



NETFLIX

Disk | Crypt | Net

High performance video streaming

Ilias Marinos, Robert Watson (Cambridge),

Mark Handley (UCL),

Randall Stewart (Netflix)

Modern Video Streaming

- Just lots of HTTP requests for video chunks.
- Client picks chunks, so as to adapt rate.
- Server is pretty dumb – just has to go fast.
- HTTP/1.1 persistent connections.
- TLS has become important (YouTube is 95% TLS).

- More than 50% of Internet traffic.
- Important to make good use of expensive hardware. [How fast can you go?](#)

BBC Digital Media Distribution: How we improved throughput by 4x

Thursday 17 December 2015, 10:09



Alistair Wooldrige
Senior Software Engineer

Tagged with: [Media Distribution](#)

COMMENTS

BBC Digital Media Distribution has been working to deliver more throughput from their caching infrastructure. Senior Software Engineer Alistair Wooldrige explains how the team diagnosed poor performance with existing software and why replacing it achieved a 4x increase in performance.

Within the BBC Digital Media Distribution team, we used **Varnish cache** for the first version of our Radix caching servers. A Radix server caches HTTP responses from origin servers - usually video and audio content for iPlayer, delivered using one of the HTTP **Adaptive bitrate streaming** formats such as **MPEG-DASH**, **HLS** or **HDS**. For more information on Radix and our overall caching strategy, see [Digital Distribution: How on demand content reaches audiences](#).

About this Blog



Staff from the BBC's online and technology teams talk about BBC Online, BBC iPlayer, BBC Red Button and the BBC's digital services. The blog is reactively moderated. Your host is Robert Sheehy.

[Follow Internet Blog on Twitter](#)

[Blog home](#)
[Explore all BBC blogs](#)

Blog Updates

Stay updated with the latest posts from the

New iPlayer setup, Dec 2015:

- nginx on Linux, **24 cores** on two Intel Xeon E5-2680v3 processors, **512 GB DDR4 RAM**, 8.6TB RAID array of SSDs.
- **20Gb/s** per server. ← **Not so impressive!**

NETFLIX

- FreeBSD, but tweaked.
 - Asynchronous sendfile()
 - Non-blocking zero copy from disk buffer cache to Net.
 - VM scaling
 - Fake NUMA domains to avoid lock contention
 - Proactive cleanup of disk buffer cache.

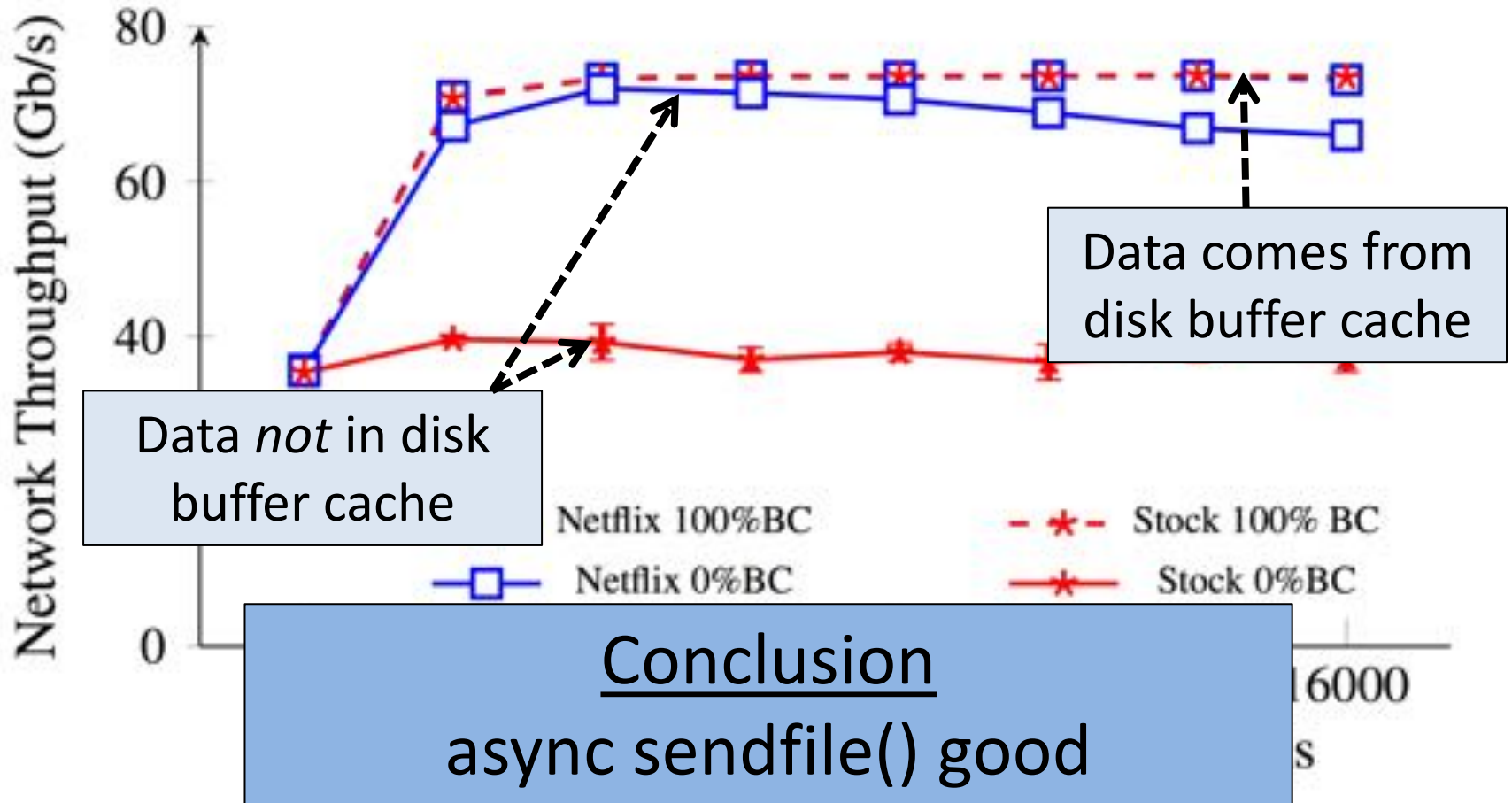
Lets Do Some Experiments

- 8 core server, 2x 40Gb/s NICs, 4x NVMe SSDs
- Synthetic workload, middlebox adds delay to reverse path to make RTT realistic.
- Theoretical limits: about 73Gb/s

Compare:

- Stock FreeBSD/nginx
- Netflix version of FreeBSD/nginx
 - production codebase in autumn 2016

Unencrypted video streaming workload



Encryption Problem

sendfile():

- Zero copy from disk buffer cache

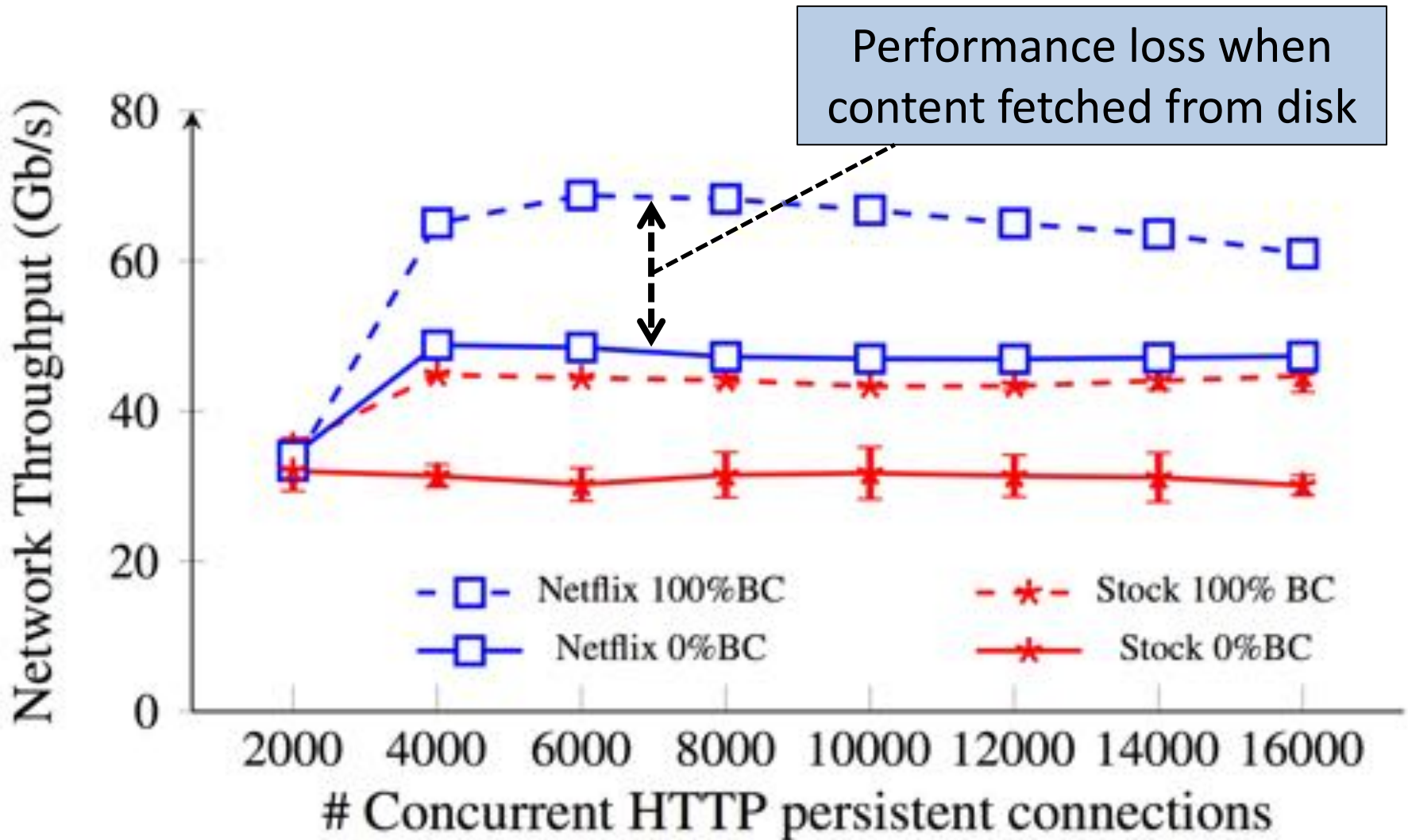
TLS:

- Different encrypted stream per user.

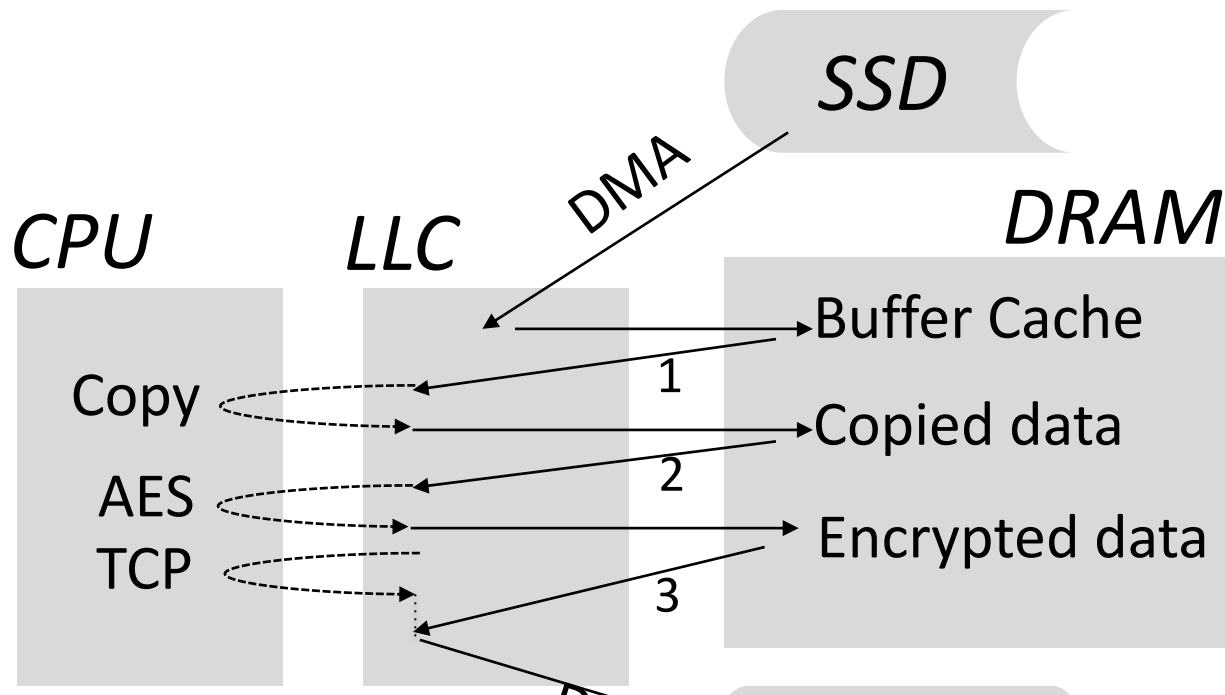
Sendfile() and TLS are incompatible!

- Conventional TLS stack gave Netflix 8.5Gb/s
- Netflix implemented kernel stream encryption for encrypted sendfile.
 - But can't be zero copy anymore.

Encrypted video streaming workload



What's happening?



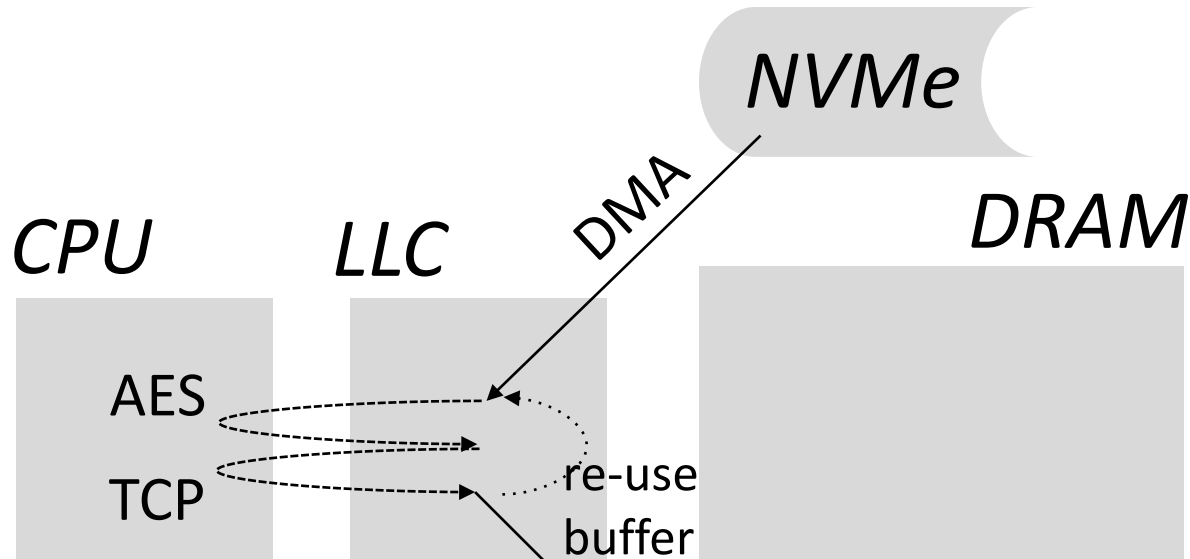
The stack is too asynchronous.

Data keeps getting flushed from the LLC, and re-loaded.
System is bottlenecked on memory.

Production Netflix workload

- 192GB of RAM as disk buffer cache.
- Only gets a 10% hit ratio.
- Modern NVMe SSDs have low latency.
- Modern Intel CPUs DMA directly to the L3 cache.
- Can we eliminate the disk buffer cache completely, and fetch everything from SSD on demand?

Ideal Stack



To achieve this, we must:

- Fetch on demand from the SSD when TCP needs data.
- As soon as the SSD returns data, process it to completion and DMA it to the NIC.

Solution Outline

1. A TCP ACK arrives, freeing up congestion window.
2. Trigger storage stack to request more data from SSDs to fill that congestion window.

***Atlas*: a complete user-space stack:**

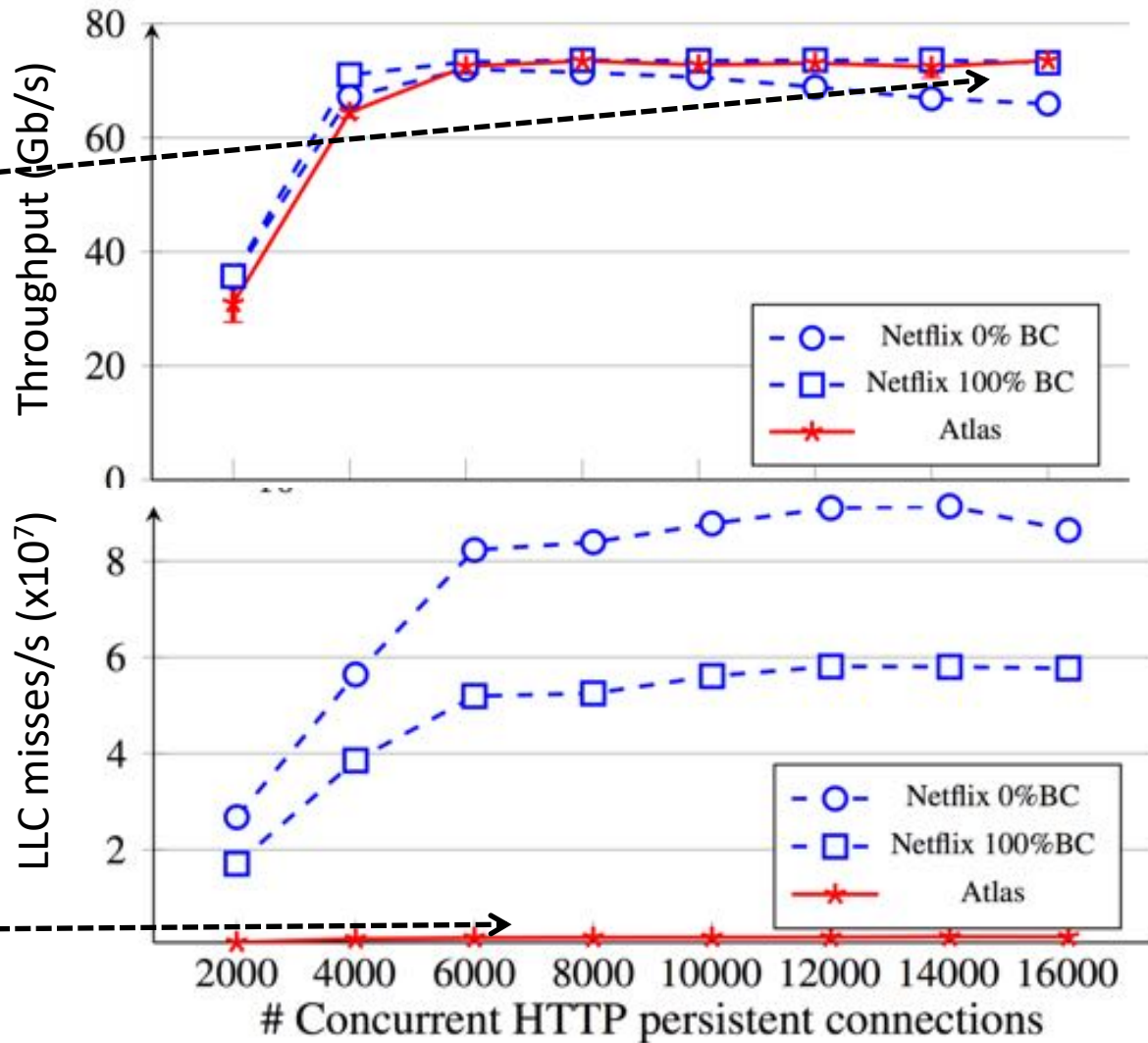
- *Netmap* to handle the network
- User-space TCP.
- Implemented *diskmap*, storage equivalent of netmap
 - Maps NVMe buffer rings to userspace and manages submission and completion queues.

Atlas vs Netflix, unencrypted content

15% better throughput than Netflix when cache hit ratio is low.

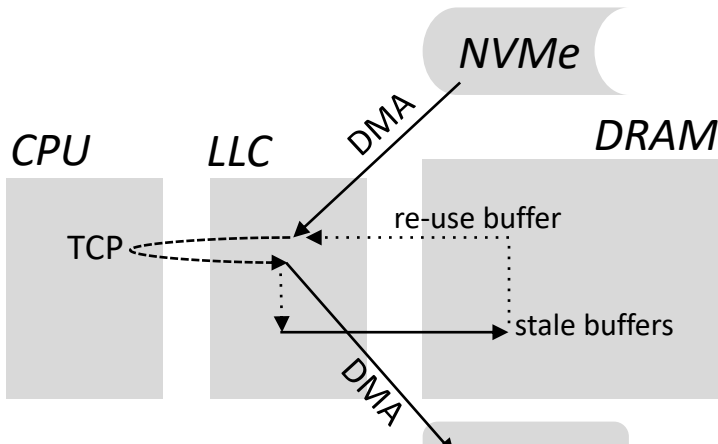
Netflix needs 8 cores, Atlas only needs 4

Almost no LLC misses. Data in LLC when we want it.

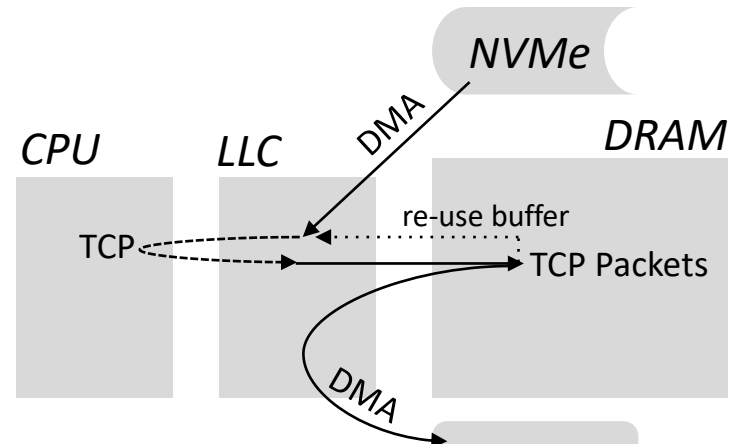


Still some memory traffic though

If we're lucky:



If we're unlucky:

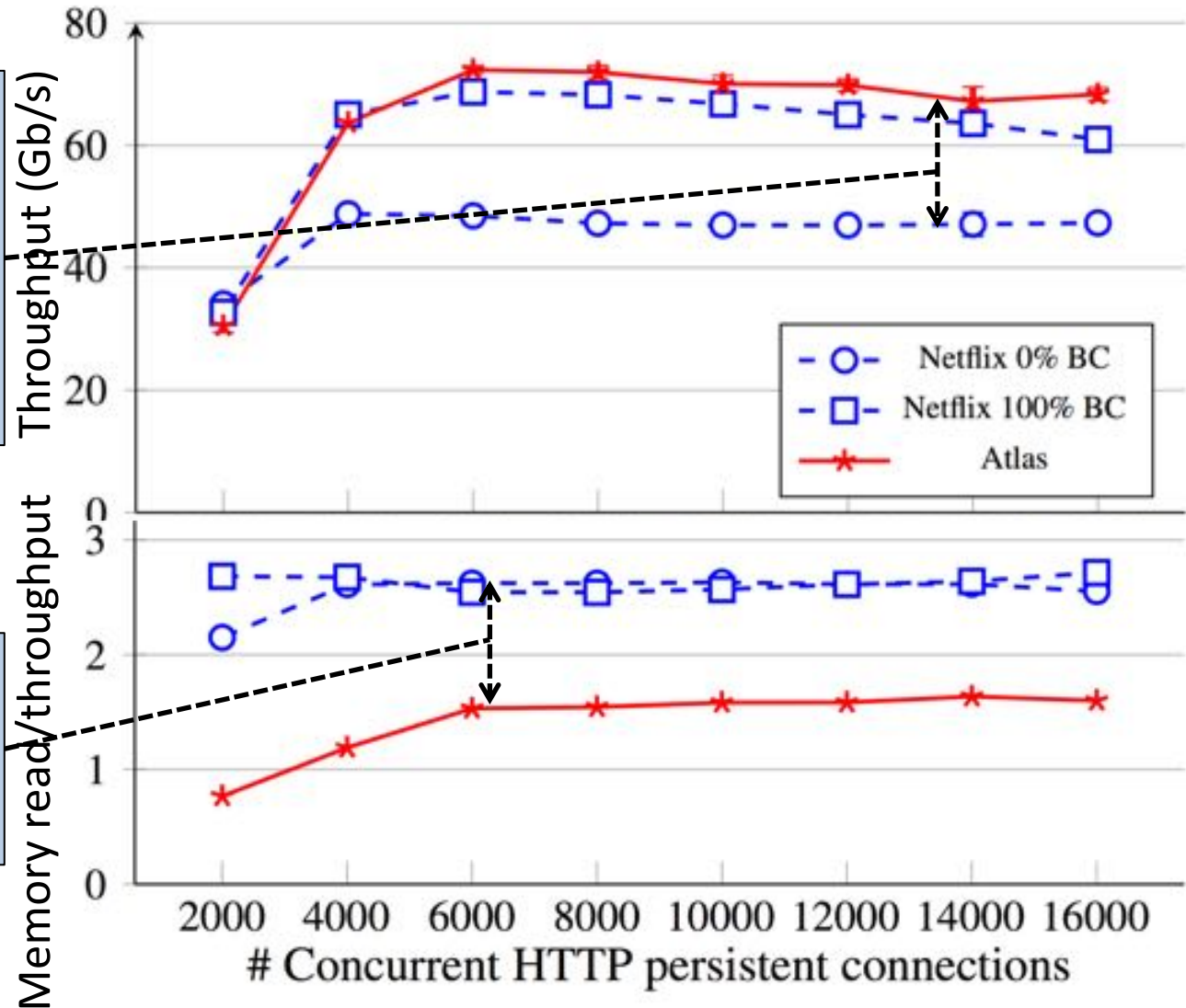


Netmap doesn't provide a good low-delay fine-grained way to handle DMA completions. Can't reuse buffers fast enough, and this adds some cache pressure.

Atlas vs Netflix, Encrypted Content

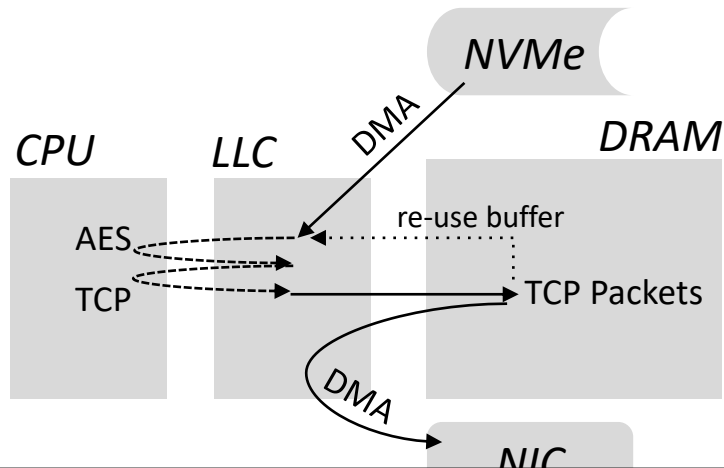
When cache hit ratio is low, 50% more throughput using half the cores.

Almost half the memory reads for each packet sent.

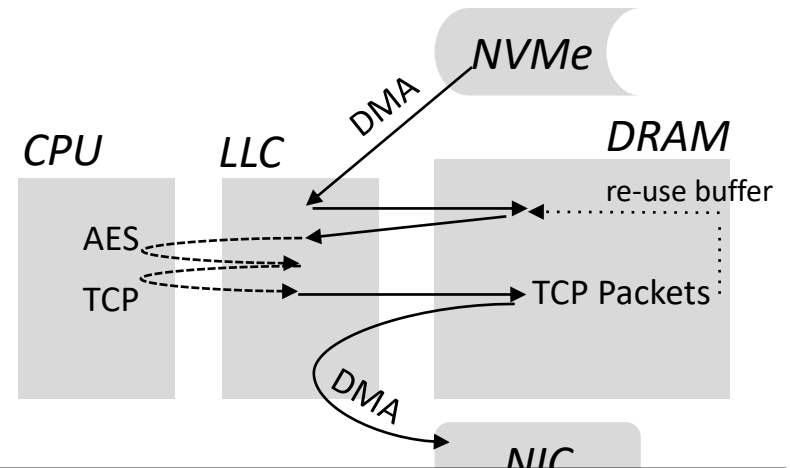


Atlas TLS memory usage

With no premature cache eviction:



When some other DMA causes cache eviction



DDIO can only DMA to 10% of the LLC

To operate entirely from the LLC, we'd need to recycle netmap buffers faster, using a LIFO stack for free buffers. We'd need more control over DDIO policy (newer CPU!)

Summary

- Netflix addressed all the low-hanging fruit
 - Very fast, but now **bottlenecked on memory**
- Atlas is a **specialized stack**
 - Puts SSD in TCP control loop
 - Immediately processes disk reads to completion and transmits.
 - 50% throughput improvement with encrypted content, close to 50% reduction in memory reads
- Netflix inspired by Atlas
 - Now experimenting with how to directly trigger encryption off of disk DMA completions in their FreeBSD stack.