Parallelizable Hashing

An industry viewpoint

Scott Fluhrer
Cisco Systems

8/22/14

Requirements

- Cryptographical Strength
- Architectural Independence
  - We have no idea what processors will look like 10 years from now
    - SIMD
    - Multiple Core
    - Mixture of the two
    - Pipelined???

Requirements

- We need different processors to compute the same hash
  - Including processors not designed when we pick the hash function parameters
  - This means we would prefer not to have a parameterized hash tuned to a specific implementation

We would prefer one hash function that allows implementation options

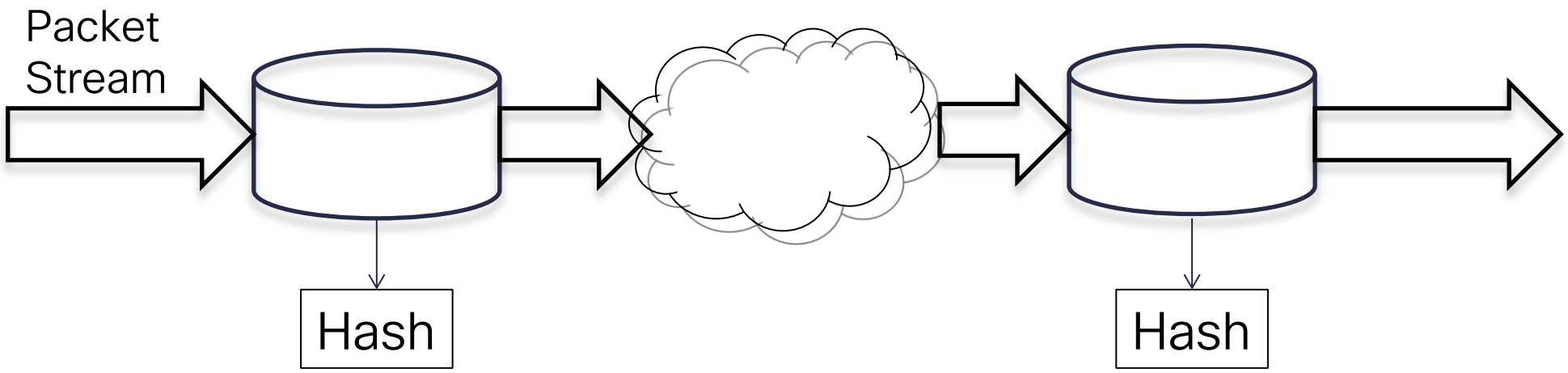
Requirements

- Online Computation
  - We don’t always have all the data we’re hashing in front of us at once
  - Hashing network data needs to deal with small updates
    - The segmentation option doesn’t do this well

Requirements

Here is one scenario where this can happen:

- We have a stream of packet data we are hashing
- Because each packet is small, we are never able to present a large segment to the hashing function



What does this mean?

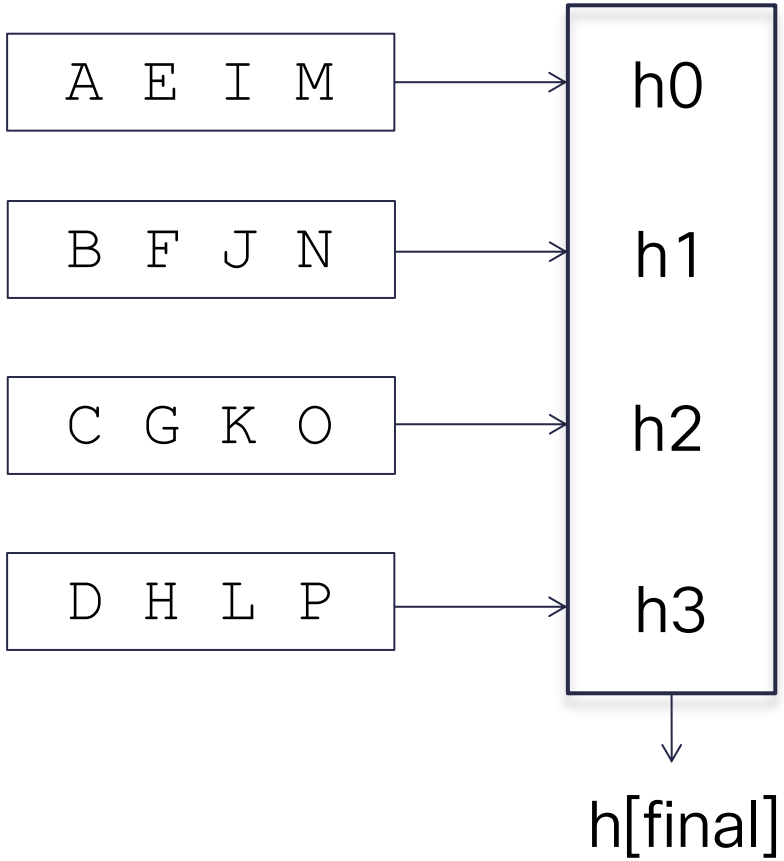
- We want one hash function that works well everywhere, and does everything
  - Can we actually design something that is both SIMD and multi-core friendly, is not tied to a number of tracks, and can handle short updates?
  - Perhaps we can come closer if we consider a multilevel design

One level Interleave Design

It has the number of tracks inherent in the design

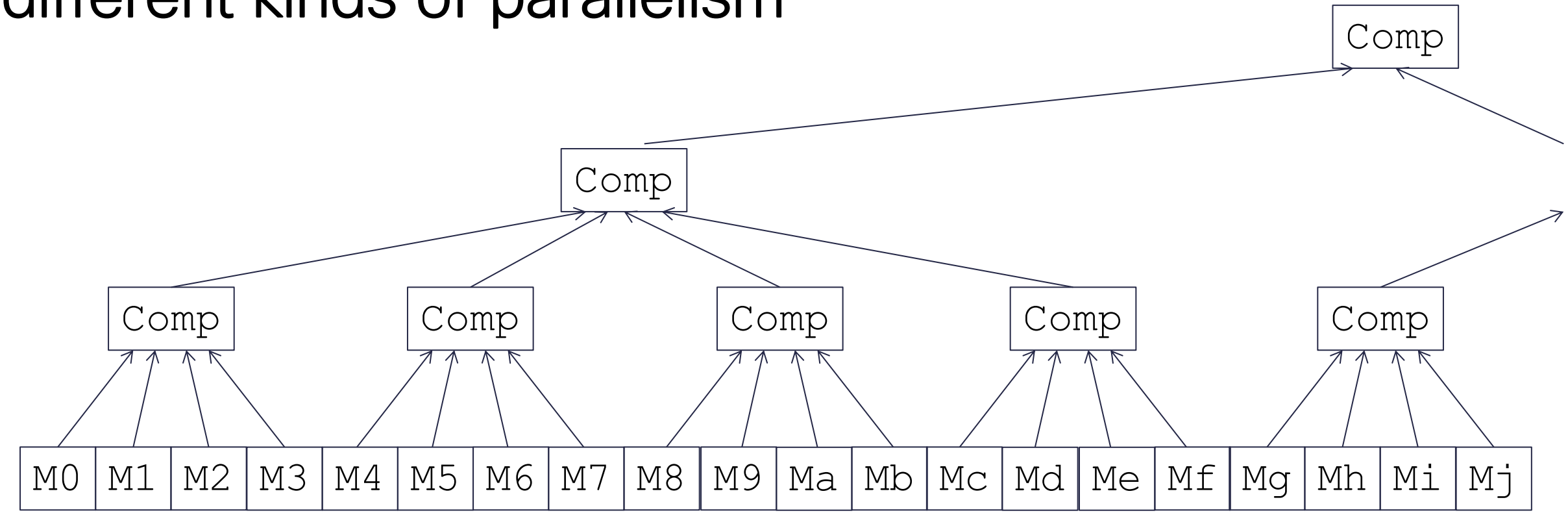
This interleaved design can easily handle SIMD parallelism of 4, but can't take advantage of any more

To take advantage of more, we'd have to change the hash



Multilevel Design

A multilevel design can take advantage of different kinds of parallelism



What is the trade-off?

- Multilevel can give us one hash function that works well in a variety of scenarios
- However, the implementation is more complex

Is the trade-off worth it?