Title: CACTUS as a Collaborative Tool for Discrete Quantum Gravity

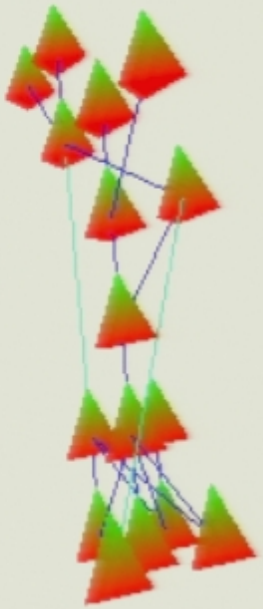
Date: Oct 29, 2004 04:40 PM

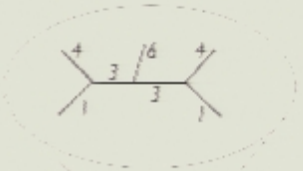
URL: http://pirsa.org/04100026

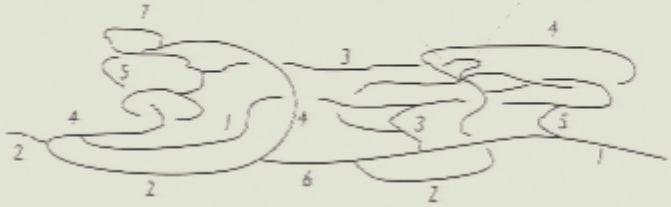
Abstract:

Hamilton

Cactus as a collaborative tool for discrete quantum gravity computation







David Rideout

Hamilton College

drideout@hamilton.edu

The Cactus Development Team

LSU

cactusmaint@cactuscode.org

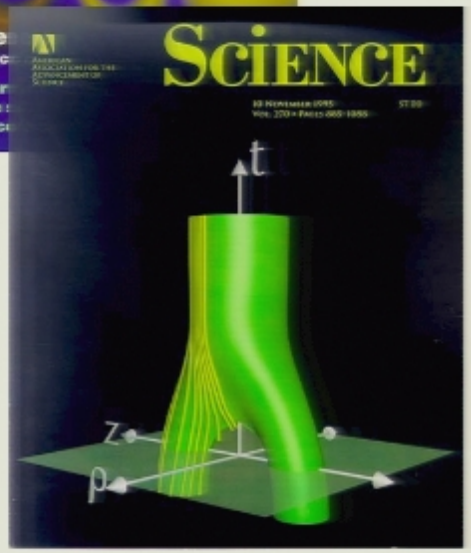
www.CactusCode.org

Outline

- Introduction to Cactus Framework
- Causal Set Introduction and API
  ("Application Programming Interface")
- LQG / Spin Network Computation
- Data structures and parallelization

What is Cactus?

CACTUS is a generic, freely available, modular, portable and manageable environment for collaboratively developing parallel, high-performance multi-dimensional simulations



Modularity: "Plug-and-play" Executables

Computational Thorns		Numerical Relativity Thorns	
PUGH	PAGH	ADMConstraint	IDAxiBrillBH
Carpet	HLL	PsiKadelia	Zorro
CartGrid3D	Cartoon2D	AHFinder	Extract
Time	Boundary	Maximal	ADM
EllSOR	EllBase	SimpleExcision	ADM_BSSN
IOFlexIO	IOASCII	FishEye	ConfHyp
IOHDF5	IOJpeg	IDAnalyticBH	BAM_Elliptic
IOUtil	IOBasic	LegoExcision	IDLinearWaves
HTTPD	HTTPDExtra	TGRPETSc	IDBrillWaves

ISCO with AMR ??

Faster elliptic solver ??

Excision

Why use a Framework?

- Separate physics from computational details:
  - (e.g. Make system, Parallellization, I/O, Elliptic Solvers, ...)
- Collaboration, modular frameworks enable sharing code
- Community building
- Leverage of other people's developments
  - physics
  - CS: vizualization, automatic parallelization, grid, ...

Why use a Framework?

- Separate physics from computational details:
  - (e.g. Make system, Parallellization, I/O, Elliptic Solvers, ...)
- Collaboration, modular frameworks enable sharing code
- Community building
- Leverage of other people's developments
  - physics
  - CS: vizualization, automatic parallelization, grid, ...

Why use a Framework?

- Separate physics from computational details:
  - (e.g. Make system, Parallellization, I/O, Elliptic Solvers, ...)
- Collaboration, modular frameworks enable sharing code
- Community building
- Leverage of other people's developments
  - physics
  - CS: vizualization, automatic parallelization, grid, ...

Why use a Framework?

- Separate physics from computational details:
  - (e.g. Make system, Parallellization, I/O, Elliptic Solvers, ...)
- Collaboration, modular frameworks enable sharing code
- Community building
- Leverage of other people's developments
  - physics
  - CS: vizualization, automatic parallelization, grid, ...

Why use a Framework?

- Separate physics from computational details:
  - (e.g. Make system, Parallellization, I/O, Elliptic Solvers, ...)
- Collaboration, modular frameworks enable sharing code
- Community building
- Leverage of other people's developments
  - physics
  - CS: vizualization, automatic parallelization, grid, ...

Why use a Framework?

- Separate physics from computational details:
  - (e.g. Make system, Parallellization, I/O, Elliptic Solvers, ...)
- Collaboration, modular frameworks enable sharing code
- Community building
- Leverage of other people's developments
  - physics
  - CS: vizualization, automatic parallelization, grid, ...

Why use a Framework?

- Separate physics from computational details:
  - (e.g. Make system, Parallellization, I/O, Elliptic Solvers, ...)
- Collaboration, modular frameworks enable sharing code
- Community building
- Leverage of other people's developments
  - physics
  - CS: vizualization, automatic parallelization, grid, ...

Why use a Framework?

- Separate physics from computational details:
  - (e.g. Make system, Parallellization, I/O, Elliptic Solvers, ...)
- Collaboration, modular frameworks enable sharing code
- Community building
- Leverage of other people's developments
  - physics
  - CS: vizualization, automatic parallelization, grid, ...

Why use a Framework?

- Separate physics from computational details:
  - (e.g. Make system, Parallellization, I/O, Elliptic Solvers, ...)
- Collaboration, modular frameworks enable sharing code
- Community building
- Leverage of other people's developments
  - physics
  - CS: vizualization, automatic parallelization, grid, ...

Why use a Framework?

- Separate physics from computational details:
  - (e.g. Make system, Parallellization, I/O, Elliptic Solvers, ...)
- Collaboration, modular frameworks enable sharing code
- Community building
- Leverage of other people's developments
  - physics
  - CS: vizualization, automatic parallelization, grid, ...

Why use a Framework?

- Separate physics from computational details:
  - (e.g. Make system, Parallellization, I/O, Elliptic Solvers, ...)
- Collaboration, modular frameworks enable sharing code
- Community building
- Leverage of other people's developments
  - physics
  - CS: vizualization, automatic parallelization, grid, ...

Why use a Framework?

- Separate physics from computational details:
  - (e.g. Make system, Parallellization, I/O, Elliptic Solvers, ...)
- Collaboration, modular frameworks enable sharing code
- Community building
- Leverage of other people's developments
  - physics
  - CS: vizualization, automatic parallelization, grid, ...

Why use a Framework?

- Separate physics from computational details:
  - (e.g. Make system, Parallellization, I/O, Elliptic Solvers, ...)
- Collaboration, modular frameworks enable sharing code
- Community building
- Leverage of other people's developments
  - physics
  - CS: vizualization, automatic parallelization, grid, ...

Why use a Framework?

- Separate physics from computational details:
  - (e.g. Make system, Parallellization, I/O, Elliptic Solvers, ...)
- Collaboration, modular frameworks enable sharing code
- Community building
- Leverage of other people's developments
  - physics
  - CS: vizualization, automatic parallelization, grid, ...

Why use a Framework?

- Separate physics from computational details:
  - (e.g. Make system, Parallellization, I/O, Elliptic Solvers, ...)
- Collaboration, modular frameworks enable sharing code
- Community building
- Leverage of other people's developments
  - physics
  - CS: vizualization, automatic parallelization, grid, ...

Deciding to use a Framework

- Does it support the languages/machines you want to use?
- Is it supported and developed?
- Is there documentation?
- Can it handle you want to do?
- Is it open source?
- How does it interact with other packages and tools (visualization, adaptive mesh refinement, linear algebra packages...)?

What Computational Physicists Need From Their Software ...

Primarily, it should enable the physics they want to do, and that means it must be:

- Collaborative
- Portable
- Large scale !
- Supported and developed
- Flexible
- Reproducible
- Have generic computational toolkits
- Incorporate other packages/technologies
- Easy to use/program

What Computational Physicists Need From Their Software ...

Primarily, it should enable the physics they want to do, and that means it must be:

- Collaborative
- Portable
- Large scale !
- Supported and developed
- Flexible
- Reproducible
- Have generic computational toolkits
- Incorporate other packages/technologies
- Easy to use/program

Collaborative

- Many different sub-parts of complex physics problems, e.g. Numerical Relativity:
  - initial data, evolution, horizons, waves, elliptic solvers, AMR, excision, shift conditions
- Need "modular" framework which connects (geographically distributed) experts in each of these areas
- Just one code version !!
- Also:
  - Need not know about the whole code
  - Short startup times for new people
  - Code reuse after people leave

Community: Share common code and experiences, validate results, compare

Example ThornList

ADM_BSSN (Miguel)

FishEye (John B)

Zorro (Manuela/Carlos)

AHFinder (Miguel)

PUGH (Tom)

FlexIO (John S)

IOFlexIO (Thomas R)

BAM_Elliptic (Bernd)

PsiKadelia (Steve)

Time (Gabrielle)

EllBase (Gerd)

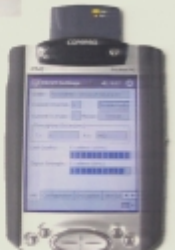
Einstein (All)

Portability

- Develop and run on many different architectures

  (laptop, workstations, supercomputers)
- Set up and get going quickly

  (new computer, visits, new job, wherever you get SUs)
- Make immediate use of free resources
- Portability crucial for "Grid Computing"



RECENT GROUP RESOURCES

Origin 2000 (NCSA)

IBM SP4 (NCSA)

Compaq Alpha (PSC)

Linux Cluster (Peyote)

Hitachi SR-8000 (LRZ)

Cray T3E (Garching)

SP4 (ZIB)

Institute Workstations

Linux Laptops

Ed's Mac

Very different architectures, operating systems, compilers and MPI implementations

Large Scale

- Typical BH run (but we want bigger!) needs 45GB of memory:
  - 171 Grid Functions
  - 400x400x200 grid
- Typical run makes 3000 iterations with 6000 Flops per grid point:
  - 600 TeraFlops !!
- Output of just one Grid Function at just one time step
  - 320 MB
  - (320 GB for 10GF every 50 time steps)
- One simulation takes longer than queue times
  - Need 10-50 hours
- Computing time is a valuable resource
  - One simulation: 2500 to 12500 SUs
  - Need to make each simulation count

Requirements

Parallelism

Optimization

Parallel/Fast IO, Data Management, Visualization

Checkpointing

Interactive monitoring, steering, visualization, portals

Generic Toolkits

Provide common functionality: quicker to develop applications, well tested, written by experts in an area. Continually added to and improved

Computational Toolkit:

- Drivers (Unigrid/FMR/AMR)
- Coordinates
- Boundary conditions
- I/O Methods
- Interpolation operators
- Reduction operators
- Elliptic solvers
- Web server
- Steering
- Notification
- Example applications

Einstein Toolkit:

- Evolution (ADM)
- Maximal slicing
- Analysis
  - Wave extraction
  - Scalar invariants
  - Apparent horizons
  - Constraints
- Initial data
  - Black holes
  - Gravitational waves

Easy to Use and Program

- Program in favorite language (C,C++,F90,F77)
- Hidden parallelism
- Computational Toolkits
- Good error, warning, info reporting
- Modularity !! Transparent interfaces with other modules
- Extensive parameter checking
- Work in the same way on different machines
- Interface with favorite visualization package
- Documentation

Easy to Use and Program

- Program in favorite language (C,C++,F90,F77)
- Hidden parallelism
- Computational Toolkits
- Good error, warning, info reporting
- Modularity !! Transparent interfaces with other modules
- Extensive parameter checking
- Work in the same way on different machines
- Interface with favorite visualization package
- Documentation

Easy to Use and Program

- Program in favorite language (C,C++,F90,F77)
- Hidden parallelism
- Computational Toolkits
- Good error, warning, info reporting
- Modularity !! Transparent interfaces with other modules
- Extensive parameter checking
- Work in the same way on different machines
- Interface with favorite visualization package
- Documentation

Cactus in a Nutshell

- Cactus acts as "main" routine of your code, it takes care of e.g. parallelism, IO, checkpointing, parameter file parsing, and provides computational infrastructure such as reduction operators, interpolators, coordinates, elliptic solvers, ...
- Everything Cactus "does" is contained in thorns (modules), which you need to compile-in. If you need to use interpolation, you need to find and add a thorn which does interpolation.
- It is very extensible, can add your own interpolators, IO methods etc.
- Not all the computational infrastructure you need is necessarily there, but hopefully all of the APIs etc are there to allow you to add what you need.
- Provides easy-to-use environment for collaborative, high-performance computing, from easy compilation on any machine, to

Cactus

Plug-In "Thorns" (modules)

driver

input/output

vizualization

parallel Monte Carlo

generalized percolation

multigrid

coordinates

extensible APIs

ANSI C

parameters

scheduling

Core "Flesh"

error handling

make system

grid variables

homology

remote steering

random sprinkling

Fortran/C/C++

equations of state

Your Physics !!

Your Computational Tools !!

black holes

Cactus

Plug-In "Thorns" (modules)

driver

input/output

vizualization

parallel Monte Carlo

generalized percolation

multigrid

coordinates

extensible APIs

ANSI C

parameters

scheduling

Core "Flesh"

error handling

make system

grid variables

homology

remote steering

random sprinkling

Fortran/C/C++

equations of state

Your Physics !!

Your Computational Tools !!

black holes

Cactus

Plug-In "Thorns" (modules)

driver

input/output

vizualization

parallel Monte Carlo

generalized percolation

multigrid

coordinates

extensible APIs

ANSI C

parameters

scheduling

Core "Flesh"

error handling

make system

grid variables

homology

remote steering

random sprinkling

Fortran/C/C++

equations of state

Your Physics !!

Your Computational Tools !!

black holes

Cactus

Plug-In "Thorns" (modules)

driver

input/output

vizualization

parallel Monte Carlo

generalized percolation

multigrid

coordinates

extensible APIs

ANSI C

parameters

scheduling

Core "Flesh"

error handling

make system

grid variables

homology

remote steering

random sprinkling

Fortran/C/C++

equations of state

Your Physics !!

Your Computational Tools !!

black holes

Cactus

Plug-In "Thorns" (modules)

driver

input/output

vizualization

parallel Monte Carlo

generalized percolation

multigrid

coordinates

extensible APIs

ANSI C

parameters

scheduling

Core "Flesh"

error handling

make system

grid variables

homology

remote steering

random sprinkling

Fortran/C/C++

equations of state

Your Physics !!

Your Computational Tools !!

black holes

Modularity: "Plug-and-play" Executables

Computational Thorns

PUGH	PAGH
Carpet	HLL
CartGrid3D	Cartoon2D
Time	Boundary
EllSOR	EllBase
IOFlexIO	IOASCII
IOHDF5	IOJpeg
IOUtil	IOBasic
HTTPD	HTTPDExtra

Numerical Relativity Thorns

ADMConstraint	IDAxiBrillBH
PsiKadelia	Zorro
AHFinder	Extract
Maximal	ADM
SimpleExcision	ADM_BSSN
FishEye	ConfHyp
IDAnalyticBH	BAM_Elliptic
LegoExcision	IDLinearWaves
	IDBrillWaves

ISCO Run

Supported Architectures

Machines	Operating Systems	Processors
PC	Linux	IA32, IA64
PC	Windows 2000/NT/XP (Cygwi	IA32
PC	OpenBSD/FreeBSD	IA32
Fujitsu VP		
HP/Compaq	OSF/Linux	Alpha
Cray T3E	Unicos	Alpha
HP Exemplar (V2500)	HP-UX	PA8500
Macintosh	MacOS X/Linux	PowerPC
NEC SX-5	SuperUX	
Sun	Solaris	Sparc II,Sparc III
IBM SP2	AIX	RS-6000
SGI Origin, O2	Irix	R8000, R10000, R12000
Hitachi SR8000F1	HIUX-MP	PowerPC

Cactus User Community

Using and Developing Physics Thorns

Numerical Relativity

LSU

Southampton

Wash U

RIKEN

Goddard

Penn State

Thessaloniki

Tuebingen

TAC

SISSA

Portsmouth

Brownsville

Pittsburg

AEI

EU Astrophysics Network

Austin

Benchmarking

UNAM

????

Garching

Plasma Physics (Princeton)

Other Applications

Chemical Engineering (U.Kansas)

Climate Modelling (NASA, Utrecht)

CFD (KISTI)

Bio-Informatics (Canada, Chicago)

Linear Algebra (Lebanon)

Quantum Gravity (Hamilton)

Prototype Apps (many CS projects)

Astrophysics (Zeus, LSU)

Causal Set Computation and API

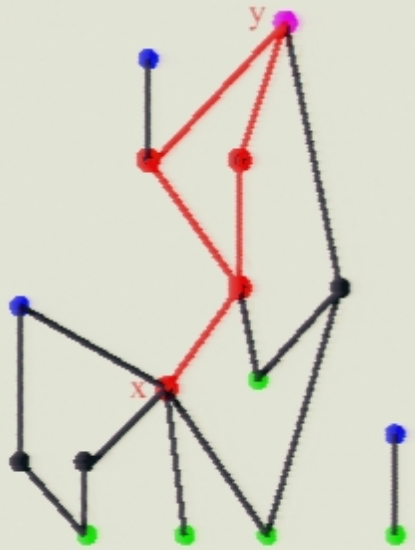
What are Causal Sets?

A Causal set (or causet) is a discrete causal order.

Properties of order relation $\prec$

- irreflexive ($x \not\prec x$) (say)
- transitive ($x \prec y$ and $y \prec z \Rightarrow x \prec z$)
- antisymmetric (**not** $x \prec y \prec x$)

Hasse diagram:



Interval: Alexandrov set of elements

$[x, y] = \{z \in \mathcal{C} | x \prec z \text{ and } z \prec y\}$

- locally finite:
  'Interval' [x,y] finite for any $x, y \in \mathcal{C}$

Cactus Causal Set API

What are the computational needs of Causal Set Quantum Gravity?

- many different sorts of computations — some huge, some fit on laptop
- in past each physicist must completely reinvent the wheel – from representing order to vizualization
- need unifying framework which allows physicists to build on each other's efforts

Cactus Causal Set API

What are the computational needs of Causal Set Quantum Gravity?

- many different sorts of computations — some huge, some fit on laptop
- in past each physicist must completely reinvent the wheel – from representing order to vizualization
- need unifying framework which allows physicists to build on each other's efforts

Cactus Causal Set API

What are the computational needs of Causal Set Quantum Gravity?

- many different sorts of computations — some huge, some fit on laptop
- in past each physicist must completely reinvent the wheel – from representing order to vizualization
- need unifying framework which allows physicists to build on each other's efforts

Cactus Causal Set API

What are the computational needs of Causal Set Quantum Gravity?

- many different sorts of computations — some huge, some fit on laptop
- in past each physicist must completely reinvent the wheel – from representing order to vizualization
- need unifying framework which allows physicists to build on each other's efforts

Cactus Causal Set API

What are the computational needs of Causal Set Quantum Gravity?

- many different sorts of computations — some huge, some fit on laptop
- in past each physicist must completely reinvent the wheel – from representing order to vizualization
- need unifying framework which allows physicists to build on each other's efforts

Cactus Causal Set API

What are the computational needs of Causal Set Quantum Gravity?

- many different sorts of computations — some huge, some fit on laptop
- in past each physicist must completely reinvent the wheel – from representing order to vizualization
- need unifying framework which allows physicists to build on each other's efforts

Cactus Causal Set API

What are the computational needs of Causal Set Quantum Gravity?

- many different sorts of computations — some huge, some fit on laptop
- in past each physicist must completely reinvent the wheel – from representing order to vizualization
- need unifying framework which allows physicists to build on each other's efforts

Cactus Causal Set API

What are the computational needs of Causal Set Quantum Gravity?

- many different sorts of computations — some huge, some fit on laptop
- in past each physicist must completely reinvent the wheel – from representing order to vizualization
- need unifying framework which allows physicists to build on each other's efforts

Cactus Causal Set API

What are the computational needs of Causal Set Quantum Gravity?

- many different sorts of computations — some huge, some fit on laptop
- in past each physicist must completely reinvent the wheel – from representing order to vizualization
- need unifying framework which allows physicists to build on each other's efforts

Cactus Causal Set API

What are the computational needs of Causal Set Quantum Gravity?

- many different sorts of computations — some huge, some fit on laptop
- in past each physicist must completely reinvent the wheel – from representing order to vizualization
- need unifying framework which allows physicists to build on each other's efforts

Dream: "The Causal Set Playground"

- create easy-to-use environment to test ideas involving causal sets
- code up idea and get results within hours
- simple, rapid testing of ideas
- seamless interface to other mathematical structures
- portability & parallelization of Cactus → easily scale up to supercomputers

first step in making this a reality ... the CausetBase API

Cactus Causal Set API

What are the computational needs of Causal Set Quantum Gravity?

- many different sorts of computations — some huge, some fit on laptop
- in past each physicist must completely reinvent the wheel – from representing order to vizualization
- need unifying framework which allows physicists to build on each other's efforts

Dream: "The Causal Set Playground"

- create easy-to-use environment to test ideas involving causal sets
- code up idea and get results within hours
- simple, rapid testing of ideas
- seamless interface to other mathematical structures
- portability & parallelization of Cactus → easily scale up to supercomputers

first step in making this a reality ... the CausetBase API

Cactus Causal Set API

What are the computational needs of Causal Set Quantum Gravity?

- many different sorts of computations — some huge, some fit on laptop
- in past each physicist must completely reinvent the wheel – from representing order to vizualization
- need unifying framework which allows physicists to build on each other's efforts

Dream: "The Causal Set Playground"

- create easy-to-use environment to test ideas involving causal sets
- code up idea and get results within hours
- simple, rapid testing of ideas
- seamless interface to other mathematical structures
- portability & parallelization of Cactus → easily scale up to supercomputers

first step in making this a reality ... the CausetBase API

Cactus Causal Set API

What are the computational needs of Causal Set Quantum Gravity?

- many different sorts of computations — some huge, some fit on laptop
- in past each physicist must completely reinvent the wheel – from representing order to vizualization
- need unifying framework which allows physicists to build on each other's efforts

Dream: "The Causal Set Playground"

- create easy-to-use environment to test ideas involving causal sets
- code up idea and get results within hours
- simple, rapid testing of ideas
- seamless interface to other mathematical structures
- portability & parallelization of Cactus → easily scale up to supercomputers

first step in making this a reality ... the CausetBase API

Cactus Causal Set API

What are the computational needs of Causal Set Quantum Gravity?

- many different sorts of computations — some huge, some fit on laptop
- in past each physicist must completely reinvent the wheel – from representing order to vizualization
- need unifying framework which allows physicists to build on each other's efforts

Dream: "The Causal Set Playground"

- create easy-to-use environment to test ideas involving causal sets
- code up idea and get results within hours
- simple, rapid testing of ideas
- seamless interface to other mathematical structures
- portability & parallelization of Cactus → easily scale up to supercomputers

first step in making this a reality ... the CausetBase API

Cactus Causal Set API

What are the computational needs of Causal Set Quantum Gravity?

- many different sorts of computations — some huge, some fit on laptop
- in past each physicist must completely reinvent the wheel – from representing order to vizualization
- need unifying framework which allows physicists to build on each other's efforts

Dream: "The Causal Set Playground"

- create easy-to-use environment to test ideas involving causal sets
- code up idea and get results within hours
- simple, rapid testing of ideas
- seamless interface to other mathematical structures
- portability & parallelization of Cactus → easily scale up to supercomputers

first step in making this a reality ... the CausetBase API

Cactus Causal Set API

What are the computational needs of Causal Set Quantum Gravity?

- many different sorts of computations — some huge, some fit on laptop
- in past each physicist must completely reinvent the wheel – from representing order to vizualization
- need unifying framework which allows physicists to build on each other's efforts

Dream: "The Causal Set Playground"

- create easy-to-use environment to test ideas involving causal sets
- code up idea and get results within hours
- simple, rapid testing of ideas
- seamless interface to other mathematical structures
- portability & parallelization of Cactus → easily scale up to supercomputers

first step in making this a reality ... the CausetBase API

Cactus Causal Set API

What are the computational needs of Causal Set Quantum Gravity?

- many different sorts of computations — some huge, some fit on laptop
- in past each physicist must completely reinvent the wheel – from representing order to vizualization
- need unifying framework which allows physicists to build on each other's efforts

Dream: "The Causal Set Playground"

- create easy-to-use environment to test ideas involving causal sets
- code up idea and get results within hours
- simple, rapid testing of ideas
- seamless interface to other mathematical structures
- portability & parallelization of Cactus → easily scale up to supercomputers

first step in making this a reality ... the CausetBase API

Cactus Causal Set API

What are the computational needs of Causal Set Quantum Gravity?

- many different sorts of computations — some huge, some fit on laptop
- in past each physicist must completely reinvent the wheel – from representing order to vizualization
- need unifying framework which allows physicists to build on each other's efforts

Dream: "The Causal Set Playground"

- create easy-to-use environment to test ideas involving causal sets
- code up idea and get results within hours
- simple, rapid testing of ideas
- seamless interface to other mathematical structures
- portability & parallelization of Cactus → easily scale up to supercomputers

first step in making this a reality ... the CausetBase API

Cactus Causal Set API

What are the computational needs of Causal Set Quantum Gravity?

- many different sorts of computations — some huge, some fit on laptop
- in past each physicist must completely reinvent the wheel – from representing order to vizualization
- need unifying framework which allows physicists to build on each other's efforts

Dream: "The Causal Set Playground"

- create easy-to-use environment to test ideas involving causal sets
- code up idea and get results within hours
- simple, rapid testing of ideas
- seamless interface to other mathematical structures
- portability & parallelization of Cactus → easily scale up to supercomputers

first step in making this a reality ... the CausetBase API

Cactus Causal Set API

What are the computational needs of Causal Set Quantum Gravity?

- many different sorts of computations — some huge, some fit on laptop
- in past each physicist must completely reinvent the wheel – from representing order to vizualization
- need unifying framework which allows physicists to build on each other's efforts

Dream: "The Causal Set Playground"

- create easy-to-use environment to test ideas involving causal sets
- code up idea and get results within hours
- simple, rapid testing of ideas
- seamless interface to other mathematical structures
- portability & parallelization of Cactus → easily scale up to supercomputers

first step in making this a reality ... the CausetBase API

Cactus Causal Set API

What are the computational needs of Causal Set Quantum Gravity?

- many different sorts of computations — some huge, some fit on laptop
- in past each physicist must completely reinvent the wheel – from representing order to vizualization
- need unifying framework which allows physicists to build on each other's efforts

Dream: "The Causal Set Playground"

- create easy-to-use environment to test ideas involving causal sets
- code up idea and get results within hours
- simple, rapid testing of ideas
- seamless interface to other mathematical structures
- portability & parallelization of Cactus → easily scale up to supercomputers

first step in making this a reality ... the CausetBase API

Cactus Causal Set API

What are the computational needs of Causal Set Quantum Gravity?

- many different sorts of computations — some huge, some fit on laptop
- in past each physicist must completely reinvent the wheel – from representing order to vizualization
- need unifying framework which allows physicists to build on each other's efforts

Dream: "The Causal Set Playground"

- create easy-to-use environment to test ideas involving causal sets
- code up idea and get results within hours
- simple, rapid testing of ideas
- seamless interface to other mathematical structures
- portability & parallelization of Cactus → easily scale up to supercomputers

first step in making this a reality ... the CausetBase API

Cactus Causal Set API

What are the computational needs of Causal Set Quantum Gravity?

- many different sorts of computations — some huge, some fit on laptop
- in past each physicist must completely reinvent the wheel – from representing order to vizualization
- need unifying framework which allows physicists to build on each other's efforts

Dream: "The Causal Set Playground"

- create easy-to-use environment to test ideas involving causal sets
- code up idea and get results within hours
- simple, rapid testing of ideas
- seamless interface to other mathematical structures
- portability & parallelization of Cactus → easily scale up to supercomputers

first step in making this a reality ... the CausetBase API

Cactus Causal Set API

What are the computational needs of Causal Set Quantum Gravity?

- many different sorts of computations — some huge, some fit on laptop
- in past each physicist must completely reinvent the wheel – from representing order to vizualization
- need unifying framework which allows physicists to build on each other's efforts

Dream: "The Causal Set Playground"

- create easy-to-use environment to test ideas involving causal sets
- code up idea and get results within hours
- simple, rapid testing of ideas
- seamless interface to other mathematical structures
- portability & parallelization of Cactus → easily scale up to supercomputers

first step in making this a reality ... the CausetBase API

The CausetBase API

- abstract notion of causal set — generic CausetBase implementation
- different implementations may have different advantages — languages, efficiency for certain classes of problems, ...
- researchers do not need to agree on these, only on the API!

Parallelization

- causets are 'non-local' → Parallelization by domain decomposition does not quite make sense
- stochastic dynamics → Monte-Carlo 'integration':
  - start with trivial 'decomposition'
  - SPRNG package generates parallel statistically independent seeds

The CausetBase API

- abstract notion of causal set — generic CausetBase implementation
- different implementations may have different advantages — languages, efficiency for certain classes of problems, ...
- researchers do not need to agree on these, only on the API!

Parallelization

- causets are 'non-local' $\longrightarrow$ Parallelization by domain decomposition does not quite make sense
- stochastic dynamics $\longrightarrow$ Monte-Carlo 'integration':
  - start with trivial 'decomposition'
  - SPRNG package generates parallel statistically independent seeds

The CausetBase API

- abstract notion of causal set — generic CausetBase implementation
- different implementations may have different advantages — languages, efficiency for certain classes of problems, ...
- researchers do not need to agree on these, only on the API!

Parallelization

- causets are 'non-local' $\longrightarrow$ Parallelization by domain decomposition does not quite make sense
- stochastic dynamics $\longrightarrow$ Monte-Carlo 'integration':
  - start with trivial 'decomposition'
  - SPRNG package generates parallel statistically independent seeds

The CausetBase API

- abstract notion of causal set — generic CausetBase implementation
- different implementations may have different advantages — languages, efficiency for certain classes of problems, ...
- researchers do not need to agree on these, only on the API!

Parallelization

- causets are 'non-local' $\longrightarrow$ Parallelization by domain decomposition does not quite make sense
- stochastic dynamics $\longrightarrow$ Monte-Carlo 'integration':
  - start with trivial 'decomposition'
  - SPRNG package generates parallel statistically independent seeds

The CausetBase API

- abstract notion of causal set — generic CausetBase implementation
- different implementations may have different advantages — languages, efficiency for certain classes of problems, ...
- researchers do not need to agree on these, only on the API!

Parallelization

- causets are 'non-local' $\longrightarrow$ Parallelization by domain decomposition does not quite make sense
- stochastic dynamics $\longrightarrow$ Monte-Carlo 'integration':
  - start with trivial 'decomposition'
  - SPRNG package generates parallel statistically independent seeds

The CausetBase API

- abstract notion of causal set — generic CausetBase implementation
- different implementations may have different advantages — languages, efficiency for certain classes of problems, ...
- researchers do not need to agree on these, only on the API!

Parallelization

- causets are 'non-local' $\longrightarrow$ Parallelization by domain decomposition does not quite make sense
- stochastic dynamics $\longrightarrow$ Monte-Carlo 'integration':
  - start with trivial 'decomposition'
  - SPRNG package generates parallel statistically independent seeds

The CausetBase API

- abstract notion of causal set — generic CausetBase implementation
- different implementations may have different advantages — languages, efficiency for certain classes of problems, ...
- researchers do not need to agree on these, only on the API!

Parallelization

- causets are 'non-local' → Parallelization by domain decomposition does not quite make sense
- stochastic dynamics → Monte-Carlo 'integration':
  - start with trivial 'decomposition'
  - SPRNG package generates parallel statistically independent seeds

The CausetBase API

- abstract notion of causal set — generic CausetBase implementation
- different implementations may have different advantages — languages, efficiency for certain classes of problems, ...
- researchers do not need to agree on these, only on the API!

Parallelization

- causets are 'non-local' $\longrightarrow$ Parallelization by domain decomposition does not quite make sense
- stochastic dynamics $\longrightarrow$ Monte-Carlo 'integration':
  - start with trivial 'decomposition'
  - SPRNG package generates parallel statistically independent seeds

The CausetBase API

- abstract notion of causal set — generic CausetBase implementation
- different implementations may have different advantages — languages, efficiency for certain classes of problems, ...
- researchers do not need to agree on these, only on the API!

Parallelization

- causets are 'non-local' $\longrightarrow$ Parallelization by domain decomposition does not quite make sense
- stochastic dynamics $\longrightarrow$ Monte-Carlo 'integration':
  - start with trivial 'decomposition'
  - SPRNG package generates parallel statistically independent seeds

The CausetBase API

- abstract notion of causal set — generic CausetBase implementation
- different implementations may have different advantages — languages, efficiency for certain classes of problems, ...
- researchers do not need to agree on these, only on the API!

Parallelization

- causets are 'non-local' $\longrightarrow$ Parallelization by domain decomposition does not quite make sense
- stochastic dynamics $\longrightarrow$ Monte-Carlo 'integration':
  - start with trivial 'decomposition'
  - SPRNG package generates parallel statistically independent seeds

The CausetBase API

- abstract notion of causal set — generic CausetBase implementation
- different implementations may have different advantages — languages, efficiency for certain classes of problems, ...
- researchers do not need to agree on these, only on the API!

Parallelization

- causets are 'non-local' $\longrightarrow$ Parallelization by domain decomposition does not quite make sense
- stochastic dynamics $\longrightarrow$ Monte-Carlo 'integration':
  - start with trivial 'decomposition'
  - SPRNG package generates parallel statistically independent seeds

The CausetBase API

- abstract notion of causal set — generic CausetBase implementation
- different implementations may have different advantages — languages, efficiency for certain classes of problems, ...
- researchers do not need to agree on these, only on the API!

Parallelization

- causets are 'non-local' $\longrightarrow$ Parallelization by domain decomposition does not quite make sense
- stochastic dynamics $\longrightarrow$ Monte-Carlo 'integration':
  - start with trivial 'decomposition'
  - SPRNG package generates parallel statistically independent seeds

Some details of the CausetBase API and its implementation

BinaryCausets

- order represented as (lower-diagonal) array of bits
- 1D CCTK_INT grid array signed_causet[]
- grid variables (data representation) are private
- DECLARE_CAUSET(GH) macro — casts data pointers on GH to unsigned ints

Some details of the CausetBase API and its implementation

BinaryCausets

- order represented as (lower-diagonal) array of bits
- 1D CCTK_INT grid array signed_causet[]
- grid variables (data representation) are private
- DECLARE_CAUSET(GH) macro — casts data pointers on GH to unsigned ints

Some details of the CausetBase API and its implementation

BinaryCausets

- order represented as (lower-diagonal) array of bits
- 1D CCTK_INT grid array signed_causet[]
- grid variables (data representation) are private
- DECLARE_CAUSET(GH) macro — casts data pointers on GH to unsigned ints

Some details of the CausetBase API and its implementation

BinaryCausets

- order represented as (lower-diagonal) array of bits
- 1D CCTK_INT grid array signed_causet[]
- grid variables (data representation) are private
- DECLARE_CAUSET(GH) macro — casts data pointers on GH to unsigned ints

Sets of elements

- macros and aliased functions to manipluate sets of elements
- identified by integer handles
- handles stored in grid scalars, to allow inter thorn communication
- internal representation of sets is unspecified by API
- to use macros:

  ```
  uses include header: SetsBase_C.h
  ```

- start of each function needs:

  ```
  DECLARE_SETS(GH)
  ```

  to initialize data structures

- BinaryCauset uses array of bits to store sets

The CausalSets arrangement

AntichainEvol	'evolves' an antichain
BHEntropy	computes entropy by counting 'links crossing horizon'
BinaryCauset	implements the CausetBase API
CFlatSprinkle	causets which faithfully embed into regions of conformally flat spacetimes
	provides an IO method which displays the embedding
	one can use different spatial topologies
	can boost the embedding
CausetIO	displays causet, currently just a text display on stdout
GeneralizedPercolation	will simulate generic causet dynamics
Nerve	computes simplicial complex from maximal antichain
RandomAntichain	generates a random maximal antichain
SimpleCauset	a (potential) alternate implementation of CausetBase
TransitivePercolation	simulates transitive percolation dynamics

so far...

The MonteCarlo arrangement

Monte Carlo computations consume a large fraction of the world's supercomputing resources.

RandomNumbers	'Base' thorn which provides random numbers to 'application' thorns manages parallelism of Monte Carlo computation
SPRNG	provides access to SPRNG library, for generating independent parallel pseudorandom numbers
Statistics	computes various statistical quantities

Sample computation: Sprinkling

- select points at random in spacetime manifold
- deduce causal relations among sprinkled points
- hard in non-conformally flat spacetimes . . . use null surface evolvers from numerical relativity!

The MonteCarlo arrangement

Monte Carlo computations consume a large fraction of the world's supercomputing resources.

RandomNumbers	'Base' thorn which provides random numbers to 'application' thorns manages parallelism of Monte Carlo computation
SPRNG	provides access to SPRNG library, for generating independent parallel pseudorandom numbers
Statistics	computes various statistical quantities

Sample computation: Sprinkling

- select points at random in spacetime manifold
- deduce causal relations among sprinkled points
- hard in non-conformally flat spacetimes . . . use null surface evolvers from numerical relativity!

The MonteCarlo arrangement

Monte Carlo computations consume a large fraction of the world's supercomputing resources.

RandomNumbers	'Base' thorn which provides random numbers to 'application' thorns manages parallelism of Monte Carlo computation
SPRNG	provides access to SPRNG library, for generating independent parallel pseudorandom numbers
Statistics	computes various statistical quantities

Sample computation: Sprinkling

- select points at random in spacetime manifold
- deduce causal relations among sprinkled points
- hard in non-conformally flat spacetimes . . . use null surface evolvers from numerical relativity!

Spin Networks of Loop Quantum Gravity

- basic variables of Loop Quantum Gravity is spin network SU(2) connection $A_a^i(x)$ and a densitized triad $\tilde{E}_i^a$
- basis of space of gauge invariant states
- closed graph γ embedded in spatial manifold Σ (with some extra knot structure)
- possesses finite number of oriented edges
- each edge labeled by irreducible representation j_i of SU(2)
- at each vertex there is an 'intertwining operator' from tensor product of representations of incoming edges to those of outgoing edges
- can think of spin network as dual 1-skeleton of simplicial triangulation of Σ

- compute volumes at vertices; effects of gauge invariance conditions
- modularity of Cactus $\Rightarrow$ can conceivably intermix computations involving various approaches

Spin Networks of Loop Quantum Gravity

- basic variables of Loop Quantum Gravity is spin network SU(2) connection $A_a^i(x)$ and a densitized triad $\tilde{E}_i^a$
- basis of space of gauge invariant states
- closed graph γ embedded in spatial manifold Σ (with some extra knot structure)
- possesses finite number of oriented edges
- each edge labeled by irreducible representation j_i of SU(2)
- at each vertex there is an 'intertwining operator' from tensor product of representations of incoming edges to those of outgoing edges
- can think of spin network as dual 1-skeleton of simplicial triangulation of Σ

- compute volumes at vertices; effects of gauge invariance conditions
- modularity of Cactus $\Rightarrow$ can conceivably intermix computations involving various approaches

Spin Networks of Loop Quantum Gravity

- basic variables of Loop Quantum Gravity is spin network SU(2) connection $A_a^i(x)$ and a densitized triad $\tilde{E}_i^a$
- basis of space of gauge invariant states
- closed graph γ embedded in spatial manifold Σ (with some extra knot structure)
- possesses finite number of oriented edges
- each edge labeled by irreducible representation j_i of SU(2)
- at each vertex there is an 'intertwining operator' from tensor product of representations of incoming edges to those of outgoing edges
- can think of spin network as dual 1-skeleton of simplicial triangulation of Σ

- compute volumes at vertices; effects of gauge invariance conditions
- modularity of Cactus $\Rightarrow$ can conceivably intermix computations involving various approaches

Spin Networks of Loop Quantum Gravity

- basic variables of Loop Quantum Gravity is spin network SU(2) connection $A_a^i(x)$ and a densitized triad $\tilde{E}_i^a$
- basis of space of gauge invariant states
- closed graph γ embedded in spatial manifold Σ (with some extra knot structure)
- possesses finite number of oriented edges
- each edge labeled by irreducible representation j_i of SU(2)
- at each vertex there is an 'intertwining operator' from tensor product of representations of incoming edges to those of outgoing edges
- can think of spin network as dual 1-skeleton of simplicial triangulation of Σ

- compute volumes at vertices; effects of gauge invariance conditions
- modularity of Cactus $\Rightarrow$ can conceivably intermix computations involving various approaches

Spin Networks of Loop Quantum Gravity

- basic variables of Loop Quantum Gravity is spin network SU(2) connection $A_a^i(x)$ and a densitized triad $\tilde{E}_i^a$
- basis of space of gauge invariant states
- closed graph γ embedded in spatial manifold Σ (with some extra knot structure)
- possesses finite number of oriented edges
- each edge labeled by irreducible representation j_i of SU(2)
- at each vertex there is an 'intertwining operator' from tensor product of representations of incoming edges to those of outgoing edges
- can think of spin network as dual 1-skeleton of simplicial triangulation of Σ

- compute volumes at vertices; effects of gauge invariance conditions
- modularity of Cactus $\Rightarrow$ can conceivably intermix computations involving various approaches

Spin Networks of Loop Quantum Gravity

- basic variables of Loop Quantum Gravity is spin network SU(2) connection $A_a^i(x)$ and a densitized triad $\tilde{E}_i^a$
- basis of space of gauge invariant states
- closed graph γ embedded in spatial manifold Σ (with some extra knot structure)
- possesses finite number of oriented edges
- each edge labeled by irreducible representation j_i of SU(2)
- at each vertex there is an 'intertwining operator' from tensor product of representations of incoming edges to those of outgoing edges
- can think of spin network as dual 1-skeleton of simplicial triangulation of Σ

- compute volumes at vertices; effects of gauge invariance conditions
- modularity of Cactus $\Rightarrow$ can conceivably intermix computations involving various approaches

Spin Networks of Loop Quantum Gravity

- basic variables of Loop Quantum Gravity is spin network SU(2) connection $A_a^i(x)$ and a densitized triad $\tilde{E}_i^a$
- basis of space of gauge invariant states
- closed graph γ embedded in spatial manifold Σ (with some extra knot structure)
- possesses finite number of oriented edges
- each edge labeled by irreducible representation j_i of SU(2)
- at each vertex there is an 'intertwining operator' from tensor product of representations of incoming edges to those of outgoing edges
- can think of spin network as dual 1-skeleton of simplicial triangulation of Σ

- compute volumes at vertices; effects of gauge invariance conditions
- modularity of Cactus $\Rightarrow$ can conceivably intermix computations involving various approaches

Spin Networks of Loop Quantum Gravity

- basic variables of Loop Quantum Gravity is spin network SU(2) connection $A_a^i(x)$ and a densitized triad $\tilde{E}_i^a$
- basis of space of gauge invariant states
- closed graph γ embedded in spatial manifold Σ (with some extra knot structure)
- possesses finite number of oriented edges
- each edge labeled by irreducible representation j_i of SU(2)
- at each vertex there is an 'intertwining operator' from tensor product of representations of incoming edges to those of outgoing edges
- can think of spin network as dual 1-skeleton of simplicial triangulation of Σ

- compute volumes at vertices; effects of gauge invariance conditions
- modularity of Cactus $\Rightarrow$ can conceivably intermix computations involving various approaches

Spin Networks of Loop Quantum Gravity

- basic variables of Loop Quantum Gravity is spin network SU(2) connection $A_a^i(x)$ and a densitized triad $\tilde{E}_i^a$
- basis of space of gauge invariant states
- closed graph γ embedded in spatial manifold Σ (with some extra knot structure)
- possesses finite number of oriented edges
- each edge labeled by irreducible representation j_i of SU(2)
- at each vertex there is an 'intertwining operator' from tensor product of representations of incoming edges to those of outgoing edges
- can think of spin network as dual 1-skeleton of simplicial triangulation of Σ

- compute volumes at vertices; effects of gauge invariance conditions
- modularity of Cactus $\Rightarrow$ can conceivably intermix computations involving various approaches

Spin Networks of Loop Quantum Gravity

- basic variables of Loop Quantum Gravity is spin network SU(2) connection $A_a^i(x)$ and a densitized triad $\tilde{E}_i^a$
- basis of space of gauge invariant states
- closed graph γ embedded in spatial manifold Σ (with some extra knot structure)
- possesses finite number of oriented edges
- each edge labeled by irreducible representation j_i of SU(2)
- at each vertex there is an 'intertwining operator' from tensor product of representations of incoming edges to those of outgoing edges
- can think of spin network as dual 1-skeleton of simplicial triangulation of Σ

- compute volumes at vertices; effects of gauge invariance conditions
- modularity of Cactus $\Rightarrow$ can conceivably intermix computations involving various approaches

Spin Networks of Loop Quantum Gravity

- basic variables of Loop Quantum Gravity is spin network SU(2) connection $A_a^i(x)$ and a densitized triad $\tilde{E}_i^a$
- basis of space of gauge invariant states
- closed graph γ embedded in spatial manifold Σ (with some extra knot structure)
- possesses finite number of oriented edges
- each edge labeled by irreducible representation j_i of SU(2)
- at each vertex there is an 'intertwining operator' from tensor product of representations of incoming edges to those of outgoing edges
- can think of spin network as dual 1-skeleton of simplicial triangulation of Σ

- compute volumes at vertices; effects of gauge invariance conditions
- modularity of Cactus $\Rightarrow$ can conceivably intermix computations involving various approaches

Data structures and parallelization

Data Structures

incidence number: 'typical' number of edges incident on a vertex of a graph (or relations on a poset element)

incidence number $O(1)$

- Represent with linked list type data structure, use pointers to neighboring structures
- Can store data on this by just adding variables to the structures
- Data on edges can be easily stored in separate edge structures

incidence number $O(N^{\alpha}), 1 > \alpha > 0$

- "Physical" causal sets of this type
- Better represented with incidence matricies
- Store data on incidence matrix simply by having 'array of nodes'
- Data on edges cause a problem — use linked list type structure

Data structures and parallelization

Data Structures

incidence number: 'typical' number of edges incident on a vertex of a graph (or relations on a poset element)

incidence number $O(1)$

- Represent with linked list type data structure, use pointers to neighboring structures
- Can store data on this by just adding variables to the structures
- Data on edges can be easily stored in separate edge structures

incidence number $O(N^{\alpha}), 1 > \alpha > 0$

- "Physical" causal sets of this type
- Better represented with incidence matricies
- Store data on incidence matrix simply by having 'array of nodes'
- Data on edges cause a problem — use linked list type structure

Data structures and parallelization

Data Structures

incidence number: 'typical' number of edges incident on a vertex of a graph (or relations on a poset element)

incidence number $O(1)$

- Represent with linked list type data structure, use pointers to neighboring structures
- Can store data on this by just adding variables to the structures
- Data on edges can be easily stored in separate edge structures

incidence number $O(N^{\alpha}), 1 > \alpha > 0$

- "Physical" causal sets of this type
- Better represented with incidence matricies
- Store data on incidence matrix simply by having 'array of nodes'
- Data on edges cause a problem — use linked list type structure

Data structures and parallelization

Data Structures

incidence number: 'typical' number of edges incident on a vertex of a graph (or relations on a poset element)

incidence number $O(1)$

- Represent with linked list type data structure, use pointers to neighboring structures
- Can store data on this by just adding variables to the structures
- Data on edges can be easily stored in separate edge structures

incidence number $O(N^{\alpha}), 1 > \alpha > 0$

- "Physical" causal sets of this type
- Better represented with incidence matricies
- Store data on incidence matrix simply by having 'array of nodes'
- Data on edges cause a problem — use linked list type structure

Data structures and parallelization

Data Structures

incidence number: 'typical' number of edges incident on a vertex of a graph (or relations on a poset element)

incidence number $O(1)$

- Represent with linked list type data structure, use pointers to neighboring structures
- Can store data on this by just adding variables to the structures
- Data on edges can be easily stored in separate edge structures

incidence number $O(N^{\alpha}), 1 > \alpha > 0$

- "Physical" causal sets of this type
- Better represented with incidence matricies
- Store data on incidence matrix simply by having 'array of nodes'
- Data on edges cause a problem — use linked list type structure

Data structures and parallelization

Data Structures

incidence number: 'typical' number of edges incident on a vertex of a graph (or relations on a poset element)

incidence number $O(1)$

- Represent with linked list type data structure, use pointers to neighboring structures
- Can store data on this by just adding variables to the structures
- Data on edges can be easily stored in separate edge structures

incidence number $O(N^{\alpha}), 1 > \alpha > 0$

- "Physical" causal sets of this type
- Better represented with incidence matricies
- Store data on incidence matrix simply by having 'array of nodes'
- Data on edges cause a problem — use linked list type structure

Data structures and parallelization

Data Structures

incidence number: 'typical' number of edges incident on a vertex of a graph (or relations on a poset element)

incidence number $O(1)$

- Represent with linked list type data structure, use pointers to neighboring structures
- Can store data on this by just adding variables to the structures
- Data on edges can be easily stored in separate edge structures

incidence number $O(N^{\alpha}), 1 > \alpha > 0$

- "Physical" causal sets of this type
- Better represented with incidence matricies
- Store data on incidence matrix simply by having 'array of nodes'
- Data on edges cause a problem — use linked list type structure

Data structures and parallelization

Data Structures

incidence number: 'typical' number of edges incident on a vertex of a graph (or relations on a poset element)

incidence number $O(1)$

- Represent with linked list type data structure, use pointers to neighboring structures
- Can store data on this by just adding variables to the structures
- Data on edges can be easily stored in separate edge structures

incidence number $O(N^{\alpha}), 1 > \alpha > 0$

- "Physical" causal sets of this type
- Better represented with incidence matricies
- Store data on incidence matrix simply by having 'array of nodes'
- Data on edges cause a problem — use linked list type structure

Data structures and parallelization

Data Structures

incidence number: 'typical' number of edges incident on a vertex of a graph (or relations on a poset element)

incidence number $O(1)$

- Represent with linked list type data structure, use pointers to neighboring structures
- Can store data on this by just adding variables to the structures
- Data on edges can be easily stored in separate edge structures

incidence number $O(N^{\alpha}), 1 > \alpha > 0$

- "Physical" causal sets of this type
- Better represented with incidence matricies
- Store data on incidence matrix simply by having 'array of nodes'
- Data on edges cause a problem — use linked list type structure

Data structures and parallelization

Data Structures

incidence number: 'typical' number of edges incident on a vertex of a graph (or relations on a poset element)

incidence number $O(1)$

- Represent with linked list type data structure, use pointers to neighboring structures
- Can store data on this by just adding variables to the structures
- Data on edges can be easily stored in separate edge structures

incidence number $O(N^{\alpha}), 1 > \alpha > 0$

- "Physical" causal sets of this type
- Better represented with incidence matricies
- Store data on incidence matrix simply by having 'array of nodes'
- Data on edges cause a problem — use linked list type structure

Parallelization

- Domain decomposition works for 'incidence number $O(1)$ graphs'
- Can always do Monte Carlo parallelization
- shared memory and hybrid architectures can be useful for 'incidence number $O(N^{\alpha})$ graphs'

Conclusions

Cactus can make discrete quantum gravity computations much easier

Provides:

- language independence & modularity allows code sharing, code reuse, specialization
  $\longrightarrow$ build on each others' efforts
- automatic parallelization of Monte Carlo calculations, and '$O(1)$ graphs'
  $\longrightarrow$ greatly simplify programming task
- borrowing codes / tools from other fields, such as numerical relativity
- begins to make high performance computation easily accesible to theoretical physicist

Conclusions

Cactus can make discrete quantum gravity computations much easier

Provides:

- language independence & modularity allows code sharing, code reuse, specialization
  $\longrightarrow$ build on each others' efforts
- automatic parallelization of Monte Carlo calculations, and '$O(1)$ graphs'
  $\longrightarrow$ greatly simplify programming task
- borrowing codes / tools from other fields, such as numerical relativity
- begins to make high performance computation easily accesible to theoretical physicist

Conclusions

Cactus can make discrete quantum gravity computations much easier

Provides:

- language independence & modularity allows code sharing, code reuse, specialization
  $\longrightarrow$ build on each others' efforts
- automatic parallelization of Monte Carlo calculations, and '$O(1)$ graphs'
  $\longrightarrow$ greatly simplify programming task
- borrowing codes / tools from other fields, such as numerical relativity
- begins to make high performance computation easily accesible to theoretical physicist

Conclusions

Cactus can make discrete quantum gravity computations much easier

Provides:

- language independence & modularity allows code sharing, code reuse, specialization
  $\longrightarrow$ build on each others' efforts
- automatic parallelization of Monte Carlo calculations, and '$O(1)$ graphs'
  $\longrightarrow$ greatly simplify programming task
- borrowing codes / tools from other fields, such as numerical relativity
- begins to make high performance computation easily accesible to theoretical physicist

Conclusions

Cactus can make discrete quantum gravity computations much easier

Provides:

- language independence & modularity allows code sharing, code reuse, specialization
  $\longrightarrow$ build on each others' efforts
- automatic parallelization of Monte Carlo calculations, and '$O(1)$ graphs'
  $\longrightarrow$ greatly simplify programming task
- borrowing codes / tools from other fields, such as numerical relativity
- begins to make high performance computation easily accesible to theoretical physicist

Conclusions

Cactus can make discrete quantum gravity computations much easier

Provides:

- language independence & modularity allows code sharing, code reuse, specialization
  $\longrightarrow$ build on each others' efforts
- automatic parallelization of Monte Carlo calculations, and '$O(1)$ graphs'
  $\longrightarrow$ greatly simplify programming task
- borrowing codes / tools from other fields, such as numerical relativity
- begins to make high performance computation easily accesible to theoretical physicist

Conclusions

Cactus can make discrete quantum gravity computations much easier

Provides:

- language independence & modularity allows code sharing, code reuse, specialization
  $\longrightarrow$ build on each others' efforts
- automatic parallelization of Monte Carlo calculations, and '$O(1)$ graphs'
  $\longrightarrow$ greatly simplify programming task
- borrowing codes / tools from other fields, such as numerical relativity
- begins to make high performance computation easily accesible to theoretical physicist

Conclusions

Cactus can make discrete quantum gravity computations much easier

Provides:

- language independence & modularity allows code sharing, code reuse, specialization
  $\longrightarrow$ build on each others' efforts
- automatic parallelization of Monte Carlo calculations, and '$O(1)$ graphs'
  $\longrightarrow$ greatly simplify programming task
- borrowing codes / tools from other fields, such as numerical relativity
- begins to make high performance computation easily accesible to theoretical physicist

Conclusions

Cactus can make discrete quantum gravity computations much easier

Provides:

- language independence & modularity allows code sharing, code reuse, specialization
  $\longrightarrow$ build on each others' efforts
- automatic parallelization of Monte Carlo calculations, and '$O(1)$ graphs'
  $\longrightarrow$ greatly simplify programming task
- borrowing codes / tools from other fields, such as numerical relativity
- begins to make high performance computation easily accesible to theoretical physicist

Conclusions

Cactus can make discrete quantum gravity computations much easier

Provides:

- language independence & modularity allows code sharing, code reuse, specialization
  $\longrightarrow$ build on each others' efforts
- automatic parallelization of Monte Carlo calculations, and '$O(1)$ graphs'
  $\longrightarrow$ greatly simplify programming task
- borrowing codes / tools from other fields, such as numerical relativity
- begins to make high performance computation easily accesible to theoretical physicist

Conclusions

Cactus can make discrete quantum gravity computations much easier

Provides:

- language independence & modularity allows code sharing, code reuse, specialization
  $\longrightarrow$ build on each others' efforts
- automatic parallelization of Monte Carlo calculations, and '$O(1)$ graphs'
  $\longrightarrow$ greatly simplify programming task
- borrowing codes / tools from other fields, such as numerical relativity
- begins to make high performance computation easily accesible to theoretical physicist

Conclusions

Cactus can make discrete quantum gravity computations much easier

Provides:

- language independence & modularity allows code sharing, code reuse, specialization
  $\longrightarrow$ build on each others' efforts
- automatic parallelization of Monte Carlo calculations, and '$O(1)$ graphs'
  $\longrightarrow$ greatly simplify programming task
- borrowing codes / tools from other fields, such as numerical relativity
- begins to make high performance computation easily accesible to theoretical physicist

Conclusions

Cactus can make discrete quantum gravity computations much easier

Provides:

- language independence & modularity allows code sharing, code reuse, specialization
  $\longrightarrow$ build on each others' efforts
- automatic parallelization of Monte Carlo calculations, and '$O(1)$ graphs'
  $\longrightarrow$ greatly simplify programming task
- borrowing codes / tools from other fields, such as numerical relativity
- begins to make high performance computation easily accesible to theoretical physicist

Conclusions

Cactus can make discrete quantum gravity computations much easier

Provides:

- language independence & modularity allows code sharing, code reuse, specialization
  $\longrightarrow$ build on each others' efforts
- automatic parallelization of Monte Carlo calculations, and '$O(1)$ graphs'
  $\longrightarrow$ greatly simplify programming task
- borrowing codes / tools from other fields, such as numerical relativity
- begins to make high performance computation easily accesible to theoretical physicist

Conclusions

Cactus can make discrete quantum gravity computations much easier

Provides:

- language independence & modularity allows code sharing, code reuse, specialization
  $\longrightarrow$ build on each others' efforts
- automatic parallelization of Monte Carlo calculations, and '$O(1)$ graphs'
  $\longrightarrow$ greatly simplify programming task
- borrowing codes / tools from other fields, such as numerical relativity
- begins to make high performance computation easily accesible to theoretical physicist

Conclusions

Cactus can make discrete quantum gravity computations much easier

Provides:

- language independence & modularity allows code sharing, code reuse, specialization
  $\longrightarrow$ build on each others' efforts
- automatic parallelization of Monte Carlo calculations, and '$O(1)$ graphs'
  $\longrightarrow$ greatly simplify programming task
- borrowing codes / tools from other fields, such as numerical relativity
- begins to make high performance computation easily accesible to theoretical physicist

Conclusions

Cactus can make discrete quantum gravity computations much easier

Provides:

- language independence & modularity allows code sharing, code reuse, specialization
  $\longrightarrow$ build on each others' efforts
- automatic parallelization of Monte Carlo calculations, and '$O(1)$ graphs'
  $\longrightarrow$ greatly simplify programming task
- borrowing codes / tools from other fields, such as numerical relativity
- begins to make high performance computation easily accesible to theoretical physicist

Conclusions

Cactus can make discrete quantum gravity computations much easier

Provides:

- language independence & modularity allows code sharing, code reuse, specialization
  $\longrightarrow$ build on each others' efforts
- automatic parallelization of Monte Carlo calculations, and '$O(1)$ graphs'
  $\longrightarrow$ greatly simplify programming task
- borrowing codes / tools from other fields, such as numerical relativity
- begins to make high performance computation easily accesible to theoretical physicist

Conclusions

Cactus can make discrete quantum gravity computations much easier

Provides:

- language independence & modularity allows code sharing, code reuse, specialization
  $\longrightarrow$ build on each others' efforts
- automatic parallelization of Monte Carlo calculations, and '$O(1)$ graphs'
  $\longrightarrow$ greatly simplify programming task
- borrowing codes / tools from other fields, such as numerical relativity
- begins to make high performance computation easily accesible to theoretical physicist

Conclusions

Cactus can make discrete quantum gravity computations much easier

Provides:

- language independence & modularity allows code sharing, code reuse, specialization
  $\longrightarrow$ build on each others' efforts
- automatic parallelization of Monte Carlo calculations, and '$O(1)$ graphs'
  $\longrightarrow$ greatly simplify programming task
- borrowing codes / tools from other fields, such as numerical relativity
- begins to make high performance computation easily accesible to theoretical physicist

Conclusions

Cactus can make discrete quantum gravity computations much easier

Provides:

- language independence & modularity allows code sharing, code reuse, specialization
  $\longrightarrow$ build on each others' efforts
- automatic parallelization of Monte Carlo calculations, and '$O(1)$ graphs'
  $\longrightarrow$ greatly simplify programming task
- borrowing codes / tools from other fields, such as numerical relativity
- begins to make high performance computation easily accesible to theoretical physicist

Conclusions

Cactus can make discrete quantum gravity computations much easier

Provides:

- language independence & modularity allows code sharing, code reuse, specialization
  $\longrightarrow$ build on each others' efforts
- automatic parallelization of Monte Carlo calculations, and '$O(1)$ graphs'
  $\longrightarrow$ greatly simplify programming task
- borrowing codes / tools from other fields, such as numerical relativity
- begins to make high performance computation easily accesible to theoretical physicist

Conclusions

Cactus can make discrete quantum gravity computations much easier

Provides:

- language independence & modularity allows code sharing, code reuse, specialization
  $\longrightarrow$ build on each others' efforts
- automatic parallelization of Monte Carlo calculations, and '$O(1)$ graphs'
  $\longrightarrow$ greatly simplify programming task
- borrowing codes / tools from other fields, such as numerical relativity
- begins to make high performance computation easily accesible to theoretical physicist

Conclusions

Cactus can make discrete quantum gravity computations much easier

Provides:

- language independence & modularity allows code sharing, code reuse, specialization
  $\longrightarrow$ build on each others' efforts
- automatic parallelization of Monte Carlo calculations, and '$O(1)$ graphs'
  $\longrightarrow$ greatly simplify programming task
- borrowing codes / tools from other fields, such as numerical relativity
- begins to make high performance computation easily accesible to theoretical physicist

Conclusions

Cactus can make discrete quantum gravity computations much easier

Provides:

- language independence & modularity allows code sharing, code reuse, specialization
  $\longrightarrow$ build on each others' efforts
- automatic parallelization of Monte Carlo calculations, and '$O(1)$ graphs'
  $\longrightarrow$ greatly simplify programming task
- borrowing codes / tools from other fields, such as numerical relativity
- begins to make high performance computation easily accesible to theoretical physicist

Conclusions

Cactus can make discrete quantum gravity computations much easier

Provides:

- language independence & modularity allows code sharing, code reuse, specialization
  $\longrightarrow$ build on each others' efforts
- automatic parallelization of Monte Carlo calculations, and '$O(1)$ graphs'
  $\longrightarrow$ greatly simplify programming task
- borrowing codes / tools from other fields, such as numerical relativity
- begins to make high performance computation easily accesible to theoretical physicist

Conclusions

Cactus can make discrete quantum gravity computations much easier

Provides:

- language independence & modularity allows code sharing, code reuse, specialization
  $\longrightarrow$ build on each others' efforts
- automatic parallelization of Monte Carlo calculations, and '$O(1)$ graphs'
  $\longrightarrow$ greatly simplify programming task
- borrowing codes / tools from other fields, such as numerical relativity
- begins to make high performance computation easily accesible to theoretical physicist

Conclusions

Cactus can make discrete quantum gravity computations much easier

Provides:

- language independence & modularity allows code sharing, code reuse, specialization
  $\longrightarrow$ build on each others' efforts
- automatic parallelization of Monte Carlo calculations, and '$O(1)$ graphs'
  $\longrightarrow$ greatly simplify programming task
- borrowing codes / tools from other fields, such as numerical relativity
- begins to make high performance computation easily accesible to theoretical physicist

Conclusions

Cactus can make discrete quantum gravity computations much easier

Provides:

- language independence & modularity allows code sharing, code reuse, specialization
  $\longrightarrow$ build on each others' efforts
- automatic parallelization of Monte Carlo calculations, and '$O(1)$ graphs'
  $\longrightarrow$ greatly simplify programming task
- borrowing codes / tools from other fields, such as numerical relativity
- begins to make high performance computation easily accesible to theoretical physicist

Conclusions

Cactus can make discrete quantum gravity computations much easier

Provides:

- language independence & modularity allows code sharing, code reuse, specialization
  $\longrightarrow$ build on each others' efforts
- automatic parallelization of Monte Carlo calculations, and '$O(1)$ graphs'
  $\longrightarrow$ greatly simplify programming task
- borrowing codes / tools from other fields, such as numerical relativity
- begins to make high performance computation easily accesible to theoretical physicist