mOS: An Architecture for Extreme Scale Operating Systems

Robert W. Wisniewski, Todd Inglett, Pardo Keppel,
Ravi Murty, Rolf Riesen

Presented by: Robert W. Wisniewski
Chief Software Architect Extreme Scale Computing
Senior Principal Engineer, Intel Corporation

June 10, 2014

Legal Disclaimer

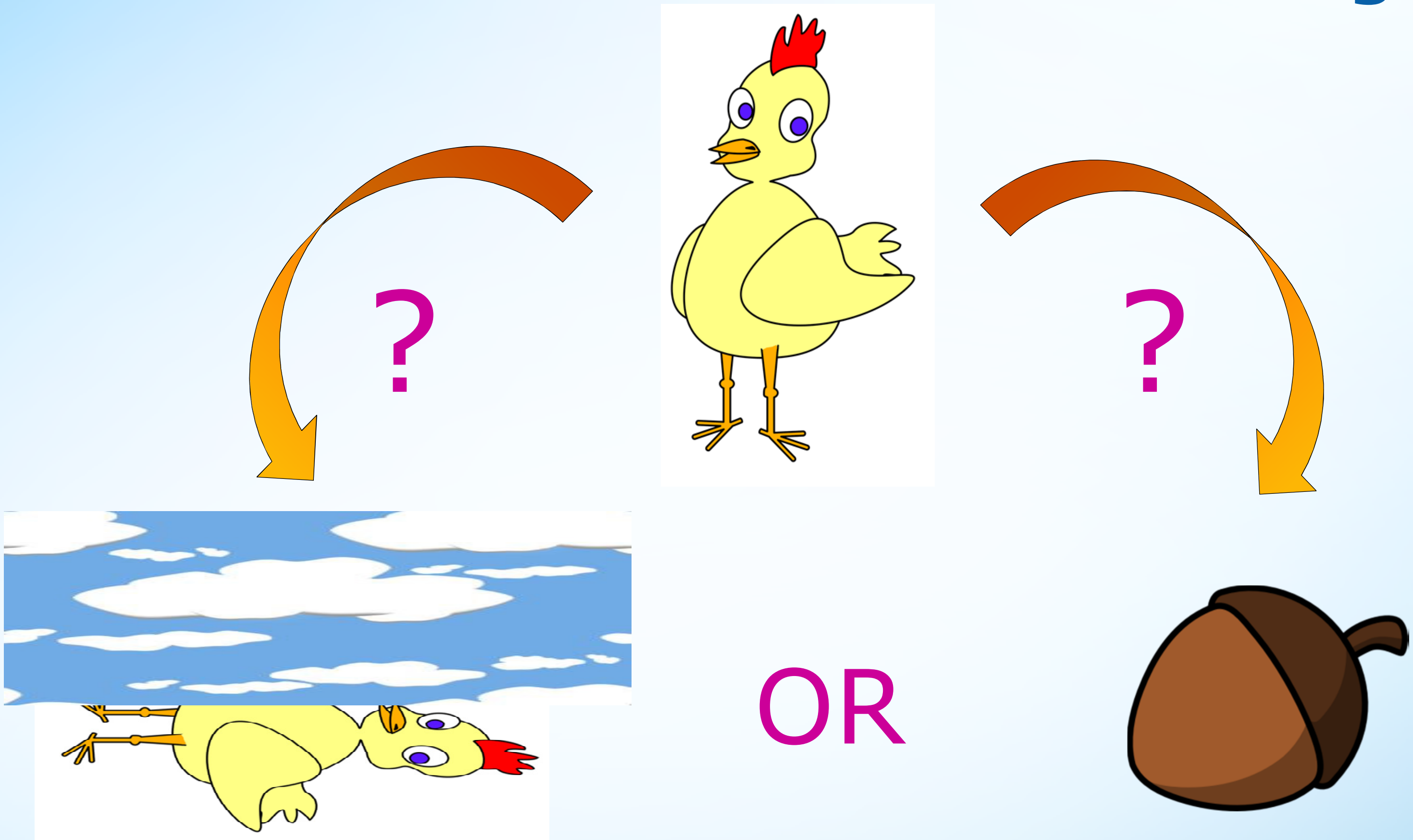
Results have been estimated based on internal Intel analysis and are provided for informational purposes only. Any difference in system hardware or software design or configuration may affect actual performance.

Intel, processors, chipsets, and desktop boards may contain design defects or errors known as errata, which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Agenda

- Extreme scale software challenges
- mOS
- Discussion
- Goal
  - High-level description of architecture ((some)details in paper)
  - ❖Motivate discussion

Extreme-Scale Software Challenge



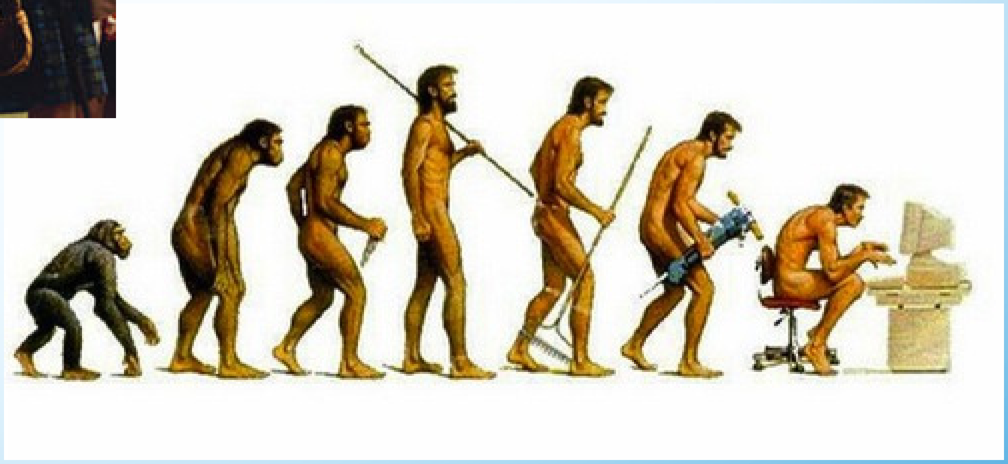
When investigations began

- Challenges too great with current SW
- Need all new OS, compiler, language…

Others advocated

- Enhance capability of existing
- Hard, drive evolutionary approach

Revolutionary versus Evolutionary



- Which one ?

Revolutionary

Imagine vendors telling their customers throw out everything you've done over the last 20+ years. Leverage tremendous investment in Intel Architecture ecosystem.

Evolutionary

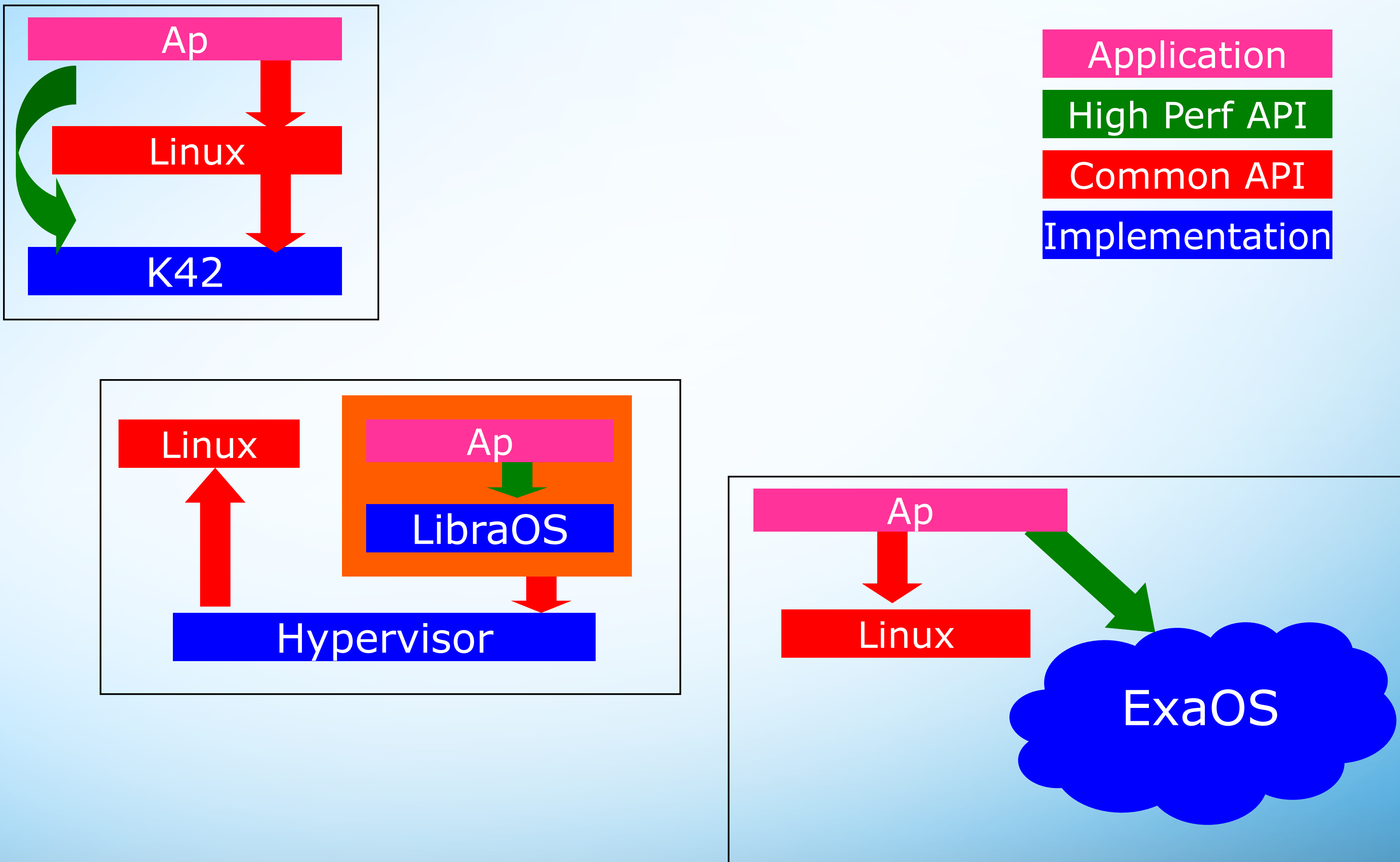
But there are serious challenges getting to exascale. Drive new innovations and invigorate the x86 ecosystem.

The Real Extreme-Scale Software Challenge

- The real challenge in moving software to extreme scale, and therefore the real solution, will be figuring out how to incorporate and support existing computation paradigms in an evolutionary model while **simultaneously** supporting new revolutionary paradigms.

AND

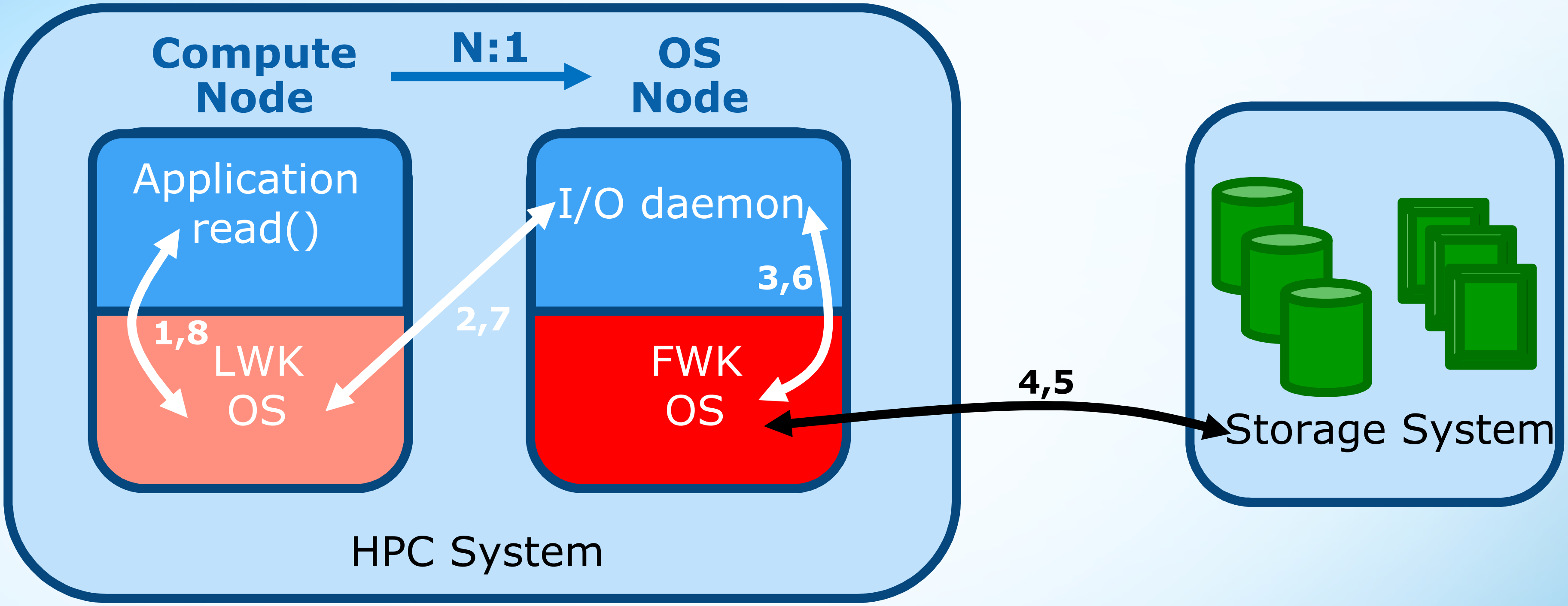
Operating System Example



mOS: Operating Systems Technical Drivers

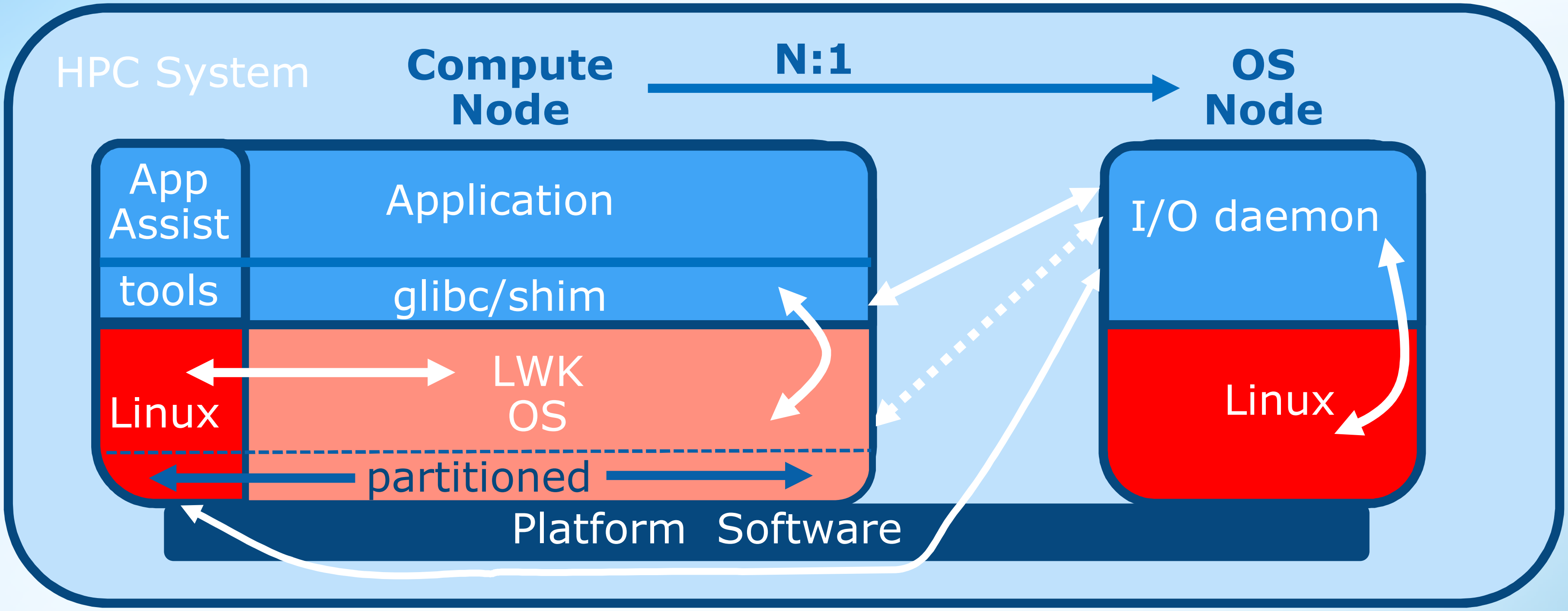
- Vision and technical direction
  - mOS that supports Linux® API and ABI (from app perspective)
    - LWK and Linux working in unison (LWK on compute cores, Linux for compatibility)
    - Simultaneously support legacy (Linux API) and new high performance calls
  - Nimble to support new technology effectively
    - Hybrid memory, many cores, new core technology, etc.
  - Move to hierarchy of OS offload for scalability
  - Support fine-grained threading and asynchronous requests
  - Enable specialized networking
  - Provide support for and be amenable to running on differentiated cores
  - Free and Open Source Software (FOSS)

OS I/O Offload



Aggregation (file systems can not handle 100sK+ clients)
Noise reduction
Reduced cache and memory pollution

OS Expanded Compute Node View



- Components
  - LWK
  - Linux Kernel
  - Intranode connection
  - System call triage
  - Offload to OS Node
  - Partitioning

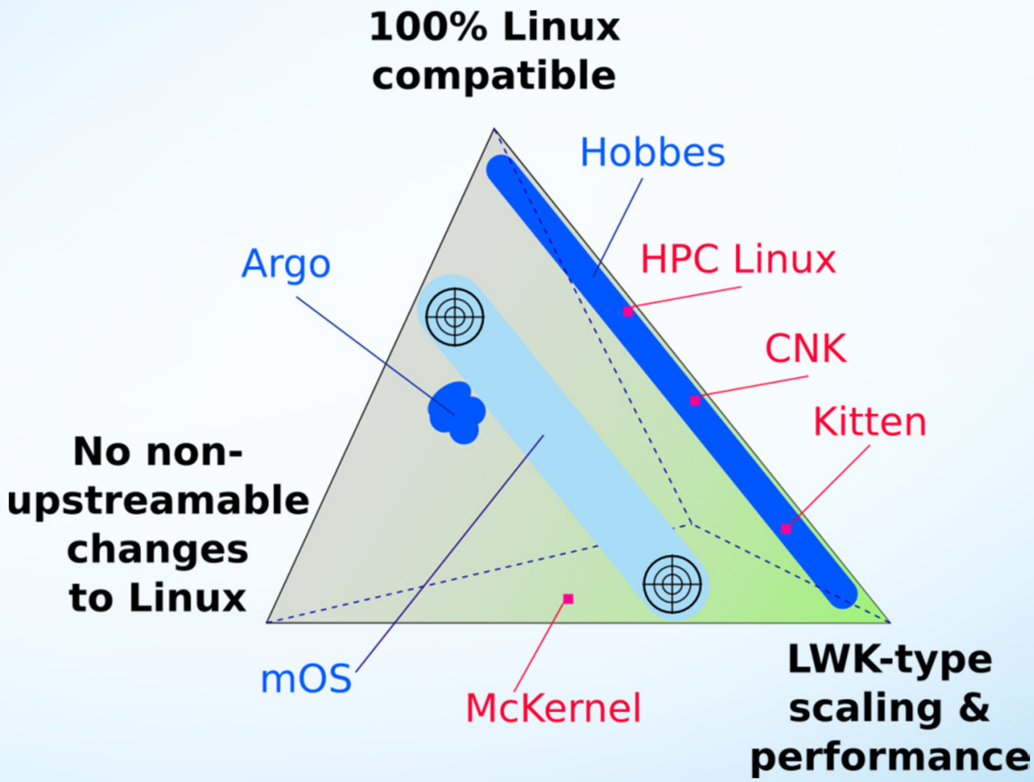
Related Work

- LWK vs FWK
  - Catamount, CNK, Kitten/Palacios, ZeptoOS
  - SGI® Linux, Cray® CNL from CLE (ESL vs CCM)

- Microkernels and virtualization
  - K42, LibraOS, Kitten/Palacios and Hobbes

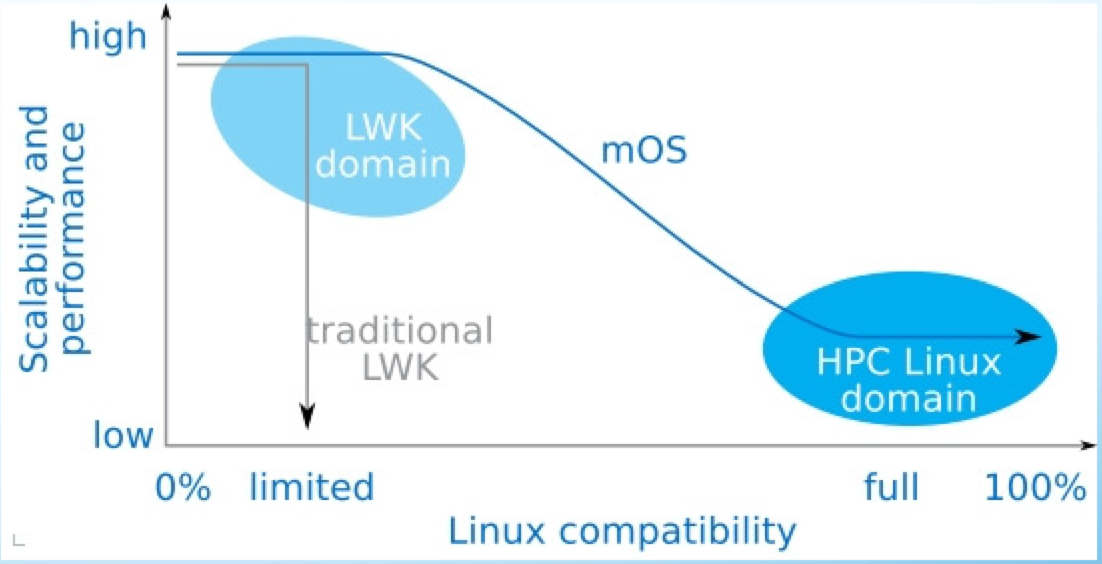
- Multiple kernels
  - Tesselation, NIX, Argo, McKernel, FusedOS

Tensions Pulling in HPC OS Design



Resolving the Tensions

- In the past it was possible to achieve high performance with ultra scalability. Or, one could run Linux. But not both.
- With an architecture like mOS, it is possible to have a more gradual path from the upper left LWK corner to the lower right FWK corner.
- An application's choice of which features it uses, influence its overall performance and scalability.



Advantages for HPC Applications

- Use large pages effectively
  - Use 1GB pages well, use 2MB well, use 4K minimally,
  - Don't age shoot-down pages the network has touched
- Provide specialized scheduling classes
- mOS will do the right things for scheduling
  - Will not take minutes to stabilize
- Guarantee globally symmetric addresses
  - Valuable for PGAS
- Low-latency network interrupts
- Use native transports for network traffic
- Easy exploration of mixed memory types
- Support new hierarchical memory architectures
  - Allocate based on bandwidth, latency, energy, and locality

Advantages for HPC Applications

- Could allow specialized simple hardware
  - Range mappings
- mOS can be quickly changed to meet new needs
- Thread placement to reflect workload and microarchitecture
  - Different cores at different distances from memory and network interfaces
  - Memory type optimization
    - In-package high BW memory, off-package DRAM and off-package NVRAM
- Can optimize specific system calls

Conclusions and Discussion

- mOS offers new OS architecture for future HPC and other
- Key points
  - mOS: OS architecture for hierarchical systems
  - Simultaneous support Linux API (FWK) and high-performance LWK
  - Nimbly leverage future generation chip technology
    - Heterogeneous cores
    - Hybrid memory
    - Tightly coupled networking
    - Chip architecture using transistors for specialized purposes
- mOS architecture mostly in place
  - Working to finalize architecture
  - Next half year focus on prototyping challenging areas
  - After that, implementation
- Discussion