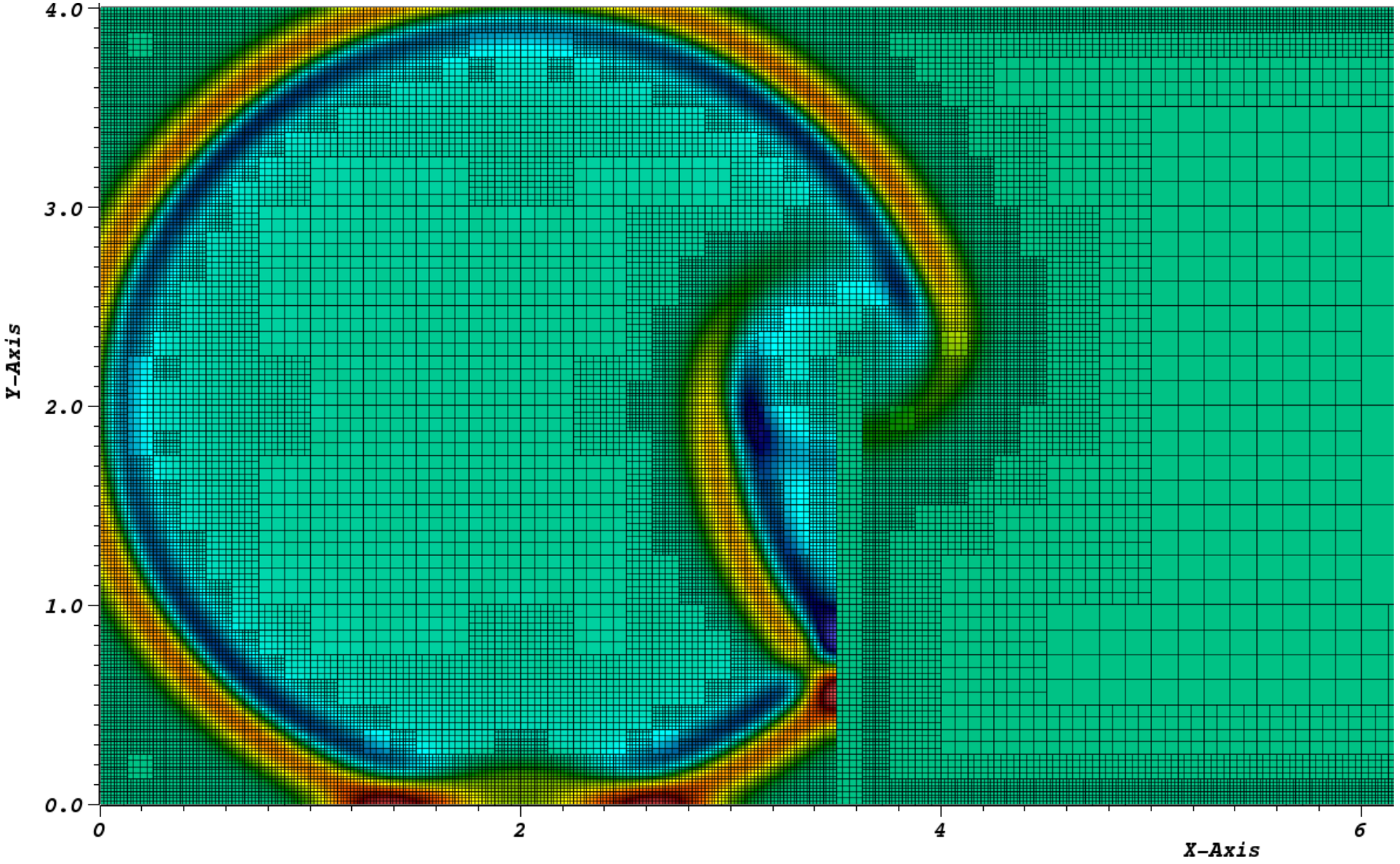
C2S@EXA Meeting
July 10, 2014

Towards a Reconfigurable HPC Component Model

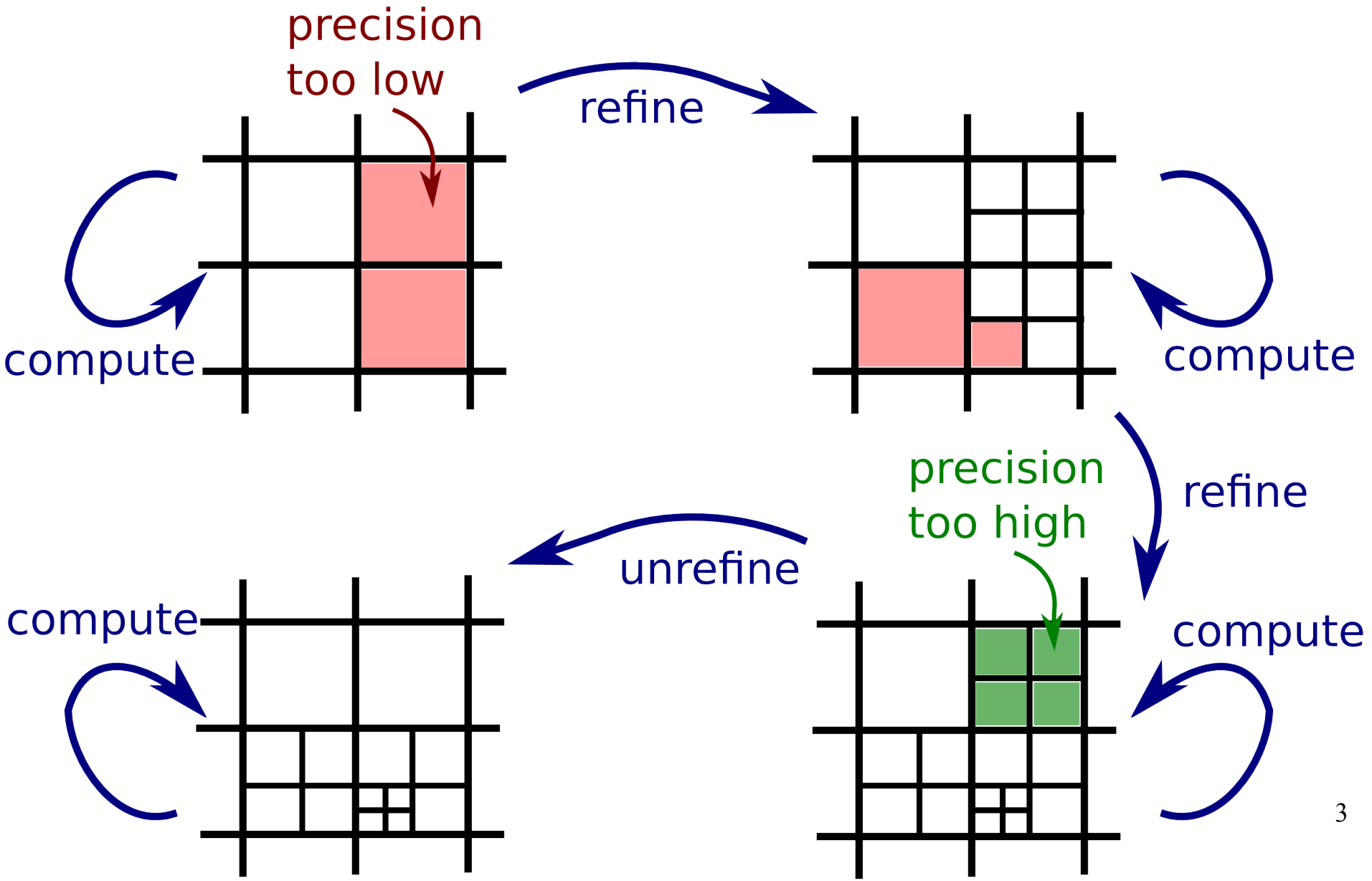
Vincent Lanore[1], Christian Pérez[2]

[1] ENS de Lyon, LIP
[2] Inria, LIP
Avalon team

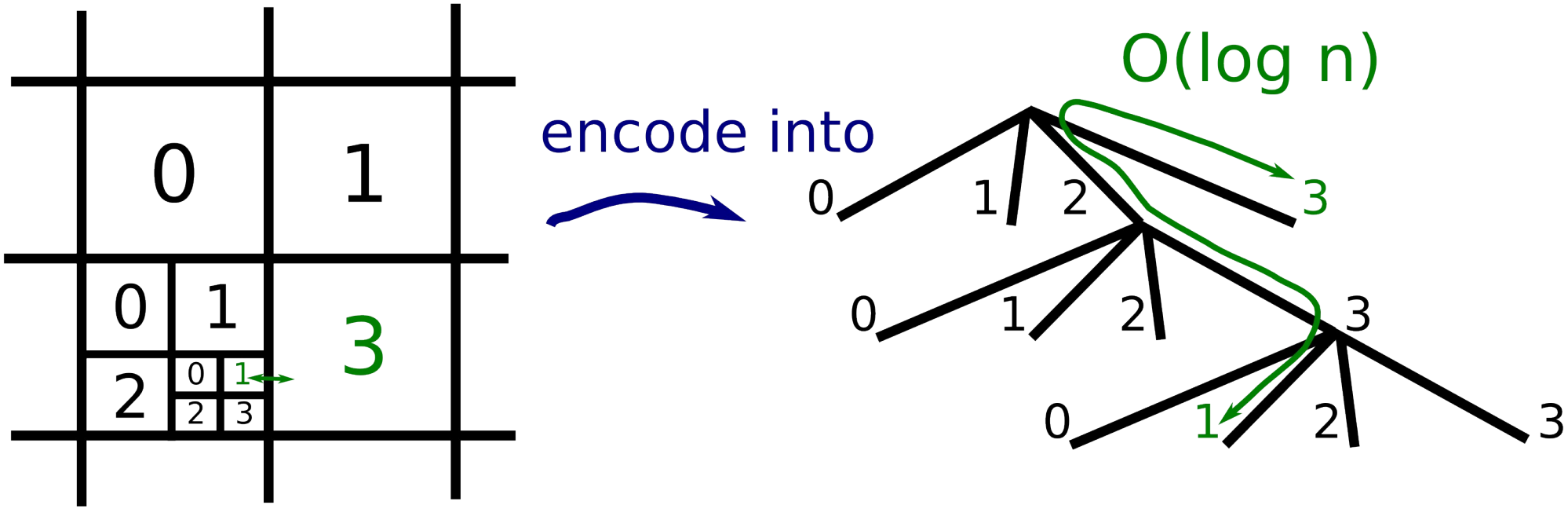
Adaptive Mesh Refinement



2D Recursive AMR

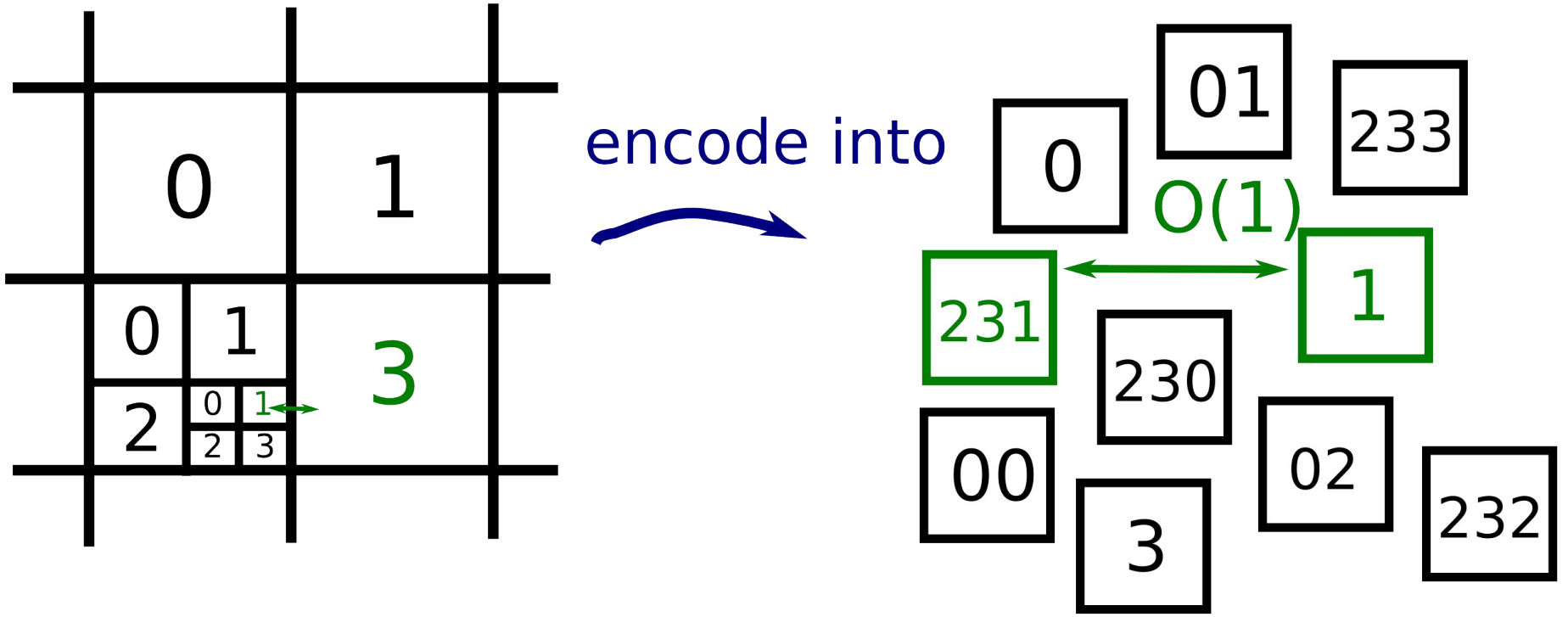


Classical AMR Algorithms



Distributed quadtree/octree structures
O(log n) costs

A More Scalable AMR Algorithm



- coordinates + runtime
- efficient O(1) communications and lookup
- easy distribution and load balancing

Langer et al, *Scalable Algorithms for Distributed-Memory Adaptive Mesh Refinement*, Computer Architecture and High Performance Computing (SBAC-PAD), 2012 IEEE 24th International Symposium

Challenges

A programming challenge

- lots of distributed computing units
- asynchronous refining and unrefining
- neighbors with unknown state

Performance constraint

- eg, benchmark at 5 ms/iteration with 2k ranks on Cray XK6 'Titan'[1]

[1] Langer et al, *Scalable Algorithms for Distributed-Memory Adaptive Mesh Refinement*, Computer Architecture and High Performance Computing (SBAC-PAD), 2012 IEEE 24th International Symposium

Programming Paradigm

Component Models

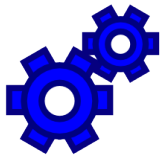
Reuse
- no reinventing the wheel
- mixing components from different sources

Separation of concerns
- low-level programming on one-side
- high-level application structure on the other side

High-level abstractions
- hierarchy
- connectors
- genericity

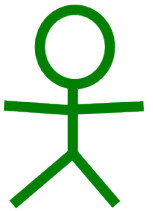
...

Component Models: Principle

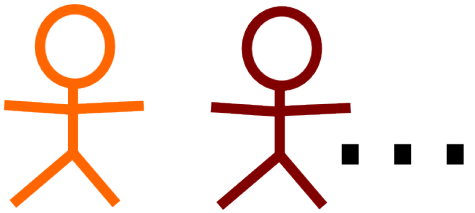


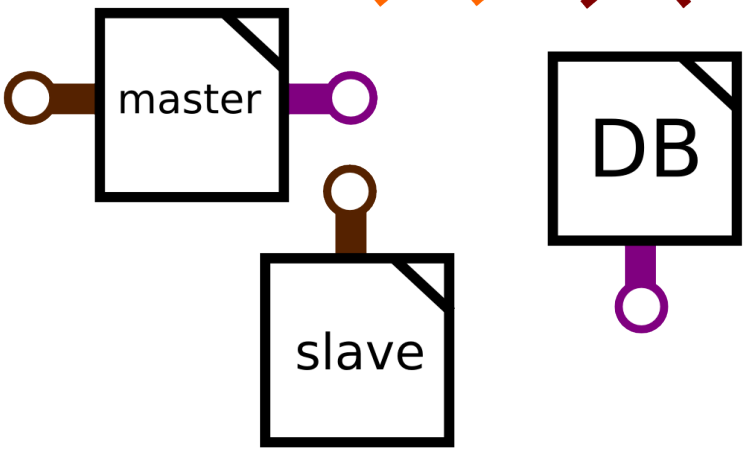
Component model

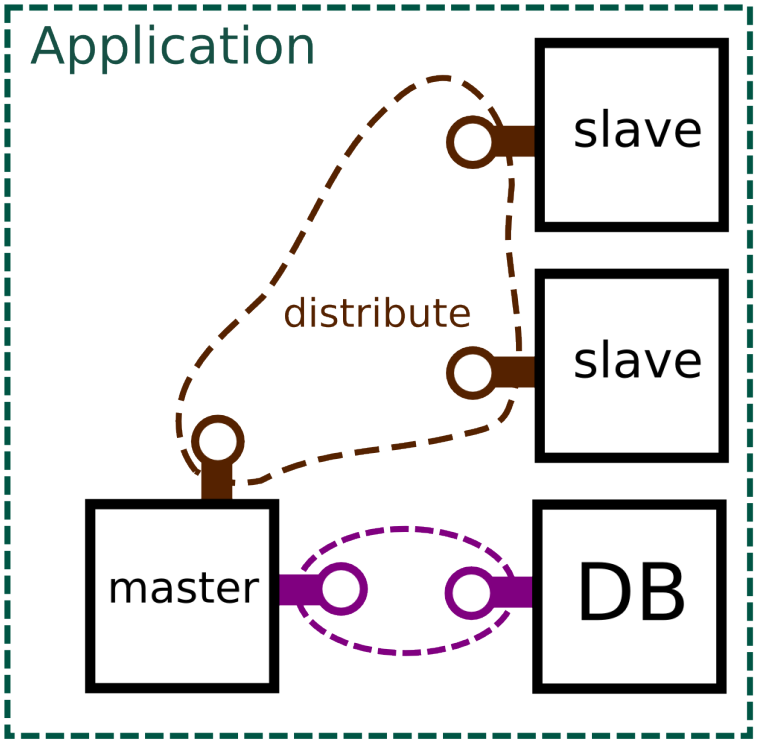
Component assembler
high-level view
reuse of 3rd-party components



Component writers
low-level concerns
well-defined interfaces







component instance

component type

port

connector

user

Example: L²C

Low-level component model

- on top of C++/Fortran
- components = objects + simple interfaces
- connectors
  - C++/Fortran ref
  - MPI
  - Corba

Developed by

J. Bigot, C. Pérez

Characteristics

No overhead at runtime

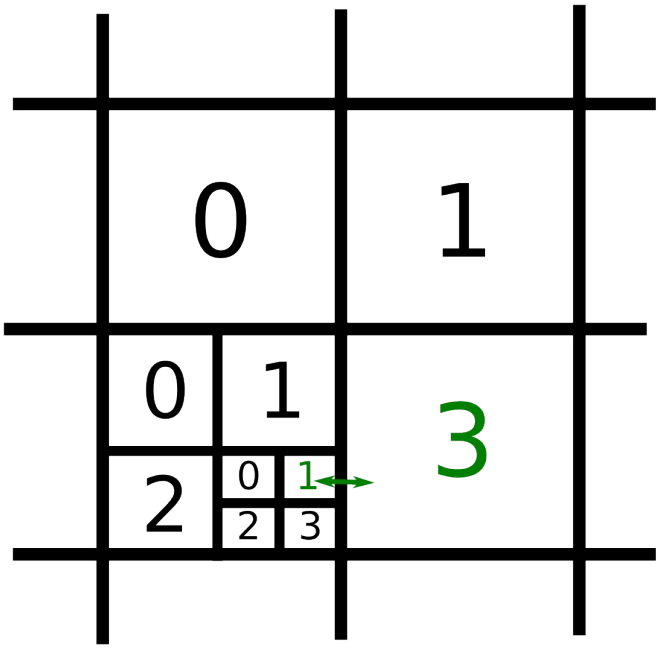
Static assembly

Also

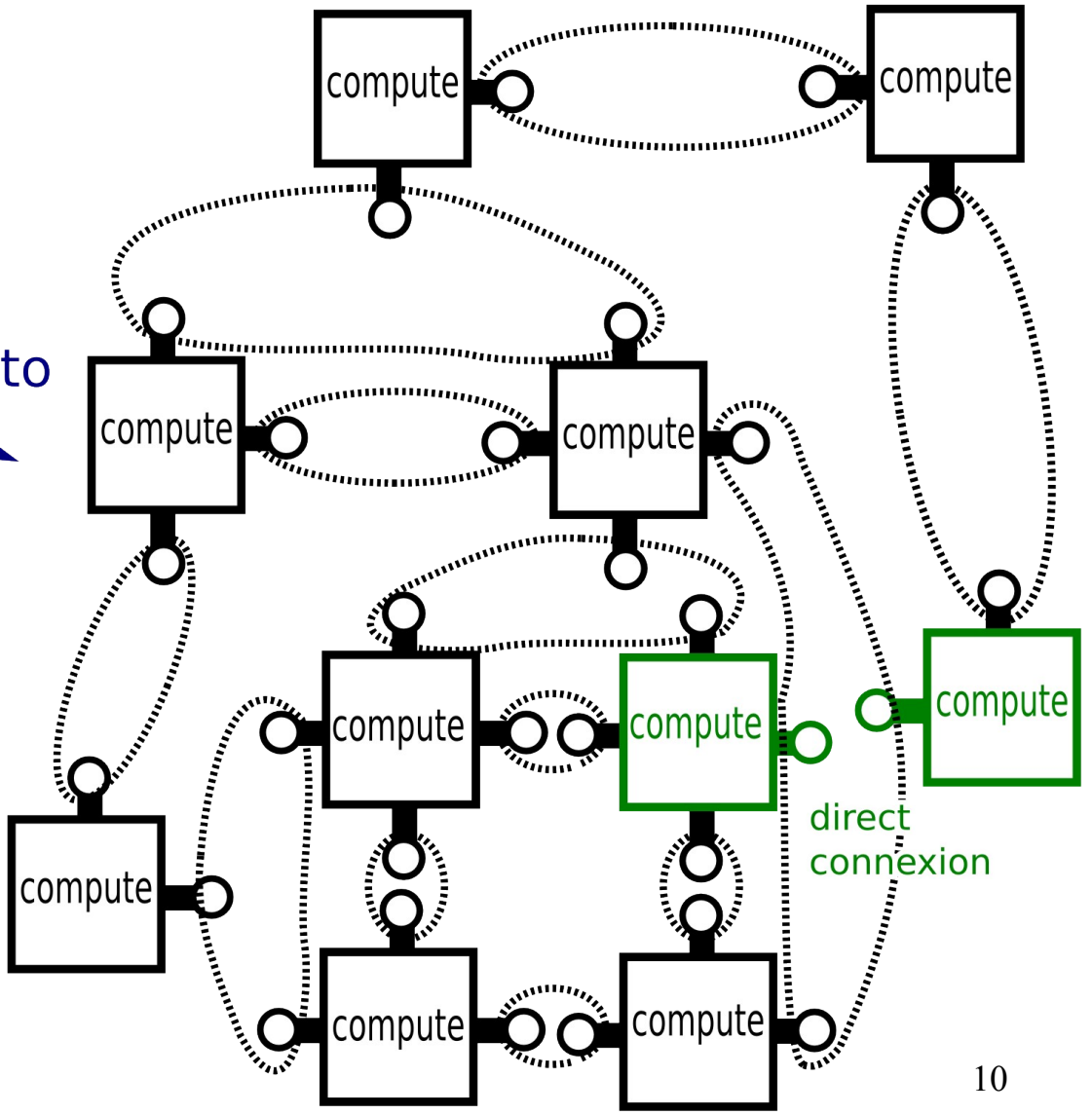
Charm++ version (gluon++)

Example

Component Models and AMR

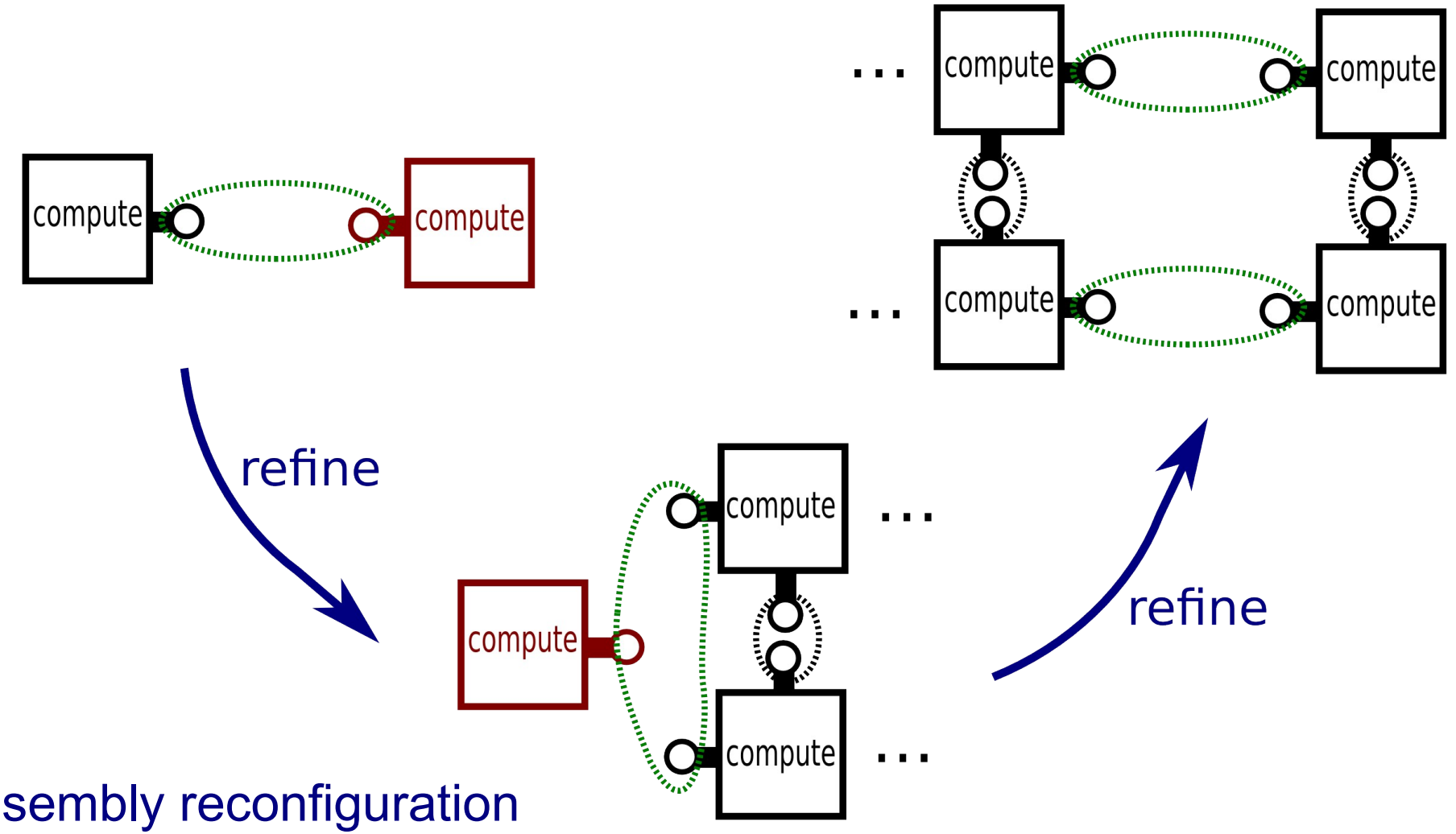


encode into



Example

Component Models and AMR



assembly reconfiguration
quiescent state

Ongoing Work

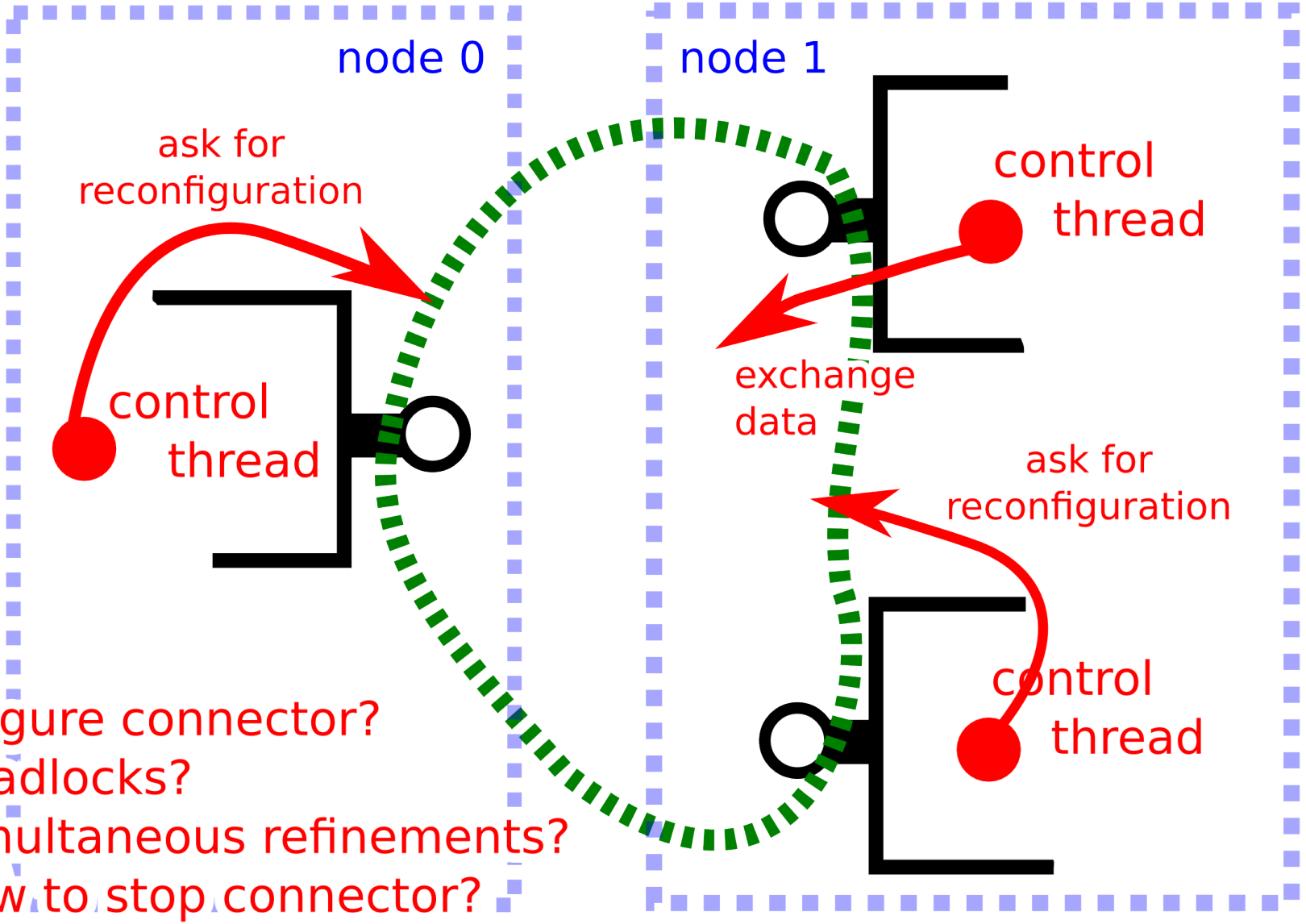
Component AMR Implementation

Implemented: L²C + pthread AMR benchmark

- as little synchronization as possible
- no actual computing
- first multicore performance tests
  - on Grid'5000 stremi node with 2x12 1.7GHz cores
  - 2-3 ms per iteration per thread up to 16k threads
  - synchronization-bound
- ~1k C++ lines
  - lots of bug-prone low-level synchronization
  - verbose component reconfiguration (eg, instantiation)
  - complex 1-to-n connexion logic

Problem

Component Models and AMR



reconfigure connector?
=> deadlocks?
=> simultaneous refinements?
=> how to stop connector?

Challenge

HPC Reconfigurable CMs

Existing Component Models

- either low-performance implementation
- or no support for reconfiguration (eg, L²C)

Goal: Component Model

- distributed
- reconfigurable
- efficient

First step

Minimalistic Low-overhead Model

Our approach

- Take a simple & efficient component model (à la L²C)
- Add a few concepts to ease reconfiguration

Lockables

Some elements can be locked

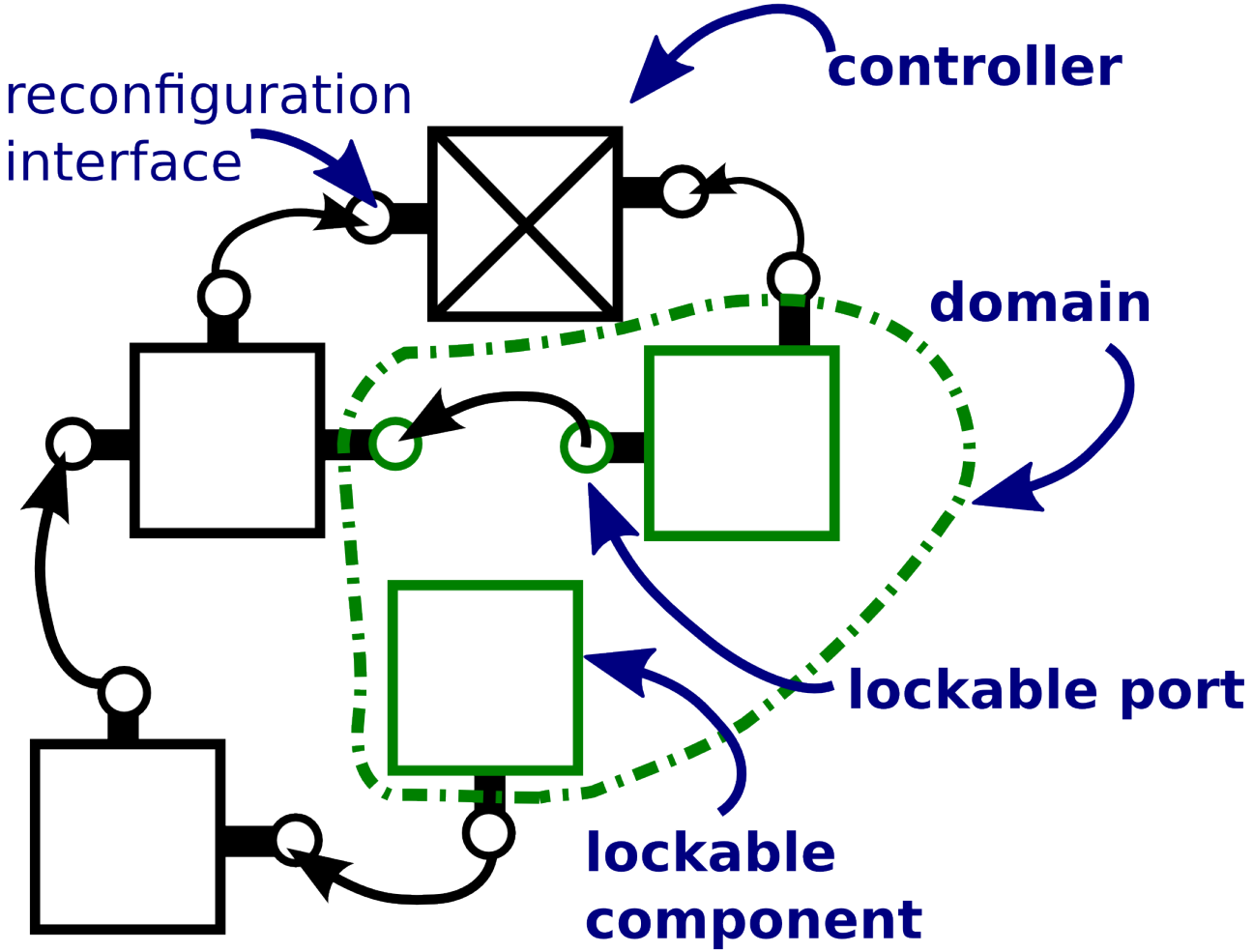
Domains

Whole subsets of the assembly can be locked under certain conditions

Controllers

Components responsible for domain locking and more

Lockables, Controllers, Domains

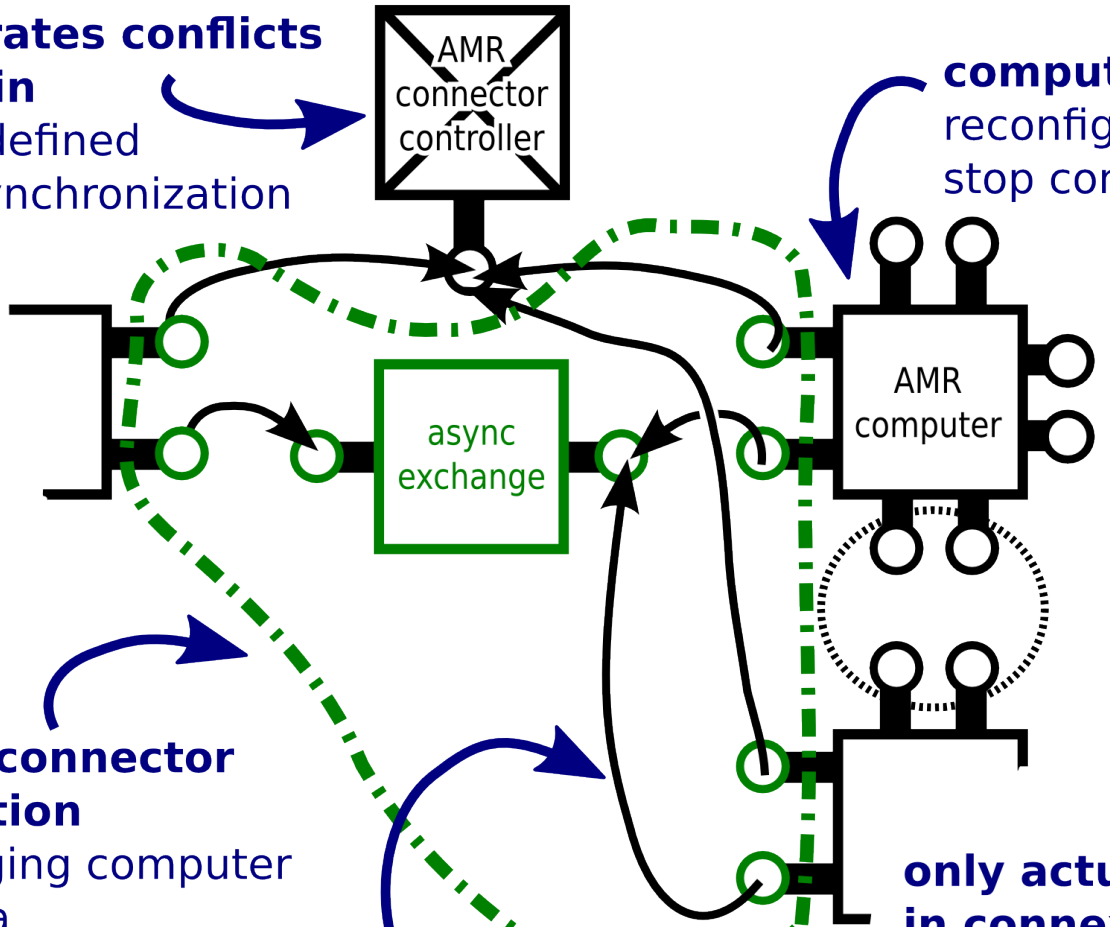


Controller API

- *create*
- *destroy*
- *connect*
- ***lock**/unlock domain*
- ***view** domain contents*
- *add/remove element*

+ user-defined reconfiguration methods

AMR Assembly



controller arbitrates conflicts and locks domain
no need for user-defined reconfiguration synchronization

computer not in domain
reconfiguration does not stop computation

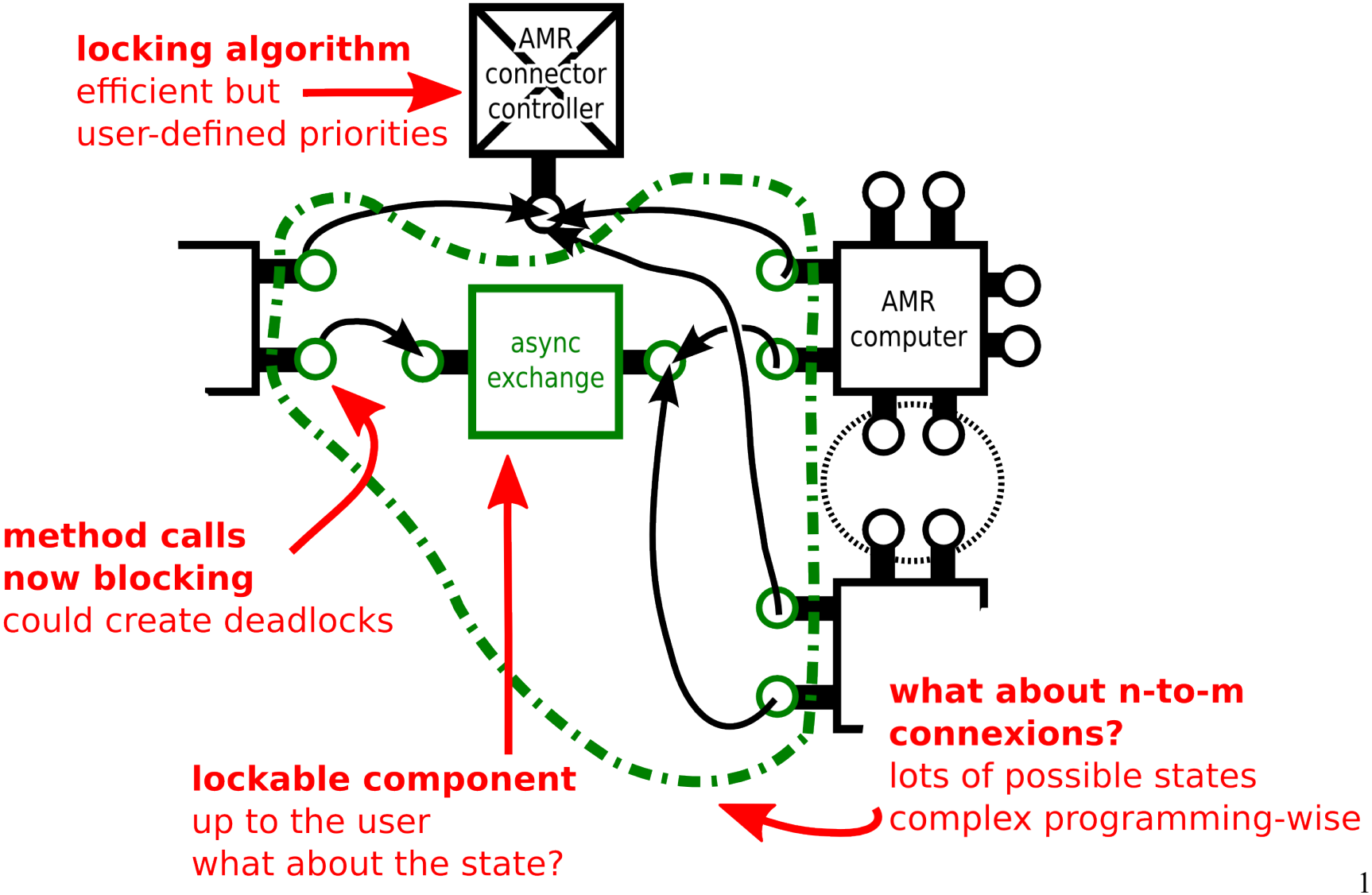
can replace connector implementation
without changing computer
eg, MPI, Corba

low-level connexion
efficient, very low overhead
eg, C++ pointer + mutex

only actual neighbors in connexion
no unnecessary synchronization with non-neighbors

Example + Problems

AMR Assembly



Ongoing Work

Formal Model

Assembly syntax

A=(C,P,o,r,E,K,d,L)

- *C,* component set
- *P,* port set
- *o,* owners
- *r,* references
- *E*, entry points
- *K*, controllers
- *d*, domains
- *L*, lockables

Semantic

- call stack
- parallel non-deterministic calls
- constraints on locked elements
- hypothesis for lockability
- well-formed assemblies

Goals and perspectives

- prove lock algorithms
- simple control hypothesis
- lockable by construction

The end

Conclusion and Perspectives

Presented today

- AMR use case
- L²C+pthreads implementation
  - up to 16k
- towards a HPC reconfigurable component model
  - lockables
  - domains
  - controllers

Perspectives

- implementation
  - distributed
  - integration
- experiments
- lockable domains formal model
- higher-level component features