Cooperative Execution on Heterogeneous Multi-core Systems

Leonel Sousa and Aleksandar Ilić
INESC-ID/IST, TU Lisbon

inesc id lisboa

INSTITUTO SUPERIOR TÉCNICO

Motivation

COMMODITY COMPUTERS = HETEROGENEOUS SYSTEMS

- Multi-core General-Purpose Processors (CPUs)
- Many-core Graphic Processing Units (GPUs)
- …
- Special accelerators, co-processors, FPGAs

=> HUGE COMPUTING POWER

- Not yet completely explored for **COLLABORATIVE COMPUTING**

HETEROGENEITY MAKES PROBLEMS MUCH MORE COMPLEX!

- Performance modeling and load balancing
- Different programming models and languages

Outline

technology
from seed

inesc id lisboa

Collaborative Environment for Heterogeneous Computers

Performance Modeling and Load Balancing

- State of the art for heterogeneous systems
- for CPU+GPU

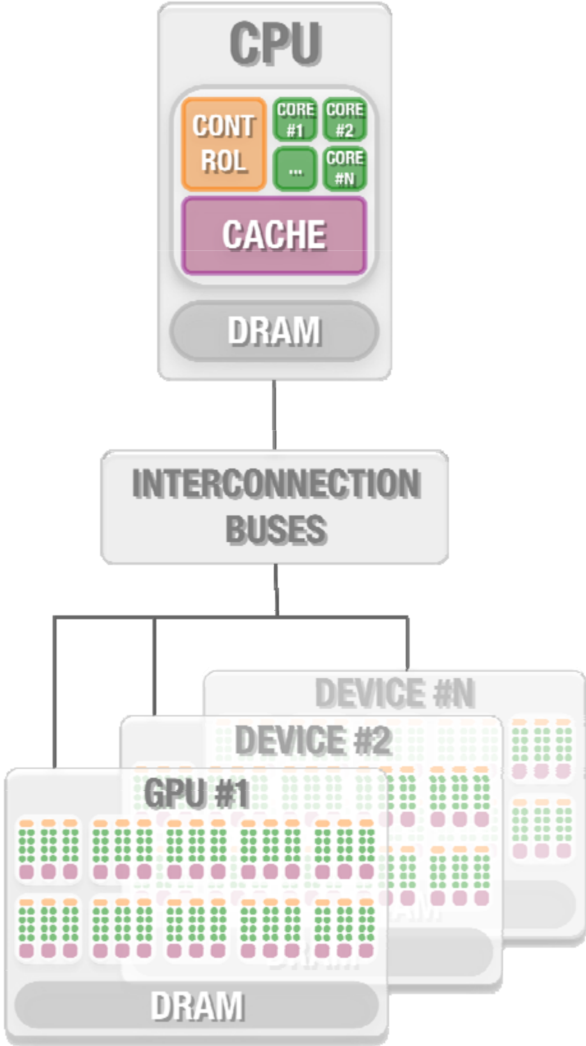
Case Study: 2D Batch Fast Fourier Transform

Conclusions and Future Work

Desktop Heterogeneous Systems

MASTER-SLAVE paradigm

- CPU (Master)
  - Global execution controller
  - Access the whole global memory
- INTERCONNECTION BUSSES
  - Limited and asymmetric communication bandwidth
  - Potential execution bottleneck
- UNDERLYING DEVICES (Slaves)
  - Different architectures and programming models
  - Computation performed using local memories



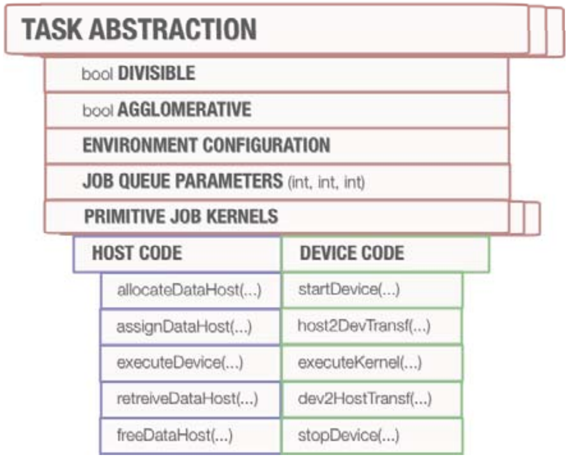
Redefining Tasks and Primitive Jobs

technology
from seed

inesc id lisboa

TASK – basic programming unit (coarser-grained)

– CONFIGURATION PARAMETERS
  – Task: application and task dependency information
  – Environment: device type, number of devices…
– PRIMITIVE JOB WRAPPER
  – DIVISIBLE TASK – comprise several finer-grained Primitive Jobs
  – AGGLOMERATIVE TASK – allows grouping of Primitive Jobs

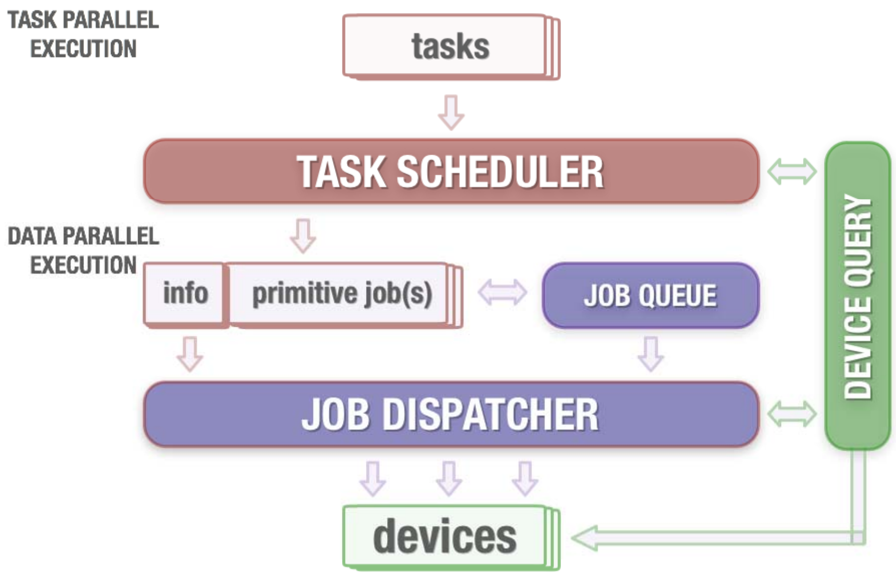


PRIMITIVE JOB – minimal program portion for parallel execution

– CONFIGURATION PARAMETERS
  – I/O and performance specifics, …
– CARRIES PER-DEVICE-TYPE IMPLEMENTATIONS
  – Vendor-specific programming models and tools
  – Specific optimization techniques

Primitive Job Granularity	Task Type	
	Divisible	Agglomerative
Coarser-grained	NO	--
Balanced	YES	NO
Finer/Balanced	YES	YES

Collaborative Execution Environment for Heterogeneous Systems*



Task Type	
Divisible	Agglomerative
NO	--
YES	NO
YES	YES

Task Level Parallelism

– TASK SCHEDULER submits independent tasks to JOB DISPATCHER in respect to task and environment configuration parameters and current platform state from DEVICE QUERY structure

Data Level Parallelism

– PRIMITIVE JOBS may be arranged into JOB QUEUES (currently, 1D-3D grid organization) for DIVISIBLE (AGGLOMERATIVE) TASKS
– JOB DISPATCHER uses DEVICE QUERY and JOB QUEUE information to map (agglomerated) PRIMITIVE JOBS to the requested devices; then initiates and controls further execution;

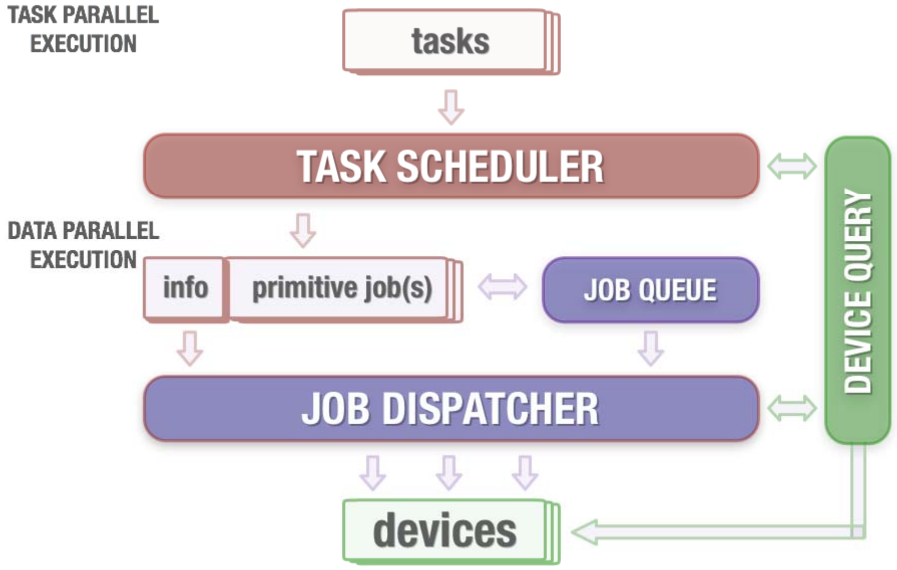
Nested Parallelism

– If provided, JOB DISPATCHER can be configured to perceive certain number of cores of a multi-core device as a single device

**Aleksandar Ilic and Leonel Sousa. "Collaborative Execution Environment for Heterogeneous Parallel Systems", In 12th Workshop on Advances in Parallel and Distributed Computational Models (APDCM/IPDPS 2010), April 2010.*

Collaborative Execution Environment for Heterogeneous Systems*

technology from seed



Task Type	
Divisible	Agglomerative
NO	--
YES	NO
YES	YES

PROBLEM

How to make good DYNAMIC LOAD BALANCING decisions using PERFORMANCE MODELS of the devices aware of :

- application demands
- implementation specifics
- platform / device heterogeneity
- complex memory hierarchies
- limited asymmetric communication bandwidth
- …

Constant Performance Models (CPM)

- Device Performance (Speed) : constant positive number
  - Typically represents relative speed when executing a serial benchmark of a given size
- Computation Distribution : proportional to the speed of device

Functional Performance Models (FPM)

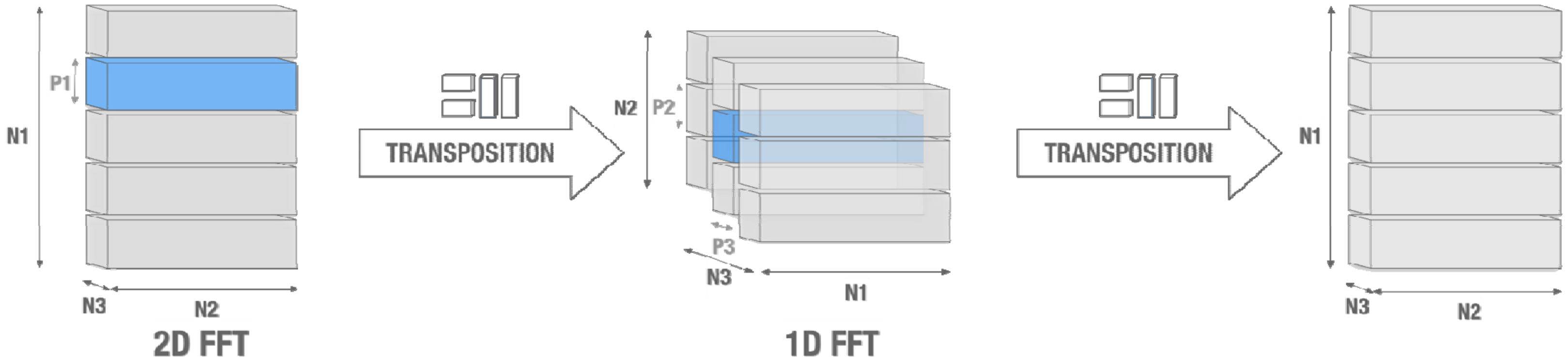
- Device Performance (Speed) : continuous function of the problem size
  - Typically require several benchmark runs and significant amount of time for building it
- Computation Distribution : relies on the "functional speed" of the processor

FPM vs. CPM

- More Realistic : integrates features of heterogeneous processor
  - Processor heterogeneity, the heterogeneity of memory structure, and the other effects (such as paging)
- More Accurate Distribution of computation across heterogeneous devices
- Application-centric approach characterize speed for different applications with different functions

Case Study : 2D FFT Batches

technology from seed



Part of a PARALLEL 3D FFT PROCEDURE : $\mathtt{H} = FFT_{1D}(FFT_{2D}(\mathtt{h}))$

– Very HIGH COMMUNICATION-TO-COMPUTATION ratio

PROBLEM DEFINITION

– $N_{in} = N_{out} = N_1N_2N_3 * \mathtt{sizeof(data)}$

– Performance [FLOPS] : $N_1N_2N_3 * \log(N_1N_2N_3)$

– $N_2 = \mathtt{const};\ N_3 = \mathtt{const};$

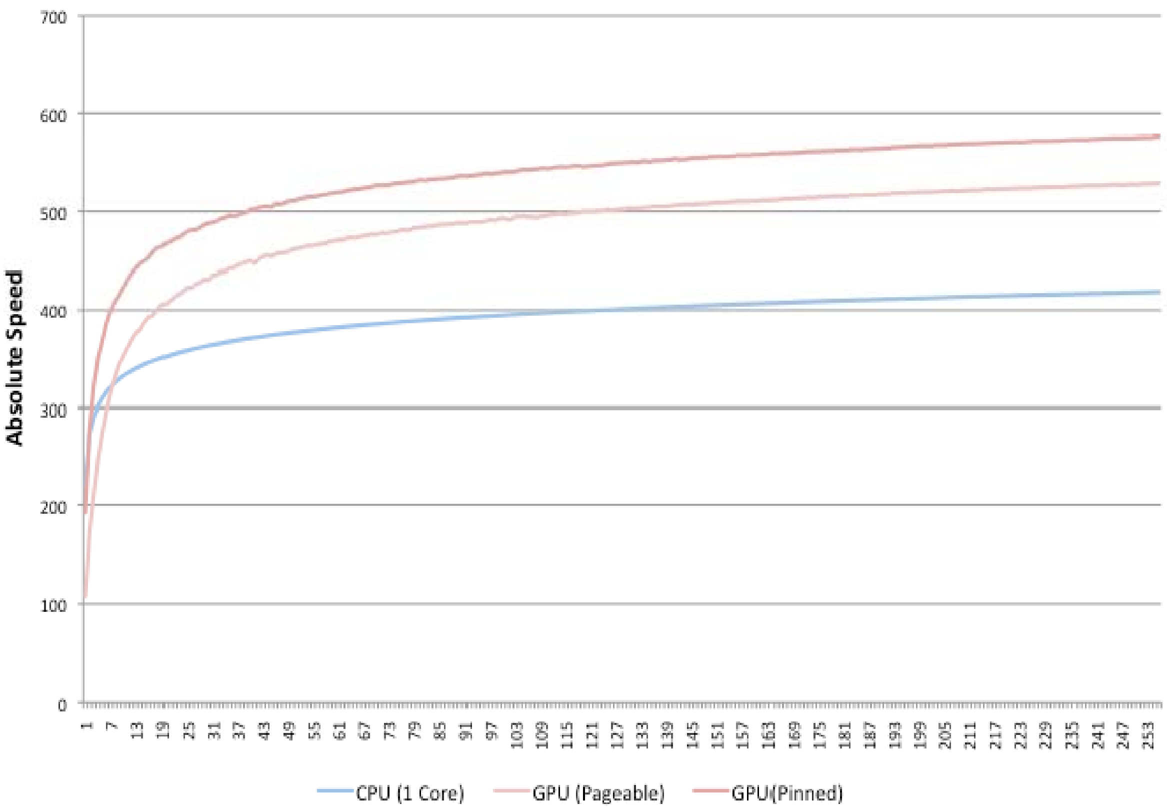
– N_1 – total number of computational chunks (PRIMITIVE JOBS)

Case Study: Performance Modeling
What we already know …* (1)

technology
from seed

inesc id lisboa

Performance Metric → Initialization → Approximation → Iteration



METRIC : ABSOLUTE SPEED

- p devices: $P_1, P_2, \ldots, P_P$
- N_1 total #Primitive Jobs (chunks)
- Device Load [chunks]: $n_1, n_2, \ldots, n_P$
- Absolute speed:

  $s_i(n_i) = n_i/t_i(n_i), \; 1 \le i \le p$

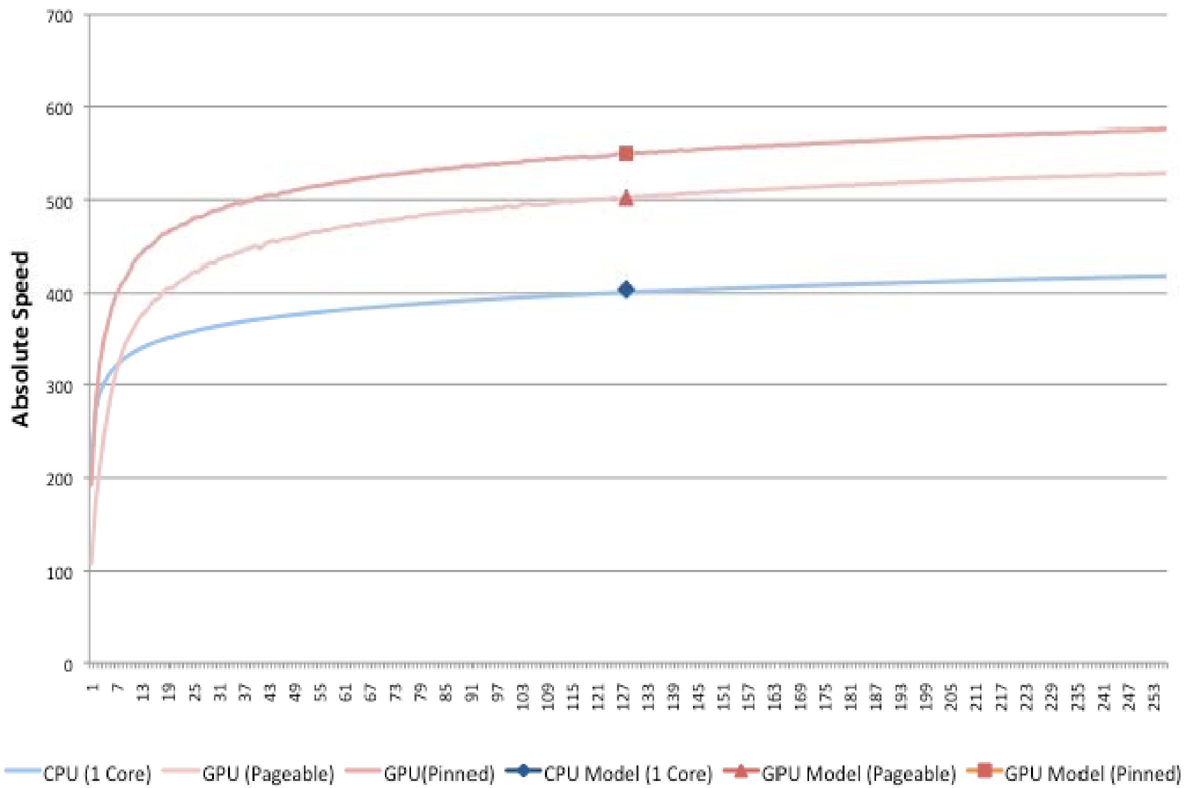
SOLUTION: OPTIMAL LOAD BALANCING

Lies on the straight line that passes through the origin of coordinate system, such that:

$$x_1/s_1(x_1) = x_2/s_2(x_2) = \ldots = x_p/s_p(x_p)$$
$$x_1 + x_2 + \ldots + x_p = N_1$$

Case Study: Performance Modeling
What we already know …* (2)

Performance Metric → **Initialization** → Approximation → Iteration



① All the P computational units execute N_1/p 2D FFT Batches **in parallel**

$$n_i = N_1/p, \quad 1 \le i \le p$$

② Record execution times: $t_i(N_1/p)$

③ IF $\max_{1 \le i,j \le p} \{ ((t_i(N_1/p) - t_j(N_1/p)) / t_i(N_1/p) \} \le \varepsilon$ THEN even distribution solves the problem and the algorithm stops;
ELSE absolute speeds of devices are calculated, such that:

$$s_i(N_1/p) = (N_1/p)/t_i(N_1/p), \quad 1 \le i \le p$$

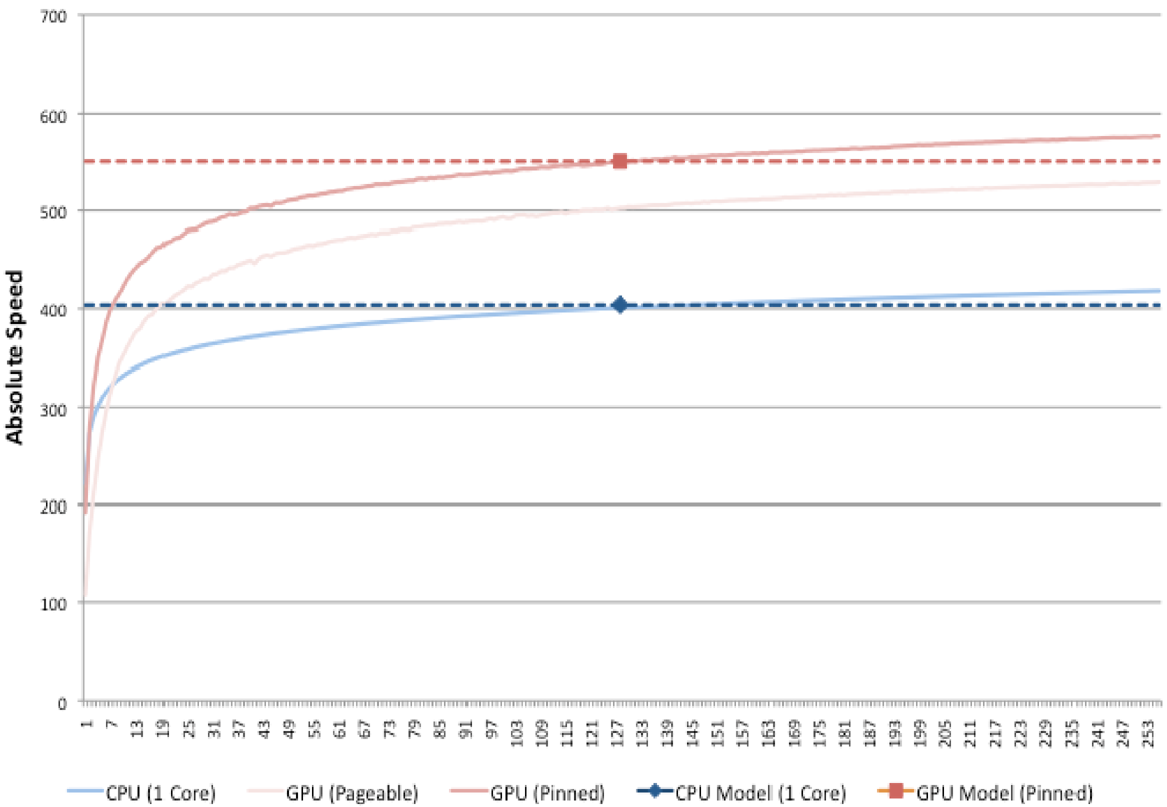
*Lastovetsky, A., and R. Reddy, "Distributed Data Partitioning for Heterogeneous Processors Based on Partial Estimation of their Functional Performance Models", HeteroPar 2009, Netherlands, Lecture Notes in Computer Science, vol. 6043, Springer, pp. 91-101, 25/9/2009, 2010.

Case Study: Performance Modeling
What we already know …* (3)

technology
from seed

inesc id lisboa

Performance Metric → Initialization → **Approximation** → Iteration



① **Performance** of each device is **model**ed as a **constant**

$$s_i(x) = s_i(N_1/p), \quad 1 \le i \le p$$

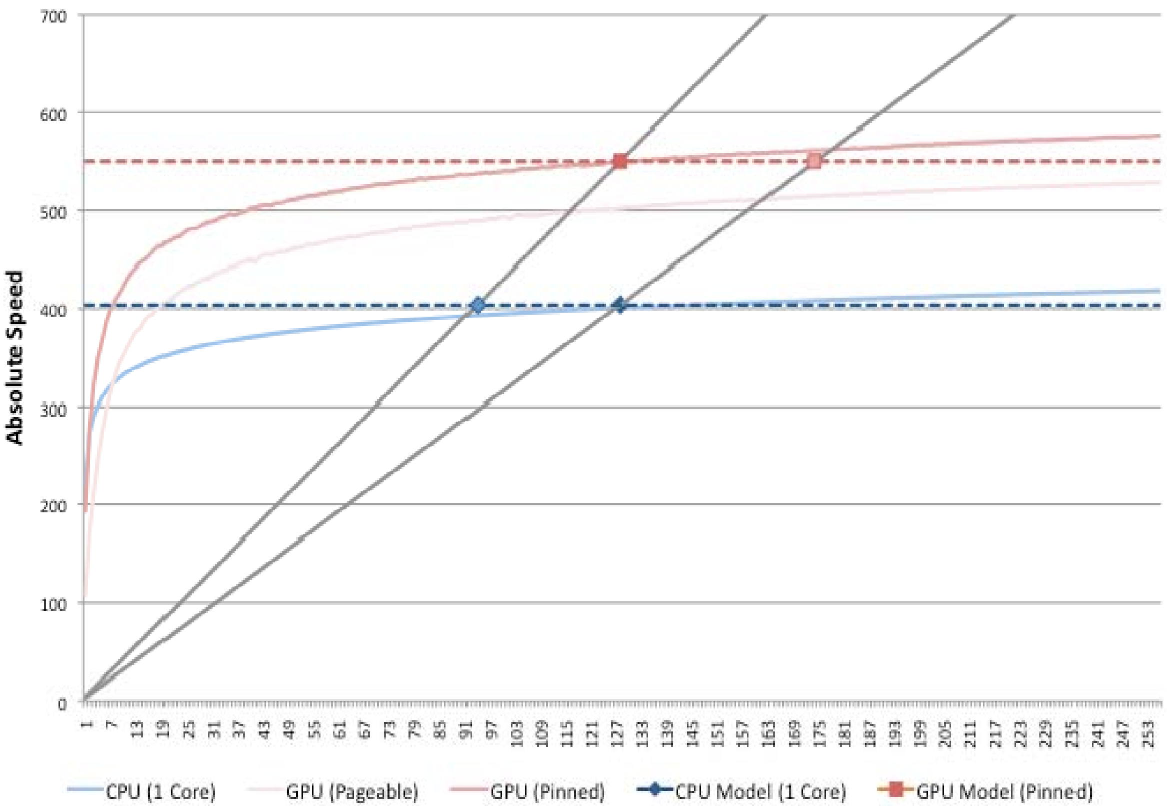
Lastovetsky, A., and R. Reddy, "Distributed Data Partitioning for Heterogeneous Processors Based on Partial Estimation of their Functional Performance Models", HeteroPar 2009, Netherlands, Lecture Notes in Computer Science, vol. 6043, Springer, pp. 91-101, 25/9/2009, 2010.

Case Study: Performance Modeling
What we already know ...** (4)

technology
from seed

inesc id lisboa

Performance Metric → Initialization → Approximation → **Iteration**



① Draw Upper $\mathtt{U}$ and Lower $\mathtt{L}$ lines through the following points:

$$(0,0),\ (N_1/p,\ \max_i\{s_i(N_1/p)\})$$

$$(0,0),\ (N_1/p,\ \min_i\{s_i(N_1/p)\})$$

② Let $x_i^{(U)}$, $x_i^{(L)}$ be the intersections with $s_i(x)$ IF exists $x_i^{(L)} - x_i^{(U)} \geq 1$
THEN go to 3
ELSE go to 5

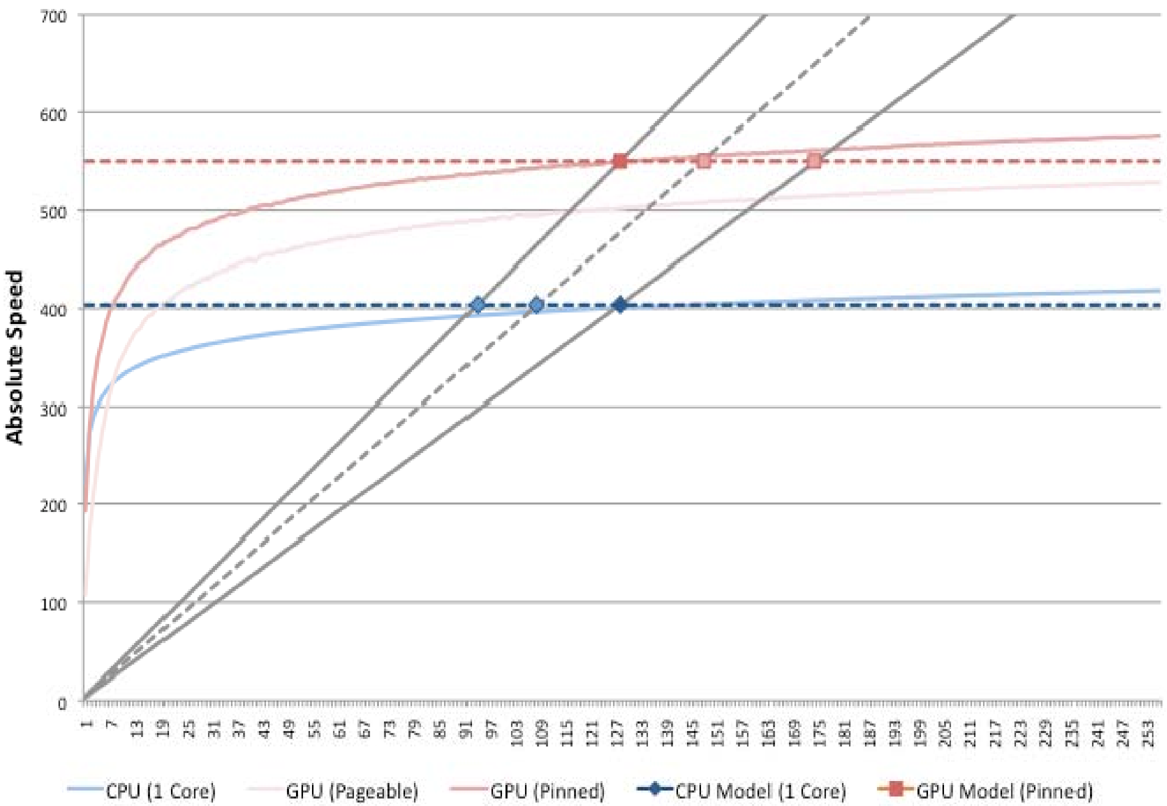
Lastovetsky, A., and R. Reddy, "Data Partitioning with a Functional Performance Model of Heterogeneous Processors", International Journal of High Performance Computing Applications, vol. 21, issue 1: Sage, pp. 76-90, 2007

Case Study: Performance Modeling
What we already know …** (5)

technology
from seed

inesc id lisboa

Performance Metric → Initialization → Approximation → **Iteration**



1 Draw Upper $\mathtt{U}$ and Lower $\mathtt{L}$ lines through the following points:

$$(0,0),\ (N_1/p,\ \max_i\{s_i(N_1/p)\})$$

$$(0,0),\ (N_1/p,\ \min_i\{s_i(N_1/p)\})$$

2 Let $x_i^{(U)}$, $x_i^{(L)}$ be the intersections with $s_i(x)$ IF exists $x_i^{(L)} - x_i^{(U)} \geq 1$
THEN go to 3
ELSE go to 5

③ Bisect the angle between $\mathtt{U}$ and $\mathtt{L}$ by the line $\mathtt{M}$, and calculate intersections $x_i^{(M)}$

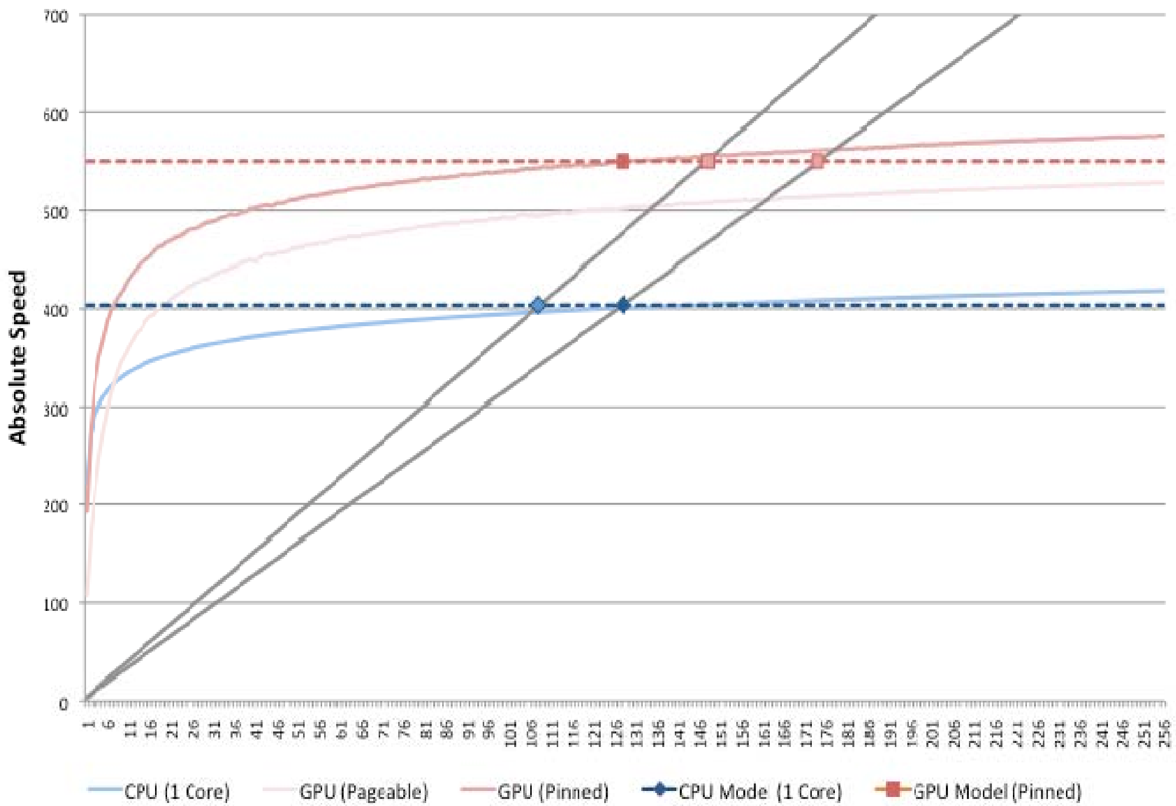
***Lastovetsky, A., and R. Reddy, "Data Partitioning with a Functional Performance Model of Heterogeneous Processors", International Journal of High Performance Computing Applications, vol. 21, issue 1: Sage, pp. 76-90, 2007*

Case Study: Performance Modeling
What we already know …** (6)

technology
from seed

inesc id lisboa

Performance Metric → Initialization → Approximation → **Iteration**



1 Draw Upper U and Lower L lines through the following points:

$(0,0),\ (N_1/p,\ \max_i\{s_i(N_1/p)\})$

$(0,0),\ (N_1/p,\ \min_i\{s_i(N_1/p)\})$

2 Let $x_i^{(U)}$, $x_i^{(L)}$ be the intersections with $s_i(x)$ IF exists $x_i^{(L)} - x_i^{(U)} \geq 1$
THEN go to 3
ELSE go to 5

3 Bisect the angle between U and L by the line M, and calculate intersections $x_i^{(M)}$

④ IF $\Sigma_i\ x_i^{(M)} \leq N_1$
THEN U=M
ELSE L=M
REPEAT 2

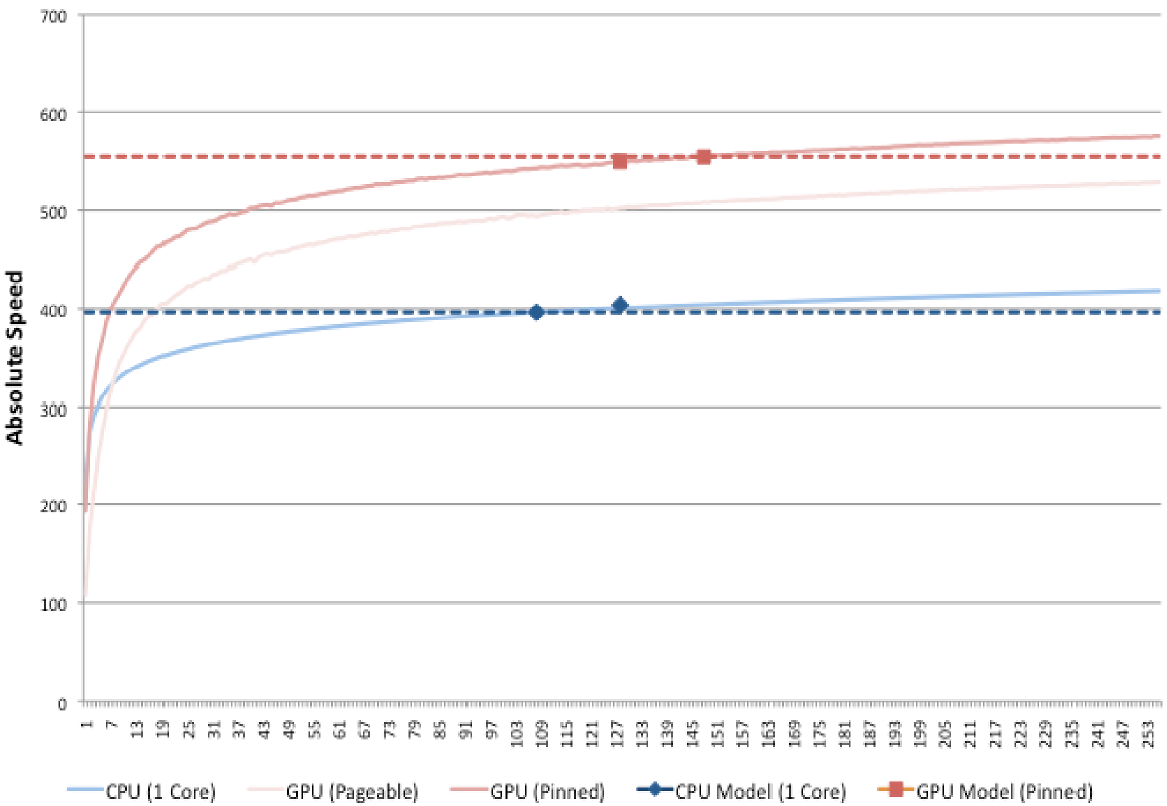
**Lastovetsky, A., and R. Reddy, "Data Partitioning with a Functional Performance Model of Heterogeneous Processors", International Journal of High Performance Computing Applications, vol. 21, issue 1: Sage, pp. 76-90, 2007

Case Study: Performance Modeling
What we already know … (7)

technology
from seed

inesc id lisboa

Performance Metric → Initialization → **Approximation** → Iteration



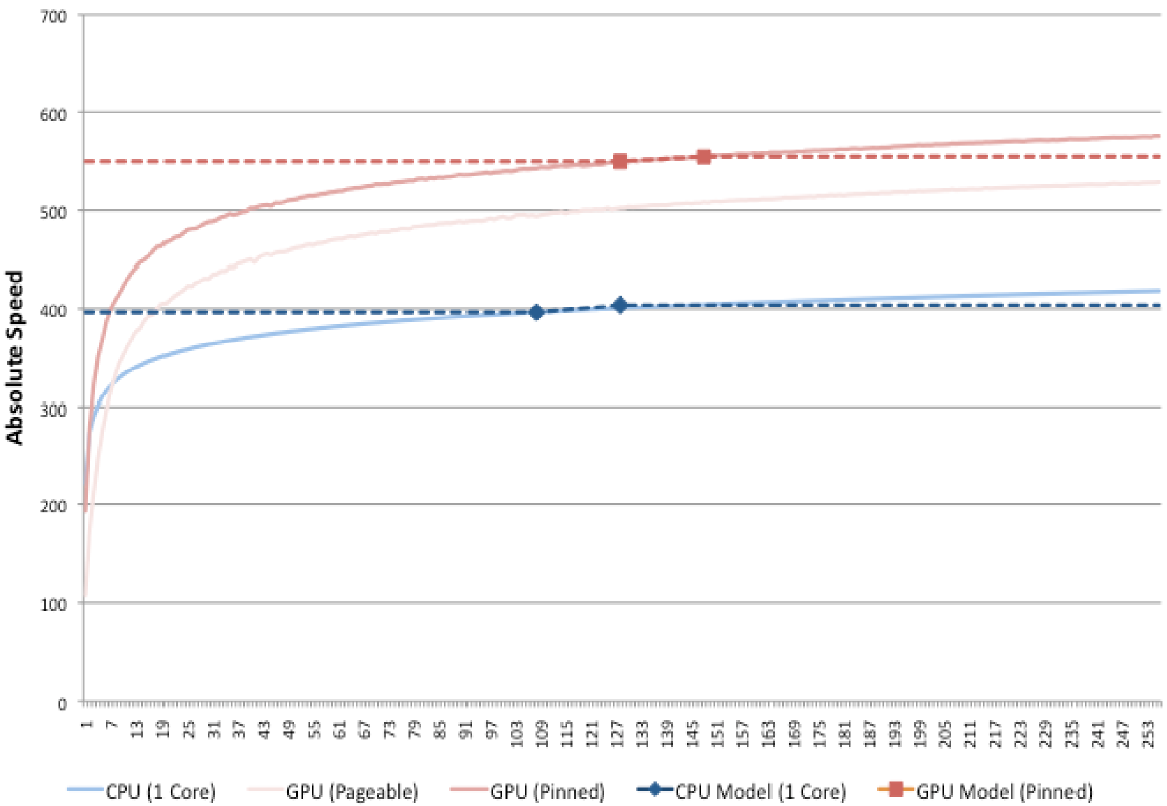
A. Speeds of each device are modeled as **constants** equal to the observed values [1]

$$s_i(x) = s_i(n_i),\ 1 \le i \le p$$

Case Study: Performance Modeling
What we already know … (8)

technology from seed

Performance Metric → Initialization → **Approximation** → Iteration



A. Speeds of each device are modeled as **constants** equal to the observed values [1]

$$s_i(x) = s_i(n_i),\ 1 \le i \le p$$

B. Functional performance modeling using **piecewise linear approximation** [2]

[1] *Galindo I., Almeida F., and Badía-Contelles J.M., "Dynamic Load Balancing on Dedicated Heterogeneous Systems", In EuroPVM/MPI 2008, Springer, pp. 64-74, 2008.*

[2] *Lastovetsky, A., and R. Reddy, "Distributed Data Partitioning for Heterogeneous Processors Based on Partial Estimation of their Functional Performance Models", HeteroPar 2009, vol. 6043, Springer, pp. 91-101, 25/9/2009, 2010*

INSTITUTO SUPERIOR TÉCNICO

Goals in the CPU + GPU Environment

technology
from seed

inesc id lisboa

Online Performance Modeling

– Performance estimation of all heterogeneous devices during the execution
  – No prior knowledge on the performance of an application is available on any of the devices
  – Modeling of the overall CPU and GPU performance for different problem sizes (+ kernel-only GPU performance)

Dynamic Load Balancing

– Optimal distribution of computations (Primitive Jobs)
  – Partial estimations of the performance should be built and used to decide on optimal mapping
  – Returned solution should provide load balancing within a given accuracy

Communication Awareness

– Modeling the bandwidth for interconnection busses during the execution
  – To select problem sizes that maximize the interconnection bandwidth
  – The algorithm should be aware of asymmetric bandwidth for Host-To-Device and Device-To-Host transfers

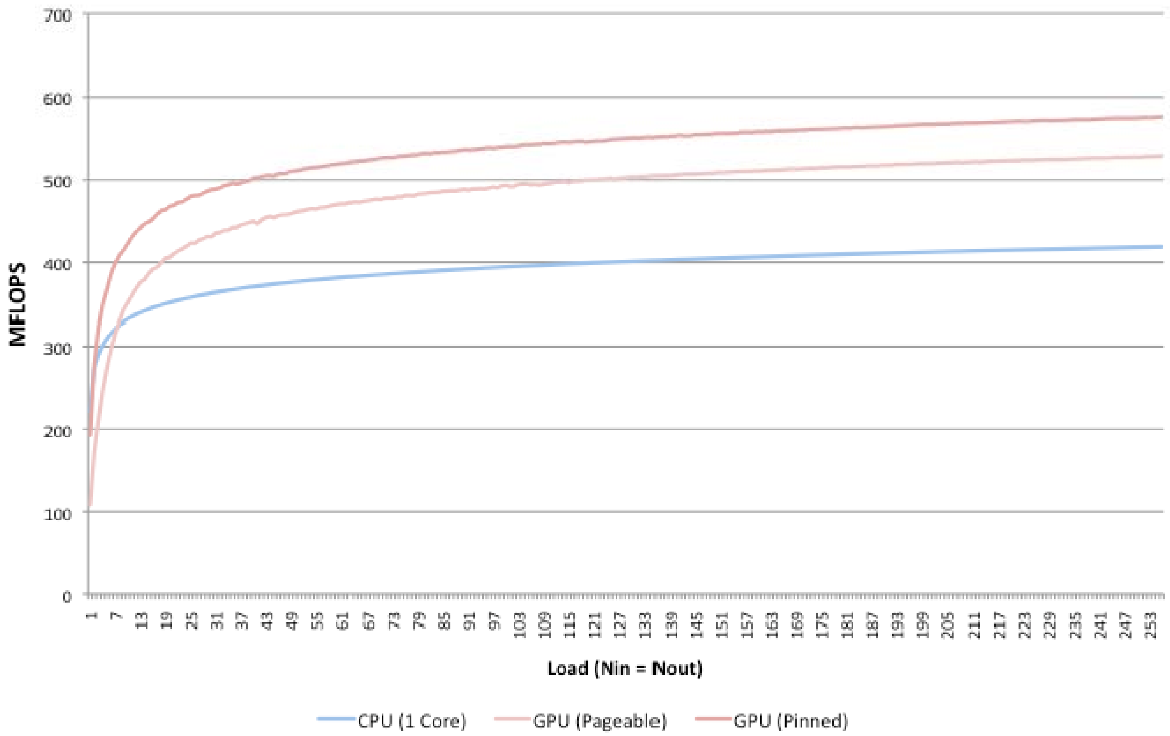
CPU+GPU Architectural Specifics

– Make use of environment-specific functions to ease performance modeling
  – Asynchronous transfers and CUDA streams to overlap communication with computation
  – Be aware of diverse capabilities of different devices, but also for devices of the same type (e.g. GT200 vs. Fermi)

Case Study:
Building Full Performance Models

technology
from seed

inesc id lisboa



Full Performance Models: Per-Device Real Performance

- Experimentally obtained using CPU+ GPU platform specifics (*Pageable/Pinned Memory*)
- Exhaustive search on the full range of problem sizes
- High cost of building it !!!

Experimental Setup	CPU	GPU
	Intel Core 2 Quad	nVIDIA GeForce 285GTX
Speed/Core (GHz)	2.83	1.476
Global Memory (MB)	4096	1024

2D FFT Batch	**CPU**	**GPU**
FFT Type	Complex; Double Precision	
Total Size ($N_1xN_2xN_3$)	256x256x256	
High Performance Software		
FFT	Intel MKL 10.2	CUFFT 3.1

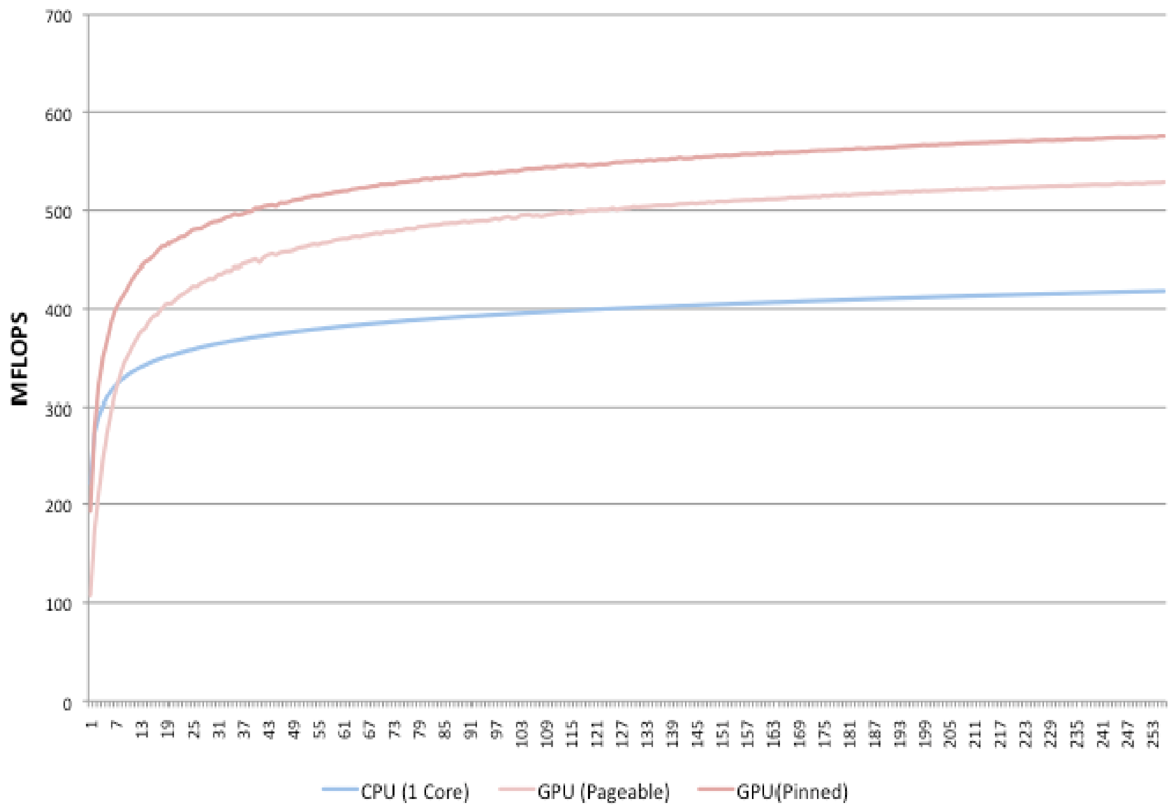
INSTITUTO SUPERIOR TÉCNICO

Case Study: Performance Modeling in CPU + GPU Environment (1)

technology
from seed

inesc id lisboa

Performance Metric → Initialization → Approximation → Iteration



Problem Definition: Primitive Job

- n^{in} – input data requirements
- n^{out} – output data requirements
- n^{size} – problem size
- n^{flops} – #float-point operations

Metric : FLOPS

- p devices: $P_1, P_2, \ldots, P_p$
- N_1 total #Primitive Jobs (chunks)
- Device Load [chunks]: $n_1, n_2, \ldots, n_p$
- $n_i = \{n_i^{in}, n_i^{out}, n_i^{size}, n_i^{flops}\}$

- FLOPS:

$$Perf_i(n_i) = n_i^{flops}/t_i(n_i), \; 1 \le i \le p$$

Solution: Optimal Load Balancing

Lies on the straight line that passes through the origin of coordinate system, such that:

$$x_i^{flops} = nf(x_i), \; x_i \text{ as } n_i^{size}$$

$$nf(x_1)/Perf_1(x_1) = \ldots = nf(x_p)/Perf_p(x_p)$$

$$x_1 + x_2 + \ldots + x_p = N_1$$

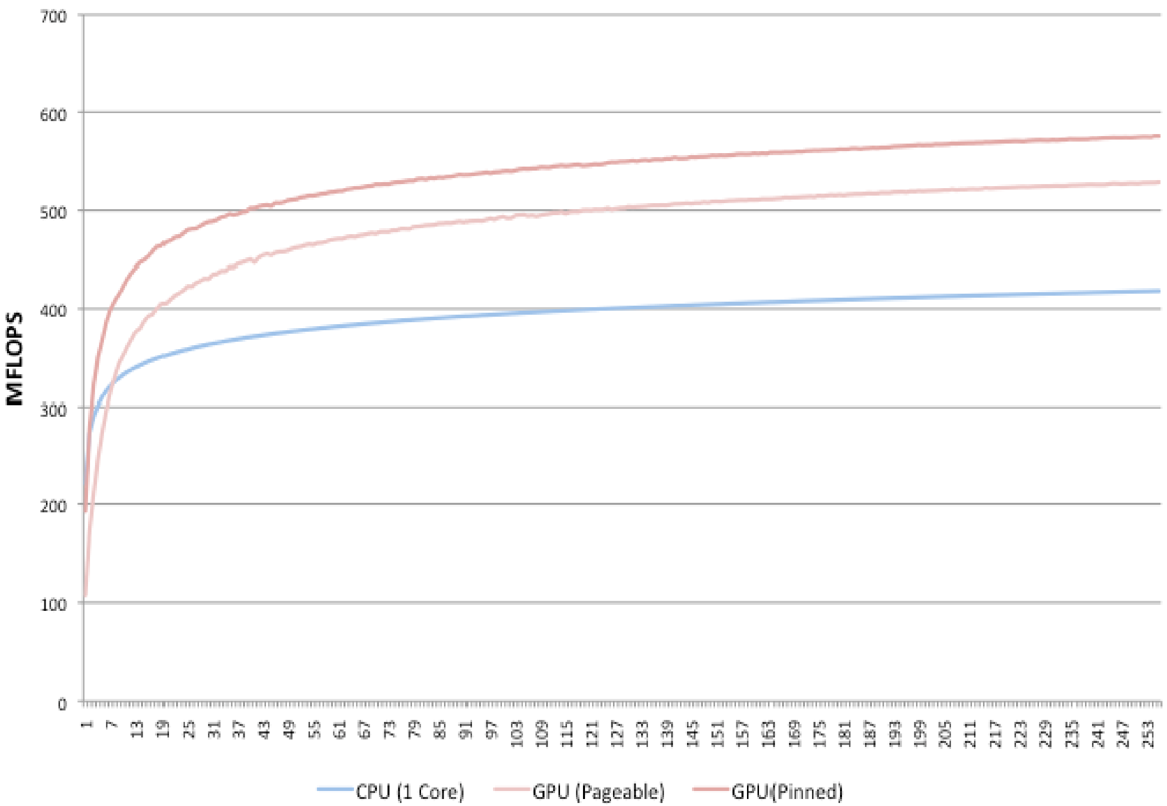
INSTITUTO SUPERIOR TÉCNICO

Case Study: Performance Modeling in CPU + GPU Environment (2)

technology
from seed

inesc id lisboa

Performance Metric → **Initialization** → Approximation → Iteration



① All the P computational units execute N_1/p 2D FFT Batches **in parallel**

$$n_i = N_1/p, \quad 1 \le i \le p$$

② IF (device is GPU) AND (task is Divisible and Agglomerative)
THEN go to 3
ELSE go to 4

③ Split given computational load into streams and use asynchronous transfers to overlap communication with computation

Case Study: Performance Modeling in CPU + GPU Environment (3)

technology from seed

inesc id lisboa

Performance Metric → Initialization → Approximation → Iteration

Streaming

- SUBDIVIDE n_i computational chunks using DIV2 STRATEGY
  - No prior knowledge on the performance of an application!
  - The next stream has half the load of a previous stream
  - Algorithm may continue splitting the workload until the last stream is assigned with load equal to 1
- BANDWIDTH-AWARE DIV2 STRATEGY
  - Interconnection bandwidth is a subject to the amount of data that should be transferred and not to the application-specific demands
  - Run small pre-calibration tests for HOST-TO-DEVICE AND DEVICE-TO-HOST transfers
  - Tests can be stopped when saturation points are detected, or when transfers reach desired value (e.g. 60% of its theoretical)
  - CASE STUDY: $n^{min\_size} = 4$

Stream 0
Stream 1
Stream 2

1 All the P computational units execute N_1/p 2D FFT Batches **in parallel**

$$n_i = N_1/p, \quad 1 \le i \le p$$

2 IF (device is GPU) AND (task is Divisible and Agglomerative)
THEN go to 3
ELSE go to 4

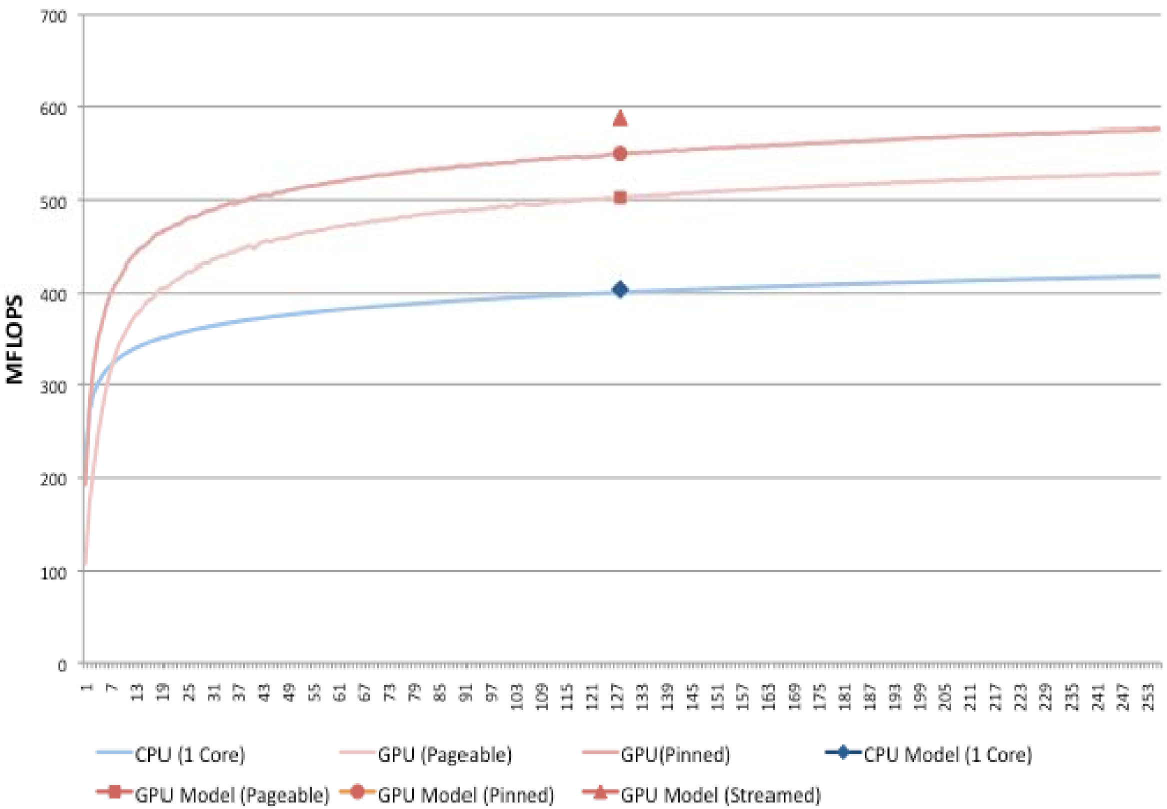
③ Split given computational load into streams and use asynchronous transfers to overlap communication with computation

Case Study: Performance Modeling in CPU + GPU Environment (4)

technology from seed

inesc id lisboa

Performance Metric → **Initialization** → Approximation → Iteration



1 All the P computational units execute N_1/p 2D FFT Batches **in parallel**

$$n_i = N_1/p, \quad 1 \le i \le p$$

2 IF (device is GPU) AND (task is Divisible and Agglomerative)
THEN go to 3
ELSE go to 4

3 Split given computational load into streams and use asynchronous transfers to overlap communication with computation

④ Execute & record execution times: $t_i(N_1/p)$

⑤ IF $\max_{1\le i,j\le p}\{((t_i(N_1/p) - t_j(N_1/p))/t_i(N_1/p)\} \le \varepsilon$
THEN even distribution solves the problem and the algorithm stops;
ELSE performance of devices is calculated, such that:

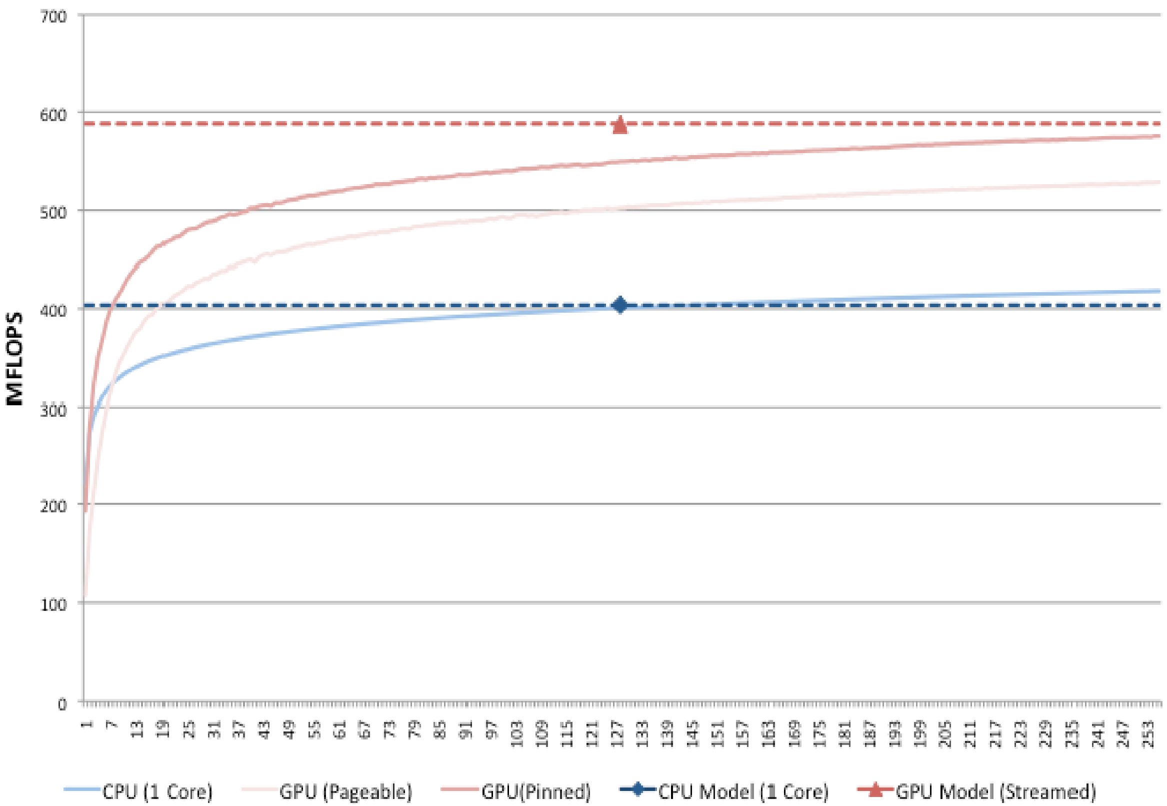
$$Perf_i(N_1/p) = nf(N_1/p)/t_i(N_1/p), \quad 1 \le i \le p$$

INSTITUTO SUPERIOR TÉCNICO

Case Study: Performance Modeling in CPU + GPU Environment (5)

technology
from seed

Performance Metric → Initialization → **Approximation** → Iteration



① Traditional approach: **Performance** of each device is **model**ed as a **constant**

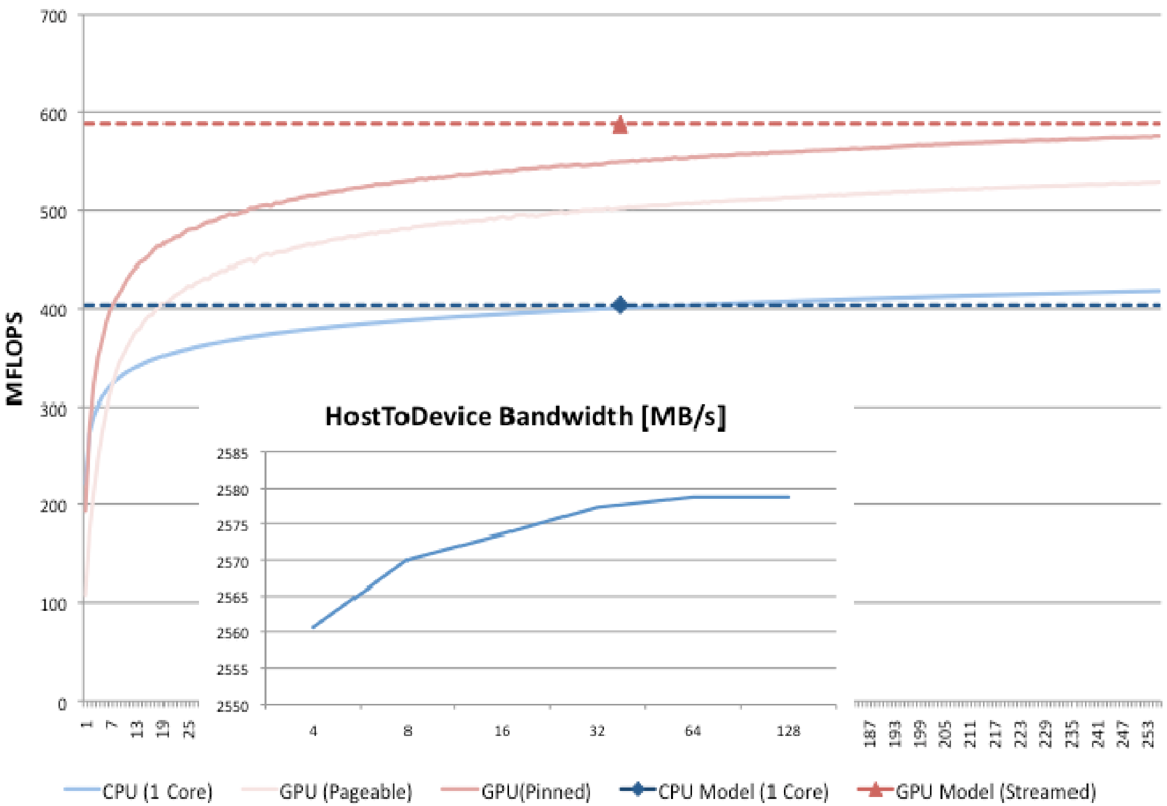
$$Perf_i(x) = Perf_i(N_1/p), \ 1 \le i \le p$$

Case Study: Performance Modeling in CPU + GPU Environment (6)

technology
from seed

inesc id lisboa

Performance Metric → Initialization → **Approximation** → Iteration



1 Traditional approach: **Performance** of each device is **model**ed as a **constant**

$$\mathrm{Perf}_i(x) = \mathrm{Perf}_i(N_1/p),\ 1 \le i \le p$$

② GPU-specific Modeling: Using the obtained values from streaming execution

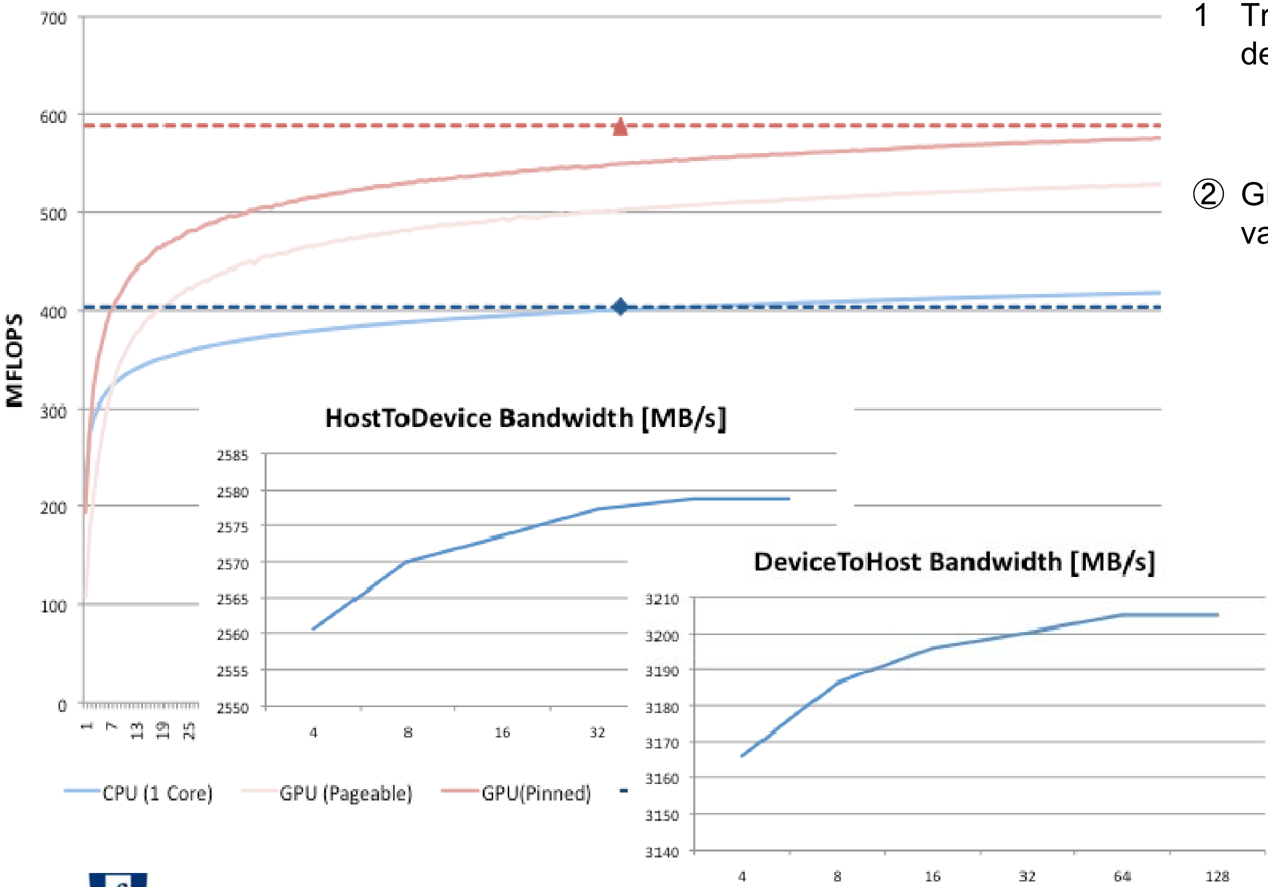
– *HostToDevice* Bandwidth

Case Study: Performance Modeling in CPU + GPU Environment (7)

technology
from seed

inesc id lisboa

Performance Metric → Initialization → **Approximation** → Iteration



1 Traditional approach: **Performance** of each device is **model**ed as a **constant**

$$Perf_i(x) = Perf_i(N_1/p),\ 1 \le i \le p$$

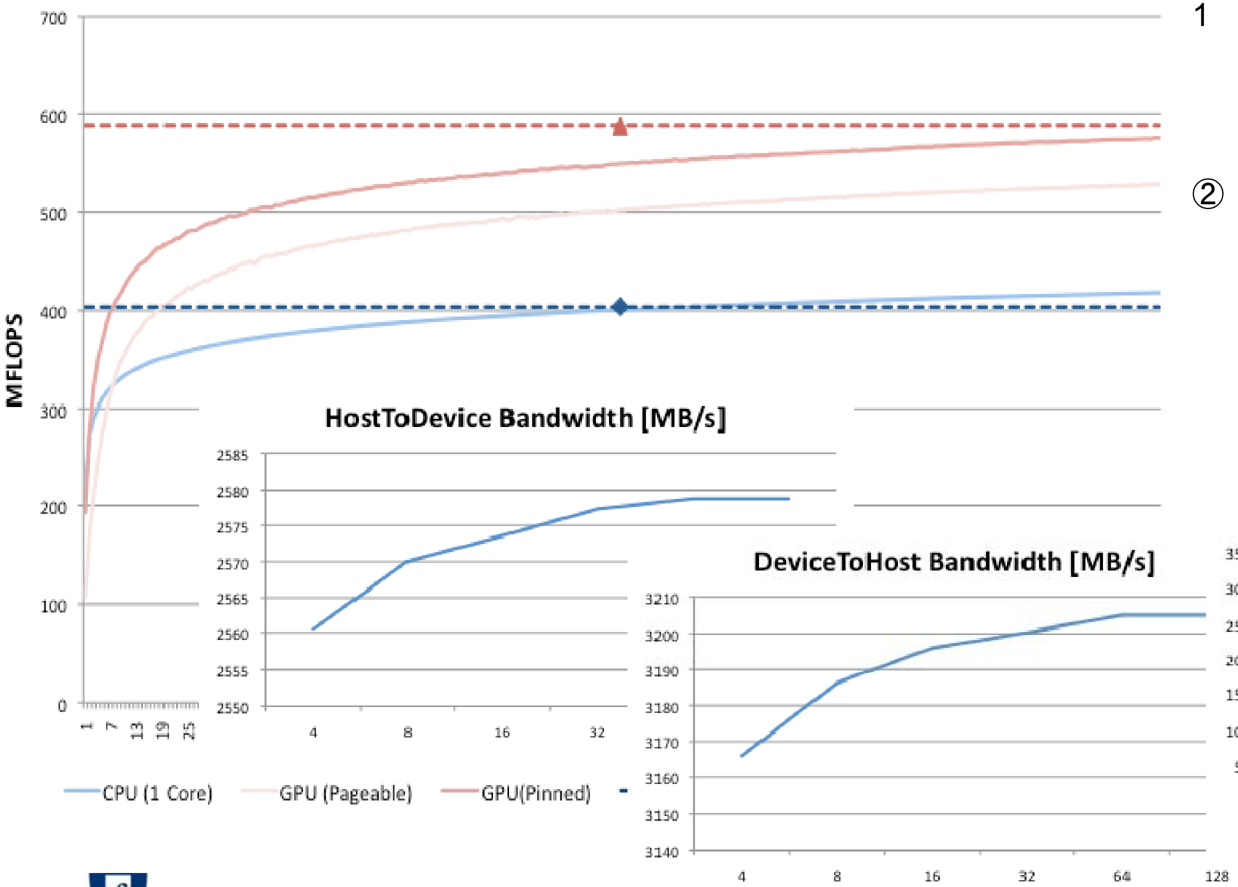
② GPU-specific Modeling: Using the obtained values from streaming execution

- *HostToDevice* Bandwidth
- *DeviceToHost* Bandwidth

Case Study: Performance Modeling in CPU + GPU Environment (8)

technology
from seed

Performance Metric → Initialization → **Approximation** → Iteration

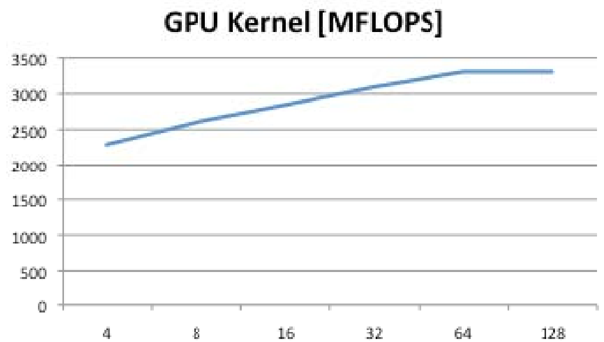


1 Traditional approach: **Performance** of each device is **model**ed as a **constant**

$$Perf_i(x) = Perf_i(N_1/p),\ 1 \le i \le p$$

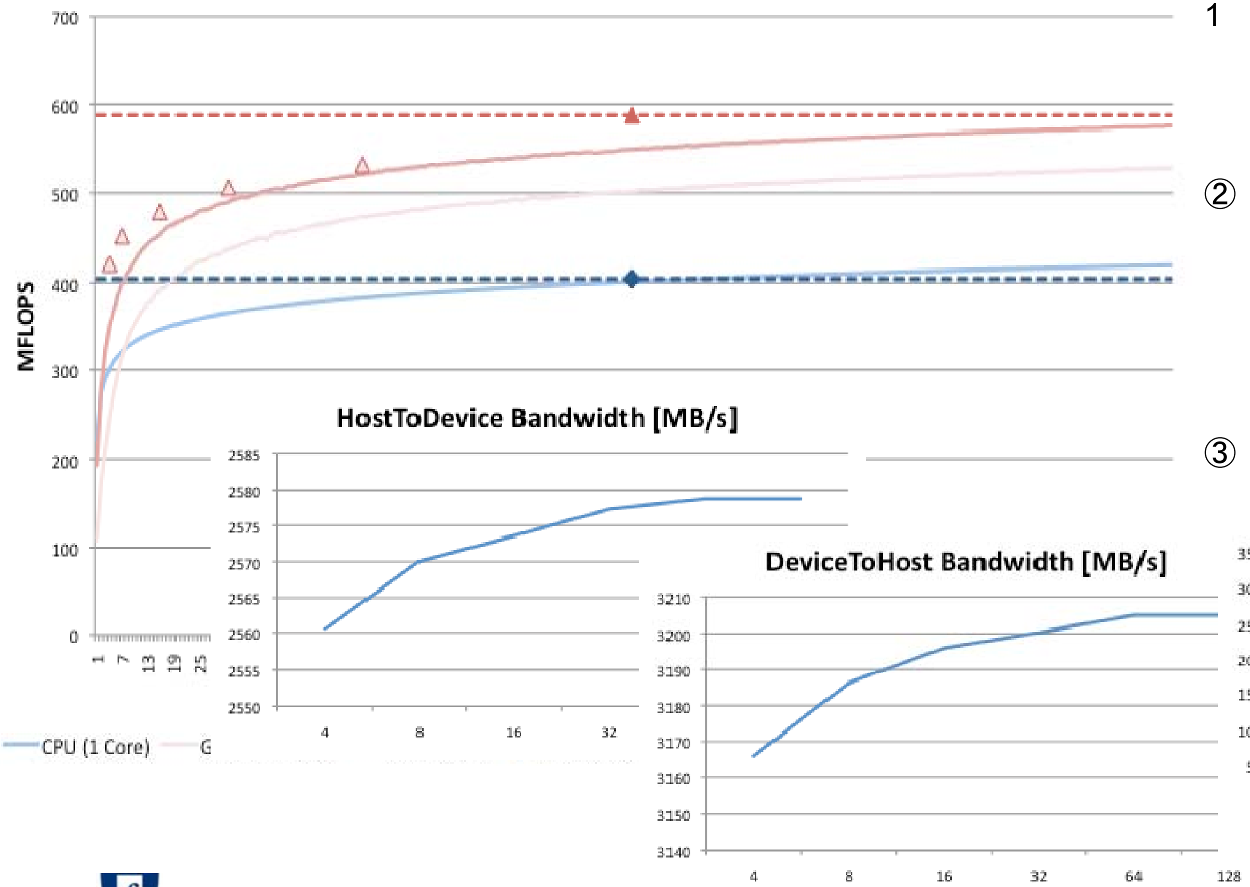
② GPU-specific Modeling: Using the obtained values from streaming execution

- *HostToDevice* Bandwidth
- *DeviceToHost* Bandwidth
- GPU *Kernel* Performance



Case Study: Performance Modeling in CPU + GPU Environment (9)

Performance Metric → Initialization → **Approximation** → Iteration



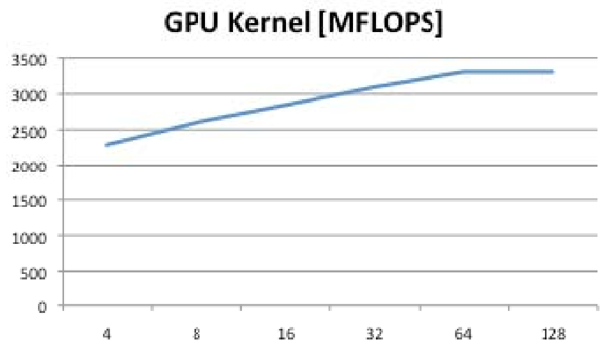
1. Traditional approach: **Performance** of each device is **model**ed as a **constant**

$$Perf_i(x) = Perf_i(N_1/p), \quad 1 \le i \le p$$

② GPU-specific Modeling: Using the obtained values from streaming execution

- *HostToDevice* Bandwidth
- *DeviceToHost* Bandwidth
- GPU *Kernel* Performance

③ Incorporate streaming results

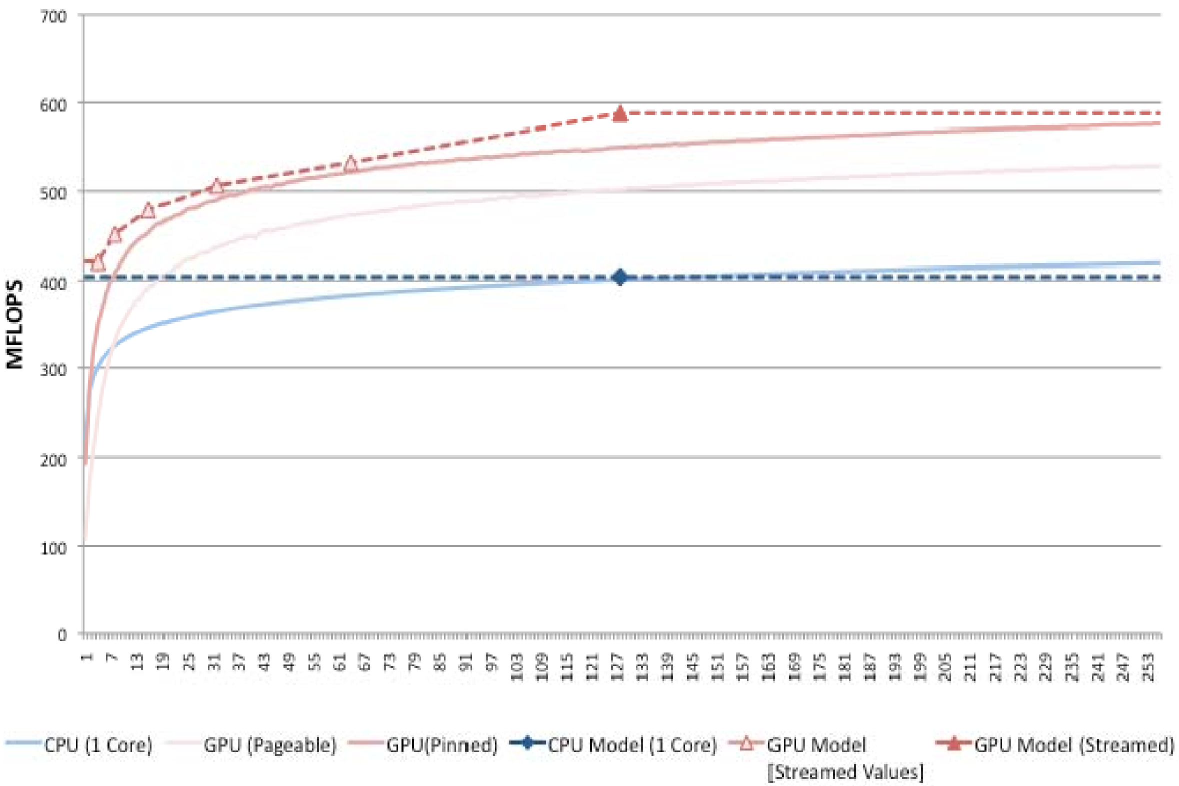


Case Study: Performance Modeling in CPU + GPU Environment (10)

technology
from seed

inesc id lisboa

Performance Metric → Initialization → **Approximation** → Iteration



1 Traditional approach: **Performance** of each device is **model**ed as a **constant**

$$Perf_i(x) = Perf_i(N_1/p),\ 1 \le i \le p$$

2 GPU-specific Modeling: Using the obtained values from streaming execution

- *HostToDevice* Bandwidth
- *DeviceToHost* Bandwidth
- GPU *Kernel* Performance

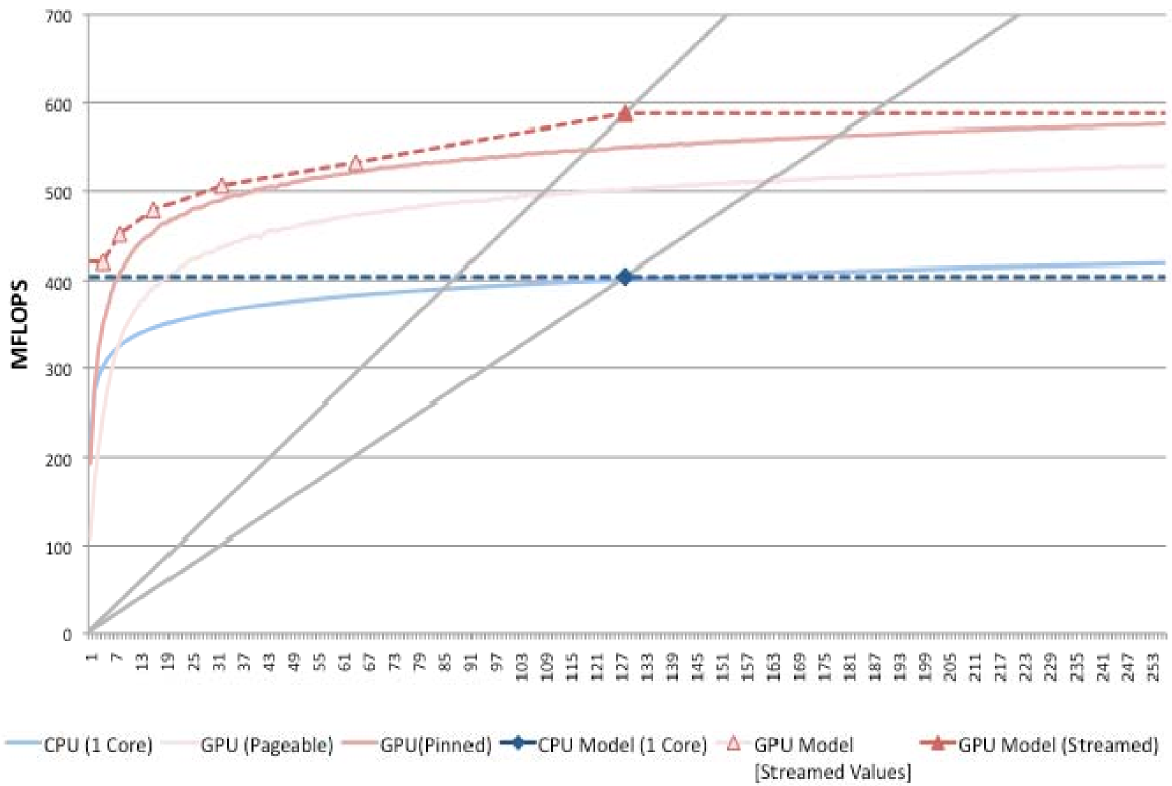
③ Incorporate streaming results

Case Study: Performance Modeling in CPU + GPU Environment (11)

technology from seed

inesc id lisboa

Performance Metric → Initialization → Approximation → **Iteration**



① Draw Upper U and Lower L lines through the following points:

$(0,0),\ (N_1/p,\ \max_i\{Perf_i(N_1/p)\})$

$(0,0),\ (N_1/p,\ \min_i\{Perf_i(N_1/p)\})$

② Let $x_i^{(U)}$ and $x_i^{(L)}$ be the intersections with $Perf_i(x)$

IF exists $x_i^{(L)} - x_i^{(U)} \geq 1$

THEN go to 3

ELSE go to 5

③ Bisect the angle between U and L by the line M, and calculate intersections $x_i^{(M)}$

④ IF $\Sigma_i\ x_i^{(M)} \leq N_1$

THEN U=M

ELSE L=M

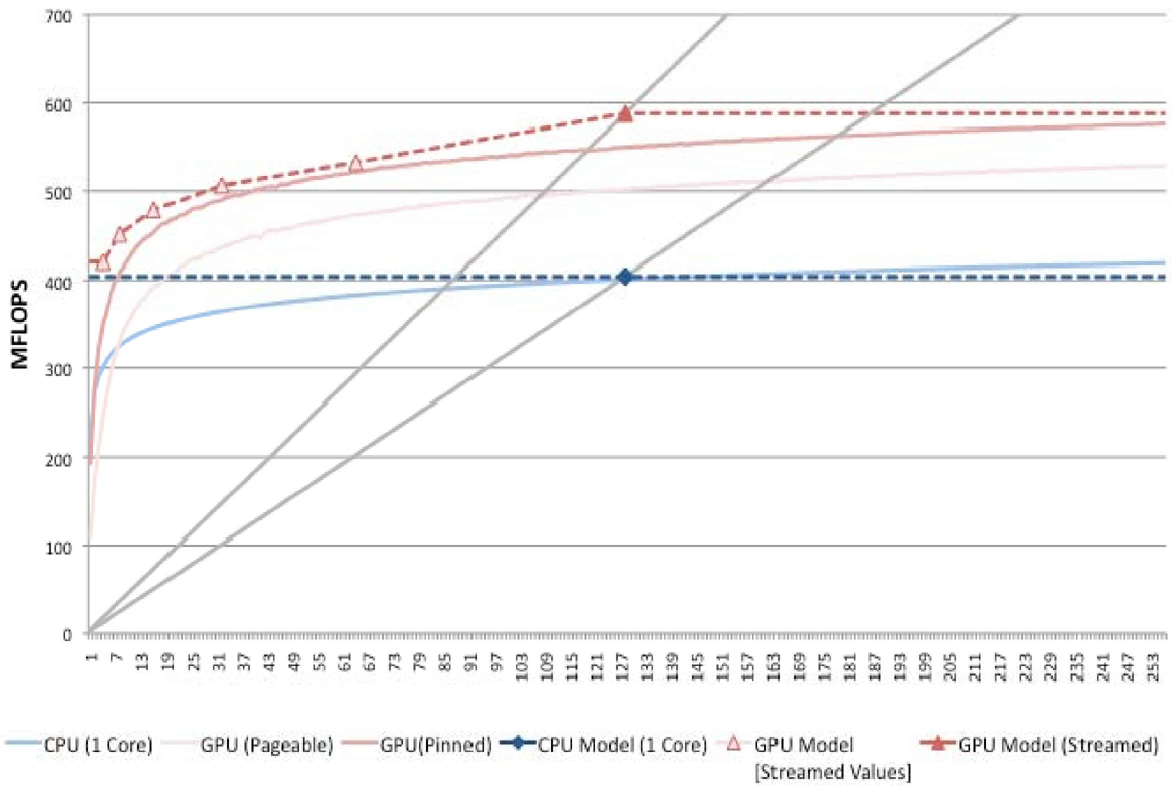
REPEAT 2

Case Study: Performance Modeling in CPU + GPU Environment (12)

technology from seed

inesc id lisboa

Performance Metric → Initialization → Approximation → **Iteration**



1 Draw Upper U and Lower L lines through the following points:

$(0,0)$, $(N_1/p, \max_i\{Perf_i(N_1/p)\})$

$(0,0)$, $(N_1/p, \min_i\{Perf_i(N_1/p)\})$

2 Let $x_i^{(U)}$ and $x_i^{(L)}$ be the intersections with $Perf_i(x)$

IF exists $x_i^{(L)} - x_i^{(U)} \geq 1$

THEN go to 3

ELSE go to 5

3 Bisect the angle between U and L by the line M, and calculate intersections $x_i^{(M)}$

4 IF $\Sigma_i x_i^{(M)} \leq N_1$

THEN U=M

ELSE L=M

REPEAT 2

⑤ Employ **streaming strategy** on the calculated workload value

Case Study: Performance Modeling in CPU + GPU Environment (13)

technology from seed

inesc id lisboa

Performance Metric → Initialization → Approximation → Iteration

Streaming

- STREAMING STRATEGY
  - Results obtained using DIV2 STRATEGY consider application characterization (e.g. communication-to-computation ratio)
  - Workload size for the next stream should be chosen in order to OVERLAP TRANSFERS WITH COMPUTATION in the previous stream
- BANDWIDTH-AWARE STREAMING STRATEGY
  - Reuses the MINIMAL WORKLOAD SIZE FROM DIV2 STRATEGY (obtained via HOST-TO-DEVICE and DEVICE-TO-HOST tests)

    IF $(n^{curr} \geq n^{min\_size})$

    THEN use strategy (cont. overlapping)

    ELSE restart strategy on n^{curr} load
  - In case that load drops under n^{min_size}, strategy is restarted on remaining load

1 Draw Upper U and Lower L lines through the following points:

$(0,0),\ (N_1/p,\ \max_i\{Perf_i(N_1/p)\})$

$(0,0),\ (N_1/p,\ \min_i\{Perf_i(N_1/p)\})$

2 Let $x_i^{(U)}$ and $x_i^{(L)}$ be the intersections with $Perf_i(x)$

IF exists $x_i^{(L)} - x_i^{(U)} \geq 1$

THEN go to 3

ELSE go to 5

3 Bisect the angle between U and L by the line M, and calculate intersections $x_i^{(M)}$

4 IF $\Sigma_i\ x_i^{(M)} \leq N_1$

THEN U=M

ELSE L=M

REPEAT 2

⑤ Employ **streaming strategy** on the calculated workload value

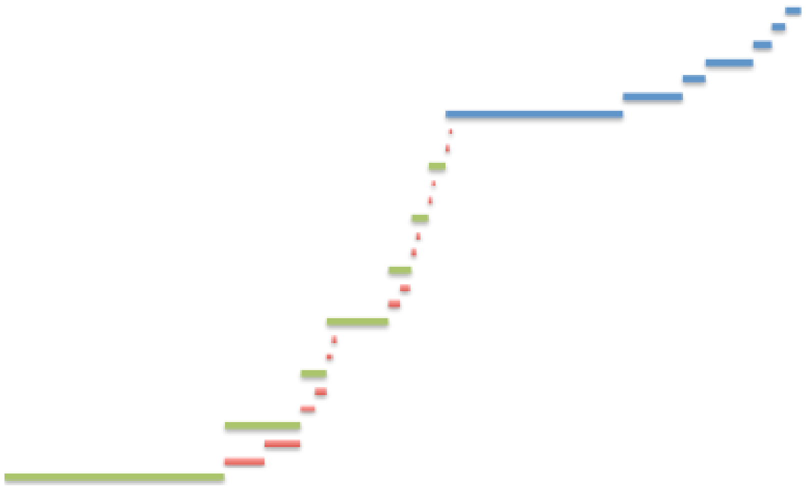
Case Study: Performance Modeling in CPU + GPU Environment (14)

technology
from seed

inesc id lisboa

Performance Metric → Initialization → Approximation → Iteration
Streaming

- CASE STUDY: **2D FFT BATCH**
  - About 83% of total execution time goes on data transfers
    - HOSTTODEVICE Transfers – 46%
    - KERNEL Execution – 17%
    - DEVICETOHOST Transfers – 37%
- BANDWIDTH-AWARE STREAMING STRATEGY
  - $n^{min\_size} = 4$, overlap HOSTTODEVICE transfers and KERNEL execution of two streams



1 Draw Upper U and Lower L lines through the following points:

$(0,0),\ (N_1/p,\ \max_i\{Perf_i(N_1/p)\})$

$(0,0),\ (N_1/p,\ \min_i\{Perf_i(N_1/p)\})$

2 Let $x_i^{(U)}$ and $x_i^{(L)}$ be the intersections with $Perf_i(x)$
IF exists $x_i^{(L)} - x_i^{(U)} \geq 1$
THEN go to 3
ELSE go to 5

3 Bisect the angle between U and L by the line M, and calculate intersections $x_i^{(M)}$

4 IF $\Sigma_i\ x_i^{(M)} \leq N_1$
THEN U=M
ELSE L=M
REPEAT 2

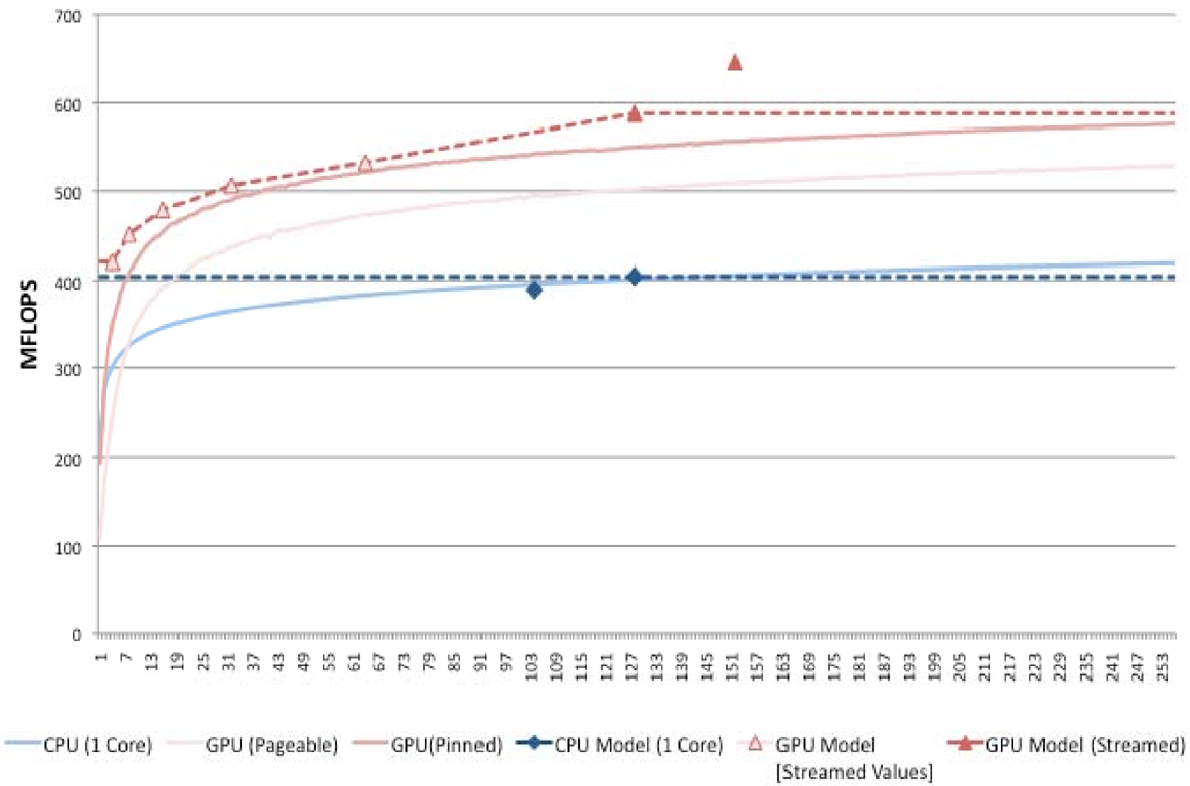
⑤ Employ **streaming strategy** on the calculated workload value

Case Study: Performance Modeling in CPU + GPU Environment (15)

technology
from seed

inesc id lisboa

Performance Metric → Initialization → Approximation → **Iteration**



1 Draw Upper U and Lower L lines through the following points:

$(0,0)$, $(N_1/p, \max_i\{Perf_i(N_1/p)\})$

$(0,0)$, $(N_1/p, \min_i\{Perf_i(N_1/p)\})$

2 Let $x_i^{(U)}$ and $x_i^{(L)}$ be the intersections with $Perf_i(x)$

IF exists $x_i^{(L)} - x_i^{(U)} \geq 1$

THEN go to 3

ELSE go to 5

3 Bisect the angle between U and L by the line M, and calculate intersections $x_i^{(M)}$

4 IF $\Sigma_i\, x_i^{(M)} \leq N_1$

THEN $U=M$

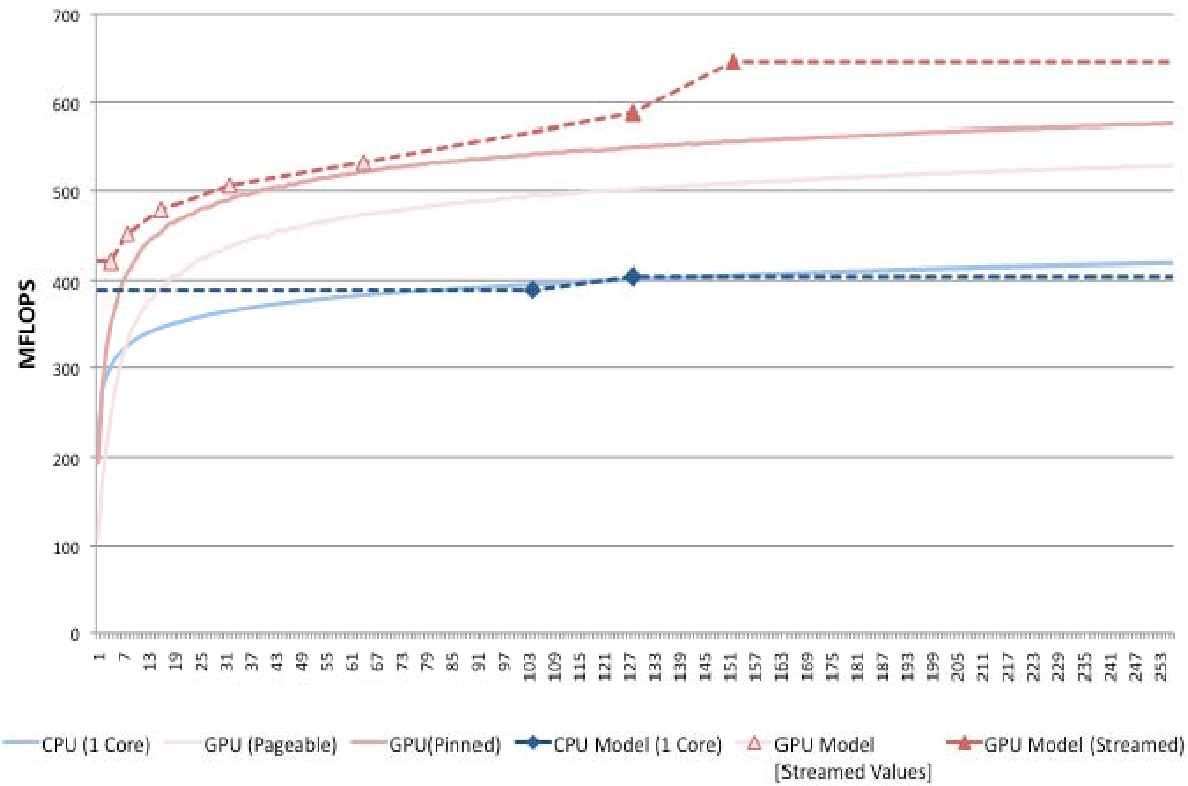
ELSE $L=M$

REPEAT 2

5 Employ streaming strategy on the calculated workload value

Case Study: Performance Modeling in CPU + GPU Environment (16)

Performance Metric → Initialization → **Approximation** → Iteration



① Refine performance models with the newly obtained results

2 GPU-specific Modeling: Using the obtained values from streaming execution

- *HostToDevice* Bandwidth
- *DeviceToHost* Bandwidth
- GPU *Kernel* Performance

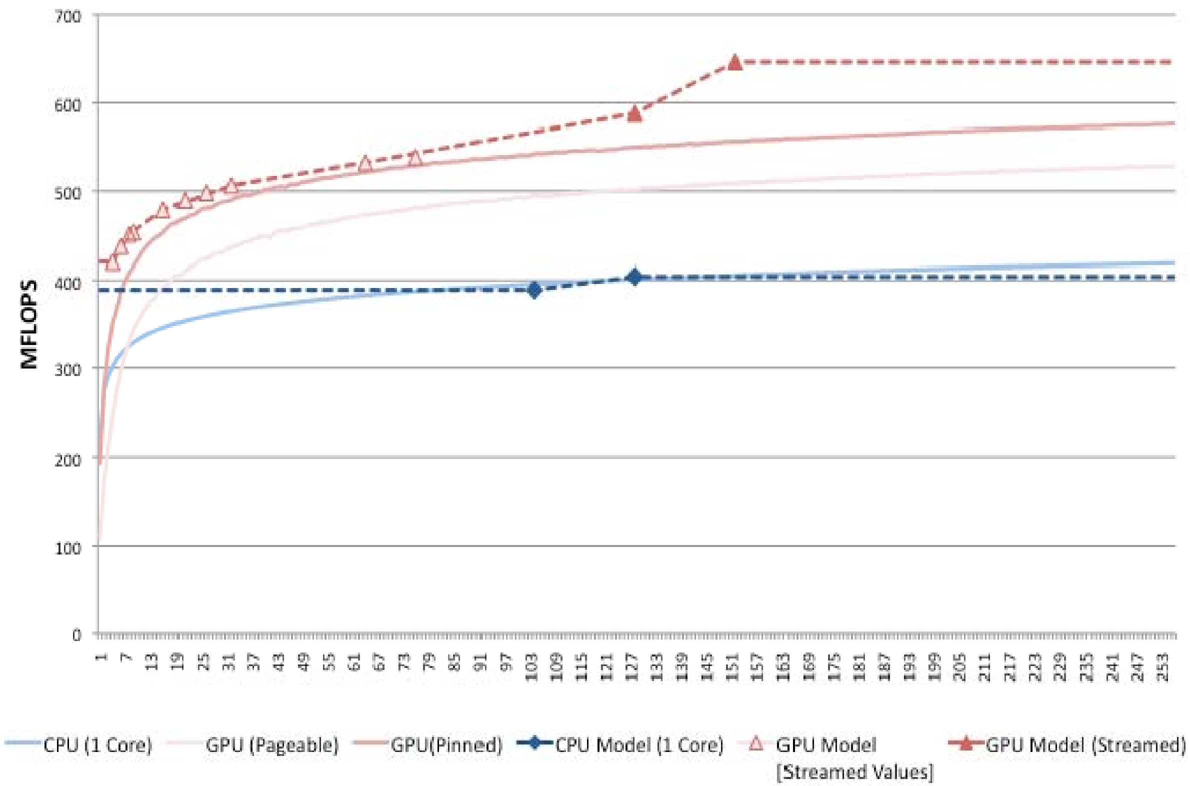
3 Incorporate streaming results

Case Study: Performance Modeling in CPU + GPU Environment (17)

technology from seed

inesc id lisboa

Performance Metric → Initialization → **Approximation** → Iteration



1 Refine performance models with the newly obtained results

② GPU-specific Modeling: Using the obtained values from streaming execution

- *HostToDevice* Bandwidth
- *DeviceToHost* Bandwidth
- GPU *Kernel* Performance

③ Incorporate streaming results

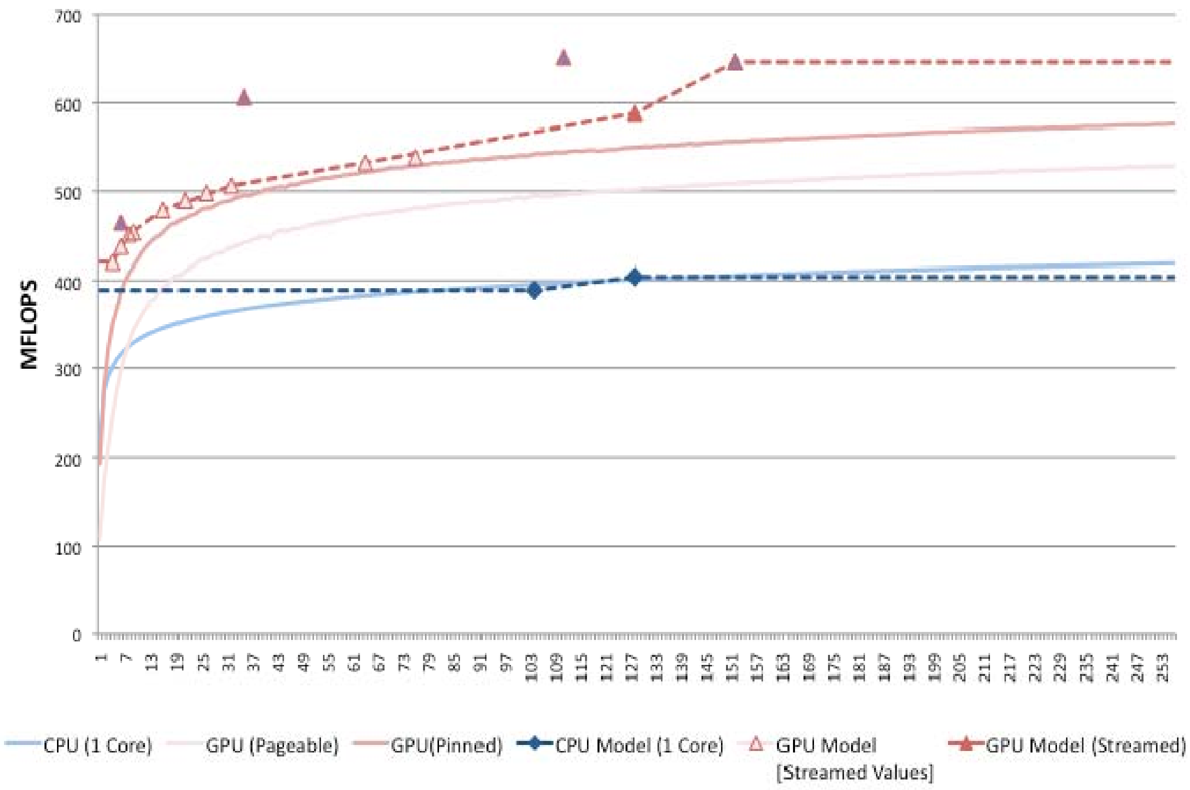
- for each stream

Case Study: Performance Modeling in CPU + GPU Environment (18)

technology from seed

inesc id lisboa

Performance Metric → Initialization → **Approximation** → Iteration



1 Refine performance models with the newly obtained results

② GPU-specific Modeling: Using the obtained values from streaming execution

- *HostToDevice* Bandwidth
- *DeviceToHost* Bandwidth
- GPU *Kernel* Performance

③ Incorporate streaming results

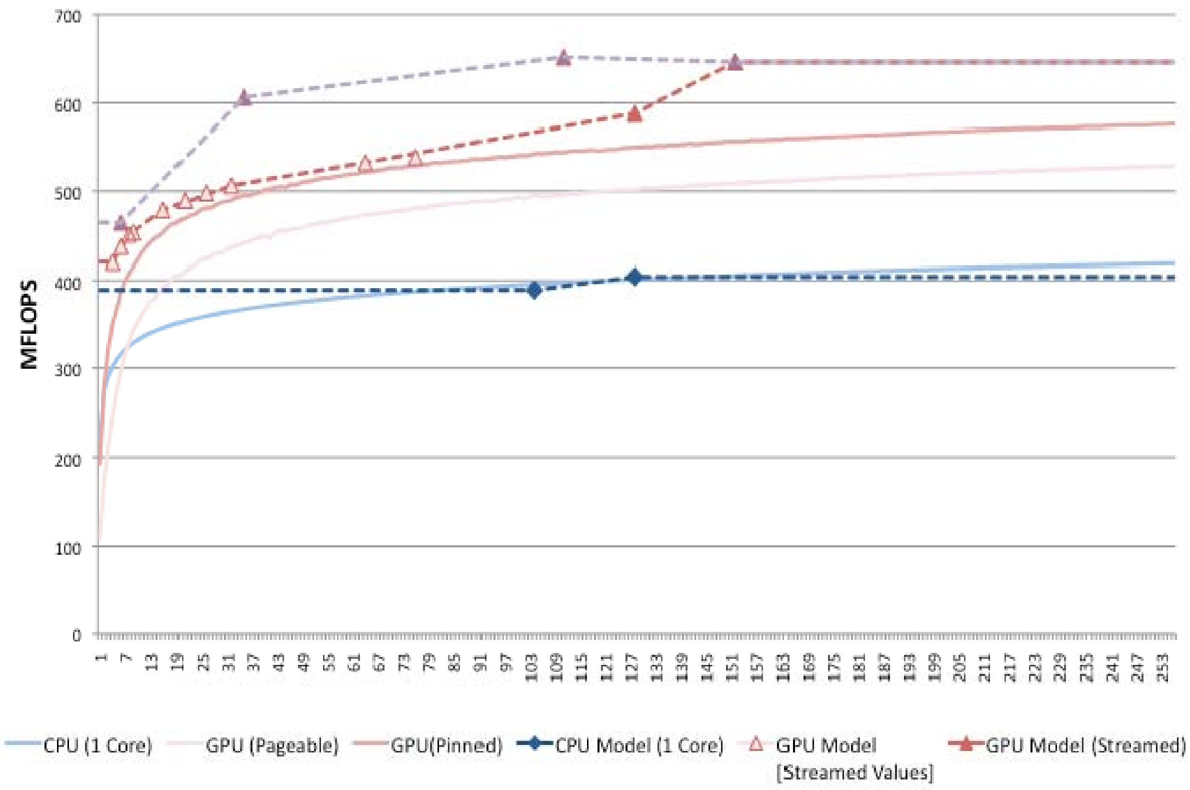
- for each stream
- for each stream restart

Case Study: Performance Modeling in CPU + GPU Environment (19)

technology from seed

inesc id lisboa

Performance Metric → Initialization → **Approximation** → Iteration



1 Refine performance models with the newly obtained results

② GPU-specific Modeling: Using the obtained values from streaming execution

- *HostToDevice* Bandwidth
- *DeviceToHost* Bandwidth
- GPU *Kernel* Performance

③ Incorporate streaming results

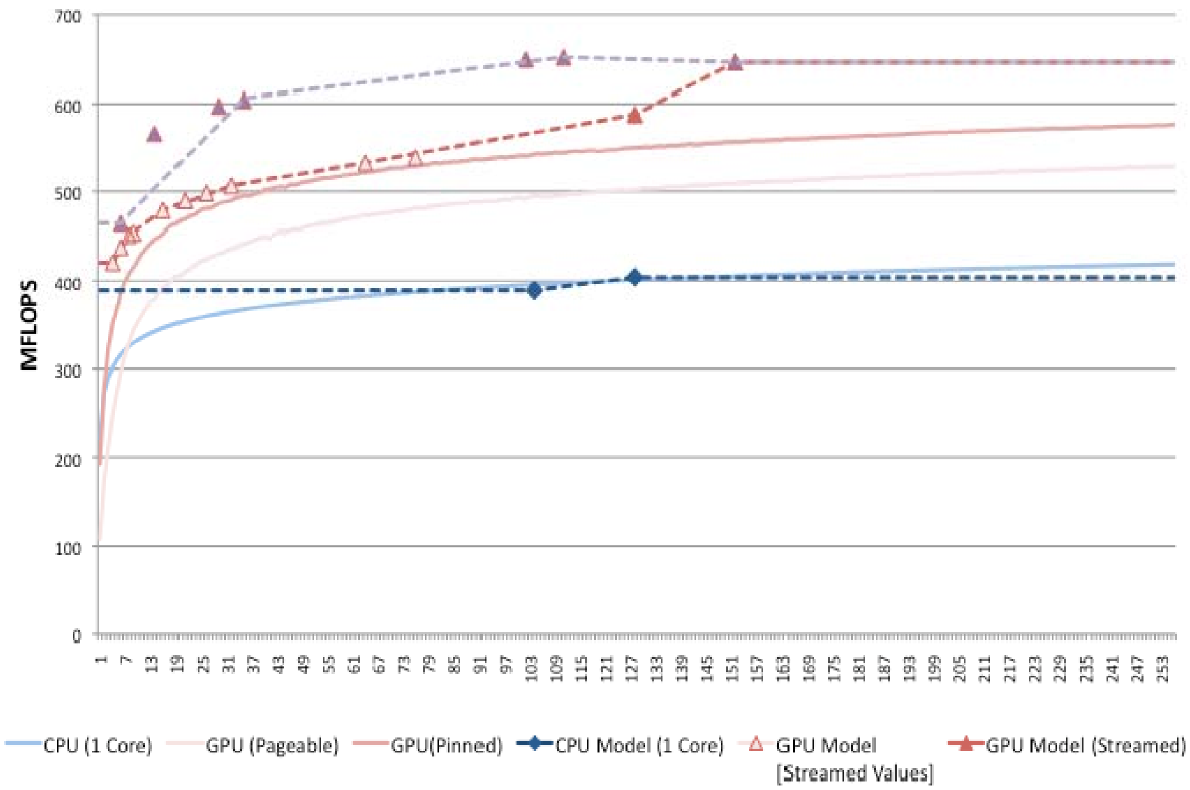
- for each stream
- for each stream restart

Case Study: Performance Modeling in CPU + GPU Environment (20)

technology
from seed

inesc id lisboa

Performance Metric → Initialization → **Approximation** → Iteration



1 Refine performance models with the newly obtained results

② GPU-specific Modeling: Using the obtained values from streaming execution

- *HostToDevice* Bandwidth
- *DeviceToHost* Bandwidth
- GPU *Kernel* Performance

③ Incorporate streaming results

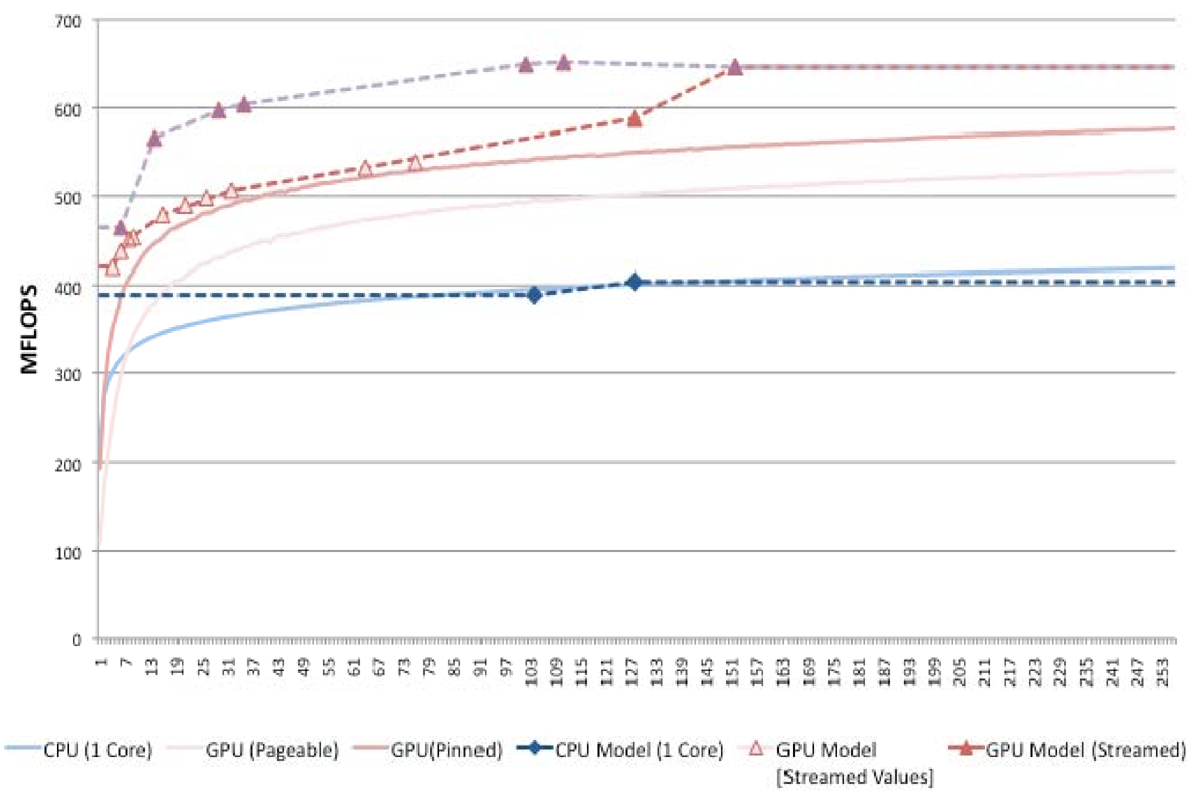
- for each stream
- for each stream restart
- for every stream combination

Case Study: Performance Modeling in CPU + GPU Environment (21)

technology from seed

inesc id lisboa

Performance Metric → Initialization → **Approximation** → Iteration



1 Refine performance models with the newly obtained results

② GPU-specific Modeling: Using the obtained values from streaming execution

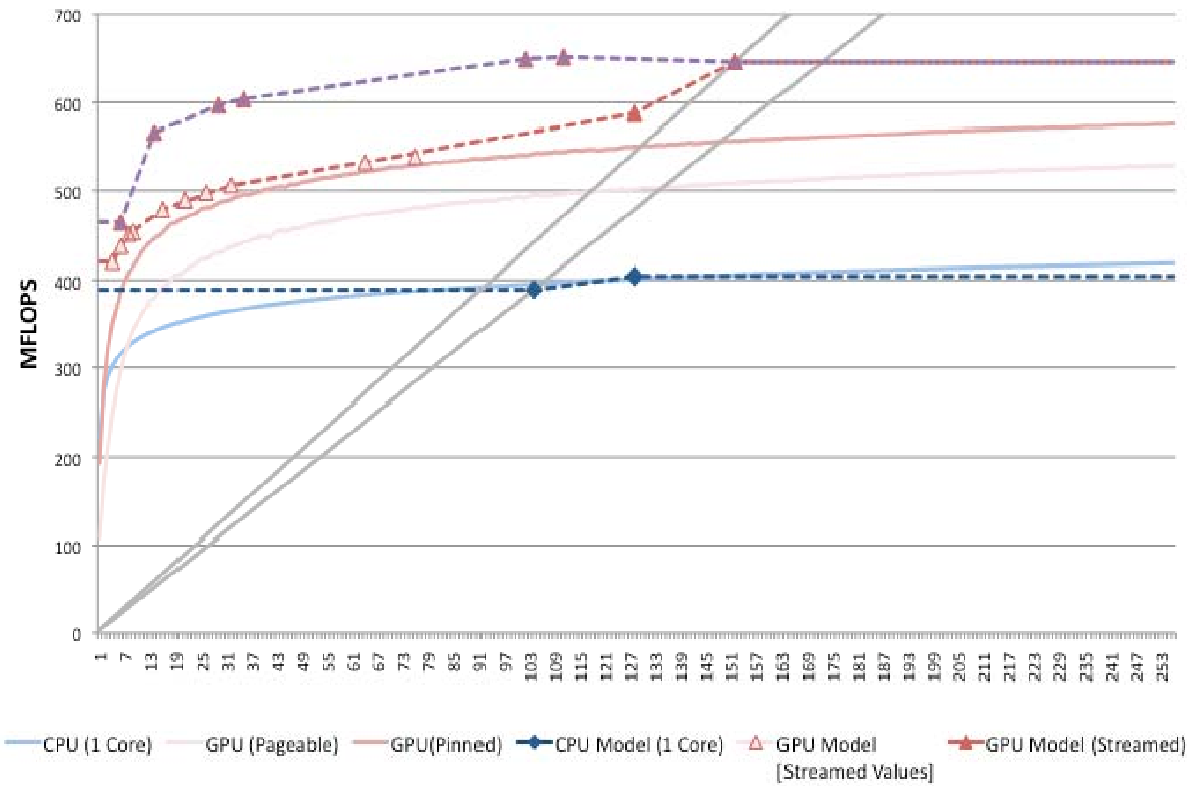
- *HostToDevice* Bandwidth
- *DeviceToHost* Bandwidth
- GPU *Kernel* Performance

③ Incorporate streaming results

- for each stream
- for each stream restart
- for every stream combination

Case Study: Performance Modeling in CPU + GPU Environment (22)

Performance Metric → Initialization → Approximation → **Iteration**



① Draw Upper U and Lower L lines through the following points:

$(0,0)$, $(N_1/p, \max_i\{Perf_i(N_1/p)\})$

$(0,0)$, $(N_1/p, \min_i\{Perf_i(N_1/p)\})$

② Let $x_i^{(U)}$ and $x_i^{(L)}$ be the intersections with $Perf_i(x)$

IF exists $x_i^{(L)} - x_i^{(U)} \geq 1$

THEN go to 3

ELSE go to 5

③ Bisect the angle between U and L by the line M, and calculate intersections $x_i^{(M)}$

④ IF $\Sigma_i x_i^{(M)} \leq N_1$

THEN U=M

ELSE L=M

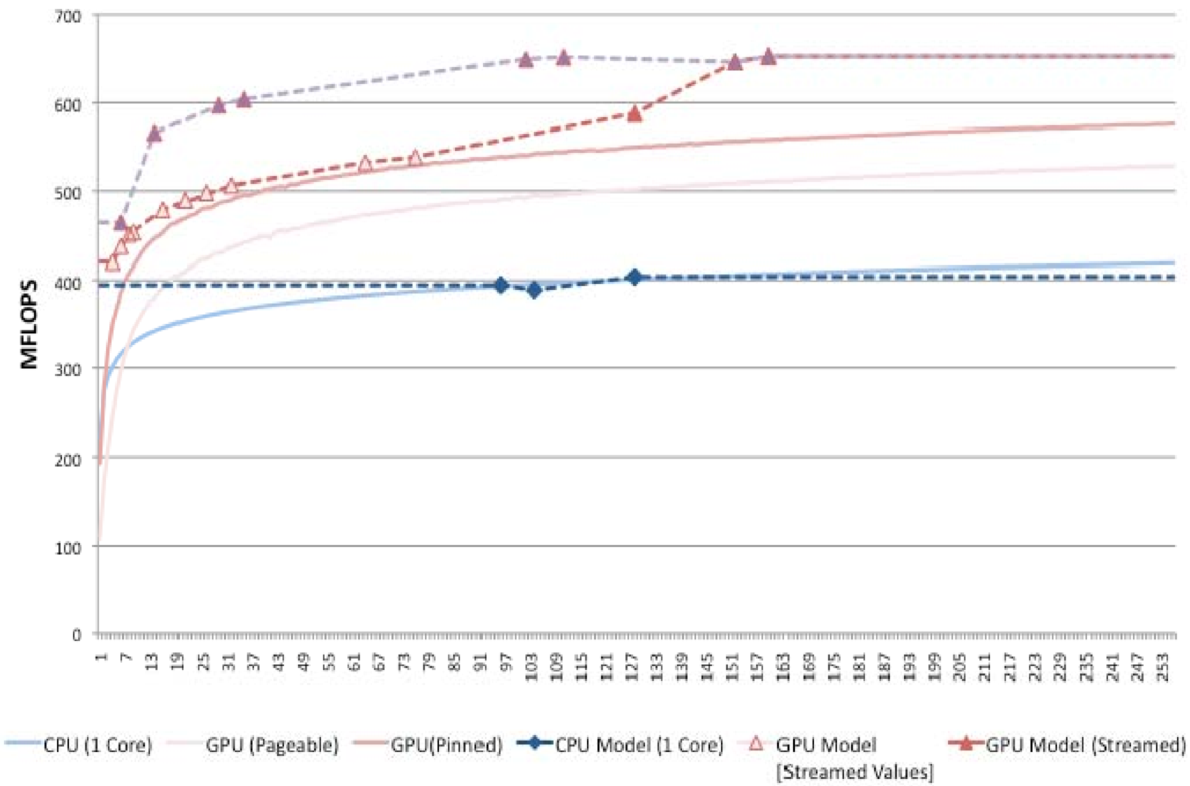
REPEAT 2

⑤ Employ streaming strategy on the calculated workload value

Case Study: Performance Modeling in CPU + GPU Environment (23)

technology from seed

Performance Metric → Initialization → **Approximation** → Iteration



① Refine performance models with the newly obtained results

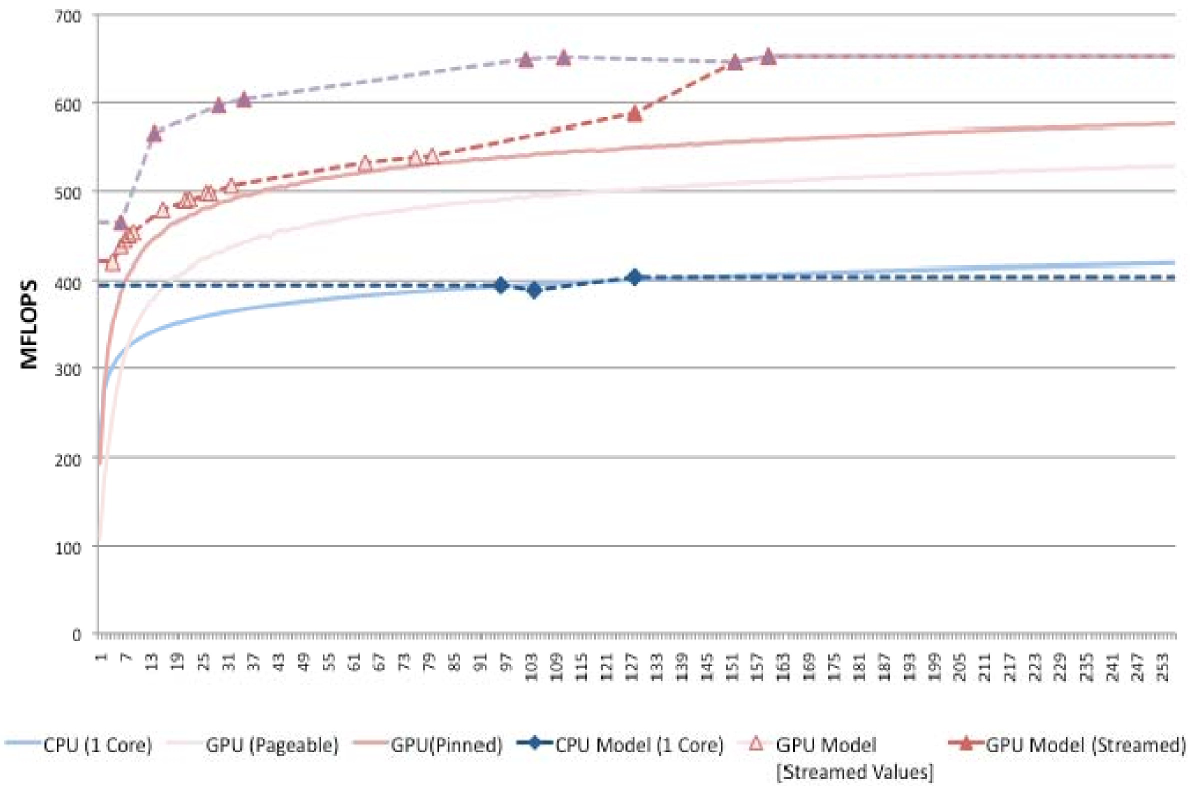
2 GPU-specific Modeling: Using the obtained values from streaming execution
   - *HostToDevice* Bandwidth
   - *DeviceToHost* Bandwidth
   - GPU *Kernel* Performance

3 Incorporate streaming results
   - for each stream
   - for each stream restart
   - for every stream combination

Case Study: Performance Modeling in CPU + GPU Environment (24)

technology
from seed

Performance Metric → Initialization → **Approximation** → Iteration



1 Refine performance models with the newly obtained results

② GPU-specific Modeling: Using the obtained values from streaming execution

- *HostToDevice* Bandwidth
- *DeviceToHost* Bandwidth
- GPU *Kernel* Performance

③ Incorporate streaming results

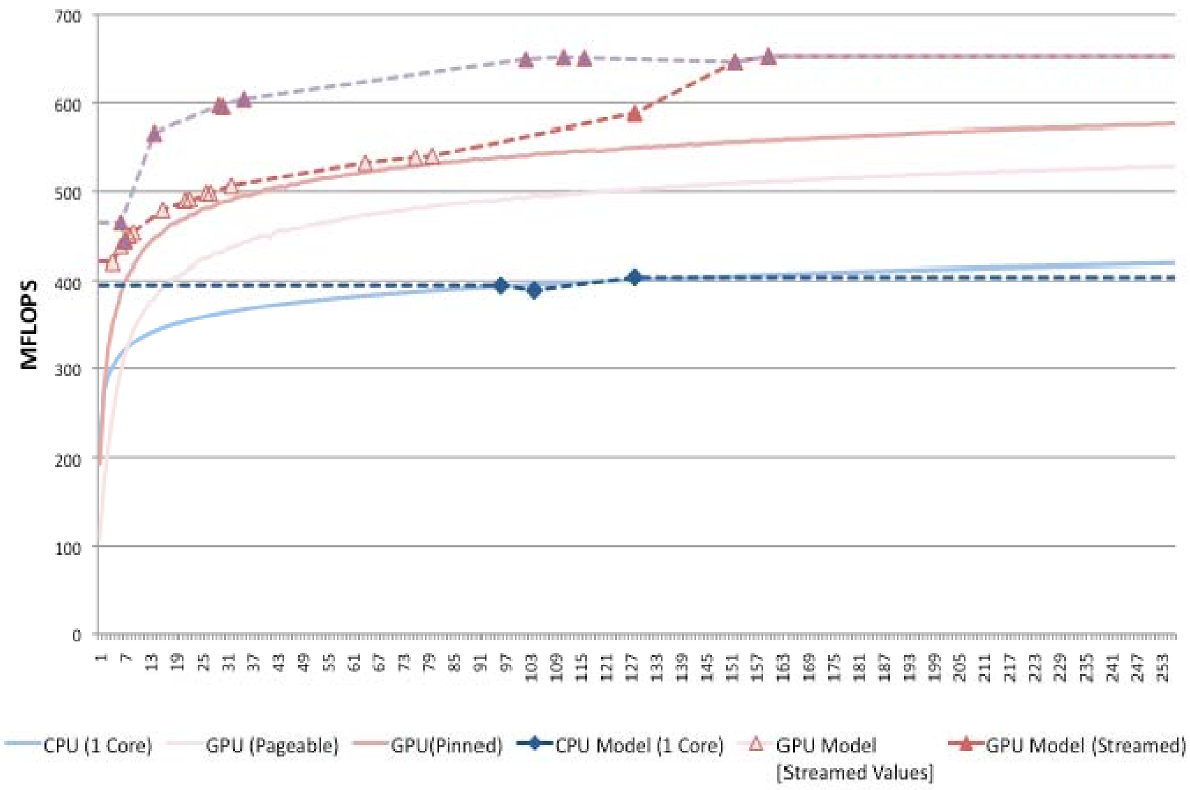
- for each stream
- for each stream restart
- for every stream combination

Case Study: Performance Modeling in CPU + GPU Environment (25)

technology
from seed

inesc id lisboa

Performance Metric → Initialization → **Approximation** → Iteration



1 Refine performance models with the newly obtained results

② GPU-specific Modeling: Using the obtained values from streaming execution

- *HostToDevice* Bandwidth
- *DeviceToHost* Bandwidth
- GPU *Kernel* Performance

③ Incorporate streaming results

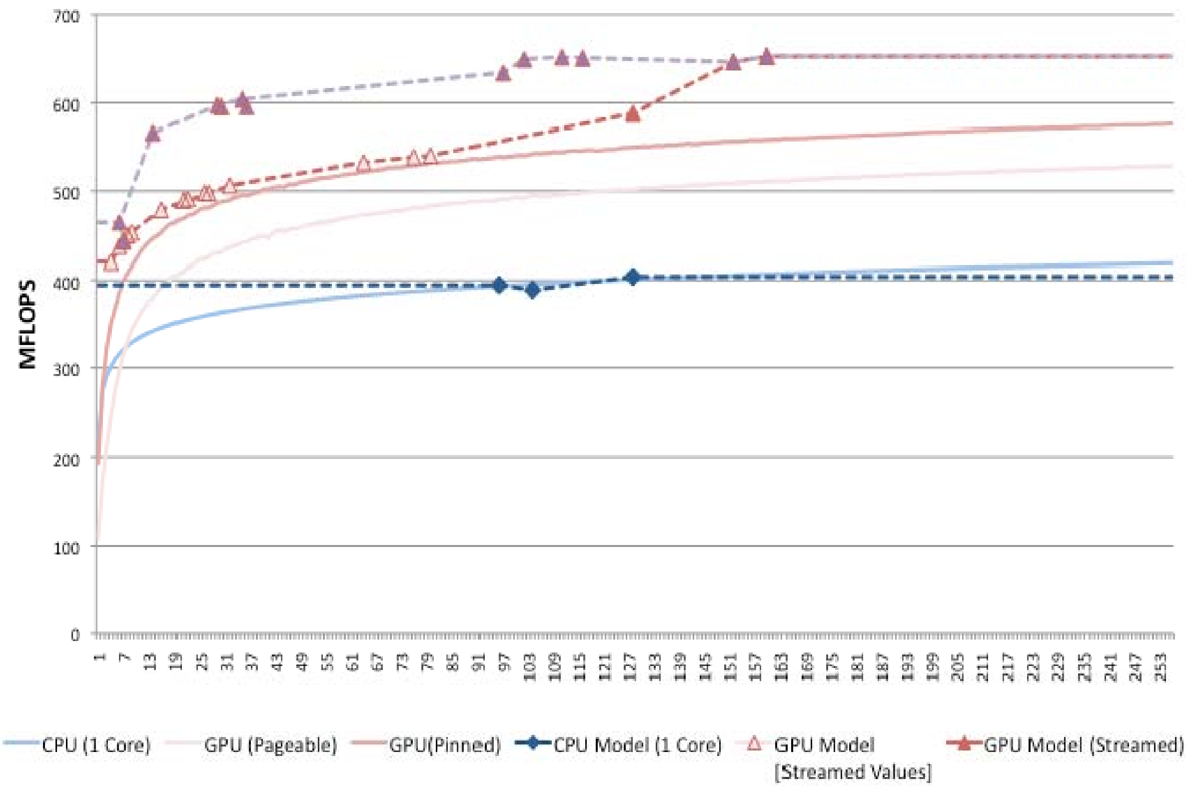
- for each stream
- for each stream restart
- for every stream combination

Case Study: Performance Modeling in CPU + GPU Environment (26)

technology from seed

inesc id lisboa

Performance Metric → Initialization → **Approximation** → Iteration



1. Refine performance models with the newly obtained results

② GPU-specific Modeling: Using the obtained values from streaming execution

- *HostToDevice* Bandwidth
- *DeviceToHost* Bandwidth
- GPU *Kernel* Performance

③ Incorporate streaming results

- for each stream
- for each stream restart
- for every stream combination

COLLABORATIVE ENVIRONMENT FOR HETEROGENEOUS COMPUTERS

TRADITIONAL APPROACHES FOR PERFORMANCE MODELING

- Approximate the performance using number of points equal to the number of iterations
- In this case, **3 POINTS** per each device

PRESENTED APPROACH FOR PERFORMANCE MODELING

- Models the performance using **MORE THAN 30 POINTS**, in this case
- **COMMUNICATION-AWARE** – schedules in respect to limited and asymmetric interconnection bandwidth
- Employs **STREAMING STRATEGIES** to overlap communication with computation across devices
- **BUILDS SEVERAL** PER-DEVICE **MODELS** AT THE SAME TIME
  - OVERALL PERFORMANCE for each device + STREAMING GPU PERFORMANCE
  - HOSTTODEVICE BANDWIDTH Modeling
  - DEVICETOHOST BANDWIDTH Modeling
  - GPU KERNEL PERFORMANCE Modeling

Questions?

Thank you