Distributed Data Management
Summer Semester 2013
TU Kaiserslautern

Dr.-Ing. Sebastian Michel

smichel@mmci.uni-saarland.de
Lecture 1

MOTIVATION AND OVERVIEW
Distributed Data Management

• What does “distributed” mean?

• And why would we want/need to do things in a distributed way?
Reason: Federated Data

- Data is per se hosted at different sites
- Autonomy of sites
- Maintained by diff. organizations
- Mashups over such independent sources
- Linked Open Data (LOD)
Reason: Sensor Data

- Data originates at different sensors
- Spread across the world
- Health data from mobile devices

Continuous queries!
Reason: Network Monitoring

E.g. find clients that cause high network traffic.

<table>
<thead>
<tr>
<th>IP</th>
<th>Bytes in kB</th>
</tr>
</thead>
<tbody>
<tr>
<td>192.168.1.7</td>
<td>31kB</td>
</tr>
<tr>
<td>192.168.1.3</td>
<td>23kB</td>
</tr>
<tr>
<td>192.168.1.4</td>
<td>12kB</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>IP</th>
<th>Bytes in kB</th>
</tr>
</thead>
<tbody>
<tr>
<td>192.168.1.8</td>
<td>81kB</td>
</tr>
<tr>
<td>192.168.1.3</td>
<td>33kB</td>
</tr>
<tr>
<td>192.168.1.1</td>
<td>12kB</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>IP</th>
<th>Bytes in kB</th>
</tr>
</thead>
<tbody>
<tr>
<td>192.168.1.4</td>
<td>53kB</td>
</tr>
<tr>
<td>192.168.1.3</td>
<td>21kB</td>
</tr>
<tr>
<td>192.168.1.1</td>
<td>9kB</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>IP</th>
<th>Bytes in kB</th>
</tr>
</thead>
<tbody>
<tr>
<td>192.168.1.1</td>
<td>29kB</td>
</tr>
<tr>
<td>192.168.1.4</td>
<td>28kB</td>
</tr>
<tr>
<td>192.168.1.5</td>
<td>12kB</td>
</tr>
</tbody>
</table>
Reason: Individuals as Providers/Consumers

- Don’t want single operator with global knowledge -> better decentralized?
- Distributed search engines
- Data on mobile phones
- Peer-to-Peer (P2P) systems
- Distributed social networks
- Leveraging idle resources
Example: SETI@Home

• Distributed Computing
• Donate idle time of your personal computer
• Analyze extraterrestrial radio signals when screensaver is running
Example: P2P Systems: Napster

- Developed in 1998.
- First P2P file-sharing system

- Central server (index)
- Client software sends information about users' contents to server.
- User sends queries to server
- Server responds with IP of users that store matching files.

→ Peer-to-Peer file sharing!
Example: Self Organization & Message Flooding
Example: Structured Overlay Networks

• Logarithmic cost with routing tables (not shown here)
• Self organizing

• Will see later twice: NoSQL Key-Value stores and P2P Systems
Reason: Size
Showcase Scenario

• Assume you got 10 TB data on disk
• Now, do some analysis of it

• With a 100MB/s disk, reading alone takes
  – 100000 seconds
  – 1666 minutes
  – 27 hours
Huge Amounts of Data

• Google:
  – Billions of Websites
    (around 50 billion, Spring 2013)
  – TBs of data

• Twitter:
  – 100s million tweets per day

• Cern’s LHC
  – 25 Petabytes of data per year
Huge Amounts of Data (2)

• Megaupload
  – 28 PB of data

• AT&T (US Telecomm. Provider)
  – 30 PB of data through its networks each day

• Facebook
  – 100 PB Hadoop cluster

http://en.wikipedia.org/wiki/Petabyte
Need to do something about it

http://www.google.com/about/datacenter

http://flickr.com/photos/jurvetson/157722937/
Scale-Out vs. Scale-Up

• Scale-Out (Many Servers-> Distributed)

• As opposed to Scale-Up
Scale-Out

• Common technique is scale-out
  – Many machines
  – Amazon’s EC2 cloud, around 400,000 machines

• Commodity machines (many but not individually super fast)

• Failures happen virtually at any time.

• Electricity is an issue (particularly for cooling)

http://huanliu.wordpress.com/2012/03/13/amazon-data-center-size/
Hardware Failures

• Lots of machines (commodity hardware) → failure is not exception but very common

• $P[\text{machine fails today}] = 1/365$
• $n$ machines: $P[\text{failure of at least 1 machine}] = 1-(1-P[\text{machine fails today}])^n$

  – for $n=1$: 0.0027
  – for $n=10$: 0.02706
  – for $n=100$: 0.239
  – for $n=1000$: 0.9356
  – for $n=10000$: ~ 1.0
Failure Handling & Recovery

• **Hardware** failures happen virtually at any time
• **Algorithms/Infrastructures** have to **compensate** that

• **Replication** of data, **logging** of state, also **redundancy** in task execution
Cost Numbers (=>Complex Cost Model)

- L1 cache reference 0.5 ns
- L2 cache reference 7 ns
- Main memory reference 100 ns
- Compress 1K bytes with Zippy 10,000 ns
- Send 2K bytes over 1 Gbps network 20,000 ns
- Read 1 MB sequentially from memory 250,000 ns
- Round trip within same datacenter 500,000 ns
- Disk seek 10,000,000 ns
- Read 1 MB sequentially from network 10,000,000 ns
- Read 1 MB sequentially from disk 30,000,000 ns
- Send packet CA->Netherlands->CA 150,000,000 ns

1 ns = 10^-6 ms

Numbers source: Jeff Dean
Map Reduce

• “Novel” computing paradigm introduced by Google in 2004.

• Have many machines in a data center.
• Don’t want to care about impl. details like data placement, failure handling, cost models.

• Abstract computation to two basic functions:

• Think “functional programming” with map and fold (reduce), but
  – Distributed and
  – Large scale

Jeffrey Dean, Sanjay Ghemawat: MapReduce: Simplified Data Processing on Large Clusters. OSDI 2004: 137-150
Map Reduce: Example Map + Count

• Line 1
  – “One ring to rule them all, one ring to find them,

• Line 2
  – “One ring to bring them all and in the darkness bind them.”
Map Line to Terms and Counts

Line 1
{"one"=>["1", "1"],
"ring"=>["1", "1"],
"to"=>["1", "1"],
"rule"=>["1"],
"them"=>["1", "1"],
"all"=>["1"],
"find"=>["1"]}

Line 2
{"one"=>["1"],
"ring"=>["1"],
"to"=>["1"],
"bring"=>["1"],
"them"=>["1", "1"],
"all"=>["1"],
"and"=>["1"],
"in"=>["1"],
"the"=>["1"],
"darkness"=>["1"],
"bind"=>["1"]}
Group by Term

{{"one"=>["1", "1"],
  "ring"=>["1", "1"],
  
  ....
},

{{"one"=>["1"],
  "ring"=>["1"],
  
  ...
},

{{"one"=>[["1","1"],["1"],
  "ring"=>[["1","1"],["1"],
  
  ...
}
Sum Up

```json
{"one"=>[['1','1'], ['1'],
  "ring"=>[['1','1'], ['1'],
...

{"one"=>['3'],
  "ring"=>['3'],
...```

Distributed Data Management, SoSe 2013, S. Michel
Application: Computing PageRank

- Link analysis model proposed by Brin&Page
- Compute authority scores
- In terms of:
  - incoming links (weights) from other pages
- “Random surfer model”

New Requirements

- Map Reduce is one prominent example that novel businesses have new requirements.

- Going away from traditional RDBMS.

- Addressing huge data volumes, processed in multiple, distributed (wide spread) data centers.
New Requirements (Cont’d)

• Massive amounts of unstructured (text) data
• Processed often in batches (with MapReduce).

• Huge graphs like Facebook’s friendship graph
• Often enough to store (key, value) pairs

• No need for RDBMS overhead
• Often wanted: open source or at least not bound to particular commercial product (vendor).
Wish List

• Data should always be consistent

• Provided service should be always quickly responding to requests

• Data can be (is) distributed across many machines (partitions)

• Even if some machines fail, the system should be up and running
Theorem (Brewer's Theorem)

- System cannot provide all 3 properties at the same time:
  - Consistency
  - Availability
  - Partition Tolerance

With Huge Data Sets ....

- Partition tolerance is strictly required

- That leaves trading off consistency and availability
Best effort: BASE

- Basically Available
- Soft State
- Eventual Consistency

see http://www.allthingsdistributed.com/2007/12/eventually_consistent.html

W. Vogels. Eventually Consistent. ACM Queue vol. 6, no. 6, December 2008.
The NoSQL “Movement”

• No one-size-fits-all
• Not only SQL (not necessarily “no” SQL at all)
• for group of non-traditional DBMS (not relational, often no SQL), for different purposes
  – key value stores
  – graph databases
  – document stores
Example: Key Value Stores

• Like Apache Cassandra, Amazon’s Dynamo, Riak
• Handling of (K,V) pairs

• Consistent hashing of values to nodes based on their keys

• Simple **CRUD** operations (create, read, update, delete) (no SQL, or at least not full)
Criticisms

• Some DB folks say “Map Reduce is a major step backward”.

• And NoSQL is too basic and will end up re-inventing DB standards (once they need it).

• Will ask in a few weeks: What do you think?
Cloud Computing

• On demand hardware
  – rent your computing machinery
  – virtualization

• Google App engine, Amazon AWS, Microsoft Azure
  – Infrastructure as a Service (IaaS)
  – Platform as a Service (PaaS)
  – Software as a Service (SaaS)
Cloud Computing (Cont’d)

• Promises “no” startup cost for own business in terms of hardware you need to buy
• Scalability: Just rent more when you need them
• And return them when there is no demand
• Prominent showcase: Animoto, in Amazon’s EC2. From 50 to 3,500 machines in few days.

• But also problematic:
  – fully dependent on a vendors hardware/service
  – sensitive data (all your data) is with vendor, maybe stored in a diff country (likely)
Dynamic Big Data

• Scalable, continuous processing of massive data streams
• Twitter’s Storm, Yahoo! (now Apache) S4

http://storm-project.net/
Last but not least: Fallacies of Distributed Computing

1. The network is reliable
2. Latency is zero
3. Bandwidth is infinite
4. The network is secure
5. Topology doesn't change
6. There is one administrator
7. Transport cost is zero
8. The network is homogeneous

source: Peter Deutsch and others at Sun
LECTURE: CONTENT & REGULATIONS
What you will learn in this Lecture

• Most of the lecture is on processing big data
  – Map Reduce, NoSQL, Cloud computing
• Will operate on state of the art research results and tools
• Middle way between pure systems/tools discussion and learning how to build algorithms on top of them (see Joins over MR, n-grams, etc.)
• But also basic (important) techniques, like consistent hashing, PageRank, Bloom filters
• Very relevant stuff. Think “CV” ;)

Distributed Data Management, SoSe 2013, S. Michel
• We will critically discuss techniques (philosophies).
Prerequisites

• Successfully attended information systems or database lectures.

• Practical exercises require solid Java skills

• Work with systems/tools requires will to dive into APIs and installation procedures
**Schedule of Lectures (Topics Tentative)**

- **VL 1** (18. April): Motivation, Regulations, Big Data
- **VL 2** (25. April): Map Reduce 1
- **VL 3** (02. May): Map Reduce 2
- No Lecture (09. Mai) (Himmelfahrt, Ascension)
- **VL 4** (16. May): NoSQL 1
- **VL 5** (23. May): NoSQL 2
- No Lecture (30. Mai) (Fronleichnam, Corpus Christi)
- **VL 7** (06. June): Cloud Computing
- **VL 8** (13. June): Stream Processing
- **VL 9** (20. June): Distributed RDBMS 1
- **VL 10** (27. June): Distributed RDBMS 2
- **VL 11** (04. July): Peer to Peer Systems
- **VL 12** (11. July): Open Topic 1
- **VL 13** (18. July): Last Lecture / Oral exams
Lecturer and TA

• Lecturer: Sebastian Michel (Uni Saarland)
  – smichel (at) mmci.uni-saarland.de
  – Building E 1.7, Room 309 (Uni Saarland)
  – Phone: 0681 302 70803
  – or better, catch me after lecture!

• TA: Johannes Schildgen
  – schildgen (at) cs.uni-kl.de
  – Room: 36/340
Organization & Regulations

• **Lecture:**
  – Thursday
  – 11:45 - 13:15
  – Room 48-379

• **Exercise:**
  – Tuesday (bi-weekly)
  – 15:30 - 17:00
  – Room 52-203
  – First session: May 7\textsuperscript{th}.
Lecture Organization

• **New Lecture** (almost all slides are new).

• **On topics** that are often **brand new**.

• **Later topics** are still **tentative**.

• Please provide **feedback**. E.g., too slow / too fast? Important topics you want to be addressed?
Exercises

• Assignment sheet, every two weeks
• **Sheet + TA session by Johannes Schildgen**
• Mixture of:
  – Practical: Implementation (e.g., Map Reduce)
  – Practical: Algorithms on “paper”
  – Theory: Where appropriate (show that …)
  – Brief Essay: Explain the difference of x and y (short summary)
• Active participation wanted! 😊
Exam

• **Oral Exam** at the end of semester/early in semester break.

• Around 20min

• Topics captured announced few (1-2) weeks before exams

• We assume you actively participated in the exercises.
Registration

• Please register by email to
  – Sebastian Michel and Johannes Schildgen
  – Use subject prefix: [ddm13]
  – With content:
    • Your name
    • Matriculation number

• In particular to receive announcements/news