GASNet Trace

Wei Tu
http://titanium.cs.berkeley.edu

U.C. Berkeley
September 9, 2004

What is GASNet trace?

- **Performance tool for Titanium program**
  - Applicable to all GAS languages
- **It presents communication information of a certain run of a program**
  - summarizes data from the tracing utility of GASNet

Why do we need it?

- **Global Address Space Languages are easier to use than message passing**
- **But Performance analysis can be more subtle**
  - Unexpected implicit communication

Why is it useful?

- **understand the communication pattern of the program**
- **discover accidental communication in implicit assignment(memcpy)**
- **detect load imbalances**
- **evaluate the language runtime library**

GET / PUT reports

- **LOCAL**
  - PUT/GET to memory which is local to the process
  - Global pointer used for local access
- **GLOBAL**
  - Global pointer used for remote access

*Local pointer not in the report.

Example - SharkFish GET

```
GET REPORT:
SOURCE  LINE  TYPE       MSG:(min    max     avg     total)   CALLS
================================================================
../../../tcbuild/../../src-arity/tlib/java/io/BufferedOutputStream.java
                71     LOCAL       4 B     4 B      4 B      7.87 K    2014
...
SharksFish.ti
                24     LOCAL       8 B     8 B      8 B    156.25 K   20000
SharksFish.ti
                25     LOCAL       8 B     8 B      8 B    156.25 K   20000
...
SharksFish.ti
               273     GLOBAL      8 B     8 B      8 B    562.50 K   72000
SharksFish.ti
               273     LOCAL       4 B     8 B      6 B    1.74 M     328000
...
```

SharksFish.ti

```
4     public class Fish. {
...
23     public void
       timestep(double dt) {
24    pos.x += dt*vel.x;
25    pos.y += dt.vel.y;
...
268   /* Modify forces on local
       fish from all the fish */
269   for(j=0;
       j<myparticles.length; j++) {
...
273   double dx =
       particles[k].pos.x -
       myparticles[j].pos.x;
...
```


BARRIER reports

- **WAIT**
  - Time spent blocking at the barrier
  - Reflects load imbalance

- **WAITNOTIFY**
  - Time interval between gasnet_notify and gasnet_wait
  - Currently, Titanium only has single phase barriers

Example - SharkFish Barrier

```
BARRIER REPORT:
SOURCE  LINE  TYPE        TIME(min    max     avg     total)  CALLS
=========================================================
../../../../src-arity/tlib/ti/lang/Reduce-guts.cti
      132 NOTIFYWAIT    14.0 us   18.0 us   15.4 us  123.0 us      2
...
SharksFish.ti
      209 NOTIFYWAIT    11.0 us   12.0 us   11.8 us   47.0 us      1
SharksFish.ti
       209      WAIT    31.0 us   15.5 ms   11.6 ms   46.5 ms       1
...
SharksFish.ti
      242 NOTIFYWAIT    10.0 us    1.6 ms   26.5 us  106.0 ms   1000
SharksFish.ti
       242      WAIT    28.0 us  117.1 ms  522.1 us    2.1  s     1000
...
```

SharksFish.ti

```
...
206         total_time.start();
207
208         comm_time.start();
209         Ti.barrier();
210         comm_time.stop();
...

240             /* Wait for everyone
      to catch up. Is this barrier
      necessary? */
241             comm_time.start();
242             Ti.barrier();
243             comm_time.stop();
...
```


What can I do with it?

- **Functionalities**
  - Sort by any FIELD
  - Filter by any TYPE
- **Views**
  - Compact - one line per communication/barrier
  - Full - filename on its own line
  - Threaded - show information for each thread

Example - ArrayCopy PUT

-t -f -filter LOCAL -sort TOTAL

```
PUT REPORT:
   SOURCE  LINE  TYPE       MSG:(min    max     avg     total)   CALLS
================================================================================
   arrayCopyTest.ti
            70    GLOBAL      56 B  128.00 K   40.80 K    6.22 M    156
   Thread 0                 2.00 K  128.00 K   88.33 K    1.55 M     18
   Thread 1                   56 B  128.00 K   34.60 K    1.55 M     46
   Thread 2                   56 B  128.00 K   34.60 K    1.55 M     46
   Thread 3                   56 B  128.00 K   34.60 K    1.55 M     46
...
   arrayCopyTest.ti
            64    GLOBAL      56 B     56 B     56 B    1008 B     18
   Thread 1                   56 B     56 B     56 B     336 B      6
   Thread 2                   56 B     56 B     56 B     336 B      6
   Thread 3                   56 B     56 B     56 B     336 B      6
```

arrayCopyTest.ti

```
...
60    long [1d] single [1d] allSrc =
        new long [0 : Ti.numProcs()-1]
        [1d];
61      long [1d] single [1d] allDest =
        new long [0 : Ti.numProcs()-1]
        [1d];
62
63    allSrc.exchange(sharedSrc);
64    allDest.exchange(sharedDest);
...
69    // remote -> local
70    prvDest.copy(allSrc[left]);
71    verifyArray(prvDest, left,
        "remote -> local");
72    Ti.barrier();
...
```


Example - ArrayCopy Barrier

-t -f -filter NOTIFYWAIT -sort TOTAL

```
BARRIER REPORT:
      SOURCE    LINE  TYPE       TIME:(min    max     avg     total)  CALLS
=============================================================================
   arrayCopyTest.ti
          70       WAIT        29.0 us  332.9 ms   6.3 ms   5.9  s    233
   Thread 0..0                          29.0 us  332.9 ms   6.2 ms   1.5  s    233
   Thread 1..1                          39.0 us  327.4 ms   5.8 ms   1.4  s    233
   Thread 2..2                          48.0 us  319.6 ms   6.5 ms   1.5  s    233
   Thread 3..3                          45.0 us  327.1 ms   6.7 ms   1.6  s    233
...
   arrayCopyTest.ti
            82       WAIT              51.0 us   1.3 ms  181.1 us   3.6 ms      5
   Thread 0..0                          56.0 us   1.3 ms  302.6 us   1.5 ms      5
   Thread 1..1                          57.0 us   1.3 ms  309.2 us   1.5 ms      5
   Thread 2..2                          51.0 us  58.0 us   55.8 us  279.0 us      5
   Thread 3..3                          56.0 us  59.0 us   56.6 us  283.0 us      5
...
```

```
arrayCopyTest.ti
...
69    // remote -> local
70    prvDest.copy(allSrc[left]);
71    verifyArray(prvDest, left, "remote
      -> local");
72    Ti.barrier();
..
..
80    // remote -> remote (same owner)
81    allDest[right].copy(allSrc[right]);
82    Ti.barrier();
83    verifyArray(sharedDest,
      Ti.thisProc(), "remote -> remote
      (same owner)");
84    Ti.barrier();
```


What to do next?

- **Increase Speed**
- **Track memory allocation & usage**
- **Track lock/unlock operations**
- **Track collective operations**
- **Track active messages**
- **Separate barriers by thread (instead of node)**
- **Data analysis**
  - Distribution of resources used in put/get reports
  - Auto detect load imbalance in barrier reports
- **Hide Internals**