CS4450

Computer Networks:
Architecture and Protocols

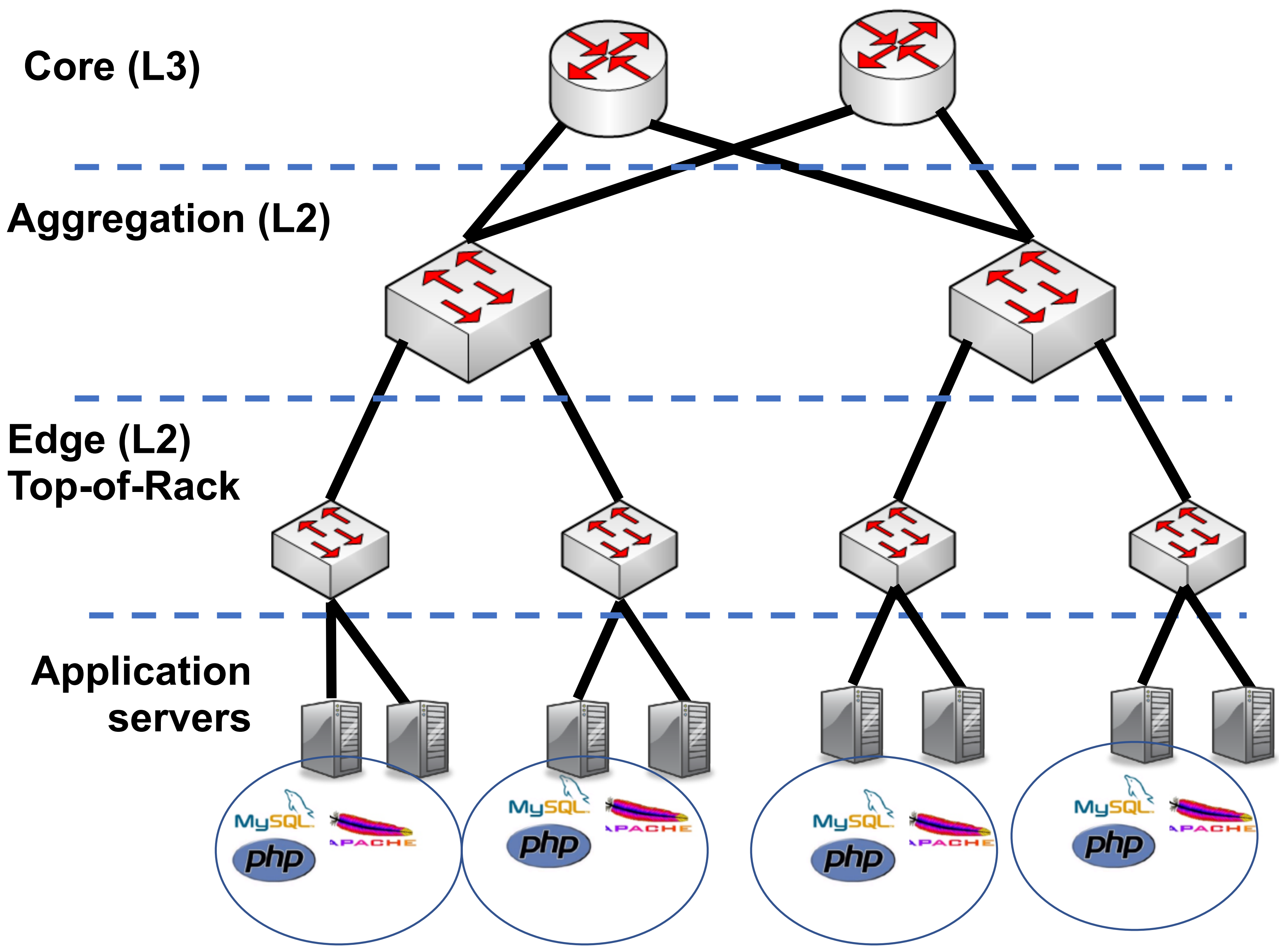
Lecture 26
Where's the puck going?

Rachit Agarwal

Announcements

- **Final: 05/12 @ 7PM, Hollister Hall B14**
- Make-up projects announced this morning
- Extra practice problems: by Sunday
- Prelim solutions posted
- Practice Finals posted (along with solutions)
- Problem solving sessions: Tuesday +
    - Tuesday: during the lecture hours; same location
- Lost sessions: thanks for using; makes me happy about my experiments
- **Please fill out the course evaluations**
    - Easy way to get 5%
    - **Please be constructive** (evaluations are for many eyes, not just me)

Recap: Canonical Datacenter Interconnect



Diameter, Bisection Width, Bisection Bandwidth, Oversubscription

Recap: Observations from the Interconnect

- **Link utilization low at edge and aggregate level**
- **Core most utilized**
  - Hot-spots exist (> 70% utilization)
  - < 25% links are hotspots
  - Loss occurs on less utilized links (< 70%)
    - Implicating momentary bursts
- **Time-of-Day variations exists**
  - Variation an order of magnitude larger at core

Recap: What is REALLY different when compared to the Internet?

What is REALLY different from the Internet

- **Single entity owns everything, from the OS to the network hardware**
  - Discussion: how could we exploit this property?
- **Link Layer and Network Layer**
  - Increasingly less separation between the two layers
  - Do we still **need** BGP?
  - Could we still **use** BGP?
- **Transport Layer?**
  - A lot of failure modes of TCP go away (OS owned by Google)
  - Is TCP still a good solution?

What is REALLY different from the Internet

- **Fixed (structured) topology, complete control and knowledge**
    - Discussion: how could we exploit this property?
- **Link Layer and Network Layer**
    - More efficient algorithms for route computation
    - Could "bake in" routing **results** into switch routing tables
    - Software-defined networks, centralized control
    - **Other benefits:**
        - Better control over "load balancing"
        - Avoid convergence issues (but new issues come up)
- **Transport Layer?**
    - We never made any assumptions about topology in L4 design
    - Is TCP still a good idea?

What is REALLY different from the Internet

- **Small-scale, within a single geographic location**
  - **The entire datacenter is may be 1M machines, in a single location**
  - Discussion: how could we exploit this property?
- **Link Layer and Network Layer?**
  - Another motivating factor for centralized control
  - Routes can be computed and “installed” quickly
- **Transport layer?**
  - Next slide …

What is REALLY different from the Internet

- **Tiny round trip times**
  - **Less than 5 microseconds (for a single packet)**
  - Discussion: how could we exploit this property?
- **Link Layer and Network Layer?**
  - Millisecond-level convergence times no longer "sufficient"
  - Even more motivation for software-defined, centralized control
- **Transport layer?**
  - Most flows small; can be completed within a couple of RTT
  - Even 3-way hand-shake could have high overheads
  - TCP is not going to work well!

TCP in datacenter context

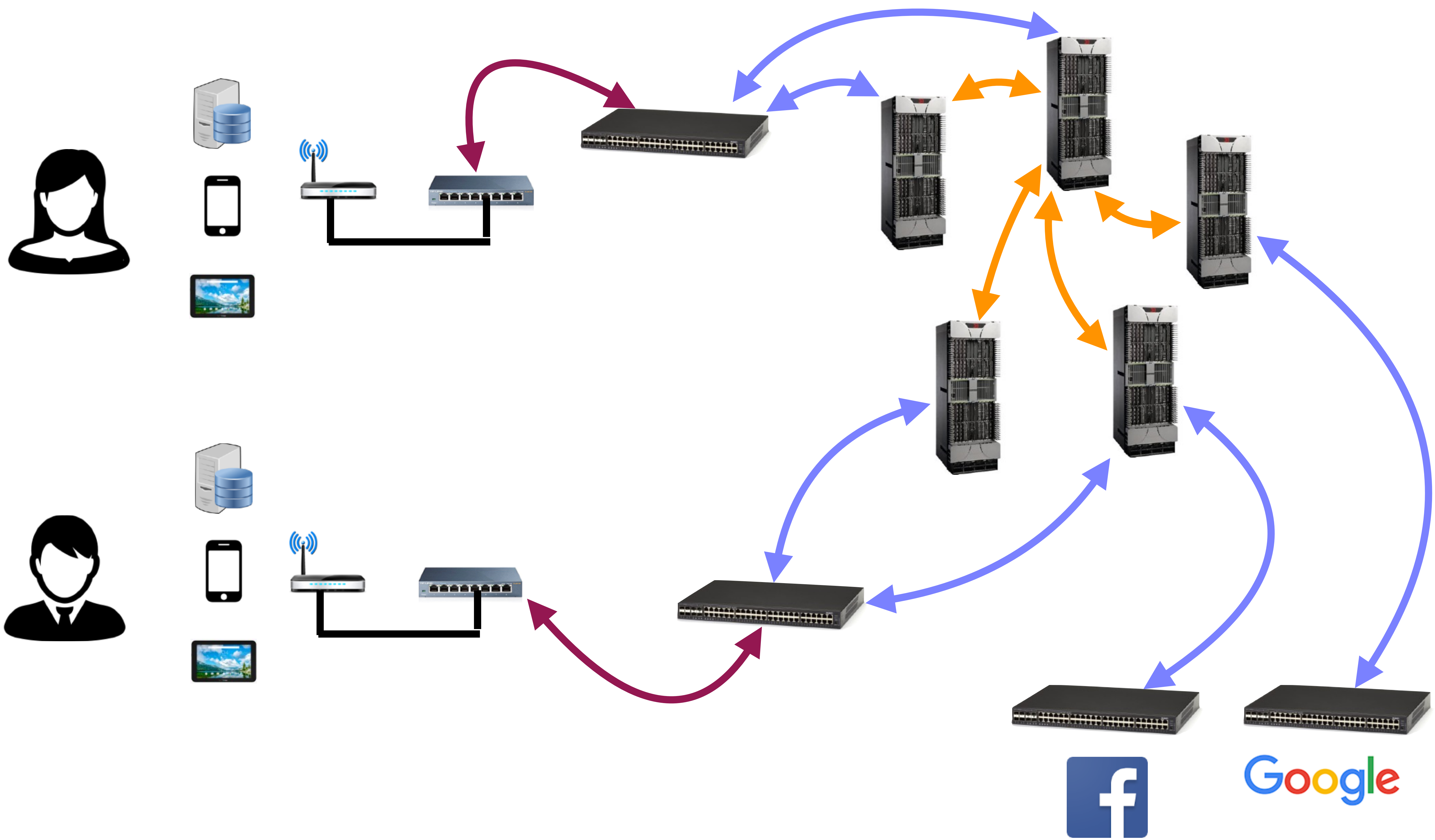
- **TCP is too inefficient**
  - Three-way handshake takes **too long**
  - Does not work well with **short flows**
  - Not designed for **low latency**
  - Has no notion of **deadlines**
  - **Queue build-up due to long flows; short flows suffer**

Datacenter Transport Design:
One of the most active research areas

Taking 25 steps back!

What is a computer network?

A set of network elements connected together, that implement a set of protocols for the purpose of sharing resources at the end hosts



Sharing networks

- **Two approaches**
  - Reservation (circuit switching)
  - Statistical multiplexing (packet switching)

- **Motivation for WHY modern networks use “packets”**

- **How to implement this?**

The end-to-end story

- Application opens a **socket** that allows it to connect to the **network stack**
- Maps **name** of the web site to its **address** using **DNS**
- The network stack at the source embeds the address and **port** for both the source and the destination in **packet header**
- Each **router** constructs a **routing table** using a distributed algorithm
- Each router uses destination address in the packet header to look up the **outgoing link** in the routing table
    - And when the link is free, forwards the packet
- When a packet arrives the destination:
    - The network stack at the destination uses the port to forward the packet to the right application

Realizing end-to-end design: Three Principles

- How to break system into modules
  - **Layering**
- Where are modules implemented
  - **End-to-End Principle**
- Where is state stored?
  - **Fate-Sharing**

Five Layers (Top - Down)

- **Application:** Providing network support for apps
- **Transport (L4):** (Reliable) end-to-end delivery
- **Network (L3):** Global best-effort delivery
- **Datalink (L2):** Local best-effort delivery
- **Physical:** Bits on wire

Link Layer (L2)

- **Broadcast medium:** Ethernet and CSMA/CD
- **We studied that Broadcast Ethernet does not scale to large networks**
  - Motivation for switched Ethernet
- **Broadcast storm:** if using broadcast on switched Ethernet
  - Motivation for Spanning Tree Protocol
- **Limitations of Spanning Tree Protocol:**
  - Low bandwidth utilization, high latency, unnecessary processing
  - Does not scale to the entire Internet
  - Motivation for **routing protocols** in the Internet

Network Layer (L3)

- **Internet Protocol:**
  - Addressing, packet header as an interface, routing
- **Routing tables:**
  - Correctness and validity: Dead ends, loops
  - A collection of spanning trees, one per destination
- **Constructing valid routing tables (within an ISP)**
  - Link-state and distance-vector protocols
  - Focused a lot on learning via examples
  - Can still have loops: failures remain to be a pain
- **How to use routing tables**
  - **Packet header as an interface**
  - Learnt why packet headers look like the way they do

Network Layer (L3), Cont.

- **Internet Protocol:**
  - Addressing, packet header as an interface, routing
- **Addressing:**
  - Link layer uses “flat” addresses
  - **Does not scale to Internet**: motivation for IP addresses
  - **Scalability challenges**: Routing table sizes, #updates
  - Solution: **Hierarchical addressing**
- **Forwarding**
  - **Switch architecture**
  - Longest Prefix matching for forwarding at line rate
  - Scheduling using priorities

Network Layer (L3), Cont.

- **Internet Protocol:**
  - Addressing, packet header as an interface, routing
- **Limitations of link-state and distance-vector routing:**
  - Require visibility of the entire Internet
  - **ISPs do not like that:** motivation for Inter-domain routing
  - **Border Gateway Protocol**
    - A simple modification of distance-vector protocol
- **Routing with policies**
  - **Customer-provider-peer relationships**
  - Gao-Rexford policies
- **Completes the network layer: provides connectivity**

Details for complete picture

- **DHCP: Dynamic Host Configuration Protocol**
  - For each host to figure out its IP address, local DNS, first-hop router
- **ARP: Address Resolution Protocol**
  - For finding other servers on the same local area network (L2)
  - Mapping from IP addresses to names (MAC addresses)
- **Domain Name System**
  - **Mapping Human readable destination names to IP addresses**
  - Hierarchical structure

Transport Layer

- **Goals of reliable transport**
  - **Correctness condition**
  - Why do we need ACKs, timers, window-based design

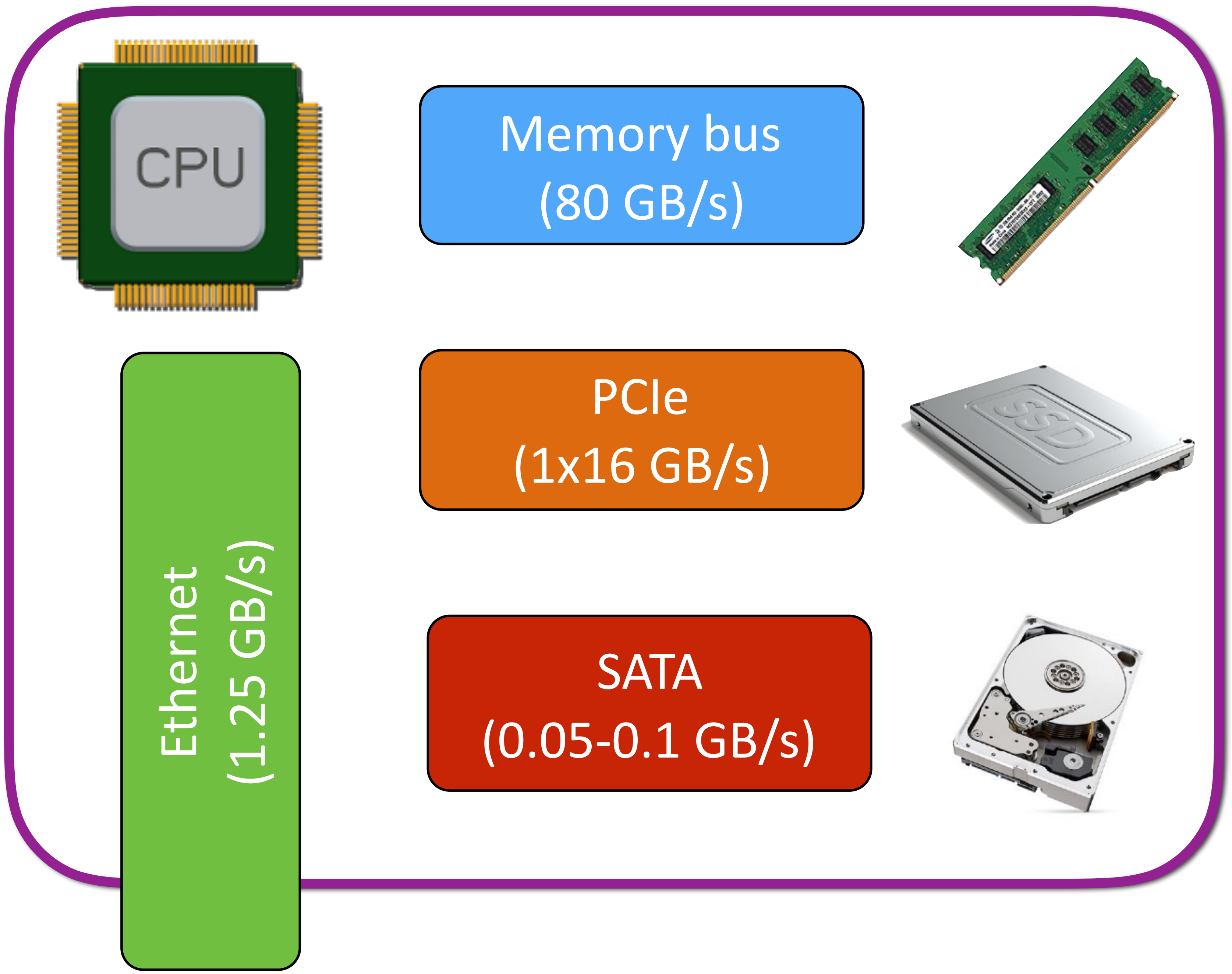
- **One realization of reliable transport: TCP**
  - Mostly implementation details following the above design
  - For **max-min fairness**, flow performance and utilization
  - **Flow control**
    - Ensuring the sender does not overwhelm the **receiver**
    - Via receiver advertised window size
  - **Congestion control**
    - Ensuring the sender does not overwhelm the **network**
    - Slow start, Additive-increase Multiplicative-decrease, timeouts

Taking 1 step forward!

Skate where the puck's going, not where it's been!

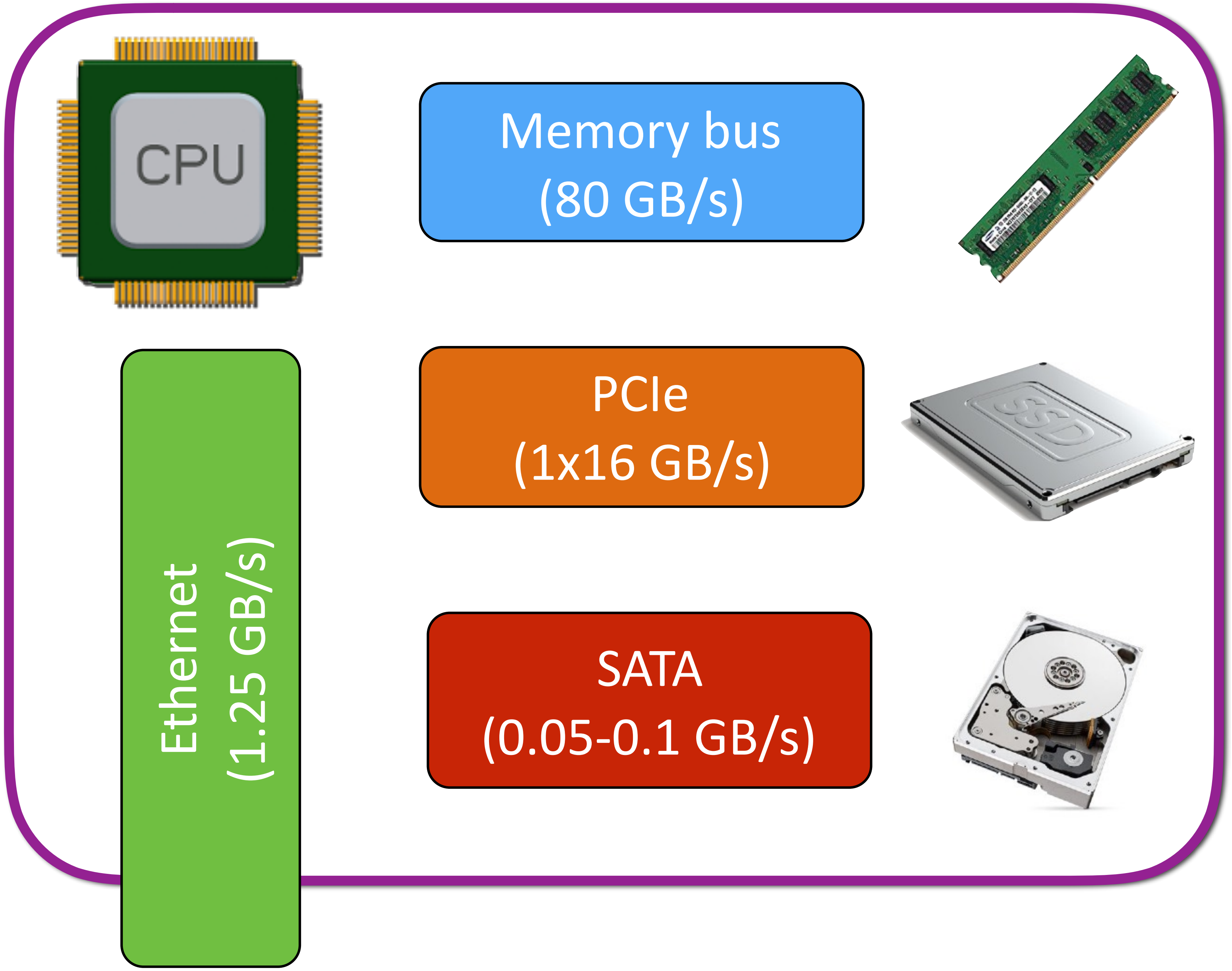
- Walter Gretzky

Where is the puck right now?

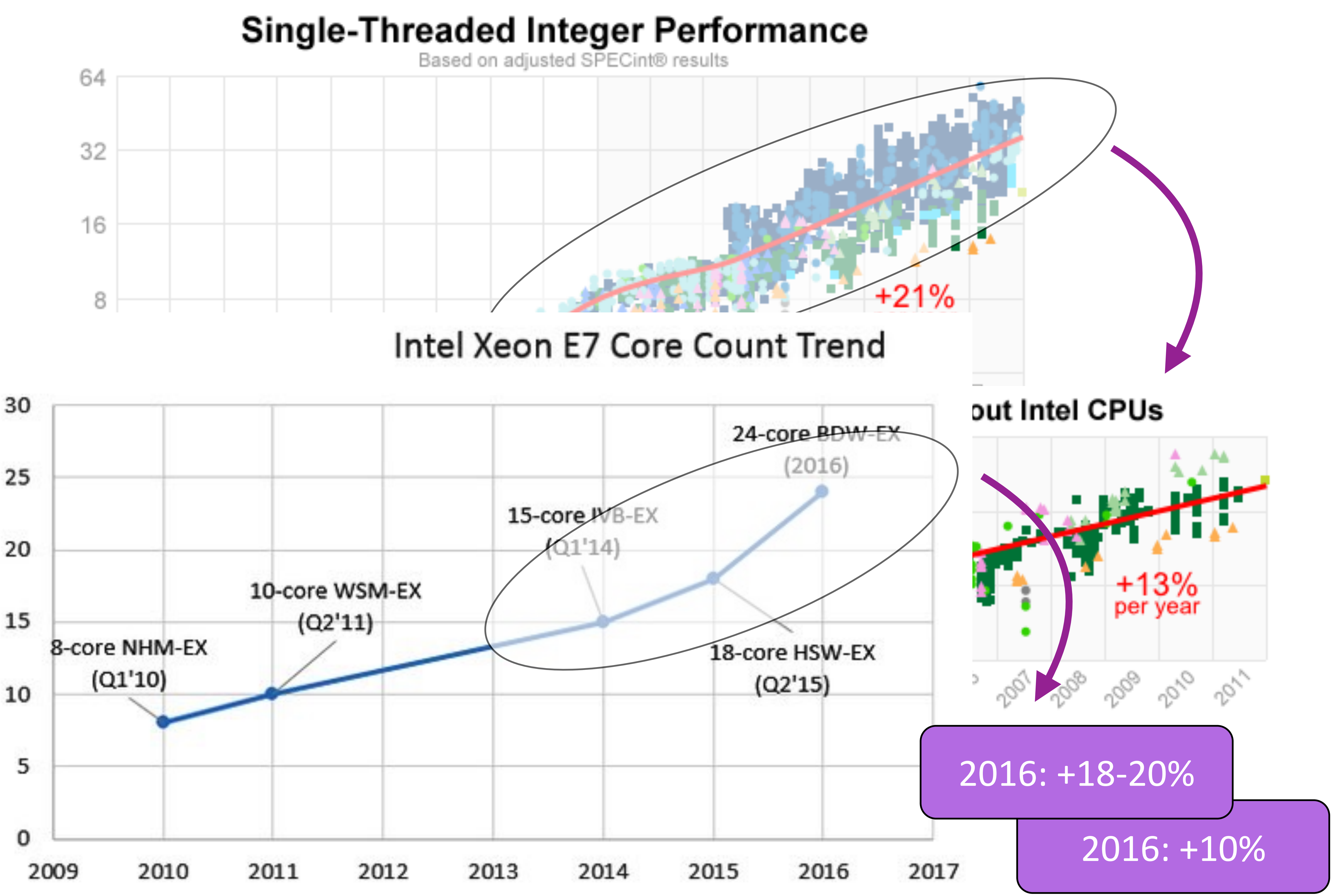


Size (TB)	Random Access (us)	Seq. Access (GB/s)
0.1	0.1	80
1	25	1x
10	4000	0.1x

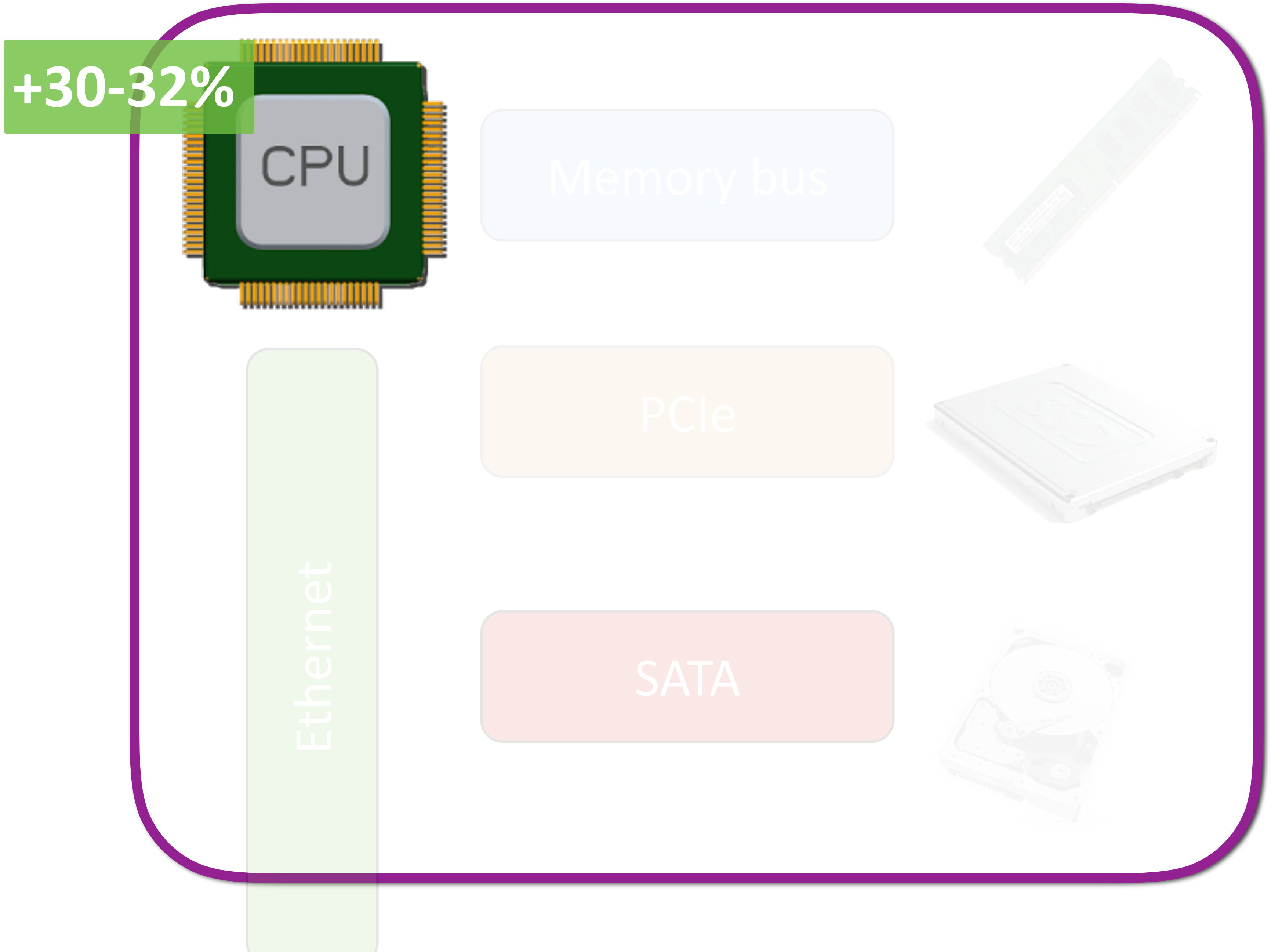
Where is the puck going?



Where is the puck going? (CPU performance)

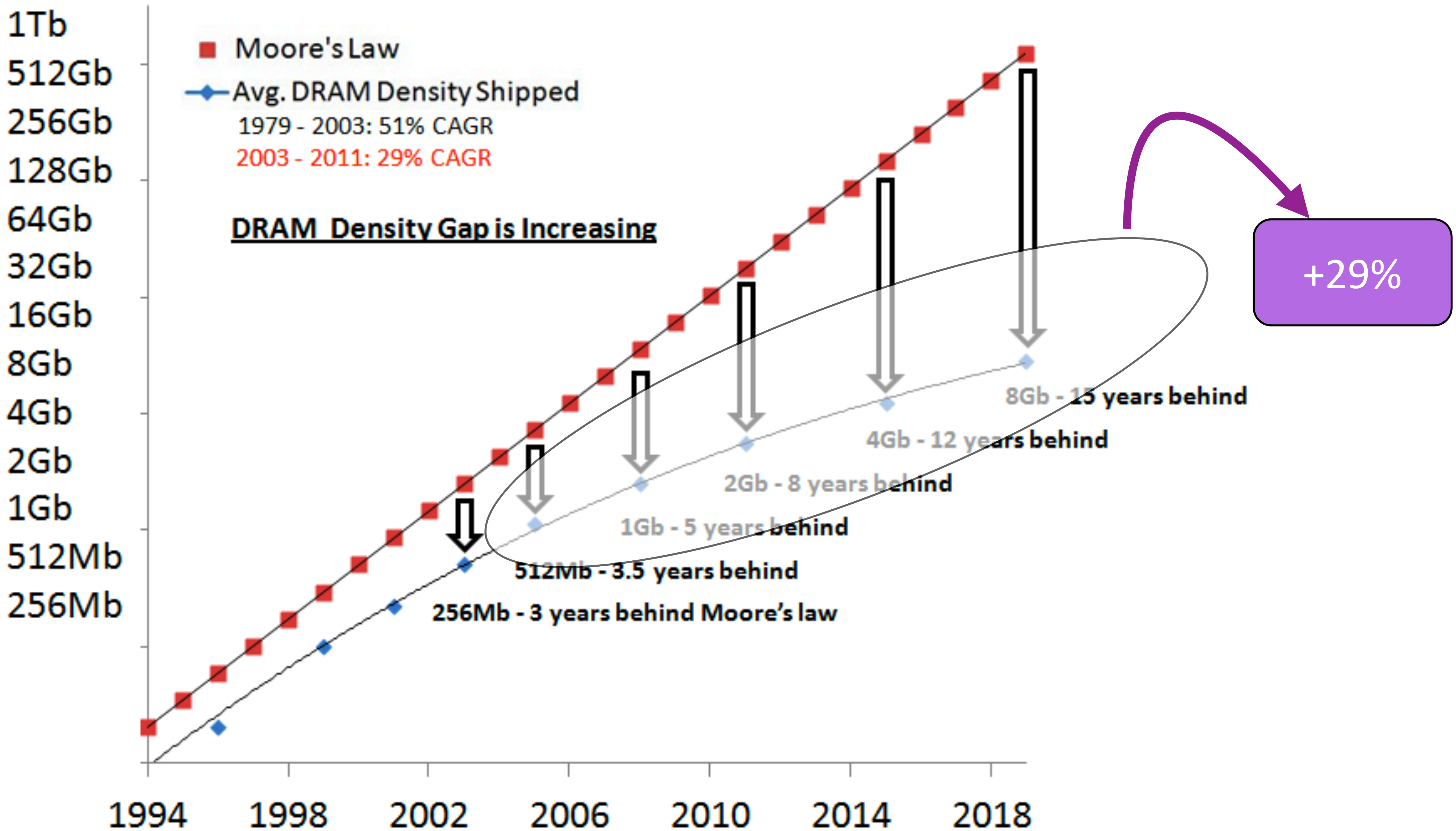


Where is the puck going?

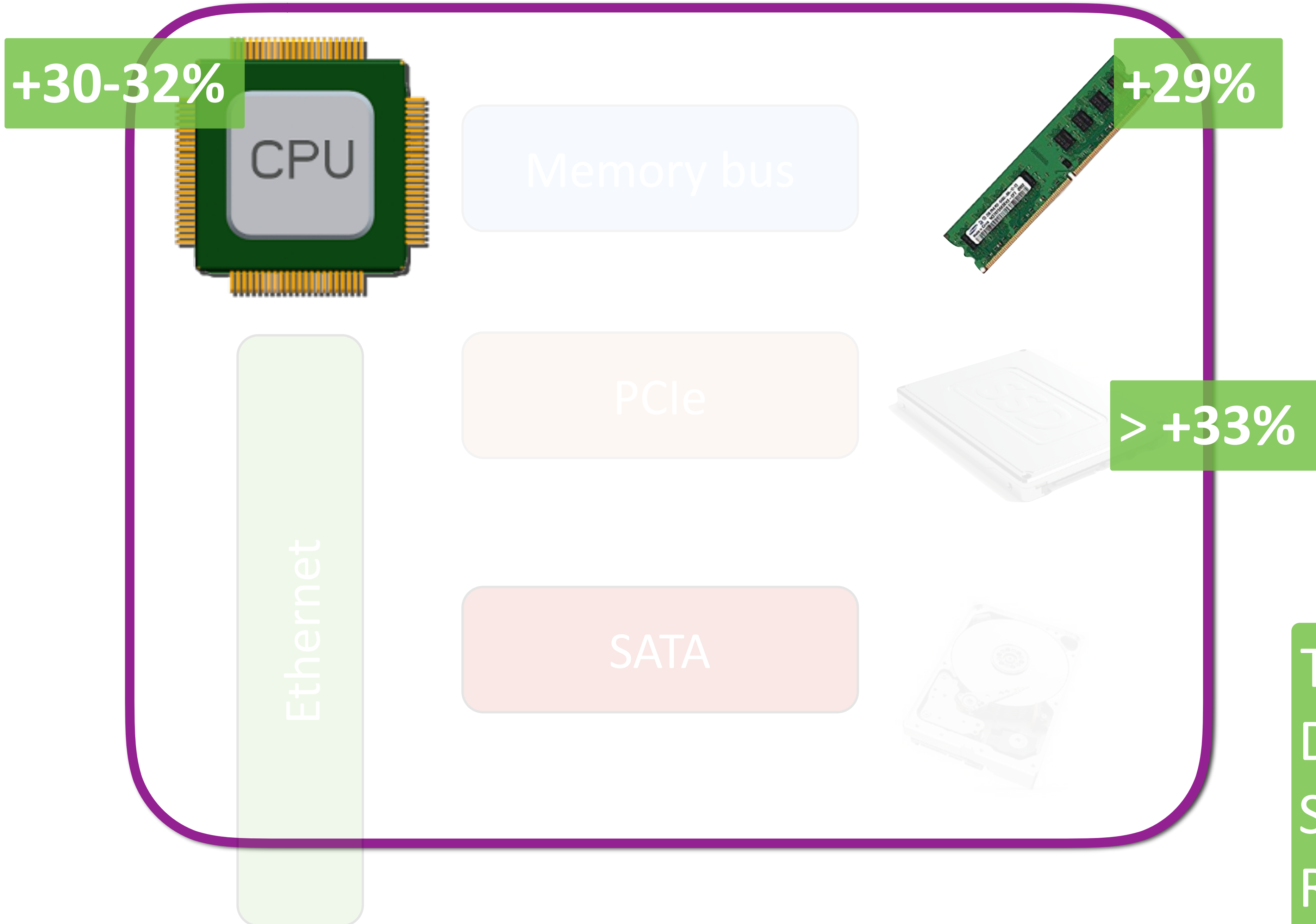


- **#Cores:** +18-20%
- **Per core: +10%**

Where is the puck going? (DRAM capacity)



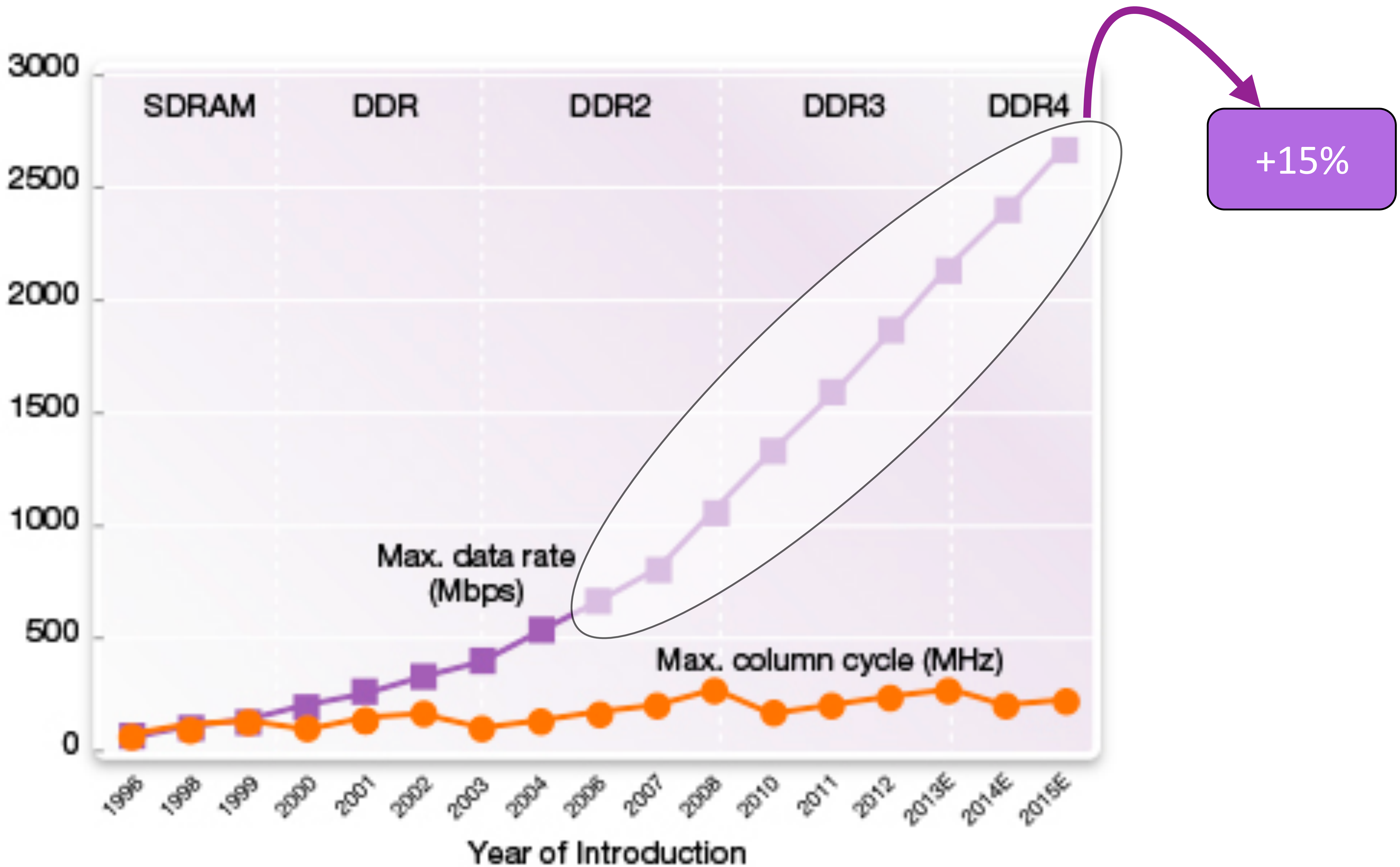
Where is the puck going?



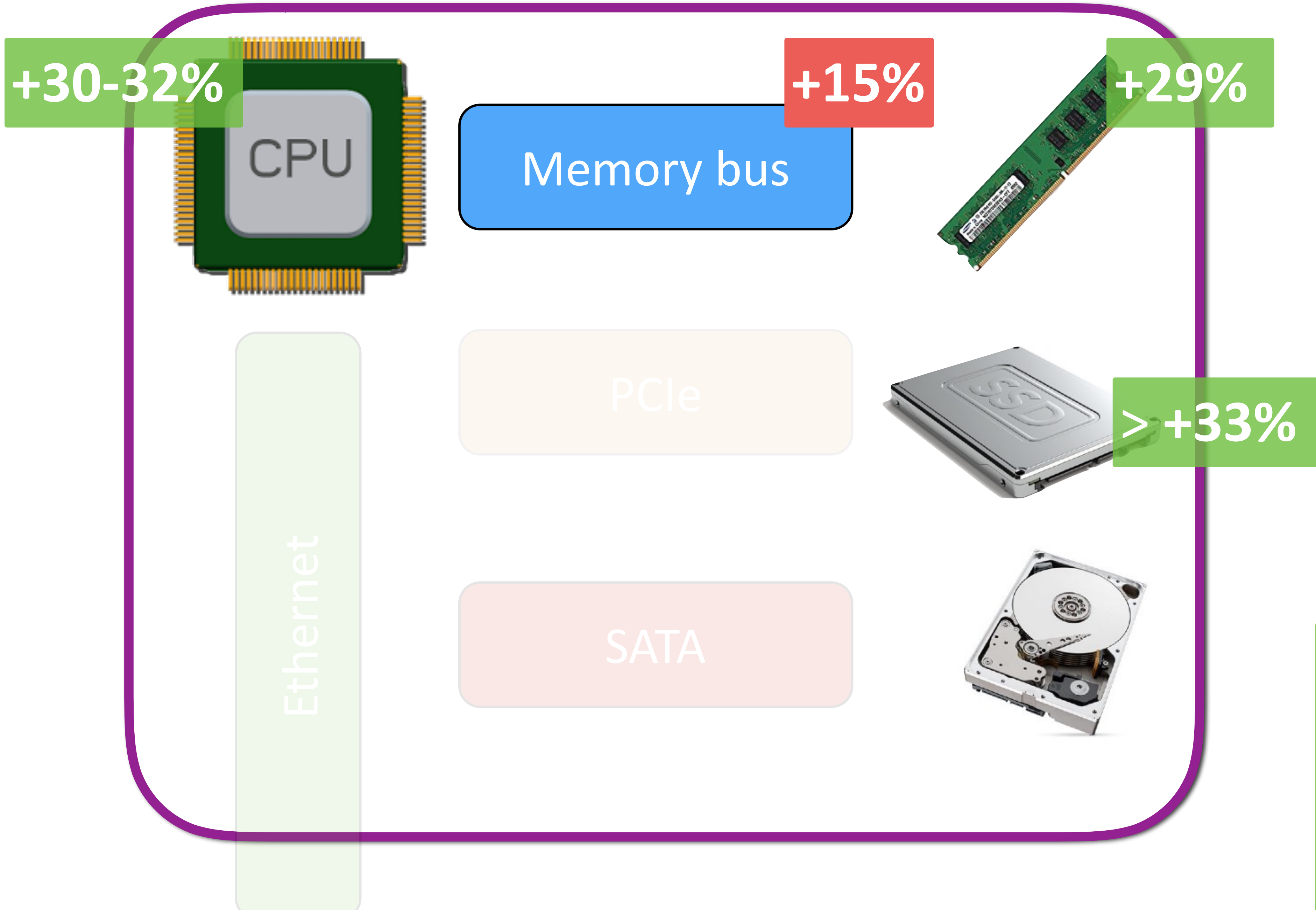
Tape is dead,
Disk is tape,
SSD is disk,
RAM is the king!

\- Jim Gray

Where is the puck going? (Memory bus)



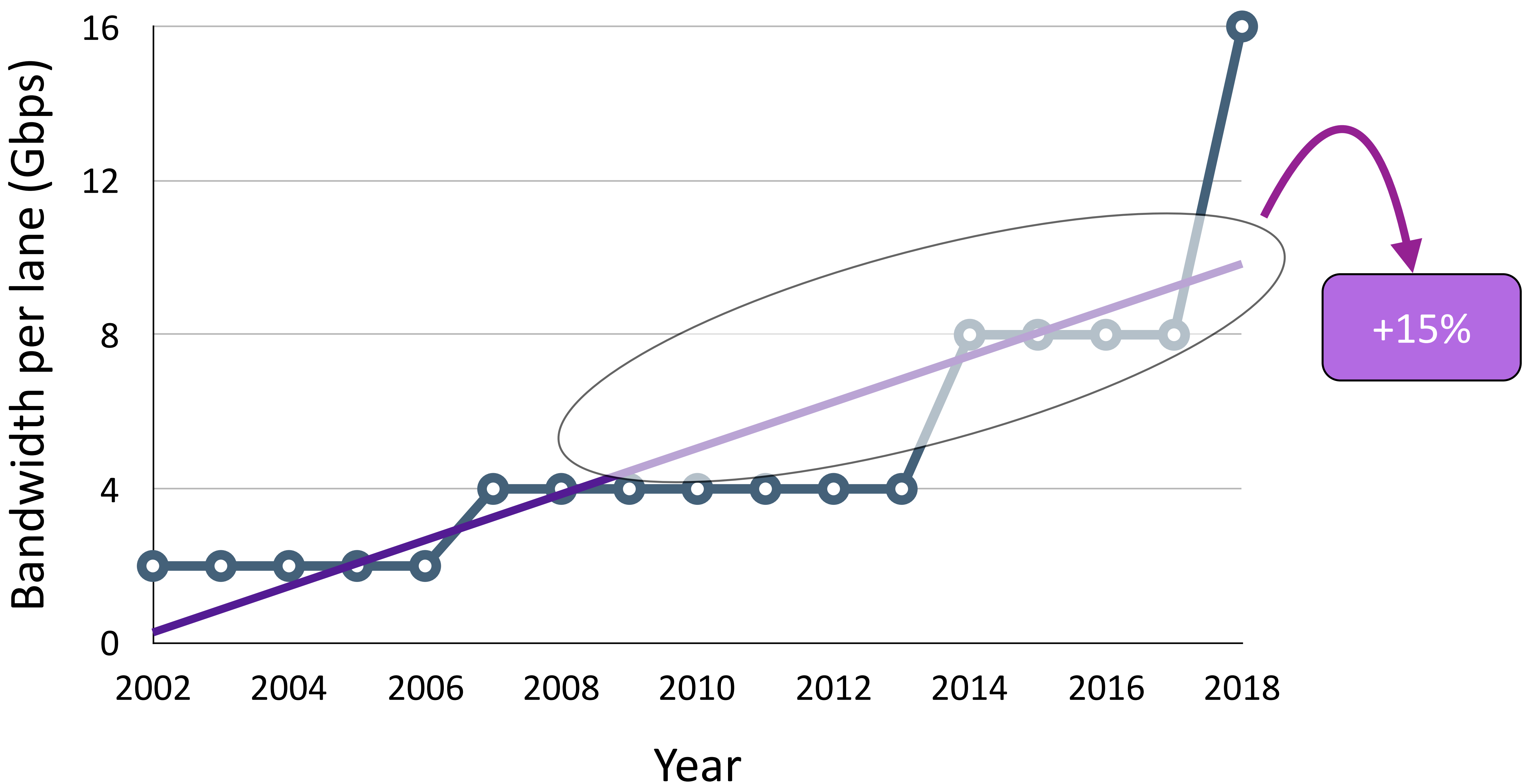
Where is the puck going?



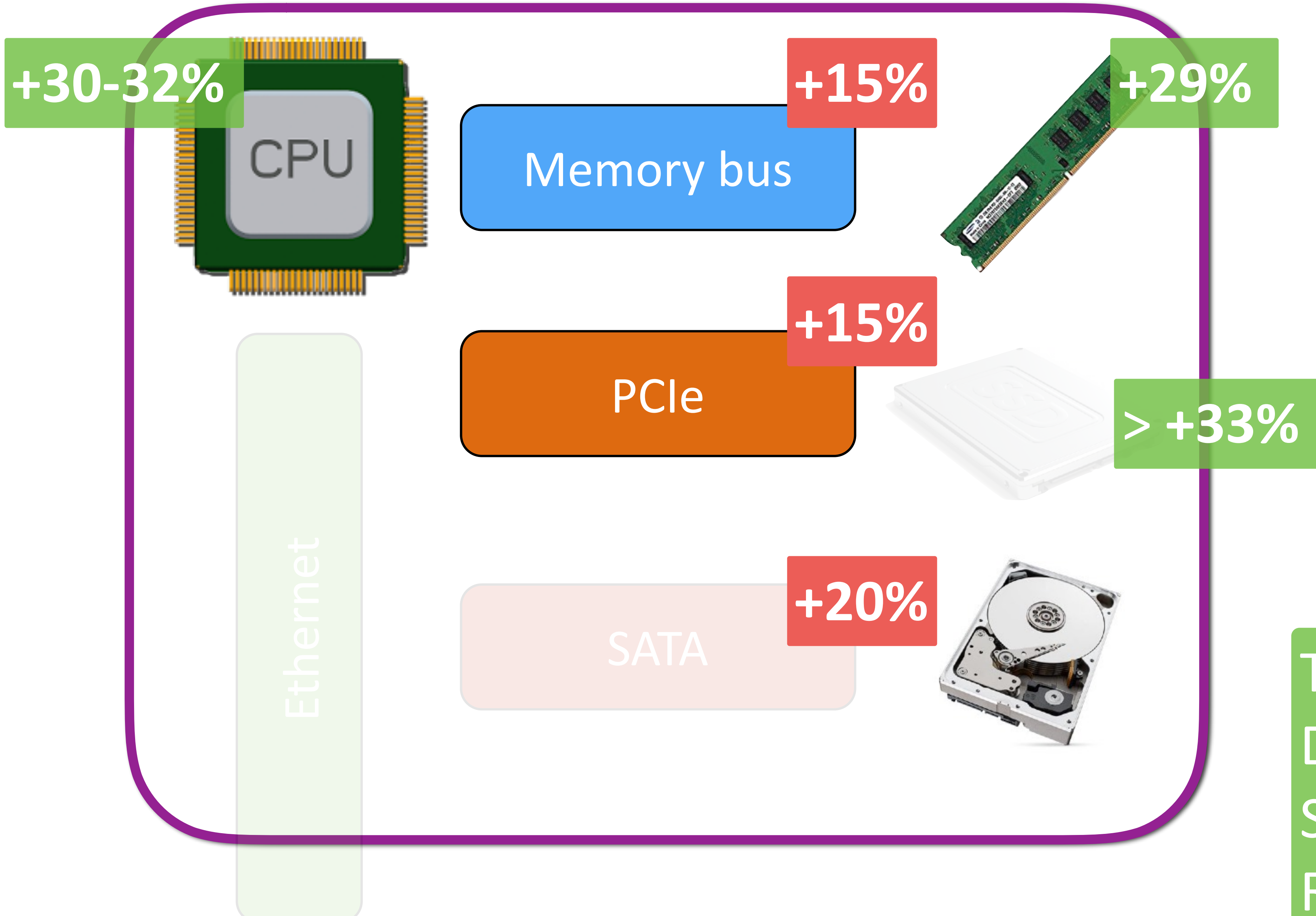
Tape is dead,
Disk is tape,
SSD is disk,
RAM is the king!

- Jim Gray

Where is the puck going? (PCIe)



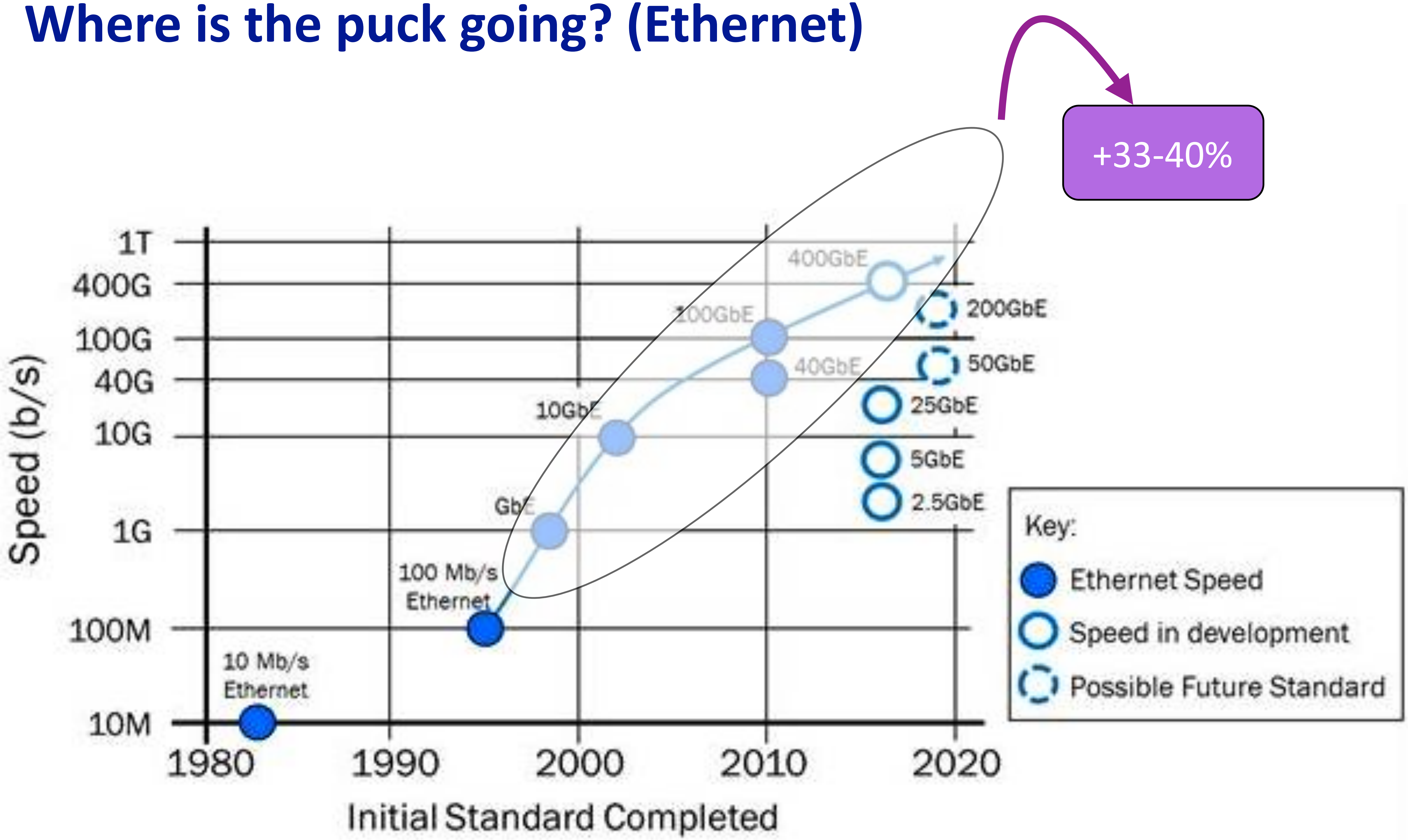
Where is the puck going?



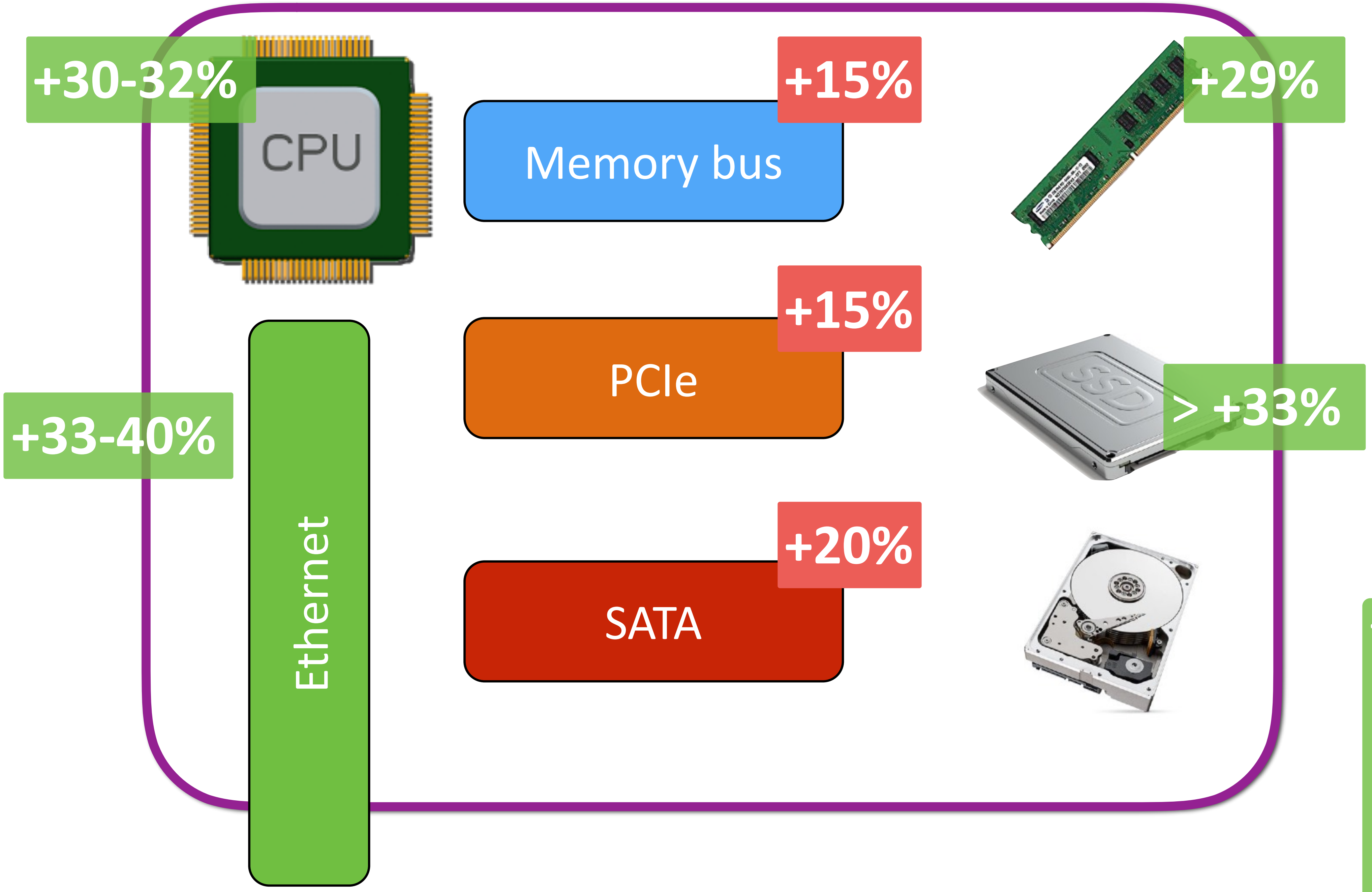
Tape is dead,
Disk is tape,
SSD is disk,
RAM is the king!

- Jim Gray

Where is the puck going? (Ethernet)



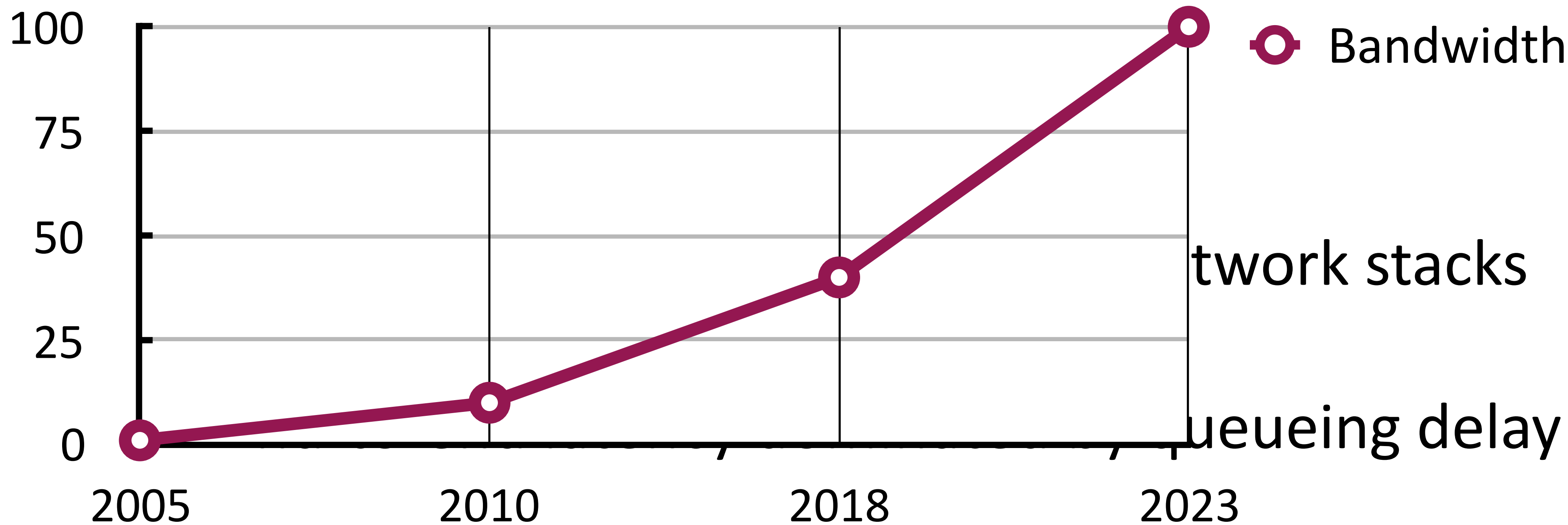
Where is the puck going?



Tape is dead,
Disk is tape,
SSD is disk,
RAM is the king!

- Jim Gray

Network Technology Trends



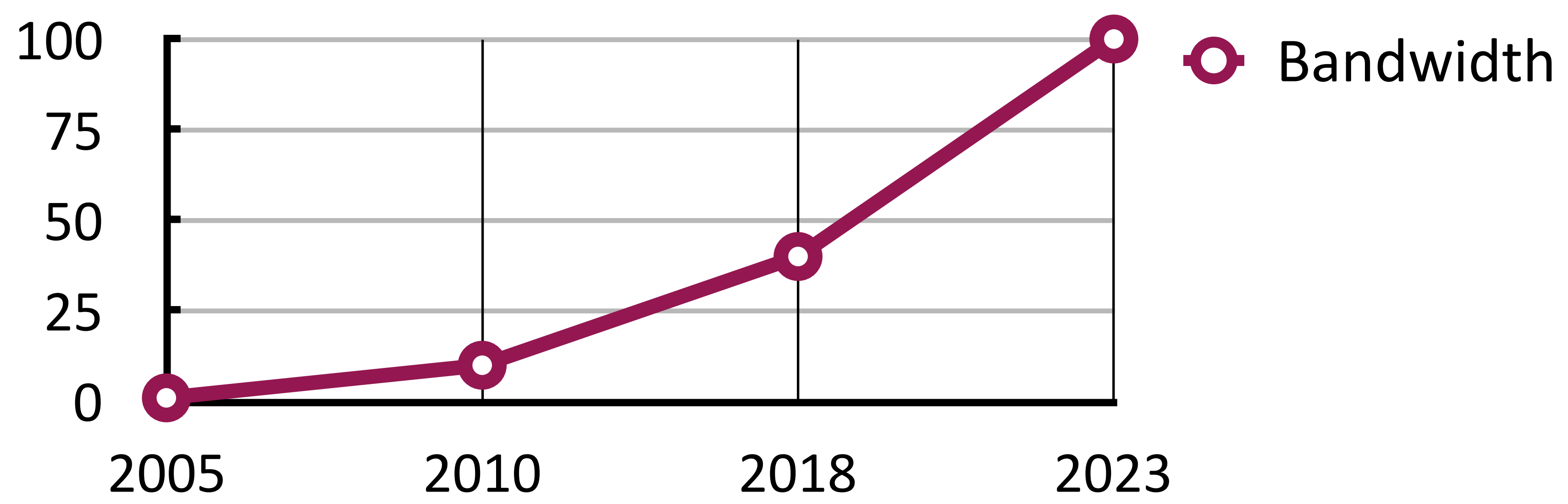
twork stacks

queueing delay

Powerful implications

- Remote memory faster than local SSD
  - When queueing delay = 0

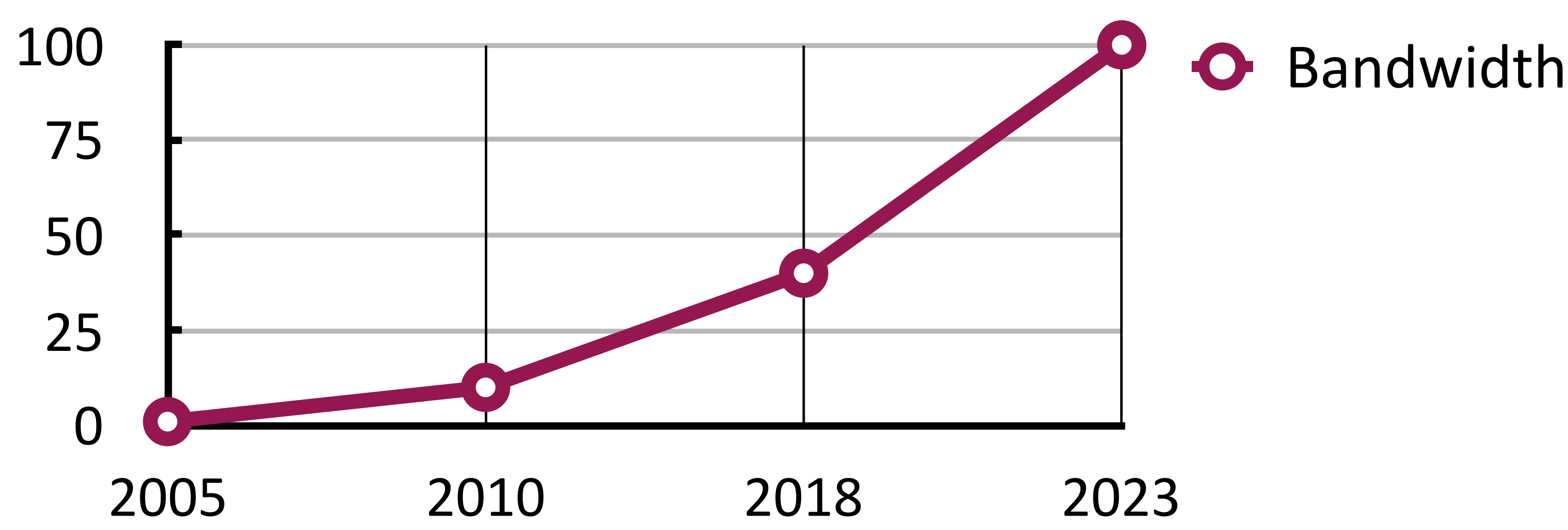
Unsustainable CPU overheads



- **Existing network stacks were designed for 1Gbps networks**
    - Known TCP problem: ~3.2Gbps per core
    - With low-level optimizations: ~9-12Gbps per core
        - 40Gbps would take >3 cores per server!
        - **100Gbps would take >8 cores per server!!**

- **Take away: unsustainable cloud economics**
    - Every core used for the stack is a core stolen from applications/ customers

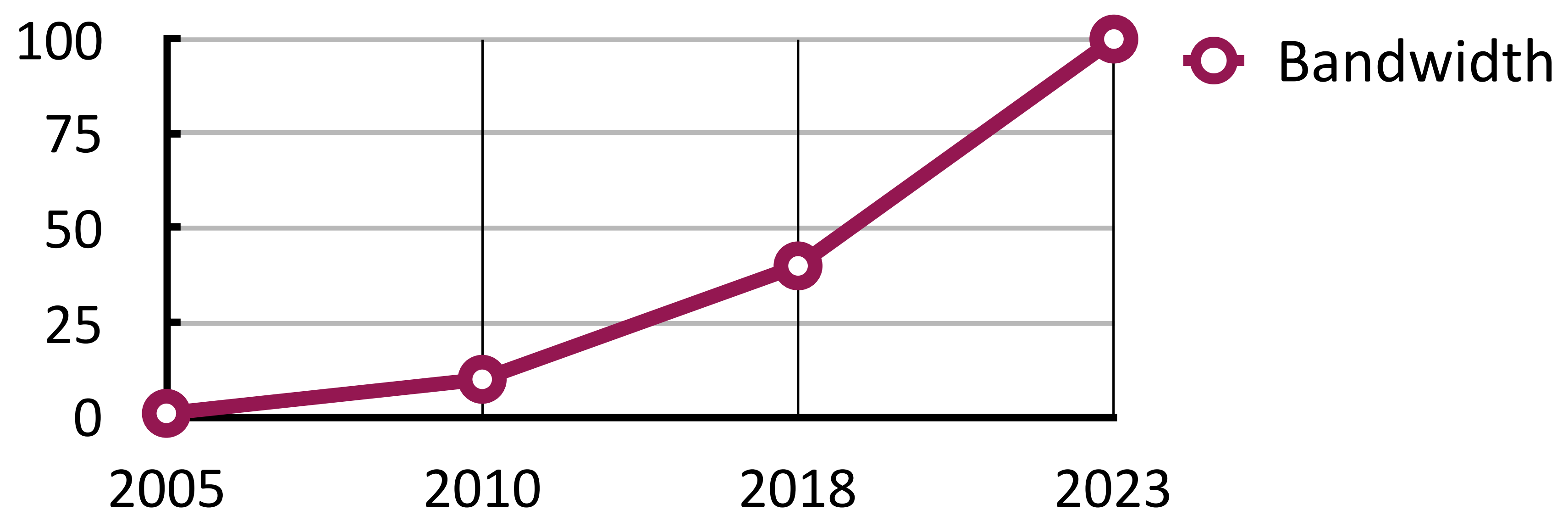
Curse of queueing delay



	~2005 (1Gbps)		2018 (40Gbps)	
	Latency (us)	%	Latency (us)	%
TOTAL	**18.92**		**6.30**	
Queueing (4MB buffers, 64 ports)	**488.3 (per congestion point)**		**12.21 (per congestion point)**	
Propagation delay	0.88	5	0.88	13
Transmission delay	11.44	61	0.29	5
TOTAL	**18.92**		**6.30**	
Queueing (4MB buffers, 64 ports)	**488.3 (per congestion point)**		**12.21 (per congestion point)**	

- **Take away: queueing delay is the core bottleneck**
  - **End-to-end latency bottlenecked by queueing delay**

Remote Memory Faster than Local Storage



- **Under zero queueing:**
    - Remote memory access takes less than 6.3us
    - Local SSD access latency today is 25us (hardware, ignoring stack)
    - Remote Direct Memory Access (RDMA) becomes feasible
    - **However, RDMA requires lossless network fabric**
        - Known problem with RDMA over Ethernet: congestion collapse
- **Take away: RDMA applicability limited by drops in network fabric**

Current Network Stacks are the Bottleneck!

- **Lot of research in "hardware offload"**
  - Implementing TCP (and other mechanisms) on hardware
  - Lots of interesting challenges

- **Lot of research in low-latency transport design**
  - TCP was not designed for low latency
  - New transport protocols for ultra low-latency

- **Lot of research in kernel-bypass**
  - TCP requires processing each and every packet
  - 1Gbps links: 90,000 packets per second
  - 100Gbps links: 9 million packets per second
  - Extremely high CPU requirements
  - Bypass the kernel entirely
    - Implement congestion control in user space, in hardware?