SpECTRE: Toward simulations of binary black hole mergers using Charm++

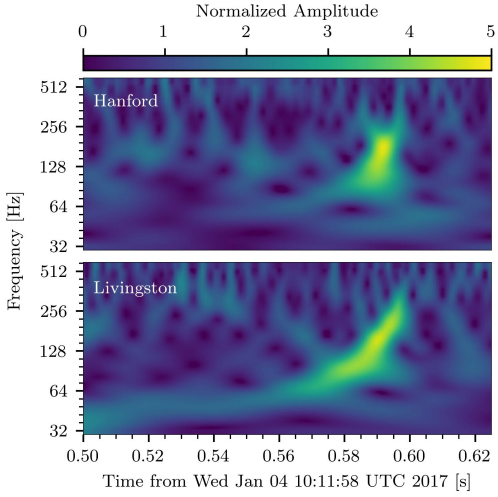
François Hébert @ Caltech
for the Simulating eXtreme Spacetimes (SXS) Collaboration

Charm++ Workshop, Oct 20 2020

Outline

1. Role of binary merger simulations
2. Current methods and challenges
3. SpECTRE: towards improved algorithms and scalability
4. Preliminary binary BH results
5. Load-balancing with Charm++

Gravitational waves



LIGO/Caltech/MIT

LIGO/Virgo detect gravitational waves from merging binary BHs (and NSs)

Simulation waveforms enable

- ▷ detection of weak signals
- ▷ characterization

Future detectors will need significantly more accurate waveforms

Modeling relativistic matter

Recent observations

- merging binary NSs
- accretion around supermassive BH

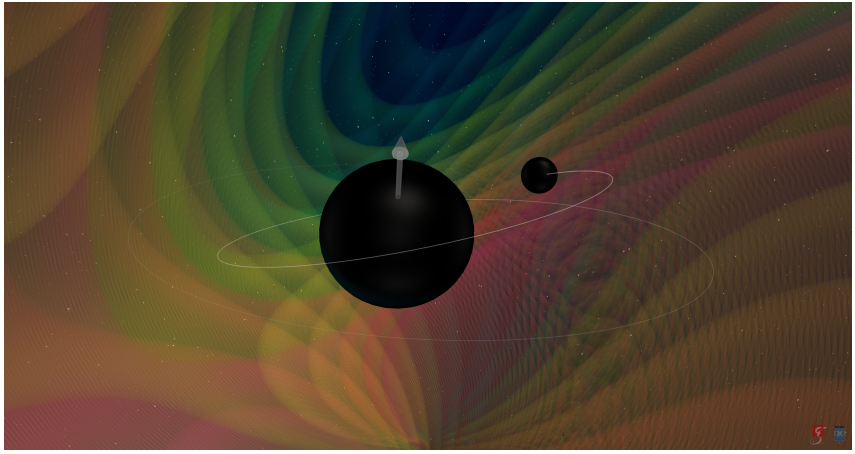
Simulations provide models for

- matter dynamics
- heavy-element creation
- electromagnetic spectra

Simulations are expensive and struggle to reach desired accuracy

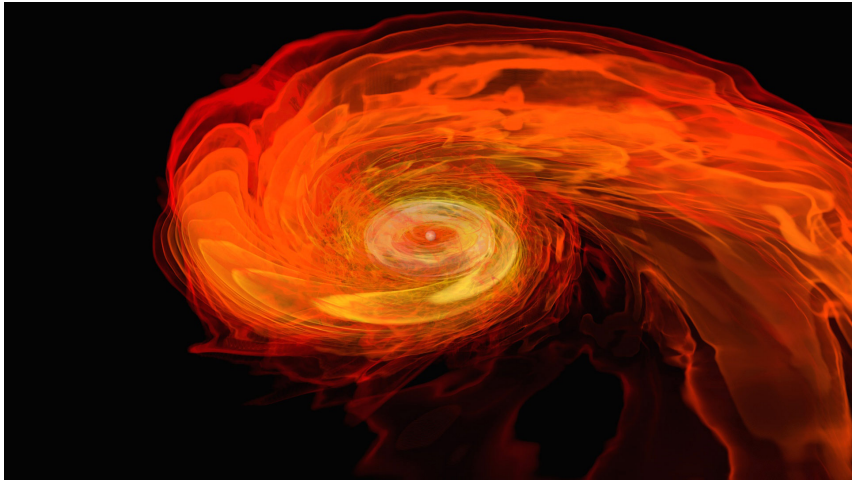
Event Horizon Telescope Collaboration

A binary BH simulation



N. Fischer/SXS/AEI

A binary NS simulation shortly after merger



NASA

Equations to solve

Many coupled PDEs

- ▷ hyperbolic equations:

$$\partial_t \mathbf{U} + \partial_i \mathbf{F}^i(\mathbf{U}) + \mathbf{B}^i \cdot \partial_i \mathbf{U} = \mathbf{S}(\mathbf{U})$$

Complicating features

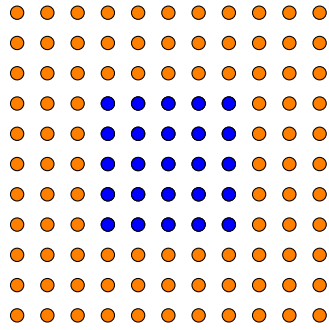
- ▷ Einstein's equations:
  - — choice of coordinates
  - — singularity inside BH
- ▷ GRMHD:
  - — turbulence & shocks
  - — neutrinos, nuclear reactions, ...

Solving the PDEs — current methods

Finite volume/difference methods

- ▷ represent solution with values at points
- ▷ overlapping cartesian grids
- ▷ shock-capturing schemes
- ▷ polynomial convergence
- ▷ “ghost zone” data from neighbors

Most binary BH, all matter simulations

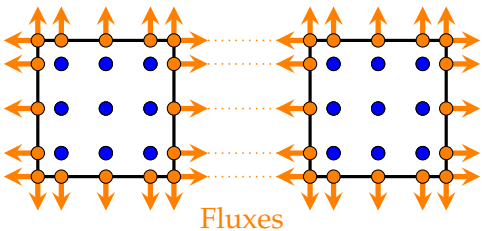


Ghost zones

Solving the PDEs — current methods

Spectral methods

- represent solution with basis functions
- geometrically-adapted grids
- smooth solutions only
- exponential convergence
- boundary data from neighbors



State of the art for binary BH simulations

Parallelism – current methods

MPI + some threading

- finite volume/difference codes scale to $\sim 10,000$ cores
- Spectral Einstein Code (SpEC)
  - ~ 1 spectral element per core
  - $\sim 100,000$ FV cells per core
  - scales to ~ 50 cores

Simulations take time

- binary BH $\sim$ week
- binary NS $\sim$ month

SpECTRE

SpECTRE: a next-generation code for relativistic astrophysics

- ▷ discontinuous Galerkin
- ▷ task-based parallelism
- ▷ `github.com/sxs-collaboration/spectre`

This talk

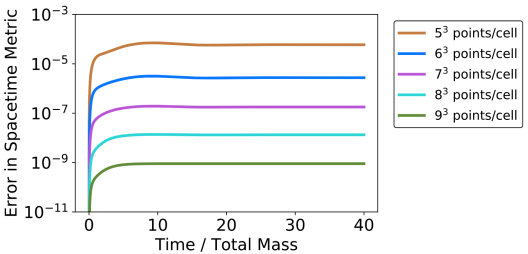
- ▷ methods for binary BHs
- ▷ preliminary binary BH results
- ▷ load balancing with Charm++

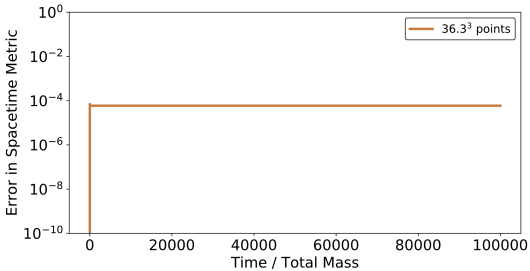
Not in this talk – improving hydrodynamics algorithms

Discontinuous Galerkin

- generalized spectral method
  - exponential convergence for smooth solutions
  - fall back to shock-capturing schemes where needed
- geometric flexibility
- nearest-neighbor boundary communication
- AMR and local timestepping

Code test: single BH





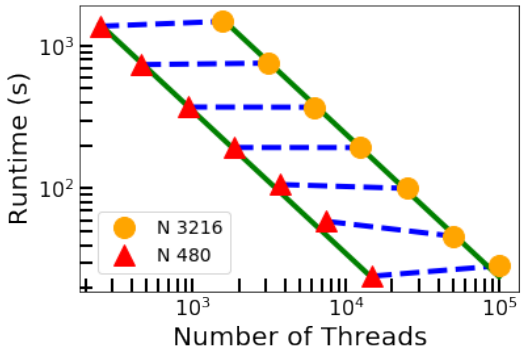
G. Lovelace

Code test: code scaling

Scaling on BlueWaters (NSCA, UIUC)

- green = strong scaling, fixed problem size
- blue = weak scaling, proportional problem size

(*) measurements made with a hydrodynamics evolution; predate an infrastructure rewrite in SpECTRE



Towards a binary BH evolution

- initial data
  - initial guess + solve elliptic constraint equations
  - in development
  - for now, use SpEC initial data
- PDE solver (discontinuous Galerkin + time stepper)
- strategy to keep the singularities off the grid

Keeping the singularities off the grid

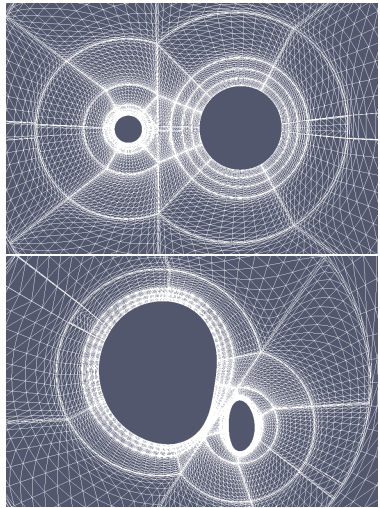
Excision

- ▷ cut out BH interior
- ▷ move excised regions with BH orbit

Control system

- ▷ measures BH positions and shapes
- ▷ updates time-dependent mappings to keep excised regions inside the BH

Time derivatives gain moving-mesh terms



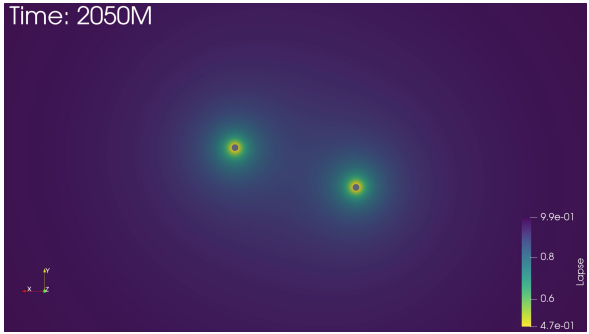
Towards a binary BH evolution

- initial data
  - initial guess + solve elliptic constraint equations
  - in development
  - for now, use SpEC initial data
- PDE solver (discontinuous Galerkin + time stepper)
- strategy to keep the singularities off the grid
  - in development
  - for now, use moving-mesh data from SpEC

Binary black hole evolution

Movie shows equatorial cut

- colored by lapse: spacetime curvature component associated with flow of time
- manually excised regions
- BHs follow excision regions for many orbits



SpECTRE use of Charm++

SpECTRE components

- DG elements = array chares
- data processing (IO, interpolations) = group and nodegroup chares
- measuring a BH position and shape = singleton chare
- computing gravitational waves = singleton chare

Evolution remains roughly in sync

- PDE structure imposes causality
- efficiency requires load balance

Load balancing in SpECTRE

Initial questions

- given a bad distribution of chares to nodes, can the LB improve it?
- given a good distribution (e.g., space-filling curve), will the LB preserve it?

Future work: balancing load and communications

Load balancing implementation

Initial implementation:

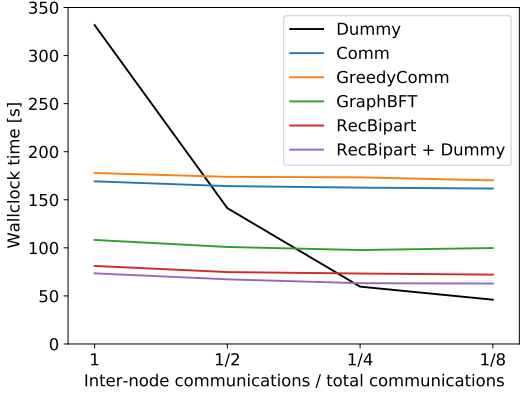
- add global synchronizations every N timesteps
- call `AtSync()`
- resume timestepping from `ResumeFromSync()`
- update registration in `pup::er` calls
  - array de-registers with group when packing
  - re-registers when unpacking

Load balancing results

A small test evolution

- 1024 array chares on 2 nodes
- $\sim$ 25 chares per proc

Best LB is within 20% of optimal



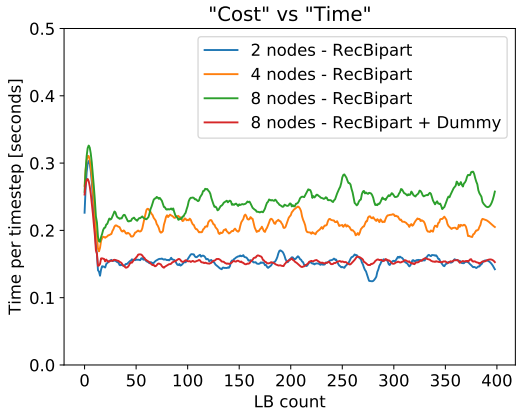
Load balancing results

Slowdown with larger problem size

- ▷ increase problem size and procs

Ongoing investigation

- ▷ normal scaling with graph size?
- ▷ is this Charm++ issue #2060?
- ▷ SpECTRE performance



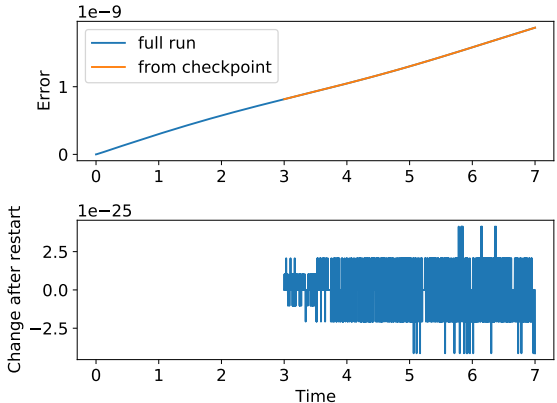
Checkpoint-restart results

Initial implementation:

- call `CkStartCheckpoint()` from global synchronization point

Works on same number of nodes

- future work: generalize



Wishlist after initial experiments

LB clarifications

- when to use which LB?
- how does each LB make its decisions? scale with graph complexity?

Checkpoint-restart clarifications

- what is order of initialization on restart?
- can group chare dependencies from program startup be enforced on restart?

Feature wishlist

- LB based on space-filling curve?
- checkpoint vs migration-aware `pup::er` will help optimize packing
  - avoid checkpointing caches to disk
  - tailor registration updates

Summary

- Future observations motivate improved simulations of binary mergers
- SpECTRE: improving algorithms and scalability
- Binary BH simulations
- Load balancing and checkpointing