Jumpgate: Automating Integration of Network Connected Accelerators

Craig Mustard, Swati Goswami, Niloofar Gharavi, Joel Nider, Ivan Beschastnikh, Alexandra Fedorova
University of British Columbia

Analytics Systems

Jumpgate

Network Connected Accelerators
Prior work showed NCAs can accelerate data analytics

Smart NICs (FPGAs) 96% increased throughput [Floem]

Programmable Switches
2-10x speedup
[NetAccel, DALET, Cheetah]

>1000x less traffic
[Sonata]
There are many places for Network Connected Accelerators
There are many places for Network Connected Accelerators

Alternative Data Path

Software Middleboxes
4.5x speedup
[NetAgg]
Remaining Challenges to actually using NCAs for analytics.

→ Integration with existing analytics systems:
  ◆ Current execution models too complex for NCAs.
  ◆ How can existing analytics systems use NCAs?
  ◆ How to read data stored in analytics formats?

→ Manage multiple devices at the same time:
  ◆ Specialized devices not good at all parts of a query

→ Each NCA is tough to program and limited:
  ◆ Not like a normal ‘worker node’
  ◆ Diverse hardware & limited storage

Need to solve these challenges to make NCAs generally usable for analytics tasks.
How should we **integrate** NCAs into analytics systems?

<table>
<thead>
<tr>
<th>Target Devices</th>
</tr>
</thead>
<tbody>
<tr>
<td>Switches</td>
</tr>
<tr>
<td>Smart NICs</td>
</tr>
<tr>
<td>Ephemeral VMs</td>
</tr>
<tr>
<td>N(etwork) PUs</td>
</tr>
<tr>
<td>FPGAs</td>
</tr>
<tr>
<td>D(at) PUs</td>
</tr>
<tr>
<td>Storage System</td>
</tr>
</tbody>
</table>
How should we integrate? One option:

Target Devices
- Switches
- Smart NICs
- Ephemeral VMs
- N(etwork) PUs
- FPGAs
- D(ata) PUs
- Storage System
How should we integrate? One option:

**Target Devices**
- Switches
- Smart NICs
- Ephemeral VMs
- N(etwork) PUs
- FPGAs
- D(ata) PUs
- Storage System

**Problems:**
- Not scalable to all analytics systems.
- Not future-proof to new devices.
- Hard to share NCA implementations.

---

**PostgreSQL**

---

**Presto**
How should we **integrate**? One option:

**All** prior work manually integrated each NCA implementation into each analytics systems, and orchestrated execution by hand.  
[Cheetah, Daiet, NetAccel, NetAgg]
Our proposal: Network Processing as a Service

Target Devices
- Switches
- Smart NICs
- Ephemeral VMs
- N(etwork) PUs
- FPGAs
- D(ata) PUs
- Storage System

Network Processing as a Service (NPaaS)
Our proposal: Network Processing as a Service

Direct Integration:
- Not scalable to all analytics systems.
- Not future-proof to new devices.
- Hard to share NCA implementations.

NPaaS:
- Abstracts devices and management.
- One time change to existing systems.
- New devices and systems can be added independently.
Jumpgate Overview

Client Systems:
- Spark
- Presto
- Python

Logical Dataflow Interface

Jumpgate

Operator & Life-cycle Interface

Relational Operator Implementations for NCAs:
- Programmable Switches
- FPGAs
- VMs
- In-Storage Compute

Contributions:
1. Establish the right API for existing analytics systems.
2. A novel execution paradigm needed to use NCAs for analytics.
3. Interfaces to simplify controlling and adding NCAs.
4. An overall evaluation of using software-based NCAs for data analytics.
Contribution #1: Jumpgate Client API:

1. User Query:

```
SELECT sum(price), t.state  
FROM sales s  
INNER JOIN stores t  
ON s.store_id = t.id  
WHERE s.item_id = 100  
GROUP BY t.state
```
Jumpgate Internals: Mapping Requests to NCAs

NCA implementations declared via **Operator Interface**:

<table>
<thead>
<tr>
<th>Relational Operations</th>
<th>Runs On</th>
<th>Protocols</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scan - Filter - Project</td>
<td>Software</td>
<td>TCP / UDP</td>
</tr>
<tr>
<td>Join - Project</td>
<td>Software</td>
<td>TCP / UDP</td>
</tr>
<tr>
<td>Partial Aggregation</td>
<td>Prog. Switch</td>
<td>UDP</td>
</tr>
</tbody>
</table>

NCA Instances to be run:

- **NCA1**: project, filter, stores
- **NCA2**: project, filter, sales
- **NCA3**: project, join
- **NCA4**: partial agg.
Jumpgate Internals: Orchestrating Execution

6 Jumpgate schedules NCA execution as Staged Networked Pipelines:

- **Stage 1**
  - NCA1
  - NCA3
  - NCA4

  * NCA3 executes over two stages so that input to join is correctly ordered.

- **Stage 2**
  - NCA2
  - NCA3*
  - NCA4
  - Spark Workers

7 During execution, NCAs exchange data in **Network Tuple Format (NTF):**

**Stage 1**
- NCA1 → NCA3
  - NULL (2)
  - store_id (i64)
  - state (i64)

**Stage 2**
- NCA2 → NCA3
  - NULL (2)
  - price (f64)
  - store_id (i64)

- NCA3 → NCA4
  - NULL (2)
  - price (f64)
  - state (i64)

- NCA4 → Spark Workers
  - NULL (2)
  - sum(price) (f64)
  - state (i64)
Evaluation Questions

1. How easy/effective is it for a client to use Jumpgate?
2. How easy is it to add a new NCA?
3. What are the overheads of using Jumpgate?
4. When are the benefits to the client?
5. When can NCAs accelerate queries and why?

NCA Disclaimer

- Did not have access to hardware NCAs that could execute all operations
- Used software NCAs running on CPUs so we could study behaviour of pipelines of NCAs on many queries
- We look at when speedup does and doesn’t happen, and explain why.
Evaluation Methodology

- **Workload & Data:**
  - TPC-DS, a widely used SQL-based analytics benchmark that represents queries used in business decision making (run with spark-sql-perf)
  - Generated data in JSON and ORC format
  - Jumgagate executes jobs from TPC-DS queries **offloaded by Apache Spark**

- **General Setup:**
  - One machine runs Jumgagate and Spark’s Manager
  - Other machines run Spark Worker nodes and Jumgagate’s software-based NCAs
  - Input stored locally on each machine, on 1.6TB NVMe SSD for high storage throughput.
  - Spark and Jumgagate given same resources.
Evaluation: Client and NCA Integration

- How easy/effective is it for a client to use Jumpgate?

- Integrated Jumpgate into Apache Spark in 2,200 Lines of Code
  - ~2% relative to 100,000 lines of code for Spark SQL.
  - Users continue to write SQL, and Spark automatically offloads to Jumpgate.

- Spark offloads ~60% of all operations from TPC-DS, creating 853 jobs to study across all ~100 queries.

- NCAs added to Jumpgate in 200-600 LoC.

- Prior work only used 9 queries, and supported a final filtering operation.
- Jumpgate’s Dataflow API allowed offloading operations starting at the scan from storage, which is where the bulk of the data processing lies.
Evaluation: Jumpgate Overhead

- What are the overheads of using Jumpgate?
  - Test: ran all of TPC-DS with practically no data.

- High startup overhead: ~3.6s up to 6 seconds.
  - Due to compiling software NCAs per-query and deploying with SSH.
  - Mitigate by not offloading short jobs that process little data. (~100 LoC)

- Low execution overhead: 13ms - 70ms for all jobs to signal NCAs to change stages.
  - Stays out of the way of fast NCAs!

- Spark takes 11ms - 950ms for the same test.
Evaluation: Query Execution
Performance bottlenecked on parsing ORC data into NTF:

Client received data reduced by 1,500,000x! -> Less work for the client to do.
Q4: What are the benefits to the client?

**Client data reduction -> Less work to do.**

94% TPC-DS queries see a reduction of materialized data.
50% of queries see reduction >22x

Q5: When can NCAs accelerate queries and why?

**NCAs operate on network data.**

Overall network data volume is on-par with what Spark materializes in memory. Query speedup will happen when data is processed more quickly by NCAs.

Our evaluation found that format conversion was the common bottleneck and reason for good and bad performance.

In the paper: full analysis of TPC-DS and programmable switch example.
Thanks for listening!

Summary of Contributions:

→ New architecture: Jumpgate is the first NPaaS system that shows how to:
  1. Integrate with existing analytics systems.
  2. Orchestrate execution of NCAs + client with a novel execution paradigm.
  3. Provide simple interfaces to add new and diverse NCA implementations.

→ New insights from the evaluation:
  ✦ Using NPaaS can reduce data transmitted to a client by orders of magnitude.
  ✦ Using NCAs trades materializing data in memory for writing it to the network.
  ✦ Accelerating storage and network formats will be key to achieving speed-ups.
  ✦ See paper for more details!

Questions?