Throughput Models of traccc

Stephen Nicholas Swatman

ACTS Parallelization Meeting
Friday, January 24th, 2024

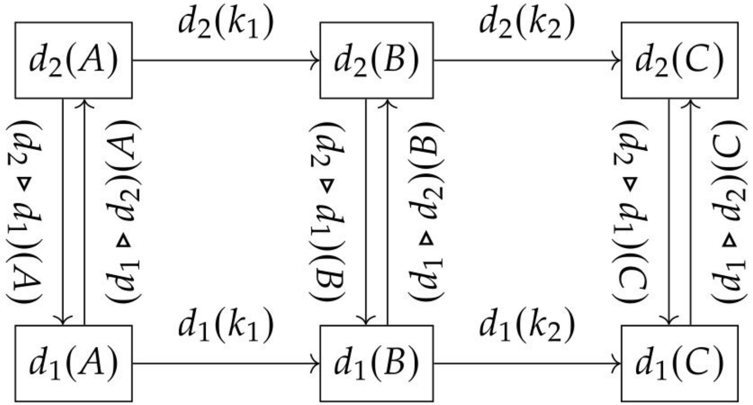
Recap: Lipstick

- Presented earlier this year
- Optimistic throughput model for heterogeneous task graph applications
- How did it work?

1. Model problem as task graph

$$\begin{tikzcd}
d_2(A) \arrow[r, "d_2(k_1)"] \arrow[d, shift right, "(d_2 \triangleright d_1)(A)"'] & d_2(B) \arrow[r, "d_2(k_2)"] \arrow[d, shift right, "(d_2 \triangleright d_1)(B)"'] & d_2(C) \arrow[d, shift right, "(d_2 \triangleright d_1)(C)"'] \\
d_1(A) \arrow[r, "d_1(k_1)"] \arrow[u, shift right, "(d_1 \triangleright d_2)(A)"'] & d_1(B) \arrow[r, "d_1(k_2)"] \arrow[u, shift right, "(d_1 \triangleright d_2)(B)"'] & d_1(C) \arrow[u, shift right, "(d_1 \triangleright d_2)(C)"']
\end{tikzcd}$$

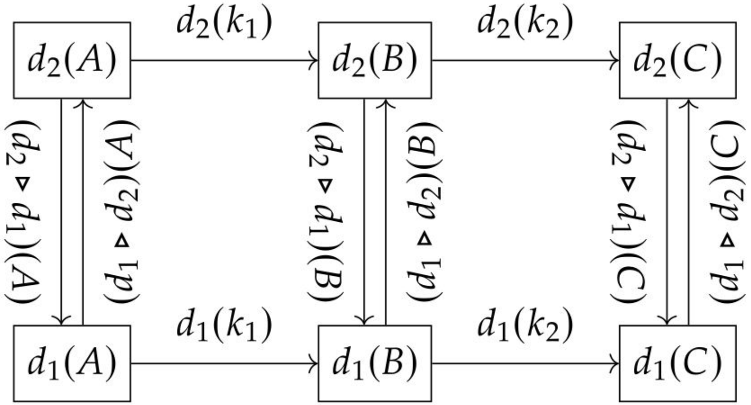
1. Model problem as task graph


$d_2(A)$
$d_2(k_1)$
$d_2(B)$
$d_2(k_2)$
$d_2(C)$
$(d_2 \triangleright d_1)(A)$
$(d_1 \triangleright d_2)(A)$
$(d_2 \triangleright d_1)(B)$
$(d_1 \triangleright d_2)(B)$
$(d_2 \triangleright d_1)(C)$
$(d_1 \triangleright d_2)(C)$
$d_1(A)$
$d_1(k_1)$
$d_1(B)$
$d_1(k_2)$
$d_1(C)$

2. Convert flow problem to LP

$$\begin{array}{ll}
\text{find} & \vec{x} \\
\text{that maximises} & \sum_{e \in E^-(t)} \vec{x}_e \\
\text{subject to} & \forall e \in E : 0 \leq \vec{x}_e \leq f(e) \\
 & \forall v \in V \setminus \{s, t\} : \sum_{e \in E^+(v)} \vec{x}_e = \sum_{e \in E^-(v)} \vec{x}_e
\end{array}$$

1. Model problem as task graph


$d_2(A)$
$d_2(k_1)$
$d_2(B)$
$d_2(k_2)$
$d_2(C)$
$(d_2 \triangleright d_1)(A)$
$(d_1 \triangleright d_2)(A)$
$(d_2 \triangleright d_1)(B)$
$(d_1 \triangleright d_2)(B)$
$(d_2 \triangleright d_1)(C)$
$(d_1 \triangleright d_2)(C)$
$d_1(A)$
$d_1(k_1)$
$d_1(B)$
$d_1(k_2)$
$d_1(C)$

2. Convert flow problem to LP

find $\vec{x}$

that maximises $\sum_{e \in E^-(t)} \vec{x}_e$

subject to $\forall e \in E : 0 \leq \vec{x}_e \leq f(e)$

$\forall v \in V \setminus \{s, t\} : \sum_{e \in E^+(v)} \vec{x}_e = \sum_{e \in E^-(v)} \vec{x}_e$

3. Add resource constraints

find $\vec{x}$

that maximises $\sum_{e \in E^-(t)} \vec{x}_e f(e)$

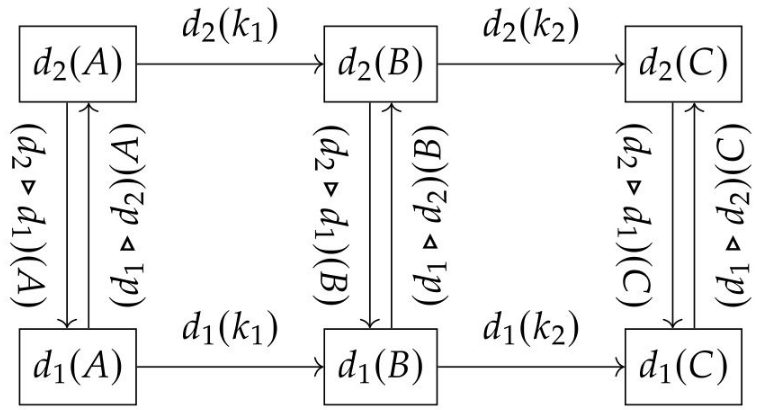
subject to $\forall e \in E : 0 \leq \vec{x}_e \leq 1$

$\forall v \in V \setminus \{s, t\} : \sum_{e \in E^+(v)} \vec{x}_e f(e) = \sum_{e \in E^-(v)} \vec{x}_e f(e)$

$\forall d \in D : 0 \leq \sum_{e \in \{d(k) : k \in K\} \cap E_K} \vec{x}_e \leq 1$

$\forall d_1, d_2 \in D : 0 \leq \sum_{e \in (\{(d_1 \triangleright d_2)(Q) : Q \in T\} \cup \{(d_2 \triangleright d_1)(Q) : Q \in T\}) \cap E_I} \vec{x}_e \leq 1$

1. Model problem as task graph


$d_2(A)$
$d_2(k_1)$
$d_2(B)$
$d_2(k_2)$
$d_2(C)$
$(d_2 \triangleright d_1)(A)$
$(d_1 \triangleright d_2)(A)$
$(d_2 \triangleright d_1)(B)$
$(d_1 \triangleright d_2)(B)$
$(d_2 \triangleright d_1)(C)$
$(d_1 \triangleright d_2)(C)$
$d_1(A)$
$d_1(k_1)$
$d_1(B)$
$d_1(k_2)$
$d_1(C)$

2. Convert flow problem to LP

$$\begin{array}{ll}
\text{find} & \vec{x} \\
\text{that maximises} & \sum_{e \in E^-(t)} \vec{x}_e \\
\text{subject to} & \forall e \in E : 0 \leq \vec{x}_e \leq f(e) \\
 & \forall v \in V \setminus \{s,t\} : \sum_{e \in E^+(v)} \vec{x}_e = \sum_{e \in E^-(v)} \vec{x}_e
\end{array}$$

3. Add resource constraints

$$\begin{array}{ll}
\text{find} & \vec{x} \\
\text{that maximises} & \sum_{e \in E^-(t)} \vec{x}_e f(e) \\
\text{subject to} & \forall e \in E : 0 \leq \vec{x}_e \leq 1 \\
 & \forall v \in V \setminus \{s,t\} : \sum_{e \in E^+(v)} \vec{x}_e f(e) = \sum_{e \in E^-(v)} \vec{x}_e f(e) \\
 & \forall d \in D : 0 \leq \sum_{e \in \{d(k) : k \in K\} \cap E_K} \vec{x}_e \leq 1 \\
 & \forall d_1, d_2 \in D : 0 \leq \sum_{e \in (\{(d_1 \triangleright d_2)(Q) : Q \in T\} \cup \{(d_2 \triangleright d_1)(Q) : Q \in T\}) \cap E_I} \vec{x}_e \leq 1
\end{array}$$

4. Solve using e.g. Z3

Z3

Input using YAML program specifications

```
datatypes:
  A:
        size: 1
        count: 1
  B:
        size: 1
        count: 1
  C:
        size: 1
        count: 1
devices:
  - d1
  - d2
interconnects:
  - source: d1
        destination: d2
        bandwidth: 100
        bidirectional: true
algorithms:
  k1:
        in_type: A
        out_type: B
        implementations:
        - device: d1
        throughput: 5000
        - device: d2
        throughput: 500000000000
  k2:
        in_type: B
        out_type: C
        implementations:
        - device: d1
        throughput: 10000
        - device: d2
        throughput: 10
source: A
sink: C
```

Can we do this for traccc?

- Input: throughput of kernels running full beans on the device
- We do not have this data 🙁
- But we can compute it!

Computing throughput

$$T(N, L) = \frac{N}{L}$$

We can measure latency

Computing throughput

We can’t measure parallelism, but we can model it

$$T(N, L) = \frac{N}{L}$$

We can measure latency

Computing parallelism

Accounting for tail effects

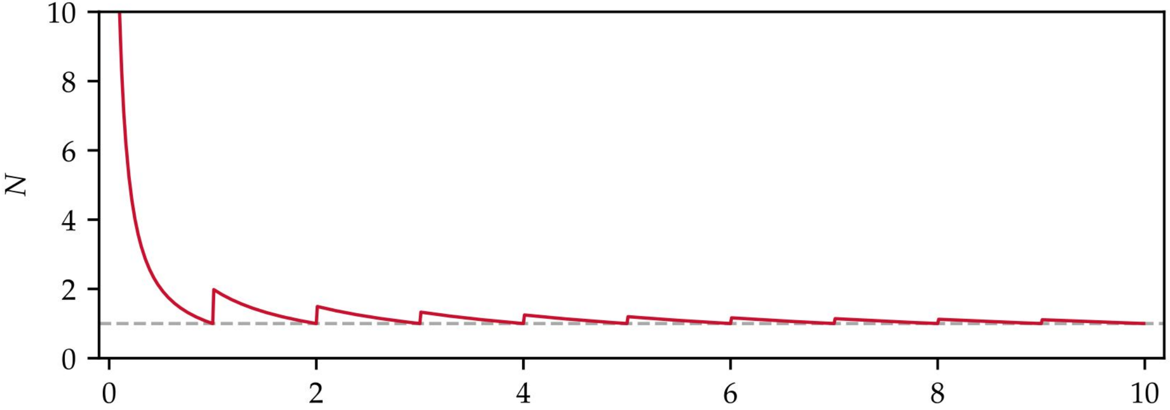
We know the number of threads required

$$N = \frac{\lceil k \rceil}{k} \quad \text{where} \quad k_{\mathrm{GPU}} = \frac{c_r}{q c_s}$$

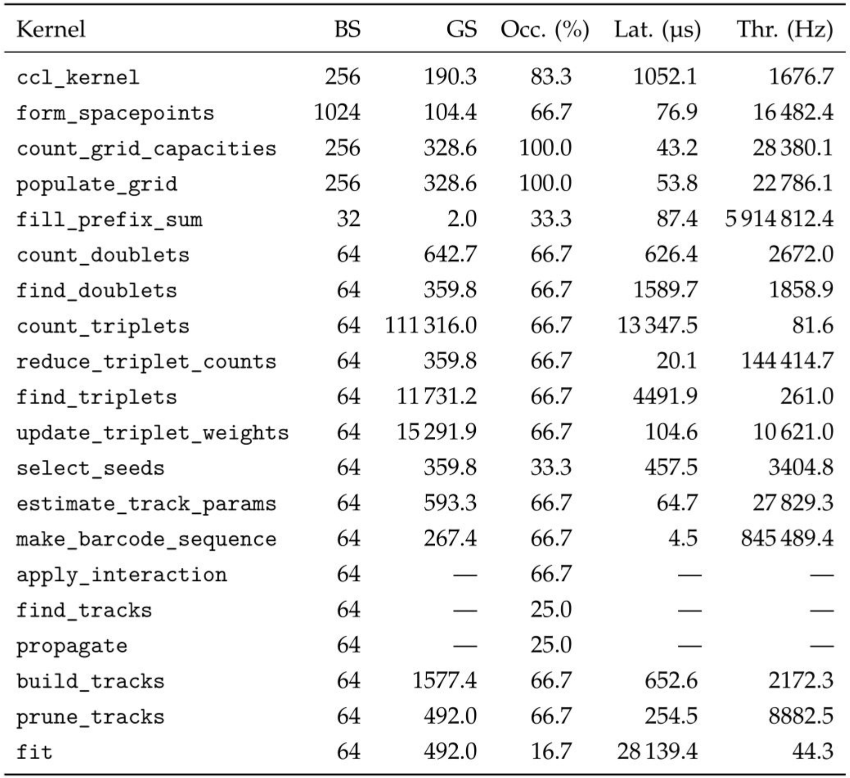
We can measure occupancy

Thread slots follow from hardware specs

Slide for sawtooth enthusiasts

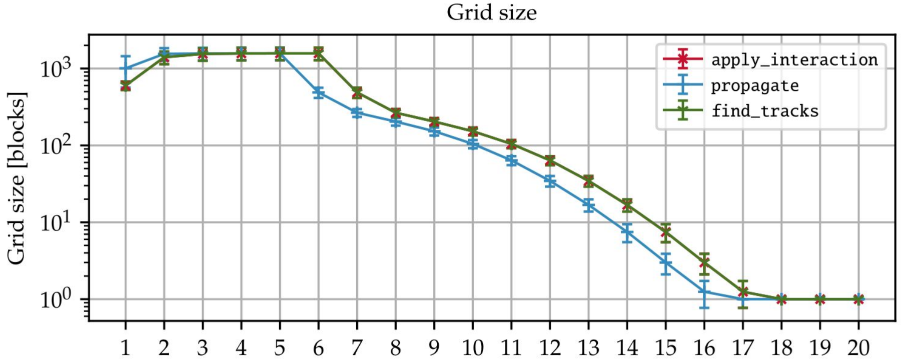


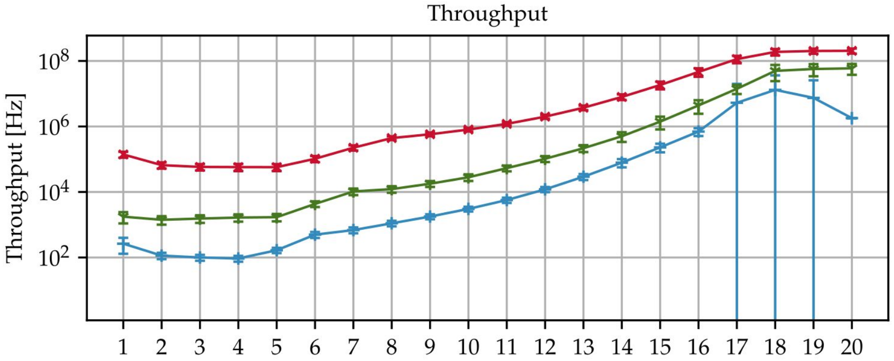
Kernel results

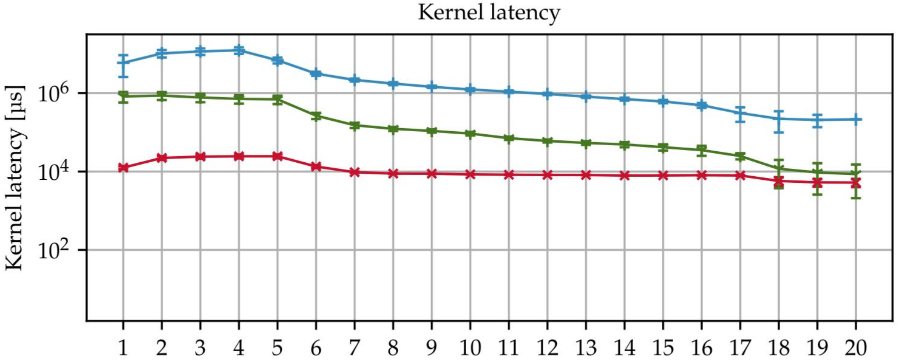


Kernel	BS	GS	Occ. (%)	Lat. (µs)	Thr. (Hz)
ccl_kernel	256	190.3	83.3	1052.1	1676.7
form_spacepoints	1024	104.4	66.7	76.9	16 482.4
count_grid_capacities	256	328.6	100.0	43.2	28 380.1
populate_grid	256	328.6	100.0	53.8	22 786.1
fill_prefix_sum	32	2.0	33.3	87.4	5 914 812.4
count_doublets	64	642.7	66.7	626.4	2672.0
find_doublets	64	359.8	66.7	1589.7	1858.9
count_triplets	64	111 316.0	66.7	13 347.5	81.6
reduce_triplet_counts	64	359.8	66.7	20.1	144 414.7
find_triplets	64	11 731.2	66.7	4491.9	261.0
update_triplet_weights	64	15 291.9	66.7	104.6	10 621.0
select_seeds	64	359.8	33.3	457.5	3404.8
estimate_track_params	64	593.3	66.7	64.7	27 829.3
make_barcode_sequence	64	267.4	66.7	4.5	845 489.4
apply_interaction	64	—	66.7	—	—
find_tracks	64	—	25.0	—	—
propagate	64	—	25.0	—	—
build_tracks	64	1577.4	66.7	652.6	2172.3
prune_tracks	64	492.0	66.7	254.5	8882.5
fit	64	492.0	16.7	28 139.4	44.3

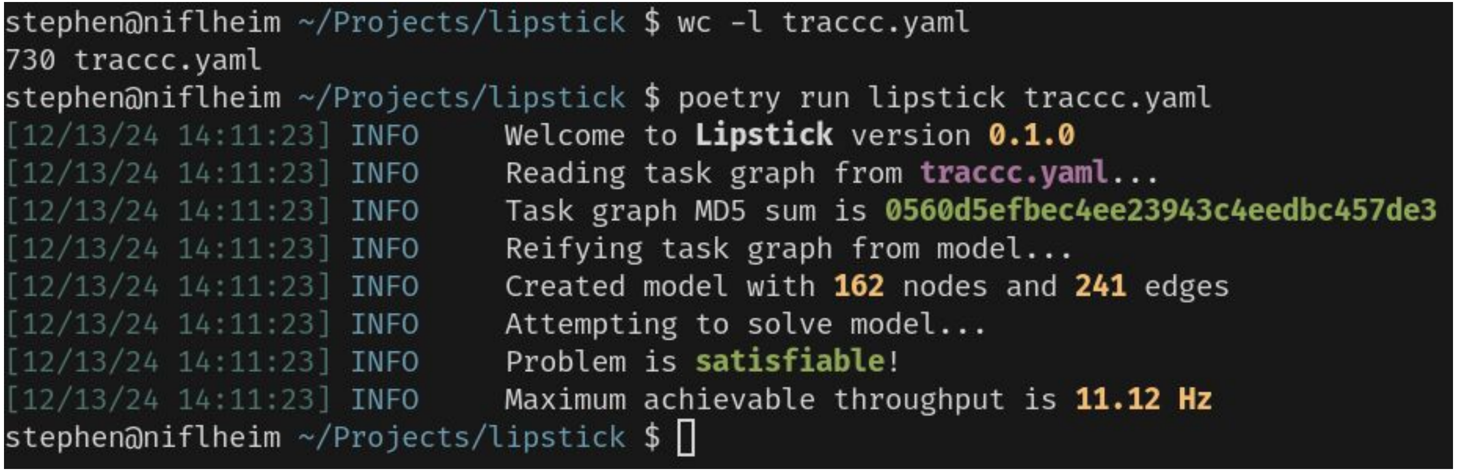
CKF kernel results





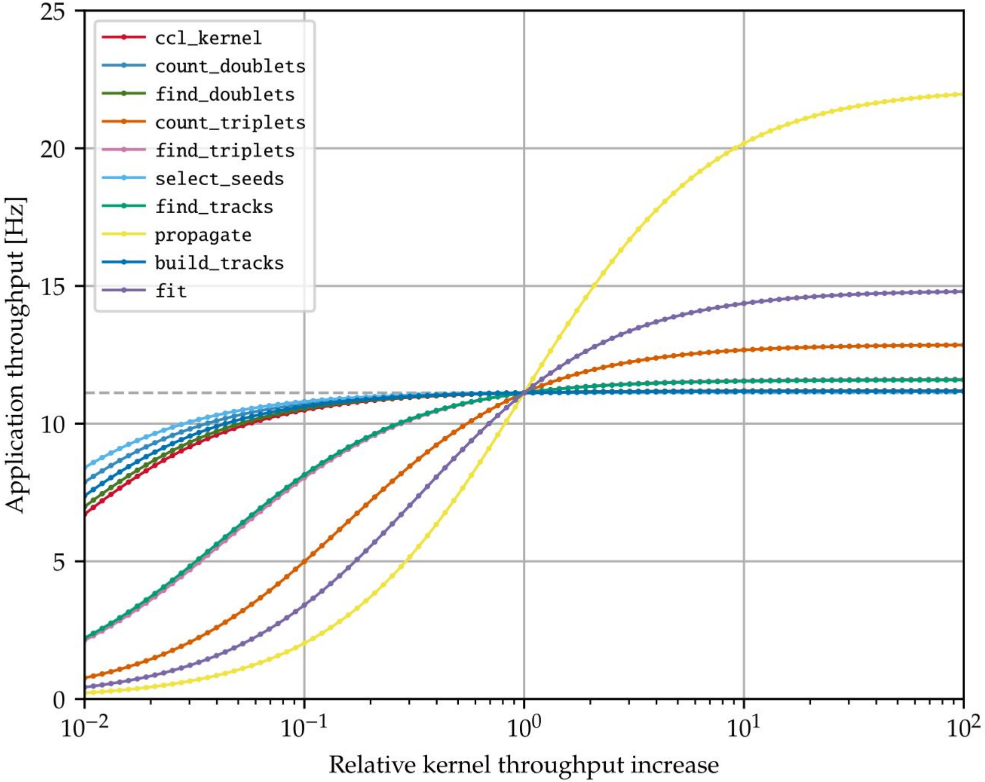


Back to throughput



```
stephen@niflheim ~/Projects/lipstick $ wc -l traccc.yaml
730 traccc.yaml
stephen@niflheim ~/Projects/lipstick $ poetry run lipstick traccc.yaml
[12/13/24 14:11:23] INFO     Welcome to Lipstick version 0.1.0
[12/13/24 14:11:23] INFO     Reading task graph from traccc.yaml...
[12/13/24 14:11:23] INFO     Task graph MD5 sum is 0560d5efbec4ee23943c4eedbc457de3
[12/13/24 14:11:23] INFO     Reifying task graph from model...
[12/13/24 14:11:23] INFO     Created model with 162 nodes and 241 edges
[12/13/24 14:11:23] INFO     Attempting to solve model...
[12/13/24 14:11:23] INFO     Problem is satisfiable!
[12/13/24 14:11:23] INFO     Maximum achievable throughput is 11.12 Hz
stephen@niflheim ~/Projects/lipstick $ 
```

Differential results


ccl_kernel
count_doublets
find_doublets
count_triplets
find_triplets
select_seeds
find_tracks
propagate
build_tracks
fit
Application throughput [Hz]
10^{-2}
10^{-1}
10^{0}
10^{1}
10^{2}
Relative kernel throughput increase