Designing Hybrid Data Processing Systems for Heterogeneous Servers

Peter Pietzuch
Large-Scale Distributed Systems (LSDS) Group
Imperial College London
http://lsds.doc.ic.ac.uk
<prp@imperial.ac.uk>
Data is the New Oil

Many new **sources of data** become available
- Most data is produced continuously

Data powers plethora of **new and personalised services**...
Data-Intensive Systems

Data analytics over web click streams
- How to maximise user experience with relevant content?
- How to analyse “click paths” to trace most common user routes?

Machine learning models for online prediction
- E.g. serving adverts on search engines

Solution: AdPredictor
- Bayesian learning algorithm ranks adverts according to click probabilities
Throughout and Result Freshness Matter

**High-throughput processing**

- Facebook Insights: Aggregates 9 GB/s, < 10 sec latency
- Feedzai: 40K credit card transactions/s, < 25 ms latency
- Google Zeitgeist: 40K user queries/s, < 1 ms latency
- NovaSparks: 150M trade options/s, < 1 ms latency

**Low-latency results**

Data-intensive system
Design Space for Data-Intensive Systems

Tension between performance and algorithmic complexity

Data amount

Result latency

Easy for most algorithms

Hard for all algorithms

Hard for machine learning algorithms

- Tension between performance and algorithmic complexity

- Performance and algorithmic complexity are in conflict.

- Most algorithms are easy for most data amounts.

- Some algorithms are hard for all data amounts.

- Machine learning algorithms are particularly hard for large data amounts.

- Latency ranges from 10s to 1ms.

- Data amounts range from MBs to TBs.
Algorithmic Complexity Increases

Topic-based filtering → Content-based filtering → Complex pattern matching → Stream queries → Online machine learning, data mining

Publish/Subscribe → Complex Event Processing (CEP) → Stream processing

T1
T2
T3

T1(a, b, c)
T2(c, d, e)
T3(g, i, h)
Scale Out Model in Data Centres
Task Parallelism vs. Data Parallelism

Task parallelism:
Multiple data processing jobs

Data parallelism:
Single data processing job

Input data

Servers in data centre

Results

select distinct W.cid
from Payments
(range 300 seconds)
as W,
Partition - by 1 row
as L
where W.cid = L.cid
and W.region != L.region

select distinct W.cid
from Payments
(range 300 seconds)
as W,
Partition - by 1 row
as L
where W.cid = L.cid
and W.region != L.region

select
highway, segment, direction,
AVG(speed)
from Vehicles
(range 5 seconds slide 1 second)
group by highway, segment, direction
having avg < 40
**Distributed Dataflow Systems**

Idea:
Execute **data-parallel tasks**
on cluster nodes

Tasks organised as **dataflow graph**

Almost all big data systems do this:
Apache Hadoop, Apache Spark, Apache Storm, Apache Flink, Google TensorFlow, ...
Nobody Ever Got Fired For Using Hadoop/Spark

2012 study of MapReduce workloads
- Microsoft: median job size < 14 GB
- Yahoo: median job size < 12.5 GB
- Facebook: 90% of jobs < 100 GB


Many data-intensive jobs easily fit into memory

One server cheaper/more efficient than compute cluster
Parallelism of Heterogeneous Servers

Servers have many parallel **CPU cores**
Heterogeneous servers with **GPUs** common

New types of **compute accelerators**: Xeon Phi, Google's TPUs, FPGAs, ...
Servers Are Becoming Increasingly Heterogeneous

How can Data-Intensive Systems Exploit Heterogeneous Hardware?
Roadmap

**SABER**: Hybrid stream processing engine for heterogeneous servers  
[ SIGMOD’16]  

(1) How to **parallelise computation** on modern hardware?  

(2) How to utilise **heterogeneous servers**?  

(3) Experimental **performance results**
Analytics with Window-based Stream Queries

Real-time analytics over data streams

**Windows** define **finite data amount** for processing

<table>
<thead>
<tr>
<th>highway segment direction speed</th>
<th>highway segment direction speed</th>
<th>highway segment direction speed</th>
<th>highway segment direction speed</th>
<th>highway segment direction speed</th>
<th>highway segment direction speed</th>
<th>highway segment direction speed</th>
<th>highway segment direction speed</th>
</tr>
</thead>
</table>

### Time-based window

**Time-based window** with size $\tau$ at current time $t$

$$[t - \tau : t]$$

Vehicles[Range $\tau$ seconds]

### Count-based window

**Count-based window** with size $n$:

last $n$ tuples

Vehicles[Rows $n$]
Defining Stream Query Semantics

Windows convert **data streams** to **dynamic relations** (database table)

- **Streams**
  - Window specification
  - Stream operators: Istream, Dstream, Rstream

- **Relations**
  - Any relational query
    - (select, project, join, group by, etc)
SQL Stream Queries

**SQL** provides well-defined declarative semantics for queries
- Based on relational algebra (select, project, join, …)

**Example: Identify slow moving traffic on highway**
- Input stream: Vehicles(highway, segment, direction, speed)
- Find highway segments with average speed below 40 km/h

```
select highway, segment, direction, AVG(speed) as avg
from Vehicles[range 5 sec slide 1 sec]
group by highway, segment, direction
having avg < 40
```
Perform query evaluation across **sliding windows** in parallel
- Exploit data parallelism across stream

![Diagram showing sliding windows](image)
How to use GPUs with Stream Queries?

Naive strategy parallelises computation along **window** boundaries

- **Window-based** parallelism results in **redundant** computation

---

Peter Pietzuch - Imperial College London
How to use GPUs with Stream Queries?

Parallel processing of non-overlapping window data?

Slide-based parallelism limits degree of parallelism
Apache Spark: Small Slides → Low Throughput

Spark relates window slide to micro-batch size used for parallelisation.

Avoid coupling system parameters with query definition.
Idea: Parallelise using task size that is best for hardware

Task contains one or more window fragments
SABER: Window Fragment Processing

Process window fragments in parallel

**Reassemble** partial results to obtain **overall result**

Partial result reassembly must also be done in parallel
API for Operator Implementation

**Fragment function** $f_f$
- Processes window fragments

**Assembly function** $f_a$
- Merges partial window results

**Batch function** $f_b$
- Composes fragment functions within task
- Allows incremental processing
SABER: Performance of Window-based Queries

Performance of window-based queries remains predictable

`select AVG(S.1) from S [rows 1024 slide x]`
How to Pick the Task Size?

Problem: Small data transfers over PCIe bus costly
- Example: `select * from S where p1` [rows 1 slide 1]

![Graph showing throughput vs task size for CPU and GPU processing]

- Limited by dispatcher and thread contention
- Limited by data movement

Peter Pietzuch - Imperial College London
Roadmap

**SABER**: Hybrid stream processing engine for heterogeneous servers

[SIGMOD’16]

1. How to **parallelise computation** on modern hardware?
   - Avoid coupling **system parameters** with **processing semantics**

2. How to utilise **heterogeneous servers**?
How to Utilise Heterogeneous Servers?

Hard to decide **acceleration potential** of heterogeneous processors
- Depends on operator semantics, window definition, data distribution, …

*Don’t leave decision about heterogeneous processors to users*
SABER: Hybrid Execution Model

Idea: Execute tasks on **all heterogeneous processors** (CPUs, GPUs, ...)

![Task queue diagram](image)

Fully utilise all hardware parallelism available in dedicated servers
Profile tasks to obtain cost model

Assign tasks to processor with **shortest execution time**
First-Come First-Serve Task Scheduling?

Assign tasks to processors first-come, first-serve
- CPU/GPU execute both $Q_A$ and $Q_B$ tasks

<table>
<thead>
<tr>
<th></th>
<th>CPU</th>
<th>GPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>$Q_A$</td>
<td>3 ms</td>
<td>2 ms</td>
</tr>
<tr>
<td>$Q_B$</td>
<td>3 ms</td>
<td>1 ms</td>
</tr>
</tbody>
</table>

**First-Come First-Served**

- CPU
  - $T_1$, $T_4$, $T_8$, $T_{10}$
- GPU
  - $T_2$, $T_3$, $T_5$, $T_6$, $T_7$, $T_9$

**Task queue**

**CPU workers**

**GPU worker**

- FCFS ignores effectiveness of processors for given task
Heterogeneous Lookahead Scheduling (HLS)

Idea: Scheduler assigns tasks to idle processors **dynamically**
- **Skips** tasks that could be executed faster by another processor

HLS achieves aggregate throughput of all heterogeneous processors
SABER Hybrid Stream Processing Engine

**Dispatching stage**
- Dispatch fixed-size tasks

**Scheduling & execution stage**
- Dequeue tasks based on HLS

**Result stage**
- Merge & forward partial window results

Java
- 15K LOC

C & OpenCL
- 4K LOC

Peter Pietzuch - Imperial College London
Roadmap

**SABER**: Hybrid stream processing engine for heterogeneous servers

[SIGMOD’16]

1. How to parallelise computation on modern hardware?
   - Avoid coupling **system parameters** with **processing semantics**

2. How to utilise **heterogeneous servers**?
   - **Hybrid execution** utilises all heterogeneous processors

3. Experimental **performance results**
Experimental Evaluation

- **Ubuntu Linux 14.04**
- **NVIDIA driver 346.47**
- **Intel Xeon 2.6 GHz**
- **NVIDIA Quadro K5200**
- **PCIe 3.0 x16**
- **10 Gbps NIC**
- **16 cores 64 GB RAM**
- **2,304 cores 8 GB RAM**

**Google Cluster Data**
- 144M jobs events from Google infrastructure

**SmartGrid Measurements**
- 974M plug measurements from houses

**Linear Road Benchmark**
- 11M car positions and speed on highway
What is SABER’s Performance?

 ![Graph showing throughput for different queries]

- **SABER** exploits both **CPUs** and **GPUs** effectively for different queries.
Is Hybrid Throughput Additive?

**Aggregate throughput** of CPU + GPU always **highest**
What is the Trade-Off between CPUs and GPUs?

Hybrid processing model benefits from GPU's ability to process complex predicates fast

Peter Pietzuch - Imperial College London
Does SABER Adapt to Workload Changes?

HLS periodically uses idle, non-preferred processor to run tasks to update query task throughput matrix.

Higher selectivity $\rightarrow$ more predicates evaluated $\rightarrow$ GPU preferred
Summary

Heterogeneous servers have huge impact on data-intensive systems
- Shift from scale out to scale up model
- Need new general-purpose system designs for heterogeneous servers

**SABER**: Hybrid Stream Processing Engine for CPUs & GPUs

(1) Parallelise computation to fit hardware capabilities
  - Decouple hardware/system parameters from processing semantics

(2) Fully utilise all heterogeneous processors independently of workload
  - Hybrid processing model to achieve aggregate CPU/GPU throughput
Acknowledgement:
LSDS Group at Imperial College London

We're Hiring!
Post-docs, PhDs

Thank you!
Any Questions?

Peter Pietzuch
http://lsds.doc.ic.ac.uk
<prp@imperial.ac.uk>