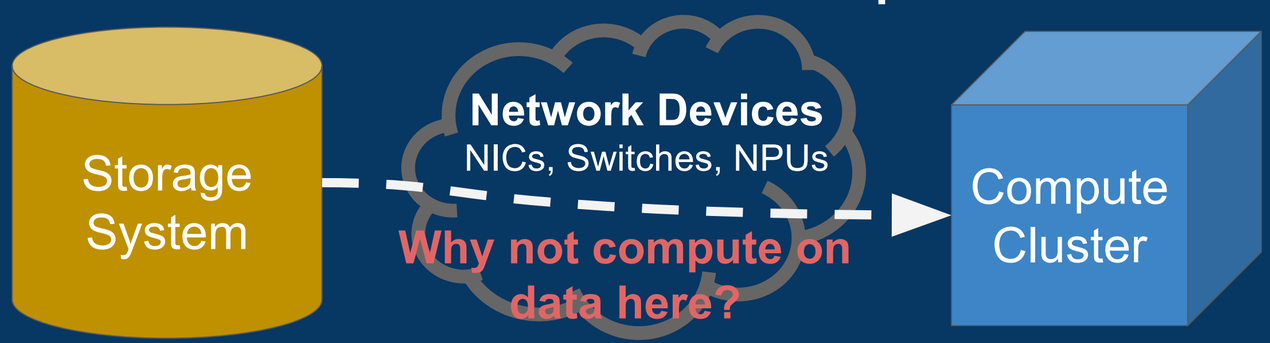
JumpGate: Towards In-Network Data Processing

Craig Mustard*, Fabian Ruffy, Alexandra Fedorova, and Ivan Beschastnikh
University of British Columbia, Vancouver, Canada
*craigm@ece.ubc.ca

Motivation

Can data be processed as it moves through the network?



We propose to apply database operations to data in transit!

Advantages:
- Reduce network traffic and compute load.
- Flexibly (de)allocate processors on demand.

Leverage existing opportunities:
- Programmable packet processors already exist.
- Many database operators are stateless.
- Query engines support operator offload.

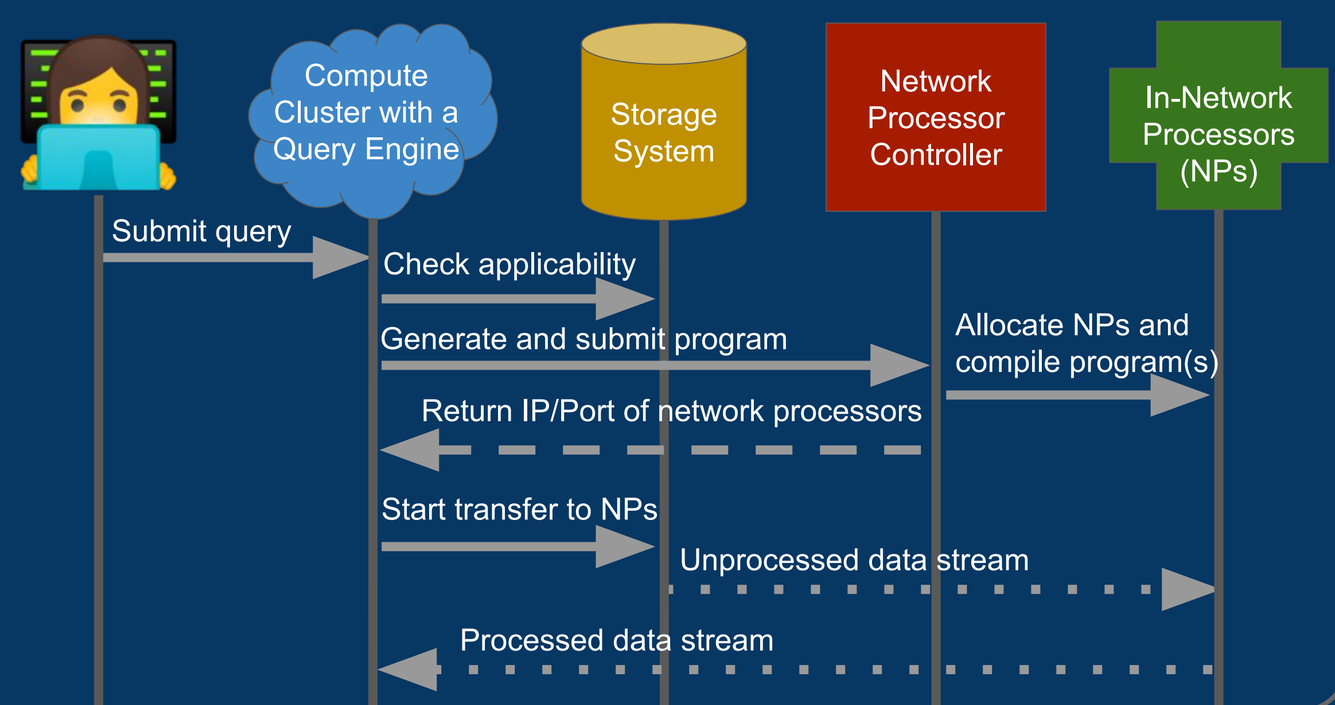
JumpGate

A research system to evaluate costs, benefits and challenges of in-network processing.

Target Features:
Data Formats: JSON + columnar format
Query Engine: Spark
Compile queries to: C / P4 / eBPF
Target both software and hardware packet processors

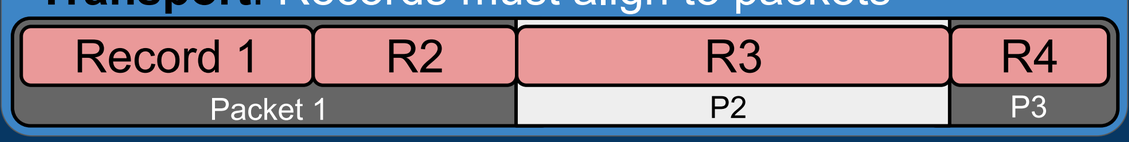
Potential Contributions:
1. Study the upper bounds on benefits in various settings
2. Operator implementations for current HW/SW
3. Fault tolerant protocol
4. Recommend improvements to packet processing HW

Example Architecture and Query Execution



Early Challenges

Transport: Records must align to packets



Record Parsing in Hardware Packet Processors:
Must parse: SCTP, JSON, columnar formats
HW packet processors limited to fixed length formats

Applicability detection:
Use JumpGate only when appropriate and beneficial

Fault detection and recovery:
Controller should detect transfer failures and adjust

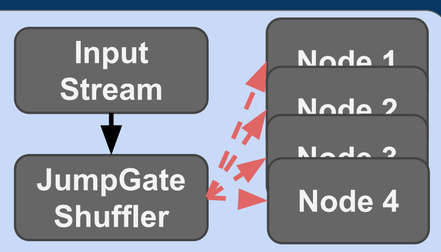
Benefits: Effects on Bandwidth

SCTP vs TCP:
~8% B/W increase

Filter and Projection:
Traffic reduced by selectivity (S)
(S% traffic reduction)

Simulation Details:
Mininet + Python
Dataset:
TPC-DS store_sales
JSON (~600b records)
Operator Type:
Precise

Shuffle: Eliminate intra-node exchange



(up to 50% reduction in traffic)

Aggregation: Early combine reduces traffic up to 10x

(includes implicit projection)

Related Work

JumpGate is inspired by lots of recent work:
Sonata / Marple: Network telemetry query processing using packet processors.
RMT, SmartNICs, NPUs: Line-rate packet processing hardware in production
Unstructured Data Parser: Parse variable length formats in hardware.
Sparser: Filter (approximately) before you parse.
Flare: Native compilation from Scala to C.

[Sonata] Gupta et al,. “Sonata: query-driven streaming network telemetry” (SIGCOMM '18)
[Marple] Narayana et al., ”Language-Directed Hardware Design for Network Performance Monitoring.” (SIGCOMM '17)
[RMT] Bosshart et al., “Forwarding metamorphosis: fast programmable match-action processing in hardware” (SIGCOMM 2013)
[SmartNICs] Firestone et al., Azure Accelerated Networking: SmartNICs in the Public Cloud” (NSDI 2018)
[UDP] Fang et al., “UDP: a programmable accelerator for extract-transform-load workloads and more” (MICRO 2017)
[Sparser] Palkar et al., “Filter before you parse: faster analytics on raw data with sparser” (VLDB 2018)
[Flare] Essertel et al., “Flare: Optimizing Apache Spark for Scale-Up Architectures and Medium-Size Data” (OSDI 2018)

Why not…

Execute operators on the storage/compute nodes?

- JumpGate is a complementary approach.
- May not want to provision resources for client compute.
- HW packet processors can have better throughput.
- JumpGate nodes not tied to storage, easily released after use.