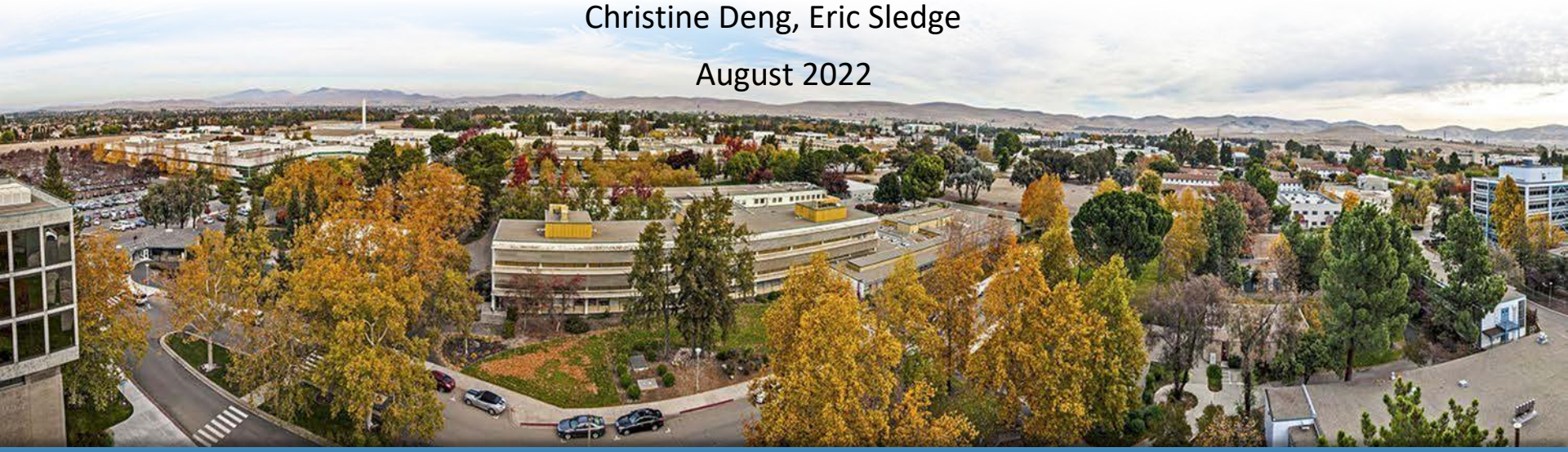
Installation of Flux

2022 HPC Academy Project

Christine Deng, Eric Sledge

August 2022



LLNL-PRES-838831

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344. Lawrence Livermore National Security, LLC

Lawrence Livermore National Laboratory

Table of Contents

The Flux Project Interns

Eric Sledge

Claflin University Orangeburg, SC

Computer Engineering

Christine Deng

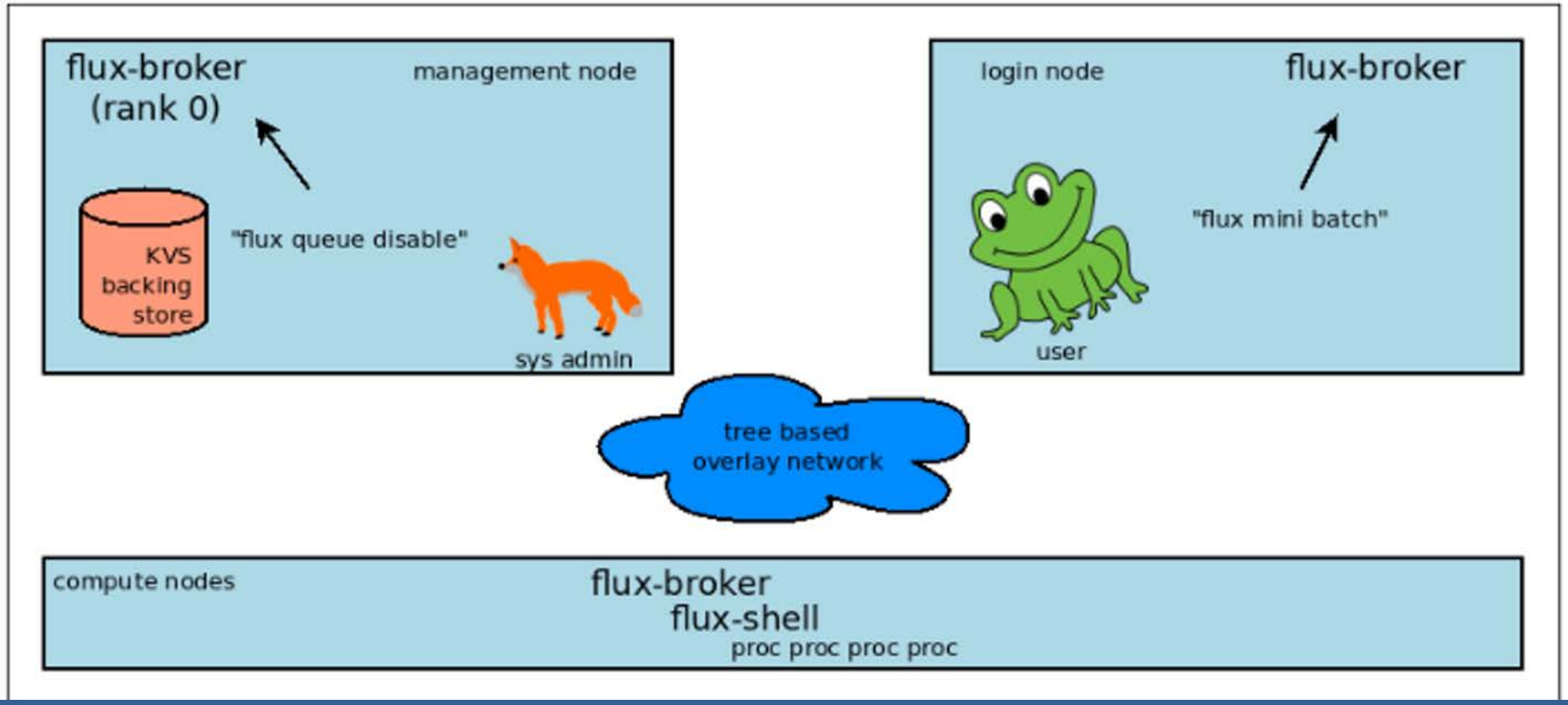
CSU East Bay

Computer Science

What is Flux?

- Resource management framework
  - Job scheduling algorithm
  - Allocation policy
- Designed for better throughput, job coordination/communication, portability

What is Flux?



https://flux-framework.readthedocs.io/en/latest/adminguide.html

Project Objective

- Install Flux on our cluster
  - flux-core, flux-security, flux-sched
- Run MPI jobs

Running jobs

```c
#include <mpi.h>
#include <stdio.h>

int main(int argc, char** argv) {
    // Initialize the MPI environment
    MPI_Init(NULL, NULL);

    // Get the number of processes
    int world_size;
    MPI_Comm_size(MPI_COMM_WORLD, &world_size);

    // Get the rank of the process
    int world_rank;
    MPI_Comm_rank(MPI_COMM_WORLD, &world_rank);

    // Get the name of the processor
    char processor_name[MPI_MAX_PROCESSOR_NAME];
    int name_len;
    MPI_Get_processor_name(processor_name, &name_len);

    // Print off a hello world message
    printf("Hello world from processor %s, rank %d"
           " out of %d processors\n",
           processor_name, world_rank, world_size);

    // Finalize the MPI environment.
    MPI_Finalize();
}
```

```
[flux@siliconi ~]$ flux mini run -n4 -N4 hostname
siliconi
silicon4
silicon3
silicon2
```

```
[flux@siliconi ~]$ cat flux-run-hello.sh
#!/bin/bash

export OMPI_MCA_btl="tcp,self"
flux mini run -n 4 /var/flux/hello 2>/dev/null | grep -v Unable
```

```
[flux@siliconi ~]$ flux mini batch -n4 -N4 flux-run-hello.sh
ƒ3B23bEkdKM
[flux@siliconi ~]$ flux jobs
       JOBID USER     NAME       ST NTASKS NNODES  RUNTIME NODELIST
 ƒ3B23bEkdKM flux     flux-run-h  R      4      4   2.811s silicon[2-5]
[flux@siliconi ~]$ cat flux-ƒ3B23bEkdKM.out
Hello world from processor silicon2, rank 0 out of 4 processors
Hello world from processor silicon4, rank 2 out of 4 processors
Hello world from processor silicon3, rank 1 out of 4 processors
Hello world from processor silicon5, rank 3 out of 4 processors
```

Challenges

- Limited Web Resources and Documentation
  - Trial and error, and digging around a lot
- Precise Command Line Usage
  - Expert-Friendly, tough if you are used to GUI's, once you know certain commands it gets easier
- Minor edits to configuration files
- Enabling Certain Repos

Future Work and High End Goals

- Explore using the flux API to manage jobs
- Setup the exclude for the mgmt node - normally we would not run real end-user work on the mgmt node
- Investigate the Prolog/epilog

Tools Used

References

https://flux-framework.readthedocs.io/en/latest/adminguide.html
https://flux-framework.readthedocs.io/projects/flux-rfc/en/latest/spec_18.html#language
https://computing.llnl.gov/projects/flux-building-framework-resource-management
https://sc18.supercomputing.org/proceedings/workshops/workshop_files/ws_works115s2-file1.pdf
https://flux-framework.readthedocs.io/en/latest/quickstart.html#spack-recommended-for-curious-users

Questions?

Installing & Configuring Lustre on KVMs

Naomi Cheeves, Gabe Maxfield

August 2, 2022

LLNL-PRES-838831

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344. Lawrence Livermore National Security, LLC

Lawrence Livermore National Laboratory

Team Members

- Naomi
  - Senior at University of California, Davis
  - Computer Science
  - Graduating December 2022

- Gabe Maxfield
  - Sophomore at Brigham Young University
  - Computer Science
  - Graduating 2025

Goals

Our mission was to find a scalable filesystem for our HPC environment.

We decided to create a small, proof of concept cluster on some kernel-based virtual machines.

Our goals were as follows:

- Investigate Lustre
- Create a miniature lustre cluster utilizing three kvms
- Investigate and install a backend filesystem for lustre (zfs vs ldiskfs).
- Run some benchmark tests

Lustre

- Why use Lustre in HPC?

  Lustre has…

  — Exascale Capacities
    - Lustre uses distributed, object-based storage managed by servers.
      - Very large files can be distributed amongst different data objects and amongst several servers.

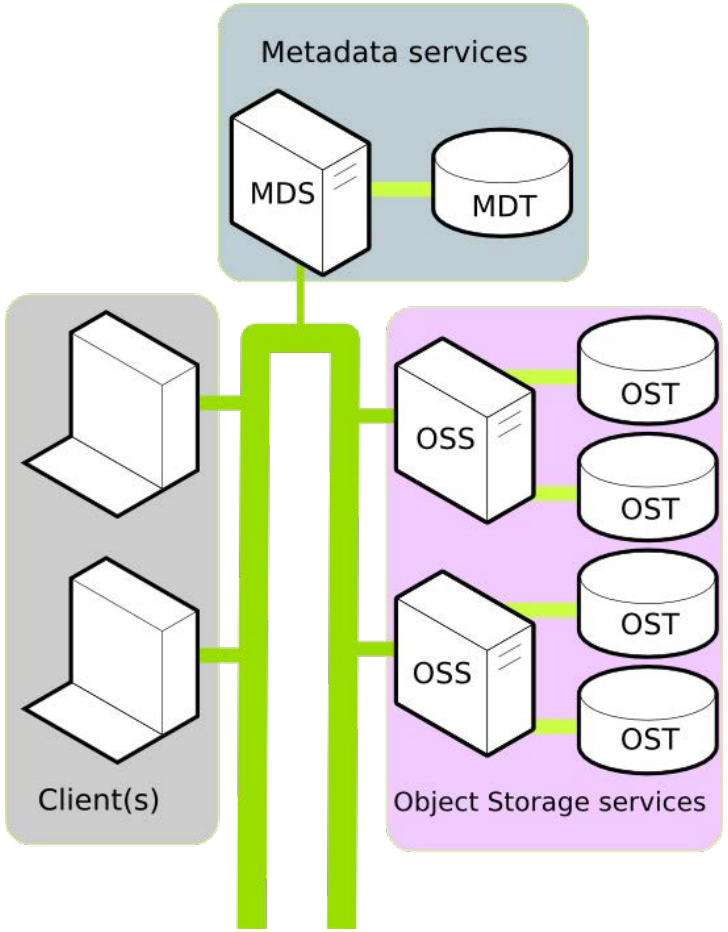
  — Data Integrity (only ZFS)
    - Data is written frequently in the event of a node crashing

  — Massive Scalability
    - Object Storage is low cost and efficient
    - I/O thoroughput and capacity are easily scaled by dynamically adding servers

  — POSIX-compliant on a Linux-based OS

The Lustre Architecture

- Components of a Cluster
  - MGS: Management Server, stores configuration information.
  - MDS: Metadata Server, manages MDTs.
  - MDT: Metadata target that stores file information/ location.
  - OSS: Object Storage Server, manages OSTs.
  - OST: Object Storage Target, storage device, hosts files.
  - Client(s): Access and use the data.



Why ZFS over LDISKFS?

- Larger Capacities
- Integrated Data Integrity

Feature	LDISKFS	ZFS
Max Volume Size	32PB	512PB
Maximum Lustre File Size	512PB	8EB
Native Data Protection	None	Mirror, RAIDZ{1,2,3}, DRAID (Future)
Detect/ repair silent data corruption	None	Yes
File system repair	Offline: FSCK	Online: ZFS Scrub

What we accomplished

In the end, our setup consisted of:

- 3 Centos 7 Kernel-based virtual machines
- A single MGS/MDS/MDT and OSS/OST setup with one client
- A small Lustre cluster built from source utilizing a ZFS backend filesystem

- What was not accomplished:

- Benchmark tests
  - Lustre utilizing local storage
  - Lustre utilizing JBODs

Challenges

- Outdated resources
  - Tutorials written for an older RHEL version
  - More niche technology with few public discussions
  - Poor documentation (Except Lustre manual)
    - Documentation used old methods and syntax
- Installing Lustre
  - Lustre packages hard to find/ Don’t support ZFS
  - Setting up the development environment
    - LNET
    - Mounting Lustre Devices

Possible Future Goals

- High Availability
  - Utilize pacemaker to maintain high availability in the event of server failure
- Explore Hierarchical Storage Management (HSM)
  - Integrate Cheap Long-term Storage Solutions
  - Automatically move old data to/ from a cheaper/ slower storage medium
  - Invisible to end client
- Explore Clustered Trivial Database (CTDB)
  - Interface for the Server Message Block protocol (Windows protocol)
  - Extend filesystem support for different clients and operating systems

Sources

- https://wiki.lustre.org/
- https://wiki.whamcloud.com/
- https://en.wikipedia.org/wiki/Lustre_(file_system)
- https://wiki.lustre.org/images/6/64/LustreArchitecture-v4.pdf
- https://www.netapp.com/data-storage/storagegrid/what-is-object-storage/

RabbitMQ and Kafka

Prabhleen Bagri and Lindsey Amaro
Mentor: David Fox

August 11, 2022

LLNL-PRES-838831

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344. Lawrence Livermore National Security, LLC

Lawrence Livermore National Laboratory

Team Members

- Prabhleen Bagri
- San Jose State University
- Computer Science
- Expected Grad: May 2023

- Lindsey Amaro
- Cal Poly – San Luis Obispo
- Mathematics
- Expected Grad: June 2024

What is a Message Broker?

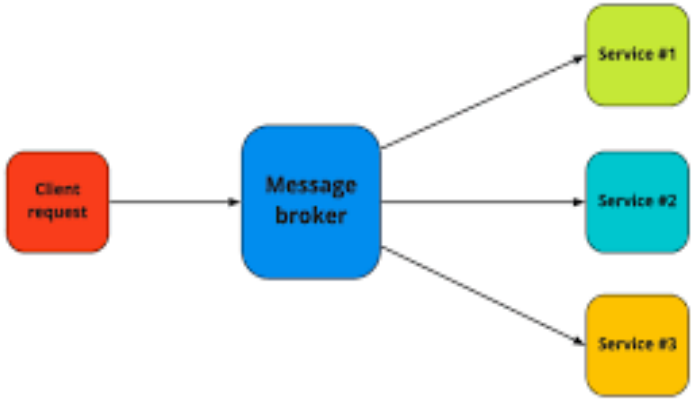


Diagram source: https://tsh.io/blog/message-broker/

Normally we would use TCP to send messages, but there are drawbacks

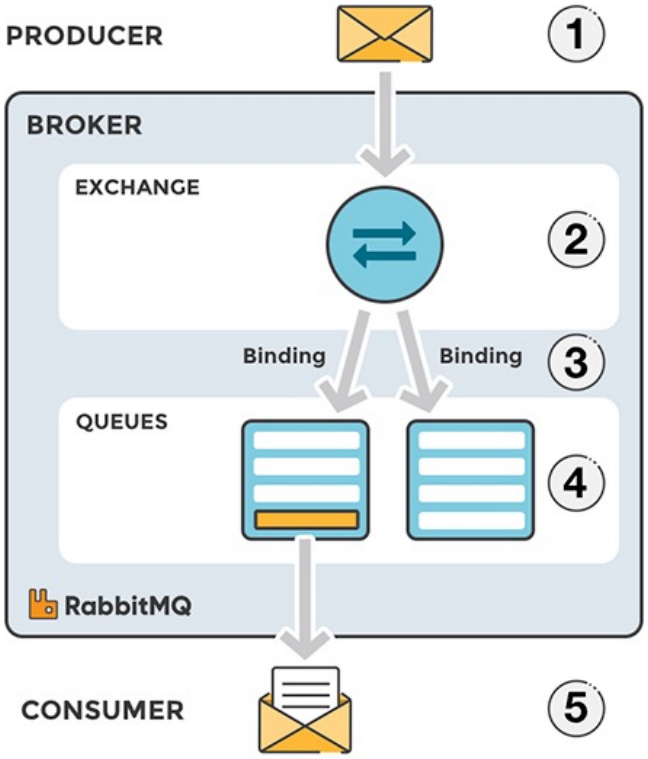
→ **SOLUTION : message brokers**

Message brokers sit in between two services that interact. They enable applications and systems to communicate with each other and exchange information.

- Acts as a buffer i.e. it holds messages until it is ready to be received
- Allows sender to issue message without knowing where receiver is
- Improved system performance because it allows asynchronous processing
- facilitates decoupling

RabbitMQ

- Message Broker
- Producer sends messages to exchanges, which then route those messages to queues that consumers can access
- Several Types of Exchanges support different routing patterns:
  - Direct
  - Fanout
  - Topic
- Messages deleted from queues once consumed
- Use Cases:
  - Situations that require complex sending patterns
  - Instances where quick message response is important
  - Applications that need to work with messaging protocols such as AMQP
- Companies that use RabbitMQ
  - T-Mobile
  - Reddit
  - Trivago



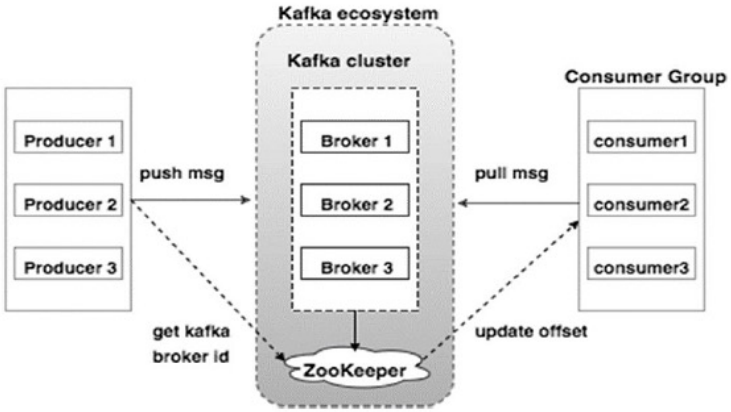
Picture Source: https://www.cloudamqp.com/blog/part1-rabbitmq-for-beginners-what-is-rabbitmq.html

Kafka

- **A distributed event store and streaming platform**
- Has servers and clients that communicate over TCP Network Protocol
- Can be deployed on hardware, VMs, containers, cloud environments
- Used by over 80% of the fortune 100

Use cases:

- Decouple dependencies by streaming events
- Messaging
- Location & activity tracking
- Commit log
- Log aggregation



- Server: is run as a cluster of servers that can span multiple datacenters
- Highly scalable and fault-tolerant
- Client: allows you to write distributed applications and microservers that read, write, and process streams of data in parallel

diagram source: https://www.tutorialspoint.com/apache_kafka/apache_kafka_cluster_architecture.htm

Objectives

- Install and configure RabbitMQ and Kafka in student cluster environment
- Create basic sender and receiver Python applications for RabbitMQ and Kafka to ensure each service is working
- Explore HPC uses cases for both services

Image Source: https://kafka.apache.org/

Image Source: https://www.rabbitmq.com/

RabbitMQ Install and Configuration

RabbitMQ on RHEL Installation Steps:

1. Import necessary RPM's
2. Configure a Yum repository for RabbitMQ
3. Install the RabbitMQ server and its dependencies
4. Create a RabbitMQ user
5. Create a Virtual Host for the RabbitMQ user to connect to
6. Set permissions, so the user and virtual host can recognize each other

```
[root@xenon4 rabbit]# python3 send.py
 [x] Sent 'Hello World!'
```

```
[root@xenon5 rabbit]# python3 receive.py
 [*] Waiting for messages. To exit, press CTRL+C
 [x] Received 'Hello World!'
```

Kafka Install and Configuration

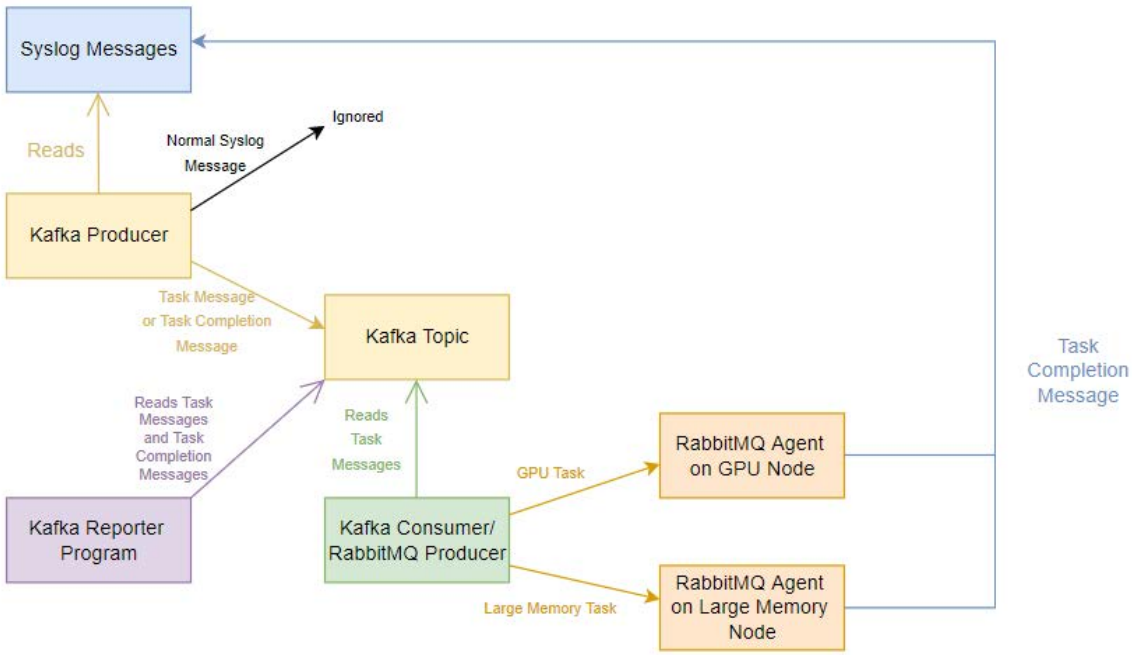
```
[root@radon4 ~]# python3 producerApp.py
sending message...
message sent successfully...
```

```
received message =  ConsumerRecord(topic='test-topic'
partition=0, offset=3, timestamp=1660178191347, times
mp_type=0, key=None, value={'name': 'abc', 'email': '
c@example.com'}, headers=[], checksum=None, serialize
key_size=-1, serialized_value_size=43, serialized_hea
r_size=-1)
```

- Download latest Kafka release and extract it
- Create user to run Kafka and Zookeeper
- Start Kafka environment (create and start a service for both)
- To send messages, create producer and consumer programs on separate nodes using KafkaProducer / KafkaConsumer APIs

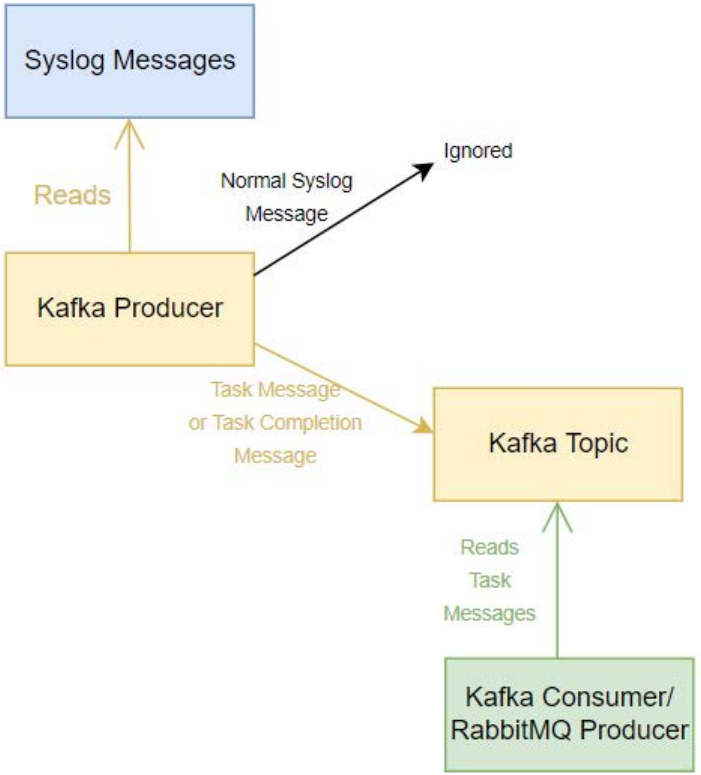
Example Use Case Model

- Besides sending messages, RabbitMQ and Kafka can also be used to send tasks
- These services could work together to send and record tasks being done
- **Objective**: Use Kafka and RabbitMQ to handle jobs issued by syslog

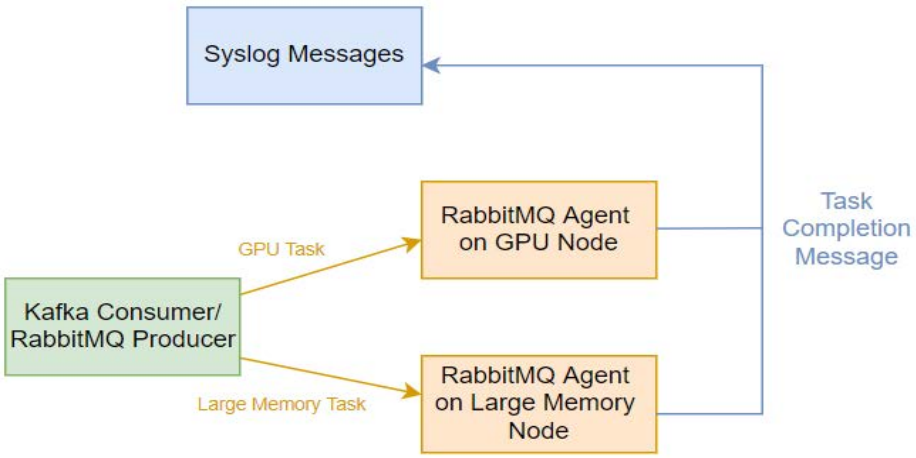


Transferring Task Messages

- Kafka Producer continually reads syslog messages
- Task message triggers Producer to publish to Kafka Topic
- Kafka Consumer reads from topic and passes task message to RabbitMQ Producer



Delegating and Recording Tasks



- RabbitMQ sends task to appropriate agent
- Task completion message sent to syslog
- Kafka Recorder program lists when tasks were issued and completed

Challenges

- Filtering Messages
  - No straightforward method for filtering data sent into the Kafka topic
- Incorporating Kafka in demo
  - Limited access to large data sets
  - Made finding use case difficult
- Connecting RabbitMQ and Kafka
  - Because of their similar functionality, finding a practical model using both services to fit one use case was a challenge

Future Goals

- Explore clustering
  - Scalability
  - High Availability
- Explore additional clients and usage models
  - Other client libraries with RabbitMQ and Kafka (C++, Java, etc.)
  - Advanced Configurations (integrating Celery with RabbitMQ, KafkaStreams with Kafka, etc.)
- Data Retention and Management, particularly with Kafka

References

- https://www.rabbitmq.com/
- https://www.cloudamqp.com/blog/part1-rabbitmq-for-beginners-what-is-rabbitmq.html
- https://www.upsolver.com/blog/kafka-versus-rabbitmq-architecture-performance-use-case#:~:text=Kafka%20offers%20much%20higher%20performance,for%20big%20data%20use%20cases
- https://kafka.apache.org/
- https://betterprogramming.pub/why-do-we-need-message-broker-7382ce0e46c6
- https://www.ibm.com/cloud/learn/message-brokers