openstack.

April 2019

Policy-Driven Fault Management for NFV Eco System

Akhil Jain (NEC) akhil.jain@india.nec.com
Eric Kao (VMware) ekcs.openstack@gmail..com

Definitions

- Network Function (NF):
  A functional building block in a network
  - packet inspection, CDNs, virus scanner, ...
- Network Function Virtualization (NFV):
  Realizing NFs as virtual appliances
- Virtual Network Function (VNF):
  A network function realized as virtual appliances

Fault Management

- Basic fault recovery is standard
- Complexities beyond the stardard cases:
  - Diversity of fault scenarios
  - Diversity of VNFs
  - Each combination may call for a different fault management response

Fault Scenarios

- Sequence of fault signals over time
- Isolated vs widespread
- Existing or predicted
- Fault types
  - Hard failure
  - Stability
  - Degraded performance
- Fault domains
  - Networking, Host, Storage, Application, etc

Context

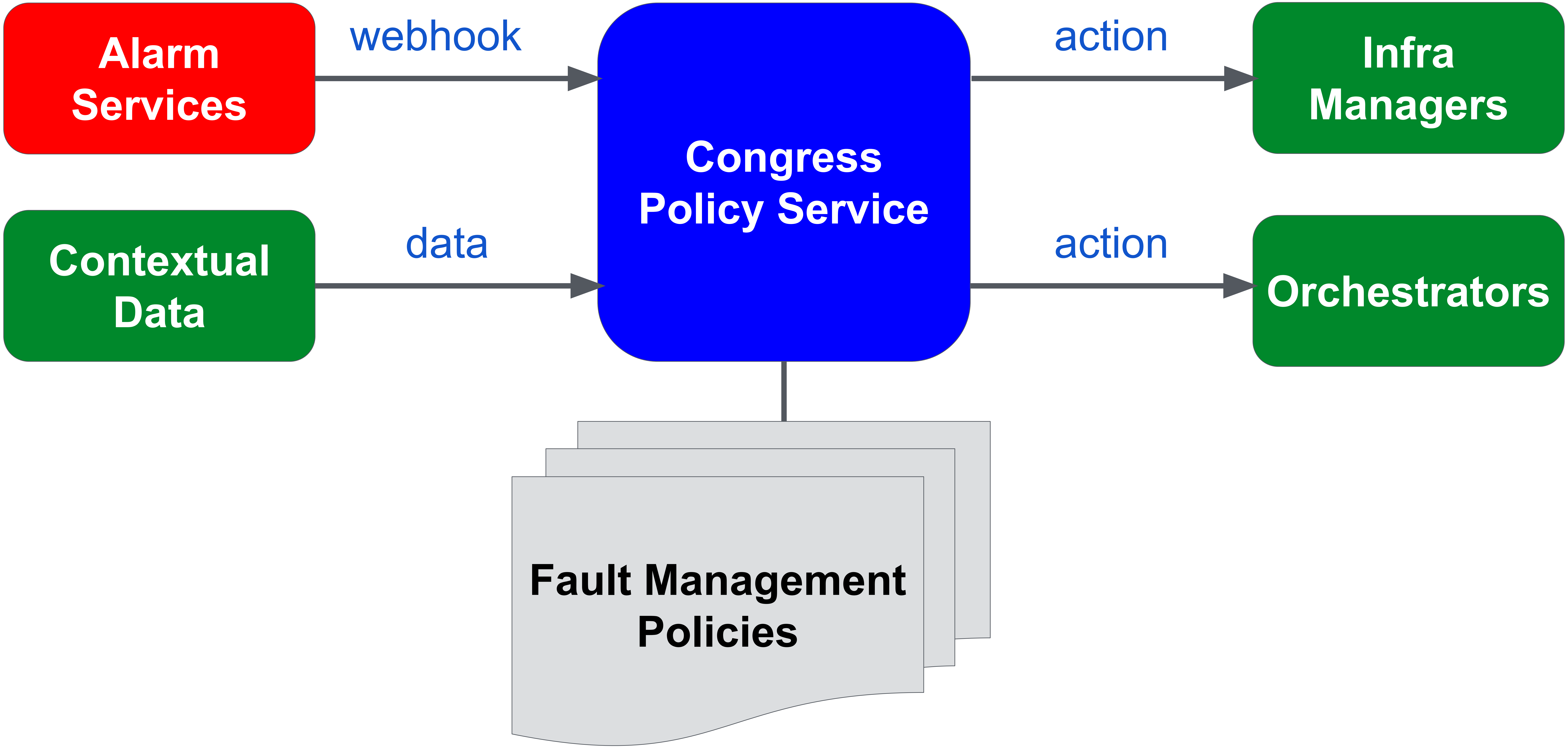
- Current & anticipated loads
- VNF capacity
- Physical infra capacity
- Example considerations:
    - If load << VNF capacity, ignore certain fault prediction signals
    - If load ~= VNF capacity, preemptively scale-out
        - When physical infra limited, may need to scale-in a less loaded or less critical VNF to make room

VNF characteristics

- Stateful vs stateless
- Monolithic vs microservices
- Interactions, topology, service function chaining
- SLAs
- Business/user impact

Solution: Policy-driven fault management

- Fine-grained monitoring & alarming
  - Monasca, Prometheus, ...
- Rich Context
  - Infra managers: Nova, Kubernetes, …
  - NFV orchestrator: Tacker, ONAP, ...
  - application-level statistics: load, latency, throughput
  - Arbitrary data sources
- Expressive policy framework
  - Congress


Alarm Services
webhook
Contextual Data
data
Congress Policy Service
action
Infra Managers
action
Orchestrators
Fault Management Policies

Congress Architecture

- Data
  - Get data from webhooks and APIs
  - Store data as tables and JSON
- Policy
  - Datalog/SQL rules transform data into decisions
- Action
  - Decisions can trigger API calls

Advantages

- Extensible
    - Arbitrary sources of data as needed by use case
- Expressive
    - Not limited by fixed vocabulary or set of properties
- Declarative
    - Well understood declarative language for expressing clear and manageable policies
    - Avoid procedural code

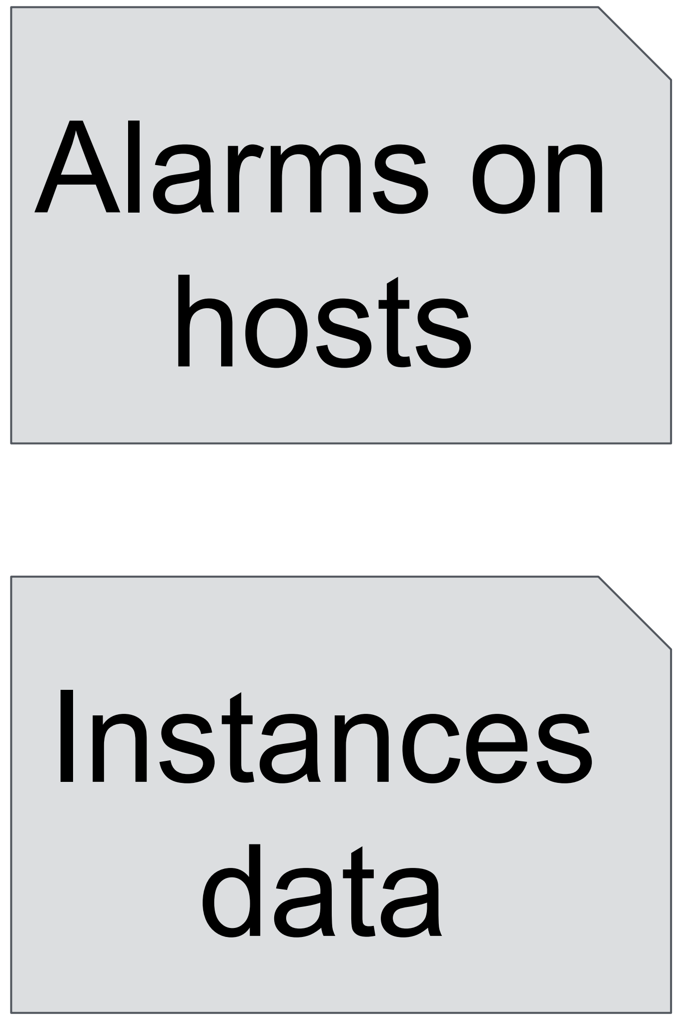
Example: preemptive scale out policy

- Predictive fault signal
- Possible response:
  - Ignore
    - failure occur
    - instances go down
    - load increases
    - autoscaling policy adjusts
- Drawback:
  - Degraded service for a time

Example: preemptive scale out policy

- Estimate service disruption/degradation
- Preemptively scale out as appropriate
- Minimize risk of degraded service

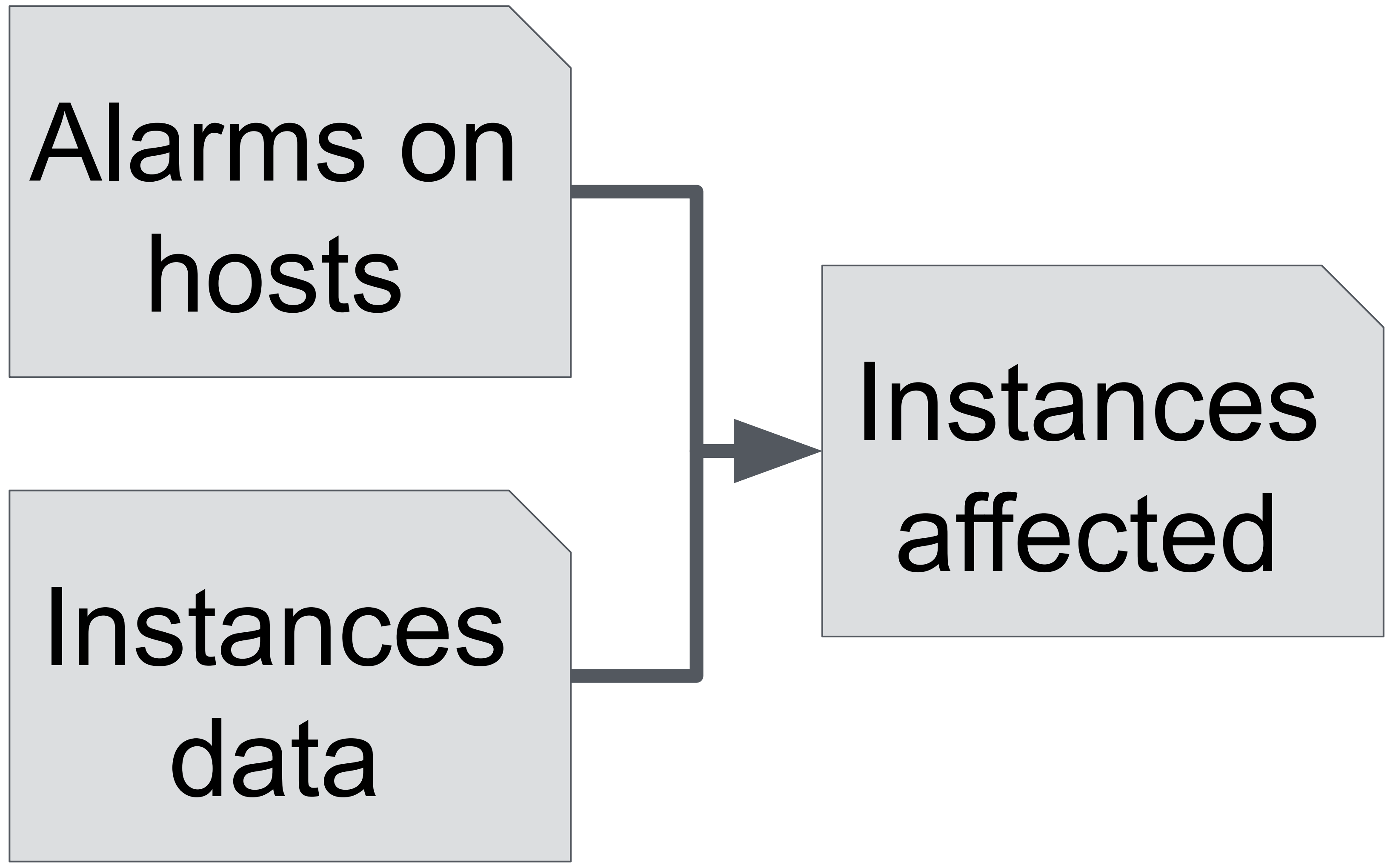
Example: preemptive scale out policy



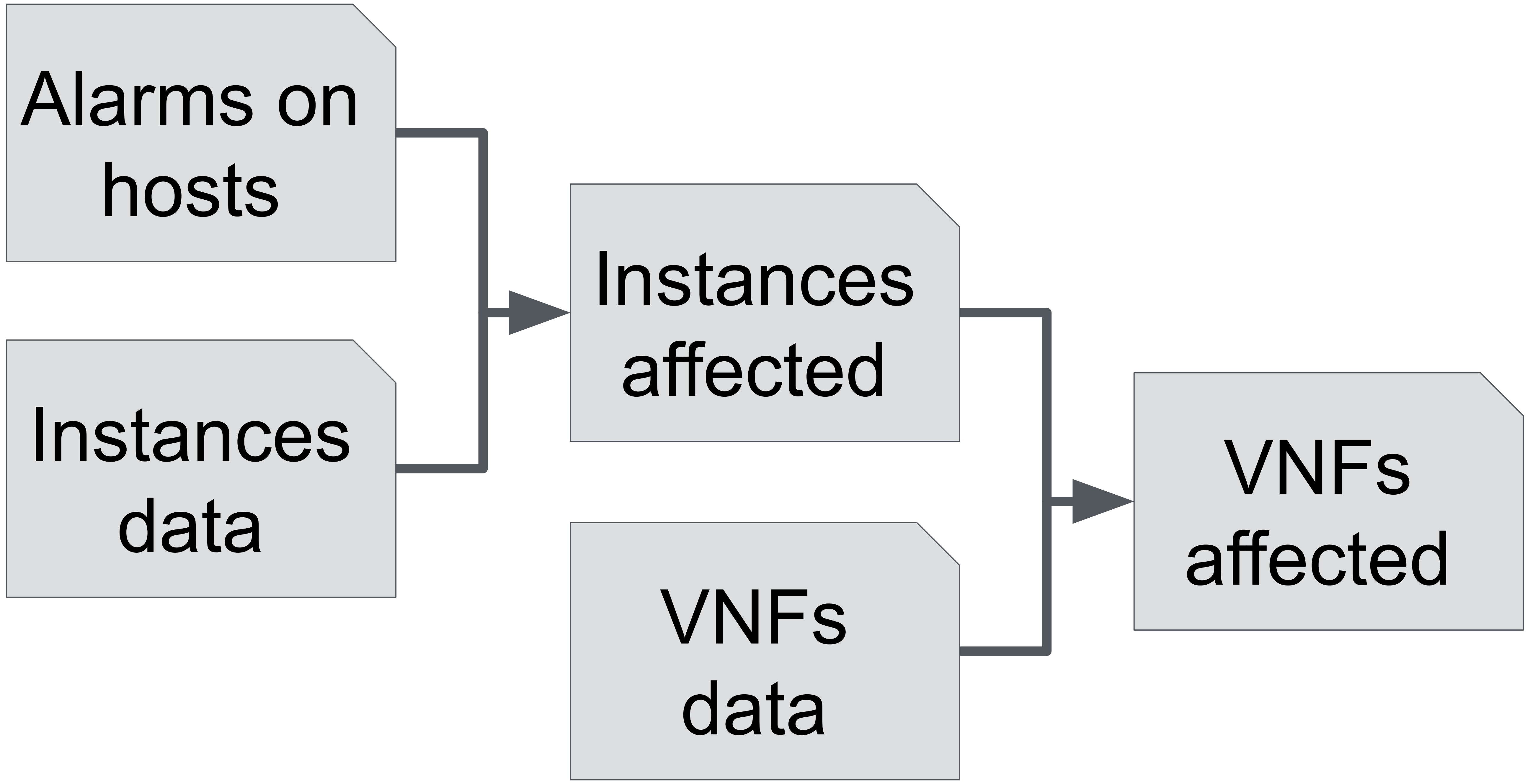
Alarms on hosts

Instances data

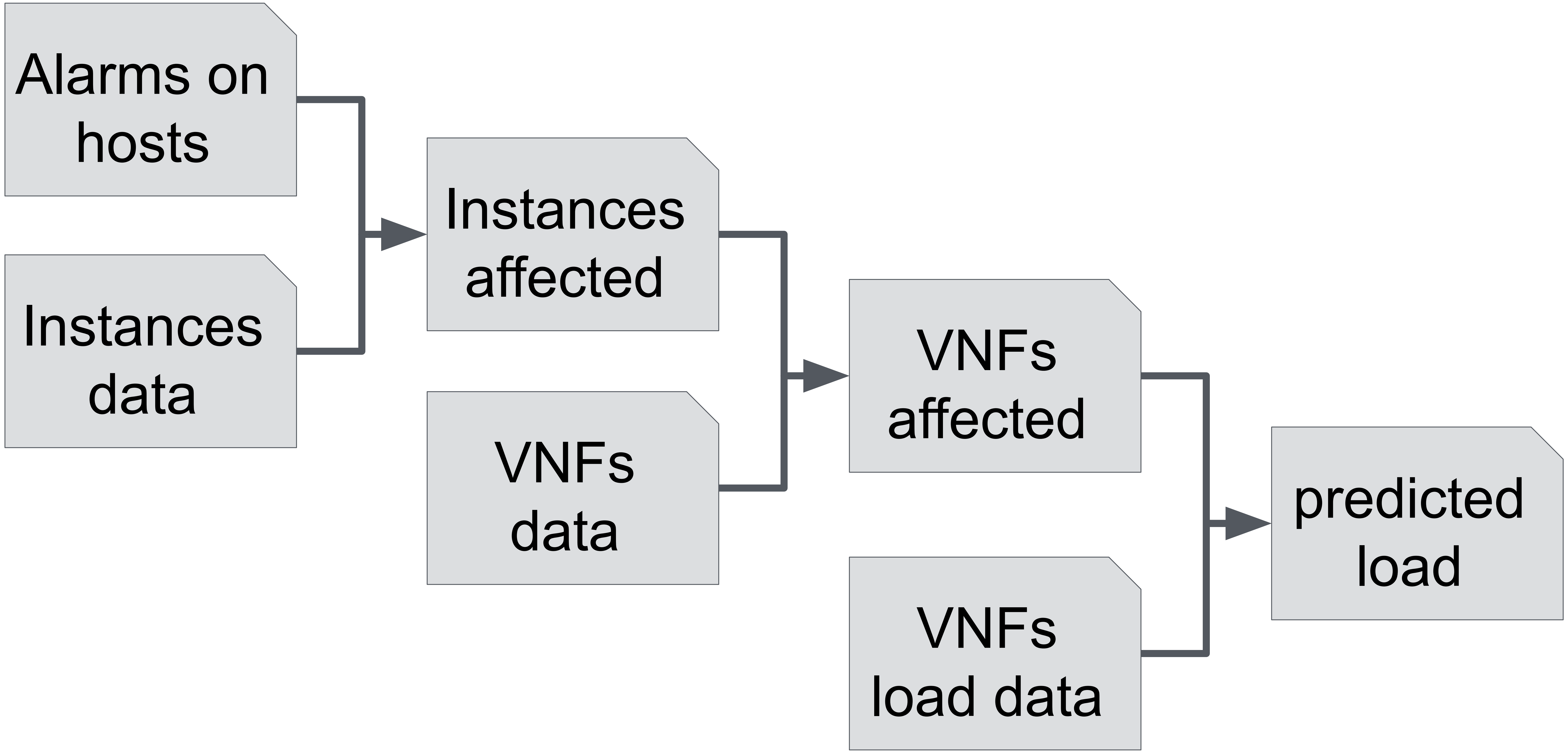
Example: preemptive scale out policy



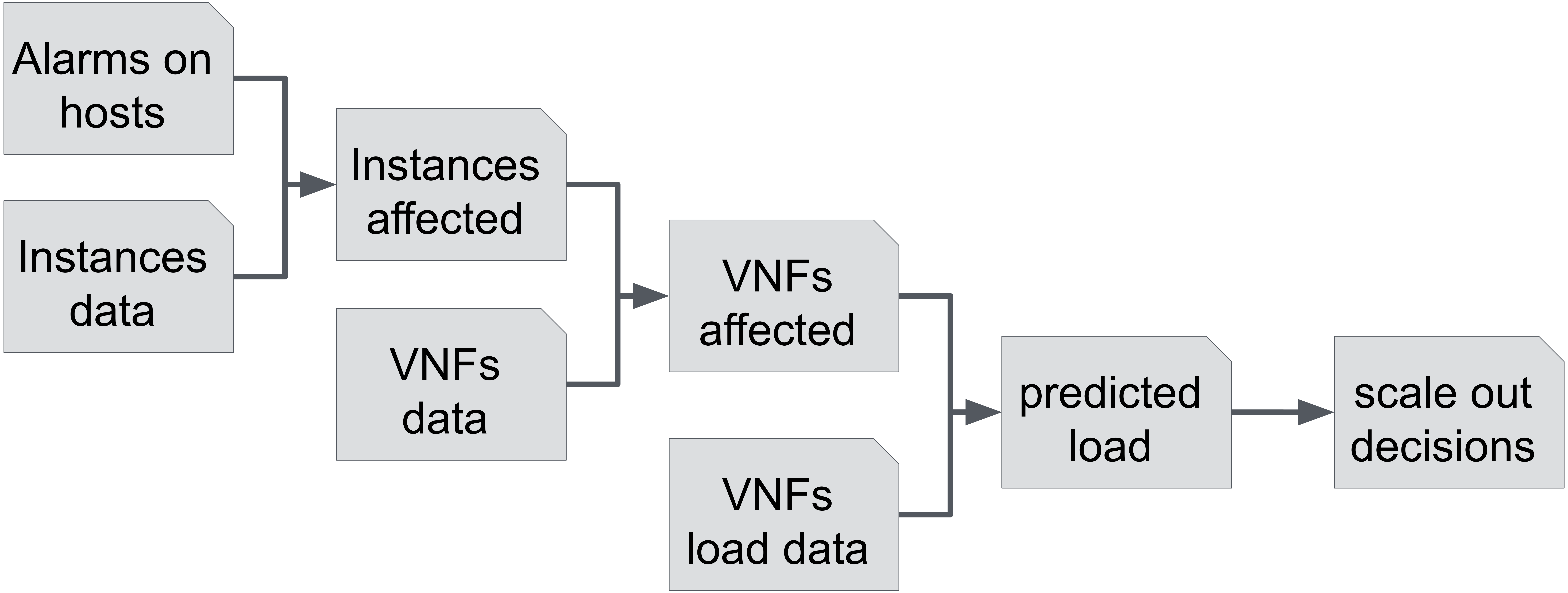
Example: preemptive scale out policy



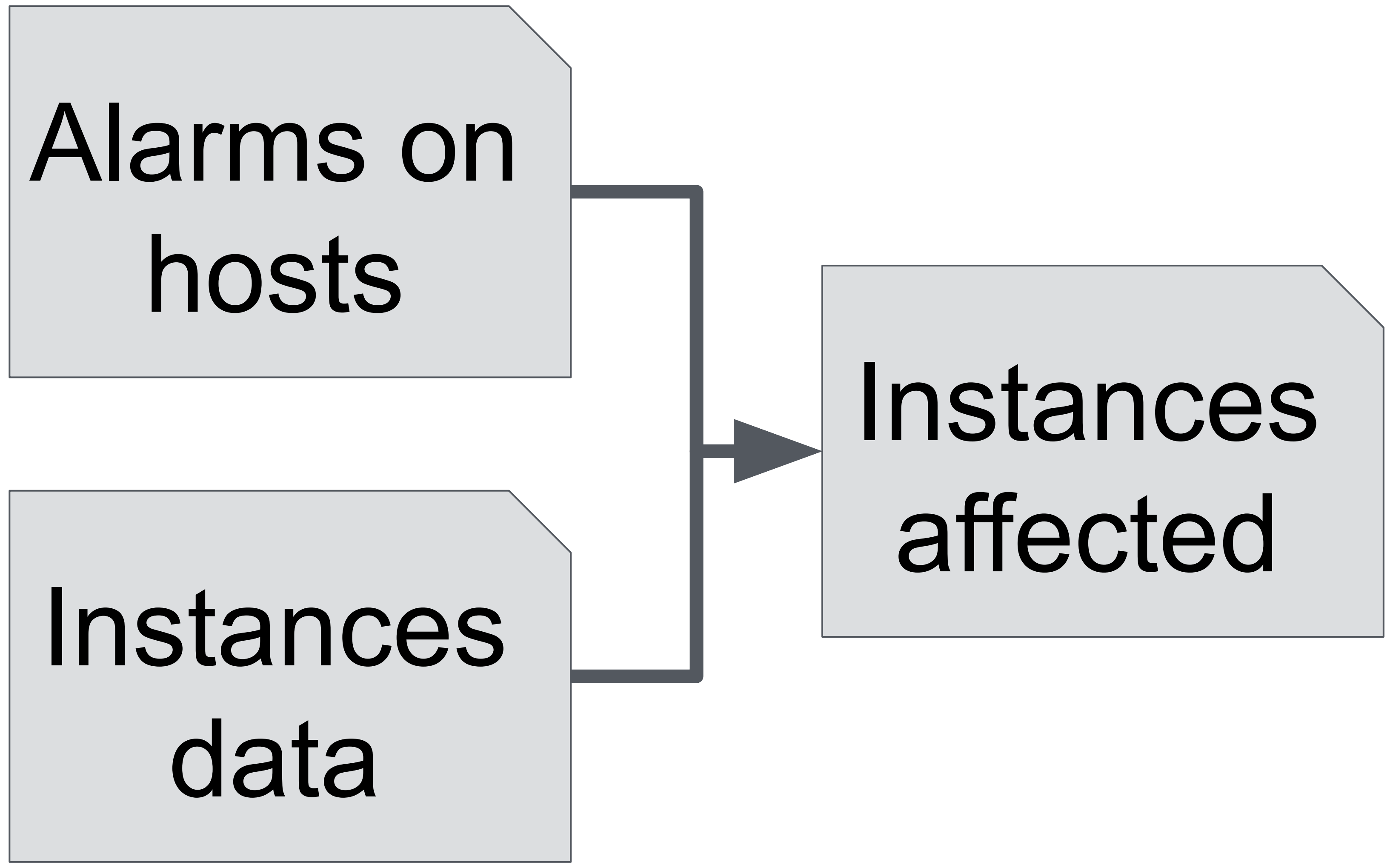
Example: preemptive scale out policy



Example: preemptive scale out policy



Example: preemptive scale out policy



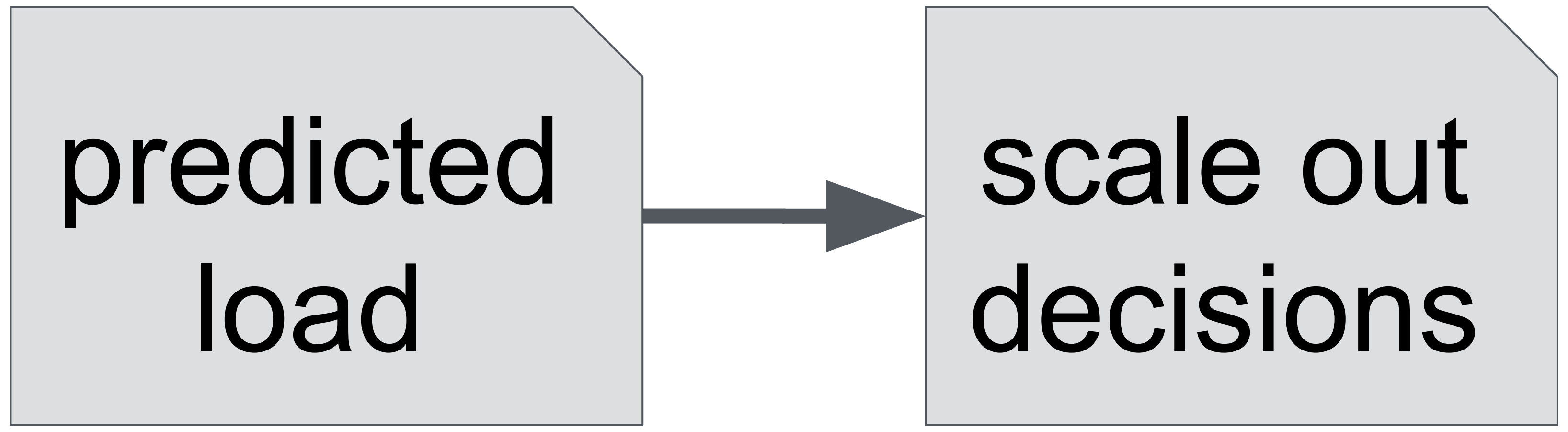
Alarms on hosts

Instances data

Instances affected

```
instances_affected(instance_id) :-
  hosts_alarmed(alarmed_host),
  nova:servers(server_id=instance_id, host_name=alarmed_host)
```

Example: preemptive scale out policy



```
scale_out(vnf_id) :-
  predicted_VNF_load(vnf_id, predicted_load),
  predicted_load > 0.9
```

Demo background

- Demonstrate the interaction between services
    - Setup VNFs with Tacker
    - Configure Congress to receive Monasca webhook
    - Configure Monasca to send webhook
    - Raise Monasca Alarm
    - See result of actions triggered by Congress policy

Summary

- Fault management is complex
  - Diversity of scenarios -> Diversity of response
- Solution
  - Fine-grained monitoring
  - Contextual data
  - Expressive policy
- Congress
  - Pluggable data sources
  - Expressive policy language
  - Triggers API calls

General purpose policy triggers

- Trigger API calls based on policy+data
  - Adv. fault management policies
  - Adv. autoscaling policies
  - Generic integration glue

Feedback welcome!

Mailing lists use **[congress]** prefix
openstack-discuss@lists.openstack.org

Eric Kao <ekcs.openstack@gmail.com>

openstack.

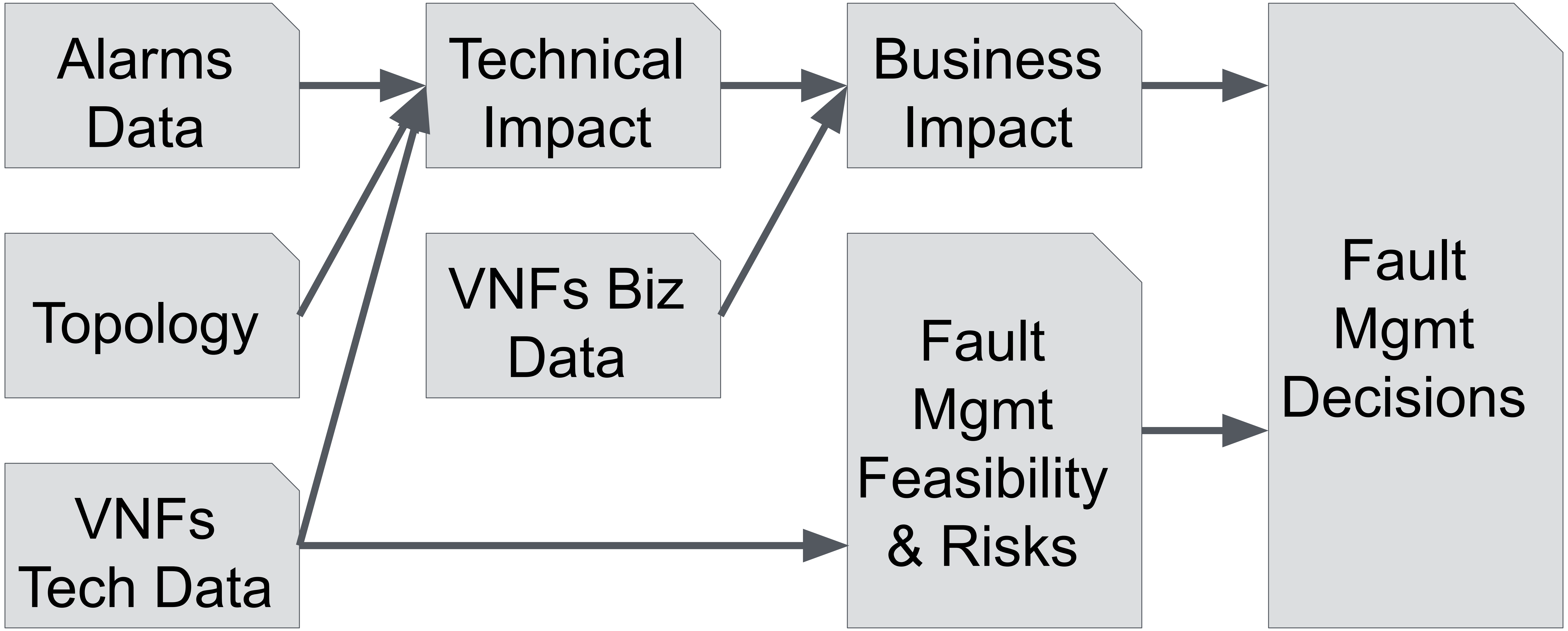
Q&A

Thank you!

Akhil Jain <akhil.jain@india.nec.com>

Eric Kao <ekcs.openstack@gmail.com>

Conceptual policy dataflow


Alarms Data
Topology
VNFs Tech Data
Technical Impact
VNFs Biz Data
Business Impact
Fault Mgmt Feasibility & Risks
Fault Mgmt Decisions