Memory Tagging in Charm++

Filippo Gioachin
Laxmikant V. Kalé

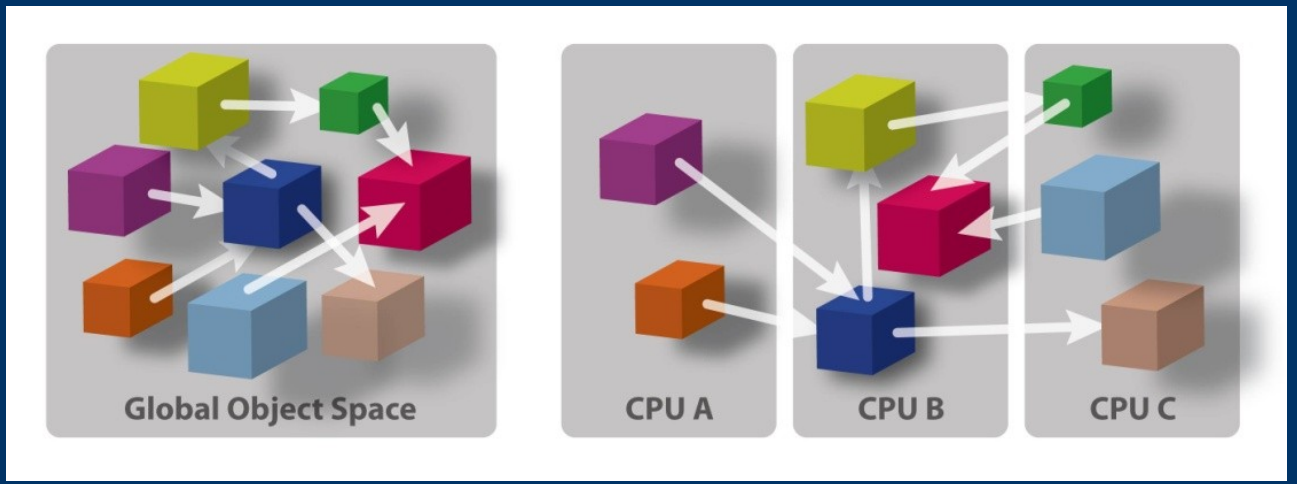
Department of Computer Science
University of Illinois at Urbana-Champaign

Outline

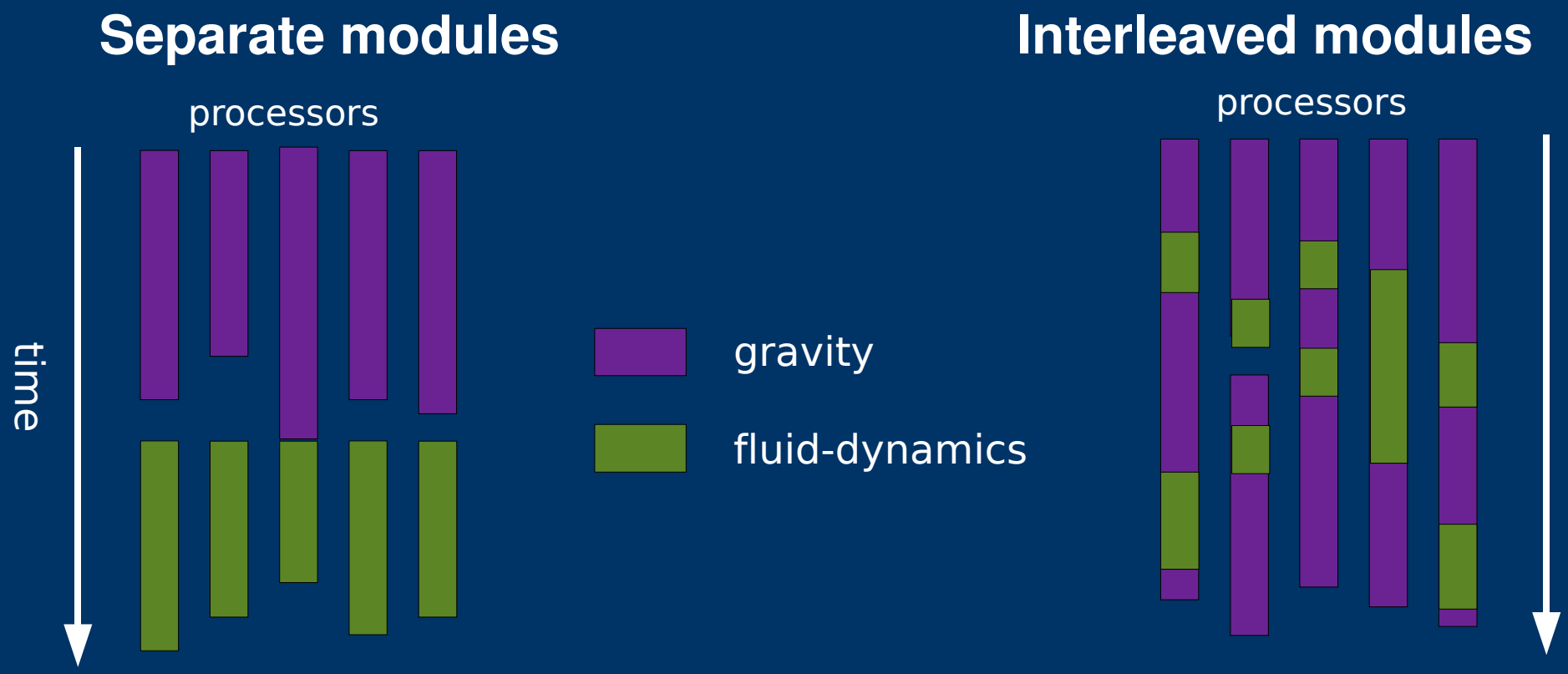
- Overview
  - Charm++ RTS
  - CharmDebug
- Memory tagging
  - Charm++ memory subsystem
  - Detecting memory violations
- Future work

Charm++ Overview

- Middleware written in C++
- User decomposes work among objects (*chares*)
- System maps chares to processors
  - automatic load balancing
  - communication optimizations
- Communicate through asynchronous messages



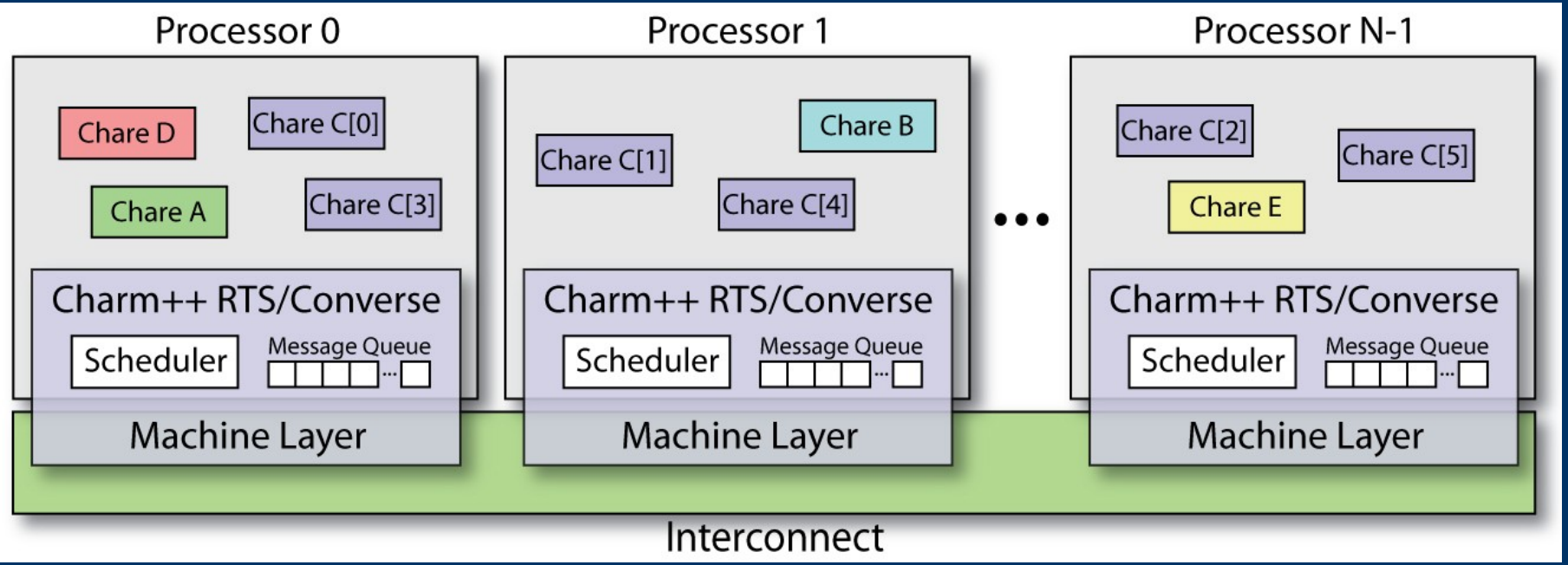
Example: cosmology



- Both sequential and parallel
- Extendable outside Charm++

Charm++ RTS

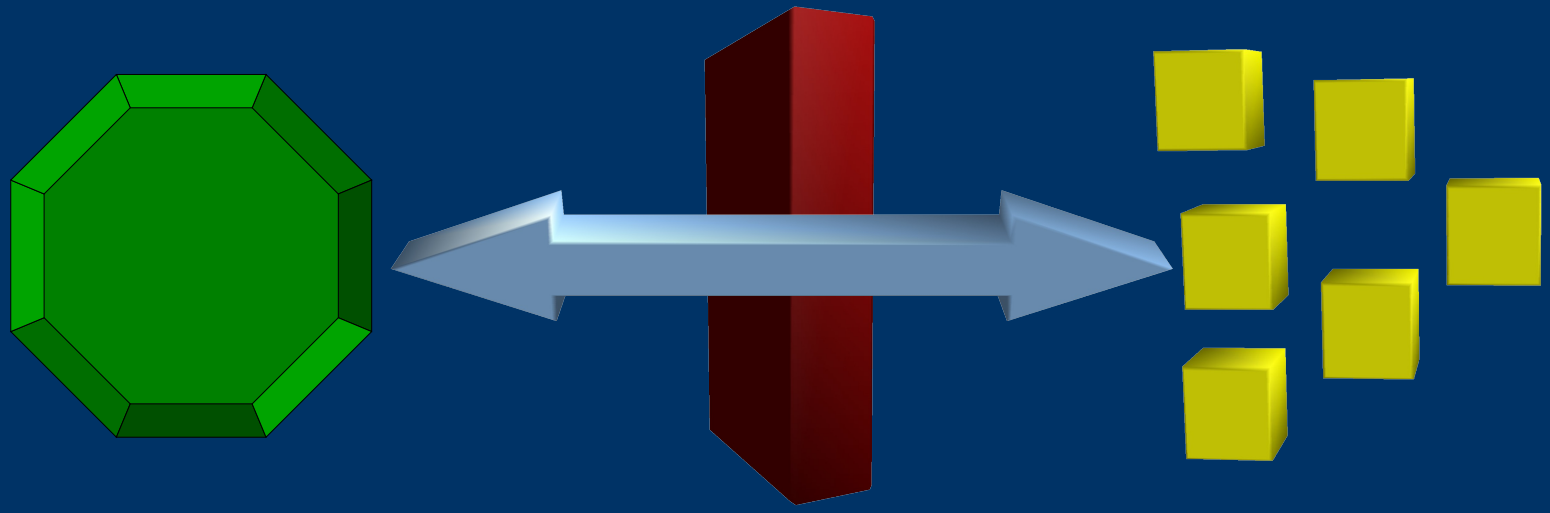
- Message contains:
  - destination chare ID
  - method to be invoked



CharmDebug Overview

- Developed specifically with Charm++ in mind
  - Provides information at the Charm++ abstraction level
- Composed of two modules:
  - Java GUI (client)
  - Plugin inside Charm++

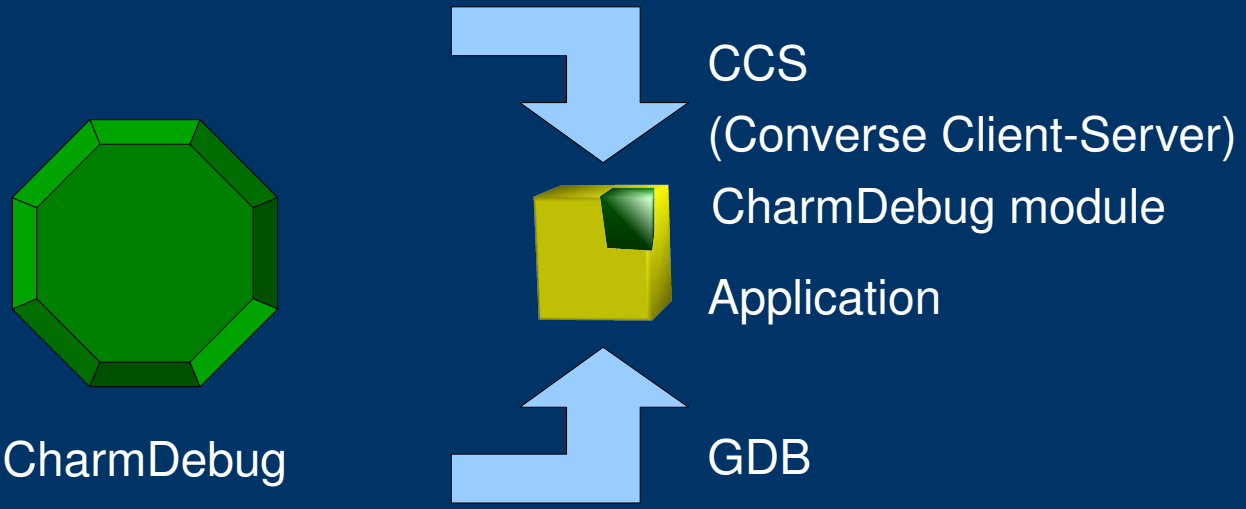
CharmDebug Architecture



CharmDebug Java GUI
(local machine)

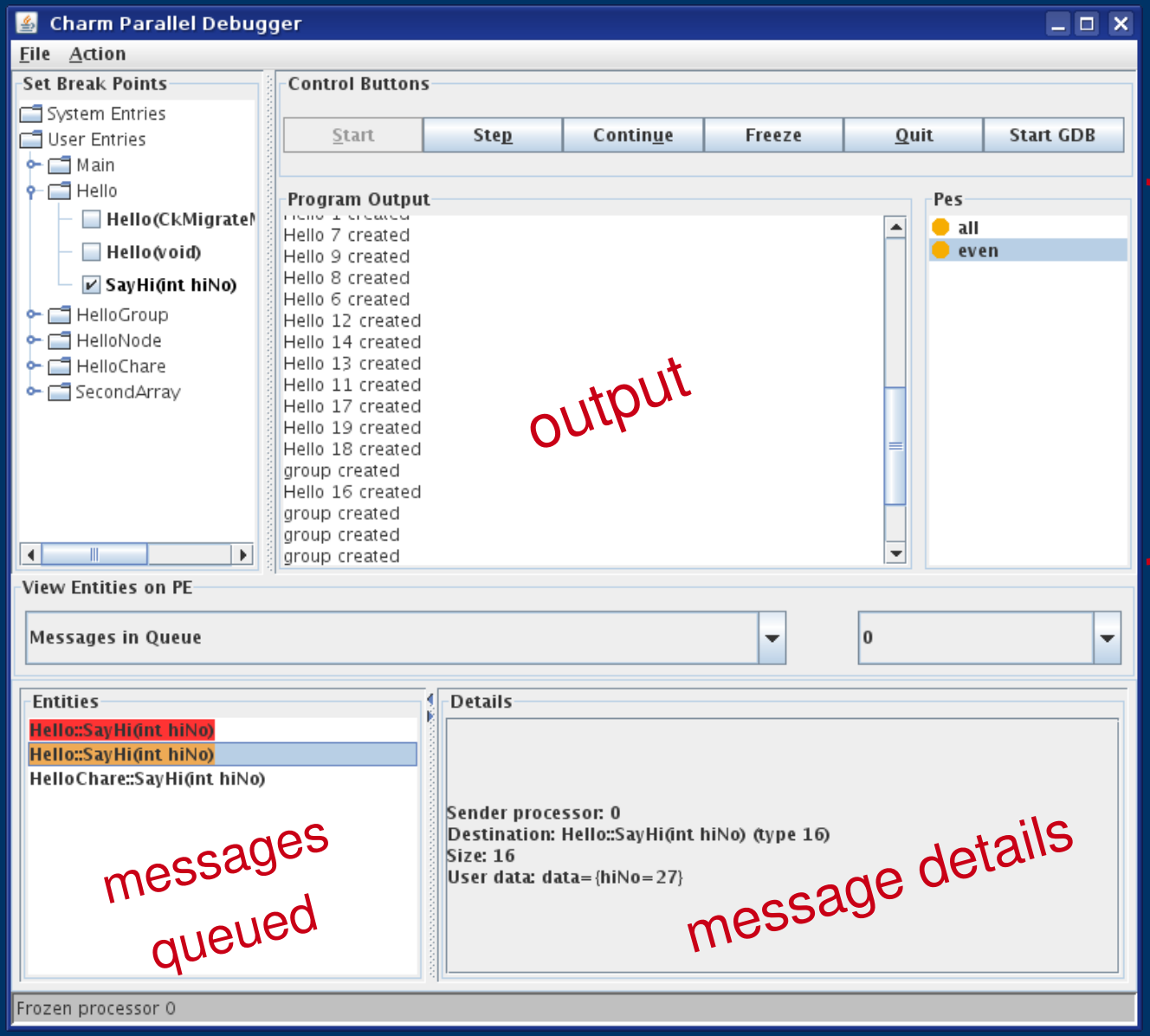
Firewall

Parallel Application
(remote machine)



CharmDebug

Main View



entry methods

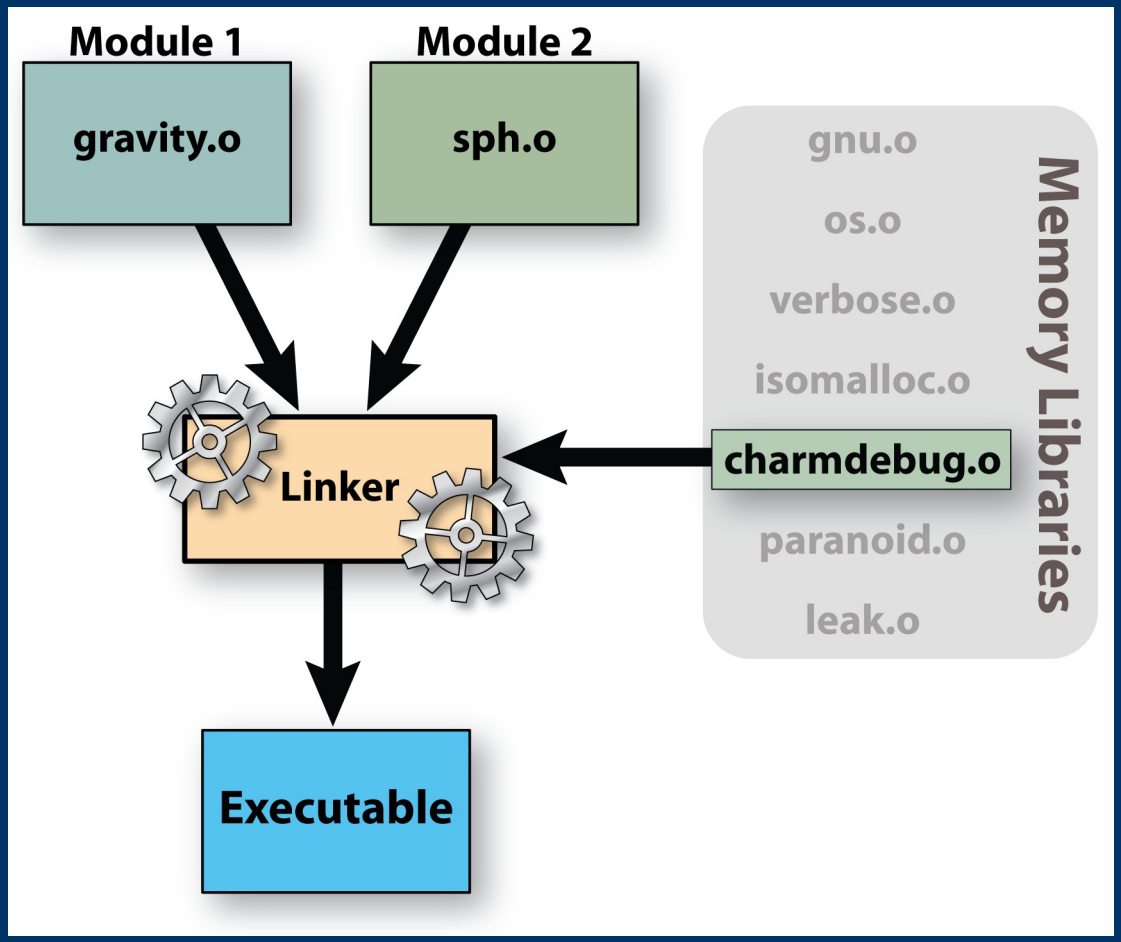
output

processor subsets

messages queued

message details

Charm++ Memory Subsystem

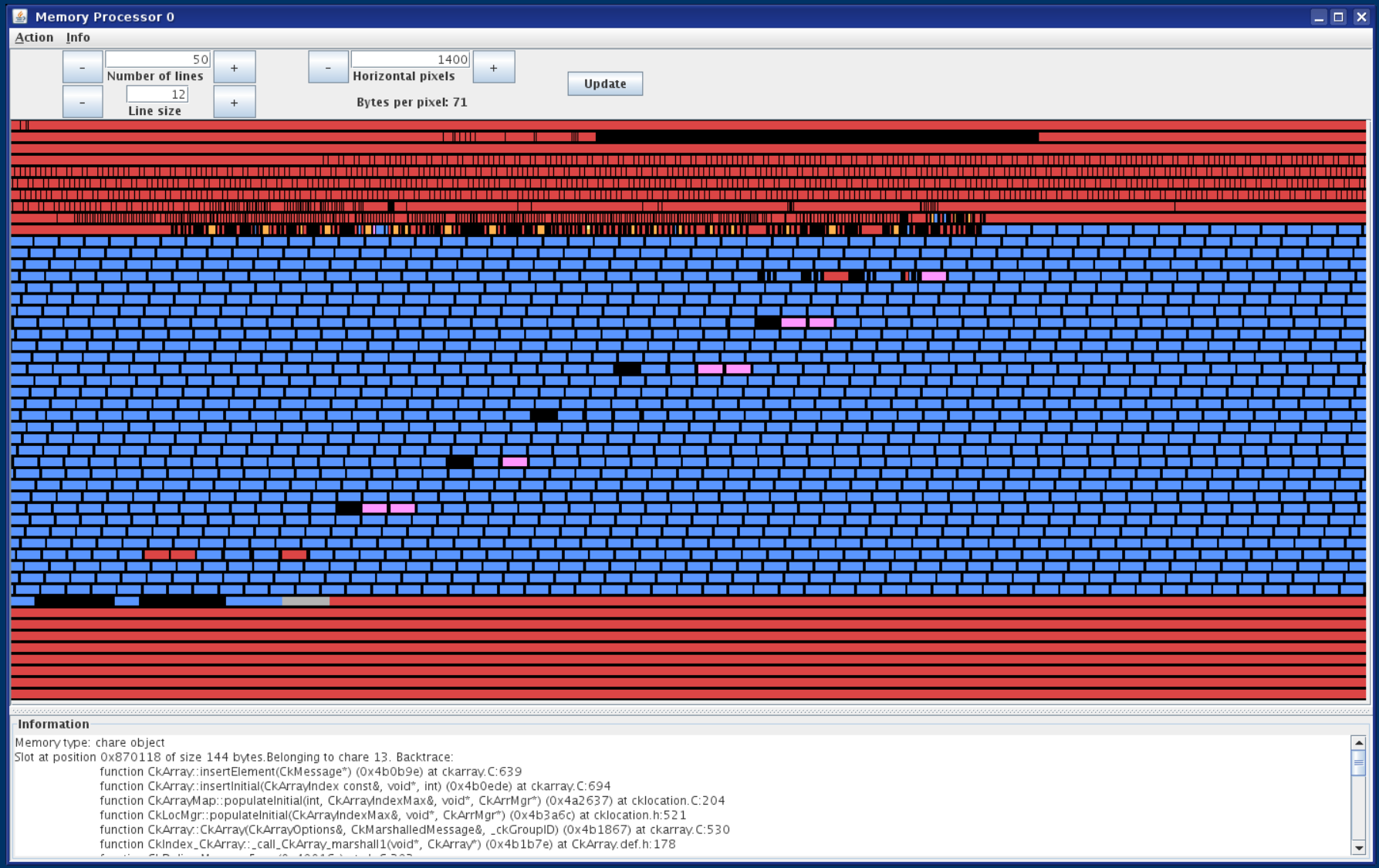


- At link time compile a memory library in
- Re-implement malloc, free, etc.
- Extend debugger capabilities

Memory Debugging

- Memory problems are typically subtle and hard to trace
- In Charm++ multiple chares reside on the same processor and share the same address space
- Focus
  - Memory leak
  - Cross-chare corruption

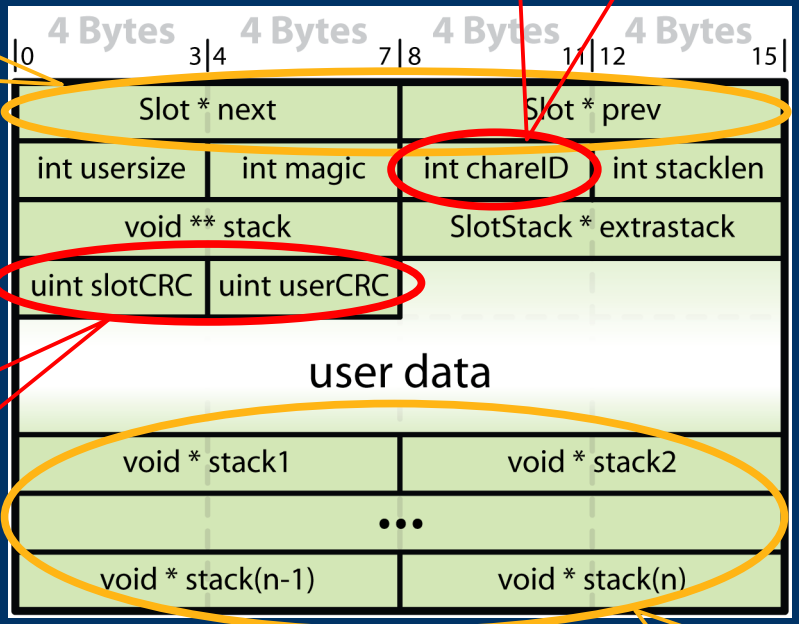
Memory view


Memory Processor 0
Action
Info
50
Number of lines
12
Line size
1400
Horizontal pixels
Bytes per pixel: 71
Update
Information
Memory type: chare object
Slot at position 0x870118 of size 144 bytes.Belonging to chare 13. Backtrace:
function CkArray::insertElement(CkMessage*) (0x4b0b9e) at ckarray.C:639
function CkArray::insertInitial(CkArrayIndex const&, void*, int) (0x4b0ede) at ckarray.C:694
function CkArrayMap::populateInitial(int, CkArrayIndexMax&, void*, CkArrMgr*) (0x4a2637) at cklocation.C:204
function CkLocMgr::populateInitial(CkArrayIndexMax&, void*, CkArrMgr*) (0x4b3a6c) at cklocation.h:521
function CkArray::CkArray(CkArrayOptions&, CkMarshalledMessage&, _ckGroupID) (0x4b1867) at ckarray.C:530
function CkIndex_CkArray::_call_CkArray_marshall1(void*, CkArray*) (0x4b1b7e) at CkArray.def.h:178

Memory Tagging

Associate to each memory block the ID of the owner chare

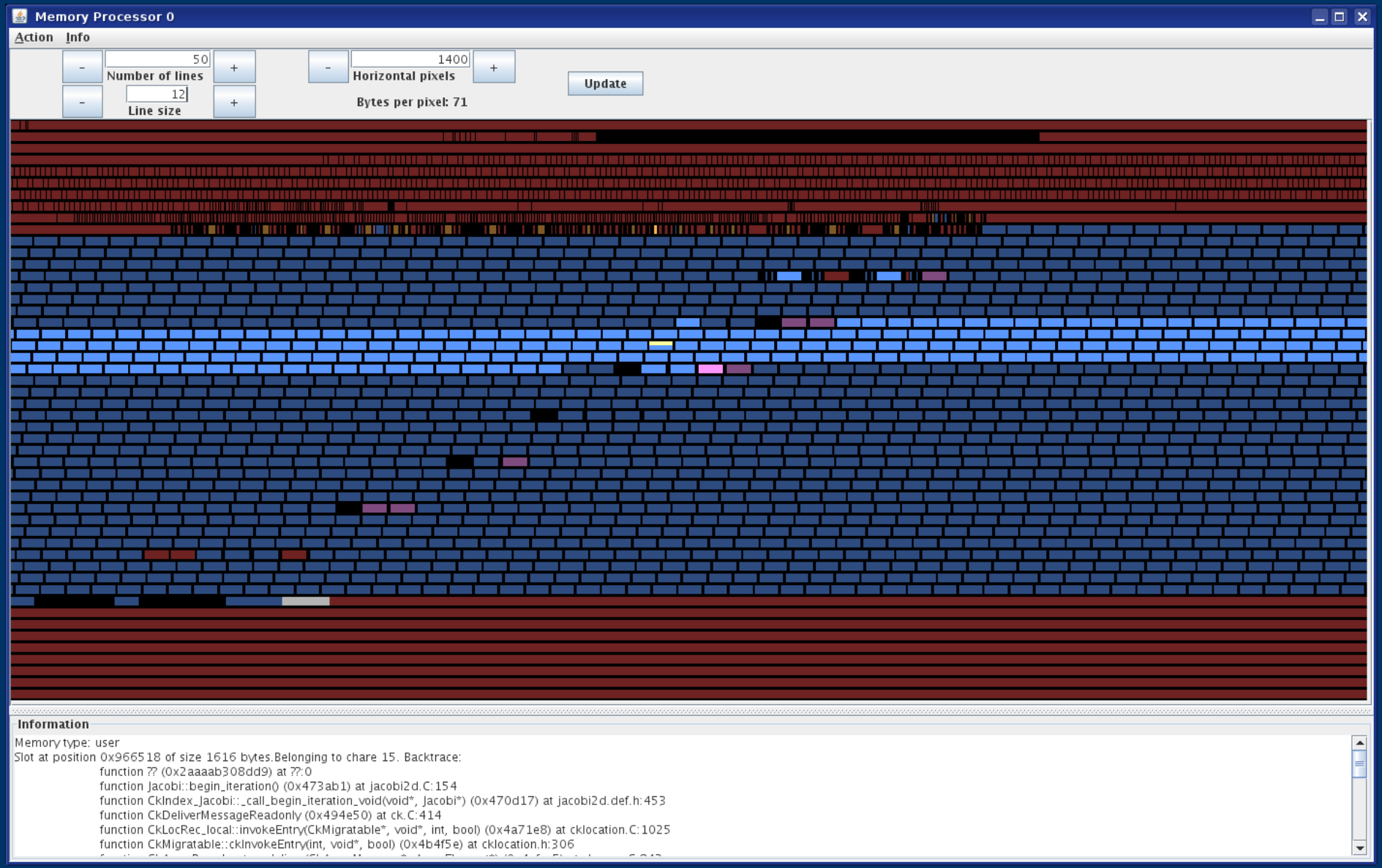
Linked list: All allocated memory is known by the debugger



Detect modification to the memory block

Stack trace of the moment where the block was allocated

View by Chare ID

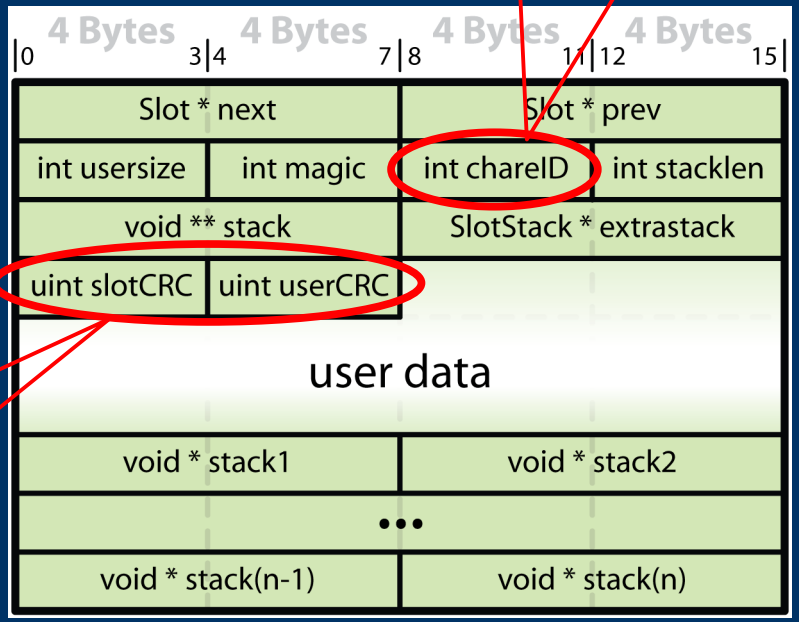


Cross-chare corruption

- A chare should access only to its data structures
  - Inside a processor the address space is shared
  - A chare can write some other chare's data
- Associate each chare an ID, and mark all its memory with that ID
- After an entry method, check if the chare modified some memory not belonging to it

Detecting violations

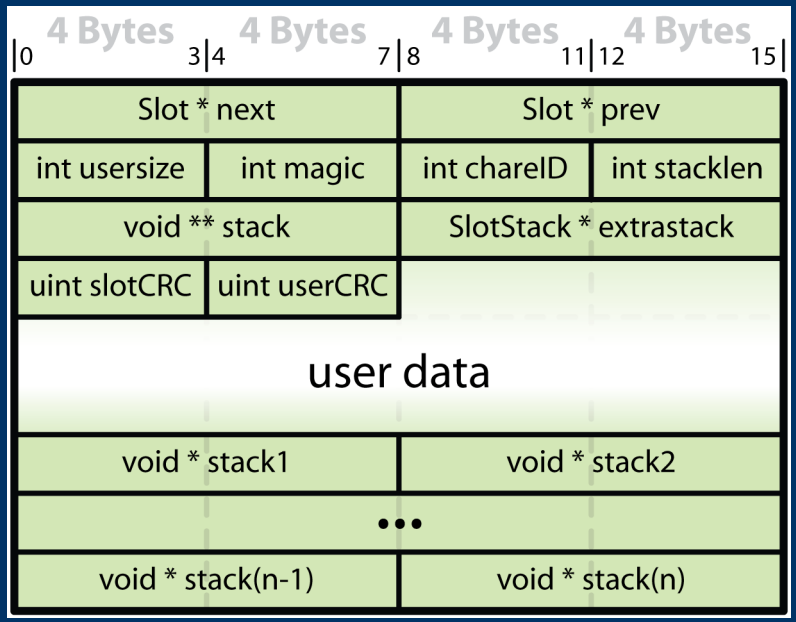
Associate to each memory block the ID of the owner chare



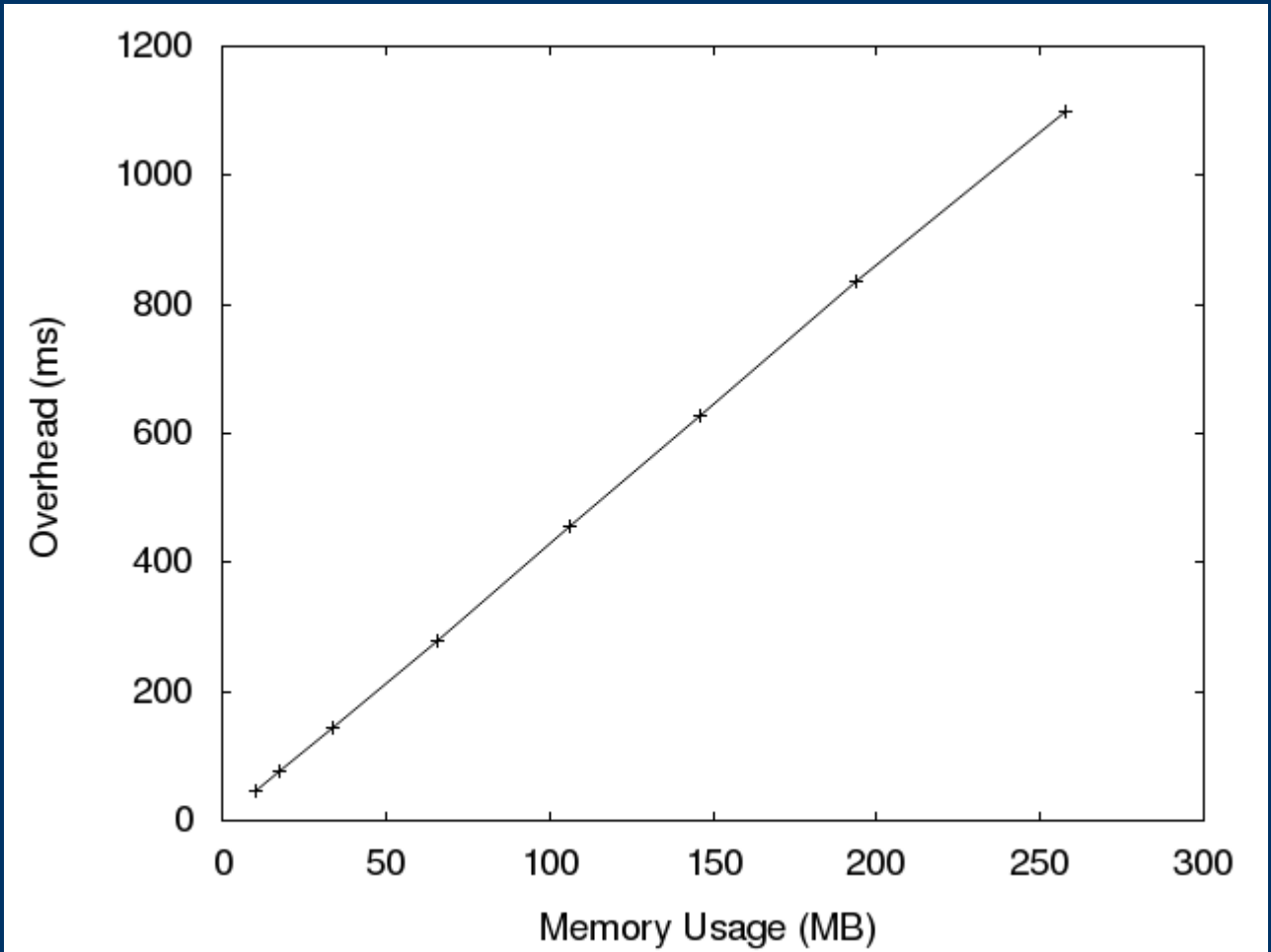
Detect modification to the memory block

Buffer overflow violations

- Utilize the other CRC field to protect the system data laying around the user data
- Detect violations similarly to cross-chare violations



Performance



$$1 + \frac{4.3 * M}{t}$$

M: allocated memory

t: interval between methods

Future directions

- Reduce checking time
  - Reduce frequency of checking
    - Check only some entry methods
  - Reduce amount of memory scanned
    - Check only some memory
  - Reduce time to scan
    - Faster error detection codes
    - Memory shadowing
    - mmap and mprotect
      - allows detection of reads as well as writes

Future directions (2)

- Applying technique to real applications
- Presenting results of test to the user
  - Allow multiple ownership of data blocks
  - Filter errors (simple warnings)
- Re-execute erroneous code
  - Roll-back to state before message delivered
  - Re-run under more detailed debugging
  - Connected to live record-replay

Questions?

Thank you

http://charm.cs.uiuc.edu/