

# A Balanced Accumulation Scheme for Parallel PDE Solvers

Manfred Liebmann  
Institute for Mathematics and Scientific Computing  
University of Graz  
`manfred.liebmann@uni-graz.at`



September 17, 2014

## SFB MOBIS: Principal Investigators

- **Medical University of Graz, Austria**

- Gernot Plank
- Christoph Augustin
- Caroline Costa

- **University of Graz, Austria**

- Gundolf Haase
- Manfred Liebmann
- Aurel Neic



## (1) Motivation: CARP Virtual Heart Project

The virtual heart model is based on the bidomain equations, a set of coupled partial differential equations, which describe the current flow in the myocardium. The bidomain equations in the elliptic-parabolic form the equations are given by

$$\begin{bmatrix} -\nabla \cdot (\boldsymbol{\sigma}_i + \boldsymbol{\sigma}_e) \nabla \phi_e \\ -\nabla \cdot \boldsymbol{\sigma}_b \nabla \phi_e \end{bmatrix} = \begin{bmatrix} \nabla \cdot \boldsymbol{\sigma}_i \nabla V_m + I_{ei} \\ I_{eb} \end{bmatrix} \quad (1)$$

$$\beta C_m \frac{\partial V_m}{\partial t} = (\nabla \cdot \boldsymbol{\sigma}_i \nabla \phi_i) - \beta (I_{ion}(V_m, \boldsymbol{\eta}) - I_i) \quad (2)$$

$$V_m = \phi_i - \phi_e \quad (3)$$

where  $\phi_i$  and  $\phi_e$  are the intracellular and extracellular potentials,  $V_m = \phi_i - \phi_e$  is the transmembrane voltage,  $\boldsymbol{\sigma}_i$  and  $\boldsymbol{\sigma}_e$  are the intracellular and extracellular conductivity tensors and  $I_{ion}$  is the membrane ionic current density which depends on  $V_m$  and a set of state variables  $\boldsymbol{\eta}$ .

The PDEs given in (1) to (2) are spatially discretized using linear tetrahedral FEs. Parabolic and elliptic PDE are decoupled and operator splitting based on a Strang scheme is employed to solve the parabolic PDE. The inner kernel of the compute scheme is then given by

$$\mathbf{K}_{\mathbf{i}+\mathbf{e}}\phi_e^{t+1} = -\mathbf{P}\mathbf{K}_{\mathbf{i}}\mathbf{v}_{\mathbf{m}}^t + \mathbf{M}_{\mathbf{e}}\mathbf{I}_{\mathbf{e}}^t \quad (4)$$

$$\frac{\partial \boldsymbol{\eta}}{\partial t} = f(\mathbf{v}_{\mathbf{m}}^t, \boldsymbol{\eta}^{t+1/2}) \quad (5)$$

$$\mathbf{I}_{\mathbf{ion}}^{t+1/2} = g(\mathbf{v}_{\mathbf{m}}^t, \boldsymbol{\eta}^{t+1/2})$$

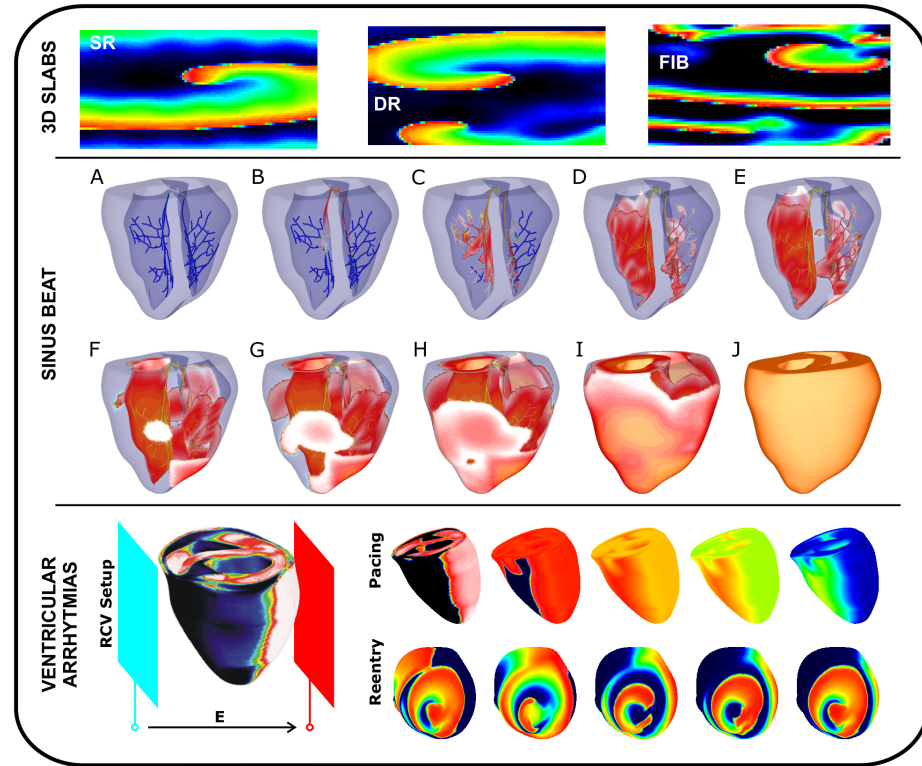
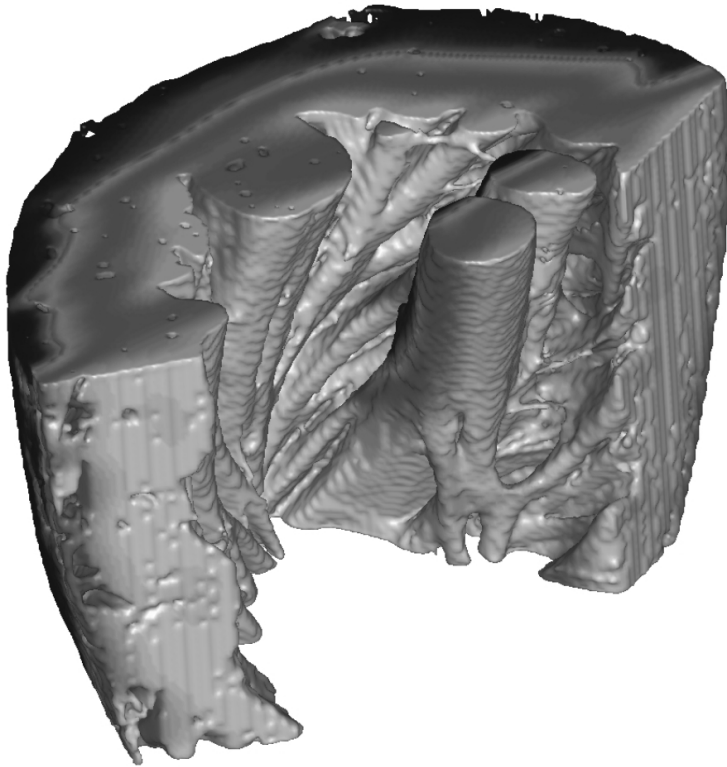
$$\mathbf{v}_{\mathbf{m}}^{t+1/2} = \mathbf{v}_{\mathbf{m}}^t - \frac{\Delta t}{C_m} \mathbf{I}_{\mathbf{ion}}^{t+1/2}$$

$$\mathbf{M}_{\mathbf{i}} \frac{\partial \mathbf{v}_{\mathbf{m}}^{t+1/2}}{\partial t} = -\frac{1}{\beta C_m} \mathbf{K}_{\mathbf{i}} \left( \mathbf{v}_{\mathbf{m}}^{t+1/2} + \mathbf{P}^{-\mathbf{T}} \phi_e^{t+1} \right) \quad (6)$$

where  $\mathbf{M}_{\zeta}$  and  $\mathbf{K}_{\zeta}$  are FEM mass and stiffness matrices.  $\mathbf{P}$  is a projection operator which transfers data between intracellular and extracellular grid, and  $\boldsymbol{\eta}$  is a discrete representation of the set of state variables of the ionic model.

For time discretization of the diffusion part of the parabolic PDE, the fully implicit Crank-Nicholson scheme is employed, with  $\kappa$  defined as  $\frac{\beta C_m}{\Delta t}$ :

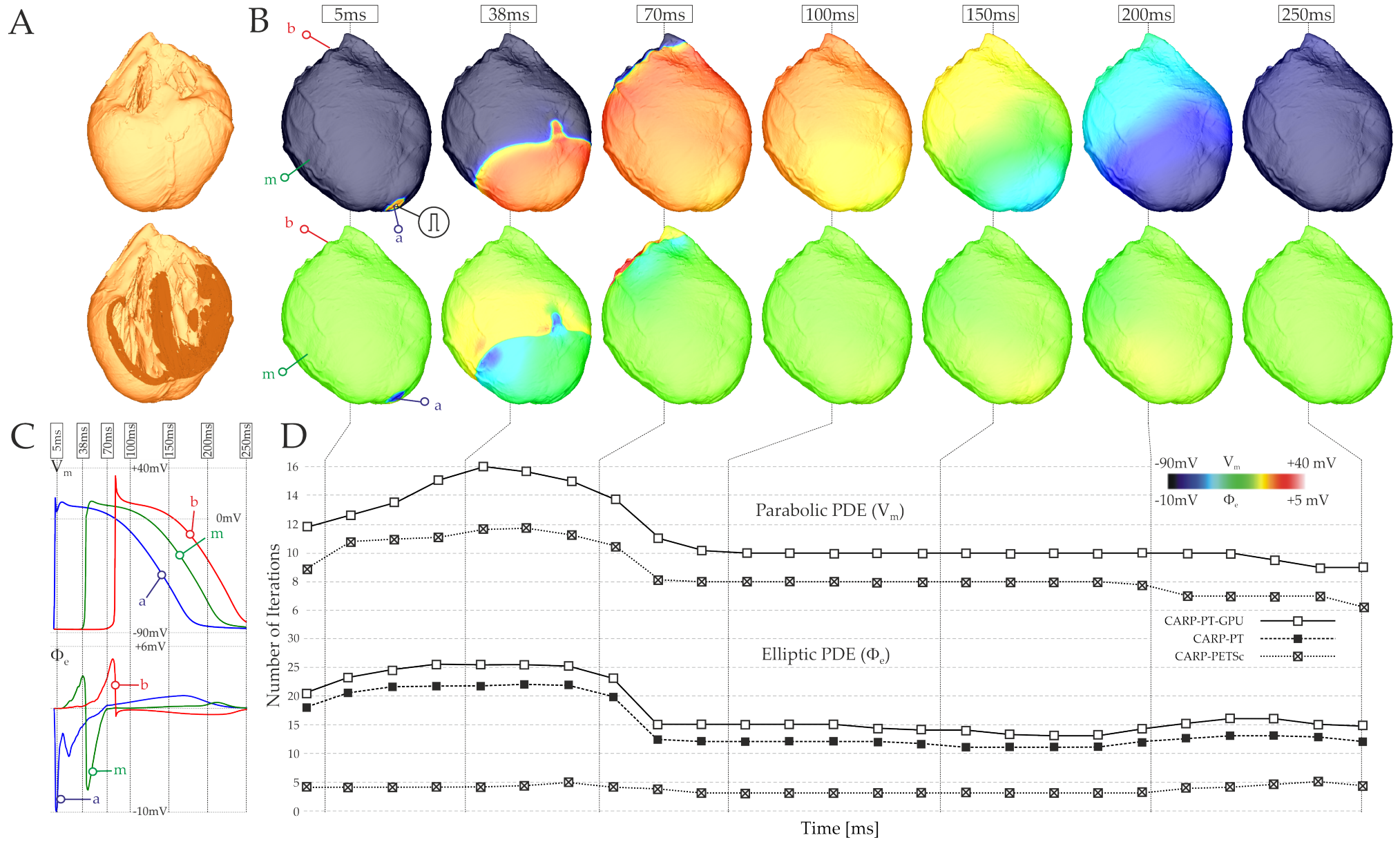
$$\left( \kappa \mathbf{M}_i + \frac{\mathbf{K}_i}{2} \right) \mathbf{v}_m^{t+1} = -\mathbf{K}_i \left( \frac{\mathbf{v}_m^{t+1/2}}{2} + \mathbf{P}^{-T} \phi_e^{t+1} \right) + \kappa \mathbf{M}_i \mathbf{v}_m^{t+1/2} \quad (7)$$

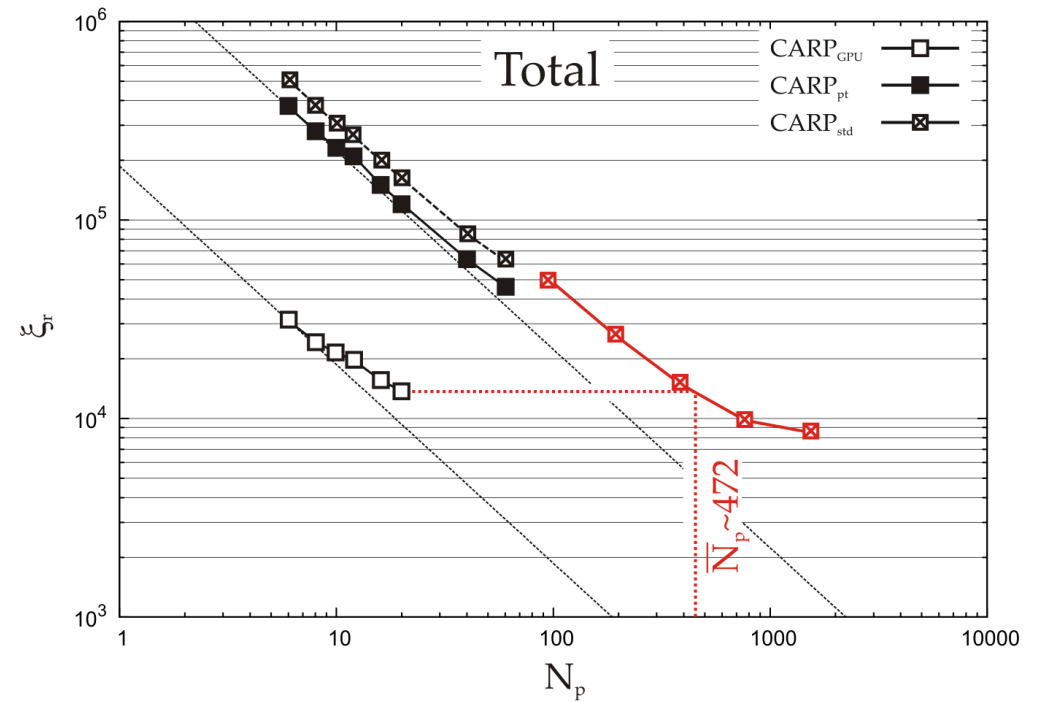
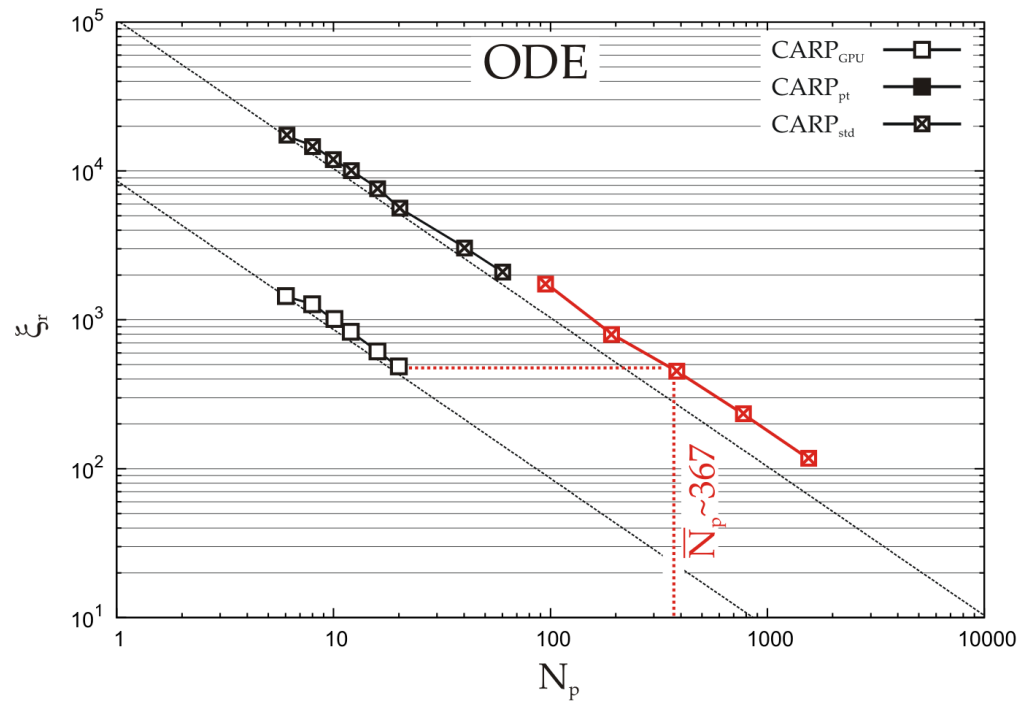
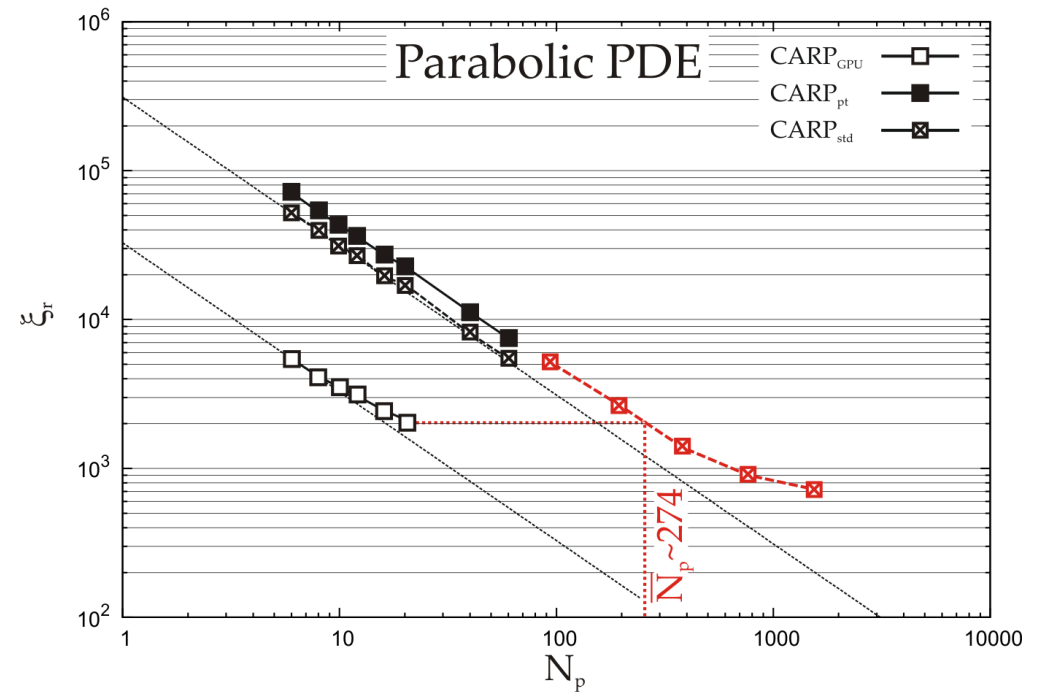
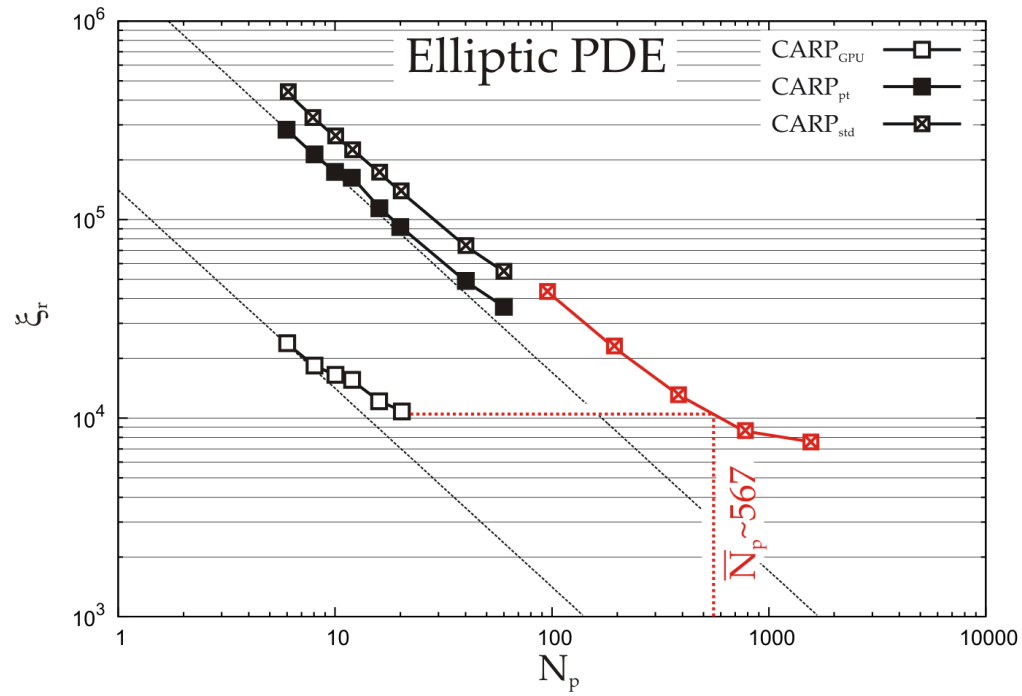


## (2) CARP Virtual Heart Solver Benchmark on CPU and GPU Clusters

**Figure 1:** A) Image-based high-resolution FE mesh of rabbit ventricles. Clipping plane exposes view on trabeculation and papillary muscles in the LV cavity. B) Shown is the activation sequence used for benchmarking. Wavefront propagation is initiated by delivering a transmembrane stimulus at the LV apex. C) Time traces are shown for  $V_m$  and  $\Phi_e$  at three different sites, close to the stimulus site (blue), halfway between apex and base (green) and at the latest activating site (red) at the base of the ventricles. D) Convergence history as a function of time.

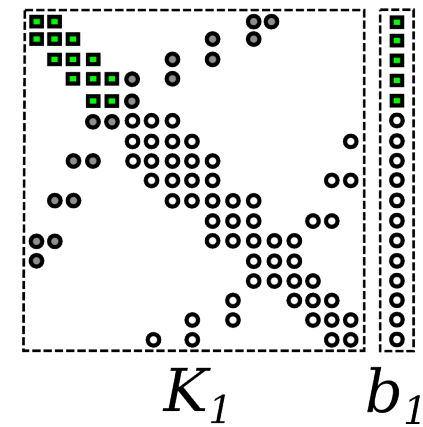
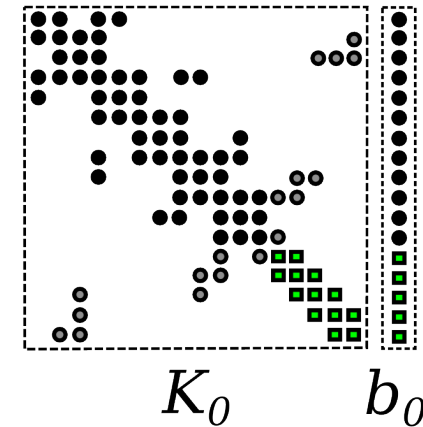
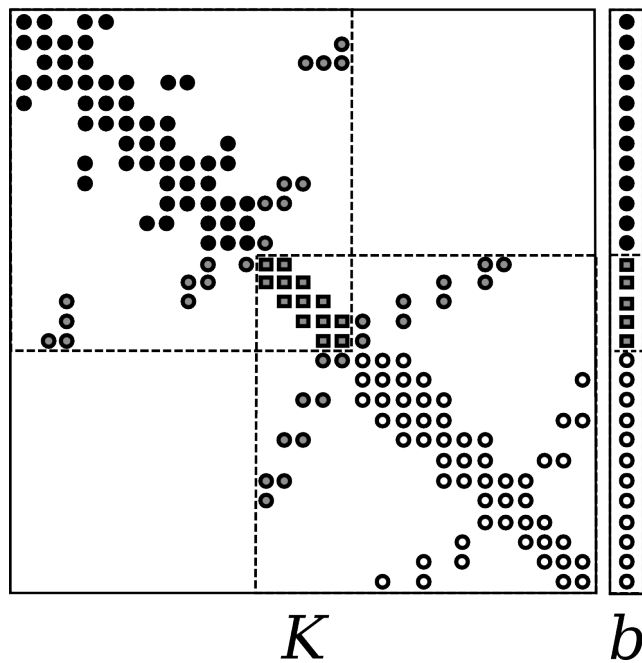
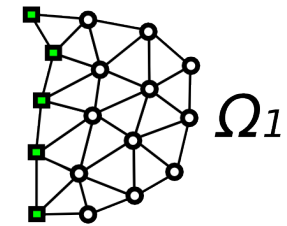
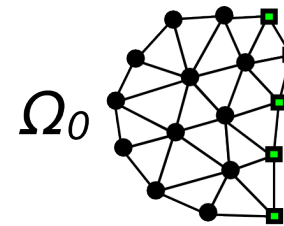
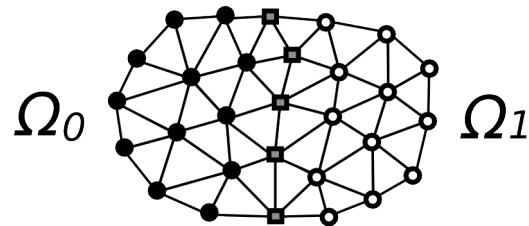
**Figure 2:** Realtime lag  $\xi_r$  as a function of  $N_p$  for the solution of elliptic PDE, parabolic PDE, ODE and overall compute time. Red traces were measured with the UK national supercomputing facility HECToR. Number of cores required to match fastest GPU setup,  $\bar{N}_p$ , are indicated in each panel.







### (3) Domain Decomposition: A Closer Look



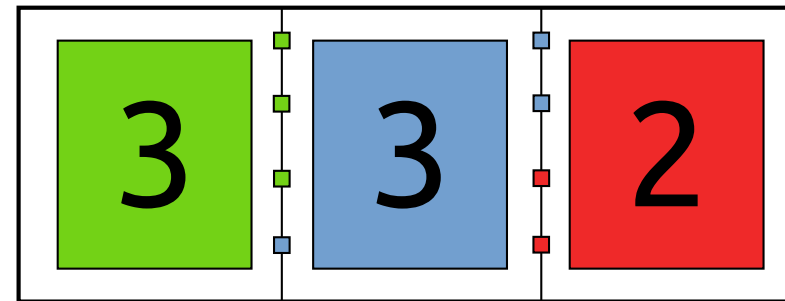
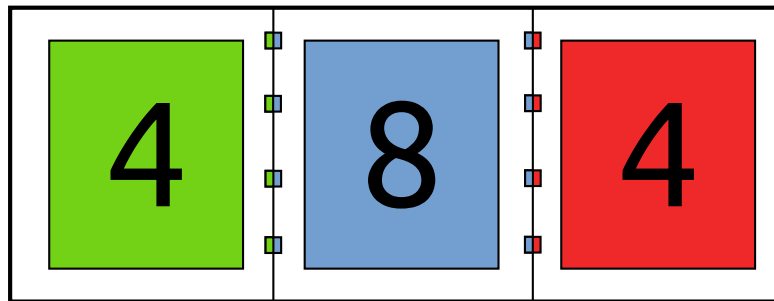
## Parallel Scalability Challenges: Data Accumulation

*Parallel linear algebra requires an accumulation of the data values on the boundary nodes.*

### What is holding us back?

- Unbalanced domain decomposition w.r.t. the boundary nodes.
- METIS balances the number of elements per process, i.e. computation
- Up to 100% variation in the boundary sizes, i.e. communication

## Performance Tuning: New Data Accumulation Scheme



Assign to every shared boundary node a unique master process. The master process is then responsible for the data accumulation of the boundary node. Use an optimization algorithm to evenly distribute the master processes.

## New Data Accumulation Scheme Advantages

The communication complexity is reduced to  $\mathcal{O}(2(r-1))$  with  $r$  processors participating, instead of  $\mathcal{O}(r(r-1))$  for the standard approach, leading to a balanced data accumulation.

### Standard Accumulation (STD)

*MPI process  $p$  executes:*

$\text{CopyToBuffer}(f_p, B_{p,q}, \tilde{f}_p)$	{Fill the communication buffer}
$\text{MPIAlltoall}(\tilde{f}_p, B_{p,q}, \tilde{g}_p, B_{p,q})$	{Send data to all neighbors}
$\text{AccumulateInPlace}(\tilde{g}_p, B_{p,q}, f_p)$	{Accumulate in place}

### Optimized Accumulation (OPT)

*MPI process  $p$  executes:*

$\text{MPIAlltoall}(f_p, S_{p,q}, \tilde{g}_p, R_{p,q})$	{Send data to masters}
$\text{Accumulate}(\tilde{g}_p, R_{p,q})$	{Accumulation on master processes}
$\text{MPIAlltoall}(\tilde{g}_p, R_{p,q}, f_p, S_{p,q})$	{Receive data from masters}

## CPU Execution Times for Accumulation Algorithm

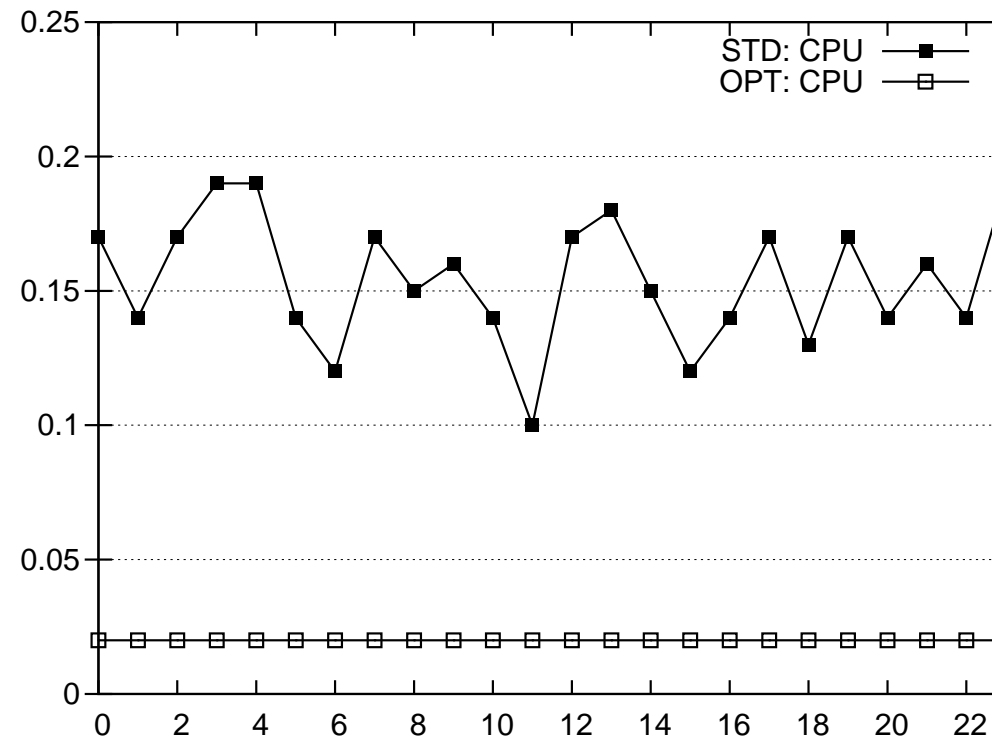


Figure 1: CPU execution times excluding MPI communication times in seconds for the standard (STD) and the optimized (OPT) accumulation algorithm for 24 MPI processes for the benchmark example measured with Scalasca.

## MPI Communication Times for Data Exchange

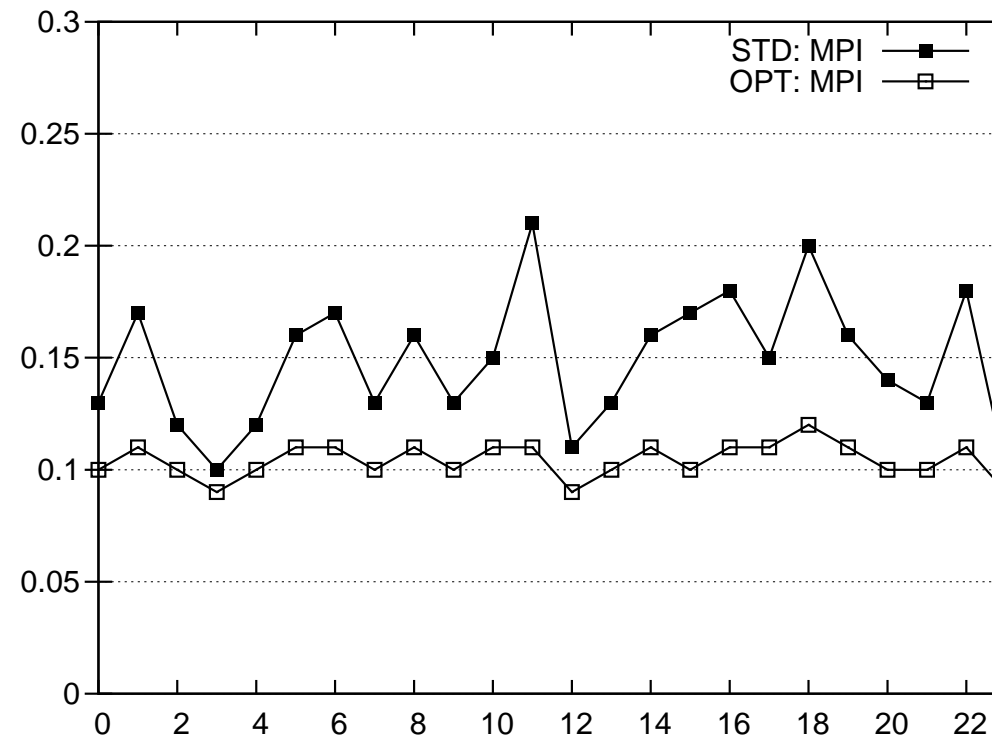


Figure 2: MPI communication times for the standard (STD) and the optimized (OPT) accumulation algorithm for 24 MPI processes for the benchmark example measured with Scalasca.

## Total Runtimes for the Full Accumulation Algorithm

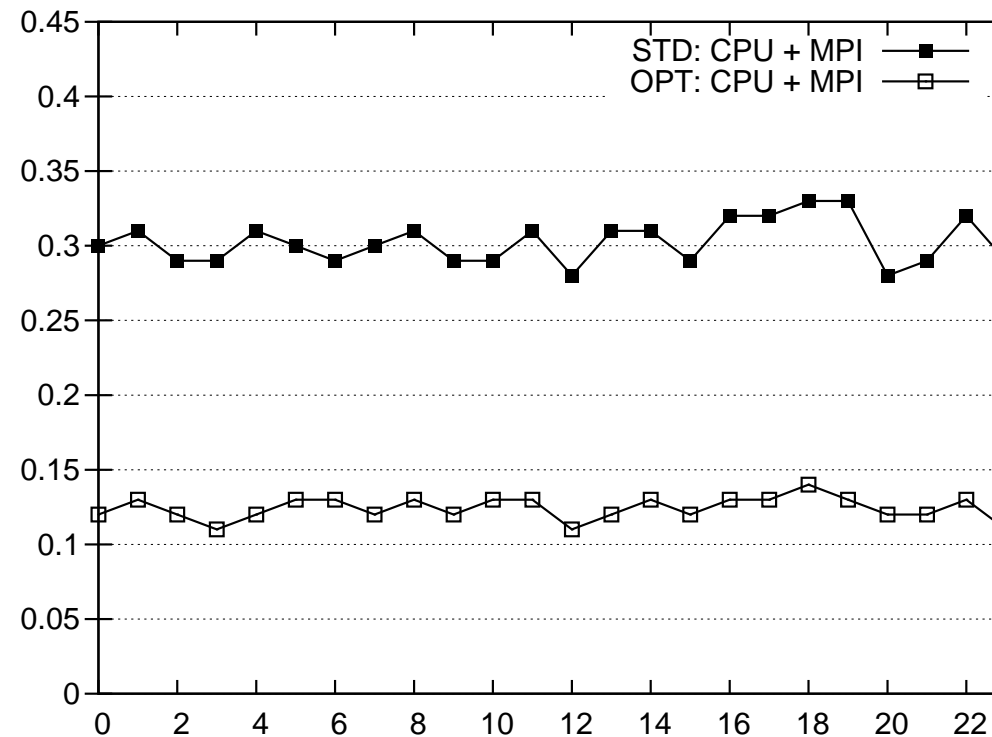


Figure 3: Total runtimes in seconds for the standard (STD) and the optimized (OPT) accumulation algorithm for 24 MPI processes for the benchmark example measured with Scalasca.

## Optimization Algorithm

Minimize the function

$$J = \sum_{p=1}^P J^p, \quad J^p = \sum_{q=1}^P (n_q^p - \bar{n}_q^p)^2, \quad p \in I \quad (8)$$

w.r.t. the given approximate uniform distribution of the master nodes.

$$\bar{n}_q^p := \left\lfloor \frac{\nu_q^p N^p + N^p}{P} \right\rfloor - \left\lfloor \frac{\nu_q^p N^p}{P} \right\rfloor, \quad p, q \in I \quad (9)$$

with the shift function defined as  $\nu_q^p := q + p - 2 \pmod{P}$ ,  $p, q \in I$  and  $N^p := |\hat{B}_p|$  the size of the restricted boundary node set  $\hat{B}_p := \{v \in B : p - 1 \equiv v - 1 \pmod{P}\}$ .



## Optimization Algorithm: Implementation

### Parallel efficiency is of concern!

- Strategy: Minimize  $J^p = \sum_{q=1}^P (n_q^p - \bar{n}_q^p)^2$ ,  $p \in I$ .
- Solve the local optimization problems independently in parallel!
- Randomized local search strategy inspired by the simulated annealing approach.
- Stronger requirement than necessary for the solution of the global optimization problem.
- Advantage: Fully parallel algorithm with no communication during the local optimization.
- Communication: Exchange the master node configuration between the processes.

## Optimization Algorithm: Results for Multigrid Hierarchy

Level	Processors			
	6	12	24	48
1	29461	44581	62756	83738
2	14003	20948	29225	38326
3	4721	6874	9305	11609
4	1285	1793	2180	2413
5	306	347	358	360
6	60	61	65	58
7	15	13	14	12
8	5	5	4	5
9	2	1	2	2
10	1		1	1

Table 1: The global number of boundary vertices for different levels of the AMG algorithm and different numbers of processors.

## Optimization Algorithm: Results for Multigrid Hierarchy

Level	Processors			
	6	12	24	48
1	0	0	4	146
2	0	0	10	168
3	0	2	14	372
4	0	12	70	420
5	0	12	88	262
6	0	4	38	82
7	0	2	16	18
8	0	0	6	8
9	0	0	2	4
10	0		2	2

Table 2: Computed values of the functional  $J$  for different levels of the AMG algorithm and different numbers of processors.

## MPI Communication Times for PCG-AMG Benchmark

PCG-AMG Benchmark: The tetrahedral mesh of a simplified model of a rabbit heart, with 5,082,272 elements and 862,515 vertices.

Processors	OPT			STD		
	min	mean	max	min	mean	max
6	0.03	0.05	0.06	0.02	0.07	0.08
12	0.04	0.05	0.06	0.03	0.06	0.14
24	0.09	0.11	0.12	0.10	0.15	0.21
48	0.27	0.27	0.28	0.52	0.53	0.58

Table 3: MPI communication times in seconds for the standard (STD) and optimized (OPT) accumulation algorithms measured with Scalasca.

## PCG-AMG Solver Performance

**Parallel efficiency of the PCG-AMG solver is significantly increased.**

Processors	Runtime		Efficiency	
	OPT	STD	OPT	STD
6	2.38	2.40	1.00	1.00
12	1.22	1.24	0.97	0.97
24	0.69	0.77	0.86	0.78
48	0.57	0.84	0.52	0.36

Table 4: Total runtimes in seconds and parallel efficiency for the PCG-AMG solver for two linear solves.

**Mephisto cluster:** Five compute nodes with two six-core Intel Xeon X5650 CPUs running at 2.67 GHz and 96 GB RAM per node and a Mellanox 40 Gb/s QDR-Infiniband network.

Thank you!