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

CharmDebug → GDB
Main Program View

- **entry methods**
  - System Entries
  - User Entries
  - Main
  - Hello
    - Hello(CkMigrate)
    - Hello(void)
    - SayHidnt hiNo
  - HelloGroup
  - HelloNode
  - HelloChara
  - SecondArray

- **processor subsets**
  - Output
  - Messages queued
  - Message details

- **View Entities on PE**
  - Messages in Queue
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  - Manual (outdated)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
Severe leak: ghost layer messages leaked every iteration
Leak CkArgMsg message in mainchare constructor
Leak two rows of matrix per Chare
Allocation Tree:
a different way to search for memory leaks