Stratosphere – Data Management on the Cloud

Odej Kao
Complex and Distributed IT Systems
Computer Science and Electrical Engineering
Technische Universität Berlin

This presentation is a joint work with Volker Markl, Andreas Kliem, Björn Lohrmann and Daniel Warneke

Stratosphere
Above the Clouds

Fraunhofer FIRST
TU Berlin
Explore the power of Cloud computing for complex information fusion

Database-inspired approach
Analyze, aggregate, and query
Textual and (semi-) structured data
Research and prototype a web-scale data analytics infrastructure
**Current Research Landscape**

- Large scale data management is an area of vivid research
  - Google, Yahoo!, Microsoft, Facebook, IBM, UC Berkeley, UC Irvine, etc.

<table>
<thead>
<tr>
<th>Higher-Level Language</th>
<th>Pig, Jaql, Hive</th>
<th>Scope, DryadLINQ</th>
<th>AQL, Pig, Hive</th>
<th>SIMPLE/Sopremo</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parallel Programming Model</td>
<td>Map/Reduce</td>
<td>Hadoop</td>
<td>Algebricks</td>
<td>PACT</td>
</tr>
<tr>
<td>Execution Engine</td>
<td>Dryad</td>
<td>Dryad Stack (Microsoft)</td>
<td>Hyracks</td>
<td>Stratosphere Stack (TU, HU, HPI)</td>
</tr>
<tr>
<td></td>
<td>Hadoop</td>
<td>Hadoop Stack (Yahoo!, Facebook)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Odej Kao – Stratosphere – Data Management on the Cloud
Outline

- Overview Stratosphere
- Massive-parallel execution with Nephele
- Topology detection and streaming
- Conclusions
Stratosphere in a Nutshell

- **PACT Programming Model**
  - Declarative definition of data parallelism
  - Centered around second-order functions
  \[\Rightarrow\text{Generalization of map/reduce}\]

- **Nephele**
  - Executes schedules compiled from PACTs
  - Exploits scalability/flexibility of clouds
  - Fault tolerance mechanisms
  - Designed to run on top of IaaS
  - Heterogeneity through different VM types
Architecture: Nephele Layer

- Key Concepts
  - Massively parallel, fault-tolerant engine
Architecture: PACT Layer

- Key Concepts
  - Massively parallel, fault-tolerant engine
  - Declarative specification through parallelization contracts (PACTs)

Odej Kao – Stratosphere – Data Management on the Cloud
Key Concepts

- Massively parallel, fault-tolerant engine
- Declarative specification through parallelization contracts (PACTs)
- Adaptive execution
Architecture: Robustness

- Key Concepts
  - Massively parallel, fault-tolerant engine
  - Declarative specification through parallelization contracts (PACTs)
  - Adaptive execution
  - Robust Optimization

Odej Kao – Stratosphere – Data Management on the Cloud
Architecture: SOPREMO Layer

- **Key Concepts**
  - Massively parallel, fault-tolerant engine
  - Declarative specification through parallelization contracts (PACTs)
  - Adaptive execution
  - Robust Optimization
  - Semi-structured/text data model
  - Uncertainty
  - Declarative data flow programs with compute- and data intensive operations
  - Information extraction
  - Data cleansing
What is a PACT?

- Second-order function that defines properties on the input and output data of its associated first-order function

- Input Contract
  - Generates independently processable subsets of data
  - Generalization of map/reduce
  - Enforced by the system

- Output Contract
  - Describes properties of the output of the first-order function
  - Use is optional but enables certain optimizations
  - Guaranteed by the user
Map and reduce as PACTs

- Map and reduce are PACTs in our context

- **Map**
  - All pairs are independently processed

- **Reduce**
  - Pairs with identical key are grouped
  - Groups are independently processed
PACTs beyond Map and Reduce

- **Cross**
  - Cartesian product of multiple inputs is built
  - All combinations are processed independently

- **Match**
  - Multiple inputs
  - All combinations of pairs with identical key over all inputs are built and processed independently
  - Contract resembles an equi-join on the key

- **CoGroup**
  - Pairs with identical key are grouped for each of multiple input
  - Groups of all inputs with identical key are processed together
Outline

- Cloud Computing for Data Management
- Massive-parallel execution with Nephele
- Topology detection and streaming
- Conclusions
**Research Question**

“How to improve the efficiency of massively parallel data processing on Infrastructure as a Service (IaaS) platforms”

<table>
<thead>
<tr>
<th>Opportunities: Elasticity</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Scale-up/scale-down to respond to changes in the workload</td>
</tr>
<tr>
<td>- Exploit resource heterogeneity to improve cost efficiency</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Challenges: Loss of control due to required virtualization</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Shared infrastructure, loss of knowledge about I/O capacities</td>
</tr>
<tr>
<td>- Network topology between machines is unknown</td>
</tr>
</tbody>
</table>
Requirements

- **Shared resource management**
  - Abandon assumption that execution engine “owns” nodes
  - Instead nodes are temporarily “leased”

- **Job must express tasks’ data dependencies**
  - Which task’s input is required as which task’s output
  - Required to safely terminate virtual machines

- **Mapping between tasks and VM types**
  - Which task shall run on which type of virtual machine?
  - Information could be provided by programmer
Research Prototype: Nephele

- Standard master worker pattern
- Workers can be allocated on demand
Nephele Job Description

- Nephele job is represented as DAG
  - Vertices represent tasks
  - Edges denote communication channels

- Mandatory information for each vertex
  - Task program, (Input/output data location)

- Optional information for each vertex
  - Degree of parallelism
  - Degree of parallelism per node
  - Node type (#CPU cores, RAM…)
  - Channel types, …
Internal Scheduling Representation

- Explicit parallelization
  - Individual degree of parallelization for each task

- Explicit assignment to VMs

- Communication channels
  - Network channels
  - In-memory channels
  - File channels
Experimental Evaluation

- MR jobs on Hadoop
- MR jobs on Nephele
Challenges for Exploiting Elasticity

- Which degree of parallelization is suitable for which task?
  - Cloud philosophy: one core x 1000 hours = 1000 cores x one hour
  - Hard to anticipate for arbitrary user code, must be assessed online

CPU Bottlenecks

Task 1: Avg. CPU Util.: 
Input 1: Avg. CPU Util.: 
Output 1: Avg. CPU Util.: 

I/O Bottlenecks

Task 1: Avg. CPU Util.: 
Input 1: Avg. CPU Util.: 
Output 1: Avg. CPU Util.: 

Odej Kao – Stratosphere – Data Management on the Cloud
Bottleneck Detection

- Profiling component runs on every worker node

- Profiling provides
  - $pt(v_i)$: % of time parallel instance $i$ of vertex $v$ used its given CPU time during last $t$ seconds (seq. code, independence of par. instances)
  - $st(e_j)$: % of time parallel instance $j$ of edge $e$ was saturated during last $t$ seconds (capacity contr. channels)

- Values of $pt(v_i)$ and $st(e_j)$ are propagated to master every $t$ seconds
Bottleneck Detection Algorithm

\[ L_{RTS} \leftarrow ReverseTopologicalSort(G) \]

for all \( v \) in \( L_{RTS} \) do
\( v.isCpuBottleneck \leftarrow IsCpuBottleneck(v, G) \)
end for

if \( \exists v \in L_{RTS} : v.isCpuBottleneck \) then
for all \( v \) in \( L_{RTS} \) do
\( E_v = \{(v, w) \mid w \in V_G, (v, w) \in E_G\} \)
for all \( e \in E_v \) do
\( e.isIOBottleneck \leftarrow IsIOBottleneck(e, G) \)
end for
end for
end if

Criteria CPU bottleneck:
- \( pt(v) > \alpha \) (\( \alpha = 90\% \))
- No successor vertex of \( v \) is CPU bottleneck

Criteria I/O bottleneck:
- \( st(e) > \beta \) (\( \beta = 90\% \))
- No successor edge of \( e \) is I/O bottleneck

Odej Kao – Stratosphere – Data Management on the Cloud

pt(v) = 10%

pt(v) = 27%

st(e) = 16%

st(e) = 9%

pt(v) = 99%

pt(v) = 35%

pt(v) = 100%
Evaluation (1/2)

● Evaluation job
  ■ Conversion of article DB
  ■ 40 GB of bitmap images to PDF

● Properties of job
  ■ Different computational complexities of tasks
  ■ Each parallel instance runs on separate VM (with 1 CPU core)
  ■ Input data reside on external storage

● Goal of evaluation
  ■ Find ideal degree of parallelization for each task
Evaluation (2/2)

a) OCR Task (1), PDF Creator (1), Inverted Index Task (1)

b) OCR Task (4), PDF Creator (1), Inverted Index Task (1)

Duration: 1:15 h
7 VMs

Duration: 5:10 h
4 VMs

Duration: 0:25 h
22 VMs

CPU Bottleneck
I/O Bottleneck

Duration: 0:24 h
23 VMs

Odej Kao – Stratosphere – Data Management on the Cloud
Outline

- Cloud Computing for Data Management
- Massive-parallel execution with Nephele
  - Topology detection and adaptive compression
- Conclusions
Motivation

- The network is a scarce resource
  - Used for communication among nodes
  - Used by distributed file system
  - Possibly used by other virtual machines

- Network performance hard to predict
  - Available throughput may change over time
  - Can lead to I/O bottlenecks starvation

- Idea: Handle varying I/O performance on application layer
  - Adaptive compression
  - Topology detection
Adaptive Online Compression

- Selection of different compression algorithms
  - Each algorithm has different time/size ratio

- Calibration of decision model during data transfer
  - Try out different compression levels
  - Learn from previous compression decisions
  - Reward good decisions, penalize bad ones

Uncompressed Data → Adaptive Online Compression → Compressed Data → I/O Layer of OS

Decision Model

- No comp.
- Algo. X
- Algo. Y
- Algo. Z

Feedback Data

Odej Kao – Stratosphere – Data Management on the Cloud
Detecting network topology
Detecting network topology

- Rack Switch 1
- Rack Switch 2
- Rack Switch 3
- Rack Switch 4

Backbone Switch / IP Router

Servers

Odej Kao – Stratosphere – Data Management on the Cloud
Detecting network topology

Backbone Switch / IP Router

< 1 GBit/s

~ 1 GBit/s (regular Ethernet)

> 1 GBit/s (no actual bits on the wire)
Detecting network topology

- Cloud customer's perspective:
  - IP addresses to VMs only ⇒ Underlying network topology is not revealed
  - Data locality cannot be exploited inside application

- Can we infer the physical network topology from the VMs?
Topography Inference (TI) from End Nodes

- Rely on assistance of internal network nodes
  - Use ICMP, traceroute-like tools

<table>
<thead>
<tr>
<th>Benefits</th>
<th>Challenges</th>
</tr>
</thead>
<tbody>
<tr>
<td>✓ Simple</td>
<td>✗ Unable to detect switches/bridges</td>
</tr>
<tr>
<td>✓ Robust for IP-level topologies</td>
<td>✗ Anonymous routers</td>
</tr>
</tbody>
</table>

- Do not rely on assistance of internal network nodes
  - Observe network behavior from end nodes only
  - Use observations to infer existence of internal network nodes

<table>
<thead>
<tr>
<th>Benefits</th>
<th>Challenges</th>
</tr>
</thead>
<tbody>
<tr>
<td>✓ &gt; 10 years research history for WANs</td>
<td>✗ No research for data center networks</td>
</tr>
<tr>
<td>✓ Potentially identifies switches/bridges</td>
<td>✗ Impact of virtualization unknown</td>
</tr>
</tbody>
</table>
TI based on End-to-End Measurements

- One sender node, two or more receiver nodes
  - Connected through unknown, tree-like network
  - Sender sends probe packets to receivers
  - Receivers observe link characteristics like throughput, delay, packet loss

![Diagram showing network topology and loss rates](image-url)
Packet loss hard to observe due to high throughput links

Virtualization destroys packet correlation on shared link
- Poor delay correlation for KVM with unmodified device drivers
- Modest increase of interarrival times for both KVM and XEN (paravirtualization)
Link Characteristic Delay (RTT)

- RTT can be used to detect co-located VMs with paravirt.

Statistically significant gap between intra- and intra-host RTT for XEN paravirt.

High variance of RTTs for KVM full virt.

Statistically significant gap between intra- and intra-host RTT for KVM paravirt.
Inferred Tree is always Binary

- Binary trees fit measured data most closely
  - Highest degree of freedom
  - "Overfitted" version of actual network topology

![Diagram of physical and inferred routing trees](image)
Re-Rooting the Inferred Tree

- Remember: Data center networks have regular structure

- Idea:
  - Determine depth of each leaf node
  - New root minimizes difference between smallest and highest depth
Limiting Depth of Inferred Tree

- After Re-rooting, depth of the inferred tree is reduced
  - Assumption: Tree depth greater than \( d \) is unlikely to occur in data center

- Idea:
  - Until tree depth \( \leq d \), identify leaf node with highest depth
  - Merge parent and parent’s parent

Tree depth: 6
Limiting Depth of Inferred Tree

- After Re-rooting, depth of the inferred tree is reduced
  - Assumption: Tree depth greater than $d$ is unlikely to occur in data center

- Idea:
  - Until tree depth $\leq d$, identify leaf node with highest depth
  - Merge parent and parent’s parent

Tree depth: 5

Odej Kao – Stratosphere – Data Management on the Cloud
Limiting Depth of Inferred Tree

- After Re-rooting, depth of the inferred tree is reduced
  - Assumption: Tree depth greater than $d$ is unlikely to occur in data center

- Idea:
  - Until tree depth $\leq d$, identify leaf node with highest depth
  - Merge parent and parent’s parent

Tree depth: 4
Limiting Depth of Inferred Tree

- After Re-rooting, depth of the inferred tree is reduced
  - Assumption: Tree depth greater than $d$ is unlikely to occur in data center
- Idea:
  - Until tree depth $\leq d$, identify leaf node with highest depth
  - Merge parent and parent’s parent

Robinson-Foulds Distance: 1.5

Physical routing tree

Inferred logic routing tree
Current Work: Streaming

- Nephele and PACTs currently focus on batch-job workloads
  - Usual goal: „minimize time-to-solution“
  - Translates to „maximize throughput“

- What about streaming workloads?
  - Possible with Nephele, but (as of now) not PACTs
  - May have different goals
    - Meet pipeline latency and throughput requirements
    - Minimize pipeline latency, don’t care about throughput
    - Max/Min other custom metrics
Conclusion

- Parallel data processing on clouds is promising research area
  - Elasticity/cost model provides new use cases

- Future work
  - Streaming and profile comparisons
  - CloudNets – move part of the computation into the networks

- Plenty of opportunities for future work
  - Currently 20+ developers, Apache License
  - Check www.stratosphere.eu for downloads, tutorials
Thank you