pocket

Elastic Ephemeral Storage for Serverless Analytics

Ana Klimovic[*], Yawen Wang[*], Patrick Stuedi[+], Animesh Trivedi[+], Jonas Pfefferle[+], Christos Kozyrakis[*]

[*]Stanford University, [+]IBM Research

OSDI 2018

Serverless Computing

- Serverless computing enables users to launch short-lived tasks with high elasticity and fine-grain resource billing

Serverless Computing

- Serverless computing enables users to launch short-lived tasks with high elasticity and fine-grain resource billing
- Serverless computing is increasingly used for **interactive analytics**

PyWren (SoCC'17)

ExCamera (NSDI'17)

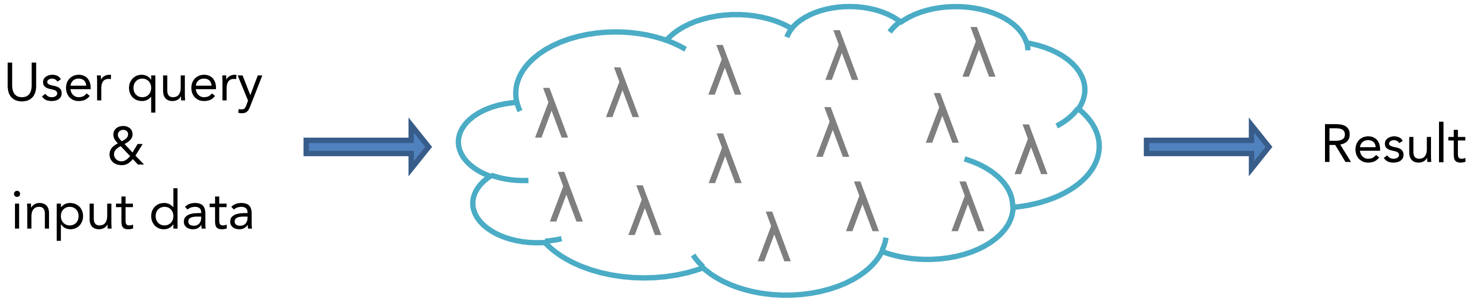
databricks serverless

gg: The Stanford Builder

Amazon Aurora Serverless

Serverless Computing

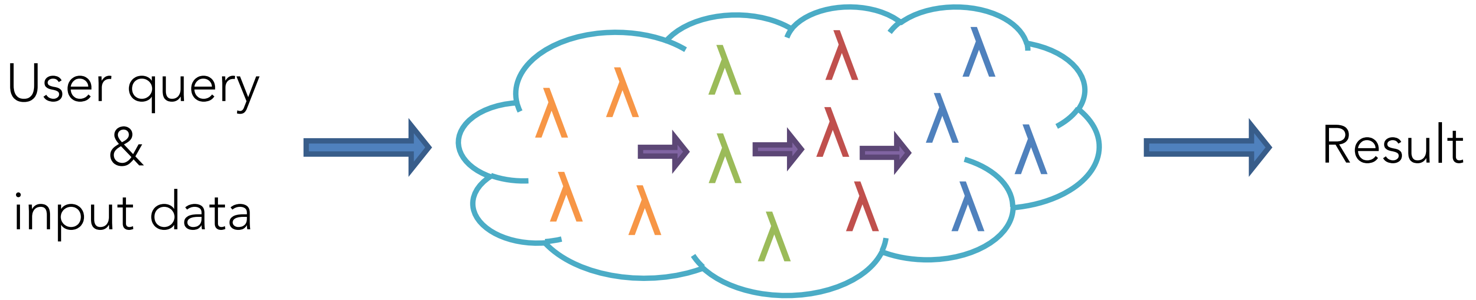
- Serverless computing enables users to launch short-lived tasks with high elasticity and fine-grain resource billing
- Serverless computing is increasingly used for **interactive analytics**
  - Exploit massive parallelism with large number of serverless tasks



The Challenge: Data Sharing

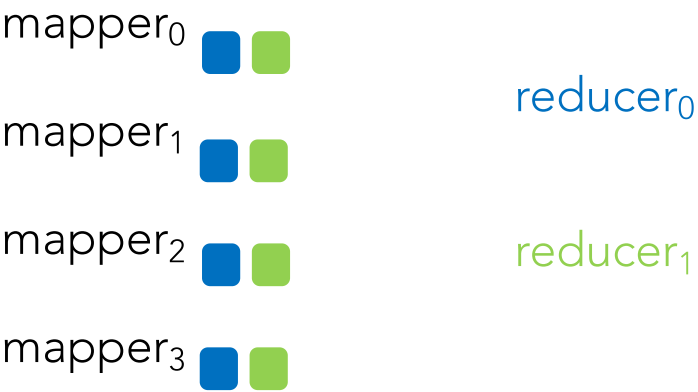
- Analytics jobs involve multiple stages of execution
- Serverless tasks need an efficient way to communicate **intermediate data** between different stages of execution

ephemeral data



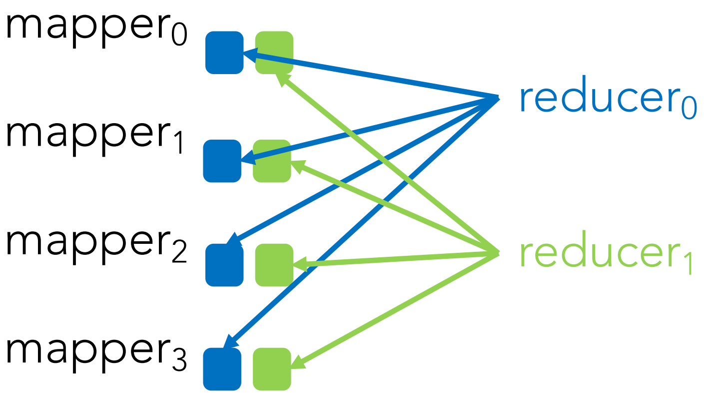
In traditional analytics…

- Ephemeral data is exchanged directly between tasks



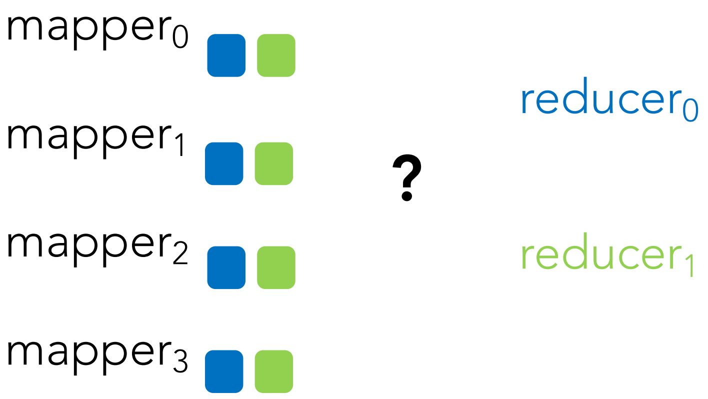
In traditional analytics…

- Ephemeral data is exchanged directly between tasks



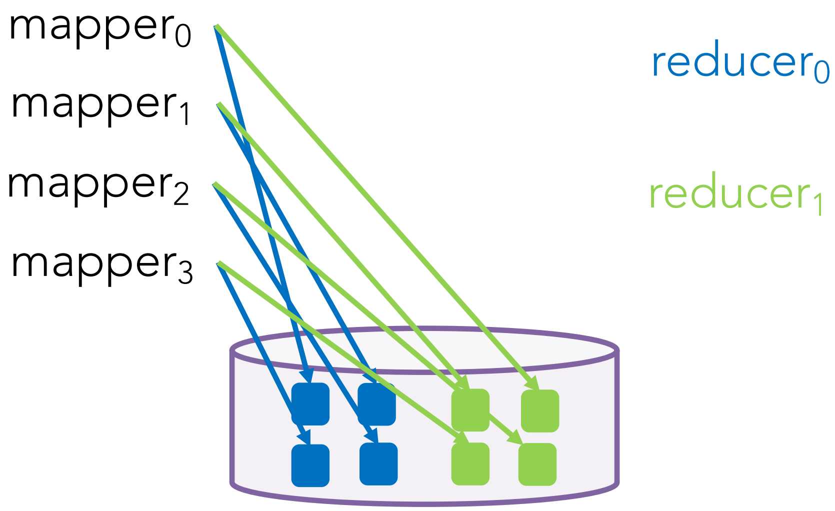
In serverless analytics…

- Direct communication between serverless tasks is difficult:
  - Tasks are short-lived and stateless



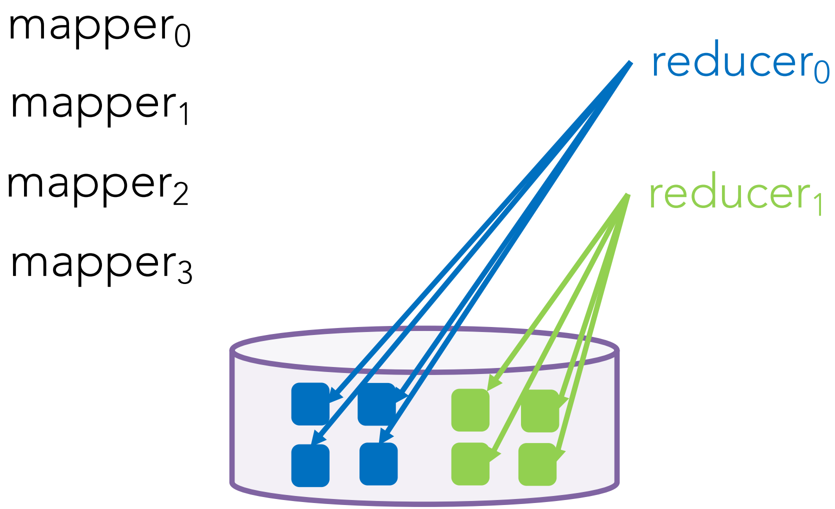
In serverless analytics…

- The natural approach for sharing ephemeral data is through a common data store



In serverless analytics…

- The natural approach for sharing ephemeral data is through a common data store



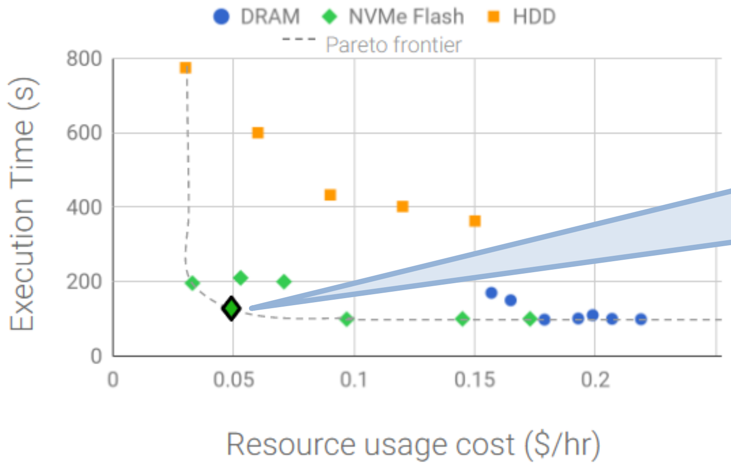
Requirements for Ephemeral Storage

1. High performance for a wide range of object sizes
2. Cost efficiency, i.e., fine-grain, pay-what-you-use resource billing

Understanding Ephemeral Storage for Serverless Analytics. Ana Klimovic, Yawen Wang, Christos Kozyrakis, Patrick Stuedi, Jonas Pfefferle, Animesh Trivedi. **ATC'18**, 2018.

Requirements for Ephemeral Storage

1. High performance for a wide range of object sizes
2. Cost efficiency, i.e., fine-grain, pay-what-you-use resource billing
   - Example of performance-cost tradeoff for a serverless video analytics job with different ephemeral data store configurations



Finding the Pareto optimal resource allocation is non-trivial…and gets harder with multiple jobs.

Requirements for Ephemeral Storage

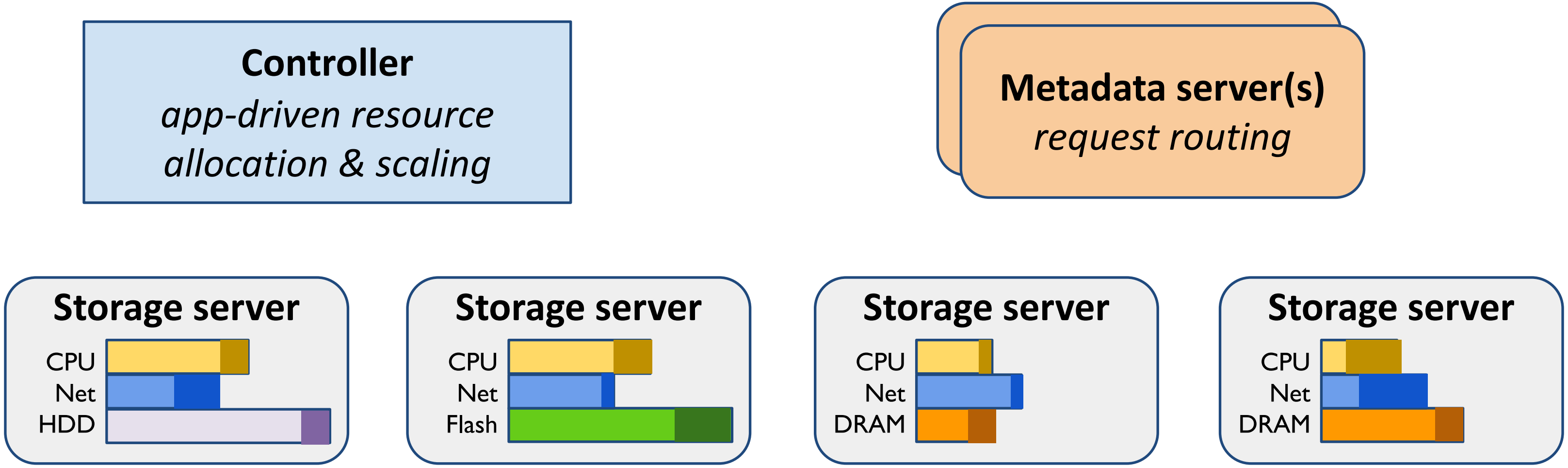
1. High performance for a wide range of object sizes
2. Cost efficiency, i.e., fine-grain, pay-what-you-use resource billing
3. ~~Fault-tolerance~~

Existing cloud storage systems do not meet the elasticity, performance and cost demands of serverless analytics jobs.

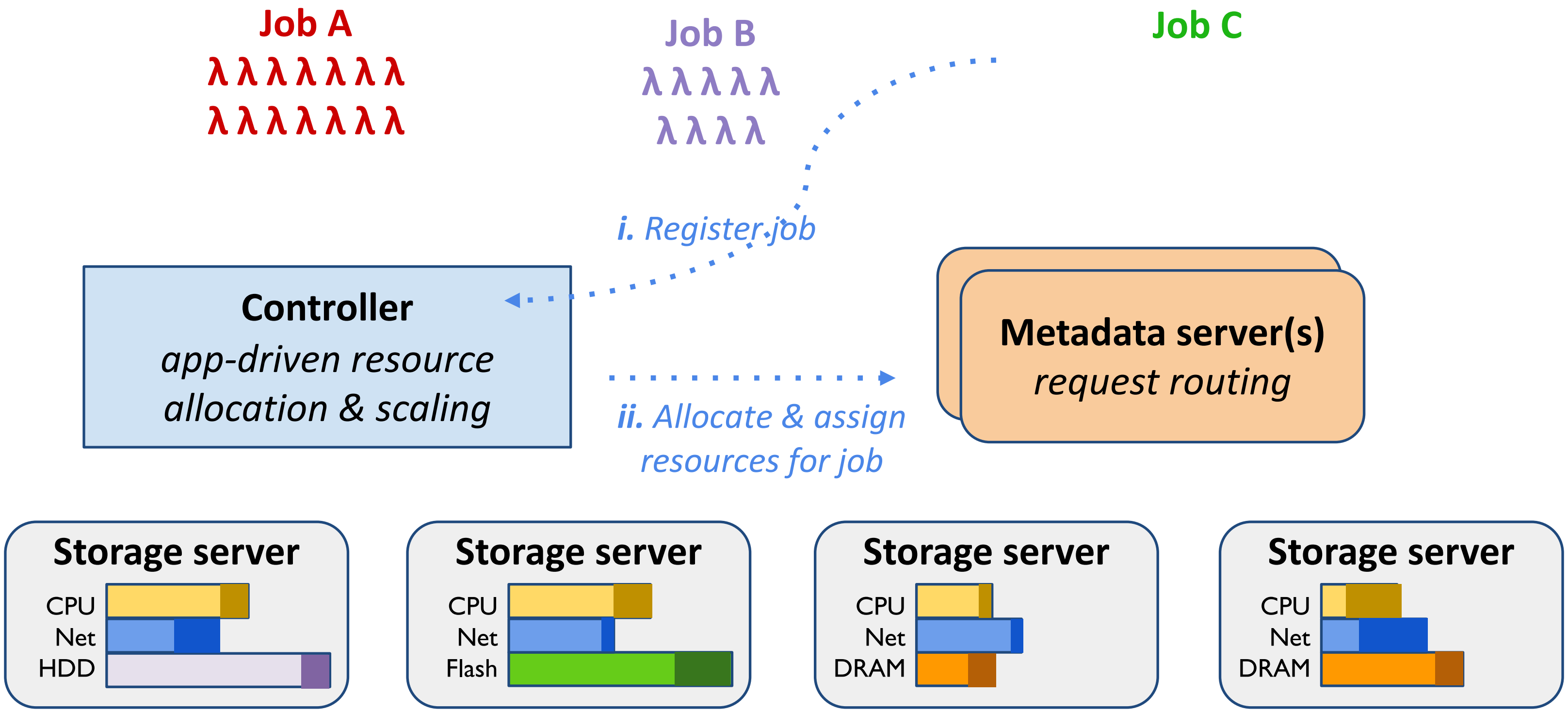
Pocket

- An elastic, distributed data store for ephemeral data sharing in serverless analytics
- Pocket achieves high performance and cost efficiency by:
  - Leveraging multiple storage technologies
  - Rightsizing resource allocations for applications
  - Autoscaling storage resources in the cluster based on usage
- Pocket achieves similar performance to Redis, an in-memory key value store, while saving ~60% in cost for various serverless analytics jobs

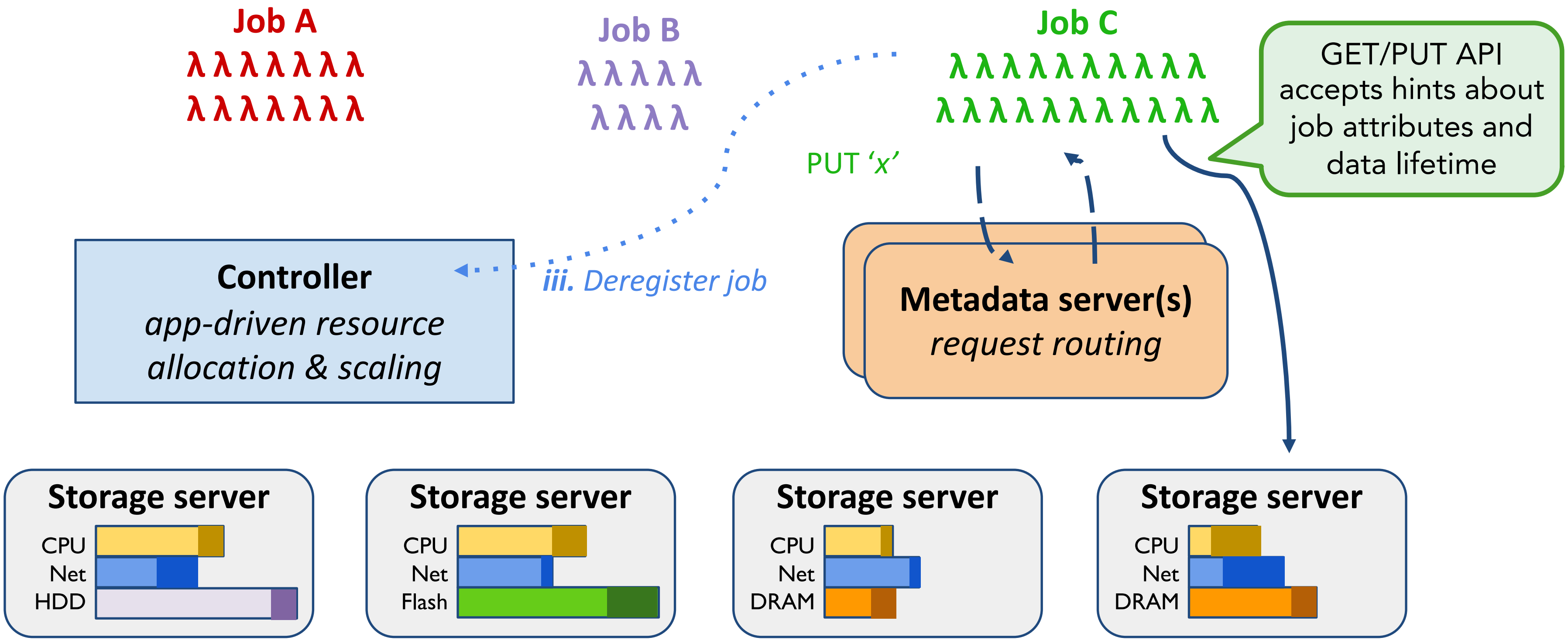
Pocket Design



Using Pocket

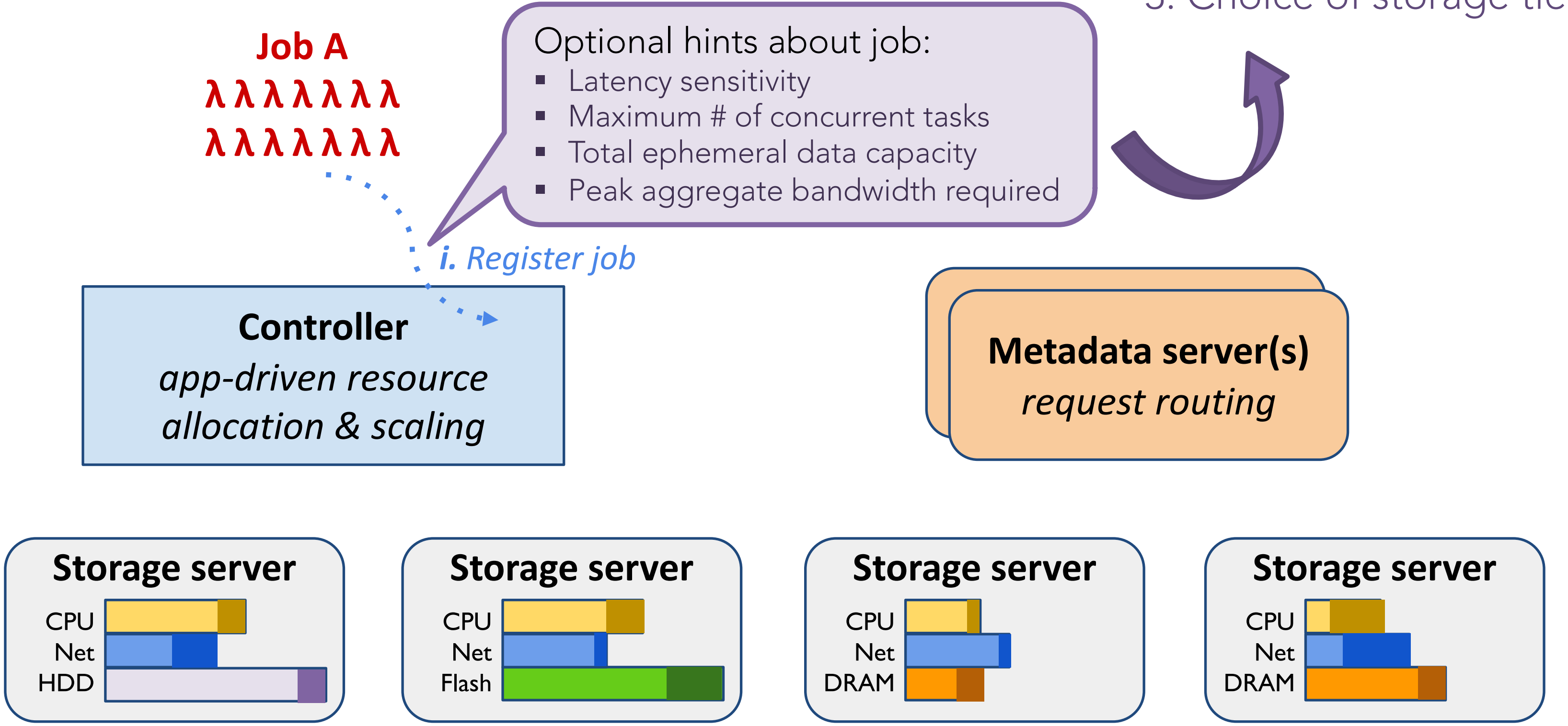


Using Pocket



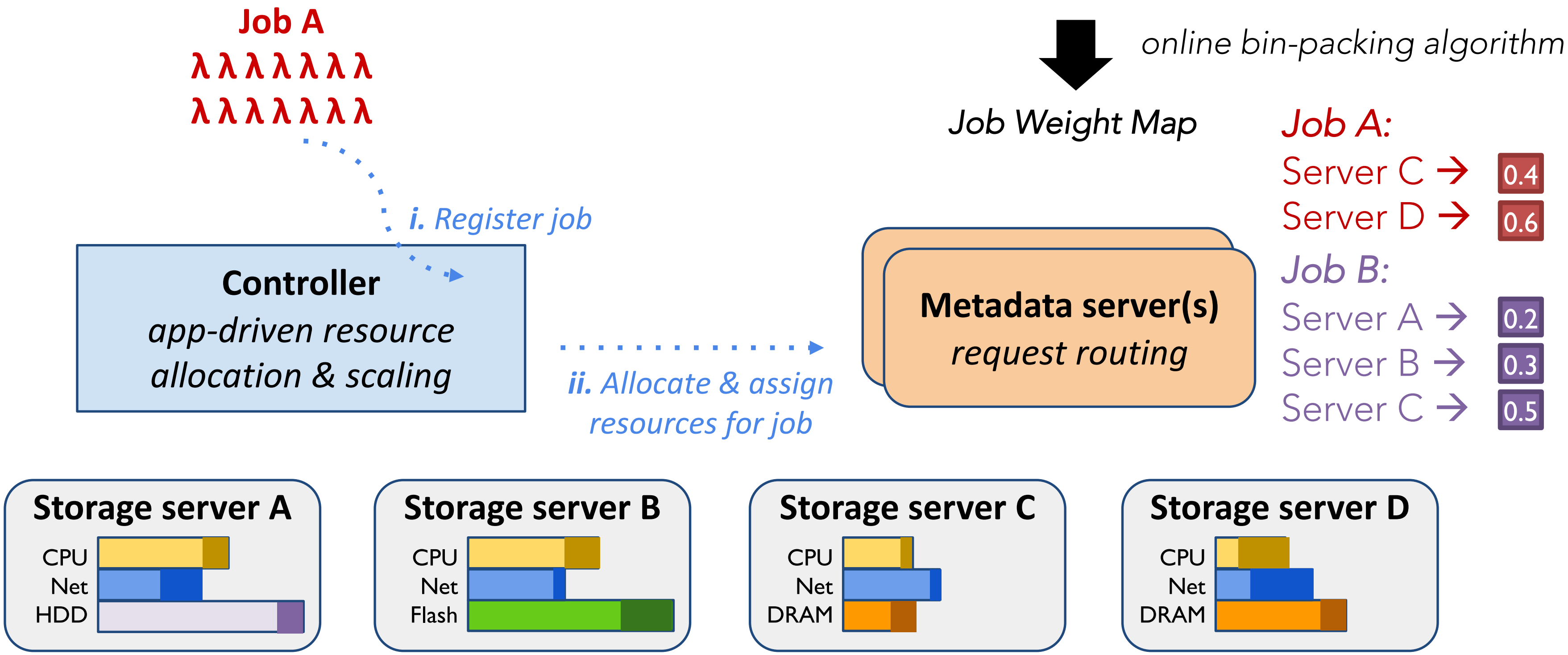
Assigning Resources to Jobs

1. Throughput allocation
2. Capacity allocation
3. Choice of storage tier(s)



Assigning Resources to Jobs

1. Throughput allocation
2. Capacity allocation
3. Choice of storage tier(s)



Autoscaling the Pocket Cluster

- Goal: scale cluster resources dynamically based on resource usage

- Mechanisms:
  - Monitor CPU, network bandwidth, and storage capacity utilization
  - Add/remove storage & metadata nodes to keep utilization within range
  - Steer data for incoming jobs to active nodes
  - Drain inactive nodes as jobs terminate

- Avoid migrating data

Implementation

- Pocket's metadata and storage server implementation is based on the **Apache Crail** distributed storage system [1]
- We use **ReFlex** for the Flash storage tier [2]
- Pocket runs the storage and metadata servers in containers, orchestrated using **Kubernetes** [3]

[1] Apache Crail (incubating). http://crail.apache.org/

[2] *ReFlex: Remote Flash == Local Flash*. Ana Klimovic, Heiner Litz, Christos Kozyrakis. **ASPLOS'17**, 2017.

[3] Kubernetes. https://kubernetes.io/

Pocket Evaluation

- We deploy Pocket on Amazon EC2

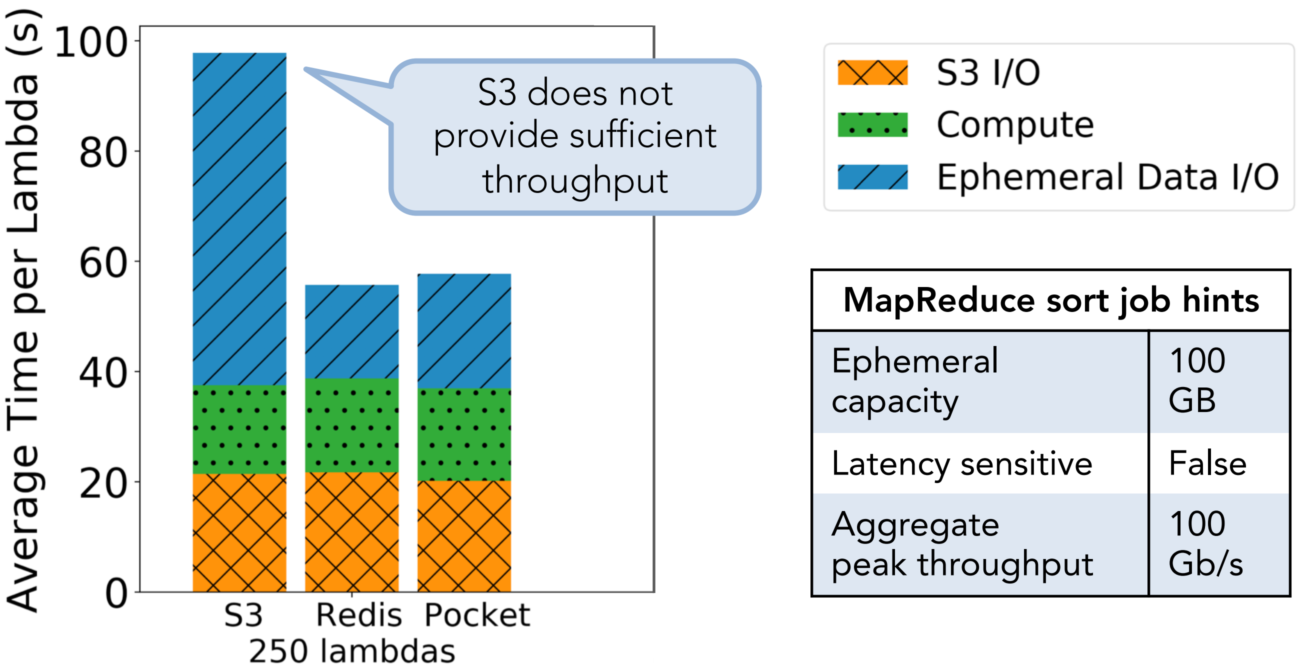
Controller	m5.xlarge
Metadata server	m5.xlarge
DRAM server	r4.2xlarge
NVMe Flash server	i3.2xlarge
SATA/SAS SSD server	i2.2xlarge
HDD server	h1.2xlarge

Amazon EC2

- We use AWS Lambda as our serverless platform
- **Applications**: MapReduce sort, video analytics, distributed compilation

Application Performance with Pocket

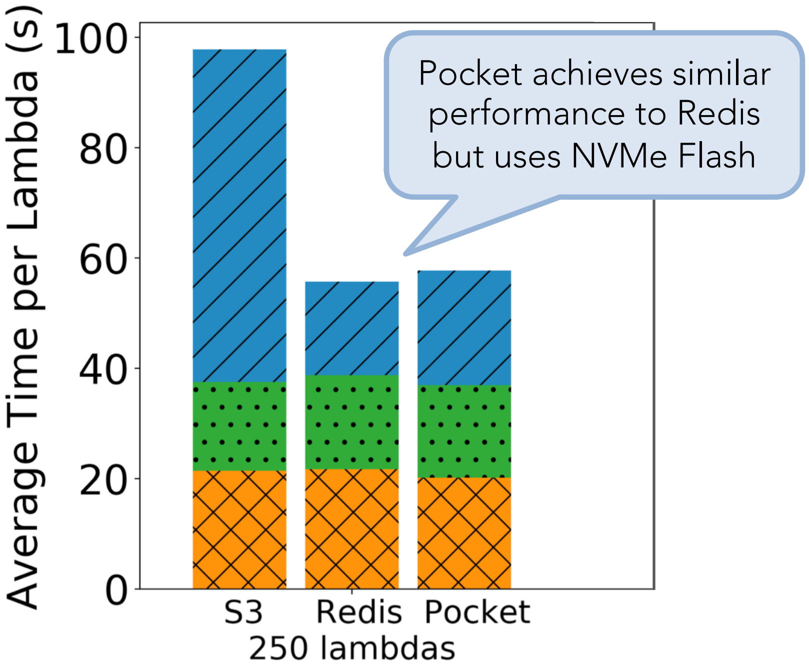
- Compare Pocket to S3 and Redis, which are commonly used today

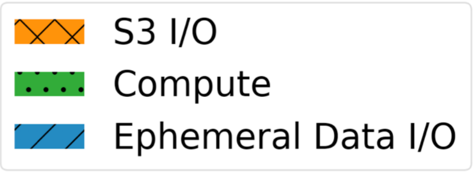


MapReduce sort job hints	
Ephemeral capacity	100 GB
Latency sensitive	False
Aggregate peak throughput	100 Gb/s

Application Performance with Pocket

- Compare Pocket to S3 and Redis, which are commonly used today

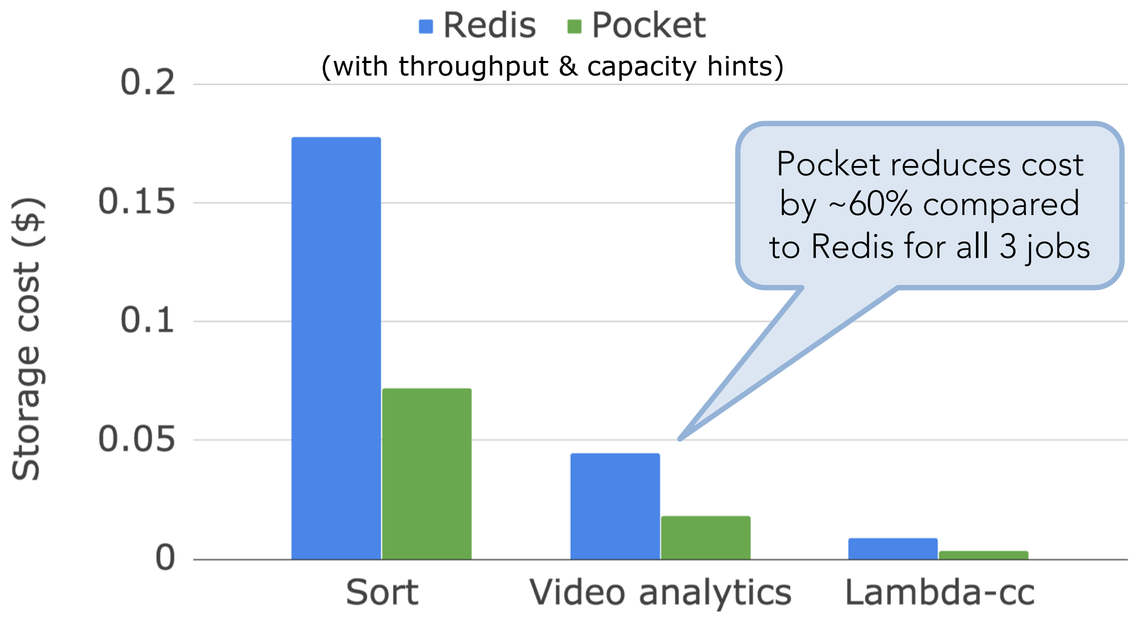




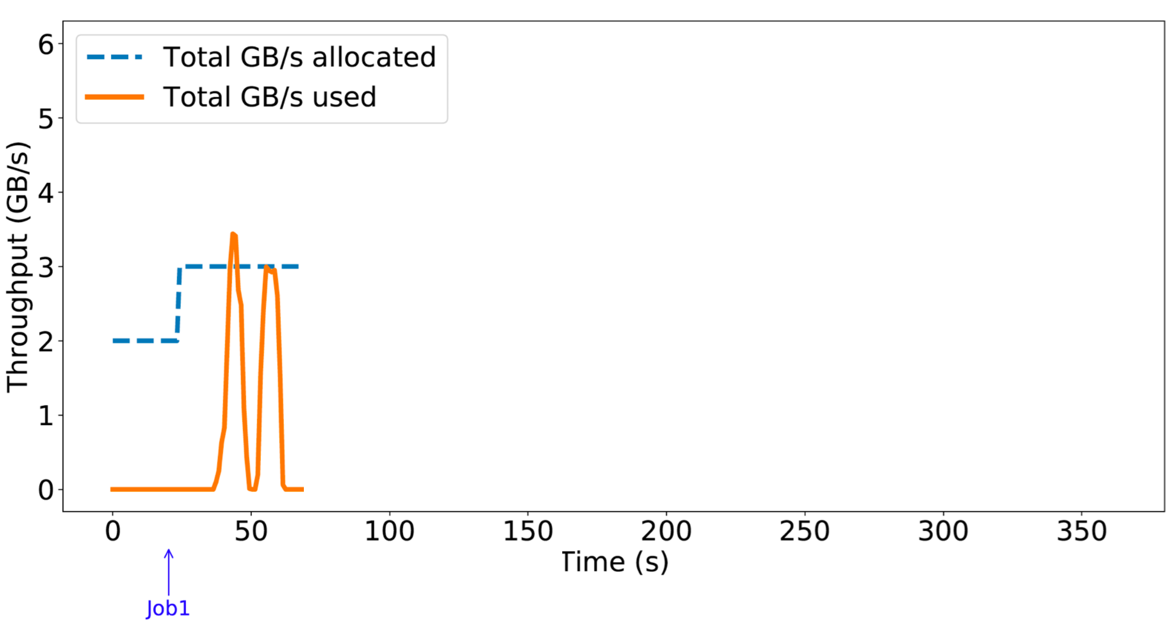
MapReduce sort job hints	
Ephemeral capacity	100 GB
Latency sensitive	False
Aggregate peak throughput	100 Gb/s

Application Storage Cost with Pocket

- Pocket leverages job attribute hints for cost-effective resource allocation and amortizes VM costs across multiple jobs, offering a pay-what-you-use model

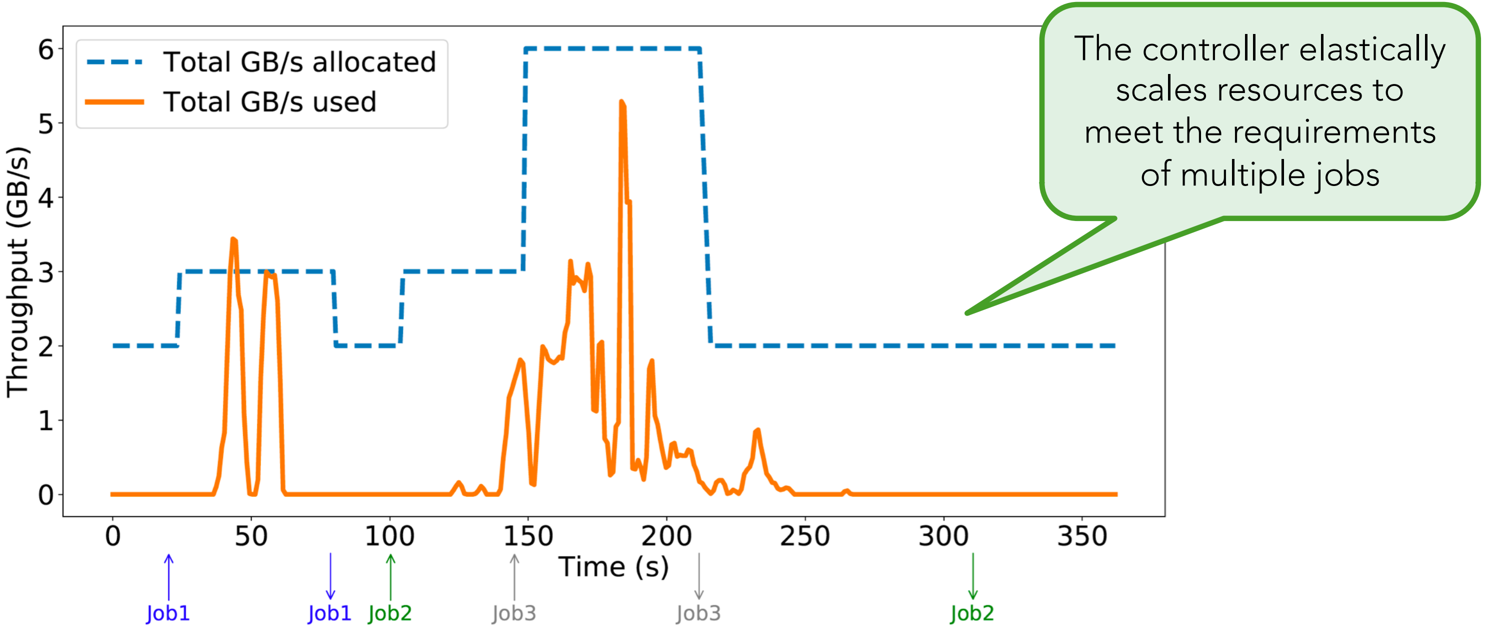


Autoscaling the Pocket Cluster



Job hints	Job1: Sort
Latency sensitive	False
Ephemeral data capacity	10 GB
Aggregate throughput	3 GB/s

Autoscaling the Pocket Cluster



Job hints	Job1: Sort	Job2: Video analytics	Job3: Sort
Latency sensitive	False	False	False
Ephemeral data capacity	10 GB	6 GB	10 GB
Aggregate throughput	3 GB/s	2.5 GB/s	3 GB/s

Conclusion

- Pocket is a distributed ephemeral storage system that:
  - Leverages multiple storage technologies
  - Rightsizes resource allocations for applications
  - Autoscales storage cluster resources based on usage
- We designed Pocket for ephemeral data sharing in serverless analytics. More generally, Pocket is an elastic, distributed `/tmp`.

www.github.com/stanford-mast/pocket

pocket