A down-to-earth look at the cloud host OS

Malte Schwarzkopf Steven Hand


UNIVERSITY OF CAMBRIDGE

“[...] In this position paper, we make a passionate and necessarily opinionated argument [...]”

“The abstract accurately describes this paper! It is passionate and opinionated and full of sensibilities.”

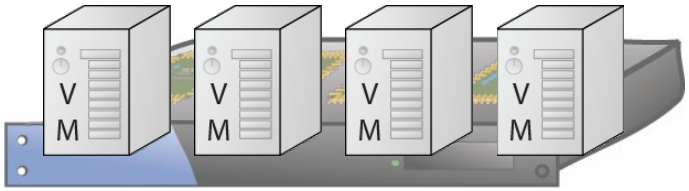
THE FOLLOWING **PRESENTATION** HAS BEEN APPROVED FOR

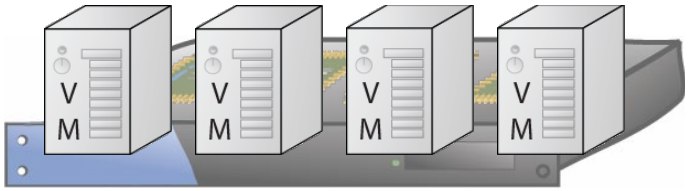
SELECTED AUDIENCES ONLY

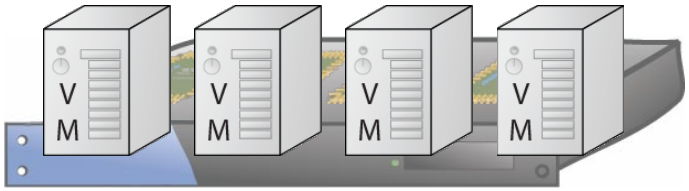
BY THE **AUTHORS**

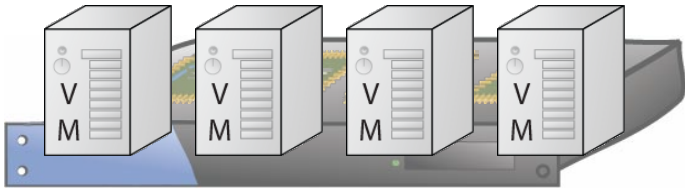
CO CONTROVERSIAL OPINIONS

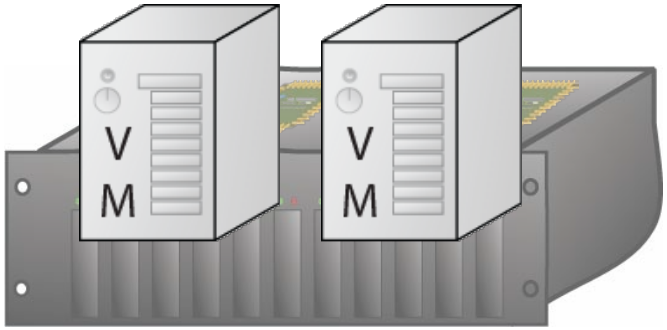
PROVOCATIVE, SPECULATIVE, OPINIONATED CONTENT

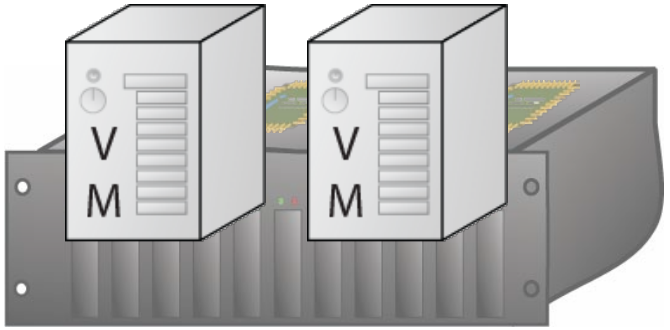

V
M
V
M
V
M
V
M

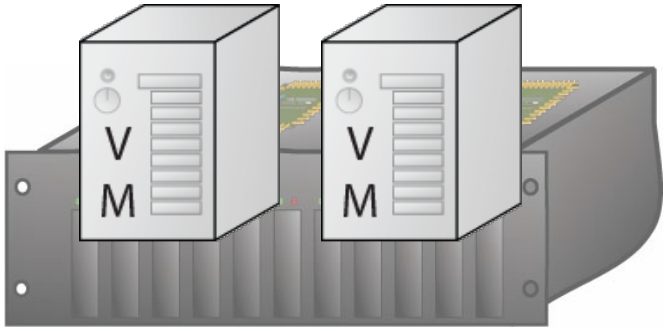

V
M
V
M
V
M
V
M

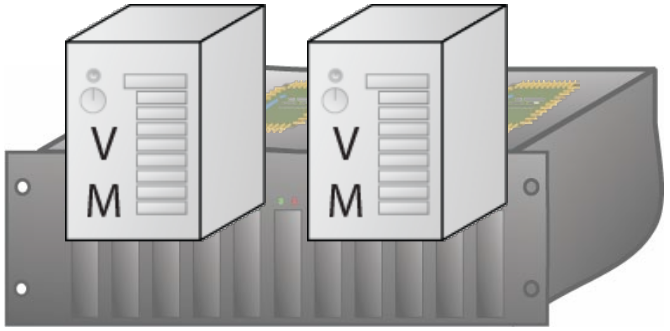

V
M
V
M
V
M
V
M

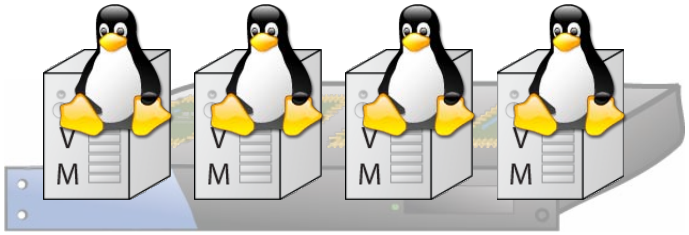

V
M
V
M
V
M
V
M

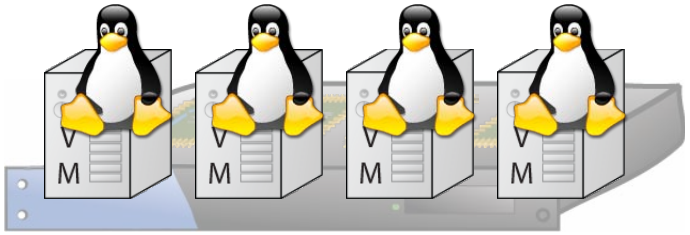

V
M
V
M

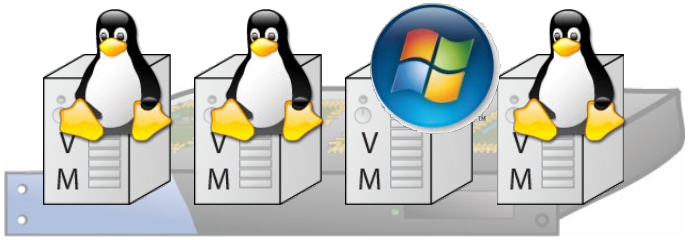

V
M
V
M

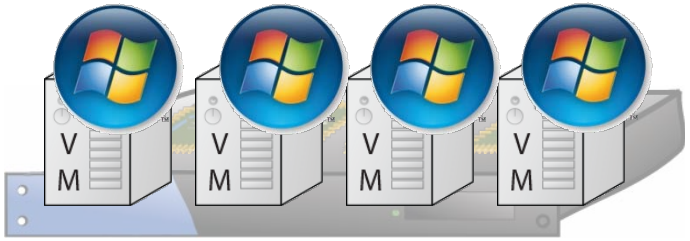

V
M
V
M

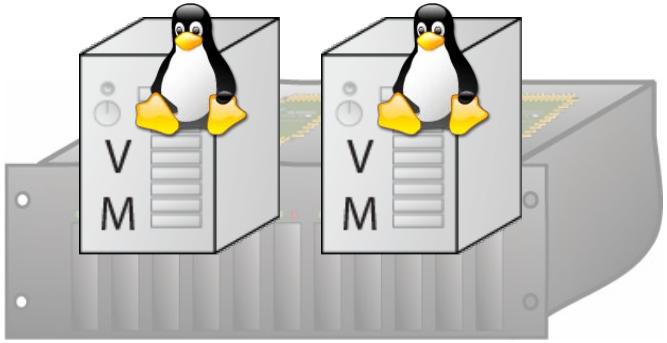

V
M
V
M

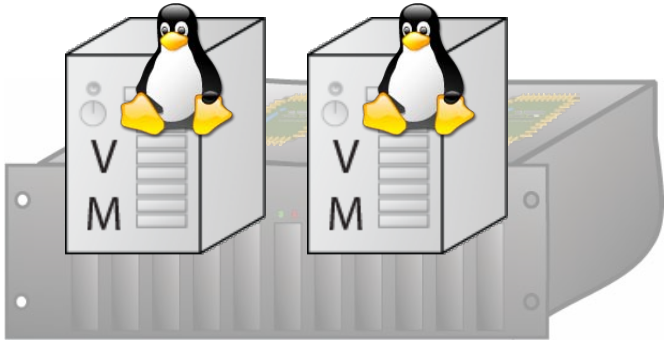

V
M
V
M
V
M
V
M

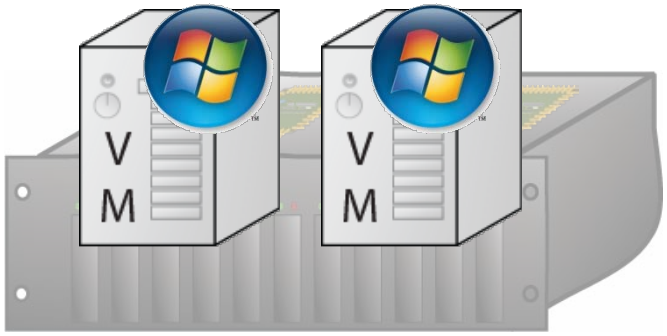

V
M
V
M
V
M
V
M

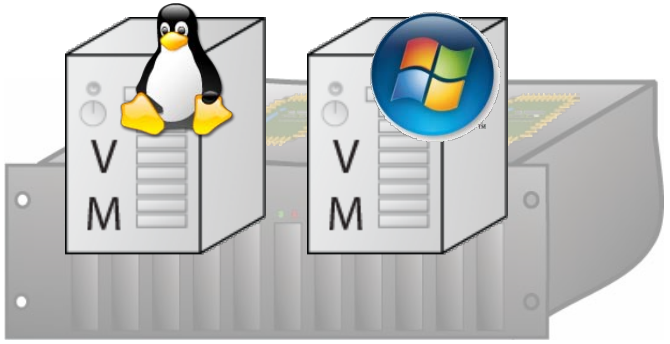

V
M
V
M
V
M
V
M

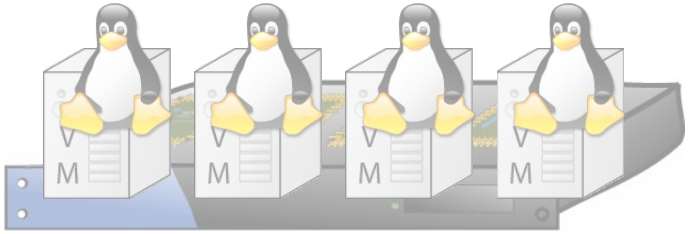

V
M
V
M
V
M
V
M

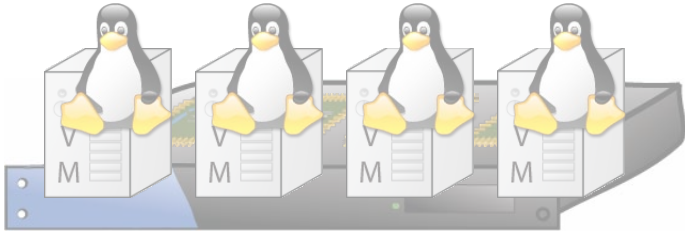

V
M
V
M

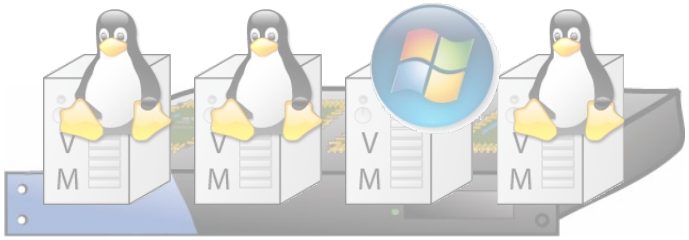

V
M
V
M

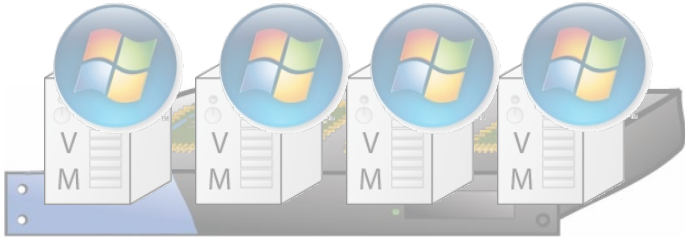

V
M
V
M

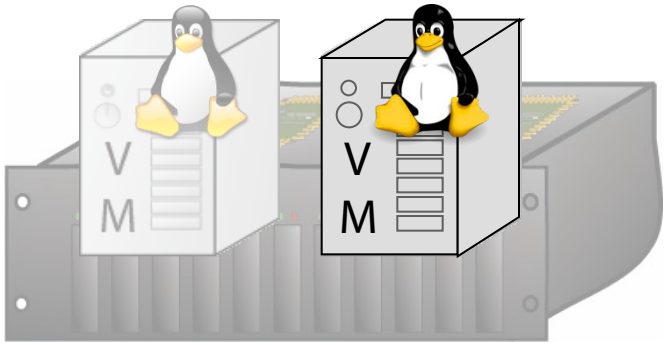

V
M
V
M

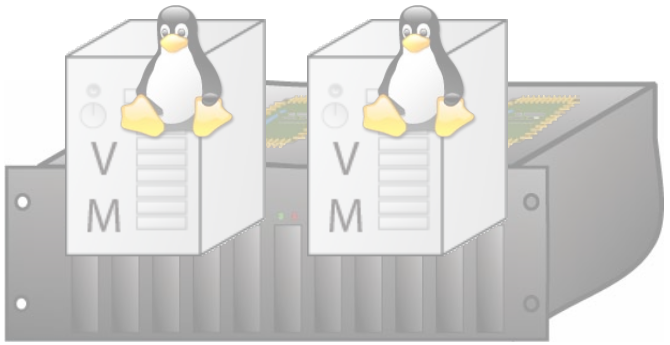

V
M
V
M
V
M
V
M

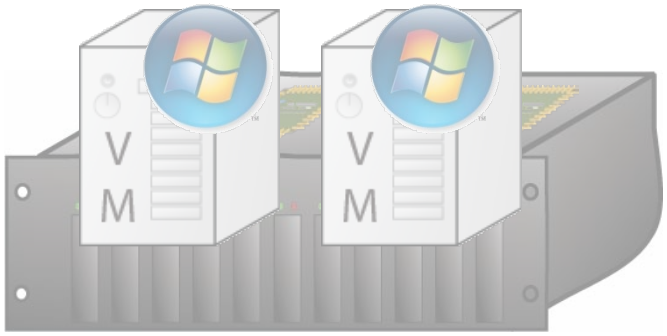

V
M
V
M
V
M
V
M

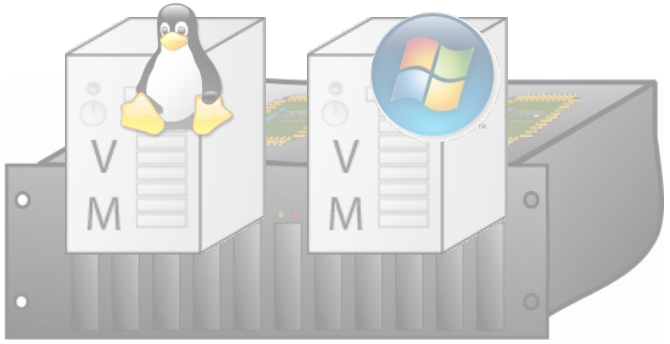

V
M
V
M
V
M
V
M

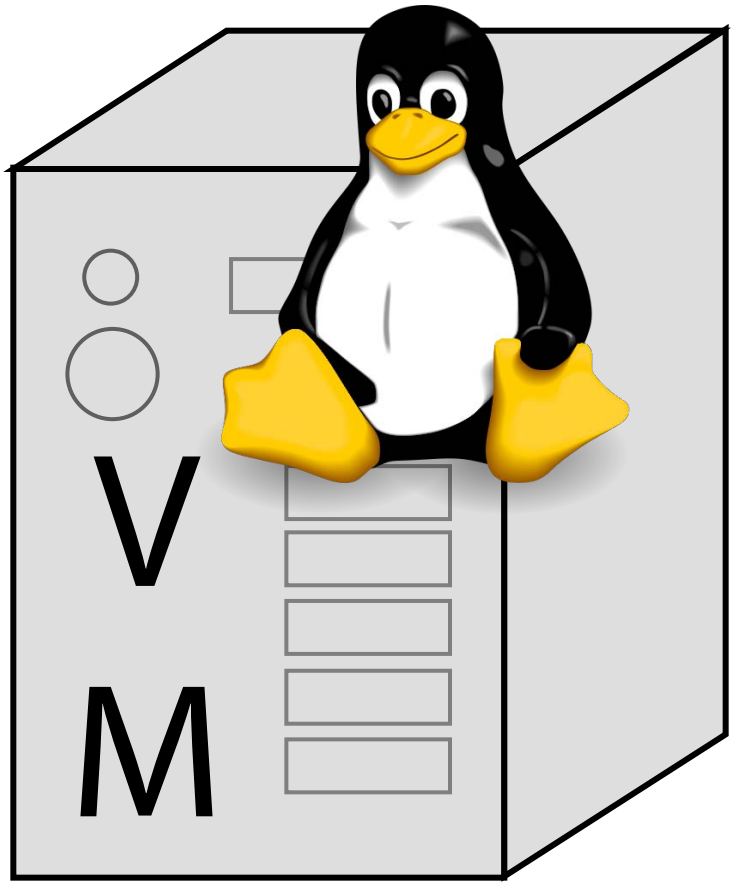

V
M
V
M
V
M
V
M

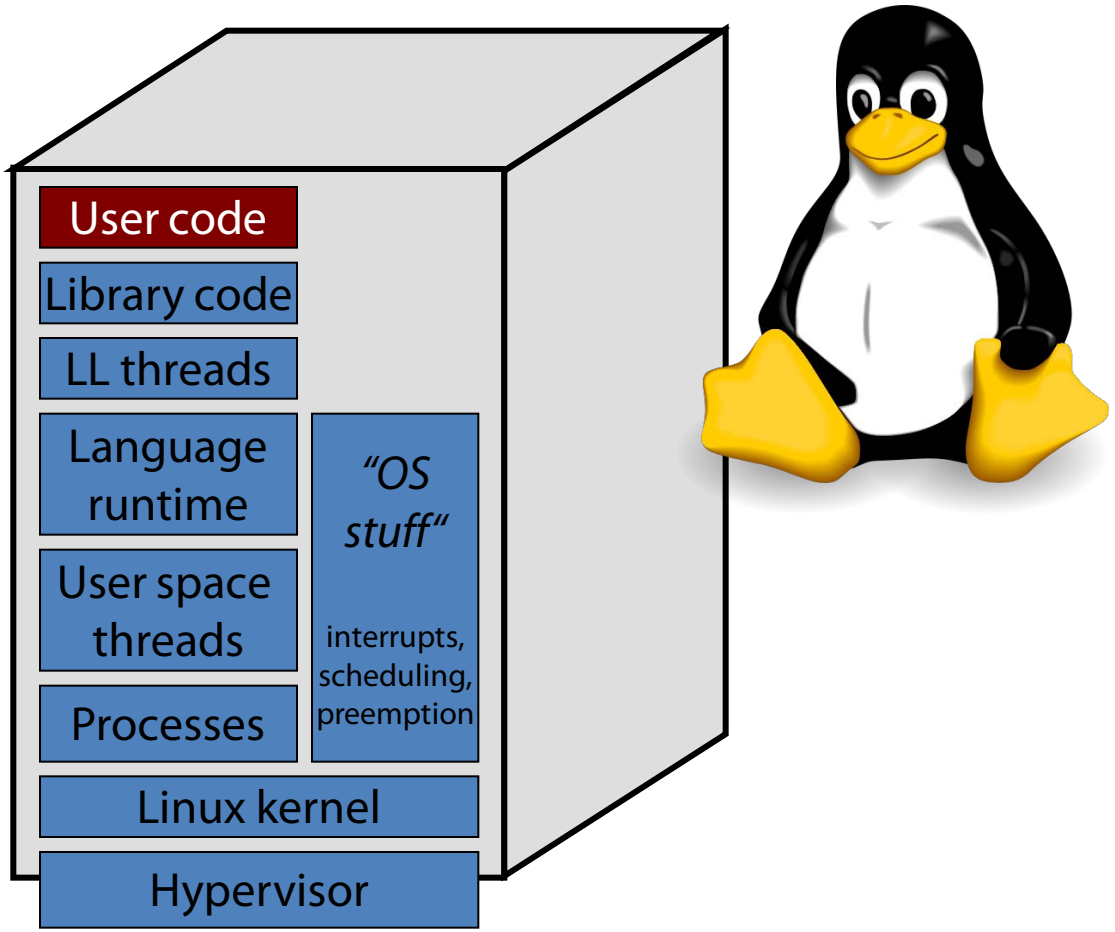

V
M
V
M


V
M
V
M


V
M
V
M


V
M
V
M


V
M


User code
Library code
LL threads
Language runtime
User space threads
Processes
"OS stuff"
interrupts, scheduling, preemption
Linux kernel
Hypervisor

Highly general

Familiar environment

Existing tools

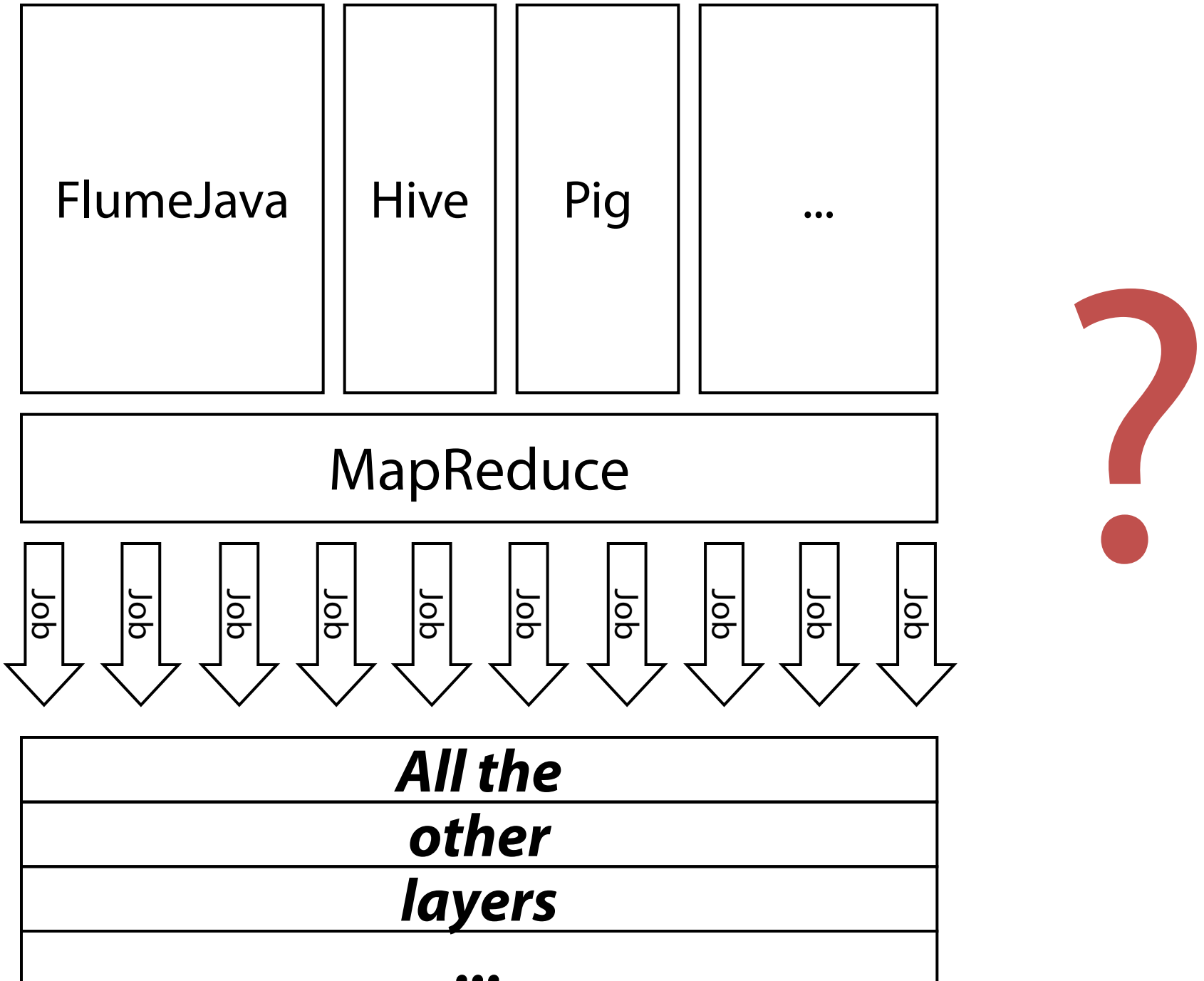
Large base images

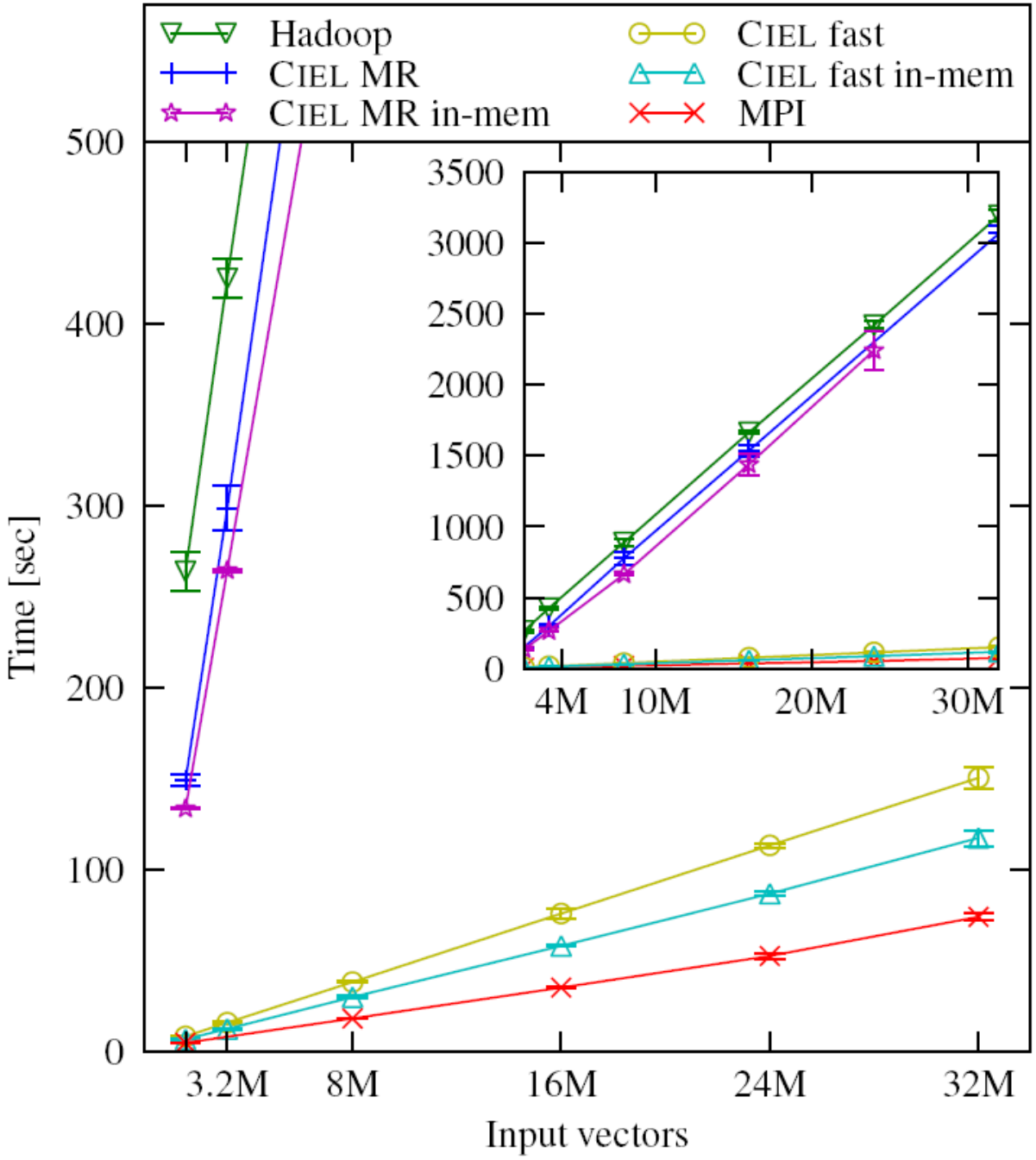
Slow to spawn

Death by generality

Layering

[and yet, the programming models are often restrictive!]

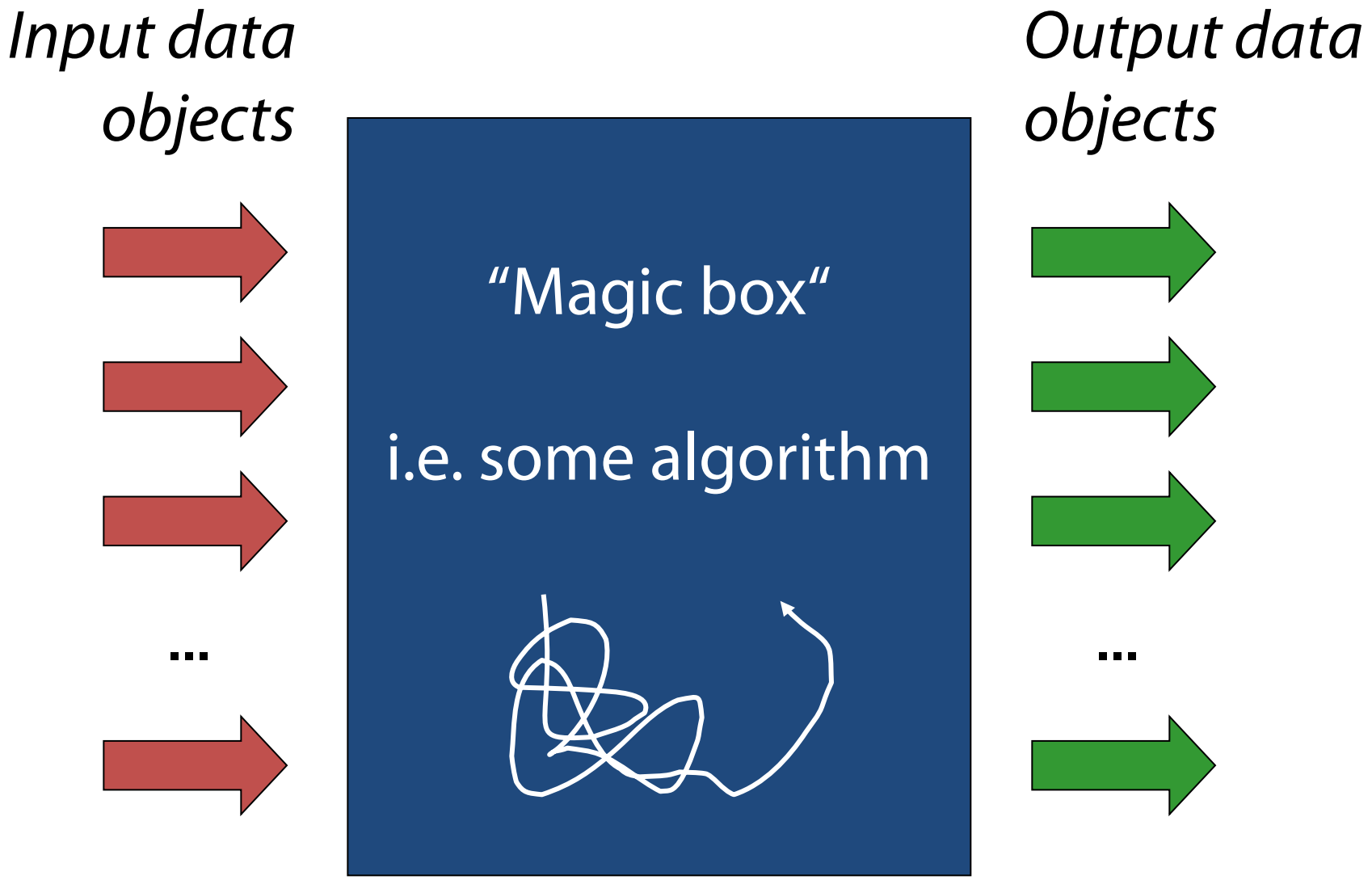

FlumeJava
Hive
Pig
...
MapReduce
Job
Job
Job
Job
Job
Job
Job
Job
Job
Job
All the
other
layers
...



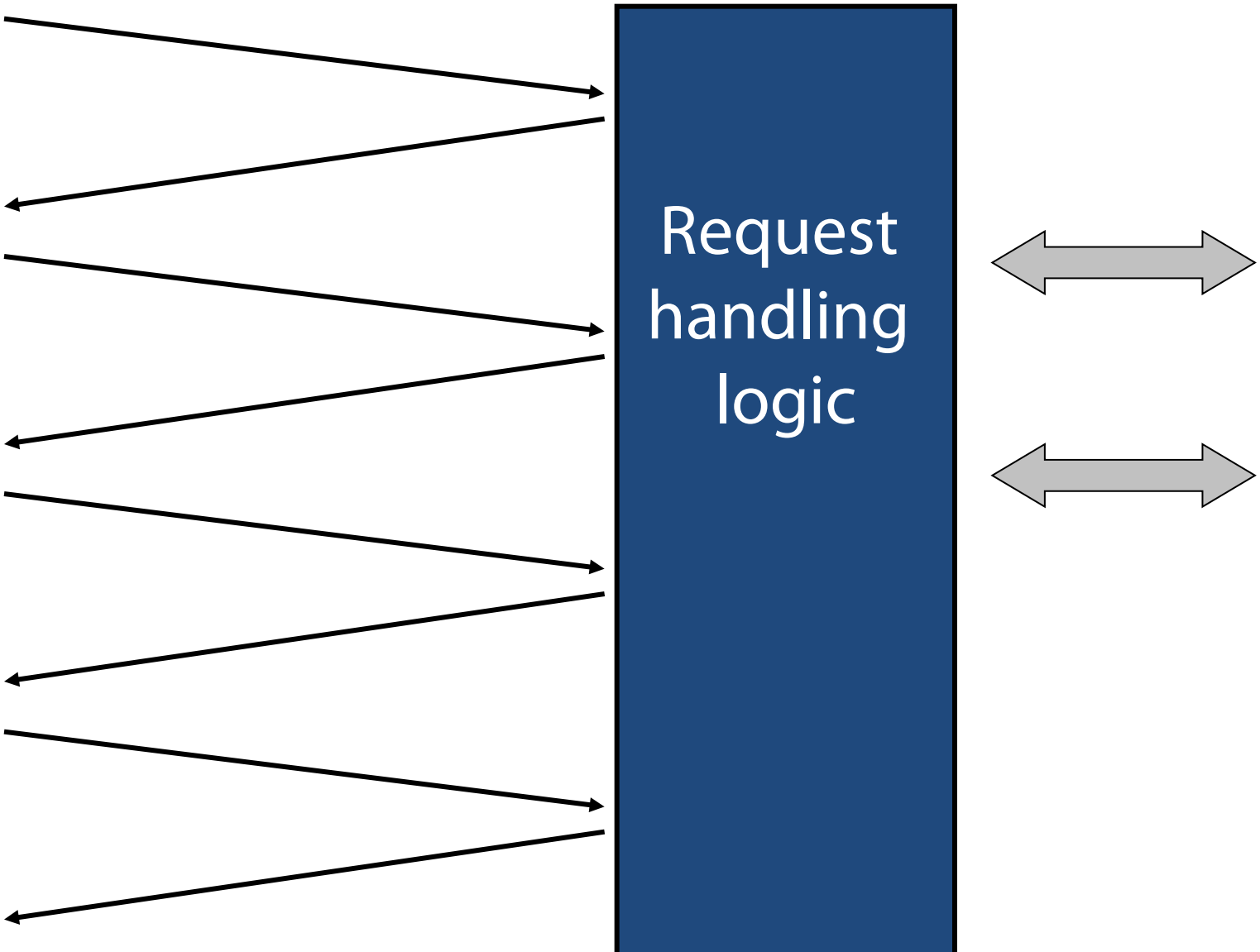
Experiment from D. Murray,
A distributed execution engine supporting data-dependent control flow.
PhD thesis, University of Cambridge, 2011.

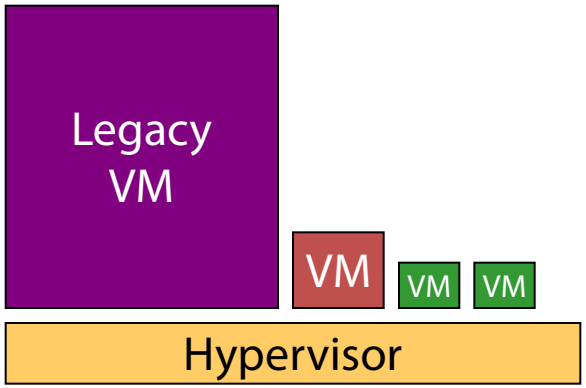
What do we **really** need?

Batch


Input data objects
Output data objects
"Magic box"
i.e. some algorithm
...
...

Serving

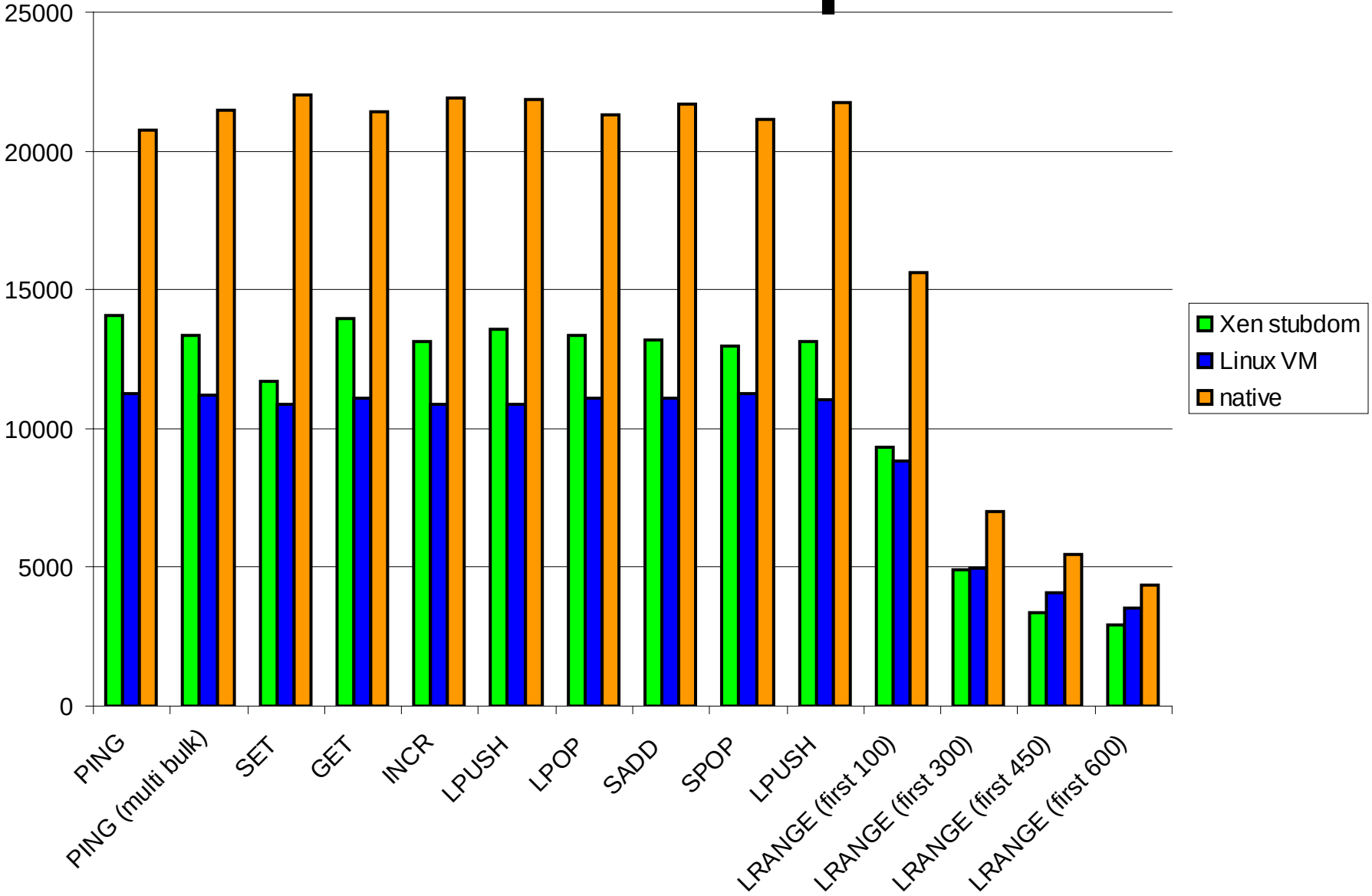




Virtualize custom μVMs

Back to the Eighties!

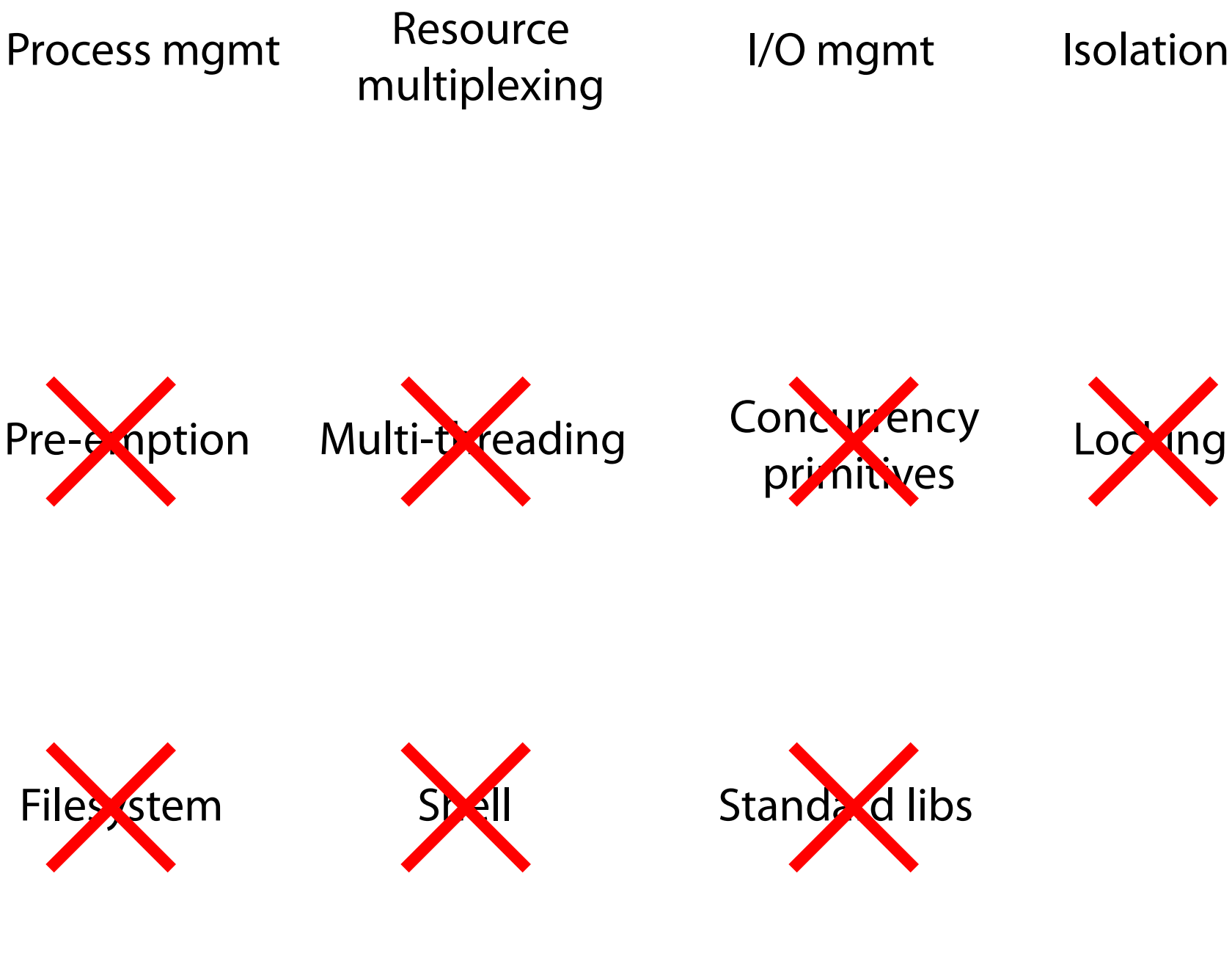
Redis example



Numbers and experiment by Sören Bleikertz: http://openfoo.org/blog/redis-native-xen.html

Mantra:

Make the OS do exactly (and just) what is needed.


Process mgmt
Resource multiplexing
I/O mgmt
Isolation
Pre-emption
Multi-threading
Concurrency primitives
Locking
Filesystem
Shell
Standard libs

Execution control

Resource management

Isolation

Data access

Execution control

Non-preemptive scheduling

Dedicated cores

Centralize I/O mgmt

Statically link all user binaries

Resource Management & Isolation

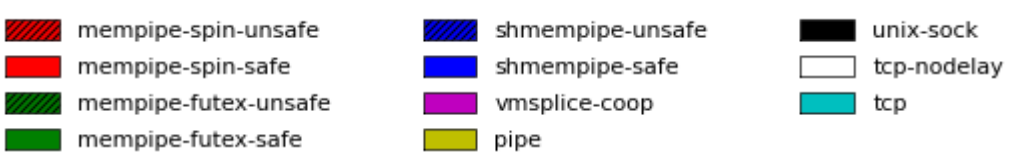
Principle of OS buffer mgmt: request/commit

Request/commit interface

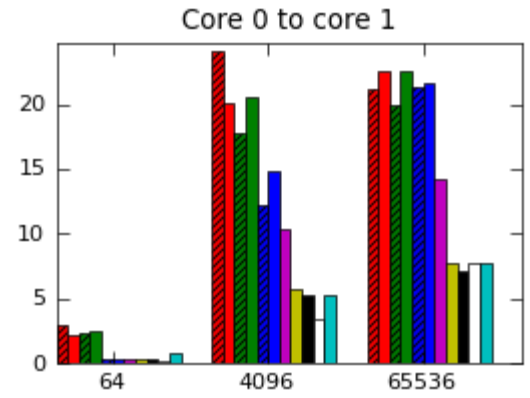
Backpressure for fairness

Embrace hardware heterogeneity

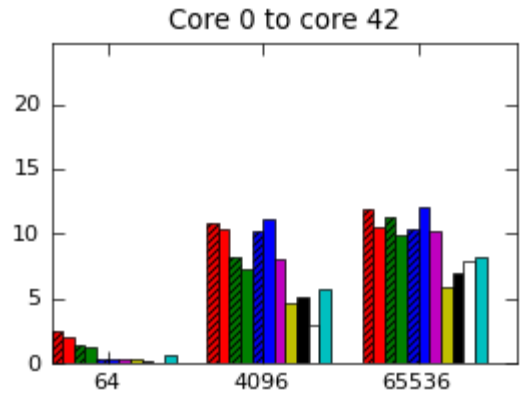
Resource Management & Isolation



AMD Opteron 6168

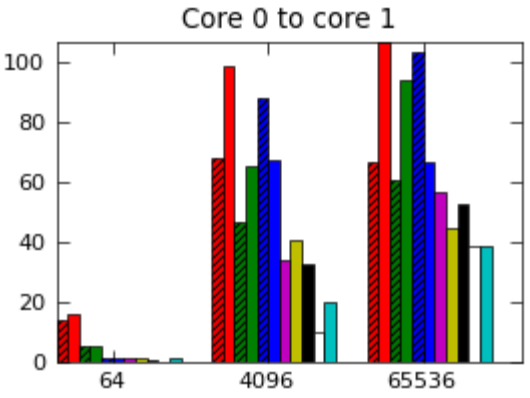


Same NUMA node

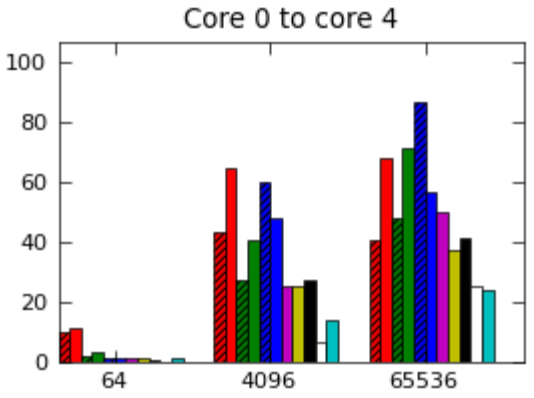


Different NUMA node

Intel i7-2600K



Different phys. core



Hyperthread

Data access

"Data object" abstraction

Global, deterministic naming

Transparent DO & buffer mgmt

[N.B. binaries are just DOs, too!]

Data access

- Capabilities
- Consistency levels

“People should not need to know about OS-level stuff in order to program the cloud!“

I hear your cries…

“This is going to be a nightmare to program!“

Compiler support

New, bespoke toolchain

Simple interfaces

“Everything is a task“

Take-away:

How about we push the good things about MapReduce into the OS?

Take-away:

How about we restrict the OS to have simplicity and predictable performance?