Paralelization Strategies for Ant Colony Optimization on GPUs

José M. Cecilia, José M. García. University of Murcia (Spain)
Andy Nisbet, Martyn Amos. Manchester Metropolitan University (UK)
Manuel Ujaldón. University of Málaga (Spain)

Jornadas de divulgación de aplicaciones científicas y visión por computador sobre procesadores gráficos (JGPU 2011)
Alicante.

12 May 2011

Agenda

- Motivation (1 slide).
- Graphics Processing Units (GPUs) (3 slides).
- Ant Colony Optimization for TSP (3 slides).
- Parallelization Strategies for ACO on GPUs (5 slides).
- Performance Evaluation (5 slides).
- Conclusions and Future Work (2 slides).

Agenda

- Motivation (1 slide).
- Graphics Processing Units (GPUs).
- Ant Colony Optimization for TSP.
- Parallelization Strategies for ACO on GPUs.
- Performance Evaluation.
- Conclusions and Future Works

Motivation

- GPUs have democratized HPC.
  - Great FLOP/$, massively parallel chip on a commodity PC….
- However, this is not for free.
  - New programming model.
  - New challenges.
- Algorithms need to be re-implemented and rethought.
- Bio-inspired computations are challenging applications for GPUs.
  - They are intrinsically parallel by its definition.
  - They were optimized for sequential platforms.

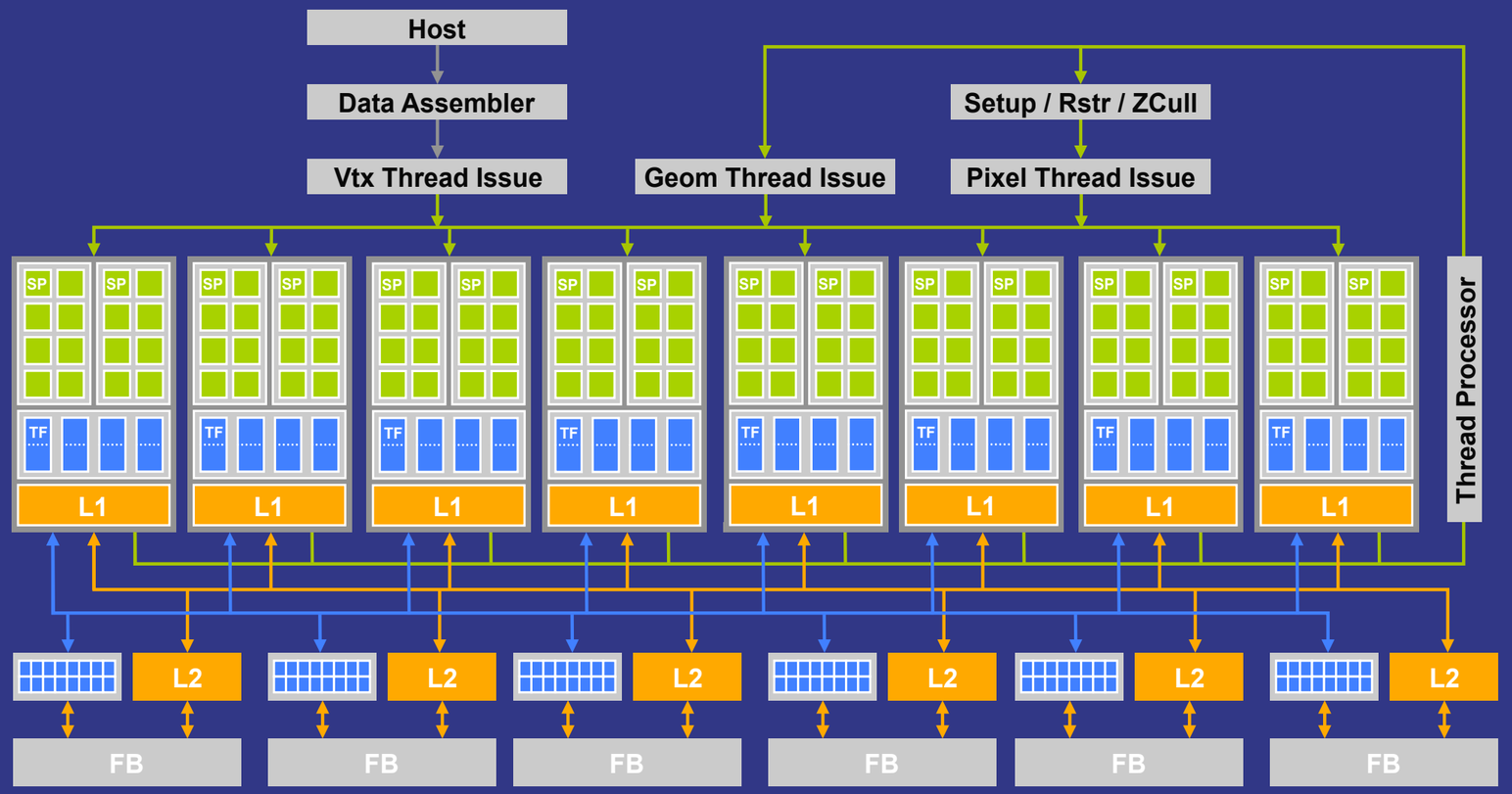
Agenda

- Motivation.
- Graphics Processing Units (GPUs) (3 slides).
- Ant Colony Optimization for TSP.
- Parallelization Strategies for ACO on GPUs.
- Performance Evaluation.
- Conclusions and Future Works.

Graphics Processing Units (GPUs)

- More silicon for processing.
- Throughput-oriented architecture.
- High Memory Bandwidth.
- Great Floating-Point Performance (up to 1 TeraFLOP).
- SP, SMs, Shared Memory, Device Memory…….

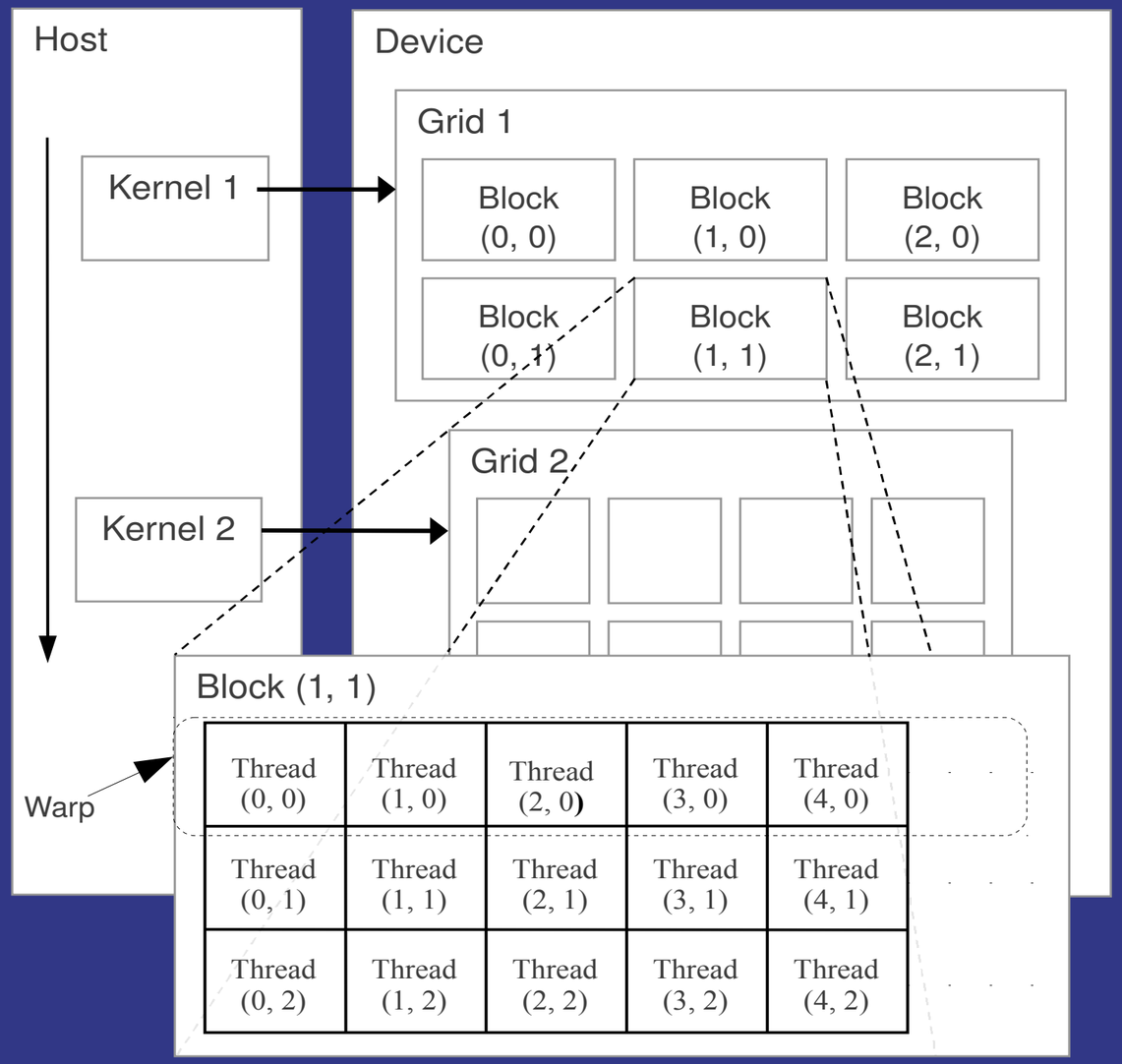
- But, different Programming approach (CUDA).



G80 NVIDIA´s Architecture (source).

Graphics Processing Units (GPUs)

- Host = CPU.
- Device = GPU.
- Thread = basic execution unit = SP.
- Block = batch of threads = sharing the SM
- Grid = batch of blocks = MIMD
- Warp = Scheduling unit = SIMD



CUDA Generations

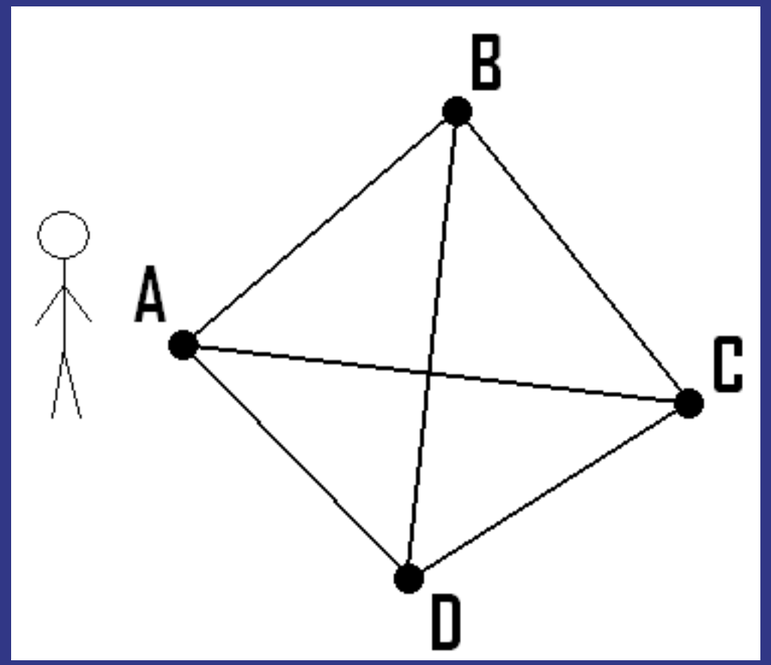
Parameter	Value as of GPU gener.			Limi-tation	Impact
CUDA Compute Capabilities	**1.0** & 1.1	1.2 & **1.3**	Fermi		
Multiprocessors / GPU	16	30	16	HW.	Scalability
Processors / Multiprocessor	8	8	32	HW.	Scalability
Threads / Warp	32	32	32	SW.	Throughput
Thread blocks / Multiprocessor	8	8	8	SW.	Throughput
Threads / Block	512	512	512	SW.	Parallelism
Threads / Multiprocessor	768	1 024	1 536	SW.	Parallelism
32 bit registers / Multiprocessor	8 192	16 384	4 096	HW.	Working set
Shared memory / Multiprocessor	16 384	16 384	16K 48K	HW.	Working set

Agenda

- Motivation.
- Graphics Processing Units (GPUs).
- Ant Colony Optimization for TSP (3 slides).
- Parallelization Strategies for ACO on GPUs.
- Performance Evaluation.
- Conclusions and Future Works.

Ant Colony Optimization for TSP

- The Travelling Salesman Problem (TSP):
  - Finding the shortest (or "cheapest") round-trip route that visits each of a number of "cities" exactly once.
  - The TSP is a well-known NP-hard optimization problem.
  - Complete weighted graph, $d_{i,j}=d_{j,i}$;
  - The first problem solved by ACO.



- The Ant Colony Optimization (ACO):
  - Uses simulated "Ants" (or agents).
  - Each ant moves through the graph until it completes a tour.
  - Then, offers this tour as its suggested solution, dropping "pheromone".
  - The amount of pheromone depends on the tour's quality.
  - Ant's movement are driven by heuristic and pheromone information.
  - Finally, evaporation to avoid stalling in a local minimum.

Ant System for the TSP

- The Ant System (AS) is an early variant of ACO.
  - Proposed by Dorigo.
- Divided into two main stages.
  - Tour Construction and Pheromone Update.
- Tour Construction:
  - m ants building tours in parallel.
  - Initially ants are randomly placed.
  - Then they apply *random proportinal rule*.
  - Each ant mantains a memory (*tabu list*).
  - The cities are selected randomly (roulette wheel)
  - Some techniques: NN-List, Choice Info matrix…

$$p_{i,j}^k = \frac{[\tau_{i,j}]^\alpha [\eta_{i,j}]^\beta}{\sum_{l \in N_i^k} [\tau_{i,l}]^\alpha [\eta_{i,l}]^\beta}, \qquad if\ j \in N_i^k,$$

Ant System for the TSP (II)

- Pheromone Update:
  - After tour construction.
  - First, lowering pheromone in all edges by a constant.
  - Adding pheromone on the tour´s edges.

$$\Delta\tau_{i,j}^{k} = \begin{cases} 1/C^{k} \text{ if } e(i,j)^{k} & \text{belongs to } T^{k} \\ 0 & \text{otherwise} \end{cases}$$

Agenda

- Motivation.
- Graphics Processing Units (GPUs).
- Ant Colony Optimization for TSP.
- Parallelization Strategies for ACO on GPUs (5 slides).
- Performance Evaluation.
- Conclusions and Future Works.

Parallelization Strategies for ACO on GPUs.

- Two main stages of the ACO algorithm, represented by two different kernels (global synchronization):
  - Tour construction
  - Pheromone Update

Tour construction kernel.

- Up to now, main strategy an ant = thread.

Advantages:

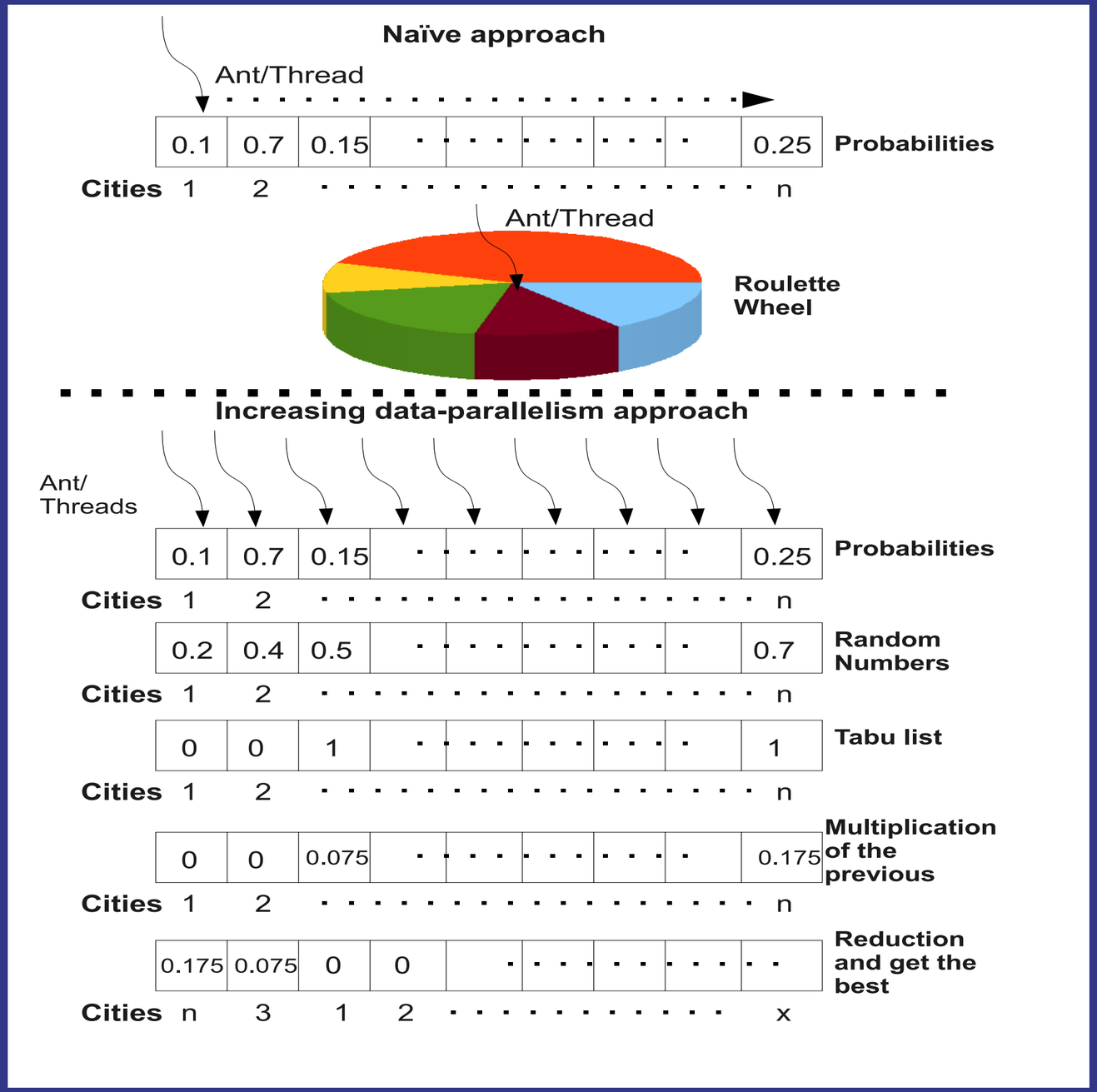
- It is the natural parallelism of ACO (*”m ants building tours in parallel.”*)

Disadvantages:

- Threads represent ant´s task.
- Threads are independent of each other (Warp divergences).
- Threads need many resources (low occupancy resources).
- Low-level of parallelism for the GPU. (800 ants = 800 cities)

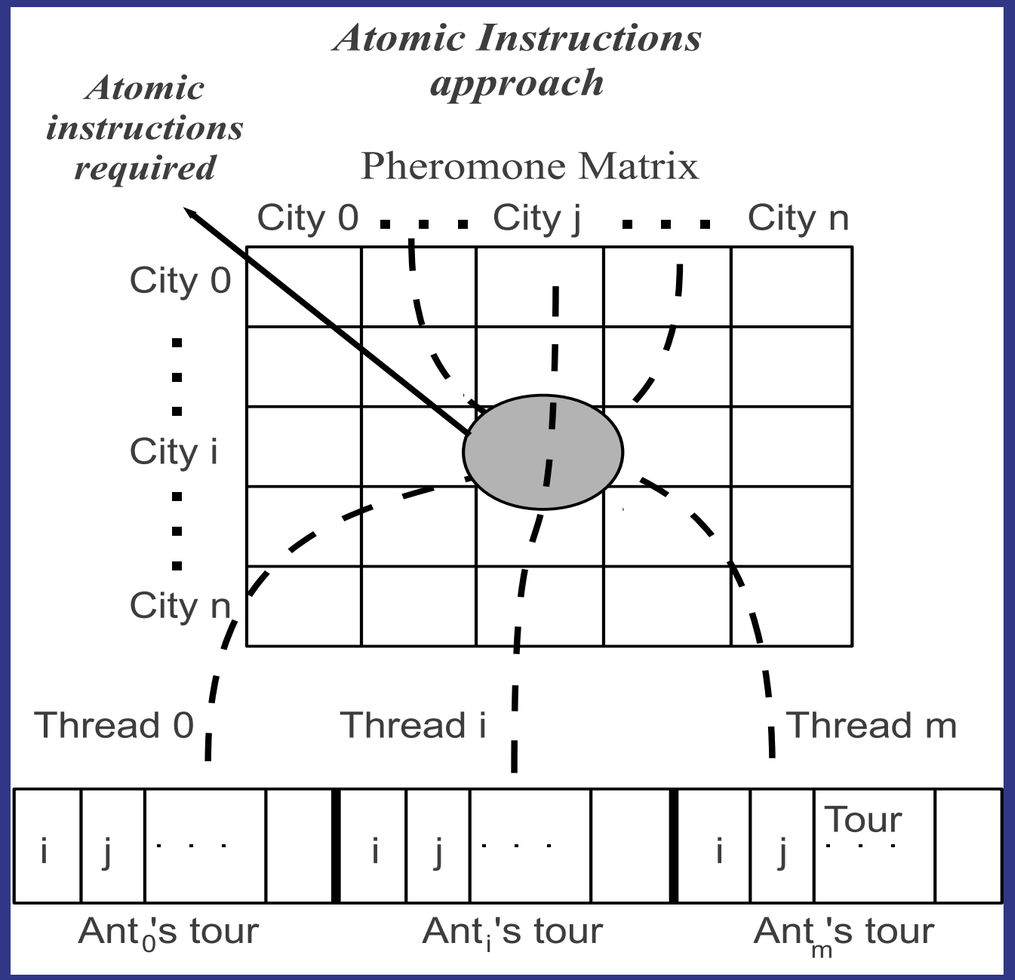
Increasing the parallelism.

- Now, a thread per each city.
- A block for each ant.



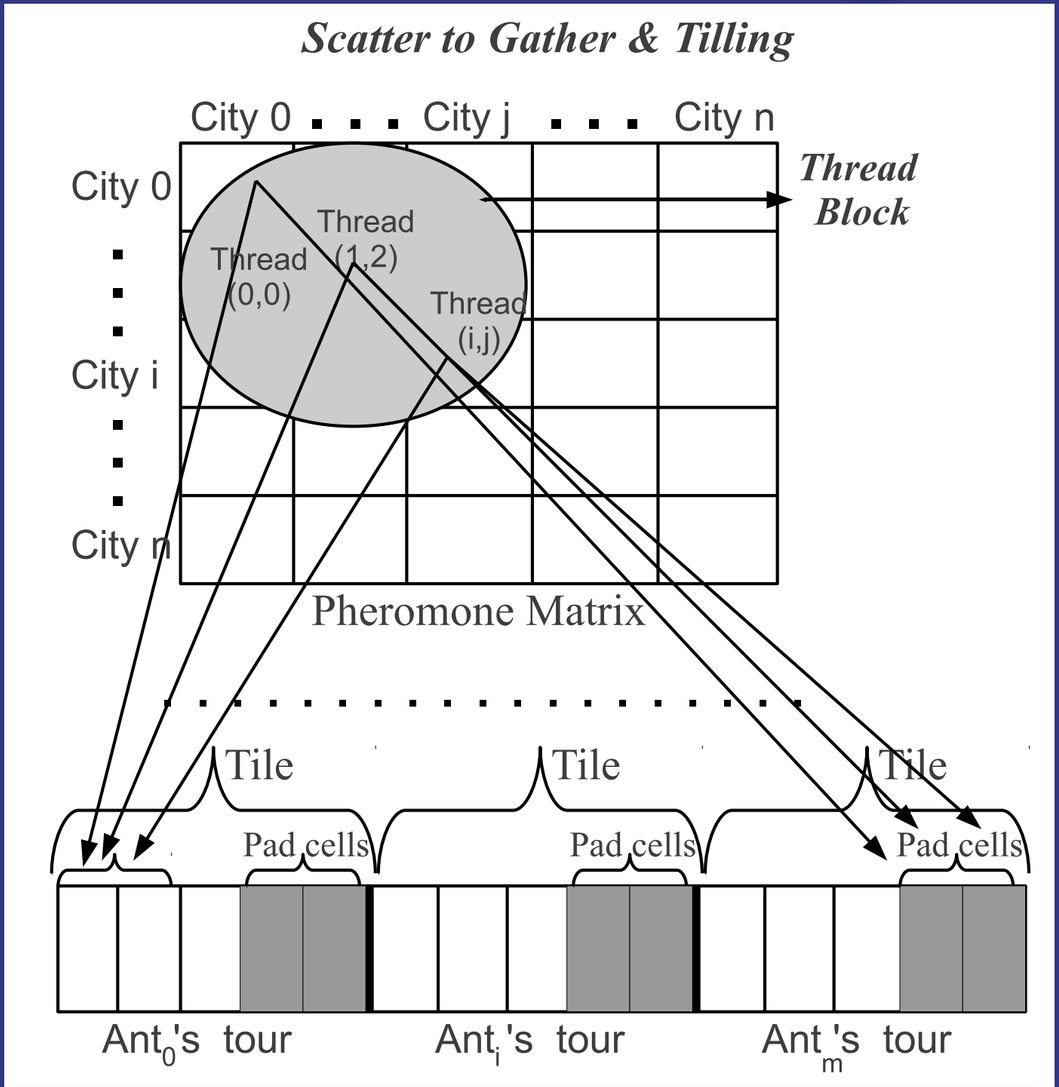
Pheromone update kernel.

- Threads per city on the ant´s tour.
- More than one ant may visit the same city.
- Atomic instructions are needed.
- Problem: Atomic operations are costly.



Alternative design: Scatter to Gather transformation+Tilling

- Old generation of GPUs was possible Gather but not Scatter.
- Gather: read from different memory address but writes in the same memory location.
- Scatter: Read from different memory location and writes into different memory address.
- With CUDA, it is possible but…
- High preassure on device memory, then tiling and thread reduction



Agenda

- Motivation.
- Graphics Processing Units (GPUs).
- Ant Colony Optimization for TSP.
- Parallelization Strategies for ACO on GPUs.
- Performance Evaluation (5 slides).
- Conclusions and Future Works.

Experimental setup

- TSPLIB Benchmark.
- Main ACO parameters:
  $\alpha, \beta, m=n$
- CPU code by provided by Stüzle.
- Results in simple-precision and per iteration, averaged 100 iterations.
- Two GPUs:
  Tesla C1060 and Tesla M2050

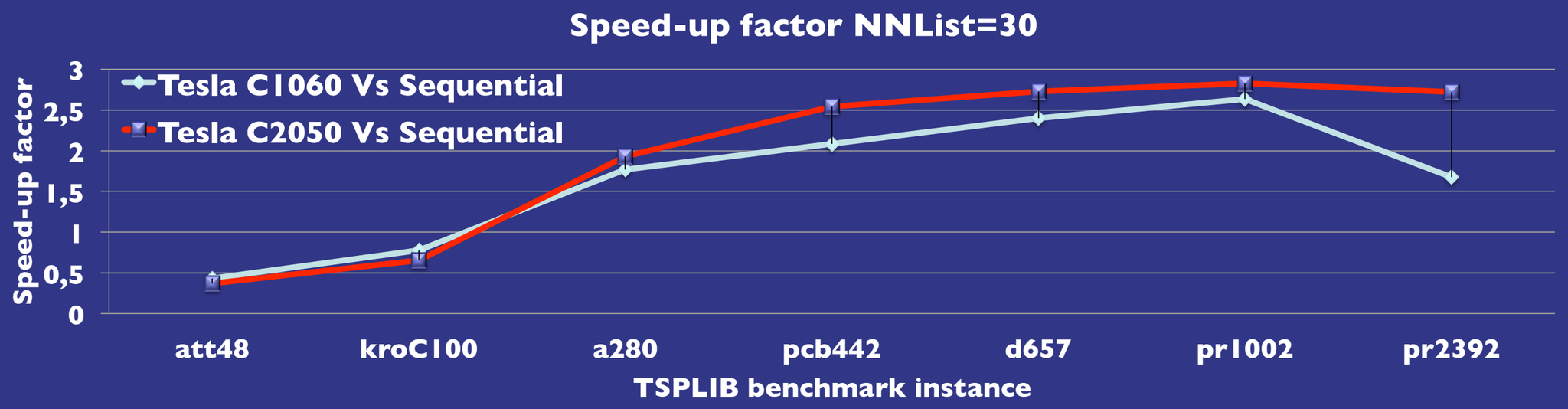
CUDA AND HARDWARE FEATURES FOR THE TESLA C1060 GPU AND THE TESLA M2050.

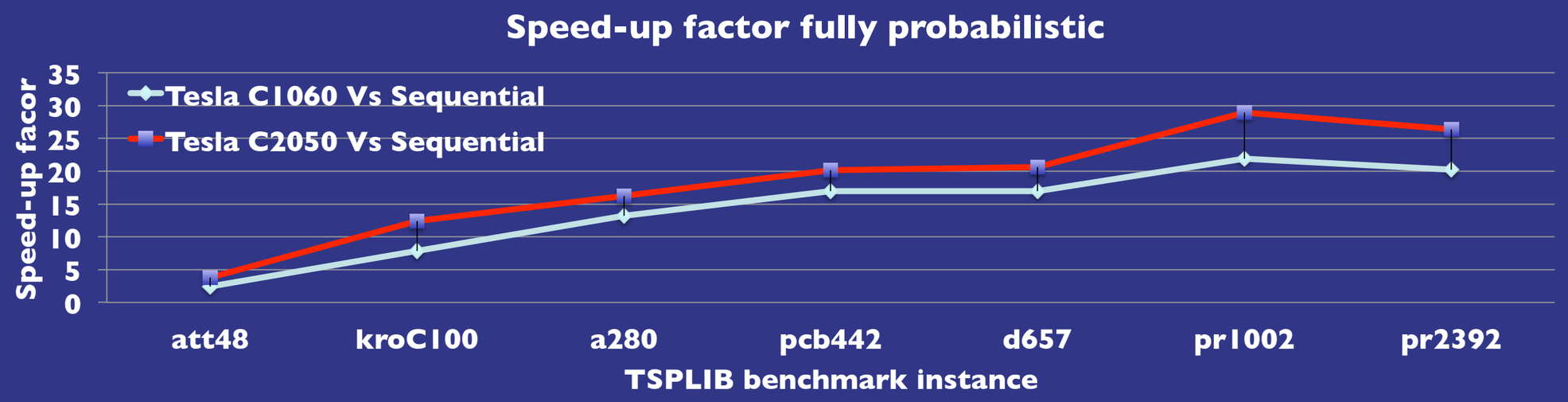
GPU element	Feature	Tesla C1060	Tesla M2050
Streaming processors (GPU cores)	Cores per SM	8	32
	Number of SMs	30	14
	Total SPs	240	448
	Clock frequency	1 296 MHz	1 147 MHz
Maximum number of threads	Per multiprocessor	1 024	1 536
	Per block	512	1 024
	Per warp	32	32
SRAM memory available per multiprocessor	32-bit registers	16 K	32 K
	Shared memory	16 KB	16/48 KB
	L1 cache	No	48/16 KB
	(Shared + L1)	16 KB	64 KB
Global (video) memory	Size	4 GB	3 GB
	Speed	2x800 MHz	2x1500 MHz
	Width	512 bits	384 bits
	Bandwidth	102 GB/sc.	144 GB/sc.
	Technology	GDDR3	GDDR5

Tour construction evaluation on the GPU

Code Version	TSPLIB benchmark instance (problem size) on Tesla C1060 (msec.)						
	att48	***kroC100***	***a280***	***pcb442***	***d657***	***pr1002***	***pr2392***
Baseline version	13,14	56,89	497,93	1201,52	2770,32	6181	63357,7
Choice Kernel	4,83	17,56	135,15	334,28	659,05	1912,59	18582,9
Without CURAND	4,50	15,78	119,65	296,31	630,01	1624,05	15514,9
NNList	2,36	6,39	33,08	72,79	143,36	338,88	2312,98
NNList + Shared	1,81	4,42	21,42	44,26	84,15	203,15	2450,52
NNList + Shared +Texture	1,35	3,51	16,97	38,39	75,07	178,30	2105,77
Increasing Data Parallelism	0,36	0,93	13,89	37,18	125,17	419,53	5525,76
Data parallelism + Texture	0,34	0,91	12,12	36,57	123,17	417,72	5461,06
Total Speedup attained	38,09x	62,83x	41,09x	32,86x	22,49x	14,8x	11,6x

Tour construction evaluation: GPU Vs CPU

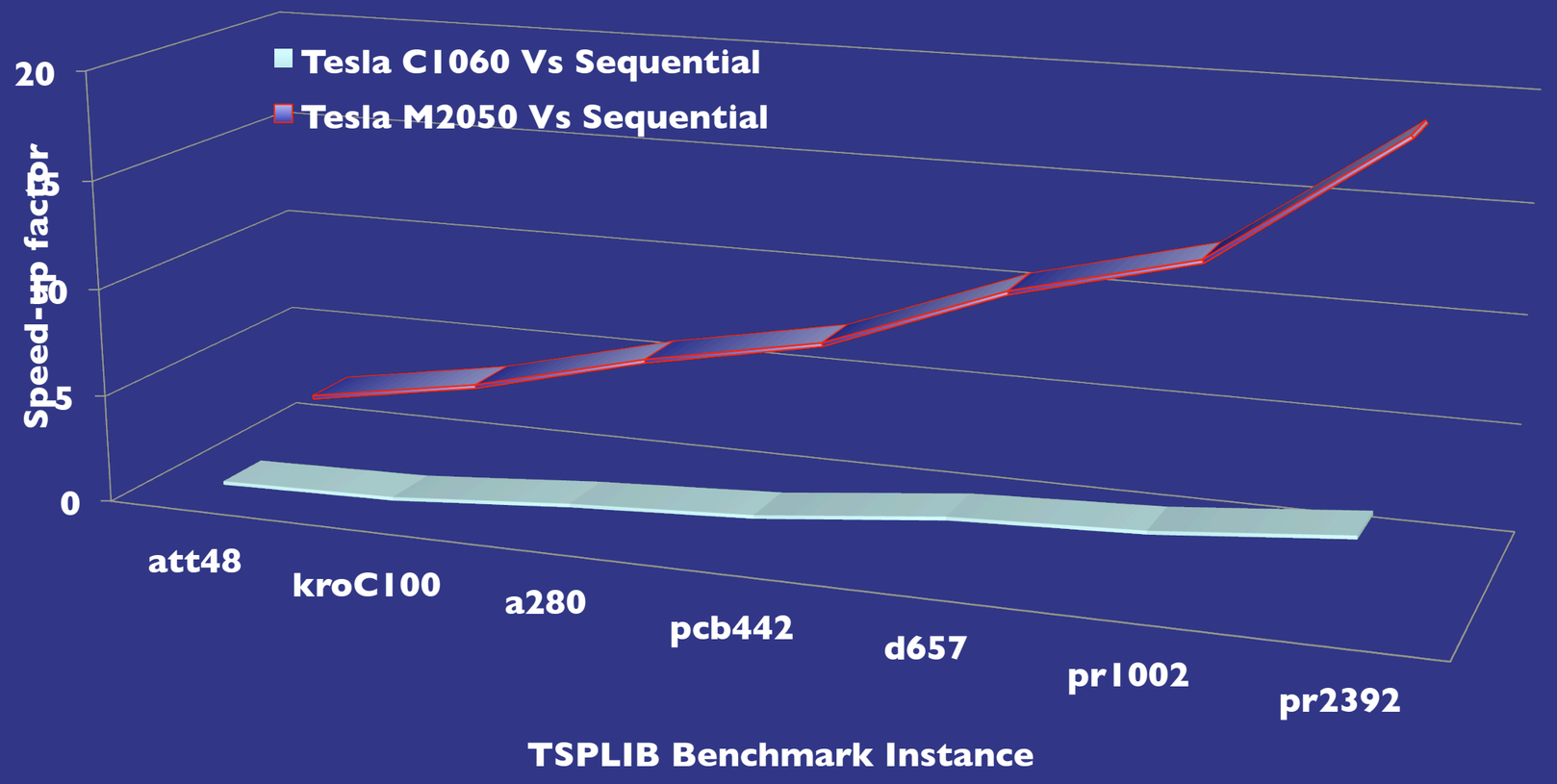




Pheromone update evaluation on the GPU

Code Version	TSPLIB benchmark instance (problem size) on Tesla C1060 (msec.)					
	att48	***kroC100***	***a280***	***pcb442***	***d657***	***pr1002***
Atomic Ins. + Shared	0,15	0,35	1,76	3,45	7,44	17,45
Atomic Ins.	0,16	0,36	1,99	3,74	7,74	18,23
Instruction & Thread Reduction	1,18	3,8	103,77	496,44	2304,54	12345,4
Scatter to Gather + Tiling	1,03	5,83	242,02	1489,88	7092,57	37499,2
Scatter to Gather	2,01	11,3	489,91	3022,85	14460,4	200201
Total Slowdowns	12,73x	31,42x	278,7x	875,29x	1944,23x	11471,59x

Pheromone update evaluation: GPU Vs CPU



Agenda

- Motivation.
- Graphics Processing Units (GPUs).
- Ant Colony Optimization for TSP.
- Parallelization Strategies for ACO on GPUs.
  - Tour Construction.
  - Pheromone Update.
- Performance Evaluation.
- Conclusions and Future Works (2 slides).

Conclusions and Future work

- Ant Colony Optimisation (ACO) is a meta-heuristic that has been successfully applied to many NP-complete problems.
- It is intrinsically parallel, and thus well-suited to implementation on parallel architectures.
- Two main stages; *Tour Construction* and *Pheromone Update*.
- Previous efforts on GPUs focused on *Tour Construction*, using task-based parallelism.
- We demonstrated that this approach does not fit well on the GPU architecture, and provided an alternative approach based on *data parallelism*.

Conclusions and Future work (II)

- We discuss the implementation of the *pheromone update stage* on the GPU, to the best of our knowledge for the first time.
- Possible future directions will will include investigating the effectiveness of GPU-based ACO algorithms on other NP-complete optimisation problems.
- There are many other types of ACO algorithm still to explore, it is a potentially fruitful area of research.
- We hope that this paper stimulates further discussion and work.

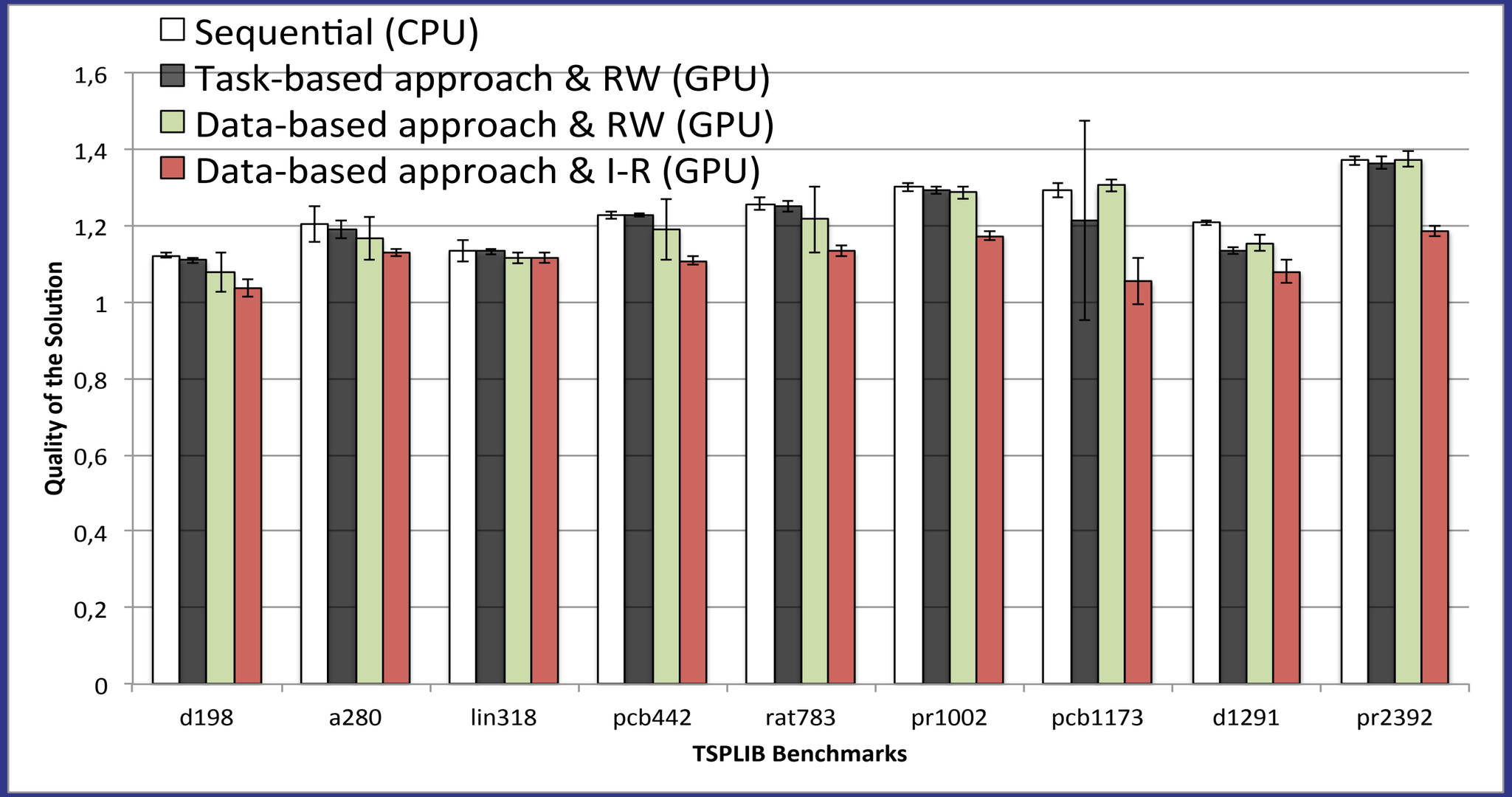
José M. Cecilia, José M. García. University of Murcia (Spain)
Andy Nisbet, Martyn Amos. Manchester Metropolitan University (UK)
Manuel Ujaldón. University of Málaga (Spain)

JGPU 2011.ALICANTE 12 de MAYO 2011

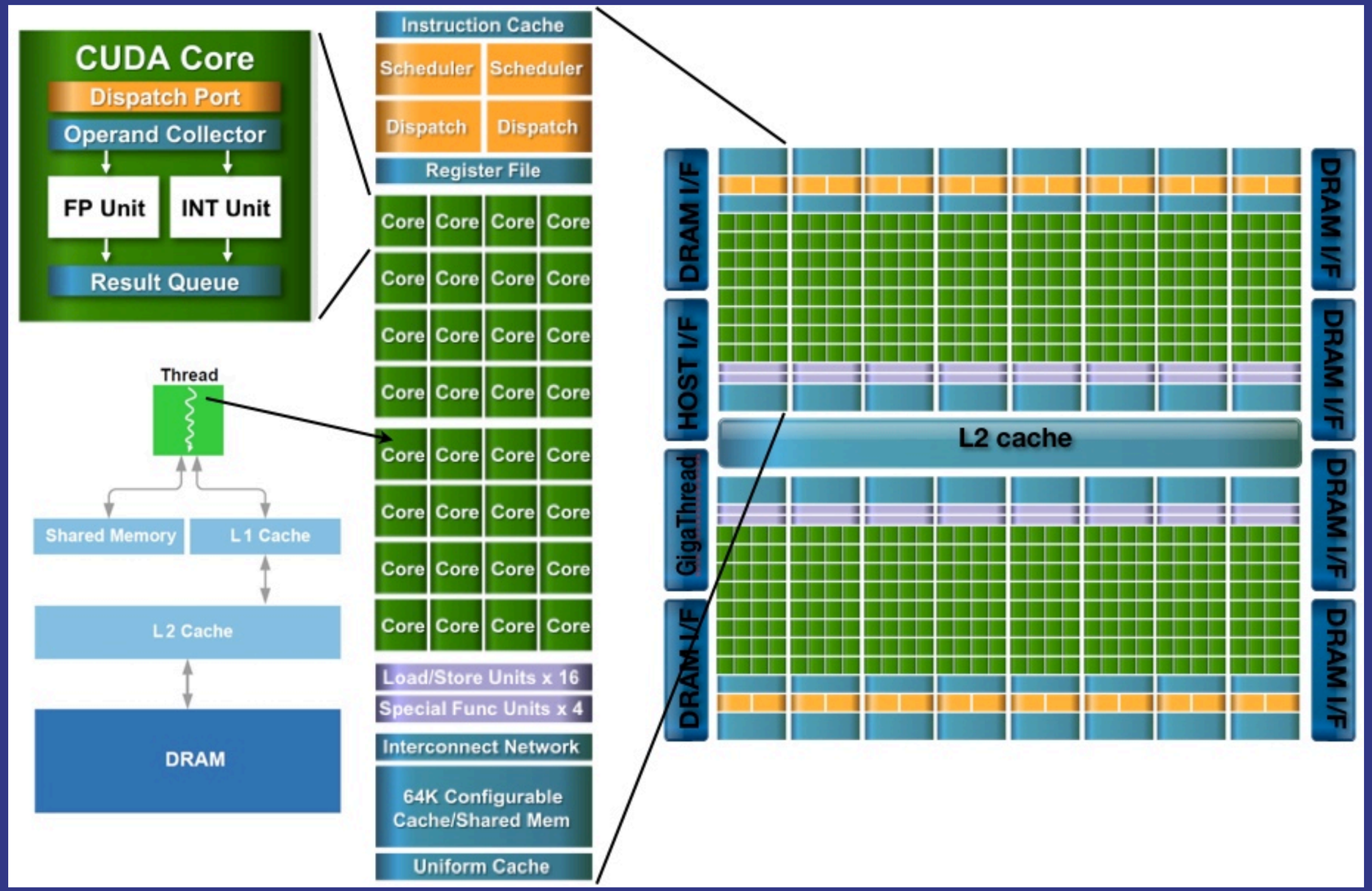
Thanks for you attention
Any Question?

José M. Cecilia
Phd Student at University of Murcia (Spain)
email chema@ditec.um.es
Tél : +34 968 83 57 83
http://skywalker.inf.um.es/~chema

Appendix


Sequential (CPU)
Task-based approach & RW (GPU)
Data-based approach & RW (GPU)
Data-based approach & I-R (GPU)
Quality of the Solution
1,6
1,4
1,2
1
0,8
0,6
0,4
0,2
0
d198
a280
lin318
pcb442
rat783
pr1002
pcb1173
d1291
pr2392
TSPLIB Benchmarks

Appendix


CUDA Core
Dispatch Port
Operand Collector
FP Unit
INT Unit
Result Queue
Instruction Cache
Scheduler
Scheduler
Dispatch
Dispatch
Register File
Core
Load/Store Units x 16
Special Func Units x 4
Interconnect Network
64K Configurable Cache/Shared Mem
Uniform Cache
Thread
Shared Memory
L1 Cache
L2 Cache
DRAM
DRAM I/F
HOST I/F
GigaThread
L2 cache