THOUGHTS ON “WHITEBOARD”

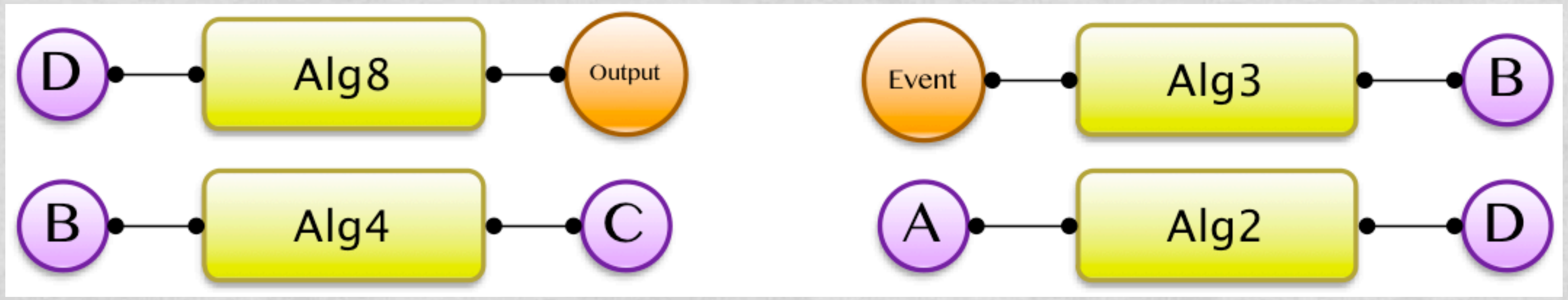
ideas on a concurrent physics framework

Riccardo-Maria BIANCHI *(CERN PH-ADT)*

Forum on Concurrent Programming Models and Frameworks - 14.03.2012

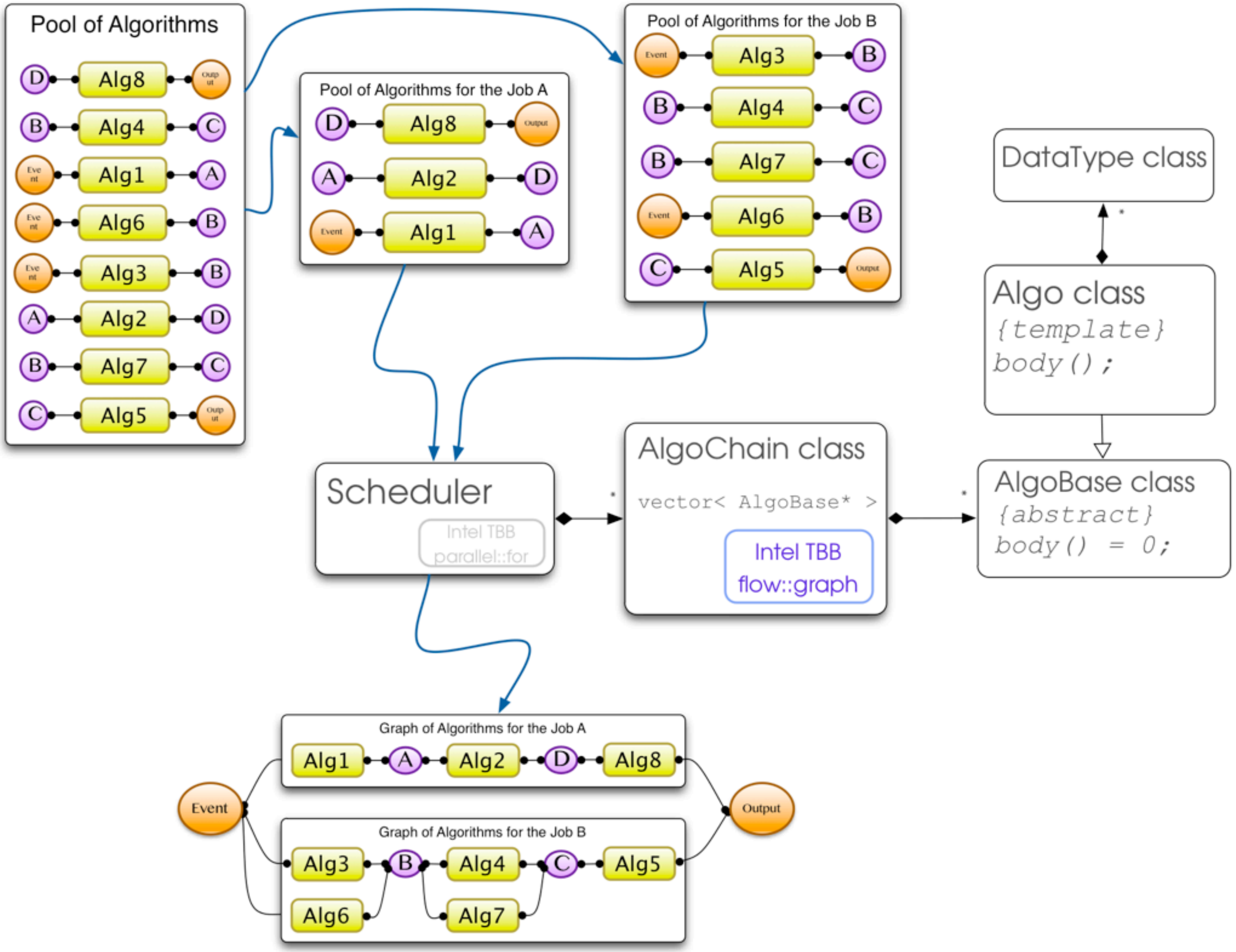
ALGORITHMS AS TASKS... LET'S TRY TBB!

- Our physics data processing mechanism is mainly based on chains of inter-dependent, but *independent*, algorithms.
- Our algorithms are *de-facto* "tasks", which can be seen as nodes of a "graph"... so let's try to implement a prototype with the Intel TBB *flow::graph!*
- Algorithms can have one or more IN and OUT data objects
- Algorithms can be seen as nodes featuring single/multi ports as input and output

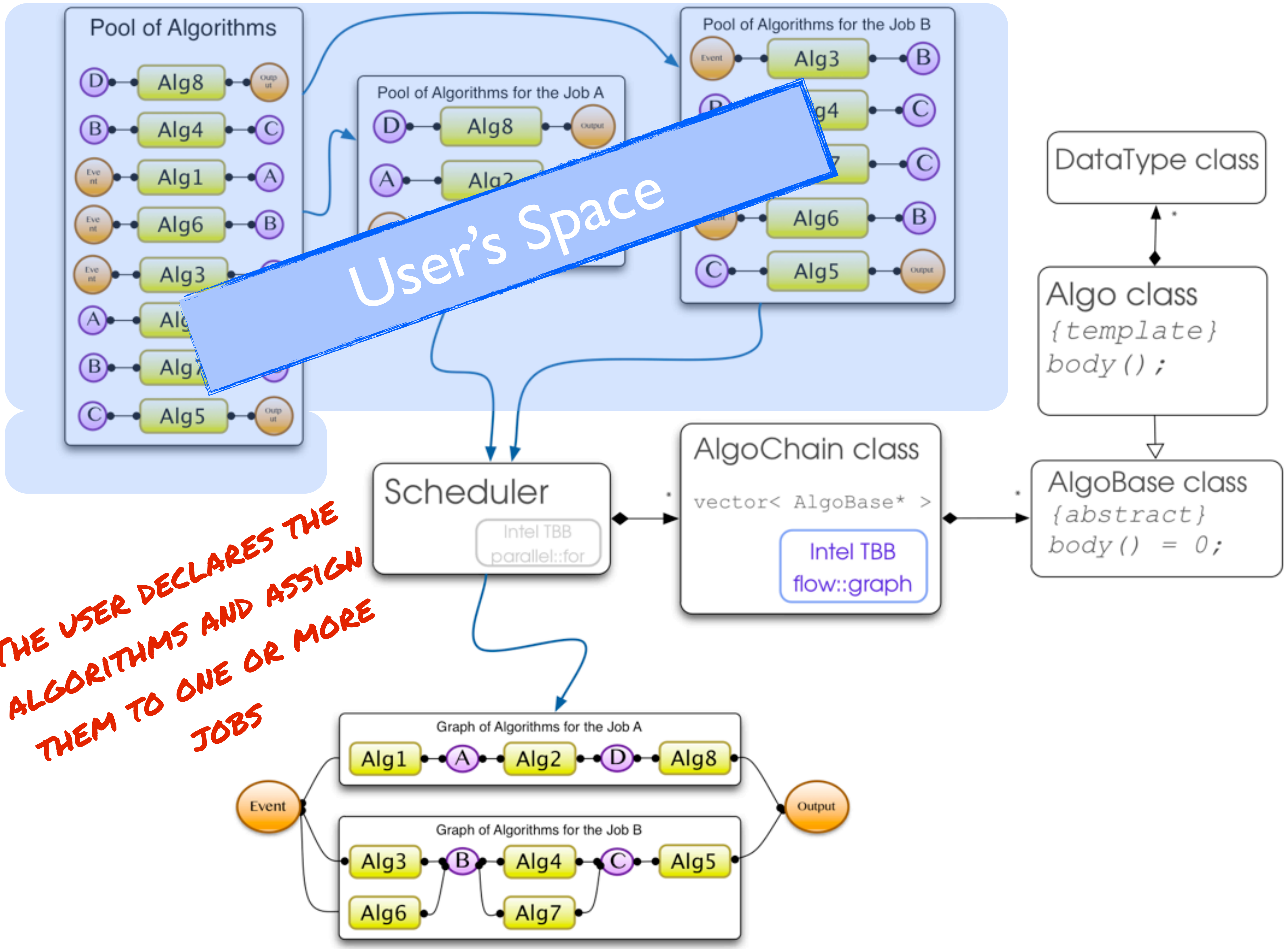


- Nodes are then interconnected by edges, according to the IN and OUT data types of the nodes

A FIRST DEMONSTRATOR

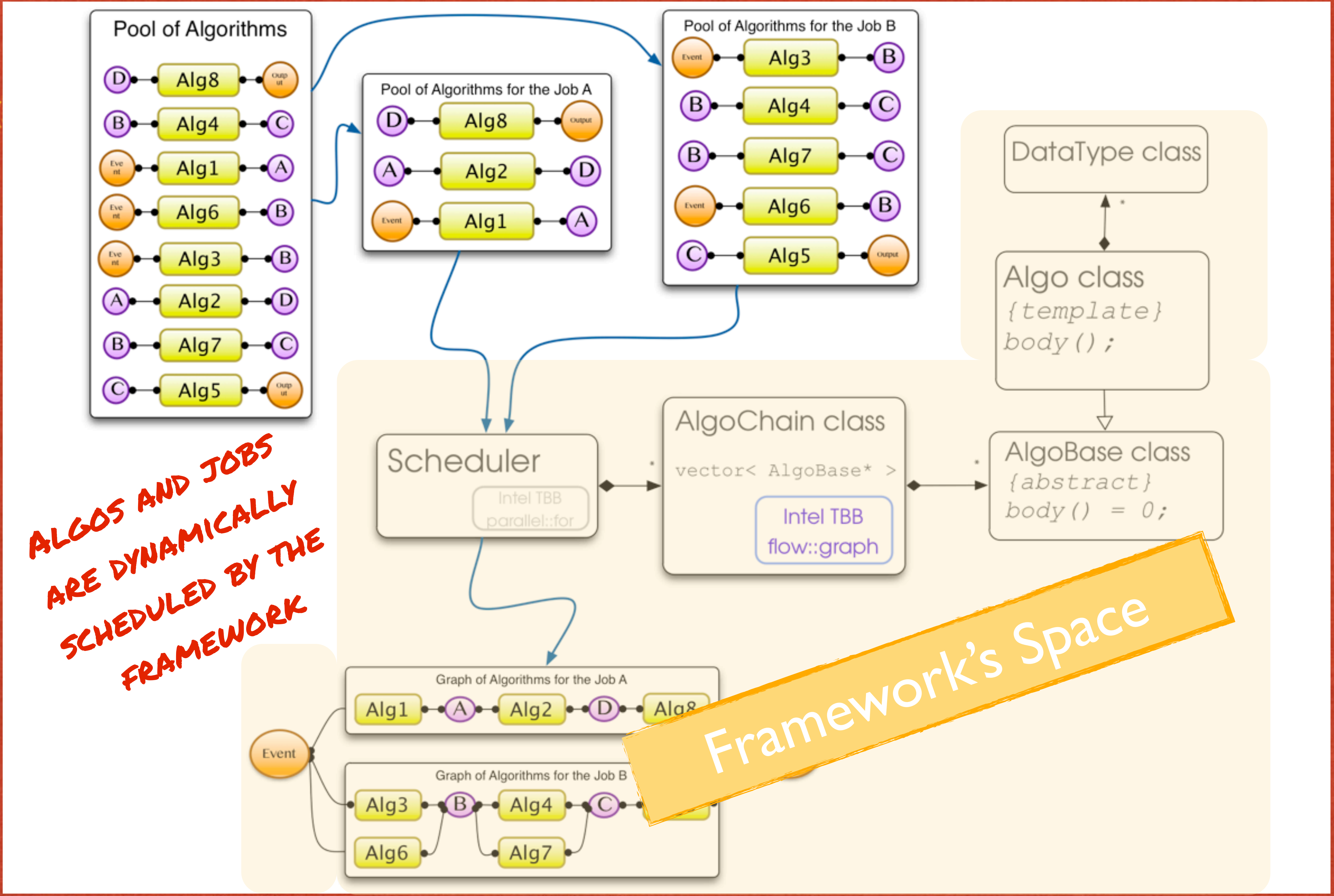


A FIRST DEMONSTRATOR



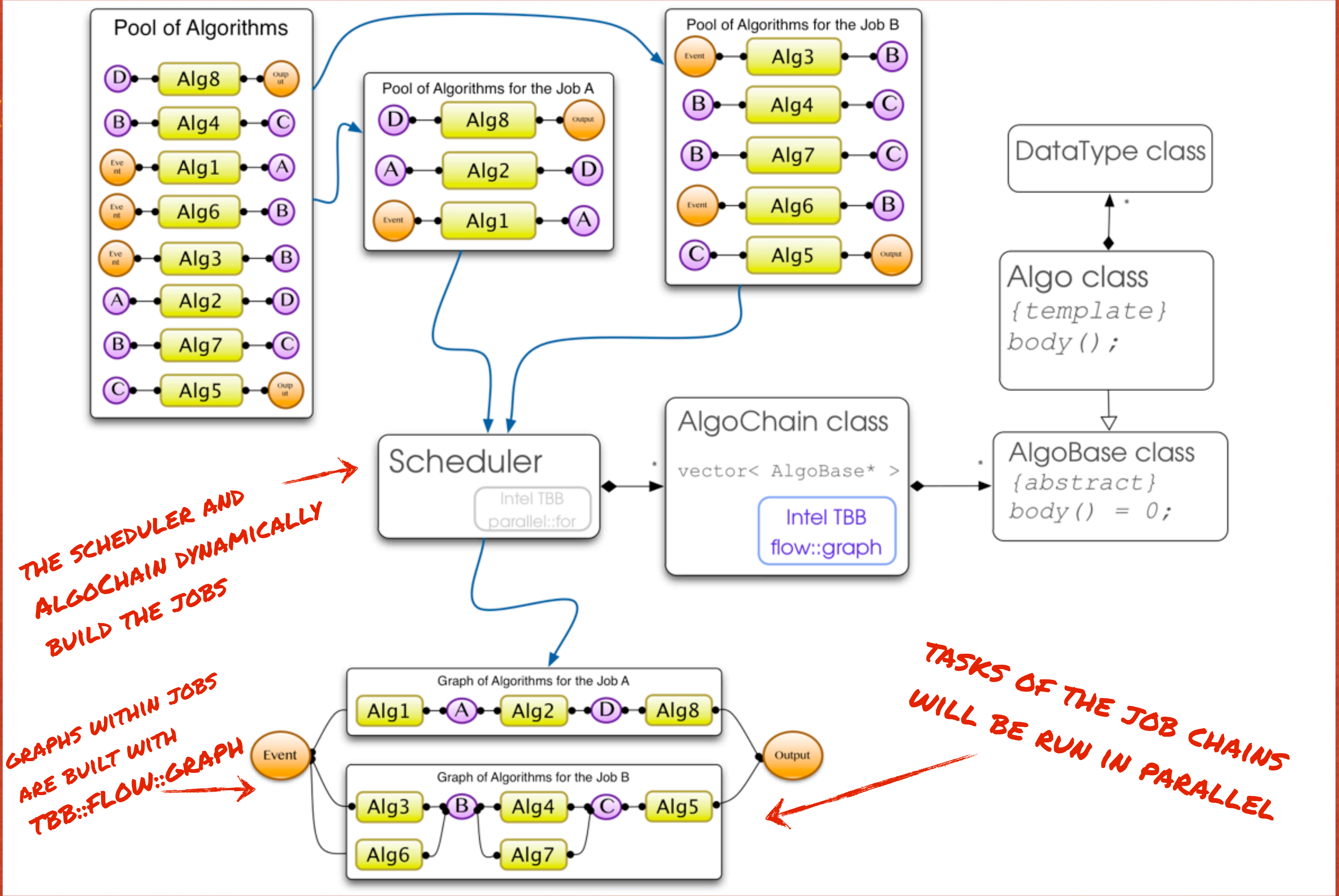
THE USER DECLARES THE ALGORITHMS AND ASSIGN THEM TO ONE OR MORE JOBS

A FIRST DEMONSTRATOR

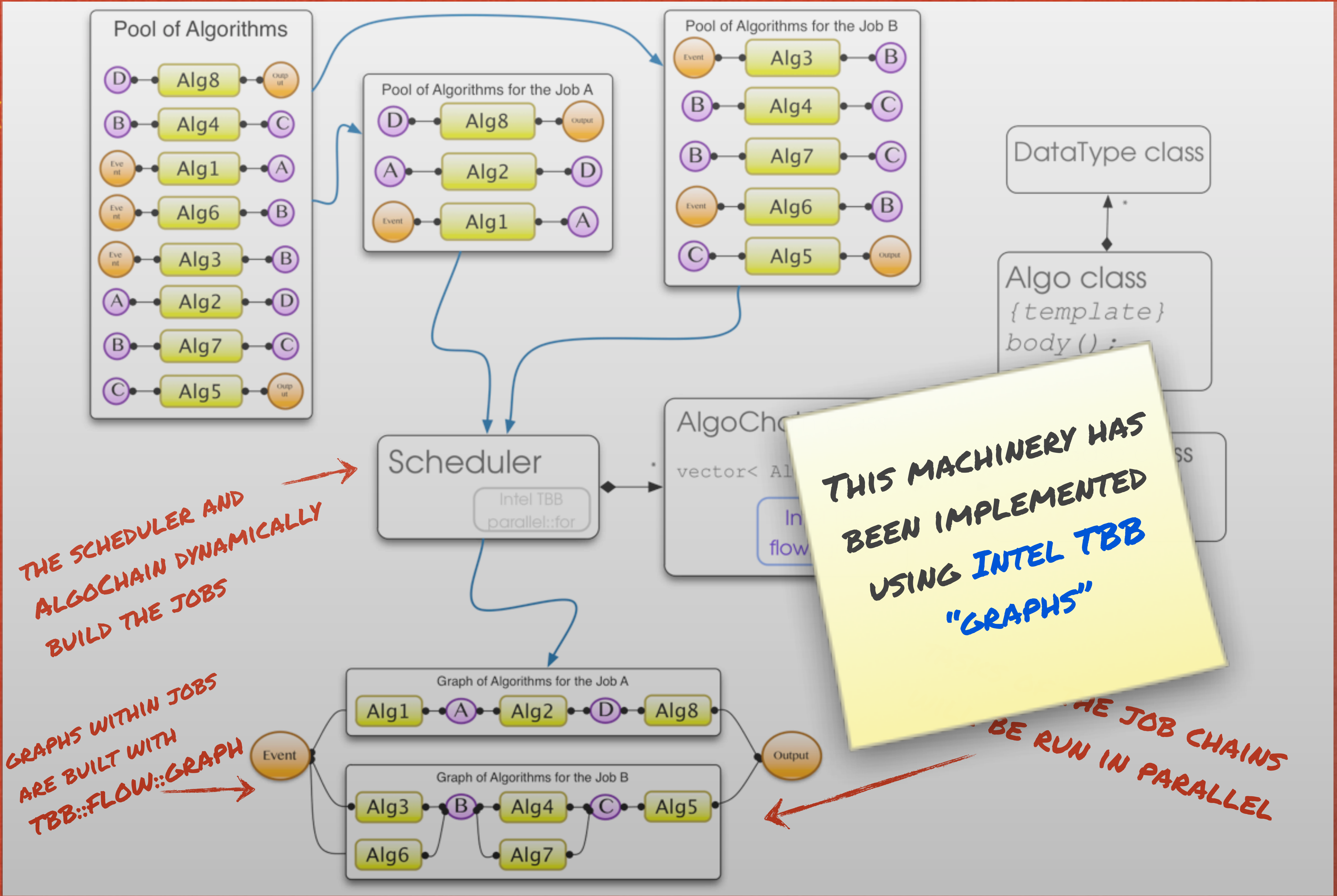


ALGOS AND JOBS ARE DYNAMICALLY SCHEDULED BY THE FRAMEWORK

A FIRST DEMONSTRATOR



A FIRST DEMONSTRATOR



THE SCHEDULER AND ALGOCHAIN DYNAMICALLY BUILD THE JOBS

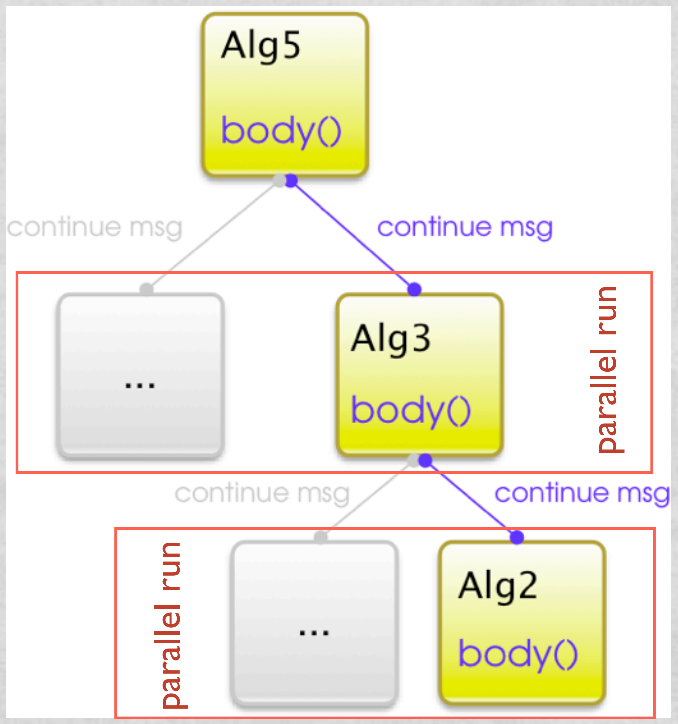
GRAPHS WITHIN JOBS ARE BUILT WITH TBB::FLOW::GRAPH

THIS MACHINERY HAS BEEN IMPLEMENTED USING INTEL TBB "GRAPHS"

THE JOB CHAINS BE RUN IN PARALLEL

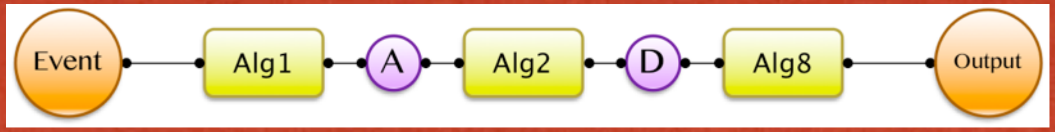
JOBS CREATED DYNAMICALLY, TASKS RUN IN PARALLEL

- *N* jobs are created, with *M* algorithms each
- Each AlgoChain inspects the IN and OUT data types of each algorithm, and it builds a `tbb::flow::graph`, connecting the algorithms in a producer/consumer way
- The jobs are then run, and algorithms are run in parallel when not dependent from the completion of predecessors
- each job (`AlgoChain` instance) does some work in `Setup()`, then it runs the scheduled algorithms
- When triggered by its predecessors, each Algorithm does some work in its `body()` function. When finished, it sends a message to all its successors
- For the time being all `Setup()` and `body()` have been implemented with a call to `Sleep(2)` to simulate some workload

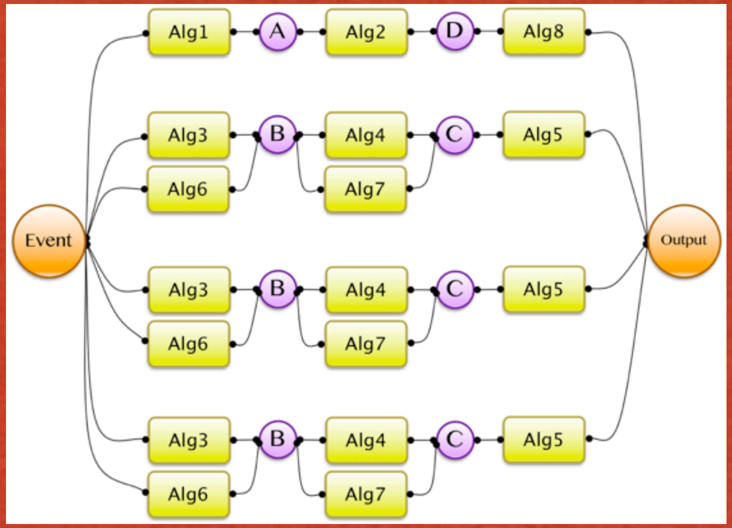


FIRST TESTS WITH THE DEMONSTRATOR

- To test the machinery 2 pools of algorithms are declared
- Then the Scheduler assigns them to a number of jobs (instances of the AlgoChain class)
- Each job dynamically builds its graphs, ending up with the 2 test configurations below

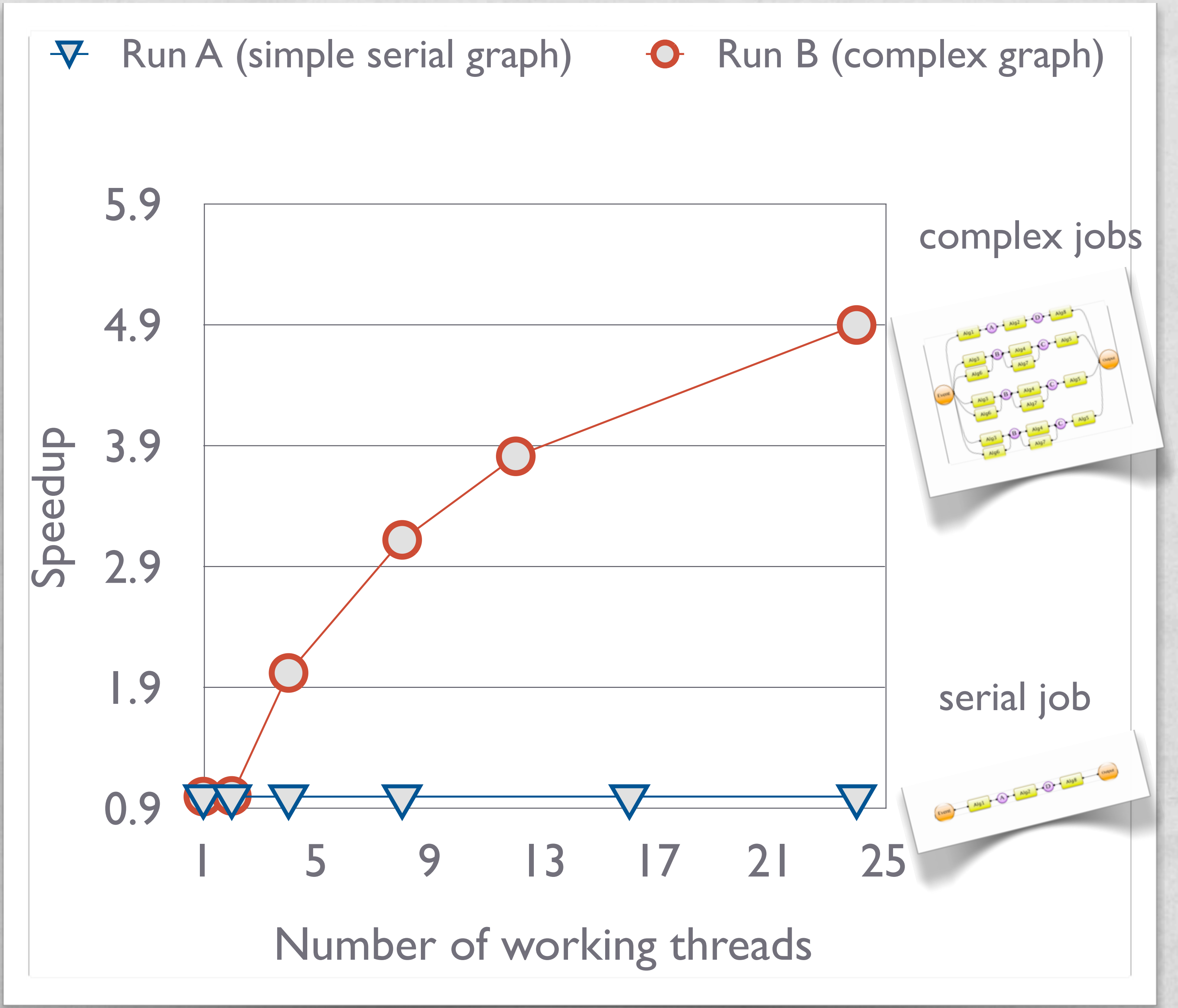


A single job with 3 algorithms connected in a serial way



the same serial job plus 3 jobs with 5 algorithms connected in a more complex pattern

COMPLEX JOBS CAN BE BOOSTED UP

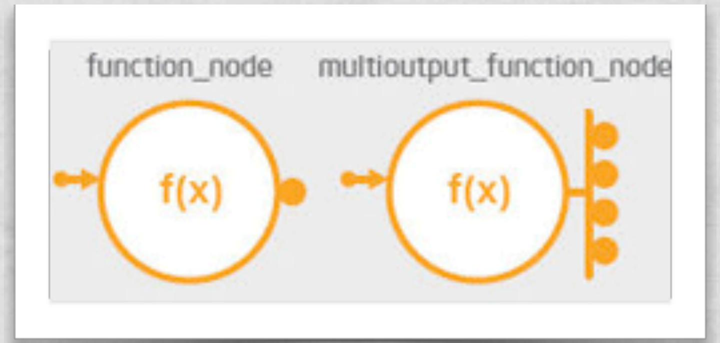


- Of course the *Run A* cannot profit from parallelization
- while *Run B* is boosted with a large speed-up factor
- But this only shows that this graph-based model works...

tested on a 2-cores MacBook using `tbb::task_scheduler_init` to set the number of working threads

MORE TO BE EXPLORED

- function_node and multifunction_node can be used for Algorithms: they can have one or multiple outputs. The concurrency level of those nodes can be set.



- Tuning options of the task-scheduler should be tested

- The TBB (as like as GCD) the non-preemptive task-scheduler is optimized to handle many non-blocking tasks in flight. Long-waiting tasks destroy the concurrency of those schedulers, loosing in performances.

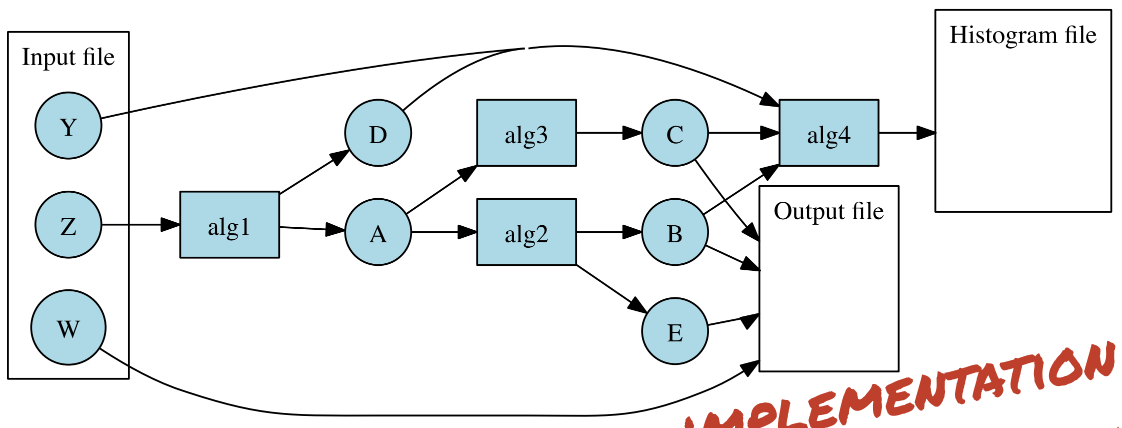
- Hence alternative mechanisms should be found and tested for long-waiting operations, like network or disk I/O.

STILL TO DO FOR A BETTER VERSION OF THE DEMONSTRATOR

- IMPLEMENTATION OF SOME MORE REALISTIC WORK IN THE ALGORITHM BODY, AND TEST OF PASSING REAL VALUES (FLOAT, INT, ROOT OBJECTS, ...)

- IMPLEMENTATION OF A BASIC PROTOTYPE OF A TRANSIENT DATA STORE

- PROTOTYPES OF PERSISTENCY SERVICES



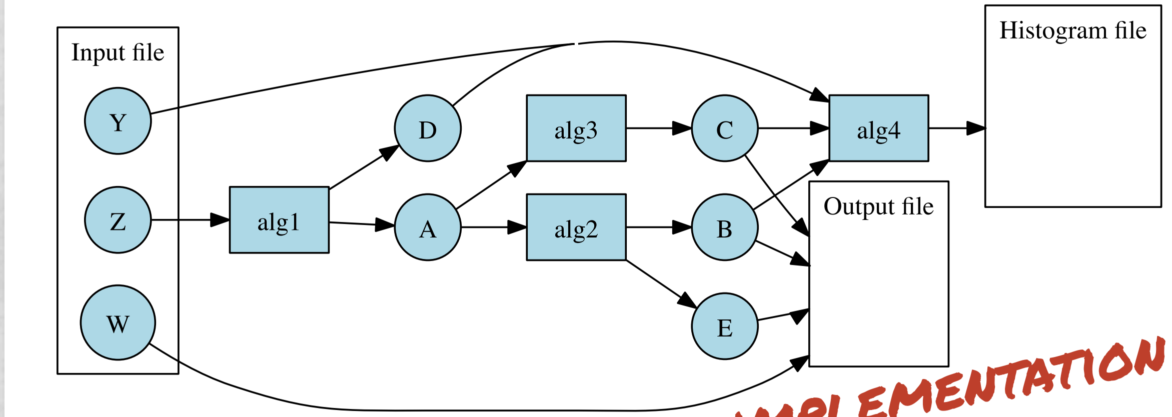
IMPLEMENTATION OF THE "STANDARD GRAPH" FROM M. PATERNO AND C.JONES

STILL TO DO FOR A BETTER VERSION OF THE DEMONSTRATOR

- IMPLEMENTATION OF SOME MORE REALISTIC WORK IN THE ALGORITHM BODY, AND TEST OF PASSING REAL VALUES (FLOAT, INT, ROOT OBJECTS, ...)

- IMPLEMENTATION OF A BASIC PROTOTYPE OF A TRANSIENT DATA STORE

- PROTOTYPES OF PERSISTENCY SERVICES



IMPLEMENTATION OF THE "STANDARD GRAPH" FROM M. PATERNO AND C.JONES

... SUGGESTIONS ARE VERY WELCOME!