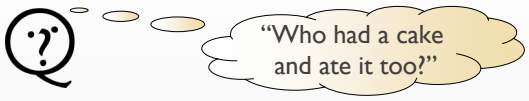
Powerset (a Microsoft company)

Combinatorial Set Matching Using GPUs

Anand Madhavan and Eitan Frachtenberg
{anand, eitan}@powerset.com

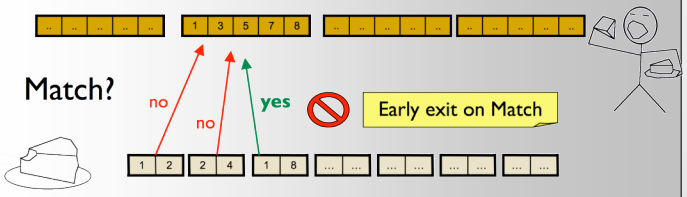


(Refer to the right sidebar titled 'Motivating example' for more information)

The Problem

Given a number of E_{max} tuples, we are interested in finding the ones for which there exists at least one E_{min} tuple, as a subset

E_{max} tuples (where entity *having* and entity *eating* cake are the same)



E_{min} tuples (facts where cake being *had* and cake being *eaten* are the same)

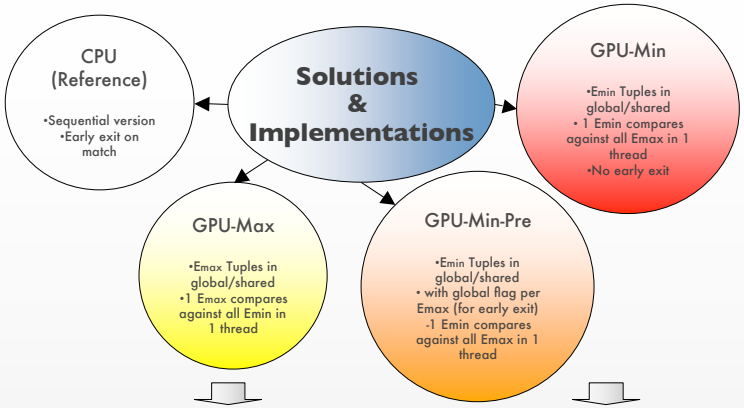
Challenges

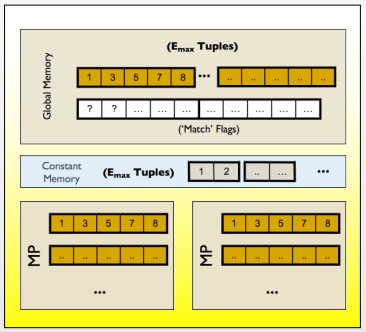
Low arithmetic intensity
`…(t1[i]==t2[i])..&&..(t1[j]==t2[j])…`
mostly boolean operations and comparisons

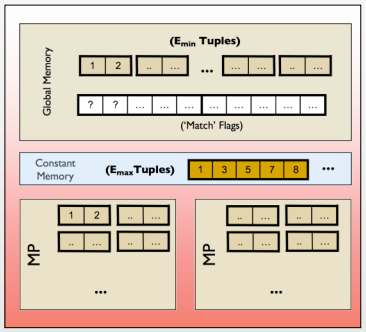
Typical size does not fit in constant memory
100's of kB of tuples

Sequential version very effective when early matches occur

Early exit on Match

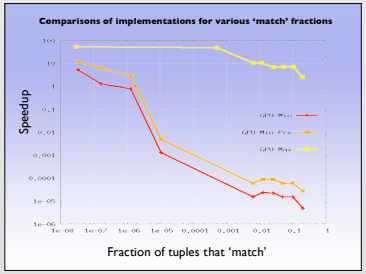


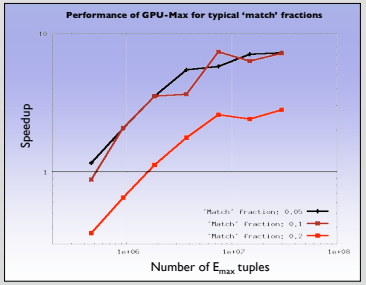




Results and Conclusions

- Speedups are measured against the CPU (Reference) implementation
- Speedups are sensitive to the fraction of all tuples that 'match'. The Sequential algorithm takes advantage of 'early exit' at higher fractions.
- Comparison of GPU-Min-Pre and GPU-Min illustrate that it might be worthwhile doing 'pre-check's (device global memory access) for every E_{max} tuple to facilitate early exit
- Although low 'match' ratios are not typical, we are able to see speedups of 50x **despite** the low arithmetic intensity





Since the **GPU-Max** implementation outperforms the others, we study its performance for other sizes of inputs and typical hit-ratios…

•GPU-Max with single E_{max} in a thread matched against all other E_{min} takes advantage of 'early exit' just like the sequential algorithm

• For the typical 0.1 'match' fraction, we see speedups of upto 8x. With increasing problem size, we see better speedups (as expected) as the cost of data transfer to card gets amortized

Motivating example

"Who had a cake and ate it too?"

Mary and **John** were participants in a **cake**-making competition. Making a **cake** was a piece of **cake** to **Mary** and **John**. As the competition got started, **Mary** started making a chocolate **cake**. **She** put **it** in the oven, whilst **John** was busy making his **cheesecake**. Towards the end of the competition, **many** of them **had** some delicious **cakes** displayed on their tables. **Mary** gave a piece of her **cake** to **John** for tasting. **John** took the piece of **cake**. **He had it** in his hand, while eyeing his own **cheesecake**. **He** ended up **eating** a piece of the **cheesecake** instead. **Mary** on other hand, **had** her chocolate **cake** in her hand and was waiting for **John** to try it. While waiting **she ate** some of her chocolate **cake**.

Who *ate* cakes?

Who *had* cakes?

Who *ate* the exact same cake they *had*?

The problem can be described as combinatorial set generation and matching.

The Emin tuples are generated as part of a linguistic constraint. The Emax tuples are generated as part of another linguistic constraint and the combinatorial set matching is required as part of a combined constraint.

Although the number of cakes or entities in a document is bounded, the number of set combinations can grow exponentially.

For more information contact
{anand, eitan}@powerset.com

© 2008 Powerset