Imperial College London

Microsoft

LSDS
Large-Scale Data & Systems Group

NEPTUNE
Scheduling Suspendable Tasks for Unified Stream/Batch Applications

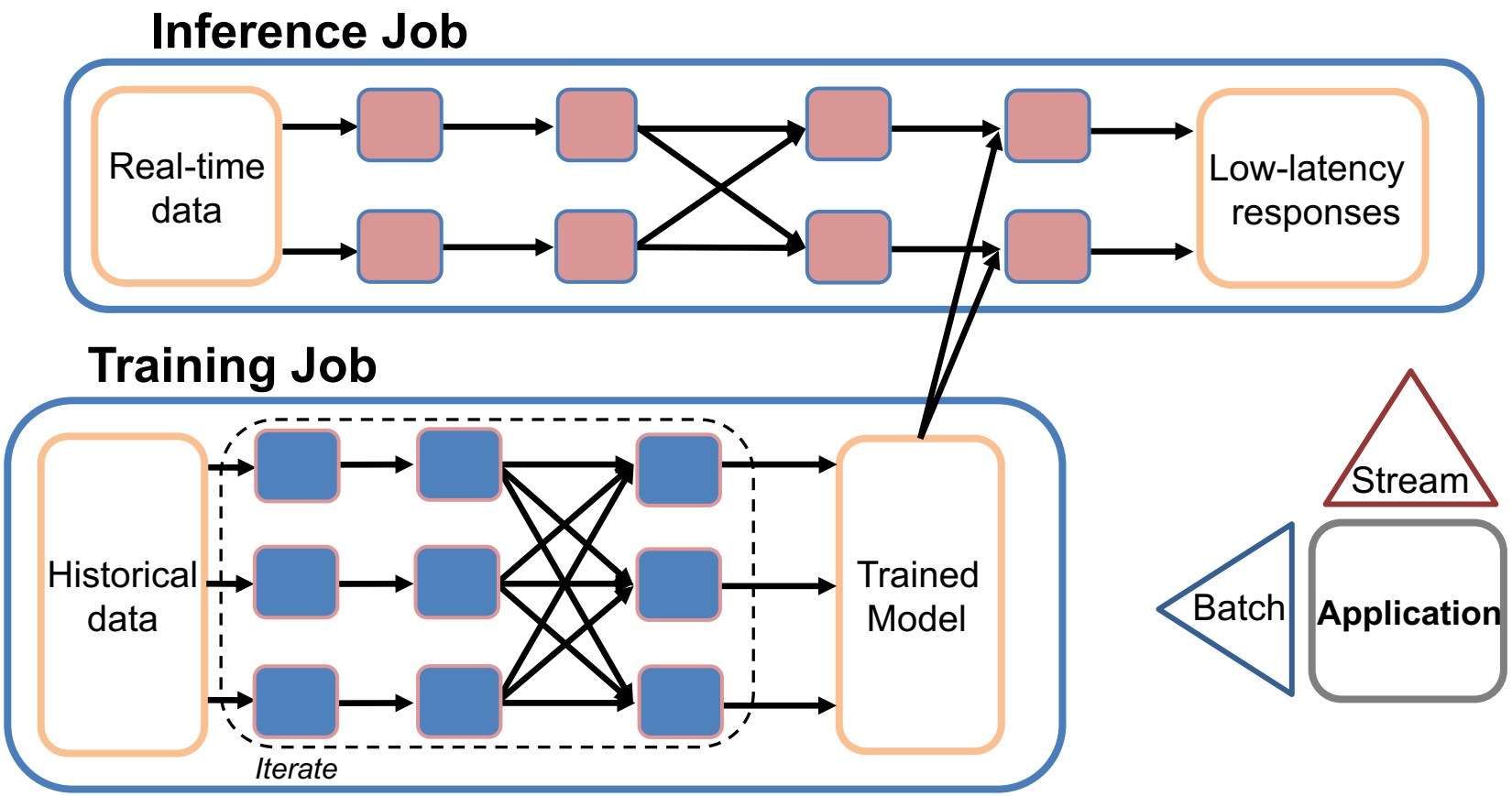
Panagiotis Garefalakis
Imperial College London
pgaref@imperial.ac.uk

Konstantinos Karanasos
Microsoft
kokarana@microsoft.com

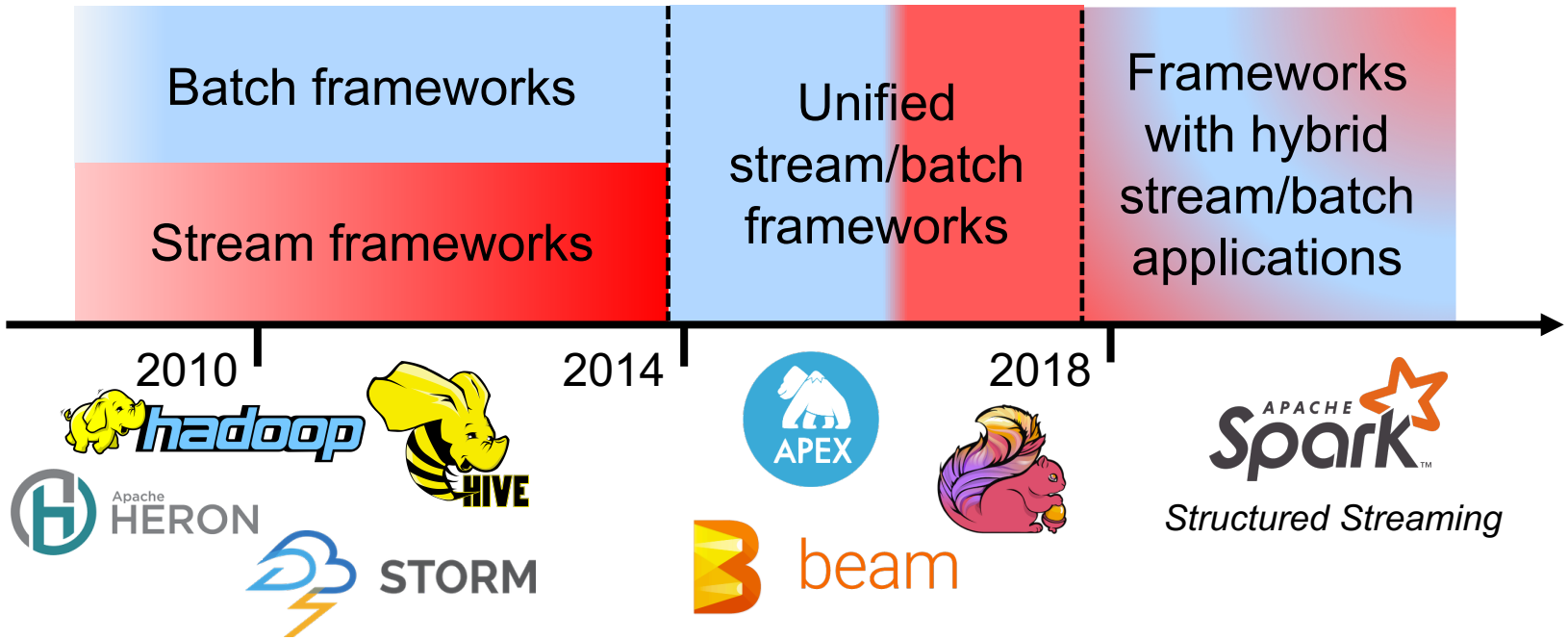
Peter Pietzuch
Imperial College London
prp@imperial.ac.uk

SoCC, Santa Cruz, California, November 2019

Unified application example



Evolution of analytics frameworks



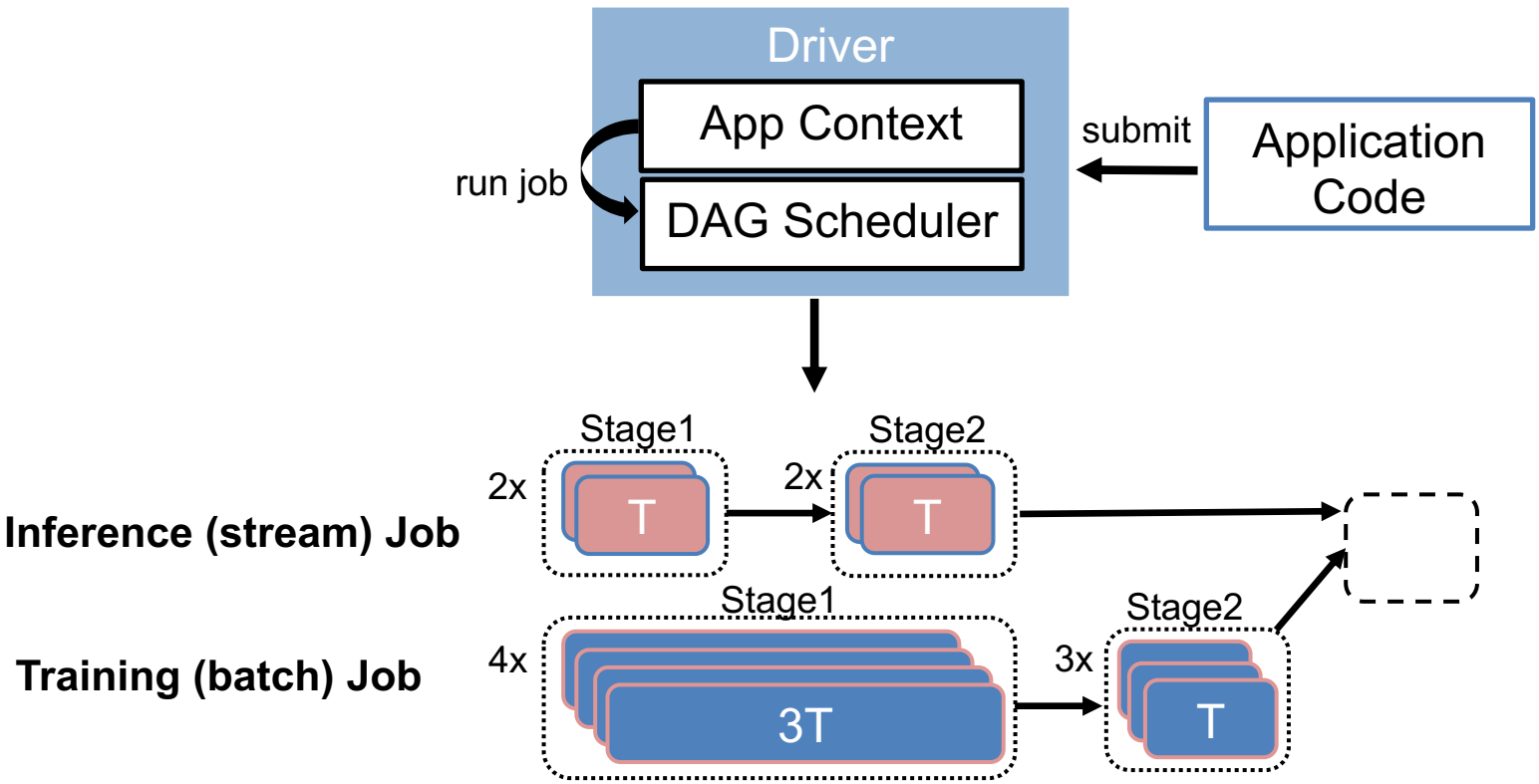
Stream/Batch application requirements

Requirements

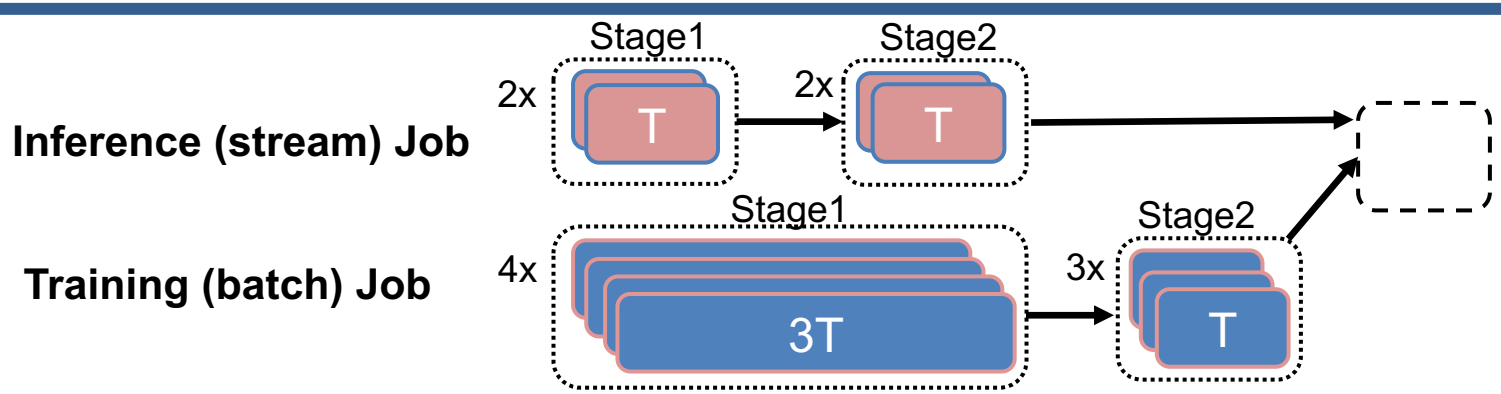
- **Latency:** Execute inference job with minimum delay
- **Throughput:** Batch jobs should not be compromised
- **Efficiency:** Achieve high cluster resource utilization

Challenge: schedule stream/batch jobs to satisfy their diverse requirements

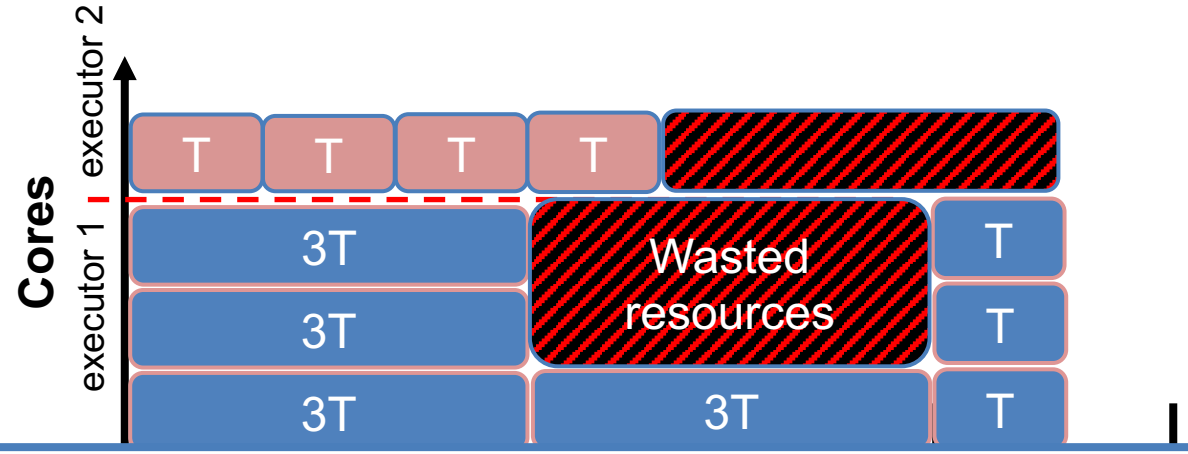
Stream/Batch application scheduling



Stream/Batch application scheduling

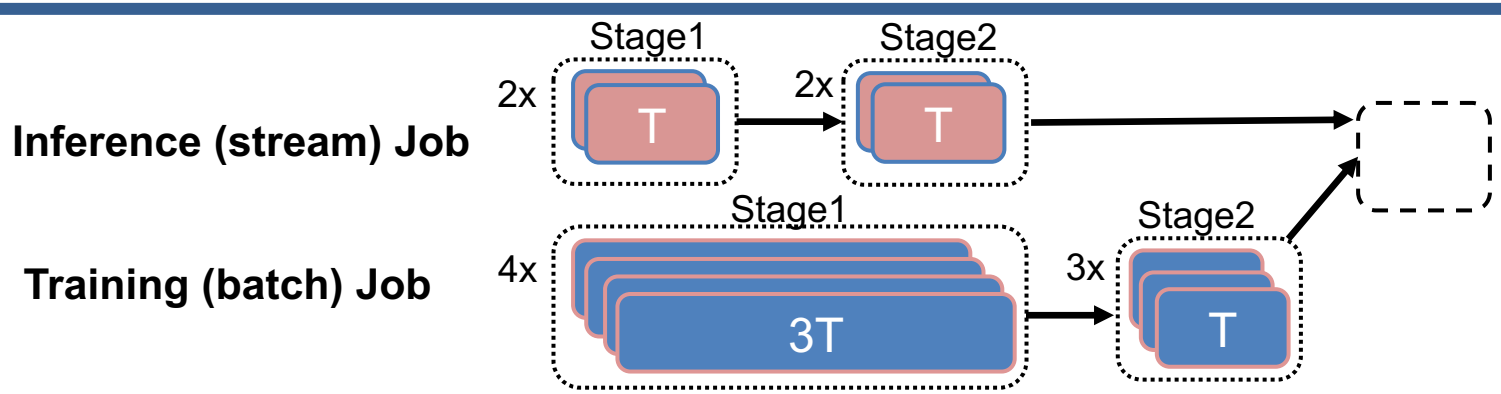


> **Static allocation:** dedicate resources to each job

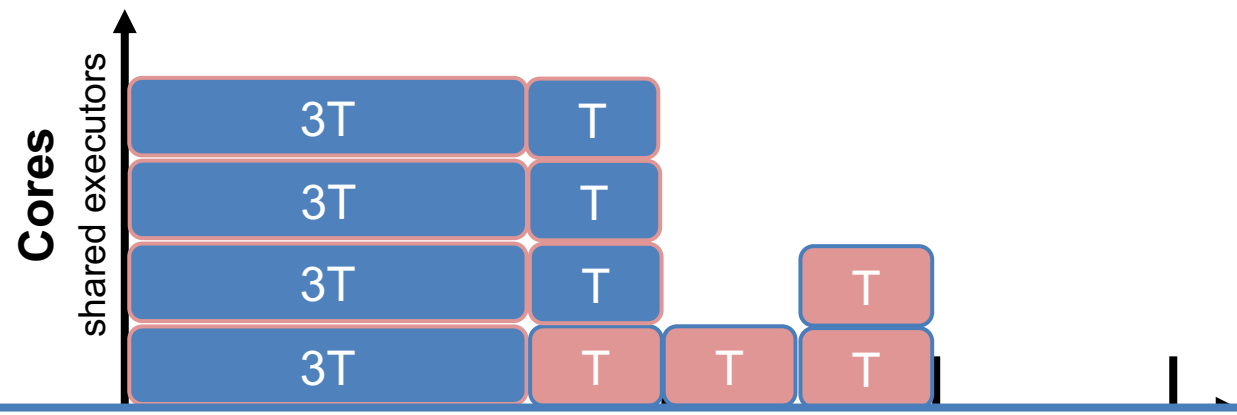


Resources can not be shared across jobs

Stream/Batch application scheduling

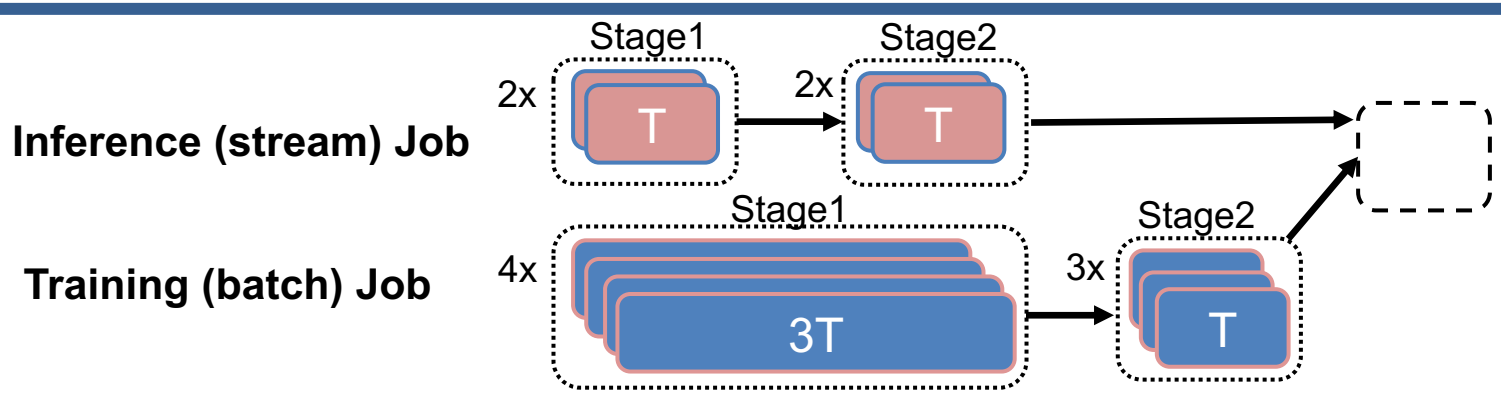


> **FIFO:** first job runs to completion

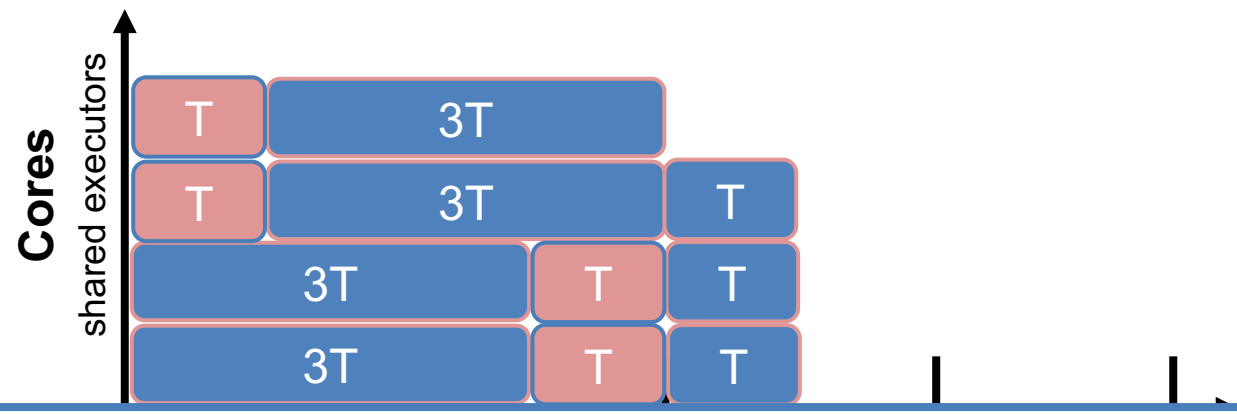


Long batch jobs increase stream job latency

Stream/Batch application scheduling

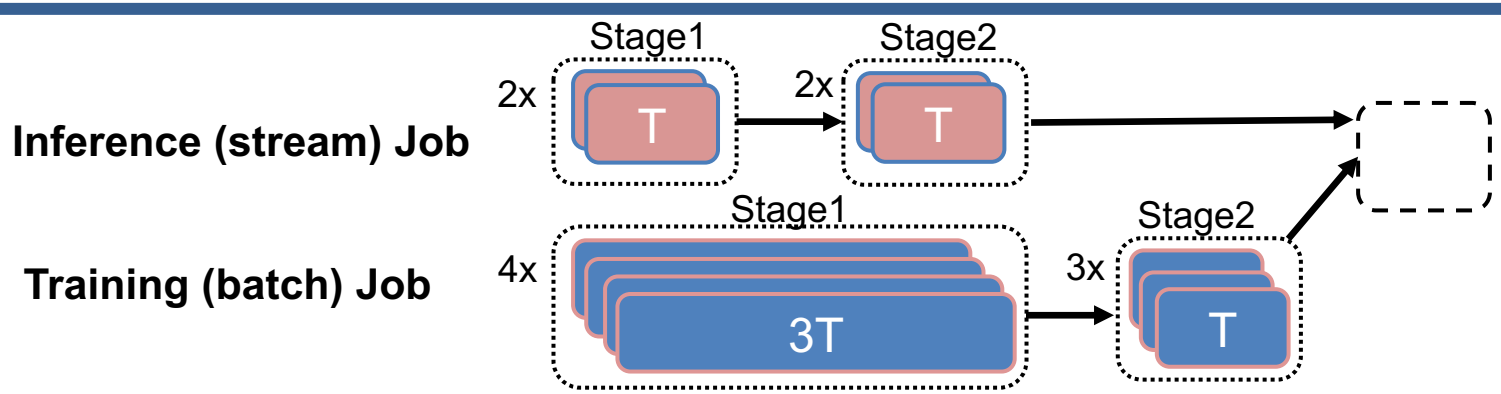


> **FAIR:** weight share resources across jobs

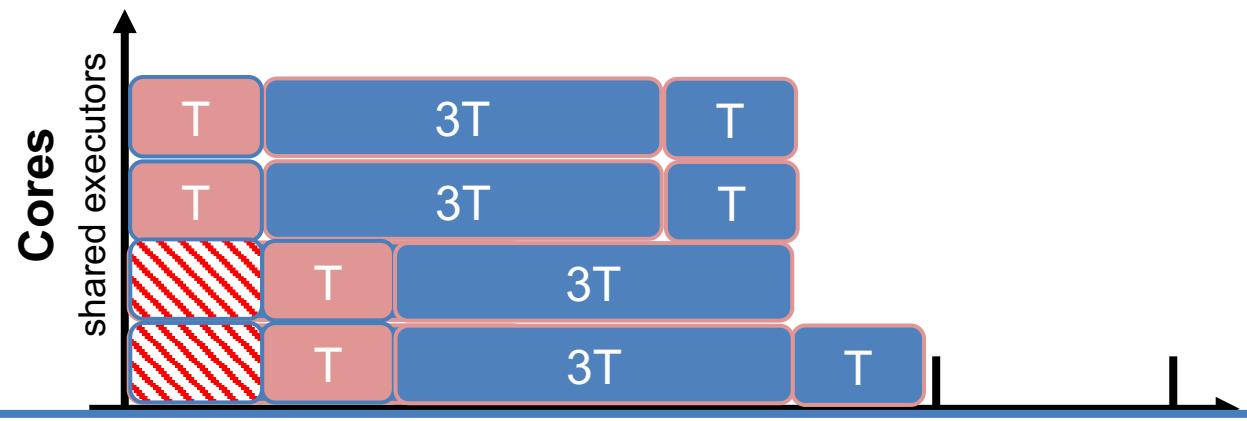


Better packing with non-optimal latency

Stream/Batch application scheduling

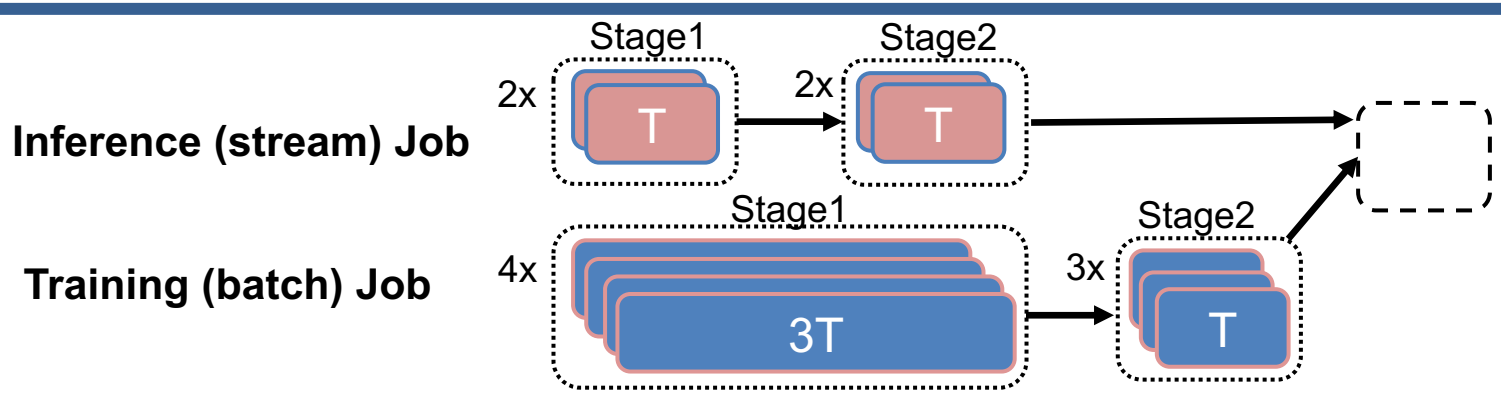


> **KILL:** avoid queueing by preempting batch tasks

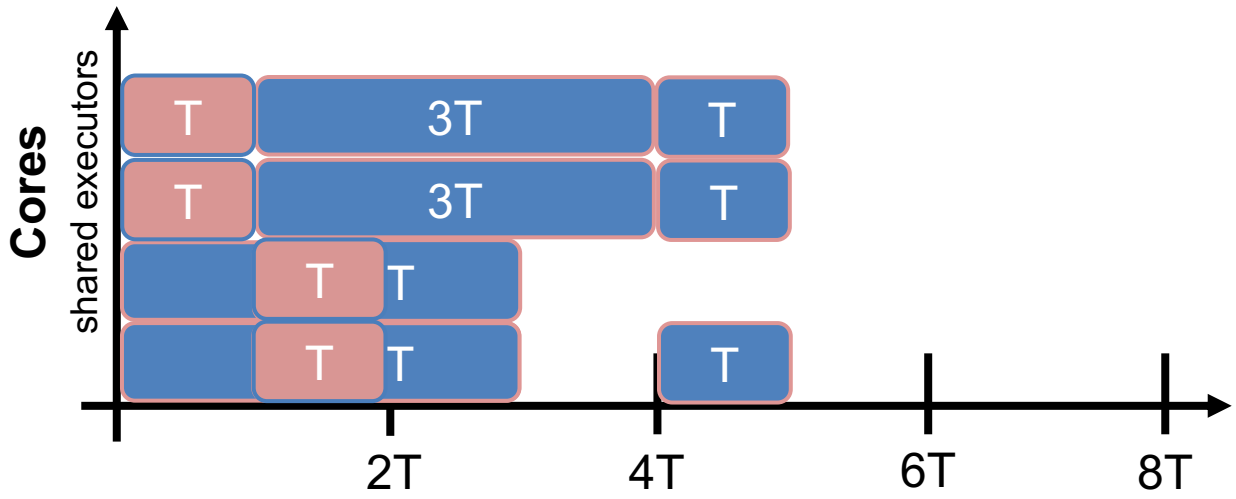


Better latency at the expense of extra work

Stream/Batch application scheduling



> **NEPTUNE:** minimize queueing and wasted work!



Challenges

- How to minimize queuing for latency-sensitive jobs and wasted work?
- How to natively support stream/batch applications?
- How to satisfy different stream/batch application requirements and high-level objectives?

NEPTUNE

Execution framework for Stream/Batch applications

> How to minimize queuing for latency-sensitive jobs and wasted work?

Support suspendable tasks

> How to natively support stream/batch applications?

Unified execution framework on top of Spark

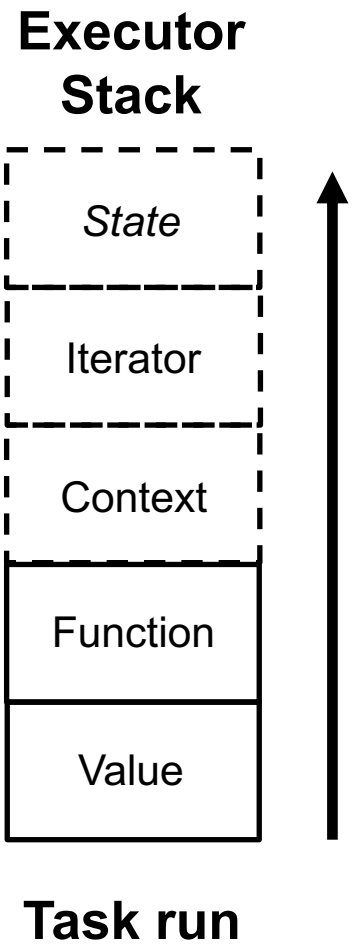
Structured Streaming

> How to satisfy different stream/batch application requirements and high-level objectives?

Introduce pluggable scheduling policies

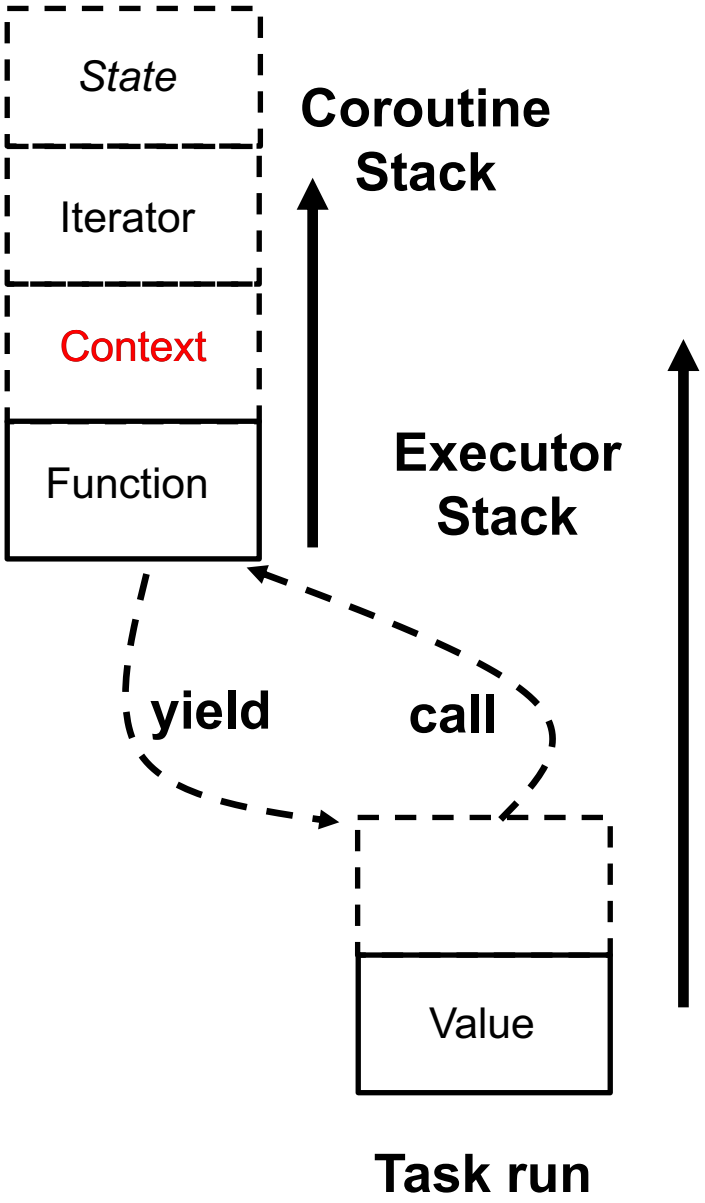
Typical tasks

- **Tasks:** apply a function to a partition of data
- Subroutines that run in executor to completion
- Preemption problem:
  - Loss of progress (kill)
  - Unpredictable preemption times (checkpointing)



Suspendable tasks

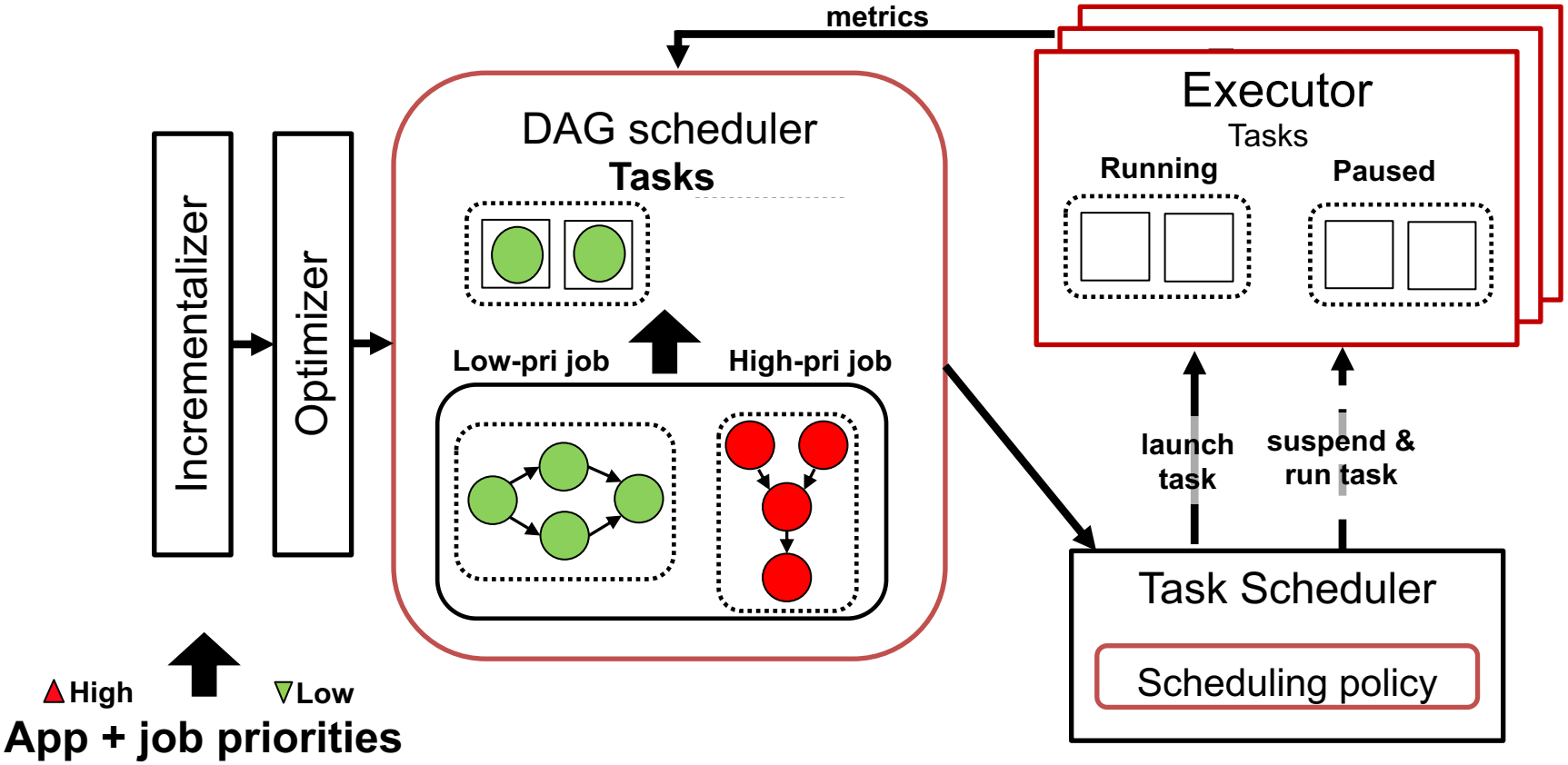
- **Idea:** use coroutines
  - Separate stacks to store task state
  - Yield points handing over control to the executor
- Cooperative preemption:
  - Suspend and resume in *milliseconds*
  - Work-preserving
- Transparent to the user



https://github.com/storm-enroute/coroutines

Execution framework

- **Problem**: not just assign but also suspend and resume
- **Idea**: centralized scheduler with pluggable policies



Scheduling policies

- **Idea**: policies trigger task suspension and resumption
  - Guarantee that stream tasks bypass batch tasks
  - Satisfy higher-level objectives i.e. balance cluster load
  - Avoid starvation by suspending up to a number of times

- Load-balancing (LB): takes into account executors' memory conditions and equalize the number of tasks per node

- Locality- and memory aware (LMA): respect task locality preferences in addition to load-balancing

Implementation

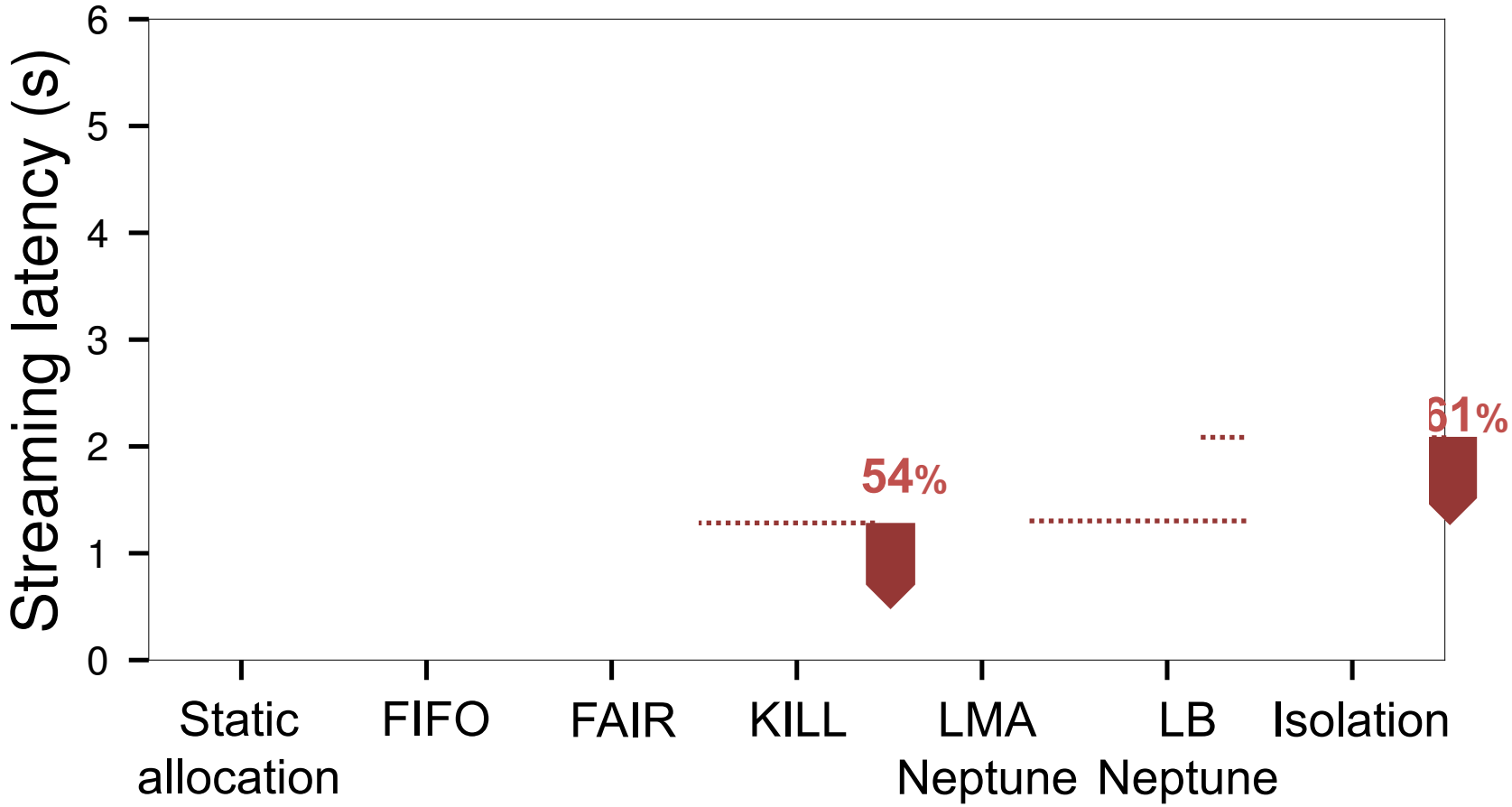
- Built as an extension to Spark **2.4.0** (https://github.com/lsds/Neptune)
- Ported all ResultTask, ShuffleMapTask functionality across programming interfaces to coroutines
- Extended Spark's DAG Scheduler to allow job stages with different requirements (priorities)
- Added additional Executor performance metrics as part of the heartbeat mechanism

Azure deployment

- Cluster
  - 75 nodes with 4 cores and 32 GB of memory each

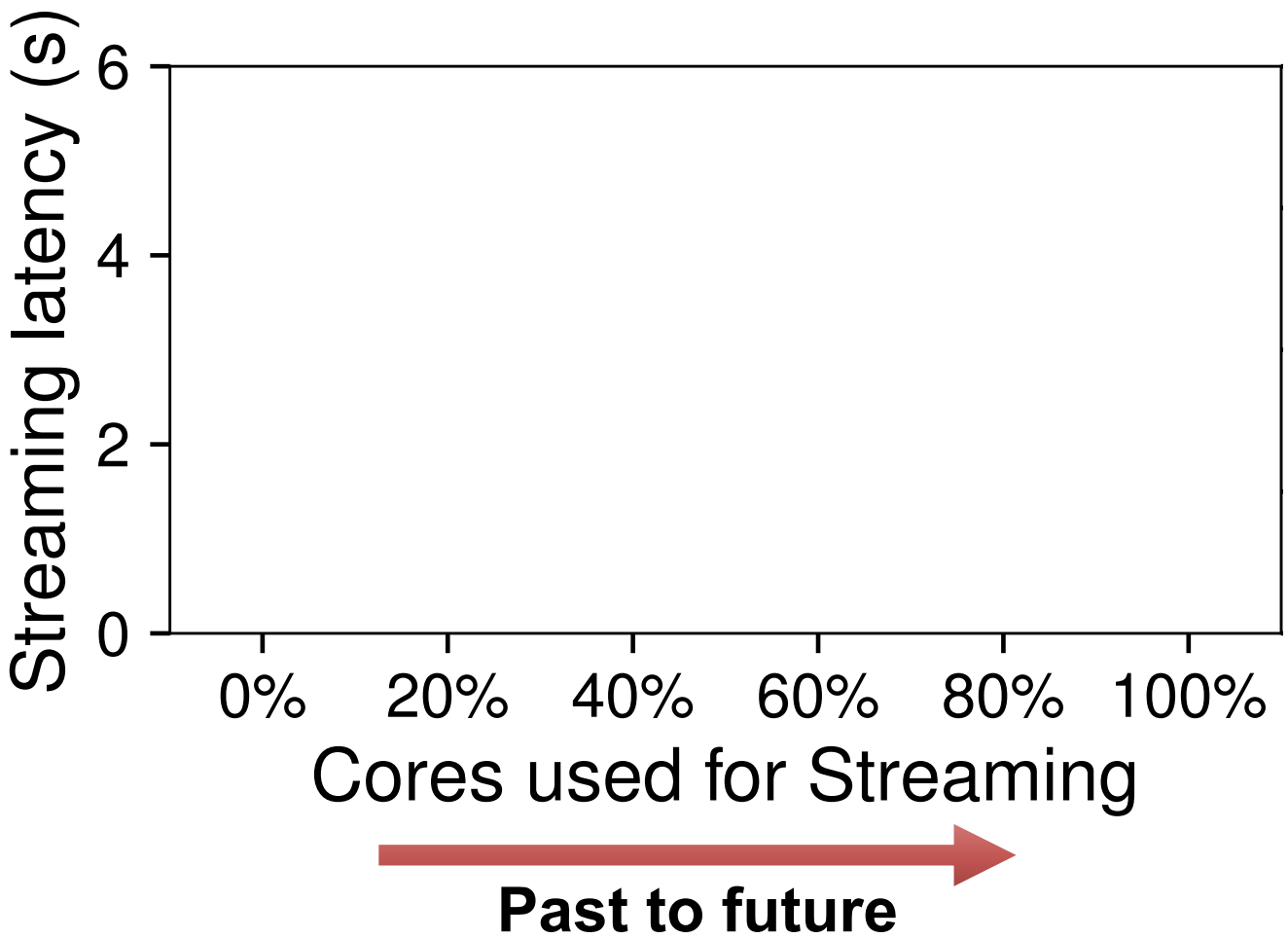
- Workloads
  - **LDA**: ML training/inference application uncovering hidden topics from a group of documents
  - **Yahoo Streaming Benchmark**: ad-analytics on a stream of ad impressions
  - **TPC-H** decision support benchmark

Benefit of NEPTUNE in stream latency



NEPTUNE achieves latencies comparable to the ideal for the latency-sensitive jobs

Impact of resource demands in performance



Efficiently share resources with low impact on throughput

Summary

NEPTUNE supports complex unified applications with diverse job requirements!

- Suspendable tasks using coroutines
- Pluggable scheduling policies
- Continuous unified analytics

https://github.com/lsds/Neptune

Thank you!
Questions?

Panagiotis Garefalakis
pgaref@imperial.ac.uk