Towards Plan-aware Resource Allocation in Serverless Query Processing

Malay Bag

Alekh Jindal

Hiren Patel

Resource Allocation Issue in Serverless Query Processing

- Hard to estimate resource requirement at compile time
- Resource requirement changes over execution period
- For long running analytical query, over-allocation leads to significant inefficiencies.

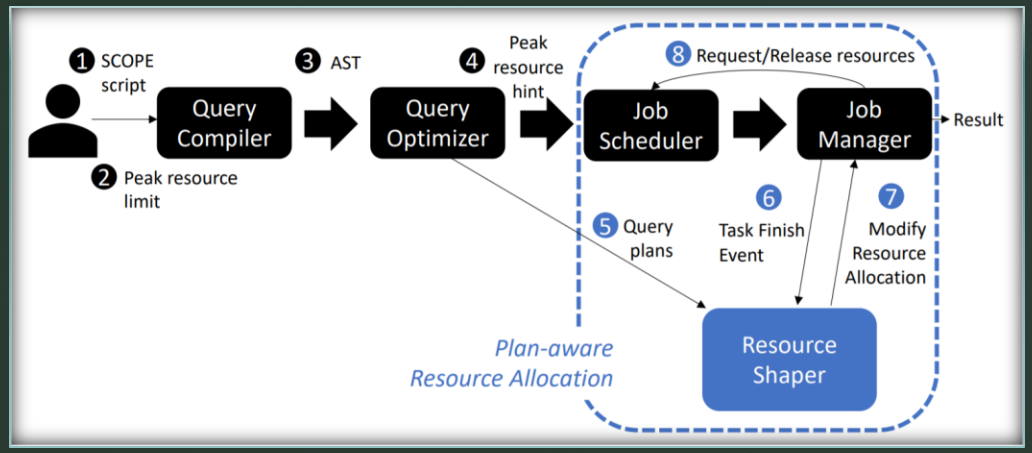
Prior Work

- SCOPE does not consider the query plan, instead treat the job as black box
- Allocate resource based on the past history and/or query plan (Morpheus, Ernest, Perforator)
- Dynamic re-allocation using expensive estimator based on previous run (Jockey)
- Find optimal resources for each operator during compile/optimize step (Raqo)

In summary prior approaches does not tune resource allocation to fine grained behavior of the query execution over time

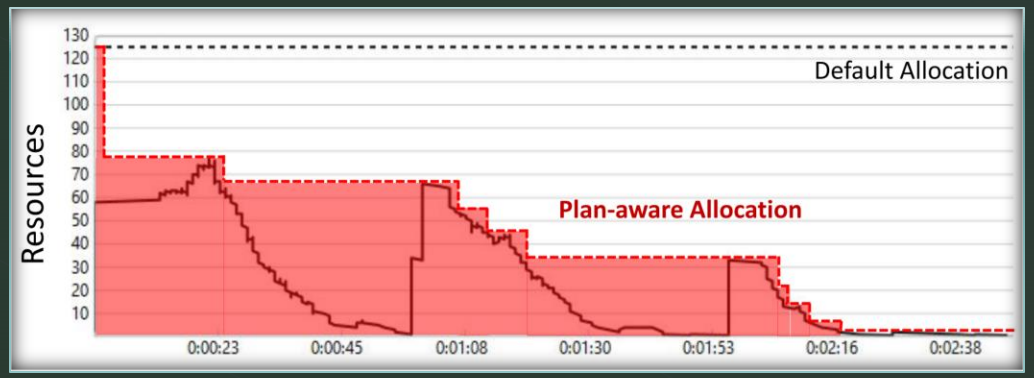
Plan-aware Resource Allocation

- Periodically invokes resource shaper to calculate new resource requirement.
- Resource shaper handles dynamic changes in the graph
- Calculates new requirement based on remaining part of the job graph



Plan-aware Resource Allocation

- At any point, if new requirement is less than current allocation, Job Manager updates Job Scheduler
- No performance impact, transparent to the user



Greedy Resource Shaper

Algorithm 1: `Resource Shaper`

Input : stage graph G, stage vertices V, current resources R, completion state W

Output : updated peak P

```
T = Treeify(G)
maxRemaining = Empty
foreach root ∈ T.roots do
    maxRemaining.Add(RemainingPeak(root, V, W))
if maxRemaining < R then
    GiveUpResources(R − maxRemaining)
```

Greedy Resource Shaper

Algorithm 2: `RemainingPeak`

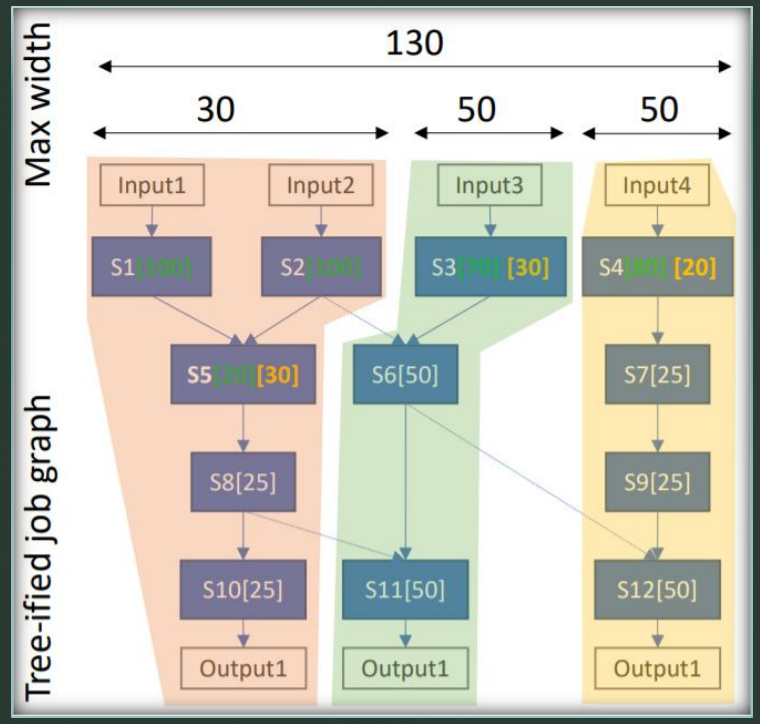
Input : root stage s, stage vertices V, completion state W
Output : updated peak P

if $W[s] \geq V[s]$ **then**
    return Empty;

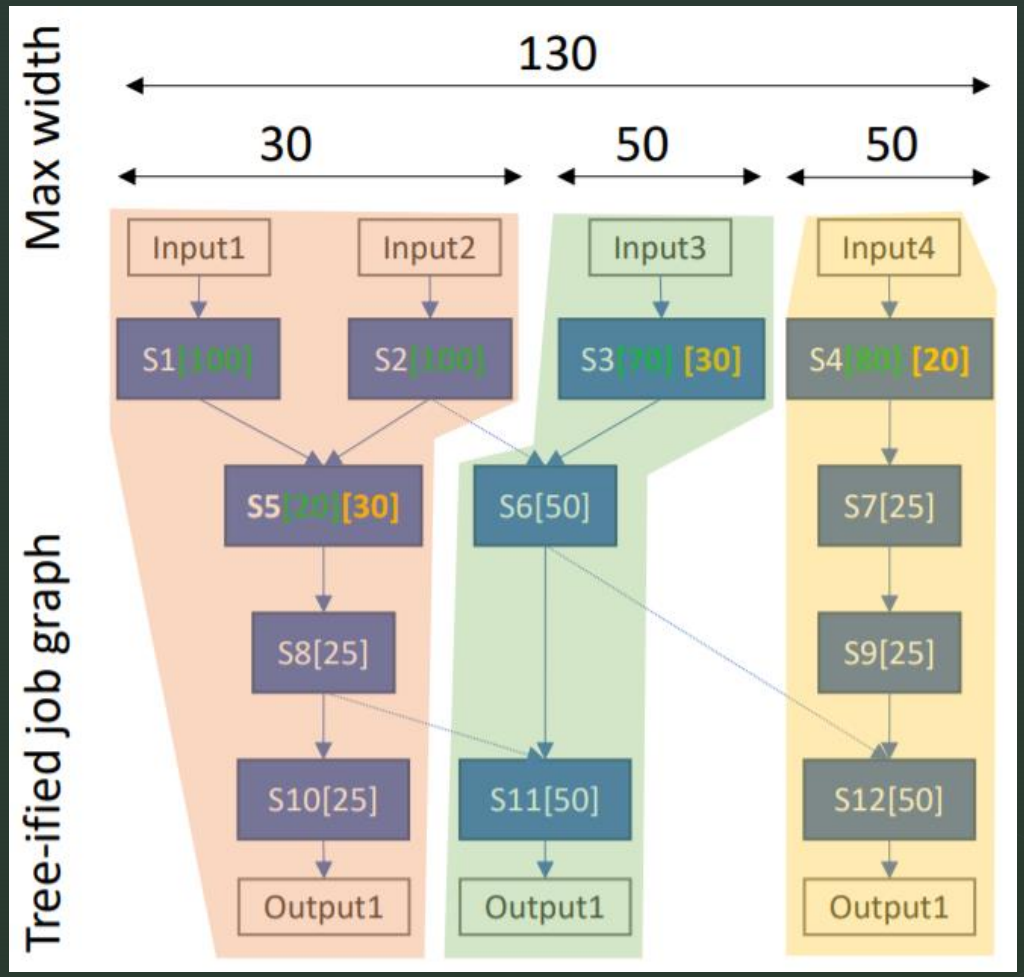
childResources = Empty;
foreach *child* $\in$ *s.ChildStages* **do**
    childResources.Add(RemainingPeak(child, V, W));
return `Max(Resources(s),` *childResources*`)`

Tree–ification

- Convert DAG to a tree by removing one of the output edges of spool operator (which has multiple consumers)
- Remove edges to the consumer with maximum in-degree, until the DAG become a tree
- Break ties with random selection
- Output is an inverted tree

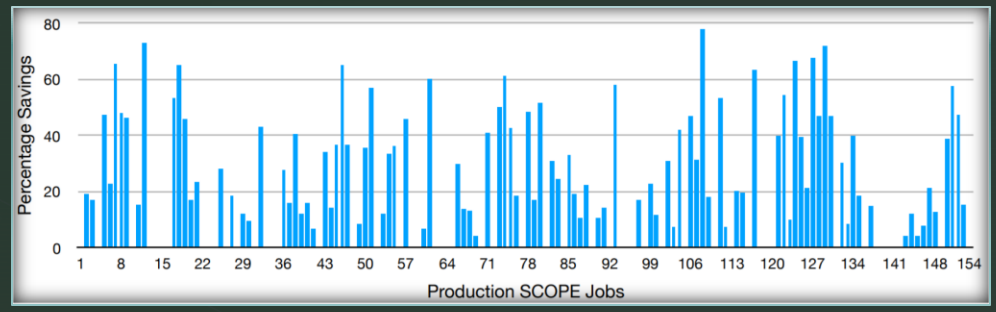


Max Vertex Cut example



Evaluation

- Run 154 jobs on a virtual cluster
- Overall 8.3% savings of cumulative resource usage
- Potentially there are 8-19% saving opportunity in our 5 production clusters, which would save us tens of millions of dollars in operating cost



Thank you!

Please contact {malayb, alekh.jindal, hirenp} @microsoft.com for any questions.