

Binary black holes on the grid

Denis Pollney

*Max-Planck-Institut für Gravitationsphysik
(Albert-Einstein-Institut)
Golm, Germany*

April 2003

Outline:

- Introduction to numerical relativity
- Physicist vs. Supercomputer
- Where the grid fits in

Numerical relativity

- Einstein's equations relate spacetime geometry to distribution of matter

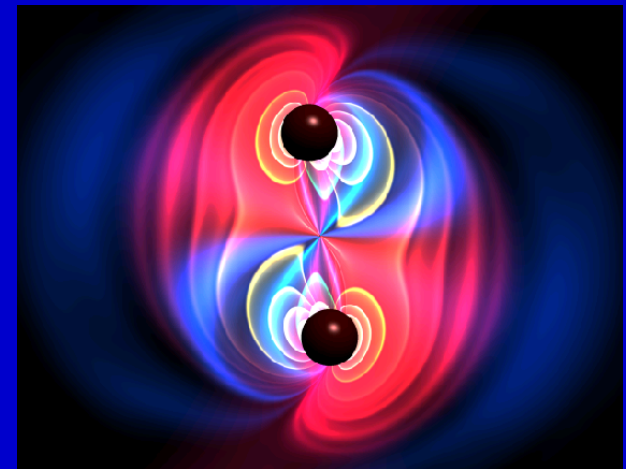
$$R_{\alpha\beta} - \frac{1}{2}g_{\alpha\beta}R = 8\pi T_{\alpha\beta}$$

(geometry) \approx (matter)

- ★ a set of 10 second-order PDEs in space and time
- ★ highly non-linear
- ★ exact solutions only known for very special cases
 - * eg. cosmological solutions, plane waves, single stationary black holes
- ★ equations in full generality can only be studied using computers
- Predict the existence of *gravitational radiation*
 - ★ movement of matter \longrightarrow propagating waves in the geometry
- Strongest potential source of detectable gravitational radiation: a pair of coalescing *black holes*
 - ★ even in this case, signal is incredibly weak
 - ★ would like accurate models



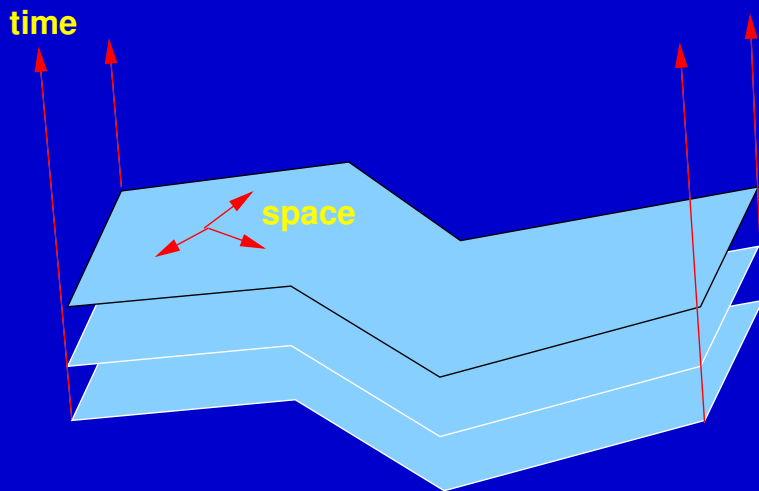
(geo600 detector, hannover)



(scientific american, april 2002)

Evolution system:

- numerically, we treat the Einstein equations as an *initial-boundary value problem*:
 - ★ choose variables describing
 - the geometry of a 3D spacelike “slice”
 - its embedding in the 4D spacetime
 - ★ fix these variables at a given time $t = 0$
 - ★ evolve forward in time using the Einstein equations
- the Einstein equations become a constrained evolution system



Evolution equations:

$$\partial_t \tilde{\gamma}_{ab} = -2\alpha \tilde{A}_{ab} + \mathcal{L}_\beta \tilde{\gamma}_{ab}$$

$$\partial_t \phi = -\frac{1}{6}\alpha K + \mathcal{L}_\beta \phi$$

$$\partial_t \tilde{A}_{ab} = e^{-4\phi} (D_a D_b \alpha + \alpha R_{ab})^{\text{TF}} + \alpha (K \tilde{A}_{ab} - 2 \tilde{A}_{ai} \tilde{A}^i_b) + \mathcal{L}_\beta \tilde{A}_{ab}$$

$$\partial_t K = -D^i D_i \alpha + \alpha (\tilde{A}_{ij} \tilde{A}^{ij} + \frac{1}{3} K^2) + \mathcal{L}_\beta K$$

$$\begin{aligned} \partial_t \tilde{\Gamma}^a = & \tilde{\gamma}^{ij} \partial_i \partial_j \beta^a + \frac{1}{3} \tilde{\gamma}^{ai} \partial_i \partial_j \beta^j \\ & + \beta^i \partial_i \tilde{\Gamma}^a - \tilde{\Gamma}^i \partial_i \beta^a + \frac{2}{3} \tilde{\Gamma}^a \partial_i \beta^i \\ & - 2 \tilde{A}^{ai} \partial_i \alpha + 2\alpha (\tilde{\Gamma}^a_{ij} \tilde{A}^{ij} + 6 \tilde{A}^{ai} \partial_i \phi \\ & - \frac{2}{3} \tilde{\gamma}^{ai} \partial_i K) \end{aligned}$$

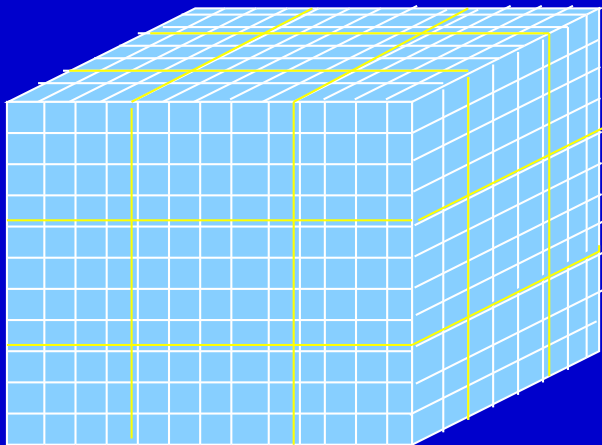
Constraints:

$$R + K^2 - K_{ij} K^{ij} = 0$$

$$D^i (K_{ai} - \gamma_{ai} K) = 0$$

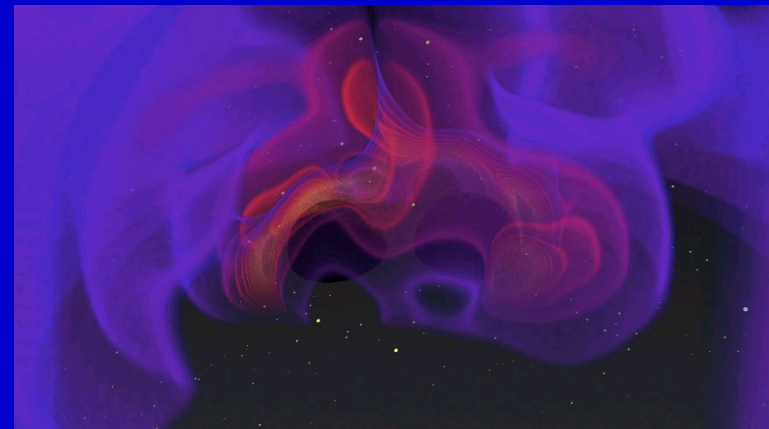
Implementation details

- Our “space” is a cartesian grid
 - ★ equally spaced (x, y, z) points
 - ★ flat faces (boundaries)
- Evolution equations are discretised
 - ★ functions take values at grid points
 - ★ spatial derivatives become finite differences
 - ★ converge to continuous solution (to some order) as $dx \rightarrow 0$
 - ★ time evolution via method of lines (eg. iterated crank-nicholson, runge-kutta)



Cactus:

- parallelisation
- modular structure
 - ★ gauge/coordinate evolution conditions
 - ★ outer boundary conditions
 - ★ elliptic equation solvers
 - ★ horizon finders (apparent and event)
 - ★ wave extraction
 - ★ etc. . .
- advanced I/O
- cross-platform



A typical run:

- uses a grid of size $384 \times 384 \times 384$
- requires 50-100 GB memory
- runs 24h to merger, using 256 alpha processors
- produces 100 – 1000 GB of data

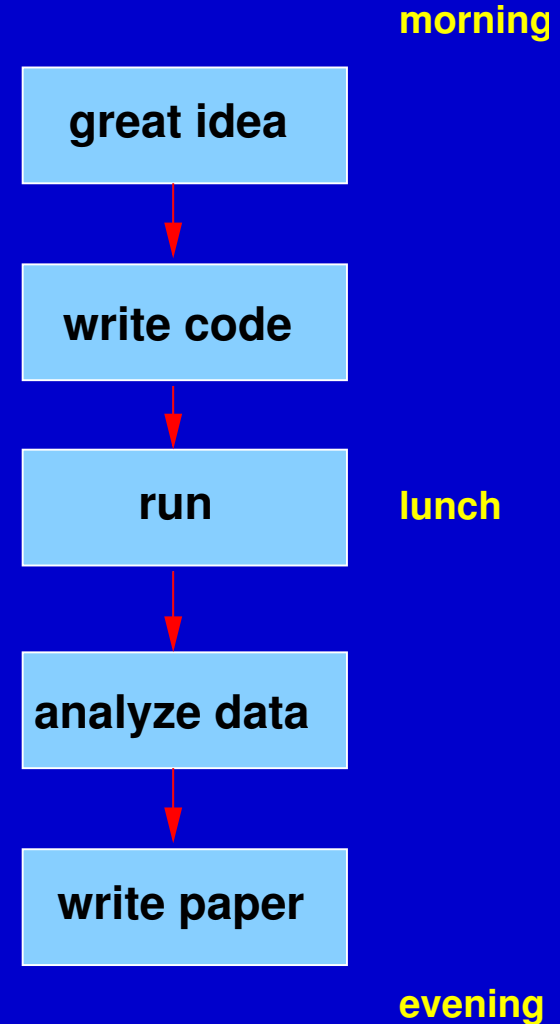
Where are we running?

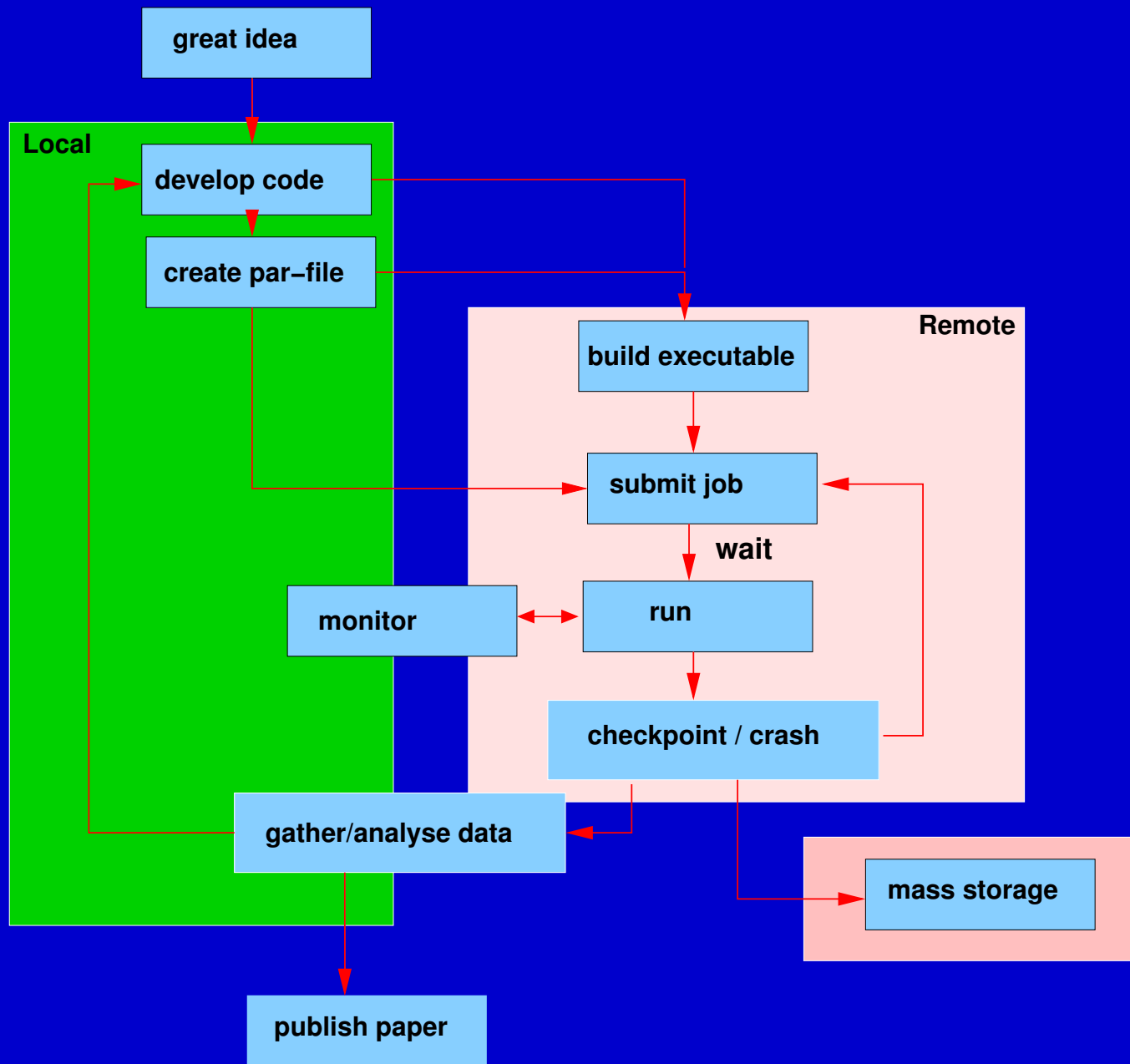
machine	org	proc	compiler
platinum.ncsa.uiuc.edu	NCSA	ia32	intel7/gcc
lemieux.psc.edu	PSC	alpha	native
seaborg.nerisc.gov	NERSC	sp3	native
psi.rzg.mpg.de	RZGarching	regatta	native
sr8000.lrz-muenchen.de	LRZ	sr8000	native
titan.ncsa.uiuc.edu	NCSA	ia64	intel7
origin.aei.mpg.de	AEI	origin	native
AEI workstations	AEI	ia32	intel5/gcc
laptops	AEI	ia32	gcc/intel

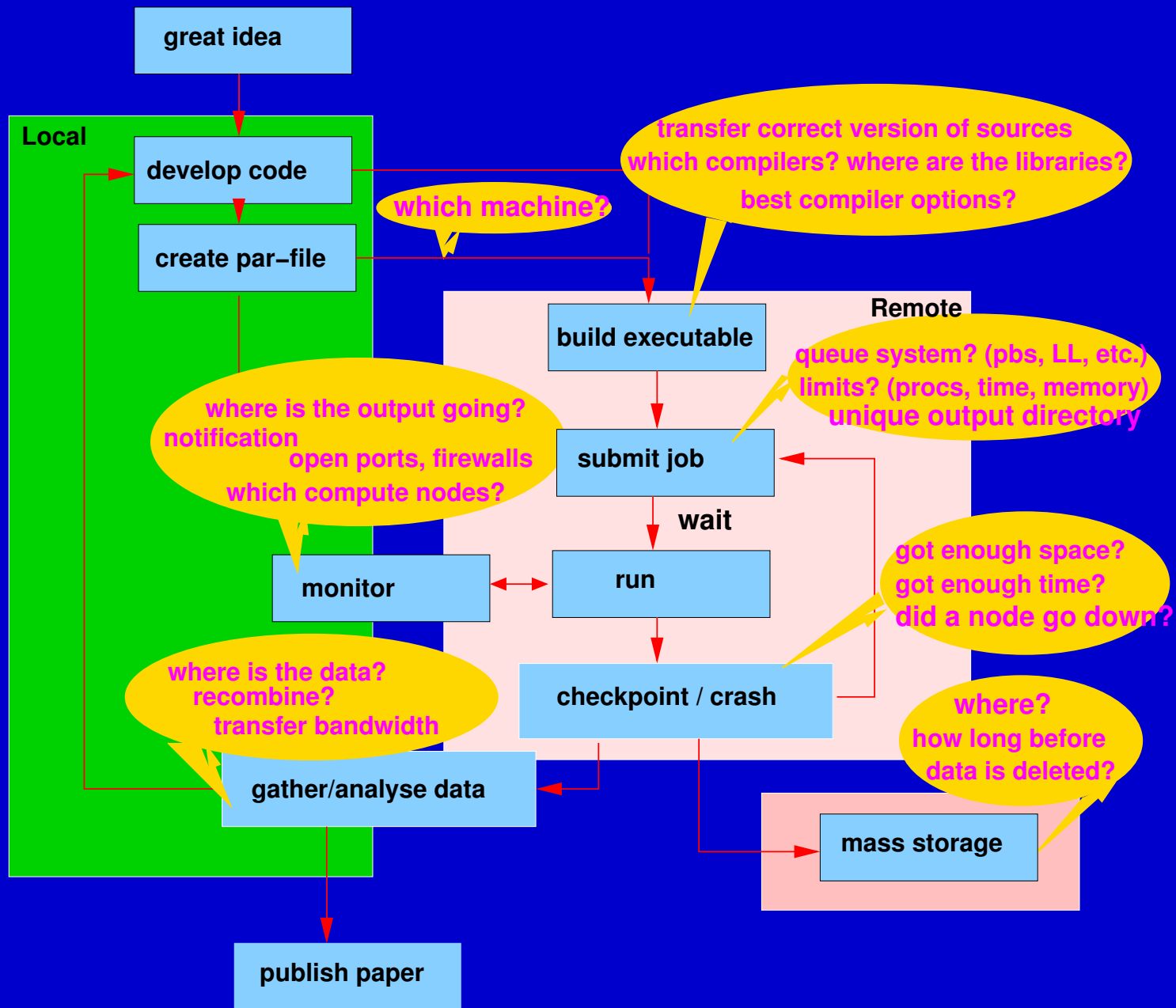
A good day:

Philisophical interlude:

- Physicists would like to do physics
- Computer tools are most useful when they
 - ★ make it easier to turn an idea into analyzable data
 - ★ bring the physicist closer to the data he'd like to analyze







Obstacles that we face:

- inhomogeneous environments (in space and time)
- moving data around
- data analysis
- information about available resources

How do we get around them?

- gather folklore
 - ★ AEIWeb web site
 - ★ collected information about machines, queues, compilers, libraries
- construct a “uniform environment” across machines
 - ★ use env variables and config files to locate libraries, set compiler options, etc.
 - ★ directory structure relative to home
 - ★ scripts
 - * job submissions (qs2)
 - * parameter file sequences (parseq)
 - * building executables (GetCactus, MakeThornList)
- monitor jobs via a web server started by the job itself
 - ★ (need permission to connect, and to know where to connect)
- limit our production of data to manageable quantities :-)

How do we make use of grid tools currently?

- Many of the tasks mentioned are particularly difficult to implement in a uniform way across platforms (eg. compiling code)
- Our most useful experiences with grid tools are with job monitoring and data manipulation
- **Job monitoring:**
 - ★ ASC portal
 - * notifies (via email) of job start
 - * lists active jobs, with links to monitoring web servers
 - ★ Remote visualisation (gridftp, OpenDX, Amira)
 - * Allows for streaming of data from running jobs
 - * Data can be downsampled for bandwidth
 - * Standard visualisation networks (developed by the physicists themselves) provide the views we want to see
- **Data manipulation:**
 - ★ Transfer of HDF5 data via gridftp
 - * Secure movement of large files
 - * Access to pieces of data files

How can the grid be helpful to us right away?

- Heterogeneous environments mean that many of the tasks we'd like to see will take some time to mature to stability
- Certain tasks do lend themselves to immediate implementation
- Improved notification and monitoring via the portal:
 - ★ more information about available resources, queue status, machine loads
 - ★ pick up some of the job monitoring tasks of the HTTPD thorn:
 - * profiling information
 - * simple visuals of data (eg. jpeg profiles)
 - ★ launch local visualisation tools for convenient access to streamed run data
- Task farming:
 - ★ parameter surveys – submit multiple jobs for a sequence of parameter files
 - ★ steering – spawn small 'predictor' runs in parallel to a large run, feed analysis from these runs back to the main simulation
- Automated testing:
 - ★ reduce the time we spend reproducing results
 - ★ for a given set of parameter files, regularly and automatically go to a set of machines:
 - * checkout, compile, run, compare, report