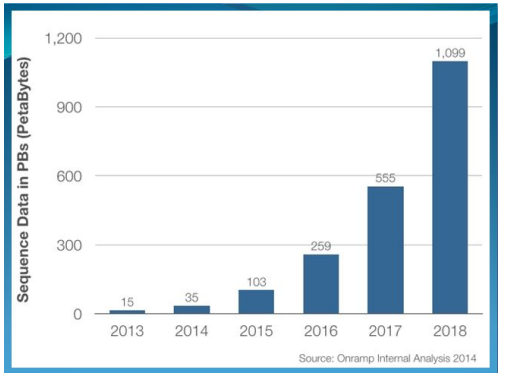
Responsive Storage: Home Automation for Research Data Management

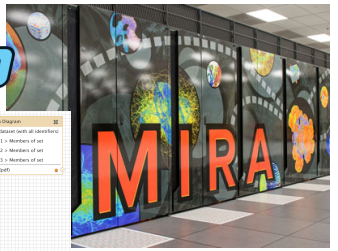
Ryan Chard

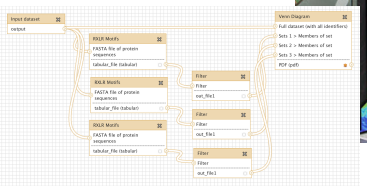
Postdoc Fellow, Argonne National Laboratory

The Problem

- Data generation rates are exploding
- Complex analytics processes
- The data lifecycle often involves multiple organisations, machines, and people







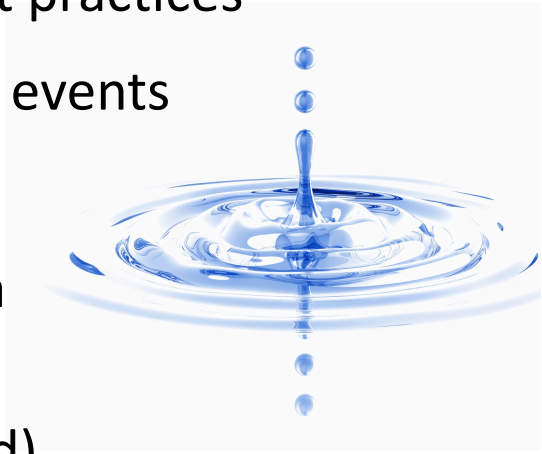
This creates a significant strain on researchers

- ➢Best management practises (cataloguing, sharing, purging, etc.) can be overlooked
- ➢Useful data may be lost, siloed, and forgotten

RIPPLE: A prototype responsive storage solution

Transform static data graveyards into active, responsive storage devices

- Automate data management processes and enforce best practices
- Event-driven: actions are performed in response to data events
- Users define simple if-trigger-then-action recipes
- Combine recipes into flows that control end-to-end data transformations
- Passively waits for filesystem events (very little overhead)
- Filesystem agnostic – works on both edge and leadership platforms



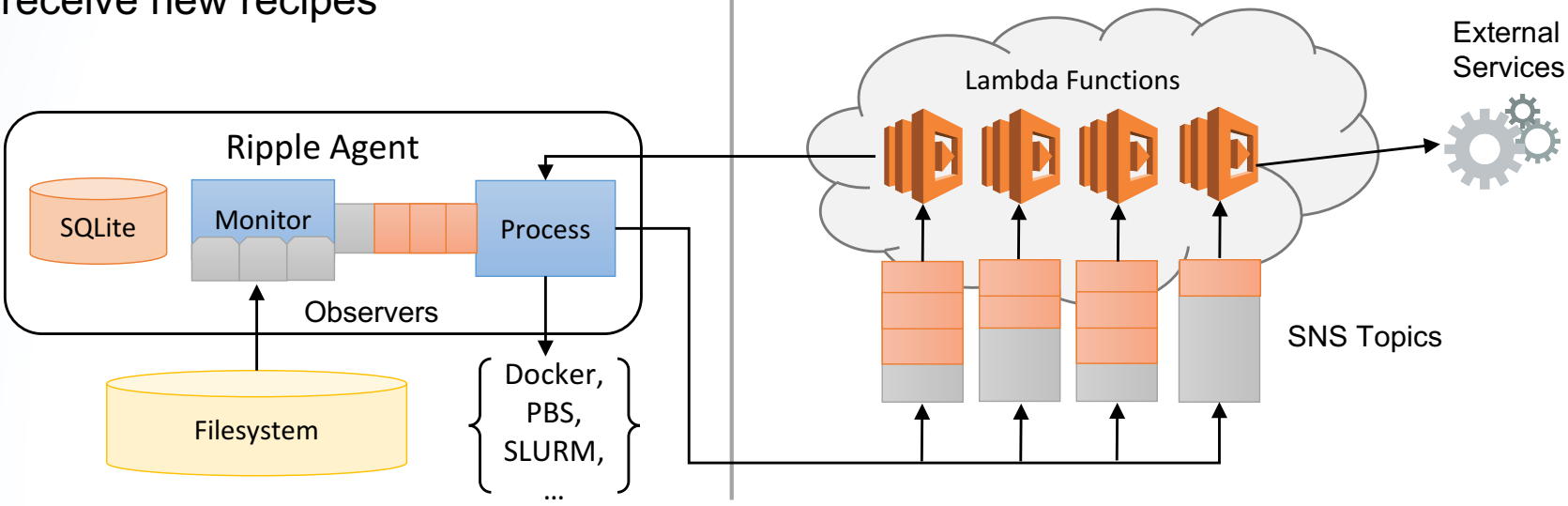
RIPPLE Architecture

Agent:

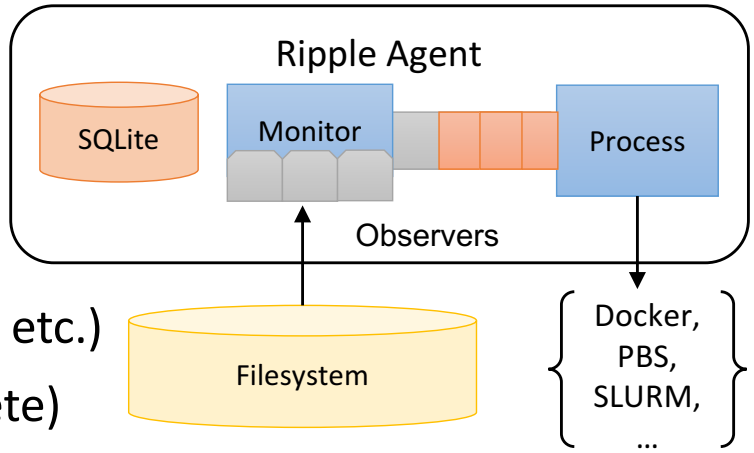
- Sits locally on the machine
- Detects & filters filesystem events
- Facilitates execution of actions
- Can receive new recipes

Service:

- Serverless architecture
- Lambda functions process events
- Orchestrates execution of actions



RIPPLE Agent



Python Watchdog observers listen for events

- inotify, polling, for filesystem events (create, delete, etc.)
- Globus Transfer API for events (transfer, create, delete)

Recipes are stored locally in a SQLite database

Local and cloud-based actions

- Docker containers and subprocesses act on local files (metadata extraction, dispatch jobs, etc.)
- AWS Lambda performs other tasks (Globus transfers, create shared endpoints, send emails, invoke other Lambda functions etc.)

RIPPLE Recipes

IFTTT-inspired programming model:

Triggers describe where the event is coming from (filesystem create events) and the conditions to match (/path/to/monitor/.*.h5)

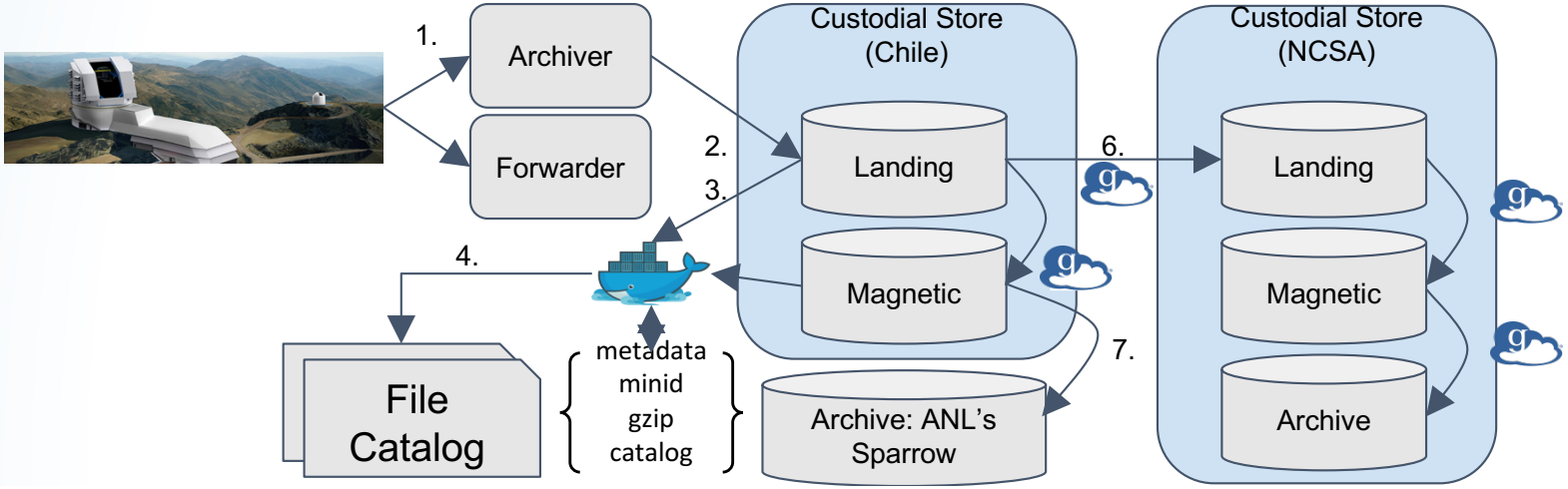
Actions describe what service to use (e.g., globus transfer) and arguments for processing (source/dest endpoints).

```
"recipe":{
  "trigger": {
    "username": "ryan",
    "monitor": "filesystem",
    "event": "FileCreatedEvent",
    "directory": "/path/to/monitor/",
    "filename": ".*.h5$"
  },
  "action": {
    "service": "globus",
    "type": "transfer"
    "source_ep": "endpoint1",
    "dest_ep": "endpoint2",
    "target_name": "$filename",
    "target_match": "",
    "target_replace": "",
    "target_path": "/~/$filename.h5",
    "task": "",
    "access_token": "<access token>"
  }
}
```

Scenario: Large Synoptic Survey Telescope

Developed a representative testbed of the LSST storage requirements

- Automatically propagate data between storage tiers and facilities
- Invoke Docker containers to extract metadata and maintain a file catalog
- Compress and archive files
- Recover deleted/corrupted files when delete and modification events occur



Scenario: Advanced Light Source

Deployed Ripple on an ALS and NERSC machine to automate data analysis

- **At ALS:** Detect new heartbeat beamline data and initiate transfer to NERSC
- **At NERSC:** Extract metadata, create sbatch file, dispatch analysis job to Edison queue, detect result and transfer back to ALS
- **At ALS:** create a shared endpoint, notify collaborators of result via email