Locality-Aware Batch Scheduling of Jobs Sharing Input Files

Uppmax meeting

Maxime GONTHIER - Loris MARCHAL - Samuel THIBAULT - Elisabeth Larsson - Carl Nettelblad

maxime.gonthier@ens-lyon.fr

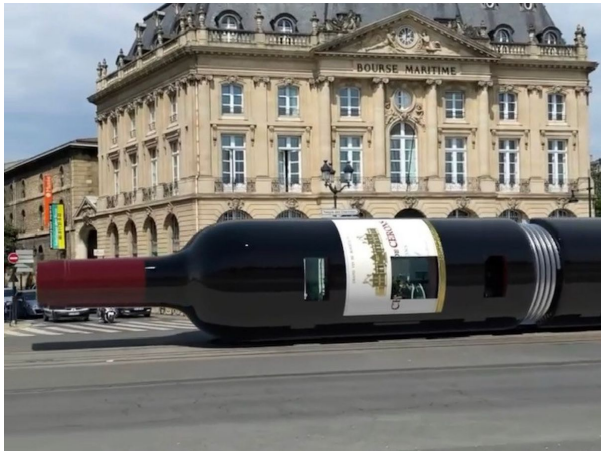
LIP - ROMA - LaBRI - STORM

June 13, 2022

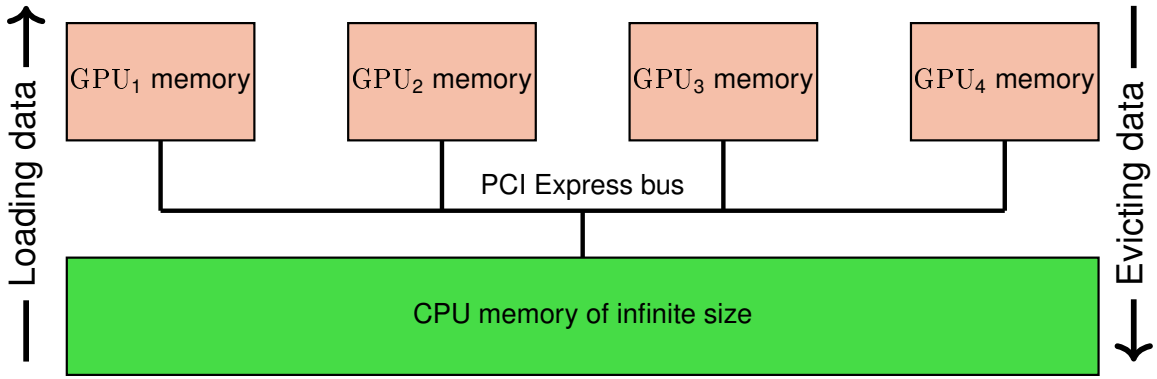
ENS DE LYON université de BORDEAUX

Who am I ?

From Bordeaux (France)



PhD on locality-aware scheduling of tasks sharing data on GPUs

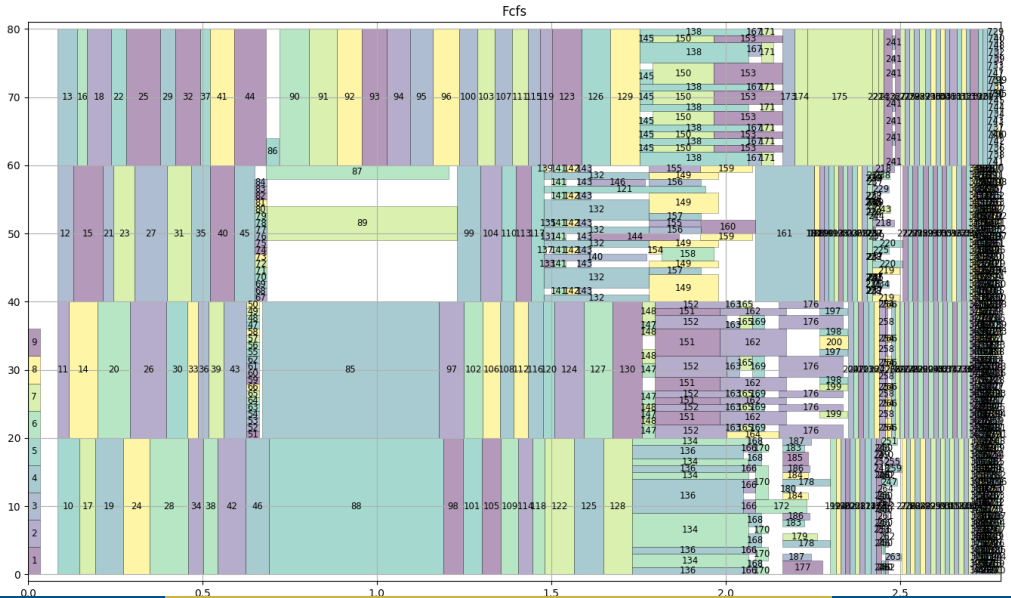


ENS DE LYON université de BORDEAUX

Motivation

- Users may submits tens of hundreds of separate jobs using the same large input files
- Jobs will read inputs directly from a shared file system
- Large files increase the queue times of the next jobs on the node

How can we minimize the amount of transfers between the shared file system and the nodes ?



Framework

Jobset $\mathbb{J}$

- Set of jobs from Rackham's history with an added input file on each job
- A file is shared by consecutive jobs submitted by the same user
- Each job require only one node and between 1 and 20 cores.

Nodes

- Rackham's cluster (128, 256 and 1024 GB nodes)
- A file of size 1024 GB can only be scheduled on 1024 GB nodes
- Bandwidth of 0.1 GB/s for each node
- Each node has $N = 20$ cores

Two different constraints

Constraint 1: Maximize file re-use

A file is re-used if it's used by consecutive or parallel jobs on the same node

Constraint 2: Allow smaller jobs to be computed on bigger nodes

For example we want to allow 128 GB jobs on 256 and 1024 GB nodes. It is not the case currently on Rackham.

Metrics

We define the $flow_{J_i}$ of a job as:

$$flow_{J_i} = Completiontime(J_i) - Subtime(J_i)$$

Obj. 1 Minimize total flow : $$minimize \sum_{i=0}^{|\mathbb{J}|} flow_{J_i}$$

Obj. 2 Minimize flow stretch : $$minimize \frac{\sum_{i=0}^{|\mathbb{J}|} \frac{flow_{J_i}}{Duration(J_i) + \frac{\mathcal{M}(\mathcal{F}(J_i))}{BW}}}{|\mathbb{J}|}$$

A secondary objective is to minimize the amount of file load.

Experimental settings

General

- Reduced cluster and set of jobs
- Get a set of jobs from Rackham's history and compute the starting state the day before the day we want to evaluate.

Evaluate file sharing

- Each jobs is using a file

Evaluate size constraints

- Tiled Cholesky decomposition (without dependencies) ($A = L \times L^T$)

Algorithms

To focus on data locality

FCFS with a score

To focus on the size constraint

Area filling
Backfill on bigger nodes

To focus on data locality: FCFS with a score

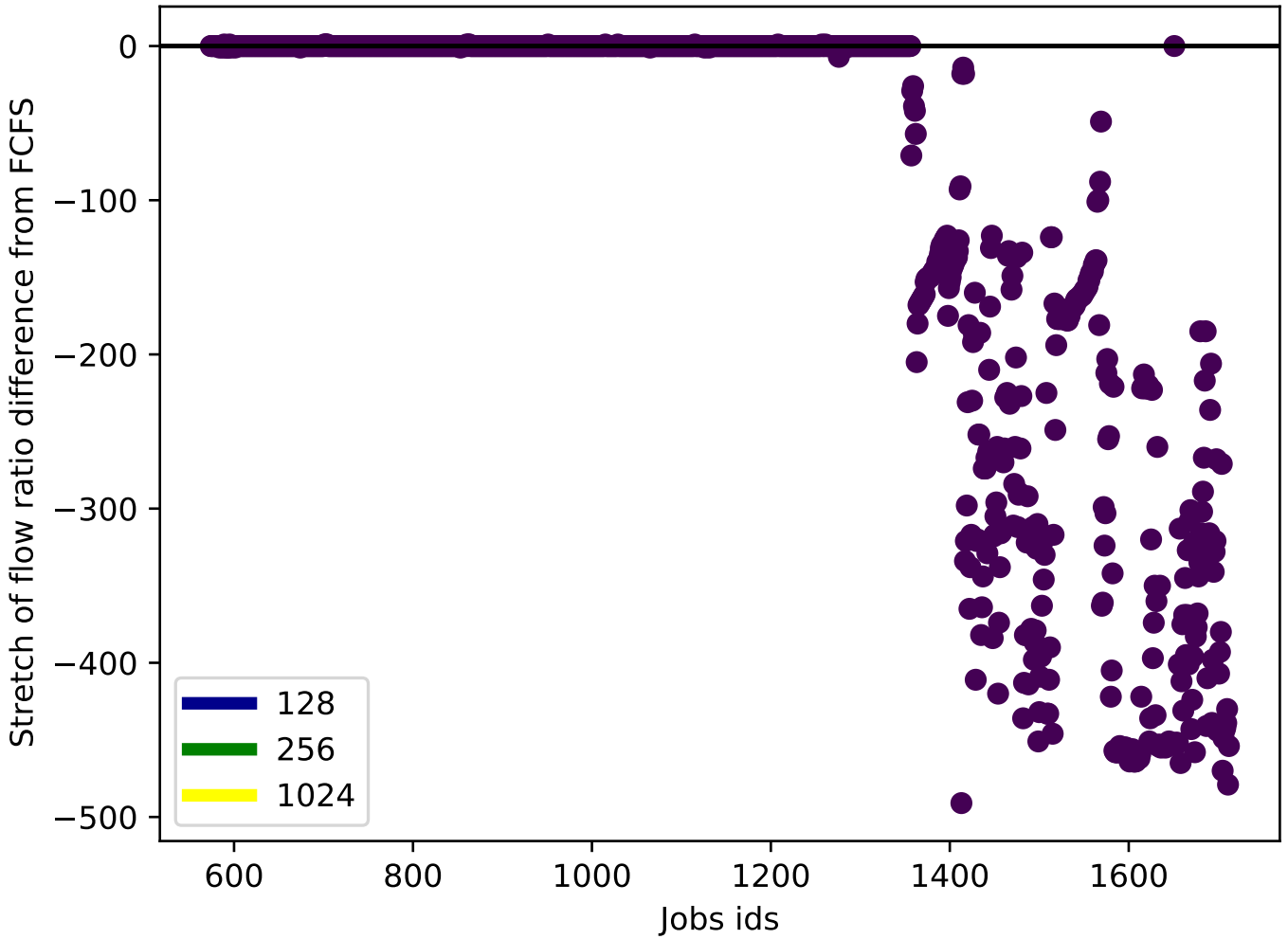
We want to schedule job J_i

For each node do:

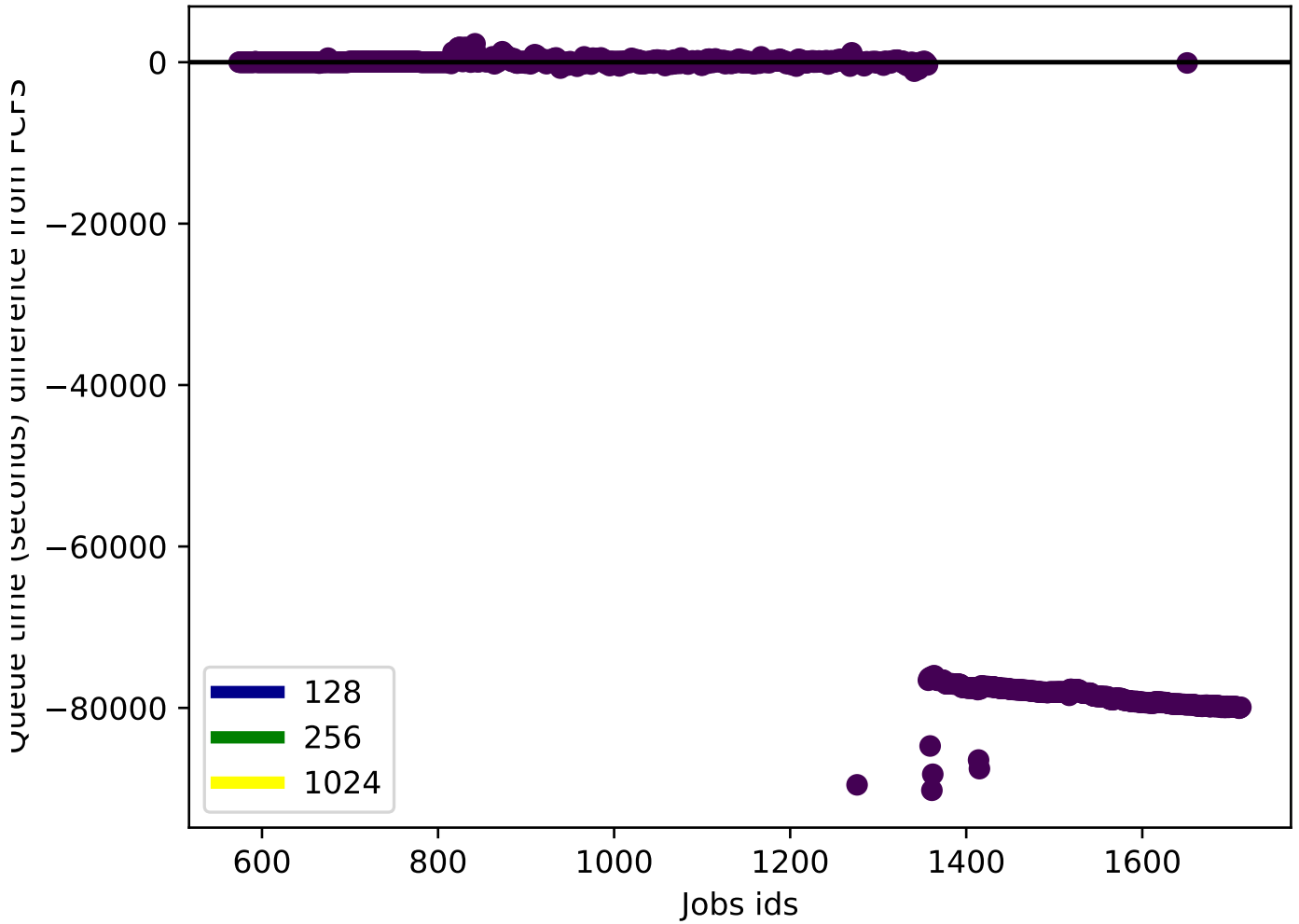
1. A = Earliest available time to host job J_i
2. B = Time to load J_i input not yet in memory
3. C = Time to re-load evicted files
4. D = Time to load copies of J_i input on other nodes
5. $Score = A + B + C + D$ (with some multipliers)

Schedule J_i on the node with the lowest score

Flow stretch of FCFS with a score compared to FCFS on the file sharing constraint



Queue times of FCFS with a score compared to FCFS on the file sharing constraint



To focus on the size constraint: Backfill on bigger nodes

Intuition: Only schedule jobs that can be started immediately on big nodes.

1. Sort job list by size of required node (biggest to smallest) and then by submission time
2. Need to schedule job J_i, requiring a node of size x
   1. If J_i can start immediately on a node of size x, start it there
   2. Else if J_i can start immediately on a node of size $x + 1$, start it there
   3. ...
   4. Else schedule J_i on a node of size x

To focus on the size constraint: Area filling

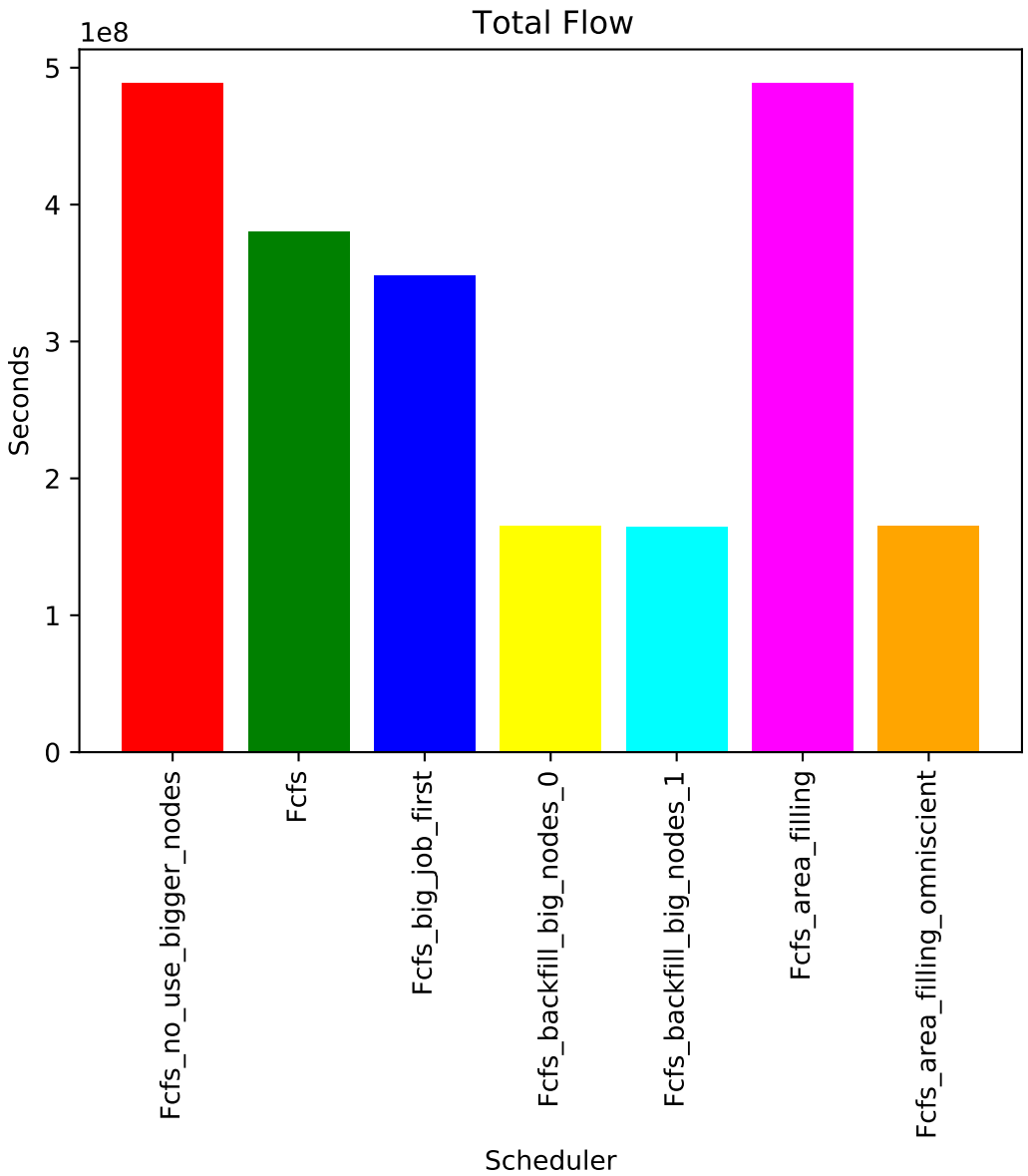
We compute statistics from previous workloads:

- For each job we compute it's usage area with $Area(J_i) = Cores(J_i) \times Walltime(J_i)$
- $Area(\mathbb{J}) \leftarrow \sum_{i=0}^{|\mathbb{J}|} Area(J_i)$
- $T_{max} \leftarrow \frac{Area(\mathbb{J})}{K}$, with K the total number of nodes
- For each size x, compute:
  $Area\_Jobs(x) \leftarrow \sum_{i=0}^{n} Area(J_i)$ *such that input size of* $J_i = x$

We compute for each size: $Area(x) \leftarrow \frac{Area\_Jobs(x)}{Nb\ of\ nodes\ of\ size\ x}$ which correspond to the area jobs of size x will occupy if they are scheduled only on nodes of size x.
We can then compute $Planned\_Area(x)$, that will indicate which area of each job size can be computed on nodes of size x.

If $Planned\_Area(x) < T_{max}$ ➔ jobs of size $x - 1$ can be scheduled on nodes of size x

Total flow


Total Flow
1e8
5
4
3
2
1
0
Seconds
Fcfs_no_use_bigger_nodes
Fcfs
Fcfs_big_job_first
Fcfs_backfill_big_nodes_0
Fcfs_backfill_big_nodes_1
Fcfs_area_filling
Fcfs_area_filling_omniscient
Scheduler

Conclusion and future work

Building strategies that focus on file re-use can reduce the flow of each job

Our contribution → **FCFS with a score and a strategy that allows smaller jobs to use bigger nodes**

Future work

- Continue working on Area filling
- Combine the 2 constraints
- Add backfilling
- Compare against FCFS EASY Backfilling
- Reduce complexity to test on full scale workloads and cluster
- Try our algorithms on SLURM ?