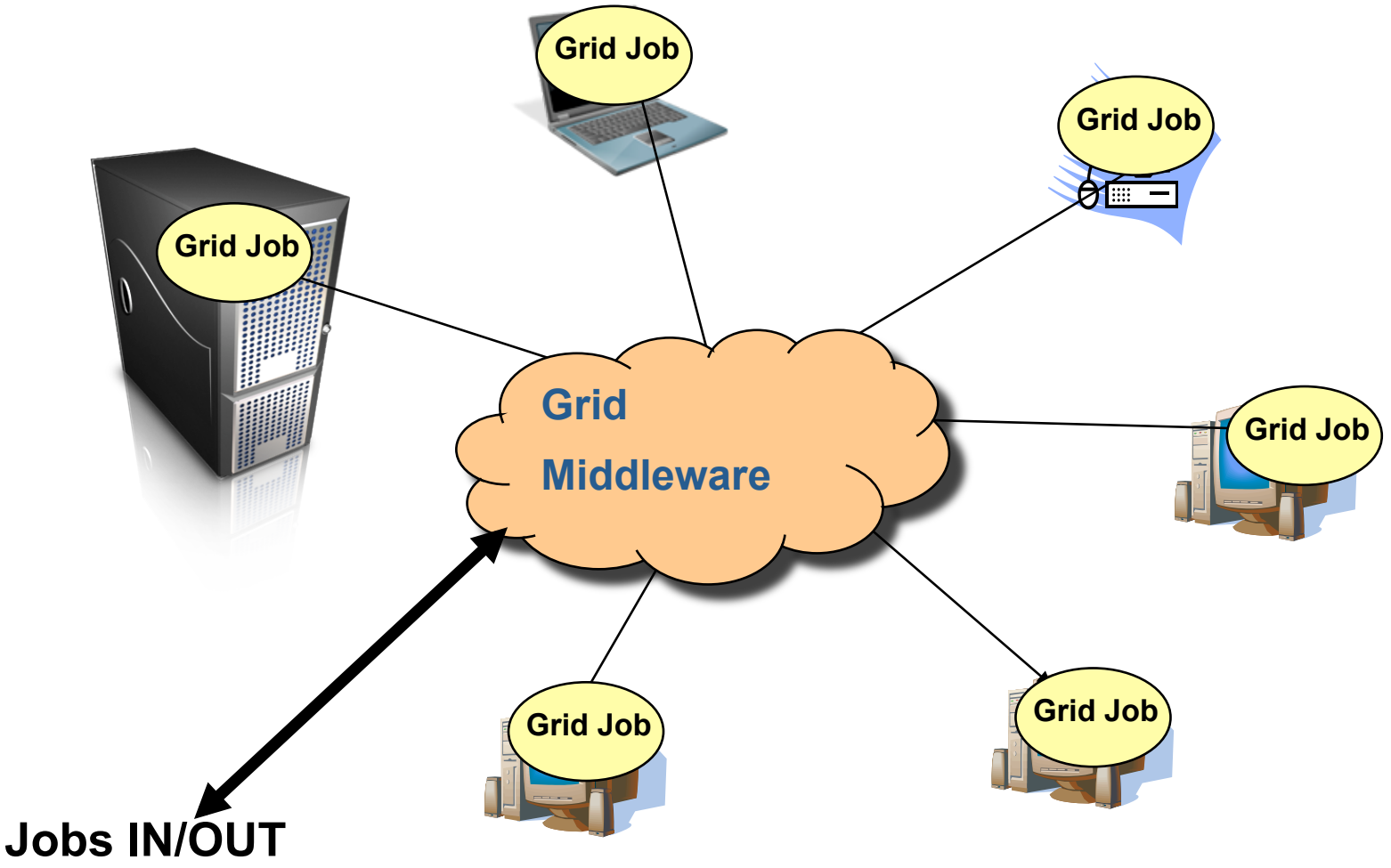
Problems with grid

- Security: grid job can compromise host
- Isolation: resource isolation grid/host task
- Service level guarantees: needed for enterprise
- OS heterogeneity limits targets
- Application needs to be re-written
- Utilization of grid nodes
- Management of hardware
- Use of traditional OS:
  - Large attack surface, high overhead, management complexity…

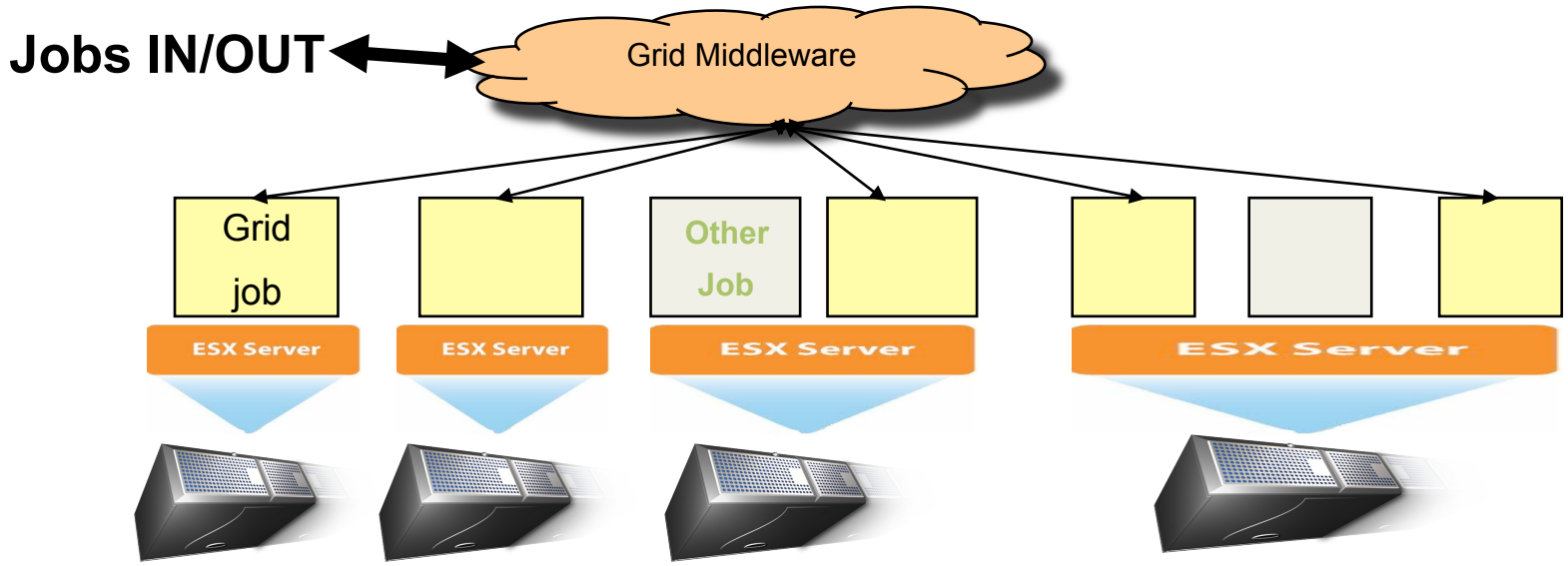
Virtualization and the grid

- Isolation:
  - VM's can't compromise other jobs or host
  - Resources VMs isolated
- Service level guarantees
- OS can be created on demand
- Application do not need to be re-written
- Grid node can be fully used
- Management of hardware
- Can write special purpose OS:
  - Reduce attack surface, increase reliability, simplicity…
  - Lets grid job get closer to the metal.

Grid Computing



Virtualization based Grid

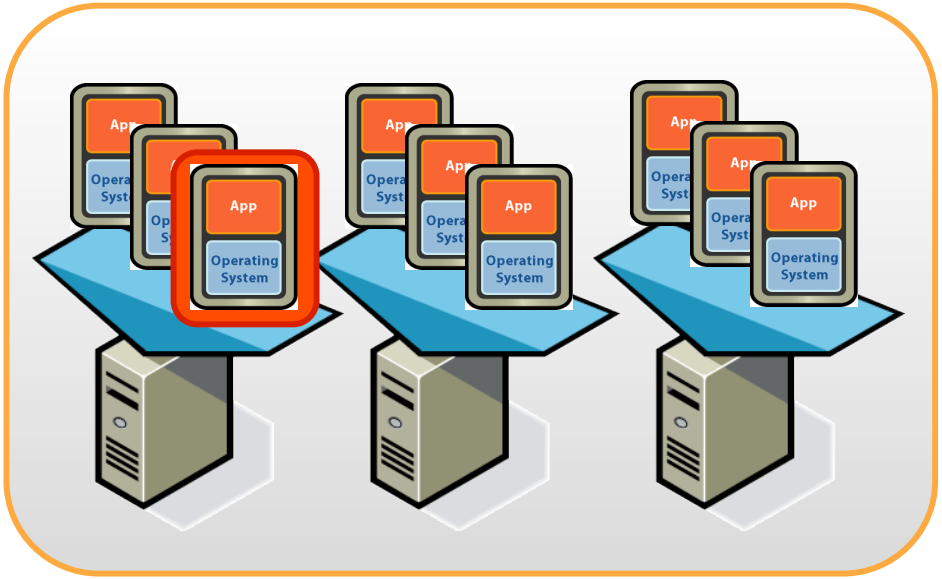


- Grid middleware can focus on “core” Grid issues.
- Virtual infrastructure provides simplified abstraction used by Grid middleware and other applications.

Automatic load balancing across hosts

Distributed Resource Scheduling (DRS)

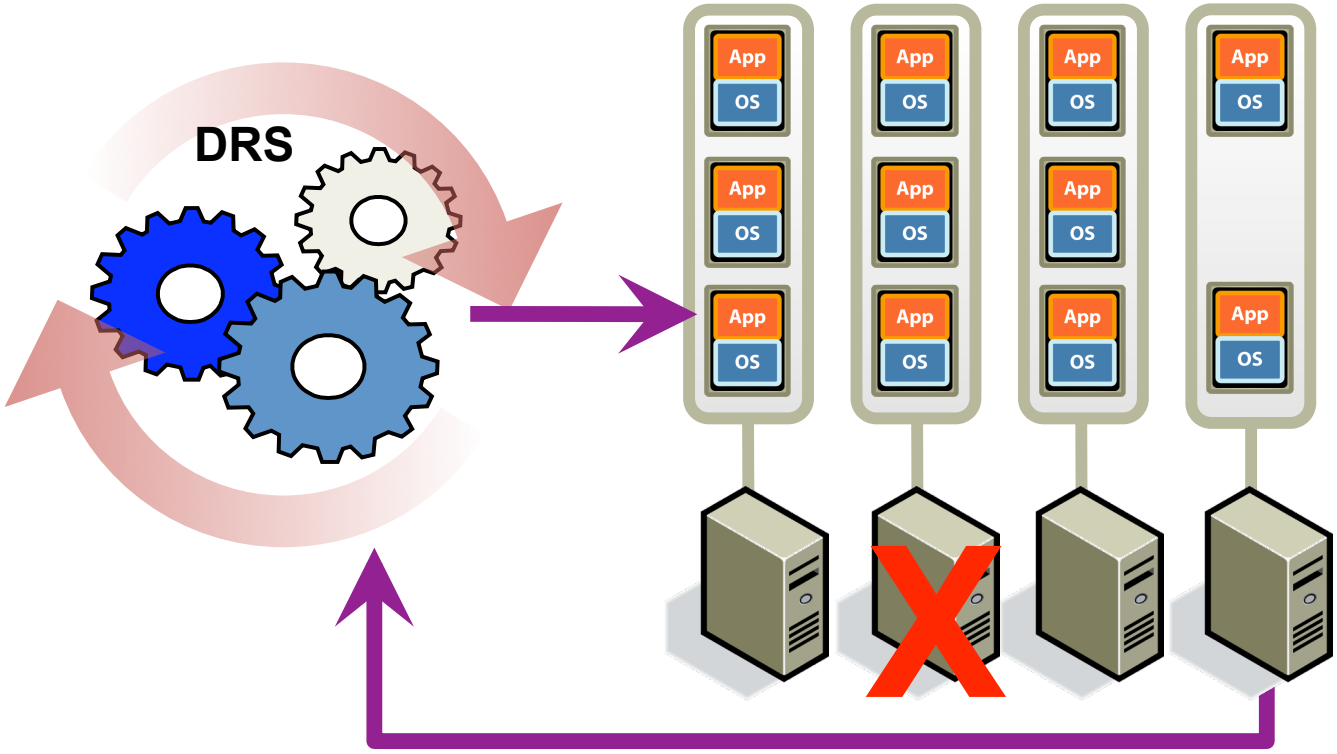
> **Dynamic Balancing**
> **Continuous Optimization**



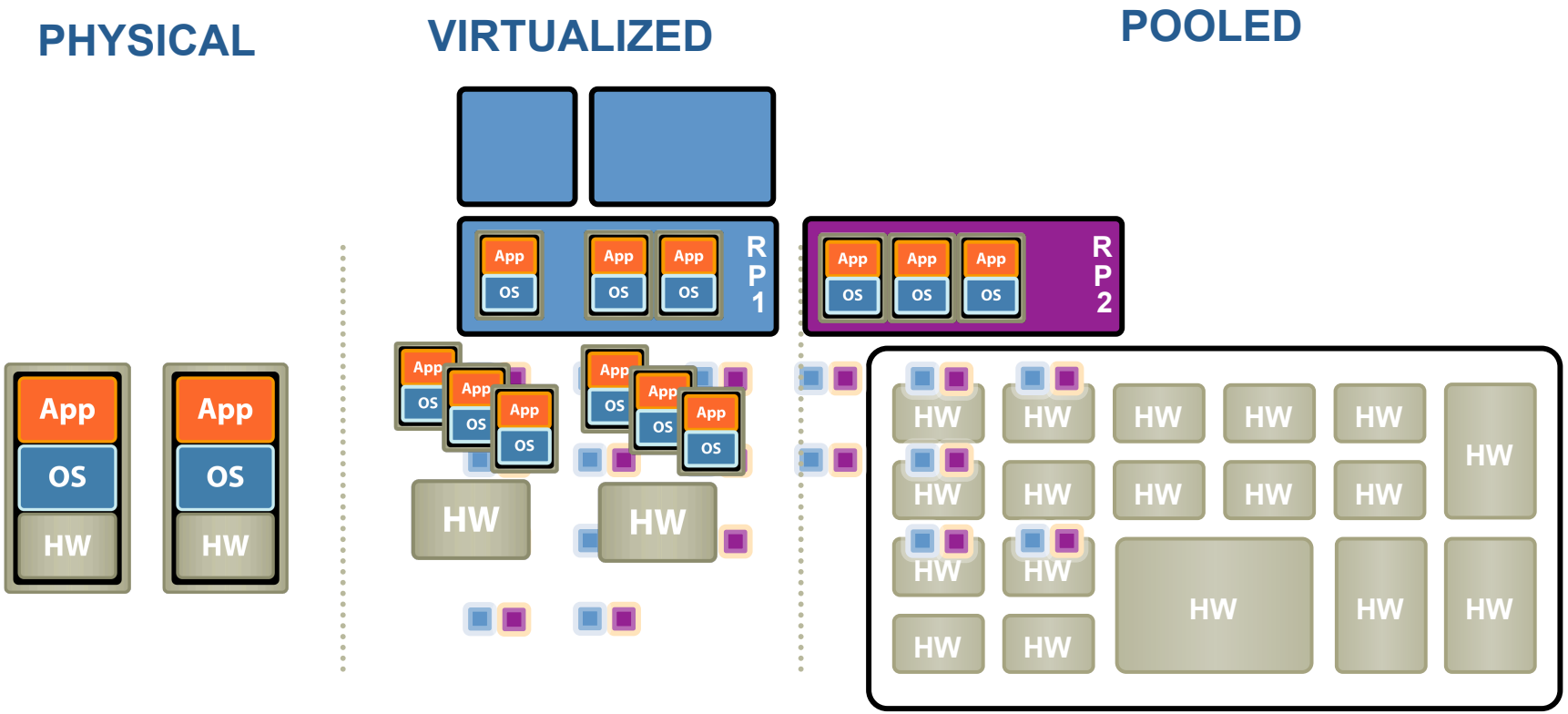
Adding and removing hosts

Hot-plug machines

- Add/remove capacity on demand
- Distributed power optimization
- Improve application availability



Moving to Utility model



> Logical Resource Pooling (RP)
> Distributed Resource Scheduler (DRS)

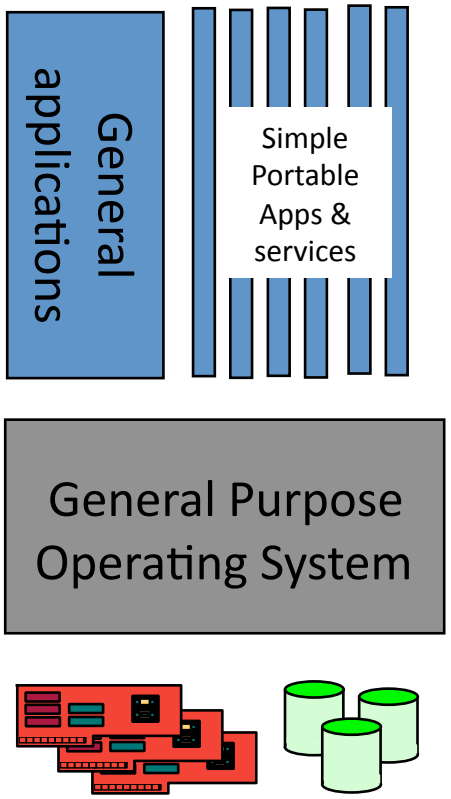
Research

- Quality of service for different physical resources.
- Detecting resource use and impact on application.
- Expressing application requirements to infrastructure.
  - Scale up/down web application?
  - Co-scheduling of HPC applications.
  - What is the right abstracting between grid/cloud middleware… and virtual infrastructure.
- Enabling very heterogeneous HW.

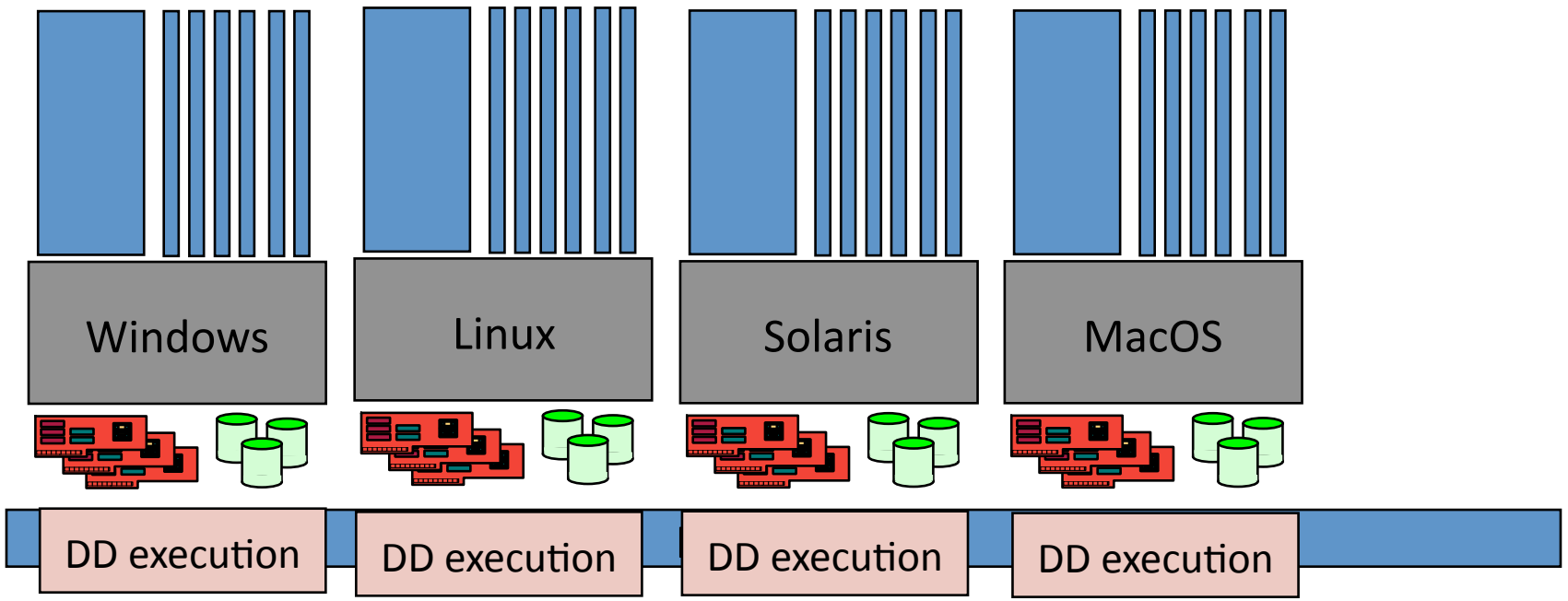
Outline

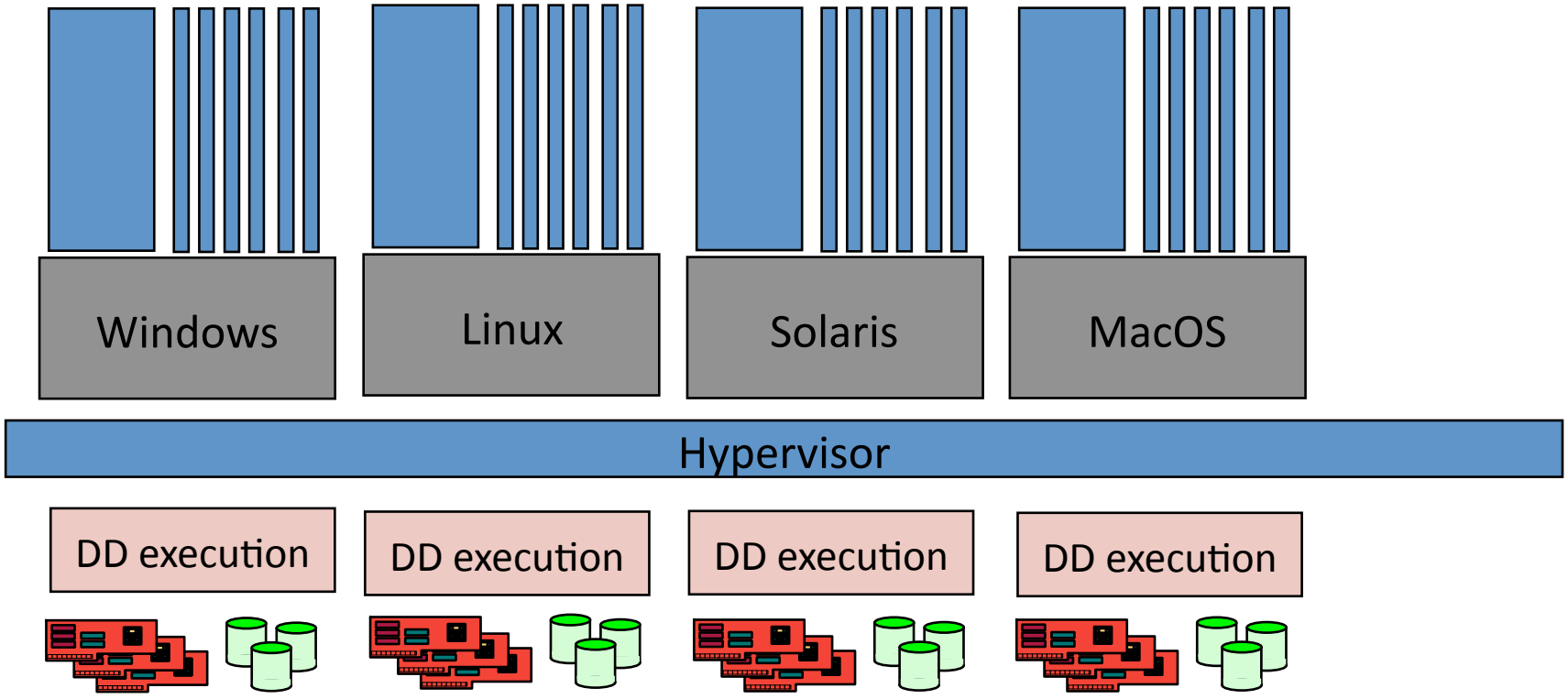
- Virtualization will be pervasive
- Why does a 1960s technology make sense?
- New uses of virtualization:
  - Application distribution
  - Utility computing
  - The OS of the future

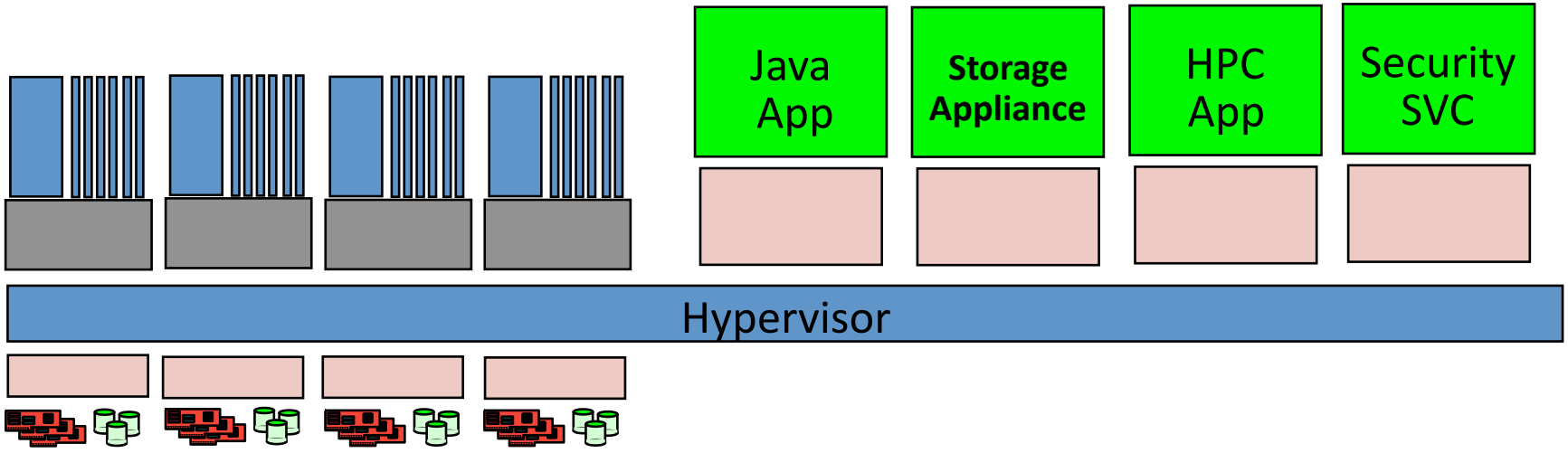
What's the problem again?



- Our general purpose OSes are a compromise between:
  - An execution environment for running device drivers.
  - An execution environment for running complex general-purpose applications.
  - An execution environment for portable applications and services with few OS needs.
- The general application support has gotten enormously complicated.
- They are enormously difficult to customize to support new workloads, or to exploit new specialized HW.
- Massive investment to support all the different OSes, e.g., validation in application, device driver development…
- Come with substantial management overhead.
- Virtualization has made it worse…

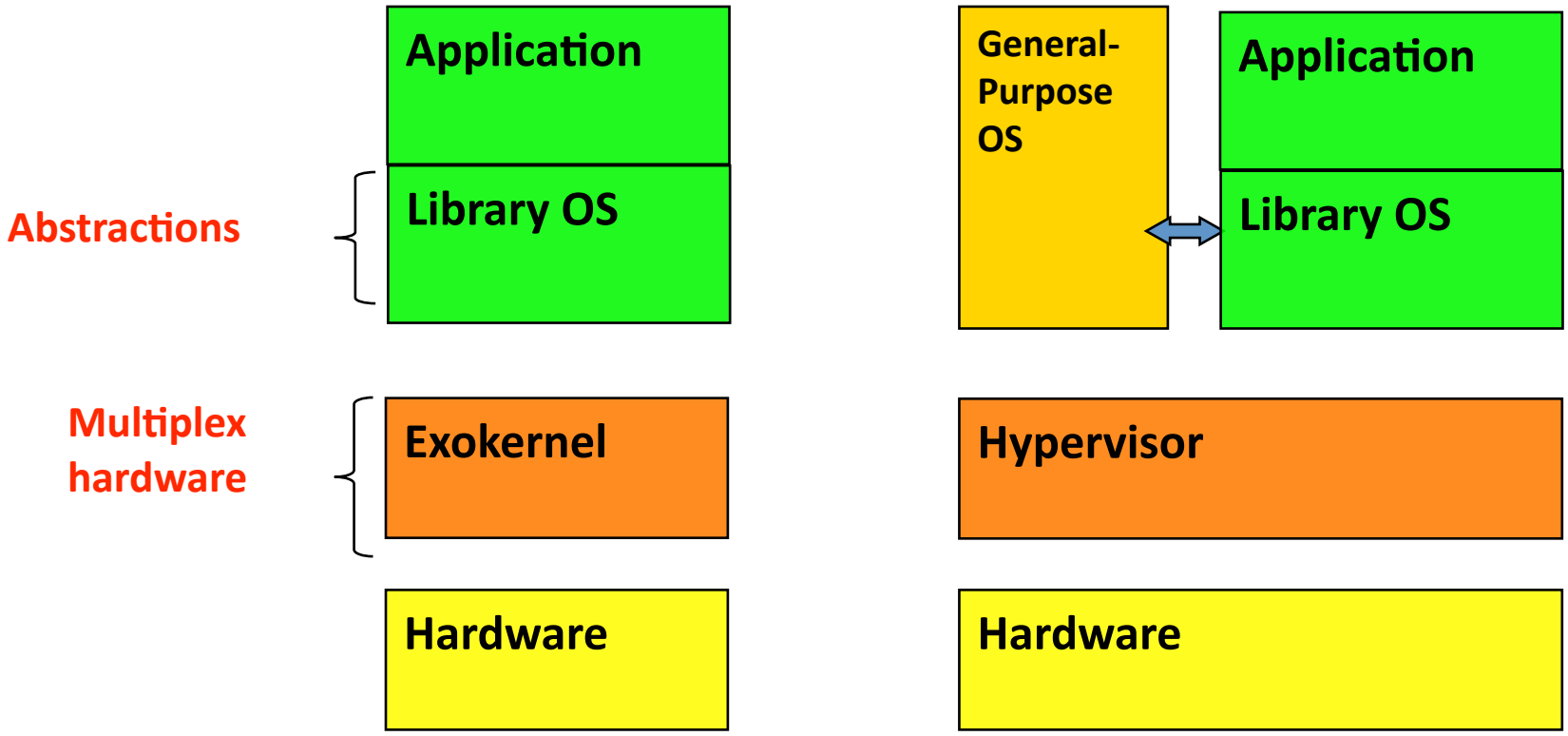

Windows
Linux
Solaris
MacOS
DD execution
DD execution
DD execution
DD execution


Windows
Linux
Solaris
MacOS
Hypervisor
DD execution
DD execution
DD execution
DD execution

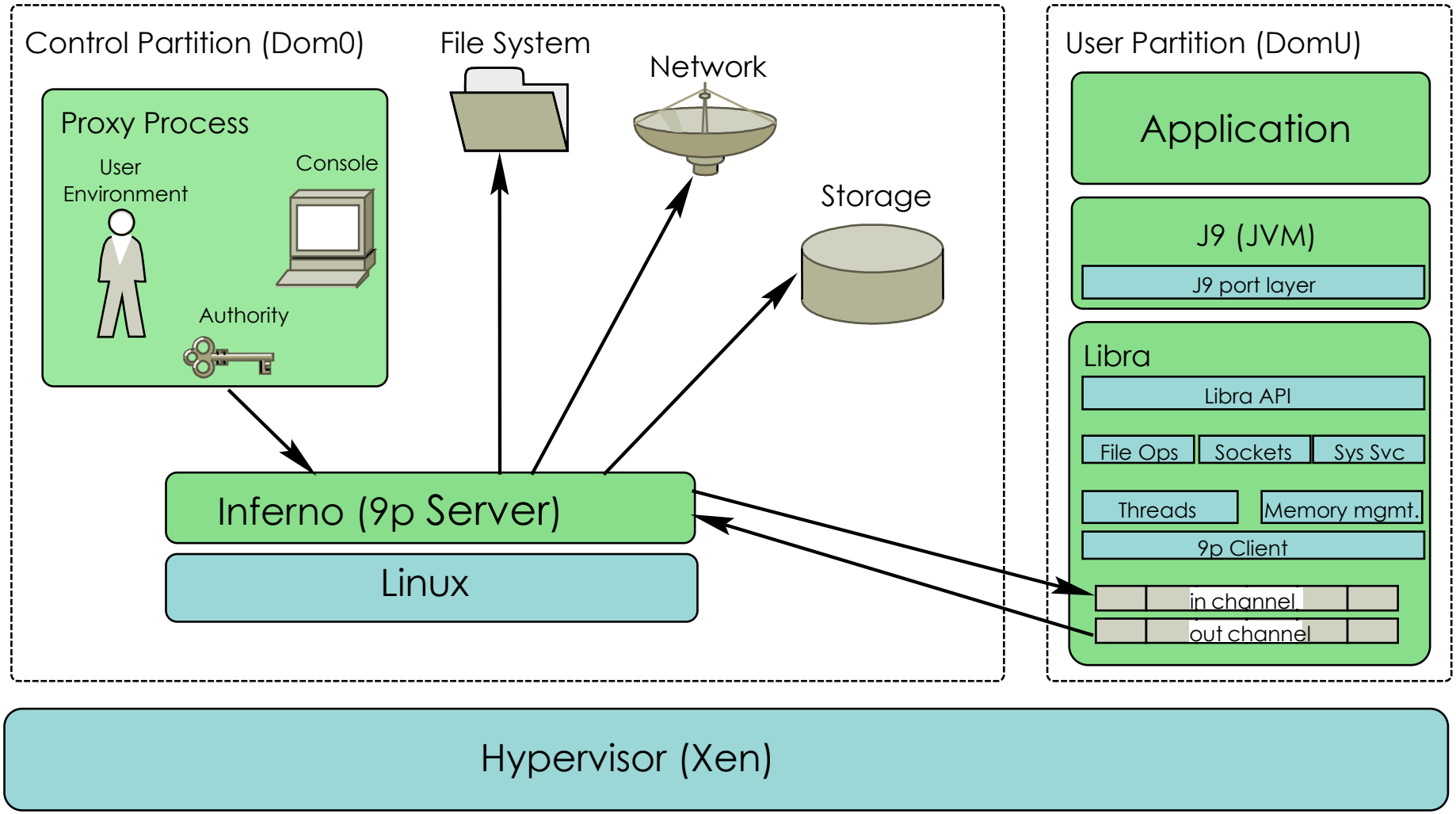


- Application's with reduced needs can be moved off of general purpose OS.
  - Application OS can be a reduced OS, or a highly customized library OS:
  - more easily exploit new HW, massive multi-core, extra blades
- Java applications require restricted interfaces, native code that invokes OS services can be shipped to legacy environment.
- Cluster services require highly deterministic real-time environment.
- HPC applications require specialized services (e.g., scheduling & memory management)
- Security services can be implemented with a reduced TCB

Example: Libra/Prose



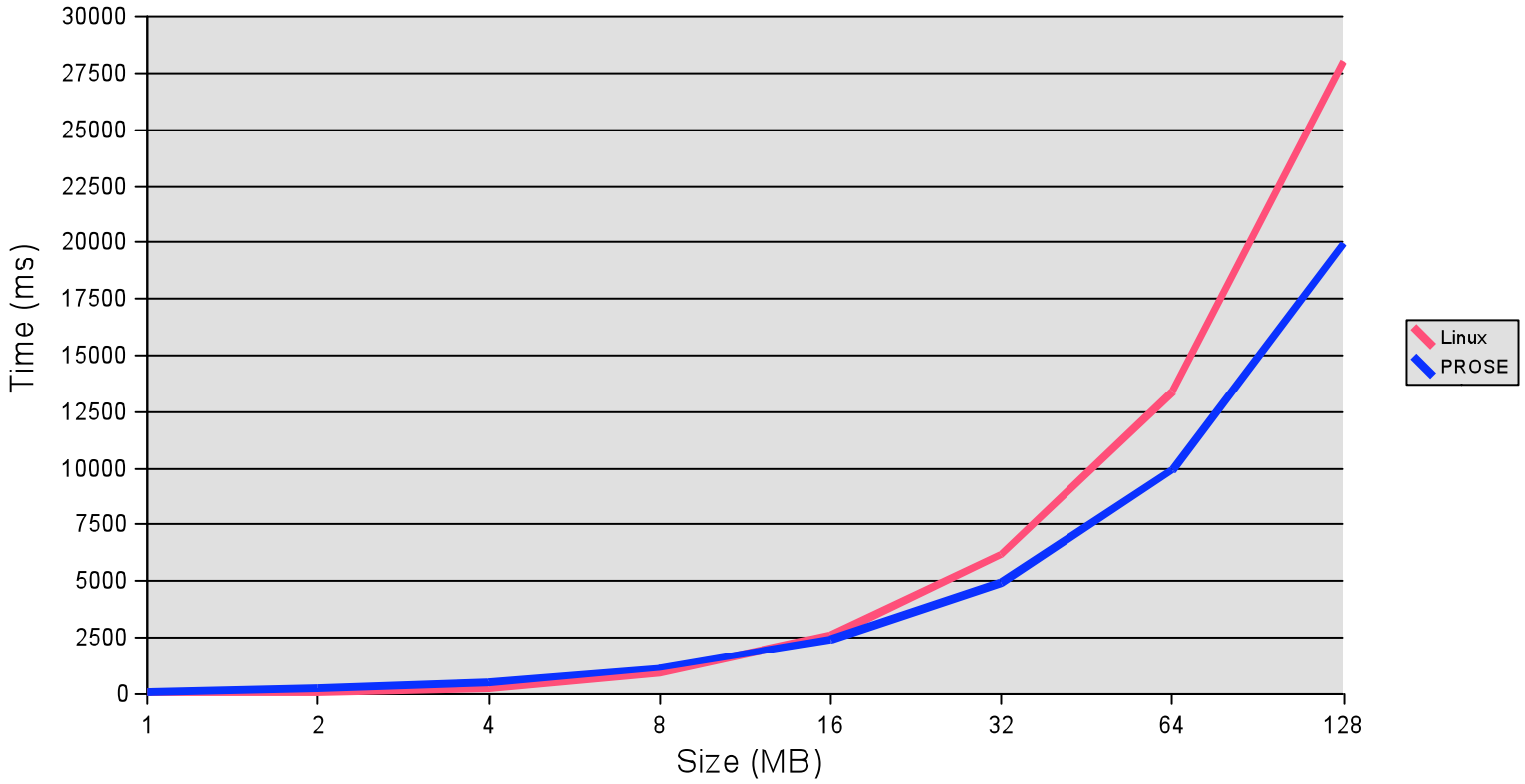
Prose and J9/Libra Architecture



9P2000 Characteristics

- Simple architecture-independent asynchronous RPC driven resource sharing protocol with built-in support for mutual authentication and encryption
- Only requires an underlying reliable, in-order interface
- Based on 11 primitive operations, most common file operations
- Integrated support for hierarchical namespace

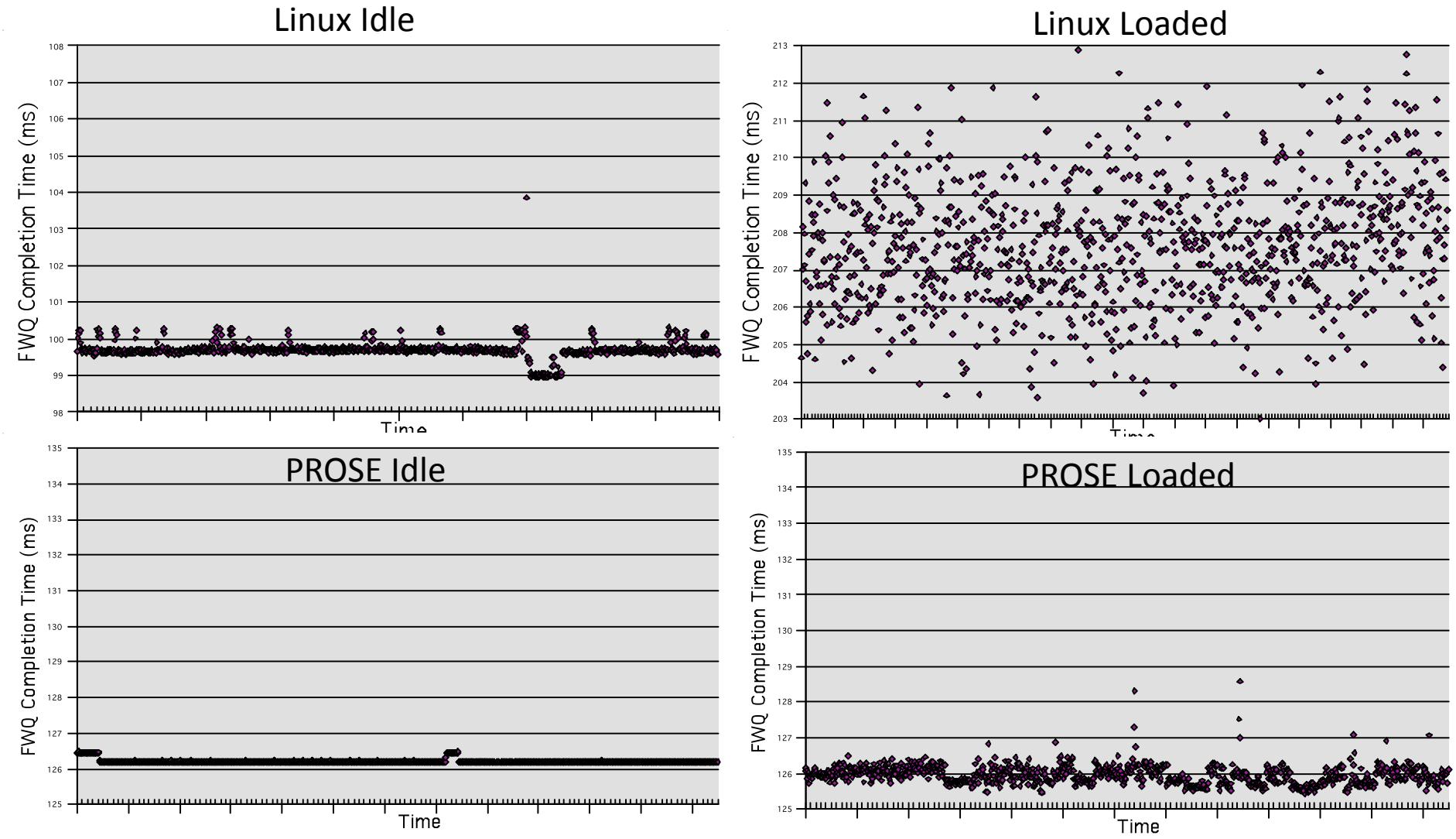
Sparse Memory Benchmark
Performance



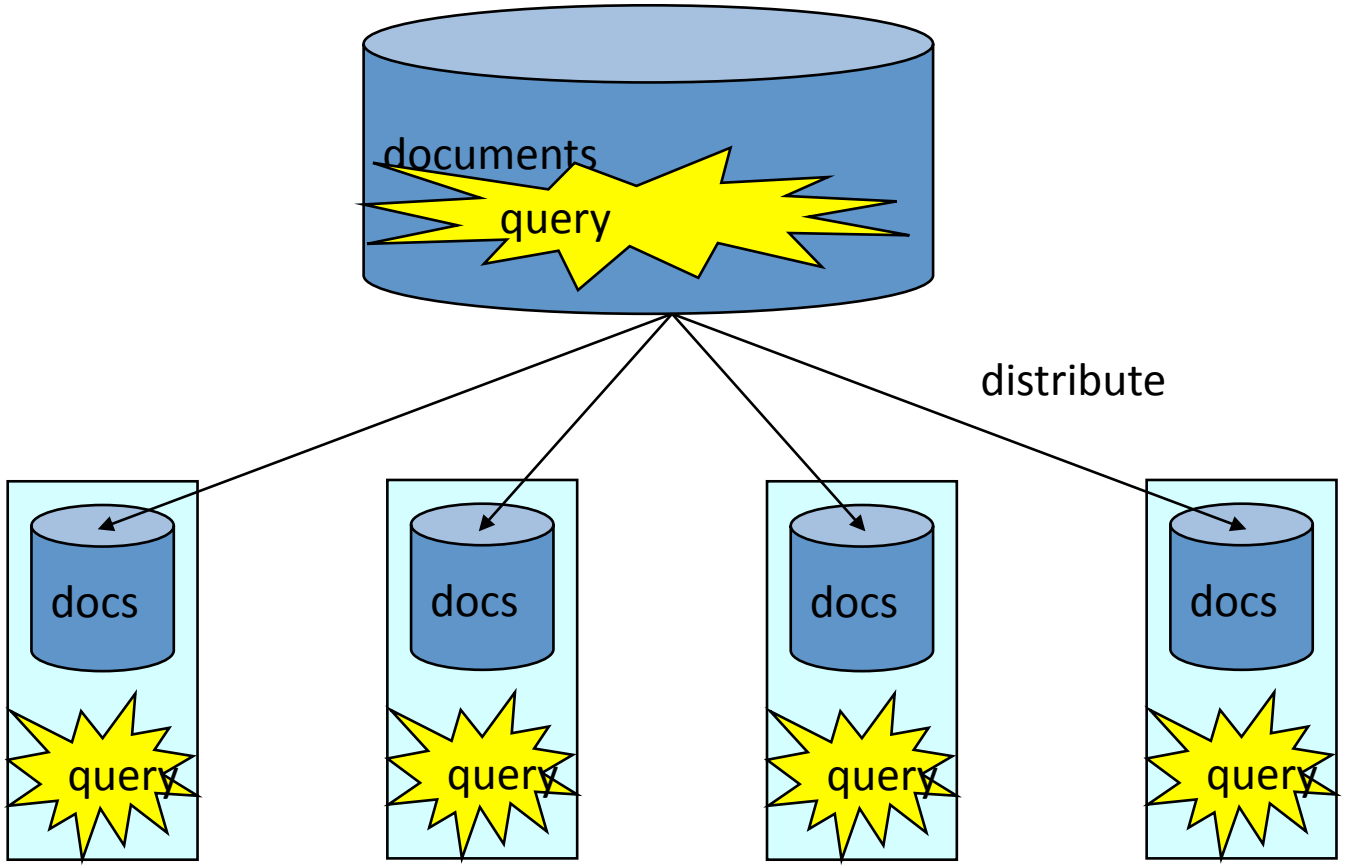
Noise Control w/PROSE & Hypervisors

- Allow strict control of percentage of CPU devoted to application versus system daemons and I/O requests
- Can eliminate jitter associated with interrupt service routines
- Provides a higher degree of determinism that vanilla Linux, but does so at a performance cost

Noise Comparison



Libra target workload: Java, Nutch/Lucene Query

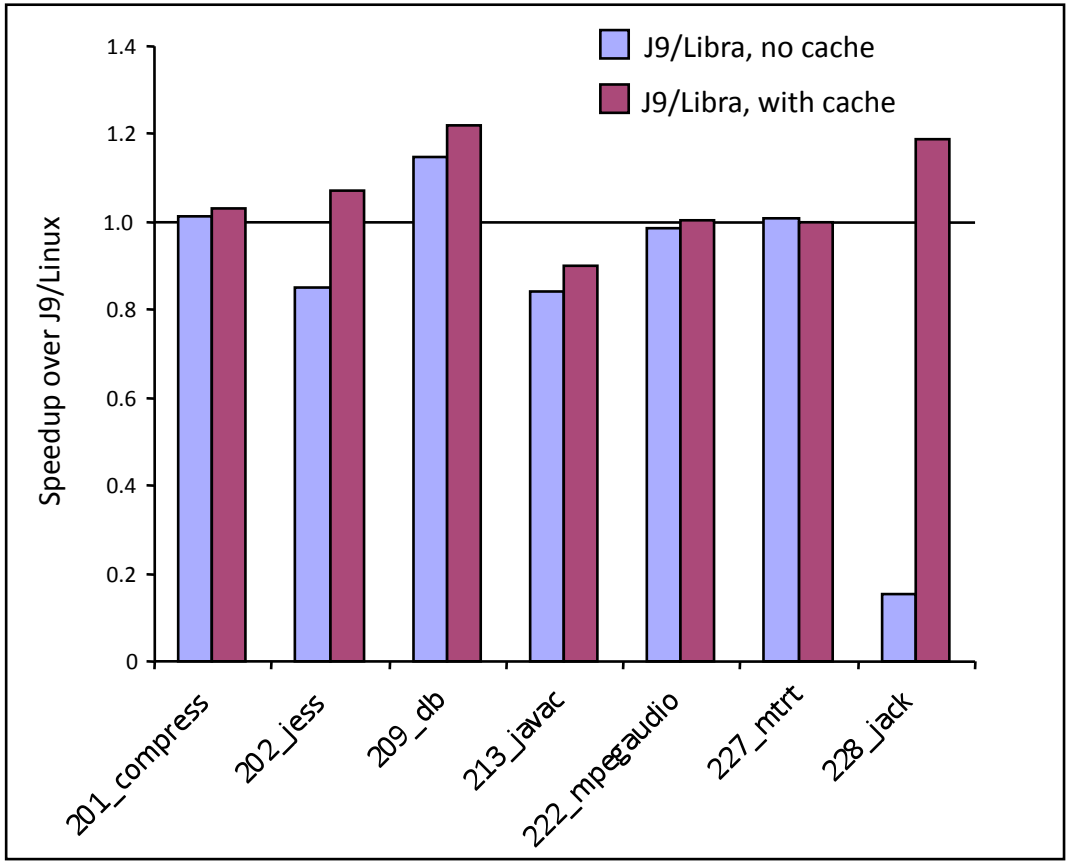


Optimizations

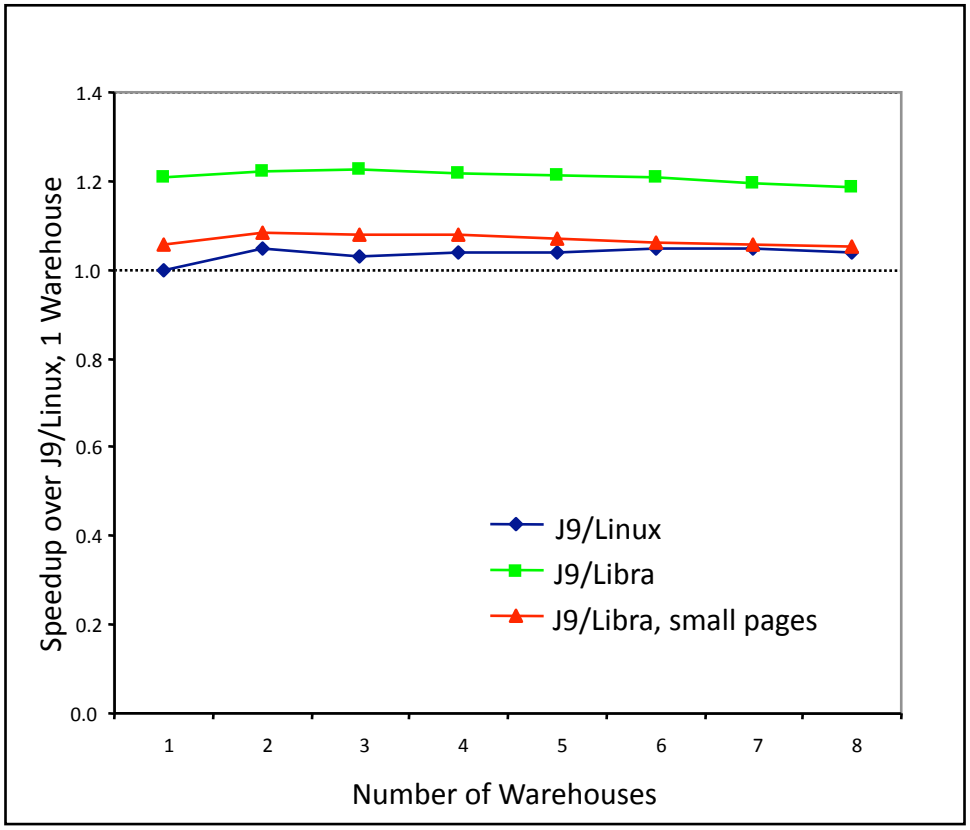
- File cache:
  - average lseek() & read() cost for back-end:
    J9/Linux: 2.25 usec
    J9/Libra: 0.9 usec
- Socket streaming:
  - stage socket data into/out of Libra partition
  - New requests are always available locally
  - Results are sent asynchronously in batches
- We never got to:
  - “safe-points” to support type-accurate garbage collection
  - Real-Time Java support
  - TLB control
  - …

SPECjvm98

- **jess & jack: file cache**
- **javac: open/stat**
- **db: large pages**

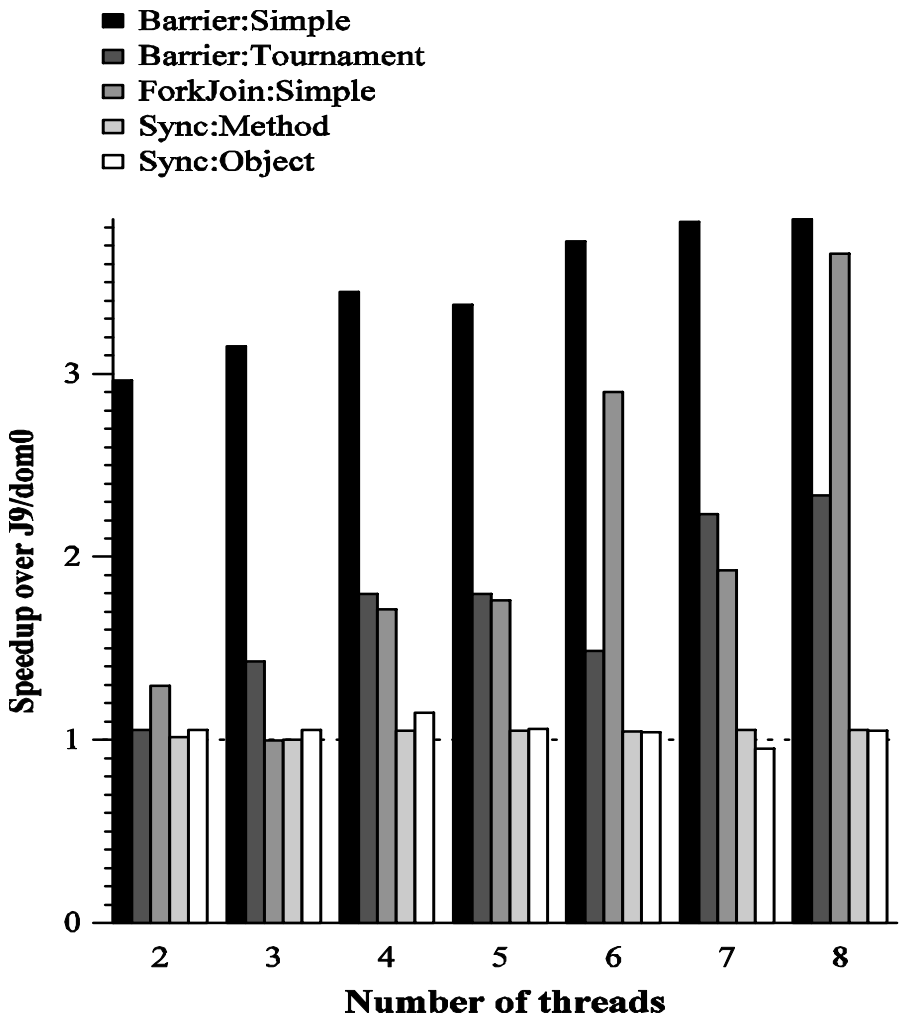


SPECjbb2000

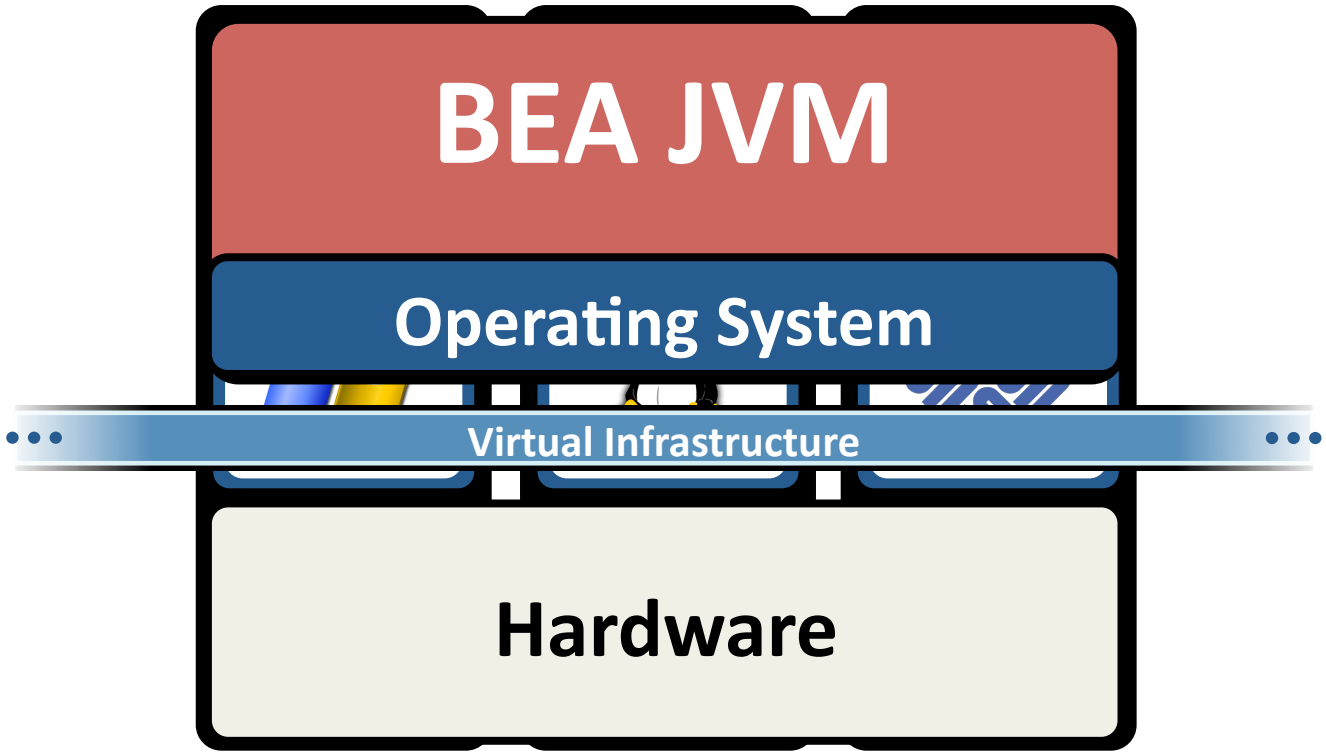


Java Grande Forum: Multi-threaded Benchmarks

J9/Libra vs J9/Linux

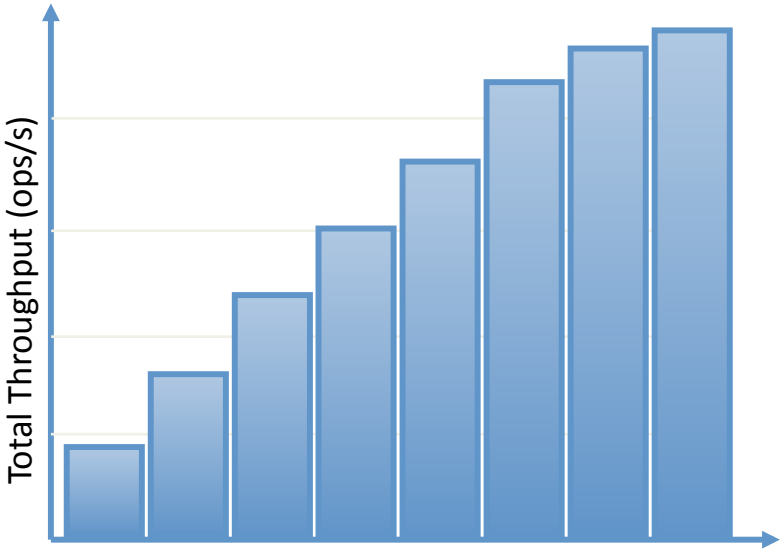


BEA Liquid VM



50% less memory consumption

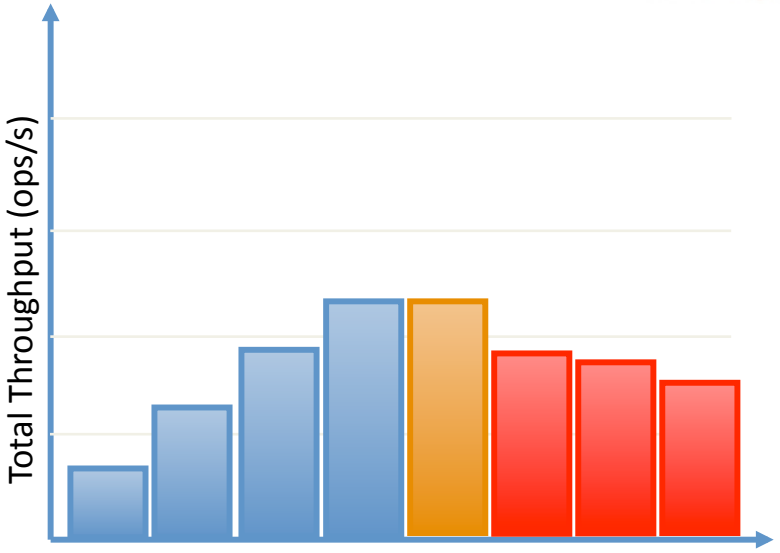
Do More Total Work With WLS-VE in LiquidVM Over Standard Virtualized OS Container



Java on LiquidVM

Virtual Machines

1 2 3 4 5 6 7 8



Java on Linux

Virtual Machines

1 2 3 4 5 6 7 8

- 3-Tier client-server Java benchmark measuring num. of business transactions per sec.
- Intel Xeon 3.2 GHz, 2 GB RAM, VMware ESX 3.0, BEA LiquidVM 1.1, BEA JRockit R27.3, BEA WLS 9.2 MP2, RHEL 4.0.
- Each VM allocated 1 vCPU and 1 GB vMem. JVM -Xmx=800MB, 135 MB live data.

Research

- How tiny for grid applications?
- Scalable deployment for HPC.
- Communication protocol.
- Developing library OS that is re-usable.
  - Minimizing library dependencies.
- Scalability for massive multi-core.
- Migration to and from generic OS.
  - Real time
  - Scalability
- Control of TLB for managed code
- Code and file system sharing

Conclusions

- Virtualization is going to be ubiquitous
- Gives us the opportunity to re-vitalize OS research.
  - Ability to deploy new innovations without the problems we had previously in the OS community.
  - New model for data center as a utility.
  - New model for application specific OSes