Nephele
Efficient Parallel Data Processing in the Cloud

Daniel Warneke and Odej Kao
daniel.warneke@tu-berlin.de

Complex and Distributed IT-Systems
Technische Universität Berlin
Cloud Computing provides IT infrastructure on demand
- Infrastructure-as-a-Service (IaaS)
- Different types of virtual machines
- No long-term obligations, pay as you go

Data processing as major cloud application
- Flexible and easy deployment
- No need for own compute center
- Amazon integrated Hadoop as core service
Compute Clouds and Data Processing (2/2)

- Current situation:
  - Allocate set of virtual machines
  - Deploy processing framework
  - Submit processing job
  - Destroy virtual machines

- Imitation of static clusters
  - Cloud’s features remain unused
  - Poor resource utilization → higher processing cost!
Outline

- Data Processing and Compute Clouds ✓
- Opportunities
- Nephele
  - Architecture
  - Job Description
  - Scheduling
- Evaluation
- Conclusion
Opportunities

- Embrace dynamics and heterogeneity offered by clouds!
- Enables new ways of job scheduling
- VMs can be allocated/deallocated according to job progress, or to respond to peaks in workload

Requirements:
- Processing framework must be aware of the cloud
- Job description/schedule must be able to express when to allocate/deallocate virtual machines
Nephele

- Data processing framework for compute clouds
  - Runs on clouds following IaaS abstraction
  - Allocates/deallocates VMs on behalf of user according to job progress

- Job description based on directed acyclic graphs (DAGs)
  - Vertices represent job’s individual tasks
  - Edges denote communication channels
Nephele Architecture

- Classic master worker pattern
- Workers are allocated on demand

![Diagram showing the Nephele Architecture with a timeline of workload over time]
Job Description

- Job Graphs focus on simplicity
  - No explicit modeling of parallelization
  - No explicit assignment to VMs
  - No explicit assignment of channel types

- Users can provide annotations to influence construction of schedule

```
<table>
<thead>
<tr>
<th>Output 1</th>
<th>Task: LineWriterTask.program</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Output: s3://user:key@storage/output</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Task 1</th>
<th>Task: MyTask.program</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Input 1</th>
<th>Task: LineReaderTask.program</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Input: s3://user:key@storage/input</td>
</tr>
</tbody>
</table>
```
Execution Graph

- Primary scheduling data structure
- Explicit parallelization
  - Tasks can be declared „parallelizable“
  - Tasks specify „wiring“ of subtasks
- Explicit assignment to virtual machines
  - Specified by ID and type
  - Type refers to hardware profile
Dealing with On-Demand VM Allocation

- Issues with on-demand allocation:
  - When to allocate virtual machines?
  - When to deallocate virtual machines?
  - No guarantee of resource availability!

- Stages ensure three properties:
  - VMs of upcoming stage are available
  - All workers are set up and ready
  - Data of previous stages is stored in persistent manner
Channel Types

- **Network channels (pipeline)**
  - Vertices must be in *same* stage

- **In-memory channels (pipeline)**
  - Vertices must run on same VM
  - Vertices must be in *same* stage

- **File channels**
  - Vertices must run on same VM
  - Vertices must be in *different* stages
Evaluation Job

- Given 100 GB of integer numbers, 100 bytes each
  - Find the smallest 20% of these numbers
  - Calculate the average of these 20%

- Implemented job for both Nephele and Hadoop
  - Popular open-source data processing framework
  - Highly „advertized“ as major cloud application

- Hardware platform:
  - Private cloud based on Eucalyptus (EC2 WS API)
  - Commodity servers with Intel Xeon 2.66 GHz (8 cores)
Results Hadoop

- 1. Task: Terasort
  - Map (a-c), Reduce (b-d)

- 2. Task: Aggregation 1
  - Map (d-f), Reduce (e-g)

- 3. Task: Aggregation 2
  - Map, Reduce (g-h)

Used cloud resources:
6 x c1.xlarge (8 CPU cores, 18 GB RAM)
Results Nephele

- (a) Experiment starts
- (b) Sorting starts
- (c) Merging starts
- (d) Deallocation of six c1.xlarge VMs
- (e) Experiment ends

Used cloud resources:
6 x c1.xlarge (8 CPU cores, 18 GB RAM)
1 x m1.small (1 CPU core, 1 GB RAM)

Transfer penalty for changing VMs
Conclusion

● Nephele facilitates new ways of data processing in clouds

● Variety of research issues to address in the future:
  ■ Dynamic compression to mitigate transfer penalty
  ■ Feedback-based construction of Execution Graphs
  ■ More sophisticated ways to improve fault tolerance

● Nephele will become open source soon!
  ■ Feel free to contact me: daniel.warneke@tu-berlin.de