Computer Science Department

Smart Mapping of Scientific Workflows onto Heterogeneous Resources

Erdem G. YILMAZ, Julian M. KUNKEL

https://hps.vi4io.org

2021-07-02

Outline

Drivers of this research/What are the problems?

University of Reading

- Data sizes are approaching exa-byte scale
  - Impossible to load/work on entire data set
  - Its not feasible to move the data around
  - Paralellism and use of accelerated hardware is critical
- New and Accelerated hardware are consistently being integrated into HPCs and Data centers.
  - It is hard to adapt the existing scientific workflows onto new/modern hardware
  - Utilising such hardware in existing workflows, requires domain level knowledge

How can we address these problems?

Exa-byte scale data handling

- Code $\rightarrow$ Data solutions.
- Data streaming where possible
- Smart utilisation of accelerated devices
  - Detect/Declare underlying hardware resource capability
  - Resource(CPU, GPU, SSD, ...) aware mapping/scheduling of tasks
  - In-situ & In-transit processing techniques

Adapting to new execution environments

- Abstract the workflow generation from the execution
  - Such an abstraction will increase adaptability and scalability
  - Scientists should care less about optimisation and execution

A Simple Use Case

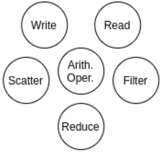
University of Reading

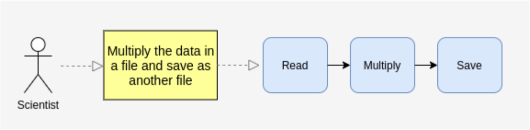
A scientist wants to read an N-Dimensional data from a file, multiply the content with a scalar and then save it back to another file.

- Three operators are involved: file read, multiply and file save.
- Lets assume we are given 3 nodes with different computational capabilities
  - Different number of CPU cores
  - Different storage types, HDD, SSD...
- What are the challenges around running this job/task(s) most efficiently on a given set of resources ?
  - Re-structring of code when the workflow is run on a different set of nodes
  - Domain knowledge required to benefit from the available accelerated devices
  - Which combination of computational node(s) will perform better?
  - How can we adapt to exa-scale data size?

What do we propose

- Scientist declares the workflow in terms of well-defined operators and creates the operator DAG

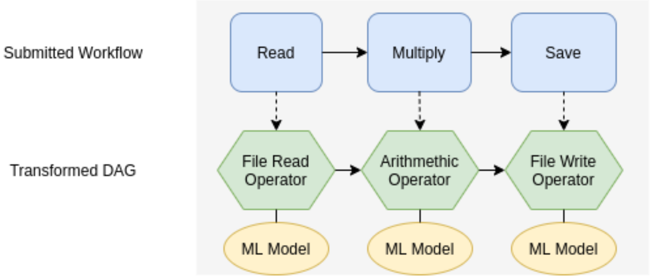




Operator defined DAG to Task DAG

University of Reading

- The framework will transform the Operator DAG into a Task DAG (Read,timesN,write)

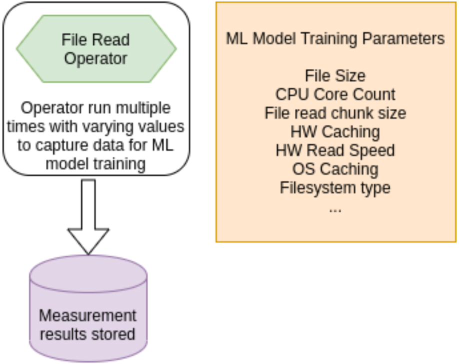


- Each operator has an associated ML model to be used in prediction of its makespan[1] when run on a node with an input size.

[1] The time it takes to complete a task, sometimes referred to as *end-to-end delay*

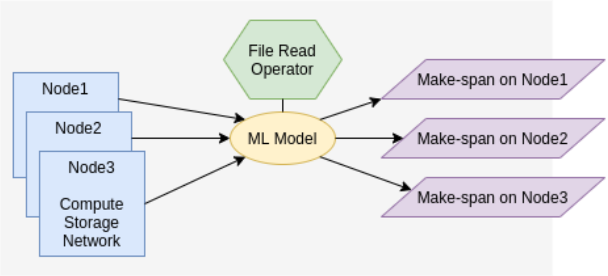
The ML model

- Develop a model (ML) for each operator using following features;
  - Varying Input size
  - HW capabilities(CPU, GPU, Storage, RAM, OS)



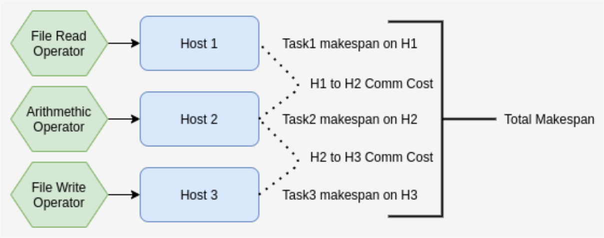
Makespan Prediction

- Given a set of resources, the trained ML model will be used per host with the following input features
  - Input size
  - Computational capabilities



Total Makespan

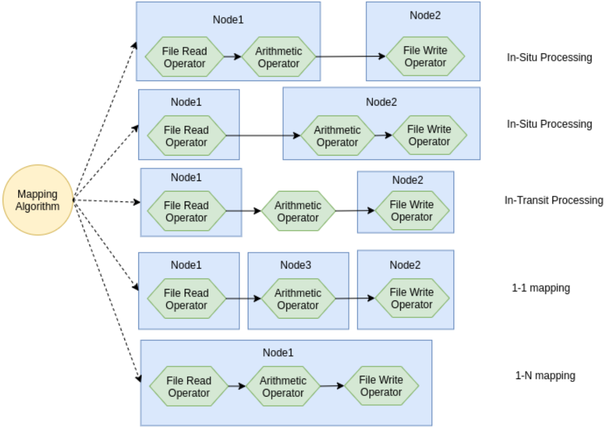
What does a mapping and its total makespan look like?



- Individual makespan predictions will be used in the overall cost model as part of a mapping decision.
- Due to accurate individual makespan prediction, Total makespan will be highly accurate

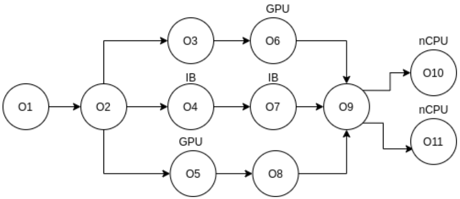
Mapping tasks to resources

Which task should be mapped to which node? Below is a possible combination of task to resource mappings.



A more representative use case

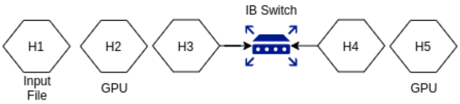
Lets assume we have 11 operators formed into a workflow.



- By default all operators have CPU based implementations
- O1, due input data file, is fixed to work on a certain node
- O4 and O7 have alternative implementations targeting InfiniBand Protocol
- O5 and O6 have alternative implementations targeting GPU
- O10 and O11 can run on multiple CPUs, data parallel tasks.

A more representative use case

Let’s further assume that we are given 5 hosts with various hardware resources (CPU, GPU, Storage, OS, Network)



- H2 and H5 have GPU
- H3 and H4 are connected via InfiniBand switch
  - Benefit from CPU off-loading (in-transit processing) via InfiniBand sw.

How to decide Task to Resource mapping?

- Does using GPU implementation provide speed ups?
- How about gathering all tasks to the best node to avoid comm cost?
- What will be the effect of resource sharing on same node?
- Offloading computation to external device, will that help?
- Longest task to fastest node? or Shortest tasks to fastest node?
- Is there a mapping that can out-perform our informed/educated decisions?

Scheduling/Mapping problem

- Classified as an optimisation problem, Scheduling/Mapping is NP-Hard[2].
- With 10 tasks over 5 hosts, there are 5^{10} permutations.
- Which mapping algorithms can we use?
  - Greedy : Under utilisation of resources
  - List based : Assumptions based on experience, biased
  - Dynamic : No time for in-depth analysis, missed opportunities
  - Static : Unable to adapt changing constraints during execution
  - Nature Inspired Algorithms
    - Genetic Algorithm : Can get stuck in a local optimum or take too long to converge.
    - Simulated Annealing : Cannot exploit full solution space
    - Particle Swarm Optimisation : can take too long in a big solution space
- A hybrid of the above algorithms will help with the shortcomings of the above algorithms

[2] Class of problems that cannot be solved deterministically in polynomial time wrt to input size

Genetic Algorithms

- Immitate how Evolution works, random mutations and survival of the fittest.
- Main challenge is to represent your problem in genetic domain.
  - Individuals : a candidate mapping between tasks and resources
  - Fitness : makespan of candidate mapping
  - Crossover, Mutations : exchanging part of the mapping between parents
  - Next generation : best fitness values are selected, with some random candidates
- No Free Lunch!
  - You may get stuck in a local optimum looking for the fittest individual
  - It might take too long to converge to the solution, make the solution unusable.

Creating individuals

- Specific tasks can run on specific hosts, which compacts the solution space with the help of educated assumptions.

T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	T11
H1	H1 H2 H3 H4 H5	H1 H2 H3 H4 H5	H3 H4	H3 H4	H3 H4	H3 H4	H1 H2 H3 H4 H5	H1 H2 H3 H4 H5	H1 H2 H3 H4 H5	H1 H2 H3 H4 H5

- Two individuals, candidate solutions for our mapping problem.

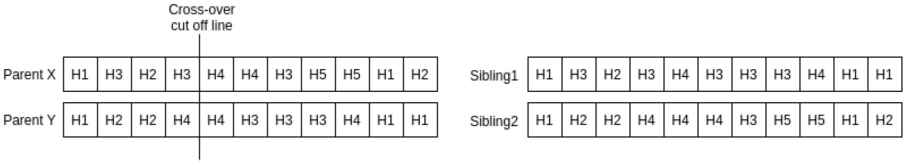
H1	H3	H2	H3	H4	H4	H3	H5	H5	H1	H2

H1	H2	H2	H4	H4	H3	H3	H3	H4	H1	H1

Crossover and Mutation

Once we select the parents, we can generate siblings,

- Crossover; mapping beyond cut off line is exchanged



- Mutation; randomly selected individual task-resource mappings will be altered on siblings

Particle Swarm Optimisation

University of Reading

- Starting from a random set of candidate solutions(the swarm)
- Using of a fitness/quality metric (cost of total makespan)
- Each particle(candidate solution) is moved to a new random position (new mapping)
- The best movement within the swarm is kept as a pivotal position to guide the swarm
- Successive iterations will converge the swarm to the optimised solution

The Proposed Solution

University of Reading

- Utilising accelerated hardware
  - GPU for data parallel operations.
  - Infiniband Protocol for computational offloading
- Pre-start cost model preperation : task makespan calculation with ML Model
- Dynamic mapping with sliding window and data streaming
  - Sliding Window : limit solution space, cater for # tasks > # nodes case.
  - Particle Swarm Optimisation : identify prominent paths to decrease solution space
  - Genetic Algorithm : near global optimum mapping

University of Reading

Where I am with my research?

- Literature survey has matured, updating it as I go along

	SA-BWS [51]	DOL [52]	ELPC [49]	PYLIGHT [15]	MSI [50]	Com-aware sched [28]	Resource Co-allocation [30]	XEFT [33]	UtlMin-Min [34]	Parallel GA [42]	CPGA & TDGA [43]	Modified SA [44]	ACO [47]	MEVSP [18]	DASK [13]	Flink [53]	Spark [14]	Pegasus	Smart Multi-task [20]	**Our Work**
Makespan Prediction																				
Random Data				+	+		+	+	+		+	+	+							
Historical Data														+						+
Stochastic/Cost Model	+													+						+
Machine Learning																			+	+
Performance Model		+											+	+					+	+
Simulation		+	+	+		+			+	+		+								
Supported Hardware Resources																				
Heterogeneous CPUs		+	+	+			+	+	+	+		+	+	+	+	+	+			+
GPUs								+							+				+	+
Memory	+	+												+	+	+	+			+
Storage	+													+		+	+			+
Network	+					+								+	+	+	+			+
InfiniBand																				+
Heuristic Scheduling/Mapping Algorithms																				
Min-Min							+		+											
Greedy			+	+	+		+								+		+			
HEFT								+												
Evolutionary Scheduling/Mapping Algorithms																				
Genetic Algorithm		+				+				+	+									+
Simulated Annealing												+								
Swarm/Colony Optimisation													+							
Supported Execution Environments																				
Virtual Machines				+											+	+	+			
Cloud				+											+	+	+			

What's next?

Need to investigate:

- Computational off-loading with InfiniBand supporting hardware
- Experiment with ML models to find the best fit for our makespan predictor
- Experiment with Moving window to cater the use case for limited number of hosts with too many tasks
- Experiment with Particle Swarm Optimisation to limit the solution space