Towards A Scalable Network Management Collector

Athanasios Douitsis

National Technical University of Athens

Network Operations Centre

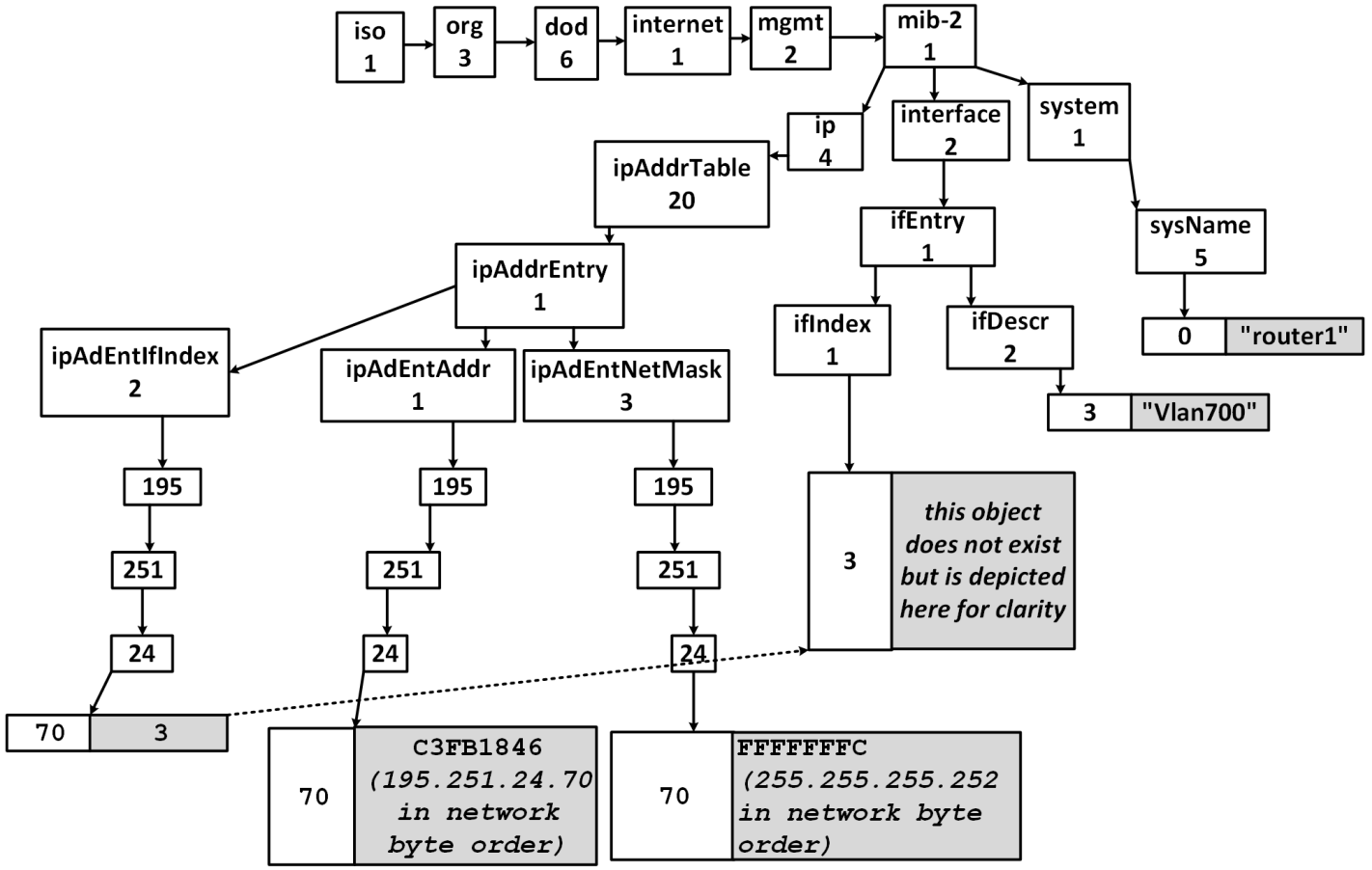
Overview

- Problem Statement
- Contribution overview
- Knowledge Representation
- Architecture
- Implementation
- Performance and Scalability
- Future directions

Problem

- Network Management Data Collection Scalability
  - **SNMP agents in each Network Device or Host**
  - **Moderate/Large** amounts of data per device
  - Short collection cycle
  - **Data transformation** and calculations
  - Storage of processed/transformed data into **Scalable Storage Clusters** such as **ElasticSearch**
- Network Management Knowledge Access and Representation
  - SNMP Structure of Management Information (SMI) meta-schema unsuitable for direct storage in ElasticSearch
  - SNMP protocol transport difficult to use
- NMS Modularity and Extensibility
  - SMI interdependencies difficult to handle
  - MIB specifications in flux and implementations incomplete
  - Semantics and structure of SMI mismatch with native programming languages
    - Only basic functionality from available SNMP libraries

Problem Example: Get interface address

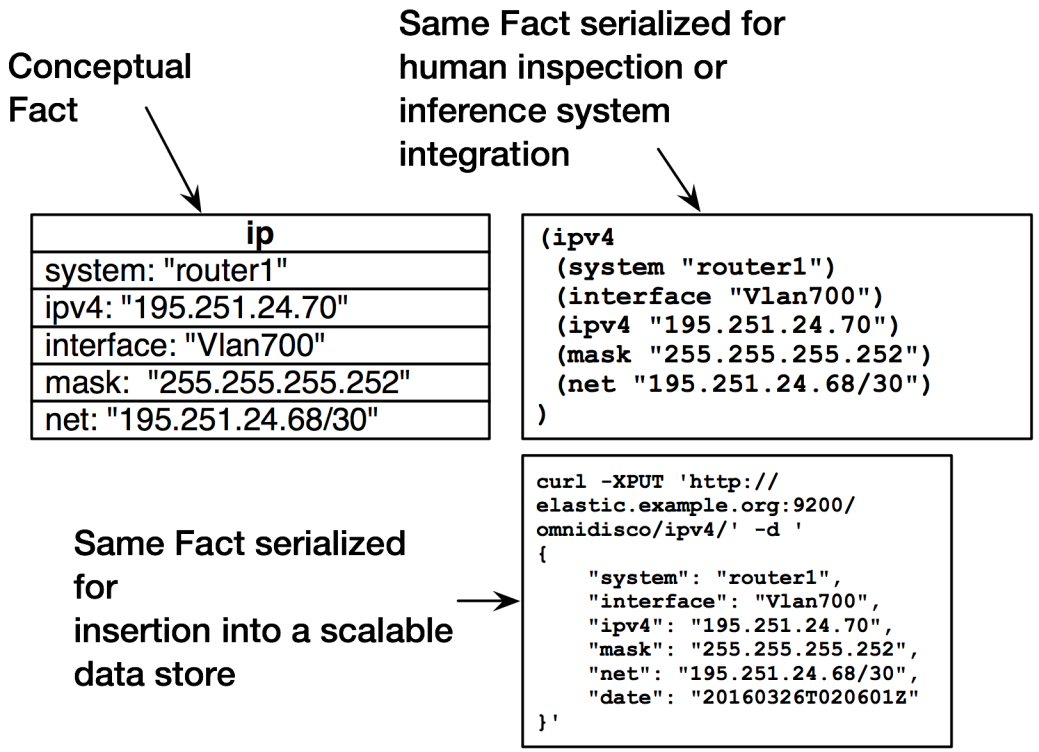


Our Contribution - The **OmniDisco** Management Collector

- Definition of Knowledge Representation Schema
  - Suitable for storage in document-based NoSQL stores, e.g. ElasticSearch
  - Fairly easy for translation from other protocols and specifications, e.g. SNMP SMI
  - Natural representation of common concepts, minimal interdependencies
  - Easy to work with (serialization, handling),
  - **Minimal unnecessary complexity**
    - **Conceptual Complexity**
    - **Programmatic Complexity**
- **Scalable Collection Architecture**
  - Distributed and Network Transparent
  - Efficient
  - Optimal for abovementioned knowledge representation schema
- Implementation
  - SNMP collector reference implementation
  - **Extensible and Pluggable Module Organization** to handle SNMP difficulties in NMS code

Knowledge representation: Facts and FactSets

- Fact:
  - Type (1)
  - Slots (many)
  - Metadata (creation date, etc.)
- Efficiently representable in:
  - Memory
  - JSON
  - YAML
- FactSet:
  - A bag of Facts



OmniDisco Distributed Collector Architecture

- Workers
- Master
- Task Queue
- Cache

Also:

- Scalable Store
- Managed Devices

1. Worker

- Gathering of management data from managed devices
- Use of a Network Management Protocol Implementation
  - SNMP
  - …
- Transformation of collected SNMP raw data to **Facts**
- Production of **FactSets**
- RPC-like semantics, worker callable like a function
- Multiple independent processes
- Multiple processes in many computing nodes

2. Task Queue

- Rendezvous point between:
  - Workers
  - Master
- Asynchronous calling and returning of Tasks
- Network transparent fan-out
- Language agnostic API
- Redundancy possible

3. Master

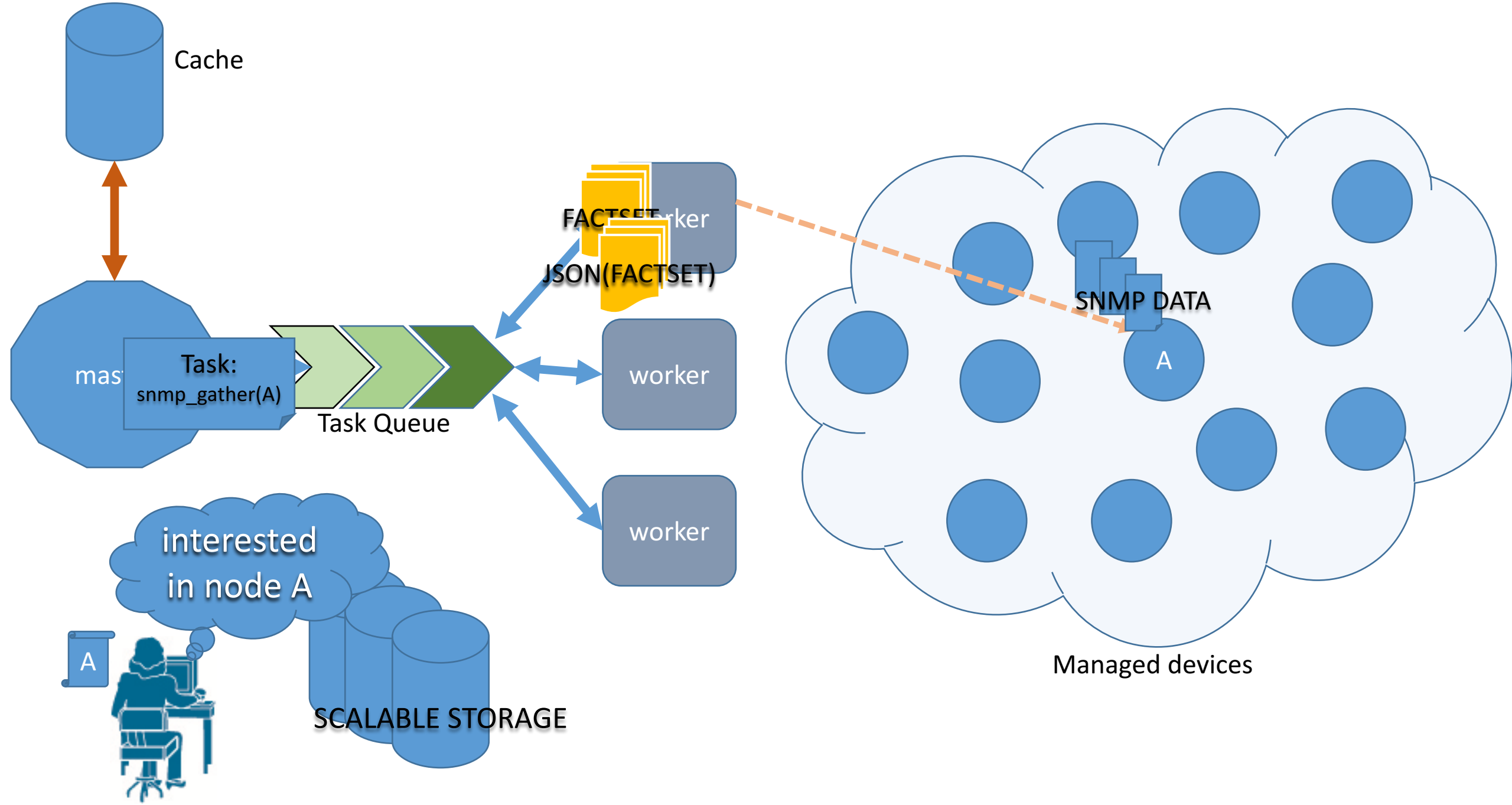
- Receiving of instructions from human operators
- Assignment of Jobs to Workers via the Task Queue
- Receiving of Facts from Workers
- Usage of Facts for inference
- **No communication with managed devices**
  - Delegated to Workers completely
- **Minimized specific management protocol awareness**
  - **No SNMP semantics in Master**
  - Dealing only with Facts
- Minimized internal memory storage
- All internal state in Cache
  - Restartable without significant loss
- Asynchronous event loop, no blocking operations

4. Cache

- Used by the Master (*presently*) for ephemeral fact storage
  - Unrelated to Scalable Storage Cluster

- Simple key-value data store
- Accessible through simplified API
- Expiration of keys
- Atomic operations

- Storage of all internal Master state
  - Master can restart with no loss of information


Cache
mas
Task:
snmp_gather(A)
Task Queue
JSON(FACTSET)
worker
worker
SNMP DATA
A
Managed devices
interested
in node A
A
SCALABLE STORAGE

Architecture Advantages

- Asynchronous architecture design
  - Language-agnostic Communication between Master and Workers
  - Different implementations, functions
- Scalable
  - Network transparent
  - Direct path from Workers to Scalable Storage Cluster
    - (many Workers, many Storage nodes)

Implementation

- Worker:
  - Perl 5
  - Moose Meta Object Framework
- Master:
  - Perl 5
  - Anyevent Asynchronous Event Loop library
  - Experimental Client in JS
- Cache: **Redis** Data Structure Server
- Task Queue: **Gearman** Job Queue
- Scalable Data Store: **ElasticSearch** Cluster

Omnidisco Master

- **Gearman** Client
- Event loop, no blocking points

- Rudimentary Fact processing and inference
  - Topology discovery
  - Performance analysis
- CLI interface
- Cache currently in use: **Redis**

Omnidisco Worker

- **Multiprocess Gearman** worker
- Usage of the SNMP::Class library
- Internal organization: Multiple Personalities producing Facts
  - Combination of all personality-produced Facts → FactSet

- Production of serialized FactSet
- Pushing of Serialized Facts → ElasticSearch Cluster

SNMP::Class

- **SNMP** instrumentation
- Minimal SNMP knowledge required for usage
- Object Oriented interface
  - Encapsulation of OIDs, Varbinds, Sessions, etc.
  - ResultSet Role with method chaining filter query API
    - Alleviation of SNMP and SMI MIB complexity
- Exceptions
- Role hierarchy for Varbinds
  - Example: IPv4Address Varbind value type

SNMP::Class example

```
my $s = SNMP::Class->new( ‘router1.mydomain’ );
$s->add('interfaces’,'ip’);

# ORM-like method chaining, find IP Address of interface ”Vlan1”
say $s->filter_label_under('ipAddrTable')
      ->find(
            ipAdEntIfIndex
            =>
            $s->ifDescr
              ->find(
                  ifDescr
                  =>
                  'Vlan1’
            )
            ->get_instance_oid
            ->to_number
      )
      ->ipAdEntAddr
      ->value;
```

SNMP::Class::Role::Personality

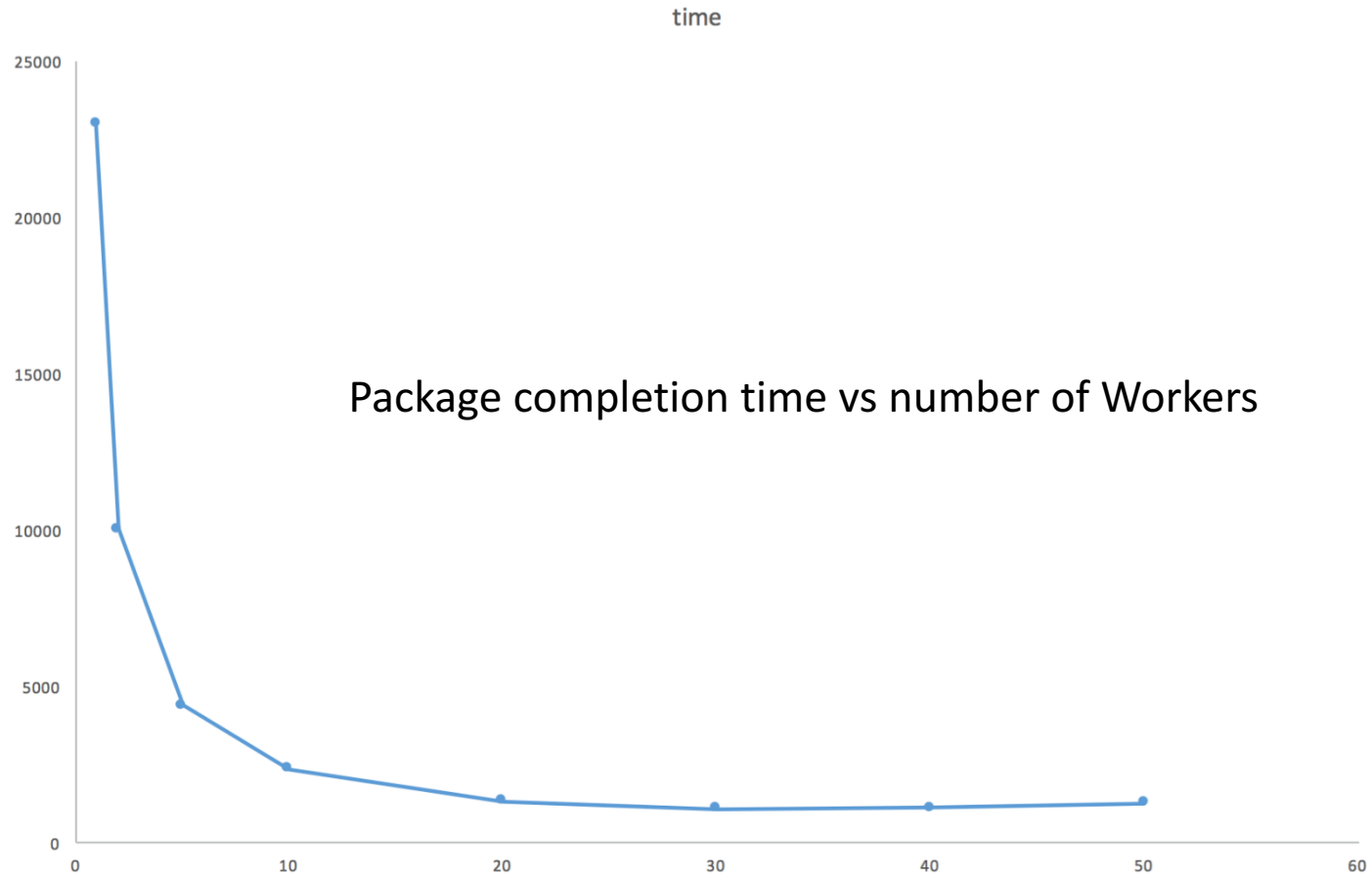
- Concept: A **Trait or Role** fulfilled by a managed device
  - E.g. an IPv4 Router, a Dot1d bridge, a device with interfaces etc.
- Personality: Translation from SNMP to Facts for a specific **Trait**
- SNMP MIBs grouping / aligning not suitable:
  - Chaos: Vendors implement MIBs partially
  - Too heavy: Many heterogeneous objects inside a single MIB
  - **MIBs ⇔ Schema** whereas **Personalities ⇔ Implementation**
- Personality: Fulfillment of predicates
  - Other Personalities:
    - Dependence tree of Personalities
  - Existence of a set of specific SNMP objects (possibly from different MIBs)
  - SNMP Objects with specific values
- Implementation as Moose Roles
  - Dynamically applicable to an SNMP::Class session object at runtime
    - New methods appear in object as soon as Personality is detected and primed.
- Personality: A Producer of Facts

Scalability

- Scalable Worker cluster
- Minimal memory requirements for Master
- Handling of large data volumes by Cache (Redis)
- Direct **Worker → ElasticSearch** path
  - no significant barrier to scalability

- Measurement Scenario:
  - Full cycle: Gathering SNMP data from 300+ devices

Performance measurements



Package completion time vs number of Workers

Future Work directions

- Netconf, CIM/WBEM, other Workers
- Removal of Gearman, transfer of Task Queue to Redis
  - Entirely doable
- Multiple simultaneous Masters
  - Almost possible already
- Elimination of Master(s)
  - Communication of Workers with Redis Cache directly
  - Need of direct client for Job submission
- Docker Image for Worker

Thank you very much!

Questions?

Ideas?