Make It So !

A Software Paradigm for PEZ(Y) Computing

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

July 9, 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.

Intel, Intel Xeon, Intel Core microarchitecture, and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

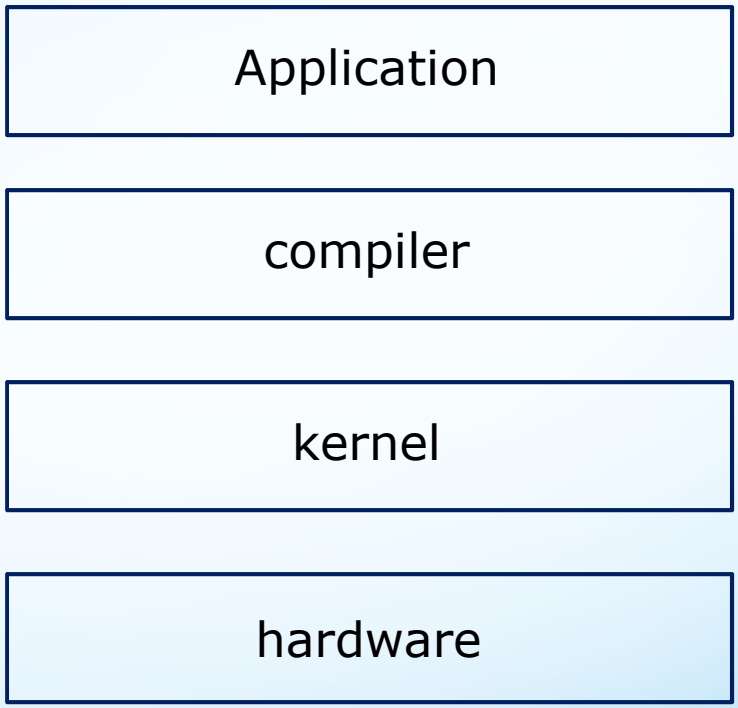
*Other names and brands may be claimed as the property of others

Motivation for this Talk

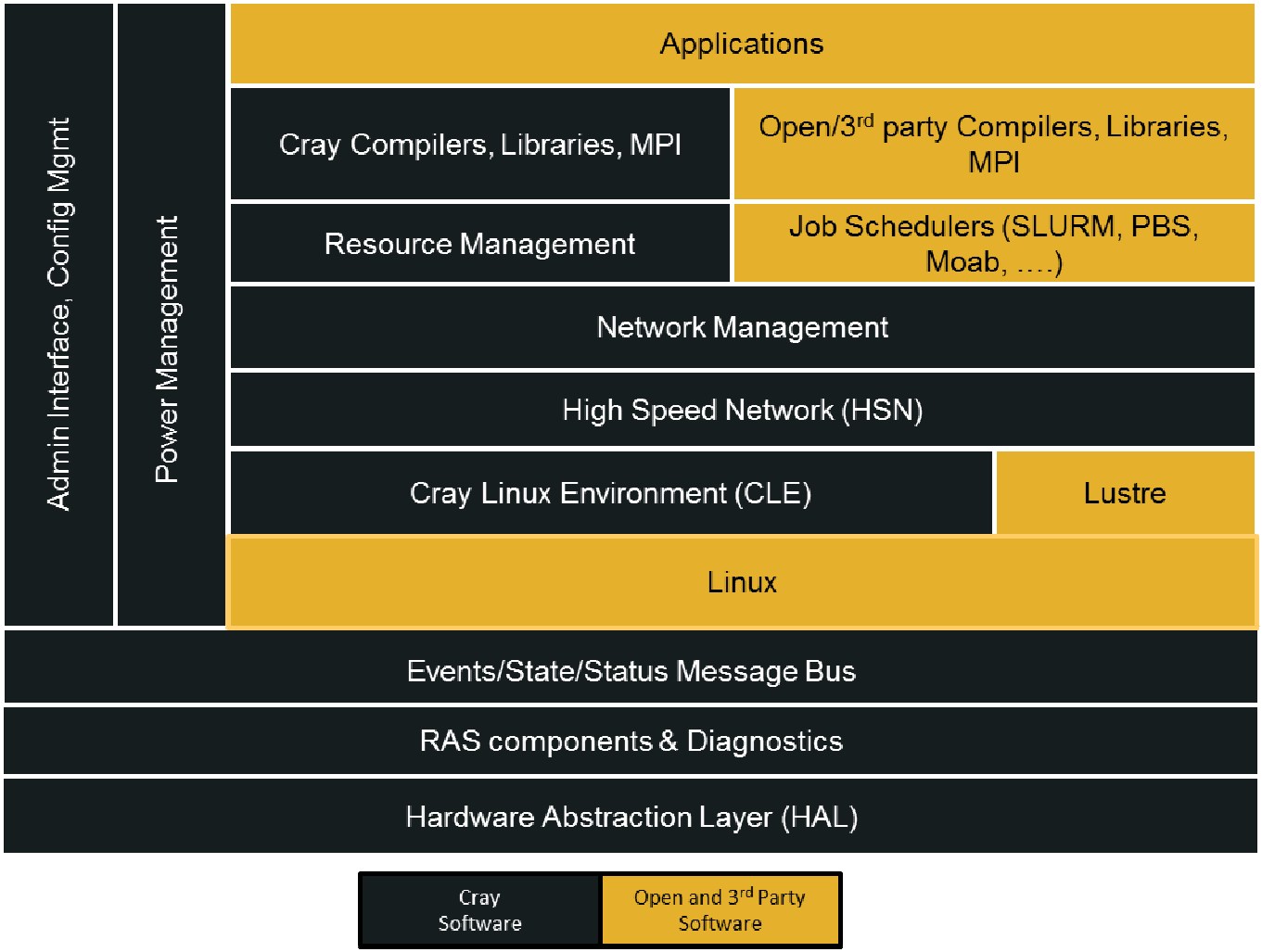
- Previous talks
- Abstract submission
- Grounded right track

Motivation for this Talk

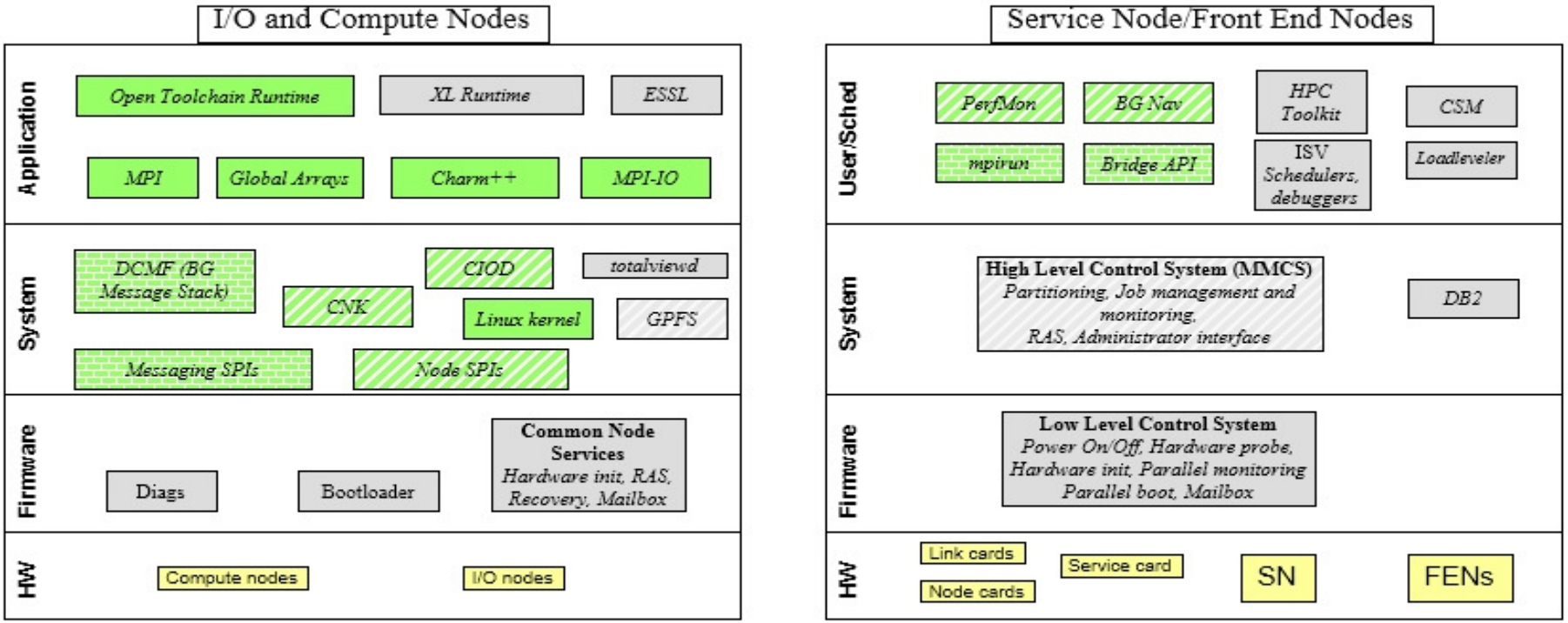
Your Supercomputer 25 Years Ago



Cray XC30 Software Stack



Current Blue Gene



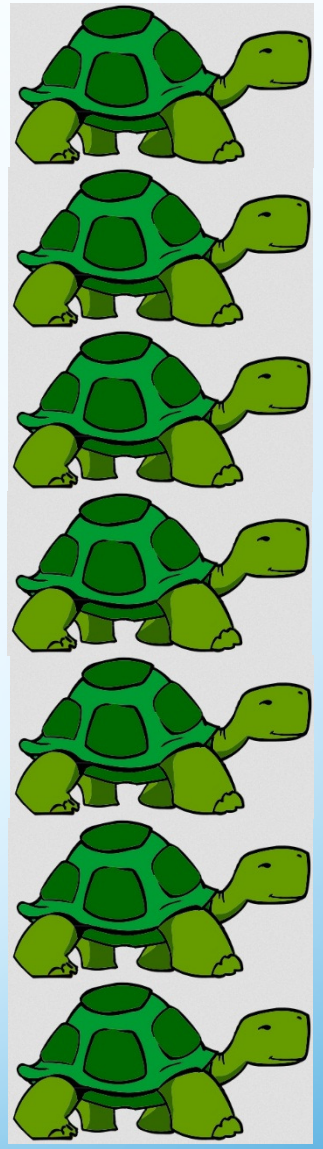
New open source reference implementation licensed under CPL.
New open source community under CPL license. Active IBM participation.
Existing open source communities under various licenses. BG code will be contributed and/or new sub-community started..
Closed. No source provided. Not buildable.
Closed. Buildable source available

Motivation for this Talk

- Building Web Pages
- Recent PhD talk

- Turtles all the way down

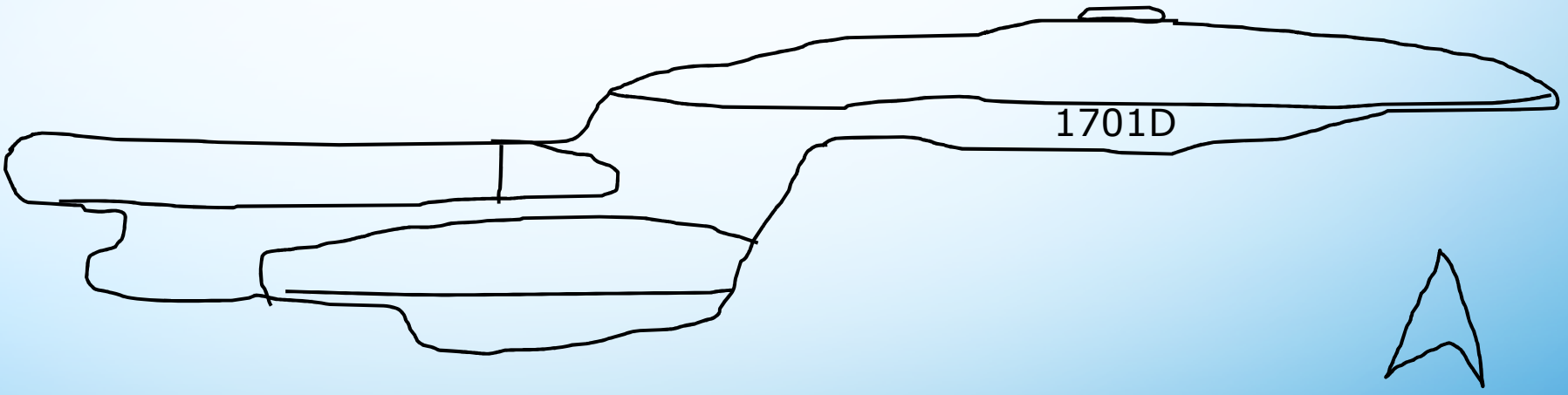
- HW and SW getting more complex



mOS

Make It So

- Supposition: analogous to dark silicon, let the “free” cycles do the work



Before Proceeding

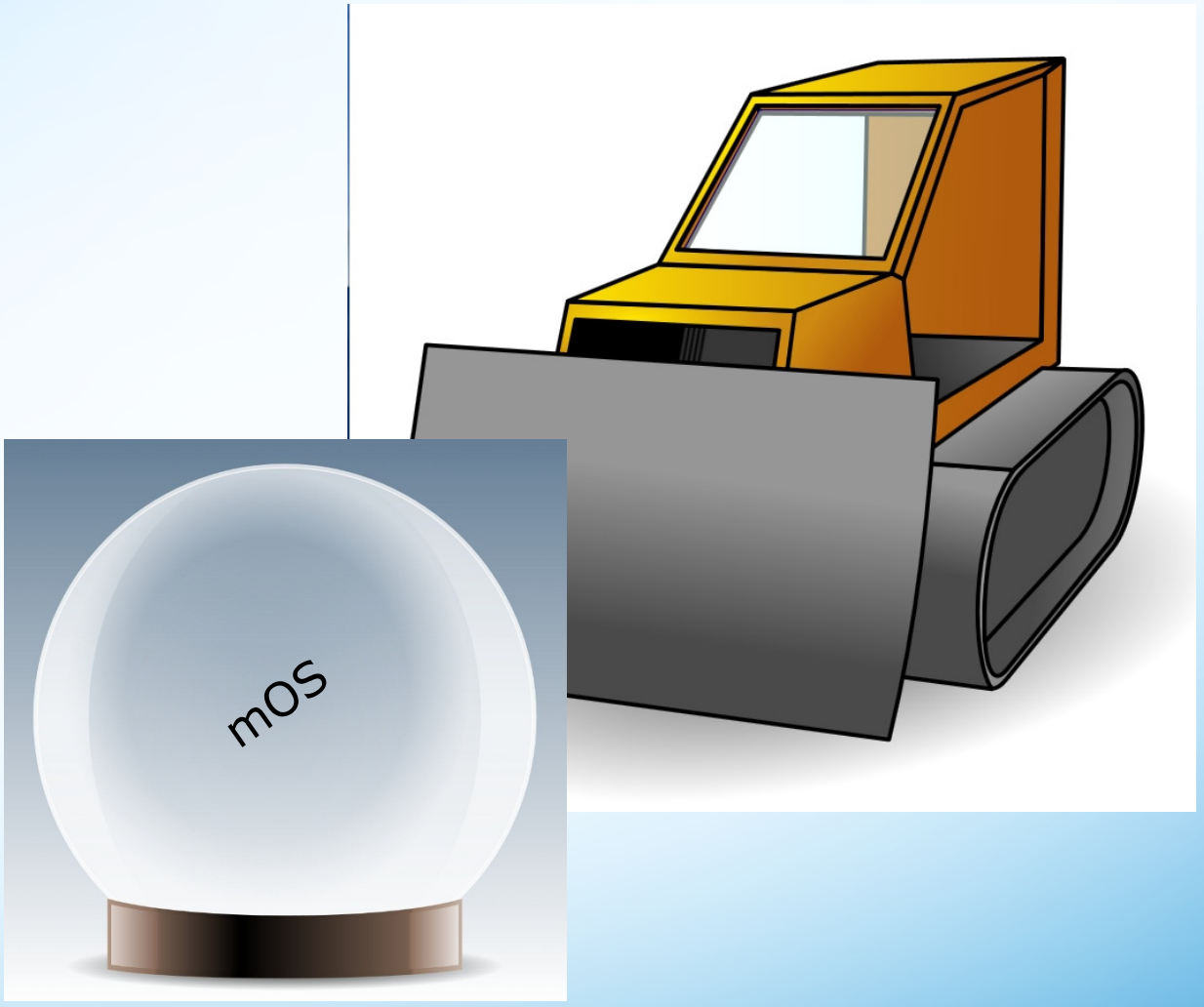
- Not for everyone
  - Consider cut through high performance
- Good for new applications
  - Biological for example
- Previous talks' theme:
  - The real challenge in moving software to extreme scale, and therefore the real solution, will be figuring out how to incorporate and support existing computation paradigms in an evolutionary model while **simultaneously** supporting new revolutionary paradigms.

Maybe not so far fetched, done this a bit already

- Compilers

- Demand paging
  - No
  - Yes

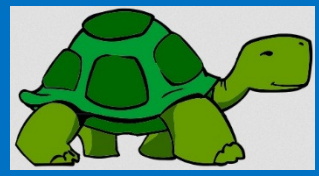
Peering into the Future with Hard Work



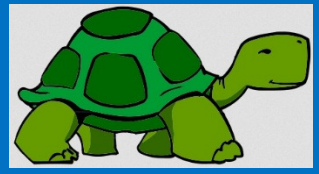
PEZ(Y)
Exascale is only a point on the continuum

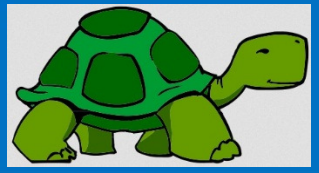
Zeta

Exa

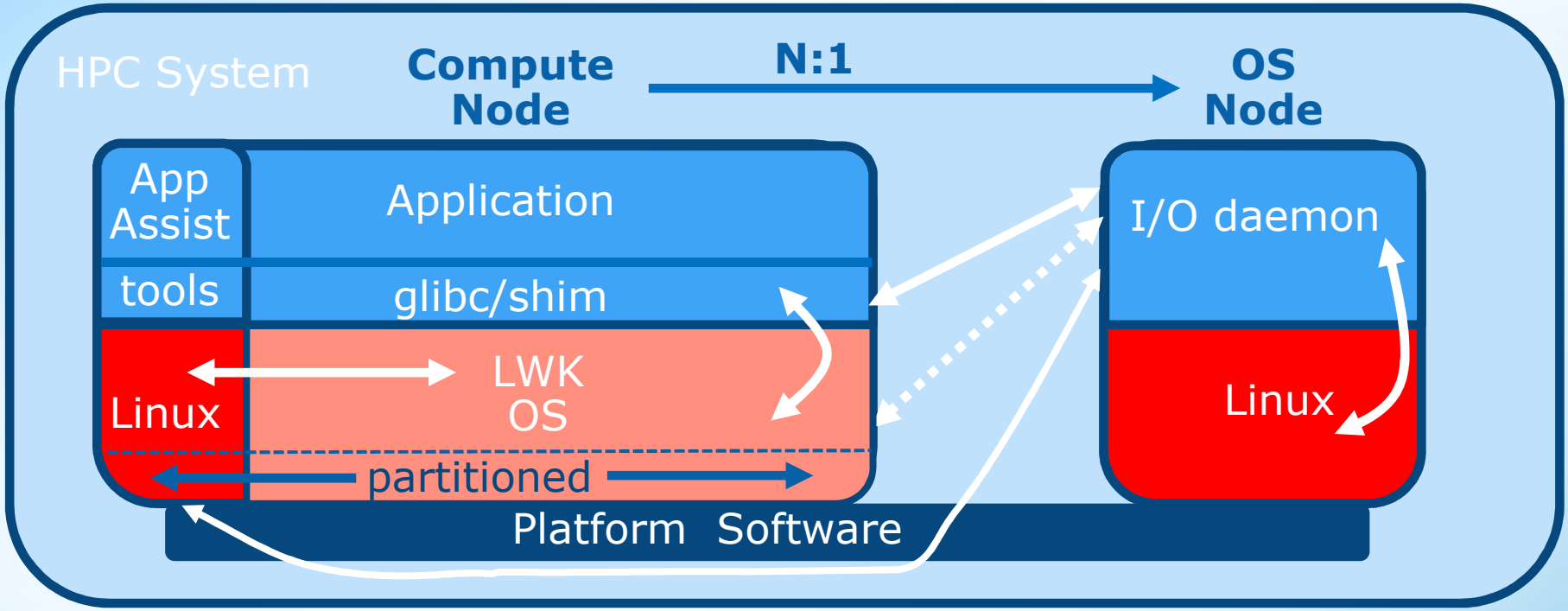


Peta





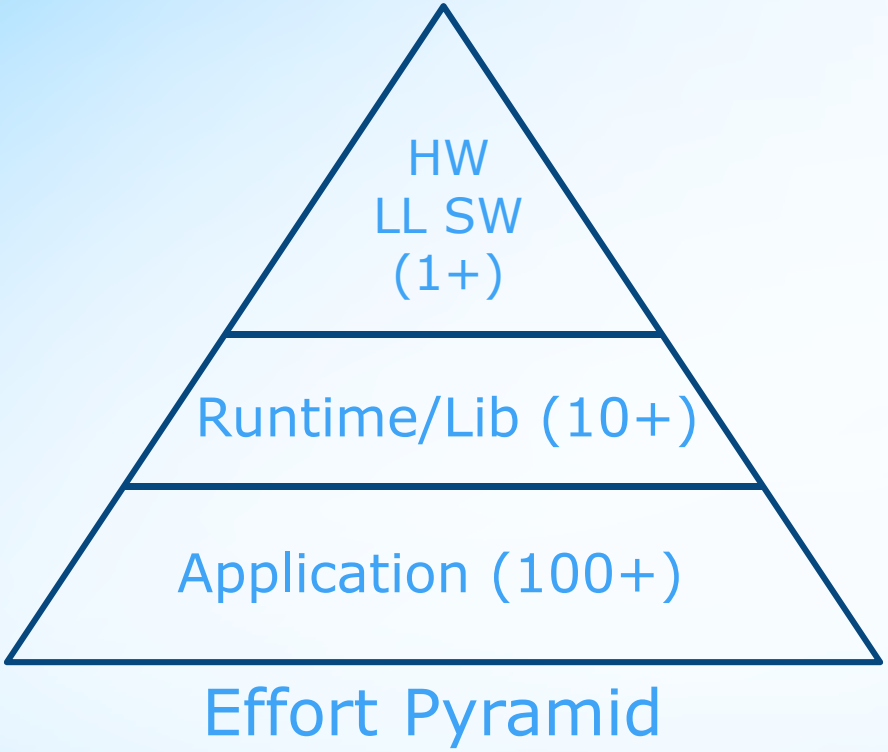
mOS: multiple Operating System



- Run multiple OSes on node simultaneously
- Linux API with LWK performance

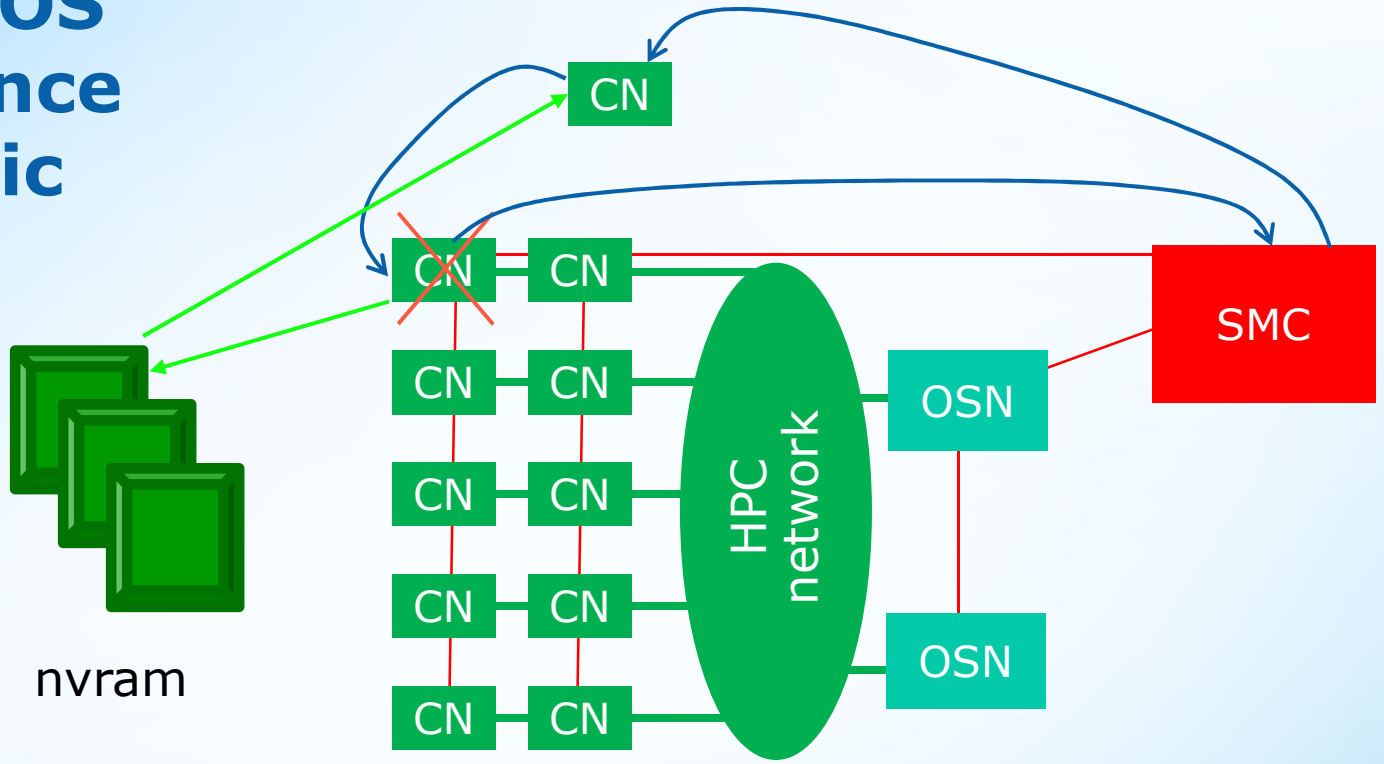
- Kernel is configured for the application
  - Provides compatibility and performance/scalability/reliability

Runtimes and Libraries



- Over commit
  - Adaptive, asynchronous, load balance, fault tolerance
- Building block approach

Global OS Resilience Dynamic

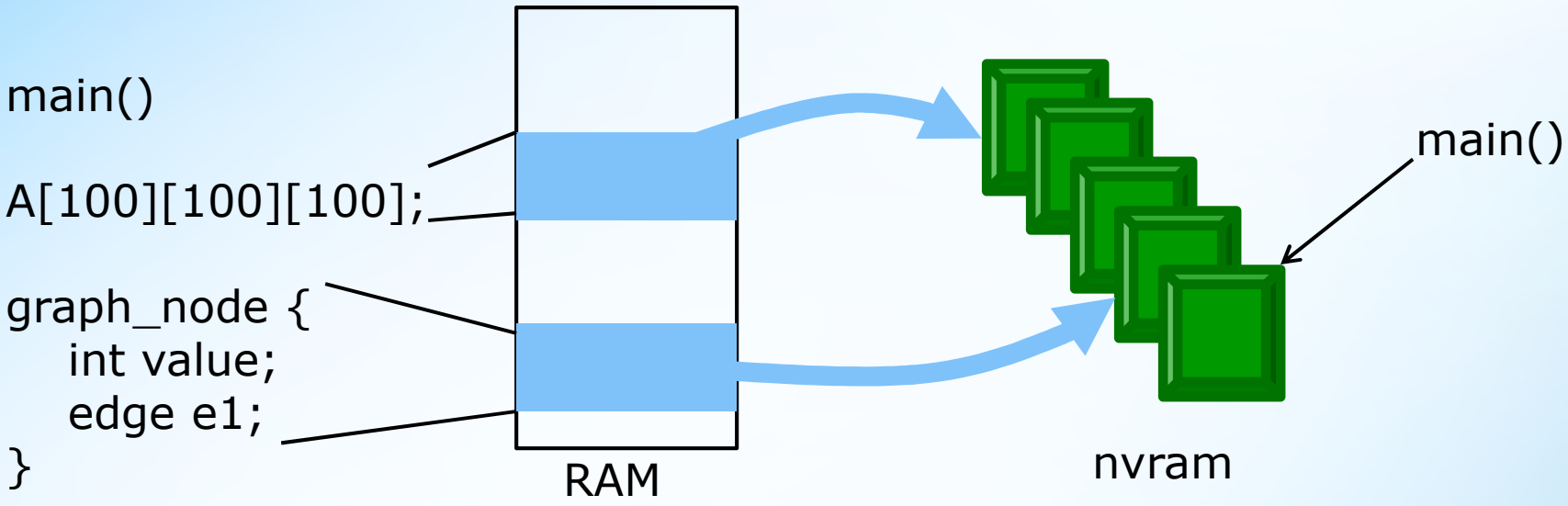


- Resolve as many faults as possible
  - Extra nodes swapped in automatically (proactively if possible)
    - NVRAM accessible independent of node
    - Automatic checkpoint of DRAM to NVRAM
  - For unresolvable provide fault information to application

Compilers / Fine-Grained Parallelism

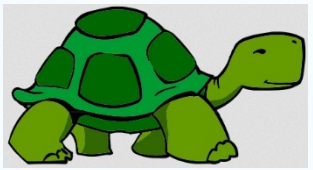
- There is more compute capability at lower power that can be unlocked if we figure out how to translate programing models into execution threads that can utilize it
- Current approaches to unlocking this potential have challenges
- Need a joint effort between compilers between what is possible and what would be needed from hardware to achieve

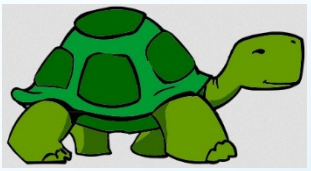
Data Management for Big Data



- Smooth and automatic representation between
  - Application data structure in memory
  - Representation and access to NVRAM
  - Storage to disk
- Moving compute to data
- Application makes system call
  - make_permanent(*data), make_durable(*data)

Make It So





- Leverage PEZ(Y) cycles to provide higher level and more productive abstractions to applications
- You're building tomorrow's turtles