Cloud computing gives the *illusion* of $\infty$ (virtual) resources.

Actually there is a finite amount of (physical) resources.

We would like to efficiently share those resources:

1. being able to distinguish high priority (serving customer *now*) from low priority (batch) requests;
2. schedule accordingly.

Therefore, we should be able to *plan ahead* computations.
Dynamic Scheduling: use work queues, priorities, but limited.

Without knowledge of jobs, this is the best you can do.

We need to ask the user for:
- what kind of resources his job require;
- a deadline/priority for his job.

In exchange we can give him an expected completion time.

We can also offer choice. (time is money.)
Flextic Overview

- Job Parser
- Program
- Execution Plan
- Cloud Representation
- Task finish updates
- Job Scheduler
- Schedules
- User chosen schedule
- User Interface
- User Choice
- Job Execution Platform

Damien Zufferey

Static Scheduling in Clouds

HotCloud’11
Giving incentive to plan in advance

The scheduler returns not one but many possible schedules with different finish times.
Use a pricing model to associate a cost to the schedules.
Include the “scheduling difficulty” in the cost, give a discount to schedule with later finish time.

Problem: static scheduling is hard.
Only possible if the scheduler can handle the work load.
A Job is a directed acyclic task (DAG) of tasks.
- Node are marked with worst case duration.
- Edges are marked with data transfer.
- Duration and data can be parametric in the input.
Parametric Jobs

User Job

Schema

Connections

Mappers

Reducers

Job Parser

Execution Plan

Task Details

Object Sizes

Database

Input Data Size

Damien Zufferey Static Scheduling in Clouds
HotCloud’11 7 / 13
Datacenter as a tree-like graph:
- internal nodes are router;
- leaves are compute nodes (computation speed);
- edges specifies the bandwidth.
Assumption: job and infrastructure **regularity**

Idea: regularity makes large scale scheduling feasible

How: Using abstraction techniques
Abstraction for jobs:

Group independent tasks as per a topological sort. Merge them into an abstract task.
Abstraction for jobs:

Group independent tasks as per a topological sort. Merge them into an abstract task.
Abstraction for jobs:

Group independent tasks as per a topological sort. Merge them into an abstract task.
Abstraction for infrastructure:

Merge nodes to according to network topology:

Concrete System

Medium abstraction

Coarsest abstraction
Experiments: compared to Hadoop

Caution: static scheduling alone will not work.

- Task duration are conservative estimates;
- Variability of the performance of the compute node.

We use static scheduling with backfilling.

- The jobs are MapReduce jobs doing image transformation.
- Hadoop streaming version 0.19.0
Questions ?