A study on adaptive algorithms for numerical quadrature on heterogeneous multicore and GPU based systems

Giuliano Laccetti[1], Marco Lapegna[1], Valeria Mele[1], Diego Romano[2]

A parallel adaptive algorithm for the computation of a multidimensional integral on heterogeneous systems is described. Two different strategies have been combined together in a single algorithm: a first procedure is in charge of the load balancing among the threads on the multicore CPU and a second one is in charge of an efficient execution on the GPU of the computational kernel. Experimental results on a system with a quad-core CPUs Intel Core I7 950 @ 3Ghz and two GPUs NVIDIA C1060 have been achieved.

PROBLEM: $I(f) = \int_U f(t_1, \ldots, t_d)dt_1 \cdots dt_d$

$U = [a_1, b_1] \times \cdots \times [a_d, b_d] \qquad 2 \le d \le 10$

Basic rule: $r = \sum_{i=1}^{n} A_i f(x_i) \sim \int_U f(t_1, \ldots, t_d)dt_1 \cdots dt_d$

$O(10^3) \le n \le O(10^4)$ *(Genz & Malik rule)*

Adaptive algorithm:

An iterative procedure that refines the integration domain U, evaluating the quadrature rule r only in the subdomains s^* with the largest error e^* in a suitable data structure H

initialize H, Q^(0) and E^(0)
while (stopping criterion not satisfied) do iteration j
1) select $s^* \in H$ such that $e^* = \max e(k)$
2) divide s^* in two parts $s(\lambda)$ and $s(\mu)$
3) compute $r(\lambda)$, $r(\mu)$, $e(\lambda)$ and $e(\mu)$
4) sort the subdomains in H according the errors e(k)
5) update Q^(j) and E^(j)
endwhile

$r = \sum_{i=1}^{n} A_i f(x_i)$

the subdomains s in H are independent and can be processed by different threads of the CPU

Host algorithm based on Parallelism at Subdomains Level

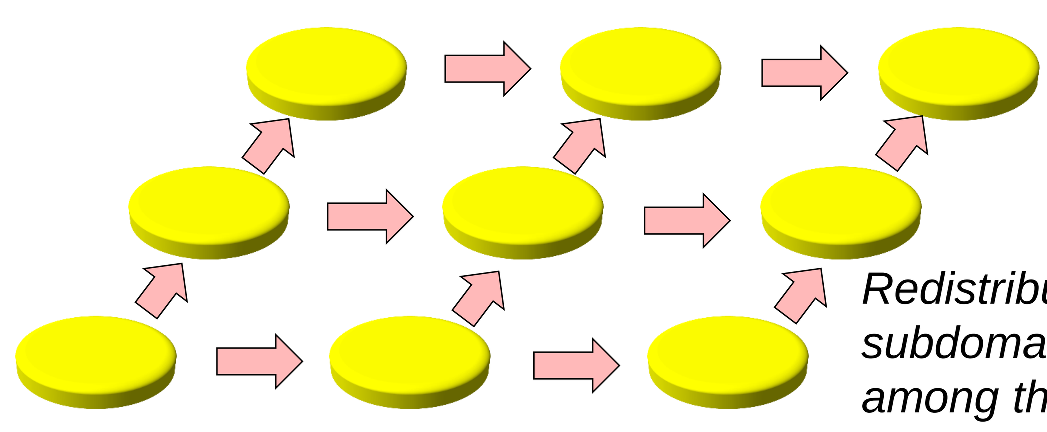
the function evaluations $f(x_i)$ in the basic rule r are independent and can be executed by different threads of the GPU

Device algorithm based on Parallelism at Integration Formula Level

Main issue: design of a scalable H

The access to a single centralized H produces fast numerical convergence but several synchronizations among all threads (therefore very poor scalability)

Solution: No centralized data structure! The threads are logically organized according to a 2-dimensional periodical mesh M_2. Each thread T_i manages a private sub-structure H_i. Then it attempts to share its item $s*_i \in H_i$ with largest error $e*_i$, only with the neighbor threads in the mesh M_2.



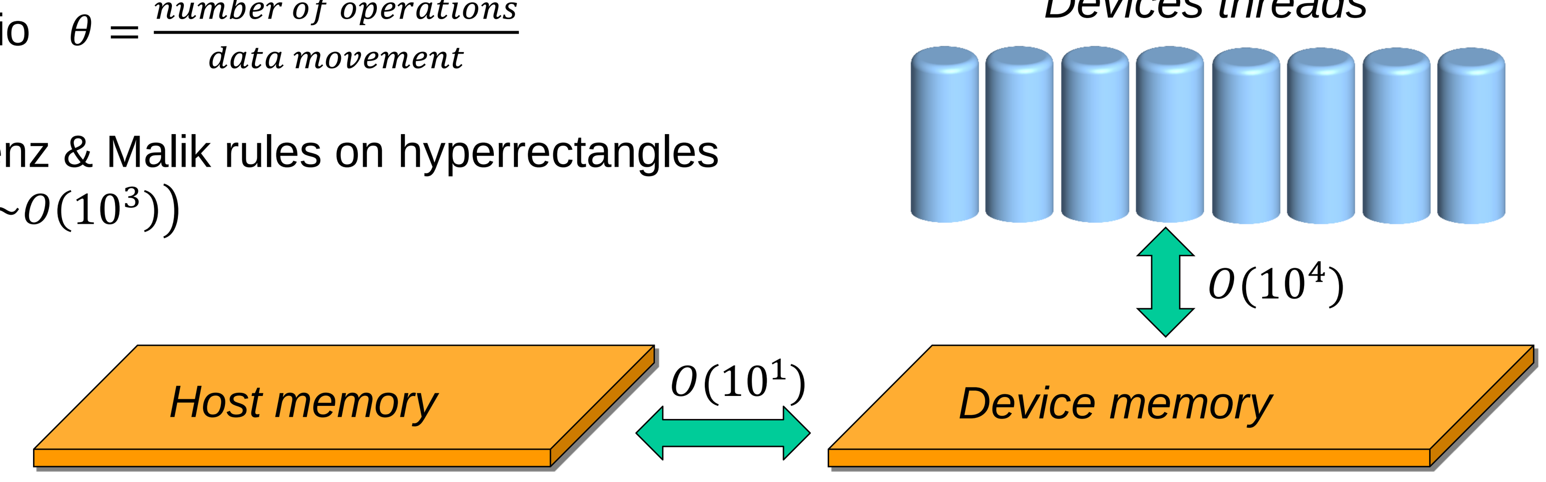
Redistribution scheme of the subdomains with large errors (red) among the threads (yellow)

Main issues: management of the fine grain parallelism

1) Summation is a hard to optimize kernel (e.g. many idle threads)
Solution: use of the reduce functions in CUDA toolkit 4.0

2) Low bandwidth between device and host memories
Solution: use of quadrature rules with high ratio $\theta = \frac{\text{number of operations}}{\text{data movement}}$

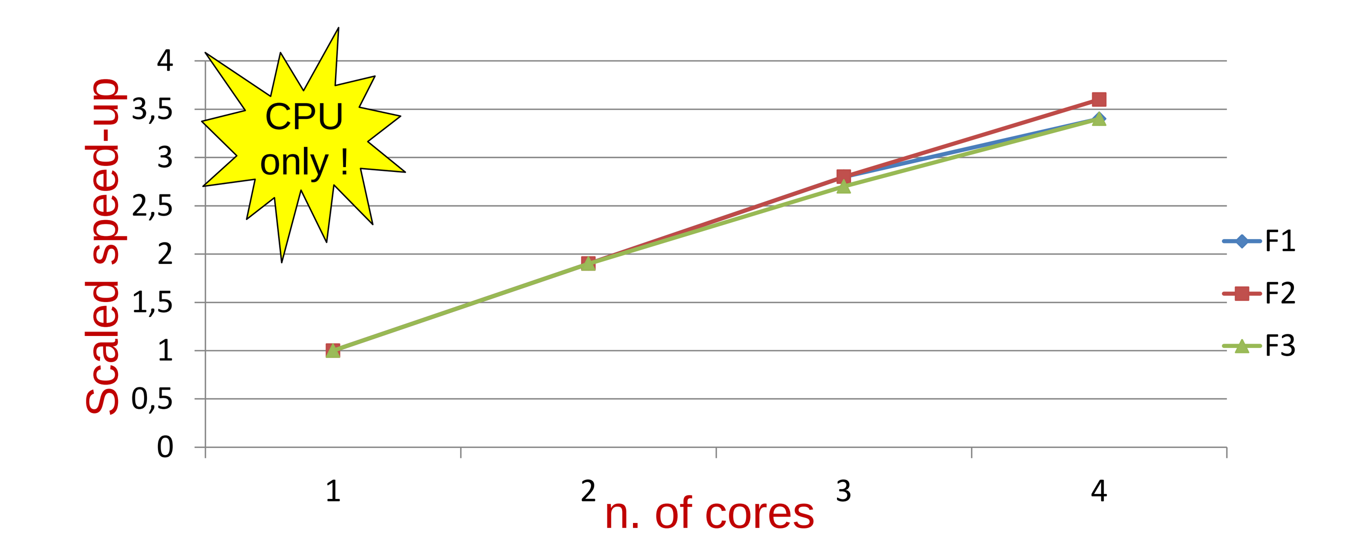
Genz & Malik rules on hyperrectangles $(\theta \sim O(10^3))$



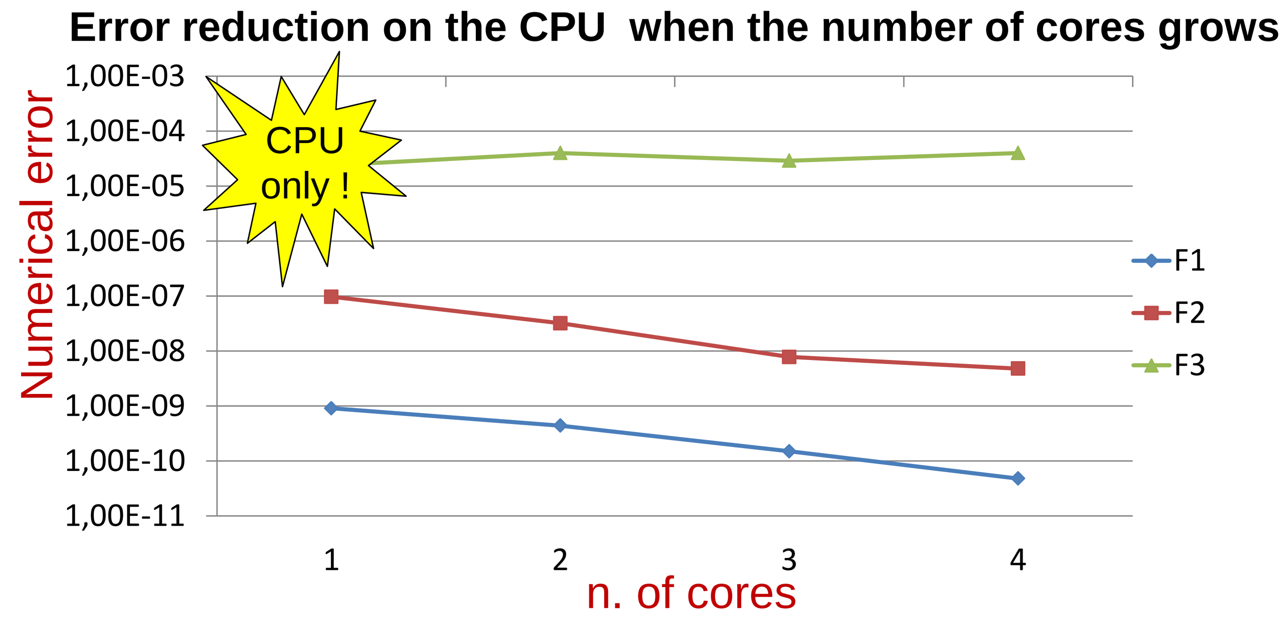
Relationship between data movement and floating point operations in Genz & Malik rule

TEST RESULTS

Scalability test (fixed time) on the CPU when the number of cores grows



Error reduction on the CPU when the number of cores grows



10 functions in each family

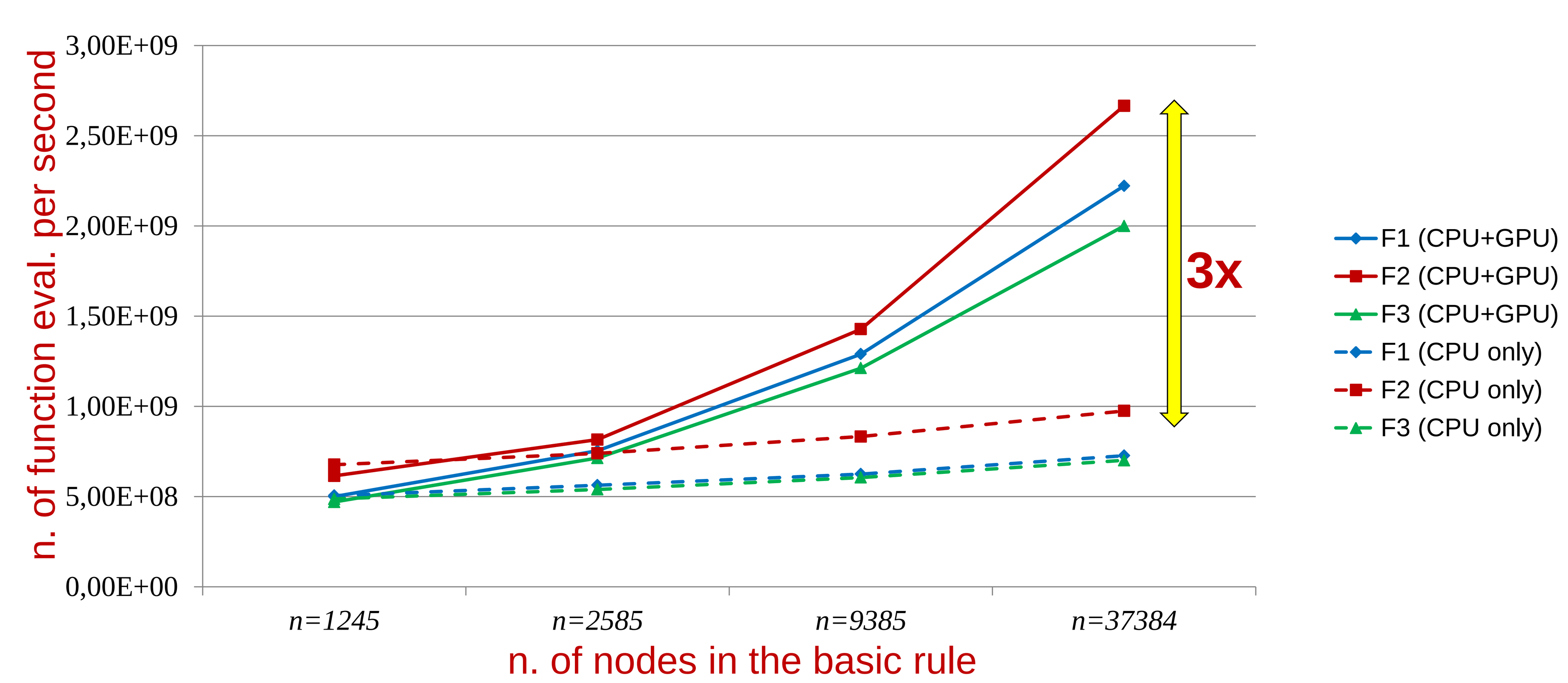
$F1 = \cos(2\pi\beta_1 + \sum_{i=1}^{d} \alpha_i x_i)$ oscillating functions

$F2 = (1 + \sum_{i=1}^{d} \alpha_i x_i)^{-d-1}$ corner peak functions

$F3 = \exp(-\sum_{i=1}^{d} \alpha_i |x_i - \beta_i|)$ $C^{(0)}$ functions

- Basic rule with $n=1245$ function evaluations in $d=10$ dimensions
- 10×10^6 function evaluations per core (4016 iterations)

Performance gain of the algorithm with (solid line) and without (dashed line) the use of GPU



CONCLUSIONS

- *We presented a heterogeneous multicore CPU/GPU algorithm with a performance gain of 3x with respect to a traditional quadrature adaptive algorithm running just on current homogeneous multicore CPUs.*
- *The approach demonstrates the utility of graphics accelerators for multidimensional quadrature mainly with a large number of function evaluations of the basic rule ($n>10^4$) and in large dimension ($d>10$).*
- *Our approach can be combined with other levels of parallelism (e.g. cluster level)*

[1] Department of Mathematics and its Applications
University of Naples Federico II,
Via Cintia Monte S. Angelo,
80126 Naples, Italy

[2] Institute of High Performance Computing and Networking ICAR - Naples branch
National Research Council
via P. Castellino, Naples, Italy

Second Workshop on Models, Algorithms and Methodologies for Hierarchical Parallelism in new HPC Systems –
jointly with Parallel Processing and Applied Mathematics Conference PPAM2013 – Warsaw, Poland September 8-11, 2013