Distributed Computation in P4

Chesley Tan and Danny Qiu

Introduction

- Network switches are becoming increasingly powerful, but they are still often highly memory-constrained.
- With the growth in programmable switches, end-users are able to take full advantage of the computing capabilities of their network switches.
- The ubiquity of network switches lends itself to distributed applications.

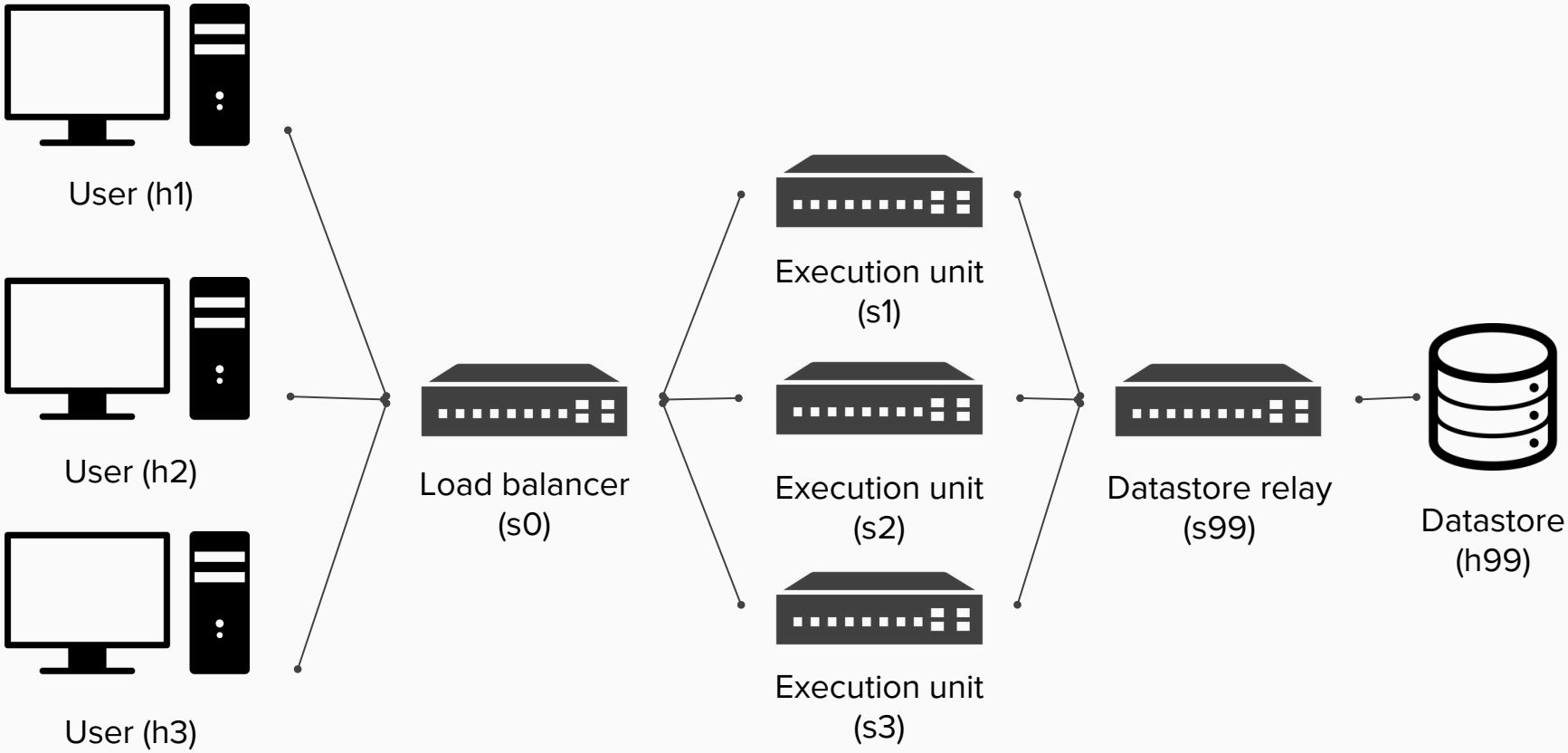
Distributed Computation in P4

- We can model the execution of a multi-processor computer on a network of programmable switches running P4 code.
- The distributed computing system leverages an instruction-based execution model with an assembly language derived from the RISC-V ISA.
- To provide the abstraction of “unlimited” memory on the programmable switches, we use an external datastore with two-way communication with the switches.

Assembly

- Register-register arithmetic instructions
  - ADD, SUB, AND, OR, XOR, SLT, SLTU, SRA, SRL, SLL, MUL
- Register-immediate arithmetic instructions
  - ADDI, ANDI, ORI, XORI, SLTI, SLTIU, SRAI, SRLI, SLLI, LUI, AUIPC
- Memory instructions
  - LW, SW
- Unconditional jump instructions
  - JAL, JR, JALR
- Conditional branch instructions
  - BEQ, BNE, BLT, BGE, BLTU, BGEU

System Architecture



Properties

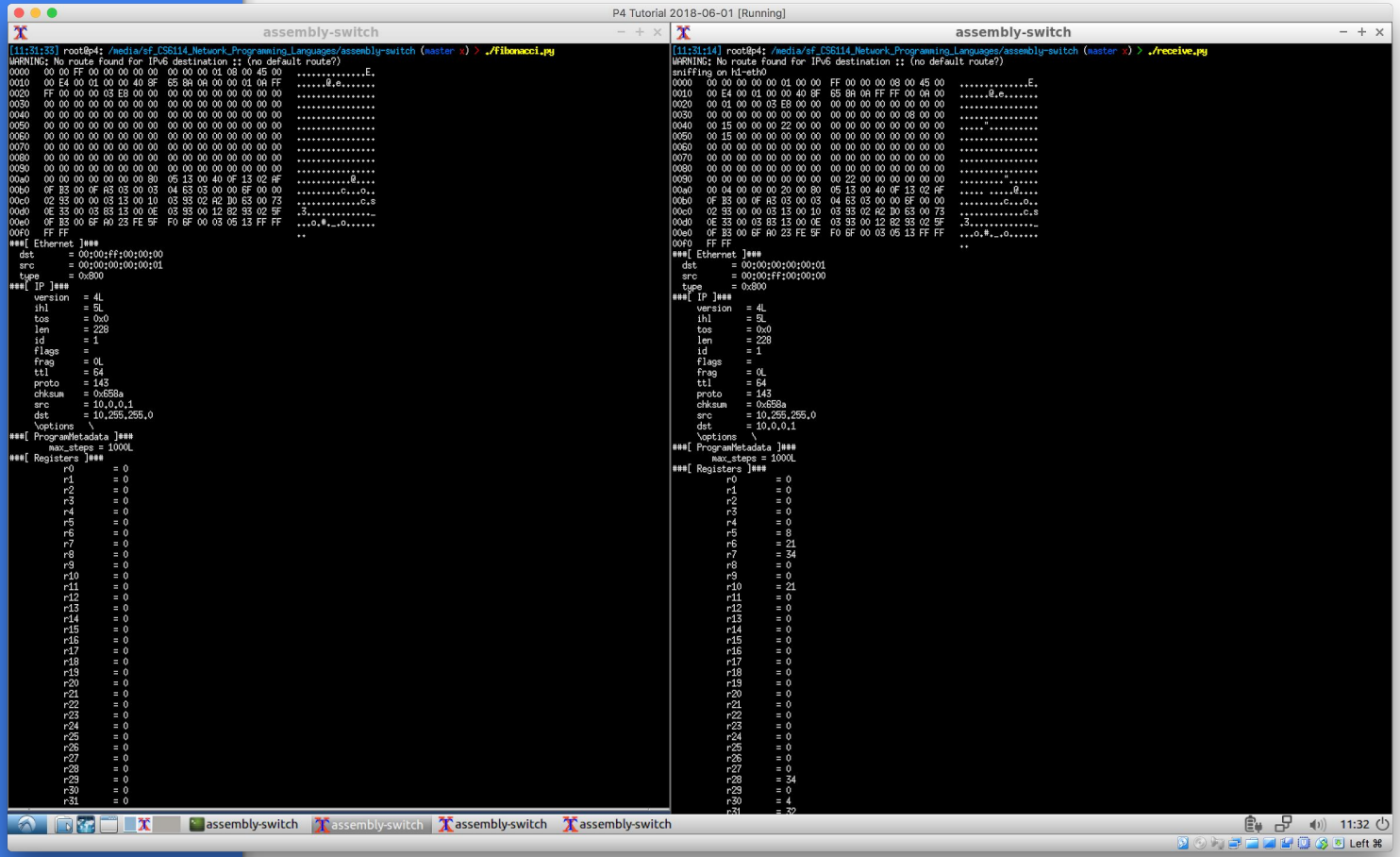
- The system allows us to take advantage of the computing power offered by programmable switches to execute arbitrary programs.
- The system allows programmable switches to leverage “unlimited” memory using the datastore.
- The computation system naturally simulates multi-programming, with the packet scheduler acting as a thread scheduler.

Progress

- Fully implemented ¯_(ツ)_/¯
- Tested using unit tests and (memoized) fibonacci program

In Action:

Memoized Fibonacci!



Future work

- Limitations
  - Reliability
    - The network is unreliable and packet buffers could fill up or packets may be corrupted. Dropping packets will cause the program to fail to completely execute.
    - A simple, but inefficient workaround is to re-transmit packets after a timeout when a response has not been received, but partial execution can leave memory in an unpredictable state.

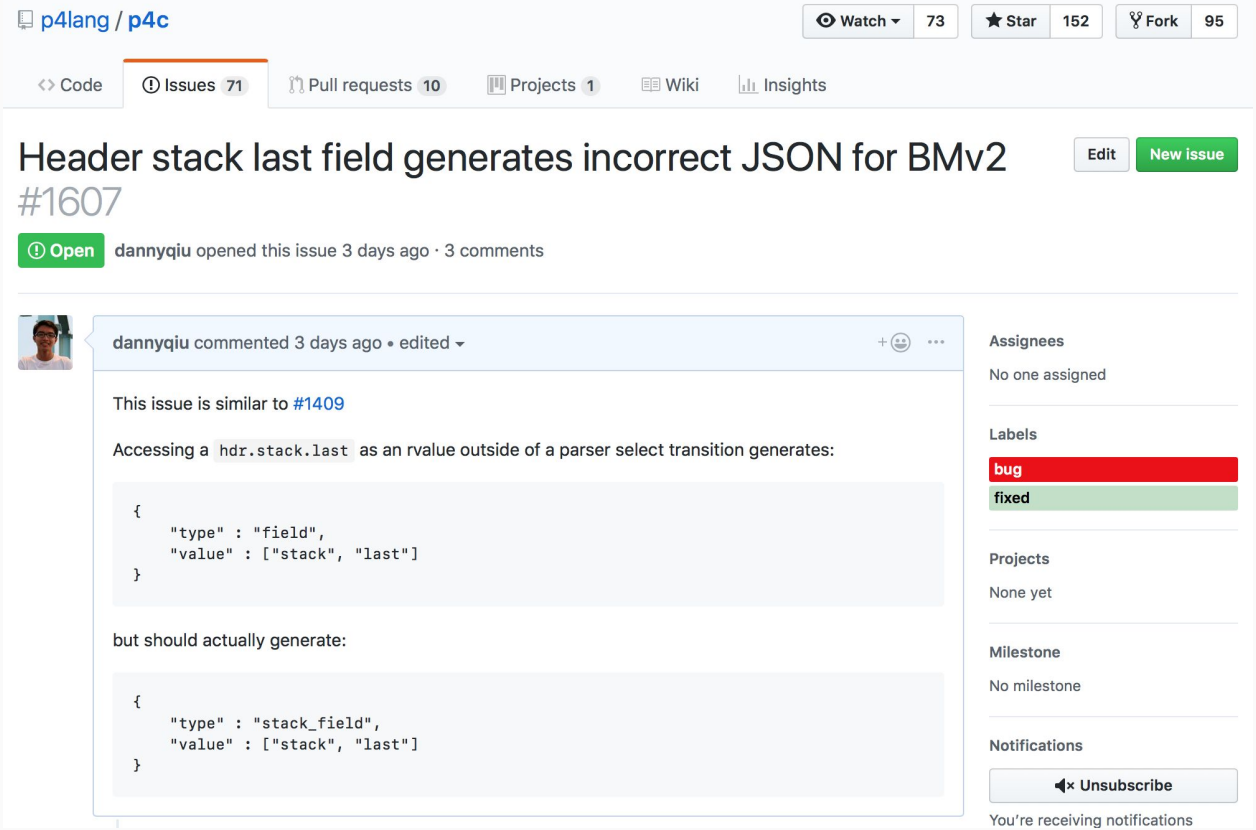
Future work

- Limitations
  - Program size limitation
    - The entire program must be stored within the headers of the packet. Our system enforces a limit of 300 instructions per program to satisfy the maximum ethernet packet size constraint.
    - Several potential solutions: separating large programs into chunks, executing programs as sequences of basic blocks, storing the program assembly in memory and executing them via interpretation.
- Custom extensions to assembly language
  - The assembly language can be extended with custom instructions, such as for retrieving information about the state of the switch or the network.

Future work

- Failure reporting
  - Loads/stores with non word-aligned addresses cause the program execution to abort, but no failure indication is sent back to the user.

Finding bugs in P4c (secretly what this class is about)



p4lang / p4c

Watch 73 | Star 152 | Fork 95

Code | Issues 71 | Pull requests 10 | Projects 1 | Wiki | Insights

Header stack last field generates incorrect JSON for BMv2 #1607

Edit | New issue

Open dannyqiu opened this issue 3 days ago · 3 comments

dannyqiu commented 3 days ago • edited

This issue is similar to #1409

Accessing a `hdr.stack.last` as an rvalue outside of a parser select transition generates:

```
{
    "type" : "field",
    "value" : ["stack", "last"]
}
```

but should actually generate:

```
{
    "type" : "stack_field",
    "value" : ["stack", "last"]
}
```

Assignees
No one assigned

Labels
bug
fixed

Projects
None yet

Milestone
No milestone

Notifications
Unsubscribe
You're receiving notifications

Questions?