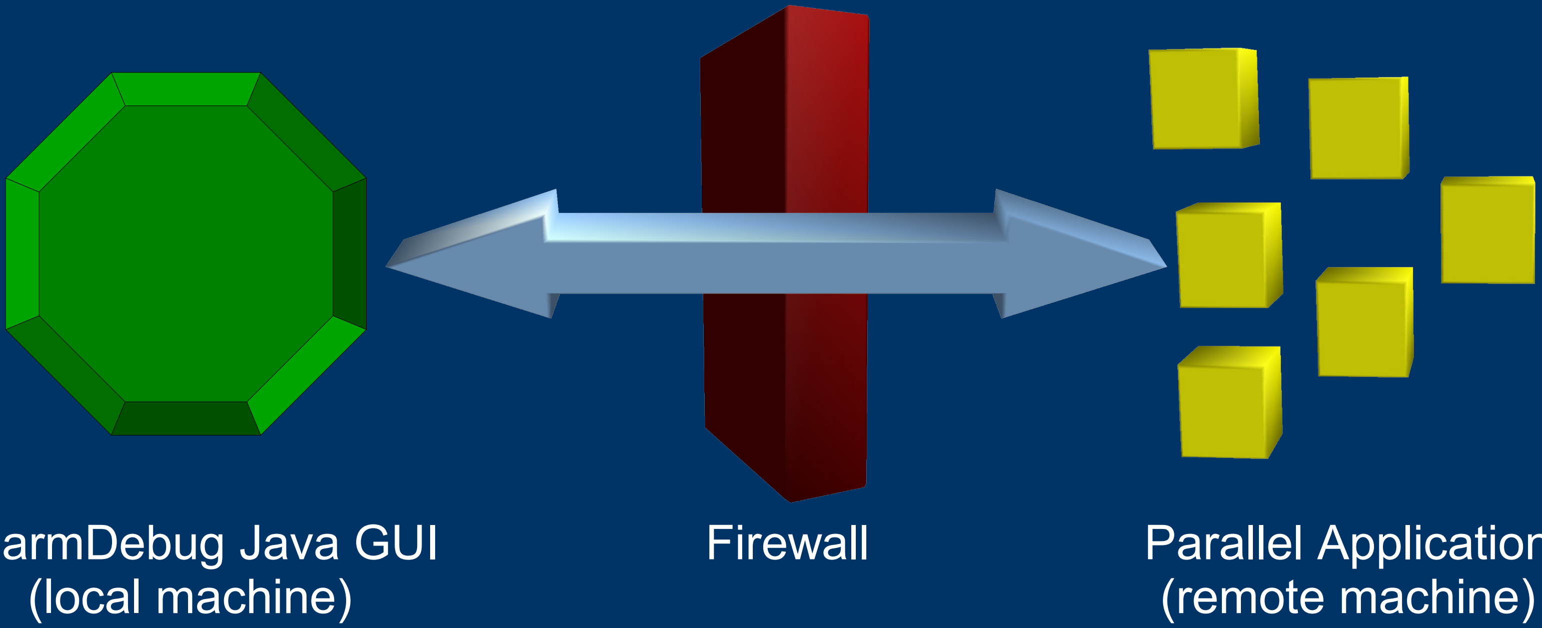
CharmDebug

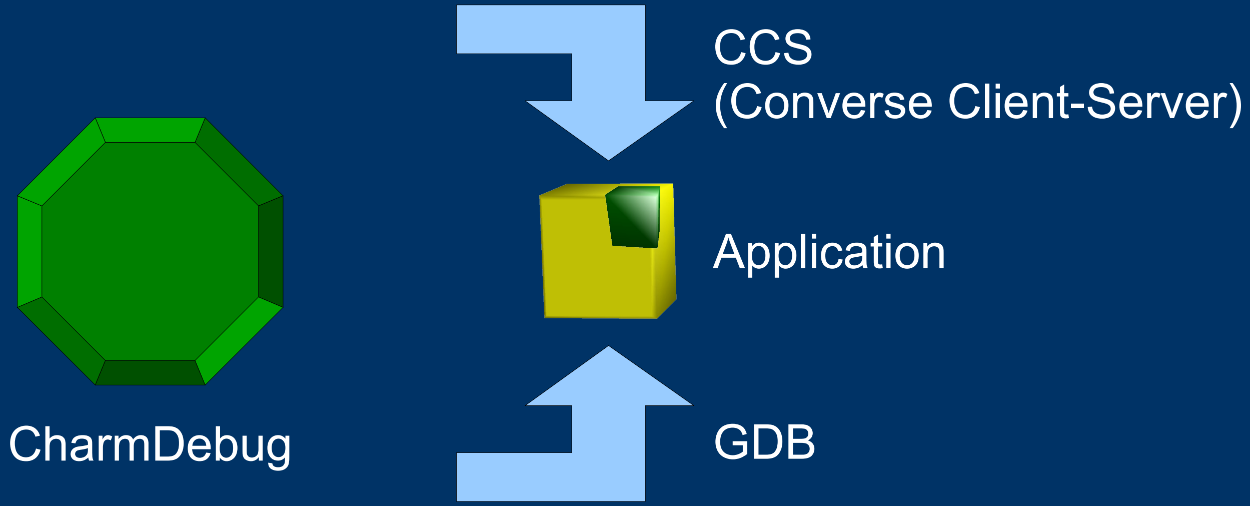
Filippo Gioachin

Outline

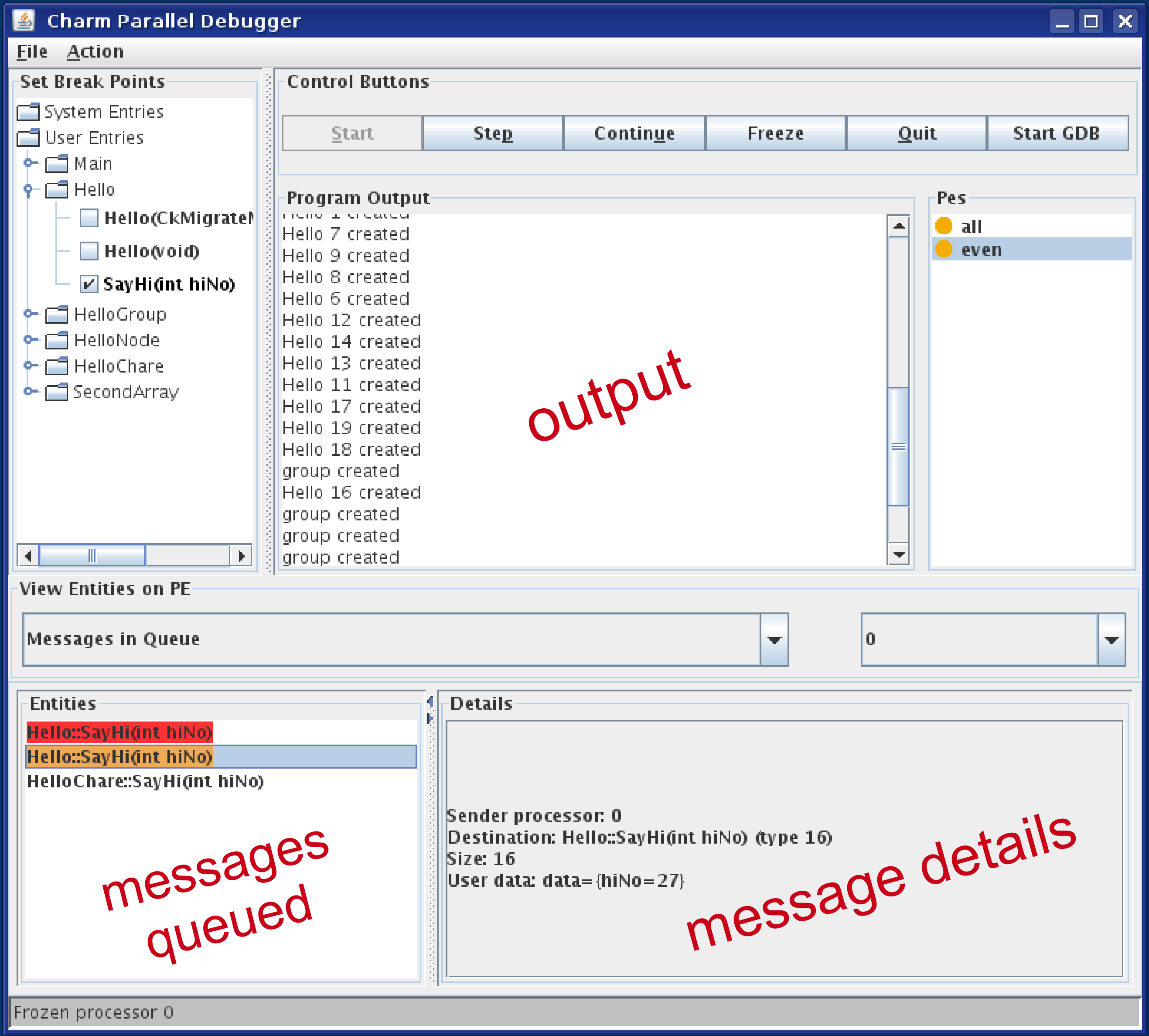
- Overview
  - Compilation
  - Startup
- Debugging
  - Incorrect values
    - Python scripting
  - Memory leak
- Miscellaneous
  - Breakpoints
  - Processor sets
  - Record/replay

Overview


CharmDebug Java GUI
(local machine)
Firewall
Parallel Application
(remote machine)


CCS
(Converse Client-Server)
Application
GDB
CharmDebug

Main Program View



```
Hello 7 created
Hello 9 created
Hello 8 created
Hello 6 created
Hello 12 created
Hello 14 created
Hello 13 created
Hello 11 created
Hello 17 created
Hello 19 created
Hello 18 created
group created
Hello 16 created
group created
group created
group created
```

Hello::SayHi(int hiNo)
Hello::SayHi(int hiNo)
HelloChare::SayHi(int hiNo)

```
Sender processor: 0
Destination: Hello::SayHi(int hiNo) (type 16)
Size: 16
User data: data={hiNo=27}
```

Frozen processor 0

entry methods

output

processor subsets

messages queued

message details

Getting charmdebug

- It is part of Charm++
  - charm/java
- Precompiled for java 6
  - `ant` to recompile
- Help
  - Manual (outdated)
  - charm@cs.uiuc.edu (preferred)
  - ppl@cs.uiuc.edu
  - gioachin@uiuc.edu
- Here we use Charm++ version 6.1.2

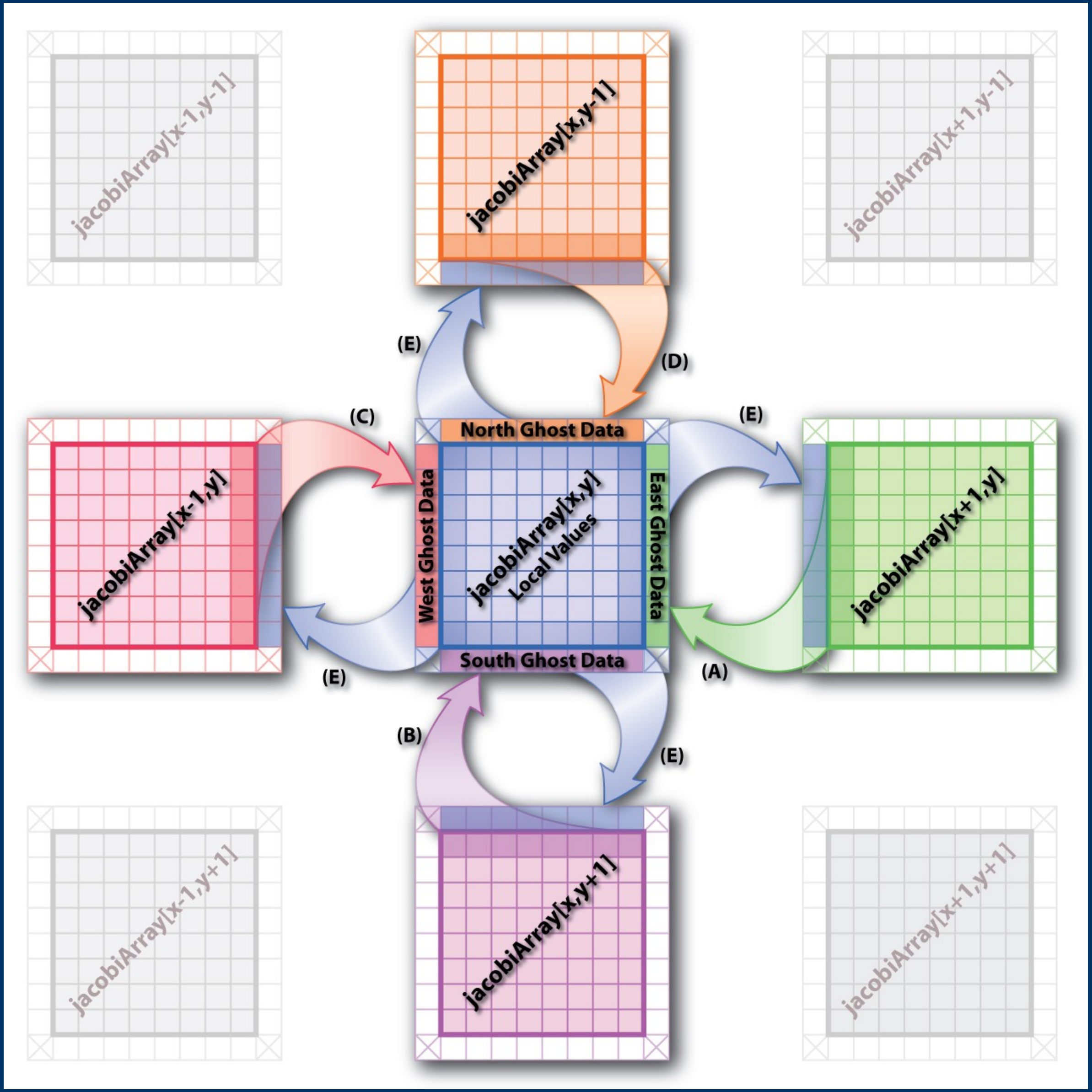
Compiling your application

- Charm++
  - Use `-g`
  - No `-O3` or `-DCMK_OPTIMIZE`
- Application
  - `-debug`
    - Adds `-g -O0`, -`memory charmdebug`, Python modules
    - Other memory options:
      - `os-charmdebug`
      - `hooks-charmdebug`
- Running
  - `+netpoll`
  - Or set CMK_NETPOLL in conv-mach.h

Starting an application

- Attach to running application in net- build
  - Uses CCS to receive application output
- Attach to running application in other builds
  - Read the output file of the application
- Start a new application in net- build
  - Can use tunnels
- Options available also in command line
  - Use `charmdebug -help` to see them

Jacobi 2D (5-point stencil)

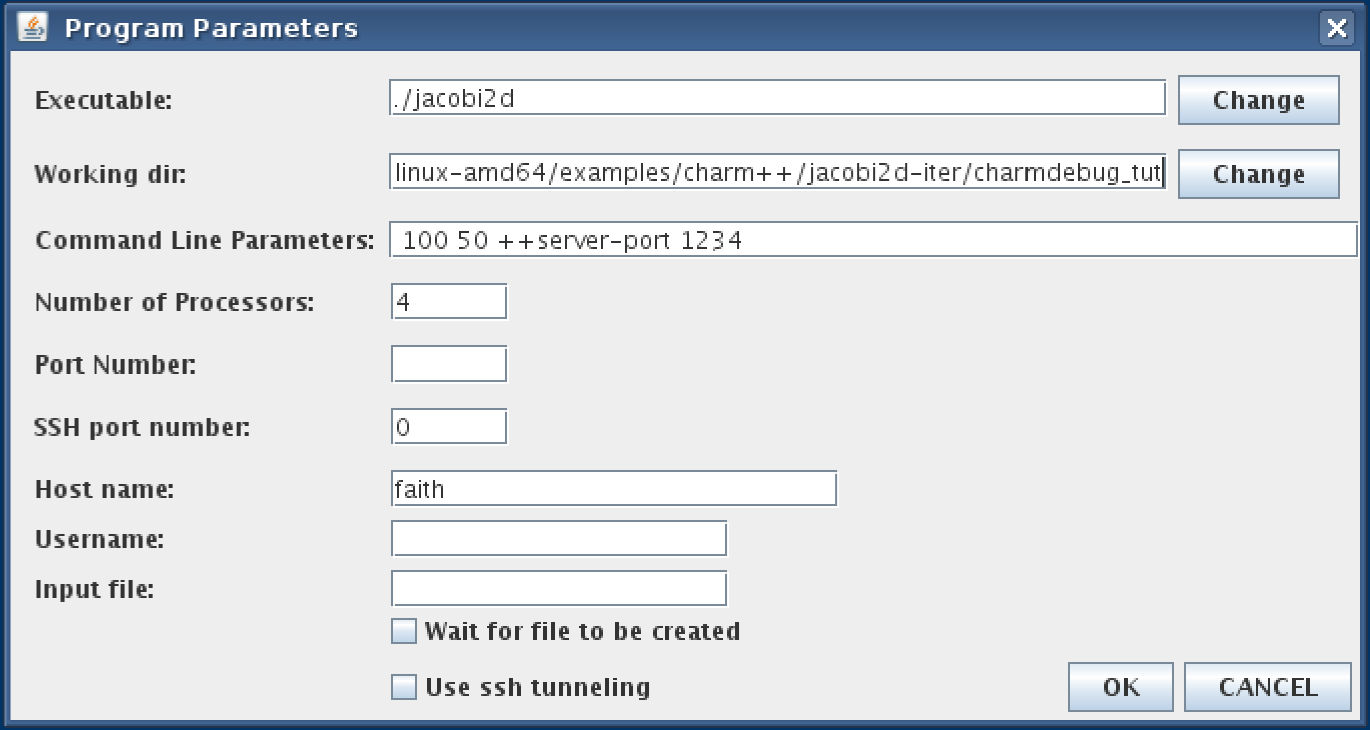


Python functions

- `getStatic(name)`
- `getCast(obj, type, newtype)`
- `getValue(obj, type, name)`
- `getArray(obj, type, num)`
- `getMessage()`

- Return value to freeze application

Snapshots from demo

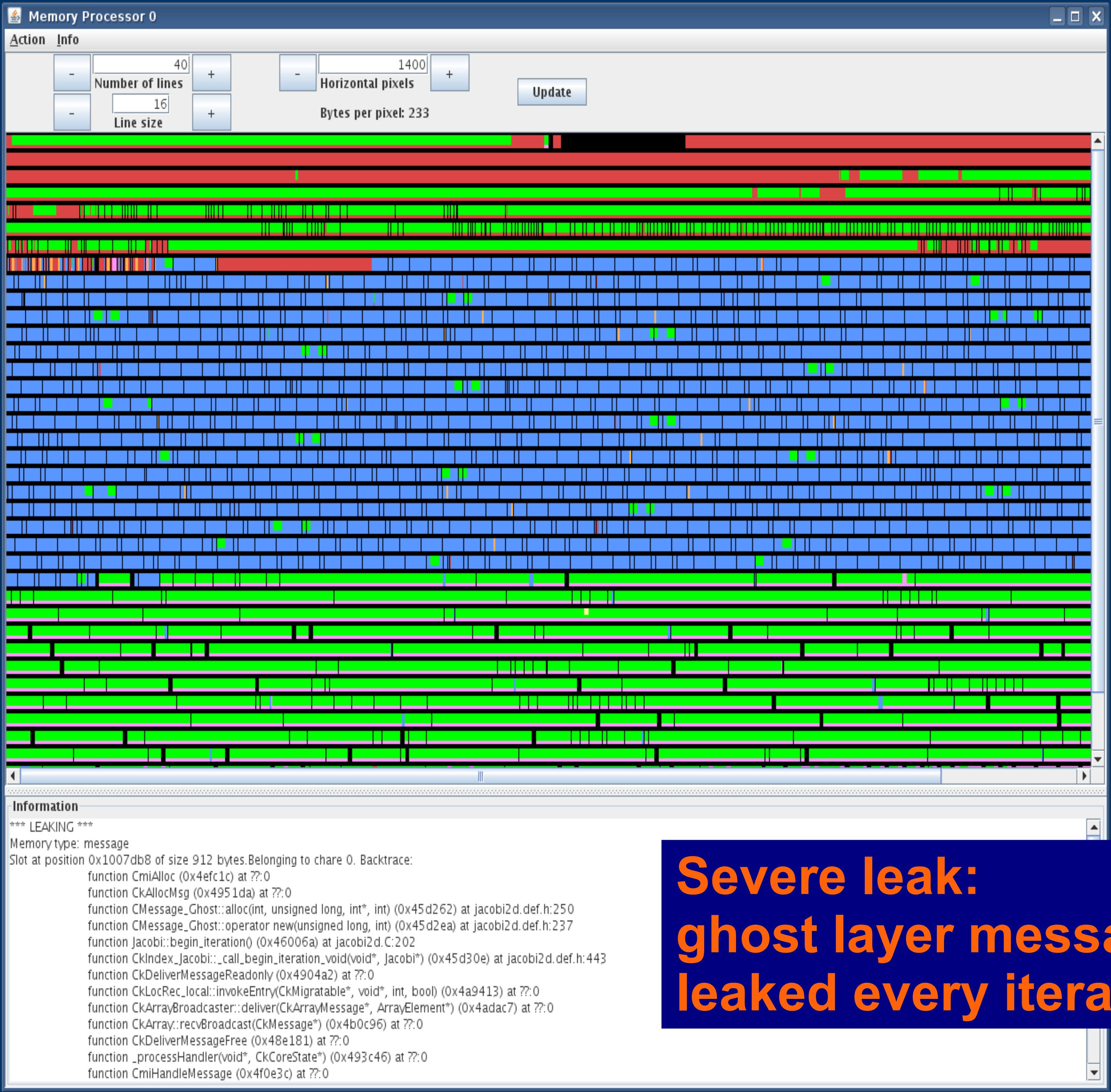


Program Parameters

Executable:	./jacobi2d	Change
Working dir:	linux-amd64/examples/charm++/jacobi2d-iter/charmdebug_tut	Change
Command Line Parameters:	100 50 ++server-port 1234	
Number of Processors:	4	
Port Number:		
SSH port number:	0	
Host name:	faith	
Username:		
Input file:		

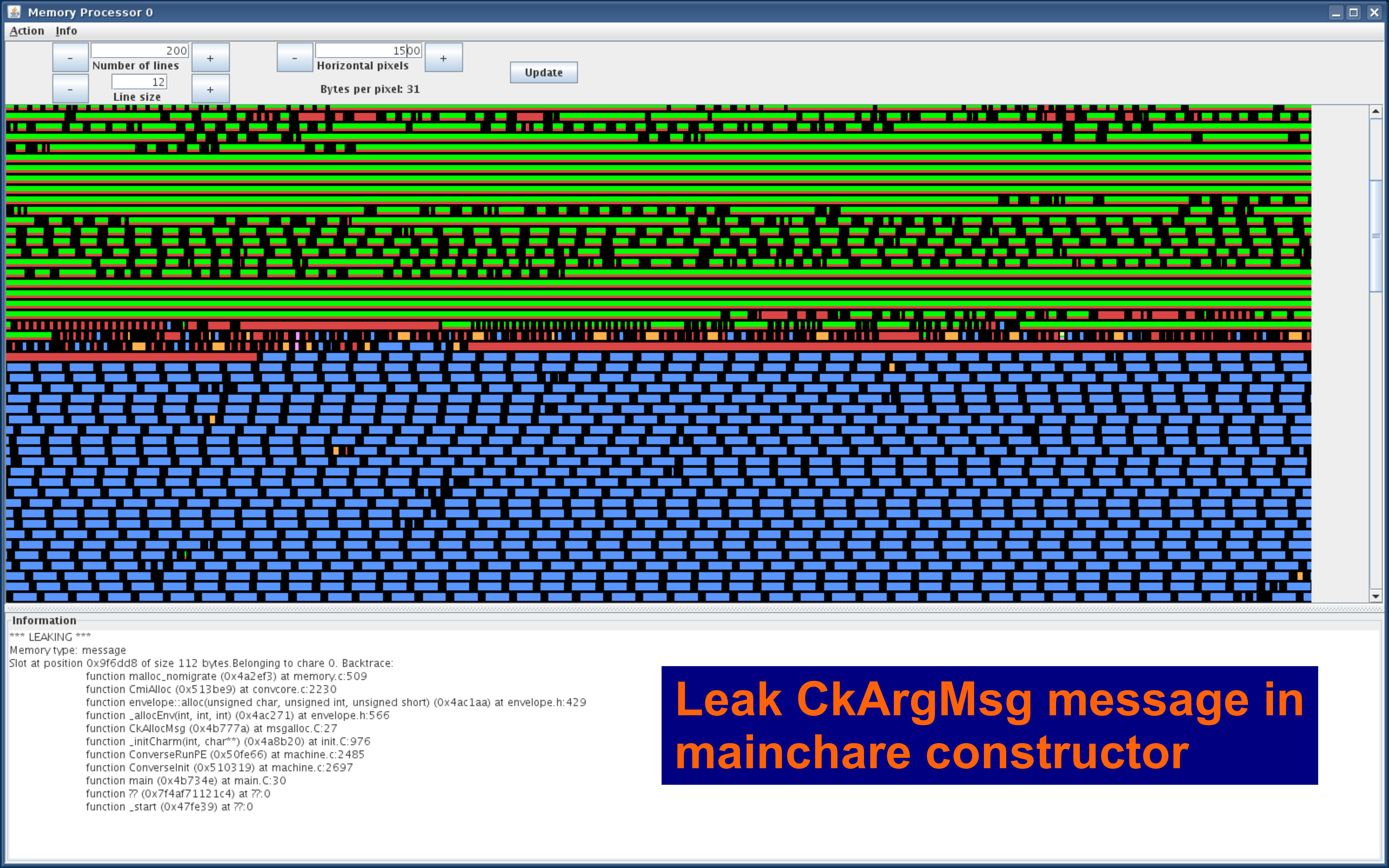
[ ] Wait for file to be created

[ ] Use ssh tunneling



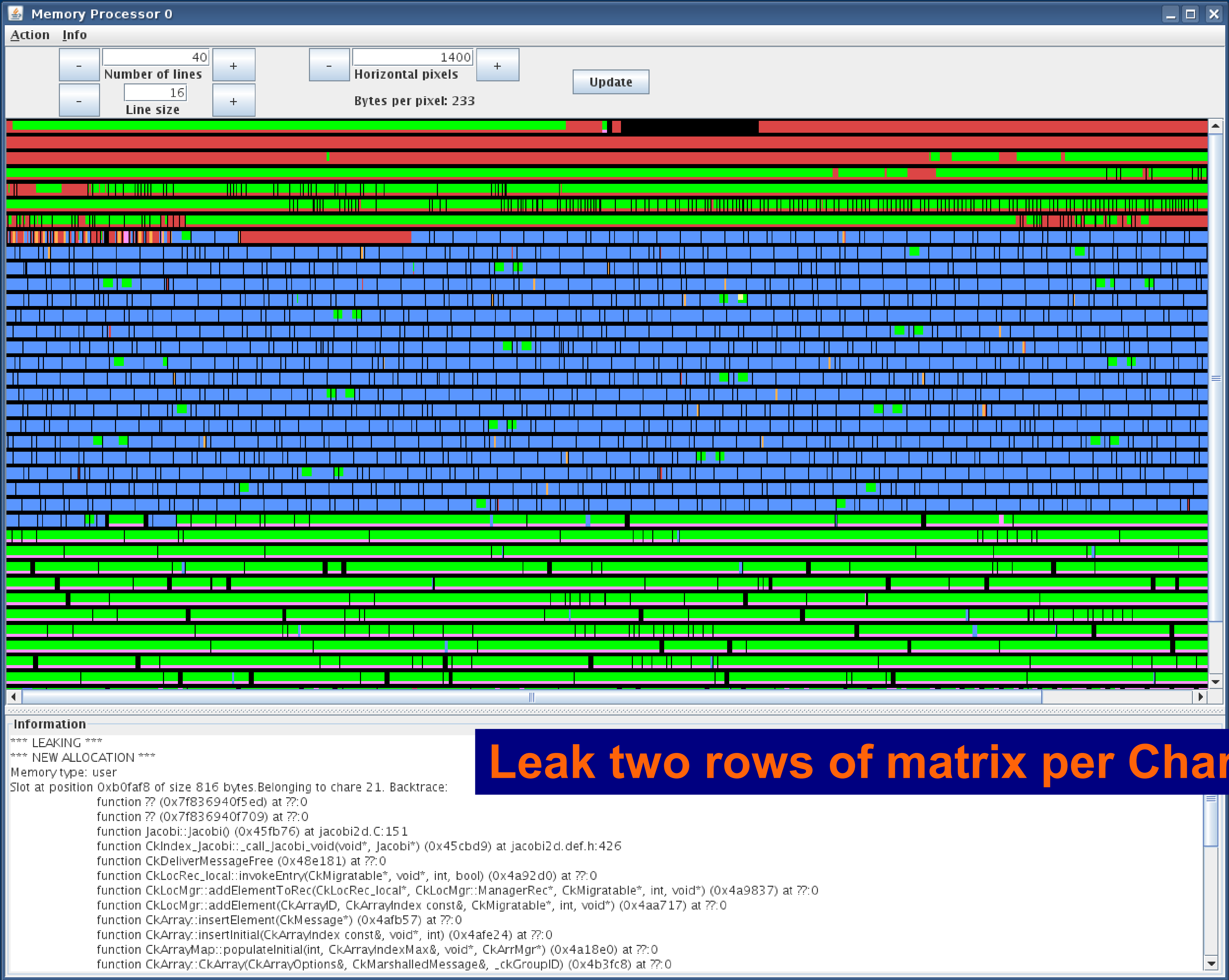
```
*** LEAKING ***
Memory type: message
Slot at position 0x1007db8 of size 912 bytes.Belonging to chare 0. Backtrace:
        function CmiAlloc (0x4efc1c) at ??:0
        function CkAllocMsg (0x4951da) at ??:0
        function CMessage_Ghost::alloc(int, unsigned long, int*, int) (0x45d262) at jacobi2d.def.h:250
        function CMessage_Ghost::operator new(unsigned long, int) (0x45d2ea) at jacobi2d.def.h:237
        function Jacobi::begin_iteration() (0x46006a) at jacobi2d.C:202
        function CkIndex_Jacobi::_call_begin_iteration_void(void*, Jacobi*) (0x45d30e) at jacobi2d.def.h:443
        function CkDeliverMessageReadonly (0x4904a2) at ??:0
        function CkLocRec_local::invokeEntry(CkMigratable*, void*, int, bool) (0x4a9413) at ??:0
        function CkArrayBroadcaster::deliver(CkArrayMessage*, ArrayElement*) (0x4adac7) at ??:0
        function CkArray::recvBroadcast(CkMessage*) (0x4b0c96) at ??:0
        function CkDeliverMessageFree (0x48e181) at ??:0
        function _processHandler(void*, CkCoreState*) (0x493c46) at ??:0
        function CmiHandleMessage (0x4f0e3c) at ??:0
```

**Severe leak:
ghost layer messages
leaked every iteration**



```
*** LEAKING ***
Memory type: message
Slot at position 0x9f6dd8 of size 112 bytes.Belonging to chare 0. Backtrace:
                function malloc_nomigrate (0x4a2ef3) at memory.c:509
                function CmiAlloc (0x513be9) at convcore.c:2230
                function envelope::alloc(unsigned char, unsigned int, unsigned short) (0x4ac1aa) at envelope.h:429
                function _allocEnv(int, int, int) (0x4ac271) at envelope.h:566
                function CkAllocMsg (0x4b777a) at msgalloc.C:27
                function _initCharm(int, char**) (0x4a8b20) at init.C:976
                function ConverseRunPE (0x50fe66) at machine.c:2485
                function ConverseInit (0x510319) at machine.c:2697
                function main (0x4b734e) at main.C:30
                function ?? (0x7f4af71121c4) at ??:0
                function _start (0x47fe39) at ??:0
```

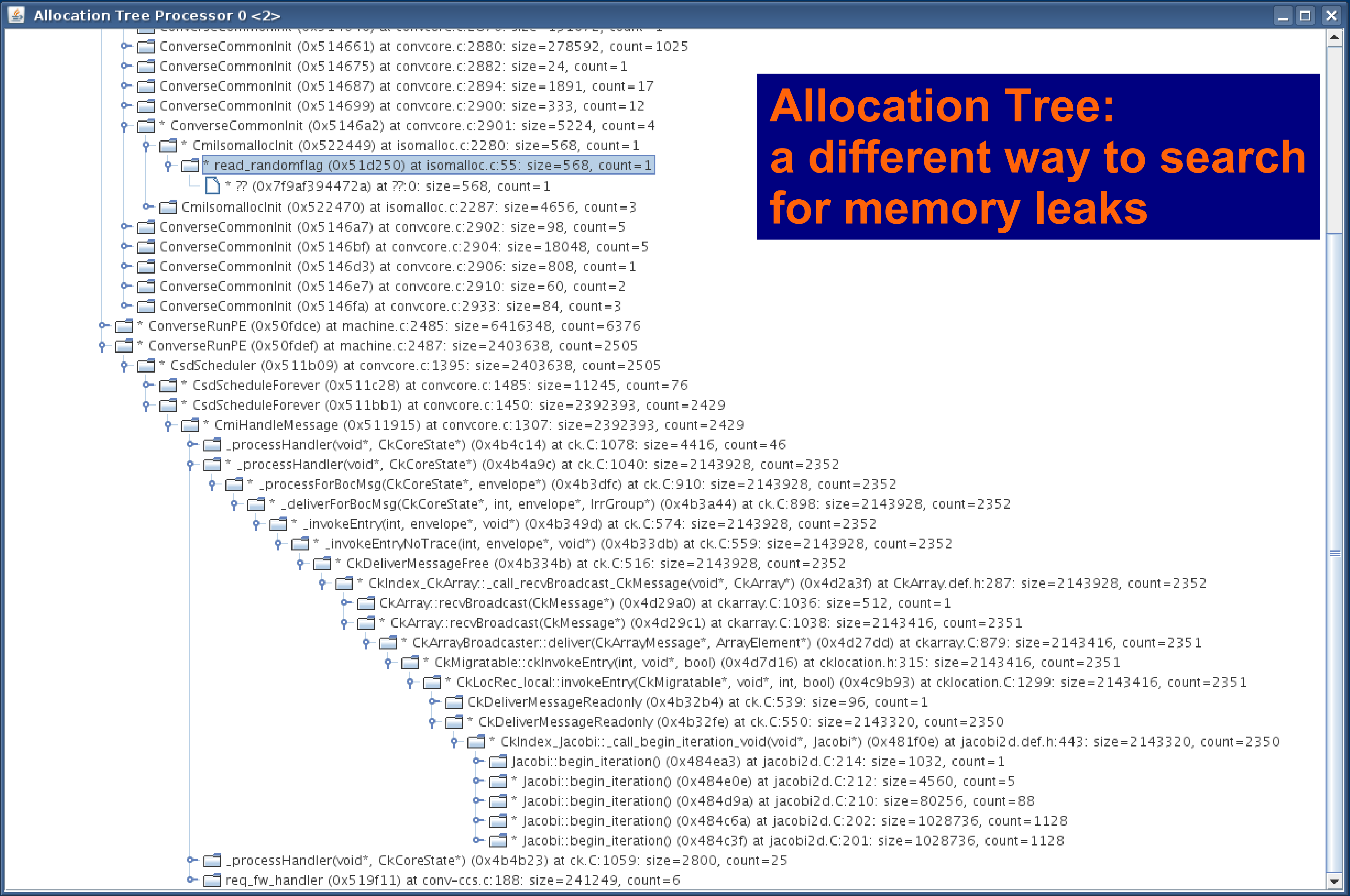
Leak CkArgMsg message in mainchare constructor



```
*** LEAKING ***
*** NEW ALLOCATION ***
Memory type: user
Slot at position 0xb0faf8 of size 816 bytes.Belonging to chare 21. Backtrace:
        function ?? (0x7f836940f5ed) at ??:0
        function ?? (0x7f836940f709) at ??:0
        function Jacobi::Jacobi() (0x45fb76) at jacobi2d.C:151
        function CkIndex_Jacobi::_call_Jacobi_void(void*, Jacobi*) (0x45cbd9) at jacobi2d.def.h:426
        function CkDeliverMessageFree (0x48e181) at ??:0
        function CkLocRec_local::invokeEntry(CkMigratable*, void*, int, bool) (0x4a92d0) at ??:0
        function CkLocMgr::addElementToRec(CkLocRec_local*, CkLocMgr::ManagerRec*, CkMigratable*, int, void*) (0x4a9837) at ??:0
        function CkLocMgr::addElement(CkArrayID, CkArrayIndex const&, CkMigratable*, int, void*) (0x4aa717) at ??:0
        function CkArray::insertElement(CkMessage*) (0x4afb57) at ??:0
        function CkArray::insertInitial(CkArrayIndex const&, void*, int) (0x4afe24) at ??:0
        function CkArrayMap::populateInitial(int, CkArrayIndexMax&, void*, CkArrMgr*) (0x4a18e0) at ??:0
        function CkArray::CkArray(CkArrayOptions&, CkMarshalledMessage&, _ckGroupID) (0x4b3fc8) at ??:0
```

Leak two rows of matrix per Chare

Allocation Tree: a different way to search for memory leaks



Allocation Tree Processor 0 <2>

```
    ConverseCommonInit (0x514661) at convcore.c:2880: size=278592, count=1025
    ConverseCommonInit (0x514675) at convcore.c:2882: size=24, count=1
    ConverseCommonInit (0x514687) at convcore.c:2894: size=1891, count=17
    ConverseCommonInit (0x514699) at convcore.c:2900: size=333, count=12
    * ConverseCommonInit (0x5146a2) at convcore.c:2901: size=5224, count=4
      * CmiIsomallocInit (0x522449) at isomalloc.c:2280: size=568, count=1
        * read_randomflag (0x51d250) at isomalloc.c:55: size=568, count=1
          * ?? (0x7f9af394472a) at ??:0: size=568, count=1
      CmiIsomallocInit (0x522470) at isomalloc.c:2287: size=4656, count=3
    ConverseCommonInit (0x5146a7) at convcore.c:2902: size=98, count=5
    ConverseCommonInit (0x5146bf) at convcore.c:2904: size=18048, count=5
    ConverseCommonInit (0x5146d3) at convcore.c:2906: size=808, count=1
    ConverseCommonInit (0x5146e7) at convcore.c:2910: size=60, count=2
    ConverseCommonInit (0x5146fa) at convcore.c:2933: size=84, count=3
  * ConverseRunPE (0x50fdce) at machine.c:2485: size=6416348, count=6376
  * ConverseRunPE (0x50fdef) at machine.c:2487: size=2403638, count=2505
    * CsdScheduler (0x511b09) at convcore.c:1395: size=2403638, count=2505
      * CsdScheduleForever (0x511c28) at convcore.c:1485: size=11245, count=76
      * CsdScheduleForever (0x511bb1) at convcore.c:1450: size=2392393, count=2429
        * CmiHandleMessage (0x511915) at convcore.c:1307: size=2392393, count=2429
          _processHandler(void*, CkCoreState*) (0x4b4c14) at ck.C:1078: size=4416, count=46
          * _processHandler(void*, CkCoreState*) (0x4b4a9c) at ck.C:1040: size=2143928, count=2352
            * _processForBocMsg(CkCoreState*, envelope*) (0x4b3dfc) at ck.C:910: size=2143928, count=2352
              * _deliverForBocMsg(CkCoreState*, int, envelope*, IrrGroup*) (0x4b3a44) at ck.C:898: size=2143928, count=2352
                * _invokeEntry(int, envelope*, void*) (0x4b349d) at ck.C:574: size=2143928, count=2352
                  * _invokeEntryNoTrace(int, envelope*, void*) (0x4b33db) at ck.C:559: size=2143928, count=2352
                    * CkDeliverMessageFree (0x4b334b) at ck.C:516: size=2143928, count=2352
                      * CkIndex_CkArray::_call_recvBroadcast_CkMessage(void*, CkArray*) (0x4d2a3f) at CkArray.def.h:287: size=2143928, count=2352
                        CkArray::recvBroadcast(CkMessage*) (0x4d29a0) at ckarray.C:1036: size=512, count=1
                        * CkArray::recvBroadcast(CkMessage*) (0x4d29c1) at ckarray.C:1038: size=2143416, count=2351
                          * CkArrayBroadcaster::deliver(CkArrayMessage*, ArrayElement*) (0x4d27dd) at ckarray.C:879: size=2143416, count=2351
                            * CkMigratable::ckInvokeEntry(int, void*, bool) (0x4d7d16) at cklocation.h:315: size=2143416, count=2351
                              * CkLocRec_local::invokeEntry(CkMigratable*, void*, int, bool) (0x4c9b93) at cklocation.C:1299: size=2143416, count=2351
                                CkDeliverMessageReadonly (0x4b32b4) at ck.C:539: size=96, count=1
                                * CkDeliverMessageReadonly (0x4b32fe) at ck.C:550: size=2143320, count=2350
                                  * CkIndex_Jacobi::_call_begin_iteration_void(void*, Jacobi*) (0x481f0e) at jacobi2d.def.h:443: size=2143320, count=2350
                                    Jacobi::begin_iteration() (0x484ea3) at jacobi2d.C:214: size=1032, count=1
                                    * Jacobi::begin_iteration() (0x484e0e) at jacobi2d.C:212: size=4560, count=5
                                    * Jacobi::begin_iteration() (0x484d9a) at jacobi2d.C:210: size=80256, count=88
                                    * Jacobi::begin_iteration() (0x484c6a) at jacobi2d.C:202: size=1028736, count=1128
                                    * Jacobi::begin_iteration() (0x484c3f) at jacobi2d.C:201: size=1028736, count=1128
          _processHandler(void*, CkCoreState*) (0x4b4b23) at ck.C:1059: size=2800, count=25
          req_fw_handler (0x519f11) at conv-ccs.c:188: size=241249, count=6
```