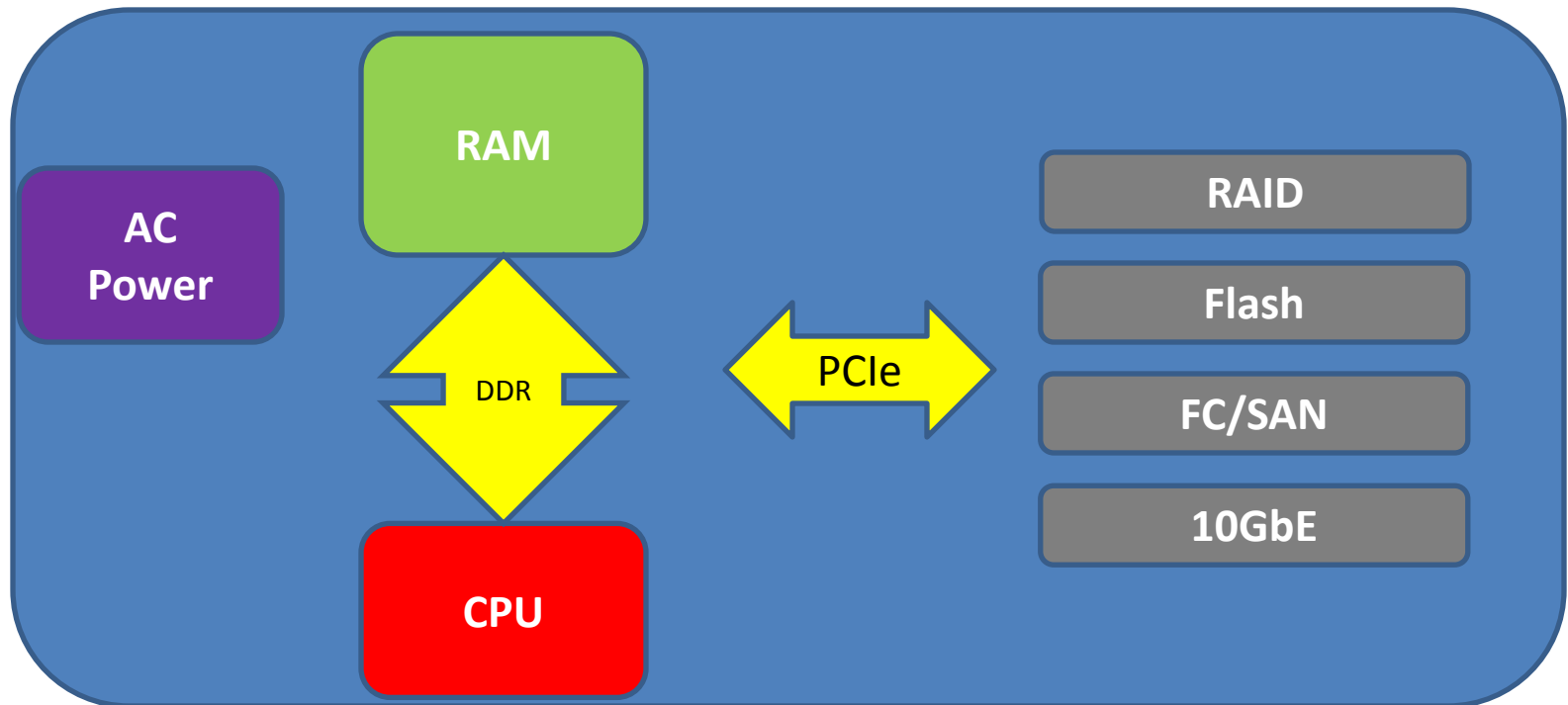


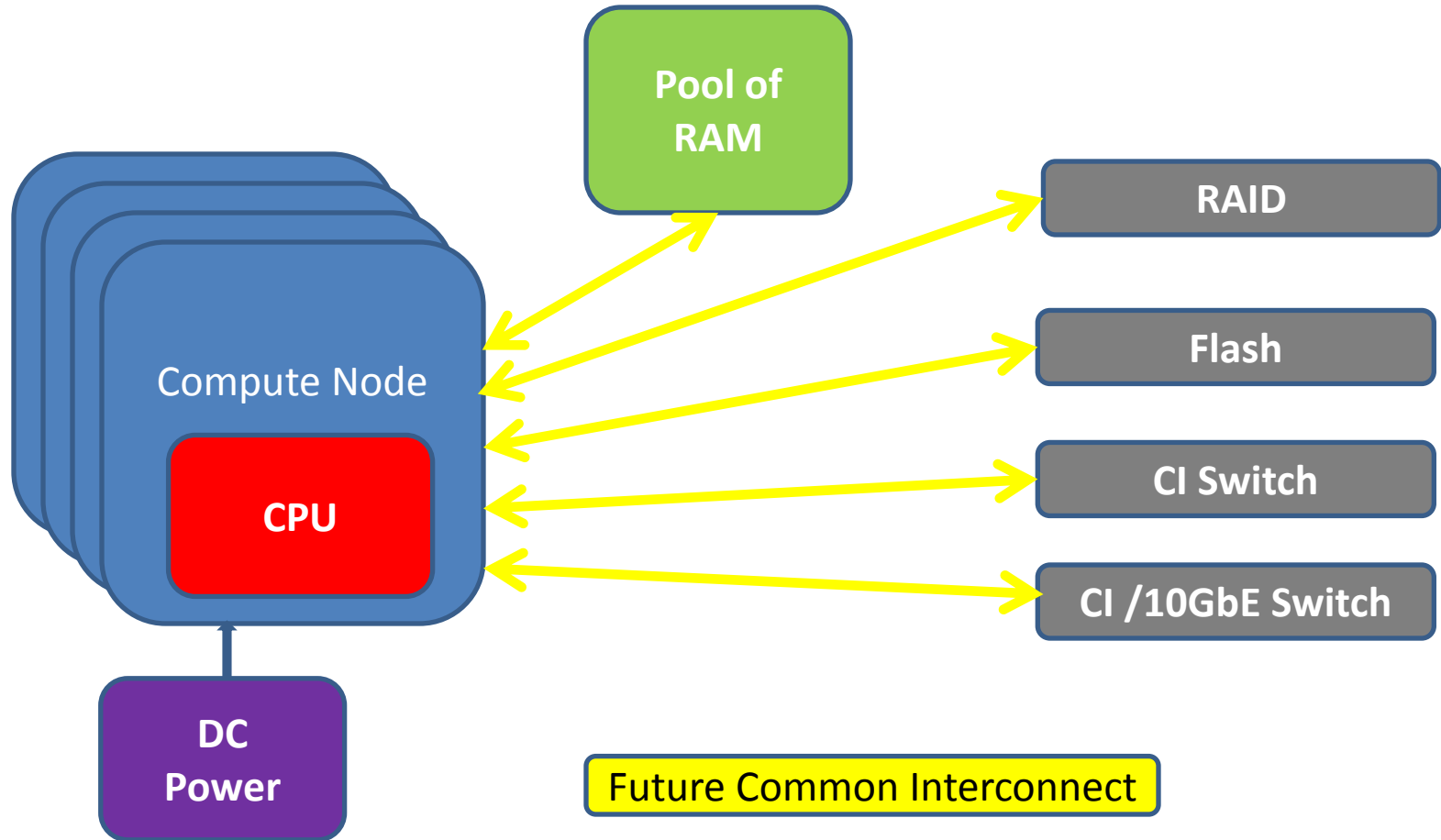
Appmem Devices

Disaggregating RAM

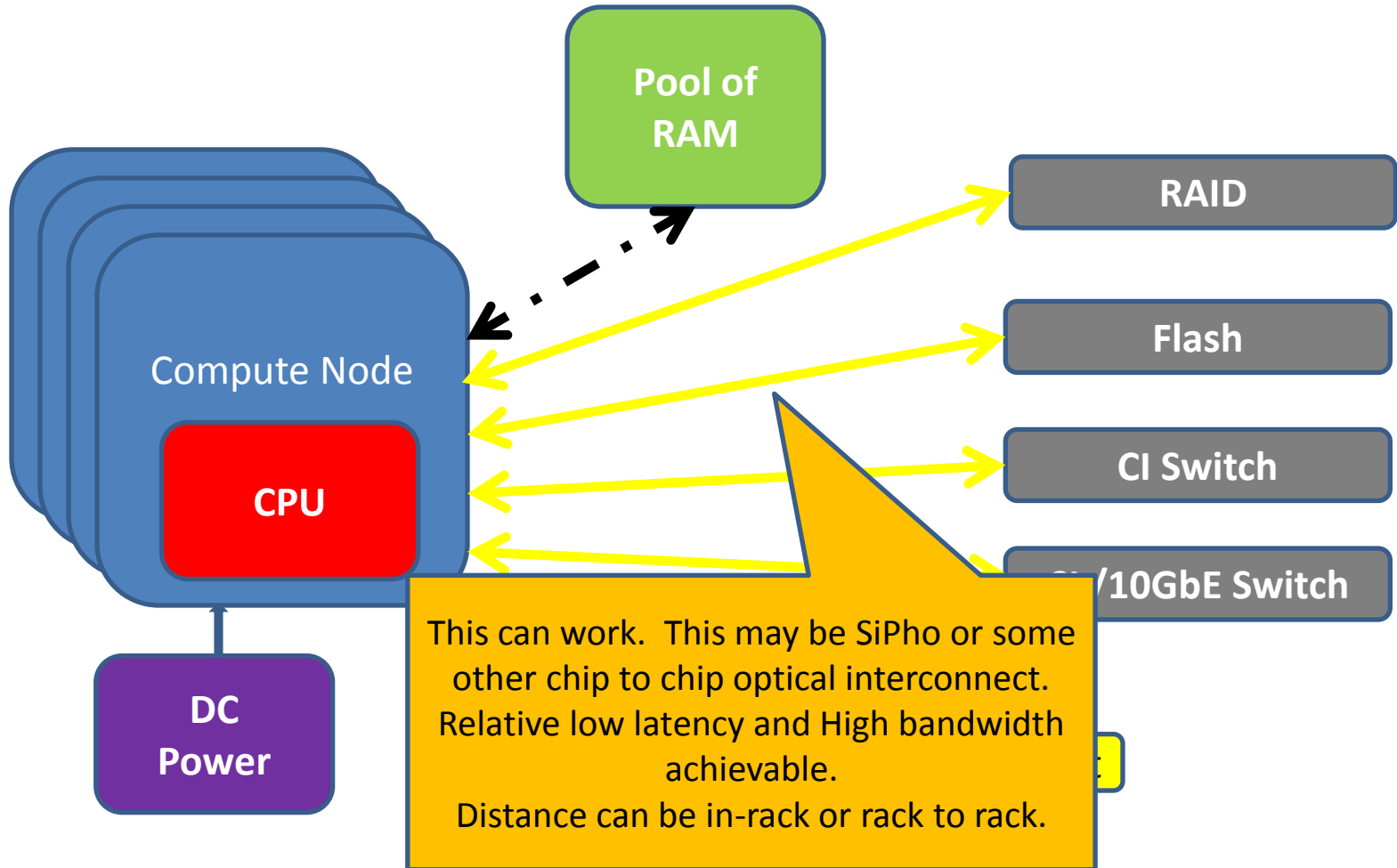
Traditional Server



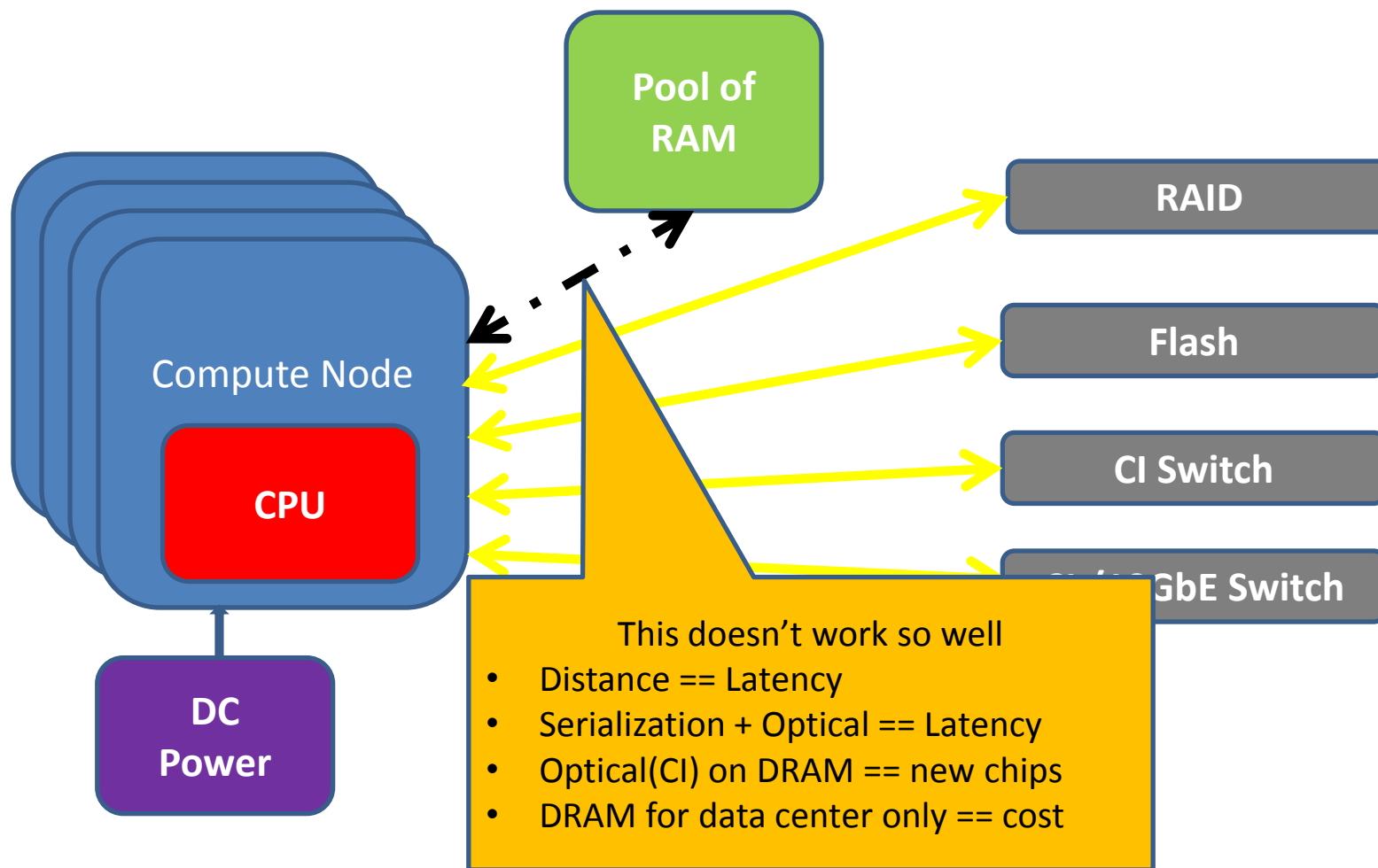
Future Common Interconnect (CI) Disaggregated Rack



Could work 😊



Maybe Not ☹️



Common interconnect

- DDRx level bandwidth achievable (40+ GB/s).
- Chip to chip optical might be technically challenging and costly but worth it in the data center.
- Lowest possible latency for the given protocol. (PCIe latencies or better + distance).
- Optical Avoids the electrical design and layout problems in todays boards.
- Fabrication, Marketing, and business hurdles to overcome. No fundamental technological or architectural roadblocks.

RAM Latency is a killer

- Generations of CPU design have dealt with RAM latency, yet we still need to keep it as low as possible.
 - multi-level caches
 - Out-of-order execution
 - Hyper-threading
- Longer stalls waiting on System RAM have a cascading effect on system performance.
- Architecturally to applications, all System RAM is treated the same (compilers and interpreters).
- Adding higher-latency RAM to system without application awareness will lead to inefficient programs.

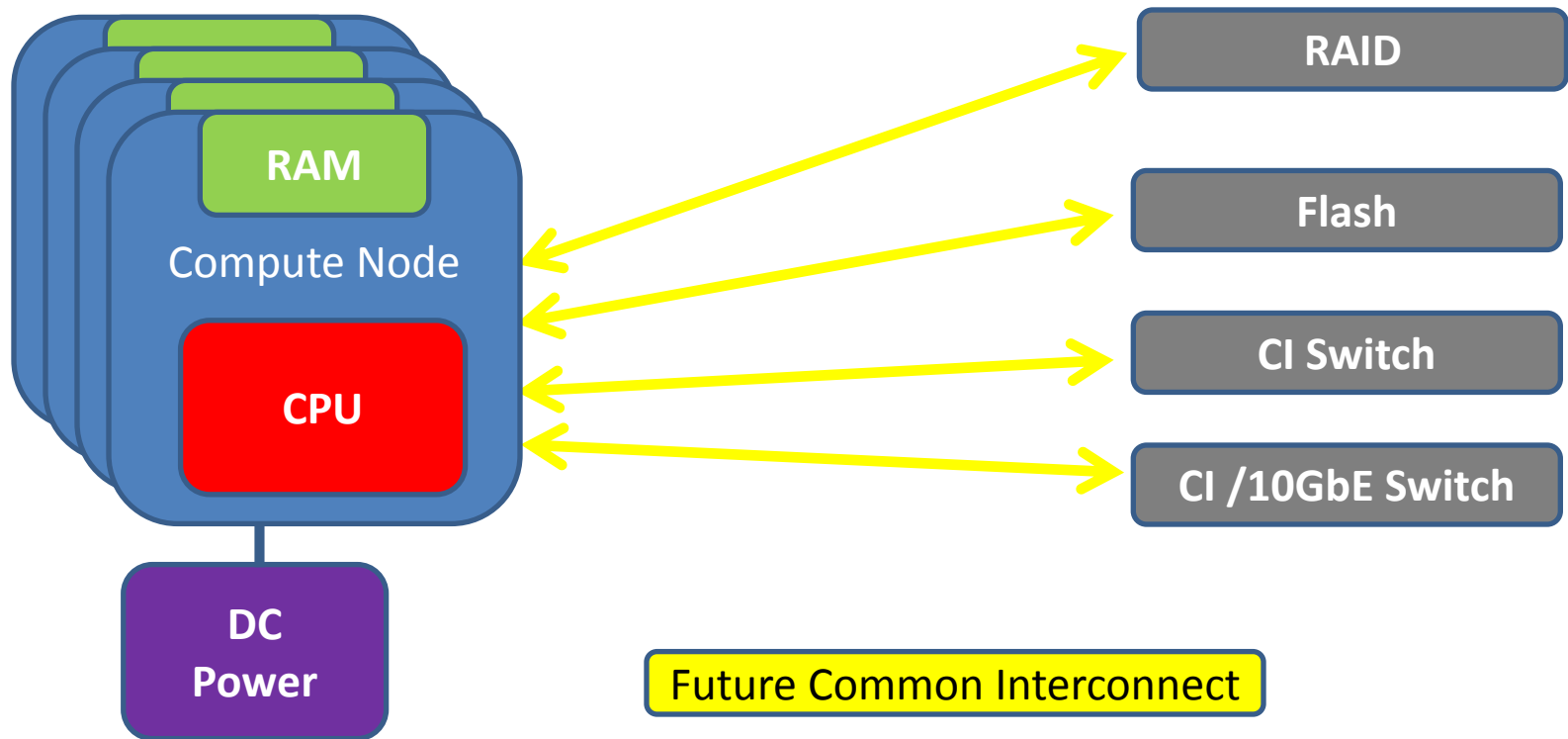
Disaggregating RAM adds latency

- Physical Optical latency = 5ns per meter.
 - Theoretical latency of DDR4 is around 15 ns.
 - DDR traces lengths are inches.
- Serdes of request/data stream add latency.
 - Logic at both ends needs to covert bit stream into addressable memory (10s of ns).
 - Currently DDR uses parallel bus and can transfer full address and multi-word data in single cycle.
 - Transport latency of PCIe is about 150-300ns (10-20x DDR)

Market issues

- DRAM chips are used everywhere
 - Adding CI to DRAM chips or adapters for data center adds costs that can't easily be amortized across consumer application in the near term.
 - Specialized DRAM chips for data center only add costs reducing benefit of disaggregation.
 - DRAM chips are commodity parts – increasing complexity will increase fabrication difficulties and cost.
 - Needs of serial memory in consumer space are different than those in data-center.

So we have to live with this?



Aggregated CPU and RAM

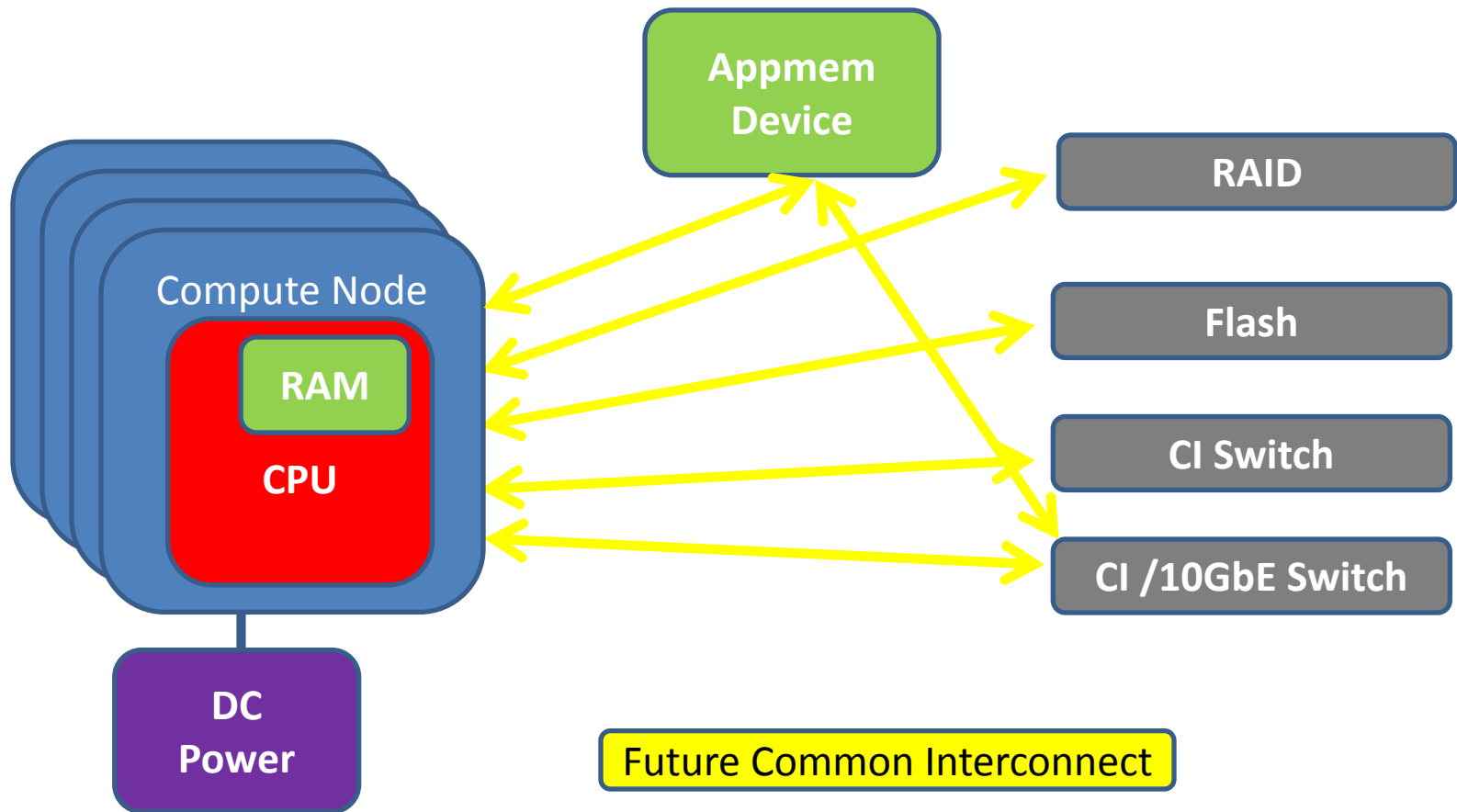
- Many applications are CPU/IO limited before they become RAM limited.
- 4-16 GB is plenty for many data center applications.
- Cost/layout reduction trend may be RAM on CPU die or package.
- Swap to flash is a solution for applications that occasionally exceed physical RAM.
- No architecture or application changes.

Disaggregated CPU and RAM

- For some applications RAM needs aren't known or change over time.
- Larger chunks of RAM may be used for short amounts of time while data is collated and processed.
- Using secondary storage for transitory data is both inefficient and costly.
- With CPU and RAM tied, you can't really have the "lego blocks" vision of a self-configuring data center.

Future Disaggregated Rack

Maybe it will look like this



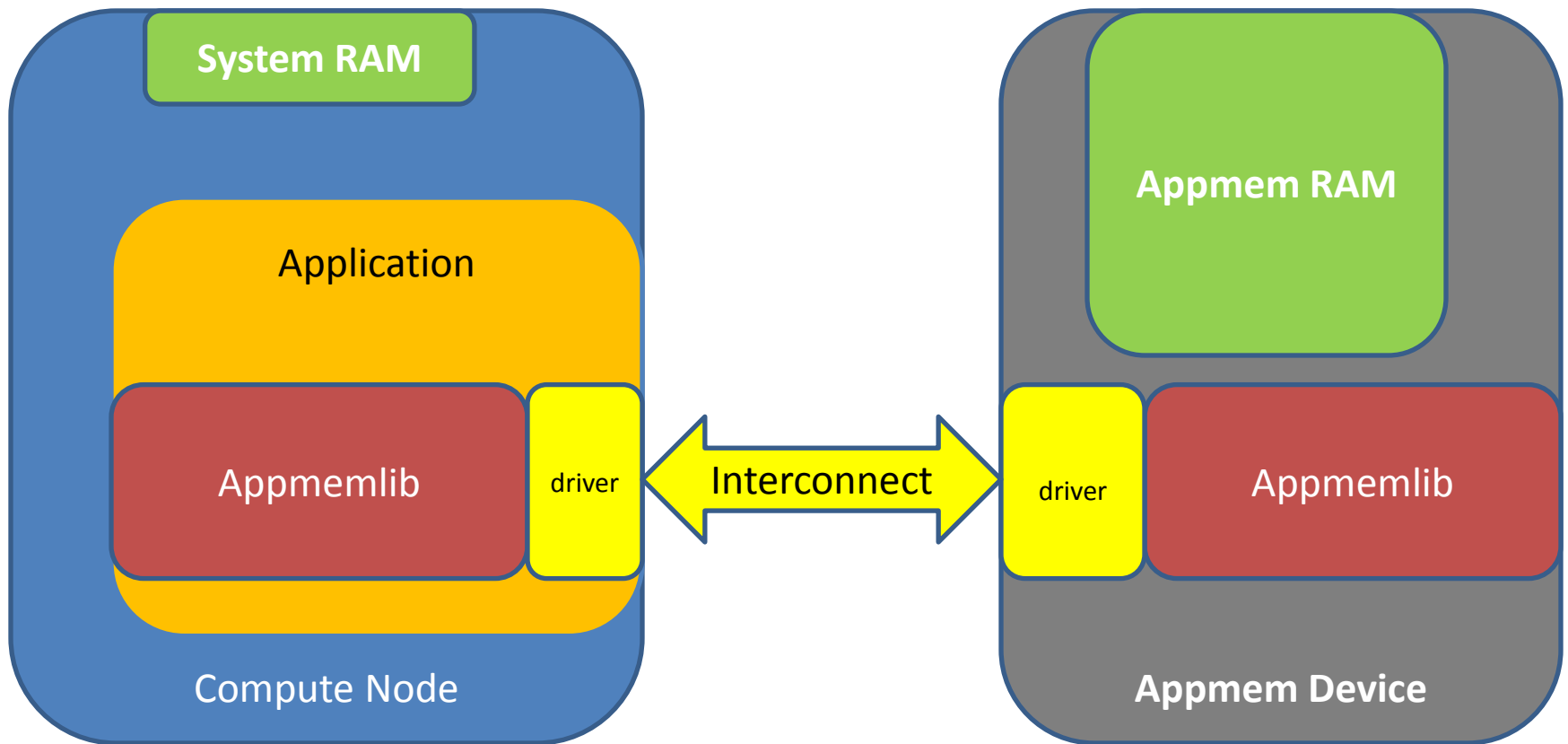
Appmem Device

- Device that has a large amount of RAM
 - May be a traditional server with lots of RAM
 - May be an embedded device
- Available to Applications
- Has “functions” – common data structures used by applications
 - Associative and Dynamic Arrays
 - Lists, Queues, Stacks
 - Trees
 - Application defined structures

Appmem Device Contentions

1. Memory may or may not be persistent
2. Memory may or may not be shared
3. Interconnect is irrelevant
4. Categorized by latency **at the application.**
5. Accessed using a common library.

Software Architecture



Appmemlib

- Uses the same library on the client and device.
- Application programmers can define their own functions and test them in System RAM, then integrate library on the device.
 - If device is linux based, just compile appmemlib on the device.
 - If device is embedded non-linux, vendor can provide SDK based on appmemlib.
- Interconnect specific driver either standardized or provided by the device vendor.

Appmem Device

Basic Advantages

- Direct access operations should have the same latency as the underlying interconnect.
 - PCIe Appmem device write/read of an object to memory should have same order of latency as mapped PCIe device memory read/write.
 - Some overhead penalty for the API.
- Complex Direct access operations are offloaded.
 - Example - In an Associative Array, hashing occurs on the device. Depending on speed of the device, overall latency may be greater.
 - The hashing is offloaded from the main CPU, freeing it to process other data.

Appmem Device

Compute Advantages

- Some Compute intensive operations can be completely offloaded.
 - Sort, Find, Copy, Delete, etc.
 - Any operation where the data elements can be described generically (offset, data type).
 - More complex and application specific functions can be created when Device needs to know structure of the data.

Appmem Device Value Add

- Persistence

- Device may be have flash/block or file based persistent storage.
- Persistence may be dynamically persistent or client directed (flush or close).

- Sharing

- Single Client may own the function or it maybe shared globally or through some authorization mechanism.
- Single Client mode sharing (one at a time) or live sharing .
- Locks/atomicity handled by the Device.

Appmem Device

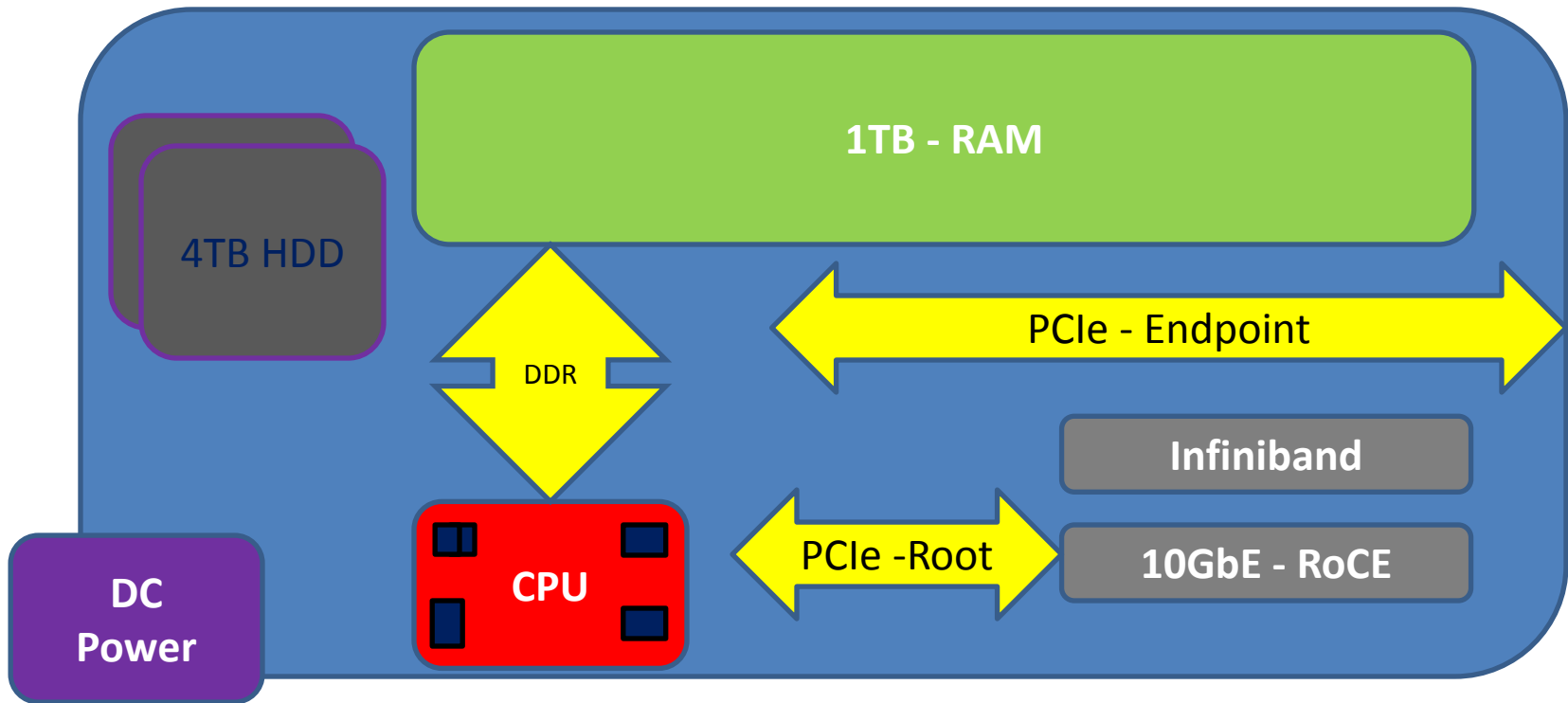
Value Add advantages

- Data structures can be swapped out of RAM to make room for other applications when some are idle.
- One client may perform collation operation from many sources, create a huge multi-GB Tree or List, and then hand off the handle to second client who can begin some processing operations on the data. No file read/write delay between the operations.
- Many other uses cases.

A High End Appmem Device

Current technology

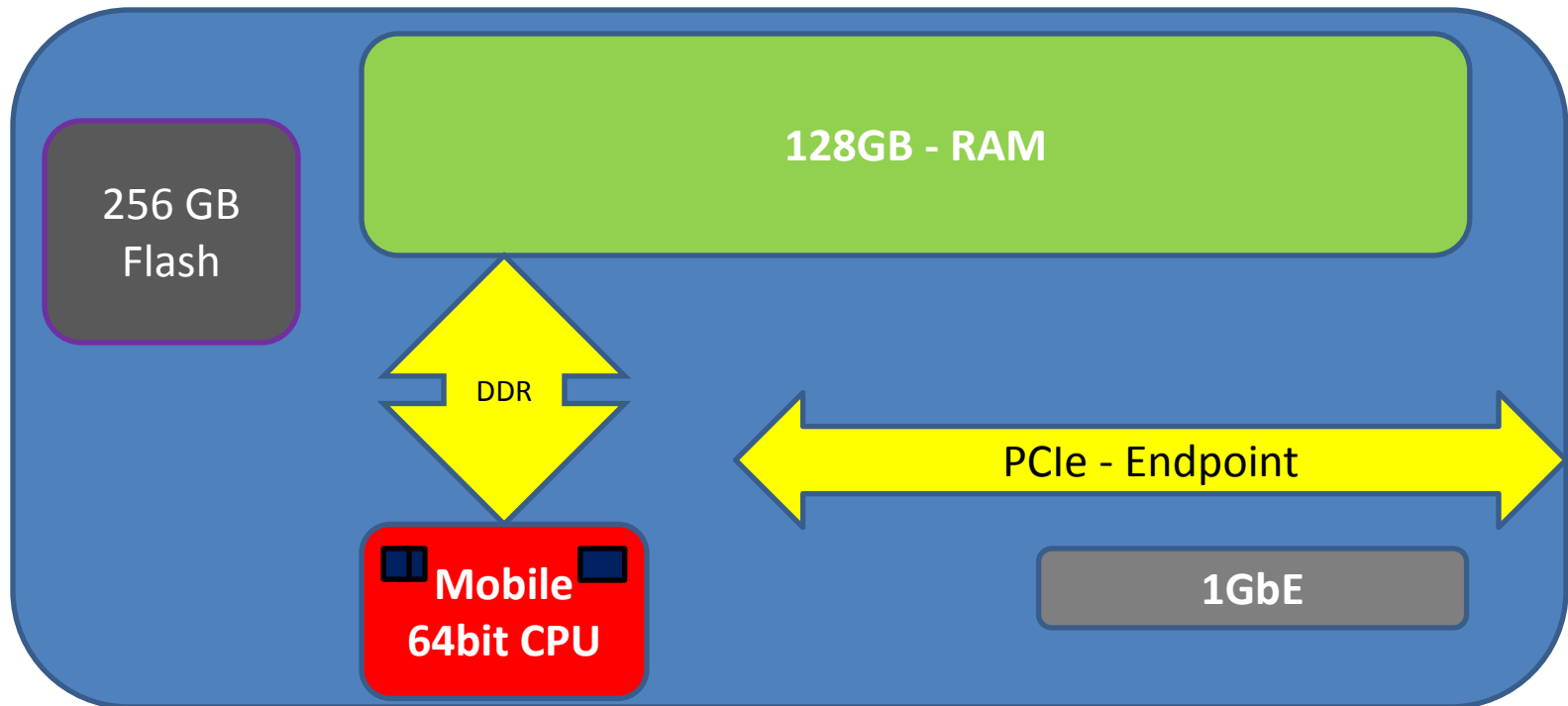
- Basically, a 4 core x86 server with a bunch of RAM, serving as a PCIe endpoint and also has Infiniband and 10GbE Adapters. May be in a rack or card slot.



A Embedded Appmem Device

Current technology

- A 64 bit Mobile-Class CPU, 128 GB RAM, with Flash. PCIe Endpoint with a remote access through 1GbE.



Appmem Device classification

- Devices are categorized by latency – at the client.
- Will use multi-level cache nomenclature
 - A0 has 0-10ns latency
 - A1 100ns
 - A2 1us
 - A3 10us. ($A5 = 10ns \wedge (5+1) = 1ms$)
- When client discovers devices, it will perform simple latency test to determine classification.
 - High end device might be A1 when connected via PCIe.
 - Same device could be A4 when connected via Ethernet and multiple switch hops.

Appmem project

- <https://github.com/scuzzydude/appmem>
- Very early concept code
 - Appmemtest utility. Tests both C and C++ interfaces to appmemlib.
 - Appmemlib – shared library with interfaces and basic data structures. User mode (emulation) target.
 - Appmemk – kernel driver with Memory Mapped and IOCTL interfaces and kernel emulated target.
 - Am_targ – very basic networked target
- Supports Flat Memory, Static Arrays, and Associative Arrays.

Next Steps

- Hardware
 - Demo real device as PCIe endpoint.
 - Server using PCIe NT as an endpoint.
 - Modify firmware of an existing PCIe card
 - FPGA based solution.
- Appmemlib
 - Real world use cases and realistic “big data” benchmark.
 - Add more data structures (Lists, Queues, Stacks, Trees, but also looking for more complex structures and algorithms to demonstrate offload.)
 - Terminology refinement (i.e. spec)
 - Refinement of library->driver layer
 - Wrappers for other languages (Java, JS, PHP).

Reference and Contact

- [Open Compute Summit – Disaggregated Rack](#)
- [Appmem Overview](#)
- [Github Appmem Project](#)
- [Appmem API and Implementations Notes](#)

Contact

- Brandon Awbrey – scuzzydude@hotmail.com