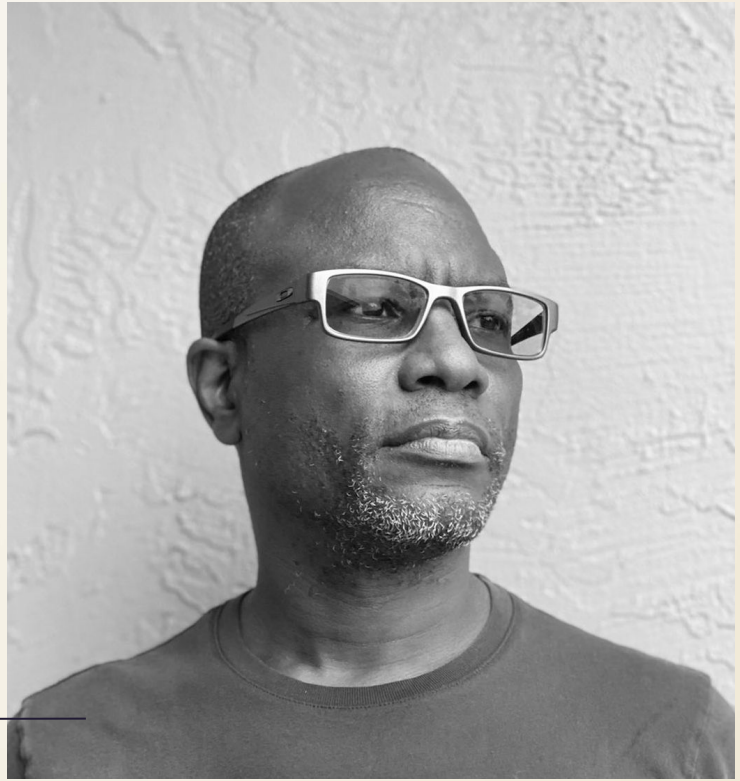
NSTREAM

Data Locality

The path to effective net-zero latency

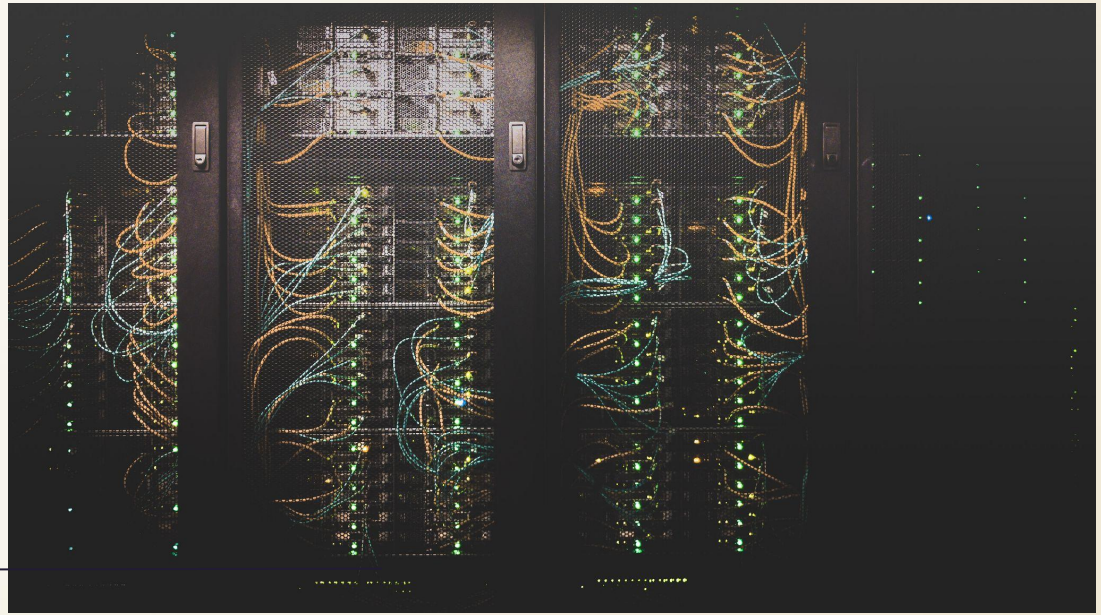
Fred Patton

- Developer Evangelist at Nstream
- Long-time backend engineer, generalist, and polyglot
- Serverless and fullstack by night
- Avid technologist
- Lover of the arts and various spoken languages



Keep your processing jobs local

Data locality is the concept of placing computation in the proximity of the data it requires. This can reduce network latency and improve overall system performance by avoiding large, recurring data movements.



Ground Zero data locality

Nstream's open-source SwimOS platform doesn't simply situate its processing unit near its requisite datastores. As a stateful object, the processing unit is its own datastore.

Giants steps in magnitude

Varying orders of magnitude in data access speeds differentiate system component types. Data locality allows for keeping to a low order of milliseconds, or even microseconds or nanoseconds, by avoiding slower access methods.

Power laws of latency

Registers (0.5ns), CPUs (1ns), Branch mispredict (3-20 cycles), M1/main memory (60-80ns), L3 cache (10-30ns)

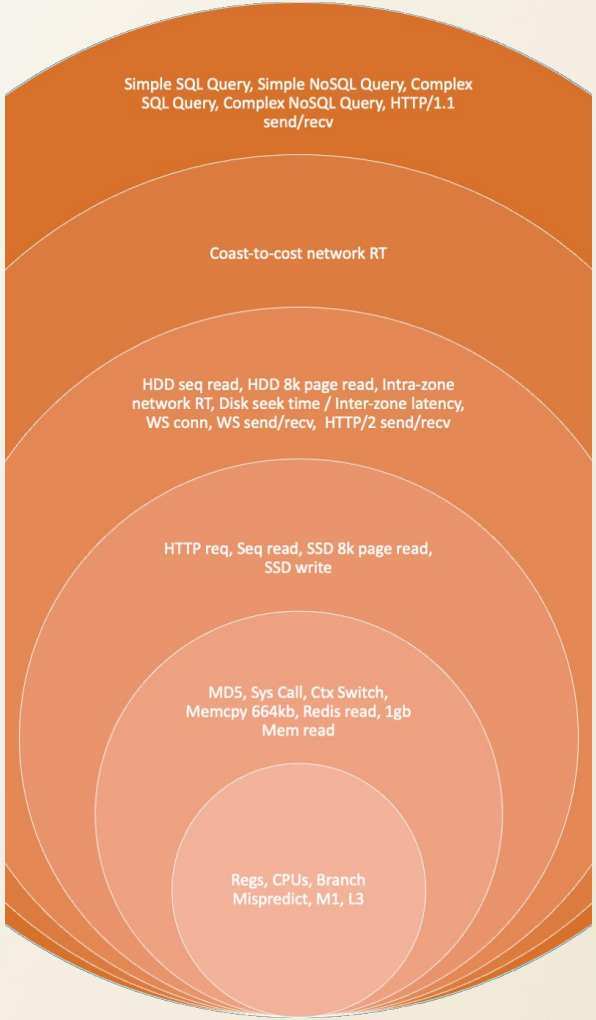
MD5 (0.4-1.5µs), System call (1-10µs), Context switch (2-10µs), Memory copy 664KB (1-3µs), Redis read (5-20µs), 1GB DDR4 memory read (10-30µs)

HTTP request (100-500µs), Sequential read (50-150µs), SSD 8K page read (50-150µs), SSD write (100-300µs)

HDD sequential read (5-20ms), HDD 8K page read (5-20ms), Websockets conn (20-200ms), Websockets send/recv (1-2ms), HTTP/2 conn (20-200ms), HTTP/2 send/recv (1-10ms), Intra-zone network RT (1-10ms), Disk seek time / Intra-zone latency (2-10ms)

Coast-to-coast network RT (50-100ms)

Simple SQL/NoSQL query (a few to 10s of ms), Complex SQL/NoSQL query (10s of ms - few secs), HTTP/1.1 send/recv (few - several secs)



Fitting under optimal latency profiles

- Optimal latency is the minimum latency required to move data from processing source to destination.
- To achieve the effect of net-zero latency, added processing must not increase latency by more than an order of magnitude below the current baseline.

How can this be achieved?

- It begins with mechanical sympathy
- Understanding varying data access costs
- Understanding the effect of operations on a systems level
- Avoiding the cost of polling with streaming APIs and uninterrupted streaming
- Minimizing the extent of transmission with differential state sync
- Keep requisite state together in continuous readiness through stateful processing

How has it been achieved

- Nstream intended to be a customer, but when the tools failed, it had to create them
- Nstream didn't recreate real-time data, but they needed to build applications running on top of it
- Nstream's SwimOS platform is the embodiment of fir principles for real-time applications:
- (1) Streaming APIs adding net-zero latency
- (2) Stateful Objects abstracting away requisite mechanical sympathy
- (3) Real-time UIs realizing end-to-end uninterrupted streaming

Get started!

- Visit us at: https://www.nstream.io/
- READMEs: https://github.com/swimos/swim
- Developer site: https://www.swimos.org/
- Sample Applications: https://github.com/swimos
- Cookbook: https://github.com/swimos/cookbook
- Tutorials: https://github.com/swimos/tutorial

Thank You!