Cactus-G:
Experiments with a Grid-Enabled Computational Framework

Dave Angulo, Ian Foster

Chuang Liu, Matei Ripeanu, Michael Russell

Distributed Systems Laboratory

University of Chicago & Argonne National Laboratory

Gabrielle Allen, Thomas Dramlitsch,

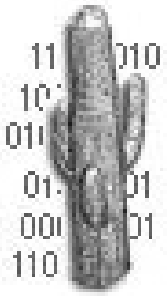
Ed Seidel, Thomas Radke

Max-Planck-Institut für Gravitationsphysik

UCSD, UIUC, U.Tenn GrADS Groups

Overview

- Research goals: Why Cactus-G
- Context: Numerical relativity, Cactus, dynamic Grid computing
- The Cactus-G Grid-enabled framework
- Cactus and GrADS
  - The Cactus Worm model problem
  - Dynamic resource selection & code migration
  - Experimental results
- Future directions & lessons learned

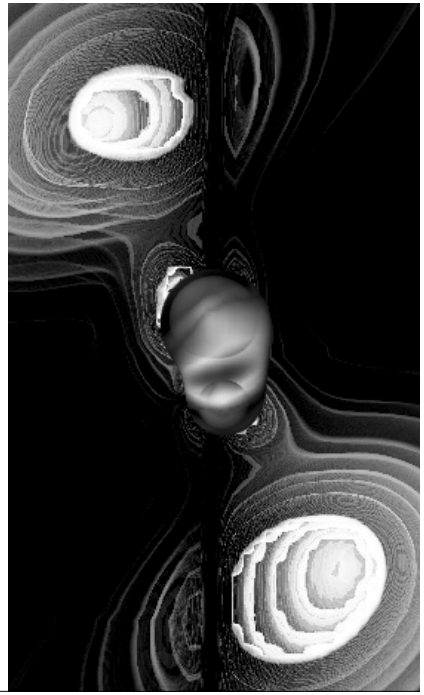


Research Goals

- Investigate methods and structures for efficient Grid execution via in-depth study of a demanding application, including
  - Constructs for adapting to heterogeneity
  - Constructs for dynamic resource acquisition
- Create testbed for GrADSoft components, as they emerge
- Investigate utility of computational frameworks as facilitator of Grid computing

Context (1): Numerical Relativity

- Numerical simulation of extreme astrophysical events: colliding black holes, neutron stars, etc.
  - Understand physics
  - Predict gravitational wave forms
- Relativistic effects => Einstein eqns
  - Computationally intensive (can be 1000s flops/grid point)
  - 3-D simulations only recently possible: demanding users



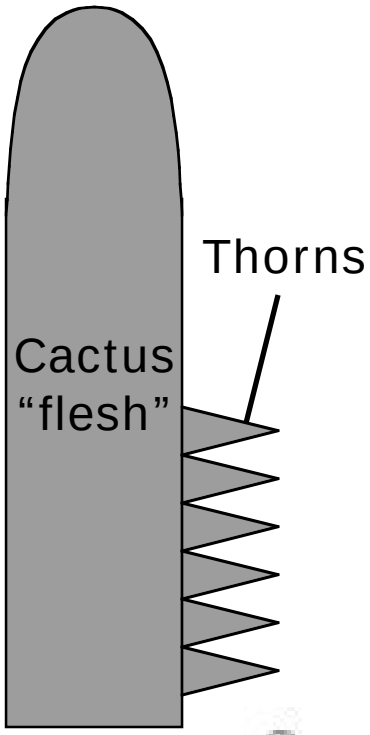
Colliding black holes

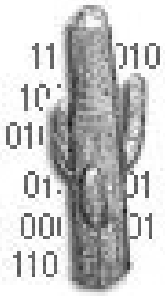
LIGO gravitational wave observatory

Context (2): Cactus

(Allen, Dramlitsch, Seidel, Shalf, Radke)

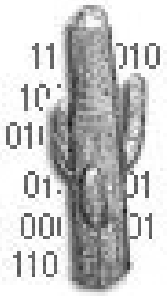
- Modular, portable framework for parallel, multidimensional simulations
- Construct codes by linking
  - Small core (flesh): mgmt services
  - Selected modules (thorns): Numerical methods, grids & domain decomps, visualization and steering, etc.
- Custom linking/configuration tools
- Developed for astrophysics, but not astrophysics-specific





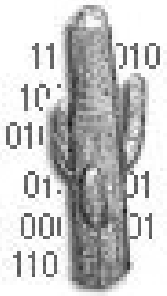
Context (3): Dynamic Grid Computing

- Application behaviors in a Grid environment:
  - Identify fastest/cheapest/biggest resources
  - Configure for efficient execution
  - Detect need for new resources or behaviors (e.g., due to resource slowdown, new subtasks, new appln regime, user steering, new resource available)
  - Adapt, and/or discover new resources; invoke subtasks on new resources and/or migrate
- We have users who want these behaviors; we also have the enabling machinery

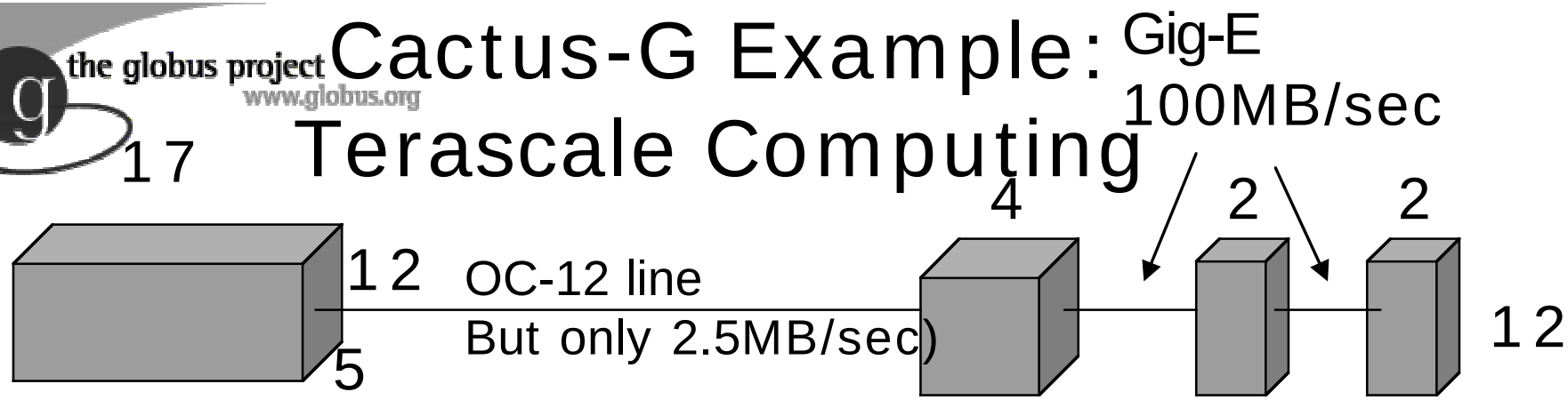


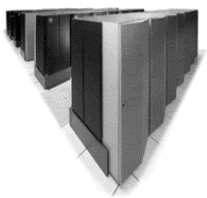
Cactus-G: An Application Framework for Dynamic Grid Computing

- Cactus thorns for active management of application behavior and resource use
- Heterogeneous resources, e.g.:
  - Irregular decompositions
  - Variable halo for managing message size
  - Msg compression (comp/comm tradeoff)
  - Comms scheduling for comp/comm overlap
- Dynamic resource behaviors/demands, e.g.:
  - Perf monitoring, contract violation detection
  - Dynamic resource discovery & migration
  - User notification and steering



Cactus-G Example: Terascale Computing





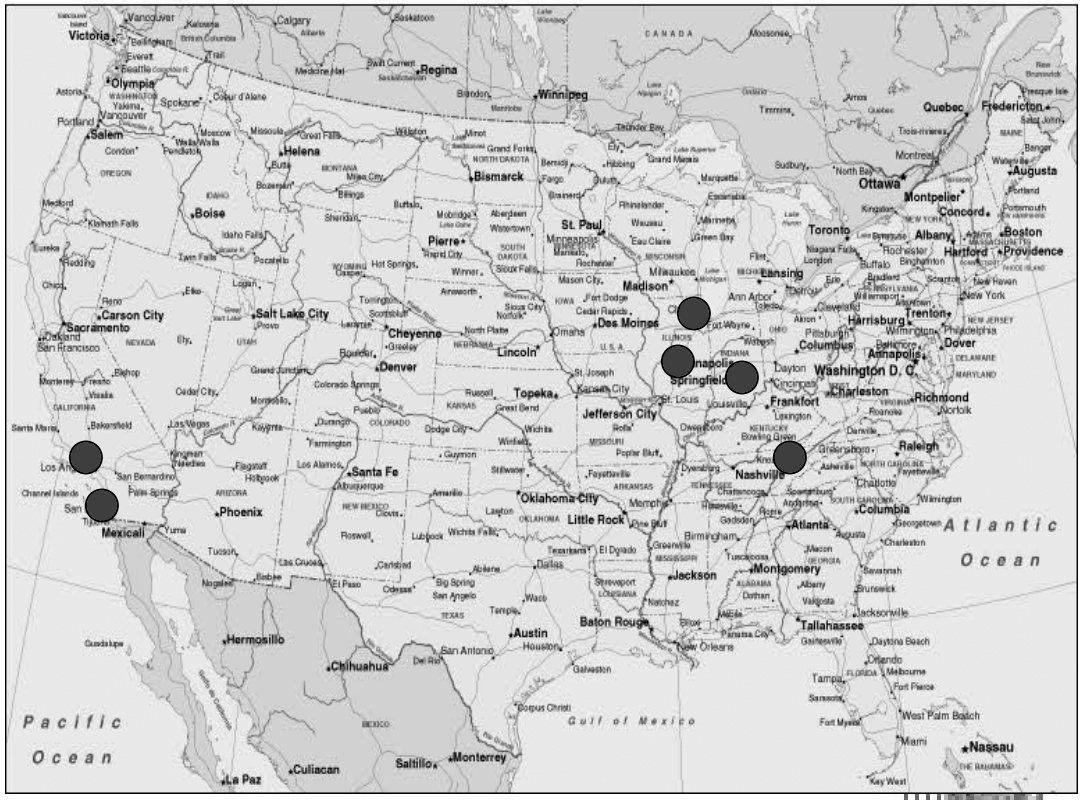
SDSC IBM SP
1024 procs
5x12x17 =1020

NCSA Origin Array
256+128+128
5x12x(4+2+2) =480

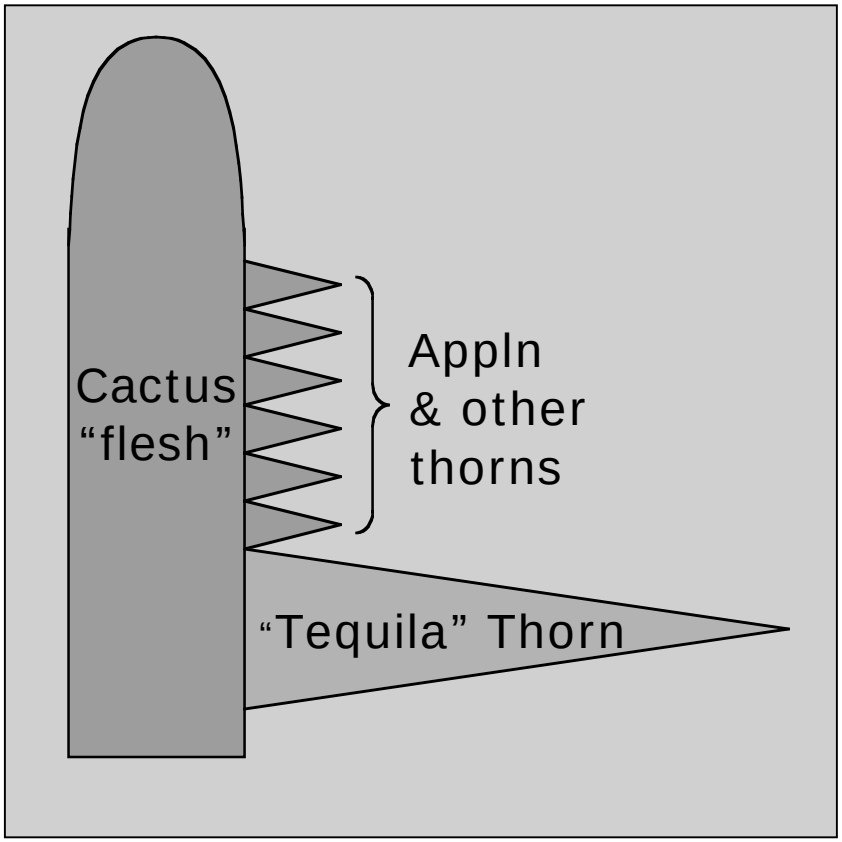
- Solved EEs for gravitational waves (real code)
  - Tightly coupled, communications required through derivatives
  - Must communicate 30MB/step between machines
  - Time step take 1.6 sec
- Used 10 ghost zones along direction of machines: communicate every 10 steps
- Compression/decomp. on all data passed in this direction
- Achieved 70-80% scaling, ~200GF (only 14% scaling without tricks)

Cactus-G Model Problem: The Cactus Worm

- Migrate to “faster/ cheaper” system
  - When better system discovered
  - When requirements change
  - When characteristics change (e.g., competition)
- Tests most elements of Cactus-G & GrADS



Cactus Worm Architecture



GrADS Mechanisms

Resource selector

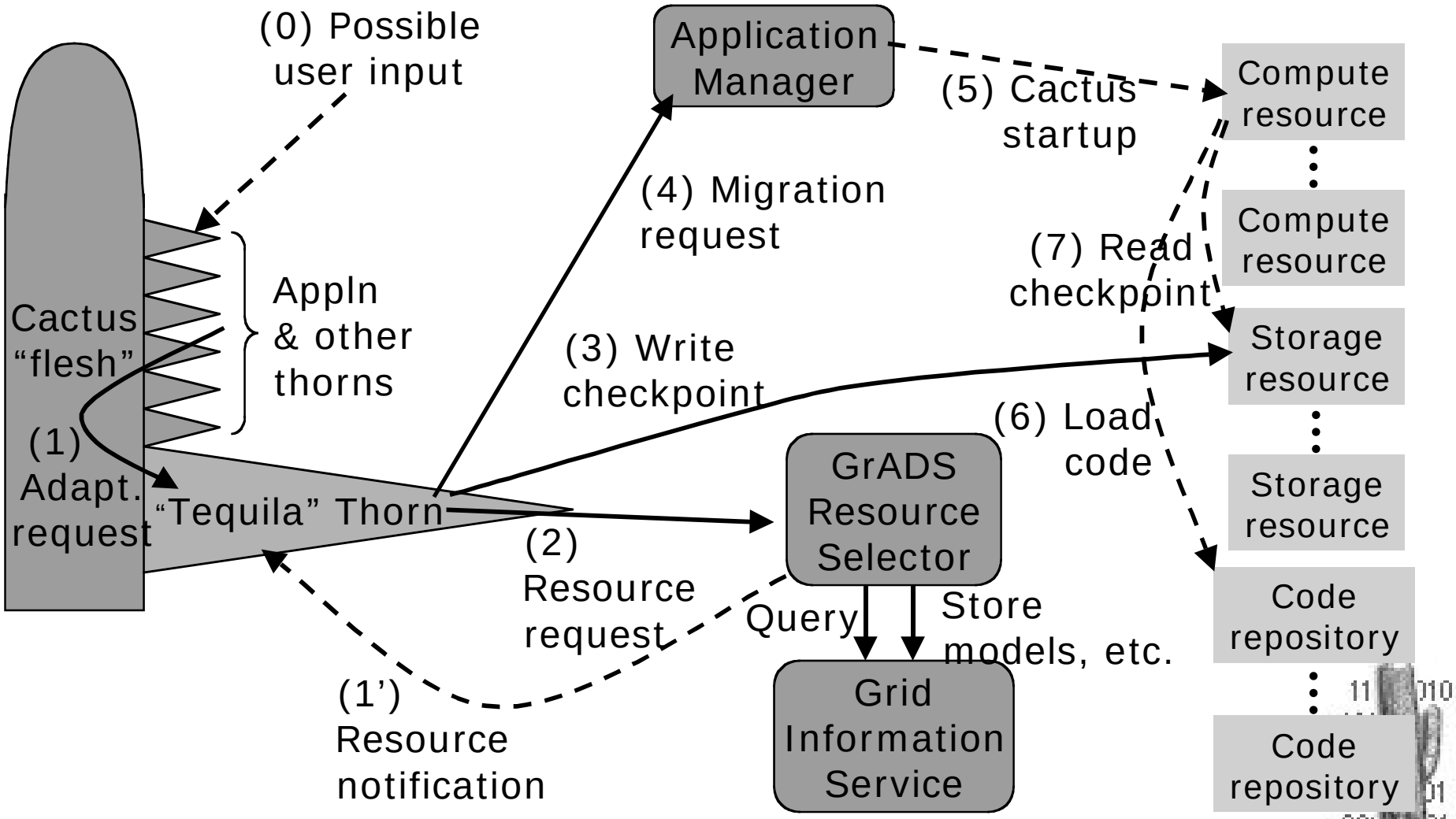
Application manager

Globus Toolkit substrate: resource discovery, allocation, management

Tequila Thorn Functions

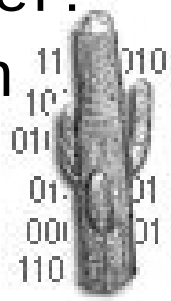
- Initiate adaptation on any one of
  - User request (e.g., HTTP thorn)
  - Notification of new resources
  - Application monitoring: contract violation
- Request resources (ClassAd protocol)
  - E.g., GrADS ResourceSelector
- Checkpoint application
- Contact App Manager to request restart
  - Security, robustness advantages vs. direct restart

Cactus Worm
Detailed Architecture & Operation



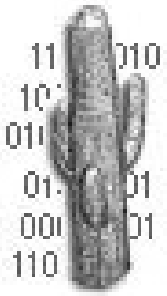
Contract Monitor

- Driven by three user-controllable parameters
  - Time quantum for “time per iteration”
  - % degradation in time per iteration (relative to prior average) before noting violation
  - Number of violations before migration
- Potential causes of violation
  - Competing load on CPU
  - Computation requires more processing power: e.g., mesh refinement, new subcomputation
  - Hardware problems

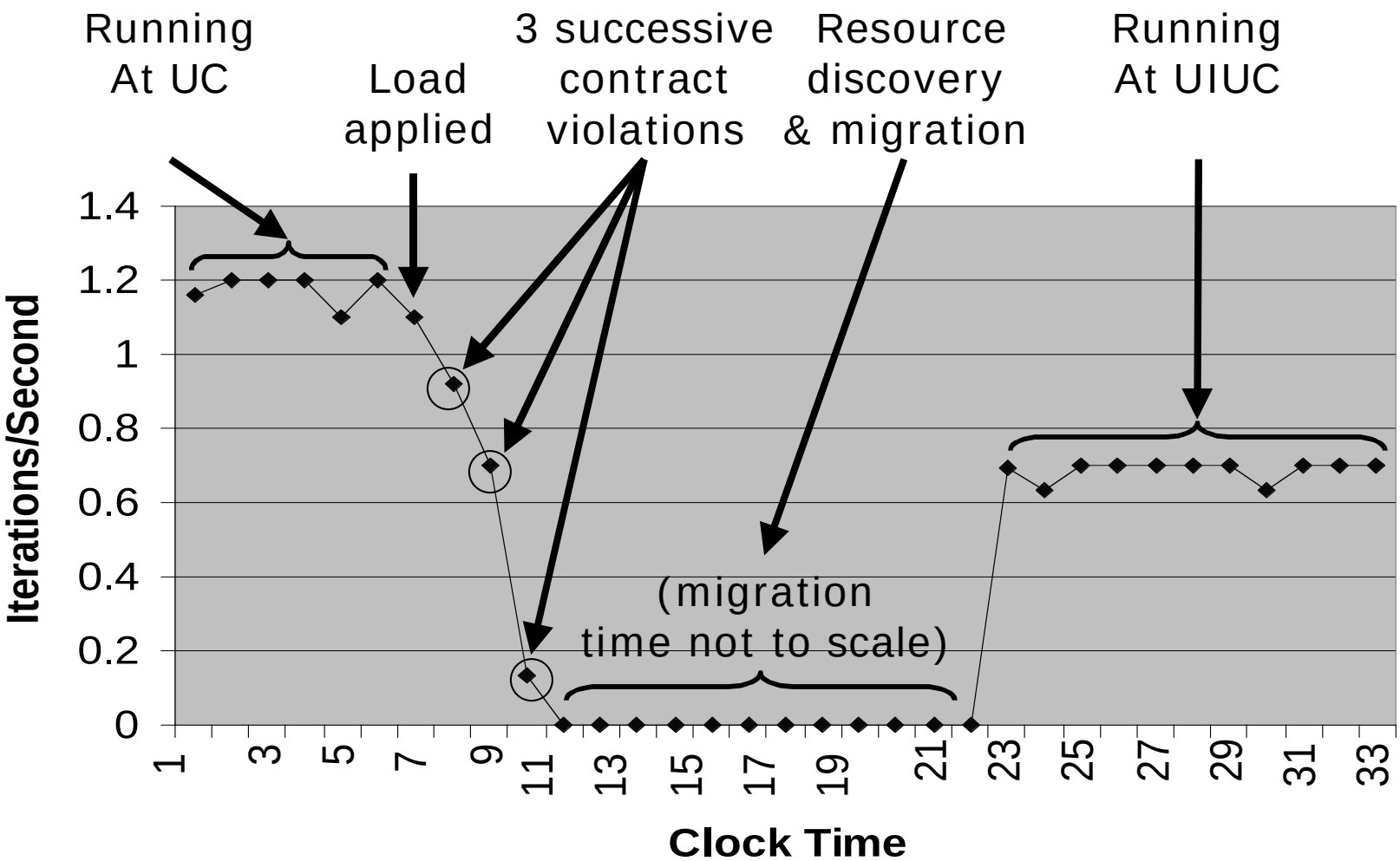


Current Status

- We have developed
  - Tequila thorn: monitoring, selection, control
  - ResourceSelector (via ClassAd protocol)
  - Cactus performance model
- We have demonstrated on GrADS Macro Grid
  - Contract monitoring for multiprocessor runs
  - Dynamic resource selection
  - Migration



Migration in Action



Ongoing and Future Work: New and Improved Capabilities

- Optimize migration process
- Use performance models during selection
  - And include cost of migration & information about future computation in the model
- Matchmaker-based ResourceSelector
  - Separation of concerns between resource characterization and selection
  - Study resource characterization process, use NWS-based prediction techniques
- Dynamic notification of availability of "better" resources

Ongoing and Future Work: Further Integration with GrADSoft

- Contract monitoring
  - Pablo
  - Issues: determining which thorns are monitored, or if flesh is monitored
- Program Preparation System
- Configurable Object Program and Application Launcher
  - Cactus has its own launcher and compiles its own code

Lessons Learned and Outcomes

- Lessons learned
  - A real & demanding application can exploit adaptive techniques to execute efficiently in Grid environments
  - Even a relatively regular application can incorporate a range of useful mechanisms for adaptive behaviors & resource demands
- Outcomes
  - Prototype Cactus-G framework: wonderful experimental platform