A Parallel Framework for Solving PDEs with Radial Basis Functions

Evan F. Bollig, Advisor: Gordon Erlebacher

Department of Scientific Computing, Florida State University

`bollig@scs.fsu.edu`

Abstract

In the last four decades, new methods for the solution of PDEs have surfaced emphasizing the use of a non-orthogonal class of basis functions commonly referred to as Radial Basis Functions (RBFs). RBFs are spherically symmetric functions. That is, univariate functions centered at some point and symmetric about their center in any dimension. Common examples of these functions (illustrated in Figure 1) include Thin Plate Splines (TPS), Gaussians (GA) and Multiquadrics (MQ).

We are investigating the use of RBFs to solve 2D and 3D linear and non-linear PDEs over a collection of nodes contained within ellipsoidal geometries (e.g., see Figure 2). Our chosen differentiation method, called RBF-FD, expresses derivatives as linear combinations of function values with weights dependent on stencils of nodes and RBFs. We are currently devoted to the development of an efficient parallel RBF compute framework for problems with large numbers of nodes/stencils spanning many processors.

Additionally, over the past decade, commodity Graphics Processing Units (GPUs) specialized for 2D and 3D scene rendering have seen an explosive growth in raw compute capability compared to their general purpose counterpart, the CPU (see Figure 3). Currently capable of near teraflop speeds and sporting gigabytes of on-board memory in a single unit, GPUs have transformed from accessory video game hardware to truly general purpose computational co-processors. In an effort to leverage this computational potential, we present plans to extend the parallel RBF framework to efficiently compute on heterogeneous multi-core clusters (i.e. a multi-GPU, MPI implementation on clusters with many CPUs and GPUs).

1. Radial Basis Functions

Definition 1 *A function $\Phi : \mathbb{R}^s \to \mathbb{R}$ is called radial provided a univariate function $\varphi : [0, \infty) \to \mathbb{R}$ exists such that*

$$\Phi(x) = \varphi(\varepsilon\,||x - x_j||)$$

where x_j is the center or point of origin *for the function, and $||\cdot||$ is some norm (typ. ℓ_2 norm) on $\mathbb{R}^s$. ε is a support scaling parameter [1].*

For simplification we substitute $r = ||x - x_j||$

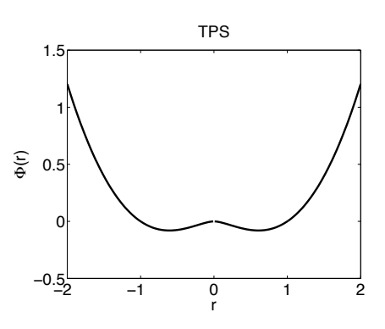
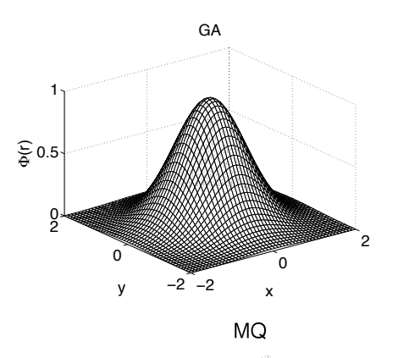
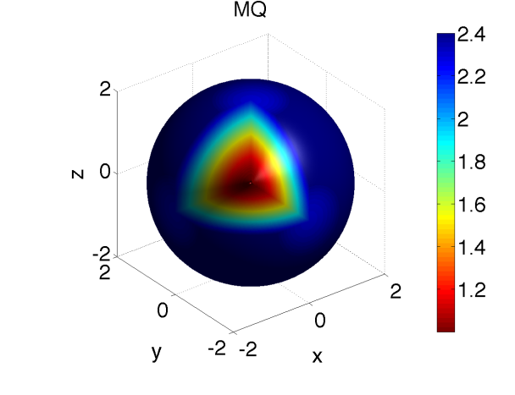
Thin Plate Spline (TPS)	$(\varepsilon r)^2 ln\|\varepsilon r\|$		(1D)
Gaussian (GA)	$e^{-(\varepsilon r)^2}$		(2D)
Multiquadric (MQ)	$\sqrt{1 + (\varepsilon r)^2}$		(3D)

Figure 1: *Three common Radial Basis Functions (RBFs) used for the solution of PDEs.*

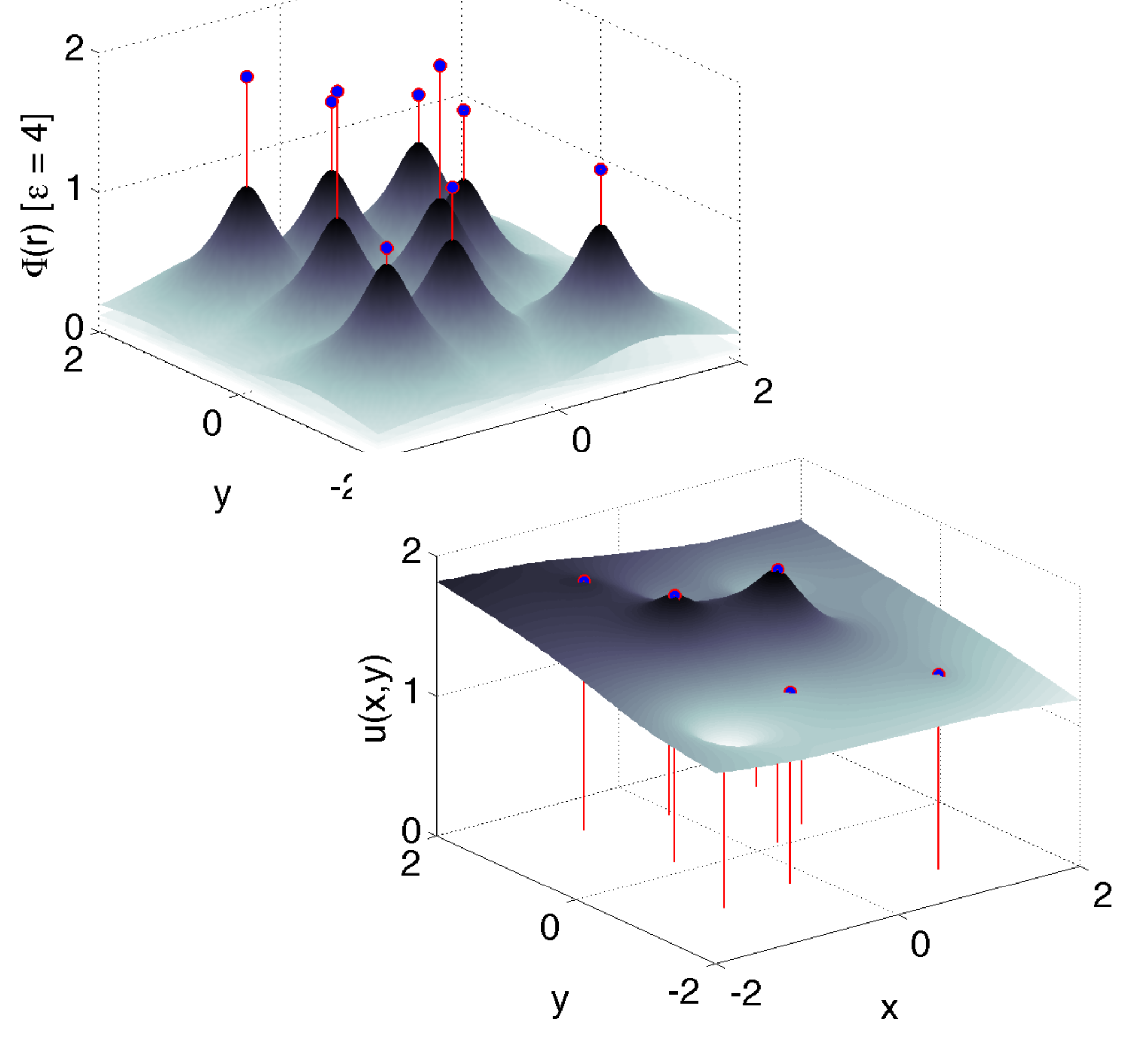


Figure 3: *Every node in the domain is assigned an RBF and the functions are used to construct a solution manifold at each timestep.*

2. RBF-FD

We solve PDEs using a generalized FD scheme called RBF-FD [2].

$$f(x) = \sum_{j=1}^{N} c_j \Phi_j(x) + \sum_{l=1}^{M} d_l P_l(x), \qquad P_l(x) \in \Pi_m^D$$

$M = \binom{m+D}{D}$. In 2D (m=2): $\mathbb{P} = 1, x, y, xy, x^2, y^2$ Reformulate as a linear system:

$$\Phi c + \mathbb{P} d = f$$

But we also need constraints:

$$\mathbb{P}^T c = 0$$

$$\begin{bmatrix} \Phi & \mathbb{P} \\ \mathbb{P}^T & 0 \end{bmatrix} \begin{pmatrix} c \\ d \end{pmatrix} = \begin{pmatrix} f \\ 0 \end{pmatrix}$$
$$Ac = f$$

Set $f = u$ and solve for coefficients, c:

$$c = A^{-1} u$$

Then for any linear operator D construct the approximate formula:

$$\begin{aligned} Du &= \sum_{j=1}^{N_S} c_j D\Phi_j(x) + \sum_{l=1}^{M} d_l DP_l(x) = \left[\, D\Phi \;\; D\mathbb{P} \,\right] \begin{pmatrix} c \\ d \end{pmatrix} \\ &= A_D c \\ &= A_D A^{-1} u \end{aligned}$$

- Derivatives are approximated as combinations of neighboring function values
- Node weights depend on RBFs
- $\Phi_j(x)$ and $D\Phi_j(x)$ are analytic evaluations

3. Example Problem

Linear and non-linear PDE solutions can be computed over a collection of nodes contained within 2D and 3D ellipsoidal geometries.

$$\begin{aligned} \frac{\partial T(\mathbf{r}, t)}{\partial t} &= \nabla \cdot [D(T, \mathbf{r}) \nabla T(\mathbf{r}, t)] \\ &= \nabla D \cdot \nabla T + D \nabla^2 T \end{aligned}$$

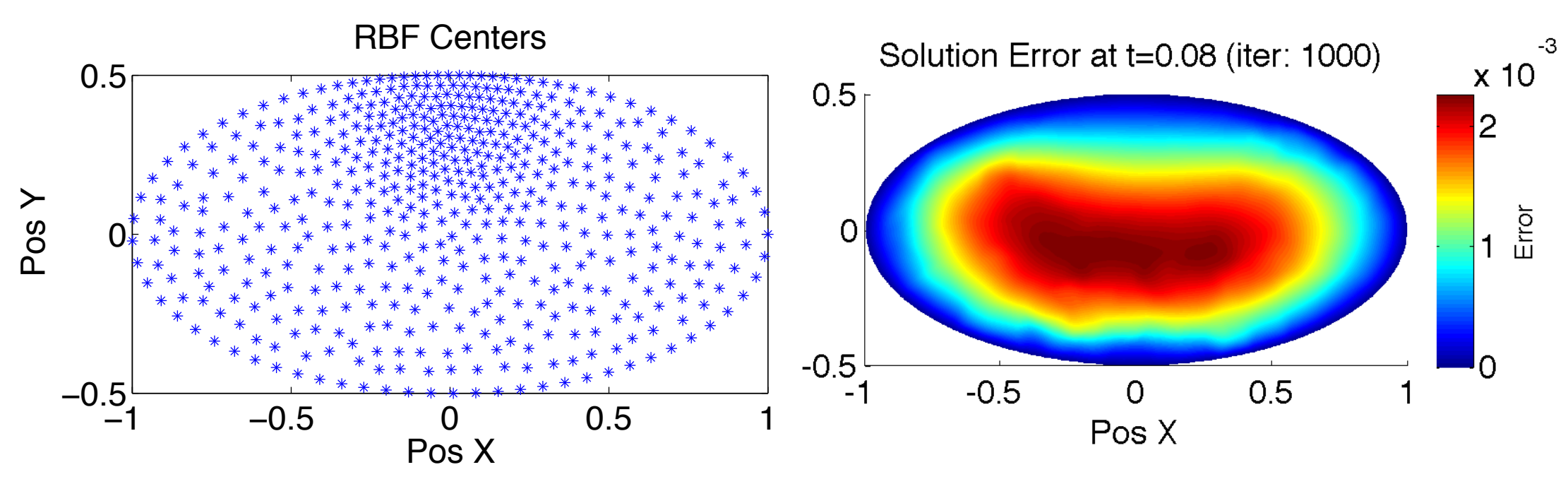
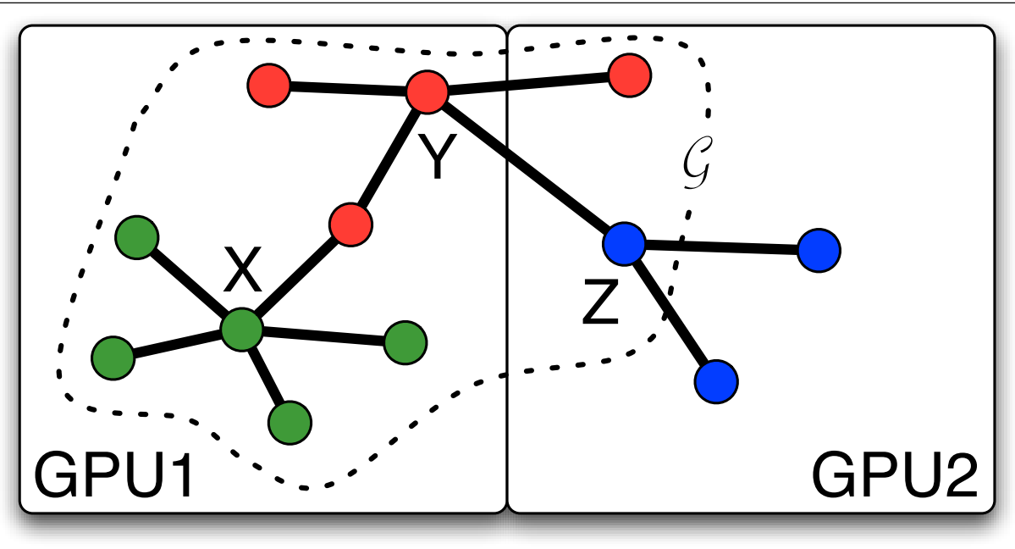


Figure 4: *An example domain and solution error with non-uniformly distributed nodes. Nodes distributed by Centroidal Voronoi Tessellation over ellipse with $\rho = 0.05 + e^{(-15*(x^2+(y-0.4)^2))}$. $\Delta t = 0.00008$, 1000 timesteps. Exact solution: $\cos(\frac{\pi}{2} * \sqrt{x^2 + 0.5 * y^2}) * e^{(-t)}$*

4. Parallel Implementation

RBF stencils are partitioned across hosts (CPU/GPU), and sets are structured in memory to enable overlapping computation and communication for increased efficiency.

$\mathcal{G}$	: all nodes received and contained on the GPU g
$\mathcal{Q}$	: stencil centers managed by g (equivalently, stencils computed by g)
$\mathcal{B}$	: stencil centers managed by g that require nodes on another GPU
$\mathcal{R}$	: nodes required by g that are managed by another GPU
$\mathcal{O}$	: nodes managed by g that are sent to other GPUs



$x, y, z \in \mathcal{G}$	$x, y \in \mathcal{Q}$
$y \in \mathcal{B}$	$z \in \mathcal{R}$
$y \in \mathcal{O}$	$\mathcal{G} = \mathcal{R} \cup \mathcal{Q}$
$x \in \mathcal{Q} \backslash \mathcal{B}$	$\mathcal{B} \subset \mathcal{Q}$

Centers in memory: $\mathcal{G} = \{\mathcal{Q} \backslash \mathcal{O} \;\; \mathcal{O} \;\; \mathcal{R}\}$

Table 1: *Stencil set partitioning by compute node (GPU1).*

Multi-stage derivative calculation:

$$u'_k = (A_D A^{-1})_k u_{k,\mathcal{Q}\backslash\mathcal{O}} + (A_D A^{-1})_k u_{k,\mathcal{O}} + (A_D A^{-1})_k u_{k,\mathcal{R}}$$

- First and second products do not require communication (ideal for overlapping computation and communication)
- Last inner product blocks until communication finishes

5. Heterogeneous Multi-Core Computing

New compute clusters are integrating GPUs to provide joint multi-core CPU and multi-core GPU compute environments. This new design requires advanced code design to efficiently leverage both sets of hardware.

ACM.sc.fsu.edu
(A Heterogeneous Multi-Core Cluster)

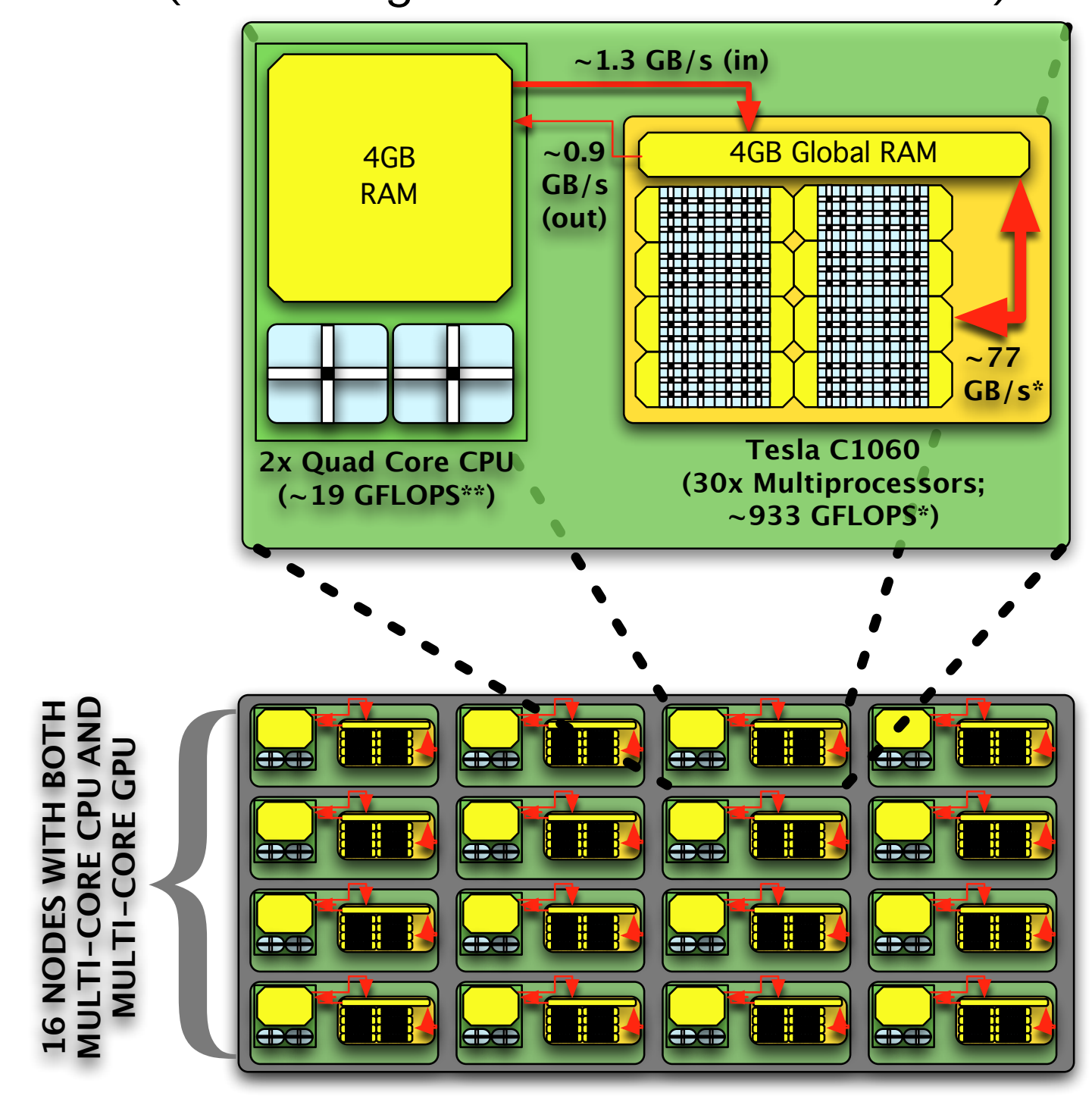

~1.3 GB/s (in)
4GB RAM
~0.9 GB/s (out)
4GB Global RAM
~77 GB/s*
2x Quad Core CPU (~19 GFLOPS**)
Tesla C1060 (30x Multiprocessors; ~933 GFLOPS*)
16 NODES WITH BOTH MULTI-CORE CPU AND MULTI-CORE GPU

Figure 5: *The ACM cluster (maintained by the Department of Scientific Computing).*

6. Why Compute on GPUs?

Performance growth of the GPU in recent years, joined by improved GPU programming languages demands consideration of GPU as general purpose co-processor.

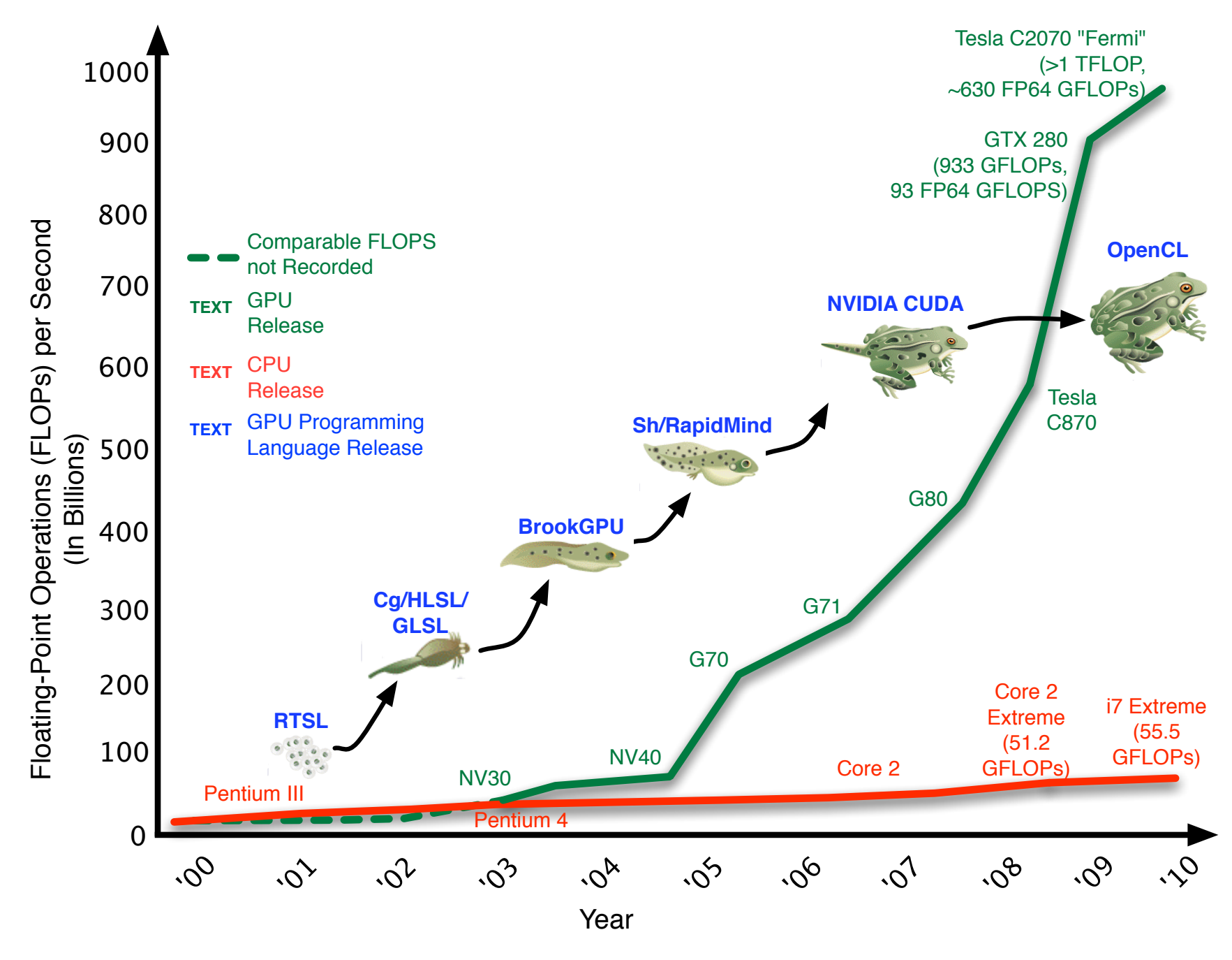


Figure 6: *The growing divide between estimated peak performance of CPUs and GPUs, joined by landmark releases of maturing languages to control the hardware.*

7. Computing on Heterogeneous Multi-Core Architectures

Programming parallel kernels to run on CPU and GPU hardware is simplified with the OpenCL language.

OpenCL Memory Model

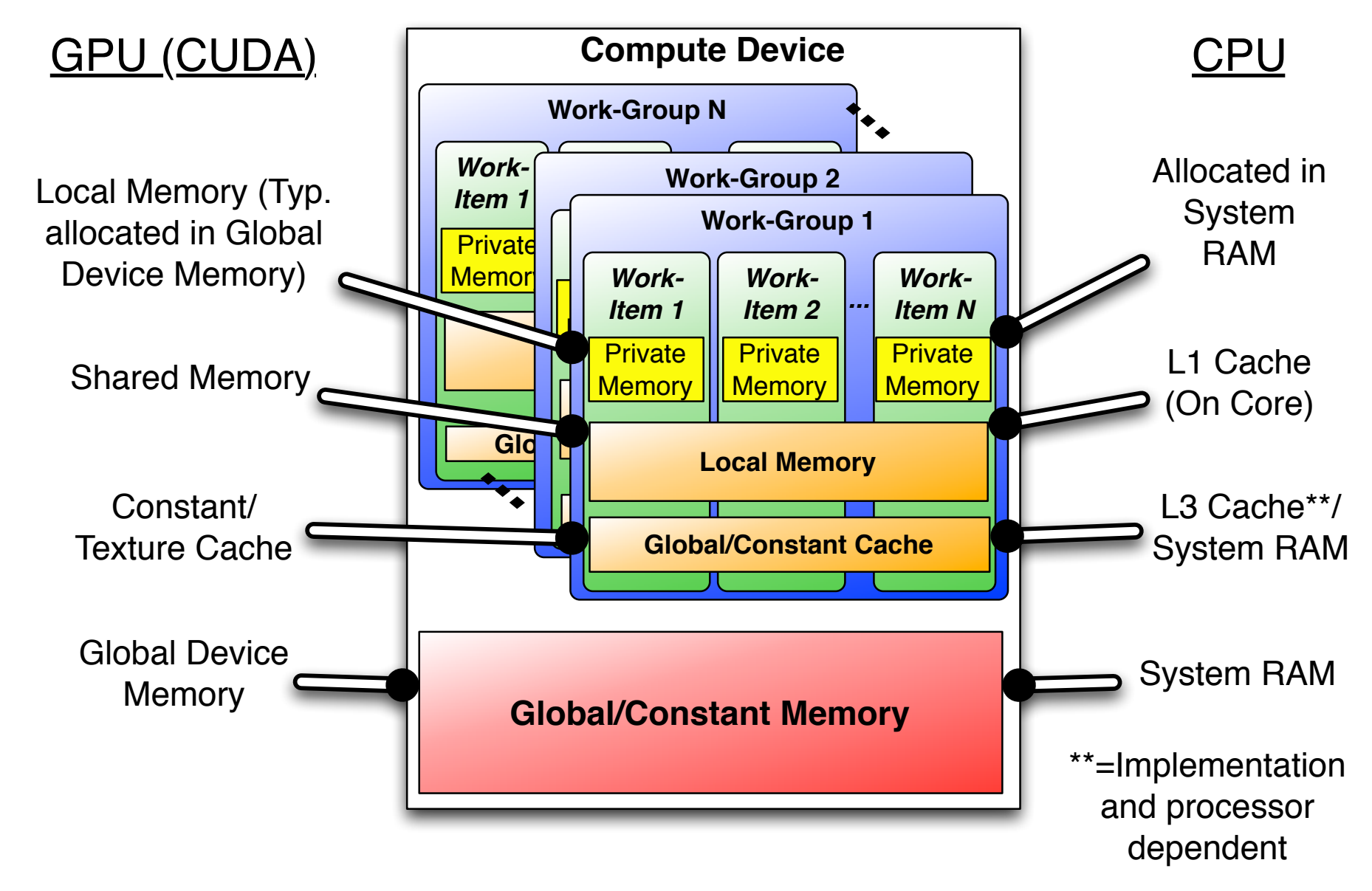


Figure 7: *OpenCL allows "functional portability"; that is, an abstract notion of parallel architecture that maps to both CPU and GPU.*

8. Acknowledgements

This work is supported by NSF award No. #0934331: "CMG Collaborative Research: Fast and Efficient Radial Basis Function Algorithms for Geophysical Modeling on Arbitrary Geometries".

References

[1] Fasshauer, G. E. *Meshfree Approximation Methods with MATLAB*, vol. 6 of *Interdisciplinary Mathematical Sciences.* World Scientific Publishing Co. Pte. Ltd., 5 Toh Tuck Link, Singapore 596224, 2007.

[2] Tolstykh, A. I., and Shirobokov, D. A. On using radial basis functions in a "finite difference mode" with applications to elasticity problems. In *Computational Mechanics*, vol. 33. Springer, December 2003, pp. 68 – 79.