Peraton | LABS

FLEET: Dynamically Configurable Network Topologies for Legion

Fred Douglis fdouglis@peratonlabs.com

Eric van den Berg evandenberg@peratonlabs.com

Distribution "A" (Approved for Public Release, Distribution Unlimited). This research was developed with funding from the Defense Advanced Research Projects Agency (DARPA) .The views, opinions and/or findings expressed are those of the author and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government.

SLAC

Acknowledgments

SLAC

- Prof. Alex Aiken
- Xi Luo
- Seema Mirchandaney

COLUMBIA UNIVERSITY
IN THE CITY OF NEW YORK

- Prof. Keren Bergman
- Prof. Mingoo Seok
- Madeleine Glick
- Maarten H.N. Hattink
- Richard Dai
- James Thomas Robinson
- Mao Li
- Daniel Jang

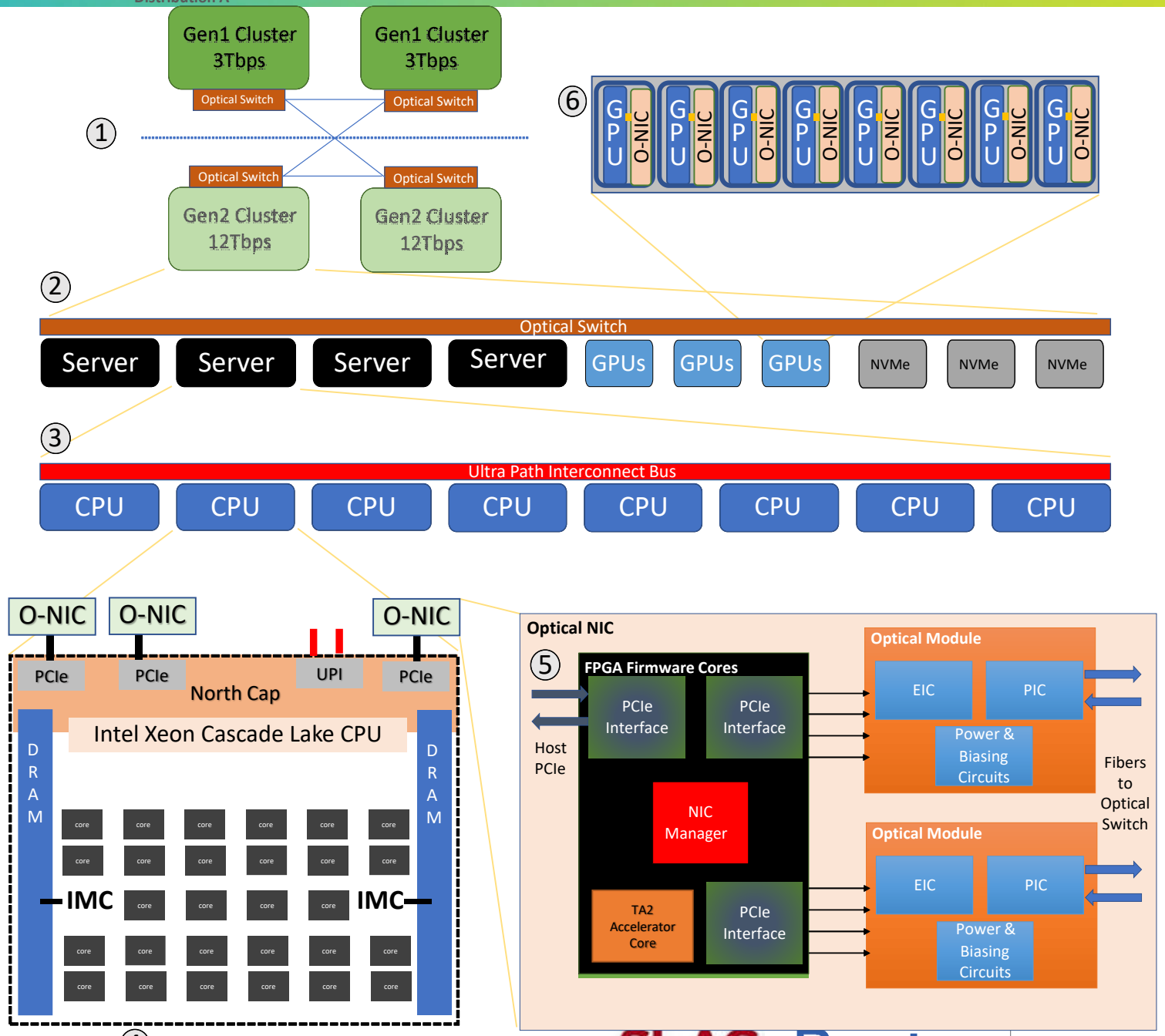
SLAC Peraton LABS

Agenda

- FLEET Introduction
- Legion Integration
- NUMA-level routing
- FLEET Planner
- HPC Simulation

FLEET Introduction

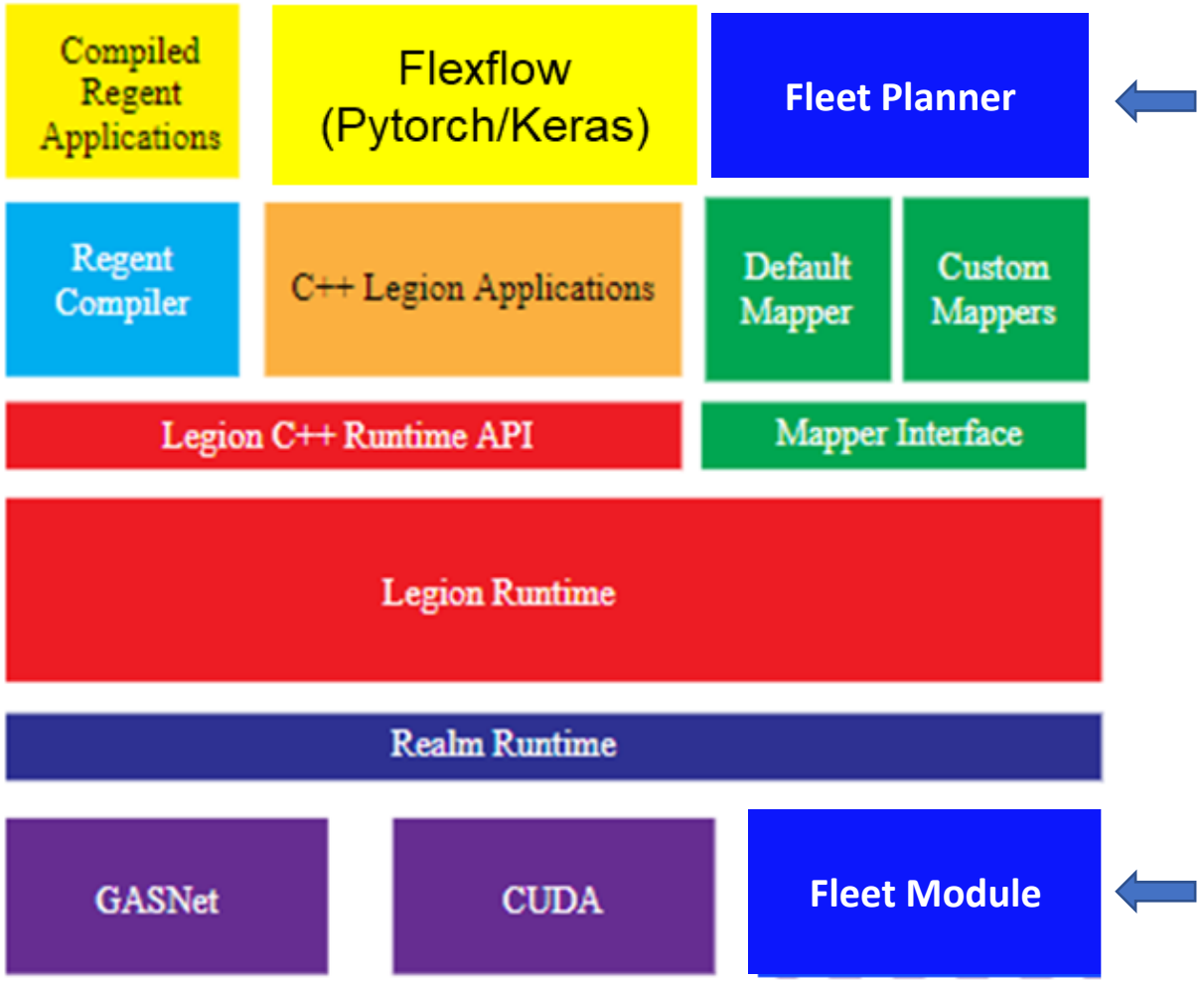
- Part of the DARPA FastNICs program
  - Goal: demonstrate capability to support 10Tbps networking, for acceleration of applications such as DNN training
- Creation of Optical Network Interface Cards (O-NICs) for 10 Tbps networking
  - Multi-blade supermicros allow population of >20 O-NICs
  - Optical switch to allow dynamic optical connectivity
- Creation of software allowing programs to take advantage of hardware
  - Plan location of data, program execution threads, and optical connections
  - Legion for program orchestration



FLEET Software

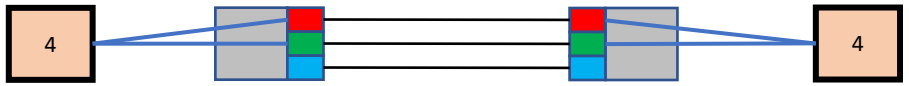
- **Legion, plus ….**
- **Fleet Planner**: Allocates parts of the app to nodes and decides the network configuration
- **Fleet Module**: Data movement library within Realm to interface with FLEET hardware

System Architecture

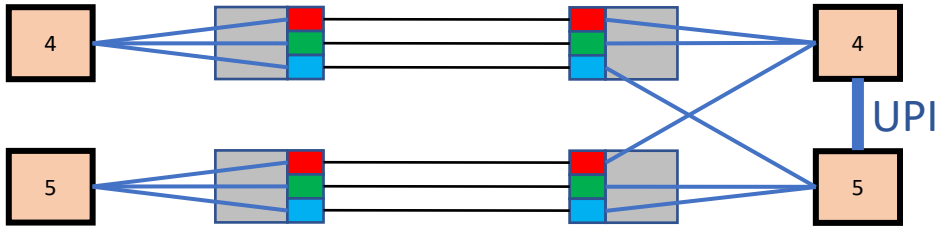


Much Depends on Hardware

- Supermicros – initial focus of FLEET and primary scalability target
  - Many blades, with UPI backplane that can connect NUMA nodes in 1 or sometimes 2 hops
  - Each blade can support 2-3 PCIe cards: 21 O-NICs maximum per system. PCIe root complex supports 100Gbps per card ➔ 2.1 Tbps aggregate throughput
  - Writing *into* memory from multiple cards tends to be limited (170Gbps for 3 cards)



  - We can improve performance by writing simultaneously from PCIe and UPI (535 Gbps over two nodes)



Much Depends on Hardware

- Lambda Labs GPU Servers – added specifically for DNN users
  - Two blades plus 8 GPUs & NVLink
  - O-NICs show asymmetric performance: each blade can transmit at 100Gbps but receive only at 50Gbps
  - Appears to be PCIe root complex issues, still under investigation

FLEET + Legion

FLEET Module

- FLEET runs as a module, in place of GASNet or other means of connectivity
  - It might be desirable to support *multiple* networks in parallel and use FLEET for custom hardware while using GASNet for other interconnections
- O-NICs connect NUMA nodes
  - One O-NIC can have up to four *engines*, each with an optical connection.
  - These can go to the same node or different ones
  - PCIe limits aggregate throughput across O-NIC
  - An O-NIC on one NUMA node can still directly access memory in another node (e.g. over UPI)
- Polatis switch allows for dynamic reconfiguration of topology
  - Can specify connections prior to execution
  - O-NIC configuration file informs FLEET module which O-NICs connect which ranks and which NUMA nodes within each rank
  - Ultimate goal is to also dynamically change topology *during* execution

FLEET Module

- O-NICs communicate via *messages* or *RDMAs*
  - Messages are fixed 8K including overhead
  - Anything larger is sent via RDMA – hardware supports PUT or GET operations
- Fast path for data that already resides in pinned FLEET memory includes fragmentation for parallelism
  - A configuration parameter specifies the size into which large transfers will be fragmented
    - FLEET supports RDMAs of GBs of data, but sending that over a single engine is limited to about 60Gbps
    - If there are N engines that can efficiently transfer the data, can send 1/N over each engine
  - Anything no larger than the fragmentation unit goes in a single PUT; larger transfers are managed by the receiver, which does GETs in parallel
  - Anything needing to go through a staged send or receive area uses a single engine, which copies data gradually
- Biggest current issue: RDMAs that take up to 100x longer than "normal"
  - Affects all engines on a NUMA node simultaneously
  - Some NUMA nodes seem to *never* encounter, while others encounter frequently, and some sporadically

Additional Legion Support

- Changes to machine model
  - More configurable
  - Modeling NIC delay and congestion
  - Modeling fragmentation of large messages for parallel transfers
- Realm FPGA module
  - Using an FPGA as a coprocessor
  - In-flight data transformations
- Internode path selection based on bandwidth rather than hop minimization

All done at SLAC

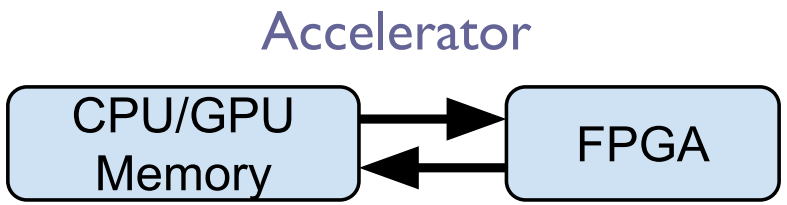
Effect of Machine Model Fidelity

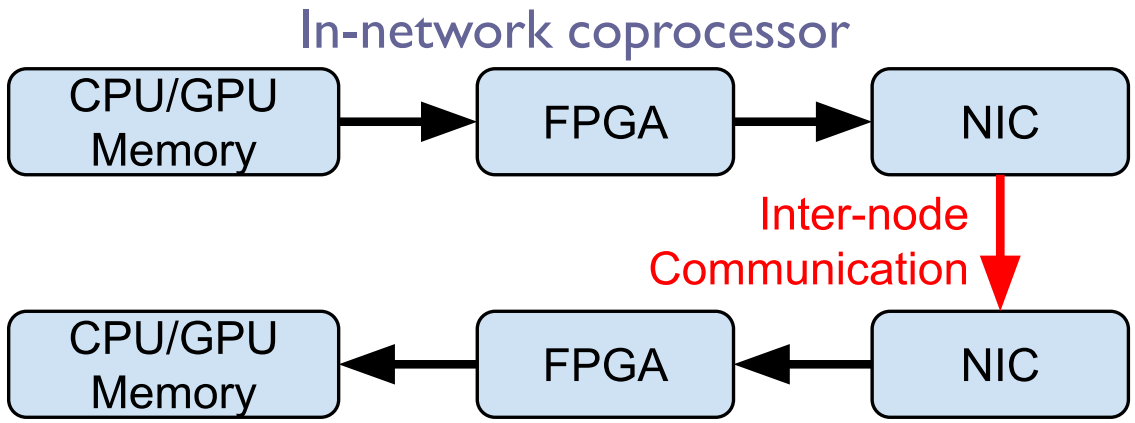
Improvements in the machine model result in the simulated times being closer to the actual time, with comparable simulation overhead

Merged into FlexFlow

FPGA Module in Realm

- Realm (low-level runtime of Legion) changes needed to support FPGA acceleration
  - New module added, allowing FlexFlow and Legion to compute on FPGAs

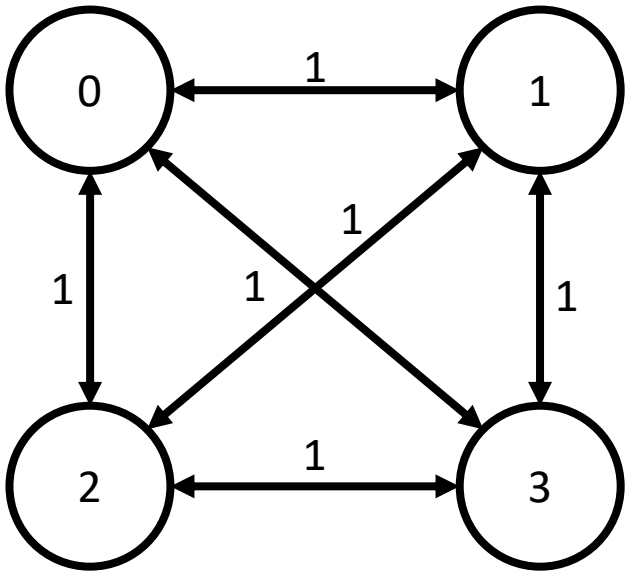




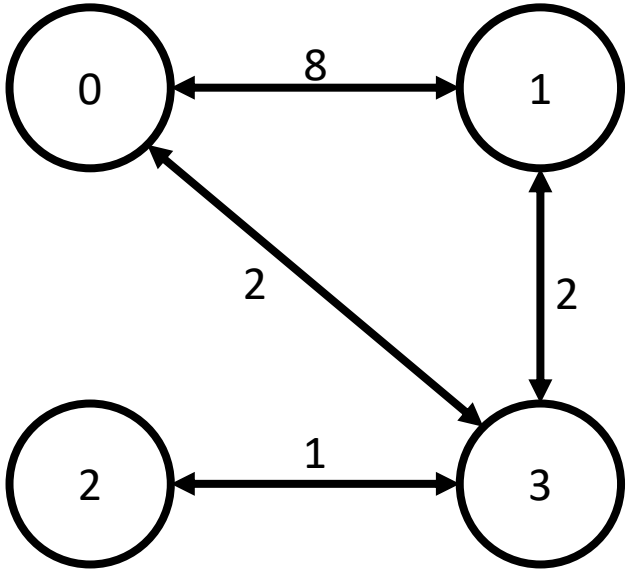
  - FPGA module implemented using the OpenCL extension of the Xilinx Runtime Library (XRT) with a low-level framework to also support FPGAs with or without XRT support
    - Can launch FPGA kernels on a new type of processor (FPGA_PROC) in Legion
    - Can automatically creating and managing physical instances of Legion regions on the FPGA device memory
    - Can transfer instances between FPGA device memory and system memory
- **Benefits:** take advantage of "free" in-band and energy efficient FPGA computation with low-latency access to data
- *Pull request pending*

Path Optimizations

- Legion connects all intermediate buffer memories and *assumes the inter-node links have the same bandwidth and latency*
- In FLEET, we can customize how the memories are linked and specify the bandwidth and latency for each link



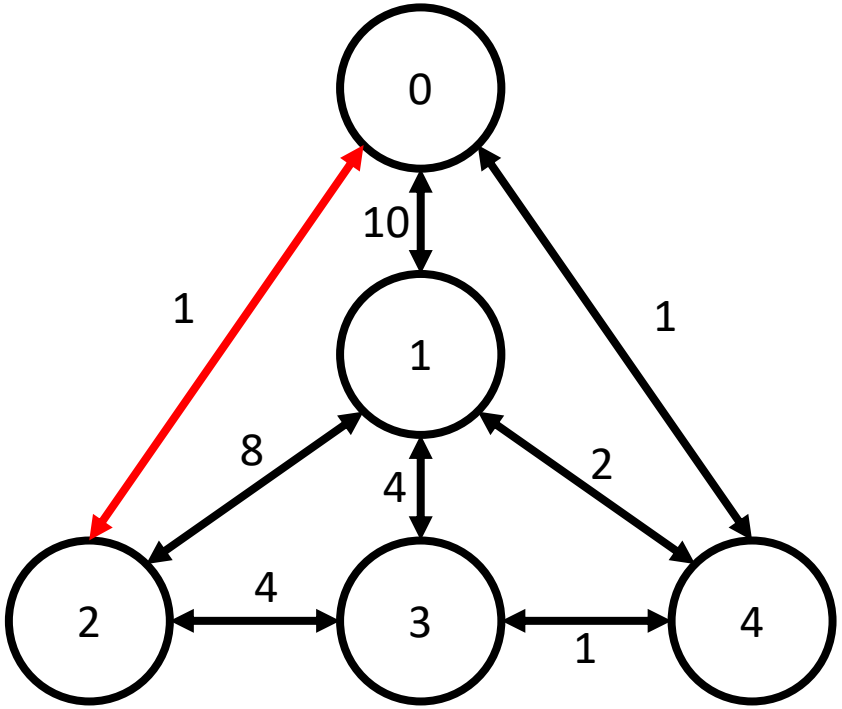
Old



New

Finding Paths Across Multiple Nodes

- Original Realm algorithms in Realm can only find direct paths



- For example, path 0->2 with bandwidth 1

- We replaced the path finding algorithm
- Performs Breadth First Search
  - Returns vector of paths & effective bandwidth

- For example:
  - Path 0→2 with bandwidth 1
  - **Path 0→1→2 with bandwidth 8**
  - Path 0→1→3→2 with bandwidth 4
  - Path 0→4→3→2 with bandwidth 1

- *Currently refining to integrate with FLEET-level NUMA-to-NUMA routing*

FLEET Planner

Analytical Sketches

Approach

- Automatically create and solve optimization problem
  - Model of computation, communication and memory loads
  - Solution using automatic differentiation and root-finding
- Find good values to use
  - # Nodes, # GPUs, batch size, etc
  - Trades off computation/communication/memory
- Balance parallelism type and parameters for hybrid model/data
- Next: Create 'sketches' for different objectives, constraints and applications
  - Energy efficiency, IO constraints, CPU-based apps

Example: Data Parallelism

Computation costs

$$\min_{p,b}\left\{\frac{D}{p}\sum_{l=1}^{G}(FW_l+BW_l)+\frac{D}{b\cdot p}\sum_{l=1}^{G}WU_l + 2\frac{D}{b\cdot p}(p-1)\left(\alpha+\frac{\sum_{l=1}^{G}|w_l|}{p}\delta\beta\right)\right\}$$

Such that:

$$\gamma\delta\sum_{l=1}^{G}(2b(|x_l|+|y_l|)+2|w_l|+|bi_l|)\le M$$

$$b\ge 1$$

$$1\le p\le p_{max}$$

- Variables/parameters: p=#gpus, b=batch/gpu, M=device memory, D=dataset size
- Model parameters: layer FW/BW compute/ Weight Update times, tensor sizes, weights, biases
- Network parameters: α=latency, β=1/throughput

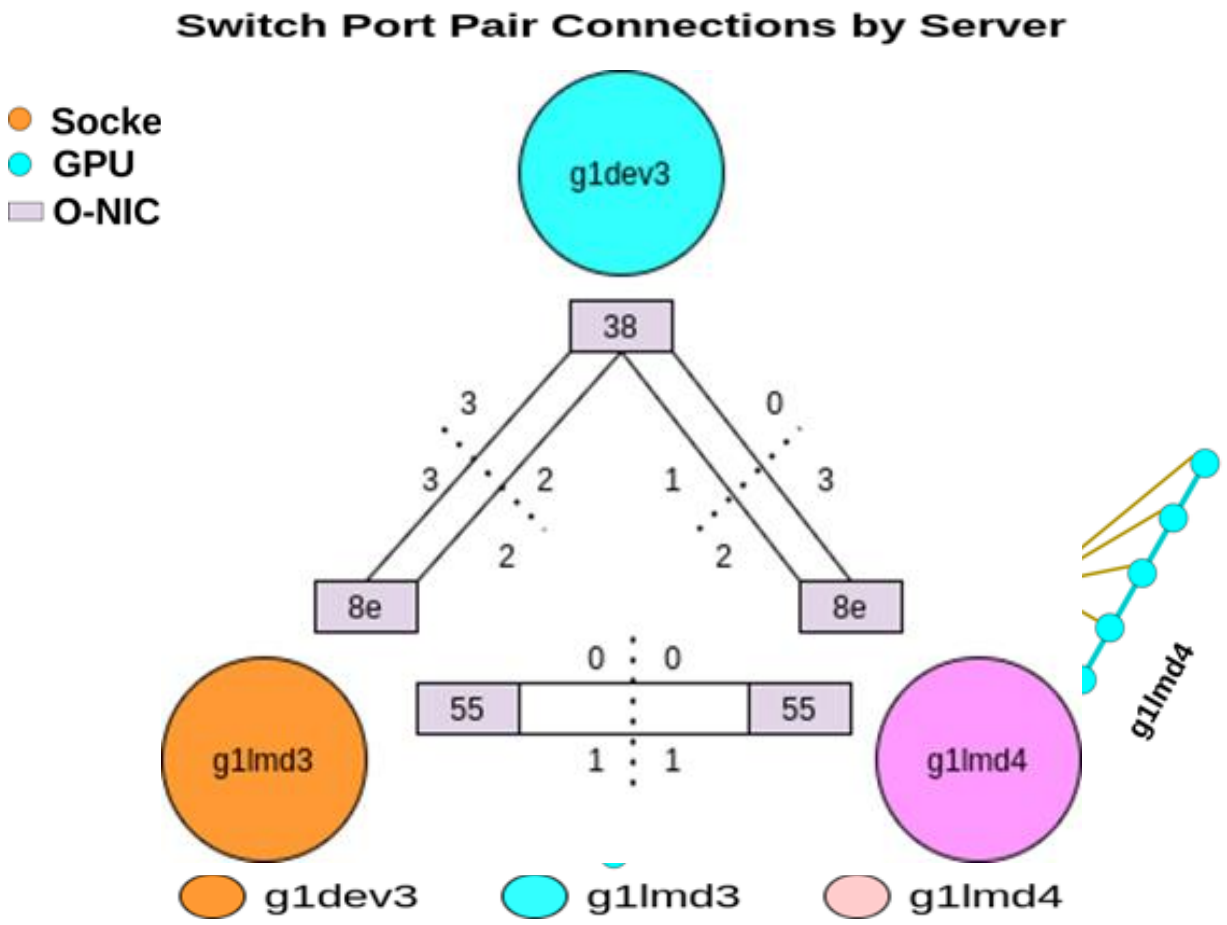
Communication costs

Memory constraint

SLAC Peraton LABS

Switch Configuration

- FLEET plans and implements switch configuration by FLEET O-NIC bandwidth steering algorithm
  - Ranks which exchange more data are connected by more optical links
  - NUMA effects may require alternate paths that enable parallel access combining O-NICs and UPI
- Currently implemented for Lambdas, under development for Supermicro NUMA



Generic Legion Applications

- Make FLEET Planner more useful for more applications
  - Planner dealt with FlexFlow models to date
  - Task graph easily extracted from FlexFlow
- Extend to allow planning for generic Legion applications
  - Step one: using Legion Prof and Legion Spy to extract task dependencies and performance profiles

Legend	
Index Task	Medium Slate Blue
Refinement Op	Royal Blue
Inter Close Op	Orange Red
Fence Op	Dark Orchid
Deletion Op	Maroon
Dependent Partition Op	Steel Blue
Pending Partition Op	Honeydew
Timing Op	Turquoise

Example: Legion Circuit app

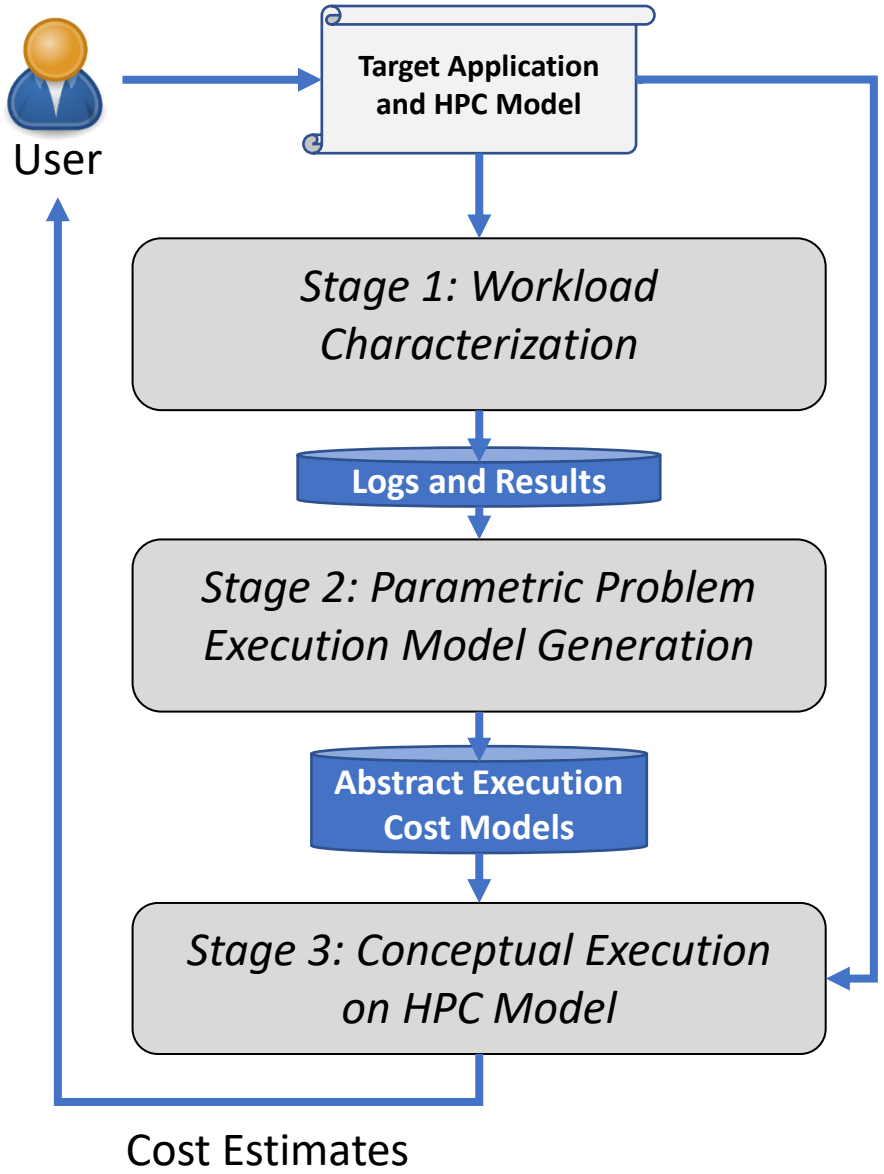
HPC Simulation

Background

- Early this year the FLEET project started a separate sub-effort specifically on doing better HPC simulation
  - Primary goal suggested by demonstrating how FLEET would do on future applications without actually building a 10Tbps system as originally planned
  - Intended for all flavors of HPC, and not just FLEET
- This is just a teaser to seed discussion – let us know if interested!

Conceptual Execution for Estimating Resources (CEER)

- Purpose: Create a simulation system to allow:
  - Simulation of the benefits of FastNICs hardware on customer workloads
  - Simulation of the costs of executing an algorithm on different proposed HPC hardware configurations
  - Evaluating of execution time, energy usage, resource utilization
  - Discovery of execution bottlenecks to improve algorithms or hardware configurations
- Approach:
  - Characterize existing workload on existing hardware
  - Extract model of how execution varies based on changes to hardware platform or application configuration
  - Predict resource costs and bottlenecks based on conceptually executing model of application on model of testbed



Status and Future Directions

Status

- With respect to optical transport, FLEET is demonstrating the programmability and flexibility afforded by NUMA-level optical connectivity
  - So far have focused on COTS 4-transceiver O-NICs, but look forward to trying Columbia's more efficient 2-transceiver O-NICs soon
- The PCIe behavior of the Supermicros continues to be an interesting challenge
  - Adding traffic to one link can increase performance of another
  - At the same time, there are still inexplicable delays
- Future steps
  - Integrating FPGA coprocessing
  - Continued improvements to FLEET planner and optimization
  - Expanding our set of Legion applications, possibly with a custom FLEET mapper

Parting Thoughts

- We are very interested in any government organizations, which use Legion, and might find these extensions relevant to their work
  - Commercial/academic too, but especially government

- We are hiring!

SLAC Peraton LABS