Timing and efficiency of PFASST

Matthew Emmett (memmett)

Center for Computational Sciences and Engineering
Lawrence Berkeley National Laboratory

May 28 2014

Timing and efficiency of PFASST

and time-stepping on extreme scales

- Supercomputers and time-stepping
  - Memory bandwidth, FLOP density, loop structure, weak scaling
- Parallel time-stepping
  - Across the method vs across the domain
  - Spectral Deferred Corrections (SDC) and Multilevel SDC (MLSDC)
  - Parallel Full Approximation Scheme in Space and Time (PFASST)
- PFASST
  - Applications
  - Timings
- MLSDC + Adaptive Mesh Refinement (AMR)

Collaborators

Lots of us are Matts; most of us are German; only one Canuck

PFASST and applications

- Michael Minion (Stanford/LBNL - USA)
- Robert Speck (JSC - Germany)
- Daniel Ruprecht (ICS - Switzerland)
- Rolf Krause (ICS - Switzerland)
- Mathias Winkel (JSC - Germany)
- Matthias Bolten (Wuppertal - Germany)

SDC+AMR

- John Bell (CCSE@LBNL)
- Weiqun Zhang (CCSE@LBNL)
- Michael Minion (Stanford/CCSE@LBNL)

This work was supported by the US Department of Energy.

What's the end goal?

When choosing a time-stepping algorithm, what are we looking for?

Typically we want some combination of

- accurate
- stable
- robust
- X-preserving

And, we want it to be

- fast
- Y-efficient

Time-stepping on supercomputers

Credit to J. Brown for these ideas

General trend in super-computing seems to be

$$\text{Balanced arithmetic intensity} = \frac{\text{Peak GF/s}}{\text{Bandwidth GB/s}} \quad \text{is increasing}$$

As such, is this the right paradigm for time-stepping?

```
for step in time_steps:
  for stage in stages:
    dudt(stage) = some_black_box(u(step, stage))
  u(step+1) = ...
```

The trap of weak scaling

Credit to M. Minion for these ideas

For most time dependent PDE simulations, increasing the number of spatial degrees of freedom will also increase the number of time steps required for a given problem

more grid points = more time steps

Hence, if processor speeds do not increase with core counts, increasing problem size implies longer run times *even with perfect weak spatial scaling*

more grid points = longer run time

Goal for temporal parallelization for ODEs

Credit to M. Minion for these ideas

We want to provide another direction (literally dimension) for increasing computational concurrency.

- For any problem of fixed size, parallel efficiency of computing the right-hand side will eventually saturate as more processors are used.
- For many PDE applications, it is widely predicted that the "adequate" threshold will decrease as we move closer to exascale computing.

Our goal is to construct parallel methods in the temporal direction that achieve "adequate" parallel efficiency

Goal for temporal parallelization for ODEs

What does "adequate" mean?

OpenMP strong scaling study for a high-order compressible Navier-Stokes combustion code (128^3 grid). Machine was a 61-core 1.1 GHz Intel Xeon Phi coprocessor that supported up to 244 hyperthreads.

Threads	Wall time per step (s)	Speedup	Efficiency
1	1223.58		
2	626.13	2.0	100.0
4	328.97	3.7	92.5
8	159.88	7.7	96.3
16	79.73	16	100.0
32	40.12	31	96.9
60	24.55	50	83.3
128	18.19	67	52.3
240	14.28	86	35.8

PFASST

Parallel full approximation scheme in space and time

PFASST is

- An iterative "across-the-domain" time-parallel integrator.
- Built upon the Spectral Deferred Correction (SDC) time-stepping method.
- A multi-level scheme with a hierarchy of space/time discretisations.

Parallel time-stepping across the method

For time-stepping algorithms in which each stage is independent

- RIDC (A. Christlieb, B. Ong *et al.*)
- Extrapolation (see e.g. D. Ketcheson and U. bin Waheed, 2013)
- DIRK
- Picard iterations

These techniques have very good scaling to a handfull of time processors.

Parallel time-stepping across the domain

Here we operate on many time-steps concurrently, in contrast to operating on a single time-step in parallel.

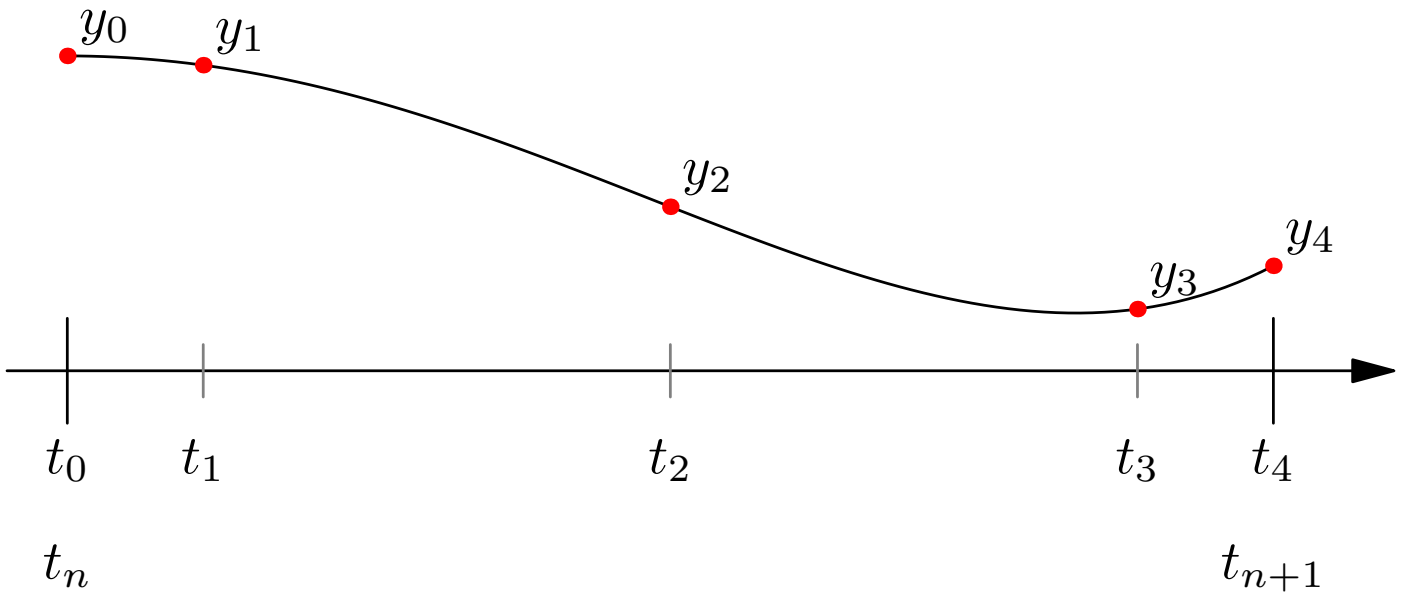
- Parareal (see e.g. J. Lions, Y. Maday, and G. Turnici)
- PITA (see e.g. J. Cortial and C. Farhat)
- PFASST (see M. Emmett and M.L. Minion)
- MGRIT

Can use *lots* of time processors. Are they efficient? Do they scale?

Collocation methods

Consider $y' = f(t, y)$ or equivalently $y = y_0 + \int_{t_0}^{t} f(\tau, y)\, d\tau.$

Define collocation points t_m between time t_n and t_{n+1}



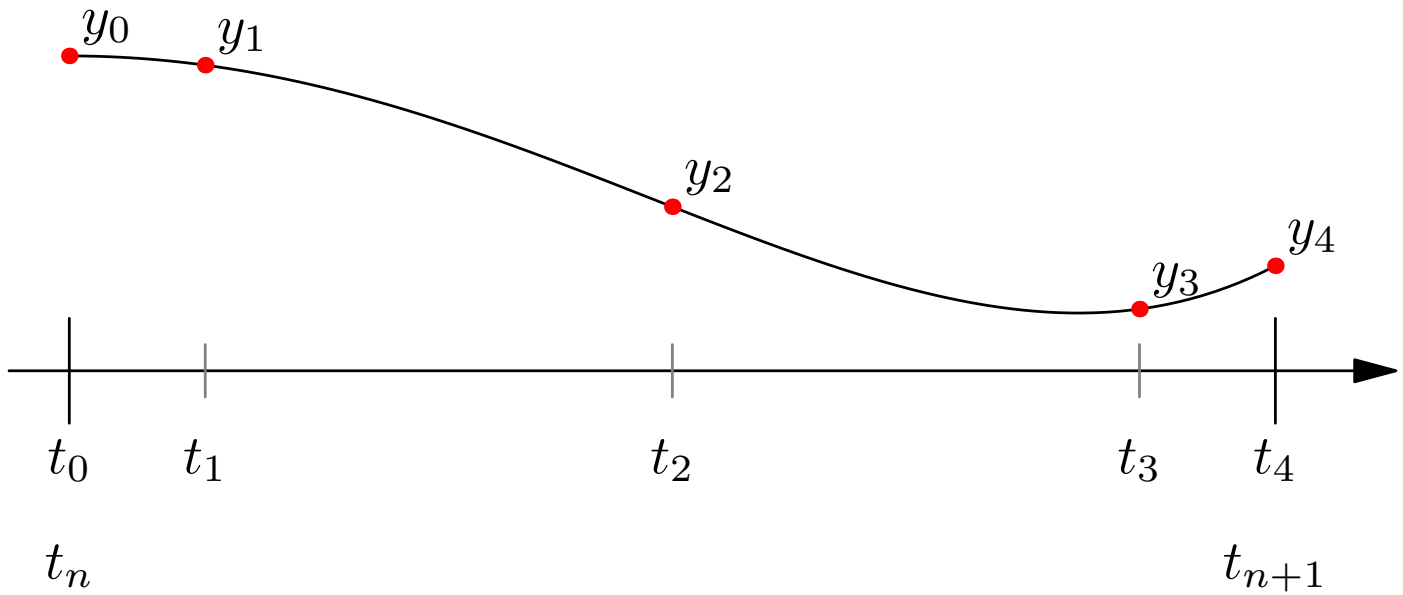
Collocation method: find polynomial $p(t)$ such that

$$p'(t_m) = f(t_m, p(t_m))$$

Collocation methods

Consider $y' = f(t, y)$ or equivalently $y = y_0 + \int_{t_0}^{t} f(\tau, y)\, d\tau.$

Define collocation points t_m between time t_n and t_{n+1}



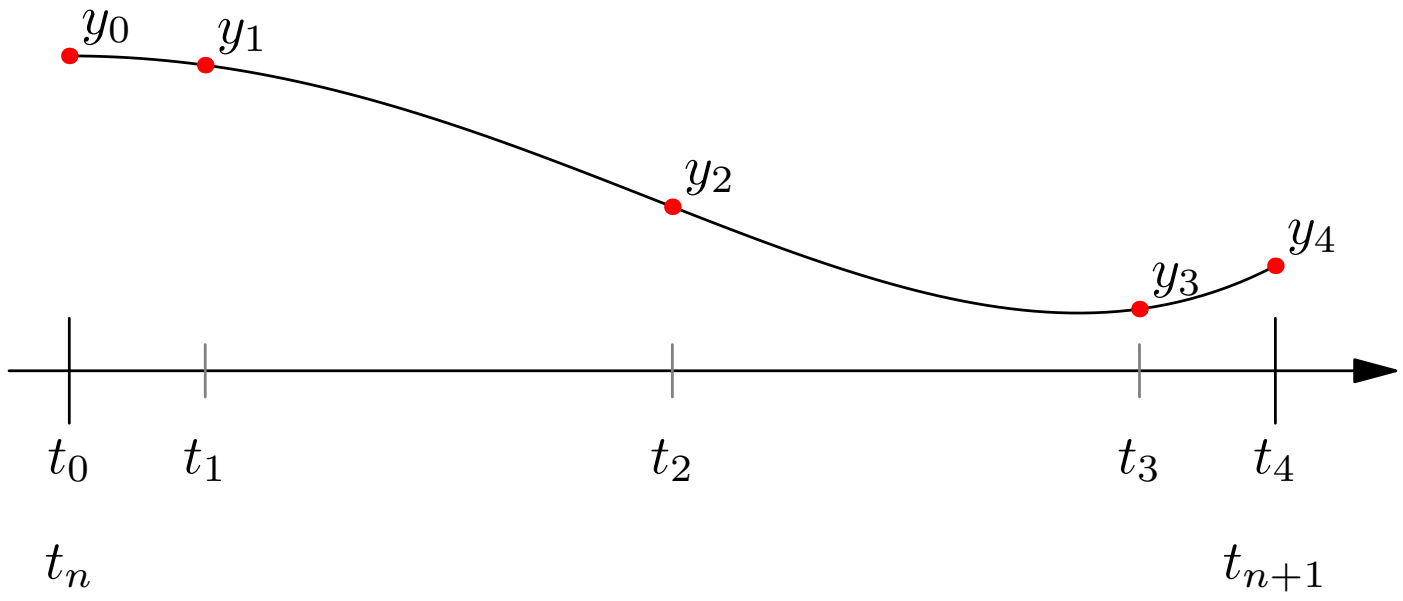
Equivalently: find $y_m \approx y(t_m)$ such that

$$y_m \approx y_0 + \Delta t \sum_{j=0}^{M} q_{mj} f(t_j, y_j)$$

Collocation methods

Consider $y' = f(t, y)$ or equivalently $y = y_0 + \int_{t_0}^{t} f(\tau, y)\, d\tau.$

Define collocation points t_m between time t_n and t_{n+1}



More compactly

$$\mathbf{Y} = \mathbf{Y}_0 + \Delta t\, \mathbf{Q}\, \mathbf{F}(\mathbf{Y})$$

Spectral Deferred Corrections (SDC)

SDC is an **iterative** time-stepping method for solving

$$y' = f(t, y) \implies \mathbf{Y} = \mathbf{Y}_0 + \Delta t\, \mathbf{Q}\, \mathbf{F}(\mathbf{Y})$$

SDC methods are built from low-order time-stepping schemes. Once the discretization is chosen, a simple recipe yields an **update** equation.

The update equation for the forward-Euler based SDC method is

$$y_{m+1}^{k+1} = y_m^{k+1} + \Delta t_m \left[f(t_m, y_m^{k+1}) - f(t_m, y_m^k) \right] + S_m^{m+1}$$

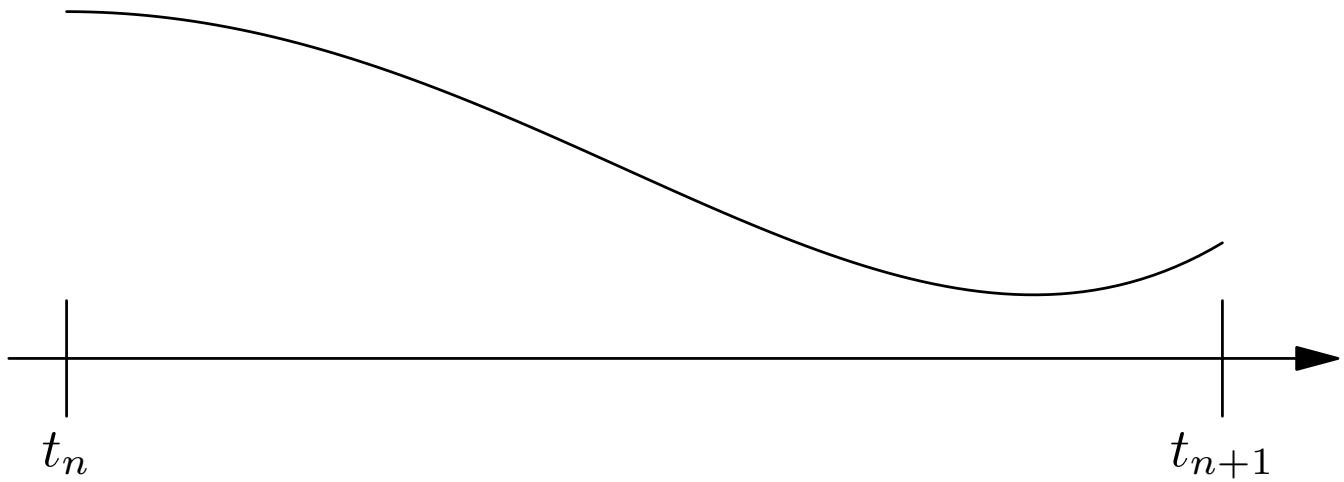
where

$$S_m^{m+1} = \Delta t \sum_{j=0}^{M} s_{m,j} f(t_j, y_j) \approx \int_{t_m}^{t_{m+1}} f(\tau, y^k(\tau))\, d\tau$$

One SDC iteration consists of performing one **sweep** of the update equation.

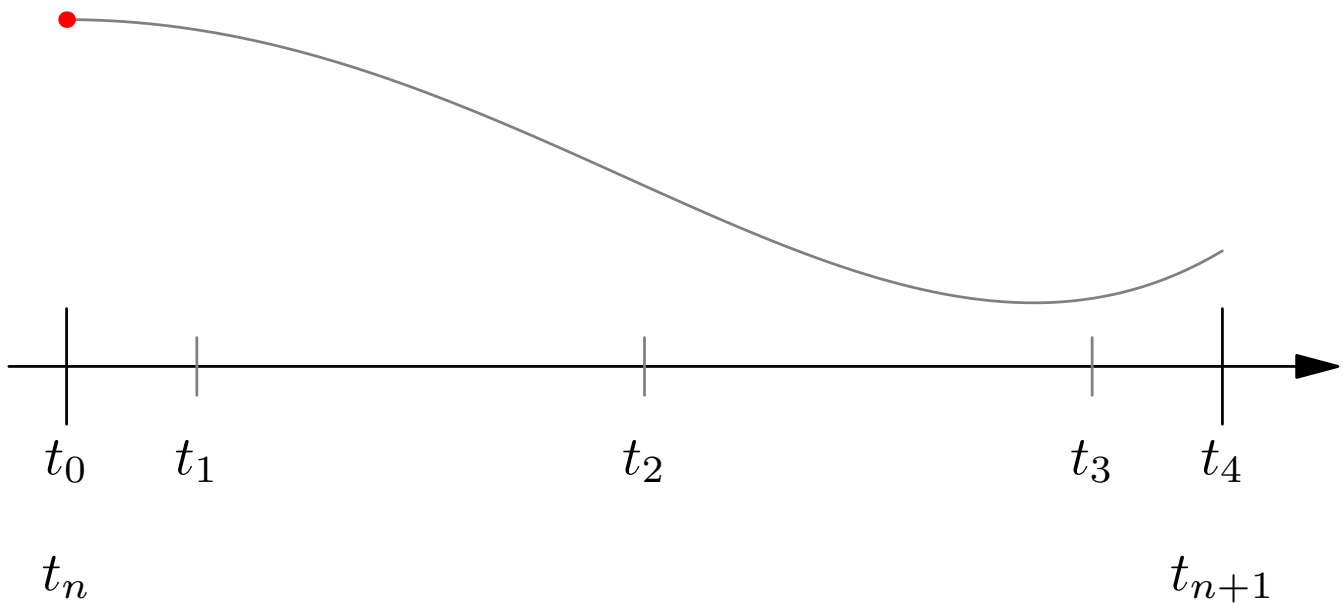
Spectral Deferred Corrections (SDC)

$$y_{m+1}^{k+1} = y_m^{k+1} + \Delta t_m \left[f(t_m, y_m^{k+1}) - f(t_m, y_m^k) \right] + S_m^{m+1}$$



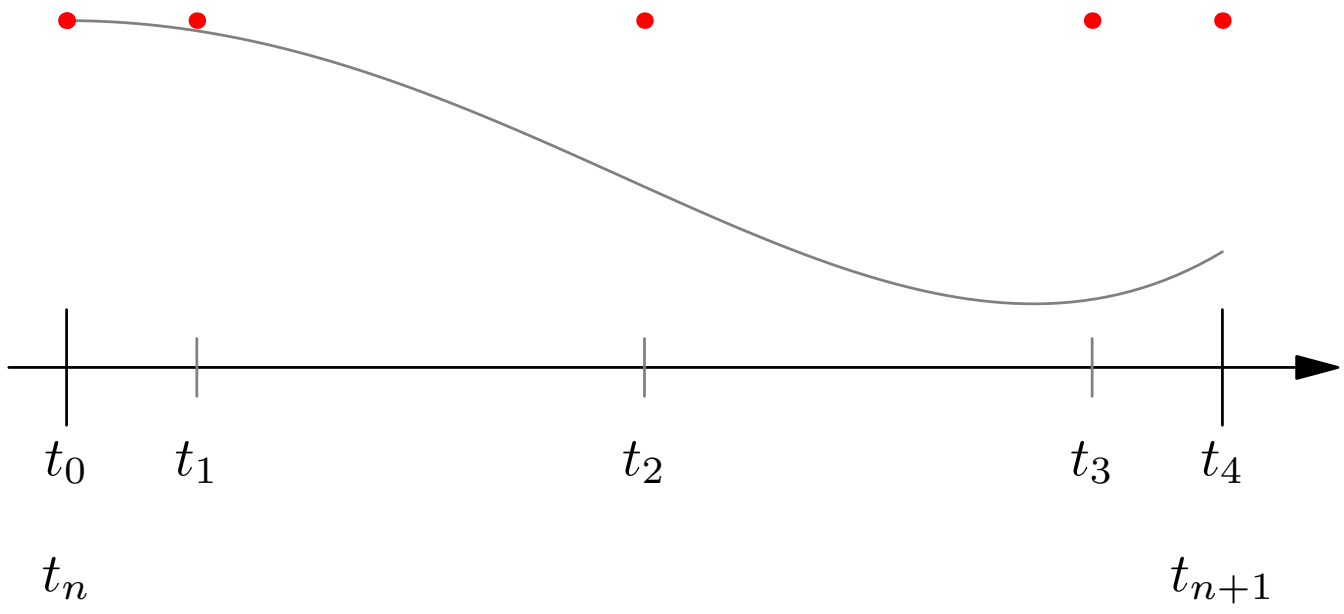
Spectral Deferred Corrections (SDC)

$$y_{m+1}^{k+1} = y_m^{k+1} + \Delta t_m \left[f(t_m, y_m^{k+1}) - f(t_m, y_m^k) \right] + S_m^{m+1}$$



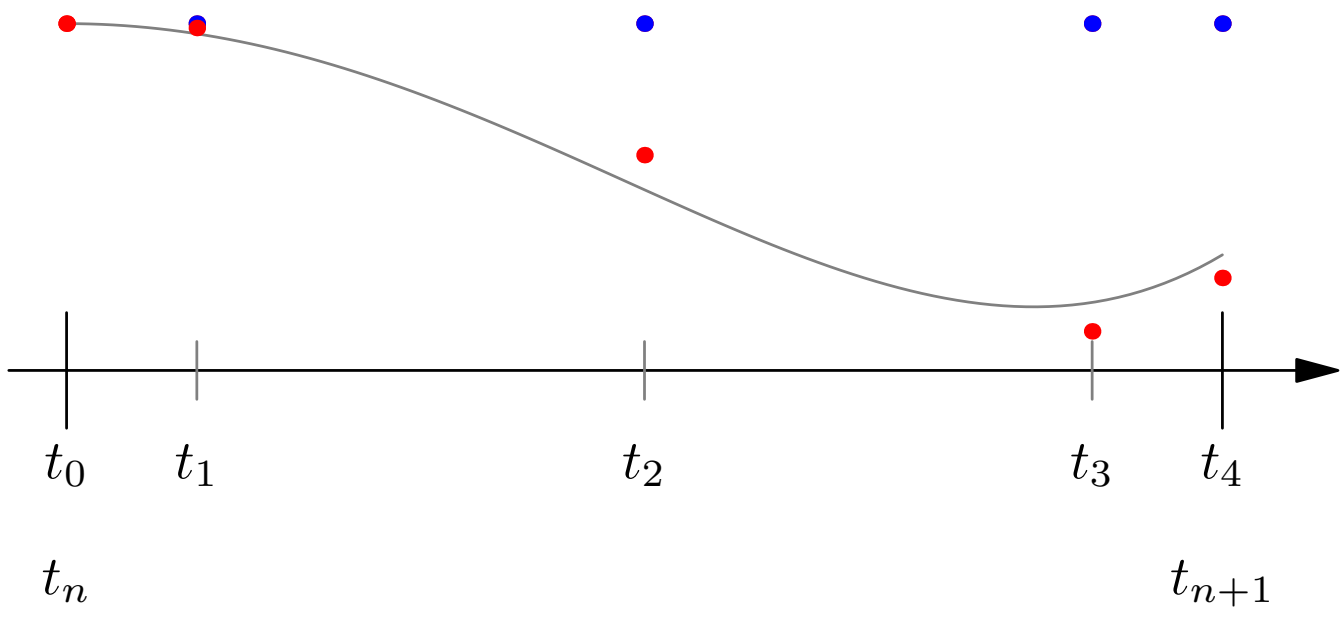
Spectral Deferred Corrections (SDC)

$$y_{m+1}^{k+1} = y_m^{k+1} + \Delta t_m \left[f(t_m, y_m^{k+1}) - f(t_m, y_m^k) \right] + S_m^{m+1}$$



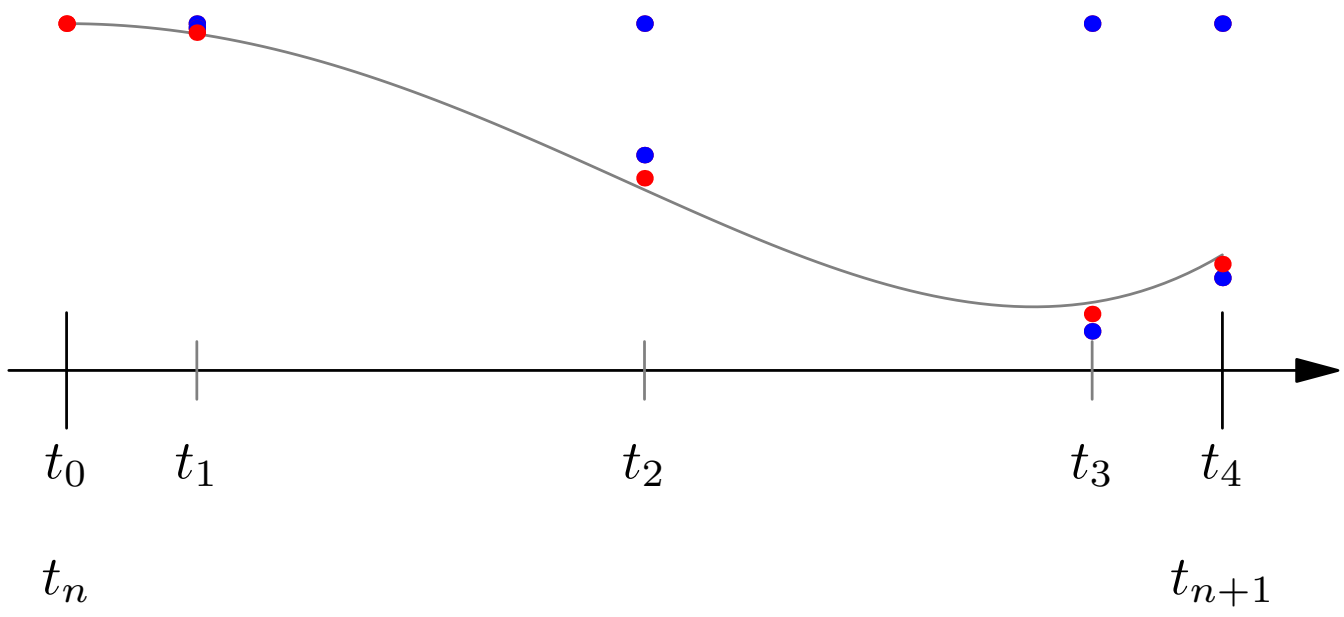
Spectral Deferred Corrections (SDC)

$$y_{m+1}^{k+1} = y_m^{k+1} + \Delta t_m \left[f(t_m, y_m^{k+1}) - f(t_m, y_m^k) \right] + S_m^{m+1}$$



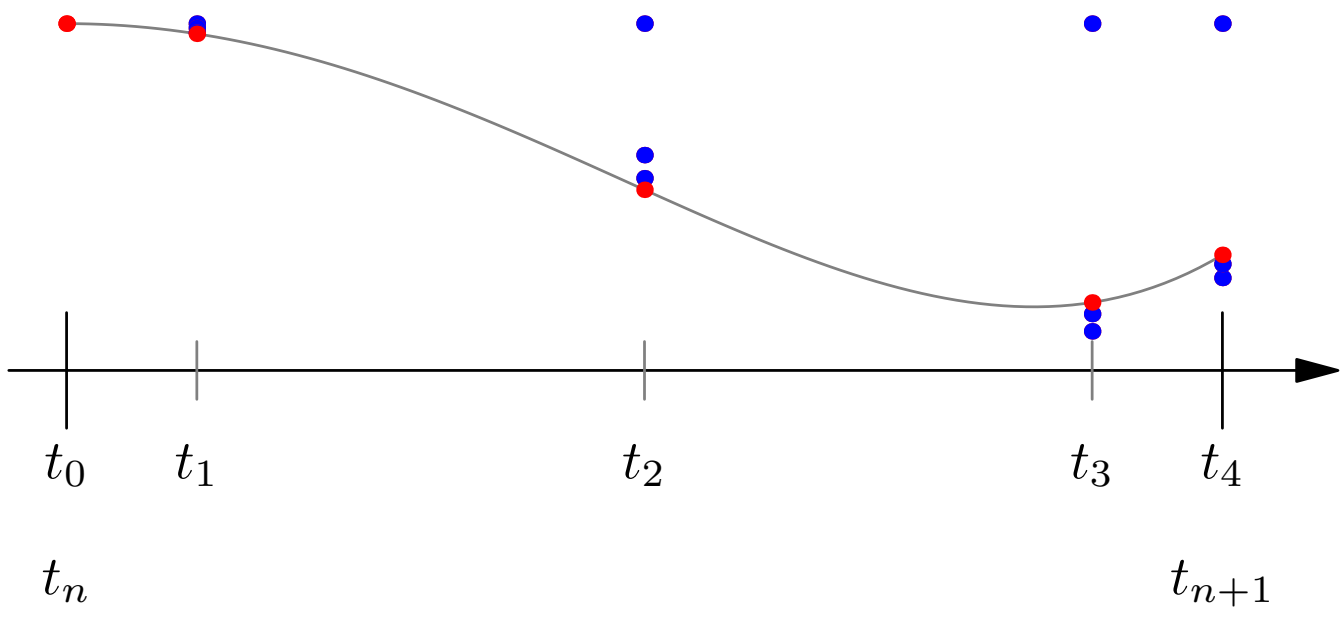
Spectral Deferred Corrections (SDC)

$$y_{m+1}^{k+1} = y_m^{k+1} + \Delta t_m \left[f(t_m, y_m^{k+1}) - f(t_m, y_m^k) \right] + S_m^{m+1}$$



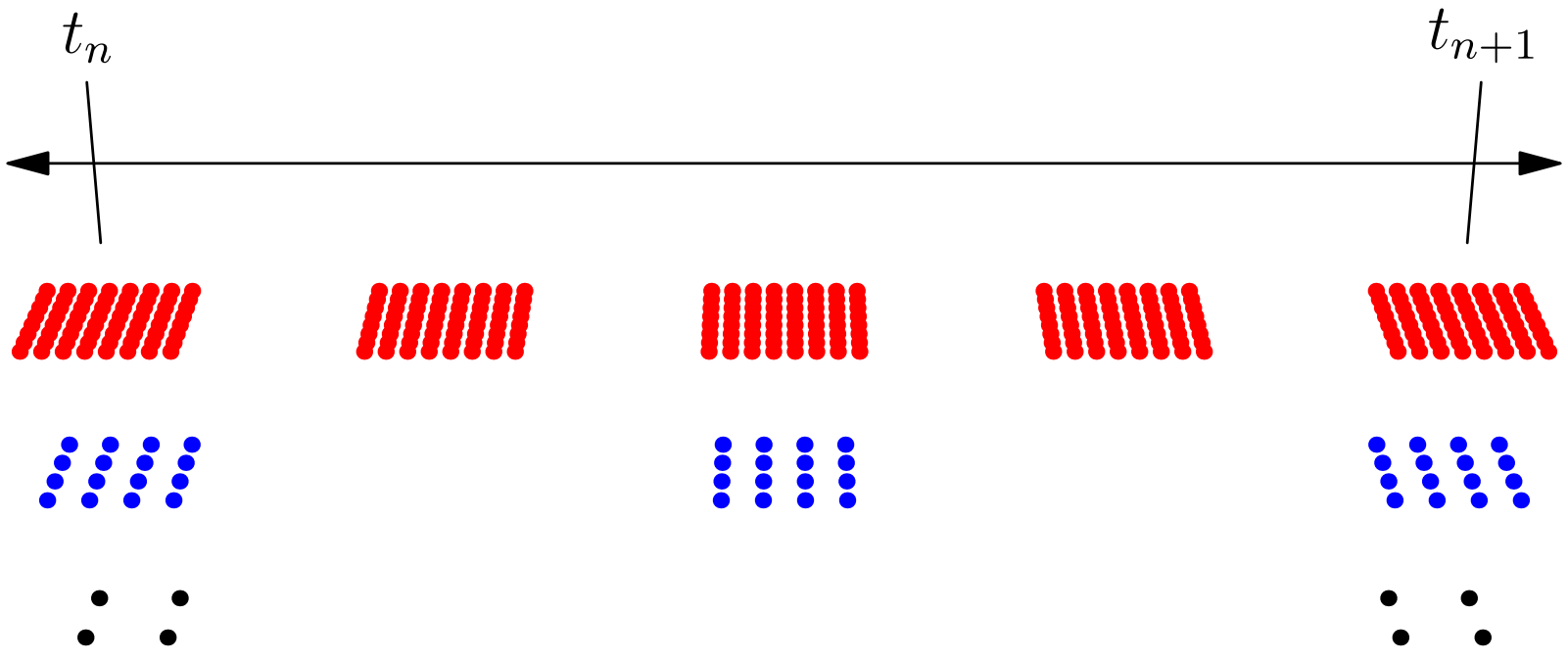
Spectral Deferred Corrections (SDC)

$$y_{m+1}^{k+1} = y_m^{k+1} + \Delta t_m \left[f(t_m, y_m^{k+1}) - f(t_m, y_m^k) \right] + S_m^{m+1}$$



Multi-level SDC (MLSDC)

For PDEs we can build hierarchies of space/time grids.



Use Full Approximation Scheme (FAS) corrections to ensure coarse level converges to fine level.

FAS correction for MLSDC

Recall compact notation for collocation solution

$$\mathbf{Y} = \mathbf{Y}_0 + \Delta t\, \mathbf{Q}\, \mathbf{F}(\mathbf{Y})$$

Quadrature operators across SDC nodes yield natural FAS correction for coarse SDC sweeps

$$\boldsymbol{\tau} = \Delta t \Big[R_{t/s} \mathbf{Q}^F \mathbf{F}^F \left(\mathbf{U}^F\right) - \mathbf{Q}^C \mathbf{F}^C \left(R\mathbf{U}^F\right) \Big].$$

Intuitively, this is

$$\tau = \left(\text{Coarse representation of } \int f^F(t, y(t))\, dt \right) - \int f^C(t, y(t))\, dt.$$

Coarsening strategies for MLSDC

To reduce the relative cost of coarse levels in MLSDC, one could

- reduce the number of nodes in time
- reduce the resolution in space
- reduce the order in space
- reduce the accuracy of implicit solves in space
- reduce the complexitiy of physics

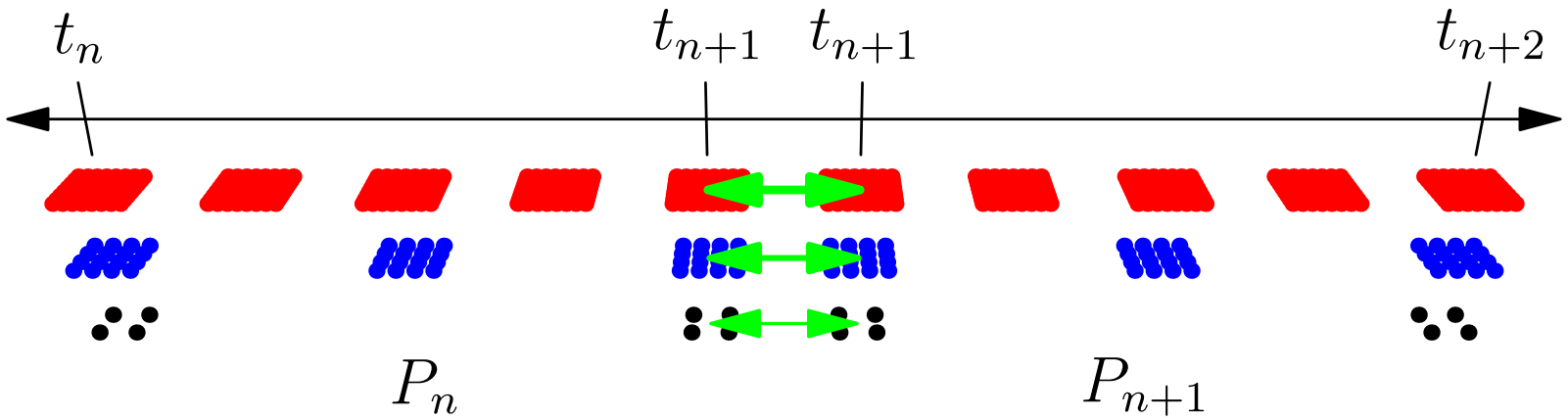
Can also speedup fine level as well

- Adaptively tigthen tolerances of implicit solves

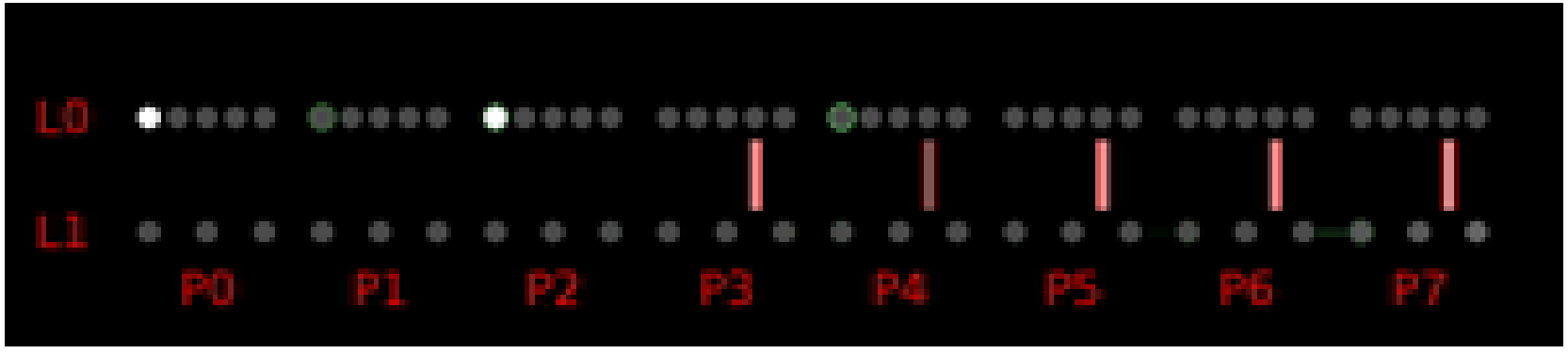
What is PFASST?

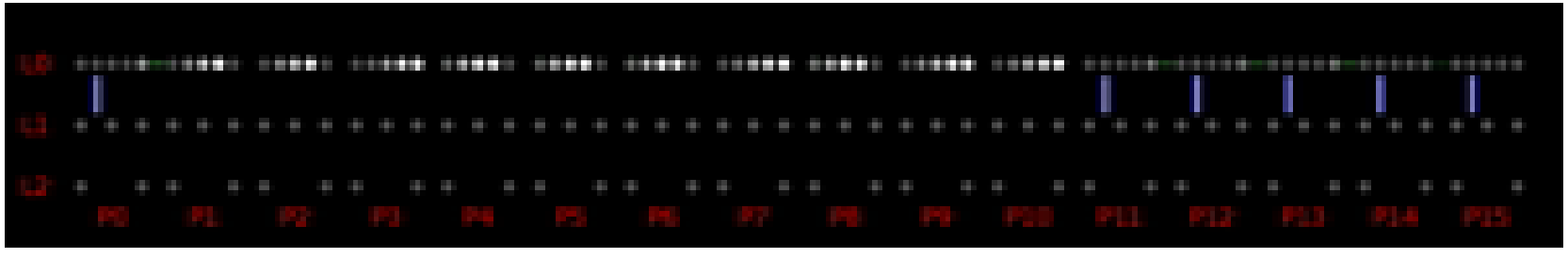
Parallel MLSDC

- PFASST is a hybrid scheme that intertwines SDC and Parareal iterations.
- PFASST is not a naive embedding of SDC within Parareal.

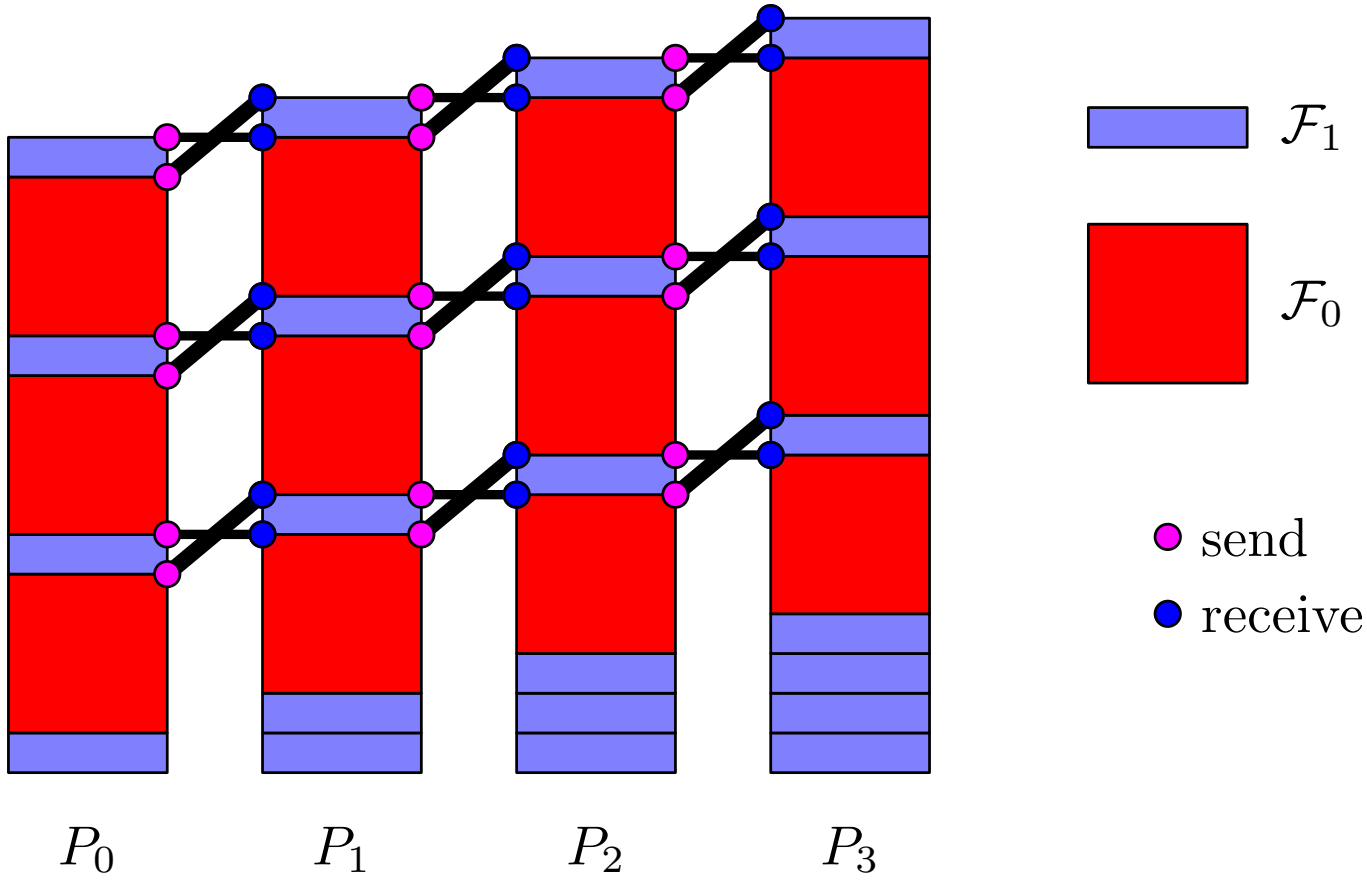


PFASST in action

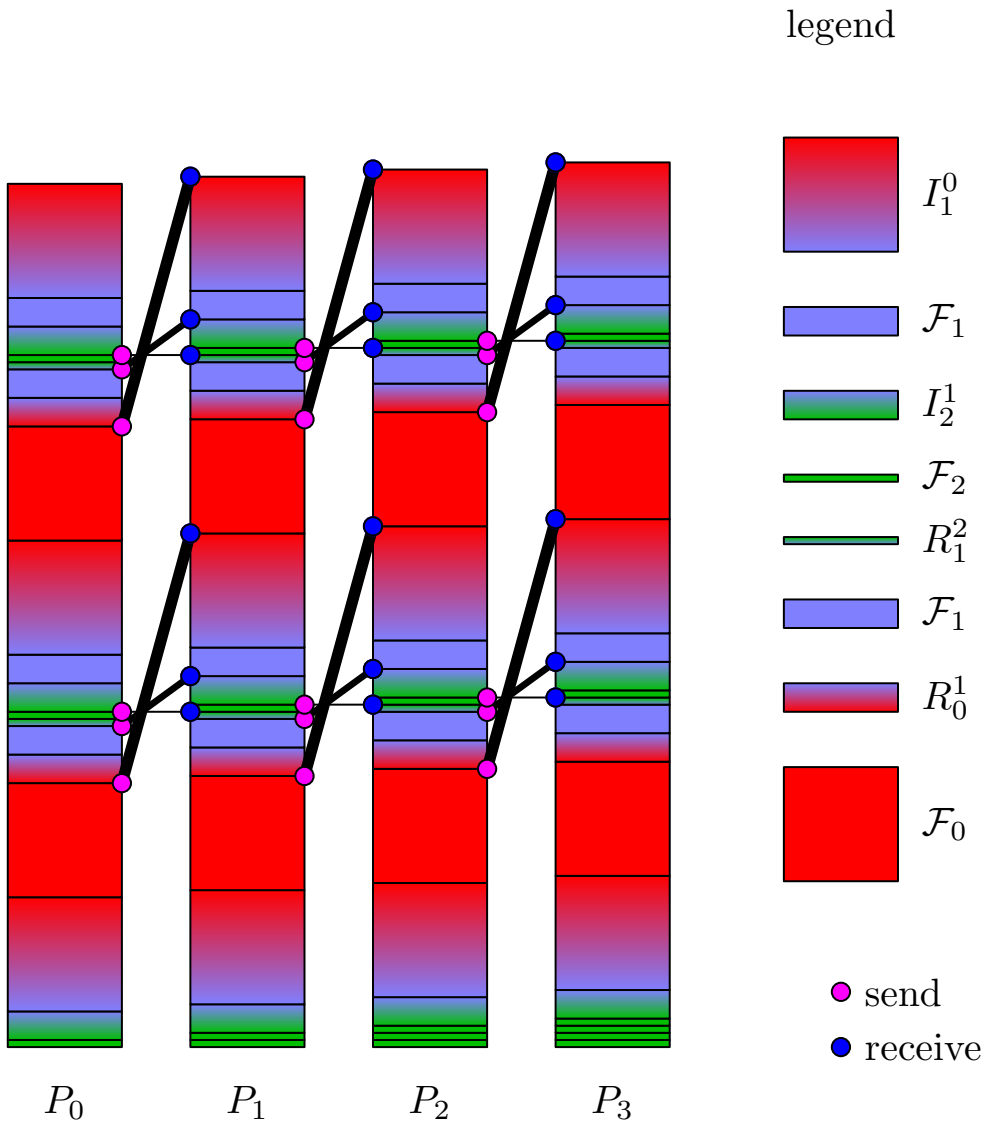




Two-level PFASST diagram



Three-level PFASST diagram



Total cost for PFASST

Three levels

- Cost

$$C = P\tau_2 + K_P(\tau_0 + \tau_1 + \tau_2 + \tau_o)$$

  where τ_ℓ is the cost of SDC sweep on level ℓ and τ_o is overhead from communication and FAS interpolation and restriction.

- Efficiency

$$E = \frac{K_S}{P\alpha_2 + K_P(1 + \alpha_1 + \alpha_2 + \alpha_o)}$$

  where $\alpha_* = \tau_*/\tau_0$

- For 3d PDEs $\alpha_2 = \tau_2/\tau_0$ is quite small (e.g. $1/256$!)
- Remember that for each K_P iteration, multiple SDC sweeps are done at each level (K_P/K_S can be less than 1)

Theoretical speedup of PFASST

More concisely, speedup is approximately

$$S \approx \frac{N}{\frac{K_p}{K_s}(1+\beta_0)}$$

with parallel efficiency

$$E \approx \frac{1}{\frac{K_p}{K_s}(1+\beta_0)}.$$

Hence if PFASST converges in fewer iterations than serial SDC, it is possible to achieve parallel efficiency of greater than 50% even with many iterations.

Applications of PFASST

Selected (notable) applications

FFTW	1D and 2D Kuramoto-Sivashinsky, 2D shear layer vorticity.
PEPC	Pretty Efficient Parallel Coulomb solver on JUGENE.
PMG	Parallel Multigrid solver for the 3D heat equation on JUQUEEN.
FFTW++	Pseudo-spectral Navier-Stokes using multi-threaded parallel FFTW++ convolutions on Edison.
OpenMM	Proof of concept, CPU/CUDA+MPI molecular dynamics.

Kuramoto-Sivashinsky

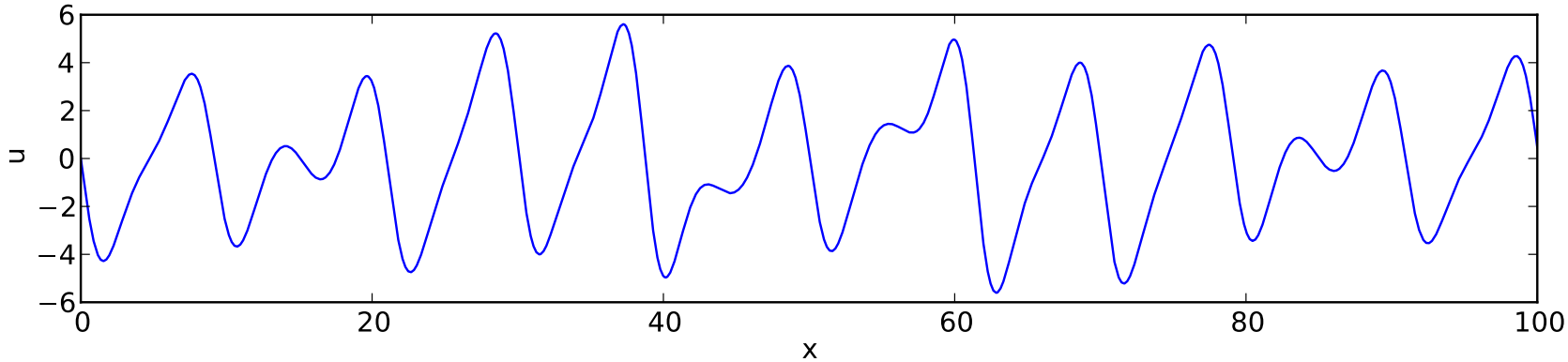
The 1d KS equation is

$$u_t + uu_x + u_{xx} + u_{xxxx} = 0$$

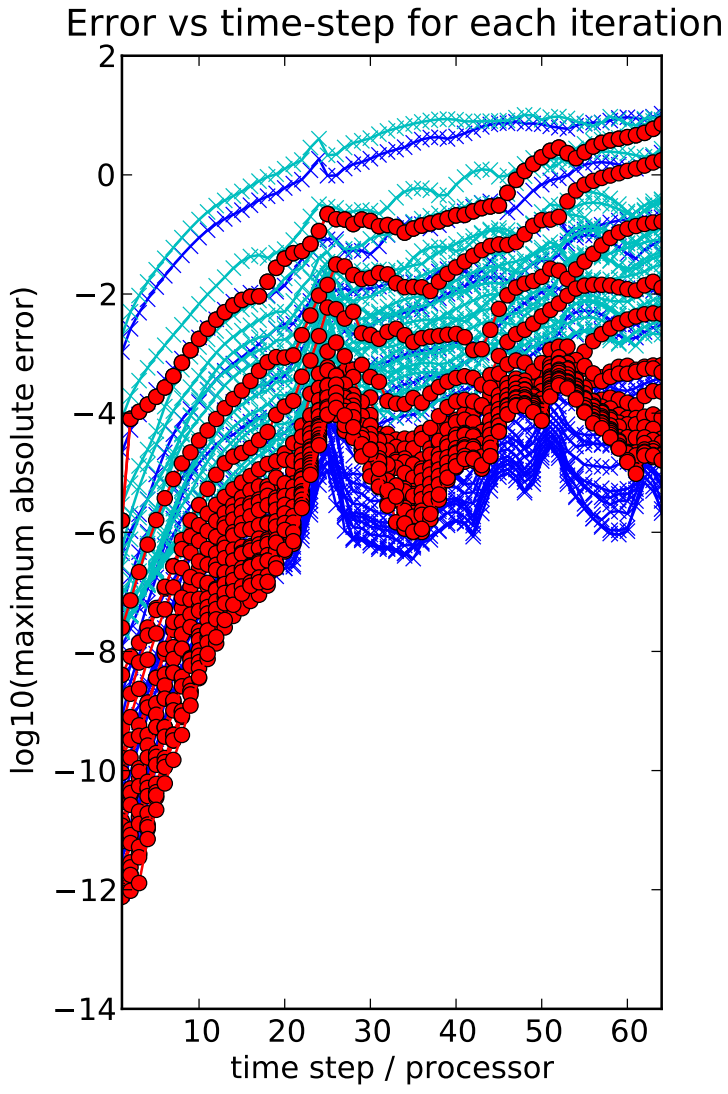
Periodic domain of length $L = 100$ with initial conditions

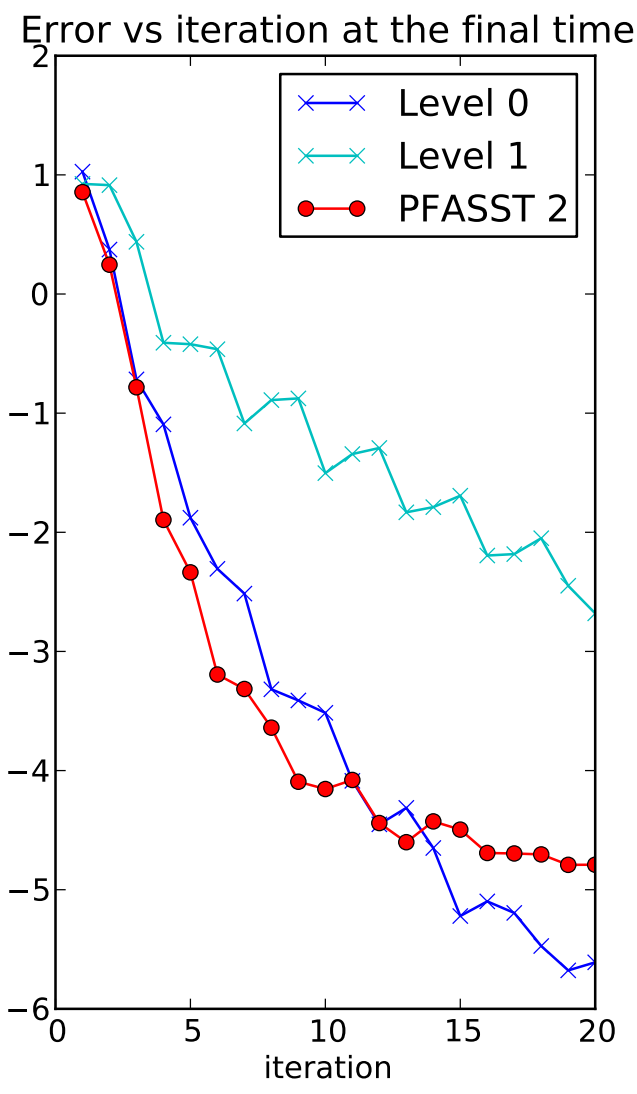
$$u_0(x) = 0.1\sin(6\pi x/L) + 0.2\sin(8\pi x/L) + 0.3\sin(14\pi x/L).$$

Solution at time $t = 64$



Kuramoto-Sivashinsky





Vorticity

Vorticity formulation of the 2D incompressible Navier-Stokes equation is

$$\partial_t \omega + \mathbf{u} \cdot \nabla \omega = \nu \nabla^2 \omega$$

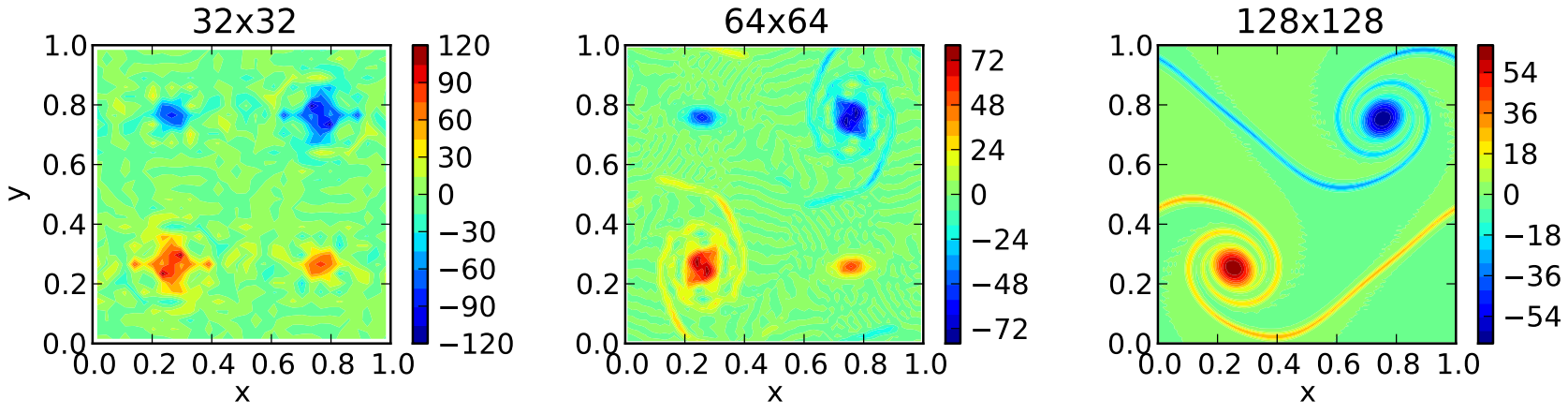
Doubly periodic domain $[0,1] \times [0,1]$ with initial conditions

$$u_0(x,y) = -1.0 + \tanh\big(\rho(0.75 - y)\big) + \tanh\big(\rho(y - 0.25)\big)$$
$$v_0(x,y) = -\delta \sin\big(2\pi(x + 0.25)\big)$$

where $\rho = 100$ and $\delta = 0.05$.

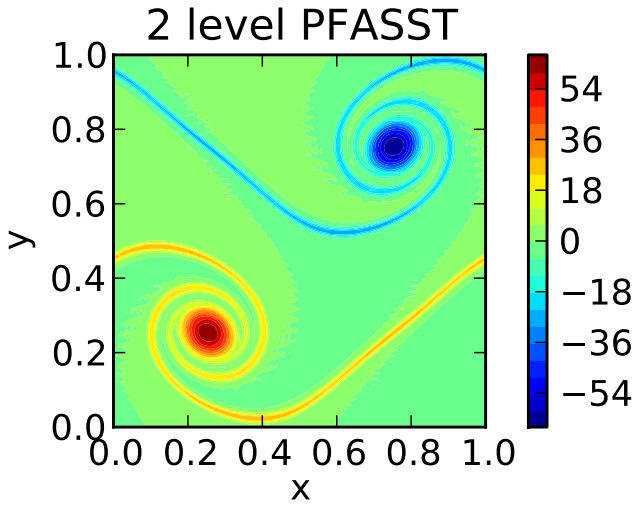
Vorticity

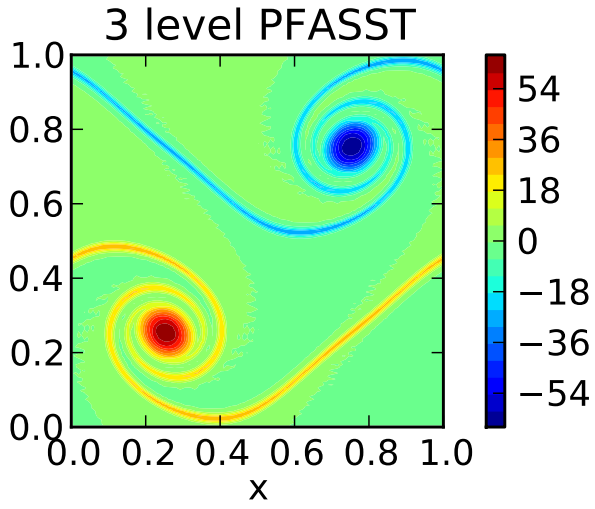
For small grids the solution is severely under-resolved.



Vorticity

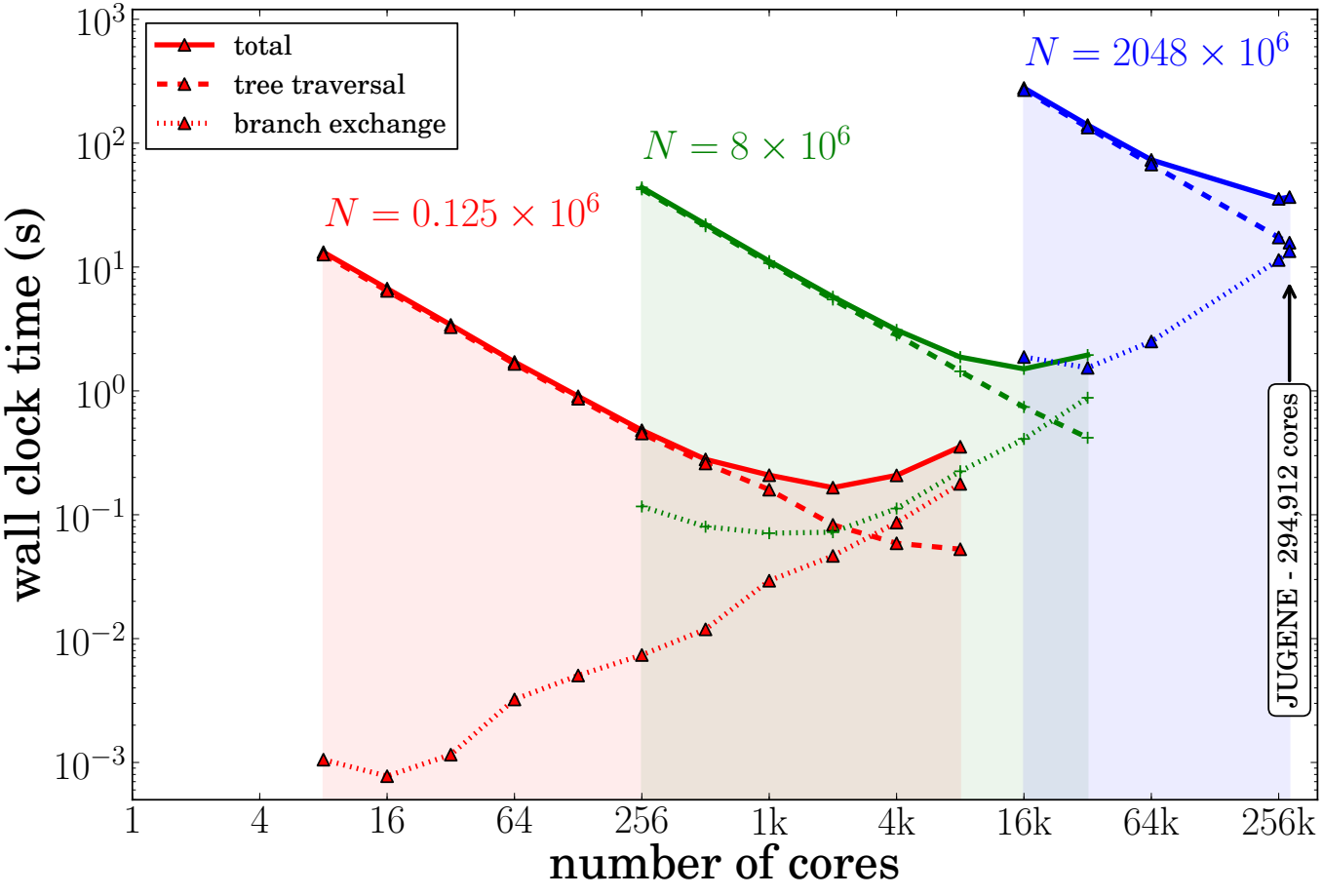
Despite the under-resolution of the coarser levels, the multi-level PFASST solutions are essentially the same as the high-resolution serial run.





PEPC+PFASST

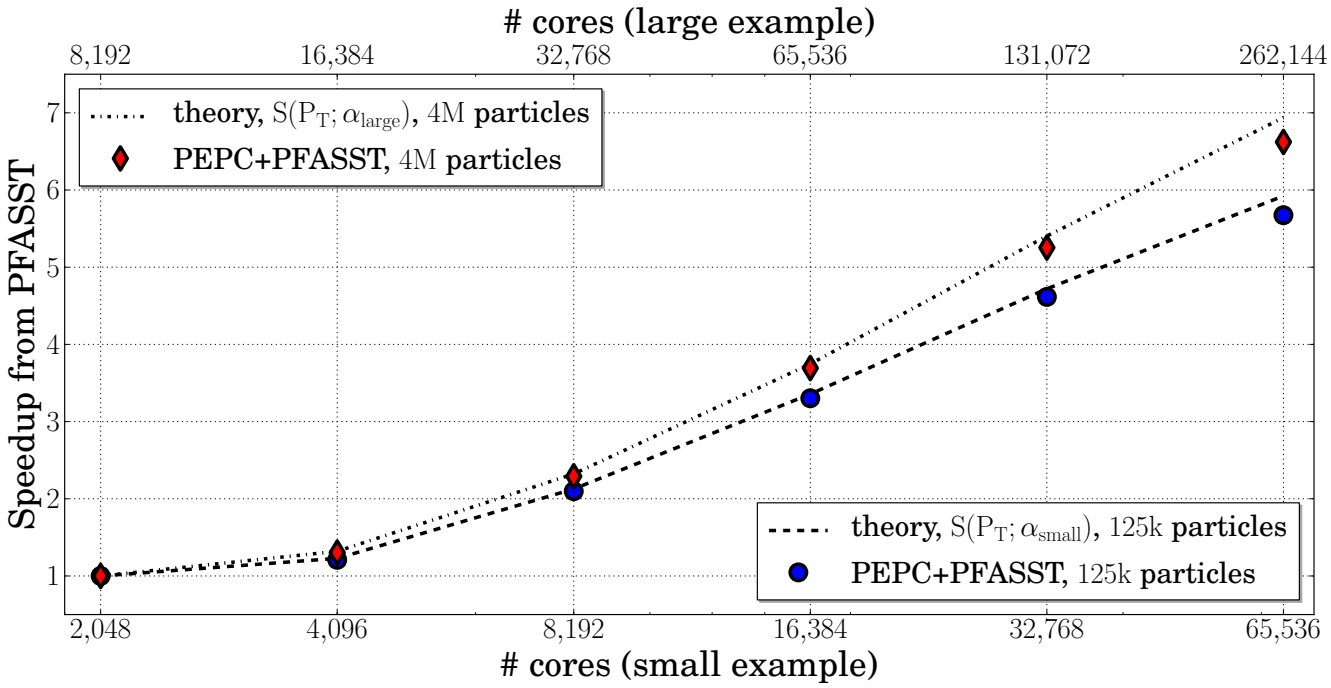
Scaling of PEPC across all 300k cores of JUGENE (Blue Gene/P).



M. Winkel, R. Speck, H. Hübner, L. Arnold, R. Krause, and P. Gibbon, 2012.

PEPC+PFASST

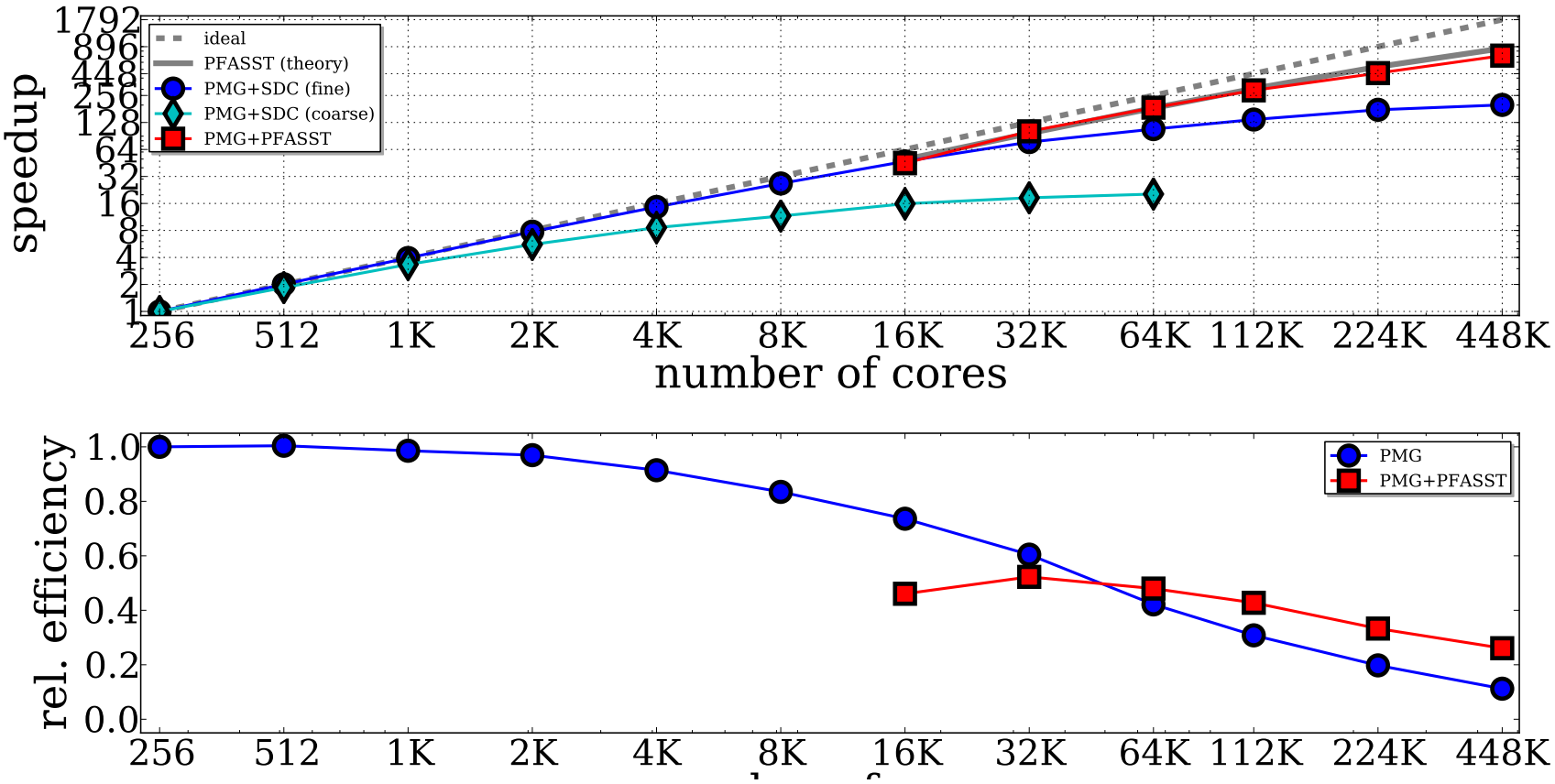
Speedup of PEPC+PFASST compared to SDC.



R. Speck, D. Ruprecht, R. Krause, M. Emmett, M. Minion, M. Winkel, and P. Gibbon, 2012

PMG+PFASST

Scaling of PMG+PFASST



D. Ruprecht, R. Speck, M. Emmett, M. Bolten, and R. Krause, 2013

Multi-physics in MLSDC

Force-fields in OpenMM

AMBER includes bonded and non-bonded forces

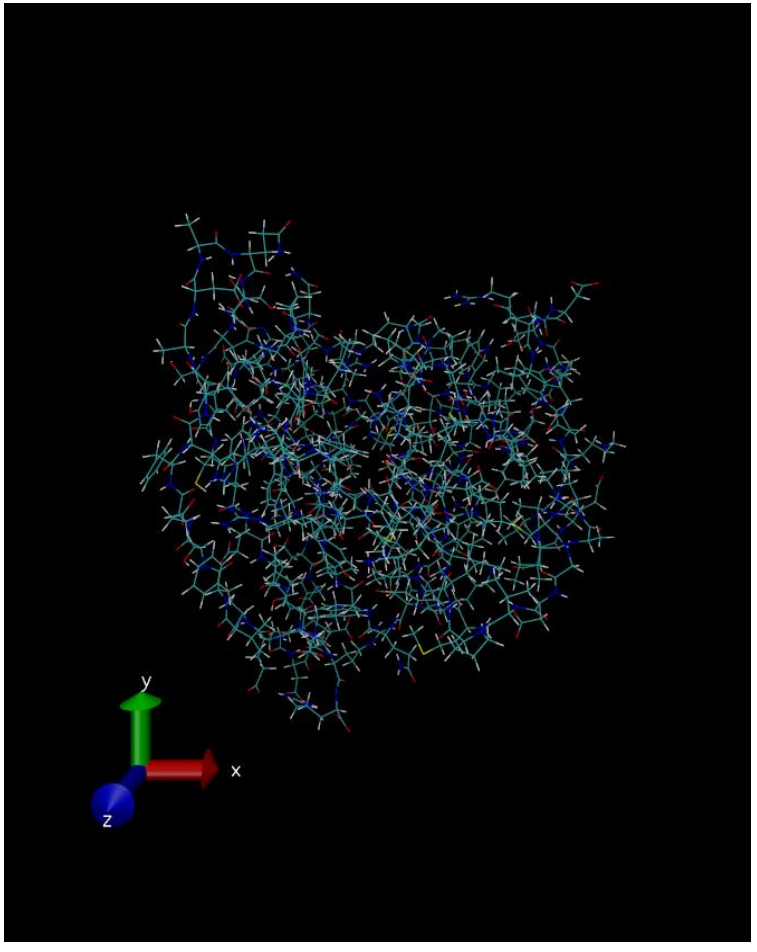
- bonded "Harmonic" forces representing covalently bonded atoms
- bonded "Angle" forces representing energy due to the geometry of electron orbitals
- bonded "Torsion" forces representing the energy of twisting a bond
- non-bonded van der Waals forces
- non-bonded electrostatic forces
- does not include polarization

AMOEBA approximates polarization and is much more expensive.

Multi-physics in MLSDC

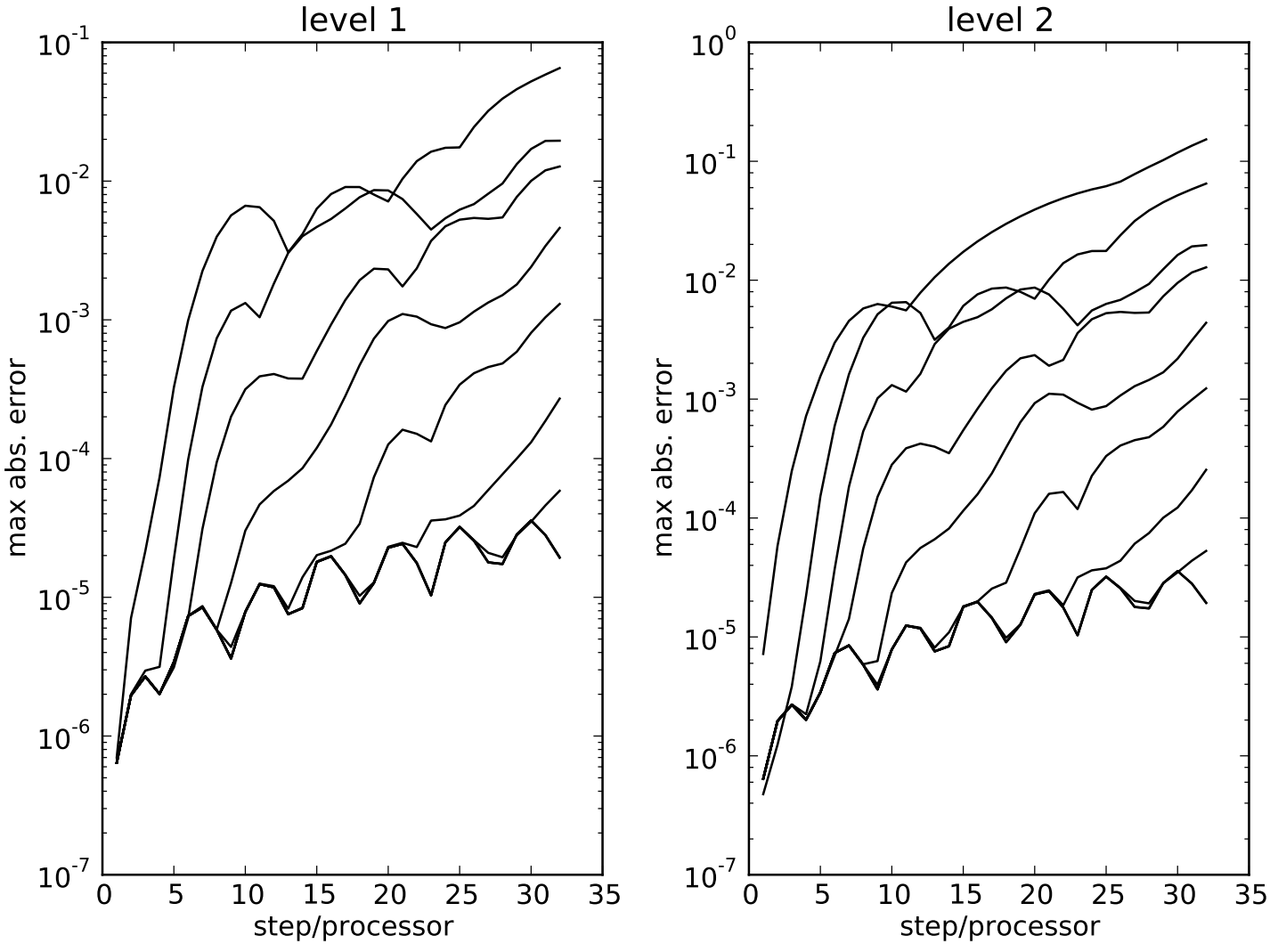
Example test problem: 5dfr protein

- 5dfr is a 1 chain of *E. Coli*
- For this example we remove the water
- There are 2489 molecules (14,934 variables)
- AMOEBA is a factor of 30-40 more expensive for this problem than AMBER

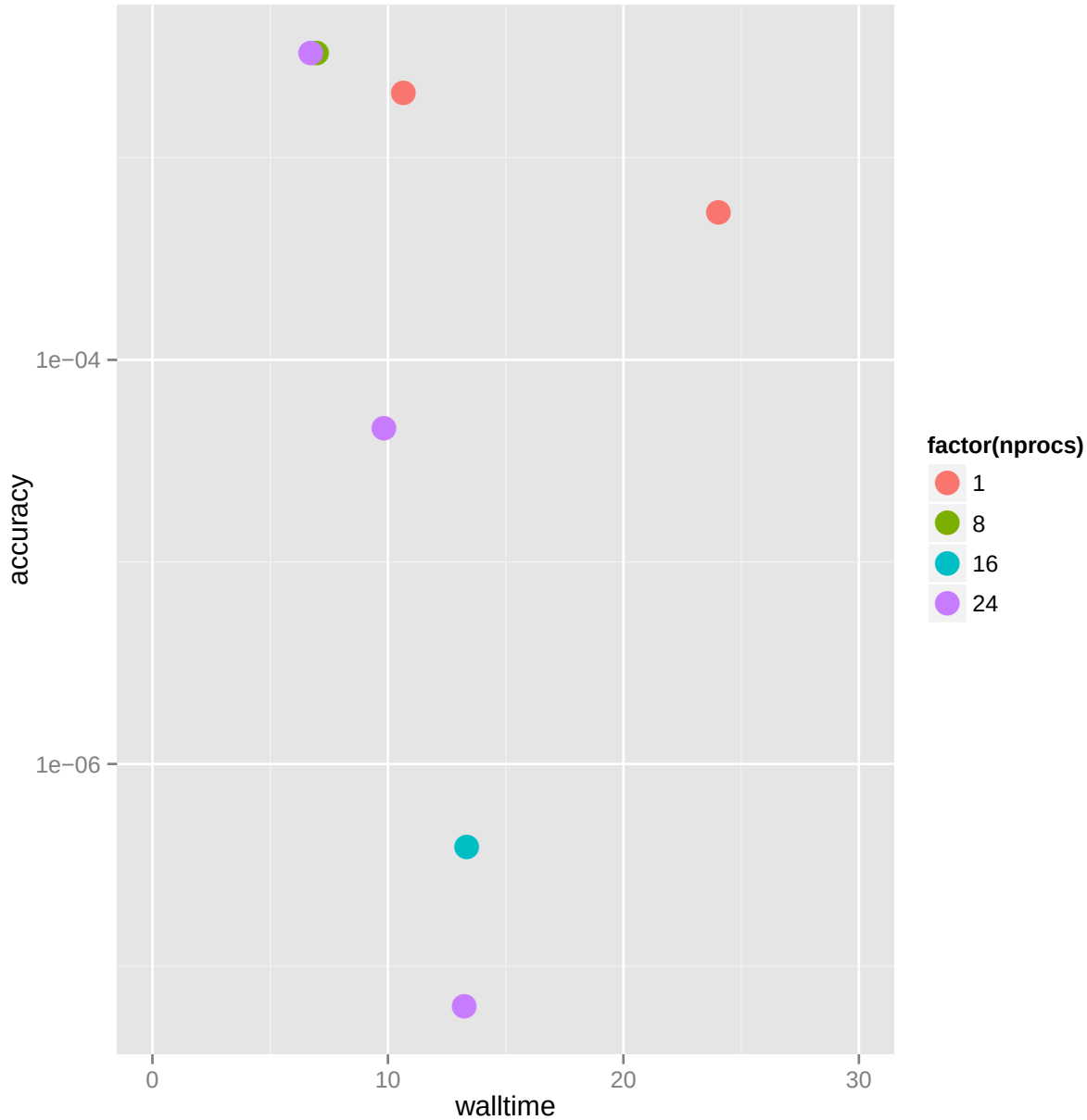


Multi-physics in MLSDC

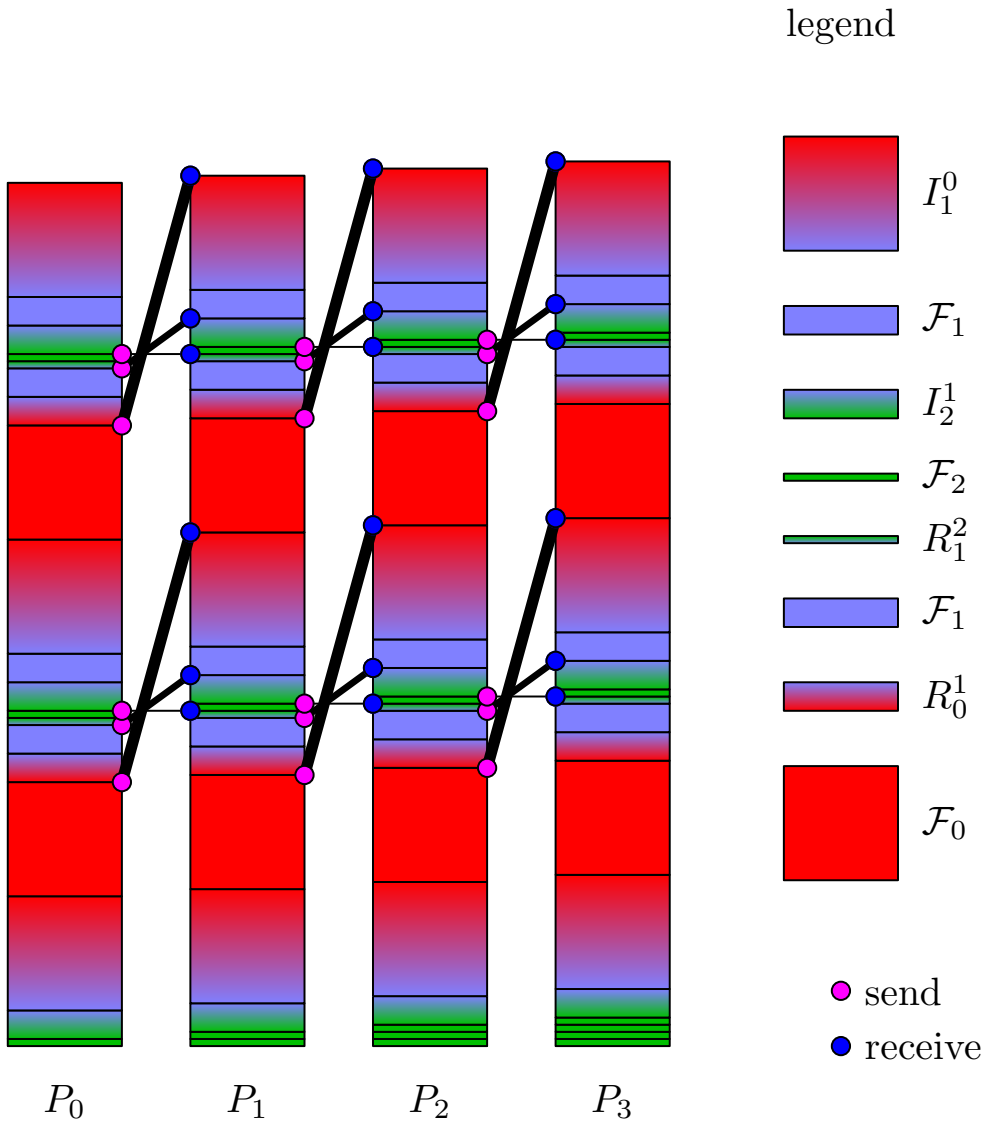
PFASST convergence on 5dfr



OpenMM+PFASST

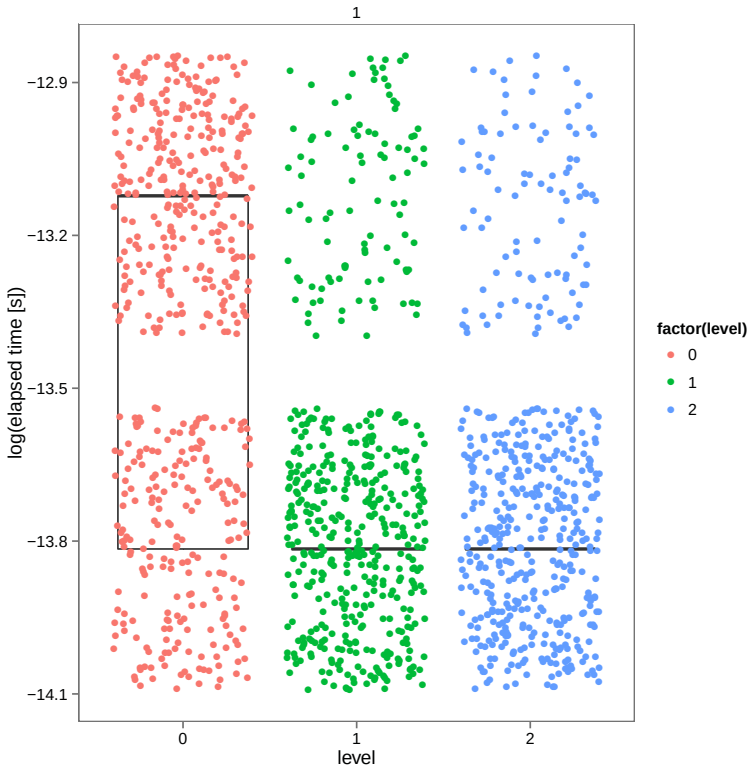


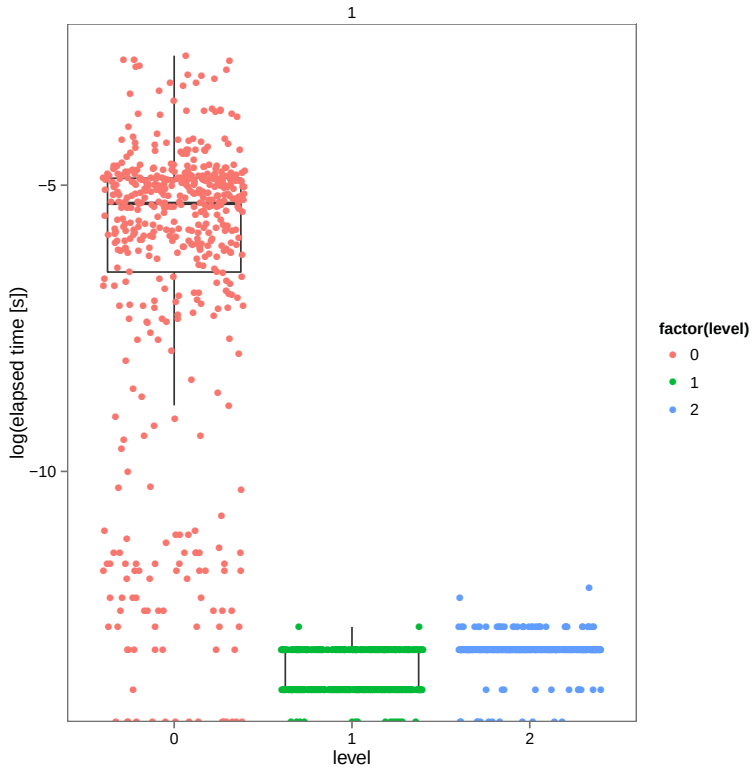
Overlapping communication and computation



PFASST timing results for a 1D eqn

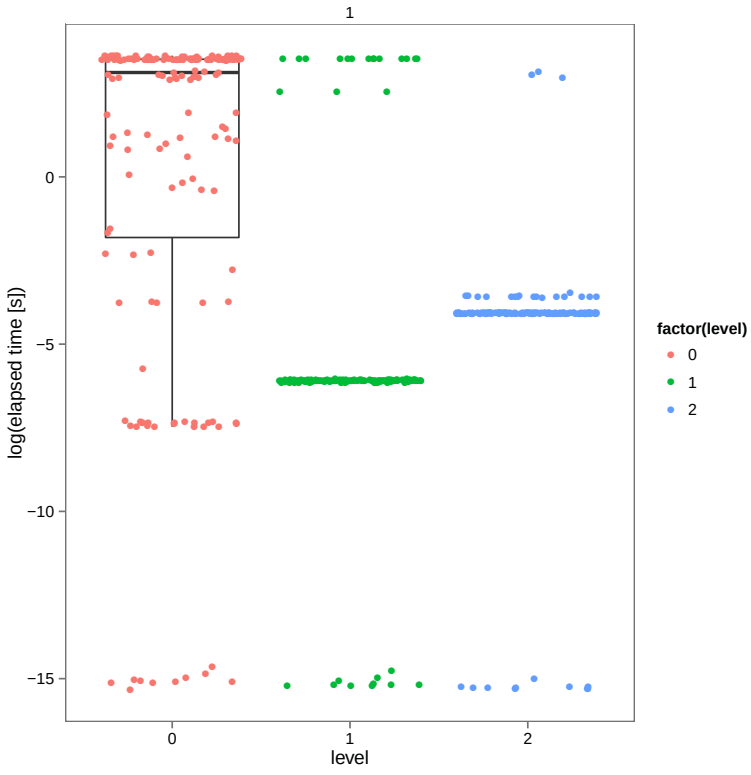
JUQUEEN (IBM) vs Edison (Cray) MPI smackdown!

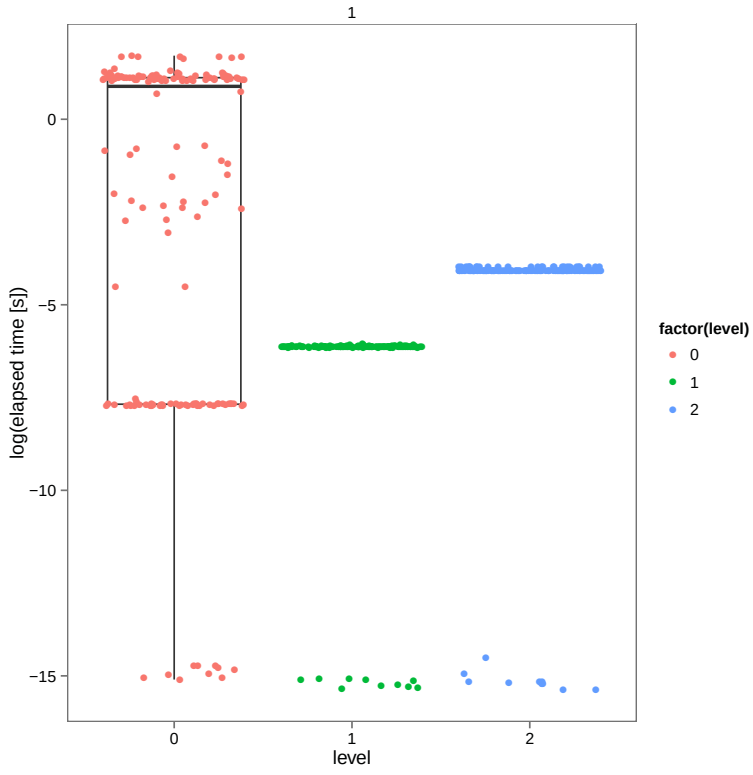




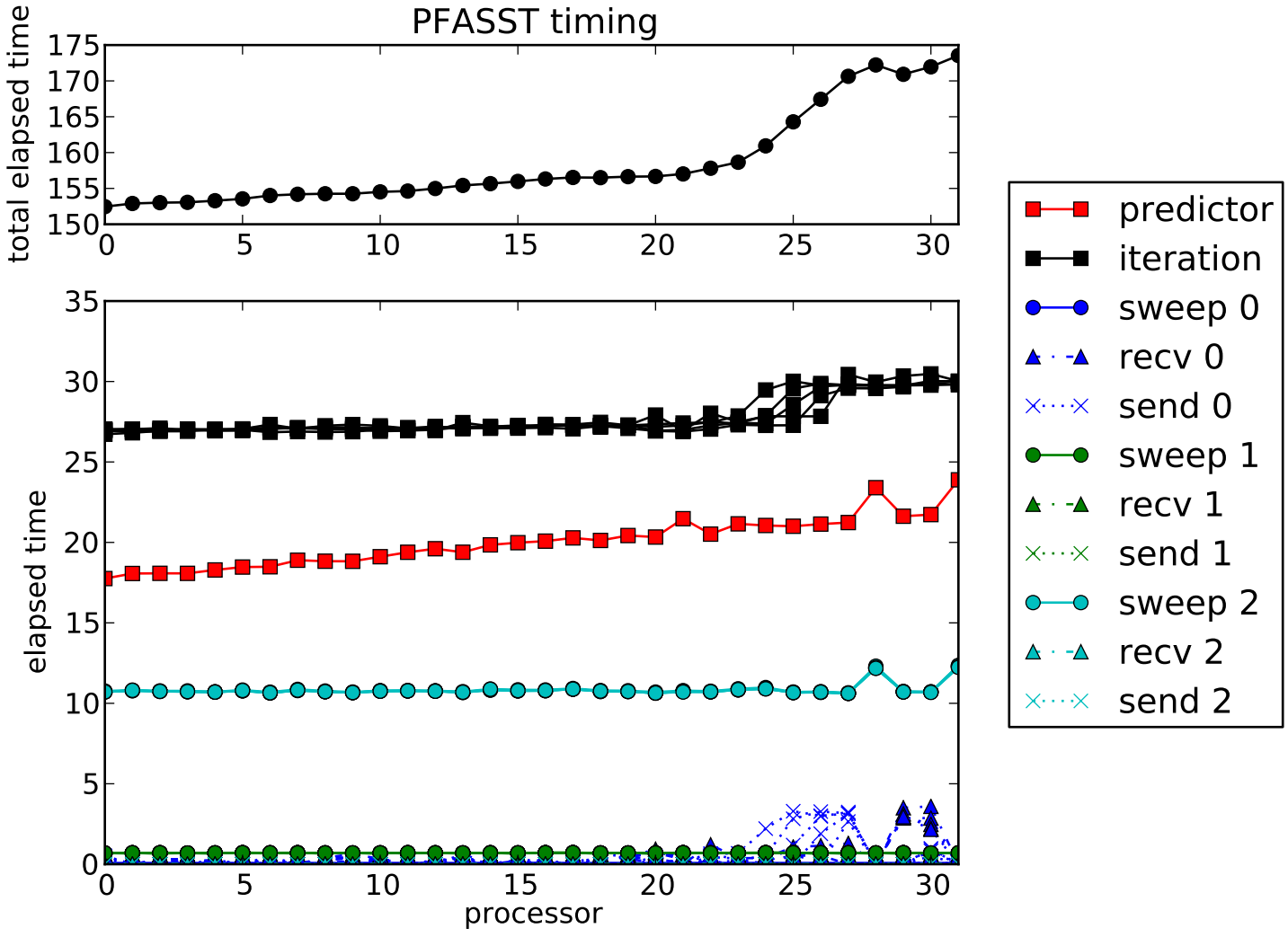
PFASST timing results for a 3D eqn

Edison with and without CPU specialization





PFASST timing results for a 3D eqn



Efficiency and implementation issues for PFASST

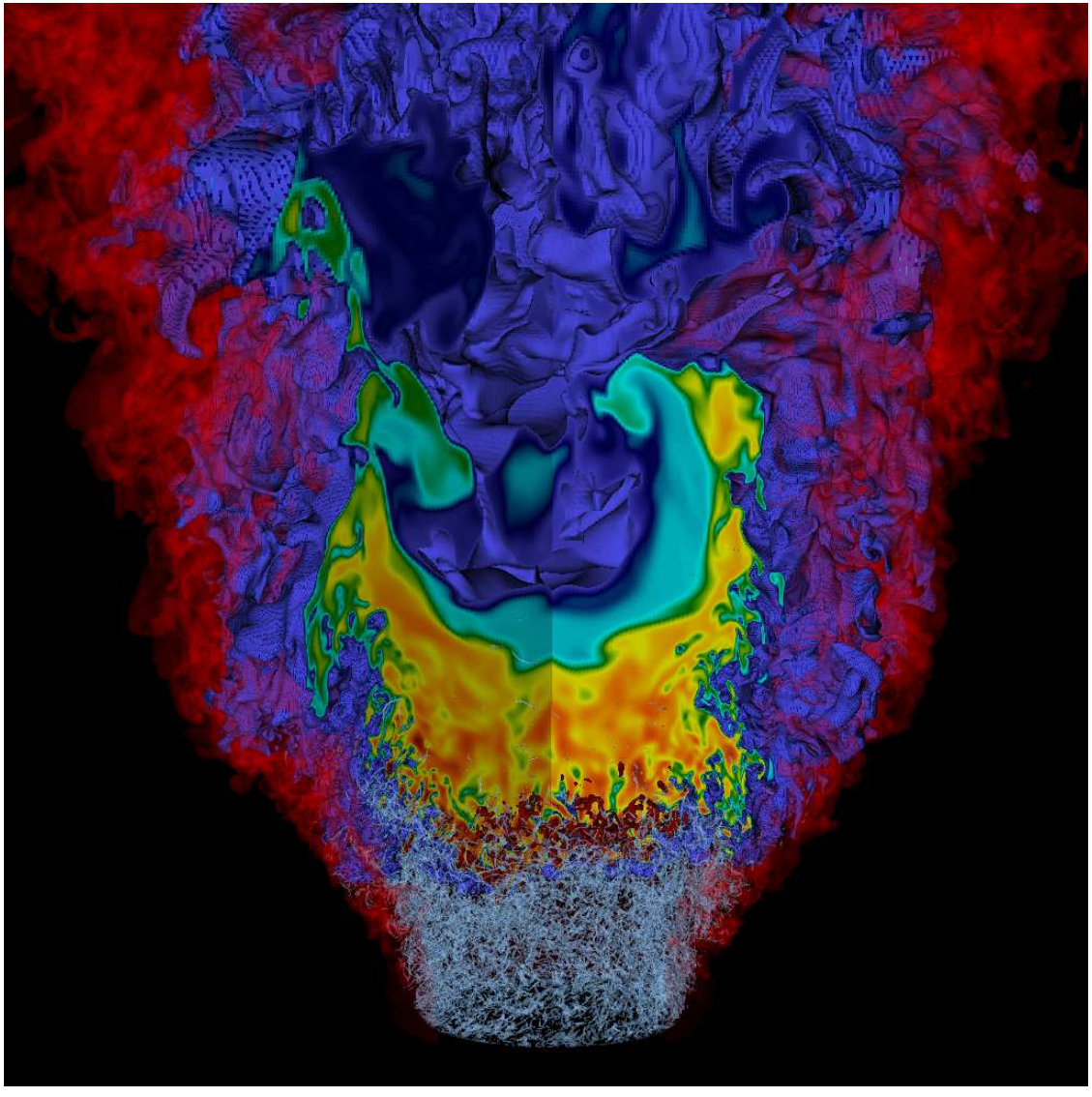
Efficiency of PFASST is governed by

- cost ratio of coarse/fine levels
- convergence rate
- communication overhead

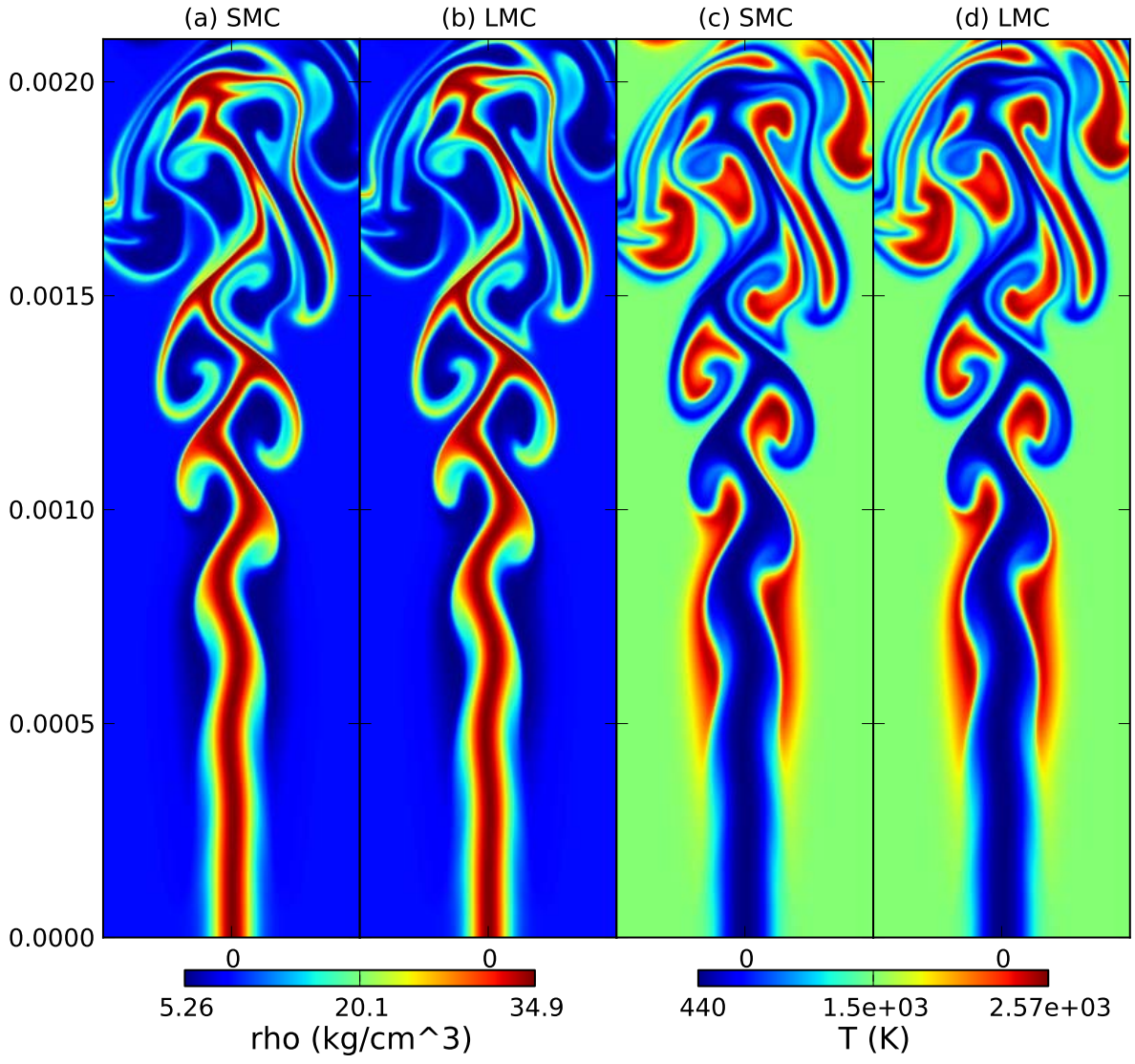
Things get complicated because of the time-communication strategy and implementation.

IMEX SDC

Low-Mach number Combustion (LMC) from CCSE

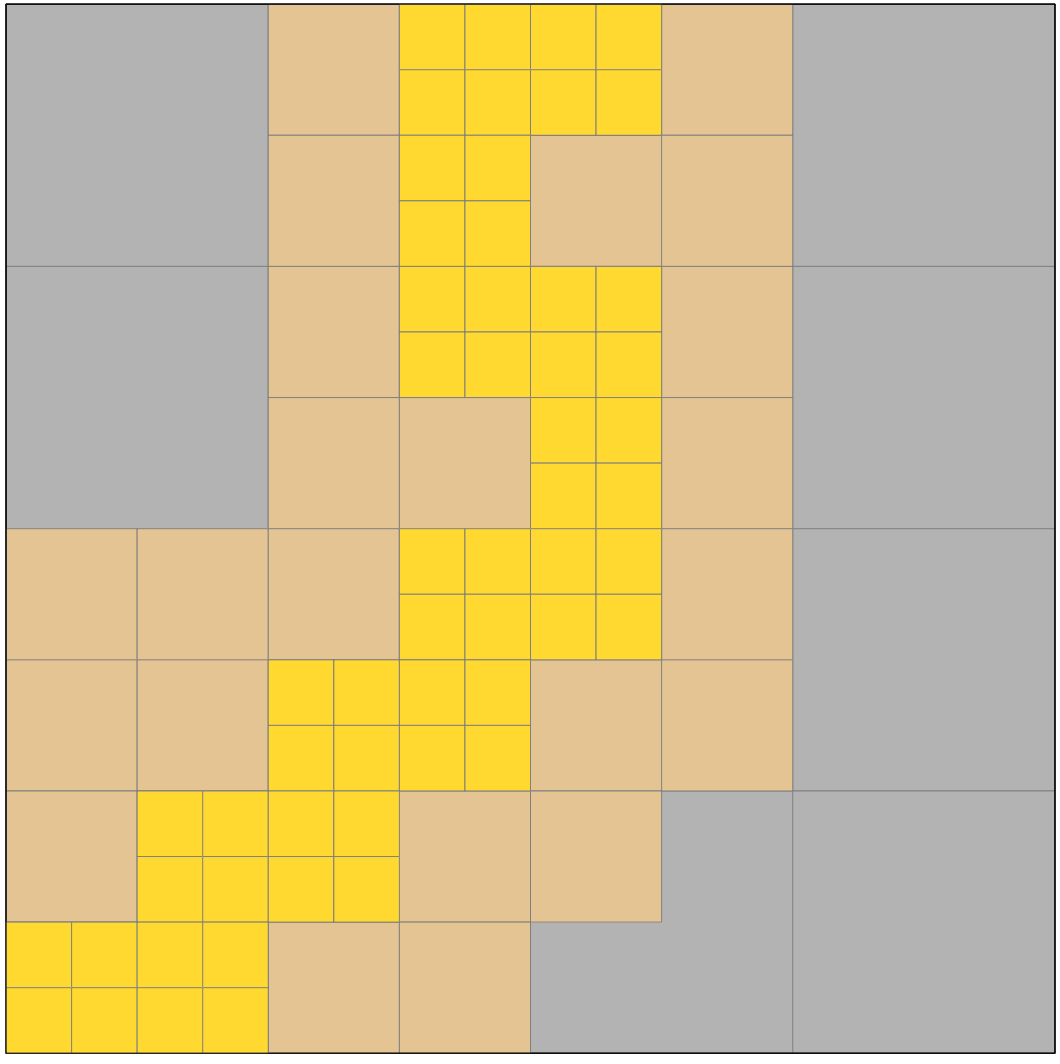


SMC dimethyl ether jet



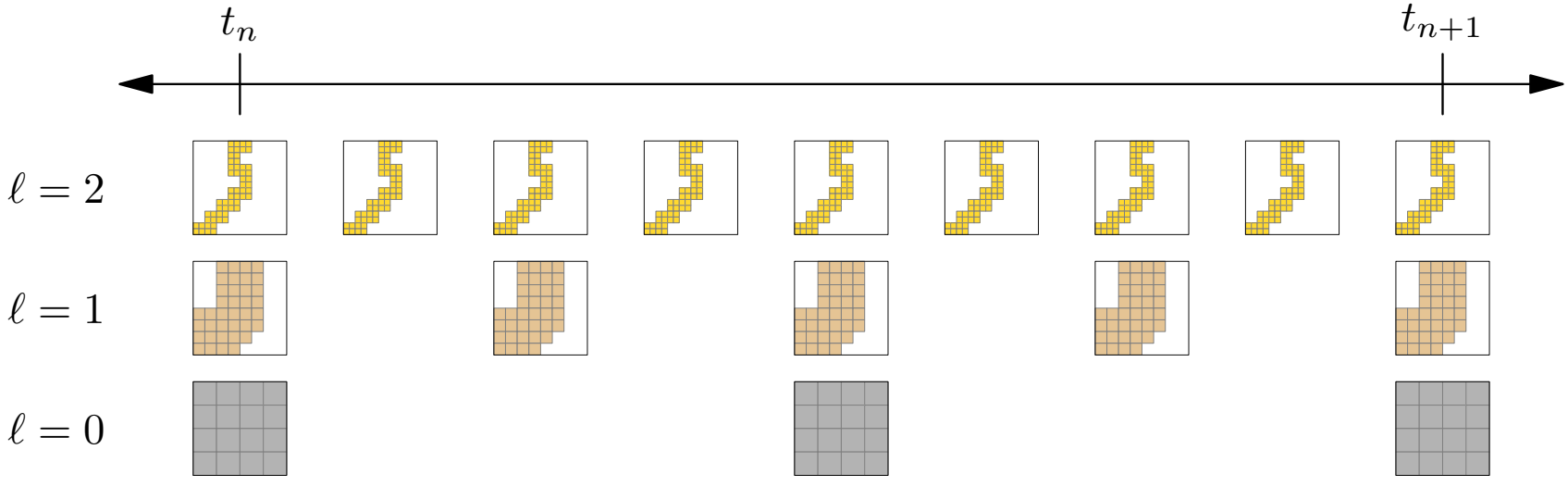
Adaptive Mesh Refinement

For finite-volume methods



MLSDC+AMR

Blazing throught the forest



MLSDC+AMR

Preliminary results for Reacting Navier-Stokes (RNS)

- AMR re-fluxing naturally folded into FAS correction.
- 4^{th} order space/time convergence for compressible Navier-Stokes.
- Coarse/fine boundaries are formally 3^{rd} order accurate.
- Chemistry is done point-wise using an IMEX technique, where the implicit solve is actually a call to a variable order/variable step BDF scheme with an analytic Jacobian.

Thanks!

Thanks to R. Speck, M. Bolten, D. Ruprecht, and R. Krause for organizing the conference.