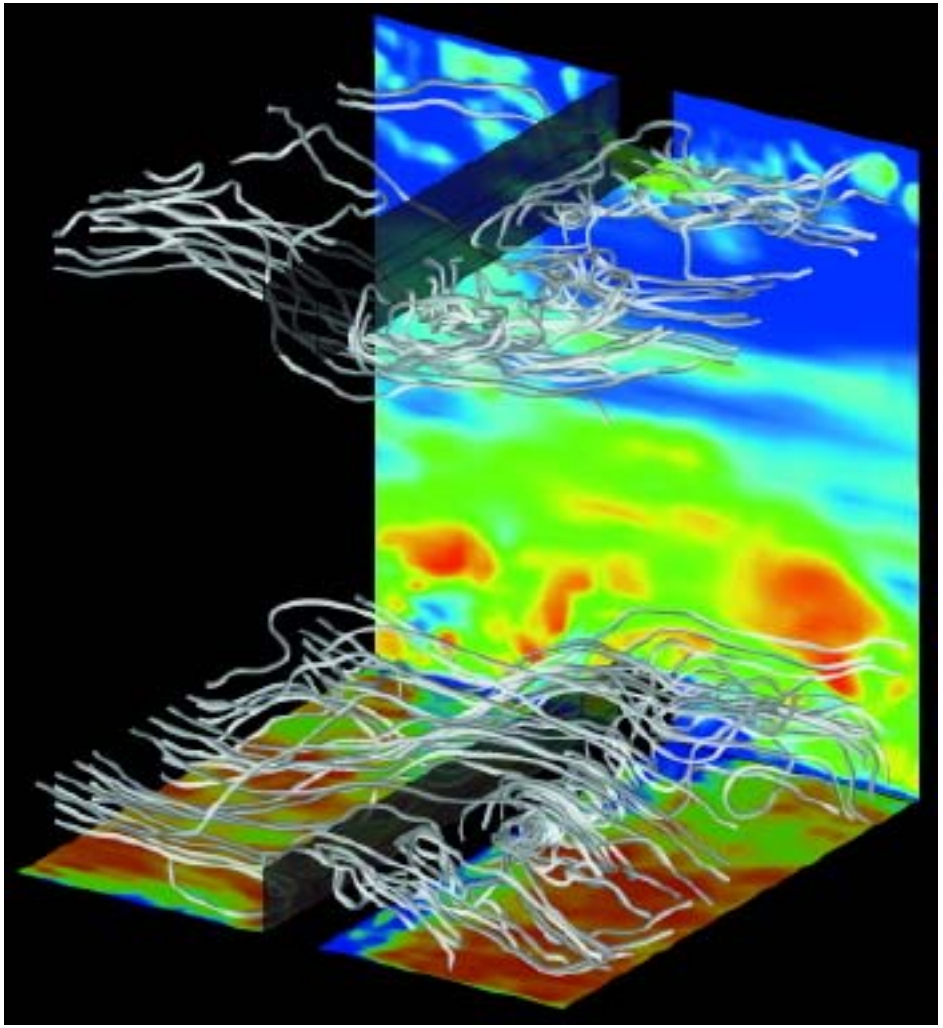
The Adaptive Code Kitchen:
Flexible Tools for Dynamic Application Composition

Pilsung Kang, Mike Heffner, Joy Mukherjee,
Naren Ramakrishnan, Srinidhi Varadrajan,
Calvin J. Ribbens, and Danesh K. Tafti

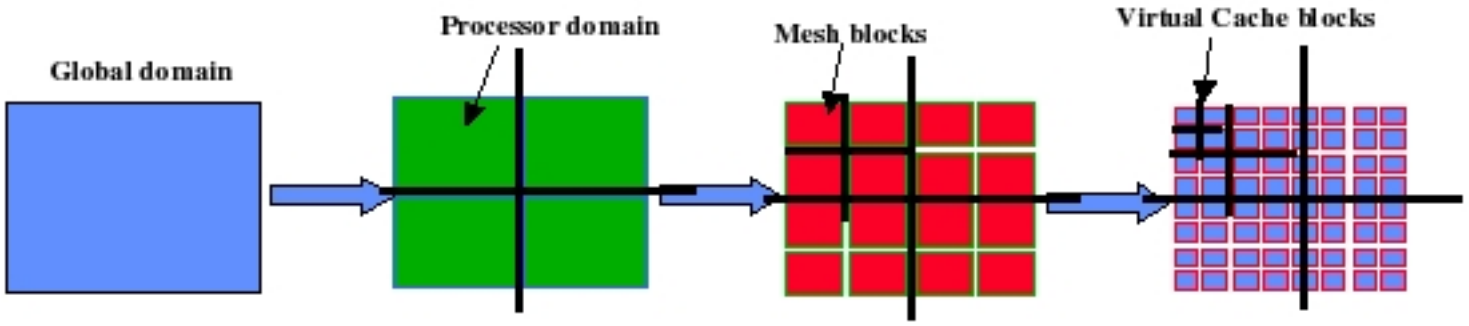
NSF CNS-0615181

Problem context



How can we abstract out adaptivity in complex scientific codes?

Adaptivity: its various incarnations



- Not just tinkering with
  - partitioning parameters
  - data decompositions
  - scheduling policies
- But working at a logical unit of
  - algorithms/object codes/models

Adaptivity: more..

- In an evolving turbulent flow
  - re-route calls from linear solver S1 to S2
- Integrate optimistic and exploratory algorithms
  - “use knowledge from the future to guide decisions in the present”
- In general
  - dynamically re-wire an application
  - support arbitrary code expansion/ contraction

Some related projects

- Performance modeling of solvers
  - SALSA, SANS [Dongarra et al.]
- Application level checkpointing
  - C^3 [Pingali et al.]
- Generalized frameworks
  - PCL [Adve et al.]
  - Tunability interfaces [Karamcheti et al.]
- Adaptive programming
  - AOP [Lieberherr et al.]

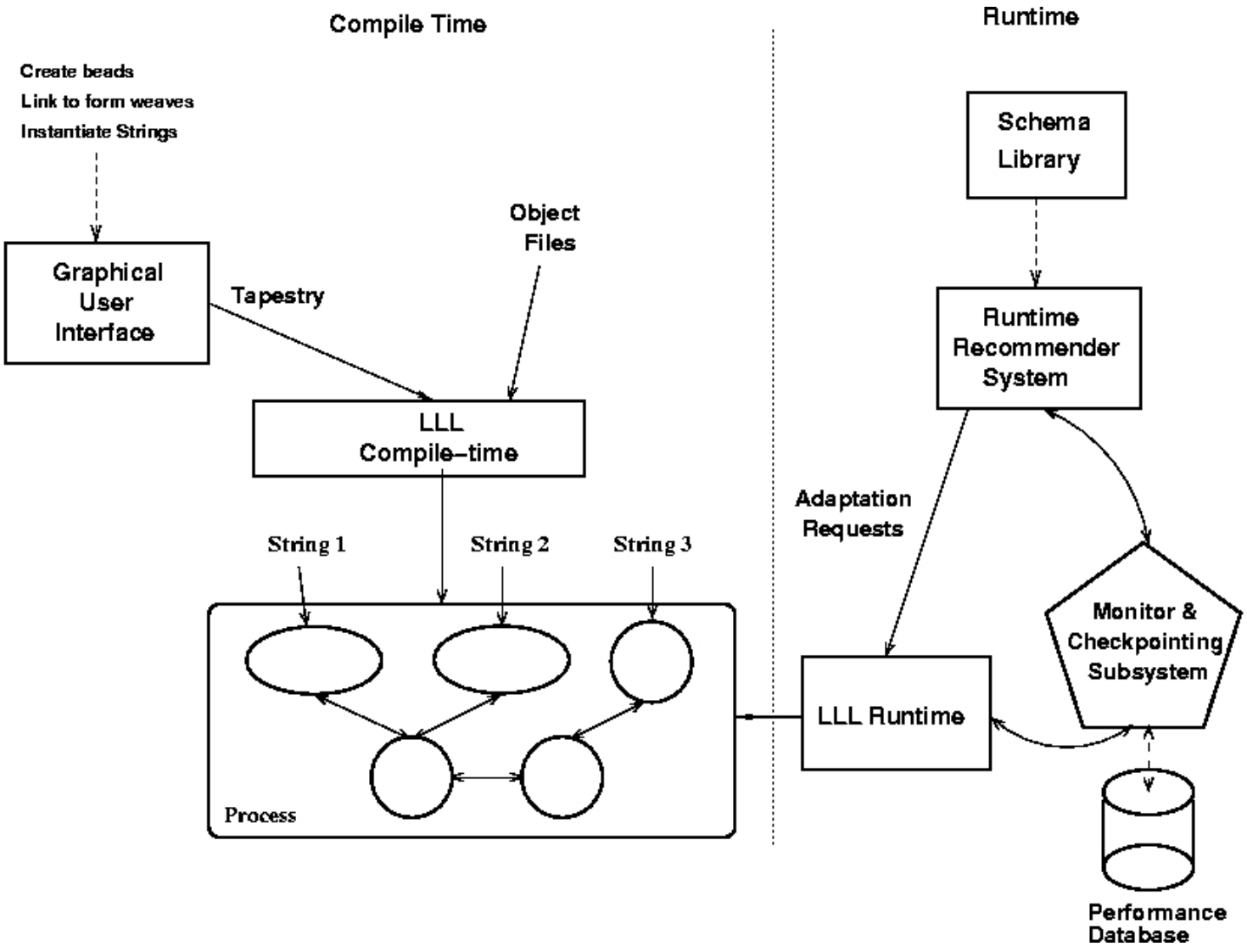
Goals of the “Kitchen”

- Componentization without OO
  - Support for legacy scientific codes
- Runtime system for instrumenting
  - Function interception
  - Continuation modification
  - Dynamic process checkpointing/rollback
- Adaptivity schemas
  - Recipes of how composition/adaptation will occur

The Adaptive Code Kitchen builds upon

- NSF CAREER EIA-0133840
  - Network emulation
- NSF CAREER EIA-9984317
  - Runtime recommender systems

How the Kitchen works



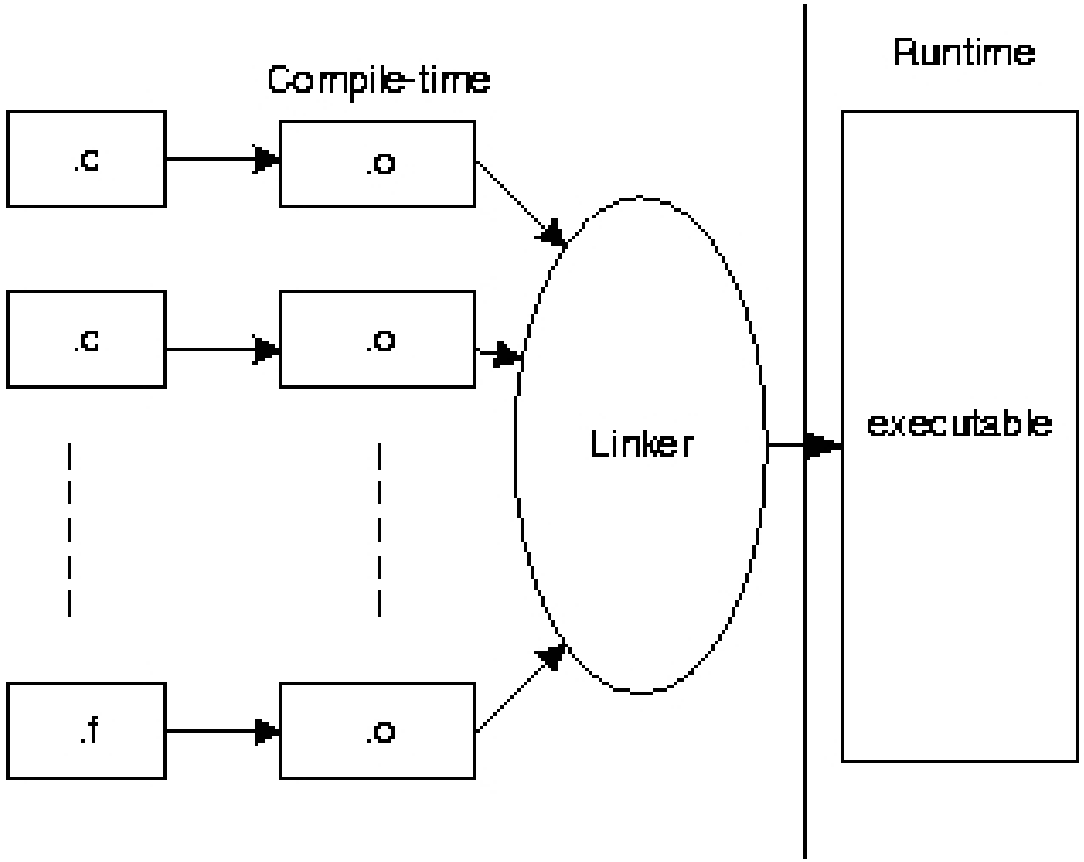
Building the Kitchen

- Load and Let Link (LLL)
- Primitives for Runtime Composition
- Adaptivity Schemas

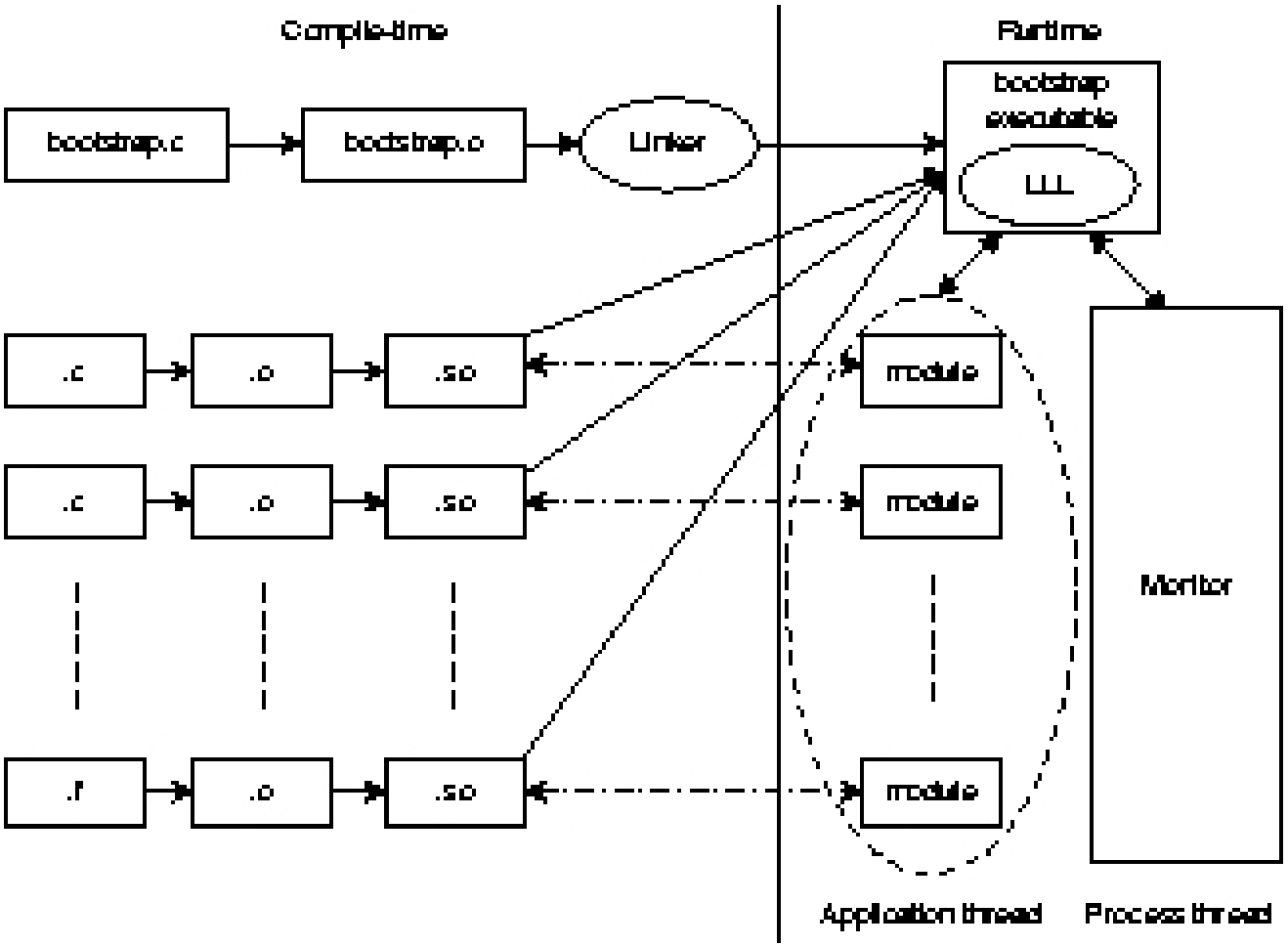
LLL

- Agile and flexible loading/composition of native code components
- Module: unit of encapsulation
  - runtime image of an object file compiled from source written in *any* language
- Runtime control over addition/modification of module context table (MCT)

Traditional compile-time linking



What LLL does



Runtime Composition Primitives

- Function interception
  - Wrap (at caller end) with pre-/post- handlers
- Continuation modification
  - Recommender system triggers pre-/post-callbacks for desired functions
- Dynamic process checkpointing
  - Rollback using ‘Dejavu’ snapshot library

Adaptivity Schemas

- High level “recipes” of rewiring
  - Staged composition
  - Adaptation of problem decompositions
  - Algorithm switching
  - Graphs of models

Whats cookin currently

- Multiple levels of adaptivity
  - Algorithms: steady flow, time-dependent flow, compressible high-speed flow...
  - Models: RANS, LES, DNS, ...
  - Solvers: scaling + preconditioner + algo.
- Grand goal
  - Simulate leading edge film cooling flows for gas turbine blades

Questions?

Contact:

Naren Ramakrishan

naren@cs.vt.edu

http://people.cs.vt.edu/~naren