Designing Accelerators for Data Analytics: A Dynamically Scheduled Architecture

Marco Minutoli, Vito Giovanni Castellana, Antonino Tumeo,
Marco Lattuada and Fabrizio Ferrandi

Pacific Northwest NATIONAL LABORATORY

Proudly Operated by **Battelle** *Since 1965*

Introduction

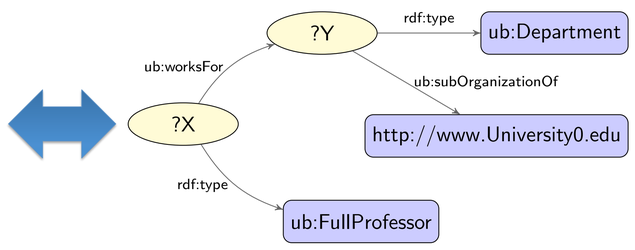
The Resource Description Framework (RDF) is a standard data model that represent data as triples (subject-predicate-object).

RDF databases:
- directly map to directed labeled graphs
- can be queried using SPARQL

SPARQL queries can be translated into Graph Pattern Matching methods that are intrinsically irregular in their behavior
- The execution is highly **data-dependent**
- At the same time they are characterized by high level of **data-parallelism**

Our Contributions:
- We analyze the behavior of LUBM queries to understand how the load unbalance between tasks affects the execution
- We propose an architecture template to tolerate the unbalancing between tasks that is suitable for adoption in High Level Synthesis Flows

```
SELECT ?x ?y
WHERE {
 ?y ub:subOrganizationOf
  <http://www.University0.edu> .
 ?y rdf:type ub:Department .
 ?x ub:worksFor ?y .
 ?x rdf:type ub:FullProfessor
}
```

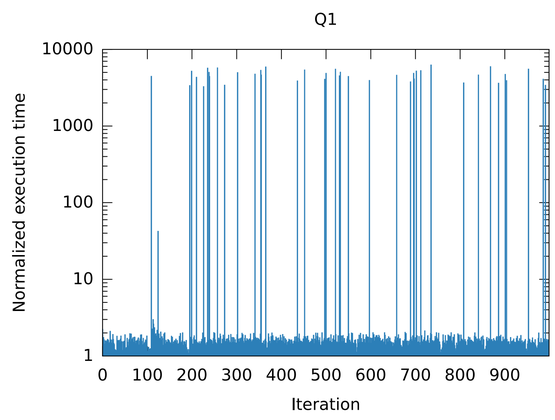


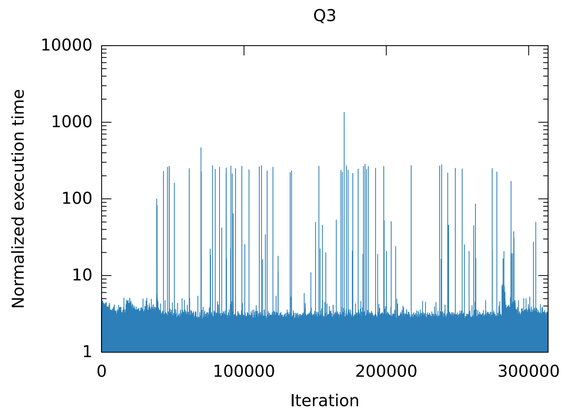
Query Execution Analysis

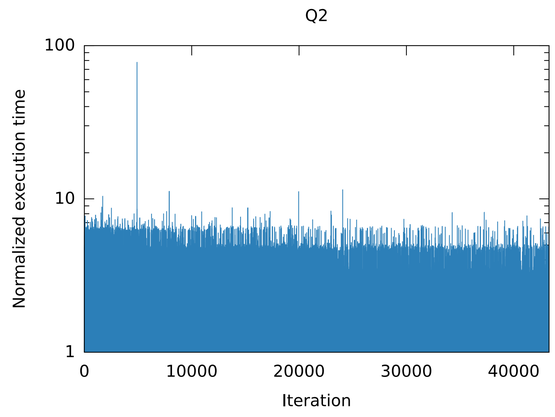
We consider 3 queries from LUBM (Q1-Q3):
- Input graph: LUBM-40 (5,309,056 triples)

The execution time of each outer loop iteration can vary of few order of magnitude.

Forking and joining tasks in groups can lead to resource under utilization when the workload between task is highly unbalanced (e.g., the group needs to wait for the slowest task: Q1 and Q3).



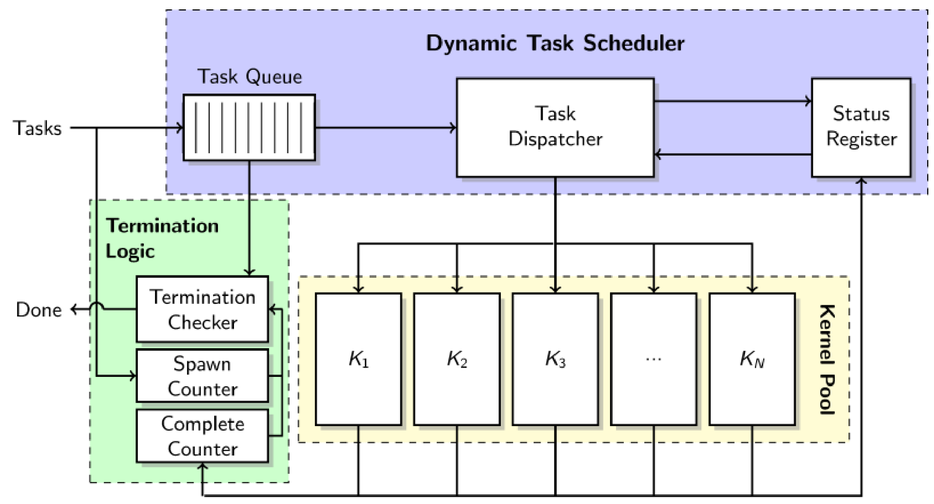




The Dynamic Task Scheduling Architecture

Dynamic Task Scheduler:
- New tasks are inserted in the **Task Queue**
- The **Status Register** keeps track of resource availability
- When there are available resources, the **Task Dispatcher** pops a task from the queue and start its execution

Kernels in the Pool:
- are interfaced to the memory using a Hierarchical Memory Controller Interface supporting atomic memory operations.
- notify the **Status Register** when they are ready to accept another task.



Termination Logic:
- The **Spawn Counter** records the number of spawned tasks (pushed into the queue)
- The **Complete Counter** registers the number of tasks consumed by the kernels
- The **Termination Checker** monitors the status of the Task Queue and the two counters to verify the Termination Condition and to assert the done signal accordingly

Termination Condition:
When the two counters are equal and the Task Queue is empty all the Tasks that have entered the queue have been consumed.

Experimental Evaluation

The architecture implementing the **Dynamic Task Scheduling** shows a **speed up** over the Serial implementation between **2.75** and **3.76**.

The **area overhead** is between **1.72-1.94 (LUTs)** and **1.62-1.95 (Slices)**.

The memory profiling shows that with 8 kernels the architecture is able to use 3 (out of 4) memory ports for the 80% of the computation time.

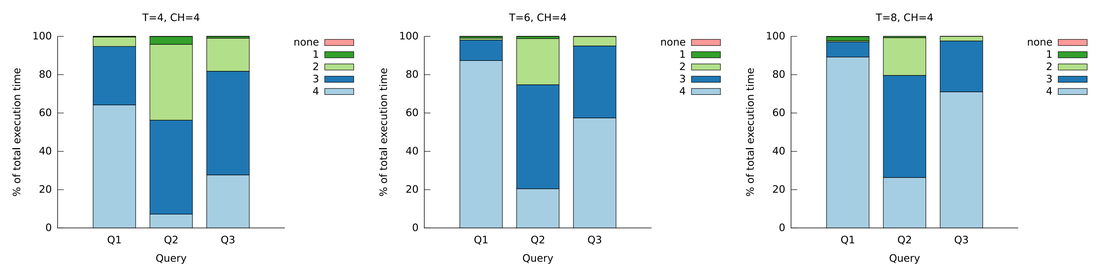
Increasing the number of kernels over 8 shows diminishing returns when the number of memory channels is fixed to 4.

	Serial				Dynamic Scheduler				Area Overhead over Serial		SpeedUp
	LUTs	Slices	Latency (#)	Max. Freq.	LUTs	Slices	Latency (#)	Max. Freq.	LUTs	Slices	
Q1	5,600	1,802	1,082,526,974	130.34MHz	10,844	3,503	287,527,463	113.60MHz	1.94	1.94	3.76
Q2	2,690	824	7,359,732	143.66MHz	4,636	1,335	2,672,295	132.87MHz	1.72	1.62	2.75
Q3	5,525	1,775	308,586,247	121.27MHz	10,664	3,467	95,154,310	116.92MHz	1.93	1.95	3.24

Comparison between a Serial Architecture and one implementing Dynamic Scheduling (T=4)

	T=6, CH=4				T=8, CH=4			
	LUTs	Slices	Latency	Max. Freq	LUTs	Slices	Latency	Max. Freq
Q1	15,305	4,822	268,093,088	111.58MHz	20,286	6,469	268,491,462	104.08MHz
Q2	6,507	1,942	2,355,699	113.45MHz	8,429	2,381	2,268,763	112.47MHz
Q3	15,259	4,943	83,327,993	106.19MHz	20,078	6,486	79,649,000	102.67MHz

Architecture implementing Dynamic Scheduling with T=6 and T=8



Memory profiling of the architecture implementing the Dynamic Scheduling

Marco Minutoli, Vito Giovanni Castellana, Antonino Tumeo
Pacific Northwest National Laboratory
P.O. Box 999, MS-IN: J4-30, Richland, WA 99352
marco.minutoli@pnnl.gov,
vitoGiovanni.castellana@pnnl.gov,
antonino.tumeo@pnnl.gov

Marco Lattuada, Fabrizio Ferrandi
Politecnico di Milano – DEIB
P.Za Leonardo da Vinci, 32 20133 Milano IT
marco.lattuada@polimi.it,
fabrizio.ferrandi@polimi.it

www.pnnl.gov