WESTERN WASHINGTON UNIVERSITY

Active Minds Changing Lives

chplvis

A Communication and Task Visualization Tool for Chapel

Philip Nelson, Western Washington University
Greg Titus, Cray Inc.

Tasks:

```
coforall tid in 0..#numTasks do
    writeln (“Hello from “ + tid);
```

Locales:

```
coforall loc in Locales do
    on loc do writeln (“Hello from “ + here.id);
```

Issues for debugging tasks and locales:

- tasks can be hard to visualize
- data distributions implicitly create tasks and run on locales
- may be difficult to verify computation on all locales
- no direct methods to verify

Issues for debugging tasks and locales:

- tasks can be hard to visualize
- data distributions implicitly create tasks and run on locales
- may be difficult to verify computation on all locales
- no direct methods to verify

Solution: chplvis

- Use “VisualDebug” standard module
- startVdebug(name) - starts data collection in file
- stopVdebug() - terminates data collection
- tagVdebug(tagName) – tag, point of interest
- pauseVdebug() / resumeVdebug()

Runtime Additions

Data Collection for task and communication events

Chapel Tasking Runtime has “call-back hooks”
Task Events recorded:
- Task creation
- Task starting execution
- Task termination

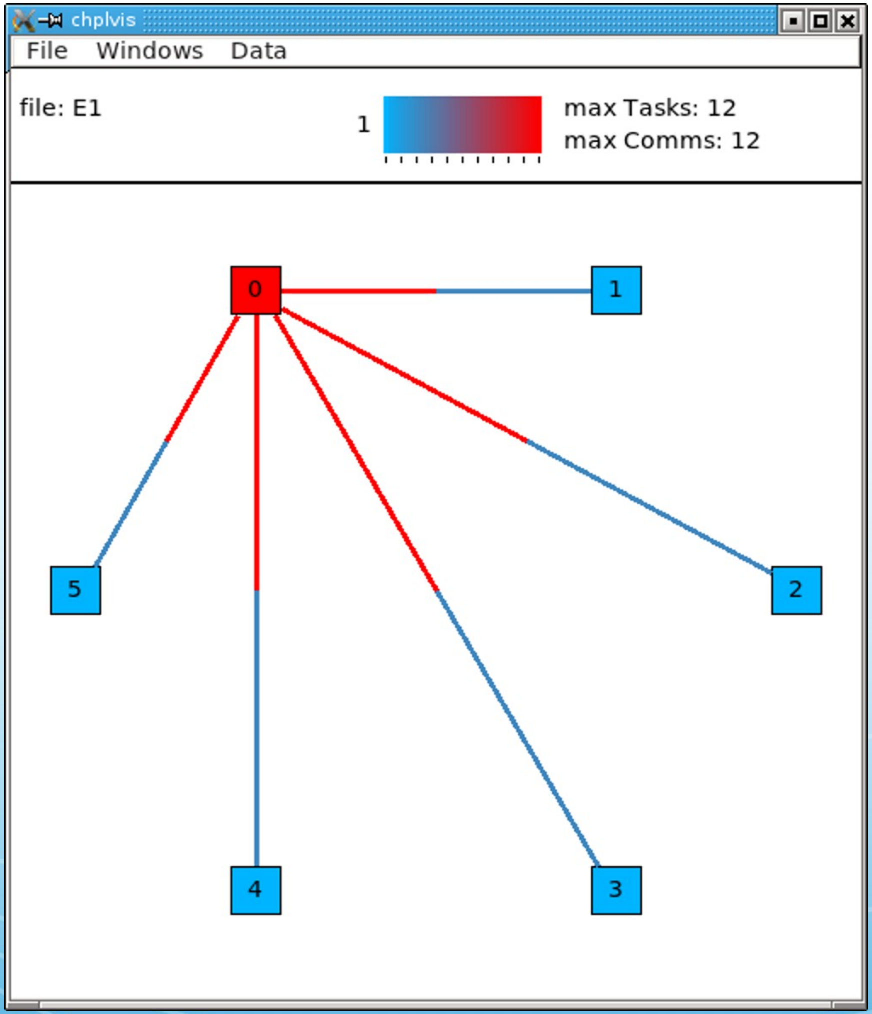
Chapel Communication Runtime for multi-locale execution
Communication Events recorded:
- Put
- Get
- Fork (put with remote task start)

Example 1 – The basics

(All example results from Chapel 1.12.0)

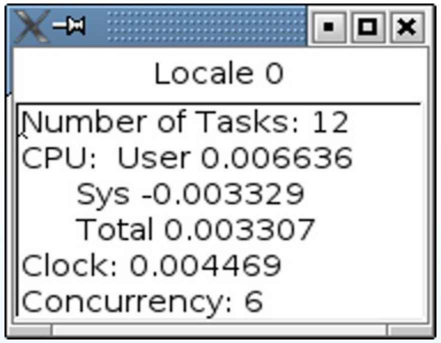
```
startVdebug (“E1”);
coforall loc in Locales do
    on loc do writeln (“Hello from locale “ + here.id + “.” );
stopVdebug();
```

Example 1 chplvis display



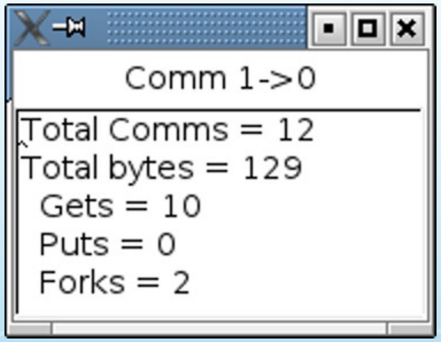
file: E1

max Tasks: 12
max Comms: 12



Locale 0

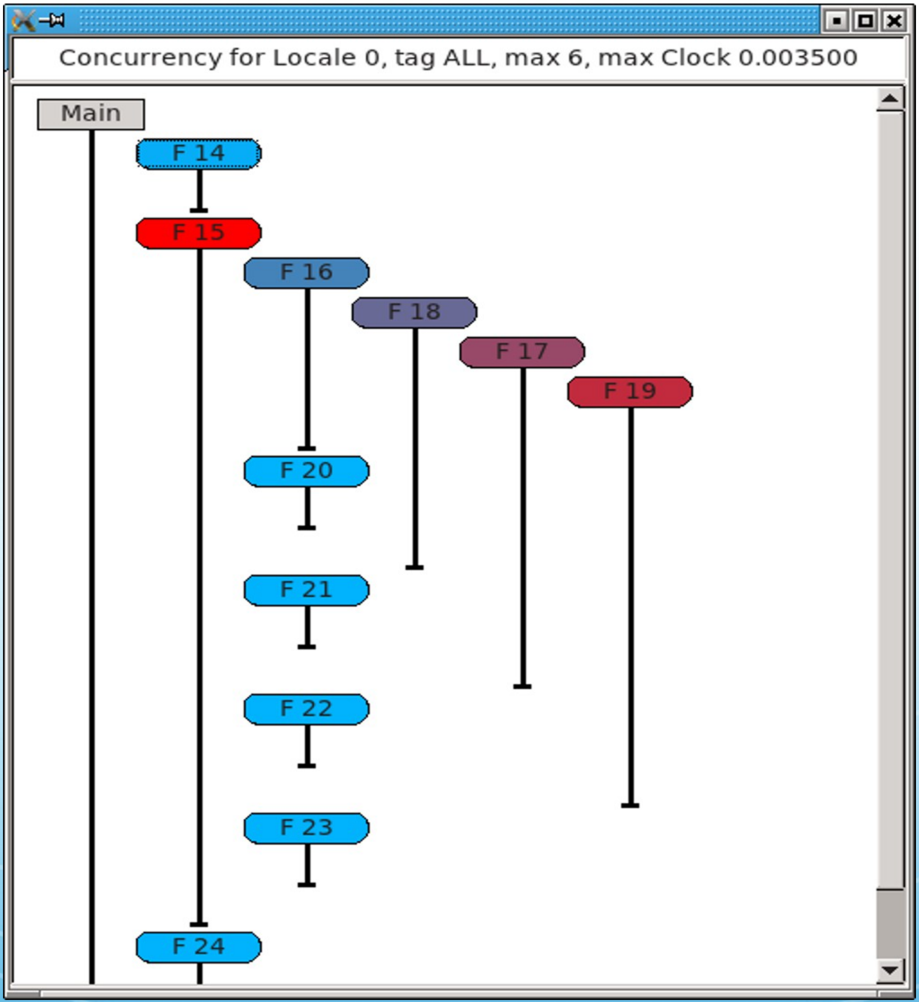
```
Number of Tasks: 12
CPU:  User 0.006636
      Sys -0.003329
      Total 0.003307
Clock: 0.004469
Concurrency: 6
```



Comm 1->0

```
Total Comms = 12
Total bytes = 129
  Gets = 10
  Puts = 0
  Forks = 2
```

Example 1 chplvis display


Concurrency for Locale 0, tag ALL, max 6, max Clock 0.003500

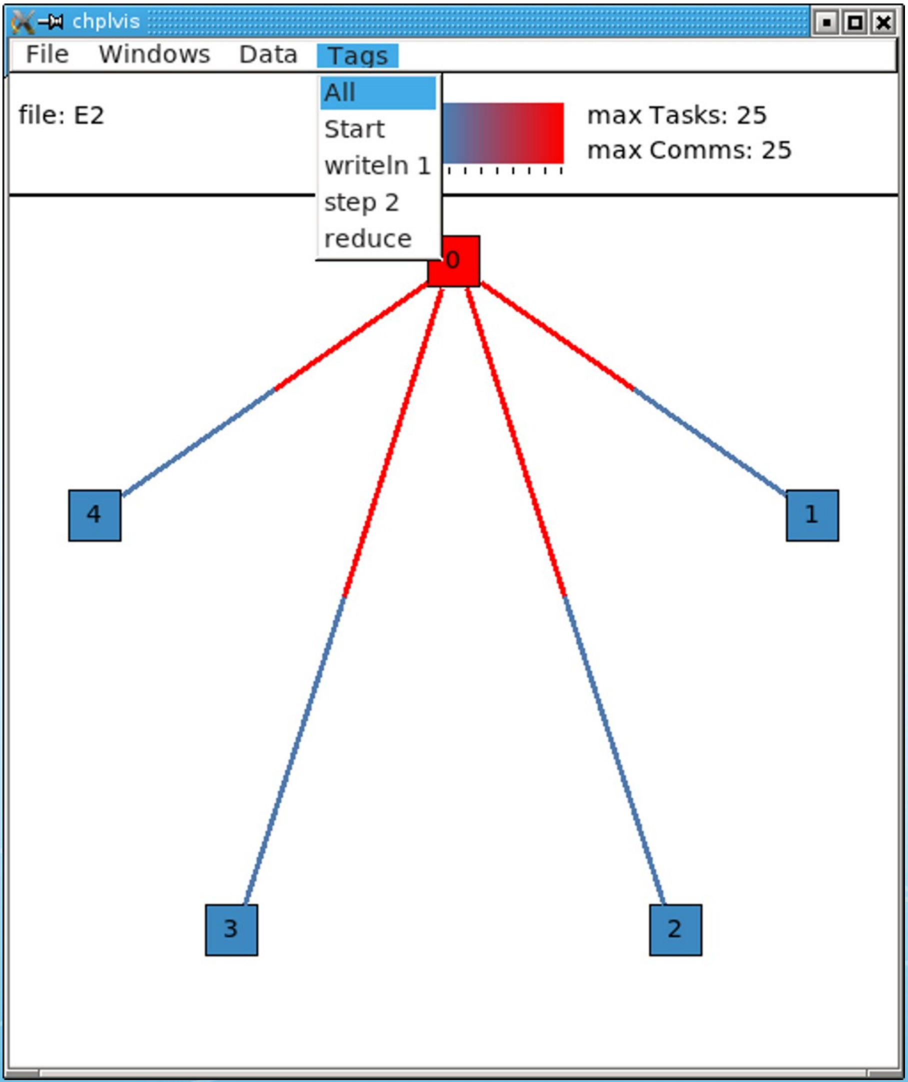
Example 2 – Tags

```
const Domain = { 1 .. ncells };
const mapDomain = Domain dmapped Block(Domain);
var    data : [mapDomain] int = 1;

startDebug (“E2”);

forall i in Domain do  data[i] += here.id + 1;

tagVdebug (“writeln 1”);
writeln (“data = “, data);

tagVdebug (“step 2”);
forall i in mapDomain do  data[i] += here.id+1;

pauseVdebug();
writeln (“data2 =”, data);
```

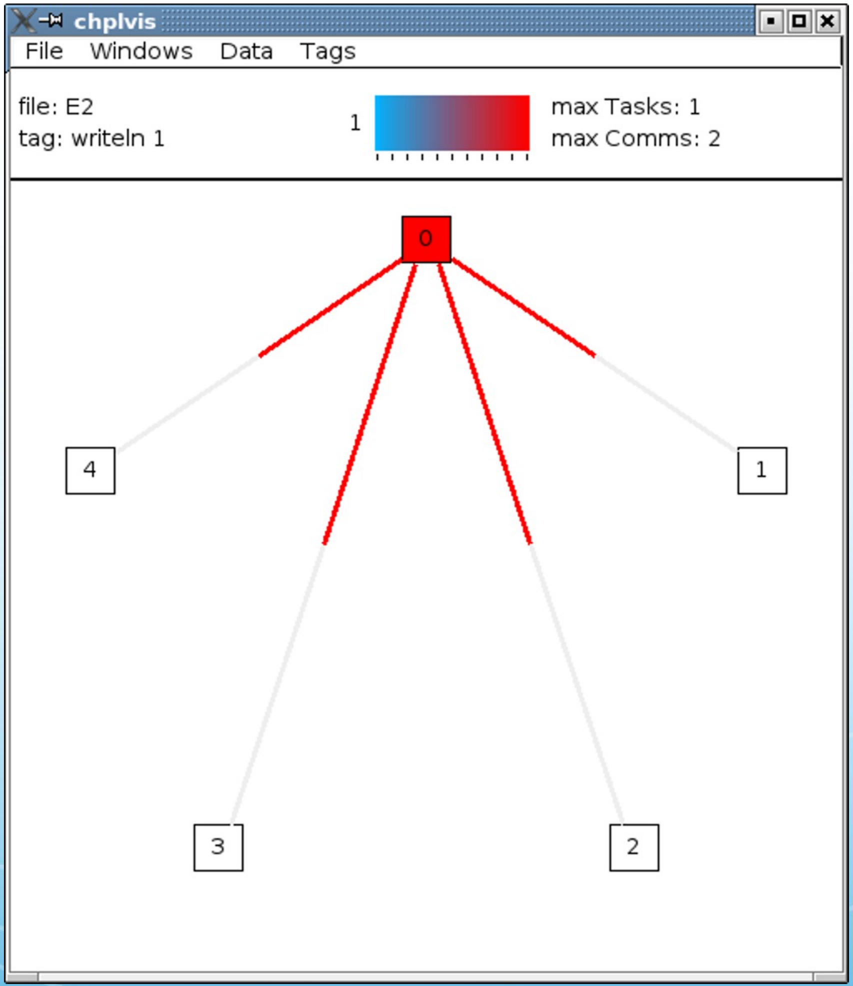
Active Minds Changing Lives

Example 2 – Tags (page 2)

```
tagVdebug("reduce");
var i = + reduce data;

StopVdebug();

Writeln ("sum is " + I + ".");
```



Example 2



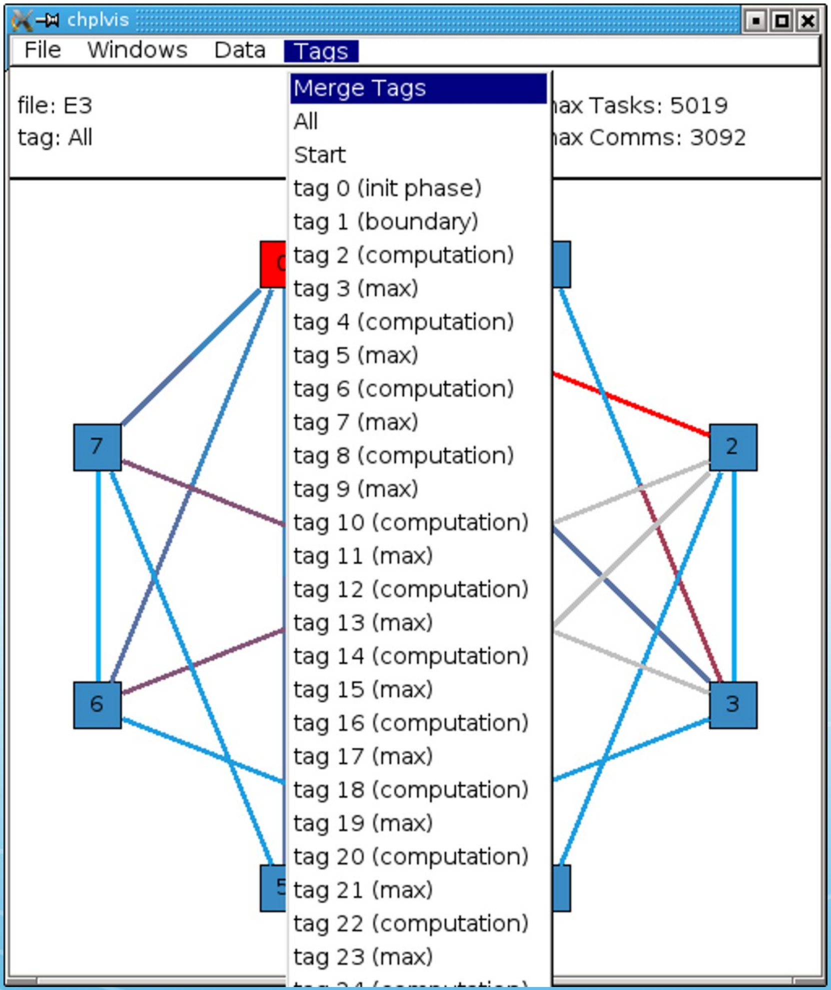
file: E2
tag: writeln 1

max Tasks: 1
max Comms: 2

Example 3 – tags in loops

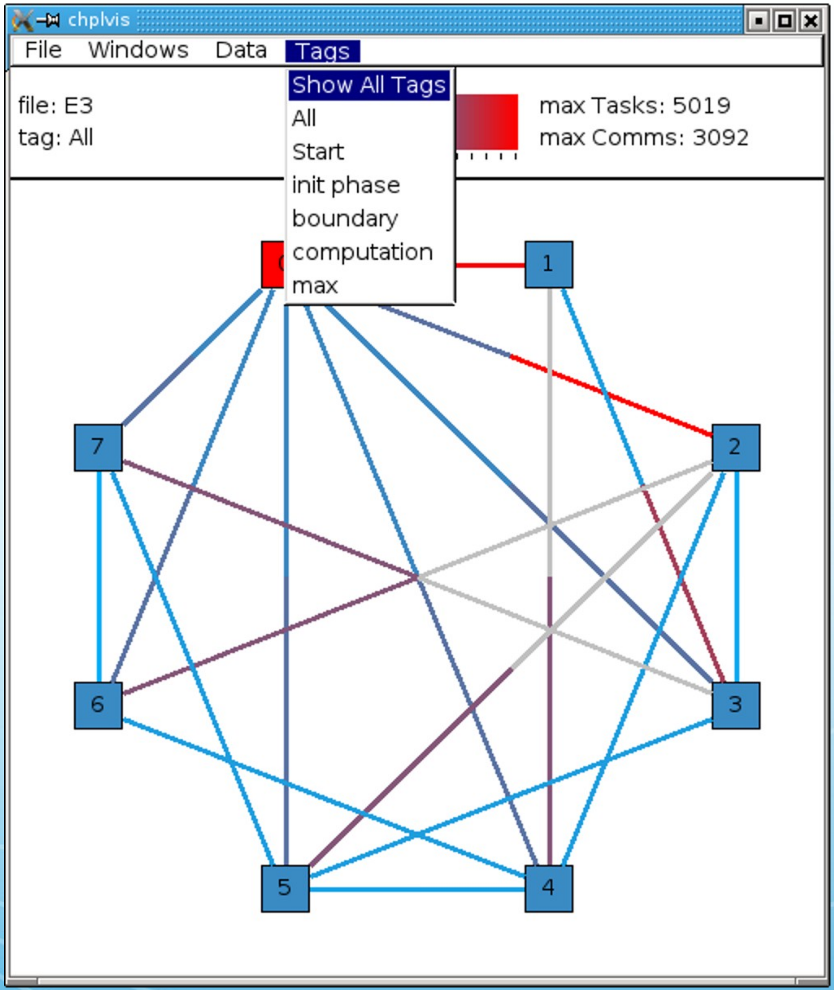
```
while (delta > epsilon) {

  tagVdebug("computation");
  for t in 1 .. compLoop do {
    forall (i,j) in R do
      A(i,j) = Temp(i,j);
    forall (i,j) in R do
      Temp(i,j) = (A(i-1,j) + A(i+1,j) + A(i,j-1) + A(i,j+1)) / 4.0;
  }

  tagVdebug("max");
  forall (i,j) in R {
    Diff(i,j) = abs(Temp(i,j)-A(i,j));
  }
  delta = max reduce Diff;
}
```

Example 3

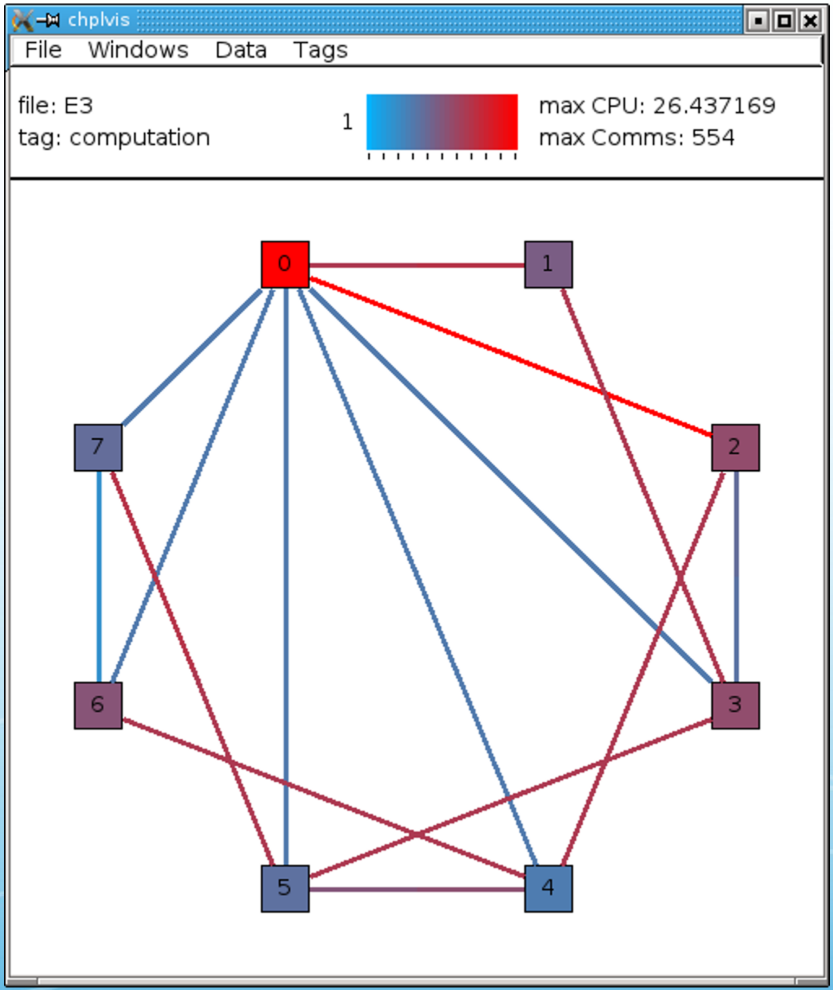

chplvis
File
Windows
Data
Tags
file: E3
tag: All
ax Tasks: 5019
ax Comms: 3092
Merge Tags
All
Start
tag 0 (init phase)
tag 1 (boundary)
tag 2 (computation)
tag 3 (max)
tag 4 (computation)
tag 5 (max)
tag 6 (computation)
tag 7 (max)
tag 8 (computation)
tag 9 (max)
tag 10 (computation)
tag 11 (max)
tag 12 (computation)
tag 13 (max)
tag 14 (computation)
tag 15 (max)
tag 16 (computation)
tag 17 (max)
tag 18 (computation)
tag 19 (max)
tag 20 (computation)
tag 21 (max)
tag 22 (computation)
tag 23 (max)
7
2
6
3
5

Active Minds Changing Lives

Example 3



file: E3
tag: All
max Tasks: 5019
max Comms: 3092



file: E3
tag: computation
max CPU: 26.437169
max Comms: 554

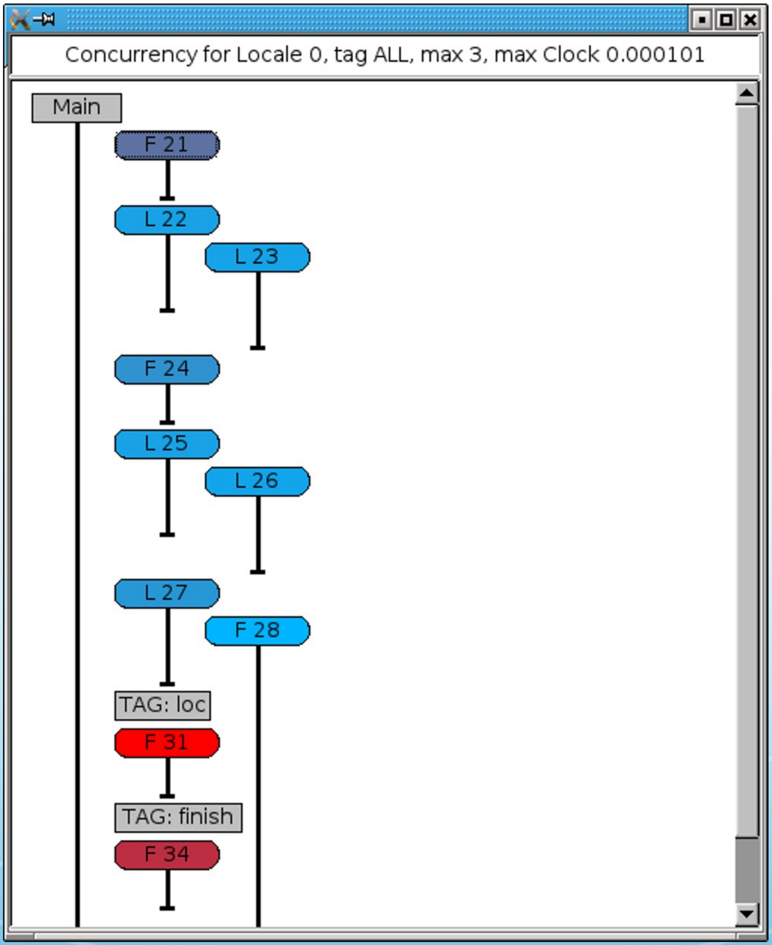
Example 4 – asynchronous tasks

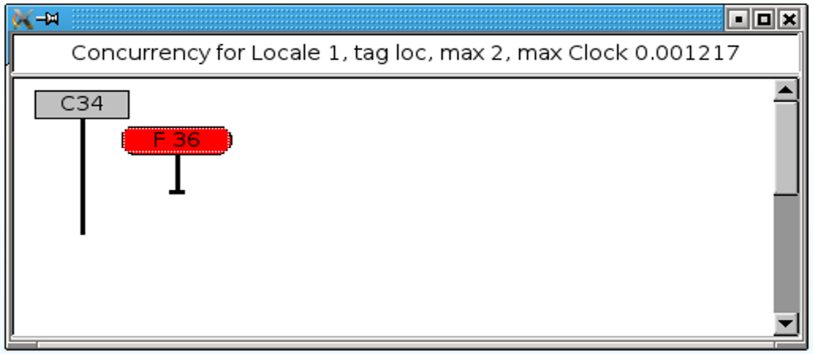
```
const space =  { 0 .. #numLocales };
const Dspace = space dmapped Block (boundingBox=space);

startVdebug("E4");

var go$: [Dspace] single bool;
var done$: [Dspace] single bool;

// Start a begin task on all locales.  The task will start and then block.
coforall loc in Locales do
  on loc do begin { // start a async task

              go$[here.id]; // Block until ready!
              writeln ("Finishing running the 'begin' statement on locale "
                        + here.id + ".");
              done$[here.id] = true;
           }
```

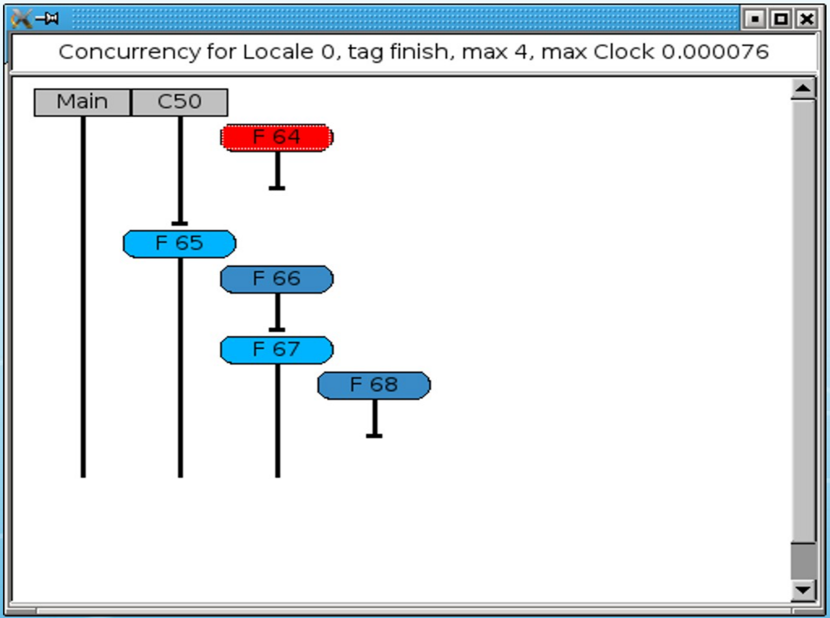
Example 4 – asynchronous tasks

```
tagVdebug("loc");

coforall loc in Locales do
    on loc do writeln("Hello from " + here.id);

tagVdebug("finish");

// Let all tasks go
go$ = true;

// Wait until all tasks are finished
done$;

stopVdebug();
```

Example 4 – asynchronous tasks


Concurrency for Locale 0, tag ALL, max 3, max Clock 0.000101


Concurrency for Locale 1, tag loc, max 2, max Clock 0.001217


Concurrency for Locale 0, tag finish, max 4, max Clock 0.000076

Experience with chplvis

Released with Chapel 1.12.0 on October 1, 2015

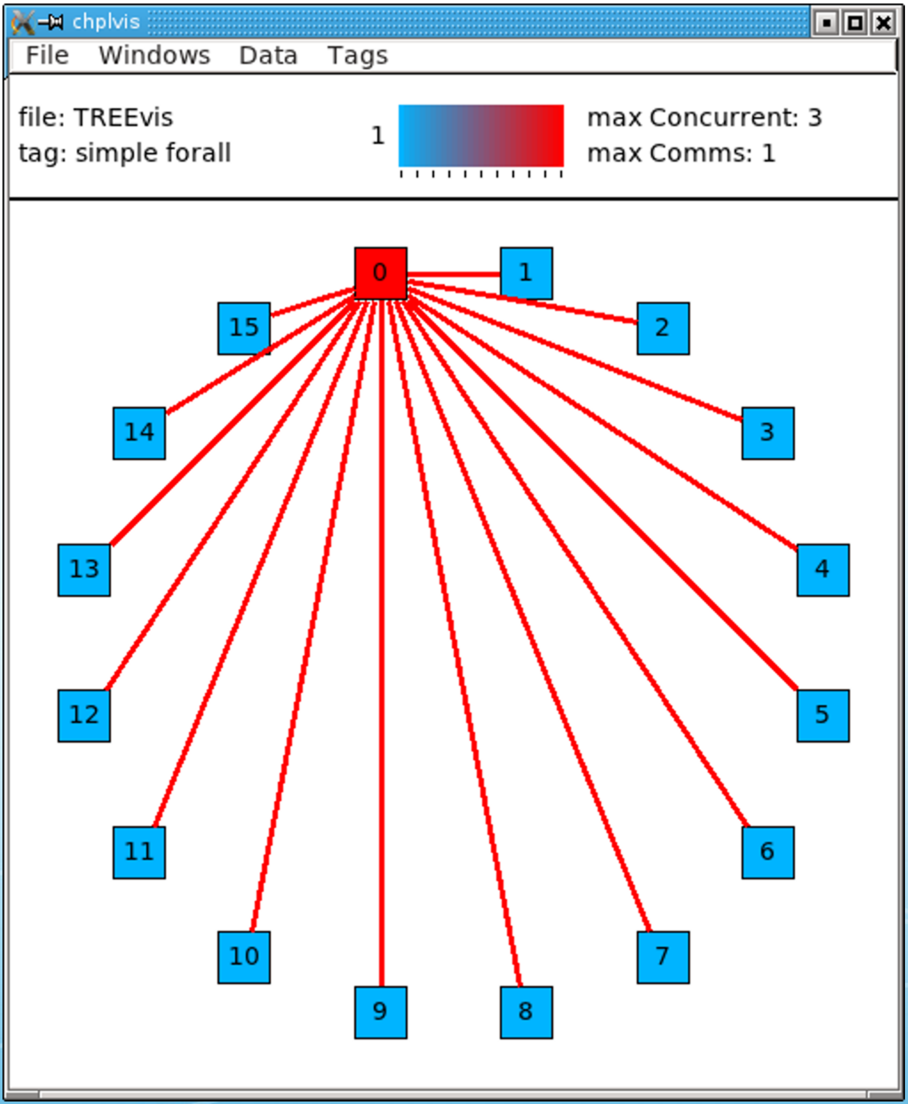
LLNL user found chplvis “an extremely useful tool”

During development, chplvis exposed sequential issues in parallel iterators.

Parallel Iterator Example

Sequential forall:

```
Const space = {0..#numLocales};
const  Bspace = space dmapped
      Block(boundingBox=space);

Var b: [Bspace] int;

tagVdebug(“simple forall”);
forall I in Bspace do b[i] = here.id;
```

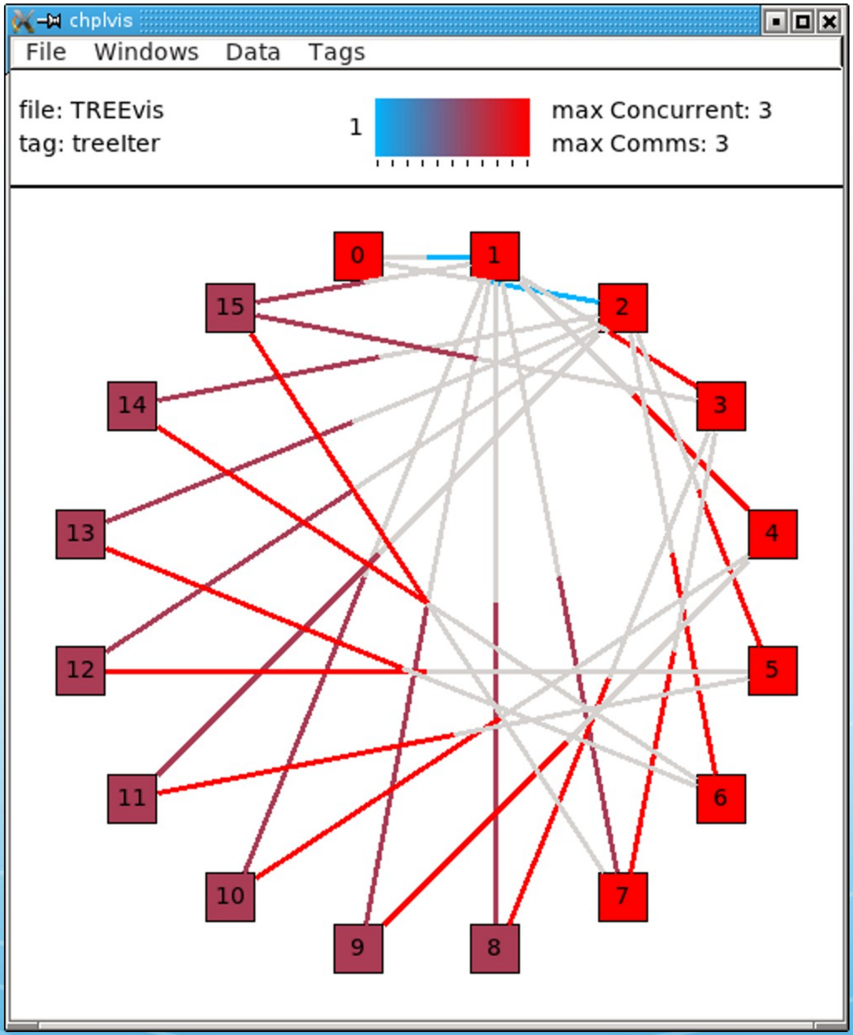


file: TREEvis
tag: simple forall
max Concurrent: 3
max Comms: 1

Parallel Iterator Example (Parallel iterator, tree pattern)

```
iter treeiter ( n: int, id: int=0) : int {
    If (id < n) {
        yield id;
        for z in treeiter (n, 2*id + 1) do yield z;
        for z in treeiter (n, 2*id + 2) do yield z;
    }
}

iter  treeiter ( param tag: iterKind, n: int, id: int=0) : int
       Where tag == iterKind.standalone {
    If (id < n) {
          yield id:
          cobegin {
               if 2*id +1 < n then on Locales[2*id+1] do
                   for z in treeiter(n=n, id=2*id+1, tag=iterKind.standalone) do yield z;
               if 2*id +2 < n then on Locales[2*id+2] do
                   for z in treeiter(n=n, id=2*id+2, tag=iterKind.standalone) do yield z;
          }
     }
}
```

Parallel Iterator Example (Parallel iterator, tree pattern)



file: TREEvis
tag: treeIter

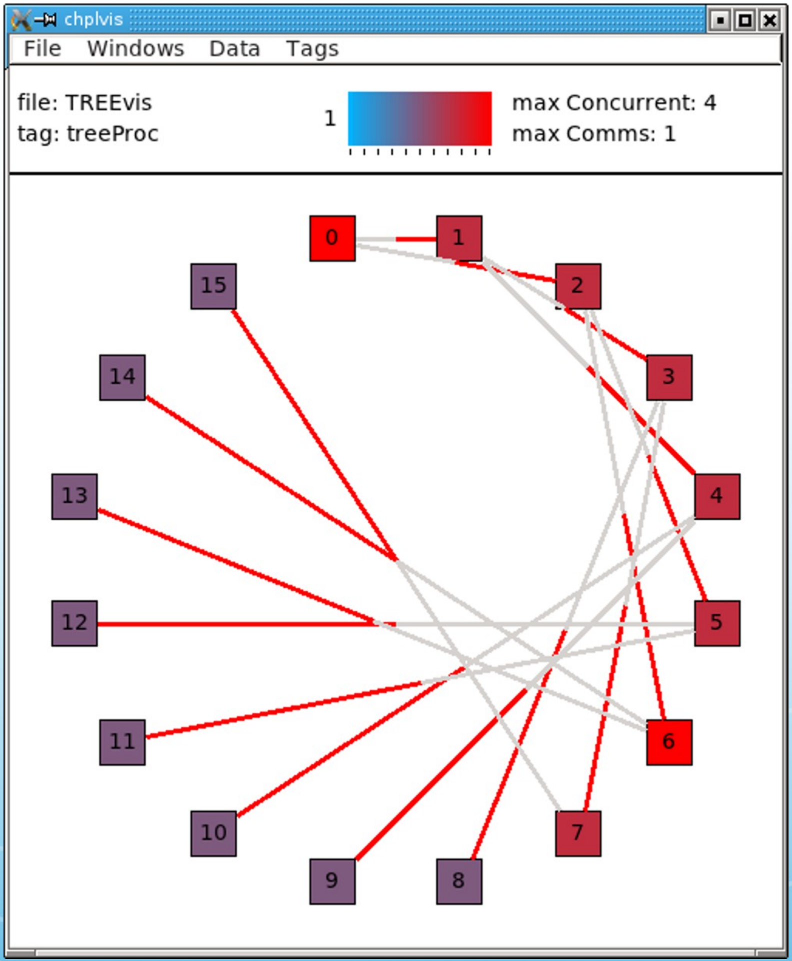
max Concurrent: 3
max Comms: 3

Active Minds Changing Lives

Parallel Iterator Example (Parallel procedure, tree pattern)

```
proc SetAPar ( id: int = 0 ) {
   Var child = id * 2 + 1;

  cobegin {
     If child < numLocales then
        On Locales[child] do SetAPar(child);
     If child+1 < numLocales then
        On Locales[child+1] do SetAPar(child+1);
   }
   a[id] = here.id;
}
```

Parallel Iterator Example (Parallel procedure, tree pattern)



file: TREEvis
tag: treeProc

max Concurrent: 4
max Comms: 1

Future work

Improve data collection from the runtime system

Changes to view large numbers of locales

Add ways to “drill down” to find user code that generates tasks or communication

Users request new features?