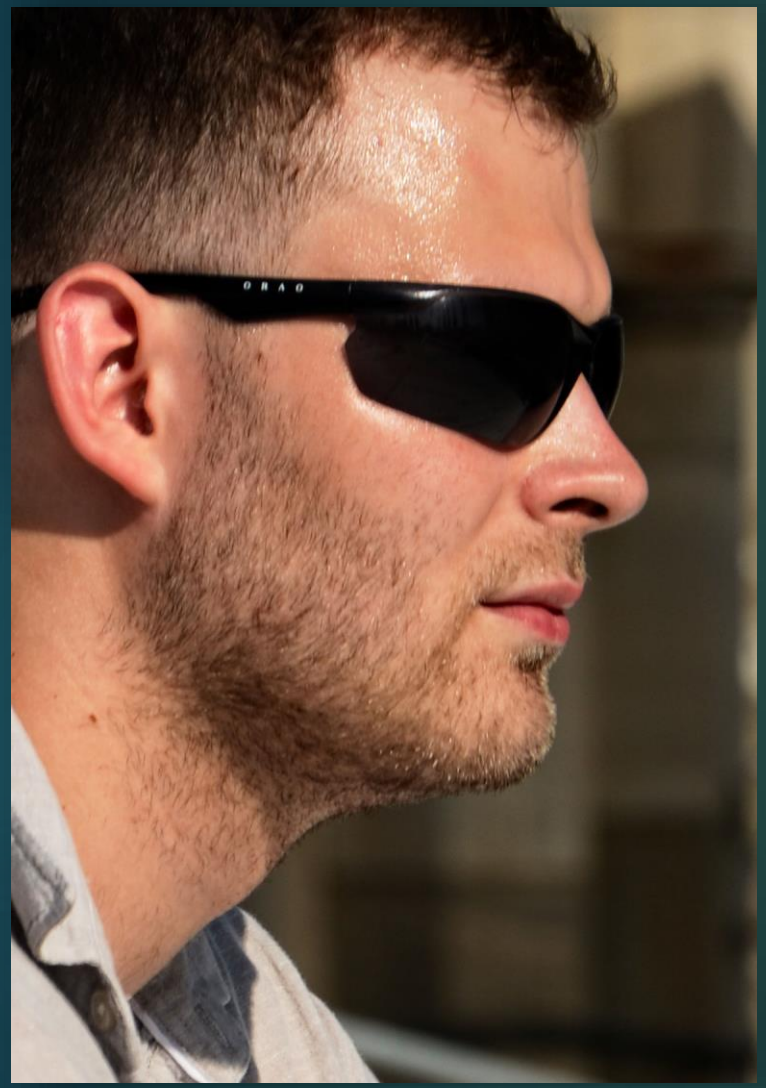
Serenity

MESOS OVERSUBSCRIPTION MODULE



Szymon Konefał

SOFTWARE ENGINEER

INTEL CORPORATION

Agenda

- Oversubscription Basics
- Oversubscription in Mesos
- Serenity Architecture
- Next steps for Serenity & Mesos

Oversubscription Basics

OVERSUBSCRIPTION FROM MESOS PERSPECTIVE

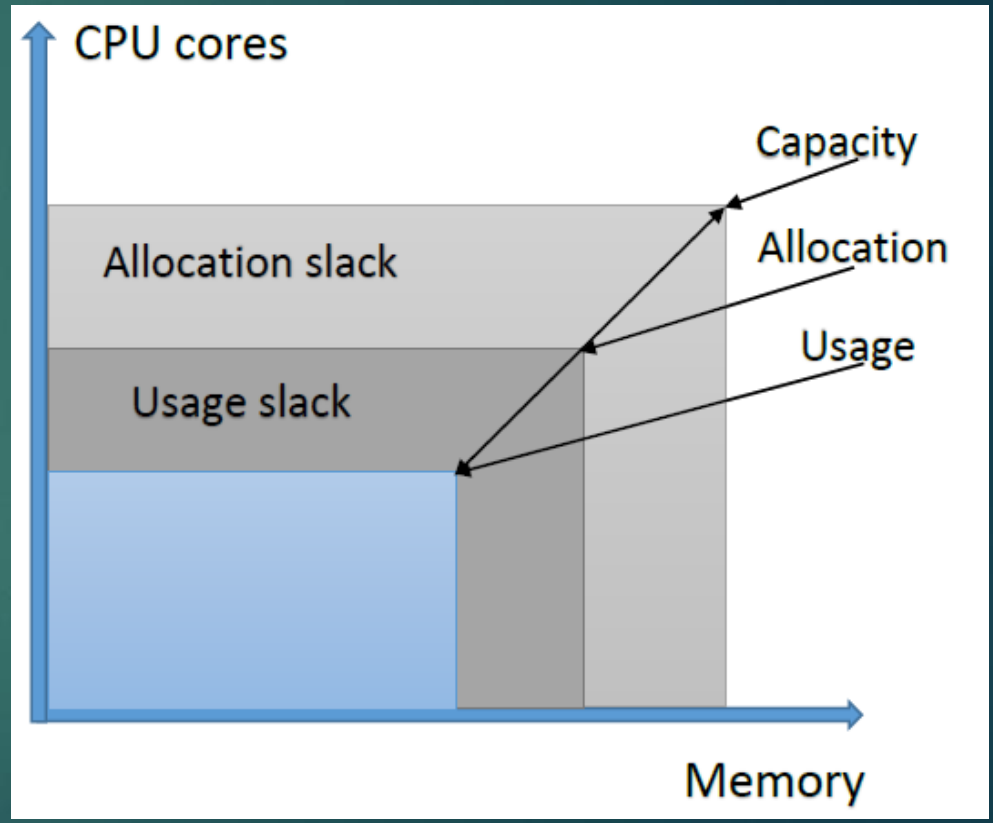
Oversubscription Basics

- Recycling of **reserved** but **unused** resources
- Spinning up **revocable** („best effort") tasks
- Throttle or revoke **BE** tasks when production task needs more resources (Quality of Service)
- **Goal: Increase overall data center utilization**

Oversubscription Basics

RESOURCE ESTIMATOR & BEST EFFORT TASKS

- Exposes Slack Resources to Mesos Agent, who passes them to allocator
- Allocator offers Slack Resources to Frameworks
- Frameworks which are registered as consumers of oversubscribed resources can reserve them
- Jobs running on slack resources are considered „revocable”



Oversubscription Basics

Quality of Service & Task throttling and revocation

- Throttle best effort tasks when production task needs more of it's isolated compressible resource, eg. **cpu time**
- Revoke best effort tasks when production task needs more of a **shared resource** or non-compressible one
  - Competition for shared resource is considered a „noisy neighbour" situation
  - Shared resources examples:
    - L3 CPU cache*
    - Memory bandwith

* Actually you can isolate that using Intel Cache Allocation Technology ;-)

Oversubscription Modules

POWERED BY YOU

Mesos Oversubscription API

- Introduced in Mesos 0.23.0
- Defines Resource Estimator and Quality of Service controller
  - Mesos is shipped with fixed RE and stubbed QoS controller
- You are expected to provide your own modules, if you want to use oversubscription features

Mesos Oversubscription API

RESOURCE ESTIMATOR

```
class ResourceEstimator
{
public:
  virtual Try<Nothing> initialize(
      const lambda::function<process::Future<ResourceUsage>()>& usage) = 0;
  virtual process::Future<Resources> oversubscribable() = 0;
};
```

Mesos Oversubscription API

QoS Controller

```
class QoSController
{
public:
  virtual Try<Nothing> initialize(
      const lambda::function<process::Future<ResourceUsage>()>& usage) = 0;
  virtual process::Future<std::list<QoSCorrection>> corrections() = 0;
};
```

Mesos Oversubscription API

Framework

- Framework needs to register with REVOCABLE_RESOURCES capability set

```
FrameworkInfo framework;
framework.set_name("Revocable framework");

framework.add_capabilities()->set_type(
    FrameworkInfo::Capability::REVOCABLE_RESOURCES);
```

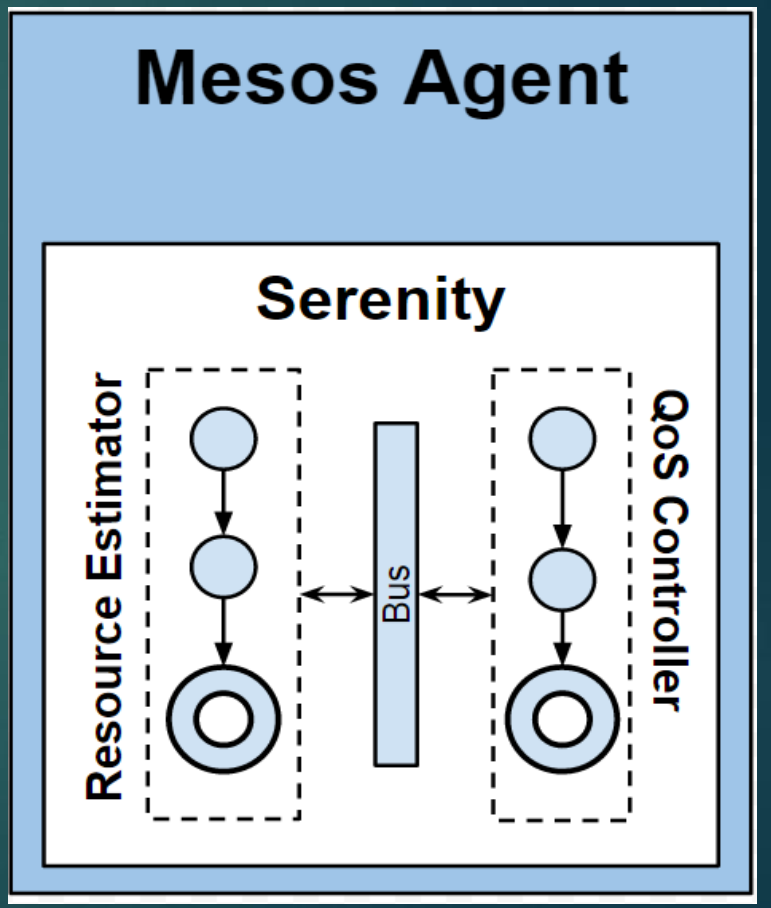
Serenity Architecture

POWER OVERWHELMING

Serenity Architecture

- Flexible solution with interchangeable components
- Estimation and correction is done in pipeline approach
- Filters inside pipelines smoothen, shape and transforms the input
- Open source on Github

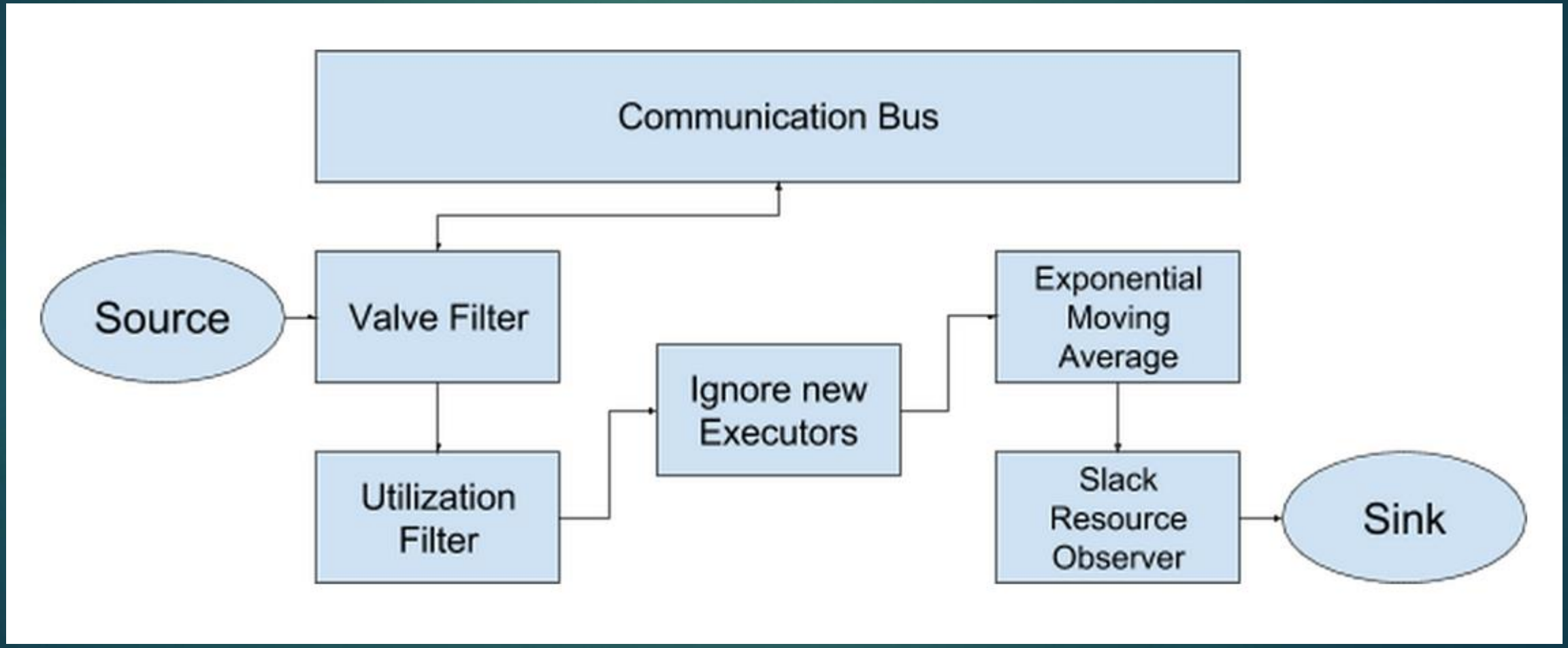
https://github.com/mesosphere/serenity



Serenity Architecture

- Pipeline can consists of different components:
  - Input smoothing: **Exponential Moving Average filter**
  - Input shaping: **PR-executor pass filter, Ignore new executors**
  - Interference signal indicator: **Changepoint detector**
  - Flow control: **Valve filter, Utilization threshold**
  - **Slack Resource Estimator** – estimates slack
  - **QoS Controller** – decides, which BE tasks need to be revoked

Resource Estimator Pipeline



Serenity Quality of Service

- We look at HW performance counters of production tasks to identify Noisy Neighbour situation
- QoS Controller revokes BE tasks until HW counters returns back to previous values
- To make enviroment more stable during resource contention, the QoS controller sends StopOversubscription message to RE Valve filter

Serenity & Mesos Future

IN A WORLD OF MAGNETS AND MIRACLES
THERE'S A HUNGER STILL UNSATISFIED

Next steps for Serenity

- Make QoS Algorithms more sophisticated
- Expose Noisy Neighbour situations as a hint for schedulers
  - Cluster-level Serenity?
- Pipelines drawn & configured in simple config file
- Integrate with Application Performance Metrics

Mesos Environment

- Enable oversubscription features in frameworks
- Enable CPU Set isolator
- Enable Cache Partitioning isolator

What's left to answer in Mesos?

- How to fully isolate of BE tasks and latency critical tasks on CPU level?
- What does it mean, when BE tasks has „4 cpus"?
- How to signal framework that performance of tasks is affected?
- What to do with BE jobs, when PR job finishes it's work?

Application Performance Metrics

THE NEXT BIG THING

Application Performance Metrics

- Let frameworks report their Service Level Indicators (SLIs) and Service Level Objectives (SLOs)
- Report global and local cluster performance
- Support in identifying noisy neighbour situation
- Still in design exploration
- Design docs: http://bit.ly/MesosAPM

https://github.com/mesosphere/serenity