Parallelization in Time: Applications to Plasma Turbulence
or
How Development by Computer Scientists Led to Easier, Faster, and More Computationally Efficient Applications of Parareal

Lee A. Berry (lab@ornl.gov)
and
Colleagues and Collaborators

SciDAC 2011

Denver, CO

July 14, 2011

Parareal Team

L. A. Berry, W. Elwasif—***ORNL Fusion Energy/ORNL Computer Science***

D. E. Newman, D. Samaddar—***U. Alaska, Fairbanks (D.S. now at ITER)***

R. Sanchez, J.M Reynolds-Barredo—***U. Carlos III, Madrid***

SWIM SciDAC Project

D. B. Batchelor, L. A. Berry, E. F. Jaeger—***ORNL Fusion Energy***

D. E. Bernholdt, E. D'Azevedo, W. Elwasif, S. Foley—***ORNL Computer Science***

S. C. Jardin, E. Feibush, D. McCune, J. Chen, M. Chance, J. Breslau —***PPPL***

G. Abla, D. P. Schissel – ***General Atomics*** , **R. W. Harvey**—***CompX***

R. Bramley– ***Indiana University,*** **D. Keyes** – ***Columbia University,*** **D. Schnack**—***U. Wisconsin***

P. T. Bonoli, J. Ramos, J.Wright – ***MIT,*** **S. Kruger, T. Jenkins**—***TechX*****, G. Bateman**—***Lehigh University***

Unfunded participants:

L. Sugiyama – ***MIT,*** **J. D. Callen, C. C. Hegna, C. Sovinec** – ***University of Wisconsin,*** **E. Held** – ***Utah State***

H. St. John – ***General Atomics,*** **A. Kritz** – ***Lehigh Univ*****.**

Summary

- **Wall clock times can be an issue for some simulations—parareal is one approach to addressing this issue.**
- **At least one model of plasma turbulence can be accelerated (wall clock) with parareal.**
- **Use of a simulation framework, the Integrated Plasma Simulator, will enable new applications with**
  - **less coding work;**
  - **less demanding solver development;**
  - **reduced and more efficient use of HPC resources.**

Parareal is an algorithm for iterative parallelization of time

- **J. Lions, Y. Maday, G. Turinici, "A parareal in time discretization of pde's," CR Acad. Sci. I – Math. 332 (7) (2001) 661–668.**
- **Applications include CFD, heat transport, MD, chemical kinetics, and many more.**
- **Parareal requires a fast coarse solver, G, and a fine solver, F**
  - **F contains the desired physics and resolution**
  - **G is a fast coarse-grained solver that has enough physics, resolution, etc., to allow effective convergence.**
- **Also required are:**
  - **A measure of convergence;**
  - **Operators for transforming states of G and F as inputs to F and G, respectively;**
  - **Initial states for F and G.**

Parareal: the algorithm

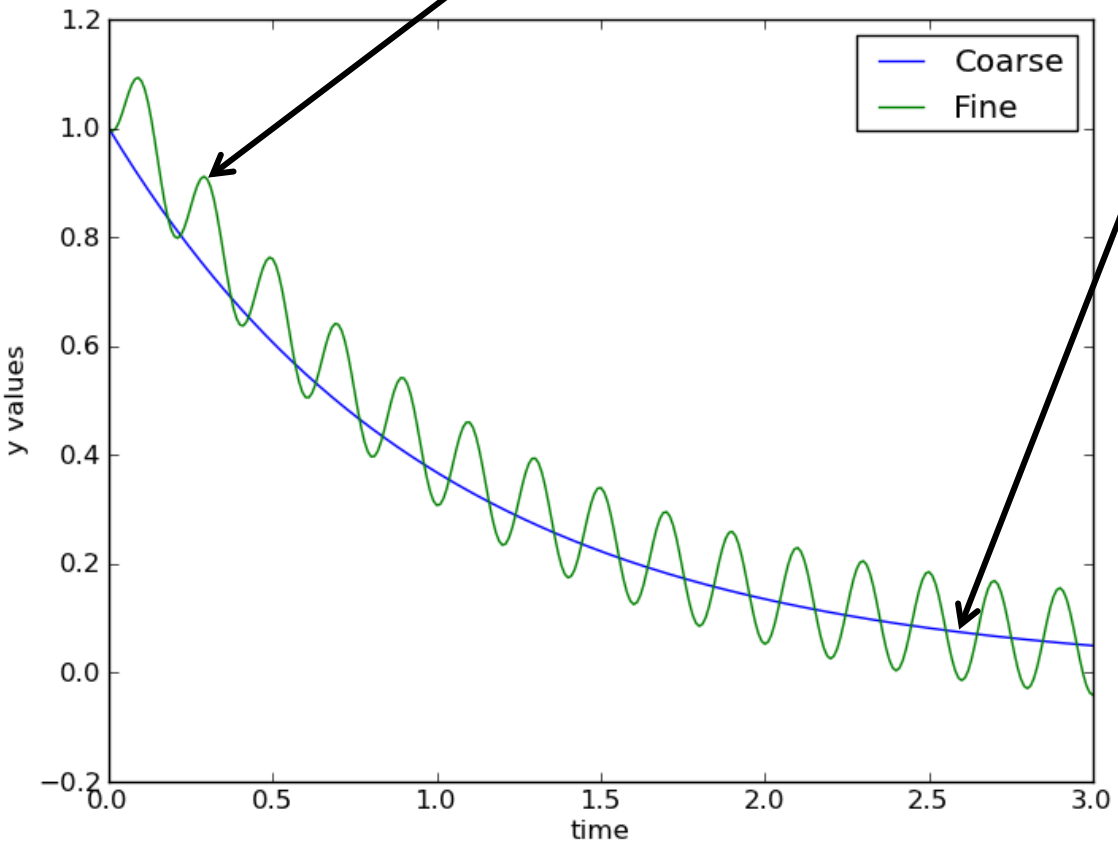
- **Represent the state of the system by:** λ_i^k
  where
  i **=> time index and**
  k **=> iteration index.**
- **The coarse and fine solvers can be represented as propagators that take** λ_i^k **to** λ_{i+1}^k **.**
- **Namely** $\lambda_{i+1}^k = G(\lambda_i^k)$ **or** $\lambda_{i+1}^k = F(\lambda_i^k)$ **.**
- **The heart of parareal is the iterate:** $\Lambda_{i+1}^k = G(\lambda_i^k) - G(\lambda_i^{k-1}) + F(\lambda_i^{k-1})$

  with initial conditions λ_0^1 **and first iteration** $\lambda_{i+1}^1 = G(\lambda_i^1)$ **and** $\lambda_{i+1}^1 = F(\lambda_i^1(\text{from G}))$ **.**

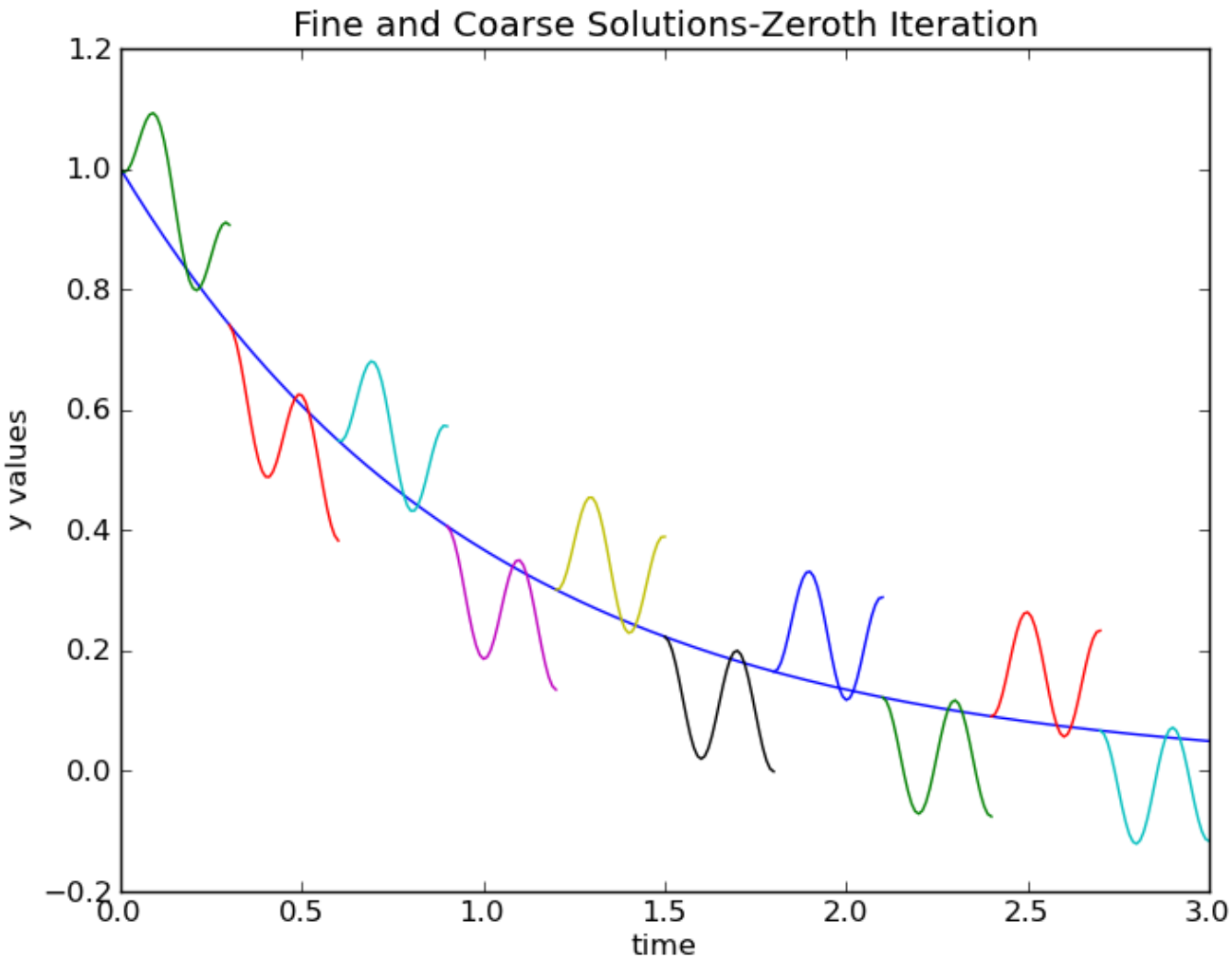
A simple example

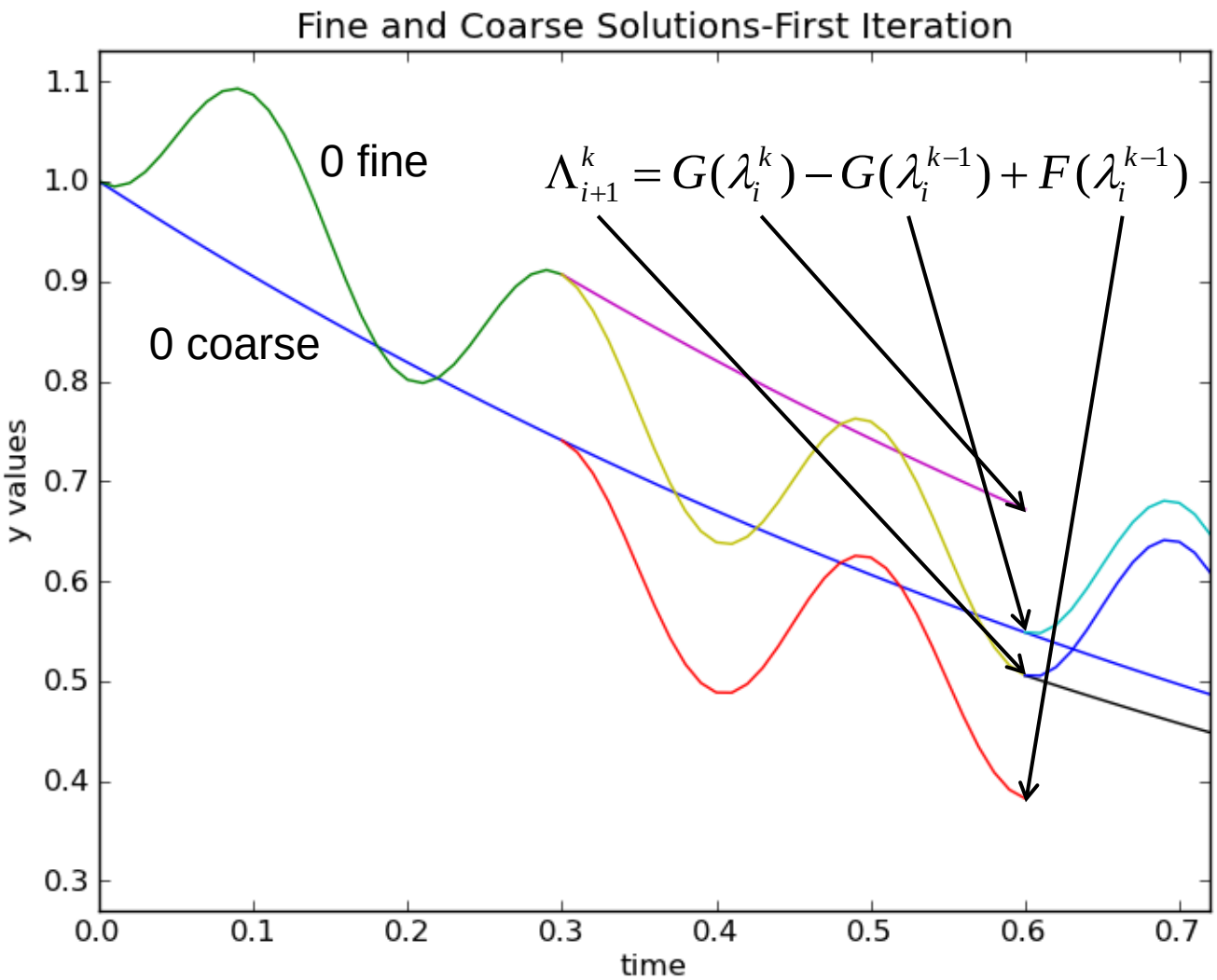
$$\frac{du}{dt} + \lambda u = \sin(10\pi t) \equiv F_{n,\Delta T}$$

$$\frac{du}{dt} + \lambda u = 0 \equiv G_{n,\Delta T}$$

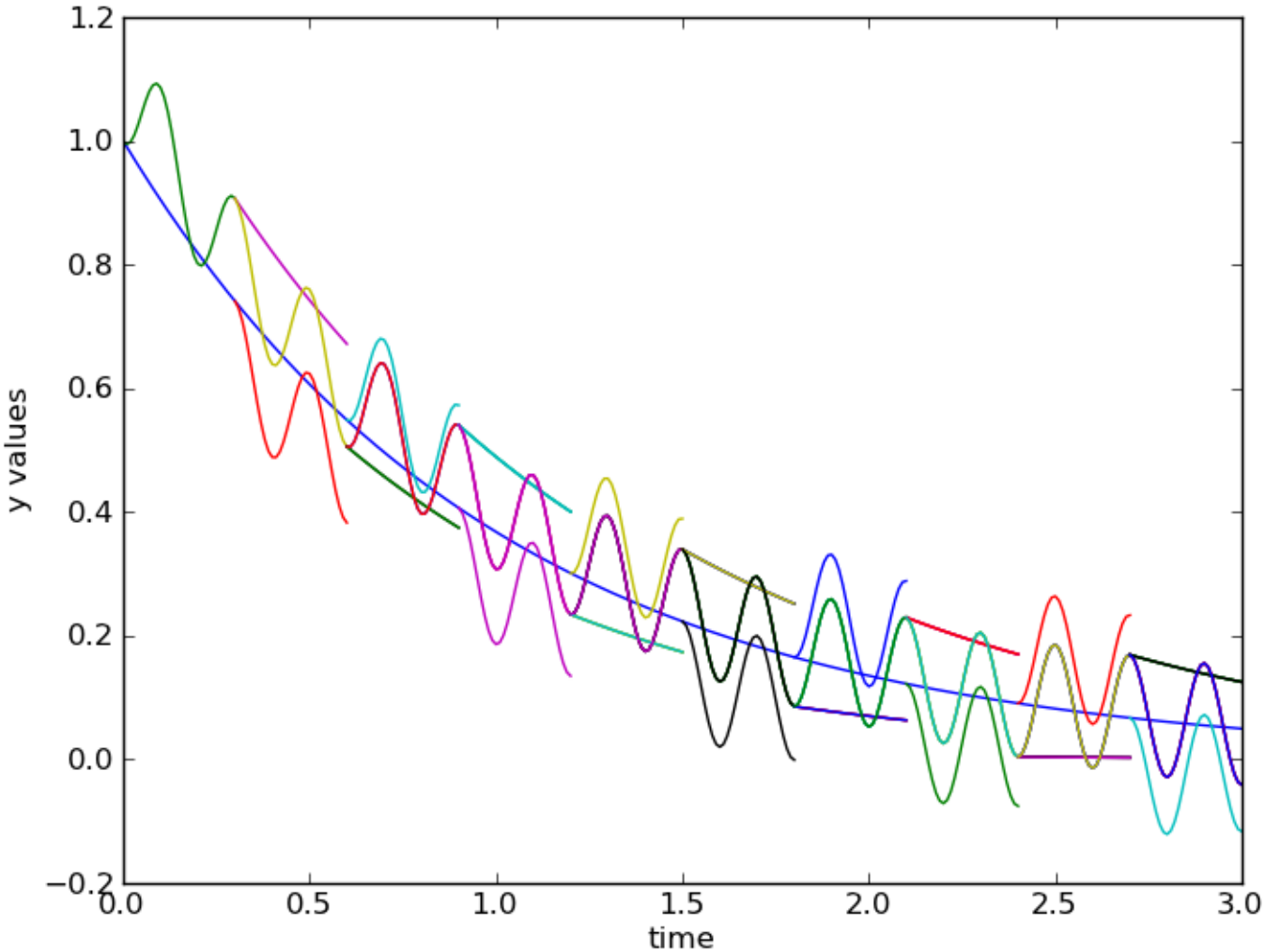


First (0th) iteration: run coarse all the way, use as initial condition for parallel fine calculations on the slices (10 total)




Fine and Coarse Solutions-First Iteration
0 fine
0 coarse
$\Lambda_{i+1}^{k} = G(\lambda_i^k) - G(\lambda_i^{k-1}) + F(\lambda_i^{k-1})$
y values
time
1.1
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7

Four superposed iterations show convergence in ~two iterations



Large high-performance computers with > $10^{5=>6,7,...}$ processors are the tool of choice to meet our goal of high-fidelity simulations

- **Spatial domain decomposition is an effective tool for using HPC capability.**
- **However, many fusion simulations are effectively limited by time dependence. Examples include:**
  - **Extended MHD (although improved, implicit, algorithms (e.g., PIXIE3D, M3D-C1) may (at least partially) break this seeming impasse);**
  - **Integrated simulations;**
  - **Microturbulence.**
- **Implicit algorithms, when developed, are often only a partial solution.**
- **In many of these cases, we would be willing to use more compute cycles in favor of reduced wall-clock time.**

Comments

- **Parareal always converges, but how well depends on the coarse solver.**
- **Developing parareal for an application requires:**
  - **Finding a good coarse solver;**
  - **Finding operators to move states back and forth between coarse and fine solvers;**
  - **Finding a convergence measure for statistical states.**
- **Given that we know what to do (above), implementation within existing code is often tedious:**
  - **Requires embedding fine, coarse, convergence, and raising and lowering procedures within one executable using, for example, multiple MPI com worlds for parallelization;**
  - **Must be done again for every new application.**

The Coarse solver is the determining factor: a wide variety of techniques have proven effective for particular applications

- **Reduced spatial and/or temporal resolution/precision.**
- **Reduced models (POD or eliminate small eigenvalues).**
- **Implicit time advances.**
- **Simplified physics.**

The key is to be able to rapidly try a range of approaches

BETA is a pseudo-spectral solver for model DTEM physics [Samaddar, Newman, Sanchez*]

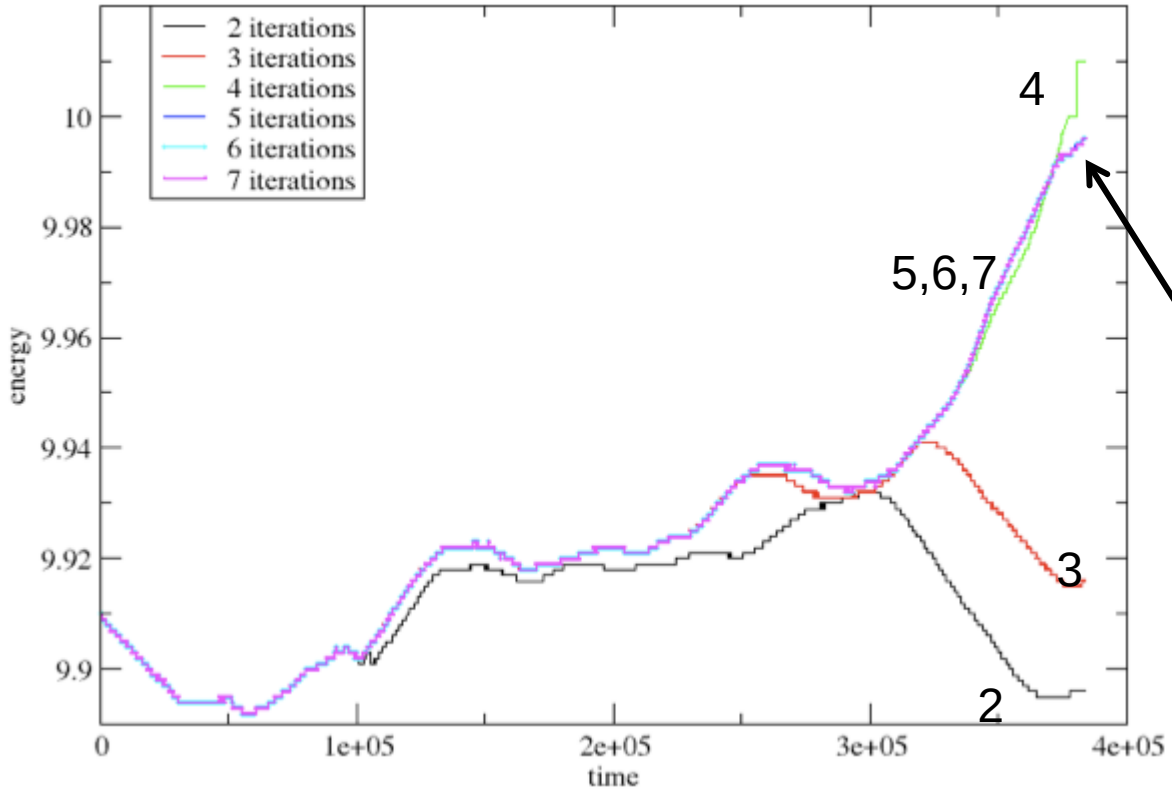
- **Hasagawa-Mima (model drift wave turbulence):**

$$\frac{\partial}{\partial t}\left(1-\rho_s^2\nabla_\perp^2\right)\phi + D\frac{\partial^2\phi}{\partial y^2} + \frac{V_D}{2}\frac{\partial\phi}{\partial y} - \frac{4L_\parallel}{\varepsilon^{1/2}}\left[\nabla_\perp\left(\frac{\partial\phi}{\partial y}\right)\times\hat{z}\right]\bullet\nabla_\perp\phi = Sources - Sinks$$

- **For the coarse solver:**
  - **Reduced spatial resolution;**
  - **Faster, less precise time integrator**
- **For fine to coarse: truncation**
- **For coarse to fine: keep high wave number coefficients from previous iteration (match slope, use random phase for $k = 1$).**
- **For convergence: total mode energy was shown to be a good proxy for convergence of low k modes.**

*J. Comput. Phys. **229** (18) (2010) 6558.

Example: convergence (to 1.5e-6) in seven iterations for 88 time slices--speed up ~10

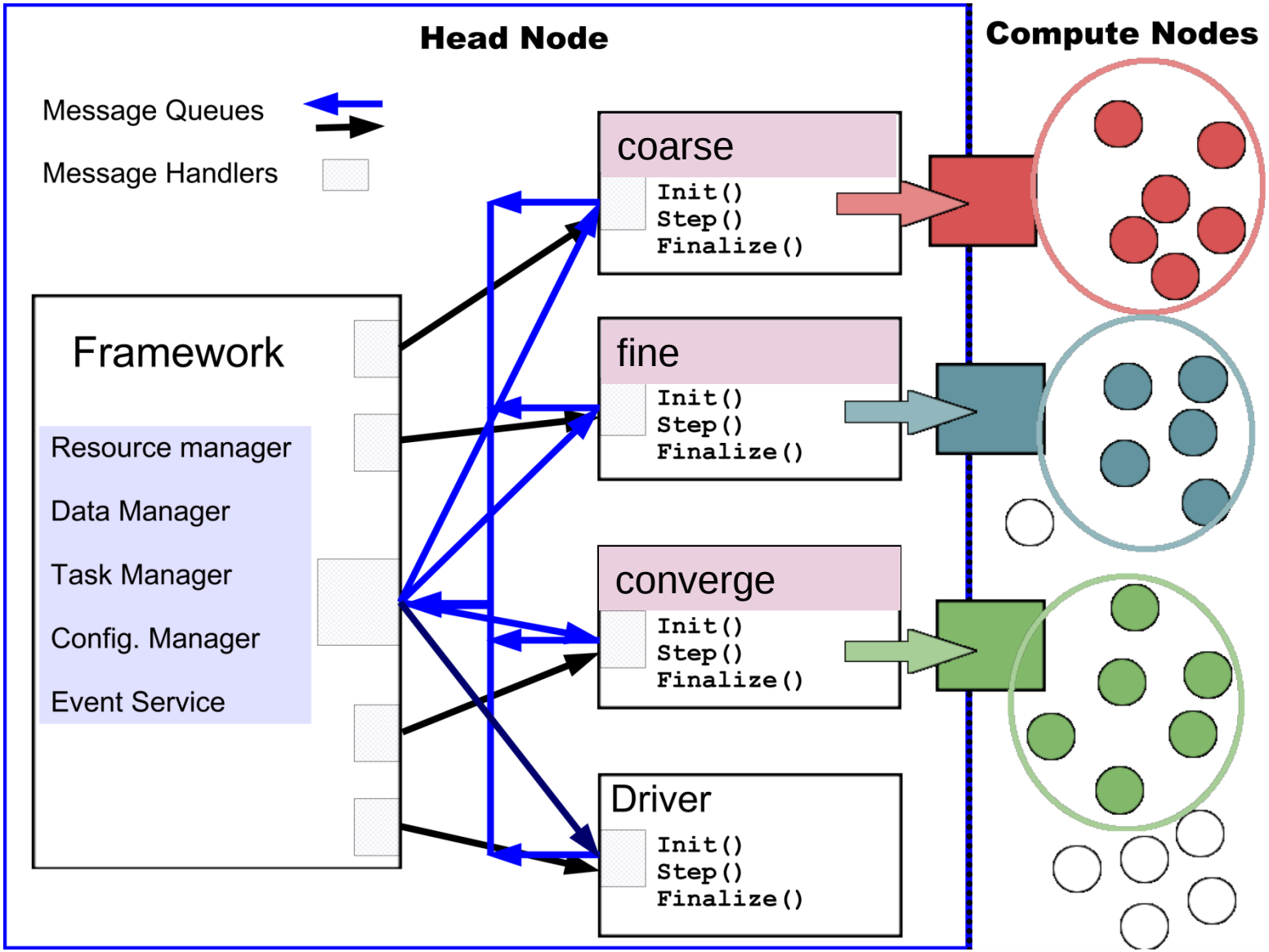


to 1.5e-6 normalized error

The successful application of parareal to BETA motivated the implementation of parareal using the IPS

- **The goal of this effort was to enable much faster implementation and development of parareal for new applications:**
  - **avoid programming issues of bringing multiple components of parareal into one executable;**
  - **a separable coarse solver component would facilitate experimentation with different approaches;**
  - **BETA implementation would provide a template for new applications of parareal.**
- **The Integrated Plasma Simulator (IPS) is a framework that was developed by the Simulation of Wave Interaction with MHD (SWIM) SciDAC project that was jointly sponsored by the Fusion Energy Science and Advanced Scientific Computing.**

The Integrated Plasma Simulator (IPS): A Light-Weight Python Framework



Ongoing applications of the IPS

Examples of fusion applications within the SWIM project

- **Whole device modeling – ITER discharge simulations**
- **Studies of RF stabilization of MHD instabilities in tokamaks**
- **Studies of generation of energetic particle populations by RF waves**

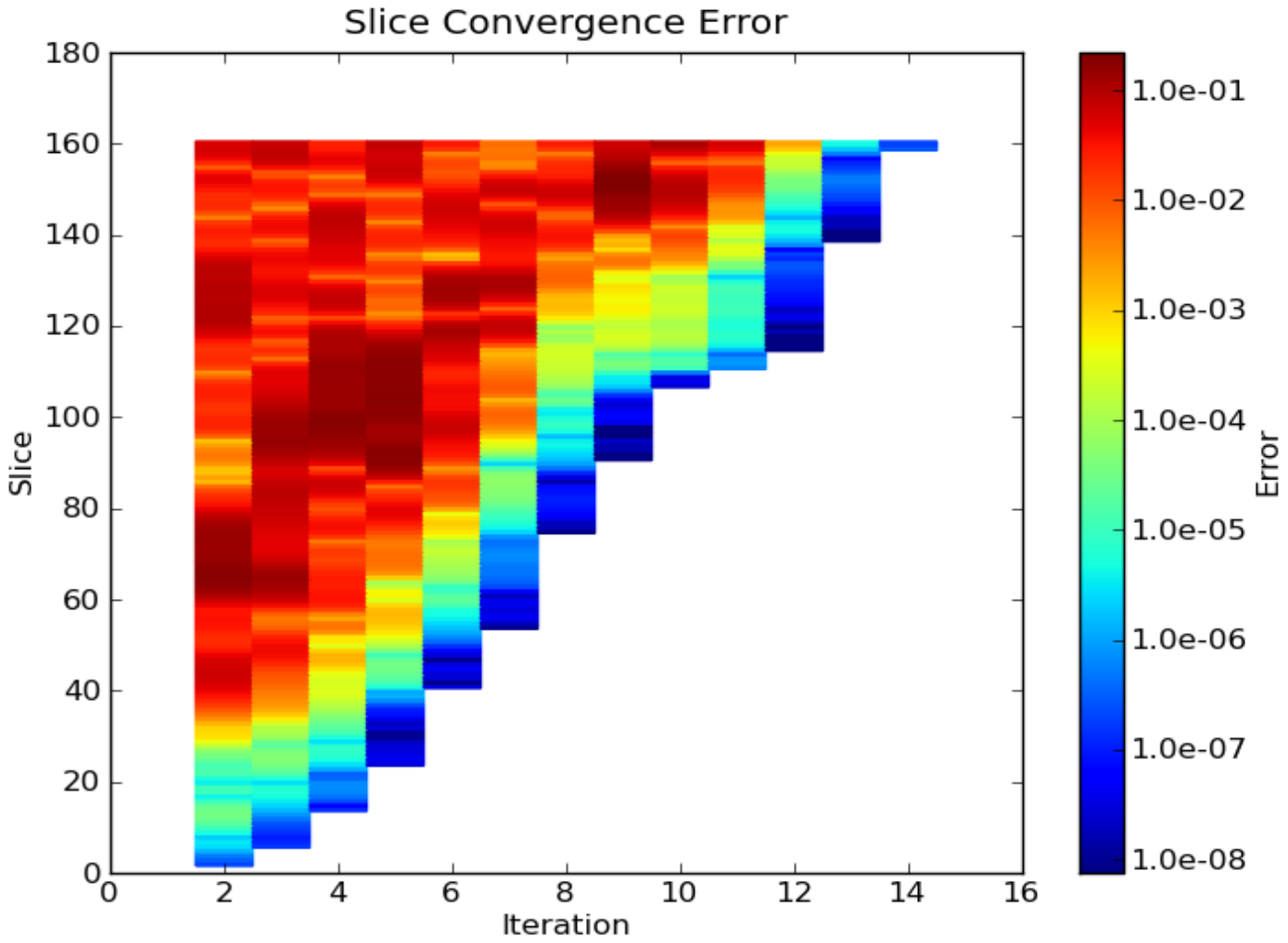
Fusion applications external to the SWIM project

- **ITER organization (Cadarache France) – transport modeling**
- **U. Alaska – extending parareal algorithm to gyrokinetics simulations**
- **Plasma State adopted by several projects as a medium of data exchange**

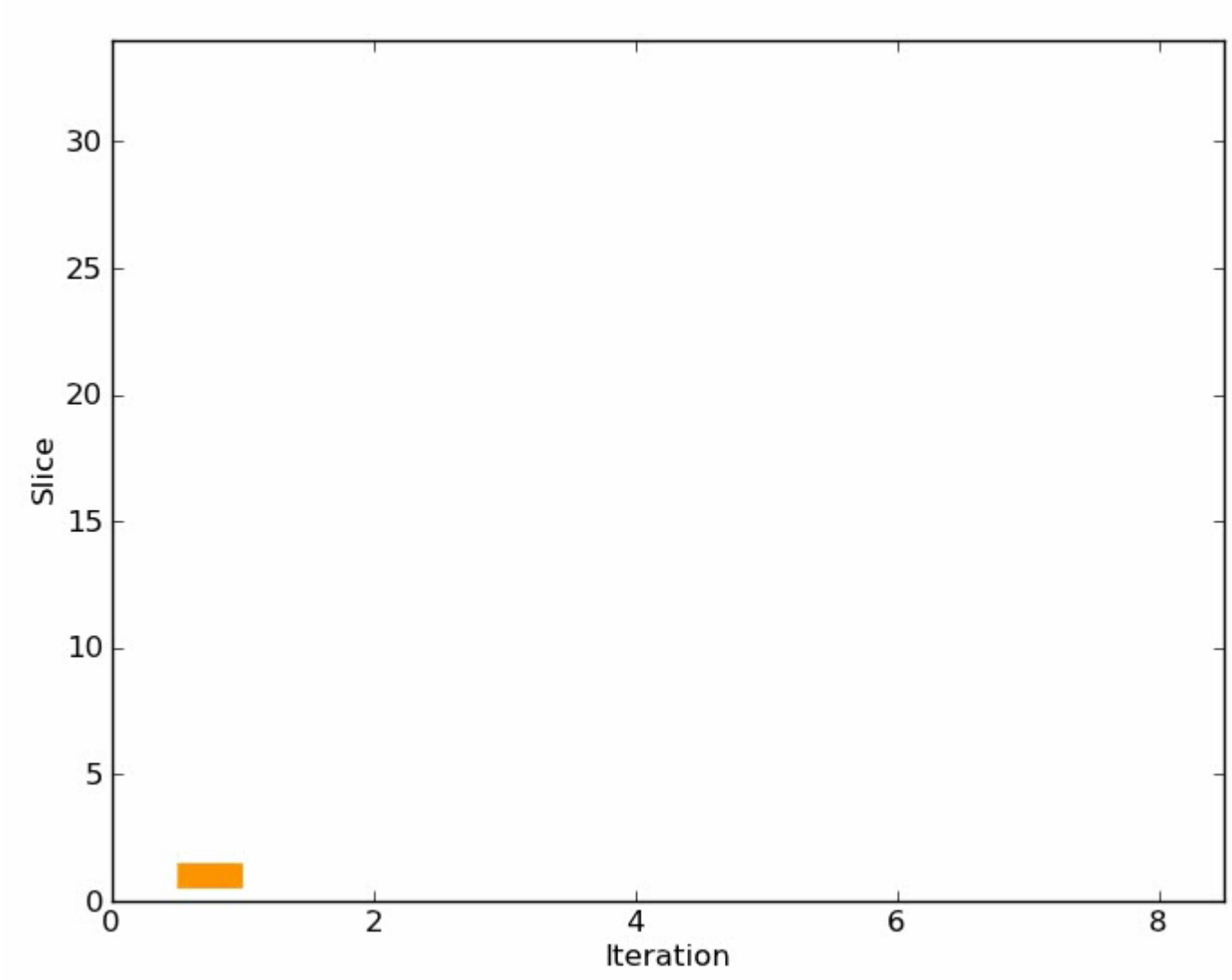
Applications outside fusion

- **Being used in project to simulate lithium ion batteries under Energy Efficiency and Renewable Energy (EERE)--Computer Aided Engineering for Batteries (CAEBAT)**
- **Discussions with UC Berkeley on application of IPS to ground water percolation**

IPS sequential implementation reproduced (exactly) the MPI, single-executable result



Task flow for sequential parareal



coarse start

coarse stop

fine start

fine stop

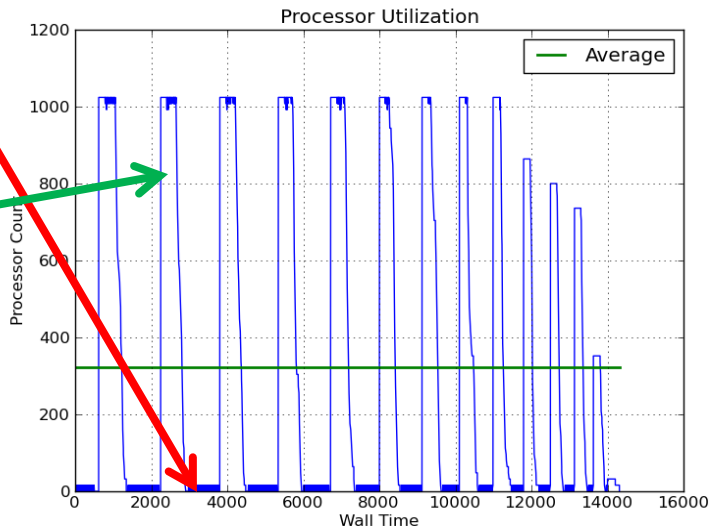
IPS Parareal employs multi-task parallelism for fine solves using a task pool

```
first_slice = 1
num_converged = 0
for iteration = 1, max_iterations
   for slice = first_slice..num_slices
      coarse_solve(iteration, slice)

   forall slice = first_slice..num_slices
      fine_solve(iteration, slice)

   for slice = first_slice..num_slices
      test_convergence(slice)

   num_converged +=
      first_non_converged_slice - first_sli
   if (num_converged == num_slices)
      end // SUCCESS
   else
      first_slice = first_non_converged_slice
end //Failed to converge in max_iteration
```



OAK RIDGE National Laboratory

The importance of the coarse solver, can be seen if we estimate the speed up

- **Model parameters;**
  - **Time for one coarse slice** T_G
  - **Time for one fine slice** T_F
  - **Number of slices:** N
  - **With** $\beta = \frac{T_F}{T_G}$
  - **The time for a sequential run is give by** $T_{sequential} = NT_F$
  - **For a parareal run the time is** $T_{parareal} \approx K(N)(NT_G + T_F)$
  - **where** K **is the number of iterations to convergence**
  - **And the speed up given by the sequential/parareal ratio**

$$H = \frac{NT_F}{K(NT_G + T_F)} = \frac{N}{K}\frac{1}{\left(\frac{N}{\beta}+1\right)}$$

An examination of the work flow suggests that the time for the coarse solver can be hidden by changing the work flow to execute a task when the data are available

- **Demand on the coarse solver is reduced by ~*K***

$$T^{event}_{parareal} \approx KT_F + (N-K)T_G$$

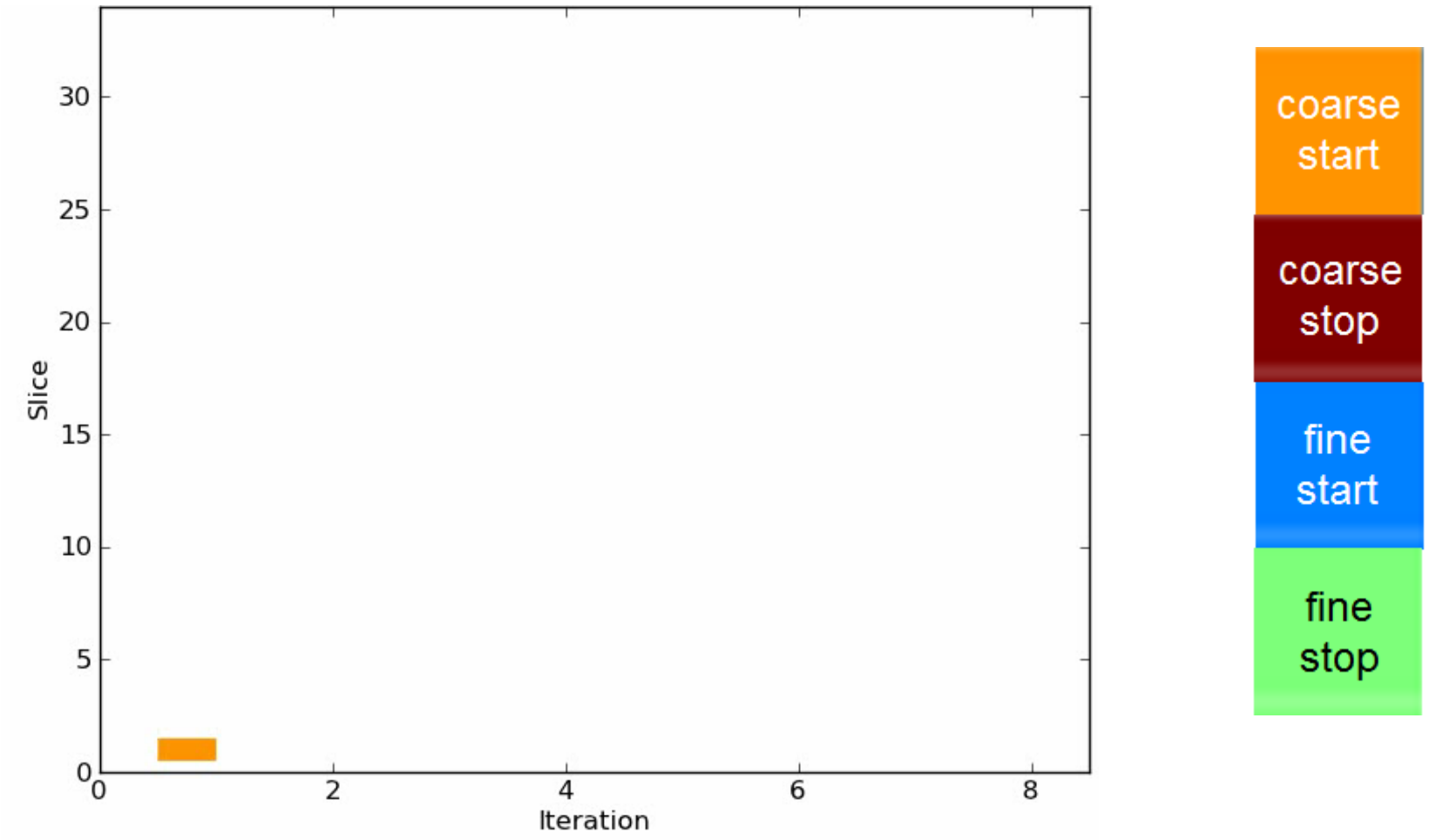
$$T_{serial} = NT_F$$

$$H_{event} = \frac{NT_F}{KT_F + (N-K)T_G} \approx \frac{N}{K}\frac{1}{\left(\frac{N}{K\beta}+1\right)}$$

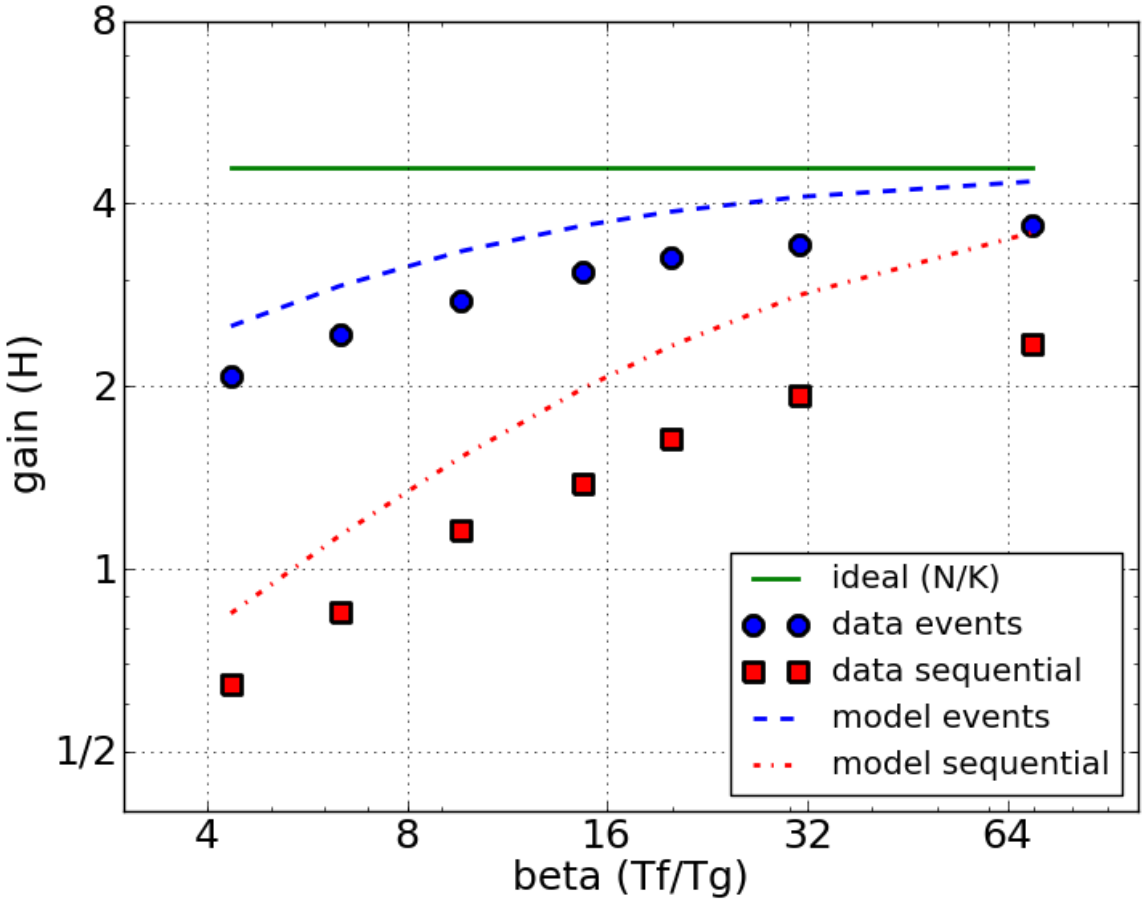
compared to

$$H_{sequential} = \frac{NT_F}{K(NT_G + T_F)} = \frac{N}{K}\frac{1}{\left(\frac{N}{\beta}+1\right)}$$

Coarse and fine executions overlap, and tasks from several iterations are carried out in parallel



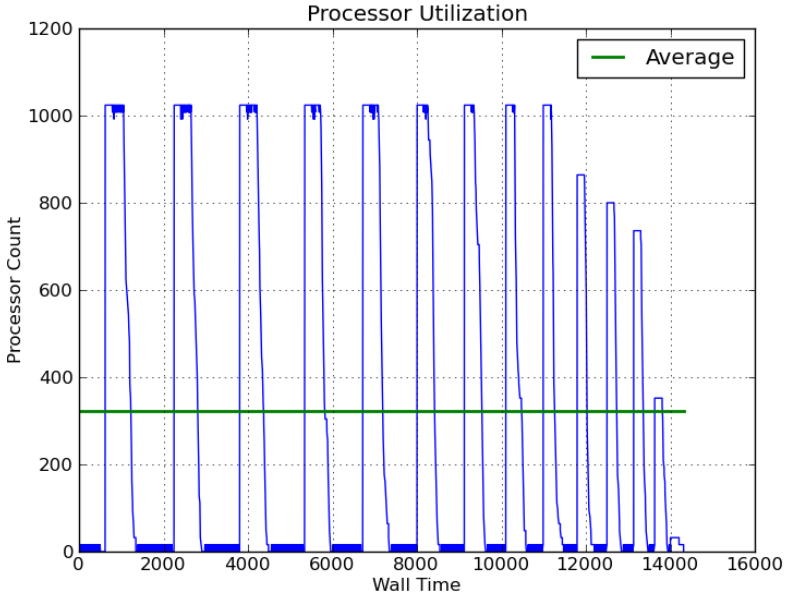
The event based parareal implemented with the IPS showed the expected gains in wall-clock times

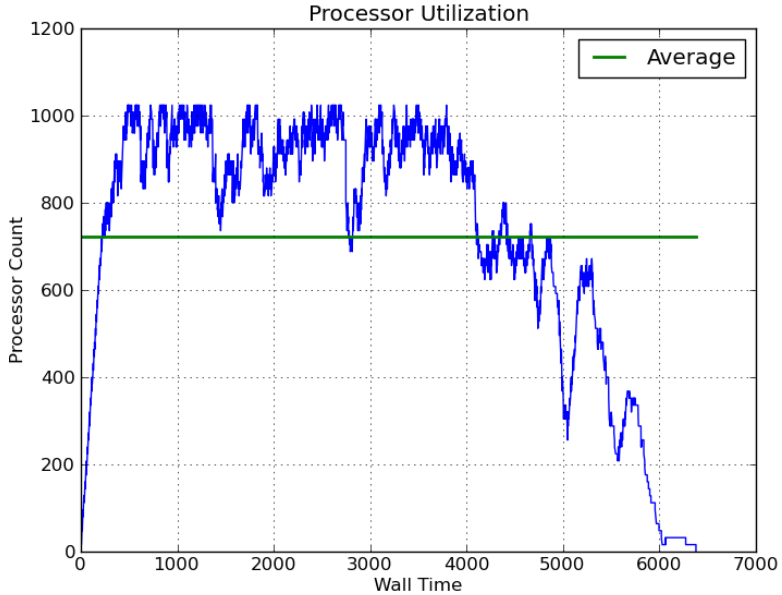


$$H_{event} \approx \frac{N}{K} \frac{1}{\left(\frac{N}{K\beta}+1\right)}$$

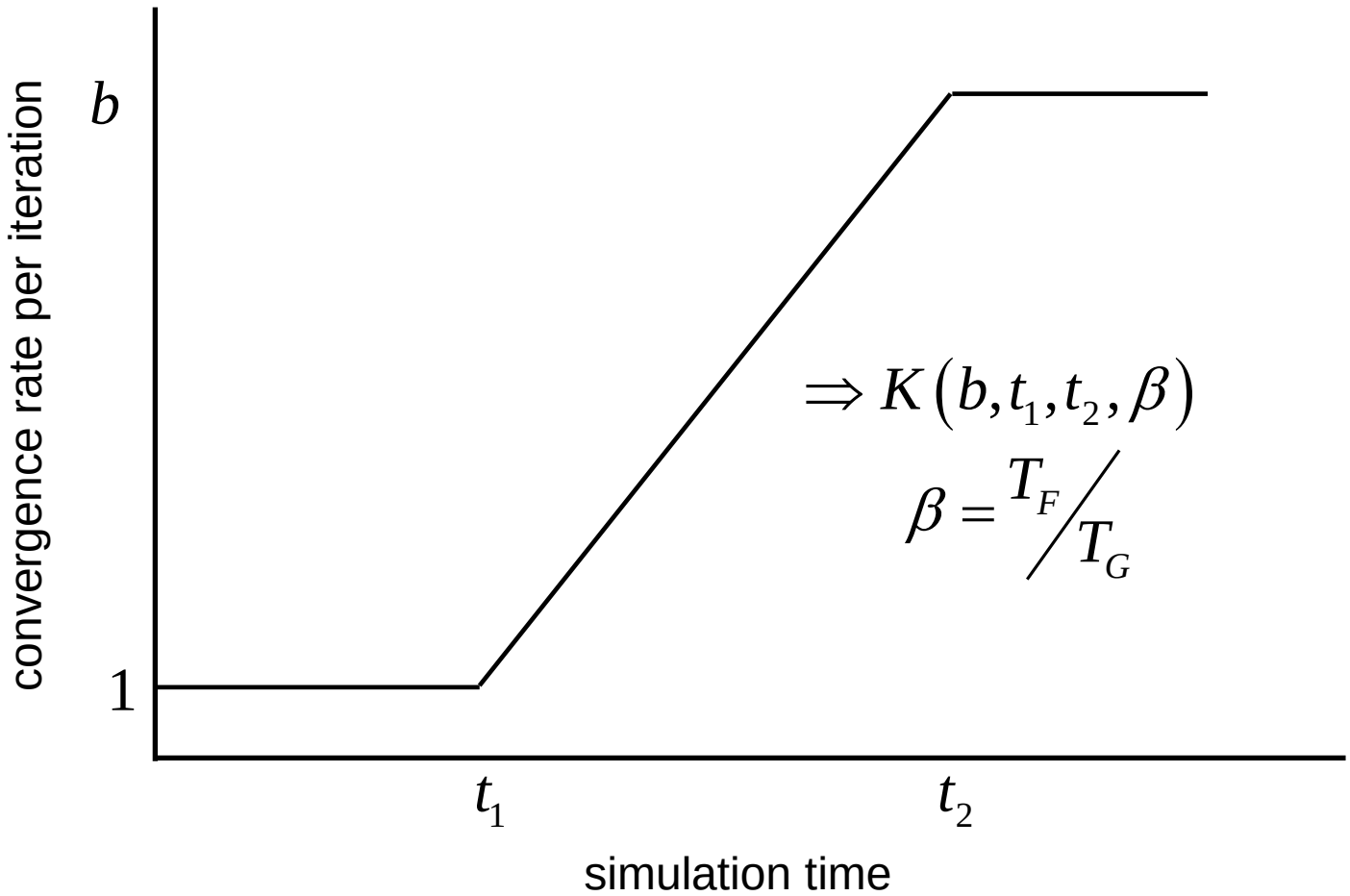
$$H_{sequential} = \frac{N}{K} \frac{1}{\left(\frac{N}{\beta}+1\right)}$$

A 2x gain in processor utilization was also realized

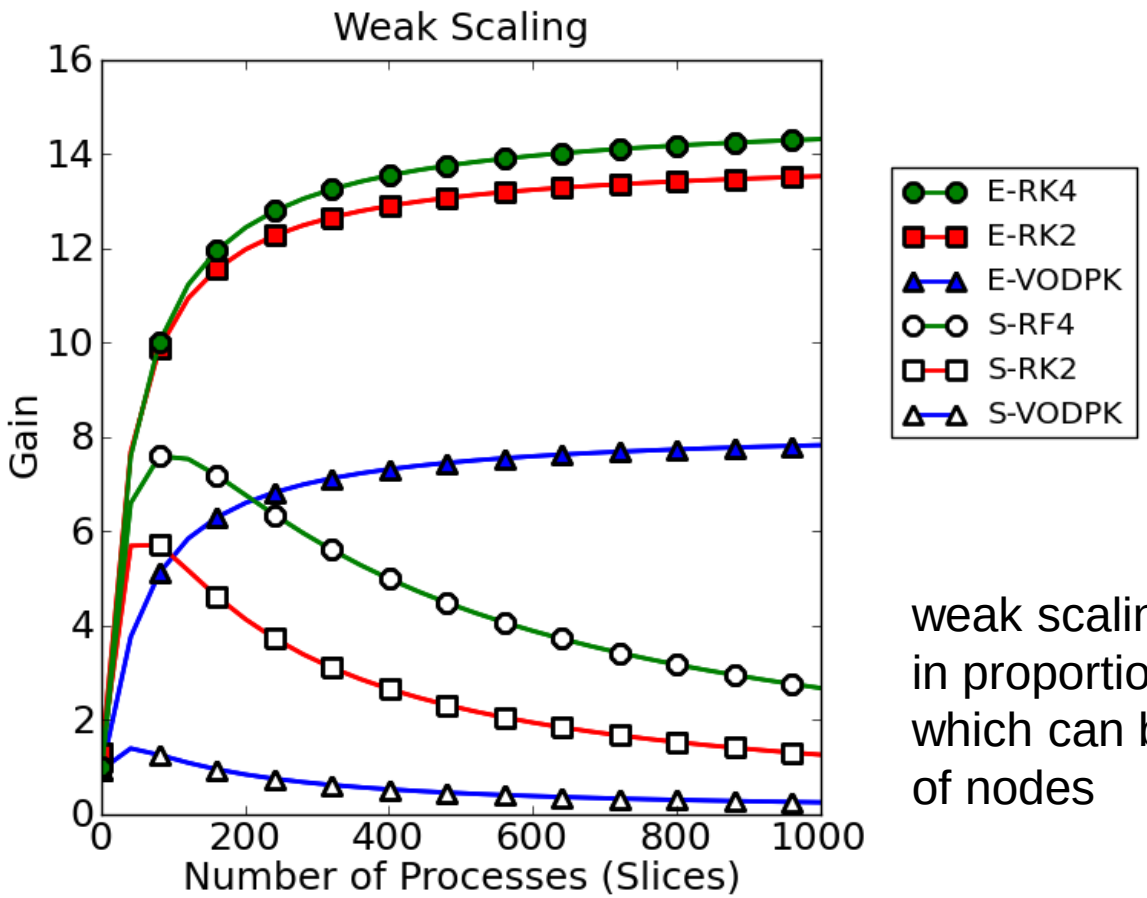




A semi quantitative empirical model for the convergence rate, $K(N)$, was developed for each coarse solver (VODPK and 2nd and 4th order Runge-Kutta) for BETA

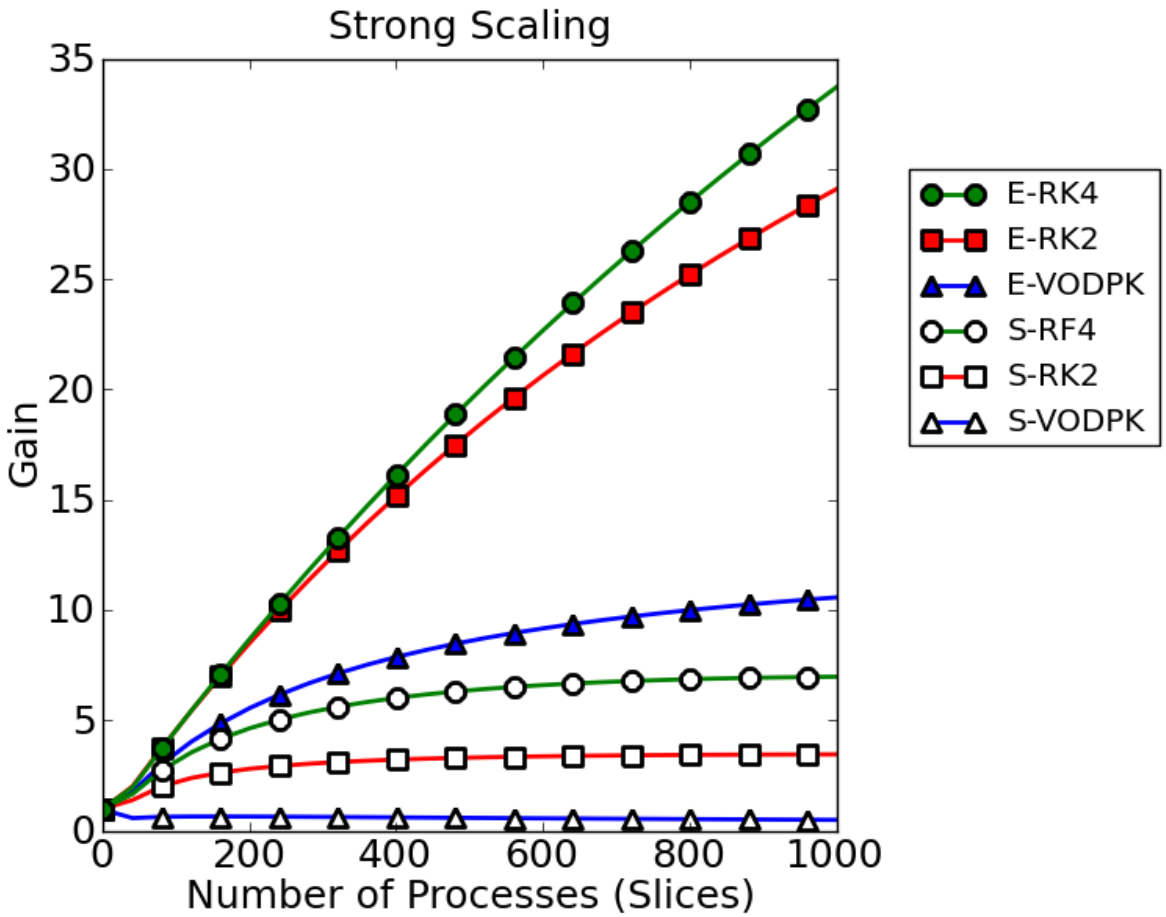


The data-driven parareal shows substantial gains even for previously ineffective coarse solvers—all coarse solvers are now 'good' for weak scaling

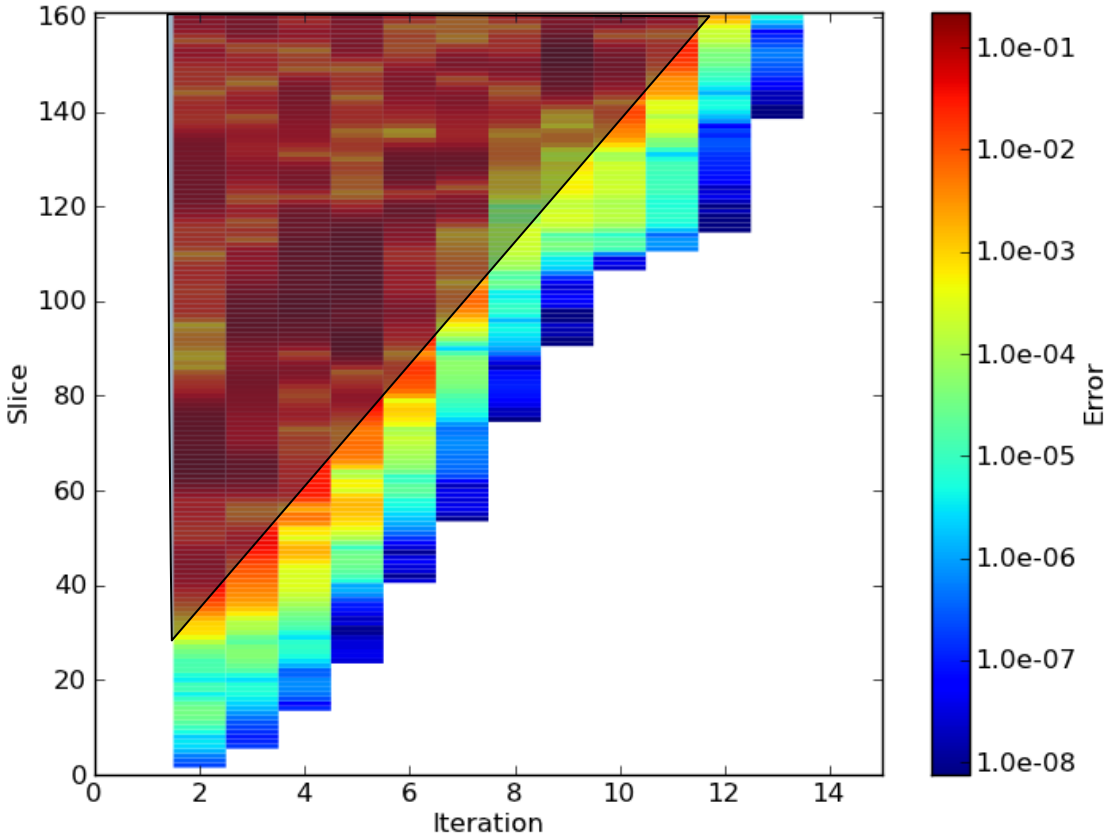


weak scaling adds work (slices) in proportion to resources (processes) which can be CPUs, nodes or groups of nodes

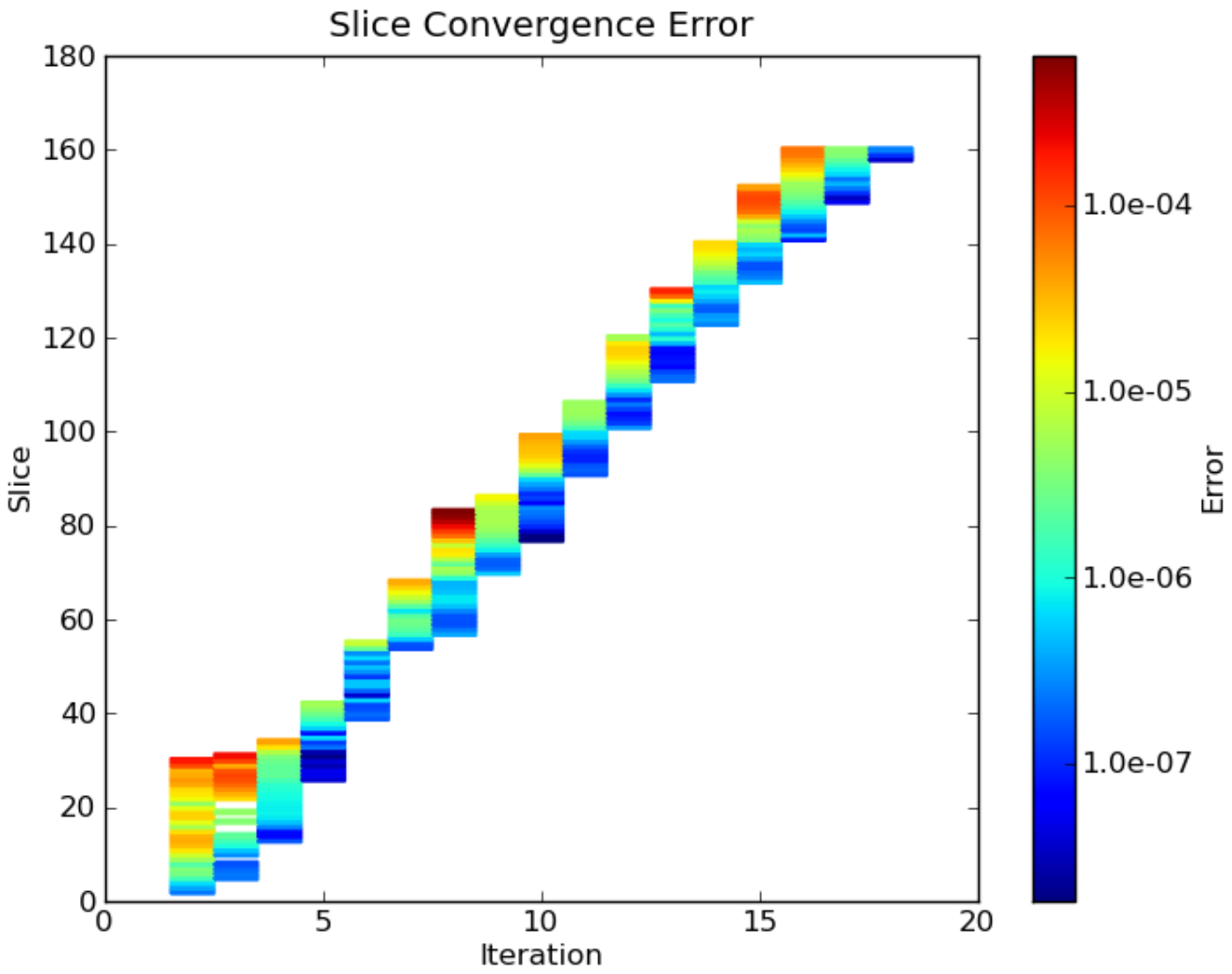
Large improvements in gain for all solvers are also expected for strong scaling



Work on slices with ~unity error is likely wasted



Greater than 5x reduction in work was obtained with 'dynamic slice addition'



Summary

- **Implementation of parareal in the IPS enables faster and more efficient application of parareal to new problems.**
- **The use of data-driven task execution using the event capability of the IPS does substantially ease the difficulty of finding an effective coarse solver for these new applications and improve processor utilization.**
- **Dynamic slice addition provides longer simulation times without increases in computational resources.**

The IPS framework, developed by computer scientists, enabled these advances

http://swim.gat.com:8080/
http://cswim.org/

Acknowledgements

Part of the research was carried out at the University of Alaska Fairbanks, funded by the DOE Office of Science Grant No. DE-FG02-04ER54741. Work at ORNL supported by the U.S. DOE under Contract DE-AC05-00OR22725 with UT-Battelle, LLC. Support was also provided by Spanish National Project No. ENE2009-12213-C03-03. The authors are grateful for grants of supercomputing resources at the University of Alaska's Arctic Region Supercomputing Center (ARSC) in Fairbanks.