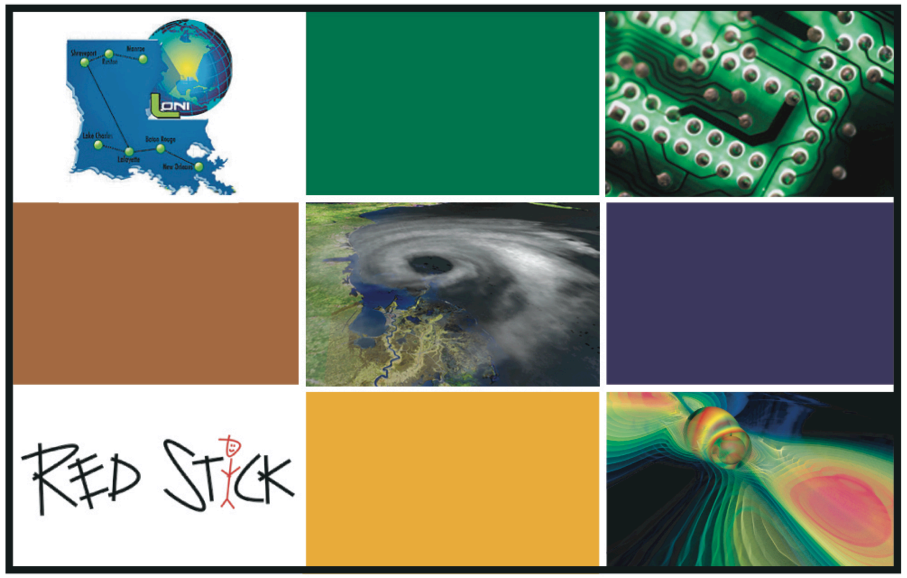
Cactus Concepts for Distributed HPC Applications

Erik Schnetter, Gabrielle Allen, Jian Tao

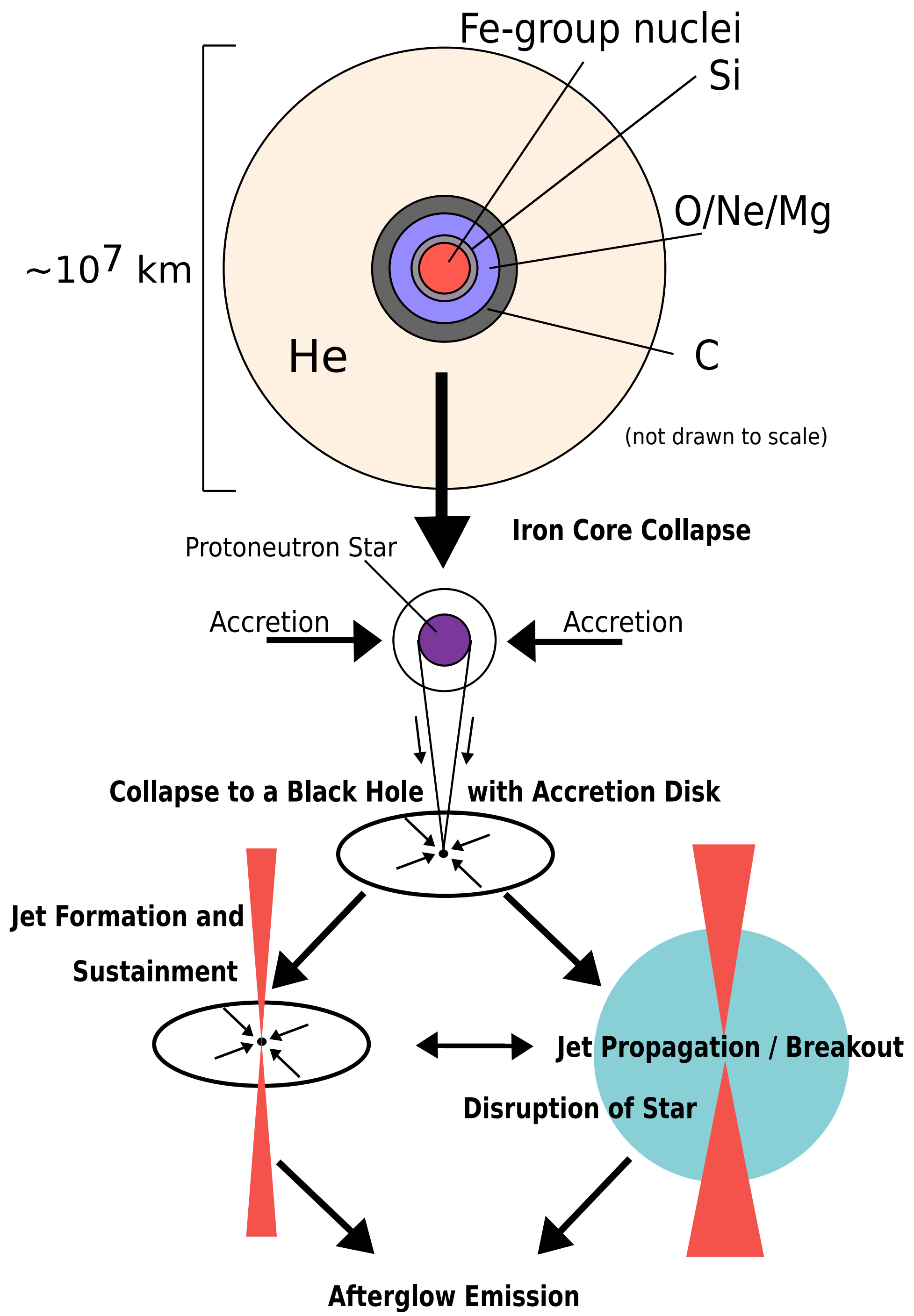
Baton Rouge, January 2008

LSU

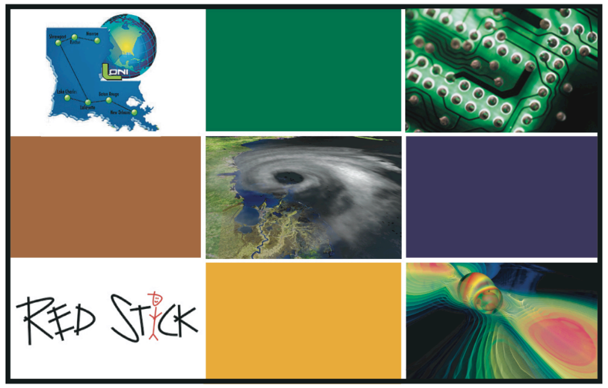
CENTER FOR COMPUTATION
& TECHNOLOGY



Gamma Ray Bursts: Science Driver Problem



- Most energetic events known in universe
- Grand challenge in astrophysics; likely to be detected by LIGO in coming years
- Combines many fields of physics
- Requires (at least) petascale computing for modelling



Cactus

- Framework for (tightly coupled) HPC: supports code development, simulation control, analysis, visualisation
- Manage increased complexity with high level abstractions, e.g. for inter-node communication, intra-node parallelisation
- Active user community, 10+ years old
- Supports collaborative development

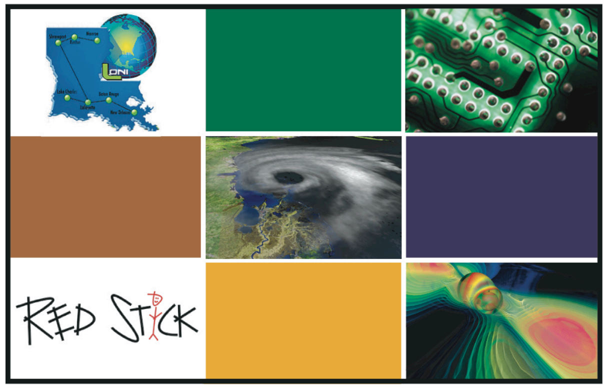
LSU
CENTER FOR COMPUTATION
& TECHNOLOGY

Cactus in Astrophysics

- Three layers of abstraction in a typical code:
- *Top*: specific physics codes, developed by single research groups
- *Middle*: numerical relativity toolkit, developed by community
- *Bottom*: computational infrastructure, developed by computer scientists

Cactus

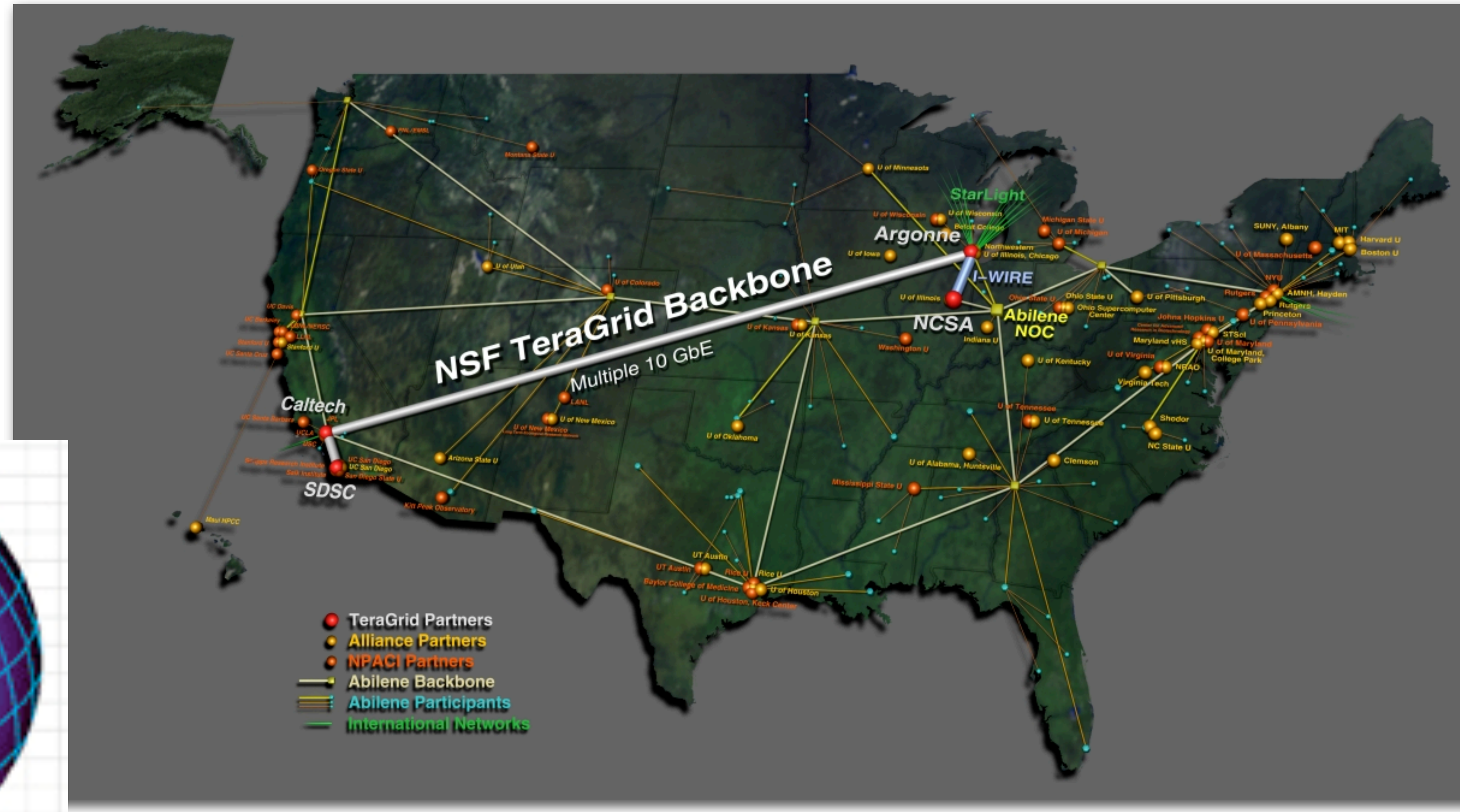
Physics code(s)
Einstein Toolkit
Computational Toolkit



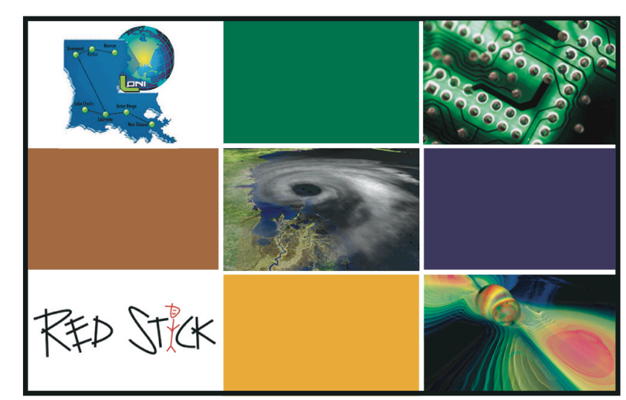
CENTER FOR COMPUTATION
& TECHNOLOGY

TeraGrid, LONI, LSU, ...

Also: NERSC,
Germany, ...



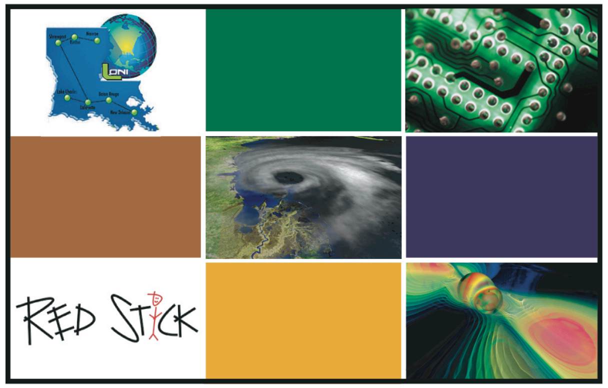
Many machines,
all subtly different



LSU
CENTER FOR COMPUTATION
& TECHNOLOGY

Fungible Computing

- Too many machines: need to use them as exchangeable tools, not as unique systems
- TeraGrid Software Stack – excellent idea, but not (yet) successful
- We are building domain-specific abstractions around the HPC machines we use; *need to generalise this*



CENTER FOR COMPUTATION
& TECHNOLOGY

BBH Factory:
HPC front-end for numerical relativity

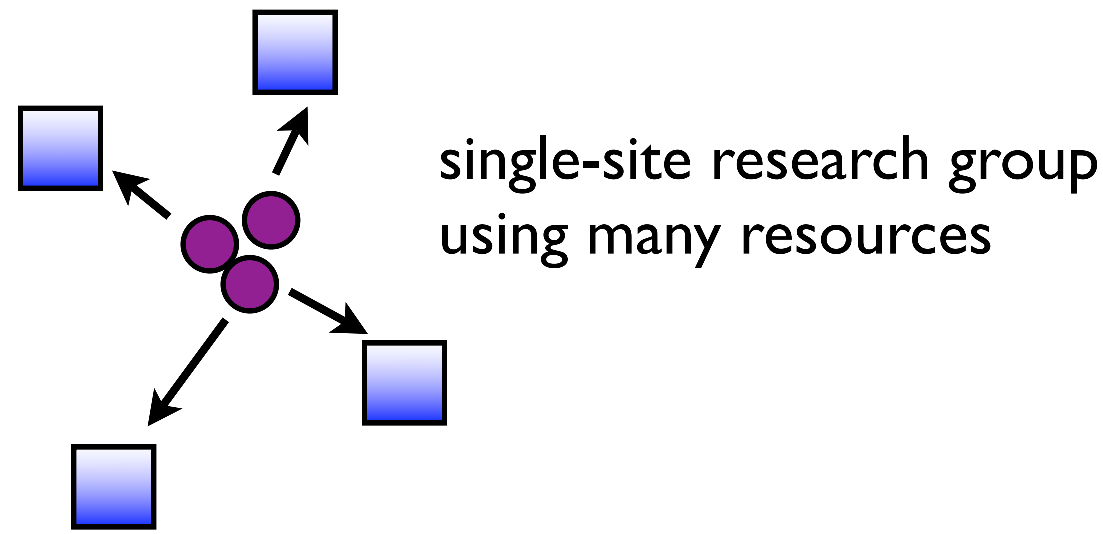
- Contains information on: remote access, file system layout, configuring and building, installed software, job submission methods
- Not really domain specific – but application specific and research group specific
- *Works great,*
  but is built on simple tools (e.g. ssh), doesn't scale beyond single group

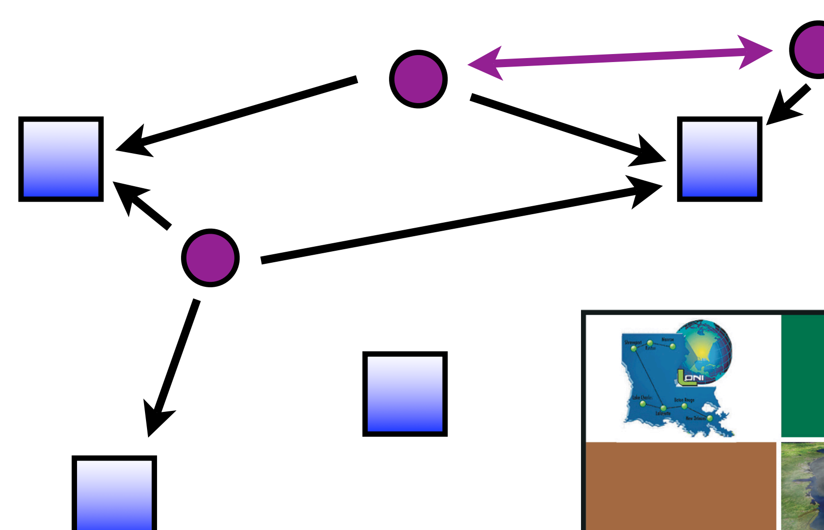
LSU

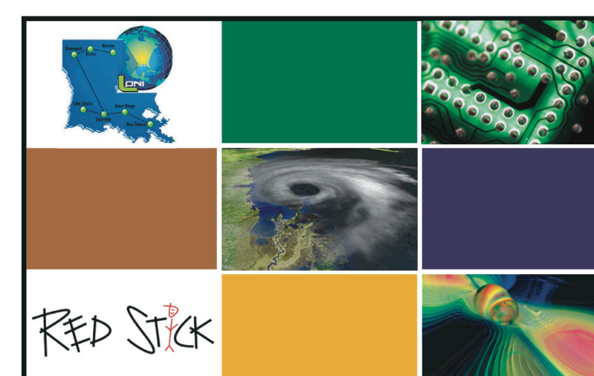
CENTER FOR COMPUTATION
& TECHNOLOGY

Fungible Places

- "Places change, people remain the same"
- Cactus supports a truly distributed code development model
- Code components are *both developed and stored separately*, and are only integrated by the end user
- Numerical relativity groups are "competitive"

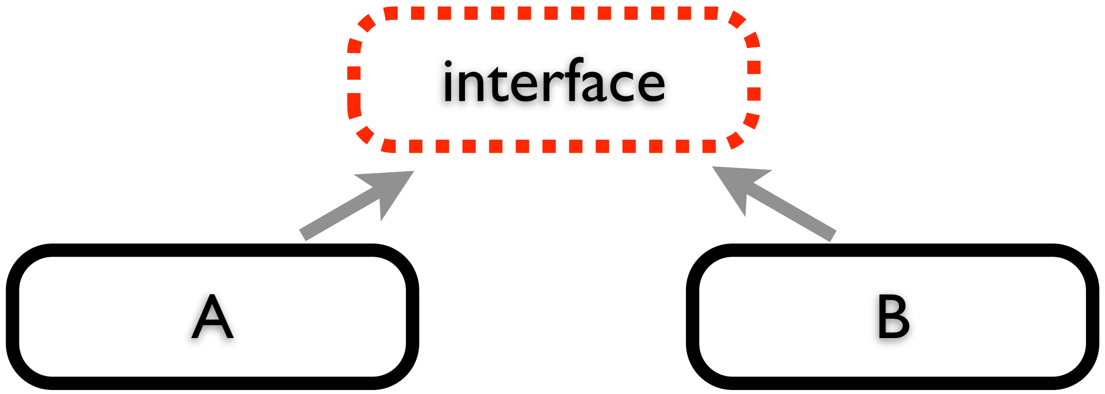






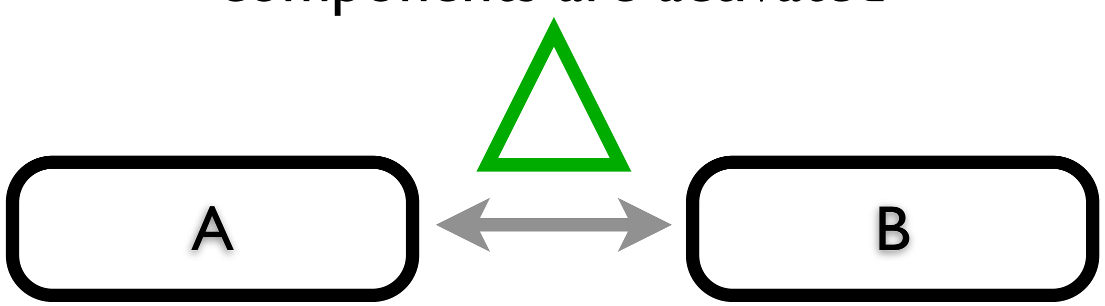
LSU
CENTER FOR COMPUTATION
& TECHNOLOGY

Distributed Code Development



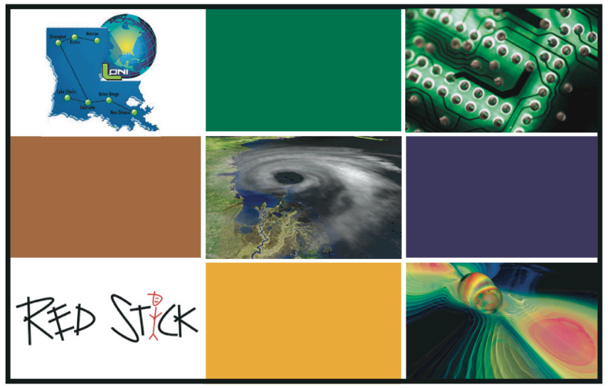
standard object-oriented

type checking when
components are activated



ad-hoc interfaces

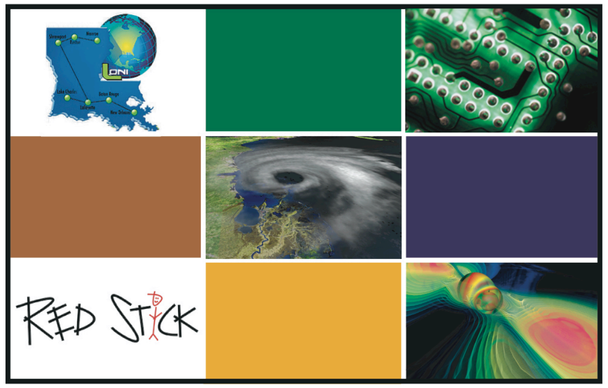
- Mechanism: *ad-hoc interfaces* (Bazaar, no Cathedral)
- Each component describes its interface – there is no abstract base class, *no central authority*
- Only most important interfaces are designed by community



LSU
CENTER FOR COMPUTATION
& TECHNOLOGY

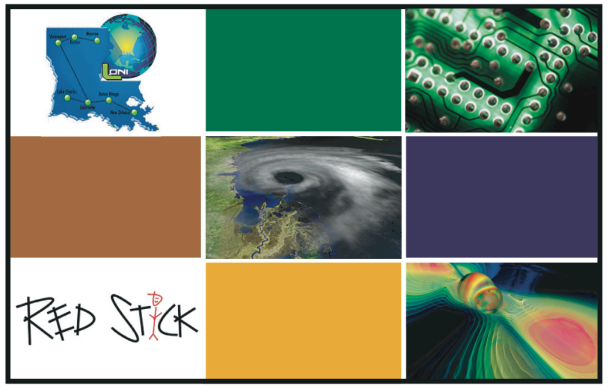
What's Next?

- Above mechanisms are used in production, 24/7 – need to be *reliable*, hence are *boring*
- Other, more exciting Cactus features have been prototyped and demonstrated (see below)
- Not always easy to begin use these in production: need reliability, ubiquity, user buy-in, help desk support



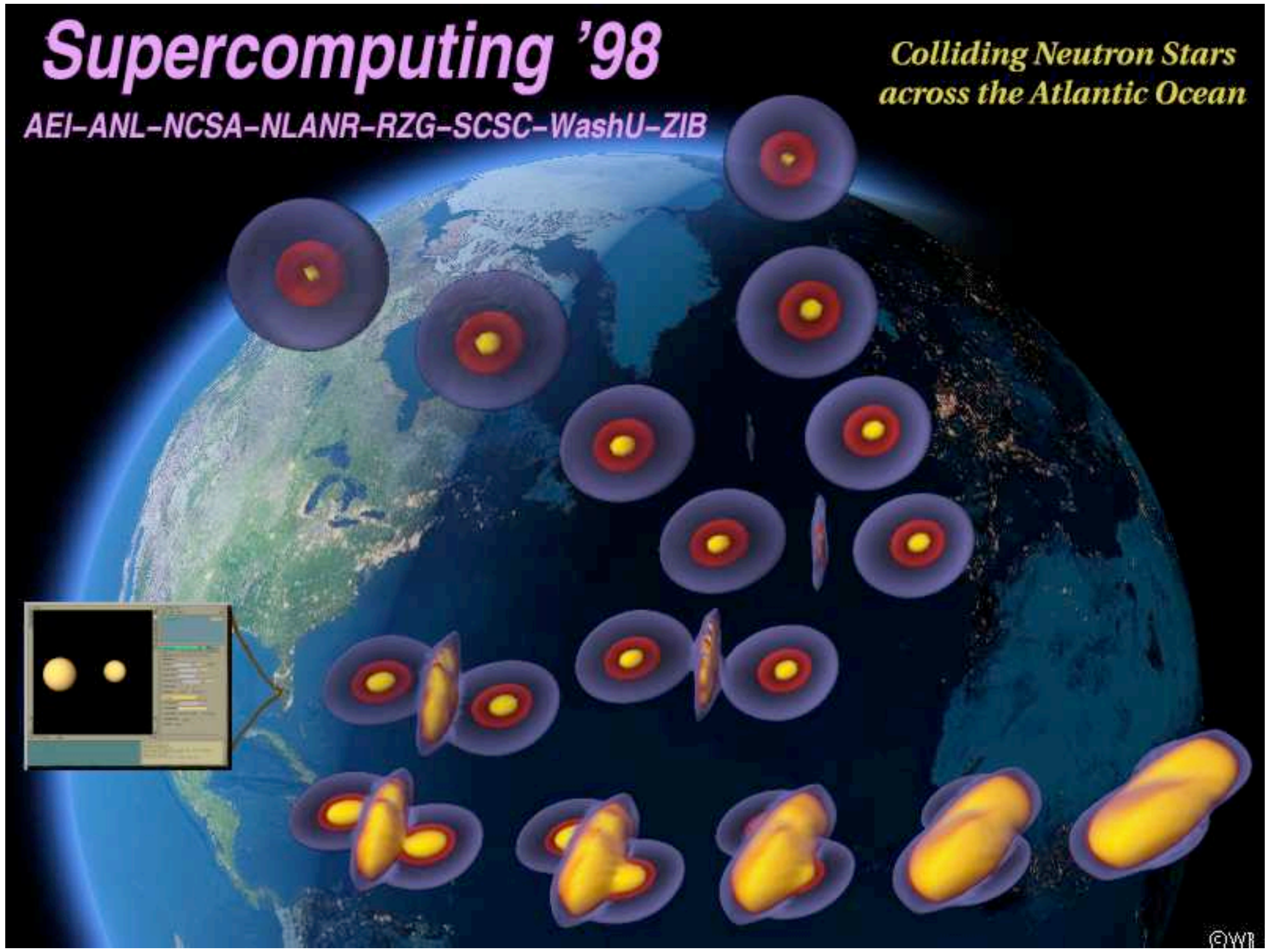
Cactus Framework

- Framework controls execution and manages data
- Components declare what data they access (interface.ccl)
- Components declare which functions they provide (schedule.ccl)
- Components should be *functional*, i.e., keep no state information
- Thus: Framework has *complete state information*
- Allows: Checkpointing, correctness checks, metadata collection, and much more...

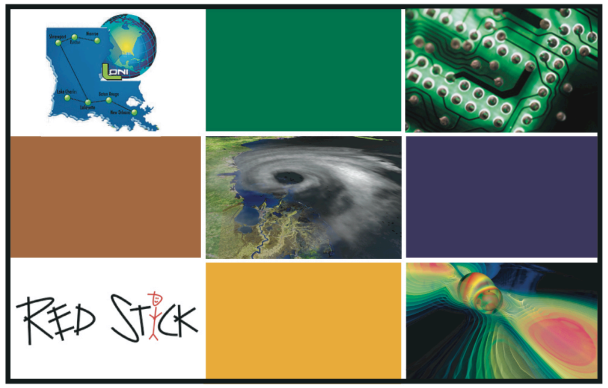


Multi-Machine Simulations

- LONI: many mid-size machines, fast network: ideal environment to combine compute power
- Using HARC for co-scheduling, Globus for job start and communication
- Can optimise AMR and communication algorithms for heterogeneous networks, since physics and AMR are separated

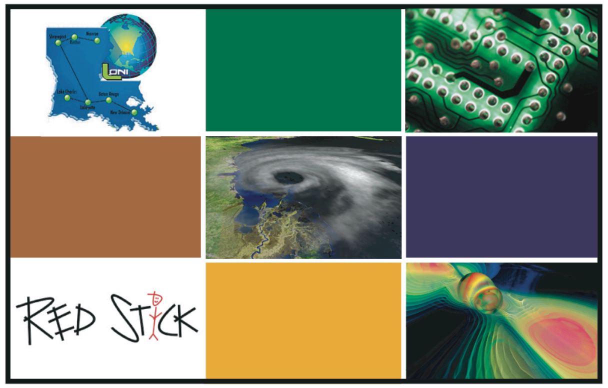


Using three T3E's in Garching (Germany), Berlin (Germany), and SDSC (USA)



Task Spawning, Job Migration

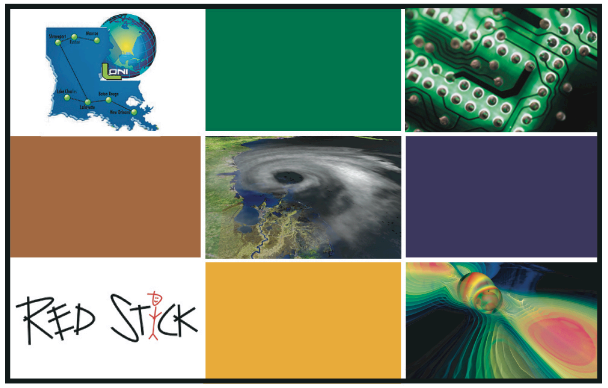
- As framework, Cactus has complete information about state of the simulation
- Components can query framework, then spawn post-processing jobs, or migrate whole simulation
- Cactus won the High-Performance Computing Challenge Award (SC2002) for a task farming application written with Cactus, which deployed Cactus Black Hole simulations across 70 diverse machines in 12 different countries



LSU
CENTER FOR COMPUTATION
& TECHNOLOGY

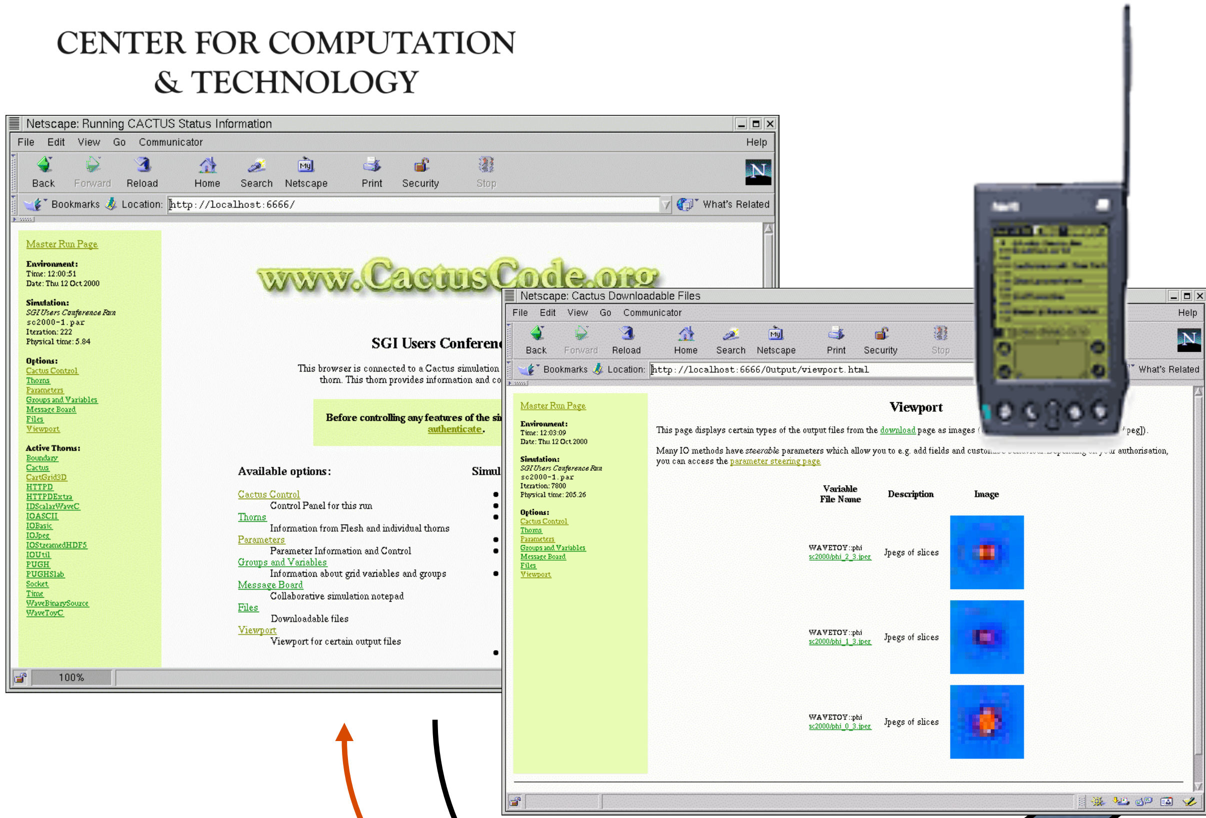
Automated Metadata Collection

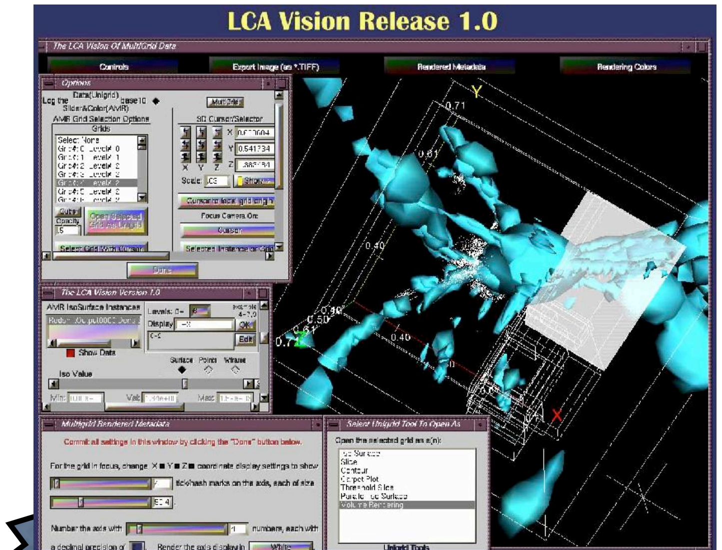
- Need to preserve metadata about simulations for many reasons, e.g. scientific integrity
- Framework can collect metadata automatically: component *Formaline* saves parameters/events to file, announces them to database
- User does not need to explicitly pass metadata
- Can collect *more data than envisioned by the user*: allows data mining



Remote Visualisation/Steering

CENTER FOR COMPUTATION & TECHNOLOGY





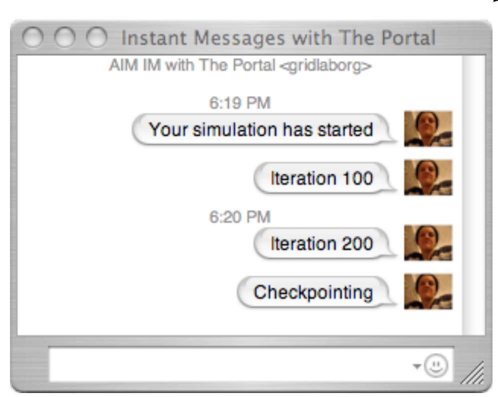
Any visualisation client:

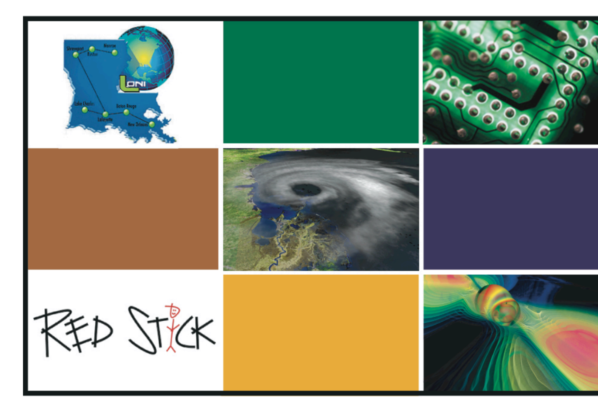
Amira, OpenDX, VisIt

Changing steerable parameters

- Parameters
- Physics, algorithms
- Performance

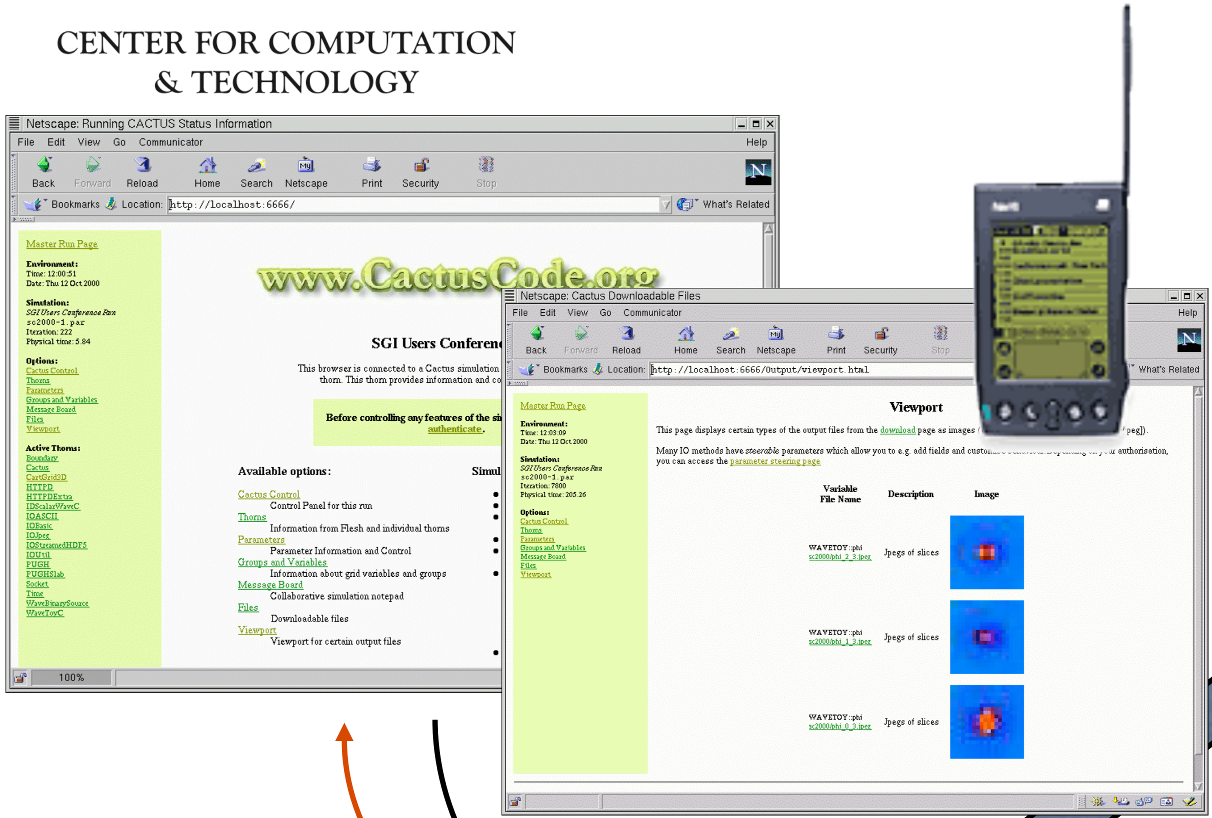
Streaming HDF5
auto-downsample

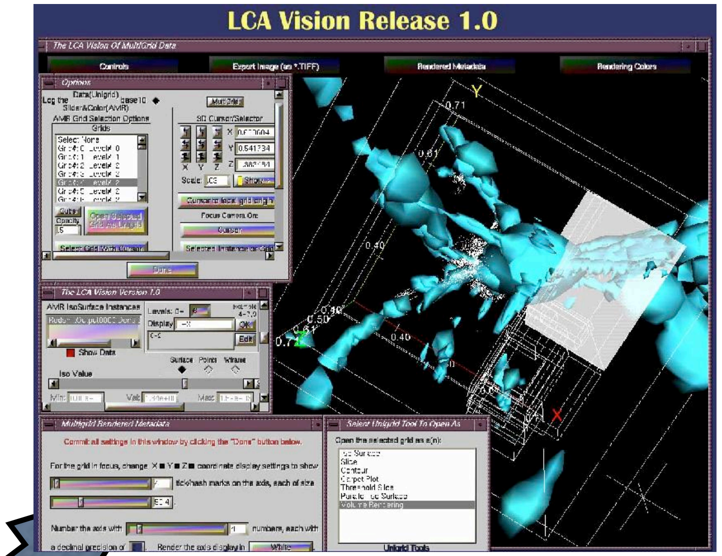




CENTER FOR COMPUTATION & TECHNOLOGY

Remote Visualisation/Steering





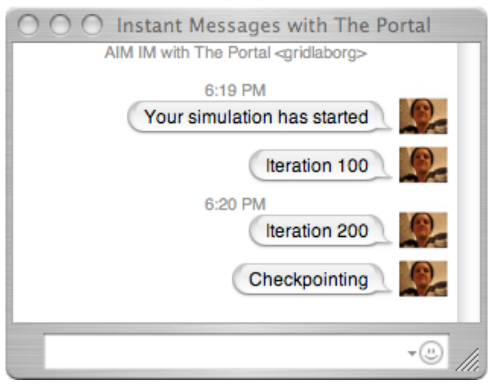
Any visualisation client:

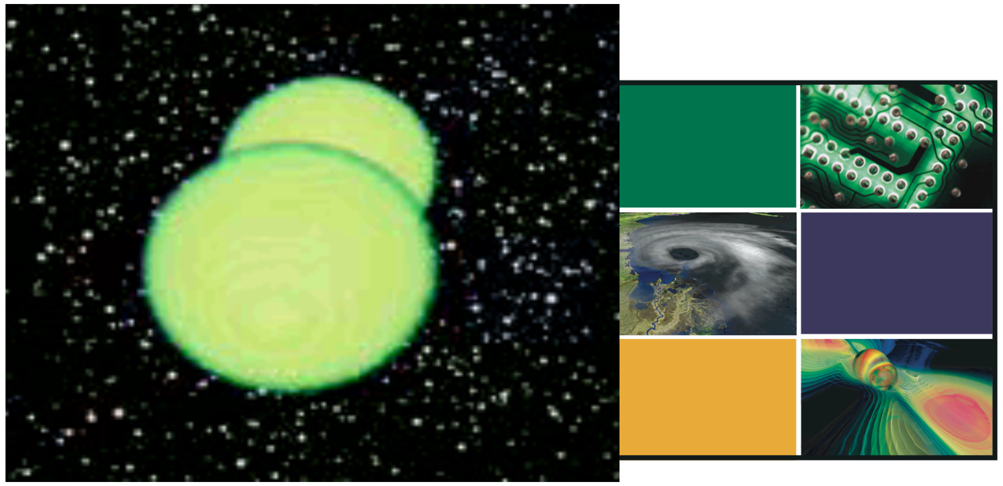
Amira, OpenDX, VisIt

Changing steerable parameters

- Parameters
- Physics, algorithms
- Performance

Streaming HDF5
auto-downsample





Summary

- Cactus has long history of distributed/grid computing; made possible by framework model separating data representation from computations
- Distributed computing is beginning to be production-mode reality for us (“us” = numerical relativity)
- Important: need to stay in control of infrastructure, need to be able to override services
- Problem: not really supported by policies at HPC centres