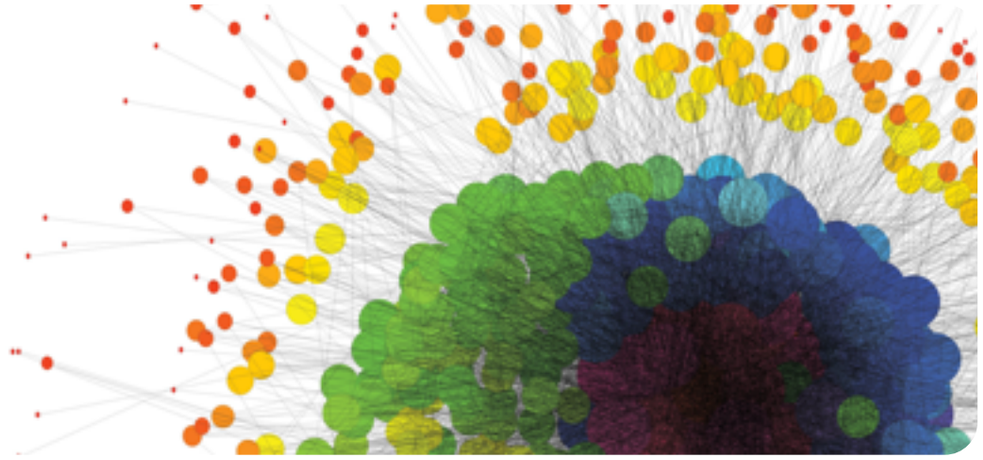
Exceptional service in the national interest

Sandia National Laboratories



CRAY CHAPEL

Qthreads

Eureka! Task Teams!

Kyle Wheeler

SC'12 Chapel Lightning Talk

U.S. DEPARTMENT OF ENERGY

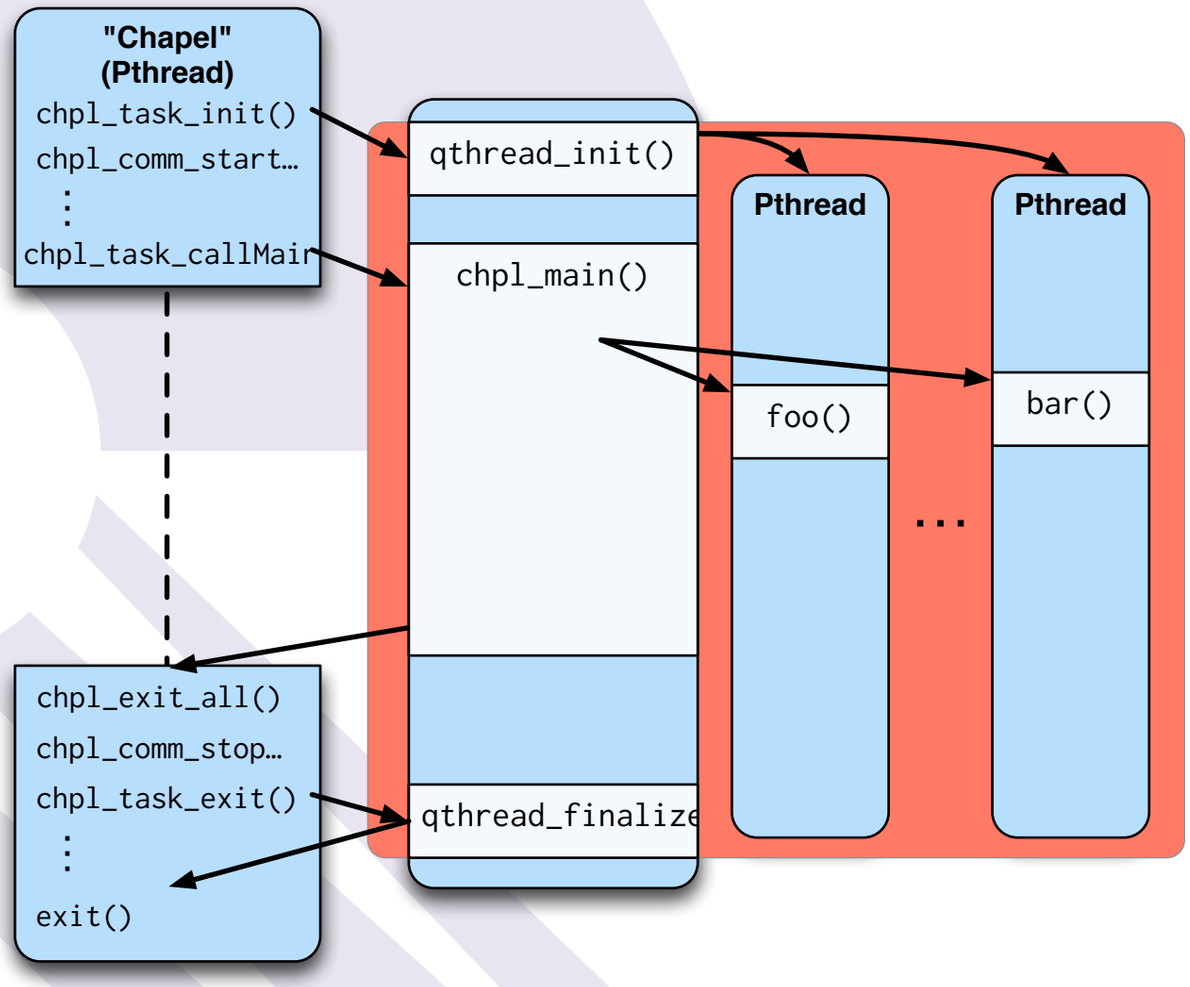
National Nuclear Security Administration

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

SAND: 2012-9463P

Chapel over Qthreads Summary

- What's Qthreads?
  - Locality-aware lightweight tasking library
  - Highly portable
    - IA32/64, AMD64, PPC32/64, SparcV9+, SST, Tilera, ARM
    - Linux, BSD, Solaris, MacOS X, Cygwin
  - Fine-grained synchronization: FEBs built-in
  - Advanced Locality-aware Scheduler (Sherwood)
- Chapel on Qthreads?
  - Thin translation layer
  - Qthreads environment "bolted on" the side
  - Separate from GASNet, so that they work together
  - **`CHPL_TASKS=qthreads`** when building



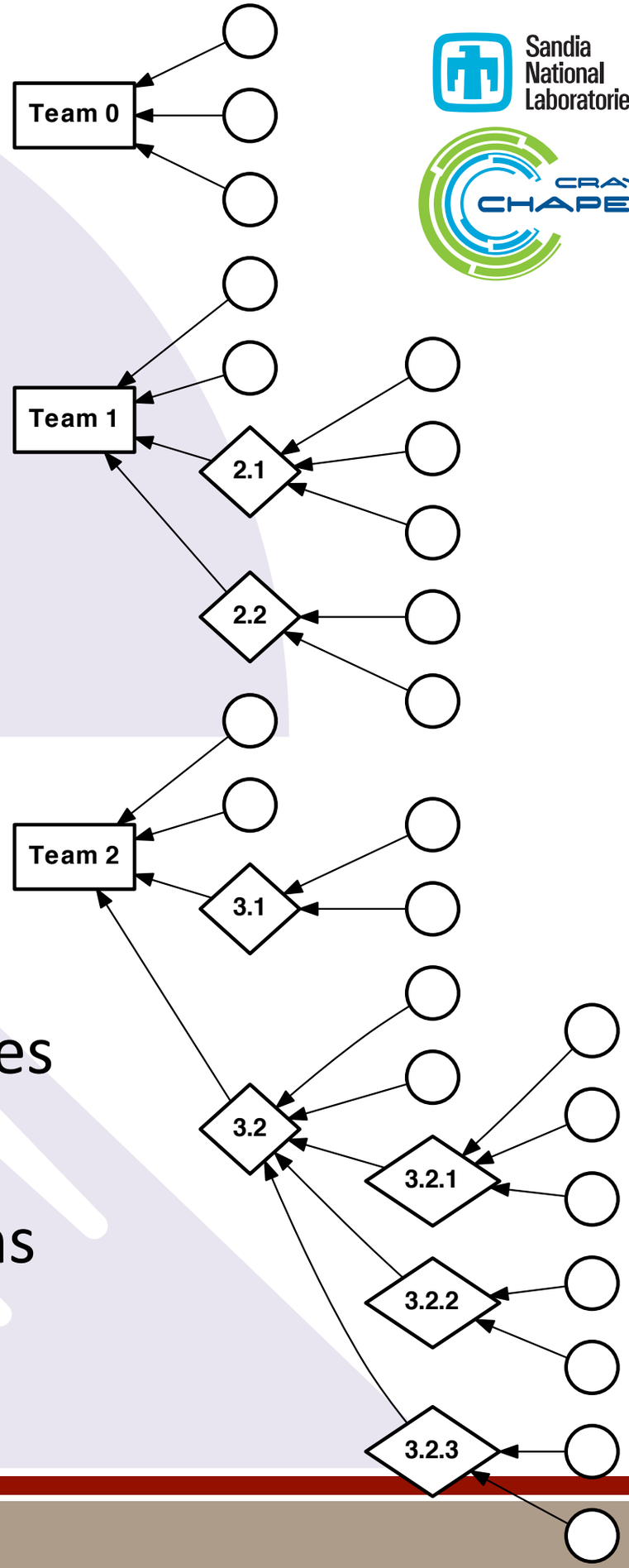
Progress in 2012…

- ✓ Better I/O handling
- ✓ System call interception & external synchronization handling
- ✓ Better operation under oversubscription
- ~ Better mapping from Chapel sync to Qthreads' FEBs

- This talk:
  - Task Teams
  - Eurekas and other collectives
  - Integrated inter-locale communication layer

Task Teams Concept

- All tasks belong to a team
  - Team “0” always runs “main”
- A task only belongs to one team
- New tasks can be spawned into:
  - Same team as parent
  - A new team
  - A new team dependent upon the parent team’s existence (subteam)
- An execution comprises a forest of team trees
  - Dynamically growing and contracting
- Tree structure encodes dependence relations
  - Recursive cascading kill of subteam tasks



Constructs that **could** use teams

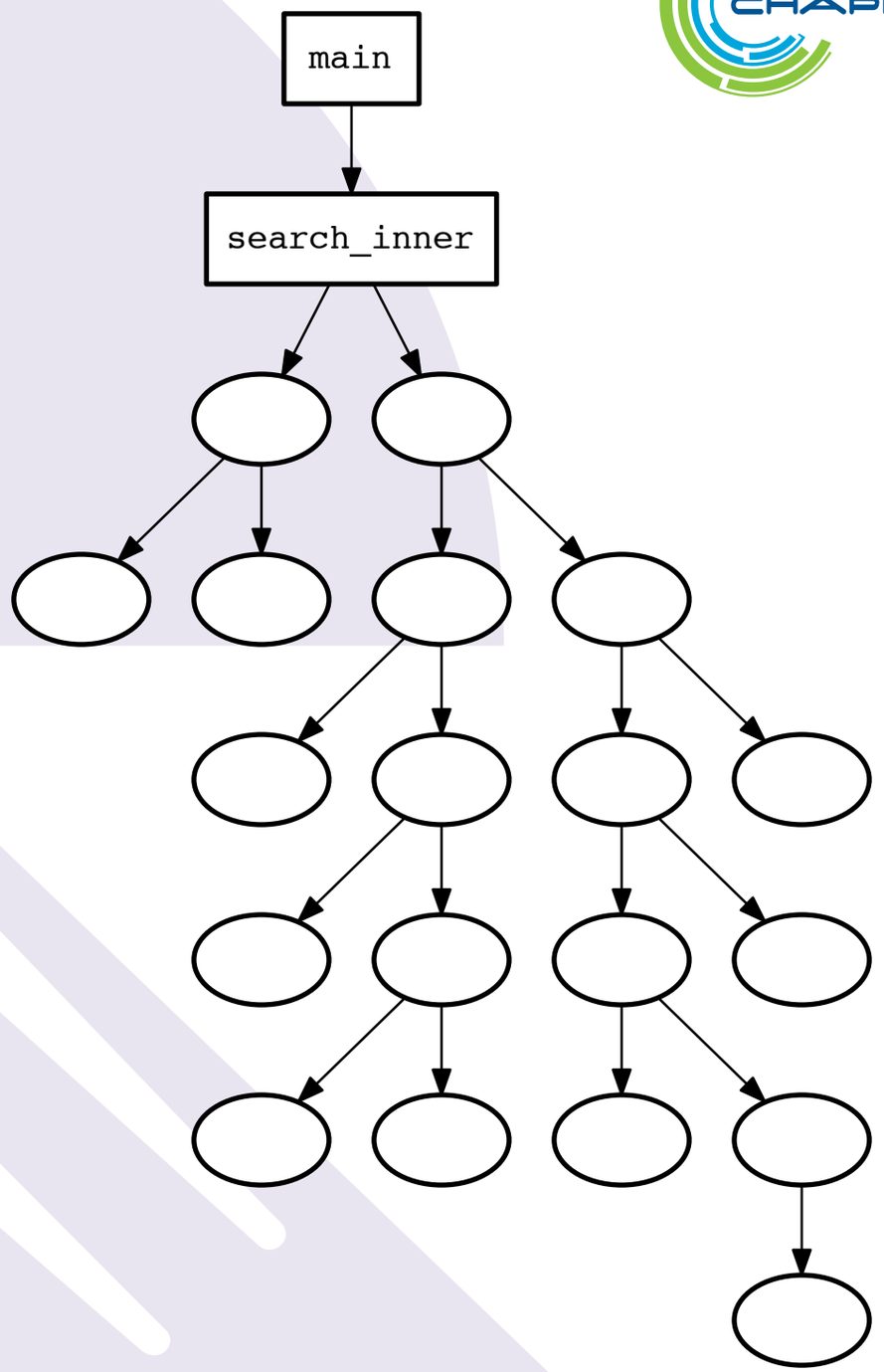
- **Existing:**
  - Termination Detection
    - `forall`
    - `coforall`
    - `cobegin`
    - `sync`
- **New:**
  - Team-oriented Reductions
    - `submit();`
  - Team-oriented Broadcasts
    - `set();`
  - Team-oriented Barriers
    - `cobegin {`
      `func(1);`
      `func(2);`
      `func(3);`
      `}`
  - Eurekas
    - Parallel break
    - Recursive escape
    - Algorithm Race
  - Restartable Scopes
    - Resilience
    - `err_restart();`

Eureka Concept Example

```
proc search_inner(var target:int,
                  var n:node):node {
  if (n.value == target) {
    eureka;
    return n;
  } else {
    begin search_inner(target, n.leftchild);
    begin search_inner(target, n.rightchild);
  }
}

proc search(var target:int, var n:node):node {
  var f:node;
  newteam_begin f = search_inner(target, n);
}

var found : node = search(5, bigtree.root);
writeln("text of node 5 is: " + found.text);
```



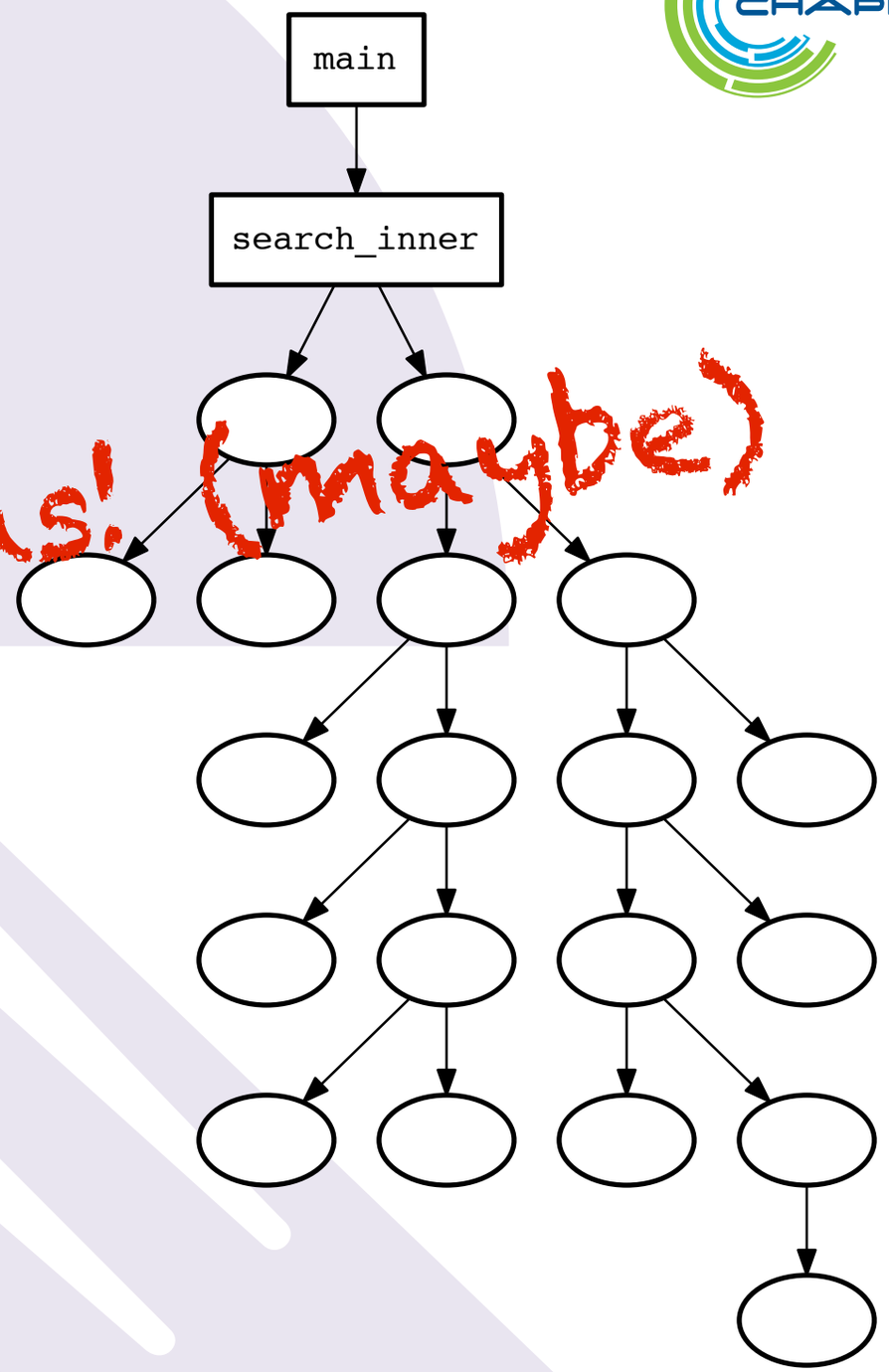
Eureka Concept Example

```
proc search_inner(var target:int,
                  var n:node):node {
  if (n.value == target) {
    eureka;
    return n;
  } else {
    begin search_inner(target, n.leftchild);
    begin search_inner(target, n.rightchild);
  }
}

proc search(var target:int, var n:node):node {
  var f:node;
  newteam_begin f = search_inner(target, n);
}

var found : node = search(5, bigtree.root);
writeln("text of node 5 is: " + found.text);
```

New Keywords! (maybe)

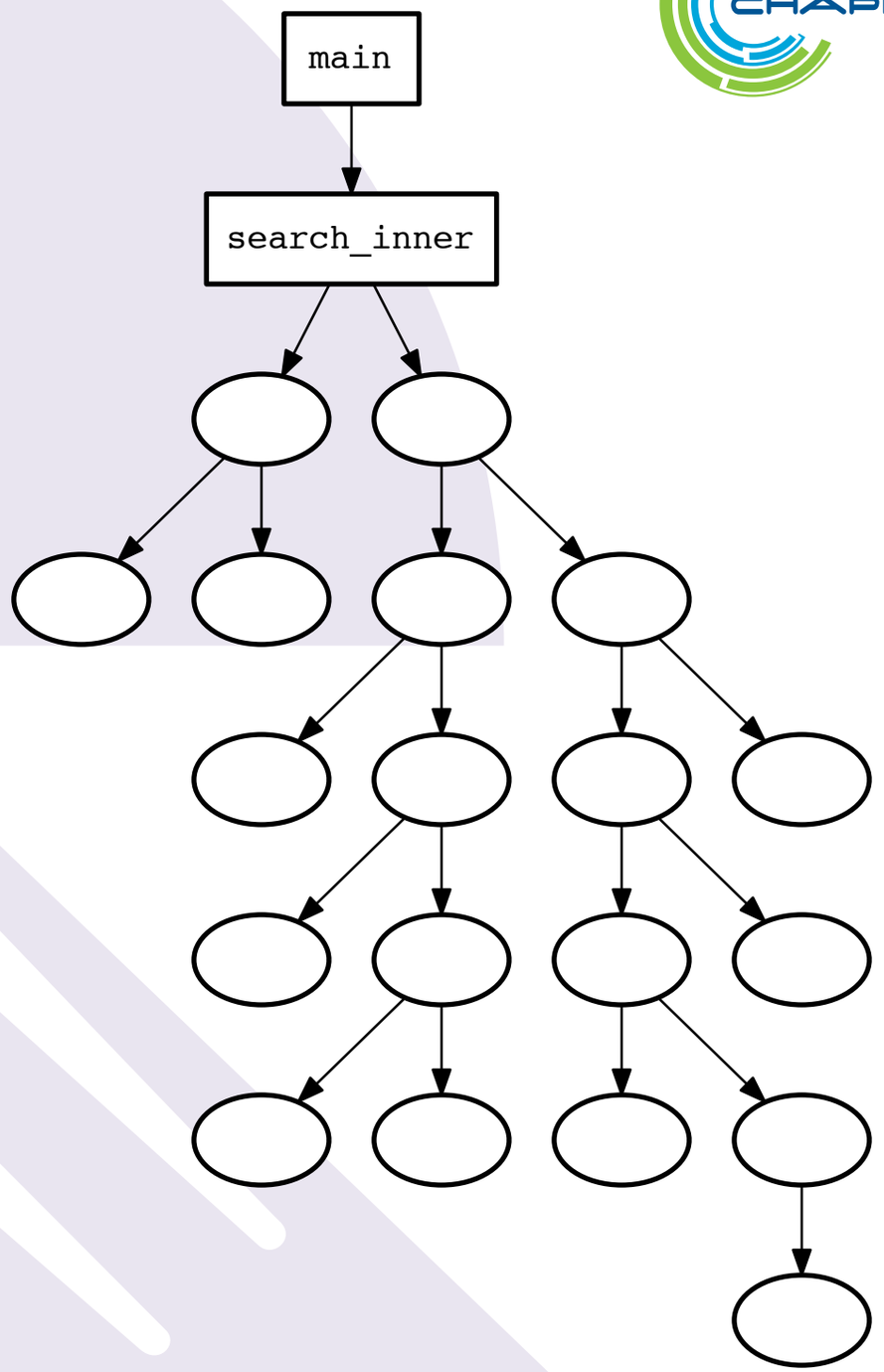


Eureka Concept Example

```
proc search_inner(var target:int,
                  var n:node):node {
  if (n.value == target) {
    eureka;
    return n;
  } else {
    begin search_inner(target, n.leftchild);
    begin search_inner(target, n.rightchild);
  }
}

proc search(var target:int, var n:node):node {
  var f:node;
  newteam_begin f = search_inner(target, n);
}

var found : node = search(5, bigtree.root);
writeln("text of node 5 is: " + found.text);
```

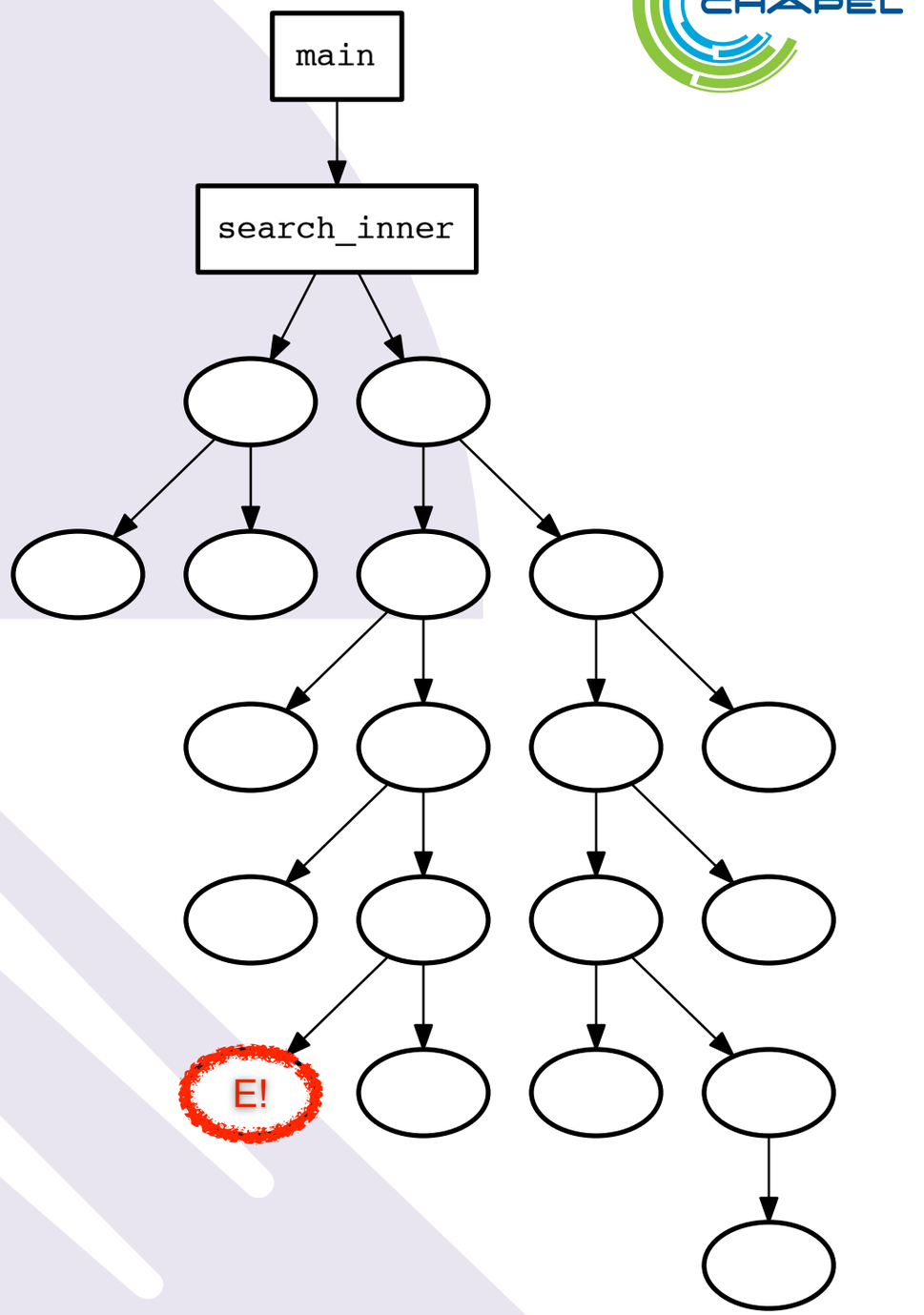


Eureka Concept Example

```
proc search_inner(var target:int,
                  var n:node):node {
  if (n.value == target) {
    eureka;
    return n;
  } else {
    begin search_inner(target, n.leftchild);
    begin search_inner(target, n.rightchild);
  }
}

proc search(var target:int, var n:node):node {
  var f:node;
  newteam_begin f = search_inner(target, n);
}

var found : node = search(5, bigtree.root);
writeln("text of node 5 is: " + found.text);
```

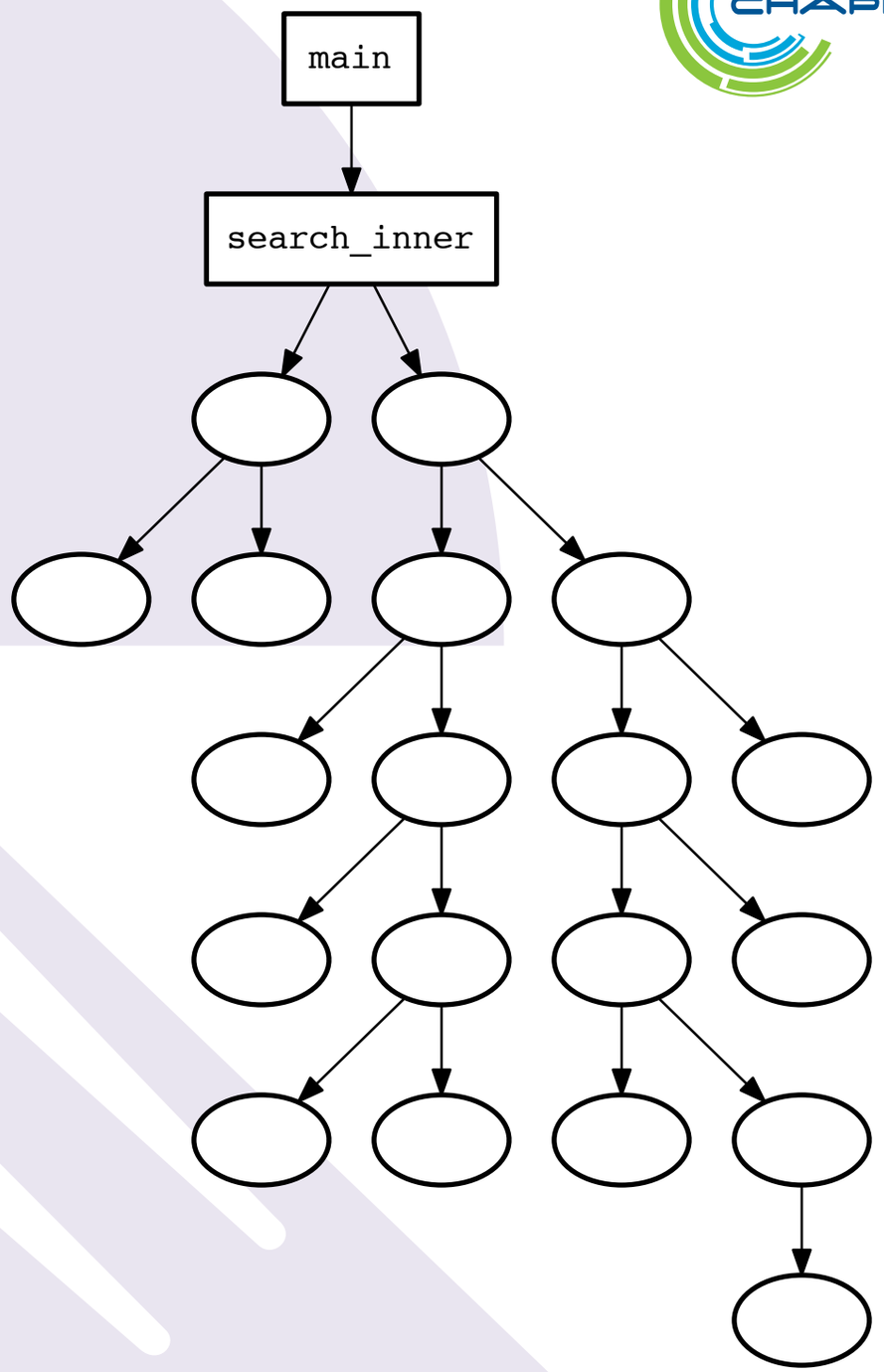


Eureka Concept Example

```
proc search_inner(var target:int,
                  var n:node):node {
  if (n.value == target) {
    eureka;
    return n;
  } else {
    begin search_inner(target, n.leftchild);
    begin search_inner(target, n.rightchild);
  }
}

proc search(var target:int, var n:node):node {
  var f:node;
  newteam_begin f = search_inner(target, n);
}

var found : node = search(5, bigtree.root);
writeln("text of node 5 is: " + found.text);
```

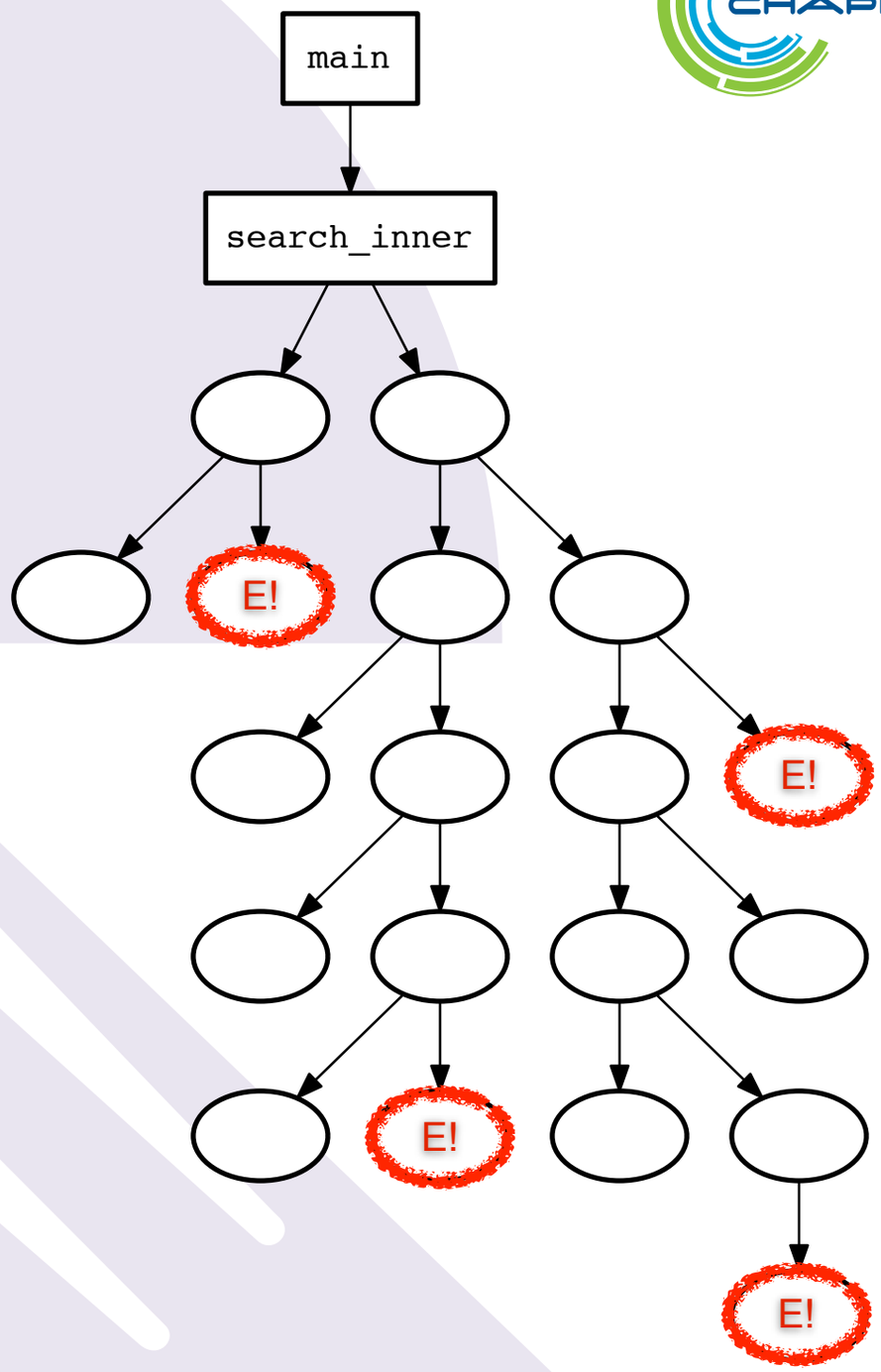


Eureka Concept Example

```
proc search_inner(var target:int,
                  var n:node):node {
  if (n.value == target) {
    eureka;
    return n;
  } else {
    begin search_inner(target, n.leftchild);
    begin search_inner(target, n.rightchild);
  }
}

proc search(var target:int, var n:node):node {
  var f:node;
  newteam_begin f = search_inner(target, n);
}

var found : node = search(5, bigtree.root);
writeln("text of node 5 is: " + found.text);
```



Current Work: SPR

- Integrated tasking runtime and communication
  - Remote task spawn
  - Local task synchronization
  - Data movement
- Opportunities for optimized interplay
  - Synchronization across the network
  - Work-stealing and load-balancing
  - Progress support
- Transparent to the user
  - `CHPL_TASKS=spr`
  - `CHPL_COMM=spr`