Using Python to Construct a Scalable Parallel Nonlinear Wave Solver

Kyle Mandli
University of Washington
Collaborators

• Aron Ahmadia (KAUST supercomputing lab)
• Amal al-Ghamdi (KAUST)
• Lisandro Dalcin (CIMEC)
• Matthew Knepley (U. Chicago)
• Randall LeVeque (U. Washington)
• Manuel Quezada (KAUST)
• HPC^3 Participants
Clawpack

**Conservation Laws Package**

\[
\frac{\partial}{\partial t} q + \frac{\partial}{\partial x} f(q) + \frac{\partial}{\partial y} g(q) + \frac{\partial}{\partial z} h(q) = \Psi(q)
\]

- Finite volume Godunov-type method
- Authors: Randy LeVeque, Marsha Berger, et al
- Fortran77 (90)
- Multiple applications already developed

[http://www.clawpack.org](http://www.clawpack.org)
Clawpack

Finite volume Godunov-type method

Authors: Randy LeVeque, Marsha Berger, et al

Fortran77 (90)

Multiple applications already developed
Clawpack Flow

Solution at $t = t_n$

Solver

Riemann Solver

Boundary Conditions

Source Terms

Other Solver

Solution at $t = t_{n+1}$
PyClaw

• Python implementation of Clawpack
• Modular and extensible design
• Includes visualization and data tools
PyClaw

• Python implementation of Clawpack
• Modular and extensible design
• Includes visualization and data tools
PyClaw

- Python implementation of Clawpack
- Modular and extensible design
- Includes visualization and data tools
PetClaw

• Authors:
  • Amal Alghamdi, Aron Ahmadia, David Ketcheson
  • Matt Knepley (PETSc)
  • Lisandro Dalcín (mpi4py and petsc4py)

• Transparent to user

  import pyclaw

• Opaque to developer

  import petclaw as pyclaw

  300 lines of code
Full Architecture

- pyclaw
- numpy
- petsc4py
- Clawpack kernels
- f2py
- PETSc
- MPI
- Python
- Python Extension
- Fortran
- C
On-Core Efficiency

[Graph showing performance comparison between Clawpack and PetClaw on Intel Xeon and PowerPC 450 for Acoustics and Shallow Water simulations.]
Parallel Efficiency

- Efficiency
- Processes
  - Acoustics
  - Euler

- Thursday, July 14, 2011
Total Simulation Time

- Acoustics
- Euler

Processes: 1, 4, 16, 64, 256, 1024, 4096, 16384

Time (s): 0, 1000, 2000, 3000, 4000

Thursday, July 14, 2011
Walla

- Answer for dynamic loading scalability
- William Scullin (Argonne National Lab)
- Aron Ahmadia (KAUST)
Walla

- Answer for dynamic loading scalability
- William Scullin (Argonne National Lab)
- Aron Ahmadia (KAUST)
A Gross Simplification

Core
Core
Core
Core
Core
Core
Core

IO System
A Gross Simplification
A Gross Simplification

IO System
A Gross Simplification

Core
Core
Core
Core
Core
Core
Core
Core
Core
Core

Python
IO System
Walla’s Solution

Core → Core → Core → Core → Core → Core → Core

Core → Core → Core → Core → Core → Core → Core

Core → Core → Core → Core → Core → Core → Core

Python

IO System

Thursday, July 14, 2011
Walla \textit{import} Timings

\begin{itemize}
  \item Time (s)
  \item Processors
\end{itemize}
Conclusions
Conclusions

pyclaw

numpy

petsc4py

Clawpack kernels

f2py

Python

Python Extension

Fortran

PETSc

MPI

C
Conclusions

- pyclaw
- numpy
- petsc4py
- Clawpack kernels
- f2py
- Python
- Python Extension
- Fortran
- PETSc
- MPI
- C
Future

• New versions being released
• Moving to github!