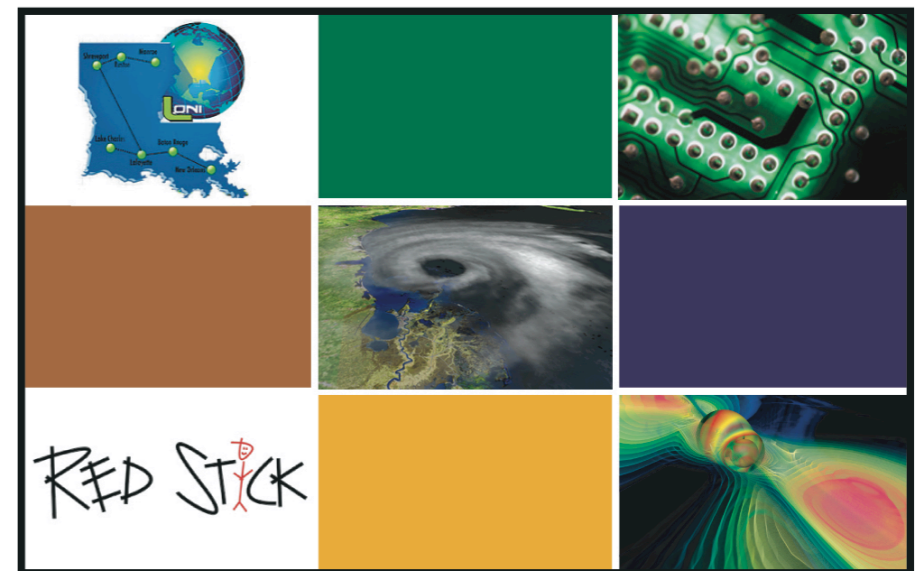


A Case Study for Petascale Applications in Astrophysics: Simulating Gamma-Ray Bursts

C. D. Ott, E. Schnetter, G. Allen, E. Seidel, J. Tao, B. Zink
Baton Rouge, February 2008



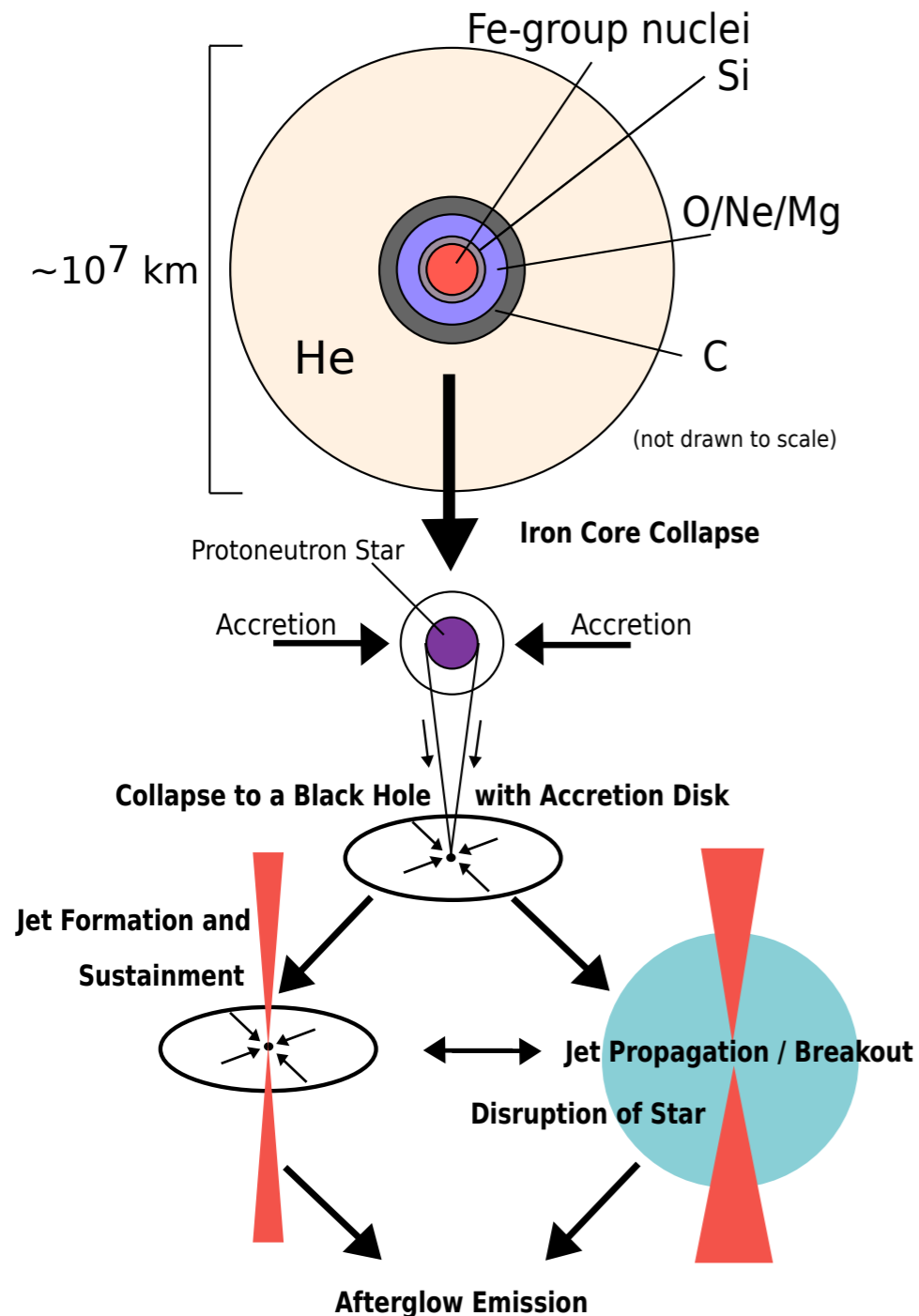
CENTER FOR COMPUTATION
& TECHNOLOGY



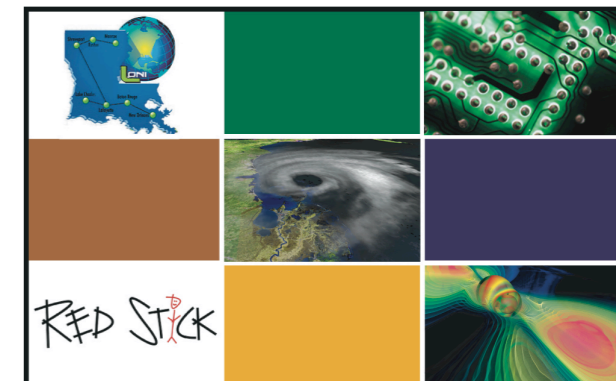


CENTER FOR COMPUTATION
& TECHNOLOGY

Gamma Ray Bursts



- Intense narrowly-beamed flashes of high-energy photons; most energetic events in universe
- Mechanism still a riddle (*grand challenge* in astrophysics); gravitational waves likely to be detected by LIGO in coming years
- Combine many fields of physics
- Require (at least) petascale computing for modelling

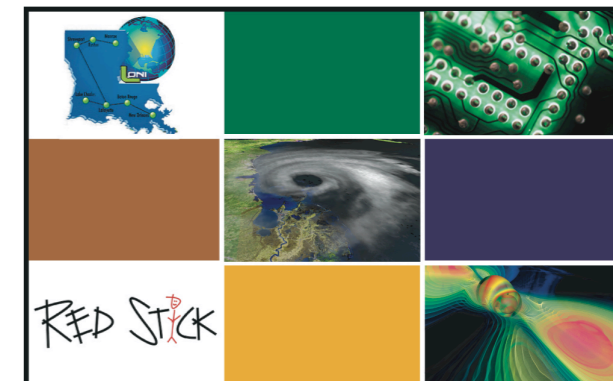
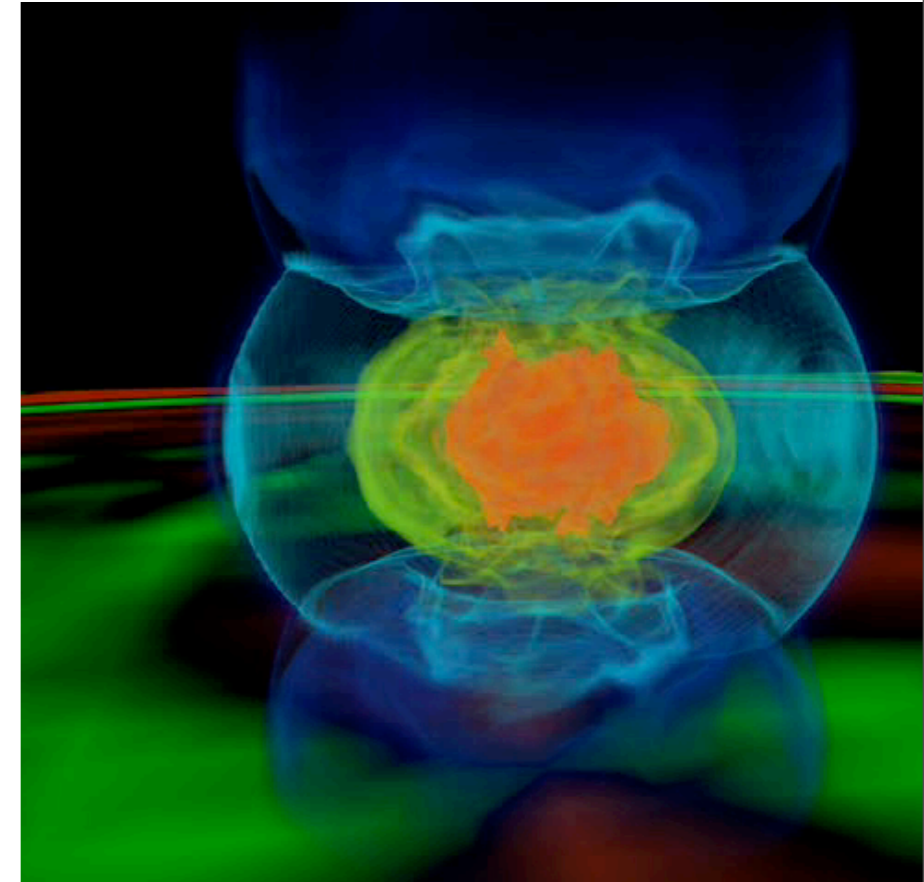




CENTER FOR COMPUTATION
& TECHNOLOGY

Petascale Needs of Gamma Ray Bursts

- Treat entire GRB with multiple physics codes, within one managing infrastructure
- Multitude of time- and length-scales
- Several phases with different relevant physics



Estimated Requirements to Model a GRB

	Core collapse, Supernova	Accretion, Jet Formation	Break-out	Afterglow
Physics	EOS, neutrino transport, MHD, GR	neutrino transport, MHD, GR	photon transport, MHD, coupled to accretion	photon transport/absorption, nuclear decay
	25 m... 10,000 km 1 ms...2 s	25 m...1,000 km 1 ms...200 s	1,000 km... 1,000,000 km 1 ms...200 s	even larger months
Computation	AMR: 11 levels 640M cells 5M steps	AMR: 10 levels 80M cells 600M steps	AMR: 15 levels 100M cells 6M steps	Monte Carlo?
	18 TByte 270,000 PFlop (3 days)	3 TByte 6,000,000 PFlop (70 days)	25 TByte 300,000 PFlop (3 days)	?

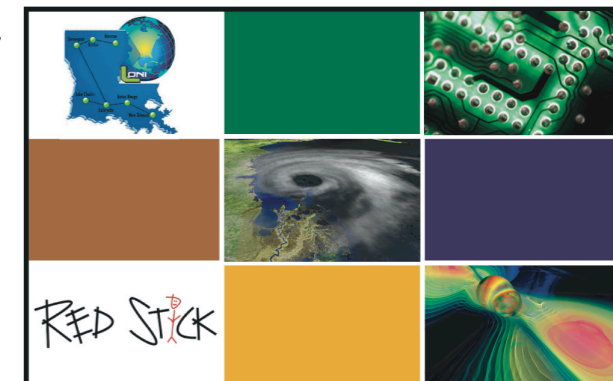


Accuracy Problem

- Many sequential time steps
- Not only slow – errors also accumulate:

$$E(t) = E_0 \frac{t}{t_0} \left(\frac{h}{h_0} \right)^\alpha$$

- Requires either smaller time steps, or *very high accuracy* per time step: 5E-8 relative error in the accretion/jet formation phase!

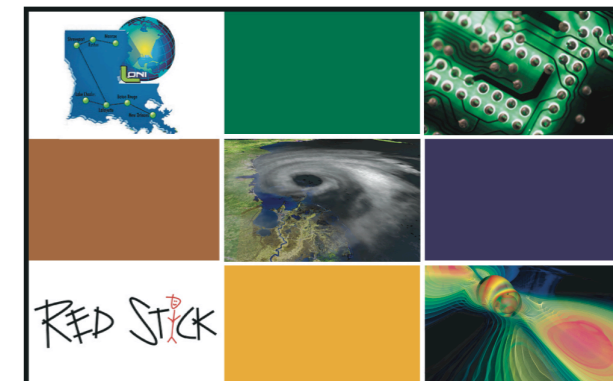




CENTER FOR COMPUTATION
& TECHNOLOGY

Petascale Challenges

- Future will bring $O(1,000,000)$ cores per petascale machine
- Architectures will be more difficult to program (Cell, GPU, deeper hierarchies)
- Applications will combine different kinds of physics – need to be developed and tested separately





CENTER FOR COMPUTATION
& TECHNOLOGY

Computational Infrastructure

Software framework Cactus:

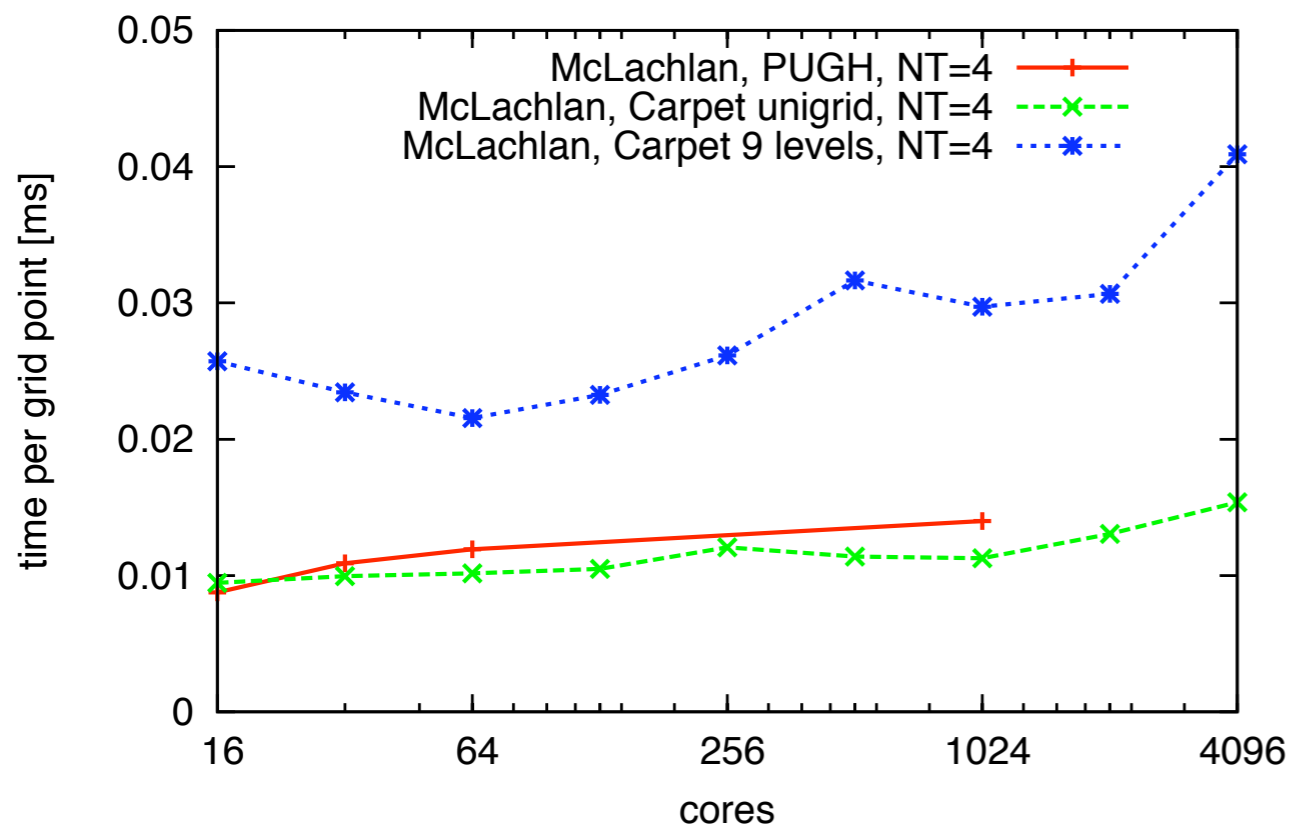
<http://www.cactuscode.org/>

AMR driver Carpet:

<http://www.carpetcode.org/>

- Software framework connects different codes
- Highly efficient adaptive mesh refinement (AMR) covers many time/length scales

Weak Scaling on Ranger

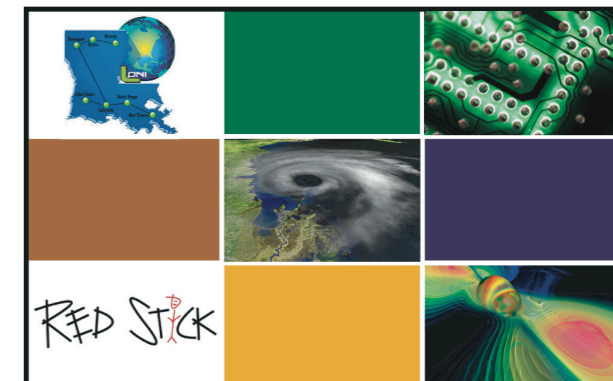




CENTER FOR COMPUTATION
& TECHNOLOGY

Cactus Framework

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

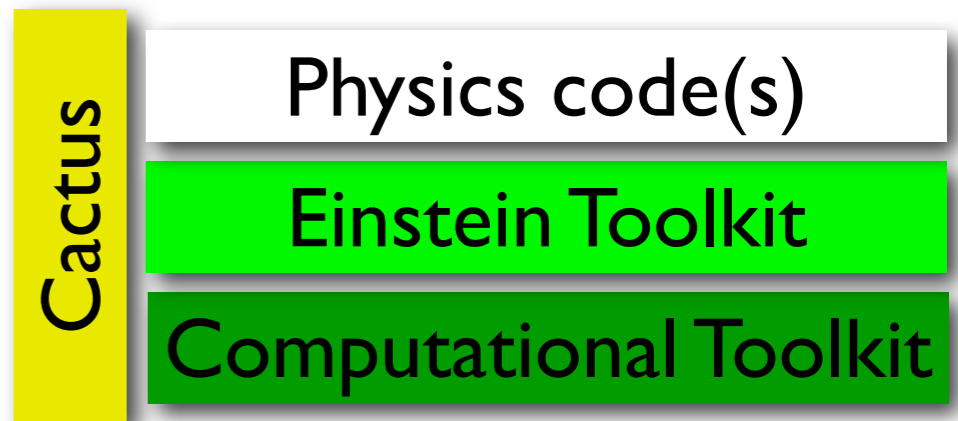




CENTER FOR COMPUTATION
& TECHNOLOGY

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

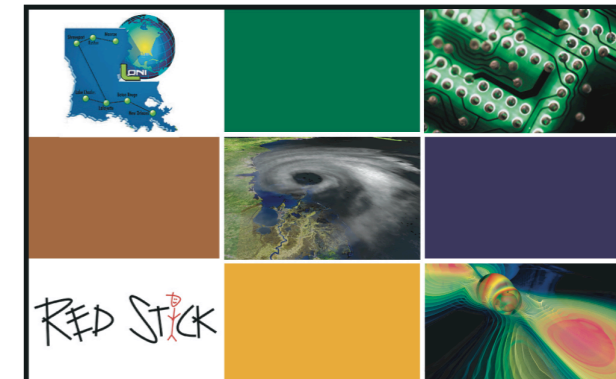
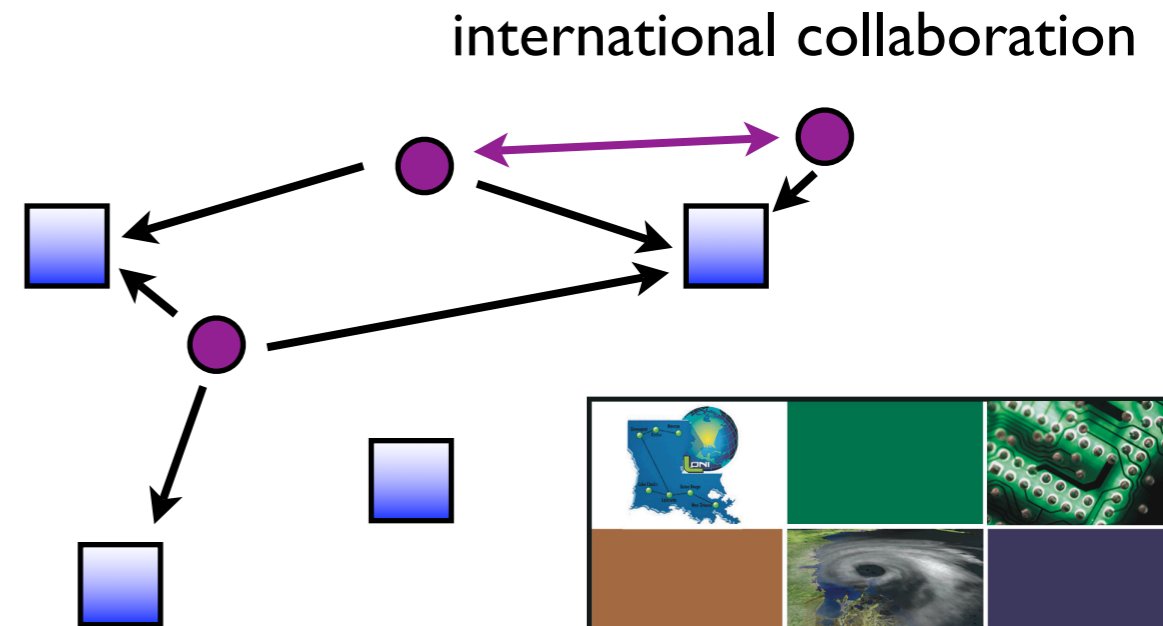
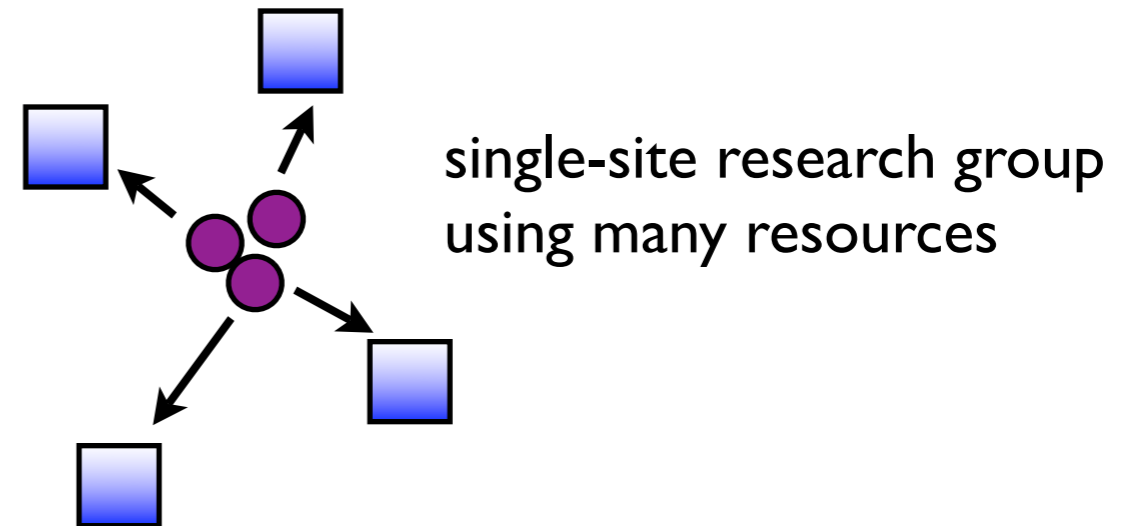




CENTER FOR COMPUTATION
& TECHNOLOGY

Distributed Application Development

- “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”

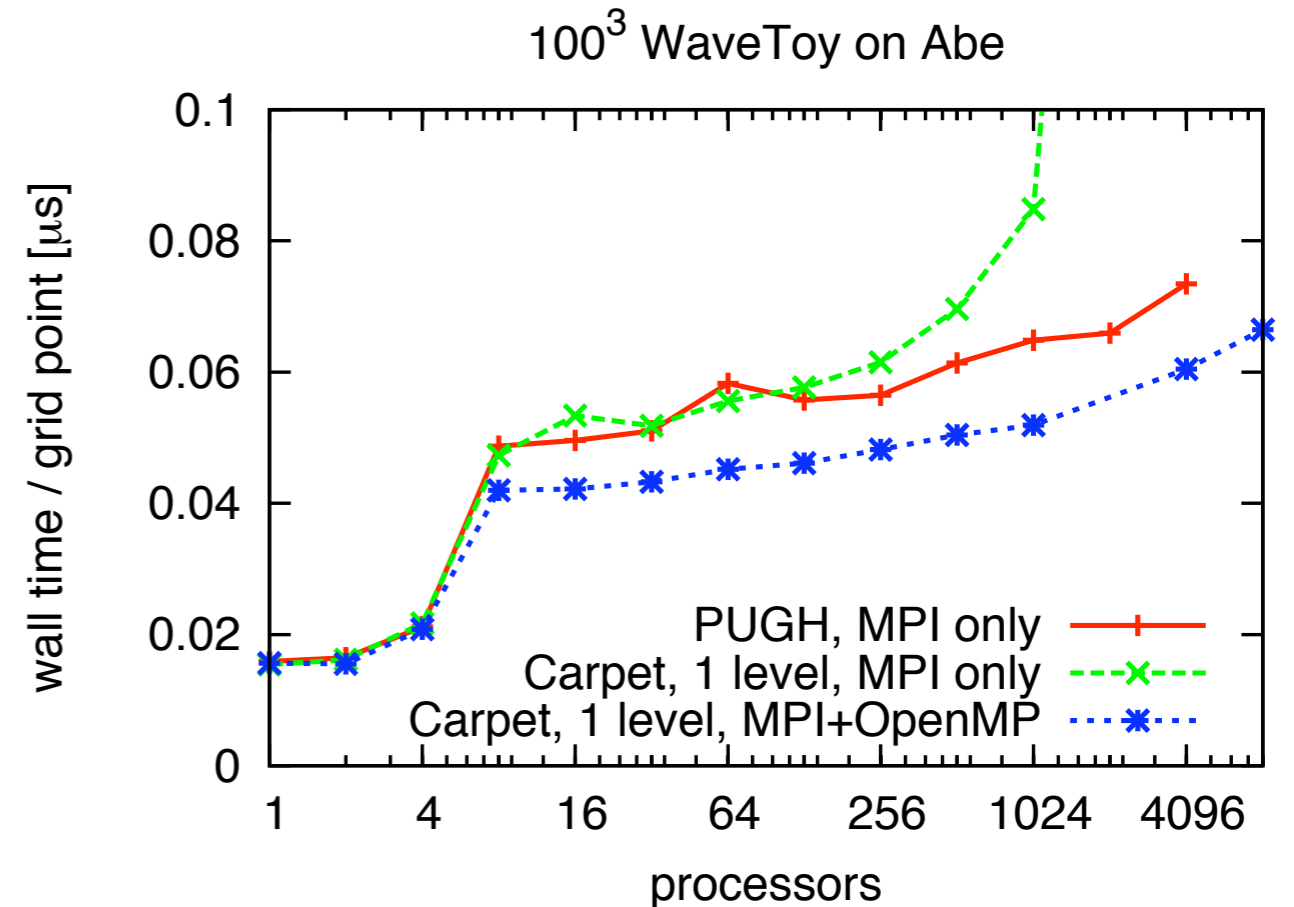




CENTER FOR COMPUTATION
& TECHNOLOGY

Hybrid Communication Schemes

- Combine MPI and OpenMP for multi-core machines (e.g. Abe, Ranger)
- Reduce parallelisation overhead
- Allow more cache optimisations
- On Abe (NCSA):
>10% faster



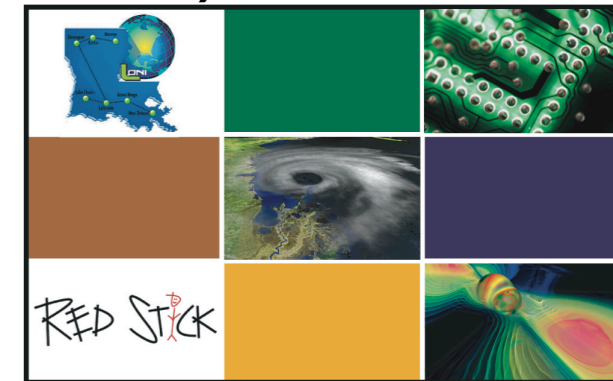


CENTER FOR COMPUTATION
& TECHNOLOGY

Future Hybrid Accelerated Architectures

- Cell, GPGPU, etc. promise great performance at low hardware (but high programmer) cost
- High-level abstractions (much higher than CUDA) required to program these
- We are designing such abstractions for Cactus for block-structured grids (AMR)
- One test case has speed-up of **26**

Einstein code using Cactus and CUDA described in
<http://www.cct.lsu.edu/CCT-TR/CCT-TR-2008-1>





CENTER FOR COMPUTATION
& TECHNOLOGY

Automated Code Generation

- Write equations in Mathematica, then discretise and implement them automatically
- Out of the box as fast as tediously hand-optimised Fortran code
- Allows consistency checks, code reorganisation, many optimisations possible (e.g. vectorisation, cache tiling)

Kranc: KRanc Assembles Numerical Code
<http://numrel.aei.mpg.de/Research/Kranc/>





CENTER FOR COMPUTATION
& TECHNOLOGY

Multi-Machine Simulations

- LONI: many machines, high-speed network
- Combine power to increase performance, reduce wait time (*metacomputing*)
- Combine machines running different codes (*coupled models*)
- Off-load post-processing or visualisation (*task spawning*)

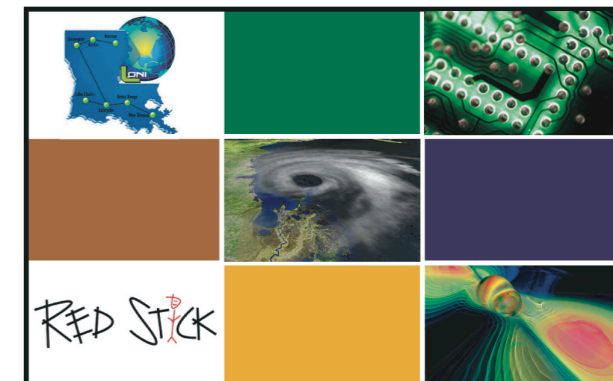




CENTER FOR COMPUTATION
& TECHNOLOGY

Data and Metadata Management

- Efficient I/O necessary (~20 TByte per run)
- Validation, provenance, archival/retrieval (*metadata management*)
- Core metadata describe simulation, Key metadata describe physics
- Cactus offers integrated metadata management (e.g. thorn *Formaline*)





Summary and Question Marks

- Estimated petascale requirements of a current grand challenge problem
- Presented currently-known issues, and concepts to address them
- But: should we think differently about this?
Should we drop Flop/s, Bandwidth, Latency?
Maybe instead measure MTBF?
Data mobility? Hardware elasticity?

