Arches Part 1: Introduction to the Uintah Computational Framework

Charles Reid
Scientific Computing Summer Workshop
July 14, 2010
What is a computational framework?

✶ Common code
✶ Generic functionality
✶ Reusable abstractions
✶ Wrapped through API (Application Programming Interface)
✶ Abstracts low-level coding details
✶ Similar to libraries
What is the UCF?

- Computational framework
- Designed for solving time-dependent PDEs
- Object-oriented C++
- Parallelizes to thousands of processors
UCF Design

- Common component architecture (CCA)
- Components have common features
- Components use common interface to framework
- Components solve time-dependent PDEs on a structured Cartesian mesh
Standalone Uintah Simulation (sus)

- Framework needs a driver - something with a main() function
- “sus” is the main executable
- Location:
  uintah/src/StandAlone/sus.cc
Framework needs a driver - something with a main() function.

"sus" is the main executable.

Input File
Format: XML
Extension: .ups

Component
e.g. Arches

Driver Program
"sus"

Scheduler
MPI Manager
etc...

Saved Simulation Data
What does the framework provide?

✴ (Cartesian) Grids, patches, cells
✴ Levels - for automatic mesh refinement (AMR)
✴ Variables (cell-centered, face-centered) & stencils
✴ Variable labels
✴ Data warehouse (abstraction for data storage and retrieval)
✴ Tasks (computations)
✴ Scheduler (organizes the tasks)
✴ MPI Manager
What does the framework provide?

- (Cartesian) Grids, patches, cells
- Levels - for automatic mesh refinement (AMR)
- Variables (cell-centered, face-centered) & stencils
- Variable labels
- Data warehouse (abstraction for data storage and retrieval)
- Tasks (computations)
- Scheduler (organizes the tasks)
- MPI Manager

Domain/mesh abstractions
What does the framework provide?

✴ (Cartesian) Grids, patches, cells
✴ Levels - for automatic mesh refinement (AMR)
✴ Variables (cell-centered, face-centered) & stencils
✴ Variable labels
✴ Data warehouse (abstraction for data storage and retrieval)
✴ Tasks (computations)
✴ Scheduler (organizes the tasks)
✴ MPI Manager

Data abstractions
What does the framework provide?

- (Cartesian) Grids, patches, cells
- Levels - for automatic mesh refinement (AMR)
- Variables (cell-centered, face-centered) & stencils
- Variable labels
- Data warehouse (abstraction for data storage and retrieval)

Parallelization abstractions

- Tasks (computations)
- Scheduler (organizes the tasks)
- MPI Manager
Domain & Mesh Abstractions

- Input file: user specifies domain size and number of cells
- Patch - a chunk of the grid
- Each patch is “owned” by a processor

# patches = # processors
Data Abstractions

✴ Data warehouse as a library
✴ Framework as librarian
✴ Simulation data as the books
✴ VarLabels as book call numbers: unique identifiers
Data Abstractions

* To get a book:
  Have to check out

* To return a book:
  Have to check in

* Can’t check in books that aren’t registered with library (no call number)
Parallelization Abstraction

* Each component is split into tasks
* Any given task will have:
  * things it requires
  * things it calculates
* If each task specifies what it requires and what it calculates, the workload can be balanced *(BEFORE the work is done)*
\[
\frac{\partial \rho_i}{\partial t} + \nabla \cdot (\rho_i v) = -\nabla \cdot j_i
\]

\[j = \rho D \nabla \cdot \omega\]
Tasks and the Taskgraph

- Each task run on each processor
- But, each task run on different portions of the data (or grid)
- Each task has two data warehouses: OldDW and NewDW
Examples

* Located in:
  uintah/src/CCA/Components/Examples

* Poisson example walkthrough
  uintah/src/CCA/Components/Examples/
  Poisson1.cc,Poisson1.h

* Important functions to focus on:
  Poisson1::scheduleTimeAdvance
  Poisson1::timeAdvance
**Poisson1::scheduleTimeAdvance()**

- Specifies that task *requires* phi
- Specifies that task *computes* phi
- Specifies that task *computes* a residual

**Poisson1::timeAdvance()**

- Call to GET phi from old data warehouse
- Call to ALLOCATE AND PUT phi in new data warehouse
- Call to PUT residual in new data warehouse
**Poisson1::scheduleTimeAdvance()**

- Specifies that task *requires* phi
- Specifies that task *computes* phi
- Specifies that task *computes* a residual

**Poisson1::timeAdvance()**

- Call to GET phi from old data warehouse
- Call to ALLOCATE AND PUT phi in new data warehouse
- Call to PUT residual in new data warehouse
Poisson1::scheduleTimeAdvance()
  Specifies that task \textit{requires} \phi
  Specifies that task \textit{computes} \phi
  Specifies that task \textit{computes} a residual

Poisson1::timeAdvance()
  Call to \textit{GET} \phi from old data warehouse
  Call to \textit{ALLOCATE AND PUT} \phi in new data warehouse
  Call to \textit{PUT} residual in new data warehouse
Examples

**Located in:**

* uintah/src/CCA/Components/Examples

**Poisson1 example walkthrough**

**Poisson1::scheduleTimeAdvance()**

- Specifies that task *requires* phi
- Specifies that task *computes* phi
- Specifies that task *computes* a residual

**Poisson1::timeAdvance()**

- Call to GET phi from old data warehouse
- Call to ALLOCATE AND PUT phi in new data warehouse
- Call to PUT residual in new data warehouse

**Poisson1::timeAdvance**
Example Component: Poisson Solver

Poisson example input file:

```
uintah/src/StandAlone/inputs/Examples/poisson1.ups
```

To run the Poisson example:

```
/path/to/sus /path/to/inputs/poisson1.ups
```
Other Examples

- **Poisson2:** Demonstrates use of sub-schedulers
  Non-linear solver iterations within timestep

- **HeatEquation:**
  Uses a Poisson solver to solve heat equation

- **SolverTest1:**
  Interfaces with generic SolverInterface
  This can then create linear solvers for stencil matrices
Other Examples

✴ Poisson1: Demonstrates use of sub-schedulers Non-linear solver iterations within timestep

✴ HeatEquation: Poisson solver to solve heat equation

✴ SolverTest1: Interfaces with generic SolverInterface. This can then create linear solvers for stencil matrices

\[
A = \begin{bmatrix}
-2 & 1 & 1 & 1 & 1 \\
1 & -2 & 1 & 1 & 1 \\
1 & -2 & 1 & 1 & 1 \\
1 & -2 & 1 & 1 & 1 \\
1 & -2 & 1 & 1 & 1
\end{bmatrix}
\]
Other Examples

✴ Poisson1: Demonstrates use of sub-schedulers
Non-linear solver iterations within timestep

✴ HeatEquation: Poisson solver to solve heat equation

✴ SolverTest1: Interfaces with generic SolverInterface
This can then create linear solvers for
stencil matrices

\[
A = \begin{bmatrix}
-2 & 1 \\
1 & -2 & 1 \\
1 & -2 & 1 \\
1 & -2
\end{bmatrix}
\]

Interface: uintah/src/CCA/Ports/SolverInterface.*

Solvers: uintah/src/CCA/Components/Solvers/*
Other Examples

*Burgers:*
Solves the Burgers equation using Forward Euler

\[ \frac{\partial u}{\partial t} = -u \nabla \cdot u \]
Where to go for more info

✴ User’s Guide/Installation Guide
uintah/doc/

✴ Contains general information applicable to all Uintah users

✴ Uintah Wiki:
http://www.uintah.utah.edu/trac

✴ Contains non-general information

✴ Configure lines for specific clusters
✴ Machine-specific scripts/compiling info
Where to go for more info

[*] Uintah Doxygen:
   http://www.uintah.utah.edu/trac/doxygen/

[*] Homebrew email list - Uintah developers
   csafe-homebrew@cs.utah.edu

[*] If you can’t find what you’re looking for in the documentation... Tell John Schmidt!
   John.Schmidt@utah.edu
   (He’s the guy in charge of documentation)
A Final Word:
How you can contribute
A Final Word:
How you can contribute

✴ The Uintah documentation SUCKS
✴ Most developers don’t care enough to contribute
✴ It’s up to users to improve documentation and make it better
✴ Don’t be afraid to contribute to the documentation!
✴ Add Doxygen comments to classes you “figure out”, add to the User Guide and Installation Guide, add to the wiki, etc... Don’t be shy!
Arches Part 2: Introduction to Arches

Charles Reid
Scientific Computing Summer Workshop
July 14, 2010
Where is Arches?

📌 From the Uintah tree:
src/CCA/Components/Arches/
What is Arches?

✴ Finite-volume LES code
✴ Explicit time integration
✴ Tabulated chemistry (i.e. table lookup)
✴ Pressure projection method
What is Arches?

- Finite-volume LES code
- Explicit time integration
- Tabulated chemistry (i.e. table lookup)
- Pressure projection method
Capabilities: Subgrid Models

- Smagorinsky (SmagorinskyModel.*)
- Incompressible dynamic model (IncDynamicProcedure.*)
- Variable density dynamic model (CompDynamicProcedure.*)
- Local dynamic model (CompLocalDynamicProcedure.*)
Capabilities: Subgrid Models

- Smagorinsky
  (SmagorinskyModel. *)
- Incompressible dynamic
  (IncDynamicProcedure.*)
- Variable density dynamic model
  (CompDynamicProcedure.*)
- Local dynamic model
  (CompLocalDynamicProcedure.*)

All are child classes of TurbulenceModel class
Capabilities: Combustion Chemistry

✴ Arches uses table lookups
✴ Tabulated combustion chemistry tables
✴ Reaction models vs. Mixing models:
Capabilities: Combustion Chemistry

- Arches uses table lookups
- Tabulated combustion chemistry tables
- Reaction models vs. Mixing models:

\[
\phi = \phi(\eta)
\]

\(\phi = \text{Full thermochemical state}\)
\(\eta = \text{Reduced set of parameters}\)
Capabilities: Combustion Chemistry

✴ Arches uses table lookups
✴ Tabulated combustion chemistry tables
✴ Reaction models vs. Mixing models:

Mixing Models:

\[ \tilde{\phi} = \int \ldots \int_{\eta'} \phi(\eta') \hat{P}(\eta') \, d\eta' \]
Capabilities: DQMOM

★ Novel multiphase application of Arches
★ Applying Arches to coal combustion and gasification
★ Models for two-phase drag, heat transfer, reactions, &c.
Capabilities: Transport Equation Framework

- Handle arbitrary numbers of transport equations
- Simple to add via input file
- Flexible: much more flexible than existing Arches transport equations
- Can apply arbitrary discretization schemes (currently superbee and upwind), arbitrary time integrators (currently 1st/2nd/3rd order RK)
How it all fits together

✴ Remember the Poisson component?
✴ The most important method was timeAdvance()
✴ Arches is very similar
How it all fits together

✴ Remember the Poisson component?
✴ The most important method was timeAdvance()
✴ Arches is very similar

Arches Component

Arches::timeAdvance()

Nonlinear Solver

NonlinearSolver::*
ExplicitSolver.*
PicardNonlinearSolver.*

NonlinearSolver::nonlinearSolve()

Mixture Fraction Transport Equation
ScalarSolver.*

Enthalpy Transport Equation
EnthalpySolver.*

Momentum Transport Equation
MomentumSolver.*

Pressure
PressureSolver.*

Radiation
RadiationSolver.*
DORadiationModel.*
Radiation/

Chemistry Table
Properties.*
NewStaticMixingTable.*
ColdFlowMixingModel.*
TabPropsInterface.*
Mixing/
ChemMix/

Boundary Conditions
BoundaryCondition.*
BoundaryCond_new.*

Transport Equations
ScalarEqn.*
EqnFactory.*
Discretization_new.*
ExplicitTimeInt.*

DQMOM
DQMOM.*
DQMOMEqn.*
DQMOMEqnFactory.*
CoalModelFactory.*
CoalModels/
Questions so far about the code side of Arches?
Algorithm
Algorithm

\[
\text{scalars } S = \{ f, h, w_\alpha, \langle \xi_i \rangle_\alpha, \ldots \} \\
\text{state space vector } \phi = \{ \text{state space variables} \} \\
\psi = \{ s, \phi, \ldots \} \\
t = 0 \\
\text{for time } n = 0 \ldots N \\
\quad \text{define } t^{n+1} = t^n + \Delta t \\
\quad \text{for } RK_{step} \text{ } k = 1 \ldots K \\
\quad \quad \text{for scalars } s = 1 \ldots S \\
\quad \quad \quad (\bar{\rho} \bar{s})^{k+1} = (\bar{\rho} \bar{s})^k + \Delta t^k \left( \text{RHS} \right)_s^k \\
\quad \quad \text{end for} \\
\quad \text{Compute } \bar{\rho}^* \text{ from equation (37)} \\
\quad \text{if } \bar{\rho}^* \text{ non-realizable} \\
\quad \quad \bar{\rho}^* = \bar{\rho}^k \\
\quad \text{end if} \\
\quad \text{for scalars } s = 1 \ldots S \\
\quad \quad \bar{s}^{k+1} = \frac{(\bar{\rho} \bar{s})^{k+1}}{\bar{\rho}^*} \\
\quad \text{end for} \\
\quad \begin{cases} 
\text{Table lookup} & \bar{\phi}^{k+1} = f(\bar{s}^{k+1}) \\
\text{Filter} & \nu_T \\
\text{RTE} \\
\text{DQMOM} & Ax = B \\
\end{cases} \\
\text{Compute unprojected velocities } u^* \\
\text{RK}_{step} \text{ averaging } \psi^k = \alpha_k \psi^k + \beta_k \psi^{k-1} \\
\text{Pressure projection } \nabla^2 p = \text{div} (\bar{\rho} u^*) \\
\text{Compute projected velocities } \bar{u}^{k+1} = \bar{u}^k + \Delta t^k \nabla p \\
\text{Calculate } u_p \text{ (equilibrium Eulerian approach): } \frac{u_g - u_p}{u_g} = \text{RHS} \\
\text{end for} \\
\text{end for}
Questions about the Arches algorithm?
Wrap-up