Alpaca

Erik Schnetter
Baton Rouge, February 2008
Alpaca


- NSF SDCI, $590k, 3yr (2007-2010)

- 1 postdoc, 1 student

- Connected to Blue Waters (Eclipse interface for Cactus)
Research

1. High level debugging tools for HPC applications
   (debugging the physics, not the code)

2. High level profiling tools for HPC applications
   (profiling the application as a whole)

3. Hardware fault tolerance

4. User interface for the above
   (for physicists, not programmers)
Ail
(the drops that made my barrel overflow)

• Physicist at AEI debugged problem by making movies of simulations

• Inter-thorn dependencies with AMR are horribly complicated (data flow, call tree)

• printf is a fine debugger

• rdtsc is a fine profiler
Main Idea

- A software framework has detailed knowledge about the state of the application
- Debugger, profiler should interact with the framework, instead of only inspecting the application from the outside
- Need to involve end user in this, since debugging and profiling are holistic tasks
Plans: Debugging

- Use VisIt (LLNL) as (remote) visualisation tool
- VisIt’s “libsim” supports user interaction with running simulations
- Expand existing web server debugging UI
- Add fine-grained schedule control
- Fancy ideas: modify variables, modify code, go back in time, ...
Plans: Profiling

• Existing tools: Cactus timers, FFTW clocks, TAU; health indicators: M/h, Flop/s, gpu/s

• Problem: output format, display to user (too much data)

• Incorporate framework’s knowledge on domain topology

• Problem: teach users about AMR algorithms, about prejudices, about looking outside the circle of light
Plans: User Interface

• Plan A: Extend current web server
• Plan B: Build UI using VisIt’s libsim (QT)
• Plan C: Use Eclipse (PTP, TPTP)

• Note: HPC platforms are fungible
• Note: People have laptops these days

• Need seamless transition between production - debugging - profiling
Almost-Achievements

• Updates to web server
• Fine grained schedule control
• Remote, online visualisation with VisIt
• Lightweight timers (FFTW)
• New model for checking schedule correctness
• Incorporate meta-data from automatically generated code
Next Steps

• Implement remote UI, remote visualisation (basis for other work)

• Fine grained step size control for debugging (to attract early users)

• Define efficient output format for high-resolution low-level timers (necessary for XiRel)