Communications in Applied Mathematics and Computational Science

TIME-PARALLEL GRAVITATIONAL COLLAPSE SIMULATION

ANDREAS KREIENBUEHL, PIETRO BENEDUSI, DANIEL RUPRECHT AND ROLF KRAUSE

vol. 12 no. 1 2017

msp

COMM. APP. MATH. AND COMP. SCI.
Vol. 12, No. 1, 2017
dx.doi.org/10.2140/camcos.2017.12.109

msp

TIME-PARALLEL GRAVITATIONAL COLLAPSE SIMULATION

ANDREAS KREIENBUEHL, PIETRO BENEDUSI,
DANIEL RUPRECHT AND ROLF KRAUSE

This article demonstrates the applicability of the parallel-in-time method Parareal to the numerical solution of the Einstein gravity equations for the spherical collapse of a massless scalar field. To account for the shrinking of the spatial domain in time, a tailored load balancing scheme is proposed and compared to load balancing based on number of time steps alone. The performance of Parareal is studied for both the subcritical and black hole case; our experiments show that Parareal generates substantial speedup and, in the supercritical regime, can reproduce Choptuik's black hole mass scaling law.

1. Introduction

Einstein's field equations of general relativity (GR) consist of ten coupled, nonlinear, hyperbolic-elliptic partial differential equations (PDEs). Because gravity couples to all forms of energy, there is an enormous dynamic range of spatiotemporal scales in GR. Hence, usually only the application of advanced numerical methods can provide solutions and in numerical relativity [1; 3] extensive use of high-performance computing (HPC) is made [32; 26].

Today, almost all HPC architectures are massively parallel systems connecting large numbers of compute nodes by a high-speed interconnect. In numerical simulations, the power of these systems can only be harnessed by algorithms that feature a high degree of concurrency; every algorithm with strong serial dependencies can only provide inferior performance on massively parallel computers. For the solution of PDEs, parallelization strategies have been developed mainly for spatial solvers. However, in light of the rapid increase in the number of cores in supercomputers, methods that offer additional concurrency along the temporal axis have recently begun to receive more attention.

The idea of parallelization in time was introduced in 1964 [35]. In the 1980s and 1990s, time and spacetime multigrid methods were studied [22; 23; 24]. More recently, the now widely used time-parallel method Parareal was proposed [31].

MSC2010: 35Q76, 65M25, 65Y05, 83C57.
Keywords: Einstein–Klein–Gordon gravitational collapse, Choptuik scaling, Parareal, spatial coarsening, load balancing, speedup.

Other recently introduced parallel-in-time methods are PFASST [33; 12], RIDC [9], or MGRIT [13]. A historical overview is offered in [17].

Given the demonstrated potential of parallel-in-time integration methods for large-scale parallel simulations [42], these methods could be beneficial for the numerical relativity community. However, their application is not straightforward and often it is unclear a priori if good performance can be achieved. In this article, we therefore investigate the *principal applicability* of the time-parallel Parareal method to solving Einstein's equations describing spherical, gravitational collapse of a massless scalar field. The system is also referred to as an Einstein–Klein–Gordon system because it is equivalent to a Klein–Gordon equation expressed in the context of GR, i.e., on a back-reacting, curved geometry. It defines a basic gravitational field theory and is of interest therefore not only in numerical relativity but also in, e.g., quantum gravity [25; 44; 29]. A summary of numerically derived results is given in [21]; the work by Choptuik [7] brought forward novel, physical results and is of particular interest here because we will show that Parareal correctly reproduces the expected mass scaling law.

Mathematical theory shows that Parareal performs well for diffusive problems with constant coefficients [19]. For diffusive problems with space- or time-dependent coefficients, numerical experiments show that Parareal can converge quickly too [30]. However, given the theory for basic constant-coefficient hyperbolic PDEs [19], it can be expected that Parareal applied to convection-dominated problems converges too slowly for meaningful speedup to be possible. Special cases with reasonable performance are discussed in [16], and for certain hyperbolic PDEs it was found that some form of stabilization is required for Parareal to provide speedup [18; 40; 11; 6]. Surprisingly, no stabilization is required for the equations describing gravitational collapse; we demonstrate that plain Parareal can achieve significant speedup. A detailed analytical investigation of why this is the case would definitely be of interest but is left out for future work. One reason could be that we solve in characteristic coordinates for which the discretization is aligned with the directions of propagation [16; 28].

In Section 2 we define the system of Einstein field equations that we solve using Parareal. In addition, we give details on the numerical approach and discuss the interplay between Parareal and the particular structure of the spatial mesh. In Section 3 we discuss the Parareal method. Then, in Section 4 numerical results are presented. Finally, in Section 5 we conclude with a summary and discussion.

2. Equations

2.1. ***Gravitational collapse.*** The Einstein field equations in Planck units normalized to $4\pi G/c^4 = 1$ are

$$G_{\mu\nu} = 2T_{\mu\nu}, \tag{2.1.1}$$

where $\mu, \nu \in \{0, 1, 2, 3\}$ index time (via 0) and space (via 1, 2, and 3).[1] Once the nongravitational matter content is specified by a definition of the energy-momentum tensor $T_{\mu\nu}$, possibly along with equations of state that together satisfy the continuity equations $\nabla^\mu T_{\mu\nu} = 0$, (2.1.1) defines a set of ten partial differential equations for ten unknown metric tensor field components $g_{\mu\nu}$.[2] In all generality, the equations are coupled, nonlinear, and hyperbolic-elliptic in nature. Six of the ten equations are hyperbolic evolution equations, while the remaining four are elliptic constraints on the initial data; they represent the freedom to choose spacetime coordinates. For the matter content, we consider a minimally coupled massless scalar field ϕ with energy-momentum tensor

$$T_{\mu\nu} = \nabla_\mu \phi \, \nabla_\nu \phi - \tfrac{1}{2} g_{\mu\nu} g^{\alpha\beta} \, \nabla_\alpha \phi \, \nabla_\beta \phi. \tag{2.1.2}$$

For the metric tensor field $g_{\mu\nu}$ in spherical symmetry, it is natural to introduce a parametrization in terms of Schwarzschild coordinates (t, r). Here, t is the time coordinate of a stationary observer at infinite radius r, which measures the size of spheres centered at $r = 0$. In [7] the resulting Einstein field equations are analyzed numerically. In particular, adaptive mesh refinement [4] is used to resolve the black hole formation physics. In [20] the same investigation is carried out in double null or characteristic coordinates (τ, ρ) without mesh refinement (see, however, [39; 43]). Finally, in [29] the effect of quantum gravity modifications on the collapse is studied in adjusted characteristic coordinates. Here we use characteristic coordinates (τ, ρ) as well but exclude quantum gravity modifications. Also, for simplicity, we will refer to τ as a time coordinate and to ρ as a space coordinate.

Making the ansatz

$$g_{\mu\nu} \, \mathrm{d}x^\mu \, \mathrm{d}x^\nu = -2\partial_\rho r H \, \mathrm{d}\tau \, \mathrm{d}\rho + r^2 (\mathrm{d}\vartheta^2 + [\sin(\vartheta) \, \mathrm{d}\varphi]^2) \tag{2.1.3}$$

for the metric tensor field and using an auxiliary field h for the spacetime geometry along with an auxiliary field Φ for the matter content, the complete field equations are

$$\partial_\tau r = -\tfrac{1}{2} h, \qquad \partial_\tau \Phi = \frac{(H - h)(\Phi - \phi)}{2r} \tag{2.1.4}$$

for r and Φ and

$$\partial_\rho \phi = \frac{\partial_\rho r}{r} (\Phi - \phi), \qquad \partial_\rho H = \frac{\partial_\rho r}{r} H (\Phi - \phi)^2, \qquad \partial_\rho h = \frac{\partial_\rho r}{r} (H - h) \tag{2.1.5}$$

for ϕ, H, and h [20]. Overall the system can be seen as a wave equation for the massless scalar field ϕ on a back-reacting, curved geometry. Boundary conditions

[1] We omit the addition of the cosmological constant term $\Lambda g_{\mu\nu}$ on the left-hand side in (2.1.1) because observations suggest $0 < \Lambda \ll 1$ (see, e.g., [27]); the term's impact on black hole formation as studied here can be neglected.

[2] We use the Einstein summation convention.

at $(\tau, \rho = \tau)$ are $r = 0$ and regularity of Φ, ϕ, H, and h, which implies $\Phi = \phi$ and $H = h$ at the boundary [10; 28]. Consistent initial data at $(\tau = 0, \rho)$ are

$$r = \tfrac{1}{2}\rho, \qquad \Phi = (1 + \rho\,\partial_\rho)\phi, \tag{2.1.6}$$

where we choose for ϕ the Gaussian wave packet

$$\phi(0, \rho) = \phi_0 \frac{\rho^3}{1+\rho^3} \exp\left(-\left[\frac{\rho - \rho_0}{\delta_0}\right]^2\right). \tag{2.1.7}$$

We also performed tests for initial data similar in shape to the hyperbolic tangent function much like Choptuik did in [7] for purely serial time stepping. Since in this case we found Parareal's performance to resemble strongly that for the case of the Gaussian wave packet, we do not include these results here. The initial scalar field configuration is thus characterized by an amplitude ϕ_0, mean position ρ_0, and width δ_0. Depending on the value of these parameters, the solution to (2.1.4) and (2.1.5) can describe a bounce of the wave packet or black hole formation near the boundary at $r = 0$. A black hole appears when the outward null expansion

$$\Theta^+ = \frac{1}{r}\sqrt{\frac{2h}{H}}, \tag{2.1.8}$$

which measures the relative rate of change of a cross-sectional area element of a congruence of outgoing null curves, approaches zero [36]. The black hole mass is

$$M = \tfrac{1}{2}r, \tag{2.1.9}$$

evaluated at the point (τ^+, ρ^+) toward which Θ^+ vanishes.

2.2. ***Numerical solution.*** The numerical grid is depicted in Figure 1, left. It is parametrized by the characteristic coordinates τ and ρ, which are used for numerical integration; τ is used as the coordinate representing time and ρ as the coordinate representing space. Integration thus takes place on a right triangle with initial data defined along the lower right-hand leg. Clearly, the spatial domain becomes smaller as the solution is advanced in τ. Note that the domain is not exactly a right triangle because at the upper-most corner a small subtriangle is missing. This "buffer" zone of extent λ is needed for the spatial part of the numerical stencil to fit. The computational domain thus consists of all points $(\tau, \rho) \in [0, L - \lambda] \times [0, L]$ with $L = 80$, $\lambda = 0.625$, and $\rho \geq \tau$.

As a time-stepping method for the solution of the equations in (2.1.4), we use a second-order Lax–Wendroff Richtmyer two-step method on a fine spacetime grid [28]. To employ the time-parallel method Parareal (see Section 3), we need a second, computationally cheap, time-integration method. Here, we choose the explicit first-order Euler method on a coarse spacetime mesh. For Parareal to be efficient, the cost of the coarse method has to be small compared to that of the fine one: by choosing

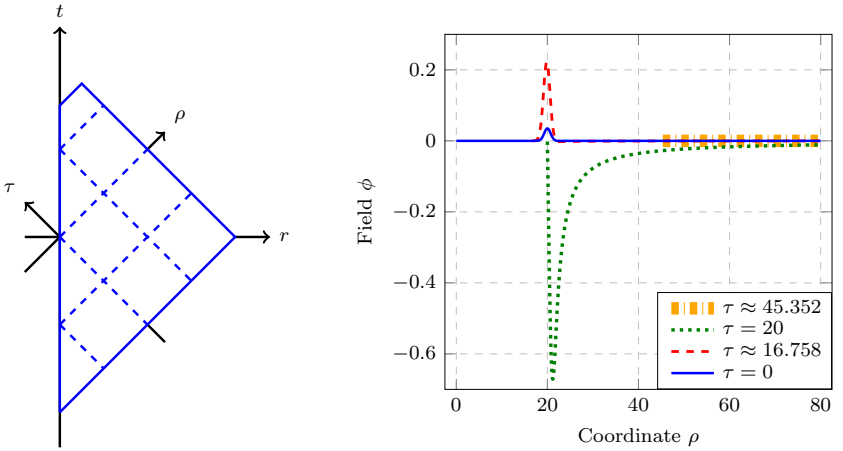

Figure 1. Left: the numerical domain. It is parametrized by the characteristic coordinates τ and ρ. Right: subcritical gravitational scalar field evolution and scalar field solution snapshots for a black-hole-free setting. The peak of the Gaussian evolves along the constant coordinate value $\rho \approx 20$, which is also when the bounce occurs in τ.

a simple first-order method on the coarse grid for $\mathscr{C}$, we obtain a good coarse-to-fine ratio (see Section 3.4). For optimal speedup, the right balance between the difference in accuracy and difference in cost between $\mathscr{C}$ and $\mathscr{F}$ has to be found.

For the integration in space of the equations in (2.1.5), we use a second-order Runge–Kutta method [28]. Snapshots of scalar field evolution resulting from the chosen fine grid discretization are shown in Figure 1, right, where ϕ evolves along constant lines of ρ until a bounce occurs at $r = 0$. The figure also shows how the size of the domain decreases during the evolution: for $\tau = 0$ the left boundary is at $\rho = 0$ while for $\tau = 20$ it is at $\rho = 20$.

2.3. ***Mass scaling.*** In practice, the simulation terminates when a black hole forms because H grows without bound in this case (see [10] for details). Figure 2, left, provides a simplified illustration of a black hole region (dotted portion) and shows where the simulation comes to a halt (dashed line). Thus, to determine the black hole mass M, we record minimal expansion values via the scalar $(r\Theta^+)_{\mathrm{mi}} = \min_\rho\{r\Theta^+\}$ derived from (2.1.8). The last such recorded minimal value before the termination of the simulation defines a characteristic coordinate (τ^+, ρ^+) (see again Figure 2, left), which we can use to define an r and M via (2.1.9). The scalar $(r\Theta^+)_{\mathrm{mi}}$ approaches 0 when (τ, ρ) nears (τ^+, ρ^+), as is shown in the lower portion of Figure 2, right.

Based on numerical experiments, Choptuik presents, among other things, a relation between the amplitude ϕ_0 of the Gaussian in (2.1.7) and the black hole mass M [7]. He shows that there is a critical value $\phi_0^\star$ such that for $\phi_0 < \phi_0^\star$ there is a bounce (subcritical case), while for $\phi_0 > \phi_0^\star$ there is a black hole (supercritical case).

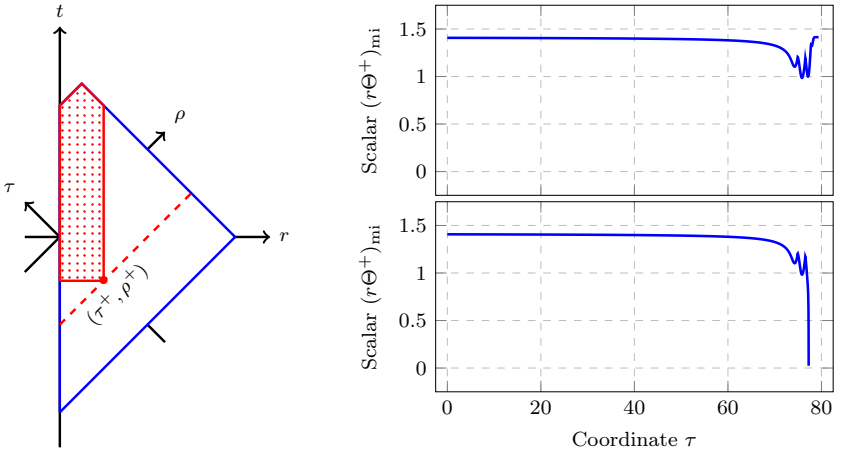

Figure 2. Illustrations to clarify supercritical gravitational collapse. Left: the simulation terminates at τ^+, when a black hole forms at ρ^+. Right: minimal weighted outward null expansion indicating a bounce (top) and black hole formation (bottom) are shown.

Based thereon, he demonstrates that the black hole mass scales with $\phi_0 - \phi_0^\star > 0$ according to the law $M \propto (\phi_0 - \phi_0^\star)^\gamma$ with γ being a positive constant of the same value for various initial data profiles. We demonstrate that Parareal can correctly capture this black hole mass scaling law although our coarse-level Euler method alone cannot. Also, Parareal requires less wall-clock time than $\mathcal{F}$, which can be beneficial for the investigation of the high-accuracy-demanding critical solution [7; 21] that requires the simulation of numerous black holes [20]. This analysis however is omitted in this article and left for future work.

3. Parareal

3.1. ***Algorithm.*** Parareal [31] is a method for the solution of initial value problems

$$\partial_\tau u(\tau) = f(\tau, u(\tau)), \quad u(0) = u_0, \ 0 \le \tau \le T. \tag{3.1.1}$$

Here, as is outlined in the previous section, f comes from discretizing (2.1.4) and (2.1.5), and $T = L - \lambda$ marks the end time. Parareal starts with a decomposition of the time domain into N_{pr} temporal subintervals (TSs) defined in terms of times τ^p such that

$$[\tau^1, \tau^2] \cup \cdots \cup [\tau^{N_{\mathrm{pr}}-1}, \tau^{N_{\mathrm{pr}}}] = [0, L - \lambda]. \tag{3.1.2}$$

Now denote by $\mathcal{F}$ some serial time-integration method of high accuracy and cost (in our case this is the second-order Lax–Wendroff Richtmyer two-step method) and by $\mathcal{C}$ a cheap and possibly much less accurate method (in our case this is the explicit first-order Euler method). Instead of running the fine method subinterval

by subinterval serially in time, Parareal performs the iteration

$$u_{[i+1]}^{p+1} = \mathscr{C}(u_{[i+1]}^{p}) - \mathscr{C}(u_{[i]}^{p}) + \mathscr{F}(u_{[i]}^{p}), \tag{3.1.3}$$

where superscripts index time or process number $p \in \{1, \ldots, N_{\mathrm{pr}}\}$ and subscripts iterations $i \in \{1, \ldots, N_{\mathrm{it}}\}$. The advantage is that the expensive computation of the fine method can be performed in parallel over all TSs at once. Here, we assume that the number of TSs is equal to the number N_{pr} of cores (or *processes*) used for the time direction. Good speedup can be obtained if $\mathscr{C}$ is fast in comparison to $\mathscr{F}$ but still accurate enough for Parareal to converge rapidly. See Section 3.4 for a more detailed discussion of Parareal's speedup.

In Section 2.2 we hinted at the interchangeability of the characteristic coordinates τ and ρ for the numerical integration. Therefore, theoretically, Parareal could *also* be used for the spatial integration to *simultaneously* parallelize both time and space. However, such an interweaving of two Parareal iterations is not discussed in this article; it is put aside for future work.

3.2. ***Spatial coarsening in Parareal.*** In order to make $\mathscr{C}$ cheaper and improve speedup, we not only use a less accurate time stepper for $\mathscr{C}$ but also employ a coarsened spatial discretization with a reduced number of degrees of freedom. Therefore, we need a spatial interpolation $\boldsymbol{I}$ and restriction $\boldsymbol{R}$ operator. In this case (see, e.g., [14]), the Parareal algorithm is given by

$$u_{[i+1]}^{p+1} = \boldsymbol{I}\mathscr{C}(\boldsymbol{R}u_{[i+1]}^{p}) - \boldsymbol{I}\mathscr{C}(\boldsymbol{R}u_{[i]}^{p}) + \mathscr{F}(u_{[i]}^{p}). \tag{3.2.1}$$

As the restriction operator $\boldsymbol{R}$, we use point injection. For the interpolation operator $\boldsymbol{I}$, we use polynomial (i.e., Lagrangian) interpolation of order 3, 5, and 7.[3] It has been shown that, even for simple toy problems, convergence of Parareal can deteriorate if spatial coarsening with low-order interpolation is used. As demonstrated in Section 4.1, this also holds true for the problem studied here.

3.3. ***Implementation.*** We have implemented two different realizations of Parareal. In a "standard" version $\mathscr{P}_{\mathrm{st}}$ (see Listing 1, left), the Parareal correction is computed on each TS up to a uniformly prescribed iteration number. In contrast, in the "modified" implementation $\mathscr{P}_{\mathrm{mo}}$ (see Listing 1, right), Parareal corrections are only performed on TSs where the solution may not yet have converged. Because Parareal always converges at a rate of at least one TS per iteration, we only iterate on a TS if its assigned MPI rank is greater than or equal to the current Parareal iteration number (see line 8 in Listing 1, right). Otherwise, no further iterations are needed or performed, and the process remains idle. Thus, as the iteration progresses, more and

[3] We also tested barycentric interpolation [5; 15] but found the performance in terms of runtimes and speedup (see Sections 3.4 and 4) to be inferior.

```
1  if p > 1 then // Initialization
2    Coarse(co; τ^1 → τ^p)
3  Interp(co ↦ fi[0])
4  if p < N_pr then // Prediction
5    Coarse(co; τ^p → τ^{p+1})
6  Interp(co ↦ fi[2])
7  for i = 1 : N_it do // Iteration
8    if p < N_pr then
9      Fine(fi[0]; τ^p → τ^{p+1})
10   fi[1] = fi[0]
11   fi[1] −= fi[2]
12   if p > 1 then
13     MPI_Recv(fi[0]; p ⇐ p−1)
14   else
15     Init(fi[0])
16   Restrict(fi[0] ↦ co)
17   if p < N_pr then
18     Coarse(co; τ^p → τ^{p+1})
19   Interp(co ↦ fi[2])
20   fi[1] += fi[2]
21   if p < N_pr then
22     MPI_Send(fi[1]; p ⇒ p+1)
```

```
1  if p > 1 then // Initialization
2    Coarse(co; τ^1 → τ^p)
3  Interp(co ↦ fi[0])
4  if p < N_pr then // Prediction
5    Coarse(co; τ^p → τ^{p+1})
6  Interp(co ↦ fi[2])
7  for i = 1 : N_it do // Iteration
8    if p >= i then
9      j = (i+1) % 2
10     k = i % 2
11     if p < N_pr then
12       Fine(fi[j]; τ^p → τ^{p+1})
13     if p > i then
14       MPI_Recv(fi[k]; p ⇐ p−1)
15       fi[j] −= fi[2]
16       Restrict(fi[k] ↦ co)
17       if p < N_pr then
18         Coarse(co; τ^p → τ^{p+1})
19       Interp(co ↦ fi[2])
20       fi[j] += fi[2]
21     if p < N_pr then
22       MPI_Send(fi[j]; p ⇒ p+1)
```

Listing 1. Pseudocode for the standard and modified Parareal implementations. Variable "co" denotes the coarse grid solution and "fi" an array of three fine grid buffers. Left: the standard Parareal implementation $\mathcal{P}_{\text{st}}$. Right: the modified Parareal implementation $\mathcal{P}_{\text{mo}}$.

more processes enter an idle state. In an implementation to be realized in future work, the criterion for convergence used here will be replaced by a check for some residual tolerance [2]. This could negatively affect the observed performance since it requires essentially one more iteration to compute the residual.[4] It also bears mentioning that it has very recently been demonstrated that parallel-in-time integration methods are good candidates to provide algorithm-based fault tolerance [34; 41].

Another difference between the standard and modified implementations is that in the former, after each time-parallel fine evolution, a copy of the fine-grid solution has to be created (see line 10 in Listing 1, left). In the modified Listing 1, right, this copying is circumvented by the use of two alternating indices "j" and "k" in lines 9 and 10, respectively. The iteration number determines their values, which in turn determines the fine-grid solution buffer that is used to send or receive data by means of the corresponding MPI routines (see lines 14 and 22 in Listing 1, right). The two implementations also have slightly different requirements in terms of storage.

[4] In [2] a version of Parareal is discussed that can be used to proceed the integration beyond a given end time. It is based on an optimized scheduling of those tasks which become idle in our implementation.

As can be seen in line 15 in Listing 1, left, in $\mathcal{P}_{\text{st}}$ on the first TS or, equivalently, for the first MPI rank, the fine-grid solution has to be assigned initial data at the beginning of each iteration. This requires one additional buffer to be held in storage. Other than that both implementations need one coarse-grid solution buffer and three fine-grid buffers for each TS.

3.4. ***Speedup.*** We denote by R_{co} the coarse and by R_{fi} the fine time stepper's runtime. Recalling that N_{it} denotes the number of iterations required for Parareal to converge given N_{pr} processes, Parareal's theoretically achievable speedup is

$$S = \left[\left(1 + \frac{N_{\text{it}}}{N_{\text{pr}}}\right)\frac{R_{\text{co}}}{R_{\text{fi}}} + \frac{N_{\text{it}}}{N_{\text{pr}}}\right]^{-1} \leq \min\left\{\frac{N_{\text{pr}}}{N_{\text{it}}}, \frac{R_{\text{fi}}}{R_{\text{co}}}\right\}, \tag{3.4.1}$$

as is discussed, e.g., in [33]. The estimate is valid only for the ideal case, where runtimes across subintervals are perfectly balanced. In the presence of load imbalances in time, however, i.e., differences in the runtimes of $\mathscr{C}$ and $\mathscr{F}$ across TSs, maximum speedup is reduced [30]. Because the spatial domain we consider is shrinking in time, a tailored decomposition of the time axis has to be used to provide well balanced computational load, as is discussed in the next section.

3.5. ***Load balancing.*** Because we integrate over a triangular computational space-time domain (see Figure 1, left), a straightforward, uniform partitioning of the time axis results in imbalanced computational load in time. The first load balancing (LB) strategy, to which henceforth we will refer as LB1, is based on this straightforward, basic decomposition of the time axis. It assigns to each TS the same *number* of time steps without regard to their computational *cost*. Because of the shrinking domain, TSs at later times carry fewer spatial degrees of freedom so that the per-process runtimes R_{co}^p and R_{fi}^p of the coarse and fine time steppers, respectively, are larger for the earlier TSs than for the later ones. Figure 3, left, shows how this partition leads to an imbalanced computational load in time because the portion extending across the "early-middle" TS $[e, m]$ covers a larger area and thus a larger number of grid points than the portion over the "middle-late" TS $[m, l]$.

Figure 3 suggests that early-in-time TSs should have a shorter extent in time than later ones. Thus, in the second strategy, to which in the following we will refer as LB2, we also consider the *cost* of time steps in order to balance the runtime $R_{\text{co}}^p + R_{\text{fi}}^p$ over all processes p. We use a decomposition of the time axis in TSs such that the sum of the total coarse and fine runtime is balanced over all TSs, i.e., such that $R_{\text{co}} + R_{\text{fi}} = N_{\text{pr}}(R_{\text{co}}^p + R_{\text{fi}}^p)$ for any process p. This is done by a bisection approach, making use of the fact the we use explicit rather than implicit time integrators (see the discussion in [30]) and thus that the cost of a time step from τ to $\tau + \Delta\tau$ is directly proportional to the number of spatial degrees of freedom at τ. Therefore, the total spacetime domain is first divided into two parts of roughly

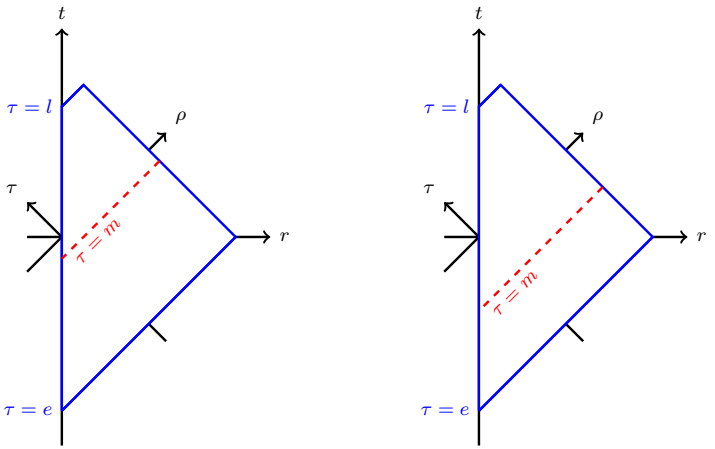

Figure 3. Illustration of two different approaches for the decomposition of the time domain. Left: imbalanced load in time from load balancing LB1. Right: balanced load in time from load balancing LB2.

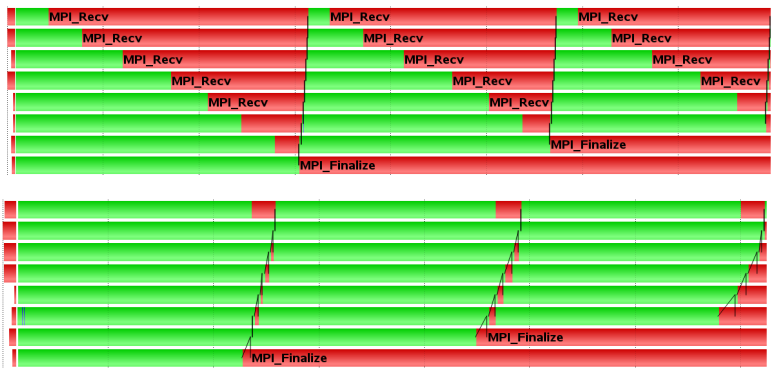

Figure 4. Vampir traces for the implementation $\mathcal{P}_{\text{mo}}$ with $(N_{\text{pr}}, N_{\text{it}}) = (8, 3)$ for two different load balancing strategies. Top: Vampir trace for LB1. The Parareal runtime is $R_{\text{pa}} = 7.964$ s. Bottom: Vampir trace for LB2. The Parareal runtime is $R_{\text{pa}} = 5.436$ s.

equal number of grid points as is sketched in Figure 3, right. Then, each part is divided again and again until the required number of TSs is reached. Note that this limits the possible numbers of TSs to powers of 2.

Figure 4 shows Vampir[5] traces for one simulation featuring LB1 (Figure 4, top) and one LB2 (Figure 4, bottom). The horizontal axes correspond to runtime, while the vertical axes depict MPI rank numbers from 1 (lower) to 8 (upper). In each case, three Parareal iterations are performed. Green regions indicate the coarse and fine integrators carrying out work. Time spent in MPI receives (including waiting time) is shown in red. We observe how LB1 leads to load imbalance and incurs

[5] https://www.vampir.eu/

significant wait times in processes handling a later TS. In contrast, the processes' idle times (shown in red) in MPI receives are almost invisible in the case of LB2. Elimination of wait times leads to a significant reduction in runtime and increase in speedup, as will be shown in Section 4.

4. Results

Speedup and runtime measurements were performed on the Cray XC40 supercomputer Piz Dora[6] at the Swiss National Supercomputing Center (CSCS) in Lugano. It features 1256 compute nodes, which all hold two 12-core Intel Xeon E5-2690v3 processors. This results in a total of 30144 compute cores and a peak performance of 1.254 PFlops; it occupies position 56 in the Top500 November 2014 list.[7] On Piz Dora, we used the GNU Compiler Collection[8] version 4.9.2 and the runtimes we provide do not include the cost of I/O operations. Some simulations measuring convergence were performed on a machine located at the Università della Svizzera italiana that is maintained by members of the Institute of Computational Science of the Faculty of Informatics.[9]

For the results presented in the following, we use a coarse grid resolution of $(\Delta\tau)_{\text{co}} = (\Delta\rho)_{\text{co}} = \Delta_{\text{co}} = L/2048 \approx 0.039$ and a fine grid resolution of $\Delta_{\text{fi}} = \Delta_{\text{co}}/8 \approx 0.005$. We have also determined a *reference* solution to approximately measure the serial fine stepper's discretization error. For this we have used again the serial fine time stepper but with a step size of $\Delta_{\text{re}} = \Delta_{\text{fi}}/4 \approx 0.001$.

4.1. ***Subcritical.*** First we consider the subcritical case, where no black holes form. Figure 5 shows for $N_{\text{pr}} = 256$ and two different sets of initial data parameters the relative defect

$$D_{[i]} = \frac{\|r_{[i]} - r_{\text{fi}}\|_2}{\|r_{\text{fi}}\|_2}, \tag{4.1.1}$$

which measures the difference between the Parareal solution $r_{[i]}$ after i iterations and the serial fine solution r_{fi} as a function of the characteristic coordinate τ.

In Figure 5, left, we use the initial data parameters $(\phi_0, \rho_0, \delta_0) = (0.035, 20, 1)$, which results in an "early" bounce of the wave packet at about $\tau = 20$. For the simulations in Figure 5, right, the values are $(\phi_0, \rho_0, \delta_0) = (0.01, 75, 1)$, which leads to a "late" bounce at about $\tau = 75$. Defects are plotted for $N_{\text{it}} \in \{1, 2, 3, 4\}$ along with the estimated discretization errors $\|r_{\text{co}} - r_{\text{re}}\|_2/\|r_{\text{fi}}\|_2$ of serial coarse and $\|r_{\text{fi}} - r_{\text{re}}\|_2/\|r_{\text{fi}}\|_2$ of serial fine solutions. We observe that in Figure 5, left, the data for $N_{\text{it}} = 3$ is somewhat jagged because for LB2 there are various start

[6] http://www.cscs.ch/computers/piz_daint_piz_dora/

[7] http://www.top500.org/list/2014/11

[8] https://gcc.gnu.org

[9] https://www.ics.usi.ch/index.php/ics-research/resources

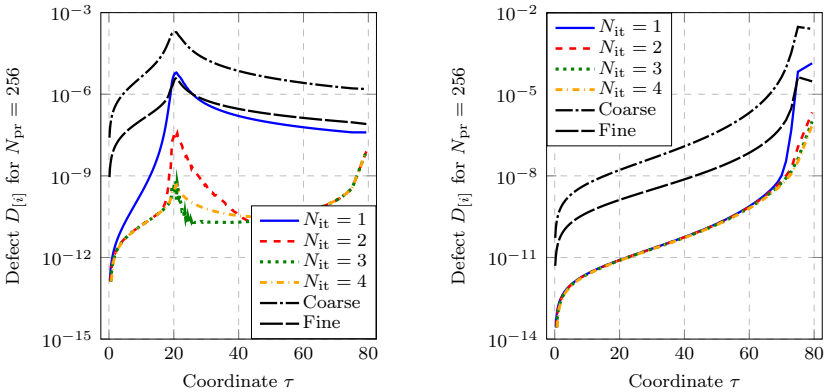

Figure 5. Defect in r between Parareal and the fine method over time for fixed $N_{pr} = 256$. Left: early bounce scenario. Right: late bounce situation.

and end times of TSs near the bounce region. In any case, Parareal converges in two iterations: for $N_{it} = 2$, the defect is below the discretization error for all τ. In fact, without the bounce region near $\tau = 20$, only one iteration would be required for convergence. For the late bounce scenario in Figure 5, right, we also observe that the rate of convergence at the final time $\tau = L - \lambda$ gives an indication of the convergence at all τ. In the following we thus focus on convergence at the final time. Convergence for the other evolved field Φ is not shown but was found to be at least as good as for r.[10]

Figure 6, left and middle, illustrate the defect of Parareal at the end of the simulation at $\tau = L - \lambda$ for various values of N_{pr} with third-order interpolation (left) and fifth-order interpolation (middle). For third-order interpolation, Parareal does not converge at all. The configuration stalls at a defect of about 10^{-2} until the iteration count equals N_{pr}. There, Parareal converges by definition but cannot provide any speedup. In contrast, Parareal shows good convergence behavior for fifth-order interpolation. For N_{pr} less than 64, the defect of Parareal falls below the approximate discretization error of the fine method after a single iteration. Otherwise, for $N_{pr} \geq 64$ up to $N_{pr} = 512$, two iterations are required.

The resulting speedups with correspondingly adjusted values for N_{it} are shown in Figure 6, right, for both load balancing strategies (see the discussion in Section 3.5). In addition, the projected speedup according to (3.4.1) is shown. The fine-to-coarse ratio R_{fi}/R_{co} was determined experimentally and found to be about 74. Up to $N_{pr} = 64$, for the advanced load balancing, speedup closely mirrors the theoretical curve while the basic load balancing performs significantly worse. For $N_{pr} \geq 64$, measured speedups fall short of the theoretical values, peak at $N_{pr} = 256$, and

[10]Convergence seems to be unaffected by the load balancing. In tests not documented here, we found that for LB1 it takes two iterations for Parareal to converge as well.

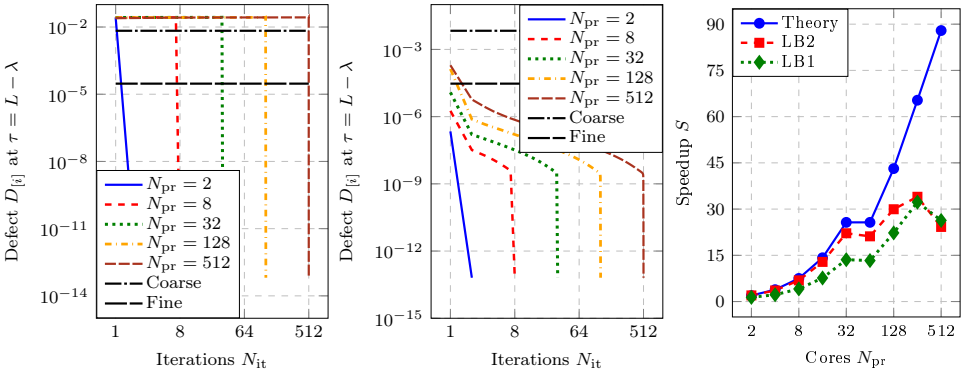

Figure 6. Parareal's performance for the subcritical case in terms of convergence for polynomial interpolation orders 3 and 5 and in terms of speedup. Left: defect for late bounce and interpolation order 3. Middle: defect for late bounce and interpolation order 5. Right: Parareal speedup for fifth-order interpolation.

then start to decrease. Note that the theoretical model (blue line in Figure 6, right) does take into account the scaling limit from the serial correction step according to Amdahl's law. The difference between theory and measured speedup is therefore due to other overheads (communication and transfer between meshes) as analyzed below.

Although the load balancing strategy LB2 results in significantly better speedup than the basic approach LB1, the peak value provided by both schemes is essentially the same. This is because, for increasingly large numbers of cores, the computational load per TS eventually becomes small and imbalances in computational load insignificant. Instead, runtime is dominated by overhead from, e.g., communication in time. The communication load is independent of the chosen load balancing and depends solely on the number of TSs; for every TS one message has to be sent and received once per iteration (save for the first and last TS). Therefore, it can be expected that ultimately both approaches to load balancing lead to comparable peak values. Below we demonstrate that the saturation in speedup is related to a significant increase in time spent in MPI routines; eventually, communication cost starts to dominate over the computational cost left on each time slice and the time parallelization saturates just as spatial parallelization does.

Figure 7 illustrates the reason behind the drop-off in speedup beyond $N_{pr} = 256$. First, define

$$R^p_{pa} = R^p_{co} + R^p_{fi} + \sum_{st} R^p_{st}, \tag{4.1.2}$$

where R^p_{st} denotes runtime spent in *stages* that are *different* from coarse and fine integration on the TS assigned to process p. For now, we consider only overhead from sending and receiving data as well as from interpolation; other overheads are

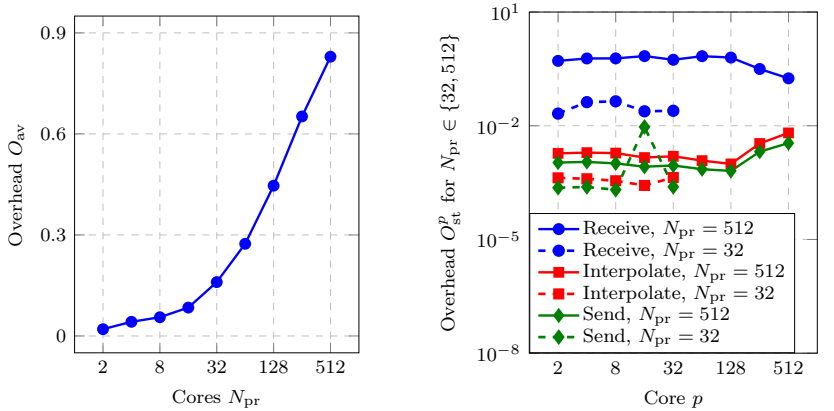

Figure 7. Overhead from communication and other sources increases with N_{pr}, which leads to Parareal's speedup decay. Left: average overhead. Right: overhead caused by three different Parareal stages.

not further analyzed here. Next, we introduce the *total* overhead on a TS as the sum of all stage runtimes or

$$O^p_{to} = \sum_{st} R^p_{st}, \tag{4.1.3}$$

which is also the runtime spent *neither* in the coarse *nor* fine integrator for a given p. The *average* overhead is now defined as the geometric mean value of O^p_{to} over all TSs, which is

$$O_{av} = \frac{\sum_{p=1}^{N_{pr}} O^p_{to}}{N_{pr}}. \tag{4.1.4}$$

Finally, we define the relative overhead for individual *stages* on a TS as

$$O^p_{st} = \frac{R^p_{st}}{R^p_{pa}}, \tag{4.1.5}$$

where R^p_{pa} is the runtime of Parareal at processor p. Ideally, as is assumed for the derivation of the speedup model given in (3.4.1), R^p_{co} and R^p_{fi} are the dominant costs. In this case, $R^p_{co} + R^p_{fi} \approx R^p_{pa}$ so that according to (4.1.2) we have $O^p_{to} \approx 0$ and therefore $O_{av} \approx 0$ by definition. However, as can be seen in Figure 7, left, O_{av} is small only for small values of N_{pr}. For $N_{pr} \geq 32$ it increases rapidly, which indicates that the overhead from communication and other sources starts to play a more dominant role when N_{pr} is increased.

Figure 7, right, shows the relative overhead from (4.1.5) for $N_{pr} \in \{32, 512\}$ and $p \in \{1, \ldots, N_{pr}\}$ for the three different stages st $\in$ {Interpolation, Send, Receive}; "Send" and "Receive" refer to the corresponding MPI routines. There is a significant

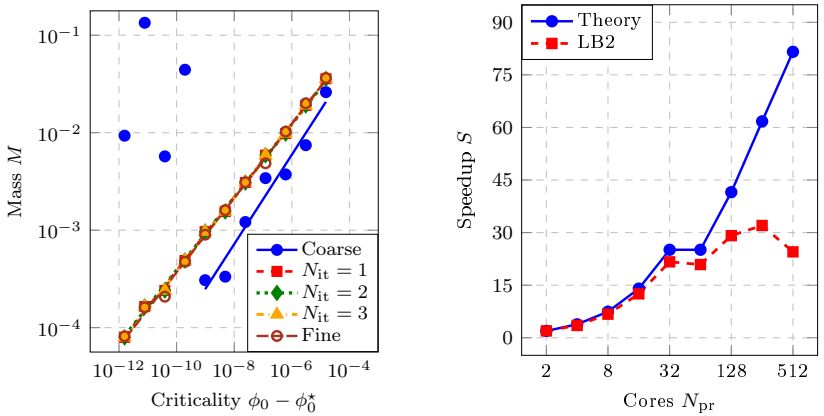

Figure 8. Parareal's performance for the supercritical case. Left: Choptuik scaling from Parareal. Right: Parareal speedup.

increase in relative overhead in all three stages as the number of cores grows, causing the eventual drop-off in speedup for increasing N_{pr}.

4.2. ***Supercritical.*** We consider now the more complex case in which a black hole forms at some time during the simulation. The goal is to compute the black hole's position via (2.1.8) so that its mass can be determined from (2.1.9) (see Section 2.3). Because the characteristic coordinates (τ, ρ) do not allow us to continue the simulation past the black hole formation event, we need a way to keep the simulation from terminating when Θ^+ approaches 0 (see Figure 2, right).

To avoid the need to adaptively modify the decomposition of the time domain, we carry out the supercritical case study using initial data parameter values near $(\phi_0, \rho_0, \delta_0) = (0.01, 75, 1)$, which we have also used for the results in Figure 5, right. With these parameters and in particular for $\phi_0 \geq 0.01$, for all investigated partitions of the time axis with $N_{\text{pr}} \leq 256$, the black hole generated by the fine time integrator forms in the *last* TS unless ϕ_0 becomes too large (ρ_0 and δ_0 are fixed). Thus, Parareal can be used over all TSs except for the last one, where only the fine method is executed to compute the black hole's position. The C++ implementation uses a try-throw-catch approach to prevent complete termination of the simulation; if the radicand in the definition of Θ^+ in (2.1.8) fails to be nonnegative, an exception is thrown such that the Parareal iteration can continue. As the Parareal iteration converges and better and better starting values are provided for $\mathcal{F}$ on the last TS, the accuracy of the computed black hole position improves. A more general implementation aiming at production runs would need to allow for black hole formation in TSs before the last one, but this is left for future work. In this article, the focus lies on investigating the *principal applicability* of Parareal to the simulation of gravitational collapse.

	$\phi_0^\star$		γ	
	Value	Error (%)	Value	Error (%)
Coarse	0.01057748	$7.25 \cdot 10^{-1}$	0.458	20.21
$N_{\text{it}} = 1$	0.01055915	$5.51 \cdot 10^{-1}$	0.377	1.05
$N_{\text{it}} = 2$	0.01050240	$1.01 \cdot 10^{-2}$	0.370	2.89
$N_{\text{it}} = 3$	0.01050135	$9.52 \cdot 10^{-5}$	0.381	0
Fine	0.01050134	0	0.381	0

Table 1. Approximate values and relative errors for the critical amplitude $\phi_0^\star$ and resulting straight line slope γ.

Figure 8, left, depicts the Choptuik scaling that results from solutions computed with Parareal for $N_{\text{pr}} = 256$ after the first three iterations. Table 1 lists the generated values of $\phi_0^\star$ and γ (see Section 2.3) and errors compared to the value provided by the fine integrator, which agrees with the result in [20]. As can be seen in Figure 8, left, the coarse integrator $\mathcal{C}$ alone cannot adequately resolve black holes with $\phi_0 - \phi_0^\star \lesssim 10^{-9}$ (they are too small for $\mathcal{C}$ to be "visible") and its γ is wrong by about 20%. This means that the coarse method is too "coarse" in the sense that, on its own, it cannot correctly capture the physics underlying the investigated problem. Nonetheless, *Parareal* is not only capable of generating the correct black hole physics but can do so after only one iteration.

Figure 8, right, visualizes the speedup achieved in the supercritical case including the theoretical estimate according to (3.4.1). The numbers of iterations required for Parareal to converge are derived from an analysis just like the one plotted in Figure 6, middle, for the subcritical case, and basically the values are identical. Up to 64 processes, good speedup close to the theoretical bound is observed. For larger core numbers, however, speedup reaches a plateau and performance is no longer increasing. As in the subcritical case, as N_{pr} increases, the computing times per TS eventually become too small and Parareal's runtime becomes dominated by, e.g., communication (see Figure 7). Even though the temporal parallelization eventually saturates, substantial acceleration of almost a factor of 30 using 128 cores in time is possible, corresponding to a parallel efficiency of about 23%.

5. Conclusion

The article assesses the performance of the parallel-in-time integration method Parareal for the numerical simulation of gravitational collapse of a massless scalar field in spherical symmetry. It gives an overview of the dynamics and physics described by the corresponding Einstein field equations and presents the employed numerical methods to solve them. Because the system is formulated and solved in characteristic coordinates, the computational spacetime domain is triangular so that

later time steps carry fewer spatial degrees of freedom. A strategy for balancing computational *cost* per subinterval instead of just number of steps is discussed, and its benefits are demonstrated by traces using the Vampir tool. Numerical experiments are presented for both the sub- and supercritical case. Parareal converges rapidly for both and, for the latter, correctly reproduces Choptuik's mass scaling law after only one iteration despite the fact that the used coarse integrator alone generates a strongly flawed mass scaling law. This underlines the capability of Parareal to quickly correct a coarse method that does not resolve the dynamics of the problem. The results given here illustrate that Parareal and presumably other parallel-in-time methods as well can be used to improve utilization of parallel computers for numerical studies of black hole formation.

Multiple directions for future research emerge from the presented results. Evaluating performance gains for computing the critical solution [7; 21] would be valuable. Next, more complex collapse scenarios such as in the Einstein–Yang–Mills system [8], axial symmetry [37], or binary black hole spacetimes [38] could be addressed. An extended implementation of Parareal could utilize a more sophisticated convergence criterion [2], more flexible black hole detection, and parallelism in space via, e.g., again Parareal. The latter would be possible because the integration along the characteristic we took to represent space is for the solution of initial value problems just like in the temporal direction. Another topic of interest is that of adaptive mesh refinement (J. Thornburg, personal communication, 2015): how it can be used efficiently in connection with Parareal or other time-parallel methods seems to be an open problem. As discussed in the introduction, a mathematical analysis of the convergence behavior of Parareal for Einstein's equations would be of great interest as well, particularly since the good performance is unexpected in view of the negative theoretical results for basic hyperbolic problems. Finally, incorporating a parallel-in-time integration method into a software library widely used for black hole or other numerical relativity simulations would be the ideal way to make this new approach available to a large group of domain scientists.[11]

Acknowledgments

We would like to thank Matthew Choptuik from the University of British Columbia in Vancouver, Canada and Jonathan Thornburg from the Indiana University in Bloomington for providing feedback and suggestions on an earlier version of the manuscript. We would also like to thank Jean-Guillaume Piccinali and Gilles Fourestey from the Swiss National Supercomputing Center (CSCS) in Lugano and

[11] A copy of the library Lib4PrM for the Parareal method can be obtained by cloning the Git repository https://scm.ti-edu.ch/repogit/lib4prm.

Andrea Arteaga from the Swiss Federal Institute of Technology Zurich (ETHZ) for discussions concerning the hardware at CSCS.

This research is funded by the Deutsche Forschungsgemeinschaft (DFG) as part of the "ExaSolvers" project in the Priority Programme 1648 "Software for Exascale Computing" (SPPEXA) and by the Swiss National Science Foundation (SNSF) under the lead agency agreement as grant SNSF-145271. The research of Kreienbuehl, Ruprecht, and Krause is also funded through the "Future Swiss Electrical Infrastructure" (FURIES) project of the Swiss Competence Centers for Energy Research (SCCER) at the Commission for Technology and Innovation (CTI).

References

[1] M. Alcubierre, *Introduction to $3+1$ numerical relativity*, International Series of Monographs on Physics, no. 140, Oxford University, 2008.

[2] E. Aubanel, *Scheduling of tasks in the parareal algorithm*, Parallel Comput. **37** (2011), no. 3, 172–182.

[3] T. W. Baumgarte and S. L. Shapiro, *Numerical relativity: solving Einstein's equations on the computer*, Cambridge University, 2010.

[4] M. J. Berger and J. Oliger, *Adaptive mesh refinement for hyperbolic partial differential equations*, J. Comput. Phys. **53** (1984), no. 3, 484–512.

[5] J.-P. Berrut and L. N. Trefethen, *Barycentric Lagrange interpolation*, SIAM Rev. **46** (2004), no. 3, 501–517.

[6] F. Chen, J. S. Hesthaven, and X. Zhu, *On the use of reduced basis methods to accelerate and stabilize the parareal method*, Reduced order methods for modeling and computational reduction (A. Quarteroni and G. Rozza, eds.), Modeling, Simulation and Applications, no. 9, Springer, Cham, 2014, pp. 187–214.

[7] M. W. Choptuik, *Universality and scaling in gravitational collapse of a massless scalar field*, Phys. Rev. Lett. **70** (1993), no. 1, 9–12.

[8] M. W. Choptuik, E. W. Hirschmann, and R. L. Marsa, *New critical behavior in Einstein–Yang–Mills collapse*, Phys. Rev. D **60** (1999), no. 12, 124011.

[9] A. J. Christlieb, C. B. Macdonald, and B. W. Ong, *Parallel high-order integrators*, SIAM J. Sci. Comput. **32** (2010), no. 2, 818–835.

[10] D. Christodoulou, *Bounded variation solutions of the spherically symmetric Einstein-scalar field equations*, Comm. Pure Appl. Math. **46** (1993), no. 8, 1131–1220.

[11] X. Dai and Y. Maday, *Stable parareal in time method for first- and second-order hyperbolic systems*, SIAM J. Sci. Comput. **35** (2013), no. 1, A52–A78.

[12] M. Emmett and M. L. Minion, *Toward an efficient parallel in time method for partial differential equations*, Commun. Appl. Math. Comput. Sci. **7** (2012), no. 1, 105–132.

[13] R. D. Falgout, S. Friedhoff, T. V. Kolev, S. P. MacLachlan, and J. B. Schroder, *Parallel time integration with multigrid*, SIAM J. Sci. Comput. **36** (2014), no. 6, C635–C661.

[14] P. F. Fischer, F. Hecht, and Y. Maday, *A parareal in time semi-implicit approximation of the Navier–Stokes equations*, Domain decomposition methods in science and engineering (Berlin, 2003) (R. Kornhuber, R. Hoppe, J. Périaux, O. Pironneau, O. Widlund, and J. Xu, eds.), Lecture Notes in Computational Science and Engineering, no. 40, Springer, Berlin, 2005, pp. 433–440.

[15] M. S. Floater and K. Hormann, *Barycentric rational interpolation with no poles and high rates of approximation*, Numer. Math. **107** (2007), no. 2, 315–331.

[16] M. J. Gander, *Analysis of the parareal algorithm applied to hyperbolic problems using characteristics*, Bol. Soc. Esp. Mat. Apl. (2008), no. 42, 21–35.

[17] ———, *50 years of time parallel time integration*, Multiple shooting and time domain decomposition methods (Heidelberg, 2013) (T. Carraro, M. Geiger, S. Körkel, and R. Rannacher, eds.), Contributions in Mathematical and Computational Sciences, no. 9, Springer, Cham, 2015, pp. 69–113.

[18] M. J. Gander and M. Petcu, *Analysis of a Krylov subspace enhanced parareal algorithm for linear problems*, ESAIM Proc. **25** (2008), 114–129.

[19] M. J. Gander and S. Vandewalle, *Analysis of the parareal time-parallel time-integration method*, SIAM J. Sci. Comput. **29** (2007), no. 2, 556–578.

[20] D. Garfinkle, *Choptuik scaling in null coordinates*, Phys. Rev. D **51** (1995), no. 10, 5558–5561.

[21] C. Gundlach and J. M. Martín-García, *Critical phenomena in gravitational collapse*, Living Rev. Relativ. **10** (2007), no. 5.

[22] W. Hackbusch, *Parabolic multigrid methods*, Proceedings of the sixth International Symposium on Computing Methods in Applied Sciences and Engineering (Versailles, 1983) (R. Glowinski and J.-L. Lions, eds.), North-Holland, Amsterdam, 1984, pp. 189–197.

[23] G. Horton, *The time-parallel multigrid method*, Comm. Appl. Numer. Methods **8** (1992), no. 9, 585–595.

[24] G. Horton, S. Vandewalle, and P. Worley, *An algorithm with polylog parallel complexity for solving parabolic partial differential equations*, SIAM J. Sci. Comput. **16** (1995), no. 3, 531–541.

[25] V. Husain, *Critical behaviour in quantum gravitational collapse*, Adv. Sci. Lett. **2** (2009), no. 2, 214–220.

[26] L. E. Kidder, M. A. Scheel, S. A. Teukolsky, E. D. Carlson, and G. B. Cook, *Black hole evolution by spectral methods*, Phys. Rev. D **62** (2000), no. 8, 084032.

[27] E. Komatsu, J. Dunkley, M. R. Nolta, C. L. Bennett, B. Gold, G. Hinshaw, N. Jarosik, D. Larson, M. Limon, L. Page, D. N. Spergel, M. Halpern, R. S. Hill, A. Kogut, S. S. Meyer, G. S. Tucker, J. L. Weiland, E. Wollack, and E. L. Wright, *Five-year Wilkinson microwave anisotropy probe observations: cosmological interpretation*, Astrophys. J. Suppl. S. **180** (2009), no. 2, 330–376.

[28] A. Kreienbuehl, *Quantum cosmology, polymer matter, and modified collapse*, Ph.D. thesis, University of New Brunswick, 2011.

[29] A. Kreienbuehl, V. Husain, and S. S. Seahra, *Modified general relativity as a model for quantum gravitational collapse*, Classical Quant. Grav. **29** (2012), no. 9, 095008.

[30] A. Kreienbuehl, A. Naegel, D. Ruprecht, R. Speck, G. Wittum, and R. Krause, *Numerical simulation of skin transport using Parareal*, Comput. Vis. Sci. **17** (2015), no. 2, 99–108.

[31] J.-L. Lions, Y. Maday, and G. Turinici, *Résolution d'EDP par un schéma en temps "pararéel"*, C. R. Acad. Sci. Paris I **332** (2001), no. 7, 661–668.

[32] F. Löffler, J. Faber, E. Bentivegna, T. Bode, P. Diener, R. Haas, I. Hinder, B. C. Mundim, C. D. Ott, E. Schnetter, G. Allen, M. Campanelli, and P. Laguna, *The Einstein Toolkit: a community computational infrastructure for relativistic astrophysics*, Classical Quant. Grav. **29** (2012), no. 11, 115001.

[33] M. L. Minion, *A hybrid parareal spectral deferred corrections method*, Commun. Appl. Math. Comput. Sci. **5** (2010), no. 2, 265–301.

[34] A. S. Nielsen and J. S. Hesthaven, *Fault tolerance in the Parareal method*, Proceedings of the ACM Workshop on Fault-Tolerance for HPC at Extreme Scale (Kyoto, 2016), ACM, New York, 2016.

[35] J. Nievergelt, *Parallel methods for integrating ordinary differential equations*, Comm. ACM **7** (1964), no. 12, 731–733.

[36] E. Poisson, *A relativist's toolkit: the mathematics of black-hole mechanics*, Cambridge University, 2004.

[37] F. Pretorius, *Numerical simulations of gravitational collapse*, Ph.D. thesis, University of British Columbia, 2002.

[38] ———, *Evolution of binary black-hole spacetimes*, Phys. Rev. Lett. **95** (2005), no. 12, 121101.

[39] F. Pretorius and L. Lehner, *Adaptive mesh refinement for characteristic codes*, J. Comp. Phys. **198** (2004), no. 1, 10–34.

[40] D. Ruprecht and R. Krause, *Explicit parallel-in-time integration of a linear acoustic-advection system*, Comput. Fluids **59** (2012), 72–83.

[41] R. Speck and D. Ruprecht, *Toward fault-tolerant parallel-in-time integration with PFASST*, Parallel Comput. **62** (2017), 20–37.

[42] R. Speck, D. Ruprecht, R. Krause, M. Emmett, M. Minion, M. Winkel, and P. Gibbon, *A massively space-time parallel N-body solver*, SC '12: proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis (Salt Lake City, 2012), IEEE, Los Alamitos, CA, 2012.

[43] J. Thornburg, *Adaptive mesh refinement for characteristic grids*, Gen. Relativity Gravitation **43** (2011), no. 5, 1211–1251.

[44] J. Ziprick and G. Kunstatter, *Dynamical singularity resolution in spherically symmetric black hole formation*, Phys. Rev. D **80** (2009), no. 2, 024032.

Received April 24, 2016. Revised December 28, 2016.

ANDREAS KREIENBUEHL: akreienbuehl@lbl.gov
Center for Computational Sciences and Engineering, Lawrence Berkeley National Laboratory, 1 Cyclotron Road, Berkeley, CA 94720, United States

PIETRO BENEDUSI: pietro.benedusi@usi.ch
Institute of Computational Science, Faculty of Informatics, Università della Svizzera italiana, Via Giuseppe Buffi 13, CH-6904 Lugano, Switzerland

DANIEL RUPRECHT: d.ruprecht@leeds.ac.uk
School of Mechanical Engineering, University of Leeds, Woodhouse Lane, Leeds, LS2 9JT, United Kingdom

ROLF KRAUSE: rolf.krause@usi.ch
Institute of Computational Science, Faculty of Informatics, Università della Svizzera italiana, Via Giuseppe Buffi 13, CH-6904 Lugano, Switzerland