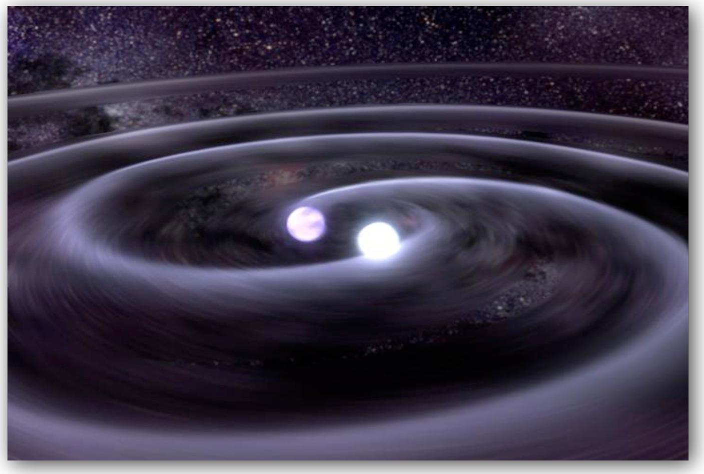
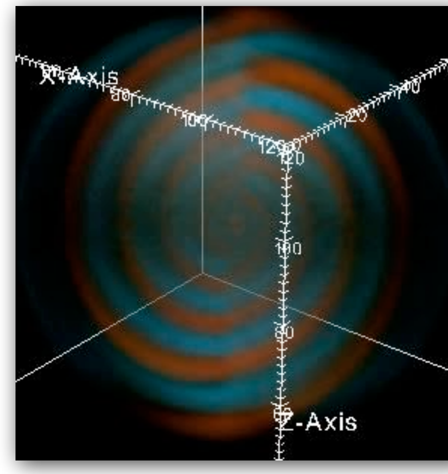




High Performance Computing, Simulation, and Relativity

Ian Hinder

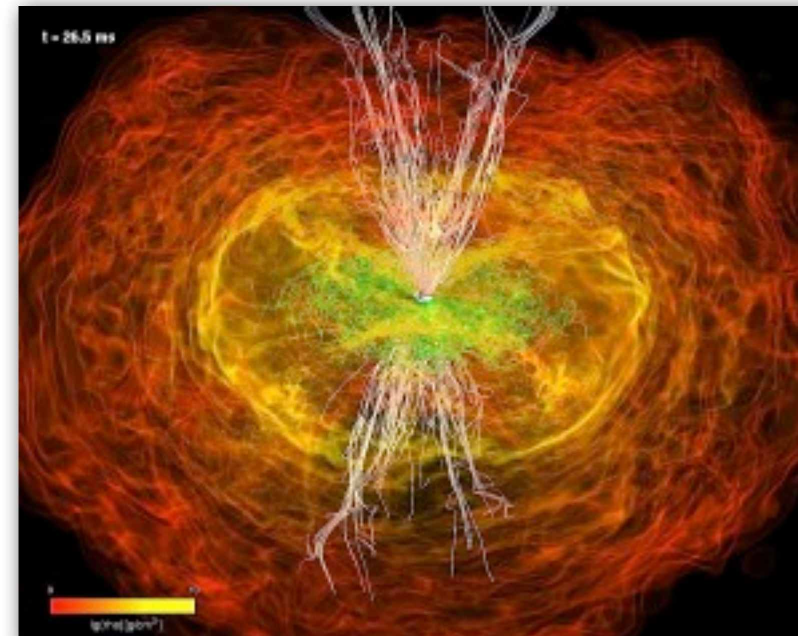
Max Planck Institute for Gravitational Physics,
Potsdam, Germany

5th July 2017
Golm

Too big for a lab? Simulation!

- Can’t **experiment** on black holes/ neutron stars

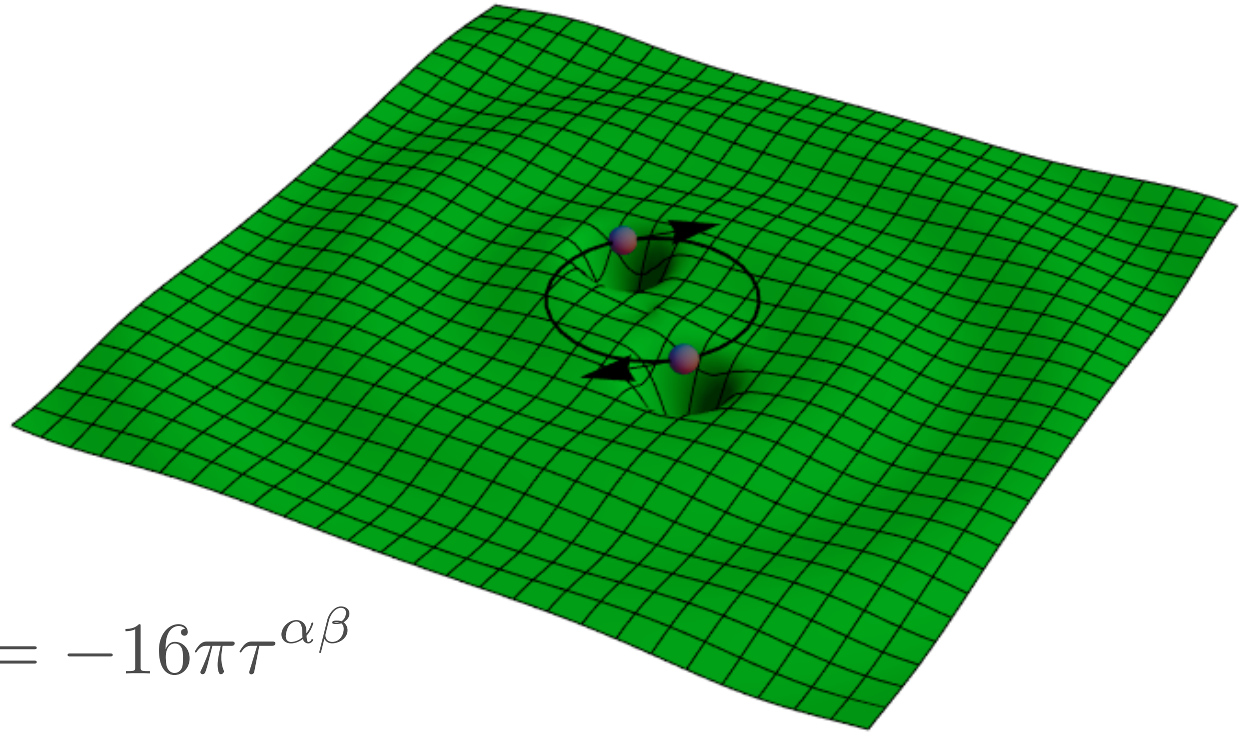
- Use computer **simulations** to see how they behave (assuming certain physics)



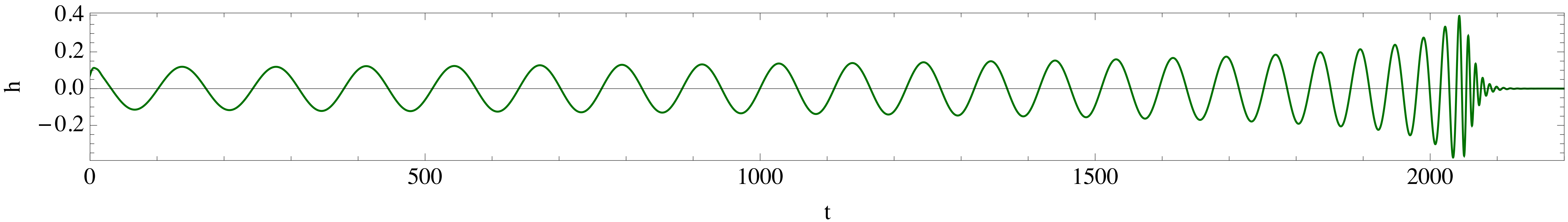
- Compare results with astrophysical **observations**

- Was our **physics model** right?

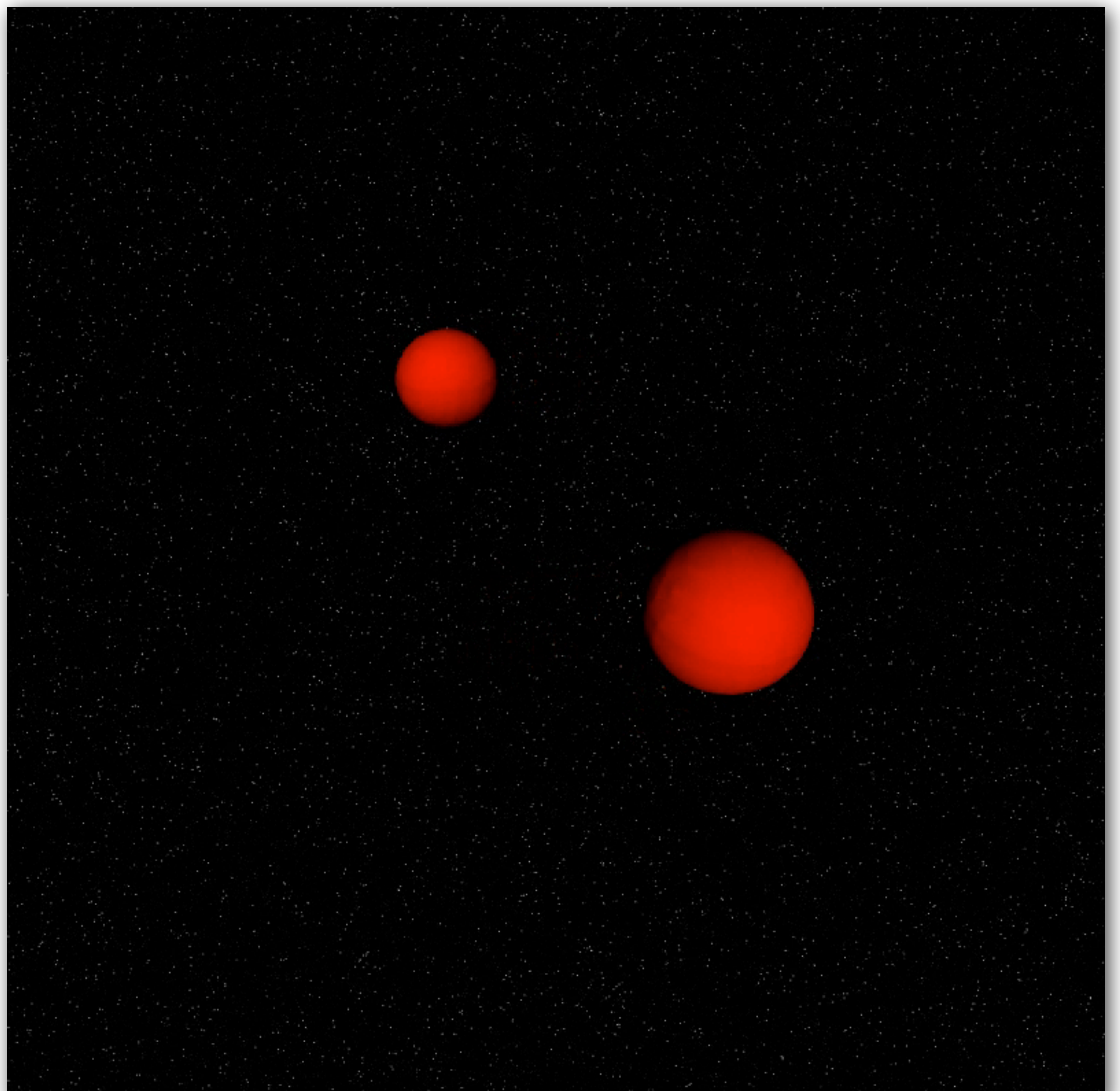
Gravitational waves



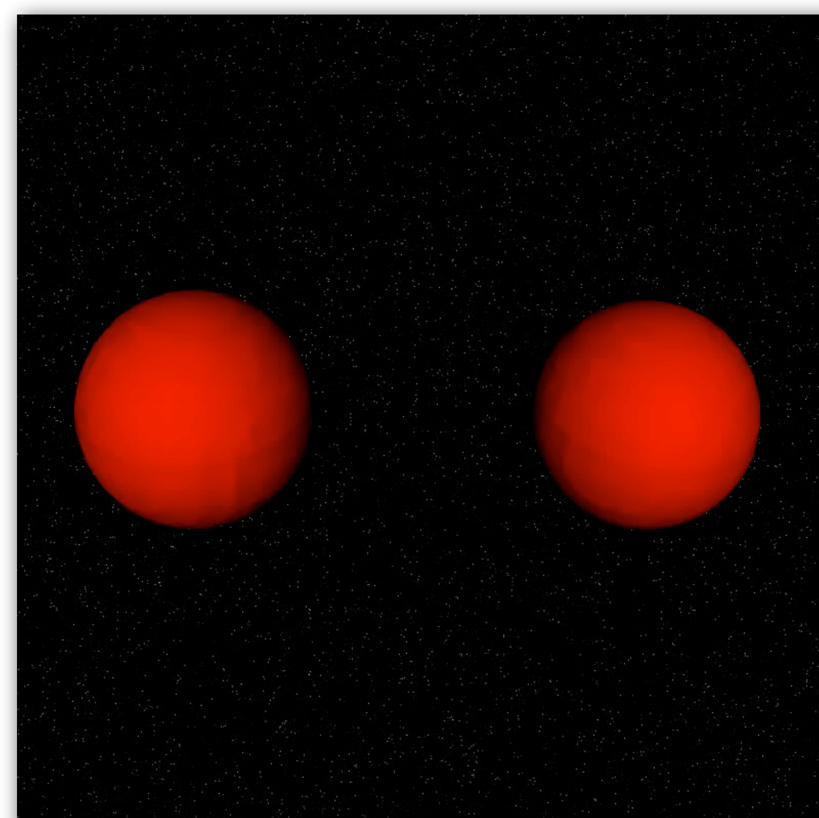
- Dense, fast astrophysical systems can produce **Gravitational Waves**
- First observed in **2015** $(-\partial_t^2 + \nabla^2)h^{\alpha\beta} = -16\pi\tau^{\alpha\beta}$
- Gravitational Wave detectors: **Advanced LIGO/VIRGO**
- **Very weak** signals; need to know what to look for!
- Need **Numerical Relativity** simulations to model the dynamics and predict the waves



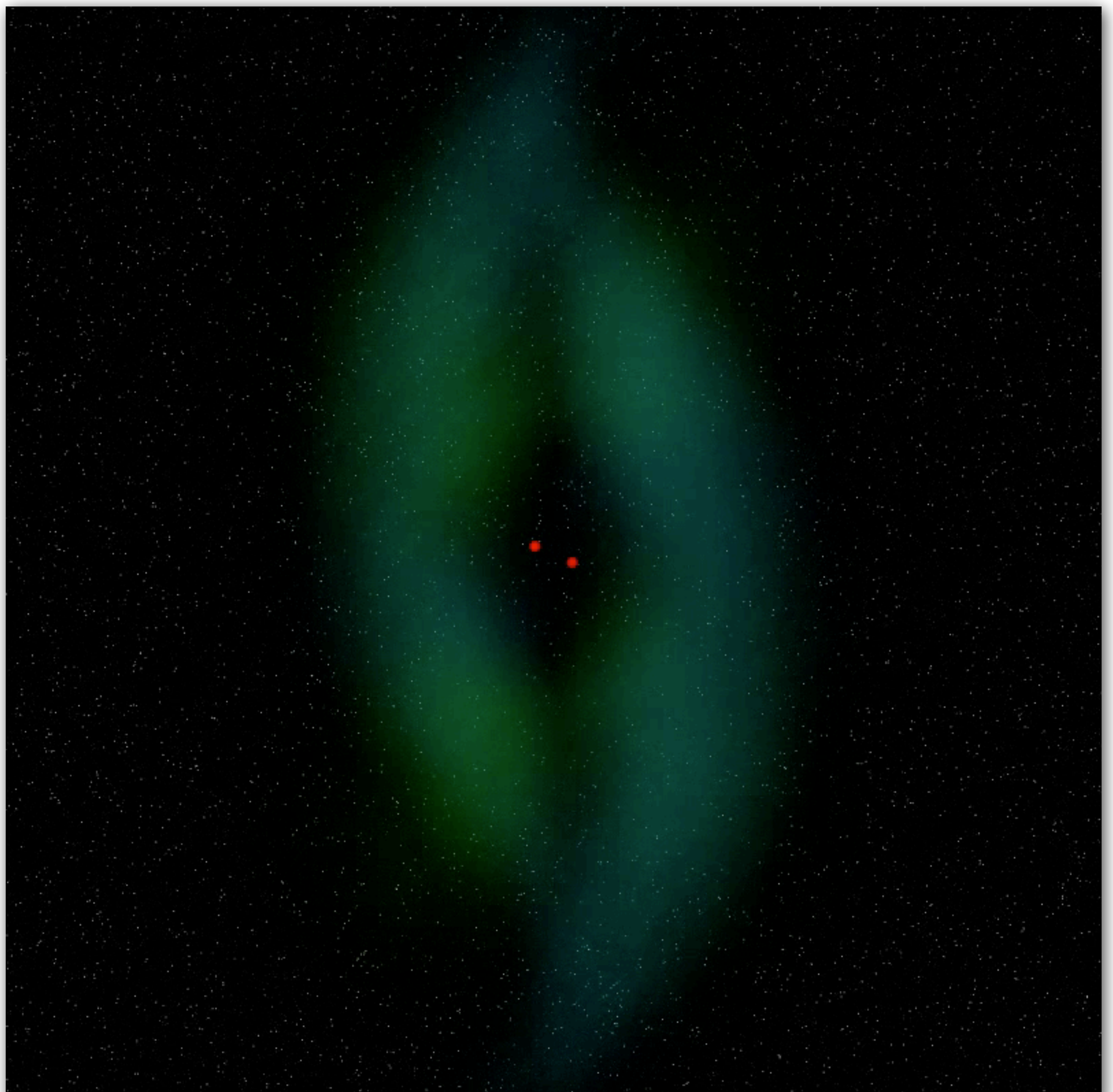
Inspiral and merger of **black hole** binary system



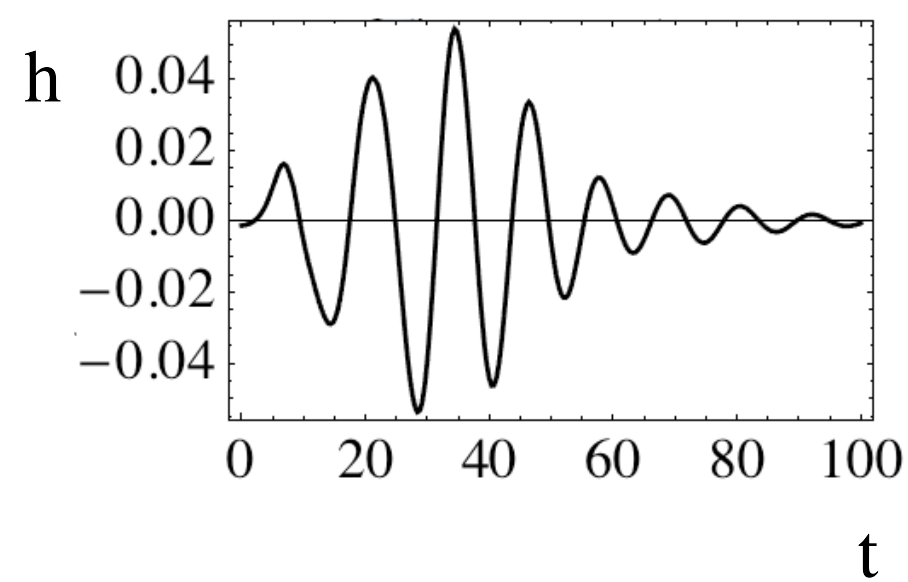
- Black holes **orbiting** around each other
- Lose potential energy by emission of **Gravitational Waves**
- Separation shrinks: black holes **merge**



Gravitational waves from a black hole binary

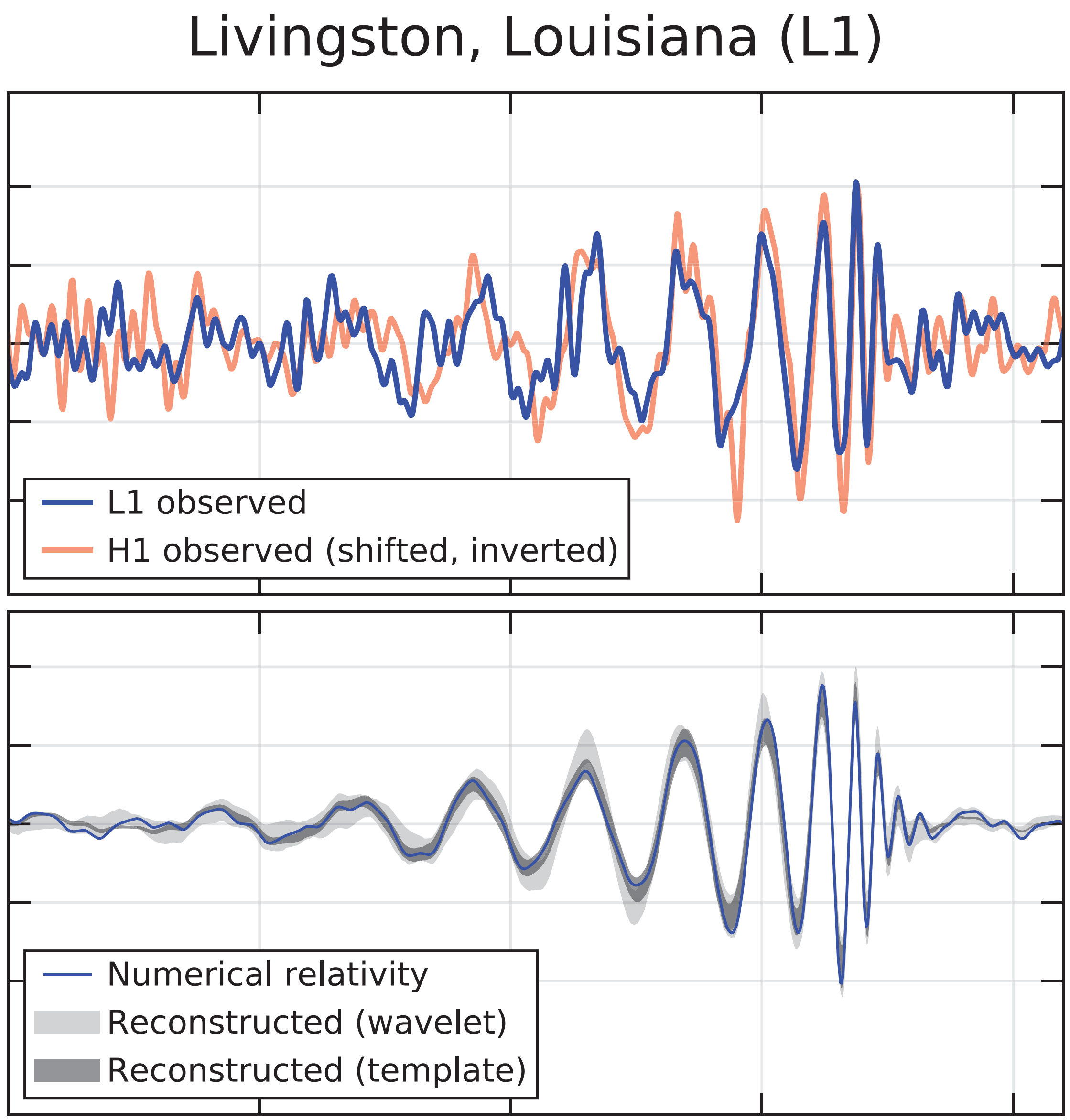


- Gravitational wave strain
  - $h_{\mu\nu}(t, r, \theta, \Phi)$
- Detector measures at a fixed (r, θ, Φ) as a function of time:



GW150914: Observation vs simulation

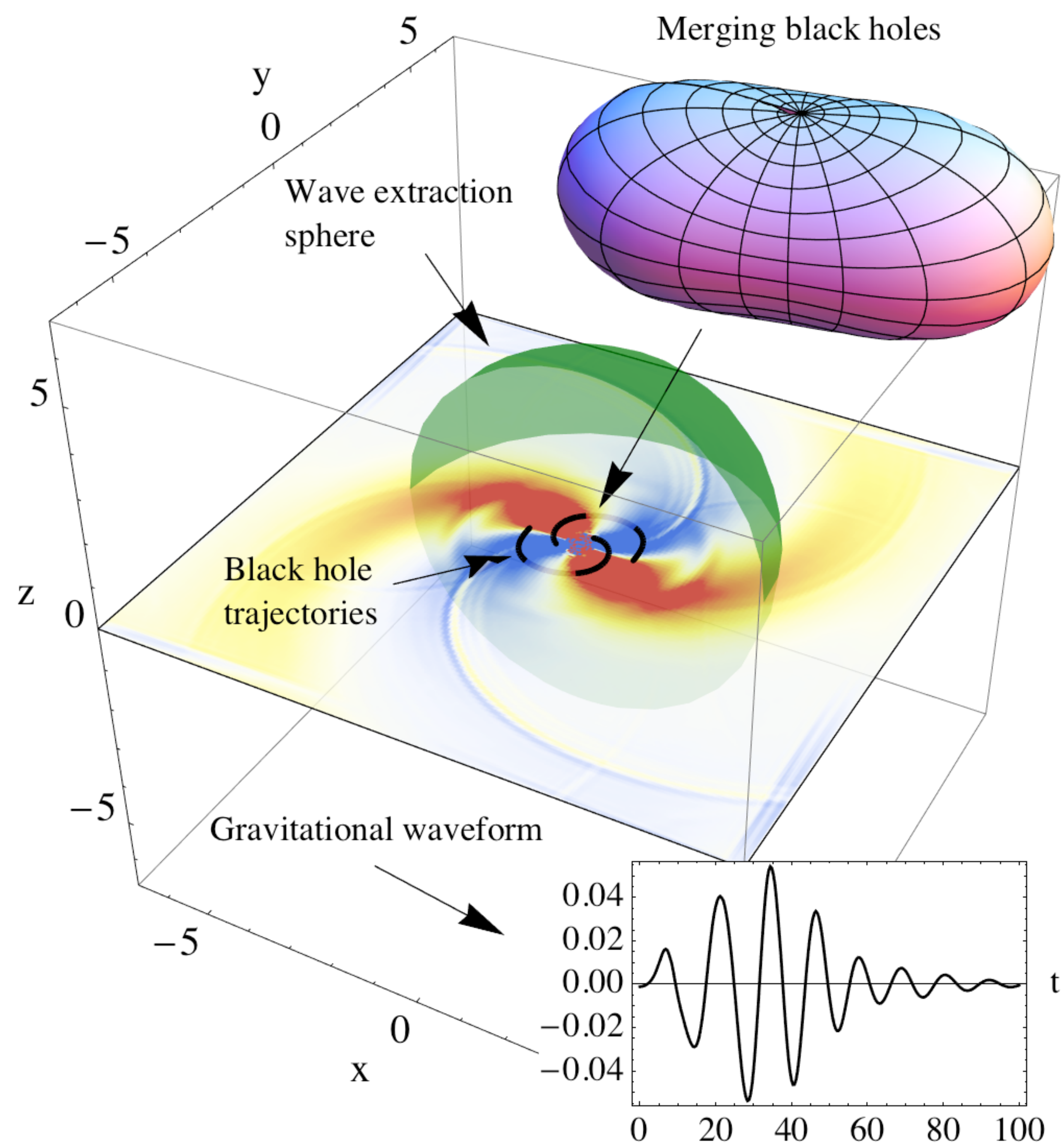
- September 2015: First direct **detection** of gravitational waves (LIGO)
- Excellent **agreement** between observed signal and Numerical Relativity simulations
- In general, require Numerical Relativity to infer **properties** (masses, spins, etc)



Abbott et al. 2015

Numerical Relativity

- Direct solution of **Einstein's equations** on supercomputers
- Major applications:
  - Binary **black holes** and binary **neutron stars**
  - **Supernova** core collapse
- Size: 100 - **1000 cores**
- Simulation time: days/weeks/months



Time evolution partial differential equations (PDEs)

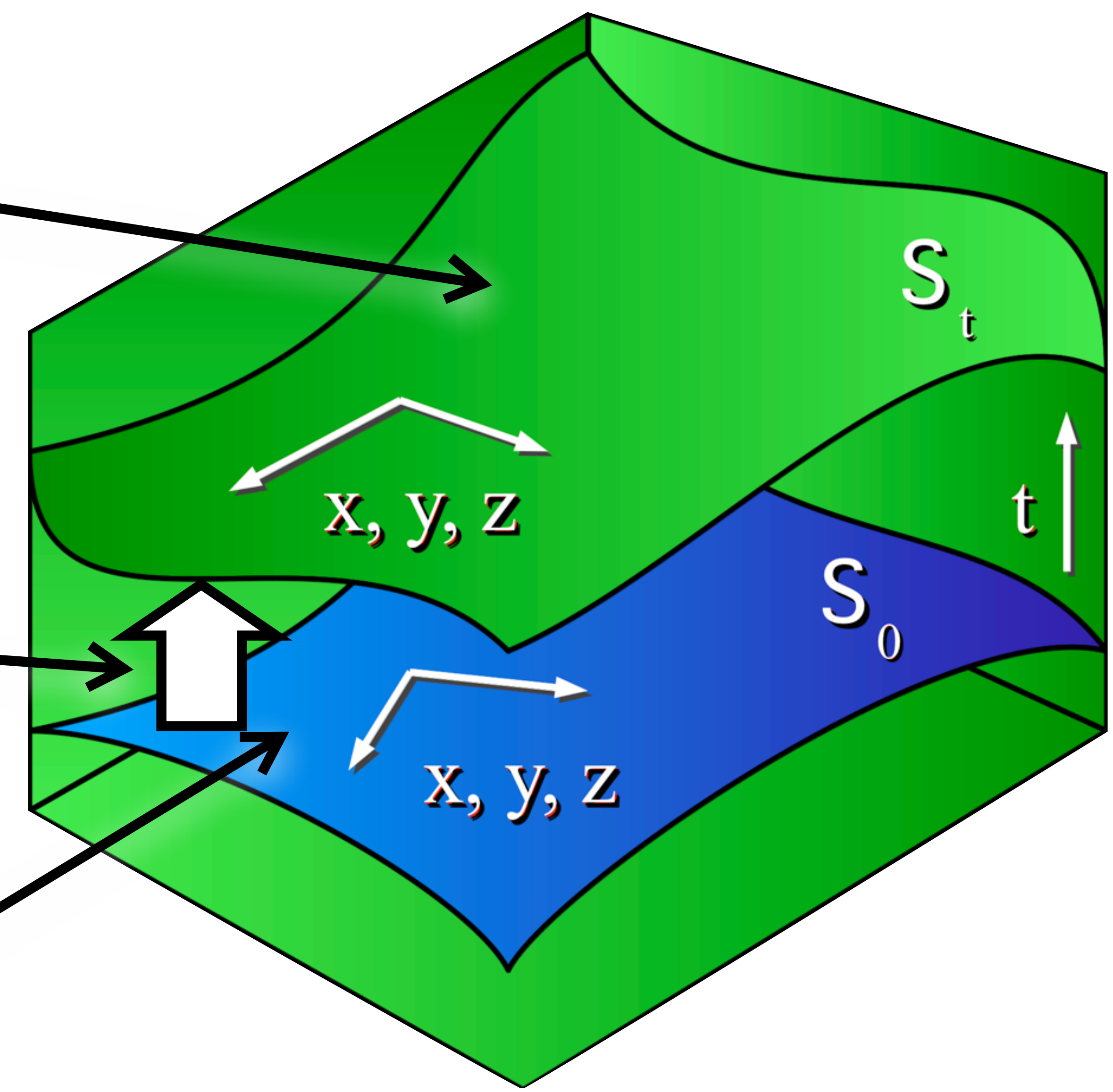
Solution at time t:

$$u(t, \vec{x}) = \begin{pmatrix} u_1 \\ u_2 \\ \dots \\ u_m \end{pmatrix}$$

Evolution equation:

$$\partial_t u(t, \vec{x}) = F(u, \partial u, \partial^2 u)$$

Initial data: $u(0, \vec{x}) = f(\vec{x})$



Einstein equations in time-evolution form

$$\partial_t \hat{\phi}_\kappa = \frac{2}{\kappa}\hat{\phi}_\kappa \alpha K + \beta^i \partial_i \hat{\phi}_\kappa - \frac{2}{\kappa}\hat{\phi}_\kappa \partial_i \beta^i,$$

$$\partial_t \tilde{\gamma}_{ab} = -2\alpha \tilde{A}_{ab} + \beta^i \partial_i \tilde{\gamma}_{ab} + 2\tilde{\gamma}_{i(a}\partial_{b)}\beta^i - \frac{2}{3}\tilde{\gamma}_{ab}\partial_i \beta^i,$$

$$\partial_t K = -D_i D^i \alpha + \alpha(A_{ij}A^{ij} + \frac{1}{3}K^2) + \beta^i \partial_i K,$$

$$\partial_t \tilde{A}_{ab} = (\hat{\phi}_\kappa)^{\kappa/3}(-D_a D_b \alpha + \alpha R_{ab})^{\mathrm{TF}} + \beta^i \partial_i \tilde{A}_{ab} + 2\tilde{A}_{i(a}\partial_{b)}\beta^i - \frac{2}{3}A_{ab}\partial_i \beta^i,$$

$$\partial_t \tilde{\Gamma}^a = \tilde{\gamma}^{ij}\partial_i \beta_j \beta^a + \frac{1}{3}\tilde{\gamma}^{ai}\partial_i \partial_j \beta^j - \tilde{\Gamma}^i \partial_i \beta^a + \frac{2}{3}\tilde{\Gamma}^a \partial_i \beta^i - 2\tilde{A}^{ai}\partial_i \alpha + 2\alpha(\tilde{\Gamma}^a_{ij}\tilde{A}^{ij} - \frac{\kappa}{2}\tilde{A}^{ai}\frac{\partial_i \hat{\phi}_\kappa}{\hat{\phi}_\kappa} - \frac{2}{3}\tilde{\gamma}^{ai}\partial_i K),$$

$$R_{ij} = \widetilde{R}_{ij} + R^{\phi}_{ij},$$

$$R^{\phi}_{ij} = -2\widetilde{D}_i \widetilde{D}_j \phi - 2\widetilde{\gamma}_{ij}\widetilde{D}^k \widetilde{D}_k \phi + 4\widetilde{D}_i \phi \widetilde{D}_j \phi - 4\widetilde{\gamma}_{ij}\widetilde{D}^k \phi \widetilde{D}_k \phi,$$

$$\widetilde{R}_{ij} = -\frac{1}{2}\widetilde{\gamma}^{lm}\partial_l \partial_m \widetilde{\gamma}_{ij} + \widetilde{\gamma}_{k(i}\partial_{j)}\widetilde{\Gamma}^k + \widetilde{\Gamma}^k \widetilde{\Gamma}_{(ij)k} + \widetilde{\gamma}^{lm}(2\widetilde{\Gamma}^k{}_{l(i}\widetilde{\Gamma}_{j)km} + \widetilde{\Gamma}^k{}_{im}\widetilde{\Gamma}_{klj}).$$

$$\partial_t \alpha - \beta^i \partial_i \alpha = -2\alpha K,$$

$$\partial_t \beta^a - \beta^i \partial_i \beta^a = \frac{3}{4}\alpha B^a,$$

$$\partial_t B^a - \beta^j \partial_j B^i = \partial_t \tilde{\Gamma}^a - \beta^i \partial_i \tilde{\Gamma}^a - \eta B^a,$$

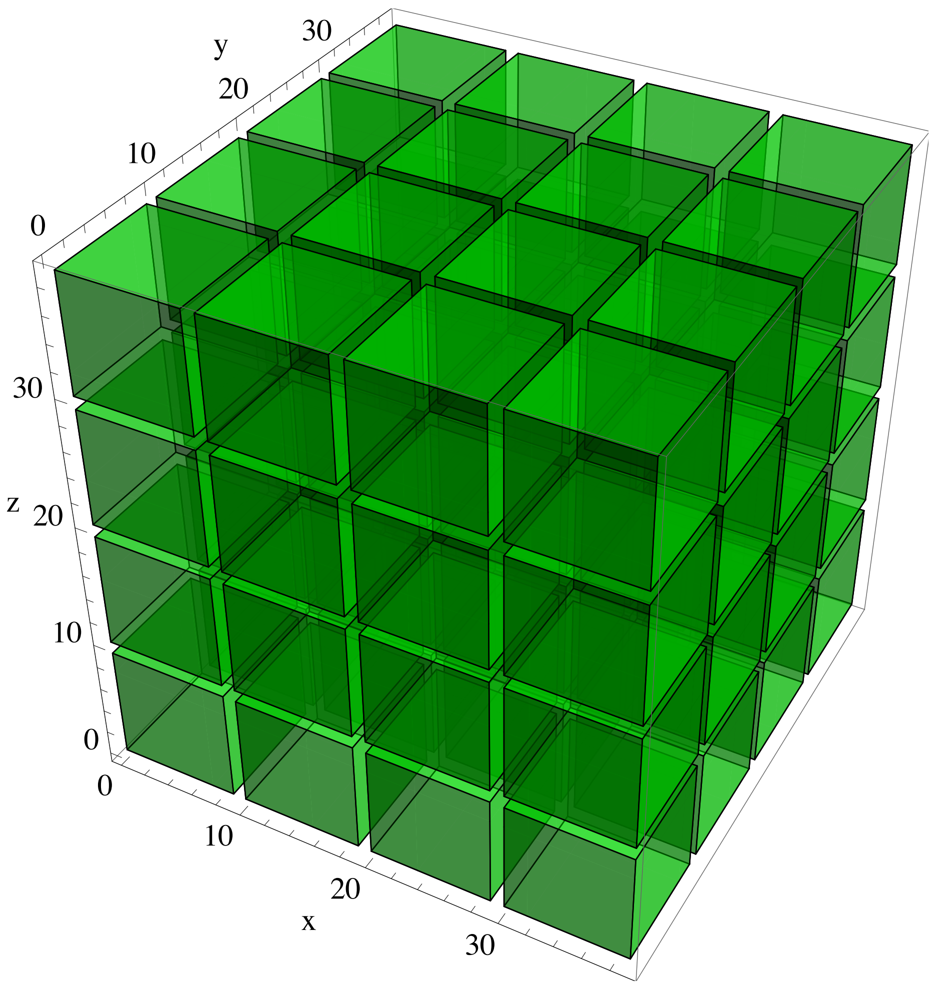
$$\mathcal{H} \equiv R^{(3)} + K^2 - K_{ij}K^{ij} = 0,$$

$$\mathcal{M}^a \equiv D_i(K^{ai} - \gamma^{ai}K) = 0.$$

- **Tensor** equations
- Use **computer algebra** to manipulate/optimise
- Automatically generate **C code** to solve them (15000 lines)

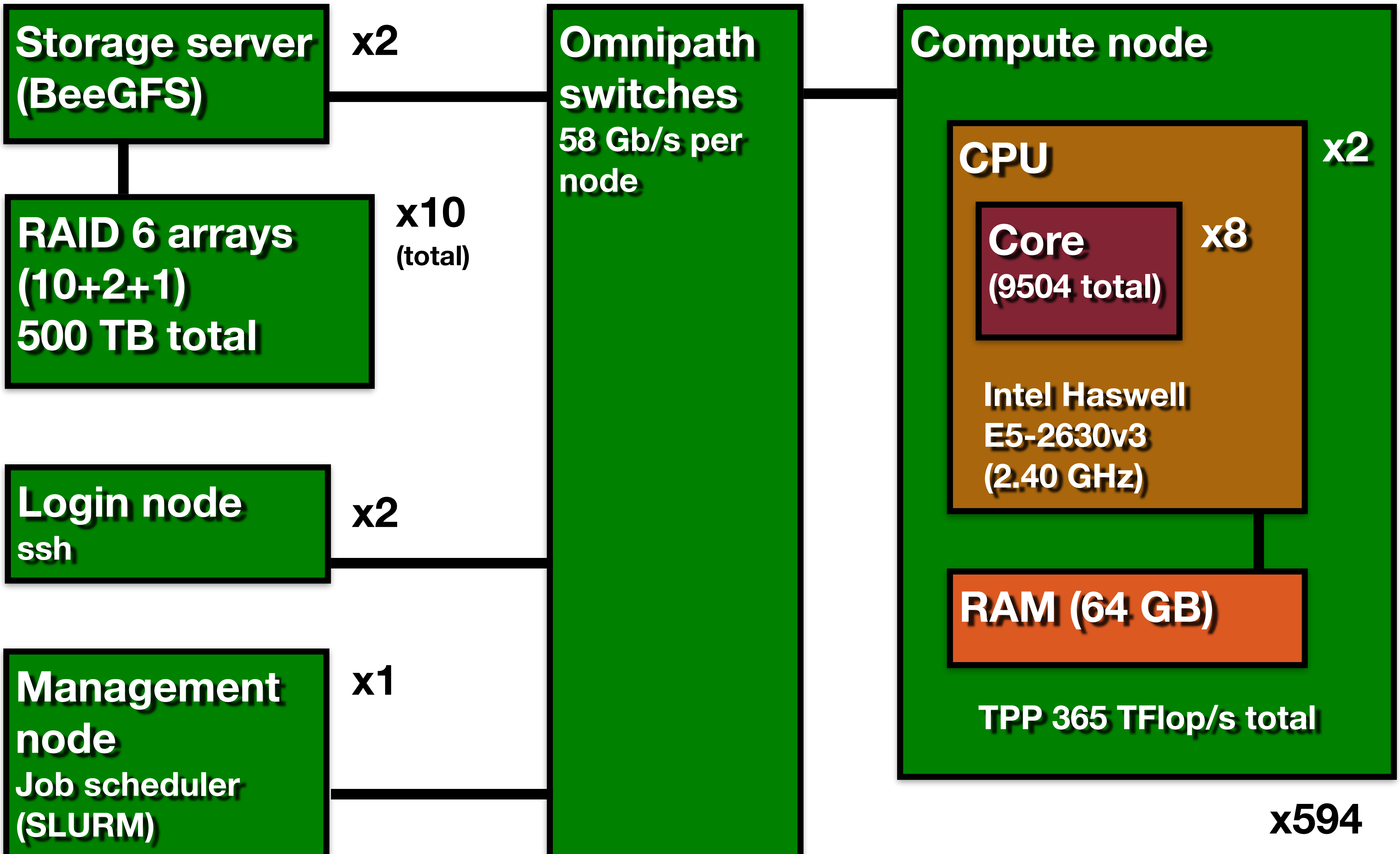
Why clusters?

- Need to **store** at least one 3D t = const grid of data **in memory**
- **Too many points** and too many variables to fit in a single workstation



- Many **nodes** connected by a fast **interconnect**
- **Split up** the grid into blocks and run each on a separate node
- **Parallel programming** required!

AEI Minerva cluster



Running relativity simulations

HPC in practice

Workflow

- **Log in** to cluster (ssh from terminal): `ssh minerva.aei.mpg.de`
- **Compile** code: `sim build`
- Create a **parameter file**: `emacs bbh.par`
- **Submit** a simulation: `sim submit --cores 256 bbh.par`
- Wait for the simulation to **start**
- **Monitor** the simulation: `https://minerva.aei.mpg.de/.../simulations`
- Wait for the simulation to **finish**
- **Post-process** the data: Python/Mathematica
- **Analyse** the data: workstation, plots, movies, ...
- Write a **paper**!

Simplified parameter file concept

bbh.rpar

```
BlackHoleInitialData::separation                   = 10
BlackHoleInitialData::mass1                        = 0.8
BlackHoleInitialData::mass2                        = 0.2

Grid::outer_boundary                                                = 2000

AdaptiveMeshRefinement::cells_across_black_hole = 10
AdaptiveMeshRefinement::cells_across_wavelength = 16

Waveforms::extraction_radius = 100

Output::variables                 = "gxx h22"
Output::out_every                 = "256"
```

```
U:---   bbh.rpar        All L14     (Fundamental)
```

Job characteristics

- Run time: **weeks** or **months**
- Job runtime limit: **24 hours**
- Simulation split into a **sequence of jobs**
- End of job: write **checkpoint file**
- Start of job: read checkpoint file and **continue**

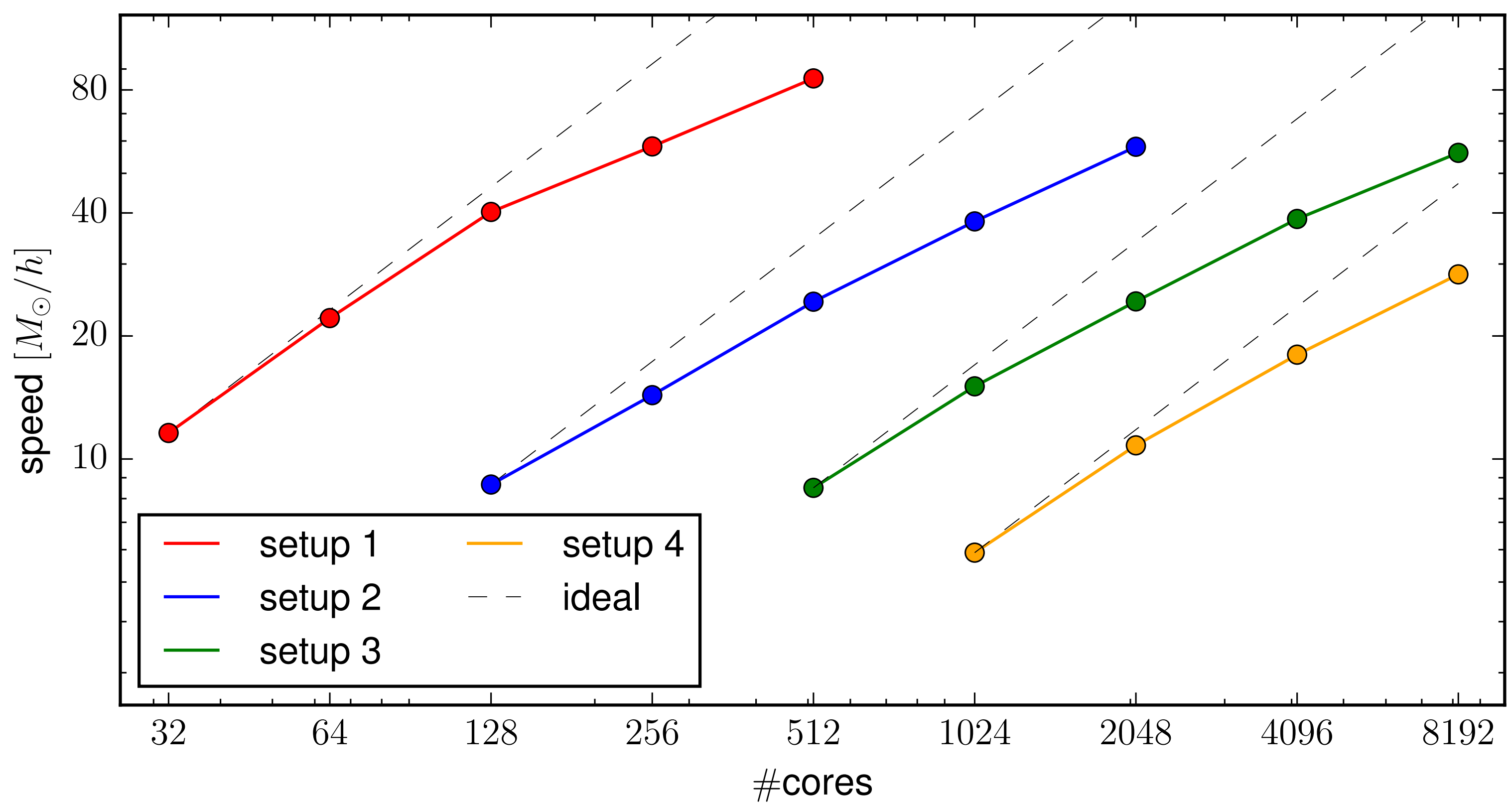
- Job sizes: from **~64** cores to **~1000** cores
  - 64 cores: **binary black hole**; multi-domain spectral methods, adapted grids, poor parallelisation, very accurate, long run time
  - 1000 cores: **binary neutron star**; finite difference methods, Cartesian grids, good parallelisation and scalability, lots of fine detail, low order methods, high resolution

Scaling

- Grid with #CELLS cells **distributed** among #CORES cores

- **Ideal** scaling:

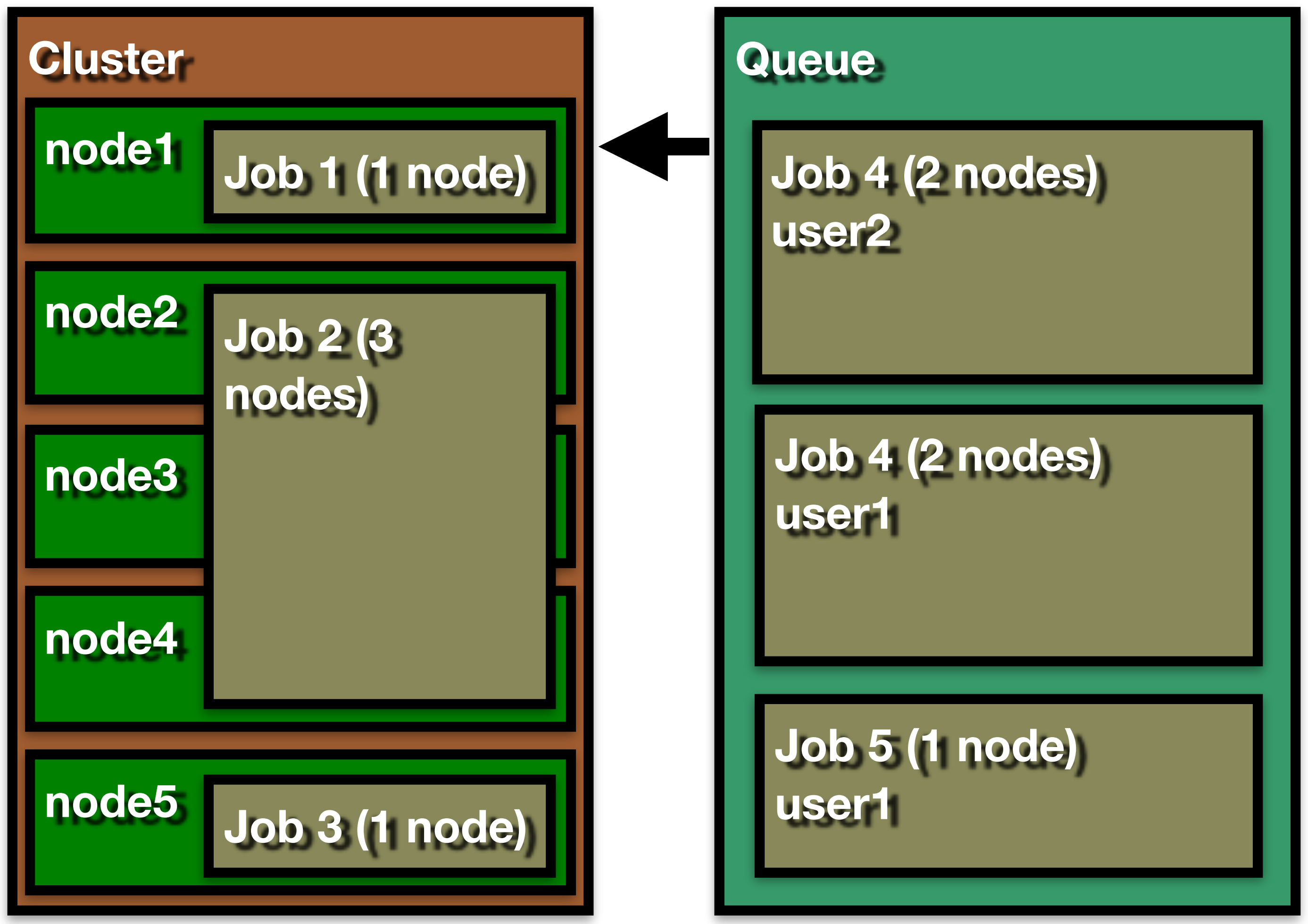
$$\text{speed} = \text{const} \times \frac{\#\text{CORES}}{\#\text{CELLS}}$$



- Why not ideal?
    - Simple model above neglects **communication**
    - Not possible to achieve perfect **load balancing**
    - Same #CELLS but different **shapes** of 3D grid: different speeds (memory layout, **CPU cache** characteristics)

Queuing system

- Multiple **users**, each with multiple **jobs**
- How to **choose** which jobs to run **where** and **when**?
- Submit job to **queue**
- **Queueing system** (SLURM) chooses **which jobs** to run, and **which nodes** to run them on
- Queuing **policy** tries to implement some **fairness**



Open-source Numerical Relativity

- **Cactus** framework: open source, developed by **Ed Seidel**'s group at the **Albert Einstein Institute** in the late 90s

- **Einstein Toolkit** is an open source set of relativity codes based around Cactus

- See einsteintoolkit.org/gallery.html for examples

- Binary black hole gravitational wave **GW150914 example**, from the first LIGO detection, including parameter file, **tutorials** for analysis and visualisation [Wardell, Hinder, Bentivegna]

- Simulate GW150914 on ~100 cores in a few days **yourself**!