# Pip: Detecting the Unexpected in Distributed Systems

Patrick Reynolds, Janet Wiener, Amin Vahdat
*Duke University*
Jeff Mogul, Mehul Shah
*HP Labs*
*UCSD*

## Overview

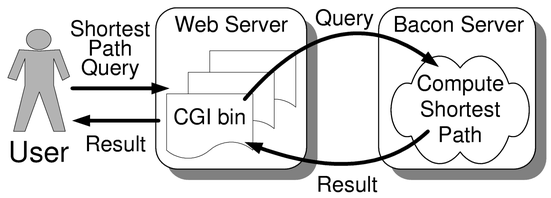
- Many bugs result from mismatches between actual and expected system behavior
  - Structural – inappropriate order or placement of processing and/or communication
  - Performance – resource consumption or delays higher or lower than expected
- Pip reports behavior that violates expectations – potential bugs
- To use Pip, the programmer writes:
  - *Annotations* – additions to system source code to trace relevant activity
  - *Expectations* – description of expected system behavior in high-level language

## Using Pip

1. **Annotate** application source code
2. **Write** expectations for the application
3. **Run** application to collect traces of actual behavior
4. Use Pip to **import** reconciled events into SQL database
5. Run Pip's checker to **check** application behavior against expectations
6. **Explore** some or all application behavior

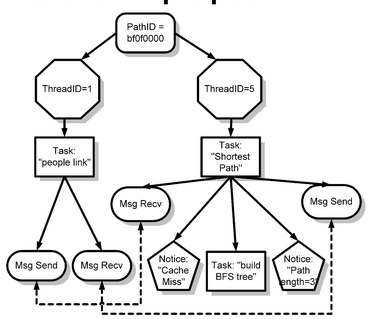
Checking and exploring can be done off-line or in real time

## Bacon: a simple example system




- *Oracle of Bacon* – find shortest paths between actors
- One server, many clients
- Significant CPU processing on server
  - Up to 2 seconds per query

### One example path




## Annotations

- Small source-code changes to important modules
  - Highlight key communication and processing

**Client annotations**

```
ANNOTATE_INIT();
pathid = get_time_in_microseconds();
ANNOTATE_SET_PATH_ID(pathid);
ANNOTATE_START_TASK("link people");
query = parse_user_query();
msgid = get_time_in_microseconds();
ANNOTATE_SEND(msgid, sizeof(query));
write(fd, {pathid, msgid, query});
read(fd, {return_msgid, answer});
ANNOTATE_RECEIVE(return_msgid, sizeof(answer));
display_answer(answer);
ANNOTATE_END_TASK("link people");
ANNOTATE_END_PATH_ID(pathid);
```

**Server annotations**

```
read(fd, {pathid, msgid, query});
ANNOTATE_SET_PATH_ID(pathid);
ANNOTATE_START_TASK("shortest path");
ANNOTATE_RECEIVE(msgid);
index_A = lookup_name(query.name_A);
index_B = lookup_name(query.name_B);
if (tree_A = cache_lookup(index_A)) {
   ANNOTATE_NOTICE("cache hit");
}
else {
   ANNOTATE_NOTICE("cache miss");
   ANNOTATE_START_TASK("build BFS tree");
   tree_A = build_bfs_tree(index_A);
   ANNOTATE_END_TASK("build BFS tree");
}
walk_tree(tree_A, index_B, &answer, &nhops);
ANNOTATE_NOTICE("path length=%d", nhops);
write(fd, {return_msg=count++, answer});
ANNOTATE_SEND(return_msgid, sizeof(answer));
ANNOTATE_END_TASK("shortest path");
```

- Bacon annotations: 84 lines for 1919 lines of source
- Other systems:
  - Bullet: 77 lines of annotations, 6425 lines of source
  - RanSub: 17 lines of annotations, 1711 lines of source
  - SplitStream: 108 lines of annotations, 3118 lines of source
- Typically ~10 lines per 1000

## Expectations

- Expectations file describes how to recognize and classify paths
  - Separate from application source code
  - Paths not recognized by any expectation may indicate bugs
  - Aggregate expectations apply to sets of paths

**Sample expectations**

```
validator ShortestPath
     thread Client(*, 1)
        task("link people")
           send(Server) limit(SIZE, {20b,200b});
           recv(Server);
     thread Server(*, 1)
        task("shortest path")
           recv(Client);
           repeat between 0 and 2
              task("find person") limit(CPU_TIME, 400ms);
           xor
              branch: notice("cache miss");
                      task("build BFS tree") limit(CPU_TIME, 1.0s);
              branch: notice("cache hit");
              branch:  // person not found
           notice(m/^path length=\d+$/);
           send(Client);

assert(average(CPU_TIME, ShortestPath) < 0.6s);
assert(instances(CacheHit) / instances(CacheMiss) >= 0.9);
```

**Resource metrics**

- Real time, CPU time, page faults, context switches
- Message count, size, and latency
- Tree height, thread count

## Experience

**Sources of unexpected behavior**

- Actual bugs: errors in application source code
- Incorrect annotations: too many, too few, wrong Ids
- Incorrect expectations

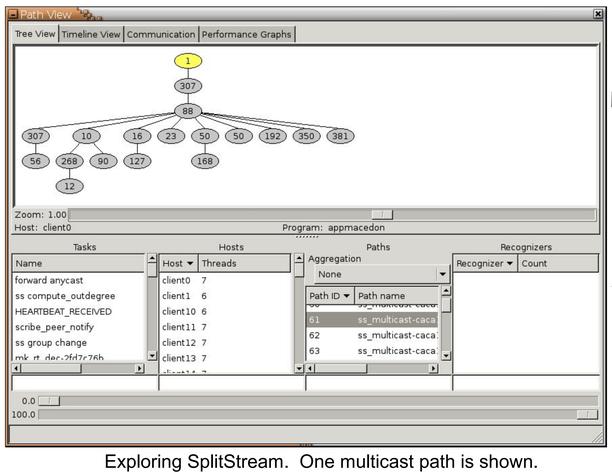
**Oracle of Bacon**

- Found two instances of unexpected behavior
  - Operations with high delay but low CPU time
    - Blocked on I/O due to single-threaded server design
  - Unusually low cache-hit rate for one trace

**Other systems**

- **RanSub**
  - Erroneous start-up behavior
    - Some variables uninitialized
    - Causes extra messages and notices – unexpected structure
  - Excessive CPU time in recent reimplementation
- **Bullet**
  - Truncated paths at the end of the trace
    - Some messages lost – unexpected structure
    - Not a bug worth fixing
- **SplitStream**
  - *Heartbeat* paths with hundreds of events
    - Path IDs reused – annotation error

## Exploring behavior




See one path or aggregate performance information

Pick a task, host, path, or set of paths to examine

Constrain view to a limited time range

Exploring SplitStream. One multicast path is shown.