

Towards an Operating System for Big Data

April 24, 2015

Byung-Gon Chun
Computer Science and Engineering Department
Seoul National University

Joint work with the REEF Apache Incubator project team

About Me

Backgrounds on operating and networked systems

My research centers around big data systems, cloud computing, mobile systems, and security

Leader: Cloud and Mobile Systems Lab
(SW Star Lab)

Co-Founder: REEF Apache Incubator Project,
TaintDroid

About Me: Selected Projects

Mobile

TaintDroid
(OSDI '10, CACM '14, TOCS '14)
Mantis
(USENIX ATC '13)

Mobile-Cloud

CloneCloud
(HotOS '09, EuroSys '11)
Mobius
(MobiSys '12)

Backend

REEF (SIGMOD '15, VLDB '13)
Data Analytics Profiling (NSDI '15)
Elastic Memory (HotOS '15)
Pinterest Analysis
(SIGMETRICS '14)
MegaPipe (OSDI '12)
RouteBricks (SOSP '09)
NetComplex (NSDI '08)
Attested AM (SOSP '07)
DONA (SIGCOMM '07)

How This Project Got Started

A Plethora of Open Source Big Data Systems



REEF



APACHE
HBASE

Dryad

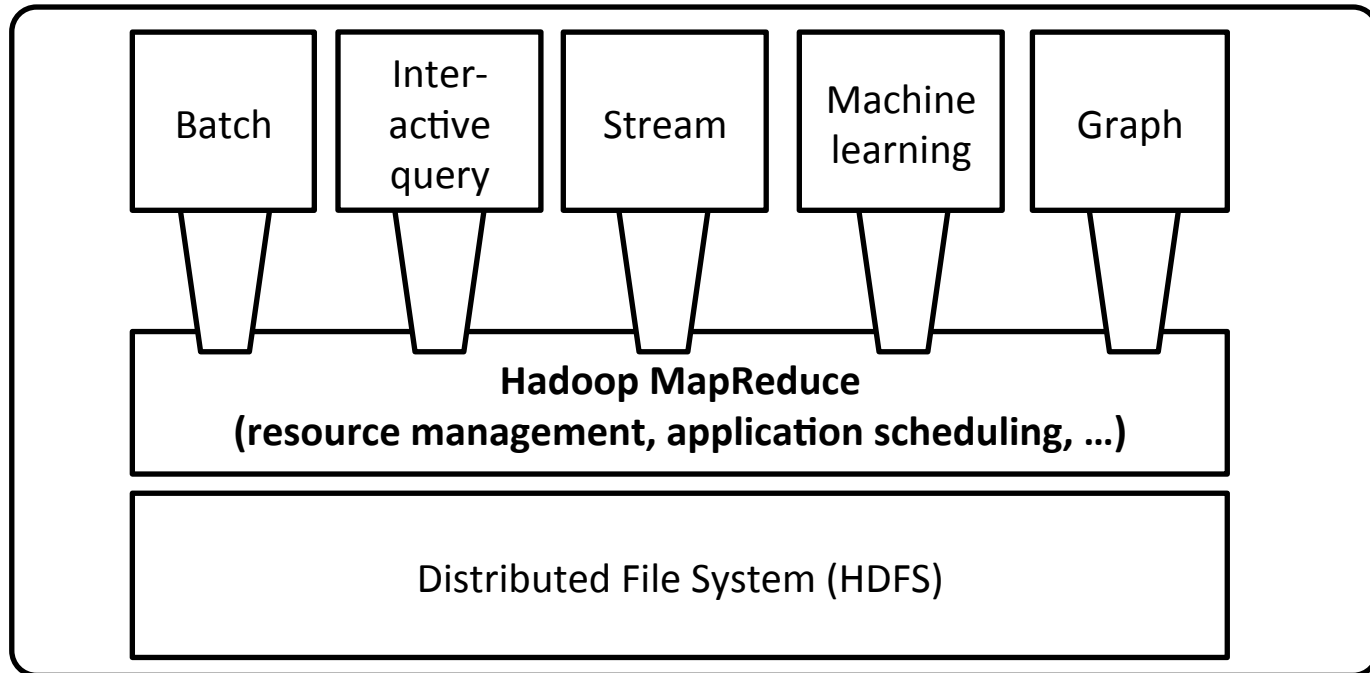


Storm



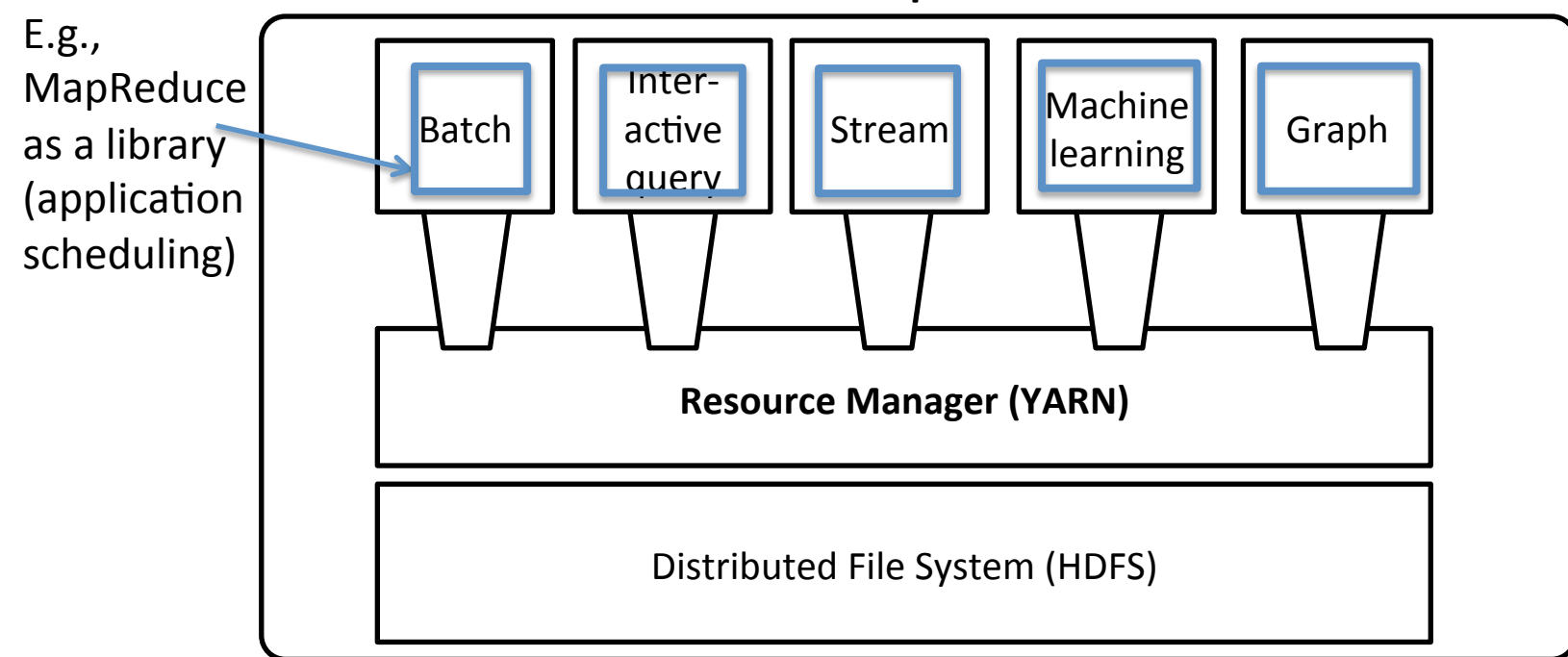
From a Monolithic Big Data Processing System

Hadoop v1



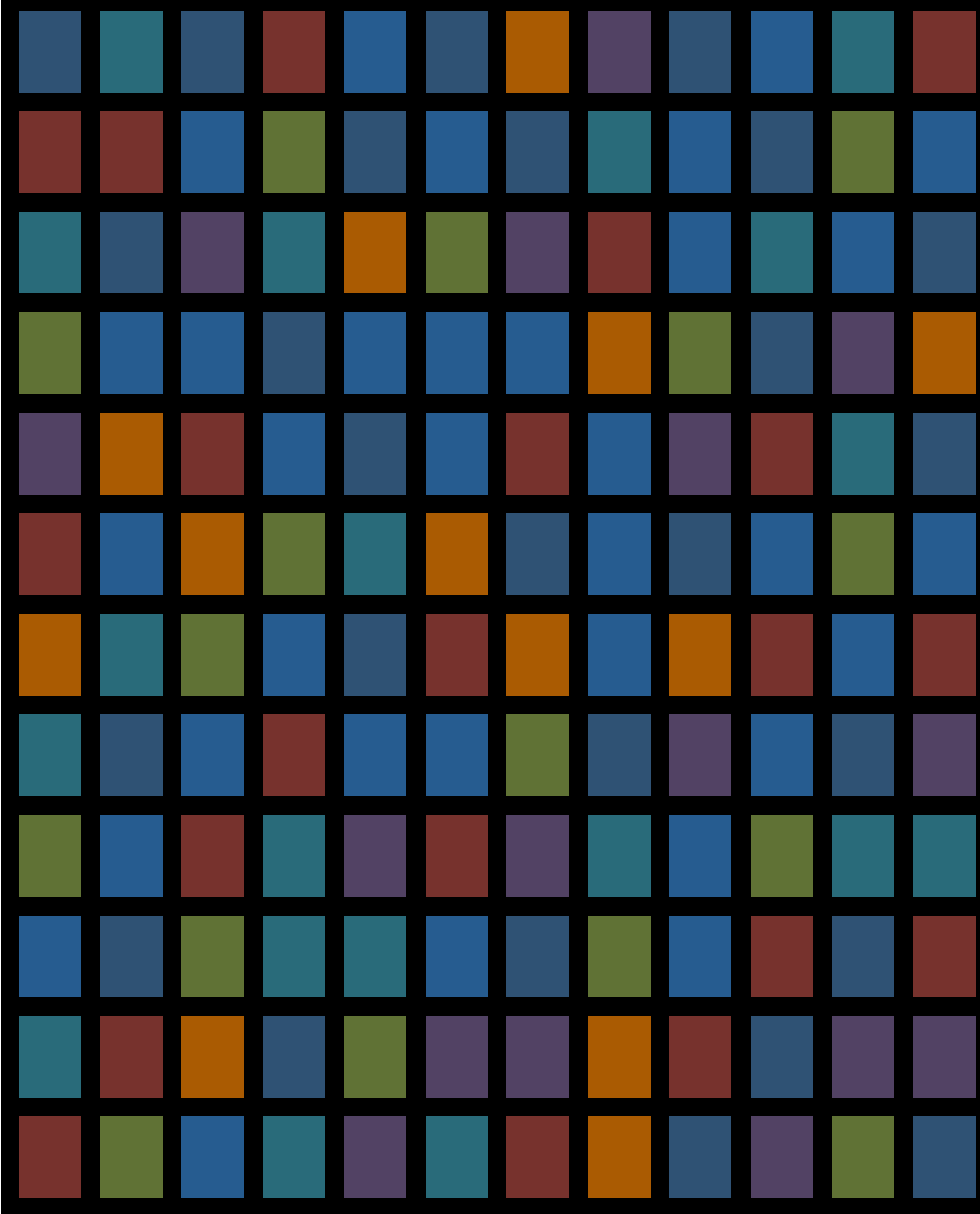
A Recent Step Towards Refactoring Big Data Processing Systems

Hadoop v2

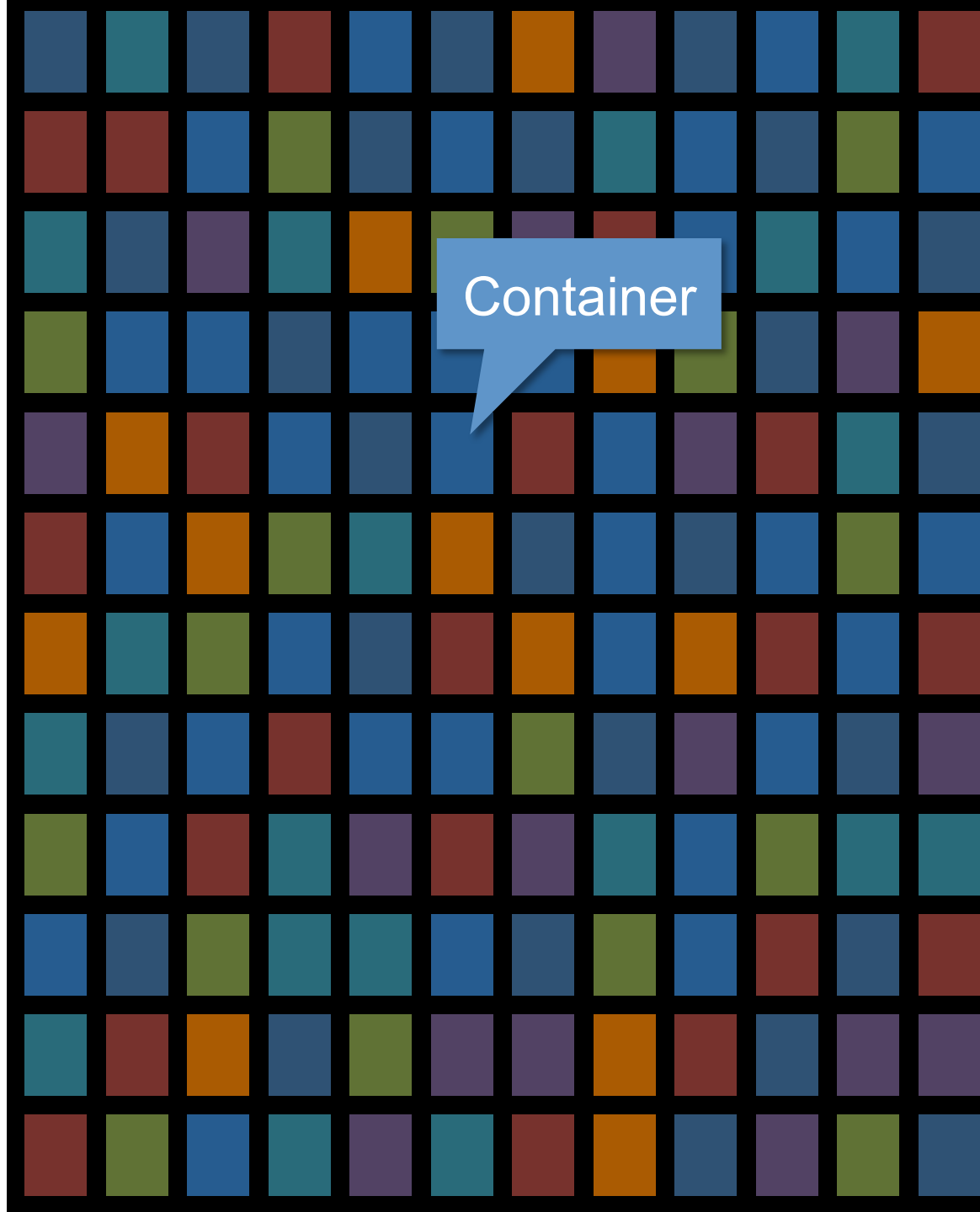


- Split up resource management and application job scheduling

Resource Managers



Resource Managers



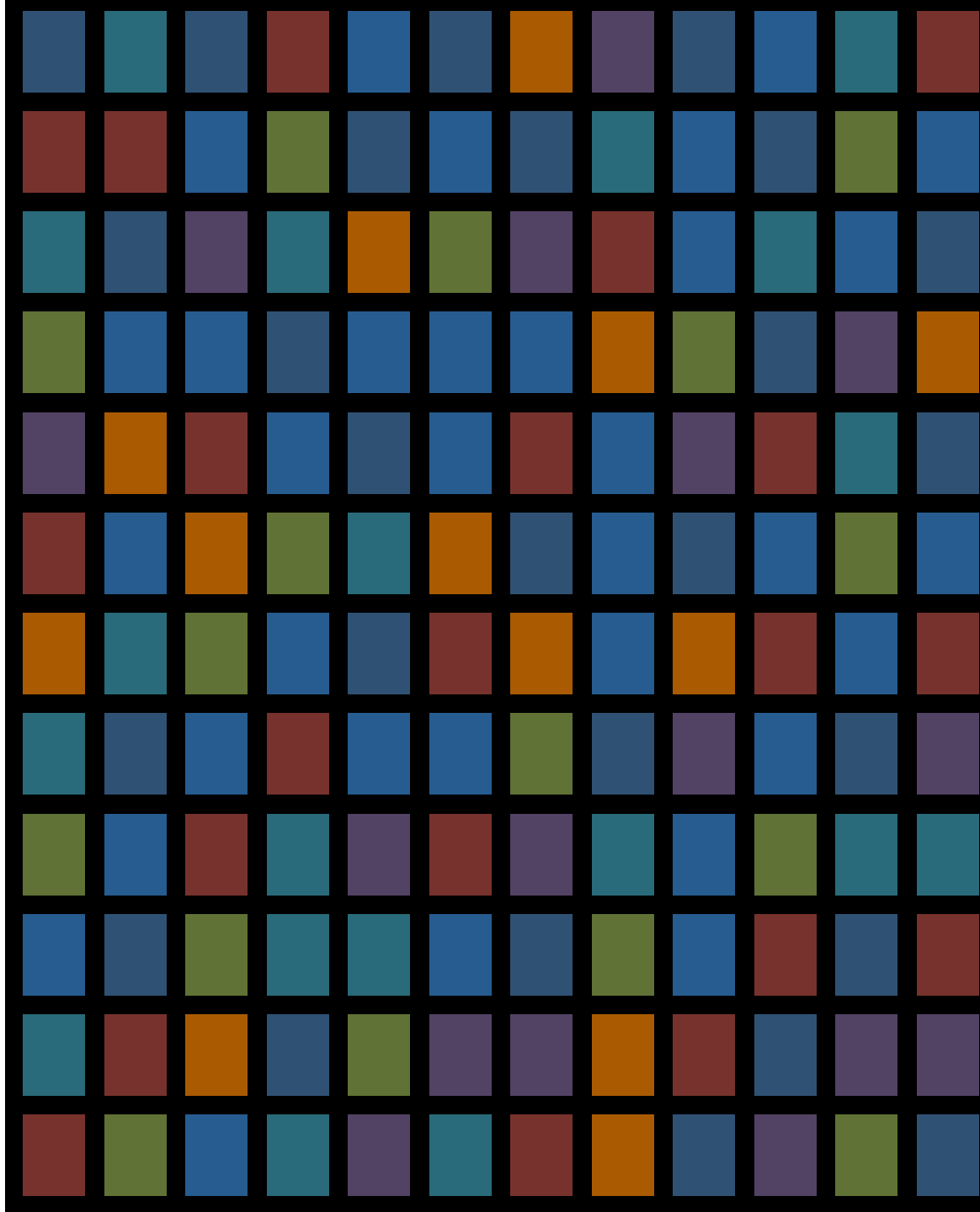
Resource Managers

True multi-tenancy...

Many **workloads**: Streaming,
Batch, Interactive, ...

Many **users**: Production Jobs,
Ad-Hoc Jobs, Experiments, ...

...but, only for sophisticated
apps



Resource Managers

True multi-tenancy...

Many **workloads**: Streaming, Batch, Interactive, ...

Many **users**: Production Jobs,
Ad-Hoc Jobs, Experiments, ...

...but, only for sophisticated apps

Fault tolerance

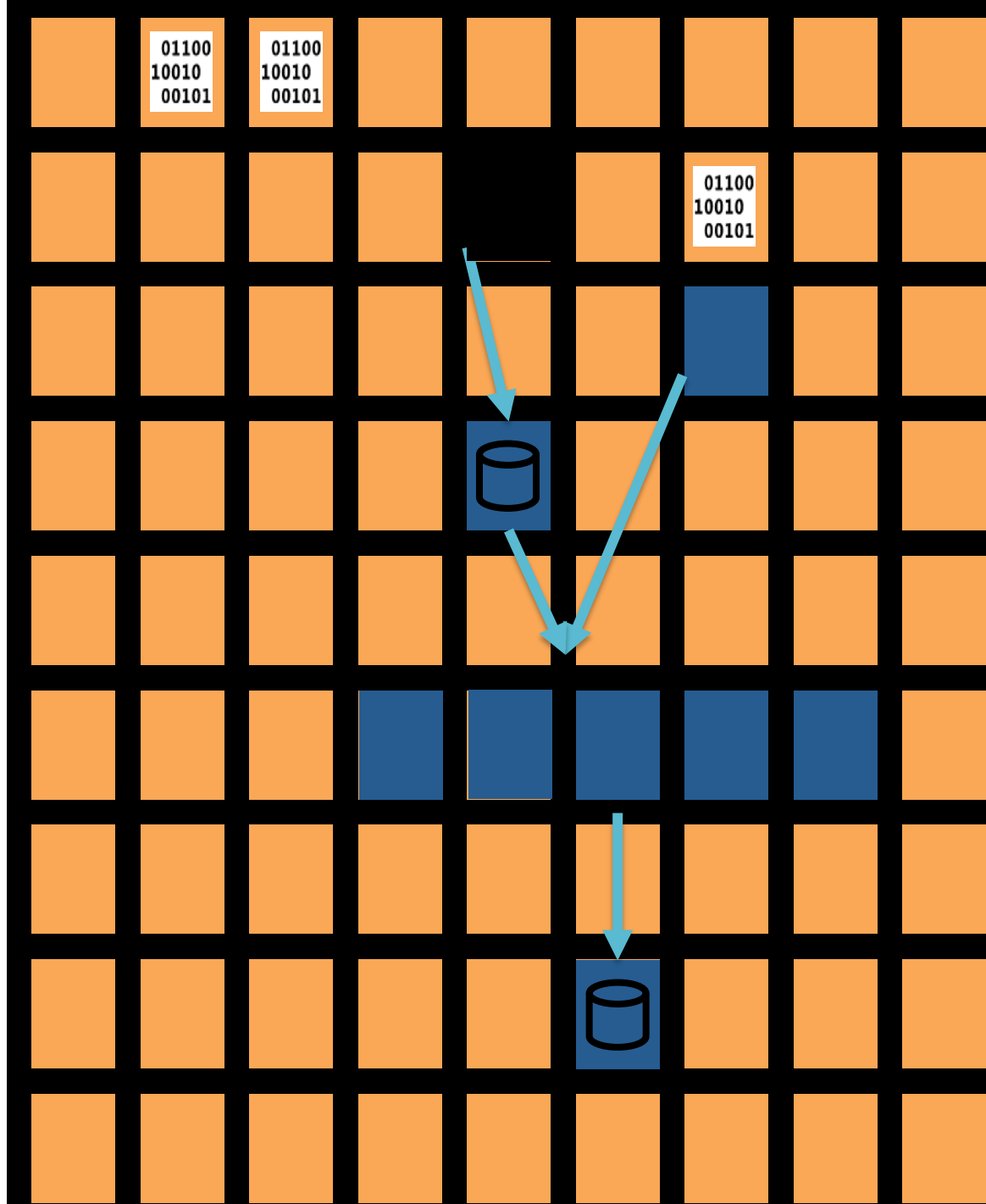
Pre-emption

Elasticity

Example 1: SQL/MapReduce

Fault tolerance

Elasticity

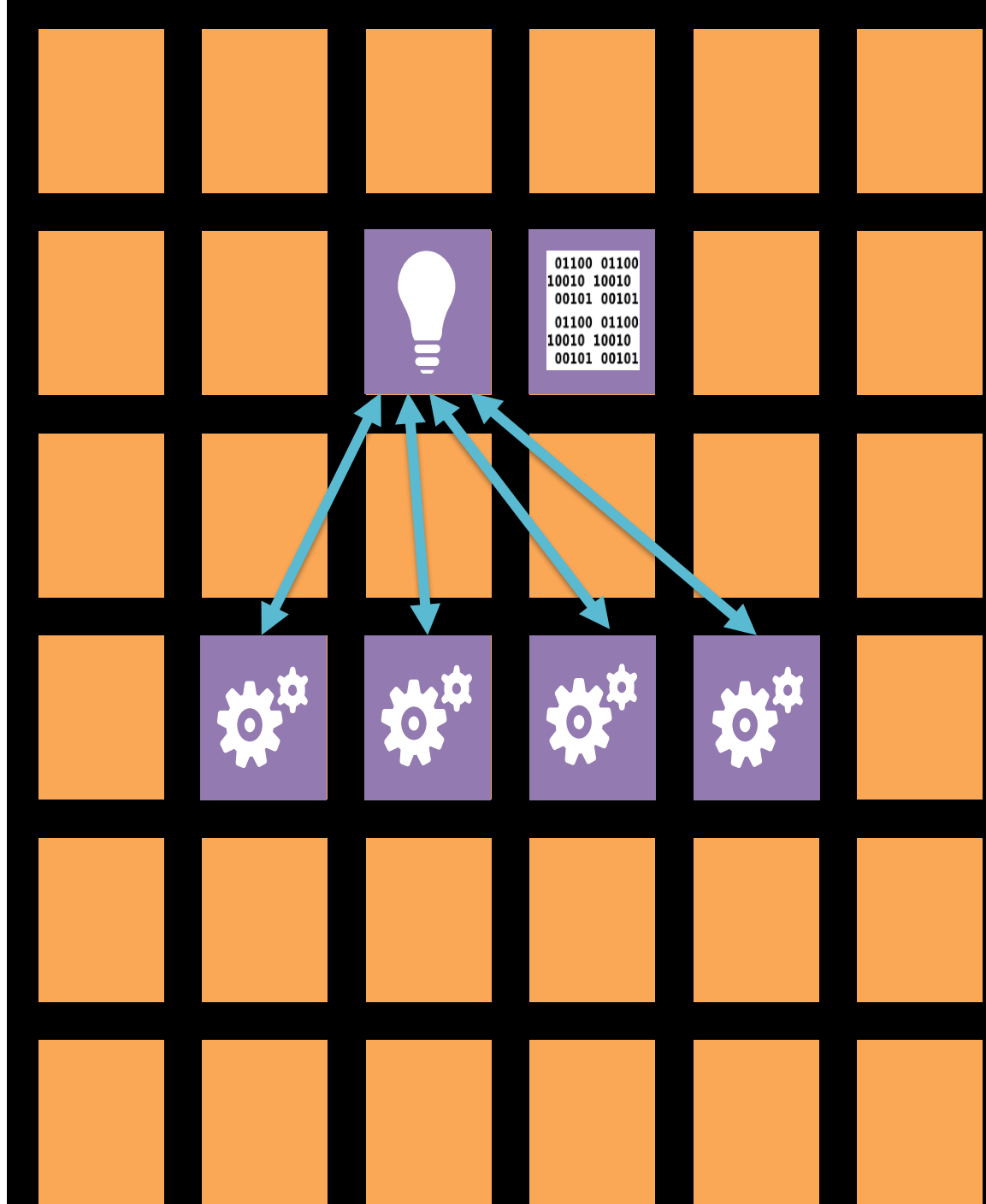


Example 2: Machine learning

Fault tolerance

Elasticity

Iterative computations



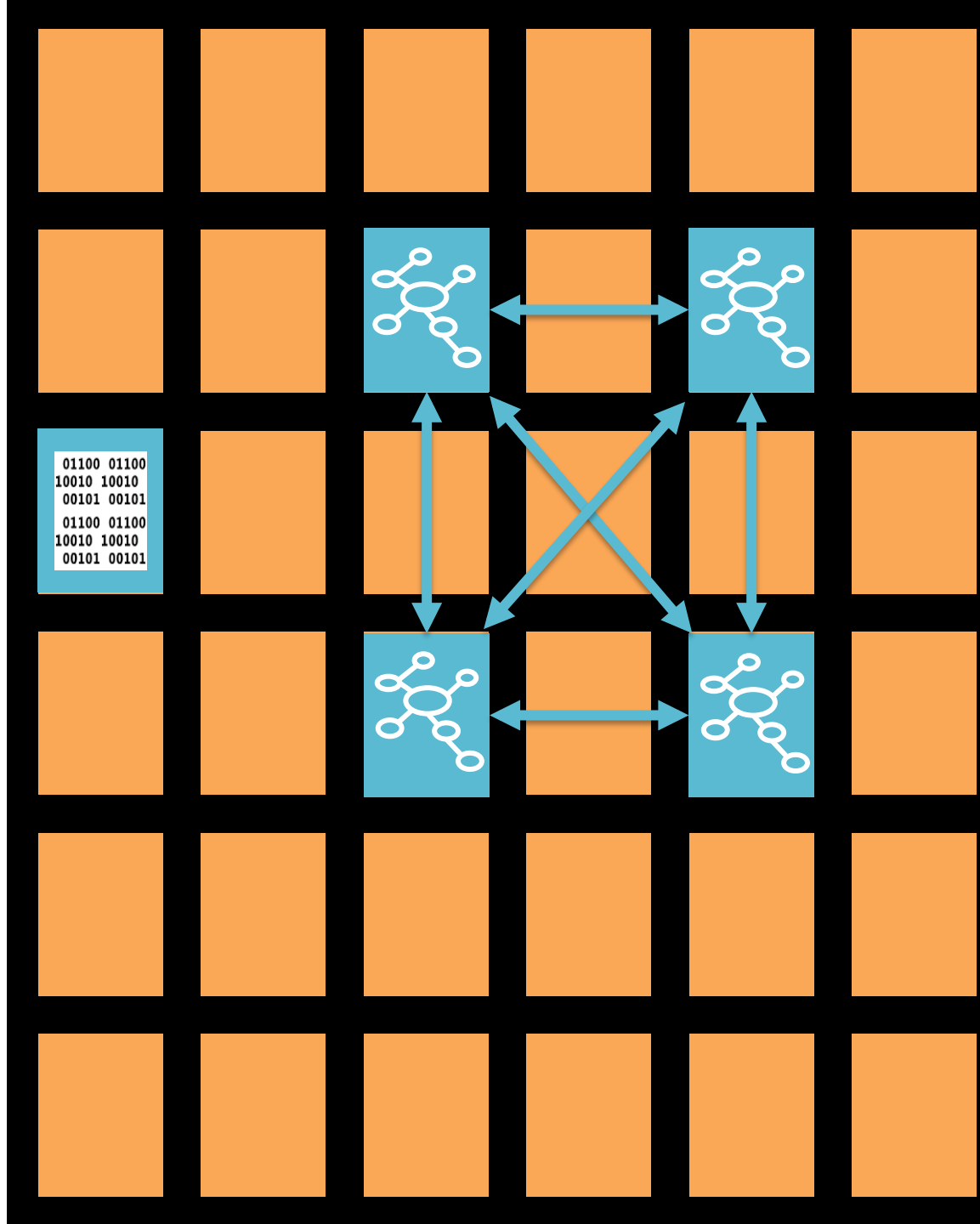
Example 3: Graph processing

Fault tolerance

Elasticity

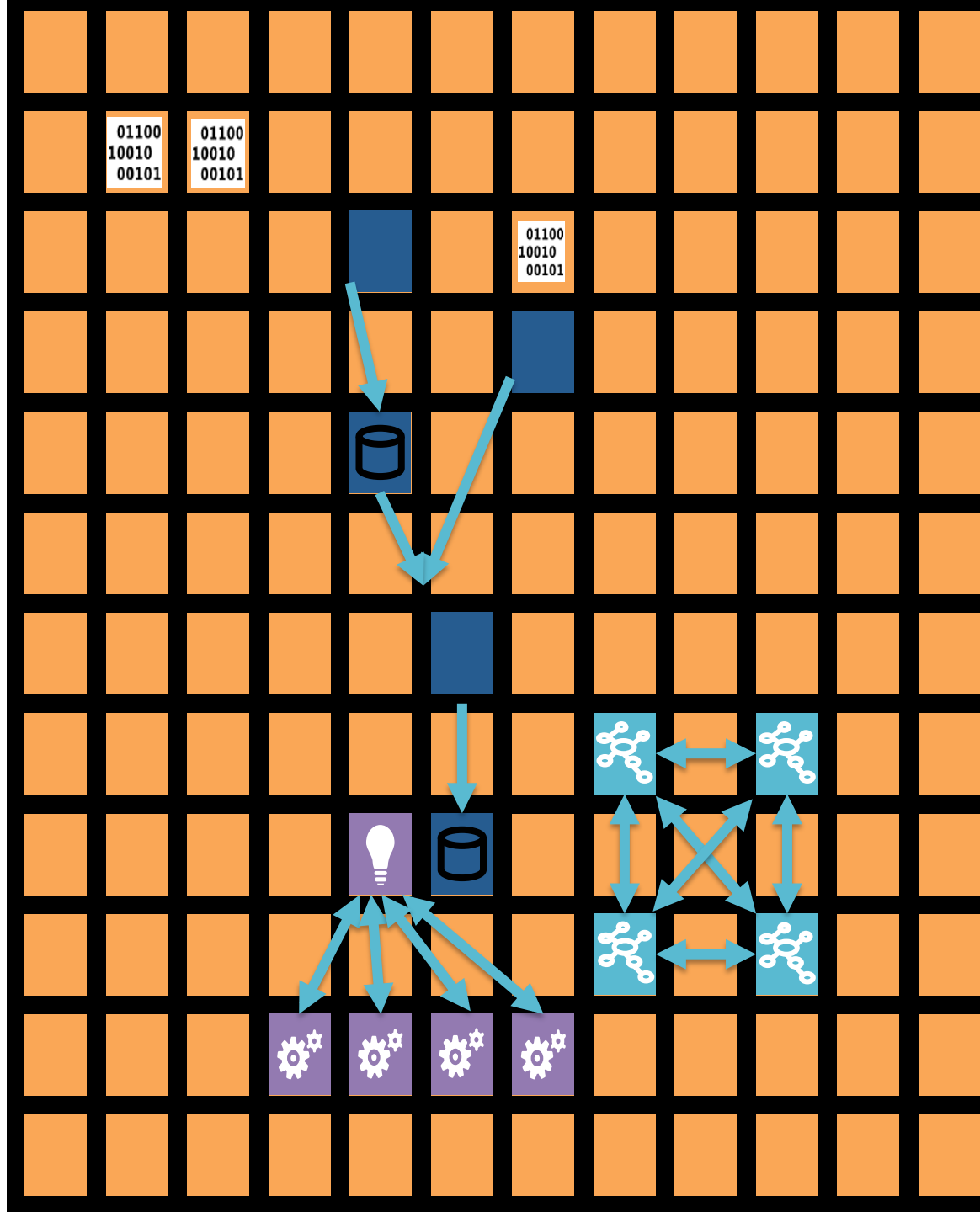
Iterative computations

Low latency
communication



Silos

The flexibility comes
at a high cost in
development and
operation



Decomposing Big Data Applications

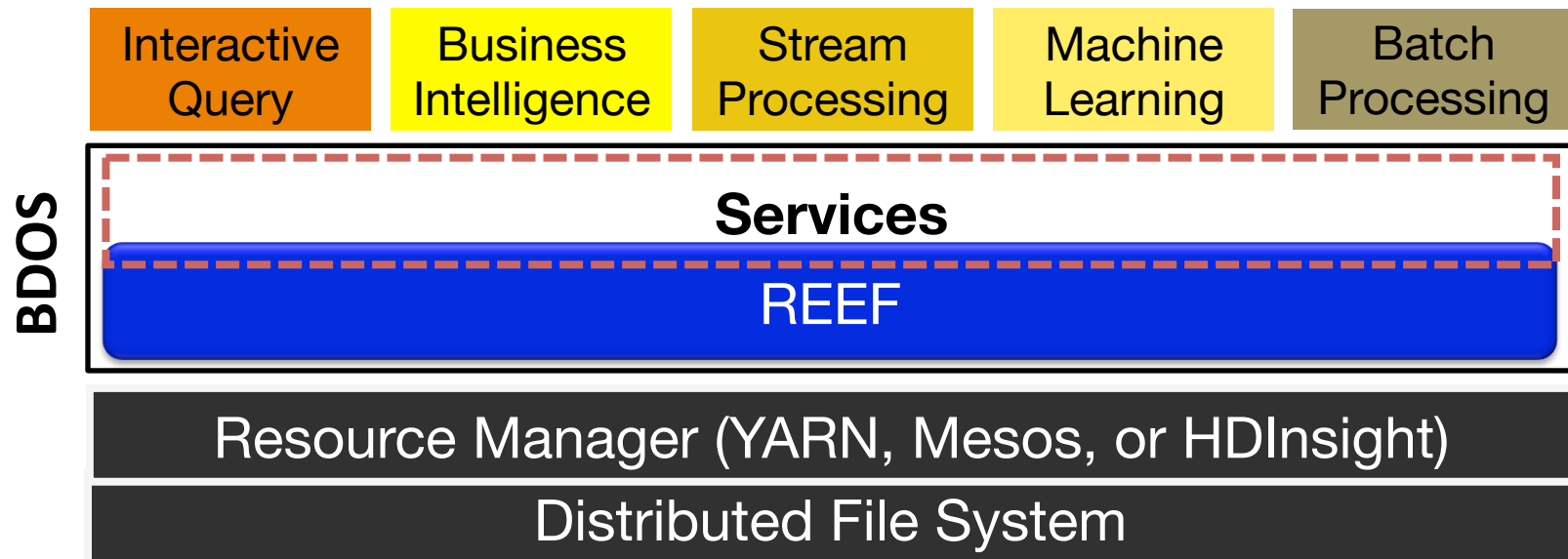
- **Control plane:** coordinates application job tasks, handles faults, provides heartbeats, configures jobs, etc.
 - Master-slave pattern
- **Data plane:** moves and processes data

Dive into Apache REEF

What is REEF (Retainable Evaluation Execution Framework)?

A scale-out computing fabric that eases the development of Big Data applications on top of resource managers

REEF: Towards a Big Data Operating System



- ✓ Reusable control plane for coordinating data plane tasks
- ✓ Virtualization of resource managers
- ✓ Container and state reuse across tasks from heterogeneous frameworks
- ✓ Simple configuration management
- ✓ Scalable event handling

Motivating Architectural Choice #1

- How to structure master and slave code?
- Initial prototype based on Java Future:
pull-based API
- Problems
 - Blocking call
 - Tricky error handling
 - Tricky performance tuning

Architectural Choice #1

- Application master based on event-driven programming and engine (Wake)
- Asynchronous event push-based API
 - User code (event handlers) that reacts to various REEF events
- Progressive resource acquisition
- Easy error handling
- Easy performance tuning by properly allocating thread pools to event handlers by Wake

Motivating Architectural Choice #2

- Typical way of configuration: A key-value model for configuration
 - More than 90% of options in seven open-source projects totaling million lines of code (Rabkin and Katz, 2011)
- Prone to misconfiguration
 - Pervasive documentation errors for options that do not exist and undocumented options
 - Mismatching value type
 - Overriding configurations
 - Configuration mistakes (no default.fs.name)

Architectural Choice #2

- Configuration using Dependency Injection (Tang)
 - Bind an implementation to an interface
 - Bind a value to a configuration parameter
 - Static configuration checks / object construction time checks
 - Avoid dynamic configuration done by subscriptions at run time (e.g., use statically configured event flows)

Architectural Choice #2

- Tang properties
 - “Configuration” are just data
 - Configuration options can be set at most once
 - A large subset of Tang’s public API is commutative
 - IDE, static analysis and documentation tools
 - Cross-language support

At Most Once

```
final Configuration driverConfiguration = DriverConfiguration.CONF
    .set(DriverConfiguration.DRIVER_IDENTIFIER, "Hello")
    .set(DriverConfiguration.DRIVER_IDENTIFIER, "Hello2")
    .set(DriverConfiguration.GLOBAL_LIBRARIES,
        EnvironmentUtils.getClassLocation(HelloDriver.class))
    .set(DriverConfiguration.ON_DRIVER_STARTED,
        HelloDriver.StartHandler.class)
    .set(DriverConfiguration.ON_EVALUATOR_ALLOCATED,
        HelloDriver.EvaluatorAllocatedHandler.class);
```

```
java -cp target/reef-tutorial-1.0-SNAPSHOT-shaded.jar
```

```
edu.snu.cms.reef.tutorial.HelloREEF
```

```
Exception in thread "main" java.lang.IllegalArgumentException: Attempt to re-add:
[com.microsoft.tang.formats.RequiredParameter@507895d8] old value: Hello new
value Hello2
```

```
    at com.microsoft.tang.util.MonotonicHashMap.put(MonotonicHashMap.java:28)
    at
```

```
com.microsoft.tang.formats.ConfigurationModule.set(ConfigurationModule.java:138)
    at edu.snu.cms.reef.tutorial.HelloREEF.getDriverConfiguration(HelloREEF.java:46)
    at edu.snu.cms.reef.tutorial.HelloREEF.runHelloReef(HelloREEF.java:60)
    at edu.snu.cms.reef.tutorial.HelloREEF.main(HelloREEF.java:79)
```


Motivating Architectural Choice #3

- Delay in spawning containers
 - Container creation
 - Execution code, library shipping
- Slow state sharing
 - Write to/read from backend distributed file system (e.g., HDFS)
 - Sharing state across frameworks

Architectural Choice #3

- Retain container through an abstraction in REEF
- State reuse across heterogeneous frameworks
 - Local context that keeps state across tasks
 - Multiple stackable contexts

Motivating Architectural Choice #4

- Control plane: master-slave pattern
- Not easy to set up the plane
 - manually set up master, workers, and channels between them
 - Make the channels scalable
 - Provide heartbeats

Architectural Choice #4

- Centralized control flow
 - Container allocation & configuration
 - Task configuration & submission
- Centralized error handling
 - Task exceptions are thrown at the master
 - Container failure is reported to the master
- Scalable communication channels
 - Multiplexing events between two endpoints
 - Periodic heartbeats from workers to master
 - Multicast from master to workers

Architectural Choices Summary

- Event-driven programming and engine
- Configuration using dependency injection
- Container and state reuse across heterogeneous frameworks
- Scalable control plane master-worker messaging

REEF Runtime Infrastructure

- Driver Runtime: hosts the application control-flow logic implemented in the Driver module
- Environment Adapter: translate Driver resource actions to the underlying resource manager protocol
 - Local Process, Apache YARN, Apache Mesos, Azure HDInsight
- Wake: asynchronous event processing framework
- Tang: configuration through dependency injection

REEF Runtime Infrastructure

Evaluator

executes application tasks
implemented in the Task module, and
manages application state in the form
of Contexts

Context

Retains state across Task executions

Service

Factors out core functionalities that
can be re-used across a broad range of
applications

REEF Application Framework

Driver

Application code that implements the resource allocation and task scheduling components.


1. Runtime Events
2. Evaluator Events
3. Task Events


Task






Application code executed within an Evaluator.

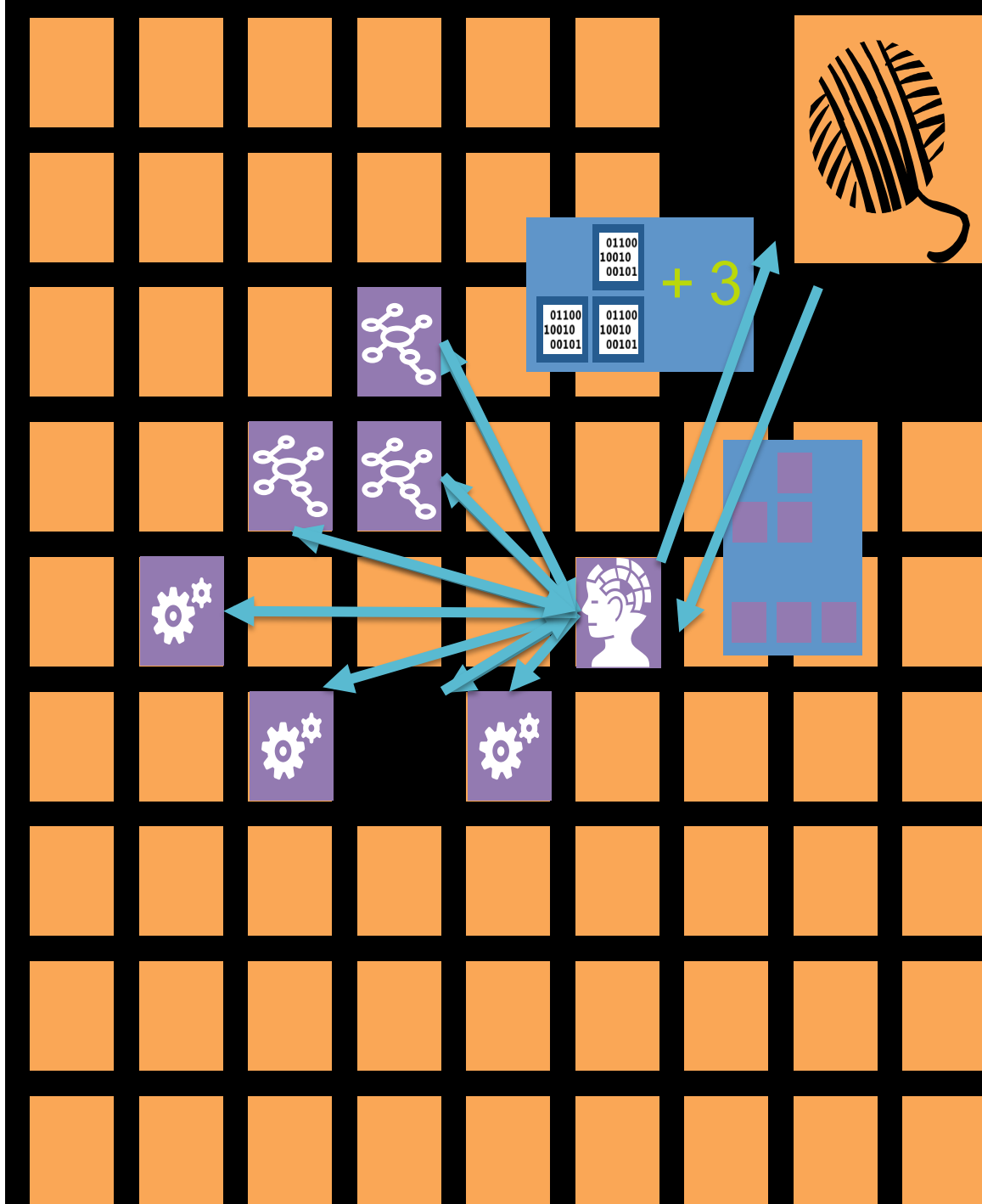
The application-supplied Task implementation has access to its configuration parameters and the Evaluator state, which is exposed as Contexts

REEF Control Flow

Yarn () handles resource management (security, quotas, priorities)

Per-job REEF Drivers () request resources, coordinate computations, and handle events: faults, preemption, etc...

REEF Evaluators () hold hardware resources, allowing multiple REEF Tasks ( ,  ,  ,  , etc.) to use the same cached state through REEF Contexts.

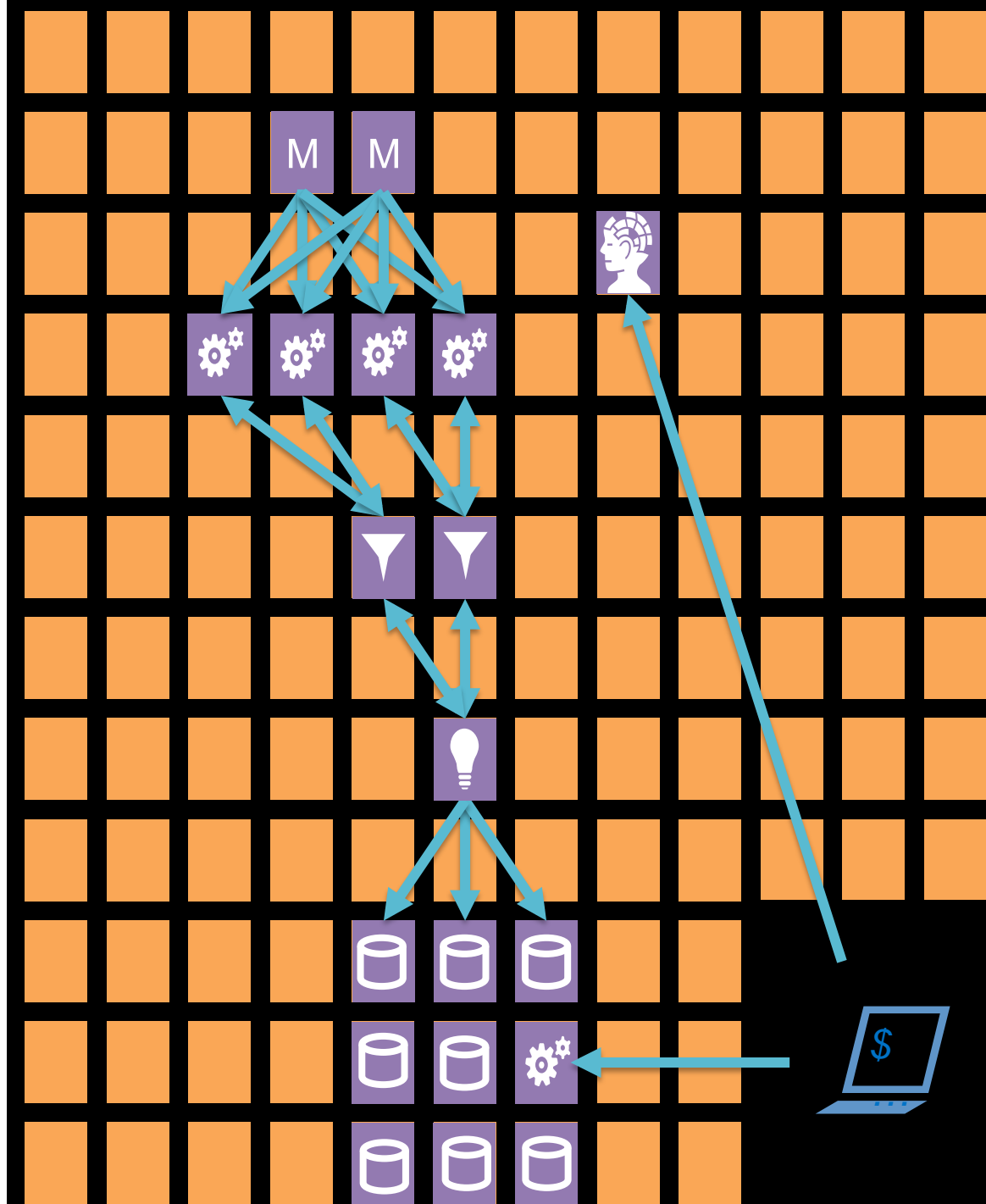


Retained Evaluator & Context

Handover of pre-partitioned
and parsed data between
frameworks

Iterative computation

Interactive queries

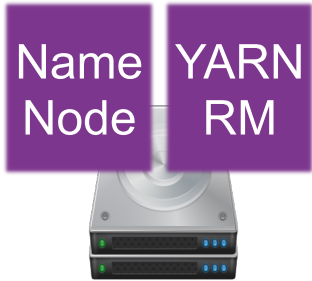


REEF Control Flow: Job Life Cycle



Client

submit job

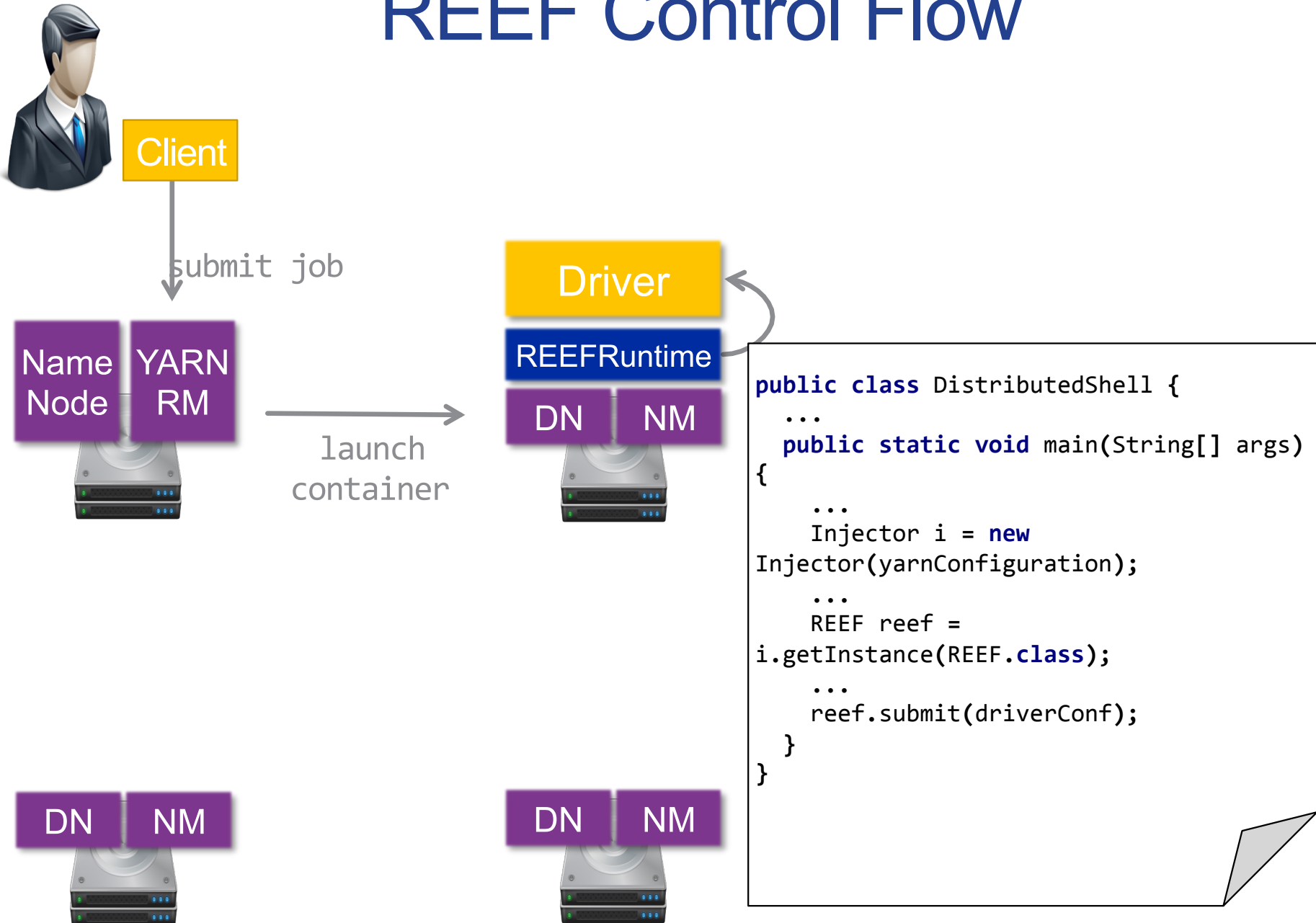


launch
container



```
public class DistributedShell {  
    ...  
    public static void main(String[] args)  
    {  
        ...  
        Injector i = new  
        Injector(yarnConfiguration);  
        ...  
        REEF reef =  
        i.getInstance(REEF.class);  
        reef.submit(driverConf);  
    }  
}
```

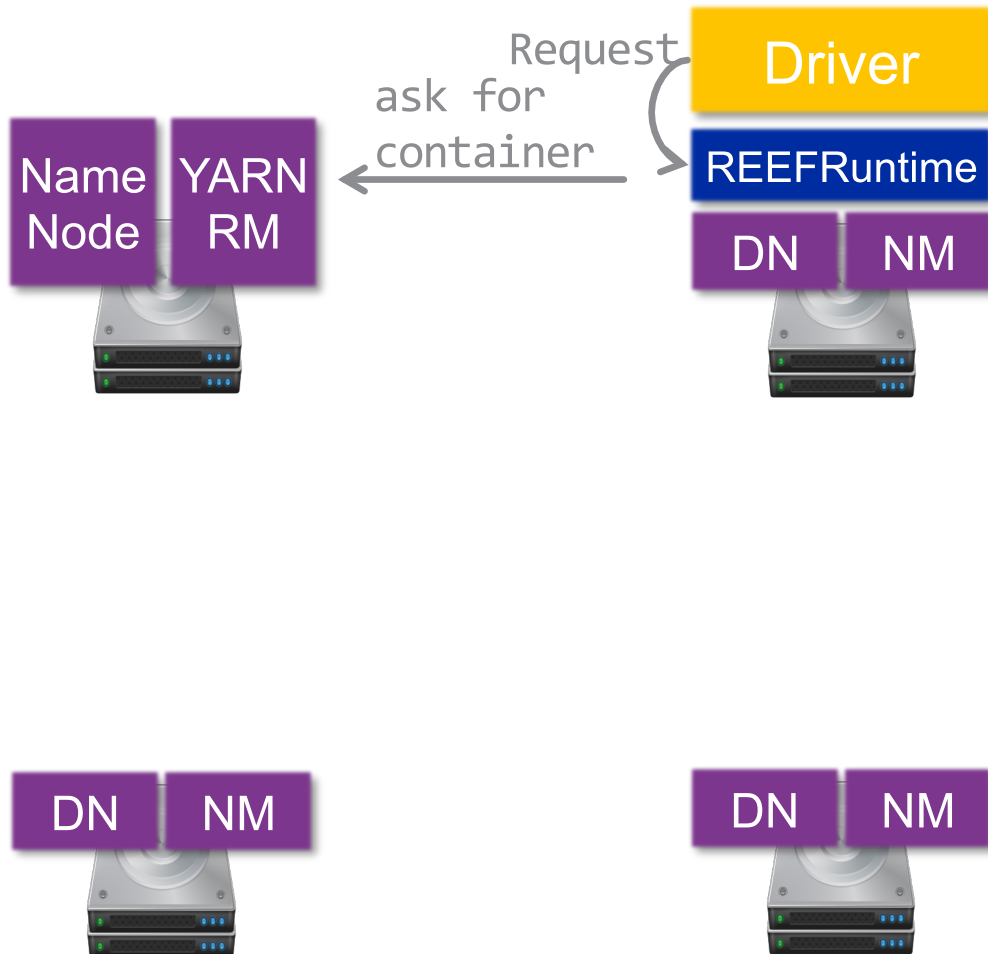

REEF Control Flow



REEF Control Flow



Client

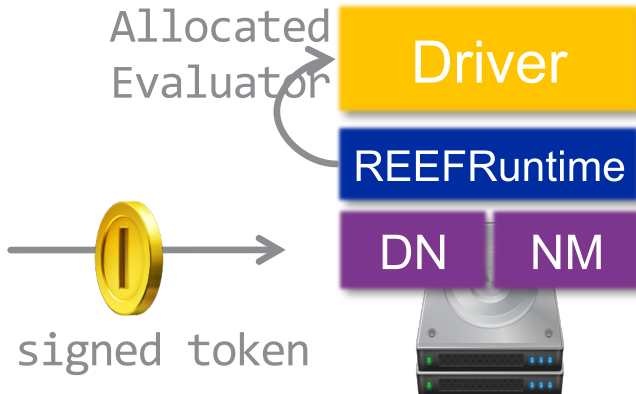
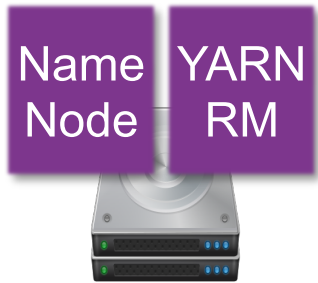


```
public class DistributedShellJobDriver {  
    private final EvaluatorRequestor  
    requestor;  
    ...  
  
    public void onNext(StartTime time) {  
  
        requestor.submit(  
            EvaluatorRequest.Builder()  
                .setSize(SMALL).setNumber(2)  
                .build()  
        );  
  
    }  
  
    ...  
}
```


REEF Control Flow



Client

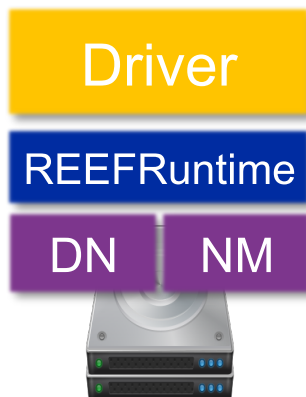
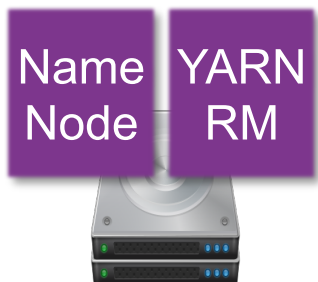


```
public class DistributedShellJobDriver {  
    private final EvaluatorRequestor  
    requestor;  
    ...  
    public void onNext(AllocatedEvaluator  
    eval) {  
        Configuration contextConf = ...;  
        eval.submitContext(contextConf)  
    }  
    ...  
}
```


REEF Control Flow



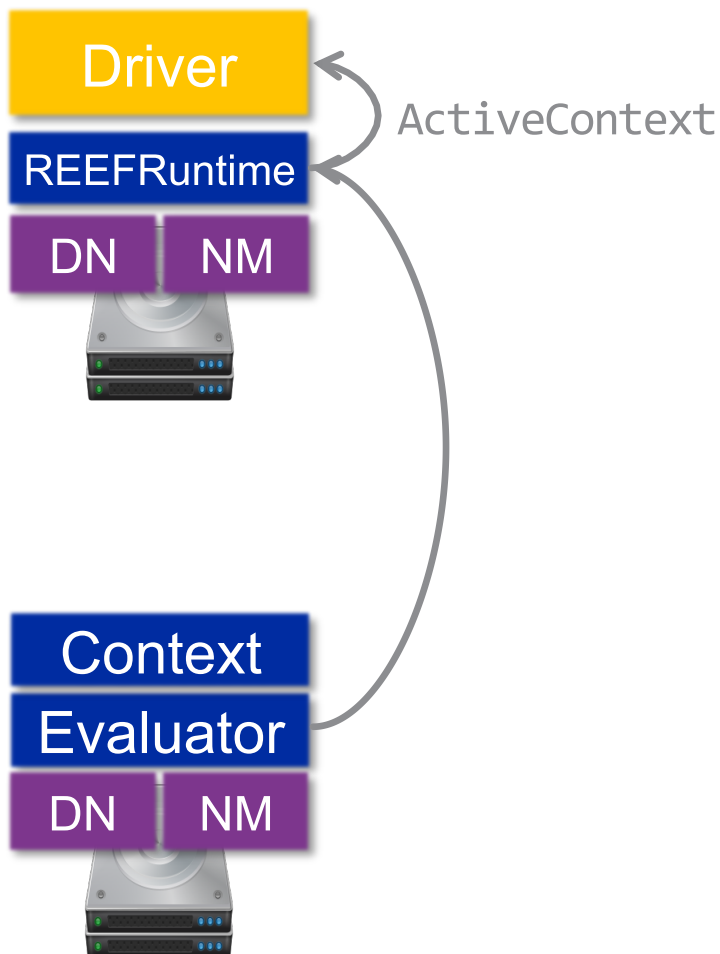
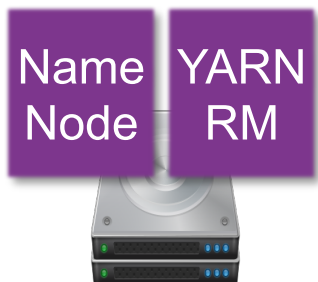
Client



REEF Control Flow



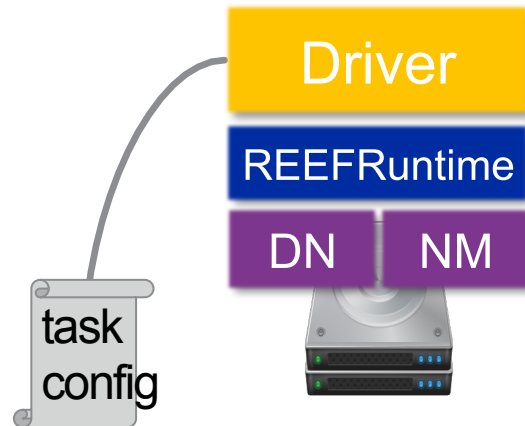
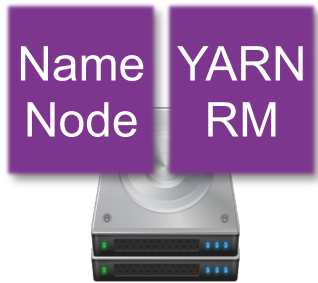
Client



REEF Control Flow



Client

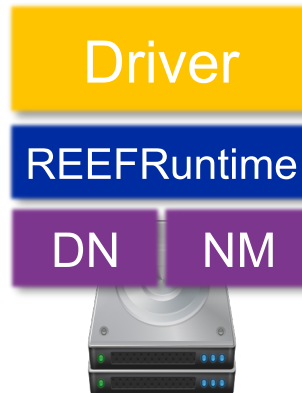
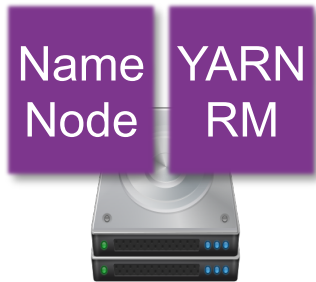


```
public class DistributedShellJobDriver {  
    private final String cmd = "dir";  
  
    [...]  
  
    public void onNext(ActiveContext ctx) {  
        final String taskId = [...];  
  
        Configuration taskConf = Task.CONF  
            .set(IDENTIFIER, "ShellTask")  
            .set(TASK, ShellTask.class)  
            .set(COMMAND, this.cmd)  
            .build();  
  
        ctx.submitTask(taskConf);  
    }  
  
    [...]  
}
```


REEF Control Flow



Client

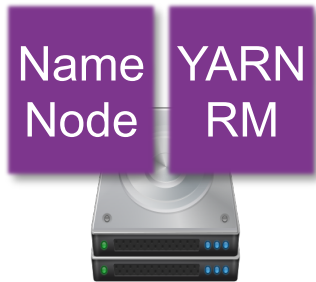


```
class ShellTask implements Task {  
  
    private final String command;  
  
    @Inject  
    ShellTask(@Parameter(Command.class)  
String c) {  
        this.command = c;  
    }  
  
    private String exec(final String  
command){  
        ...  
    }  
  
    @Override  
    public byte[] call(byte[] memento) {  
        String s = exec(this.cmd);  
        return s.getBytes();  
    }  
}
```


REEF Control Flow



Client

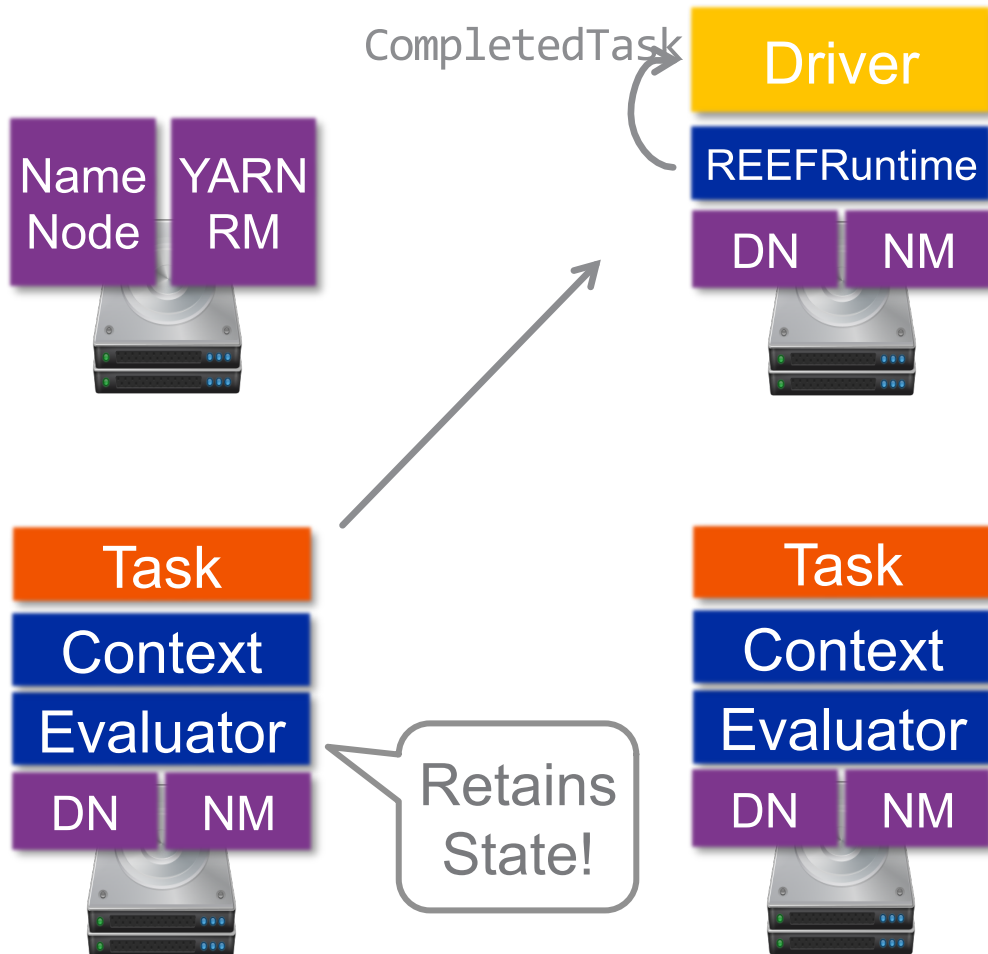


heartbeat()

REEF Control Flow



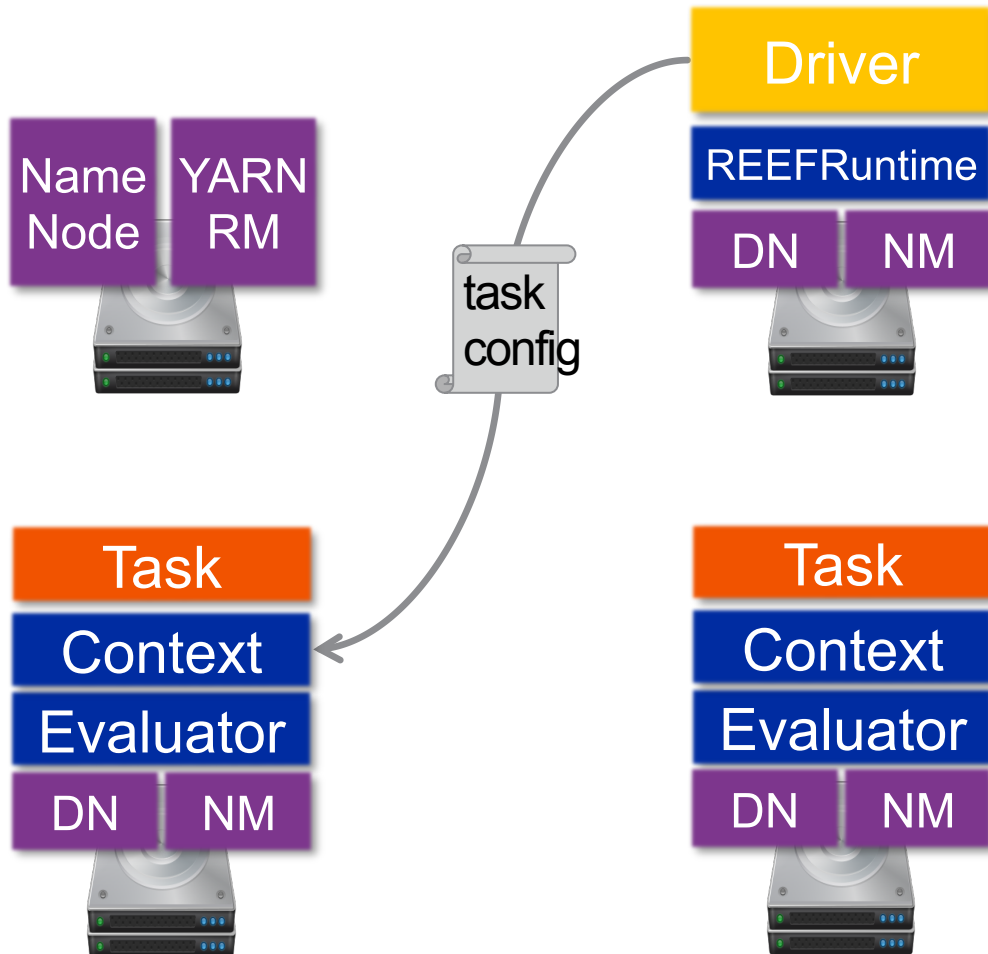
Client



REEF Control Flow



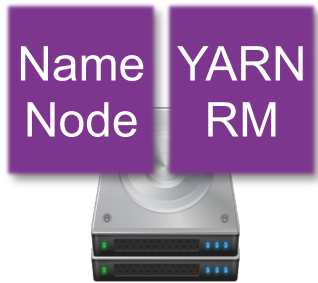
Client



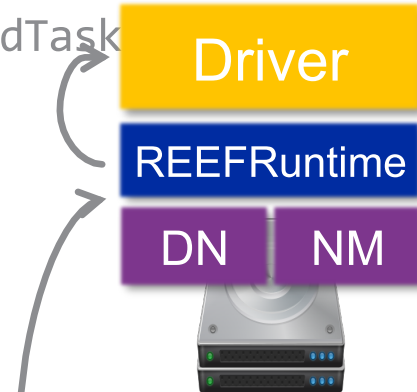
REEF Control Flow



Client



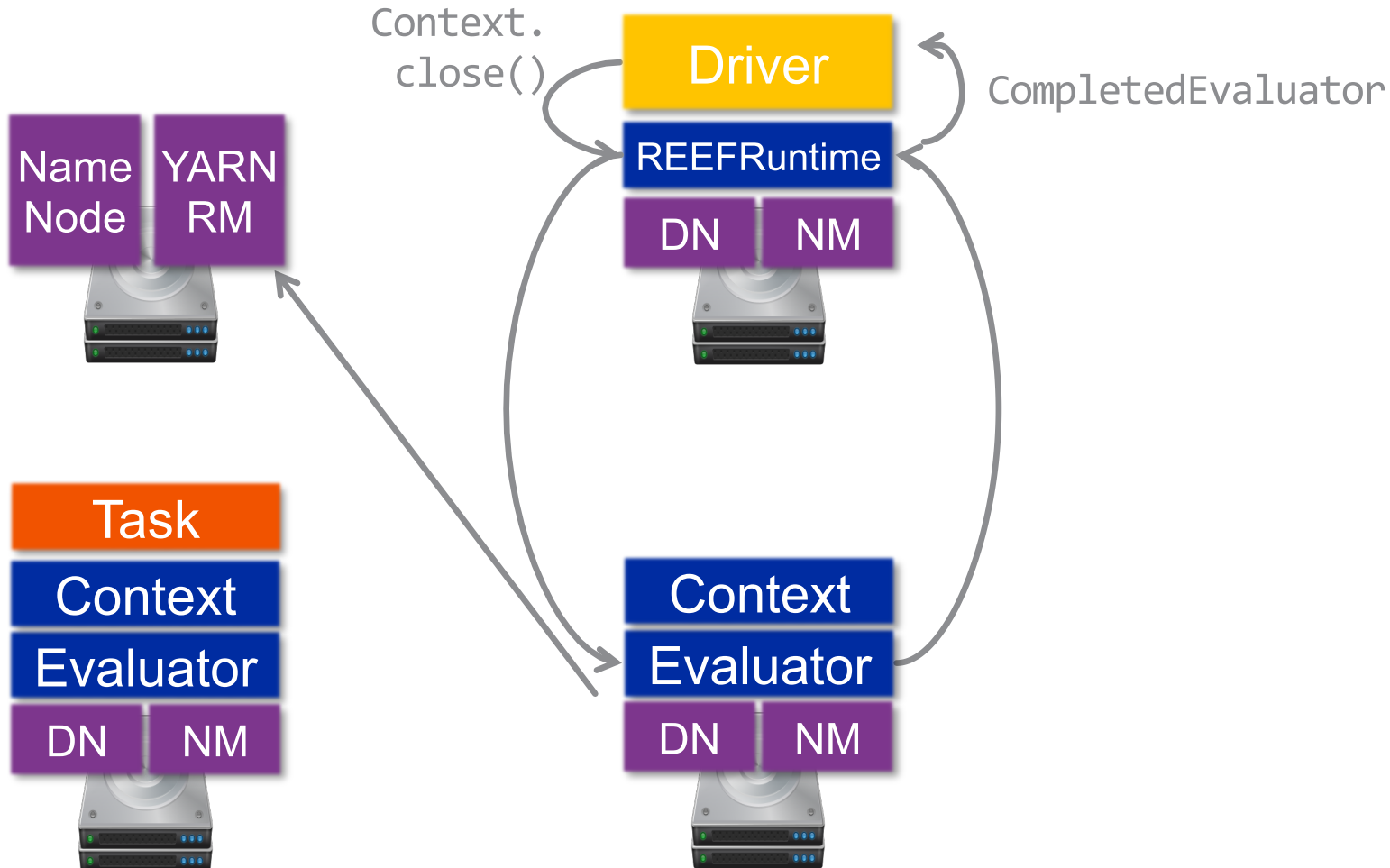
CompletedTask



REEF Control Flow



Client



Current REEF Services

Storage

Abstractions: Map and Spool
Local and Remote

Network

Name-based data passing
Collective communication

State Management

Checkpointing
Task suspend-resume

Adding Blocked Time Analysis to REEF

- Making Sense of Performance in Data Analytics Frameworks (NSDI 2015): Spark
 - Network optimizations can reduce job completion time by at most 2%
 - CPU (not I/O) often the bottleneck
 - 19% reduction in completion time from optimizing disk
 - Many straggler causes can be identified and fixed
- Importance of adding instrumentation for blocked time analysis to understand how to best focus on performance improvements

REEF Implementation

	C#	Java	CPP	Total
Tang	10,567	6,976	0	17,543
Wake	7,749	4,681	0	12,430
REEF	13,136	15,118	1,854	30,108
Services	0	5,319	0	5,319
Total	31,452	32,094	1,854	65,400

Lines of code by component and language

REEF Status

- Open-source Apache Incubator project (3rd Apache incubation from Korea) since August 12, 2014
- 22 committers; 7 PMC members from CMSLab
- Release 0.10.0
- <http://reef.apache.incubator.org>



General

- [Welcome](#)
- [Incubation Overview](#)
- [Incubation Policy](#)
- [Incubation Guides](#)
- [Roles and Responsibilities](#)
- [General FAQ](#)
- [Incubator Wiki](#)

▶ REEF Project Incubation Status

This page tracks the project status, incubator-wise. For more general project status, look on the project website.

▶ Description

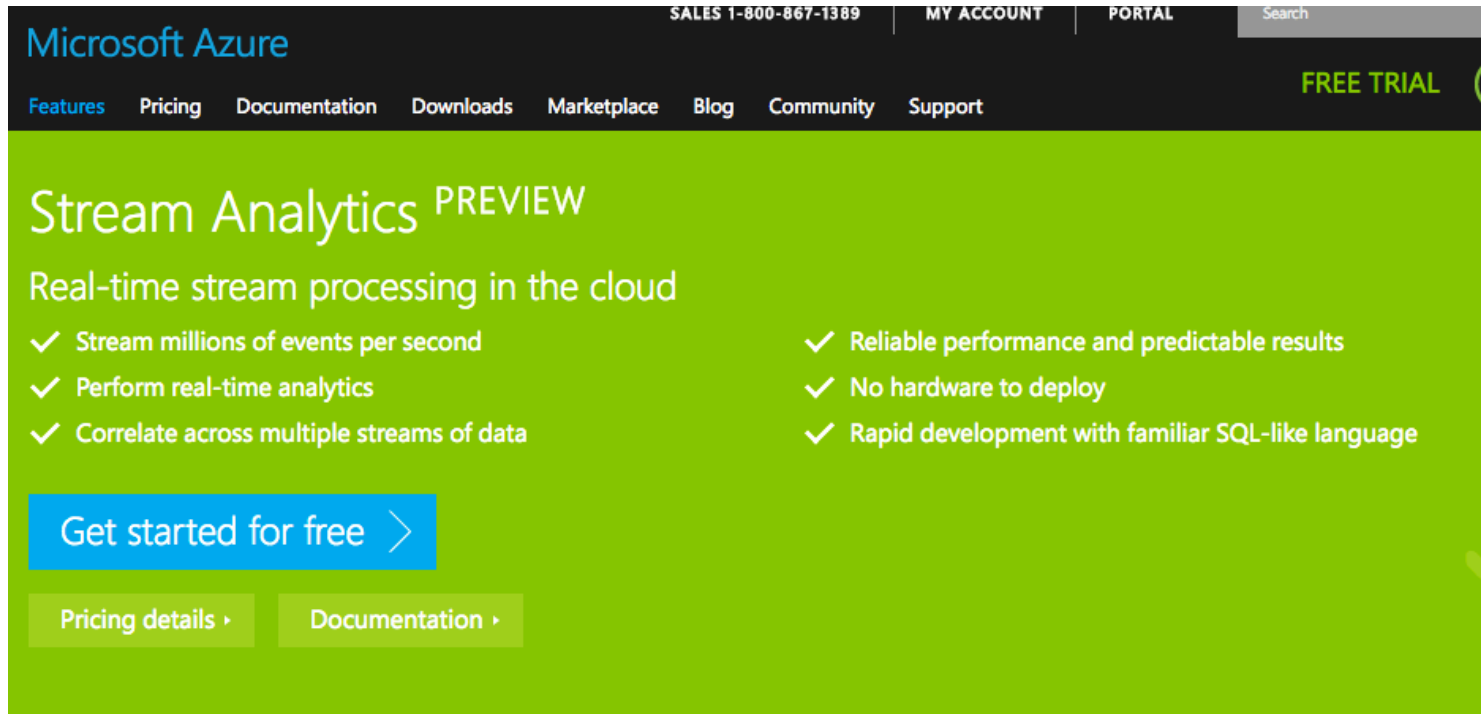
REEF (Retainable Evaluator Execution Framework) is a scale-out computing fabric that eases the development of Big Data applications on top of resource managers such as Apache YARN and Mesos.

▶ News

- 2014-08-12 Project enters incubation.

Applications

MS Azure Stream Analytics Powered by REEF



The screenshot shows the Microsoft Azure Stream Analytics website. The top navigation bar includes links for Features, Pricing, Documentation, Downloads, Marketplace, Blog, Community, and Support. A 'FREE TRIAL' button is visible in the top right. The main heading is 'Stream Analytics PREVIEW' with the subtitle 'Real-time stream processing in the cloud'. Below this, there are two columns of features, each starting with a checkmark. A large blue button labeled 'Get started for free' is positioned below the features. At the bottom, there are two smaller buttons: 'Pricing details' and 'Documentation'.

Microsoft Azure

SALES 1-800-867-1389 | MY ACCOUNT | PORTAL | Search

Features Pricing Documentation Downloads Marketplace Blog Community Support

FREE TRIAL

Stream Analytics ^{PREVIEW}

Real-time stream processing in the cloud

- ✓ Stream millions of events per second
- ✓ Perform real-time analytics
- ✓ Correlate across multiple streams of data
- ✓ Reliable performance and predictable results
- ✓ No hardware to deploy
- ✓ Rapid development with familiar SQL-like language

[Get started for free >](#)

[Pricing details >](#) [Documentation >](#)

Real-time business insights

Stream Analytics is an event processing engine that helps uncover real-time insights from devices, sensors, infrastructure, applications and data. It will enable various opportunities including Internet of Things (IoT) scenarios such as real-time fleet management or gaining insights from devices like mobile phones or connected cars.



CORFU on REEF (VMWare)

- CORFU (NSDI 2012): a distributed logging tool providing applications with consistency and transactional services at extremely high throughput
- CORFU master deployed in a Driver
 - A logging unit fails, the Driver is informed, and the CORFU master reacts
- REEF Service for triggering checkpointing and restarting a Driver from a checkpoint
- REEF decouples CORFU's resource deployment from its state, allowing CORFU to be elastic

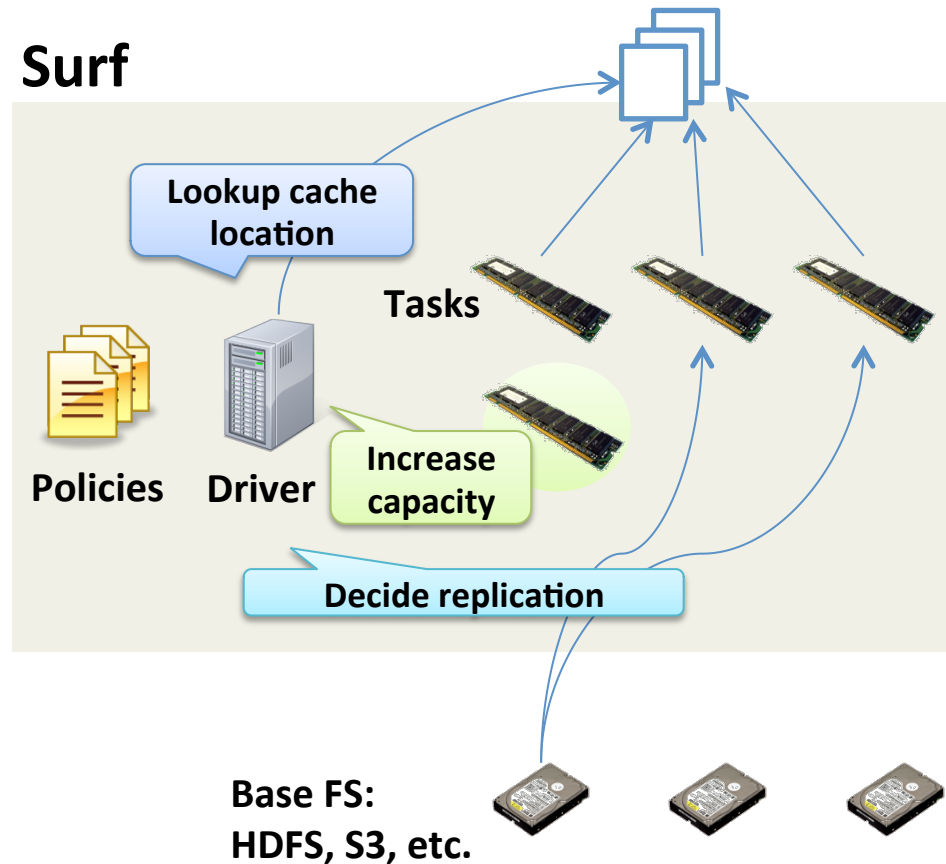
CORFU on REEF

- Using CORFU from REEF: a CORFU log may be used from other REEF jobs by linking with a CORFU client-side library
- CORFU as a REEF service: CORFU and application event handlers jointly implement the control flow of the application

Surf: In-Memory Store for Big Data Analytics (SNU, SKT)

- In-memory distributed caching tier on REEF
 - Flexible caching policies
 - Elasticity
 - Easy read access across frameworks and base file systems

Computation Frameworks:
Hadoop MapReduce, Hive, Spark, etc.



Surf: Interfaces

- Transparent backend FS-compatible Read/Write Interface
 - Address with surf://
 - Compatible with existing frameworks
- Consistent addressing even on restart
 - surf://yarn.reef-job-InMemory, instead of surf://host:port
- Command-Line Interface
 - Preload data
 - Manage policies
 - Replication, Pinning, Write, Elasticity
 - Manage caches

Elastic Machine Learning (SNU, MS)

- Elastic group communication in REEF
 - Scatter, gather, broadcast, reduce, ...
 - Elastically add/remove nodes
 - Handle faults at the task level
- Elastic memory
 - Elastically change memory resources for in-memory big data analytics
 - Solves the problem of the static allocation of resources for low-latency interactive services, machine learning, etc.
- Elastic ML runtime & optimizer
- A new breed of elastic Machine Learning algorithms

The Applications Described Underscore REEF's Power

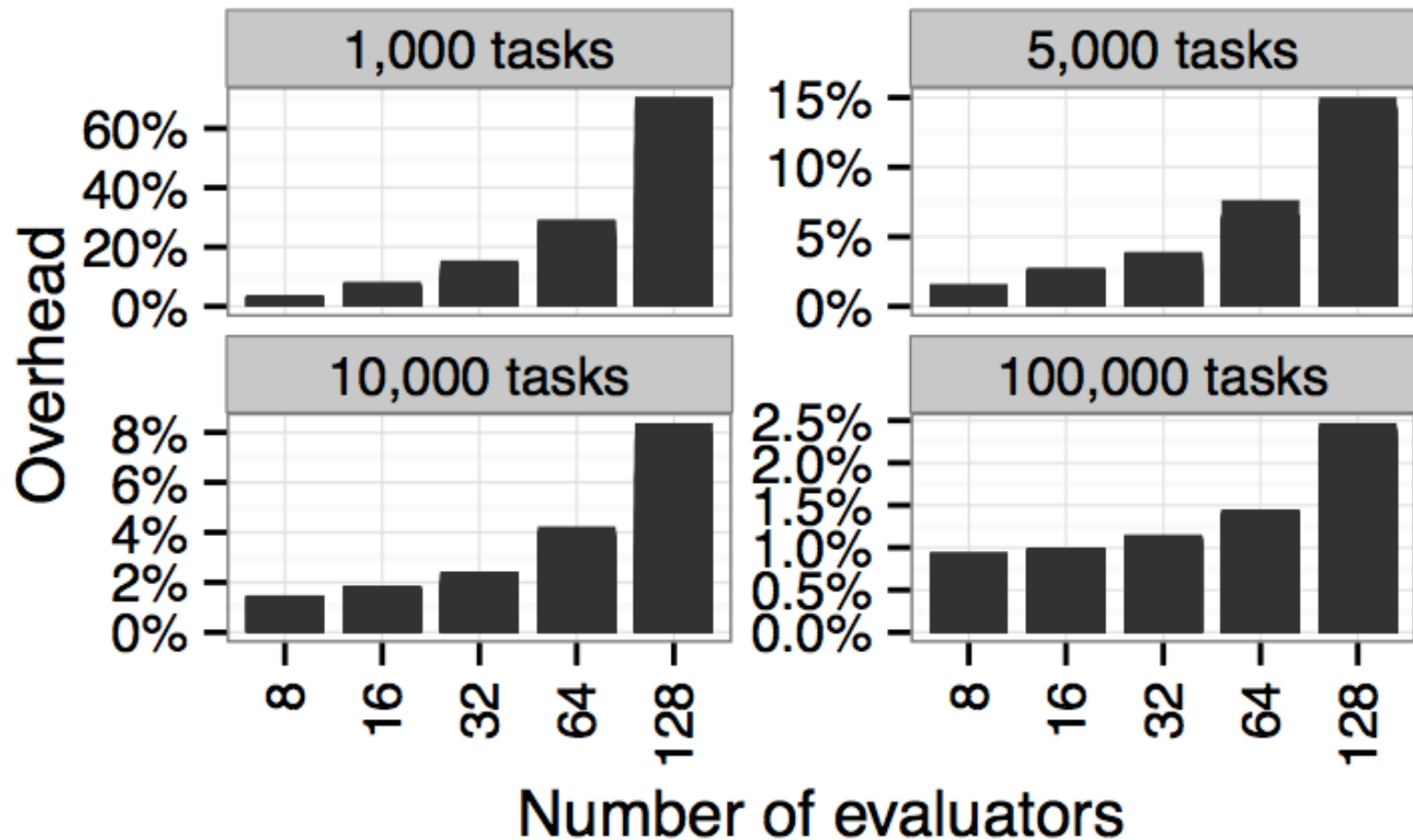
- A flexible framework for developing distributed applications on Resource Manager services
- A standard library of reusable system components that can be easily composed (via Tang) into application logic

REEF in Action

Cluster Setup

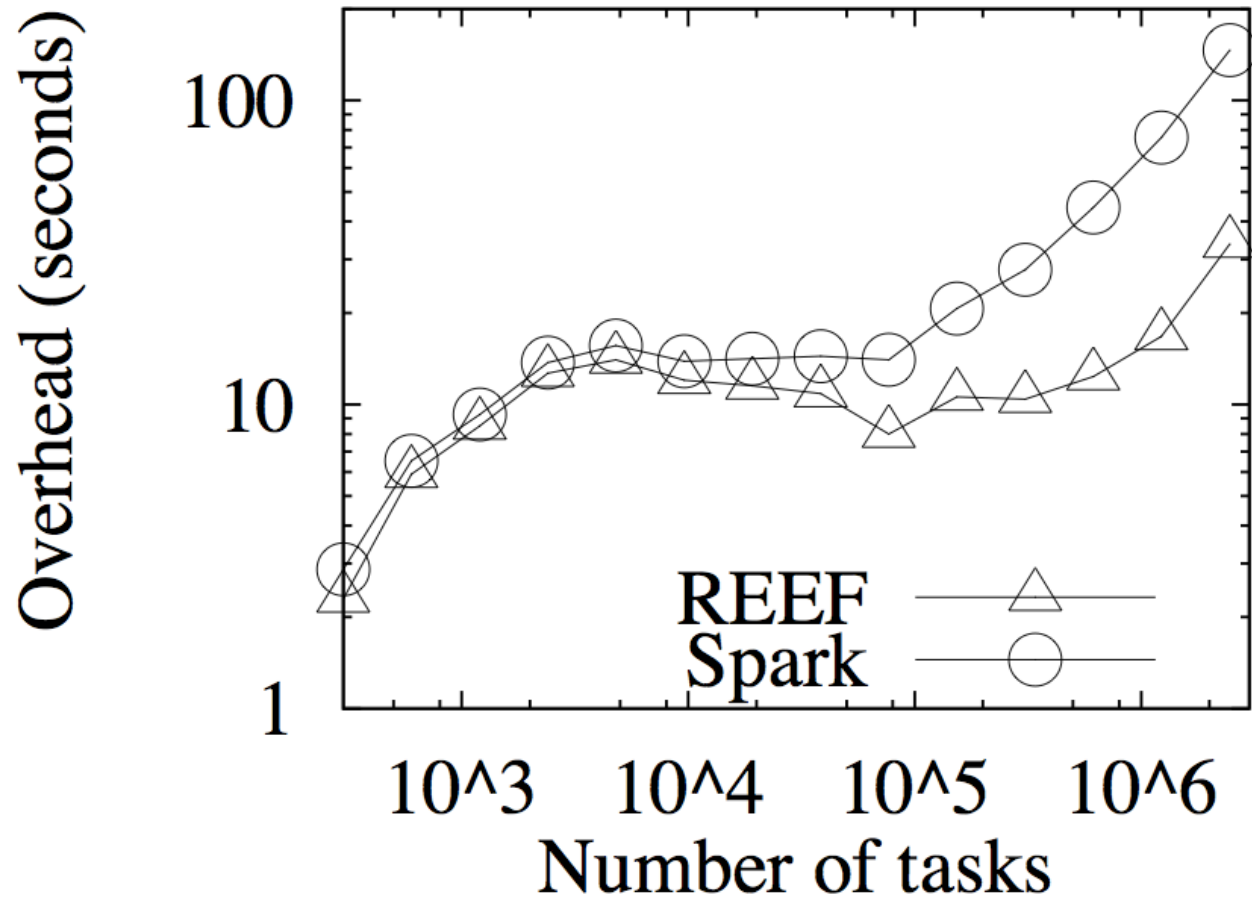
- YARN version 2.6 running on a cluster of 35 machines equipped with 128GB of RAM and 32 cores
- Microsoft Azure to allocate 25 D4 instances (8 cores, 28 GB of RAM and 400 GB of SSD disk each): Overheads of REEF vs. Spark

Combined (REEF + YARN) Overheads for Jobs with Short-lived (1s) Tasks

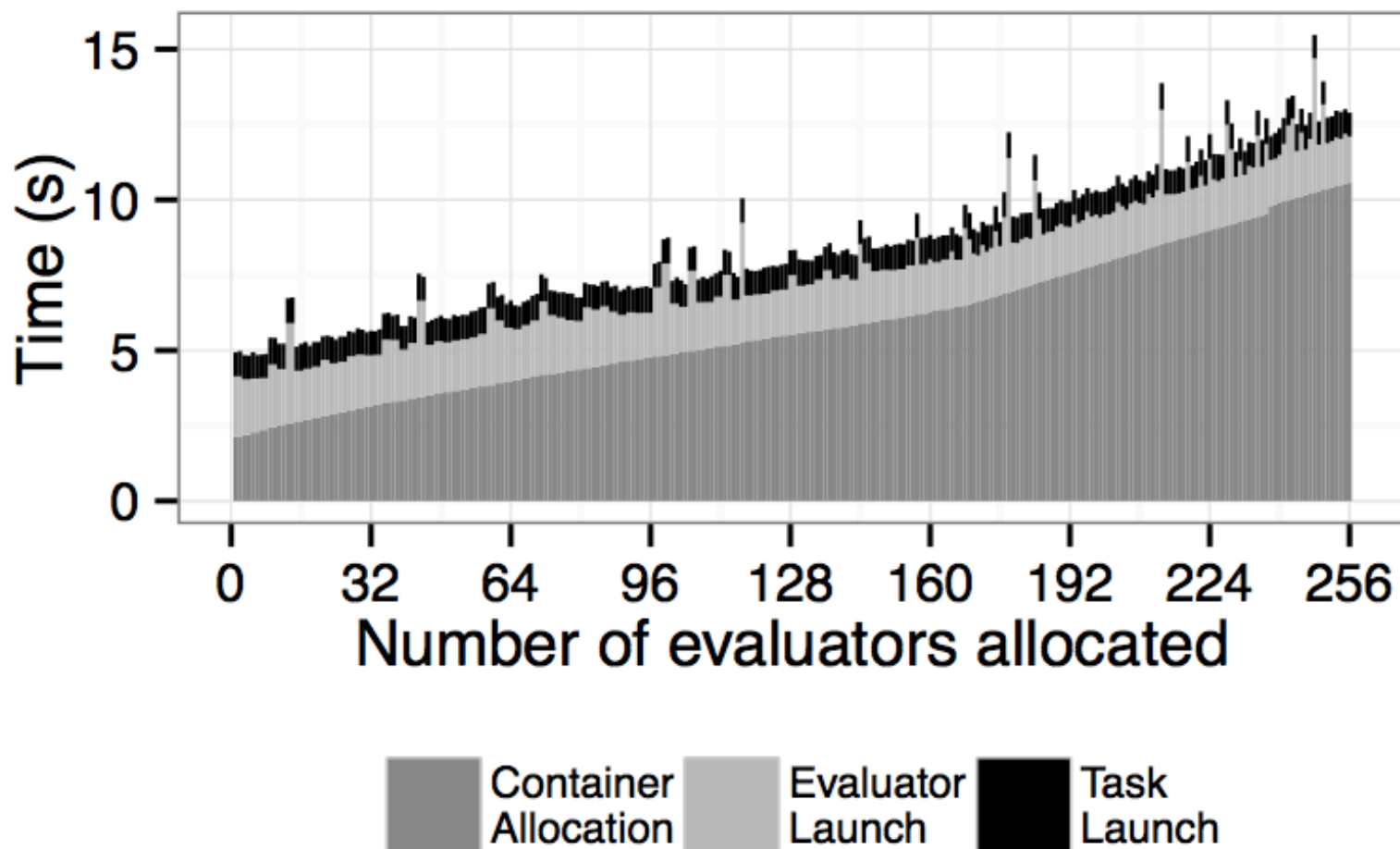


$$\text{Overhead} = \text{actual_runtime} / \text{ideal_runtime} - 1$$

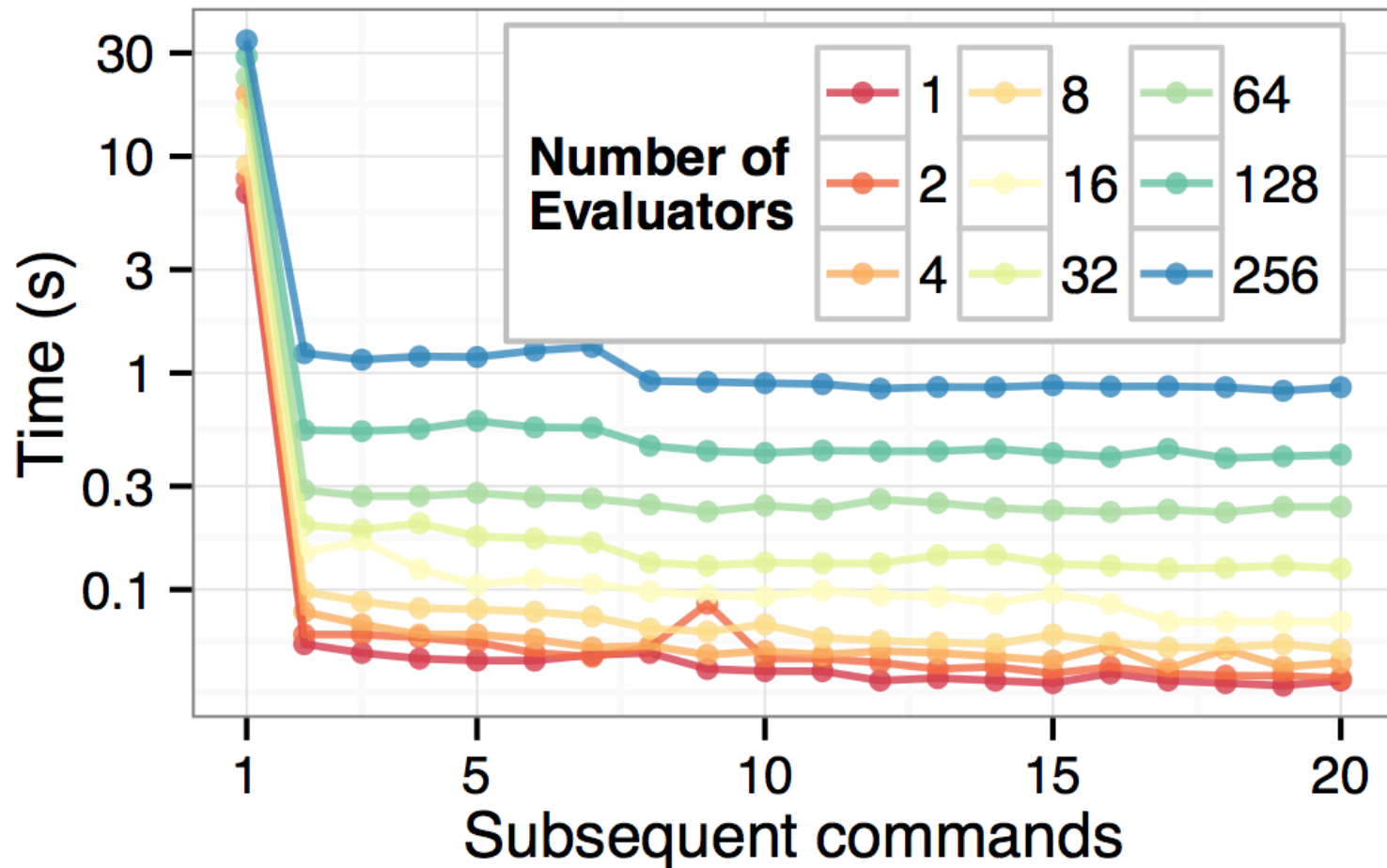
Overhead of REEF and Spark for Jobs with Short-lived Tasks (100ms)



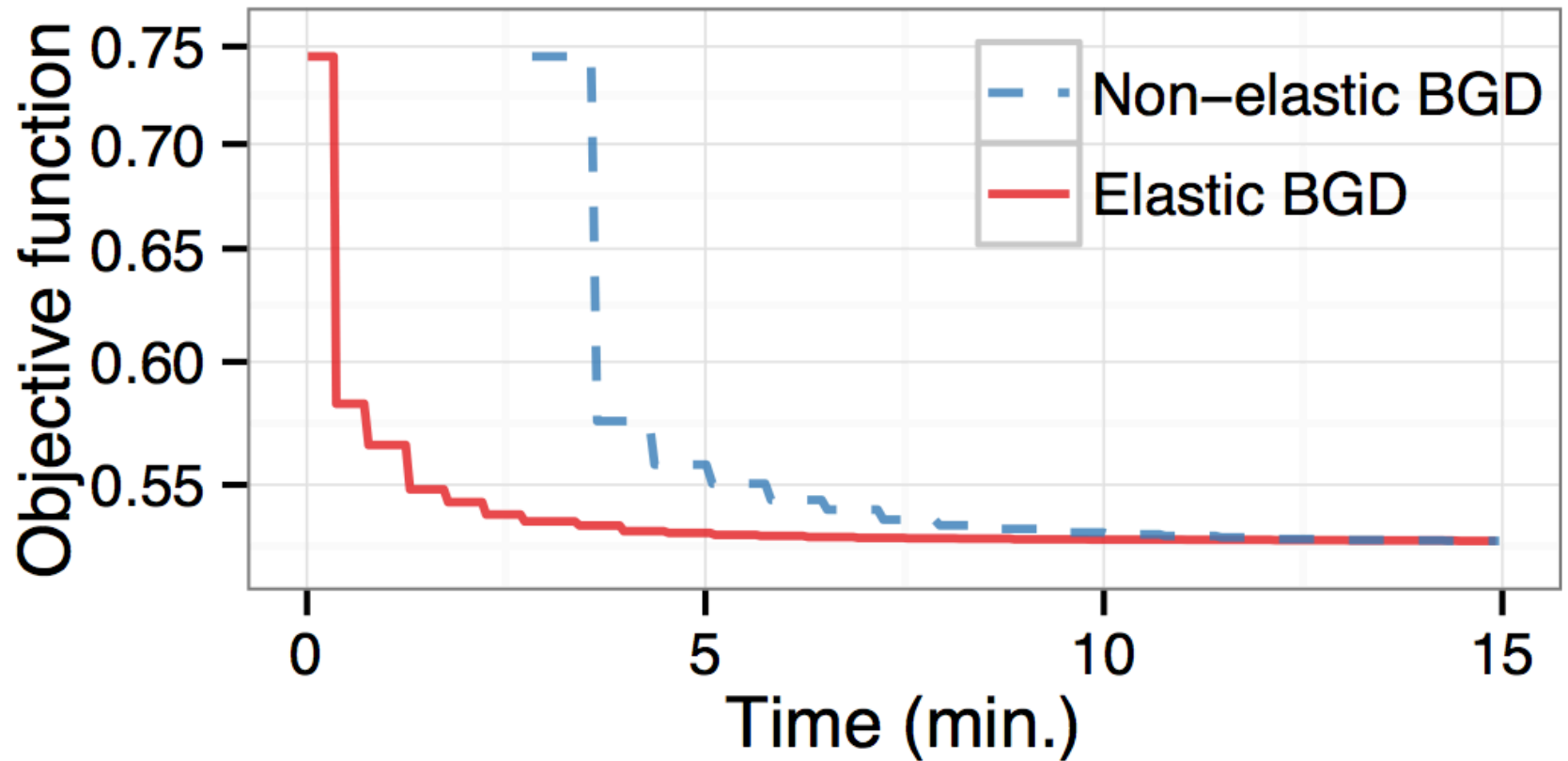
Evaluator/Task Allocation and Launch Time Breakdown



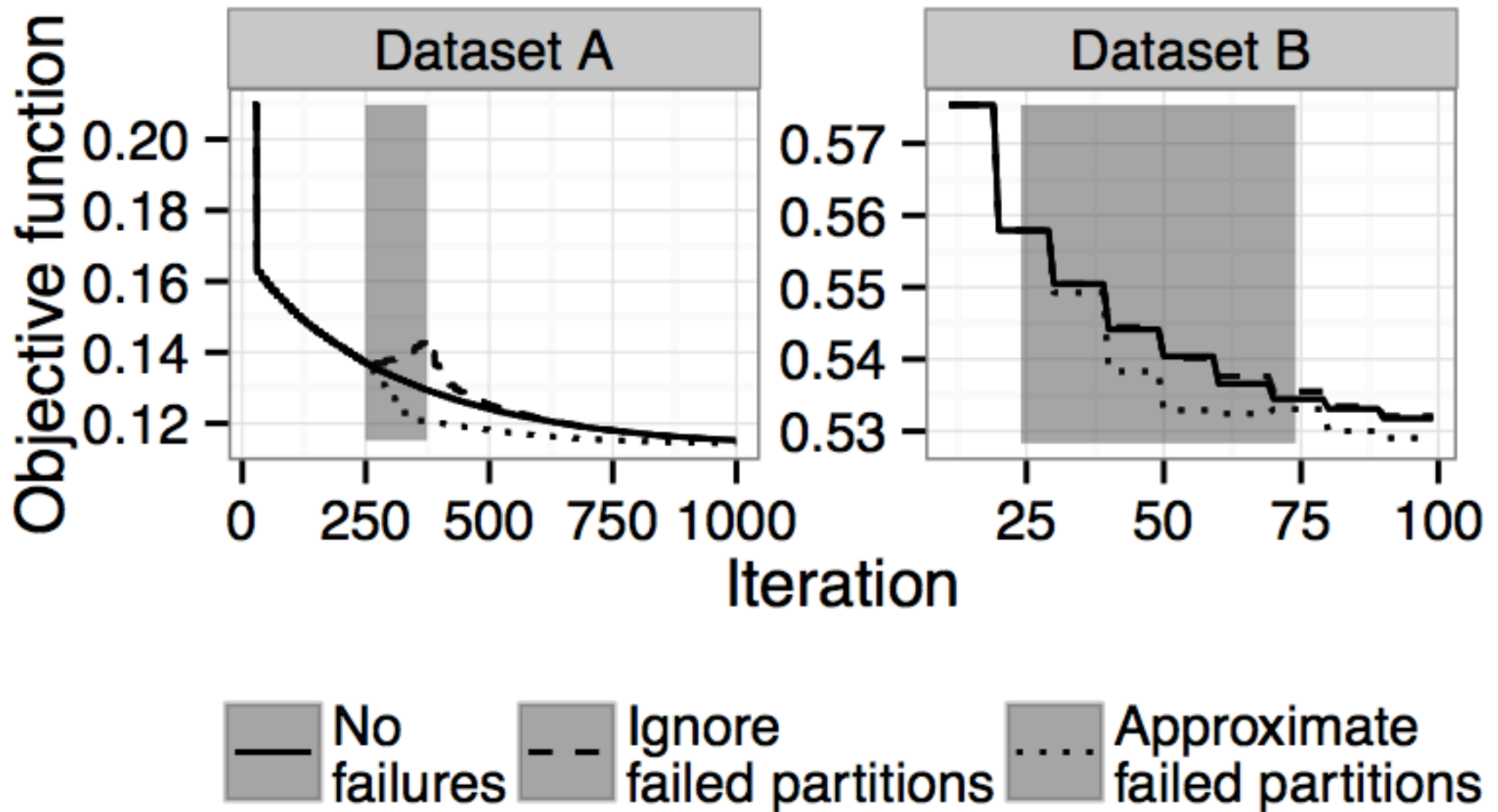
Interactive Distributed Shell



Quick Ramp Up Thanks to Elastic Machine Learning



Learning Progress over Iterations with Faulty Partitions

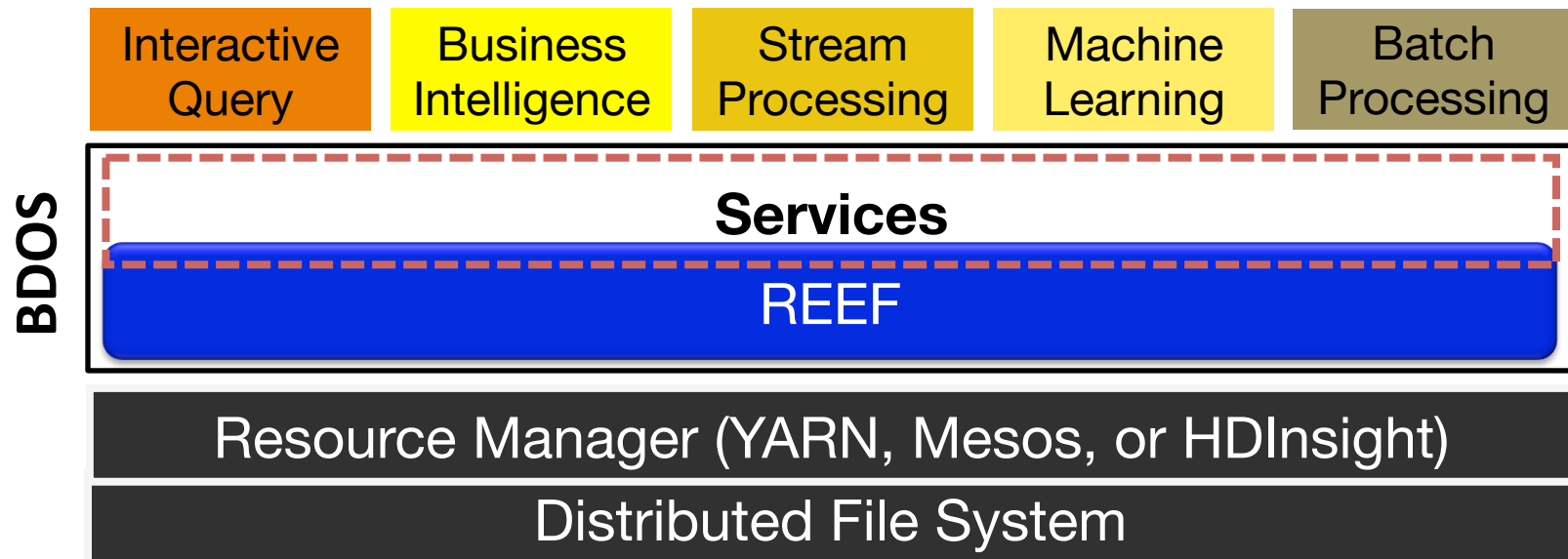


What's Next?

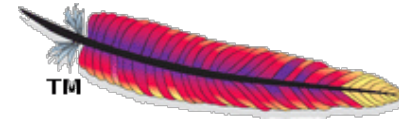
What's Next?

- Create a high-performance Big Data stack based on REEF
 - Big data operating system
 - Exciting infrastructure services (e.g. Elastic Memory (HotOS 2015))
 - Unified data management
 - Programming layer
- Grow the Apache REEF incubator project to become a TLP project

REEF: Towards a Big Data Operating System



- ✓ Reusable control plane for coordinating data plane tasks
- ✓ Virtualization of resource managers
- ✓ Container and state reuse across tasks from heterogeneous frameworks
- ✓ Simple configuration management
- ✓ Scalable event handling



THANK YOU!

<http://reef.incubator.apache.org>

Contact: Byung-Gon Chun
bgchun@snu.ac.kr