Debugging Tools
for
Charm++ Applications

Filippo Gioachin

University of Illinois at Urbana-Champaign

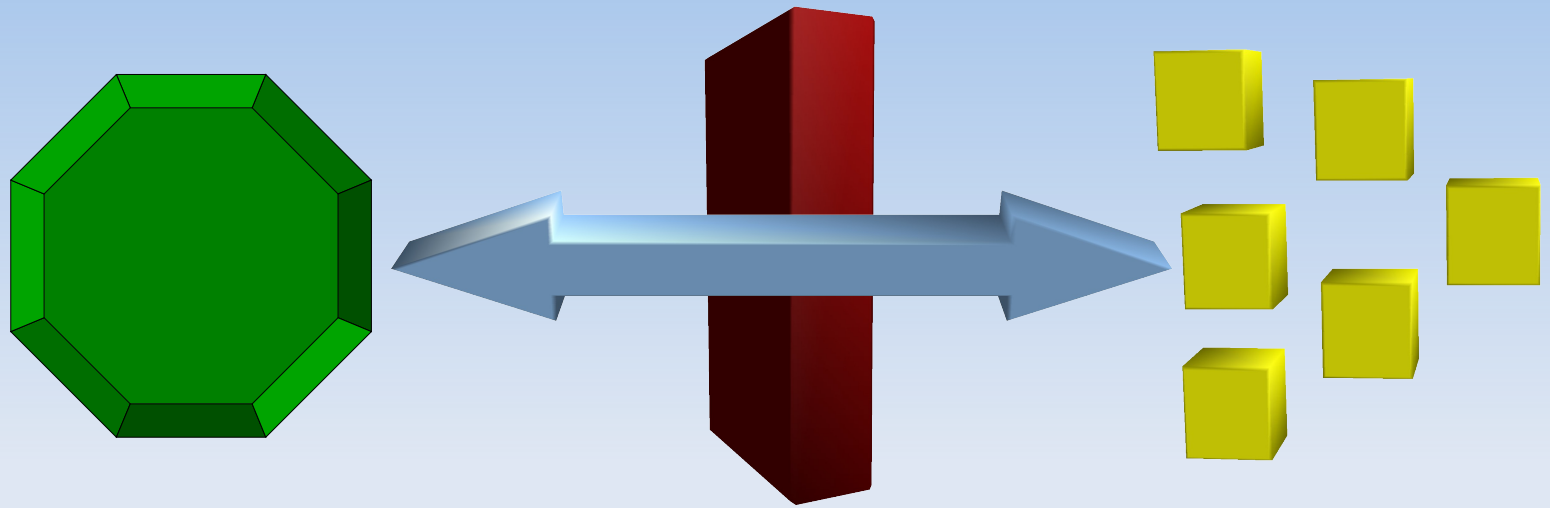
Outline

- Overview of CharmDebug
- Introspection
- Memory Debugging
  - Leak detection
  - Cross-chare corruption detection

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++

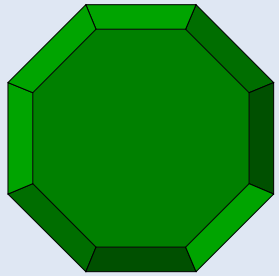
CharmDebug Architecture



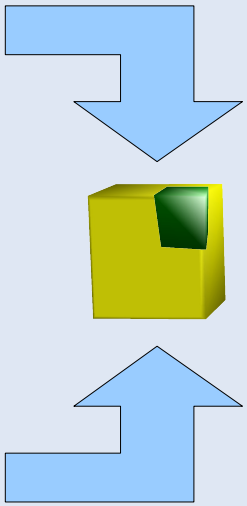
CharmDebug Java GUI
(local machine)

Firewall

Parallel Application
(remote machine)



CharmDebug



CCS
(Converse Client-Server)

CharmDebug module

Application

GDB

Introspection

- The capability of a system or application to inspect its own state.
- For an application this includes the capability to
  - Identify the type of a variable;

```
MyType var;
```

  - Identify the layout of a type, and browse it.

```
class MyType {
    int a;
    double b;
}
```

Introspection in Charm++

- Interactivity
  - Want to run introspection code on demand
  - “Are the values in my array getting too big?”
- Use CharmDebug's connection to the app.
- Charm++ Python interface

Charm++ Python Interface

- Send Python code through a CCS channel
- Execute the code on the parallel application

```
group [python] myGroup {
    entry [python] void getValue();
}
```

  - Code is bound to a chare element (myGroup)
  - Python can use method getValue

```
def check(self):
   value = charm.getValue(data, 10)
```

Browsing variables

```
class MyClass {
    int length;
    double * data;
    MyExtraData * extra;
}
```

- C++ is not reflective
  - Cannot access data structure layout
- We use the information already collected in CharmDebug to provide the extra information needed at runtime

Introspection with CharmDebug

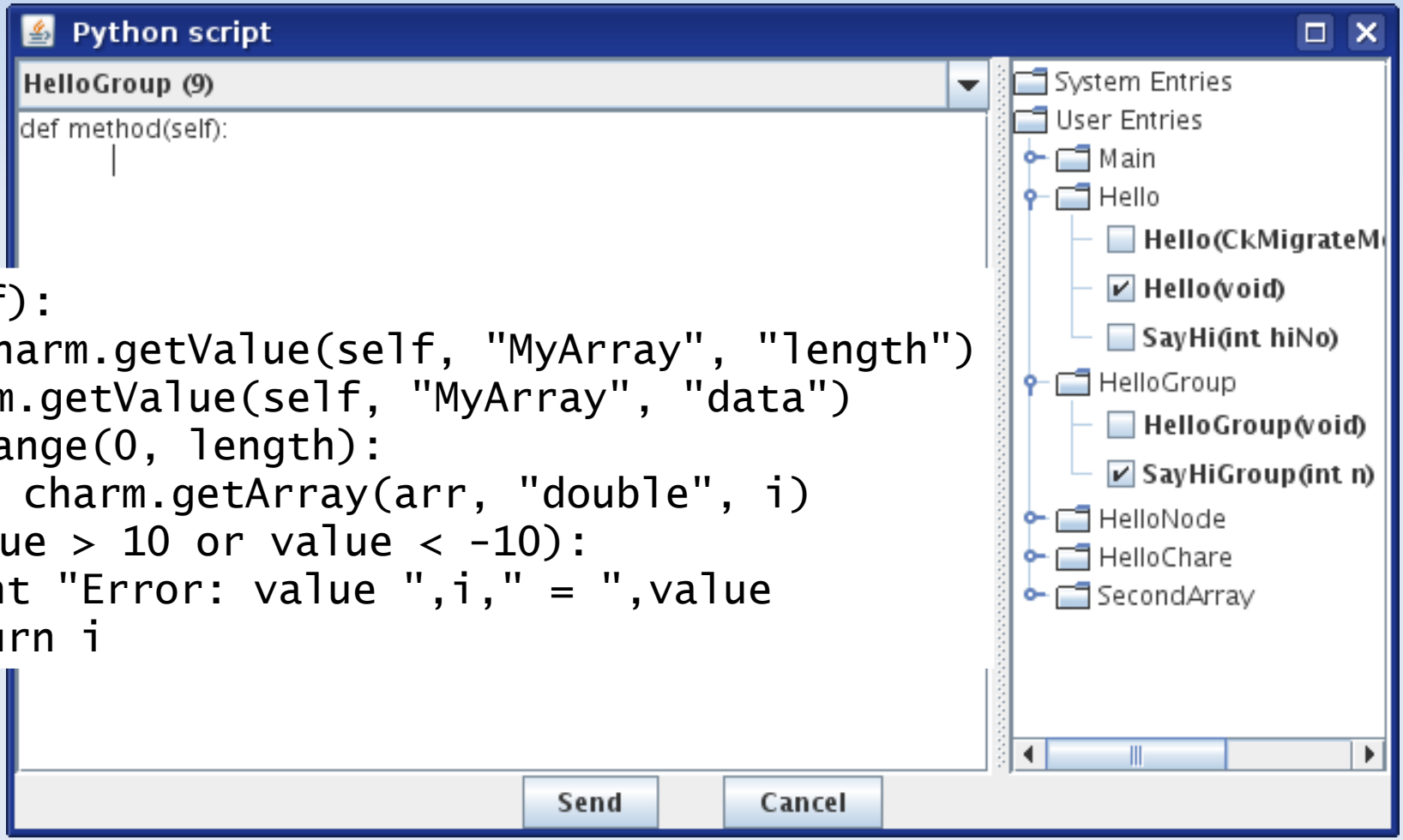
- A *group* part of CharmDebug plugin is bound to Python
  - No need to modify user code
- Set of functions exported:

```
group [python] CpdPythonGroup {
   entry [python] void getValue();
   entry [python] void getArray();
   entry [python] void getCast();
   entry [python] void getStatic();
}
```

An Example

- Are the values in my array getting too big?

```
def check(self):
    length = charm.getValue(self, "MyArray", "length")
    arr = charm.getValue(self, "MyArray", "data")
    for i in range(0, length):
        value = charm.getArray(arr, "double", i)
        if (value > 10 or value < -10):
            print "Error: value ",i," = ",value
            return i
```

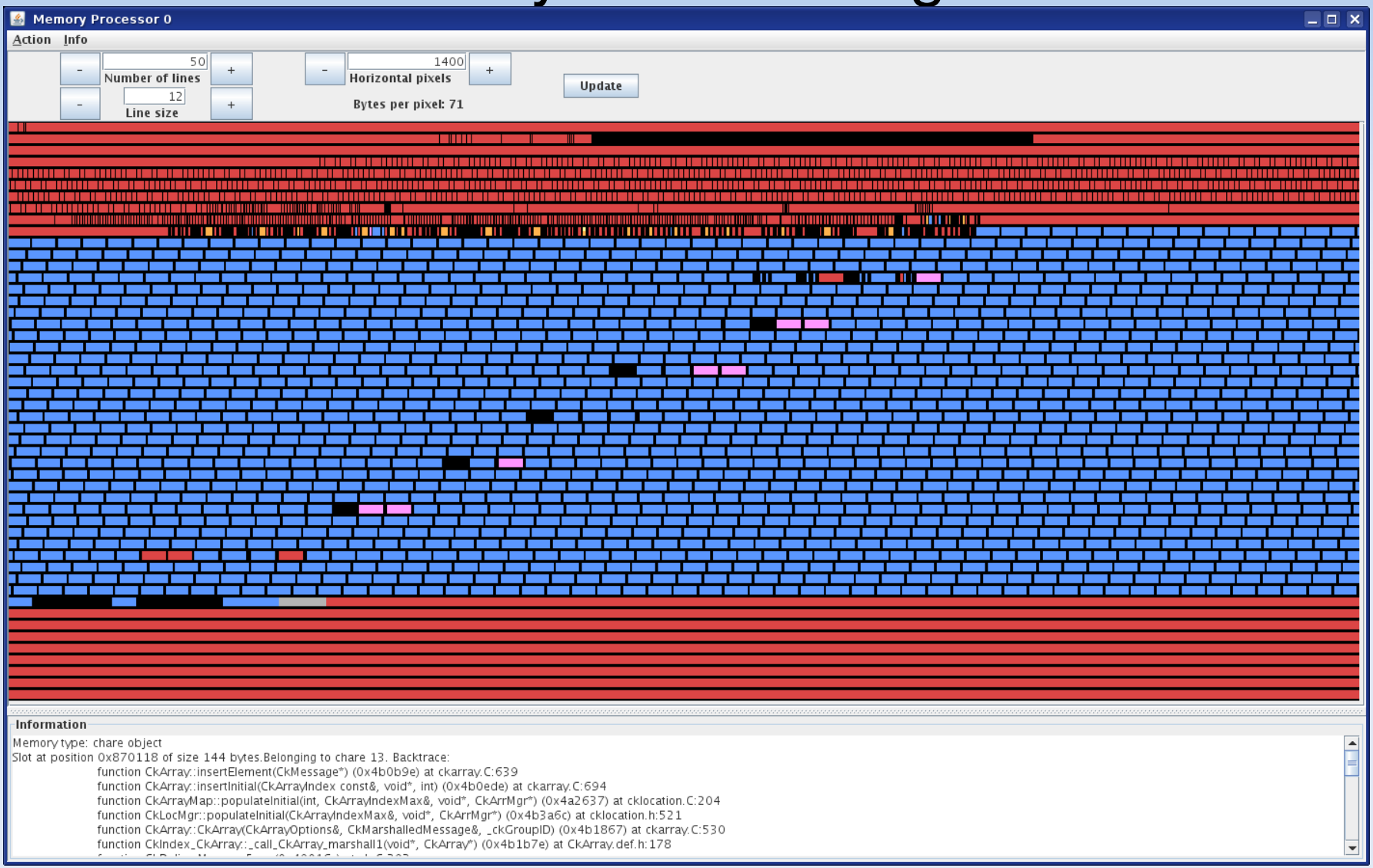


Memory Debugging

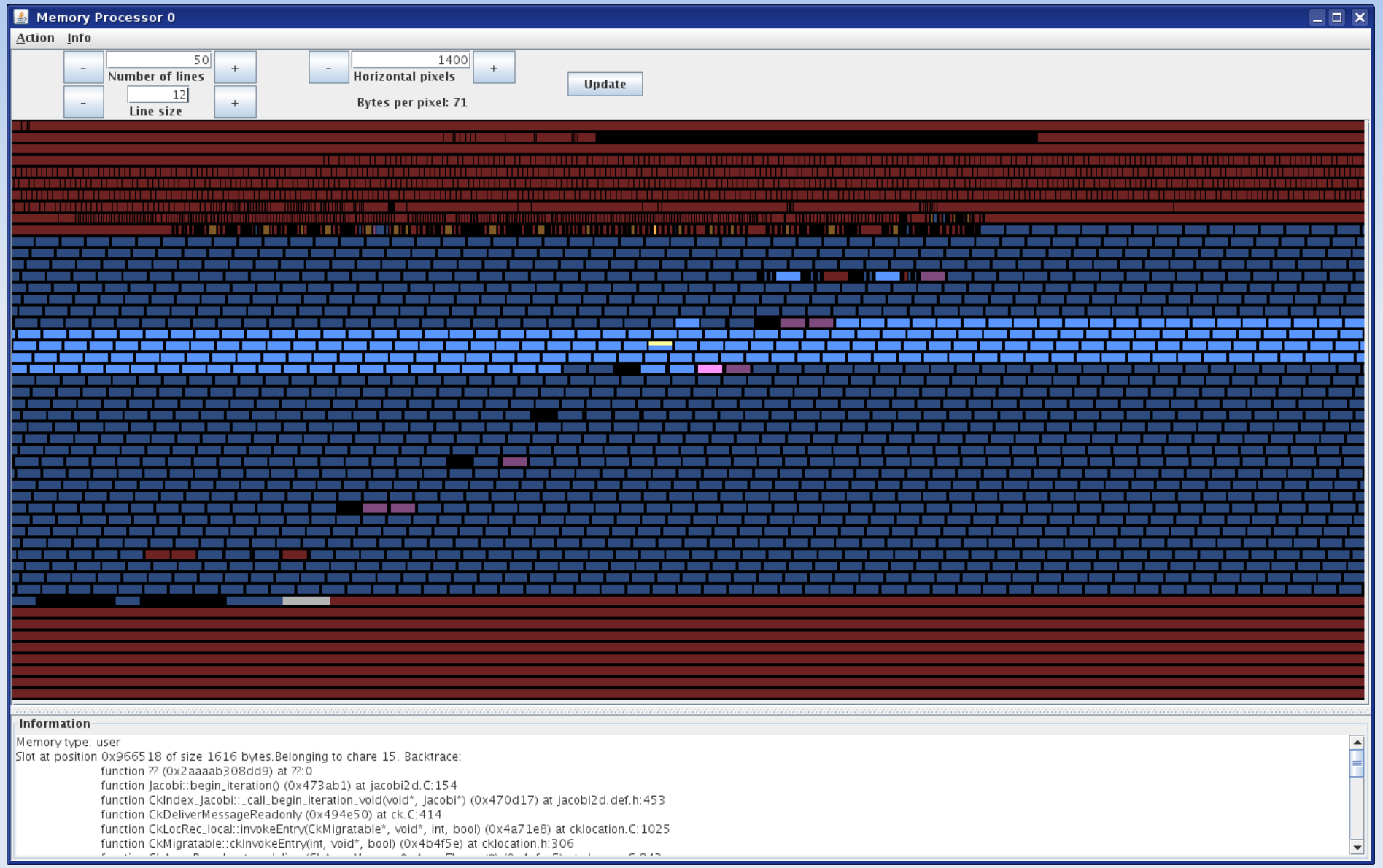
- Memory problems are typically subtle and hard to trace
- In Charm++ multiple chares reside on the same processor and share the same address space
- Focus
  - Memory leak
  - Cross-chare corruption

Main View

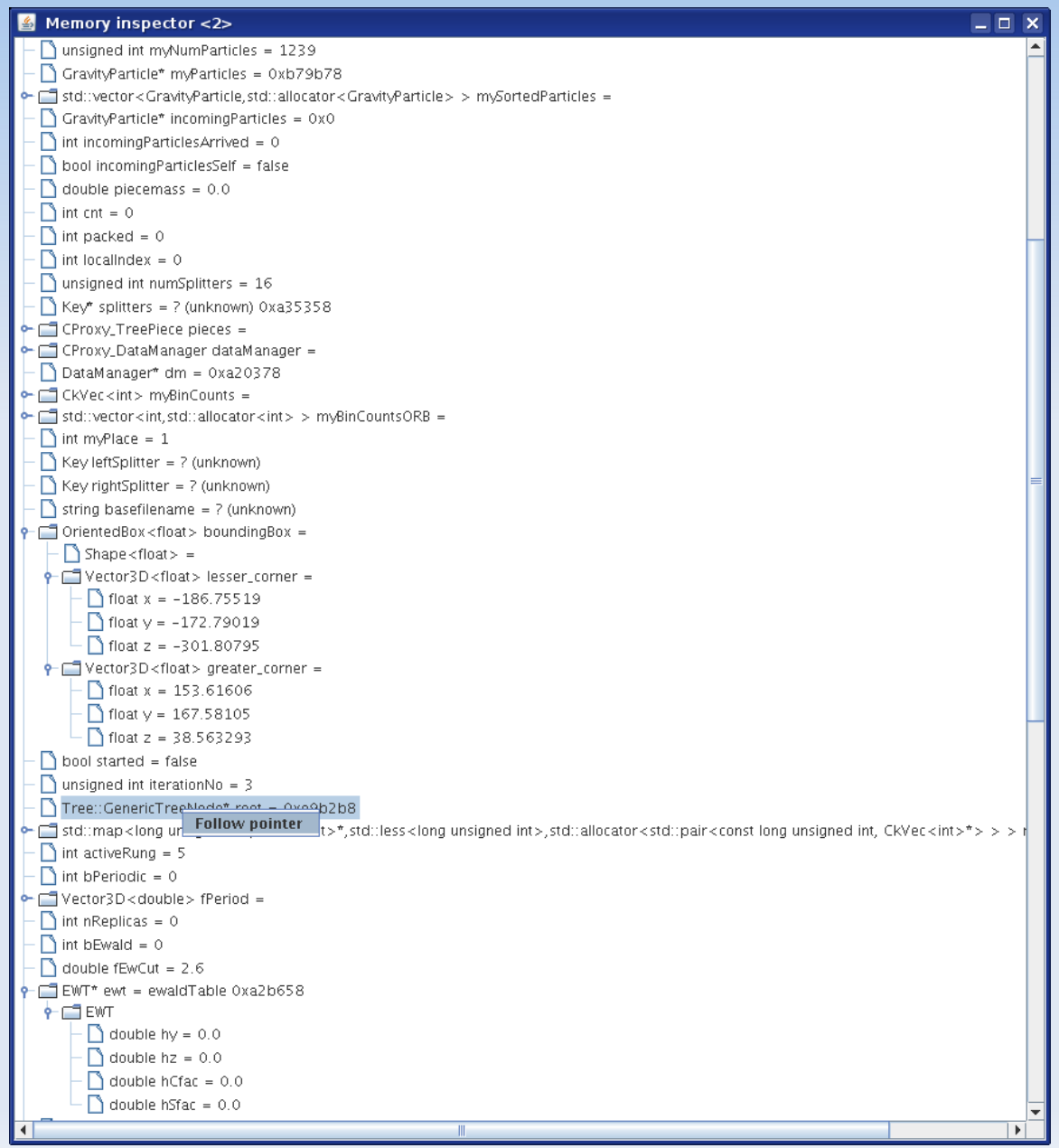
- link with “-memory charmdebug”



View by Chare ID



Inspecting Memory

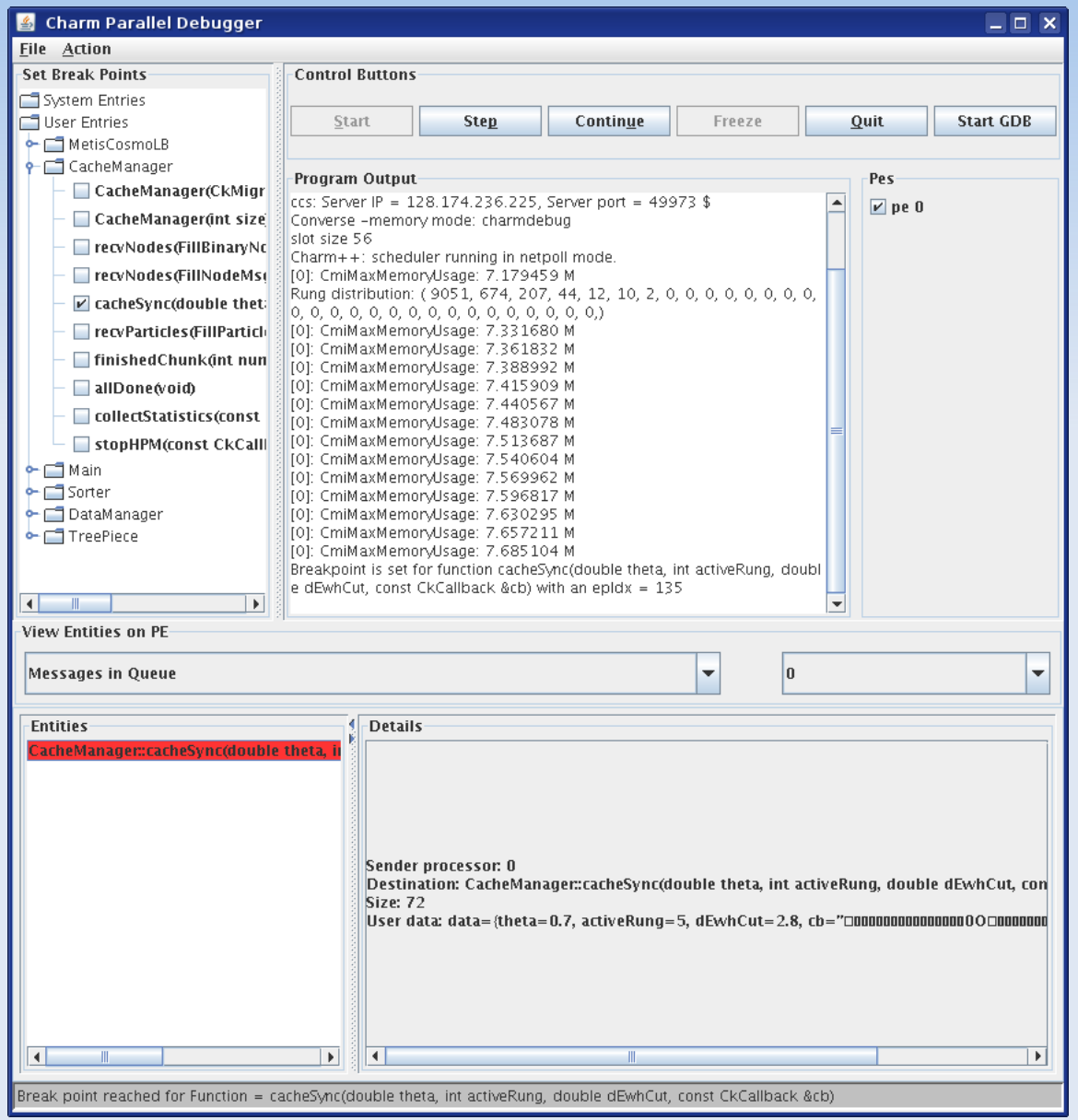


- Example from ChaNGa
- Inspect memory
- Data is loaded in the client
- Still evolving

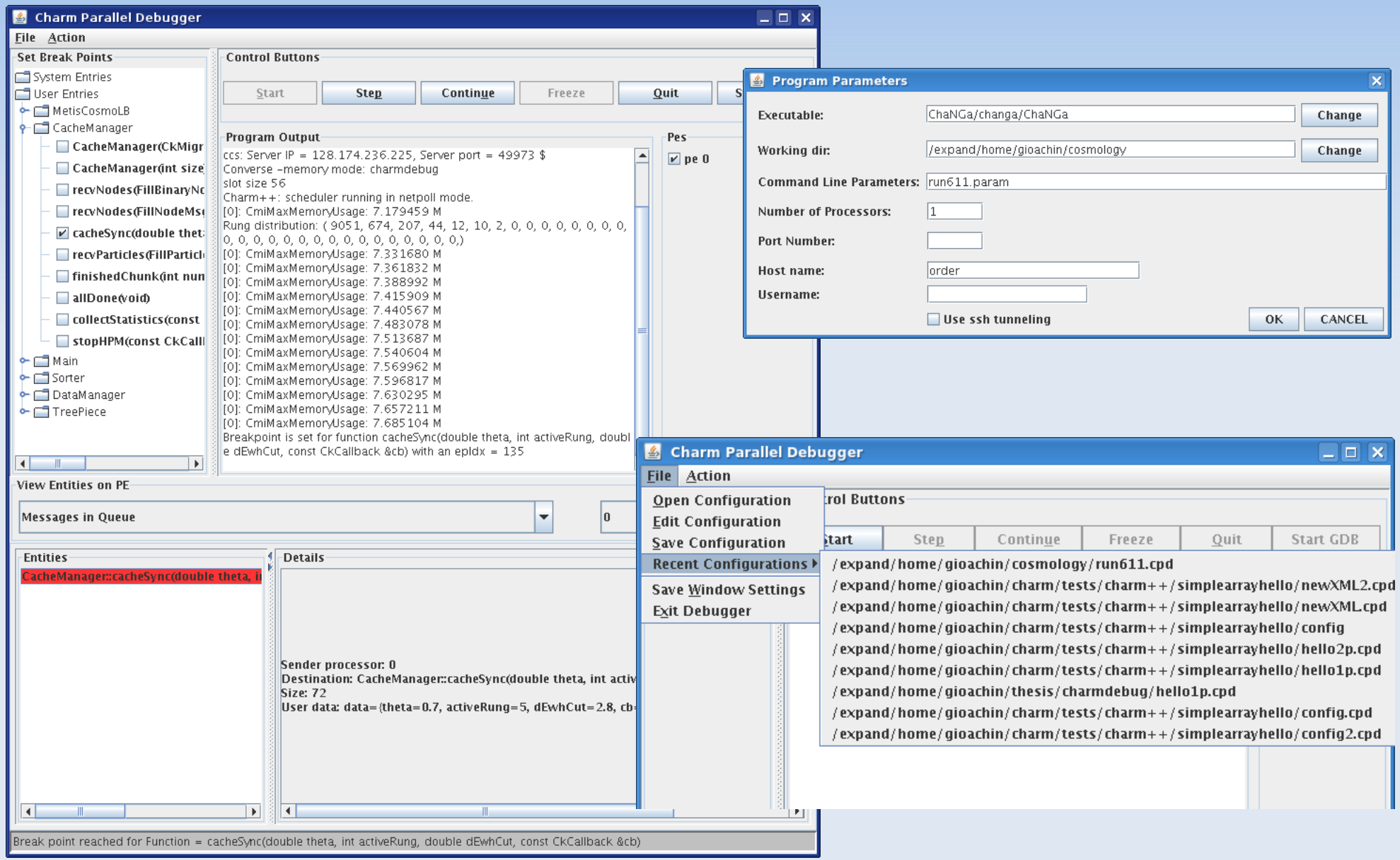
Searching for Leaks...

- ChaNGa
  - Cosmological simulator
  - Runs in a step-wise fashion
- Observed effect: the allocated memory keeps increasing
- We ran the application for a few steps, then froze it

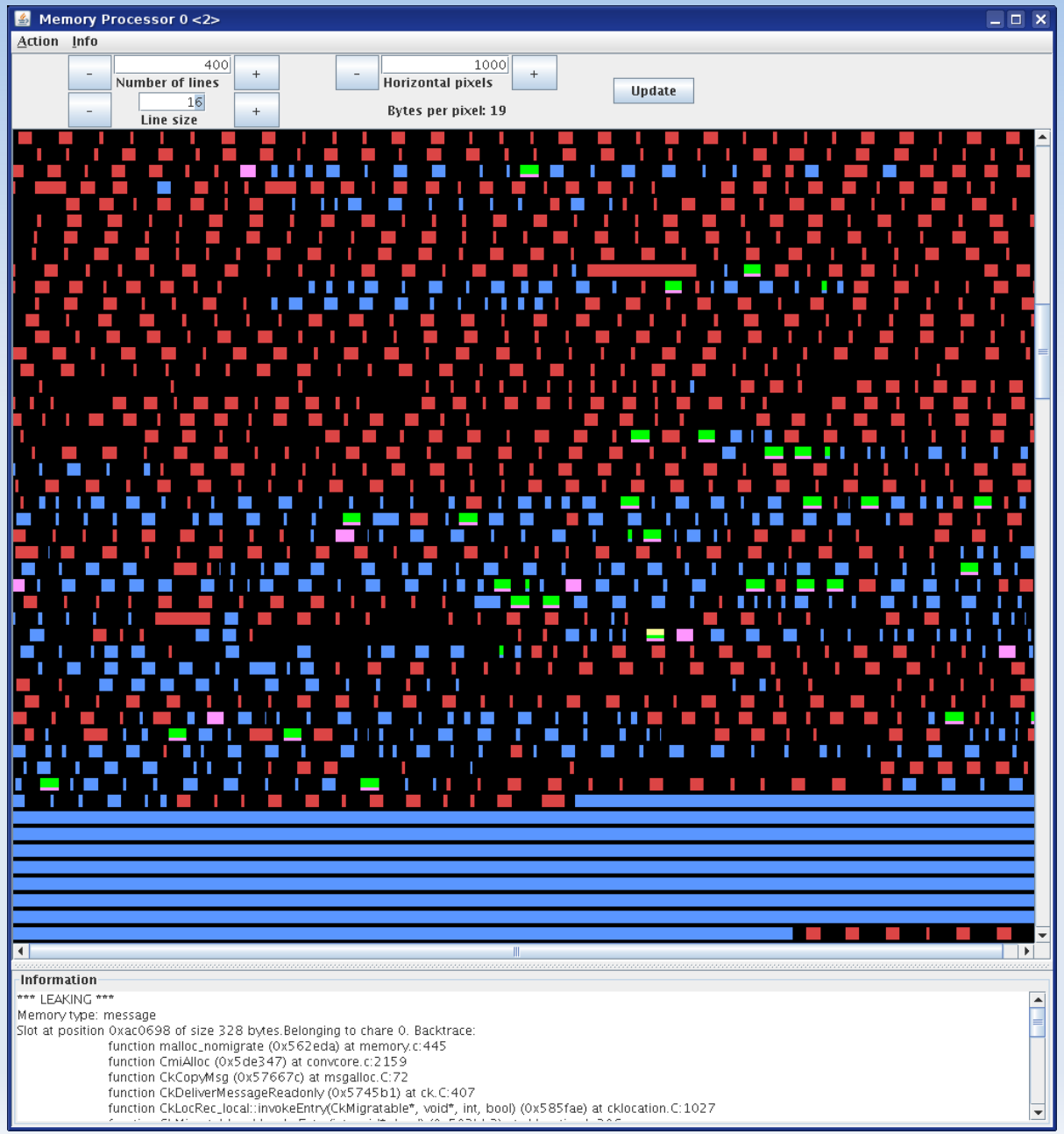
Searching for Leaks... (2)



Searching for Leaks... (2)



Searching for Leaks... (3)



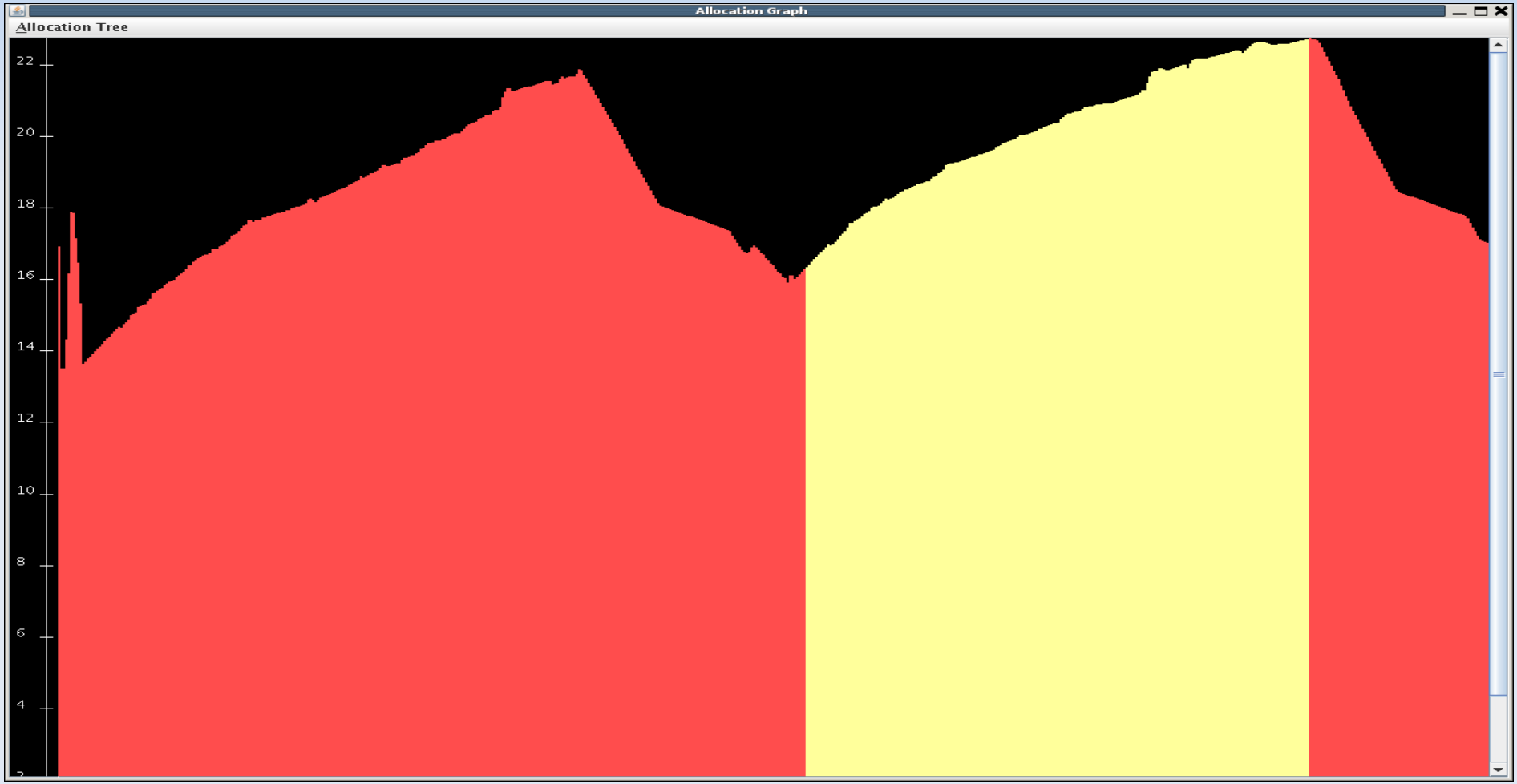
- Leaks are in green
- Messages are leaking (pink)

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

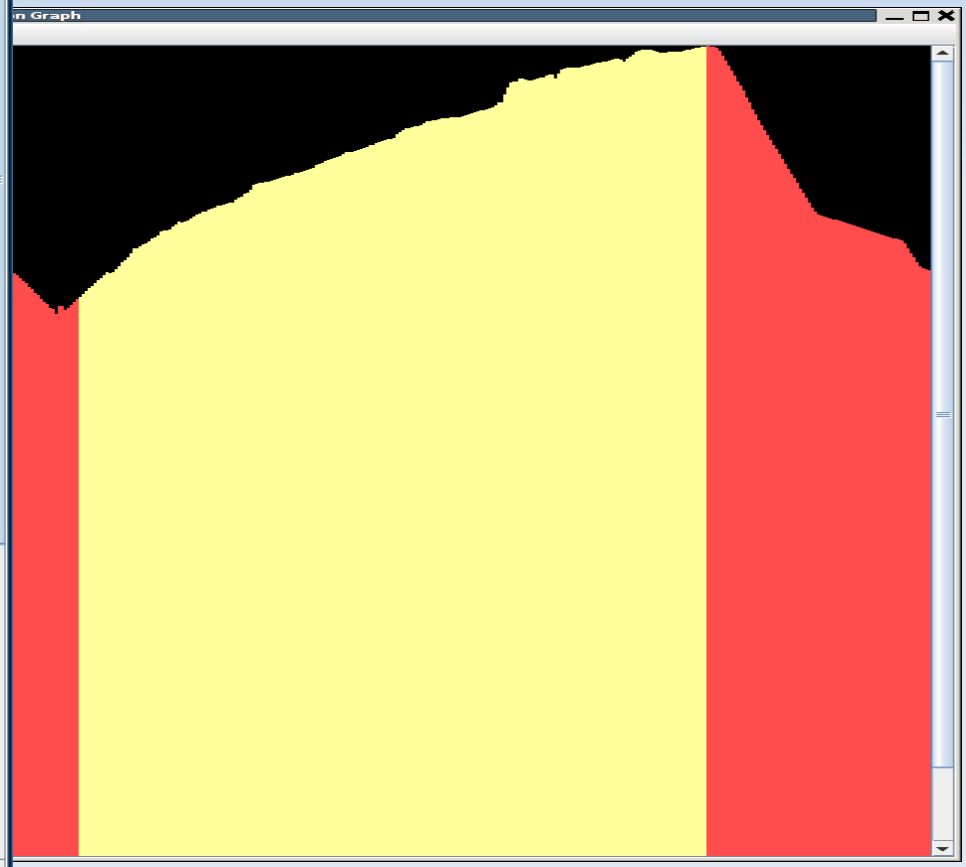
Allocation Graph

- link with “-tracemode memory”



Allocation Graph & Allocation Tree

...nemory”



Questions?

Thank you