جامعة الملك عبدالله
للعلوم والتقنية
King Abdullah University of
Science and Technology

In-Network Computation is a Dumb Idea Whose Time Has Come

Marco Canini

KAUST

Is this a dumb idea?

- increased complexity
- new kinds of failure modes to address
- could affect application correctness
- will put application-specific logic in the network…

But I didn’t wake up at 4AM to tell you something dumb

network programmability is the holy grail

I was already “programming the network” with network processors 13 years ago…

today’s context is very different

Tremendous requirements: reliability, velocity, performance

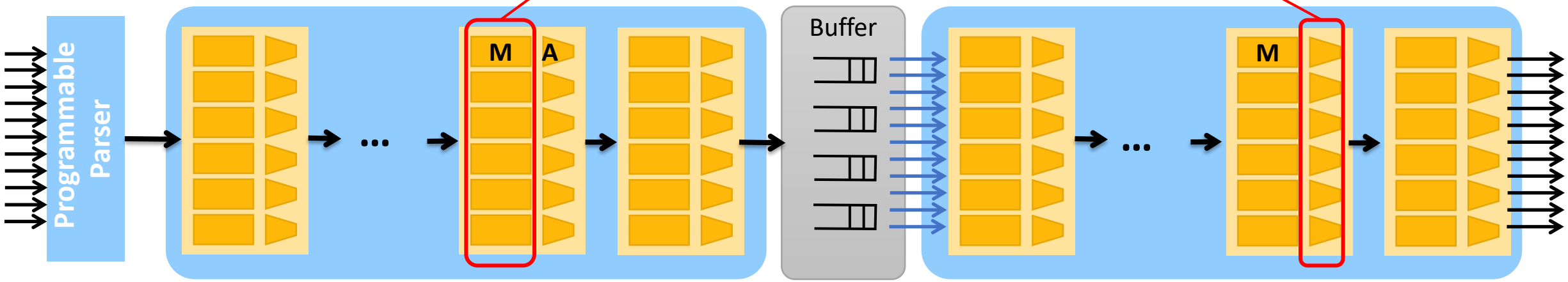
New HW breed: programmable ASICs, no power or cost penalties

as networks become capable of computation,
what kinds of computation should networks perform?

Network machine architecture w/ PISA

Match Logic
(Mix of SRAM and TCAM for lookup tables, counters, meters, generic hash tables)

Action Logic
(ALUs for standard boolean and arithmetic operations, header modification operations, hashing operations, etc.)



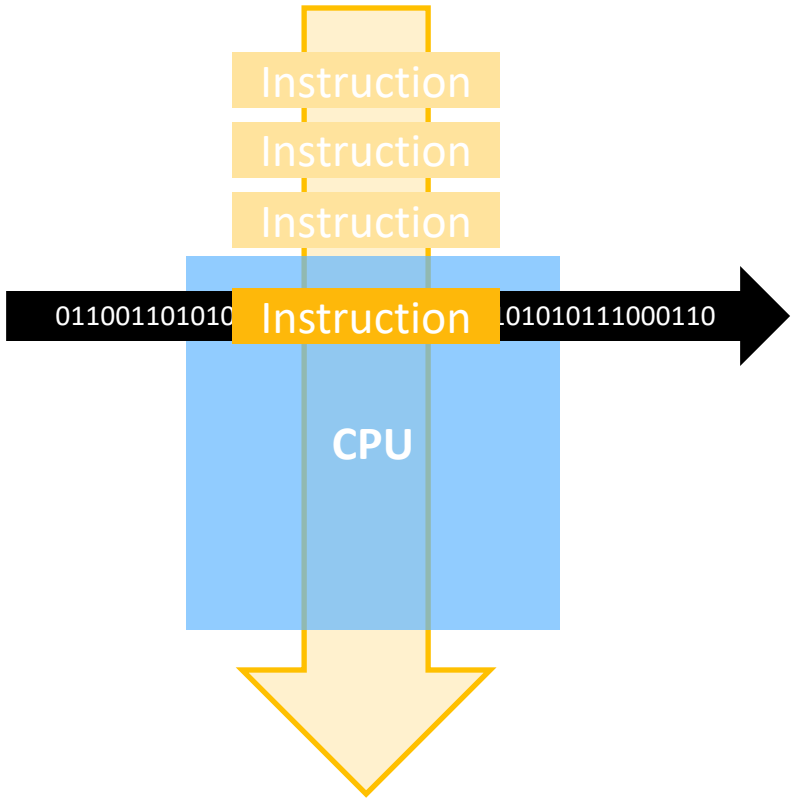
Ingress match-action stages (pre-switching)

Egress match-action stages (post-switching)

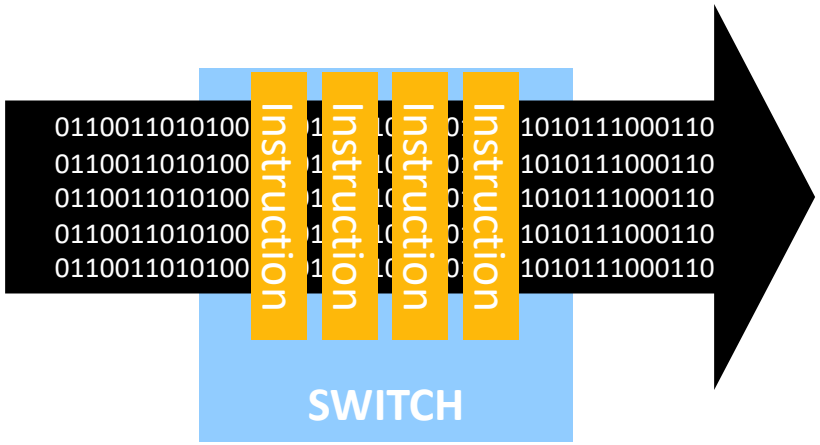
Slide courtesy: Changhoon Kim

Why is compute in the network different than a general purpose CPU?

Temporal vs spatial compute



Temporal compute



Spatial compute

Slide idea from: Microsoft Build 2017: Inside the Microsoft FPGA-based configurable cloud

Inherent challenges

- Computing on this architecture corresponds to executing streaming algorithms with stringent constraints on number and type of ops

- Not Turing complete
- Limited memory size
- Limited set of actions (per stage & pipeline length)
- Few operations per packet (line rate execution guarantee)

Judicious in-network computing

- Bring high benefits at low costs
  - Is the network the bottleneck? Reduce bytes or reduce latency?

- Must live within the confines of HW constraints
- Application correctness is paramount
- Should give benefits in common case and possibly during failures too
- In the limit, should be no worse than if without
- Should focus on primitives that are broadly applicable to class of apps
  - Identify reusable, high-level abstractions that promote easy adoption

Tremendous opportunity

- Target important, widely used distributed services in the network
- Examples:
  - Fault-tolerance protocols
  - KV storage systems
  - Distributed ML training

Consensus at line rate

SIGCOMM CCR 2016

Paxos Made Switch-y

Huynh Tu Dang* Marco Canini† Fernando Pedone* Robert Soulé*
*Università della Svizzera italiana †Université catholique de Louvain
{huynh.tu.dang,fernando.pedone,robert.soule}@usi.ch marco.canini@uclouvain.be

ABSTRACT

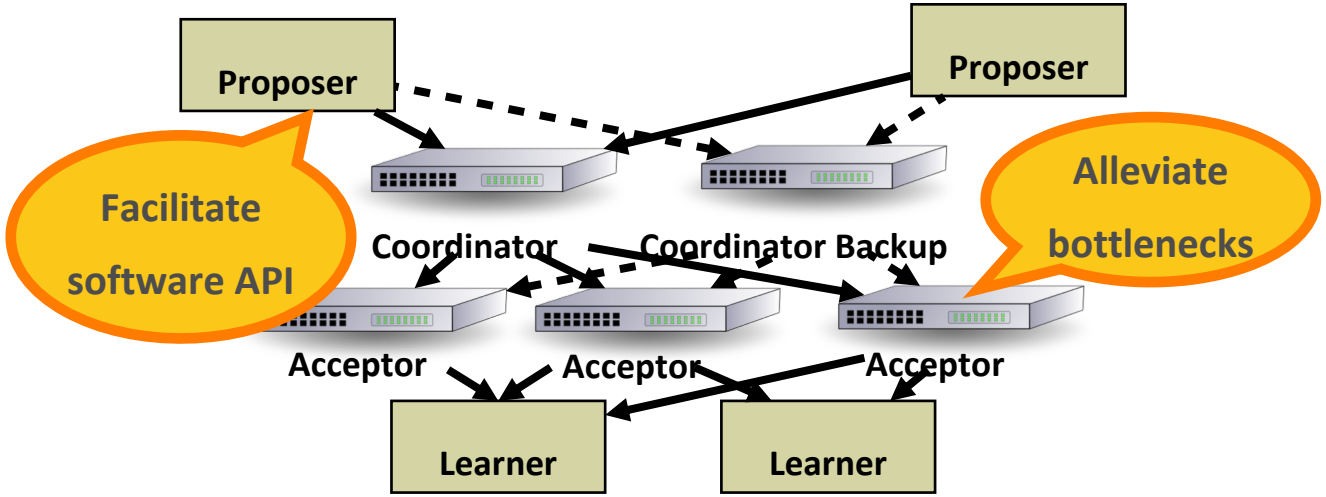
The Paxos protocol is the foundation for building many fault-tolerant distributed systems and services. This paper posits that there are significant performance benefits to be gained by implementing Paxos logic in network devices. Until recently, the notion of a switch-based implementation of Paxos would be a daydream. However, new flexible hardware is on the horizon that will provide customizable packet processing pipelines needed to implement Paxos. While this new hardware is still not readily available, several vendors and

cessing pipelines, including Protocol Independent Switch Architecture (PISA) chips from Barefoot networks [2], FlexPipe from Intel [11], NFP-6xxx from Netronome [20], and Xpliant from Cavium [30]. Such hardware significantly lowers the barrier for experimenting with new dataplane functionality and network protocols.

While this new hardware is still not readily available for researchers and practitioners to experiment with, several vendors and consortia have made *programming languages* that target these devices available. Notable examples include Huawei's POF [28], Xilinx's PX [3], and the P4 Consortium's P4 [1]. Consequently, it is

Example: NetPaxos

Consensus is a fundamental problem for fault-tolerant systems



- Offering consensus as a network service has significant performance benefits
- Implement Paxos logic in network devices
- Demonstrate consensus at 9 M msgs / s (4.3x improvement) and low latency (80% reduction)

µs	P4FPGA	SDNet	Netronome
Forwarding	0.37	0.73	-
Coordinator	0.72	1.21	0.33±0.01
Acceptor	0.79	1.44	0.81±0.01

Data aggregation for ML and graph analytics

HotNets 2017

In-Network Computation is a Dumb Idea Whose Time Has Come

Amedeo Sapio‡, Ibrahim Abdelaziz, Abdulla Aldilaijan,
Marco Canini, Panos Kalnis
KAUST

ABSTRACT

Programmable data plane hardware creates new opportunities for infusing intelligence into the network. This raises a fundamental question: what kinds of computation should be delegated to the network?

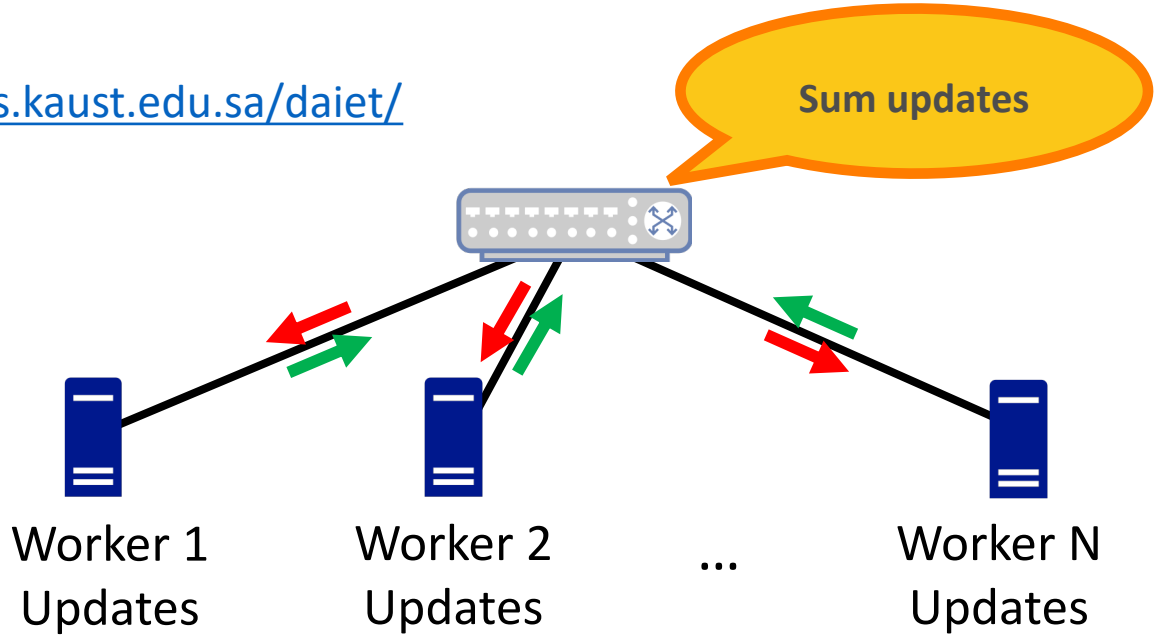
In this paper, we discuss the opportunities and challenges for co-designing data center distributed systems with their network layer. We believe that the time has finally come for offloading part of their computation to execute in-network. However, in-network computation tasks must be judiciously crafted to match the limitations of the network machine architecture of programmable devices. With the help of our experiments on machine learning and graph analytics workloads,

Programmable networks create the opportunity for in-network computation, i.e., offloading a set of compute operations from end hosts into network devices such as switches and smart NICs. In-network computation can offer substantial performance benefits, as it is for example the case with consensus protocols [9, 10] and in-network caches [20]. Although traditional networks are not capable of computation, the idea of using the network not just to move data, but also to perform computation on transmitted data is reminiscent of *Active Networks* [30], which proposed to replace packets with small programs called "capsules" that are executed at each traversed switch. However, for the past two decades the hardware capabilities were lacking. This appears to be

Example: DAIET

https://sands.kaust.edu.sa/daiet/

Data aggregation is a common task in many DC apps; high potential for ML



- Offload aggregation task to switches to alleviate communication bottlenecks and improve overall training time
- Exploit full network bandwidth of workers

Aggregation micro-benchmark:

- 1.28GB, 320M-element tensor
- Tofino switch
- 2 to 8 workers at 10Gbps

Results:

Transfer time 1.9 s (1.56s optimal limit)

Time has come for in-network computing
The ball is in your court

- Some other ideas:
  - Congestion control, resource allocation (HotCoCoA, Sweden, Sharma et al. NSDI ’17)
  - Network measurements (Marple, Sonata, Dapper, INT)
  - Load balancing (SilkRoad)
  - Consensus protocols (NetPaxos, NoPaxos, Eris, NetChain)
  - Caching (IncBricks, NetCache)
  - Stream processing (Linear Road)

With great powers come great responsibilities

Use in-network computing judiciously!