Scaling a Highly-Available Scheduler Using the Mesos Replicated Log

Kevin Sweeney (@kts)
Twitter

Apache Aurora

- Long-running services
- Cron jobs
- Ad-hoc jobs

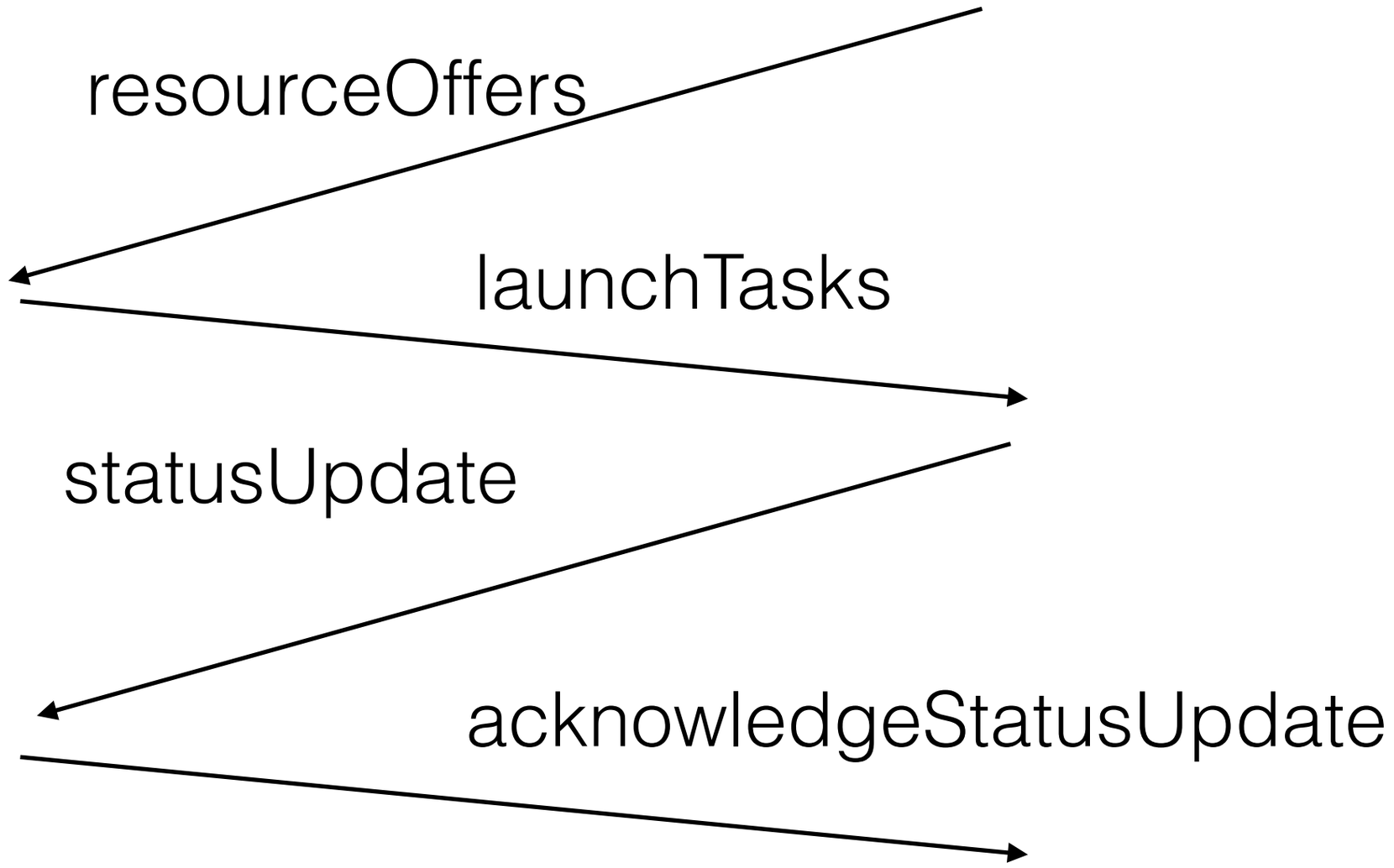
resourceOffers

launchTasks

statusUpdate

acknowledgeStatusUpdate

Any Scheduler



resourceOffers

launchTasks

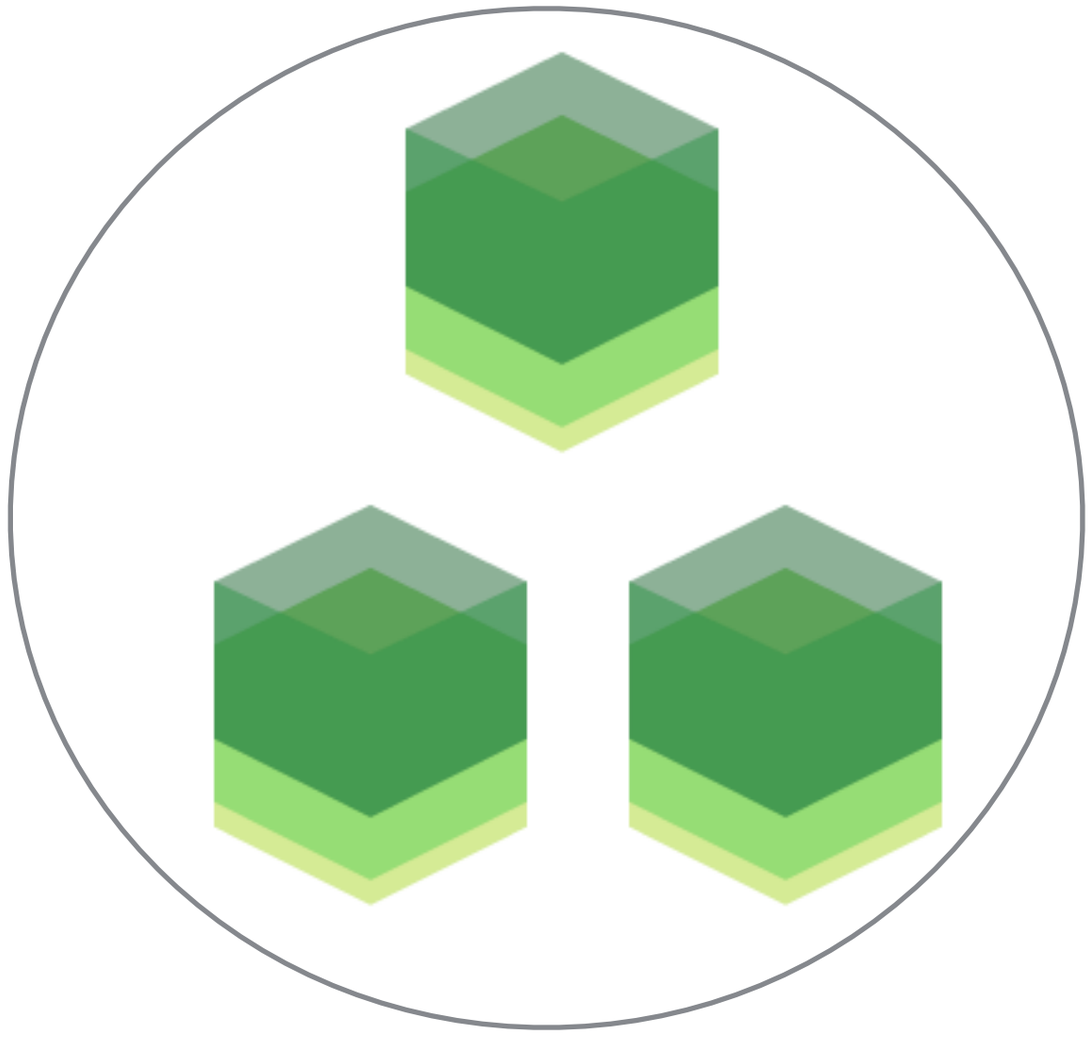
statusUpdate

acknowledgeStatusUpdate


LEVELDB

+


Apache
MESOS
™



Mesos Log API

- `Writer`
  - `Position append(byte[] data);`
  - `Position truncate(Position to);`
- `Reader`
  - `List<byte[]> read(`
    `Position from,`
    `Position to);`

Apache Thrift ™

- Interface Definition Language

- Field Numbers, not Field Names

Apache Thrift ™

```
struct Task {
  1: string id
}
```

Apache Thrift ™

```
struct TaskID {
    1: string clusterName
    2: string id
}
```

Apache Thrift ™

```
struct Task {
    1: string deprecatedId
    2: TaskID id
}
```

Apache Thrift ™

```
struct Task {
 2: TaskID id
}
```

Mesos Log API

- `Writer`
  - `Position append(byte[] data);`
  - `Position truncate(Position to);`
- `Reader`
  - `List<byte[]> read(`
    `Position from,`
    `Position to);`

byte[]s?

```
union LogEntry {

   1: Snapshot snapshot

   2: Transaction transaction

   3: bool noop

}
```

byte[]s?

```
union LogEntry {

  1: Snapshot snapshot

  2: Transaction transaction

  3: bool noop

}
```

byte[]s?

```
struct Snapshot {

   // …

   4: set<api.ScheduledTask> tasks

   // …

}
```

How Big?

```
% du -sm scheduler-backup-2015-08-17-05-33

1546        scheduler-backup-2015-08-17-05-33
```

byte[]s?

```
union LogEntry {

   // …

   5: binary deflatedEntry

   // …

}
```

Solution: Compression

```
% gzip -3c scheduler-backup-2015-08-17-05-33 |\
    wc -c

296351535

# 296 MiB
```

Solution: Normalization

```
union LogEntry {

  // …

  6: DeduplicatedSnapshot deduplicated

  // …

}
```

Solution: Normalization

```
struct Snapshot {

  // …

  4: set<api.ScheduledTask> tasks

  // …

}
```

Solution: Normalization

```
struct Snapshot {

  // …

  4: set<api.ScheduledTask> tasks

  // …

}
```

Solution: Normalization

```
struct ScheduledTask {

  /** The task that was scheduled. */

  1: AssignedTask assignedTask

  /** The current status of this task. */

  2: ScheduleStatus status

}
```

Solution: Normalization

```
struct AssignedTask {

  /** Globally-unique Mesos task ID. */
    1: string taskId

    4: TaskConfig task

}
```

Solution: Normalization

```
struct DeduplicatedSnapshot {
   2: list<DeduplicatedTask> partialTasks

   3: list<api.TaskConfig> taskConfigs

}
```

Solution: Normalization

```
% snapshot-tool deduplicate \

    < scheduler-backup-2015-08-17-05-33 \

    | wc -c

295598543

# 281 MiB
```

Solution: Both

```
% snapshot-tool deduplicate \

  < scheduler-backup-2015-08-17-05-33 \

  | gzip -3c - \

  | wc -c

    47030119

# 44 MiB
```

resourceOffers

launchTasks

statusUpdate

acknowledgeStatusUpdate

Conclusions: Performance

- A performant scheduler needs a performant database…

Conclusions: Performance

- …so you might as well **start** with a performant database

Conclusions: Normalization

- Normalization is really important

Conclusions: Normalization

```
CREATE TABLE tasks(

  id IDENTITY,

  task_id VARCHAR NOT NULL,

  task_config_row_id INT NOT NULL

      REFERENCES task_configs(id),

  UNIQUE(task_id)

);
```

Conclusions: Normalization

- SQL is a pretty good tool for expressing normalization

Questions?

Get Involved

- @ApacheAurora on Twitter
- #aurora on irc.freenode.net
- aurora.apache.org